SIEMENS

SIMATIC

STEP 7 and WinCC Engineering V15.1

System Manual

Printout of the online help

10/2018

System overview of STEP 7 and WinCC 1

What's new in TIA Portal 2

Readme 3

Installation 4

Migrating projects and programs 5

Introduction to the TIA Portal 6

Editing projects 7

Editing project data 8

Using user administration 9

Using libraries 10

Editing devices and networks 11

Programming the PLC 12

Visualizing processes 13

Using technology functions 14

Using online and diagnostics functions 15

Processing energy data 16

Using Team Engineering 17

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 11/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

SIEMENS

Scaling of STEP 7 and WinCC in the TIA Portal 1

Options for STEP 7 and WinCC Engineering 2

Options for S7-1500 Runtime 3

Options for STEP 7 Engineering System 4

Options for WinCC Engineering and Runtime systems 5

System overview of STEP 7 and WinCC

Programming and Operating Manual

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Scaling of STEP 7 and WinCC in the TIA Portal.....5
2 Options for STEP 7 and WinCC Engineering.....7
3 Options for S7-1500 Runtime.....9
4 Options for STEP 7 Engineering System.....11
5 Options for WinCC Engineering and Runtime systems.....13

Scaling of STEP 7 and WinCC in the TIA Portal

1

Scope of performance of the products

The following graphic shows the scope of performance of the individual products of STEP 7 and WinCC:

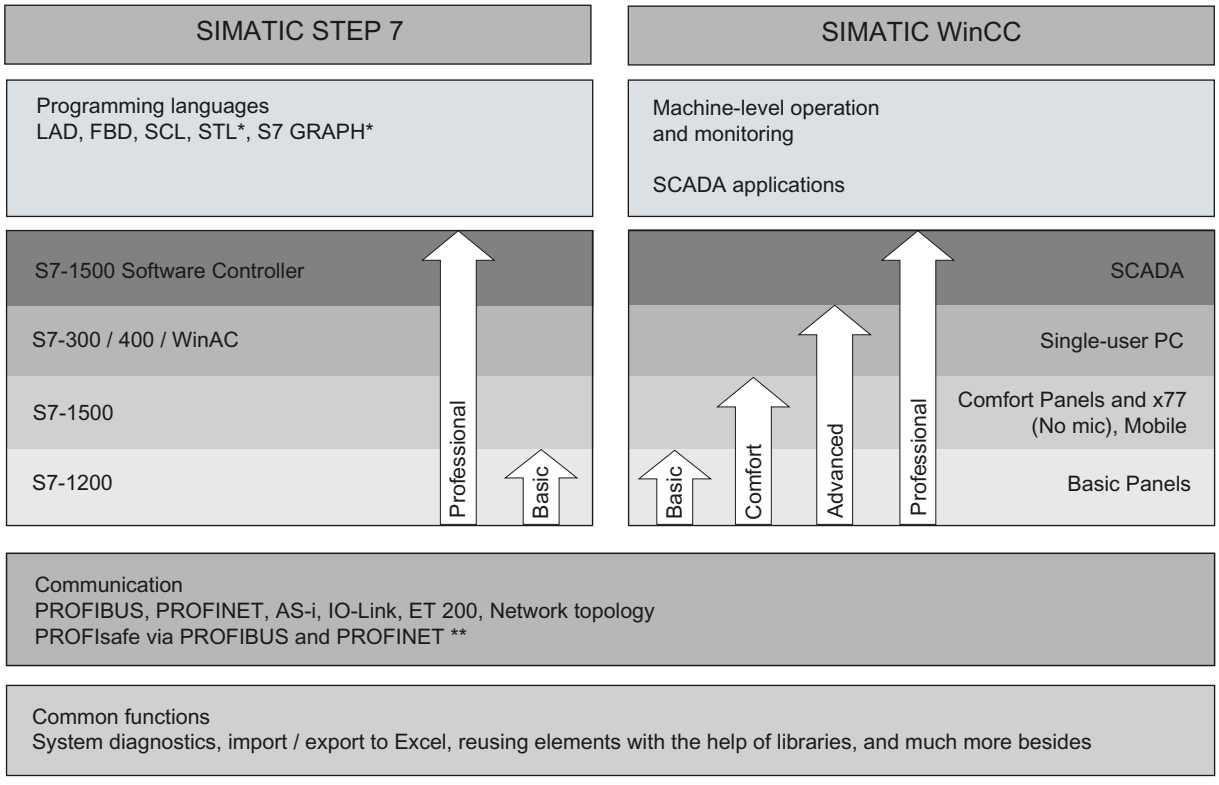

* Only for STEP 7 Professional for S7-300 / 400 / WinAC and S7-1500
** With installed optional package "STEP 7 Safety Advanced"

STEP 7

STEP 7 (TIA Portal) is the engineering software for configuring the SIMATIC controller families S7-1200, S7-1500, S7-300/400 and software controllers (WinAC). STEP 7 (TIA Portal) is available in two editions, depending on the configurable controller families:

- STEP 7 Basic for configuring the S7-1200
- STEP 7 Professional for configuring S7-1200, S7-1500, S7-300/400 and Software-Controller (WinAC)

WinCC

WinCC (TIA Portal) is an engineering software for configuring SIMATIC Panels, SIMATIC Industrial PCs, and Standard PCs with the WinCC Runtime Advanced or the SCADA System WinCC Runtime Professional visualization software.

WinCC (TIA Portal) is available in four editions, depending on the configurable operator control systems:

- WinCC Basic for configuring Basic Panels
  WinCC Basic is included with every STEP 7 Basic and STEP 7 Professional product.
- WinCC Comfort for configuring all panels (including Comfort Panels, Mobile Panels)
- WinCC Advanced for configuring all panels and PCs with the WinCC Runtime Advanced visualization software
  WinCC Runtime Advanced is a visualization software for PC-based single-station systems. WinCC Runtime Advanced can be purchased with licenses for 128, 512, 2k, 4k, 8k and 16k PowerTags (tags with a process interface).
- WinCC Professional for configuring panels and PCs with WinCC Runtime Advanced or SCADA System WinCC Runtime Professional. WinCC Professional is available in the following editions: WinCC Professional for 512 and 4096 PowerTags as well as "WinCC Professional max. PowerTags".
  WinCC Runtime Professional is a SCADA system for structuring a configuration ranging from single-station systems to multi-station systems including standard clients or web clients. WinCC Runtime Professional can be purchased with licenses for 128, 512, 2k, 4k, 8k, 64k, 100k, 150k and 256k PowerTags (tags with a process interface).

Options for STEP 7 and WinCC Engineering 2

The following cross-product options are available for STEP 7 and WinCC Engineering:

- TIA Portal Multiuser (multiple users work jointly on a TIA Portal project)
- TIA Portal Cloud Connector (access to local interface using RDP)
- TIA Portal Teamcenter Gateway (connection to Teamcenter)
- SIMATIC Visualization Architect (creating HMI contents based on STEP7 projects)
- SIMATIC Energy Suite (energy management)
- SIMATIC Safe Kinematics (Safe zones and velocity monitoring of kinematics motions)

Options for S7-1500 Runtime 3

The engineering of the following functions is integrated in the engineering packages of SIMATIC STEP 7 Professional and/or SIMATIC Energy Suite. The enabling takes place in CPU context:

- SIMATIC ProDiag (machinery and plant diagnostics for S7-1500 and SIMATIC HMI)
- SIMATIC OPC UA S7-1500 (connection of any third-party devices to the S7-1500)
- SIMATIC Energy Suite S7-1500 (energy management)

Options for STEP 7 Engineering System

4

The following options are available for STEP 7 Engineering:

- SIMATIC STEP 7 Safety Basic/Advanced (safety programs for F-CPUs)
- SIMATIC S7 PLCSIM Advanced (Virtual Controller S7-1500)
- SIMATIC Target 1500S™ for Simulink® (add-on for Simulink®)

Options for WinCC Engineering and Runtime systems 5

SIMATIC Panels as well as WinCC Runtime Advanced and WinCC Runtime Professional contain all essential functions for operator control and monitoring of machines or plants. Additional options allow you to extend the functionality in some cases to increase the range of available tasks.

Option for Basic Panel

The following add-on is available for Basic Panels:

- WinCC Sm@rtServer (remote operation)

Options for Comfort Panels, Mobile Panels

The following extension options are available for Comfort Panels and Mobile Panels:

- WinCC Audit (audit trail and electronic signature for regulated applications)
- SIMATIC ProDiag (machinery and plant diagnostics for S7-1500 and SIMATIC HMI)

Options for WinCC Runtime Advanced

The following possible extensions are available for WinCC Runtime Advanced:

- WinCC SmartServer (remote operation)
- WinCC Recipes (recipe system)
- WinCC Logging (logging of process values and alarms)
- WinCC Audit (audit trail for regulated applications)
- SIMATIC ProDiag (machinery and plant diagnostics for S7-1500 and SIMATIC HMI)

Note

In contrast to WinCC flexible 2008, functions from the WinCC flexible /Sm@rtService, WinCC flexible /Sm@rtAccess options as well as the WinCC flexible /OPC Server option are incorporated into the basic functionality.

Options for WinCC Runtime Professional

The following possible extensions are available for WinCC Runtime Professional:

- WinCC Client (standard client for structuring multi-station systems)
- WinCC Server (supplements WinCC Runtime to include server functionality)
- WinCC Recipes (recipe system, formerly WinCC / UserArchives)
- WinCC WebNavigator (Web-based operator control and monitoring)

- WinCC DataMonitor (display and evaluation of process states and historical data)
- WebUX (platform and browser-independent operation and monitoring via the Web)
- SIMATIC Information Server 2014 (Web-based and browser-independent analysis and reports of historic process data)
- SIMATIC Process Historian 2014 (plant-wide archive server for messages and process data)
- Industrial Data Bridge (configurable connections to databases and IT systems)
- Redundancy (increased availability due to redundant server)
- SIMATIC ProDiag (machinery and plant diagnostics for S7-1500 and SIMATIC HMI)

Note

In contrast to WinCC V7, functions from the WinCC /OPC-Server and WinCC / ConnectivityPack options are incorporated into the basic functionality. Likewise, the basic functionality includes the Runtime API from WinCC /ODK.

SIEMENS

What's new in TIA Portal V15.1 1

What's new in TIA Portal V15 2

What's new in TIA Portal V14 SP1 3

What's new in TIA Portal V14 4

What's new in TIA Portal

Programming and Operating Manual

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 What's new in TIA Portal V15.1 5

1.1 SIMATIC STEP 7 5

1.2 SIMATIC WinCC 8

1.3 SINAMICS Startdrive 9

1.4 Engineering options 11

1.5 Runtime options 16

2 What's new in TIA Portal V15 19

2.1 Introduction 19

2.2 SIMATIC STEP 7 20

2.3 SIMATIC WinCC 24

2.4 SINAMICS Startdrive 25

2.5 SIMOCODE ES 26

2.6 Engineering options 27

2.7 Runtime options 31

3 What's new in TIA Portal V14 SP1 33

3.1 Introduction 33

3.2 SIMATIC STEP 7 34

3.3 SIMATIC WinCC 38

3.4 SINAMICS Startdrive 39

3.5 Engineering options 40

3.6 Runtime options 43

4 What's new in TIA Portal V14 45

4.1 Introduction 45

4.2 SIMATIC STEP 7 46

4.3 SIMATIC WinCC 49

4.4 SINAMICS Startdrive 51

4.5 SIMOTION SCOUT TIA 52

4.6 SIMOCODE ES V14 53

4.7 Engineering options 54

4.8 Runtime options 56

What's new in TIA Portal V15.1

1

1.1 SIMATIC STEP 7

All important new features of STEP 7 are summarized here. You can find additional details on the various topics in the sections of the product documentation.

Software units

Software units allow for breaking down the user program into separately loadable units. This means you can load changes to different software units independently of one another to the controller, especially when these changes were made by different users.

The modular processing of the PLC program by multiple users is made a lot easier by this approach.

System functions

Keyboard shortcuts are defined in the system for many functions of TIA Portal. As of V15.1 you can also use your own keyboard shortcuts as an alternative to the system-defined keyboard shortcuts. You can find an overview of all keyboard shortcuts and the option of assigning user-defined keyboard shortcuts in the settings for TIA Portal.

Editors for programming languages

- Textual block interface in SCL
  When creating new SCL blocks, you can have the block interface displayed as table or text. With text display, you have once again access to the familiar programming environment of STEP 7 V5.x. You can enjoy the benefits of creating comment sections in the block interface, for example, or using basic copying actions with other text editors.

- Actual parameters in SCL
  For better readability of programs in SCL blocks, you can left align the actual parameters for block calls.

- Setpoints in PLC data types (UDTs)
  You can select or deselect the setpoints predefined in a user-defined PLC data type (UDT) in the instance used.

- Snapshot of the current values
  A snapshot of current values is now also retained even in case of structural changes of the data block.

- Monitoring tags
  The keyboard shortcut "Ctrl+T" for enabling/disabling the monitoring of tags is now available in all STEP 7 editors (DB editor, watch table, tag table, etc.).

- Monitoring of PLC data types (UDTs)
  You can mark an input or output parameter of a UDT data type in the program during monitoring and show the associated current values in the inspector window.

- Monitoring of block calls
  Current values for non-interconnected outputs are now also displayed when monitoring block calls.

Hardware configuration

- S7-1500 CPUs for the configuration of redundant automation systems
  The following S7-1500 CPUs are available for configuration of redundant and high availability automation systems:
  - CPU 1513R-1 PN
  - CPU 1515R-2 PN
  - CPU 1517H-3 PN

- Isochronous mode on the backplane bus
  Modules which are plugged centrally in an S7-1500 station can be operated in isochronous mode. Isochronous coupling of centrally plugged modules with distributed plugged modules is also possible.

- MRP domain configuration across project boundaries
  Redundancy managers and redundancy clients of an MRP domain can be located in different projects.

- Quickly change firmware version for multiple modules
  You can change the configured firmware version of modules in an IO device by multiple selection of a group of modules in one step.

- Fast switching between I/O tag in the program editor and affected input/output in the device view
  The shortcut menu "Go to > Device view" enables fast navigation from the programmed I/O address from the program editors to the configured input or output in the device view.

- Simplified configuration of series machine projects
  For the configuration of optional IO devices in a series machine project, the necessity of port interconnection (topology configuration) is eliminated.

S7-1500 Motion Control

Technology version V4.0 contains the following new features:

- Exchange of torque data with the drive in the technological units of the technology object:
  - Additive setpoint torque
  - Current actual torque
  - Permissible torque range
- Time requirements for the technology object "Measurement jobs" via "MC_MeasuringInput"
- Extension of the data structure of the positioning axis and synchronous axis for using the kinematic technology object
- Use of optimized data blocks (drive/encoder connection)

Technology version V4.0 for the technology CPUs (S7-1500T) contains the following additional new features:

- Kinematic technology object (S7-1500T)
- Motion specification via "MotionIn" instructions (S7-1500T)
- Direct synchronous setting with MC_CamIn V4.0 (S7-1500T)

Technology version V4.0 contains the following new or extended Motion Control instructions for these functions.

Trace

Easier handling of chart configurations:

- Settings relevant for the display can already be made during configuration. This includes signal grouping, color selection, display format and creating formulas.
- Changes made online can be applied with "Transfer of trace configuration from the device". The settings in the "Chart" tab are retained even if you make changes during the configuration and add signals, for example, or change the number of measuring points.

1.2 SIMATIC WinCC

WinCC RT Professional

- OPC UA server
  - Support of OPC UA server alarm and condition
  - WinCC alarms can be transferred to a third-party application via OPC UA servers.

- ProDiag Control
  - Display of the entire Call Interface block

- Additional functional extensions
  - Setting the WebUX / WebNavigator user rights in runtime
  - Automatic login for operator role
  - HMI Compiler provides additional information for users (e.g. number of compiled HMI objects)
  - Software control with WinCC Professional on a shared computer

Mobile Panel

- KTP Mobile V15 images and SmartServer option for F-devices
  https://support.industry.siemens.com/cs/ww/en/view/109758056 (https://support.industry.siemens.com/cs/ww/en/view/109758056)

- Option+ V2 for Comfort and Mobile Panel
  - Support of the Mobile Panel 2nd Generation
  - Code is written directly into a tag with QR reader MV320/325
  - Display of the Panel CPU load directly in runtime
  - Optimization of the certificate handling of SIMATIC Logon
  - Hiding of individual desktop icons
  - Expansion of service file for technical support
  - Switch of communication between Option+ and Panel Runtime from SOAP to OPC UA

- Mobile Panel 2nd Gen., Comfort Panel and Basic Panel 2nd Gen.
  Configuration of the alarm buffer

1.3 SINAMICS Startdrive

All drive families (G120, S120, S210)

- Extensions of the Openness interface:
  - Functional interfaces for important applications (e.g. input of motor data)
  - For G120 and S210: Generation of the TIA Portal project based on an AML-based import of EPLAN
- Startdrive Advanced V15.1: Safety Integrated acceptance test also for S210 and S120
  - Guided acceptance test wizard for all drive-based Safety Integrated Functions (Basic, Extended and Advanced)
  - Automatic and Safety Integrated function-specific creation of traces for analysis of the machine behavior
  - Generation of an acceptance report as an Excel file (xlsx format, can also be used with OpenOffice)

SINAMICS G120

- User-definable parameter lists
- Data synchronization between G120 and S7-1500 technology objects also possible offline (in the project)
- Usability improvements:
  - Overview screen form for G120 configuration
  - Simplified device identification via LED flashing in the "Connect online" dialog

Integration of SINAMICS S210 and 1FK2

- Integration of the S210 drives with 200 V and 400 V as of SINAMICS firmware V5.2
- Support of 1FK2 motors
- Usability: Easier configuration procedure compared to S120
- Data synchronization between S210 and S7-1500 technology objects online and offline (in the project)
- Trace (including recommendation of typical parameters)
- Firmware update via TIA Portal
- Auto Servo Tuning via One Button Tuning
- Safety Integrated configuration

SINAMICS S120 (CU320-based drives)

- Support of the SINAMICS firmware V5.1 SP1
- Support of 1FK2 motors

- Operation on S7-1500 Open/Software controllers
- Data synchronization between S120 and S7-1500 technology objects also offline (in the project)
- Connection to S7-1500 technology object type "Probe"
- Support of the technology data block telegram 750
- Support of Isochronous Safe Position (for support of the PLC function "Cartesian Safety")
- Parameter list with functional structure (parameter groups)
- Auto Servo Tuning via One Button Tuning

1.4 Engineering options

TIA Portal Multiuser Engineering

- Improved performance at low network bandwidths
  New network profiles have been added for improved connectivity between the engineering station and the multiuser server.

- Functional extensions for multiuser power tools (command line tools)
  Export and import functions are now available for Multiuser server projects and local sessions.

TIA Portal Multiuser Commissioning

Multiuser engineering supports users during commissioning with a guided commissioning workflow. Download to the device is automatically synchronized in commissioning mode. This enables a consistent version between the device and the multiuser server project and the associated local sessions.

PLCSIM Advanced

- Control Panel extensions
  - Double-click the Control Panel to open it as a floating window. It can be moved freely and can always be shown in the foreground, if necessary.
  - Instances that were created in the past can be created in the floating Control Panel by selecting the virtual memory card in the Explorer and moving the instances from the Explorer to the Control Panel with drag and drop.

- API extensions
  The maximum cycle time of the virtual controller can be freely defined over the API without having to change the maximum cycle time in the TIA Portal project.

SIMATIC TIA Portal Openness

- Openness DLLs of V14 SP1, V15 and V15.1 in the scope of delivery
  Because the Openness DLLs of V14 SP1, V15 and V15.1 are included in the scope of delivery, applications based on V14S P1 or V15 also run in V15.1 without any changes. To make use of the functions of V15.1, you must integrate the DLL of V15.1 and recompile the application.

- Accessing projects
  - Multiple projects in a TIA Portal instance can be opened via Openness.
  - Projects can be archived and restored via Openness.
  - For UMAC-protected projects Openness enables read access to objects. The same restrictions apply to this type of access as for a user with the "Read only" authorization.

- Save as
  Projects and global libraries can be saved via Openness under a different name.

- Online/offline comparison
  The online and offline comparison of data is possible via Openness.

- Accessing protection levels
  A protection level can be set or removed for blocks.

- Accessing fingerprints
  The fingerprint can be queried for blocks and PLC data types (UDTs).

- Accessing names
  The name can be set for blocks, data blocks and UDTs.

- Fault-tolerant import
  In addition to the strict import that can still be used, a fault-tolerant import is now available. The import is still possible even if linked user data types or called blocks do not match, for example.

- Export and import
  - Watch tables as well as force tables can be exported and imported.
  - Snapshots of the current values can be exported as XML from an offline DB. This means different snapshots can be compared with the help of the XML files.

- ET200SP
  Read and write access is possible for most attributes of the ET200SP modules.

- R/H systems
  Download to the primary PLC and the backup PLC is possible for R/H systems.

- Fail-safe PLC
  - The upload is now also possible indirectly via NAT router.
  - Recipes, archives, user files and the passwords of the protection levels are also taken into account during the upload.

SIMATIC STEP 7 Safety

- New fail-safe CPU-CPU communication "Flexible F-Link"
  A new fail-safe CPU-CPU communication, "Flexible F-Link", is available with STEP 7 Safety V15.1 for the fail-safe S7-1200 and S7-1500 CPUs. This means fail-safe data can be easily exchanged as fail-safe ARRAY between two fail-safe CPUs by means of standard communication mechanisms.

- Fail-safe transfer of large amounts of fail-safe data over standard communication blocks
  Flexible F-Link supports the fail-safe transfer of large amounts of fail-safe data by means of standard communication blocks even across network limits (Layer 3 communication). Requirement is the consistent and deterministic transfer of data.
  The data to be transferred is bundled into user-defined F-compliant PLC data types (UDTs). Any fail-safe data types (Bool, Word, INT, DINT, TIME) can be used. Up to 100 bytes per UDT are supported.

- Easy parameter assignment and automatic generation of fail-safe communication DBs
  The parameters of the communication connection are assigned in the Safety Administration Editor (SAE) whereby each communication connection is created with name, UDT, monitoring time, send or receive direction, and a communication DB is automatically generated by the system. This communication DB provides the data to be transferred in form of a coded F-ARRAY which can then be easily transferred over standard communication blocks such as TSEND or USEND.

- Fail-safe runtime group communication
  Flexible F-Link also enables easy implementation of fail-safe runtime group communication. To do so, the fail-safe data can, for example, be transferred between the F-runtime groups with the "UMOVE_BLK" instruction.

- Unique communication ID (UUID) throughout the network
  The Flexible F-Link also provides a solution when it comes to ensuring the unique identification of communication partners across network limits. A unique communication ID (Universally Unique Identifier UUID) can be generated with STEP 7 Safety which ensures that the ID is unique.

- Communication signature
  A separate communication signature that is independent of the hardware and software signature makes it easy to clearly distinguish between changes to the UUID and changes to the program or hardware.

- Pre-processing and post-processing
  For easier configuration and an easily verifiable time behavior, especially in the context of fail-safe communication, STEP 7 Safety in version V15.1 offers the possibility of placing standard blocks for pre-processing and post-processing of process values around the F-runtime groups.

- Variable communication ID (DP_DP_ID)
  As of version V15.1, STEP 7 Safety offers the possibility of assigning tags instead of constants to the communication IDs (DP_DP_IDs) required for the SENDDP and RCVDP blocks. This enables users in plants with a large number of identical iDevices (high-bay warehouses, AGVs, suspended monorails, etc.), for example, to use safety programs with the same F-collective signature. The variable communication ID thus enables easier commissioning, maintenance and project management.
  Notice: If variable communication IDs (DP_DP_ID) are used, the user must ensure that from an application viewpoint the communication IDs are unique throughout the network at any time.

- Openness
  Additional functions are provided with STEP 7 Safety V15.1 over the Openness interface
  - Reading and configuring i-parameters of the ET200SP F-IOs
  - Station Upload
  - Hardware and software comparison (offline/offline)
  - Import and export of consistent (unchanged) F-blocks

TIA Portal Teamcenter Gateway

- Multiuser Engineering
  Multiple persons can now work on a TIA Portal project simultaneously with the help of "TIA Portal Teamcenter Gateway" and "Multiuser Engineering" options.

- Opening reference projects from Teamcenter
  You can open projects from Teamcenter directly as reference projects, for example, to compare and merge different revisions.

- Saving projects under existing Teamcenter elements
  You can save TIA Portal projects and libraries from within the TIA Portal under already existing Teamcenter elements.

- Compatibility
  Teamcenter versions V11.2/11.3/11.4/11.5 are supported.

SIMATIC Energy Suite

- Energy Screens
  Now part of the scope of delivery of Energy Suite V15.1

- Reports
  New: Cost center report including rates

- SINAMICS
  The configuration of SINAMICS devices with MDD is supported (until now only GSDML has been supported)

- Usability improvements
  Various smaller usability improvements

SIMATIC Visualization Architect

- Access protection to the SiVArc rule editors by means of UMAC

- X/Y properties can be set using SiVArc expressions.

- Support of STEP 7 blocks written in SCL

- Generation on template screens

- SiVArc – Openness:
  - Copying individual rules or entire rule groups from the library
  - Starting SiVArc generation

1.5 Runtime options

SIMATIC ProDiag

- Text export/import for translations
  All texts relevant for ProDiag or S7-GRAPH can be exported for translation, for example, with one action.
  The following alarm texts can be exported:
  - User-defined text lists for supervisions
  - Display names for steps and transitions in GRAPH
  - Comments of all supervised tags
  - Comments of all supervised parameters of a block
  - Instance name and instance comment of multi-instances

  The following texts can be exported for the HMI display:
  - Tag comments in Transition and Interlock
  - Titles and comments of supervised networks or tags

- Display of the entire block (LAD, FBD) in the HMI PLC Code Viewer
  The PLC Code Viewer can be used even if static parameters of a block are being monitored (user blocks with own monitoring logic).

- Edit block supervisions in the ProDiag overview editor
  Existing supervisions in the "FB supervision definitions" tab can be edited or deleted.

- Jump into TIA Portal from WinCC Advanced
  - System-supported jump into TIA Portal in the context of a GRAPH or ProDiag alarm
  - Relevant block is opened automatically
  - Read mode or write/read mode is possible
  - Automatic status display

- Consideration of hierarchical comments in ProDiag alarms
  - Easy mapping of the technological view on the machine / plant using hierarchical comments whereby the path is generated automatically.

- Subcategories in GRAPH
  - Identical alarm structure can be configured as in ProDiag.
  - Especially relevant for control systems in which a follow-up action is derived from additional information in the alarm text.

OPC UA

For S7-1500 CPUs as of firmware V2.6 you can benefit from the following extension with a corresponding OPC UA Runtime license:

In addition to the OPC UA server, an OPC UA client is integrated in the CPU which offers the following functions over corresponding OPC UA communication instructions:

- Method calls
- Reading and writing of data

The client opens up the following application areas, for example:

- Vertical communication to MES systems or cloud services
- Controller-Controller communication

What's new in TIA Portal V15

2

2.1 Introduction

Description

All important new features compared to existing versions are summarized here. You can find additional details on the various topics in the individual sections of the product documentation.

2.2 SIMATIC STEP 7

Description

All important new features of STEP 7 are shown grouped together based on the engineering workflow.

Hardware configuration

- The following new CPUs of the S7-1500 family complete the portfolio:
  - CPU 1518(F)-4 PN/DP MFP (multifunctional platform)
  - CPU 1516T(F)
- Hardware recognition of real existing PROFINET IO devices
  The online function "hardware recognition" detects IO devices on the connected PROFINET subnet. You can transfer a detected device to your project: STEP 7 inserts the IO device with all the modules and submodules, thus saving the manual insertion of IO devices and modules from the hardware catalog.
- The "Go to device view" function enables fast navigation from the PLC tag table to the configured input or output in the device view.
- Extended alarm display with new filter functions.

Editors for programming languages

- Multilingual project texts, such as block and network titles or comments, can be displayed and edited directly in the programming editors in all available languages. You can also export and import the texts for external translation. The following editors support local display of project texts: PLC tag table, programming editor (LAD, FBD, SCL, STL, GRAPH), data blocks and PLC data types.
- The status bar has been extended in SCL. The current cursor position is now displayed. By double-clicking on the line number, you can call the "Go to line number" function to navigate to additional lines. The editing mode (insert/overwrite) is also shown in the status bar. You can switch the editing mode by double-clicking on the display field.

New instructions

- You can use the "FileReadC" and "FileWriteC" instructions to read data from an ASCII file on the SIMATIC memory card on S7-1500 CPUs or write data to an ASCII file on the SIMATIC memory card.
- You can use the "GetClockStatus" instruction to read the following information from the CPU-internal clock on S7-1500 CPUs:
  - Information about time synchronization via an NTP Server
  - Information about whether automatic switching of daylight saving to standard time is activated
- With the help of the "EQ_TypeOfDB" instruction (SCL: "TypeofDB"), the data type of a data block which is addressed via a DB_ANY tag can be determined.

- The existing "SCATTER" and "GATHER" instructions have been extended and now also support an anonymous STRUCT data type and PLC data types with Boolean elements only.
- There are two new PID auxiliary functions available: They reduce the programming effort for tasks in control technology.
  - "SplitRange"
    Is used for distribution of the controller output to several actuators. Dependent on the controller output, this enables different actuators to be controlled
  - "RampFunction"
    This limits the sweep rate and the maximum values of signals. The slew rate can be individually limited in four operating areas: For positive or negative signals, for rising or falling signals.

Language innovations

- "References" provides a new type of pointer. References are typed pointers that refer to a specific data type. When you use references, you define the data type during the creating of the program. Because the data type does not have to be determined at runtime, the program will perform better and be more clearly structured. "Dereferencing" gives can direct write or read access to the referenced tag without having to copy the tag beforehand or integrate additional instructions in your program.
  A CPU of the S7-1500 series as of version V2.5 is required to use referencing.
- Constants of the BOOL type can be used at inputs of instructions in LAD/FBD. This facilitates the testing or "bridging" of current paths.
  An S7-1200 CPU as of FW V4.0 or an S7-1500 as of FW V1.8 is required for this function.

System functions

- The information system offers the possibility to display hardware manuals integrated. This has the advantage that you can search, filter or use the content as a favorite. Some hardware manuals are already included in the installation of the TIA Portal V15. If necessary, you can download additional hardware manuals available as support packages. To do this, select the function "Check for updates" in the toolbar of the table of contents.
- Local user and rights management
  - Managing project users
  - Open new function rights as read only / open project as read only + write / create users and roles
  - Managing project roles
  - Assignment of users to project roles
  - A block for industrial security with TIA Portal
  - Efficient user administration for products in a TIA Portal project

Libraries

- Global libraries can be exported as read-only libraries. The contained library types have permanent write protection and are therefore protected against changes. Write protection of the library types is retained even when they are used as a type instance. Write protection of protected global libraries cannot be revoked. Read-only libraries are available for PLC and HMI types.

New functionality for controlling operands

- Boolean operands can be controlled by double-clicking on the monitor value. The actual values can be toggled easily and quickly.
- Non-Boolean operands can also be controlled easily and quickly via the stored dialog.
- Both global tags and instance tags are supported in the DB.

Cross-references for instructions used

- User-defined "Instruction" type filters enable you to display all the instructions used in a CPU in the cross-references with the respective instruction versions.

Loading PLC tag tables

- PLC tag tables can be uploaded to and downloaded from the device in the specified structure and enable improved team engineering on the CPU.
- The PLC tag tables on the CPU are also displayed with "Accessible devices" and on the memory card.
- The online/offline comparison gives you a detailed overview of the tags available online and offline in the individual PLC tag tables.

Breakpoints for S7-1500 as of firmware V2.5

- Testing with breakpoints is also possible in mixed LAD/FBD blocks for SCL and STL programs.
- When a breakpoint is reached, the CPU goes to the "HOLD" operating state. Tags can also be monitored and controlled when the breakpoint is reached.
- This allows the STL and SCL program code to be tested step-by-step using breakpoints in order to find and correct logical errors during program creation. This enables easy and fast analysis of complex programs before the actual commissioning.

PLCSIM

- Know-how protected blocks of an S7-1500 CPU can be simulated with PLCSIM V15 (S7-1200 CPUs are currently not supported).
- Using a slider for analog values and pushbuttons for Boolean values, you can very easily make changes to values within the SIM table for quick testing of the STEP 7 user program.
- PLCSIM V15 and PLCSIM Advanced V2.0 can be installed on the same PC. Simultaneous use of both simulation tools is excluded, however.

Trace

- Generation of virtual signals based on mathematical functions from the recorded signals. The following functions are supported:
  - Basic arithmetic
  - Absolute value, square root, square, 1/x, modulo operation
  - Integral, differentiation
  - Filter function
- Calculation of the mean, RMS and integral of the selected signal in the range of the measurement cursors
- Time tags of type TIME, LTIME, TOD, LTOD, DATE and LDT can be recorded as a signal and used to specify trigger conditions
- Import and export of superimposed measurements
- Moving the measurement cursors using the arrow keys

2.3 SIMATIC WinCC

HMI devices

The following HMI devices can also be configured in the TIA Portal:

- TP1200 Comfort PRO
- TP1500 Comfort PRO
- TP1900 Comfort PRO
- TP2200 Comfort PRO

HMI device versions

To optimize the size of the standard installation, older versions of the HMI device images are swapped out to an additional DVD. The device versions affected are:

- V13
- V13 SP1
- V13 SP2
- V14

SVG graphics

In WinCC, the format SVG is also supported for graphics.

Extension of the ProDiag functionality

The criteria analysis display can be configured for improved process diagnostics. This makes it possible to display the operands with error that have triggered a selected ProDiag or GRAPH alarm in the user program.

OPC UA Client in WinCC RT Professional

Communication via OPC UA has been extended.

- Arrays and array variables are supported.
- Communication via an OPC UA connection can be protected with a password.

Number of connections with WinCC RT Professional

The number of possible integrated connections for S7-1200 and S7-1500 was increased to 128.

2.4 SINAMICS Startdrive

SINAMICS Startdrive Basic (formerly SINAMICS Startdrive)

- Enhancements for CU320-2 PN-based drive devices:
  - Support of G130, G150, S150 and Sinamics MV (up to 85.00 kW)
  - Support of Chassis and Cabinet Modules (not Blocksize Modules)
  - Support of SIMOTICS asynchronous motors and non-Siemens motors
  - Vector control
  - Parameter comparison (online/offline, comparison to factory setting)
- Enhancements for SINAMICS G120 product family:
  - Support of SINAMICS Firmware Version V4.7 SP9
  - Further optimization and expansion of the commissioning wizard:
    - Configuration of motor holding brake
    - Cancel online
    - CU250D-2: SSI encoder as motor encoder
  - PROFINET name assignment without restart of G120 Control Unit also in the list of accessible devices
  - Support of CU240D-2/CU250D-2 with polymer optical fiber (POF)
- Openness for drive devices:
  - Creating drive devices and components (including frame configuration)
  - Setting of selected drive parameters (offline and online, reading and writing)
  - Download to the device (no upload)

SINAMICS Startdrive Advanced

- Introduction of Startdrive Advanced license for use of additional engineering functions with high added value
- Only license key is needed, no additional installation
- Free trial license without license key (21 days)
- Functions in V15: Safety acceptance test for G120 family
  - Guided acceptance test wizard for all Safety Integrated functions (Basic und Extended Safety)
  - Automatic and safety function-specific creation of traces
  - Generation of an acceptance report as an Excel file

2.5 SIMOCODE ES

Description

The following new features and innovations are available at SIMOCODE ES in the TIA Portal:

- User-configurable display of device parameters
  - A new view of the device parameters, the "Expert list", is available in the parameter editor.
  - Groups of parameters as well as individual parameters can be marked as favorites.
  - A filtered parameter view is possible based on these favorites.
  - Export and import this selection of favorites (without values)
- Bulk engineering
  - High-performance function for bulk engineering of devices within a project (also for different SIMOCODE for each device)
  - Selected and appropriately set parameters can be transferred to a group of devices in a single step.
- Parameter wizard
  - Step-by-step configuration of all important device parameters
  - Settings for the fieldbus interface, device configuration, motor protection and monitoring functions can be queried and stored
  - User can call up the wizard directly from the Parameter Editor at any time.
- Automatic generation of PLC tags
  - If STEP 7 is additionally installed and one SIMOCODE per device is connected to a PLC, PLC tags are automatically generated for all inputs and outputs that are exchanged with a PLC via PROFIBUS or PROFINET.
  - Thanks to this function, the user can directly access these data in the user program using symbolic addressing.
- Automatic layout of graphical blocks
  - Placement of the blocks in fixed columns similar to the IPO principle (IPO = input, processing, output)
- All parameters can now be edited in the graphics editor, not just the connections.
  - Values can be entered directly at the function block using the keyboard.

2.6 Engineering options

PLCSIM Advanced

- Synchronization of PLCSIM Advanced with co-simulations tools on process image partitions of cyclic OBs (e.g. cyclic interrupt OBs)
- Support of acyclic services (RDREC/WRREC) and interrupts (e.g. hardware interrupts)
- Hardware interrupts configured in the TIA Portal can be read out via the API
- Simple backup and reloading of the software und hardware configuration of PLCSIM Advanced instances
- The storage path of the virtual SIMATIC memory card is freely selectable.
- GUI expansions:
  - Autocomplete
  - Buttons for RUN / STOP and Memory Reset integrated in the GUI of PLCSIM Advanced
- Computer-to-computer transfer of SIMATIC memory cards via the API
- Improved performance for symbolic access
- Parallel installation of PLCSIM and PLCSIM Advanced on a PC

User Management Component

- Central. cross-project user management in the system
- Managing user groups
- Import of Windows users and user groups is possible
- Efficient administration of users / user groups in a system
- Fault tolerance through redundant design of a UMC domain
- Load distribution of login request surges by means of several UMC stations in a UMC domain
- Licensing via the number of users

Multiuser Engineering

- Multiuser objects are now automatically selected when editing. Manual selection is still possible. Automatic selection increases the reliability when working together with Multiuser Engineering.
- Working with Multiuser Engineering is now also possible offline and without a direct server connection.
- Comment functions for documentation of the type and scope of the changes made have been added to the multiuser editors.
- When checking in, a filter is available for viewing objects with "conflicts".

- The multiuser server provides extended project history with a restore function. This enables traceability of the project process on the multiuser server. The milestones of a project can be commented and saved. The project history can be exported for extended evaluation.
- Multiuser Server V14 and V15 can be used side-by-side. Multiuser Server V15 also supports TIA Portal projects from V14 with the corresponding scope of functions.
- The external multiuser tools for the configuration and administration of the multiuser server are now available in all languages of the TIA Portal.

SIMATIC Visualization Architect

- Generation of alarms, alarms classes and alarm groups
- Support of the HMI objects "trend view", "f(x) trend view" and "f(t) trend view"
- Support of screen templates

SIMATIC Energy Suite

- Visualization of energy data in energy objects does not require additional PowerTags in WinCC RT Professional.
- S7 energy efficiency monitor for machines (new S7 instruction in STEP 7: "EnS_EEm_Calc" and "EnS_EEm_Report"):
  - For production-related and standardized determination of energy consumptions in machines (according to VDMA 34179)
  - Easy integration in machine controller (S7-1200/1500) and local visualization of the efficiency status
  - Automatic long-term measurements (e.g. batch, shift)
  - Efficiency report (.csv) for data evaluation and documentation
- Support of Sentron PAC 3200/4200 configured using the Hardware Catalog (MDD) – previously only GSDML was supported.

SIMATIC TIA Portal Openness

- Openness DLLs of V14SP1 and V15 in scope of delivery
  Because the Openness DLLs of V14SP1 and V15 are included in the scope of delivery, applications based on V14SP1 also run in V15 without modification. To make use of the functions of V15, you must integrate the DLL of V15 and recompile the application
- Export and import of SCL blocks
  Both SCL blocks as well as LAD and FBD blocks with SCL networks can be exported and imported as XML.
- PLC download
  The download of standard S7-1500 PLCs can be automated. A stop and start of the PLC is implicitly performed for this. Protection level passwords and binding passwords can be transferred from the application.

- Reading a checksum
  The checksum of the standard PLC program can be read from an offline PLC.
- ProDiag
  FBs, instances and assignments for ProDIAG can be created.
- System folder for UDTs
  System UDTs can be accessed in the system group within the user data types.
- Block numbering
  The automatic block numbering can be switched on and off. In addition, block numbers can be changed.
- Technology objects
  Openness includes enhancements for downloading and saving data of the TO Plotter and for new TO features.
- Startdrive
  DriveObjects and frames can be created for the SINAMICS G120 and SINAMICS S120 drives. Selected drive parameters can be set online and offline. Downloading to the device is possible.
- SINAMICS in AutomationML
  V90 slaves und SINAMICS G120 Profinet head modules can be exported and imported using AutomationML.

SIMATIC STEP 7 Safety

SIMATIC STEP 7 Safety Basic/Advanced V15, the high-performance option package for programming fail-safe S7 controllers for the Totally Integrated Automation Portal V15.

- Fail-safe arrays (read) for data types INT and DINT
  Safety V15 provides new instructions for reading fail-safe integer and double integer values from fail-safe arrays. The arrays can be created in global fail-safe data blocks (F-global DBs) and may contain up to 10,000 elements of the data type INT or DINT.
  The array elements are accessed with the instructions RD_ARRAY_I or RD_ARRAY_DI.
  The instructions are available for the fail-safe S7-1500 CPUs.
- Hardware and software signatures
  The F-collective signature uniquely identifies a particular status of the safety program and the safety-relevant parameters of the F-CPU and F-I/O.
  In addition to the F-collective signature, with Safety V15 you receive additional information regarding the fail-safe software and hardware signatures for the fail-safe S7-1200 and S7-1500 CPUs. These additional signatures enable a better differentiation between hardware- and software-relevant changes, thus simplifying the acceptance of changes.
  The F-signatures are displayed in the Safety Administration Editor (SAE) and are part of the Safety printout.

- Overflow handling
  When converters or mathematical functions such as additions, multiplications etc. are used, there is the possibility of a range overflow, which is detected by fail-safe processing and results in a STOP of the CPU.
  This STOP can be prevented by reliable overflow handling.
  Fail-safe S7-300 / S7-400 CPUs require the placing of a corresponding overflow instruction "Query status bit OV" in the following network.
  Overflow handling can be programmed considerably easier and by analogy with the STEP 7 Standard for the fail-safe S7-1200 / S7-1500 CPUs.
  Overflow handling is programmed and a STOP of the CPU in the event of an overflow can be prevented just by wiring the enable output ENO. If the result of the instruction is outside the range permitted for the data type, then the enable output ENO returns the signal state "0". The result of the instruction then behaves like the corresponding instruction in a standard block.
  Overflow handling is supported for the following instructions:
  - ADD: Add
  - SUB: Subtract
  - MUL: Multiply
  - DIV: Divide
  - NEG: Create two's complement
  - ABS: Form absolute value (S7-1200, S7-1500)
  - CONVERT: Convert value
- Isochronous fail-safe OB
  The isochronous F-OBs of Safety V15 enable the connection of isochronous PROFIsafe devices such as SINAMICS S120 (as of FW V5.1) for minimum response times and jitter.
  Requirement: F-CPUs S7-1500 firmware version 2.0 and higher that support IRT.
- Usability improvements
  With V15, Safety Basic / Advanced offers a number of usability improvements which simplify the creation of fail-safe user programs:
  - Readback of fail-safe F-FB Out tags also on S7-1200 and S7-1500 CPUs
  - Writing von F-FB input tags analogous to the STEP 7 standard
  - Start values of instance DBs can be changed
- Other innovations
  - The monitoring of system-integrated F-IO-DBs and F-runtime group information DBs using ProDiag is supported.
  - DINT -> INT converter (S7-1200, S7-1500)
  - ABS: Form absolute value (S7-1200, S7-1500)

See also

SIMOCODE ES (Page 26)

2.7 Runtime options

Description

The following new options are only offered as a runtime option with the TIA Portal. ProDiag is already integrated into the existing STEP 7 and WinCC products and only needs to be licensed when used on the real hardware. Unlike the engineering options, the runtime options are not version-dependent.

SIMATIC ProDiag

- S7 GRAPH blocks as of version 5.0 allow GRAPH step and GRAPH transition names to be compiled, as well as the initial values of interlocks and transitions to be reset.
- Improvements for multiple selection in global data blocks, function blocks and tag tables
  - New monitoring functions added, even when the multiple selection contains non-Boolean elements
  - Deletion includes all monitoring functions, even if an element contains more than one monitoring function
  - All the properties of the monitoring function are included in copying when a monitoring function is copied. During insertion, new monitoring functions are created with the copied properties and existing monitoring functions are overwritten. If an element contains several monitoring functions, these are deleted and one or more new monitoring functions with the copied properties are created.
  - During automatic filling of the cells, new monitoring functions are created in the marked cells of a global data block or of a function block and the properties of the selected monitoring function(s) are applied.
- Within the tag table, the FB block interface or global data blocks the "Monitoring" column displays how many monitoring functions the absolute address contains.
- The global search now also finds the ProDiag texts of the created monitoring functions from the Inspector window, from the ProDiag function blocks and from the ProDiag monitoring functions.

OPC UA

For S7-1500 CPUs firmware V2.5 and higher, with a corresponding Runtime license you can benefit from the following expansions of the integrated OPC UA server:

- Methods can be provided via the user program of an S7-1500 CPU. These methods use OPC UA clients in order, for example, to process a manufacturing order via the method call by the S7-1500 CPU.
- Import of standardized OPC UA information models (companion specifications). For this, you use the uniform interfaces for a large selection of devices or machines - e.g. for modeling of RFID readers or injection molding machines. Examples of this are the companion specifications AutoID and Euromap 77.

What's new in TIA Portal V14 SP1

3

3.1 Introduction

Description

All important new features compared to the present version V14 are summarized here. You can find additional details on the various topics in the individual sections of the product documentation.

3.2 SIMATIC STEP 7

Description

All important new features of STEP 7 are shown grouped together based on the engineering workflow.

Hardware configuration

- New PS 60W 24/48/60VDC HF system power supply for the S7-1500 provides an energy buffer for data backup of up to 20 MB in the CPU.
- Tabular configuration of the I/O channels for modules of the ET 200MP DI HF and ET 200MP DO HF including copy-and-paste and auto-complete function
- Editing of devices names in the graphic views
- Enhanced zooming and optimized (multiple) selection in the graphic network and topology view
- Overview of I/O tags of an entire device (such as an ET 200SP station)
- Easy removal of unused GSD files in the project
- Wizard enhancement for UDP Multicast for Open User Communication (OUC) with S7-1500

Editors for programming languages

- In SCL program code structured tags and ARRAY elements can be replaced by other tags using drag-and-drop operation.
- In the SCL editor, it is possible to switch between insert and overwrite mode.
- In SCL, selected program code segments can be simply enclosed with specific structure elements, such as "IF..Then".
- The detailed comparison of individual blocks can be started directly from the project tree (offline/offline, but also online/offline).
- In LAD/FBD, the tag comment can be shown directly at the point of use.
- The F7 functions ("Open block/PLC data type…") can also be started with focus inside editors.
- Generating external source file from blocks: The export of objects as external source file takes into consideration the PLC data types (UDTs), multi-instances and all called blocks defined in the block interface.
- All the PLC instructions used can be updated to the latest version release in the "Instructions" task card.

New instructions

- The values can be moved between tags of the data type BYTE, WORD, DWORD, LWORD and tag ARRAYs of the data type BOOL using the new instructions "SCATTER" and "GATHER" for the S7-1200/1500. As a result, control and status words can, for example, be simply broken down into single bit signals and reassembled.
- Information about the inserted SIMATIC memory card - such as the memory size, the amount of storage space used or the number of already completed write/delete actions (as percentage) – can be read out using the "GetSMCInfo" instruction.
- The "Polyline" instruction maps the input value to the output value using a characteristic curve. The characteristic curve is defined as a polyline (with up to 50 interpolation points).

Language innovations

- PLC data types (UDT) and ARRAY comments can be modified instance-specific.
- The assignment of various PLC data types (UDTs) with the same structure but different symbolic names is supported.
- S7-GRAPH blocks as of version 4.0 enable the initial value acquisition of the disturbed operands. The actual values of the signal states of Boolean operands and the results of comparator operations in interlocks and transitions are recorded continuously from the time of activation of the initial value acquisition, even if no error occurs. The signal states are stored for each cycle in the GRAPH instance data block. A maximum of 32 signal states are recorded per interlock or per transition of a GRAPH step.

Libraries

- Project libraries and global libraries can now be separately translated into additional foreign languages.
- Simplified upgrading of types (blocks, PLC data types) in an existing project library.
- The user-defined help can be now also be started in the context of the project language. Until V14, this was only possible in the context of the TIA Portal language.

Messages/diagnostics

- The display of the alarm display in the Inspector window has been optimized and functionally enhanced with new functions such as "Receive alarms" and "Freeze alarms".

Migration

- The local constants for ARRAY limits that are used in the SCL sources are retained during the migration from STEP 7 V5.x to TIA Portal.

Trace

- Switching the display format (for example: A WORD can be shown in hexadecimal format, as signed or unsigned INT)
- The scroll bar of the X axis has a small preview window ("Time range display"), with which you can recognize which are has just been zoomed.

System functions

- CAx data based on XML, for exchange of ECAD system for example, can be exported and re-imported.
- Filters are supported for cross-references that are freely definable by the user, for example, in order to find multiple accesses to tags.
- The TIA Portal can be connected to an external password manager via a password provider. This offers the advantage that passwords do not have to be released. This increases the security of the program code.

SIMATIC Ident

The technology object (TO) for SIMATIC Ident connects the hardware configuration of communication modules and readers to the programming of an S7-1200 and S7-1500 controller. The TO hereby prevents incorrect settings and detects errors in the parameter assignment.

Hardware configuration:

- The plant builder assigns a "unique name" to each communication module / reader with which it can be identified by both commissioning engineers and programmers.
- The parameter assignment of each communication module, reader or reading point is set via the TO.
- The Ident devices can be configured with a single click for operation with the TO using the "Auto Configuration" button.

New instructions:

- All instructions for SIMATIC Ident have been revised for the collaboration with the TO. You can find the instructions in the TIA Portal under "Optional packages" > "SIMATIC Ident".
- The hardware addressing of a reader to the instructions has been changed so that the reader is selected through the "unique name" which was defined in the TO. There is no need, therefore, for the previous searching, noting and transferred of the HW identifier and logical address.
- Optional inputs at the instructions are hidden. The user thus recognizes immediately the important parameters that are required for a function.
- The programmer no longer has to assign parameters to the hardware. Only the RESET_READER instruction, which has no other input parameters, is needed for the PLC startup, regardless of the reader connected. The instruction acquires the required parameters from a data area which was configured with the TO.

Diagnostics:

- The last messages which were reported by the instructions to the application can be displayed with the TO.
- A time stamp and the exact error text help to achieve fast analysis of the error.

3.3 SIMATIC WinCC

Function enhancement - Openness

- Support of HMI faceplates
- Support of HMI slide-in screens
- Support of HMI pop-up screens

Enhancements for SIMATIC HMI panels

- SIMATIC HMI BASIC Panels:
  - Unicode CSV export of recipes for Basic Panels
  - Changing of IP addresses of controllers and additional SIMATIC components (such as CPs) via the HMI device.
  - Support for Chinese, Japanese, Korean, Russian and Arabic fonts in the browser.
- Updating of tested network cameras (AXIS M1013, AXIS M1011, D-Link DCS-942L, D-LINK 4701E, Siemens CCMS2025).
- Installation/deinstallation of Word, Excel, Win Mediaplayer, IE via ProSave.
- Unlimited number of usable scripts for Comfort.
- Configuration for Startup (additional value "60s" for RT start time and system function for Restart Panel).
- Additional system functions in the "Zones" editor for Mobile Panels.

Enhancement for SIMATIC Runtime Advanced

- Camera displays for showing live video streams from network cameras
- Enhancements of the PDF display (search and copy)
- License-free use of the Sm@rtServer on Runtime Advanced (similar to Comfort Panels)

Integration of SINUMERIK Operate (GUI and data) in Runtime Advanced

- Access to SINUMERIK Operate user interface via new NC operator display in screens of RT Advanced (also multitouch operation)
- Access to SINUMERIK NCU data via separate SINUMERIK channel

Enhancements for SIMATIC Runtime Professional

- Support for OPC UA DA client
- New VBS methods for screen windows. This provides you with the option of temporarily switching off the operability of the screen window.
- Customization of the recipe view. This provides you with the option of adapting the properties of the controls (buttons or table elements).

3.4 SINAMICS Startdrive

Startdrive for G120

- Optimized and enhanced commissioning wizard:
  - The two former wizards for the standard commissioning and the operation under a SIMATIC Motion Control axis are combined in a single wizard
  - Integration of the selection of message frame, brake resistor, motor-side filter, temperature sensor, and selection of the motor, also via the motor code number
- Online/offline parameter comparison
- Support for Windows 10

Startdrive option for S120

- Automatic hardware configuration
- Support for drive-based extended safety with graphic parameterization masks
- Support for torque motors (1FW3, 1FW6)
- Networking with ET200 SP CPU
- Support for Windows 10

3.5 Engineering options

SIMATIC STEP 7 Safety

SIMATIC STEP 7 Safety Basic/Advanced V14 SP1, the high-performance option package for programming fail-safe S7 controllers for the Totally Integrated Automation Portal V14 SP1.

- Station upload
  The consistent hardware and software upload enables a loading of project data (including safety-related project data) from a fail-safe S7-1500 CPU*) to a programming device / PC. The project data are loaded without an additional generation step and are consistently available in the TIA Portal for immediate continued work.
  *) As of firmware version V2.1
- Configuration control
  With Safety V14 SP1, the configuration control (option handling) for SIMATIC fail-safe I/O devices is supported in the following configurations:
  - Distributed in an F-CPU S7-300/400/1200/1500
  - Centrally in an F-CPU S7-1500 (including fail-safe ET200SP CPUs)
- Cross-project IO controller I-device communication
  Safety V14 SP1 enables cross-project communication of fail-safe S7-1200/1500/1500 software controllers with S7-300/400/1200/1500 I-devices. The I-device project can be easily created with STEP 7 Safety V13 (or higher) or also S7 Distributed Safety V5.4, exported as GSD file, and imported into the IO controller project.

- Global F-IO Status block
  Fail-safe F I/O devices provide a wide range of status information about the status of the channel and module using the QBAD tags and the value status bits.
  In the Safety Administration editor, you can create a default FB for each F-runtime group in order to record and further process any channel, communication or fail-safe I/O errors, even in automation systems with very large volumes of project data. The value status bits and the QBAD tags of the fail-safe I/O used in the corresponding F-runtime group are evaluated and the result of the evaluation is made available at the "QSTATUS" output.
  You can use the default FB in the standard program.
- Openness
  You can use the Openness command set in STEP 7 Safety, as in the standard version. This requires that no password is set for the safety-relevant project data. The use of Openness in production operation is not permitted.
  Supported functions include the following:
  - Working with libraries (deleting, copying, instancing and upgrading elements etc.)
  - Importing tags, tag tables, PLC data types (UDTs), etc.
  - Compiling of SW, blocks, PLC data types (UDTs).
  - Plugging/deleting F-CPUs and fail-safe I/O
  - Copying/deleting F-CPUs and fail-safe I/O from master copies
  - Configuring networks
  - Compiling software (including the safety program)
  - Reading/configuring F-parameters of the F-CPU
  - Reading/configuring F-parameters of the fail-safe I/O
  - Reading, declaring or deleting fail-safe tags in the PLC tag table
  - Updating the project to the latest type versions of F-blocks

SINUMERIK STEP 7 Toolbox

The SINUMERIK STEP 7 Toolbox V14 SP1 is an option package for SIMATIC STEP 7 Professional V14 SP1 (TIA Portal) with additional setup.

The SINUMERIK STEP 7 Toolbox V14 SP1 enables the configuration of the SINUMERIK hardware in the TIA Portal through the following functionalities and tools:

- Addition of the following modules of SINUMERIK 840D sl to the hardware catalog (as of firmware V4.5 SP2):
  - NCU 710.3B
  - NCU 720.3B
  - NCU 730.3B
  - NX10.3
  - NX15.3
- Addition of the ADI4 module to the hardware catalog
- Basic SINUMERIK PLC program for the versions
  - V4.5.x.x
  - V4.7.x.x
  - V4.8.x.x
    The basic PLC program is installed in the form of the "SINUMERIK 840D sl PLC Basic Program" system library using the SINUMERIK STEP 7 Toolbox V14 SP1.
- Support for hardware migration
- Creation of SINUMERIK PLC commissioning logs, PLC hardware upgrade logs and reload logs
- Export of PLC symbols for SINUMERIK Operate
- Import of SINUMERIK user alarm texts
- Creation of SINUMERIK PLC logs
- Support of PROFINET IO IRT for NCK
- Support of SINUMERIK Safety Integrated and Safety Integrated plus
- NC-VAR-Selector (external tool)

3.6 Runtime options

Description

The following new options are only offered as a runtime option with TIA Portal V14 SP1. The ProDiag product is are already integrated into the existing STEP 7 and WinCC products and only needs to be licensed when used on the real hardware. In comparison with the Engineering options, the runtime options are not version-dependent.

SIMATIC ProDiag

- S7-GRAPH blocks as of version 4.0 enable the initial value acquisition of the disturbed operands.
- The initial values can be displayed on the HMI device using the "PLC code view" object and the faulty operands can be detected immediately in the case of error (criteria analysis). The initial values remain stored until they are overwritten by a new error.
- For the initial value acquisition, all instructions are supported which are supported in an S7-GRAPH function block for transitions and interlocks. For bit logic operations, the status of the operand is stored. For comparator operations, the result of the comparison is stored.

SIMATIC Energy Suite S7-1500

SIMATIC Energy Suite as an integrated option for the TIA Portal for the first time efficiently combines energy management with automation, bringing energy transparency to production. In addition, the configuration work is significantly reduced due to the simplified configuration of components for energy recording.

Functions:

- Integrated and intuitive configuration of energy management
- Automatic generation of the PLC energy program for S7-1500 for basic energy data (energy and power)
- Archiving to WinCC Runtime Professional or to a PLC-internal SIMATIC memory card
- Seamless connection to SIMATIC Energy Manager PRO

New in V14 SP1:

- Enhancement of the automatic PLC program generation with additional energy data (currents, voltages, frequency)
- In addition to the manual energy data export, the automatic export is now also offered (for example, start of the export of the energy data of the last week, each Monday at 00:15 hrs)
- Program generator: Detailed representation of the generation progress and configuration error messages.

What's new in TIA Portal V14

4

4.1 Introduction

Description

All important new features compared to the present version V13 SP1 are summarized here. You can find additional details on the various topics in the individual sections of the product documentation.

4.2 SIMATIC STEP 7

Description

All important new features of STEP 7 are shown grouped together based on the engineering workflow.

Hardware configuration

- Support of the S7-1500 T-CPU family, with the Motion Control functions "constant velocity" and "cam disks".
- Support of the new CPU 1518(F)-4 PN/DP ODK (Open Development Kit), which enables C/C++ functions created via the SIMATIC ODK 1500S to be integrated into the user program
- Support of the CPU 1516pro (F)-2 PN
- Support of the new S7-1500 failsafe software controller
- IO devices can be grouped in the project tree to quickly find and technologically merge devices.
- The arrangement of the devices in the topological view can be synchronized with the network view.
- Own profiles can be defined in the hardware catalog for easy and quick access to frequently used modules.

Editors for programming languages

- Separate editors are available for the new technology objects Output Cam, Cam Track and Measuring Input. There is also a convenient cam disk editor for the S7-1500T controllers.
- SCL networks can be inserted into a LAD/FBD block.
- "Regions" can be used in SCL program code to improve structuring and navigation.
- You can hide the block parameters when calling the block in LAD or FBD.
- The comparison of PLC programs has been thoroughly revised and enables the quick and convenient comparison of source code with many filter options.
- Networks in the programming editors for LAD / FBD / STL are automatically displayed as they were when you last exited the block.

Language innovations

- ARRAYs of variable length can be passed with ARRAY[*].
- Multiple instances can be declared in ARRAYs in the block interface and called in loops. This helps to significantly reduce the program code and provides better readability and more efficient programming when processing the same program objects.
- The data type DB_ANY can also be used for technology objects. This allows technology objects to be handled more flexibly in the program. An "ARRAY of DB_ANY" may be a list of positioning axes, for example.

- Instances can be passed as a parameter in the "InOut" section of the block interface so that only one dedicated instance needs to be selected in the call.
- Message texts and comments for up to 3 languages can be loaded into the PLC S7-1500 and changed online.

Messages/diagnostics

- Using the new "Get_Alarm" instruction, messages can be directly read from the message server of the S7-1500 CPU and, sent and to a higher-level reporting system (e.g. process control system), for example.
- With the instruction "GET_DIAG" in Mode 1, the diagnostic data are output in the DIS structure. Bit 15 of the parameter IOState of this structure outputs the signal status "1" when a network / hardware fault has occurred.

Trace

- The Trace function enables different measurements to be displayed in a shared diagram ("overlaid measurements").
- In addition, with the Trace function, after completion, messages can be stored on the memory card independently of the power supply, and if required, can be automatically restarted.

Online functions

- Restructuring of the toolbar in the DB editor with the possibility to reinitialize complete data blocks with the defined start values.
- Structures in the block can be observed directly in the Inspector window including the call environment.

System functions

- The new global search enables cross-project searching for PLC and HMI objects. In addition, the search results can be restricted by filter. A "GoTo" function is also available.
- TIA Portal settings can be exported or imported and stored on a central server. If the server path is available, for example, the stored settings are used automatically as "default" every time you restart the TIA Portal.

- Innovative help/information system with tab technology, improved search function and the possibility to filter information by device families.
- The new cross-reference list provides a quick overview of the use of objects and devices within the project.
  - Support of overlapping accesses (PLC tags)
  - Support of know-how-protected blocks
  - Access to Individual elements of an ARRAY
  - Representation of objects which occur both in HMI and PLC devices
  - Free entry of objects to be searched for in the cross-reference list possible
  - Various filter options
  - Convenient and central provision of updates in company networks.
  - A company server can be configured with the TIA Automation Corporate Configuration Tool. The available updates / support packages are stored on the server and are thus available to users.

S7-PLCSIM

- Quick start of S7-PLCSIM in compact mode and without a simulation project.
- Devices can be exchanged in the simulation project so that existing TIM tables and sequence entries can continue to be used without any problems.
- Sequences can be automatically started on the basis of a trigger condition.
- Sequences can be temporarily deactivate.
- Decentralized I/O devices can be simulated in the device view.

Motion Control applications

- Pre-configured and integrated connection and configuration of the SINAMICS V90 PN drive message to a Motion Control technology object of SIMATIC S7-1500.
- Automatic data synchronization between the controller and the drive.

See also

Engineering options (Page 54)

Runtime options (Page 56)

4.3 SIMATIC WinCC

Configuring screens

- New view on distributed I/Os of the system diagnostic controls.
- Improvements of controls:
  - New scroll bar with configurable colors.
  - Dynamic number of commands in the drop-down menu for recipe control.
  - Expansion of the slider, bar, gauge controls with up to four limits.
- New "Siemens TIA Portal Icons" fonts.
- Graphics library includes new graphics and completely revised structure.

Working with tags

- Symbolic address multiplexing of PlcUDT instances within a ARRAY variable of a data block.
- Hierarchical IntelliSense for connecting PLC data to HMI tags.

SIMATIC Panels and SIMATIC WinCC Runtime Advanced

- Transfer of configuration via USB stick / SD card for Basic Panels 2ndGeneration, Comfort Panels, Outdoor Panels, Mobile Panels and WinCC Runtime Advanced.
- Sm@rt Server Option for Basic Panel 2nd Generation.
- The safety of the following features has been enhanced:
  - The following components are signed:
    Comfort and Mobile Panel: Runtime, ProSave options and Runtime add-ons.
    Basic Panels 2nd Generation: Image of the panel
  - Security of passwords used in Runtime
  - Web browser connection to a Sm@rtServer via HTTPS protocol possible and secure connection from a Web SmartClient to the Sm@rtServer possible.
- The maximum number of power tags in WinCC Runtime Advanced was increased to 16K.

WinCC Runtime Professional

- WebUX option
  Platform and browser-independent operation and monitoring via the Web.
- Process diagnostics option
  When a process fault occurs, process diagnostics provide information about the cause and location of the error and supports troubleshooting.
- Extension: Quantity structure for 1200/1500 PLCs
  There can now be 64 connections to S7-1200/1500.

- Extension: Multitouch for certain controls
  Gestures can now be used to move the display area of controls (trend-based controls, grid-based controls, axis controls).
- Harmonization: Licenses for WEBNAVIGATOR, DATAMONITOR and WebUX
  The license type of Web options can be counted consistently. These SCADA options as of V7.4 or V14 in the future will have uniform MLFBs and therefore there will no longer be separate MLFBs for Runtime Professional.

See also

Engineering options (Page 54)

Runtime options (Page 56)

4.4 SINAMICS Startdrive

Simplified commissioning and simplified (remote) servicing

Extended routing functionality for communication between Startdrive and drive:

- Routing across various PROFINET systems
- Routing from PROFIBUS to PROFINET

Simple modification of a project to the machine

Upgrade G120 firmware version in the Startdrive project (beginning with firmware version 4.4)

Reuse of the parameter assignment across various performance ranges.

Also for G120C

Faster access to Smartdrive download package in SIOS

- No waiting time for export control
- Starting with the Startdrive V13 SP1, export restriction no longer applies (export identifier AL=N, ECCN=N)

More efficient configuration

Optimized compilation time of the S7-CPU for networked drives configured with Startdrive

Modification of the supplied S7 data blocks for drive control

Drive library now without know-how protection

Simplified commissioning

Assignment of PROFINET device names is now easy to do in the project navigation and in the HWCN network view via the shortcut menu of the drive.

Simplified update process

Automatic installation of Startdrive HSPs and updates via the TIA Portal Updater

Support for new hardware

New 1FP1 synchronous Reluktanz motors are supported.

No manual update required

HSPs for Sinamics Firmware V4.7.3 and V4.7.6 are integrated.

4.5 SIMOTION SCOUT TIA

Description

The SIMOTION P platform (PC-based) is now available in the TIA Portal. (Planned for approx. 3 months after release of STEP 7)

- You can now upload the hardware configuration for SIMOTION devices.
- Additional PROFINET functions for SIMOTION devices:
  - Shared I-device
  - Shared device
  - Seamless Media Redundancy (MRPD)
  - "Check of VendorID/DeviceID" and "Overwriting of the NameOfStation" (automatic configuration of the topology)
  - Address tailoring (modular machines)

4.6 SIMOCODE ES V14

Routing via Ethernet - PROFINET

SIMOCODE ES V14 supports routing to SIMOCODE pro devices with PROFINET. Devices can now also be reached from an engineering station with SIMOCODE ES V14 Premium via Ethernet, if they are not directly connected to the PROFINET network to which the SIMOCODE pro devices are connected. This improves the capabilities of the software in systems in which SIMOCODE pro devices are used with PROFINET.

Migration of SIMOCODE ES 2007 group files

Configurations created with SIMOCODE ES 2007 in which the parameter files for SIMOCODE pro devices have been assembled into "group files", can now be migrated directly to one and the same TIA Portal V14 project without additional configuration steps. This function improves the handling of the software when migrating from SIMOCODE ES 2007 to SIMOCODE ES V14 TIA Portal.

Improved editor functions for charts with SIMOCODE pro interconnections

With SIMOCODE ES V14 Standard and Premium, CFC-based charts can be copied in whole or part, saved as library elements and reused. This feature simplifies the handling of recurrent configurations for typical configurations.

Clear presentation of all the important parameter settings in the parameter editor

All important parameter settings for fast commissioning of a motor feeder can be displayed and edited in a new overview in SIMOCODE ES V14, which significantly helps to improve clarity.

Uniform presentation of recorded trends for measured values

Measured values and data recorded online and offline with SIMOCODE ES "Live Trend" and "Analog Value Recording" functions are displayed using the TIA Portal "Trace" function. The operation of this function has been significantly improved thanks to the TIA Portal-wide standardization.

4.7 Engineering options

TIA Portal Multiuser

TIA Portal Multiuser Engineering enables several users to simultaneously work together on a project. This reduces configuration times considerably and enables quicker commissioning of projects.

The basic principle:

- An independent server application carries out project management. The server application can be installed independent of a TIA Portal.
  or
  project management is located on a server. The server can be installed independent of a TIA Portal.
- Different users work in local sessions, based on the projects managed by the server.
- Users work independently from each other in local sessions; changes made by other editors are displayed and can be applied.
- The changes from the local sessions are transferred to the server project on check-in.

TIA Portal Teamcenter Gateway

The TIA Portal Teamcenter Gateway allows you to store and manage TIA Portal projects and global libraries in the Teamcenter. The operation is performed in the TIA Portal.

TIA Portal Cloud Connector

The TIA Portal Cloud Connector lets you access local PG/PC interfaces and the SIMATIC hardware connected to them in the TIA Portal engineering even though the engineering itself is operated in a private cloud using Remote Desktop.

SIMATIC Energy Suite

SIMATIC Energy Suite as an integrated option for the TIA Portal for the first time efficiently combines energy management with automation, bringing energy transparency to production. In addition, the configuration work is significantly reduced due to the simplified configuration of components for energy recording.

- Integrated and intuitive configuration of energy management
- Automatic creation of PLC energy program for S7-1500
- Archiving to WinCC Runtime Professional or to a PLC-internal SIMATIC memory card
- Seamless connection to SIMATIC Energy Manager PRO

SIMATIC S7-PLCSIM Advanced

S7-PLCSIM Advanced is suitable for the simulation of an S7-1500 PLC. Compared to standard S7-PLCSIM, S7-PLCSIM Advanced provides the following additional features:

- API for interfacing with co-simulations
- Multiple, distributed instances possible
- Adjustable, virtual time
- Web server and OPC UA access

SIMATIC ODK 1500S

The SIMATIC ODK 1500S enables the development of function libraries with the high-level languages C/C++ for ODK-capable SIMATIC S7-1500 PLCs. These function libraries can be directly loaded and run directly from the user program. ODK-capable S7-1500 PLCs:

- S7-1500 Software Controller (CPU 1507S (F))
- ET 200SP Open Controller (CPU 1515SP PC (F))
- S7-1500 Advanced Controller (CPU 1518(F)-4 PN/DP ODK)

SIMATIC Target 1500S™ for Simulink®

SIMATIC Target 1500S™ for Simulink® is an add-on for Simulink from MathWorks. It enables the integration of Simulink models directly in the program cycle of an ODK-capable S7-1500 PLC.

- Enables model-based development with MATLAB® and Simulink for SIMATIC controllers
- Seamless integration in Simulink as a so-called target file
- No C/C++ or ODK know-how required
- Support of the Simulink external mode for direct monitoring and control of model parameters in the controller from Simulink
- The generated program code can be executed on the CPUs 1507S(F) V2.0, CPU 1515SP PC (F) V2.0 and CPU 1518(F)-4 PN/DP ODK.

SIMATIC Visualization Architect

The SIMATIC Visualization Architect (SiVArc) enables easy, fast and flexible automated production of HMI content based on the STEP 7 user program.

- Supported HMI devices:
  Comfort Panel, Mobile Panel 2nd Generation, WinCC RT Professional, WinCC RT Advanced
- Supported HMI objects:
  Screens, text lists, power tags, captions, buttons, I/O fields, symbolic I/O fields, graphic I/O fields, switches, round buttons
- Configuration via control editor
- Templates are managed through libraries

4.8 Runtime options

Description

The following new options are only offered as a runtime option with TIA Portal V14. The products ProDiag and OPC UA S7-1500 are already integrated into the existing products STEP 7 and WinCC and only need to be licensed when used on the real hardware. In comparison with the Engineering options, the runtime options are not version-dependent.

SIMATIC ProDiag

SIMATIC ProDiag enables precise and fast machine and plant diagnostics for SIMATIC S7-1500 and SIMATIC HMI:

- Central cycle-precise time stamp of error messages
- Automatic generation of monitoring logic and message calls
- Automatic update of the SIMATIC HMI when changes are made to message configuration for 3 languages
- HMI systems do not have to leave runtime mode in the event of changes
- Is directly available in the language editors LAD, FBD, SCL and STL.
- Monitoring can be parameterized later on F blocks and know-how protected blocks
- Central project definition of message structure
- Alarm texts are automatically derived from existing information in the project
- Simple visualization of the HMI through prepared controls

SIMATIC OPC UA S7-1500

The OPC UA S7-1500 option allows easily connection of any third-party devices to the S7-1500 via the OPC UA server integrated in the S7-1500 CPU:

- OPC UA server directly in S7-1500
- OPC UA data access, read/write, subscription to value changes
- Security
- XML export for offline configuration of OPC UA clients

SIMATIC Energy Suite S7-1500

10 energy objects are supplied with the SIMATIC Energy Suite Engineering. Additional energy objects can be extended by the Runtime packages.

WinCC / WebUX

WinCC / WebUX allows us to monitor plant processes over the Internet or Intranet using mobile devices and if required, to also control them. The software offers a visualization solution which is suitable for HTML5 and SVG-capable browsers, independent of the operating system – and which does not require engineering work.

SIEMENS

Notes on the TIA Portal 1

STEP 7 2

WinCC 3

Readme

Programming and Operating Manual

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Notes on the TIA Portal....5

1.1 General notes....5

1.2 Notes on libraries....8

1.3 Notes on memory cards....9

1.4 Notes on the hardware configuration....10

1.5 Notes on using the TIA Portal in a virtual environment (private cloud)....11

1.6 Notes on using the TIA Portal Teamcenter Gateway....12

1.7 Information on working with Multiuser Engineering....14

1.8 Information about engineering options....16

2 STEP 7....17

2.1 Security information....17

2.2 Notes on use....19

2.3 Editing devices and networks....21
2.3.1 General information on devices and networks....21
2.3.2 Information about OPC UA functions....21
2.3.3 Hardware detection of IO devices....24
2.3.4 HW ID of type "Hw_SubModule" for an HMI Panel IO device....25
2.3.5 Use of modules on the S7-1200....25
2.3.6 CP 343-2 on SIMATIC S7 Embedded Controller EC31-RTX....26
2.3.7 Loading of S7-1500 CPUs (SIPLUS)....26
2.3.8 ET 200MP (IM 155-5 DP ST) on a WinLC RTX....27
2.3.9 Compiling ET 200SP HF V2.1 and higher, ET 200SP HS V4.0....28
2.3.10 Notes on online and diagnostics....28
2.3.11 Network components....29
2.3.11.1 Network and communication modules....29
2.3.12 Notes on the S7-1500R/H redundant system....34

2.4 Programming a PLC....35
2.4.1 General notes on PLC programming....35
2.4.2 Notes on instructions....38
2.4.3 Note on software units....39
2.4.4 Testing the user program....39
2.4.4.1 Testing with the watch table....39
2.4.4.2 Testing with the force table....40
2.4.5 Compatibility of PLC programs from older versions....41
2.4.5.1 Compatibility of PLC programs from versions prior to V15.1....41
2.4.5.2 Compatibility of PLC programs from versions prior to V15....43
2.4.5.3 Compatibility of PLC programs from versions prior to V14 SP1....46
2.4.5.4 Compatibility of PLC programs from versions prior to V14....48
2.4.5.5 Compatibility of PLC programs from versions prior to V13 SP1....53

2.5 Inter Project Engineering (IPE)....58
2.5.1 Notes on IPE....58

2.6 Technological functions....59
2.6.1 Notes on S7-1500 Motion Control (S7-1500)....59
2.6.2 Notes on additional technology functions (S7-1500)....61
2.6.3 Notes on technological functions (S7-1200)....62
2.6.4 Notes on technological functions (S7-300, S7-400)....63
2.6.5 SIMATIC STEP 7 Easy Motion Control....63
2.6.5.1 Notes on SIMATIC STEP 7 Easy Motion Control....63
2.6.6 SIMATIC STEP 7 PID Professional....64
2.6.6.1 Notes on SIMATIC STEP 7 PID Professional....64

3 WinCC....65

3.1 Security information....65

3.2 Notes on use....72

3.3 Migration....75

3.4 Engineering System....77
3.4.1 Screens and Screen Objects....77
3.4.2 Tags and connections....81
3.4.3 Alarm system and alarm displays....83
3.4.4 System functions and scripts....84
3.4.5 Reports....87
3.4.6 Recipes....87
3.4.7 User administration....87
3.4.8 Communication....87

3.5 System-wide functions....92

3.6 Compiling and loading....93

3.7 Runtime....96
3.7.1 Notes on operation in Runtime....96
3.7.2 Notes on operation of panels in Runtime....98
3.7.3 Notes on operation of Runtime Advanced....99
3.7.4 Notes on operation of Runtime Professional....100

3.8 HMI devices....105
3.8.1 Notes on HMI devices....105

3.9 Add-ons....110
3.9.1 Sm@rtServer....110
3.9.2 Audit....110
3.9.3 DataMonitor....110
3.9.4 WebNavigator....111
3.9.5 WebUX....113
3.9.6 Redundancy....113
3.9.7 SIMATIC Information Server....114

Index....115

Notes on the TIA Portal 1

1.1 General notes

The information in this readme file supersedes statements made in other documents.

Read the following notes carefully because they include important information for installation and use. Read these notes prior to installation.

Windows 7 / Windows Server 2008 R2: Invalid certificate validation

The new certificates used for signing may possibly not be validated on the Windows 7 and Windows Server 2008 R2 operating systems.

If you do not have the possibility to update the certificate store online, the signature of the product files is displayed as invalid. In this case, update the root certificates provided by Microsoft on the offline systems. To do so, follow the instructions in FAQ 109748887 (https://support.industry.siemens.com/cs/ww/en/view/109748887).

Display of Asian characters in the TIA Portal

Due to a change in behavior in Microsoft Windows, it may occur that texts are not displayed correctly in the TIA Portal when a Chinese TIA Portal is installed on another Asian operating system (e.g. Korean). To view the texts in TIA Portal correctly, open the Windows Control Panel and select "English" under "Language for non-Unicode programs". Note, however, that this may cause display problems in other programs.

Installing new .Net versions or .Net service packs

- Close the TIA Portal before installing a new .Net version or a new .Net service pack on your programming device/PC.
- Restart the TIA Portal only after successful installation of the new .Net version or the new .Net service pack.

Notes on handling

- If a project in the list of projects last used is located on a network drive that is not connected, you may experience delays when opening the "Project" menu.
- When you insert a CPU, you may need to wait for some time if the project editor is open at the same time. This generally takes longer when you insert the first CPU in a newly created project. To be able to continue working at a good pace, you should close the project text editor before inserting a CPU.
- The alarm "Application is not responding" may appear in Windows 7 with functions that take a long time to run (loading the CPU for example). If this occurs, wait until the function has correctly finished.

- If you have installed a Microsoft mouse with IntelliPoint, you may find that it superimposes components over the buttons of the title bar. If this is the case, uninstall the IntelliPoint software from Microsoft.
- Enabling the "Virtual Desktop" options with NVIDIA graphics cards can cause problems. In this case, disable the "nView virtual desktop manager" of your NVIDIA graphics driver.

Working with automatically synchronized network drives

A TIA Portal project consists of multiple files that are saved together in one directory. If you store a project that is automatically synchronized on a network drive or in a cloud directory (e.g. Dropbox, Syncplicity or GoogleDrive), this can lead to data loss if the synchronization only takes place partially or asynchronously. For this reason, we do not recommend editing TIA Portal projects directly on synchronized network drives or in cloud directories. Always close the TIA Portal project before synchronization and make sure that all directories and files from the project directory are synchronized together and completely. You may wish to disable automatic synchronization while working with the TIA Portal. The synchronization itself must be implemented in such a way that the current (local) project data replaces the project data on the network drive or in the cloud directory.

Entry of decimal places

With certain Windows language settings, it may occur that the entry of values with a comma as decimal place is not recognized (entering "1,23" leads to an error). Instead, use the international format ("1.23").

Information on the TIA Portal in online support

Overview of the most important technical information and solutions for the TIA Portal in the Siemens Industry online support.

Internet link: TIA Portal in Siemens Industry online support (https://support.industry.siemens.com/cs/ww/en/view/65601780)

All information on service and support in the Siemens Industry online support:

Internet link: Service and support in Siemens Industry online support (https://support.industry.siemens.com/cs/de/en/)

Here, you can also subscribe to the newsletter that provides you with latest information relating to your products.

Opening projects with missing software components

If a project cannot be opened in TIA Portal due to missing software components, you can download and install it as trial software from Siemens Industry online support. You can then open the project. Note that the trial software can only be used for a limited period of time.

Use the following link to open the page in Siemens Industry online support, where the various software products are available to you as trial software: www.siemens.com/tia-portal-trial (www.siemens.com/tia-portal-trial)

Starting the TIA Portal

When you start the TIA Portal, Windows attempts to update the Certificate Revocation List (CRL) of "windowsupdate.com".

If no Internet access is available and there are multiple DNS servers, a timeout may occur and delay the start of the TIA Portal.

1.2 Notes on libraries

Contents

Information that could not be included in the online help and important information about product characteristics.

Comparing library elements

If master copies and types have the same name, the corresponding objects in the project are overwritten when the action "Copy" is used. Please note that this takes place without a prompt. The same behavior occurs when the name of the master copy is different to the name of the type, but an object within the master copy has the same name as the type.

1.3 Notes on memory cards

Contents

Information that could not be included in the online help and important information about product characteristics.

Notes on SIMATIC memory cards

The SIMATIC memory cards have been formatted and set up by Siemens for use with S7-1200 and S7-1500 modules. This format must not be overwritten; otherwise, the card will no longer be accepted by the modules. Formatting with Windows tools is therefore not permitted.

Behavior in case of open force job

Note that an active force job is retained even after you have loaded a new project to the SIMATIC memory card. This means you should first delete the active force job before you remove a SIMATIC memory card from the CPU and before you overwrite the card in the PC with a new project. If you use a SIMATIC memory card with unknown content, you should format the SIMATIC memory card before the next download.

Access protection for memory cards in USB card readers

By improving the security mechanisms for online access and engineering of S7-1500 CPUs, the data storage on memory cards has been changed. For this reason, this version of STEP 7 cannot evaluate the passwords of the configured protection level when reading project data from memory cards that is accessed via a USB card reader. The changed response affects the memory cards for CPUs of the S7-1200/1500 series. Therefore, use physical safeguards to protect critical project data on memory cards for these devices.

Note

This restriction is not related to online access to devices or the know-how protection of program blocks.

1.4 Notes on the hardware configuration

Content

Information that could not be included in the online help and important information about product characteristics.

Editing a device IP address

Do not use the address range from 192.168.x.241 to 192.168.x.250 when editing a device IP address. If necessary, this address range is automatically assigned by the system to a programming device. Depending on the subnet mask, this applies also for all network classes.

Drivers for the CP 5512 communications processor

The drivers for the CP 5512 are no longer included in this software version. If you want to use the CP 5512, the following needs to be installed prior to the installation of this software:

http://support.automation.siemens.com/WW/view/78453460 (http://support.automation.siemens.com/WW/view/en/78453460)

Please note that the CP 5512 can be used with a maximum of Windows 7 32-bit. As of Windows 8 or with 64-bit operating systems, the CP 5512 is not compatible.

Note

Since 2009, the functional successor the USB 2.0 module CP 5711 has been available.

1.5 Notes on using the TIA Portal in a virtual environment (private cloud)

Contents

Information that could not be included in the online help and important information about product characteristics.

Instructions for using the TIA Portal in a virtual environment (private cloud)

You can find instructions on how to use the TIA Portal in a virtual environment (private cloud) on the installation disk in the directory "Documents\Readme\<language directory>". You can open the PDF document "TIAPortalCloudConnectorHowTo<language ID>.pdf" here.

The TIA Portal Cloud Connector can be used with the Windows 7 (64-bit) and Windows 10 (64-bit) operating systems.

Environment variables "TiaUserSettingsPath" and "TiaDefaultProjectPath"

The environment variables "TiaUserSettingsPath" and "TiaDefaultProjectPath" can be used to set the location of the user and project settings different from the system default. This can be used, for example, to save the user and project settings at a central location.

You can find additional information on the use of environment variables in the Instructions on Using the TIA Portal in a Virtual Environment (Private Cloud).

Note on using self-signed certificates when using HTTPS as the communication protocol

Microsoft regularly checks whether the certificates in the Windows Certificate Store are from a trusted source. Certificates from untrusted sources are deleted. Therefore, ensure that your self-signed certificates are signed with a secure certificate.

1.6 Notes on using the TIA Portal Teamcenter Gateway

Contents

Information that could not be included in the online help and important information about product characteristics.

Information on the installation of the TIA Portal Teamcenter Gateway

You can find instructions for installing the TIA Portal Teamcenter Gateway in the TIA Portal Teamcenter Gateway online help under "Installing and uninstalling TIA Portal Teamcenter Gateway".

Information on the use of the TIA Portal Teamcenter Gateway

This readme file provides information on using the TIA Portal Teamcenter Gateway.

The installation of the TIA Portal Teamcenter Gateway gives you access to the TIA Portal Teamcenter Gateway function that is available as of TIA Portal V14 and higher.

Notes on installing the TIA Portal Teamcenter Gateway data model

You can find a description for installing the TIA Portal Teamcenter Gateway data model on the installation data medium in the directory "\Support\Teamcenter_11\ServerSetupDocument\" on the STEP 7/WinCC DVD2. Open the PDF document "Datamodel_Installation_enUS". The instructions for installing the TIA Portal Teamcenter Gateway data model are only available in English.

Note on updating the existing TIA Portal Teamcenter Gateway data model

Note the following:

- In the TIA Portal Teamcenter Gateway V15, no changes were made to the existing TIA Portal Teamcenter Gateway data model and no changes specific to the Teamcenter as well.

Notes on Teamcenter

Note the following:

- To work with the TIA Portal Teamcenter Gateway, Teamcenter Rich Application Client (RAC) or Teamcenter Client Communication System (TCCS) as of version 11.2 must be installed on your PG/PC.

Note on TCCS (Teamcenter Client Communication System)

Note the following:

- If you have completed the installation of TCCS (Teamcenter Client Communication System) as a "standalone" installation, you also need to install "Microsoft Visual C++ 2013 – Redistributable".

Note on UTF-8

Note the following:

- In order to display data from Teamcenter with non-English character sets, e.g. for Asian languages, in TIA Portal, Unicode character sets need to be used.

Note on cloud computing

Note the following:

- If you run TIA Portal in a Cloud (for example virtual machine), this system has to be integrated into the Teamcenter environment. Direct integration via a remote connection is not supported by TIA Portal Teamcenter Gateway.

Notes on the basics of working with TIA Portal Teamcenter Gateway

As of TIA Portal Teamcenter Gateway version V15, the Teamcenter operation "Teamcenter > Export..." is available for TIA Portal projects and global libraries.

With this operation, you create a copy of the current project, which is completely free of any Teamcenter information. If you need to work separated from Teamcenter, use this operation instead of "Save as" because "Save as" retains Teamcenter information in the project.

Setting the TIA Portal Teamcenter Gateway cache path

By default, the value entered under "Options > Settings > General > Teamcenter Gateway" in the "TIA Portal Teamcenter Gateway Cache Path" area is taken as "TIA Portal Teamcenter Gateway Cache Path".

By default, this value refers to the project directory path.

If you change the value of the "TIA Portal Teamcenter Gateway Cache Path", Teamcenter Gateway checks the validity of the values entered by default and the changed values.

Note

The default value entered cannot be empty.

Note on TIA Portal Teamcenter Gateway operations during a started WinCC Professional simulation

Do not perform Teamcenter Gateway operations, such as "Check in" or "Undo check out" during a started WinCC Professional Simulation.

1.7 Information on working with Multiuser Engineering

Contents

Information that could not be included in the online help and important information about product features of Multiuser Engineering and about the multiuser server.

Installation of the Multiuser Server gives you access to the corresponding "Multiuser Engineering" function that is available with TIA Portal.

Microsoft system account for multiuser server

When the multiuser server is installed, a Microsoft system account is created for the multiuser server. Microsoft changes the password cyclically for this account.

If the password for this account has expired, the password needs to be changed and then the multiuser server needs to be restarted.

Error message during start of the multiuser server

If an error message is displayed during or after the installation of the multiuser server indicating that the service cannot be started for the Multiuser Server, the following procedure is recommended:

1. Open the Windows event view and check whether an error entry with the error number "1069" is present.
2. If this is the case, information on how to proceed is provided below.
   Internet link: TIA Portal in Siemens Industry online support (https://support.industry.siemens.com/cs/ww/en/view/65601780)
3. Alternatively, you can contact Customer Support directly.
   Internet link to Customer Support: FAQ with ID 109739926 (https://support.industry.siemens.com/cs/ww/en/view/109739926)

Working with active monitoring jobs and forcing jobs in the local session

Active monitoring jobs and forcing jobs must be closed before a check-in or update in the local session. It is not possible to check in or update in the local session while monitoring or forcing jobs are active.

Working in the local session with the "Keep local session" option

If the "Keep local session" option is selected after the check-in, all checked-in objects in the local session appear as outdated, even if their content is identical.

Working with different versions of TIA Portal

Local sessions created with <x.y> version of TIA Portal can only be worked on with this version.

Example: A local session created with TIA Portal V15.1 for a V15.1 project cannot be edited with an earlier version of TIA Portal.

Working with Multiuser Commissioning

- When you work with Multiuser Commissioning, a new revision of the server project is created after each download to the CPU. However, a new revision is not created when objects that **cannot** be marked were edited in the local session. Changes to objects that cannot be marked are lost during check-in and are not downloaded to the CPU.
- When you start download to the device in Commissioning mode from the project tree with the menu command "Download to device > Hardware configuration", the dialog which informs you that different data exists on the CPU and in the server project is always displayed.
- When you work in Commissioning mode, only S7-1500 CPUs are fully supported. When using an S7-300/400 CPU, the dialog which informs you that different data exists on the CPU and in the server project is always displayed in Commissioning mode. The reason is that S7-300/400 CPUs do not support synchronization. To prevent this message from being displayed, you should disable the option "Check for data mismatch before loading (recommended)" in the Administration tool.

See also

TIA Portal in Siemens Industry online support (https://support.industry.siemens.com/cs/ww/en/view/65601780)

FAQ with ID 109739926 (https://support.industry.siemens.com/cs/ww/en/view/109739926)

1.8 Information about engineering options

Contents

Information that could not be included in the online help and important information about product characteristics.

What's new at SINAMICS in AutomationML

Contrary to the statement in the section "What's new in TIA Portal V15" in the section "SIMATIC TIA Portal Openness", V90 slaves and SINAMICS G120 PROFINET head modules cannot be exported and imported using AutomationML.

STEP 7 2

2.1 Security information

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit

https://www.siemens.com/industrialsecurity (https://www.siemens.com/global/en/home/company/topic-areas/future-of-manufacturing/industrial-security.html)

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under

https://www.siemens.com/industrialsecurity (https://www.siemens.com/global/en/home/company/topic-areas/future-of-manufacturing/industrial-security.html)

Network settings

The following tables show the network settings of each product you need to analyze the network security and to configure external firewalls:

STEP 7 Professional					
Name	Port number	Transport protocol	Direction	Function	Description
ALM	4410*	TCP	Inbound/outbound	License service	This service provides the complete functionality for software licenses and is used by both the Automation License Manager as well as all license-related software products.
RFC 1006	102	TCP	Outbound	S7 communication	Communication to the S7 controller via Ethernet/PROFINET for programming and diagnostic purposes.

STEP 7 Professional					
DCP	---	Ethernet	Outbound	PROFINET	The DCP protocol (Discovery and Basic Configuration Protocol) is used by PROFINET and provides the basic functionality for locating and configuring PROFINET devices.
SNMP	161	UDP	Outbound	PROFINET	The SNMP client functionality is used by STEP 7 to read status information from PROFINET devices.
* Default port that can be changed by user configuration					

WinCC ES Basic (without simulation)					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
ALM	4410*	TCP	Inbound/outbound	License service	This service provides the complete functionality for software licenses and is used by both the Automation License Manager as well as all license-related software products.
HMI Load	1033	TCP	Outbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.
* Default port that can be changed by user configuration					

Simulation RT Basic					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
HMI Load	1033	TCP	Inbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.
Ethernet/IP	44818	TCP	Outbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
	2222	UDP	Inbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
Modbus TCP	502	TCP	Outbound	Modbus TCP channel	The Modbus TCP protocol is used for connections to Schneider PLCs.
RFC 1006	102	TCP	Outbound	S7 channel	Communication to the S7 controller via Ethernet/PROFINET
Mitsubishi MC	5002	TCP	Outbound	Mitsubishi MC channel	The Mitsubishi protocol is used for connections to Mitsubishi PLCs.

2.2 Notes on use

Contents

Information that could not be included in the online help and important information about product characteristics.

Online operation

The simultaneous online operation of STEP 7 V5.6 or earlier and STEP 7 Professional V15.1 has not been approved.

Simultaneous online connections on an S7-1200 CPU (S7-1200)

It is not possible to establish an online connection from multiple instances of TIA Portal simultaneously to the same S7-1200 CPU.

Parallel use

If the term "Combo" appears in the name or license key of the software after installation, the use of the following products/versions is permitted in accordance with Article 1.6 of the General Terms and Conditions (see also setup text):

- STEP 7 Professional 2017
- STEP 7 Professional 2010
- STEP 7 Professional 2006

PLC migration

No instruction profile should be activated during PLC migration.

Deactivate your instruction profile before you perform the PLC migration. You can then reactivate the profile again.

Removing/inserting the memory card

After removing or inserting a memory card, always perform a memory reset on the CPU in order to restore the CPU to a functional condition.

Removing and inserting Ethernet modules

If Ethernet modules are removed and re-inserted during operation, you must boot the PC; otherwise, the "Accessible devices" functionality in STEP 7 or NCM PC will not display all devices. While the PC boots, Ethernet modules must be activated.

Loading project data with TIA Portal V12 and V13 (S7-1200)

If you load the project data of an S7-1200 CPU with TIA Portal V13, you can no longer use TIA Portal V12 to access this data. To do this, first restore the factory settings of the CPU. Read the additional information on this in the online help under "How to reset a CPU to factory settings".

Compatibility (S7-1200)

The device configuration and program of an S7-1200 CPU must always be configured with the same STEP 7 version. Usually, TIA Portal makes sure that no version conflicts occur by outputting appropriate notifications during loading to the device.

This automatic verification is not possible with S7-1200 CPUs with firmware version V1.x. In this case, users themselves must ensure that no version conflicts occur.

2.3 Editing devices and networks

2.3.1 General information on devices and networks

Content

Information that could not be included in the online help and important information about product characteristics.

S7-PCT IO-Link (S7-1200)

The S7 Port Configuration Tool is available for free download at the following link.

http://support.automation.siemens.com/WW/view/37936752 (https://support.industry.siemens.com/cs/ww/en/view/32469496)

Port Configuration Tool PCT

If you use the PCT tool, make sure that you always use the current version that is available free of charge with the following link:

http://support.automation.siemens.com/WW/view/32469496 (http://support.automation.siemens.com/WW/view/en/32469496)

The PCT tool is available for installation on the DVD in the "Support" folder.

2.3.2 Information about OPC UA functions

Contents

Information that could not be included in the online help and important information about product characteristics.

Schema files used for OPC UA functionality

The OPC Foundation provides its members with schemes that manufacturers can use for their respective server/client implementations. These files are used, for example, to validate imported information models / node sets.

The following scheme files are used for TIA Portal version V15.1 or S7-1500 CPU firmware version V2.6:

Scheme	Explanation
https://opcfoundation.org/UA/schemas/1.04/UANodeSet.xsd	XML scheme for NodeSet data
https://opcfoundation.org/UA/schemas/1.04/Opc.Ua.NodeSet2.xml	OPC UA ("CORE") Model from 2018-06-20
https://opcfoundation.org/UA/schemas/DI/1.1/Opc.Ua.Di.NodeSet2.xml	OPC UA for Devices ("DI") Model from 2018-06-06

PLC object not visible in the server address space

With very large projects, it can happen in rare cases that, despite successful compiling of the project and successful downloading to the CPU, the CPU node or PLC object is not visible in the address space of the OPC UA server. In this case the memory reserved for OPC UA in the CPU is insufficient.

Diagnostics, e.g. using UaExpert, returns the "Failed" server status.

Remedy

Disable the standard SIMATIC server interface in the CPU properties if it is not required.

Reduce the quantity or size of the elements that contribute towards the consumption of OPC UA resources. Reduce the file size to less than 250 MB and the number of nodes to fewer than 30000.

Examples:

- Reduce the number of tags in the address space of the OPC UA server, in particular of tags with the data type structure.
  To remove tags from the address space, disable the option "Accessible from HMI/OPC UA" in the PLC tag table or in the data blocks.
  With data blocks, you also have the possibility to remove the DB with all tags completely from the address space. To do this, disable the option "DB accessible from OPC UA" in the properties of the DB.
- Reduce the length of names of the elements in extensive data structures.
- Shorten the descriptions of the nodes.
- Reduce the number of implemented methods.

Resource bottleneck during connection with fast establishment and termination of connections

Error 0x80810000 (OpcUa_BadTcpNotEnoughResources) can occur during connection establishment (OPC_UA_Connect) of the client CPU, although free connection resources are still temporarily displayed.

Possible cause

During the termination of an OPC UA connection (OPC_UA_Disconnect), the connection resource used for this connection remains occupied for approx. 60 seconds.

This behavior is also called "Graceful Shutdown" and conforms to TCP RFC 793.

The following applies to the period of the Graceful Shutdown:

- The TIA Portal ("Diagnostics > Connection information" area) counts these resources that are still occupied as free resources, so that you may not diagnose a resource bottleneck even though there is a resource bottleneck.
- The instruction to terminate an OPC UA connection (OPC_UA_Disconnect) indicates that the connection has been terminated with DONE=1.
  However, the connection resource in the CPU remains occupied for approx. 60 seconds after the transition DONE=0 => DONE=1.

Remedy

The described behavior quickly has an effect on connection and disconnection: Correct termination of a connection still binds the connection resource used for 60 seconds.

If a client initiates connection establishment again in this phase, an additional resource is used, since the resource of the disconnected connection is still being used. If connection and disconnection occur in quick succession in your application and the above message occurs, the frequency of this sequence should therefore be reduced if possible.

Description of the OPC UA instructions OPC_UA_ServerMethodPre and OPC_UA_ServerMethodPost

Declaration of the tags in the description of the "OPC_UA_ServerMethodPre" instruction: Preparation of the server method call

Contrary to the tag declaration (table) described in the help for this server instruction, you cannot declare an InOut tag "UAMethod_OutParameters". Otherwise, the method call is rejected with an error message.

Declaration of the tags in the description of the "OPC_UA_ServerMethodPost" instruction: Post preparation of the server method call

Contrary to the tag declaration (table) described in the help for this server instruction, you cannot declare an InOut tag "UAMethod_OutParameters". Otherwise, the method call is rejected with an error message.

Example program for OPC_UA_TranslatePathList

In the example program for OPC_UA_TranlatePathList, the call of the instruction "OPC_UA_NamespaceGetIndexList" (#OPC_UA_NamespaceGetIndexList_Instance (...)) is incorrect because not all required variant parameters are supplied.

The following applies to all client instructions: All required variant parameters must be supplied.

These are the parameters "NamespaceUris" and "NamespaceIndexes" in the case of the "OPC_UA_NamespaceGetIndexList" instruction used.

Restart of the OPC UA server

The OPC UA server is stopped at each download to the CPU (e.g. after downloading a configuration or a block) and then restarts.

Please note that OPC UA clients cannot continue to use the running services for this server in this case and that the resources therefore need to be released again.

These services need to be requested again:

- Re-initiate connection setup
- Regenerate subscriptions
- Recreate read/write jobs

2.3.3 Hardware detection of IO devices

Contents

Information that could not be included in the online help and important information about product characteristics.

Incomplete hardware recognition at ET 200SP stations

With STEP 7 V15 you can use the online function "Hardware detection" to determine the station configuration of IO devices and to import them in your project including all plugged modules.

If you import an ET 200SP via the hardware detection in your project, you have to complete the determined station configuration manually yourself:

- Check and correct the configured potential group properties (light or dark BaseUnit) for each slot. STEP 7 cannot determine through the hardware detection which BaseUnit actually exists on which slot.
- Check at the interface module whether the suitable bus adapter was determined in the hardware configuration (for example BA 2xRJ45 instead on the BA 2xFC really present).
- At a mixed configuration ET 200SP/ET 200AL with bus adapter BA-Send the ET 200AL distributed modules are not recognized. Complete the station configuration manually, meaning supplement the missing modules from the hardware catalog.

Modules/submodules with ET 200S (GSDML) are not detected

Although the following IO devices of the ET 200S (GSDML) are taken into account during hardware detection, their plugged-in modules/submodules are not recognized. If such IO devices are added to the project, any modules/submodules that may be used must be subsequently taken from the HW catalog and configured.

Affected are:

- 6ES7 151-3AA00-0AB0 (with firmware V1.0)
- 6ES7 151-3AA10-0AB0 (with firmware V2.0)
- 6ES7 151-3BA00-0AB0 (with firmware V2.0)
- 6ES7 151-3AA20-0AB0 (with firmware V3.0)
- 6ES7 151-3BA20-0AB0 (with firmware V3.0)

2.3.4 HW ID of type "Hw_SubModule" for an HMI Panel IO device

Contents

Information that could not be included in the online help and important information about product characteristics.

Determination of the HW ID with the "Hw_SubModule" type for an HMI Panel IO device

The "ID" parameter of the PROFIenergy "PE_START_END" program block must be supplied with the hardware identifier (HW ID) with the type "Hw_SubModule" of the IO device. The hardware ID with the "Hw_SubModule" type of the HMI Panel IO device is not displayed in the system constants on an S7-1500/S7-1200/ET200SP CPU, but it can be determined. To do this, the symbol for the IO device in the HMI panel with the type "Hw_Device" is determined in the list of PLC tags in the "System constants" tab and reduced by one, for example, "HMI_1_IE_CP_1~PROFINET_Interface_1~IODevice -1".

2.3.5 Use of modules on the S7-1200

Content

Information that could not be included in the online help and important information about product characteristics.

Use of modules on the S7-1200

The modules listed below are not supported on the S7-1200.

Family	Module	Order number
S7-300 FMs	SM 338	6ES7 338-4BC01-0AB0
	FM 350-1	6ES7 350-1AH03-0AE0
	FM 350-2	6ES7 350-2AH00-0AE0, 6ES7 350-2AH01-0AE0
	FM 351	6ES7 351-1AH01-0AE0, 6ES7 351-1AH02-0AE0
	FM 352	6ES7 352-1AH02-0AE0
	FM 355 S	6ES7 355-1VH10-0AE0
	FM 355 C	6ES7 355-0VH10-0AE0
	FM 355-2 C	6ES7 355-2CH00-0AE0
	FM 355-2 S	6ES7 355-2SH00-0AE0
S7-300 PtP-CP	CP 340	6ES7 340-1AH02-0AE0, 6ES7 340-1BH02-0AE0, 6ES7 340-1CH02-0AE0
	CP 341	6ES7 341-1AH01-0AE0, 6ES7 341-1AH02-0AE0, 6ES7 341-1BH01-0AE0, 6ES7 341-1BH02-0AE0, 6ES7 341-1CH01-0AE0, 6ES7 341-1CH02-0AE0

Family	Module	Order number
Network component	Diagnostics repeater	6ES7 972-0AB01-0XA0
ET 200M	SIWAREX	7MH4900-2AA01, 7MH4900-3AA01, 7MH4950-1AA01, 7MH4950-2AA01

Loading S7-1200 module comments to the PG/PC

In central configurations with S7-1200, comments of modules, submodules and signal boards are not loaded. With CPs/CMs, only the comments of the IE interface or DP interface are loaded. In distributed configurations with ET 200SP or ET 200MP, only the comment of the channels is loaded from the I/O modules.

2.3.6 CP 343-2 on SIMATIC S7 Embedded Controller EC31-RTX

Contents

Information that could not be included in the online help and important information about product characteristics.

CP 343-2 on SIMATIC S7 Embedded Controller EC31-RTX

The module AS-Interface CP 343-2 (article no.: 6GK7 343-2AH01) can be inserted in an expansion rack of the SIMATIC S7 Embedded Controller EC31-RTX (article no.: 6ES7 677-1DDxx-0BB0), but the CP 343-2 cannot be operated with the EC31-RTX.

2.3.7 Loading of S7-1500 CPUs (SIPLUS)

Contents

Information that could no longer be included in the online help and important information about product features.

Behavior of a SIMATIC S7-1500 CPU SIPLUS (6AG1...) FW <V2.5 with formatted memory card

Components involved:

- CPU 151x SIPLUS (6AG1...) with firmware version prior to V2.5
- SIMATIC memory card

If you load a SIPLUS CPU to a formatted SIMATIC memory card and then set the CPU to RUN, the CPU runs without errors. But once the power supply has been switched off and on again or if a memory reset (MRES) has been performed, the SIPLUS CPU goes into STOP with an error message. The error message states that the memory card is not recognized because it is missing, is of an incorrect type, contains incorrect data or is protected.

This behavior does not occur with a CPU 151x SIPLUS (6AG1...) with firmware version V2.5 or higher or if a configuration with a S7-1500 SIMATIC CPU (6ES7...) already exists on the memory card and is overwritten by a configuration with a SIPLUS CPU (6AG1...) of the same type.

Remedy:

1. A configuration with SIMATIC CPU must be loaded on the SIMATIC memory card. Only a CPU of the corresponding type must be available; the I/O is not required in this case.
2. The SIPLUS CPU is loaded to the card. The existing SIMATIC configuration on the memory card is overwritten with the SIPLUS configuration.

This procedure prevents an error message of the TIA Portal.

Note

The procedure ONLY works for online loading to the CPU. The error remains if the card is programmed by means of an external card reader.

If you want to load a CPU 1511-1 PN SIPLUS (6AG1 511-1AK00...), for example, do the following.

1. Insert a blank memory card with sufficient available memory into the SIPLUS CPU.
2. Configure a SIMATIC CPU of the same type, here an S7-1511-1 PN (6ES7 511-1AK00...). I/O modules are not required for the first load step.
3. Load the configuration onto the SIPLUS CPU.
4. Replace the SIMATIC CPU in STEP 7 with the SIPLUS CPU (6AG1 511-1AK00...) and complete the configuration with the planned I/O modules and distributed I/O devices.
5. Load the changed configuration onto the SIPLUS CPU.

2.3.8 ET 200MP (IM 155-5 DP ST) on a WinLC RTX

Contents

Information that could not be included in the online help and important information about product characteristics.

ET 200MP (IM 155-5 DP ST) can currently not be operated on a WinLC RTX via PROFIBUS.

Components involved:

- PC systems with logic controller functionality WinLC RTX as DP master (all versions: EC31-RTX (F), WinAC RTX (F) software PLC), referred to as "WinLC RTX" below.
- ET 200MP (IM 155-5 DP ST) as DP slave

If you configure standard modules or F-modules in an ET 200MP and use a WinLC RTX as DP master, these modules are not configured by the DP master in the ET 200MP. After loading the configuration and startup of the components involved, the modules in the ET 200MP signal the missing parameter assignment by means of flashing green LEDs.

STEP 7 does not show the readiness for operation of the modules online (status "OK").

Remedy:

- Recommendation: Use an IM 155-5 PN interface module to connect the ET 200MP to a WinLC RTX via PROFINET IO.
  or
- If no F-modules are inserted in the ET 200MP: Use the GSD version of the ET 200MP on an WinLC RTX to configure the standard modules.

2.3.9 Compiling ET 200SP HF V2.1 and higher, ET 200SP HS V4.0

Contents

Information that could not be included in the online help and important information about product characteristics.

Compilation error for ET 200SP HF V2.1 and higher, ET 200SP HS V4.0

Affected components:

- ET 200SP HF V2.1 and higher
- ET 200SP HS V4.0

A compilation error can occur in isochronous mode of the ET200SP (ET 200SP HF V2.1 and higher, ET 200SP HS V4.0) with the setting "From OB" even if the settings are valid. The typical error message is: "The specific Ti value is invalid" or "The specific To value is invalid". But other error messages are possible as well.

Remedy:

Upgrade the module description of the ET200SP header module in this case. You can upgrade the module description of the ET 200SP header module in the network view or in the device view of the Inspector window with the function "Update module description". The error can still occur with the current module description after the first compilation. If you have selected valid settings, the error will no longer occur with the subsequent compilation.

2.3.10 Notes on online and diagnostics

Contents

Information that could not be included in the online help and important information about product characteristics.

Hardware detection followed by online connection

When the "Online > Hardware detection" command is performed for an unspecified CPU, the online configuration is not loaded from the CPU. If you do not load the configuration resulting from the hardware detection to the CPU, the device and network views will always show a difference between the offline and online configurations. It will appear that there are different configurations in the online and diagnostic views, although the MLFBs are identical in the actual CPU and the offline CPU.

Deleting measurements in the device (memory card) (S7-1200)

Do not delete any measurements in the device as long as a trace in the device still generates new measurements on the memory card.

2.3.11 Network components

2.3.11.1 Network and communication modules

Online connections for special diagnostics and firmware loader

Special diagnostics: Establishing an online connection

Some network components (S7-300/400 CPs, PC CPs, gateways) provide expanded diagnostics data in special diagnostics.

Requirement:

A PG/PC interface is physically connected to the destination device, e.g. via an Ethernet cable.

Procedure:

Follow the steps outlined below to start special diagnostics:

1. Open the online and diagnostics view of the module.
2. In the "Functions" folder, select the group "Special diagnostics".
3. In the "Special diagnostics" area, click the "Start special diagnostics" button.

As of STEP 7 Professional V14 for several CPs only one online connection can be established from the engineering station via the CP. This connection can be established either with STEP 7 or special diagnostics.

If special diagnostics is not started by the steps listed above and an error message is output follow the steps below:

1. Disconnect the existing online connection to the station.
2. Open the online and diagnostics view of the module again as described above and start special diagnostics.

32-bit applications; NCM S7 diagnostics, firmware loader

If you have established a connection to a PC CP from a 32-bit application and want to establish another online connection via the same interface of the PC CP, you must first terminate the connection of the 32-bit application.

Special diagnostics and firmware loader: Chinese GUI

When you open the special diagnostics or the firmware loader with a Chinese GUI, you need to set the language for non-Unicode applications in the operating system to Chinese.

S7-CM/CP

No S7 routing with CP 300/400 < V3 as router

With S7 routing between S7-1200 or S7-1500 stations, no connection is established when older S7-300/400 CPs are connected in between as routers. The following types of communication that run over ISO transport or ISO-on-TCP connections are affected:

- S7 communication
- PG communication
- HMI communication

The behavior applies to S7-300/400 CPs with a firmware version < V3.0 for which the TSAP length is limited to 2 bytes.

In contrast, like the S7-1200/1500, S7-300/400 CPs as of V3.0 support long TSAPs and thus also S7 routing.

CM 1542-1: Remove retentive storage of the IP address if there are duplicate addresses

The IP address and the device name of the CM 1542-1 remain installed retentively:

If, for example during startup, the CM detects a duplicate address in another network, the CM is not connected to the network. The CM changes to RUN and cannot be reached via the Ethernet interface.

To be able to connect the CM to the network, you can remove the retentively stored IP address as follows:

1. Remove the memory card of the CPU.
2. Using DCP with the Primary Setup Tool (PST) and with the CPU in STOP, set the IP address of the CM to 0.0.0.0 without configuration.
   You have removed the retentively stored IP address of the CM. The CM can be connected into the network.
3. Insert the memory card in the CPU again.

PC CPs and applications

PC CPs for VMware vSphere Hypervisor ESXi

The following communications processors have been released for operation with VMware vSphere Hypervisor ESXi for STEP 7 Professional V14:

- CP 5711
- VMware E1000 network adapter

The following communications processors have not been released for operation with VMware vSphere Hypervisor ESXi:

- CP 5611
- CP 5612
- CP 5621
- CP 5622
- CP 5512

Other communications processors such as the CP 1623 and CP 1628 are released following installation of the SIMATIC NET PC software. You will find information on this in the readme file of the SIMATIC NET PC software.

Download to PC stations

If you download project data to a PC station with PC-CP, make sure that the CP is available.

In the "Extended download to device" dialog, do not enable the check box for the search for compatible nodes.

If you want to download to a different CP from the one selected earlier in the project, change the IP address of the CP manually in the "Extended download to device" dialog in the list of configured access nodes.

PROFIBUS S7 connections with OPC servers

The configuration of an S7 connection via PROFIBUS is not possible between an OPC server as of V12 and an S7-1200 as of V4 or an S7-1500.

Remedy: Use an OPC server / PC application < V12 or use an S7-1200 < V4 or an S7-300 or S7-400 as the S7 station.

Visibility of PLC tags for OPC UA communication

If in the S7 communication between a PC station and an S7 station you configure PLC tags with the option "Reachability from HMI/OPC UA" deactivated, this does not affect the visibility in the OPC UA server of the PC station.

Access rights of an OPC client for access to PLC tags

If you want to write PLC tags of S7 stations that are provided via S7 communication in the OPC UA server of a PC station from an OPC client, make sure that the following settings are made in the configuration of the tags:

- The option "Can be written from HMI/OPC UA" is activated for the PLC tags.
- The access right "Read/Write" is activated in the symbol configuration of the OPC UA server application of the PC station.

Security modules

Migration of projects with Ethernet CPs and activated security functions

In STEP 7 V5.5 projects that contain Industrial Ethernet CPs with activated security functions, the security settings are disabled during migration to STEP 7 Professional.

If necessary, follow these steps after the migration:

1. Activate the security functionality.
2. Configure the required security settings.

Migration of IP access protection lists when activating the security functions

An active IP access protection list is converted to firewall rules when security functions are activated. These rules are visible in advanced firewall mode and can be adjusted there. The advanced firewall mode is activated automatically.

Security online diagnostics of S7 CPs

Security online diagnostics of a CP with security capability is only possible if the online connection is established directly via the CP. If the online connection was established by STEP 7 via the CPU to the station, with the "Connect online" button on the security diagnostics page "Security" > "Status" of the security CP you can establish a direct connection to the CP to run security online diagnostics. As an alternative, you can also terminate the online connection to the CPU and enter the IP address of the CP in the "Station address" input box under the entry "Online access" in online diagnostics.

IKE mode

When negotiating the key in phase 1 the IKE mode "Main" should be preferred. This mode normally provides a reliable procedure compared with the "Aggressive" mode. One reason for using the "Aggressive" setting is when you have VPN groups with different pre-shared keys.

The IKE "Aggressive" mode should not be used in conjunction with certificates. In the IKE "Aggressive" mode, only use pre-shared keys.

A security module must not be used in VPN groups that use different IKE modes.

Downloading configuration data to an S7-300/400 via a VPN tunnel

When you download configuration data via the gigabit interface of a CP x43-1 Advanced to an S7-300/S7-400 station, the path via which the download takes place is stored in the project. If the project is then downloaded via a VPN tunnel established between a SCALANCE S module and the CP x43-1 Advanced, the download fails due to the changed path.

To download via the VPN tunnel, follow the steps below:

1. Using the "Go online" button, connect the engineering station to the gigabit interface of the CP x43-1 Advanced.
2. Disconnect the online connection to the CP x43-1 Advanced.
3. Download the project to the station via the gigabit interface of the CP x43-1 Advanced.

VPN tunnel establishment with 1200/1500 CPs capable of VPN

The establishment of VPN tunnel connections by 1200/1500 CPs capable of VPN with the authentication method "pre-shared key" is only possible if the VPN connection partner is also a 1200/1500 CP capable of VPN. The establishment of VPN tunnel connections by a 1200/1500 CP capable of VPN to all other VPN connection partners is only possible with the "certificate" authentication method.

Security configurations from old projects

Before a security configuration from an older STEP 7 version in STEP 7 V14 can be used, a user with administrator rights needs to log in to this security configuration in STEP 7 V13 SP1 and save the project. After opening the project in STEP 7 V14, a user must log on again to the security configuration with administrator rights. Thereafter, it is possible to use this security configuration in STEP 7 V14.

Copying security modules between STEP 7 instances

Copying security modules between different instances of STEP 7 is not supported.

Device certificates for CP 1543-1 V2.0 and CP 1543SP-1

For CP 1543-1 V2.0 and CP 1543SP-1, only those device certificates that are assigned to the module via the local certificate manager as device certificates are loaded to the module. This assignment takes place in the local security settings of the module in the "Certificate manager" entry via the "Device certificates" table editor. The certificates of the global certificate manager are available for the certificate assignment.

No fallback VPN profiles for CP 1200/1500

Fallback VPN profiles are not supported by the CP 1200/1500. Configured fallback VPN profiles are therefore not loaded on the CP 1200/1500.

Media converters

When you migrate a project, you need to reconfigure the media converters.

2.3.12 Notes on the S7-1500R/H redundant system

Content

Information that could not be included in the online help and important information about product characteristics.

S7-1500R/H redundant systems

GRAPH blocks, ProDiag blocks and the "Program_Alarm" instruction are not supported in V15.1 of S7-1500R/H-CPUs. If your program contains one of these elements, a compilation message notifies you of this.

You can find additional information under "Principle of operation of the redundant S7-1500R/H system" and "Programming S7-1500R/H CPUs" in the information system and in the section "Basics of program processing" in the S7-1500R/H system manual.

2.4 Programming a PLC

2.4.1 General notes on PLC programming

Contents

Information that could not be included in the online help and important information about product characteristics.

Compiling blocks

If the message "The network is not supported in this version" is displayed, use a TEMP tag of the BOOL data type to distribute multiple assignments that are not connected to the first Boolean input of the instruction across several networks.

Instance-specific attributes when importing blocks via TIA Openness

In certain situations, the import rules can mean the loss of instance-specific attributes, such as start values.

Display of the CPU name in system alarms

As of V15.1, the name of the CPU (or for H or R CPUs, the name of the station) is displayed in the "Additional text 1" column of the system alarms. To do this, you need to compile the hardware of the device in question. If the CPU name is not displayed via the "Compile > Hardware (only changes)" option, use the option "Compile > Hardware (rebuild all)".

Make sure that loading to the device is actually performed; the message that the hardware is up-to-date is not enough!

Modifying the application cycle of an isochronous cycle interrupt OB

When you change the "Application cycle (ms)" parameter of an isochronous cycle interrupt OB, you implicitly also modify the hardware configuration. To ensure that the modified value for the application cycle duration becomes effective after the next loading process of the CPU you have to explicitly include the hardware configuration in loading. To this purpose select either the menu command "Load to device > Hardware and software" or the command "Load to device > Hardware configuration".

Information for setting breakpoints in the standard user program at S7-1500 F-CPUs

If a breakpoint is enabled and is reached, the CPU immediately goes to "STOP" after "HOLD". If you want to change back to RUN mode after the "HOLD" in order to continue testing your default user program, you can simulate this with S7-PLCSIM.

Further information about testing with breakpoints at F-CPUs is available in the "SIMATIC Safety – Configuring and Programming" manual.

Using the system data type "TDiag_StatusExt" (S7-1200/1500)

Alternatively to the "TDiag_Status" system data type, the extended system data type "TDiag_StatusExt" can be used to obtain more detailed information about the cause of a lost connection. In particular, the reading out of the structure element "LastConnErrTimeStamp" causes the CPU to switch to "Defect" system state and must not be used currently.

Total size of the associated values for ProDiag (S7-1500)

The alarm procedure of the S7-1500 CPU allows a maximum of 512 bytes for associated values. For internal information, ProDiag requires part of this associated value total. This means that the three possible associated values in the specific text field can use a maximum of 400 bytes.

This value was not validated in TIA Portal V14 during compilation. As of TIA PortalV14 SP1, there is a corresponding validation, which returns an error if this value is exceeded.

This must be taken into account during migration of projects from version 14 to version 14 SP1.

Upgrading know-how-protected blocks (S7-1200/1500)

After an upgrade from V13 SP1 to a version <V15, know-how-protected devices that have been set up with a version >=V15 can be loaded in the controller without entering a password and are executable. To edit a know-how-protected block in >=V15, remove the know-how protection and set it again. The block can then be renumbered and renamed in know-how-protected state.

- For know-how protected blocks that were set up with V13 SP1:
  - To edit the know-how-protected block in >=V15, remove the know-how protection and set it again. The block can then be renumbered and renamed in know-how-protected state.
- For know-how protected blocks that were set up PRIOR TO V13 SP1:
  - To edit, remove the know-how protection in V13 SP1 and reset it when done.

Upgrading of GRAPH blocks (S7-300, S7-400, S7-1500)

After upgrading a project to V15.1 from a version lower than V15, check whether repaired GRAPH blocks are listed in the log file and test whether these blocks are executable in your program.

If a project contains GRAPH blocks and is upgraded from V15.0 to V15.1, the program blocks have to be compiled before they can be monitored.

Information about network security

For communications access between TIA Portal and CPU or between HMI (except for HMI access using "GET/PUT communication") and CPU, there are integrated security functions. These provide greater protection from manipulation and higher access protection. To protect against unauthorized network access to a CPU with standardized communications access such as "GET/PUT", "TSEND/TRCV", "Modbus", "FETCH/WRITE", you should also take suitable additional measures (e.g. cell protection concept).

Functions from the global library "Long Functions" with PLCSIM (S7-1500)

The compilation process links the functions of the S7-1500 CPU family to the CPU to protect them from manipulation. Function from the global library "Long Functions" that are used in the CPU and have such a link cannot be loaded to a simulation. However, it is possible to replace such elements with their originals from the global library. These are not linked to a CPU.

Loading inconsistent programs to an S7-1500 CPU (S7-1500)

In TIA Portal, it is not possible to download inconsistent programs to an S7-1500 CPU without a consistency check. During the loading process, all blocks of the program are implicitly checked and are compiled again in the event of inconsistencies. If, however, there are programs on your CPU which were loaded with earlier versions of STEP 7, these programs could demonstrate inconsistencies.

In this case, note the following:

If you load an inconsistent program from a device, you will not be able to load the program back to an S7-1500 CPU unchanged afterwards, because a consistency check always takes place during the loading process and existing inconsistencies are corrected.

Changed access rights for displaying the comparison status (S7-1200/1500)

Valid for:

- CPU 1500 < V2.0 (as well as ET200 SP and Software Controller)
- CPU 1200 V4.0 and V4.1

As of TIA Portal V14, the required access rights for displaying the comparison status of blocks have changed for the specified CPUs from the S7-1200/1500 series.

Previously, you only needed the "HMI access" access level for the listed CPUs to display the comparison status. As of V14, you need the "read access" access level to display the comparison status.

This change means that you are prompted to enter the password for "read access" when you go online.

If you only have the password for "HMI access", you can still establish an online connection.

In this case, click "Cancel" in the password prompt for read access and enter the password for "HMI access" in the subsequent dialog. The online connection is then established with the available "HMI access rights", but no comparison status of the blocks is displayed.

Only questions marks are displayed in place of the icons for the comparison status.

Valid for:

- CPU 1200 <=V3.x.

As of TIA Portal V14, there is now a password prompt for the above specified CPUs when you go online for displaying the comparison status of blocks, provided that these CPUs are configured with write protection and/or read protection.

If you do not know the password for access to read-protected blocks on a protected CPU, you can still establish an online connection.

In this case, leave the displayed password prompt dialog with "Cancel".

The online connection is still established, but no comparison status of the blocks is displayed.

Only questions marks are displayed in place of the icons for the comparison status.

Data exchange between standard and F-program when using Team Engineering

To ensure data consistency between standard and safety program, you should:

- Do not exchange data via bit memory, use global data blocks instead.
- Limit access between safety programs and standard user programs to two standard data blocks.

You can find additional information on this in section 5.13 of the Programming Guidelines under: https://support.industry.siemens.com (https://support.industry.siemens.com/cs/document/81318674/programming-guideline-and-programming-styleguide-for-s7-1200-and-s7-1500?dti=0&lc=en-WW)

Compare offline/online

Corrections to internal project data have been made in the current version in order to enhance data integrity. After installation, this enables individual blocks to be corrected automatically upon opening, which is why the offline/online comparison displays different checksums.

Online/offline differences in the project tree (S7-1200 FW V2.0 and V2.1)

If you use the "WRIT_DBL" instruction to change a data block, the resulting difference between online and offline block is initially not displayed correctly by the symbols in the project tree. The difference is only shown when you terminate an online connection and then go back online.

2.4.2 Notes on instructions

Contents

Information that could not be included in the online help and important information about product characteristics.

Instruction "TSEND_C: Establishing a connection and sending data" for S7-1200 and S7-1500 CPUs

Contrary to the information in the online help, the value 7002 of the "STATUS" parameter has the following meaning:

- STATUS=7002: Connection is currently being established (REQ irrelevant)

Instruction "TRCV_C: Establishing a connection and receiving data" for S7-1200 and S7-1500 CPUs

Contrary to the information in the online help, the value 7002 of the "STATUS" parameter has the following meaning:

- STATUS=7002: Connection is currently being established (REQ irrelevant)

Instruction "GetSMCinfo: Read information about the SIMATIC memory card"

In addition to the values described in the online help, the "Status" parameter can also have the following value:

- 8091: The selected value of the "Mode" parameter is only permitted for S7-1500-R/H CPUs.

Instruction "RH_CTRL: Influence sequences in R/H systems"

Contrary to the information in the online help, the values 3 and 4 for the "Mode" parameter have the following meaning:

- Mode=3: Disable execution of SYNCUP
- Mode=4: Enable execution of SYNCUP again

2.4.3 Note on software units

Contents

Information that could not be included in the online help and important information about product characteristics.

Using software units for shared commissioning (S7-1500)

Note that only blocks can be synchronized with shared commissioning. Relations are ignored during the synchronization process.

Manual comparison of software units (SIMATIC S7-1500)

When comparing software units manually, the property comparison overview icon does not always display the correct result. However, the comparison results of the individual source and target data are correct.

2.4.4 Testing the user program

2.4.4.1 Testing with the watch table

Contents

Information that could not be included in the online help and important information about product characteristics.

Multiple access to the same CPU (S7-300/400/1200)

Online access to a CPU (S7-300/400/1200) from a PG/PC is permitted only when a TIA Portal is open. Multiple access to the same CPU of this family is not permitted and can lead to errors.

Loading data blocks during an active control job

Note

Loading changed data blocks during an active control job can result in unforeseen operating states. The control job continues to control the specified address, although the address assignment may have changed in the data block. Complete active control jobs before loading data blocks.

"Enable peripheral outputs" function

The function "Enable peripheral outputs" in not available for CPUs of the S7-1500 series in the TIA Portal.

This function can only be executed with an S7-300, S7-400 or S7-1200 CPU in the TIA Portal.

Testing programs converted from older STEP 7 versions

To monitor and test a program converted from an older STEP 7 version, you first need to compile and download the program with the current STEP 7 version.

Changed display formats in the watch table

The number of display formats has been reduced so that display formats that are used but no longer supported are reset to the defaults during a project upgrade.

Example: Pointers can now only be shown in "Pointer" or "Hexadecimal" format.

2.4.4.2 Testing with the force table

Contents

Information that could not be included in the online help and important information about product characteristics.

Forcing tags for direct I/O access (S7-300)

If you use direct I/O access for an S7-300 CPU in your user program, forcing this I/O address is not permitted.

Example

If I/O access to the address "IB0:P" takes place in the user program, it is not permitted to force the following I/O address areas: I0.0:P, IB0:P, IW0:P and ID0:P.

Changed display formats in the force table

The number of display formats has been reduced so that display formats that are used but no longer supported are reset to the defaults during a project upgrade.

Example: Pointers can now only be shown in "Pointer" or "Hexadecimal" format.

2.4.5 Compatibility of PLC programs from older versions

2.4.5.1 Compatibility of PLC programs from versions prior to V15.1

Contents

Information that could not be included in the online help and important information about product characteristics.

Compatibility

Basically, you can continue to use all PLC programs that were created with older versions of TIA Portal in V15.1. However, because improvements were made to the compiler and errors corrected there in V15.1, in rare cases the program may react differently after the upgrade or you may have to adapt the program code manually. These cases are described in detail below.

Processing sequence of function calls in SCL (S7-1200/S7-1500)

In older versions of the TIA Portal, the processing sequence of assignments containing expressions or function calls was not always clear. Processing was not carried out consistently from right to left; rather, in certain cases, parts of the expression on the left side were evaluated first.

This behavior was corrected in V15.1 so that the right side of the assignment is always calculated first and then the left side is determined.

Example:

```
#myArr["FC_1"()]:= #x + #y;
```

In earlier versions, the "FC1" was run here first in order to determine an ARRAY element. Then, #x + #y was calculated.

As of V15.1, the program is now correctly processed from right to left: First, #x + #y is calculated and then "FC1" is run.

This correction can have the result that your program behaves differently after upgrade, e.g. when the tags #x or #y are set in "FC1". Therefore, please check your program after upgrading.

Comparators (SCL S7-1200 >= 4.2 and S7-1500 >= 2.0)

When comparing two instance data blocks via comparison instructions in SCL, not the structure of the blocks but only the block numbers are compared. As of V15.1, an error message during compiling notifies you that this instruction may not return the desired result (FALSE is generally returned because the numbers of the blocks are different).

Example:

The result of the following comparison instruction is FALSE even if the structure of the two blocks is identical.

```
IF "MyDBofFB" = "MyDBofSameFB" THEN …
```

Importing IEC timers (S7-1200/S7-1500)

IEC timers contain read-only parameters, e.g. ET (current time value). In older versions of the TIA Portal, the start values of these read-only parameters were overwritten in some cases when external source files were imported. This behavior was corrected with V15.1: Start values for read-only parameters are now ignored when source files are imported. Instead, the default values of the data type are used. Therefore, please check your program after an import.

Replacing invalid Unicode characters during upgrade

Invalid Unicode characters in your program are corrected when upgrading to V15.1. For example, invalid characters in string constants are replaced with the escape sequences defined by IEC. This can change the checksum of the program and indicate differences between the online and offline program. Compile your program and reload it in this case.

Instructions "SCATTER" / "SCATTER_BLK" and "GATHER" / "GATHER_BLK" (SIMATIC S7-1200/1500)

You can use the "SCATTER" / "SCATTER_BLK" instructions to parse the bit sequence into individual bits. You can use the "GATHER" / "GATHER_BLK" instructions to combine individual bits into a bit string.

The values with which the instructions work must not be in the I/O area or in the DB of a technology object, since data consistency cannot be guaranteed in these areas.

As of V15.1, this rule undergoes a more precise syntax check. If you have the above-mentioned instructions with invalid input parameters in your program, an error is now reported during compiling.

As a remedy, you can first copy the tags to a temporary memory area and transfer them from there to "SCATTER" or "GATHER" instructions.

Instructions "BLKMOV" / "UBLKMOV" and "FILL" / "UFILL"

Use the "BLKMOV" / "UBLKMOV" and "FILL" / "UFILL" instructions to copy the contents of a memory area (source range) to another memory space (target range). If these instructions access a memory area that does not exist, a programming error is output. In previous versions, in rare cases an error code was generated at the "RET_VAL" parameter instead of a programming error. This behavior has been corrected for S7-1500 CPUs as of firmware version 2.6.

"RUNTIME" instruction (SCL/STL S7-1200/S7-1500)

The "Mem" parameter of the "RUNTIME" instruction has the data type "LREAL" and can only be used with a tag of this data type. Implicit conversion is not possible here. In V15.1, this rule undergoes a more precise syntax check. If you have provided the "Mem" parameter of the "RUNTIME" instruction with an invalid data type in your program, an error is now reported during compiling.

2.4.5.2 Compatibility of PLC programs from versions prior to V15

Contents

Information that could not be included in the online help and important information about product characteristics.

Compatibility

In principle, you can continue to use in V15 all PLC programs that were created with older versions of TIA Portal. However, because improvements were made to the compiler and errors corrected there in V15, it can occur in rare cases that the program reacts differently after the upgrade or that you have to adjust the program code manually. These cases are described in detail below.

S7-GRAPH: Implicit conversions "Hardware data type <> INT"

As of TIA Portal V15 it is no longer possible to program implicit conversions of hardware data types (for example INT -> DB_ANY or DB_ANY -> INT), as is already the case in LAD, FBD, STL and SCL.

Please use either the UINT data type or an explicit conversion instead.

Subcategories in ProDiag (S7-1500)

From TIA Portal V14 SP1 update 2 the individual subcategories for ProDiag are no longer added manually, but activated or deactivated in the "Activation" column.

This results in the fact that following the upgrade of a project <V14 SP1 Update 2, which uses already existing subcategories in at least one monitoring, the "Repair" button must first be selected before you can create new subcategories.

Instructions "(U)MOVE_BLK" and "(U)FILL_BLK" (S7-1500)

The instructions "(U)MOVE_BLK" and "(U)FILL_BLK" only accessed the process image in TIA Portal <= V15 when directly accessing the I/O.

This behavior has been corrected and now causes a runtime error because direct I/O access is not permitted for BLK instructions.

Program loops in SCL (S7-1200/S7-1500)

In S7-1200/S7-1500 it is not possible to change the index of a program loop from within the loop with SCL.

The following program example would therefore be invalid:

```
FOR #i := 1 TO 10 DO
#i := #i + 1;
END_FOR;
```

As of TIA Portal V15, a more exact syntax check is conducted in this regard during compilation and a compilation error is output.

Return values for functions in SCL

The behavior for functions whose return values have a structured data type has been adjusted. Structured data types are e.g. PLC data types or STRING.

Regardless of whether or not the ENO output uses the function, until now, it could happen that a temporary working copy of the function value was created although no working copy was created by the same call without ENO. The assignment of these temporary copies to the interconnected tag is made when the function is exited.

This behavior was changed as follows:

Using the output parameter ENO now no longer has any influence on whether the return value is transferred with a copy or with a reference. If the return value has a structured data type, it is generally transferred as a reference. The transfer rules are described in detail in the section "Transfer parameter as copy or as pointer".

LDT_TO_DATE in SCL (S7-1500)

The reaction of LDT_TO_DATE to a conversion error in runtime was corrected.

Comparison of an ARRAY element with a variable of the data type "VARIANT" in SCL (S7-1200/1500)

In the past the comparison of a variably indexed ARRAY element with a VARIANT was not carried out correctly in some cases. Instead of the ARRAY element the complete ARRAY was used for the comparison.

This behavior has been corrected in V15: The indexed ARRAY element is now evaluated for the comparison. If you use such comparisons in your program, you have to check the respective block after upgrading.

Example:

```
IF (#my_Array[#1] = #my_variant) THEN…
```

In the past "my_variant" was compared with "my_Array". As of V15 the comparison is carried out correctly and "my_variant" is compared with Element #1 of "my_Array".

Instruction "GetSymbolPath" (S7-1200/S7-1500)

If an FB interconnects a static variable to a formal parameter of a call and a nested block uses "GetSymbolPath" V1.0 or V1.1 for the parameter, compiling this block results in an additional display of the DB name before the name of the static variable, however, without a period as a delimiter and without replacement of the #. Version V1.2 of "GetSymbolPath" shows the path correctly with a period as a delimiter and without #.

If you use the "GetSymbolPath" instruction in your program, you should use the "GetSymbolPath" V1.2 instruction after upgrading to >=V15.

Instruction "GetInstancePath" (S7-1200/S7-1500)

If an FB interconnects a static variable to a formal parameter of a call and a nested block uses "GetInstancePath", compiling this block results in an additional display of the DB name before the name of the static variable.

Additions and subtractions with DTL/LDT/DT

In additions and subtractions with the data types DTL/LDT/DT, non-typed constants can no longer be used. A syntax error is signaled.

The following arithmetic operations, for example, are no longer possible:

```
#myDTL + 1
```

```
#myLDT + 1
```

```
#myDate_And_Time +1
```

Use constants of the data type "Time" or "LTime" instead

```
#myDTL + TIME#1d
```

```
#myLDT + TIME#1ms
```

```
#myDate_And_Time + LTIME#1ns
```

Runtime information (S7-1200/S7-1500)

The extended instructions "GetInstanceName", "GetInstancePath", "GetSymbolName", "GetSymbolPath" and "GetBlockName" now behave in the same way in all programming languages: If the character string to be output does not fit in the output parameter on account of size restrictions, the name is truncated and the character "..." is displayed at the end of the character string.

ARRAYs of multi-instances

In the past ARRAYs of multi-instances could not be addressed correctly when variables were used to address the index of a multi-instance that was in turn instantiated in a further multi-instance, that in turn was an element of an ARRAY of multi-instance.

Example:

```
#MyMultiArray[#index].MyMultiMember := 10;
```

This behavior has been corrected in V15:

If you have used such accesses in your program, you may have to check the affected block after upgrading and possibly recompile it.

Block parameters of the data type "VARIANT" (S7-1200/1500)

As is already the case in LAD, FBD, STL and GRAPH, it is now no longer possible either in SCL to assign and instance DB as the actual parameter to a formal parameter of the data type "VARIANT".

An exception is formed by instance DBs that are derived from a PLC data type or a system data type. You assign these to a block parameter of the data type "VARIANT" in all programming languages.

It is no longer possible to assign a write-protected tag for a block parameter of the data type "VARIANT" as an actual parameter.

If possible, remove the write protection of the variable. If this is not possible, check how you can change your program to prevent access to the write-protected variable. If you only want to have read access, for example, you can copy the content of the write-protected variable in the calling block into a different variable without write protection. Subsequently you can hand over this variable as an actual parameter.

Importing external sources

When importing an external source in which an instruction is used as a multi- or single-instance, the entire program was changed to the instruction version used in the source in the past. As of Version V15 of TIA Portal this behavior has been corrected: The instructions already contained in the project are no longer changed over, but rather retain the instruction version configured by you.

If the project contains different versions of an instruction after the import, a syntax error is reported. In this case compile the complete project again. To do so, select the "Program blocks" folder in the project tree and select the command "Compile > Software (only changes)" or "Compile > Software (compile all)" from the shortcut menu.

Size of ARRAY of BOOL/BYTE/CHAR (S7-1500)

In the past ARRAYS of the data types BOOL, BYTE or CHAR had different sizes depending on whether they were used within a structure or not. As of V15 the size of the ARRAYs has been standardized. If you access absolutely in your program, for example by using an ANY pointer to an ARRAY of the type of BOOL, BYTE or CHAR, you have to check the program after upgrading.

2.4.5.3 Compatibility of PLC programs from versions prior to V14 SP1

Contents

Information that could not be included in the online help and important information about product characteristics.

Compatibility

In V14 SP1 you can generally continue to use all programs that were created with older versions of TIA Portal. However, because improvements were made to the compiler and errors corrected there in V14 SP1, it can occur in rare cases that the program reacts differently after the upgrade or that you have to adjust the program code manually. These cases are described in detail below.

In addition, you have the option of editing the project in compatibility mode. You can find additional information on the compatibility under "Compatibility of projects".

Total size of the associated values for program messages

The alarm procedure of the S7-1500 CPU allows a maximum of 512 bytes for associated values (SD parameters) of the instruction "Program_Alarm".

You will only see a warning regarding this number in TIA Portal <= V14 during compilation. There is a stricter check as of TIA Portal V14 SP1. If this number is exceeded, an error is output during compilation to rule out error scenarios during runtime. This check from now on no longer refers to the actual size of the SD parameters at the time the instruction "Program_Alarm" is called, but to the maximum possible size of the SD parameters. This means you must decide beforehand with what length you declare the character string tags you are transferring, as these take up a large number of bytes.

This must be taken into account in the migration of projects from version <= V14 to version V14 SP1 Update 1.

Program loops in SCL (S7-1200/S7-1500)

In S7-1200/S7-1500 it is not possible to change the index of a program loop from within the loop with SCL.

The following program example would therefore be invalid:

```
FOR #i := 1 TO 10 DO
#i := #i + 1;
END_FOR;
```

As of TIA Portal V14.0.1, a more exact syntax check is conducted in this regard during compilation and a compilation error is output.

CASE statements in SCL

Labels without associated "GOTO" instruction are no longer permitted in a CASE statement. A syntax error is signaled.

ARRAYs of multi-instances

In some cases ARRAYs of multi-instances could not be correctly be addressed in V14 when individual tags within a multi-instance had the retentivity setting "Set in IDB". This behavior has been corrected in V14 SP1.

If you have used such tags in your program, you may have to recompile the affected block after upgrading.

Parameter type "Block_DB" for entry of the instance DB

In LAD and FBD, you cannot enter the instance DB of an instruction using an input of the data type "BLOCK_DB". This behavior has been made uniform for all CPU families and all block types.

As of TIA Portal V14 SP1, a syntax error is output.

If you have transferred instance DBs in your program with the help of the "Block_DB" data type, you change your program. Instead use a parameter instance to transfer the instance during runtime.

I/O accesses as input parameters of blocks (S7-1500 as of FW V2.1)

In the event of an I/O access error, direct I/O accesses which are interconnected as input parameters to blocks no longer result in the block being run through. Instead of this, the system uses the replacement value of the signal in the block.

For CPUs S7-1500 with FW smaller than V2.1 and all CPUs S7-1200, system behavior is unchanged:

The block is not called as a result of the I/O access error. Program execution is continued after the block call. If OB 122 exists or local error handling is enabled, these are executed.

2.4.5.4 Compatibility of PLC programs from versions prior to V14

Contents

Information that could not be included in the online help and important information about product characteristics.

Compatibility

In principle, you can continue to use in V14 all PLC programs that were created with older versions of TIA Portal. However, because improvements were made to the compiler and errors corrected there in V14, it can occur in rare cases that the program reacts differently after the upgrade or that you have to adjust the program code manually. These cases are described in detail below.

STL: Instructions "SRW", "SLW" and "SSI" (S7-300, S7-400, S7-1500)

The permitted value range of the offset number of these instructions has changed in TIA Portal V14 from TIA Portal V13 SP1.

In version V13 SP1, it is possible to specify a number in the range 0 to 31 as an offset number on a CPU of the S7-1200/1500 series and a number in the range 0 to 15 as an offset number on a CPU of the S7-300/400 series.

As of Version V14, the value ranges for all CPU series (S7-300/400/1200/1500) was set to the uniform value from 0 to 15.

Representation of the BCD format

The representation of the BCD format has changed in regard to the sign from TIA Portal V13 SP1 to TIA Portal V14.

The values in BCD format are show without signs in version V13 SP1.

As of version V14, the values in BCD format are show with sign. This can lead to an altered representation of the values in BCD format.

Integers (decimal system)	Hexadecimal numbers	Representation in V13 SP1	Representation in V14
0	16#0000	BCD#0	BCD#0
-26215	16#9999	BCD#9999	BCD#-999
1365	16#0555	BCD#555	BCD#555
21845	16#5555	BCD#5555	BCD#555
4096	16#1000	BCD#1000	BCD#0
-28672	16#9000	BCD#9000	BCD#0

Instructions "SET_BF: Set bit field" and "RESET_BF: Reset bit array" (S7-1200, S7-1500)

The response of the instruction in regard to the structures has changed from TIA Portal V13 SP1 to TIA Portal V14.

In version V13 SP1, the number of bits that set or reset are always indicated at the input. This also applies, for example, if you have specified a structure of data type of the STRUCT or ARRAY PLC data type. If you want, for example, you want set or reset 10 bits and the structure contains only 5 bits, then the following 5 bits within the address sequence is also set or reset.

With structures of the type PLC data type, as of version V14 STRUCT or ARRAY the number of bits contained in the structure represents the maximum number of bits that can be reset: If you specify the value "20", for example, and the structure only contains 10 bits, only these 10 bits are set. If you specify the value "5", for example, and the structure contains 10 bits, then exactly 5 bits are set.

Instructions "SCALE: Scale" and "UNSCALE: Unscale" (S7-1500)

The response of the instruction has changed from TIA Portal V13 SP1 to TIA Portal V14 in regard to specifying the limits (LO_LIM> HI_LIM).

In version V13 SP1, you received an error message when the low limit was greater than the high limit.

As of version V14, this value is allowed and the result is inversely proportional to the input value.

On CPUs of the S7-300/400 series, the instruction has always scaled the result inversely proportional to the input value.

Unused bits of PLC data types (UDT) with firmware >= V1.8.1

The unused bits of PLC data types in standard memory areas are occupied or overwritten, for example, for a PLC data type that contains 4 bits.

With firmware versions < V1.8.1, you could not use the unused bits of a PLC data type elsewhere.

With firmware version >= V1.8.1, all bits are occupied or overwritten even if only 4 bits are used.

Note

Address assignments

Ensure you do not assign the same absolute address twice to different symbolic address assignments.

Explicit data type conversion in SCL (S7-1200) with firmware >= V4.2

With firmware versions < V4.2, the string was transferred aligned to the right and filled with leading spaces during explicit data type conversion of SINT/INT/DINT/REAL_TO_STRING/WSTRING in SCL.

Example: REAL_TO_WSTRING(12) = ' 1.200000E+1'

As of TIA Portal V13, the string is displayed with a leading sign during explicit data type conversion of SINT/INT/DINT/REAL_TO_STRING/WSTRING in SCL and transferred aligned to the left.

Example: REAL_TO_WSTRING(12) = '+1.200000E+1'

SCL: EN/ENO mechanism with block parameters of data type (W)STRING (S7-1200/1500)

As of TIA Portal V14, it is checked whether block parameters of data type (W)STRING are truncated when parameters are transferred during runtime. This can occur if formal and actual parameters have different declared lengths. If the declared length of the target parameter during runtime is insufficient to accept the (W)STRING, the (W)STRING is truncated and the enable output ENO is set to "FALSE".

If you evaluate the enable output ENO in your program, the semantics of your program may change.

Forwarding STRING parameters between optimized blocks and standard blocks (S7-1200/1500)

Prior to TIA Portal V14, no length information about STRINGs was transferred when parameters were transferred between optimized and non-optimized blocks. This meant that access errors not detected by the ENO mechanism could occur during the further processing of the STRING. The enable output ENO remained at TRUE despite an access error.

As of TIA Portal V14, the STRING length information is transferred and access errors are monitored by the ENO mechanism. If you evaluate the enable output ENO in your program, the semantics of your program may change.

Length declaration for constants of the "STRING" or "WSTRING" data type

Length declarations are no longer permitted for local and global (W)STRING constants. If your program includes length declarations, for example, "MyStringConst [7]", these are automatically removed during the upgrade. This change has no semantic effect on your program, however.

Changing the base data type for HW_ANY from WORD to UINT

The hardware data type "HW_ANY" is used to identify any hardware component, such as a module. In V14, the base data type of "HW_ANY" was changed from WORD to UINT. After the conversion to V14, compiling errors can therefore occur if you have assigned an explicit WORD constant (such as W#16#1) or a tag of data type WORD at a parameter of data type "HW_ANY".

In this case, change the notation of the constant (e.g. to 16#1) or change the data type of the tag to UINT.

Length of block comments

As of TIA Portal V14, the block comments must not exceed a length of 32767 Unicode characters.

Syntax check of the instructions "SR: Set/reset flip-flop" and "RS: Reset/set flip-flop"

Inputs, outputs, bit memories as well as static or local data can be used as operands at the "S" input. Use of constants is not permitted. So far, the use of the constants "0" and "1" was not rejected by the syntax check. As of TIA Portal V14 the syntax check was improved so that the use of "0" or "1" is now rejected at the "S" input. You may now receive an error message during the compiling of a program that has been compiled without errors up to this point. Change your program in this case and use a permitted operand.

Any pointer: Access to optimized data

For a CPU of the S7-1500 series, the ANY pointer can also only point to memory areas with "Standard" access mode. Access to optimized data is not permitted with an ANY pointer.

A more detailed syntax check of ANY pointers is performed as of TIA Portal V14. Access to optimized data is now rejected. You may now receive an error message during the compiling of a program that has been compiled without errors up to this point.

"(L)REAL" data type

With 64-bit Windows versions, the LREAL data type is sometimes evaluated differently than with the previous 32-bit versions of Windows. A difference may occur when rounding and (L) real values of constants in the least significant bit.

The potential difference is in the last digit of the mantissa and in most cases therefore only has a slight effect on the accuracy.

If you evaluate the last digit of the mantissa, the semantics of your STEP 7 program may change after the conversion.

"ST" parameter for IEC timers

The "ST" parameter of an IEC timer is used internally and it is prohibited to write it. This why this parameter is no longer visible as of TIA Portal V14. If you have write access in your program to the "ST" parameter, an error message is generated during compiling after the upgrade.

If you exchange data with Inter Project Engineering and you have access to the "ST" parameter with HMI, you must first export the project and then re-import it into the proxy CPU. Only then does the "ST" parameter disappear in the HMI configuration.

Associated values for messages (S7-1500)

Projects created with an older version of the TIA Portal can contain embedded associated values with different structures in the different project languages. The associated values may have a different order or may be missing in a different language. When a project of this type is upgraded, the order of the associated values is harmonized. It is based on the reference language set in the original project. In this case, it is recommended to have the translated texts checked and corrected.

Implicit data type conversion for block parameters of the "DB_ANY" data type

When accessing a block parameter of the "DB_ANY" data type with the syntax DB_ANY.%DB(B|W|D), no implicit data type conversion was performed previously. The bit pattern of the source value was simply converted into the target data type. If the target data type was smaller than the source data type, an overflow occurred and the written value may have been wrong or inaccurate. The enable output ENO is not set to "FALSE" when accuracy is lost or runtime errors occur.

As of V14, conversion is performed according to the rules of implicit data type conversion when parameters are passed with the syntax DB_ANY.%DB(B|W|D). Note that after the upgrade a different value may be written and the semantics of your program may change. In addition, the enable output ENO is set to "FALSE" when a runtime error occurs.

Interface of organization blocks with standard access

The interface of organization blocks with standard access must have a minimum size of 20 bytes. In older versions of TIA Portal, only the interface of OB1 was checked with regard to the minimum size during the compilation run. As of V14, interfaces of all organization blocks are checked. You may now receive an error message during the compiling of a program that has been compiled without errors up to this point.

Comparing tags of WORD data type to tags of the S5TIME data type (S7-1500)

As of TIA Portal V14, both variables are converted to the TIME data type when comparing a variable of WORD data type to a variable of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the variables cannot be converted, the comparison is not performed and the result is FALSE. After successful conversion, the comparison is performed based on the selected comparison expression.

2.4.5.5 Compatibility of PLC programs from versions prior to V13 SP1

Content

Information that could not be included in the online help and important information about product characteristics.

Compatibility

You can continue to use in V13 SP1 all programs that were created with TIA Portal V12 SP1 or V13. However, because improvements were made to the compiler and errors corrected there (compilation of the program code) in V13 SP1, it can occur in rare cases that the program reacts differently after the upgrade or that you have to adjust the program code manually. These cases are described in detail below.

Unused bits of PLC data types (UDT) with firmware >= V1.8.1

The unused bits of PLC data types in standard memory areas are occupied or overwritten, for example, for a PLC data type that contains 4 bits.

With firmware versions < V1.8.1, you could not use the unused bits of a PLC data type elsewhere.

With firmware version >= V1.8.1, all bits are occupied or overwritten even if only 4 bits are used.

Note

Address assignments

Ensure you do not assign the same absolute address twice to different symbolic address assignments.

Explicit data type conversion in SCL (S7-1200) with firmware >= V4.2

With firmware versions < V4.2, the string was transferred aligned to the right and filled with leading spaces during explicit data type conversion of SINT/INT/DINT/REAL_TO_STRING/WSTRING in SCL.

Example: REAL_TO_WSTRING(12) = ' 1.200000E+1'

As of TIA Portal V13, the string is displayed with a leading sign during explicit data type conversion of SINT/INT/DINT/REAL_TO_STRING/WSTRING in SCL and transferred aligned to the left.

Example: REAL_TO_WSTRING(12) = '+1.200000E+1'

Reading an invalid peripheral input

In the TIA Portal as of version 12, the error code 16#2942 for the reading of an invalid peripheral input is only output if you have programmed the faulty access in such a way that this becomes effective, for example %MW10 := “InvalidWordAccess":P, and does not lie within an irrelevant sequence. For example, this would be #tmp := “InvalidWordAccess":P, if the tag #tmp is not used in the rest of the program block.

Instruction "S_CONV: Convert character string"

The EN/ENO mechanism behaves differently in TIA Portal V13 SP1 than in TIA Portal V13.

In version V13, the ENO enable output returns the signal state "0" in case of error, even if you have deactivated the ENO enable output. If you have switched an additional instruction to the ENO enable output, this is then not executed.

In version V13 SP1, the ENO enable output returns the signal state "1" in case of error, if you have deactivated the ENO enable output. If you have switched an additional instruction to the ENO enable output, this is then executed as expected.

Instruction "SET: Set bit array" (S7-300/S7-400)

In the TIA Portal, you can also interconnect an element of a data block at the N parameter of the SET instruction.

Instruction "MUX: Multiplex" (SCL)

Up to and including TIA Portal V13, the value of the tag at the input parameter was output unchanged as a function value, even if the K parameter had a negative integer. This behavior has changed in TIA Portal V13 SP1.

In TIA Portal V13 SP1, if you use a tag with a valid data type at the input parameters of the MUX instruction and the K parameter is a negative integer, the value of the tag is changed. You can find the valid data types in the description of the MUX instruction in the information system.

Instruction "DEMUX: Demultiplex" (SCL)

Up to and including TIA Portal V13, no value was output at the OUTELSE output parameter if the value of the K parameter was < 0. In contrast, the value of the IN input parameter was output at the OUTELSE output parameter if the value of the K parameter was > available outputs. This behavior has changed in TIA Portal V13 SP1.

In the TIA Portal V13 SP1, if you specify at the K parameter a value that is outside the available outputs (K < 0 or K > available outputs), then the value of the IN input parameter is output at the OUTELSE output parameter.

Instruction "MOVE: Move value" (STL) (S7-1500)

As of the TIA Portal V13 SP1, a stricter syntax rule applies to the "MOVE" instruction in STL:

Up to and including TIA Portal V13. it was possible to specify non-typed constants, such as "0" at the input parameter IN.

If you want to copy, for example, the value "0" in the TIA Portal V13 SP1, you need to specify a typed constant according to the data type of the output parameter (e.g. INT#0, DINT#0, etc.).

Instruction "Program_Alarm: Generate program alarm with associated values" (S7-1500)

As of TIA Portal V13 SP1 the data type INT is no longer permitted for use for the associated values or an index tag of a text list. Use the UINT data type instead.

Instructions "READ_DBL: Read from data block in the load memory" and "WRIT_DBL: Write to data block in the load memory"

TIA Portal V13 SP1 reports a compilation error if you use "READ_DBL" V1.0 or "WRIT_DBL" V1.0 and access a data block with the attribute "Only store in load memory" using the parameter DSTBLK.

Instead, use version 1.2 of both instructions.

"READ_DBL instructions: Read from data block in the load memory" and "WRIT_DBL: Write to data block in the load memory" (SCL)

As of TIA Portal V13 SP1, the "READ_DBL" and "WRIT_DBL" instruction in SCL are subject to the same strict syntax rules as all other programming languages:

In TIA Portal V13, it was possible to specify "standard" data blocks at the SRCBLK and DSTBLK parameters that contained tags with different data types.

As of TIA Portal V13 SP1, the individual tags contained must have the same data type for both "standard" as well as for "optimized" data blocks; otherwise, compiling errors are reported.

Function values (Return)

As of TIA Portal V13 SP1, more stringent syntax rules apply to function call:

It is checked that the function value (Return) is written in any case, even if multiple possible program paths can be run through in the function. Therefore, there is no longer any risk that the function values will accidentally not be written during runtime.

However, you may possibly receive syntax errors in functions during compiling, which it was still possible to compile without errors in V13. In such cases change the program code so that the function valve is written in all possible program paths.

Example:

SCL

```
IF #MyIn1 = #MyIn2 THEN
   #Block_3 := #MyIn1 + 1;
END_IF;
```

In this example, the function value of "Block_3" is not written if the condition of the IF instruction is not fulfilled. The function value then contains an undefined value.

SCL

```
#Block_3 := #MyIn1;
IF #MyIn1 = #MyIn2 THEN
   #Block_3 := #MyIn1 + 1;
END_IF;
```

In this example the function valve of "Block_3" is definitely written in the program, since "MyIn1" is set as default before the IF loop is completed.

Comparing the hardware data types HW_IO and HW_DEVICE

As of TIA Portal V13 SP1, there is a stricter syntax rule that is valid for the comparison of the data types HW_IO and HW_DEVICE:

Up until and including TIA Portal V13 it was possible to compare the data types HW_IO and HW_DEVICE directly with each other.

If you want to compare these data types in TIA Portal V13 SP1 you have to first create a tag of the data type HW_ANY in the section "Temp" for the block interface and then copy the LADDR (from data type HW_DEVICE) to the tag. It is then possible to compare HW_ANY and HW_IO.

Implicit conversion of floating-point numbers to DWORD (GRAPH) (S7-1500)

As of TIA Portal V13 SP1 stricter syntax rules apply for the implicit conversion of floating-point numbers to DWORD:

Up to and including TIA Portal V13, it was possible to convert a non-typed constant (e.g. 1.0) implicitly to the data type DWORD if the value was less than 32 bits.

With TIA Portal V13 SP1, if you want to convert a floating-point number implicitly to the data type DWORD, you can only use typed constants (e.g. REAL#1.0) since a non-typed constant (e.g. 1.0) is interpreted as data type LREAL and can no longer be converted implicitly to DWORD.

Assigning a value to a LTIME constant (S7-1500)

As of the TIA Portal V13 SP1, a stricter syntax rule applies to the definition of a value to a LTIME constant:

Up until and including TIA Portal V13 it was possible to assign a TIME value (e.g. T#10s) to a LTIME constant.

If you want to assign a value to a LTIME constant in TIA Portal V13 SP1, you have to use LTIME syntax (e.g. LT#10s).

Please note that the corresponding TIA Portal projects from previous versions already had to have been adjusted before the upgrade to TIA Portal V13 SP1.

Indirect indexing of ARRAY components of data type bit string in SCL (S7-1200/S7-1500)

In TIA Portal V13, you can address the components of an ARRAY for a CPU of the S7-1200/1500 series in SCL with a tag of the data type BYTE, WORD, DWORD or LWORD as index in addition to a tag of the integer data type if the IEC check is not set. This is not possible for a CPU of the S7-300/400 series.

The result is an incompatibility with TIA Portal V12 SP1 in which indirect indexing with the data type bit string is also permitted on a CPU of the S7-300/400 series.

EN/ENO mechanism for STRING conversion (S7-1200/S7-1500)

Conversion	Description
Strg_TO_Chars: Convert character string to Array of CHAR	The ENO enable output returns the signal state "0" even if you have deactivated the ENO enable output: • For an invalid character at the CHARS parameter • For an invalid ARRAY index at the PCHARS parameter • If the sum of the PCHARS and STRG parameters exceeds the length of the target ARRAY.
Chars_TO_Strg: Convert Array of CHAR to character string	The ENO enable output returns the signal state "0" even if you have deactivated the ENO enable output: • If the sum of the PCHARS and CNT parameters exceeds the length of the source ARRAY.

Overflow of the local data stack (S7-300/S7-400)

As of TIA Portal V13 SP1, the size of the local data stack is checked to ensure it does not exceed what you have defined for the OB priorities in the hardware configuration.

This means that you may get error messages during compilation in projects that could be compiled without errors in V13. If this happens, change the maximum size of the local data stack in the hardware configuration.

2.5 Inter Project Engineering (IPE)

2.5.1 Notes on IPE

Contents

Information that could not be included in the online help and important information about product characteristics.

Using system diagnostics in the device proxy

To use the "system diagnostics" function in an IPE device proxy, for example, a system diagnostics view, insert the PLC alarms as content of a device proxy.

Connections to the SIMATIC S7-1500 software controller

Connections to the SIMATIC S7-1500 software controller as device proxy are only possible via the interfaces assigned to the software PLC.

Support of Alarm_S alarms

The integrated configuration with WinCC and SIMATIC Manager support the Alarm_S alarms, which are available via data blocks and Continuous Function Chart (CFC).

Compatibility

With TIA Portal V15.1, you can open the IPE files of the same version and earlier versions from V13 SP1.

IPE files created with TIA Portal version V15.1 cannot be opened with earlier versions.

2.6 Technological functions

2.6.1 Notes on S7-1500 Motion Control (S7-1500)

Exception error when fixed stop detection is activated with MC_TorqueLimiting

When you start an "MC_TorqueLimiting" job with parameter "Mode" = 1 and the technology object is not enabled, an exception error may occur in the execution system of the CPU. Make sure that the technology object is enabled (MC_Power.Status = TRUE) before you start an "MC_TorqueLimiting" job.

Configuration of TM PosInput 1/TM PosInput 2 for position detection with the technology object measuring input

- Technology object measuring input
  For use with Motion Control and a technology object measuring input, select the measuring input type "Measuring using PROFIdrive telegram (drive or external encoder)" under "TO_MeasuringInput > Hardware interface > Input measuring input".
- TM PosInput 1/TM PosInput 2
  You set the parameters for the encoder signals of the channel under "Module parameters" in "Position detection for Motion Control" mode. The parameters must be set depending on the encoder used.

Note

When a single-turn absolute encoder is used and two edges are to be measured ("MC_MeasuringInput.Mode" = 2, 3 or 4), the distance between the measured edges of the measurment input must be < 1 encoder revolution. Otherwise, use a multi-turn absolute encoder.

Display of the geometry of the kinematics type "Cylindrical robot" (S7-1500T)

In contrast to the documented display of the kinematics type "Cylindrical robot", the length L1 represents the distance between the kinematics zero point (KZP) and the bottom edge of the articulated arm. The flange length LF represents the distance between the bottom edge of the articulated arm and the flange coordinate system (FCS). This applies both to the "3D" variant and the "3D with orientation" variant.

The graphic below shows the side view of the kinematics type "Cylindrical robot 3D" (xz plane) as an example:

- The position of the axes and the coordinate systems KCS and FCS
- The zero positions of the axes A1 and A2
- The deflection of the kinematics is indicated (dashed)

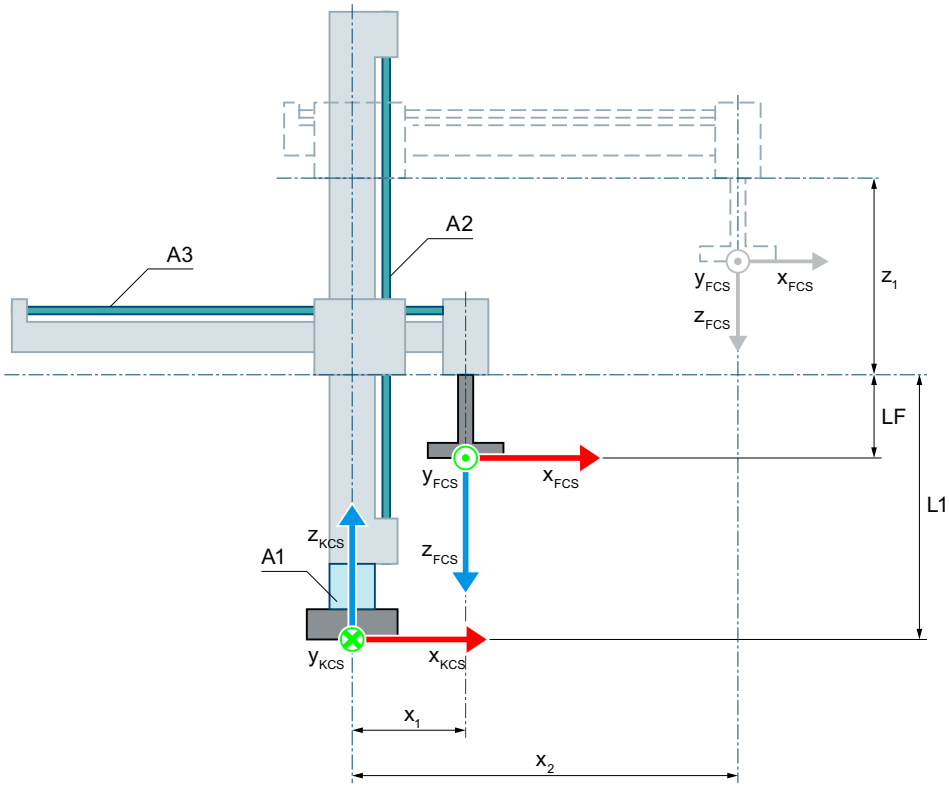

Zero position of the axes A1 and A2

L1 At zero position of the axis A2:

Distance of the FCS to the kinematics zero point (KZP) and flange length LF in z direction of the KCS

LF Flange length before the FCS in the z direction of the FCS

x_1 Positive deflection of the axis A3

At the zero position of axis A3, the z axis of the FCS is located on the z axis of the KCS. For mechanical reasons, the kinematics shown cannot approach the zero position of the axis A3.

Deflection of the kinematics

x_2 Positive deflection of the axis A3

z_1 Positive deflection of the axis A2

Interrupt execution of the motion with an "MC_GroupInterrupt" job (S7-1500T)

Contrary to the documented behavior, you can only interrupt the execution of the motion at the kinematics technology object during an active kinematics motion with an "MC_GroupInterrupt" job. If the kinematics is already at a standstill, the execution of the motion is not interrupted for subsequent motion jobs. The kinematics is not in "Interrupted" state (StatusWord.X17 (Interrupted)).

By entering the value "0.0" for the velocity override, you can suspend the motion until all requested jobs have been entered in the job chain. Then you can continue the motion with a velocity override "100.0".

Effect of the velocity override on kinematics motion jobs (S7-1500T)

If you change the velocity override, different results occur for motion jobs of the kinematics technology object, in which the orientation motion determines the duration of the overall motion. This depends on whether you change the velocity override before or after the motion job is issued.

For motion jobs that are already in the job sequence ("Busy" = TRUE) or being executed ("Active" = TRUE), the kinematics motion is already reduced to the possible orientation velocity. This means that the change to the velocity override only affects the path velocity.

For motion jobs that are added to the job sequence after the change to the velocity override, the changed velocity override also affects the orientation velocity.

For this reason, change the value of the velocity override before you issue motion jobs.

Controlling velocity override with the Kinematics control panel (S7-1500T)

Contrary to the behavior described in the documentation, you can use the slider to set the velocity only in the world coordinate system (WCS) and object coordinate system (OCS). Kinematics motions in the machine coordinate system (MCS) are executed with the velocity override (<TO>.Override.Velocity) of the individually interconnected axes.

Executing "MC_GroupStop" following a "MC_GroupInterrupt" (S7-1500T)

If you execute an "MC_GroupStop" job after an "MC_GroupInterrupt" job, you have to leave the parameter "Execute" at the "MC_GroupStop" set to TRUE until the job sequence is empty (StatusMotionQueue.NumberOfCommands = 0). If the job sequence is not empty, the remaining motion jobs in the job sequence are executed.

2.6.2 Notes on additional technology functions (S7-1500)

Parameters of the technology objects in TIA Portal Openness

You can find a list of the available technology object parameters in the product information "Parameters of the technology objects in TIA Portal Openness" on the internet (https://support.industry.siemens.com/cs/ww/en/view/109744932).

S_USSI on S7-1500

Note the following when using the instruction S_USSI of the distributed I/O ET 200S 1SI on an S7-1500.

The parameter ANZ shows the value 0 even in the case of an error.

If the CPU is switched to stop with the S_USSI instruction, check all possible errors listed at the parameter ANZ.

PtP module with CM 1243-5 PROFIBUS Master

You have to make the following settings in the instance DB of the instructions when using the PtP modules CM PtP RS232 BA, CM PtP RS422/485 BA, CM PtP RS232 HF, CM PtP RS422/485 HF and CM PtP with a CM 1243-5 PROFIBUS Master with firmware ≤ V1.3.4:

- Send_P2P
  max_record_len = 240
- Modbus_Master
  Send_P2P.max_record_len = 240
- Modbus_Slave
  Send_P2P.max_record_len = 240

PID_Temp on CPU 1500 V1.7

Note the following when using PID_Temp on a CPU 1500 with firmware version V1.7:

- The integrated dead zone in PID_Temp must not be used on CPU 1500 V1.7.
  Do not change the preset 0.0 of the associated parameters Retain.CtrlParams.Heat.DeadZone and Retain.CtrlParams.Cool.DeadZone.
- If the cooling output is active (PidOutputSum < 0.0, OutputCool <> 0.0) for a closed loop controller with active cooling and PID parameter switchover (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE) while "Pretuning cooling" is started (Mode = 1, Heat.EnableTuning = FALSE, Cool.EnableTuning = TRUE), aborts with error ErrorBits = 16#0040_0000.
  - In this case use "Fine tuning cooling" (Mode = 2, Heat.EnableTuning = FALSE, Cool.EnableTuning = TRUE)

  or

  - Before starting the pretuning, switch to manual mode with a manual value larger than 0.0 (Mode = 4, ManualValue ≥ 0.0).

As of firmware version V1.8, these two restrictions no longer apply.

2.6.3 Notes on technological functions (S7-1200)

Display errors in the online/offline comparison for S7-1200 Motion Control

If you upgrade a TIA Portal project V15 to V15.1 and establish an online connection to the CPU on which the project with TIA Portal V15 was loaded, differences in the online/offline comparison are shown which do not actually exist. This can also be an invalid configuration.

Load the upgraded project to the CPU. The online/offline comparison shows the correct results once again.

Parameters of the technology objects in TIA Portal Openness

You can find a list of the available technology object parameters in the product information "Parameters of the technology objects in TIA Portal Openness" on the internet (https://support.industry.siemens.com/cs/ww/en/view/109744932).

PtP module with CM 1243-5 PROFIBUS Master

You have to make the following settings in the instance DB of the instructions when using the PtP modules CM PtP RS232 BA, CM PtP RS422/485 BA, CM PtP RS232 HF, CM PtP RS422/485 HF and CM PtP with a CM 1243-5 PROFIBUS Master with firmware ≤ V1.3.4:

- **Send_P2P**
  max_record_len = 240
- **Modbus_Master**
  Send_P2P.max_record_len = 240
- **Modbus_Slave**
  Send_P2P.max_record_len = 240

2.6.4 Notes on technological functions (S7-300, S7-400)

Parameter assignment for the FM x51 and FM x52

In contrast to the information in the Online Help, if you change the measuring system, the previously entered parameter values are not converted to the new measuring system.

Programming FM x50-1, FM x51 and FM x52

The "initial values" specified in the parameter tables in the online help regarding the instructions correspond to the "default value" in the software interface of the user programming.

IM 174 migration

During the migration of projects with IM 174, the parameter assignment of IM 174 is set to default values. It is absolutely necessary to set parameters for IM 174 again after the migration.

2.6.5 SIMATIC STEP 7 Easy Motion Control

2.6.5.1 Notes on SIMATIC STEP 7 Easy Motion Control

There are no notes available on SIMATIC STEP 7 Easy Motion Control.

2.6.6 SIMATIC STEP 7 PID Professional

2.6.6.1 Notes on SIMATIC STEP 7 PID Professional

There are no notes available on SIMATIC STEP 7 PID Professional.

WinCC 3

3.1 Security information

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

Passwords

Various passwords are set by default in WinCC. For security reasons, you should change these passwords.

- For HMI devices with version 12, the default password for the Sm@rtServer and for the embedded Web server is "100". A default password is not preset for HMI devices with version V13.
- For the user "Administrator", the default password is "administrator".

Integrated Web server

It is always possible on a PC to access HTML pages in Runtime, even though the option"HTML pages" is disabled. Setup always installs the standard pages of the Web Server on the PC. Assign an administrator password to prevent unauthorized access to the pages.

Communication via Ethernet

In Ethernet-based communication, end users themselves are responsible for the security of their data network. The proper functioning of the device cannot be guaranteed in all circumstances; targeted attacks, for example, can lead to overload of the device.

Use of SSL 3.0

For security reasons, the use of the protocol SSL 3.0 is not recommended on Comfort Panels or in Runtime Advanced. The use of the protocol SSL 3.0 is disabled by default on Comfort Panels. If you nevertheless wish to activate the use of SSL 3.0, select the following in Internet Explorer or in "Start Center > Settings: Internet Options > Advanced > Use SSL 3.0".

For RT Advanced, the use of SSL 3.0 can be disabled in Internet Explorer or in the Control Panel under "Internet Options > Advanced" by deactivating the "Use SSL 3.0" option.

Network settings

The following tables show the network settings of each product which you need in order to analyze the network security and for the configuration of external firewalls:

WinCC Professional (without simulation)					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
ALM	4410*	TCP	Inbound, Outbound	License service	This service provides the complete functionality for software licenses and is used by both the Automation License Manager as well as all license-related software products.
HMI Load	1033	TCP	Outbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.
HMI Load	2308	TCP	Outbound	HMI Load (RT Advanced)	This service is used to transmit images and configuration data to panels.
RPC	**	UDP	Inbound, Outbound	Client / server & ES communication (CCAgent)	This service is used by WinCC Professional and WinCC Runtime Professional.
* Default port that can be changed by user configuration ** Port is assigned automatically					

WinCC Simulation for Basic Panels					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
HMI Load	1033	TCP	Inbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.
EtherNet/IP	44818	TCP	Outbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
	2222	UDP	Inbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
Modbus TCP	502	TCP	Outbound	Modbus TCP channel	The Modbus TCP protocol is used for connections to Schneider PLCs.

WinCC Simulation for Basic Panels					
RFC 1006	102	TCP	Outbound	S7 channel	Communication with the S7 controller via Ethernet/PROFINET
Mitsubishi MC	5002	TCP	Outbound	Mitsubishi MC channel	The Mitsubishi protocol is used for connections to Mitsubishi PLCs.

WinCC Simulation for Panels and Runtime Advanced					
Name	**Port number**	**Transport protocol**	**Direction**	**Function**	**Description**
DCP	---	Ethernet	Outbound	PROFINET	The DCP protocol (Discovery and basic Configuration Protocol) is used by PROFINET and provides the basic functionality for locating and configuring PROFINET devices.
LLDP	---	Ethernet	Inbound, Outbound	PROFINET	The LLDP protocol (Link Layer Discover Protocol) is used by PROFINET for topology detection.
SMTP	25	TCP	Outbound	SMTP Communication	This service is used by WinCC Runtime Advanced to send e-mails.
HTTP	80*	TCP	Inbound	Sm@rtServer	The Web server is only available when Sm@rtService is activated. The used port may differ depending on automatically selected settings.
RFC 1006	102	TCP	Outbound	S7 channel	Communication with the S7 controller via Ethernet/PROFINET
NTP	123	UDP	Outbound	Time synchronization	The NTP protocol (Network Time Protocol) is used for time synchronization in IP-based networks.
SNMP	161	UDP	Outbound	PROFINET	The SNMP client functionality is used by STEP 7 to read status information from PROFINET devices.
HMI Load	2308	TCP	Outbound	HMI Load (RT Advanced)	This service is used to transmit images and configuration data to panels.
HTTPS	443*	TCP	Inbound	Sm@rtServer	The Web server with HTTPS protocol is only available when Sm@rtService is activated. The used port may differ depending on automatically selected settings.
VNC server	5900*	TCP	Inbound	Sm@rtServer	This service is only available when Sm@rtService is activated.
	5800*	TCP	Inbound	Sm@rtServer	This service is only available when Sm@rtService is activated.
VNC client	5500	TCP	Outbound	Sm@rtServer	This service is only available when Sm@rtService is activated.
* Default port that can be changed by user configuration					

WinCC Simulation for Runtime Professional					
Name	Port number	Transport protocol	Direction	Function	Description
RPC	**	UDP	Inbound, Outbound	Client / server & ES communication (CCAgent)	This service is used by WinCC Professional and WinCC RT Professional.
RPC	**	UDP	Inbound, Outbound	Client / server communication (CCEServer / CCEClient)	This service is used by WinCC Runtime Professional.
HTTP	80	TCP	Inbound, Outbound	Client / server communication (CCEServer / CCEClient)	This service is used by WinCC Runtime Professional.
RFC 1006	102	TCP	Outbound	S7 channel	Communication with the S7 controller via Ethernet/PROFINET
OPC UA	4840	TCP	Inbound	OPC UA server	This service is required for primary communication via OPC UA. It is activated and configured during installation.
OPC UA discovery	52601	TCP	Inbound	OPC UA server	This service provides information about the installed OPC server. It is installed and configured by the OPC UA server.
DCOM	135	TCP	Inbound	OPC server	This service is part of the Windows operating system. Since communication via OPC (DA) is based on DCOM, this service is required to initialize OPC (DA) connections.
DCOM	**	TCP	Inbound	OPC server	The communication via OPC (DA) is based on DCOM and uses unspecified ports assigned by the system. This should be taken into consideration when using OPC (DA) and creating rules for the firewall.
HTTP	80	TCP	Inbound	OPC server	This service is required for primary communication via OPC XML. It is activated and configured during installation.
NetBIOS	137	UDP	Inbound	OPC server	This service is part of the Windows operating system. Access to this service is required by OPC-Scout, for example, for browsing.
NetBIOS	138	UDP	Inbound	OPC server	This service is part of the Windows operating system. Access to this service is required by OPC-Scout, for example, for browsing.
SNMP	161	UDP	Outbound	SNMP OPC server	This service is used by the SNMP OPC server to change or query data on network drives, for example.
SNMP Traps	162	UDP	Inbound	SNMP OPC server	This service is used by the SNMP OPC server to query events from network drives, for example.
** Port is assigned automatically					

PROFINET protocols for Panels and Runtime Advanced					
Name	Port number	Transport protocol	Direction	Function	Description
DCP	---	Ethernet	Outbound	Lifelist, PROFINET Discovery and configuration	The DCP protocol (Discovery and basic Configuration Protocol) is used by PROFINET and provides the basic functionality for locating and configuring PROFINET devices.
LLDP	---	Ethernet	Inbound, Outbound	PROFINET Link Layer Discovery protocol	The LLDP protocol (Link Layer Discover Protocol) is used by PROFINET for topology detection.
MRP	---	Ethernet	Outbound	PROFINET medium redundancy	The MRP protocol (Medium redundancy protocol) enables control of redundant transmission paths using a ring topology.
PROFINET IO Data	---	Ethernet	Inbound, Outbound	PROFINET Cyclic IO data transfer	Cyclic data exchange is used by panels for direct keys and LEDs.
NARE	---	Ethernet	Inbound, Outbound	Name Address Resolution	This protocol is used to resolve network names and assign IP addresses.
PROFINET Context Manager	34964	UDP	Inbound, Outbound	PROFINET connection less RPC	The PROFINET Context Manager provides an endpoint mapper in order to establish an application relation (PROFINET AR).

Communication connections for Panels and WinCC Runtime Advanced					
Name	Port number	Transport protocol	Direction	Function	Description
Telnet	23	TCP	Inbound	Telnet	This service can be used for maintenance.
SMTP	25	TCP	Outbound	SendEMail	This service is used by Windows CE / PC Runtime to send e-mails.
HTTP	80*	TCP	Inbound	Hypertext Transfer Protocol	The HTTP protocol is used for communication with the internal Web server.
RFC 1006	102	TCP	Outbound	S7 channel	Communication with the S7 controller via Ethernet/PROFINET.
NTP	123	UDP	Outbound	Time synchronization	The NTP protocol (Network Time Protocol) is used for time synchronization in IP-based networks.
DCOM***	135	TCP	Inbound	OPC server	This service is a component of the Microsoft Windows operating system. Communication via OPC (DA) is based on DCOM. This service is therefore required to initialize OPC (DA) connections.
DCOM***	**	TCP	Inbound	OPC server	The communication via OPC (DA) is based on DCOM and uses unspecified ports assigned by the system. This should be taken into consideration when using OPC (DA) and creating rules for the firewall.

Communication connections for Panels and WinCC Runtime Advanced					
NetBIOS over TCP/IP	137	UDP	Outbound	With the use of Remote File Share	Register / log on to a remote server.
NetBIOS over TCP/IP	138	UDP	Outbound	With the use of Remote File Share	Register / log on to a remote server.
SNMP	161	UDP	Outbound	Simple Network Management Protocol	The SNMP client functionality is used by STEP 7 to read status information from PROFINET devices.
HTTPS	443*	TCP	Inbound	Secure Hyper-text Transfer Protocol	The HTTP protocol is used for communication with the CPU-internal Web server via Secure Socket Layer (SSL).
Modbus TCP	502*	TCP	Outbound	Modbus TCP channel	The Modbus TCP protocol is used for connections to Schneider PLCs.
Mitsubishi MC	1025*	TCP	Outbound	Mitsubishi MC channel	The Mitsubishi protocol is used for connections to Mitsubishi PLCs.
Printing	1032	TCP	Outbound	Printing	Printing on the control panel (via Ethernet).
HMI Load	2308	TCP	Outbound	Transfer	This service is used to transmit images and configuration data to panels. On Comfort Panels, this service has been replaced by Device-Manager and SCS as of V13. This service is used to transmit configuration data to WinCC Runtime Advanced.
HMI Load	50523	TCP	Outbound	Transfer	This port is used if port 2308 is not available. This service is used to transmit images and configuration data to panels. On Comfort Panels, this service has been replaced by Device-Manager and SCS as of V13. This service is used to transmit configuration data to WinCC Runtime Advanced.
ALM	4410*	TCP	Inbound, Outbound	Application License Manager	This service of RT Advanced makes available the complete functionalities for software licenses and is used by the Automation License Manager.
OPC UA	4870*	TCP	Inbound	OPC UA server	This service is required for communication via OPC UA.
HMI Load	5001	TCP	Outbound	Device Manager	This service is used to transmit images and Runtime to panels.
HMI Load	5002	TCP	Outbound	SCS (System Configuration Server)	This service is used to transmit configuration data to panels.
VNC client	5500	TCP	Outbound	Sm@rtServer	VNC client connection
VNC server	5800*	TCP	Inbound	Sm@rtServer	VNC server connection HTTP
	5900*	TCP	Inbound	Sm@rtServer	VNC server connection
SIMATIC Logon	16389*	TCP	Outbound	UMAC (User Management to the Access Control)	Register / log on to a remote server.
Allen Bradley Ethernet IP	44818	TCP	Outbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.

Communication connections for Panels and WinCC Runtime Advanced					
Reserved	49152 ... 65535	TCP/UDP	Outbound		Dynamic port range is used, for example, to connect to the remote file sharing.
* Default port that can be changed by user configuration ** Port is assigned automatically. *** Supported by WinCC Runtime Advanced only.					

3.2 Notes on use

Contents

Information that could not be included in the online help and important information about product features.

Change in response after upgrading

The new WinCC version contains new features and improvements as compared to the previous version. The new version may therefore demonstrate slight differences in the names or the response of properties and functions after the upgrade. Unlike in previous versions, the standard properties of newly created objects can also be changed.

These differences in response may lead to errors when compiling the project.

Upgrading with subsequent device replacement (RT Advanced)

If you upgrade a project with a "Runtime Advanced" device and then replace the device with a panel with HMI version 14.0.1.0 or 15.0.0.0, the configured data can be lost. After upgrading, it is recommended that you either:

- First change the device to an IPC without adapting the screen and then switch back to a PC station with Runtime Advanced. You can then change the device to a panel without losing data.
- First change the device to a panel with an HMI device version lower than 14.0.1.0. You can then change the device to a panel with the desired device version without losing data.

Working with standard user rights

If you are working with standard user rights in Windows 7, "User Account Control (UAC)" must not be disabled.

The "User Account Control" is enabled in Windows 7 by default.

For more information on the "User Account Control", refer to the online help for Windows 7.

On-screen keyboard (RT Advanced)

Once you have opened the TIA Portal, you can no longer access the on-screen keyboard.

To access the on-screen keyboard in Windows, use the following command: "Start > All Programs > Accessories > Ease of Access > On-screen keyboard".

Installation on a computer with the regional setting "Turkish" (RT Professional)

If the computer is operated with the regional setting "Turkish" at the time of the installation, WinCC Runtime Professional cannot be started.

Device change – character sets (RT Professional)

If you have configured different fonts on screen objects for multilingual texts in the various project languages, you should check the screens after an HMI device replacement from a panel to a PC-based Runtime. The configured fonts may change if the device is replaced.

Texts in the simulation (RT Professional)

If a Runtime language which does not correspond to an installed product language is used in the simulation, system texts are shown in an installed product language. Depending on the operating system and the installed product languages, this will be either English or Chinese.

If you wish to see all texts in Japanese, Korean or Taiwanese in the relevant language in the simulation, you should first install WinCC without simulation. Then install WinCC Runtime Professional with the required languages.

Online delta loading with user-defined cycles (RT Professional)

If you have configured new custom cycles in the "Cycles" editor, "Load > Software (compile all)" command when loading the project. Online delta loading is not possible.

License transfer via S7USB

You always need to run WinCC to transfer a license to a panel via S7USB.

Transferring licenses to a panel on 64-bit operating systems

If you are running a 64-bit operating system and the "Edit > Connect target systems > Connect HMI device" menu command is not available in Automation License Manager, open command line input and run the following command with administrator rights:

"%WINDIR%\system32\RegSvr32.exe" "%CommonProgramFiles%\siemens\AlmPanelPlugin\ALMPanelParam.dll"

Installing East Asian project languages on a PC without an Asian operating system (Basic)

If you select an East Asian project language on a PC that does not have an Asian operating system installed, the default font is marked as invalid in the "Runtime settings > Language & font" editor.

To resolve this problem, open the "Regional and Language Options > Languages" dialog in the Control Panel and enable the "Install files for East Asian languages" option.

Operation on a network server (RT Professional)

The operation of WinCC Professional on a network server (e.g., domain controller, file and name utility server, router, software firewall, media server, exchange server) is not permissible.

File browser on a Windows 10 PC with touchscreen

You can only operate the file browser dialog with a mouse, keyboard or on-screen keyboard (without using the touch function) on a Windows 10 PC with touchscreen. We recommend using the file browser dialog of the Windows operating system with the help of a script on a touchscreen PC with Windows 10.

PostScript fonts

Only fonts that contain TrueType data can be used in the TIA Portal. PostScript fonts and open type fonts that contain PostScript data are not supported.

3.3 Migration

Contents

Information that could not be included in the online help and important information about product features.

Migrating an integrated project with ProTool objects

If you migrate a WinCC flexible project integrated in STEP 7 and the error message "PROTOOL option package(s) missing in STEP 7" is output, you have installed WinCC flexible 2008 SP3 or later. Moreover, the project still contains objects that were configured using ProTool. Do not open the project with WinCC flexible. To migrate the project, proceed as follows:

1. Open the project in the SIMATIC Manager.
2. Execute the "Save as" command in the "File" menu.
3. Enable the option "With reorganization" in the "Save project as" dialog.
4. Click OK.
5. Copy the project back to the original computer.
6. Restart the migration.

Migrating a WinCC V7 project: Border line of rectangles (RT Professional)

In a WinCC V7 project, you have configured a rectangle with the settings "Line weight = 1" and "Draw insider border = yes".

You migrate the WinCC V7 project to WinCC V15. To have the rectangle displayed correctly, follow these steps.

1. Open the Inspector window of the rectangle.
2. Open the property list.
3. Disable "Widen border line inwards".

Progress bar

As long as the progress bar still shows a value of 100%, the software is still busy running remaining tasks such as the closing of references. The software will not respond to user input while this status is given.

Supported project languages in WinCC V7 (RT Professional)

Due to different Windows DotNet versions, there may be different names in WinCC V7 and the TIA Portal within the same language. This can result in different sorting orders in the same project.

The following table shows some examples with English spelling:

WinCC V7.x	TIA Portal
Bengali (Bangladesh)	Bangla (Bangladesh)
Bengali (India)	Bangla (India)
Chinese (Simplified, PRC)	Chinese (People's Republic of China)
Chinese (Simplified, Singapore)	Chinese (Singapore)
Chinese (Traditional, Hong Kong S.A.R)	Chinese (Hong Kong S.A.R)
Chinese (Traditional, Macao S.A.R)	Chinese (Macao S.A.R)
Chinese (Traditional, Taiwan)	Chinese (Taiwan)
French (Monaco)	French (Principality of Monaco)
Oriya (India)	Odia (India)
Sami (Inari, Finland)	Sami, Inari (Finland)
Sami (Skolt, Finland)	Sami, Skolt (Finland)
Sami, Northern (Norway)$Sami	Sami, Northern (Norway)
Spanish (Bolivarian Republic of Venezuela)	Spanish (Venezuela)
Spanish (Spain, Traditional Sort)	Spanish (Spain)
Spanish (Spain, International Sort)	Spanish (Spain)
Yakut (Russia)	Sakha (Russia)

3.4 Engineering System

3.4.1 Screens and Screen Objects

Contents

Information that could not be included in the online help and important information about product features.

Screen objects after HMI device replacement

If you upgrade a device to the new HMI device versions, you should check the screens contained in the project. Because of the new appearance and improved operability, texts of symbolic I/O fields may not be completely legible and may be concealed by operator controls.

Grouping of screen objects

When you group screen objects in WinCC, performance problems can arise in WinCC in the case of large nesting depths.

ActiveX and .NET controls

For reasons of stability and performance, not all registered Active X controls can be configured in WinCC. Only the controls visible in the selection dialog can be used in WinCC.

ActiveX and .NET controls are always positioned in the foreground in runtime.

The configuration of ActiveX and .NET controls on levels is not supported.

Use of WinCC Control Development

WinCC Control Development can be used without a license.

You can find the current version of WinCC Control Development for download in the Siemens Industry Online Support, entry 109760182 "Control Development Kit" (https://support.industry.siemens.com/cs/de/de/view/109760182).

Border line of rectangles (RT Professional)

In a WinCC V7 project, you have configured a rectangle with the settings "Line weight = 1" and "Draw insider border = yes".

You migrate the WinCC V7 project to WinCC. To have the rectangle displayed correctly, follow these steps.

1. Open the Inspector window of the rectangle.
2. Open the property list.
3. Disable "Widen border line inwards".

Preview in screen window (RT Professional)

You use your own designs with shadows for screen objects. The screen objects can be displayed in a screen window.
The shadows of the screen objects are not displayed in the preview of the screen window. The response occurs only in the engineering system. It is displayed correctly in runtime.

Many visually different screen objects

The use of screen objects with many visually different properties (e.g. many different styles) can reduce performance in runtime and can increase the amount of available memory space used. Avoid using, for example, very many different corner radii: 0 pixel, 1 pixel, 2 pixels, 3 pixels, etc.

Dynamization of grouped objects (RT Professional)

For groupings with multiple nestings (group in group, faceplate in group, group in faceplate, etc.), only the events of the outermost group and the innermost objects can be used for dynamization with system functions. System functions that are configured at events of the lower-level group or lower-level faceplate are not executed.

The configuration of simultaneous motion animation of an object group within this group can be executed in runtime in a way other that what is expected. Check the sequence of the execution in runtime and adapt the configuration if necessary.

Number of characters in text fields, lists and alarm texts

The number of characters that can be used in the text of a screen object is not constant. Control instructions and formatting are taken into consideration when entering text data and the maximum number of characters is reduced accordingly.

Transparency in WinCC as of V13

Transparent graphics can be displayed without any problems in runtime. This is true for all Comfort Panels and WinCC Runtime Advanced as of version 13.0.
To use the transparency in a graphic view or in a graphic I/O field, the "Fill pattern" property must be set to "Transparent" and the property "Use transparent color" must be disabled.

If the property "Use transparent color" is enabled in a device as of version 13.0, the transparency of the graphic is lost and the transparent pixels are displayed in black. We recommend that you check the "Use transparent color" property at the points of use of transparent graphics after upgrading existing projects to device version 13.0 or higher. If the graphics are not displayed correctly, disable this property.

Displaying SIMOTION PLC websites in the HTML browser

If the SIMOTION PLC websites are not displayed correctly in the HTML browser, insert ""/basic"" after the website URL to display the websites in basic mode.

Display differences between the configuration and the display on the HMI device

The display of the text configured in a screen object may be different on the HMI device due to the display configuration. If you are using the options for automatic size adjustment in the configuration, check the display on your HMI device in every language.

If texts that were configured with the "Fit object to contents" option cannot be displayed in full, they are reduced slightly on the HMI device. If this reduction leads to a distorted display of texts, disable the "Fit object to contents" option and expand the text with additional blanks. Optionally, you can increase the width of the object or use a shorter formulation for the text.

GRAPH overview (RT Professional)

When upgrading to V14, the signature of the C function at the "Click" event is changed. The propertyName parameter from the previous versions has been removed. Functions that access this parameter are shown as faulty after the upgrade.

Dynamization in ProDiag Overview (RT Professional)

Dynamization of the "Allow operator control" property with a variable is not possible in the "ProDiag Overview" object.

Initial value acquisition and criteria analysis: Supported instructions

The FBD instructions "AND" and "OR" are also supported for initial value acquisition and criteria analysis.

The following instructions are supported for initial value acquisition and criteria analysis up to and including the instruction box:

- "Set/reset flip-flop"
- "Reset/set flip-flop"
- TP, TOF and TONR timers
- CTU, CTD, CTUD counters

Updating faceplates with the UDTs of the type "IEC timer"

The "ST" element was automatically removed from the user-defined PLC data types (UDT) that use the "IEC timer" data type. Adapt the faceplates containing these UDTs to a new version of the UDT without an ST element.

A detailed description of this can be found in the FAQ with the entry ID "109740393" in the SIMATIC Customer Online Support.

Higher-level controls with visibility animation

The higher-level configuration of table-based controls with visibility animation can lead to problems for touch operation with devices using the operating system Windows CE/Embedded Compact.

SVG graphics

The SVG graphics are converted to Siemens SVG Standard. Note the following restrictions when using SVG graphics:

- The CSS definitions are converted to inline attributes.
- The embedded scripts and non-local URL links are not supported in the SVG graphics and are removed from the original graphics during conversion.
- The use of SVG graphics with embedded graphics and animations is not supported.
- The use of large SVG graphics affects performance due to the load associated with the increased characters.

The representation of SVG graphics depends on the browser used.

Editing SVG graphics

It is not possible to open SVG graphics in an external editor using the "Edit" command.

System diagnostics view: "HMI access" access protection

When using the "HMI access" protection level in conjunction with the SIMATIC S7-1200 PLC, correct display of the diagnostic data can only be ensured as of firmware version 4.1.

Fonts in the header of window objects

The font that you configure in the runtime settings of the device is not used in the alarm window, system diagnostics window and dialog headers.

Invalid graphic types in graphics lists after upgrading (WinCC Runtime Professional)

When you upgrade your project to TIA Portal version V15, invalid graphic types in the graphics list result in compiling errors that were displayed as warnings in V14.

To avoid error messages during compiling, you need to manually correct the graphic type in the graphics list if necessary.

Multilingual graphics in faceplates

In a faceplate, insert a graphic in a screen object. This graphic is used in all specified project languages.

If you open the graphic using the shortcut menu command "Edit" with the option "Edit graphic centrally", the graphic is changed only for the currently set project language after editing and saving in an external editor, such as MS Paint. The change of the corresponding graphic is visible in all objects that use this graphic.

Use of multilingual graphics (RT Professional)

Graphics are stored in the runtime project in the language that you specified in the Runtime settings under "Language & font > Language selection > Runtime language for non-multilingual objects".

The use of of different graphics for individual runtime languages has no effect when switching languages in runtime.

Remedy: Configure individual graphic displays with the desired graphic in the set runtime language for non-multilingual objects and switch these to visible or invisible according to the runtime language in the script.

PLC code view (RT Professional)

The button "Show whole function block" of the PLC code view can only be executed correctly when the names of the following objects do not contain any special characters, such as a comma:

- PLC name
- DB/FB name
- DB/FB interface element name

Network changes in the PLC code display (RT Professional)

When a ProDiag alarm is pending, network changes in the PLC code view are updated in runtime.

In order for network changes in the PLC code display to be displayed correctly for criteria analysis, the message must be triggered again.

PDF view (RT Advanced)

With the PDF view, no pop-up notes (sticky notes) can be displayed in PDF files.

See also

http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/62101921)

3.4.2 Tags and connections

Contents

Information that could not be included in the online help and important information about product features.

Multiplexing tags (Basic Panels)

If you multiplex a tag with an external tag on a Basic Panel, the address is read from the PLC in runtime during the first read cycle. The value of the address read is not available until the second read cycle.

Tag names in faceplates (RT Professional)

Use of the "." or "@" character in names of tags in faceplate types is not permitted. Do not use these special characters in the tag names in faceplates.

Tags with symbolic addressing and "Char Array" data type (RT Professional)

Tags with symbolic addressing and the "WString" data type are not released for communication of RT Professional and SIMATIC S7-1200 V3.

Length specification for tags of the "String" type (RT Professional)

The length specification for the internal HMI tags of the "String" type have no effect in the "Tag" editor. If you want to limit the tag length of an input field or output field, for example, limit the length using VB scripting or C scripting.

Tag of the "String" type for connections with SIMOTION PLCs (Panels, RT Advanced)

The length of the String tags must not exceed 210 characters for SIMOTION controllers.

It is possible to create a String tab with more than 210 character in the engineering system. However, the tag in question is not written in Runtime. A message stating that the value limit is outside the tag appears.

In this case, adjust the length of the String tag.

Tags under "Tag box ID"

The tag that you configure under "Tag box ID" with "Runtime Settings > General" may only be used in read-only mode.

No values may be written in this tag, however, they may be read in scripts, for example, or output to screens.

Array elements in WinCC

If you have connected an HMI tag with an array from a STEP 7 data block which does not start with a low limit of 0, the array elements are mapped in WinCC to the low limit of 0.

To ensure that you do not have to rethink between the STEP 7 indices and the WinCC indices when accessing the individual array elements, the low limits of arrays should also start at 0 in STEP 7.

Use of 64-bit data types

The use of 64-bit data types may result in a slight loss of accuracy as these data types are mapped in the HMI channel to the data type Double. It is therefore possible that integer data types are displayed with decimal places.

System diagnostics of PLCs in the R/H system in non-redundant mode

Two PLCs and two corresponding connections are configured in an R/H system. If the R/H system is not in redundant mode, it may happen that the system diagnostics views of the connections show different diagnostics status symbols for the same PLC.

3.4.3 Alarm system and alarm displays

Contents

Information that could not be included in the online help and important information about product properties.

Boolean tags in alarm logs

Bool type tags are recognized as 0 and -1 in the alarm log. If you use a text list that is controlled by a tag of the type Bool in an alarm log, then add the entry for the value -1 to the corresponding text list.

PLC alarms

A PLC alarm is only visible in the "Alarms" editor of an HMI device if there is an instance data block for this FB.

Controller alarms in runtime (WinCC Professional)

If upgrade a project from versions V13SP1 or V13 SP2 to versions V14, V14 SP1, V15.x, you need to compile the project and load it in the controller and HMI device. Otherwise, the controller alarms are shown crossed out in the alarm view.

Controller alarms in runtime (Panels and RT Advanced)

If the property "Alarm > Basic settings > Information only" was selected for a controller alarm, it was not displayed in the alarm view in WinCC.

"Automatic update" of controller alarms activated on the HMI device (Panels and RT Advanced)

For controller alarms that are displayed in the alarm log, the following applies:

The info texts of the alarms are not logged and show "###Text missing###" when an alarm log is displayed in runtime.

Copying projects with integrated connections

If you copy the devices with their integrated connections (between HMI devices and controllers) from a project to a global library and insert it from there in a new project, then the information about controller alarms can get lost on the HMI device. To avoid this loss of information, first copy all controllers and only then all HMI devices to the new project.

Alarm 20013 on global scripts (Panels and RT Advanced)

In contrast to the description given in the help, the alarm with the number 20013 is only output on Runtime start if the script engine has not yet been initialized.

This alarm has no effect on the functioning of VB scripts in Runtime.

Logging alarm texts with more than 6 values (Panels and RT Advanced)

If an alarm text contains more than 6 values, only the first 6 values are saved in the log file after logging.

An alarm view configured for the display of the logs shows all values contained in the alarm text.

3.4.4 System functions and scripts

Contents

Information that could not be included in the online help and important information about product features.

Scripts for service projects (RT Professional)

Since no interactive user is usually logged in service projects, C scripts and VB scripts lead to problems in the following cases:

- When the scripts require interaction, for example, operator input.
- When the scripts show message boxes.

There is no common data area for C scripting in service mode. Thus, for example, no global C variables can be exchanged between the "Scheduled tasks" and "Screens".

Trigger tags of user-defined functions (RT Professional)

If you have configured an HMI or PLC user data type on a screen, you cannot use the elements of the user data type as trigger tags in newly created or changed user-defined user functions.

Find and Replace in local scripts (RT Professional)

Use the "Find and Replace" function to find and replace texts within editors. It is not possible to use Find and Replace in local scripts.

User-defined functions in faceplates

If you want to access tags in a faceplate with a user-defined function, you cannot dynamically compose the name of the tag.

"SimulateTag" system function (Basic Panels)

If you use the system function "SimulateTag" with a short cycle time on a Basic Panel, the HMI device may be overloaded.

System functions of the bit processing (Mobile Panels and RT Advanced)

Because of events it is not possible to manipulate bit arrays. System functions, e.g. InvertBit, SetBit etc., must not contain any elements of a bit array as argument.

Properties no longer used (Comfort Panels and RT Advanced)

In WinCC V13, many new visual properties have been introduced and many existing properties have been harmonized.

In the course of revision of the object properties, some properties that could be used in VB scripts in earlier versions of WinCC are no longer supported for use in WinCC V13. Scripts that use properties that are no longer supported remain valid, but the relevant calls within the scripts have no function. These limitations pertain exclusively to Comfort Panels and RT Advanced in device version V13.0.

The following table shows the properties of screen objects that are no longer being used:

Property	Screen object
BorderBrightColor3D	Slider
BorderColor	Text field, I/O field, symbolic I/O field, date/time field
BorderInnerWidth3D	Slider
BorderOuterWidth3D	Slider
BorderShadeColor3D	Slider
BorderStyle3D	Text field, I/O field, button, symbolic I/O field, graphic I/O field, date/time field, bar, switch
BorderWidth	Text field, symbolic I/O field
CenterColor	Gauge
DialColor	Gauge
EdgeStyle	Text field, I/O field

Debugging VB scripts

You can use the "Debug" object to enter custom texts in a debugger. The object has the methods "Write" and "WriteLine". You can find additional information about the "Debug" object on the Internet (https://msdn.microsoft.com/en-us/library/wk3y866c(v=vs.84).aspx).

You can use the "Err" object to respond to error messages that are generated by another object. The object has the methods "Raise" and "Clear".

You can find additional information about the "Err" object on the Internet (https://msdn.microsoft.com/en-us/library/sbf5ze0e(v=vs.84).aspx).

Transferring parameters to VBScripts (Panels and RT Advanced)

In WinCC, you can define whether the parameters of VB scripts "ByRef" (address of the parameter) or "ByVal" (value of the parameter) are transferred.

Transferring the address of the parameter using "ByRef" (Call by Reference) is possible only for the internal tags of the VB script. With HMI tags, the value of the parameter is always transferred regardless of whether the parameter was defined as "ByRef" or as "ByVal".

System functions with the parameter "Processing status".

For system functions, such as "SaveDataRecord" for example, you can only specify HMI tags for the "Processing status" parameter.

Functions of the PLC code view (RT Professional)

The functions of the PLC code view "ShowCodeViewerFromAlarm" can only be executed correctly when the names of the following objects do not contain any special characters, such as a comma:

- PLC name
- DB/FB name
- DB/FB interface element name

Accessing the "Height" and "Width" properties of some VBS objects in runtime

Contrary to what is described in the help, you have read access to the "Height" and "Width" properties for the following VBS objects:

AlarmView, Bar, BatteryView, Button, CameraControl, Clock, DateTimeField, FunctionTrendControl, Gauge, GraphicIOField, GraphicView, HTMLBrowser, IOField, Line, MediaPlayer, PDFView, PLCCodeViewer, Polygon, Polyline, ProDiagOverview, ProtectedAreaNameView, RangeLabelView, RangeQualityView, RecipeView, S7GraphOverview, Slider, SmartClientView, StatusForce, Switch, SymbolicIOField, SymbolLibrary, SysDiagControl, TextField, TrendView, UserView, WLanQualityView, ZoneLabelView, ZoneQualityView

With the following objects you have no access to the "BorderColor" property for Runtime Advanced:

TextField

You can find additional information on the VBS object model in the FAQ with entry ID "53752382" in the SIMATIC Customer Online Support:

https://support.industry.siemens.com/cs/ww/en/view/53752382 (https://support.industry.siemens.com/cs/ww/en/view/53752382)

3.4.5 Reports

Contents

Information that could not be included in the online help and important information about product features.

Printing reports (Panels)

Depending on the printer used, the objects placed at the edge of the reports can be cut off during printing.

Make sure that your report has a suitable size.

3.4.6 Recipes

Contents

Information that could not be included in the online help and important information about product features.

Arrays in recipe elements

If you have configured both an array as well as the elements of this array for recipe elements of a recipe, the loading of data records aborts with the following error message: "290055: Import of data records aborted with error"

Use either the array or just the array elements for recipe elements of a single recipe.

3.4.7 User administration

Contents

Information that could no longer be included in the online help and important information about product features.

3.4.8 Communication

Contents

Information that could not be included in the online help and important information about product features.

Establishing a connection to WinCC Runtime Advanced

Please ensure that the device version configured in the Engineering System matches the installed Runtime version; otherwise, communication when the project is loaded will not be established or will be delayed.

If the device version does not match the installed Runtime version, you have the following options:

1. Change the device version configured in the Engineering System.
2. Install the version of WinCC Runtime Advanced corresponding to the configured device version.

Connection changing for non-integrated connections

If you want to change the communication from a non-integrated connection to another non-integrated connection, for example from Omron to Mitsubishi, you must ensure that the data types of the tags are also supported in the new connection.

Connection interruptions with Mitsubishi PLCs

After multiple connection interruptions, a situation may arise where all the connection resources of the Mitsubishi PLC are in use and the connection can no longer be established. We recommend that you check these connection resources in the PLC program of the controller and enable them again.

Self-signed certificates (RT Professional)

Self-signed certificates are no longer supported.

If a self-signed certificate is detected on the server side, the authentication takes place via the previously distributed PSK key (pre-shared key). If the PSK keys of the communication partners do not match, no communication is established.

Communication between WebClient and Sm@rtServer

Java is no longer supported by future browser versions.

In these cases, use the following alternatives as Sm@rtClient:

- Sm@rtClient.exe
- Sm@rtClient app for smartphones

Accuracy of the "DTL" data type

The "DTL" data type supports time information down to the nanosecond range. Because panels only support time information down to the millisecond, the following restrictions arise when used at the area pointers:

- Area pointer "Date/time"
  For the transmission of time information from a panel to the PLC, the smallest unit of time is 1 millisecond. The value range from microseconds to nanoseconds of the "DTL" data type is filled with zeros.
- Area pointer "Date/time PLC"
  For the transmission of time information from a PLC to a panel, the range from microseconds to nanoseconds is ignored. The time information down to and including milliseconds continues to be processed on the panel.

Limited number of possible HMI connections

An error message is displayed during compilation of a device indicating that the configuration of the HMI connection in the "Devices & Networks" editor is invalid. The reason may be that the maximum number of possible connections of the HMI device or PLC has been exceeded.

Check the maximum number of available connections. Consult the device manuals of the devices you are using.

Use of PROFINET IO with panel HMI devices

When using PROFINET IO to connect the direct keys and LEDs of HMI devices to the PLC, you can define an offset for the address area of the inputs and outputs during configuration in HW Config.

The following restriction applies when a PROFINET IO-capable S7-400 CPU is used with one of the HMI devices listed below:

The offset for the start of the address area of the inputs must not be greater than the offset for the start of the address area of the outputs.

The restriction applies to the following HMI devices:

- Mobile Panel 177

To configure the address parameters, open the PLC that contains the CPU of the 400 series in HW Config. Select the HMI device that is connected via PROFINET IO from the station window of HW Config. A table with the properties of the HMI device is displayed at the bottom of the station window in the detail view. Select the line containing the addresses of the HMI device in the table and open the object properties using the shortcut menu.

Enable the "Addresses" tab in the "Object properties" dialog. Configure the offset for the inputs under "Inputs > Start". Configure the offset for the outputs under "Outputs > Start".

RT Advanced communication via Station Manager (SIMATIC NET) with an S7-1200

For communication of a SIMATIC S7-1200 with a PC with WinCC RT Advanced via a router, the following restrictions apply to the PC:

- Windows 7: Only with installed SIMATIC NET 8.1

These restrictions also apply when you use the Station Manager. Connections with the help of the Station Manager of Runtime Advanced are always treated as routed connections.

Changing the IP settings and device name of a PLC in the Control Panel of the HMI device (Basic Panels)

The Control Panel is open in the "Service and Commissoning > IP-Adaptation" menu on the HMI device. If you want to change the IP settings or the device name of a PLC, please note:

In the Engineering System, you need to have selected the following options in the Inspector window of the PLC under "Properties > General > PROFINET interface > Ethernet addresses":

- "Set IP address using a different method"
- "Set PROFINET device name using a different method"

HMI connections to SIMATIC S7-1200 PLCs

HMI connections to SIMATIC S7-1200 PLCs with firmware versions earlier than V2.0 are not possible.

"Set the IP suite (address) of the PLC in the Control Panel" with SIMATIC S7-1200 V1 (Basic Panels)

The function "Set the IP suite (address) of the PLC in the Control Panel" has not been approved for the following PLCs:

- SIMATIC S7-1200 V1

Connections via PROFIBUS DP

When a connection between a PLC and an HMI device via PROFIBUS DP is interrupted and then re-established, sporadically all other PROFIBUS DP connections in the communication network are interrupted and re-established.

De-energize the disconnected station before reconnecting it.

Switching a connection

A connection may be interrupted when it is switched from an HMI device to a SIMATIC S7-300/400, SIMATIC S7-1500 or SIMATIC S7-1200 controller.

Note the following settings in the SIMATIC S7-1500 or SIMATIC S7-1200 PLCs:

- Absolute addressing of tags
- The "Disable PUT-GET communication" option must be selected
- The "Complete protection" protection level may not be set

Communication of Runtime Professional with SIMATIC S7-1200

In productive operation, communication of WinCC Runtime Professional with SIMATIC S7-1200 is only released for single-user systems.

RT Advanced in the Station Manager

If RT Advanced and WinAC RTX use the same communication processor of a PC, HMI communication with SIMATIC S7-1200 and SIMATIC S7-1500 is not possible.

Raw data communication in redundant projects (RT Professional)

Simatic.NET, Named Connections and various communication blocks, such as BSEND/BRCV, for example, can only be used to a limited extent in a redundantly configured PC station because the connection parameters for the redundant partner server cannot be configured.

Non-integrated connection to a SIMATIC S7-1500 software PLC (RT Advanced)

A non-integrated connection between an HMI device and a SIMATIC S7-1500 software PLC is not supported in WinCC.

Transfer areas of the operating mode IO device and DP slave of the HMI devices

If you have activated the operating mode "IO device" or "DP slave" for HMI devices, no transfer areas should be added or deleted in the properties of the HMI device. If you have inadvertently deleted or added a transfer area, disconnect and reconnect the controller.

Getting diagnostics data by means of HMI devices

In contrast to the description of diagnostics icons for modules and devices in the information system, the following icon has the meaning specified here for HMI devices:

Icon	Meaning
	No diagnostics data is available.

PROFIenergy communication

To set up PROFIenergy communication, contact Customer Support.

3.5 System-wide functions

Contents

Information that could not be included in the online help and important information about product features.

Using system diagnostics in the device proxy

To use the system diagnostics function in an IPE device proxy, for example, a system diagnostics view, insert the PLC alarms as content of a device proxy.

Initialize the device proxy with data from a project of the previous version

A device proxy cannot be initialized with the data from a project of the associated full version in a project with the Service Pack version.
Upgrade the source project to the Service Pack version to initialize the device proxy in the target project with the data from the source project.

Import and export of library texts

The following objects are not supported when importing and exporting library texts:

- Import: Recipes, Recipe elements, Recipe data records, HMI alarms, Text list entries, Screens, Reports.
- Export: Text list entries, Screens, Reports.
- Import and export of library type "Screen".

3.6 Compiling and loading

Contents

Information that could no longer be included in the online help and important information about product features.

Compiling and loading

If internal errors or warnings occur during compiling, compile the complete project using the command "Compile > Software (rebuild all)" in the shortcut menu of the HMI device.

Before you start productive operation with your project, compile the entire project using the "Compile > Software (rebuild all)" command from the shortcut menu of the HMI device.

If you are using HMI tags that are connected to control tags in your project, compile all modified blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Incorrect installation of ProSave

If you receive an error message during installation of ProSave when loading data to a target device or maintenance of an HMI device, then you cannot remedy this error using the repair function of setup. Remove ProSave via the Control Panel. Then start setup and install the "ProSave" component again.

Checking the address parameters

When an HMI device is compiled in the project tree with the command "Compile > Hardware and software (changes only)" in the shortcut menu, the address parameters of the HMI device, such as the IP address, are not checked. If you want to ensure that the address parameters are checked as well, you will have to compile the HMI device using the "Compile" button in the "Devices & Networks" editor of the toolbar.

Error message when downloading data to the PLC

A panel and a PLC are connected and communicating with other.

If a tag is accessed while downloading data from the panel to the PLC, an error message is displayed on the panel.

Delayed reaction in the "Extended download to device" dialog

If the settings in the "Extended download to device" dialog for "Type of the PG/PC interface" and "PG/PC interface" do not match the settings on the HMI device, this can result in the application not responding for up to a minute.

Extended download with an S7-1200 and a Comfort Panel

An S7-1200 PLC and a Comfort Panel are located in the same physical network as the PG/PC. You open the "Extended download to device" dialog for the Comfort Panel.

If you enable the option "Show all accessible devices", it may occur that the application stops responding.

Project transfer via USB

If you have connected more than one HMI device via USB to your configuration PC, project transfer is only possible to the last connected device.

Comfort panels as of device version 13.0: Backing up data while loading projects

If the transfer is interrupted for Comfort Panels with a device version 13.0 or higher, WinCC automatically ensures that no data is lost and that existing data is only deleted on the HMI device after complete transmission.

Displaying characters in transfer alarms

If characters are not displayed correctly in the transfer alarms during transfers to HMI devices with the device version V12 or older, please check the region and language settings in Windows. Set the corresponding language under "Language for non-Unicode programs".

Missing fonts (panels)

The following fonts no longer belong to the standard installation pack in Windows 10 and are not pre-installed:

- MS PGothic (is used as standard for displaying Japanese characters)
- Gulim (is used as standard for displaying Korean characters)
- MingLiU for Chinese (is used as standard for displaying Chinese characters for Taiwan, Hong Kong and Macao)

If a font is not installed, an error message is displayed in the engineering system. The error message appears in every TIA Portal session if you navigate to “Runtime settings > Language and font” or open screen objects which use the missing font. The missing font is replaced with a suitable font by the TIA Portal.

Missing fonts cause problems during compilation and in Runtime.

For example, if you open an error-free V13 SP1 project that was created on a configuration PC with Windows 7 on a configuration PC of V14 SP1 with the Windows 10 operating system, error messages can be output during compilation because of the fonts that are not installed.

You can find more detailed information on fonts in Windows 10 at https://support.microsoft.com/en-us/kb/3083806 (https://support.microsoft.com/en-us/kb/3083806)

Solution

1. Compile the project before you transfer it to your HMI device. How to determine whether the fonts you require are missing.
2. Install these fonts and/or the corresponding Windows language pack.

Alternatively, you can also compile the projects with substitute fonts.

The missing font is saved in the configuration. As soon as this font is available again, it is displayed and used in the configuration.

Reducing the project size

When you compile your HMI device, an alarm appears informing you that the size of your project is approaching the system limits for the corresponding HMI device. In this case, perform a complete compilation of the software to reduce the project size. To do this, use the command "Compile > Software (rebuild all)".

3.7 Runtime

3.7.1 Notes on operation in Runtime

Contents

Information that could not be included in the online help and important information about product features.

Focus in runtime

If you have configured a low-contrast combination of focus color and border color for an HMI device with version 12.0.0 or earlier, the focus may no longer be identifiable in runtime after you change the device version in the TIA Portal. Change one of the two colors.

Language behavior - Layout of on-screen keyboard

The layout of the on-screen keyboard does not change when you switch to a runtime language that is not installed for the keyboard layout.

In this case, the language setting for the keyboard remains set at the most recent valid language or the language setting for the default keyboard layout of Windows is used.

Tag values exceed the maximum length

You enter a character string in a string tag via an I/O field. If the character string exceeds the configured number of tags, the character string will be shortened to the configured length.

Empty alarm texts

Runtime is running with a project. The project is saved on a network drive.

In the event of interruptions to the network drive connection, Runtime may attempt to load alarm texts from the network drive.

In the event of disconnection, the alarm window or the alarm view remains empty.

To avoid this, copy the project to a local drive before the starting the project in Runtime.

Duration of log initialization (Panels, RT Advanced)

Initialization of the logs on some storage media can take up to 5 minutes. The successful completion of initialization is immediately confirmed by a system event. If there is no storage medium for logging when Runtime starts, the appearance of the system event can also take up to 5 minutes.

Large logs delay the ending of Runtime (Basic Panels 2nd Generation)

When very large logs are used, ending Runtime can take a long time. Use segmented logs as an alternative to very large circular logs.

Slow reaction of SmartServer

The following programs may start and respond very slowly with Windows 7:

- HMI TouchInputPC
- SmartServer: <Ctrl+Alt+Del> shortcut in the logon dialog

The delay is caused by the callback for the Internet certificate validation.

Remedy:

You can find the following files
on the product DVD under:
Support\Windows7\CRL_Check or CD_RT\ Support\Windows7\CRL_Check\:

- DisableCRLCheck_LocalSystem.cmd
- DisableCRLCheck_CurrentUser.cmd

1. Run the "DisableCRLCheck_LocalSystem.cmd" file with administrator rights. Select the command "Run as administrator" from the shortcut menu of the file.
2. Reboot the PC.

If the problem persists, follow these steps:

1. Double-click the file and run the "DisableCRLCheck_CurrentUser.cmd" file with user rights.
2. Reboot the PC.

Note

The callback for the certificate validation is disabled for all users or PCs. To restore the original state, perform the following files:

- RestoreDefaults_LocalSystem.cmd
- RestoreDefaults_CurrentUser.cmd

You can find the files in the following directory of the product DVD:

- Support\Windows7\CRL_Check or CD_RT\Support\Windows7\CRL_Check\

Ending screensaver on the Sm@rtServer

When the screensaver is active on the Sm@rtServer on the server HMI device, you require write access to the Sm@rtClient side in order to end the screensaver on the server HMI device.

Avoiding corrupt files during power failure

If a power failure occurs in Windows systems while the WinCC system is active, files may be corrupt or destroyed. Operation with the NTFS file system provides better security.

Secure, continuous operation is only ensured by using an uninterruptible power supply (UPS).

3.7.2 Notes on operation of panels in Runtime

Contents

Information that could not be included in the online help and important information about product features.

Using the mouse wheel in Runtime

The use of the mouse wheel in Runtime is not supported on all panels.

Backup on the memory card of the PLC (Basic Panels)

Create the backup file "A.psb" on the memory card of the PLC. An error, for example a connection break, occurs when creating the backup.
This will create a corrupt file on the memory card of the PLC. Such a file has "~$" as prefix. Delete the file with the prefix "~$" if you want to save a backup again under the same name "A.psb".

Panel Data Storage and S7-1500F (Basic Panels)

The "Panel Data Storage" PDS function cannot be used on Basic Panels in conjunction with S7-1500F when the password for the protection level "Full access incl. fail-safe" is used.

"Panel Data Storage" function (Basic Panels)

The "Panel Data Storage" (PDS) function provided by Basic Panels is only supported by SIMATIC S7-1200 as of firmware V4.0 and SIMATIC S7-1500. For the PDS function, the panel must be connected directly with the CPU and must not be connected via the CP.

Fonts with file extension ".ttx"

The fonts with the file extension ".ttx" are available on all panels and Runtime Advanced and can be correctly displayed on all devices, for example, the "WinCC_flexible_smart" font.

Languages in Runtime (Comfort Panels)

If the language set under "Region and Language" is not supported in Runtime on the HMI, all dialogs are displayed in the default language "English".

Mass data transfer with USB

If you have installed Windows 10 on the configuration PC, the "USB" transfer channel is not available.

This restriction applies to the following devices and tools:

- Basic Panels: KP300 Basic, KP400 Basic, TP1500 Basic
- Mobile Panels: 170 series and 270 series
- WinCC Engineering System
- "ProSave" service tool

Note

The "USB" transfer channel remains available if Windows 7 is installed on the configuration PC.

Solution

Use other transfer channels to transfer the project data, such as the "Industrial Ethernet" or "PROFIBUS DP" channel.

Modifying the IP address and device name for Panels

On Comfort Panels, the IP address or the device name can be modified even during runtime.

On Basic Panels, modification of the IP address or the device name during runtime is not possible.

3.7.3 Notes on operation of Runtime Advanced

Contents

Information that could not be included in the online help and important information about product features.

Starting Runtime

Only WinCC Runtime V15 can be started in TIA Portal V15. WinCC Runtime V11.02, V12, V13, V13 SP1, V14 and V14 SP1 can be simulated in TIA Portal V14 SP1.

Screen saver on computers with Windows 10

On a computer with Windows 10, an activated screen saver is no longer terminated when an alarm of the "error" class is output.

Windows 10 in tablet mode

It is not recommended to use Windows 10 in tablet mode.

When running applications in tablet mode in Windows 10, for example Runtime Advanced, Sinumerik Operate, etc., these applications are minimized after starting.

Authorization for starting Runtime (RT Advanced)

On a computer running the 32-bit version of Windows 7, WinCC Runtime Advanced can only be started if a user is a member of the automatically created group "Siemens TIA Engineer".

.Net-Controls in Runtime

If you have incorporated a .Net control in your project as "Custom .Net control", you have to copy the files belonging to these controls to the installation directory of WinCC Runtime, e.g. "C:\ProgramFiles\Siemens\Automation\WinCC RT Advanced". Otherwise, the control cannot be loaded in Runtime.

Disabling automatic checking for software updates

If the Engineering System is installed together with Runtime on a PC, the operator gets notifications above software updates. For the system to run reliably on a multi-user system, the same software version must be installed on all PCs.

It is possible to disable the automatic checking for software updates and to thus improve performance.

To disable the automatic checking for software updates, go to "Settings > General > Software updates" and clear the "Check daily for updates" check box.

3.7.4 Notes on operation of Runtime Professional

Contents

Information that could not be included in the online help and important information about product features.

User authorizations within the operating system

1. All user must be added to the "SIMATIC HMI" user group. This also applies to users who want to open WinCC projects remotely.
2. The storage folders of the projects must have the NTFS authorizations "SIMATIC HMI" with full access and "SIMATIC HMI Viewer" with read access. The authorizations must be inherited for all subordinate objects.
3. Members of the Windows user group "SIMATIC HMI" should not simultaneously be members of the Windows user group "SQLServerMSSQLUser$<Computername>$WINCC". The members of this group have administrator rights to the SQL server. Therefore remove all Windows users from this group for whom restricted access to the WinCC database is sufficient.

Windows 7: Locking shortcuts

If you want to lock shortcut keys in Windows 7, you must change the group policies in the administrative tools of the operating system.

A detailed description of this can be found in the FAQ with the entry ID "44027453" in the SIMATIC Customer Online Support:

Internet: WinCC FAQ 44027453 (http://support.automation.siemens.com/WW/view/en/44027453)

Undocked toolbars

If the setting "Always on Top" is used for the Windows taskbar, undocked toolbars can be hidden behind the Windows taskbar in Runtime. Follow the steps below to show the toolbars again:

1. Select the "Properties" command in the shortcut menu of the taskbar.
2. Disable "Keep taskbar on top of other windows".

Fault in the connection between the server and the client

If the connection between the server and client is faulty, check the settings of the PG/PC interface. TCP/IP(Auto) should not be used for the "Interface Parameter Assignment Used". Used fixed IP addresses instead.

Fault in the connection between the server and the client

If the computer is simultaneously used as a server with the engineering system and a client cannot establish the connection to the server, you should check the releases set on the server.

1. Exit Runtime on the server.
2. Select the command "Find computer..." in the shortcut menu of the network environment on the desktop of the server.
3. Enter the name of the server as the computer to be found.
4. Open the computer found to see the released directories.
5. Remove all the releases that begin with "WinCC_Project_HMI". Information on this is available in the documentation for the operating system.

This malfunction is caused by the use of the command "Save as..." in conjunction with the startup of the Runtime of this new project. You can make copies of the projects for backup purposes using the "Save as..." command. You should, however, continue to work with the original project.

Starting WinCC Runtime Professional

If Engineering System and Runtime are operated on one computer, Runtime or simulation for a project opened in the Engineering System should only be started and ended using the TIA Portal. Other options, for example, the symbol in the taskbar information area, should not be used.

When you configure autostart on the client, you are required to enter the logon name and password for each registered server project. To assign each server a logon name, the entry of the respective project path must be confirmed with the "Apply" button. This means that you must enter a logon name and password twice for redundant or alternative projects.

Setting the services for the SQL server

In order to ensure the full functionality of the SQL server for WinCC and WinCC Runtime, you need to check the SQL server settings.

1. Start "Programme > Microsoft SQL Server 2014 > Configuration Tools > SQL Server 2014 Configuration Manager" in the Start menu.
2. Click "SQL Server Services" in the tree.
3. Check the services "SQL Server (WinCC)", "SQL Server (WINCCPLUSMIG)" and "SQL Server Browser". "Automatic" must be entered for "Start Mode". "LocalSystem" must be entered for "Log On As". Change the settings, if necessary.
4. Click "SQL Server Network Configuration" in the tree.
5. Click "Protocols for WinCC".
6. Check the "TCP/IP" protocol. "Enabled" must be entered for "Status". Change the settings, if necessary.

"Report system error" in WinCC Runtime Professional

The "Report system error" functionality is limited in WinCC Runtime Professional in conjunction with an S7-300 or S7-400. In an alarm view, only one alarm is ever displayed for each diagnostics type. Alarms on other errors of the same diagnostics type are not displayed.

WinCC interface and 64-bit operating system

The open interfaces of WinCC Runtime Professional do not offer native 64-bit support. Runtime API, VBS and the WinCC OleDB provider are primarily affected by this. To be able to use WinCC interfaces with a 64-bit operating system, note the following:
- VB scripts cannot be started by simply double-clicking. You must explicitly use the 32-bit version under "syswow64\wscript.exe".
- .NET applications that use the WinCC API have to be explicitly compiled as 32-bit applications, not with "AnyCPU", but with "x86".
- C++ applications must not be compiled as 64-bit applications.

Dynamic graphics in WinCCViewer RT

If you use dynamic graphics from a graphics collection, e.g. linked with scripts, these graphics are not updated in the WinCCViewer RT.

Select the "User-defined" option under "Runtime Settings > Graphic Settings".

Runtime Professional: Loading a project without connection to the configuration PC

Before you begin to load a project on the operator station, backup data such as "User administration" and "Recipe data".

1. User administration data are overwritten by default.
   Therefore, configure two buttons in your project for exporting and importing the user administration with the system function "ExportUserManagement". The data only needs to be exported if you have made changes to the user view in Runtime.
2. In the "Load preview" dialog, specify whether recipe data should be overwritten when loading the project.
   You can save recipe data in the recipe view using the operator controls "Export log" and "Import log".

Note that the most recently imported or loaded data are used in Runtime.

Operation on a network server

Operation of WinCC Runtime Professional on a network server (e.g. domain controller, file server, name utility server, software firewall, media server, exchange server) is not permitted.

Connection with S7-1200 in system diagnostics

Only a WinCC Runtime Professional HMI system can be connected with an S7-1200 (up to V3) when you use system diagnostics.

Disabling automatic checking for software updates

If the Engineering System is installed together with Runtime on a PC, the operator gets notifications above software updates. For the system to run reliably on a multi-user system, the same software version must be installed on all PCs.

It is possible to disable the automatic checking for software updates and to thus improve performance.

To disable the automatic checking for software updates, go to "Settings > General > Software updates" and clear the "Check daily for updates" check box.

Restriction in the online delta loading capability

When you replace a design, the online delta loading capability of the project is lost. TIA Portal does not output an error message in this case.

Screen objects in reports as PDF

The quality of the screen objects in the reports that you generate in Runtime as PDF files depends on the PDF printer drivers used.

Projects in Runtime after Windows 10 update

You have installed Windows 10 version 1709 on a PC with WinCC Professional V15 and a PC with WinCC Professional RT V15. You have performed a Windows 10 update to version 1803 on the Runtime PC. After the Windows 10 update, it may no longer be possible in Runtime to manually open a project loaded from the engineering system PC.

Remedy: Set the start type of the service for the WinCC SQL instance manually to "Automatic (Delayed Start)" and restart the system.

3.8 HMI devices

3.8.1 Notes on HMI devices

Contents

Information that could not be included in the online help and important information about product properties.

Unsupported HMI devices (as of TIA Portal V15)

All HMI devices with version 11.x are no longer supported as of TIA Portal V15.

The following HMI devices are no longer supported as of TIA Portal V15:

- WinAC MP 177, WinAC MP 277, WinAC MP 377
- Panels: OP and TP of the 70 series, 170 series and 270 series
- Multi Panels: Multi Panels of the 170 series, 270 series and 370 series

In order to use other devices in TIA Portal V15, for example Basic, Comfort or Mobile Panels, that have been configured with version 11.x, change the device version of the device in question.

The storage structure in the installation directory also no longer contains data for 11.0. This helps to reduce the size of the TIA Portal installation package compared to the previous versions.

Unsupported HMI devices (as of TIA Portal V15.1)

In addition to the devices that are not supported in TIA V15, the following devices are no longer supported as of TIA Portal V15.1:

- WinCC Runtime Advanced V11

Multi-key operation

Unintentional actions can be triggered by multi-key operation:

- When you are using a key device, you cannot press more than two function keys at the same time.
- When you are using a touch device, a standard PC or a panel PC, you can only press one function key or button at the same time.

HMI devices with high communication load

S7 Diagnostics should be enabled if a Panel is assigned many connections to PLCs or other HMI devices. Otherwise, you will risk overload on the Panel.

TS Adapter with Ethernet interface

If an HMI device is connected via Ethernet and a TS adapter, it can not be reset to factory settings.

Simulation with real PLC connection

The access point used by the simulation is independent from the settings of the engineering system and can only be altered in the Control Panel with the "Setting PG/PC Interface" tool. If the PLC connection is terminated right after the start of the simulation with alarm 140001, you should check the access point used by the simulation with "Setting PG/PC Interface".

1. Double-click "Setting PG/PC Interface" in the Control Panel. A dialog opens.
2. Select "S7ONLINE" in the "Access point of application" field as standard for HMI.
3. Select the interface in the "Interface Parameter Assignment Used" area.
4. Exit the "Set PG/PC Interface" dialog with OK.

Upgrading Basic Panels to device version V12

Before you upgrade Basic Panels from device version V11 to device version V12, transfer the image of the V11 SP2 Update 5 or higher to the devices.
In the "SIMATIC ProSave [OS Update]" dialog, select the setting "Reset to factory settings".

In this way, you always start a functional update of the image.

Affected devices:

- KP300 Basic mono PN
- KP400 Basic color PN
- KTP400 Basic color PN

Simulation of the Basic Panels

Use an output field in an alarm text to output an external tag. The content of this output field will then always be displayed with "0" during simulation.

Basic Panels 2nd Generation

Basic Panels 2nd Generation support the Sm@rtServer option.

If you are not using a USB hub, select the USB port USB_X60.1 as storage path.

HMI devices with operating system Windows CE 5.0 or higher

Owing to a modified client-server communication security setting, the time difference between the HMI device (client) and PC (server) must not exceed 1 day. If you back up recipe data from the HMI device on a network drive, for example, make sure that the time is set correctly on the PC (server) and the HMI device (client).

Connection switch in the Control Panel with Basic Panels

If you use the "Override protected connection information" function, the following restriction applies:

You cannot perform a connection switch in the Control Panel of a Basic Panel from a PLC without a protection level to a PLC with a "Complete protection" level.

Mobile Panels 277F IWLAN

Mobile Panels 277F IWLAN are approved for use with CPUs of the type S7-1500F with WinCC V13 SP1 Update 4.

If this connection is not established automatically after the panel is switched on or after a fail-safe connection to a CPU S7-1500F is interrupted, a STOP/RUN transition needs to be performed on the CPU.

Affected MLFBs:

- 6AV6645-0EB01-0AX1
- 6AV6645-0EC01-0AX1
- 6AV6645-0EF01-0AX1
- 6AV6645-0GB01-0AX1
- 6AV6645-0GC01-0AX1
- 6AV6645-0GF01-0AX1
- 6AV6645-0EF02-0AX1
- 6AV6645-0EF01-0AX1

Using Mobile Panel 277F IWLAN in the industrial environment

To avoid communication problems between the HMI device and access point, observe the following:

IWLAN planning: Wireless connection between the HMI device and access point

- Wireless field planning needs to be carried out. Radar influences need to be taken into consideration for channels requiring Dynamic Frequency Selection (DFS).
- Avoid radio frequency interference from other devices, such as hand-held devices with Bluetooth or WLAN functionality.
- For adjacent access points, use data channels that are separated by at least one wireless channel (e.g. 36 and 44 rather than 36 and 40).
- Do not place several access points in the immediately vicinity of one another. We recommend a minimum distance of 5 m.
- For the Mobile Panel 277F IWLAN, use the "Software retry number" 15 instead of the default value 3.

Using the "Rapid roaming" function with iPCF-MC

- Always use iPCF or iPCF-MC (rapid roaming) for PROFIsafe communication via PROFINET.
- Use the same SSID for all access points.
- The "Background scan" should be switched off on the HMI device ("Background scan interval" for the HMI device = 1 (default value)).
- Use the same "Management channel" and different, non-overlapping "Data channels" for all access points.
- Configure the "PN-IO cycle time" higher than the "iPCF scan cycle time".
- Observe the configured timeout when assigning PROFIsafe parameters.

"Zone ID/Connection point ID" tag of a Mobile Panel 277 IWLAN V2

The tag used for the "Zone ID/Connection point ID" must be of data type INT for Mobile Panel 277 IWLAN V2 devices. Adapt this data type if necessary when migrating a project.

Mobile Panels 277F IWLAN (RFID tag): F_DB_STATES

The F_DB_STATES block is no longer supplied with WinCC V14. The block F_DB_STATES serves only for data exchange. You can configure an F_DB to replicate the function of the F_DB_STATES. Note the description of the F_DB_STATES in the operating instructions of your HMI device or in the information system.

IPC427D

If an IPC427D is configured in a V12 or V13 project in which IE1 of the PC station is assigned and the device version of the PC station changes from V1 to V2 after upgrading to V14, the interface assignment of CP_IE_1 must initially be set to "None, or different Windows settings" and then changed to "SIMATIC PC Station" before compiling the project . Then, the project can be compiled without errors.

Printing alarms (Mobile Panels 2nd Generation)

It is not possible to print alarms as PDF or HTML when the internal memory "/flash" is selected as "Storage Location". To print alarms as PDF or HTML, select an SD memory card or a USB stick as storage location.

DHCP on KTP 400 PN, KTP 700 PN and KTP 900 PN

Operation with DHCP is not recommended for the KTP 400 PN, KTP 700 PN and KTP 900 PN devices.

In some circumstances, it is possible that the device will no longer boot. If that is the case, detach the network cable and reboot the device. After it has successfully started up, please assign a fixed IP address.

2nd Generation Mobile Panels

Mobile Panels 2nd Generation can be configured with TIA Portal V15.1 in device version V15. The compiled project cannot be downloaded to these devices, however.

An upgrade to device version V15.1 is required to work with Mobile Panels 2nd Generation in TIA Portal V15.1.

Sm@rtServer with Mobile Panels 2nd Generation

As of WinCC V15.1 the Sm@rtServer option is available and configurable for all 2nd generation Mobile Panels without a license. See the important notes in the Siemens Industry Online Support entry ID 109758056 (https://support.industry.siemens.com/cs/ww/en/view/109758056).

Sm@rtServer with KTP Mobile Panels

The evaluation of the safety operation via PROFIsafe is only guaranteed if the Sm@rtServer is disabled on the panel.

The use of Sm@rtServer on devices without a safety function is not recommended.

Simulation of Basic Panels 2nd Generation

3D hardware acceleration of the graphics card should be enabled for simulating the Basic Panels 2nd Generation.

Installing fonts

If you have installed new fonts on a panel with Prosave, you will need to restart the panel. The duration of the runtime startup process depends on the number and size of fonts.

Interruption of the DH485 connection

If you connect the Ethernet cable to the "PN_X1" connector of the HMI panel with Comfort Panels, the DH485 connection may be interrupted.

Remedy for Comfort Redesign/PRO Panels: When running the DH485 project on the HMI Panel, use the connector "PN_X3(Gigabit)" for Ethernet communication.

Sm@rtServer with Fail-safe Mobile Panels KTP400F/700F/900F Mobile

For correct evaluation of safety-related operation, ensure that Sm@rtServer is disabled on the HMI device.

Configure secure e-mail notification from Runtime

E-mails encrypted with TLS1.2 can be sent as of TIA Portal V14 on Comfort Panels and KTP Mobile Panels.

3.9 Add-ons

3.9.1 Sm@rtServer

Content

Information that could not be included in the online help and important information about product features.

File size for file broswer

The maximum file size for upload or download is 2 GB.

Opening SmartClient *.sac files under Windows 10

You save the data of the current SmartServer connection using the executable file SmartClient.exe with the *.sac format. To open the files generated in this way automatically under Windows 10, a *.sac file with SmartClient.exe must be opened once manually via the selection button and dialog in SmartClient.exe.

3.9.2 Audit

Contents

Information that could not be included in the online help and important information about product properties.

WinCC Audit Viewer 7.2 Update

An update for manual installation is available for WinCC Audit Viewer 7.2, which can be installed optionally.

Install the update for WinCC Audit Viewer 7.2 by running the "AuditViewer_V72_00_Upd1.exe" file from the Support directory on the product DVD.

3.9.3 DataMonitor

Contents

Information that could no longer be included in the online help and important information about product features.

Removal of WinCC Runtime

If you also want to use DataMonitor after removing WinCC Runtime Professional, you have to perform a repair of the installation as follows:

1. Open the Control Panel.
2. Double-click on "Add/Remove Programs".
3. Select "SIMATIC WinCC/DataMonitor client" in the list of programs installed
4. Click on the "Change/Remove" button.
5. Select the setting "Repair" in the WinCC DataMonitor setup.

Name of the user groups

The name of the user groups differs at certain locations of the Online Help from the actually created user groups. The following assignment is valid:

SIMATIC Report Administrators corresponds to DM_Admin

SIMATIC Report Users corresponds to DM_User

Excel-Workbook on a PC with Windows 7

If you operate Excel-Workbook on a PC with Windows 7, you need to disable the "Aero Glass" display mode.

Importing configuration data

If you load a project to a device via an external data medium, DataMonitor cannot export a correct XML file on this device.

The exported XML file can then no longer be used for importing configuration data.

Channel diagnostics view

The "Channel diagnostics view" object is not supported by the option WinCC DataMonitor.

3.9.4 WebNavigator

Contents

Information that could not be included in the online help and important information about product features.

Removal of WinCC Runtime

If you also want to use WebNavigator after removing WinCC Runtime Professional, you have to perform a repair of the installation as follows:

1. Open the Control Panel.
2. Double-click on "Add/Remove Programs".
3. Select "SIMATIC WinCC/WebNavigator client" in the list of programs installed
4. Click on the "Change/Remove" button.
5. Select the setting "Repair" in the WinCC/WebNavigator setup".

WebNavigator Client Manual logoff

Users can now also log off manually from the WebNavigator client using the ODK function "PWRTLogout()". You can find one description of the function in the Runtime API documentation under "Functions of the user administration > Functions for logon, logoff".

Virus scanner Trend Micro OfficeScan

It is not recommended to use the virus scanner of Trend Micro OfficeScan in combination with WebNavigator, as this virus scanner may have an adverse impact on screen loading times.

ProDiag overview, GRAPH overview and PLC code view

The WinCC option Web Navigator does not support the use of the WinCC objects "GRAPH overview", "ProDiag overview" and "PLC code view".

Time zone of Web parts

When you create and configure a Web part in a specific time zone in the Web center, the time zone is stored in the Web center.

If the time zone is changed in the operating system, this can lead to an incorrect display of the time in the Web part.

Characters not permitted in WinCC WebNavigator

Use the characters of the ASCII character set in the picture names and project names. Do not use any special characters or national special characters.

Channel diagnostics

The object "Channel diagnostics" is not supported by the option WinCC WebNavigator.

Automatic trigger for SmartTags in scripts

Use the automatic trigger if you use SmartTags in scripts in the WebNavigator screens.

Delay when outputting the WebNavigator screens

Due to the Web server technology, it may occur that updated WebNavigator screens are output with a delay in Runtime.

New function for WebNavigator client

As of TIA Portal Version V15.1, you have the option to define a start screen and a start language for WebNavigator client in the ActiveX control UserAdminControl.

3.9.5 WebUX

Content

Information that could not be included in the online help and important information about product features.

Update cycles

When accessing via WebUX, the refresh of the dynamics that use the screen cycle is always 2 seconds, even if a different screen cycle is configured.

Automatic login

If automatic login is no longer desired, you must delete the browser history in the respective browser.

New function for WebUX client

As of TIA Portal Version V15.1, you have the option to define a start screen and a start language for WebUX client in the ActiveX control UserAdminControl.

3.9.6 Redundancy

Contents

Information that could not be included in the online help and important information about product features.

Name in configuration of a redundant system

Only the computer name of the HMI devices is relevant for the configuration of a redundant system. You set the computer name of the HMI device in its operating system. The name of the HMI device from the project tree is only used for identification of the HMI device in the project.

3.9.7 SIMATIC Information Server

Contents

Information that could not be included in the online help and important information about product features.

Installation of SIMATIC Information Server and WinCC Runtime Professional on one PC

To set up the SIMATIC Information Server and WinCC Runtime Professional on one PC, adhere to the following installation sequence:

1. Install WinCC Runtime Professional.
2. Install the Information Server.

If you do not follow the installation sequence, connection problems may occur.

Index

.

.Net service packs, 5
.Net versions, 5

A

Asian, 5

C

Character string, 54
Characters, 5
Comparing library elements, 8
Compatibility, 20, 53
Conversions, 57
Convert character string, 54
CP 343-2, 26
CP 5512, 10

D

Deactivate, 57
Decimal places, 6
DEMUX, 54

E

EC31-RTX, 26
ENO, 57
Ethernet module
    Removal/insertion, 19

F

Firmware loader - Chinese GUI, 30
Force job on SD card, 9
Force table
    Changed display formats, 41
Function value, 55

G

Go online
    Multiple TIA Portal instances, 19

H

Hardware
    Detection, 29

I

I/O, 40
IP address, 10
ISO transport, 30
ISO-on-TCP, 30

K

K parameter, 54

L

Library, 8
Loading
    from a device, 20, 37
    Module comments, 26
    to the device, 20

M

Memory card, 9
    Removal/insertion, 19
Migration of IP access protection lists when activating Security, 32
Migration of projects with Ethernet CPs and Security, 32
Module
    Inserting, 19
    Removing, 19
Module comments, 26
MUX, 54

N

Network, 36
Network drive, 6

O

Online connection
    Multiple TIA Portal instances, 19
Online operation, 19
Open force job, 9

P

PLC migration
    Instruction profile, 19
PLCSIM, 37
Private Cloud, 11
Program_Alarm, 55

R

Return, 55

S

S_CONV, 54
S7 routing, 30
S7-1200, 25
S7-1200 modules, 25
S7-Port Configuration Tool, 21
SD card, 9
Security, 36
Security online diagnostics of S7 CPs, 32
SET, 54
SIMATIC S7 Embedded Controller, 26
Special diagnostics, 29
Special diagnostics - Chinese GUI, 30
STRING, 57

T

TiaDefaultProjectPath, 11
TiaUserSettingsPath, 11
Trace
    Deleting measurements, 29
    Deleting measurements in the device, 29
    Deleting measurements on the memory card, 29
TSAP length, 30

U

USB card readers, 9

V

Virtual environment, 11
VMware - PC CPs, 31

W

Watch table
    Changed display formats, 40
    Loading data blocks during an active control job, 40
    Multiple access to the same CPU, 40

SIEMENS

Installation

Programming and Operating Manual

Notes on the installation 1

Parallel installation 2

Licensing 3

System requirements for installation 4

Installation log 5

Starting installation 6

Displaying the installed software 7

Modifying or updating installed products 8

Repairing installed products 9

Starting to uninstall 10

Installing updates and support packages 11

Installing support packages automatically 12

Installing and uninstalling the migration tool 13

Installing and uninstalling the Multiuser server 14

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Notes on the installation....5
2 Parallel installation....7
3 Licensing....9
3.1 Notes on licenses....9
3.2 Licensing STEP 7 and WinCC....11
3.3 Handling licenses and license keys....17
3.4 Licensing of HMI devices....22
3.5 Licensing WinCC Runtime on PC-based HMI devices....23
3.6 Licensing WinCC Runtime Professional Asian....24
3.7 STEP 7 without license for PC-CP....25
4 System requirements for installation....27
4.1 Notes on the system requirements....27
4.2 General software and hardware requirements....29
4.3 Product-specific special characteristics....33
4.3.1 Installation of WinCC Runtime Professional....33
4.3.2 Removing an SQL instance....33
4.3.3 Notes on installing TIA Portal Teamcenter Gateway....36
5 Installation log....39
6 Starting installation....41
7 Displaying the installed software....45
8 Modifying or updating installed products....47
9 Repairing installed products....49
10 Starting to uninstall....51
11 Installing updates and support packages....53
11.1 Checking availability of updates and support packages and installing them....53
11.2 Working with a company-internal server....57
11.2.1 Properties and advantage of a corporate server....57
11.2.2 Configuring a company-internal server for updates....58
11.2.3 Providing updates on a company-internal server....61
12 Installing support packages automatically....65
12.1 Installing support packages automatically....65
12.2 Return values from the installation process....67
12.3 Log file....68

13 Installing and uninstalling the migration tool....69
13.1 System requirements....69
13.2 Installing the migration tool....70
13.3 Uninstalling the migration tool....71
14 Installing and uninstalling the Multiuser server....73
14.1 Notes on the installation of the multiuser server....73
14.2 Installing the multiuser server....75
14.3 Uninstalling the multiuser server....77
Index....79

Notes on the installation

1

Contents

Information that could not be included in the online help and important information about product characteristics.

Installation in a virtual environment (private cloud)

You can find instructions on how to install the TIA Portal in a virtual environment (private cloud) on the installation data medium in the directory "Documents\Readme\<language directory>". You can open the PDF document "TIAPortalCloudConnectorHowTo<language ID>.pdf" here.

Automated Installation

A description of automated installation is available on the product DVD in the directory "Documents\Readme\<Language directory>".

Use of the same versions of TIA Portal products during installation

When installing different TIA Portal products, make sure that you use the same versions of service packs and updates for the installation. For example, if you have installed SP1 for STEP 7 V13, you must also install SP1 for WinCC V13. The service packs and updates must be installed for all products at the same time. Do not start the TIA Portal until all products have been upgraded.

You can download the service packs from the Internet under Siemens Industry online support (https://support.industry.siemens.com/cs/de/en/).

Target directory of the installation

Do not use any UNICODE characters (for example, Chinese characters) in the installation path.

The installation path cannot be changed after installing one of the TIA Portal products.

Security settings for the installation of WinCC

When you install WinCC, security settings are changed in your operating system. These security settings are listed during the installation.

You must confirm the changes to the security settings.

If you make changes to your operating system after installing it, the security settings may be changed by installing TIA Portal.

You can restore the security settings by installing TIA Portal as follows:
"Start > All Programs > Siemens Automation > Security Controller > Restore settings".

Use of antivirus programs

During the installation, read and write access to already installed files is necessary. Some antivirus programs block this access. We therefore recommend that you disable antivirus programs during installation of the TIA Portal and enable them again afterwards.

Compatibility with V12 SP1 and V13

Empty projects from previous versions are installed in the installation directory under .. \<INSTALLDIR>\Sample Projects to allow the current TIA Portal to be opened in compatibility mode. These projects must be copied to a local directory with full access before they can be used. For more information on this, refer to FAQ ID 82169157.

FAQs on the TIA Portal

FAQs on the TIA Portal are available under FAQs (http://support.automation.siemens.com/WW/view/en/28919804/133000).

Parallel installation

2

Parallel installation of WinCC V15.1 and other SIMATIC HMI products

Parallel installation of WinCC V15.1 and versions of WinCC flexible earlier than WinCC flexible 2008 SP1 is not allowed.

Parallel installation of WinCC V15.1 with versions of WinCC earlier than WinCC V7.0 SP2 is not allowed. Parallel installation of WinCC V15 with WinCC V7.0 SP2 or WinCC V7.0 SP3 is not allowed for:

- WinCC V15.1 Professional
- WinCC V15.1 Runtime Professional

Parallel use

If the term "Combo" appears in the name or license key of the software after installation, the use of the following products/versions is permitted in accordance with paragraph 4.3 of the General Terms and Conditions (see also setup text):

- With the license "STEP 7 Professional Combo": older versions of STEP 7 (STEP 7 Professional 2006 - STEP 7 Professional V15)
- With the license "WinCC V15.1 Comfort Combo": WinCC flexible 2008 Standard
- With the license "WinCC V15.1 Advanced Combo": WinCC flexible 2008 Advanced

Licensing 3

3.1 Notes on licenses

Availability of licenses

The licenses for the products of the TIA Portal are usually supplied on an installation data medium and transferred automatically by the Automation License Manager during the installation process of the TIA Portal.

Before you uninstall the TIA Portal, you must transfer and back up the licenses still required. Use the Automation License Manager for this purpose.

Provision of the Automation License Manager

The Automation License Manager is supplied on the installation data medium and is transferred automatically during the installation process.

If you uninstall the TIA Portal, the Automation License Manager remains installed on your system.

Working with the Automation License Manager

The Automation License Manager is a product of Siemens AG, which is used for handling license keys (technical representatives of licenses).

Software products that require license keys for operation, such as the TIA Portal, register the need for license keys automatically with the Automation License Manager . If the Automation License Manager finds a valid license key for this software, the software can be used according to the license usage terms associated with this license key.

Note

For additional information on how to manage your licenses with the Automation License Manager , refer to the documentation supplied with the Automation License Manager .

See also

Notes on the system requirements (Page 27)

Starting installation (Page 41)

Displaying the installed software (Page 45)

Modifying or updating installed products (Page 47)

Repairing installed products (Page 49)

Starting to uninstall (Page 51)
Installation log (Page 39)

3.2 Licensing STEP 7 and WinCC

Introduction

You require a License Key to license the following STEP 7 editions:

- STEP 7 Basic
- STEP 7 Professional

You can install the corresponding License Key for STEP 7 using the Automation License Manager after the installation has been completed.

You need a License Key in WinCC for the following:

- WinCC Engineering System
- WinCC Runtime
- Options for WinCC Engineering System

You transfer licenses for WinCC and the WinCC add-ons after installation with the Automation License Manager.

Note

The licensee also recognizes that the software (SW) from the Microsoft Corporation or subsidiaries contains licensed software. The licensee hereby agrees to be bound by and comply with the terms of the attached license agreement between Microsoft SQL Server and End User.

Licenses

The following licenses with the corresponding License Keys are available:

- STEP 7 Basic
- STEP 7 Professional
- STEP 7 Professional Combo
- WinCC Basic
- WinCC Comfort
- WinCC Comfort Combo
- WinCC Advanced
- WinCC Advanced Combo
- WinCC Professional
- WinCC Professional Combo
- WinCC RT Advanced
- WinCC RT Professional

Validity of license keys for older versions of STEP 7

With a valid License Key for V15.x of STEP 7 Professional and STEP 7 Professional Combo, you can also operate older versions of STEP 7 without restrictions. The following table provides more detailed information about this:

Edition	License	Valid for
STEP 7 Basic V15.x	STEP 7 Basic	• STEP 7 Basic V15.x • STEP 7 Basic V14.x • STEP 7 Basic V13.x • STEP 7 Basic V12.x • STEP 7 Basic V11.x • STEP 7 Basic V10.5
STEP 7 Professional V15.x	STEP 7 Professional	• STEP 7 Professional V15.x • STEP 7 Professional V14.x • STEP 7 Professional V13.x • STEP 7 Professional V12.x • STEP 7 Professional V11.x • STEP 7 Basic V14.x • STEP 7 Basic V13.x • STEP 7 Basic V12.x • STEP 7 Basic V11.x • STEP 7 Basic V10.5
STEP 7 Professional V15.x	STEP 7 Professional Combo	• STEP 7 Professional V15.x • STEP 7 Professional V14.x • STEP 7 Professional V13.x • STEP 7 Professional V12.x • STEP 7 Professional V11.x • STEP 7 Basic V14.x • STEP 7 Basic V13.x • STEP 7 Basic V12.x • STEP 7 Basic V11.x • STEP 7 Basic V10.5 • STEP 7 V5.6 • STEP 7 V5.5 • STEP 7 V5.4 • STEP 7 Professional 2017 • STEP 7 Professional 2010 • STEP 7 Professional 2006

Validity of license keys for older versions of WinCC

With a valid License Key for WinCC Professional V15.x, you can also operate older versions of WinCC without restrictions.

You can find more detailed information in the following table:

WinCC Engineering System

Edition	License	Valid for
WinCC Basic V15.x	WinCC Basic	• WinCC Basic V11.x • WinCC Basic V12.x • WinCC Basic V13.x • WinCC Basic V14.x • WinCC Basic V15.x
WinCC Comfort/Advanced V15.x	WinCC Comfort	• WInCC Basic V11.x • WinCC Comfort V10.x • WinCC Comfort V11.x • WinCC Basic, Comfort V12.x • WinCC Basic, Comfort V13.x • WinCC Basic, Comfort V14.x • WinCC Basic, Comfort V15.x
	WinCC Comfort Combo	• WinCC flexible Compact, Standard >= 2005 • WInCC Basic V11.x • WinCC Comfort V10.x • WinCC Comfort V11.x • WinCC Basic, Comfort V12.x • WinCC Basic, Comfort V13.x • WinCC Basic, Comfort V14.x • WinCC Basic, Comfort V15.x
	WinCC Advanced	• WInCC Basic V11.x • WinCC Comfort, Advanced V10.x • WinCC Comfort, Advanced V11.x • WinCC Basic, Comfort, Advanced V12.x • WinCC Basic, Comfort, Advanced V13.x • WinCC Basic, Comfort, Advanced V14.x • WinCC Basic, Comfort, Advanced V15.x
	WinCC Advanced Combo	• WinCC flexible Compact, Standard, Advanced >= 2005 • WInCC Basic V11.x • WinCC Comfort, Advanced V10.x • WinCC Comfort, Advanced V11.x • WinCC Basic, Comfort, Advanced V12.x • WinCC Basic, Comfort, Advanced V13.x • WinCC Basic, Comfort, Advanced V14.x • WinCC Basic, Comfort, Advanced V15.x

Edition	License	Valid for
WinCC Professional V15.x	WinCC Professional (512) WinCC Professional (4096) WinCC Professional (max.)	• WinCC Comfort, Advanced, Professional V10.x • WinCC Comfort, Advanced, Professional V11.x • WinCC Basic, Comfort, Advanced, Professional V12.x • WinCC Basic, Comfort, Advanced, Professional V13.x • WinCC Basic, Comfort, Advanced, Professional V14.x • WinCC Basic, Comfort, Advanced, Professional V15.x
	WinCC Professional (512) Combo WinCC Professional (4096) Combo WinCC Professional (max.) Combo	• WinCC flexible Compact, Standard, Advanced >= 2008 SP3 • WinCC SCADA V7.0 SP3 ES • WinCC Comfort, Advanced, Professional V10.x • WinCC Comfort, Advanced, Professional V11.x • WinCC Basic, Comfort, Advanced, Professional V12.x • WinCC Basic, Comfort, Advanced, Professional V13.x • WinCC Basic, Comfort, Advanced, Professional V14.x • WinCC Basic, Comfort, Advanced, Professional V15.x

WinCC Runtime Advanced and Runtime Professional

Edition	License	Valid for
WinCC RT Advanced V15.x	WinCC RT Advanced (128) WinCC RT Advanced (512) WinCC RT Advanced (2048) WinCC RT Advanced (4096) WinCC RT Advanced (8192) WinCC RT Advanced (16384)	• WinCC RT Advanced V11.x • WinCC RT Advanced V12.x • WinCC RT Advanced V13.x • WinCC RT Advanced V14.x • WinCC RT Advanced V15.x
WinCC RT Professional V15.x	WinCC RT Professional (128) WinCC RT Professional (512) WinCC RT Professional (2048) WinCC RT Professional (4096) WinCC RT Professional (8192) WinCC RT Professional (65536) WinCC RT Professional (102400) WinCC RT Professional (153600) WinCC RT Professional (262144)	• WinCC RT Advanced V11.x • WinCC RT Advanced V12.x • WinCC RT Advanced V13.x • WinCC RT Advanced V14.x • WinCC RT Professional V11.x • WinCC RT Professional V12.x • WinCC RT Professional V13.x • WinCC RT Professional V14.x • WinCC RT Professional V15.x

Validity of licenses for WinCC options

WinCC options generally have version-independent License Keys. You can use WinCC options V15.x with V11, V12, V13 and V14 License Keys and vice versa.

Starting without a valid license key

If you start the TIA Portal without a valid License Key , the system alerts you that you are working in non-licensed mode. You have the one-time option of activating a Trial License. However, this license is valid for a limited period only and expires after 21 days.

When the Trial License expires, the following scenarios can occur:

- TIA Portal has never been licensed on the PC in question:
  - No license-based actions can be performed in the TIA Portal.
- TIA Portal was already licensed on the PC in question:
  - A window requiring acknowledgment presents an alert for non-licensed mode every 10 minutes and for every action requiring a license.

License requirements for simulation

You do not require additional licenses when you use the menu command "Online > Simulation" to start the simulation in STEP 7.

If the following conditions are met, the appropriate licenses for the edition of STEP 7 you have installed are also required for the simulation.

- The engineering station is connected to a PLC.
- The connection to the PLC is configured and active.

If you start simulation in WinCC using the menu command "Online > Simulation > With tag simulator", you do not need licenses for WinCC Runtime or for licensed options.

If the following conditions are fulfilled, you also need the appropriate licenses for WinCC Runtime and for licensed options for the simulation:

- The engineering station is connected to a PLC.
- The connection to the PLC is configured and active.
- Start the simulator with the menu command "Online > Simulation".

Licenses for the TIA Portal Cloud Connector

For working with the TIA Portal Cloud Connector, you need a valid License Key for each device that you specify as "User device" in the TIA Portal Cloud Connector. The License Key is also required if the TIA Portal is installed on this device. A License Key is not required for devices that you are using as "remote device".

See also

Handling licenses and license keys (Page 17)

3.3 Handling licenses and license keys

Introduction

You need a valid a License Key in each case to use STEP 7 Basic, STEP 7 Professional as well as WinCC Engineering System, options for WinCC Engineering System and WinCC Runtime.

NOTICE
Destruction of license keys by copying A License Key cannot be copied. The copy protection prevents the License Keys from being copied. If you attempt to copy a License Key, it will be destroyed.

Trial License

If you start the TIA Portal without a valid License Key, the system alerts you that you are working in non-licensed mode. You can activate a trial license once. The trial license expires after 21 days.

If the trial license has expired, the following situations may occur:

- TIA Portal has never been licensed on the PC in question:
  The TIA Portal cannot be started.
- TIA Portal was already licensed on the PC in question:
  TIA Portal can be started. Non-licensed mode is indicated every 10 minutes by a window, which you must acknowledge.

Installing license keys for STEP 7

When you install STEP 7 Basic, the License Key is installed automatically during setup. When you install STEP 7 Professional, you are prompted at the end of the setup to transfer the license from the supplied storage medium to your PC using the Automation License Manager.

If you want to install additional License Keys, you have to use the Automation License Manager to do this.

When you transfer a license, the associated License Key is removed from the License Keys storage location.

Note

Following installation, always keep the original storage location of the License Keys.

Installing license keys for WinCC

When you install WinCC Basic, the License Key is installed automatically during setup. When you install WinCC Professional, you are prompted at the end of the setup to transfer the license from the supplied storage medium to your PC using the Automation License Manager.

You need additional License Keys to use WinCC Runtime or simulation on the Engineering-PC with the menu command "Online > Simulation > Start". You need to use Automation License Manager for this.

When you transfer a license, the associated License Key is removed from the License Keys storage location.

Note

Following installation, always keep the original storage location of the License Keys.

Transferring license keys to the HMI device

You must transfer the License Keys to the HMI device to operate WinCC.

You transfer a License Key to the HMI device in the following cases:

- To operate WinCC Runtime
- To use add-ons for WinCC Runtime

When you transfer a license to an HMI device, the associated License Key is removed from the License Keys storage location.

If you no longer need the license or want to back up data, you must transfer the License Keys from the operator panel. You can then use this license on another PC or HMI device.

Uninstalling license keys

License Keys are always uninstalled using the Automation License Manager. You uninstall a License Key in the following cases:

- When backing up data.
- If you no longer require the license.

You can also use a valid license on another PC.

Modification to the data media of the engineering system

Modifying the data medium of the engineering system can destroy the License Key. To do this before such modifications, open the Automation License Manager and back up License Key to be uninstalled to another storage location.

NOTICE
Destruction of license keys on PCs First, uninstall all License Keys if you do one of the following: • Format the hard disk • Compress the hard disk • Restore the hard disk • Start an optimization program that moves fixed blocks • Install a new operating system Read the description of Automation License Manager ("Start > Siemens Automation > Documentation"). Observe all warnings and notices.

Backup of backup devices (Backup/Restore)

When backing up data on the HMI device, remove the License Keys on the HMI device. To do this, open the Automation License Manager and back up the uninstalled License Key to another storage location.

NOTICE
Destruction of license keys on non-PC-based HMI devices License keys transferred as a result of backup/restore operations are destroyed in the case of the following HMI devices: • 270 series • 370 series Carry out the following before beginning a restore operation: • Use the Automation License Manager and ProSave to check if there are License Keys on the HMI device. • Transfer the existing License Keys from the HMI device. After restoring has been carried out, transfer the License Keys back to the HMI device.

Invalid license after time zone change

The installed licenses will no longer work if you change the time zone on a PC as follows: From a full-hour time zone to a time zone that is not based on a full hour.

To avoid this inconvenience, uninstall the License Key with the Automation License Manager under the time zone setting that was set when the License Keys was installed.

Example: You want to use the HMI device from the time zone "GMT +3:00" in the time zone "GMT +3:30".

1. Uninstall the License Key with the time zone setting "GMT +3:00".
2. Change the time zone settings accordingly on your PC.
3. Install the License Key.

This behavior does not apply to the trial license.

Defective license

A license is defective in the following cases:

- If the License Key is no longer accessible at the storage area.
- If the License Key disappears during its transfer to the destination drive.

Note

If you reset the system date to an earlier time, all licenses are invalidated.

You can use the Automation License Manager to repair the defective license. Use the "Restore" function or the "Restore wizard" of the Automation License Manager for this purpose. Contact Customer Support to restore them. You can find more detailed information on the Internet: http://support.automation.siemens.com (https://support.industry.siemens.com/cs/de/en/)

Note

Runtime can also be operated without errors if the license is missing or defective. The system alerts you at brief intervals that you are working in non-licensed mode.

Microsoft SQL Server 2014

A license is required for use of the Microsoft SQL Server database. The license is included in a properly licensed and installed version of WinCC.

The SQL Server which is licensed with the installation of WinCC must only be used in connection with WinCC.

Its use for other purposes requires an additional license. These include, for example:

- Use for custom databases
- Use for third-party applications
- Use of SQL access mechanisms which are not provided via WinCC.

Uninstalling

After uninstalling WinCC, you must uninstall the "WinCC" SQL server instance. To do this, under "Control Panel > Programs" select the entry "Microsott SQL Server 2014" and uninstall it.

See also

Licensing STEP 7 and WinCC (Page 11)

3.4 Licensing of HMI devices

Non-PC-based HMI devices are equipped with Runtime licenses. A License Key is not required for runtime operation.

A license may be required for an add-on for non-PC-based HMI devices. The License Key of the respective license always activates one add-on for use.

License key

In order to license non-PC-based HMI devices with License Keys, the "SIMATIC HMI License Manager Panel Plugin" add-on is needed.

WinCC Setup installs this add-on by default. You open the License Manager Panel Plugin in the Automation License Manager with the "Edit > Connect Target System > Connect HMI Device" menu command.

If WinCC is not installed, an installation of ProSave 7.2 or higher is required.

Notes on handling licenses

- Further information on license handling can be found in the Automation License Manager help.
- Verify that the current version of the operating system (or higher) is installed on the HMI device before you start licensing. If necessary, update the operating system using ProSave.
- The licensee also recognizes that the software (SW) from the Microsoft Corporation or subsidiaries contains licensed software. The licensee hereby agrees to be bound by and comply with the terms of the attached license agreement between Microsoft SQL Server and End User.

Non-licensed mode

Runtime add-ons can be used for a short time without restrictions without a valid license. Non-licensed mode is indicated every 10 minutes by a window, which you must acknowledge.

3.5 Licensing WinCC Runtime on PC-based HMI devices

License key

With PC-based HMI devices, you require a license key for the following:

- WinCC Runtime with 128 tags, for example.
- WinCC add-ons

You can install the license key for WinCC Runtime during the installation of WinCC. You transfer licenses for the WinCC add-ons after installation with the Automation License Manager.

Note

Only the Certificate of License for WinCC Runtime V15.1 authorizes operation of WinCC Runtime V15.1

The appropriate new license is required to license WinCC Runtime.

You can upgrade the runtime licenses of WinCC flexible 2008 and WinCC V7 with an upgrade to WinCC Runtime V15.1.

Productive operation of the software is only allowed with a valid Certificate of License that matches the version listed on it.

Notes on handling licenses

- When you upgrade an older version to version V15, you need new licenses for WinCC Runtime Professional.
- The licensee also recognizes that the software (SW) from the Microsoft Corporation or subsidiaries contains licensed software. The licensee hereby agrees to accept the terms of the accompanying license agreement between Microsoft SQL Server and end users and to abide by them.

Non-licensed mode

WinCC Runtime and the Runtime add-ons can be used freely without license. Non-licensed mode is indicated every 10 minutes by a window, which you must acknowledge.

3.6 Licensing WinCC Runtime Professional Asian

Introduction

If you are using WinCC Runtime Professional in an Asian language, a dongle is additionally used for the validation of the required license. The dongle comes in the form of a certified USB stick together with the required license key for WinCC Runtime Professional Asian. The license key is stored on the certified USB flash drive.

The following Asian languages are protected by a dongle:

- Chinese Simplified
- Chinese Traditional
- Japanese
- Korean

Using the dongle for an Asian Runtime Professional

In order to successfully license a WinCC project in an Asian language, transfer the required license key from the certified USB stick to the HMI device using the "Automation License Manager". Plug the USB into the USB port of the HMI device. The USB stick acts as a dongle during runtime. The dongle must always remain plugged in the USB port in during runtime.

If you remove the dongle during runtime, WinCC Runtime switch to Demo mode.

The USB sticks supplied for the Asian markets are certified for permanent use in production. Therefore, you cannot transfer or copy the content to another disk or USB medium. If you lose the certified USB stick or it becomes defective, you need a new, certified USB stick.

Simulation of a WinCC project in an Asian language

You then also need a certified USB stick as dongle for the simulation. If no dongle is available, the simulation runs in Demo mode.

3.7 STEP 7 without license for PC-CP

Functions of STEP 7 for PC-CP

STEP 7 for PC-CP is a full version of STEP 7 that was installed without license. This version only supports configuration and diagnostic options for the following components:

- PC-CPs and applications (OPC server) for PC stations
- SCALANCE X switches
- SCALANCE W / SCALANCE M routers
- SCALANCE S security module
- PROFINET IO devices and PROFIBUS DP slaves
- IE/PB LINK PN IO, IWLAN/PB LINK PN IO
- Readers (RFID) and optical readers (MV)

SIMATIC stations and WinAC components are only available in the network and topology view (no device view). Programming functions are not available.

Information system

The information system for PC-CP available in STEP 7 describes the full scope of STEP 7 functions. You should therefore remember that the functions are restricted to the configuration of the previously listed device groups.

Upgrading to the range of functions of STEP 7 Professional

The full scope of STEP 7 functions is already available on your engineering station with the installation of STEP 7 for PC-CP, although its use is restricted.

By installing a license later, you can upgrade at any time to the full functionality of STEP 7. There is no need for a new installation. Any projects created previously in STEP 7 for PC-CP can still be used without restrictions once a license is installed.

System requirements for installation 4

4.1 Notes on the system requirements

System requirements for individual products

The system requirements may differ depending on the products you want to install. You should therefore check the individual system requirements of your products.

If you want to install several products, make sure that your system meets the demands of the product with the highest requirements.

Displaying PDF files

To be able to read the supplied PDF files, you require a PDF reader that is compatible with PDF 1.7 e.g. Adobe (R) Reader version 9.

Displaying the Welcome Tour

You need the Adobe (R) Flash Player Version 9 or higher to start the Welcome Tour for the TIA Portal.

Windows 7 and Windows Server 2008 R2: Invalid certificate validation

The new certificates used for signing may possibly not be validated on the Windows 7 and Windows Server 2008 R2 operating systems.

If you do not have the possibility to update the certificate store online, the signature of the product files is displayed as invalid. In this case, update the root certificates provided by Microsoft on the offline systems. To do so, follow the instructions in FAQ 109748887 (https://support.industry.siemens.com/cs/ww/en/view/109748887).

Note for installation under Windows 7

The TIA Portal can only be run on Windows systems that support the secure hash algorithm "SHA-2". "SHA-2" is only supported under Windows 7 once a security update has been installed (KB 3033929). Therefore, make sure that the latest Windows security updates are installed on your system.

See also

Notes on licenses (Page 9)

Starting installation (Page 41)

Displaying the installed software (Page 45)

Modifying or updating installed products (Page 47)

Repairing installed products (Page 49)
Starting to uninstall (Page 51)

4.2 General software and hardware requirements

System requirements for installation

The following table shows the minimum software and hardware requirements that have to be met for the installation:

Hardware/software	Requirement
Processor	Intel® Core™ i3-6100U, 2.30 GHz
RAM	8 GB
Hard disk	S-ATA with at least 20 GB available space
Network	From 100 Mbit
Screen resolution	1024 x 768
Operating systems *	**Windows 7 (64-bit)** • Windows 7 Home Premium SP1 ** • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 10 (64-bit)** • Windows 10 Home Version 1709 ** • Windows 10 Home Version 1803 ** • Windows 10 Professional Version 1709 • Windows 10 Professional Version 1803 • Windows 10 Enterprise Version 1709 • Windows 10 Enterprise Version 1803 • Windows 10 Enterprise 2016 LTSB • Windows 10 IoT Enterprise 2015 LTSB • Windows 10 IoT Enterprise 2016 LTSB **Windows Server (64-bit)** • Windows Server 2012 R2 StdE (full installation) • Windows Server 2016 Standard (full installation)

* Including all security updates up to the Microsoft Security Bulletin Summary for September 2018 (MS18-Sep). For more detailed information on operating systems, refer to the help on Microsoft Windows or the Microsoft Web site.

** Only for Basic editions

Recommended PC hardware

The following table shows the recommended hardware requirements:

Hardware/software	Requirement
Computer	SIMATIC FIELD PG M5 Advanced or higher (or comparable PC)
Processor	Intel® Core™ i5-6440EQ (up to 3.4 GHz)
RAM	16 GB or more (32 GB for large projects)

Hardware/software	Requirement
Hard disk	SSD with at least 50 GB available memory space
Network	1 GBit (for multiuser)
Monitor	15.6" Full HD Display (1920 x 1080 or higher)

Supported virtualization platforms

You can install the software packages "SIMATIC STEP 7" and "SIMATIC WinCC" on a virtual machine. For this purpose, use one of the following virtualization platforms in the specified version or a newer version:

- VMware vSphere Hypervisor (ESXi) 6.5
- VMware Workstation 12.5.7
- VMware Player 12.5.7
- Microsoft Hyper-V Server 2016

The following operating systems are recommended as host operating systems for these virtualization platforms:

Operating system	VMware vSphere Hypervisor (ESXi)	VMware Workstation	VMware Player	Microsoft Hyper-V
Windows Server 2016 Standard (full installation, 64-bit)	-	-	-	x
Windows 7 Professional SP1	x	x	x	-
Windows 7 Enterprise SP1	x	x	x	-
Windows 7 Ultimate SP1	x	x	x	-
Windows 10 Professional Version 1703	x	x	x	-
Windows 10 Enterprise Version 1703	x	x	x	-
Windows 10 Enterprise 2016 LTSB	x	x	x	-

x: Can be used as host operating system

-: Cannot be used as the host operating system

The following guest operating systems are recommended for installation within the selected virtualization platform:

Operating system	VMware vSphere Hypervisor (ESXi)	VMware Workstation	VMware Player	Microsoft Hyper-V
Windows Server 2012 R2 StdE (full installation, 64-bit)	-	-	-	x
Windows Server 2016 Standard (full installation, 64-bit)	-	-	-	x
Windows 7 Professional SP1	x	x	x	x
Windows 7 Enterprise SP1	x	x	x	x
Windows 7 Ultimate SP1	x	x	x	x
Windows 10 Professional Version 1703	x	x	x	x

Operating system	VMware vSphere Hypervisor (ESXi)	VMware Workstation	VMware Player	Microsoft Hyper-V
Windows 10 Enterprise Version 1703	x	x	x	x
Windows 10 Enterprise 2016 LTSB	x	x	x	x

x: Can be used as guest operating system

-: Cannot be used as the guest operating system

Note

- The same hardware requirements apply to the host operating system as for the respective TIA products.
- The plant operator must ensure that sufficient system resources are available for the host operating systems.
- The hardware certified by the manufacturers is recommended for the use of HyperV server and ESXi.
- When you use Microsoft Hyper-V, accessible stations cannot be displayed.

Supported security programs

The following security programs are compatible with "SIMATIC STEP 7 Basic" and "SIMATIC WinCC":

- Antivirus programs:
  - Symantec Endpoint Protection 14
  - Trend Micro Office Scan Corporate Edition 12.0
  - McAfee VirusScan Enterprise 8.8
  - Kaspersky Anti-Virus 2017
  - Windows Defender
  - Qihoo 360 "Safe Guard 11.0" + "Virus Scanner 5.0"

  Note

  Make sure that your virus scanner and its databases are always up-to-date.

  The last update of the virus pattern was on 1 October 2018.
- Encryption software:
  - Microsoft Bitlocker
- Host-based Intrusion Detection System:
  - McAfee Application Control 8.0

See also

Notes on licenses (Page 9)

Starting installation (Page 41)

Checking availability of updates and support packages and installing them (Page 53)
Displaying the installed software (Page 45)
Modifying or updating installed products (Page 47)
Repairing installed products (Page 49)
Starting to uninstall (Page 51)

4.3 Product-specific special characteristics

4.3.1 Installation of WinCC Runtime Professional

Installation in domains and workgroups

WinCC is generally approved for operation in a domain or workgroup.

However, be aware that domain group policies and restrictions of the domain may hinder the installation. In this case, remove the computer from the domain before installing Microsoft Message Queuing, Microsoft SQL Server and WinCC. Log on to the local machine with administrator rights. Perform the installation. After successful installation, you can restore the WinCC computer to the domain. If the domain group policies and restrictions of the domain do not impede the installation, the computer need not be removed from the domain during the installation.

Be aware that domain group policies and restrictions of the domain may also hinder operation. If you cannot avoid these restrictions, run the WinCC computer in a workgroup.

Consult with the domain administrator if needed.

Windows computer name

Do not change the Windows computer name after installing WinCC.

If you change the Windows computer name, you must uninstall the SQL server and reinstall it.

The following characters are not permitted for the computer name:

- . , ; : ! ? " ' ^ ´ ` ~ _
- + = / \ ¦ @ * # $ % & § °
- ( ) [ ] { } < >
- Blanks

Follow these recommendations when assigning the Windows computer name:

- Only uppercase letters can be used.
- The first character must be a letter.
- The first 12 characters of the computer name must be unique.

4.3.2 Removing an SQL instance

Contents

A new WinCC SQL 2014 instance is installed with the installation of WinCC V15.1.

If you have already installed a version of WinCC Runtime Professional V13 or WinCC Runtime Professional V13 SP1, before you can install WinCC Professional V15.1 you must either

- Install WinCC Runtime Professional V15.1 first or
- Uninstall the "WINCC" instance of the SQL Server 2008 R2.

If you have installed a version older than V13 of WinCC Runtime Professional, you need to uninstall the WinCC instance of SQL Server 2008 R2 or the WinCC instance of SQL Server 2005.

Note

By removing WinCC Runtime Professional or WinCC Professional you do not remove the instance of the SQL server.

Uninstalling an SQL instance

To uninstall an older SQL instance of WinCC, follow these steps:

1. Open the Control Panel.
2. Click "Remove program".
3. Select "Microsoft SQL Server 2008 R2" in the list of programs installed and click "Remove/change".
4. In the "SQL Server 2008 R2" dialog, click "Remove".
5. In the "Setup Support Rules" dialog, click "OK".

6. In the "Select Instance" dialog, select the "WINCC" instance and click "Next".

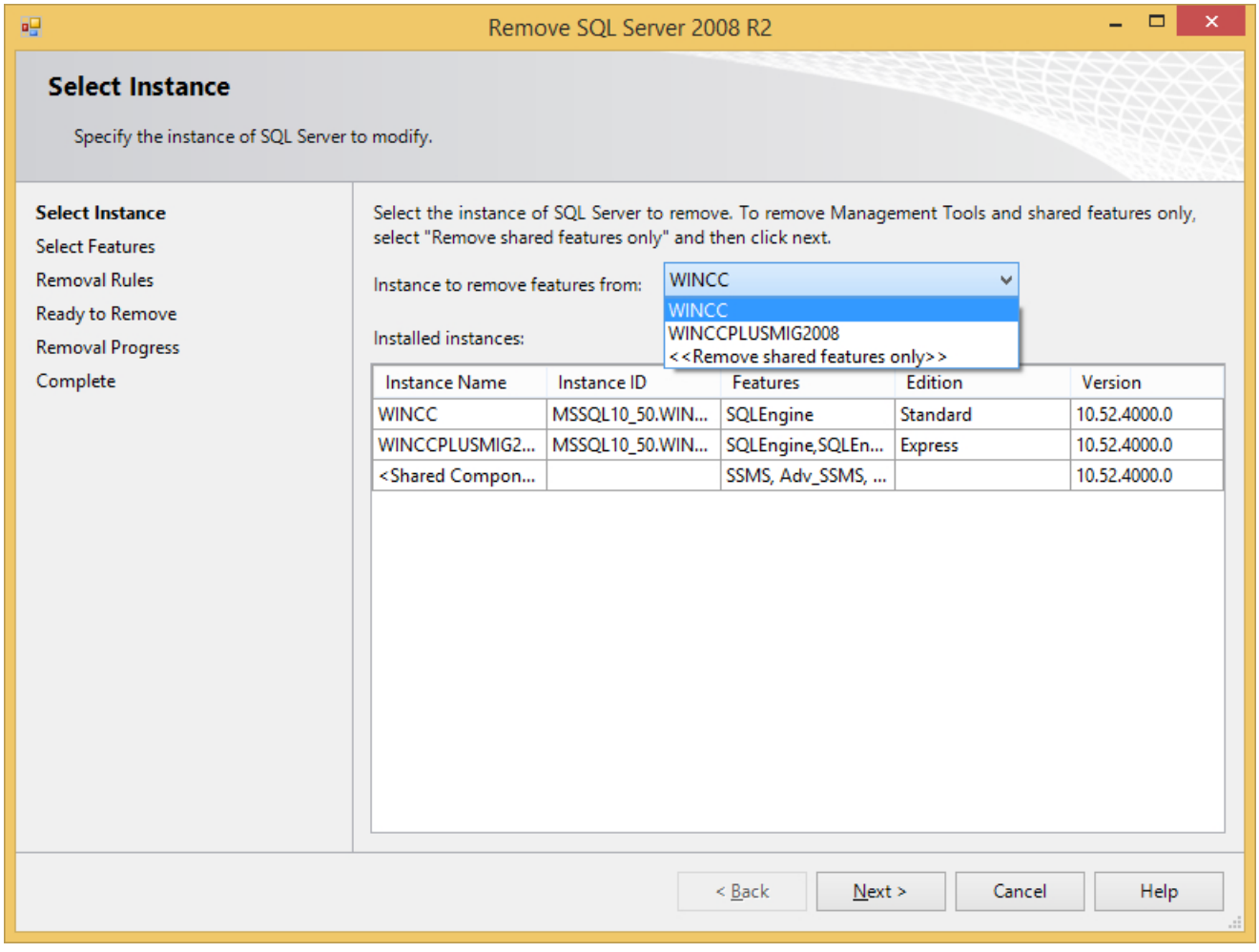

7. In the "Select Features" dialog, click "Select All" and then "Next".
8. In the "Removal Rules" dialog, click "Next".
9. In the "Ready to Remove" dialog, click "Remove".
10. After removal is complete, click "Close" in the "Complete" dialog.

The procedure for removal of an SQL Server 2005 instance is similar.

4.3.3 Notes on installing TIA Portal Teamcenter Gateway

Introduction

You can install the TIA Portal Teamcenter Gateway as follows:

- Together with the installation of the TIA Portal

Note

Administrator rights are required for the server installation.

Requirements for installation

Your computer must have:

- Teamcenter Rich Application Client (referred to as "RAC" below) or Teamcenter Client Communication System (TCCS) version 11.2 or higher installed.

Note

If TCCS was installed as "standalone", you must also install "Microsoft Visual C++ 2013 – Redistributable". This ensures that you can establish a connection between TIA Portal and TIA Portal Teamcenter Gateway.

- A connection from FCC/FMS (FileClientCache/FileManagementSystem) to "Teamcenter Server 11.2 or higher" must exist.

Note

- After a reinstallation or a restart, check whether a connection is available from FCC/FMS (FileClientCache/FileManagementSystem) to the "Teamcenter Server 11.2 or higher".
- After a reinstallation or a restart, check the registry key. You can find additional information on checking the registry key in the "TIA Portal Teamcenter Gateway" online help in the section Installing and uninstalling TIA Portal Teamcenter Gateway" > "Checking the **installation** of TIA Portal Teamcenter Gateway".
- After a re-installation or restart, check whether you can save an item type in Teamcenter and if you can add a dataset to the item revision. Additional information on saving element types is available in the "TIA Portal Teamcenter Gateway" online help in the chapter "Managing TIA Portal projects with TIA Portal Teamcenter Gateway" in the section "Saving a TIA Portal project as new item in Teamcenter".
- After a reinstallation or a restart, check that a data transfer from Teamcenter Rich Application Client to the Teamcenter server is possible.

- The supplied data model consisting of several files must be installed on the Teamcenter server.

Installation of the TIA Portal Teamcenter Gateway - data model

Installation of the supplied TIA Portal Teamcenter Gateway data model must be performed with the "Environment Manager".

You can find a description of the installation on the product DVD in the directory "\Support \Teamcenter_11\ServerSetupDocument\".

Identical TIA Portal versions during installation

When installing different TIA Portal products, make sure that you use the same versions of service packs and updates for the installation.

If you have installed TIA Portal V14 or higher, for example, you also have to install TIA Portal Teamcenter Gateway V14 or higher.

Installation path

Do not use any UNICODE characters (for example, Chinese characters) in the installation path.

Antivirus programs

During the installation, read and write access to installed files is necessary. Some antivirus programs block this access. We therefore recommend that you disable antivirus programs during the installation of the TIA Portal Teamcenter Gateway and enable them again afterwards.

Installation log

5

Function of the installation log

The progress during the following installation processes is logged in a file:

- Installing products
- Modifying or updating already installed products
- Repairing an existing installation
- Uninstalling products

If errors occur during the installation process or warnings are issued, these can be evaluated with the help of the log file. You can do this yourself or contact product support.

Installation logs storage location

The log file is the most recent file with the file extension ".log" and the name of which that starts with "SIA".

The location of the log file is stored in the environment variable "%autinstlog%". You can enter this environment variable in the address bar of Windows Explorer to open the folder with the log files. Alternatively, you can navigate to the corresponding directory by entering "CD %autinstlog%" in the command line.

The storage location is dependent on the operating system, e.g. "C:\ProgramData\Siemens \Automation\Logfiles\Setup" in English-language Windows.

Setup_Report (CAB file)

To make it easier to provide Product Support with all necessary files, an archive file that contains the installation log and all other required files is saved in CAB format. This log can be found at "%autinstlog%\Reports\Setup_report.cab". Send this CAB file to Product Support if you need assistance with installation. With this information, Product Support can determine whether the installation was executed properly. CAB files that were generated during earlier installation processes are saved with a date ID in the "Reports" directory.

See also

Notes on licenses (Page 9)

Starting installation (Page 41)

Checking availability of updates and support packages and installing them (Page 53)

Displaying the installed software (Page 45)

Modifying or updating installed products (Page 47)

Repairing installed products (Page 49)

Starting to uninstall (Page 51)

Starting installation 6

Introduction

Software packages are installed automatically by the setup program. The setup program starts once the installation medium has been inserted in the drive.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To install the software packages, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click "Read Notes" or "Installation Notes".
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click "Next".
   The dialog for selecting the product languages opens.
6. Select the languages for the product user interface, and click "Next".

   Note

   "English" is always installed as the basic product language.

   The dialog for selecting the product configuration opens.
7. Select the products you want to install:
   - If you wish to install the program in a minimal configuration, click "Minimal".
   - If you wish to install the program in a typical configuration, click "Typical".
   - If you wish to personally select the products to be installed, click "User-defined". Then select the check boxes for the products you wish to install.
8. If you want to create a shortcut on the desktop, select the "Create desktop shortcut" check box.

9. Click "Browse" if you want to change the target directory for the installation.

   Note

   Please note the following:
   - The length of the installation path must not exceed 89 characters.
   - You can only change the installation path if no other product from the software package which you intend to install, has been installed, yet.
   - To increase security, select a directory that is protected by administrative rights as the target directory.

10. Click "Next".
    The dialog for the license terms opens.
11. To continue the installation, read and accept all license agreements and click "Next".
    If changes to the security and permission settings are required in order to install the TIA Portal, the security settings dialog opens.
12. To continue the installation, accept the changes to the security and permissions settings, and click "Next".
    The next dialog displays an overview of the installation settings.
13. Check the selected installation settings. If you want to make any changes, click "Back" until you reach the point in the dialog where you want to make changes. Once you have completed the desired changes, return to the overview by clicking on "Next".
14. Click "Install".
    Installation is started.

    Note

    If no license key is found during installation, you have the chance to transfer it to your PC. If you skip the license transfer, you can register it later with the Automation License Manager.

    Following installation, you will receive a message indicating whether the installation was successful.

15. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now.". Then click "Restart".
16. If the computer does not reboot, click "Exit".

Result

The TIA Portal along with the products and licenses you have ordered and the Automation License Manager have been installed on your computer.

See also

Installation log (Page 39)

Notes on the system requirements (Page 27)

Notes on licenses (Page 9)

Displaying the installed software (Page 45)
Modifying or updating installed products (Page 47)
Repairing installed products (Page 49)
Starting to uninstall (Page 51)

Displaying the installed software 7

You can find out which software is installed at any time. In addition, you can display more information on the installed software.

Procedure

To display an overview of the software installed, follow these steps:

1. Click "Installed software" in the "Help" menu.
   The "Installed software" dialog opens. You will see the installed software products in the dialog. Expand the entries to see which version is installed in each case.
2. If you would like to display additional information on the installed automation software, click the link on the "Detailed information about installed software" dialog.
   The "Detailed information" dialog opens.
3. Chose the topic you want more information about in the area navigation.

See also

Notes on the system requirements (Page 27)

Notes on licenses (Page 9)

Starting installation (Page 41)

Modifying or updating installed products (Page 47)

Repairing installed products (Page 49)

Starting to uninstall (Page 51)

Installation log (Page 39)

Modifying or updating installed products 8

You have the option to modify installed products using the setup program or to update to a new version.

Blocks with know-how protection from earlier versions of the TIA Portal are not automatically upgraded with the project. Remove the know-how protection of the blocks before you update the TIA Portal. Then set up the know-how protection with the current version of TIA Portal. For more detailed information, refer to the information system.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To modify or update installed products, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the installation variant opens.
6. Select the "Modify/Upgrade" option button and click the "Next" button.
   The dialog for selecting the product languages opens.
7. Select the check boxes of the product languages that you want to install. You can remove previously installed product languages by clearing the corresponding check boxes.

   Note

   Note that the product language "English" cannot be removed.

8. Click the "Next" button.
   The dialog for selecting the product configuration opens.
9. Select the check boxes of the components that you want to install. You can remove previously installed components by clearing the corresponding check boxes.

10. Click the "Next" button.

    Note

    Note that you cannot change the target directory because the existing installation is being modified.

    If changes to the security and permission settings are required in order to install the TIA Portal, the security settings dialog opens.

11. To continue the installation, accept the changes to the security and permissions settings, and click the "Next" button.
    The next dialog displays an overview of the installation settings.

12. Click the "Modify" button.
    This starts the installation of the additional components.

    Note

    Following installation, you will receive a message indicating whether the existing installation was successfully changed.

13. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".

14. If the computer does not reboot, click "Exit".

Result

The existing installation has been modified on your computer.

See also

Notes on the system requirements (Page 27)

Notes on licenses (Page 9)

Starting installation (Page 41)

Displaying the installed software (Page 45)

Repairing installed products (Page 49)

Starting to uninstall (Page 51)

Installation log (Page 39)

Repairing installed products

9

You have the option to repair installed products by completely reinstalling them using the setup program.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To repair installed products, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the installation variant opens.
6. Select the "Repair" option button, and click the "Next" button.
   The next dialog displays an overview of the installation settings.
7. Click the "Repair" button.
   This starts the repair of the existing installation.

   Note

   Following installation, you will receive a message indicating whether the existing installation was successfully repaired.

8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

Result

The installed products have been reinstalled.

See also

Notes on the system requirements (Page 27)
Notes on licenses (Page 9)
Starting installation (Page 41)
Displaying the installed software (Page 45)
Modifying or updating installed products (Page 47)
Starting to uninstall (Page 51)
Installation log (Page 39)

Starting to uninstall 10

Introduction

Software packages are removed automatically by the setup program. Once started, the setup program guides you step-by-step through the entire removal procedure.

You have two options for removing:

- Removing selected components via the Control Panel
- Removing a product using the installation medium

Note

The Automation License Manager will not be removed automatically when you remove the software packages, because it is used for the administration of several license keys for products supplied by Siemens AG.

Removing selected components via the Control Panel

To remove selected software packages, follow these steps:

1. Open the Control Panel with "Start > Control Panel".
2. Click "Programs and Features".
   The "Uninstall or change a program" dialog opens.
3. Select the software package to be removed and click the "Uninstall" button.
   The dialog for selecting the setup language opens.
4. Select the language in which you want the dialogs of the setup program to be displayed and click "Next".
   The dialog for selecting the products you want to remove opens.
5. Select the check boxes for the products that you want to remove and click "Next".
   The next dialog displays an overview of the installation settings.
6. Check the list with the products to be removed. If you want to make any changes, click the "Back" button.
7. Click the "Uninstall" button.
   Removal begins.
8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

Removing a product using the installation medium

To remove all software packages, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Select the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click "Read product information" or "Read installation notes".
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the installation variant opens.
6. Select the "Uninstall" option button and click the "Next" button.
   The next dialog displays an overview of the installation settings.
7. Click the "Uninstall" button.
   Removal begins.
8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

See also

Installation log (Page 39)

Notes on the system requirements (Page 27)

Notes on licenses (Page 9)

Starting installation (Page 41)

Displaying the installed software (Page 45)

Modifying or updating installed products (Page 47)

Repairing installed products (Page 49)

Installing updates and support packages 11

11.1 Checking availability of updates and support packages and installing them

By default, the TIA Portal checks automatically if new software updates or support packages are available, for example, Hardware Support Packages (HSPs). The automatic search for updates takes place after each computer restart and then cyclically every 24 hours. You can also deactivate the automatic search at any time or reactivate it. You can also search for updates manually.

If updates are found, you can download and install them.

Note

Updates and support packages from TIA Portal V13 or higher are supported.

Deactivate or activate automatic search for software updates

To deactivate or reactivate the automatic search for software updates, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Software Updates" group in the area navigation.
3. Deselect the "Check for updates daily" check box if you want to deactivate the automatic search for software updates.
4. Select the "Check for updates daily" check box if you want to reactivate the automatic search for software updates.

Manually searching for software updates

If you want to search for software updates manually, follow these steps:

1. Click "Installed software" in the "Help" menu.
   The "Installed software" dialog opens.
2. Click "Check for updates".
   The "TIA Updater" opens and the available updates are displayed.

Or:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Software Updates" group in the area navigation.
3. Click "Check for updates now".
   The "TIA Updater" opens and the available updates are displayed.

Or:

1. Open the TIA Updater with "Start > All Programs > Siemens Automation > Totally Integrated Automation UPDATER".
2. Click "Check for updates".
   The available updates are displayed.

Setting the server

You must set the corresponding server depending on whether you want to download updates and/or support packages from the TIA Automation Update Server or from a company-internal server. To do this, follow these steps:

1. Open the TIA Updater.
2. Click "Options" and select the "TIA Automation Software Update Server" option in the dialog that appears in the "Server used to check for updates" area.
   The software searches for available updates on the server of the manufacturer.

Or:

1. Open the TIA Updater.
2. Click "Options" and select the "User-defined server" option in the dialog that appears in the "Server used to check for updates" area.
3. Enter the server URL which you received from your administrator. The URL must be entered either in the format https://[URL of the server]/[path to the file directory] or in the format https://[server IP address]/[path to the directory]. If you are not sure which format to use, contact your administrator!
   The software looks for available updates on the server of your company.
4. Check the name of the production line and change this if necessary.
   You can find detailed information on creating projects and working with production lines in the help on the TIA Updater Corporate Configuration Tool.

You can set a different server on which the updates are to be searched for at any time. However, changing this setting is blocked during a download process. After switching the server, all downloaded updates and support packages are displayed, even if they are not available on the currently set server.

Downloading and installing software updates separately

To download and install available software updates and support packages, follow these steps:

1. If the TIA Updater is not displayed, open it in one of these ways:
   - If the automatic search has found updates, you receive a message in the information area of the Windows taskbar and the icon for the TIA Update is displayed in the information area. Click either the message or the icon to open the TIA Updater.
   - Click "Installed software" in the "Help" menu and then "Check for updates".
   - In the Windows Start menu, click "Start > All Programs > Siemens Automation > Automation Software Updater".
2. Click "Download" in the line of the update or support package that you want to install.
   The update or support package is downloaded. The associated "Install" button becomes active as soon as the download process has been completed.

   Note

   Please note the following:

   1. You can initiate multiple download processes simultaneously.
   2. You can log off or also shut down the computer while the download process is running. In such cases, the download process continues in the background as soon as you log on again.
   3. In some cases, a link to an external web page is displayed instead of the "Download" button. In this case, download the software update from this site and install it manually.

3. Close the TIA Portal if it is still open.
4. In the TIA Updater, click the "Install" button of the software package you want to install.
   The installation dialog appears.

   Note

   Please note the following:

   1. It is not possible to install multiple updates simultaneously.
   2. Do not log off during an installation and do not shut down the computer. This avoids inconsistent versions of the software on your computer.

5. Click "Next".
   The selected product is installed.

Downloading and installing several support packages in one step

To download and install available support packages in one step, proceed as follows:

1. Activate the required lines. You have several options for this:
   - To download and install all displayed support packages, activate the check mark in the title of the list.
   - To download and install several support packages, select the first required support package, hold down the <Ctrl> key and then select all further required support packages. Then set the check mark in one of the selected lines.
   - To download and install several support packages in a certain order, select the first required support package, hold down the <Shift> key and then select the last required support package.
2. Click "Download and Install".
3. After downloading click "Next" in the displayed dialog to start the installation. A dialog box is then displayed which informs you about the progress of the installation.

Note

Only support packages that are not subject to export restrictions can be downloaded and installed.

Alternative procedures for the installation of support packages

Another procedure is available for the installation of a support package. To do this, follow these steps:

1. Click "Support packages" in the "Options" menu of the TIA Portal.
   The "Detailed information" dialog opens. A table lists all support packages from the directory that you selected as the storage location for support packages in the settings.
2. If you want to install a support package that is not in the list, you have the following options:
   - If the support package is already on your computer, you can add it to the list by selecting "Add from file system".
   - If you add a support package from the "Service & Support" page on the Internet, first you download it by selecting "Download from the Internet". Then you can add it from the file system.
3. Select the support package that you want to install.
4. Click "Install".
5. Close and then restart the TIA Portal.

See also

Installation log (Page 39)

11.2 Working with a company-internal server

11.2.1 Properties and advantage of a corporate server

Introduction

Using a corporate server, you can place selected updates/support packages on a local server and make them available to users. This has the advantage that users do not have to access the Internet, but can install updates via the intranet, an external hard disk, etc. Since users do not need direct access to the Internet, the protection against trojans or malicious software that contacts the Internet from the internal company network is significantly increased.

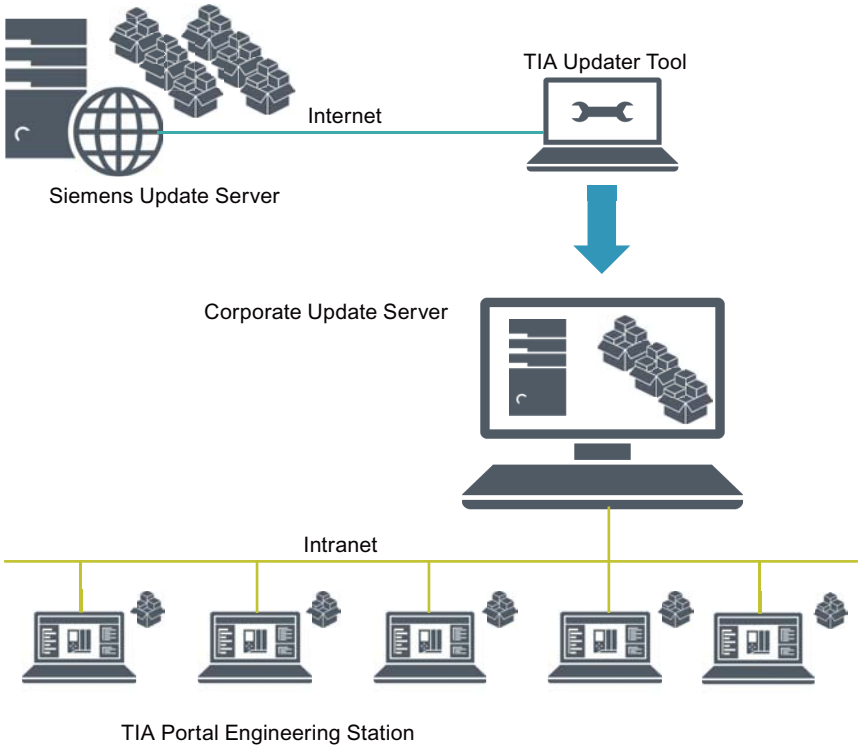

Configuring the server and creating projects

In the first phase, the corporate server is configured by a server administrator and the updates / support packages are deployed using the TIA Updater Corporate Configuration Tool. In addition, projects for different production lines can be created through which users can receive the updates they require. Users need to have access to the server area and, if applicable, have the name of the production line and be informed about the storage directory.

You can find detailed information on creating projects and working with production lines in the help on the TIA Updater Corporate Configuration Tool.

Working with the server

With the TIA Updater, users can download and install the updates and support packages that are relevant and relevant for them. Multiple download operations can be initiated at the same time. The installation of the updated / support packages must be done one after the other.

11.2.2 Configuring a company-internal server for updates

Introduction

To provide available updates and support packages to users from a central location, you must configure a company-internal server. Use the Microsoft Server Manager to do this.

The following is an example of the steps required to create and configure the server using the TIA Updater Corporate Configuration Tool (with the Microsoft Server 2016 operating system). Further settings which may be required for operation in your company are **not** covered in this description!

Note

Please note that the structure of the start menu and the storage location of the programs may vary depending on the different operating systems.

For more detailed information on configuration and operating the Server Manager, refer to the Microsoft help.

Requirement

You must have administrator rights.

Install Web server role (IIS)

To install the required Web server role, follow these steps:

1. Open the start menu and select "Server Manager".
2. Click "Add roles and features" in the dashboard.
3. Use the "Add roles and features" wizard to add the Web server roles.
   Before the wizard starts, it is verified that a complex password has been assigned to the administrator, that the network settings have been configured accordingly and that the latest Windows security updates have been installed.
4. Select "Role-based or feature-based installation" as the installation type and click "Next".
5. Select the target server and click "Next".
6. Note the pre-selected roles that are installed by default, and then select the additional role "Web Server (IIS)".
7. Click "Add features" followed by "Next". The features for the web server are displayed.

8. Click "Role 'Services" and make sure that the following features are selected or select them:
   "Web server" area
   - Default document
   - Directory browsing
   - HTTP errors
   - Static content

   "Health and diagnostics" area
   - HTTP logging
   - Request monitor

   "Performance features" area
   - Static content compression

   "Security" area
   - Request filtering

   "Application Development Features" area
   - .NET Extensibility 3.5
   - .NET Extensibility 4.6
   - ASP.NET 3.5
   - ASP.NET 4.6
   - ISAPI Extensions
   - ISAPI Filters
9. Click "Next".
10. Check your selection in the "Confirm installation selection" dialog box and click "Install".

"IIS" is added in the dashboard and can be further configured.

Create website

1. In the navigation area, click "IIS" and right-click in the "Servers" area and select "Internet Information Services (IIS) Manager".
2. In the "Connections" area, click "Sites" and then "Default Web site".
3. Right-click to select the "Add Virtual Directory..." option.
4. In the dialog that opens, enter a display name in the "Alias" field, for example, "TIAPortalUpdates".
5. In the "Physical path" field, enter the physical path of the folder in which the website is located or click the button to browse (**...**), in order to search for the folder in the file system.
6. Click "Test Settings..." to check whether the settings are correct.
7. Click "OK".

Activating virtual directory

1. In the area "Connections > Sites > Default Web site", select the virtual directory you created. In the "Features" view, double-click the item "Browse directory".
2. Click "Enable" in the "Actions" area.

Adding and configuring MIME type

1. In the "Connections" area, select the website you created.
2. Double-click "MIME type" in the "Features" view.
3. Click "Add" in the "Actions" area.
4. Enter .* in the "File name extension" text field in the "Add MIME type" dialog.
5. Enter "all/files" in the "MIME type" text field.
6. Click "OK".

Activate BITS-IIS server extension

1. Switch to the virtual directory in the IIS Manager.
2. Double-click "BITS-Uploads" in the Features view of the virtual directory.
3. Select the "Allow clients to upload files" check box and click "Apply".

Create a self-signed server certificate

1. In the "Connections" area, navigate to the level which you want to manage.
2. Double-click "Server Certificates" in the "Features" view.
3. Click "Create Self-Signed Certificate" in the "Actions" area.
4. In the "Create Self-Signed certificate" dialog, enter a display name for the certificate in the "Specify a friendly name for the certificate" field and click "OK".

Create SSL binding

1. Expand the "Sites" entry in the "Connections" area, and then click the site to which you want to add a binding.
2. Click "Bindings" in the "Actions" area.
3. In the "Site Bindings" dialog, click "Add".
4. In the "Add Site Binding" dialog, select the certificate you created under "Type" "https" and under "SSL certificate" and then click "OK".

Verify SSL binding

1. In the "Actions" area, click the previously created binding under "Browse website".
   An error page is opened in Internet Explorer, because the self-signed certificate was created by your computer.
2. Click "Continue to this website (not recommended)".
   This message is no longer displayed when you add the certificate to the "Trusted Root Certification Authorities" certificate memory.

Configure SSL settings

1. Double-click "SSL settings" in the "Features" view.
2. Use one of the following procedures in the "Client certificates" area of the "SSL Settings" dialog:
   – Select "Ignore" if no client certificates are to be accepted even if a client has a certificate.
   – Select "Accept" if client certificates are to be accepted.
   – Select "Require" if client certificates are to be required. The "Require SSL" option must be activated to allow the "Require client certificates" option to be used.
3. Click "Apply" in the "Actions" area.

Distributing updates to different areas

It may be the case that different departments need different updates or support packages. In this situation, we recommend operating multiple servers which offer different updates and support packages. To do this, follow these steps:

1. Set up different company-internal servers as described above.

or

1. Install the Web server role.
2. Set up multiple websites.
   Make sure that you assign unique names for these websites and the physical paths so that there will be no confusion.
3. Set the features described above for the Web server or the websites.

You can now store the required updates and support packages for the different departments in the defined directories.

11.2.3 Providing updates on a company-internal server

Introduction

In the TIA Updater Corporate Configuration Tool, you can set a company-internal server on which the available updates/support packages are stored and can be made available to the users.

Requirement

You must have administrator rights.

A corporate server can only be set up on a Microsoft server operating system, because it requires the operation of the BITS IIS service and a running IIS sever.

Adding updates from the TIA Automation Software Update Server

To add updates from the TIA Automation Software Update Server, follow these steps:

1. Open the TIA Updater Corporate Configuration Tool.
   The available updates are displayed.
2. Click the "Add update" button and select the "Add TIA Automation Software Update Server" check box in the dialog that appears.
3. In the dialog "Add update from TIA Automation Software Update Server", select the required updates (software and support packages) and click "Add".
   If updates are already located on the company-internal server, these are grayed out. You cannot load them again. During the download process, the status and the remaining time is displayed in the dialog.

Canceling the download process

To cancel the process, follow these steps:

1. Click "Cancel download".
2. Click "Yes" to confirm the dialog that appears.
   The download of the update is cancelled and it is deleted from the list.

Deleting updates from the company-internal server

To delete updates, follow these steps:

1. Select the required updates and click "Remove".
2. Click "Yes" to confirm the dialog that appears.
   The selected updates have been deleted from the list.

Server options

In the dialog "TIA Automation Software Update Server", click "Settings" to determine the following:

1. Under "Server path", specify the folder in which the downloads are to be saved. You can select either a local drive or a network drive.

   Note

   When you work directly on the server, the target directory is a equivalent to the provision directory on the server. All changes are applied directly. This procedure is not recommended, as conflicts can occur with active downloads.

   We recommend that you work on another computer; the target directory can be any directory. The content of this directory must then still be copied to the provision directory. Here, also, make sure that no conflicts occur with active downloads.

2. Select the check box "Always run server in background (icon is shown in the taskbar)", if the server is to always run in the background.
3. Confirm your entries with "OK".

See also

Properties and advantage of a corporate server (Page 57)

Configuring a company-internal server for updates (Page 58)

Installing support packages automatically 12

12.1 Installing support packages automatically

Introduction

As of V15.1, you can use the Support Package Installer to install or upgrade all support packages automatically via the command line; these include HSPs (hardware support packages) that are compatible with the installed version of the TIA Portal (e.g. isp.15_1).

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs have been closed.

Procedure

To start the installation with the desired options directly via the command line, proceed as follows:

1. Open the Windows command prompt with "Start > Run > cmd".
2. Switch to the directory that contains the "Siemens.Automation.SupportPackageInstaller.exe" file. This is the installation directory of your installed TIA Portal.
3. In the command prompt, enter the following command:
   %Installation directory% > Siemens.Automation.SupportPackageInstaller.exe <Support Package(HSP)-storage directory> [-l <Name of log file>] [-warnaserror]

<Support Package-storage_directory>
Mandatory directory containing the support package files (*.isp). This can be either a local folder or a Remote Share.

-l <Storage location of log file>
This parameter is optional and specifies the path to the log file. If it is not set, the log file is created in the binary directory. However, you need to have write permissions in the storage folder.

-warnaserror
This parameter is optional and indicates that every message that would usually have been reported as a warning will be reported as an error instead. If a HSP does not meet the necessary conditions, for example, a corresponding return value/abort value is output to the command line and the affected HSP is skipped.

-?
This parameter is optional and shows the Help.

Note

- If the storage directory or the name of the log file contains spaces, you need to place the name in quotation marks.
- Only support packages located directly in the storage directory are considered; subdirectories are ignored. This means that the *.isp files must be located directly in the storage directory.

See also

Return values from the installation process (Page 67)

Log file (Page 68)

12.2 Return values from the installation process

Core statement

The following table shows the return values and their descriptions:

Code	Description
0	The installation was successful.
1	The installation was successful and a restart is required.
-1	The installation failed.
-2	The installation failed: The TIA Portal is still running.
-3	The installation failed: Another installation is already running.
-4	The installation failed: You do not have sufficient rights. The installation requires administrator rights.
-6	The installation failed: Insufficient memory space.
-7	The installation failed: A restart is required before the installation can start.

See also

Installing support packages automatically (Page 65)

Log file (Page 68)

12.3 Log file

Contents of the log file

The log file contains detailed information about every installation. If the log file already exists, it is supplemented. Headers and footers are created for each log file between which you can find the desired information.

An entry is generated for the following events:

- Support packages that are listed in the storage directory and are available during the installation but could not be installed due to missing products. The entry in the log file indicates which products and which versions of these products are missing.
- Support packages that could not be installed successfully.
- Support packages that were installed successfully.

The file name of the HSP including the file extension is logged for each entry.

See also

Installing support packages automatically (Page 65)

Return values from the installation process (Page 67)

Installing and uninstalling the migration tool 13

13.1 System requirements

System requirements for the migration tool

The following system requirements apply to the use of the migration tool:

- All products used to create the source project must be installed. The following products are supported:
  - As of STEP 7 V5.4 SP5
  - WinCC V7.4 SP1 with the latest updates
  - As of WinCC flexible 2008 SP2
  - Integrated projects from STEP 7 V5.4 and the WinCC products listed above
  - STEP 7 Distributed Safety V5.4 SP5
  - F-Configuration Pack V5.4 SP5 up to V5.5 SP12
  - SINUMERIK STARTER and Startdrives
  - As of SIMOTION SCOUT V4.4
    You need the SCOUT Migration Tool PlugIn in the corresponding SCOUT TIA version to migrate SIMOTION SCOUT projects.
- All optional packages needed to process the STEP 7 project are installed. For example, all HSPs for the devices used in the source project are required.
- Microsoft .Net Framework 4.6.2 is installed. If Microsoft .Net Framework is not installed, the required version will be installed using the migration tool.

13.2 Installing the migration tool

Distribution of the migration tool

You can find the migration tool in the "Support" directory on the installation DVD of the TIA Portal. Alternatively, it is available for download from the Siemens Industry Online Support. For some products, (such as SIMATIC Failsafe or SIMOTION) additional plug-ins are required for the migration tool. These plug-ins can also be downloaded from the Siemens Industry Online Support or installed from the installation DVD of the specific products.

Normally, the migration tool is installed without the TIA Portal. Because the TIA Portal has its own integrated migration function, a separate installation of the migration tool is not necessary.

Procedure

To install the migration tool, proceed as follows:

1. Download the installation file from the Siemens Industry Online Support, or use the installation file from the "Support" directory of the installation DVD of the TIA Portal to perform the installation.
2. Run the installation file.
   The setup program for the migration tool will open.
3. First, select the language in which the setup should be displayed and click "Next".
   The page for selecting the software language is displayed.
4. Since the migration tool is provided exclusively in English, you cannot choose any other language for the installation. Therefore, click "Next" to proceed to the next step.
   The page for selecting the product is displayed.
5. The migration tool consists solely of a software component. Therefore, the migration tool is already selected. Click "Next".
   The page for confirming the licensing terms is shown.
6. Click on the entries in the list of license terms to read the license terms and security information. If you agree to the license terms, select the check box "I accept all terms of the listed license agreement(s)" and "I confirm that I have read and understood the security information on the safe operation of the products".
7. Then click "Next".
   An overview of the installation is displayed.
8. Click "Install".
   The installation is performed with the displayed settings.
   A restart of the computer is required after the installation of the migration tool.

13.3 Uninstalling the migration tool

The migration tool can be removed using the Control Panel.

Procedure

To remove the migration tool, follow these steps:

1. Open the Control Panel.
2. Double-click on "Programs and Functions" in the control panel.
   The "Uninstall or Change a Program" dialog opens.
3. Select the entry for the migration tool in the "Uninstall or Change a Program" dialog, and click "Uninstall".
   A security prompt appears.
4. Click "Yes" to confirm this prompt.
   The setup program starts.
5. Select the installation language and click "Next".
6. Select the migration tool to uninstall and click "Next".
7. Click "Uninstall".
   The migration tool will be removed.
8. Click on "Exit" to close the setup program.

Installing and uninstalling the Multiuser server 14

14.1 Notes on the installation of the multiuser server

Introduction

Installation of the multiuser server enables you to use the Multiuser Engineering functionality.

You can install the multiuser server as follows:

- Together with the installation of the TIA Portal.
- As standalone installation by means of a separate installation process without TIA Portal.

Note

Administrator rights are required for the server installation.

Installation of the multiuser server with the TIA Portal

The multiuser server is installed together with the following TIA Portal products:

- SIMATIC STEP 7 Basic
- SIMATIC STEP 7 Professional
- SIMATIC WinCC Basic
- SIMATIC WinCC Comfort/Advanced
- SIMATIC WinCC Professional

The multiuser server is usually activated in the product selection by default and is then installed with the TIA Portal. Check the settings during installation of the TIA Portal. You can find the specific default settings for the Multiuser Server during setup when selecting the product in the "Options" directory. By selecting or clearing the options, you can specify if the multiuser server is to be installed along with the TIA Portal or not.

Installation of the multiuser server as standalone installation

The installation of the multiuser server can also be a standalone installation.

A description of automated installation is also available on the product DVD in the directory "Documents\Readme\<language directory>".

Identical TIA Portal versions during installation

When installing different TIA Portal products on various clients, make sure that all participants use the same versions of software products for the installation.

For example, if you have installed STEP 7 Professional, WinCC Advanced and Safety as well as the multiuser server with version Vx.y, all other Team Engineering devices must also have installed this software in identical versions.

The service packs and updates must be installed for all products at the same time.

Installation path

Do not use any UNICODE characters (for example, Chinese characters) in the installation path.

Installation of a newer multiuser server

Make sure that you **install the new multiuser server first** and then uninstall the older version of the multiuser server.

In this way, you ensure that the user authorizations for the server can be taken over for the new server. If all existing multiuser servers are completely deleted before the new installation, the user settings are lost and must be created again.

Antivirus programs

During the installation, read and write access to installed files is necessary. Some antivirus programs block this access. We therefore recommend that you disable antivirus programs during the installation of the multiuser server and enable them again afterwards.

Supported virtualization platforms

You can also install the multiuser server as well as the TIA Portal in a virtual machine.

For this purpose, use one of the following virtualization platforms in the specified version or a newer version:

- VMware vSphere Hypervisor (ESXi) 6.5
- VMware Workstation 12.5.7
- VMware Player 12.5.7
- Microsoft Hyper-V Server 2016

Installation of a visualization platform has the same requirements as the installation of the TIA Portal.

See also

Installing the multiuser server (Page 75)

Uninstalling the multiuser server (Page 77)

14.2 Installing the multiuser server

Introduction

The multiuser server can be installed as part of the installation of the TIA Portal or standalone. If the multiuser server is installed with the TIA Portal, the version of the multiuser server installed will always correspond to that of the TIA Portal.

A description of automated installation is also available on the product DVD in the directory "Documents\Readme\<language directory>".

Note

New installation before uninstallation

Make sure that you **install the new multiuser server first** and then uninstall the older version of the multiuser server.

In this way, you ensure that the user authorizations for the server can be taken over for the new server. If all existing multiuser servers are completely deleted before the new installation, the user settings are lost and must be created again.

Installation rules

Multiple versions of the multiuser server can be installed on a computer and can use different project versions of the TIA Portal.

Usable project versions:

Multiuser server version	TIA Portal project version
Server V15.1	TIA Portal V15.1, V15 and V14
Server V15	TIA Portal V15 and V14
Server V14	TIA Portal V14

Note, however, that each version of the TIA Portal and the multiuser server only has the functionality contained in this version.

Multiuser Engineering does not upgrade the project versions.

Requirements for installation

Installation of the multiuser server has the same requirements as the installation of the TIA Portal.

Installing the multiuser server with the TIA Portal

Follow these steps for installation:

1. Note the requirements for the installation of the selected software package.
2. Start the setup program of the required product by inserting the installation medium into the respective drive.

3. Select the preferred settings and click the "Install" button.
4. Make sure that the multiuser server is selected in the product selection and observe the notes in the setup dialogs.

Result

The multiuser server is installed on your computer together with the TIA Portal.

Installing the multiuser server as standalone product

Follow these steps for installation:

1. Note the requirements for the installation of the multiuser server.
2. Start setup for the standalone installation of the multiuser server with a double-click on the self-extracting .exe file "TIA_Portal_Multiuser_Server_V<x.y>.exe". You can find these files in the directory "Support" on the corresponding product DVD.
3. Select the preferred settings and click the "Install" button.
4. Follow the instructions in the setup dialogs.

Result

The multiuser server is installed on your computer.

See also

Notes on the installation of the multiuser server (Page 73)

Uninstalling the multiuser server (Page 77)

14.3 Uninstalling the multiuser server

Introduction

You can use the Control Panel to remove the multiuser server.

Note

New installation before uninstallation

Make sure that you **install the new multiuser server first** and then uninstall the older version of the multiuser server.

In this way, you ensure that the user authorizations for the server can be taken over for the new server. If all existing multiuser servers are completely deleted before the new installation, the user settings are lost and must be created again.

Requirements for removal

There is a multiuser server installed on your computer.

Uninstalling the multiuser server

Proceed as follows for removal:

1. Open the Control Panel.
2. Click on "Uninstall a program" in the Control Panel.
   The "Uninstall or change a program" dialog opens.
3. Right-click the entry for the "Multiuser server" in the "Uninstall or change a program" dialog, and click "Uninstall".
   A security prompt appears.
4. Click "Yes" to confirm this prompt.

Result

The multiuser server is removed from your computer.

See also

Notes on the installation of the multiuser server (Page 73)

Installing the multiuser server (Page 75)

Index

A

Antivirus programs, 6, 31

D

Data backup, 19
Defective license, 20
Destruction of license keys, 17
Display PDF files, 27
Display Welcome Tour, 27

E

Encryption software, 31

H

Hardware requirements, 29
Hardware support package
    Installing, 55
HMI device, 22
HMI device licensing
    Non-PC-based, 22
HSP, (See support package)

I

Install update, 55
Installation
    Displaying software, 45
    Licenses, 9
    Log, 39
    Modifying products, 47
    Repairing products, 49
    run automatically, 5
    Starting, 41
    Support package, 55
    System requirements, 27
    Target directory, 5
    Updates, 55
    Updating products, 47
    Virtual environment (private cloud), 5
Installing license keys, 17, 18
Installing support packages, 55
Intrusion Detection System (IDS), 31
Invalid license
    With a time zone change, 19

L

License, 24
    Defective, 20
    Defective license, 20
    Starting without valid license, 15
    WinCC Runtime Professional Asian, 24
License for PC-CPs, 25
License key, 18
    Working with license keys, 18
License Manager Panel Plugin, 22
Licenses, 9

M

Migration tool
    Distribution and sources, 70
    Removing, 71
    System requirements, 69
Multiuser Engineering
    Installation of the multiuser server as standalone installation, 73
    Installation of the multiuser server with the TIA Portal, 73
    Introduction, 73
    Rules and requirement for installation, 75
    Starting the installation of the multiuser server, 75
    Uninstalling the multiuser server, 77

N

Non-licensed mode
    ES, 15
    HMI devices, 22
    Runtime, 23

O

Operating system, 29

R

Restore process
    License key, 19

S

Security program, 31
Software requirements, 29
Starting removal, 51
System requirements, 29

U

Uninstalling license keys, 18

V

Virtual Machine (VM)
    Supported virtualization platforms, 30

W

WinCC Runtime Professional Asian
    Licensing, 24
Windows, 29

SIEMENS

Overview of Migration Options 1

Migrating projects to a TIA Portal project 2

Migrating PLC programs to a S7-1500 CPU / ET 200SP 3

Migrating S7-1200 to firmware as of V4 (S7-1200) 4

Migrating projects and programs

Programming and Operating Manual

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Overview of Migration Options....7

2 Migrating projects to a TIA Portal project....9

2.1 Migration of projects with the TIA Portal....9

2.2 Check migration readiness of hardware components....11

2.3 Preparing projects with the migration tool....12
2.3.1 Migrating projects with the migration tool....12
2.3.2 Calling the migration tool....13
2.3.3 Creating a migration file....13

2.4 Migrating projects....15

2.5 Displaying the history of the migration....17

2.6 Display migration log....18

2.7 Migrating STEP 7 projects (S7-300, S7-400)....19
2.7.1 Migration of STEP 7 projects (S7-300, S7-400)....19
2.7.2 Requirements for the migration of STEP 7 projects (S7-300, S7-400)....20
2.7.3 Checking whether STEP 7 projects can be migrated (S7-300, S7-400)....21
2.7.4 Consistency check in the initial project (S7-300, S7-400)....22
2.7.5 Removing unsupported hardware components (S7-300, S7-400)....23
2.7.6 Special points to observe during migration (S7-300, S7-400)....24
2.7.6.1 Device names (S7-300, S7-400)....24
2.7.6.2 PROFINET IO configurations (S7-300, S7-400)....24
2.7.6.3 PROFIBUS configurations (S7-300, S7-400)....26
2.7.6.4 GSD and GSDML devices (S7-300, S7-400)....26
2.7.6.5 Connections (S7-300, S7-400)....26
2.7.6.6 Multiprojects (S7-300, S7-400)....27
2.7.6.7 Migrating program blocks (S7-300, S7-400)....27
2.7.6.8 Migration of LAD/FBD/STL programs (S7-300, S7-400)....31
2.7.6.9 Migrating GRAPH programs (S7-300, S7-400)....32
2.7.6.10 Migrating SCL programs (S7-300, S7-400)....33
2.7.6.11 Messages (S7-300, S7-400)....39
2.7.6.12 Reporting system errors (S7-300, S7-400)....40
2.7.6.13 Migration of project texts (S7-300, S7-400)....40

2.8 Migrating WinCC projects....41
2.8.1 Migrating WinCC flexible 2008 projects....41
2.8.1.1 Basics (WinCC flexible)....41
2.8.1.2 Migrating engineering data (WinCC flexible)....47
2.8.1.3 Migrating runtime data (WinCC flexible)....68
2.8.1.4 Migrating integrated projects (WinCC flexible)....71
2.8.1.5 Reference (WinCC flexible)....75
2.8.2 Migrating WinCC V7 projects....89
2.8.2.1 Basics (WinCC V7)....89
2.8.2.2 Migrating engineering data (WinCC V7)....96
2.8.2.3 Copying runtime data (WinCC V7)....126

2.8.2.4 Reference (WinCC V7)....126
2.8.2.5 Integrated projects....128

2.9 Migrating integrated projects....133
2.9.1 Migrating an integrated project....133
2.9.2 Post-editing integrated projects....135
2.9.3 Converting unspecified CPUs into specified CPUs....136
2.9.4 Creating an integrated HMI connection....137
2.9.5 Re-linking HMI tags....139
2.9.6 Delete non-integrated HMI connection....140

3 Migrating PLC programs to a S7-1500 CPU / ET 200SP....143

3.1 Basic information on migrating PLC programs (S7-1500)....143

3.2 Performing migration (S7-1500)....146

3.3 Displaying a log file (S7-1500)....148

3.4 Special considerations for migrating PLC programs (S7-1500)....149
3.4.1 Information on migrating PLC programs (S7-1500)....149
3.4.1.1 Migrating organization blocks (S7-1500)....149
3.4.1.2 Migrating hardware IDs (S7-1500)....150
3.4.1.3 Migrating IEC timers and IEC counters (S7-1500)....150
3.4.1.4 Migrating CPU data blocks (S7-1500)....151
3.4.1.5 Migrating diagnostics functions (S7-1500)....151
3.4.1.6 Migrating absolute access to local data (S7-1500)....152
3.4.1.7 Block parameters in S7-1500 (S7-1500)....153
3.4.1.8 Multi-instance capability in S7-1500 (S7-1500)....154
3.4.1.9 Floating-point numbers in S7-1500 (S7-1500)....155
3.4.1.10 Writing individual characters of a STRING to S7-1500 (S7-1500)....155
3.4.1.11 Access to the status word in S7-1500 (S7-1500)....156
3.4.1.12 Loading software changes to S7-1500 (S7-1500)....156
3.4.2 Migration of instructions to S7-1500 (S7-1500)....156
3.4.2.1 Information on migrating instructions (S7-1500)....156
3.4.2.2 Overview of non-migratable instructions (S7-1500)....158
3.4.2.3 Migration of data block instructions (S7-1500)....167
3.4.2.4 Migration of the instruction RDSYSST (S7-1500)....169
3.4.2.5 Migration of the T_CONV instruction (S7-1500)....171
3.4.2.6 Migrating alarms with associated values (S7-1500)....172
3.4.2.7 Migration of table instructions (S7-1500)....175
3.4.2.8 Migration of communication instructions (S7-1500)....179
3.4.3 Migration of LAD/FBD programs to S7-1500 (S7-1500)....185
3.4.3.1 Information on migrating LAD/FBD programs (S7-1500)....185
3.4.3.2 Partially qualified addresses in LAD/FBD (S7-1500)....186
3.4.3.3 Access to status bits in LAD/FBD (S7-1500)....187
3.4.4 Migration of STL programs to S7-1500 (S7-1500)....188
3.4.4.1 Information on migrating STL programs (S7-1500)....188
3.4.4.2 Logic operation sequences and jumps (S7-1500)....190
3.4.4.3 Passing values using registers in case of language change (S7-1500)....192
3.4.4.4 Passing values using registers or the status word with a block call (S7-1500)....193
3.4.4.5 Passing values using registers with the CC and UC instructions (S7-1500)....194
3.4.4.6 Fully qualified addresses in STL (S7-1500)....195
3.4.4.7 Partially qualified addresses in STL (S7-1500)....196
3.4.4.8 Access to the instance DB in STL (S7-1500)....198

3.4.5 Migration of SCL programs to S7-1500 (S7-1500)....200
3.4.5.1 Information on migrating SCL programs (S7-1500)....200
3.4.5.2 Indirect addressing in SCL (S7-1500)....201
3.4.5.3 Instructions in SCL (S7-1500)....203
3.4.6 Migration of GRAPH programs to S7-1500 (S7-1500)....205
3.4.6.1 Information on migration of GRAPH programs (S7-1500)....205
3.4.6.2 Changes in the GRAPH program (S7-1500)....206
3.4.6.3 Adaptation of the interface of GRAPH blocks (S7-1500)....208

4 Migrating S7-1200 to firmware as of V4 (S7-1200)....211

4.1 Basic information on upgrading to V4 (S7-1200)....211

4.2 Migrating to V4 (S7-1200)....214

4.3 Special considerations after migrating to V4 (S7-1200)....216

Index....219

Overview of Migration Options

1

Migration paths

If you want to continue to use existing projects or programs with the latest version of the TIA Portal and the S7-1500, you have several options for migrating projects.

The following graphic provides an overview of the migration options:

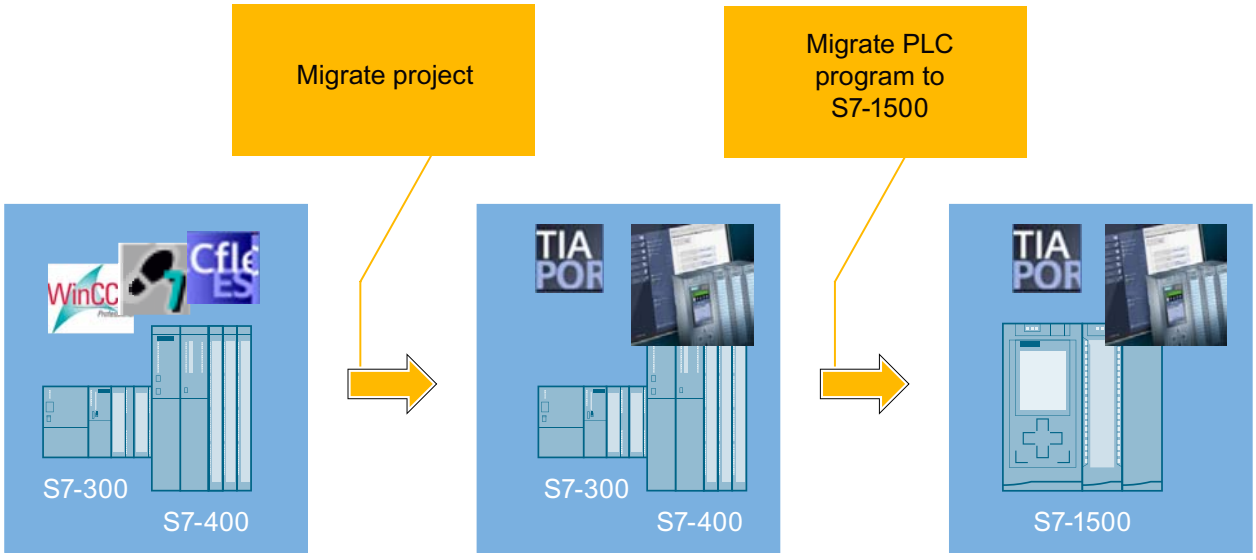

Migrating projects to the TIA Portal

You use the "Migrate project" function to migrate projects that were created outside of the TIA Portal, for example, with STEP 7, WinCC, WinCC flexible, or SINUMERIK.

The result of the project migration is a TIA Portal project that you can use with your existing S7-300/400 series hardware and existing HMI devices.

Migrating PLC programs from S7-300/400 to S7-1500

To migrate a PLC program within the TIA Portal from an S7-300/400 series device to an S7-1500, you use the PLC migration.

The result of the PLC migration is an executable PLC program that is automatically adapted to the new system architecture of the S7-1500 as far as possible.

Optimizing PLC program for S7-1500

You also have the option to optimize your program for the S7-1500 by employing newly introduced programming methods. The optimization enables you to effectively use the higher performance, innovated memory technology and new features of the S7-1500 system.

Upgrading projects

You can also continue to use projects from previous versions of the TIA Portal. However, these projects do not need to be migrated. You can upgrade projects from previous versions to the current version of the product. You can find additional information on compatibility and upgrading projects in the section "AUTOHOTSPOT".

Note

Additional support for migration

You can find the latest information about PLC migration in the Siemens Industry online support:

Migration of complete systems:

http://support.automation.siemens.com/WW/view/en/83558085 (http://support.automation.siemens.com/WW/view/en/83558085)

Migration of controllers:

http://support.automation.siemens.com/WW/view/en/83557459 (http://support.automation.siemens.com/WW/view/en/83557459)

Migration of visualization:

http://support.automation.siemens.com/WW/view/en/76878921 (http://support.automation.siemens.com/WW/view/en/76878921)

Migration of communication:

http://support.automation.siemens.com/WW/view/en/83558087 (http://support.automation.siemens.com/WW/view/en/83558087)

If you need further support, please contact SIMATIC Customer Support.

Migrating projects to a TIA Portal project 2

2.1 Migration of projects with the TIA Portal

Migration of existing projects

You can migrate projects from earlier automation solutions to the TIA Portal. Each time you migrate, a new project is created for the migrated data with which you can then work. Any TIA Portal projects already open are closed first.

The migration is then displayed in the table of the project history. From there, you have access to the migration log that is created automatically for the migration.

After the migration of hardware configurations and program blocks from earlier automation solutions, first check the functionality of the migrated project before you use it in productive operation.

Supported products for migration

Refer to the chapter "System overview STEP 7 and WinCC" to find out which products are available for the TIA Portal. All products listed there are generally supported by the TIA Portal during migration.

Any additional requirements that must be met depend on the initial products that were used and the currently installed products. For more information on the migration options for your products, you can, for example, refer to the Siemens Industry Online Support and the documentation of your software products.

See also: AUTOHOTSPOT

Procedure during migration

The migration process is divided into the following basic steps:

1. Preparing the initial project
   If the software for editing the initial project is not installed or not fully installed on the programming device/PC with the TIA Portal, or if the initial project is an integrated project, you must first convert the initial project to a migration file. To do this, install the migration tool on a programming device/PC on which the required software for editing the initial project is installed. Then, use the migration tool to convert the initial project, and copy the file to the programming device/PC on which the TIA Portal is installed. You can omit this step if the initial project and its associated software are on the same programming device/PC as the TIA Portal, and if the initial project is not an integrated project.
2. Performing migration
   Perform the migration within the TIA Portal. For the migration, specify as source either the migration file which you created with the migration tool or the initial project when all required software has been installed.

3. Checking the migration log
   A migration log is created for each migration. It contains information about modified project parts. You can call the log under "Common data > Logs" in the project tree or in the project history. After completion of the migration, the migration log will be displayed in the TIA Portal. Check the log following completion of the migration.
   If the migration failed, an XML file is created as a log under "\Logs" in the project directory. You can use any XML editor to open this log and view the reasons why the migration failed.
4. Correcting the migrated project
   Because the configurations of the initial project may not always be completely compatible with the TIA Portal, all configurations may not be reproduced identically in the migrated project. You should therefore work through the points in the migration log systematically. If you have not included the hardware configuration in the migration, you also have to convert the unspecified devices to the appropriate hardware.

Including the hardware configuration in the migration

By default, only the software parts of the project are included in the migration. An unspecified device is generated in the migrated project for the devices contained in the initial project. The hardware and network configurations and the connection are not migrated. You therefore convert the unspecified devices into suitable devices after the migration and recreate any network configurations and connections manually.

If you are certain that the hardware used in the initial project has a corresponding equivalent in the TIA Portal, you can include the hardware configuration in the migration. In this case, both the hardware configuration and the software are migrated. You can check with a tool (Page 11) to see which hardware components are supported.

See also

Display migration log (Page 18)

2.2 Check migration readiness of hardware components

Introduction

Siemens offers a tool that can be used to check whether the hardware configuration used in an initial project is ready for migration to the TIA Portal.

Components integrated via GSD or GSDML files cannot be checked. For such modules, check manually in the TIA Portal whether the modules are available in the hardware catalog. If the modules are not available there, install the required GSD or GSDML files in the TIA Portal. You can obtain the required files from the manufacturer of the components.

In the result you can also see which software products and licenses have to be available on the programming device/PC with the installation of the TIA Portal to perform a migration. You can also see as of which firmware version the individual modules of the initial project are supported in TIA Portal. The result of the check can be output in a Microsoft Excel file or in PDF format.

Download

The tool for checking the migration readiness is available for download in the FAQs of the Siemens Industry Online Support under entry number 60162195 (http://support.automation.siemens.com/WW/view/en/60162195).

Source files for the check

To check readiness, you require one of the following source files which contains the article number of the hardware used in the initial project:

- .cfg file
  You can export the .cfg file from HW Config (STEP 7) with the menu command "Export as .cfg file" in the "Station" menu. The .cfg file contains all MLFBs of the devices used in the currently open station.
- Microsoft Excel file (in .xls file format)
  Regardless of the initial project used, you can create a Microsoft Excel list containing all MLFBs of the devices you want to migrate.
- File in .csv format
  The article numbers that are to be checked can also be saved to a .csv file instead of a Microsoft Excel list. To do this, use a standard text editor and enter the MLFBs separated by comma and without space after the comma. Save the text file with the extension ".csv".

See also

Tool for checking migration readiness (http://support.automation.siemens.com/WW/view/en/60162195)

2.3 Preparing projects with the migration tool

2.3.1 Migrating projects with the migration tool

Preparation for migration

In many cases, a project that you wish to migrate will not be located on the same programming device/PC on which the latest version of the TIA Portal is installed. Therefore, the initial project must first be converted to a compatible format for the migration. The same applies to integrated projects.

After creation of the migration file, you copy the migration file to the programming device/PC on which the current version of the TIA Portal is installed. In the TIA Portal, you enter the migration file as the source for the migration and can create a project in the current file format of the TIA Portal.

Procedure for migration with the migration tool

The following steps are necessary to prepare a migration with the migration tool:

1. Install the migration tool on the programming device/PC where the source project is located. To do this, download the installation file from the Siemens Industry Online Support or install the migration tool from the setup DVD of the TIA Portal.
2. Start the migration tool, and use it to convert the source project to the migration file format with the file extension ".am15".
   For this step, make sure that all software needed to process the source project is installed on the programming device/PC. This also includes all necessary service packs, hardware support packages and all expansion software that is needed to process the initial project. If individual products are not installed it may not be possible to perform the migration or the migration may be incomplete.
3. Copy the migration file to the target system on which a current version of the TIA Portal is installed.
   Note that the target system must have been installed with all software needed to configure the complete set of devices contained in the migration.
4. Perform the migration within the TIA Portal and specify the migration file with the extension ".am15" as the source.
5. Once migration is complete, check the migration log and systematically work through the information provided there for the newly created project. Read the information in the Inspector window with special care after the first compilation of the configuration.

Including the hardware configuration in the migration

By default, only the software parts of the project are included in the migration. An unspecified device is generated in the migrated project for the devices contained in the initial project. The hardware and network configurations and the connection are not migrated. You therefore convert the unspecified devices into suitable devices after the migration and recreate any network configurations and connections manually.

If you are certain that the hardware used in the initial project has a corresponding equivalent in the TIA Portal, you can include the hardware configuration in the migration. In this case, both the hardware configuration and the software are migrated. You can check with a tool to see which modules are supported.

See also

Migration of projects with the TIA Portal (Page 9)

Migrating projects (Page 15)

Calling the migration tool (Page 13)

Creating a migration file (Page 13)

2.3.2 Calling the migration tool

Starting the migration tool

During the installation, a "Migration to TIA Portal V15.1" shortcut is created by default in the Start menu under "Siemens Automation > Migration Tool". Click this shortcut.

Alternatively, you can call the migration tool directly in Windows Explorer. During the installation, the migration tool is saved by default in one of the following folders:

- On a 64-bit operating system:
  C:\Program Files (x86)\Siemens\Automation\MIGTOOL_V15_1\Bin
- On a 32-bit operating system:
  C:\Program Files\Siemens\Automation\MIGTOOL_V15_1\Bin

To start the migration tool, click the "Siemens.Automation.MigrationApplication.exe" file in one of the directories.

See also

Creating a migration file (Page 13)

2.3.3 Creating a migration file

The section below describes how you can use the migration tool to convert the initial project to a migration file that can be read by the TIA Portal. Following conversion, this file is transferred to the target system and migrated there.

You can specify whether the migration file should contain the entire project, including the complete hardware configuration and the associated software, or whether you want to migrate the software only.

Requirement

- The suitable, original software with a valid license is installed for all configurations used in the initial project.
- The initial project is not provided with access protection.
- The initial project must be consistent, otherwise problem-free migration cannot be assured.

Procedure

To create the migration file, proceed as follows:

1. Select the path of the source file for the migration in the "Storage Location (Path)" field.
2. Specify the project parts that are to be migrated:
   - Select the "Include HW and Network data during the migration" check box to migrate not only the software but also the complete hardware parts and the network configuration of the project.
   - Select the "Copy SCADA runtime data" check box if you also want to migrate the runtime data, such as alarm archives, tag archives and user archives, in addition to the data of the engineering system.
3. Select the path and the file name for the migration file in the "Intermediate file" field.
4. Click the "Migrate" button.

Result:

A migration file is created. You now copy this file to the target system and migrate this file in the TIA Portal.

See also

Migrating projects (Page 15)

Calling the migration tool (Page 13)

Migrating projects with the migration tool (Page 12)

2.4 Migrating projects

Requirement

- A converted file in the format ".am15" is already available or the original software with a valid license is installed for all configurations used in the initial project.
- The initial project is not provided with access protection.
- The initial project must be consistent, otherwise problem-free migration cannot be assured.

Read the additional information on the requirements in the help for the respective products installed.

Note

System hibernation during the migration

While a migration is running, the system should not be changed to the standby or hibernate mode. Otherwise the migration will be aborted.

Procedure

To migrate a project, follow these steps:

1. Select the "Migrate project" command in the "Project" menu.
   The "Migrate project" dialog opens.
2. Specify the path and the file name for the project to be migrated in the "Source path" field.
   Select a project either in the ".am15" migration format or in the format of the initial project.
3. To include the hardware configuration in the migration, select the "Include hardware configuration" check box.
   If you have selected a migration file that was created with the migration tool, the check box cannot be selected. In this case, you must specify if you wish to include the hardware configuration in the migration before the conversion with the migration tool .
4. Select the "Copy WinCC Runtime Professional data" check box, if you also want to migrate the runtime data, such as alarm archives, tag archives and user archives, in addition to the data of the engineering system.
   If you have selected a migration file that was created with the migration tool, the check box cannot be selected. In this case, you must specify if you wish to include the SCADA runtime data in the migration before the conversion with the migration tool .
5. Select a name for the new project in the "Project name" box.
6. Select a path in the "Target path" box where the new project is to be created.
7. Enter your name or the name of another person responsible for the project in the "Author" field.
8. Enter a comment in the "Comment" box, if you require one.
9. Click "Migrate".

Result

The initial project is converted and a message appears after conversion is complete. The newly created project is then opened in the project view, and the migration log is opened in the TIA Portal.

Even if the migration failed, a project directory is created and a migration log in the form of an XML file is generated in this directory. The completion message that appears after the migration contains a link to this XML file. Click the link to open the XML file. Alternatively, you can find the XML file in the project directory under "\Logs".

See also

Post-editing integrated projects (Page 135)

Display migration log (Page 18)

Migrating projects with the migration tool (Page 12)

Creating a migration file (Page 13)

2.5 Displaying the history of the migration

If a project was created by migration, the migration will be listed in the table of the project history. You can open the migration log in the table. The time of the migration is also shown.

Procedure

To display the migration in an overview table, follow these steps:

1. Select the open project in the project tree.
2. Select "Properties" in the shortcut menu of the project.
   The dialog with the properties of the project opens.
3. Select the "Project history" group in the area navigation.
   The overview table is displayed.

2.6 Display migration log

A log is created for each successful migration. The log contains the following information:

- Migrated objects
- Modifications to objects made during migration
- Errors that occurred during migration
- In certain cases a link to more help with specific events.
  In this case, click the question mark to obtain more help.

Procedure

To display the log file of the migration, follow these steps:

1. Open the "Common data > Logs" folder in the project tree.
2. Double-click the desired log in the list.
   The contents of the log are displayed in the work area.

See also

Migration of projects with the TIA Portal (Page 9)

2.7 Migrating STEP 7 projects (S7-300, S7-400)

2.7.1 Migration of STEP 7 projects (S7-300, S7-400)

Introduction

You can migrate a project from SIMATIC STEP 7 V5.4 SP5. You can then continue to use this project in the TIA Portal. If you want to migrate projects from older STEP 7 versions, you must open, compile and save these projects first in STEP 7 V5.4 SP5. It is also possible to migrate integrated projects with devices from other software products.

Scope of migration

Generally, you can migrate all configurations and objects from SIMATIC STEP 7 V5.4 SP5 that the installed version of the TIA Portal supports. This includes, for example, the following devices and configurations:

- Devices of the S7-300 and S7-400 family.
- PROFIBUS configurations with connected distributed I/O. These include GSD-based slaves, I-slaves as well as direct data exchange.
- PROFINET configurations with distributed I/O. These include GSD-based devices and I-devices.
- Network configurations
- Connections
- Blocks created in the programming languages LAD, FBD or STL, S7-SCL, S7-GRAPH
- PLC tags
- User-defined data types (UDT)
- Alarms and alarm classes
- Interrupts
- User-defined attributes (UDA)
- User text libraries

Including the hardware configuration in the migration

By default, only the software parts of the project are included in the migration. An unspecified device is generated in the migrated project for each of the devices contained in the initial project. The hardware and network configurations and the connection are not migrated. You can convert the unspecified devices into suitable devices after the migration and create any network configurations and connections manually.

If you are certain that the hardware used in the initial project has a corresponding equivalent in the TIA Portal, you can include the hardware configuration in the migration. In this case, both the hardware configuration and the software are migrated.

Note

Additional support for migration

You can find the latest information about migration in the Siemens Industry Online Support:

Migration of controllers (http://support.automation.siemens.com/WW/view/en/83557459)

If you need further support, contact SIMATIC Customer Support.

See also

Reporting system errors (Page 40)

Migrating program blocks (Page 27)

2.7.2 Requirements for the migration of STEP 7 projects (S7-300, S7-400)

Various requirements for the software installed on the original PG/PC and for the initial project apply to the migration.

Requirements for the original PG/PC

The following requirements apply to the PG/PC:

- A licensed version of STEP 7 V5.4 SP5 or higher (we recommend V5.5 SP4) must be installed. For projects that only contain PC stations, it is sufficient if the SIMATIC NET PC software is installed.
- For all of the configurations used in the project, the corresponding additional software must be installed with a valid license, for example, optional packages
- All HSPs for modules not included in the hardware catalog must be installed.
- All GSD files used in the project must comply with the specifications.
- You must be logged on to the operating system with administrative rights.
- One of the following software products must be installed:
  - The TIA Portal with STEP 7 Professional V13 SP1 or higher
    If you do not migrate with the most recent version of the TIA Portal on the original programming device/PC, the functional scope of the project in more recent versions of the TIA Portal is initially limited. You first have to upgrade the project before you can use the full functional scope of more recent versions.
  - The migration tool

Requirements for the initial project

- The initial project must not be configured with access protection.
- The hardware and software must be consistent.
- The alarm number assignment must be set to CPU-oriented.
- The project cannot contain any protected blocks with time stamp conflict.
- It must be possible to compile all programs and their sources without errors.
- All called blocks must be available in the block folder.
- The block folder must contain no blocks that are not called, in particular no instance data blocks.
- We recommend that you migrate blocks unencrypted, because blocks encrypted with S7-Block Privacy cannot be read or modified in the TIA Portal.

Requirements for the PG/PC with STEP 7 Professional

- One of the following software products must be installed:
  - STEP 7 Professional V13 SP1 or later
  - STEP 7 Professional V13 SP1 or later (without license) when the project to be migrated only contains components that have been configured with the SIMATIC NET PC software.
- For all of the configurations used in the project, the corresponding additional software must be installed with a valid license, for example, optional packages.
- All HSPs for modules that are not contained in the hardware catalog and all GSD files used in the project must be installed.

2.7.3 Checking whether STEP 7 projects can be migrated (S7-300, S7-400)

Before you start the migration, check to determine whether all necessary requirements for migration have been met in the initial project.

Note

Note that the actual values are reset in the initial project when the following steps are carried out. Also note that running the block consistency check resets the start values in the instance DBs to the default values. If you are using technology functions, this behavior is not desired.

Procedure

To check whether a project can be migrated, follow these steps:

1. Open the initial project in SIMATIC STEP 7 V5.4 SP5. For correct matching, only the optional packages and hardware components available in the TIA Portal, including HSPs and GSD files, may be installed.
2. Open the individual stations. If no alarms indicating missing components are displayed when the stations are opened, all the components necessary for migration are available.
3. Perform a block consistency check for each block container contained in the project. For more information on the consistency check, refer to Chapter "Consistency check in the initial project (Page 22)".
4. Compile the entire project. If no errors are displayed during the compilation, the programs can be migrated.
5. Check whether all modules with an identical MLFB and the same firmware version are contained in the TIA Portal hardware catalog.
6. Check whether the CPU-oriented alarm number assignment is activated for each CPU.
7. Compile the project in NetPro. The compilation in NetPro must also be error-free.

See also

Consistency check in the initial project (Page 22)

FAQ entry on consistency check (http://support.automation.siemens.com/WW/view/en/5416540)

2.7.4 Consistency check in the initial project (S7-300, S7-400)

Consistent initial project

A consistent initial project is a requirement for the successful migration of a project to the TIA Portal. It is therefore advisable to perform a consistency check on the initial project before migration. You can find a guide to the consistency check in SIMATIC STEP 7 in an FAQ entry (http://support.automation.siemens.com/WW/view/en/5416540) on the Siemens Industry Online Support page or in the SIMATIC STEP 7 help.

Migrate start values in instance DBs for technology blocks

The compilation in the block consistency check causes the tags in instance DBs to be reset to the default values. The set start values are lost in the process.

To migrate the start values anyway, we recommend one of the following procedures:

- For smaller projects: Backup the instance DBs outside the block folder
  Backup the instance DBs before you start the consistency check. After the consistency check has been completed, copy the instance DBs back to the block folder.
- For larger projects: Create a copy of the project
  Make a copy of the project and then carry out the check on the copy of the project. Correct any errors found during the consistency check in the initial project.

You can then migrate the project with the configured values. In this way, the configuration of the technology functions is also retained.

2.7.5 Removing unsupported hardware components (S7-300, S7-400)

If you have created configurations in the initial project for which not all of the right software is installed, then you can still migrate the project by manually deleting the unsupported configurations from the project. Alternatively, you can use the "Reorganize" function.

Procedure

To clean the original project, follow these steps:

1. Use an installation of SIMATIC STEP 7 V5.4 SP5 that contains only the optional packages contained in the TIA Portal and all required modules.
2. Save the project again and select the "With reorganization" option when saving. Any unsupported configurations will be removed from the project.

For modules that are only available in a new version or with a new firmware version in the TIA Portal, use the "Exchange object" function in the station to exchange an older module for one that can be migrated.

2.7.6 Special points to observe during migration (S7-300, S7-400)

2.7.6.1 Device names (S7-300, S7-400)

Device names of PROFINET IO devices

In STEP 7 V5.5 the device name of a PROFINET IO device is used as the name of the IO devices. After the migration to a TIA Portal project, the device name of the PROFINET IO device from the initial project is used as follows:

- As the PROFINET device name
- As the name for the interface module
- As the name of the station
  A maximum of 24 characters is permitted for the name of the station in the TIA Portal. The name of the station is therefore truncated to a maximum of 24 characters where required. If truncating of the station name would result in several stations with the same name, the last characters of the station name are used to number the stations consecutively.

Names of modules and racks

In SIMATIC STEP 7 V5.5, the names of modules and racks do not have to be unique. In the TIA Portal, however, unique names are required. Therefore, the names of the modules and racks are incremented during the migration to make sure that unique names are assigned. This does not apply to PROFINET IO devices and PROFINET CPs. You must provide unique names for these yourself.

2.7.6.2 PROFINET IO configurations (S7-300, S7-400)

Unique name assignment required

The names of PROFINET IO devices must be unique within the subnet. The same applies to the names of PROFINET CPs. For this reason, ensure that the name assignment is unique before the migration into the initial project or provide unique names after migration and before the first compilation.

Media redundancy with sync domains

If a sync domain with activated media redundancy is found in the initial project, the sync domain configuration is imported during migration. If, however, several sync domains with activated media redundancy are found in the initial project, they are removed and the project is migrated without the sync domains. In any case, you will receive a message which sync domains were imported.

In any case, use SIMATIC STEP 7 V5.5 to configure the media redundancy in the initial project.

Isochronous mode with PROFINET IO

Note the following for PROFINET IO configurations in isochronous mode:

- IRT with high performance
  If you have configured IRT with high performance when planning the topology in the initial project, the option is changed to "RT" during migration. You will receive a message informing you of the change.
- Use of OB 61
  If the isochronous mode for PROFINET IO with OB 61 is activated in the original configuration in the STEP 7 project, it will be deactivated by the migration. The devices will still be imported. PROFINET IRT configurations with the option "high flexibility" without the use of OB 61 can be migrated.

CBA for IO controllers

If you have used the cyclical service CBA (Component Based Automation) in the initial project, it is deactivated during migration. You will receive a message informing you of the change.

Use of Shared Devices

If Shared Devices are configured in the initial project, they are removal during migration. You will receive a message informing you of their removal.

Transfer areas

If transfer areas for data exchange between the IO controller and I-device are configured in the initial project, they are removed during migration. You will receive a message informing you when a transfer area is removed during migration.

Update groups for PROFINET IO

If necessary, also remove update groups from the STEP 7 project before you start the migration. If any update groups are still configured, then the migration can not be carried out.

I slaves and I-devices

Only connected and consistent I slave and I-device configurations can be migrated. If the configuration is inconsistent, then the configurations in question will be excluded from the migration.

PROFINET IO systems that are not connected

Devices with PROFINET IO systems that are not connected can be migrated. The IO devices and the PROFINET IO systems are located in the network view without connection after migration. Connect these IO devices after migration with an IO controller or delete any unnecessary IO devices.

2.7.6.3 PROFIBUS configurations (S7-300, S7-400)

Isochronous PROFIBUS

It is possible to migrate isochronous PROFIBUS configurations. You should compile the project again after the migration, however, to rule out any possible inconsistencies. Also pay attention to slightly changed SDBs.

Orphaned master systems

Devices with orphaned DP master systems can be migrated. The DP slaves of the abandoned DP master system are located in the network view without connection after migration. Connect these DP slaves after migration with a DP master or delete any unnecessary DP slaves.

2.7.6.4 GSD and GSDML devices (S7-300, S7-400)

For devices that were configured with PROFIBUS GSD or PROFINET GSD files, the corresponding GSD files must also be installed in the TIA Portal. If GSD files are not yet installed in the TIA Portal, they will be installed automatically during the migration. The GSD files must comply with the specifications (e.g., GSDML Specification for PROFINET IO, see www.profinet.com (http://www.profinet.com)), in order for the project to be migrated successfully. If GSD files of a device manufacturer do not comply with the specification, the project cannot be migrated. In the migration log, you receive alarms notifying you of non-compliant GSD files.

If the GSD files do not comply with specifications, check whether a newer version of the GSD file is available from the manufacturer. Install the current version. If a version of the GSD file that complies with the specification is not available, contact the device manufacturer.

2.7.6.5 Connections (S7-300, S7-400)

Connections from SIMATIC STEP 7 projects can generally be migrated if they are also supported by the installed version of the TIA Portal.

Migratable types of connection

The following connection types can be migrated:

- S7 connection one-way
- S7 connection two-way
- Failsafe S7 connection
- TCP connection
- ISO-on-TCP connection
- ISO connection
- UDP connection
- E-mail connection

- FDL connection
- Point-to-point connection

FMS connections on the PROFIBUS are not accepted during the migration. Adapt the connections or modify the user program.

2.7.6.6 Multiprojects (S7-300, S7-400)

Multiprojects in SIMATIC STEP 7

In SIMATIC STEP 7, you can organize projects into a multiproject if, for example, a project is very large or if several people are working on it. The subprojects may contain cross-project references, for example connections. All subprojects of a multiproject and the associated libraries are stored in the same directory.

Migration of parts of a multiproject

You can migrate the subprojects of a multiproject. To do this, select one of the subprojects during the migration. All of the devices and configurations contained in the respective subproject will be imported during the migration.

Observe the following when migrating a subproject:

- Cross-project connections
  Cross-project connections between subprojects are created as unspecified connections.
- Cross-project networking
  Information on networking of devices across project boundaries will be lost during the migration. Links between devices in the same subproject will be retained however.

Grouping a multiproject for the migration.

It is only possible to migrate an entire multiproject with all of its associated subprojects if you manually regroup the different subprojects prior to the migration. To regroup a multiproject, copy all devices, for example, and paste the devices into a single project.

2.7.6.7 Migrating program blocks (S7-300, S7-400)

You can generally migrate all blocks created in LAD, FBD, SCL, STL, and GRAPH. Please note the following:

Programs with absolute addressing

In the TIA Portal, symbolic operands are automatically declared for all absolute addresses. If absolute addressing is used in the STEP 7 program to be migrated, symbolic operands are declared for the absolute addresses during the migration. This automatic declaration can lead to data type conflicts, particularly if the IEC check is activated in the TIA Portal. A data type conflict could occur, for example, when a 32-bit data word is interpreted as a tag with the type DWORD by the automatic declaration, but the program expects a tag with the data type REAL.

In this case, it is not enough to deactivate the IEC check. Instead, you must correct the declaration in the PLC tag table.

Data type conflicts can also occur if you access the same absolute address multiple times in the program, but use different data types for the different access operations. The following figure shows an example of such multiple access:

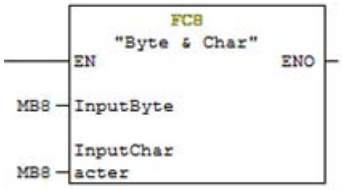

In these cases, a separate tag is declared automatically for each data type during the migration. Since all tags reference the same absolute address, warnings are output in the PLC tag table. You need to manually correct these address overlaps in the PLC tag table.

Migration of communication blocks

During the migration of communication instructions from STEP 7 V5.x, pointers of the data type ANY are replaced by PLC tags within the own S7 station.

- After the migration a warning is displayed at the affected parameters.
- Replace the PLC tags with a pointer at the affected parameters.
  For absolute addressing of a data block with data type ANY, the data block must have been created with standard access, since symbolic addressing with data type ANY is not possible. Example: To input values, use the appropriate parameters of instruction P#M10.0 BYTE 4 instead of %MD10.

Symbolic names for blocks and user-defined data types (UDT)

In the TIA Portal, each block or UDT has a number and a name. It is therefore no longer necessary to declare symbolic names. If the migrated program contains symbol declarations for blocks or UDTs, they will be used as names after the migration.

Blocks from STEP 7 libraries

In the TIA Portal, standard functions and function blocks are not provided in libraries. Instead, they are available as instructions in the "Instructions" task card. The instructions are sorted by function and can be found under their symbolic name.

If calls of standard functions or function blocks are contained in the migrated program, they are replaced during the migration by the instructions that correspond to the original standard function or the original standard function block.

If the library block is no longer supported in the TIA Portal, you have the following options:

- The library block is replaced by a compatible instruction. In this case, you will receive an alarm informing you that you have to compile the program after migration. Note that the calling block may not carry any know-how protection, as the block call has to be adapted during compilation.

- If there is no compatible instruction, the block is migrated as user block. It is then available as a know-how protected block in the folder "Program blocks" with the extension "_LF" (legacy function).

User-defined block libraries

User-defined block libraries are no longer available in the familiar form in the TIA Portal. However, you can migrate user-defined libraries by integrating these into a project prior to migration and then migrating the project. After this, you can copy these from this project in the TIA Portal and use them.

Instructions in a modified display

The display of some of the LAD, FBD, STL, SCL and GRAPH commands in the TIA Portal differs from the display in earlier versions of STEP 7. So, for example, the mathematical functions and the comparators are no longer specific for data types. Instead, there is a central instruction, which can be used for all types of data. The "ADD_I" command is no longer permitted, for example; the instruction "ADD" is used instead.

Some other commands also have a new display in the TIA Portal, e.g., edge commands, word logic operations, conversions, IEC timers, IEC counters, etc.

The commands will be converted for the new display during the migration.

Stricter IEC check

Stricter rules for checking data type compatibility will be used in the TIA Portal. In addition, errors that can lead to runtime errors are already detected during the syntax check. It is no longer permitted to write your own input parameters or to read your own output parameters in functions (FC). Observe the following rules to avoid syntax errors:

- Only use tags with compatible data types when transferring the parameters.
- In comparison instructions or arithmetic instructions, only use tags with compatible data types.
- You may not write to input parameters or read from output parameters.

For more information on the block interface and on data type conversion with IEC check, refer to the Help on data types:

Data types

Stricter data type check with the "MOVE" and "FILL" instructions

With the "MOVE" and "FILL" instructions, the compatibility of the source and destination areas is checked according to stricter rules. It is no longer possible, for example, to move a tag of the data type "Integer" to a destination area of the data type "WORD" using "MOVE". Instead of this, use the "CONVERT", "T_CONV" or "S_CONV" instructions in the TIA Portal.

You will find more detailed information on the data types compatible with "MOVE" and "FILL" in the help on the instructions:

AUTOHOTSPOT

AUTOHOTSPOT

Programs that were created in LAD or in FBD and use invalid data types at the "MOVE" or "FILL" instruction are shown in STL after the migration. However, the language is still set as LAD or FBD in the block properties.

In the block properties, change the program again to STL. Then reset the language to LAD or FBD.

Suffix for I/O access ":P"

In the TIA Portal, the access ID ":P" is used for direct addressing of the I/O. The following notation is not permitted, for example:

```
%PIW3
```

The following notation is used instead:

```
%IW3:P //absolute display
MyTag:P //symbolic display
```

The accesses are converted into the new notation during the migration. However, symbolic names, which were declared to be I/O tags in the original program, cannot be accepted. New declarations will be created instead. For more information on the I/O access refer to the Help on addressing the I/O:

Address I/O

No distinction is made between upper and lower case for jump labels

No distinction is made between upper and lower case for jump labels in the TIA Portal. If the source program contains jump labels that differ only through upper/lower case, these will be converted into unique jump labels. The log file provides information on the modified jump labels.

German special character "ß" is not distinguished from "ss"

In tag names, the German special character "ß" is not distinguished from "ss" in the TIA Portal. If, for example, the source program contains tag names that differ only in these two characters, they will be converted into unique tag names.

Migration of REAL and STRING values

In the TIA Portal it is checked whether the start values of REAL and STRING tags are in the permitted and/or declared range. The following rules apply:

- REAL tags must have a value in the range from +1.175494E-38 to +3.402823E+38.
- The maximum length of the STRING value must not exceed the declared length of the STRING.

If these rules are not satisfied, the program will be successfully migrated but it will not then be possible to compile it.

As remedy, you can open the corresponding data block in STEP 7 (TIA Portal) and change the start values. Then compile the program with the menu command "Software (rebuild all blocks)".

Migration of global data blocks

The display and handling of data values in data blocks is different in the TIA Portal compared with SIMATIC STEP 7. In global data blocks which are not derived from a higher-level object, for example a UDT, the tags are always assigned the default value of the data type as the default value, for example FALSE for the data type BOOL. This default value is not editable. If offline initial values were assigned in the declaration view, they are not imported during the migration. If you require user-specific default values in the program, use a global data block based on a PLC data type. You can assign user-specific default values for tag values in a PLC data type.

For more information on data values in data blocks refer to the Help on data blocks:

Programming data blocks

Tag tables

The tag tables from the original program are not migrated to the TIA Portal. To monitor tags online in the TIA Portal, you create so-called "watch tables". These have the same function as the tag tables in SIMATIC STEP7.

Note

Additional support for migration

You can find the latest information about migration in the Siemens Industry Online Support:

Migration of controllers (http://support.automation.siemens.com/WW/view/en/83557459)

If you need further support, please contact SIMATIC Customer Support.

"Ret_val" as tag name in the Block interface

If the keyword "Ret_val" is used as tag name in the reference program in a block interface, the block is not migrated correctly. In this case, correct the reference program and re-migrate it.

2.7.6.8 Migration of LAD/FBD/STL programs (S7-300, S7-400)

LAD, FBD and STL blocks can run after the migration without being recompiled or reloaded.

STL sources

STL sources cannot be migrated to the TIA Portal. Create STL blocks from your sources prior to migration. You can then migrate these once again as sources to the TIA Portal. To do so, use the "Generate source from blocks" shortcut menu command in the project tree.

See also: AUTOHOTSPOT

Know-how protection

Blocks that were know-how protected before the migration remain know-how protected after the migration. Because the sources are not migrated, the know-how protection cannot be canceled. The blocks can thus not be opened or edited. It is possible, however, to load the blocks into the CPU and to run them.

The following options are available for disabling a block's know-how protection:

- Prior to the migration, remove the KNOW_HOW_PROTECT attribute from the source and create a block without know-how protection from this. Then migrate this block.
- Import the source into the TIA Portal and create a block without know-how protection from this. Then use the "Edit > Know-how protection" menu command to provide the block with a password.

Switch-over from LAD/FBD to STL

More precise syntax checks are performed in some areas in the TIA Portal than in older versions of STEP 7. However, the migration ensures where possible that your program can still be compiled: If a LAD/FBD block contains operations that would result in a syntax error in the TIA Portal, the affected networks are shown in STL after the migration. The semantics of the source program remain unchanged.

A message in the migration log informs you of the conversion of the programming language of individual networks.

2.7.6.9 Migrating GRAPH programs (S7-300, S7-400)

Requirement

The prerequisite for migrating GRAPH blocks is to have the "S7-GRAPH" optional package, version 5.3 SP6 or later, installed on the original device.

GRAPH-specific block settings

The GRAPH-specific block settings have been significantly reduced in the TIA Portal. The following table gives an overview of how the original block settings are transferred to the TIA Portal.

Setting in the initial project		Setting in the TIA Portal
FB parameter	Minimum	Existing parameters will be migrated
	Standard	
	Maximum	
	Definable	
Interface description	Minimized memory space	Create minimized DB
	Structure fields	Individual structures are always created
	Individual structures	

Setting in the initial project		Setting in the TIA Portal
Run capability	FC70/71	FC 73 if the option "Create minimized DB" is active; otherwise FC 72
	FC72	
	FC73	
Criteria analysis in the DB		Setting not relevant in the TIA Portal
Skip steps		Skip steps
Acknowledge errors		Acknowledge supervision errors
Synchronization		Always activated in the TIA Portal
Permanent processing of all interlocks in manual operation		Permanent processing of all interlocks in manual operation
Lock operating mode selection		Lock operating mode selection
Safe activation mode		Always activated in the TIA Portal
Warnings	None	Always activated in the TIA Portal
	Any	
Activate alarms		Activate alarms
Alarms with WR_USMSG		Alarms in the TIA Portal always with ALARM_SQ/ALARM_S

Step name extension

Step name extensions are not supported in the TIA Portal. They are converted to step-specific text during the migration.

Compiling the migrated block

GRAPH blocks must be recompiled and reloaded after the migration.

The following errors may occur when compiling:

- Data type conflicts may occur due to the stricter IEC check. In this case, change the declaration.
- Since the "structure fields" setting for the interface is no longer available in the TIA Portal, interface elements can no longer be addressed with G7S[1].X, for example. Access to the element must therefore be corrected, e.g. to STEP1.X.

Changing the interface of the GRAPH block

The changed settings can lead to interface changes of the block. Therefore, it may be necessary to regenerate the instance DB or to update block calls.

2.7.6.10 Migrating SCL programs (S7-300, S7-400)

The prerequisite for migrating SCL blocks is to have the "S7-SCL" option package, version 5.3 SP5 or later, installed on the original device.

SCL blocks must be recompiled and reloaded after the migration.

Basic procedure for the migration

A complete migration of SCL blocks is only carried out if the associated sources exist in the initial project.

The table below gives you an overview of the basic procedure:

Existing in the initial project	Existing after migration
SCL blocks with source	Editable SCL blocks
Know-how protected SCL blocks with source	Editable SCL blocks. After the migration, the blocks no longer have know-how protection but can be re-protected, if necessary.
SCL blocks without source	Know-how protected SCL blocks
Know-how protected SCL blocks without source	Know-how protected SCL blocks

SCL-specific compiler options

Only the compiler options that are defined directly in an SCL source are adopted as block properties in the TIA Portal during migration. If no compiler options are defined in the original SCL source, the options in the properties of the migrated block are deactivated.

Compiler options defined as settings in the SCL Editor or in an SCL compile control file are not migrated.

The following table gives an overview of the compiler options that are transferred as block property to the TIA Portal.

Compiler option in the SCL source	Block property in the TIA Portal
scl_monitorarraylimits	Check ARRAY limits
scl_createdebuginfo	Create extended status information
scl_setokflag	Set ENO automatically

Calling function blocks

A block call with indication of the function block, for example "DBX"."FBX()", is no longer permitted in the TIA Portal. The call syntax is converted to the notation "DBX()" during the migration.

New EN/ENO mechanism

SCL uses the EN/ENO mechanism of the TIA Portal. For this reason, all usages of the OK flag are replaced by ENO during the migration. The points where ENO was used in the original program are highlighted. You must check these points after the migration and adapt them to the new mechanism.

See also:

AUTOHOTSPOT

Operator "DIV"

The "DIV" operator is no longer available in the TIA Portal. All usages of "DIV" will be converted to the standard notation "/" during the migration.

Local constants

Local, symbolic constants are declared in the "CONST" section of the block interface in the TIA Portal. They always have a data type. If symbolic constants are declared without data type in the source program, an appropriate data type will be assigned to them during the migration.

In classic SCL programs it was possible to use local constants as ARRAY limits. These declarations are retained during migration.

Example for the declaration of ARRAY limits with local constants in a classic SCL program:

```
VAR_INPUT
MyArray : Array[MyConst1 .. MyConst2] of Int;
END_VAR
```

This declaration looks as follows after the migration:

1	Input	
2	MyArray	Array[#MyConst1..#MyConst2] of Int

However, it is not possible to form expressions with local constants in the block interface of the TIA Portal and use them for the declaration of ARRAY limits or initialization of values. Instead the event of the expression is formed during migration and entered as value in the declaration. The expression from the source program is stored as comment.

Example for the declaration of ARRAY limits with the help of expressions in a classic SCL program:

```
CONST
MyConst1Value10 := int#10;
MyConst2Value10 := int#10;
MyConst20 := MyConst1Value10 + MyConst2Value10 ;
MyConst0 := MyConst1Value10 - MyConst2Value10 ;
MyConst100 := MyConst1Value10 * MyConst2Value10 ;
END_CONST
```

This declaration looks as follows after the migration:

Constant			
MyConst1Value10	Int	10	
MyConst2Value10	Int	10	
MyConst20	Int	20	MigChange: MyConst1Value10 + MyConst2Value10
MyConst0	Int	0	MigChange: MyConst1Value10 - MyConst2Value10
MyConst100	Int	100	MigChange: MyConst1Value10 * MyConst2Value10

See also:

AUTOHOTSPOT

AUTOHOTSPOT

Nested ARRAYs

It is not possible to nest ARRAYs in the TIA Portal. The following declaration is not permitted, for example:

```
ARRAY[1..5] OF ARRAY[0..3] OF INT
```

Nested ARRAYs are converted into multi-dimensional ARRAYs during migration. The example would look as follows after the migration:

```
ARRAY[1..5, 0..3] OF INT
```

Declaration of jump labels (LABEL)

The declaration of jump labels is not possible in the TIA Portal. The following declaration would not be accepted from the initial project, for example:

```
LABEL
MARKE1, MARKE2, MARKE3 ;
END_LABEL
```

Set jump labels, however, will be retained in the program code and can be used for GOTO operations.

Indexed I/O access

In the TIA Portal, the syntax with rounded parentheses is used for indexed I/O access. The following notation is not permitted, for example:

```
PEB[1]
```

The following notation is used instead:

```
EB(1):P
```

The accesses are converted into the new notation during the migration.

See also:

AUTOHOTSPOT

Indexed memory accesses

In the TIA Portal, the syntax with rounded parentheses is used for indexed memory accesses. The following notation is not permitted, for example:

```
EB[2]
```

The following notation is used instead:

```
EB(2)
```

The accesses are converted into the new notation during the migration. For additional information on addressing, refer to "See also".

Indexed access to data blocks

In the TIA Portal, the syntax with rounded parentheses is used for indexed data block accesses. The following notation is not permitted, for example:

```
%DB100.DW[5]
```

The following notation is used instead:
`%DB100.DW(5)`

The accesses are converted into the new notation during the migration. For additional information on addressing, refer to "See also".

Absolute access to data blocks

For absolute access, the absolute designator of the data block must be used. Access using the symbolic designator is not permitted in the TIA Portal.

The following notation is not permitted, for example:
`DB100.DW3`

The following notation is used instead:
`%DB100.DW3`

During the migration, the marker "%" will be added for a detected absolute data block access. For additional information on addressing, refer to "See also".

Logarithmic functions "EXPD" and "LOG"

Logarithmic functions "EXPD" and "LOG" are no longer available in the TIA Portal. All usages of "EXPD" are converted to the standard notation "10**(<expression>)" during the migration. All usages of "LOG" are converted to the standard notation "LN(<expression>)" during the migration.

Mathematical instruction "MAX"

The "MAX" instruction cannot process any time data types in the TIA Portal. However, you can use comparators instead.

The following example shows how you can find the "larger" of two time data types using comparators:

SCL

```
#xs_tod := #xi_tod + #xi_time;
   // statt #xstod1 := MAX(IN1:=#xtod1, IN2:=#xstod1);
   IF #xs_tod < #xi_tod THEN
         #xs_tod := #xi_tod;
   END_IF;

   // statt #xsd1 := MAX(IN1:=#xd1, IN2:=#xsd1);
   IF #xi_date <#xs_date THEN
         #xs_date := #xi_date;
   END_IF;
#xs_date_time := #xi_date_time + #xi_time;
   // statt #xs_date_time := MAX(IN1:=#xi_date_time, IN2:=#xs_date_time);
   IF #xi_date_time <#xs_date_time THEN
          #xs_date_time := #xi_date_time;
   END_IF;
```

SCL

Floating point numbers - exponential display

The exponential notation of floating point numbers without a decimal point is no longer permitted in the TIA Portal. The following notation is not permitted, for example:
`3E10`

The following notation is used instead:
`3.0E10`

During the migration, all usages of "3E10" will be converted to the standard notation "3.0E10".

See also:

AUTOHOTSPOT

Blocks from STEP 7 libraries

The functions FC 1 to FC 40 from the STEP 7 classic library "IEC standard functions" are no longer available in the TIA Portal. Calls to the standard functions are converted to calls to the corresponding extended instructions during the migration. If there is no clear conversion possibility, a system error is issued.

In this case, change the SCL block after the migration and use SCL instructions or operators instead of IEC standard functions.

Examples:

Use S5TIME_TO_TIME instead of S5TI_TIM (FC 33)
```
"TAG_TIME" := S5TIME_TO_TIME ("TAG_S5TIME");
```

Use the comparator operator <> instead of NE_DT (FC 9):
```
IF #D1 <> #D2 THEN "MyOutput" : = 10;
END_IF;
```

Use the comparator operator = instead of EQ_STRING (FC 10):
```
IF #My_String1 = #MyString2 THEN "MyOutput" : = 10;
END_IF;
```

Interruption of strings with "$"

In STEP 7 Classic, strings in SCL could be interrupted by special characters, e.g., control characters or non-printable characters. These characters were identified by the interruption sequences "$>" and "$<".

Example:
```
#myString := 'Test1$>     $<Test1';
```

This option no longer exists in the TIA Portal. String interruptions will be deleted during the migration.

NULL pointer

Null pointers are no longer specified in the TIA Portal with "NIL" but instead with "NULL". The notation is replaced automatically during the migration.

Comments

Comments from the migrated source are transferred when possible. The following rules apply:

- Comments in the output block directly after the block definition are converted as block comments.
- Comments that appear as line comments after a tag declaration are transferred to the "Comment" column in the block interface.
- All other comments are not transferred.

Example:

```
// Comment that is not transferred.
FUNCTION_BLOCK FB57
// Comment that is transferred to the block properties.
(*Comment that is transferred to the block properties. *)
VAR_TEMP
    myVar1 : INT;// Comment that is transferred to the interface.
    myVar2 : INT;(* Comment that is not transferred because it is a
comment section *)
END_VAR
// Comment that is not transferred.
BEGIN
    ;
END_FUNCTION_BLOCK
```

Block calls in S7-SCL sources

In the S7-SCL sources with STEP 7 V5.x, you can set the block parameters before or after the actual call of the block. This means that the block can be called with incomplete parameters and the input/output parameters of the block can be suppled with tags or values at a different position with the S7-SCL source. This type of programming is not possible in STEP 7 (TIA Portal) due to the system and will also not be supported in future. Although you can migrate and compile the STEP 7 V5.x project without errors in TIA portal, the function of this block call is no longer possible within the S7-SCL program.

Before you compiled your S7-SCL source in STEP 7 V5.x and migrate the project to the TIA Portal, note the following points:

- All parameters of the called block must be supplied within the call instruction.
- No parameters of the called block may be deleted and inserted or set at a different position of the S7-SCL source.

2.7.6.11 Messages (S7-300, S7-400)

PLC alarms are migrated with all parameters and including all alarm classes. Alarm types and alarm instances are also migrated and stored under the PLC alarms in the TIA Portal.

Requirement

A requirement is that the CPU-wide assignment of alarm numbers is selected in the initial project. Before the migration, select the "Always assign CPU-oriented unique message numbers" option in the initial project, if necessary, and perform "Save as > With reorganization".

Restrictions

- Symbol-based alarms cannot be migrated.
- The keywords "$$AKZ$$", "$$BlockComment$$", and "$$Area$$" are transferred but no replacement takes place during runtime.

2.7.6.12 Reporting system errors (S7-300, S7-400)

In principle, you can migrate system error messages that you have configured using SIMATIC STEP 7.

Restrictions

When migrating the "Report system errors" application, only the settings that were set by the user in the SIMATIC STEP 7 project will be migrated (e.g. basic settings, block numbers, block names).

The following elements are not migrated:

- SFM blocks
- SFM messages and the settings that were made to configure the message texts
- System text libraries

After migration, the settings can be manually adjusted once again. The missing elements will be recreated during the first compilation of the hardware configuration.

2.7.6.13 Migration of project texts (S7-300, S7-400)

When you migrate an initial project, project texts such as block titles and block comments are adopted in the TIA Portal. However, only project texts in the set editing language of the initial project are migrated. Project texts in other languages are not migrated.

See also

Migration of STEP 7 projects (Page 19)

2.8 Migrating WinCC projects

2.8.1 Migrating WinCC flexible 2008 projects

2.8.1.1 Basics (WinCC flexible)

Migration (WinCC flexible)

Introduction

You can continue to use projects in WinCC from WinCC flexible. The following versions of WinCC flexible are supported:

- WinCC flexible 2008 SP2
- WinCC flexible 2008 SP3
- WinCC flexible 2008 SP5

The following sections describe the HMI devices that are supported and the required basic conditions for a successful migration.

Projects from ProTool Pro and earlier WinCC flexible versions cannot be migrated directly to WinCC. If you want to continue using such projects in WinCC, you have to migrate these to a supported version of WinCC flexible first and change the HMI device type.

If the project to be migrated contains components of a supported option package, the option package must be installed for successful migration to WinCC. If the option package is not installed, migration is canceled. This concerns the following option packages:

- SINUMERIK

The following option packages are not supported by the migration:

- ProAgent
- Open Platform Program - OPP

See also

Basics on migration (WinCC flexible) (Page 42)

Migrating projects from WinCC flexible (WinCC flexible) (Page 44)

Compiling and loading a migrated project (WinCC flexible) (Page 46)

Supported HMI devices (WinCC flexible) (Page 47)

Object support in the migration (WinCC flexible) (Page 52)

Migration of connections (WinCC flexible) (Page 55)

Migration of tags (WinCC flexible) (Page 59)

Migrating runtime data (WinCC flexible) (Page 68)

Migrating integrated projects (WinCC flexible) (Page 71)

Migrating data types (WinCC flexible) (Page 75)

Basics on migration (WinCC flexible)

Introduction

In the migration the project data are converted from a WinCC flexible project into the new data format of WinCC. The data will not be evaluated to see if they are consistent in the project you want to migrate. If errors or warnings are output in a source project during compilation, these will not be resolved as part of the migration. This means you should be able to compile the project without errors prior to migration. Note the scope of a project during migration. The features of WinCC apply for migration. For additional information refer to the online help in section "Process visualization > Performance features > Engineering System".

Unique object names

The objects are clearly identified by the folders in which they are contained in WinCC flexible. Screen objects in groups are clearly identified by the group name.

In WinCC, an object name must be unique within an HMI device. The name of screen objects must be unique within a screen.

The uniqueness of the name is verified during migration. If a name is not unique according to the new rule, the object in question will be renamed. A renamed object will receive the suffix "#Mign", where "n" stands for a sequential number.

Note

Changed OPC DA server name

If you have configured an OPC-DA server in the WinCC flexible project, the OPC-DA server is no longer available under the old name after migration.

In the migrated project, change the name of the OPC-DA server to the following value for the OPC clients concerned: "OPC.SimaticHMI.CoRtHmiRTm".

Points to note when renaming tags

If you have structured tags in folders in WinCC flexible, the name of the tag will be formed during migration from the folder name and the tag name. The names of the folders and tags are separated by the character \. The name of the tag after migration is then, for example: Plant1\Line3\Tag17.

If the name would otherwise be longer than 128 characters after migration, the name is formed by the character string #mig, a consecutive number, the character # and the tag name from WinCC flexible, for example #mig2#Tag17.

If you dynamically compose tag names in scripts, you must check the tags whose names were changed during migration with "#mig".

Affected objects

The following objects are renamed if necessary:

- Screens
- Faceplates
- Screen objects
- Graphics
- Recipes
- Structures
- Structural elements
- Alarm logs
- Tags
- Data logs
- Connections

Canceling migration

The migration is canceled in the following cases:

- If the project to be migrated is opened in the engineering system or in Runtime.
- If not enough memory space is available on the hard disk to create a copy for migration of the project.
- If migration cannot address the project database due to problems with the installed SQL server.
- If migration cannot address the project database due to missing user authorization.
- If you select the "*.hmi" file for the migration in an integrated project. You must select the "*.s7" file for the migration in an integrated project.
- If the project was created with a version not supported by the migration.
- If devices that are not supported in the project are configured.

Saving the project in the migration format

You do not have to execute the migration of a WinCC flexible project completely on the PC on which the project is available. You can prepare the migration by saving the project in the migration format. The migration tool is available for saving a WinCC flexible project in the migration format. The migration tool exports the engineering data from the WinCC flexible project and saves the data in the migration format "*.am".

For the actual migration, copy the data in the migration format to a PC on which the TIA Portal is installed.

For more detailed information, please refer to the migration tool documentation.

See also

Migration (WinCC flexible) (Page 41)

HMI device change by the migration (WinCC flexible) (Page 49)

Migrating projects from WinCC flexible (WinCC flexible)

Introduction

When you migrate a project, data from a WinCC flexible project is loaded into a new project for WinCC. A new project is therefore created automatically for project migration. You cannot migrate to an existing project.

The migration can be started in both the Portal view and the Project view.

You should only migrate a project to a re-started TIA Portal.

Information on the migration of an integrated project can be found in the section Migrating integrated projects (WinCC flexible) (Page 71).

If you only want to save the project in migration format, you can use the migration tool. See Basics on migration (WinCC flexible) (Page 42) for additional information.

Note

Further support for migration

You find current information on migration in the Siemens Industry Online Support:

Migration of visualization (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=100&lang=en&referer=%2fWW%2f&func=cslib.cssearch&nodeid0=4000024&viewreg=WW&siteid=csius&extranet=standard&groupid=4000002&objaction=cssearch&content=adsearch%2Fadsearch%2Easpx)

If you need further assistance, please contact the SIMATIC Customer Support.

Requirement

- A project from WinCC flexible is available.
- The project is not open in WinCC flexible.

Procedure

Migrate a project in the Portal view as follows:

1. Select the action "Start > Migrate Project".

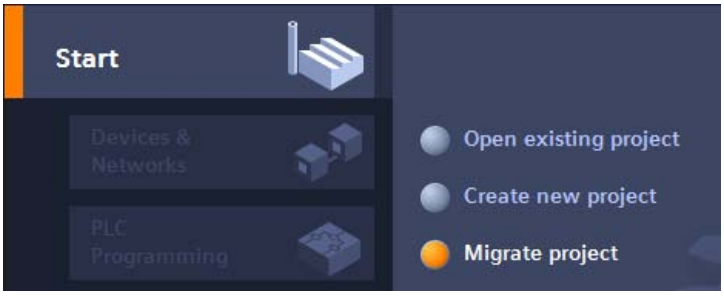

2. In the "Source path" box, navigate to the project you want to migrate.
3. Select the WinCC flexible project file "*.hmi" or "*.am".
4. Change the information for the project to be created, if necessary. For example, change the project name or project path. The data to be migrated is created in the new project.
5. Click "Migrate".
   A new project is created and migration of the data is started:
   - The Project view opens.
   - The progress of the migration is shown in a migration window.
   - Warnings and errors about the migration process are displayed in the Inspector window under "Info > General".
   - All information about the migration is saved in a log file.
   - The project is saved and a message displayed upon completion of the migration. The message contains a link that you can use to open the log file.

When migration is complete, you will find a newly created device for each migrated HMI device in the project tree. These devices contain the migrated data, such as screens, alarms and tags.

Opening the migration log at a later point in time

The migration log is saved together with the migrated project. You can view the log at a later point in time. Open the log file as follows:

1. Open the "Common data > Logs" folder in the project tree. It contains the logs of all previously performed migrations.
2. Double-click the required migration log.
   The log is opened.

See also

Migration (WinCC flexible) (Page 41)

Migrating integrated projects (WinCC flexible) (Page 71)

Basics on migration (WinCC flexible) (Page 42)

Compiling and loading a migrated project (WinCC flexible)

Compiling a migrated project

Once you have successfully migrated a WinCC flexible project, you need to recompile it before loading it to the HMI device. The project will only compile successfully if it was capable of error-free compiling prior to migration.

If errors occur during compilation of the migrated project, they have to be eliminated.

Once compiling is successfully completed, load the project to the HMI device.

Settings for download to the HMI device

The settings for loading the HMI device are not included in the migration. Once you have migrated the project, you must configure the settings for loading.

Select the HMI device in the project tree and select "Loading in device > Software (complete loading)" from the shortcut menu. The "Extended download to device" dialog opens. Configure the required settings for the interface. Click the "Load" button. The project is recompiled and the dialog "Load preview" is opened.

Expand the "Overwrite" entry and verify the settings for the following options:

- Would you like to overwrite the existing user administration data from this device
- Would you like to overwrite the existing recipe data on HMI system

Configure the options as you want to use them in the project in the future. Subsequently, load the project to the HMI device.

See also

Migration (WinCC flexible) (Page 41)

2.8.1.2 Migrating engineering data (WinCC flexible)

HMI devices (WinCC flexible)

Supported HMI devices (WinCC flexible)

Introduction

When migrating projects from WinCC flexible you must bear in mind that WinCC does not support all HMI devices. You have to distinguish between the following cases:

- HMI device is supported by WinCC.
  The project is migrated 1:1 and gets the same HMI device after migration as before migration.
- The HMI device is replaced by a compatible successor model.
  The project is migrated. The migration replaces the HMI device with a compatible successor model. See HMI device change by the migration (WinCC flexible) (Page 49) for additional information.
- HMI device is not supported.
  If your WinCC flexible project contains an HMI device that is not supported by WinCC, the migration process is cancelled. To migrate the project, you must change the HMI device in WinCC flexible to a HMI device type supported by WinCC.
  If the unsupported HMI device is included in the library, the migration is performed. In the migrated project in the TIA Portal, corresponding master copies for the unsupported devices are created in the project library.

The following HMI device types are supported both by WinCC flexible and WinCC:

Basic Panels

- KTP400 Basic mono PN
- KTP400 Basic mono PN Portrait
- KTP600 Basic DP
- KTP600 Basic DP Portrait
- KTP600 Basic PN
- KTP600 Basic PN Portrait
- KTP600 Basic mono PN
- KTP600 Basic mono PN Portrait
- KTP1000 Basic DP
- KTP1000 Basic PN
- TP1500 Basic PN

Mobile Panels

- Mobile Panel 177 6" DP
- Mobile Panel 177 6" PN
- Mobile Panel 277 8"
- Mobile Panel 277 8" IWLAN V2
- Mobile Panel 277F 8" IWLAN V2
- Mobile Panel 277F 8" IWLAN (RFID Tag)
- Mobile Panel 277 10"

SINUMERIK PC

- OP 010 Key
- OP 012 Key
- OP 015 Key
- OP 015A Key
- TP 015A Touch+Key

HMI applications

- WinCC flexible Runtime

WinCC only supports the functions provided by these HMI device types.

Other functions which are not migrated because of the restricted device selection are documented in the following sections.

Adaptations before migration

If the HMI device was changed to an HMI device with a different screen size in the project to be migrated, you must recompile and save the project in WinCC flexible before the migration. The compilation process will adjust the size of the screens and screen elements.

See also

Migration (WinCC flexible) (Page 41)

HMI device change by the migration (WinCC flexible) (Page 49)

Configuration change after HMI device change (WinCC flexible) (Page 51)

Object support in the migration (WinCC flexible) (Page 52)

Migration of connections (WinCC flexible) (Page 55)

HMI device change by the migration (WinCC flexible)

Introduction

WinCC flexible supports some HMI devices which are discontinued in the future. These HMI devices are no longer supported by WinCC. When migrating a WinCC flexible project, an HMI device that is not supported is replaced by a compatible successor device.

Only the HMI device type is changed during the migration. HMI device-specific data are not changed by the migration.

Inconsistencies in the project may occur due to a change in the HMI device. In a project which was compilable before the migration, errors may occur in the project compilation after changing the HMI device. This may occur, for example, because the changed HMI device supports different memory media to the previous HMI device.

HMI device change by the migration

The following table shows which successors replace the HMI devices that are not supported.

Unsupported HMI devices	Successors
HT 8	PCU50.3 -C (resolution: 640*480)
Mobile Panel 170 6"	Mobile Panel 177 6" DP
Mobile Panel 277 8" IWLAN	Mobile Panel 277 8" IWLAN V2
Mobile Panel 277F 8" IWLAN	Mobile Panel 277F 8" IWLAN V2
OP010 Key	PCU50.3B -C (resolution: 640*480)
OP012 Key	PCU50.3B -C (resolution: 800*600)
OP015 Key	PCU50.3B -C (resolution: 1024*768)
OP 08T	PCU50.3 -C (resolution: 640*480)
TP015A Touch Key	PCU50.3B (resolution: 1024*768)
PC 477 12" Key	PC 477B 12" Key PB
PC 477 12" Touch	PC 477B 12" Touch PB
PC 477 15" Key	PC 477B 15" Key PB
PC 477 15" Touch	PC 477B 15" Touch PB
PC 477 19" Touch	PC 477B 19" Touch PB
PC 577 12" Key	IPC 577C 12" Key PB
PC 577 12" Touch	IPC 577C 12" Touch PB
PC 577 15" Key	IPC 577C 15" Key PB
PC 577 15" Touch	IPC 577C 15" Touch PB
PC 577 19" Touch	IPC 577C 19" Touch PB
PC 670 10" Key	HMI IPC677C 12" Key PB
PC 670 12" Key	PC 677B 12" Key PB
PC 670 12" Touch	PC 677B 12" Touch PB
PC 670 15" Key	PC 677B 15" Key PB
PC 670 15" Touch	PC 677B 15" Touch PB
PC 677 12" Key	PC 677B 12" Key PB
PC 677 12" Touch	PC 677B 12" Touch PB

Unsupported HMI devices	Successors
PC 677 15" Key	PC 677B 15" Key PB
PC 677 15" Touch	PC 677B 15" Touch PB
PC 677 17" Touch	PC 677B 17" Touch PB
PC 677 19" Touch	PC 677B 19" Touch PB
PC 870 12" Key	HMI IPC677C 12" Key PB
PC 870 15" Key	HMI IPC677C 15" Key PB
PC 870 15" Touch	HMI IPC677C 15" Touch PB
PC 877 12" Key	HMI IPC677C 12" Key PB
PC 877 15" Key	HMI IPC677C 15" Key PB
PC 877 15" Touch	HMI IPC677C 15" Touch PB
PC 877 19" Touch	HMI IPC677C 19" Touch PB
PC IL 70 12" Touch	HMI IPC577C 12" Touch PB
PC IL 70 15" Touch	HMI IPC577C 15" Touch PB
PC IL 77 12" Touch	HMI IPC577C 12" Touch PB
PC IL 77 15" Touch	HMI IPC577C 15" Touch PB
PC IL 77 19" Touch	HMI IPC577C 19" Touch PB
PC IL 77 12" Key	HMI IPC577C 12" Key PB
PC IL 77 15" Key	HMI IPC577C 15" Key PB
P012T Touch	IPC 677C 12" Touch PB (if a P350 is connected)
P012T Touch	IPC 477C 12" Touch PB (if a P320 is connected)

The following devices are not supported and are therefore not migrated:

- OP 73 and OP 73 micro
- TP 170 micro 6"
- TP 177 micro 6" and TP 177 micro 6" Portrait
- C7-635 6" Key and C7-635 6" Touch
- C7-636 6" Key
- C7-636 10" Touch
- OP 77A and OP 77B
- TP 170A 6"
- TP 177A 6" and TP 177A 6" Portrait
- TP 170B 6" mono and TP 170B 6" color
- TP 177B 6" mono DP and TP 177B 6" color PN/DP
- OP 170B 6" mono
- OP 177B 6" mono DP and OP 177B 6" color PN/DP
- TP 177B 4" color PN/DP
- TP 270 6"
- TP 270 10"

- OP 270 6"
- OP 277 6"
- OP 270 10"
- TP 277 6"
- MP 177 6" Touch
- MP 270 6" Touch
- MP 270 10" Key and MP 270 10" Touch
- MP 277 8" Key V1 and V2, MP 277 8" Touch V1 and V2
- MP 277 10" Key V1 and V2, MP 277 10" Touch V1 and V2
- MP 370 12" Key and MP 370 12" Touch
- MP 370 15" Touch
- MP 377 12" Key and MP 377 12" Touch
- MP 377 15" Touch and MP 377 15" Touch Daylight
- MP 377 19" Touch
- TP 012 Touch
- PC-R Key
- PC-R Touch and PC-R Key

HMI device change by the user

If a project contains an HMI device that is not supported by WinCC in the TIA Portal and there is no compatible successor, the migration is canceled. If you want to migrate the project, you must change the HMI device to WinCC yourself before the migration. Use an HMI device which is supported both by WinCC flexible and WinCC for this.

See also

Supported HMI devices (WinCC flexible) (Page 47)

Basics on migration (WinCC flexible) (Page 42)

Configuration change after HMI device change (WinCC flexible)

Note

If you migrate a project with embedded screens, this can result in the project file becoming larger than the old project file.

HMI device replacement by the migration

If the migration makes an HMI device replacement, it replaces the existing HMI device with a compatible successor model. The successor models are further developed and therefore more efficient than their predecessors. The new models therefore support all the properties of their predecessors. Therefore only a little rework is to be expected on the project due to the HMI device replacement.

HMI device replacement by the user

if you change an HMI device yourself before the migration, you must ensure that the HMI device used supports all properties which are used in the project. Compile the project after the HMI device replacement in WinCC flexible. The project must be perfectly compilable before the migration. Then migrate the project to WinCC.

See also

Supported HMI devices (WinCC flexible) (Page 47)

Object support in the migration (WinCC flexible)

Introduction

When migrating projects from WinCC flexible, all configuration data involving an HMI device supported by WinCC will be migrated. Basically, all object types and functions that are available and can be mapped to the new project environment will be fully migrated.

Some global object types are not migrated, for example, dictionaries and global libraries.

Supported object types

The following object types are supported for migration:

- Animations
  Disabled animations are not migrated.
- Audit settings
- Audit reports
- Scheduler
- User administration
- Area pointers
- Faceplates
- Screens
- Screen template
- Data types
- Function lists

- Graphics lists
- Display and operating elements
  All display and operating elements that are available on the supported HMI devices are supported for migration.
- Alarms
- Alarm classes
- Alarm groups
- Project library
- Project languages
- Reports
- Recipes
- Runtime languages
- Runtime scripting
- Sm@rtAccess/Sm@rtService
- Structures
- System events
- System functions
- Texts
- Text lists
- Tags
- Connections
- Effective ranges
- Zones
- Cycles

Unsupported object types

The following object types are not supported by migration:

- Global libraries
- Dictionaries
- Project versions
- Change log

Mapping of the screen navigation

WinCC does not support the screen navigation from WinCC flexible. The data of the screen hierarchy from WinCC flexible are not migrated. To map the functionality of a screen navigation from WinCC flexible, the naivigation buttons are migrated to the button available in WinCC. The system functions migrated to the "ActivateScreen" system function are migrated to the "Click" event of the respective button. The following system functions used in WinCC flexible are not available for the screen navigation in WinCC:

- ActivateRootScreen
- ActivateLeftScreen
- ActivateRightScreen
- ActivateParentScreen
- ActivateFirstChildScreen

These system functions are migrated to the "ActivateScreen" system function. The "Screen name" parameter is taken from the data of the screen hierarchy. If one of the named system functions is called from a screen which is not contained in the screen hierarchy, the "ActivateScreen" function is created without parameters. You must configure the desired screen to this system function after the migration.

Tab sequence in pictures with faceplates

In pictures with faceplates, the tab sequence is changed in both the screen and the faceplate as a result of the migration.

Migration of the screen template

WinCC offers an extended concept for working with screen templates. WinCC offers a global screen and several templates for each device. During migration of a template from WinCC flexible, the objects contained there and the properties configured in the template are migrated to the extended concept of the screen templates of WinCC.

The following objects are migrated to the "global screen" of WinCC:

- Alarm window
- Alarm indicator
- Help indicator
- Function keys of HMI devices with function keys

All other objects and properties are migrated to a template of WinCC.

The connection of the objects and properties to the respective template is automatically adapted.

See also

Supported HMI devices (WinCC flexible) (Page 47)

Migration (WinCC flexible) (Page 41)

Changes of values of object properties by the migration (WinCC flexible) (Page 55)

Changes of values of object properties by the migration (WinCC flexible)

Introduction

The standardization of object properties from WinCC V7 and WinCC flexible requires changes to the object properties during the migration process. The migration calculates the changes in such a way that the representation of the objects after migration is the same as prior to migration. Changes made during migration result in different units of measurements and values in the configuration for some object properties.

Migrating the font settings of an object

In WinCC V7 and WinCC flexible, the unit of measurement "point" is used to denote the size of the fonts used for an object. In WinCC, the unit of measurement "pixel" is used to denote the size of the fonts used for an object. During migration, the font size is converted accordingly to ensure that the representation of the font is the same size at zoom level 100%. Due to the different units of measurement, the numerical values for the font sizes change after migration.

Example:

Font style before migration	Font style after migration
Arial 10 points	Arial 13 pixels
Arial 16 points	Arial 21 pixels
Tahoma 10 points	Tahoma 13 pixels
Tahoma 16 points	Tahoma 21 pixels

Migration of object margins

In WinCC flexible, some objects permit the entry of values <0 and >127 for setup of the object margins for the configuration of the representation. In WinCC, the value range for object margins is limited to values between 0 and 127. Migration changes values <0 to "0" and values >127 to "127".

See also

Object support in the migration (WinCC flexible) (Page 52)

Connections (WinCC flexible)

Migration of connections (WinCC flexible)

Introduction

If you migrate a project in which a supported communication driver is used, this driver is used further in WinCC. Objects which communicate via this driver are migrated 1:1. No reworking is necessary.

Not all communication drivers which are available in WinCC flexible are supported in WinCC. If an unsupported driver is used in the project to be migrated, there are two possible scenarios:

1. A compatible spare driver is available for the used driver.
2. No compatible spare driver is available for the used driver.

Compatible spare driver is available

If a driver is available in WinCC which addresses the used PLC or a comptaible PLC, the driver is automatically replaced during the migration.

You get an appropriate warning if the used driver is replaced.

In this case, check whether all the external tags and area pointers are valid after the migration by means of the migration report.

If the used driver is replaced, all the connection parameters are reset to the standard values. The CPU type is adapted to the appropriate PLC. The properties of the connected tags and area pointers are not changed.

Compatible spare driver is not available

If no driver is available in WinCC which addresses the used PLC or a compatible PLC, the configured connection is not migrated. All external tags which were connected to the PLC by this driver are converted into internal tags. External tag properties are lost in the conversion, e.g. the address in the PLC. All changes to the tags are recorded in the migration report.

You must reconnect the tags with the PLC after the migration and configuring of a connection.

You get an appropriate warning if the driver is not migrated.

Converting the communication driver

The following tables show the mapping of communication drivers from WinCC flexible to WinCC.

Supported communication drivers

WinCC flexible	WinCC
Allen Bradley DF1	Allen Bradley DF1
Mitsubishi FX	Mitsubishi FX
Modicon MODBUS	Modicon MODBUS RTU
Modicon MODBUS TCP/IP	Modicon MODBUS TCP/IP
Omron Hostlink/Multilink	Omron HostLink
OPC	OPC
SIMATIC HMI HTTP Protocol	SIMATIC HMI HTTP Protocol
SIMATIC S7 200	SIMATIC S7 200
SIMATIC S7 300/400	SIMATIC S7 300/400

Compatible communication drivers

WinCC flexible	WinCC
Allen Bradley DH485	Allen Bradley DF1
Allen Bradley E/IP C.Logix	Allen Bradley EtherNet/IP
Mitsubishi Protocol 4	Mitsubishi MC TCP/IP [1)]
1) If you are using an FX2N controller, you must set the Mitsubishi FX communication driver after the migration.	

Not supported communication drivers

WinCC flexible	WinCC
GE Fanuc SNP	Not supported
ISAC	Not supported
LG GLOFA-GM	Not supported
SIMATIC 500/505 DP	Not supported
SIMATIC 500/505 serial	Not supported
SIMATIC S5 AS511	Not supported
SIMATIC S5 DP	Not supported
Telemecanique Uni-Telway	Not supported

See also

Migration (WinCC flexible) (Page 41)

Supported HMI devices (WinCC flexible) (Page 47)

Adaptation of the configuration for non-migrated connection (WinCC flexible) (Page 57)

Migration of area pointers (WinCC flexible) (Page 58)

Migration of tags (WinCC flexible) (Page 59)

Adaptation of the configuration for non-migrated connection (WinCC flexible)

Introduction

If a connection cannot be migrated you have the following options:

- Change the configuration in WinCC flexible before the migration
- Change the configuration after the migration

Change a connection before the migration

If the communication driver selected for the connection is not supported by the migration, you must select a driver which is supported by the migration. The HMI device must also support the selected communication driver.

If no suitable driver is available for the used HMI device, you must use a suitable HMI device. Then adapt the configuration in WinCC flexible and recompile the project. Migrate the project after successful adaptation and compiling. The connection is then also migrated.

Changing a connection before the migration

If you migrate a WinCC flexible project in which the connection is not migrated, all external tags of this connection are mapped to internal tags. You receive an appropriate entry in the migration protocol for every tag concerned.

You must configure a new connection after the migration. For this connection you select a communication driver which supports the used HMI device. If no suitable driver is available for the used HMI device, you must use a suitable HMI device.

If you have configured the new connection, you must reconnect all tags which were mapped to internal tags before the migration. Open the migration report to make reconfiguration easier. Reconfigure the tags according to the log entry.

If the project contained area pointers, you must also reconfigure these. You will also find corresponding entries for the area pointers in the migration log.

See the section Migrating projects from WinCC flexible (WinCC flexible) (Page 44) for further information about the migration log.

See also

Migration of connections (WinCC flexible) (Page 55)

Migrating projects from WinCC flexible (WinCC flexible) (Page 44)

Migration of area pointers (WinCC flexible)

Introduction

The migration of area pointers depends on the used communication driver.

- If the used communication driver is supported by the migration, area pointers from WinCC flexible are taken over unchanged in the migration.
- if the used communication driver is not supported by the migration, area pointers are not migrated.

Migration of the area pointers

If the connection used by the area pointers is fully migrated, the area pointers are also fully migrated. The parameters of the area pointers are taken over unchanged.

If you replace the communication driver of the connection before the migration, check whether all the parameters of the area pointers are still valid after the migration.

If the connection used by the area pointer is not migrated, the area pointers are not migrated either. See Adaptation of the configuration for non-migrated connection (WinCC flexible) (Page 57) for additional information.

See also

Migration of connections (WinCC flexible) (Page 55)

Adaptation of the configuration for non-migrated connection (WinCC flexible) (Page 57)

Migration of tags (WinCC flexible)

Introduction

You need to make some special considerations when migrating tags. The following aspects should be distinguished:

- Migrating data types of tags
- Migrating internal tags
- Migrating external tags
- Tag names
- Tag limits

Migrating data types

WinCC features some other data types and uses different data type names than WinCC flexible. When migrating a relevant tag, the data type from WinCC flexible is mapped to the corresponding data type in WinCC. See Migrating data types (WinCC flexible) (Page 75) for additional information.

Migrating internal tags

Internal tags are always migrated completely. Only the data type names and tag names may change due to migration.

Migrating external tags

If the connection used by the external tags is migrated, the external tags are also fully migrated. The migration of external tags depends on whether the used communication driver is supported by the migration. See Migration of connections (WinCC flexible) (Page 55) for additional information.

If the connection used by the external tags is not migrated, the external tags are not migrated either. The external tags are then mapped to internal tags. You must configure a new connection and reconnect the tags with the tags of the PLC after the migration.

Migrating names of tags

In WinCC flexible, tags located in different folders can have the same name. In WinCC, the tag name must be unique on the configured HMI device. This means that tags with the same name from different folders will be renamed during migration. See Basics on migration (WinCC flexible) (Page 42) for additional information.

See also

Migration of connections (WinCC flexible) (Page 55)

Migration (WinCC flexible) (Page 41)

Migration of alarm classes and alarm groups (WinCC flexible) (Page 61)

Migrating scripts (WinCC flexible) (Page 63)

Migration of language-dependent contents (WinCC flexible) (Page 65)

Migration of libraries (WinCC flexible) (Page 67)

Migrating data types (WinCC flexible) (Page 75)

Basics on migration (WinCC flexible) (Page 42)

Migrating structures (WinCC flexible)

The "StringChar" data type is not supported in HMI user data types in WinCC.

If you have used this data type in a structure in a WinCC flexible project, an invalid element is created by the migration in the HMI user data type.

After migration , you must re-work this user data type in WinCC and enable the user data type.

At the same time, check the offset of the following elements and any existing interconnections between a faceplate and the elements of the user data type. Adapt the elements if required.

Migration of logs

Storage location of logs

WinCC flexible allowed you to store logs in a database which was automatically set up upon WinCC flexible installation ("System-defined data source" setting). This option is not available in WinCC.

If you have used this setting in your WinCC flexible project, it will be changed to "User-defined name of data source" during migration. Before you can store logs in a database, you must configure an ODBC data source in the Windows control panel and configure the name of the user data source specified there as the "name of the data source" in the log in WinCC.

Migration of alarm classes and alarm groups (WinCC flexible)

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Error	Alarms
System	System
Warnings	Events

The names of the alarm classes can be changed as necessary after migration.

Migrating alarm groups

Migration will migrate only those alarm groups actually in use.

Alarm groups with an ID from 1-31 will be migrated 1:1.

A corresponding alarm group is created in WinCC for each alarm class in the system. These alarm groups created by the system are assigned IDs beginning with the number 32 and consecutively incremented. The 4 pre-defined alarm classes in every WinCC project are automatically given IDs 32-35 by their alarm groups. Additionally created alarm group and an additional ID is assigned to each user-defined alarm class. Therefore, the IDs for alarms groups with IDs > 31 may be changed after migration. This step also changes the assignment of the alarm group names to the IDs.

Example:

In the example, you can see the assignment of the IDs in WinCC for the migration.

Alarm groups	ID in WinCC flexible	ID in WinCC	
Alarm group 1-16	1-16	1-16	Default for alarm groups from system alarms
Alarm group 17-31	17-31	17-31	Custom alarm groups
		32-35	Default in WinCC for alarm groups of pre-defined alarm classes.
Alarm group 32	32	36	Changed assignment of ID to alarm group in WinCC
Alarm group 33	33	37	Changed assignment of ID to alarm group in WinCC

Also note:

When migrating alarm groups that supposedly have the same group name, the migration adapts the name. This occurs, for example, when a group name contains a space at the end of the name. The migration deletes all existing spaces at the end of names. If two groups obtained the same group names due to this deletion, the migration adds the suffix "# Mign" to the group name of the following alarm groups, where "n" stands for a sequential number.

Example:

The following alarm groups exist in WinCC flexible:

"AlarmGroup_18"

"AlarmGroup_18 " - group name contains one space

"AlarmGroup_18 " - group name contains two spaces

"AlarmGroup_18" is the alarm group with the highest number.

Result after migration:

"AlarmGroup_18"

"AlarmGroup_18#Mig1"

"AlarmGroup_18#Mig1.1"

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Error	Errors
System	System
Warnings	Warnings

The names of the alarm classes can be changed as necessary after migration.

Display of ALARM_S messages and SIMATIC SFM messages

In WinCC flexible you can activate the display classes for ALARM_S messages in integrated projects. In WinCC flexible, you activate the display of SIMATIC SFM messages via a separate setting. The separate setting for activating the display of SIMATIC SFM messages is not required in WinCC. You control the display of SIMATIC SFM messages, and also the display of ALARM_S messages in WinCC only by activating the corresponding display class.

The changed concept may cause the display of messages to change following migration.

If all the display classes for ALARM_S messages are activated and the display of SIMATIC SFM messages is deactivated in the WinCC flexible project , ALARM_S messages and SIMATIC SFM messages are displayed following migration.

To ensure that only ALARM_S messages are displayed following migration, you have to assign the SIMATIC SFM messages to an unused display class after migration to STEP 7. You then have to deactivate this display class in WinCC.

If all the display classes for ALARM_S messages are deactivated and the display of SIMATIC SFM messages is activated in the WinCC flexible project , ALARM_S messages and SIMATIC SFM messages are not displayed following migration.

To ensure that only SIMATIC SFM messages are displayed following migration, you have to assign the SIMATIC SFM messages to an unused display class after migration to STEP 7. You then have to activate this display class in WinCC.

The display class is dependent on the settings in STEP 7. The default setting for SIMATIC SFM messages in Step 7 is the display class "0". To activate the display in WinCC, the display class "0" must be activated.

You activate the display class in WinCC in the Runtime settings of the respective HMI device in the "Messages" category.

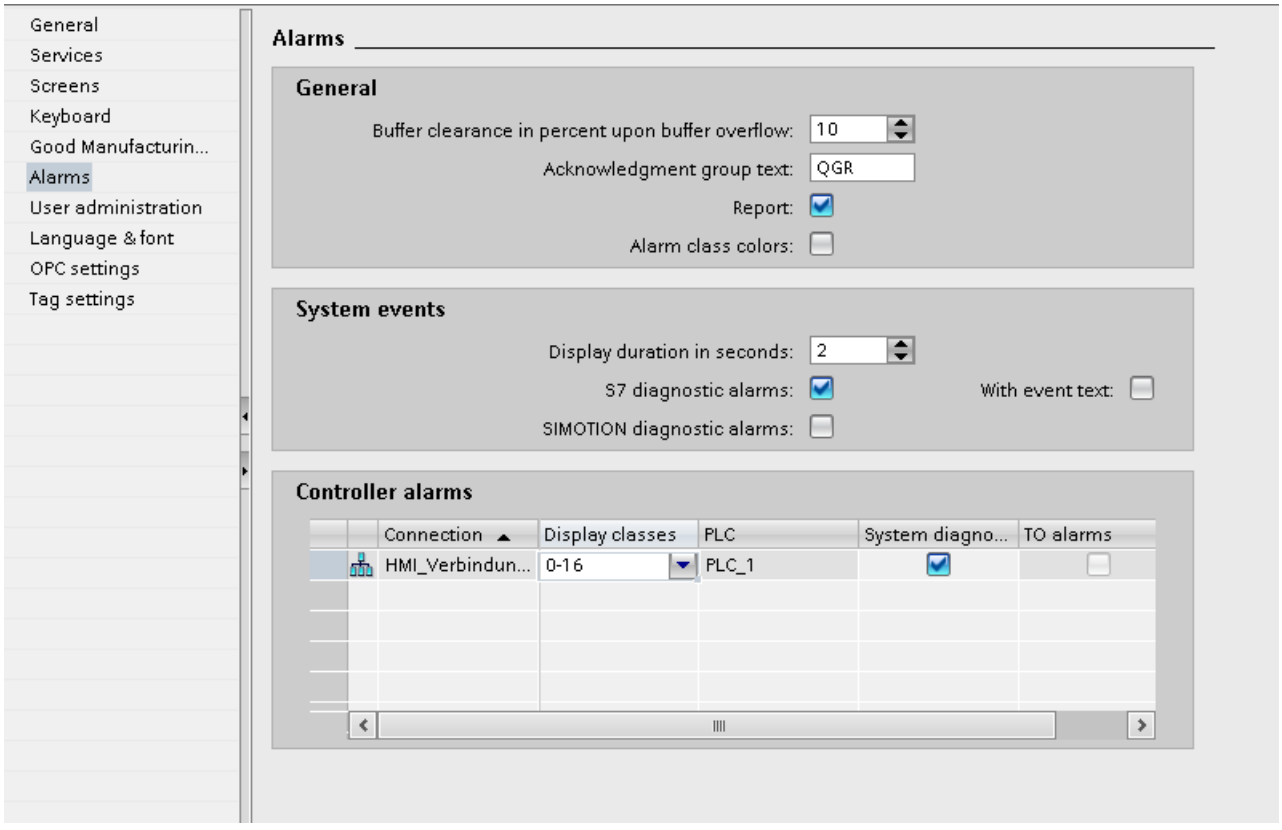

See also

Migration of tags (WinCC flexible) (Page 59)

Migrating scripts (WinCC flexible)

Introduction

The migration supports VB-scripts which were created in WinCC flexible. For a VB-script to be migrated successfully it must be functionable in WinCC flexible first.

Note

Script errors

The most efficient way to locate scripting errors in the course of the initial test run after migration is to use an installed Script Debugger and the diagnostics controls.

Migration of a VB script

A script is analyzed in the migration and adapted to the system behavior of WinCC if necessary. The following is adapted:

- System functions which have changed their names are renamed.
  This concerns the following system functions:

Function name in WinCC flexible	Function name in WinCC
IncreaseValue	IncreaseTag
DecreaseValue	DecreaseTag
SetBrightness	SetAndGetBrightness
SetValue	SetTag

- The access to tags as parameters of system functions is adapted to the system behavior of WinCC. In WinCC scripts are no longer transfromed as in WinCC flexible. The scripts are executed directly as a source code. Therefore the stricter rules for the VBS syntax apply. A tag call is always migrated with inverted commas. If several parameter types are allowed for a system function, these are migrated with the keyword "SmartTags".
  Example 1:
  The value of the "temperature" tag should be incremented by the value "1".
  Valid expressions in WinCC flexible:
  IncreaseValue temperature, 1
  IncreaseValue "temperature", 1
  IncreaseValue SmartTags("temperature"), 1
  Valid expression in WinCC:
  IncreaseTag "temperature", 1
  Example 2:
  You use a system function on which several parameters are allowed. The value of the "temperature" tag should be incremented by the value of the "heatcontrol" tag.
  Valid expression for WinCC flexible:
  IncreaseValue "temperature", "heatcontrol"
  Valid expression for WinCC:
  IncreaseTag "temperature", SmartTags("heatcontrol")
- Objects renamed by the migration are also renamed in the script when using in a script. See the section Basics on migration (WinCC flexible) (Page 42) for further information about the renaming objects. You only have to observe the following rules when renaming objects: If you address objects whose names are dynamically generated by the script with the help of a script, the object names in the script can not be automatically changed by the migration. In such a case, you have to correct the generation of the object names in the script after migration.

See also

Migration of tags (WinCC flexible) (Page 59)

Basics on migration (WinCC flexible) (Page 42)

Migration of language-dependent contents (WinCC flexible)

Introduction

WinCC offers the same options for configuring projects in different languages as those available in WinCC flexible. All languages supported by WinCC are included in the migration of a project.

Migrating language-dependent content

The following language-dependent content is migrated:

- Project languages
- Project texts
- Fonts for display in runtime
- Language-dependent graphics

You need to consider the following when migrating language-dependent content:

- The operating system on the PC performing the migration must support the project languages used in the project.
- The fonts used for runtime display must be installed on the PC performing the migration.
- Dictionaries are not supported by the migration.

Editiing language of integrated projects following migration

During migration of an integrated project, the project components to be migrated from STEP 7 and WinCC flexible also bring their respective settings for the editing language. In WinCC there is only one editing language for all project components. Migration activates for the mgrated project the editing language which was set in STEP 7 prior to migration. If this setting is not the same as the setting from WinCC flexible, the configured texts are no longer visible in WinCC. No text is displayed at the usage locations, or only the entry "Text" can be seen. To make the texts visible, you must change the editing language. Click the "Tasks" taskcard at the right-hand edge of the TIA portal and select the correct editing language in the "Language & Resources" area.

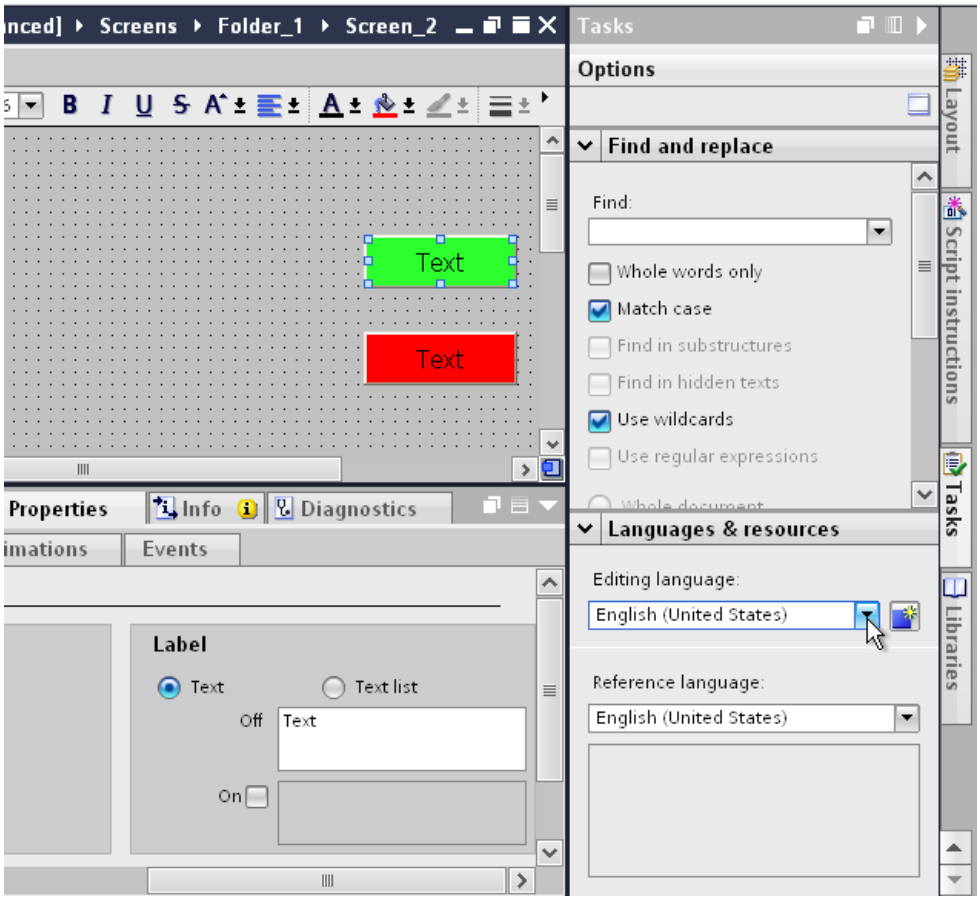

Unsupported languages

The migration of language-dependent content depends on whether or not WinCC supports the respective language.

If a project only contains project languages not supported by WinCC, the project will not be migrated.

If a project contains supported and unsupported project languages, only the supported languages will be migrated. The editing language and reference language are set to a supported language.

The following languages are not supported by WinCC:

- Arabic
- Hebrew
- Dhivehi
- Gujarati
- Kannada
- Tamil
- Telugu
- Urdu
- Punjabi
- Persian
- Syrian

See also

Migration of tags (WinCC flexible) (Page 59)

Migration of libraries (WinCC flexible)

Introduction

You need to consider two different cases when migrating from libraries:

1. Migrating a project library
2. Migrating a global library

Migrating a project library

A project library is stored together with the project data in the project file. For this reason, a project library is migrated with the same restrictions as the project data.

Migrating a global library

Global libraries are not supported by the migration. The library objects used in the project will be migrated, however. The library objects are copied when used in the project and then no longer have a connection to the library.

To migrate a global library, you must copy or move the objects contained in the library to the project library. The objects are then included in the migration. In WinCC, you move the migrated objects to a new global library that is created. You can copy or move both individual objects or entire library categories.

See also

Migration of tags (WinCC flexible) (Page 59)

2.8.1.3 Migrating runtime data (WinCC flexible)

Migrating runtime data (WinCC flexible)

Introduction

Only the configuration data are migrated by the migration when migrating a project. The runtime data are not affected. You need to update the runtime data following migration.

The runtime data consists of the following:

- Runtime project
  The runtime project contains the compiled project data.
- Recipe data and user administration
  The recipe data and user administration are data that can be changed in runtime.
- Log data
  The data of tag logs and alarm logs are acquired and logged in runtime.

Migrating recipe data and user administration

If the recipe data and user administration were changed in runtime, you need to back up this information from the HMI device before you load the migrated project.

Depending on the used HMI device, you have different options for saving the above data.

If the HMI device supports external memory media and the recipe data are saved there, the data remain on the memory medium. External memory media are for example StorageCard or a network drive. You use the data again after the migration. The user adminsitration is not saved on an external memory medium so you have to save these data, e.g. with ProSave.

If the recipe data are saved in the internal memory of the HMI device, save these data on an external memory medium. Use ProSave to save the recipe data. If you have already configured appropriate system functions in your project, use the system functions. The following system functions are available for the recipe data:

- "ExportDataRecords" for the backup
- "ImportDataRecords" for the restore

You update the runtime project by compiling the project in WinCC again and loading it to the HMI device.

After you have loaded the migrated project on the HMI device, restore the recipe data and the user administration to the HMI device.

See also

Migration (WinCC flexible) (Page 41)

Backing up recipe data and user administration (WinCC flexible) (Page 69)

Restoring recipe data and user administration (WinCC flexible) (Page 70)

Backing up log data (WinCC flexible) (Page 71)

Backing up recipe data and user administration (WinCC flexible)

Introduction

To continue using the recipe data and user administration in a migrated project, you first need to back up this data from the HMI device. Then load the data into the migrated WinCC project. Use ProSave to back up the data.

Requirement

- The WinCC flexible project is running on the HMI device in Runtime.
- The HMI device is connected to a PC on which ProSave is installed.

Procedure

Proceed as follows to back up the recipe data and user administration:

1. Start ProSave.
2. Select the device type and the connection parameters in the "General" tab.
3. Open the "Backup" tab.
4. Select the "Recipes from the device memory" entry in the "Data type" box.
   Do not select "Complete backup" because otherwise you will not be able to select separately when restoring the recipe data.
5. Navigate to the desired location in the "Save as" box and click "Start Backup".
   The recipe data are saved.
6. Select "User administration" in the "Data type" box and click "Start Backup".
   The user administration is saved.

For additional information refer to the online help for ProSave.

Alternative procedure

ProSave is automatically installed with WinCC flexible. The entire functional range of ProSave is available on the configuration PC within WinCC flexible via the menu command "Project > Transfer".

Alternatively, you can back up the recipe data and user administration via the ProSave integrated in WinCC flexible. Start WinCC flexible and select the menu command "Project > Transfer > Backup". Back up the recipe data and user administration as described in steps 4-6.

See also

Migrating runtime data (WinCC flexible) (Page 68)

Restoring recipe data and user administration (WinCC flexible)

Introduction

To continue using saved recipe data and user administration after the migration, you first need to compile the migrated project and load it to the HMI device. You can then transfer the saved data to the HMI device. Use ProSave to restore the data.

Requirement

- The migrated project has been transferred to the HMI device and is running in runtime.
- The HMI device is connected to a PC on which ProSave is installed.

Procedure

Proceed as follows to load the saved recipe data and user administration to the HMI device:

1. Start ProSave.
2. Select the device type and the connection parameters in the "General" tab.
3. Open the "Restore" tab.
4. Navigate to the location of the saved recipe data in the "Opening..." box and select the file.
5. Click "Start Restore".
   The recipe data will be transferred to the HMI device..
6. Repeat steps 4-5 to restore the user administration.
   The user administration will be transferred to the HMI device.

For additional information refer to the online help for ProSave.

Alternative procedure

ProSave is automatically installed with WinCC. The entire functional range of ProSave is available on the configuration PC within WinCC flexible via the menu command "Project > Transfer".

You can also restore the recipe data and user administration via the ProSave integrated in WinCC. Start WinCC and select the menu command "Online > Device maintenance > Restore". Restore the recipe data and user administration as described in steps 4-6.

See also

Migrating runtime data (WinCC flexible) (Page 68)

Backing up log data (WinCC flexible)

Introduction

If an HMI device supports external memory media and the log data are saved there, the data remain on the memory medium. External memory media are for example StorageCard or a network drive. If a log is saved on an external memory medium, the migrated project accesses this storage location again after the migration. In this case the log data must not be backed up.

Backing up log data

If you want to back up the log data externally before the migration, you have the following options:

- Backup with the "ArchiveLogFile".
  If you have already configured the "ArchiveLogFile" function in the WinCC flexible project, use this function to back up the data.
- Copy the log files by Copy&Paste with the Windows Explorer to an external memory medium or network drive.

See also

Migrating runtime data (WinCC flexible) (Page 68)

2.8.1.4 Migrating integrated projects (WinCC flexible)

Migrating integrated projects (WinCC flexible)

Introduction

The controllers and HMI devices contained in a project integrated in STEP 7 are linked together by the configuration. The configuration data of WinCC flexible and STEP 7 are also connected. When an integrated project is migrated, the complete project will be migrated with components from WinCC flexible and STEP 7. The connections remain intact.

Note

It is advisable to compile and save an integrated project in WinCC flexible before you migrate it. You can be sure that the data in WinCC flexible and STEP 7 is synchronized if compilation was completed without errors.

Migrating an integrated project

When migrating an integrated project, the same requirements apply for the WinCC flexible component as those for the migration of a non-integrated WinCC flexible project. The objects and properties contained in the WinCC flexible component must be supported by WinCC, for example, the HMI device or the communication driver. The "Online" property must be activated on the configured connection. A connection with deactivated "Online" property is not migrated.

In addition to the requirements for the WinCC flexible component, there are also requirements for the STEP 7 component of the integrated project. The objects and properties contained in the STEP 7 V5.4 SP5 or V5.5 component must be supported in STEP 7. For detailed information, refer to the documentation for STEP 7.

To fully migrate an integrated project and then edit it, the following components must be installed on the PC performing the migration:

- STEP 7 V5.4 SP5 or STEP 7 V5.5
- WinCC flexible 2008 SP2 or WinCC flexible 2008 SP3 or WinCC flexible 2008 SP5
- STEP 7

If you only want to save the project in migration format, you can use the migration tool. See Basics on migration (WinCC flexible) (Page 42) for additional information.

An integrated project is always fully migrated. If you only want to migrate the WinCC flexible project it contains, you need to separate it from the STEP 7 project before the migration. To separate the project from the integrated form, open the project in STEP 7 V5.4 SP5 or V5.5. Open the WinCC flexible project in the SIMATIC Manager. The project is opened with WinCC flexible. In WinCC flexible, select the menu command "Project > Copy project from STEP 7". WinCC flexible saves a non-integrated copy of the project.

See also

Migration (WinCC flexible) (Page 41)

Migrating an integrated project (WinCC flexible) (Page 72)

Basics on migration (WinCC flexible) (Page 42)

Migrating an integrated project (WinCC flexible)

Introduction

When migrating an integrated project, the components from both the WinCC flexible project and the STEP 7 project will be migrated. This means you need to select the project file with the file extension "*.s7p" for migration. During migration, the data is copied from the existing project and migrated to a new project. You cannot migrate to an existing project.

The migration can be started in both the Portal view and the Project view.

You should only migrate a project to a re-started TIA Portal.

If you only want to save the project in migration format, you can use the migration tool. For more information, refer to basics of migration (WinCC flexible).

Requirement

- STEP 7 V5.4 SP5 or STEP 7 V5.5 and all option packages used are installed.
- STEP 7 and all option packages used are installed.
- The TIA Portal is restarted.
- No project is open in WinCC.
- An integrated project is available.
- The integrated project is not open.

Procedure

To migrate an integrated project in the portal view, follow these steps:

1. Select the action "Start > Migrate Project".

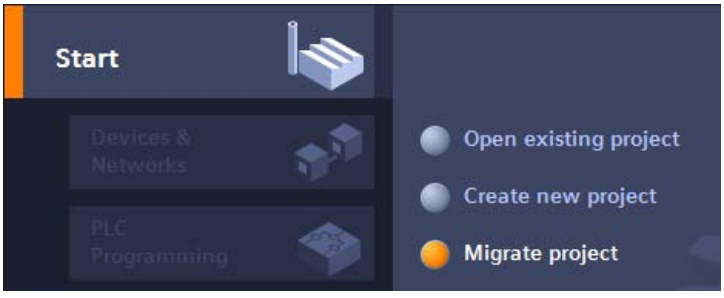

2. In the "Source path" box, navigate to the project you want to migrate.

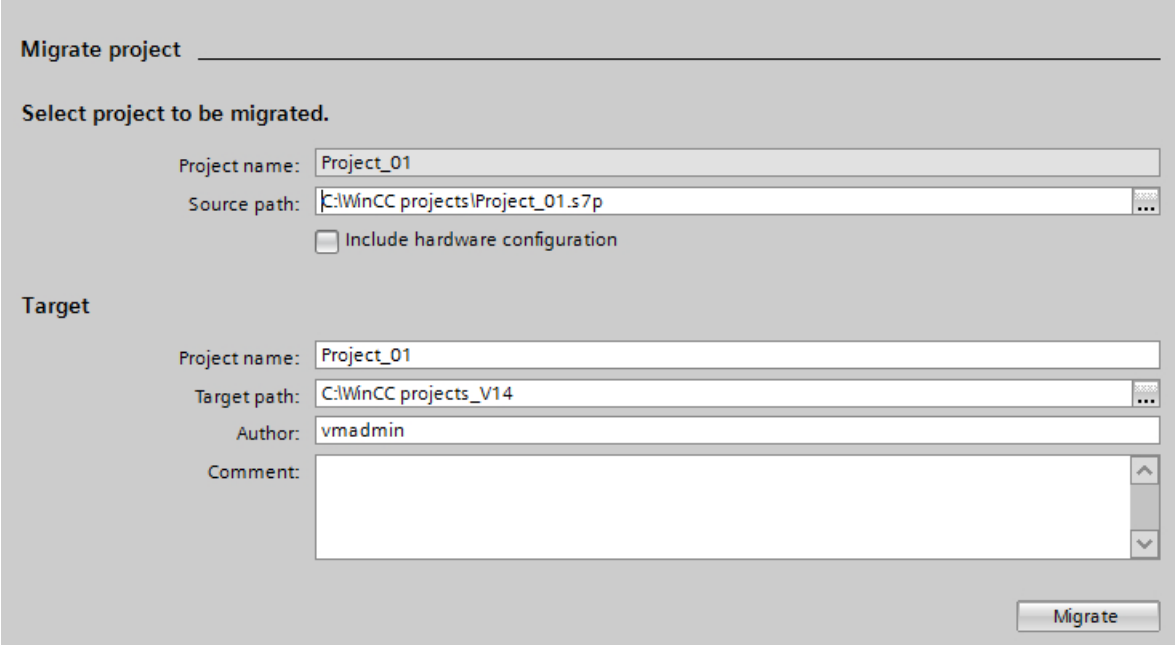

3. Select the "*.s7p" project file.
4. Change the information for the project to be created, if necessary. For example, change the project name or project path. The data to be migrated is created in the new project.

5. To migrate the project with the hardware configuration, enable "Include hardware configuration".
6. Click "Migrate".
   A new project is created and migration of the data is started:
   – The Project view opens.
   – The progress of the migration is shown in a migration window.
   – Warnings and errors about the migration process are displayed in the Inspector window under "Info > General".
   – All information about the migration is saved in a log file.
   – A message is displayed upon completion of the migration. The message contains a link that you can use to open the log file.
7. Once migration is completed, save the project.

Once the migration is complete, you will find a newly created device for each migrated HMI device and controller in the project tree. These devices include the migrated data.

Note

Integrated projects with alarm views

The alarm classes in the alarm view can be disabled during migration of the project. Once migration of the project is complete, check the settings in the alarm view. Enable the required alarm classes in the Inspector window of the alarm view if needed under "Properties > General".

Opening the migration log at a later point in time

The migration log is saved together with the migrated project. You can view the log at a later point in time. Open the log file as follows:

1. Open "Common data > Logs" in the project navigation.
2. Double-click the log file. The migration log opens.

See also

Migrating integrated projects (WinCC flexible) (Page 71)

Basics on migration (WinCC flexible) (Page 42)

2.8.1.5 Reference (WinCC flexible)

Migrating data types (WinCC flexible)

Introduction

To harmonize the data types used by PLCs and HMI systems, some data types of the internal HMI tags are renamed. The naming takes place in accordance with IEC conventions. Because only the names change, there are no changes to the internal tags for the configuration.

The following table describes the mapping of the internal data types from WinCC flexible to the data types in WinCC.

Migrating internal data types

The internal data types are mapped as follows during migration:

Internal data types WinCC flexible	Internal data types WinCC
Bool	Bool
Char	SInt
Byte	USInt
Int	Int
UInt	UInt
Long	DInt
ULong	UDInt
Float	Real
Double	LReal
String	WString
DateTime	DateTime

Migrating external data types

See the following pages for how to map the available communication drivers.

See also

Migration (WinCC flexible) (Page 41)

Migrating data types of Allen-Bradley DF1 (WinCC flexible) (Page 76)

Migrating data types of Allen-Bradley DH485 (WinCC flexible) (Page 77)

Migrating data types of Allen-Bradley Ethernet IP (WinCC flexible) (Page 77)

Migrating data types of GE Fanuc SNP (WinCC flexible) (Page 78)

Migrating data types of LG GLOFA GM (WinCC flexible) (Page 78)

Migrating data types of Mitsubishi FX (WinCC flexible) (Page 79)

Migrating data types of Modicon Modbus (WinCC flexible) (Page 80)

Migrating data types of Modicon Modbus TCP/IP (WinCC flexible) (Page 81)
Migrating data types of Omron Hostlink/Multilink (WinCC flexible) (Page 81)
Migrating data types of OPC (WinCC flexible) (Page 82)
Migrating data types of SIMATIC 500/505 DP (WinCC flexible) (Page 82)
Migrating data types of SIMATIC 500/505 serial (WinCC flexible) (Page 83)
Migrating data types of SIMATIC S5 AS511 (WinCC flexible) (Page 84)
Migrating data types of SIMATIC S5 DP (WinCC flexible) (Page 84)
Migrating data types of SIMATIC S7 200 (WinCC flexible) (Page 85)
Migrating data types of SIMATIC S7 300/400 (WinCC flexible) (Page 85)
Migrating data types of Telemecanique Uni-Telway (WinCC flexible) (Page 88)
Migrating data types of Mitsubishi Protocol 4 (WinCC flexible) (Page 79)
Migrating data types of SIMATIC HMI HTTP Protocol (WinCC flexible) (Page 83)

Migrating data types of Allen-Bradley DF1 (WinCC flexible)

Migrating data types Allen-Bradley DF1

The data types of the Allen-Bradley DF1 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
ASCII	ASCII
BCD4	UInt
BCD8	UDInt
Bit	Bool
Int	Int
Long	DInt
Real	Real
UInt	UInt
ULong	UDInt

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of Allen-Bradley DH485 (WinCC flexible)

Migrating data types Allen-Bradley DH485

The Allen-Bradley DH485 communication driver is not supported by WinCC, it is replaced by the Allen-Bradley DF1 driver. The data types of the Allen-Bradley DH485 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
ASCII	ASCII
Bit	Bool
Int	Int
Long	DInt
Real	Real
UInt	UInt
ULong	UDInt

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of Allen-Bradley Ethernet IP (WinCC flexible)

Migrating data types Allen-Bradley Ethernet IP

The data types of the Allen-Bradley Ethernet IP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
DInt	DInt
Int	Int
Real	Real
SInt	SInt
String	String
UDInt	UDInt
UInt	UInt
USInt	USInt

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of GE Fanuc SNP (WinCC flexible)

Migrating data types GE Fanuc SNP

The GE Fanuc SNP communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the GE Fanuc SNP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
BCD4	UInt
BCD8	UDInt
Bit	Bool
Byte	USInt
DInt	DInt
Word	UInt
Int	Int
Real	Real
UInt	UInt
DWord	UDInt

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of LG GLOFA GM (WinCC flexible)

Migrating data types LG GLOFA GM

The LG GLOFA GM communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the LG GLOFA GM communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	USInt
DInt	DInt
DWord	UDInt
Int	Int
SInt	SInt
String	WString
Time	UDInt
UDInt	UDInt
UInt	UInt
USInt	USInt
Word	UInt

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of Mitsubishi FX (WinCC flexible)

Migrating data types Mitsubishi FX

The data types of the Mitsubishi FX communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
12 Bit Block	12-Bit Block
16 Bit Block	16-Bit Block
20 Bit Block	20-Bit Block
24 Bit Block	24-Bit Block
28 Bit Block	28-Bit Block
32 Bit Block	32-Bit Block
4 Bit Block	4-Bit Block
8 Bit Block	8-Bit Block
Bit	Bool
Double	DWord
IEEE-Float	Real
String	String
Word	Word

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of Mitsubishi Protocol 4 (WinCC flexible)

Migrating data types Mitsubishi Protocol 4

The Mitsubishi Protocol 4 communication driver is not supported by WinCC, it is replaced by the Mitsubishi MC TCP/IP driver. The data types of the Mitsubishi Protocol 4 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
4 Bit Block	4-Bit Block
8 Bit Block	8-Bit Block
12 Bit Block	12-Bit Block
16 Bit Block	16-Bit Block
20 Bit Block	20-Bit Block
24 Bit Block	24-Bit Block

Data type in WinCC flexible	Data type in WinCC
28 Bit Block	28-Bit Block
32 Bit Block	32-Bit Block
Bit	Bool
DInt	DInt
DWord	DWord
Int	Int
Real	Real
String	String
Word	Word

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of Modicon Modbus (WinCC flexible)

Migrating data types Modicon Modbus

The Modicon Modbus communication driver is not supported by WinCC, it is replaced by the Modicon Modbus RTU driver. The data types of the Modicon Modbus communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	+/- Double
+/-Int	+/- Int
16 Bit Group	16 Bit Group
ASCII	ASCII
Bit	Bit
Double	Double
Float	Float
Int	Int

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of Modicon Modbus TCP/IP (WinCC flexible)

Migrating data types Modicon Modbus TCP/IP

The data types of the Modicon Modbus TCP/IP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	+/- Double
+/-Int	+/- Int
16 Bit Group	16 Bit Group
ASCII	ASCII
Bit	Bit
Double	Double
Float	Float
Int	Int

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of Omron Hostlink/Multilink (WinCC flexible)

Migrating data types Omron Hostlink/Multilink

The Omron Hostlink/Multilink communication driver is not supported by WinCC, it is replaced by the Omron Host Link driver. The data types of the Omron Hostlink/Multilink communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-DEC	Int
+/-LDEC	DInt
ASCII	String
BIN	Bool
BYTE	Byte
DEC	UInt
IEEE	Real
LDEC	UDInt

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of OPC (WinCC flexible)

Migrating data types OPC

The data types of the OPC communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	VT_BOOL
Byte	VT_UI1
Char	VT_I1
Date	VT_DATE
Double	VT_R8
DWord	VT_UI4
Float	VT_R4
Long	VT_I4
Short	VT_I2
String	VT_BSTR
Word	VT_UI2

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of SIMATIC 500/505 DP (WinCC flexible)

Migrating data types SIMATIC 500/505 DP

The SIMATIC 500/505 DP communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the SIMATIC 500/505 DP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	DInt
+/-Int	Int
ASCII	WString
Bit	Bool
Double	UDInt
Int	UInt
Real	Real

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of SIMATIC 500/505 serial (WinCC flexible)

Migrating data types SIMATIC 500/505 seriell

The SIMATIC 500/505 seriell communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the SIMATIC 500/505 seriell communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	DInt
+/-Int	Int
ASCII	WString
Bit	Bool
Double	UDInt
Int	UInt
Real	Real

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of SIMATIC HMI HTTP Protocol (WinCC flexible)

Migrating data types SIMATIC HMI HTTP Protocol

The data types of the SIMATIC HMI HTTP Protocol communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	USInt
Char	SInt
DateTime	DateTime
Double	LReal
Float	Real
Int	Int
Long	DInt
String	WString
UInt	UInt
ULong	UDInt

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of SIMATIC S5 AS511 (WinCC flexible)

Migrating data types SIMATIC S5 AS511

The SIMATIC S5 AS511 communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the SIMATIC S5 AS511 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bit in D	Bool
Bit in W	Bool
DF	DInt
DH	UDInt
KC	WString
KF	Int
KG	Real
KH	UInt
KM	UInt
KT	UDInt
KY	UInt
KZ	UInt

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of SIMATIC S5 DP (WinCC flexible)

Migrating data types SIMATIC S5 DP

The SIMATIC S5 DP communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the SIMATIC S5 DP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bit in D	Bool
Bit in W	Bool
DF	DInt
DH	UDInt
KC	WString
KF	Int
KG	Real
KH	UInt
KM	UInt
KT	UDInt

Data type in WinCC flexible	Data type in WinCC
KY	UInt
KZ	UInt

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of SIMATIC S7 200 (WinCC flexible)

Migrating data types SIMATIC S7 200

The data types of the SIMATIC S7 200 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	Byte
Char	Char
DInt	DInt
DWord	DWord
Int	Int
Real	Real
StringChar	StringChar
Timer	Timer
Word	Word

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of SIMATIC S7 300/400 (WinCC flexible)

Migrating data types SIMATIC S7 300/400

The data types of the SIMATIC S7 300/400 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	Byte
Char	see below
Counter	see below
Date	Date

Data type in WinCC flexible	Data type in WinCC
Date and Time	Date_And_Time
DInt	DInt
DWord	DWord
Int	Int
Real	Real
String	String
StringChar	see below
Time	Time
Time of Day	Time_Of_Day
Timer	see below
Word	Word

Special considerations for some data types

There are special considerations to be made when migrating external tags that contain data types of a SIMATIC S7-300/400 PLC.

Mapping of the S7 data type "Char"

The S7 data type "Char" is a data type for mapping characters according to the specification. However, since this data type is often used for reading and writing numerical values, it is mapped in WinCC to the S7 data type "Byte". If this should be the case during migration, an alarm will appear in the output window.

If the S7 data type "Char" is used for numerical values and negative numbers were configured at the point of use, the result is an error in mapping to the S7 data type "Byte". The S7 data type "Byte" cannot map any negative numbers. You have to adapt the configuration accordingly to correct the error. Use a signed data type, such as the data type "Int", for processing positive and negative numerical values.

If the S7 data type "Char" is used for mapping characters, you must change the configuration after migration. To represent characters, use the data type "String".

When an integrated project is migrated, the data type "Char" in WinCC is also migrated to the data type "Byte". With a connected PLC tag, the data type "Char" remains "Char". As a result of changing the data type of the HMI tag, symbolic addressing of the tags in question is not migrated. After migration. the tags are interconnected by absolute addresses and continue to work. If you want to restore symbolic addressing, you have to change the configuration accordingly after the migration.

Mapping an array of the S7 data type "Char"

An array of the S7 data type "Char" is mapped to an array of the data type "Byte" during migration.

If an array of the S7 data type "Char" is used for numerical values and negative numbers were configured at the point of use, the result is an error in mapping to an array of the S7 data type "Byte". The S7 data type "Byte" cannot map any negative numbers. You have to adapt the

configuration accordingly to correct the error. Use a signed data type, such as the data type "Int", for processing positive and negative numerical values.

Mapping of the S7 data type "Counter"

An external tag with the S7 data type "Counter" with counter address is mapped to the S7 data type "Counter". The address will be retained.

If an external tag with the S7 data type "Counter" addresses a data block or a bit memory address, it is is mapped to the S7 data type "Word". The address will be retained. The migration sets the coding to "SimaticBCDCounter".

The S7 data type "Counter" has a value range of 0-999. When supplied by the S7 data type "Word" the value range may be exceeded on the PLC side. Ensure that you are observing the value range.

Example:

WinCC flexible

Tag	S7 data type	Address	Comment
Counter_Actual_Value	Counter	C10	BCD coded counter value
Counter_Setpoint_Value	Counter	DB10.DBW200	BCD coded counter value
Counter_Setpoint_Value#2	Counter	MW20	BCD coded counter value

WinCC

Tag	S7 data type	Address	Coding	Comment
Counter_Actual_Value	Counter	%C10	<Standard>	BCD coded counter value
Counter_Setpoint_Value	Word	%DB10.%DBW200	SimaticBCDCounter	BCD coded counter value
Counter_Setpoint_Value#2	Word	%MW20	SimaticBCDCounter	BCD coded counter value

Mapping of the data type "StringChar"

In WinCC there is no corresponding data type to which the "StringChar" data type can be mapped. Mapping in WinCC depends on the property "Length" of the S7 data type.

A tag of the "StringChar" data type with the "Length" property > 1 is migrated to an array of the S7 data type "Char". The length of the array corresponds to the length of the originally configured data type "StringChar".

If the property "Length" = 1, the data type in WinCC is migrated to an array of the S7 data type "Char" with length = 1. The expression for an array with an element is "Array[0 ..0] of Char".

Mapping of the S7 data type "Timer"

An external tag with the S7 data type "Timer" with timer address is mapped to the S7 data type "Timer". The address will be retained.

If an external tag with the S7 data type "Timer" addresses a data block or a bit memory address, it is is mapped to the S7 data type "S5 Time". The address will be retained.

Example:

WinCC flexible

Tag	S7 data type	Address	Comment
Timer_Actual_Value	Timer	T10	BCD coded timer value
Timer_Setpoint_Value	Timer	DB10.DBW200	BCD coded timer value
Timer_Setpoint_Value#2	Timer	MW20	BCD coded timer value

WinCC

Tag	S7 data type	Address	Comment
Timer_Actual_Value	Timer	%T10	BCD coded timer value
Timer_Setpoint_Value	S5Time	%DB10.%DBW200	BCD coded timer value
Timer_Setpoint_Value#2	S5Time	%MW20	BCD coded timer value

See also

Migrating data types (WinCC flexible) (Page 75)

Migrating data types of Telemecanique Uni-Telway (WinCC flexible)

Migrating data types Telemecanique Uni-Telway

The Telemecanique Uni-Telway communication driver is not supported by WinCC, the data types are mapped to the internal data types of WinCC. The data types of the Telemecanique Uni-Telway communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Int	Int
+/-Long	DInt
ASCII	WString
Bool	Bool
Float	Real
Int	UInt
Long	UDInt

See also

Migrating data types (WinCC flexible) (Page 75)

2.8.2 Migrating WinCC V7 projects

2.8.2.1 Basics (WinCC V7)

Migration (WinCC V7)

Supported versions

You can migrate projects from WinCC V7.4 SP1 to WinCC V15.1.

You cannot migrate projects from earlier WinCC versions directly. If you want to continue using such projects, you need to first migrate them to WinCC V7.4 SP1.

1. Open the project in WinCC V7.4 SP1.
2. Confirm the query for the migration or convert the project data in the WinCC Explorer using the "Options" menu command. All objects must be converted to the current version.

Projects which can be migrated without constraints

The following project types are supported:

- Single-user projects
- Multi-user projects

Projects from WinCC V7 are migrated to WinCC Runtime Professional devices.

Migrating projects with redundant systems

When a redundant system is configured in WinCC V7, the configuration of the redundant partner is migrated.

Projects which cannot be migrated

It is not possible to migrate WinCC V7 projects with the following devices or functions:

- Multi-Client
- CAS
- Connectivity Station
- Maintenance Station

The migration is terminated and an alarm is generated.

See also

Basics on migration (WinCC V7) (Page 90)

Migrating projects from WinCC V7 (WinCC V7) (Page 91)

Starting Simulation and Runtime (WinCC V7) (Page 94)

Changing of the project name by the migration (WinCC V7) (Page 94)
Object support in the migration (WinCC V7) (Page 96)
Migration of connections (WinCC V7) (Page 99)
Migration of tags (WinCC V7) (Page 101)
Migration of user objects (WinCC V7) (Page 108)
Migration of alarms and alarm groups (WinCC V7) (Page 114)
Migration of functions and actions (WinCC V7) (Page 116)
Basics on migration of logs (WinCC V7) (Page 119)
Copying runtime data (WinCC V7) (Page 126)
Migrating data types (WinCC V7) (Page 126)
Filtering the project languages for the migration (Page 95)

Basics on migration (WinCC V7)

Introduction

During migration the project data are converted from a WinCC V7 project into the new data format of WinCC. The data will not be evaluated to see if they are consistent in the project you want to migrate. If the source project contains errors, these will not be remedied in the migration. This means that the project to be migrated should be free from errors prior to migration. Note the scope of a project during migration. The features of WinCC apply for migration. For additional information, please refer to the online help in "Process visualization > Performance features > Device performance features > WinCC Runtime Professional (RT Professional)".

Note

Use of the characters "ss" and "ß" in the TIA Portal

The names of the HMI objects (tags, screens, recipes, connections, text lists etc.) must be unique in the TIA Portal. No distinction is made between the characters "ss" and "ß". It is therefore not possible to use both character strings in otherwise identical names of the HMI objects for differentiation.

Requirements for the migration

Several basic requirements need to be considered in order to perform the migration.

- The project to be migrated must have been saved in WinCC V7.4 SP1.
- Make sure that the project to be migrated is not open, neither in WinCC 7 nor in Runtime.
- There must be enough free space on the PC hard disk for the migration. During migration, the requirement for hard-disk memory can amount to 10 times the size of the WinCC V7 project without runtime data.

Canceling migration

The migration is canceled in the following cases:

- If the project to be migrated is opened in the engineering system or in Runtime.
- If not enough memory space is available on the hard disk to create a copy for migration of the project.
- If the migration cannot address the project database due to problems with the installed SQL-Server.
- If the migration cannot address the project database due to missing user authorization.
- If the project was created with a version not supported by the migration.
- If a control which is not registered in the operating system is used in the project.

Preparing project with the migration tool

You do not have to execute the migration of a WinCC V7 project on the PC on which WinCC is installed.

You can use the migration tool to prepare the migration on a PC on which WinCC V7.4 SP1 is installed. The migration tool exports the engineering data from the WinCC V7 project and saves the data in the format "*.am".

For the actual migration, copy the file generated by the migration tool to a PC on which the TIA Portal is installed.

See the section "Preparing project with the migration tool" for more information on the migration tool.

See also

Migration (WinCC V7) (Page 89)

Migrating projects from WinCC V7 (WinCC V7)

Introduction

Data from a WinCC V7 project are copied to a new project for WinCC when migrating a project. A new project is therefore created automatically for project migration. You cannot migrate to an existing project.

The migration can be started in both the Portal view and the Project view.

Requirement

- A project from WinCC V7.4 SP1 is available.
- The project is not opened in WinCC V7.

Procedure

Migrate a project in the Portal view as follows:

1. Select the action "Start > Migrate Project".

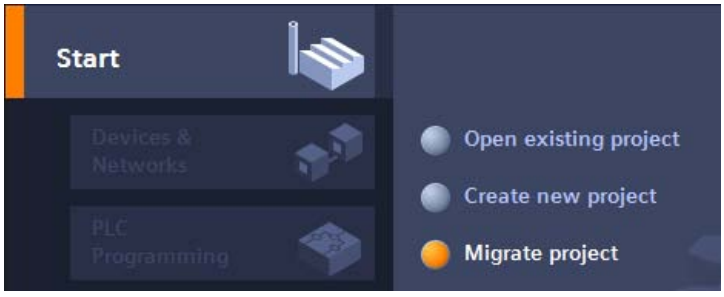

2. In the "Source path" box, navigate to the project you want to migrate.
3. Select the WinCC V7 project file "*.mcp".
4. Change the information for the project to be created, if necessary. For example, change the project name or project path. The data to be migrated is created in the new project.
5. To migrate all of the runtime data of the project, select "Copy WinCC Runtime Professional data".
   These runtime data are visible only after they are loaded on the target device.
6. Click "Migrate".
   A new project is created and the migration of the data is started:
   - The Project view opens.
   - The progress of the migration is shown in a migration window.
   - Warnings and errors about the migration process are displayed in the Inspector window under "Info > General".
   - All information about the migration is saved in a log file.
   - The project is saved and a message displayed upon completion of the migration. The message contains a link that you can use to open the log file.

After the migration you can find a newly created device in the project tree for every migrated PC system. These devices contain the migrated data, such as screens, alarms and tags. In a multistation project, a device is created for every Client and the Server in the project tree of WinCC.

Once the project is successfully migrated, a notice appears reporting the success.

Opening the migration log at a later point in time

The migration log is saved together with the migrated project. To view the log at a later point in time, proceed as follows:

1. Open the "Common data > Logs" folder in the project tree.
2. Double-click the migration log. The log is opened.

Note

Alarms as tree structure

As of TIA Portal V13.0 alarms are sorted into a tree structure in the migration log. This means all alarms that are part of a specific subsystem are stored in one folder. Press the shortcut <NUM + x(Multiplication)> to expand the currently selected folder and all subfolders in one single step.

See also

Migration (WinCC V7) (Page 89)

Adapting a project (WinCC V7)

Causes:

You must adapt a project when you migrate it. The required post-configuration is caused by:

- Messages that occur during the migration and are entered in the migration log.
- Data which was inconsistent or invalid in the project to be migrated. WinCC in the TIA Portal has stringent criteria for checking, for example, the validity of the tag addresses or the length of object names. Invalid information is not corrected by the migration.
- Folders and files you have created in the WinCC V7 project directory are not copied to the new project directory during the migration. You must adapt all scripts that access such directories and files after the migration.

Criteria for messages during the migration

Messages are issued in the migration log during the migration:

- If a property was changed by the migration.
- If a property has the state "Invalid" after the migration, because a tag is outside the value range, for example, no message is issued in the migration log. You can recognize the "Invalid" state in the Engineering System, by the entry fields highlighted in color, for example. When you compile a project, messages for the invalid information are output in the inspector window.

Adapting a project

1. Edit the messages in the migration log.
2. In the project tree, click on the migrated project.
3. Select the "Compile > Software" command in the shortcut menu.
4. Edit the error messages in the Inspector window:

Starting Simulation and Runtime (WinCC V7)

Start simulation

After error-free compilation of the project, you can simulate the Runtime on a PC with installed engineering system. This gives you the opportunity to check the functionality of the necessary changes in the project after the migration without having to download the project to the target system. To do so, select the "Online > Simulation > Start" menu command.

Load the project on a PC with WinCC Runtime Professional.

WinCC also provides the option of transferring the project to a PC on which only WinCC Runtime Professional is installed. To do this, you must load the project onto the relevant target computer. No information for loading onto another target computer is available in a WinCC V7 project. The migration cannot therefore migrate any data for communication with another target computer. To load a migrated project onto another target computer, you must configure the necessary settings in WinCC.

See also

Migration (WinCC V7) (Page 89)

Changing of the project name by the migration (WinCC V7)

Introduction

If you have migrated a WinCC V7 project, the migrated project is displayed in the WinCC project tree as a new device with the name of the WinCC V7 project.

When you compile the migrated project, WinCC renames the Runtime project and the folder that contains it. This prevents the path length of 256 characters allowed in Windows from being exceeded. The Runtime project therefore has a different name to the device in the project tree.

Starting Runtime on the target system

You can ignore the new name when you simulate Runtime on the configuration PC. Open the project in WinCC, and start the simulation using the menu command "Online" > "Simulation" > "Start".

The following rules apply if you want to start the Runtime on a PC on which no WinCC engineering system is installed.

- You must use the menu command "Extended download to device" for the first loading process.
- WinCC Runtime creates the "WinCCProjects" authorization directory during the installation. In the "Extended loading" dialog, you enter the path for the authorization directory on the target device in the "Target path" field. Alternatively you navigate to the authorization directory with the navigation button.

- When you subsequently load the project, it is saved in the "WinCCProjects" authorization directory.
- If you want to start the project, use the "WinCC StartTool".

Adapt scripts

If you have created an executable program which references a migrated project by its name, you must adapt this script accordingly. The new project name is the sequence of numbers in curly brackets in the path specification of the MCX file described above.

See also

Migration (WinCC V7) (Page 89)

Filtering the project languages for the migration

Introduction

For the optimal migration of projects, project languages not listed in the text library and not having texts that deviate from the default value are automatically filtered out of an existing WinCC V7 project. This project languages are not migrated.

Criteria for filtering during migration

The filtering applies only to project languages that are not explicitly contained in the text library.

If a project language is not in the text library and the texts contain only the following text strings, the project language is not migrated:

- Text string with six question marks
- Empty text strings
- Text strings with automatically generated content "text"
- Text strings for check boxes or option buttons, starting with six question marks or automatically generated content "text" and ending with a number, e.g. "??????5" or "Text5"

Note

Project languages that are used on control have no influence on the filter behavior.

Note

If a project language is not migrated, an alarm is displayed in the migration log.

See also

Migration (WinCC V7) (Page 89)

Migrating texts in Spanish (WinCC V7)

Texts in Spanish are migrated differently depending on their sort.

Migrating texts with traditional and international sort

If a WinCC V7 project includes texts in Spanish (traditional sorting), these texts are migrated as Spanish (Spain).

If a WinCC V7 project includes texts in Spanish (international), these texts are migrated as Spanish (Spain).

If a WinCC V7 project includes texts in Spanish (international sorting) as well as Spanish (traditional sorting), only the texts from Spanish (traditional sorting) are migrated as Spanish (Spain). The texts from Spanish (International Sort) are not taken into consideration.

2.8.2.2 Migrating engineering data (WinCC V7)

Object support (WinCC V7)

Object support in the migration (WinCC V7)

Introduction

When migrating projects from WinCC V7, all object types and functions that are available and can be mapped to the new project environment will be fully migrated.

Supported object types

The following object types are supported for migration:

- Screens
- Display and operating objects
- Faceplates
- Tags
- Data types
- Structures
- Connections
- Cycles
- Alarms
- Alarm classes
- Alarm groups
- Alarm types
- User logs

- Alarm logs
- Data logs
- Reports
- Print jobs
- Layouts
- Project functions
- Standard functions
- Actions
- User administration
- Texts
- Project languages
- Runtime languages
- Runtime scripting
- ODK functions of the RT

Constraints and conditionally supported objects are described in the following sections.

Unsupported object types

The following object types are not supported by the migration:

- 3D bar
- Screen hierarchy
- Extended analog display
- Extended status display
- Inserted layout
- Global libraries
- Project library
- Acoustic alarms
- Group display
- Server to server communication
- XAML Controls, e.g. WPF Controls
- ODK functions of the CS
- Default screens of the PCS7 Standard/Advanced libraries

Unsupported languages

Over 200 languages and dialects are available for configuration in WinCC V7. Approx. 100 languages and dialects are supported in the TIA Portal, which are listed in the "Project languages" editor under "Languages&Resources". All the texts of a WinCC V7 project in languages not supported in the TIA Portal are lost during migration. Appropriate alarms are output in the migration log.

The programmed Runtime language changes in the languages which are not supported are not valid in the TIA Portal project. You must remove these language changes before commissioning the project. If a WinCC V7 project only contains texts in languages which are not supported, no texts are migrated. In this case, the default language "English (USA)" is set as a substitute in the "Project languages" editor under "Languages&Resources".

Prohibited characters

WinCC does not support exactly the same character set for object names as WinCC V7. For example, the following characters are prohibited for tags in WinCC: ? " %

Object names are not changed by the migration, even if the name contains a character which is not permitted in WinCC. Unallowed characters will make the object name invalid and an alarm will be output during compilation. In WinCC, you can change object names centrally. All other occurrences are always changed as well.

See also

Changed objects (WinCC V7) (Page 98)

Migration (WinCC V7) (Page 89)

Migration of user objects (WinCC V7) (Page 108)

Changed objects (WinCC V7)

Introduction

Some objects which are available in WinCC V7 are no longer supported by WinCC. The migration maps these objects to the objects available in WinCC. Since not all properties can be mapped, rework is partly necessary. Changed and non-migrated properties are recorded in the migration report.

Changed objects

The following objects are mapped to other objects or partly have different properties:

- OLE object
- Push Button Control
- Text list
- Status display
- WinCC Controls before WinCC V7

- SVG graphics
- User objects
- Faceplates
- Object groups
- Dynamizations

See the section Screens and screen objects (WinCC V7) (Page 103) for additional information.

See also

Object support in the migration (WinCC V7) (Page 96)

Screens and screen objects (WinCC V7) (Page 103)

Connections (WinCC V7)

Migration of connections (WinCC V7)

Introduction

The following cases must be distinguished in the migration of connections:

- Communication driver is supported in WinCC.
- Communication driver is not supported in WinCC.

See the following description for how to migrate the individual communication drivers.

Migration with supported communication driver

If WinCC supports the used communication driver, the connection is migrated with this communication driver. The following communication drivers are supported:

WinCC V7	WinCC
SIMATIC S7 PROTOCOL SUITE	SIMATIC S7-300/400
SIMATIC S7-1200	SIMATIC S7-1200
SIMATIC S7-1500	SIMATIC S7-1500
ALLEN BRADLEY - ETHERNET IP	Allen Bradley Ethernet/IP
MITSUBISHI ETHERNET	Mitsubishi MC TCP/IP
MODBUS TCPIP	Modicon MODBUS TCP/IP
OPC	OPC
OPC UA DA	OPC UA DA

The "Industrial Ethernet (I + II)" channel units of the S7 Protocol Suite are migrated to the Industrial Ethernet channel unit of the SIMATIC S7-300/400 driver.

The "PROFIBUS (I + II)" channel units of the S7 Protocol Suite are migrated to the PROFIBUS channel unit of the SIMATIC S7-300/400 driver.

The "Named Connection" channel unit will not be migrated.

Migration of an unsupported communication driver

Not all communication drivers which are available in WinCC V7 are supported in WinCC. If a communication driver is not supported, the configured connection is not migrated.

If you migrate a WinCC V7 project which includes a connection that is not supported, all external tags of this connection are mapped to internal tags in a tag table. The PLC address is stored as comment with the tag. You receive a tag table and an entry in the migration log for each affected connection.

You must reconnect the converted tags with the PLC after the migration and configuring of a new connection.

You get an appropriate warning if the driver is not migrated.

The following communication drivers are not supported by WinCC:

WinCC V7	WinCC
Profibus DP	not supported
Profibus FMS	not supported
SIMATIC TI Serial	not supported
SIMATIC 505 TCP IP	not supported
SIMATIC S5 Ethernet Layer 4	not supported
SIMATIC S5 Ethernet TF	not supported
SIMATIC S5 Profibus FDL	not supported
SIMATIC S5 Programmers Port AS 511	not supported
SIMATIC S5 Serial 3964R	not supported
SIMATIC TI Ethernet Layer 4	not supported
SIMATIC TI Serial	not supported
System Info	not supported
Windows DDE	not supported

Existing functions from Windows DDE can be converted to WinCC by means of OPC. You have to do the conversion yourself.

See also

Adaptation of the configuration for non-migrated connection (WinCC V7) (Page 101)

Migration (WinCC V7) (Page 89)

Migration of tags (WinCC V7) (Page 101)

Adaptation of the configuration for non-migrated connection (WinCC V7)

Introduction

If a connection cannot be migrated you have the following options:

- Change the configuration in WinCC V7 before the migration
- Change the configuration after the migration

Change a connection before the migration

If the communication driver selected for the connection is not supported by the migration, you must select a driver which is supported by the migration.

If no suitable driver is available, you must use suitable components for the communication. Then adapt the configuration in WinCC V7. Migrate the project after successful adaptation. The connection is then also migrated.

Changing a connection before the migration

If you migrate a WinCC V7 project which includes a connection that is not supported, all external tags of this connection are mapped to internal tags in a tag table. The PLC address is stored as comment with the tag. You receive a tag table and an entry in the migration log for each affected connection.

You must configure a new connection after the migration. For this connection you select a communication driver which supports the used communication partner. If no suitable driver is available for the communication partner, you must use suitable components for the communication.

If you have configured the new connection, you must reconnect all tags which were mapped to internal tags before the migration. Open the migration report to make reconfiguration easier. Reconfigure the tags according to the log entry.

See also

Migrating projects from WinCC V7 (WinCC V7) (Page 91)

Migration of connections (WinCC V7) (Page 99)

Migration of tags (WinCC V7)

Introduction

You need to make some special considerations when migrating tags. The following aspects should be distinguished:

- Migrating data types of tags
- Migrating internal tags
- Migrating external tags

Migrating data types

WinCC features the same data types but sometimes uses different data type names than WinCC V7. You will have to take certain special features into consideration during migration of external tags that contain data types of a SIMATIC S7 control. See the section Migrating data types (WinCC V7) (Page 126) for additional information.

Migrating internal tags

Internal tags are always migrated completely. Only the data type name can be changed by the migration. See the section Migrating data types (WinCC V7) (Page 126) for additional information.

Migrating external tags

If the connection used by the external tags is migrated, the external tags are also fully migrated. The migration of external tags depends on whether the used communication driver is supported by the migration. See the section Migration of connections (WinCC V7) (Page 99) for additional information.

If the connection used by the external tags is not migrated, the external tags are not migrated either. The external tags are then mapped to internal tags. You must configure a new connection and reconnect the tags with the tags of the PLC after the migration.

See also

Migrating data types (WinCC V7) (Page 126)

Migration of connections (WinCC V7) (Page 99)

Migration of structures (WinCC V7) (Page 102)

Migration (WinCC V7) (Page 89)

Migration of structures (WinCC V7)

Migration of structures

Structured tags are implemented in WinCC V7 by means of structure types and structure elements. The structure types are valid for the device in which they are created in WinCC V7. Structure types are created in the project library in WinCC and are valid for all devices of a project. The structure of the structure types in WinCC and WinCC V7 is identical.

See also

Migration of tags (WinCC V7) (Page 101)

Screens and screen objects (WinCC V7)

Migration of screens and screen objects (WinCC V7)

Introduction

The majority of screen objects available in WinCC V7 is applied without restrictions after migration to WinCC.

For some screen objects, such as text list and status display, improved options are available in WinCC. Through the migration these objects are converted in such a manner that no reworking is required.

Single screen objects such as WinCC Push Button Control are no longer supported by WinCC. The migration maps as far as possible these objects to the objects available in WinCC.

The following special features must be taken into account during the migration.

Newly introduced features since WinCC V7.3

The new features introduced since WinCC V7.3 are not migrated to WinCC. If one of the new properties is dynamic, an alarm is generated in the migration log about the non-migrated dynamic properties. In contrast, no message is generated for non-migrated static properties.

Tab sequences in screens with faceplates or object groups

In screens with faceplates or object groups, the migration causes tab sequences to change both in the screen and in the faceplate or in the object group.

Renaming events

Some events are assigned new names in WinCC; for example, the event "Mouse click" is called "Click" in WinCC. The migration converts this renaming completely. No reworking is necessary.

Migrating triggers

Triggers from WinCC V7 of the window cycle type are migrated to the screen update cycle valid in WinCC.

Migrating a text list

In WinCC V7, the "Text list" object enables the output of texts based on the value of a tag. For this, the output texts must re-specified for each object. In WinCC, a text list independent of the point of use is available. The "Symbolic IO field" object is used to display texts from a text list.

The texts used in WinCC V7 are migrated to a text list and connected with the symbolic IO field. The used texts are stored as project text. You can find project texts under "Languages & Resources" in the project tree.

Migration of a status display

In WinCC V7, the "Status display" object enables the output of graphics based on the value of a tag. For this, the output graphics must be re-specified for each object. In WinCC, a graphic list independent of the point of use is available. The "Graphic IO field" object is used to display graphics from a graphics list.

The screens used in the status display are migrated to a graphics list and connected with the graphic I/O field. The used screens are stored in the Graphics of WinCC. The Graphics can be found in the project tree under "Languages & Resources".

Migration of a faceplate

The faceplates in WinCC have several differences to the faceplates in WinCC V7. Therefore the following special considerations need to be made in the migration of a faceplate:

- A property of an inner object can only be interconnected to one property respectively of the faceplate in a faceplate. If a property of an inner object is interconnected to several properties of the faceplate, only the connection of the first configured property is migrated. An alarm is output in the migration report for the properties that are not migrated.
- If an event of an inner object is multiply bound to the event interface of the faceplate, only the last event is connected to the inner object after the migration. The other events of the event interface are migrated to the inner object without a connection. If dynamizations are configured on these events, the events are migrated. Binding of the dynamization to the inner objects will be lost.
- If an object property is dynamized within a faceplate type, and this property is simultaneously used at the event interface of the faceplate, the connection to the event interface is deleted during migration. The inner dynamization is migrated. The property at the event interface is also migrated, only the connection to the property of the inner object is lost.
  Example:
  A faceplate type has the inner object "Circle". The "Background color" property of the circle is dynamized with an inner event, e.g. by a faceplate tag. The "background color" property of the circle is passed out to the event interface of the faceplate and is additionally dynamized externally in the faceplate instance. This configuration is no longer allowed in WinCC. The inner dynamization is migrated in the migration. The external dynamization and the property at the event interface of the faceplate are also migrated. The connection between the external property and the inner "Background color" property of the circle is resolved.
- If a faceplate type contains an I/O field, its "Output format" property cannot be bound to the event interface.
- For faceplates in WinCC V7, internal tags can be interconnected directly with the interface. This is not possible with faceplates in WinCC. To achieve the same behavior in faceplates, an invisible I/O field is generated as an auxiliary object through migration. Two VB functions are generated at this I/O field for each internal tag and interconnected with an interface property. You reach the internal tag by means of this interface property. However, only a maximum of 23 tags interconnected with the interface can be migrated.

Migrating an OLE object

The OLE object is not supported in WinCC. The functions of the OLE object are mapped by the migration to a graphic display. An alarm is output in the migration report.

If the OLE object is connected to a text-oriented file, such as an Excel table, a screenshot of the Excel table is generated by the migration and connected with the graphic display.

Migration of a WinCC Push Button Control

WinCC Push Button Control is not supported in WinCC. The functions of the Push Button Control are mapped to a button by the migration.

The following properties of the Push Button Control are not available on a button and are therefore not migrated:

- Outline
- FontStrikeThru
- FocusRect
- AutoSize
- Transparent

If you have configured a dynamic on one of these properties, the dynamization will be lost in the migration. If a property or a dynamic of the object cannot be migrated, a corresponding entry is made in the migration log.

Since a button supports other events, the following special features have to be taken into account for the migration:

- The event "Click" of the Push Button Control is migrated without constraints to the "Click" event of the button.
- The events "KeyDown" and "KeyUp" of the Push Button Control are migrated to the events "Press key on keyboard" and "Release keyboard key". However, the parameters cannot be uniquely mapped for this. If you use these parameters in a script, the script will cause an alarm during compilation.
- The event "MouseDown" of the Push Button Control is migrated to the two events "Press left mouse button" and "Press right mouse button". The event "MouseUp" of the Push Button Control is migrated to the two events "Release left mouse button" and "Release right mouse button". However, the parameters cannot be uniquely mapped for this. If you use these parameters in a script, the script will cause an alarm during compilation.
- It is not possible to migrate the events "KeyPress" and "MouseMove" of the Push Button Control. The configuration on the two events is lost during the migration. An alarm is output in the migration report.

See also

Migration of user objects (WinCC V7) (Page 108)

Changes of values of object properties by the migration (WinCC V7) (Page 106)

Changes of values of object properties by the migration (WinCC V7)

Introduction

The standardization of object properties from WinCC V7 and WinCC flexible requires changes to the object properties during the migration process. The migration calculates the changes in such a way that the representation of the objects after migration is the same as prior to migration. Changes made during migration result in different units of measurements and values in the configuration for some object properties.

Migrating the font settings of an object

In WinCC V7 and WinCC, various units of measurement are used in some cases to denote the size of the fonts used. During migration, the font size is converted to ensure that the representation of the font is the same size at zoom level 100%. Due to the different units of measurement, the numerical values for the font sizes are changed after migration.

Example:

Font style before migration	Font style after migration
Arial 10 points	Arial 13 pixels
Arial 16 points	Arial 21 pixels
Tahoma 10 points	Tahoma 13 pixels
Tahoma 16 points	Tahoma 21 pixels

Migrating the size data of an object

In WinCC V7, position data and size data are specified for objects of the Report Designer in accordance with the default setting in Report Designer, for example, in cm, inch or points. In WinCC, position data and size data are specified in pixels. During migration, the values for the size data are converted so that the representation of the objects is identical prior to and after the migration. The conversion produces different numerical values for the data in the object properties after the migration. The object properties affected are, for example, X position, Y position, width and height.

See also

Migration of user objects (WinCC V7) (Page 108)

Migration of screens and screen objects (WinCC V7) (Page 103)

Migration of object groups (WinCC V7)

Object group with objects at several levels

In WinCC V7 the objects of an object group may be in different layers of the screen and are subject to the layer administration of the screen. The object group itself has no layer administration. In WinCC V7 it is therefore possible to position an object in a layer between two objects of an object group. The Z order applies for all objects in the screen. Therefore it is possible to hide an object in an object group in WinCC V7 by deactivating a layer.

In WinCC, an object group and the objects that it contains are in the same layer of the screen. An object group has its own layer administration in WinCC. The Z order of the objects only applies within the object group in WinCC. If an object is positioned above an object group and a layer is moved back, this object is behind the object groups and is no longer visible. The object is not included in the Z order of the object group either. Therefore it is possible to hide an object in an object group in WinCC by deactivating a layer. If the migration cannot fully map the layout of an object group, an alarm is output in the migration log. Check whether the object group is displayed correctly after the migration.

Object group with objects that are not supported

An object group in WinCC consists of at least 2 objects. If WinCC does not support an object of a group, the group is still migrated. The unsupported object is not migrated and is missing from the group after the migration. If the object groups only contains 2 objects before the migration and one of them is not migrated, an invalid object group results. The object group is marked as an invalid object in WinCC. Select the invalid object and cancel the grouping. The valid object is displayed.

If not all objects in a group can be migrated, the coordinates and the size of the group are recalculated and adapted accordingly.

An entry is made in the migration report in both the above cases.

Migration of properties of an object group

Static values and dynamizations of properties of grouped objects are migrated without constraints.

Object groups in WinCC V7 have their own properties: X-position, Y-position, width, height, object name, level, tooltip text, authorizations, operator control enable and visibility. Static values and dynamizations of these properties are migrated without constraints.

In addition to the above properties, an object group also has a collection of all properties of the grouped objects. These collected properties can be used, for example, to influence the background colors of all grouped objects in WinCC V7. Changes are stored both on the object group and on the grouped objects. The changes to the object group are not migrated. The dynamizations on the collected properties are lost during the migration.

See also

Migration of user objects (WinCC V7) (Page 108)

Migration of user objects (WinCC V7)

Introduction

User objects are no longer supported in WinCC. The migration maps user objects to faceplates. Identical user objects are grouped to a faceplate type. Each user object is replaced by a faceplate instance.

User objects

A faceplate does not offer exactly the same characteristics as a user object, therefore the following special considerations must be made in the migration:

- A property of an inner object can only be interconnected to one property of a faceplate in a faceplate. If a property of an inner object is interconnected to several properties of the faceplate, the connection to the first property is migrated. An alarm is output in the migration report for the properties that are not migrated.
- The coordinates for the position of an object within the user object cannot be interconnected to the properties of a faceplate. The property of the user object on which a position is configured is migrated. If a dynamic is configured on this property, this is also migrated. However, the property and the dynamic in the faceplate are no longer interconnected with the object position of the inner object.
- Faceplates cannot be rotated or mirrored. If the project contains rotated or mirrored user objects, a message is entered in the migration report.
- If a user object contains objects which are not supported by the migration, these objects are missing in the resulting faceplate after the migration. The faceplate then only contains the objects supported by the migration. If a dynamic is configured on an unsupported object, this dynamic will be lost. If an inner object of a user object is not migrated, the size of the resulting faceplate may change. The object position of the user object is retained and applied to the faceplate.
- The position of an inner object is related to the coordinate system of the screen in which the user object is positioned in the user project. A faceplate has its own coordinates system which is independent of the screen in which it is positioned. The coordinates of the inner objects therefore change in the migration. If the project contains a script which uses or changes the coordinates of an inner object, you must adapt the script accordingly. Changing of the coordinates of an inner object is not logged.
- Use of the "Connector" object is not allowed in a faceplate. Therefore a connector is not migrated within a user object.
- Use of WinCC Controls is not allowed in a faceplate. Therefore a control is not migrated within a user object. The following controls are exceptions; these are also migrated as inner objects of a user object:
  - WinCC Digital/Analog Clock Control
  - WinCC Gauge Control
  - Slider Object

If the migration makes changes to the configuring data due to these described special considerations, these are entered in the migration report.

See also

Migration (WinCC V7) (Page 89)
Object support in the migration (WinCC V7) (Page 96)
Migration of screens and screen objects (WinCC V7) (Page 103)
Migration of object groups (WinCC V7) (Page 107)
Migration of dynamizations by Dynamic-Dialog (WinCC V7) (Page 109)
Migration of dynamizations by script (WinCC V7) (Page 110)
Migration of controls (WinCC V7) (Page 112)
Migrating color settings (WinCC V7) (Page 113)
Migration of a dynamization by direct connection (WinCC V7) (Page 111)
Changes of values of object properties by the migration (WinCC V7) (Page 106)

Migration of dynamizations by Dynamic-Dialog (WinCC V7)

Introduction

The Dynamic-Dialog is available in WinCC V7 for the animation of library objects. The functions of the Dynamic dialog are mapped as animation of a property in WinCC. This animation offers you similar dynamization functions to the Dynamic dialog in WinCC V7. If you have configured a dynamization to a property in a WinCC V7 project, this dynamization is converted into an animation of WinCC by the migration.

Note

The new features introduced since WinCC V7.3 are not migrated to WinCC. If one of the new properties is dynamic, an alarm is generated in the migration log about the non-migrated dynamic properties.

Migration of dynamizations by Dynamic dialog

The expression of a dynamization which is created using the Dynamic-Dialog from WinCC V7 is based on the syntax of C-script. The migration attempts to convert the expression of a dynamization from WinCC V7 to the syntax of VB script. This form of conversion can generally be performed if the dynamization uses the following C operators:

- Arithmetic operators [+, -, *, /, %, ^]
- Logical operators [&&, ||]
- Comparative operators [<, >, !=, <=, >=, ==]
- Numerical constants

If the conversion to the syntax from VB script is not possible, the dynamization is mapped to a C script. Mapping to a C script takes place in the following cases:

- If the expression of the Dynamic dialog uses no tag or cyclic trigger. If a tag and a trigger tag are used, these must match. No cyclic trigger is allowed when using tags in the expression.
- If the result of the expression in the Dynamic dialog is an "Analog" data type and the range of validity "other" in the "Result of expression" field has a different value to the "Static" value of the dynamized property.
- If several tags are used in the expression for the Dynamic dialog and not all tags have the same configuration for the Quality Code.
- If a tag is used whose tag status is used.
- If the expression of the Dynamic dialog cannot be converted to an equivalent in VB script.

See also

Migration of user objects (WinCC V7) (Page 108)

Migration of dynamizations by script (WinCC V7)

Introduction

The migration executes various measures to ensure that the VB scripts and C scripts from WinCC V7 remain functional: These measures above also apply to actions on events.

Migration of VB scripts for the dynamization

In WinCC V7 you can use several functions in a dynamization by VB-script, this is no longer allowed in WinCC. If the migration finds several functions in a script, the function which is called when triggering the event is migrated. The other functions are migrated in the declaration area. This leads to an alarm during the compiling of the project.

In WinCC the function names of VB-scripts sometimes change in relation to WinCC V7. The function name "LeftTrigger" changes to "LeftValue" for example. If the function has to return a value, the source code contains a statement in which the value is assigned to the function name. The migration finds and replaces the function name in the function and the function header. If the new function name is already used as the name of a function, tag or parameter, an alarm is output in the migration log. Adapt the configuration yourself after the migration.

Migration of C scripts for the dynamization

In WinCC V7 some parameters for C-scripts have different names than in WinCC. To keep a script functional, the migration inserts a tag with the previous parameter name for every changed parameter name.
Example:
The parameter name "lpszPictureName" is replaced by the parameter name "ScreenName".
The migration adds the following tag to the script: "char* lpszPictureName = ScreenName".
If a tag or parameter with this name already exists, an alarm is generated and entered in the migration log.
The following parameters are affected:

WinCC V7	WinCC
lpszPictureName	ScreenName
lpszObjectName	ObjectName
lpszPropertyName	PropertyName

In WinCC the function names of C-scripts change in relation to WinCC V7. The function name of a dynamic is given the expression "_main". In WinCC the expression changes to <Name of property>_value. The migration corrects the names in the same way as it changes the function names of VB scripts described in the previous paragraph.

Migrating triggers

Triggers from WinCC V7 of the window cycle type are migrated to the screen cycle valid in WinCC.

See also

Migration of user objects (WinCC V7) (Page 108)

Migration of a dynamization by direct connection (WinCC V7)

Migration of a direct connection

A dynamization by direct connection is migrated as a function list.

See also

Migration of user objects (WinCC V7) (Page 108)

Migration of controls (WinCC V7)

Introduction

A WinCC V7 project may contain the following controls:

- WinCC controls prior to V7
  The controls prior to WinCC V7 are still supported in WinCC V7. These controls are not supported in WinCC. If a project contains controls prior to WinCC V7, these controls are migrated to the version as of WinCC V7. The names of properties, buttons or toolbars can also change in the process.
  Changes can occur in the display in runtime.
  Scripts that acted on controls prior to WinCC V7 have to be adapted after migration.
- WinCC Controls as of V7
  The controls as of WinCC V7 listed below are supported by WinCC.
- Controls of third parties
  The migration also supports ActiveX controls and simple .Net controls from third parties. XAML controls, for example, WPF (Windows Presentation Foundation) controls, are not supported. The ActiveX controls must be registered in the operating system. If an ActiveX control is not registered, the migration is terminated.
  When you save a project with the migration tool and perform the migration itself on another PC, the ActiveX controls must be registered on this PC.

Supported WinCC Controls

WinCC supports the following controls from WinCC V7:

- Siemens HMI Symbol Library 1.4.1
- WinCC Channel Diagnosis Control
- WinCC Digital/Analog ClockControl
- WinCC Gauge Control
- WinCC Slider Control
- WinCC Ruler Control
- WinCC Media Control
- WinCC AlarmControl
- WinCC FunctionTrendControl
- WinCC OnlineTableControl
- WinCC OnlineTrendControl
- WinCC UserArchiveControl
- CCDiskSpace.Diskspace
- WinCC_WB.WinCCWebBrowser

Special features of WinCC OnlineTableControl

In WinCC V7, WinCC OnlineTableControl can display exceeded values with different colors. For this, at least two high and two low limits and associated colors can be configured.

WinCC does not support these properties. An alarm is output in the migration log, when the corresponding values are configured.

Special features of WinCC AlarmControl

The new "Class Priority" alarm text block in WinCC AlarmControl from WinCC V7 is not supported by WinCC. If the alarm_text_block was configured, an alarm is output in the migration log.

Unsupported WinCC Controls

The following controls are not supported by WinCC:

- WinCC push-button control
  The WinCC push-button control is mapped to a button in WinCC. See Migration of screens and screen objects (WinCC V7) (Page 103) for additional information.
- XAML controls, e.g. WPF controls
  If a project contains XAML controls, these controls are not migrated. An alarm is output in the migration report for every control that is not migrated.

See also

Migration of user objects (WinCC V7) (Page 108)

Migration of screens and screen objects (WinCC V7) (Page 103)

Migrating color settings (WinCC V7)

Migrating the color settings of an object

In WinCC V7 it is possible to configure the color settings of objects by the central color palette. The central color palette is omitted in WinCC. The migration takes over the color settings for an object from the central color palette and defines the colors directly on the respective object. The color settings can no longer be changed centrally in the migrated project.

See also

Migration of user objects (WinCC V7) (Page 108)

Migration of alarms and alarm groups (WinCC V7)

Introduction

The alarm system of WinCC clearly differs in structure from the alarm system of WinCC V7.

WinCC V7 offers up to 18 alarm classes with 16 alarm types per alarm class.

WinCC only offers alarm classes but no alarm types.

In the migration of a project the alarm types are converted into alarm classes in the migrated project. The alarm types are omitted completely.

Migration of alarm groups

The alarm classes and alarm types are migrated according to the following scheme:

The migration assigns IDs for all alarm classes and their alarm types throughout. The IDs 1-63 are reserved for the system side in WinCC. The migration begins assignment of the IDs at ID 64.

Alarm class WinCC V7	Alarm type WinCC V7	Alarm class WinCC
ID 1	ID 1-16	ID 64-79
ID 2	ID 17-32	ID 80-95
...	...	...
ID 16	ID 241-256	ID 304-319
ID 17	ID 257-272 257-258 of which are used	ID 320-335
ID 18	ID 273-288 273-274 of which are used	ID 336-351

The migration computes new IDs for all alarm classes and alarm types which are contained in the project to be migrated.

After assignment of the IDs, the alarm classes and alarm types existing in the project are migrated. To ensure unique names for the alarm classes, the migration links the names of the alarm classes with the alarm types contained therein.

Example:

The alarm class "Tolerance" with alarm type "ToleranceHigh" exists in WinCC V7. This becomes alarm class "Tolerance_ToleranceHigh" after the migration.

With the exception of scripts and ODK functions, all application points in the project are adapted accordingly. For example, all static filters of an alarm view are adapted to the new alarm class.

The migration does not correct any scripts or ODK functions which process an alarm type. If the migration detects the use of an alarm type in a script, an alarm is output in the migration log. Adapt such a script yourself after the migration.

Migration of alarms

In WinCC V7, system alarms have IDs from the 1000000 to 1019999 range. However, alarms can also be freely configured in this number range.

In WinCC, the 1000000 to 1019999 range is reserved for system alarms.

WinCC V7 system events are migrated. User-defined alarms with IDs in this range are not migrated.

If the user alarms contain non-supported ASCII characters, each non-supported ASCII character is replaced by the following expression: "#<ASCII-Code>".

Logging Settings

In WinCC V7 you can define the settings for the logging for every alarm separately. In WinCC you can only define the settings for the logging for an alarm class and not for single alarms. If the logging is activated on an alarm, the migration accepts this setting for the entire alarm class. This ensures that no logging settings are lost. Then all alarms which are contained in this alarm class are logged after the migration even if they were not logged before the migration. An alarm is output in the migration report for each affected alarm.

See also

Migration (WinCC V7) (Page 89)

Migration of user logs (WinCC V7) (Page 115)

Migration of user logs (WinCC V7)

Migration of user logs

User logs from WinCC V7 are called recipes in WinCC. Data fields of a user log are called recipe elements.

The configuration data are mapped directly to recipes during the migration of user logs. Views are mapped to recipe queries. You have to observe the following rules in the migration:

- If you have configured a relation in a view, this relation may not be "NULL".
  If the relation results in "NULL", an error message is generated when the migrated project is compiled. To remedy the error, you must reconfigure the relation in the recipe view so that a valid relation results.
- In WinCC V7, control tags can be configured independent of the specifications for the communication. Control tags are only migrated if WinCC tags are used for communication.
- In WinCC V7, the PLC ID is used as unique name of the log for communication via raw data tags. In WinCC, the recipe number is used to uniquely identify a recipe. Information on PLC ID is not migrated.
- If an alias is configured in WinCC V7 for a user log, this is migrated to the display names of the recipe.

The Runtime data of user logs can be copied during migration.

See also

Copying runtime data (WinCC V7) (Page 126)

Migration of alarms and alarm groups (WinCC V7) (Page 114)

Scripts and standard functions (WinCC V7)

Migration of functions and actions (WinCC V7)

Introduction

Like WinCC V7, WinCC also supports both ANSI-C and VBS. Some things need to be considered for the migration of functions and actions due to the system-side changes. The basic conditions for migration are described below.

Migration of functions and actions

Note the following points for the migration of functions and actions:

- VBA scripts are no longer supported in WinCC so that VBA scripts are not migrated.
- Some functions are given new names in WinCC. No function names are adapted during the migration. Scripts continue to function without adapting because the old names can still be used in scripts. However, the old names are no longer supported by auto-completion and help.

Special features of VB scripts

- In WinCC V7 VB-scripts are structured as modules which may contain one or more procedures. In WinCC every script may only contain one procedure. The migration splits modules with multiple procedures and creates a new script for each procedure. Declarations for several global tags can be included in a script from WinCC V7. The global tags are initialized when calling the script and are available for all procedures of the script. When a script is split, the global tags are migrated to the script which follows the tag declaration. After the migration you must ensure that this first script is also called first so that the global tags are initialized. The global tags are only available for the split scripts after initialization.
- WinCC does not support the declaration of VB classes.
- In WinCC, the parameters in a VB script must always be transferred with the keywords "ByRef" or "ByVal". In WinCC V7-scripts there is no defined transfer with the keywords. In the migration, the transfer of the parameters is supplemented by the keyword "ByRef".

Stricter criteria in the checking of functions may make it necessary to rework scripts and functions after the migration. Errors and warnings are caused in WinCC by, for example:

- Objects, such as tags of faceplates, which are referenced in the functions but do not exist.
- Incorrect use of the VB object model, e.g. HmiRuntime.ActivateScreen.Item.
- Different number of parameters in the definition and call of a function.

- VB functions that deliver no return value.
- If the return value of another function is set within the VB function.
- Tags which are used in a VB function, but not defined in this function.

Special features of C-functions

WinCC V7 is sensitive to upper case/lower case in the names of C functions. In WinCC, names which only differ by upper/lower case letters are not allowed. Upper/lower case letters of function names remain unchanged during the migration. If a migrated project contains function names which only differ by upper/lower case letters, one of the functions in the engineering system is identified as invalid. An error message is generated during compiling.

Stricter criteria in the checking of functions may make it necessary to rework scripts and functions after the migration. Errors and warnings are caused in WinCC by, for example:

- Objects, such as tags of faceplates, which are referenced in the functions but do not exist.
- Different number of parameters in the definition and call of a function.
- If the external definition of a used function does not match the function definition. In this case you can either remove the external definition or adapt it.
- If the names of functions or tags in the Defines contain special characters which are not permitted.

ODK functions

ODK functions for both changing CS data and for changing Runtime data are supported in WinCC V7. In WinCC, only functions for changing Runtime data are still supported. The ODK functions for changing CS data are no longer supported. Functions and applications which call these ODK functions may not work as desired after the migration. Further information about using ODK functions can be found in the Runtime API online help.

See also

Migration (WinCC V7) (Page 89)

Migration of header files (WinCC V7) (Page 117)

Migration of actions (WinCC V7) (Page 118)

Migration of standard functions (WinCC V7) (Page 119)

Changing of function names (WinCC V7) (Page 128)

Migration of header files (WinCC V7)

Introduction

Some things need to be considered for the migration of header files due to the system-side changes. The changes due to the migration are described below.

Migration of header files

In WinCC V7 the standard storage procedure for header files is in the project. The header files stored in the project are migrated and stored in WinCC in the project tree in the respective device under "Functions > C-Header". If you have saved header files elsewhere and bound them into the project in WinCC V7, these header files are not migrated. You have to bind the header files that have not been migrated into the project yourself.

See also

Migration of functions and actions (WinCC V7) (Page 116)

Migration of actions (WinCC V7)

Introduction

Actions in WinCC V7 are migrated to tasks in WinCC. Jobs are managed in WinCC in the "Jobs" editor. Some things need to be considered for the migration of actions due to the system-side changes.

Migration of actions

In WinCC only one trigger is ever assigned to a task. In WinCC V7, you have the option of configuring several triggers for an action, for example a tag trigger and a cyclic trigger. An action with several triggers is split up into several jobs in the migration. A new task is created for each trigger.

In WinCC V7 there are global actions and local actions. In WinCC there are only global jobs. A global action is migrated to a global job. A local action is also migrated to a global job. However, the script of a local job is extended. In the extension of the script the system tag "@LocalMachineName" is evaluated so that the job only runs on the PC which is provided for the local action. Therefore, always check the name of the PC in the migrated script and change it if necessary.

If a script is extended, an alarm is output in the migration log.

Migration of password-protected actions

An action can be protected by a password in WinCC V7. A task and a password-protected script are created from this during the migration.

See also

Migration of functions and actions (WinCC V7) (Page 116)

Migration of standard functions (WinCC V7)

Migration of standard functions

Standard functions are part of the Runtime installation of WinCC and are therefore not migrated. if you have changed the source code of a standard function in your WinCC V7 project, the change is lost in the migration.

Standard functions " OnDeactivateExecute"

The standard function "OnDeactivateExecute" is migrated to a task with the "Runtime-Stop" trigger.

Customized standard functions

Customized standard functions are saved in WinCC V7 in the "APLIB" folder. You should copy these functions into the project directory to ensure that user-defined standard functions are migrated.

You can find customized C standard functions in the "wincc\aplib" folder of the WinCC V7 installation directory. Copy these functions to the "library" folder in the project directory.

You can find customized C standard functions in the "wincc\aplib\ScriptLibStd" folder of the WinCC V7 installation directory. Copy these functions to the "ScriptLib" folder in the project directory.

See also

Migration of functions and actions (WinCC V7) (Page 116)

Migration of logs (WinCC V7)

Basics on migration of logs (WinCC V7)

Introduction

The log system of WinCC V7 can process and output both configuration data and runtime data. WinCC only supports the documentation of runtime data.

The definition of which data are documented is stored in WinCC V7 in layout files. The output of contents is controlled by print jobs.

Migration of reports

The migration recognizes the layout files for the Runtime documentation from their content. If a layout file contains objects for the project documentation, this layout file is not migrated. If a layout file contains embedded layouts which are also used for the project documentation, this layout file is not migrated. if a print job references a layout file for the project documentation, this print job is not migrated. An alarm is output in the migration log for every layout file that is not migrated.

Note

If a WinCC V7 project was already created with earlier versions of WinCC, for example V5.0, a log can contain objects which were supplied with data in runtime by providers from the earlier version. These objects are not migrated.

Language-dependent contents

The log system of WinCC V7 does not support multilingual contents. Therefore the report system requires a separate layout file for every runtime language. The language assignment of the files is defined by name conventions of the contained folders or file names. The language-dependent layout files are combined in a multilingual report file in the migration. The migration must therefore consider different files to create a multilingual report. The following rules apply for merging the layout files for different languages:

- The first found layout file is used respectively for the migration of report objects. Differences in the layout of the different languages are therefore lost in the migration.
- All language-dependent files of a report are used to extract the language-dependent contents.
- The text entries of a language-independent report are adopted for all configured runtime languages.

Migration of print jobs

In WinCC V7 the start parameters for a print job are defined in the properties of the respective print job. In WinCC you cannot define start parameters on a print job. The job planner is available in WinCC for starting print jobs. There is a job of the "Print job" type in the job planner.

The print job is split into two parts in the migration of a cyclic print job. The migration creates a job of the "Print job" type in the job planner. The migration also creates a print job in the report system and connects it with the newly created job.

Migration of system layouts

The following system layouts are migrated from WinCC V7:

WinCC V7 Print job	WinCC System print job	WinCC System report
AlarmControl - Picture	Alarm Control - Picture	@Alarm Control - Picture
AlarmControl - Table	Alarm Control - Table	@Alarm Control - Table
FunctionTrendControl	Function Trend Control - Picture	@Function Trend Control - Picture
OnlineTrendControl - Picture	Online Trend Control - Picture	@Online Trend Control - Picture

WinCC V7 Print job	WinCC System print job	WinCC System report
OnlineTableControl - Picture	Online Table Control - Picture	@Online Table Control - Picture
OnlineTableControl - Table	Online Table Control - Table	@Online Table Control - Table
UserArchiveControl - Picture	Recipe Control - Picture	@Recipe Control - Picture
UserArchiveControl - Table	Recipe Control - Table	@Recipe Control - Table
RulerControl - Table	Ruler Control - Table	@Ruler Control - Table
RulerControl - Picture	Ruler Control - Picture	@Ruler Control - Picture

Changes in one of these system layouts are also available after the migration.

See also

Migration (WinCC V7) (Page 89)

Migration of log layouts (WinCC V7) (Page 121)

Migration of the "tag" object (WinCC V7) (Page 122)

Migration of a message sequence report (WinCC V7) (Page 122)

Migration of log layouts (WinCC V7)

Introduction

The layout file for a log from WinCC V7 is split up into several sections. The structure of a layout in WinCC differs in several details. Therefore the output of a log may differ slightly from the output in WinCC V7 after the migration.

Migration of report layouts

The following table shows the different structure of the layouts of WinCC V7 and WinCC:

WinCC V7	WinCC
Static layer of the cover sheet	Watermark cover sheet
Cover sheet	Title page
Static layer of the header	Header detail page
Static layer of the report content	Watermark detail page
Log content	Detail page
Static layer of the footer	Footer detail page
End page	Back page
Line layout	Line report

If a report layout contains page feeds, the migration generates a new detail page for every page feed in the report of WinCC.

Which log objects are allowed in a certain section of a log differs in WinCC V7 and WinCC. If a log object from a WinCC V7 log is not allowed in WinCC, this object is not migrated. If a report object is not migrated, an alarm is output in the migration log.

See also

Basics on migration of logs (WinCC V7) (Page 119)

Migration of the "tag" object (WinCC V7)

Introduction

Some log objects which are available in WinCC V7 are no longer supported by WinCC. The migration maps no longer available objects to objects available in WinCC as far as possible.

Static metafile

When a copy of the WinCC V7 is migrated, the assigned *.emf file cannot be resolved by the migration. A corresponding note is entered in the migration report.

I/O field of a report

The "tag" log object from WinCC V7 is no longer available in WinCC. The object is replaced in WinCC by an I/O field. When mapping the "Tag" object to the I/O field, the properties "Calculation" and "Output value" are replaced by the "Process value" property. The "Calculation" property has priority over the "Output value" property. If both properties are used in the project to be migrated, the value of the "Calculation" property is migrated. The value of the "Output value" property is not migrated.

The migration takes places as shown in the table below:

WinCC V7		WinCC
Calculation	Output value	Process value
not set	not set	not set
Script	not set	Script
not set	Tag	Tag
Script	Tag	Script

If the value of a property is not migrated, an alarm is output in the migration log.

See also

Basics on migration of logs (WinCC V7) (Page 119)

Migration of a message sequence report (WinCC V7)

Introduction

A print job is available for an alarm sequence report in WinCC. You can configure a system report for line printers or a system report for page printers for this print job. In WinCC V7 only one print job is available for a line log also but several layouts can be configured for the output.

Migration of a message sequence report

Since only a line layout or a page layout is allowed for the alarm sequence report in WinCC, the layout configured on the print job is migrated. If you want to use another type of alarm sequence report after migration, you need to execute the "Add system reports" command in WinCC in the report shortcut menu.

If a report layout is not migrated, an alarm is output in the migration log.

See also

Basics on migration of logs (WinCC V7) (Page 119)

Migration of the user administration (WinCC V7)

Migrating user administration

The user administration of WinCC V7 differs from that in WinCC in several details. The migration of the user administration may therefore lead to changes.

In WinCC V7 a user may have authorizations which differ from the authorizations of his user group. In WinCC the authorizations are assigned at the user group level.

To migrate all authorizations, the migration creates a new user group for every user group first. A new group is created additionally for every user whose authorizations differ from the user group. If several users have the same deviation from the group authorization, they all become members of the same, newly created group.

Example 1

WinCC V7	WinCC
Group 1(Auth A, Auth B) User 1(Auth A, Auth B) User 2(Auth C) User 3(Auth C)	Group 1(Auth A, Auth B) User 1
	Group 1_1(Auth C) User 2 User 3

Example 2

WinCC V7	WinCC
Group 2(Auth A, Auth D) User 4(Auth D) User 5(Auth A, Auth B)	Group 2(Auth A, Auth D) User empty
	Group 2_1(Auth D) User 4
	Group 2_2(Auth A, Auth B) User 5

Example 3

WinCC V7	WinCC
Group 3(Auth C, Auth E) User empty	Group 3(Auth C, Auth E) User empty

An alarm is output in the migration log for every newly created user group.

Migrating passwords

Passwords are not migrated. The password "administrator" is set by default for the administrator. Users are not assigned any passwords. This means that you must assign new passwords in WinCC.

Migration of cycles (WinCC V7)

Migration of cycles

In WinCC V7, there are two types of cycles:

- Update cycles
- Logging cycles

These types of cycles are edited in WinCC V7 in different editors.

In WinCC there is only one editor for editing all cycles.

The cycles of a project are migrated as follows:

- The 10 standard update cycles are used unchanged.
- The migration creates five user-defined update cycles.
- The migration creates a new cycle for every update cycle.

WinCC manages internally which cycles are used for updating and which for logging.

Unsupported settings

Not all of the cycle settings available in WinCC V7 are supported by WinCC. During migration, an unsupported setting is replaced by an existing one in WinCC and an alarm is generated in the migration log.

The log tags with an unsupported acquisition cycle are therefore logged in runtime with a cycle other than the one configured.

Example: If a cycle was configured for "Monday", "Wednesday" and "Friday", for example, the tag is logged "Once Daily" after migration.

Migration of process value logging (WinCC V7)

Introduction

The settings for process value logging in WinCC V7 are migrated to WinCC. There are special features in this regard.

Manual input

In WinCC V7, values can be entered manually in logs and in individual log tags. There is also the "Recalculation with manual input" function.

Manual input and recalculation are not supported in WinCC. During migration, these settings are ignored and an alarm is generated in the migration log.

Basic Process Control (WinCC V7)

Introduction

A project in which you have used Basic Process Control can always be migrated to WinCC. However, specific configurations for Basic Process Control are lost in the migration of such a project.

Migration of Basic Process Control

Default screens of Basic Process Control as well as screens generated by Basic Process Control are not migrated. Other screens of project are migrated. But these screens can include objects that are not supported by WinCC. These objects are not migrated. This concerns the following objects:

- Acoustic alarms
- 3D bar
- Extended analog display
- Extended status display
- Group display

A screen hierarchy which was created with Basic Process Control will not be migrated. Only those screens are migrated which are configured for the display in the work area. Screen in screen characteristics are not migrated.

Unsupported components and functions

The following components of Basic Process Control are not supported by WinCC:

- OS Project Editor
- Lifebeat Monitoring
- Picture Tree Manager

- Block List Editor
- Area-specific user administration
- Multi-VGA

Entries in the start list for processes from Basic Process Control are not supported by the migration.

If you call API functions of Basic Process Control in a script, you get an error message in the compiling of the project.

Appropriate alarms are output in the migration log.

2.8.2.3 Copying runtime data (WinCC V7)

Runtime data

When migrating a project, the configuration data is migrated. You can copy the runtime data during migration to continue using it.

The runtime data consists of the following:

- User logs from user archives
  User logs are process data that are saved in a database in Runtime.
- Log data
  The data of data logs and alarm logs are acquired and logged in runtime.

See also

Migration (WinCC V7) (Page 89)

2.8.2.4 Reference (WinCC V7)

Migrating data types (WinCC V7)

Introduction

The following table describes the mapping of the data types from WinCC V7 to the data types in WinCC.

Migrating data types

The internal data types are mapped as follows during migration:

Internal data types WinCC V7	Internal data types of WinCC
Bool	Bool
Char	SByte
Byte	UByte

Internal data types WinCC V7	Internal data types of WinCC
Int	Short
UInt	UShort
Long	Long
ULong	ULong
Float	Float
Double	Double
String	WString
DateTime	DateTime

Mapping of the S7 data type "Timer"

An external tag with the S7 data type "Timer" with timer address is mapped to the S7 data type "Timer". The address will be retained.

An external tag of the S7 data type "Timer" addressing a data block or a memory address will be mapped to the S7 data type "S5 Time". The address will be retained.

Example:

WinCC V7

Tag	S7 data type	Address	Comment
Timer_Actual_Value	Timer	T10	BCD coded timer value
Timer_Setpoint_Value	Timer	DB10.DBW200	BCD coded timer value
Timer_Setpoint_Value#2	Timer	MW20	BCD coded timer value

WinCC

Tag	S7 data type	Address	Comment
Timer_Actual_Value	Timer	%T10	BCD coded timer value
Timer_Setpoint_Value	S5Time	%DB10.%DBW200	BCD coded timer value
Timer_Setpoint_Value#2	S5Time	%MW20	BCD coded timer value

See also

Migration (WinCC V7) (Page 89)

Changing of function names (WinCC V7) (Page 128)

Changing of function names (WinCC V7)

Changed function names

In WinCC the function names of C-functions and VB-functions sometimes change in relation to WinCC V7. This concerns the following function names:

WinCC V7	WinCC
GetLinkedVariable	GetLinkedTag
GetLocalPicture	GetLocalScreen
GetParentPicture	GetParentScreen
GetParentPictureWindow	GetParentScreenWindow
PASSCheckLevelPermission	IsUserAuthorized
OpenHomePicture	ActivateStartScreen
OpenNextPicture	ActivateNextScreen
OpenPrevPicture	ActivatePreviousScreen
OpenStoredPicture	ActivateStoredScreen
StorePicture	StoreScreen
GetLanguage	GetLanguageByLocaleID
InquireLanguage	InquireLocale or InquireLocaleID
SetLanguage	SetLanguageByLocaleID
OpenPicture	ActivateScreen
PASSLoginDialog	ShowLogonDialog
ProgramExecute	StartProgram
DeactivateRTProject	StopRuntime
ExitWinCC	StopRuntime

See also

Migration of functions and actions (WinCC V7) (Page 116)

2.8.2.5 Integrated projects

Basics on migration of integrated projects (WinCC V7)

Introduction

PLCs and HMI devices contained in a project integrated into STEP 7 are connected with each other by the configuration. The configuration data of WinCC V7 and STEP 7 are also connected with each other. In the migration of an integrated project, the complete project is migrated with the parts from WinCC V7 and STEP 7. The connections are retained.

Observe the following when migrating integrated projects:

- Before you migrate an integrated WinCC V7 project to the TIA Portal, create the migration file (Intermediate file) using the migration tool.
- Only projects created with WinCC V7.4 SP1 can be migrated.
- Controller alarms and integrated tags are only created for TCP/IP, MPI and PROFIBUS connections. Named connections are not supported.
- For an operating station (OS) to be migrated, it has to be located below a PC station and the WinCC application in SIMATIC Manager.

Migrating an integrated project

In the migration of an integrated project, the same requirements apply for the WinCC V7 part as for the migration of a non-integrated WinCC V7 project. The objects and properties contained in the WinCC V7 part must be supported by WinCC, e.g. the communication driver.

In addition to the requirements for the WinCC V7 component, there are also requirements for the STEP 7 component of the integrated project. The objects and properties contained in the STEP 7 component must be supported in STEP 7. For detailed information, refer to the documentation for STEP 7.

In order to be able to fully migrate an integrated project, the following components must be installed on the PC for the migration:

- STEP 7 V5.4 SP5 ... STEP 7 V5.6
- WinCC V7.4 SP1

In order to be able to fully edit an integrated project, the following components must be installed on the PC for the editing:

- STEP 7
- WinCC

If you only want to save the WinCC V7 project in the migration format, you can use the migration tool.

An integrated project is always fully migrated. You cannot migrate the WinCC project contained in the integrated project on its own. You can only migrate the contained STEP 7 project on its own. If you want to migrate the STEP 7 project on its own, you must delete the integrated WinCC project in the SIMATIC-Manager. Then you migrate the STEP 7 project without the WinCC project.

Multi-Client

Multi-Clients are not supported in WinCC. If you create a WinCC client in the SIMATIC Manager, a Multi-Client is created automatically. The Multi-Client project is integrated in STEP 7. If the migration recognizes a Multi-Client, it cancels the migration. To migrate the STEP 7 project despite this, you must delete the Multi-Client in the SIMATIC Manager. Then migrate the project.

Storage location of an integrated WinCC project

If you want to migrate an integrated project, the HMI parts which is contained must be on the same PC as the STEP 7 part of the project. If the HMI part is on another PC, only the STEP 7 part is migrated, the HMI part of the project is missing after the migration.

Unsupported objects

The following components are not supported by the migration:

- STEP 7 multiproject
  A STEP 7 multiproject cannot be migrated; the migration cancels the procedure.
- Central Archive Server - CAS
  If the migration recognizes a Multi-Client, it cancels the migration.

See also

Migrating an integrated project (WinCC V7) (Page 130)

Migrating an integrated project (WinCC V7)

Introduction

When migrating an integrated project, the components from both the WinCC V7 project and the STEP 7 project are migrated. Therefore, migrate the Intermediate file generated by the migration tool. The Intermediate file has the file format "*.am". During migration, the data is copied from the existing project and migrated to a new project. You cannot migrate to an existing project.

Requirement

- STEP 7 Professional and all the option packages used in the project are installed.
- WinCC Professional is installed.
- The TIA Portal is restarted.
- The Intermediate file of the project to be migrated is on your drive.

Procedure

Proceed as follows to migrate an integrated project:

1. Select "Start > Migrate Project".

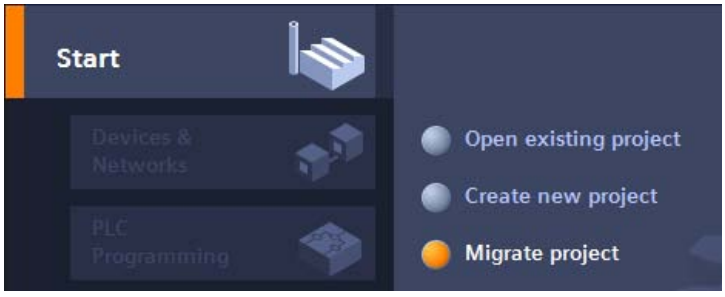

2. Navigate to the Intermediate file in the "Source path" field.
3. Change the information for the project to be created, if necessary. For example, change the project name or project path. The data to be migrated is created in the new project.
4. Click "Migrate".
   A new project is created and migration of the data is started:
   - The Project view opens.
   - The progress of the migration is shown in a migration window.
   - Warnings and errors about the migration process are displayed in the Inspector window under "Info > General".
   - All information about the migration is saved in a log file.
   - A message is displayed upon completion of the migration. The message contains a link that you can use to open the log file.

Alternatively, start the migration from the project view via "Project > Migrate Project".

When migration is complete, you will find a newly created device for each migrated HMI device in the project tree. You will also find a newly created device for every migrated PLC. These devices include the migrated data.

Note

Integrated projects with alarm views

The alarm classes in the alarm view can be disabled during migration of the project. Once the migration of the project is completed, check the settings in the alarm view. If needed, enable the required alarm classes in the Inspector window of the alarm view under "Properties > General".

Opening the migration report

The migration log is saved together with the migrated project. You can view the log at a later point in time. Open the log file as follows:

1. Click "Shared data" in the project navigation.
2. Click on "Logs".
3. Double-click the migration log.

See also

Basics on migration of integrated projects (WinCC V7) (Page 128)

2.9 Migrating integrated projects

2.9.1 Migrating an integrated project

Introduction

In integrated projects, you use SIMATIC controllers and WinCC components together in a project. When an integrated project is migrated, the complete project will be migrated with components from WinCC and STEP 7. Configured connections between control and visualization remain intact.

Migrating an integrated project

When migrating an integrated project, the same requirements apply for the STEP 7 component as those for migration of a non-integrated STEP 7 project. The objects and properties contained in the WinCC component must also be supported in WinCC (TIA Portal).

For an operating station (OS) to be migrated, it has to be located below a PC station and the WinCC application in the project tree of SIMATIC Manager. The following figure shows the assignment of the operating station within the initial project:

Additional migration requirements for integrated projects can be found in the documentation for WinCC.

Also note that the initial project must be compiled before the migration.

In order to fully migrate an integrated project, the following components must be installed on the programming device/PC for the migration:

- STEP 7 V5.4 SP5 or higher
- WinCC V7.4 SP1 with the latest update or WinCC Flexible 2008 SP2 or higher

To be able to fully post-edit an integrated project, the latest version of the following components must be installed on the PC for post-editing:

- STEP 7 Professional
- WinCC Basic, WinCC Comfort/Advanced or WinCC Professional, depending on the components used

Using the migration tool

It is necessary to use the migration tool under the following circumstances:

- The initial project is not located on the same programming device/PC as the installation of the TIA Portal.
- SCADA devices are included in the initial project. These can only be migrated with the migration tool.
- WinCC Professional V15 and STEP 7 with WinCC V7.4 SP1 cannot be installed on the same programming device/PC. Therefore, integrated projects with WinCC V7.4 SP1 parts must be prepared for migration with the migration tool.

Migration of the STEP 7 part of an integrated project

An integrated project is always fully migrated. Individual components cannot be migrated on their own. You can only migrate the included STEP 7 project alone, if you have previously deleted all HMI stations in the SIMATIC stations in the SIMATIC Manager and then recompiled the project in NetPro.

Alternatively, you can open the project in an installation of STEP 7 V5.4 SP5 or higher without an installation of WinCC. Then, save the project again and select the "Reorganize" function during saving. The WinCC parts are then automatically removed when the copy is saved.

You can then migrate the STEP 7 project without the WinCC project.

Migration of an integrated project with the hardware configuration

In integrated projects, HMI devices are migrated even if the hardware configuration is not included in the migration. The STEP 7 part of the hardware configuration, including network configurations, and connections and interrupts, is migrated only if you include the hardware configuration in the migration. Otherwise, unspecified modules will be created for the STEP 7 devices and you will need to convert them into suitable modules after the migration.

HMI modules that are plugged into a PC station are converted to a separate Station during the migration. If you perform the migration without including the hardware configuration, the migrated project then contains a non-specified SIMATIC PC Station and a SIMATIC PC Station with the HMI devices. References to HMI devices are not imported during migration. When the hardware configuration is included, the migrated project contains two separate stations: the HMI Station and the PC Station.

Storage location of an integrated WinCC project

If you migrate an integrated project, the HMI part of it must be on the same PG/PC as the STEP 7 part of the project. If the HMI part is on a different PG, then only the STEP 7 part will be migrated.

Unsupported objects

The following components are not supported for migration:

- STEP 7 multiproject
  A STEP 7 multiproject cannot be migrated. Migration will be canceled.
- Central Archive Server - CAS
  If a CAS is part of an integrated project, then the migration will be carried out but the CAS data will not be migrated.

See also

Post-editing integrated projects (Page 135)

2.9.2 Post-editing integrated projects

If you have migrated an integrated project without hardware configuration, unspecified CPUs are used instead of the CPUs of the original project. Since no connection can exist between an unspecified CPU and an HMI device, connections from the source project are also imported only unspecified.

Procedure

To continue to use an integrated project after the migration, follow these steps:

1. Convert the unspecified devices into suitable devices again.
2. Restore the integrated HMI connection between the HMI device and the PLC.
3. Connect all HMI tags to the newly created integrated connection.
4. Restore the connection between HMI tags and PLC tags.
5. Delete the non-integrated HMI connection.

In the following chapters a sample project is used to describe the individual steps in more detail.

See also

Converting unspecified CPUs into specified CPUs (Page 136)

Creating an integrated HMI connection (Page 137)

Re-linking HMI tags (Page 139)

Delete non-integrated HMI connection (Page 140)

2.9.3 Converting unspecified CPUs into specified CPUs

The first step after the migration without hardware configuration is the conversion of the unspecified CPUs into specified CPUs. Unspecified CPUs are placeholders for certain CPUs from the hardware catalog that are not currently known. You can define general parameters and home the CPUs already in the user program. However, the project is not fully functional until the unspecified CPU has been specified.

Specifying a CPU using module replacement

To use module replacement to specify an unspecified CPU, follow these steps:

1. Select the unspecified CPU in the network or device view.
2. Select the "Replace device" command in the shortcut menu.
   The "Replace device" dialog opens.

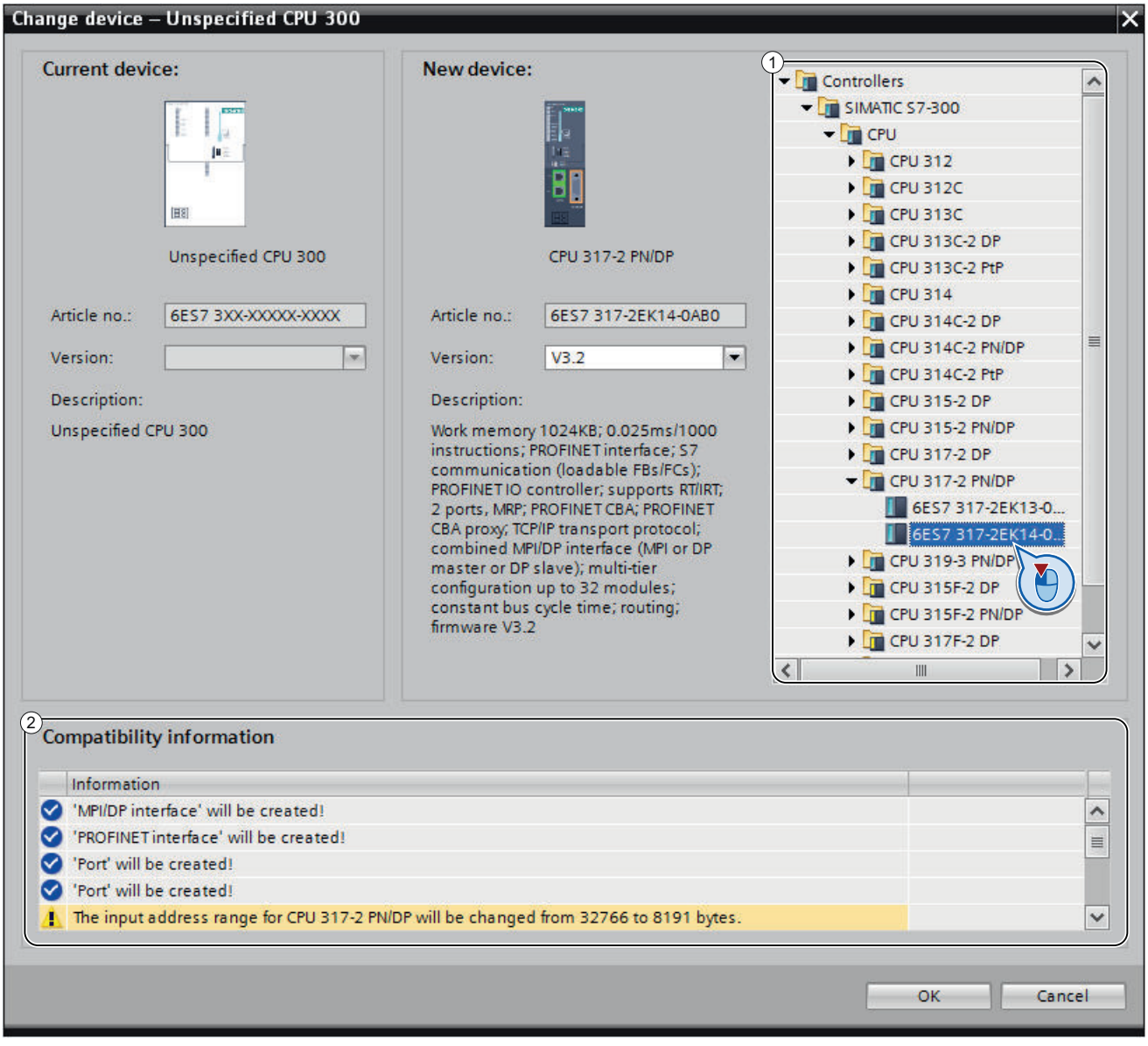

3. Under "New device" in the tree structure, select the module with which you want to replace the unspecified CPU. (Area 1)
   "Compatibility information" provides you with information on the extent to which the selected CPU is compatible with the configuration in source project. (Area 2)
4. Click "OK".
5. Perform the above-described steps for all unspecified CPUs.

See also

Creating an integrated HMI connection (Page 137)

2.9.4 Creating an integrated HMI connection

After you have specified the unspecified CPU, establish the connection to the HMI-device.

Procedure

To create a connection graphically, follow these steps:

1. On the toolbar, click the "Connections" icon. This activates connection mode.

2. Select the connection type "HMI connection" in the adjacent drop-down list.
   The network view highlights in color all CPUs and HMI devices that can be used for an HMI connection.
3. You can now have the connection path automatically determined, or explicitly select a connection path via specific interfaces:
   – Allow connection path to be automatically determined
     Select the source CPU for a connection. Drag the mouse to the target components. Confirm the connection endpoint with another mouse click.
     Alternatively: While holding down the shift button, select the target components and with the right mouse button select the "Add new connection" command.
   – Selecting an explicit connection path from interface to interface
     Click on the subnet interface in the device for which you want to create a connection. Hold down the mouse button, drag the cursor to the relevant interface in the target device and then release the mouse button.

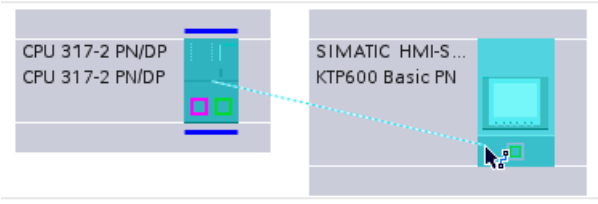

Result

The following figure shows the state after the integrated connection has been created:

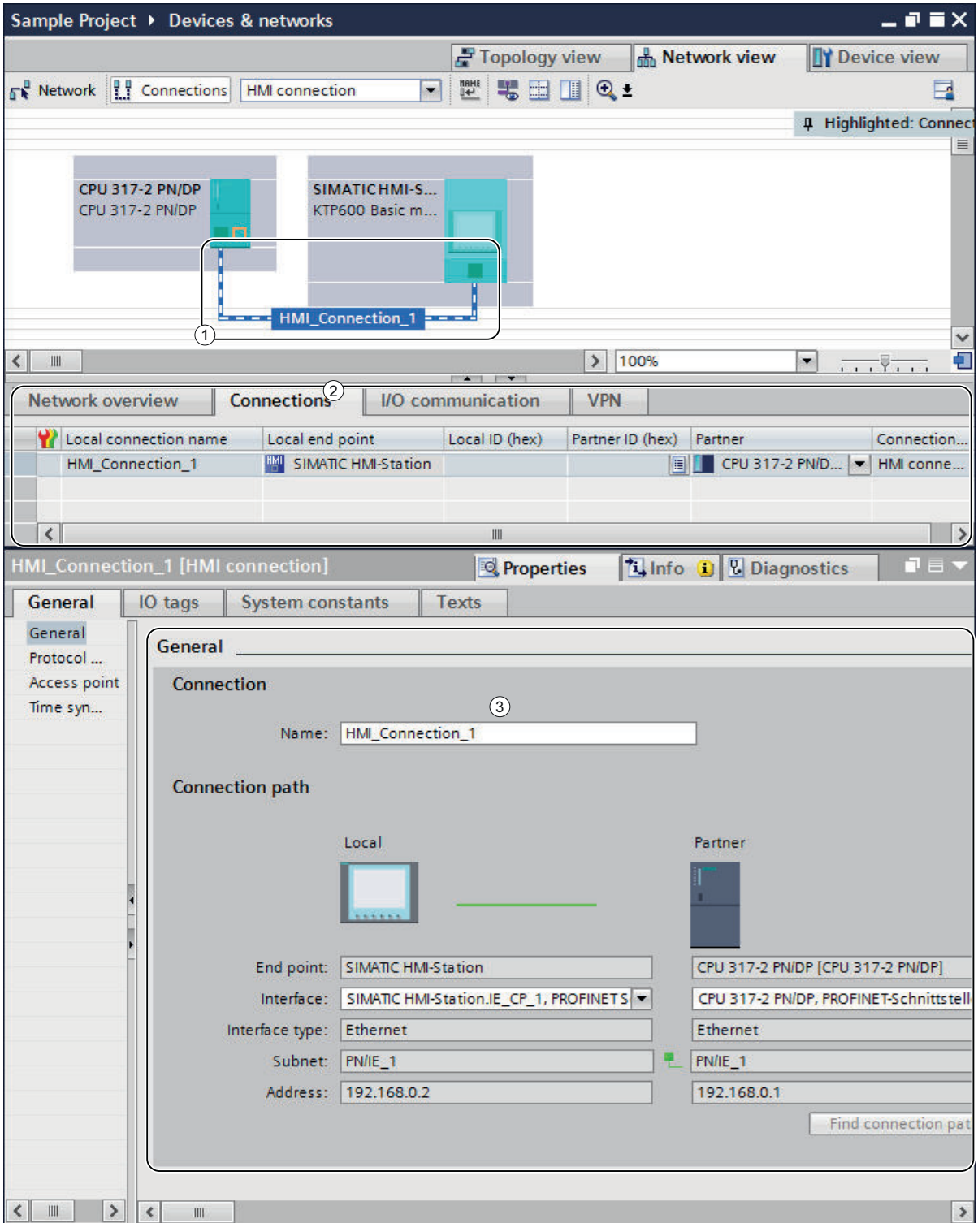

① An integrated HMI connection is created and highlighted in the network view.

② The connection is shown in the connection table of the components.
③ The connection can be edited in the connection properties.

See also

Re-linking HMI tags (Page 139)

2.9.5 Re-linking HMI tags

When you have created a new HMI connection between the CPU and HMI device, you have to assign the existing HMI tags to the new connection. Perform the following steps for each line in the relevant tag table.

Procedure

To re-link HMI tags, follow these steps:

1. In the project tree, navigate to the HMI tags and double-click the relevant tag table to show this in the work area.
   The tag table opens.

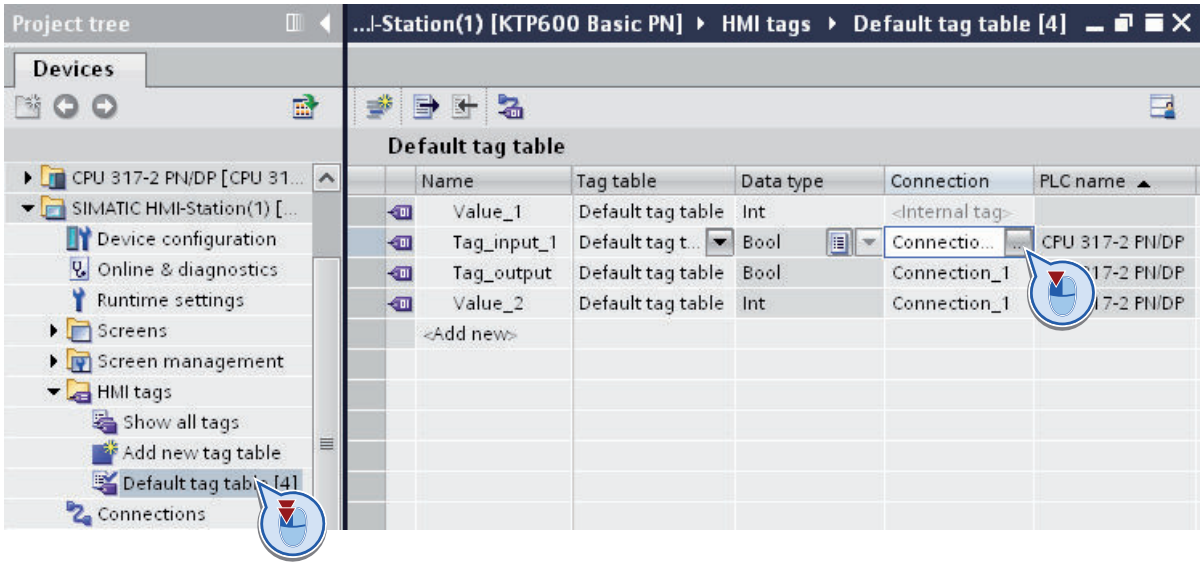

2. Click the " ... " button in the "Connection" column.
   A dialog box for selecting the connection opens.

3. Select the newly established HMI connection.

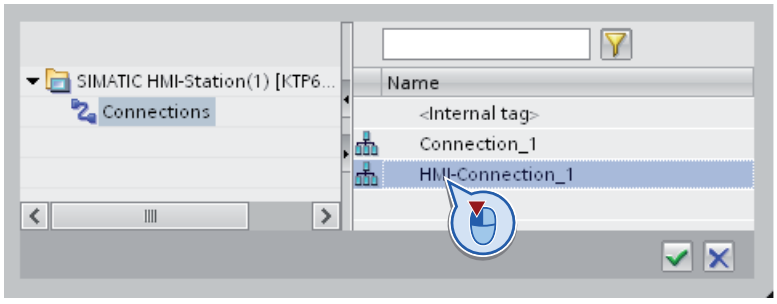

4. Click the "✓" button to apply the selected connection.
5. On the toolbar, click the "Re-connect PLC tag" button.

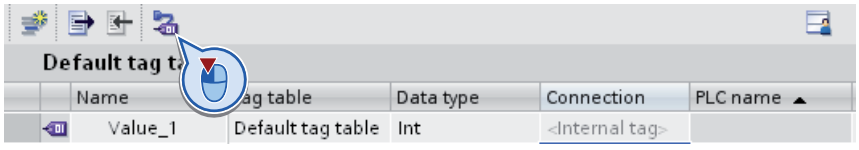

See also

Delete non-integrated HMI connection (Page 140)

2.9.6 Delete non-integrated HMI connection

Finally, you can remove non-integrated HMI connections that still remain from the source project.

Procedure

To delete the non-integrated HMI connections, follow these steps:

1. In the project tree, open the HMI device and double-click the "Connections" entry. The connection table opens.

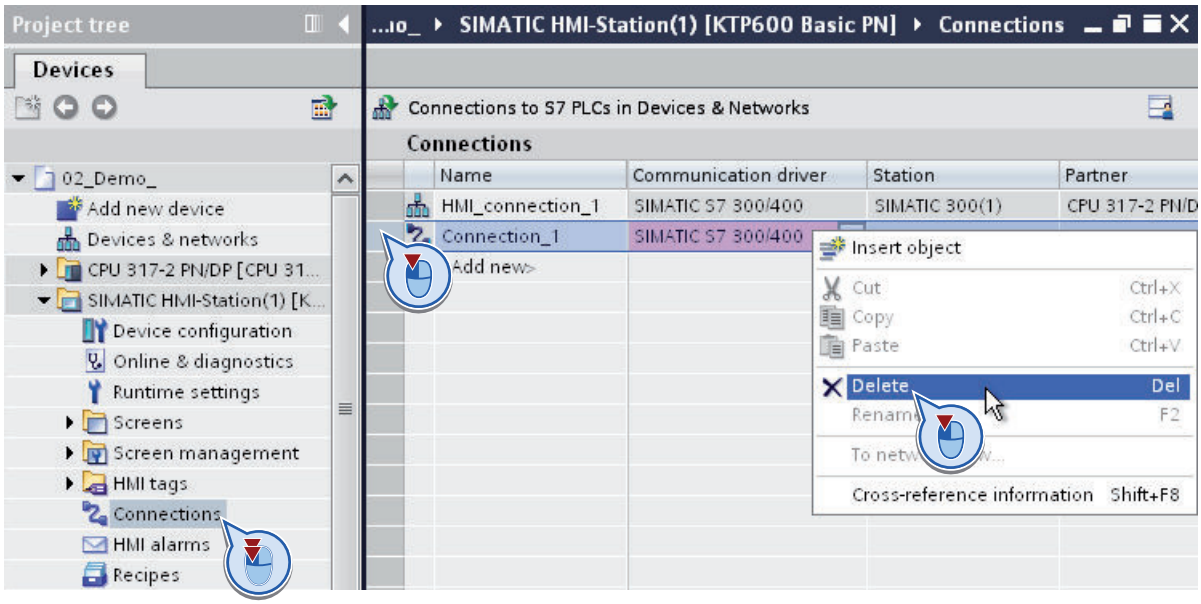

2. Select the row with the old connection in the table.
3. Select the "Delete" command in the shortcut menu of the connection line.
4. Perform the above-described steps for all non-integrated HMI connections of the source project.

Migrating PLC programs to a S7-1500 CPU / ET 200SP 3

3.1 Basic information on migrating PLC programs (S7-1500)

Introduction

You can migrate PLC programs within the TIA Portal. Migration within the TIA Portal is called "PLC migration". With this migration, the new module will be created in the project and the existing PLC program copied to the new module. The old module remains available unchanged in the project.

The advantage of migration compared to copying is that the program is automatically adapted to the new CPU family as much as possible. Program structures that are no longer permitted are updated. Instructions that are not available on the new module are replaced with corresponding S7-1500 instructions.

The table below gives you an overview of the modules that can be migrated:

Original block	Target block
S7-300/400 CPU	S7-1500 CPU
ET 200S CPU	ET200SP CPU
ET 200S CPU	S7-1500 CPU
ET 200S CPU	ET 200pro CPU
IM 154-8 CPU	CPU 1516-2 PN

The migration to S7-1500 and the migration to ET 200SP follow the same rules. The PLC programs are treated identical in both migration operations.

Scope of migration

During PLC migration, all components of the PLC program are copied to the newly created module. The following objects are included, for example:

- Program blocks, whose programming language is supported by S7-1500 and that have no know-how protection.
- PLC tag tables
- Watch and force tables
- PLC data types
- Technological objects
- User-defined groups in the project navigation

The hardware configuration of the output module is not transferred automatically.

WARNING

Preventing personal injury and material damage

Changes are made to the program during PLC migration in some cases. It is therefore very important that you thoroughly review the program in a test environment after migration before putting it in operation.

Note

Additional support for PLC migration

You can find the latest information about PLC migration in the Siemens Industry Online Support:

Migration of complete systems:

http://support.automation.siemens.com/WW/view/en/83558085 (http://support.automation.siemens.com/WW/view/en/83558085)

Migration of controllers:

http://support.automation.siemens.com/WW/view/en/83557459 (http://support.automation.siemens.com/WW/view/en/83557459)

Migration of visualization:

http://support.automation.siemens.com/WW/view/en/76878921 (http://support.automation.siemens.com/WW/view/en/76878921)

Migration of communication:

http://support.automation.siemens.com/WW/view/en/83558087 (http://support.automation.siemens.com/WW/view/en/83558087)

FAQs on migration:

http://support.automation.siemens.com/WW/view/en/67858106 (http://support.automation.siemens.com/WW/view/en/67858106)

If you need further support, please contact SIMATIC Customer Support.

Note

Additional information on the TIA Portal

Additional information on all aspects of the TIA Portal can be found in the TIA Portal in the Siemens Industry Online Support.

http://support.automation.siemens.com/WW/view/en/65601780 (http://support.automation.siemens.com/WW/view/en/65601780)

There you will find information on our training offers. For experienced users of the classic SIMATIC S7 system who want to upgrade efficiently to SIMATIC S7-1500 with the TIA Portal, we recommend "TIA-SYSUP - SIMATIC TIA Portal system retraining course for SIMATIC S7-1500".

3.2 Performing migration (S7-1500)

Requirement

- A consistent reference program is available.
- All blocks are compiled up-to-date. This also applies to blocks that are included in the project, but will not be called during program execution.
- The program was created with TIA Portal V12 or later or upgraded to this version.

Preparing for migration

Proceed as follows to prepare the program for migration:

1. Open the device that contains the reference program in the project navigation.
2. Open the folder "Program blocks" and check whether it contains know-how protected blocks.
3. Remove the know-how protection from the blocks.
4. Open the folder "Program blocks > System blocks" and check whether the block contained therein include the extension "_LF" (legacy functions).
   These are library blocks from STEP 7 that are no longer supported in the TIA Portal. These blocks are not transferred to the new device during PLC migration because they are know-how protected.
5. If possible, replace the blocks with an instruction from the "Instructions" task card.
6. If the program contains instructions on alarm configuration, follow the instructions for migrating alarms with associated values.
   See also: Migrating alarms with associated values (Page 172)

Procedure

To migrate a PLC program within the TIA Portal, follow these steps:

1. Open the device that contains the reference program in the project navigation.
2. Open the device configuration.
3. Select the module that contains the reference program in the device view or network view.
4. Select the "Migrate to S7-1500" command in the shortcut menu.
   The "Migrate to S7-1500" dialog opens.
5. In the "New device" area, select the device to which you want to migrate your program and confirm with "OK".
   A security message appears informing you that the program will be changed during the migration.
6. Confirm the security prompt.
   A dialog opens in which you can specify which instructions you want to use for point-to-point communication in the migrated program.

7. Select the required set of instructions.
   - Select the option "Use new PtP instructions for the integrated communication modules of the S7-1500" to use the new PtP instructions of the S7-1500.
   - Select the option "Continue to use PtP instructions for S7-300/400 communications processors" to continue using the PtP instructions of the CP 300/400 or ET200.

   You must select one of the two options even if you do not use point-to-point communication. The selection does not have an effect on the program in this case.
   The PLC program is migrated. An alarm then appears notifying you whether errors occurred during the migration. The alarm also contains a link to the migration log.
8. Open the migration log. It contains details on the migration and informs you about program changes which you must perform to make your program executable on the new device.
9. Process all the information contained in the migration log.
10. Compile the migrated project.
    See also: Migrating point-to-point program blocks (Page 183)

Result

A new device is created next to the original device in the project navigation. It contains the migrated PLC program.

To access the full functionality of the S7-1500, we recommend that you enable optimized block access for the migrated blocks. You can find additional information in the Help under Blocks with optimized access:

Blocks with optimized access

Note

Note that only the PLC program is converted by the PLC migration. The hardware configuration of the original device is not automatically transferred to the new device. Perform the hardware configuration of the new device manually after the migration.

3.3 Displaying a log file (S7-1500)

A log file is created for each migration. The log file contains the following information:

- Modifications to objects made during migration.
- Information on necessary adaptations you need to make to the program.

Procedure

To display the log file of the migration, follow these steps:

1. Open the "Common data > Logs" folder in the project tree. It contains the logs of all previously performed migrations.
2. Double-click the required migration log.
   The log opens.
3. Messages that relate to a certain position in the program are marked with an arrow in the "Go to" column. Double-click on the arrow to jump to the relevant position in the program.
4. Messages for which additional information is available are marked with a question mark in the "?" column. To obtain additional information on the message, click the question mark.

3.4 Special considerations for migrating PLC programs (S7-1500)

3.4.1 Information on migrating PLC programs (S7-1500)

3.4.1.1 Migrating organization blocks (S7-1500)

Organization blocks

The following rules apply to migration of organization blocks:

- Block names and numbers are transferred unchanged.
- The block interface is also accepted unchanged.
- The OB is assigned to the event corresponding to the OB type.
- The OB parameters, such as the priority, remain unchanged. If additional parameters can be assigned in the new CPU, they will be assigned default values.

Some organization blocks (OBs) are not supported by S7-1500 and therefore cannot be migrated. The following table provides an overview of these organization blocks and contains information on how you can reproduce the functionality in your program.

Non-migratable OB	Notes
OB60 (multicomputing interrupt)	The OB cannot be replaced. Multicomputing is not available on S7-1500.
OB65 (technology synchronization interrupt)	Check whether the OB in your program can be replaced by the MC servo OB or the MC interpolator OB.
OB7x (redundancy error)	The OB cannot be replaced. Redundancy is not available on S7-1500.
OB81 (power supply error)	Check whether the OB in your program can be replaced by the diagnostic interrupt OB.
OB84 (CPU hardware fault)	Check whether the OB in your program can be replaced by the diagnostic interrupt OB.
OB85 (program execution error)	Check whether the OB in your program can be replaced by the pull/plug OB or the rack error OB.
OB87 (communications error)	Check whether the OB in your program can be replaced by the diagnostic interrupt OB.
OB88 (processing interrupt)	Check whether the OB in your program can be replaced by the programming error OB.
OB90 (background cycle)	The OB cannot be replaced. Background processing is not available on S7-1500.
OB101 (hot restart)	The OB cannot be replaced. Hot restart is not available on S7-1500.
OB102 (cold restart)	Check whether the OB in your program can be replaced by the startup OB.

See also

FAQs on migration (http://support.automation.siemens.com/WW/view/en/67858106)

3.4.1.2 Migrating hardware IDs (S7-1500)

Addressing modules by means of hardware IDs

If your program addresses hardware modules, for example, at the "LADDR" or "ID" parameter, these addresses are no longer valid after migration to the new hardware. The addresses must be replaced. The hardware modules of the S7-1500 are addressed by hardware IDs. This means you have to enter the hardware identifier of the new module at the "LADDR" or "ID" parameter after migration.

To do this, follow these steps:

1. Open the device configuration.
2. Select the module that you want to address.
3. Select the "Properties > System constants" tab in the Inspector window.
   The table contains constants for all modules used and the required HW identifiers.
4. Select the constants for the module to be addressed and select the "Copy" command from the shortcut menu.
5. Insert the constants at the "LADDR" or "ID" parameter of the migrated instruction.

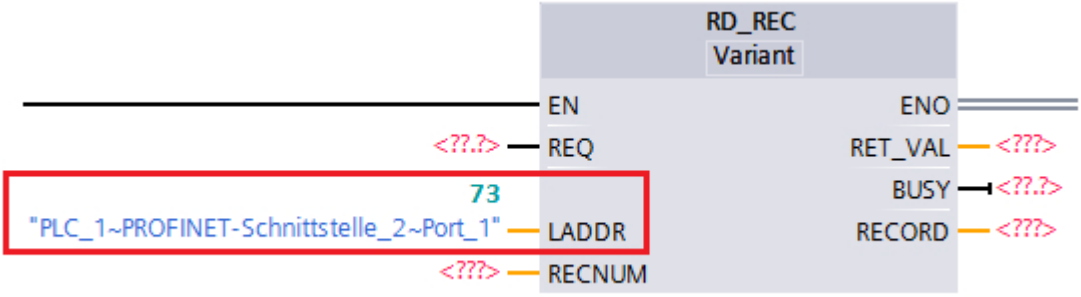

3.4.1.3 Migrating IEC timers and IEC counters (S7-1500)

Updating IEC timers

The "Q" and "ET" outputs of the IEC timers TP, TON and TOF are determined in S7-300/400 at the point in time at which the program runs the IEC timer. After this, the status of "Q" and "ET" remains unchanged. If you access these outputs multiple times in the program, you therefore always receive the same value.

This behavior has changed in S7-1500: Here, the program checks the current timer value at each access and determines the "Q" and "ET" outputs again.

Thus, after the migration, the behavior of your program may change if you access "Q" or "ET" multiple times.

For the program to demonstrate the same behavior as in S7-300/400, you can assign the value of "Q" or "ET" to a tag. In the program, you then reference this tag instead of the output.

IEC timers and IEC counters with EN/ENO interconnection

The instructions have been adapted to the IEC 1131-3 standard for S7-1500. They are now connected to the current path via the "IN" parameter and therefore no longer have the "EN" and "ENO" parameters. The migration reports an error if the reference program includes an IEC timer or IEC counter with preceding logic that is evaluated using the "EN" input parameter.

Insert a jump instruction between the preceding logic and the IEC timer or IEC counter to call the IEC timer or IEC counter based on the RLO. The following jump instructions are available:

- ---(JMP): Jump if RLO = 1
- ---(JMPN): Jump if RLO = 0

IEC counters: New parameters for the counter status in the block interface

After the migration, the block interface contains the "QU" and "QD" parameters instead of the "Q" output parameter for the counter status of an IEC counter. Only one of the two parameters is read out, depending on the IEC counter type. The other parameter in each case is not used.

If you have accessed the "Q" parameter in the program code, you need to manually adjust the access after the migration. Use "QU" if you are counting up and "QD" if you are counting down.

3.4.1.4 Migrating CPU data blocks (S7-1500)

CPU data blocks

Blocks that were created in the CPU by the CREAT_DB or CREATE_DB instruction and are only available online are not migrated.

See also

Migration of data block instructions (Page 167)

3.4.1.5 Migrating diagnostics functions (S7-1500)

System status lists

CPUs of the S7-1500 series do not provide system status lists. However, you can read some of the information using the "GET_DIAG", "Geo2Log", "DeviceStates", or "ModuleStates" instruction.
Additional information on migration of system status lists is available under: Migrating the RDSYSST instruction (Page 169)

System diagnostics with "Report System Errors"

The CPUs of the S7-1500 series have integrated system diagnostics. "Report System Errors" is therefore no longer supported. If the program contains blocks for "Report System Errors", disable system diagnostics in the reference program prior to migration and recompile the program. The blocks in question are deleted during this process.

System diagnostics for the S7-1500 CPUs is enabled by default. You do not have to make any additional settings.
To change the settings for the system diagnostics, select the CPU in the device configuration. The settings for the system diagnostics are then shown in the Inspector window under "Properties > System diagnostics". The alarms of the system diagnostics are enabled and can be deselected if they are not needed.

PROFINET/PROFIBUS diagnostics

The following rules apply to migration of PROFINET/PROFIBUS diagnostics:

- "Report system errors" is replaced by the integrated system diagnostics.
- FB 126 is not supported and can therefore not be migrated.
- Use the "DeviceStates" and "ModuleStates" instructions for application-specific evaluation.

3.4.1.6 Migrating absolute access to local data (S7-1500)

Absolute access to local data

Absolute access to the local data of a block was possible in S7-300/400 without declaring a symbol for the address.

Because pure absolute access in S7-1500 is no longer supported, migration deals with this as follows:

- If a temporary tag is declared in the block interface for an area within the local data, the absolute address is converted to a symbolic address.
- If there is no suitable temporary tag, the absolute address is retained. Because additional local tags may be created in some circumstances during migration, addresses may shift within the local data. Therefore, check to see if the absolute address you are using is still correct and if the correct data is being addressed. If possible, create a tag in the "Temp" section and use it for addressing. To specifically address areas within declared tags, you can overlay existing tags with AT or address individual areas with the syntax .X, .B, .W or .D.

Examples

The following examples show migration of absolute access to local data.

In the first table, you can see a program section prior to migration:

STL
L %LW20
L %LW3
+I
T %LW5

In the second table, you can see a program section after migration:

```
STL
L "MyTempVar1"
L "MyTempVar2"
+I
T %LW5
```

- Temporary tags were already declared for "LW20" and "LW3". These are used in the program after migration.
- No tag was declared for "LW5". You need to check if access is still correct after the migration.

3.4.1.7 Block parameters in S7-1500 (S7-1500)

Using block parameters

The following rules apply to the use of block parameters within the block in S7-1500:

- Input parameters may only be read.
- Output parameters may only be written.
- In/out parameters may be read and written.
- Function values (Ret_Val) may only be written.

If your program does not conform to these rules, a warning is output during compilation. In this case, convert the relevant input or output parameter to an in/out parameter.

Additional rules apply to the parameter assignment of functions (FC).

See also:

Parameter assignment of functions

Block parameter as edge memory bit in functions (FC)

The above-mentioned rules for parameter assignment must also be observed when edge evaluations are programmed in functions (FC):

As edge memory bit, you require a data value that may be both read and written and that is retained for several cycles. Since input parameters (input) may only be read and output parameters (output) may only be written, these cannot be used as edge memory bits. Temporary local data (Temp) can also not be used as edge memory bits, because it is only available for one cycle.

Therefore, use only in/out parameters (InOut) as edge memory bits in functions (FC). You can also use a single bit of an in/out parameter as edge memory bit, if you address this via a slice access.

The following example shows you how to use an in/out parameter as edge memory bit.

STL

```
FP #InOutFlagStore.x0
```

"InOutFlagStore" is an in/out parameter of the BYTE data type. The address "#InOutFlagStore.x0" addresses the bit address 0, which is used as edge memory bit.

Automatic initialization of block parameters in functions (FC)

In S7-300/400, temporary local data (Temp) of a function must be pre-assigned with a value. Otherwise, it may happen that the program operates with undefined values.

In S7-1500, the risk of continuing to process undefined values is considerably reduced because the parameters below are automatically initialized when the block is called:

- Temporary local data of the STRING and WSTRING data types are always pre-assigned with the maximum length 254 and actual length 0.
- Temporary local data with the elementary data type is automatically initialized in functions (FC) with optimized access. It is then assigned the value that has been predefined for the specified data type. For example, the value "false" is predefined for BOOL. Elements of the PLC data types are pre-assigned with the default value that is specified in the declaration of the PLC data type (UDT). Elements of the data type ARRAY, STRING or WSTRING are pre-assigned with the value "0", even if they are used within a PLC data type.

In S7-300/400, output parameters (Output) in functions had to be assigned a value, because unwanted reactions could otherwise have occurred in the program. When jump instructions or RLO instructions were used, each supported program path then had to be checked.

In S7-1500, the risk of undefined output parameters is considerably reduced because the output parameters of the elementary data type are automatically initialized when blocks are called. They are then pre-assigned the value that was predefined for the specified data type. For example, the value "false" is predefined for BOOL. However, structured output parameters are not pre-assigned with a value. They are transferred as pointer when a block is called and therefore cannot be undefined.

3.4.1.8 Multi-instance capability in S7-1500 (S7-1500)

Multiple instance capability

All function blocks in the CPUs of the S7-1500 series have multiple instance capability. This means FBs that were labeled as not having multiple instance capability in the reference program are converted to blocks with multiple instance capability by the migration. A note is output in the migration log for each converted FB. If you continue to use the FB as a single instance after migration, your program runs unchanged.

3.4.1.9 Floating-point numbers in S7-1500 (S7-1500)

Invalid floating-point numbers

The handling of invalid floating-point numbers in S7-1500 differs from S7-300/400. Example:

- S7-300/400: The result of the expression "Invalid floating-point number <> 1.0" is FALSE.
- S7-1500: The result of the expression "Invalid floating-point number <> 1.0" is TRUE.

Note that the instructions in your program can produce other results due to this difference.

3.4.1.10 Writing individual characters of a STRING to S7-1500 (S7-1500)

Writing individual characters of a STRING

If a character or a byte within a STRING is to be written, an S7-1500 CPU checks whether the target address lies within the actual length of the STRING. If not, the character or byte is not written. The only exception is when the character is written directly after the actual length of the STRING.

The following example shows the string 'hello' with the actual length of 5. The 27th character of the STRING lies outside the actual length and can therefore not be written. The STRING remains unchanged; the result of the assignment is 'hello'.

SCL

```
MyDB.mystring := 'hello';
MyDB.mystring[27] :=
CHAR_TO_BYTE('!');
```

The following example shows the specified exception: The character is written directly behind the STRING to the 6th character. The result of the assignment is 'hello!'.

SCL

```
MyDB.mystring := 'hello';
MyDB.mystring[6] :=
CHAR_TO_BYTE('!');
```

Where possible, use instructions from the "Extended instructions > String + Char" pane to handle STRINGs.

SCL

```
CONCAT(IN1 := 'hello', IN2 := '!');
```

3.4.1.11 Access to the status word in S7-1500 (S7-1500)

Status word

The status word is no longer fully supported in S7-1500. It contains less information and can now be evaluated only in STL. LAD and FBD no longer support the evaluation of the status word. The chapters below provide information on the migration of status word access in the individual programming languages.

See also

Migration of LAD/FBD programs to S7-1500 (Page 185)

Migration of STL programs to S7-1500 (Page 188)

3.4.1.12 Loading software changes to S7-1500 (S7-1500)

Downloading software changes

CPUs of the S7-1500 series behave differently than CPUs of the S7-300/400 series with regard to the downloading of software changes. For example, they allow changes to be downloaded in STOP and RUN without affecting the actual values of previously loaded tags.

3.4.2 Migration of instructions to S7-1500 (S7-1500)

3.4.2.1 Information on migrating instructions (S7-1500)

Automatic migration of instructions

During migration, the instructions used in the initial program are carried over to the new program as much as possible. In the process, necessary adaptations are made automatically whenever possible without altering the semantics of the initial program. If multiple versions of an instruction exist, the migration always uses the newest instruction version.

Instructions that are not available in S7-1500 are replaced automatically whenever possible by a compatible or similar instruction.

Migration scenarios

The following table shows the possible scenarios for migrating instructions:

Class	Scenario	System reaction	Manual adaptation of the program
1	The instruction is identical in both CPU families.	The instruction is migrated.	No adaptation is required. The semantics of the migrated program is unchanged and can be compiled.
2	The instruction is no longer available in S7-1500, but there is a compatible, new instruction.	The instruction is replaced by the new instruction. Additional instructions that are necessary to preserve the semantics of the initial program may be inserted. The migration log reports the replacement.	No adaptation is required. The semantics of the migrated program is unchanged and can be compiled.
3	The instruction is no longer available in S7-1500, but there is a similar instruction.	The instruction is replaced by the similar instruction. The migration log reports the replacement. Locations to be checked are marked in the program. You receive information regarding necessary adaptations in the form of a comment at the appropriate location in the program or when compiling.	You must check the program and possibly adapt it.
4	The instruction is no longer available in S7-1500 and there is no similar instruction.	The instruction is highlighted in red. The program cannot be compiled.	You must check the program and possibly adapt it.

Information regarding instructions of migration class 3

Some instructions that were available in S7-300/400 are no longer offered in S7-1500 because there are now easier or more efficient methods available for solving certain tasks. The PLC migration supports you during the migration by replacing these instructions with appropriate new instructions. However, the instructions from scenario 3 cannot be completely migrated automatically. For example, new formal parameters may need to be added. The locations that cannot be migrated automatically are marked in the program. You receive information regarding necessary adaptations in the form of a comment directly at the appropriate location in the program, in the migration log, or when compiling.

The following manual adaptations may be necessary:

- If the new instruction has additional formal parameters, assign these parameters with suitable actual parameters.
- If the new instruction has modified parameter names, change the parameter assignment.
- If the migration cannot unequivocally ensure that the parameter assignment is semantically identical to the reference program, the actual parameters to be checked are marked. Check the semantics at these locations in the program after migration and correct it if necessary.
- If the reference program contains hardware identifiers, for example at the "LADDR" parameter, these are also marked. Enter the new HW identifier at the "LADDR" parameter.
- If the new instruction has modified error codes, adapt the error handling in the program.

Some examples of instructions of migration class 3 can be found in the chapter "Migration of data block instructions (Page 167)".

Information regarding instructions of migration class 4

Some instructions are no longer offered in S7-1500 because the functionality has fundamentally changed. These instructions are not replaced by the migration. They are shown in red in the program and prevent the program from being compiled. You receive information regarding necessary adaptations in the form of a comment directly at the appropriate location in the program, in the migration log, or when compiling.

In the following chapters, you can find an overview of the non-migratable instructions of migration class 4 as well as information on alternative solutions.

Note

Further support

You can find the latest information about PLC migration in the Siemens Industry Online Support:

Migration of controllers:

http://support.automation.siemens.com/WW/view/en/83557459 (http://support.automation.siemens.com/WW/view/en/83557459)

If you need further assistance for PLC migration, please contact SIMATIC Customer Support.

See also

FAQs on migration (http://support.automation.siemens.com/WW/view/en/67858106)

3.4.2.2 Overview of non-migratable instructions (S7-1500)

Introduction

Some instructions from S7-300/400 cannot be migrated to S7-1500. This is the case, for example, when the instructions can be used only in connection with certain modules or when the functionality of the instruction in S7-1500 needs to be implemented differently.

Information on replacing non-migratable instruction (Class 4)

The following are examples of instructions that cannot be migrated and information on their replacement:

Instruction group	Instruction which cannot be migrated	Notes
Other instructions	SETP	Check whether the instruction in your program can be replaced by the "SET_BF" instruction.
	RESETP	Check whether the instruction in your program can be replaced by the "RESET_BF" instruction.
	DRUM_X	Check whether the instruction in your program can be replaced by the "DRUM" instruction.
	TONR_X	Check whether the instruction in your program can be replaced by the "TONR" instruction.
	RESET	Check whether the instruction in your program can be replaced by the "RESET_BF" instruction.
	SET	Check whether the instruction in your program can be replaced by the "SET_BF" instruction.
	WSR	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	SHRB	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	RESETI	Check whether the instruction in your program can be replaced by the "RESET_BF" instruction.
	SETI	Check whether the instruction in your program can be replaced by the "SET_BF" instruction.
Runtime control	COMPRESS	Delete the instruction. In S7-1500, it is no longer necessary to compress the work memory of the CPU or the load memory.
	PROTECT	Configure the protection level and password via the front panel of the S7-1500 CPU or in the module properties of the CPU.
	CIR	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.

Instruction group	Instruction which cannot be migrated	Notes
Date and time	SET_CLKS	Check whether the instruction in your program can be replaced by the "WR_SYS_T" instruction.
	LOC_TIME	Check whether the instruction in your program can be replaced by the "RD_SYS_T" instruction.
	BT_LT	Check whether the instruction in your program can be replaced by the "RD_SYS_T" instruction.
	LT_BT	Check whether the instruction in your program can be replaced by the "RD_SYS_T" instruction.
	S_LTINT	Check whether the instruction in your program can be replaced by the "SET_TINTL" instruction.
	SET_SW	Check whether the instruction in your program can be replaced by the "WR_SYS_T" or "RD_SYS_T" instruction.
	SET_SW_S	Check whether the instruction in your program can be replaced by the "WR_SYS_T" or "RD_SYS_T" instruction.
	TIMESTMP	Check whether the instruction in your program can be replaced by the "WR_SYS_T" or "RD_SYS_T" instruction.
	WS_RULES	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
Distributed I/O	D_PRAL	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	ASi_3422	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	SALRM	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
Module parameter assignment	PARM_MOD	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
Interrupts	MP_ALM	Multicomputing is not available on S7-1500.
	REPL_VAL	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.

Instruction group	Instruction which cannot be migrated	Notes
Alarms	EN_MSG	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	DIS_MSG	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	ALARM_SQ	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction. You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
	ALARM_S	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction. You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
	ALARM_SC	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction. You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
	WR_USMSG	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	READ_SI	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	DEL_SI	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	ALARM_DQ	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction. You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
	ALARM_D	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction. You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
	NOTIFY_8P	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction. You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
	ALARM	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction. You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
	ALARM_8	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.

Instruction group	Instruction which cannot be migrated	Notes
		You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
	ALARM_8P	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction. You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
	NOTIFY	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction. You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
	AR_SEND	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction. You can find additional information in the chapter "Migrating alarms with associated values (Page 172)" and from SIMATIC Customer Support.
Diagnostics	RDSYSST	The instruction "RDSYSST: Read system status list" is no longer available in S7-1500 as the CPUs of this product series do not provide system status lists. However, you can read some of the information using the "GET_DIAG", "Geo2Log", "DeviceStates", or "ModuleStates" instruction. You can find additional information in the chapter "Migration of the instruction RDSYSST (Page 169)" and from SIMATIC Customer Support.
	OB_RT	Check whether the instruction in your program can be replaced by the "RD_SYS_T" instruction.

Instruction group	Instruction which cannot be migrated	Notes
Table functions	ATT	The instructions cannot be replaced. If you need the instructions in your program, you have to implement them yourself. For additional support, refer to the chapter "Migration of table instructions (Page 175)" and to SIMATIC Customer Support.
	FIFO	
	TBL_FIND	
	LIFO	
	TBL	
	TBL_WRD	
	WRD_TBL	
	DEV	
	CDT	
	TBL_TBL	
	PACK	
Addressing	LGC_GADR	The instructions cannot be used for the modules behind gateways (for example, IE/PB link). Instead, use the instructions "GEO2LOG" and "LOG2GEO". The "LGC_GADR" and "GADR_LGC" instructions continue to work for modules that are not connected through gateways (for example, IE/PB link) to the CPU.
	GADR_LGC	
iSlave	SET_ADDR	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
PID Control	TUN_EC	Check whether the instructions in your program can be replaced by the "PID_Compact" or "PID_3Step" technology objects.
	TUN_ES	

Instruction group	Instruction which cannot be migrated	Notes
Function modules	CNT_CTRL	Check whether the instructions in your program can be replaced by the "High_Speed_Counter" technology object in conjunction with the "S7-1500 TM Count 2x24V" or "TM PosInput 2" counter module. Alternatively, check whether the required functionality can be implemented with the "ET 200S 1Count24V" or "1Count5V" module.
	DIAG_INF	
	CNT_CTL1	
	CNT_CTL2	
	CNT2WRPN	
	CNT2RDPN	
	CNT2_CTR	
	DIAG_RD	
	ABS_INIT	Check whether the functions can be implemented by "S7-1500 Motion" or "ET 200S 1PosU". Check whether the functions can be implemented using the "TM Count 2x24V" or "TM PosInput 2" modules.
	ABS_CTRL	
	ABS_DIAG	
	CAM_INIT	Check whether the encoder acquisition is possible via the modules "S7-1500 TM Count 2x24V" or "TM PosInput 2" and the use of the two DQs for faster reactions. Additional DQs of standard DQs can be switched depending on the encoder value from the application program.
	CAM_CTRL	
	CAM_DIAG	
	PID_FM	Check whether the instructions in your program can be replaced by the technology objects "PID_Compact", "PID_3Step" or PID basic functions.
	FUZ_355	
	FORCE355	
	READ_355	
	CH_DIAG	
	PID_PAR	
	CJ_T_PAR	
	FMT_PID	
	FMT_PAR	
	FMT_CJ_T	
	FMT_DS1	
	FMT_TUN	
	FMT_PV	
300C functions	ANALOG_300C	Check whether the instructions in your program can be replaced by the instructions from the "Motion Control" group.
	DIGITAL_300C	
	COUNT_300C	Check whether the instructions in your program can be replaced by the "High_Speed_Counter" technology object in connection with counter modules.
	FREQUENC_300C	
	PULSE_300C	Check whether the instruction in your program can be replaced by the "CTRL_PWM" instruction.
	SEND_RK_300C	The communication instructions of the S7-1500 do not support RK 512. If you need RK functionality, you must implement it yourself.
	FETCH_RK_300C	
	SERVE_RK_300C	

Instruction group	Instruction which cannot be migrated	Notes
Communication with iSlave/iDevice	I_GET	The instructions cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	I_PUT	
	I_ABORT	
S7 communication	PRINT	Check whether the instructions in your program can be replaced by the instructions of S7 communication for S7-1500. The "CONTROL" and "C_CNTRL" instructions, for example, can be replaced by the "T_DIAG" instruction.
	START	
	STOP	
	RESUME	
	STATUS	
	USTATUS	
	CONTROL	
	C_CNTRL	
Open user communication	TCON_PAR	These instructions are not required for the Open User Communication in S7-1500. The connection parameters are set in the block properties.
	TADDR_PAR	
	TCP_conn_active	
	TCP_conn_passive	
	ISOonTCP_conn_active	
	SOonTCP_conn_passive	
	ISOonTCP_conn_CP_active	
	ISOonTCP_conn_CP_passive	
	UDP_local_open	
	UDP_rem_address_and_port	
Communications processor	Instructions of the SIMATIC NET CPs group Instructions of the PTP CP 340, PTP CP341, PTP CP 440, PTP CP 441 groups	You can find more detailed information on these instructions in the chapter "Migration of communication instructions (Page 179)".
PROFINET / CBA	PN_IN	PROFINET/CBA communication is not supported by S7-1500. Use S7 communication or open user communication.
	PN_OUT	
	PN_DP	
MPI communication	X_SEND	MPI communication is not supported by S7-1500. Use S7 communication or open user communication.
	X_RCV	
	X_GET	
	X_PUT	
	X_ABORT	
Global data communications	GD_SND	Global data communications is not supported by S7-1500. Use S7 communication or open user communication.
	GD_RCV	

Instruction group	Instruction which cannot be migrated	Notes
Point-to-point communication	P_PRINT	The print function is not supported by the point-to-point communication modules.
	P_PRINT341	The print function is not supported by the point-to-point communication modules.
	MODB_341	These blocks for CP 341/CP 441 are not supported by S7-1500. Use the instructions for Modbus (RTU) instead. • Modbus_Comm_Load • Modbus_Master • Modbus_Slave In connection with the S7-1500 point-to-point modules • CM PtP RS232 HF • CM PtP RS422/485 HF You can find more detailed information on these instructions in the chapter "Migrating point-to-point program blocks (Page 183)".
	MODB_441	
	S_MODB	These instructions for ET 200S 1SI are not supported by S7-1500. Instead, use the instructions for Modbus (RTU) or for USS in connection with the point-to-point module "ET 200SP CM PtP". You can find more detailed information on these instructions in the chapter "Migrating point-to-point program blocks (Page 183)".
	S_USST	
	S_USSR	
	S_USSI	
Teleservice	PG_DIAL	The MPI interface is not supported by S7-1500 as the default interface. The instruction is no longer available.
	AS_DIAL	The MPI interface is not supported by S7-1500 as the default interface. The instruction is no longer available.
	SMS_SEND	The MPI interface is not supported by S7-1500 as the default interface. Check whether the instruction in your program can be replaced by the "TMAIL_C" instruction.
	AS_MAIL	The MPI interface is not supported by S7-1500 as the default interface. Check whether the instruction in your program can be replaced by the "TMAIL_C" instruction.
User library	PNIO_DIAG	The "PNIO_DIAG" instruction could be downloaded to a user library for S7-300/400 and used for diagnostic purposes. "PNIO_DIAG" has been replaced in S7-1500 by integrated diagnostics and is thus no longer needed. In S7-1500, the system diagnostics is enabled by default. You can use the system diagnostics without making any additional settings. If required, you can edit the settings for the system diagnostics in the device configuration of the S7-1500 on the "Properties > Diagnostics" tab. You can also configure the interrupts of the system diagnostics there. The interrupts are enabled by default. Following the migration, you can delete the diagnostic screens used in conjunction with "PNIO_DIAG" and the associated tags in the HMI Panel. Use the "System diagnostics display" control in your diagnostic screens instead.

Note

Support for instructions that cannot be migrated

If you need further assistance for instructions that cannot be migrated, please contact SIMATIC Customer Support.

You can find the latest information about PLC migration in the Siemens Industry Online Support:

Migration of controllers:

http://support.automation.siemens.com/WW/view/en/83557459 (http://support.automation.siemens.com/WW/view/en/83557459)

See also

FAQs on migration (http://support.automation.siemens.com/WW/view/en/67858106)

3.4.2.3 Migration of data block instructions (S7-1500)

Procedure for migration of data block instructions (migration class 3)

S7-1500 provides a revised set of instructions of the "Data blocks" category. These instructions are replaced automatically during the migration. Afterwards you still have to make manual adaptations.

The table below provides an overview of the migration of data block instructions:

Instruction in S7-300/400	Instruction in S7-1500	Migration class
CREA_DBL	CREATE_DB	3
CREAT_DB	CREATE_DB	3
CREA_DB	CREATE_DB	3
DEL_DB	DELETE_DB	3
TEST_DB	ATTR_DB	3

NOTICE

Asynchronous processing

The new instructions work asynchronously in S7-1500, which means their execution is spread across several programming cycles.

Manual adaptations after migration

The instructions have additional formal parameters in S7-1500 which are required for asynchronous execution, for example . You need to provide these additional parameters with appropriate actual parameters following migration.

The block number of the created DB must be between 60000 and 60999 in S7-1500. Therefore, adapt the "LOW_LIMIT" and "UP_LIMIT" parameters to the new number range.

You can find a detailed description of the parameters in the reference help for S7-1500.

See also:

AUTOHOTSPOT

AUTOHOTSPOT

Example

The following example shows the migration of the "CREAT_DB" instruction to the "CREATE_DB" instruction.

In the first figure, you can see the call for "CREAT_DB" before migration:

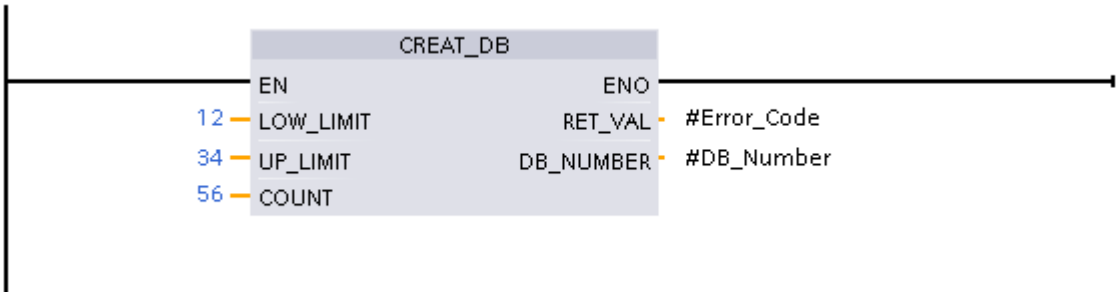

In the second figure, you can see the call for "CREATE_DB" after migration:

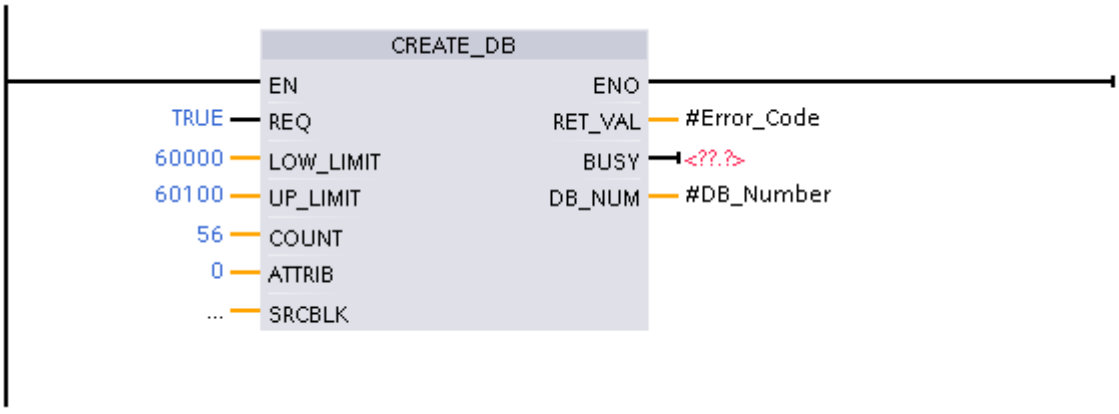

- The "REQ" parameter was not available for "CREAT_DB" and was amended by the migration. It gets the default value "TRUE".
- The "LOW_LIMIT" and "UP_LIMIT" parameters were outside the permitted number range of 60000 to 60999. The values "12" and "34" were therefore marked as invalid during the migration. The permitted values "60000" and "60100" were then entered manually.
- The "ATTRIB" parameter was not available for "CREAT_DB" and was amended by the migration. It gets the default value "0". You can specify the properties of the DB to be created at this parameter.

- The "SRCBLK" parameter was not available for "CREAT_DB" and was amended by the migration. Here you can specify the data area whose contents are to be written to the DB to be generated.
- The "BUSY" parameter was not available for "CREAT_DB" and was amended by the migration. The parameter retains the signal "1" until the process for generating the data block is complete. To evaluate the signal, you need to interconnect the output parameter with an operand.

3.4.2.4 Migration of the instruction RDSYSST (S7-1500)

Procedure for migration of RDSYSST (migration class 4)

The instruction "RDSYSST: Read system status list" is no longer available in S7-1500 as the CPUs of this product series do not provide system status lists. However, the option exists to use one of the following instructions to read out some of the information you received previously via the system status list:

- "RDREC" (Extended instructions > Distributed I/O)
- "GET_DIAG" (Extended instructions > Diagnostics)
- "GEO2LOG" (Extended instructions > Diagnostics)
- "DeviceStates" (Extended instructions > Diagnostics)
- "ModuleStates" (Extended instructions > Diagnostics)
- "RALRM" (Extended instructions > Distributed I/O)
- "LED" (Extended instructions > Diagnostics)

Reading system status information in S7-1500

The following table shows which information about the system status can be read in S7-1500.

You will find the IDs that you have previously read with the RDSYSST instruction in the "SSL-ID" column. The two final digits are relevant. The "xy" placeholder varies depending on the module used.

The "Migration" column indicates whether the information is still available in S7-1500 and how you can read it out.

Partial list	SZL-ID	Migration
Module ID	W#16#xy11	This information can be read with the "RDREC" instruction. Read data record 16#AFF0 in the "INDEX" parameter.
CPU characteristics	W#16#xy12	This information is not currently available in S7-1500.

Partial list	SZL-ID	Migration
User memory areas	W#16#xy13	This information is not currently available in S7-1500.
System areas	W#16#xy14	This information is not currently available in S7-1500.
Block types	W#16#xy15	This information is not currently available in S7-1500.
Identification of one component	W#16#xy1C	This information can be read with the "RDREC" instruction for central modules and PROFINET IO. Read data record 16#AFF0 in the "INDEX" parameter.
Interrupt status	W#16#xy22	This information is not available in S7-1500.
Assignment of partial process images and OBs	W#16#xy25	This information is not available in S7-1500.
Communication: Status data	W#16#xy32	This information is not available in S7-1500.
H-CPU group information	W#16#xy71	Not relevant
Status of the module LEDs	W#16#xy74	This information can be read with the "LED" instruction. Read data record W#16#xy74 in the "LED" parameter.
Switched DP slaves in the H-system	W#16#xy75	Not relevant
DP master system information	W#16#xy90	This information is not available in S7-1500.
Module status information	W#16#xy91	This information can be read with the "GET_DIAG" or "ModuleStates" instruction.
Rack/station status information	W#16#xy92	This information can be read with the "GET_DIAG" or "DeviceStates" instruction.
Rack/station status information	W#16#xy94	This information can be read with the "GET_DIAG" or "DeviceStates" instruction.
Extended DP master system information	W#16#xy95	This information is not available in S7-1500.
Module status information PROFINET IO and PROFIBUS DP	W#16#xy96	This information can be read with the "GET_DIAG", "ModuleStates" or "DeviceStates" instruction.
Tool changer information (PROFINET IO)	W#16#xy9C	This information is not available in S7-1500.
Diagnostics buffer	W#16#xyA0	This information is not available in S7-1500.
Module diagnostics information (data record 0)	W#16#00B1	This information can be read with the "GET_DIAG" or "RDREC" instruction. Within the interrupt OB, you also get the diagnostics information using "RALRM".
Module diagnostics information (data record 1), physical address	W#16#00B2	This information can be read with the "RDREC" or "GEO2LOG" instruction. Within the interrupt OB, you also get the diagnostics information using "RALRM".

Partial list	SZL-ID	Migration
Module diagnostics information (data record 1), logical address	W#16#00B3	This information can be read with the "GET_DIAG" or "RDREC" instruction. Within the interrupt OB, you also get the diagnostics information using "RALRM".
Diagnostics data of a DP slave	W#16#00B4	This information can partially be read with the "ModuleStates" or "GET_DIAG" instruction. Within the interrupt OB, you also get the diagnostics information using "RALRM". Some of the information is not available in S7-1500.

Note

Support on information that cannot be read

If you need further assistance on information that cannot be read, please contact SIMATIC Customer Support.

You can find the latest information about PLC migration in the Siemens Industry Online Support:

Migration of controllers:

http://support.automation.siemens.com/WW/view/en/83557459 (http://support.automation.siemens.com/WW/view/en/83557459)

See also

FAQs on migration (http://support.automation.siemens.com/WW/view/en/67858106)

3.4.2.5 Migration of the T_CONV instruction (S7-1500)

Extracting weekdays from DATE_AND_TIME with T_CONV

The "T_CONV" instruction is available both in S7-300/400 and in S7-1500. The instruction is migrated automatically in most cases. However, if you have used "T_CONV" to extract the weekday from the DATE_AND_TIME (DT) data type, you need to note the following in S7-1500:

In S7-300/400, an output parameter "OUT" of the data type INT was expected for the extraction of the weekday. In S7-1500, the output parameter must be of the data type DTL instead when you are expecting a weekday as a result.

If the output parameter "OUT" has a different data type, an error is output in the migration log during the PLC migration. In this case, convert the data type of the output parameter to DTL and use the component "Weekday".

For additional information, refer to "See also".

3.4.2.6 Migrating alarms with associated values (S7-1500)

Alarm configuration with S7-1500

In S7-1500, you can carry out all functions for alarm configuration centrally with the instruction "Program_Alarm".

"Program_Alarm" replaces the following instructions of S7-300/400:

Instruction for alarm configuration (S7-300/400)	Instruction for alarm configuration (S7-1500)
ALARM_SQ	Program_Alarm (migration class 4)
ALARM_S	
ALARM_SC	
ALARM_DQ	
ALARM_D	
NOTIFY_8P	
ALARM	
ALARM_8	
ALARM_8P	
NOTIFY	
AR_SEND	

Procedure for the migration of alarms

Alarms are not migrated automatically. You need to manually reconfigure the alarms after migration.

Below, you will learn how to prepare your program in such a way that alarm texts are retained after migration and manual follow-up work is minimal.

The procedure only applies for alarm instructions that use the data types "C_Alarm_8", "C_Alarm_8p", "C_Alarm_t"; "C_Notify_8p" or "C_Ar_send".

To migrate alarms, follow these steps:

1. Open the block containing the alarm.
   The interface of the block contains a parameter that has an alarm data type (e.g. the data types "C_Alarm_s", " C_Notify", etc.). This parameter defines the alarm number input.
2. Move the parameter for the alarm number input to the "Static" section.

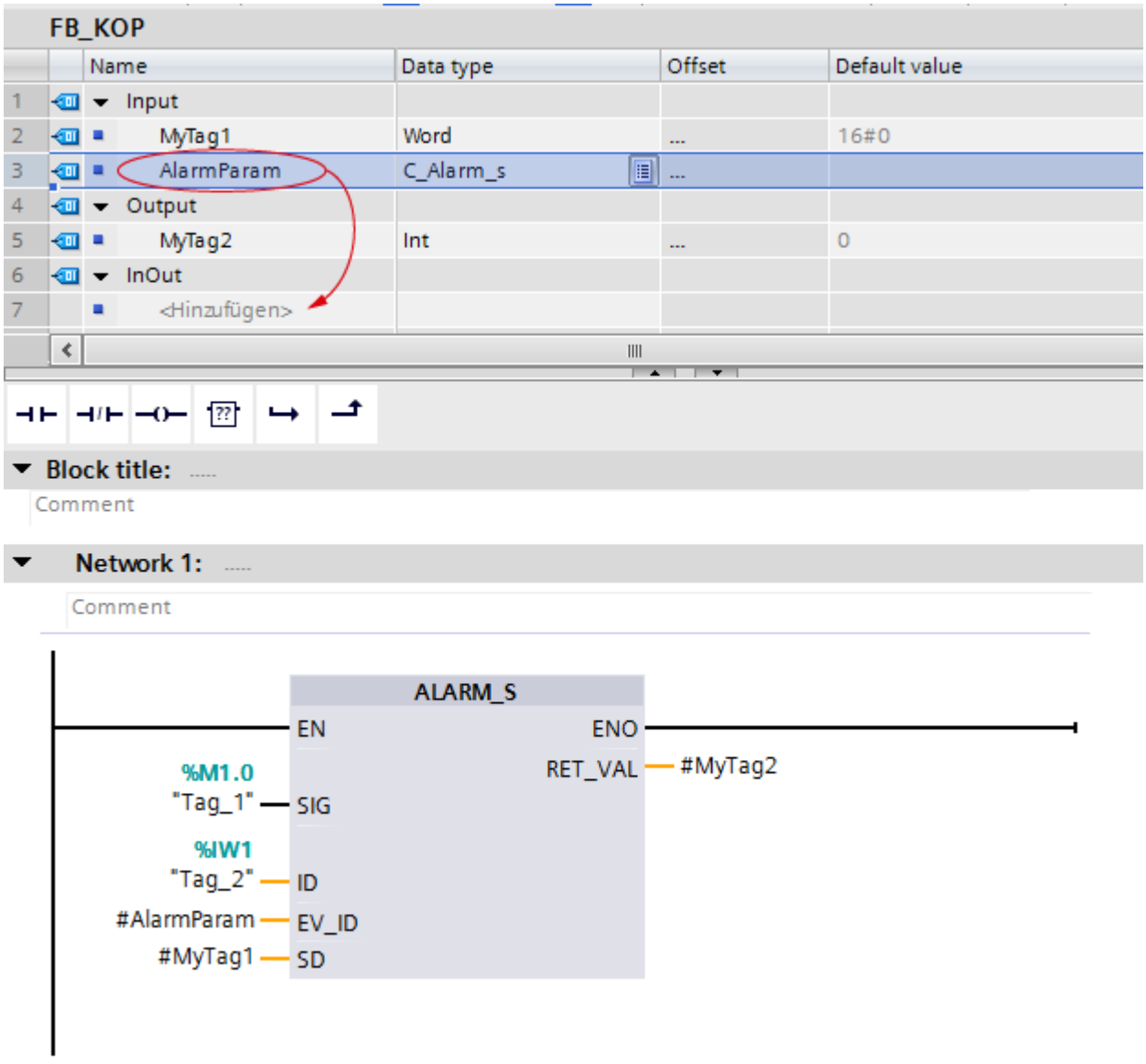

3. Save and compile the block.
4. You then perform the PLC migration.
   See also: Performing migration (Page 146)
5. In the newly created CPU, open the block that contains the alarm again.
   The instruction for alarm configuration could not be migrated and is shown in red.

6. Replace the alarm data type (e.g., "C_Alarm_s") in the block interface with the new data type "Program_Alarm". If the data type "Program_Alarm" is not offered in the selection list, enter the name using the keyboard.
7. Open the "Instructions" task card and navigate to the "Alarms" folder in the "Extended instructions" pane.
8. Drag the "Program_Alarm" instruction to your network.
9. In the "Call options" dialog, select the previously re-configured alarm tag in the "Name in the interface" input field.
10. Open the tag properties and change to the "Alarm" tab.
    The alarm text from the output program is entered.
11. Delete the character for the element type from the associated values (e.g., Y, W, X, I, etc.).
    Associated values do not include information on the element type in S7-1500.
    See also: AUTOHOTSPOT
12. Delete the old alarm block from the network.
    The alarm configuration was migrated. The alarm text has been taken from the output program.

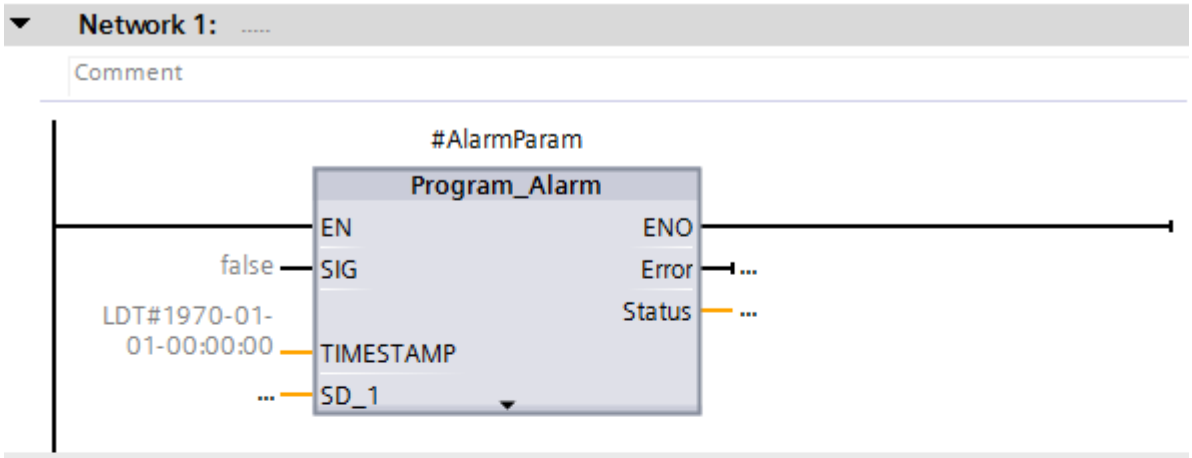

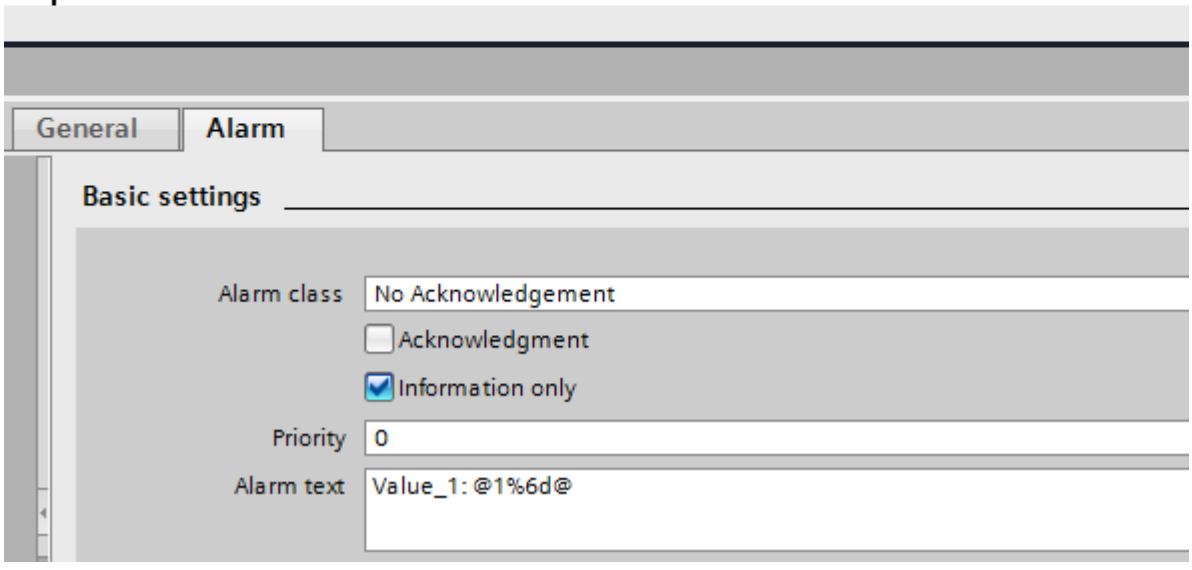

3.4.2.7 Migration of table instructions (S7-1500)

Example for migration of "ATT: Add value to table" (S7-1500)

Procedure for migration of table instructions (migration class 4)

The instructions from the "Table instructions" group are no longer available in S7-1500 and cannot be replaced during the migration because they are based on pure absolute addressing. However, S7-1500 provides convenient functions that allow you to implement the instructions yourself. Below is an example for the implementation of the instruction "ATT": Add value to table".

Implementation

The following SCL program shows how to implement the "ATT" instruction. The function in this example has been implemented as an FC in SCL.

The following table shows the declaration of the tags used:

Operand		Declaration	Data type	Description
val		Input	INT	Entered value
tab		InOut	PLC data type "Table10"	Table with 10 rows
	x	InOut	ARRAY[0..9] of INT	Each ARRAY element represents a row. An ARRAY of INT is used in the example. However, you can create ARRAYs with any data type. An ARRAY of STRUCT or an ARRAY of a PLC data type is also possible.
	lng	InOut	INT	Length of the buffer. The value depends on the number of ARRAY elements of "tab.x".
	first	InOut	INT	First written value
	last	InOut	INT	Last written value
h		Temp	INT	Auxiliary tag
Ret_Val			BOOL	Function value

The following table shows the FC "Attend":

The function writes the value of the input parameter "#val" to the last row of the table.

```
SCL
#h:=(#tab.last+1) MOD #tab.lng;    // increment index
IF #h <> #tab.first THEN          // Scan to see if indices are overtaking each
other
   #tab.x[#tab.last] := #val;      //
Write value
   #tab.last := #h;                // Save index of the next free element
   #Attend:= false;
ELSE
  #Attend:= true;  // Error >> The table is full; the value cannot be entered
END_IF;
```

Example for migration of "FIFO: Output first value of the table" (S7-1500)

Procedure for migration of table instructions (migration class 4)

The instructions from the "Table instructions" group are no longer available in S7-1500 and cannot be replaced during the migration because they are based on pure absolute addressing. However, S7-1500 provides convenient functions that allow you to implement the instructions yourself. Below is an example for the implementation of the instruction "FIFO": Output first value of the table".

Implementation

The following SCL program shows how to implement the "FIFO" table function. The function in this example has been implemented as an FC in SCL.

The following table shows the declaration of the tags used:

Operand		Declaration	Data type	Description
val		Output	INT	Value returned
tab		InOut	PLC data type "Table10"	Table with 10 rows
	x	InOut	ARRAY of INT [0..9]	Each ARRAY element represents a row. An ARRAY of INT is used in the example. However, you can create ARRAYs with any data type. An ARRAY of STRUCT or an ARRAY of a PLC data type is also possible.
	lng	InOut	INT	Length of the buffer. The value depends on the number of ARRAY elements of "tab.x".
	first	InOut	INT	First written value
	last	InOut	INT	Last written value

Operand	Declaration	Data type	Description
h	Temp	INT	Auxiliary tag
Ret_Val		BOOL	Function value

The following table shows the FC "Fifo":

The function returns the first value written to the table (first in, first out). The function value "true" indicates that the table is empty.

SCL

```
#h:=(#tab.first+1) MOD #tab.lng; // increment index
  IF #h <> #tab.last THEN        // Scan to see if indices are
overtaking each other
    #tab.first := #h;    ·       // Save index of the next value
    #val := #tab.x[#h];      //
Output desired value
    #Fifo:= false;
  ELSE
    #Fifo:= true; // Error >> The table is empty; no value is
output
    #val := #tab.x[#tab.first]; //
Output last valid value
  END_IF;
```

Example for migration of "LIFO: Output last value in the table" (S7-1500)

Procedure for migration of table instructions (migration class 4)

The instructions from the "Table instructions" group are no longer available in S7-1500 and cannot be replaced during the migration because they are based on pure absolute addressing. However, S7-1500 provides convenient functions that allow you to implement the instructions yourself. Below is an example for the implementation of the instruction "LIFO": Output last value in the table".

Implementation

The following SCL program shows how to implement the "LIFO" table function. The function in this example has been implemented as an FC in SCL.

The following table shows the declaration of the tags used:

Operand	Declaration	Data type	Description
val	Output	INT	Value returned
tab	InOut	PLC data type "Table10"	Table with 10 rows

Operand		Declaration	Data type	Description
	x	InOut	ARRAY of INT [0..9]	Each ARRAY element represents a row. An ARRAY of INT is used in the example. However, you can create ARRAYs with any data type. An ARRAY of STRUCT or an ARRAY of a PLC data type is also possible.
	lng	InOut	INT	Length of the buffer. The value depends on the number of ARRAY elements of "tab.x".
	first	InOut	INT	First written value
	last	InOut	INT	Last written value
h		Temp	INT	Auxiliary tag
Ret_Val			BOOL	Function value

The following table shows the FC "Lifo":

The function returns the last value written in the table (last in, first out). The function value "true" indicates that the table is empty.

SCL

```
#h:=(#tab.last-1) MOD #tab.lng;           // decrement index
  IF #h<0 THEN #h:=#h + #tab.lng; END_IF; // allow only posi-
tive indices
   IF #h <> #tab.first THEN               // Scan to see if
indices are overtaking each other
    #tab.last := #h;                      // Save index of
the next value
    #val := #tab.x[#h];            //
Output desired value
    #Lifo:= false;
  ELSE
    #Lifo:= true;  // Error >> The table is empty; no value
is output
    #val := #tab.x[#tab.last];           //
Output last valid value
  END_IF;
```

3.4.2.8 Migration of communication instructions (S7-1500)

SIMATIC NET CP/CM (S7-1500)

Migrating program blocks (S7-1500)

Introduction

S7-300/400 uses specific program blocks for communication functions via SIMATIC NET CPs. S7-1500 provides a revised set of instructions for these communication functions.

The following tables provide an overview of the migration of the program blocks used in S7-300/400.

You can find more detailed information about the specification in the "Class" column in chapter Information on migrating instructions (Page 156).

Program blocks for industrial Ethernet / PROFINET

Communication service / functional area	Type of instruction	Library		Instruction in S7-1500	Class
		CP 300	CP 400		
SEND / RECEIVE (Open communication services)	AG_SEND Migration AG_SEND/ AG_LSEND (Page 181)	x	x	T_SEND	3
	AG_LSEND Migration AG_SEND/ AG_LSEND (Page 181)	x	x	T_SEND	3
	AG_SEND (via UDP) Migration AG_SEND/ AG_LSEND (Page 181)	x	x	TUSEND	3
	AG_SEND (e-mail) Migration AG_SEND/ AG_LSEND (Page 181)	x	x	TMAIL_C	3
	AG_RECV Migration AG_RECV/ AG_LRECV (Page 183)	x	x	T_RCV	3
	AG_LRECV Migration AG_RECV/ AG_LRECV (Page 183)	x	x	T_RCV	3
	AG_LOCK	x	x	-	4
	AG_UNLOCK	x	x	-	4
	AG_CNTRL	x	x	T_DIAG, T_RESET, TCON, TDISCON	3
Programmed communication connections	IP_CONFIG	x	x	T_CONFIG	3
FTP	FTP_CMD	x	x	FTP_CMD	3

Communication service / functional area	Type of instruction	Library		Instruction in S7-1500	Class
		CP 300	CP 400		
PROFINET IO	PNIO_SEND	x		-	4 *)
	PNIO_RECV	x		-	4 *)
	PNIO_RW_REC	x		-	4 *)
	PNIO_ALARM	x		-	4 *)

*) Other access mechanisms should be used.

Program blocks for PROFIBUS

Communication service / functional area	Type of instruction	Library		Instruction in S7-1500	Case
		CP 300	CP 400		
SEND / RECEIVE (Open communication services)	AG_SEND Migration AG_SEND/ AG_LSEND (Page 181)	x	x	T_SEND	3
	AG_LSEND Migration AG_SEND/ AG_LSEND (Page 181)	-	x	T_SEND	3
	AG_RECV Migration AG_RECV/ AG_LRECV (Page 183)	x	x	T_RCV	3
	AG_LRECV Migration AG_RECV/ AG_LRECV (Page 183)	-	x	T_RCV	3
PROFIBUS DP	DP_SEND	x	-	-	4 *)
	DP_RECV	x	-	-	4 *)
	DP_DIAG	x	-	-	4 *)
	DP_CTRL	x	-	-	4 *)

*) Other access mechanisms should be used.

Additional formal parameters

The instructions have additional formal parameters in S7-1500 which are required for asynchronous execution, for example . You need to provide these additional parameters with appropriate actual parameters following migration.

You can find a detailed description of the parameters in the reference help for S7-1500.

See also:

AUTOHOTSPOT

Example

You can find a basic example of the representation of a migrated instruction in the chapter Migration of data block instructions (Page 167)

Migration AG_SEND/AG_LSEND (S7-1500)

Migration of the call parameters - communication via ISO Transport, ISO-on-TCP, TCP

Table 3-1 Migration AG_SEND/AG_LSEND

AG_SEND / AG_LSEND		T_SEND		Description / notes
Parameter	Type	Parameter	Type	
ACT	BOOL	REQ	BOOL	
ID	INT	ID	CONN_OUC (WORD)	
LADDR	WORD	--	--	
SEND	ANY	DATA	VARIANT	
--	--	ADDR	VARIANT	
LEN	INT	LEN	UINT	
DONE	BOOL	DONE	BOOL	
--	--	BUSY	BOOL	
ERROR	BOOL	ERROR	BOOL	
STATUS	WORD	STATUS	WORD	

Additional steps

- Configure the corresponding connection type with the relevant address parameters.

Migration of the call parameters - communication via UDP

Table 3-2 Migration AG_SEND/AG_LSEND

AG_SEND / AG_LSEND		TUSEND		Description / notes
Parameter	Type	Parameter	Type	
ACT	BOOL	REQ	BOOL	
ID	INT	ID	CONN_OUC (WORD)	
LADDR	WORD	--	--	
SEND	ANY	DATA	VARIANT	
--	--	ADDR	VARIANT	
LEN	INT	LEN	UINT	
DONE	BOOL	DONE	BOOL	
--	--	BUSY	BOOL	

AG_SEND / AG_LSEND		TUSEND		Description / notes
Parameter	Type	Parameter	Type	
ERROR	BOOL	ERROR	BOOL	
STATUS	WORD	STATUS	WORD	

Additional steps

- Configure the corresponding connection type with the relevant address parameters.

Migration of the call parameters - communication via E-MAIL

Table 3-3 Migration AG_SEND/AG_LSEND

AG_SEND / AG_LSEND		TMAIL_C		Description / notes
Parameter	Type	Parameter	Type	
ACT	BOOL	REQ	BOOL	
ID	INT	--	--	
LADDR	WORD	--	--	
SEND	ANY	TO_S	STRING	
SEND	ANY	CC	STRING	
SEND	ANY	SUBJECT	STRING	
SEND	ANY	TEXT	STRING	
SEND	ANY	ATTACHMENT	VARIANT	
--	--	ATTACHMENT _NAME	STRING	
--	--	MAIL_ADDR_PARAM	VARIANT	
LEN	INT	--	--	
DONE	BOOL	DONE	BOOL	
--	--	BUSY	BOOL	
ERROR	BOOL	ERROR	BOOL	
STATUS	WORD	STATUS	WORD	

Additional steps

- Configure the corresponding connection type with the relevant address parameters.

Migration AG_RECV/AG_LRECV (S7-1500)

Migration of call parameters

Table 3-4 Migration AG_RECV/AG_LRECV

AG_RECV / AG_LRECV		TRCV / TURCV		Description / notes
Parameter	Type	Parameter	Type	
		EN_R	BOOL	
ID	INT	ID	CONN_OUC (WORD)	
LADDR	WORD	--	--	
--	--	LEN	UDINT	
RECV	ANY	DATA	VARIANT	
--	--	ADDR	VARIANT	
LEN	INT	LEN	UINT	
NDR	BOOL	NDR	BOOL	
--	--	BUSY	BOOL	
ERROR	BOOL	ERROR	BOOL	
STATUS	WORD	STATUS	WORD	

Additional steps

- Configure the corresponding connection type with the relevant address parameters.

Point-to-point CP/CM (S7-1500)

Migrating point-to-point program blocks (S7-1500)

Introduction

S7-300/400 uses specific program blocks for communication functions via point-to-point CPs.

S7-1500 provides a revised set of instructions for these communication functions.

During migration, you are asked whether you want to use the revised set of instructions for the communication modules of the S7-1500 or the program blocks for the communication processors of the S7-300/400.

The table below provides an overview of the migration of the program blocks used in S7-300/400 on instructions of the S7-1500.

You can find more detailed information about the specification in the "Class" column in chapter Information on migrating instructions (Page 156).

Program blocks for point-to-point communication

Communication service / functional area	Block	Library			Instruction in S7-1500	Class
		CP 300	CP 400	ET 200S		
Blocks for CP 340	P_RCV	x	-	-	Receive_P2P	3 [1]
	P_SEND	x	-	-	Send_P2P	3 [1]
	P_RESET	x	-	-	Receive_Reset	3 [1]
	V24_STAT_340	x	-	-	Signal_Get	3 [1]
	V24_SET_340	x	-	-	Signal_Set	3 [1]
	P_PRINT	x	-	-	-	4
Blocks for CP 341	P_RCV_RK	x	-	-	Receive_P2P	3 [1]
	P_SND_RK	x	-	-	Send_P2P	3 [1]
	V24_STAT	x	-	-	Signal_Get	3 [1]
	V24_SET	x	-	-	Signal_Set	3 [1]
	P-PRT341	x	-	-	-	4
Blocks for CP 440	RECV_440	-	x	-	Receive_P2P	3 [1]
	SEND_440	-	x	-	Send_P2P	3 [1]
	RES_RECV	-	x	-	Receive_Reset	3 [1]
Blocks for CP 441	V24_STAT_441	-	x	-	Signal_Get	3 [1]
	V24_SET_441	-	x	-	Signal_Set	3 [1]
Blocks for Modbus	MODB_341	x	-	-	-	4 [2]
	MODB_441	-	x	-	-	4 [2]
Blocks for ET 200S 1SI	S_RCV	-	-	x	Receive_P2P	3 [1]
	S_SEND	-	-	x	Send_P2P	3 [1]
	S_VSTAT	-	-	x	Signal_Get	3 [1]
	S_VSET	-	-	x	Signal_Set	3 [1]
	S_XON	-	-	x	Port_Config	3 [1]
	S_RTS	-	-	x	Port_Config	3 [1]
	S_V24	-	-	x	Send_Config	3 [1]
	S_MODB	-	-	x	-	4 [2]
	S_USST	-	-	x	-	4 [2]
	S_USSR	-	-	x	-	4 [2]
	S_USSI	-	-	x	-	4 [2]
Blocks for compact CPUs S7-300	SEND_PTP	x	-	-	Send_P2P	3 [1]
	RCV_PTP	x	-	-	Receive_P2P	3 [1]
	RES_RCVB	x	-	-	Receive_Reset	3 [1]

[1] You can find information on migrating this instruction at this link (Page 156).

[2] These instructions for ET 200S 1SI are not supported by S7-1500. Instead, use the instructions for Modbus (RTU) or for USS in connection with the point-to-point module "ET 200SP CM PtP".

Additional formal parameters

The instructions have additional formal parameters in S7-1500. You need to provide these additional parameters with appropriate actual parameters following migration.

Note

Additional support for migration of communication instructions

You can find the latest information about migrating communication instructions in the Siemens Industry Online Support.

Migration of communication:

http://support.automation.siemens.com/WW/view/en/83558087 (http://support.automation.siemens.com/WW/view/en/83558087)

Migration of controllers:

http://support.automation.siemens.com/WW/view/en/83557459 (http://support.automation.siemens.com/WW/view/en/83557459)

If you need further support, please contact SIMATIC Customer Support.

See also

FAQs on migration (http://support.automation.siemens.com/WW/view/en/67858106)

3.4.3 Migration of LAD/FBD programs to S7-1500 (S7-1500)

3.4.3.1 Information on migrating LAD/FBD programs (S7-1500)

---(SAVE): Save RLO to BR memory

The instruction is not available in S7-1500. Migration handles the instruction as follows:

- If the "---(SAVE)" instruction was used in the last network of a block, it is replaced by the "---(RET)" instruction with the parameter "RLO".
- If the "---(SAVE)" instruction was used in a network that is not the last network of the block, it cannot be replaced. Change the program manually. For example, save the RLO in a PLC tag, a DB tag or a local tag and return it to the calling block at the end of the block with the --(RET) instruction.

---(CALL) Call block without parameters

The instruction is not available in S7-1500. Migration adapts the program so that the call is implemented via a box.

Master control relay

The master control relay is not available on S7-1500. Migration reports an error. Change the program manually. Use temporary tags, for example, to conditionally execute instructions or networks.

Language change within a block

The contents of the registers, accumulators and the status word are only available within STL networks. When an LAD or FBD network follows an STL network, you cannot access the register contents that were previously set in STL from the LAD or FBD network. However, the register contents are again available in a downstream STL network. The result of logic operation is an exception: It is set to "undefined" at a language change and is no longer available in downstream networks.

See also: Passing values using registers in case of language change (Page 192)

3.4.3.2 Partially qualified addresses in LAD/FBD (S7-1500)

Introduction

The addressing of DB tags without specification of the DB name is referred to as partially qualified addressing. Partially qualified addressing is not supported in S7-1500 for LAD/FBD. The "---(OPN)" and "---(OPNI)" instructions, which were often used in conjunction with partially qualified addressing, are not supported in S7-1500 either.

Migration of partially qualified addresses

Migration handles partially qualified addresses as follows, if possible:

- If the data block can be uniquely determined, migration replaces partially qualified addresses with full addresses and specifies the DB.
- If the DB is specified indirectly by the data block register, the data block cannot be clearly determined. In this case, migration adds a parameter of the data type "DB_Any" to the block interface. The data block name is passed to this parameter. The migration then replaces all partially qualified addresses with addresses having the following syntax: `#<parameter name>.%<absolute address>.`

Example

The following example shows migration of partially qualified addresses. In the first figure, you can see the program before migration:

Network 1: Tags are addressed partially qualified.

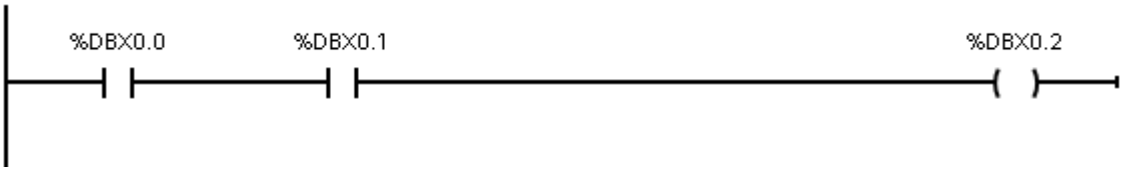

The following figure shows the program after migration:

Network 1: The DB containing the tags could be determined unambiguously. The program was migrated automatically. The tags in "DB1" are addressed fully qualified.

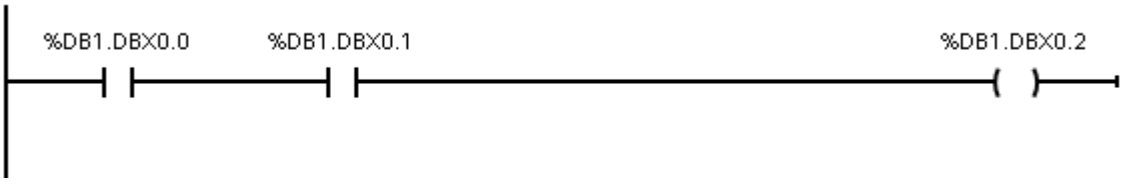

Note

Migration of partially qualified block parameters

The migration also converts partially qualified DB parameters into fully qualified parameters. Note that the type of parameter transfer to the called block can change because of this: The called block may possibly no longer access the actual parameter directly using the fully qualified address but rather work with a copy that is transferred during the block call.
You should therefore check whether the semantics of the migrated program match that of the reference program.
Additional information is available under: Access to block parameters during program execution

If this adaptation is not desired, you can, for example, use a parameter with structured data type rather than an elementary block parameter.
Define PLC data type (UDT) as formal parameter and transfer a tag of this type or a DB that was derived from the PLC data type (UDT).

3.4.3.3 Access to status bits in LAD/FBD (S7-1500)

Introduction

To achieve the highest possible program execution performance on CPUs of the S7-1500 series, values are only passed between blocks by means of the block interface, by using global data blocks or PLC tags. It is not possible to pass values by using the status word in LAD and FBD.

Migration of the instructions "--| |--: Get status bit" / "--|/|--: Get negated status bit"

The instructions are not available in S7-1500. The migration distinguishes between the following cases:

- If a status bit query is placed directly after a mathematical instruction in the same network, a comparator is used instead.
- In all other cases the program is faulty after migration. You have to change your program in this case. For example, use a query of the ENO output instead of a query of the OV status bit. If you link the negation of several ENO queries with OR, you can replace the query of the OS status bit.

Example

The following example shows the migration of a status bit query. In the first figure, you can see the program before migration.

Network 1: The "<=0" instruction queries the status bit "A1". It indicates whether the result of multiplication is less than or equal to zero.

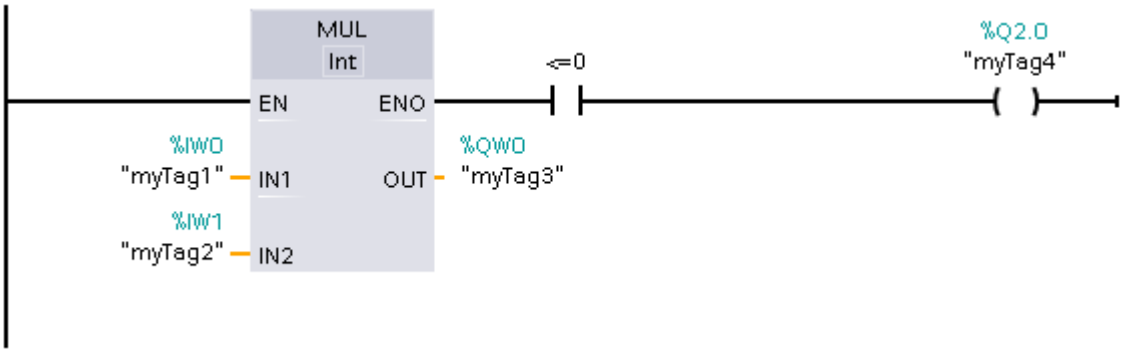

The following figure shows the program after migration.

Network 1: The program was able to be migrated automatically. The "<=0" instruction was removed. Instead, the result of the multiplication is queried by using a comparator with the "myTag3" tag.

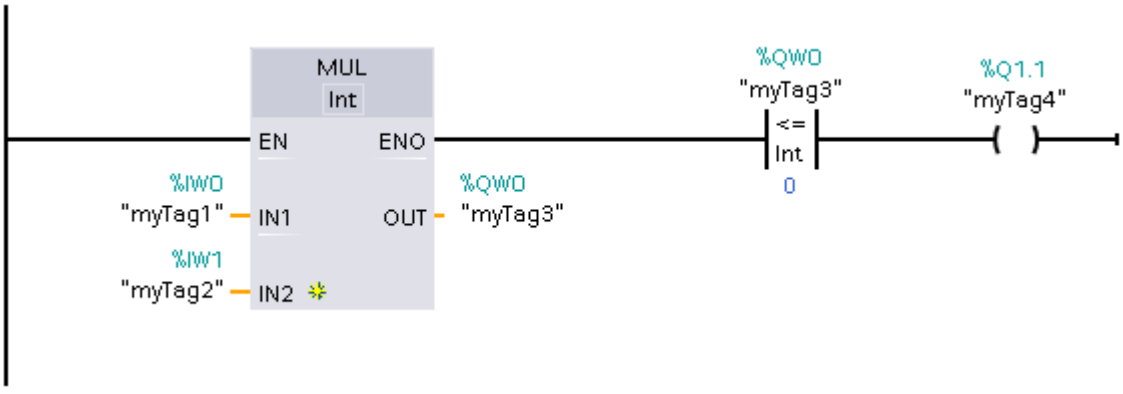

3.4.4 Migration of STL programs to S7-1500 (S7-1500)

3.4.4.1 Information on migrating STL programs (S7-1500)

Passing values using registers and status words

To achieve the highest possible program execution performance on CPUs of the S7-1500 series, values are only passed between blocks by means of the block interface, by using global data blocks or PLC tags.

There is no way to pass values using registers (for example, accumulators, AR1, AR2, DB, DI) or the status word in LAD and FBD. In STL, values can be passed with some restrictions. Keep in mind that program execution slows down when you use these areas to pass values between various blocks.

The following rules apply for STL:

- The contents of the registers, accumulators and the status word are only available within STL networks. When an LAD or FBD network follows an STL network, you cannot access the register contents that were previously set in STL from the LAD or FBD network. However, the register contents are available again in a downstream STL network.
  The RLO bit is an exception: It is set to "undefined" at a language change and is no longer available in downstream networks.
- The values from the registers, accumulators and the status word are never transferred to called blocks. The instructions "CC" and "UC" are the only exceptions. If you use "UC" or "CC" and want to pass parameters to the called block by using registers, the status word or accumulators, you have to enable the "Parameter passing via registers" option in the properties of the called block. Note that this option is only available for STL blocks with standard access and the block can have no formal parameters. When the option is enabled, you can transfer register contents between blocks. The RLO bit is an exception here as well: It is set to "undefined" when the block is exited and is no longer available after a block call.
- To pass an error message to the calling block, you can use the BR bit. You first need to store the error message in the calling block with the instruction "SAVE" in BR bit. You can then read the BR bit in the calling block.
- The data block register DB is set to "0" after each access to a data block with specification of a fully qualified address (for example, `%DB10.DBW10`). Subsequent partially qualified access leads to an error during compilation.
- If you symbolically address a local formal parameter (e.g. with the instruction L #myIn) from the block interface of an FB in S7-1500, you always access the data block that you specified as an instance in the block call. Although the OPN DI, L AR2, +AR2, TDB, TAR instructions change the contents of the DI or address register, the registers are not evaluated when addressing local formal parameters.

You can find programming examples in the sections below.

Master control relay

The master control relay is not available on S7-1500. Migration reports an error. Change the program manually. Formulate conditions, for example, in the form of block parameters to conditionally execute instructions or networks.

LEAVE and ENT

The "LEAVE" and "ENT" instructions are not available in S7-1500 because only two accumulators are available. Migration reports an error. Change the program manually. Use temporary tags, for example, to store intermediate results.

Block parameters of the "Block_DB" parameter type

The "Block_DB" parameter type is not available in S7-1500. Migration changes this parameter and assigns it the "DB_Any" data type instead.

It is not possible to match library instructions with the call of an instance in the form of a parameter of the "DB_Any" type in S7-1500.

The following example shows the call of a library block with a variable instance in an S7-300/400 CPU. This sequence cannot be migrated to S7-1500.

STL	Explanation
CALL GET, #myBlock_DB	/ / The library block "GET" is called with the instance data block that is currently applied at the "myBlock_DB" block parameter.
REQ := #Start	
ID := W#16#100	
NDR := #Done	
ERROR := #Error	
STATUS := #Status	
ADDR_1 := P#DB10.DBX5.0 BYTE 10	
ADDR_2 :=	
ADDR_3 :=	
ADDR_4 :=	
RD_1 := P#DB10.DBX6.0 BYTE 10	
RD_2 :=	
RD_3 :=	
RD_4 :=	

3.4.4.2 Logic operation sequences and jumps (S7-1500)

Introduction

In S7-300/400, it is possible to place jump labels in STL at any instruction to enable a program-controlled jump to the labeled position. However, if you define the jump destination within a logical program unit in which the RLO does not have a defined state, unexpected results can occur during program execution. The CPU may switch to STOP state.

You must therefore take steps during program creation to ensure that jump labels are only set at the start of a logic operation sequence. You should also avoid jumps from the middle of a logic operation sequence.

The CPU families behave differently when a jump occurs in a logic operation sequence:

- With the S7-300, the first input bit scan behavior of an instruction after a jump label is determined by the instructions executed linearly beforehand. It does not matter here whether the instruction after the jump label was reached linearly or by means of a jump.
- With the S7-400, the first input bit scan behavior of an instruction after a jump label is determined by the actually executed program flow. The first input bit scan behavior can therefore depend on whether the jump label was reached linearly or by means of a jump.
- In S7-1500, a compilation error was output when programs contained jumps out of or into logic operation sequences which could lead to different runtime behavior when executed on an S7-300 or S7-400. The program must be adapted at this point in order to determine the intentional behavior of the program.

Example

The following example shows a jump to a logic operation sequence. The instruction in line 6 at the jump label n_OK does not have clearly defined first input bit scan behavior. It is thus not possible to form an RLO. This program sequence would lead to a compilation error on an S7-1500.

```
1        A     #Motor_Enabled
2        AN    #Motor_EmergencyStop
3        JU    n_OK
4        =     #Motor_Start
5        O     #Motor_SpeedHigh
6  n_OK: ON    #Motor_BrakesEnabled
7        =     #Motor_Stop
```

Migration of jumps to logic operation sequences

The migration does not change the jump labels of your STL program. However, an error is reported when compiling if jump label conflicts are detected. Check the program logic of your block at the reported points of conflict. Use the SET or CLR instructions to label the start of a new "logical unit" and to force a first input bit scan.

Recommendation

Avoid jumps in logic operation sequences, for example, to optimize the program code. This makes the code too confusing and hard to maintain.

3.4.4.3 Passing values using registers in case of language change (S7-1500)

Introduction

The contents of the registers, accumulators and the status word are only available within STL networks. When an LAD or FBD network follows an STL network, you cannot access the register contents that were previously set in STL from the LAD or FBD network. However, the register contents are available again in a downstream STL network.

The RLO bit is an exception: It is set to "undefined" at a language change and is no longer available in downstream networks.

Migration of value passing using registers in case of language change

When a register is accessed in a migrated LAD or FBD network, an error is reported during compilation. Change the program in such a way that registers are only set and read in STL networks.

Example

The following example shows the migration of an accumulator access. In the first two figures, you can see the program before migration:

Network 1: The "myIN1" operand is downloaded to accumulator 1. Then "myIN2" is downloaded to accumulator 1, which transfers "myIN1" to accumulator 2. Now both values are added together. The result is stored in accumulator 1 and from there it is assigned to the "myOUT" operand.

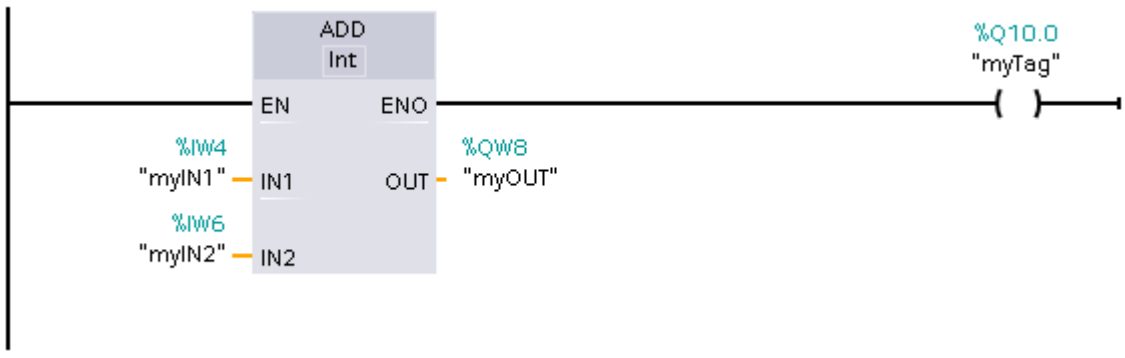

Network 2: The "myIN3" operand is downloaded to accumulator 1. The existing "myOUT" operand is thereby transferred to accumulator 2 and can be immediately added to the "myIN3" operand. The "myOUT" operand does not need to be explicitly downloaded again.

```
1    L    "myIN3"     %IW12
2    +I
3    T    "myResult"  %IW14
4
```

After the migration, network 2 is faulty because values are no longer written to the accumulators by the LAD network. The program must be adapted manually so that both values are explicitly loaded to the accumulators in STL.

The following figure shows network 2 after correction of the error.

Network 2: The two values are explicitly loaded to the accumulator in STL before they are added together.

```
1    L    "myIN3"     %IW12
2    L    "myOUT"     %QW8
3    +I
4    T    "myResult"  %QW14
5
```

3.4.4.4 Passing values using registers or the status word with a block call (S7-1500)

Migration of passing values using registers or the status word with a block call

The values from registers, accumulators and the status word are set to "0" or given the status "undefined" at a block change. This means they cannot be passed to called blocks.

The instructions "CC" and "UC" are the only exceptions. If you use "UC" or "CC" and want to pass parameters to the called block by using registers, the status word or accumulators, you have to enable the "Parameter passing via registers" option in the properties of the called block. Note that this option is only available for STL blocks with standard access and the block can have no formal parameters. When the option is enabled, you can transfer register contents between blocks. The RLO bit is an exception: It is always set to "undefined" at a block transition and is no longer available after a block call.

To pass an error message to the calling block, you can use the BR bit. You first need to store the error message in the calling block BR bit. To do this, use the instructions "SAVE" or "JNB", for example. You can then read the BR bit in the calling block.

An error is reported if you access register contents that were set in the called block after a block call. In this case, change your program. Use the tags in data blocks or PLC tags, for example, to return values to the calling block.

Examples

The following example shows the changes you need to make to your program in order to pass values to a calling block via registers.

In the first table, you can see the program prior to migration:

STL	Explanation

`CALL "MyFB", "MyFB_DB"`
 	
`= #MyBit`
	The RLO of the "MyFB" block is assigned to the "MyBit" operand after execution.

The second table shows you which changes you need to make to the program

STL	Explanation
CALL "MyFB", "MyFB_DB"	In the called "MyFB" block, write the currently pending RLO to the BR bit with the instruction "SAVE" at any location.
A BR	The BR bit is read in the calling block.
= #MyBit	The value of the BR bit is assigned to the "MyBit" operand.

3.4.4.5 Passing values using registers with the CC and UC instructions (S7-1500)

Introduction

In S7-300/400, block calls can be programmed using the instructions UC and CC. The parameters are not passed to the calling block via the interface, but rather via registers, for example, the AR1, AR2, DB, DI, accumulators or the status word.

These calls slow down program execution and can therefore no longer be set as the default in S7-1500. The CALL instruction is used instead in S7-1500. But the CALL instruction cannot be used to implement indirect block calls. If you wish to use UC or CC for indirect block calls, you need to select the "Parameter passing via registers" option in the properties of the called block. It is then possible for you to transfer register contents between different blocks. The RLO bit is an exception: It is set to "undefined" at a block transition and is no longer available after a block call.

Note that this option is only available for STL blocks with standard access and the block can have no formal parameters. When the option is enabled, you can transfer register contents between blocks. The RLO bit is an exception here as well: It is set to "undefined" at a block transition and is no longer available after a block call.

It is no longer necessary to use indirect block calls for the creation of know-how protected library elements. The TIA Portal supports you here automatically: Any number conflicts that occur with blocks that are already in the user program are automatically resolved when a block is inserted from a library.

Migration of block calls via "UC" or "CC"

The migration handles block calls via "UC" or "CC" as follows:

- The "UC FC" instruction with specification of a block number is replaced by the "CALL" instruction.
- The "CC FC" instruction with specification of a block number is replaced by the "CALL" instruction. A jump instruction is also added that implements the conditional call.
- The "UC FC" and "CC FC" instructions with indirect specification of the block number remain unchanged.
- The "UC FB" and "CC FB" instructions with indirect specification of a block number remain unchanged.

Example

The following example shows migration of block calls via "UC".

In the first table, you can see the program prior to migration:

STL
UC FC 10
UC FC[#temp0]
UC FB 10
UC FB [#temp0]

In the following table, you can see the program after migration:

STL	Description
CALL FC 10	
UC FC[#temp0]	The "Parameter passing via registers" option must be set at the called block.
UC FB 10	The "Parameter passing via registers" option must be set at the called block.
UC FB [#temp0]	The "Parameter passing via registers" option must be set at the called block.

3.4.4.6 Fully qualified addresses in STL (S7-1500)

Introduction

The addressing of DB tags with specification of the DB name or DB number is referred to as fully qualified addressing. The data block register is set to "0" after each access to the data block with specification of a fully qualified address. If you want to access the DB register again after fully qualified access, you must first re-assign a value using the OPEN DB command.

Migration of fully qualified addresses

Migration inserts the "OPN" instruction after fully qualified access, if needed, to reload the current data block into the data block register.

Example

The following example shows migration of a fully qualified address.

In the first table, you can see the program prior to migration:

STL	Explanation
AUF "MyDB"	"MyDB" is loaded to the data block register.
L %DBW1	Data elements from "MyDB" are addressed partially qualified.

STL	Explanation
L "Global_DB".Data1	The elements from a global DB are then addressed fully qualified. "Global_DB" is loaded implicitly to the data block register by this step.
L "Global_DB".DBW2	
T DBW[AR1, P#0.0]	A subsequent access addresses the "Global_DB", because it is in the DB register.

In the following table, you can see the program after migration:

STL	Explanation
AUF "MyDB"	
L "MyDB".DBW1	The partially qualified access is converted into a fully qualified access.
L "Global_DB".Data1	
L "Global_DB".DBW2	The DB register is reset after fully qualified access.
OPN "Global_DB"	The "Global_DB" must be loaded to the DB register again with OPN.
T DBW[AR1, P#0.0]	

3.4.4.7 Partially qualified addresses in STL (S7-1500)

Introduction

The addressing of DB tags without specification of the DB name or DB number is referred to as partially qualified addressing. Partially qualified addressing accesses a defined value in the data block that is currently stored in the DB register.

The following restrictions apply to partially qualified addresses in S7-1500:

- In S7-1500, partially qualified addresses are only allowed if the DB register has been set explicitly in the current block. You set the DB register with the instruction "OPN", for example. In data blocks with standard access, you can only address tags partially qualified.
- The data block register is set to "0" with the block call in S7-1500. This means it is not possible to open a data block in a block and to address data elements from the data block in a subordinate block partially qualified. Set the DB register in the current block before you address a DB tag partially qualified.
- The DB data block register is also to "0" with each fully qualified access (for example, %DB10.DBW10). Set the DB register again after fully qualified access before you address a DB tag partially qualified.
- The use of partly qualified addresses slows down the processing of the program during runtime.

Migration of partially qualified addresses

Migration handles partially qualified addresses in STL as follows:

- The partially qualified address is converted into a fully qualified address, if possible.
- If the data block is opened in the calling code block and the DB cannot be clearly identified, migration inserts a parameter of the "DB_Any" data type in the interface of the called block. The data block name is passed to this parameter. Migration inserts the "OPN" instruction in the called block to open the data block.

Example

The following example shows the migration of partially qualified addresses.

In the first table, you can see the program prior to migration:

STL	Explanation
L DBW10 L DBW12 +I T DBW14	A data block is opened in the calling block and transferred to the DB register. In the current block, the values "DBW10", "DBW12" and "DBW14" are adopted from the data block that is currently open in the DB register.

In the following table, you can see the program after migration:

STL	Explanation
OPN "PlcmigTempBlockDB" L DBW10 L DBW12 +I T DBW14	Migration inserts a parameter of the data type "DB_Any" in the interface of the called block. The data block name is passed to this parameter. Migration inserts the "OPN" instruction in the called block to open the data block passed in the interface.

Note

Migration of partially qualified block parameters

The migration also converts partially qualified DB parameters into fully qualified parameters. Note that the type of parameter transfer to the called block can change because of this: The called block may no longer access the actual parameter directly using the fully qualified address but works with a copy that was transferred during the block call instead.
Make sure that the semantics of the migrated program matches that of the reference program.
Additional information is available under: Access to block parameters during program execution

If this adaptation is not desired, you can, for example, use a parameter with structured data type rather than an elementary block parameter.
Define PLC data type (UDT) as formal parameter and transfer a tag of this type or a DB that was derived from the PLC data type (UDT).

Examples:

```
CALL  "MyFC"
          InStruct :="DBofUDT"
```

or

```
CALL  "MyFC"
          InStruct := "DBArrayOfUDT".a[#i]
```

3.4.4.8 Access to the instance DB in STL (S7-1500)

Introduction

The "OPN DI" or "CDB" instructions place a data block in the DI register. In S7-300/400, the block opened there is considered an instance data block. The subsequent symbolic addressing of a local formal parameter from the block interface of an FB (IN, OUT, InOut, Static) no longer addresses the data block that was specified as an instance in the block call, but rather the data block that is located in the DI register. To subsequently symbolically address a local formal parameter from the block interface, the instance DB must be loaded to the DI register.

Even after the instructions "L AR2", "+ AR" and "TAR", it is not possible in S7-300/400 to symbolically address a formal parameter from the block interface, because the basis for the parameter access has been destroyed by the instructions.

This behavior has been corrected in S7-1500: If you symbolically address a local formal parameter (e.g. with the instruction L #myIn) from the block interface in S7-1500, you always access the data block that you specified as an instance in the block call. Although the OPN DI, L AR2, +AR2, TDB, TAR instructions change the contents of the DI or address register, the registers are not evaluated when addressing local formal parameters.

Migration of accesses to local tags in the instance DB

Migration does not change the programmed accesses. However, if you have changed the instructions OPN DI, L AR2, +AR2, TDB, TAR, etc. in the reference program, the semantics of the program may have changed.
In order to achieve the original semantics again, you must change the program manually. It is often no longer necessary to address data via registers. Instead, avail of the possibility to use ARRAYs within the instance DB and index the ARRAY elements indirectly.

Example 1

The following example shows the modified semantics of the instruction "OPN DI":

STL	Explanation
L #MyIn1 L #MyIn2 +I T #MyOut3	The "L" and "T" instructions address local tags that were declared in the block interface. The values of the tags are located in the instance DB that was specified in the block call.
OPNDI "MyDB" L #MyIn1 L #MyIn2 +I T #MyOut3	The global data block "MyDB" is written to the DI register. The "L" and "T" instructions address tags that were declared in "MyDB" prior to migration. After migration, the "L" and "T" instructions address tags that were declared in the block interface. The DI register is not evaluated for access in S7-1500.

Example 2

The following example shows the modified semantics of the instruction "LAR2":

STL	Explanation
L P#M23.0	
LAR2	Prior to migration, the basis for parameter access is destroyed with the assignment to AR2.
L #MyIn1	Prior to migration, access to "MyIn1" is not possible or results in an error. After migration, access to the formal parameter "MyIn1" is correctly executed.
L IW [AR2, P#1.0]	Access to %IW24.0
A [AR2, P#0.4]	Access to %M23.4

Example 3

The following example shows how a DB tag can be addressed indirectly in S7-1500 without using the address register:

STL	Explanation
OPN "MyDB" L #index LAR1 L DBW [AR1 , P#10.0]	Prior to migration, area-internal indirect register addressing is used. A variable value (#index) is hereby loaded to the address register 1. Depending on this value, a data word from "MyDB" is loaded to accumulator 1.
L "MyDB".MyArray1[#index]	After migration, you can store the data values in "MyDB" in an ARRAY. The individual ARRAY elements can be variably indexed using the input parameter "#index".

3.4.5 Migration of SCL programs to S7-1500 (S7-1500)

3.4.5.1 Information on migrating SCL programs (S7-1500)

Representation of SCL blocks after migration

SCL blocks that were created with S7-300/400 are automatically migrated to S7-1500, if their semantics are clearly interpretable. The program positions that are not clearly interpretable are marked with a question mark after the migration, which results in syntax errors. Check the semantics at these positions in the program after migration and correct them if necessary.

In some cases, the program code is changed by the migration. The corresponding section of code from the original program is then transferred to the migrated block as a comment section. This makes it easy for you to track the changes that were made by the migration.

See also

Basic information on migrating PLC programs (Page 143)

Information on migrating PLC programs (Page 149)

3.4.5.2 Indirect addressing in SCL (S7-1500)

Introduction

The options for indirect addressing were standardized for all programming languages in S7-1500. Indirect addressing that was possible in S7-300/400 is therefore partially converted during the migration.

The following table provides an overview of the conversion options. The migration of individual language constructs is explained below in more detail.

Indirect addressing	S7-300/400	S7-1500
Indirect addressing of a DB	Data type "BLOCK_DB"	"DB_ANY" data type
Indirect addressing of DB tags	#block.%DBW3	"PEEK"/"POKE" instructions
	WORD_TO_BLOCK_DB(#myWord).%DBW3	"PEEK"/"POKE" instructions
	#block.DW(IDX := #myInt)	"PEEK"/"POKE" instructions
	WORD_TO_BLOCK_DB(#myWord).DW(IDX:=#myInt)	"PEEK"/"POKE" instructions
	%DB1.DW(IDX :=#myInt)	"PEEK"/"POKE" instructions
Indirect addressing of I/O	QB(IDX :=#myInt):P	"PEEK"/"POKE" instructions
Indirect addressing of PLC tags	IX(IDX :=#myInt1,Bit:=#myInt2)	"PEEK"/"POKE" instructions
	QB(IDX :=#myInt)	"PEEK"/"POKE" instructions
	MW(IDX :=#myInt)	"PEEK"/"POKE" instructions

Migration of the "BLOCK_DB" data type to "DB_ANY"

The "BLOCK_DB" data type is no longer available in S7-1500. Tags of this data type are converted to the "DB_ANY" data type by the migration. The "BLOCK_DB_TO_WORD" and "WORD_TO_BLOCK_DB" conversion functions are migrated to "UINT_TO_WORD(DB_ANY_TO_UINT)" and "UINT_TO_DB_ANY(WORD_TO_UINT)".

The following examples show the migration of the "BLOCK_DB" data type.

In the first table, you can see three program sections prior to migration:

SCL

```
#myBlock  //Data type  BLOCK_DB;
//...
#myWord :=BLOCK_DB_TO_WORD(#myBlock);
//...
#myBlockDB := WORD_TO_BLOCK_DB(#myWord);
```

In the second table, you can see three program sections after migration:

SCL

```
#myBlock  //Datentyp DB_Any;
//...
#myWord := UINT_TO_WORD(DB_ANY_TO_UINT(#myBlock));
//...
```

SCL

```
#myDBANY:= UINT_TO_DB_ANY(WORD_TO_UINT(#myWord));
```

Migration of indirect addressing of DB tags

Indirect specification of data blocks or DB tags is implemented with the "PEEK"/"POKE" instructions in S7-1500. The migration automatically converts them to indirect addressing.

The following examples show the migration of indirect addressing of DB tags.

In the first table, you can see some program sections prior to migration:

SCL

```
#myWord := %DB1.DW(IDX :=#myInt);
//...
#myBool := %DB1.DX(IDX :=#myByteOffset,Bit:=#myBitOffset);
//...
%DB1.DW(IDX := #myInt) := 12;
//...
```

In the second table, you can see the program sections after migration:

SCL

```
#myWord := PEEK_WORD(area:=16#84, dbNumber:=1, byteOffset:=#myInt);
//...
#myBool:=PEEK_BOOL(area:=16#84, dbNumber:=1, byteOffset:=#myByteOffset, bitOffset:=#myBitOffset);
//...
POKE(area:=16#84,dbNumber:=1,byteOffset:=#myInt,value:=12);
//...
```

Migration of indirect addressing of PLC tags

Indirect specification of PLC tags is implemented with the "PEEK"/"POKE" instructions in S7-1500. The migration automatically converts them to indirect addressing.

The following examples show the migration of indirect addressing of DB tags.

In the first table, you can see a program section prior to migration:

SCL

```
#myWord := MW(IDX := #myInt);
```

In the second table, you can see the program sections after migration:

SCL

```
#myWord := PEEK(area:=16#83,dbNumber:=0,byteOffset:=#myInt);
```

3.4.5.3 Instructions in SCL (S7-1500)

Migration of instructions

S7-1500 has a slightly different set of instructions. Some instructions have been innovated or replaced by new instructions.

During migration, the instructions used in the program are retained as much as possible. If an instruction is not available in S7-1500, an attempt is made to replace it with a compatible or similar instruction. You then have to check the migrated program again and adapt it if necessary.

The following changes may occur, for example:

- The new instruction has additional formal parameters. Configure the additional parameters.
- A formal parameter of the new instruction has a different data type. In this case, the migration automatically adds an instruction to convert the data types.

If the migration cannot unequivocally ensure that the parameter assignment is semantically identical to the initial program, the relevant parameters are commented out. Check the semantics at these locations in the program after migration and correct it if necessary.

The following examples show the migration of the "AG_SEND" instruction to the "TSEND" instruction.

In the first table, you can see a call of "AG_SEND" prior to migration:

SCL

```
AG_SEND (ACT:=#myBool,
         ID:=#myInt1,
         LADDR:=#myWord,
         SEND:=#myAny,
         LEN:=#myInt2,
         DONE=>#myBool,
         ERROR=>#myBool,
         STATUS=>#myWord);
```

In the first table, you can see the call of "TSEND" after migration:

SCL

```
"TSEND_DB"((* ToReplace: REQ:=#myBool *)
         (* ToReplace: ID:=#myInt1 *)
         LEN:=INT_TO_UDINT(#myInt2),
         DONE=>#myBool,
         ERROR:=#myBool,
         STATUS:=#myWord,
         DATA:=#myAny;
```

The actual parameter of "REQ" was applied from the "ACT" parameter of the "AG_SEND" instruction. It is commented out because the program must be checked at this location.

The actual parameter of "ID" was applied from "AG_SEND" and also needs to be tested.

The "LEN" parameter has another data type with "AG_SEND". The migration automatically adds a conversion.

Migration of instructions in expressions

In some cases, migration generates several instructions from a single instruction, for example, when an output parameter has another data type after migration and the formal parameter must be converted. The conversion instruction is inserted after the migrated instruction. This means, however, that the original instruction can no longer be used as an expression. In this case, the migration adds a temporary tag in the interface of the migrated block and assigns it the result of the instruction. This temporary tag can then be used as an expression. Automatic migration cannot always be performed for complex expressions. Expressions that are not clearly interpretable are marked with a question mark after the migration, which results in syntax errors. Check the semantics at these locations in the program after migration and correct it if necessary.

The following examples shows the migration of the "READ_RTM" instruction to the "RTM" instruction.

In the first table, you can see a call of "READ_RTM" prior to migration:

SCL

```
IF READ_RTM(NR:=#myByte, CQ=>#myBool, CV=>#myOutInt)= 1 THEN
...;
END_IF;
```

In the second table, you can see a call of "RTM" after migration.

SCL

```
(* classic code: IF READ_RTM(NR:=#myByte, CQ=>#myBool, CV=>#myOutInt)= 1 THEN*)
#SCL_MIGRA_TEMP_INT_1:=RTM(NR:=#myByte,
                           MODE:=0,
                           PV:=#Migra_PV,
                           CQ:=#myBool,
                           CV:=#Migra_CV);
#myOutInt:=DINT_TO_INT(#Migra_CV);

IF #SCL_MIGRA_TEMP_INT_1= 1 THEN;
...
END_IF;
```

The instruction from the original program is transferred as a comment section to the migrated block. This makes it easy for you to track the changes that were made by the migration.

Because CV has the DINT data type for "RTM", a conversion instruction is inserted that converts the actual parameter from INT to DINT. This means the result of RTM can no longer be used as an expression within the IF instruction.

The temporary tag "SCL_MIGRA_TEMP_INT_1" is inserted in the interface of the migrated block. The result of the RTM instruction is assigned to the temporary tag.

The temporary tag is used as the expression in the IF instruction instead of the "RTM" instruction.

3.4.6 Migration of GRAPH programs to S7-1500 (S7-1500)

3.4.6.1 Information on migration of GRAPH programs (S7-1500)

Introduction

GRAPH programs that were created with S7-300/400 can be migrated automatically to S7-1500. The PLC migration optimizes the programs for running on a CPU of the S7-1500 series. The migration log reports all automatic program changes made during the migration and informs you of any necessary manual changes you still have to perform.

Adaptation of the GRAPH program by the PLC migration

The following list shows some examples of automatic adaptations that are made during the PLC migration. All adaptations are described in detail in the sections belows.

- Adaptation of the block interface
  The interface of the GRAPH blocks is changed slightly. This enables use of the optimized block access of the S7-1500 for GRAPH blocks.
- Integrated symbolic addressing
  SIMATIC S7-1500 is characterized by its integrated symbolic programming. This makes the program more efficient and reduces the risk of access errors.
  For this reason, if the initial program uses addresses for which a symbolic name is not declared, the migration assigns a symbolic name for the address whenever possible.
- Memory model of the GRAPH DB
  Based on the performance data of the S7-1500, it is no longer necessary to use GRAPH DBs with minimized memory space requirements. For this reason, GRAPH no longer offers different memory space models. You have access to the full parameter set and thus the complete functionality in all GRAPH DBs.
  During migration, DBs with minimized memory space requirements are expanded to include the full parameter set.

New functions of the S7-1500

Following migration of a GRAPH program, you can make full use of the functionality of the S7-1500. The following list shows some examples of the new functions:

- Optimized block access
  With optimized block access, the declared data elements are arranged automatically in the available memory area of the block in such a way as to make optimal use of its capacity. The data is structured and saved in a manner that is optimal for the CPU used. This allows you to increase the performance of the CPU. Access errors, from the HMI, for example, are not possible.
  The "Optimized block access" attribute is always enabled for GRAPH blocks in S7-1500 and cannot be deselected.
  See also: AUTOHOTSPOT
- New instructions
  The "Instructions" task card contains the complete set of instructions of the S7-1500. Numerous instructions have been newly developed or innovated. For example, the "Program control" folder in the "Basic instructions" pane contains the "GET_ERROR" instruction for programming local troubleshooting.
  See also: AUTOHOTSPOT
- LAD/FBD functions
  For S7-300/400, GRAPH only supported the programming of a preceding logic operation at the first Boolean input of a box. In S7-1500, preceding logic operations can be programmed for all Boolean inputs.
  The enable output (ENO) can be enabled or disabled selectively for individual instructions.
  See also: AUTOHOTSPOT
- New data types
  S7-1500 provides an expanded set of data types and new data type conversion options.
  See also: AUTOHOTSPOT
- Constants
  S7-1500 provides additional options for constant notation. Thus you can also assign data types to constants.
  See also: AUTOHOTSPOT

3.4.6.2 Changes in the GRAPH program (S7-1500)

Introduction

The PLC migration automatically adapts the GRAPH program to the new CPU family as much as possible. In so doing, both the actions as well as the LAD or FBD networks that you used for programming of conditions are analyzed. Program structures that are no longer permitted are updated. Instructions that are not available on the new CPU are replaced with appropriate S7-1500 instructions.

Symbolic addressing

SIMATIC S7-1500 is characterized by its integrated symbolic programming. This makes the program more efficient and reduces the risk of access errors.

If the initial program uses absolute addresses for which a symbolic name is not declared, the absolute address is marked as invalid. You must then declare a symbolic name before you can compile the migrated program.

If the initial program uses absolute addresses for which a symbolic name is declared, the migration uses the symbolic name instead of the absolute address. Accesses to individual bits within a declared symbolic operand are replaced by slice accesses.

The following example shows an absolute access before the migration:

```
%DB1.DBx1.1
```

The following shows you how the migration replaces the absolute access. Since the "FirstDataByte" tag was declared for the first data byte in "DataBlock1", the access to the first bit within this tag can be replaced by a slice access.

```
"DataBlock1".FirstDataByte.x1
```

Fully qualified addressing

S7-300/400 programs frequently use partially qualified addresses to access addresses indirectly. In this case only the address of a DB tag within a data block is specified. The DB name or DB number is not specified in the address (e.g., `%DBX0.2`). At runtime, the program accesses the data block that is currently stored in the DB register. Partially qualified addressing is not supported in S7-1500. In order to address DB tags indirectly, use block parameters of data type "DB_ANY".

If you have used partly qualified addresses in GRAPH programs, these are reported as errors after the migration. The block cannot be compiled. Then replace the partly qualified address with a fully qualified address.

LAD/FBD elements in conditions

The LAD or FBD networks in Transition, Interlock, Supervision, and Permanent instructions are migrated according to the same rules as the networks in pure LAD or FBD blocks.

See also: Migration of LAD/FBD programs to S7-1500 (Page 185)

GRAPH actions

SIMATIC S7-1500 provides a slightly modified set of instructions. Some instructions have been innovated or replaced by new instructions.

If you have used instructions in GRAPH actions, the migration checks whether these are still supported on S7-1500. The instructions are carried over as much possible. If an instruction is not available in S7-1500, an attempt is made to replace it with a compatible or similar instruction.

Examples of this are the "RLDA: Rotate left by status bit CC 1" and "RRDA: Rotate right by status bit CC 1" instructions. These are replaced by the "ROR: Rotate right" and "ROL: Rotate left" instructions.

If instructions are replaced during the migration, this is noted in the migration log. You then have to check the migrated program again and adapt it if necessary.

See also: Migration of instructions to S7-1500 (Page 156)

3.4.6.3 Adaptation of the interface of GRAPH blocks (S7-1500)

Modification of the block interface during migration

The interface of the GRAPH blocks is adapted during the migration. The adaptations enable you to use the optimized memory concept of the S7-1500 with GRAPH. Optimized block access is always active for GRAPH blocks on the S7-1500 and cannot be deselected. Changes are made in the following areas:

- Static parameters: "STATIC"
- Output parameters: "OUTPUT"
- Retentivity setting for all internal parameters

See also: Blocks with optimized access

If you have used parameter names in the initial program that are reserved in S7-1500, these are renamed during the migration.

Static parameters

Static parameters are adapted as follows during the migration:

- The interface contains a structure in the "Static" section for each step and transition of the GRAPH FB. In S7-300/400, the "GraphStep", "GraphStepMin", "GraphTransition", and "GraphTransitionMin" data types were used. In S7-1500, the new "G7_StepPlus" and "G7_TransitionPlus" data types are introduced for reproducing the step and transition structures.
  The migration proceeds as follows here:
  - The "SNO" and "TNO" parameters are transferred together with their default values from the old structures; all other default values are not transferred. GRAPH programs with a DB with minimized memory space requirements do not contain the "SNO" and "TNO" parameters. If you migrate a DB with minimized memory space requirements, the migration assigns default values for these parameters starting with "1".
  - The locations of use of the modified parameters in the GRAPH program are not adapted automatically but rather are marked as errors. You must manually adapt the locations of use after the migration.
- All other static parameters are transferred to the "RT_DATA" structure. The default values are thus carried over as much as possible and the locations of use in the program are adapted automatically.

Output parameters

Because the output parameter "S_CRITSUP" is no longer supported during criteria analysis on S7-1500, it is removed during migration.

The locations of use in the GRAPH program are not adapted automatically but rather are marked as errors. You must manually adapt the locations of use after the migration.

Retentivity setting for all internal parameters

The retentivity setting of the internal GRAPH parameters affects the behavior of the sequential control system after a power loss:

- A GRAPH block whose parameters are retentive continues with the last active step after a power loss.
- A GRAPH block whose parameters are not retentive starts again with the initial step after a power loss.

GRAPH blocks that were created with S7-300/400 have retentive interface parameters by default.

The retentiveness of the internal parameters is not kept when you migrate the GRAPH FB from an S7-300/S7-400 CPU to an S7-1500 CPU. The internal parameters are declared as non-retentive after the migration.

Proceed as follows after the migration to declare the internal parameters of a GRAPH FB as retentive again:

1. Open the GRAPH FB after the migration.
2. Select the "Retentive internal parameters" command in the "Edit" menu.
   The settings for the internal parameters in the "Retain" column change from "Retain" to "Non-retain" in the block interface of the GRAPH FB.
3. Save and compile your project.
   The settings for the parameters are then applied to all instance DBs.

Migrating S7-1200 to firmware as of V4 (S7-1200) 4

4.1 Basic information on upgrading to V4 (S7-1200)

Introduction

If you have used a CPU with firmware version V3 in your project and want to upgrade to a CPU with firmware V4.0 or later, you can easily replace the device.

The TIA Portal offers the "Change device" function for this purpose. The project remains unchanged when the device is replaced. You can continue using the programs that you created with firmware version V3.

Rules

The following basic rules apply when replacing a device:

- Replacing a device is only possible if the project was created based on a CPU with firmware version V3.0. If your project was created based on firmware version V1.0 or V2.0, create a new CPU with firmware version V3.0 offline in the project, and copy your program to this CPU.
- It is not possible to replace a V4 CPU with a V3 CPU. If you want to continue using the existing V3 CPU, create a copy of this CPU before replacing the device.
- The program cannot be transferred to the new CPU via a memory card. Instead, use the "Change device" function, which is described in the following sections.

HMI panels

Configured HMI panels are treated differently during device replacement depending on the firmware version of the panel and the communication mode.

The following table shows the HMI connections that are supported with the migration:

Firmware version of the panel	PUT/GET communication	Migration to V4
V11 or later	No	S7-1200 does not support this configuration. Upgrade the firmware of the HMI panel to V12.0. Then compile and load the configuration.
V11 or later	Yes	S7-1200 supports this configuration. The connection is established automatically while you compile and load the project after replacing the device.
V12 or later	No	S7-1200 supports this configuration. The connection is established automatically while you compile and load the project after replacing the device.

During compilation of the program, you receive specific information on migrating the HMI panel.

Note

HMI TP 177B 4"

The HMI TP 177B 4" with firmware version V11.0.2 cannot be operated with S7-1200 V4. Replace the panel with a new device, if necessary.

S7-1200 expansion modules

If you are already using the following centrally plugged S7-1200 modules in your system, you must perform a firmware update for these modules in order to guarantee operation with S7-1200 V4.

- ASi - Master - CM 1243
- DP - Master - CM 1243-5
- WAN CP - CP1243-1

Newly shipped S7-1200 modules have the latest firmware installed ex works.

Protected blocks

Blocks equipped with know-how protection or copy protection cannot be converted to V4. If the project contains protected blocks, you must remove the protection prior to migration.

If these are supplied blocks and you do not know the password, ask your supplier either for the password or for a V4-compatible block.

WARNING

Preventing personal injury and material damage

Changes are made to the program during device replacement in some cases. Therefore, thoroughly check the program in a test environment after replacing the device and before putting it in operation.

Note

Additional support

You can find the latest FAQs about migrating to S7-1200 V4 in Siemens Industry Online Support:

http://support.automation.siemens.com/WW/view/en/82140966 (http://support.automation.siemens.com/WW/view/en/82140966)

If you need further assistance with migrating to S7-1200 V4, please contact SIMATIC Customer Support.

See also

Migrating to V4 (Page 214)
Special considerations after migrating to V4 (Page 216)

4.2 Migrating to V4 (S7-1200)

Requirement

- A CPU with firmware version V3 is available in the project.
- The project contains no protected blocks.

Procedure

Proceed as follows to replace a CPU:

1. Select the V3 CPU you want to replace.
2. Select the "Change device" command in the shortcut menu.
   The "Change device" dialog box appears.
3. Under "New device" in the tree structure, select the V4 CPU with which you want to replace your current V3 CPU.
4. Click "OK".
   The existing CPU is replaced by the new one.
5. Select the new CPU and select the "Compile > Hardware and software (only changes)" command in the shortcut menu.
   The device configuration and the user program are compiled again.
6. Optional: If necessary, apply know-how protection or copy protection to individual blocks in the program.
7. Select the new CPU and select the "Download to device > Hardware and software (only changes)" command in the shortcut menu.
   The device configuration and the user program are loaded into the new CPU.
   This completes the device replacement.

WARNING

Preventing personal injury and material damage

Changes are made to the program during device replacement in some cases. Therefore, thoroughly check the program in a test environment after replacing the device and before putting it in operation.

Note

Additional support

You can find the latest FAQs about migrating to S7-1200 V4 in Siemens Industry Online Support:

http://support.automation.siemens.com/WW/view/en/82140966 (http://support.automation.siemens.com/WW/view/en/82140966)

If you need further assistance with migrating to S7-1200 V4, please contact SIMATIC Customer Support.

See also

Basic information on upgrading to V4 (Page 211)

Special considerations after migrating to V4 (Page 216)

4.3 Special considerations after migrating to V4 (S7-1200)

Functional changes in V4

S7-1200 V4 offers significantly enhanced functionality. The most important functional changes that you need to consider after migrating from V3 are described briefly below.

You can find more information on S7-1200 in the "SIMATIC S7-1200 Automation System" system manual:

https://support.industry.siemens.com/cs/ww/en/view/65601780 (http://support.automation.siemens.com/WW/view/en/65601780)

Organization blocks

With S7-1200 V4, you can specifically set the interruptibility of each organization block used. When a device is replaced, all the organization blocks are configured as non-interruptible, to ensure that the executability of your V3 program remains unchanged. The OB priorities from the V3 program also remain unchanged. After migration, you can change the settings for priority and interruptibility as needed.

The behavior of diagnostic interrupts in V4 has changed as follows:

In V3, the start information always contained information on the triggering module, including the channel number. In V4, this information is only generated for a pending diagnostics event. If no diagnostic event is pending, for example, because the fault has already been corrected, only the triggering module is indicated.

Access levels

S7-1200 V4 offers an extended access level concept. The following table shows how the protection levels of the V3 firmware are indicated in V4:

V3 protection level	V4 access level	Meaning
No protection	Full access (no protection)	Unrestricted access without password protection.
Read-only	Read access	HMI access and unimpeded communication between CPUs without password protection. A password is required for changes (write access) in the CPU and for changing the operating mode of the CPU (RUN/STOP).
Write/read protection	HMI access	HMI access and unimpeded communication between CPUs without password protection. A password is required to read the data in the CPU, for changes (writing) in the CPU, and for changing the operating mode of the CPU (RUN/STOP).
-	No access (complete protection)	No access without password input. A password is required for HMI access, for reading the data in the CPU, for changing (writing) data in the CPU, and for changing the operating mode of the CPU (RUN/STOP).

Instruction libraries

After migration to S7-1200 V4, instructions from the libraries of the firmware version V3 are still available. This ensures that you can continue to use your program unchanged. In addition, S7-1200 V4 offers many new instructions which are also compatible with the instructions of the S7-1500.

You can find more information on the instruction libraries in the "SIMATIC S7-1200 Automation System" system manual:

https://support.industry.siemens.com/cs/ww/en/view/65601780 (http://support.automation.siemens.com/WW/view/en/65601780)

Motion Control

When a device is replaced, the Motion Control objects from the libraries of the firmware versions V1 and V2 are replaced with the corresponding objects from the V3 libraries. The objects from V3 libraries are compatible so that you can continue to use the programs unchanged.

The libraries of the S7-1200 V4 offer many new Motion Control functions, which are compatible with the functions of the S7-1500. If you want to use V4 libraries, select them on the "Instructions" task card after replacing the device.

You can find more information on the new Motion Control functions in the "SIMATIC S7-1200 Programmable controller" system manual.

https://support.industry.siemens.com/cs/ww/en/view/65601780 (http://support.automation.siemens.com/WW/view/en/65601780)

Web server

The following settings for operation via a web server are transferred from the V3 CPU to the V4 CPU during device replacement:

- Activate web server on this module
- Permit access only with HTTPS

If you want to operate the V4 CPU via a web server, you need to set up user accounts with assigned user rights in the user management. Only the standard web pages are available to standard users without any additional rights.

Note

Additional support

You can find the latest FAQs about migrating to S7-1200 V4 in Siemens Industry Online Support:

http://support.automation.siemens.com/WW/view/en/82140966 (http://support.automation.siemens.com/WW/view/en/82140966)

If you need further assistance with migrating to S7-1200 V4, please contact SIMATIC Customer Support.

Communication via PUT/GET

Communication via PUT/GET is enabled after the device replacement. Note that the new integrated connection types offer a security standard higher than PUT/GET communication. If you do not use PUT/GET communication, you should disable it.

See also

Basic information on upgrading to V4 (Page 211)

Migrating to V4 (Page 214)

Index

A

Action
    Migrate, 116, 118
Alarm classes
    Migrate, 114
    Name change through migration, 62
Alarm group
    Migration, 61
Alarm groups
    Migrate, 114
Alarm log
    Migrating, 68
Alarm system
    Migrate, 114
Alarms
    Migrate, 114
Allen-Bradley DF1
    Migrating data types, 76
Allen-Bradley DH485
    Migrating data types, 77
Allen-Bradley Ethernet IP
    Migrating data types, 77
Area pointer
    Migration, 58

B

Basic Process Control
    Migrate, 125
Basics
    Migration, 42, 90

C

Check whether project can be migrated, 22
Cleaning project for migration, 23
Color settings
    Migrate, 113
Compiling
    Migrated project, 46
Connection
    Change after migration, 57, 101
    Migrate, 99
Consistency check before migration, 22
Controls
    Migrate, 112
Converting an unspecified CPU, 136
Cycles
    Migrate, 124

D

Data types
    Migration, 75, 126
Deleting a connection, 141
Device name
    Changing by the migration, 94
Dynamic
    Migrate, 109

F

Faceplate
    Migrate, 104
FBD
    Migration, 32
Filtering of project languages, 95
Font settings
    Migrate, 55, 106
Function
    Migrate, 116

G

GE Fanuc SNP
    Migrating data types, 78
Global library
    Migrating, 67
GRAPH
    Migration, 32

H

Header file
    Migrate, 118
HMI connection, 137
HMI device replacement
    by the migration, 52

I

Integrated project
    Converting an unspecified CPU, 136

Creating an integrated HMI connection, 137
Deleting an unspecified connection, 141
Linking HMI tags, 139
Migrate, 73, 129, 131
Migrating, 72, 133
Post-editing, 135
IX, 201

L

LAD
Migration, 32
Languages
Migrating, 65
LG GLOFA GM
Migrating data types, 78
Line report
Migrate, 123
Linking HMI tags, 139
Log
Migrate, 120
Log data
Migrating, 68

M

Migrated project
Compiling, 46
Starting, 94
Migrating alarms to S7-1500, 172
Migrating PLC programs to S7-1500, 156, 158, 167, 169, 179, 183, 200, 201, 203
Migrating projects
Procedure, 15
Requirements, 15
Migration, 27, 32, 34, 156, 158, 167, 169, 179, 183, 200, 201, 203
Actions, 116, 118
Adaptations beforehand, 48
Alarm classes, 114
Alarm groups, 61, 114
Allen-Bradley DF1 data types, 76
Allen-Bradley DH485 data types, 77
Allen-Bradley Ethernet IP data types, 77
Area pointer, 58
Basics, 42, 90
Changing the device name, 94
Color settings, 113
Connection, 99
Controls, 112
Cycles, 124
Data types Modicon Modbus, 80
Data types Modicon Modbus TCP/IP, 81
Data types Omron Hostlink/Multilink, 81
Data types OPC, 82
Data types SIMATIC 500/505 DP, 82
Data types SIMATIC 500/505 serial, 83
Data types SIMATIC HMI HTTP Protocol, 83
Data types SIMATIC S5 AS511, 84
Data types SIMATIC S5 DP, 84
Data types SIMATIC S7 200, 85
Data types SIMATIC S7 300/400, 85
Data types Telemecanique Uni-Telway, 88
Displaying a log file, 18
Displaying history, 17
Dynamic, 109
Faceplate, 104
Font settings, 55, 106
Functions, 116
GE Fanuc SNP data types, 78
Global library, 67
Header file, 118
HMI device replacement, 52
Including the hardware configuration, 10
Integrated project, 133
Integrated projects, 131
Introduction, 41
Introduction to migration, 9
Languages, 98
LG GLOFA GM data types, 78
Line report, 123
Log, 120
Migrating an integrated project, 72, 129, 133
Migrating ET200S, 143
Migrating PLC programs to S7-1500, 143, 146, 148, 149, 150, 151, 152, 153, 154, 155, 156, 172, 175, 176, 177, 185, 186, 187, 188, 190, 192, 193, 194, 195, 196, 198, 205, 206, 208
Migrating projects, 45, 73, 92
Mitsubishi FX data types, 79
Mitsubishi Protocol 4 data types, 79
Multilingual reports, 120
Object group, 107
of Basic Process Control, 125
of external tags, 75, 126
of languages, 65
of log data, 68
of project text, 65
of standard functions, 119
OLE object, 105
Position data, 106
Print job, 120
Project library, 67

Recipe data, 68
Reconfigure connection, 57, 101
Runtime data, 68, 126
Script, 64
Script for dynamization, 110
Size data, 106
Supported HMI devices, 47
Supported object types, 96
Supported products, 9
Tags, 101
Text, 65
The migration process, 9
User administration, 68, 123, 126
User log, 115, 126
VB-script, 64
WinCC Controls, 112
WinCC Push Button Control, 105
Migration of STEP 7 projects
Alarms, 40
CBA for IO controllers, 25
Device names of PROFINET IO devices, 24
GSD and GSDML devices, 26
I slaves and I-devices, 25
Introduction, 19
Isochronous mode with PROFINET IO, 25
Media redundancy, 24
Migratable types of connection, 26
Migration of project texts, 40
Multiproject, 27
Names of modules and racks, 24
PROFIBUS isochronous mode, 26
Requirements, 20
Requirements for the project, 21
Scope of migration, 19
Shared devices, 25
System error messages, 40
Transfer areas, 25
Update groups PROFINET IO, 25
Migration tool, 43, 91
Creating a migration file, 14
Including the hardware configuration, 12
Using the migration tool, 12
Migration: Check migration readiness of the hardware, 11
Mitsubishi FX
Migrating data types, 79
Mitsubishi Protocol 4
Migrating data types, 79
Modicon Modbus
Migrating data types, 80
Modicon Modbus TCP/IP
Migrating data types, 81
Module replacement, 136

N

Names of alarm classes
Change through migration, 62

O

Object group
Migrate, 107
Object names
Renaming upon migration, 42
Uniqueness, 42
OLE object
Migration, 105
Omron Hostlink/Multilink
Migrating data types, 81
OPC
Migrating data types, 82

P

Position data
Migrate, 106
Preparing migration, 23
Print job
Migrate, 120
Project
Migrate, 45, 92
Project library
Migrating, 67
Project name
Changing by the migration, 94
Project texts
Migrating, 65

R

Recipe data
Migration, 68, 126
Runtime data
Migration, 68, 126

S

SCL
Migration, 34
Script
Migrating, 64

Script for dynamization
Migrate, 110
SIMATIC 500/505 DP
Migrating data types, 82
SIMATIC 500/505 serial
Migrating data types, 83
SIMATIC HMI HTTP Protocol
Migrating data types, 83
SIMATIC S5 AS511
Migrating data types, 84
SIMATIC S5 DP
Migrating data types, 84
SIMATIC S7 200
Migrating data types, 85
SIMATIC S7 300/400
Migrating data types, 85
Size data
Migrate, 106
Start
Migrated project, 94
Starting the migration tool, 13
STL
Migration, 32

T

Tag log
Migrating, 68
Tags
Migrate, 101
Migration, 75, 126
Telemecanique Uni-Telway
Migrating data types, 88
Texts
Migrating, 65

U

Uniqueness
of object names, 42
User administration
Migrate, 123
Migration, 68, 126
User log
Migrate, 115
User logs
Migrate, 115

V

VB-script
Migrating, 64

W

WinCC flexible
Migrating an integrated project, 73
WinCC flexible project
Migrate, 45
WinCC Push Button Control
Migration, 105
WinCC V7 project
Migrate, 92

SIEMENS

Getting started 1

User interface and operation 2

Help on the information system 3

Find and replace 4

Undoing and redoing actions 5

Introduction to the TIA Portal

Programming and Operating Manual

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Getting started....7

1.1 Getting Started Documentation....7

2 User interface and operation....9

2.1 Starting, setting up, and exiting the TIA Portal....9
2.1.1 Starting and exiting the TIA Portal....9
2.1.2 Overview of the program settings....10
2.1.3 Overview of the settings for software updates....12
2.1.4 Overview of the script and text editor settings....13
2.1.5 Overview of the print settings....14
2.1.6 Overview of library settings....15
2.1.7 Overview of settings for searching in a project....15
2.1.8 Overview of the settings for cross-references....15
2.1.9 Overview of the settings for managing window layouts....16
2.1.10 Overview of settings for the basic integrity check....16
2.1.11 Overview of the online and diagnostic settings....16
2.1.12 Changing the settings....17
2.1.13 Exporting and importing settings....18
2.1.13.1 Exporting and importing settings....18
2.1.13.2 Exporting settings....18
2.1.13.3 Importing settings automatically....19
2.1.13.4 Importing settings manually....20

2.2 Layout of the user interface....21
2.2.1 Views....21
2.2.2 Portal view....21
2.2.3 Project view....23
2.2.4 Library view....26
2.2.5 Project tree....26
2.2.5.1 Function and structure of the project tree....26
2.2.5.2 Showing and hiding columns....30
2.2.6 Work area....30
2.2.6.1 Basics of the work area....30
2.2.6.2 Maximizing and minimizing the work area....32
2.2.6.3 Splitting the work area....33
2.2.6.4 Floating the work area elements....34
2.2.6.5 Using grouped elements of the work area....35
2.2.6.6 Minimizing and maximizing elements of the work area....37
2.2.6.7 Switching between the elements in the work area....37
2.2.7 Inspector window....38
2.2.8 Task cards....40
2.2.8.1 Basics on task cards....40
2.2.8.2 Changing the pane mode....42
2.2.9 Reference projects....42
2.2.10 Details view....43
2.2.11 Overview window....45

2.2.11.1 Overview window....45
2.2.11.2 Comparing objects in the overview window....47
2.2.11.3 Sorting the details view of the overview window....48
2.2.12 User interface layout....49
2.2.12.1 Save user interface layout....49
2.2.12.2 Save window layout....49
2.2.12.3 Load window layout....50
2.2.12.4 Managing window layouts....51
2.2.12.5 Saving a layout of editors and tables....51
2.2.12.6 Resetting the user interface layout....52

2.3 Running external applications....53
2.3.1 Basics for running external applications....53
2.3.2 Configuring external applications....53
2.3.3 Start the external application....55

2.4 Keyboard operation in the TIA Portal....57
2.4.1 Operation of the TIA Portal with the keyboard....57
2.4.2 Displaying an overview of all keyboard shortcuts....57
2.4.3 Basic functions of the TIA Portal....57
2.4.4 Using project-related functions....59
2.4.5 Arranging windows....59
2.4.6 Navigating through the program interface....60
2.4.7 Customizing editors....62
2.4.8 Editing objects....63
2.4.9 Text editing....64
2.4.10 Editing tables....65
2.4.11 Using online functions....67
2.4.12 Using the on-screen keyboard....67
2.4.13 Using user-defined keyboard shortcuts....68
2.4.13.1 Assigning user-defined keyboard shortcuts....68
2.4.13.2 Deleting user-defined keyboard shortcuts....68
2.4.13.3 Resetting user-defined keyboard shortcuts to default setting....69
2.4.13.4 Importing and exporting user-defined keyboard shortcuts....69
2.4.13.5 Selecting a list of keyboard shortcuts....70

2.5 Special features specific to the operating system....72
2.5.1 Influence of user rights....72
2.5.2 Expanding user rights....72

3 Help on the information system....75

3.1 General remarks on the information system....75

3.2 Opening the information system....82

3.3 Using the information system....83

3.4 Searching the information system....85
3.4.1 Basics of searching....85
3.4.2 Searching help topics....86
3.4.3 Examples of searching in the information system....87

3.5 Using favorites....90

3.6 Copying code examples into the program....92

3.7 Displaying more information in the Siemens Industry Online Support....93

3.8 Viewing hardware manuals....94
3.9 Printing help topics....96
3.10 Configuring the display of tooltips and tooltip cascades....97
3.11 Assembling customized documentation....98
3.12 Safety Guidelines....99
4 Find and replace....101
4.1 Basics of searching....101
4.2 Search entire project....102
4.2.1 Basics for search in the project....102
4.2.2 Overview of the search editor....103
4.2.3 Search in entire project....105
4.2.4 Creating a new search index manually....106
4.2.5 Enabling and disabling "Search in project"....107
4.2.6 Examples for searching in the project....108
4.3 Searching and replacing in the editor....111
4.3.1 Basics for search in open editors....111
4.3.2 Overview of the "Find and replace" palette....112
4.3.3 Searching and replacing in the editor....113
5 Undoing and redoing actions....115
5.1 Basics of undoing and redoing actions....115
5.2 Undoing an action....117
5.3 Redoing an action....118
Index....119

Getting started 1

1.1 Getting Started Documentation

Getting Started with the TIA Portal using the TIA Portal Tutorial Center

For a quick start with the TIA Portal, Getting Started documents are available in form of videos in the TIA Portal Tutorial Center. You can either watch individual videos or download the entire media system.

The videos have the following contents:

- They give you an overview of the functions and tools offered by the TIA Portal.
- They give you an overview of the smooth interaction of the configuration and programming tools.
- They show you how to work effectively with the TIA Portal.
- They show you how to implement the various automation tasks in the TIA Portal.

The videos of the TIA Portal Tutorial Center are intended for planners and configuration engineers as well as commissioning and service personnel. The videos are mainly of a technical nature and offer assistance in all phases of the plant lifecycle.

You access the TIA Portal Tutorial Center at the following link:

http://www.siemens.com/tia-portal-tutorial-center (http://www.siemens.com/tia-portal-tutorial-center)

See also

Getting Started Basic 10.5 (http://support.automation.siemens.com/WW/view/en/40263542/0/en)

Service and Support (https://support.industry.siemens.com/cs/de/en/)

Getting Started Professional 11.0 (https://support.industry.siemens.com/cs/ww/en/view/54430386)

Getting Started Professional 12.0 (http://www.automation.siemens.com/salesmaterial-as/interactive-manuals/getting-started_simatic-s7-1500/_content/EN/content_en.html)

User interface and operation 2

2.1 Starting, setting up, and exiting the TIA Portal

2.1.1 Starting and exiting the TIA Portal

Starting the TIA Portal

To start the TIA Portal, follow these steps:

1. In Windows, select "Start > All Programs > Siemens Automation > TIA Portal V14".
   The TIA Portal opens with the last settings used.

Exiting the TIA Portal

To exit the TIA Portal, follow these steps:

1. In the "Project" menu, select the "Exit" command.
   If the project contains any changes that have not been saved, you will be asked if you wish to save them.
   - Select "Yes" to save the changes in the current project and close the TIA Portal.
   - Select "No" to close the TIA Portal without saving the most recent changes in the project.
   - Select "Cancel" to cancel the closing procedure. The TIA Portal will remain open if you select this option.

2.1.2 Overview of the program settings

Overview

The following table shows the application settings that you can make:

Group	Setting	Description
General settings	User name	The user name of the user. The user name is stored in the project properties when a new project is created.
	User interface language	Language for the program interface
	Mnemonic	Specifies the mnemonics for programming: "German" uses the German mnemonics, for example, "E1.0". "International" uses international mnemonics, for example, "I1.0". For information on the differences in the mnemonics of the individual commands, refer to the description of the relevant programming language.
	Show list of recently used projects	Number of entries in the list of recently used projects in the "Project" menu
	Load most recent project during startup	The last opened project is opened automatically after the TIA Portal starts.
	Displaying truncated text in full	Texts which are truncated due to their length are displayed in a tooltip.
	Show tooltips (context-sensitive help is available)	Tooltips are displayed and you get context-sensitive help. If this function is disabled, you can open the tooltip with <F1>. The setting also affects rollouts with context-related syntax errors and input aids in STL.
	Open cascade automatically in tooltips	After a brief time, the tooltips automatically expand to display a cascade containing additional help. If this option is cleared, the tooltips must be manually expanded.
	Show banner minimized	Banners with additional information in the editor are minimized in display by default.
Reset to default	All application settings	All changes that you made in the TIA Portal after installation are undone.
	Layout of the editors	Resets the complete layout of the application to the factory state.
	Show all alarm windows	All alarm windows whose appearance was manually suppressed are displayed again.
Start view	Most recent view	Starts the program in the last view that was used. This can be either the portal view or the project view.
	Portal view	Starts the TIA Portal in the portal view every time, irrespective of the last view you worked in.
	Project view	Starts the TIA Portal in the project view every time, irrespective of the last view you worked in.
Import/export settings	Export settings to file	Opens a dialog in which you can export the selected program settings to a file.
	Import settings from file	Enables the import of previously exported program settings from a file.
Central settings	Storage location for central settings	Here you define the storage location for the file in which the settings are stored centrally.

Group	Setting	Description
View for objects in overview	Details	When several views are available, the detail view opens by default, for example, in the overview window.
	List	When several views are available, the list view opens by default, for example, in the overview window.
	Thumbnails	When several views are available, the symbol view opens by default, for example, in the overview window.
Storage settings	Recently used storage location	When a project is saved the first time, the most recently used file path is set by default.
	Default storage location	Enables the specification of file paths for: • Projects • Libraries • Configuration file for corporate libraries
	Storage location for new projects	Enables the specification of the storage path for projects. New projects are created automatically in this path.
	Storage location for libraries	Enables the specification of the storage path for global libraries. When you create a new library, it is automatically saved to this path.
	Storage location for the configuration file for corporate libraries	Shows the path for the configuration file for the company libraries. The path cannot be changed.
Storage settings for archiving	Last storage settings used for archiving	During saving the last path that you used is preset.
	Specify default settings for storage locations for archiving	Enables the specification of file paths for: • Project archives • Retrieved projects • Library archives • Retrieved libraries
	Storage location for project archives	Enables the specification of the storage path for project archives. Project archives are created automatically in this path.
	Storage location for retrieved projects	Enables the specification of the storage path for retrieved projects. Retrieved projects are created automatically in this path.
	Storage location for library archives	Enables the specification of the storage path for library archives. Library archives are created automatically in this path.
	Storage location for retrieved libraries	Enables the specification of the storage path for retrieved libraries. Retrieved libraries are created automatically in this path.
Data exchange	Storage location for data import	Imported files are searched for at this storage path by default.
	Storage location for data export	This is the default storage path for the data export.
	Storage location for support packages	When support packages are downloaded, they are stored in the specified storage path and can be installed from there.
	Storage location for log files	Log files are stored at the location specified here.

Group	Setting	Description
User documentation	Show user-defined documentation in the selected editing language	The user documentation is opened in the project editing language (Shift + F1). If the user documentation does not exist in the selected editing language, the English version is opened by default.
	Display call log for user-defined documentation	Shows call logs for user documentation. Alarms for calling user-created help files are output in the Inspector window. This makes it easier to identify and correct difficulties in calling user-created help. This function is particularly useful when creating your own help files.
	Search for user-defined documentation in a central directory	Enables the saving of user-defined documentation in a directory outside the current project directory in order, for example, to make documentation available across projects.
	Central directory for user-defined documentation	Enables the specification of the storage path on which user help files for user-defined documentation can be stored.

See also

Starting and exiting the TIA Portal (Page 9)

Resetting the user interface layout (Page 52)

Changing the settings (Page 17)

Configuring the display of tooltips and tooltip cascades (Page 97)

2.1.3 Overview of the settings for software updates

Overview

The following table shows the available settings for software updates:

Group	Setting	Description
TIA Portal update	Check for updates daily	Checks if updates are available after every restart of the computer and then every 24 hours. If updates are found, you receive a message in the information area of the Windows taskbar.
	Check for updates now	Starts the search for updates and displays the result in the TIA Updater.

2.1.4 Overview of the script and text editor settings

Overview

The following table shows the available settings for script and text editors:

Group	Setting	Description
Font	Font type and size	Sets the font type and size for the text in text editors.
Font colors	Color settings	You can choose the colors for individual text elements from the respective drop-down lists in the text editors. Optional settings are available for the following text elements: • Text • Keywords • Comments • Operators • Scripts • Standard functions • Instructions/system functions • Constant strings • Numeric constants • Constant tags • Tags • Object models • Formal parameters
	Reset to default	Resets all font colors in editors to their factory settings.
Tabs	Tab width	Sets the width of tabs.
	Use tabs	Enables the use of tabs.
	Use spaces	Specifies use of space characters instead of tabs.
Indent	None	Text entries are not automatically indented.
	Block	The lines or the selected paragraph are automatically indented.
	Smart	The lines or the selected paragraph are automatically indented. All unnecessary spaces are also removed.
View	Show white spaces	Shows control characters within a text.
STL (Statement List)	Font type and size	Sets the font type and size for STL program code.
SCL (Structured Control Language)	Show line numbers	Shows the row numbers in SCL programs.

See also

Changing the settings (Page 17)

2.1.5 Overview of the print settings

Overview

The following table shows the available settings for printing:

Group	Setting	Description
General	Always print table data as pairs of values	Tables are printed as a list and not in tabular form. The corresponding values are listed for each column. Enable this option, for example, if you want to print a table that is too large for the print area.
	Timeout for print job (minutes)	If elements cannot be completely printed, a message is displayed after the configured timeout in minutes.
Hardware configuration	Active graphic view	The graphics of network and device view are included in the printout.
	Active table	A table associated with an editor is included when printing with the editor.
PLC Programming	Zoom factor	Specifies the size in which blocks are to be printed out.
	With interface	The interfaces of blocks are included in the printout.
	With comments	Comments on blocks are included in the printout.
	With line numbers	The line numbers of the program code are printed for text-based programming languages.
Motion & Technology	Dialog display/graphic	The content of the editor is printed out as a graphic if the editor supports this.
	Table	The parameters of the technology objects are printed out in the form of a table.
HMI screens	Show tab order	In the printout you can specify the order in which the runtime objects can be selected with the TAB key.

See also

Changing the settings (Page 17)

2.1.6 Overview of library settings

Overview

The following table shows the library settings that you can make:

Group	Setting	Description
Type version	Disable editing of the type version number	If this setting is activated, the type version number cannot be changed by the user during the "Release version" and "Assign version" actions.
Type connection	Disable the shortcut menu command "Terminate connection to type"	When the setting is enabled, the instance of a type in the project tree can no longer be separated from the type in a library with the help of the shortcut menu. The shortcut menu command "Terminate connection to type" is disabled.
Updating the project	Do not delete unused type instances when you update the project from the library	If the setting is activated, unused type instances are not deleted when the project is updated. If an unused type does not have a newer type version instance, then it should not be deleted. If an unused type has a newer type version instance, then it will be deleted only if it remains unused after selected update project.

2.1.7 Overview of settings for searching in a project

Overview

The following table shows the available settings for searching in a project:

Group	Setting	Description
Search in project	Activate "Search in project"	Enables or disables the search in project.
	Recreate search index	Regenerates the search index for the search in project.

2.1.8 Overview of the settings for cross-references

Overview

The following table shows the available settings for cross-references:

Group	Setting	Description
Recreate the cross-reference information	Recreate the cross-reference information	Creates new cross-reference information about the selected project.

2.1.9 Overview of the settings for managing window layouts

Overview

The following table shows the available settings for managing window layouts:

Group	Setting	Description
Manage all window layouts	Active	Enables the selection of a window layout when additional window layouts are created in addition to the default window layout.

2.1.10 Overview of settings for the basic integrity check

Overview

The following table shows the settings that you can make for the basic integrity check:

Group	Setting	Description
Basic integrity check	Activate basic integrity check when opening a project	By default activates the option to run the basic integrity check in the "Open project" dialog. In addition, the basic integrity check is performed automatically when opening projects from the list of recently opened projects, the reference projects as well as in multi-user engineering. Depending on the scope and application, the check may be quite time-consuming especially when using multi-user engineering.
	Activate basic integrity check when opening a library	Activates the automatic basic integrity check when opening libraries. Depending on the scope and application, the check may be quite time-consuming.

See also

Changing the settings (Page 17)

2.1.11 Overview of the online and diagnostic settings

Overview

The following table shows the settings that you can make for the online and diagnostic functions:

Group	Setting	Description
Preset connection path for online access	Type of the PG/PC interface	Specifies the type of the PG/PC interface that is used as the presetting in the dialogs for online access, e.g., in the "Go online" dialog.
	PG/PC interface	Specifies a particular PG/PC interface that is used as the presetting in the dialogs for online access, e.g., in the "Go online" dialog.

Group	Setting	Description
	Use preset connection path for online connection	Activates or deactivates the presettings for the PG/PC interface. If the check box is selected, the connection path specified in the settings is used as the presetting in the dialogs for online access.
Alarm display	Multiline	Displays the alarms in the Inspector window using multiple lines.
	Autoscroll	When a new alarm occurs, the display scrolls automatically to this alarm.
	Archive size	Maximum number of alarms that can be displayed in the Inspector window. If the set number of alarms is exceeded, older alarms are automatically deleted.

2.1.12 Changing the settings

Procedure

To change the settings, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation to change the settings described in the previous sections. Or click on one of the other entries in area navigation to make settings for your installed products.
3. Change the settings.

Result

The change will be adopted immediately, there is no need to save it explicitly.

See also

Overview of the program settings (Page 10)

Overview of the script and text editor settings (Page 13)

Overview of the print settings (Page 14)

2.1.13 Exporting and importing settings

2.1.13.1 Exporting and importing settings

You can export settings of the TIA Portal to a file. When you import the settings, they are applied together. This function makes it possible to send settings to other users. Likewise, system settings can be specified centrally, to provide a unified environment for multiple engineering PCs. The TIA Portal searches for a initialization file at startup. When a initialization file is found, the settings saved therein are automatically applied after rebooting. If a more recent version of this file is available, the local settings are updated.

You can export all settings, for example:

- Language of the user interface
- Path of the company libraries
- Path of the user-defined documentation
- Window layouts
- Profiles of the hardware catalog

The initialization files are version-specific and can be imported automatically within a product version, but not in other versions of the product.

2.1.13.2 Exporting settings

You can export settings of the TIA Portal to a file, for example, to back them up or to share them with other users.

Procedure

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Select the "General" entry in the area navigation.
3. In the "Import/export settings" area, click on the "Export settings ..." button.
   The "Export settings" dialog opens.
4. Select the settings for the export.
   - Click "Select all" to select all the settings for the export.
   - Click "Deselect all" to then select individual settings for the export. It all the nested settings are selected as well.
5. If you want to accept the settings selected from previous export operations, for example for a different use of the initialization files, select the check box "Use selection from a previous export file" and enter the file in the box.

6. Enter a file name for the initialization file in the "File name" box.
7. Click "Export".
   Settings are exported to a file with the extension ".tps15_1".

Note

Empty entry fields in the export file

Entry fields, such as "Central settings", are available in the settings. If you do not fill out one of these entry fields and nevertheless select the initialization for the export, an empty initialization will be exported. If the subsequently import the initialization file, the corresponding local settings are overwritten by the empty contents of important initialization file.

If you do not want these fields to be deleted by an import, disable the non-filled entry fields during export of the initialization file.

2.1.13.3 Importing settings automatically

When the TIA Portal starts up, you can automatically import the settings from a initialization file, for example, to incorporate a centrally defined environment.

Procedure

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Select the "General" entry in the area navigation.
3. In the "Central settings" area, click on the "Browse" button.
4. Select the setting file with the extension ".tps15_1".
5. Click "Open".
   The settings are applied after a restart of the TIA Portal.

Note

Priority of central settings

Central settings have priority over manually imported settings and override them.

Note

Display of memory path for central settings

When you have entered a memory path for a setting file in the settings in the "Central settings" area, this path will no longer be displayed after a restart of the TIA Portal. The settings are nevertheless loaded after a restart.

To configure the central settings for the system, change the path for the storage location of the setting file first and then export the settings to the setting file.

2.1.13.4 Importing settings manually

You can import settings of the TIA Portal manually from a initialization file, for example, to take over settings from other users.

Procedure

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Select the "General" entry in the area navigation.
3. In the "Import/export settings" area, click on the "Import settings ..." button.
4. Select the setting file with the extension ".tps15_1".
5. Click "Open".
   The settings are applied after a restart of the TIA Portal.

2.2 Layout of the user interface

2.2.1 Views

Views

Three different views are available for your automation project:

- The portal view is a task-oriented view of the project tasks.
- The project view is a view of the components of the project, as well as the relevant work areas and editors.
- The library view (Page 26) shows the elements of the project library and the open global libraries.

You can change over between the two views using a link.

2.2.2 Portal view

Purpose of the portal view

The portal view provides you with a task-oriented view of the tools. Here, you can quickly decide what you want to do and call up the tool for the task in hand. If necessary, the view changes automatically to the project view (Page 23) for the selected task.

Layout of the portal view

The following figure shows an example of the components in the portal view:

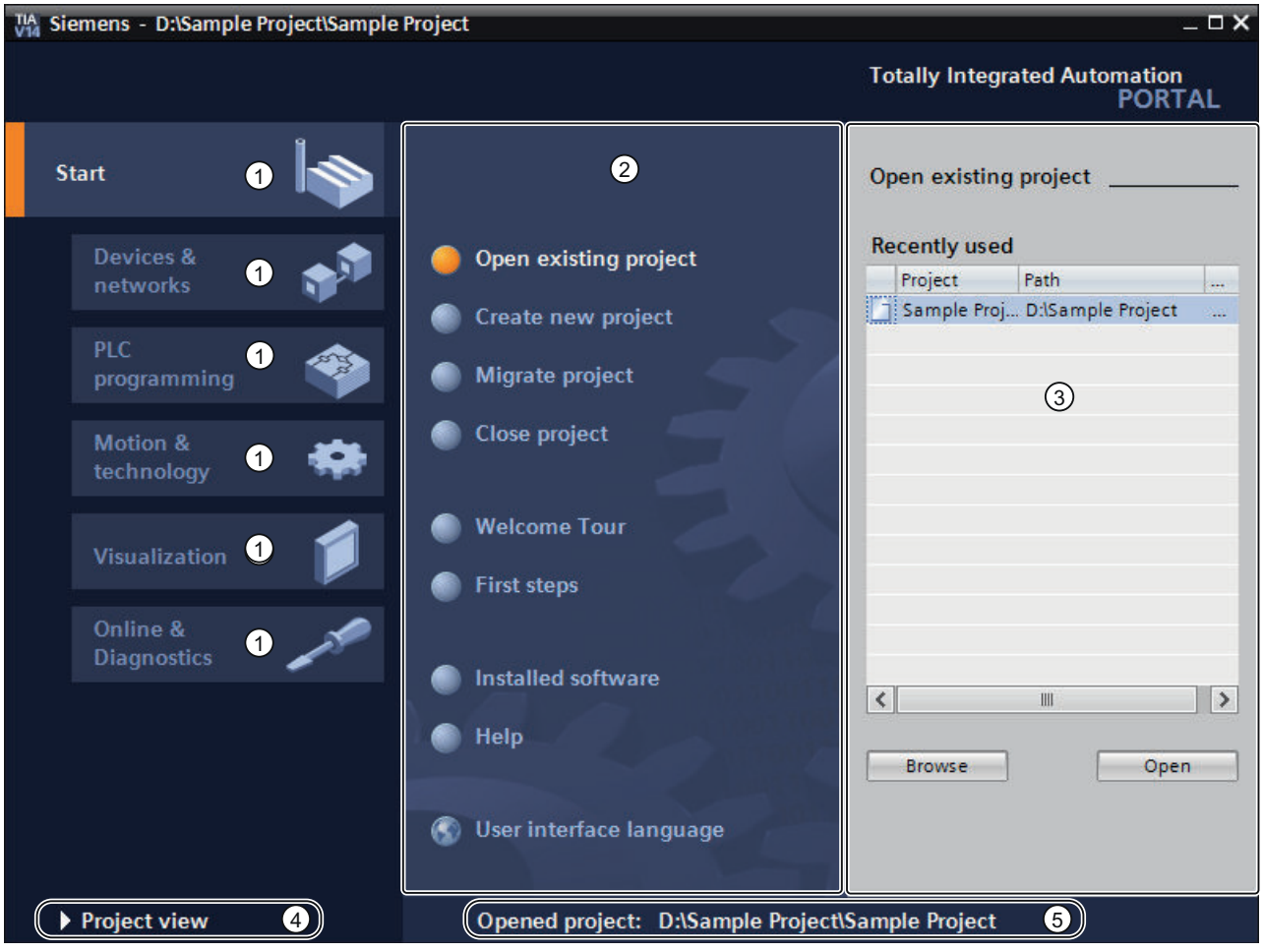

① Portals for different tasks
② Actions for the selected portal
③ Selection panel for the selected action
④ Switch to project view
⑤ Display of the project that is currently open

Portals

The portals provide the basic functions for the individual task areas. The portals that are provided in the portal view depends on the products that have been installed.

Actions for the selected portal

Here, you will find the actions available to you in the portal you have selected. You can call up the help function in every portal on a context-sensitive basis.

Selection panel for the selected action

The selection panel is available in all portals. The content of the panel adapts to your current selection.

Switch to project view

You can use the "Project view" link to change to the project view.

Display of the project that is currently open

Here, you can obtain information about which project is currently open.

See also

Function and structure of the project tree (Page 26)

Basics of the work area (Page 30)

Inspector window (Page 38)

Basics on task cards (Page 40)

Details view (Page 43)

2.2.3 Project view

Purpose of the project view

The project view is a structured view of all components of the project.

Layout of the project view

The following figure shows an example of the components of the project view:

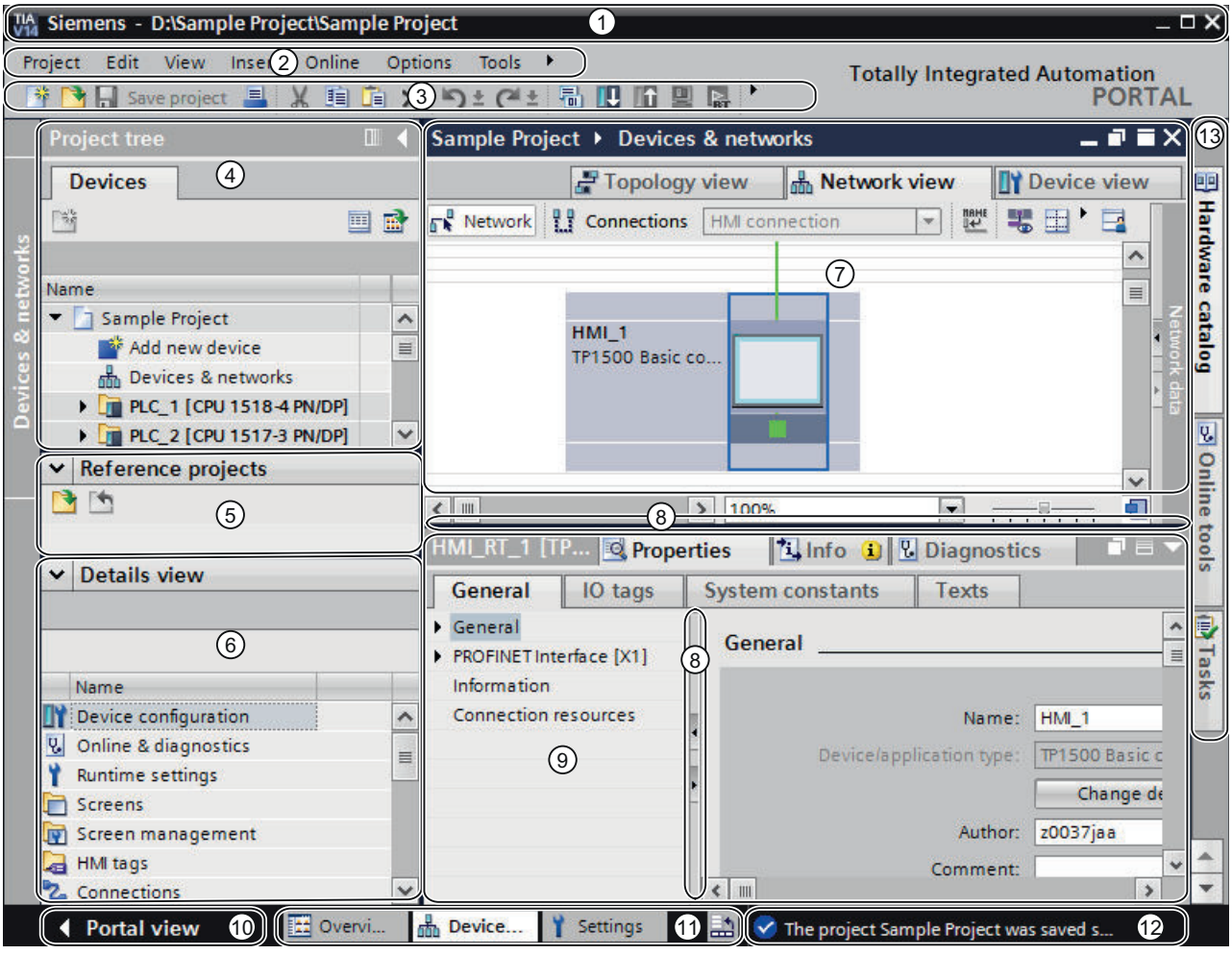

①	Title bar
②	Menu bar
③	Toolbar
④	Project tree (Page 26)
⑤	Reference projects (Page 42)
⑥	Details view (Page 43)
⑦	Work area (Page 40)
⑧	Dividers
⑨	Inspector window (Page 38)
⑩	Changing to the portal view (Page 21)
⑪	Editor bar
⑫	Status bar with progress display
⑬	Task cards (Page 40)

Title bar

The name of the project is displayed in the title bar.

Menu bar

The menu bar contains all the commands that you require for your work.

Toolbar

The toolbar provides you with buttons for commands you will use frequently. This gives you faster access to these commands.

Dividers

Dividers separate individual components of the program interface. The arrows on the dividers allow you to display and hide the adjacent sections of the user interface.

Changing to the portal view

You can use the "Portal view" link to change to the portal view.

Editor bar

The Editor bar displays the open editors. This allows you to quickly switch between open elements. If you have opened numerous editors, you can display the editors of the same type as a group.

Status bar with progress display

In the status bar, you will find the progress display for processes that are currently running in the background. This also includes a progress bar that shows the progress graphically. Hover the mouse pointer over the progress bar to display a tooltip providing additional information on the active background process. You can cancel the background processes by clicking the button next to the progress bar.

If no background processes are currently running, the status bar displays the last generated alarm.

See also

Basics of the work area (Page 30)

2.2.4 Library view

Function of the library view

The library view provides an overview of the elements in the project library and the open global libraries. You can switch to the library view using the "Libraries" task card.

See also: Overview of the library view

2.2.5 Project tree

2.2.5.1 Function and structure of the project tree

Function of the project tree

Using the project tree features gives you access to all components and project data. You can perform the following tasks in the project tree:

- Add new components
- Edit existing components
- Scan and modify the properties of existing components

You can select the objects of the project tree either with the mouse or via the keyboard by typing the first letter of the desired object. If more than one object begins with the same letter, the next lower object is selected. The project tree must be the focused user interface element in order for you to select an object with its initial letter.

Layout of project tree

The following figure shows an example of the project tree areas:

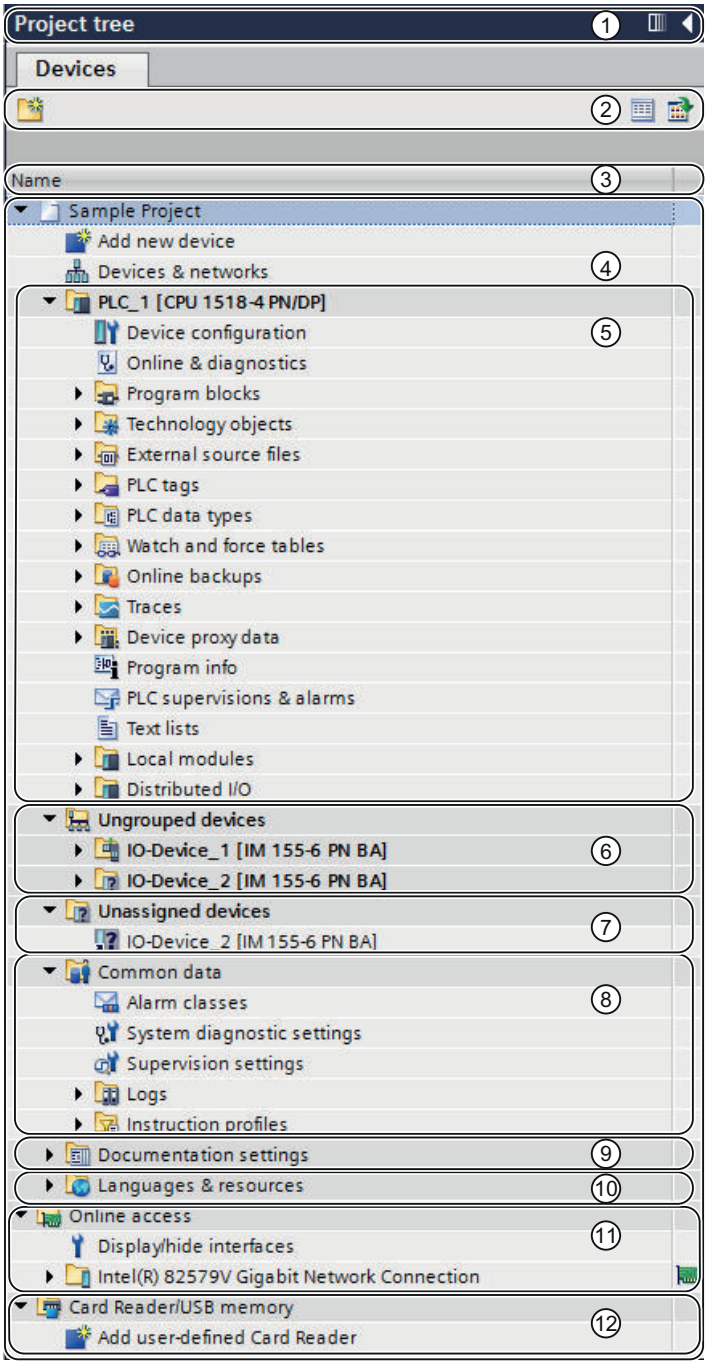

① Title bar
② Toolbar
③ Table header
④ Project

⑤ Devices
⑥ Ungrouped devices
⑦ Unassigned devices
⑧ Common data
⑨ Documentation settings
⑩ Languages & resources
⑪ Online access
⑫ Card Reader / USB memory

Title bar

The title bar of the project tree has a button for automatically and manually collapsing the project tree. Once it is collapsed manually, the button is "Reduced" to the left-hand margin. It changes from an arrow pointing left to one that is pointing right, and can now be used to reopen the project tree. You can use the "Collapse automatically" button collapse to project tree automatically when you do not need it.

See also: Maximizing and minimizing the work area (Page 32)

Toolbar

You can do the following tasks in the toolbar of the project tree:

- Create a new user folder; for example, in order to group blocks in the "Program blocks" folder or group devices.
- Navigate forward to the source of a link and back to the link itself.
  There are two buttons for links in the project tree. You can use these to navigate from the link to the source and back.
- Show an overview of the selected object in the work area.
  When the overview is displayed, the lower-level objects and actions of the elements in the project tree are hidden.

Table header

The "Name" column is shown by default. You can also show the columns "Type name" and "Version". If you are showing the additional columns, you see the name of the respective type as well as the version used for instances of types from the library.

Project

You will find all the objects and actions related to the project in the "Project" folder, e.g.:

- Devices
- Languages & resources
- Online access

Device

There is a separate folder for each device in the project, which has an internal project name. Objects and actions belonging to the device are arranged inside this folder.

Ungrouped devices

All distributed I/O devices in the project are collected together in the "Ungrouped devices" folder.

Unassigned devices

Distributed I/O devices that are not assigned to a distributed I/O system appears as links in the "Unassigned devices" folder.

Common data

This folder contains data that you can use across more than one device, such as common alarm classes, logs and scripts.

Documentation settings

In this folder, you can specify the layout for the printed project documentation.

Languages & resources

You can determine the project languages and texts in this folder.

Online access

This folder contains all the interfaces of the programming device / PC, even if they are not used for communication with a module.

Card Reader / USB memory

This folder is used to manage all card readers an other USB storage media connected to the programming device / PC.

See also

Portal view (Page 21)

Project view (Page 23)

Basics of the work area (Page 30)

Inspector window (Page 38)

Basics on task cards (Page 40)

Details view (Page 43)

Showing and hiding columns (Page 30)

2.2.5.2 Showing and hiding columns

You can show additional columns in the project tree, if necessary. These additional columns show the name of the type associated with an instance as well as its version number.

Procedure

To show or hide additional table columns for the associated type and its version number, follow these steps:

1. Click the "Display/hide column heading" in the toolbar of the project tree.
   The column view of the project tree is enabled and a table header is displayed.
2. Right-click on the table header of the project tree.
3. Select the "Show/Hide" command in the shortcut menu, and select the columns you want to display.
   The selected columns are displayed or hidden.

See also

Function and structure of the project tree (Page 26)

2.2.6 Work area

2.2.6.1 Basics of the work area

Function of the work area

The objects that you can open for editing purposes are displayed in the work area. These objects include, for example:

- Editors and views
- Tables

You can open several objects. However, normally it is only possible to see one of these at a time in the work area. All other objects are displayed as tabs in the Editor bar. If, you would like to view two objects at the same time when performing certain tasks, you can tile the work area either horizontally or vertically or undock elements of the work area. If no objects are open, the work area will be empty.

Layout of the work area

The following figure shows an example of a vertically split work area:

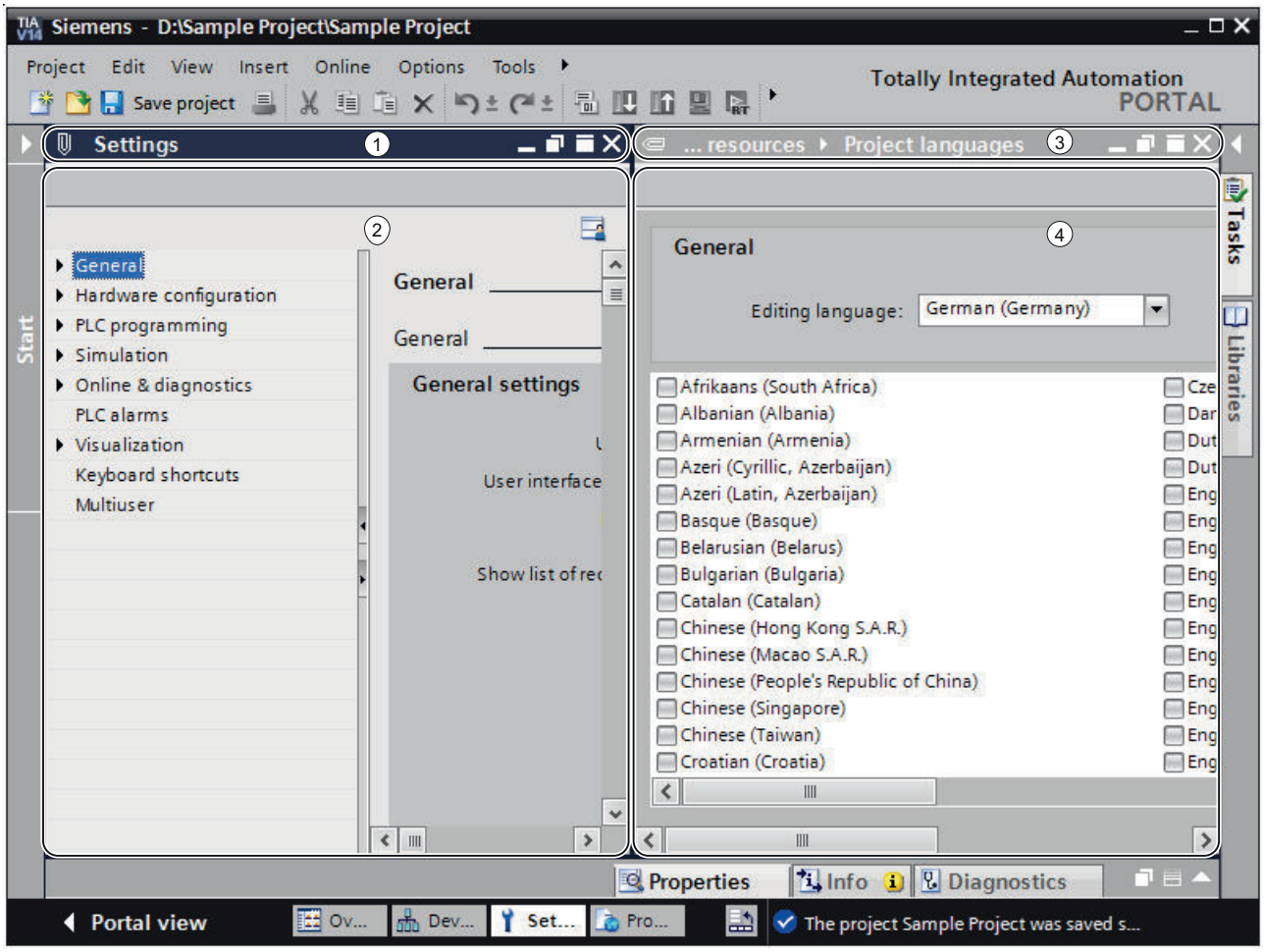

① Title bar of left-hand editor
② Work area of left-hand editor
③ Title bar of right-hand editor
④ Work area of right-hand editor

See also

Maximizing and minimizing the work area (Page 32)

Splitting the work area (Page 33)

Floating the work area elements (Page 34)

Using grouped elements of the work area (Page 35)

Minimizing and maximizing elements of the work area (Page 37)

Switching between the elements in the work area (Page 37)

Saving a layout of editors and tables (Page 51)

Save user interface layout (Page 49)

2.2.6.2 Maximizing and minimizing the work area

You have the option to adapt the work area to make it as large as possible. You can use the following function for this:

- Maximizing the work area
  You can close the task cards, project tree and inspector window with a single click. This increases the size of the work area. You can minimize the work area again at any time in order to return to the previous view.
- Collapsing task cards, project tree, and Inspector window automatically
  You can use the "Collapse automatically" option for the task cards, project tree, and Inspector window. This function causes these items to collapse automatically when you don't need them.

Maximizing and minimizing the work area

To maximize the work area, follow these steps:

1. Open an element such as an editor or a table.
   The element appears in the work area.
2. Click the "Maximize" button in the title bar of the element.
   The task cards, project tree and inspector window collapse, and the work area is shown with its maximum dimensions.

To minimize the work area again, follow these steps:

1. Click the "Embed" button in the title bar of the displayed element.
   This restores the view that existed before the work area was maximized. That is, if the task cards, project tree, or Inspector window were expanded before, they will be expanded again.

Collapsing task cards, project tree, and Inspector window automatically

To collapse the task cards automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the task cards.
   The task cards collapse when you click anywhere outside the task cards.
2. To use the task cards, click the collapsed task cards.
3. The task cards expand and are available for use. The "Collapse automatically" option remains enabled.

To collapse the project tree automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the project tree.
   The project tree collapses when you click anywhere outside the project tree.
2. To use the project tree, click the collapsed project tree.
   The project tree expands and is available for use. The "Collapse automatically" option remains enabled.

To collapse the Inspector window automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the Inspector window.
   The Inspector window collapses when you click anywhere outside the Inspector window.
2. To use the Inspector window, click the collapsed Inspector window.
   The Inspector window expands and is available for use. The "Collapse automatically" option remains enabled.

To disable the automatic collapse option, follow these steps:

1. Click "Expand permanently" again in the relevant window.
   The Collapse automatically" option is disabled, and the window remains expanded.

See also

Basics of the work area (Page 30)

Splitting the work area (Page 33)

Floating the work area elements (Page 34)

Using grouped elements of the work area (Page 35)

Minimizing and maximizing elements of the work area (Page 37)

Switching between the elements in the work area (Page 37)

Saving a layout of editors and tables (Page 51)

2.2.6.3 Splitting the work area

You can split the work area vertically or horizontally.

Procedure

To split the work area vertically or horizontally, follow these steps:

1. In the "Window" menu, select the "Split editor space vertically" or "Split editor space horizontally" command.
   The element you have clicked and the next element in the Editor bar will be displayed either next to one another or one above the other.

Note

If no elements are open in the work area, the "Split editor space vertically" and "Split editor space horizontally" functions will not be available.

See also

Basics of the work area (Page 30)

Maximizing and minimizing the work area (Page 32)

Floating the work area elements (Page 34)

Using grouped elements of the work area (Page 35)

Minimizing and maximizing elements of the work area (Page 37)

Switching between the elements in the work area (Page 37)

Saving a layout of editors and tables (Page 51)

2.2.6.4 Floating the work area elements

You can float work area elements in their own separate window:

- Editors
- Tables
- Setting windows
- Task cards
- Inspector window

You can embed floating elements again in the work area at any time.

Floating the work area elements

To float work area elements, follow these steps:

1. Click the "Float" button in the title bar of the element.
   The element will be released from the work area and displayed in its own window. You can now place the window wherever you wish. If you have minimized the window, you can restore it via the editor bar.

Embedding elements in the work area

To embed elements in the work area again, follow these steps:

1. Click the "Embed" button in the title bar of the element.
   The element will appear in the work area again.

See also

Basics of the work area (Page 30)
Maximizing and minimizing the work area (Page 32)
Splitting the work area (Page 33)
Using grouped elements of the work area (Page 35)
Minimizing and maximizing elements of the work area (Page 37)
Switching between the elements in the work area (Page 37)
Saving a layout of editors and tables (Page 51)

2.2.6.5 Using grouped elements of the work area

If you have opened several similar elements as editors or spreadsheets, you can have them displayed in the editor bar as a group. You can use these groups as follows:

- Displaying individual elements of a group
- Displaying all elements of a group in separate windows
- Embedding all displayed elements of a group in the work area
- Minimizing all displayed elements
- Closing all elements of a group

You can undo the grouping of similar elements at any time.

Grouping or ungrouping similar elements

To group similar elements in the editor bar or ungroup them again, follow these steps:

1. Click on the "Group editors/Ungroup" button on the right in the editor bar.
   Each time you click the button, similar elements are grouped or existing groups are dissolved. The icon of the button changes slightly here to enable you to quickly identify the mode in which you are working. The configured mode is retained even after the elements are closed.

Displaying individual elements of a group

To display individual elements of a group, follow these steps:

1. In the editor bar, click the group containing the element you want to display.
   All list of all available elements of the group is displayed.
2. Click the element that you want to display.

Displaying all elements of a group in separate windows

To display all elements of a group in separate windows, follow these steps:

1. In the editor bar, right-click the group whose elements you want to display.
2. Select "Restore group" in the shortcut menu.
   All elements of the group are displayed in separate, overlapping windows. Move the windows in order to see the individual element, or choose an element via the group in the editor bar.

Embedding all displayed elements of a group in the work area

To embed all elements of a group displayed in separate windows in the work area again, follow these steps:

1. In the editor bar, right-click the group whose elements you want to embed.
2. Select "Embed group" in the shortcut menu.
   All elements of the group are embedded in the work area again.

Minimizing all displayed elements

To minimize all elements of a group, follow these steps:

1. In the editor bar, right-click the group whose elements you want to minimize.
2. Select "Minimize group" in the shortcut menu.
   All elements of the group are minimized. However, the minimized elements remain open and can be quickly maximized again via the group in the editor bar.

Closing all elements of a group

To close all elements of a group, follow these steps:

1. In the editor bar, right-click the group whose elements you want to close.
2. Select "Close group" in the shortcut menu.
   All elements of the group are closed. The group is removed.

See also

Basics of the work area (Page 30)

Maximizing and minimizing the work area (Page 32)

Splitting the work area (Page 33)

Floating the work area elements (Page 34)

Minimizing and maximizing elements of the work area (Page 37)

Switching between the elements in the work area (Page 37)

Saving a layout of editors and tables (Page 51)

2.2.6.6 Minimizing and maximizing elements of the work area

You can minimize the elements that are open in the work area, such as editors or tables, as needed. However, an element remains open even if it has been minimized, and can quickly be maximized again using the editor bar.

Minimizing elements in the work area

To minimize elements in the work area, follow these steps:

1. Click the "Minimize" button in the title bar of the element.
   The element is minimized, but can still be accessed via the editor bar.

To minimize all elements at the same time, follow these steps:

1. In the "Window" menu, select the "Minimize all" command.

Maximizing elements in the work area

To maximize elements in the work area again, follow these steps:

1. Click the required element in the editor bar.
   The element is maximized and appears in the work area.

See also

Basics of the work area (Page 30)

Maximizing and minimizing the work area (Page 32)

Splitting the work area (Page 33)

Floating the work area elements (Page 34)

Using grouped elements of the work area (Page 35)

Switching between the elements in the work area (Page 37)

Saving a layout of editors and tables (Page 51)

2.2.6.7 Switching between the elements in the work area

You can switch between the elements in the work area at any time.

Switching between the elements in the work area

To switch to the previous or next editor, follow these steps:

1. In the "Window" menu, select the "Next editor" or "Previous editor" command.
   The next or previous editor will be displayed.

See also

Basics of the work area (Page 30)
Maximizing and minimizing the work area (Page 32)
Splitting the work area (Page 33)
Floating the work area elements (Page 34)
Using grouped elements of the work area (Page 35)
Minimizing and maximizing elements of the work area (Page 37)
Saving a layout of editors and tables (Page 51)

2.2.7 Inspector window

Function of the Inspector window

Additional information on an object selected or on actions executed are displayed in the inspector window.

Layout of the Inspector window

The following figures show the components of the Inspector window:

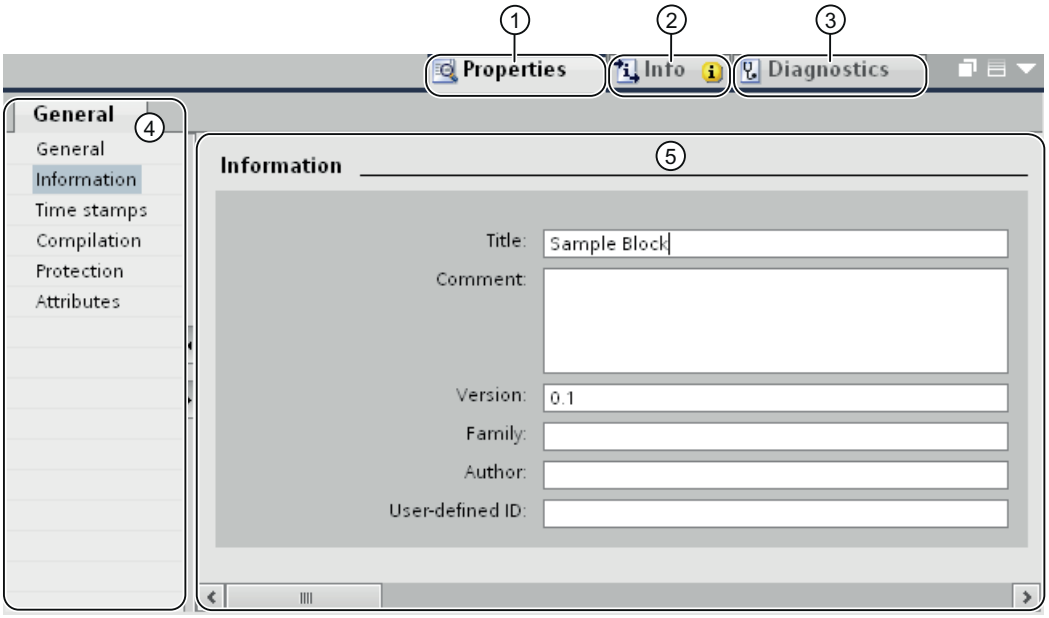

① "Properties" tab
② "Info" tab
③ "Diagnostics" tab
④ Area navigation within the "Properties" tab
⑤ Content of the "Properties" tab

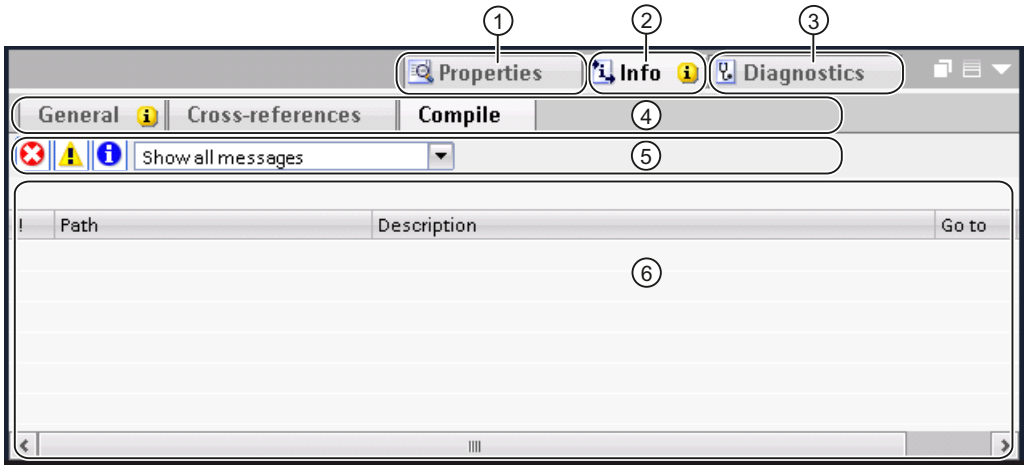

① "Properties" tab
② "Info" tab
③ "Diagnostics" tab
④ Navigation through additional tabs within a tab (only available in the "Info" and "Diagnostics" tabs)
⑤ Toolbar (only available in the secondary "General" and "Compile" tabs of the "Info" tab)
⑥ Content of the "Compile" tab in the "Info" tab

"Properties" tab

This tab displays the properties of the object selected. You can change editable properties here.

"Info" tab

This tab displays additional information on the object selected, as well as alarms on the actions executed (such as compiling).

"Diagnostics" tab

This tab provides information on system diagnostics events, configured alarm events, and connection diagnostics.

Navigation within the tabs

You can use area navigation and the lower-level tabs to display the information you require within the tabs.

Toolbar

You can use the toolbar in the "General" and "Compile" tabs within the "Info" tab to specify which types of alarms are to be displayed. You can enable or disable the display for the following alarm types:

- Errors
- Warnings
- Information

See also

Function and structure of the project tree (Page 26)

Basics of the work area (Page 30)

Portal view (Page 21)

Project view (Page 23)

Basics on task cards (Page 40)

Details view (Page 43)

2.2.8 Task cards

2.2.8.1 Basics on task cards

Function of task cards

Depending on the edited or selected object, task cards that allow you perform additional actions are available. These actions include:

- Selecting objects from a library or from the hardware catalog
- Searching for and replacing objects in the project
- Dragging predefined objects to the work area

The task cards available can be found in a bar on the right-hand side of the screen. You can collapse and reopen them at any time. Which task cards are available depends on the products installed. More complex task cards are divided into panes that you can also collapse and reopen.

Layout of task cards

The following figure shows an example of the bar with the task cards:

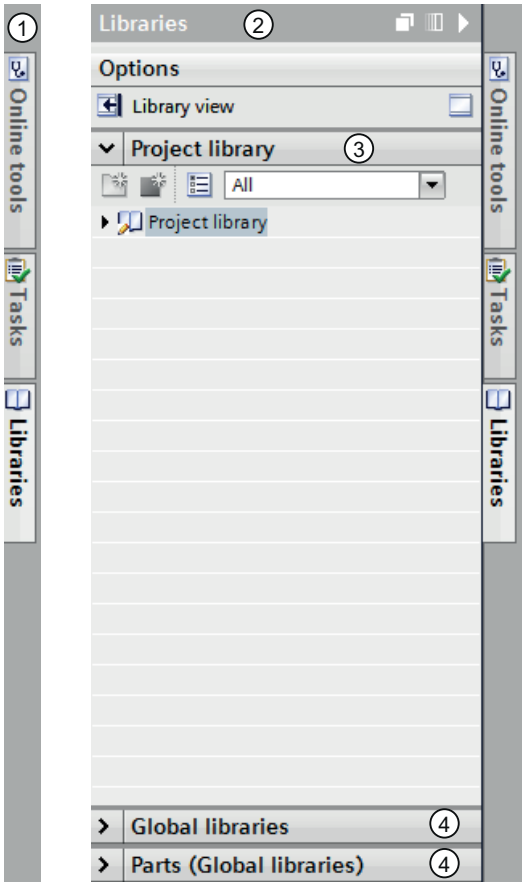

① Task cards closed
② Task card open
③ Opened palette of a task card
④ Closed palette of a task card

See also

Changing the pane mode (Page 42)
Function and structure of the project tree (Page 26)
Basics of the work area (Page 30)
Inspector window (Page 38)
Portal view (Page 21)
Project view (Page 23)
Details view (Page 43)

2.2.8.2 Changing the pane mode

You can choose between two pane modes:

- Single pane mode:
  Only one pane is open at any given time. If you open another pane, the previously opened pane is closed automatically.
- Multi-pane mode:
  You can open several panes at the same time.

Procedure

To change the pane mode, follow these steps:

1. Click the "Change pane mode" button above the panes inside a task card.

See also

Basics on task cards (Page 40)

2.2.9 Reference projects

Function of reference projects

In the "Reference projects" palette, you can open other projects in addition to the current project. These reference projects are write-protected and cannot be edited. However, you can drag the objects of a reference project into your current project and further edit them there. You can also compare the objects of a reference project to the objects of your current project.

Layout of the "Reference projects" palette

The following figure shows the layout of the "Reference projects" palette:

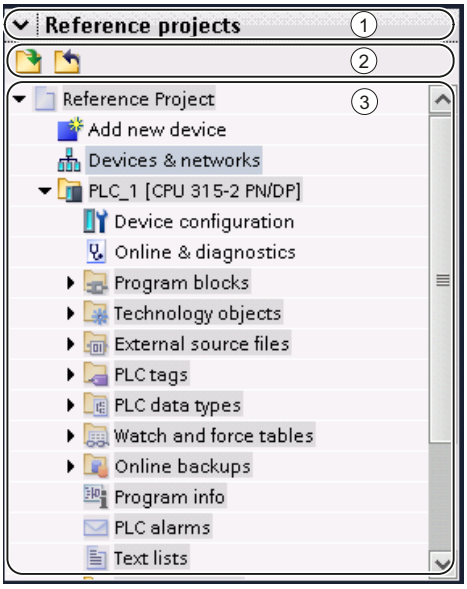

① Title bar
② Toolbar
③ Opened reference projects

Title bar

The arrow for closing the palette is located in the title bar of the "Reference projects" palette. Once it is closed, the direction in which the arrow is pointing changes from downwards to right. It can now be used to reopen the palette.

Toolbar

The toolbar contains buttons for opening and closing reference projects.

Opened reference projects

Opened reference projects are displayed as read-only with their objects and their hierarchical structure.

2.2.10 Details view

Purpose of the details view

The detail view shows certain content of the selected object is in the overview window or in the project tree. This might include text lists or tags.

Layout of the details view

The following figure shows an example of the details view:

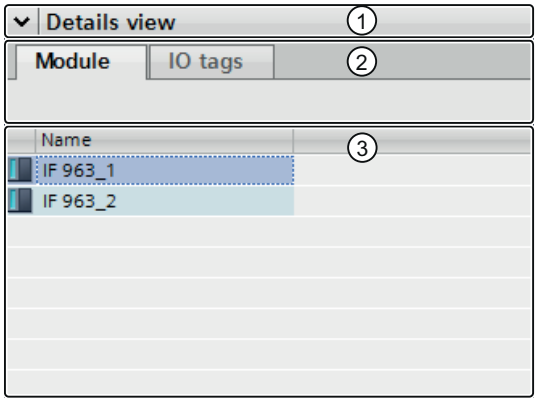

① Title bar
② Tab
③ Content of the selected object

Title bar

The arrow for closing the details view is located in the title bar of the details view. After it has closed, the direction in which the arrow is pointing changes from left to right. It can now be used to reopen the details view.

Tab

The tab displayed in the detail view depends on the selected object.

- The "Modules" tab is displayed when you select a device, a device link or a folder that contains one of these objects.
  The "Modules" tab displays the installed modules or sub-modules of a device in the detail view, when you select the device itself or a device link in the project tree.
- The "IO tags" displays existing IO tags for selected devices and device links to below the "Local modules" folder or a device folder. The name of the simple tag or data element of a PLC data type is displayed in addition to the data type and the address.

  Note

  If a simple tag is defined for a channel address but it is also within the address range of the data element of a PLC data type, only the name of the simple tag is displayed.

- The "System constants" tag is shown if you select a device link below a device folder in the "Distributed I/O" folder or the "Unassigned devices" folder.
  The "System constants" tab displays the global constants used by the device with HW identifiers, for example, of the modules, submodules and interfaces.

Objects

The displayed content varies depending on the selected object. You can move the content of objects from the details view to the required location using drag-and-drop.

See also

Function and structure of the project tree (Page 26)

Basics of the work area (Page 30)

Inspector window (Page 38)

Basics on task cards (Page 40)

Portal view (Page 21)

Project view (Page 23)

2.2.11 Overview window

2.2.11.1 Overview window

Functions of the Overview window

The Overview window supplements the project tree. The Overview window shows the contents of the folder currently selected in the project tree.

In addition, you can perform the following actions in the Overview window:

- Open objects
- Display and edit the properties of objects in the Inspector window
- Rename objects
- Call object-specific actions from the shortcut menu
- Compare objects side by side
- Perform various object operations, such as inserting objects from the library via drag-and-drop and moving, copying, pasting, and deleting objects

Layout of the Overview window

The following figure shows the components of the Overview window:

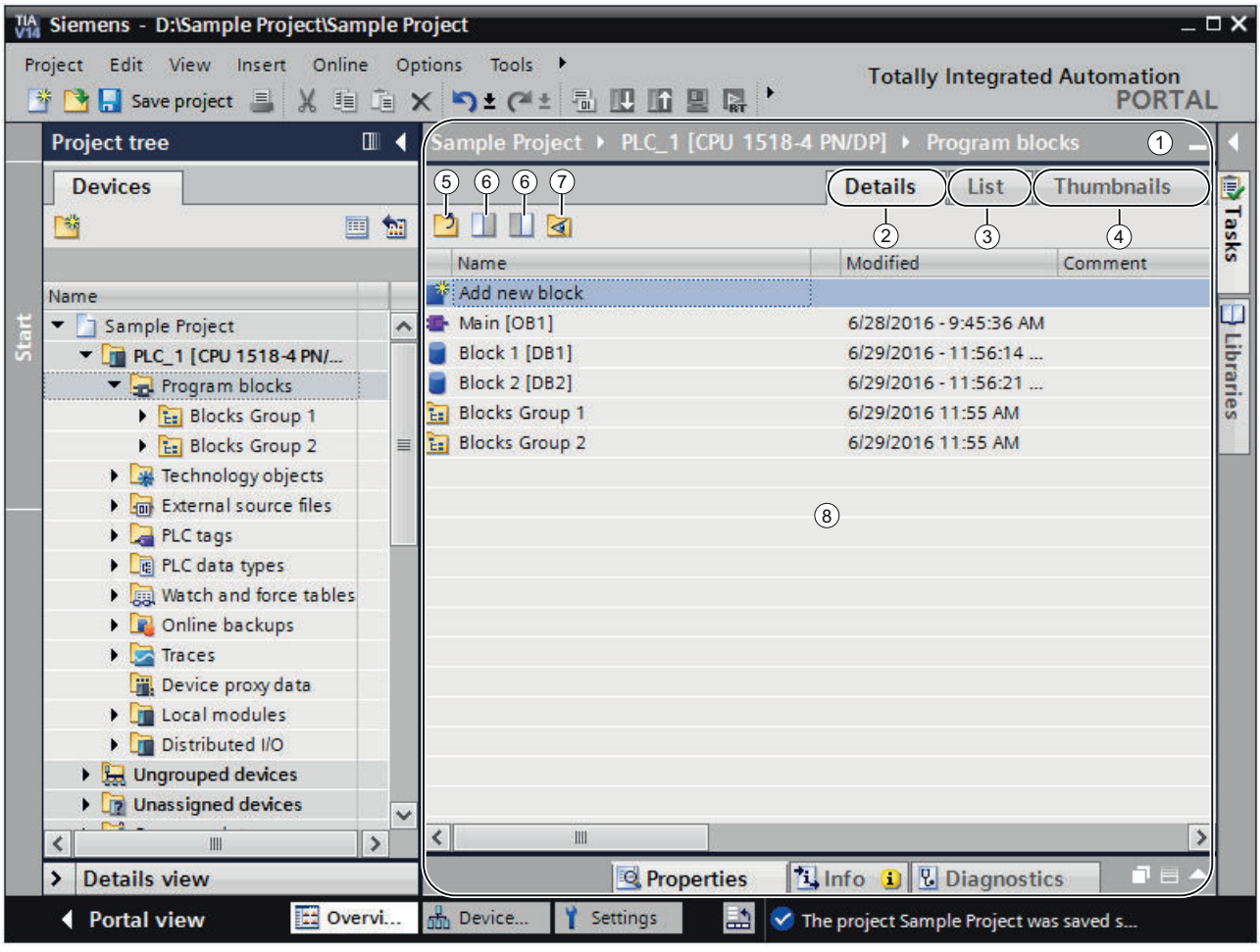

① Overview window

② Switch to the Details view

③ Switch to the List view

④ Switch to the Icon view

⑤ Move to higher level

⑥ Split the overview window in two. Either the right or left half of the overview window is synchronized. Clicking again cancels the split.

⑦ All elements within a selected folder are displayed even if these are located in lower-level groups. This option is only available in details view.

⑧ Contents of the object selected in the project tree.

Display forms of the Overview window

The content of the Overview window can be displayed as follows:

- Details view
  The objects are displayed in a list with additional information, such as the date of the last change.
- List view
  The objects are displayed in a simple list.
- Icon view
  The objects are displayed as icons according to category.

See also

Comparing objects in the overview window (Page 47)

Sorting the details view of the overview window (Page 48)

2.2.11.2 Comparing objects in the overview window

You can display the contents of two folders or objects side by side in the Overview window. The Overview window is split in half and you can display different contents on the left and right sides.

In addition, you can use a drag-and-drop operation to move objects between the split windows. Thus, for example, you can move contents from one window to the other.

Procedure

To split the Overview window in half or cancel the split, follow these steps:

1. In the toolbar, click on the "Synchronize left side" or "Synchronize right side" icon to split the overview window. Either the left or the right side of the overview window synchronized with the contents of the selected object in the project tree.
2. To cancel the split, click again on the previously selected icon.

See also

Overview window (Page 45)

2.2.11.3 Sorting the details view of the overview window

You have several options for adapting the display in the details view of the overview window:

- Adding additional columns
  Some columns are hidden by default to increase clarity. You can display hidden columns if needed. The columns available depend on the selected object.
- Display of folder contents in a flat hierarchy
  Folder contents can be displayed in a flat hierarchy. All the content is displayed at once even if it is located in different groups.
- Sorting the table columns
  You can sort individual columns of the table in ascending or descending order.

Showing or hiding columns

To show or hide additional table columns, follow these steps:

1. Right-click the title bar of the table.
2. Select the "Show/Hide" command in the shortcut menu, and select the columns you want to display.

Displaying folder contents in flat hierarchy

To display the content of a folder in a flat hierarchy, follow these steps:

1. Select the required folder in the project tree or in the library navigation of the library view.
2. Click the "Show subordinate elements" icon in the toolbar.
   All elements are displayed at once in the table even if they are located in subfolders.

Sorting a table in ascending or descending order

To sort the table by a column in ascending or descending order, follow the steps below:

1. Click the table header of a column if you want to sort the column in ascending order.
2. Click again on the same column of the table header to sort the column in descending order.
3. Click a third time on the table header of the same column to cancel the sorting.

See also

Overview window (Page 45)

2.2.12 User interface layout

2.2.12.1 Save user interface layout

Options for backing up the user interface layout

When you make a change to the user interface, this is retained even after a restart of the TIA Portal. A change to the user interface layout includes, for example, moving a window or adjusting the size of an editor.

In addition to the automatic saving of the user interface layout, you have the option of manually backing up specific layouts:

- Save the window layout
  You can save the layouts of the windows and editors of the TIA Portal manually and restore these at a later time. It is possible to call five window layouts using a key combination. Use this function, for example, if you are working with a notebook to which you connect an external monitor when necessary. You can create a window layout for mobile use on the notebook display and another layout for when you work at the office with an external monitor.
- Save the layout within editors
  With some editors, you can adjust the display. You can, for example, adjust the width of tables or show or hide individual table columns.

See also

Save window layout (Page 49)

Load window layout (Page 50)

Managing window layouts (Page 51)

Saving a layout of editors and tables (Page 51)

Resetting the user interface layout (Page 52)

Basics of the work area (Page 30)

2.2.12.2 Save window layout

You can save the current window layout in order to call it again in the same form at a later time.

Procedure

To save a window layout, follow these steps:

1. Arrange all windows in the way in which you want to save them.
2. In the "Window" menu, select the "Save window layout as" command.
   The "Save current window layout as" dialog opens.
3. Enter a name for the window layout in the "Name" field.

4. Enter a description of the window layout in the "Description" field in order to be able to identify the window layout more easily later.
5. Click "Save".

Result

The new window layout is saved in the last position after the existing saved window layouts. The first five window layouts can be called using a key combination.

See also

Save user interface layout (Page 49)

2.2.12.3 Load window layout

If you have already saved a window layout, you can load this, allowing you to quickly adjust your work environment to the respective conditions. You can load the first five window layouts using quick access via the "Window" menu or via a key combination.

If you load a window layout and then make changes to the arrangement of the window, you can restore the originally saved window layout.

Using quick access to load window layouts 1 to 5

To load one of the first five saved window layouts, follow these steps:

1. In the "Window" menu, select a window layout or select the key combination <Alt+Shift+[1 ... 5]>.

Loading additional window layouts

To load a window layout that is not among the first five window layouts, follow these steps:

1. In the "Window" menu, select the "Additional window layouts" command.
   The "Manage all windows layouts" dialog opens.
2. Select the desired window layout.
3. Click "OK".

Restore window layout

To go back to a saved window layout, follow these steps:

1. In the "Window" menu, select the "Restore window layout" command or select the key combination <Alt+Shift+0>.

See also

Save user interface layout (Page 49)

2.2.12.4 Managing window layouts

You can carry out the following actions with existing window layouts:

- Changing the order of window layouts
  The order of the window layouts is important, as the first five window layouts can be called directly via the "Window" menu and via a key combination.
- Select a window layout
  If a window layout is not one of the first five window layouts, you can call it using the "Manage all window layouts" dialog box.
- Deleting window layouts

Procedure

To manage the existing window layouts, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. In the area navigation, navigate to the entry "General > Manage all window layouts".
3. Select a window layout.
   The selected window layout is immediately activated.
4. Click the "Up" or "Down" symbol to move the window layout up or down.
5. Click the "Delete" symbol to delete the selected window layout.

See also

Save user interface layout (Page 49)

2.2.12.5 Saving a layout of editors and tables

You have the option of adapting editors and tables to meet your requirements. For example, you can hide columns in tables that you don't need. You can then save your customized view.

Procedure

To save the layout of editors and tables in the work area, follow these steps:

1. Adapt the editor or table according to your requirements.
2. Click the "Save window settings" button in the editor or table.

Result

The layout is saved. When you reopen the editor or table, this layout will be used.

See also

Basics of the work area (Page 30)

Maximizing and minimizing the work area (Page 32)

Splitting the work area (Page 33)

Floating the work area elements (Page 34)

Using grouped elements of the work area (Page 35)

Minimizing and maximizing elements of the work area (Page 37)

Switching between the elements in the work area (Page 37)

Save user interface layout (Page 49)

2.2.12.6 Resetting the user interface layout

Every change you make to the layout of the user interface is saved. The changes are thus available even after a restart of the TIA Portal. For example, if you change the height and width of a text editor or the division of a table, your changes are retained so that you don't have to re-customize elements every time.

In some cases, however, it may be helpful to restore the original layout settings; for example, if another user prefers a different arrangement of the user interface.

Procedure

To reset the user interface settings to the default, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation.
3. Click the "Reset to default" button under "Reset to default > Editor layout".

Result

The default settings for the user interface are restored.

See also

Overview of the program settings (Page 10)

Save user interface layout (Page 49)

2.3 Running external applications

2.3.1 Basics for running external applications

Introduction

You have the option of configuring up to twenty external applications, which you can run directly from the TIA Portal. You can set arguments for the configuration for these applications. You can define your own arguments, which the application understands or you can select the following system variables of the TIA Portal as arguments:

- ProjectFileName: Name of the currently open TIA Portal project
- ProjectVersion: Version of the currently open TIA Portal project
- ProjectPath: Path of the currently open TIA Portal project
- ProjectDirectory: Project list of the currently open TIA Portal project
- ProjectExtension: Filename extension of the currently open TIA Portal project
- BinaryDirectory: Path for the "Bin" directory of the opened TIA Portal instance
- ProductVersion: Version of the open TIA Portal

You can also mix your own arguments with the system variables.

You can also specify the start path for the application. Here you can either use one of the following two pre-defined values or enter your own start path:

- ProjectDirectory: Project list of the currently open TIA Portal project
- BinaryDirectory: Path for the "Bin" directory of the opened TIA Portal instance

See also

Configuring external applications (Page 53)

Start the external application (Page 55)

2.3.2 Configuring external applications

Before you can open external applications from the TIA Portal, you must first configure these applications. For external applications already available, you can subsequently modify the configuration and edit the order in which they are displayed in the menu "Tools > External Applications". You can delete external applications that you no longer want to call.

Adding and configuring a new external application

To configure a new external application, follow these steps:

1. Click on the "Configure" command in the menu "Tools > External Applications" or open the list of applications from the menu "Options > Settings > External Applications".
   The list of applications is displayed in the working area.
2. Double-click on "Add" or select the "Add" command from the shortcut menu.
   The "Add new external application" dialog opens.
3. Enter the name of the external application. This name cannot be used for a previously configured application and must not exceed 100 characters.
4. Enter the command or select the path to the executable file.
5. Optional: Enter the arguments to be passed when the external application starts. You can also use the "+" button next to the "Arguments" field to open a list of system variables of the TIA Portal and select these as arguments.

   Note

   If an argument, for example, a system variable, cannot be transferred when the external application starts, this argument is passed as an empty string.

6. Optional: Click the "+" button next to the "Start in" field and select the start position.
7. Click on "Add".
   The new external application is added to the list of applications and can now be started via the menu "Tools > External Applications" with the name entered. The order of the display in the menu "Tools > External Applications" corresponds to the order in the list of applications. You can change this order as desired.

Configuration of an existing external application

To change an existing external application, follow these steps:

1. Click on the "Configure" command in the menu "Tools > External Applications" or open the list of applications from the menu "Options > Settings > External Applications".
   The list of applications is displayed in the working area.
2. Select the external application that you want to change.
3. Click on the "Edit" button or select "Edit" from the shortcut menu.
   The "Change external application" dialog opens.
4. Change the configuration to meet your requirements.
5. Click "Change".

Change the order of the external applications in the list

To change the order of the external applications in the list and thus also in the "Tools > External Applications" menu, follow these steps:

1. Click on the "Configure" command in the menu "Tools > External Applications" or open the list of applications from the menu "Options > Settings > External Applications".
   The list of applications is displayed in the working area.
2. Select the external application whose position you want to change.
3. Click the "Move up" button to move the application up in the list.
4. Click the "Move down" button to move the application further down in the list.
5. Repeat this for all applications for which you want to change the order.

Deleting an existing external application

To delete an existing external application, follow these steps:

1. Click on the "Configure" command in the menu "Tools > External Applications" or open the list of applications from the menu "Options > Settings > External Applications".
   The list of applications is displayed in the working area.
2. Select the external application that you want to delete.
3. Click on the "Delete" button or select "Delete" command in the shortcut menu.
   The "Confirm delete" dialog opens.
4. Click "Yes" to delete the external application.

See also

Basics for running external applications (Page 53)

Start the external application (Page 55)

2.3.3 Start the external application

Requirement

You have configured an external application.

Procedure

To launch an external application, follow these steps:

1. In the "Tools > External Application" menu, click the external application that you want to start.
   If you start the application for the first time or have changed the configuration for this external application since the last start, the "Start application" dialog is displayed. Then perform step 2.
2. Click "OK" if you want to start the external application despite possible security risks. Otherwise, cancel the startup by clicking "Cancel".

See also

Basics for running external applications (Page 53)

Configuring external applications (Page 53)

2.4 Keyboard operation in the TIA Portal

2.4.1 Operation of the TIA Portal with the keyboard

You can navigate through the TIA Portal using the keyboard, for example if you do not have a mouse available at the given moment. Additional keyboard shortcuts are available for many functions; alternatively, you can use user-defined keyboard shortcuts. You can find an overview of all keyboard shortcuts and the option of assigning user-defined keyboard shortcuts in the settings for the TIA Portal.

In the following sections, you will learn how to navigate in the TIA Portal using the keyboard, edit objects, and customize the TIA Portal to your needs.

See also

Displaying an overview of all keyboard shortcuts (Page 57)

Using user-defined keyboard shortcuts (Page 68)

2.4.2 Displaying an overview of all keyboard shortcuts

You can display an overview of all keyboard shortcuts.

Procedure

To display an overview of all available keyboard shortcuts, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Open the "Keyboard shortcuts" entry in the area navigation.
   You can see an overview of all default and user-defined keyboard shortcuts, which are valid for the currently installed products.

2.4.3 Basic functions of the TIA Portal

Below, you will learn how you can use the basic functions of the TIA Portal using only your keyboard.

Using the basic functions of the TIA Portal with the keyboard

The following table shows how you can access basic functions of the TIA Portal with keyboard shortcuts:

Function	Keyboard shortcut	Menu command
Change between the project view and the portal view	<Alt+F7>	
Change between the project view and the library view	<Alt+F5>	

Function	Keyboard shortcut	Menu command
Open the online and diagnostics view	<Ctrl+D>	
Start Runtime	<Ctrl+Shift+Z>	
Display cross-references	<Shift+Alt+F11>	
Cross-reference information	<Shift+F11>	
Open the Help system If you need help on the TIA Portal, press <F1>.	<F1>	Help > Show help
Cancel the current action	<Esc>	
Find	<Ctrl+F>	
Replace an object You can replace found objects when searching in the editor.	<Ctrl+H>	
Find next If you have started a search in the editor, you can jump to the next hit with <F3>.	<F3>	
Print object	<Ctrl+P>	Project > Print
Run default action of the object	<Enter>	
Action	<F5>	
Scroll horizontally to the right	<Ctrl+Arrow right>	
Scroll horizontally to the left	<Ctrl+Arrow left>	
Open block/PLC data type	<F7>	

Operating menus

The following table shows how you can navigate through menus using the keyboard:

Function	Keyboard shortcut
Start keyboard shortcuts in the menu You can access the menu using the <Alt> key and then continue to navigate with the arrow keys to scroll through the menu. Confirm your selection of the menu command with <Enter>.	<Alt>
Go directly to a specific menu You can go directly to an individual menu command by holding down the <Alt> key. There is an underlined letter for each menu command. Press the <Alt> key along with the underlined letter.	<Alt+underlined letter in respective menu>
Open shortcut menu of an object With the shortcut menu key (on Microsoft Windows compatible keyboards), you can open the shortcut menu of the selected object. Alternatively, you can use <Shift+F10> if you are not using a Microsoft Windows compatible keyboard. You can use the arrow keys to scroll through the shortcut menu and select a menu command with <Enter>.	<Shortcut menu key> Alternative: <Shift+F10>

Operating expandable elements

The following table shows how you can operate expandable elements using the keyboard:

Function	Keyboard shortcut
Expand a folder in a tree With <Arrow right>, for example, you expand a folder in the project tree.	<Arrow right>
Close a folder in a tree With <Arrow left>, for example, you collapse a folder in the project tree.	<Arrow left>
Open a drop-down list You can open drop-down lists with <F4> and then navigate with the arrow keys to scroll through the drop-down list. Finally, press the <Enter> key to confirm your selection.	<F4>
Open autocompletion	<Ctrl+Space> <Ctrl+I>
Show object selection	<Ctrl+J>

2.4.4 Using project-related functions

Editing a project

Function	Keyboard shortcuts	Menu command
Open a project	<Ctrl+O>	Project > Open
Close a project	<Ctrl+W>	Project > Close
Save a project	<Ctrl+S>	Project > Save
Save a project under a different name	<Ctrl+Shift+S>	Project > Save as
Delete a project	<Ctrl+E>	Project > Delete project
Print project	<Ctrl+P>	Project > Print
Undo last action	<Ctrl+Z>	Edit > Undo
Redo last action	<Ctrl+Y>	Edit > Redo

Calling up the help function

Function	Keyboard shortcuts	Menu command
Calling up the help function	<F1> or <Shift+F1>	Help > Show help

2.4.5 Arranging windows

Below, you will learn how to open and close individual windows of the TIA Portal and work with saved window layouts using the keyboard.

Opening and closing windows

The following table shows how you can open and close windows with keyboard shortcuts:

Function	Keyboard shortcut	Menu command
Open/close project tree	<Ctrl+1>	View > Project tree
Opening/closing the detailed view	<Ctrl+4>	View > Details view
Opening/closing the overview	<Ctrl+2>	View > Overview
Opening/closing a task card	<Ctrl+3>	View > Task card
Open libraries	<Ctrl+Shift+L>	
Open hardware catalog If you are in the device or network view, the hardware catalog opens.	<Ctrl+Shift+C>	
Open/close inspector window	<Ctrl+5>	View > Inspector window
Open the "Properties" tab in the inspector window	<Ctrl+6>	
Open the "Info" tab in the inspector window	<Ctrl+7>	
Open the "Diagnostics" tab in the Inspector window	<Ctrl+8>	
Display or hide reference projects	<Ctrl+9>	
Display the on-screen keyboard You can display a keyboard on the screen, for example, to operate with a touch screen.	<Ctrl+Shift+K>	
Close all editors	<Ctrl+Shift+F4>	Window > Close all

Using saved window layouts

You can save individual window arrangements and restore them at a later point in time. The following table shows how you to access saved window layouts with keyboard shortcuts:

Function	Keyboard shortcut	Menu command
Restore active window layout If you use a saved window layout and have made changes to the program interface in the meantime, you can restore the original state of the active window layout with <Alt+Shift+0>.	<Shift+Alt+0>	Window > Restore window layout
Load window layout You can use <Alt+Shift+[number of the window layout]> to activate the first of the five saved window layouts.	<Shift+Alt+[Number of the window layout]>	Window > Window layout 1 to 5

2.4.6 Navigating through the program interface

The TIA Portal is divided into various user interface areas, for example, individual windows, toolbars, and editors. If you want to work with the keyboard within an interface area, you first have to place the focus on it. Below, you will learn how to place the focus on individual interface areas using the keyboard. You will also learn how to move within an interface area in the TIA Portal using the keyboard.

Switching between interface areas and editors

The following table shows how to move between individual interface areas of the TIA Portal:

Function	Keyboard shortcut
Move clockwise between the interface areas You can use the <F6> key to move clockwise between the individual interface areas of the TIA Portal. The interface area currently in focus is highlighted with a blue title bar. If you are in the project tree, for example, and press the <F6> key, you jump to the currently open editor. If you press <F6> again, the task cards are in focus. If you press <Shift+F6>, on the other hand, you move counterclockwise between the work areas.	<F6>
Move counterclockwise between the interface areas With <Shift+F6> you move counter-clockwise between the interface areas of the TIA Portal.	<Shift+F6>
Go to the next open editor With <Ctrl+Alt+Arrow right> you go to the next open editor. You can see the open editors in the editor toolbar.	<Ctrl+Alt+Arrow right> Alternative: <Ctrl+F6>
Go to the previous open editor With <Ctrl+Alt+Arrow left>, you go to the last opened editor.	<Ctrl+Alt+Arrow left> Alternative: <Ctrl+Shift+F6>
Go to the next higher section of the interface area With <Shift+Esc>, you move to the next higher section of the program interface. If, for example, you have selected a device in the project tree and you press <Shift+Esc>, the entire project navigation is put into focus.	<Shift+Esc> Alternative: <Alt+Arrow up>
Go to the next lower section of the interface area With <Enter>, you place the focus on the next lower section of the program interface. For example, if you have just opened the properties of a device in the Inspector window in order to assign parameters to the device, press <Enter> to go one level deeper in the program interface. You can then navigate to the desired parameter using the Tab key.	<Enter> Alternative: <Alt+Arrow down>

Navigation within interface areas and editors

The following table shows how you can navigate within a user interface area using the keyboard:

Function	Keyboard shortcut
Jump to the next element within an interface area You can use the Tab key to jump from one element to the next within a work area. If, for example, you have opened the properties of a device and want to jump from one field to the next, press the Tab key. Any changes you have made to the current text box are applied in this case.	<Tab>
Jump to the previous element within an interface area With <Shift+Tab>, you can jump to the previous element in a work area, for example, a previous text box. Any changes you have made to the current text box are applied in this case.	<Shift+Tab>
Jump to the higher-level element within an interface area With <Shift+Home> you can jump to the higher-level element of a work area, e.g., to the higher-level folder in the project tree.	<Shift+Home>

Function	Keyboard shortcut
Move to the next tab within an interface area If an interface area is divided into separate tabs, you can move between the tabs with the shortcut keys <Ctrl+Tab>. If, for example, you are in the "Properties" tab of the Inspector window and you want to jump to the "Info" tab, press the shortcut keys <Ctrl+Tab>.	<Ctrl+Tab>
Go to the previous tab With <Ctrl+Shift+Tab>, you move to the most recently open tab within an interface area.	<Ctrl+Shift+Tab>
Jumping to the toolbar of an editor You can use the <Alt+F10> key to jump to the toolbar of an editor. For example, if you have opened the print preview and want to switch to the next page of the printout in the toolbar, press <Alt+F10>. Then, use the arrow keys to navigate to the appropriate icon in the toolbar and confirm the selection with <Enter>.	<Alt+F10>
Use arrows on the divider to display or hide user interface components The table in the work area can be minimized and maximized. First, navigate to the work area and use the Tab key to place the focus on one of the little arrows on the separator line above the table. The arrows have the focus as soon as they are highlighted in blue. Then, press the space bar to minimize or maximize the table.	<Space>

2.4.7 Customizing editors

Below, you will learn how to arrange editors using the keyboard. You will also learn how to select the display size and the area within a graphical editor.

Arranging and customizing editors

The following table shows how to arrange open editors above and below each other or side-by-side, and how to close an open editor:

Function	Keyboard shortcut	Menu command
Close active editor	<Ctrl+F4>	
Split editor space vertically If, for example, you have opened the overview window and the network view and want to display them side-by-side, press the <F12> key.	<F12>	Window > Split editor space vertically
Split editor space horizontally You can display two open editors in the work area above and below each other.	<Ctrl+F12>	Window > Split editor space horizontally
Remove window split If you have displayed two editors in the work area horizontally or vertically in split mode, you can remove the split with <Alt+Shift +F12>.	<Alt+Shift+F12>	Window > Unsplit editor space

Customizing the display in an editor

The following table shows how you how to zoom in and out in graphical editors and how to move the area selection in an editor:

Function	Keyboard shortcut
Zoom in step-by-step in an editor With <Ctrl> and the <Plus> key on the numeric keypad of the keyboard, you can zoom in on the display of the editor.	<Ctrl+Plus> Alternative: <Ctrl+Scroll up>
Zoom out step-by-step in an editor Use the <Ctrl> and the <Minus> key on the numeric keypad of the keyboard to zoom out of the display of the editor.	<Ctrl+Minus> Alternative: <Ctrl+Scroll down>
Set view in the editor to 100% Press <CTRL+0> to enlarge or reduce the current view in a graphical editor to 100%.	<Ctrl+0>
Move the area selection of the editor If you hold down the spacebar, you can move the displayed section of an editor using the mouse.	<Space>

2.4.8 Editing objects

Selecting objects

The following table shows how to select individual objects, for example, devices in the project tree:

Function	Keyboard shortcut	Menu command
Select an object located at the left, right, above or below	<Arrow keys>	
Jump to the first object within the current interface area The first object in the interface area currently in focus is selected. In the project tree, this would be the project node at the top, for example.	<Home>	
Jump to the last object within the current interface area The last object in the interface area currently in focus is selected, for example, the last item in the project tree.	<End>	
Select all objects in an area All objects in the work area currently in focus are selected.	<Ctrl+A>	Edit > Select all
Select multiple objects If you want to select several objects that are not located directly next to each other, you first have to move the focus (gray outline of an object) to the next desired object using <Ctrl+Arrow keys>. The current selection is maintained. Then, press the space bar to select the new focused object as well. Repeat this process until all desired objects are selected.	<Ctrl+Arrow keys> + <Space>	

Editing objects

The following table provides an overview of all the keyboard shortcuts required for editing objects:

Function	Keyboard shortcut	Menu command
Insert new object A new object is inserted depending on your current context. If, for example, you are in the device view, the "Add Device" dialog opens for creating a new device.	<Ctrl+N>	
Open object	<Enter>	
Rename an object	<F2>	Edit > Rename
Copy an object	<Ctrl+C> Alternative: <Ctrl+Ins>	Edit > Copy
Cut an object	<Ctrl+X> Alternative: <Shift+Delete>	Edit > Cut
Paste an object	<Ctrl+V> Alternative: <Shift+Insert>	Edit > Paste
Delete an object	<Del>	Edit > Delete
Compile an object	<Ctrl+B>	Edit > Compile
Open properties of an object Many objects in the TIA Portal have editable properties. Press the shortcut keys <Alt+Enter> to display the properties of an object.	<Alt+Enter>	-

2.4.9 Text editing

Below, you will learn how to operate text editing functions using only the keyboard.

Editing text

The following table shows the basic editing functions for text:

Function	Keyboard shortcuts
Switch to insert or overwrite mode	<Insert>
Exit edit mode	<Esc>
Delete	<Del>
Delete characters	<Backspace>
Confirm entry in a text box and leave the text box	<Return>
Line break in a multiline text box In a multiline text box, hold down the <Shift> button to create a line break.	<Shift+Return>
Reset input in a text box If you are in an text box and press <Esc>, you exit the box and the changes are discarded.	<Esc>

Navigating within a text area

The following table shows how to navigate in a text area with the keyboard:

Function	Keyboard shortcuts
Jump to start of line	<Home>
Jump to end of line	<End>
Jump to start of text	<Ctrl+Home>
Jump to end of text	<Ctrl+End>
Jump to the previous page	<PgUp>
Jump to the next page	<PgDn>
Confirm entry in a text box and leave the text box	<Return>
Line break in a multiline text box	<Shift+Return>
Reset input in a text box If you are in an text box and press <Esc>, you exit the box and the changes are discarded.	<Esc>

Selecting text

The following table shows how to select text with the keyboard:

Function	Keyboard shortcuts
Expand selection to the word at the left or right The text or current text selection is marked up to the end of word. If you are at the beginning or end of a word, the previous or next word is selected.	<Ctrl+Shift+Arrow left or Arrow right>
Expand selection to beginning of line	<Shift+Home>
Expand selection to end of line	<Shift+End>
Expand selection to beginning of text The text is selected up to the beginning or the end.	<Ctrl+Shift+Home>
Expand selection to end of text The text is selected up to the beginning or the end.	<Ctrl+Shift+End>

2.4.10 Editing tables

Below, you will learn how to navigate with the keyboard in tables, edit individual fields, and select parts of tables.

General keyboard operation in tables

The following table shows how you can edit tables using only the keyboard:

Function	Keyboard shortcut
Place a cell in edit mode	<F2> or <Return>
Confirm entry and exit edit mode	<Enter>
Cancel editing and discard changes	<Esc>

Function	Keyboard shortcut
Open drop-down list in a cell Open the drop-down list with <F4>. Use the arrow keys to select the desired entry and then confirm the selection with <Return>.	<F4>
Close drop-down list in a cell and discard changes	<Esc>

Navigate in tables

The following table shows how you can navigate within a table using the keyboard:

Function	Keyboard shortcut
Go to line	<Ctrl+G>
Go to the next cell	<Arrow keys>
Go to the next editable cell on the right	<Tab>
Go to the next editable cell on the left	<Shift+Tab>
Move a screen upwards	<PgUp>
Move a screen downwards	<PgDn>
Go to the first cell in the row	<Home>
Go to the last cell in the row	<End>
Go to the first cell in the table	<Ctrl+Home>
Go to the last cell in the table	<Ctrl+End>
Go to the top cell in the column	<Ctrl+Arrow up>
Go to the bottom cell in the column	<Ctrl+Arrow down>
Insert table row below	<Alt+Ins>
Insert line break in an entry field/insert table row above	<Ctrl+Return>

Selecting areas in tables

The following table shows how you can select areas within a table using the keyboard:

Function	Keyboard shortcut
Select column	<Ctrl+Space>
Select line	<Shift+Space>
Select all cells	<Ctrl+A>
Expand selection by one cell	<Shift+arrow keys>
Extend selection up one page	<Shift+PgUp>
Extend selection down one page	<Shift+PgDn>
Expand selection up to the first row	<Ctrl+Shift+Arrow up>
Expand selection down to the last row	<Ctrl+Shift+Arrow down>
Expand selection to the first cell in the row	<Ctrl+Shift+Arrow left>
Expand selection to the last cell in the row	<Ctrl+Shift+Arrow right>

2.4.11 Using online functions

Controlling online functions with the keyboard

The following table provides an overview of the shortcut keys that you can use for the online functions of the TIA Portal:

Function	Keyboard shortcut	Menu command
Establish an online connection	<Ctrl+K>	Online > Go online
Go offline	<Ctrl+M>	Online > Go offline
Download project data to the device	<Ctrl+L>	Online > Download to device
Show accessible devices This opens a dialog showing all devices that are connected to the PG/PC interface of the PG/PC.	<Ctrl+U>	Online > Show accessible devices
Start CPU The CPU is set to "RUN" mode. The CPU must be online for this.	<Ctrl+Shift+E>	Online > Start CPU
Stop CPU The CPU is set to "STOP" mode. The CPU must be online for this.	<Ctrl+Shift+Q>	Online > Stop CPU
Start simulation The hardware and software of the project can be tested in a simulated online environment, without the modules actually being online.	<Ctrl+Shift+X>	Online > Simulation > Start

2.4.12 Using the on-screen keyboard

Introduction

When working with the TIA Portal, you also have the Microsoft on-screen keyboard available.

Displaying the on-screen keyboard

To display the on-screen keyboard, follow these steps:

1. In the "View" menu, select the "Screen keyboard" command.

Exiting the on-screen keyboard

To exit the on-screen keyboard, follow these steps:

1. In the "File" menu of the on-screen keyboard, select the "Exit" command.

2.4.13 Using user-defined keyboard shortcuts

2.4.13.1 Assigning user-defined keyboard shortcuts

You can set your own keyboard shortcuts for operating the TIA Portal.

Procedure

To assign user-defined keyboard shortcuts, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Open the "Keyboard shortcuts" entry in the area navigation.
   You can see a table with an overview of all keyboard shortcuts, which are valid for the currently installed products.
3. Click on a row in the table.
   A button appears in the "User-defined keyboard shortcuts" cell.
4. Click on the button.
   The "Edit user-defined keyboard shortcuts" dialog opens. In the upper part of the dialog you can see the currently selected action, the editor and the currently assigned keyboard shortcut.
5. Select the "Assign new keyboard shortcuts" option.
   The option is set by default.
6. Open the drop-down list under the "Assign new keyboard shortcuts" option.
   The available keyboard shortcuts, not used in TIA Portal, are displayed.
7. Select a keyboard shortcut.
8. Click "OK" to apply your selection and close the dialog box.
9. Restart the TIA Portal in order to apply the assigned keyboard shortcut.
   The assigned keyboard shortcut is no longer available for new user-defined keyboard shortcuts. The previously used keyboard shortcut is available for new, user-defined keyboard shortcuts.
   User-defined keyboard shortcuts overwrite default keyboard shortcuts.

2.4.13.2 Deleting user-defined keyboard shortcuts

If you have assigned user-defined keyboard shortcuts for functions in the TIA Portal, you can also delete these keyboard shortcuts.

Procedure

To delete user-defined keyboard shortcuts, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Open the "Keyboard shortcuts" entry in the area navigation.
   You can see a table with an overview of all keyboard shortcuts, which are valid for the currently installed products.
3. Click on a row in the table.
   A button appears in the "User-defined keyboard shortcuts" cell.
4. Click on the button.
   The "Edit user-defined keyboard shortcuts" dialog opens. In the upper part of the dialog you can see the currently selected action, the editor and the currently assigned keyboard shortcut.
5. Select the "Remove keyboard shortcut" option.
6. Click "OK" to apply your selection and close the dialog box.
7. Restart the TIA Portal in order to apply the deletion of the keyboard shortcut.
   The deleted keyboard shortcut is again available for new, user-defined keyboard shortcuts.

2.4.13.3 Resetting user-defined keyboard shortcuts to default setting

You have the option to discard the user-defined keyboard shortcuts and to reset to the default setting.

Procedure

To reset user-defined keyboard keyboard shortcuts to the default setting, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Open the "Keyboard shortcuts" entry in the area navigation.
   You can see a table with an overview of all keyboard shortcuts, which are valid for the currently installed products.
3. Click the "Reset to default setting" button on the right below the overview of the keyboard shortcuts.
   A confirmation dialog appears.
   The keyboard shortcuts are reset to the default setting.

2.4.13.4 Importing and exporting user-defined keyboard shortcuts

To use your user-defined keyboard shortcuts in another installation or another product, you can export and import these keyboard shortcuts.

Importing user-defined keyboard shortcuts

To import user-defined keyboard shortcuts, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Open the "General" entry in the area navigation.
3. In the "Import/export settings" area, click on the "Import settings ..." button.
4. Select the setting file with the extension ".tps15".
5. Click "Open".
   The settings are applied after a restart of the TIA Portal.

Exporting user-defined keyboard shortcuts

To export user-defined keyboard shortcuts, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Open the "General" entry in the area navigation.
3. In the "Import/export settings" area, click on the "Export settings ..." button.
   The "Export settings" dialog opens.
4. Select the "Keyboard shortcuts" check box.
5. Enter a file name for the initialization file in the "File name" box.
6. Click "Export".
   Settings are exported to a file with the extension ".tps15".

2.4.13.5 Selecting a list of keyboard shortcuts

You can use either the user-defined keyboard shortcuts or the system-defined keyboard shortcuts. You can make a selection in the settings.

Procedure

To select a list for keyboard shortcuts, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Open the "Keyboard shortcuts" entry in the area navigation.
   You can see a table with an overview of all keyboard shortcuts, which are valid for the currently installed products.
3. In the selection list above the overview of the keyboard shortcuts, select one of the following lists:
   - User-defined keyboard shortcuts
   - System-defined keyboard shortcuts

Note

Assignment of user-defined keyboard shortcuts

If the "System-defined keyboard shortcuts" list is selected, no user-defined keyboard shortcuts can be assigned.

2.5 Special features specific to the operating system

2.5.1 Influence of user rights

Restrictions when user rights are limited

The software provides several functions that require direct access to the hardware of the programming device / PC and therefore also to the installed operating system. To make full use of the range of functions, the software must cooperate closely with the operating system. To ensure problem-free interaction, you should therefore be logged on to the operating system with adequate user rights.

In particular, you may not be able to use functions requiring an online connection or those that change the settings of interface cards if you work with limited user rights.

Recognizing restricted functions

You can recognize functions requiring special rights as follows:

- A shield icon is displayed beside the function.
  The function can be used but is regulated by the user account control.
- A box is grayed out and cannot be accessed.
  You require administrator privileges to access the box. In some operating system environments, you can obtain administrator privileges by entering an administrator password.

Note

A box being grayed out does not necessarily mean a lack of rights. You should also check the additional information in the tooltip cascades to find out the conditions for editing the box.

2.5.2 Expanding user rights

Counteracting restrictions due to user rights

Certain functions may not be available if you are not logged on to the operating system with adequate rights. You can counteract these restrictions in the following ways:

- Enabling of extended rights using Windows user account control
- Logging on to the operating system with administrator privileges
- Using temporary administrator rights

Enabling extended rights using Windows user account control

To be able to use a function indicated by the shield icon of the Windows user account control, follow these steps:

1. Click on the box or button with the shield icon.
   The security prompt of the Windows user account control opens.
2. Follow the instructions of the Windows user account control and, when prompted enter an administrator password, if possible.

The function can now be used once without restrictions.

Logging on to the operating system with administrator privileges

To be able to use a function that is disabled due to lack of user rights, follow these steps:

1. Close the software.
2. Log off from the operating system.
3. Log on to the operating system with administrator privileges.
4. Restart the software.

Using temporary administrator rights

To obtain administrator privileges temporarily, follow these steps:

1. Click the "Change settings" button. You will find this button in dialogs that allow the temporary assignment of administrator privileges.
   An operating system dialog box for entering an administrator password opens.
2. Enter an administrator password.

The settings can be temporarily changed. When you call the dialog again, the procedure must be repeated.

Note

This function is not supported by all operating systems. If no "Change settings" button is present or the button is grayed out, you will need to log on to the operating system with administrator privileges instead.

Help on the information system 3

3.1 General remarks on the information system

Quick answers to your questions

The information system of the TIA Portal helps you solve your problems and offers the required help topics at each step of the configuration.

While working with the program, you receive the following support:

- Information system with all the background information, step-by-step instructions and examples that you need when working with the TIA Portal.
- Tooltips for information on elements of the user interface, for example text boxes, buttons and icons. Some of the tooltips are supplemented by cascades containing more precise information.
- Help on the current context, for example on menu commands when you press the <F1> key.
- Help on messages
- Roll-outs for correct inputs in dialog boxes
- Banner on the software interface

Information system

The information system opens in a separate window. The following figure shows the information system for the TIA Portal:

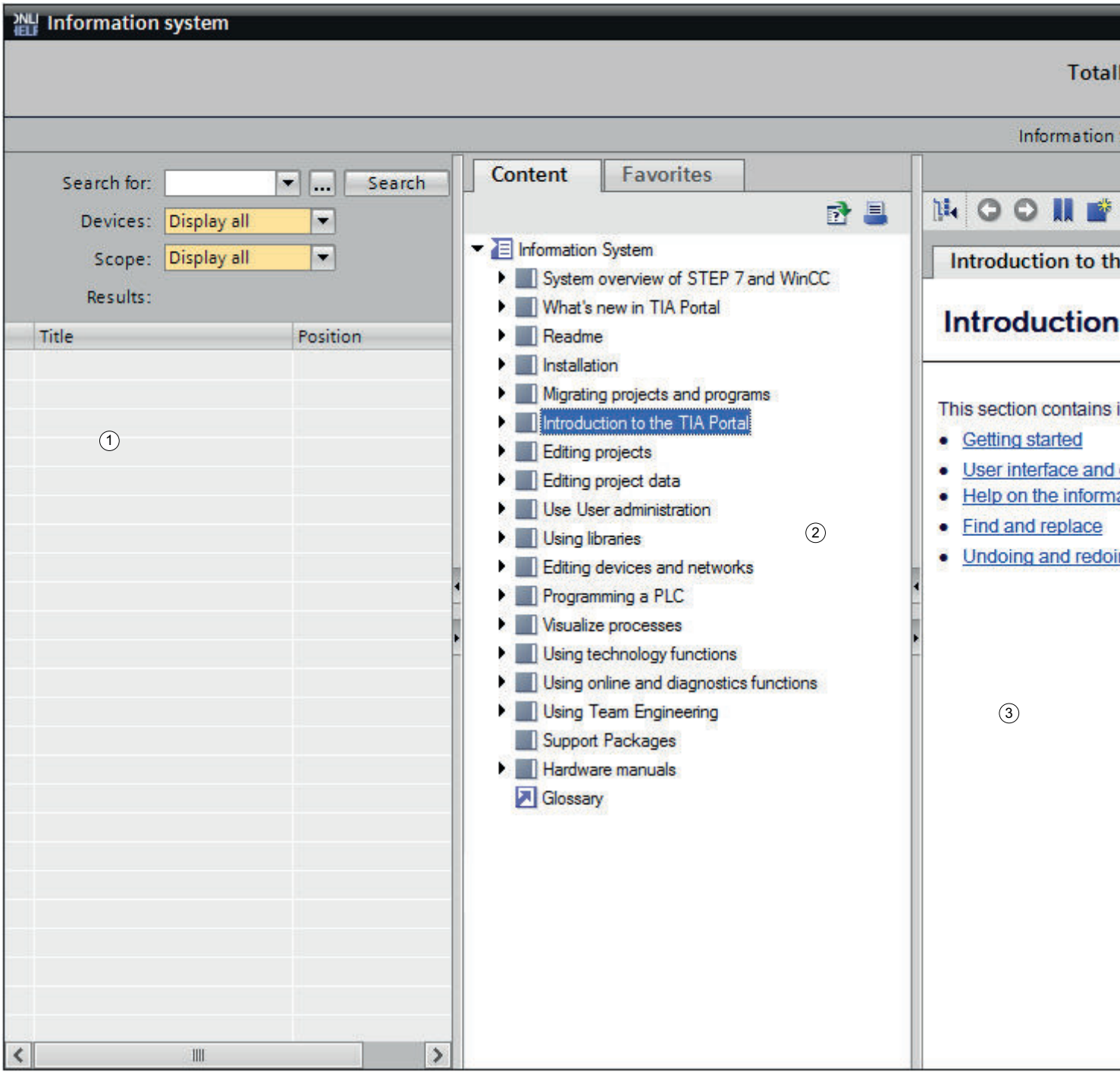

① Search area

In the search area, you can perform a full text search across all help topics.

② Navigation area

You can find the table of contents and favorites in the navigation area.

③ Content area

The help pages appear in the content area. You can open multiple tabs to view various help pages at the same time.

The arrows on the window dividers allow you to display and hide the individual sections. You can open both the search area and the navigation area to increase the contents area as needed.

Buttons in the navigation area

The following buttons are available in the navigation area of the information system:

Button	Function
	Search for updates Starts the search for hardware manuals, which are available as an update.
	Print Prints out pages or sections of the information system.

Buttons in the contents area

The following buttons are available in the contents area of the information system:

Button	Function
	Synchronize Displays the position of the open help topic in the table of contents.
	Forward / backward Navigates through the history of the help topics previously opened in this tab.
	Favorites Adds the current help topic to the favorites.
	New tab Opens a new, empty tab.
	Home page Displays the start page of the information system.
	Printing Prints the current help topic.
	Help Displays help on the information system.

Identification of the topics according to the information type

The information system provides you with a wide range of information. The help topics are identified by different symbols depending on the type of information they contain. Symbols tell you at a glance what kind of information you have just opened.

Symbol	Information type	Explanation
	Operating instructions	Describes the steps to follow in order to carry out a particular task.
	Example	Contains a practical application example that explains how to solve an automation task.
	Factual information	Provides background information on the functions of the TIA Portal and provides basic knowledge.
	Reference	Provides detailed reference information about instructions and objects for reference.

Identification of the validity of help topics

The information system describes the full range of functions for the installed product. However, the range of functions that can be used by you depends on the license purchased and the devices used.

For this reason, help topics which are only valid for certain devices or target systems are marked as follows:

- Indicates the validity within the table of contents
- Indicates the validity in the search results in full-text search
- Indicates the validity in the title bar of a section or help topic. The title bar contains a symbol whose tooltip indicates the validity.

The following figure shows the validity indication in the title bar of a help topic:

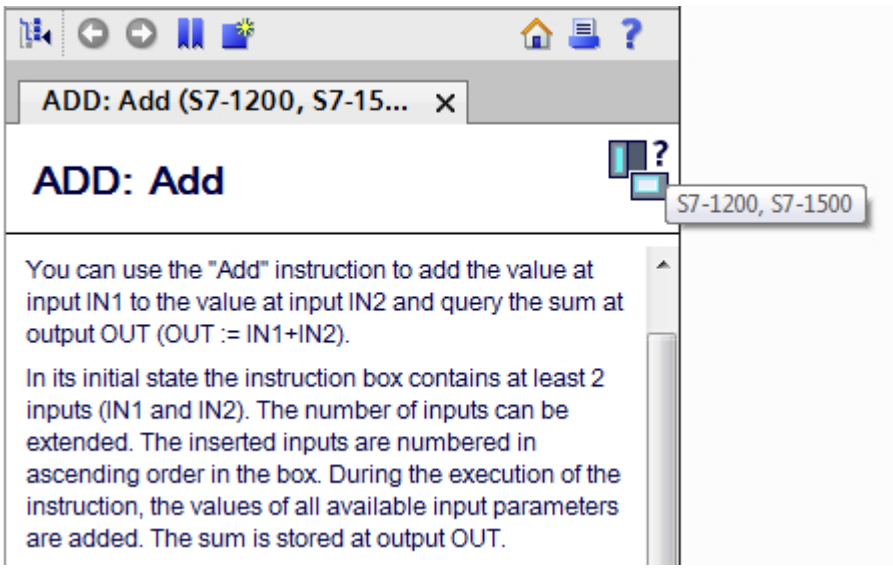

In addition, you can use the "Devices" search filter to limit the search to a specific device or device series.

See also: Searching help topics (Page 86)

Roll-out

Certain text boxes offer information that rolls out and helps you to enter valid parameters and values. The roll-out informs you about permitted value ranges and data types of the text boxes.

The following figure shows a roll-out (yellow) and a roll-out error message (red), which indicates an invalid value:

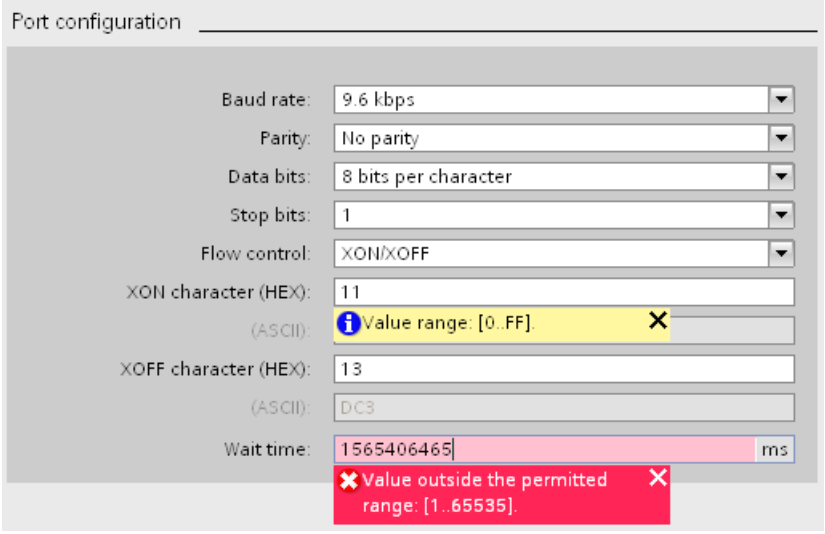

The roll-out is closed as soon as you exit the window or click the x in the upper right corner.

Tooltip

User interface elements offer brief information in the form of a tooltip.

Tooltips, which have an arrow icon on the left, contain additional information in tooltip cascades. If you position the mouse pointer briefly over the tooltip or click the arrow icon, this information is displayed. The automatic display of tooltip cascades can be disabled.

If additional information is available, a link to the corresponding help topic appears in the cascade. If you click on the link, the corresponding topic opens in the information system.

The following figure shows a tooltip with opened cascade:

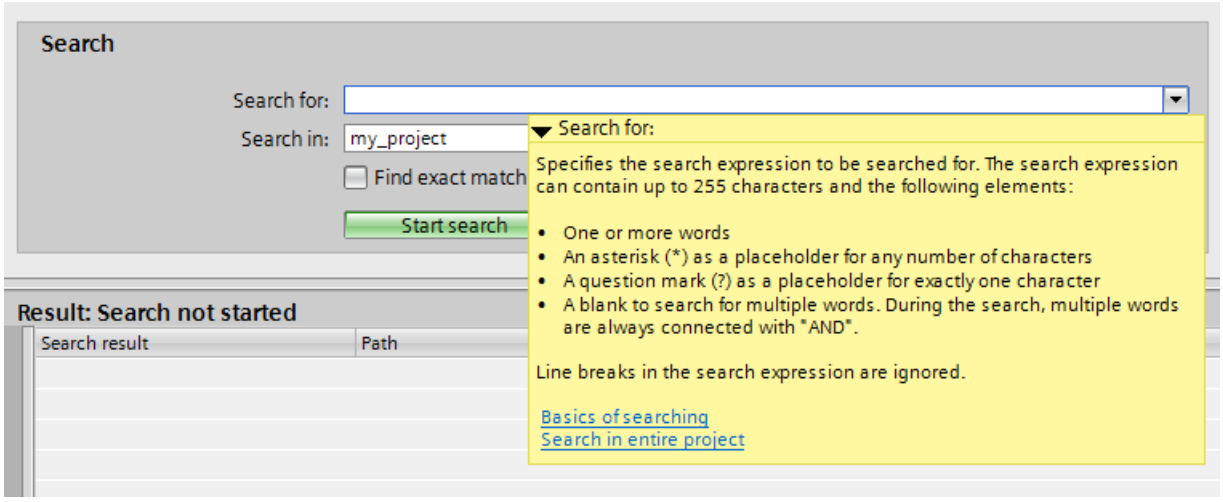

Help on messages

Numerous actions are accompanied by messages in the Inspector window of the TIA Portal. The messages give information about whether or not an action was successful. In addition, you see which changes have been made in the project. Further help is available for some messages. If further help is available for an message, you access the help by clicking the question mark symbol.

The following figure shows the "Info" tab in the Inspector window with several messages and a question mark symbol:

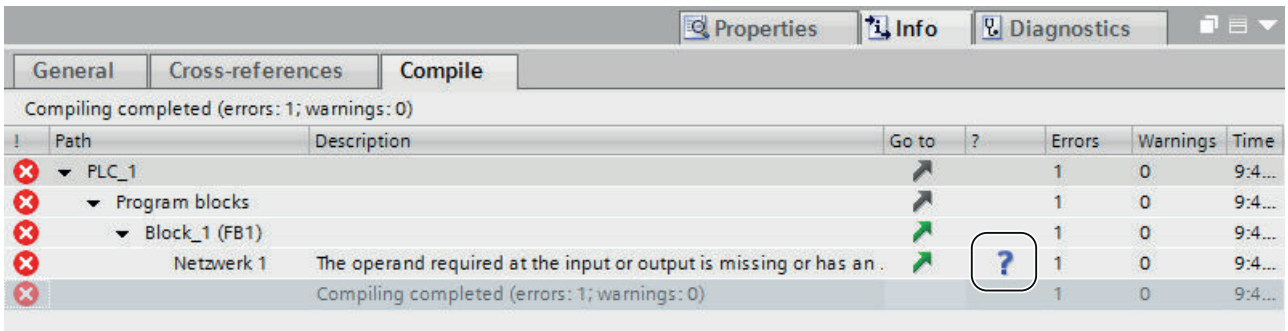

Banner on the software interface

In some editors, you can obtain additional support with banners at the top of the editor. The following figure shows a banner for an open library type:

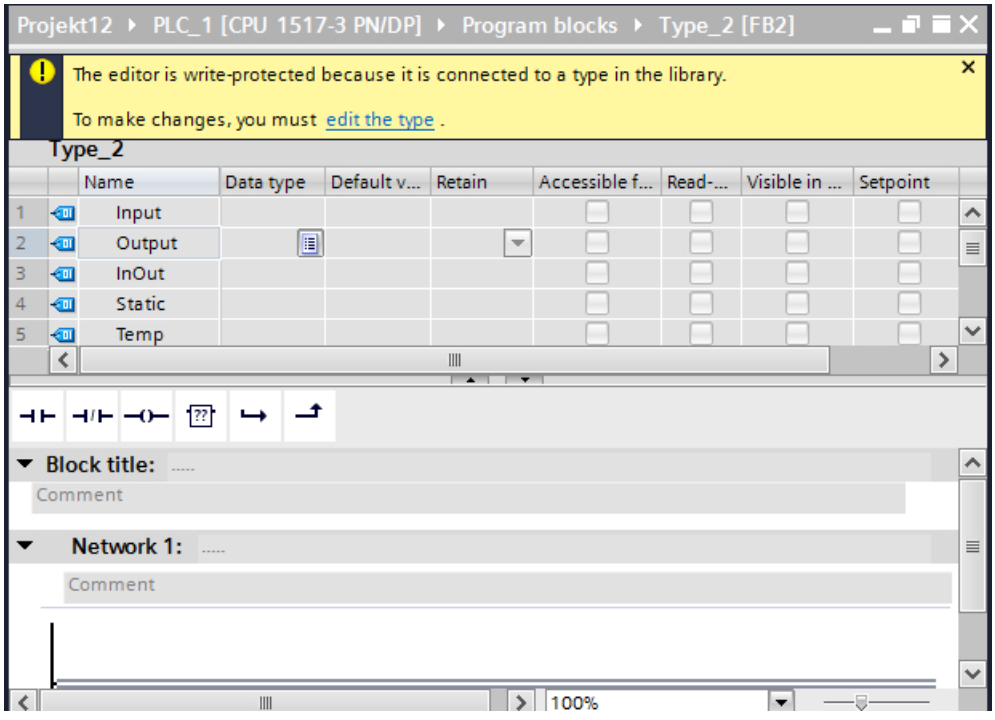

To hide the banner, click on the vertical blue bar at the left edge of the banner or on the x in the upper right corner. To expand a collapsed banner, click on the information symbol at the top left of the opened editor. In the settings of the TIA Portal, you can specify that banner is to be hidden by default.

If multiple banners are available in an editor, the most relevant banner is displayed first. To move to the next banner, scroll to the next banner at the bottom right corner.

See also

Configuring the display of tooltips and tooltip cascades (Page 97)

3.2 Opening the information system

Opening the information system with the menu

To access the information system on the home page, follow these steps:

1. Select "Display help" command from the "Help" menu.

The start page of the information system opens.

Opening the information system with <F1>

To access the information system and help for the current context, proceed as follows:

1. Select an object for which you want to view the help, for example, a menu command or a program element.
2. Press <F1>.

The information system opens. If information on the current context is available, the appropriate topic appears. If no information on the current context is available, the home page of the information system is displayed.

Opening the information system via tooltip cascades

To open the information system from a tooltip cascade, proceed as follows:

1. Move the mouse pointer over an object with a tooltip cascade.
   The tooltip cascade opens. If additional information is available, the tooltip cascade contains links to the corresponding help topics.
2. Click on a link.

The information system opens and the additional help topic appears.

Opening help for error messages

Numerous actions are accompanied by messages in the Inspector window of the TIA Portal. If help on such a message is available, a blue question mark appears behind the message.

To view the help for a message, follow these steps:

1. Display the Inspector window.
2. Click on the blue question mark behind a message.

The information system opens and the help error message appears. It contains an exact description of cause of the error and how to remedy it.

See also

Using the information system (Page 83)

3.3 Using the information system

Opening a help topic in the current tab

To open a help topic in the current tab, follow these steps:

1. Click on the help topic in the table of contents.

Or:

1. Double-click on the help topic in the list of search results.

Opening a help topic in a new tab

To open a help topic in a new tab, follow these steps:

1. Select the help topic in the table of contents or in the list of search results.
2. Select the "Open in new tab" command from the shortcut menu.

Or:

1. Click on the "New tab" button in the contents area of the information system.
   A new empty tab opens.
2. Click on the help topic in the table of contents or double-click on the help topic in the list of search results.

Closing a tab

To close a tab, follow these steps:

1. Click on the cross at the right in the tab.
   The current tab is closed.

Or:

1. Click on the tab.
2. Select the "Close other tabs" command from the shortcut menu.
   All tags except the current one will be closed.

Showing the history

The history of the previously opened help topics is stored individually for each tab. To remove a help topic from the history, follow these steps:

1. In the contents area of the information system, click the "Forward" or "Backward" buttons to scroll chronologically through the history.

Displaying the position of a help topic in the table of contents

To display the position of the open help topic in table of contents, follow these steps:

1. Position the mouse pointer on the help topic.
2. Click "Synchronize" in the contents area of the information system.

Displaying help for the information system

To display the help on the information system, follow these steps:

1. Click on the "Help" button in the contents area of the information system.

3.4 Searching the information system

3.4.1 Basics of searching

Introduction

You have the option to search the information system for terms. You are provided a search area for this.

Search options

You can search for the following text:

- Search for a word:
  `<Search term>`
  This finds all help pages that contain the search term exactly or in a slightly changed form.
- Search for several words:
  `<Search_term_1 Search_term_2 ... Search_term_n>`
  This finds all help pages that contain all search terms exactly or in a slightly changed form. Neither the order nor the distance between the search terms plays a role here.
- Search for phrases
  `<"Search_term_1 Search_term_2 ... Search_term_n">`
  This finds all help pages that contain all search terms in the exact order specified. To find a phrase, place your search expression in quotation marks.

You can further specify your search with additional criteria:

- Boolean search with "AND":
  `<Search_term AND Search_term>`
  `<Search_term + Search_term>`
  You can link your search terms using the Boolean "AND" operator. This finds all help pages that contain all search terms exactly or in a slightly changed form.
  Alternatively, you can enter multiple search terms even without the operators "AND" or "+". This too finds all help pages that contain all search terms.
- Boolean search with "OR":
  `<Search_term OR Search_term>`
  `<Search_term - Search_term>`
  You can link your search terms using the Boolean "OR" operator. This finds all help pages that contain one of the search terms exactly or in a slightly changed form.
- Negate search with "NOT"
  `<Search_term_1 NOT Search_term_2>`
  `<Search_term_1 ! Searchterm_2>`
  You can negate a search term with "NOT". You must enter at least one additional search term for this purpose, however. This finds all help pages in which "Search_term_1" occurs. Pages in which "Search_term_2" occurs are not included in the hits, since this search term has been negated.

- Fuzzy search with "~"
  `<Search~>`
  You can place a tilde ("~") after the search term. This finds all help pages in which the search term appears with approximately this spelling.
- Distance search with "~<proximity>"
  `<"Search_term_1 Search_term_2" ~proximity>`
  The proximity search allows you to find help pages where two search terms are no further apart than the defined distance. Enter the proximity as a positive integer. Enter the two search terms in quotes. There should be no gap between the tilde ("~") and the space.

Search filter

You can restrict the search by specifying search filters:

- If you select a device as a filter criterion, the keyword is only searched for in the pages or sections that apply to the specified device.
- If you select an area as a filter criterion, the search term is searched either only in the software documentation or only in the integrated hardware manuals.

The two search filters can also be combined.

Extended search

You can also use the extended search to search the information system. It offers the same search options as the default search. However, you also obtain direct information about the possible selections for search options.

See also

Searching help topics (Page 86)

Examples of searching in the information system (Page 87)

3.4.2 Searching help topics

To quickly find the required information, you can perform a full text search across all help topics.

Procedure

To search the information system for a specific terms, follow these steps:

1. Open the search area.
2. In the "Find what" box, enter the search term or select a recently searched phrase from the drop-down list.
3. Optional: To obtain information on the search options, click "...".
   The "Extended search" dialog opens. It shows an overview of all options.

4. Optional: In the "Devices" selection box, select a device as a search filter.
5. Optional: Select "Software documentation" or "Hardware manuals" in the "Area" selection field.
6. Click "Find".
   The search is performed and the results are listed:
   – The title of the help topic found is displayed in the "Title" column.
   – The "Position" column shows the section containing the help topic. The section from hardware manuals are marked with the prefix "HW".
   – The relevance of the topic to the search term appears in the "Rank" column.

   You can change the sorting by clicking on the header of a column.
7. Double-click on a search result or select the "Open in new tab" command in the shortcut menu.

Result

The help topic opens in the contents area.

To display the location of the open help topic in the table of contents, click on the "Synchronize" button.

See also

Basics of searching (Page 85)

Examples of searching in the information system (Page 87)

3.4.3 Examples of searching in the information system

Examples of the search for one word

Searching for a word finds all help pages that contain the search term exactly or in a slightly changed form. Terms that contain special characters must be enclosed in quotes:

- Search for `<PLC data type>`
  This finds all pages containing the words "PLC data type", "PLC data types" or similar related terms.

Examples if searching for several words

Searching for multiple words finds all help pages that contain all search terms exactly or in a slightly changed form. Neither the order nor the distance between the search terms plays a role.

- Search for `<variant example>`
  This finds all pages containing the terms "variant", "VARIANT", "variant instruction" and "example", "examples" or similar related terms.
- Search for `<LED tag>`
  This finds all pages containing the terms "LED tag", "LED-tag", "LED status of the tag" or similar related terms.

Examples of searching with phrases

Searching with phrases finds all help pages that contain all search terms in the exact order specified. To find a phrase, place your search expression in quotation marks.

- Search for `<"examples for using recipes">`
  This finds all pages containing the terms "Examples for using recipes" or similar terms in the specified order.

Examples of the Boolean search with "AND"

A Boolean search with "AND" finds all help pages that contain all search terms exactly or in a slightly changed form. Neither the order nor the distance between the search terms plays a role.

- Search for `<user groups AND authorization>`
  This finds all pages containing the terms "user groups", "user group" and "authorization", "authorizations" or similar related terms.
- Search for `<examples AND "Use in the recipes">`
  In this example, the Boolean search using "AND" is combined with the phrase search: This finds all pages containing exactly the terms "use in the recipes" in the specified order and "examples" at any position.

Examples of the Boolean search with "OR"

A Boolean search with "OR" finds all help pages that contain one or more of the specified search terms exactly or in a slightly changed form.

- Search for `<block OR function>`
  This finds all pages containing the terms "block", "blocks" and/or "function", "functions" or similar related terms.
- Search for `<block OR function OR function block>`
  This finds all pages containing the terms "block", "blocks" and/or "function", "functions" and/or "function block", "function blocks" or similar related terms.

Examples of the Boolean search with "NOT"

A Boolean search with "NOT" finds all help pages that contain the specified search terms but not the negated search terms.

- Search for `<alarm CPU NOT GRAPH>`
  This finds all pages containing the terms "alarms", "alarm" and "CPU" but not the word "GRAPH".
- Search for `<alarm NOT GRAPH CPU>`
  This finds all pages containing the terms "alarms", "alarm" and "CPU" but not the word "GRAPH".
- Search for `<alarm NOT CPU NOT GRAPH>`
  All pages containing the term "message" are found but not the word "GRAPH" and not the word "CPU".

Examples of the fuzzy search with "~"

A fuzzy search finds all help pages in which the search term appears with approximately this spelling.

- Search for `<trends~>`
  This finds all pages containing the terms "trend", "trend bit", "trend value", "trend display" "trend recording" or similar related terms.

Examples of a proximity search with "~ <distance>"

A proximity search finds all help pages where two search terms are no further apart from each other than the defined distance.

- Search for `<trend measurement~10>`
  This finds all pages in which the terms "trend" and "measurement" are no more than 10 words apart.

Examples of searching with filters

When searching with a filter, keywords are searched for only in the pages or sections that are valid for the specified device.

- Search for `<example>` with search filter "S7-1500"
  This finds all pages containing the terms "example", "examples" or similar related terms and that apply to S7-1500.
- Search for `<SCL>` with search filter "S7-400"
  This finds all pages containing the term "SCL" and that apply to S7-400.

3.5 Using favorites

Using favorites

You can save individual help pages or sections as favorites. This saves you searching for the help topic a second time.

Favorites are retained even if you close the TIA Portal. They are still available when the information system opens the next time.

Saving favorites

To save a page or section as a favorite, follow these steps:

1. Open the page or the section that you want to save as a favorite.
2. Click on the "Add to favorites" button.
   Or:

1. Select a page or section in the table of contents.
2. Select the "Add to favorites" command in the shortcut menu.

The help topic or chapter is saved as a favorite and is available the next time you open the help system.

Calling favorites

To open a page from your favorites, follow these steps:

1. Open the "Favorites" tab.
2. Double-click on the help topic that you want to open.
   Or:

1. Open the "Favorites" tab.
2. Select the shortcut menu command "Open in new tab".

The help topic opens in the contents area of the current tab or in a new tab.

Go to position in the table of contents

To navigate from your favorites to the position of the subject in the table of contents, follow these steps:

1. Select the topic in the favorites where you want to go in the table of contents.
2. Select the shortcut menu command "Go".
   The menu command is enabled only when you select a topic that is included in the table of contents. Context-sensitive help topics, for example, help on syntax errors, are not included in the table of contents. The "Go to" is not executable for this.

The table of contents opens and expands at the desired position.

Deleting favorites

To delete an entry from the favorites, proceed as follows:

1. Open the "Favorites" tab.
2. Select the topic you want to remove from the list.
3. Select the "Remove" command from the shortcut menu.

3.6 Copying code examples into the program

Many help topics include code examples that illustrate instructions and functions for programming in the TIA Portal. You can copy code examples in the programming languages SCL and STL and use them directly in your program.

Example

You recognize an example that can be copied by the "Copy" symbol in the header of the program example:

SCL

```
"Tag_Result1" := ABS("Tag_Value");
"Tag_Result2" := ABS("Tag_Value1"*"Tag_Value2");
```

Procedure

To copy a code example, follow these steps:

1. Open the Help topic with the sample program you require.
2. Click on the "Copy" button in the header of the program example.
   The program code is copied to the clipboard.
3. Open the programming editor and insert the copied code. To do this, select the "Paste" command in the shortcut menu.
   The copied program code is inserted from the clipboard into the program code being edited.

3.7 Displaying more information in the Siemens Industry Online Support

The information system provides an introduction to the "TIA Portal Information Center" in the Siemens Industry Online Support. Here you obtain additional up-to-date information about the TIA Portal, for example, application examples, training opportunities, FAQs, hardware documentation and much more.

Note

Internet connection required

To access the information in Siemens Industry Online Support, you need an Internet connection.

Procedure

To open more information in Siemens Industry Online Support, follow these steps:

1. Open the start page of the information system. You have the following options here:
   - Select the top entry in the table of contents. The entry has the title "Information system".
   - Select the "Home page" button in the contents area.

   The start page of the information system opens. The "TIA Portal Information Center" displayed at the bottom of the home page.
2. Click on the screen of the "TIA Portal Information Center".
   The "TIA Portal Information Center" opens.

3.8 Viewing hardware manuals

Introduction

You can configure a wide range of hardware using the TIA Portal. You can find the corresponding information in the "Hardware Manuals" section of the information system. You can use the manuals in two ways:

- Hardware manuals in the information system
  You have the option of displaying hardware manuals as an integrated part of the information system. This has the advantage of allowing you to search the integrated contents, filter, or set as a favorite. Integrated manuals can also be printed using the print function of the information system.
  In addition, you have the option to download additional hardware manuals, which are available as a support package.
- Hardware manuals in the Siemens Industry Online Support
  You can find all the current manuals, operating instructions and FAQs as well as updates for your devices in the Siemens Industry Online Support. To access the Online Support, you need an Internet connection.

Checking for updates for other hardware manuals

To check whether additional hardware manuals are available as an update and download them, follow these steps:

1. Open the table of contents of the information system.
2. Click the "Check for updates" button in the toolbar of the contents directory.
   All available support packages are displayed.
3. Check whether a support package for the hardware documentation is available.
4. Download the support package.
   See also: AUTOHOTSPOT
   The next time the information system starts, you will find the downloaded hardware manuals in the "Hardware manuals" section.

Search the information system for information from hardware manuals

To search specifically for information from integrated hardware manuals, follow these steps:

1. Open the search area.
2. Enter a search term in the "Search for" field.
3. Optional: Select a search filter from the "Devices" selection box to limit the search to a specific device or device series.
4. In the "Area" selection box, select the "Hardware manuals" search filter to restrict the search to integrated hardware manuals.

5. Click "Find".
   The search is performed and the results are listed.
   - The title of the help topic found is displayed in the "Title" column.
   - The "Position" column shows the section containing the help topic. The section from hardware manuals are marked with the prefix "HW".
   - The relevance of the topic to the search term appears in the "Rank" column.

   You can change the sorting by clicking on the header of a column.
6. Double-click on a search result or select the "Open" or "Open in new tab" command in the shortcut menu.
   The help topic opens in the contents area.

See also

Searching the information system (Page 85)

3.9 Printing help topics

You have the following options for printing the contents of the information system:

- You can print the selected sections or pages from the index together. You can select any content of the information system. It need not involve contents of contiguous sections. The selected content will be printed in a print-optimized layout with pagination.
- You can print a single open page.

Print sections or pages from the table of contents

To print the contents from the table of contents, follow these steps:

1. Open the table of contents of the information system.
2. Click on the "Print" button in the "Contents" tab.
   The "Print" dialog opens.
3. In the "Print" dialog, select the sections and pages that you want to print.
4. Click the "Print" button to print the selected information.
   The standard print dialog box opens.
5. Select the printer on which you want to print the information.
6. Click "Properties" if you want to make additional printer settings.
7. Confirm your entries with "OK".
   The information is printed on the selected printer.

Printing single opened page

To print a single page, follow these steps:

1. Open the page to be printed in the contents area of the information system.
2. Select the "Print" button in the contents area.
3. The standard print dialog box opens.
4. Select the printer on which you want to print the information.
5. Click "Properties" if you want to make additional printer settings.
6. Confirm your entries with "OK".
   The page is printed on the selected printer.

3.10 Configuring the display of tooltips and tooltip cascades

Configuration options for tooltips and tooltip cascade

You can customize the display of tooltips and tooltip cascades to suit your needs. You can make the following settings:

- Display or hide truncated texts
  Sometimes texts may be too long for a text field. The texts are then fully displayed in a tooltip when you hover your mouse over the text field. You can enable or disable this function.
- Enable or disable tooltips
  Tooltips provide more detailed information about an element of the user interface. You can also have tooltips displayed in a cascade. If you disable the tooltips, the cascade with context-sensitive help is also no longer displayed. However, you have the option of manually displaying the tooltip for the currently active interface element by pressing <F1>. The setting also affects rollouts with context-related syntax errors and input aids in STL.
- Enable or disable automatic opening of tooltip cascades
  By keeping the mouse pointer over a tooltip for a brief time, any available cascades are displayed automatically. You can enable or disable the automatic display of cascades. When automatic display is disabled, you must open the cascade manually if necessary. To do this, click on the arrow icon in the tooltip.

Procedure

To configure the display of tooltips and tooltip cascades, follow these steps:

1. Select the "Settings" command in the "Options" menu.
2. Select the "General" group in the area navigation.
3. Select or clear the individual check boxes in the "Tooltips" area to suit your needs. The "Open cascade automatically in tooltips" check box is only available if you have enabled the display of tooltips.

See also

General remarks on the information system (Page 75)

3.11 Assembling customized documentation

Customized documentation

In the Siemens Industry Online Support, in the "mySupport Documentation" category, you can assemble customized documentation that is tailored to your needs. All configurable manuals and operating instructions of the Siemens Industry Online Support are available to you for this purpose. You can select the parts that are of interest to you and combine them into in a library to form personal documentation. You can organize the documentation using folders in your personal library. The folders will later become the individual chapters of your custom documentation.

You can open "mySupport Documentation" here:

https://support.industry.siemens.com/My/ww/en/documentation (https://support.industry.siemens.com/My/ww/en/documentation)

Requirement

- The manuals or operating instructions used must be configurable. You can recognize configurable manuals by the suffix "configurable" in their name.
- To use all the functions, you have to register in the Siemens Industry Online Support and log on.

Documentation in different languages

You can change the language of the assembled documentation to German, French, Spanish, Italian and Chinese. This gives you the possibility, for example, to gather relevant information for a specific project and make it available to colleagues who speak other languages.

Export function in the documentation

You can perform an export at any part of your library in various formats (PDF, XML, RTF).

Help on creating documentation

You can find more help on creating and using custom documentation in the Siemens Industry Online Support:

https://support.industry.siemens.com/My/ww/en/documentation (https://support.industry.siemens.com/cs/helpcenter/en/index.htm)

3.12 Safety Guidelines

Safety guidelines

This Help manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.

⚠ CAUTION
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

Note

indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by qualified personnel. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Prescribed Usage

Note the following:

⚠ WARNING
This device may only be used for the applications described in the catalog or the technical description and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens. Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

Trademarks

All names identified by ® are registered trademarks of the Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Find and replace 4

4.1 Basics of searching

Introduction

You can use the following search options within TIA Portal:

- Search entire project
- Searching and replacing in the editor
- Browsing the hardware catalog

In addition, individual products in TIA Portal offer more search options.

Search entire project

You can search the entire project for a specific text. The search editor helps you in the search, for example, by further limiting the search. All objects containing the search text are displayed clearly in a table. You can open each object in the search editor to view the relevant location.

Searching and replacing in the editor

You can search for texts in the editors. The search function finds all texts containing the search key in the currently opened editor. The results are selected in sequence in the opened editor.

You also have the following options:

- Narrowing down the search with additional options
- Replacing found texts

Which additional options are available and which types of text can be searched for depends on the installed products and the open editor.

Browsing the hardware catalog

You can search for a specific hardware component in the hardware catalog.

See also: Browsing the hardware catalog

See also

Searching and replacing in the editor (Page 113)

Search in entire project (Page 105)

4.2 Search entire project

4.2.1 Basics for search in the project

Introduction

You can search the open project for a specific text (search term). The search term can be one word, several words or parts of words. Only spaces are considered as word limits. You can use search terms that are up to 255 characters long. You can create any complex search terms using wildcards. You can also limit the search to specific areas, objects or object properties. Uppercase and lowercase letters are not taken into account in the search.

For the search, an index of the texts of the editors which can be searched in the project is built up automatically in the background. The search index is updated with each change. You cannot start the search before indexing is complete. You can generate the search index again at any time.

Note that the search index is saved with the project. This may increase the file size of the project. If you want to pass on the project, you can reduce the file size by archiving or minimizing the project. The search index is removed in this case. You can also completely disable the search in the project if necessary.

In the case of changes to STEP 7 objects, e.g. name changes of tags, PLC data types or blocks, you must update the point of use of the these object before any search in the project in order that the search can take these changes into account. To do this you can, for example, recompile your program.

The search only includes the texts in the set editing language. The search index is generated again when you change the editing language. This may take some time, depending on the number of texts in the project.

The search editor helps you with your search. In it, you can specify search options and start the search. After the search is completed, the search editor lists the objects which contain the search text. You can open the individual objects directly from the search editor. If the objects have viewable properties, you can display them in the Inspector window.

Search options

The following options are available for the search:

- Limit search:
  - Search in: You can select the sections of the project that are to be searched.
  - Find exact match: Finds exactly the text you have entered. The search term is searched as individual word or as phrase. You cannot use wildcards in the search term when you use this option, for example, when you enter "*" or "?" the search is exactly for these characters.
  - Limit search: You can limit the search to specific object properties or object types.
- Use "*" as wildcard for any number of characters and "?" as wildcard for exactly one character within a word:
  When you use the "*" or "?" wildcards in a search term, you will receive whole words that match this search term as hits. When you enter "Motor*start", for example, you will receive hits such as "Motor starter" or "Motor1_Start" but not "Starting motors" or "Start motor1". An implicit wildcard search is always conducted. If you enter "Motor" as search term, for example, the search also finds the texts "Motor_ON", "Motor starter" and "123_Motor". This corresponds to the explicit wildcard search with the search term "*Motor*". You can deactivate the implicit wildcard search by enabling the option "Find exact match".

Replace texts in the project

You have the option to replace individual texts in the project. To do this, you can open the location immediately after a search by double-clicking on the hit list and editing the text. Once it is changed, the text is automatically indexed again. If you then want to open the next match, you receive a message that offers you the chance to refresh the hit list. In this way, you can replace a text in the entire project until the hit list no longer shows results.

See also

Basics of searching (Page 101)

Overview of the search editor (Page 103)

Search in entire project (Page 105)

Creating a new search index manually (Page 106)

Enabling and disabling "Search in project" (Page 107)

Examples for searching in the project (Page 108)

4.2.2 Overview of the search editor

Function of the search editor

The search editor searches the entire project for specific texts and represents the search results clearly in a table.

Layout of the search editor

The figure below shows the components of the search editor:

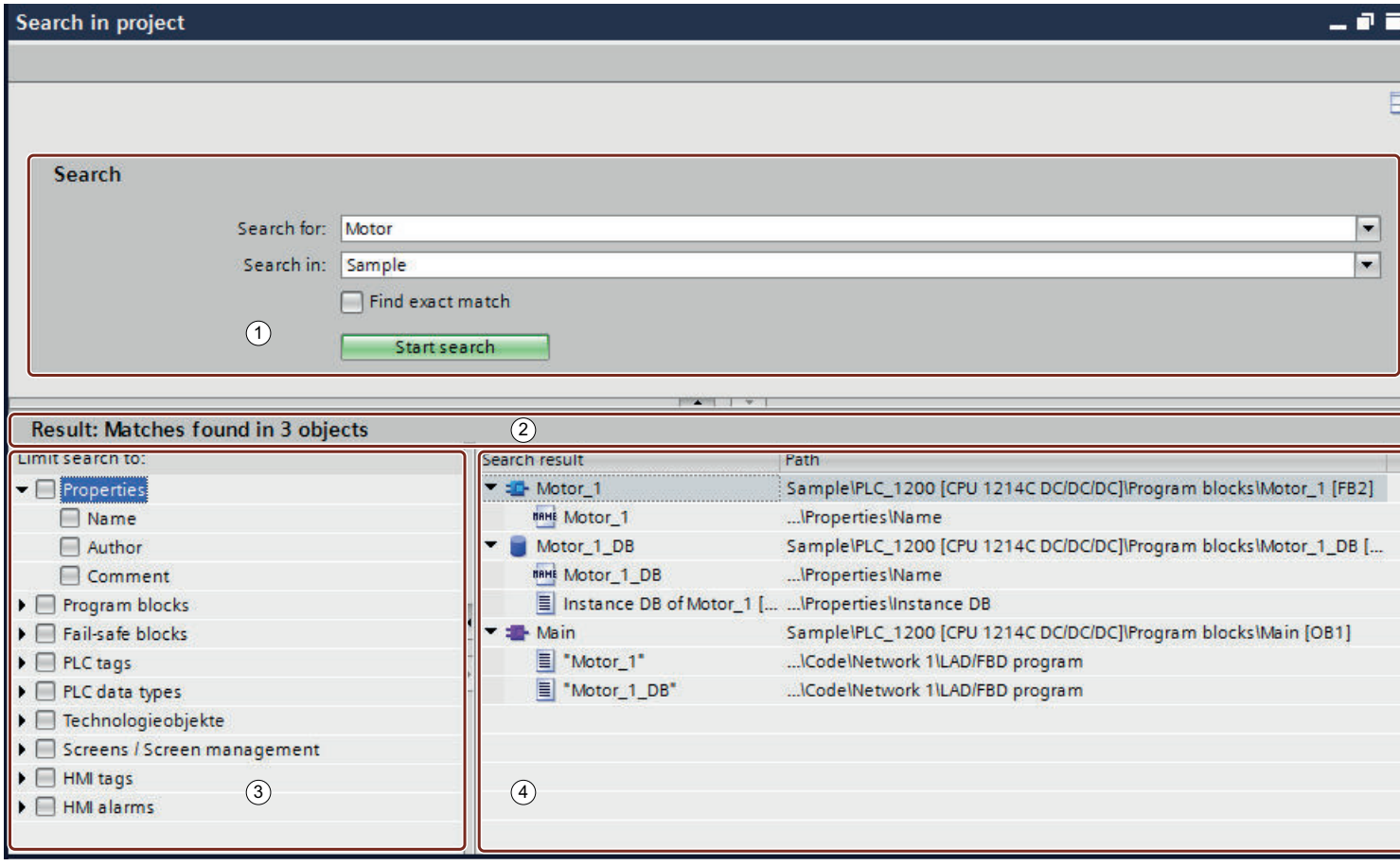

① Search mask

② Status bar

③ Search filter

④ Results list

Search mask

You enter the search term in the search mask and limit the search area. You also have the option to select "Find exact match" which means the search is limited exactly to the entered search term in the project and similar text passages are ignored. You start the search in the search mask.

Status bar

The status bar includes information about the search result.

Search filter

You can use the search filter to further limit the search results. If you have not activated any filters, the search includes all objects and object properties. If you activate a filter, only the search results which match the filter setting are displayed.

Results list

The results list shows all objects which include the search term. Double-click on a search result to open the corresponding editor and highlight the reference.

See also

Basics of searching (Page 101)

Basics for search in the project (Page 102)

Search in entire project (Page 105)

Creating a new search index manually (Page 106)

Enabling and disabling "Search in project" (Page 107)

Examples for searching in the project (Page 108)

4.2.3 Search in entire project

Requirement

A project is open.

Procedure

To search the current project for a specific text, follow these steps:

1. Enter the search term in the text box for the search in the TIA Portal toolbar.
2. Click "Search in project".
   The search editor opens and a search for the search term is conducted across the entire project. The objects containing the search text are listed in the overview.

   Note

   You cannot enter any search options with this search. The entire project is searched and all objects that can be searched are included in the search. If you want to further limit the search or use search options, you can start the search once again. To do so, follow the alternative instructions under "Or" as of step 2.

Or:

1. Open the search editor in one of two ways:
   - Select the "Search in project" command in the "Edit" menu.
   - Right-click an object in the project tree and select the "Search in project" command from the shortcut menu. The object is set as default search area in the "Search in" box.
   - Make sure that the focus is outside of an editor that supports an editor-specific search and press the shortcut <Ctrl+F>.
2. Enter the search term in the "Search" box, or select a term from the list of terms for which you have already searched.
3. To search exactly for the search term, select the check box "Find exact match". You cannot use wildcards in the search term when you use this option, for example, when you enter "*" or "?" the search is exactly for these characters.
4. To limit the search area, select the sections you want to include in the search under "Search in".
5. To limit the search to specific objects or object properties, select the objects or object properties you want to include in the search under "Limit search to".
6. Click "Start Search".
   The objects containing the search text are listed in the overview.
7. Double-click an object to open it.

See also

Basics of searching (Page 101)

Basics for search in the project (Page 102)

Overview of the search editor (Page 103)

Creating a new search index manually (Page 106)

Enabling and disabling "Search in project" (Page 107)

Examples for searching in the project (Page 108)

Searching and replacing in the editor (Page 113)

4.2.4 Creating a new search index manually

For the search, an index of the texts of the editors which can be searched in the project is built up automatically in the background. The search index is updated with each change. You cannot start the search before indexing is complete. You can also generate the search index manually at any time.

Procedure

To automatically generate the search index, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the group "General > Search in project".
3. Click "Recreate search index".

Result

The indexes of the project are generated again.

See also

Basics of searching (Page 101)

Basics for search in the project (Page 102)

Overview of the search editor (Page 103)

Search in entire project (Page 105)

Enabling and disabling "Search in project" (Page 107)

Examples for searching in the project (Page 108)

4.2.5 Enabling and disabling "Search in project"

You can disable "Search in project" centrally, if you do not need it. If necessary, you can enable again it at any time. By default, "Search in project" is enabled.

Procedure

To disable or enable "Search in project", follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the group "General > Search in project".
3. Select or clear the "Search in project" check box.

See also

Basics of searching (Page 101)

Basics for search in the project (Page 102)

Overview of the search editor (Page 103)

Search in entire project (Page 105)

Creating a new search index manually (Page 106)

Examples for searching in the project (Page 108)

4.2.6 Examples for searching in the project

Examples for use of property and object filters

The search editor offers filters in the "Limit search to" area which you can use to limit the search to specific properties or objects. Keep in mind the following mechanism:

- If you do not activate any filters, the search term is searched in all objects and properties that can be searched within the set search area.
- If you activate filters, only the selected properties or objects are searched.

Examples for use of property and object filters:

- Search for an object with the name "Motor1":
  Enter the search term and activate the filter "Name". All objects which include "Motor1" in their name are found.
- Search for an object which includes the text "Motor1" either in its name or in a comment:
  Enter the search term and activate the filters "Name" and "Comment". All objects which include "Motor1" in their name or in the comment are found.
- Search for a block with the name "Motor1":
  Enter the search term and activate the filters "Name" and "Block". All blocks which include "Motor1" in their name are found.
- Search for a block which includes the text "Motor1" either in its name or in a comment:
  Enter the search term and activate the filters "Name", "Comment" and "Block". All blocks which include "Motor1" in their name or in the comment are found.
- Search for a block or a technology object with the name "Motor1":
  Enter the search term and activate the filters "Name", "Block" and "Technology object". All blocks or technology objects which include "Motor1" in their name are found.

When entering the search term, note the following examples for searching for a term or several terms.

Examples for the search for one term

When your search entry consists of only one term and the option "Find exact match" is disabled, the search is always an implicit wildcard search. The system adds the wildcard "*" in front of and behind the search term. You can insert wildcards in the search term yourself to conduct an explicit wildcard search. If you want to find exactly the term you have entered, select the option "Find exact match". Uppercase and lowercase letters are not taken into account in the search.

Examples for the search for one term:

- Search for "motor", "Find exact match" option is disabled
  An implicit wildcard search is conducted. The search term is interpreted as "*motor*" and all terms which include "motor" are found, for example, "motor", "motor_1", "Motorstart" or "Start_Motor".
- Search for "m*tor" (explicit wild card search), "Find exact match" option is disabled
  The search term is interpreted as "*m*tor*" in this wildcard search. This means terms, such as "motor", "motor_1", "Start_Motor", "GLOB.Mode.motor[1]" or "Mo_tor" are also found.
- Search for "motor", "Find exact match" option is enabled
  No wildcards are used for the search, which means only the terms "motor" and "Motor" are found.
- Search for "m*tor" (explicit wild card search), "Find exact match" option is enabled
  The character "*" is no longer a wildcard and is interpreted as a regular character. The search only finds the character strings "m*tor" and "M*tor" but not, for example, the terms "motor", "", "Motormotor_1".

Application examples:

- If you want to determine from where the "GLOB_DB.MODE.FG[6].IN.ERROR_DROP_START" element of a data block is accessed, you can use the cross-reference list. If access takes place by indirect addressing (e.g. with "#GLOB.MODE.FG[#FgNr].IN.ERROR_DROP_START := TRUE;"), however, the cross-reference list does not return any results. Use a wildcard search for "MODE.FG[*].IN.ERROR_DROP_START" across the entire project in this case.
- You want to find all instances where "ERROR_DROP_START" is used. Perform a search for "ERROR_DROP_START" across the entire project. The search also finds all "ERROR_DROP_START" elements within the "FG[]" array.

Examples for the search for multiple terms

When you enter a search term which consists of multiple terms and the "Find exact match" option is disabled, you will find those objects which include all these terms. The texts can appear in any order. If you want the terms to appear exactly as you have entered them in an object, select the "Find exact match" option. Uppercase and lowercase letters are not taken into account in the search.

Examples for the search for multiple terms:

- Search for "motor start", "Find exact match" option is disabled
  All objects which include "motor" and "start" are found. It does not matter how far apart they are from each other or in which order they appear. Possible hits therefore include: "Motor start", "start my motor", "my motor is in start mode"

  Note

  The texts of a column are treated as a text object in data blocks and in block interfaces. Therefore, the search also returns data blocks and blocks in which the terms occur distributed over different lines.

- Search for "motor start*", "Find exact match" option is disabled
  As with the search for a word, an implicit wildcard search is always conducted for all words with a multiple word search. This means it is not necessary to explicitly enter a wildcard character. "motor start*" is interpreted as "*motor* *start*" and the same hits are found as with "motor start".
- Search for "motor start", "Find exact match" option is enabled
  When the option "Find exact match" is enabled, only objects which include both terms one after the other as a phrase are found. The search finds the search term, for example, in objects which include the texts "push motor start button" or "motor start on". No hits are returned for "start motor", "motorstart", "start my motor", "my motor is in start mode".
- Search for "motor start*", "Find exact match" option is enabled
  The character "*" is no longer a wildcard and is interpreted as a regular character. The search only finds objects in which the character string "motor start*" appears in uppercase or lowercase letters. "motor starting", "start my motor", "motor start", "Motor Start", "motor start_mode" do not result in any hits.

Application examples:

- You want to find out where you have used the function "Setbit" to set the "MotorOn" tag. To do so, conduct a search for "setbit motoron".
- You want to find a specific network comment but you only know that it includes the words "CPU" and "Cycle". To do so, start a search for "CPU cycle".

See also

Basics of searching (Page 101)

Basics for search in the project (Page 102)

Overview of the search editor (Page 103)

Search in entire project (Page 105)

Creating a new search index manually (Page 106)

Enabling and disabling "Search in project" (Page 107)

4.3 Searching and replacing in the editor

4.3.1 Basics for search in open editors

Introduction

You can start a search which is limited to the editor currently open in the workspace. You can customize the search as follows to suit requirements:

- You can optimize the search with additional options.
- You can specify the search area.
- You can specify the search direction.

Additional options for searching

You can refine the search with the help of the following additional options:

- Whole words only
  Only whole words are taken into account for the search. Compound words that contain the search key as a part are ignored.
- Match case
  Upper- and lowercase letters are taken into account in the search.
- Find in substructures
  The search also includes texts contained in another object.
- Find in hidden texts
  Texts that are assigned to another text but that are currently hidden are also included in the search.
- Use wildcards
  Enter an asterisk as the wildcard for any number of characters. Example: You want to search for all words starting with "Device". Type in "Device*" in the search key box.
  Enter a question mark as the wildcard, however, if you only want to exclude a single character.

The additional options available depend on the installed products and the editor opened.

Search area and search direction

You have the following options for define the search area:

- Whole document
  The entire is searched regardless of the current selection and position.
- From current position
  The search starts at the current position. Depending on the search direction, all texts above or to the left or below or to the right of the current position are not included in the search.
- Selection
  The search term is only searched for in the current selection.

You can define the search direction independent of the search area:

- Down
  Search takes place in editor from top to bottom or from left to right.
- Up
  Search takes place in editor from bottom to top or from right to left.

4.3.2 Overview of the "Find and replace" palette

Function of the "Find and replace" palette

The "Find and replace" palette in the "Tasks" task card lets you search in an open editor. It includes all options that you need for an efficient search.

Layout of the "Find and replace" palette

The following figure shows the components of the "Find and replace" palette:

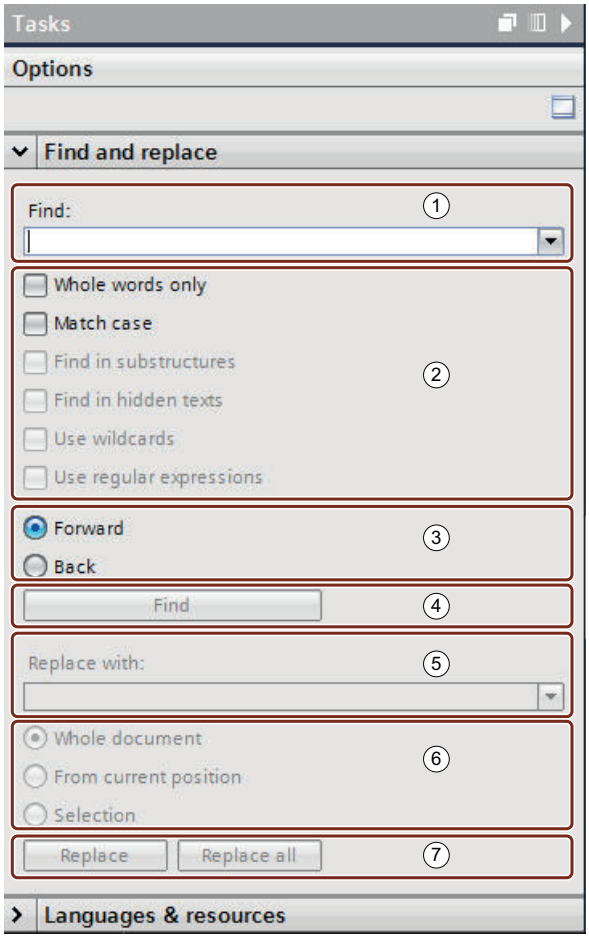

① Enter search terms

② Additional options

③ Search direction

④ Start search

⑤ Enter replacement term

⑥ Replace area

⑦ Replace

4.3.3 Searching and replacing in the editor

You have the option to search for text in an editor or to replace it.

Start search

Follow these steps to start the "Find and replace" function:

1. Select the "Find and replace" command in the "Edit" menu or open the "Find and replace" pane in the "Tasks" task card.
   The "Find and replace" pane opens.
2. Enter a term in the "Find" drop-down list.
   As an alternative, you can select the most recent search key from the drop-down list.
3. Select the options desired for the search.
4. Using the option buttons, select the starting point for the search and the search direction.
5. Click "Find".
   The first hit is marked in the editor.
6. Click "Find" again to display the next hit.
   The next hit is marked in the editor. Repeat this step as needed until you reach the last hit.

Replacing the search key

You have the option of replacing hits individually or automatically replacing all the found texts, if the respective editor supports this function. Follow these steps to replace terms:

1. Enter a term in the "Find" drop-down list.
   As an alternative, you can select the most recent search key from the drop-down list.
2. Select the options desired for the search.
3. Click the "Find" button to start a search for the specified search key.
   The first hit is displayed in the editor.
4. In the "Replace" drop-down list, enter the text you wish to use to replace the search key.
   As an alternative, you can select one of the last entered texts from the drop-down list.

5. Click the "Replace" button to replace the selected hit with the specified text.
   The found text is replaced and the next hit is marked in the editor.
   Repeat this procedure until you have replaced all the desired hits. To skip to the next hit without replacing the marked word, click the "Find" button instead of "Replace".
6. Click "Replace all" to automatically replace all hits at once.

See also

Basics of searching (Page 101)

Search in entire project (Page 105)

Undoing and redoing actions 5

5.1 Basics of undoing and redoing actions

Function

You can undo performed actions at any time. For this purpose, every action you perform is saved in an action stack. When undoing actions, the stack is processed from top to bottom. In other words, if you undo an action that lies further down in the stack, all actions located above it in the stack will also be undone automatically.

You can redo previously undone actions until you execute a new action. Once you execute a new action, it is no longer possible to redo previously undone actions.

Particularities for undoing

There are a few actions that empty the action stack. You cannot undo these actions or the actions performed before these actions. The following actions empty the action stack:

- Saving
- Project management (creating a new project, opening project, closing a project, deleting a project)
- Deleting unused GSD files
- Establishing an online connection

Keep in mind that the action stack is emptied every time you establish an online connection. This means that the actions executed offline cannot be undone once the online connection is established. You can undo all actions which you subsequently execute online, even after the online connection has been terminated.

Displaying the action stack

The "Undo" button in the toolbar is enabled as soon as you perform an action that can be undone. This button is split; you can use the arrow down portion to open a drop-down list. containing all actions of the action stack that you can undo. If you had performed actions in an editor other than the currently displayed editor, the corresponding editor is also displayed as a subheading. This allows you to always identify the point at which the undo operation will be applied. The subheadings are removed from the list when the editor responsible can no longer undo actions.

Actions you have undone are entered in the action stack from where they can be redone. Here, you can redo actions you have undone. The display of actions you can redo it is analogous to the display of the actions that you can undo.

Example of undoing actions

The figure below shows how actions performed in various editors and tables are undone:

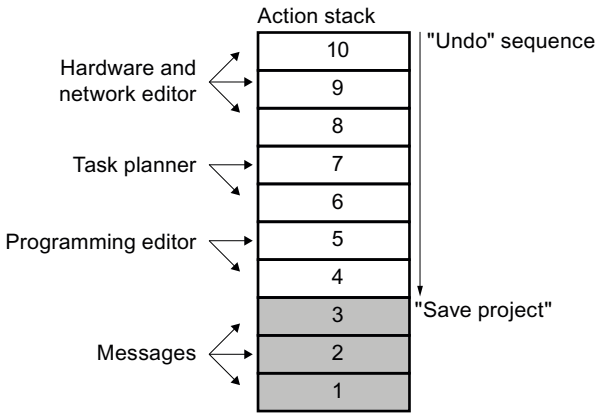

In this example, you cannot undo actions 1 to 3 because the project was saved. You can undo actions 4 to 10 in the order indicated by the direction of the arrow. In other words, you must undo action 10 first. Once you have undone action 8, you cannot then undo action 5. You must first undo actions 7 and 6. As the final step in the sequence, you can then undo action 4. You also have the option of undoing several actions in a single step by undoing an action located further down in the action stack. All actions located above it in the stack will be undone automatically.

The same principle also applies to redoing of actions.

See also

Undoing an action (Page 117)

Redoing an action (Page 118)

5.2 Undoing an action

The following options are available for undoing actions:

- Undoing the last action only
  Only the last action performed is undone.
- Undoing as many actions as required
  Multiple actions in the action stack are undone in a single step.

Undoing the last action only

To undo the last action performed, follow the steps below:

1. Click the "Undo" button in the toolbar.
   - If the action was not performed in the currently displayed editor, a confirmation prompt appears.
   - If the undo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
2. Click "Yes" to confirm.
3. Enter the password, if necessary.
   The editor in which the action was performed is displayed and the action is undone.

Undoing as many actions as required

To undo multiple actions in the action stack in a single step, follow these steps:

1. Click the Down arrow next to the "Undo" button in the toolbar.
   This opens a drop-down list containing all actions you can undo. Actions performed in other editors are identified by the editor name in the subheading.
2. Click the action you want to undo.
   The chosen action and all actions in the stack located above the chosen action are undone. If the undo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
3. Enter the required passwords, if necessary.
   The editors in which the actions were performed are displayed and the actions are undone.

See also

Basics of undoing and redoing actions (Page 115)

Redoing an action (Page 118)

5.3 Redoing an action

You have the option of redoing an action that has been undone, so that you can return to the status present before "Undo" was performed. However, this is only possible until you perform a new action. The following options are available for redoing actions:

- Redoing the last undone action only
  Only the last undone action is redone.
- Redoing as many undone actions as required
  Multiple undone actions in the action stack are redone in a single step.

Redoing the last undone action only

To redo the last undone action, follow the steps below:

1. Click the "Redo" button in the toolbar.
   – If the action is not being redone in the currently displayed editor, a confirmation prompt appears.
   – If the redo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
2. Click "Yes" to confirm.
3. Enter the password, if necessary.
   The editor in which the action was undone is displayed and the action is redone.

Redoing as many undone actions as required

To redo multiple undone actions in the action stack in a single step, follow these steps:

1. Click the Down arrow next to the "Redo" button in the toolbar.
   This opens a drop-down list containing all actions you can redo. Actions performed in other editors are identified by the editor name in the subheading.
2. Click the action you want to redo.
   The chosen action and all actions in the stack located above the chosen action are redone. If the redo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
3. Enter the required passwords, if necessary.
   The editors in which the actions were undone are displayed and the actions are redone.

See also

Basics of undoing and redoing actions (Page 115)

Undoing an action (Page 117)

Index

A

Actions
- Basics of redoing actions, 115
- Basics of undoing, 115
- Redoing, 118
- Undoing, 117

Assembling manuals, 98

B

Backing up the table layout, 49
Backing up the user interface, 49
- Backing up the layout in editors, 49

C

Creating custom documentation, 98

D

Details view, 43

H

Help
- Browsing, 86
- Call topic from favorites, 90
- Components of the information system, 75
- Delete topic from favorites, 91
- Full-text search, 86
- Help on messages, 80
- Identification of help topics, 78
- Navigation view, 83
- Open tooltip cascades automatically, 97
- Opening, 82, 83
- Printing help topics, 96
- Roll-out, 79
- Save topics in favorites, 90
- Tooltip, 79

I

Information system
- Components of the information system, 75
- Help on messages, 80
- Roll-out, 79
- Tooltip, 79

Inspector window
- Layout, 38
- Reducing automatically, 32

K

Keyboard operation
- Basic functions of the TIA Portal, 57
- Customizing editors, 62
- Editing objects, 64
- Editing texts, 64
- Navigation in the TIA Portal, 61
- Online functions, 67
- Project editing, 59
- Selecting objects, 63
- Tables, 65
- Window, 60

L

Load window layout
- Loading additional window layouts, 50
- Loading via quick access, 50

M

MDM, 98
My Documentation Manager, 98

O

Obtaining user rights
- Logging on to the operating system with administrator privileges, 73
- With Windows user account control, 73

Overview window, 45
- Comparing objects, 47
- Display all blocks, 48
- Showing and hiding columns, 48
- Sorting details view, 48

P

Pane mode, 42

Portal view, 21
Program editor
General settings, 10, 13, 14, 16
Project tree
Function, 26
Reducing automatically, 32
Showing additional columns, 30
Project view, 23

R

Redoing actions
Basics of redoing actions, 115
Redoing actions, 118
Resetting the user interface layout, 52
Restore window layout, 50
Restrictions due to user rights, 72
Recognizing, 72

S

Safety guidelines, 99
Save window layout, 49
Saving the user interface
Saving the window layout, 49
Search
Additional options for searching in the editor, 111
Basics of searching, 101
Basics of searching in the project, 102
Examples for searching in the project, 108
Find and replace in the editor, 113
Find and replace palette, 112
Replacing search terms in the editor, 113
Search editor, 103
Search in entire project, 105
Settings
Changing, 17
General, 10, 12, 13, 14, 15, 16
Online and diagnostic functions, 16

T

Task card
Changing the pane mode, 42
Function, 40
Reducing automatically, 32
TIA Portal
Exiting, 9
Starting, 9

U

Undoing actions
Basics of undoing, 115
Undoing last action, 117
Undoing multiple actions, 117
User interface
"Reference projects" palette, 42
Details view, 43
Inspector window, 38
Maximizing the work area, 32
Minimizing the work area, 32
Overview window, 45
Portal view, 21
Project tree, 26
Project view, 23
Task card, 40
Views, 21
Work area, 30
Using the on-screen keyboard, 67

W

Window layouts
Changing order, 51
Deleting window layouts, 51
Work area
Embedding floating elements, 34
Floating elements, 34
Function, 30
Maximizing, 32
Maximizing elements, 37
Minimizing, 32
Minimizing elements, 37
Saving a layout of editors and tables, 51
Splitting, 33
Switching between elements, 37
Using grouped elements, 35

SIEMENS

Creating and managing projects 1

Compatibility of projects 2

Archiving and retrieving projects 3

Using reference projects 4

TIA Portal Teamcenter Gateway 5

Editing projects

Function Manual

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Creating and managing projects....7
1.1 The basics of projects....7
1.2 Creating a project....9
1.3 Opening projects....10
1.4 Saving projects....12
1.5 Closing projects....13
1.6 Removing projects from the project list....14
1.7 Deleting projects....15
1.8 Displaying properties of the project....16
1.9 Using logs....17
1.10 Security of industrial plants....18
1.11 Importing and exporting CAx data....20
2 Compatibility of projects....23
2.1 Compatibility between TIA Portal versions....23
2.2 Using projects created with optional software....24
2.3 Upgrading projects....25
2.3.1 General information on upgrading projects....25
2.3.2 Upgrading projects....26
2.3.3 Upgrading multiuser engineering projects....28
2.3.4 Upgrading know-how protected blocks....28
2.4 Compatibility with the plant configuration....30
3 Archiving and retrieving projects....33
3.1 Working with project archives....33
3.2 Creating a project archive....34
3.3 Retrieving compressed project....35
4 Using reference projects....37
4.1 Basics of reference projects....37
4.2 Opening and closing a reference project....38
4.3 Comparing reference projects....39
5 TIA Portal Teamcenter Gateway....41
5.1 Introduction to TIA Portal Teamcenter Gateway....41
5.1.1 Basic information on working with TIA Portal Teamcenter Gateway....41
5.1.2 Application possibilities of TIA Portal Teamcenter Gateway....42

5.2 System requirements....43
5.2.1 System requirements for TIA Portal Teamcenter Gateway....43

5.3 Installing and uninstalling TIA Portal Teamcenter Gateway....45
5.3.1 Notes on installing TIA Portal Teamcenter Gateway....45
5.3.2 Notes on compatibility of TIA Portal Teamcenter Gateway....46
5.3.3 Parallel installation....46
5.3.4 Starting the installation of TIA Portal Teamcenter Gateway....46
5.3.5 Displaying installed TIA Portal Teamcenter Gateway software....47
5.3.6 Checking the installation of TIA Portal Teamcenter Gateway....48
5.3.7 Modifying or updating TIA Portal Teamcenter Gateway....49
5.3.8 Repairing the installation of TIA Portal Teamcenter Gateway....50
5.3.9 Starting the uninstallation of TIA Portal Teamcenter Gateway....51

5.4 Licenses for TIA Portal Teamcenter Gateway....53
5.4.1 Licenses....53
5.4.2 Notes on licenses....53

5.5 Supported Teamcenter server versions and Teamcenter client versions....55
5.5.1 Supported Teamcenter server versions and Teamcenter client versions....55

5.6 Basics of working with TIA Portal Teamcenter Gateway....56

5.7 Setting the TIA Portal Teamcenter Gateway cache path....57

5.8 Confirming check-in when closing the project/library....58

5.9 Establishing and terminating a connection between the TIA Portal and Teamcenter....59
5.9.1 Establishing a connection between the TIA Portal and Teamcenter....59
5.9.2 Terminating the connection between the TIA Portal and Teamcenter....61
5.9.3 Display connection status between TIA Portal and Teamcenter....61

5.10 Difference between TIA Portal projects and global libraries....63
5.10.1 Copying different dataset types to a new revision....63

5.11 Managing TIA Portal projects with TIA Portal Teamcenter Gateway....65
5.11.1 Saving a TIA Portal project as a new item in Teamcenter....65
5.11.1.1 "General" tab in the "Save as new item" dialog - (TIA Portal projects)....67
5.11.1.2 "Additional properties" tab in the "Save as new revision..." dialog - (TIA Portal projects)....68
5.11.1.3 "General" tab in the "Save as new item" dialog - (TIA Portal projects)....68
5.11.1.4 "Additional properties" tab in the "Save as new item" dialog - (TIA Portal projects)....69
5.11.1.5 Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects)....70
5.11.2 Check out TIA Portal project....71
5.11.3 Check in TIA Portal project....72
5.11.4 Save changes to a TIA Portal project connected to Teamcenter....74
5.11.5 New revision of a TIA Portal project connected to Teamcenter....75
5.11.5.1 "General" tab in the "Save as new revision..." dialog - (TIA Portal projects)....77
5.11.5.2 "Additional properties" tab in the "Save as new revision..." dialog - (TIA Portal projects)....77
5.11.5.3 Structure of "General" tab in the "Save as new revision" dialog - (TIA Portal projects)....78
5.11.5.4 Structure of "Additional properties" tab in the "Save as new revision" dialog - (TIA Portal projects)....78
5.11.5.5 Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects)....79
5.11.6 Undo checkout of a TIA Portal project....81
5.11.7 Export a TIA Portal project....82
5.11.7.1 "Info" tab in the "Export..." - (TIA Portal projects) dialog....83
5.11.8 Saving a TIA Portal project from the TIA Portal to an item that exists in Teamcenter....83

5.11.8.1 Checks when using the function "Save to item..." - (TIA Portal projects).....83
5.11.8.2 Saving a TIA Portal project from the TIA Portal to an item that exists in Teamcenter.....84
5.11.8.3 Structure of dialog "Save to item...".....87
5.11.9 Save the TIA Portal project which is saved in the Teamcenter cache folder to the associated item again.....88
5.11.10 Opening a TIA Portal project connected to Teamcenter Gateway as reference project.....89

5.12 Workflows between the TIA Portal Teamcenter Gateway and TIA Portal multiuser engineering - TIA Portal projects.....92
5.12.1 Saving a TIA Portal project connected to Teamcenter as a multiuser server project.....92
5.12.1.1 Server project view of a multiuser server project.....92
5.12.1.2 Creating a local session of a specific multiuser server project.....93
5.12.1.3 Example of how to handle inconsistencies between TIA Portal projects in Teamcenter and in the multiuser server.....94
5.12.2 Check out (locked only).....94
5.12.3 Check in (unlocked only).....96
5.12.4 Saving changes to a multiuser server project to Teamcenter.....96
5.12.4.1 Examples of the data synchronization between the various dataset versions in the Teamcenter cache and in the multiuser server project.....100
5.12.5 Saving a multiuser server project as a new item in Teamcenter.....103
5.12.6 Saving the multiuser server project as a new revision in Teamcenter.....107
5.12.7 Saving a TIA Portal project (multiuser project) from the TIA Portal to an item that exists in Teamcenter.....111
5.12.7.1 Saving a TIA Portal project to an item that exists in Teamcenter in the server project view.....111
5.12.8 Opening a TIA Portal project connected to Teamcenter Gateway and added to the multiuser server as a reference project.....114
5.12.8.1 Opening a TIA Portal project connected to Teamcenter Gateway and added to the multiuser server as a reference project.....114

5.13 Managing global libraries with TIA Portal Teamcenter Gateway.....117
5.13.1 Saving a global library as a new item in Teamcenter.....117
5.13.1.1 "General" tab in the "Save as new item" dialog - (global libraries).....120
5.13.1.2 Structure of the "Additional properties" tab in the "Save as new item" dialog (global libraries).....120
5.13.1.3 Structure of "General" tab in the "Save as new item" dialog - (global library).....120
5.13.1.4 Structure of the "Additional properties" tab in the "Save as new item" dialog (global library).....121
5.13.1.5 Editing values in the tabular view of the "Additional properties" tab - (global libraries).....122
5.13.2 Check out global library.....123
5.13.3 Check in global library.....124
5.13.4 Save changes to a global library connected to Teamcenter.....126
5.13.5 Creating new revision of a global library connected to Teamcenter.....127
5.13.5.1 "General" tab in the "Save as new revision" dialog - (global libraries).....129
5.13.5.2 "Additional properties" tab in the "Save as new revision" dialog (global libraries).....129
5.13.5.3 Structure of "General" tab in the "Save as new revision" dialog - (global library).....129
5.13.5.4 Configuration of the "Additional properties" tab in the "Save as new revision" dialog (global library).....130
5.13.5.5 Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects).....131
5.13.6 Undo checkout of a global library.....132
5.13.7 Exporting a global library.....133
5.13.7.1 "Info" tab in the "Export" - (global library) dialog.....134
5.13.8 Saving a global library from TIA Portal to an item that exists in Teamcenter.....134
5.13.8.1 Checks when using the function "Save to item..." - (global libraries).....134

5.13.8.2 Saving a global library from TIA Portal to an item that exists in Teamcenter....135
5.13.8.3 Structure of dialog "Save to item...".....138
5.13.9 Save the global library which is saved in the Teamcenter cache folder to the associated item again....139

5.14 Finding and opening TIA Portal projects or global libraries with TIA Portal Teamcenter Gateway....141
5.14.1 Opening TIA Portal projects from the search result....144
5.14.1.1 Opening TIA Portal projects from the search result....144
5.14.1.2 Structure of the "Properties" tab in the "Open TIA Portal project from Teamcenter" dialog....146
5.14.2 Opening TIA Portal projects from the TIA Portal project list....148
5.14.3 Opening global libraries from the search result....148
5.14.3.1 Opening global libraries from the search result....148
5.14.3.2 Structure of the "Properties" tab in the "Open global Library from Teamcenter dialog....150
5.14.4 Displaying the dataset status in the "Open project from Teamcenter" or "Open global library from Teamcenter" dialog....152
5.14.4.1 Displaying the dataset status....152
5.14.4.2 Displaying the item revision status....153
5.14.4.3 Displaying the cache status....153

5.15 Manual comparison of two TIA Portal projects (dataset versions) with the same revision....154

5.16 Symbols....155
5.16.1 Open object status in Project dialog from Teamcenter....155
5.16.2 Object status in the Open global Library from Teamcenter dialog....156
5.16.3 Symbols that display the connection status between the TIA Portal and Teamcenter....157

Index....159

Creating and managing projects 1

1.1 The basics of projects

Projects contain data and programs resulting from the creation of an automation solution. The data assembled in a project include, for example:

- Configuration data on the hardware structure and parameter assignment data for modules
- Project engineering data for communication over networks
- Project engineering data for the devices
- Logs for important events in the life cycle of the project

Project version

The project version defines the version of the TIA Portal for which a project is compatible. A project that is compatible with the current version of the TIA Portal is provided in project version V15.1 of the TIA Portal.

You can view the project version of an already opened project in the project properties. To do so, select the "Software products in the project" entry in the "General" tab of the properties dialog. In the table, you can see all products and their versions, which are required for editing the open project.

Project hierarchy

Data is stored in a project in the form of objects. Within the project, the objects are arranged in a tree structure (project hierarchy).

The project hierarchy is based on the devices and stations along with the configuration data and programs belonging to them.

Common data of the project and online access, for example, are also displayed in the project tree.

Basic integrity check

The engineering software features a base-protective mechanism to detect changes in the essential custom project data (e.g. blocks, hardware configuration). Causes for changes to the project data, for example, can be tampering outside the engineering software or defects in the data storage medium.

The test of the configuration data is optional upon request, because, depending on the scope and application, it as can be time consuming, especially if you use multiuser engineering. You change the default for the test in the settings of the engineering software.

If a change is detected during a test, contact Siemens Customer Support. You can continue to work with the project data. Continue work at your own risk.

Note

Reduced working speed when basic integrity check is activated

If you enable the basic integrity check in the settings of the engineering software, the operating speed of the engineering software is reduced.

Project protection

You can activate user administration for a project and thereby protect your project. If protection is active for a project, you can only log on authorized users in the project and work with it.

See also: Basics of user administration property in the TIA Portal

See also

Using logs (Page 17)

Creating a project (Page 9)

Using projects created with optional software (Page 24)

Opening projects (Page 10)

Upgrading projects (Page 26)

Displaying properties of the project (Page 16)

Saving projects (Page 12)

Closing projects (Page 13)

Removing projects from the project list (Page 14)

Deleting projects (Page 15)

1.2 Creating a project

Procedure

To create a new project, follow these steps:

1. Select the "New" command in the "Project" menu.
   The "Create a new project" dialog opens.
2. Enter a project name and path or accept the suggested settings.
3. Click the "Create" button.

Result

The new project is created and displayed in the project tree.

See also

Compatibility between TIA Portal versions (Page 23)
The basics of projects (Page 7)
Using projects created with optional software (Page 24)
Opening projects (Page 10)
Upgrading projects (Page 26)
Displaying properties of the project (Page 16)
Saving projects (Page 12)
Closing projects (Page 13)
Removing projects from the project list (Page 14)
Deleting projects (Page 15)

1.3 Opening projects

You recognize projects of the TIA Portal by their file name extension ".ap[version number]". Projects of the current version have the file name extension ".ap15_1". Projects with the version V15.1 can be opened immediately. Upgrade projects with the project version V13 SP1, V14, V14 SP1 and V15 before opening.

Procedure

To open an existing project, follow these steps:

1. Select the "Open" command in the "Project" menu.
   The "Open project" dialog box opens and the list of recently used projects is displayed.
2. Select a project from the list and click "Open".
3. If the project you require is not included in the list, click the "Browse" button. Navigate to the desired project folder and open the project file.
4. To check whether the project has been changed without permission, select the check box "Activate basic integrity check". The basic integrity check may take some time to complete.
5. Click the "Open" button.
   If protection is active for the selected project, the "Login" dialog opens. Then perform the following steps:
   - Select the required user type.
   - Enter your user name.
   - Enter your password.
   - Click "OK".

   The project is opened if either project protection is not active or if you have logged on with a user account with the corresponding rights. Further information on user administration can be found under "Basics of user administration property in the TIA Portal".
   If you have selected a project from the V13 SP1, V14, V14 SP1 or V15 version of the TIA Portal, the "Upgrade project" dialog opens. You can find more information on upgrading projects under "Upgrading projects (Page 26)".

Checking for missing software components

When opening existing projects, an automatic check is made to determine if the appropriate software is installed for all modules used within the project. If you try to open a project with modules that are not supported by the current scope of installation of the TIA portal, a message appears on opening the project informing you of the missing software components. If the software components are not absolutely required to open the project, the project can still be opened.

If a project cannot be opened in the TIA Portal due to missing software components, you can download and install it as trial software from Siemens Industry online support. You can then open the project. Note that the trial software can only be used for a limited period of time.

Use the following link to open the page in Siemens Industry online support, where the various software products are available to you as trial software: www.siemens.com/tia-portal-trial (www.siemens.com/tia-portal-trial)

Reaction to missing software components

Unsupported modules of a project are handled as follows:

- Display of the modules on the user interface
  - The unsupported modules are displayed in the project tree with all of their lower-level objects. However, the modules themselves cannot be processed in editors or in the inspector window. When possible, a replacement module is used that best matches the original module. Replacement modules are indicated by an exclamation mark.
  - Display of properties in tables is limited. This applies in particular to the display of network parameters, such as the IP address.
- Functional limitations
  - Unsupported modules cannot be printed out or compiled.
  - An online connection cannot be established to the modules. It is therefore also impossible to download.
  - To change the device type, the device must first be deleted and then re-inserted. The "Change device type" function is not supported.
  - Copying and inserting lower-level objects, such as blocks, is possible although the device itself cannot be copied and inserted.
  - The network configuration cannot be changed with replacement modules within the network view.
  - Cross-references can be displayed. However, the cross-references only reflect the status last saved within the project because on online comparison with the original module cannot be made.

See also

The basics of projects (Page 7)

Creating a project (Page 9)

Using projects created with optional software (Page 24)

Upgrading projects (Page 26)

Displaying properties of the project (Page 16)

Saving projects (Page 12)

Closing projects (Page 13)

Removing projects from the project list (Page 14)

Deleting projects (Page 15)

1.4 Saving projects

You can save the project at any time either under the same or a different name. You can even save a project when it still contains elements with errors.

Saving a project

To save a project, follow these steps:

1. Select the "Save" command in the "Project" menu.
   All changes to the project are saved under the current project name.

Project Save as

To save a project under another name, follow these steps:

1. Select the "Save as" command in the "Project" menu.
   The "Save current project as" dialog opens.
2. Select the project folder in the "Save in" box.
3. Enter the new project name in the "File name" box.
4. Confirm your entry with "Save".
   The project is saved under the new name and opened.

Note

Undoing actions

Keep in mind that you cannot undo actions once you have saved the project.

See also

The basics of projects (Page 7)

Creating a project (Page 9)

Using projects created with optional software (Page 24)

Opening projects (Page 10)

Upgrading projects (Page 26)

Displaying properties of the project (Page 16)

Closing projects (Page 13)

Removing projects from the project list (Page 14)

Deleting projects (Page 15)

1.5 Closing projects

Procedure

To close a project, follow these steps:

1. Select the "Close" command in the "Project" menu.
   If you have made changes to the project since the last time you saved it, a message is displayed.
2. Decide whether or not you want to save the changes.

See also

The basics of projects (Page 7)

Creating a project (Page 9)

Using projects created with optional software (Page 24)

Opening projects (Page 10)

Upgrading projects (Page 26)

Displaying properties of the project (Page 16)

Saving projects (Page 12)

Removing projects from the project list (Page 14)

Deleting projects (Page 15)

1.6 Removing projects from the project list

You can remove projects from the list of recently used projects. The project data is retained on the storage medium.

Procedure

To remove a project from the list of recently used projects, follow these steps:

1. Select the "Delete project" command in the "Project" menu.
   The "Delete project" dialog opens and includes the list of most recently used projects.
2. Select a project from the list.
3. Click the "Remove" button.
4. Click "Yes" to confirm the prompt in order to remove the project from the list.

Result

The project is no longer displayed in the list of recently used projects. If you open the project again, it will be added to the list again.

See also

The basics of projects (Page 7)

Creating a project (Page 9)

Using projects created with optional software (Page 24)

Opening projects (Page 10)

Upgrading projects (Page 26)

Displaying properties of the project (Page 16)

Saving projects (Page 12)

Closing projects (Page 13)

Deleting projects (Page 15)

1.7 Deleting projects

Deleting removes the entire project data from the storage medium.

Requirement

The project you want to delete is not open.

Procedure

Follow the steps below to delete an existing project:

1. Select the "Delete project" command in the "Project" menu.
   The "Delete project" dialog opens and includes the list of most recently used projects.
2. Select a project from the list.
   If the project you require is not included in the list, click the "Browse" button. Navigate to the desired project folder, and open the project file.
3. Click the "Delete" button.
4. Click "Yes" to confirm. This starts the deletion of the project.

Result

The entire project folder is deleted from the file system.

See also

The basics of projects (Page 7)

Creating a project (Page 9)

Using projects created with optional software (Page 24)

Opening projects (Page 10)

Upgrading projects (Page 26)

Displaying properties of the project (Page 16)

Saving projects (Page 12)

Closing projects (Page 13)

Removing projects from the project list (Page 14)

1.8 Displaying properties of the project

You can display the properties of a project. In the "General" tab you find the following project properties in the area navigation:

- Metadata for the project
  This includes the following information: creation time, author, file path, project size, copyright, project languages, etc. Many of the properties can be changed.
- Project history
  The project history contains an overview with important events in the project life cycle. Here, for example, you can see the version of the TIA Portal used to create a project and whether it has been upgraded in the meantime. If a project was created during a migration, for example, this is also indicated in the project history table with the date and time of the migration. If a log was created for an event, you can also call the log directly.
- Support packages in the project
  An overview of the add-on software needed to work with all devices in the project is displayed. In addition, installed GSD files are listed (device description files for other devices in the hardware catalog).
- Software products in the project
  You can display an overview of all installed software products needed for the project. The "Version" column shows the project version of the respective software with which the project is stored.

In the "Protection" tab you specify that blocks can be simulated after the compilation with SIMATIC S7-PLCSIM Advanced. To do this, select the check box "Support simulation during block compilation". By default, the function is disabled.

Procedure

To display the project properties, follow these steps:

1. Select the open project in the project tree.
2. Select "Properties" in the shortcut menu of the project.
   The dialog with the properties of the project opens.
3. Select the project properties in the area navigation that you want to have displayed.

See also

The basics of projects (Page 7)

Creating a project (Page 9)

Using projects created with optional software (Page 24)

Opening projects (Page 10)

Upgrading projects (Page 26)

Saving projects (Page 12)

Closing projects (Page 13)

Removing projects from the project list (Page 14)

Deleting projects (Page 15)

1.9 Using logs

For some operations within the TIA Portal, logs are created automatically in the background. These logs document changes in the project. Logs are created automatically, for example, when you migrate projects and programs or when you update instances from the library.

Logs are displayed in the "Common data" folder in the project tree. They are stored in zip files together with the project in the project folder and can therefore be read as soon as you open the project independent of the programming device / PC used. The log can be filtered for errors, warnings, and information.

In addition to displaying them in the TIA Portal, logs can also be printed.

Displaying logs

To open a log, follow these steps:

1. Open the "Common data > Logs" folder in the project tree.
2. Double-click the desired log in the list.
   The contents of the log are displayed in the work area.
3. Optional: To show or hide a particular category of alarms, activate or deactivate the button for "Errors", "Warnings", or "Information" in the toolbar.

Deleting logs

To delete a log, follow these steps:

1. Select the log in the project tree.
2. Press <Del>.
   The selected log is deleted from the project directory and removed from the project tree.

1.10 Security of industrial plants

In addition to numerous advantages, the ongoing digitization of industrial automation is accompanied by increased vulnerability of industrial plants. The TIA Portal and the SIMATIC products already have several features that ensure secure operation of your plant, for example:

- Know-how protection of blocks
- Basic integrity protection for projects to protect projects against manipulation
- The protection and security functions of CPUs

Further information and the possibility for reporting security problems can be found online at the following link:

Whitepapers and additional information (https://www.industry.siemens.com/topics/global/en/industrial-security/support/Pages/white-papers.aspx)

Industrial Security

Industrial Security goes beyond the security regulations and measures in office environments. Industrial Security is a comprehensive approach to protecting industrial plants. The approach is designed to protect against unauthorized access, sabotage, espionage and malicious tampering. Industrial security ensures the following factors:

- Availability
  Availability has highest priority in industrial plants. Downtime must be kept as low as possible.
- System and data integrity
  A third party should not be able to unlawfully modify the plant and any of its data.
- Confidentiality
  Information on the industrial plant and data must not fall into the hands of unauthorized third parties.

Comprehensive approach

In addition to the security features integrated in the TIA Portal, Industrial Security requires more comprehensive measures. Observe the following security aspects:

- Plant safety
  - Access protection against unauthorized persons
  - Physical access protection for critical components
- Network security
  - Controlled interfaces between an office and plant network, for example, through firewalls
  - Segmentation of the plant network
- System integrity
  - Use of anti-virus and whitelisting software
  - Compliance with maintenance and update processes
  - User authentication for the machine or plant operator
  - Integrated access protection in automation components

1.11 Importing and exporting CAx data

CAx files are XML files that you create in the TIA Portal and that you can swap between different installations or with other tools such as ECAD systems. The files contain the following information:

- Hardware configuration of a single device or an entire project (without parameter assignment)
- Folder hierarchies (groups)
  The folder hierarchy is restored for the import.
- Association of a device to a subnet
  When importing, the devices stored in the CAx file are reconnected to the same subnet.
- Addresses of the devices
- Address areas of inputs and outputs
- Profinet I/O systems and PROFIBUS DP master systems

Requirements

- TIA Openness is installed.
  TIA Openness is installed automatically by the setup program of the TIA Portal. To do so, the check box for TIA Openness must be selected in the setup program under "Options" in the "Configuration" menu.
- A project is open.

Exporting the CAx file of a project

1. In the "Tools" menu, click on "Export CAx data...".
2. In the "CAx Export" dialog, select a directory for the file to be exported.
3. If necessary, change the file name and confirm with "Save".
   A dialog shows the export progress. If you need to cancel the export, click "Cancel".
   You see the status of the export in the "Info > General" tab of the Inspector window. In addition, a log file is created in the directory that you entered as the storage location for log files under "Settings > General > Data exchange".
   The exported file receives the file extension ".aml".

Exporting the CAx file of a device

1. Highlight the exported devices in the project tree.
2. Select "Export CAx data ..." from the shortcut menu.
3. In the "CAx Export" dialog, select a directory for the file to be exported.
4. If necessary, change the file name and confirm with "Save".
   A dialog shows the export progress. If you need to cancel the export, click "Cancel".
   You see the status of the export in the "Info > General" tab of the Inspector window. In addition, a log file is created in the directory that you entered as the storage location for log files under "Settings > General > Data exchange".
   The exported file receives the file extension ".aml".

Import CAx file of a device or a project

1. In the "Tools" menu, click on "Import CAx data...".
2. Select a CAx file in a directory in the "CAx Import" dialog.
3. Click the "Open" button.
   A dialog shows the import progress. If you need to cancel the import, click "Cancel".
   You receive feedback about the status of the import of devices, for example error messages, in the inspector window in the "Info > General" tab. In addition, a log file is created in the directory that you entered as the storage location for log files under "Settings > General > Data exchange".

Permitted exchange formats for CAx data

The standardized data exchange format "AutomationML" is used to exchange CAx data. The exchange files receive the extension "*.aml". Exporting and importing files with the extension "*.xml" is not supported.

The log files generated during the data exchange receive the extension "*.log". Logs files with the extension ".txt" are not supported.

Exchanging CAx data with other applications

Plugged in submodules such as bus adapters can have a different hierarchical structure in the TIA Portal and in other applications. The AML files for exchanging CAx data therefore display the submodules in a customized hierarchy that can be read by all programs. During import, the hierarchy of submodules in the AML file does not affect the hierarchy of submodules in the TIA Portal.

Note

Compatibility between different product versions.

If you want to exchange files with devices that contain bus adapters, note the following information on compatibility:

AML files created with an older version of the TIA Portal can be imported with the current version.

AML files created with the current version of the TIA Portal cannot be imported with older versions.

Import options with name conflicts of stations

If you want to import a CAx file and the currently open TIA Portal project included stations with identical names, a conflict arises. A dialog for setting the import options opens. Before you import the file, you select how the stations with the same name are to be merged. The stations from the CAx file can be optionally renamed and grouped together in a new folder.

Note

Default name of the folder

The folder for the stations to be imported are named "ParkingLot" by default.

You can specify the name of the new folder yourself. To do this, open the settings and navigate to the "CAx" item. Enter the name for the folder with "Settings for conflict resolution" in the "Name of store for devices that already exist in the project" field.

Optionally, you can also specify that the stations triggering a conflict are either not to be imported or replaced by stations of the same name in the currently open TIA Portal project.

Compatibility of projects 2

2.1 Compatibility between TIA Portal versions

With the TIA Portal V15.1, you can open projects that were created with this version of the TIA Portal. You need to upgrade projects from previous versions of the TIA Portal to the current version before opening them.

Opening projects from previous versions of the product

The following table shows the response of the TIA Portal when opening projects from a previous version of the product:

Version of the TIA Portal (file extension of the respective version)	Response when opened with the current version of the TIA Portal
V10.5 (.ap10) V11.x (.ap11) V12 (.ap12) V12 SP1 (.ap12) V13 (.ap13)	Projects from previous versions of V13 SP1 of the TIA Portal cannot be opened directly with the TIA Portal V15.1. When you attempt to open the project, a message prompts you to open the project with the version V13 SP1. If you open the project with TIA Portal V13 SP1 version, the project is upgraded to the project version V13 SP1 after your confirmation and is assigned the file extension ".ap13".
V13 SP1 (.ap13) V14 (.ap14) V14 SP1 (.ap14) V15 (.ap15)	After your confirmation the project is upgraded to project version V15.1 upon opening and receives the file extension ".ap15_1".

Backward compatibility of the current version of the TIA Portal

Projects that you save in the current project version V15.1 are not backward compatible due to the expanded functionality over previous versions. Projects in version V15.1 can only be opened with TIA Portal V15.1 or higher.

Open projects with assigned role permissions

If specific role permissions were set up in the user administration, the user administration for this project can only be opened in the product versions that support these role permissions.

2.2 Using projects created with optional software

A wide range of optional software is available for the TIA Portal. This includes hardware support packages, for example. When you open a project that was created with optional software but the optional software is not installed, the following will occur:

- Software components which are not absolutely required are missing:
  A dialog appears listing the missing software components. After the project is opened, its properties are displayed. You now have the opportunity to install missing components. All the devices contained in the project are available even if you do not install the missing components. However, you can only work with the devices that are supported by the currently installed software.
  Devices which are not supported because software components are missing are identified by the following symbol in the project tree:

- Essential software components are missing:
  A dialog appears listing the missing software components. The essential software components are marked. The project can only be opened if you install the missing software components.

See also

The basics of projects (Page 7)

Creating a project (Page 9)

Opening projects (Page 10)

Upgrading projects (Page 26)

Displaying properties of the project (Page 16)

Saving projects (Page 12)

Closing projects (Page 13)

Removing projects from the project list (Page 14)

Deleting projects (Page 15)

2.3 Upgrading projects

2.3.1 General information on upgrading projects

A project from the version V13 SP1 of the TIA Portal can be opened only after they have upgraded the project to the current version of the TIA Portal. When you open the project, you are prompted to upgrade the project.

To learn about the options available with other product versions of the project, refer to the section "Compatibility between TIA Portal versions (Page 23)".

Upgrading modules with know-how protection

Blocks with know-how protection are not automatically upgraded with the project. Remove the know-how protection of the blocks before you upgrade the project:

- For V13 SP1 projects
  Remove the know-how protection with TIA Portal V13 SP1.
- For V14 projects
  Remove the know-how protection with TIA Portal V14 or the TIA Portal V14 SP1 in compatibility mode.
- For V14 SP1 projects
  Remove the know-how protection with TIA Portal V14 SP1.
- For V15 projects
  Remove the know-how protection with TIA Portal V15.

Set up know-how protection again afterwards with the latest version of the TIA Portal. You can find additional information in the section "Upgrading know-how protected blocks (Page 28)".

Upgrading protected projects

If you want to upgrade a project which is protected with the administration property from an earlier product version, you have to be registered in this project as a project administrator and need the access data for the project.

When protected projects are upgraded, the "Login" dialog opens. To continue with the procedure select the user type and enter your user name and your password.

You can find additional information in the section "Basics of user administration in the TIA Portal".

Upgrading global libraries

Global libraries are not upgraded automatically together with the project because they are independent of projects. If you want to continue using global libraries from older versions of the TIA Portal, upgrade the global libraries as well. You can find more information on upgrading global libraries in the section "AUTOHOTSPOT".

Instruction version

The instructions that you use to create your user program may be in different versions. Different product versions of the TIA Portal can contain different instruction versions. You have the option to upgrade the project and instructions to the latest version. You can then use the new instruction versions. After upgrading, the new instruction versions are not automatically used in your program. You can update your program to use the new versions. You can find more about "Instruction versions" in the "Programming editor" section.

Multiuser Engineering

To upgrade multiuser engineering projects, read about the prescribed procedure in the section "Upgrading multiuser engineering projects (Page 28)".

Upgrading projects with a missing license

No license is required to upgrade projects. If you do not have a license or do not have a suitable license, you cannot use the full range of functions in the products.

With a STEP 7 Basic license, for example, you cannot use functions provided by a STEP 7 Professional license.

See also

Using projects created with optional software (Page 24)

Upgrading projects (Page 26)

2.3.2 Upgrading projects

The following will show you how to upgrade projects from earlier versions of the TIA Portal.

Upgrading projects from TIA Portal V14 or higher

To upgrade a project from TIA Portal V14, follow these steps:

1. Select the "Open" command in the "Project" menu.
   The "Open project" dialog box opens and the list of recently used projects is displayed.
2. Select a project from the list and click "Open".
3. If the project you require is not included in the list, click the "Browse" button. Navigate to the desired project folder and open the project file.
   The "Open project" dialog opens.
4. Click "Upgrade".
   The "Upgrade project/library" dialog box opens.
5. Click "OK".
   The project is upgraded and opened in the current version of the project.

Upgrading projects from TIA Portal V13 SP1

To upgrade a project from TIA Portal V13 SP1, follow these steps:

1. Select the "Open" command in the "Project" menu.
   The "Open project" dialog box opens and the list of recently used projects is displayed.
2. Select a project from the list and click "Open".
3. If the project you require is not included in the list, click the "Browse" button. Navigate to the desired project folder and open the project file.
   The "Open project" dialog opens.
4. Click "Upgrade".
   The "Upgrade project/library" dialog box opens.
5. Click "OK".
6. Click "OK" to confirm the prompt window.
   The project is upgraded and opened in the current version of the project.
7. Compile the hardware and software of all devices in the project.

Upgrading TIA Portal V13 SP1 projects with master copies from TIA Portal V10.5

Before upgrading a project, check if the master copies created with TIA Portal V10.5 are contained in its project library. This is necessary, for example, if you receive a message during the upgrade about an S7-1200 CPU with firmware version V1.0 used in the project, even though you have not configured such a CPU. The CPU is a master copy in the project library in this case.

Before upgrading a project with master copies from TIA Portal V10.5, therefore, do the following:

1. Use each master copy created with the TIA Portal V10.5 once in the project.
2. Delete the affected master copies from the project library.
3. Create the master copy again with the objects from the project.
4. Save the project in TIA Portal V13 SP1.

You can then upgrade the project to the latest version.

Upgrading projects from the TIA Portal V13 or earlier

Projects of product version V13 or earlier cannot be upgraded and opened with the TIA Portal. First upgrade projects from TIA Portal V13 or earlier with the product version V13 SP1. Then upgrade the projects with the current version of the TIA Portal again to the project version V15.1.

You can find additional information on upgrading projects from earlier product versions in the Siemens Industry online support (https://support.industry.siemens.com/cs/ww/en/view/109476392).

See also

Upgrading know-how protected blocks (Page 28)
General information on upgrading projects (Page 25)
Upgrading multiuser engineering projects (Page 28)
The basics of projects (Page 7)
Creating a project (Page 9)
Using projects created with optional software (Page 24)
Opening projects (Page 10)
Displaying properties of the project (Page 16)
Saving projects (Page 12)
Closing projects (Page 13)
Removing projects from the project list (Page 14)
Deleting projects (Page 15)

2.3.3 Upgrading multiuser engineering projects

The following will show you how to upgrade multiuser engineering projects.

Procedure

To upgrade multiuser engineering projects, follow these steps:

1. Before upgrading, save all existing local sessions in the multi-user server project.
2. Save the multiuser server project as a single-user project.
3. Upgrade the single-user project.
4. After the upgrade, create a multiuser project again from the single-user project.
5. Create the local sessions again from the upgraded multiuser project for all workers.

See also

Upgrading projects (Page 26)

2.3.4 Upgrading know-how protected blocks

For security reasons, the know-how protection of blocks is tied to the version of the TIA Portal with which the know-how protection was set up. This means blocks that were assigned know-how protection in an earlier product version cannot be opened and edited with the previous password after the upgrade. However, the blocks can be downloaded to the controller and are executable. If you want to open and edit the block after the upgrade, remove the know-how protection of the blocks and set it up again.

Procedure

To upgrade know-how protected blocks, follow these steps:

1. Remove the know-how protection of the blocks from the previous version in the "Edit" menu with the "Know-how protection" command.
   For more detailed information, refer to section "Removing know-how protection for blocks".
2. Upgrade the project as described in the section "Upgrading projects (Page 26)".
3. Set up the know-how protection of the blocks from the current product version again in the "Edit" menu with the "Know-how protection" command.
   For more detailed information, refer to section "Setting up know-how protection for blocks".

2.4 Compatibility with the plant configuration

To be able to use all the online functions, the plant configuration must be compatible with TIA Portal V15.1 and the project version of the opened project. The following factors for the compatibility are especially relevant when designing the plant:

- Hardware and its firmware version
  The individual devices and their firmware version must be supported by the TIA Portal V15.1 and within the opened project.
- The user program on the respective devices
  All blocks and instructions on the devices must be available in TIA Portal V15.1 and ready for using with the project version of the opened project.

Influence of the project version on compatibility

With TIA Portal V15.1, you can edit projects for project version V15.1. You can only load the hardware configuration and software of the system which is compatible with your version of the TIA Portal.

With CPUs of the S7-1200 and S7-1500 product series, you see the project version with which the blocks on the CPU are compatible on the display, on the Web server and in the online and diagnostics view of the TIA Portal. Note, however, that a compatible version of the project alone is not enough to work with the CPU. The CPU itself and its firmware version must be compatible with the installed version of the TIA Portal and the various installed products.

You can see the project version of the currently open project in the project properties.

Online connection after upgrading a project

When you upgrade a project from version V14, V14 SP1 or V15 to version V15.1, you can then connect devices online that were loaded with version V14, V14 SP1 or V15. The blocks of these devices remain functional and do not have to be recompiled.

You can this expand the plant configuration or add functionality to devices, for example, without the changes affecting other devices in the plant configuration.

The difference in the project version is indicated in the project tree and in the Compare editor by an overlay symbol on the device.

The following symbol indicates the difference in project version when no program blocks have been modified:

If no blocks were modified, no online/offline difference is indicated for the blocks.

The following symbol indicates the difference in project version when program blocks have been modified:

If blocks were modified, an online/offline difference is indicated for the blocks. You cannot load these blocks from the device because only blocks of the same version can be loaded to TIA Portal V15.1.

Incompatible blocks

An offline/online comparison is performed for the configured devices when an online connection exists. If the modules available online are incompatible with the project version of one project, incompatible blocks in the project tree are indicated as such by the following symbol:

Loading hardware configuration and software on a device

If you want to load the hardware configuration or software on a device that was last loaded with TIA Portal V15 or an earlier version, then load the entire hardware configuration and software to the device again. It is not possible in this case to only load the changes. On devices that were last loaded with TIA Portal V15.1 or higher, you can only load the changes to the hardware configuration and software.

See also

The basics of projects (Page 7)

General information on upgrading projects (Page 25)

Upgrading projects (Page 26)

Displaying properties of the project (Page 16)

Archiving and retrieving projects 3

3.1 Working with project archives

Archiving and transferring projects

If you work for a long time with a project, large files may result, especially with extensive hardware configurations. Therefore, you may want to reduce the size of the project, for example, when you archive it to an external hard drive, or when you send it via e-mail and require a smaller file size.

You can create a project archive to reduce the size of the project. TIA Portal project archives are compressed or uncompressed files, each containing an entire project including the entire folder structure of the project. Before the project directory is compressed into the archive file, all files are reduced to their essential components to further decrease the size of the project.

Project archives have the file extension ".zap[version number of the TIA Portal]". Projects that were created with TIA Portal V15.1 have the file extension ".zap15_1".

To open a project archive, retrieve the project archive. By retrieving it, the archive file with the included project files is extracted to the original project directory structure.

See also

Retrieving compressed project (Page 35)

Creating a project archive (Page 34)

3.2 Creating a project archive

Projects of the current product version can be archived as compressed or uncompressed files. The projects to be archived do not have to be open in TIA Portal to do this.

The storage space of projects can be reduced by archiving projects in compressed files.

Note

The most recently saved state of an open project is used for archiving. Therefore, save the project before using the archiving function. This will ensure that your most recent changes are included in the archived project.

Procedure

To archive a project, follow these steps:

1. Select the "Archive ..." command from the "Project" menu.
   The "Archive" dialog opens.
2. In the "Source path" field select a project file with the extension ".ap15_1".
3. To create a compressed archive file, select the "Archive as compressed file" option.
4. If you do not want to archive the search index and the HMI compile result, select the option "Discard restorable data".
   You can restore the discarded data if needed.
5. To add a date and time automatically, select the "Add date and time to the target name".
6. In the "Target path" field select the directory where you want to save the archive file or the new directory of the project.
   You can set the default directory under "Options > General > Archive settings > Storage location for project archives".
7. Click "Archive".

Result

An archive of the project is created. Compressed projects contain the file extension ".zap15_1". The archive file contains the complete project directory. The individual files of compressed archives are additionally reduced to the essential components in order to save space.

At uncompressed projects only a copy of the original project directory is created at the desired storage location.

See also

Working with project archives (Page 33)

Retrieving compressed project (Page 35)

3.3 Retrieving compressed project

You extract project archives of the TIA Portal with the "Retrieve" function. This restores the project directory structure including all project files.

Requirement

No project is open.

Procedure

To extract a project archive, follow these steps:

1. Select the "Retrieve" command in the "Project" menu.
   The "Retrieve archived project" dialog opens.
2. Select the project archive.
3. Click "Open".
4. The "Find folder" dialog opens.
5. Select the target directory to which the archived project should be extracted.
6. Click "OK".

Result

The project is extracted to the selected directory and opened immediately. If you extract a project archive that includes a project from the product version V13 SP1, you may have to upgrade the project. You will automatically receive the corresponding prompt as soon as you open the project. The same rules apply that are described in the chapter "Using projects created with optional software (Page 24)". The search index and HMI compile result are automatically restored in the background.

See also

Working with project archives (Page 33)

Opening projects (Page 10)

Using projects created with optional software (Page 24)

Upgrading projects (Page 26)

Using reference projects

4

4.1 Basics of reference projects

Introduction

You can open other projects as a reference in addition to the current project. You can use these reference projects as follows:

- You can drag individual objects from a reference project into the current project and then edit them.
- You can open specific objects, for example, code blocks from a reference project as read-only. But this is not possible for all elements.
- You can use an offline/offline comparison to compare devices of the reference project to devices from the current project.

Note that reference projects are read-only. You cannot change the objects of a reference project. Therefore pending changes, such as name changes for objects, cannot be applied to the referencing object until the project is opened as a normal project. After the changes were applied you can continue using the project as a reference project.

Projects that were created with an older version of the TIA Portal or with a different installation package can also be opened as reference projects. The same compatibility rules apply here as for normal opening of a project from an older version of the TIA Portal.

See also: Compatibility of projects (Page 24)

See also

Comparing reference projects (Page 39)

Opening and closing a reference project (Page 38)

4.2 Opening and closing a reference project

Opening a reference project

To open a reference project, follow these steps:

1. In the "View" menu, select the "Reference projects" check box.
   The "Reference projects" palette is displayed in the project tree.
2. In the "Reference projects" palette of the project tree, click on "Open reference project" in the toolbar.
   The "Open reference project" dialog box opens.
3. Navigate to the desired project folder and open the project file. TIA Portal V15.1 projects have the extension ".ap15_1". Older projects of the TIA Portal have the extension ".ap[version number]".
4. Click "Open".
   The selected project is opened as a read-only reference project.

Closing a reference project

To close a reference project, follow these steps:

1. In the "Reference projects" palette of the project tree, select the reference project you want to close.
2. Click on "Close reference project" in the toolbar.
   The selected reference project is closed.

See also

Basics of reference projects (Page 37)

Comparing reference projects (Page 39)

4.3 Comparing reference projects

Introduction

You can compare devices from reference projects with devices from both the current project as well as from the same or another reference project or from a library.

Note

Please note the following:

- You cannot specify actions for the comparison objects, since the reference projects are write-protected.
- You can perform a detailed comparison for the comparison objects, if the type of comparison object generally allows a detailed comparison.
- When comparing reference projects, you can always switch between automatic and manual comparison.

Procedure

To compare the objects of a reference project to the device data of the current project, follow these steps:

1. In the project tree, select the device whose data you want to compare to the data of a reference project and which allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Open the "Reference projects" palette in the project tree.
4. Select the device of a reference project that you want to compare to the device data from the current project.
5. Drag the device from the reference project into the right drop area of the compare editor.
   You can identify the status of the objects based on the symbols in the status and action area. When you select an object, the object properties and the corresponding object of the assigned device are clearly shown in the property comparison.
   You can drag other devices into the drop areas from the current project, a library or from a reference point at any time to start a new comparison. It does not matter which device you drag into the drop area.
6. If needed, perform a detailed comparison of objects for which this type of comparison is possible.

   Note

   You can start the detailed comparison directly from the reference project for blocks, PLC tags and PLC data types. You can find additional information in the section "Comparing PLC programs".

See also

Basics of reference projects (Page 37)
Opening and closing a reference project (Page 38)

TIA Portal Teamcenter Gateway 5

5.1 Introduction to TIA Portal Teamcenter Gateway

5.1.1 Basic information on working with TIA Portal Teamcenter Gateway

Introduction

As of TIA Portal Version 14 and WinCC Version 15, you have the possibility to work with the TIA Portal Teamcenter Gateway option. This has the advantage that you can save and manage your TIA Portal projects and global libraries in Teamcenter.

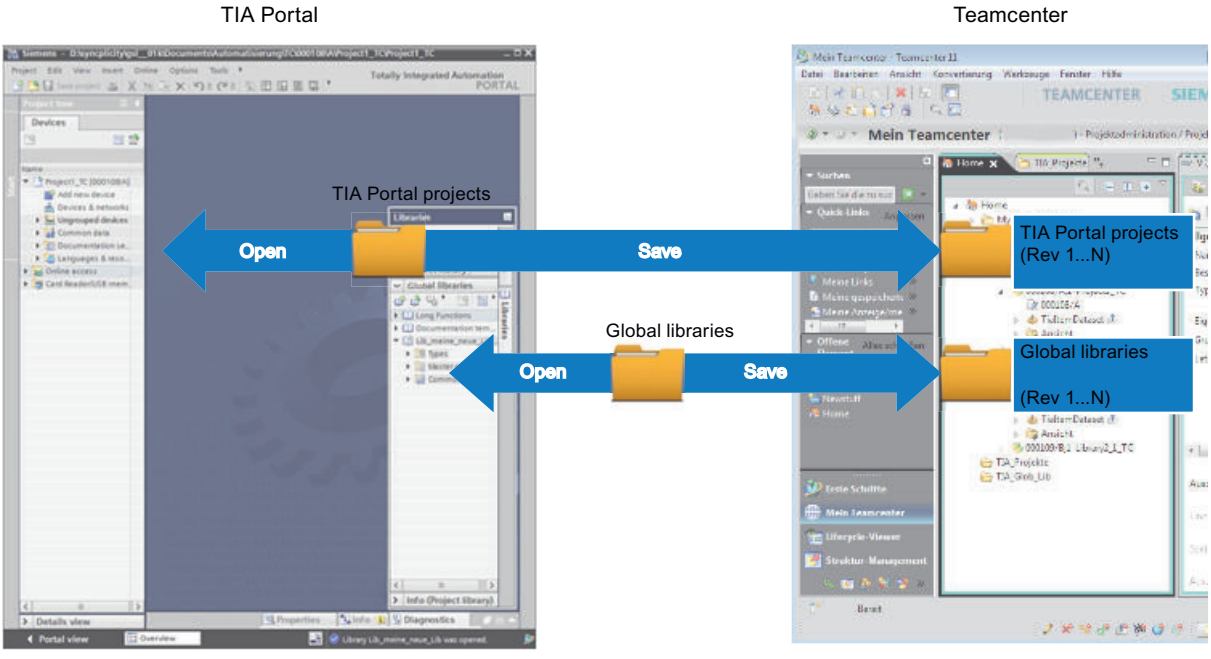

Advantages

The TIA Portal Teamcenter Gateway option provides the following advantages:

- You can save TIA Portal projects and global libraries in Teamcenter.
- You can assign an item type to a TIA Portal project and to a global library.
- You can revise/version TIA Portal projects and global libraries that are saved in Teamcenter.
- You can open TIA Portal projects and global libraries saved in Teamcenter for further editing in TIA Portal.

- You can add TIA Portal projects that are connected to Teamcenter to the multiuser server as multiuser server projects. You can save changes that you have made to multiuser server projects as a new version (dataset version) to Teamcenter.
- You can export TIA Portal projects or global libraries to remove the Teamcenter metainformation and to save the modified project or the global library in the selected folder.

5.1.2 Application possibilities of TIA Portal Teamcenter Gateway

Application possibilities of TIA Portal Teamcenter Gateway

You can do the following with TIA Portal Teamcenter Gateway:

- Save TIA Portal projects or global libraries as a new item in Teamcenter.
- Assign an item type or derived types to TIA Portal projects or global libraries that you want to save as new item in the Teamcenter.
- Save changes to a TIA Portal project or a global library in Teamcenter.
- Save TIA Portal projects or global libraries as new revision in Teamcenter.
- Save Element types of the TIA Portal projects or global libraries or dependenet types als new revisions in Teamcenter.
- Check out TIA Portal projects or global libraries.
- Undo checkout of a TIA Portal project or global library.
- Check in TIA Portal projects or global libraries.
- Open TIA Portal projects or global libraries saved in Teamcenter for editing in TIA Portal.
- Export TIA Portal projects or global libraries to remove the Teamcenter metainformation and to save the modified project or the global library in the selected folder.
- You can add TIA Portal projects that are connected to Teamcenter to the multiuser server as multiuser server projects.

5.2 System requirements

5.2.1 System requirements for TIA Portal Teamcenter Gateway

System requirements

Working with TIA Portal Teamcenter Gateway in V14 or higher is subject to the same hardware and software requirements as working with TIA Portal V14 or higher.

Note

Identical software installation as requirement for working with TIA Portal Teamcenter Gateway

To work with TIA Portal Teamcenter Gateway without any restrictions, it is important that the same software products of the TIA Portal and the TIA Portal Teamcenter Gateway are installed with identical product versions.

System requirements for installation of TIA Portal Teamcenter Gateway

- You have administrator privileges on your computer.
- You have installed Teamcenter Rich Application Client (referred to as "RAC" below) or Teamcenter Client Communication System (TCCS) version 11.2 or higher on your PG/PC.

  Note

  If TCCS was installed as "standalone", you must also install "Microsoft Visual C++ 2013 – Redistributable". This ensures that you can establish a connection between TIA Portal and TIA Portal Teamcenter Gateway.

- You have made sure there is a connection of FCC/FMS (FileClientCache/FileManagementSystem) to the "Teamcenter Server 11.2 or higher".

  Note

  After re-installation or a restart, also check that a connection is available from FCC/FMS (FileClientCache/FileManagementSystem) to the "Teamcenter Server 11.2 or higher".

- You have installed on the Teamcenter server the Teamcenter data model included in the scope of delivery of TIA Portal Teamcenter Gateway.
  You can find a detailed description for installation of the Teamcenter data model in the installation manual "TIA Portal Teamcenter Gateway - Installing a data model".
- You have imported into Teamcenter the Teamcenter search queries "__T4Tia_Item.xml", "__T4Tia_ItemRevision.xml", "__T4Tia_ItemWithRevision.xml" and "__T4Tia_RevisionOrderbyCreatedDateTime.xml" included in the product package of TIA Portal Teamcenter Gateway.
  For a detailed description of importing Teamcenter search queries, refer to the installation manual "TIA Portal Teamcenter Gateway - Installing a data model" in the section "Importing search queries in Teamcenter".

Licenses for TIA Portal Teamcenter Gateway

To work with TIA Portal Teamcenter Gateway, you need a valid license key on every device on which you install TIA Portal Teamcenter Gateway.

You can either include the license key in the installation or transfer it using the Automation License Manager after the installation. When you work with the TIA Portal in combination with TIA Portal Teamcenter Gateway, the presence of the valid license key is checked.

Installing TIA Portal Teamcenter Gateway in conjunction with Windows 10

Note the following:

- Restart the computer after the installation of the TIA Portal Teamcenter Gateway in conjunction with Windows 10.

Requirements for working between TIA Portal Teamcenter Gateway and Multiuser Engineering

- You have access to the multiuser server.
- You have access to the multiuser server project.

5.3 Installing and uninstalling TIA Portal Teamcenter Gateway

5.3.1 Notes on installing TIA Portal Teamcenter Gateway

Introduction

You can install TIA Portal Teamcenter Gateway as follows:

- Together with the installation of the TIA Portal.

Note

Administrator rights are required for the server installation.

Installing TIA Portal Teamcenter Gateway with the TIA Portal

TIA Portal Teamcenter Gateway is usually activated in the product selection by default and is then installed with the TIA Portal. Check the settings during installation of the TIA Portal. You can find the specific default settings for TIA Portal Teamcenter Gateway during setup when selecting the product in the "Options" directory. By selecting or clearing the options, you can specify whether or not the TIA Portal Teamcenter Gateway is should be installed along with the TIA Portal.

Identical TIA Portal versions during installation

When installing different TIA Portal products, make sure that you use the same versions of service packs and updates for the installation.

If you have installed TIA Portal V14 or higher, for example, you also have to install TIA Portal Teamcenter Gateway V14 or higher.

Installation path

Do not use any UNICODE characters (for example, Chinese characters) in the installation path.

Antivirus programs

During the installation, read and write access to installed files is necessary. Some antivirus programs block this access. We therefore recommend that you disable antivirus programs during the installation of TIA Portal Teamcenter Gateway and enable them again afterwards.

Note on virtualization

TIA Portal Teamcenter Gateway has been tested with the following VMware (virtual machine):

- VMware Workstation 12 Pro

5.3.2 Notes on compatibility of TIA Portal Teamcenter Gateway

Compatibility of TIA Portal Teamcenter Gateway with other software products

TIA Portal Teamcenter Gateway has been tested for compatibility with the following software products:

TIA Portal installations

- WinCC Professional
- WinCC UAES
- WinCC Advanced

Add-on packages

- STEP 7 Safety
- Multiuser-Server
- Cloud Connector

5.3.3 Parallel installation

Note on parallel installations

The following parallel installations have been tested:

- TIA Portal V14 SP1 and TIA Portal Teamcenter Gateway V14 SP1 with TIA Portal V13 SP1
  and
- TIA Portal V14SP1 with Teamcenter Gateway V14 SP1 with STEP 7 Classic 5.5
- TIA Portal V15 and TIA Portal Teamcenter Gateway V15 with TIA Portal V13 SP2
  and
- TIA Portal V15 with Teamcenter Gateway V15 with STEP 7 Classic 5.6
- TIA Portal V15.1 and TIA Portal Teamcenter Gateway V15.1 with TIA Portal V14 SP1
  and
- TIA Portal V15.1 and TIA Portal Teamcenter Gateway V15.1 with STEP 7 Classic 5.6

5.3.4 Starting the installation of TIA Portal Teamcenter Gateway

Introduction

The TIA Portal Teamcenter Gateway can be installed as part of the TIA Portal installation. If TIA Portal Teamcenter Gateway is installed with the TIA Portal, the version of TIA Portal Teamcenter Gateway installed will always correspond to that of the TIA Portal.

A description of automated installation is also available on the product DVD in the directory "Documents\Readme\<language directory>".

Requirements for installation of TIA Portal Teamcenter Gateway

The following requirements apply to the installation of TIA Portal Teamcenter Gateway:

For more details, see: System requirements for TIA Portal Teamcenter Gateway (Page 43)

Installing TIA Portal Teamcenter Gateway with the TIA Portal

Follow these steps for installation:

1. Note the requirements for the installation of the selected software package.
2. Start the setup program of the required product by inserting the installation medium into the respective drive.
3. Select the preferred settings and click the "Install" button.
4. Make sure that TIA Portal Teamcenter Gateway is selected in the product selection and observe the notes in the setup dialogs.

Result

TIA Portal Teamcenter Gateway is installed on your computer together with the TIA Portal.

The installation of TIA Portal Teamcenter Gateway adds the "Open Teamcenter project..." submenu to the "Project" menu in the project view.

Furthermore, the "Teamcenter" submenu is added to the "Tools" menu in the main menu of TIA Portal.

5.3.5 Displaying installed TIA Portal Teamcenter Gateway software

You can find out which software is installed at any time. In addition, you can display more information on the installed software.

Procedure

To display an overview of the software installed, follow these steps:

1. Click "Installed software" in the "Help" menu.
   The "Installed software" dialog opens. You will see the installed software products in the dialog. Expand the entries to see which version is installed in each case.
2. If you would like to display additional information about the installed software, click the link in the dialog "Detailed information about installed software".
   The "Detailed information" dialog opens.
3. Chose the topic you want more information about in the area navigation.

5.3.6 Checking the installation of TIA Portal Teamcenter Gateway

Procedure

Ensure that

- either Teamcenter Rich Application Client (RAC) 11.2 or higher
- or Teamcenter Client Communication System (TCCS) 11.2 or higher

is correctly installed on your TIA Portal computer.

The following table shows the "corresponding" registration key. The use of the Windows 7 operating system (64-bit) on the TIA Portal computer is required for described scenarios.

Product	64-bit installation Registration key	Value Name	Value Data
TC RAC 11.2	HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Teamcenter**11000_2** [1]	-	-
TC RAC 11.3	HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Teamcenter**11000_3** [1]	-	-
TC RAC 11.4	HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Teamcenter**11000_4** [1]	-	-
TC RAC 11.5	HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Teamcenter**11000_5** [1]	-	-
TCCS 11.2	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Teamcenter client communication system	"DisplayVersion"	11.2.* [2]
TCCS 11.3	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Teamcenter client communication system	"DisplayVersion"	11.3* [2]
TCCS 11.4	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Teamcenter client communication system	"DisplayVersion"	11.4* [2]
TCCS 11.5	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Teamcenter client communication system	"DisplayVersion"	11.5* [2]

* Additional specific values may be available, which you can ignore. Use a wildcard character for this purpose.

[1] TIA Portal Teamcenter Gateway checks whether the existing registration key starts with "11000_". All earlier versions of TC RAC are invalid.

[2] TIA Portal Teamcenter Gateway checks whether the value of the displayed version starts with "11.". All earlier versions of TCCS are invalid.

5.3.7 Modifying or updating TIA Portal Teamcenter Gateway

You have the option of modifying installed products using the setup program or to update to a new version.

Requirement

- TIA Portal V14 or higher and TIA Portal Teamcenter Gateway V14 or higher is installed on your PG/PC.
- Teamcenter Rich Application Client (referred to as "RAC" below) or Teamcenter Client Communication System (TCCS) version 11.2 or higher is installed on your PG/PC.
- A connection is available from FCC/FMS (FileClientCache/FileManagementSystem) to "Teamcenter Server 11.2".
- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- You have exited all active TIA Portal processes.

Procedure

To modify or update TIA Portal Teamcenter Gateway, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking on the "Start.exe" file.
   The dialog "Installation language" for selecting the installation language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read installation notes" or "Read product information" button.
   The help file containing the instructions opens.
5. Once you have read the instructions, close the help file and click the "Next" button.
   The dialog "A previous installation has been detected. Please select" opens.
6. Select the "Modify/Upgrade" check box and click the "Next" button.
   The dialog "Please select the product features you want to install. If you deselect components which are already installed, they will be uninstalled." opens.
7. Select the products you want to "Modify/Upgrade" and click the "Next" button.

   Note

   Note that you cannot change the target directory because the existing installation is will be modified.

8. To continue the installation, read and accept all license agreements and click the "Next" button.
   The "Overview" dialog displays the installation settings.

9. Click the "Modify/Upgrade" button.
   The modification/upgrading of the existing installation is started and the "License transfer" dialog opens.

   Note

   If no license key is found during installation, you have the chance to transfer it to your PC. If you skip the license transfer, you can register it later with the Automation License Manager.

   After modifying/upgrading, you will see the message "Setup has successfully completed".

10. To complete the installation click the "Exit" button.

Result

TIA Portal Teamcenter Gateway has been installed again on your computer.

5.3.8 Repairing the installation of TIA Portal Teamcenter Gateway

You have the option to of repairing installed products by completely reinstalling them using the setup program.

Requirement

- TIA Portal V14 or higher and TIA Portal Teamcenter Gateway V14 or higher is installed on your PG/PC.
- Teamcenter Rich Application Client (RAC) or Teamcenter Client Communication System (TCCS) version 11.2 or higher is installed on your PG/PC.
- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- You have ended all active TIA Portal processes.

Procedure

To repair the TIA Portal Teamcenter Gateway option, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking on the "Start.exe" file.
   The dialog "Installation language" for selecting the installation language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read installation notes" or "Read product information" button.
   The help file containing the instructions opens.

5. Once you have read the instructions, close the help file and click the "Next" button.
   The dialog "A previous installation has been detected. Please select" opens.
6. Select the "Repair" check box and click the "Next" button.
7. Select the products you want to repair and click the "Next" button.
8. To continue the installation, read and accept all license agreements and click the "Next" button.
9. The "Overview" dialog opens.
10. Click the "Repair" button.
    This starts the repair of the existing installation.

    Note

    After repairing, you will see the message "Setup has successfully completed".

11. Click "Finish" to complete the installation.

Result

TIA Portal Teamcenter Gateway has been installed again on your computer.

5.3.9 Starting the uninstallation of TIA Portal Teamcenter Gateway

TIA Portal Teamcenter Gateway can be uninstalled using the installation data medium.

Requirement

- You have administrator privileges on your computer.
- You have ended all active TIA Portal processes.

Uninstalling TIA Portal Teamcenter Gateway using the installation medium

To uninstall TIA Portal Teamcenter Gateway, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking on the "Start.exe" file.
   The dialog for selecting the installation language opens.
3. Select the language in which you want to uninstall the dialogs of the setup program to be displayed and click "Next".
   The dialog "A previous installation has been detected. Please select" opens.
4. Select the "Uninstall" check box and click the "Next" button.
   The "Overview" dialog opens.

5. Click the "Uninstall" button.
   Uninstallation begins.

   Note

   After uninstalling, you will see the message "Setup has successfully completed".

6. Click "Finish" to complete the uninstallation.

Result

TIA Portal Teamcenter Gateway installed on your computer has been uninstalled.

5.4 Licenses for TIA Portal Teamcenter Gateway

5.4.1 Licenses

Licensing of TIA Portal Teamcenter Gateway

To work with the TIA Portal Teamcenter Gateway, you need a valid license key on every device on which you install TIA Portal Teamcenter Gateway. The license key is required even if the TIA Portal is installed on this device.

You can either include the license key in the installation or transfer it using the Automation License Manager after the installation. When you work with the TIA Portal in combination with TIA Portal Teamcenter Gateway, the presence of the valid license key is checked.

See also

Notes on licenses (Page 53)

5.4.2 Notes on licenses

Licenses for TIA Portal Teamcenter Gateway

The licenses for TIA Portal Teamcenter Gateway are usually supplied on an installation data medium and installed automatically during the installation process.

1. When you select the "Connect" command in the "Options > Teamcenter" menu in the TIA Portal, a check is made to see whether the license required for TIA Portal Teamcenter Gateway is available.
   The following license types are differentiated:

License key	Product name	Standard license type	License type	Runtime
SIFLTEAGAW01500	TIA Portal Teamcenter Gateway	Floating license	unlimited	unlimited
SITTTEAGAW1500	TIA Portal Teamcenter Gateway	Single license	Trial license	21 days
SIUPTEAGAW1500	TIA Portal Teamcenter Gateway	Upgrade license	---	unlimited
SIFLTEAGAW01400	TIA Portal Teamcenter Gateway	This license key is required for upgrading to newer versions.		

2. If a valid license key is not found, you will receive an error message.

3. If a valid license key is found, the "Connect to Teamcenter" dialog opens and you can establish a connection between TIA Portal and Teamcenter.
   This means the license key has been used.
4. When you select the "Disconnect" command in the "Options > Teamcenter" menu in the TIA Portal or when the TIA Portal is closed, the license key should be transferred back using the Automation License Manager.

Note

For additional information on how to manage your licenses with the Automation License Manager, refer to the documentation supplied with the Automation License Manager.

5.5 Supported Teamcenter server versions and Teamcenter client versions

5.5.1 Supported Teamcenter server versions and Teamcenter client versions

The following combinations of Teamcenter server versions and Teamcenter client versions are supported by TIA Portal Teamcenter Gateway.

Table 5-1 Supported Teamcenter server versions and Teamcenter client versions

TIA Portal Teamcenter Gateway	Teamcenter client (RAC or TCCS)	Teamcenter server
V14.0	11.2	11.2
V14 SP1	11.2	11.2
V14 SP1 Update 1	11.2	11.2
V15.0	11.2	11.2
V15.0	11.3	11.3
V15.1	11.2	11.2
V15.1	11.3	11.3
V15.1	11.4	11.4
V15.1	11.5	11.5

Note

Supported Teamcenter server versions and Teamcenter client versions

Only the combinations listed in the table are supported by TIA Portal Teamcenter Gateway.

All other combinations of Teamcenter server versions and Teamcenter client versions (RAC, TCCS) were **not tested** and are **not supported** by TIA Portal Teamcenter Gateway .

5.6 Basics of working with TIA Portal Teamcenter Gateway

Introduction

You can do the following with TIA Portal Teamcenter Gateway:

- Establish a connection between TIA Portal and Teamcenter.
- Perform operations in TIA Portal specific to Teamcenter.
- You can add TIA Portal projects that are connected to Teamcenter to the multiuser server as multiuser server projects.

TIA Portal Teamcenter Gateway together with TIA Portal offers the following advantages:

- Saving TIA Portal projects or global libraries in Teamcenter
- When saving TIA Portal projects or global libraries in Teamcenter you can assign them item types or derived types.
- You can save changes to TIA Portal projects or global libraries connected to Teamcenter.
- Revising TIA Portal projects or global libraries connected to Teamcenter.
- Global libraries or TIA Portal projects that are saved in Teamcenter can be opened in "Read only" or "Write" mode.
- You can save changes that you have made to multiuser server projects as a new version (dataset version) to Teamcenter.

5.7 Setting the TIA Portal Teamcenter Gateway cache path

Introduction

When you work with TIA Portal Teamcenter Gateway, you can set the "TIA Portal Teamcenter Gateway Cache path" in the settings for the TIA Portal.

Requirement

You have installed the TIA Portal V14 and higher and TIA Portal Teamcenter Gateway.

Setting the Teamcenter cache path

To set the "TIA Portal Teamcenter Gateway cache path", follow these steps:

1. In the TIA Portal under "Options" > "Settings" > "General" > "Teamcenter Gateway", click the "Browse" button in the "TIA Portal Teamcenter Gateway cache path" area.
   The "Search folder" dialog is displayed.
2. Select the corresponding folder and click the "OK" button to confirm the selection.
   The "TIA Portal Teamcenter Gateway cache path" dialog is displayed.
   – If you click "OK", the selected folder will be displayed in the text field.
   – If you click "Cancel", this action will be canceled.

   Alternatively, you can "manually" enter the absolute directory path in the text field.

Note

Scenarios in which two TIA Portal instances are using the same Teamcenter cache path are not supported.

5.8 Confirming check-in when closing the project/library

Check-in (automatic check-in.)

When you close a TIA Portal project or a global library stored in Teamcenter, you get a prompt to check in the TIA Portal project or global library. The display of the prompt depends on whether the "Confirm check-in when closing project/library" check box is enabled or disabled. The "Confirm check-in when closing the project/library" check box is disabled by default.

Procedure

To disable or enable check-in of a TIA Portal project or global library, follow these steps:

1. Under "Options" > "Settings" > "General" > "Teamcenter Gateway" > select the "Confirm check-in when closing project/library" check box in the "Check in" area.

- By default, the "Confirm check-in when closing the project/library" check box is disabled, this causes the TIA Portal project or global library to be checked in without prior prompting.
- If the "Confirm check-in when closing the project / library" check box is selected, a message is displayed:
  - Clicking "Yes" causes the TIA Portal project or the global library to be checked in.
  - Clicking "No" causes the TIA Portal project or the global library to not be checked in.

Result

The TIA Portal project or global library is checked in.

5.9 Establishing and terminating a connection between the TIA Portal and Teamcenter

5.9.1 Establishing a connection between the TIA Portal and Teamcenter

To work with TIA Portal Teamcenter Gateway, you need to log on to a specific Teamcenter server with your valid Teamcenter user logon.

Requirements

The following requires that:

- You have started TIA Portal.

Procedure

To establish a connection between TIA Portal and Teamcenter, proceed as follows:

1. Select the "Connect" command in the "Options > Teamcenter" menu of the main menu in TIA Portal.
2. The "Connect to Teamcenter" dialog box opens. Enter the following logon information:
   - User name: Enter your "Teamcenter user name".
   - Password: Enter your "Teamcenter password".
     Enter your Teamcenter logon information in the "User name" and "Password" fields.
   - Group: Enter your specific Teamcenter group, which is to be used for the current application session.
     If you leave this field 'empty', the default settings in Teamcenter are applied.
   - Role: Enter your specific Teamcenter role, which is to be used for the current application session.
     If you leave this field 'empty', the default settings in Teamcenter are applied.
   - Host: Enter the server name in the following format:
     http://<server name>:<port>
   - Instance: Enter the name of the "Teamcenter instance" (e.g. tc).
3. Click "OK" to establish the connection to Teamcenter.
   The "Establishing connection..." dialog is displayed while the connection is being set up. This background process cannot be canceled.
   Once the connection to Teamcenter is established, a dialog opens indicating the successful connection.
4. Confirm this dialog by clicking "OK".

5. When an authentication fails, a dialog is displayed with possible error messages. This error message contains information on troubleshooting.
   Possible errors:
   – The combination of user name and password is not valid.

   Select the "Connect" command again in the "Options > Teamcenter" menu of the main menu in TIA Portal. Correct the errors in the "Connect to Teamcenter" dialog.
6. The "Additional properties" of the project / library item types are read while the connection is being established. When an error occurs while reading the "Additional properties", a dialog is displayed with possible error messages.
7. Confirm this dialog by clicking "OK".

Note

The "Connect" command is deactivated when there is a successful establishment of the connection between TIA Portal and Teamcenter.

The "Disconnect" command is deactivated when there is a failed establishment of the connection between TIA Portal and Teamcenter.

License check

Note

When you select the "Connect" command in the "Options > Teamcenter" menu of the main menu in TIA Portal, a check is made to see whether the appropriate and valid licenses for TIA Portal Teamcenter Gateway are available.

You can find additional information on appropriate and valid licenses in the section "Licenses for TIA Portal Teamcenter Gateway" under Notes on licenses (Page 53).

Result

The connection between TIA Portal and Teamcenter has been established.

Establishing a connection as Teamcenter member of two different Teamcenter user groups or user roles

Note

The current user groups or user role cannot be changed during an active application session.

To change your user group or user role, follow these steps:

1. End the current application session between TIA Portal and Teamcenter with the menu "Options > Teamcenter" and the command "Disconnect".
2. Then create a new connection between TIA Portal and Teamcenter in the main menu of TIA Portal with the menu "Options > Teamcenter" and the command "Connect".
3. Make the required settings in the "Connect to Teamcenter" dialog.

5.9.2 Terminating the connection between the TIA Portal and Teamcenter.

Procedure

To terminate the connection between TIA Portal and Teamcenter, proceed as follows:

1. Select the "Disconnect" command in the "Options > Teamcenter" menu of the main menu in TIA Portal.

Result

The connection between TIA Portal and Teamcenter has been closed. No additional Teamcenter-specific operations can be performed.

5.9.3 Display connection status between TIA Portal and Teamcenter

Displaying the connection status

The following options are possible to show the connection status between TIA Portal and Teamcenter:

- A successful establishment of the connection between TIA Portal and Teamcenter is indicated in the status bar of TIA Portal by the Teamcenter icon on the bottom right. The Teamcenter icon is displayed in color as soon as the connection has been established. If you hover the mouse cursor over the Teamcenter icon for a time, a tooltip is displayed showing the connection information in plain text.
- A failed establishment of the connection between TIA Portal and Teamcenter is indicated in the status bar of TIA Portal by the Teamcenter icon on the bottom right. The Teamcenter symbol is displayed in black-and-white if no connection was established.
  If you hover the mouse cursor over the Teamcenter icon for a time, a tooltip is displayed showing the connection information in plain text.
- If the Teamcenter server is not activated for UTF-8, this is indicated in the status bar of the TIA Portal by the following Teamcenter icon on the bottom right.

Displaying the connection status in the "Info" tab.

The successful establishment of the connection between TIA Portal and the Teamcenter is displayed in the following format:

- http://hostname [User name] - Group / Role
  Example: http://WIN-MLMCU5LQTP6:8080 [User name] - Group / Role

In the "Info" tab, a failed establishment of the connection between TIA Portal and the Teamcenter is indicated in the following format:

- There is no connection to Teamcenter.

Displaying the connection status in the TIA Portal status bar

In the status bar of TIA Portal, the successful establishment of the connection between TIA Portal and Teamcenter is indicated in the following format:

- http://hostname [User name] - Group / Role
  Example: http://WIN-MLMCU5LQTP6:8080 [User name] - Group / Role

In the status bar of TIA Portal, the failed establishment of the connection between TIA Portal and Teamcenter is indicated in the following format:

- There is no connection to Teamcenter.

Note

The "Connect" command is deactivated when there is a successful establishment of the connection between TIA Portal and the Teamcenter.

The "Disconnect" command is deactivated when there is a failed establishment of the connection between TIA Portal and the Teamcenter.

5.10 Difference between TIA Portal projects and global libraries

Introduction

TIA Portal projects and global libraries are identified by the name of the dataset and the file extension in Teamcenter.

Name of the dataset

Note that:

- When saving a TIA Portal project with the "Save as new item" and "Save as new revision" Teamcenter operations in Teamcenter, the name of the TIA Portal project is used as dataset name (e.g. "TIAProjectDataset").
- When saving a global library with the "Save as new item" and "Save as new revision" Teamcenter operations in Teamcenter, the name of the global library is used as dataset name (e.g. "TIALibraryDataset").

File extensions

Note that:

- When saving a TIA Portal project with the "Save as new item" and "Save as new revision" Teamcenter operations, the TIA Portal project is saved with the file extension *.zap* in Teamcenter.
- When saving a global library with the "Save as new item" and "Save as new revision" Teamcenter operations, the global library is saved with the file extension *.zal* in Teamcenter.

5.10.1 Copying different dataset types to a new revision

Save TIA Portal projects and global libraries with the "Save as new revision..." Teamcenter operation in TIA Portal Teamcenter Gateway

You use the "Save as new revision..." Teamcenter operation in TIA Portal Teamcenter Gateway to specify during saving which attached datasets (non-TIA datasets) are transferred from the old revision to the new one.

This behavior depends on the copy rules "Deep Copy Rules".

Additional information on the copy rules "Deep Copy Rules" is available in the "Teamcenter 11 – Business Modeler IDE" operating manual in the section "Deep Copy Rules".

Working with the "Teamcenter Deep Copy Rule" function

The "Teamcenter Deep Copy Rule" function supports the following actions when you use the "Save as new revision..." Teamcenter operation:

- CopyAsObject
- CopyAsReference
- NoCopy
- RelatedToLatest
- ReviseAndReleateToLatest
- SystemCopy
- ReviseObject

5.11 Managing TIA Portal projects with TIA Portal Teamcenter Gateway

5.11.1 Saving a TIA Portal project as a new item in Teamcenter

Requirements

The following requires that:

- There is an active connection between TIA Portal and Teamcenter.
- Multiple selection of several projects is not supported.

Procedure

To save a TIA Portal project for the first time in Teamcenter, proceed as follows:

1. Open the TIA Portal project in the TIA Portal that you want to save as a new item in Teamcenter.
2. In the "Project tree" view, select the TIA Portal project to save to Teamcenter.
3. Right-click on the TIA Portal project and select the command "Teamcenter > Save as new item...".
   The "Properties of the new item" dialog opens.
   The "General" tab is displayed by default.
4. Enter a name in the "Name" field in the "General" tab or accept the suggested data. The name of the selected TIA Portal project is displayed by default.
   A maximum of 94 characters can be entered.
5. In the "General" tab in the field "Item type" select an Item type for the TIA Portal project that you want save as new item in Teamcenter.
   – Select a derived type in the "Item type" field.
     The "TIA Project" type is selected by default.
     A drop-down list can be edited if derived types are available.
6. Assign an already valid or random Teamcenter item ID / revision ID to the "Item ID / Revision ID" field in the "General" tab.
   If you leave the field "Item ID / Revision ID" 'empty', the next possible "item ID / revision ID" is assigned automatically by Teamcenter.

7. When you click "Assign", Teamcenter automatically assigns the next possible item ID / revision ID to the selected project.
   Result:
   - The assigned item ID / revision ID is shown in the respective text field.
   - The text fields cannot be edited and the "Assign" button is disabled.

   Note
   Adaptation of the item types

   The adaptation of the item types has the following effects on the "Item ID / Revision ID" fields and on the "Assign" button:
   1. the IDs assigned to the "item ID / revision ID" are reset.
   2. the "Assign" button is set to operable.

8. Optional: In the "Teamcenter project" field in the "General" tab, select the Teamcenter project to which you want to assign the TIA Portal project.
   - Click the "Browse" (...) button.
     The "Select projects" dialog opens.
   - Select the corresponding projects and click the "OK" button to confirm each selection.
     Clicking "OK" displays the selected Teamcenter projects in the "Teamcenter project" field separated by commas.
     If you leave the "Teamcenter project" field 'empty', the TIA Portal project is not assigned to a Teamcenter project.
9. Optional: In the "Teamcenter folder" field in the "General" tab, select the folder to which the newly created item is to be assigned.
   - Click the "Browse" (...) button.
     The "Select Teamcenter folder" dialog opens.
     The default for this folder is *"Home*".
     All newly created items are saved by default in your personal *"Home"* directory.
   - Select the "*Home*" folder, click the "OK" button to confirm the selection.
     Clicking "OK" displays the selected folder in the text field.
     Clicking "Cancel" causes the operation to be aborted.
10. Optional: If necessary, add a comment in the "Comment" field in the "General" tab.
    A maximum of 240 characters can be entered.
11. Confirm your entry with "OK".
    The TIA Portal project is saved in the specified folder structure under the specified name in Teamcenter.
    Once the TIA Portal project has been saved in Teamcenter, the "The project was successfully saved in Teamcenter" dialog opens.
    - Confirm this dialog by clicking "OK".
      An overlay symbol in the "Project tree" of TIA Portal indicates that TIA Portal project is connected to Teamcenter.
12. Clicking "Cancel" causes the "Properties of the new item" dialog to close.

Result

Once the TIA Portal project has been saved in Teamcenter, a new item including the item revision with the assigned item ID / revision ID is available in the "Home" folder. The dataset of the TIA Portal project is stored in Teamcenter as zap file under the corresponding revision ID.

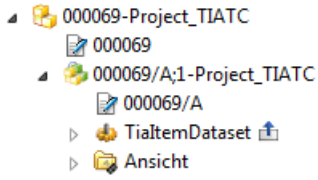

The associated dataset is "checked out" and marked with the symbol . You can see the symbol in the folder view of Teamcenter.

The following Teamcenter operations for working with the TIA Portal project are available to you in the TIA Portal:

- Save as new revision....
- Save as new item....
- Check in.
- Undo checkout.

See also

New revision of a TIA Portal project connected to Teamcenter (Page 75)

Check in TIA Portal project (Page 72)

Undo checkout of a TIA Portal project (Page 81)

Save changes to a TIA Portal project connected to Teamcenter (Page 74)

"General" tab in the "Save as new item" dialog - (TIA Portal projects) (Page 68)

"Additional properties" tab in the "Save as new revision..." dialog - (TIA Portal projects) (Page 68)

"General" tab in the "Save as new item" dialog - (TIA Portal projects) (Page 67)

"Additional properties" tab in the "Save as new item" dialog - (TIA Portal projects) (Page 69)

Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects) (Page 70)

5.11.1.1 "General" tab in the "Save as new item" dialog - (TIA Portal projects)

In the "General" tab of the "Save as new item" dialog, you define the properties of a TIA Portal project that you want to save in Teamcenter.

See also

"General" tab in the "Save as new item" dialog - (TIA Portal projects) (Page 68)

5.11.1.2 "Additional properties" tab in the "Save as new revision..." dialog - (TIA Portal projects)

In the "Additional properties" tab, you edit the values of the uploaded additional properties of a TIA Portal project.

In the "Additional properties" tab, you have the option of entering uploaded item types or derived types. The content of the "Additional properties" tab depends on the item types that are used, the derived types or the properties of the inherited types.

See also

"Additional properties" tab in the "Save as new item" dialog - (TIA Portal projects) (Page 69)

Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects) (Page 70)

5.11.1.3 "General" tab in the "Save as new item" dialog - (TIA Portal projects)

Structure of the "General" tab in the "Save as new item" dialog

The view of the "General" tab consists of the following fields and buttons:

Field	Description
Name:	Entry field for the name under which you want to save the TIA Portal project as new item in Teamcenter. The name of the selected TIA Portal project is displayed by default.
Item type:	Entry field for the item type of the TIA Portal project. The TIA Project type is selected by default.
Item ID / revision ID:	Entry field for the item ID / revision ID which you want to assign to the new item.
Teamcenter project:	Selection box for selecting the Teamcenter project to which you want to assign the new item.
Teamcenter folder:	Selection box for selecting the Teamcenter folder in which you want to save the new item.
Comment:	Entry field for a comment.

Button	Description
Drop-down list	You can select and add derived types by clicking on the drop-down list. The drop-down list can only be edited when derived types are defined.
Assign	By clicking this button you automatically assign an item ID / revision ID to the TIA Portal project.
"Browse" (...) - (Teamcenter project)	By clicking this button, the dialog "Select projects" is opened.
"Browse" (...) - (Teamcenter folder)	By clicking this button, the dialog "Select Teamcenter folder" is opened.
OK	You confirm your entries by clicking the "OK" button. The dialog is closed.
Cancel	To exit the dialog, click the "Cancel" button. Your entries are not applied.

See also

"General" tab in the "Save as new item" dialog - (TIA Portal projects) (Page 67)

5.11.1.4 "Additional properties" tab in the "Save as new item" dialog - (TIA Portal projects)

Structure of the "Additional properties" tab in the "Save as new item" dialog

The information shown in tabular format in the "Additional properties" tab depends on:

- Which item types or derived types you have selected.
- Which properties the selected item types and derived types have inherited, and which properties are mapped in the attribute mapping file.

Properties of the "Additional properties" tab

The properties of item types and derived types are displayed in read/write mode in the "Additional properties" tab.

The view of the "Additional properties" tab consists of the following columns and buttons:

Column	Meaning
Name	Name of the additional properties.
Type	Type of the additional properties.
Value	Value of additional properties.

Button	Meaning
Drop-down list	You can add obligatory attributes by clicking on the drop-down list.
OK	You confirm your entries by clicking the "OK" button. The dialog is closed.
Cancel	To exit the dialog, click the "Cancel" button. Your entries are not applied.

Note on the "OK" button in the "Additional properties" tab in the "Save as new item" dialog.

The following rules apply to the operation of the "OK" button in the "Additional properties" tab in the "Save as new item" dialog.

- If the required obligatory attributes are not filled, the "OK" button is greyed out and not available.
- The "OK" button is operable when no obligatory attributes are available or when all obligatory attributes are filled.

See also

"Additional properties" tab in the "Save as new revision..." dialog - (TIA Portal projects) (Page 68)

Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects) (Page 70)

5.11.1.5 Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects)

The attributes of a TIA Portal project that are upload to the Teamcenter server are displayed in read/write mode in the tabular view of the "Additional Properties" tab.

Requirements for editing the values of uploaded attributes of a TIA Portal project

The requirements for editing the values of the uploaded attributes of a TIA Portal project are as follows:

- the attributes of the selected TIA Portal project item type are uploaded.
- the attributes of the selected item revision of the TIA Portal project item type are uploaded.
- the uploaded attributes of the item master form are connected with the TIA Portal project item type.
- the uploaded attributes of the item revision master form are connected with the TIA Portal project item revision.

Note

Note the following:

Only the scenarios described above as of TIA Portal Teamcenter Gateway V14SP1 are supported. All other scenarios are ignored.

Note the following:

- If no attributes are uploaded to the Teamcenter server, the following information is displayed on the "Additional properties" tab:
  No "Additional properties" were uploaded.
- If it is not possible to read uploaded attributes during connection establishment with TIA Portal Teamcenter Gateway, the following information is displayed in the "Additional properties" tab:
  No "Additional properties" were uploaded.

When using the same attribute in the "General" tab and in the "Additional properties" tab

Note the following:

When using the same attribute in the "General" tab and in the "Additional properties" tab, only the entry of the attribute from the "General" tab is always saved in Teamcenter.

Multiple use of attributes in the "Additional properties" tab in the "Save as a new item" dialog

Note the following:

When the same attribute is used multiple times in the "Additional properties" tab in the "Save as a new item" dialog, only the first entry of the attribute from the table view is always saved in Teamcenter.

Use of special characters for 'Char' data type

The following describes the naming conventions for the 'Char' data type, which must be observed for "Save as new item" .

Note the following:

- Use only ASCII values 32-126 in the input boxes of the 'Char' data type in the "Additional properties" tab in the "Save as new item" dialog.

See also

"Additional properties" tab in the "Save as new revision..." dialog - (TIA Portal projects) (Page 68)

"Additional properties" tab in the "Save as new item" dialog - (TIA Portal projects) (Page 69)

5.11.2 Check out TIA Portal project

To exclusively reserve one or more TIA Portal projects for editing, you can check out the TIA Portal projects. The TIA Portal projects are locked for other users as a result.

Requirements

The following requires that

- The TIA Portal project is connected to Teamcenter.
- The TIA Portal project is opened in "Read only" mode.
- There is no support for multiple selection of several TIA Portal projects.

Procedure

To edit a TIA portal project, you need to check it out as follows:

1. In the "Project tree" view, select the respective TIA Portal project.
2. Right-click on the TIA Portal project and select the command "Teamcenter > Check out".

Note

Check out TIA Portal project

Note: If the TIA Portal project was changed locally or the dataset version is older than the currently available dataset version in Teamcenter, the "Confirm checkout" dialog opens.

1. Clicking the "Yes" button overwrites the currently available dataset version.
2. Clicking the "No" button opens the currently available dataset version as local copy.
3. Clicking the "Cancel" button cancels the checkout.

If you continue the "Checkout", the dataset version and the corresponding revision of the TIA Portal project are checked out of Teamcenter. The dataset version is locked for editing by another user.

Result

The TIA Portal project is opened again in "Write" mode.

The dataset version of the TIA Portal project is checked out and marked with the following symbol in Teamcenter.

The dataset version is locked for editing by another user.

Result

The following Teamcenter operations for working with the TIA Portal project are available to you in the TIA Portal after a successful "Checkout":

- Save as new revision....
- Save as new item....
- Check in.
- Undo checkout.

See also

New revision of a TIA Portal project connected to Teamcenter (Page 75)

Saving a TIA Portal project as a new item in Teamcenter (Page 65)

Undo checkout of a TIA Portal project (Page 81)

Check in TIA Portal project (Page 72)

5.11.3 Check in TIA Portal project

TIA Portal projects that you have saved in Teamcenter are in the "checked out" state. These TIA Portal projects are locked for other users as a result.

Requirements

The following requires that:

- The TIA Portal project is connected to Teamcenter.
- The dataset of the corresponding revision of the TIA Portal project has been checked out by you.
- There is no support for multiple selection of several TIA Portal projects.

Procedure

To revoke the edit lock of the TIA Portal project, follow these steps:

1. In the "Project tree" view, select the respective TIA Portal project.
2. Right-click on the TIA Portal project and select the command "Teamcenter > Check in".

Automatic check-in

Automatic check-in takes place when the TIA Portal project is closed in the TIA Portal under the following conditions:

- If the "VersionInfo.xml file" is available for a TIA Portal project.
- If the TIA Portal project is opened in "Write" mode.

Automatic check-in does **not** take place when the TIA Portal project is closed in the TIA Portal under the following conditions:

- If the "VersionInfo.xml file" is not available for a TIA Portal project.
- If the TIA Portal project is opened in "Read/Write" mode.
- If the TIA Portal project is checked out in RAC.

Result

The TIA Portal project is opened again in "Read" mode.

The TIA Portal project has been checked into Teamcenter and the following symbol is removed from the folder view of Teamcenter.

The edit lock of the TIA Portal project is revoked.

The following Teamcenter operations for working with the TIA Portal project are available to you in the TIA Portal:

- Save as new revision....
- Save as new item....
- Check out.

See also

New revision of a TIA Portal project connected to Teamcenter (Page 75)

Saving a TIA Portal project as a new item in Teamcenter (Page 65)

Check out TIA Portal project (Page 71)

5.11.4 Save changes to a TIA Portal project connected to Teamcenter

Requirements

The following requires that

- The TIA Portal is connected to Teamcenter.
- The dataset version differs from the local copy.
- The selected TIA Portal project is connected to Teamcenter and changes have been made.
- There is no support for multiple selection of several TIA Portal projects.

Note

Note the following:

- When there is no connection between the TIA Portal and Teamcenter, the "Teamcenter > Save" command is disabled.

Procedure

To save changes to a TIA Portal project connected to Teamcenter, proceed as follows:

1. In the "Project tree" view, select the project.
2. Right-click on the TIA Portal project and select the command "Teamcenter > Save".
   The new dataset version of the TIA Portal project is exported in Teamcenter as a zap file under the respective revision ID.

Restrictions

The following restrictions must be observed:

The number of dataset versions stored in Teamcenter can be configured by a Teamcenter administrator (default = 3). When the maximum is reached, the oldest dataset version is deleted and the new dataset version created.

Result

A new dataset version of the TIA Portal project has been created in Teamcenter.

The following Teamcenter operations for working with the TIA Portal project are available to you in the TIA Portal after "Save":

- Save as new revision....
- Save as new item....
- Check in.
- Undo checkout.
- Export....

See also

Save changes to a global library connected to Teamcenter (Page 126)

New revision of a TIA Portal project connected to Teamcenter (Page 75)

Saving a TIA Portal project as a new item in Teamcenter (Page 65)

Check in TIA Portal project (Page 72)

Undo checkout of a TIA Portal project (Page 81)

5.11.5 New revision of a TIA Portal project connected to Teamcenter

Requirements

The following requires that

- The TIA Portal project is connected to Teamcenter.
- There is no support for multiple selection of several TIA Portal projects.

Procedure

To create a new revision of a TIA Portal project saved in Teamcenter, proceed as follows:

1. In the "Project tree" view, select the respective TIA Portal project.
2. Right-click on the TIA Portal project and select the command "Teamcenter > Save as new revision...".
   With this function, the currently opened TIA Portal project is used as the basis for creating the new revision.
   - The "Save as new revision" dialog opens.
   - The "General" tab is displayed by default.
   - The name of the selected TIA Portal project is displayed in read-only mode in the "Name" field (e.g. t4tia_object_name).
   - The "item ID" of the selected TIA Portal project is displayed in read-only mode in the "Item ID" field.

3. Optional: Assign a valid Teamcenter revision ID to the TIA Portal project in the "General" tab in the "Revision ID" field.
   A maximum of 32 characters can be entered.
4. When you click the "Assign" button, the Teamcenter automatically assigns the next possible revision ID to the TIA Portal project.
   Result:
   – The assigned revision ID is displayed in the "Revision ID" field.
   – The "Revision ID" field cannot be edited.
   – The "Assign" button is disabled.

   If you leave the field 'empty', the next possible revision ID is assigned automatically by Teamcenter.
5. Optional: If necessary, add a comment in the "Comment" field in the "General" tab.
   A maximum of 240 characters can be entered.
   Comments from the previous version are taken over by default. You can edit the comments from the previous version or leave them unchanged.
6. Confirm your entry with "OK".
   The TIA Portal project is saved in the specified folder structure under the specified name in Teamcenter.
   Once the TIA Portal project has been saved in the Teamcenter, the "The project was successfully saved in the Teamcenter" dialog opens.
   – Confirm this dialog by clicking "OK".
     An overlay symbol in the "Project tree" of TIA Portal indicates that TIA Portal project is connected to Teamcenter.
7. Clicking "Cancel" closes the "Save as new revision..." dialog.

Result

The new revision of the TIA Portal project is created in Teamcenter and displayed as a new object below the last revision.

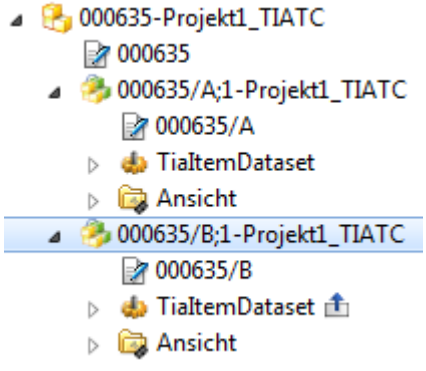

When revising the TIA Portal project, note that the dataset of the currently open TIA Portal project must be transferred.

The dataset of the TIA Portal project which belongs to the new revision is checked out and marked with the following symbol in Teamcenter.

The dataset of the older revision is checked in, the following symbol is removed in Teamcenter.

You can see the symbol in the folder view of Teamcenter.

The older revision of the TIA Portal project is closed in TIA Portal. The new revision of the TIA Portal project is opened in "Write" mode in TIA Portal.

The following Teamcenter operations for working with TIA Portal projects are available after "Save as new revision...".

- Save as new revision....
- Save as new item....
- Check in.
- Undo checkout.

See also

Creating new revision of a global library connected to Teamcenter (Page 127)

Check in TIA Portal project (Page 72)

Undo checkout of a TIA Portal project (Page 81)

Saving a TIA Portal project as a new item in Teamcenter (Page 65)

"General" tab in the "Save as new revision..." dialog - (TIA Portal projects) (Page 77)

Structure of "General" tab in the "Save as new revision" dialog - (TIA Portal projects) (Page 78)

5.11.5.1 "General" tab in the "Save as new revision..." dialog - (TIA Portal projects)

In the "General" tab of the "Save as new revision" dialog, you define the properties of a TIA Portal project that you want to save in Teamcenter.

See also

Structure of "General" tab in the "Save as new revision" dialog - (TIA Portal projects) (Page 78)

5.11.5.2 "Additional properties" tab in the "Save as new revision..." dialog - (TIA Portal projects)

In the "Additional properties" tab, you edit the values of the uploaded additional properties of TIA Portal projects that you want to save as new revision in Teamcenter.

In the "Additional properties" tab, you have the option of entering uploaded item types or derived types. The content of the "Additional properties" tab depends on the item types that are used, the derived types or the properties of the inherited types.

See also

Structure of "Additional properties" tab in the "Save as new revision" dialog - (TIA Portal projects) (Page 78)

Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects) (Page 79)

5.11.5.3 Structure of "General" tab in the "Save as new revision" dialog - (TIA Portal projects)

Structure of the "General" tab in the "Save as new revision..." dialog

The view of the "General" tab consists of the following fields and buttons:

Field	Description
Name (read only):	The name of the selected global library is displayed in read-only mode in the "Name" field (e.g. t4tia_object_name).
Item ID: (read only)/	The "item ID" of the selected global library is displayed in read-only mode in the "Item ID" field.
Revision ID:	Entry field for a valid Teamcenter revision ID which you want to assign to the new revision.
Comment:	Entry field for a comment.

Button	Description
Assign	By clicking this button you automatically assign a revision ID to the TIA Portal project.
OK	You confirm your entries by clicking the "OK" button. The dialog is closed.
Cancel	To exit the dialog, click the "Cancel" button. Your entries are not applied.

See also

"General" tab in the "Save as new revision..." dialog - (TIA Portal projects) (Page 77)

5.11.5.4 Structure of "Additional properties" tab in the "Save as new revision" dialog - (TIA Portal projects)

Structure of the "Additional properties" tab in the "Save as new revision" dialog

The information shown in tabular format in the "Additional properties" tab depends on:

- Which item types or derived types you have selected.
- Which properties the selected item types and derived types have inherited.

Properties of the "Additional properties" tab

The properties of item types and derived types are displayed in read/write mode in the "Additional properties" tab.

The view of the "Additional properties" tab consists of the following columns and buttons:

Column	Meaning
Name	Name of the additional properties.
Type	Type of the additional properties.
Value	Value of additional properties.

Button	Meaning
Drop-down list	You can add obligatory attributes by clicking on the drop-down list.
OK	You confirm your entries by clicking the "OK" button. The dialog is closed.
Cancel	To exit the dialog, click the "Cancel" button. Your entries are not applied.

Note on the "OK" button in the "Additional properties" tab in the "Save as new revision" dialog.

The following rules apply to the operation of the "OK" button in the "Additional properties" tab in the "Save as new revision" dialog:

- If the required obligatory attributes are not filled, the "OK" button is greyed out and not available.
- The "OK" button is operable when no obligatory attributes are available or when all obligatory attributes are filled.

See also

Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects) (Page 79)

"Additional properties" tab in the "Save as new revision..." dialog - (TIA Portal projects) (Page 77)

5.11.5.5 Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects)

The attributes of a TIA Portal project that are upload to the Teamcenter server are displayed in read/write mode in the tabular view of the "Additional Properties" tab.

Requirements for editing the values of uploaded attributes of a TIA Portal project

The requirements for editing the values of the uploaded attributes of a TIA Portal project are as follows:

- the attributes of the selected TIA Portal project item type are uploaded.
- The attributes of the selected item revision of the TIA Portal project item type are uploaded.

- the uploaded attributes of the item master form are connected with the TIA Portal project item type.
- the uploaded attributes of the item revision master form are connected with the TIA Portal project item revision.

Note

Note that

only the above named attributes are supported as of TIA Portal Teamcenter Gateway V14SP1. All other attributes are ignored.

Note the following:

- If the reading of uploaded attributes during connection establishment with TIA Portal Teamcenter Gateway failed, the following information is displayed on the "Additional properties" tab:
  No "Additional properties" were uploaded.
- If no attributes are uploaded to the Teamcenter (server), the following information is displayed on the "Additional properties" tab:
  No "Additional properties" were uploaded.

When using the same attribute in the "General" tab and in the "Additional properties" tab

Note the following:

When using the same attribute in the "General" tab and in the "Additional properties" tab, only the entry of the attribute from the "General" tab is always saved in Teamcenter.

Multiple use of attributes in the "Additional properties" tab in the "Save as a new revision" dialog

Note the following:

When the same attribute is used multiple times in the "Additional properties" tab in the "Save as a new revision" dialog, only the first entry of the attribute from the table view is always saved in Teamcenter.

Use of special characters for 'Char' data type

The following describes the naming conventions for the 'Char' data type, which must be observed for "Save as new revision".

Note the following:

- Only use ASCII values 32-126 in the input boxes of the 'char' data type in the "Additional properties" tab in the "Save as new revision" dialog.

See also

Structure of "Additional properties" tab in the "Save as new revision" dialog - (TIA Portal projects) (Page 78)

"Additional properties" tab in the "Save as new revision..." dialog - (TIA Portal projects) (Page 77)

5.11.6 Undo checkout of a TIA Portal project

Undo checkout

To discard any changes made to a checked-out TIA Portal project, you have the possibility to undo the checkout.

Requirements

The following requires that

- The TIA Portal project is connected to Teamcenter.
- The TIA Portal project is open in "Write" mode.
- There is no support for multiple selection of several TIA Portal projects.

Procedure

1. In the "Project tree" view, select the respective TIA Portal project.
2. Right-click on the TIA Portal project and select the command "Teamcenter > Undo checkout".
   The "Undo checkout" dialog box opens.
3. Click "OK" to undo the checkout.
4. Click "Cancel" if you want to cancel "Undo checkout".

Note

All changes made since the last checkout are discarded.

Result

The checkout of the TIA Portal project has been reversed. The symbol is removed from the folder view of Teamcenter.

The edit lock of the TIA Portal project is revoked.

The following Teamcenter operations for working with the TIA Portal project are available to you in the TIA Portal after "Undo checkout":

- Save as new revision....
- Save as new item....
- Check out.

See also

New revision of a TIA Portal project connected to Teamcenter (Page 75)

Saving a TIA Portal project as a new item in Teamcenter (Page 65)

Check out TIA Portal project (Page 71)

5.11.7 Export a TIA Portal project

The Teamcenter shortcut menus "Save" and "Export..." are interdependent. The status of the "Save" and "Export..." shortcut menus depends on the following conditions.

Teamcenter shortcut menu "Export..."

The Teamcenter shortcut menu "Export..." is activated if

- the selected TIA Portal project is connected to Teamcenter and no changes have been made.

Procedure

To export a TIA Portal project, follow these steps:

1. Select the relevant TIA Portal project in the "Project tree" view.
2. Right-click the TIA Portal project and select the "Teamcenter > Export" command.

Note

Note the following:

If the "Teamcenter > Export..." command is not enabled for the TIA Portal project, changes have been made to the TIA Portal project. Save the TIA Portal project with the "Teamcenter > Save" command. The "Teamcenter > Export..." command is then active.

Result

The "Teamcenter > Export..." command removes the Teamcenter meta information from the TIA Portal project and saves the modified TIA Portal project to the folder you selected.

Note

Note the following:

An existing TIA Portal project cannot be exported to the same location under the same name.

Note the following:

1. The Teamcenter shortcut menu "Save" is deactivated if no connection exists between the TIA Portal and Teamcenter.
2. The Teamcenter shortcut menu "Export..." is deactivated if changes have been made in the selected TIA Portal project.

5.11.7.1 "Info" tab in the "Export..." - (TIA Portal projects) dialog

The "Info" tab displays the information about the storage location of the TIA Portal project after export.

5.11.8 Saving a TIA Portal project from the TIA Portal to an item that exists in Teamcenter

5.11.8.1 Checks when using the function "Save to item..." - (TIA Portal projects)

When the "Save to item..." function is used, Teamcenter performs the following checks:

1. With the item ID / revision ID, a check is made to determine whether the item ID / revision ID exists in Teamcenter.
   If the item ID / revision ID does not exist in Teamcenter, Teamcenter displays a dialog with a corresponding message.
   – Click "OK" or close the dialog.
2. With the item ID / revision ID, a check is carried out to determine whether the item type in Teamcenter matches the "T4TiaProject" item type and its derived types.
   If the item type in Teamcenter does not match the "T4TiaProject" item type and its derived types, Teamcenter displays a dialog with a corresponding message.
   – Click "OK" or close the dialog.
3. With the item ID / revision ID, a check is made to determine whether a valid TIA dataset exists in the item and the associated revision.
   If no valid TIA dataset exists in the item and the associated revision, Teamcenter displays a dialog with a corresponding message.
   – Click "OK" or close the dialog.

4. With the item ID / revision ID, a check is made to determine whether the name of the TIA Portal object (name of the TIA Portal project) matches the name of the dataset.
   If the name of the TIA Portal project does not match the name of the dataset, Teamcenter displays a dialog with a corresponding message.
   – Click "OK" or close the dialog.
5. With the item ID / revision ID, a check is made to determine whether the dataset of the item and the associated revision in Teamcenter is checked out by another user.
   If the dataset of the item and the associated revision in Teamcenter is checked out by another user, Teamcenter displays a dialog with a corresponding message.
   – Click "OK" or close the dialog.
6. With the item ID / revision ID, a check is carried out to determine whether the version of the TIA Portal project differs from the Teamcenter version.
   If the version of the TIA Portal project differs from the version in Teamcenter, the "Critical message" dialog prompts you to decide whether or not you want to overwrite the version in Teamcenter with your changes.
   – Click "Yes" to overwrite the version of the project in Teamcenter with the new or modified version.
   – Click "No" to cancel the "Save to item..." operation.
7. With the item ID / revision ID, a check is carried out to determine:
   – Whether the user has the required access rights.
   – Which release status the item ID / revision ID has.
     If the user does not have the required access rights and the item ID / revision ID has the "Released" release status, a dialog with a corresponding message is displayed by Teamcenter.
   – Click "OK" or close the dialog.

5.11.8.2 Saving a TIA Portal project from the TIA Portal to an item that exists in Teamcenter

With the "Save to item..." function, you connect a TIA Portal project that was processed outside the cache server of Teamcenter to Teamcenter again.

When changes are made to the TIA Portal project outside the cache server of Teamcenter, the "versionInfo.xml" file is updated.

Requirement

- An active connection exists between the TIA Portal and Teamcenter.

Procedure

To connect a TIA Portal project to Teamcenter again, proceed as follows:

1. Select the TIA Portal project in the "Project tree" view.
2. Right-click on the TIA Portal project and select the command "Teamcenter > Save to item...".
   The "Save to item..." dialog opens.

3. The name of the selected TIA Portal project is displayed in the "Name" field by default.
   If necessary, change the name in the "Name" field.
4. The item ID associated with the TIA Portal project from the "versionInfo.xml" file is displayed in the "Item ID" field by default.
   - Optional: If necessary, change the existing item ID in the "Item ID" field.
   - If the "versionInfo.xml" file does not exist, the "Item ID" field is empty.
     Assign an already valid or any Teamcenter ID to the "Item ID" field.
     A maximum of 128 characters can be entered.
5. The revision ID associated with the TIA Portal project from the "versionInfo.xml" file is displayed in the "Revision ID" field by default.
   - Optional: If necessary, change the existing revision ID in the "Revision ID" field.
   - If the "versionInfo.xml" file does not exist, the "Revision ID" field is empty.
     Assign an already valid or any Teamcenter revision ID to the "Revision ID" field.
     A maximum of 32 characters can be entered.
6. Confirm your entry with "OK".
   The TIA Portal project is saved under the specified item ID / revision ID in Teamcenter.
7. If you click "Cancel", the "Save to item..." operation will be canceled.

Note

Note the following:

The Item ID / Revision ID fields are obligatory fields. The "OK" button can only be edited if a value is entered in the Item ID / Revision ID fields.

Preliminary check for the "Save to item..." operation

When the "Save to item..." function is used, Teamcenter performs the following checks:

For more details, see: Checks when using the function "Save to item..."

The following is checked during the save operation:

1. Whether the Teamcenter-saved data version and associated revision ID are checked out by the currently logged-on user or another user.
2. Whether the dataset version was checked out by a different user.
   If the dataset version is not checked out by another user, the dataset version is implicitly checked out when the "Save to item..." function is executed.

3. Whether the TIA Portal project was exported and archived with the file extension "*.zap".
   No "versionInfo.xml" file is permitted to be present for the archived project.
   If errors occur while exporting and archiving, a corresponding message is displayed indicating that the export and archive operation failed and that the export and archive operation was canceled.
   To confirm the message, follow these steps:
   – Click "OK" or close the message:

   Note

   Remedy following cancelation of export and archive operation:

   1. Create a new dataset version in Teamcenter.
   2. Upload the archived project file to the dataset.
      The dataset is checked out in the process.
      If the creation of the item revision is not possible, an error message is displayed indicating that the creation of the item revision failed and that the export and archive operation was canceled.
      To confirm the error message, follow these steps:
      – Click "OK" or close the message.
        A copy of the TIA Portal project is not saved in the TcCache.
        A TcCache path for the TIA Portal project is not created.

4. Transfer the project changes to Teamcenter.
   If the transfer of project changes to Teamcenter is not possible, a corresponding error message is displayed.
   To confirm the message, follow these steps:
   – Click "OK" or close the message.

Actions on "Save to item..."

Once all checks are finished, the following actions are performed:

1. The TIA Portal project is archived and saved under the specified item ID / revision ID in Teamcenter.
2. The associated dataset has status "checked out" and is marked with the following symbol.

   You can see the symbol in the folder view of Teamcenter.
3. After completion of these actions, the "The project was successfully saved to Teamcenter" dialog opens.
   – Confirm this dialog by clicking "OK".

"Rollback" functions

1. If the export and archive operation failed, the older version of the "versionInfo.xml" file is used.
2. If the creation of a new item revision failed, remove the exported and archived files from the local hard drive.
3. If the creation of a new item revision failed, remove all the created objects (dataset version) in Teamcenter. The original version of the "versionInfo.xml" file is then accessed in the TIA Portal project. As soon as the TIA Portal project has been saved in Teamcenter with the "Save to item" function, a dialog with a corresponding message opens. Confirm this dialog by clicking "OK".

Result

After the server project view is reopened, the following Teamcenter shortcut menu entries are available:

- Save as new item...
- Save as new revision...
- Save to item...
- Check in.
- Undo checkout.
- Export...

5.11.8.3 Structure of dialog "Save to item..."

In the "Save to item" dialog, you define the properties of a TIA Portal project that you want to save in Teamcenter.

Structure of dialog "Save to item..."

The view of the "Save to item..." dialog consists of the following fields and buttons.

Field	Description
Name:	Input box for the name under which you want to save the TIA Portal project in Teamcenter. The name of the selected TIA Portal project is displayed by default. The "Name" field cannot be edited.
Item ID / Revision ID:	Input box for the item ID / revision ID that you want to assign to the TIA Portal project.

Button	Note
OK	You confirm your entries by clicking the "OK" button. The dialog closes.
Cancel	You exit the dialog by clicking the "Cancel" button. Your entries are not applied.

5.11.9 Save the TIA Portal project which is saved in the Teamcenter cache folder to the associated item again.

Using the the function "Save to item...", save a TIA Portal project from the Teamcenter cache folder to the associated item again.

Requirement

- An active connection exists between the TIA Portal and Teamcenter.

Procedure

To connect a TIA Portal project to Teamcenter again, proceed as follows:

1. Select the TIA Portal project in the "Project tree" view.
2. Right-click on the TIA Portal project and select the command "Teamcenter > Save to item...". The "Save to item..." dialog opens.
3. The name of the selected TIA Portal project is displayed in the "Name" field by default. The "Name" field cannot be edited.
4. Assign an item ID to the "Item ID" field.
   – A maximum of 128 characters can be entered.
5. Enter a revision ID in the "Revision ID" field.
   – A maximum of 32 characters can be entered.
6. Confirm your entry with "OK".
   The TIA Portal project is saved under the specified item ID / revision ID in Teamcenter.
7. If you click "Cancel", the "Save to item..." operation is canceled.

Note

The Item ID / Revision ID fields are obligatory fields. The "OK" button can only be edited if a value is displayed in the Item ID / Revision ID fields.

Actions on "Save to item..."

When the "Save to item..." function is executed, the following actions are performed:

1. The TIA Portal project is archived and saved under the specified item ID / revision ID in Teamcenter.
2. The associated dataset has the status "Checked out" and is marked with the following symbol.

   You can see the symbol in the folder view of Teamcenter.
3. After completion of the actions, the "The TIA Portal project was successfully saved to Teamcenter" dialog opens.
   – Confirm this dialog by clicking "OK".

Result

The following Teamcenter operations for working with TIA Portal projects are available to you in TIA Portal:

- Save as new item...
- Save as new revision...
- Save to item...
- Check in.
- Undo checkout.
- Export...

5.11.10 Opening a TIA Portal project connected to Teamcenter Gateway as reference project

The "Open as reference project" option allows you to open a TIA Portal project as a reference project in the "Open project from Teamcenter" dialog. The check box "Open as reference project" is cleared by default.

Procedure

To open a TIA Portal project in the "Open project from Teamcenter" dialog as a reference project, follow these steps:

1. Select the "Open project from Teamcenter..." command in the "Project" menu in the project view.
   The "Open project from Teamcenter" dialog opens.
   You can find additional information about the "Open project from Teamcenter" dialog in: Finding and opening TIA Portal projects or global libraries with TIA Portal Teamcenter Gateway (Page 141).
2. Select a corresponding TIA Portal project from the search result.
   You can find additional information on this in: Opening TIA Portal projects from the search result (Page 144).
3. Select the "Open as reference project" check box.
   The "Open as reference project" check box is cleared by default.

   Note

   If you select the "Open as reference project" check box, the "Check out" check box is disabled.

4. Click "Open".

Result

When the TIA Portal project is opened as a reference project, the following actions are triggered:

1. A temporary "RF" folder (reference project folder) is created parallel to the Teamcenter cache folder.
   The "RF" folder (reference project folder) has to be 2 characters long and parallel to the TcCache folder.
   Example:

Current Teamcenter cache folder	<TIA default projectlocation>\TC\<ItemID>\<RevID>\ProjectFolder
"RF" folder (reference project folder)	<TIA default projectlocation>\RF\<ItemID>\<RevID>\ProjectFolder

2. The same rules apply to the path length of the temporary "RF" folder (reference project folder) as to the path length of the Teamcenter cache folder.
3. The selected TIA Portal project is downloaded to the temporary "RF" folder (reference project folder) and opened in the "Reference projects" pane.
   – The TIA Portal project is displayed with the name extension and the Teamcenter overlay symbol in the "Reference projects" pane.
   – The TIA Portal project is opened, irrespective of whether it is checked out by the current user or by another user.
4. If you open an older version of a TIA Portal project as the reference project, you receive a corresponding message to update the TIA Portal project.

5. When you update the TIA Portal project, the new name with the name extension and Teamcenter overlay symbol is displayed in the "Reference projects" pane.
6. If errors occur while downloading a TIA Portal project from Teamcenter, a corresponding message with detailed error information is displayed.
7. Recommendation: Use the comparison editor to compare the required objects of the TIA Portal project from the project navigation with the reference project from the reference project pane.
8. If you close a TIA Portal project in the reference projects pane, the revision folder containing the TIA Portal project is deleted from the temporary reference project folder. At the same time, the TIA Portal project is deleted from the temporary folder.
9. Note that the TIA Portal project is not deleted from the temporary folder when you close the TIA Portal.

Note

Always open the "Reference projects" pane first before you open the TIA Portal project from Teamcenter.

5.12 Workflows between the TIA Portal Teamcenter Gateway and TIA Portal multiuser engineering - TIA Portal projects

5.12.1 Saving a TIA Portal project connected to Teamcenter as a multiuser server project

This section contains information on how to save a TIA Portal project that has been saved to Teamcenter as a new multiuser server project.

Requirement

The following requires that:

- An active connection exists between the TIA Portal and Teamcenter.
- You have installed a multiuser server that manages your multiuser projects.
- The "Automatic check-in" check box has been selected in the TIA Portal under "Tools" > "Settings" > "General" > "Teamcenter Gateway".

Procedure

To save a TIA Portal project that has been saved to Teamcenter as a new multiuser server project, follow these steps:

1. Open the relevant TIA Portal project.
2. To do so, select the "Open project from Teamcenter..." command in the "Project" menu of the project view.
   The "Open project from Teamcenter" dialog is displayed.
3. Search in the "Open project from Teamcenter" dialog for the relevant TIA Portal project to open it.
4. To add the TIA Portal project to the multiuser server, click the menu command "Project > Multiuser > Manage multiuser server projects".
   While the TIA Portal project is being added to the multiuser server, the TIA Portal project in the project tree is closed. If the "Automatic check-in" check box has been selected in the TIA Portal under "Tools" > "Settings" > "General" > "Teamcenter Gateway", the TIA Portal project is automatically checked in in Teamcenter. The automatic check-in is logged by a message in a dialog.

5.12.1.1 Server project view of a multiuser server project

Server project view of a multiuser server project

The name of the TIA Portal project added to the multiuser server is displayed in the server project view. The displayed name consists of:

- Project name
- Name extension

- Item ID / Revision ID
- Teamcenter overlay symbol
  The Teamcenter overlay symbol indicates that the TIA Portal project was saved on the multiuser server and is linked to Teamcenter.

Commands in the shortcut menu of the server project view

The following Teamcenter operations for working with the multiuser server project are available to you in the server project view:

- Save.
- Save as new revision....
- Save as new item....
- Check out (locked only).
- Check in (unlocked only).

5.12.1.2 Creating a local session of a specific multiuser server project

This section contains information on how to create a local session from a TIA Portal project that has been saved to Teamcenter and added to the multiuser server.

Requirement

The following requires that:

- The TIA Portal project is saved to Teamcenter and added to the multi-user server.

Creating a local session of a specific multiuser server project

To create a local session from a TIA Portal project that has been saved to Teamcenter and added to the multiuser server, follow these steps:

- To create a local session of a specific multiuser server project, use the standard multiuser workflow.
  You can find additional information in the online help for "Use multiuser engineering" under "Establish local session".

The name of the TIA Portal project is displayed in the local session and consists of:

- Project name
- Name extension
- Item ID / Revision ID
- Teamcenter overlay symbol
  The Teamcenter overlay symbol indicates that the TIA Portal project is linked to Teamcenter.

Saving a local session

You save changes that you have made on the multiuser server in a local session with the standard multiuser workflow.

You can find additional information in the online help for "Use multiuser engineering" under "Establish local session".

Operations specific to Teamcenter must be carried out in the server project view.

5.12.1.3 Example of how to handle inconsistencies between TIA Portal projects in Teamcenter and in the multiuser server

This section contains examples on how you eliminate inconsistencies between TIA Portal projects in Teamcenter and in the multiuser server.

Examples

Examples of how inconsistencies can arise between TIA Portal projects in Teamcenter and in the multiuser server are explained below.

Example 1:

- Dataset Version A in Teamcenter was updated with the last revision of the multiuser server with the "Teamcenter > Save" command.
- Dataset Version A was changed outside of the multiuser server. As a result of this, the new Dataset Version B was created in Teamcenter.

Example 2:

- Dataset Version A in Teamcenter was updated with the last version of the multiuser server with the "Teamcenter > Save" command.
- Changes that were made on the multiuser server in a local session were checked in on the multiuser server in parallel.

Remedy

Update the TIA Portal project in the multiuser server by manually comparing the contents of Dataset Version B. To compare Dataset Version B of the TIA Portal project in Teamcenter with Dataset Version B in the multiuser server, you must open the dataset version in the reference project view.

See also: AUTOHOTSPOT

5.12.2 Check out (locked only)

To reserve one or more TIA Portal projects that you have added to the multiuser server for editing exclusively by you, you can check out the TIA Portal projects in the multiuser server view. This locks the TIA Portal projects in Teamcenter for other users.

Requirement

The following requires that:

- You have opened the TIA Portal project with the "Open project from Teamcenter..." command in the "Project" menu of the project view.
- You have added the TIA Portal project to the multiuser server.
- You have installed a multiuser server that manages your multiuser projects.

Procedure

1. Select the relevant "TIA Portal project" in the "Project tree" view.
2. Right-click the TIA Portal project and select the "Teamcenter > Check out (locked only)" command.

   Note

   Check out (locked only) causes the following:

   - The dataset version of the TIA Portal project is checked out but not downloaded to the multiuser server.
   - The message "The project is already checked out by another user: <User name>." is displayed:
     - if the TIA Portal project is already checked out by another user,
     - and the status of the TIA Portal project in the multiuser server view is not to be changed.
   - A check is made to determine whether the dataset version is already checked out by the current user. If so, the message "The project was already checked out by the current user" appears.

   Note

   Check out (locked only) checks to determine whether the current dataset version of the TIA Portal project in the multiuser server differs from that in the Teamcenter server.

   - If the current dataset version of the TIA Portal project in the multiuser server differs from that in the Teamcenter server, the following message is displayed:
     "The dataset version of the TIA Portal project in the multiuser server is different to that in the Teamcenter server. Do you still want to check out the dataset version?"
     - If you click "Yes", the dataset version will be checked out.
     - If you click "No", the "Check out (locked only)" operation will be canceled.
   - If the current dataset version of the TIA Portal project in the multiuser server does not differ from that in the Teamcenter server, the dataset version of the TIA Portal project will be checked out and locked for editing by other users.

   The message "The project was successfully checked out from Teamcenter" is displayed.
3. Click "OK" to confirm the check out.

Result

The dataset version of the TIA Portal project is checked out.

The dataset version is locked for editing by other users. This prevents changes being added to the dataset version while the TIA Portal project on the multiuser server is being edited.

5.12.3 Check in (unlocked only)

To release a TIA Portal project that you have added to the multiuser server for processing, you must check in the TIA Portal project in the multiuser server view. This releases the TIA Portal project for other users in Teamcenter.

Requirement

The following requires that

- The TIA Portal project has already been saved in Teamcenter and added to the multiuser server.
- You have installed a multiuser server that manages your multiuser projects.
- The dataset is checked out.

Procedure

To revoke the edit lock of the TIA Portal project that you have added to the multiuser server, follow these steps:

- Select the relevant "TIA Portal project" in the server project view.
- Right-click the TIA Portal project and select the "Teamcenter > Check in (unlocked only)" command.

Note

Use of menu command "Check in (unlocked only)":

- When menu command "Check in (unlocked only)" is used and the dataset version is not checked out, the following warning is displayed:
  - "The project is not checked out by the current user".

Note

Use of menu command "Check in (unlocked only)":

- When menu command "Check in (unlocked only)" is used, data is not transferred from the multiuser server to Teamcenter.
- When menu command "Check in (unlocked only)" is used, there is no consistency check between the dataset version ID of the multiuser server project and the dataset version ID in Teamcenter.

5.12.4 Saving changes to a multiuser server project to Teamcenter

This section describes how you save changes to a TIA Portal project (multiuser server project) that is already saved in Teamcenter and added to the multiuser server back to Teamcenter.

Use the "Save" command in the Teamcenter shortcut menu in the server project view to save the multiuser server project back to Teamcenter as a new version (dataset version).

The "Save" command in the "Teamcenter" shortcut menu in the server project view is activated provided the following requirements are met.

Requirement

- An active connection exists between the TIA Portal and Teamcenter.
- The revision ID of the multiuser server project saved in the "versionInfo.xml" file is different from the current revision ID of the multiuser server project. This means that the TIA Portal project in Teamcenter and the TIA Portal project in the multiuser server are different.
  If the revision IDs are different, the shortcut menu can be used for saving.

Reasons and indicator for different revision IDs

The following causes for different revision IDs of the TIA Portal project in Teamcenter and the TIA Portal project in the multiuser server project are possible:

- The revision ID of the TIA Portal project in the multiuser server project was changed, e.g. by checking in your "local session" or another "local session".
- Changes to a TIA Portal project were saved in the server project view with the "Save changes" option.

Note

Note that each multiuser server project has its own revision ID, which is a component of the "versionInfo.xml" file.

If a multiuser server project is edited and the changes are not saved, the object is marked with a flag in the server project view and the "Save" command is activated in the Teamcenter shortcut menu.

Preliminary check for the "Save" operation

The following is checked during the save operation:

1. Whether the dataset version saved in Teamcenter and the associated revision are checked out by the currently logged-on user.
   If the dataset version and the associated revision are not checked out by the currently logged-on user, a message appears and the "Save" operation is not performed.
2. Whether the dataset version ID of the multiuser server project differs from the dataset version ID in Teamcenter.
   If the dataset version ID of the multiuser server project differs from the dataset version ID in Teamcenter, the "Critical message" dialog prompts you to decide whether you want to overwrite the dataset version ID in Teamcenter with your changes.
   The following message is displayed in the "Critical message" dialog: "The dataset version ID of the project in Teamcenter differs from the dataset version ID of the project you want to save in Teamcenter. Do you want to overwrite the dataset version ID in Teamcenter with your changes?"
   Click "Yes" to overwrite the dataset version ID in Teamcenter with the dataset version ID from the server project view.
   Click "No" to cancel the "Save" operation.

You can find additional information under Examples of the data synchronization between the various dataset versions in the Teamcenter cache and in the multiuser server project (Page 100).

Actions on "Save"

Once all preliminary checks are finished, the following actions can be performed:

1. The "versionInfo.xml" file is updated and saved as a component of the multiuser server project.
   A loading symbol is displayed during the save operation.
   As a result of the automatic check-in, you are prompted to check in the dataset. You can also decline the prompt to check in the dataset.
   After the multiuser server project is saved in Teamcenter, the server project view is closed.
   The newly created item of the multiuser server project is opened in a local session.
2. The multiuser server project is exported and archived with file extension "*.zap". The exported and archived project is a normal TIA Portal project. No "versionInfo.xml" file is permitted to be present for the archived project.
   If errors occur while exporting and archiving, an error message is displayed indicating that the export and archive operation failed and that the export and archive operation was canceled.
   To confirm the error message, follow these steps:
   – Click "OK" or close the message.

   Note

   Remedy following cancelation of export and archive operation:
   1. Create a new dataset version in Teamcenter.
   2. Upload the archived project file to the dataset.
      The dataset is checked out in the process.
      If the creation of the item revision is not possible, an error message is displayed indicating that the creation of the item revision failed and that the export and archive operation was canceled.
      To confirm the error message, follow these steps:
      – Click "OK" or close the message.
        A copy of the TIA Portal project is not saved in the TcCache.
        A TcCache path for the TIA Portal project is not created.
3. Transfer the project changes from the server project view to the multiuser server.
   If the transfer of project changes from the server project view to the multiuser server is not possible, an error message is displayed.
   To confirm the error message, follow these steps:
   – Click "OK" or close the message.

The following rollback functions are available if one of the previously indicated actions was canceled by an error message:

1. If the export and archive operation of the multiuser server project failed, an older version of the "versionInfo.xml" file is accessed in the multiuser server project.
2. If the creation of a new item revision failed, remove the exported and archived files from the local hard drive.
3. If the creation of a new item revision failed, remove all the created objects (dataset version) in Teamcenter. The original version of the "versionInfo.xml" file is then accessed in the multiuser server project.

Recommendation

Perform an update of the local session in the multiuser server in order to update the objects of the local session with the current data of the multiuser server.

The name extension is displayed in the local session.

Shortcut menu entries

After the server project view is reopened, the following Teamcenter shortcut menu entries are available:

- Save
- Save as new item
- Save as new revision
- Check out (locked only)
- Check in (unlocked only)
- Export...

5.12.4.1 Examples of the data synchronization between the various dataset versions in the Teamcenter cache and in the multiuser server project

Examples of data synchronization

The following examples show dataset versions that are different in Teamcenter, the Teamcenter cache and the multiuser server.

Example 1

Example 1 shows the status of the dataset version of the TIA Portal project (Project 1) that was downloaded from the Teamcenter server and added to the multiuser server.

Storage location of the project	Project name	Dataset version
TcServer	Project1	1
TcCache	Project1	1
MUServer	Project1	1

"Teamcenter" shortcut menu, "Save" command in the multiuser server (data synchronization)

Use the "Save" command in the "Teamcenter" shortcut menu to save the multiuser server project (Project 1).

Storage location of the project	Project name	Dataset version
TcServer	Project1	2
TcCache	Project1	1
MUServer	Project1	2

Note

To update the dataset version in the TcCache, open the TIA Portal project using the "Open TIA Portal projects from the search result list" dialog.

Example 2

Example 2 shows the status of the dataset version of the TIA Portal project (Project 1) that was added to the multiuser server and changed via the project tree. To save changes you have made in the project tree, right-click the TIA Portal project and select the "Save" command in the "Teamcenter" shortcut menu. The "Save" command triggers the following:

- The dataset version in Teamcenter and the dataset version in the TcCache are updated.

Storage location of the project	Project name	Dataset version
TcServer	Project1	2
TcCache	Project1	2
MUServer	Project1	1

"Teamcenter" shortcut menu, "Save" command in the multiuser server (data synchronization)

You can use the "Save" command in the "Teamcenter" shortcut menu to synchronize the different dataset versions between the Teamcenter server and the multiuser server.

To do this, follow these steps:

- Save the TIA Portal project in Teamcenter. This creates a new dataset version.
  or
- Cancel the "Save" operation.
- Perform manual data synchronization.
- Repeat the save operation.
  The message indicating that the dataset versions are different is displayed again. If you continue with the "Save" operation, a new dataset version will be created.

Storage location of the project	Project name	Dataset version
TcServer	Project1	3
TcCache	Project1	2
MUServer	Project1	3

Example 3

When a TIA Portal project that was added to the multiuser server is changed (e.g. by another user) and then saved in the project tree with the "Save" command in the "Teamcenter" shortcut menu, the dataset versions in Teamcenter and the TcCache are updated.

Storage location of the project	Project name	Dataset version
TcServer	Project1	3
TcCache	Project1	2
MUServer	Project1	1

"Teamcenter" shortcut menu, "Save" command in the multiuser server (data synchronization)

You can use the "Save" command in the "Teamcenter" shortcut menu to synchronize the different dataset versions between the Teamcenter server and the multiuser server.

To do this, follow these steps:

- Save the TIA Portal project in Teamcenter. This creates a new dataset version.
  or
- Cancel the "Save" operation.
- Perform manual data synchronization.
- Repeat the save operation.
  The message indicating that the dataset versions are different is displayed again. If you continue with the "Save" operation, a new dataset version will be created.

Storage location of the project	Project name	Dataset version
TcServer	Project1	4
TcCache	Project1	2
MUServer	Project1	4

Example 4

Example 4 shows the status of a dataset version of a TIA Portal project that was changed in the local cache after it was added to the multiuser server, and the TcCache changes were not saved in Teamcenter.

Storage location of the project	Project name	Dataset version	Corrupted?
TcServer	Project1	1	-
TcCache	Project1	1	Yes
MUServer	Project1	1	-

Note

Note that the transfer of project changes from the local cache to the multiuser server is not possible.

Note that addition of a project with the same name to the multiuser server is not allowed.

For further support, contact your multiuser administrator.

5.12.5 Saving a multiuser server project as a new item in Teamcenter

This section describes how to save a multiuser server project as a new item in Teamcenter in the server project view. The "Teamcenter" shortcut menu must be active in the server project view for this. The "Teamcenter > Save as new item..." shortcut menu entry is activated in the server project view provided the following requirements are met.

Requirement

The following requires that:

- An active connection exists between the TIA Portal and Teamcenter.

  Note

  Note the following:

  The Teamcenter shortcut menu item "Save as new item " is activated for the following multiuser server projects:

  - Multiuser server projects that are saved in Teamcenter.
  - Multiuser server projects that are not yet saved in Teamcenter.

Procedure

To save a multiuser server project as a new item in Teamcenter in the server project view, follow these steps:

1. Select the relevant "TIA Portal project" in the server project view.
2. Right-click on the TIA Portal project and select the command "Teamcenter > Save as new item...".
   - The "Properties of the new item" dialog opens.
   - The "General" tab is displayed by default.
   - The name of the selected TIA Portal project is displayed in read-only mode in the "Name" field and cannot be changed.
     A maximum of 94 characters can be entered.

3. In the "Item type" field on the "General" tab, select an item type for the TIA Portal project that you want to save as a new item in Teamcenter.
   – Select a derived type in the "Item type" field.
     The "TIA Project" type is selected by default.
     The drop-down list can only be edited if derived types are available.
4. Assign an already valid or any Teamcenter item ID / revision ID to the "Item ID / Revision ID" field on the "General" tab.
   If you leave the field "Item ID / Revision ID" empty, the next possible "item ID / revision ID" is assigned automatically by Teamcenter.
5. When you click "Assign", Teamcenter automatically assigns the next possible item ID / revision ID to the selected project.
   Result:
   – The assigned item ID / revision ID is shown in the respective text field.
   – The text fields cannot be edited and the "Assign" button is disabled.

Note

Adaptation of the item types

The adaptation of the item types has the following effects on the "Item ID / Revision ID" fields and the "Assign" button:

1. The IDs assigned to the "Item ID / Revision ID" fields are reset.
2. The "Assign" button is set to operable.
3. If you intend to change the name of the TIA Portal project, you have to save the TIA Portal project in the project view using the shortcut menu command "Teamcenter > Save as new item...". See also: Saving a TIA Portal project as a new item in Teamcenter (Page 65).

Character length of the TIA Portal Teamcenter Gateway cache path:

Note

Note that the character length of the TIA Portal Teamcenter Gateway cache path, consisting of the project name and item ID/revision ID, is checked in the "General" tab. A character length of 143 characters must not be exceeded.

You can find more information about the "General" tab under "General" tab in the "Save as new item" dialog - (TIA Portal projects) (Page 68).

You can find more information about the "Additional information" tab under Structure of "Additional properties" tab in the "Save as new revision" dialog - (TIA Portal projects) (Page 78).

---

6. Optional: In the "Teamcenter project" field on the "General" tab, select the Teamcenter project to which you want to assign the TIA Portal project.
   - Click the "Browse" (...) button.
     The "Select projects" dialog opens.
   - Select the relevant Teamcenter projects and click the "OK" button to confirm each selection.
     When you click "OK", the selected Teamcenter projects are displayed in the "Teamcenter project" field separated by commas.
     If you leave the "Teamcenter project" field empty, the TIA Portal project will not be assigned to a Teamcenter project.
7. Optional: In the "Teamcenter folder" field on the "General" tab, select the folder to which the newly created item is to be assigned.
   - Click the "Browse" (...) button.
     The "Select Teamcenter folder" dialog opens.
     The default setting for this folder is *"Home*".
     All newly created items are saved by default in your personal *"Home"* directory.
   - Select the "*Home*" folder, and click the "OK" button to confirm the selection.
     If you click "OK", the selected folder will be displayed in the text field.
     If you click "Cancel", this action will be canceled.
8. Optional: If necessary, enter a comment in the "Comment" field on the "General" tab.
   A maximum of 240 characters can be entered.
9. Confirm your entry with "OK".
   The multiuser server project is saved in the specified folder structure under the specified name in Teamcenter.
   While the multiuser server project is saved in Teamcenter, the following actions are initiated.
   - The "versionInfo.xml" file is created and saved as a component of the multiuser server project.
   - The multiuser server project was exported and archived with file extension "*.zap". The exported and archived project is a normal TIA Portal project. No "versionInfo.xml" file is permitted to be present for the archived project.
     If errors occur while exporting and archiving, a message is displayed indicating that the export and archive operation failed and the save operation was canceled.
     To confirm the error message, follow these steps:
     - Click "OK" or close the error message

   As soon as the multiuser server project has been saved in Teamcenter, the "The project was successfully saved in Teamcenter" dialog opens.
10. Create a new item with your specific properties in Teamcenter and upload the archived project file to the dataset. During this process, the dataset must **not** be checked out.
    If the item cannot be created, a message is displayed indicating that the creation of the new item failed, and the save operation is canceled.
    To confirm the error message, follow these steps:
    - Click "OK" or close the error message.

11. If you do not intend to work in the multiuser server environment, the TIA Portal project must be opened with the "Open project from Teamcenter..." command in the "Project" menu of the project view.

    Note

    Note the following:

    - A copy of the TIA Portal project is not saved in the Teamcenter cache.
    - A Teamcenter cache path will not be created for the TIA Portal project.

12. Transfer the project changes from the server project view to the multiuser server.
    If the project changes cannot be transferred from the server project view to the multiuser server, an error message is displayed indicating that the transfer of project changes to the multiuser server failed and the save operation was stopped.
    To confirm the error message, follow these steps:

    – Click "OK" or close the error message.

"Rollback" functions

The following rollback functions are available if one of the previously described actions was canceled by an error message.

1. If the export and archive operation of the multiuser server project failed, an older version of the "versionInfo.xml" file is accessed in the multiuser server project.
2. If the creation of a new item failed, the exported and archived files must be removed from the local hard drive.
3. If the creation of a new item failed, you must remove all generated objects (dataset version) in Teamcenter in order to return to the original version of the "versionInfo.xml" file in the multiuser server project.

   Note

   Note the following:

   If you use the "Teamcenter > Save as new item..." command in the server project view for a multiuser server project that is not yet saved in Teamcenter, the "versionInfo.xml" file must be removed from the multiuser server project.

   As soon as the TIA Portal project has been saved in Teamcenter, the "The project was successfully saved in Teamcenter" dialog opens.

   – Confirm this dialog by clicking "OK".

Note the following:

1. A loading symbol is displayed during the save operation of the new item.
2. If you want to use the "Teamcenter > Save as new item" command in the server project view to create a new item from a multiuser server project that is saved in Teamcenter, you are prompted to check in the dataset. You can also decline the prompt to check in the dataset.
3. After the multiuser server project is saved in Teamcenter, the server project view is closed.
4. The newly created item of the multiuser server project is opened in a local session.

Recommendation

Perform an update in the local session in order to update the objects of the local session with the current data from the multiuser server.

The file extension and the new revision ID are displayed in the local session. The Teamcenter overlay symbol indicates that the newly created item of the multiuser server project was saved on the multiuser server and is linked to Teamcenter.

Note

After the server project view is reopened, the following Teamcenter shortcut menu entries are available:

- Save
- Save as new item...
- Save as new revision...
- Check out (locked only)
- Check in (unlocked only)

5.12.6 Saving the multiuser server project as a new revision in Teamcenter

This section describes how to save a multiuser server project as a new revision in Teamcenter in the server project view. The "Teamcenter" shortcut menu must be active in the server project view for this. The "Teamcenter > Save as new revision..." shortcut menu entry is activated in the server project view provided the following requirements are met.

Requirement

The following requires that:

- the TIA Portal project is connected to Teamcenter.
- the multiuser server project is saved in Teamcenter.

Procedure

To save a multiuser server project as a new revision in Teamcenter in the server project view, follow these steps:

1. Select the relevant "TIA Portal project" in the server project view.
2. Right-click the multiuser server project and select the "Teamcenter > Save as new revision..." command.
   With this function, the currently open multiuser server project is used as the basis for creating the new revision.
   - The "Save as new revision" dialog opens.
   - The "General" tab is displayed by default.
   - The name of the selected TIA Portal project is displayed in read-only mode in the "Name" field (e.g. t4tia_object_name).
   - The "item ID" of the selected TIA Portal project is displayed in read-only mode in the "Item ID" field.
3. Optional: Assign a valid Teamcenter revision ID to the TIA Portal project in the "Revision ID" field on the "General" tab.
   A maximum of 32 characters can be entered.
4. When you click the "Assign" button, the next possible revision ID is automatically assigned to the TIA Portal project by Teamcenter.
   Result:
   - The assigned revision ID is displayed in the "Revision ID" field.
   - The "Revision ID" field cannot be edited.
   - The "Assign" button is disabled.

   If you leave the field empty, the next possible revision ID is assigned automatically by Teamcenter.
   Character length of the TIA Portal Teamcenter Gateway cache path:

   Note

   Note that the character length of the TIA Portal Teamcenter Gateway cache path, consisting of the project name and item ID/revision ID, is checked in the "General" tab. A character length of 143 characters must not be exceeded.

   You can find more information about the "General" tab under Structure of "General" tab in the "Save as new revision" dialog - (TIA Portal projects) (Page 78).

   You can find more information about the "Additional information" tab under Structure of "Additional properties" tab in the "Save as new revision" dialog - (TIA Portal projects) (Page 78).

5. Optional: If necessary, enter a comment in the "Comment" field on the "General" tab.
   A maximum of 240 characters can be entered.
   Comments from the previous version are taken over by default. You can edit the comments from the previous version or leave them unchanged.

6. Confirm your entry with "OK".
   The multiuser server project is saved in the specified folder structure under the specified name in Teamcenter.
   While the multiuser server project is saved in Teamcenter, the following actions are initiated:
   – The "versionInfo.xml" file is created and saved as a component of the multiuser server project.
   – The multiuser server project was exported and archived with file extension "*.zap". The exported and archived project is a normal TIA Portal project. No "versionInfo.xml" file can be present for the archived project.
     If errors occur while exporting and archiving, a message is displayed indicating that the export and archive operation failed and the save operation was canceled.
     To confirm the error message, follow these steps:
     - Click "OK" or close the error message

   As soon as the multiuser server project has been saved in Teamcenter, the "The TIA Portal project was successfully saved in Teamcenter" dialog opens.
7. Create a new revision with your specific properties in Teamcenter and upload the archived project file to the dataset. During this process, the dataset must **not** be checked out.
   If the new revision cannot be created, a message is displayed indicating that the creation of the new revision failed, and the save operation is canceled.
   To confirm the error message, follow these steps:
   – Click "OK" or close the error message.
8. If you do not intend to work in the multiuser server environment, the TIA Portal project must be opened with the "Open project from Teamcenter..." command in the "Project" menu of the project view.

   Note

   Note the following:
   - A copy of the TIA Portal project is not saved in the Teamcenter cache.
   - A Teamcenter cache path will not be created for the TIA Portal project.

9. Transfer the project changes from the server project view to the multiuser server.
   If the project changes cannot be transferred from the server project view to the multiuser server, an error message is displayed indicating that the transfer of project changes to the multiuser server failed and the save operation was stopped.
   To confirm the error message, follow these steps:
   – Click "OK" or close the error message.

The following rollback functions are available if one of the previously described actions was canceled by an error message.

1. If the export and archive operation of the multiuser server project failed, an older version of the "versionInfo.xml" file is accessed in the multiuser server project.
2. If the creation of a new revision failed, the exported and archived files must be removed from the local hard drive.
3. If the creation of a new revision failed, you must remove all generated objects (dataset version) in Teamcenter in order to return to the original version of the "versionInfo.xml" file in the multiuser server project.

   Note

   Note the following:

   If you use the "Teamcenter > Save as new revision..." command in the server project view for a multiuser server project that is not yet saved in Teamcenter, the "versionInfo.xml" file must be removed from the multiuser server project.

   As soon as the TIA Portal project has been saved in Teamcenter, the "The project was successfully saved in Teamcenter" dialog opens.

   – Confirm this dialog by clicking "OK".

Note the following:

1. A loading symbol is displayed during the save operation of the new revision.
2. If you want to use the "Teamcenter > Save as new revision..." command in the server project view to create a new revision from a multiuser server project that is saved in Teamcenter, you are prompted to check in the dataset. You can also decline the prompt to check in the dataset.
3. After the multiuser server project is saved in Teamcenter, the server project view is closed.
4. The newly created revision of the multiuser server project is opened in a local session.
5. Perform an update in the local session in order to update the objects of the local session with the current data from the multiuser server.
   The file extension and the new revision ID are displayed in the local session. The Teamcenter overlay symbol indicates that the newly created revision of the multiuser server project was saved on the multiuser server and is linked to Teamcenter.

Note

After the server project view is reopened, the following Teamcenter shortcut menu entries are available:

- Save
- Save as new item...
- Save as new revision...
- Check out (locked only)
- Check in (unlocked only)

See also

New revision of a TIA Portal project connected to Teamcenter (Page 75)

5.12.7 Saving a TIA Portal project (multiuser project) from the TIA Portal to an item that exists in Teamcenter

5.12.7.1 Saving a TIA Portal project to an item that exists in Teamcenter in the server project view

With the "Save to item..." function, you connect a TIA Portal project (multiuser project) that was processed outside the cache server of Teamcenter to Teamcenter again.

When changes are made to the TIA Portal project (multiuser server project) outside the Teamcenter cache server, the "versionInfo.xml" file is updated.

Requirement

- An active connection exists between the TIA Portal and Teamcenter.
- You have installed a multiuser server that manages your multiuser projects.

Procedure

To connect a TIA Portal project in the server project view to Teamcenter again, proceed as follows:

1. Select the TIA Portal project in the "Project tree" view.
2. Right-click on the TIA Portal project and select the command "Teamcenter > Save to item...". The "Save to item..." dialog opens.
3. The name of the selected TIA Portal project is displayed in the "Name" field by default. If necessary, change the name in the "Name" field.
4. The item ID associated with the TIA Portal project from the "versionInfo.xml" file is displayed in the "Item ID" field by default.
   - Optional: If necessary, change the existing item ID in the "Item ID" field.
   - If the "versionInfo.xml" file does not exist, the "Item ID" field is empty.
     Assign an already valid or any Teamcenter ID to the "Item ID" field.
     A maximum of 128 characters can be entered.
5. The revision ID associated with the TIA Portal project from the "versionInfo.xml" file is displayed in the "Revision ID" field by default.
   - Optional: If necessary, change the existing revision ID in the "Revision ID" field.
   - If the "versionInfo.xml" file does not exist, the "Revision ID" field is empty.
     Assign an already valid or any Teamcenter revision ID to the "Revision ID" field.
     A maximum of 32 characters can be entered.

6. Confirm your entry with "OK".
   The TIA Portal project is saved under the specified item ID / revision ID in Teamcenter.
7. If you click "Cancel", the "Save to item..." operation will be canceled.

Note

Note the following:

The Item ID / Revision ID fields are obligatory fields. The "OK" button can only be edited if a value is entered in the Item ID / Revision ID fields.

Preliminary check for the "Save to item..." operation

When the "Save to item..." function is used, Teamcenter performs the following checks:

- The check for the existence of the item.
- The check of the release state of the data set.
- The check of the old data.

For more details, see: Checks when using the function "Save to item..." (Page 83)

The following is checked during the save operation:

1. Whether the dataset version saved in the multiuser server and the associated revision ID are checked out by the currently logged-on user or another user.
2. Whether the dataset version is checked out by a user.
   If the dataset version is not checked out by another user, the dataset version is implicitly checked out when the "Save to item..." function is executed.

3. Whether the multiuser server project was exported and archived with the file extension "*.zap". No "versionInfo.xml" file is permitted to be present for the archived project.
   If errors occur while exporting and archiving, a corresponding message is displayed indicating that the export and archive operation failed and that the export and archive operation was canceled.
   To confirm the message, follow these steps:
   – Click "OK" or close the message:

   Note

   Remedy following cancelation of export and archive operation:

   1. Create a new dataset version in Teamcenter.
   2. Upload the archived project file to the dataset.
      The dataset is checked out in the process.
      If the creation of the item revision is not possible, an error message is displayed indicating that the creation of the item revision failed and that the export and archive operation was canceled.
      To confirm the error message, follow these steps:
      – Click "OK" or close the message.
        A copy of the TIA Portal project is not saved in the TcCache.
        A TcCache path for the TIA Portal project is not created.

4. Transfer the project changes from the server project view to the multiuser server.
   If the transfer of project changes from the server project view to the multiuser server is not possible, a corresponding error message is displayed.
   To confirm the message, follow these steps:
   – Click "OK" or close the message.

Actions on "Save to item..."

Once all checks are finished, the following actions are performed:

1. The TIA Portal project is archived and saved under the specified item ID / revision ID in Teamcenter.
2. The associated dataset has status "checked out" and is marked with the following symbol.

   You can see the symbol in the folder view of Teamcenter.
3. After completion of these actions, the "The project was successfully saved to Teamcenter" dialog opens.
   – Confirm this dialog by clicking "OK".

"Rollback" functions

1. If the export and archive operation of the multiuser server project failed, an older version of the "versionInfo.xml" file is accessed in the multiuser server project.
2. If the creation of a new item revision failed, remove the exported and archived files from the local hard drive.
3. If the creation of a new item revision failed, remove all the created objects (dataset version) in Teamcenter. The original version of the "versionInfo.xml" file is then accessed in the multiuser server project.

Recommendation

Perform an update of the local session in the multiuser server in order to update the objects of the local session with the current data of the multiuser server.

The name extension is displayed in the local session.

The file extension and the new revision ID are displayed in the local session. The Teamcenter overlay symbol indicates that the newly created item of the multiuser server project was saved on the multiuser server and is linked to Teamcenter.

Result

After the server project view is reopened, the following Teamcenter shortcut menu entries are available:

- Save as new item...
- Save as new revision...
- Save to item...
- Check in.
- Check in (unlock only)
- Check out (locked only)
- Undo checkout.
- Export...

5.12.8 Opening a TIA Portal project connected to Teamcenter Gateway and added to the multiuser server as a reference project

5.12.8.1 Opening a TIA Portal project connected to Teamcenter Gateway and added to the multiuser server as a reference project

The "Open as reference project" option allows you to open a TIA Portal project (multiuser server project) as a reference project in the "Open project from Teamcenter" dialog. The check box "Open as reference project" is cleared by default.

Procedure

To open a TIA Portal project in the "Open project from Teamcenter" dialog as a reference project, follow these steps:

1. Select the "Open project from Teamcenter..." command in the "Project" menu in the project view.
   The "Open project from Teamcenter" dialog opens.
   You can find additional information about the "Open project from Teamcenter" dialog in: Finding and opening TIA Portal projects or global libraries with TIA Portal Teamcenter Gateway (Page 141).
2. Select the corresponding TIA Portal project from the search result.
   You can find additional information on this in: Opening TIA Portal projects from the search result (Page 144).
3. Select the "Open as reference project" check box.

   Note

   If you select the "Open as reference project" check box, the "Check out" check box is disabled.

4. Click "Open".

Result

When the TIA Portal project (multiuser server project) is opened as a reference project, the following actions are triggered:

1. A temporary "RF" folder (reference project folder) is created parallel to the Teamcenter cache folder.
   The "RF" folder (reference project folder) has to be 2 characters long and parallel to the TcCache folder.
   Example:

Current Teamcenter cache folder	<TIA default projectlocation>\TC\<ItemID>\<RevID>\ProjectFolder
"RF" folder (reference project folder)	<TIA default projectlocation>\RF\<ItemID>\<RevID>\ProjectFolder

2. The same rules apply to the path length of the temporary "RF" folder (reference project folder) as to the path length of the Teamcenter cache folder.
3. The selected TIA Portal project is downloaded to the temporary "RF" folder (reference project folder) and opened in the "Reference projects" pane.
   - The TIA Portal project is displayed with the name extension and the Teamcenter overlay symbol in the "Reference projects" pane.
   - The TIA Portal project is opened, irrespective of whether it is checked out by the current user or by another user.
4. If you open an older version of a TIA Portal project as the reference project, you receive a corresponding message to update the TIA Portal project.
5. When you update the TIA Portal project, the new name with the name extension and Teamcenter overlay symbol is displayed in the "Reference projects" pane.

6. If errors occur while downloading a TIA Portal project from Teamcenter, a corresponding message with detailed error information is displayed.
7. Recommendation: Use the comparison editor to compare the required objects of the TIA Portal project from the project navigation with the reference project from the reference project pane.
8. If you close the TIA Portal project in the "Reference projects" pane, the revision folder containing the TIA Portal project is deleted from the temporary reference project folder. At the same time, the TIA Portal project is deleted from the temporary folder.
9. Note that the TIA Portal project is not deleted from the temporary folder when you close the TIA Portal.

Note

Always open the "Reference projects" pane first before you open the TIA Portal project from Teamcenter.

5.13 Managing global libraries with TIA Portal Teamcenter Gateway

5.13.1 Saving a global library as a new item in Teamcenter

Requirement

The following requires that:

- An active connection exists between the TIA Portal and Teamcenter.
- Multiple selection of several global libraries is not supported.

Procedure

To save a global library for the first time in Teamcenter, proceed as follows:

1. Open the global library in TIA Portal that you want to save as a new item in Teamcenter.
2. Select the global library to save in Teamcenter from the "Global Library" pane.
3. Right-click on the global library and select the command "Teamcenter > Save as new item...".
   The "Save as new item" dialog opens.
   The "General" tab is displayed by default.
4. Enter a name in the "Name" field in the "General" tab or accept the suggested data. The name of the selected global library is displayed by default.
   A maximum of 94 characters can be entered.
5. In the "General" tab in the "Item type" field, select an item type for the global library that you want save as new item in Teamcenter.
   – Select a derived type in the "Item type" field.
     The "T4TiaProject" type is selected by default.
     The drop-down list can only be edited if derived types are available.
6. Assign an already valid or any Teamcenter item ID / revision ID to the "Item ID / Revision ID" field on the "General" tab.
   If you leave the field "Item ID / Revision ID" empty, the next possible "item ID / revision ID" is assigned automatically by Teamcenter.

7. When you click "Assign", Teamcenter automatically assigns the next possible item ID / revision ID to the selected library.
   Result:
   - The assigned item ID / revision ID is shown in the respective text field.
   - The text fields cannot be edited and the "Assign" button is disabled.

   Note
   Adaptation of the item types

   The adaptation of the item types has the following effects on the "Item ID / Revision ID" fields and on the "Assign" button:
   1. The IDs assigned to the "Item ID / Revision ID" fields are reset.
   2. The "Assign" button is set to operable.

8. Optional: In the "General" tab in the "Teamcenter Project" field, select the Teamcenter projects to which you want to assign the global library.
   - Click "Browse" (...).
     The "Select projects" dialog opens.
   - Select the relevant Teamcenter projects and click "OK" to confirm each selection.
     Clicking "OK" displays the selected Teamcenter projects in the text box separated by commas.
     If you leave the field 'empty', the global library is not assigned to a Teamcenter project.
9. Optional: In "Teamcenter folder" field of the "General" tab, select the folder to which the newly created item is to be assigned.
   - Click "Browse" (...).
     The "Select Teamcenter folder" dialog opens.
     The default setting for this folder is *"Home"*.
     All newly created items are saved by default in your personal *"Home"* directory.
   - Select the *"Home"* folder and click "OK" to confirm the selection.
     If you click "OK", the selected folder will be displayed in the text field.
     If you click "Cancel", this action will be canceled.
10. Optional: If necessary, enter a comment in the "Comment" field on the "General" tab.
    A maximum of 240 characters can be entered.
11. Confirm your entry with "OK".
    The global library is saved in the specified folder structure under the specified name in Teamcenter.
    Once the global library has been saved in Teamcenter, the "The global library was successfully saved in Teamcenter" dialog opens.
    - Confirm this dialog by clicking "OK".
      An overlay symbol in the "Global Library" pane of TIA Portal indicates that the global library is connected to Teamcenter.
12. Clicking "Cancel" causes the "Properties of the new item" dialog to close.

Result

Once the global library has been saved in Teamcenter, the new item including the item ID / revision ID is available in the *"Home*" folder. The dataset of the global library is stored in Teamcenter as a zip file under the respective revision ID.

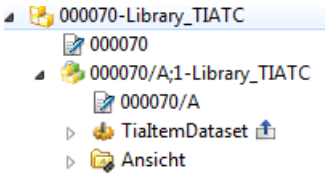

The associated dataset is "checked out" and marked with the symbol . You can see the symbol in the folder view of Teamcenter.

The following Teamcenter operations for working with global libraries are available to you in TIA Portal:

- Save as new revision....
- Save as new item....
- Check in.
- Undo checkout.

Global corporate libraries

Note

When saving global libraries, make sure that you do not save global library marked as "Corporate libraries" in Teamcenter. "Corporate libraries" are only buffered temporarily in "TC Cache" which can result in unspecified errors between TIA Portal and Teamcenter.

See also

Creating new revision of a global library connected to Teamcenter (Page 127)

Check in global library (Page 124)

Undo checkout of a global library (Page 132)

Save changes to a global library connected to Teamcenter (Page 126)

"General" tab in the "Save as new item" dialog - (global libraries) (Page 120)

Structure of the "Additional properties" tab in the "Save as new item" dialog (global libraries) (Page 120)

Structure of "General" tab in the "Save as new item" dialog - (global library) (Page 120)

Structure of the "Additional properties" tab in the "Save as new item" dialog (global library) (Page 121)

5.13.1.1 "General" tab in the "Save as new item" dialog - (global libraries)

In the "General" tab of the "Save as new item" dialog, you define the properties of a global library that you want to save in Teamcenter.

See also

Structure of "General" tab in the "Save as new item" dialog - (global library) (Page 120)

5.13.1.2 Structure of the "Additional properties" tab in the "Save as new item" dialog (global libraries)

In the "Additional properties" tab, you edit the values of the uploaded additional properties of a global library.

In the "Additional properties" tab, you have the option of editing uploaded item types or derived types. The content of the "Additional properties" tab depends on the item types that are used, the derived types or the properties of the inherited types.

See also

Structure of the "Additional properties" tab in the "Save as new item" dialog (global library) (Page 121)

Editing values in the tabular view of the "Additional properties" tab - (global libraries) (Page 122)

5.13.1.3 Structure of "General" tab in the "Save as new item" dialog - (global library)

Structure of the "General" tab in the "Save as new item" dialog

The view of the "General" tab consists of the following fields and buttons:

Field	Description
Name:	Entry field for the name under which you want to save the global library as new item in Teamcenter. The name of the selected global library is displayed by default.
Item type:	Entry field for the item type of the global library. The "T4TiaLibrary" type is selected by default.
Item ID / revision ID:	Entry field for the item ID / revision ID which you want to assign to the new item.
Teamcenter project:	Selection box for selecting the Teamcenter project to which you want to assign the new item.
Teamcenter folder:	Selection box for selecting the Teamcenter folder in which you want to save the new item.
Comment:	Entry field for a comment.

Button	Description
Drop-down list	You can select and add derived types by clicking on the drop-down list. The drop-down list can only be edited when derived types are defined.
Assign	By clicking this button you automatically assign an item ID / revision ID to the global library.
"Browse" (...) - (Teamcenter project)	By clicking this button, the dialog "Select projects" is opened.
"Browse" (...) - (Teamcenter folder)	By clicking this button, the dialog "Select Teamcenter folder" is opened.
OK	You confirm your entries by clicking the "OK" button. The dialog is closed.
Cancel	To exit the dialog, click the "Cancel" button. Your entries are not applied.

See also

"General" tab in the "Save as new item" dialog - (global libraries) (Page 120)

5.13.1.4 Structure of the "Additional properties" tab in the "Save as new item" dialog (global library)

Structure of the "Additional properties" tab in the "Save as new item" dialog

The information shown in tabular format in the "Additional properties" tab depends on:

- Which item type and which of its derived types you have selected.
- Which properties the selected item types and their derived types have inherited, and which properties are mapped in the attribute mapping file.

Properties of the "Additional properties" tab

The properties of item types and their derived types are displayed in read/write mode in the "Additional properties" tab.

The view of the "Additional properties" tab consists of the following columns and buttons:

Column	Meaning
Name	Name of the additional properties.
Type	Type of the additional properties.
Value	Value of additional properties.

Button	Meaning
Drop-down list	You can add obligatory attributes by clicking on the drop-down list.
OK	You confirm your entries by clicking the "OK" button. The dialog is closed.
Cancel	To exit the dialog, click the "Cancel" button. Your entries are not applied.

Note on the "OK" button in the "Additional properties" tab in the "Save as new item dialog.

The following rules apply to the operation of the "OK" button in the "Additional properties" tab in the "Save as new item" dialog:

- If the required obligatory attributes are not filled, the "OK" button is greyed out and not available.
- The "OK" button is operable when no obligatory attributes are available or when all obligatory attributes are filled.

See also

Editing values in the tabular view of the "Additional properties" tab - (global libraries) (Page 122)

Structure of "General" tab in the "Save as new item" dialog - (global library) (Page 120)

5.13.1.5 Editing values in the tabular view of the "Additional properties" tab - (global libraries)

The attributes of a global library that are upload to the Teamcenter (server) are displayed in read/write mode in the tabular view of the "Additional Properties" tab.

Requirements for editing the values of uploaded attributes of a global library

The requirements for editing the values of the uploaded attributes of a global library are as follows:

- The attributes of the selected item type of a global library are uploaded.
- The attributes of the selected item revision of the item type of a global library are uploaded.
- the uploaded attributes of the item master form are connected with the TIA Portal project item type.
- The uploaded attributes of the item revision master form are connected with the item revision of the global library.

Note

Note that

only the above named scenarios are supported as of TIA Portal Teamcenter Gateway V14SP1. All other scenarios are ignored.

Note:

- If no attributes are uploaded to the Teamcenter (server), the following information is displayed in the "Additional properties" tab:
  No "Additional properties" were uploaded.
- If it is not possible to read uploaded attributes during connection establishment with TIA Portal Teamcenter Gateway, the following information is displayed in the "Additional properties" tab:
  No "Additional properties" were uploaded.

When using the same attribute in the "General" tab and in the "Additional properties" tab

Note the following:

When using the same attribute in the "General" tab and in the "Additional properties" tab, only the entry of the attribute from the "General" tab is always saved in Teamcenter.

Multiple use of attributes in the "Additional properties" tab in the "Save as a new item" dialog

Note the following:

When the same attribute is used multiple times in the "Additional properties" tab in the "Save as a new item" dialog, only the first entry of the attribute from the table view is always saved in Teamcenter.

Use of special characters for 'Char' data type

The following describes the naming conventions for the 'Char' data type, which must be observed for "Save as new item" .

Note the following:

- Use only ASCII values 32-126 in the input boxes of the 'Char' data type in the "Additional properties" tab in the "Save as new item" dialog.

See also

Structure of the "Additional properties" tab in the "Save as new item" dialog (global libraries) (Page 120)

Structure of the "Additional properties" tab in the "Save as new item" dialog (global library) (Page 121)

5.13.2 Check out global library

To exclusively reserve one or more global libraries for editing, you can check out the global libraries. The global libraries are locked for other users as a result.

Requirements

The following requires that

- The global library is connected to Teamcenter.
- The global library is open in "Read only" mode.
- There is no support for multiple selection of several global libraries.

Procedure

To edit a global library, you need to check it out as follows:

1. Select the corresponding global library from the "Global Library" pane.
2. Right-click on the global library and select the command "Teamcenter > Check out".

Note

Check out global library

Note: If the global library was changed locally or the dataset version is older than the currently available dataset version in Teamcenter, the "Open Teamcenter library" dialog opens.

1. Clicking the "Yes" button overwrites the currently available dataset version.
2. Clicking the "No" button opens the currently available dataset version as local copy.
3. Clicking the "Cancel" button cancels the checkout.

If you continue checking out, the dataset version of the global library is checked out in Teamcenter. The dataset version is locked for editing by another user.

Result

The global library is opened in "Write" mode again.

The dataset version of the global library is checked out and marked with the symbol in Teamcenter.

The dataset version is locked for editing by another user.

The following Teamcenter operations for working with global libraries are available to you in TIA Portal after a successful "Checkout":

- Save as new revision....
- Save as new item....
- Check in.
- Undo checkout.

See also

Saving a global library as a new item in Teamcenter (Page 117)

Check in global library (Page 124)

Creating new revision of a global library connected to Teamcenter (Page 127)

Undo checkout of a global library (Page 132)

5.13.3 Check in global library

Global libraries that you have saved in Teamcenter are in the "checked out" state. These global libraries are locked for other users as a result.

Requirement

The following requires that:

- the global library is connected to Teamcenter.
- the dataset of the associated revision of the global library is checked out by you.
- multiple selection of several global libraries is not supported.

Procedure

To revoke the edit lock of the global library, follow these steps:

1. Select the corresponding global library in the "Global Library" pane.
2. Right-click on the global library and select the command "Teamcenter > Check in".

Automatic check-in

Automatic check-in takes place when the global library is closed in the TIA Portal under the following conditions:

- If the "VersionInfo.xml file" is available for a global library.
- If the global library is opened in "Write" mode.

Automatic check-in does **not** take place when closing the global library under the following conditions:

- If **no** "VersionInfo.xml file" is available for a global library.
- If the global library is opened in "Read/Write" mode.
- If the global library is checked out in RAC.

Result

The global library is opened in "Read" mode again.

The global library has been checked into Teamcenter and the symbol removed from the folder view of Teamcenter. The edit lock of the global library is revoked.

The following Teamcenter operations for working with global libraries are available to you in TIA Portal:

- Save as new revision....
- Save as new item....
- Check out.

See also

Saving a global library as a new item in Teamcenter (Page 117)

Creating new revision of a global library connected to Teamcenter (Page 127)

Check out global library (Page 123)

5.13.4 Save changes to a global library connected to Teamcenter

Requirements

The following requires that

- The TIA Portal is connected to Teamcenter.
- The global library is opened in "Read/Write" mode.
- The selected global library is connected to Teamcenter and changes have been made.
- There is no support for multiple selection of several global libraries.

Note

Note the following:

- When there is no connection between the TIA Portal and Teamcenter, the "Teamcenter > Save" command is disabled.

Procedure

To save changes to a global library connected to Teamcenter and the corresponding dataset, proceed as follows:

1. Select the corresponding global library from the "Global Library" pane.
2. Right-click on the global library and select the command "Teamcenter > Save".
   The new dataset version of the global library is stored in Teamcenter as a zip file under the respective revision ID.

Restrictions

The following restrictions must be observed:

The number of dataset versions stored in Teamcenter can be configured by a Teamcenter administrator (default = 3). When the maximum is reached, the oldest dataset version is deleted and the new dataset version created.

Result

A new dataset version of the global library has been created in Teamcenter.

The following Teamcenter operations for working with global libraries are available to you in TIA Portal after "Save":

- Save as new revision....
- Save as new item....
- Check in.
- Undo checkout.
- Export...

See also

Creating new revision of a global library connected to Teamcenter (Page 127)
Saving a global library as a new item in Teamcenter (Page 117)
Undo checkout of a global library (Page 132)
Check in global library (Page 124)

5.13.5 Creating new revision of a global library connected to Teamcenter

Requirements

The following requires that

- The global library is connected to Teamcenter.
- There is no support for multiple selection of several global libraries.

Procedure

To create a new revision of a global library saved in Teamcenter, proceed as follows:

1. Select the corresponding global library from the "Global Library" pane.
2. Right-click on the global library and select the command "Teamcenter > Save as new revision...".
   With this function, the currently opened revision of the global library is used as the basis for creating the new revision.
   - The "Save as new revision" dialog opens.
   - The "General" tab is displayed by default.
   - The name of the selected global library is displayed in read-only mode in the "Name" field (e.g. t4tia_object_name).
   - The "item ID" of the selected global library is displayed in read-only mode in the "Item ID" field.
3. Optional: Assign a valid Teamcenter revision ID to the global library in the "Revision ID" field of the "General" tab.
   A maximum of 32 characters can be entered.
4. When you click the "Assign" button, Teamcenter automatically assigns the next possible revision ID to the global library.
   Result:
   - The assigned revision ID is displayed in the "Revision ID" field.
   - The "Revision ID" field cannot be edited.
   - The "Assign" button is disabled.

   If you leave the field 'empty', the next possible revision ID is assigned automatically by Teamcenter.

5. Optional: If necessary, add a comment in the "Comment" field in the "General" tab.
   A maximum of 240 characters can be entered.
   Comments from the previous version are taken over by default. You can edit the comments from the previous version or leave them unchanged.
6. Confirm your entry with "OK".
   The global library is saved in the specified folder structure under the specified name in Teamcenter.
   Once the global library has been saved in Teamcenter, the "The global library was successfully saved in Teamcenter" dialog opens.
   – Confirm this dialog by clicking "OK".
     An overlay symbol in the "Global Library" pane of TIA Portal indicates that the global library is connected to Teamcenter.
7. Clicking "Cancel" closes the "Save as new revision..." dialog.

Result

The new revision of the global library is created in Teamcenter and displayed as a new object below the last revision.

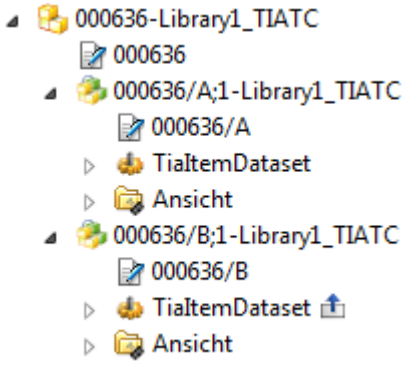

When revising the global library, the dataset of the currently open global library is adopted.

The corresponding dataset for the new revision of the global library is "checked out" and marked with the symbol in Teamcenter.

The dataset of the older revision is checked in, the symbol is removed in Teamcenter.

You can see the symbol in the folder view of Teamcenter.

The older revision of the global library is closed in TIA Portal. The new revision of the global library is opened in "write" mode in TIA Portal.

The following Teamcenter operations for working with global libraries are available after "Save as new revision".

- Save as new revision....
- Save as new item....
- Check in.
- Undo checkout.

See also

Saving a global library as a new item in Teamcenter (Page 117)

Check in global library (Page 124)

Undo checkout of a global library (Page 132)

5.13.5.1 "General" tab in the "Save as new revision" dialog - (global libraries)

In the "General" tab of the "Save as new revision" dialog, you define the properties of a global library that you want to save as new revision in Teamcenter.

5.13.5.2 "Additional properties" tab in the "Save as new revision" dialog (global libraries)

In the "Additional properties" tab, you edit the values of the uploaded additional properties of a global library that you want to save as new revision in Teamcenter.

In the "Additional properties" tab, you have the option of editing uploaded item types or derived types. The content of the "Additional properties" tab depends on the item types that are used, the derived types or the properties of the inherited types.

See also

Configuration of the "Additional properties" tab in the "Save as new revision" dialog (global library) (Page 130)

5.13.5.3 Structure of "General" tab in the "Save as new revision" dialog - (global library)

Structure of the "General" tab in the "Save as new revision" dialog

The view of the "General" tab consists of the following fields and buttons:

Field	Description
Name (read only):	The name of the selected TIA Portal project is displayed in read-only mode in the "Name" field (for example, t4tia_object_name)
Item ID (read only):	The "item ID" of the selected TIA Portal project is displayed in read-only mode in the "Item ID" field.
Revision ID:	Entry field for a valid Teamcenter revision ID which you want to assign to the new revision.
Comment:	Entry field for a comment.

Button	Description
Assign:	By clicking this button you automatically assign a revision ID to the global library.
OK	You confirm your entries by clicking the "OK" button. The dialog is closed.
Cancel	To exit the dialog, click the "Cancel" button. Your entries are not applied.

See also

"General" tab in the "Save as new revision" dialog - (global libraries) (Page 129)

5.13.5.4 Configuration of the "Additional properties" tab in the "Save as new revision" dialog (global library)

Structure of the "Additional properties" tab in the "Save as new revision" dialog

The information shown in tabular format in the "Additional properties" tab depends on:

- Which item type and which of its derived types you have selected.
- Which properties the selected item types and their derived types have inherited.

Properties of the "Additional properties" tab

The properties of item types and their derived types are displayed in read/write mode in the "Additional properties" tab.

The view of the "Additional properties" tab consists of the following columns and buttons:

Column	Meaning
Name	Name of the additional properties.
Type	Type of the additional properties.
Value	Value of additional properties.

Button	Meaning
Drop-down list	You can add obligatory attributes by clicking on the drop-down list.
OK	You confirm your entries by clicking the "OK" button. The dialog is closed.
Cancel	To exit the dialog, click the "Cancel" button. Your entries are not applied.

Note on the "OK" button in the "Additional properties" tab in the "Save as new revision" dialog.

The following rules apply to the operation of the "OK" button in the "Additional properties" tab in the "Save as new revision" dialog:

- If the required obligatory attributes are not filled, the "OK" button is greyed out and not available.
- The "OK" button is operable when no obligatory attributes are available or when all obligatory attributes are filled.

See also

"Additional properties" tab in the "Save as new revision" dialog (global libraries) (Page 129)

Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects) (Page 131)

5.13.5.5 Editing values in the tabular view of the "Additional properties" tab - (TIA Portal projects)

The attributes of a TIA Portal project that are upload to the Teamcenter server are displayed in read/write mode in the tabular view of the "Additional Properties" tab.

Requirements for editing the values of uploaded attributes of a TIA Portal project

The requirements for editing the values of the uploaded attributes of a TIA Portal project are as follows:

- The attributes of the selected TIA Portal project item type are uploaded.
- The attributes of the selected item revision of the TIA Portal project item type are uploaded.
- the uploaded attributes of the item master form are connected with the TIA Portal project item type.
- the uploaded attributes of the item revision master form are connected with the TIA Portal project item revision.

Note

Note that

only the above named scenarios are supported as of TIA Portal Teamcenter Gateway V14SP1. All other scenarios are ignored.

Note:

- If no attributes are uploaded to the Teamcenter (server), the following information is displayed in the "Additional properties" tab:
  No "Additional properties" were uploaded.
- If it is not possible to read uploaded attributes during connection establishment with TIA Portal Teamcenter Gateway, the following information is displayed in the "Additional properties" tab:
  No "Additional properties" were uploaded.

When using the same attribute in the "General" tab and in the "Additional properties" tab

Note the following:

When using the same attribute in the "General" tab and in the "Additional properties" tab, only the entry of the attribute from the "General" tab is always saved in Teamcenter.

Multiple use of attributes in the "Additional properties" tab in the "Save as a new revision" dialog

Note the following:

When the same attribute is used multiple times in the "Additional properties" tab in the "Save as a new revision" dialog, only the first entry of the attribute from the table view is always saved in Teamcenter.

Use of special characters for 'Char' data type

The following describes the naming conventions for the 'Char' data type, which must be observed for "Save as new revision".

Note the following:

- Only use ASCII values 32-126 in the input boxes of the 'char' data type in the "Additional properties" tab in the "Save as new revision" dialog.

See also

"Additional properties" tab in the "Save as new revision" dialog (global libraries) (Page 129)

Configuration of the "Additional properties" tab in the "Save as new revision" dialog (global library) (Page 130)

5.13.6 Undo checkout of a global library

To discard changes made to a checked-out global library, you have the option of undoing the checkout.

Requirement

The following requires that:

- the global library is connected to Teamcenter.
- the global library is open in "Write" mode.
- multiple selection of several global libraries is not supported.

Procedure

To undo the checkout of a global library, follow these steps:

1. Select the corresponding global library in the "Global Library" pane.
2. Right-click on the global library and select the command "Teamcenter > Undo checkout". The "Undo checkout" dialog box opens.
3. Click "OK" to undo the checkout.
4. Click "Cancel" if you want to cancel "Undo checkout".

Note

All changes made since the last checkout are discarded.

Result

The checkout of the global library has been reversed. The symbol is removed from the folder view of Teamcenter.

The edit lock of the global library is revoked.

The following Teamcenter operations for working with global libraries are available to you in the TIA Portal after "Undo checkout":

- Save as new revision....
- Save as new item....
- Check out.

See also

Check out global library (Page 123)

Saving a global library as a new item in Teamcenter (Page 117)

Creating new revision of a global library connected to Teamcenter (Page 127)

5.13.7 Exporting a global library

The Teamcenter shortcut menus "Save" and "Export..." are interdependent. The status of the "Save" and "Export..." shortcut menus depends on the following conditions.

Teamcenter shortcut menu "Export..."

The Teamcenter shortcut menu "Export..." is activated if:

- The selected global library is connected to Teamcenter and no changes have been made.

Procedure

To export a global library, follow these steps:

1. Select the corresponding global library in the "Global Library" pane.
2. Right-click the global library and select the "Teamcenter > Export..." command.

Note

Note the following:

If the "Teamcenter > Export..." command is not active for the global library, changes have been made in the global library. Save the global library with the "Teamcenter > Save" command. The "Teamcenter > Export..." command is then active.

Result

The "Teamcenter > Export..." command removes the Teamcenter meta information from the global library and saves the modified global library to the folder you selected.

Note

Note the following:

An existing global library cannot be exported again to the same location under the same name.

Note the following:

1. The Teamcenter shortcut menu "Save" is deactivated if no connection exists between the TIA Portal and Teamcenter.
2. The Teamcenter shortcut menu "Export..." is deactivated if changes have been made in the selected global library.

5.13.7.1 "Info" tab in the "Export" - (global library) dialog

The "Info" tab displays the information about the storage location of the global library after export.

5.13.8 Saving a global library from TIA Portal to an item that exists in Teamcenter

5.13.8.1 Checks when using the function "Save to item..." - (global libraries)

When the "Save to item..." function is used, Teamcenter performs the following checks:

1. With the item ID / revision ID, a check is made to determine whether the item ID / revision ID exists in Teamcenter.
   If the item ID / revision ID does not exist in Teamcenter, Teamcenter displays a dialog with a corresponding message.
   – Click "OK" or close the dialog.
2. With the item ID / revision ID, a check is carried out to determine whether the item type in Teamcenter matches the "T4TiaProject" item type and its dependent types.
   If the item type in Teamcenter does not match the "T4TiaProject" item type and its dependent types, Teamcenter displays a dialog with a corresponding message.
   – Click "OK" or close the dialog.
3. With the item ID / revision ID, a check is made to determine whether a valid TIA dataset exists in the item and the associated revision.
   If no valid TIA dataset exists in the item and the associated revision, Teamcenter displays a dialog with a corresponding message.
   – Click "OK" or close the dialog.

4. With the item ID / revision ID, a check is carried out to determine whether the name of the TIA Portal object (name of the global library) matches the name of the dataset.
   If the name of the global library does not match the name of the dataset, Teamcenter displays a dialog with a corresponding message.
   – Click "OK" or close the dialog.
5. With the item ID / revision ID, a check is made to determine whether the dataset of the item and the associated revision in Teamcenter is checked out by another user.
   If the dataset of the item and the associated revision in Teamcenter is checked out by another user, Teamcenter displays a dialog with a corresponding message.
   – Click "OK" or close the dialog.
6. With the item ID / revision ID, a check is carried out to determine whether the global library version differs from the Teamcenter version.
   If the global library version differs from the Teamcenter version, the "Critical message" dialog prompts you to decide whether or not you want to overwrite the version in Teamcenter with your changes.
   – Click "Yes" to overwrite the version of the global library in Teamcenter with the new or modified version.
   – Click "No" to cancel the "Save to item..." operation.
7. With the item ID / revision ID, a check is carried out to determine:
   – Whether the user has the required access rights.
   – Which release status the item ID / revision ID has.
     If the user does not have the required access rights and the item ID / revision ID has the "Released" release status, a dialog with a corresponding message is displayed by Teamcenter.
   – Click "OK" or close the dialog.

5.13.8.2 Saving a global library from TIA Portal to an item that exists in Teamcenter

With the "Save to item..." function, you connect a global library that was processed outside the cache server of Teamcenter to Teamcenter again.

When changes are made to a global library outside the cache server of Teamcenter, the "versionInfo.xml" file is updated.

Requirement

- An active connection exists between the TIA Portal and Teamcenter.

Procedure

To connect the global library to Teamcenter again, proceed as follows:

1. Open the global library in TIA Portal that you want to save as a new item in Teamcenter.
2. Select the corresponding global library in the "Global library" pane.
3. Right-click the global library and select the command "Teamcenter > Save to item...".
   The "Save to item..." dialog opens.

4. The name of the selected global library is displayed in the "Name" field by default.
5. The item ID associated with the global library from the "versionInfo.xml" file is displayed in the "Item ID" field by default.
   - Optional: If necessary, change the existing item ID in the "Item ID" field.
   - If the "versionInfo.xml" file does not exist, the "Item ID" field is empty.
     Assign an already valid or any Teamcenter ID to the "Item ID" field.
     A maximum of 128 characters can be entered.
6. The Revision ID associated with the global library from the "versionInfo.xml" file is displayed in the "Revision ID" field by default.
   - Optional: If necessary, change the existing revision ID in the "Revision ID" field.
   - If the "versionInfo.xml" file does not exist, the "Revision ID" field is empty.
     Assign an already valid or any Teamcenter revision ID to the "Revision ID" field.
     A maximum of 32 characters can be entered.
7. Confirm your entry with "OK".
   The global library is saved under the specified item ID / revision ID in Teamcenter.
8. If you click "Cancel", the "Save to item..." operation is canceled.

Note

Note the following:

The Item ID / Revision ID fields are obligatory fields. The "OK" button can only be edited if a value is displayed in the Item ID / Revision ID fields.

Preliminary check for the "Save to item..." operation

When the "Save to item..." function is used, Teamcenter performs the following checks:

- The check for the existence of the item.
- The check of the release state of the data set.
- The check of the old data.

For more details, see: Checks when using the function "Save to item..." (Page 83)

The following is checked during the save operation:

1. Whether the dataset version saved in Teamcenter and the associated revision are checked out by the currently logged-on user or another user.
   If the dataset version is not checked out, the dataset version is implicitly checked out when the "Save to item..." function is executed.
2. Whether the global library was opened outside the cache server of Teamcenter.
   If the global library was opened outside the Teamcenter cache path, you must save the global library to the Teamcenter cache path.
3. Whether the global library was opened outside the cache server of Teamcenter or whether a global library exists under the same item ID / revision ID in the cache server of Teamcenter.
   If the global library was opened outside the cache server of Teamcenter and a global library exists under the same item ID / revision ID in the cache server of Teamcenter, the following message is displayed in the "Critical message" dialog:
   – "A dataset version of the global library is located in the cache server of Teamcenter. Do you want to overwrite the dataset version?"
     Click "Yes" to overwrite the dataset version in the cache server of Teamcenter.
     Click "No" to cancel the "Save to item..." function or close the dialog.

   If errors occur while exporting and archiving, a dialog with a corresponding error message is displayed.

Note

Remedy following cancellation of export and archive operation:

1. Create a new dataset version in Teamcenter.
2. Upload the global library to the dataset.
   The dataset is checked out in the process.
   If the creation of the item revision is not possible, an error message is displayed indicating that the creation of the item revision failed and that the export and archive operation was canceled.
   To confirm the error message, follow these steps:
   – Click "OK" or close the message.
     A copy of the TIA Portal project is not saved in the TcCache.
     A TcCache path for the TIA Portal project is not created.

Actions on "Save to item..."

Once all checks are finished, the following actions are performed:

1. The global library is archived and saved under the specified item ID / revision ID in Teamcenter.
2. The associated dataset has the status "Checked out" and is marked with the following symbol.

   You can see the symbol in the folder view of Teamcenter.
3. After completion of the actions, the "The global library was successfully saved to Teamcenter" dialog opens.
   – Confirm this dialog by clicking "OK".

"Rollback" functions

1. If the export and archive operation failed, the older version of the "versionInfo.xml" file is used.
2. If the export and archive operation failed, remove the exported and archived files from the local hard drive.
3. If the export and archive operation failed, remove all the created objects (dataset version) in Teamcenter. The original version of the "versionInfo.xml" file is accessed.
   As soon as the global library has been saved in Teamcenter with the "Save to item" function, the "The global library was successfully saved to Teamcenter" dialog opens.
   – Confirm this dialog by clicking "OK".

Result

The following Teamcenter operations for working with global libraries are available to you in TIA Portal:

- Save as new item...
- Save as new revision...
- Save to item...
- Check in.
- Undo checkout.
- Export...

5.13.8.3 Structure of dialog "Save to item..."

In the "Save to item" dialog, you define the properties of a global library that you want to save in Teamcenter.

Structure of dialog "Save to item..."

The view of the "Save to item..." dialog consists of the following fields and buttons.

Field	Description
Name:	Input box for the name under which you want to save the global library in Teamcenter. The name of the selected global library is displayed by default.
Item ID / Revision ID:	Input box for the item ID / revision ID that you want to assign to the global library.

Button	Note
OK	You confirm your entries by clicking the "OK" button. The dialog closes.
Cancel	You exit the dialog by clicking the "Cancel" button. Your entries are not applied.

5.13.9 Save the global library which is saved in the Teamcenter cache folder to the associated item again.

Using the "Save to item..." function, save a global library from the Teamcenter cache folder to the associated item again.

Requirement

- An active connection exists between the TIA Portal and Teamcenter.

Procedure

To connect the global library to Teamcenter again, proceed as follows:

1. Select the corresponding global library in the "Global library" pane.
2. Right-click the global library and select the command "Teamcenter > Save to item...". The "Save to item..." dialog opens.
3. Assign a name in the "Name" field.
4. Assign an item ID to the "Item ID" field.
   – A maximum of 128 characters can be entered.
5. Enter a revision ID in the "Revision ID" field.
   – A maximum of 32 characters can be entered.

6. Confirm your entry with "OK".
   The global library is saved under the specified item ID / revision ID in Teamcenter.
7. If you click "Cancel", the "Save to item..." operation is canceled.

Note

Note the following:

The Item ID / Revision ID fields are obligatory fields. The "OK" button can only be edited if a value is displayed in the Item ID / Revision ID fields.

Actions on "Save to item..."

When the "Save to item..." function is executed, the following actions are performed:

1. The global library is archived and saved under the specified item ID / revision ID in Teamcenter.
2. The associated dataset has the status "Checked out" and is marked with the following symbol.

   You can see the symbol in the folder view of Teamcenter.
3. After completion of the actions, the "The global library was successfully saved to Teamcenter" dialog opens.
   – Confirm this dialog by clicking "OK".

Result

The following Teamcenter operations for working with global libraries are available to you in TIA Portal:

- Save as new item...
- Save as new revision...
- Save to item...
- Check in.
- Undo checkout.
- Export...

5.14 Finding and opening TIA Portal projects or global libraries with TIA Portal Teamcenter Gateway

Requirements

The following requires that

- there is an active connection between TIA Portal and Teamcenter.

Procedure

To search for a TIA Portal project connected to Teamcenter with TIA Portal Teamcenter Gateway and open it for further editing, proceed as follows:

1. Select the "Open project from Teamcenter..." command from the "Project" menu in the project view.
   The "Open project from Teamcenter" dialog opens.
2. Enter the project name of the TIA Portal project you want to find in the "Project name" field.
   A maximum of 128 characters can be entered.

   Note

   Searching for TIA Portal project items and their derived types

   The search compares the offered search criteria. When both the item type of a TIA Portal project "T4TiaProject" and its derived type "T4TiaSubProject" are defined, the item type "T4TiaProject" and the derived type "T4TiaSubProject" are listed in the search result list.

3. Optional: Enter the item ID of the TIA Portal project you want to find in the "Item ID" field.
   A maximum of 94 characters can be entered.
4. Optional: Enter the item name of the TIA Portal project you want to find in the "Item name" field.
   A maximum of 94 characters can be entered.

5. Optional: Enter the revision ID of the TIA Portal project you want to find in the "Revision" field.
   A maximum of 32 characters can be entered.

   Note

   All previously mentioned fields are optional. To start searching, at least one of the following search criteria must be defined however:
   - Project name:
   - Item ID:
   - Item name:

   You can also use wildcards in the search.

Wildcard characters	Meaning
*	Enter an asterisk as the wildcard for any number of characters. Example: You are searching for all words that begin with "Pro". Therefore, enter "Pro*" in the field.
?	Enter a question mark as a wildcard for any character. Example: You are searching for all words that begin with "Proj". Therefore, enter "Pro?" in the field.
;	Enter a semicolon as a wildcard within a search criterion. Example: You are searching for words that begin with "Pro*; auto". Therefore, enter "Pro*" ; "auto*" in the field.
Case is ignored during the search, which means you can the keywords in upper or lower case. Example: The search for "PROject1" and "Project1" is to return the same search results.	

6. Clicking the "Start search" button starts the search.
   The "Searching..." dialog opens. The search is performed in the background and cannot be canceled.
7. Clicking the "Reset" button deletes all entered search criteria and resets the search results.

Result

The TIA Portal projects found in Teamcenter are displayed in the "Open project from Teamcenter" dialog.

See also: Opening TIA Portal projects from the search result (Page 144).

Procedure

To search for a global library connected to Teamcenter with TIA Portal Teamcenter Gateway and open it for further editing, proceed as follows:

1. Click on a free area in the "Global libraries" pane, and right-click the "Open global library from Teamcenter..." command.
   Or select the "Global libraries > Open global library from Teamcenter..." command in the "Options" menu.
   The "Open global library from Teamcenter" dialog opens.
2. Enter the name of the global library you want to find in the "Library name" field.
   A maximum of 128 characters can be entered.

   Note

   Searching for global library and its derived types

   The search compares the offered search criteria. When both the item type of the global library "T4TiaLibrary" and its derived type "T4TiaSubLibrary" are defined, the item type "T4TiaLibrary" and the derived type "T4TiaSubLibrary" are listed in the search result list.

3. Optional: Enter the item ID of the global library you want to find in the "Item ID" field.
   A maximum of 94 characters can be entered.
4. Optional: Enter the item name of the global library you want to find in the "Item name" field.
   A maximum of 94 characters can be entered.
5. Optional: Enter the revision ID of the global library you want to find in the "Revision ID" field.
   A maximum of 32 characters can be entered.

   Note

   All previously mentioned fields are optional. To start searching, at least one of the following search criteria must be defined however:
   - Name library
   - Item ID
   - Item name

   You can also use wildcards in the search.

Wildcard characters	Meaning
*	Enter an asterisk as the wildcard for any number of characters. Example: You are searching for all words that begin with "Lib". Therefore, enter "Lib*" in the field.
?	Enter a question mark as a wildcard for any character. Example: You are searching for all words that begin with "Lib". Therefore, enter "Lib?" in the field.
;	Enter a semicolon as a wildcard within a search criterion. Example: You are searching for words that begin with "Lib; auto". Therefore, enter "Lib*" ; "auto*" in the field.
Case is ignored during the search, which means you can the keywords in upper or lower case. Example: The search for "LIBrary1" and "Library1" should return the same search results.	

6. Clicking the "Start search" button starts the search.
   The "Searching..." dialog opens. The search is performed in the background and cannot be canceled.
7. Clicking the "Reset" button deletes all entered search criteria and resets the search results.

Result

The global libraries found in Teamcenter are displayed in the "Open global library from Teamcenter" dialog.

See also: Opening global libraries from the search result (Page 148).

5.14.1 Opening TIA Portal projects from the search result

5.14.1.1 Opening TIA Portal projects from the search result

Procedure

To open the TIA Portal project from the search results, follow these steps:

1. To display the available revisions, click on the arrow symbol to the left of the TIA Portal project.

To open the required revision of the TIA Portal project, follow these steps:

1. Select the required revision of the TIA Portal project.
   Also note the "Check out" check box when "Opening TIA Portal project from the search result list" (the "Check out" check box is enabled by default).
2. Click the "Open" button.

Result

- The dataset of the TIA Portal project is checked out and marked with the symbol in Teamcenter.
- The selected revision of the TIA Portal project is opened in "write" mode in the TIA Portal.
- The "Open TIA Portal project from Teamcenter" dialog closes.

Notes on opening TIA Portal projects from the search result list.

1. Opening a TIA Portal project from the search result list **with** the "Check out" option.
   If the corresponding revision of the TIA Portal project has already been checked out by another user, the following message appears:
   "The project is already checked out by another user: User name. The project will be opened as read-only."
2. Opening a TIA Portal project from the search result list **without** the "Check out" option.
   The revision of the TIA Portal project is opened in "Read-only" mode.
3. Opening a TIA Portal project from the search result list (**with** or **without** the "Check out" option).
   If you have already opened the corresponding revision of the TIA Portal project, the following message appears: "The TIA Portal project you are attempting to open is already open."

Opening TIA Portal projects from the search result list

Note

Note the following:

A TIA Portal project, which is open in the TIA Portal at the same time as a reference project, should not be opened in the "Open TIA Portal projects from the search results" dialog with the "Check out" option.

Opening older TIA Portal projects from the search result list

Note

Note the following:

When opening older TIA Portal projects from the search result list, a dialog box appears stating that you can update the TIA Portal project.

1. Click the "OK" button to rename the updated TIA Portal project to the original TIA Portal project by overwriting the previous version in the Teamcenter cache.
2. Clicking "Cancel" causes the operation to be aborted.

Behavior when opening TIA Portal projects from the search result list or from Teamcenter cache

Note

TIA Portal projects that you open from the search result list or from Teamcenter cache are opened in "Write" mode or "Read only" mode depending on whether the "Check out" check box is enabled or disabled.

For TIA Portal projects that you open from the search result list in Teamcenter, the check-in/check-out status is synchronized with Teamcenter.

In this case, the system assumes that the TIA Portal project is checked out by the current user. This assumption is independent of the check-in/check-out status of the TIA Portal project on the Teamcenter server.

See also

Finding and opening TIA Portal projects or global libraries with TIA Portal Teamcenter Gateway (Page 141)

Structure of the "Properties" tab in the "Open TIA Portal project from Teamcenter" dialog (Page 146)

5.14.1.2 Structure of the "Properties" tab in the "Open TIA Portal project from Teamcenter" dialog

Structure

The "Properties" tab is located in the lower part of the "Open project from Teamcenter" dialog. To display the "Properties" tab, follow these steps:

1. Select the respective TIA Portal project in the search result list.
2. Click on the up arrow or down arrow in the lower section of the search result within the window divider to hide or show the adjacent sections.
3. Click the "Properties" tab to display the properties of the selected TIA Portal project. The properties are displayed in tabular form.

Structure of the "Properties" tab in the "Open project from Teamcenter" dialog

Properties of the Teamcenter item

The following table shows the properties of the selected item:

Property	Description
	Value
Type	Different information is displayed depending on the selected item revision.
Owner	"Owner" of the item (for example, user name and active identifier, for example, name, first name, z002vdyx).
Access	"Read" or "Write", depending on the access rights of the current user.
Last change	"Last change" shows the time at which the item was last changed.
Status	"Checked out by (<user ID>)" - shows the user and the user ID by which the item was checked out. "Released" - the item has the "Released" status. "None" - the item has neither the "Checked out" status nor the "Released" status.
Description	Displays the description text (defined by the T4tia_Description_Attribute) of the item, which was created by saving the project or the global library to Teamcenter as a new item.

Teamcenter dataset properties

The properties of the selected dataset are displayed in the "Properties" tab.

The following table shows the properties of the selected dataset:

Property	Description	
	Value (Revision)	Value (Dataset)
Type	Different information is displayed depending on the selected item revision.	Different information is displayed depending on the selected dataset type.
Owner	"Owner" of the item (for example, user name and active identifier, for example, name, first name, z002vdyx).	"Owner" of the dataset (for example, user name and active identifier, for example, name, first name, z002vdyx). Note: The owner of the dataset and the item revision must be the same for the TIA projects and global libraries to be saved to Teamcenter.
Access	"Read" or "Write", depending on the access rights of the current user.	"Read" or "Write", depending on the access rights of the current user.
Last modified (date of change)	"Date of change" shows the time at which the item was last changed.	"Date of change" shows the time at which the dataset was last changed.
Status	"Checked out by (<user ID>)" - shows the user and the user ID by which the item was checked out. "Released" - the item has the "Released" status. "None" - the item has neither the "Checked out" status nor the "Released" status.	"Checked out by (<userid>)" - shows the user and the user ID by which the dataset was checked out. "Released" - the item has the "Released" status. "None" - the item has neither the "Checked out" status nor the "Released" status.

Note

Note the following:

- The "Properties" are displayed as read-only and you cannot change them.

Note the following:

- If you have not selected anything in the search results view,
- If the search result view is empty,
- If you have clicked the "Reset" button,
  the following information is displayed on the "Properties" tab:
  No 'properties' are available.
  No 'properties' can be currently displayed. No object is selected.

See also

Finding and opening TIA Portal projects or global libraries with TIA Portal Teamcenter Gateway (Page 141)

Displaying the dataset status in the "Open project from Teamcenter" or "Open global library from Teamcenter" dialog (Page 152)

5.14.2 Opening TIA Portal projects from the TIA Portal project list.

Opening TIA projects from the list of recently opened projects.

If you have saved a TIA Portal project in Teamcenter and then opened it from the list of recently used projects in the TIA Portal, the Teamcenter operations "Save" and "Check out" are disabled.

Result

The TIA Portal project is opened in "read-only" mode

The following Teamcenter operations for working with the TIA Portal project are available to you in the TIA Portal after the opening of a TIA Portal project from the list of recently used projects:

- Save as new item....

Remedy

1. Close the TIA Portal project opened from the list of the recently used projects.
2. Open the TIA Portal project again from Teamcenter.
3. To do this, select the "Open Teamcenter project..." command in the "Project" menu.
   The "Open project from Teamcenter" dialog opens.
4. Enter the project name of the TIA Portal project in the "Project name" field.
5. Click the "Start search" button.
   The "Searching..." dialog opens. The search is performed in the background and cannot be canceled.
6. Then open the TIA Portal project from the search result list.
   See also: Opening TIA Portal projects from the search result (Page 144).

5.14.3 Opening global libraries from the search result

5.14.3.1 Opening global libraries from the search result

Procedure

To open global libraries from the search results, proceed as follows:

1. To display the available revisions, click on the arrow symbol to the left of the corresponding global library.

To open the required revision of the global library, follow these steps:

1. Select the required revision of the global library.
   Also note the "Check out" check box when "Opening global libraries from the search result list" (the "Check out" check box is enabled by default).
2. Click the "Open" button.

Result

- The dataset of the global library is checked out and marked with the symbol in Teamcenter.
- The selected revision of the global library is opened in "write" mode in TIA Portal.
- The "Open global library from Teamcenter" dialog closes.

Notes on opening global libraries from the search result list.

1. Opening a global library from the search result list **with** the "Check out" option.
   If the corresponding revision of the global library is already checked out by another user, the following message appears:
   "The library is already checked out by another user: User name. The library will be opened as read-only."
2. Opening a global library from the search result list **without** the "Check out" option.
   The revision of the global library opens in "Read only" mode.
3. Opening a global library from the search result list (**with** or **without** the "Check out" option).
   If you have already opened the corresponding revision of the global library, the following message appears: "The global library you are attempting to open is already open."

Opening older global libraries from the search result list

Note

Note the following:

When opening older global libraries from the search result list, a dialog box appears stating that you can update the global library.

1. Click the "OK" button to rename the updated global library to the original global library by overwriting the previous version in the Teamcenter cache.
2. Clicking "Cancel" causes the operation to be aborted.

Behavior when opening TIA Portal projects from the search result list or from Teamcenter cache

Note

Global libraries that you open from the search result list or from Teamcenter cache are opened in "Write" mode or "Read only" mode depending on whether the "Check out" check box is enabled or disabled.

For global libraries that you open from the search result list in Teamcenter, the check-in/check-out status is synchronized with Teamcenter.

In this case, the system assumes that the global library is checked out by the current user. This assumption is independent of the check-in/check-out status of the global library on the Teamcenter server.

See also

Finding and opening TIA Portal projects or global libraries with TIA Portal Teamcenter Gateway (Page 141)

Structure of the "Properties" tab in the "Open global library from Teamcenter" dialog (Page 150)

5.14.3.2 Structure of the "Properties" tab in the "Open global Library from Teamcenter dialog

Structure of the "Properties" tab in the "Open global library from Teamcenter" dialog

Structure

The "Properties" tab is located in the lower part of the "Open global library from Teamcenter" dialog. To display the "Properties" tab, follow these steps:

1. Select the corresponding global library in the search result list.
2. Click on the up arrow or down arrow in the lower section of the search result within the window divider to hide or show the adjacent sections.
3. Click the "Properties" tab to display the properties of the selected global library. The properties are displayed in tabular form.

Structure of the "Properties" tab in the "Open global library from Teamcenter" dialog

Properties of the Teamcenter item

The following table shows the properties of the selected item:

Property	Description
	Value
Type	Different information is displayed depending on the selected item revision.

Property	Description
Owner	"Owner" of the item (for example, user name and active identifier, for example, name, first name, z002vdyx).
Access	"Read" or "Write", depending on the access rights of the current user.
Last change	"Last change" shows the time at which the item was last changed.
Status	"Checked out by (<user ID>)" - shows the user and the user ID by which the item was checked out. "Released" - the item has the "Released" status. "None" - the item has neither the "Checked out" status nor the "Released" status.
Description	Displays the description text (defined by the T4tia_Description_Attribute) of the item, which was created by saving the project or the global library to Teamcenter as a new item.

Teamcenter dataset properties

The properties of the selected dataset are displayed in the "Properties" tab.

The following table shows the properties of the selected dataset:

Property	Description	
	Value (Revision)	Value (Dataset)
Type	Different information is displayed depending on the selected item revision.	Different information is displayed depending on the selected dataset type.
Owner	"Owner" of the item (for example, user name and active identifier, for example, name, first name, z002vdyx).	"Owner" of the dataset (for example, user name and active identifier, for example, name, first name, z002vdyx). Note: The owner of the dataset and the item revision must be the same for the TIA projects and global libraries to be saved to Teamcenter.
Access	"Read" or "Write", depending on the access rights of the current user.	"Read" or "Write", depending on the access rights of the current user.
Last modified (date of change)	"Date of change" shows the time at which the item was last changed.	"Date of change" shows the time at which the dataset was last changed.
Status	"Checked out by (<user ID>)" - shows the user and the user ID by which the item was checked out. "Released" - the item has the "Released" status. "None" - the item has neither the "Checked out" status nor the "Released" status.	"Checked out by (<userid>)" - shows the user and the user ID by which the dataset was checked out. "Released" - the item has the "Released" status. "None" - the item has neither the "Checked out" status nor the "Released" status.

Note

Note the following:

- The "Properties" are displayed as read-only and you cannot change them.

Note the following:

- If you have not selected anything in the search results view,
- If the search result view is empty,
- If you have clicked the "Reset" button,
  the following information is displayed on the "Properties" tab:
  No 'properties' are available.
  No 'properties' can be currently displayed. No object is selected.

5.14.4 Displaying the dataset status in the "Open project from Teamcenter" or "Open global library from Teamcenter" dialog

5.14.4.1 Displaying the dataset status

Showing the dataset status in search results

The search results are shown in tabular form in the "Open project from Teamcenter" or "Open global library from Teamcenter" dialog. The following table shows which 'Status of dataset' the selected dataset has.

Symbol	Meaning
	Checked out by current user.
	Checked out by another user.
	The dataset is in "Released" status.

Note

Note that the displayed status is not updated automatically. To update the status, proceed as follows:

1. To close and reopen the object, click the arrow symbol next to the corresponding object in the search result.
   Closing and opening of the object causes the current status information to be displayed.

5.14.4.2 Displaying the item revision status

Displaying the status of the item revision in the search result

The search results are shown in tabular form in the "Open project from Teamcenter" or "Open global library from Teamcenter" dialog. The following table shows the 'Status of item revision' of the selected dataset.

Symbol	Meaning
	Checked out by current user.
	Checked out by another user.
	The dataset is in "Released" status.

Note

Note that the displayed status is not updated automatically. To update the status, proceed as follows:

1. To close and reopen the object, click the arrow symbol next to the corresponding object in the search result.
   Closing and opening of the object causes the current status information to be displayed.

5.14.4.3 Displaying the cache status

Displaying the TC cache status in the search result

The search results are shown in tabular form in the "Open project from Teamcenter" or "Open global library from Teamcenter" dialog. The following table shows the 'TC Cache Status' of the selected dataset.

Symbol	Meaning
	The dataset is synchronized with Teamcenter.
	The dataset has been changed in the cache.
	Older dataset version in the cache.

Note

Note that the displayed status is not updated automatically. To update the status, proceed as follows:

1. To close and reopen the object, click the arrow symbol next to the corresponding object in the search result.
   Closing and opening of the object causes the current status information to be displayed.

5.15 Manual comparison of two TIA Portal projects (dataset versions) with the same revision

How to compare two TIA Portal projects (dataset versions) of the same revision with each other

To compare two TIA Portal projects (dataset versions) of the same revision with each other, follow these steps:

1. Use "RAC" to export the "Version 1" dataset of the corresponding TIA Portal project to a local directory.
2. Unzip the ZIP file.

   Note

   Make sure that you do not save the file to the TIA Portal Teamcenter Gateway cache path.

3. Open "Version 1" dataset as reference project in the TIA Portal.
4. Open "Version 2" dataset of the same revision in the TIA Portal. Select the "Open project from Teamcenter..." command from the "Project" menu in the project view ("Open and search").

5.16 Symbols

5.16.1 Open object status in Project dialog from Teamcenter

Symbols in the "Open project from Teamcenter" dialog

"Status of dataset" column

Depending on the status of the dataset, the following symbols are displayed in the "Status of dataset" column:

Symbol	Designation	Description
	"Checked out by current user" status	The symbol behind an object indicates that an object is checked out by the current user.
	"Checked out by another user" status	The symbol behind an object indicates that an object is checked out by another user.
	"TCM Released" status	The symbol behind an object indicates that an object has the "RELEASED" status.

"Status of item revision" column

Depending on the status of the item, the following symbols are displayed in the "Status of item revision" column:

Symbol	Designation	Description
	"Checked out by current user" status	The symbol behind an object indicates that an object is checked out by the current user.
	"Checked out by another user" status	The symbol behind an object indicates that an object is checked out by another user.
	"TCM Released" status	The symbol behind an object indicates that an object has the "RELEASED" status.

"Cache status" column

Depending on the status of the TC cache, the following symbols are displayed in the "Cache status" column:

Symbol	Designation	Description
	"In Sync" status	The symbol behind an object indicates that the object is synchronized with Teamcenter.
	"Locally modified" status	The symbol behind an object indicates that an object in the local cache of Teamcenter has changed.
	"Stale Dataset" status	The symbol behind an object indicates that an outdated dataset version of the object is in the cache.

5.16.2 Object status in the Open global Library from Teamcenter dialog

Symbols in the "Open global library from Teamcenter" dialog

"Status of dataset" column

Depending on the status of the dataset, the following symbols are displayed in the "Status of dataset" column:

Symbol	Designation	Description
	"Checked out by current user" status	The symbol behind an object indicates that an object is checked out.
	"Checked out by another user" status	The symbol behind an object indicates that an object has been checked out by another user.
	"TCM Released" status	The symbol behind an object indicates that an object has the "RELEASED" status.

"Status of item revision" column

Depending on the status of the item, the following symbols are displayed in the "Status of item revision" column:

Symbol	Designation	Description
	"Checked out by current user" status	The symbol behind an object indicates that an object is checked out.
	"Checked out by another user" status	The symbol behind an object indicates that an object has been checked out by another user.
	"TCM Released" status	The symbol behind an object indicates that an object has the "RELEASED" status.

"Cache status" column

Depending on the status of the item, the following symbols are displayed in the "Cache status" column:

Symbol	Designation	Description
	"In Sync" status	The symbol behind an object indicates that an object is synchronized with Teamcenter.
	"Locally modified" status	The symbol behind an object indicates that an object in the local cache of Teamcenter has changed.
	"Stale Dataset" status	The symbol behind an object indicates that an outdated dataset version of the object is in the cache.

5.16.3 Symbols that display the connection status between the TIA Portal and Teamcenter

Show connection status

Below you can find a list of symbols that display the connection status between the TIA Portal and Teamcenter.

Symbol	Description
	This symbol indicates that there is **a** connection between the TIA Portal and Teamcenter.
	This symbol indicates that there is **no** connection between the TIA Portal and Teamcenter.
	This symbol indicates that the Teamcenter server is not activated for UTF-8.

Index

Retrieving projects, 35

A

Archiving projects, 33, 34

C

Compatibility
    Backward compatibility of projects, 23
    Projects from older program versions, 23
    Projects with role permissions, 23

L

Logs
    Deleting logs, 17
    Displaying logs, 17

M

Minimizing projects, 33
Multiuser Engineering
    Introduction, 45

O

Opening a project, 10

P

Project
    Closing, 13
    Creating, 9
    Deleting, 15
    Removing, 14
    Saving, 12
    Showing properties, 16
Project archives, 33

R

Reference project
    Basics, 37
    Closing, 38
    Comparing, 39
    Opening, 38

SIEMENS

Editing project data

Programming and Operating Manual

Compiling and loading project data 1

Comparing project data 2

Protecting project data 3

Printing project contents 4

Working with multi-language projects 5

Working with text lists 6

Using memory cards 7

Using cross-references 8

Providing user-defined documentation 9

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Compiling and loading project data....7

1.1 Compiling project data....7
1.1.1 General information on compiling project data....7
1.1.2 Compiling project data....8

1.2 Loading project data....9
1.2.1 General information on loading....9
1.2.2 Downloading project data to a device....11
1.2.3 Downloading project data to a memory card....13
1.2.4 Uploading project data from a device....15
1.2.5 Loading project data from a memory card....17

2 Comparing project data....19

2.1 Basics of project data comparison....19

2.2 Comparing offline/online....23

2.3 Carrying out offline/offline comparisons....24

2.4 Using the compare editor....26
2.4.1 Overview of the compare editor....26
2.4.2 Show and hide table columns....32
2.4.3 Filtering the compare editor view....33
2.4.4 Running a detailed comparison....34
2.4.5 Updating the comparison results....36
2.4.6 Set comparison criteria....37
2.4.7 Synchronizing non-identical objects....38
2.4.7.1 Specifying actions....38
2.4.7.2 Synchronizing objects....39
2.4.8 Changing the view....40

3 Protecting project data....41

3.1 Protection concept for project data....41

3.2 Revoking access rights for devices....43

4 Printing project contents....45

4.1 Printing project documentation....45
4.1.1 Documentation settings....45
4.1.2 Printout of project contents....46
4.1.3 Changing the print settings....47
4.1.4 Specifying the print layout....48
4.1.5 Entering document information....49
4.1.6 Managing cover pages and frames....50
4.1.6.1 Using cover pages and frames....50
4.1.6.2 Creating frames....51
4.1.6.3 Creating a cover page....51
4.1.6.4 Using ready-made frames and cover pages....52

4.1.7 Designing cover pages and frames....53
4.1.7.1 Editing cover pages and frames....53
4.1.7.2 General operation of the documentation editor....54
4.1.7.3 Specifying the print area....57
4.1.7.4 Inserting placeholders for metadata....57
4.1.8 Displaying print preview....59
4.1.8.1 Creating a print preview....59
4.1.8.2 Operation within the print preview....61
4.1.9 Printing project data....62

4.2 Printing module labels....64
4.2.1 Print function for module labels....64
4.2.2 Printing labels....65
4.2.3 Exporting labeling data as XML....67
4.2.4 XML schema of the export file....68
4.2.5 Determining the print area offset....71

5 Working with multi-language projects....73

5.1 Project text basics....73

5.2 Categories of user texts....76

5.3 Select project languages....77

5.4 Setting the editing language....78

5.5 Translating all project texts in tabular form....79

5.6 Translating project texts....80

5.7 Translating project texts using reference texts....81

5.8 Exporting project texts....82

5.9 Importing project texts....85

5.10 Application examples for multilanguage projects....86

6 Working with text lists....89

6.1 Text lists....89

6.2 Creating user-defined text lists....91

6.3 Editing user-defined text lists....92

6.4 Editing system-defined text lists....93

6.5 Exporting text lists....94

6.6 Importing text lists....95

7 Using memory cards....97

7.1 Basics about memory cards....97

7.2 Adding a user-defined card reader....98

7.3 Accessing memory cards....99

7.4 Displaying properties of memory cards....100

7.5 Show data on the memory card of a controller....101

8 Using cross-references....103

8.1 Using cross-references....103

9 Providing user-defined documentation....105

9.1 Using user-defined documentation....105

9.2 Specifying settings in the TIA Portal....108

9.3 Specifying settings with an XML file....109

9.4 Creating a homepage....112

9.5 Conventions for the creation....113

9.6 Calling user-defined documentation....116

9.7 Displaying the call log....117

9.8 Creating user-defined documentation....118

Index....121

Compiling and loading project data 1

1.1 Compiling project data

1.1.1 General information on compiling project data

Compiling project data

During compilation, project data is converted so that it can be read by the device. Hardware configuration data and program data can be compiled separately or together. You can compile the project data for one or more target systems at the same time.

The following project data must be compiled prior to loading:

- Hardware project data, for example, configuration data of the devices or networks and connections
- Software project data, for example, program blocks or process screens

---

Note

While a device is being compiled, no additional compiling process can be started. Note in this regard that you can not only perform a compiling process manually, but you can also trigger it automatically for operator control and monitoring devices.

Scope of the compilation

When you compile project data, you have the following options depending on the device involved:

- Hardware and software (only changes)
- Hardware (only changes)
- Hardware (rebuild completely)
- Software (only changes)
- Software (rebuild all)
- Software (reset memory reserve)

See also

Compiling project data (Page 8)

1.1.2 Compiling project data

The following section describes the general procedure for compiling project data in the project tree. You will find details of how certain objects are compiled and any special points to note in the online help of the product.

Procedure

To compile project data, follow these steps:

1. In the project tree, select the devices whose project data you want to compile.
2. Select the option you require in "Compile" submenu of the shortcut menu.

   Note

   Note that the options available to you depend on the selected device.

   The project data is compiled. You can check whether or not the compilation was successful in the Inspector window with "Info > Compile".

See also

General information on compiling project data (Page 7)

1.2 Loading project data

1.2.1 General information on loading

Introduction

In order to set up your automation system, you need to load the project data you generated offline on the connected devices. This project data is generated, for example when configuring hardware, networks, and connections or when programming the user program or when creating recipes.

The first time you download, the entire project data is downloaded. During later loading operations, only changes are downloaded.

You can download the project data to devices and memory cards.

Note

As of TIA Portal V15 and an S7-1500 CPU as of firmware V2.5, tag tables are included when downloading to a device or to a memory card and when downloading from a device or from a memory card. This makes the tag tables available online on your CPU as well. The PLC tags are stored according to the tag tables configured offline. An online/offline comparison can be performed and tag tables can be downloaded by the device.

Note

While a device is being compiled, no additional download process can be started. Note in this regard that you can not only perform a compiling process manually, but you can also trigger it automatically for operator control and monitoring devices.

Possible options for downloading

Depending on the object you want to download, you have the following options:

- Hardware and software (only changes)
  Both the hardware configuration and software are downloaded to the destination if differences exist between the online and offline versions.
- Hardware configuration
  Only the hardware configuration is downloaded to the destination.
- Software (only changes)
  Only the objects that differ online and offline are downloaded to the destination.
- Software (all)
  The PLC program including all blocks, PLC data types and PLC tags is downloaded to the target and all values are reset to their initial values. Be aware that this also applies to retentive values.

You can also upload project data already contained in a device back to your project. You have the following options:

- Upload entire device as new station
  The hardware configuration of the device and the software on the device are loaded in the project.
  All relevant data of the device is uploaded to the project.
- Upload software of a device
  Only the blocks and parameters from the device are uploaded to an existing CPU in the project.

In both cases, during loading all instances of library types are connected again with the corresponding version of the type in the project library. If no suitable type is yet available for a loaded instance or the correct version of the type does not exist in the project library, the type or the version is added to the project library.

Downloading with synchronization

In team engineering, it is possible for several users to work on one project with several engineering systems at the same time and to access one S7-1500 CPU. To ensure consistency within the shared project, it is necessary to synchronize the changed data prior to loading so that nothing gets overwritten unintentionally.

If differences are detected between the online and offline data management within the shared project that were caused by a different engineering system, automatic synchronization of the data to be loaded is offered when loading.

In this case, the "Synchronization" dialog displays the data to be synchronized with the current status (online-offline comparison) and the possible actions.

Use case	Recommendation	Synchronization
One or more blocks on the CPU (online) are more recent than in the engineering system (offline).	These blocks should be uploaded from the CPU to the engineering system before downloading.	Automatic synchronization is possible: The blocks in the engineering system are updated prior to loading.
One or more new blocks have been created and exist only on the CPU (online).	These blocks should be uploaded from the CPU to the engineering system before downloading.	Automatic synchronization is possible: The new blocks are added to the engineering system prior to the download.
One or more blocks on the CPU have been deleted.	The blocks should also be deleted prior to the download in the engineering system.	Automatic synchronization is not possible. The blocks deleted on the CPU should be manually deleted in the offline project in the engineering system.

Use case	Recommendation	Synchronization
One or more blocks on the CPU and in the engineering system are different. This is the case when a different user has changed blocks to which you have also made corrections and has already downloaded them to the CPU.	These blocks with competing changes must be adapted manually. You decide in this case which changes you are going to accept. If the blocks on the CPU are to be retained, you should adopt these blocks from the CPU in your engineering system before downloading to the CPU. If the blocks that you have changed are to be applied, you can continue with the download without synchronization.	Automatic synchronization is not possible: The affected blocks on the CPU or in the engineering system must be adapted manually. One of the existing block versions (online or offline) will be overwritten in the process.
There are differences in the hardware configuration on the CPU (online) and in the engineering system (offline).	Differences in the hardware configuration must be adapted manually. You decide in this case which hardware configuration you are going to accept. If the existing hardware configuration on the CPU is to be retained, you should adopt this in your engineering system prior to downloading. If you want to apply the hardware configuration you changed, you can continue downloading without synchronization.	Automatic synchronization is not possible: The hardware configuration must be adapted manually. One of the existing hardware configurations (online or offline) will be overwritten.

If required, you can use the "Force download to device" command to download blocks without synchronization.

See also

Downloading project data to a device (Page 11)

Downloading project data to a memory card (Page 13)

Uploading project data from a device (Page 15)

1.2.2 Downloading project data to a device

The following section describes the general procedure for downloading project data to a device. You will find details of how certain objects are downloaded and any special points to note in the online help of the product.

Requirement

- The project data is consistent.
- Each device to which you want to download is accessible via an online access.

Procedure

To download the project data to the selected devices, follow these steps:

1. Select one or more devices systems in the project tree.
2. Right-click on a selected element.
   The shortcut menu opens.
3. Select the option you require in the shortcut menu of the "Download to device" submenu.

   Note

   Note that the options available to you depend on the selected device.

   When necessary, the project data is compiled.

   – If you had previously established an online connection, the "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for downloading.
   – If you had not previously established an online connection, the "Extended download to device" dialog opens, and you must first select the interfaces via which you want to establish the online connection to the device. You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command.
     See also: AUTOHOTSPOT
4. Check the messages in the "Load preview" dialog, and select the actions in the "Action" column, if necessary.

   WARNING

   Preventing personal injury and material damage

   Performing the proposed actions while the plant is in operation can cause serious bodily injury and property damage in the event of malfunctions or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   As soon as loading becomes possible, the "Load" button is enabled.
5. Click the "Load" button.
   The loading operation is performed. If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are required for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
6. Click the "Finish" button.

Result

The selected project data was downloaded to the devices.

See also

General information on loading (Page 9)

Downloading project data to a memory card (Page 13)

Uploading project data from a device (Page 15)

1.2.3 Downloading project data to a memory card

You have the option of loading project data to a memory card. For CPUs of the S7-300/400 series you can also explicitly download the user program to a memory card inserted in the CPU.

To download project data to a memory card, you have the following options:

- Dragging project data to a memory card
- Writing project data to a memory card
- Downloading user program to a memory card that is inserted in a CPU of the S7-300/400 series

Requirement

A memory card is displayed.

See also: Accessing memory cards (Page 99)

Downloading project data to a memory card

To download project data to a memory card, follow these steps:

1. In the project tree, drag the project data you want to download to the memory card.
   If necessary, the project data is compiled. The "Load preview" dialog then opens. This dialog displays alarms and recommends actions needed for the loading operation.
2. Check the alarms and select the actions in the "Action" column, if necessary.
   As soon as downloading becomes possible, the "Load" button is enabled.
3. Click the "Load" button.
   The loading operation is performed.

Or:

1. In the project tree, select the project data that you want to download.
2. To do this, right-click the selection and select the "Copy" command from the shortcut menu. You can also use the key combination <Ctrl+C>.
3. Right-click the memory card and select the "Paste" command from the shortcut menu. You can also use the key combination <Ctrl+V>.
   If necessary, the project data is compiled. The "Load preview" dialog then opens. This dialog displays alarms and recommends actions needed for the loading operation.

4. Check the alarms and select the actions in the "Action" column, if necessary.
   As soon as downloading becomes possible, the "Load" button is enabled.
5. Click the "Load" button.
   The loading operation is performed.

Or:

1. In the project tree, select the project data that you want to download.
2. Select the "Card Reader/USB memory > Write to memory card" command in the "Project" menu.
   The "Select memory card" dialog opens.
3. Select a memory card that is compatible with the CPU.
   A button with a green check mark is activated at the bottom of the dialog.
4. Click the button with the green check mark.
   If necessary, the project data is compiled. The "Load preview" dialog then opens. This dialog displays alarms and recommends actions needed for the loading operation.
5. Check the alarms and select the actions in the "Action" column, if necessary.
   As soon as downloading becomes possible, the "Load" button is enabled.
6. Click the "Load" button.
   The loading operation is performed.

Downloading the user program to a memory card in the CPU (S7-300/400 only)

To download the user program to a memory card in a CPU of the S7-300/400 series, follow the steps below:

1. Select a CPU of the S7-300/400 series in the project tree.
2. Select the "Download user program to memory card" command in the "Online" menu.
   The "Load preview" dialog opens. This dialog displays alarms and recommends actions needed for the loading operation.
3. Check the alarms and select the actions in the "Action" column, if necessary.
   As soon as downloading becomes possible, the "Load" button is enabled.
4. Click the "Load" button.
   The loading operation is performed and the "Load results" dialog is displayed. This dialog displays alarms and suggests possible actions.
5. Check the alarms and select the actions in the "Action" column, if necessary.
6. Click the "Finish" button.

See also

General information on loading (Page 9)

Downloading project data to a device (Page 11)

Uploading project data from a device (Page 15)

1.2.4 Uploading project data from a device

The following section describes the general procedure for uploading project data from a device. Which project data you can upload from a device depends on the products installed.

You have the following options for uploading project data from a device to your project:

- Uploading as a new station
  With this option you upload existing project data of a device to your project as a new station.
- Uploading project data of a device
  With this option you upload project data from the device to an existing CPU in the project. You will find the project data that can be loaded in the online help of the product.

In both cases, all instances of library types are connected again with the corresponding version of the type in the project library during uploading. If no suitable type is yet available for an uploaded instance or the correct version of the type does not exist in the project library, the type or the version is added to the project library. You can only upload library types in the state "in test" that are saved on the device to the project, if the released type version on which the test version is based is available in the project library.

Requirement

- A project is open.
- The hardware configuration and software to be uploaded must be compatible with the TIA Portal. If the data on the device was created with a previous program version or with a different configuration software, please make sure they are compatible.

Uploading as a new station

To upload the complete device to your project, follow these steps:

1. Select the project name in the project tree.
2. Select "Upload device as new station (hardware and software)" in the "Online" menu.
   The "Upload device to PG/PC" dialog opens.
3. Select the type of interface you want to use for the uploading operation in the "Type of the PG/PC interface" drop-down list.
4. Select the interface to be used from the "PG/PC interface" drop-down list.
5. Click the "Configure interface" button to the right of the "PG/PC interface" drop-down list to adapt the settings for the selected interface.
   See also: AUTOHOTSPOT
6. Display all compatible devices by selecting the relevant option and clicking the "Start search" command. In the accessible devices table, select the device from which you want to upload project data.
7. Click on "Load".
   Depending on the selected device, a dialog appears in which you have to enter additional information, such as the position of the module rack.
   The project data of the device is uploaded to the project. You can edit it offline and then download it to the device again.

Uploading project data of a device

To upload only project data from one device to your project, follow these steps:

1. Establish an online connection to the device from which you want to download the project data.
   See also: AUTOHOTSPOT
2. Select the device in the project tree.
   The "Upload from device (software)" command in the "Online" menu is enabled.
3. Select the "Upload from device (software)" command in the "Online" menu.
   The "Upload preview" dialog box opens.
4. Check the alarms in the "Upload preview" dialog, and select the necessary actions in the "Action" column.
   As soon as uploading becomes possible, the "Upload from device" button is enabled.
5. Click the "Upload from device" button.
   The loading operation is performed.

Loading of project data from a different project version from a device

When the commands "Upload from device (software)" and "Loading the device as new station (hardware and software)" are executed in the "Online" menu, it is checked whether the project data on the device was created with a different version than the opened version of the TIA Portal.

If the project data was created with a different project version, the "Upload preview" dialog provides you with information on whether all upload requirements are met.

Note the information on the requirements for loading and, if necessary, select a command in the selection menu in the "Action" column.

The "Upload from device" button is enabled as soon as the view has been updated and uploading is possible.

Special features when using data blocks from other product versions

Data blocks that were created with a different product version of the TIA Portal cannot be opened and edited with the current product version. You can find additional information in the section "AUTOHOTSPOT".

See also

General information on loading (Page 9)

Downloading project data to a device (Page 11)

Downloading project data to a memory card (Page 13)

1.2.5 Loading project data from a memory card

You have the following options for uploading project data from a memory card to your project:

- Upload project data of the memory card as a new station
  With this option you upload the project data of a memory card to your project as a new station.
- Upload project data of the memory card to an existing device
  With this option you upload project data of a memory card to an existing device in your project. You will find the project data that can be loaded in the online help of the product.

In both cases, all instances of library types are connected again with the corresponding version of the type in the project library during loading. If no suitable type is yet available for a loaded instance or the correct version of the type does not exist in the project library, the type or the version is added to the project library.

Requirement

- A project is open.
- The memory card is displayed.
  See also: Accessing memory cards (Page 99)
- The hardware configuration and software to be uploaded must be compatible with the TIA Portal. If the data on the memory card was created with a previous program version or with different configuration software, please make sure they are compatible.

Uploading project data as a new station

To upload project data from a memory card to your project, follow these steps:

1. In the project tree, select the project data you want to upload.
2. Select "Upload device as new station (hardware and software)" in the "Online" menu.

Or:

1. In the project tree, drag the memory card folder to the project.

Or:

1. Right-click on the memory card.
2. Select "Copy" in the shortcut menu.
3. Right-click on the project.
4. Select the "Paste" command in the shortcut menu.

Uploading project data to an existing device

To upload the project data of a memory card to an existing device, follow these steps:

1. In the project tree, drag the folder of the memory card to a device in the project or copy the memory card and paste the data into a device.
   The "Upload preview" dialog box opens.
2. Check the alarms in the "Upload preview" dialog, and select the necessary actions in the "Action" column.
   As soon as uploading becomes possible, the "Upload from device" button is enabled.
3. Click the "Upload from device" button.
   The loading operation is performed.

Comparing project data 2

2.1 Basics of project data comparison

Function

You can compare project data of the same type in order to determine possible differences. In principle, the following comparison methods are available:

- Compare offline/online
  With this comparison method you can compare the software of objects of a device with the objects of a project. This is only possible when you establish an online connection to the device.
- Offline/offline comparison
  With this comparison method you can either compare the software or the hardware. When you compare the software, you can compare objects from projects or libraries. The hardware comparison is available for devices from the currently open project or from reference projects. You can decide for the software as well as the hardware comparison whether the comparison should be performed automatically for all objects or whether you want to compare individual objects manually.
- Detailed comparison
  For some objects, for example, blocks, you can also perform a detailed comparison in addition to the offline/online and offline/offline comparison. This involves opening the blocks to be compared beside each other and highlighting the differences.

A simple offline/online comparison is performed as soon as you establish an online connection. During this process, comparable objects in the project tree are marked with icons that represent the result of the comparison. You can also run a more comprehensive offline/online and offline/

offline comparison in the compare editor. When you compare software, you can also select actions for non-identical objects in the comparison.

Note

- Not all objects allow all types of comparison. Which comparison method you can use for which project data depends on the products installed.
- Compile your user program before you start a comparison or detailed comparison. After each change of the program during a comparison, repeat this step before you update the result of the comparison. This ensures that the comparison shows the current status.
- When monitoring a module, note that a difference of the time stamp can also exist with an offline/online comparison if the block is recompiled with code change. The checksum remains unchanged. In this case download the block to the device and restart the monitoring.
- In the following cases, the time stamps of the objects are used as comparison criteria:
  - Export and re-import via the Openness interface
  - Block calls
  - Access to the global data of a data block (DB)

  This can result in different checksums, even though the objects are identical.

Comparison using checksums

The offline/online and offline/offline comparison of software objects uses checksums that are generated for certain data of the objects. The objects are only the same if their checksums are identical.

Note

Time stamps continue to be used for offline/online comparison for CPUs of the S7-300/400 series.

The data of an object are divided into the following two categories:

- Source data
  Source data are all the data of an object which you can directly influence, such as the name of the object, source code, comments or programming language. The properties that you can edit here may differ depending on the object.
- Target data
  All compiler and runtime data are the target data. These data may have no direct influence, since they are produced by the system based on their source data. Once again here, the characteristics differ depending on the object.

The following table provides an overview of the areas of the property comparison:

Category	Area	Description
Source data	Safety	Checksum for safety objects You can find additional information about the comparison of safety objects in the "Comparing safety programs" section of the online help for SIMATIC Safety.
	Interface without comments	Checksum of all the tags of the block interface of the first layer. Comments of the tags are not part of the checksum.
	Interface of published blocks without comments (software units)	Checksum of all interface checksums of published objects of a software unit. Comments are not part of the checksum.
	Textual block interface	Checksum of all tags, comments and formatting in the textual block interface of SCL blocks.
	Code without comments	Checksum of the source code. Block and network comments are not included in the checksum. Comments that are not multilingual, for example in STL, and the reassignment of free comments to other boxes in LAD and FBD, are not included in the checksum.
	Comments (multi-language)	Checksum of all block comments and network comments
	Language configuration	Compare offline/online Displays the languages that are loaded to the device during the loading process. Offline/offline comparison: Displays the enabled languages in the project.
	Event	Checksum of all event-relevant data of an organization block (OB).
	Properties	Checksum of all configurable properties of an object.
	Extended properties of program elements in software units	Checksum of the publication status of objects in software units.
	Configuration of technology objects	Checksum of the technology-specific settings of a technology object.
	Alarm configuration	Checksum on all alarm-specific settings of an object.
	Supervision configuration	Checksum of all supervision settings of an object.
	Text lists	Checksum over all text lists of an object.
	Tags without comments	Checksum over all tags of the selected PLC tag table. The "Retentivity" and "Monitoring" columns as well as the comments of the tags are not part of the checksum.
	Constants without comments	Checksum over all user constants of the selected PLC tag table. System constants and the comments of the user constants are not part of the checksum.
Target data	Compilation and runtime data	Checksum of data for compiling and relevant for loading.
	Time stamp	Time stamps for S7-300/400 CPUs that are created by the system for compiling and downloading. Only available when an online connection is available.

Note

- Note that the object determines which checksums are actually used for the comparison.
- The value settings of technology objects are also transferred to the checksum of the block interface.
- For know-how protected blocks that were created in an older version, the checksums "Code without comments" and "Comments (multi-language)" are not displayed under "Source data".

See also

Comparing offline/online (Page 23)

Carrying out offline/offline comparisons (Page 24)

Running a detailed comparison (Page 34)

Using the compare editor (Page 26)

2.2 Comparing offline/online

The offline/online comparison lets you compare objects of a device with objects of a project.

Requirement

The project tree is open.

Procedure

To perform an offline/online comparison, follow these steps:

1. Select a device in the project tree that allows offline/online comparison.
2. Select the "Compare > Offline/online" command in the shortcut menu.
3. If you have not already established an online connection to this device, the "Go online" dialog opens. In this case, set all the necessary parameters for the connection and click "Connect".
   The online connection is established and compare editor opens.

Result

All objects that exist online and offline are displayed. The symbols in the compare editor and in the project tree show you the status of the objects. In the compare editor, you can now define certain actions for the objects, depending on their status.

See also

Basics of project data comparison (Page 19)

Carrying out offline/offline comparisons (Page 24)

Using the compare editor (Page 26)

Running a detailed comparison (Page 34)

2.3 Carrying out offline/offline comparisons

With offline/offline comparison you have the option to compare the project data of two devices. You can carry out a software as well as a hardware comparison. When you compare the software, you can compare objects from projects or libraries. The hardware comparison is available for devices from the currently open project or from reference projects. You can decide whether the comparison should be performed automatically for all objects or whether you want to compare individual objects manually. For a software comparison, the following options are available as well:

- For objects that are not identical, you can specify actions to eliminate the differences.
- You can specify the criteria that should be used for the comparison.

You can drag any other device to the drop area at any time to perform further comparisons.

Requirement

The project tree is open.

Performing an offline/offline comparison for software

To perform an offline/offline comparison for software, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane.
   All existing objects of the selected devices are displayed depending on the settings of the compare editor in the "Software" tab and an automatic comparison is carried out. You can identify the status of the objects based on the symbols in the compare editor. If needed, specify an action for non-identical objects to eliminate the differences.
4. If needed, define the comparison criteria.
5. Select an object to display the details of the property comparison.
6. If you want to carry out a manual comparison, click on the button for switching between automatic and manual comparison in the status and action area. Then select the objects that you want to compare.
   The properties comparison is displayed. You can identify the status of the objects based on the symbols.

Performing an offline/offline comparison for hardware

To perform an offline/offline comparison for hardware, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane.

4. Open the "Hardware" tab.
5. If you want to carry out a manual comparison, click on the button for switching between automatic and manual comparison in the status and action area. Then select the objects that you want to compare.
   The properties comparison is displayed. You can identify the status of the objects based on the symbols.

See also

Basics of project data comparison (Page 19)

Comparing offline/online (Page 23)

Using the compare editor (Page 26)

Running a detailed comparison (Page 34)

2.4 Using the compare editor

2.4.1 Overview of the compare editor

Function

The compare editor gives an overview of the results of a comparison in a table. The appearance varies slightly depending on whether it is an offline/online comparison or a hardware/software comparison.

Layout of the compare editor

The following figure shows the layout of the compare editor for an offline/online comparison:

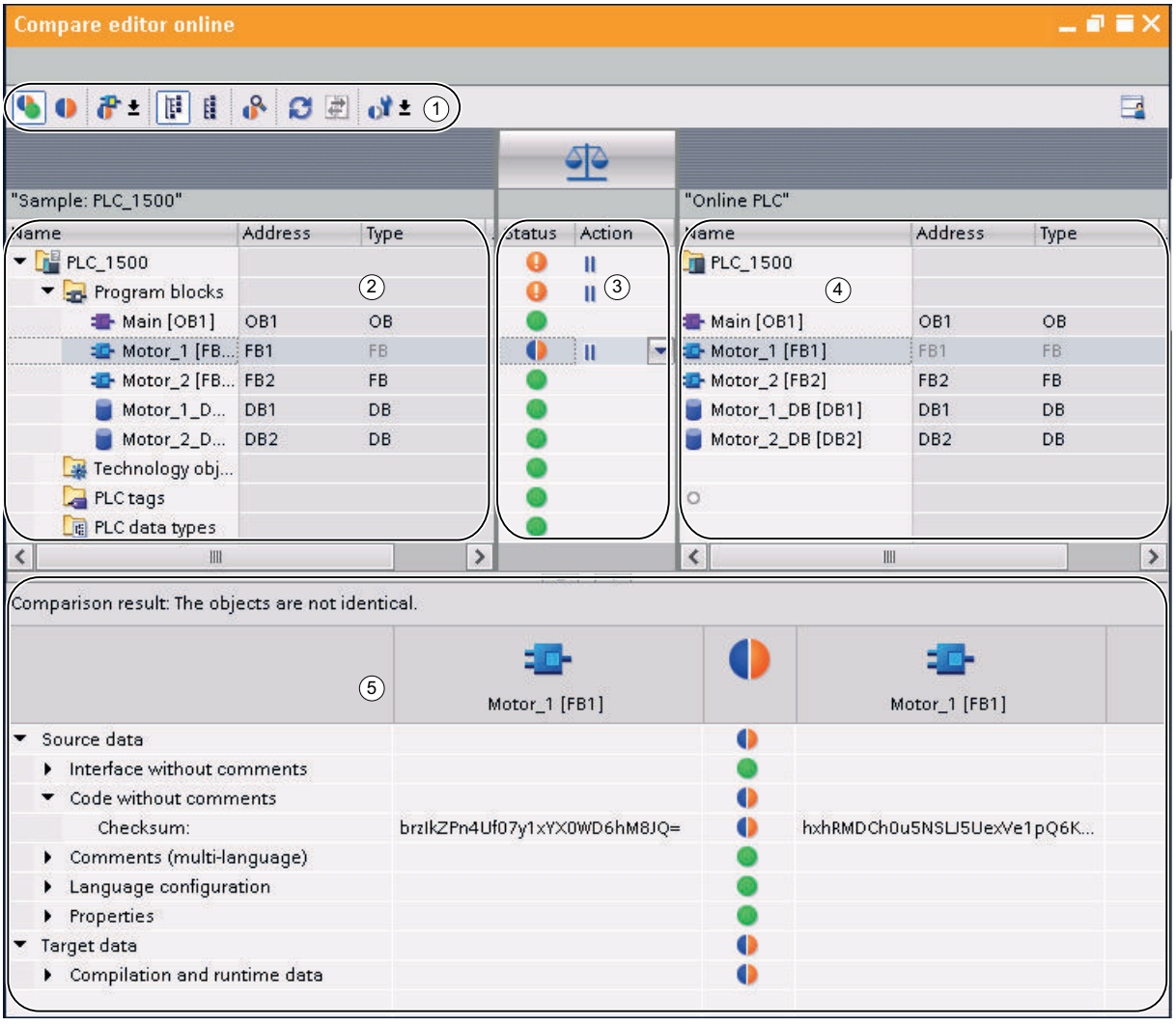

① Compare editor toolbar

② Left comparison table

③ Status and action area

④ Right comparison table

⑤ Property comparison

The following figure shows the layout of the compare editor for an offline/offline comparison (software):

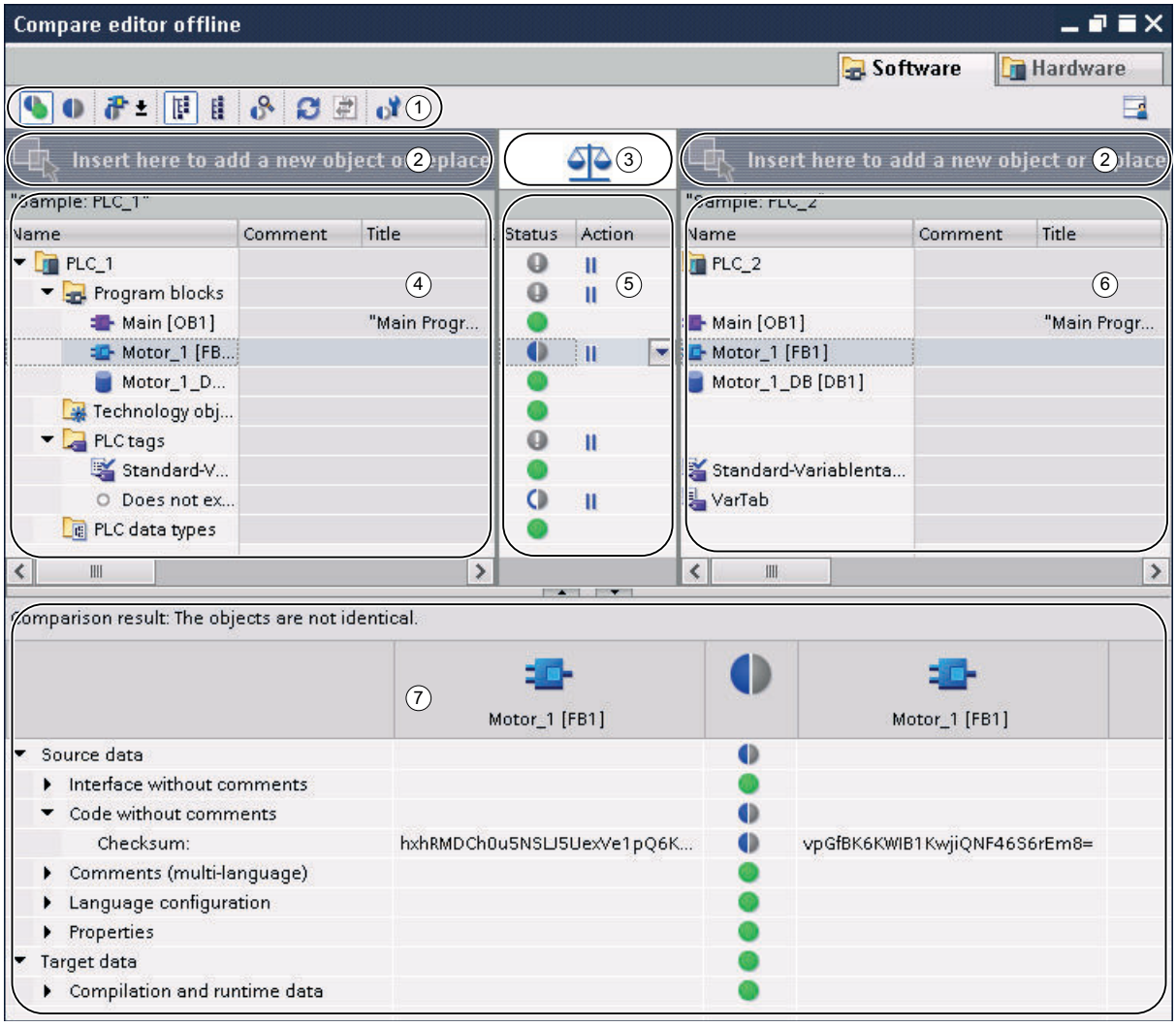

① Compare editor toolbar

② Drop areas

③ Button to toggle between automatic and manual comparison

④ Left comparison table

⑤ Status and action area

⑥ Right comparison table

⑦ Property comparison

The following figure shows the layout of the compare editor for an offline/offline comparison (hardware):

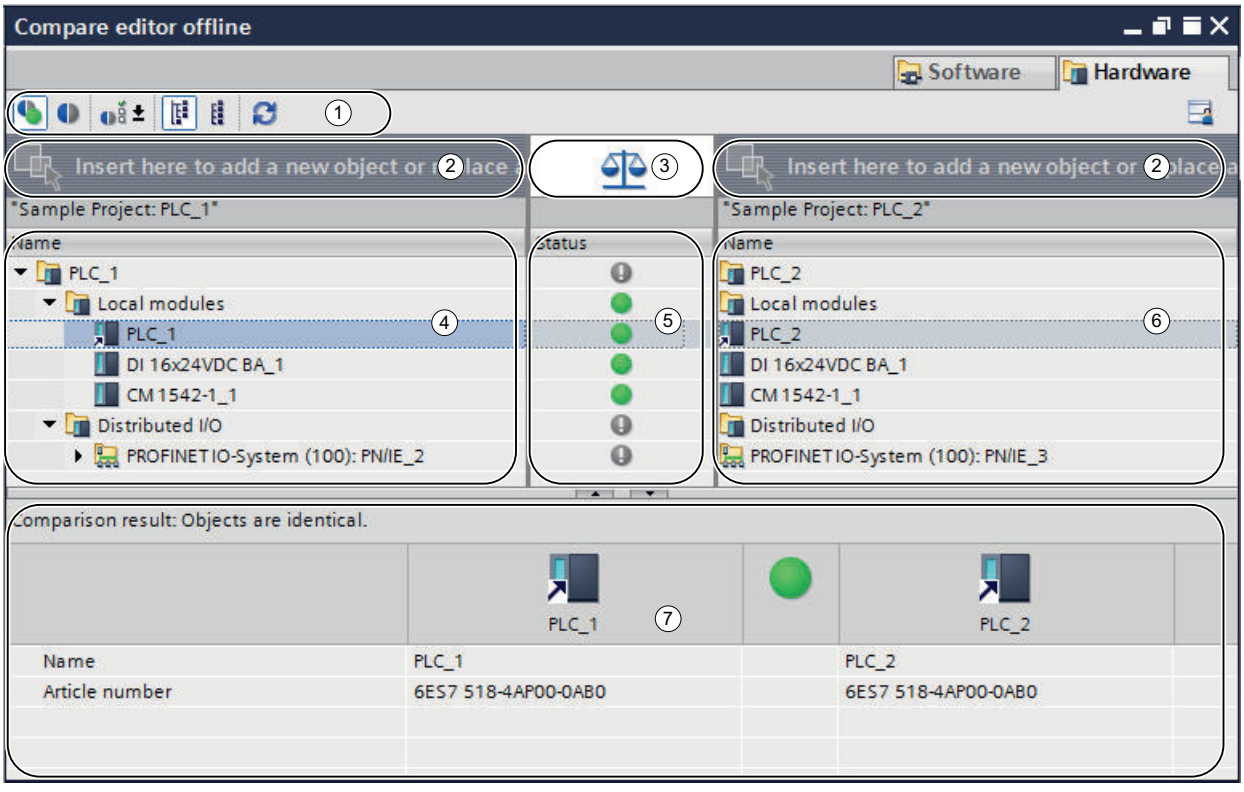

① Compare editor toolbar

② Drop areas

③ Button to toggle between automatic and manual comparison

④ Left comparison table

⑤ Status area

⑥ Right comparison table

⑦ Property comparison

Compare editor toolbar

With the toolbar, you can access the following compare editor functions:

- Show identical and different objects
  You can show identical objects if you want to view the comparison in full.
- Show only objects with differences
  You can hide identical objects to make the comparison easier to follow.
- Show additional filters available (only offline/online comparison and offline/offline comparison for software)
  You can define which objects are to be compared.

- Display available assignment criteria (only offline/offline comparison for hardware)
  You can specify the criterion according to which the modules are to be assigned to each other for the comparison.
- Changing the view
  You can choose between a hierarchical and a flat view. In the hierarchical view, the devices are shown in their structure; in the flat view, the objects of the devices are listed without structure.
- Start detailed comparison (only offline/online comparison and offline/offline comparison for software)
  You can start a detailed comparison for objects to show the individual differences. This function is, however, not available for every object.
- Refresh the view
  After you have modified objects, you can update the comparison results using this function.
- Execute actions (only offline/online comparison and offline/offline comparison for software)
  You can synchronize non-identical objects using specific actions.
- Set comparison criteria (only offline/online comparison and offline/offline comparison for software)
  You can specify which criteria are to be used in the comparison. Disabled criteria do not affect the overall result of the comparison.

Drop areas

In the case of an offline/offline comparison, you can drag the devices you want to compare into the drop areas. In the case of a software comparison, the devices to be compared can originate from the opened project, from reference projects, from the project library or from global libraries. However, note that you can only drop complete libraries into the right drop area. In the case of a hardware comparison, you can compare devices from the opened project or from reference projects.

Button to toggle between automatic and manual comparison

With an offline/offline comparison, you can switch between automatic and manual comparison. In the case of automatic comparison, the objects to be compared are assigned automatically to each other. In the case of a manual comparison, you can select the objects that are to be compared.

Comparison tables

Comparison tables show the objects of the devices being compared to one another.

The following table shows the meaning of the columns of the comparison table:

Column	Description
Name	Name of the compare object
Comment	Comment on the compare object
Title	Title of the compare object
Address	Address of the compare object
Numbering	Type of numbering for the comparison object

Column	Description
Type	Type of compare object
Language	Programming language set for the compare object.
Time stamp interface	Time of the last modification to the block interface
Time stamp code	Time of the last modification to the source code
Author	Name of the author of the compare object
Version	Version of the compare object
Family	Name of the object family
Load memory	Memory usage of the load memory of the compare object
Work memory	Memory usage of the work memory of the compare object
Modified on	Time of last modification
Optimized block access	Indicates whether "Optimized block access" is enabled for a block.
Signature	Signature of the comparison object (SIMATIC Safety*)
Interface signature	Signature of the block interface of the comparison object (SIMATIC Safety *)

* You can find additional information about the comparison of safety objects in the "Comparing safety programs" section of the online help for SIMATIC Safety.

Not all columns are available in every comparison type. For the hardware comparison, for example, the comparison tables contain only the "Name" column.

Not all columns are shown in the default setting. However, as in all table editors, you can show or hide the columns as required and sort according to individual columns.

Status and action area

The status and action area offers the following options:

- You can view the results of automatic comparison. The results are displayed with symbols.
- In an offline/online comparison and an offline/offline comparison for software, you can specify actions for non-identical objects.

Status and action symbols

The following table shows the symbols for the comparison results for an offline/online comparison:

Symbol	Description
	Folder contains objects whose online and offline versions differ
	Result of comparison is either not known or cannot be displayed for the following reasons: • No rights available for access to a protected CPU. • The loading process on the CPU was executed with a TIA Portal version earlier than V14.
	Online and offline versions of the object are identical
	Online and offline versions of the object are different

Symbol	Description
	Object only exists offline
	Object only exists online
	This comparison criterion is disabled and the associated checksum is not incorporated into the comparison result.

The following table shows the comparison results symbols for an offline/offline comparison:

Symbol	Description
	Reference program
	Version compared
	Folder contains objects of which the versions compared differ
	Results of the offline/offline comparison are not known
	The versions of the object compared are identical
	The versions of the object compared differ
	Object only exists in the reference program
	Object only exists in the version compared
	Hardware comparison only: Although the lower-level objects of the container are identical, there are differences between the containers themselves. Such a container may be a rack, for example.
	Hardware comparison only: The lower-level objects of the container are different. There are also differences between the containers. Such a container may be a rack, for example.
	This comparison criterion is disabled and the associated checksum is not incorporated into the comparison result.

The following table shows the symbols for possible actions in a software comparison:

Symbol	Description
	No action
	Overwrite the object of the compared version with the object from the reference program
	Overwrite the object of the output program with the object from the compared version
	Different actions for the compare objects in the folder

Property comparison

The property comparison compares the properties of the selected compare objects. The result is displayed with symbols. Only the property comparison is made with a manual comparison so that the status and action area remains empty. With automatic comparison, you can perform the property comparison in addition to the comparison in the comparison tables.

See also

Basics of project data comparison (Page 19)

Comparing offline/online (Page 23)

Carrying out offline/offline comparisons (Page 24)

Show and hide table columns (Page 32)

Filtering the compare editor view (Page 33)

Running a detailed comparison (Page 34)

Updating the comparison results (Page 36)

Set comparison criteria (Page 37)

Synchronizing non-identical objects (Page 38)

Changing the view (Page 40)

2.4.2 Show and hide table columns

In a software comparison, you can show or hide the columns of comparison tables as required.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.
5. In order to show or hide multiple columns, click "More" and select or clear the check boxes of the corresponding columns in the "Show/Hide" dialog.

Result

The columns in both the left and the right comparison table are shown or hidden.

See also

Basics of project data comparison (Page 19)
Comparing offline/online (Page 23)
Carrying out offline/offline comparisons (Page 24)
Overview of the compare editor (Page 26)
Filtering the compare editor view (Page 33)
Running a detailed comparison (Page 34)
Updating the comparison results (Page 36)
Set comparison criteria (Page 37)
Synchronizing non-identical objects (Page 38)
Changing the view (Page 40)

2.4.3 Filtering the compare editor view

You can improve the clarity of the compare editor using the following filters:

- Hiding identical comparison objects
  You can hide comparison objects which have identical offline/online or offline/offline versions. Any such comparison objects you have hidden can also be shown again at any time.
- Selecting displayed objects
  In an offline/online comparison or an offline/offline comparison for software, you can specify the objects for which the comparison results are to be shown.

Requirement

The compare editor is open.

Hiding identical comparison objects

To hide identical objects, follow these steps:

1. Click on the "Show only objects with differences" button in the toolbar.
   Only the elements that differ online and offline are displayed.

Showing identical comparison objects

To show identical objects again, follow these steps:

1. Click on the "Show identical and different objects" button in the toolbar.
   All elements will be displayed.

Selecting displayed objects

To select the objects for which comparison results should be displayed, follow these steps:

1. Perform an offline/online or an offline/offline comparison for software.
2. Click the arrow of the button "Show additional filters available" in the toolbar.
3. Select the required filter.

See also

Basics of project data comparison (Page 19)
Comparing offline/online (Page 23)
Carrying out offline/offline comparisons (Page 24)
Overview of the compare editor (Page 26)
Show and hide table columns (Page 32)
Running a detailed comparison (Page 34)
Updating the comparison results (Page 36)
Set comparison criteria (Page 37)
Synchronizing non-identical objects (Page 38)
Changing the view (Page 40)

2.4.4 Running a detailed comparison

Note

Please note the following:

- Not all objects allow a detailed comparison. The project data for which you can perform a detailed comparison depends on the products installed. A detailed comparison of the hardware components is not available for hardware comparison.
- For blocks, PLC tags and PLC data types, you can start the detail comparison directly from the project tree. You can find additional information in the section "Comparing PLC programs".

Procedure

Proceed as follows to perform a detailed comparison:

1. First, perform an offline/online or an offline/offline comparison for software.
   The compare editor opens.

   Note

   You can only perform a detailed comparison for objects that are listed in the left as well as the right comparison table.

2. In the compare editor, select the object for which you want to perform a detailed comparison.
3. Click the "Start detailed comparison" button in the toolbar.

Result

The object and its associated comparison partners are opened in their appropriate editors. The editors are arranged next to each other to provide you with a quick overview of the differences. You can find more information with the description of the respective data type.

See also

Basics of project data comparison (Page 19)

Comparing offline/online (Page 23)

Carrying out offline/offline comparisons (Page 24)

Overview of the compare editor (Page 26)

Show and hide table columns (Page 32)

Filtering the compare editor view (Page 33)

Updating the comparison results (Page 36)

Set comparison criteria (Page 37)

Synchronizing non-identical objects (Page 38)

Changing the view (Page 40)

2.4.5 Updating the comparison results

As soon as you change an object, the comparison results are no longer valid and must be updated.

Note

For offline/online comparisons, you should note that changes in the device may result in the system automatically updating the comparison editor if objects in the comparison are affected by the change. This can have the following results:

- Some of the actions you have defined may become invalid, for example if the device no longer contains the object in question. Objects with such invalid actions will be highlighted so you can define new, valid actions.
- The selection you made before the automatic update may also be cancelled.

Requirement

The comparison editor is open.

Procedure

To update the comparison results, follow these steps:

1. Click the "Refresh view" button in the toolbar.
   The comparison results are updated.

   Note

   Please note that the "Refresh view" button will not be available while the comparison editor is loading or synchronizing content.

See also

Basics of project data comparison (Page 19)

Comparing offline/online (Page 23)

Carrying out offline/offline comparisons (Page 24)

Overview of the compare editor (Page 26)

Show and hide table columns (Page 32)

Filtering the compare editor view (Page 33)

Running a detailed comparison (Page 34)

Set comparison criteria (Page 37)

Synchronizing non-identical objects (Page 38)

Changing the view (Page 40)

2.4.6 Set comparison criteria

You have the option to set to criteria to be taken into account for a comparison. The specified criteria are applied to all objects of a comparison, if they are available for an object. If a criterion for an object is not available, for example, "Code without comments" for data blocks, the comparison result for this checksum is considered to "match".

For a criterion that you disabled, you can still view the checksums for the overall comparison result although it is not used. Disabled comparison criteria for the property comparison are indicated by the filter icon. The filter symbol can also be displayed in the upper area of the comparison editor when no checksums should be used in the comparison result based on the selection of comparison criteria.

Note

With an offline/online comparison, the comparison criteria have no influence on the result of comparison in the project tree. Note that there may be different results in the project tree and comparison editor.

Procedure

To define the criteria to be taken into account for the comparison, follow these steps:

1. In the toolbar, click on the arrow of the "Display further comparison criteria" button.
2. Select all the criteria that should be considered for the comparison.
3. Clear all the criteria that should not be considered for the comparison.
4. Click the button with the green check mark at the bottom of the drop-down list.
5. Click on the button with the x to close the drop-down list without applying the changes.

Result

Only the enabled criteria are taken into account for the comparison.

See also

Basics of project data comparison (Page 19)
Comparing offline/online (Page 23)
Carrying out offline/offline comparisons (Page 24)
Overview of the compare editor (Page 26)
Show and hide table columns (Page 32)
Filtering the compare editor view (Page 33)
Running a detailed comparison (Page 34)
Updating the comparison results (Page 36)
Synchronizing non-identical objects (Page 38)
Changing the view (Page 40)

2.4.7 Synchronizing non-identical objects

2.4.7.1 Specifying actions

If you have performed a comparison, you can specify the actions to be performed for non-identical objects in the compare editor. You cannot select any actions for identical objects. Note that you cannot execute any actions during hardware comparison.

In the case of an offline/online comparison, only synchronization actions in one direction are permitted, in order to retain program consistency. Thus, for example, you can load multiple blocks to a device or from a device, but you cannot perform a combination of loading actions in one synchronization action. In this case, the first action you set in the compare editor determines the synchronization direction. For example, if you specify for a block that the offline block is to be loaded to the device, then the other objects can also only be loaded to the device via a synchronization action. To load objects from the device again, first select the "No action" option. You can then specify the action settings again as required. Or, you can perform a new comparison.

Note

Note the following CPU-specific aspects when defining actions:

- S7-300/400: You can define actions for the "Program blocks" folder, for folders you have created yourself or for individual blocks.
- S7-1200/1500: You can define actions for the "Program blocks" folder, for folders you have created yourself or for individual blocks. If you have performed an offline/online comparison and select download to the device as action, a consistent download is executed. If you upload the object from the device to the project, however, you can also upload individual blocks.

Requirement

The compare editor is open.

Procedure

To select an action for a non-identical object, follow these steps:

1. In the status and action area, click in the "Action" column on the cell of the object for which you want to define an action.
   The cell changes to a drop-down list.
2. Click on the drop-down list.
3. Select the action you want.
   The action set will be carried out for the object in question the next time synchronization is performed.
   If you have accidentally changed the action you had selected, you can undo the change before the next synchronization.

4. To restore the previously set action selection, right-click the action in the status and action area that you want to restore.
5. Select the "Restore last selection" command in the shortcut menu.

See also

Basics of project data comparison (Page 19)
Comparing offline/online (Page 23)
Carrying out offline/offline comparisons (Page 24)
Overview of the compare editor (Page 26)
Show and hide table columns (Page 32)
Filtering the compare editor view (Page 33)
Updating the comparison results (Page 36)
Synchronizing objects (Page 39)

2.4.7.2 Synchronizing objects

Synchronization executes the actions you have specified for non-identical objects. Note, however, that in the case of an offline/online comparison you can only perform actions in one direction in one synchronization action.

Requirement

- The compare editor is open.
- The desired actions have been selected.

Procedure

To synchronize objects, follow these steps:

1. Click the "Execute actions" button in the toolbar.

Result

The actions you specified for the objects are performed.

See also

Basics of project data comparison (Page 19)
Comparing offline/online (Page 23)
Carrying out offline/offline comparisons (Page 24)
Overview of the compare editor (Page 26)
Show and hide table columns (Page 32)

Filtering the compare editor view (Page 33)

Updating the comparison results (Page 36)

Specifying actions (Page 38)

2.4.8 Changing the view

You can choose between a hierarchical and a flat view for the left comparison table. In the hierarchical view, the devices are shown in their structure; in the flat view, the objects of the devices are listed without structure. In the right comparison table, the objects are always displayed flat.

Setting the hierarchical view

To set the hierarchical view, follow these steps:

1. Click the "Display in hierarchical view" button in the toolbar of the compare editor.

Setting the flat view

To set the flat view, follow these steps:

1. Click the "Display in flat view" button in the toolbar of the compare editor.

See also

Basics of project data comparison (Page 19)

Comparing offline/online (Page 23)

Carrying out offline/offline comparisons (Page 24)

Overview of the compare editor (Page 26)

Show and hide table columns (Page 32)

Filtering the compare editor view (Page 33)

Running a detailed comparison (Page 34)

Updating the comparison results (Page 36)

Set comparison criteria (Page 37)

Synchronizing non-identical objects (Page 38)

Protecting project data 3

3.1 Protection concept for project data

Introduction

You can protect your project data from unauthorized access. These include, for example:

- Access protection for devices
- Copy and display protection of objects
- Restrictions for printouts of know-how-protected objects

For objects with know-how protection, this protection is also retained after the object is pasted into a library. Note that every protection mechanism is not available for all objects. How to protect specific objects is described in the online help of the product.

Companies can also specify the use of a password provider to protect blocks. To do so, the central settings must be adjusted and all users must integrate the corresponding password providers in the TIA Portal. The passwords provided by a password provider can subsequently be used. You can find additional information in the section "Protecting blocks".

Revoking access rights for devices

If you want to execute a function that is password-protected by means of the device protection level, you are prompted to enter a password. When the password is entered correctly, you can execute the required function. The access right is retained on the device until you close the TIA Portal.

If you want to reactivate password protection while the TIA Portal is open, you can explicitly revoke the access rights for a device. As a result, certain functions for the protected device cannot be executed until the correct password is entered again. You specify the functions for which a password must be entered when you assign the device protection level.

Sharing protected project data

Always pass on the protected project data as project archive or library archive. In this way, you ensure that the protection measures cannot be bypassed.

WARNING

Handling know-how-protected project data

If you remove the know-how protection of an object and the TIA Portal is unexpectedly terminated before the next saving, the corresponding project data is stored unprotected in the "CrashDumpReport" and could be accessible to unauthorized persons.

To prevent access to data that is actually protected, proceed as follows:

1. Open the directory ..\Users\<user_name>\AppData\Roaming\Siemens\Automation \<tia_version>\UserDataDumps\<project_name>
2. Delete all files with the name extension "*.UDD".

See also

Printing project data (Page 62)

3.2 Revoking access rights for devices

Requirement

- A protection level has been set for the device.
- A protected function for the device has been enabled by entering the password.

Procedure

To revoke the access rights for the device, follow these steps:

1. Select the device for which you want to revoke access rights in the project tree.
2. Select the "Delete access rights" command in the "Online" menu.

Result

The access rights are revoked, and starting from now the user will be prompted to enter the password again to execute a password-protected function on the device. The function can only be executed if the correct password is entered.

If the device has an online connection, it will be disconnected.

See also

Protection concept for project data (Page 41)

Printing project contents 4

4.1 Printing project documentation

4.1.1 Documentation settings

Introduction

Once a project is created, the contents can be printed in an easy-to-read format. You may print the entire project or individual objects within the project. A well-structured printout is helpful when editing the project or performing service work. The printout can also be used for your customer presentations or as full system documentation.

You can prepare the project in the form of standardized circuit manuals and print it in a uniform layout. You can limit the scope of the printout. You have the option to print to the entire project, individual objects along with their properties, or a compact overview of the project. In addition, you can print the contents of an open editor.

Improving the printout with frames and cover pages

You can design the appearance of the printed pages according to your own requirements, for example, to add your own company logo or the corporate design of your company in the project documentation. You can create any number of design variants as frames and cover pages. The frames and cover pages are stored in the project tree under the item "Documentation settings" and are part of the project. You can insert placeholders for data from previously entered document information within the frames and cover pages. These will be filled automatically with the appropriate metadata during printing.

If you want to avoid designing your own template, there are ready-made frames and covers pages available. These include templates complying with the ISO standard for technical documentation.

Modular structure of a printout

An printout generally consists of the following components:

- Cover page (only when printing from the project tree)
- Table of contents (only when printing from the project tree)
- Name and path of an object within the project tree
- Object data

Printout of the cover page or the table of contents can be deactivated in the "Print" dialog.

See also

Creating frames (Page 51)
Creating a cover page (Page 51)
Editing cover pages and frames (Page 53)
Entering document information (Page 49)
Print function for module labels (Page 64)

4.1.2 Printout of project contents

Availability of print function

The following contents can be printed:

- An entire project in the project tree
- One or more project-related objects in the project tree
- Contents of an editor
- Tables
- Libraries
- Diagnostics view of the Inspector window

It is not possible to print in the following areas:

- Portal view
- Detailed view
- Overview window
- Compare editor
- All tabs of the Inspector window, except the diagnostics view
- All task cards, except the libraries
- Most of the dialogs
- Properties and devices of the programming device/PC not related to the project, for example online portals and connected card readers.

Scope of printout

To be able to print, at least one printable element has to be selected.

If a selected object is printed, all subordinate objects are also printed. For example, if a device is selected in the project tree, all of its data is also printed. If you select the entire project in the project tree for printing, all project contents are printed with the exception of the graphical views. These have to be printed separately. Items in the project tree that are not part of the project cannot be printed. For example, this includes online portals and connected card readers and USB memory devices.

When table contents are printed, all lines in the table in which a cell is selected are printed. In order to print one or more table columns, the desired columns must be selected. If no individual cells or columns are selected, the entire table is printed.

Limitations when printing

In general, it is possible to print all objects that can be visualized on the user interface. Conversely, this means that you cannot print objects that you do not have access to. If a printout fails, possible reasons may include the following:

- A valid license does not exist for displaying an object.
- There is no device description for an object.
- A software component needed to display an object is not installed.

See also

Printing project data (Page 62)

4.1.3 Changing the print settings

Changing the print settings

You can specify general print settings that are retained even after the TIA Portal is closed and re-opened. Some settings are dependent on the products installed. The following settings are possible in every case:

Always print table data as pairs of values

If this option is selected, tables are not printed in tabular format but rather as a pairs of key and value.

Example:

Object name	Property 1	Property 2
Object A	Value A1	Value A2
Object B	Value B1	Value B2

In this case, the printout has the following appearance:

Object A

Property 1: Value A1

Property 2: Value A2

Object B

Property 1: Value B1

Property 2: Value B2

Printing mask editors

- Always print data in tables
  All parameters of technology objects are printed in tabular format.
- Print mask graphics if possible
  If the utilized editor supports this function, the contents of the editor are not printed as a table but rather as a complete graphic as it appears on the screen.

Procedure

To change the print settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group.
3. Select the desired default settings in the "Print settings" area.
   The changes are applied immediately and are retained for all projects, even after the TIA Portal is closed.

4.1.4 Specifying the print layout

Specifying the print layout

If you do not want to rely on ready-made print templates, you can specify your own cover page or your own layout for the individual pages. Your designs are saved together with the respective project.

Your designs for the cover page and your templates for the page layout can be found in the project tree under the "Documentation information" group. You will also find metadata on the project there under the entry "Document information". For subsequent print operations, you can customize the appearance of the printout in the "Print" dialog using the saved cover pages and page layout templates and the available metadata.

Designing the cover page

The cover page can be customized. You can insert a background graphic and provide placeholders for text on the page. The placeholders are automatically filled with data from a documentation information during printing.

Cover pages are located in the project tree under the "Documentation information > Cover pages" group.

Designing the content page

The regular pages of a printout can contain the following elements:

- Frame with static content, such as a company logo
- Placeholders for text, such as the name of the project, the page number, and the time the printout was started
  Several different values for the individual placeholders can be specified in the document information. Other values, such as the project name, are preassigned and are inserted automatically during printing.
- Footnote
  The footnote is always output below the content area.
- Content area
  You can specify an area where the printed content is to be embedded.

The design of the content pages is saved in Frames. The individual frames are located in the project tree under the "Documentation information > Frames" group.

4.1.5 Entering document information

You can enter metadata in the document information for every project. In addition, a print frame and a cover page are specified in the document information. You can create different information, if required, to enable you to quickly switch between different document information containing different information, frames, cover pages, page sizes, and page orientations when printing. For example, this is useful if you want to generate printouts in different languages and different document information is provided for each language.

In the documentation editor, you can specify placeholders on the cover page or in the frame of the regular pages. These placeholders can be automatically replaced with metadata from the documentation information during printing.

The various document information are therefore part of the printing function and specify the print layout and print content.

Procedure

To add metadata, follow these steps:

1. To create new document information, double-click "Add new document information" under "Documentation information > Document information" in the project tree.
   The new document information is created and opened immediately.
2. Enter a name for the set in the "Name" field.
3. Fill in the individual fields with the metadata for the project.

4.1.6 Managing cover pages and frames

4.1.6.1 Using cover pages and frames

Uses for cover pages

You can give your plant documentation printouts a professional appearance by adding a cover page. You can design your own cover page or use ready-made cover pages. Ready-made cover pages can be adapted and stored again as a template.

Cover pages can be saved in global libraries where they are available for use across projects.

Cover pages are designed for use as a right printed page only.

Uses of frames

You can embed the regular pages of your plant documentation inside a consistently uniform page frame. The frame can contain placeholders for project metadata, which is stored in the document information. It can also contain graphic elements that you design yourself.

You can create your own frames or rely on ready-made page frames. You can adapt a ready-made page frame and then store it again as a new frame.

Like cover pages, frames can be saved in global libraries where they are available for use across projects.

Frames are designed for use on right printed pages only.

Cover pages and templates in the project tree

Cover pages and frames associated with the project are stored in the project tree under the entry "Documentation information". There are separate folders here for frames and cover pages.

The following actions are available in the project tree for cover pages and frames.

- Creating your own subfolders
- Copying and pasting
- Inserting cover pages and frames from the "Documentation templates" system library
- Copying cover pages and templates to a global library

Cover pages and templates in libraries

The "Documentation templates" system library contains a few cover pages and templates that are available in every project. The cover pages and templates can be moved from there to the project tree using a drag-and-drop operation. You can then adapt the cover pages and templates in the project tree according to the requirements of your project.

Cover pages and templates can be moved from the project tree to a global library. Afterwards, these are available in every project.

See also

Using ready-made frames and cover pages (Page 52)

Designing cover pages and frames (Page 53)

4.1.6.2 Creating frames

You can create any number of frames for each project. The frames are stored in the project tree below the "Documentation information > Frames" group. You can assign a frame to all document information. When you select document information for printing, its associated frame is used.

Procedure

To create a new frame, follow these steps:

1. Double-click the entry "Add new frame" below the "Documentation information > Frames" group in the project tree.
   The "Creating frames" dialog opens.
2. Enter a name for the frame in the "Name" field.
3. Choose the paper size from the "Paper type" drop-down list.
4. Choose whether the page is to be created in portrait or landscape format in the "Orientation" drop-down list.

Click the "Add" button.

Result

A new frame is created. The frame is then opened automatically in the documentation editor where it can be edited.

See also

Editing cover pages and frames (Page 53)

Creating a cover page (Page 51)

4.1.6.3 Creating a cover page

You can create any number of cover pages for the printout for each project. The cover pages are stored in the project tree below the the "Documentation information > Cover pages" group. You can assign a cover page to all document information. When you select specific document information for printing, its associated cover page is used.

Procedure

To create a new cover page, follow these steps:

1. Double-click the entry "Add new cover page" below the "Documentation information > Cover pages" group in the project tree.
   The "Add new cover page" dialog box opens.
2. Enter a name for the cover page in the "Name" field.
3. Choose the paper size from the "Paper type" drop-down list.
4. Choose whether the page is to be created in portrait or landscape format in the "Orientation" drop-down list.

Click the "Add" button.

Result

A new cover page is created. The cover page is then opened automatically in the documentation editor where it can be edited.

See also

Editing cover pages and frames (Page 53)

Creating frames (Page 51)

4.1.6.4 Using ready-made frames and cover pages

The TIA Portal comes with some ready-made frames and cover pages. These can change according to your wishes.

Procedure

To create and edit the ready-made frames and cover pages, follow these steps:

1. Open the "Global libraries" pane in the "Libraries" task card.
2. In the "Templates" folder, open the "Cover Pages" or "Frames" folder.
3. Drag a cover page or a frame from one of the folders into the project tree and drop it into one of the following folders:
   - For frames: "Document information > Frames"
   - For cover pages: "Document information > Cover pages".

   The ready-made frame or cover page can now be used in the project.
4. Double-click on the new entry in the project tree click to edit the frame or the cover page.

See also

Using cover pages and frames (Page 50)

Editing cover pages and frames (Page 53)

4.1.7 Designing cover pages and frames

4.1.7.1 Editing cover pages and frames

The documentation editor is a graphical editor which allows you to design frames and cover pages for your plant documentation. You can place images or text elements on the frame and the cover pages in the document editor. The text elements are either static or they are automatically filled during printing with the data from the document information that you have selected in the print dialog.

Procedure

To edit a cover page or a frame in the documentation editor, follow these steps:

1. In the project tree, double-click on the entry for an existing cover page or frame under the "Documentation information > Frames " or "Documentation information > Cover pages" group.
   The documentation editor opens.
2. Design the cover page or frame as desired.
3. Close the documentation editor.
   The changes to the cover page or frame are applied automatically.

See also

Creating a cover page (Page 51)

Creating frames (Page 51)

General operation of the documentation editor (Page 54)

4.1.7.2 General operation of the documentation editor

Components of the documentation editor

The following figure provides an overview of the components of the documentation editor:

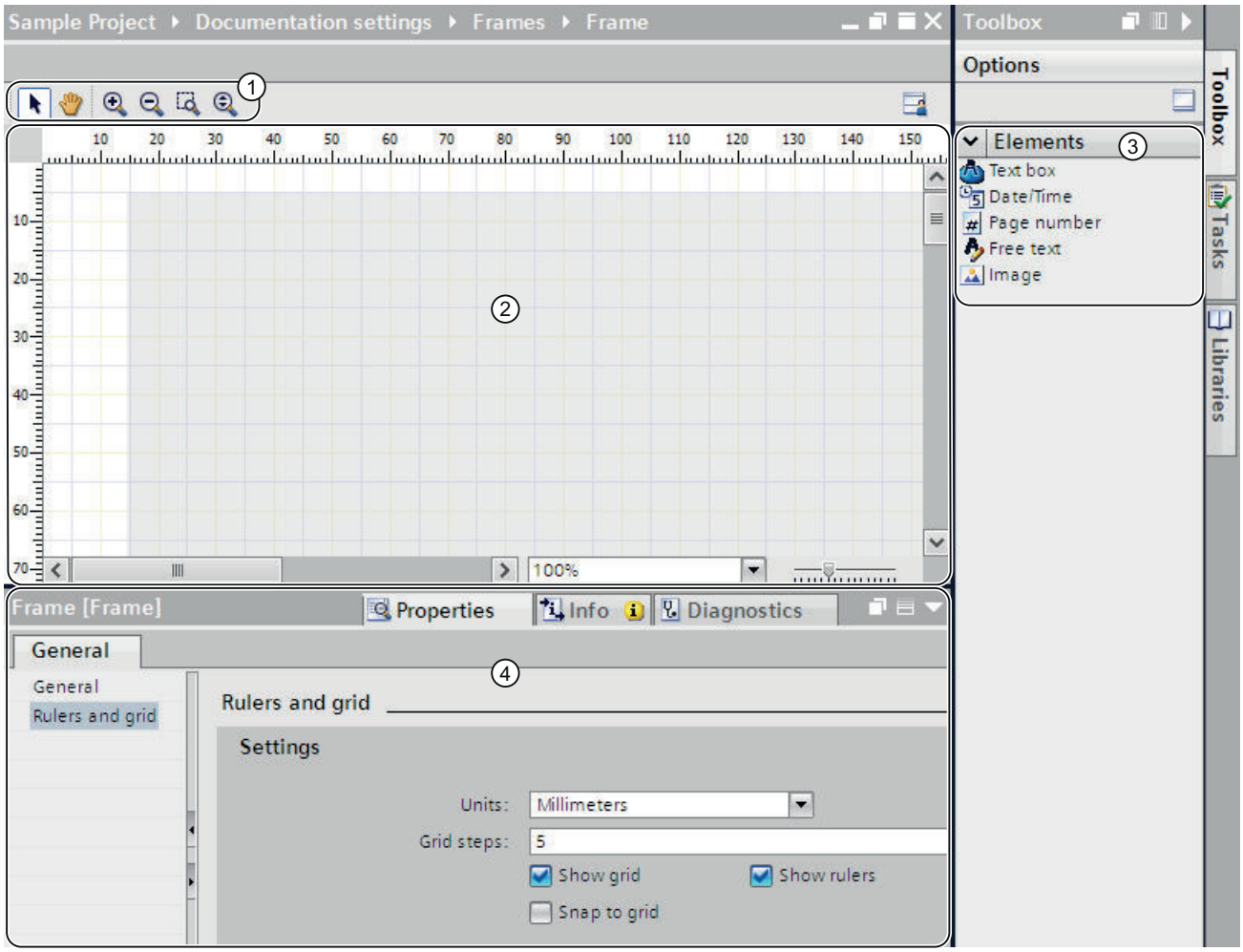

① Toolbar

The toolbar provides the following tools (from left to right):

- Arrow tool
  Enables object selection.
- Navigation tool
  Allows shifting of the partial page.
- Zoom-in button
  Magnifies the page display incrementally.
- Zoom-out button
  Reduces the page display incrementally.
- Zoom with selection
  Adapts the page size to a selected work area.
- Dynamic zoom
  Adapts the page width to the work area.

② Work area

You can design the cover page or frame in the work area.

③ "Toolbox" task card

The "Toolbox" task card contains various types of placeholders that you can use on the cover sheet or frame. The placeholders can be placed in the work place using a drag-and-drop operation.

④ Properties in the Inspector window

You can display and modify the properties of the currently selected object in the "Properties" tab of the Inspector window. For example, you can modify the properties of the page, format text, specify the position of objects on the page, etc.

Operation in the documentation editor

The following basic functions are available in the documentation editor:

- Drag-and-drop functionality
  The documentation editor is a graphic editor. which means you can place objects anywhere with the mouse. An image of the page is displayed in the work area. This image corresponds to the ultimate print layout.
  If you want to select objects on the page in order to move them or modify their properties, the arrow tool must be activated in the toolbar.
- Zoom function
  You can use the zoom function to change the size of the page display. You have two options for changing the page size:
  - Via the buttons in the toolbar
    Select the "Zoom in" or "Zoom out" magnifying glass button in the toolbar of the documentation editor. Then click on the page in order to magnify (zoom in) or reduce (zoom out) the page incrementally.
    To zoom in on a particular area, select the "Zoom with selection" tool and use the mouse to drag an outline around the area you want to focus on.
    To continuously zoom in or zoom out of the work area, use the "Dynamic zoom" tool. To magnify the page display, click anywhere on the work area, and then hold down the mouse button while dragging the mouse toward the top of the page. To reduce the page display, drag the mouse toward the bottom of the page.
  - Via the zoom bar
    You can also use the zoom bar (located in the bottom right corner of the work area) to change the display size. Select a percentage value from the drop-down list or enter a percentage value. Alternatively, you control the display size using the slider.
- Navigation over the page
  In addition to scrolling, you the option of changing the partial page with the navigation tool. To change the partial page with the navigation tool, select the Hand button in the toolbar. Then, click anywhere on the page and hold the mouse button down while moving the page to the desired position.

Using and adapting the positioning aids

You have various aids at your disposal to help you position elements on the page:

- Rulers
  Rulers are affixed to the page margins in the work area.
- Page grid
  A grid is placed underneath the page in the work area.

You can display, hide or adapt the positioning aids in the Inspector window under "Properties > Rulers and grid". You can make the following settings:

- Units:
  Specify the unit of measurement for the grid and the rulers.
- Grid steps:
  Specify the width of the grid.
- Show grid:
  Specify whether the grid is to be displayed or hidden.

- Snap to grid:
  Specify whether objects are to be aligned automatically to the grid. If the option is selected, the grid lines function like a "magnet".
- Show rulers:
  Specify whether the rulers are to be displayed.

See also

Editing cover pages and frames (Page 53)

Specifying the print area (Page 57)

Inserting placeholders for metadata (Page 57)

4.1.7.3 Specifying the print area

An area within the frame is provided for the actual printed contents. The project data is then always inserted inside this defined and uniformly consistent area within the frame. You can adjust the size of the print area.

Requirement

A frame is open in the documentation editor.

Procedure

To define an area for the printed contents, follow these steps:

1. Click on the slightly darker area within the page display in the documentation editor to select the area for the print content.
   This opens the properties of the area to be printed in the Inspector window.
2. Enter the position of the print area on the X and Y axes in the Inspector window.
3. Specify the width and height of the print area in cm in the Inspector window.

Alternatively, you can change the width and position of the print area in the graphic display of the page. To do so, use the mouse to drag the margins of the print area until the desired size and position are achieved.

See also

Creating frames (Page 51)

General operation of the documentation editor (Page 54)

4.1.7.4 Inserting placeholders for metadata

You can provide placeholders on the cover page and in a frame. The placeholders are automatically filled with metadata from documentation information during printing, if they are placeholders for text. Alternatively, you can add non-modifiable data, such as free text or an image.

All elements are arranged in numbered Z-Orders. If objects overlap, you can determine in which sequence these are to be arranged.

Types of placeholders

The following types of placeholders are available to you:

- Text field
  The text field stands as a placeholder for a text element from a document information. In the properties of the text field, you set which text from a document information should be automatically inserted during printing.
- Field for date and time
  A date and time are inserted instead of the placeholder when printing. This can be the date of creation or the point in time when the last change was made to the project. In the properties of the Inspector window, you specify which date or time is printed.
- Page number
  The correct page number is automatically applied when printing.
- Free text
  You can enter freely selectable text in the properties of the text field. The text is static and is not influenced by the document information selected at the time of printing.
- Image
  Select the image file in the properties of the placeholder in the Inspector window. Images in the formats BMP, JPEG, PNG, EMF or GIF are possible.

Requirement

An cover page or frame is open in the documentation editor.

Procedure

To insert placeholders for metadata on the cover sheet or in a frame, follow these steps:

1. Drag a field from the "Toolbox > Elements" task card to the work area of the documentation editor.
   The placeholder is inserted. The placeholder properties are shown in the Inspector window and can be edited there.
2. Select the metadata to be inserted during printing from the "Text" drop-down list in the Inspector window under "Properties > General > Text box". Alternatively, you have the option of entering free text or selecting an image depending on the type of placeholder.
3. In the Inspector window under "Properties > General > Position and size", specify the position of the placeholder on the X and Y axis and enter the width and height of the text box in cm. You specify the sequence of the objects in the "Z-Order" field, if these overlap. The smaller the value, the further down an object is located.
4. In the Inspector window, go to "Properties > View" and select the font formatting and the orientation of the text as well as the alignment of the text. You cannot make this setting for images.

See also

General operation of the documentation editor (Page 54)

4.1.8 Displaying print preview

4.1.8.1 Creating a print preview

Creating a print preview

You can create a preview of the printout. Document information can be chosen for this, in the same way as for the actual printout. In this way, you preview the selected frame and, if applicable, the cover sheet. The settings are retained for later printing.

Procedure

To create a print preview and to set the scope of the later printout, follow these steps:

1. Select the "Print preview" command in the "Project" menu.
   The "Print preview" dialog opens.
2. Select the frame layout you want to use for the printout.
   - In the "Document information" drop-down list, select the documentation information you want to use later for the printout.
   - Select the "Print cover page" check box to print the cover page, which is specified in the selected document information.
   - Select the "Print table of contents" check box to add a table of contents to the printout.

   The check boxes for printing the cover page and the table of contents can only be selected if you have started the printout in the project tree.
3. Under "Print objects/area", select what is to be printed. The selection is only possible if you have started the printout from an editor that supports this function.
   - Select "All" to print out the entire content of the editor.
   - Choose "Selection" to print only the objects currently selected in the editor.

4. Select the print scope under "Properties".
   - Select "All" to print all configuration data of the selected objects.
   - Select "Visible" to print the information of an editor that is currently visible on the screen. This option can only be chosen if you have started the printout from an editor that supports this function.
   - Select "Compact" to print out an abbreviated version of the project data.
5. Click "Preview" to generate the preview.
   A print preview is created in the work area.

Note

Wait time for extensive documents

It can take several minutes to generate the print preview in the case of very extensive projects. You can continue working normally in the meantime on systems with adequate resources. The progress of the print preview is shown in the status bar.

See also

Operation within the print preview (Page 61)

4.1.8.2 Operation within the print preview

Functions within the print preview

The print preview shows an exact image of the subsequent printout. You can use the buttons in the toolbar to modify the print preview display. The following functions are available (from left to right):

- Navigation mode
  Allows shifting of the partial page.
  To change the partial page with the navigation tool, select the arrow button in the toolbar. Then, click anywhere on the page and hold the mouse button down while moving the page to the desired position.
- Zoom function
  - "Zoom in" and "Zoom out"
    Magnifies or reduces the page display.
    To zoom in or zoom out the display incrementally, select the corresponding button. Then click on the page in order to magnify (zoom in) or reduce (zoom out) the page incrementally.
    To zoom in on a particular area, enable the "Zoom with selection" icon and use the mouse to drag an outline around the area you want to focus on.
    To zoom dynamically through the page, select the button "Zoom in / zoom out dynamically". With pressed mouse button, scroll down over the page to zoom in. Scroll up to zoom out.
  - Percentage value in the drop-down list
    Specifies the display size of the page in percent.
    Enter a percentage value or select a percentage value from the drop-down list.
    Alternatively, choose the "Fit to page" option from the drop-down list to adapt the page size to the work area. Or, choose "Fit to width" to adapt the page width to the work area.
- "Forward" and "Backward"
  Each change in the partial page, the page count, or the display size is saved in a history in the background. You can use the "Forward" or "Backward" button to return to the previous view or the next view.
- Page navigation
  - "First page"
    Jumps to the first page
  - "Previous page"
    Goes to the previous page.
  - "Page number" input field
    Shows the current page. To jump directly to a page, enter the page number of the page you want to view.
  - "Next page"
    Goes to the next page.
  - "Last page"
    Jumps to the last page.

See also

Creating a print preview (Page 59)

4.1.9 Printing project data

You have two options for printing out project data:

- Print immediately using default settings by means of the "Print" button in the toolbar.
  The button is only active if a printable object is selected.
- Printout with additional setting options with the "Project > Print" menu command.
  For example, you can choose a different printer or specific documentation information or you can specify whether a cover page and table of contents are to be printed. In addition, you can specify the print scope or display a print preview prior to printing.

Requirement

- At least one printer is configured.
- The objects to be printed are not protected.
  The print scope for protected objects is limited. Disable the know-how protection to print the objects in full.

Printing project data

To print out data from the current project or the entire project with additional setting options, follow these steps:

1. Select the entire project in the project tree in order to print out the entire project. To print only individual elements within a project, select them in the project tree.
2. Select the "Print" command in the "Project" menu.
   The "Print" dialog opens.
3. Select the printer in the "Name" box.
4. Click "Advanced" to modify the Windows printer settings.
5. Select the frame layout you want to use for the printout.
   - Select the documentation information in the "Document information" drop-down list.
     The frame stored in the document information is used for the printout. All placeholders within the chosen frame are filled with the metadata from the selected document information.
   - Select the "Print cover page" check box to print the cover page, which is stored in the selected document information.
   - Select the "Print table of contents" check box to add a table of contents to the printout.

   The check boxes for printing the cover page and the table of contents can only be selected if you have started the printout in the project tree.

6. Under "Print objects/area", select what is to be printed. The selection is only possible if you have started the printout from an editor that supports this function.
   - Select "All" to print out the entire content of the editor.
   - Select "Selection" to print only the objects currently selected in the editor.
7. Select the print scope under "Properties".
   - Select "All" to print all configuration data of the selected objects.
   - Select "Visible" to print the information of an editor that is currently visible on the screen. This option can only be chosen if you have started the printout from an editor.
   - Select "Compact" to print out an abbreviated version of the project data.
8. Click "Preview" to generate a print preview in advance.
   A print preview is created in the work area.
9. Click "Print" to start the printout.

Note

Scope of the "Print" dialog

The options available in the "Print" dialog vary depending on the elements to be printed.

Result

The project data is prepared in the background for printing and then printed on the selected printer. The status bar shows the progress of the print operation. You can continue working normally while data is being prepared for printing.

The print results and any errors or warnings are listed in the Inspector window under "Info" at the conclusion of the print job.

Canceling a print job

To cancel an active print job, follow these steps:

1. Click the blue cross in the status bar next to the progress display for the printout.
   The printout is aborted.

See also

Protection concept for project data (Page 41)

Revoking access rights for devices (Page 43)

Printout of project contents (Page 46)

Designing cover pages and frames (Page 53)

4.2 Printing module labels

4.2.1 Print function for module labels

Printing of module labeling strips for hardware modules

You can print labeling strips for the modules in your project with the help of the TIA Portal. The labeling strips are custom-fit to each module and can contain the following printed information:

- Symbolic name of the input or output
- Absolute address of the input or output
- Symbolic name and additionally the absolute address of the input or output. The order of the information can be specified.

The modules are displayed graphically in the device view. If you set the zoom level in the device view to at least 200%, the labels for the individual modules will be visible. The printout on the labeling strip corresponds to the representation of the labeling in the device view.

The following figure shows an example of two modules in the device view on which the labeling of the inputs and outputs is visible:

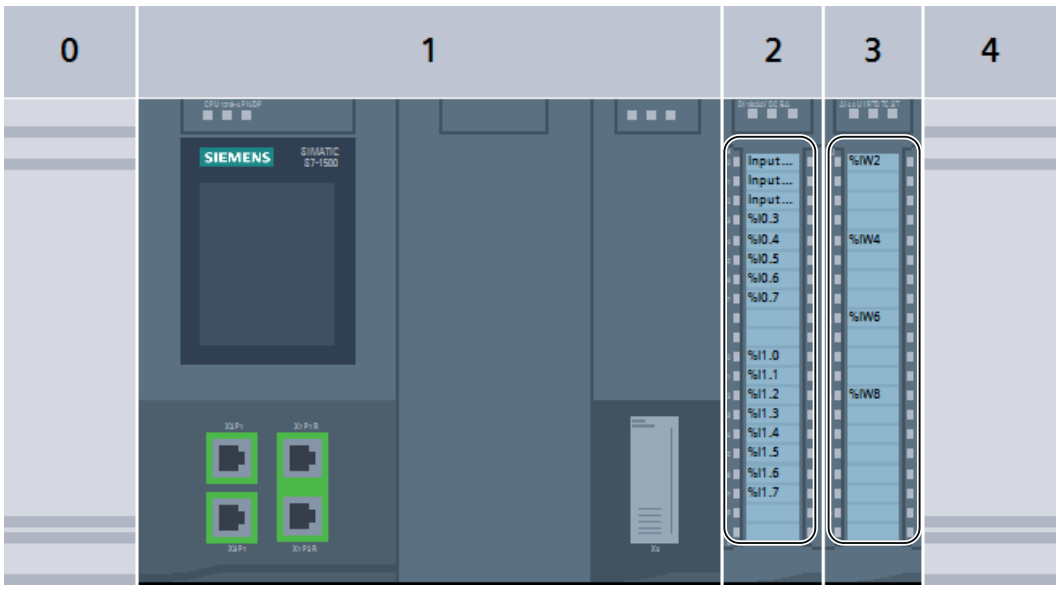

Export and further editing as Microsoft Word file

The labeling strips are first exported as a Microsoft Word file (.docx) before they are printed. The file can be further edited with commonly available word processing programs such as Microsoft Word. The individual labeling strips are represented as a table in the .docx file.

The character spacing of the text within the table is adapted by default, so that texts are not truncated. If you want to prevent this from stretching or compressing the text too much, change the character spacing of the text in the table cell properties.

Printing the labeling data to an XML file

As an alternative to printing the labeling strips, you can output the addresses of the inputs and outputs of a module to an XML file. You can use the export to an XML file for devices for which no ready-made labeling strips are available. You can also use the export to an XML file to create labeling strips with another program. The program must be able to transform the data of the XML file into an input format that is suitable for the labeling system. The schema of the XML file is available in the section "XML schema of the export file (Page 68)".

Print media

You can print the labeling strips either on ready-made print sheets or on standard DIN A4 paper. You can separate the individual labeling strips from the ready-made print sheets and insert them in the designated labeling areas of your modules. If you print on standard paper, the individual labeling strips must be cut out. Cut marks are automatically included on the printout and serve as aids.

Because the paper feeds of printers differ slightly, the printout may be slightly offset on the paper. When the labeling strips are printed on ready-made sheets, printing that is accurate to the millimeter is important. Otherwise, the text will not be fit exactly inside the stamped area. In addition, if the printing is imprecise, the labeling of an input or output may no longer be congruent with the channel status displays of the module. To ensure precise printing, you can enter an offset value for your printer in the TIA Portal. For information on how to determine the proper offset value for your printer, refer to Chapter "Determining the print area offset (Page 71)".

See also

Exporting labeling data as XML (Page 67)

XML schema of the export file (Page 68)

Printing labels (Page 65)

Determining the print area offset (Page 71)

Documentation settings (Page 45)

4.2.2 Printing labels

You can print labeling strips for the modules in your project if provision has been made for attaching labels to the utilized modules. The labels are first exported to a Microsoft Word file (.docx). A separate .docx file is created for each module family (for example, for all selected S7-1500 modules). The labels are always printed from the word processing program.

Requirement

The following requirements apply to printing of labeling strips:

- The chosen modules must support the printing of labels. Otherwise, the data can only be output to an XML file.
- A word processing program that supports Microsoft Word DOCX files must be installed, e.g., Microsoft Word 2010 or later.
- You need the ready-made labels for your modules or commercially available DIN A4 paper.

Procedure

To print labeling strips for hardware modules, follow these steps:

1. In the project tree, select the modules for which you want to print labeling strips.
   - You can select one or more stations in order to print out labeling strips for all modules plugged into these stations.
   - Alternatively, select the desired modules below the stations in the "Local modules" folder.
2. Right-click one of the devices, and select the "Export labeling strips" command from the shortcut menu.
   The "Export labeling strips" dialog opens.
3. In the "Content of the labeling strip" area, select the data to be printed on the labeling strip:
   - Select "Symbolic name" in order to print the symbolic name of the input or output (corresponds to the contents of the "Name" column in the IO tag table).
   - Select "Absolute address" in order to print the absolute address of the input or output (corresponds to the contents of the "Address" column in the IO tag table).
   - Select "Absolute and symbolic address" or "Symbolic and absolute address" in order to print both addresses. Printing takes place according to the specified order.
4. In the "Export format" area, define how the labeling data will be output.
   - Select "Print on SIEMENS labeling sheet" if you are printing on a ready-made label sheet for your modules.
   - Select "Print on plain DIN A4 page" if you are printing on standard DIN A4 paper.
5. Select correction values for your printer in the "Offset print area", if required. The correction values are used for correct alignment of the print area. Correction values are only necessary if you are printing on ready-made labeling strips.
   - Enter a correction value, in millimeters, in the "Vertical offset" field. A negative value shifts the print area upward. A positive value shifts the print area downward.
   - Enter a correction value, in millimeters, in the "Horizontal offset" field. A negative value shifts the print area to the left. A positive value shifts the print area to the right.
6. In the "Path" field, select the path where the exported files will be stored.
7. Click the "Export" button to start the export.
   The export files are created.
8. Open the DOCX files with a conventional word processing program, such as Microsoft Word, and change the design of the labeling strips if necessary.

9. Print out the labeling strips from your word processing program. Use the paper that you specified in the Export dialog for this.
10. If you are using ready-made sheets, separate the labeling strips at the stamped lines provided for that purpose. When standard DIN A4 paper is used, you must cut out the labeling strips.

See also

Determining the print area offset (Page 71)

Exporting labeling data as XML (Page 67)

4.2.3 Exporting labeling data as XML

The TIA Portal supports a large number of different modules and can be continually expanded using Hardware Support Packages, for example. Ready-made labeling strips are not available for every supported module. However, you can still use the TIA Portal to label inputs and outputs of modules that are not supported. First, you export the absolute and symbolic addresses of the inputs and outputs to a standardized XML file. Then, you import the XML file to an external program for printing the labels. In this program, you prepare the data to suit your modules and print out the labels.

Procedure

To export labeling data for hardware modules as XML, follow these steps:

1. In the project tree or network view, select the modules for which you need labeling strips.
   - You can select one or more stations in order to export the input and output addresses of all modules plugged into these stations.
   - Alternatively, select the desired modules below the stations in the "Local modules" folder.
2. Right-click one of the devices and select the "Export module labeling strips" command from the shortcut menu.
   The "Export labeling strips" dialog opens.
3. In the "Export format" area, select the "Export to XML file" option.
4. In the "Path" field, select the path where the XML file will be stored.
5. Click the "Export" button to start the export to an XML file.
   The XML file is created with the name "<Project name>_IO_Channels.xml".

See also

XML schema of the export file (Page 68)

Print function for module labels (Page 64)

Printing labels (Page 65)

4.2.4 XML schema of the export file

XML schema of an export file

The XML file for module labeling strips is structured according to the following schema:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://tempuri.org/XMLSchema.xsd"
elementFormDefault="qualified"  xmlns:mstns="http://tempuri.org/
XMLSchema.xsd"  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Stations">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Station" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
        <xs:element name="Rack" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
          <xs:element name="Module" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
            <xs:element name="IOChannel" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Address" type="xs:string">></
xs:element>
               <xs:element name ="Tag" type="xs:string"></xs:element>
              </xs:sequence>
            <xs:attribute name="Number" type="xs:int">></xs:attribute>
            </xs:complexType>
            </xs:element>
            </xs:sequence>
            <xs:attribute name="Name">></xs:attribute>
          </xs:complexType>
          </xs:element>
          </xs:sequence>
          <xs:attribute name="Name" type="xs:string">></xs:attribute>
        </xs:complexType>
        </xs:element>
        </xs:sequence>
        <xs:attribute name="Name" type="xs:string">></xs:attribute>
      </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  </xs:element>
</xs:schema>
```

Example of an XML file

The following example shows an XML file that contains the labeling data for an S7-1500 CPU with a digital input module and an analog input module:

```
<?xml version="1.0" encoding="UTF-8"?>
<Stations>
  <!-- The CPU is specified first -->
  <Station Name="S71500/ET200MP-Station_1">
    <Rack Name="Rack_0"> <!-- Name of the rack -->
      <Module Name="Sample S7-1500" /> <!-- Name of the CPU -->
     <Module Name="DI 16x24VDC BA_1"> <!-- Name of the digital input
module -->
        <!-- The individual channels of the digital input module are
specified -->
        <IOChannel Number="0">
          <Address>%I0.0</Address>
          <Tag>Input Value 1</Tag> <!-- Symbolic address of input 0
-->
        </IOChannel>
        <IOChannel Number="1">
          <Address>%I0.1</Address>
          <Tag>Input Value 2</Tag>
        </IOChannel>
        <IOChannel Number="2">
          <Address>%I0.2</Address>
          <Tag>Input Value 3</Tag>
        </IOChannel>
        <!-- All other channels follow -->
      </Module>
      <Module Name="AI 4xU/I/RTD/TC ST_1"> <!-- Name of the analog
input module -->
        <!-- The individual channels of the analog input module are
specified -->
        <IOChannel Number="0">
          <Address>%IW2</Address>
          <Tag> <!-- Symbolic addresses are not specified for the
analog input module. -->
          </Tag>
        </IOChannel>
        <IOChannel Number="1">
          <Address>%IW4</Address>
          <Tag>
          </Tag>
        </IOChannel>
        <IOChannel Number="2">
          <Address>%IW6</Address>
          <Tag>
          </Tag>
        </IOChannel>
        <IOChannel Number="3">
          <Address>%IW8</Address>
          <Tag>
          </Tag>
```

```
          </IOChannel>
        </Module>
        <Module Name="Sample S7-1500" />
        <Module Name="DI 16x24VDC BA_1">
          <IOChannel Number="0">
            <Address>%I0.0</Address>
            <Tag>Input Value 1</Tag>
          </IOChannel>
          <IOChannel Number="1">
            <Address>%I0.1</Address>
            <Tag>Input Value 2</Tag>
          </IOChannel>
          <IOChannel Number="2">
            <Address>%I0.2</Address>
            <Tag>Input Value 3</Tag>
          </IOChannel>
          <!-- All other channels follow -->
        </Module>
        <Module Name="AI 4xU/I/RTD/TC ST_1">
          <IOChannel Number="0">
            <Address>%IW2</Address>
            <Tag>
            </Tag>
          </IOChannel>
          <IOChannel Number="1">
            <Address>%IW4</Address>
            <Tag>
            </Tag>
          </IOChannel>
          <IOChannel Number="2">
            <Address>%IW6</Address>
            <Tag>
            </Tag>
          </IOChannel>
          <IOChannel Number="3">
            <Address>%IW8</Address>
            <Tag>
            </Tag>
          </IOChannel>
        </Module>
      </Rack>
    </Station>
  </Stations>
```

See also

Exporting labeling data as XML (Page 67)

4.2.5 Determining the print area offset

If you are using a ready-made label sheet, the printing on it must be applied precisely so that the text is properly oriented on the prestamped labels and will have the proper fit relative to the channel status displays of the module. However, the paper feeds vary slightly from one printer to another. For this reason, you must enter a suitable correction value for your printer in the TIA Portal, if necessary. The print area is then shifted in the exported .docx file in such a way that the printing fits precisely on the ready-made label sheet.

The settings for shifting the print area are stored for the specific Windows user. If you log on to Windows using a different user name, you have to enter the correction values again.

The procedure for determining the correction value for your printer is described below.

Requirement

- You require a ready-made label sheet.
- You must have access to the actual printer that you will use subsequently for the printout. The printer must be made ready for printing on standard DIN A4 paper.

Procedure

To determine the correction value for your printer, follow these steps:

1. Print out a label sheet on standard DIN A4 paper, as described in Chapter "Printing labels (Page 65)".
2. Compare the printout on the DIN A4 paper with the ready-made label sheet.
3. If the print area is offset, you must use correction values.
   - Using a ruler, measure the horizontal offset relative to the ready-made label sheet. This will be entered later in the "Horizontal offset" field of the Export dialog box for the printing. If the print area is offset to the right, a negative correction value must be entered. If the print area is offset to the left , a positive correction value must be entered.
   - Using a ruler, measure the vertical offset relative to the ready-made label sheet. This will be entered later in the "Vertical offset" field of the Export dialog box for the printing. If the print area is offset downward, a negative correction value must be entered. If the print area is offset upward, a positive correction value must be entered.

Working with multi-language projects 5

5.1 Project text basics

Texts in different languages in the project

When you enter texts while working on a project, you would normally do this in your own language. If you then pass on the project to someone else who does not know this language, this person will require a translation of the relevant texts to a language they know. This is why all texts can be translated. In this way, you can ensure that anyone who is subsequently confronted with the texts sees the texts in his/her language of choice.

Project language

Project languages are all languages in which a project will later be used. Based on the editing language, all the texts can be translated to the various project languages. You specify the languages that will be available in the project tree under "Languages & Resources > Project languages".

Editing language

Every project has an editing language. When you enter texts, these are always created in the editing language. You should therefore make sure that the editing language set is the language in which you enter the texts. This avoids problems if you translate the texts later.

The editing language does not depend on the language of the user interface. You could, for example, set English as the user interface language, but use Italian as the editing language. If you enter texts, these will be created in the project language "Italian" in this case, although the user interface of the TIA Portal is displayed in English.

You set the editing language in the project tree under "Languages & Resources > Project languages > Editing language".

Reference language

The reference language is used as a template for translation. The text is displayed in the reference language for each text box in the "Tasks > Languages and resources" task card. You therefore know which text that belongs in a text box, even when no text is entered in the currently selected editing language.

User texts and system texts

For clarification purposes, a distinction is made between user texts and system texts:

- User texts are texts that the user created.
- System texts are texts that are created automatically according to the configuration in the project.

You manage the project texts in the project tree under "Languages & Resources > Project texts".

Examples of multilingual project texts

You can, for example, manage the following project texts in more than one language:

- Block titles and block comments
- Network titles and network comments
- Comments in tables
- Alarm texts
- Operator-relevant texts
- Text lists
- Labels of buttons
- Display names of recipes

Translating texts

The following procedures are available to translate texts.

- Translate all texts used in the project in tabular form
  You can enter the translations for the individual project languages directly in the "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts".
- Specify text assigned to individual objects in the Inspector window
  In the Inspector window, you can translate the texts that are assigned to the currently selected objects. Columns are displayed in a table for all available project languages. You can enter the translations for each text in the columns.
- Translating texts using reference texts
  You can change the editing language for shorter texts. All the text cells are filled again with the default values and can be filled in the current language. As orientation, you can display what you last entered in the box in the reference language. To do this, select the "Tasks" task card and open the "Languages & resources".
- Exporting texts and translating them externally
  With larger volumes of text, you can export the texts to an Office Open XML file and translate them in a conventional table calculation program. You then import the translated list again into the TIA Portal.

Note

Using East Asian project languages

You need Microsoft Windows at least in the Professional version or higher to display East Asian project languages. Microsoft Windows in the Professional version must be installed in the local language. With the "Ultimate" or "Enterprise" versions, it is sufficient to install the appropriate language pack.

See also

Application examples for multilanguage projects (Page 86)

5.2 Categories of user texts

Overview of text categories

Below you will find an overview of the categories of user texts that you can select for export.

Text categories	Usage
Formatted alarm text	
HMI scree	Text that appears in an HMI screen
HMI comments	Text that appears in a comment in Runtime
HMI Runtime	
Category for text lists	
Multilingual text category	
Alarm class text	
Alarm text	Text that appears in a message display
SiVArc configuration	
Other text categories	

5.3 Select project languages

All the texts can be displayed within a project in the same language that you selected for your software user interface. This means that all project texts must exist in the corresponding language. You can select the available project languages yourself.

Procedure

To select the project languages, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree.
   The elements below this are displayed.
2. Double-click on "Project languages".
   In the work area, you will see a list of languages that you can select.
3. Select the required languages.

Result

All texts can be displayed in the activated languages if there is already a translation for these languages.

5.4 Setting the editing language

All the texts in the project are created in the editing language when they are entered. If you change the editing language, all future text input will be stored in the new editing language.

Procedure

To change the editing language, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree.
   The lower-level elements are displayed.
2. Double-click on "Project languages".
   The possible settings for the project languages are displayed in the work area.
3. Select the editing language in "General > Editing language".

5.5 Translating all project texts in tabular form

You can display and edit all project text used in the currently open project in a list. User and system texts are separated into two different lists for clarity. Both lists contain a column for each project language. Enter the translations of the texts in the respective columns.

Requirement

You have selected at least one further project language.

Procedure

To translate text in the project-wide list, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree.
   The elements below this are displayed.
2. Double-click "Project texts".
   A list with the user texts in the project is displayed in the work area.
3. Click on "System texts" if you you want to edit the list of system texts rather than the user texts.
4. You can improve the clarity of the lists if you have a lot of texts.
   - To group identical texts and to translate them all at once, click the "Switch on/off grouping" button in the toolbar.
   - To hide texts that do not have a translation, click the "Filter for empty texts on/off" button in the toolbar.
   - To further limit the displayed project texts to certain devices, select the devices for which you want to display project texts in the drop-down list.
5. Enter the translation of the project texts in the relevant column.

5.6 Translating project texts

If you want to edit the text of individual objects, it would be too difficult to locate the matching text in the table with all project texts. For this reason, there is a table in the Inspector window in which only the texts assigned to the currently selected objects are displayed. In the table, you can add missing translations for individual project languages or change existing texts.

Requirement

Enter a text in at least one project language for the texts to be translated.

Procedure

To edit the text of the currently selected object, follow these steps:

1. Select the object whose text you want to edit.
2. Open the "Properties" tab in the Inspector window.
3. Open the lower-level "Texts" tab in the inspector window.
   A table with all the texts that belong to the selected objects is displayed. The table contains one column for the currently selected editing language and the reference language, as well as additional columns for the other project languages.
4. Add or change the entries in the table for each project language.

See also

Application examples for multilanguage projects (Page 86)

5.7 Translating project texts using reference texts

Introduction

After changing the editing language, all texts are shown in input boxes in the new editing language. If there is not yet a translation available for the newly set language, the input boxes are empty or filled with default values.

If you enter text in an input box, this is saved in the current editing language. Following this, the texts exist in two project languages for this input field, in the previous editing language and in the current editing language. This makes it possible to create texts in several project languages.

You can display existing translations for an input box in other project languages. These serve as a comparison for text input in the current editing language and they are known as the reference language.

Note

The display of reference texts depends on the installed products and is not supported by every editor.

Requirement

There is at least one translation into a different project language for an input field.

Procedure

To display the translation of an input cell in a reference language, follow these steps:

1. In the "Tasks" task card, select the "Languages & Resources" pane.
2. Select a reference language from the "Reference language" drop-down list.

Result

The reference language is preset. If you click in a text box, translations that already exist in other project languages are shown in the "Tasks > Languages & Resources" task card.

5.8 Exporting project texts

You can export project texts for translation and then reimport them. The texts are exported to an Office Open XML file with the extension ".xlsx". This can be edited in Microsoft Excel or a number of other spreadsheet programs.

The following export options are available:

- Exporting individual project texts
  You can select individual texts in the project texts editor and then export the selected texts.
- Exporting project texts of a device
  When you have selected a device, the "Properties > Texts" tab of the Inspector window includes all texts that are part of the respective device. Here you can export all texts that are part of the respective device.
- Exporting all user texts or system texts at once
  You can either export all texts in the project or further limit the export by categories.

Note

Row limit in Microsoft Excel

Note that spreadsheet programs may be able to process only a certain number of rows. Microsoft Excel 2003 supports a maximum of 65536 rows, for example. Later versions of Microsoft Excel support significantly more rows.

Exporting individual project texts

To export individual project texts, follow these steps:

1. Open the "Languages & Resources" folder in the project tree.
   The lower-level elements are displayed.
2. Double-click "Project texts".
   The project texts editor opens.
3. Choose the "User texts" or "System texts" tab in the editor, depending on which texts you want to export.
4. Select the project texts you want to export.
5. Click "Export project texts" on the editor toolbar.
   The "Export" dialog box opens.
6. In the "Source language" drop-down list, select the reference language from which you want to translate.
7. In the "Target language(s)", select the editing language into which you want to translate the texts. The dialog contains the project languages you specified previously.
   If the required language is missing, you must first specify it in the project languages editor.
   If you want to export all editing languages, select the "Select all" option.
8. Specify a file path and a file name for the export file in the "Select file for export" input box.
9. Click "Export".

Exporting project texts of a device

To export project texts that are part of a specific device, follow these steps:

1. Select the device and open the device properties in the Inspector window.
2. Open the "Texts" tab in the Inspector window.
3. From the drop-down list in the function list select the texts to be exported:
   - All texts
   - ProDiag / GRAPH: Alarm texts
   - ProDiag / GRAPH: Texts for HMI display
   - ProDiag / GRAPH: All texts
4. Click the "Export project texts" icon in the toolbar.
   The "Export" dialog box opens.
5. In the "Source language" drop-down list, select the reference language from which you want to translate.
6. In the "Target language(s)", select the editing language into which you want to translate the texts. The dialog contains the project languages you specified previously.
   If the required language is missing, you must first specify it in the project languages editor.
   If you want to export all editing languages, select the "Select all" option.
7. Specify a file path and a file name for the export file in the "Select file for export" input box.
8. Click "Export".

Exporting all system or user texts

To export all project texts, follow these steps:

1. Select the "Export project texts" command in the "Tools" menu.
   The "Export" dialog box opens.
2. In the "Source language" drop-down list, select the reference language from which you want to translate.
3. In the "Target language(s)", select the editing language into which you want to translate the texts. The dialog contains the project languages you specified previously.
   If the required language is missing, you must first specify it in the project languages editor.
   If you want to export all editing languages, select the "Select all" option.
4. In "Select content", select the check box "User texts" to export user texts. To export system texts, select "System texts". To export both user texts and system texts, select both check boxes.
5. In "Select content", select the required text categories for the user texts or the system texts.
6. In the "Export file" input field, specify a file name for the export file.
7. In the "Path" input field, select a path in the data system to which the export file is to be saved.
8. Click "Export".

See also

Application examples for multilanguage projects (Page 86)
Importing project texts (Page 85)

5.9 Importing project texts

After external compilation in a table calculation program, you import the project texts into the TIA Portal. You can import the project texts at the following locations:

- The "Tools" menu
- In the toolbar of the project texts editor
- In the properties of a device
  When you have selected a device, the "Properties > Texts" tab of the Inspector window includes all texts that are part of the respective device. You can also import the texts of the device at this location.

Importing project texts

To import a file containing project texts, follow these steps:

1. Select the "Import project texts" command in the "Tools" menu.
   Alternatives:
   – Click the "Import project texts" icon in the toolbar of the project texts editor.
   – Select a device and open its properties in the Inspector window. Open the "Texts" tab and click the "Import project texts" icon in the toolbar.
2. The "Import" dialog box opens.
3. Select the path and the file name of the import file from the "Select file for import" field.
4. Select the "Import source language" check box if you have made changes to the source language in the export file and you want to overwrite the entries in the project with the changes.
5. Click "Import".

See also

Exporting project texts (Page 82)

5.10 Application examples for multilanguage projects

Let us assume you are working in a team with colleagues some of whom speak English, some French and some German. You have created a project with the TIA Portal and have already created a functioning configuration.

To allow your other colleagues to be able to keep track of the project, you would like all devices being used to have comments in English and German. First, you would like to enter the comments in German. Following this, to save time and costs, you want to have the texts translated into English in a spreadsheet program by an external translation office.

In addition to this, you also want a single comment for a particular device in French so that your French-speaking colleague can continue working on this device.

The section below describes an example of how you can achieve this with the tools of the TIA Portal.

Translating the project into English

To enter the comments in German and to have them translated into English later, follow these steps:

1. Set the editing language to "German" and fill all the comment boxes with the relevant texts in German.
   On the device selected from the French-speaking colleague, enter, for example "Unser neues Gerät" in German first.
   All the comments are now stored in German.
2. Export all user texts to an Office Open XML file with the extension ".xlsx".
3. Have the user texts contained in the file translated into English in a spreadsheet program such as Microsoft Excel.
4. Import the file into the TIA Portal after it has been translated.
   All texts are now available in German and English.

Translating a single comment field to French

To translate an individual comment field to French, follow these steps:

1. Open the comment box for the device on which the French-speaking colleague will be working.
2. Open the "Languages & resources" pane in the "Tasks" task card.
3. Set "French" as the editing language in the "Languages & Resources" pane. As the reference language, set, for example, "English".
   Since no translation has yet been installed in French, the comment box is empty. In the "Languages & Resources" pane, the English translation "Our new device" is displayed as a reference.
4. Orientating yourself on the English reference text enter "Notre nouvel appareil" in the comment box.
   The comment for this device is now available in the languages German, English and French.

See also

Project text basics (Page 73)
Exporting project texts (Page 82)
Translating project texts (Page 80)

Working with text lists 6

6.1 Text lists

Introduction

You can centrally manage texts to be referenced in alarms. All the texts are stored in text lists. Each text list has a unique name with which you can call up its content. A range of values is assigned to each text in a text list. If a value from a range of values occurs, the corresponding text is called up.

All the texts can be translated to all project languages. Here, you have two options available:

- You can enter the translation of the texts in a list. You will find the list in the project tree under "Languages & Resources > Project texts".
- You can export all texts to a file in Office Open XML format and enter the translation in a spreadsheet program. The translations can then be imported again. Only export the data to areas that are protected with appropriate access mechanisms. Only import files that originate from trusted sources.

Each device in a project can have its own text lists. Device-specific text lists relate to only one device in the project and are therefore only valid for this device. For this reason, they are arranged under a device in the project tree. Device-specific text lists can be user-defined or created by the system.

Text lists editor

The text lists editor is divided into a bottom and top area. The top area displays the individual text lists. As soon as you select a text list, the included texts and the associated value ranges are displayed in the lower area. You can sort the table columns in the text lists editor in ascending and descending order by clicking the table header of the respective column.

User-defined and system-defined text lists

There are two types of text lists:

- User-defined text lists
  You can create user-defined text lists yourself and fill them with texts; in other words, you can specify value ranges and the corresponding texts yourself. With user-defined text lists, the name of the text list begins with "USER" as default. You can change this name to any suitable name.
- System-defined text lists
  System-defined text lists are created by the system. These always involve texts relating to devices. They are automatically created as soon as you insert a device in the project. With system alarms, the name of the text list begins with "SYSTEM". The name of the text list and the ranges of values it contains cannot be modified. You can only edit texts assigned to individual value ranges.

User-defined text lists	System-defined text lists
You can create new text lists and delete existing text lists.	You cannot create new text lists or delete text lists.
You can add and delete value ranges in the text lists.	You cannot add or delete value ranges in the text lists.
You can specify both the value ranges as well as the associated texts.	You can only edit the text associated with one value range.

See also

Exporting project texts (Page 82)

6.2 Creating user-defined text lists

You can create user-defined text lists for individual devices.

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Procedure

To create user-defined text lists, follow these steps:

1. Click on the arrow to the left of a device in the project tree.
   The elements arranged below the device are displayed.
2. Double-click on "PLC message text lists".
   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Double-click on the first free row in the table.
   A new user-defined text list is created.
4. Enter a name for your new text list in the "Name" column.
5. Select if you want to specify the value ranges in decimal, binary or in bits from the drop-down list in the "Selection" column. Depending on the device, there may be further options available at this point.
6. Enter a comment in the "Comment" column.
   A new user-defined text list is created and you can now enter the value ranges and texts.

6.3 Editing user-defined text lists

You can enter value ranges and the corresponding texts in user-defined text lists. User-defined text lists are always located below a device in the project tree.

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Procedure

To add to user-defined text lists with value ranges and texts, follow these steps:

1. Click on the arrow to the left of a device in the project tree.
   The elements arranged below are displayed.
2. Double-click on "PLC message text lists".
   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Select a text list in the table.
   The contents of the selected text list are displayed in the work area. There, you can enter a value range and assign texts to the individual value ranges.
4. Enter the value ranges you require in the "Range from" and " Range to" columns. The entry must be made in the numeric format selected for the text list.
5. Enter a text for each value range in the "Entry" column.

6.4 Editing system-defined text lists

In system-defined text lists, you can modify the individual texts assigned to a value range.

System-defined text lists are always located below a device in the project tree.

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Procedure

To edit texts in system-defined text lists that are assigned to a value range, follow these steps:

1. Click on the arrow to the left of a device in the project tree.
   The elements arranged below are displayed.
2. Double-click on "PLC message text lists".
   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Select a text list in the table.
   The contents of the selected text list are displayed in the work area. Here, you can add to or edit the texts assigned to a value range.
4. Enter a text for each value range in the "Entry" column.

6.5 Exporting text lists

You can export alarm texts for translation and then reimport them. The texts are exported to an Office Open XML file with the extension ".xlsx". This can be edited in Microsoft Excel or a number of other spreadsheet programs.

Requirements

- A project is open.
- The project includes a least one device.
- Text lists have been created in the project.
- The text list editor is open.

Procedure

To export text lists follow these steps:

1. Click the arrow to the left of a device in the project tree.
   The elements arranged below the device are displayed.
2. Double-click "PLC alarm text lists".
   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Select a text list in the table.
   The entries of the selected text list are displayed in the work area.
4. In the work area, select the text list that you want to export.
5. Click the "Export text lists" icon in the toolbar of the text list editor.
   The "Export text lists" dialog opens.
6. Specify a file path and a file name for the export file in the "Select file for export" input box.
7. Select the project languages that you want to import.
8. Click "Export".
   A message about the status of the export opens.

Result

The texts lists are exported to an Office Open XML file with the extension ".xlsx". The file is stored in the specified folder.

6.6 Importing text lists

After external compilation in a spreadsheet program you import the alarm texts into the TIA Portal.

Requirements

- A project is open.
- The project includes a least one device.
- Text lists have been created in the project.
- The text list editor is open.

Procedure

To import text lists, follow these steps:

1. Click the arrow to the left of a device in the project tree.
   The elements arranged below the device are displayed.
2. Double-click "PLC alarm text lists".
   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Click the "Import text lists" icon in the toolbar of the text list editor.
   The "Import text lists" dialog opens.
4. Select the path and file name of the import file from the "Import file" field.
5. Click "Import".
   A message on the status of the import opens.

Result

The text lists are imported. The existing data in the text lists in the project are overwritten in the process and new data are added. Text lists are only imported in the languages that have been activated in the project.

Using memory cards 7

7.1 Basics about memory cards

Introduction

Memory cards are plug-in cards that come in a variety of types and can be used for a variety of purposes. Depending on the device type or device family, memory cards can be used for purposes, such as:

- Load memory of a CPU
- Storage medium for projects, firmware backups, or any other files
- Storage medium for performing a firmware update
- Storage medium for the PROFINET device name

For information regarding the technical variants of the respective memory cards and general information on their handling, refer to the respective documentation for the device. For information on handling memory cards in the TIA Portal, refer to the online help under the keyword "Memory Card".

NOTICE
Memory card is unusable for SIMATIC devices If you use a SIMATIC memory card for non-SIMATIC purposes or you format it incorrectly, you will overwrite the internal structure of the SIMATIC memory card. The structure is not recoverable and the SIMATIC memory card becomes unusable for SIMATIC devices. Do not use memory cards for non-SIMATIC-related purposes, and do not format SIMATIC memory cards with third-party devices or Windows tools.

See also

Adding a user-defined card reader (Page 98)

Accessing memory cards (Page 99)

Displaying properties of memory cards (Page 100)

7.2 Adding a user-defined card reader

Introduction

If your card reader is not detected automatically, you can add it manually.

Requirement

The project view is open.

Procedure

To add a card reader, follow these steps:

1. Open the project tree.
2. Select the "Card Reader / USB memory > Add user-defined Card Reader" command in the "Project" menu.
   The "Add user-defined Card Reader" dialog opens.
3. In the drop-down list box, select the path for the card reader.
4. Confirm your entries with "OK".

See also

Basics about memory cards (Page 97)

Accessing memory cards (Page 99)

Displaying properties of memory cards (Page 100)

7.3 Accessing memory cards

Requirement

- A memory card is inserted in the card reader.
- The project view is open.

Note

You cannot work with multiple memory cards at one time. Only insert one memory card into the card reader.

Procedure

To access memory cards, follow these steps:

1. Open the project tree.
2. Select the "Card Reader / USB memory > Card Reader / Show USB memory" command in the "Project" menu.
   The "Card reader / USB memory" folder is displayed in the project tree.
3. Open the "Card Reader / USB memory" folder.
   You can now access the memory card.

   Note

   If data from a non-installed product is stored on the memory card, the folders that contain these data are shown in gray. You receive an error message when you attempt to access such a folder. Install the corresponding product if needed.

See also

Basics about memory cards (Page 97)

Adding a user-defined card reader (Page 98)

Displaying properties of memory cards (Page 100)

7.4 Displaying properties of memory cards

You can display the properties for the utilized memory cards. Note that different memory cards with different properties must be used, depending on the device.

Requirement

- A memory card is inserted in the card reader.
- The project view is open.

Procedure

To display the properties of a memory card, follow these steps:

1. Right-click on the memory card for which you want to display the properties.
2. Select the "Properties" command in the shortcut menu.
   The "Memory Card <name of the memory card>" dialog opens. The properties are displayed in this dialog.

See also

Basics about memory cards (Page 97)

Adding a user-defined card reader (Page 98)

Accessing memory cards (Page 99)

7.5 Show data on the memory card of a controller

In addition to the configuration, data such as firmware updates of the controller, HMI images or HMI backups can be stored in the memory card in a controller. With an online connection to the controller, this data is displayed in the project tree in the "Online card data" folder under the controller.

Requirements

- A memory card is inserted in the card reader of the controller.
- The memory card contains data that can be displayed.
- The project view is open.
- There is an online connection to the controller.

Procedure

To show data on the memory card of a controller, follow these steps:

1. Open the device folder in the project tree.
2. Open the "Online card data" folder.
   The data is shown in subfolders.

Using cross-references 8

8.1 Using cross-references

Introduction to cross-references

The cross-reference list provides an overview of the use of objects within the project. You can see which objects are interdependent and where the individual objects are located. Cross-references are therefore part of the project documentation.

You can also jump directly to the point of use of an object.

Which objects you can display and localize in the cross-reference list depends on the installed products.

Providing user-defined documentation 9

9.1 Using user-defined documentation

User-defined documentation for project or library contents

Over time, you create your own contents in a project or a library. Your own contents include, for example, blocks, tags or library types. While the functionality of the TIA Portal is described in the supplied help system, there is no help for the contents you have created yourself. You can create your own user-defined documentation to explain to other employees how your project works or how to use individual library types.

You can provide user-defined documentation in the available user interface languages. The default user interface languages in the TIA Portal are English, German, French, Spanish, Italian, and Chinese. You need to observe a few conventions when you create user-defined documentation so that the help matching an object can be opened.

You create the user-defined documentation either in one of the supported Office formats or as compiled HTML help in CHM format.

Possible areas for user-defined documentation

You can offer user-defined documentation in the following areas of the TIA Portal, for example:

- Project tree
- "Libraries" task card and library view
- Some editors, depending on the products installed, for example:
  - Program editor
    The programming languages LAD, FBD, STL, SCL and GRAPH are supported. Block calls also support the connection of user-defined documentation.
  - "Screens" editor
    Uses of screens and instances of HMI faceplates are supported in the "Screens" editor.

Directories for user-defined documentation

Save the user-defined documentation in one of the following directories:

- Project folder
  If you create user-defined documentation for objects within a project, save this help in the project folder. The user-defined documentation is also included when you pass on the project.
- Directory of a global library
  If you create user-defined documentation for objects within a global library, save the user-defined documentation in the directory of the global library. The user-defined documentation is also included when you pass on the global library.
- Central directory on the hard drive or a network drive
  You can store the user-defined documentation in a central directory on the hard disk or on a network drive. In this way, you have access to the user-defined documentation in each project or you use the documentation on a network drive together in the team. You specify the central file directory for the user-defined documentation using an XML file or in the settings of the TIA Portal.

Homepage for the user-defined documentation

You can create a separate homepage for each language version of the user-defined documentation. The homepage for the user-defined documentation can contain general help for a project or for a library. The homepage must be saved in the central storage directory for user-defined documentation.

Calling the user-defined documentation

If user-defined documentation is available for an object, you call it with the keyboard shortcut <Shift+F1>. The user-defined documentation is always opened with the standard program specified for the respective file format in Microsoft Windows.

Once you have pressed <Shift+F1>, certain directories are searched in a fixed order for user-defined documentation. The search order is given below:

1. Search in the central directory for user-defined documentation
   1.1 Search for a CHM file
   1.2 Search for documentation in other file formats
2. Search in the project or library directory
   2.1 Search for a CHM file
   2.2 Search for documentation in other file formats

The search is initially performed in the language directory for the currently set user interface language of the TIA Portal. If no help is contained in this language directory, the search for user-defined documentation is performed in the same order in the English language directory.

As soon as user-defined documentation is found in one location, the user-defined documentation is opened and the search is canceled. If no user-defined documentation is found in any of the directories, a search for a homepage for user-defined documentation is performed in the order presented above. The search for a homepage is again initially performed

in the language directory for the currently set user interface language. If no homepage is found there, the English language directory is searched.

Call log

You can display a call log for the user-defined documentation for easier connection of the user-defined documentation. The alarms within the log indicate the directories in which documentation is searched for and whether the call of the user-defined documentation is successful. In addition, the file name that is expected for the file is indicated. This allows you to identify how you must name your documentation and the directories in which you must save the user-defined documentation. The call log has the same sequence as the one used to search for user-defined documentation or a homepage.

The log is displayed in the Inspector window in the "Info" tab. Before you can display the call log, you must first enable the call log in the settings of the TIA Portal. To do this, select the "Display call log for user-defined documentation" check box under "Settings > General > User documentation". Alternatively, activate the display of the call log using an XML file.

See also

Conventions for the creation (Page 113)

Specifying settings with an XML file (Page 109)

Creating a homepage (Page 112)

Creating user-defined documentation (Page 118)

Calling user-defined documentation (Page 116)

9.2 Specifying settings in the TIA Portal

You specify the following settings for user-defined documentation in the settings of the TIA Portal:

- Displaying the call log in the Inspector window
  A log of the call of user-defined documentation is displayed on the 'Info > General' tab of the Inspector window. The log helps you to adhere to the conventions for calling user-defined documentation.
- Search for user-defined documentation in a central file directory
  You can save user-defined documentation in a directory outside the current project directory in order, for example, to make documentation available across projects.
- Central directory for user-defined documentation
  You store cross-project documentation in the central file directory for user-defined documentation.

Note

XML configuration file takes precedence over the settings of the TIA Portal

If you use an XML configuration file and have specified settings for user-defined documentation there, the settings in the XML file take precedence. As soon as you refresh the XML configuration file or restart the TIA Portal, the settings from the XML file are applied. The settings that you have made in the TIA Portal lose their validity.

Procedure

To specify a central storage location for user help, follow these steps:

1. Select the "Settings" command in the "Options" menu.
2. Open the "General > General" area.
3. Navigate to the "User documentation" section.
4. Select the "Display call log for user-defined documentation" check box in order to display a log of the call of the user-defined documentation in the Inspector window.
5. Select the "Search for user-defined documentation in a central directory" check box in order to store user-defined documentation in a cross-project directory.
6. Specify the path to where you save the cross-project documentation in the "Central directory for user-defined documentation" field.

See also

Specifying settings with an XML file (Page 109)

9.3 Specifying settings with an XML file

As an alternative to settings in the TIA Portal, you can make settings for the user-defined documentation in an XML file. The XML file is the same file you use for integrating corporate libraries.

If you use an XML configuration file and have specified settings for user-defined documentation there, the settings in the XML file take precedence. As soon as you refresh the XML configuration file or restart the TIA Portal, the settings from the XML file are applied. The settings that you have made in the TIA Portal lose their validity.

You can set the following options in the XML configuration file:

- Displaying the call log in the Inspector window
  A log of the call of user-defined documentation is displayed on the 'Info > General' tab of the Inspector window. The log helps you to adhere to the conventions for calling user-defined documentation.
- Search for user-defined documentation in a central file directory
  You can save user-defined documentation in a directory outside the current project directory in order, for example, to make documentation available across projects.
- Central directory for user-defined documentation
  You store cross-project documentation in the central file directory for user-defined documentation.

Procedure

To specify settings for the user-defined documentation, follow these steps:

1. Create an XML file named "CorporateSettings.xml", if you are not yet using an XML configuration file for the integration of company libraries. If you are already using a configuration file, proceed with step 3.
   The configuration file must be saved with "UTF-8" coding.
2. Save the file in the following directory on your computer:
   C:\ProgramData\Siemens\Automation\Portal V15_1\CorporateSettings\
3. Enter the content listed below into the XML configuration file.
4. Adapt the attributes for display of the user-defined documentation. The meaning of the individual elements is available in the comments in the XML configuration file. Use the value "true" to activate a function. Use the value "false" to deactivate a function.

Content of the XML configuration file

The XML configuration file must have the following content:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
 <Settings.Settings ID="0">
  <ObjectList>
   <Settings.General ID="1" AggregationName="General">
    <!-- Here you find the settings for global company libraries, if available. -->
    <ObjectList>

<!-- Section Corporate libraries -->
     <Settings.General ID="1" AggregationName="General">
     <AttributeList>
      <CorporateLibraryPaths>
      <!-- Example for two entries -->
      <Item>F:\CorporateLibraries\Corporate_Library_1\Corporate_Library_1.al15_1</
Item>          <Item>F:\CorporateLibraries\Corporate_Library_2\CorporateLibraries.al15_1</
Item>
      <!-- Here you enter additional global libraries, if any. -->           </
CorporateLibraryPaths>
    </AttributeList>
   </Settings.General>
<!-- Section Corporate libraries end -->

<!-- Section User-defined documentation -->
     <Settings.UserDocumentation ID="2" AggregationeName="UserDocumentation">
      <!-- In the following section, you specify the values for display of the user-defined
documentation. -->
      <AttributeList>
       <!-- Activates or deactivates the display of the access log. -->
       <DisplayLogInformation>
        <Value>true</Value>
        </DisplayLogInformation>
      <!-- Activates or deactivates the search for user-defined documentation in a central
directory. -->
       <EnableLookupFromCentralStorageLocation>
         <Value>true</Value>
       </EnableLookupFromCentralStorageLocation>
       <!-- Specifies the central directory for user-defined documentation. -->
       <CentralStorageLocation>
        <Value>D:\CorporateDocumentation\UserDocumentation\</Value>
       </CentralStorageLocation>
      </AttributeList>
     </Settings.UserDocumentation>
<!-- Section User-defined documentation end -->

    </ObjectList>
   </Settings.General>
  </ObjectList>
 </Settings.Settings>
</Document>
```

See also

Using user-defined documentation (Page 105)
Specifying settings in the TIA Portal (Page 108)
Calling user-defined documentation (Page 116)
Creating a homepage (Page 112)

9.4 Creating a homepage

You can design a homepage for user-defined documentation. The homepage can be an HTML page that you save either within a CHM file or in the directory of the relevant language. You can also use other file formats approved for user-defined documentation. You design the homepage of the user-defined documentation outside the TIA Portal.

Procedure

To create a homepage, follow these steps:

1. Design a file in HTML format or in any other file format approved for user-defined help.
2. Name the file "Home".
3. Copy the file to the central directory for user-defined documentation on the hard disk or on a network drive:
   <Central directory for user-defined documentation>\<Folder for the respective language>
4. If the respective language folder does not exist yet, create the folder now.
   Alternative: If you are creating the homepage for a CHM file, place the homepage in the main directory of the CHM file.

Sample configuration for the homepage

Below you see the correct path for the following conditions:

- The user-defined documentation is in Spanish.
- The homepage is an HTML file.

The path for these conditions is as follows:

<Central directory for user-defined documentation>\es-ES\Home.html

See also

Specifying settings with an XML file (Page 109)

Using user-defined documentation (Page 105)

Conventions for the creation (Page 113)

Calling user-defined documentation (Page 116)

Displaying the call log (Page 117)

Creating user-defined documentation (Page 118)

9.5 Conventions for the creation

You must observe some conventions to ensure that user-defined documentation is called at the correct location:

- the user-defined documentation must be saved in the correct directory.
- The file name must be exactly the same as the object name in the TIA Portal.

To prevent malicious code from being executed on your computer, only file formats that are considered as relatively safe are permitted.

Supported file formats

Create the user-defined documentation in one of the following file formats:

- Microsoft Word (.docx)
- Microsoft Excel (.xlsx)
- Microsoft PowerPoint (.pptx and .ppsx)
- HTML pages (.htm or .html)
- Microsoft XPS (.xps)
- Rich Text Format (.rtf)
- Text documents (.txt)
- Compiled HTML help (.chm)
- PDF documents (.pdf)

You save the homepage of the user-defined documentation in HTML format or save the homepage within a CHM file.

NOTICE
Infection of the computer with malicious code If the user-defined documentation contains malicious code, it can infect your computer. Especially HTML pages and CHM files can contain malicious code. Make sure that the user-defined documentation comes from a trustworthy source. You should also use the standard security measures, such as the use of a firewall and an up-to-date virus scanner.

Directories for user-defined documentation

Save the user-defined documentation in one of the following directories:

- Project folder:
  UserFiles\UserDocumentation\<Folder for the respective language>\<Object category>
- Directory of a global library:
  UserFiles\UserDocumentation\<Folder for the respective language>\<Object category>
- Central directory on the hard drive or a network drive:
  <Central directory for user-defined documentation>\<Folder for the respective language>\<Object category>\

The user-defined documentation must be located in the suitable subfolder for the respective language. The table below shows the respective language folders for the user languages installed as default:

Language	Subfolder
German	\de-DE
English	\en-US
Spanish	\es-ES
French	\fr-FR
Italian	\it-IT
Chinese	\zh-CN

The language folder must contain a separate subfolder for each object category. Create the corresponding subfolders for objects for which you are providing user-defined documentation. Always use the English designation of the object category. The table below shows the English designations of the most important object categories in the TIA Portal:

Object category	English designation
HMI screens	Screens
Organization blocks (OB)	Organization Blocks
Function blocks (FBs)	Function Blocks
Functions (FCs)	Functions
Data blocks	Data Blocks
Types in the library	Library Types
Master copies in the library	Master Copies
The project node in the project tree	Projects
All types of folders in the project tree, in the project library or in global libraries	Folders
All types of links in the project tree, for example, "Add new block", "Add new device", etc.	ShortCut
Libraries in the "Libraries" task card or in the library view	Libraries

If you are not sure of the English designation for an object category, change the user interface language of the TIA Portal to English. Alternatively, open the user-defined documentation for an object with <Shift+F1> and check in the call log which designation is expected for the object category.

Permitted file names

The file name must be exactly the same as the object name in the TIA Portal.

There are, however, restrictions for file names under Microsoft Windows. The same restrictions apply to the file system used to format the hard drive. The file name may only include certain characters and must not exceed a specific length. The restrictions for file names differ depending on the Windows version and the file system used for the hard drive.

To ensure that the help call works, read up on possible restrictions in the Microsoft Windows documentation.

Special features of CHM files

You store CHM files directly in the respective language folder. The folder for the respective object category must be included in the actual CHM file. Within the compiled CHM file, the names of the individual HTML files must also be exactly the same as the object names in the TIA Portal.

Note

Opening CHM files on network drives

If CHM files are saved on a network drive, the CHM files are not displayed correctly in more recent versions of Microsoft Windows. This behavior is determined by the security guidelines of the operating system. All versions of Microsoft Windows as of Windows Server 2003 SP1 are affected.

You can bypass the security guidelines by changing the registry database in Microsoft Windows.

To not compromise the security of your computer, save the CHM files only locally on your computer and do not change the registry database.

See also

Using user-defined documentation (Page 105)

Creating a homepage (Page 112)

9.6 Calling user-defined documentation

The user-defined documentation is opened in the language that is currently set as the user interface language. If there is no user-defined documentation available in the currently set user interface language, the English version of the user-defined documentation opens. If no user-defined documentation exists, a homepage is searched for.

Requirement

You have already saved user-defined documentation or a homepage according to the conventions.

Procedure

To open the user-defined documentation, follow these steps:

1. Select the object for which you want to display the user-defined documentation.
2. Press <Shift+F1>.
   The suitable user-defined documentation or the homepage opens.

See also

Using user-defined documentation (Page 105)

Creating a homepage (Page 112)

Specifying settings with an XML file (Page 109)

Displaying the call log (Page 117)

9.7 Displaying the call log

Use the call log to check whether the user-defined documentation is connected correctly. The call log shows the directories in which the search is performed for user-defined documentation or a homepage. The call log also displays the names that the individual files must have in order to call the user-defined documentation.

Requirement

The call log is enabled in the settings of the TIA Portal or using an XML configuration file.

Procedure

To display the call log, follow these steps:

1. Open the "Info" tab in the Inspector window.
2. Open the "General" tab.
3. Select the object for which you want to call the help.
4. Press <Shift+F1>.
   If possible, the matching user-defined documentation or the homepage of the user-defined documentation is opened. In any case, you will be informed in the Inspector window about which user-defined documentation is opened. You may be shown the directories in which no user-defined documentation was found.

See also

Calling user-defined documentation (Page 116)

Creating a homepage (Page 112)

9.8 Creating user-defined documentation

You create user-defined documentation for individual elements within a project or global library outside the TIA Portal. You can create the user-defined documentation in all available user interface languages.

If you create the user-defined documentation as CHM file, the procedure for creating the help is somewhat different to the creation process for other file formats.

Note the information provided in chapter "Conventions for the creation (Page 113)".

Creating user-defined documentation as single file

To create user-defined documentation as a single file, follow these steps:

1. Create a file in a valid file format.
2. Name the file identically to the object for which you want to call the user-defined documentation.
   If you are offering help for a library type, for example, name the help file identical to the type.
3. Depending on whether you are creating the user-defined documentation for project contents or for contents of a global library, copy the file to one of the following storage locations:
   - project folder under "UserFiles\UserDocumentation\<Folder for the respective language>\<Object category>"
   - Directory of a global library under "UserFiles\UserDocumentation\<Folder for the respective language>\<Object category>"
   - Central directory on the hard drive or a network drive:
     <Central directory for user-defined documentation>\<Folder for the respective language>\<Object category>\

   If the respective language folder or the folder for the object category does not exist yet, create the required folders before copying the file.

Sample configuration for user-defined documentation

Below you see the correct path for the following conditions:

- The user-defined documentation is intended for a type in a global library.
- The user-defined documentation is in French.
- The type is called "commande de moteur".
- The user-defined documentation is supplied with the global library.
- The user-defined documentation is created in Microsoft PowerPoint format.

The path for these conditions is as follows:

<Folder of the global library>\UserFiles\UserDocumentation\fr-FR\Library Types\commande de moteur.pptx

Creating user-defined documentation as CHM file

To provide user-defined documentation in CHM format, follow these steps:

1. Create a folder in Windows Explorer for each object category for which you want to create user-defined documentation. Use the English designation for the object category.
2. Create an HTML file for each object for which you want to provide user-defined documentation. Name the HTML file identically to the object for which you want to call the user-defined documentation. If you want to provide user-defined documentation for a library type, for example, name the HTML file identically to the type.
3. Store the HTML files in the corresponding folders of the respective object category.
4. Use the Microsoft HTML Help Workshop to create the CHM file. Use the prepared folder structure with the HTML files.
5. Copy the CHM file to one of the following storage locations:
   - Project folder under "UserFiles\UserDocumentation\<Folder for the respective language>"
   - Directory of a global library under "UserFiles\UserDocumentation\<Folder for the respective language>"
   - Central directory on the hard drive or a network drive:
     <Central directory for user-defined documentation>\<Folder for the respective language>

   If the respective language folder does not exist yet, create the language folder before you copy the CHM file.

See also

Creating a homepage (Page 112)

Conventions for the creation (Page 113)

Index

B

Block comparison
- Execute action, 39
- Updating comparison results, 36

Borders
- Placeholder for document information, 58
- Specifying the print area, 57

C

Calling user-defined documentation, 116
Canceling printing, 63
Compare editor
- Changing the view, 40
- Filtering the view, 33
- Overview, 26
- Showing and hiding columns, 32
- Specifying actions, 38

Conventions for user-defined documentation, 113
Cover page
- Placeholder for document information, 58

Creating a CHM file, 119
Creating a homepage for user-defined documentation, 112
Creating a print preview, 59
Creating labeling strips, 64
Creating user-defined documentation, 118

D

Detailed comparison
- Performing, 35

Device information, 49
Directories for user-defined documentation, 114
Displaying the call log, 117
Documentation editor
- Layout, 54

Documentation settings
- Using frames and cover pages, 52

Downloading
- Downloading to a memory card, 13
- Writing to a memory card, 13

E

Editing language, 73
Example of homepage for user-defined documentation, 112
Example of user-defined documentation, 118
Export of labeling strips, 64
Exporting labeling data as XML, 67

H

Help
- Preparing your own help contents, 105
- Settings for user-defined documentation, 108, 109

L

Loading
- Downloading project data to the device, 11
- General information, 9

M

Memory card
- Accessing, 99
- Adding a card reader, 98
- Introduction, 97
- Showing properties, 100

P

Password protection
- Protection concept, 41
- Revoking access rights, 43

Permitted file names, 115
PLC tag
- Updating comparison results, 36

Print scope, 46
Printable contents, 46
Printing
- Changing the settings, 47
- Creating a cover page, 52
- Creating frames, 51
- Documentation settings, 45
- Elements in the library, 50

Elements in the project tree, 50
Non-printable contents, 46
Non-printable objects, 47
Print contents, 49
Process a cover page, 53
Process borders, 53
Specify layout, 48
Specify print layout, 48
Specifying the print area, 57
Structure of printout, 45
Use borders, 50
Use borders and cover pages, 45
Use cover page, 48
Printing a project, 62
Printing labeling strips, 66
Determining the correction value for the print shift, 71
Project
Carrying out offline/offline comparisons, 24
Comparing data, 19
Comparing offline/online, 23
Compiling project data, 8
Detailed comparison, 35
Project data
Compiling, 8
Downloading to a memory card, 13
Information on compiling, 7
Information on loading, 9
Loading, 11
Uploading from a device, 15
Writing to a memory card, 13
Project language, 73
Project languages
Changing the editing language, 78
Specifying project languages, 77
Use, 73
Project texts
Application example, 86
Changing the text of selected objects, 80
Displaying reference text, 81
Exporting all texts, 83
Exporting individual texts, 82
Exporting texts of a device, 83
Importing, 85
Translating project texts, 79
Translation into project languages, 73
Protection concept
Introduction, 41
Revoking access rights, 43
Protection level
Revoking access rights, 43

R

Reference language, 73

S

Settings for user help, 109
Showing and hiding columns, 32
Special features of CHM files, 115
Supported file formats, 113
System texts, 73
System-defined text lists
Editing, 93
Modifying texts, 93

T

Text lists
Introduction, 89
System-defined, 89
Text lists editor, 89
User-defined, 89
Texts
Exporting all texts, 83
Exporting individual texts, 82
Exporting texts of a device, 83
Importing, 85
Translating texts, 73

U

Uploading
from a device, 15
User texts, 73
User-defined documentation, 105
Enabling the call log, 108, 109
Specifying settings, 108, 109
Specifying the central file directory, 108, 109
User-defined text lists
Creating, 91
Editing, 92
Editing value ranges and texts, 92
Using frames and cover pages from the library, 52

SIEMENS

Using user administration

Programming and Operating Manual

Basics of user administration property in the TIA Portal 1

Activating user administration 2

Setting password policies 3

Managing project users 4

Managing global users and user groups 5

Managing roles 6

Assigning roles 7

Log on to a protected project 8

Changing your own password 9

Logging off the project 10

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Basics of user administration property in the TIA Portal....5
2 Activating user administration....9
3 Setting password policies....11
4 Managing project users....13
5 Managing global users and user groups....17
6 Managing roles....21
7 Assigning roles....25
8 Log on to a protected project....27
9 Changing your own password....29
10 Logging off the project....31

Basics of user administration property in the TIA Portal 1

Introduction

The TIA Portal provides the possibility to perform user administration for projects. In this way, for example, a project can be protected against unintentional or unauthorized modification. A user sets up the project protection to activate user management. This user is created as a project administrator. Once the project protection has been activated, the project can only be opened and edited by authorized users. Note that project protection cannot be revoked.

Permissible users are local project users and global users and user groups added by a project administrator:

- Local project users:
  Users who are defined and managed in a TIA Portal project. These user accounts are only valid for this one project. It makes sense to use project user accounts if the entire automation solution is engineered in one project.
- Global users and user groups:
  These user accounts are defined and managed outside the TIA Portal in UMC (User Management Component). Global users and user groups can be imported into the various TIA Portal projects in which these users will work. To add users and user groups from UMC, the corresponding rights are required in UMC.

You can assign specific roles to users or user groups, which in turn can be linked to different function rights. There are the following general engineering role permissions to which a role can be assigned:

- Open the project read-only
  A user who has only this role permission can open the project, but not modify it. However, he cannot open the editors for user administration.
- Open the project with write rights
  A user with this role permission can modify the project. However, he cannot open the editors for user administration with this.
- Managing users and roles
  A user with this role permission can manage the users and roles of the project. In addition, the user still requires the "Open the project with write rights" role permission because the project cannot be opened or edited with the "Manage users and roles" role permission alone.

In addition to these general role permissions, further runtime rights can be assigned. The descriptions of these specific function rights can be found in the corresponding areas of the help. Several function rights can be assigned to a role.

When you enable user administration, the system creates the following two roles:

- "ES Administrator"
  The first created project user is assigned the role "ES Administrator". This role has all three engineering role permissions by default. Each project needs at least one administrator who is allowed to edit the project and the security settings. In addition, other users may be assigned the right to manage users and roles.
- "ES Standard"
  Users who are assigned the role "ES Standard" have the permissions "Open project read-only" and "Open project read/write".

You cannot change or delete these system roles. You can create additional roles and assign them the required role permissions.

Note

Additional local user administrations

In addition to the user administration for projects, there are additional user administrations in certain areas of the TIA Portal, e.g. for WinCC Panels.

Settings for users

A project administrator can make the following settings for a local project user:

- User name:
  Name of the local project user who must be used to log on to the project.
- Password:
  The password of the local project user assigned by the administrator with which the project user can log onto the project. The project user can change the password later.
- Authentication method:
  Authentication methods defined by the administrator. The following two methods are available:
  - Password: Login is via a password which was defined in the TIA Portal or in UMC.
  - Radius: Login is via a RADIUS server on which the password is stored. This option can only be used for devices that support login via a RADIUS server.
- Maximum session duration
  Maximum duration for which the user can be logged on to a device. The user is logged off from the device after the time has expired. This setting is only available for devices that support a session duration. The session duration is not evaluated in the TIA Portal itself.

UMC - User Management Component

In addition, you can install the "User Management Component UMC" software package on one or multiple computers; it provides a central user administration. This creates a system of possibly interactive UMC installations (UMC ring server, UMC server). Then you can define users and user groups in the UMC system or import them from the Windows Active Directory. When UMC is installed, you can access the UMC server from the TIA Portal to add the users and user groups defined there to the user administration of the TIA Portal and use roles to assign them specific role permissions in a TIA Portal project.

However, you cannot change the data of users and user groups added from UMC within the TIA Portal. For example, even as administrator in the project you cannot change passwords or other data of UMC users or UMC user groups. This is only possible in UMC. However, you have the option to synchronize user administration in the TIA Portal with UMC or to check the synchronization status. This enables you to eliminate inconsistencies between the global users and user groups in UMC and the UMC users or UMC user groups already imported in the TIA Portal.

Note

UMC documentation

The UMC installation file and the UMC documentation in English is available on the TIA Portal installation data storage medium ("..\support", "...\Documents\Readme\English").

We highly recommend that you read the UMC documentation completely before you start working with the user administration, especially the sections on "Secure Application Data Support (SADS)". SADS is mandatory for working with the user administration in the TIA Portal.

User administration in the TIA Portal and Multiuser Engineering

Note the following information when working with the user administration in the TIA Portal in combination with Multiuser Engineering:

- You can only make changes to the user administration of a multiuser project in a server session and not in the local sessions.
- When you upload a single-user project that is protected with user administration to a multiuser server, the resulting multiuser project is also protected. The users and user groups contained in the project continue to exist with their passwords and can therefore continue to log into the local sessions of the multiuser project, provided they have the corresponding user rights.
- If you protect a previously unprotected local session using the user administration, this protection is removed once again after checking-in and updating the local session, as the multiuser project on the server is not protected as a result. This also applies when several operators are protecting their local sessions.
- If you protect the multiuser project on the server using the user administration, the local sessions created by this multiuser project are also protected as soon as these are updated.
- When you log off from your local session, which is protected using the user administration, other users remain logged on to their local sessions of the multiuser project. You only log yourself off. Your local session is the closed as a result.
- If the multiuser project on the server is protected using the user administration during processing of your unprotected local session and your user account has not been assigned the "Open project with read/write permission" role permission, you cannot check-in any changes to your local session or update your local session.

More information on Multiuser Engineering can be found in section "Using Multiuser Engineering".

See also

Activating user administration (Page 9)
Setting password policies (Page 11)
Managing project users (Page 13)
Managing global users and user groups (Page 17)
Managing roles (Page 21)
Assigning roles (Page 25)
Log on to a protected project (Page 27)
Changing your own password (Page 29)
Logging off the project (Page 31)

Activating user administration 2

You can use the user administration in a TIA Portal project to protect your project against unauthorized access. A project administrator is automatically created when you activate the user administration. The project administrator can create additional users. When the user administration is activated, the project can only be opened and changed with a user account with sufficient rights.

Note

Deactivation of user management not possible

Note that user management for the project cannot be disabled once it is enabled. This also means that the protection of the project cannot be reversed.

Procedure

Proceed as follows to activate user administration:

1. Open the "Security settings" folder in the project tree.
2. Double-click the "Settings" command.
   The editor for the user administration is opened and the area for project protection is displayed.
3. Click on "Protect this project".
   The "Protect project" dialog opens.
4. Enter a user name.
5. Enter a password.

   Note

   Password guideline

   The following guidelines are defined for the password when the user administration is activated for the first time:

   - Minimum password length: 8 characters
   - Minimum number of numeric characters: 1
   - Minimum number of special characters: 0
   - At least one uppercase and one lowercase letter: Activated
   - Number of passwords blocked for reuse: 5
   - Activate password aging: Deactivated

   As project administrator you can adapt these password guidelines afterwards to meet your requirements.

   ---

6. Re-enter the password to confirm.

7. If necessary, enter a comment.
8. Confirm your entries with "OK".

Note

Project administrator

The first created user is the project administrator. Each project needs at least one administrator.

Result

User management has been activated. You are logged on as project administrator and you can now create further local users or add global users or user groups from UMC or create new roles.

See also

Basics of user administration property in the TIA Portal (Page 5)

Setting password policies

3

As a project administrator, you can define the structure and complexity of user passwords. To do this, you can specify the following policies:

- "Minimum password length"
  The minimum number of characters that the user password must have.
- "Minimum number of numeric characters"
  The minimum number of numeric characters that the user password must contain.
- "Minimum number of special characters"
  The minimum number of special characters that the user password must contain.
- "At least one upper case and one lower case letter"
  Specifies that the user password must contain at least one uppercase and one lowercase letter.
- "Number of recently used passwords blocked for reuse"
  Sets the number of recently used passwords that cannot be used as a new password.
- "Enable password aging"
  Specifies that the password only has a certain validity period and then expires. If this option is selected, the password must be changed within the password validity.
- "Password validity (days)"
  Sets the password validity in days, in which the password must be changed when the password is activated.
- "Prewarning time (days)"
  Sets how many days before the user's password expires, a warning is provided to the user. The warning time must be less than that for the password validity.

Note

Permissible characters for usernames and passwords

You can use the following numbers, letters and special characters for usernames and passwords:

- 0123456789
- A...Z a...z
- !#$%&()*+,-./:;<=>?@ [\]_{|}~^

Requirement

You have a user account as a project user or as a global user that is associated with the following two functional rights:

- Open the project with write rights
- Managing users and roles

Procedure

To set the password policies, follow these steps:

1. Open the protected project for which you want to set the password policies.
2. Open the "Security settings" folder in the project tree.
3. Double-click the "Settings" command.
   The editor for the user administration is opened and the area for project protection is displayed.
4. Click "Password policies".
5. Make all the desired settings.
6. Save the project.

See also

Basics of user administration property in the TIA Portal (Page 5)

Managing project users

4

To manage local users in your project you can perform the following actions:

- Create a new project user
- Change the user name of a project user
- Change the password of a project user
- Change the authentication procedure of a project user
- Change the maximum session duration of a project user
- Change a comment
- Delete project users

Please note the following restrictions:

- A maximum of 256 project users can be added.
- The user name may not exceed 255 characters.
- The password may not exceed 120 characters and must meet the defined password guidelines.
- The comment may not exceed 1000 characters.

Requirement

- A protected project is open.
- You are logged on with a user account with the "Manage users and roles" and "Open the project with write rights" role permissions.

Creating a new project user

To create a new project user, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.
4. Click "Add new user".
   A submenu is opened in which you can select the user type.
5. Click "Add new project user".
6. Enter a user name.
7. Click on the arrow in the field "Password".
8. Enter a password.
9. Re-enter the password to confirm.

10. Select the authentication method and the maximum duration for the new project user.
11. If necessary, enter a comment.
    The new project user has been created. To work with this user account in the project, you should next assign the necessary roles to it.

Note

You can also create a new project user by copying an existing project user. The roles assigned to the original project user are also assigned to the copied project user. However, you must assign a new password for the copied project user.

Changing the user name of a project user

To change the user name of a project user, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.
4. Click in the field with the user name that you want to change.
5. Enter the new user name.

Changing the password of a project user

To change the password of a project user, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.
4. Click the arrow next to the password you want to change.
5. Enter the new password.
6. Re-enter the password to confirm.

Changing the authentication procedure of a project user

To change the authentication procedure of a project user, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.
4. Click the arrow next to the authentication procedure you want to change.
5. Select the desired authentication method.

Changing the maximum session duration of a project user

To change the maximum session duration of a project user, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.
4. Enter the new duration for the project user for whom you want to change the session duration. Alternatively, you can select the new duration using the arrow keys.

Changing a comment

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.
4. Click in the field with the comment that you want to change.
5. Enter the new comment.

Deleting project users

To delete a project user, proceed as follows:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.
4. Select the project user you want to delete.
5. In the shortcut menu select the "Delete" command or use the <Del> key.

Note

You must keep at least one project user account which has been assigned the role "ES Administrator".

See also

Basics of user administration property in the TIA Portal (Page 5)

Managing global users and user groups 5

You can add global users and user groups which were created in UMC (User Management Component) to a protected project. This enables these users and user groups to work with the project if they are assigned the corresponding rights.

If a user has to access several protected projects, you minimize your management effort by adding this user individually as a global user or together with a user group to each project. Global users can be managed in UMC and the changes are then effective in every project to which the users or user groups belong. You have the option to synchronize the user administration in the TIA Portal with UMC so that no inconsistencies occur if users or user groups are changed in UMC. To determine whether synchronization is necessary, you can check the synchronization status. You can find information on synchronization in the banner of the user administration.

Please note the following restrictions with regard to global users and user groups:

- A maximum of 256 global users can be added.
- A maximum of 50 global user groups can be added.
- Global users and user groups cannot be copied.

You can display the following information for global users and user groups:

- User groups: The users they contain and whether they have already been imported into the user administration.
- User: The user groups of which the user is a member and whether they have already been imported into the user administration.

Requirement

- A protected project is open.
- You are logged on with a user account with the "Manage users and roles" and "Open the project with write rights" role permissions.
- The connection to UMC is configured and you have a user account with the corresponding rights in UMC.
  For further information on installation and configuration of UMC, refer to the English documentation on UMC on the installation data carrier in the directory "/document/Readme/English".

Adding global users

To add a global user from UMC, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.

4. Click "Add new user".
   A submenu is opened in which you can select the user type.
5. Click "Add user from UMC".
   If you are not yet logged on in UMC, the "Login" dialog opens. Then also perform steps 6 and 7.
6. Enter your UMC user name and the corresponding password.
7. Click "OK".
   As soon as you are logged on in UMC, the "Add user from UMC" dialog opens. All the available users from UMC are displayed in this dialog. Already activated users are activated and write-protected.
8. Activate the users you wish to add to your TIA Portal project.

   Note

   To find the required users more quickly, you can filter the table by the columns "Name" and "Long name". To do this, enter part of the name or long name n the first line. All users whose names or long names contain the text entered are displayed. To cancel filtering, click the Filter button or select "*" from the drop-down list.

9. Click "OK" to add the selected users.
   The selected users are added as global users. Your data is write-protected and cannot be changed within the TIA Portal project. However, a global user can change his own password.

Deleting global users

To delete a global user from the TIA Portal project, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.
4. Select the global user you want to delete. You can also delete several users at the same time via multi-selection.
5. In the shortcut menu select the "Delete" command or use the <Del> key.

Adding a global user group

To add a global user from UMC, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "User groups" tab.
4. Click "Add new user group".
   If you are not yet logged on in UMC, the "Login" dialog opens. Then also perform steps 5 and 6.
5. Enter your UMC user name and the corresponding password.

6. Click "OK".
   As soon as you are logged on in UMC, the "Add user group from UMC" dialog opens. All the available user groups from UMC are displayed in this dialog. Already activated user groups are activated and write-protected.
7. Activate the user groups that you wish to add to your TIA Portal project.
8. Click "OK" to add the selected user groups.
   The selected user groups are added as global user groups. Your data is write-protected and cannot be changed within the TIA Portal project.

Deleting a global user group

To delete a global user group from the TIA Portal project, proceed as follows:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "User groups" tab.
4. Select the global user group that you want to delete. You can also delete several user groups at the same time via multi-selection.
5. In the shortcut menu select the "Delete" command or use the <Del> key.

Display users of an added global user group

To display the users included in a global user group, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "User groups" tab.
4. Select the user group for which you want to display the users. Note that you cannot use multiple selection.
5. In the bottom area of the "User groups" tab open the "Users" tab.
   The users of the selected group are displayed. The check box is enabled in the "Imported" column for users who have already been imported into the user administration.

Display user groups of a global user

To display the user groups for an added global users of which they are a member, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.

4. Select the user for which you want to display the user groups of which the user is a member. Note that you cannot use multiple selection.
5. In the bottom area of the "User" tab open the "User groups" tab.
   The user groups of which the user is a member are displayed. The check box is enabled in the "Imported" column for user groups that have already been imported into the user administration.

Checking the synchronization status

To check the synchronization status, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Show the banner with the synchronization information, and click the "Check status" link.

Or:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab or the "User groups" tab.
4. Click "Check status" in the toolbar.

Performing synchronization

To synchronize, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Show the banner with the synchronization information and click the "Synchronize" link.

Or:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab or the "User groups" tab.
4. Click "Synchronize" in the toolbar.

See also

Basics of user administration property in the TIA Portal (Page 5)

Managing roles 6

Rights are assigned to users via roles. The system-defined roles "ES Administrator" and "ES Standard" are always available in the TIA Portal. You cannot rename or remove these roles. You cannot change the assignment of the function rights to these roles.

You can, however, define some roles and assign specific function rights to them. You can also perform the following actions for the roles you defined yourself:

- Change name of role
- Change maximum session duration
- Change a comment
- Changing or deleting the assignment of function rights
- Delete a role

You can display the assigned role permissions for each project user or global user and for each global user group. This will give you an overview of the assigned rights at any time.

Requirement

- A protected project is open.
- You are logged on with a user account with the "Manage users and roles" and "Open the project with write rights" role permissions.

Defining a new role

To define a new role, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Roles" tab.
4. Click "Add new role".
5. Enter a name for the role.
6. If necessary, change the maximum session duration.
7. If necessary, enter a comment.
   The new role has been created. Next, you need to assign the necessary function rights to it.

Note

You can also create a new role by copying an existing role. This means the assigned role permissions are also assigned to the new role.

Assigning function rights

To assign function rights to a role, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Roles" tab.
4. Select a role. Please note that you cannot use multi-selection for assignment.
5. In the lower area "Function rights categories", open the categories from which you want to assign the role permissions.
6. Activate the function rights you wish to link to the role.

Changing the name of a role

To change the name of a role, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Roles" tab.
4. Click in the field with the name of the role that you want to change.
5. Enter the new role name.

Change maximum session duration

To change the maximum session duration for a role, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Roles" tab.
4. Enter the new duration for the role for which you want to change the session duration. Alternatively, you can select the new duration using the arrow keys.

Changing a comment

To change the comment for a role, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Roles" tab.
4. Click in the field with the comment that you want to change.
5. Enter the new comment.

Changing or deleting the assignment of function rights

To change the assignment of function rights for a role, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Roles" tab.
4. Select the role for which you want to change the assignment of the function rights.
5. In the lower area "Function rights categories", open the categories from which you want to assign the function rights or remove the assignment.
6. Activate the function rights that you wish to assign to the role.
7. Deactivate the function rights that are no longer assigned to the role.

Deleting a role

To delete a role, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Roles" tab.
4. Select the role that you want to delete. Note that you cannot delete the system-defined roles "ES Administrator" and "ES Standard".
5. In the shortcut menu select the "Delete" command or use the <Del> key.

Display assigned function rights of a project user or of a global user

To display the assigned function rights of a project user or of a global user, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.
4. Select the project user or the global user for which you want to display the assigned function rights. Note that you cannot use multiple selection.
5. In the bottom area of the "Users" tab open the "Assigned rights" tab.

Display assigned function rights of a global user group

To display the assigned function rights of a global user group, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "User groups" tab.

4. Select the global user group for which you want to display the assigned function rights. Note that you cannot use multiple selection.
5. In the bottom area of the "User groups" tab open the "Assigned rights" tab.

See also

Basics of user administration property in the TIA Portal (Page 5)

Assigning roles 7

You can assign roles to project users or global users and user groups that have different function rights. You can revoke the assignments at any time.

Requirement

- A protected project is open.
- You are logged on with a user account with the "Manage users and roles" and "Open the project with write rights" role permissions.

Assigning roles to project users and global users

To assign roles to a user, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.
4. Select the user to whom you want to assign roles. Note that you cannot use multiple selection.
5. Enable the desired roles in the "Assigned roles" section.

Assigning roles to global user groups

To assign roles to a global user group, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "User groups" tab.
4. Select the user group to which you want to assign roles. Note that you cannot use multiple selection.
5. Enable the desired roles in the "Assigned roles" section.

Revoking role assignments for project users and global users

To revoke a role for a user, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "Users" tab.

4. Select the user for which you want to revoke role assignments. Note that you cannot use multiple selection.
5. In the "Assigned roles" section, clear check marks for the roles that are no longer to be assigned to the user.

Revoking role assignments for global user groups

To revoke a role assignment for a global user group, follow these steps:

1. Open the "Security settings" folder in the project tree.
2. Double-click on the command "Users and roles".
3. Open the "User groups" tab.
4. Select the user group for which you want to revoke the role assignments. Note that you cannot use multiple selection.
5. In the "Assigned roles" section, clear check marks for the roles that are no longer to be assigned to the user group.

See also

Basics of user administration property in the TIA Portal (Page 5)

Log on to a protected project

8

Requirement

You have a user account as a project user or as a global user that is associated with at least one of the following two functional rights:

- Open the project read-only
- Open the project with write rights

Procedure

To log on to a protected project, follow these steps:

1. Select the "Open" command in the "Project" menu.
   The "Open project" dialog box opens and the list of recently used projects is displayed.
2. If you want to check whether the project has been inadvertently modified when you open the selected project outside the TIA Portal, select the "Activate basic integrity check" check box. The basic integrity check may take some time to complete.
   See also: The basics of projects
3. Select the protected project from the list that you want to open. If the desired project does not exist in the list, perform the following steps:
   – Click "Browse".
     The "Open existing project" dialog opens.
   – Navigate to the desired project folder and select the project file.
4. Click "Open".
   The "Log on" dialog opens.
5. Select the required user type.
6. Enter your user name.
7. Enter your password.
8. If you want to change the password for your user account, click "Change password".

   Note

   If you log in as a global user, the password change is mandatory at the first logon, if this has been defined in UMC. Then perform the steps for changing the password.

   See also: Changing your own password (Page 29)
9. Click "OK".
   The project is opened when you have logged on with a user account that has the appropriate function rights.

See also

Basics of user administration property in the TIA Portal (Page 5)

Changing your own password

9

You can change your password either when you log on to the project or at any time when working in a protected project. If you change your password as a project user, only the password for this project will be changed. If you change your password as a global user, your password will be changed in UMC. This will change your password for all projects for which you are registered as a user.

Requirement

You are logged on with a user account with the "Open the project with write rights" role permission.

Changing a password at logon

To change your password when you log on to the project, follow these steps:

1. Select the "Open" command in the "Project" menu.
   The "Open project" dialog box opens and the list of recently used projects is displayed.
2. Select a protected project from the list.
3. Click "Open".
   The "Log on" dialog opens.
4. Click "Change password".
   The "Change password" dialog opens.
5. Enter your user name.
6. Enter your current password.
7. Enter your new password.
8. Enter your new password again for confirmation.
9. Click "OK" to change your password.
   If you enter all the data correctly, your password will be changed. You can then use this new password when you log on to a protected project.

Changing a password when working in a protected project

To change your password while working in a protected project, follow these steps:

1. Open the project view.
2. Click the down arrow in the toolbar next to the "User Management" button.
   A drop-down list is opened in which the user management functions are listed.
3. Click "Change password".
   The "Change password" dialog opens.
4. Enter your user name.

5. Enter your current password.
6. Enter your new password.
7. Enter your new password again for confirmation.
8. Click "OK" to change your password.
   If you enter all the data correctly, your password will be changed. You can then use this new password when you log on to a protected project.

See also

Log on to a protected project (Page 27)

Basics of user administration property in the TIA Portal (Page 5)

Logging off the project

10

You can log off explicitly from a protected project. The project is closed.

Procedure

To log off the project, proceed as follows:

1. Open the project view.
2. Click the down arrow in the toolbar next to the "User Management" button.
   A drop-down list is opened in which the user management functions are listed.
3. Click "Log off and close the project".
   The "Log off and close project" dialog is opened.
4. Click "Yes" to log off and close the project.
5. Click "No" to stay logged on to the project.
6. If you clicked "Yes" and you have made changes to the project since the last save, a message is displayed. Then, specify whether the changes should be saved.

See also

Basics of user administration property in the TIA Portal (Page 5)

SIEMENS

Using libraries

Programming and Operating Manual

Library basics 1

Using libraries in different TIA products 2

Using the "Libraries" task card 3

Using the library view 4

Using library management 5

Using global libraries 6

Using master copies 7

Using types and their versions 8

Multiuser Engineering in libraries 9

Updating and upgrading types of a library 10

Using library texts 11

Creating folders in a library 12

Editing library elements 13

Harmonizing names and path structure 14

Comparing library elements 15

Clean up library 16

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Library basics....7
2 Using libraries in different TIA products....9
3 Using the "Libraries" task card....11
3.1 Overview of the "Libraries" task card....11
3.2 Using the element view....14
4 Using the library view....15
4.1 Overview of the library view....15
4.2 Opening and closing the library view....18
5 Using library management....19
5.1 Overview of the library management....19
5.2 Opening library management....23
5.3 Filtering the display of types....24
5.4 Displaying instances in the project....25
5.5 Displaying cross references of an instance....26
5.6 Displaying relations to other library objects....27
6 Using global libraries....29
6.1 Creating a global library....29
6.2 Create write-protected global library....30
6.3 Compatibility of global libraries....31
6.4 Opening a global library....32
6.5 Defining library languages....33
6.6 Displaying properties of global libraries....34
6.7 Displaying logs of global libraries....35
6.8 Saving a global library....36
6.9 Closing a global library....37
6.10 Deleting a global library....38
6.11 Archiving and retrieving global libraries....39
6.11.1 Working with archives of global libraries....39
6.11.2 Archiving global libraries....39
6.11.3 Retrieving global libraries....40
6.12 Using global corporate libraries....42
6.12.1 Basics of corporate libraries....42
6.12.2 Creating a configuration file for corporate libraries....43

7 Using master copies....47

7.1 Basics on master copies....47

7.2 Adding master copies....48

7.3 Filtering master copies....50

7.4 Using master copies....51

8 Using types and their versions....53

8.1 Basics on types....53

8.2 State of type versions....56

8.3 Displaying a released type version....58

8.4 Displaying properties of a type or version....59

8.5 Working with types in the project library....60
8.5.1 Adding types to the project library....60
8.5.2 Block requirements....61
8.5.3 Referencing of objects within a block....62
8.5.4 Duplicating types....63
8.5.5 Using types of the project library....64
8.5.6 Displaying types for an instance....66
8.5.7 Creating a test version of a type....67
8.5.8 Editing test version of a type....68
8.5.9 Creating an editing version of a type....69
8.5.10 Performing a consistency check for a version....69
8.5.11 Discarding versions....70
8.5.12 Discarding all versions within a folder....71
8.5.13 Release versions....71
8.5.14 Releasing all versions within a folder....72
8.5.15 Updating the project to the latest versions....73
8.5.16 Remove the link between instance and type....75

8.6 Working with types in global libraries....77
8.6.1 Adding types to a global library....77
8.6.2 Using types from a global library....78
8.6.3 Updating the project to the latest type versions....79

8.7 Assigning a version....82

9 Multiuser Engineering in libraries....85

9.1 Multiuser Engineering with library elements....85

9.2 Checking in types and master copies....87

10 Updating and upgrading types of a library....89

10.1 Updating library with the types of another library....89

10.2 Upgrading types in an existing project....91

10.3 Example of the updating and upgrading of types....94

11 Using library texts....99

11.1 Exporting library texts....99

11.2 Importing library texts....102
11.3 Defining the project languages in the project library....104
11.4 Updating imported library tests for type instances....105
12 Creating folders in a library....107
13 Editing library elements....109
14 Harmonizing names and path structure....111
15 Comparing library elements....113
16 Clean up library....115
Index....117

Library basics

1

Introduction

You can store library elements you want to reuse in libraries. For each project there is a project library that is connected to the project. In addition to the project library, you can create any number of global libraries that can be used over several projects. Since the libraries are compatible with each other, library elements can be copied and moved from one library to another. Libraries are used, for example, to create templates for blocks that you first paste into the project library and then further develop there. Finally, you copy the blocks from the project library to a global library. You make the global libraries available to other employees working on your project. The other employees continue to use the blocks and adapt them to their individual requirements as needed.

Both the project library and global libraries distinguish between two different types of library elements:

- Types
  Objects that are required to run user programs, for example, blocks, PLC data types (UDT), user-defined data types or faceplates are suitable as types. Types can be versioned and therefore support professional further development. Projects using the types can be updated as soon as new versions of the types are available.
- Master copies
  Almost every object can be saved as a master copy and pasted into the project again later. You can, for example, save entire devices with their contents or cover sheets for plant documentation as master copies.

Project library

Each project has its own library, the project library. In the project library, you store the objects you want to use more than once in the project. The project library is always opened, saved, and closed together with the current project.

Global libraries

In addition to the project library, you use global libraries if you want to use libraries over several projects. Global libraries exist in three versions:

- System libraries
  Siemens supplies global libraries for its own software products. These include off-the-peg functions and function blocks that you can use within your project. The supplied libraries cannot be changed. The supplied libraries are loaded automatically matching the project.
- Corporate libraries
  Corporate libraries are made available centrally by your organization, for example, in a central folder on a network drive. The TIA Portal manages the corporate libraries automatically. As soon as a more recent version of an existing corporate library becomes available, you receive a prompt to update the corresponding corporate library to the more recent version.
- User libraries
  Global user libraries are independent of a specific project and can therefore be passed on to other users. Shared access to global user libraries is also possible, for example on a network drive, if all users open the global user library with write protection.
  Global user libraries from older versions of the TIA Portal that you created yourself can still be used. To continue using global user libraries from older versions of the TIA Portal, they must first be upgraded.

Comparing library elements

You can compare blocks and PLC data types (UDT) with the objects of a device. This allows you to determine, for example, whether certain blocks or PLC data types (UDT) have been used in a project and whether they have been modified.

See also

Overview of the "Libraries" task card (Page 11)

Overview of the library view (Page 15)

Overview of the library management (Page 19)

Basics on master copies (Page 47)

Basics on types (Page 53)

Using libraries in different TIA products

2

Description

You can use different TIA products (e.g. STEP 7 Professional, Safety, WinCC Advanced, etc.) within a TIA Portal version (e.g. V15.1). You can also open projects with project libraries and global libraries that you have created with one TIA product with a different TIA product at any time. However, the library elements can only be edited and used in the project if they are supported in the current project and when all necessary licenses and support packages are available. You cannot, for example, edit STL program blocks, with STEP 7 Basic.

If a master copy or a type or its versions are not supported, the block icon in the library and in the project tree is also marked with the icon for library elements that are not supported.

If a master copy or a type is marked as unsupported, the following functions are then available to you:

- rename
- delete
- move within the library
- copy to a different library, as long as this library also supports the library element

Even if a type is marked as unsupported, the following functions are also available to you:

- The "Go to type" function
- The library management
- "Clean up library" function
- Upgrade library as long as it does not contain a type version that is not supported
- Assign type version as long as it does not contain a type version that is not supported

Example

The example below shows you what it looks like when you open a project library from a more comprehensive product in a less comprehensive product:

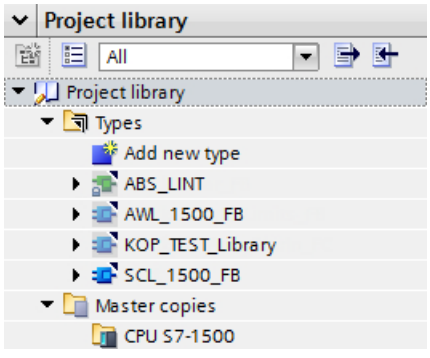

The project library was created with STEP 7 Professional and includes types that are specific to Professional.

When you open the STEP 7 Professional project with STEP 7 Basic you will see a note that the contained libraries cannot be edited without restrictions.

- Representation in the project library:

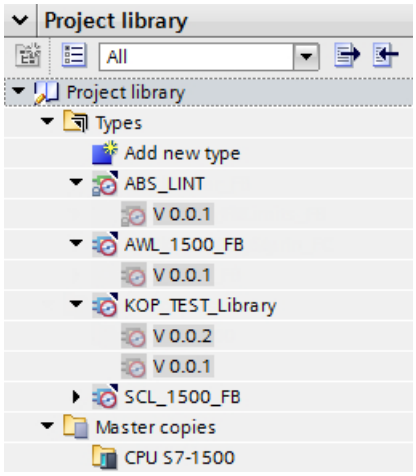

- Representation in the project tree:

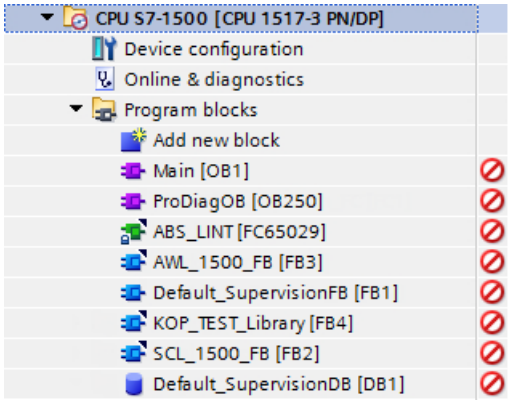

All STEP 7 Professional specific types are marked accordingly as not supported and can only be edited with restrictions.

Using the "Libraries" task card

3

3.1 Overview of the "Libraries" task card

Function of the "Libraries" task card

The "Libraries" task card enables you to work efficiently with the project library and the global libraries.

Layout of the "Libraries" task card

The "Libraries" task card consists of the following components:

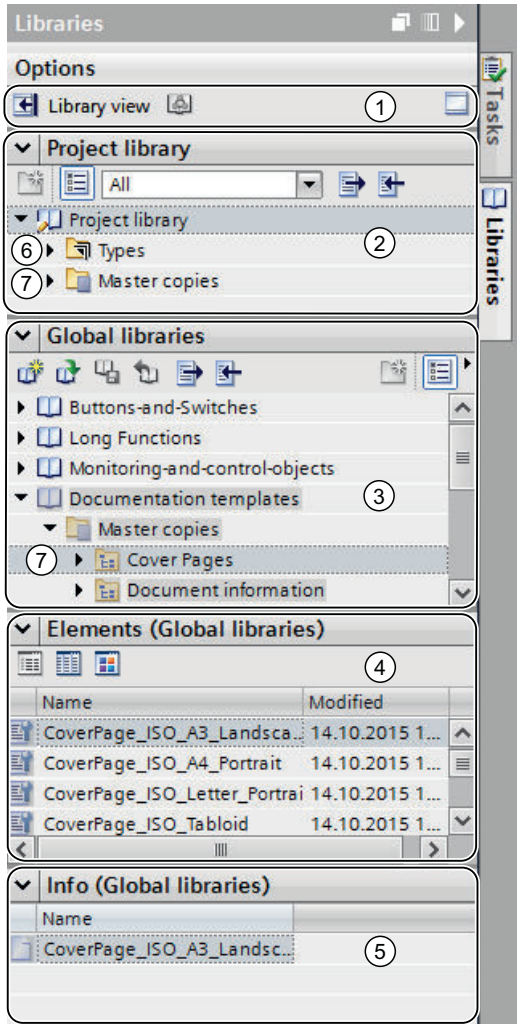

① "Library view" and "Library management" buttons
② "Project library" pane
③ "Global libraries" pane
④ "Elements" pane
⑤ "Info" pane
⑥ "Types" folder
⑦ "Master copies" folder

"Library view" button

The "Library view" button is used to change to the library view. The "Libraries" task card and the project tree are hidden with this action.

See also: Using the library view (Page 15)

"Project library" pane

In the "Project library" pane, you can store the library elements that you want to use more than once in the project.

"Global libraries" pane

In the "Global libraries" pane, you manage the global libraries whose library elements you want to reuse over several projects.

The "Global libraries" pane also lists libraries that were shipped together with the products you purchased. These libraries provide you with ready-made functions and function blocks, for example. The supplied global libraries cannot be edited.

"Elements" pane

In this pane, you can display the contents of folders in the library. The "Elements" pane is not displayed by default. If you want to display the "Elements" pane, you have to enable it first. Three view modes are available in the "Elements" palette:

- Details mode
  The properties of folders, master copies and types are shown in table form in details mode.
- List mode
  In list mode, the contents of folders are listed.
- Overview mode
  In overview mode, the contents of folders are displayed with large symbols.

See also: Using the element view (Page 14)

"Info" pane

You can display the contents of the library elements in the "Info" pane. The individual versions of types and the last revision date of the version are also displayed.

"Types" folder

In the "Types" directories, you can manage types and type versions of library elements that you use as instances in the project.

See also: Using types (Page 53)

"Master copies" folder

In the "Master copies" directories, you can manage master copies of library elements that you can use as copies in the project.

See also: Using master copies (Page 47)

See also

Library basics (Page 7)

Comparing library elements (Page 113)

3.2 Using the element view

Introduction

When you open the "Libraries" task card the first time, the "Project library" and "Global libraries" palettes are opened and the "Info" palette is closed. You can display the "Elements" palette if needed.

The elements view shows the elements of the selected library. Three view modes are available in the elements view:

- Details
  The properties of folders, master copies and types are shown in table form in details mode.
- List
  In list mode, the contents of folders are listed.
- Overview
  In overview mode, the contents of folders are displayed with large symbols.

The "Info" palette shows the contents of the selected library element. If you select a type in the elements view, for example, the type versions are displayed in the "Info" palette.

Requirement

The "Libraries" task card is displayed.

Procedure

To use the element view, follow these steps:

1. Click "Open or close the element view" in the "Project library" or "Global libraries" pane.
2. To change the view mode from the details view to the list mode or overview mode, click the corresponding icon on the toolbar.

See also

Library basics (Page 7)

Overview of the "Libraries" task card (Page 11)

Comparing library elements (Page 113)

Using global libraries (Page 29)

Using the library view 4

4.1 Overview of the library view

Function of the library view

The library view combines the functionality of the "Libraries" task card and the overview window. In the library view, the elements of a library are displayed in various views. In the details view, for example, you see additional properties of the individual elements. You can also edit and version the types in the library view.

Layout of the library view

The following figure shows the components of the library view:

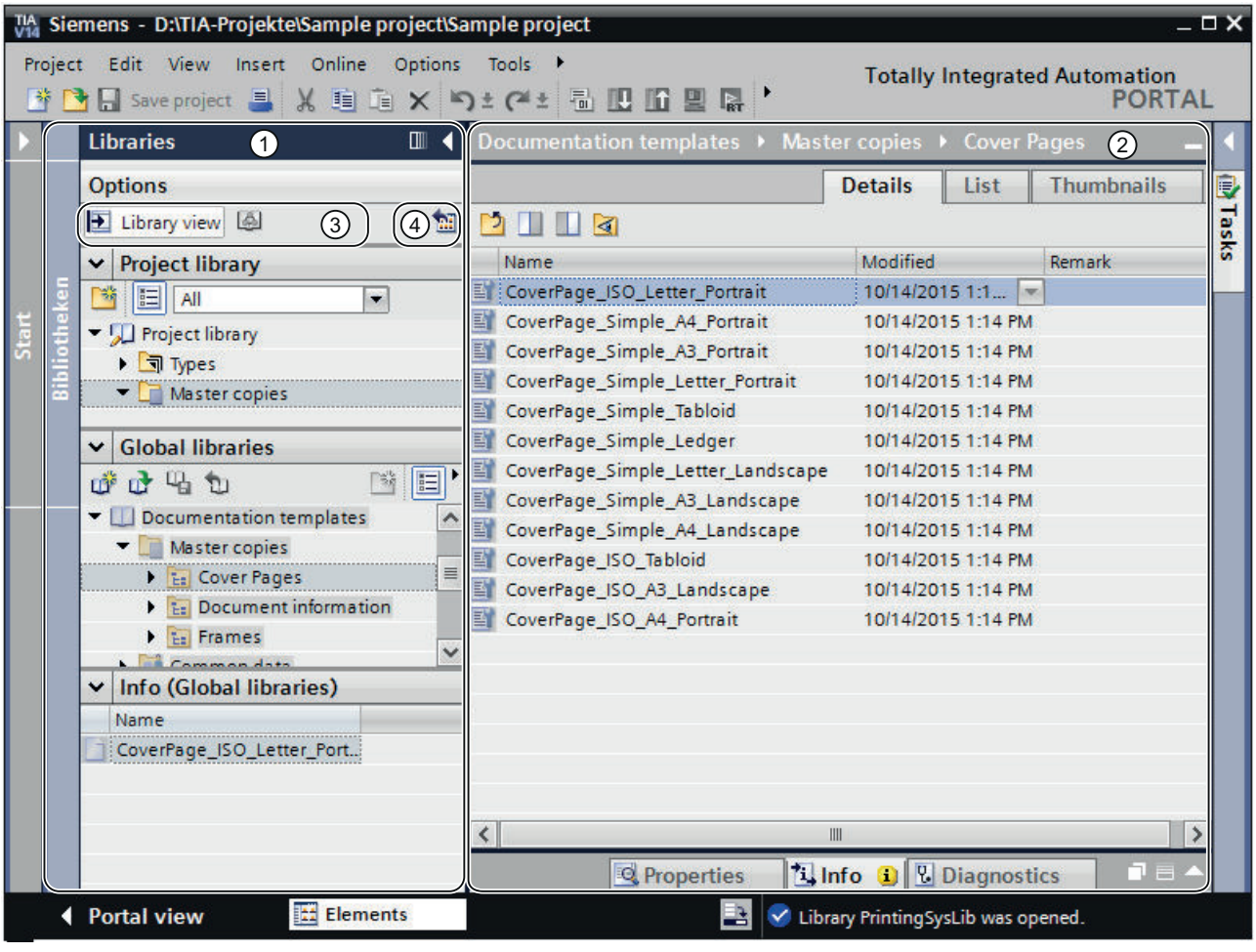

① Library tree
② Library overview
③ "Open/close the Library view" button and "Open library management"
④ "Open or close library overview" button

Library tree

The library tree is similar to the "Libraries" task card, apart from a few minor differences. In contrast to the task card, there is no "Elements" palette, because the elements are displayed in the library overview. In addition, you can close the library view in the library tree, or open and close the library overview.

See also: "Libraries" task card (Page 11)

Library overview

The library overview corresponds to the overview window and displays the elements of the currently selected object in the library tree. You can display the elements in three different views:

- Details view
  The objects are displayed in a list with additional information, such as the date of the last change.
- List view
  The objects are displayed in a simple list.
- Thumbnails view
  The objects are displayed as icons according to category.

In addition, you can perform the following actions in the library overview:

- Renaming elements
- Deleting elements
- Copying elements
- Moving elements
- Editing type instances
- Versioning types
- WinCC only: Editing faceplates and HMI user data types

See also: Overview window

See also

Basics on master copies (Page 47)

Basics on types (Page 53)

Opening and closing the library view (Page 18)

Library basics (Page 7)

Comparing library elements (Page 113)

4.2 Opening and closing the library view

The library view is opened automatically in some cases, for example, when you edit the test instance of a type or when you edit faceplates and HMI user data types. You can also open the library view manually.

Opening the library view

To open the library view manually, follow these steps:

1. Open the "Libraries" task card.
2. Click the "Open/close library view" button in the "Libraries" task card.
   The library tree opens. The "Library" task card and the project tree are closed.
3. If the library overview is not displayed, click the "Maximizes/minimizes the elements" button in the library tree.
   The library overview opens.

Exit library view

To exit the library view, follow these steps:

1. Click the "Close/open library view" button in the library tree.
   The library tree closes. The "Libraries" task card and the project tree are opened.

See also

Overview of the library view (Page 15)

Library basics (Page 7)

Comparing library elements (Page 113)

Using the "Libraries" task card (Page 11)

Using global libraries (Page 29)

Using library management 5

5.1 Overview of the library management

Function of the library management

Master copies and types with dependencies to other library elements are subject to some functional restrictions. They cannot be deleted, for example, as long as dependencies still exist. This prevents other library elements from becoming useless. The library management is used to identify the dependencies and to create an overview of the work progress.

The library management offers the following functions:

- Display of the correlations of types and master copies
  If a type is referenced in other types or master copies, the correlations are displayed in the library management. You will also be able to see which library elements reference a type or a master copy.
- Display of points of use of types in the project
- Upgrading types
- Using filters to narrow down the displayed types

Layout of the library management

The following figure shows the components of the library management:

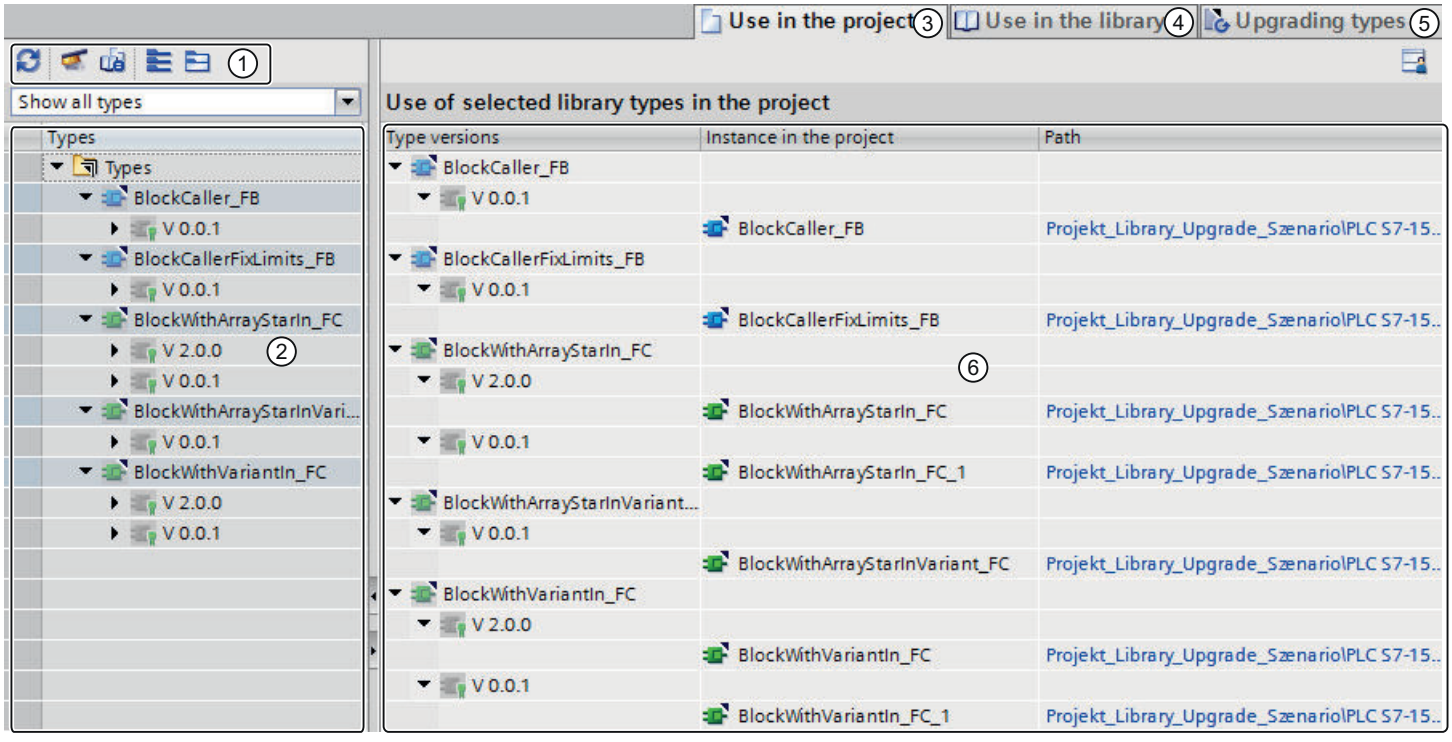

① Toolbar of the library management
② "Types" area
③ "Use in the project" tab
④ "Use in the library" tab
⑤ "Upgrading types"
⑥ "Uses" window

Toolbar of the library management

You can perform the following tasks in the toolbar of the library management:

- Update uses
  If the project was changed, you can update the view of the library management.
- Clean up library
  By cleaning up a library, you delete all types and type versions that are not linked to an instance in the project.
- Harmonize project
  By harmonizing a project, you adapt the names and the path structures of type uses in the project to the corresponding names and path structures of the types within a library.
- Collapse all
  Hides all subentries of the uppermost node in the "Types" area. The lower-level elements, such as types and individual type versions, are no longer displayed.
- Expand all
  Expands all lower-level elements in the "Types" area. The lower-level elements, such as types and type versions, are displayed in full.

"Types" area

The "Types" area displays the contents of the folder that you have selected in the library view. For each type, the types that it references are displayed. You can expand or collapse all type nodes by using the buttons in the toolbar of the "Types" area.

You can also filter the view with the "Filter" drop-down list. You will find additional information on the specific filter settings here: Filtering the display of types (Page 24)

"Use in the project" tab

The "Use in the project" tab is used to show the instances of type versions and their respective point of use in the project. When you select an instance, you can show the cross references of the instance in the project in the Inspector window.

You can find additional information here:

Displaying cross references of an instance (Page 26)

Displaying instances in the project (Page 25)

"Use in the library" tab

The "Use in the library" tab is used to show all points within the library at which a type or a master copy is used.

You can find additional information here: Displaying relations to other library objects (Page 27)

"Upgrading types"

In the "Upgrading types" tab, you can identify all types, reference changed types, and upgrade.

You can find more information on upgrading projects here:

Upgrading types in an existing project (Page 91)
Example of the updating and upgrading of types (Page 94)

See also

Opening library management (Page 23)
Library basics (Page 7)

5.2 Opening library management

Procedure

To open the library management, follow these steps:

1. Open the library view.
2. Select a type or any folder that contains types.
3. Select the "Library management" command from the shortcut menu.

Or:

1. Click the "Types" folder in the project library.
2. Select the "Library management" command from the shortcut menu.

Result

The library management opens and the types are displayed with their versions.

See also

Overview of the library management (Page 19)

5.3 Filtering the display of types

You can use filters to obtain a better overview of types in extensive libraries. A filter provides you with the option of limiting the displayed types. The following filters are available:

- Types with pending changes
  The filter displays all types that have a version "In test" or "In work".
- Released types
  The filter displays all released types that do not have versions "In test" or "In work".
- Types with multiple versions
  The filter displays all types that have more than one version.
- Types not used in the project
  The filter displays all types which have no instance in the project.
- Types with new versions for upgrading other types
  The filter displays all types that are relevant for upgrading other types. The types must also meet the following criteria:
  - The types have more than one version.
  - All versions of the respective type have been released.

Requirement

The library management is open.

Procedure

To filter the displayed types, follow these steps:

1. In the "Types" area, select the folder whose contents you want to filter.
2. Select the required filter from the drop-down list.
   In the "Types" area, types are displayed that correspond to the selected filter criteria.

See also

Overview of the library management (Page 19)

State of type versions (Page 56)

Creating a test version of a type (Page 67)

Editing test version of a type (Page 68)

Creating an editing version of a type (Page 69)

5.4 Displaying instances in the project

In the library management, you can have the instances of all versions of a type or an individual type version displayed. You can jump directly to each instance in the project.

Requirement

The library management is open.

Procedure

To display the instances of a type or its versions, follow these steps:

1. In the "Types" area select the required folders, types or versions.
2. Open the "Use in the project" tab.
   The instances in the project are displayed for each type version. The "Path" column shows the path at which the respective instance is located in the project.
3. Click the path to jump directly to the respective instance in the project tree.
   The library management is hidden and the instance is selected in the project tree.

See also

Displaying cross references of an instance (Page 26)

Using types of the project library (Page 64)

Overview of the library management (Page 19)

Displaying relations to other library objects (Page 27)

5.5 Displaying cross references of an instance

You can display the cross references of an instance without exiting the library management.

Requirement

The library management is open.

Procedure

To display the cross references of an instance in the project, follow these steps:

1. In the "Types" area, select a type version whose instances you want to display.
2. Select the instance of the required type version in the "Uses in the project" tab.
3. Open the "Info > Cross-references" tab in the inspector window.
   The cross reference of the instance are displayed in the project.

See also

Overview of the library management (Page 19)

Displaying instances in the project (Page 25)

5.6 Displaying relations to other library objects

You can display relations between individual library objects in the library management. The references of the individual type versions to the other library objects are automatically displayed in the "Types" area. In the "Uses in the library" tab, you can also view the other library objects in which the respective type version is referenced.

Requirement

The library management is open.

Procedure

To display the other library objects from which a type version is referenced, follow these steps:

1. In the "Types" area select the required folders, types or versions.
2. Open the "Uses in the library" tab.
   In the "Uses" window, you can now see which other library objects are referenced by the individual type versions.
3. To jump to the referenced library object, click on the respective path in the "Path" column.

See also

Displaying instances in the project (Page 25)

Overview of the library management (Page 19)

Referencing of objects within a block (Page 62)

Using global libraries 6

6.1 Creating a global library

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To create a new global library, follow these steps:

1. Click the "Create new global library" icon in the toolbar of the "Global libraries" palette or select the command "Global libraries > Create new library" in the "Options" menu.
   The "Create new global library" dialog opens.
2. Specify the name and the storage location for the new global library.
3. Confirm your entries with "Create".

Result

The new global library is generated and pasted into the "Global libraries" palette. A folder with the name of the global library is created in the file system at the storage location of the global library. The actual library file is given the file name extension ".al15_1".

See also

Library basics (Page 7)

Opening a global library (Page 32)

Displaying properties of global libraries (Page 34)

Saving a global library (Page 36)

Closing a global library (Page 37)

Deleting a global library (Page 38)

6.2 Create write-protected global library

You can protect global libraries by exporting them as protected libraries. These libraries are then provided with a write protection. This means the types contained in them and the global libraries are also then protected against changes. Write protection of protected global libraries cannot be revoked.

Requirements

- The "Libraries" task card is displayed or the library view is open.
- A global library that is not write-protected exists.

Procedure

Follow these steps to create a write-protected global library:

1. Select the global library from which you want to create a write-protected global library.
2. Right-click to call up the shortcut menu.
3. Select the command "Export as protected library" in the shortcut menu of the global library. The "Export as a protected library" dialog is opened.
4. Specify the name and the storage location for the write-protected global library.
5. Confirm your entries with "Export".

Result

The write-protected global library is generated and pasted into the "Global libraries" palette. A folder with the name you assigned to the protected global library is created at the location of the global library in the file system. The actual library file is given the file name extension ".al15_1".

The types and the master copies from the original global library are copied as write-protected types and master copies to the new global library.

Note

Preventing changes in write-protected types

Write-protected types can possibly be changed when they are overwritten by more up-to-date, non-write-protected versions of the same type.

Therefore, if you want to supply write-protected types to customers, create write-protected libraries with types in a newer version if possible before supplying them.

This ensures that the customer does not already use write-protected versions of this type and cannot make changes.

6.3 Compatibility of global libraries

Libraries from the TIA Portal V13 or earlier

Global libraries that were created with TIA Portal V13 or earlier cannot be used in the current version of the TIA Portal. First, upgrade the libraries with version V13 SP1 of the TIA Portal to the current library version. You can then open the library and upgrade once again.

Libraries from the TIA Portal V13 SP1 or later

You can use the TIA Portal to open global libraries that were created with TIA Portal V13 SP1 or higher. When you open the library, you are prompted to upgrade the library to the current version of the TIA Portal. Once the global library is upgraded, it cannot be opened with earlier versions of the TIA Portal.

Backward compatibility of the current library version

Global libraries saved with the library format of the current TIA Portal product version are not backward compatible with older versions due to their enhanced functionality. Global libraries in the current library format can only be used in connection with TIA Portal V15.1 projects.

See also

Opening a global library (Page 32)

6.4 Opening a global library

Global libraries can be further developed centrally and used over several projects. Several persons can open a global library simultaneously from a central storage location, provided all users open the global library in write-protected mode.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To open a global library, follow these steps:

1. Click the "Open global library" icon in the toolbar of the "Global libraries" pane or select the command "Global libraries > Open library" in the "Options" menu.
   The "Open global library" dialog box opens.
2. Select the global library you want to open. Library files are identified by the file name extension ".al[version number]". This means that global libraries that were saved with the current TIA Portal product version have the file name extension ".al15_1".
3. Write protection is activated for the library. If you want to modify the global library, clear the option "Open as read-only".
4. Click "Open".
   If the library is saved in the library version for TIA Portal V15.1, the "Global library" is opened and pasted into the "Global libraries" pane. The "Upgrade global library" dialog opens if you have selected a global library from TIA Portal V13 SP1 or later. Upgrade the library to the current version of the TIA Portal in this case.

Result

The global library is displayed in the "Global libraries" pane.

If the global library was opened write-protected, it and all its subfolders are grayed out.

See also

Retrieving global libraries (Page 40)

Compatibility of global libraries (Page 31)

Library basics (Page 7)

Creating a global library (Page 29)

Displaying properties of global libraries (Page 34)

Saving a global library (Page 36)

Closing a global library (Page 37)

Deleting a global library (Page 38)

6.5 Defining library languages

You can define various library languages for global libraries and select an editing language for all comments from them. The selection of the library languages is valid for all library elements (types, type versions and master copies) within the global library.

The comments of the types and their type versions can be translated into multiple library languages and managed in the global library. The selected editing language specifies the languages in which the comments are displayed. The comments of a type can be edited at any time. As soon as the version of a type is released, it comments can no longer be edited and are displayed in ready-only mode.

If you change the editing language of a global library, the comments of all types are displayed in the newly selected editing language, provided translations are available.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To define the library languages of a global library, follow these steps:

1. Click the "Open global library" icon in the toolbar of the "Global libraries" pane or select the command "Global libraries > Open library" in the "Options" menu.
   The "Open global library" dialog box opens.
2. Select the global library you want to open.
3. Disable the write protection of the global library by clearing the option "Open as read-only". If you do not clear the "Open as read-only" option, you can see the selection of library languages in the global library, but you cannot change them.
4. Click "Open".
   The global library is displayed.
5. Open the "Languages & Resources" folder.
6. Double-click the "Library languages" entry or select "Open" from the shortcut menu.

Result

The library languages are displayed in the Library view.

6.6 Displaying properties of global libraries

Global libraries contain properties for describing the respective library in more detail. Properties include the following:

- General information about the library
  This includes the following information: creation time, author, file path, file size, copyright, etc. Many of the attributes can be changed.
- Library history
  The library history contains an overview of the migrations performed. Here you can also call the log file for the migrations. The library history also contains information on updates of the global library.
- Support packages in the library
  You can display an overview of additional software. The additional software is needed to process all devices of the project.
- Software products in the library
  You can display an overview of all installed software products needed for the project.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To display the properties of a global library, follow these steps:

1. Right-click the global library whose properties you want to display.
2. Select the "Properties" command in the shortcut menu.
   A dialog containing the properties of the global libraries opens.
3. Select the properties in the area navigation that you want to have displayed.

See also

Opening a global library (Page 32)

Library basics (Page 7)

Creating a global library (Page 29)

Saving a global library (Page 36)

Closing a global library (Page 37)

Deleting a global library (Page 38)

6.7 Displaying logs of global libraries

Logs are created when you update or clean up global libraries or assign a shared version to several types. The logs list all changes you have made to the global library. The logs are created in ZIP-files and are stored together with the global library. They are always available once you have opened the global library.

Procedure

To open the logs of a global library, follow these steps:

1. Open the global library in the "Libraries" task card or in the library view.
2. Open "Common data > Logs" in the lower-level folder.
3. Double-click the required log.
   The log opens in the work area.

See also

Updating library with the types of another library (Page 89)

6.8 Saving a global library

After you have changed a global library, you need to save it. You can save a global library under another name using the "Save library as" command.

Note

Backward compatibility with older versions of the TIA Portal

Note that global libraries can no longer be opened in older versions of the TIA Portal once they have been saved in the current version.

Requirement

The "Libraries" task card is displayed or the library view is open.

Saving changes

To save a global library, follow these steps:

1. Right-click on the global library you want to save.
2. Select the "Save library" command in the shortcut menu.

Saving a global library under another name

To save a global library under another name, follow these steps:

1. Right-click the global library that you want to save under a different name.
2. Select the "Save library as" command in the shortcut menu.
   The "Save global library as" dialog opens.
3. Select the storage location and enter the file name.
4. Confirm your entries with "Save".
   The library is saved in the specified location under the new name. The original library is retained.

See also

Working with archives of global libraries (Page 39)

Archiving global libraries (Page 39)

Library basics (Page 7)

Creating a global library (Page 29)

Opening a global library (Page 32)

Displaying properties of global libraries (Page 34)

Closing a global library (Page 37)

Deleting a global library (Page 38)

6.9 Closing a global library

Global libraries are independent of projects. This means that global libraries are not closed together with your project. You must therefore close global libraries explicitly.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To close a global library, follow these steps:

1. Right-click on the global library you want to close.
2. Select the "Close library" command in the shortcut menu.
   If you have made changes to the global library, select whether or not you want to save the changes.
   The global library is closed.

See also

Creating a global library (Page 29)

Opening a global library (Page 32)

Displaying properties of global libraries (Page 34)

Saving a global library (Page 36)

Library basics (Page 7)

Deleting a global library (Page 38)

6.10 Deleting a global library

If you no longer require a global library, you can delete it. Libraries supplied by Siemens cannot be deleted.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To delete a global library, follow these steps:

1. Right-click the global library you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. Click "Yes" to confirm.

Result

The global library is removed from the "Global libraries" palette. The entire library for the global library is deleted from the file system.

See also

Library basics (Page 7)

Creating a global library (Page 29)

Opening a global library (Page 32)

Displaying properties of global libraries (Page 34)

Saving a global library (Page 36)

Closing a global library (Page 37)

6.11 Archiving and retrieving global libraries

6.11.1 Working with archives of global libraries

If you want to back up global libraries on an external hard drive or send them via e-mail, for example, use the archiving function to reduce the storage space of the library.

Options for reducing the size of the project

There are two ways to reduce the space required to store global libraries:

- Creating a compressed archive of global libraries
  Archives of global libraries are compressed files, each containing an entire global library, including the entire folder structure of the library. Before the directory with the global library is compressed into the archive file, all files are reduced to their essential components to further decrease the storage space. Compressed archives of global libraries are therefore well suited for sending via e-mail.
  Compressed archives of global libraries of the current product version have the file extension ".zal15_1".
  To open a compressed archive of a global library, retrieve the archive. The archive file is extracted to a location you have selected with the entire folder structure and all files.
- Minimizing global libraries
  You can skip additional compression in an archive file, and instead create a copy of the global library directory. The included files can be reduced to the essential elements. This minimizes the required storage space. The full functionality of the global library is retained and the global library can be loaded as usual.
  Minimized global libraries are especially well suited for archiving, for example, on an external medium.

See also

Archiving global libraries (Page 39)

Retrieving global libraries (Page 40)

6.11.2 Archiving global libraries

Global libraries can be archived as compressed or uncompressed files. The global libraries to be archived do not have to be open in the TIA Portal.

The memory requirements of global libraries can be reduced by archiving the global libraries in compressed files.

Procedure

To archive a global library, follow these steps:

1. Select the "Archive ..." command from the "Project" menu.
   Alternatively, select the "Archive library" command from the shortcut menu of the global library you want to archive.
   The "Archive" dialog opens.
2. Select a library file with the extension".al15_1" in the "Source path" field.
3. To create a compressed archive file, select the option "Archive as compressed file".
4. If you do not want to archive restorable data, select the "Discard restorable data" option.
5. To add date and time automatically, select the option "Add date and time to the file name".
6. In the "Destination path" box, select the location where you want to save the archive file or the new directory of the global library.
   The directory may not be located in a project directory or within the directory of a global library.
   You can set the default directory under "Options > Settings > General > Archive settings > Storage location for library archives".
7. Click "Archive".

Result

An archive is created for the global library. Compressed global libraries get the file extension ".zal15_1". The file includes the complete directory of the global library. The individual files of the global library are also reduced to the essential components in order to save space.

For uncompressed global libraries, only a copy of the original directory of the global library is created in the desired location.

See also

Working with archives of global libraries (Page 39)

Retrieving global libraries (Page 40)

6.11.3 Retrieving global libraries

Before you can use an archived global library, you have to retrieve it. The global library is extracted and then opened in the TIA Portal.

Procedure

To extract the archive of a global library, follow these steps:

1. Select the "Global libraries > Retrieve library" command in the "Options" menu.
   The "Retrieve archived global library" dialog opens.
2. Select the archive file.

3. Select the check box "Open read-only" if you want to load the global library write-protected.
4. Click "Open".
5. The "Find folder" dialog opens.
6. Select the target directory to which the archived global library should be extracted.
7. Click "OK".

Result

The global library is extracted to the selected directory and opened immediately.

See also

Working with archives of global libraries (Page 39)
Archiving global libraries (Page 39)
Opening a global library (Page 32)
Compatibility of global libraries (Page 31)

6.12 Using global corporate libraries

6.12.1 Basics of corporate libraries

Introduction

Corporate libraries are global libraries made available by an administrator and assigned to the TIA Portal. The administrator can assign new libraries or change libraries at any time. New libraries are automatically loaded in the TIA Portal following your confirmation. As soon as more recent versions of corporate libraries are available, the existing corporate libraries are automatically updated to the more recent version, also following your confirmation.

The corporate libraries are located just like normal global libraries in the "Global libraries" palette of the "Libraries" task card.

Providing corporate libraries

Corporate libraries can be saved in any directory on the hard drive of the computer or on a network drive. To include corporate libraries in the TIA Portal, use an XML file. The XML file includes the directories and names of the assigned corporate libraries. The XML file must be saved in the following directory on the hard drive of the computer:

`C:\ProgramData\Siemens\Automation\Portal V15_1\CorporateSettings\`

The XML file must be named "`CorporateSettings.xml`".

You either copy the configuration file yourself to the corresponding directory or the configuration file is assigned to you through the company network. The valid configuration is automatically applied when the TIA Portal is started. When the TIA Portal is started, it continuously monitors the directory for configuration files. If the configuration file has changed, you receive a prompt to apply the new configuration. You can reject this prompt twice. You will always receive the next prompt three hours later. You must apply the new configuration at the third prompt. You receive a changed configuration file from the project administrator, for example, if corporate libraries have been added or deleted.

If you access company libraries stored on a network drive, you only need network access to update them. The company libraries that are stored locally are automatically loaded after a restart. If you do not have network access, you can only use locally stored copies of your company libraries.

Options as project administrator

You can automatically assign the configuration file or the corporate libraries to the computers of the team members or distribute updates to the team members. This function is not part of the TIA Portal and requires a corresponding IT infrastructure in your company. If you want to administer the configuration file centrally, discuss this approach with the IT managers at your company.

The archive function for global libraries can be used to pass corporate libraries to employees of the project team. However, you must retrieve corporate libraries prior to their use in the TIA Portal.

The following figure shows an example of how you can distribute corporate libraries to individual employees within your organization:

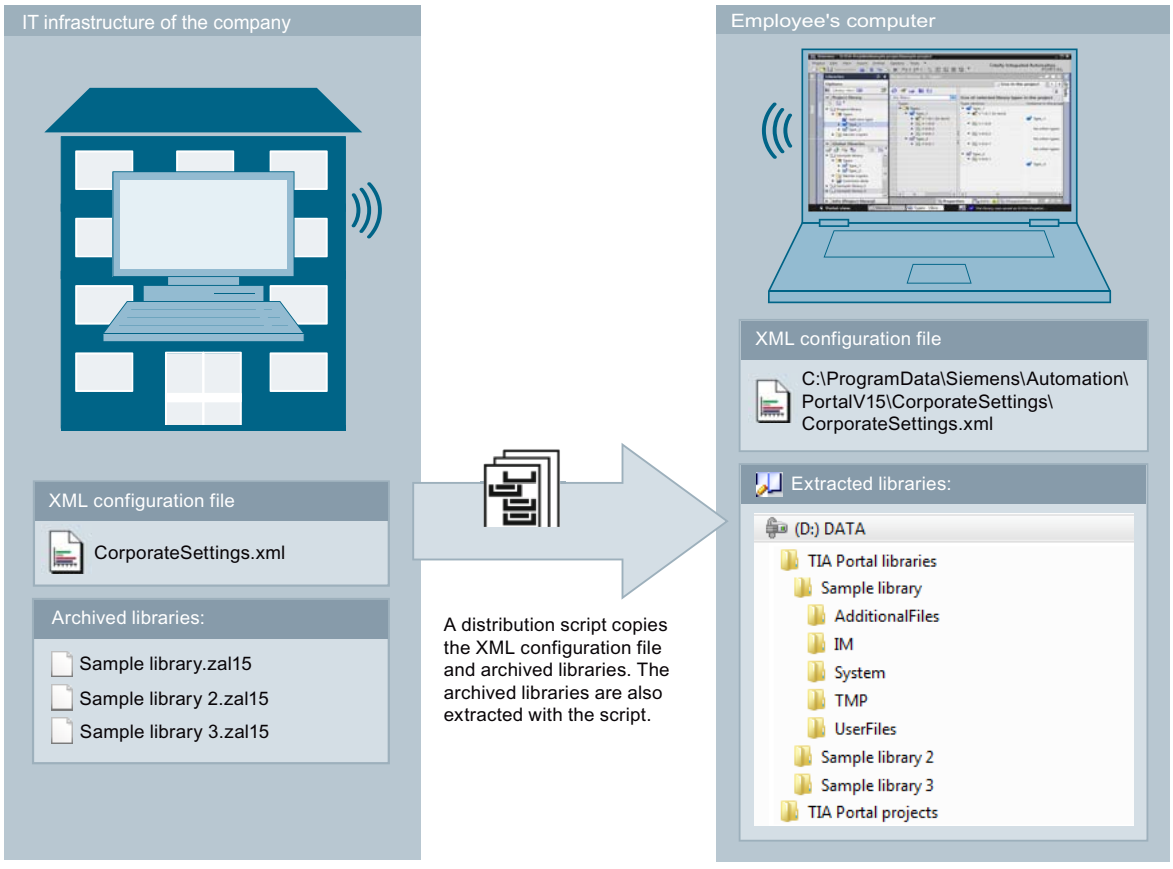

See also

Working with archives of global libraries (Page 39)

Archiving global libraries (Page 39)

Retrieving global libraries (Page 40)

6.12.2 Creating a configuration file for corporate libraries

You use a configuration file in XML format to make the corporate libraries available in the TIA Portal. The configuration file includes the directories and file names of the libraries to be loaded. The XML file is the same file that you use for user-defined documentation settings.

Below you will learn how you create the XML configuration file and where to save it.

Procedure

To provide a configuration file for corporate libraries, follow these steps:

1. Create an XML configuration file with the content listed below. Use the coding "UTF-8".
2. Replace the paths mentioned in the example in the "<CorporateLibraryPaths>" section below with the paths where your libraries are stored.
3. Save the XML file under the name "CorporateSettings.xml".
4. Save the file in the following directory on your computer:
   C:\ProgramData\Siemens\Automation\Portal V15_1\CorporateSettings\

Content of the XML configuration file

The XML configuration file must have the following content:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
 <Settings.Settings ID="0">
  <ObjectList>
   <Settings.General ID="1" AggregationName="General">
    <!-- Here you find the settings for global company libraries, if available. -->
    <ObjectList>

<!-- Section for corporate libraries -->
     <Settings.General ID="1" AggregationName="General">
     <AttributeList>
      <CorporateLibraryPaths>
      <!-- Example for two entries -->
      <Item>F:\CorporateLibraries\Corporate_Library_1\Corporate_Library_1.al15_1</
Item>          <Item>F:\CorporateLibraries\Corporate_Library_2\CorporateLibraries.al15_1</
Item>
      <!-- Here you enter additional global libraries, if any. -->           </
CorporateLibraryPaths>
    </AttributeList>
   </Settings.General>
<!-- Section for corporate libraries ends -->

<!-- Section for user-defined documentation -->
     <Settings.UserDocumentation ID="2" AggregationeName="UserDocumentation">
      <!-- In the following section, you specify the values for display of the user-defined
documentation. -->
      <AttributeList>
       <!-- Activates or deactivates the display of the access log. -->
       <DisplayLogInformation>
        <Value>true</Value>
        </DisplayLogInformation>
      <!-- Activates or deactivates the search for user-defined documentation in a central
directory. -->
       <EnableLookupFromCentralStorageLocation>
         <Value>true</Value>
       </EnableLookupFromCentralStorageLocation>
       <!-- Specifies the central directory for user-defined documentation. -->
       <CentralStorageLocation>
        <Value>D:\CorporateDocumentation\UserDocumentation\</Value>
       </CentralStorageLocation>
      </AttributeList>
     </Settings.UserDocumentation>
<!-- Section for user-defined documentation ends -->

    </ObjectList>
   </Settings.General>
  </ObjectList>
 </Settings.Settings>
</Document>
```

Result

As soon as you have saved the XML configuration file in the respective directory, you will receive a prompt in the TIA Portal to load the corporate libraries.

Using master copies 7

7.1 Basics on master copies

Master copies are used to create standardized copies of frequently used elements. You can create as many elements as needed and insert them into the project based on a master copy. The elements inherit the properties of the master copy.

You store master copies either in the project library or in a global library. Master copies in the project library can only be used within the project. When you create the master copy in a global library, it can be used in different projects.

The following elements can be created as master copies in the library, for example:

- Devices with their device configuration
- Tag tables or individual tags
- Instruction profiles
- Watch tables
- Elements from the documentation settings, for example, cover sheets and frames
- Blocks and groups containing multiple blocks
- PLC data types (UDTs) and groups which contain multiple PLC data types
- Text lists
- Alarm classes
- Technology objects

In many cases, the objects you add as master copy contain additional elements. A CPU, for example, can contain blocks. If the included elements are uses of a type version, the used versions of the types are automatically created in the library. The elements contained therein are then used as an instance and linked to the type.

See also

Adding master copies (Page 48)

Using master copies (Page 51)

Basics on types (Page 53)

Filtering master copies (Page 50)

7.2 Adding master copies

If you want to use objects multiple times, copy them as master copies in the project library or in a global library. You can choose from the following methods for creating master copies:

- You select one or more elements and create individual master copies from them
- You select multiple elements and create a single master copy from them that contains all selected elements.

Requirement

- The "Libraries" task card is displayed.
- If you add a device as a master copy, this device meets the following requirements:
  - The device is compiled and consistent.
  - The device contains no test instance of a type.
- If you add the master copy to a global library, the global library is opened with write permission.

Creating master copies from one or more elements

To create a master copy for one or more elements, follow these steps:

1. Open the library in the "Libraries" task card.
2. Select the desired elements.
3. Using a drag-and-drop operation, move the elements to the "Master copies" folder or any subfolder of "Master copies".

Alternative:

1. Select the desired elements.
2. Copy the elements to the clipboard and paste the elements at the required location.
   Each element is pasted in the library as a master copy. A type is created automatically in each case from any objects contained (such as referenced blocks).

Creating a single master copy from multiple elements

To create a single master copy for all elements from multiple elements, follow these steps:

1. Open the library in the "Libraries" task card.
2. Copy to the clipboard the elements that you want to create as master copies.
3. Right-click on the "Master copies" folder or any of its subfolders in the library.
4. In the shortcut menu, select "Paste as a single master copy" command.

Alternative:

1. Select the desired elements.
2. Using a drag-and-drop operation, move the elements to the "Master copies" folder or any subfolder of "Master copies". Meanwhile, keep the <Alt> key pressed.
   The elements are pasted in the library as a single master copy. The single master copy contains all selected elements. A type is created automatically in each case from any objects contained (such as referenced blocks).

Note

Avoiding complex structures of master copies

To prevent name conflicts and conflicts regarding the folder structure during subsequent use of master copies, avoid complex master copies. Complex master copies are, for example, master copies that consist of multiple elements and nested folders.

See also

Basics on master copies (Page 47)

Using master copies (Page 51)

Library basics (Page 7)

Adding types to the project library (Page 60)

7.3 Filtering master copies

To make it easier to keep track of a large number of master copies, you can filter the display according to the type of master copy.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To filter the view, follow these steps:

1. Open the "Master copies" folder in the project library or a global library.
2. In the drop-down list of the toolbar, select the type of objects you want to display under "Master copies".

Result

Only the selected type of master copies is displayed. You can set the filter to "All" to return to an unfiltered view.

See also

Library basics (Page 7)

Creating folders in a library (Page 107)

Basics on master copies (Page 47)

Using master copies (Page 51)

Using the element view (Page 14)

7.4 Using master copies

Master copies are contained either in the project library or in a global library. You can paste one or more master copies in the project at the same time. If you insert multiple master copies at the same time, ensure that all master copies are compatible with the desired point of use.

Requirement

The "Libraries" task card is displayed.

Procedure

To paste a master copy into the project, follow these steps:

1. In a library, open the "Master copies" folder or any subfolder of "Master copies".
2. Using a drag-and-drop operation, move the desired master copies or complete folders to the point of use.

Or:

1. Open the element view.
2. Using a drag-and-drop operation, move the desired master copies or whole folders from the "Elements" pane to the point of use.

Result

A copy of the master copies is inserted. If incompatible master copies were included in a multiple selection, these are omitted and a copy is not created in the project.

See also

Basics on master copies (Page 47)

Adding master copies (Page 48)

Filtering master copies (Page 50)

Library basics (Page 7)

Using the element view (Page 14)

Using types and their versions 8

8.1 Basics on types

Using types

Types are elements that are required for the execution of user programs. Types can be versioned and further developed from a central location.

The following elements can be stored as type in the project library or the global library:

- Functions (FCs)
- Function blocks (FBs)
- PLC data types (UDTs)
- Faceplates
- Screens
- Styles
- User-defined functions

You can use the same version of a type only once within a CPU. If various versions of a type are available, these can also be used within a CPU. Depending on the number of CPUs used, any number of instances can be derived within a project. The instances are then connected with the version of the type in the library. If you are using types from a global library, the type is also created in the project library. If the type already exists in the project library, missing type versions are added if necessary. The instance is then linked only to the respective type version in the project library.

Types and their instances are marked with a black triangle. The following figure shows an instance, marked with a black triangle, and an ordinary program block:

Basics on versioning of types

Type versioning provides you with the means to develop types centrally and then roll out the most recent version to the individual projects as an update. In this way, error corrections and added functions can be easily integrated into existing projects. If you have created a new version of a global library, you can update existing projects in an automatic process. This minimizes errors and reduces the amount of maintenance work for large automation solutions that contain numerous individual projects.

Versioning allows you to trace the development process of individual types. Before you release a version, you can try it out in a test environment to determine whether changes made to a type integrate smoothly into an existing project. You only release a version for productive use

after you have made sure that everything operates without errors. You can view the history of individual instances in the project at any time and determine the version from which the instance was derived.

The TIA Portal also checks automatically whether there are associated objects with individual versions of a type. Associated objects can be, for example, PLC data types (UDTs) or other blocks referenced in a block. All associated objects are already taken into consideration during the creation of a type or during copying between libraries. Before being released, versions of types are checked for their consistency to ensure that no inconsistencies arise in the project.

Versions of types

Versions are assigned to each type. The version number is displayed in both the "Libraries" task card as well as the library view next to the respective type. You can also display the version of a type in the project tree next to the instances. To do this, select the "Version" column in the project tree. This allows you to see at all times which version of an instance is used in the project.

The version number consists of three numbers separated by periods. You can randomly assign the first two digits. Numbers from 1 to 999 are permitted as the first two numbers. The third digit is the build number. It is automatically incremented by one when you edit an instance related to the version. The build number is reset to 1 when you release the version.

Note

Library setting: "Disable editing of the type version number"

You can use the library setting "Disable editing of the type version number" to prevent that the type version number can be assigned by the user for the actions "Release version" and "Assign version". By activating the library setting, the type version number can not be changed by the user. The build number is still incremented automatically when a version is released or assigned. When the high limit of the build number in a block (999) has been reached, the version number is incremented in the next block.

The versions of types can have three states:

- In work (faceplates and HMI user data types)
- In test (all sorts of types except faceplates and HMI user data types)
- Released

The following figure shows a type with two versions. One version is "in test" and one version is released:

If you use functions (FCs) or function blocks (FBs) as types in the project library, observe the following when compiling the block if you use different instruction versions:

- You can recompile know-how-protected blocks without password if the interfaces of the block versions are identical.
- You can recompile blocks that are not know-how-protected if the supply is not lost for any actual parameters and if no new actual parameters that you need to supply are added by the new version.
- If it is not possible to compile the type with a different instruction version, a new version of the type is required that contains the required adjustments.

See also

State of type versions (Page 56)

Basics on master copies (Page 47)

Adding types to the project library (Page 60)

Using types of the project library (Page 64)

Editing library elements (Page 109)

Duplicating types (Page 63)

8.2 State of type versions

The versions of types can be in three different states. The states can be determined from the instance or in the library.

"In work" status

Only versions of faceplates and HMI user data types have the "In work" status. If a version is in progress, "In work" appears next to the version in the library.

When you create a new type or a new version of a released type, the type is assigned the status "In work".

Types with the status "In work" can be edited in the library view. A reference to an instance in the project does not have to exist. Upon release, the compatibility of the type is tested by a consistency check.

"In test" status

All types except for faceplates and HMI user data types can have the "in test" status. If a version is in test, "in test" appears next to the instance and in the library. A version in the test state is linked to a test instance in the project. This allows you try out the effects of your changes in a test environment including all online functions before you release a type for use in actual operations.

"Released" status

The "released" status is available for all types, regardless of the point of use. If a version has been released, the symbol of the version is marked with a seal in the library:

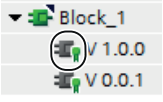

Released versions can be opened with write protection in their instance. If you want to edit a released version, you must first create a new "In work" or "In test" version.

See also

Basics on types (Page 53)

Using types of the project library (Page 64)

Creating a test version of a type (Page 67)

Editing test version of a type (Page 68)

Creating an editing version of a type (Page 69)

Performing a consistency check for a version (Page 69)

Discarding versions (Page 70)

Release versions (Page 71)

Assigning a version (Page 82)

Updating the project to the latest versions (Page 73)
Remove the link between instance and type (Page 75)
Filtering the display of types (Page 24)

8.3 Displaying a released type version

If you display a released version but do not want to edit it, open the instance with write protection. All types except faceplates and HMI user data types can be opened directly at the instance. Faceplates and HMI user data types can only be opened in the "Libraries" task card or in the library view.

Requirement

The released version has an instance in the project, except the type is a faceplate or an HMI user data type.

Opening a type version at an instance

To open a released version of a type with write protection starting from an instance, follow these steps:

1. Select the released version at the instance in the project tree.
2. To do this, right-click the instance and select the "Open" command from the shortcut menu. The instance is opened with write protection.

Opening a type version in the "Libraries" task card or library view

To open a released version of a type in the "Libraries" task card or in the library view, follow these steps:

1. Select the version.
2. To do this, right-click the version and select the "Open" command from the shortcut menu. If it is a faceplate or HMI user data type, it is opened directly in the library view. In this case, skip the remaining steps. If it is any other type, the "Open type" dialog opens.
3. Select the instance with the version that you want to display from the list of instances.
4. Click "OK" to confirm.
   The instance is opened with write protection.

8.4 Displaying properties of a type or version

You can display the properties of a type or version.

Procedure

To display the properties of a type or a version and to enter a comment, follow these steps:

1. Select a type or the version of a type in the "Libraries" task card or in the library view.
2. To do this, right-click on the type or one of its versions and select the "Properties" command from the shortcut menu.
   The "Properties" dialog box opens.
3. If needed, enter a comment on the type in the "Comment" field or edit an existing comment.

Visible and changeable properties

The following table shows the properties that you can see or change for a type or version:

Property	Description	Version	Types
Name	Name of the type	-	Visible and changeable
Version	Version number	Visible	-
Last change	If you create, release, duplicate or assign a version, this is recorded as a change to the type. The date and time of the change are entered in the "Last change" field.	Visible	-
Author	The creator of the version is indicated as the author.	Visible	-
Original library	The project and the library from which the current version of the type was generated are indicated. This information is important, for example, for finding the original of the type after it has been copied from another library.	Visible	-
Version GUID or Type GUID	The GUID is used to uniquely identify the type or the version of the type, even if, for example, there are types or versions with an identical name within the project library or the global library. The GUID cannot be changed and is assigned automatically.	Visible	Visible
Comment	Comments on the type or version	Visible and, for versions whose status is "in test" or "in work", change-able	Visible and changeable

8.5 Working with types in the project library

8.5.1 Adding types to the project library

Types for reuse in the project can be created in the project library for various elements. You can create the following elements as types, for example:

- Program blocks (FCs/FBs)
- PLC data types (UDTs)
- Faceplates
- HMI user data type
- Screens
- Styles

If you add an element as a type to the project library and this element has dependencies to other elements, the dependent elements are automatically created as a type as well.

After a type has been added to the project library, the type is linked to the added element from the project. The element from the project is thus the instance of the type.

Requirement

- The "Libraries" task card is displayed.
- The elements that you want to add as type are compiled.
- The elements are consistent.
- All other requirements for blocks are met, as described in section "Block requirements (Page 61)".

Procedure

To add an existing element to the project library as a type, follow these steps:

1. Open the project library in the "Libraries" task card.
2. Drag-and-drop one or more elements into the "Types" folder or any subfolder of "Types".
   Alternative: Copy the elements in the project tree to the clipboard and add the elements in the desired project library folder.
   The "Add type" dialog opens.

3. Enter the properties of the new type:
   - Enter a name for the new type in the "Name of type" field.
   - Enter a version number for the new type in the "Version" field.
   - Enter the name of the editor who is responsible for the type in the "Author" field.
   - Enter a comment on the type in the "Comment" field.
4. Click "OK" to confirm.
   The new type is generated with a released version. The version is linked to the element (instance) that has been added.

Note

Creating a name for the author

You can enter a user name in the settings. If you add types, create and release type versions or create master copies in the library, the user entered in the settings is stored in the properties as the author by default.

See also

Basics on types (Page 53)

Duplicating types (Page 63)

Block requirements (Page 61)

Referencing of objects within a block (Page 62)

Library basics (Page 7)

Adding master copies (Page 48)

8.5.2 Block requirements

Permitted blocks for creating a type

You can create types of the following blocks in the project library:

- Function blocks (FBs)
- Functions (FCs)
- PLC data types (UDTs)

Consistency and compilation

To create a type from a block, the block must be consistent and compiled. When you create a type, the consistency of the block is checked automatically and the block is compiled if necessary.

Block requirements

To create a type from a block, the block must meet the following requirements:

- The block must match the type of CPU.
  A block which does not match a CPU can be identified by an incompatibility symbol to the right of the block in the project tree. The case occurs, for example, if you copy a block from a S7-1500 CPU to an S7-300 CPU. A block of an S7-1500 is not compatible with an S7-300 CPU and cannot be compiled. This means that you cannot create a type from the block.
- The block is not a system data block.
- The block does not contain an individual instance call of an instance data block or a call of a data block within an STL block with the "OPN" command.

Special aspects for know-how-protected blocks

You can also create types of know-how-protected blocks. However, keep in mind the following additional requirements for know-how-protected blocks:

- Release of the block for use as type
  Know-how-protected blocks must be released to be used in a library. You make the necessary settings in the block properties.
- The block does not include access to data blocks, PLC tags or PLC constants.
  Because you cannot create types from data blocks, PLC tags or PLC constants, you also cannot create a type from know-how-protected blocks with access to the objects listed above. The instance of a know-how-protected block does not work without the associated data blocks, PLC tags or PLC constants.
- If you edit the test version of a type in the "Libraries" task card or library view, you can set up know-how protection in the block properties.
  The know-how protection can be removed later via the block properties in the project navigation.

See also

Adding types to the project library (Page 60)

Referencing of objects within a block (Page 62)

8.5.3 Referencing of objects within a block

In many cases you reference other objects within a type. For example, call a tag or a data block within block. Accordingly, there is a dependence between the block and the referenced object. The block is only consistent if all dependent objects are also available.

Referencing other types

If you call other program blocks, faceplates, PLC data types (UDTs) or HMI user data types in a block these dependent objects are also stored automatically as type in the project library. When the referenced block is reused later as instance in the project, an instance of the referenced object is also created. The TIA Portal thus automatically ensures that the block state is consistent at all times and that the user program can be executed.

Referencing additional objects

Access to data blocks, PLC tags or PLC constants is permitted in blocks without know-how protection, Therefore, types can be created from the blocks. While function blocks referenced in a block, for example, are also automatically created as type in the project library, this is not the case for access to data blocks, PLC tags and PLC constants. If data blocks, PLC tags or PLC constants are referenced in a block, only the block itself is created as type. The referenced objects are not created as type. If you reference objects which are not stored as type, note the following:

- Use the block at a different point of use
  If you create an instance from the type at another point of use (such as a different CPU), the referenced objects are missing at the later point of use. This means that you should create referenced objects at the later point of use again (for example, data blocks or tag tables). Make sure that the referenced objects have the same name at the later point of use.
- Subsequent name changes to referenced objects
  If you subsequently change the name of the referenced objects, the block is no longer consistent. The referenced object (for example, a tag) is no longer found. In the case of already released versions of the type, you can no longer correct the call of the object. In this case create a new version of the type and correct the name of the called object. Then update all the instances of the type in the project to the latest version.

See also

Block requirements (Page 61)

Adding types to the project library (Page 60)

Performing a consistency check for a version (Page 69)

Updating the project to the latest versions (Page 73)

Displaying relations to other library objects (Page 27)

8.5.4 Duplicating types

Types in the project library can be duplicated. If you duplicate a type, the following applies to the duplicate:

- The duplicate is created in the same folder.
- The duplicate is created from the highest version of the type.
- The duplicate does not have an instance in the project.

Requirement

The type is released.

Procedure

To duplicate a type in the project library, follow these steps:

1. Right-click on a type.
2. Select the "Duplicate type" command in the shortcut menu.
   The "Duplicate type" dialog opens.
3. Enter the properties of the new type:
   – Enter a name for the new type in the "Name of the type" field.
   – Enter a version number for the new type in the "Version" field.
   – Enter the name of the editor who is responsible for the type in the "User" field.
   – Enter a comment on the type in the "Comment" field.
4. Click "OK" to confirm.
   The new type is generated with a released version.

See also

Adding types to the project library (Page 60)

Editing library elements (Page 109)

8.5.5 Using types of the project library

Types from the project library can be used any number of times within a project. The use of types is always linked to a version of the respective type in the project library. If the type contains dependent elements, these are also created as a type of use at a suitable point in the project. Examples of dependent elements are PLC data types referenced in a block.

You can only assign a version of the same type to a device. If necessary, you can create uses of several types at the same time.

Uses of a type are called instances in the project tree.

Possible points of use for type versions

To use a type, create an instance of a particular version of the type at a suitable point in the project. Suitable points of use are:

- Folder in the project tree
  An instance of a type can be created in a folder in the project tree. The folder must be suitable for the particular type. If the type is a user data type, for example, you can only create the instance in the "PLC data types" folder.
- Editor
  An instance can be created from a type in a suitable editor. For example, you can create an instance from the type of a function block in another block. The type of the function block is called from another block in this way.

Options for creating an instance

You have two options for creating type instances:

- Drag the required type version from the project library and drop it at the location where you want to use it.
  Instances of the types and their dependent elements are generated and pasted at the location where you want to use them. The instances are connected to the respective type version in the project library. If you have created the instances in an editor, instances of the types are also created at the appropriate points in the project tree. By default, the folder structure from the library is reproduced in the project tree. If you want to select a different folder in the project tree, you will find the instances in the same folders as in the library.
- Copying and pasting type instances
  You can copy type instances to the clipboard and paste them at another point. This way you create another instance of the type version. The instance is still connected to the same type version in the project library. When you copy the instance of a type to the clipboard and paste it in another project, all required type versions are accepted in the project library of the other project.

Requirement

- The desired versions have been released.
- A device which supports the category of desired type is already available in the project
- The device is not yet assigned any further instance of the same type.

Procedure

To create an instance of a type, follow these steps:

1. In the project library, select the versions from which you want to create an instance.
2. Using a drag-and-drop operation, move the versions from the project library to the point of use in the project tree or an editor.
   Alternative: To automatically use the newest version, move the types themselves from the project library to the point of use using a drag-and-drop operation.
   For example, use a drag-and-drop operation to move the type of a function block to the block folder of the CPU in the project tree. To call the type directly from another block, for example, move the type from the project library to the point of use in the program editor using a drag-and-drop operation.

Alternative:

1. Copy one or more instances to the clipboard.
2. Paste the instances at a suitable point in the same project or in a different project.

See also

Basics on types (Page 53)

State of type versions (Page 56)

Displaying types for an instance (Page 66)

Library basics (Page 7)

Using master copies (Page 51)

8.5.6 Displaying types for an instance

You can jump from the project tree to the type in the library that belongs to instance in the project tree.

Procedure

To jump to the type in the project library, follow these steps:

1. Right-click the instance of the type in the project tree.
2. Select the "Go to type" command from the shortcut menu.
   The referenced type version is displayed in the project library.

See also

Using types of the project library (Page 64)

8.5.7 Creating a test version of a type

Before you release a type for productive use, you need to test the type within a project and on the automation system. The test is conducted in a specific test environment. This test environment can be a CPU, for example.

Create a version with "in test" status for the test. The creation of a version "in test" is suitable for all sorts of types except for faceplates and HMI user data types. On the other hand, versions "In work" can be created from faceplates and HMI user data types.

There are two ways to create a test version of a type and define the test environment:

- In the "Libraries" task card or library view
  Generate the new version with "in test" status in the "Libraries" task card or in the library view. You can generate the new version either directly from the type or from a specific version of the type.
- At an instance in the project tree
  You can also create the test version directly at the instance in the project tree. Since the instance is always used in a specific version in the project, a new version of the type is generated from the version used at the instance.

You can also create test versions from several types at the same time.

The following rules apply to a version "in test":

- You can set only one version to "in test" for each type at a given time.
- A version in testing may only be linked to a single instance in the project. Therefore, it is not possible to copy an instance to the clipboard, to duplicate it or to create an additional type from the instance as long as it has "in test" status.

Requirement

- There is at least one instance of the type within the project in a given version.
- If you want to create the new version from a particular version of the type, the instance must be used in this version in the project.

Procedure

To generate a new test version of a type or the version of a type, follow these steps:

1. Select the type, a version of the type, or the instance.
   When you create the test version directly at the instance, you can select several elements or folders with multiple selection. You can skip steps 3 and 4 because the test environment is already defined by the selected instance.
2. Right-click the selected element and select the "Edit type" command from the shortcut menu.
   If you have started the editing in the "Libraries" task card or in the library view, the "Edit type" dialog opens. If you have started the editing at the instance in the project tree, the test instance is immediately opened for editing in the library view.

3. Select an instance of the type from the list in the project.
   If you have started editing at the type itself, the following applies:
   – The location at which the instance is used (for example, the CPU) is also used as test environment for the subsequent editing of the type.
   – Selecting the test instance also defines the version to be edited.

   The following applies to editing a specific version:
   If your starting point is a specific version, you can only select from the list instances that are used in the same version.
4. Click "OK" to confirm.

Result

A new version of the type is created. The new version is "in test" and is identified as such in the user interface.

See also

Basics on types (Page 53)

State of type versions (Page 56)

Library basics (Page 7)

8.5.8 Editing test version of a type

When you edit a version "in test", a new version is not created. You can start the editing of the test version at the instance in the project tree, in the "Libraries" task card or in the library view.

Note

Deleting and renaming interface parameters

You can add new parameters. However, if you rename or delete existing parameters, these parameters will not be supplied when the block is called.

Procedure

To edit the test version of a type, follow these steps:

1. Right-click the test version or the instance.
2. Select the "Edit type" command from the shortcut menu.
   The test instance is opened and can be edited.

See also

Basics on types (Page 53)

State of type versions (Page 56)

Library basics (Page 7)

Performing a consistency check for a version (Page 69)

Discarding versions (Page 70)

Release versions (Page 71)

8.5.9 Creating an editing version of a type

If you want to edit a type with faceplates or HMI user data types, create a new "In work" version of the type. The new version is edited in the library view. To check the compatibility of the changes, a consistency check is automatically performed for the type prior to the release.

Requirement

The project library is opened in the "Libraries" task card or in the library view.

Procedure

To create a new version of a type in work, follow these steps:

1. Right-click the type or the version of the type.
2. Select the "Edit type" command from the shortcut menu.
   A new "In work" version is created and opened for editing in the library view.

See also

Basics on types (Page 53)

State of type versions (Page 56)

Library basics (Page 7)

8.5.10 Performing a consistency check for a version

A type version can unintentionally obtain an inconsistent state during editing. To notice errors in the development process in good time, you can regularly perform a consistency check. However, the consistency check always takes place automatically as soon as you release a version.

Details of how to start the consistency check manually for the version of a type are provided below.

Requirement

- The project library is opened in the "Libraries" task card or in the library view.
- The version is "In work" or "In test".

Procedure

To perform a consistency check for the version of a type, follow these steps:

1. Right-click on the version that you want to check for consistency.
2. Select the "Consistency check" command in the shortcut menu.
   The consistency check is carried out. You receive a message with the result of the consistency check.

See also

Release versions (Page 71)

Discarding versions (Page 70)

Editing test version of a type (Page 68)

Basics on types (Page 53)

State of type versions (Page 56)

Library basics (Page 7)

8.5.11 Discarding versions

Discard versions of a type "In test" or "In work" when you no longer need the version. You can also select several types or folders and discard all test or working versions contained therein. All uses of the deleted versions are reset to the last released status.

Requirement

- The version that you want to discard is in the state "In test" or "In work".
- Your are in the library view of the "Libraries" task card is open.

Procedure

To discard one version, follow these steps:

1. Right-click the version that you want to discard.
2. Select the "Discard changes and delete version" command from the shortcut menu.
   The version is deleted.

Alternative in the library view:

1. Click the "Discard changes and delete version" button in the toolbar while a version is opened for editing.
   The version is deleted.

See also

Basics on types (Page 53)

State of type versions (Page 56)

Library basics (Page 7)

Performing a consistency check for a version (Page 69)

8.5.12 Discarding all versions within a folder

You can discard all versions with the "In test" or "In work" status at the same time within a folder. All uses of the deleted version are reset to the last released status.

Requirement

Your are in the library view of the "Libraries" task card is open.

Discarding a version of a single type

To discard all versions within a folder, follow these steps:

1. Right-click the folder.
2. Select the "Discard all" command from the shortcut menu.
   All "In test" or "In work" versions are deleted.

8.5.13 Release versions

When you are finished editing a type version, release the version for productive use. Assign a version number for the release. You can also use a multiple selection to release several versions at the same time.

Requirement

- The "Libraries" task card is open or you are in the library view.
- The versions that you want to release are "In test" or "In work" status.
- The versions are consistent.
  A consistency check is performed as soon as you start the release. If errors that prevent a release occur during the consistency check, a message is displayed indicating how you can correct the errors.

Procedure

To release type versions, follow these steps:

1. Select the versions you want to release.
2. Right-click your selection.

3. Select the "Release version" command from the shortcut menu.
   The "Release type version" dialog box opens.
4. If necessary, change the properties of the version:
   – Enter a name for the type in the "Name" field. If you have selected several versions for release, the "Name" field cannot be changed.
   – In the "Version" field, define a main and an intermediate version number for the version to be released. If you have selected several versions for release, the "Version" field cannot be changed and the last version number is used for the release.
   – In the "Author" field, enter the editor of the version to be released.
   – In the "Comment" field, enter a comment on the version to be released.
5. Optional: Select the "Delete unused type versions from the library" check box to delete all versions from the library that are not connected to any instance in the project. Versions with dependencies on other types or master copies are not deleted.
6. Click "OK" to confirm.

Alternative in the library view:

1. Click the "Release version" icon in the toolbar while a version is opened for editing.
2. Continue with steps 3 to 5 of the description above.

Result

The selected versions are released.

The properties are applied for the types themselves, the versions to be released, and for all future versions. Versions already released remain unaffected by the changes.

If needed, all instances with the same original version are updated to the most recent version and the unused versions of the type are deleted.

See also

Releasing all versions within a folder (Page 72)

Basics on types (Page 53)

State of type versions (Page 56)

Library basics (Page 7)

Performing a consistency check for a version (Page 69)

Assigning a version (Page 82)

Adding types to a global library (Page 77)

Updating the project to the latest versions (Page 73)

8.5.14 Releasing all versions within a folder

When you are finished editing all types within a folder, release all versions simultaneously.

Requirement

- Your are in the "Libraries" task card or in the library view.
- The folder includes versions "In test" or "In work" status.
- All versions "In test" or "In work" are consistent.
  A consistency check is performed as soon as you start the release. If errors that prevent release occur during the consistency check, a message is displayed indicating how to correct the errors.

Procedure

To release all type versions within a folder, follow these steps:

1. Right-click the required folder.
2. Select the "Release all" command from the shortcut menu.
   The "Release type version" dialog box opens.
3. If necessary, change the properties of the version:
   – In the "Author" field, enter the editor of the versions to be released.
   – In the "Comment" field, enter a comment on the versions to be released.
4. Select the "Delete unused type versions from the library" check box to delete all versions from the library that are not connected to any instance in the project. Versions with dependencies on other types or master copies are not deleted.
5. Click "OK" to confirm.

Result

All type versions "In test" or "In work" status within the selected folder are released.

The properties are applied for the versions to be released and for all future versions. Versions already released remain unaffected by the changes.

Versions of types not used in the project may be deleted.

See also

Release versions (Page 71)

8.5.15 Updating the project to the latest versions

After you have updated several types in the project library, update all instances in the project to the most recent version of the types from the project library. If you do not want to apply the changes to the entire project, restrict the update to individual devices in the project.

Each of the following elements can be selected as source for the update:

- The entire project library
- Individual folders within the project library
- Individual types
  You can select multiple types.

Note

Deleting unused type instances from the project

By default, when the project is updated to the latest type instances, unused type instances are deleted from the project.

In order to prevent deleting unused type instances from the project, select the check box "Do not remove unused type version instances when updating the project from the library".

Requirement

Your are in the "Libraries" task card or in the library view.

Procedure

To update instances in a project with the contents from the project library, follow these steps:

1. Select the entire project library or elements from it.
2. Right-click the required elements and select the "Update types > Project" command from the shortcut menu.
   The "Update types in the project" dialog box opens.
3. Select either the entire project or individual devices for the update.
4. Select the options for the update process:
   - The "Update instances in the project" check box is always selected during this process.
   - Select the "Delete all unused type versions from the library" check box to delete all older versions of the updated types from the project library.
5. Click "OK" to confirm.
   The update is performed.

Result

The following changes were made to the project:

- All older versions are deleted from the project library as needed.
- All instances within the selected devices are updated to the most recent version of the linked type.

- If a CPU does not include block instances that are called in the user program, the block instances are removed from the CPU. The types still exist in the library and can be used in the project again as needed.
- You can find a log of the update process in the projection tree under "Common data".

See also

Updating the project to the latest type versions (Page 79)

Updating library with the types of another library (Page 89)

Basics on types (Page 53)

Library basics (Page 7)

State of type versions (Page 56)

8.5.16 Remove the link between instance and type

Instances of types are always connected to the version of the corresponding types. They cannot be edited like an ordinary object. If you edit the instance, a new version of the type is created automatically in the project library and the changes therefore affect the entire project.

If you remove the link between the instance and its type, you then edit the object like an ordinary object in the project tree.

Note

Disable "Terminate connection to type"

If you want to prevent instances of your types getting separated in the project, you can also deactivate the "Terminate connection to type" option.

Select the option "Shortcut menu entry 'Terminate connection to type'" under "Options > Settings > General > Library settings".

Requirement

The instance must not be in the "in test" state.

Procedure

To remove the link between instances and their type versions, follow these steps:

1. Select one or more instances in the project tree.
2. Right-click the selection and select the "Terminate connection to type" command from the shortcut menu.
3. The link to the corresponding type versions is removed.

See also

Basics on types (Page 53)
Library basics (Page 7)
State of type versions (Page 56)

8.6 Working with types in global libraries

8.6.1 Adding types to a global library

Global libraries are used as a central resource when working on multiple projects. Among the types, only the types in the project library can be edited directly. Therefore, use the project library if you want to work on types. When you are finished editing a type in the project library, you can add the type to a global library. Adding types from the project library corresponds to a normal copy process from the project library.

Requirement

- The "Libraries" task card is displayed or the library view is open.
- The global library to which you want to add types is opened and can be written to.

Procedure

To add types to a global library, follow these steps:

1. Open the required folder in the global library in the "Libraries" task card or in the library view.
2. Drag one or more types from the project library to the "Types" folder or any subfolder of the global library.

Alternative:

1. Copy the required types from the project library to the clipboard.
2. Open the required global library in the "Global library" pane of the "Libraries" task card.
3. Right-click the "Types" folder or any subfolder of "Types".
4. Select "Paste" in the shortcut menu.

Result

The types are inserted in the global library. Dependent types, such as types of HMI user data types or tags, are also copied to the global library, provided they do not already exist there. This ensures that all necessary elements for generating an instance are present in the global library.

If a type already exists in the global library, the described action corresponds to an update of the global library. In this case, the most recent released versions of the types are added to the global library.

See also

Basics on types (Page 53)

Release versions (Page 71)

Assigning a version (Page 82)

Updating the project to the latest type versions (Page 79)

Library basics (Page 7)

8.6.2 Using types from a global library

To use types from the global library, create an instance of a particular version of the type at a suitable point in the project. If necessary, you can create uses of several types at the same time. Uses of a type are called instances in the project tree.

Possible points of use for type versions

Suitable points of use for types from the global libraries are:

- Folder in the project tree
  An instance of a type can be created in a folder in the project tree. The folder must be suitable for the particular type. If the type is a user data type, for example, you can only create the instance in the "PLC data types" folder.
- Editor
  An instance can be created from a type in a suitable editor. For example, you can create an instance from the type of a function block in another block. The type of the function block is called from another block in this way.

Linking the instance with the project library

In the project, instances of types from a global library are not linked to the type in the global library. Instead, a copy of the type and its dependent elements is generated in the project library when an instance is created. Dependent elements, for example, can be PLC data types that are referenced in a block. In each case, the copy of the type and the dependent elements in the project library contains the version that you have linked to the instances. If the type or a dependent element already exists in the project library, only the missing version is added in the project library, if necessary.

The instance is finally linked to the copy of the type in the project library. You can only assign a type to a device once irrespective of the version.

Requirement

- A device which supports the category of the type is already available in the project.
- The device is not assigned to any further instances of the same type.

Procedure

To use the version of a type in the project, follow these steps:

1. In the global library, select the versions from which you want to create an instance.
2. Using a drag-and-drop operation, move the desired versions of the types to the point of use.
   Alternative: To automatically use the newest version, move the types themselves from the library to the point of use using a drag-and-drop operation.
   For example, use a drag-and-drop operation to move the type of a function block to the block folder of the CPU in the project tree. To call the type directly from another block, for example, move the type from the library to the point of use in the program editor using a drag-and-drop operation.

Result

Missing types or individual versions are added in the project library. If a type is not yet present in the project library, it is stored in the same folder as before in the global library. An instance is created from the types and their dependent elements and inserted at the point of use. The instances are connected to the respective type version in the project library.

If you have created the instances in an editor, instances of the types are also created at the appropriate points in the project tree. The folder structure from the library is reproduced in the project tree. You will therefore find the instances in the same folders as in the global library.

See also

Basics on types (Page 53)

Updating the project to the latest type versions (Page 79)

Library basics (Page 7)

Using the element view (Page 14)

8.6.3 Updating the project to the latest type versions

In large enterprises with numerous automation projects. the global libraries are frequently edited from a central location. The updated global libraries of the individual projects are made available after the completion of a new version. If you have received a more recent version of a global library, replace the outdated instances in your project with the most recent version. If you do not want to apply the changes to the entire project, restrict the update to individual devices in the project.

The project library is also updated with the new versions of the types in the global library during the updating of the project or individual devices.

The following elements can be selected as source for the updating:

- The root node of global libraries
- Individual folders within a global library
- Individual types
  You can select multiple types.

Note

Deleting unused type instances from the project

By default, when the project is updated to the latest type instances, unused type instances are deleted from the project.

In order to prevent deleting unused type instances from the project, activate the check box "Disable deleting of type instances not used in the project" in the settings under "General > Library settings".

Requirement

- Your are in the "Libraries" task card or in the library view.
- The updated global library is open.

Procedure

To update instances in a project with the contents from a global library, follow these steps:

1. Select the updated global library or the individual elements from it.
2. Right-click the global library or the required elements and select the "Update types > Project" command from the shortcut menu.
   The "Update types in the project" dialog box opens.
3. Select either the entire project or individual devices for the update.
4. Select the options for the update process:
   – The "Update instances in the project" check box is always selected during this process.
   – Select the "Delete all unused type versions from the library" check box to delete all older versions of the updated types from the project library.
5. Click "OK" to confirm.
   The update is performed.

Result

The following changes were made to the project:

- The most recent version of the select types is present in the project library. All older versions were deleted if necessary.
- All instances within the selected devices were updated to the most recent version of the linked types.
- You can find a log of the update process in the projection tree under "Common data".

See also

Updating the project to the latest versions (Page 73)

Updating library with the types of another library (Page 89)

Basics on types (Page 53)

Library basics (Page 7)

Adding types to a global library (Page 77)

8.7 Assigning a version

A library is more clearly structured if types related by content have the same version number. The identical version number reflects the work progress. When you have completed the work on multiple associated types, you can assign the same version number to these types.

You have the following options to assign a common version to types:

- One or more folders in a library
- One or more types

Requirement

- The "Libraries" task card or the library view is open.
- Your selection must not contain types with "In test" or "In work" status.

Procedure

To assign several types the same version, follow these steps:

1. Select the types to which you want to assign a common version.
2. Select the "Assign version" command from the shortcut menu.
   The "Assign version" dialog box opens.
3. If necessary, change the properties of the version:
   – In the "Version" field, determine the new version number. The version number must be higher than the highest version number of all selected types.
   – In the "Author" field, enter the person responsible for the version to be released.
   – In the "Comment" field, enter a comment on the version to be released.
4. Click "OK" to confirm.

Result

The selected type versions are changed as follows:

- A new version of all selected types is created with the specified version number.
- The properties are applied to all selected types, the new version and to all future versions. Lower versions remain unaffected by the changes. If you make no changes to the properties, the properties of the last released version of each type are applied.
- The build number of dependent types is incremented to the next free build number as long as the dependent types were not in your selection. If you had selected a dependent type as well, the version number you specified will be assigned.

A log of the changes is created. If you have versioned the types in the project library, you find the log in the project tree under "Common data > Logs". If you have versioned types in a global library, you find the log in the "Common data > Logs" folder in the level below the global library.

See also

Basics on types (Page 53)
State of type versions (Page 56)
Library basics (Page 7)
Release versions (Page 71)
Adding types to a global library (Page 77)
Displaying logs of global libraries (Page 35)

Multiuser Engineering in libraries

9

9.1 Multiuser Engineering with library elements

Library elements in the project library can be used in Multiuser Engineering.

More information on working with Multiuser Engineering can be found in section "AUTOHOTSPOT".

Markings for the check-in

Individual library elements within the project library are marked for Multiuser Engineering in the same way as in the project tree. You can mark library elements in the following areas of the TIA Portal:

- "Libraries" task card
- Library view
- Library management
- At the respective point of use, e.g. at type instances

Always mark the library elements manually.

The figure below shows the "Libraries" task card with the marking column and marked library elements for the multiuser function:

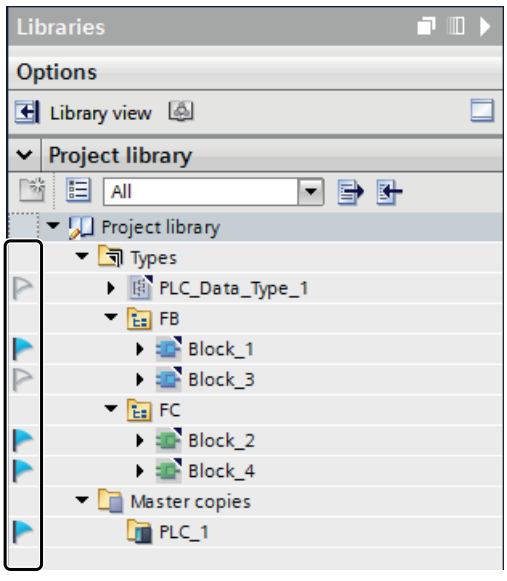

Library elements that can be marked

The following table shows which types can be marked for check-in in the project library and in the instance in the project:

	Type can be marked in the library	Instance can be marked
Function blocks (FB)	Yes	Yes
PLC Block Library Type (FC)	Yes	Yes
PLC data types (UDT)	Yes	Yes
HMI faceplates	Yes	By marking the screen in which the instance of the block is located
HMI user-defined data type (UDT)	Yes	By marking the object (e.g. screen or HMI tag), in which the UDT is used.
HMI style	Yes	No
HMI style sheet	Yes	Not available
HMI screen	Yes	Yes
HMI C script	Yes	Yes
HMI VB script	Yes	Yes

Deleting types in the local session

If you want to delete types in the local session, note the following:

- Release the type if a version is still being tested or edited.
- To delete the type in the assigned server project, mark the type in the local session before deletion.

Individual versions can be deleted in the local session, but these are retained in the server project. Because deleting is an administrative activity, you delete the types and their versions in the server project.

Updating the project

If you want to update the project, note the following:

- Mark the types before you update the project.
- Mark the HMI user data types at the point of use before you update the project.
- Changes to HMI styles are not applied in the local session when you update the project. Update the HMI style in the server project view.

9.2 Checking in types and master copies

Check changes to types and master copies into the multiuser server project in the same way as all other objects within the local session. When you check in a type, all versions of the type are always checked in. When you check in an instance, the type is also checked in automatically.

Update the server project

Types and master copies are identified and synchronized based on the unique GUID in the server project. When types and master copies are checked in, their changes from the local session are applied in the server project:

- The properties of the type or the master copy, e.g. the name, are applied in the server project
- Missing versions of types are added.
- The folder structure within the library is applied in the server project. However, empty folders are not deleted.

Conflicts can occur through updating the server project. Any conflicts are resolved as follows:

- Naming conflicts
  Two types with identical names cannot be checked in. The names of the respective types automatically receive an extension (<_1>, etc.).
  Two types instances with identical names are not permitted and result in an error when checking in.
- A type is "In test" or "In work" status.
  If a type is in one of these two states in the server project or in the local session, you cannot update the server project. First release all marked types before you update the server project.
- Version conflicts
  A message will notify you if both the server project and local session have versions with the identical version number but different content. Change the version numbers so that these exist only once.

More information on Multiuser Engineering can be found in section "AUTOHOTSPOT".

Updating and upgrading types of a library 10

10.1 Updating library with the types of another library

An existing library can be updated with the contents of another library. The following options are available for updating libraries:

- Updating a global library with types from another global library or from the project library
- Updating the project library with types from a global library

Each of the following elements can be selected as source for the update:

- Root nodes in the project library
- Root nodes of global libraries
- "Types" folder
- Individual folders within the "Types" folder
- Individual types
  You can select multiple types.

During the update, new versions are added to the types that already exist in the target library. Types that do not yet exist in the target library are copied together with all their versions to the target library.

Note

User-defined documentation when copying types

User-defined documentation is not copied together with a type to another library. You need to copy the user-defined documentation for types to the corresponding directory.

For additional help on using the user-defined documentation, refer to the section "AUTOHOTSPOT".

Requirement

If you want to update a global library, you have to open it with write permission.

Procedure

To update a library with the contents of a different library, follow these steps:

1. Select the "Types" folder or individual elements from this folder as source for the update.
2. Right-click the source and select the "Update types > Library" command from the shortcut menu.
   The "Update types in a library" dialog box opens.

3. Select the type of library you want to update:
   – Select "Update the project library" to update the project library with types from a global library.
   – Select "Update a global library" if you want to update a global library.
4. Optional: In the drop-down list, select the global library that you want to update, if you want to update a global library.
5. Select the options for the update:
   – The option "Update instances in the project" is always deactivated in this process.
   – Select the "Delete all unused type versions from the library" check box if you want to delete all versions of types from the library if these are not assigned to any instance in the project and if there are no dependencies to other types. This option cannot be selected for the update of a global library, since types of a global library never have a point of use in the project.
6. Click "OK" to confirm.
   The update is performed.

Result

The following changes were made to the target library:

- Types not yet available in the target library were copied together with all their type versions. More recent type versions were added to the types that already existed in the target library. If a more recent version of a type already existed in the target library, the latest version was nevertheless copied and automatically assigned a newer version number.
- If needed, all versions of types were deleted from the project library if these were not used in any instance in the project.
- A log listing all performed changes to the target library was created for the update process. If you have updated the project library, you can find the log in the project tree under "Common data > Logs".
  If you have updated a global library, you can find the log in the "Common data > Logs" folder in the level below the global library.

See also

Updating the project to the latest versions (Page 73)

Updating the project to the latest type versions (Page 79)

Displaying logs of global libraries (Page 35)

Library basics (Page 7)

10.2 Upgrading types in an existing project

Using the function "Upgrading types", you can upgrade the type dependencies to the latest available type versions in one step throughout the entire project.

If you edit a type that references another type within your project library and create a new type version, the referenced type is automatically set to the state "in test" or "in work". This ensures that all types within the project are up to date.

If you edit a type outside of the project and create a new type version, the referenced types are not automatically updated. To ensure that these dependencies to the individual referenced types do not have to be manually upgraded, the "Upgrading of types" function is available in the library management.

Using this function you can identify all referenced types that are dependent on the changed type and automatically upgrade these types. The "in test" or "in work" extension is added to all upgraded types in the project library. After the upgrading you can see at a glance which types are also affected by the changes and can check these individually and release them once again.

Requirement

You have created a project that, for example, contains program blocks that call other program blocks or objects, such as PLC data types (UDTs). Dependencies thus exist between the individual objects.

You have created a project library and stored the types to be edited in a global library. The types were further developed outside of your project and are now to be re-integrated with the new type versions in your existing project library. Update the project library for this purpose.

To integrate the new type versions into the existing project library, follow these steps:

1. Open the global library without write protection.
   The types to edited have a higher type version than the types that are contained in the project library.
2. In the global library, click the folder "Types" and select "Update types > Project" from the shortcut menu.
   The project library is updated and the new type versions are included.

Additional information on updating libraries is available here: Updating library with the types of another library (Page 89).

At this time the new type versions are only available in the project library, but not yet referenced in the project.

Procedure

To automatically reference new type versions in the project, these must be upgraded in the library management.

Proceed as follows:

1. In the project library, click the folder "Types" and open the library management.
2. In the "Types" area, select the entry "Types with new versions for upgrading other types" from the drop-down list.
   All types are displayed for which a new type version is available in the project library.
3. Right-click the "Types" folder and select the entry "Show upgradable types" from the shortcut menu.
   All referenced types are displayed in the "Upgrading of types" tab.
4. Select one of two entries in "Upgrading of types":
   – Only in a test environment
     All types that reference an instance in the test environment are displayed.
   – Compatible with selected test environment
     All types that reference an instance in the test environment are displayed. However, the instance was not yet created at this time.
5. Select a test environment
   With the selection of a test environment, in which test environment the types are to be upgraded.
6. All types that have to be upgraded are now displayed.

Note

Display of types

The display of types depends on which folder you selected in the library management and which settings you made in the "Upgrading types" tab.

7. Click "Start".
   The "Upgrading of types" dialog opens after the successful upgrading.

   Note

   "Error" status

   You cannot start the upgrading of types if one the types to be upgraded has the "Error" status. You have two options:

   1. Correct the error and restart the upgrading.
   2. Refresh the view of the possible types to be upgraded, by selecting the required entry from the drop-down list under "Types to be upgraded".

   All referenced type are now in the project tree and in the "in test" state in the project library.

8. Check the types "in test" and then release them. You can also use the management library for this purpose.
9. To do so, select the "Types with pending changes" from the drop-down list and release all displayed types.

Result

All types within your project now reference the current type version.

10.3 Example of the updating and upgrading of types

Description

You have a project library whose types reference one other in different ways.

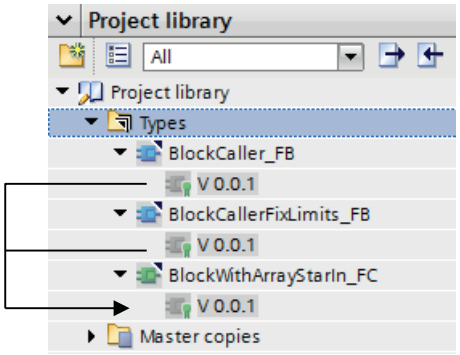

The function "BlockWithArrayStarIn_FC" is called by each of the two function blocks.

The function "BlockWithArrayStarIn_FC" should be further developed independently outside of the project library.

Requirement

You have created a project library with all relevant types.

Procedure

Proceed as follows to further develop the type "BlockWithArrayStarIn_FC" independently outside of the project:

1. Copy the type from the project library to a global library in order to be able to transfer it.

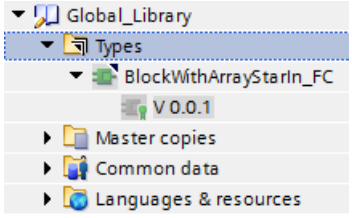

   You transfer version V0.0.1 for further development.

2. The type "BlockWithArrayStarIn_FC" was edited and released. The provider has created a new version V2.0.0 and returned the global library with the new type version:

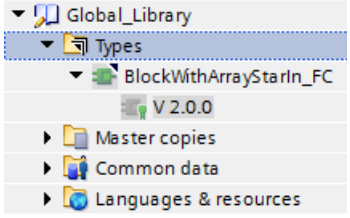

To integrate the new versions of the function into your existing project library, follow these steps:

1. Click the "Types" folder of the global library "Global_Library".
2. Select the entry "Update types > Project" from the shortcut menu.
   The "Update types in the project" dialog box opens.
3. Click "OK".
   The new type version V2.0.0 has been integrated in the project library.

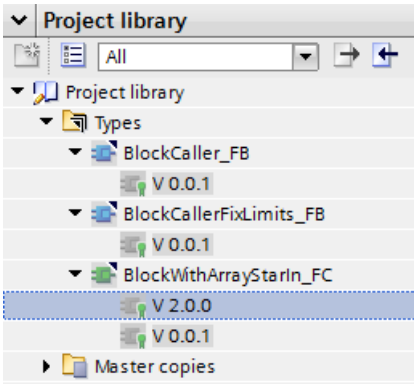

Note

Referencing the latest type version

At this time the new type version is only available in the project library. The instances in the project tree still reference the obsolete type version V0.0.1.

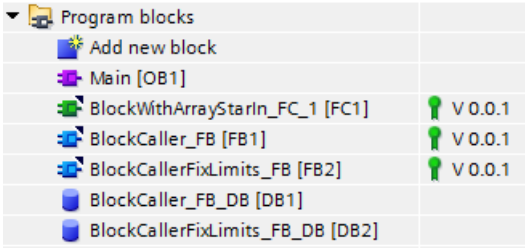

To upgrade the instances in the project tree to the new type version in one step, follow these steps:

1. Click the "Types" folder in the project library and open the library management.
2. In the "Types" area, select the entry "Types with new versions for upgrading other types" from the drop-down list.
   The function that a new type version has that is not yet used in the project tree is displayed.

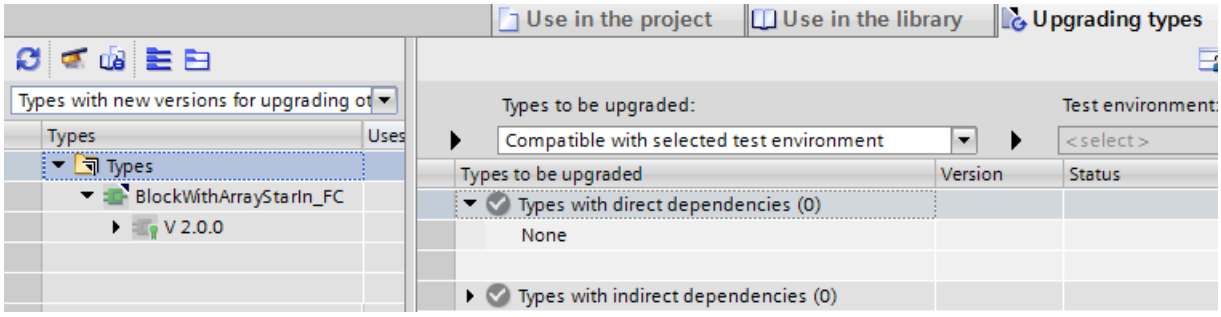

3. Click the "Types" folder and select the entry "Display upgradable type" from the shortcut menu.
   The "Upgrading types" tab opens and all types are displayed which have a direct dependency to the changed type.

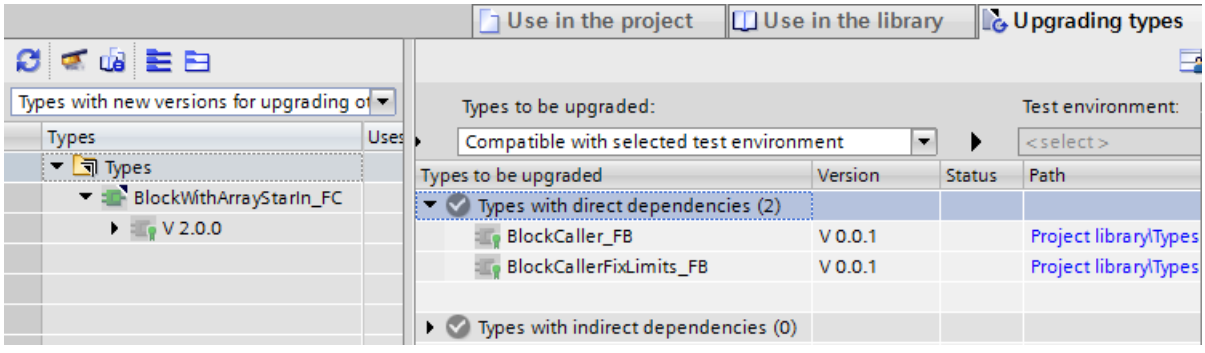

4. To filter the upgraded types for a test environment, select the entry "Only with test environment" under "Types to be upgraded".

5. Select the test environment "PLC S7-1500".

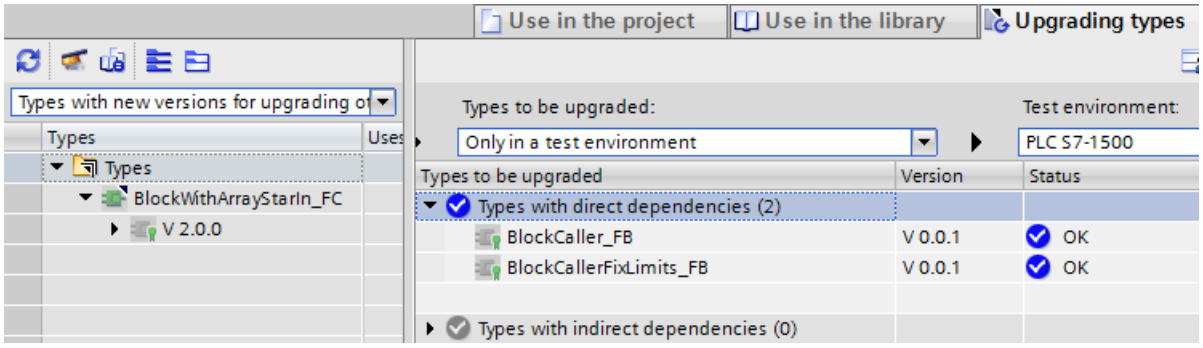

All types that can be upgraded have the status "OK".
If one or more types have the status "Error", you cannot start the upgrading. First correct the error and then repeat the steps 1 - 5.

6. Click "Start".

Note

Upgrading with indirectly dependent and non-instantiated type versions.

If you want to upgrade a type which references one or more type versions that are not instantiated in the project themselves, note the effects of the possible upgrade options in the library management.

If you have selected the "Only with test environment" option and set a test environment, the type can be upgraded. A warning is displayed for the non-instantiated type versions; these type versions are not upgraded.

If you select the "Compatible with selected test environment" option and set a test environment, the type cannot be upgraded. An error message is displayed for non-instantiated type versions; these type versions prevent upgrades.

Result

All referenced types are upgraded to a new version in the project library and in the project tree and the extension "in test" is added.

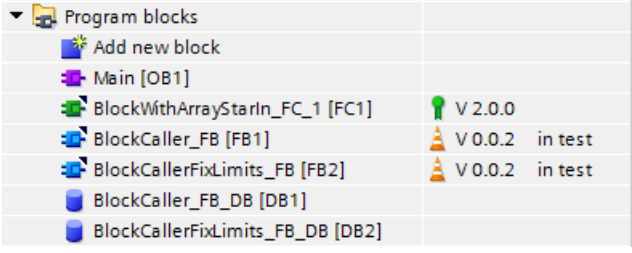

Check the new versions and then release these.

Using library texts 11

11.1 Exporting library texts

Description

You can export library element texts for translation and then reimport them. The texts are exported to an Office Open XML file with the extension ".xlsx". This can be edited in Microsoft Excel or a number of other spreadsheet programs.

The following export options are available:

- Exporting texts of selected types and master copies
  You can select individual types or master copies, and use the shortcut menu to export only the texts of the selected object.
  Or you can select and export multiple or all types or master copies by making multiple selections.
- Exporting texts of all library elements of the project library or a global library
  You can use the buttons in the toolbar of the "Project library" or "Global libraries" pane to export all texts of a project library or a global library.

If multiple versions of one type exist, the texts of the latest version are exported.

Note

No exporting of library texts

If the current version of a type has not been released yet and has the extension "In test" or "In work", no library texts are exported.

Note

Row limit in Microsoft Excel

Note that spreadsheet programs may be able to process only a certain number of rows. Microsoft Excel 2003 supports a maximum of 65536 rows, for example. Later versions of Microsoft Excel support more rows.

Defining the source and target languages in a project library

The selection of possible source and target languages in the project library depends on the languages you have defined as project languages.

You can find additional information here: AUTOHOTSPOT

Defining the source and target languages in global libraries

In a global library you can define the library languages yourself unless the library has write protection.

You can display the selection of possible languages by double-clicking on the "Library languages" item in the "Languages & Resources" subfolder. For a write-protected library the library languages are displayed as read-only in the library view.

You can find additional information here: Defining library languages (Page 33)

Requirement

Activate required target languages:

- For a project library in the project languages
- For a global library in the library languages

Exporting texts of library elements

To export the texts of the selected library elements, follow these steps:

1. Select the required library elements (individual library element, multiple library elements, a folder or a complete library).
2. Either right-click and select "Export library texts" from the shortcut menu or click the "Export library texts" button in the toolbar.
   The "Export" dialog box opens.

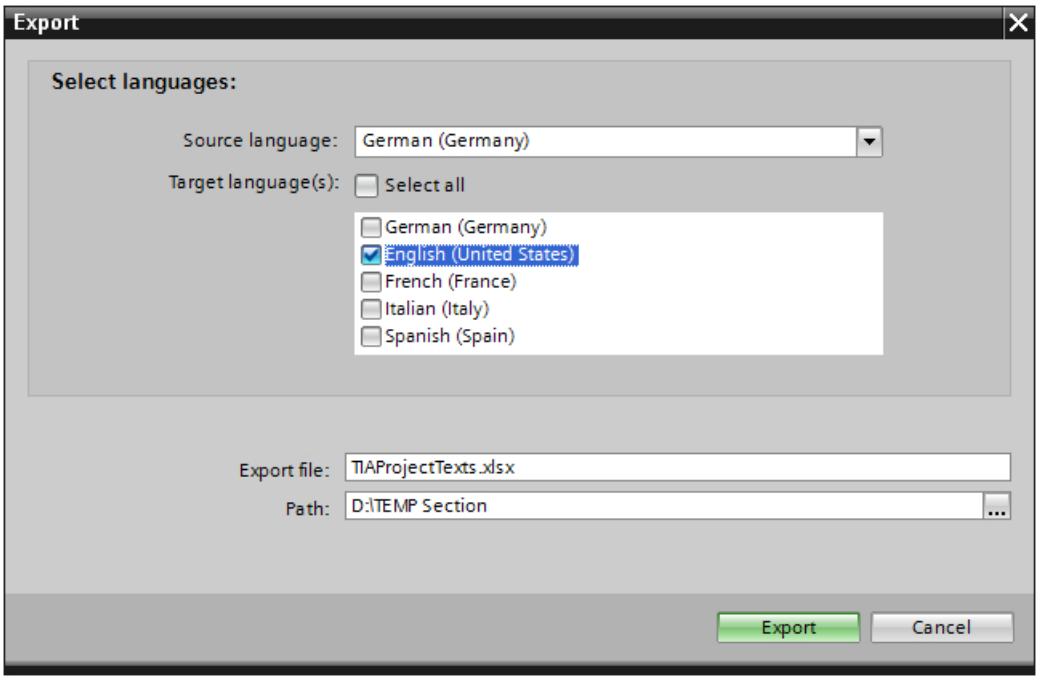

3. Select the required source language.
4. Select the required target language.
   Select the "Select all" check box if you want to export all target languages.
5. Give the export file the required name.
6. Set the required path to the storage location of the export file.
7. Click "Export".

Result

The export was successfully completed and the ".xlsx" file was created in the storage location.

	A	B	C	D	E	F
1	Category	ViewPath	Internal ID	Substitutions	de-DE*	en-US
2	<BlockCommentCategoryData>	Program blocks\	a5exBpTAQ6Af2xKwOQN99f		Start Motor	
3	<BlockCommentCategoryData>	Program blocks\	a5exBpTAQ6Af2xKwOQN99f		Stop Motor	

Write the English translations in the "en-US" column, in order to be able to then import them.

11.2 Importing library texts

Description

After external translation in a spreadsheet program, you import the texts into the TIA Portal. You can import the texts in the project library or a global library at the following locations:

- Using the "Import library texts" shortcut menu command
- Using the "Import library texts" button.

Regardless of how many library elements you have selected, all texts from the import file are imported for the entire library. The library elements from the import file must match the library elements of the library into which the texts are to be imported. The target languages you want to import must be enabled in the project or in the global library.

When you import texts for master copies, the existing texts are overwritten with the new texts. When you import texts for types, only the texts of the latest version are overwritten with the new texts.

Note

No import of library texts possible

No import of library texts possible if one of the following options applies:

- The library texts belong to a type instance which is contained in a master copy.
- A version of a type has not yet been released within the project library and has the extension "in test" or "in work".
- The global library is write-protected.
- It is a system library.

Note

Importing library texts for a multiuser project.

To also import library texts from objects that are not marked, the library texts in a multiuser project must be imported in the server project view.

Requirement

You have entered the translations in the target language in the export file.

	A	B	C	D	E	F
1	Category	ViewPath	Internal ID	Substitution	de-DE*	en-US
2	<BlockCommentCategoryData>	Program blo	a5exBpTAQ6Af2xKwOQN		Start Motor	Start the motor
3	<BlockCommentCategoryData>	Program blo	a5exBpTAQ6Af2xKwOQN		Stop Motor	Stop the motor

If you want to import library texts into a project library, you must activate the required target languages in the project languages.

If you want to import library texts into a global library, you must activate the required target languages in the global library.

Importing all texts of a project library or a global library

To import all translated library texts, follow these steps:

1. Select the project library or a global library.
2. Either right-click and select "Import library texts" from the shortcut menu or click the "Import library texts" button in the toolbar.
   The "Import" dialog box opens.
3. Select the file to be imported.
4. You have the option to also import the source language by enabling the option Import source language".
5. Click "Import".
   The "Import completed" dialog opens after the import.

Result

- Import successfully completed:
  The latest type version contains the translated texts after a successful import.
- Import completed with warnings:
  The reason for the error is either indicated directly in the dialog or you can click the link "Click here to view the log file" to view the causes of the error.
- Import failed:
  To find the log with the causes of the error, click the link "Click here to view the log file".

11.3 Defining the project languages in the project library

All defined project languages are also available in the project library.

You can define and change the project languages centrally for the entire project library. In this case, all changes to the project languages affect all existing and newly created types and master copies in the project library.

However, you can also change the project languages for individual type versions without affecting the rest of the project library.

Procedure

To define the project languages for the entire project library, follow these steps:

1. Open or edit a type or master copy.
2. Open the "Tasks" card.
3. Under "Languages & resources" either select another project language from the existing editing languages or click "Add new editing language".
4. If you click "Add new editing language", the list of all available project languages will be displayed.
5. Select the editing language you want to use.

Procedure

To define the project languages for an individual type version, follow these steps:

1. Select the appropriate type version from the project library.
2. Select the command "Tools > Project languages" from the menu bar.
   Shows the list of all available project languages.
3. Select the editing language you want to use.

Result

The project library or the type version has the requested project or editing languages.

11.4 Updating imported library tests for type instances

If you export texts of type versions and then import them to the same type version again, the changes made are not displayed in the type instances. Due to the same versions of the type and instance, for example V 0.0.1, no changes to the texts are detected on update of the project or the library.

Displaying texts from the project library in the project

You have the following options for displaying the changes from imported tests from the project library in the project:

- Delete the instance in the project and then create a new instance of the same version in the project using drag & drop. This procedure is necessary if you want to display imported texts in a different device or object in the project.
- Import the modified texts directly into the instances in the project. To do this, use the "Import project texts" in the "Tools" menu. This procedure is only necessary if you want to display imported texts in the same device and object in the project.

Displaying imported texts from global libraries

You can display changes of imported texts from global libraries in the project library or in an instance in the project. For this, create a new version of the imported object, e.g. V 0.0.2. You have the following options for displaying the imported texts:

- To display the imported texts in the project library, update the project library. For this, open the shortcut menu of the object and select the menu entry "Update types > Library".
- To display the imported texts in the project, update the project. For this, open the shortcut menu of the object and select the menu entry "Update types > Project".

Creating folders in a library

12

The library elements are stored in the libraries according to their type in the "Types" and "Master copies" folders. Create additional folders below "Types" and "Master copies" to further organize master copies and types.

Requirements

- The "Libraries" task card is displayed or the library view is open.
- If you want to create new folders within a global library, you have to open the global library with write permission.

Procedure

To create a new folder, follow these steps:

1. Right-click any folder within the library.
2. Select "Add folder" from the shortcut menu.
   A new folder is created.
3. Enter a name for the new folder.

See also

Filtering master copies (Page 50)

Working with types in the project library (Page 60)

Editing library elements 13

Types, master copies and folder can be cut, copied, pasted, moved, renamed or deleted in the usual way within the "Libraries" task card or the library view. Global libraries must be opened with write permission for each of the above-described processes.

Note

User-defined documentation for types and master copies

User-defined documentation is not affected by any of the operations within the library. If you move a master copy or a type to a different location, you also move the associated user-defined documentation in the file system to the corresponding location.

For additional help on using the user-defined documentation, refer to the chapter "AUTOHOTSPOT".

Copying types

The following rules apply when you copy types to the clipboard:

- Types are always copied to the clipboard with all associated versions. However, only versions that have previously been released are copied.
- Types are always copied to the clipboard with all dependent elements.
- Master copies are always copied to the clipboard with all type versions used in them.

Copying and pasting type versions

When you copy type versions to another library, the types must already exist in the target library.

Cutting elements

You can only paste previously cut library elements into the same library. In so doing, you can only paste master copies into the "Master copies" folder or any of its subfolders. Likewise, you can only paste types into the "Types" folder or any of its subfolders.

Pasting types

Pasting types in a different library corresponds to updating the target library.

The following rules apply when you have pasted a type into a different library:

- A type is always pasted with all its versions.
- If the type already exists in the target library, all versions that are not yet available are added to the corresponding types in the target library.

- If there is already a version with released status in the target library, this version is not pasted again.
- If the same version exists with "In test" or "In work" status in the target library, it is replaced by the released version.
- If a type needs other types, these are also added at the respective location.

Pasting master copies

When you paste master copies, all type versions contained in these copies are also pasted. If the corresponding types already exist in the library, only the missing versions are added to the individual types. If one of the types used does not yet exist, it is pasted at the highest level in the library. The type includes the type version that was used in the master copy.

Moving elements

When you move an element from one library to another, the element is copied and not moved. The same rules apply as described under "Pasting types" and "Pasting master copies".

Deleting of types and type versions

Note the following when you delete types or type versions:

- A type or a type version can only be deleted if there are no dependencies to other types.
- When you delete a type, all versions of the type are deleted.
- If you delete all versions of a type, the type is also deleted.
- If you delete a version that has instances in the project, the instances are also deleted from the project.
- If you delete a type that is also stored at the same time as a master copy, the master copy is also deleted.

Deleting instances

If you delete an instance that has dependencies to other instances, this instance is restored during the next compilation. The instance is then linked again to the original type version. This restores the consistency of the project.

See also

Library basics (Page 7)

Remove the link between instance and type (Page 75)

Updating library with the types of another library (Page 89)

Duplicating types (Page 63)

Harmonizing names and path structure 14

You can harmonize the project with a library. This helps you correct the following items:

- Names of the instances:
  Instances can be created during the development phase of a library, the names of which are appended by "_1", "_2", etc. due to an automatic correction. This extension is used to prevent duplicate names in the project. During harmonization, the instances once again receive the names of their associated types.
- Path structure:
  The original path structure can be lost due to parallel development or copying of dependent instances. This affects the clarity of the project. During harmonization, the path structure within the project is adapted to the path structure of the library.

Procedure

To harmonize the names and the path structure, follow these steps:

1. Open the library management.
2. Click "Harmonize project" in the toolbar.
   The "Harmonize project" dialog box opens.
3. Select the device with which you wish to harmonize the library.
4. Select the "Harmonize paths between project and library" check box if you want to restore the path structure.
5. Select the "Harmonize names between project and library" check box if you want to have the names corrected.
6. Confirm your entries with "OK".

Result

Depending on your settings, the names and the path structure in the project are harmonized with the library.

The changes to the project are logged. The log is available under "Common data > Logs" in the project navigation.

See also

Library basics (Page 7)

Overview of the library view (Page 15)

Overview of the library management (Page 19)

Comparing library elements 15

Introduction

You can compare devices from libraries with devices from both the current project as well as from the same or another libraries or reference projects. Note, however, that reference projects are write-protected. You can also compare instances in a device with their type version in a library. Not all actions are available for the comparison with types. You cannot, for example, overwrite an instance of a newer version with an older type version from the library.

When comparing library elements, you can always switch between automatic and manual comparison.

Procedure

To compare library elements with the device data of a project, follow these steps:

1. In the project tree, select the device whose data you want to compare to a library element and which allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Open the "Libraries" task card.
4. Select the library element that you want to compare to the device data.
5. Drag the library element into the right drop area of the compare editor.
   You can identify the status of the objects based on the symbols in the status and action area. When you select an object, the object properties and the corresponding object of the assigned device are clearly shown in the property comparison.
   You can drag other devices into the drop areas from the current project, a library or from a reference point at any time to start a new comparison. It does not matter which device you drag into the drop area.

See also

Using the library view (Page 15)

Library basics (Page 7)

Overview of the "Libraries" task card (Page 11)

Overview of the library view (Page 15)

Using the element view (Page 14)

Using global libraries (Page 29)

Clean up library 16

You can clean up the project library and the global libraries to remove types or versions that are not connected to any instance in the project. This step provides more clarity within the libraries and decreases the size of the library.

Cleaning up the project library

To clean up the project library, follow these steps:

1. Open the library management.
2. Click on "Clean up library" in the toolbar.
   The "Clean up project library" dialog box opens.
3. Select the scope in which types or types versions are to be deleted:
   – To retain the version with the highest version number, even if this has no instance, select the option "Delete old type versions and retain the newest type version".
   – Select the option "Delete complete types" to delete the complete type if no version is connected to an instance.
4. Confirm your entry with "OK".
   Depending on your selection, either unused type versions or types are removed from the project library.
   The changes are logged. The log is available under "Common data > Logs" in the project navigation.

Clean up global library

To clean up a global library, follow these steps:

1. Open the global library in not write-protected mode.
2. Open the library management.
3. Click on "Clean up library" in the toolbar.
   The "Clean up global library" dialog box opens.
4. Click "OK".
   Unused type versions are deleted. The latest version of a type is always retained.
   The changes are logged. The log is available in the "Common data > Logs" folder in the level below the global library.

See also

Library basics (Page 7)

Overview of the library view (Page 15)

Overview of the library management (Page 19)

Displaying logs of global libraries (Page 35)

Index

C

Comparing library objects, 8
Corporate libraries, 8

D

Displaying types for an instance, 66

E

Editing a version of a type in testing, 68

G

Global libraries, 8
- Archiving, 39
- Backward compatibility with older product versions, 31
- Corporate libraries, 8
- Retrieving, 40
- System libraries, 8
- User libraries, 8

Global library
- Adding types, 77
- Archiving, 40
- Cleaning up, 115
- Closing, 37
- Creating, 29
- Creating folders, 107
- Deleting, 38
- Displaying logs, 35
- Opening, 32
- Saving, 36
- Showing properties, 34
- Updating the project, 80
- Using filter view, 50
- Using types, 78

Global library with write protection
- Creating, 30

I

Instance
- Creating an instance, 65
- Displaying associated type, 66

L

Library
- Adding a master copy, 48
- Basics, 7
- Cleaning up, 115
- Comparing library elements, 113
- Copying types to the clipboard, 109
- Creating folders, 107
- Cutting library elements, 109
- Cutting master copies, 109
- Cutting types, 109
- Deleting instances, 110
- Deleting types and versions, 110
- Harmonizing names and path structure, 111
- Master copies, 7
- Moving library elements, 110
- Opening a released type version, 58
- Pasting master copies from the clipboard, 110
- Pasting types from the clipboard, 109
- Released type version, 56
- Type version in test, 56
- Types, 7
- Using filter view, 50
- Using master copies, 47, 51
- Using types, 53
- Versioning of types, 53

Library management
- Displaying cross references of an instance, 26
- Displaying instances of type versions, 25
- Displaying relations between library objects, 27
- Filtering by non-released types, 24
- Opening, 23
- Overview, 19

Library view
- Exit, 18
- Opening, 18
- Overview, 15

P

Project
- Harmonizing, 111

Project library, 7
- Adding types, 60
- Assigning a common version to types, 82
- Cleaning up, 115

Creating folders, 107
Discarding all versions, 71
Discarding the type version, 70
Duplicating types, 64
Performing a consistency check for a type version, 70
Release all versions, 73
Release the type versions, 71
Updating the project, 74
Using filter view, 50
Using types, 65

S

System libraries, 8

T

Type
Creating an instance, 78
Displaying the properties of a type, 59
Displaying the properties of a version, 59
Types
Copying instances, 65
Creating an instance, 65
Pasting instances, 65
Using types, 65

U

Updating a library, 89
Updating types to the latest version, 89
User libraries, 8

V

Versioning of types
Released version, 56
Version in test, 56

W

Write-protected global library
Creating, 30

SIEMENS

Configuring devices and networks 1

Device and network diagnostics 2

Editing devices and networks

Programming and Operating Manual

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Configuring devices and networks....9

1.1 Hardware and network editor....9
1.1.1 Overview of hardware and network editor....9
1.1.2 Overview of settings for hardware configuration....10
1.1.3 Network view....11
1.1.4 Device view....14
1.1.5 Device view: Objects in the device view....16
1.1.6 Device view: Area for unplugged modules....19
1.1.7 Topology view....20
1.1.8 Inspector window....23
1.1.9 Inspector window: Displaying UDTs....26
1.1.10 Hardware catalog....28
1.1.11 Browsing the hardware catalog....31
1.1.12 Enabling product support....33
1.1.13 Displaying product support for hardware components....33
1.1.14 Printing hardware and network configurations....34
1.1.15 Printing: Changing settings....36
1.1.16 Printing: Enabling page break preview....36
1.1.17 Keyboard operation: Navigation in the editor....37
1.1.18 Keyboard operation: Editing objects....38

1.2 Configuring devices....40
1.2.1 Introduction to configuring hardware....40
1.2.2 Adding a device to the hardware configuration....41
1.2.3 Adding an unspecified CPU....42
1.2.4 Selecting a CPU....43
1.2.5 Selecting fail-safe modules....45
1.2.6 Rack: Basics....46
1.2.7 Rack: General slot rules....47
1.2.8 Rack: Inserting a module....48
1.2.9 Select hardware component....50
1.2.10 Copying a hardware component....54
1.2.11 Moving a hardware component....55
1.2.12 Replacing a hardware component....56
1.2.13 Renaming hardware component....57
1.2.14 Deleting a hardware component....59
1.2.15 Changing or updating hardware components....60
1.2.16 Editing properties and parameters....61
1.2.17 Input and output addresses in the address overview....62
1.2.18 Distributed I/O in the project tree....65
1.2.19 Comparing devices....67
1.2.19.1 Basics of device comparison....67
1.2.19.2 Making a device comparison....68

1.3 Configure networks....69
1.3.1 Networking devices....69
1.3.1.1 Communication and networks....69

1.3.1.2 Networking devices in the network view....70
1.3.1.3 Tabular network overview....73
1.3.1.4 Networking devices in the device view....74
1.3.1.5 Checking or changing network parameters and interface parameters....75
1.3.1.6 Changing networkings....76
1.3.1.7 Copying, cutting or deleting subnets....77
1.3.1.8 MPI network configuration....78
1.3.1.9 PROFIBUS network configuration....79
1.3.1.10 Configuring Industrial Ethernet....85
1.3.1.11 Settings for interconnecting Ethernet devices....87
1.3.2 Communication via connections....91
1.3.2.1 Working with connections....91
1.3.2.2 Connections - Purpose and selection (S7-300, S7-400, S7-1500)....186
1.3.2.3 Configuring connections (S7-300, S7-400, S7-1500)....206
1.3.2.4 Connection types (S7-300, S7-400, S7-1500)....234
1.3.3 Displaying and configuring topology....285
1.3.3.1 Overview of the topology view....285
1.3.3.2 Starting the topology view....287
1.3.3.3 Displaying topology....288
1.3.3.4 Configuring topology....295
1.3.4 Secure Communication....302
1.3.4.1 Basics of Secure Communication....302
1.3.4.2 Global security settings....323
1.3.4.3 Secure Open User Communication....331
1.3.5 Industrial Ethernet Security....346
1.3.5.1 Configuring security....346
1.3.5.2 Configuring security for SCALANCE S-600....550

1.4 Creating configurations....707
1.4.1 Configuring automation systems....707
1.4.1.1 Functional description of S7-300/400 CPUs (S7-300, S7-400)....707
1.4.1.2 Mode of operation of S7-1500 CPUs....899
1.4.1.3 Principle of operation for S7-1500R/H CPUs (S7-1500)....1036
1.4.1.4 Configuring central racks....1077
1.4.1.5 Rules for modules....1077
1.4.1.6 Rules for modules....1078
1.4.1.7 Rules for modules....1080
1.4.1.8 Selecting modules....1088
1.4.1.9 Selecting fail-safe modules....1089
1.4.1.10 Changing the properties of CPUs....1090
1.4.1.11 Linking racks....1092
1.4.1.12 Linking racks....1093
1.4.1.13 Addressing modules....1096
1.4.1.14 Flexible automation concepts....1112
1.4.1.15 Using OPC UA communication....1124
1.4.1.16 Configuring the Web server....1244
1.4.2 Configuring automation systems....1290
1.4.2.1 Functional description of S7-1200 CPUs (S7-1200)....1290
1.4.2.2 Addressing modules....1377
1.4.2.3 Signal board....1381
1.4.2.4 Configurations for Web server....1382
1.4.2.5 Configuration control (option handling) for devices....1416
1.4.2.6 Additional configurations....1427

1.4.3 Configuring PC systems....1427
1.4.3.1 Overview....1427
1.4.3.2 Integrating IPCs into the TIA Portal....1457
1.4.3.3 Basic configuration of the Open Controller....1463
1.4.3.4 Parameter assignment....1465
1.4.3.5 Special features of the S7 Modular Embedded Controller....1476
1.4.4 Configuring CPs and applications for PC stations....1479
1.4.4.1 CPs for PC station 1.0....1479
1.4.4.2 Applications for PC station 1.0....1500
1.4.5 Configurations for point-to-point connections (S7-1500)....1527
1.4.5.1 Configuration for point-to-point links (S7-1500)....1527
1.4.6 Configurations for point-to-point connections (S7-300, S7-400)....1577
1.4.6.1 Configuration for point-to-point links (S7-300, S7-400)....1577
1.4.7 TeleControl....1703
1.4.7.1 Configuring telecontrol modules....1703
1.4.8 SCALANCE X, W and M....1960
1.4.8.1 Configuring SCALANCE X / W / M....1960
1.4.9 Configuring PROFIBUS DP....2728
1.4.9.1 Configuring distributed I/O....2728
1.4.9.2 The basics of configuring a DP master system....2728
1.4.9.3 DP slaves within the hardware catalog....2730
1.4.9.4 DP/DP coupler in the hardware catalog....2730
1.4.9.5 Configurations involving PROFIBUS DP....2731
1.4.9.6 Configuring distributed I/O systems....2735
1.4.9.7 Example: Configuring ET 200S as DP slave....2745
1.4.9.8 Configuring intelligent DP slaves....2755
1.4.9.9 Configuring direct data exchange....2767
1.4.9.10 Using GSD files....2775
1.4.9.11 Using PROFIBUS DPV1....2779
1.4.9.12 Constant and isochronous modes....2790
1.4.10 Configuring PROFIBUS DP....2808
1.4.10.1 The basics of configuring a DP master system....2808
1.4.10.2 DP slaves within the hardware catalog....2808
1.4.10.3 DP/DP coupler in the hardware catalog....2809
1.4.10.4 Configurations involving PROFIBUS DP....2809
1.4.10.5 Configuring distributed I/O systems....2811
1.4.10.6 Configuring intelligent DP slaves....2819
1.4.10.7 Configuring DP slaves as distributed I/O devices....2821
1.4.10.8 Using GSD files....2831
1.4.11 Configuring PROFINET IO....2835
1.4.11.1 What you need to know about PROFINET IO....2835
1.4.11.2 Configure PROFINET IO....2840
1.4.11.3 Using GSD files....2884
1.4.11.4 Special PROFINET configurations....2889
1.4.12 Bus coupling with PN/PN coupler....3026
1.4.12.1 Application and function....3026
1.4.12.2 Linking Ethernet subnets....3027
1.4.13 Integrating external tools....3028
1.4.13.1 Integrating S7-external tools....3028
1.4.13.2 Starting the SIMATIC S7-PCT....3029
1.4.14 Loading a configuration....3030
1.4.14.1 Introduction to loading a configuration....3030

1.4.14.2 Downloading a configuration to a device....3030
1.4.14.3 Loading a configuration to PG/PC....3031

1.5 Displaying alarms....3042
1.5.1 Overview of the alarm display....3042
1.5.2 "Current alarms" view....3043
1.5.3 Layout of the alarms in the "Current alarms" view....3043
1.5.4 Alarm archive....3043
1.5.5 Layout of the alarms in the alarm archive....3044
1.5.6 Clear alarm archive completely....3044
1.5.7 Export archive....3044
1.5.8 Receiving alarms....3045
1.5.9 Freeze alarms....3046
1.5.10 Acknowledging alarms....3046
1.5.11 Status of the alarms....3046
1.5.12 Sort table in alarm display....3047
1.5.13 Filter table in alarm display....3048
1.5.14 Keyboard commands in the alarm display....3048

1.6 Additional information on configurations....3049
1.6.1 Communications modules and network components....3049
1.6.1.1 Industrial Ethernet / PROFINET....3049
1.6.1.2 Industrial Ethernet (S7-1500)....3082
1.6.1.3 CP PROFIBUS (S7-300, S7-400, PC)....3095
1.6.1.4 CM/CP PROFIBUS (S7-1500)....3098
1.6.1.5 CPs settings that do not depend on the protocol or the network....3099
1.6.1.6 Background on connections....3101
1.6.1.7 IE/PB Link PN IO....3108
1.6.2 Identification systems....3111
1.6.2.1 RFID systems....3111
1.6.2.2 Optical reading systems....3187
1.6.3 Distributed I/O....3188
1.6.3.1 Distributed I/O systems....3188
1.6.3.2 Configuring HART variables....3192
1.6.3.3 ET 200iSP....3193
1.6.3.4 ET 200eco PN....3207
1.6.3.5 ET 200SP....3213
1.6.3.6 ET 200AL....3255
1.6.3.7 ET 200MP....3270
1.6.3.8 ET 200M....3283
1.6.3.9 ET 200S....3292
1.6.3.10 ET 200pro....3294
1.6.4 IPv4 routing with CP 1543-1 / CM 1542-1....3301
1.6.5 IPv6 configuration....3301
1.6.5.1 IPv6 protocol....3301
1.6.5.2 IPv6 with the CP 1543-1....3302

2 Device and network diagnostics....3303

2.1 Hardware diagnostics....3303
2.1.1 Overview of hardware diagnostics....3303
2.1.1.1 Principal methods of hardware diagnostics....3303
2.1.1.2 Determination of which of the devices that are connected online are defective....3304
2.1.1.3 Displaying diagnostics status and comparison status using icons....3305

2.1.1.4 Start online and diagnostics view....3311
2.1.1.5 Activation of the "Online Tools" task card....3313
2.1.2 Showing non-editable and current values of configurable module properties....3314
2.1.2.1 Showing general properties and system-relevant information for a module....3314
2.1.2.2 Display configured cycle times....3315
2.1.2.3 Show interfaces and interface properties of a module....3316
2.1.2.4 Displaying IO controllers that access modules of a Shared Device (S7-1500)....3318
2.1.2.5 Displaying sync domain properties of a PROFINET device....3318
2.1.2.6 Displaying MRP domain properties of a PROFINET device....3319
2.1.2.7 Displaying current firmware of a module (S7-300, S7-400, S7-1500)....3320
2.1.3 Showing the current values of dynamic modules properties....3321
2.1.3.1 Display measured cycle times....3321
2.1.3.2 Showing the current status of the LEDs of a CPU....3322
2.1.3.3 Showing fill levels of all types of memory on a CPU....3322
2.1.3.4 Displaying fill level of all types of memory of an S7-1500 CPU (S7-1500)....3323
2.1.4 Checking a module for defects....3325
2.1.4.1 Determining the diagnostic status of a module....3325
2.1.4.2 Performing communication diagnostics of PROFINET ports....3326
2.1.4.3 PROFINET IO component diagnostics....3327
2.1.4.4 Reading out the diagnostics buffer of a CPU....3327
2.1.4.5 Saving service data (S7-1500)....3331
2.1.5 Changing the properties of a module or the programming device/PC....3332
2.1.5.1 Changing the mode of a CPU....3332
2.1.5.2 Performing a memory reset....3333
2.1.5.3 Determining and setting the time of day on a CPU....3334
2.1.5.4 Resetting an S7-1200 CPU to factory settings (S7-1200)....3335
2.1.5.5 Resetting an S7-1500 CPU to factory settings (S7-1500)....3336
2.1.5.6 Resetting a PROFINET IO device to factory settings (S7-1500)....3337
2.1.5.7 Formatting an S7-1500 memory card (S7-1500)....3338
2.1.5.8 Updating the firmware of a module....3338
2.1.5.9 Assigning an IP address to a PROFINET IO device....3341
2.1.5.10 Assigning a PROFINET device name....3343
2.1.5.11 Calibrating an S7-1500 analog module (S7-1500)....3346
2.1.5.12 Loading I&M data to PROFINET IO devices and your modules....3350
2.1.6 Diagnostics in STOP mode....3351
2.1.6.1 Basic information on the diagnostics buffer....3351
2.1.6.2 Determining the cause of a STOP of a CPU....3353
2.1.7 Online accesses in the Online and Diagnostics view....3353
2.1.7.1 Displaying status of the online connection....3353
2.1.7.2 Specifying a PG/PC interface, going online....3354
2.1.7.3 Going offline....3355
2.1.7.4 Performing the flash test for a device with an online connection....3356
2.1.8 Checking PROFIBUS DP subnets for faults....3356
2.1.8.1 Basic information on the diagnostic repeater....3356
2.1.8.2 Displaying the status of the segment diagnostics using icons....3357
2.1.8.3 Displaying the status of the segment diagnostics using graphics and text....3358

2.2 Connection diagnostics....3360
2.2.1 Overview of connection diagnostics....3360
2.2.2 Displaying the connection status using icons....3361
2.2.3 Detailed connection diagnostics....3362
2.2.3.1 Detailed connection diagnostics - overview....3362
2.2.3.2 Determining online connection resources for S7-1200 (S7-1200)....3362

2.2.3.3 Determining online connection resources for S7-1500 (S7-1500)....3363
2.2.3.4 Determining connection details....3364
2.2.3.5 Determining the address details of a connection (S7-1200, S7-1500)....3364

2.3 System diagnostics with 'Report System Errors'....3366
2.3.1 Introduction to system diagnostics with 'Report System Errors'....3366
2.3.2 Basics of system diagnostics....3366
2.3.3 Modules with limited diagnostic possibilities....3367
2.3.4 Diagnostics blocks for reporting system errors....3368
2.3.5 Properties of the blocks....3369
2.3.6 Supported error OBs....3370
2.3.7 Diagnostics status DB....3371
2.3.8 Displaying settings of system diagnostics....3374
2.3.9 Settings....3375
2.3.9.1 General settings....3375
2.3.9.2 Alarm settings....3375
2.3.9.3 Settings for diagnostics support....3375
2.3.9.4 Settings for system diagnostics blocks....3376
2.3.10 Exporting and importing system diagnostic settings....3376

2.4 System diagnostics for S7-1500 PLCs (S7-1500)....3377
2.4.1 Introduction (S7-1500)....3377
2.4.2 Basics of system diagnostics (S7-1500)....3377
2.4.3 Displaying settings of system diagnostics (S7-1500)....3377
2.4.4 Settings (S7-1500)....3378
2.4.4.1 General settings (S7-1500)....3378
2.4.4.2 Alarm settings (S7-1500)....3378
2.4.5 Exporting and importing system diagnostic settings (S7-1500)....3378
2.4.5.1 Basics for exporting and importing system diagnostic settings (S7-1500)....3378
2.4.5.2 Exporting system diagnostic settings (S7-1500)....3379
2.4.5.3 Importing system diagnostic settings (S7-1500)....3379

2.5 Using additional diagnostics options (S7-300, S7-400)....3381
2.5.1 Hardware diagnostics (S7-300, S7-400)....3381
2.5.1.1 Displaying non-editable and current values of assignable module properties (S7-300, S7-400)....3381
2.5.1.2 Displaying current values of dynamic module properties (S7-300, S7-400)....3386
2.5.1.3 Checking a module for defects (S7-300, S7-400)....3389
2.5.1.4 Changing the properties of a module or the programming device/PC (S7-300, S7-400)...3392
2.5.2 Connection diagnostics (S7-300, S7-400)....3400
2.5.2.1 Detailed connection diagnostics (S7-300, S7-400)....3400

Index....3401

Configuring devices and networks 1

1.1 Hardware and network editor

1.1.1 Overview of hardware and network editor

Function of the hardware and network editor

The hardware and network editor opens when you double-click on the "Devices and Networks" entry in the project tree. The hardware and network editor is the integrated development environment for configuring, networking and assigning parameters to devices and modules. It provides maximum support for the realization of the automation project.

Structure of the hardware and network editor

The hardware and network editor consists of the following components:

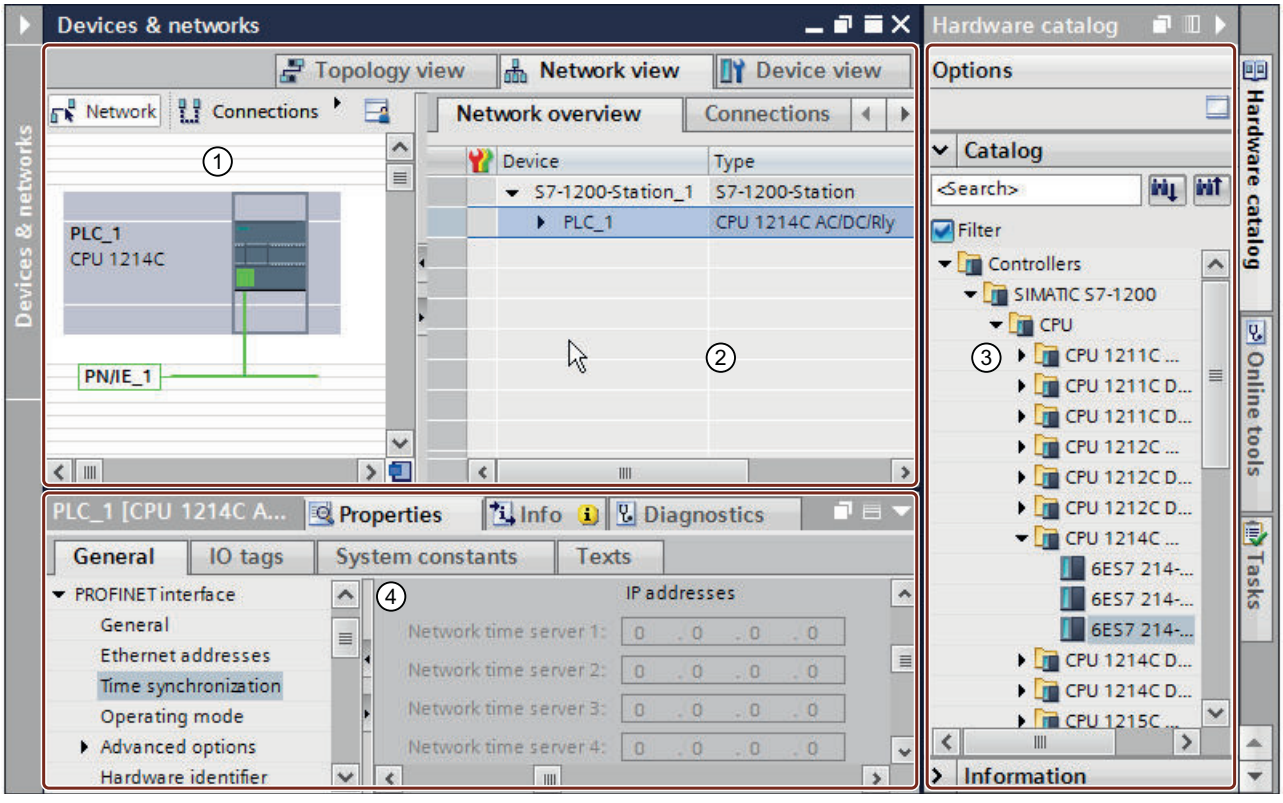

① Device view (Page 14), Network view (Page 11), Topology view (Page 20): Graphic area

② Device view (Page 14), Network view (Page 11), Topology view (Page 20): Table area

③ Hardware catalog (Page 28)

④ Inspector window (Page 23)

The hardware and network editor provides you with three views of your project. You can switch between these three views at any time depending on whether you want to produce and edit individual devices and modules, entire networks and device configurations or the topological structure of your project.

Drag the devices and modules you need for your automation system from the hardware catalog to the network, device ot topology view.

The inspector window contains information on the object currently marked. Here you can change the settings for the object marked.

1.1.2 Overview of settings for hardware configuration

In "Options > Settings > Hardware configuration", you can make various settings relating to the hardware configuration.

Overview

The following table provides an overview of the settings for the hardware configuration:

Group	Setting	Description
Information on product support	Deactivated	Prevents access to the Siemens Industry Online Support
	Via Internet	Enables access to product information about individual devices in the hardware catalog via the Internet.
Topology overview	Temporarily assigning an IP address	Assigns a temporary IP address for topology discovery if a device does not have a valid IP address. Topology information (LLDP) cannot be read from a device without valid IP address.
	Show a warning if the option is activated	Shows a warning when a temporary IP address is assigned to a device during topology discovery.
Compiling and downloading	Download module comment	Transfers any existing comments in addition to the hardware configuration when loading the hardware configuration to the device. The comment is available after the device is loaded to a programming device.
	Download I&M data	This option must be enabled if you want to be able to also download I&M data to the actual hardware when downloading the hardware configuration, the software or both to a device.

See also

Enabling product support (Page 33)

Displaying product support for hardware components (Page 33)

Topology view (Page 20)

1.1.3 Network view

Introduction

The network view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Configuring and assign device parameters
- Networking devices with one another
- Editing the device names

Structure

The following diagram shows the components of the network view:

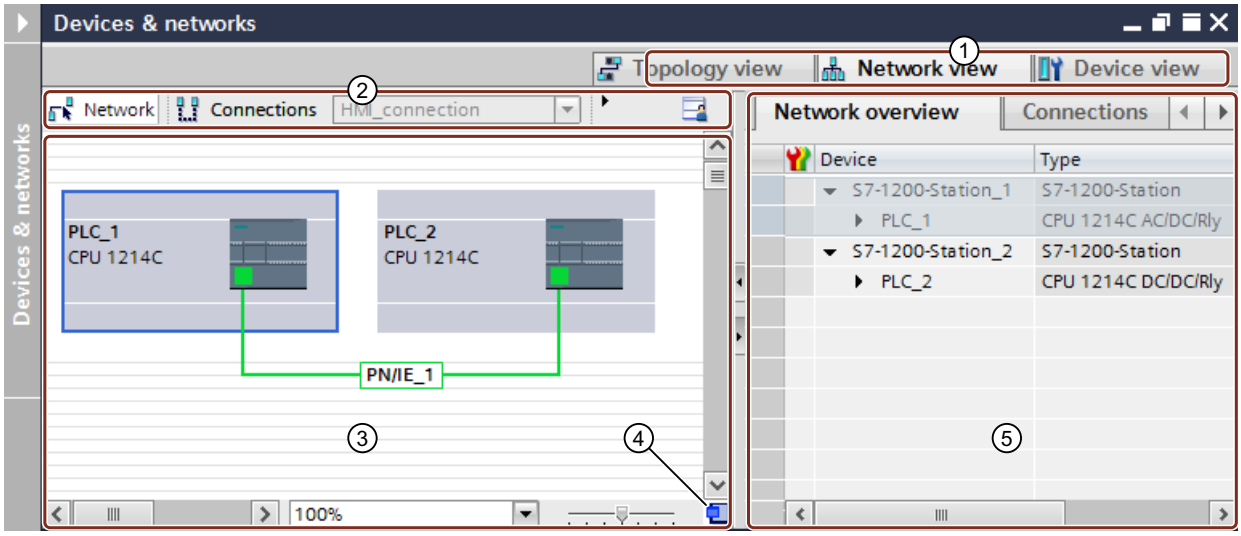

① Changeover switch: device view / network view / topology view
② Toolbar of network view
③ Graphic area of network view
④ Overview navigation
⑤ Table area of network view

You can use your mouse to change the division between the graphic and table areas of the network view. The division can be changed between horizontal and vertical using a button in the toolbar. To change the position of the divider, click between the graphic and the table areas and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize the table view, maximize the table view or restore the last selected split.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Mode to network devices.
	Mode to create connections. You can use the adjacent drop-down list to set the connection type.
	Mode to create relations.
	Opens the dialog for manual name assignment for PROFINET devices. For this purpose the IO device must be inserted and connected online with the IO system.
	Show interface addresses. You can edit the addresses for the MPI, PROFIBUS and Ethernet interfaces in the network view itself: Select the required address and then click on the selected address field or press [F2].
	Enables page break preview. Dotted lines are displayed at the positions where the pages break when printed.

Icon	Meaning
	You can zoom in (+) or zoom out (-) the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
	Changes the division of the graphical area and table area of the editor view between horizontal and vertical.
	Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the network view displays any network-related devices, networks, connections and relations. In this area, you add devices from the hardware catalog, connect them with each other via their interfaces and configure the communication settings.

The operator controls for view control are located at the bottom edge of the graphic area:

- Select the zoom leveling using the drop-down list. You can also enter a value directly into the field of the drop-down list.
- You can also set the zoom level using the slider.
- You can re-focus the window of the graphic area using the icon in the bottom right corner.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate in the overview navigation to the desired objects and display them in the graphic area.

Table area

The table area of the network view includes various tables for the devices, connections and communication settings present:

- Network overview
- Connections
- Relations
- IO Communication
- VPN

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Adding a device to the hardware configuration (Page 41)

Displaying diagnostics status and comparison status using icons (Page 3305)

Networking devices in the network view (Page 70)

Tabular network overview (Page 73)

1.1.4 Device view

Introduction

The device view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Configuring and assign device parameters
- Configuring and assign module parameters
- Editing the names of devices and modules

Structure

The following diagram shows the components of the device view:

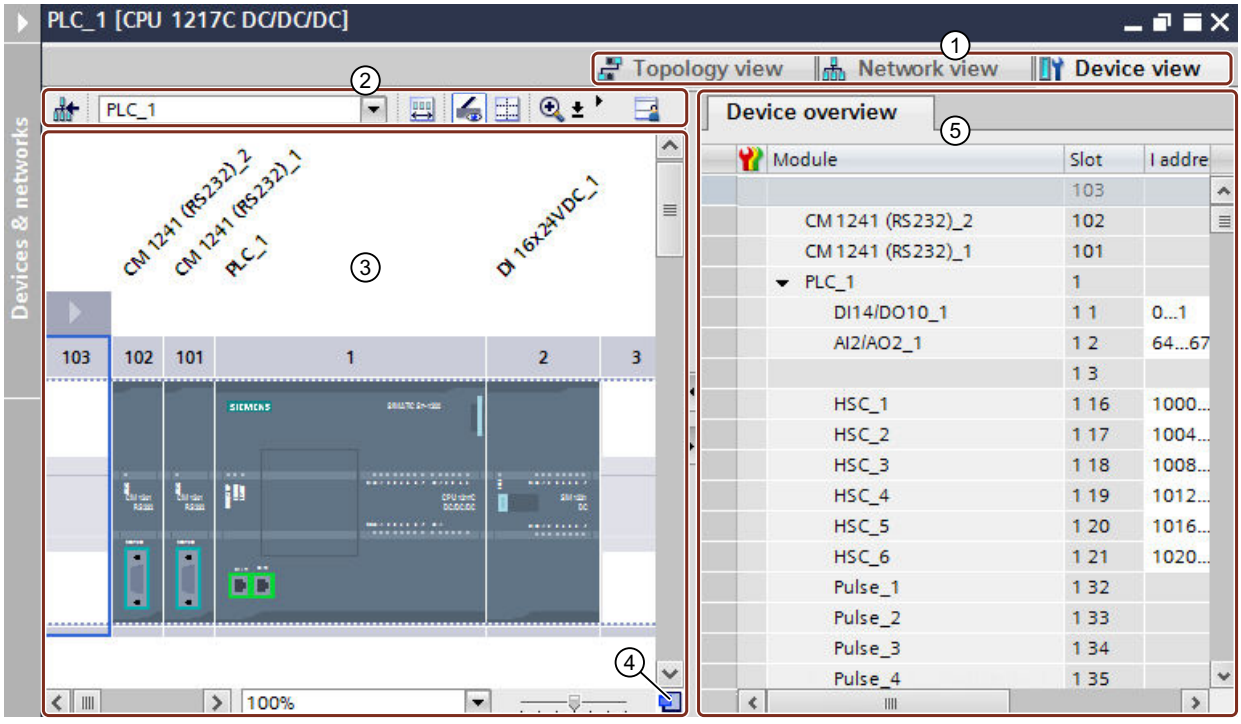

① Changeover switch: device view / network view / topology view
② Toolbar of device view
③ Graphic area of the device view
④ Overview navigation
⑤ Table area of device view

You can use your mouse to change the spacing between the graphic and table areas of the device view. The division can be changed between horizontal and vertical using a button in the toolbar. To change the position of the divider, click between the graphic and the table areas and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize the table view, maximize the table view or restore the last selected split.

Toolbar

You can switch between the devices in your project in the device view using the drop-down list in the toolbar. The following functions are also available for device selection in addition to the drop-down list:

Icon	Meaning
	Switches to the network view. The device view can switch between the existing devices using the drop-down list.
	Show the area of unplugged modules.
	Opens the dialog for manual name assignment for PROFINET devices. For this purpose the IO device must be inserted and connected online with the IO system.
	Show module labels. You can edit the labeling in the device view itself: Select the required labeling and then click on the selected text field or press [F2].
	Enables page break preview. Dotted lines are displayed at the positions where the pages break when printed.
	You can use the Zoom icon to zoom in (+) or out (-) incrementally or to drag a frame around an area to be enlarged. With signal modules, you can recognize the address labels of the I/O channels from a zoom level of 200% or higher.
	Changes the division of the graphical area and table area of the editor view between horizontal and vertical.
	Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the device view displays hardware components and if necessary the associated modules that are assigned to each other via one or more racks. In the case of devices with racks, you have the option of installing additional hardware objects from the hardware catalog into the slots on the racks.

The operator controls for view control are located at the bottom edge of the graphic area:

- Select the zoom leveling using the drop-down list. You can also enter a value directly into the field of the drop-down list.
- You can also set the zoom level using the slider.
- You can re-focus the window of the graphic area using the icon in the bottom right corner.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate in the overview navigation to the desired objects and display them in the graphic area.

Table area

The table area of the device view gives you an overview of the modules used and the most important technical and organizational data.

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Rack: Basics (Page 46)

Network view (Page 11)

Device view: Area for unplugged modules (Page 19)

Rack: Inserting a module (Page 48)

Device view: Objects in the device view (Page 16)

Displaying diagnostics status and comparison status using icons (Page 3305)

1.1.5 Device view: Objects in the device view

A graphic display of the rack and the devices plugged into it is shown in the left area of the device view. You can see the device overview in the right area of the device view. The device overview is a table showing you the most important information about the modules inserted in the rack. Both areas are displayed in a window. The size of both areas is adjusted using a divider. You can also use it to display one or the other area in full or to hide them.

Structure and content of device view

The offline configuration of the devices in the rack are displayed in the graphic device view. This is a symbolic representation of the configuration on the real rack.

The rack configuration is displayed as a table in the device view. Each line in the table contains the information for assigning a slot.

The following screenshot shows the device view with the configuration of a SIMATIC S7-1200 CPU.

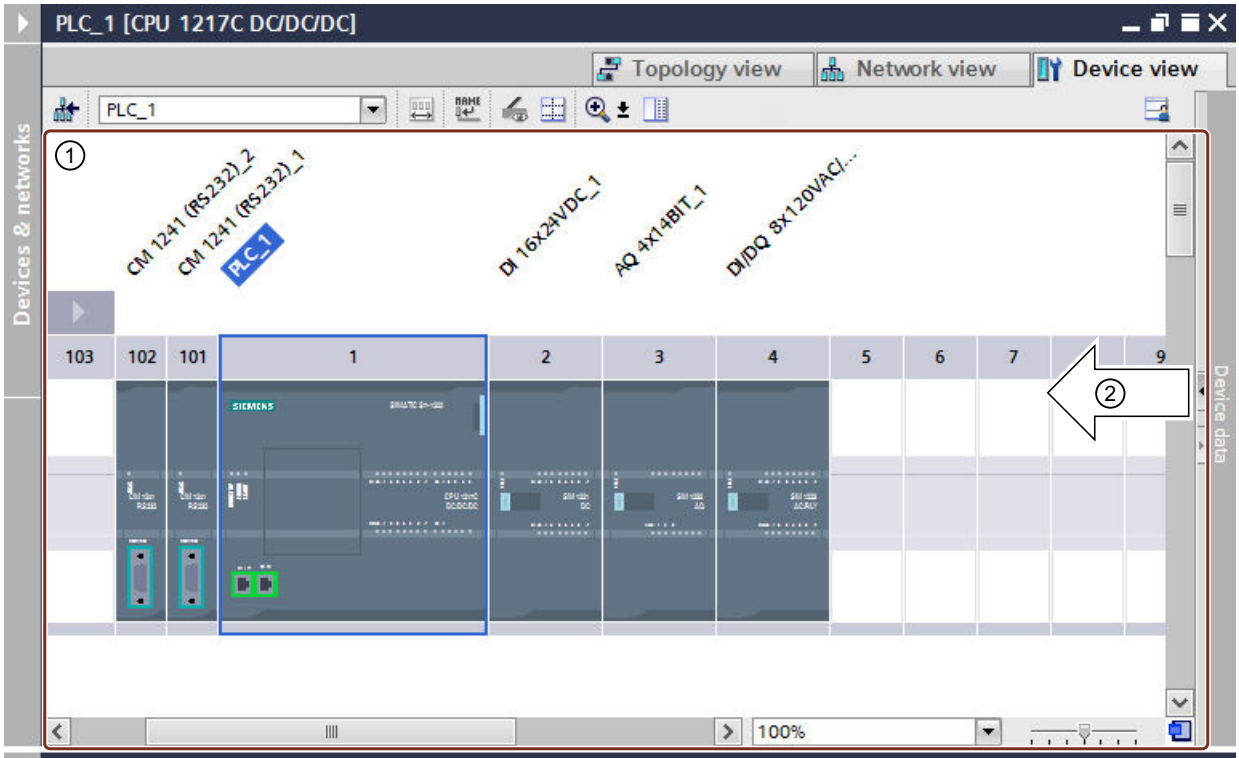

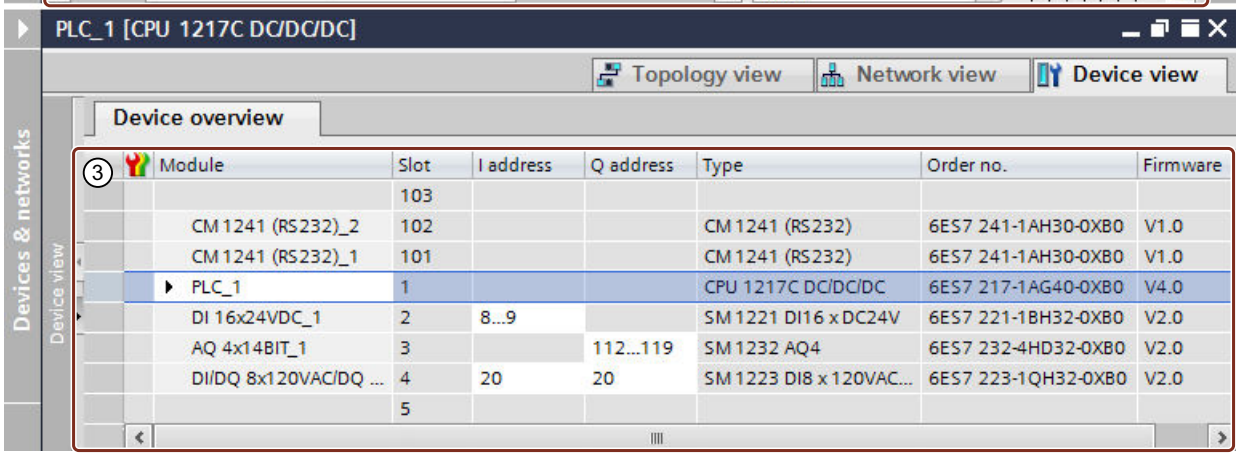

① Graphic view of the allocation of the rack by the CPU and various modules in slots 1 to 4, as well as 101 and 102.

② You can use the divider to adjust the proportion of the device view between the left area (graphic view) and the right area (device overview). If you click on the arrows, you can quickly switch the division of the separate areas.

③ Device view with the tabular representation of the rack's slots and the inserted components. You can show additional columns and hide displayed columns using the shortcut menu of the column title.

Each line in the device overview represents one slot. The key information for each slot is displayed in the various columns:

Column	Meaning
Online status	Symbolic representation of the online status
Fail-safe	Symbolic representation for fail-safe module
Module	Name of module, can be edited in any way

Column	Meaning
Rack	Number of the rack
Slot	Slot number
I address	Input address area, can be edited in any way
Q address	Output address area, can be edited in any way
F-source address	F-source address when using fail-safe I/O
F-destination address	F-destination address when using fail-safe I/O
Type	Catalog name of module
Article no.	Article number of the module
Firmware	Firmware version of the module
Comment	Optional comment

Display in I/O channels

If you set the zoom level in the device view to at least 200%, you can see the individual I/O channels for I/O modules. If you have defined PLC tags for the channels, the names of the PLC tags are displayed.

The figure below shows the input channels of the digital input module with the two PLC tags "Test1" and "Test2" at a zoom level of 400 %.

You can select the individual I/O channels and have the following options for channels with PLC tags:

- You see the general properties of the selected PLC tag in the inspector window under "Properties".
- In the inspector window under "Info > Cross-references" you find the cross-reference information on the selected PLC tag. If you have selected the PLC tag, you can also open the cross-reference information using the shortcut menu.

See also

Device view (Page 14)

1.1.6 Device view: Area for unplugged modules

In some cases, the modules for a hardware configuration are not assigned a slot for short periods. Such unplugged modules are moved to the area of unplugged modules, a special area in the device view.

Adding modules to the storage area

The modules that, for example, should be assigned to a device using a copy action but for which the corresponding rack does not have a free compatible slot, are moved automatically into the area of unplugged modules.

Under the following conditions, modules are automatically added to the area of unplugged modules:

- In the network view, a module is moved to a device but the rack does not have a compatible free slot.
- In the device view, a module is moved or copied from the rack, the hardware catalog or the project tree straight into the storage area.

CPs and FMs that occupy a network resource can be moved into the area of unplugged modules but will lose the network resources they have been assigned.

You can add modules to the area of unplugged modules by means of drag-and-drop, for example. To do this, the area must be opened.

Using the area of unplugged modules

Use the corresponding button to open the area of unplugged modules.

You can find the area of unplugged modules in the device view.

You open the area of unplugged modules with the respective icon in the toolbar of the device view (Page 14).

Icon	Meaning
	Open area of unplugged modules

Note

To free up slots, move modules from your configuration into the storage area and plug the modules you want from the storage area into the freed up slots.

You can use this approach to temporarily move modules whose parameters have already been assigned out of the configuration without deleting them.

Treatment of modules in the storage area

The following rules apply to modules in the storage area:

- The modules appear in the project tree under the corresponding device in the "Local modules" folder.
- The modules retain all settings and parameters previously assigned.
- The modules are not taken into account during loading to a target system. This means that a consistency check is not undertaken for modules in the area of unplugged modules.
- Using the context menu, the modules can be copied, cut, or deleted, for example.

1.1.7 Topology view

Introduction

The topology view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Displaying the Ethernet topology
- Configuring the Ethernet topology
- Identifying and minimizing differences between the desired and actual topology
- Editing the device names

Structure

The following figure provides an overview of the topology view.

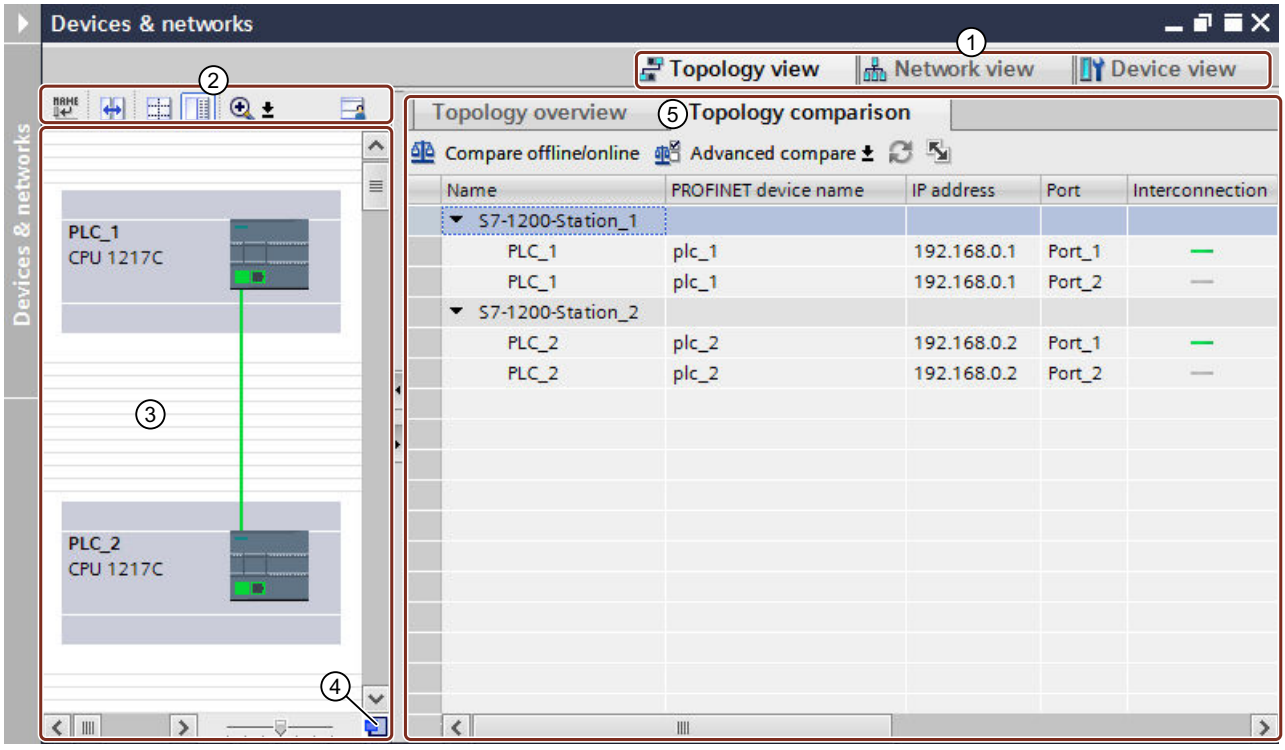

① Changeover switch: device view / network view / topology view
② Topology view toolbar
③ Graphic area of the topology view
④ Overview navigation
⑤ Table area of the topology view (topology overview)

You can use your mouse to change the spacing between the graphic and table areas of the topology view. The division can be changed between horizontal and vertical using a button in the toolbar. To change the position of the divider, click between the graphic and the table areas and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize the table view, maximize the table view or restore the last selected split.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Opens the dialog for manual name assignment for PROFINET devices. For this purpose the IO device must be inserted and connected online with the IO system.
	Changes the position of the devices in the topology view so that the new position is as close as possible to the position in the network view.
	Enables page break preview. Dotted lines are displayed at the positions where the pages break when printed.

Icon	Meaning
	You can zoom in (+) or zoom out (-) the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
	Changes the division of the graphical area and table area of the editor view between horizontal and vertical.
	Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the topology view displays Ethernet modules with their appropriate ports and port interconnections. Here you can add additional hardware objects with Ethernet interfaces. See: Adding a device to the hardware configuration (Page 41)

The operator controls for view control are located at the bottom edge of the graphic area:

- Select the zoom leveling using the drop-down list. You can also enter a value directly into the field of the drop-down list.
- You can also set the zoom level using the slider.
- You can re-focus the window of the graphic area using the icon in the bottom right corner.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate in the overview navigation to the desired objects and display them in the graphic area.

Table area

The table area consists of the following two tabs:

- "Topology overview" tab
- "Topology comparison" tab

The "Topology overview" tab displays the Ethernet or PROFINET modules, their ports and port interconnections in a table. This table corresponds to the network overview table in the network view.

In the "Topology comparison" tab, you identify and minimize differences between the desired and actual topology.

See also

Displaying diagnostics status and comparison status using icons (Page 3305)

Overview of settings for hardware configuration (Page 10)

1.1.8 Inspector window

The properties and parameters shown for the object selected can be edited in the Inspector window.

Structure

The Inspector window consists of the following components:

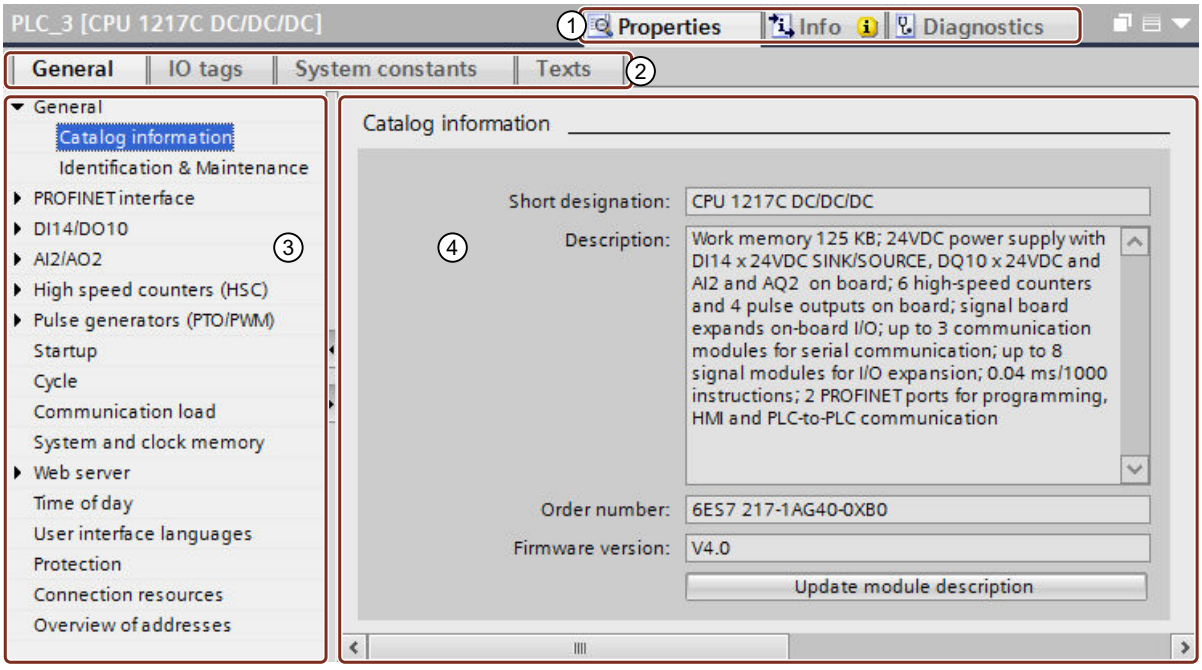

① Switch between various information and work areas

② Switch between various tabs of the information and work areas

③ Navigation between various pieces of information and parameters

④ Display showing the selected information and parameters

Description of functions

The information and parameters in the Inspector window are split into different types of information with different tabs:

- Properties
  This is where you can configure the device. You can make the settings for the hardware, edit tags and system constants and manage project texts. The following tabs are available here:
  - General (see Properties and parameters (Page 61))
  - IO tags (see AUTOHOTSPOT)
  - System constants (see AUTOHOTSPOT)
  - Texts (see AUTOHOTSPOT)
- Info
  In this area, you will find general information on the device in the project, a cross-reference list and messages and notes on compiling the device. The following tabs are available here:
  - General
  - Cross-reference list (see Cross-references in the Inspector window)
  - Compiling (see Information on compiling)
- Diagnostics
  When you go online with the device, you will find the diagnostic information for the device and its connections here. The relevant diagnostic messages are also displayed. The following tabs are available here:
  - Device information (see Hardware diagnostics (Page 3303))
  - Connection information (see Connection diagnostics (Page 3360))
  - Message display (see Message display)

To display the corresponding information and parameters, click in the relevant area.

Display of device properties

The "Properties" area is the most important one for configuring an automation system. The editing options for this area are therefore described in detail below. For information on the other areas, please see the links above to other sections of the information system.

"General" tab

Display the properties and settings of the device or module. Here you can edit the settings and parameters. The left pane of the Inspector window is used for area navigation. Information and parameters are arranged there in groups. If you click on the arrow symbol to the left of the group name, you can expand the group if sub-groups are available. If you select a group or sub-group, the corresponding information and parameters are displayed in the right pane of the Inspector window and can be edited there too.

"IO tags" tab

Display the IO tags of the device, the PLC and the individual modules. You can assign names for the tags, assign the tags to the user-defined tag tables via a drop-down list, and enter comments for the tags. The IO tags are also shown in the PLC tag table.

General | IO tags | System constants | Texts

Type	Address	Name	Tag table	Comment
Bool	%I0.0	Start	Default tag table	
Bool	%I0.1	Stop	Default tag table	
Bool	%I0.2	Check	Default tag table	
Bool	%I0.3			

You can click on the tag symbols in the table to drag and drop tags for use in other editors. For example, you can assign a tag to an instruction in the programming editor or copy a tag

General | IO tags | System constants | Texts

Type	Typ	Address	Tag table

General | IO tags | System constants | Texts

Type	Typ	Address	Tag table
X23-08/15	Bool	%I8.0	Default tag table
X23-08/16	Bool	%I8.1	Default tag table
X23-08/17	Bool	%I8.2	Tag table_1
	Bool	%I8.3	
	Bool	%I8.4	

lso click on the name of a tag above or below. A new tag is number is added to the name a number, the number is

General | IO tags | System constants | Texts

Type	Typ	Address	Tag table
X23-08/15	Bool	%I8.0	Default tag table
X23-08/16	Bool	%I8.1	Default tag table
X23-08/17	Bool	%I8.2	Tag table_1
X23-08/18	Bool	%I8.3	Tag table_1
X23-08/19	Bool	%I8.4	Tag table_1
	Bool	%I8.5	

new tags for the tag table or the drop-down list and drag new tags in the corresponding

If existing tags are affected by these actions, a dialog appears to ask whether the affected tags should be overwritten or whether new tags should only be inserted in empty fields. In the latter case, this means that only three tags are added to an area with nine fields and six existing tags; the existing tags are not changed.

General | IO tags | System constants | Texts

Show hardware system constant

Name	Type	Hardware identi.	Used by	Comment
Local~MC	Hw_SubModule	51	PLC_4	
Local~Common	Hw_SubModule	50	PLC_4	
Local~Device	Hw_Device	32	PLC_4	

With I-slaves and I-devices, the system constants of the DP master or IO controller relating to the I-slave or I-device are also displayed.

If you click on the link in the column in which the use of the constant is displayed, the device view of the corresponding device opens. The system constants are also shown in the PLC tag table.

You can click on the constant symbols in the table to drag and drop system constants to use in other editors. For example, you can assign a system constant to an instruction in the programming editor or create a system constant as a user constant in another device.

"Texts" tab

Display the reference language and specify the text source for the project texts. Please see the rest of the information system.

See also

Overview of hardware and network editor (Page 9)

Addressing modules (Page 1377)

Inspector window: Displaying UDTs (Page 26)

1.1.9 Inspector window: Displaying UDTs

Under "Properties > IO tags", the Inspector window displays self-defined data structures comprising multiple elements. These UDT (User Defined Structure) PLC data types are displayed differently from simple IO tags.

Display of UDTs in the I/O tag table

Structured PLC tags that occupy the address space of a module are displayed along with the data items they contain in combined columns. Occupied address areas are indicated by vertical lines and individual addresses by small diamonds.

The following symbols are used for display in line with IO module channel address assignment by the address area of the UDTs:

Symbol	Meaning
	Start of a data item address area covering several channels. The address area of the entire channel is used.
	Part of a data item address area covering several channels. The address area of the entire channel is used.
	End of a data item address area covering several channels. The address area of the entire channel is used.
	Complete address area of a data item over the address area of a channel used in full. Example: Only the input word of a channel is completely used for a data item.
	Start of a data item address area covering several channels. Only part of the address area of the channel is used.
	End of a data item address area covering several channels. Only part of the address area of the channel is used.
	Complete address area of a data item. Only the first part of the channel address is used. Example: Only the first byte of a word is assigned.
	Complete address area of a data item. Only the second part of the channel address is used. Example: Only the second byte of a word is assigned.
	Single address of a data item for a channel that only has one address. Example: Individual bit access to the channel address of a digital input.
	Single address of a data item for a channel that has multiple addresses. Example: Individual bit access to the channel address of an analog input.

Application example

The figure below shows a tag table with a UDT "Motor" PLC data type and its data items "Start", "Stop" and "Dummy":

		Name	Data type	Address
1		Start	Bool	%I2.0
2		Stop	Bool	%I2.1
3		ErrorCode	Byte	%IB3
4		Motor	"MotorUDT"	%I0.0
5		Dummy	Word	%IW0
6		Start	Bool	%I2.0
7		Stop	Bool	%I2.1
8		Status	Array[0..7] of B...	%I4.0
9		Status[0]	Bool	%I4.0
10		Status[1]	Bool	%I4.1

For a DI32 input module with input addresses I0.0 to I3.7, you see the combined column "Motor ("MotorUDT")" under the IO tags in the Inspector window. This combined column contains a column for the address area of the UDT "Motor" and a column for the data items contained in the UDT: "Motor.Dummy", "Motor.Start" and "Motor.Stop". The "Status" data item only starts at address I4.0 and is not displayed in the IO tags for this input module, because the address area of the 32 digital channels only runs from I0.0 to I3.7. The tag "ErrorCode (Byte)" is not part of the UDT. It is therefore not displayed in the combined column under UDT "Motor", but is instead shown in a separate column:

General | IO tags | System constants | Texts

Type	Address	Name	Tag table	Motor ("MotorUDT")		ErrorCode (Byte)
	Bool	%I0.0		Motor	Motor.Dummy	
	Bool	%I0.1				
	Bool	%I0.2				
	⋮	⋮		⋮	⋮	
	Bool	%I1.7				
Start	Bool	%I2.0	Default tag table		Motor.Start	
Stop	Bool	%I2.1	Default tag table		Motor.Stop	
	Bool	%I2.2				
	⋮	⋮		⋮		
	Bool	%I2.7				
	Bool	%I3.0				
	Bool	%I3.1				
	Bool	%I3.2				
	⋮	⋮		⋮		⋮
	Bool	%I3.7				

The UDT "Motor" in this module occupies an address area from I0.0 to I3.7. Within this address area, the "Dummy" tag occupies the address area from I0.0 to I1.7, and the two tags "Start"

and "Stop" occupy the single addresses I2.0 and I2.1. The "ErrorCode (Byte)" tag that is not part of the UDToccupies addresses I3.0 to I3.7 in its own column.

Note

The table of IO tags only shows the data in the address area of the selected device; in this case the address area of the input module DI32. If another input module with at least 16 channels is connected, the display of the address assignment is continued by the UDT with the extended address area: The address area of the UDT "Motor" runs to address I5.7, and the status data item it contains covers addresses I4.0 to I5.7. The "Status" data item occupies single addresses for individual status bits from I4.0 to I4.7. An additional column is added to the combined column of the UDT to display the status bits.

See also

Inspector window (Page 23)

1.1.10 Hardware catalog

The "Hardware catalog" task card gives you easy access to a wide range of hardware components.

Structure

The "Hardware catalog" task card consists of the following panes:

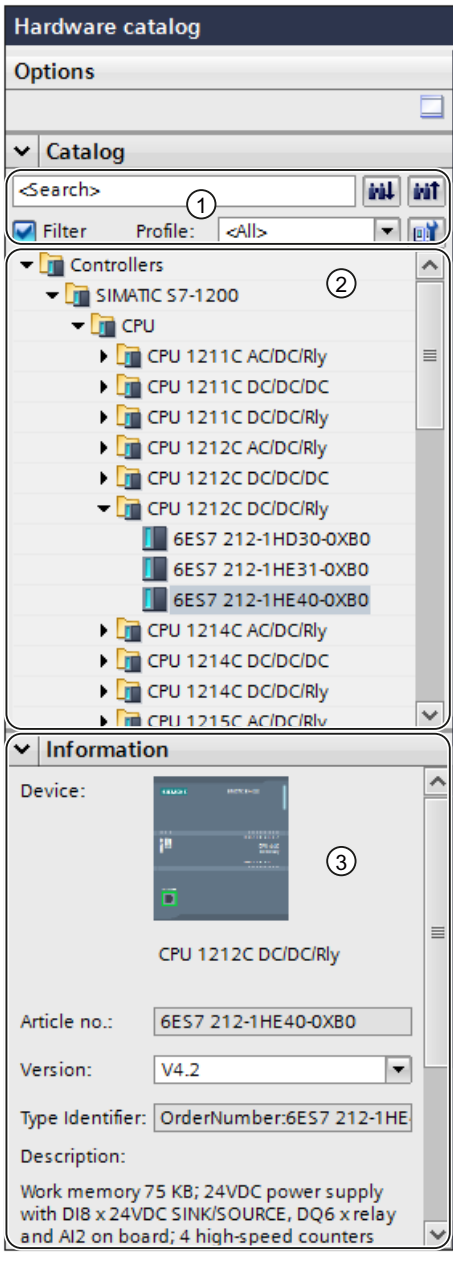

① "Catalog" pane, search and filter function

② "Catalog" pane, component selection

③ "Information" pane

Search and filter function

The search and filter functions of the "Catalog" pane make it easy to search for particular hardware components. After entering the search text, you can start a search in the upwards or downwards direction from the current catalog position.

You can limit the display of the hardware components to certain criteria using the filter function. For example, you can limit the display to objects that you can also place within the current context or which contain certain functions. Objects that can be used in the current context include, for example, interconnectable objects in the network view or only modules compatible with the device in the device view.

You can create hardware profiles with which the objects displayed in the hardware catalog are limited. For example, you could create a hardware profile in which only certain controllers can be seen and another profile in which only certain devices of the distributed I/O are included. You can switch between the created hardware profiles.

Component selection

The component selection in the "Catalog" pane contains the installed hardware components in a tree structure. You can move the devices or modules you want from the catalog to the graphic work area of the device or network view.

Installed hardware components without a license are grayed out. You cannot use non-licensed hardware components.

Hardware components belonging to various components groups thematically are partially implemented as linked objects. When you click on such linked hardware components, a catalog tree opens in which you can find the appropriate hardware components.

Information

The "Information" pane contains detailed information on the object selected from the catalog:

- Schematic representation
- Name
- Article number
- Version number
- Description
- Type Identifier (only when activated)

Note

Type Identifier

You can activate the display of the "Type Identifier" attribute in the "Options > Settings > Hardware configuration" menu. If the Type Identifier is activated, hardware objects for use in external development environments can be uniquely identified. The Type Identifier is displayed in the "Information" pane in the hardware catalog.

See also

Browsing the hardware catalog (Page 31)

Overview of hardware and network editor (Page 9)

1.1.11 Browsing the hardware catalog

Introduction

Use the "Hardware catalog" task card to select the hardware components you want for a hardware configuration. Use the hardware catalog to select the interconnectable hardware components in the network and topology view and to select the modules you want in the device view.

Context filter

You can use the "Filter" option of the hardware catalog to restrict the number of displayed hardware components and the number of hardware components that can be found by searching.

When you enable the filter, components available in the hardware catalog are displayed depending on the selected view and the selected device. If you switch between the various views, the view of the filter objects is adapted to the current context. If the filter is not enabled, the entire hardware catalog is displayed in all views.

Example: With the filter enabled in the device view of an S7-1200 CPU, only S7-1200 CPUs and matching modules are displayed in the hardware catalog.

Search options

You can use the search function to search for specific entries in the hardware catalog. Note the following rules when entering search terms:

- No distinction is made between upper and lower case text.
- Dashes and blanks are ignored during the search.
- The search function considers parts of a search term.
- Multiple search terms must be separated by a space.
- The asterisk "*" is used as a wildcard.

Note

Use of wildcards in the hardware catalog

In search queries, the asterisk "*" is used as a wildcard. A search for "1A**0", for example, also return hits with "1AD30" or "1AE40". The hardware catalog also contains modules with wildcards in their article numbers. The wildcards in the various article numbers can be either asterisks or the lowercase letters "x", "y" or "z". If you search for the specific article number "3RW4 422-1BC34", for example, your search will also return articles with the number "3RW4 422-*BC**" and "3RW4 4xx-1By3z".

You start the search from an object highlighted in the hardware catalog and either search upwards or downwards.

Icon	Meaning
	Search down
	Search up

Browsing the hardware catalog

If you want to browse the hardware catalog, proceed as follows:

1. Click in the entry field of the search function
2. Enter a search term. The search includes the following elements:
   - Name of device or module
   - Article number (MLFB)
   - Description in "Information" pane
3. Click on either the "Search down" or "Search up" button.

Note

To ensure the right search direction, note which point you have marked in the hardware catalog. To browse the entire catalog, click on the topmost object of the hardware catalog and start the search once you have entered the search term by clicking "Search down".

The first match with the search term found is displayed as the result. For more search results, again click on the "Search down" or "Search up" button.

Observe the context filter of the hardware catalog. If this is selected, the search in the hardware catalog is restricted to the displayed inserted hardware components.

See also

Hardware catalog (Page 28)

1.1.12 Enabling product support

For each device in the hardware catalog, you have the option of displaying additional information that is stored in the Siemens Industry Online Support. By default, the function is disabled. Instructions for enabling the function are given below.

Requirement

The TIA Portal has access to the Internet.

Procedure

To enable access to Siemens Industry Online Support, follow these steps:

1. In the "Options" menu, select the "Settings" command.
2. Open the "Hardware configuration" group in the area navigation.
3. Select the "Via Internet" check box.

Result

You can access product support, FAQs and manuals in the hardware catalog via the shortcut menu for the module.

See also

Displaying product support for hardware components (Page 33)

Overview of settings for hardware configuration (Page 10)

1.1.13 Displaying product support for hardware components

In the hardware catalog, you have direct access to information that is stored for each module in Siemens Industry Online Support. You can jump directly to the following pages in Siemens Industry Online Support:

- Information on product support
- FAQs
- Manuals

Requirement

- You have access to the Internet.
- Access to Product Support is enabled in the settings of the TIA Portal.
  For information on how to enable the function, refer to section "Enabling product support (Page 33)".

Procedure

To display the information for a particular module in Siemens Industry Online Support, follow these steps:

1. Navigate to the required module in the hardware catalog.
2. Right-click the module.
3. Select one of the following entries in the shortcut menu:
   - Information on product support
   - FAQs
   - Manuals

Result

The default browser set in the operating system is opened and the relevant page in the Siemens Industry Online Support is loaded.

See also

Enabling product support (Page 33)

Overview of settings for hardware configuration (Page 10)

1.1.14 Printing hardware and network configurations

Printout of hardware and network configurations

You can print out the following elements of the hardware and network view as part of the project documentation:

- Graphic network view
- Network overview table
- Graphic device view
- The device overview table
- The parameters of the object currently selected in the editor

Printout of editor content

If you start a printout within an opened editor and no module is selected, the content of the editor is always printed. This includes a graphic representation of the editor as well as the table for the editor. You can adapt the scope of the printout. You can specify whether only the graphic view, the table or both together are to be printed. Read section "Printing: Changing settings (Page 36)" for more on this.

If the graphic is larger than the page layout you have selected, the printout is continued on the next page. No content is lost this way. Alternatively, you can change the zoom level of the

graphic representation to fit the printout on one page. The printout is always made in the currently selected zoom setting.

To check that all content fits on one page, you can either use the print preview or activate the page break preview. When page break preview is activated, dashed lines are displayed within the graphic editor at the location where the page break is later made.

Printing very large tables

If a table is larger than the print area and therefore cannot be fully printed, the content of the table is not printed as a table, but instead as pairs between value and key.

Example:

Object name	Property 1	Property 2
Object A	Value A1	Value A2
Object B	Value B1	Value B2

In this case, the printout has the following appearance:

Object A

Property 1: Value A1

Property 2: Value A2

Object B

Property 1: Value B1

Property 2: Value B2

You can also preset this as a template so that tables are always printed as pairs between the key and the value. Read section "AUTOHOTSPOT" for more on this.

Printing module parameters

Parameters of selected modules are printed out along with the current value settings in text form. All parameters from corresponding modules are also printed. For example, if you have selected a CPU, the parameters of an inserted signal board, if present, are printed as well.

You can determine the scope of the module parameters to be printed. In the "Print" dialog, select whether all properties and parameters of a module are to be printed or whether to use the compact printout. If you select the compact form, only the entries in the "General" area of the module properties are printed. Comments on modules, as well as the author and module description, are excluded. In compact mode, the following module parameters are therefore printed, for example:

- Module specifications
  Name, module slot, short description, article number, firmware version
- Name of the PROFINET interface
- Subnet specifications
  Name of the subnet, ID of the S7 subnet

See also

Printing: Changing settings (Page 36)

Printing: Enabling page break preview (Page 36)

1.1.15 Printing: Changing settings

Changing the scope of the printout

When printing from an editor, you can specify whether both graphics and tables are to be printed or just one of the two. Both are printed by default.

Procedure

To change the scope of the printout, proceed as follows:

1. In the "Options" menu, select the "Settings" command.
2. In the area navigation, open the "Print settings" parameter group under "General".
3. Scroll to the "Hardware configuration" group.
4. Select or clear the "Active graphic view" check box, depending on whether you want to print the graphics of the network and device view as well.
5. Select or clear the "Active table" check box, depending on whether you want to print the table for the editor as well.

See also

Printing hardware and network configurations (Page 34)

1.1.16 Printing: Enabling page break preview

You can activate the page break preview for the printout in the graphic editor. If this option is activated, dashed lines are shown within the graphic editor at the locations where page breaks are later made during printout.

Procedure

Proceed as follows to activate the page break preview:

1. Select the graphic area of the corresponding view.
2. Click on the "Show page break" symbol in the toolbar of the graphic editor.
   Dashed lines are displayed within the graphic editor at the location a page break is later made.

3. To modify the frame layout, select the "Print" command in the "Project" menu.
4. To disable page break preview, click again on the "Show page break" symbol in the toolbar of the graphic editor.

1.1.17 Keyboard operation: Navigation in the editor

You can use shortcut keys in the network and device view to navigate between the components of the hardware and network editor and its objects.

Navigating between elements and functions

Function	Shortcut keys
Switch to the next lower selection level You can for example, use <Return> to switch from a selected rack to the lower selection level of the devices and modules that are snapped onto it. If a device is selected, you can use <Return> to switch to the lower selection level of the interfaces that are displayed on the device.	<Return>
Switch to the next higher selection level You can use <Esc>, for example, to switch from a selected interface to the higher selection level of the devices and modules. If a device is selected, you can use <Esc> to switch to the higher selection level of the rack.	<Esc>
Navigation between objects in the current selection level You can use the arrow keys to switch between the objects within a current selection level. To change the selection level, use the <Return> or <Esc> keys.	<Up arrow> <Down arrow> <Right arrow> <Left arrow>
Switches to the device view	<Ctrl+Shift+D>
Switches to the network view	<Ctrl+Shift+N>
Switches to the topology view	<Ctrl+Shift+T>
Switch between editor elements Use the <Tab> key to switch from one editor element to the next element. Use <Shift +Tab> to switch to the previous element. You can switch, for example, between the graphical view, Speedy Splitter, table view or underlying tabs.	<Tab> <Shift+Tab>
Switch between tabs Use the <Ctrl+Tab> keys to switch from one tab to the next tab on the right. Use <Ctrl +Shift+Tab> to switch to the next tab to the left. You can use these keys, for example, to switch between the device view, the network view and the topology view.	<Ctrl+Tab> <Ctrl+Shift+Tab>

Opening elements and functions

Function	Shortcut keys
Opening the online and diagnostics view When a device is selected, <Ctrl+D> opens the online and diagnostics view for the selected device.	<Ctrl+D>
Opening the download to device dialog When a device is selected, <Ctrl+L> opens the advanced download dialog.	<Ctrl+L>

Function	Shortcut keys
Add new device <Ctrl+N> opens the dialog for adding a new device.	<Ctrl+N>
Opens the "Hardware catalog" task card	<Ctrl+Shift+C>
Opens "Online Tools" task card	<Ctrl+Shift+O>

1.1.18 Keyboard operation: Editing objects

You can execute some of the functions of the network and device view directly with a combination of keyboard and mouse in the hardware and network editor. The keyboard operation in tables corresponds to standard characteristics. Here you find the keyboard operation for the graphic work area of the network and device view.

General keyboard operation

Function	Shortcut keys
Zoom in on view in frame Drag a frame in the graphical view in order to correspondingly change the size of the view.	<Ctrl+Space> + pressed mouse button
Move view Move the mouse pointer in order to move the view.	<Space> + pressed mouse button
Cancel current operation	<Esc>
Separate connection Use <Esc> or a double-click to exit connection mode when dragging a connection.	<Esc> or double-click
Zoom in graphic view The enlargement or reduction depends on the direction of rotation.	<Ctrl> + turn mouse wheel

Selected objects

Function	Shortcut keys
Select object	Mouse click
Cut an object The selected object is copied to the clipboard and deleted from the graphical view.	<Ctrl+X>
Copy object The selected object is copied to the clipboard.	<Ctrl+C>
Paste object The object from the clipboard is inserted into the selection.	<Ctrl+V>
Delete selected object	<Del>
Select several objects 1 You can add several objects to the selected objects by clicking on them individually. Alternatively, you can use <Shift> + pressed mouse key to drag a frame around the objects that are to be selected.	<Shift> + click

Function	Shortcut keys
Select several objects 2 You can add several objects to the selected objects by clicking on them individually. Alternatively, you can use <Shift> + pressed mouse key to drag a frame around the objects that are to be selected. When holding down the <Ctrl> key, you can use a mouse click to deselect selected objects.	<Ctrl> + click
Move selection When the mouse button is pressed, you can drag devices or modules to allowed slots on a rack.	Mouse button pressed
Copy selection Using <Ctrl> + pressed mouse button you can drag devices and modules to allowed slots on a rack. This copies the devices or modules.	<Ctrl> + pressed mouse button

1.2 Configuring devices

1.2.1 Introduction to configuring hardware

To set up an automation system, you will need to configure, assign parameters and interlink the individual hardware components. The work needed for this is undertaken in the device and network view.

Configuring

"Configuring" is understood to mean arranging, positioning, and networking devices and modules within the device or network view. Racks are represented symbolically. Just like "real" racks, they allow you to plug in a defined number of modules.

An address is automatically assigned to each module. The addresses can be subsequently modified.

When the automation system is started, the CPU compares the configuration that is preset by the software with the actual configuration of the system. Possible errors can be detected and reported straight away.

Assigning parameters

"Assigning parameters" is understood to mean the setting of properties of the components used. Parameter assignment is carried out for hardware components and for data exchange settings:

- Properties of modules with assignable parameters
- Settings for data exchange between components

The parameters are loaded to the CPU and transferred to the corresponding modules when the CPU starts up. Modules can be replaced with ease since the assigned parameters are automatically loaded into the new module during startup.

Adjusting the hardware to the project requirements

You need to configure hardware if you want to set up, expand or change an automation project. To do this, add hardware components to your configuration, connect them to existing components, and adapt the hardware properties to the tasks.

The properties of the automation systems and modules are preset in such a way that in many cases there is no need for additional parameter assignment. Parameter assignment is however needed in the following cases:

- You want to change the default parameter settings of a module.
- You want to use special functions.
- You want to configure communication connections.

1.2.2 Adding a device to the hardware configuration

Introduction

There are various ways of adding a connectable device from the hardware configuration in the network view and the topology view:

- Command "Add new device" in the project tree
- Double-click device in hardware catalog
- Drag-and-drop from the hardware catalog in network view or in topology view:
  - Text entry from the "Catalog" pane
  - Preview graphic from the "Information" pane
- "Add > Device" command from menu bar in network view or topology view
- Shortcut menu of a device in the hardware catalog for copying and pasting

A suitable rack is created along with the new device. The selected device is inserted at the first permitted slot of the rack.

Regardless of the method selected, the added device is visible in the project tree and the network view of the hardware and network editor.

Adding device using the project tree

To use the project tree to add a device to the hardware configuration, proceed as follows:

1. Click on the command "Add new device" in the project tree.
   The "Add new device" dialog box opens.
2. Display the required device in the tree structure:
   - Go to required device in the tree structure.
   - Enter a device name in the entry field.
3. Select the required device from the tree.
   More information about the device presently selected is displayed on the right-side of the dialog box.
4. If necessary, set the firmware version using the drop-down list in the dialog box.
5. Select the "Open device view" check box if you want to change to the device view immediately after adding the device.
   There you can immediately continue with device configuration and equipping the rack.
6. Click on "OK" to add the device selected.
   The dialog box closes.

Adding device from the hardware catalog

To add a device to the hardware configuration using the hardware catalog, proceed as follows:

1. Open the network view or the topology view.
2. Open the hardware catalog.

3. Go to the required device in the hardware catalog.
4. Click on the chosen device to select it.
5. If necessary, set the firmware version using the drop-down list in the hardware catalog.
6. Drag the device to the network view or the topology view.

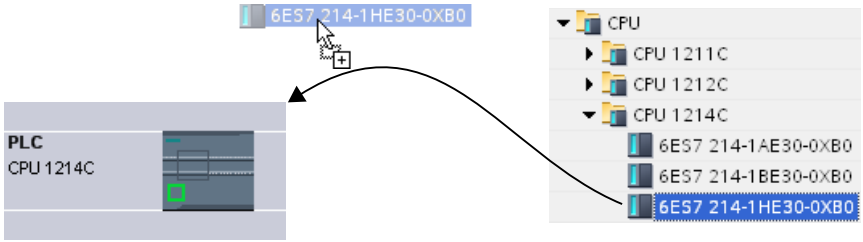

You have now placed the device in the network view or in the topology view. The displayed rectangle (in other words "Station") symbolizes the plugged device together with its rack and any lower-level modules. Double-click on the device or station to open the device view and view the new rack and inserted device. In the next steps, you can configure the device in the device view and equip the rack with modules.

See also

Network view (Page 11)

Adding an unspecified CPU (Page 42)

Topology view (Page 20)

1.2.3 Adding an unspecified CPU

Introduction

If you have not yet selected a CPU but have already started programming or want to use an existing program, you have the option of using an unspecified CPU. You can also adjust some settings with unspecified CPUs. The setting options are restricted to parameters that all CPUs of the same CPU family have in common.

Adding an unspecified CPU in the portal view

To add an unspecified CPU in the portal view, follow these steps:

1. Now, click one of the following options:
   – "Devices & networks > Add new device"
   – "PLC programming" > "Device" button
2. For a device family, select an unspecified CPU from the tree structure of the "Add new device" dialog.
3. Click on "Add".

An unspecified CPU is added and the device view for this CPU is opened.

Further options for adding unspecified CPUs

In the project view, you can add unspecified CPUs like specified CPUs:

- Using the "Add new device" button in the project tree
- In the "Hardware catalog" task card

You can also use these methods to add multiple unspecified CPUs.

Specifying unspecified CPUs

You have two options for specifying unspecified CPUs:

- Use drag-and-drop to assign an existing CPU from the hardware catalog to an unspecified CPU by means of module replacement (Page 56).
- Select a selected, unspecified CPU and then the menu command "Online > Hardware detection" and assign a CPU identified online. For this purpose, you assign an IP address using the "Add address for PG/PC" button.

Note

If you want to go online after the hardware detection, you have to first download the detected configuration to your project; otherwise, an error may occur due to inconsistent configurations. After hardware detection, the article numbers of the CPU in the project and the actually existing CPU are identical, but their parameters are not. The parameters of the CPU in the project have the default values; the parameters of the actually existing CPU have the values set by you.

After module replacement, F-CPUs behave as a CPU without failsafe functionality. In contrast to F-CPUs from the hardware catalog, after module replacement with an unspecified CPU or a hardware recognition, the F capability is turned off and the protection level is set to "Full access (no protection)".

See also

Selecting a CPU (Page 43)

Adding a device to the hardware configuration (Page 41)

Configuring an IO device through hardware detection (Page 2862)

1.2.4 Selecting a CPU

Introduction

Select a CPU from the hardware catalog and place it, together with a rack, in the network view. On this device drag the desired modules from the hardware catalog; they are arranged automatically on the rack.

Selecting the components in the hardware catalog

Each hardware component is displayed as a folder in the hardware catalog. When you open this folder, you see the different versions of the selected hardware component with its respective article numbers.

There will be an example of how to set up a CPU with a rack in network view.

Requirement

- The hardware catalog is open.
- You must be in the network view.

Procedure

To select a CPU from the hardware catalog, proceed as follows:

1. In the hardware catalog navigate to the folder with the desired CPUs.
2. Open the folder with the desired CPU type; you will see all article numbers for the selected CPU type.
3. Click on a CPU article number to get information about the selected CPU in the "Information" pane.

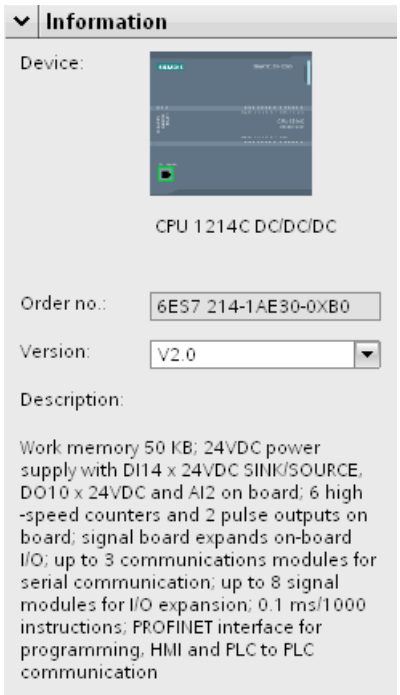

4. Set up the CPU and a rack. You have the following options:
   - Use drag-and-drop to drag the CPU from the hardware catalog into network view.
   - Use Copy & Paste to copy the CPU to the network view.
   - Double-click the CPU entry in the hardware catalog.

See also

Browsing the hardware catalog (Page 31)

Adding a device to the hardware configuration (Page 41)

Rack: Inserting a module (Page 48)

Rack: Basics (Page 46)

Adding an unspecified CPU (Page 42)

1.2.5 Selecting fail-safe modules

Introduction

Fail-safe automation systems (F systems) are used to control processes with a safe state that can be achieved immediately on shutdown. F systems control processes that can be shut down immediately without endangering human beings or the environment.

Selecting the F components in the hardware catalog

Each hardware component is displayed as a folder in the hardware catalog. Fail-safe hardware components are displayed with yellow symbols in the hardware catalog in contrast to the blue hardware components that are not fail-safe.

The following figure shows an excerpt from the hardware catalog with an F-CPU (yellow).

Based on this color coding, you can identify F components in the hardware catalog simply and quickly.

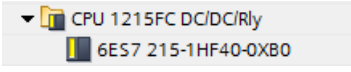

Display of the F components in the hardware and network editor

If you use fail-safe devices and "normal" devices in the graphic views of the hardware and network editor, you can recognize them simply by the yellow color-coding of the devices.

The following figure shows two CPUs. The F CPU is shown in yellow:

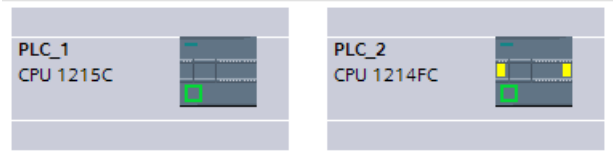

Fail-safe modules in racks or devices can also be recognized in the device view due to their yellow color coding. Since modules that are not capable of networking are not displayed in the network or topology view, devices with F components without networking capability are represented by a special symbol in the network and topology view.

The following figure shows a "normal" CPU and an F-CPU in the network view that both contain F components:

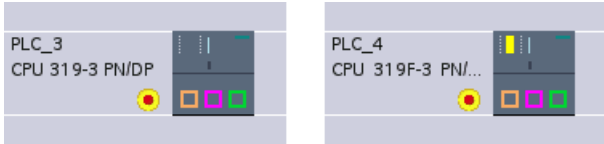

Based on the color coding of the fail-safe devices and modules, you can also identify the F components quickly in the graphic views of the hardware and network editor.

1.2.6 Rack: Basics

Introduction

To assign modules to a device, you need a rack, for example a mounting rail. Secure the modules on the rack and connect these via the backplane bus with the CPU, a power supply or other modules.

Creating a rack

If you insert a device in the network view, a station and a rack suitable for the device selected are created automatically. The rack and slots available are displayed in the device view. The number of slots available again depends on the type of device used.

Rack structure

A rack always contains the device that has been inserted in the network view. The device is permanently assigned a slot which will depend on the type of device in question. There are additional slots on the right of the device and, if necessary, on left of the device; slot numbers are located above slots in which devices are plugged.

A corresponding short description is displayed above the plugged devices and modules. You show or hide this short description via the toolbar under "View" with the command "Display module titles" or the corresponding symbol in the toolbar of the device view (Page 14).

Symbol	Meaning
	Show module titles

When modules are selected in the hardware catalog, all the slots permitted for this module are marked. This allows you to see immediately the slot into which the selected module can be inserted.

In the following screenshot, a signal module has been selected in the hardware catalog for a partially filled S7-1200 rack:

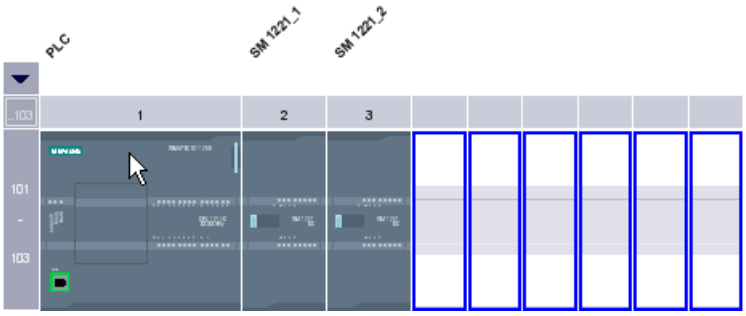

Since slots 101-103 are reserved for communications modules, only the other free slots are shown as available slots.

You can expand and collapse the front group of slots using an arrow symbol above the expandable slot. When the group of slots is collapsed, the first and last of the group's slot numbers are displayed.

The following figure shows the expanded slot group:

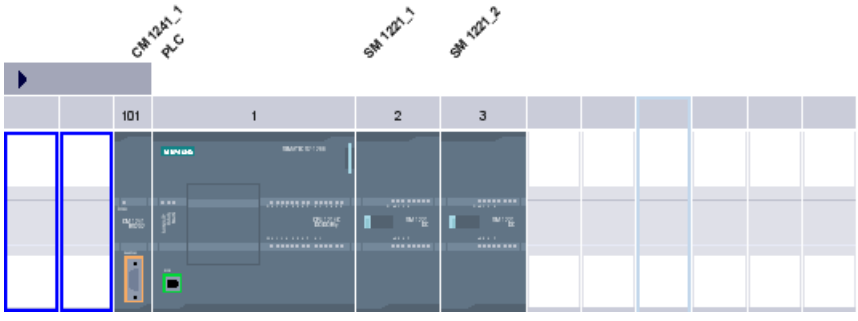

Groups of slots into which modules have already been plugged cannot be collapsed.

Multiple selection of modules and slots

There are various ways of selecting several modules or slots:

- By pressing <Shift> or <Ctrl>, you can select several modules or slots at the same time.
- Click outside the rack and then hold the mouse button and drag a frame to include the modules or slots you want to select.

1.2.7 Rack: General slot rules

Introduction

Specific slot rules apply to each automation system and module.

If you select a module from the hardware catalog in the device view, all possible slots for the module selected are marked in the rack. You can only drag modules to marked slots.

If you insert, move or swap a module, the slot rules are also applied.

Consistency

Some slot rules depend on how the environment is configured. This means that you can sometimes plug modules into the rack although this would result in inconsistencies at the current time. If you change the configuration, for example by selecting different modules or module parameter settings, you can make the configuration consistent again.

In cases where inserting a module results in an inconsistency that can be corrected, this will be permitted. A consistency check is run when the configuration is compiled. Inconsistencies are displayed as alarms in the Inspector window under "Info". Based on the result of the consistency check, you can revise your configuration and make it consistent again.

Rules for arranging modules

The following rules apply generally to modules in racks:

- You can only plug modules into a rack.
- You can only plug interface modules into a module.
- You can only use modules of the same product or system family in one rack.

There are also other special rules for some modules:

- Can only be inserted in certain slots
- Insertion depends on other modules, CPUs or settings
- Limited number of uses in a rack

1.2.8 Rack: Inserting a module

Introduction

Once you have added devices from the hardware catalog to your configuration in network view, you can add modules to the devices. There are various ways of adding a module to a rack in the device view:

- If there is an available valid slot, double-click a module in the hardware catalog.
- Use drag-and-drop to move the module from the hardware catalog to an available valid slot in the graphic or table area:
  - Text entry from the "Catalog" pane
  - Preview graphic from the "Information" pane
- Select "Copy" in the shortcut menu for a module in the hardware catalog and then select "Paste" in the shortcut menu on an available valid slot in the graphic or table area.

To access the device view from the network view, double-click a device or station in the network view or select the Device view tab. The device view contains an illustration of the device

selected within a rack. The graphic illustration of the rack in the software corresponds to the real structure, i.e. you can see the same number of slots as exist in the real structure.

Note

You can also move a module to a rack in the network view. The filter function for the hardware catalog must be deactivated in this instance. The module is automatically plugged into a free and permitted slot. If there are no slots available, the module will be moved to the area of unplugged modules (Page 19).

Equipping a rack

Arrange the modules on a rack according to the applicable slot rules.

After a module has been inserted in a rack with an already inserted CPU, the address areas are checked automatically so that addresses are not assigned twice. After it has been inserted, each module then has one valid address range. To do so, DP slaves and IO devices must be networked with a CPU via the corresponding DP master or IO system.

Requirements

- You are in the device view.
- The hardware catalog is open.

Adding module from the hardware catalog

How to insert a module from the hardware catalog into a rack is illustrated based on the example of a signal module. To do this, follow these steps:

1. Go to the required module board in the hardware catalog.

   Note

   If you activate the filter function of the hardware catalog, only those modules which match the selected device type will be displayed.

2. Select the chosen module.
3. If necessary, set the firmware version using the drop-down list in the hardware catalog.
4. Drag the signal module to a free slot in the rack.

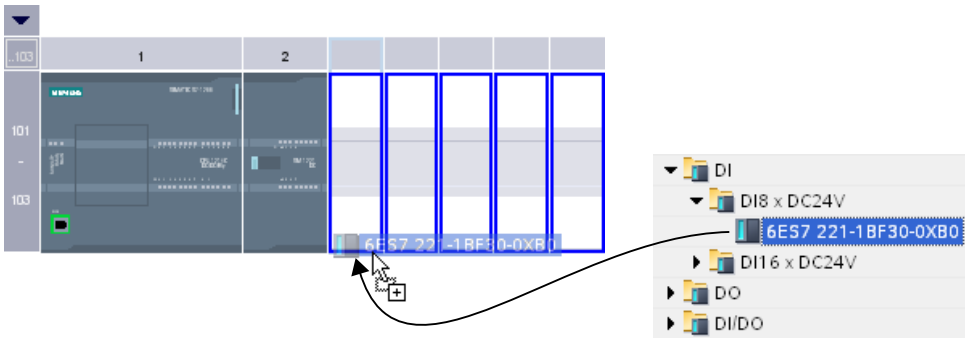

You have now inserted the digital signal module in a slot in the rack. Repeat these steps with the other modules.

The name of the module is displayed above the inserted modules. You can activate or deactivate module labeling in the menu bar with "View > Show module labels".

Inserting module

You can also drag modules and drop them between modules that have already been inserted. To do this, drag a module above and between the two existing modules while holding down the mouse button.

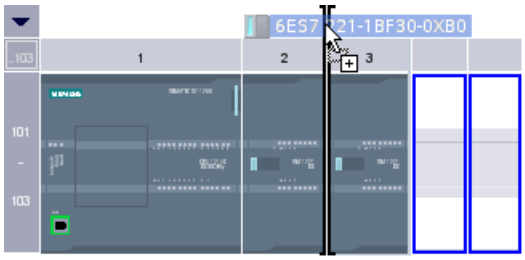

A mouse pointer appears. When you release the mouse button, all modules plugged to the right of the pointer are moved one slot to the right. Any redundant modules are moved to the area of unplugged modules. The new module is plugged at the point of the freed up slot.

See also

Device view (Page 14)

Device view: Area for unplugged modules (Page 19)

Rack: General slot rules (Page 47)

1.2.9 Select hardware component

You can select hardware components and additional objects of the graphic views by clicking the left mouse button. You can also select multiple objects by dragging the mouse while keeping the left mouse button pressed or with the combination of left-click and the <Shift> or <Ctrl> keys.

You can select the following objects in the graphic views: Devices (with CPU, rack, modules), CPUs, CPs, modules, subnets, connections and nodes.

Rules

You can select objects in the graphic views as follows:

- Left-click on object: The object is selected.
- Left-click and <Ctrl> on an object: The object is added to the current selection.

- Left-click and <Shift> on an object: All objects between the currently selected object and the previously selected objects are added to the selection.
- Drag selection frame around multiple objects while pressing left mouse button: All objects completely enclosed by the frame are added to the selection.

Multiple selection is only possible for objects of the same category:

- Category 1: Devices, subnets, connections, port interconnections
- Category 2: CPUs, CPs, interface modules / module
- Category 3: Interfaces, ports, nodes

Behavior for multiple selection with objects from different category:

- Selection frame: Only objects of the respective highest category will become part of the multiple selection
- Left-click with <Shift> or <Ctrl>: The object of a new category that was selected last remains selected; the previously selected objects are dropped from the selection.

Note

Multiple selection of different types of objects

Only the same object categories can be selected in case of multiple selection. For example, multiple CPUs or multiple networks can be selected. However, it is not possible to first select a device in a multiple selection and then only add the CPU of a second device or the interface of a third device to the multiple selection.

Multiple selection of devices with the help of the selection frame

To select multiple devices in the network view by dragging a selection frame, follow these steps:

1. Press the left mouse button and draw a frame around all devices that you want to include in your selection:

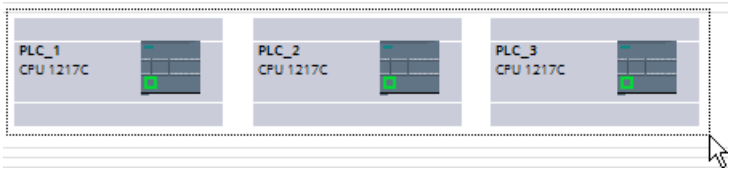

2. Release the mouse button. All devices within the drawn frame are now part of the multiple selection.

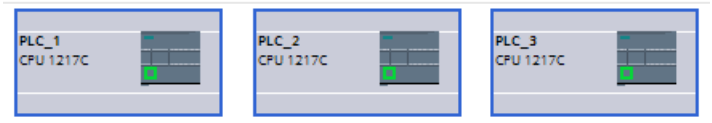

In the example above, the selection frame was drawn around entire devices. If you draw the frame so that it does not enclose entire devices but only the CPUs contained therein, only the CPUs and not the entire device is included in the multiple selection. If you draw the selection frame so that it partially encloses entire devices and partly only CPUs, the highest category of the selected objects is decisive and only the entire devices are added to the multiple selection.

Multiple selection of CPUs with the help of the <Shift> key

Instead of the multiple selection with a selection frame, you can also make the multiple selection with the <Shift> key. You select two objects with this key and all objects in between are included in the multiple selection.

To select multiple CPUs in the topology view with the <Shift> key, follow these steps:

1. Select the first CPU of your multiple selection with the left mouse button:

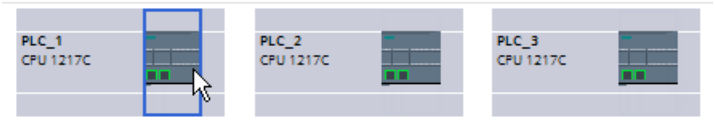

2. Select the last CPU of your multiple selection with the left mouse button while pressing the <Shift> key. The two selected CPUs behave like the margins of a selection frame indicated here by the orange line:

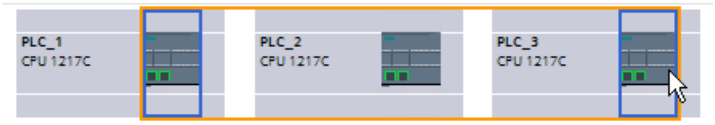

3. All CPUs between the first and the last CPU selected are included in the multiple selection:

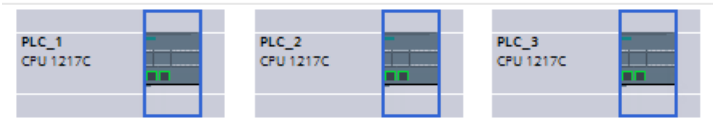

In the example above, only CPUs were selected. If you select entire devices instead, the devices between the first and the second device selected are included in the multiple selection. If you select an entire device as well as a CPU, the category of the object selected last is decisive for the category of all selected objects.

Multiple selection of devices with the help of the <Ctrl> and <Shift> keys

You can also make a multiple selection with the two keys <Ctrl> and <Shift>. You select multiple objects with a left mouse click and the <Ctrl> key. If you then select an object with a left mouse click and the <Shift> key, all objects between the previously selected objects and the last selected object are included in the multiple selection.

To select multiple devices in the device view with the <Shift> and <Ctrl> keys, follow these steps:

1. Use the left mouse button and press the <Ctrl> key to select multiple individual devices:

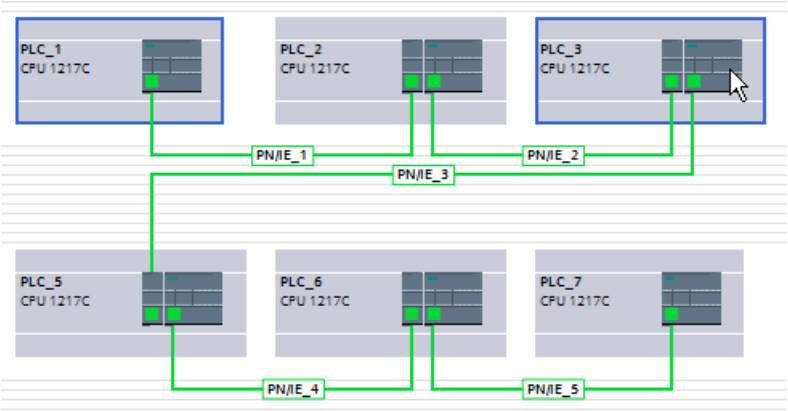

2. Select the last device of your multiple selection with the left mouse button while pressing the <Shift> key. The already selected devices behave like the margins of a selection frame indicated here by the orange line:

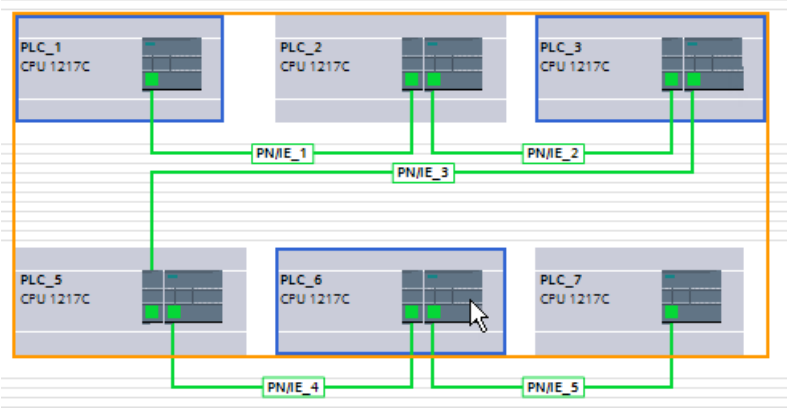

3. All devices of the virtual selection frame and the subnets in the selection frame are now included in the multiple selection.

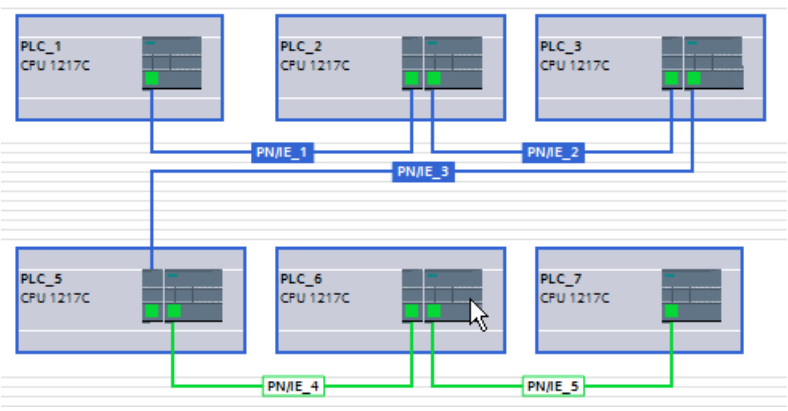

Only the same object categories are selected. In the example above this includes the subnets in addition to the corresponding devices because they were located within the virtual selection

frame and belong to the same category as the devices. The subnets PN/IE_4 and PN/IE_5 were not located in the virtual selection frame and therefore are not part of the multiple selection.

1.2.10 Copying a hardware component

You can copy hardware components in the device or network view. Copied hardware components are stored on a clipboard and can be pasted at another point from this clipboard. Copied stations are pasted as new stations in the network view. Copied devices and modules can be pasted into existing racks in the network and device view.

Rules

- Single objects as well as several objects can be copied at the same time.
- Modules inserted in the rack and in the area of unplugged modules can be copied.
- You can only copy devices and modules to free and valid slots in keeping with the slot rules.
- Racks with a CPU inserted cannot be copied individually, but only as complete units along with all inserted hardware components.

Procedure

Proceed as follows to copy a hardware component:

1. Select the hardware components you want to copy.
   - Device view: Select the module in a rack or put it in the area of unplugged modules.
   - Network or topology view: Select the station or the relevant hardware component from the network view.
   - Project tree: Select the station or module.
2. Select "Copy" from the shortcut menu or press <Ctrl+C>.
   If the "Copy" menu item is unavailable, your selection contains at least one component that cannot be copied.
3. Select the location at which the content of the clipboard is to be pasted.
   - In the device view select a free slot in the rack or area of unplugged modules.
   - In the network or topology view select a station where you want to insert devices or modules.
4. Select "Paste" from the shortcut menu or press <Ctrl+V>.
   - If you paste the copied object without selection in the network or topology view with <Ctrl +V>, the copied object is pasted to the right of the original object.
   - If you paste the copied object without selection in the network or topology view using the shortcut menu, it is placed at the location of the mouse pointer.

   If the "Paste" menu item is unavailable, the clipboard is empty or contains at least one component that cannot be pasted at this point.

The copied object is pasted at the chosen point.

If you have selected a station in the network or topology view to paste a module, the module is inserted into the first free and valid slot. If no free, valid slots are available, the object is inserted in the area of unplugged modules.

Note

You can also copy a module from one device to another:

To do so, copy a module in the hardware and network editor, select a different device in the network or topology view or the drop down list of the device view and insert the module.

You can insert the copied object directly in a slot or place it in the area of unplugged modules in the device view. If you add the copied object in the network view of a device or a station, it will be inserted in the first available slot.

If there is no slot available for the object, it is automatically placed in the area of unplugged modules (Page 19).

Note

You can use <Ctrl> and drag-and-drop to directly copy a selected hardware component.

See also

Keyboard operation: Editing objects (Page 38)

1.2.11 Moving a hardware component

You can move hardware components in the device or network view.

Rules

- You can move devices and modules from the rack and the area for unplugged modules taking the slot rules into consideration.
- CPs can be moved in the network view. The CP is plugged in a free and valid slot in the target device. If there are no free slots available, the CP to be inserted is moved to the area for unplugged modules.
- In the network view, CPU and slave head modules can be moved between the devices; depending on CPU type also within the rack.

Note

Moved CPs are disconnected from their network but keep their network parameters and address. If you reconnect the CP to the network and its address has been assigned, use a dialog to assign a new unique address to the CP.

Procedure

Proceed as follows to move a hardware component:

1. Select the hardware component you want to move.
   - Device view: Select the module in a rack or put it in the area of unplugged modules.
   - Network view: Select the hardware component of relevance to the network view.
2. Select "Cut" from the shortcut menu or press <Ctrl+X>.
   If the "Cut" menu item is unavailable, your selection contains at least one component that cannot be cut.
3. Select the location to which the cut object is to be moved.
   - Device view: Select a free slot in the rack or area of unplugged modules.
   - Network view: Select a station where you want to insert devices or modules.
4. Select "Paste" from the shortcut menu or press <Ctrl+V>.
   If the "Paste" menu item is unavailable, the clipboard is empty or contains at least one component that cannot be pasted at this point.

The selected hardware component is moved to the target. If the hardware component being moved is a networked object, it is disconnected from the network.

Note

You can use drag-and-drop to directly move a selected hardware component.

See also

Keyboard operation: Editing objects (Page 38)

1.2.12 Replacing a hardware component

You can replace hardware components with others. This, for example, allows you to replace unspecified CPUs (Page 42) with available CPUs from the hardware catalog.

Rules

You can only replace hardware components if they support module replacement and if the two components are compatible.

Procedure

To replace one module with another, proceed as follows:

1. Select the module you want to replace.
2. Open the shortcut menu:
   - If the "Replace device" entry is enabled, the module can be replaced.
   - If the "Replace device" entry is disabled, a module cannot be replaced.
3. Click on "Replace device" in the shortcut menu. The "Replace device" dialog box appears.
4. Under "New device" in the tree structure, select the module with which you want to replace your current module.
5. Click "OK".
6. Compile the hardware and software.

The existing module is replaced by the new one.

As an alternative, you can take a module by dragging it from the hardware catalog to the module you are replacing. If the module can be replaced by the selected module, this is indicated by the mouse pointer symbol.

1.2.13 Renaming hardware component

You can conveniently edit the names of hardware components in all three views (network view, device view, topology view) directly in the graphic area as well as in the corresponding table areas or in the Inspector window.

Rules

You can rename the hardware components as follows:

- In the graphic view after selection of the displayed name.
- In the table view in the table cell with the name that is to be changed.
- In the Inspector window under "Properties > General" for the selected hardware component.

Procedure

To rename a hardware component in the graphic area of a view, follow these steps:

For the device view:

1. Activate the display of the module names with the corresponding button in the toolbar.
2. Select the hardware component or its name.
3. You have two options:
   - Press <F2>.
   - Click on the previously selected name.
4. Edit the name.

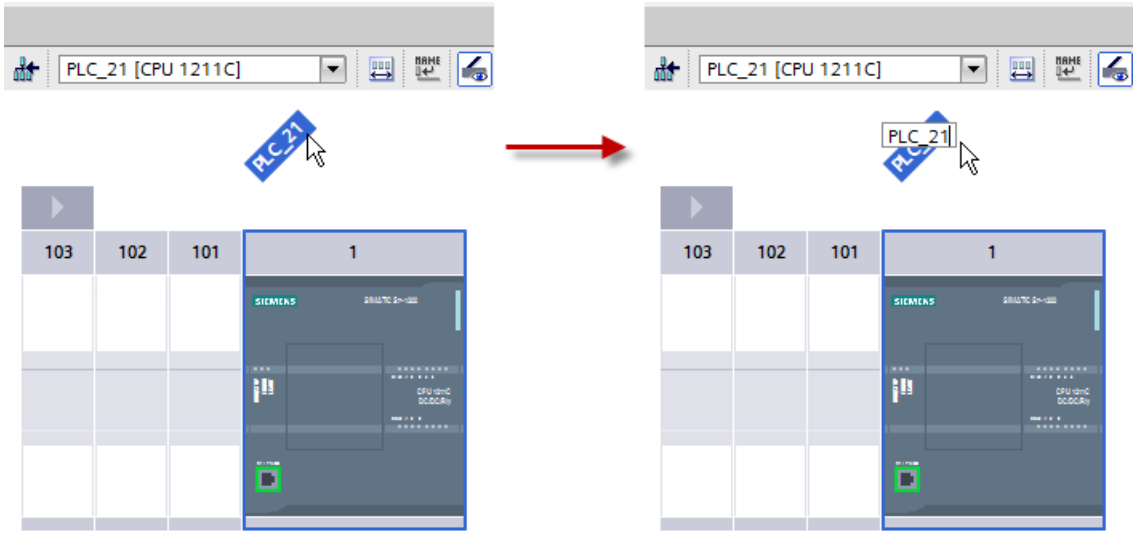

For the network and topology view:

1. Select the hardware component or its name.
2. You have three options:
   - Press <F2>.
   - Click on the previously selected name.
   - Select "Rename" from the shortcut menu.

For editing names in the table area, select the cell with the name you want to edit in the table. Then click on the name once again so that you can edit the names directly in the cell.

For editing names in the Inspector window, select the hardware component whose name you want to edit in the graphic area. The relevant parameters are then displayed in the Inspector window. Now you can edit the names under "Properties > General".

See also

Network view (Page 11)

Device view (Page 14)

Topology view (Page 20)

Inspector window (Page 23)

1.2.14 Deleting a hardware component

There are various ways of deleting hardware components. Deleted hardware components are removed from the system and assigned addresses made available again.

Rules

- CPUs or modules from the rack and from the area of unplugged modules can be deleted.
- When a rack is deleted in the device view, the plugged hardware components are moved to the area of unplugged modules.

Procedure

Proceed as follows to delete a hardware component:

1. Select the hardware components you want to delete.
   - Network view: Select devices or network relevant hardware components in the graphic view or in the network view.
   - Device view: In the graphic view or device overview, select racks or modules in racks or in the area of unplugged components.
   - Topology view: Select devices or hardware components with Ethernet interfaces in the graphic view or in the topology view.
   - Project tree: Select devices or individual hardware components from the tree structure.
2. Select "Delete" from the shortcut menu or press <Del>.
   If the "Delete" menu item is unavailable, your selection contains at least one component that cannot be deleted.

The selected hardware components are deleted.

Note

Deleting hardware components may result in inconsistencies in the project, such as violation of slot rules. Inconsistencies are reported during the consistency check. Correct the inconsistencies by taking appropriate action, for example, ensuring compliance with the slot rules.

See also

Keyboard operation: Editing objects (Page 38)

1.2.15 Changing or updating hardware components

Explanation of terms

You can update the firmware version of a hardware component to a new version (in as far as available) or possibly use new functions of the hardware components by updating the module description.

The terms "Module description" and "Firmware version" are explained in more detail in the following section.

- Module description: The specific version of the configuration software from which the technical description of the module stems.
  Example: V11.0.0.0
- Firmware version: The version of the firmware of the module whose parameters are assigned offline
  Example: V2.0

Requirement

- You have created a device configuration.
- You have installed an update or an optional package at a later date. As a result of this installation, the module description of at least one module type was updated in the hardware catalog, whereby the new version of the module description is incompatible with the previous version.
- You have used such modules in your device configuration and want to use the modified or added properties.

Procedure

Perform the following step for each affected module type.

1. Select the affected module in the device view.
2. In the Inspector window, go to "Properties > General > Catalog Information". Click the "Update module description" button there.
3. In the query that then appears, specify whether you want to update the module description only for the selected module or for all modules of this type in the current project.

Result

The selected modules are replaced by the same modules with updated module description in the current project.

In which cases is it unnecessary to update the module description?

An updating of the module description is not necessary in the following cases:

- You do not want to use the modified or added properties of the modules.
- When you open an existing project with a version of the configuration software that is more recent than the version with which you created the project, the system automatically performs a project conversion, for example, from TIA Portal V12 to V13. In this case, all older module descriptions are automatically adapted.

1.2.16 Editing properties and parameters

Once you have inserted hardware components in your rack, you can edit their default properties, such as names, parameters or addresses, in the network view, device view or topology view.

Requirement

You are in the network view, device view or topology view.

Note

You can edit the relevant properties and parameters in the different views in the graphic view as well as in the table view. It depends on the view which parameters you can edit. In the graphic network view, you have access to the network-related hardware components and the station. You can access modules and hardware components not displayed in the graphic network view using the table network view. In the topology view and in the device view, you have access to additional parameters regarding the topology or device access.

Procedure

To change the properties and parameters of the hardware components in the device view, follow these steps:

1. In the graphic view, select the CPU, module, rack or interface you want to edit.
2. Edit the settings for the selected object:
   - In the graphic area you can, for example, change the object names with activated module labeling.
   - Use the table area to edit addresses and names, for example.
   - In the Inspector window additional setting possibilities are available in "Properties".

Note that modules can only be fully parameterized if they are assigned to a CPU. Therefore, PROFIBUS or PROFINET interfaces modules must first be networked with the CPU or a centrally inserted communication module so that they form a master system or IO system. Only then is it possible, for example, to edit the addresses of the distributed components that are inserted.

Example of changing settings

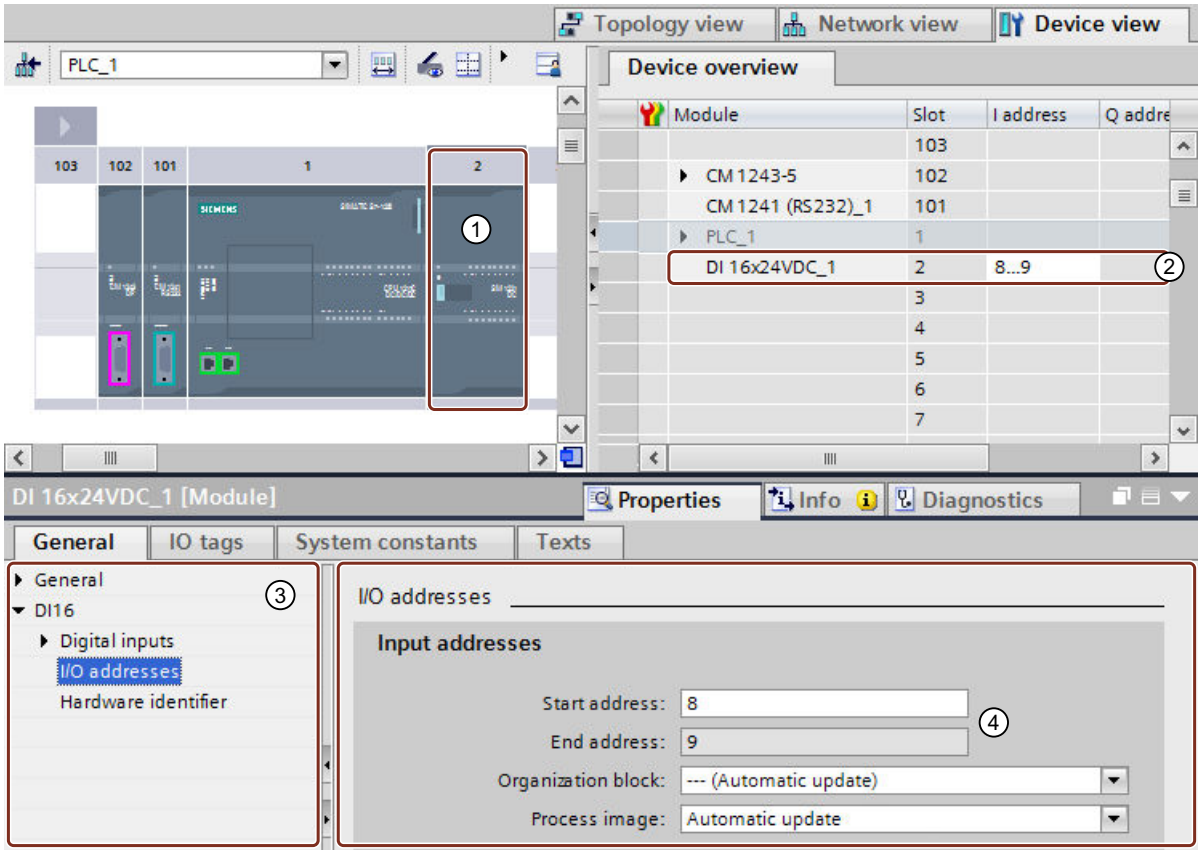

① Selection of a module
② Editing option for addresses in the device overview
③ Selection options in the inspector window
④ Editing option for addresses in the inspector window

See also

Inspector window (Page 23)

1.2.17 Input and output addresses in the address overview

Introduction

The currently used input and output addresses can be displayed in the address overview in a table form. The address overview can be found in the Inspector window under "Properties" of the CPU.

Design of the address overview

With the different check boxes, you can set which objects should be displayed in the address overview:

- Inputs: Display of the input addresses
- Outputs: Display of the output addresses
- Address gaps: Display of open address spaces
- Slot: Display of the slot number

The following information is typically shown in the address overview:

Table header	Meaning
Type	Type of the address range. Indicates whether the area is an input address area or an output address area. • I = input address of module with input I/O address space (with length > 0) • I* = input or diagnostics address of modules without input I/O address space (with length = 0) No data (signal statuses or measured values) is exchanged via these addresses in the user program. Used for reporting diagnostics events of the operating system (e.g. in the start information of error OBs). • Q = output address
Addr. from	Start address of the address range. Indicates the first assigned address of the input or output.
Addr. to	End address in the address range. Indicates the last assigned address of the input or output.
Size	Address length of the address range occupied by the module for the input or output. Examples: • With a start address of 0 and an end address of 1, the address length is 2 bytes. • With a start address of 0 and an end address of 3, the address length is 4 bytes. • With a start address of 0.1 and an end address of 0.2, the address length is 2 bits. • If the start address and end address are identical, 0 bytes is displayed.
Module	Module using the address area.
Rack	Number of the rack on which the hardware component is inserted.
Slot	Number of the slot in the rack in which the hardware component is inserted.
Device name	Name and type of device in which the module is inserted. Examples: • The module is inserted in a PLC with the name "PLC_1" of the type CPU 1516-3 PN/DP, "PLC_1 [CPU 1516-3]" is displayed. • The module is inserted in a head module with the name "IO Device_1" of the type IM 151-3 PN; "IO Device_1 [IM 151-3 PN]" is displayed

Table header	Meaning
Device number	PROFINET device number or PROFIBUS address if the module is connected to a PLC via PROFINET or PROFIBUS. If the module is inserted directly in a PLC, "-" is displayed. Examples: • The module is inserted in a head module that is connected to a PLC via a PROFINET interface; the device number of the Ethernet address is displayed. • The module is inserted in a DP slave that is connected to a PLC via a PROFIBUS interface, the address parameter of the PROFIBUS address is displayed.
Master I/O system	Name of the DP master system or IO system via which the DP slave or IO device is connected to the PLC. The corresponding number of the DP master system or IO system is displayed in square brackets. If the module is inserted directly in a PLC, "-" is displayed. Examples: • The module is inserted in an IO device and is connected to a PLC via a PROFINET IO system with the name "PROFINET IO-System_1" and the number 100. "PROFINET IO-System_1 [100]" is displayed. • The module is inserted in a DP slave and is connected to a PLC via a PROFIBUS DP master system with the name "DP-Mastersystem" and the number 1. "DP-Mastersystem [1]" is displayed.
OB	Organization block that is assigned to the process image. The "Main" cycle OB always refers to the automatic update of the process image. This column is not available for every CPU.
PIP	Process image partition. Indicates whether the address is updated automatically via the cyclic process image or is located in a process image partition that must be triggered explicitly for an update. Refer to the further description of the "Process image partition PIP2" below.

Process image partition PIP

The "PIP" table column shows the assignment of the address to the cyclic process image or to a process image partition (PIP).

For S7-300/400:

- "OB1-PA": The address is assigned to the cyclic process image. The operating system updates this address automatically in each program cycle.
- "PIP x": The address is assigned to the process image partition x (for example PIP 1, no cyclic process image). The operating system updates this PIP when the assigned OB is executed. If this PIP is not assigned to an OB, the operating system does not update this PIP. You have the option to update the PIP yourself with the instructions "UPDAT_PI" and "UPDAT_PO" (for S7-400 and some S7-300 CPUs).

For S7-1200:

- "Automatic update": The address is assigned to the cyclic process image (PIP 0). The operating system updates this address automatically in each program cycle.
- "None": The address not assigned to any process image partition. You access this address directly in the user program (direct I/O access, no process image).

- "PIP x": The operating system updates this PIP when the assigned OB is executed. If this PIP is not assigned to an OB, the operating system does not update this PIP. You have the option to read inputs or write outputs in the user program via direct I/O access. The instructions "UPDAT_PI", "UPDAT_PO", "SYNC_PI" and "SYNC_PO" are not supported for S7-1200.
- "PIP OB servo": The process image partition "PIP OB Servo" is not assigned to an organization block (fixed setting for organization block: "---(None)"). The operating system does not update this PIP or any of the addresses contained in it: You access the addresses directly in the user program (direct I/O access).

For S7-1500:

- "Automatic update": The address is assigned to the cyclic process image (PIP 0). The operating system updates this address automatically in each program cycle.
- "None": The address not assigned to any process image partition. You access this address directly in the user program (direct I/O access, no process image).
- "PIP x" (PIP 1 to PIP 31): The operating system updates PIP x when the assigned OB is executed. If this PIP is not assigned to an OB, the operating system does not update PIP x. You have the option to update the PIP x in the user program with the instructions "UPDAT_PI" and "UPDAT_PO". If PIP x is assigned to an isochronous mode interrupt OB (OB 61 to OB 64), the operating system does not update PIP x. You have the option to update PIP x in your user program with the instructions "SYNC_PI" and "SYNC_PO".
- "PIP OB servo": The process image partition "PIP OB servo" is permanently assigned to the OB "MC-Servo". STEP 7 generates this OB automatically when you create a technology object in the Motion Control area. When the OB is executed, the PIP OB Servo is updated isochronously. All drives and encoders used by Motion Control are assigned to this process image partition.

See also

Specifying input and output addresses (Page 1378)

1.2.18 Distributed I/O in the project tree

Distributed I/O devices in the project tree shown in separate folders. Depending on whether a device was assigned to a PROFINET IO system, a DP master system or no system, the device is displayed as a node or link to the device in one of the folders described below.

"Ungrouped devices" folder in the project

All distributed I/O devices in the project are collected in the "Ungrouped devices" folder.

The icons of the folders and links show the status "assigned" with colored identifiers: PROFINET IO devices are marked green, PROFIBUS DP slaves are marked magenta, other distributed I/O devices are not marked.

The icons of the folders and links show the status "not assigned" with question marks.

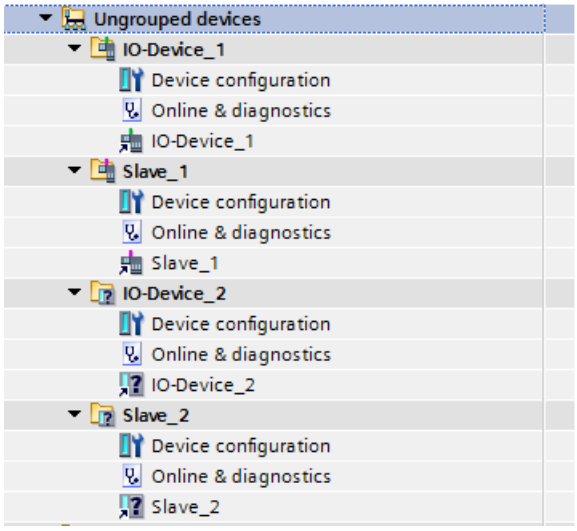

If you have established a connection to an IO device or a DP slave with the "Go online" function, the same diagnostics information (e.g. green check mark) is displayed for the link of a device as for the device itself in the project navigation. For more information, refer to the section "View in the online mode".

It is possible to copy or move distributed I/O devices from the "Ungrouped devices" folder, but not the folder itself, to another position in the project tree. Possible destinations are the project folder or device folder. You can also copy or move distributed I/O devices back to the "Ungrouped devices" folder.

By double-clicking on the link, you go to the device view and can configure the distributed I/O device.

"Unassigned devices" folder in the project

Devices that are not assigned to a distributed I/O system appear as links in the "Unassigned devices" folder.

The icon for the folder and the link of the respective device in the project tree indicate with a question mark that they are not connected to a distributed I/O system. The link is marked by a small arrow.

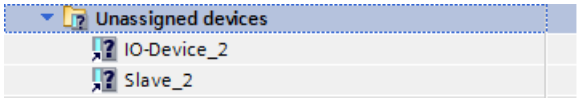

The display of the project tree as a link allows IO devices or DP slaves to be selected individually, for example to select the destination of the link. It remains possible to select device groups.

By double-clicking on the link, you go to the device view and can configure the distributed I/O device.

"Distributed I/O" folder under a device

If you assign an IO device to a PROFINET IO system, this is displayed as a link below the node "Distributed I/O > PROFINET IO system".

Accordingly, a DP slave that is assigned to a DP master system is shown as a link below the node "Distributed I/O > DP master system".

The link references the corresponding IO device or the DP slave in the project tree.

The icon for the node and the link of the respective device in the project tree indicate whether the device is connected via PROFINET (green) or via PROFIBUS (magenta). The link is marked by a small arrow.

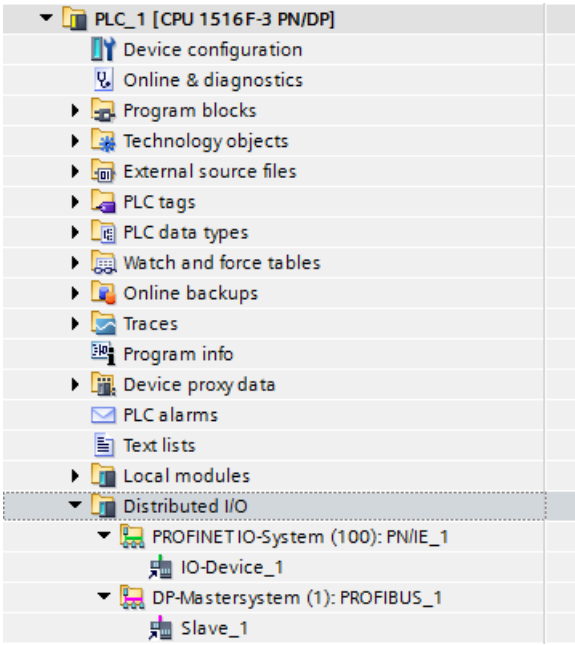

Gateways are shown as a link below the DP master/IO node to which they were assigned.

By double-clicking on the link, you go to the device view and can configure the distributed I/O device.

1.2.19 Comparing devices

1.2.19.1 Basics of device comparison

Function

You have the option of comparing the hardware components of two devices allowing you to identify any differences. You can perform an offline/offline comparison for this purpose. The devices to be compared can be from one project or different projects.

You can compare the central as well as the distributed I/O. The devices to be compared can be assigned either automatically or manually. The automatic assignment of central I/O is based

on the slot number. With the distributed I/O, the automatic assignment can be made according to the following criteria:

- Assignment using the address/HW ID: The assignment is made using the addresses or IDs of the devices. This criterion is suitable for comparing devices within a project.
- Assignment using the name: The assignment is made based on the device names. This criterion is suitable for comparing devices in different projects.

You can either specify the assignment yourself or let the system decide. In the latter case, the system selects the assignment itself depending on the context.

To compare the modules of the distributed I/O, start an offline/offline comparison for the affected distributed stations.

See also

Making a device comparison (Page 68)

1.2.19.2 Making a device comparison

Procedure

Follow the steps below to compare devices:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The comparison editor opens and the selected device is displayed in the left area.
3. Open the "Hardware" tab.
4. Drag an additional device to the drop area in the right-hand pane.
   All existing objects of the selected devices are displayed depending on the settings of the comparison editor in the "Hardware" tab and an automatic comparison is made. You can identify the status of the comparison objects based on the symbols in the compare editor. For submodules, you can find detailed results in the "Property comparison" area.
5. If you want to change the matching criterion, click on the arrow of the "Available matching criteria" button in the toolbar. Then, select the matching criterion you want to use.
6. If you want to make a manual comparison, click the button for switching between automatic and manual comparison above the status area. Then select the objects you want to compare.
   The properties comparison is displayed. You can see the status of the objects based on the symbols.

See also

Basics of device comparison (Page 67)

1.3 Configure networks

1.3.1 Networking devices

1.3.1.1 Communication and networks

Communication between devices

The basis of all types of communication is always a previously configured network. You use the network configuration to organize the required accessibility of the various network nodes for communication. The following requirements apply for the message exchange in the S7 network:

- Assignment of unique S7 topological addresses to all nodes of the network
- Ensure the connection of the interfaces via physical line sections
- Communication of the devices with consistent transmission properties

Definition

Via the network, the various computers and end devices (PC, PG, PLC, AS) are connected together with interface modules, physical cables and suitable software. Networked devices can exchange data with each other. Networking can be implemented via wiring or a wireless network, e.g. via WLAN.

The different forms of the wiring result in different S7 subnets from the perspective of the network. An S7 subnet is part of the overall S7 network. The parameters of the nodes of an S7 subnet (e.g. for PROFIBUS) must be uniform with respect to the physical wiring. A physical S7 subnet includes the bus components and all connected devices. S7 subnets on different physical wiring can be coupled to gateways or routers in the S7 network.

Network configuration

The following steps are required to configure the S7 network:

- Connect devices to an S7 subnet
- Set properties/parameters for each S7 subnet
- Specify the node properties for each networked module (e.g. unique S7 topological address)
- Load the configuration data into the devices to provide interfaces with the settings resulting from the S7 network configuration
- Document S7 network configuration

Creation and configuration of an S7 subnet is supported by the connection configuration for Open User Communication, .

Relation between network configuration and project

S7 subnets are managed with their properties within a project. Properties essentially result from adjustable S7 network parameters and from the number and communication characteristics of the connected nodes.

The devices to be networked must be in the same project.

S7 subnet name and S7 subnet ID

S7 subnets are uniquely identified within the project by an S7 subnet name and an S7 subnet ID. The S7 subnet ID is stored in all components with networkable interfaces. Components can then be clearly assigned to an S7 subnet even after uploading into a project.

S7 subnets are available in two possible versions:

- Physical Ethernet Wiring ("Green Line")
- Physical PROFIBUS connection ("violet line")

Networking options

In the project, you can create and network devices with components capable of communication. The following basic options are available for networking the devices:

- You link the interfaces of the components capable of communication with one another. This creates a new S7 subnet that matches the interface type.
- You connect the interface of the communication-capable devices to a new or an existing S7 subnet.
- You create an Open User Communication connection. An S7 subnet is automatically created between the communication partners through the connection configuration of the Open User Communication.
- You use the graphic connection configuration to configure connections; missing networks are hereby recognized and are created either automatically or via dialog.

Due to different tasks of the devices or due to the expansion of the system, it may be necessary to operate several S7 subnets. These S7 subnets are managed in a project.

1.3.1.2 Networking devices in the network view

Options

In the graphic network view, you have an overview of the subnets of the entire system in the project. You can use the tabular network overview for additional support.

Depending on the starting situation, there are various ways of undertaking configuration to network the interface for a component capable of communication. The procedures are described in the following section:

- Creating an individual subnet
- Creating several subnets at one time
- Connecting two target devices via a new subnet

- Connecting devices to existing subnet
- Selecting an existing subnet from a list
- Automatic networking during the configuration of the connection;
  See also: AUTOHOTSPOT

Possible starting situations are:

- A suitable subnet is not yet available.
- The subnet with which you want to connect the component already exists.

Procedure - creating a single subnet

To create a subnet and to connect it to an interface, proceed as follows:

1. Select the interface of a CPU / a CP.
2. Select the "Create subnet" command in the shortcut menu of the interface.

The selected interface is connected to a new subnet. The inconsistencies possibly resulting from the modified configuration are clarified automatically or per query. In the case of consistency problems, the next free address is preferably used.

The following schematic shows an interface with outgoing line connecting to a subnet:

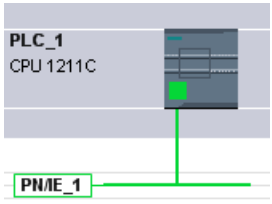

Procedure - creating several subnets at one time

To create several subnets at one time, proceed as follows:

1. Select several interfaces by clicking on them while pressing the <Ctrl> button.
2. Select the "Create subnet" command in the shortcut menu of the interface.

Each selected interface is connected to a new subnet. The inconsistencies possibly resulting from the modified configuration are clarified automatically or per query. In the case of consistency problems, the next free address is preferably used.

The following figure shows multiple subnets created by selecting multiple interfaces:

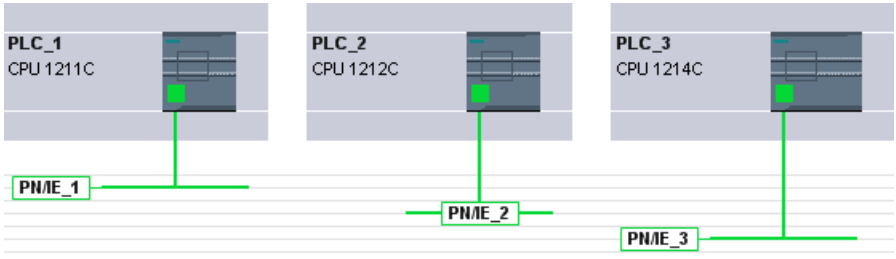

Procedure – Connecting two target devices via a new subnet

To connect an interface with another device via a subnet that does not yet exist, proceed as follows:

1. Position the mouse pointer over the interface for a component capable of communication requiring networking.
2. Click with the left mouse button and hold the button down.
3. Move the mouse pointer.
   The pointer now uses the networking symbol to indicate "Networking" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear when the pointer is on a valid target.

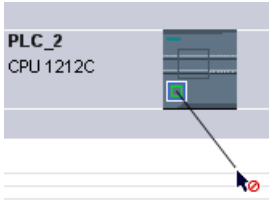

4. Now move the pointer in networking mode onto the interface of the target device. You can either keep the mouse button pressed or release it.
5. Now release the left mouse button or press it again (depending on your previous action).

Note

If you want to exit networking mode beforehand, press <Esc>, right-click or double-click in the background of the network view.

A new subnet is created. The interfaces are now connected via the new subnet. The inconsistencies possibly resulting from the modified configuration are clarified automatically or per query. In the case of consistency problems, the next free address is preferably used.

The following schematic shows two networked devices:

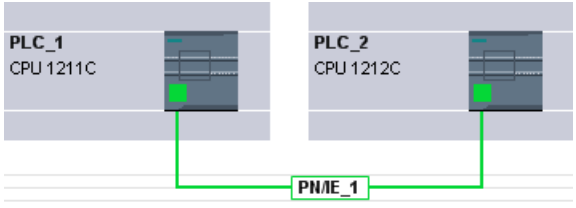

Procedure - Connecting devices to existing subnet

To connect an interface to an existing subnet, proceed as follows:

1. Position the mouse pointer on the interface of a communications-compliant component you want to network or on the existing subnet.
2. Click with the left mouse button and hold the button down.

3. Move the mouse pointer.
   The pointer now uses the networking symbol to indicate "Networking" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear once the pointer is moved to a valid target.
4. Now move the mouse pointer to the existing subnet or to the interface to be networked.
   You can either keep the mouse button pressed or release it.

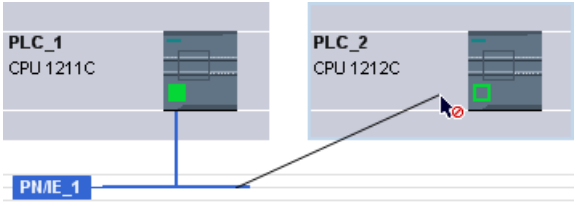

5. Now release the left mouse button or press it again.
   If you want to exit networking mode beforehand, press <Esc> or right-click.

Result:

The interface and selected subnet are now connected. The inconsistencies possibly resulting from the modified configuration are clarified automatically or per query. In the case of consistency problems, the next free address is preferably used.

Procedure - selecting an existing subnet from a list

To link an interface with a subnet that has already been created, proceed as follows:

1. Select the interface of a CPU.
2. Select the "Assign to new subnet" command in the shortcut menu of the interface.
   A list box containing the available subnets appears.
3. Select a subnet from the list.

The interface and selected subnet are now connected. The inconsistencies possibly resulting from the modified configuration are clarified automatically or per query. In the case of consistency problems, the next free address is preferably used.

1.3.1.3 Tabular network overview

Table area of network view

The table area of the network view consists of several tables:

- Network overview: The table for the network overview supplements the graphical network view with the following functions:
  - You obtain detailed information on the structure and parameter settings of the devices.
  - Using the "Subnet" column, you can connect components capable of communication with created subnets.
- Connections: Communication connections are displayed in this table. Some aspects of the connections displayed here can be edited directly in the table. Selected connections are displayed in the Inspector window and can be edited there.

- Relations: Logical connections of the HMI client-server relation are displayed here.
- I/O communication: Device communication via PROFIBUS DP and PROFINET IO is configured and displayed in this table. In the I/O communication table, only those interfaces of a device are displayed that can be used for communication via PROFIBUS DP or PROFINET IO.
- VPN: Properties of the VPN user groups and their assigned security modules are displayed here.

Basic functions for tables

The network overview supports the following basic functions for editing a table:

- Displaying and hiding table columns
  Note: The columns relevant for configuration cannot be hidden.
- Optimizing column width
- Sorting table
- Displaying the meaning of a column, a row or cell using tooltips.

1.3.1.4 Networking devices in the device view

Networking in the device view

In the device view, you can check and set all the parameters of the components belonging to a device and the interfaces in detail. Here you can also assign the interfaces to the subnets created in the project.

Requirements

- The subnet with which you want to connect an interface has already been created.
- If the subnet has not yet been created, change to the network view and perform the networking.

Procedure - connecting to an existing subnet

To connect an interface to an existing subnet, proceed in the device view as follows:

1. Select the entire communications-compliant component or the interface to be networked. The properties of the selected interface or component are displayed in the Inspector window.
2. In the Inspector window, select the parameter group for the selected interface; for example, the "Ethernet addresses" parameter group for a PROFINET interface.
3. Select the subnet to be connected from the "Interface connected with" drop-down list.

The interface and selected subnet are now connected. The inconsistencies possibly resulting from the modified configuration are clarified automatically or per query. In the case of consistency problems, the next free address is preferably used.

Procedure - creating a new subnet

To create a subnet and to connect it to the interface, proceed as follows in the device view:

1. Select the entire communications-compliant component or the interface to be networked. The properties of the selected interface or component are displayed in the Inspector window.
2. In the Inspector window, select the parameter group belonging to the selected interface, for example, the "Ethernet addresses" parameter group for a PROFINET interface.
3. In "Interface connected with", click the "Add new subnet" button.

The interface is connected to a new subnet of the appropriate subnet type. The inconsistencies possibly resulting from the modified configuration are clarified automatically or per query. In the case of consistency problems, the next free address is preferably used.

1.3.1.5 Checking or changing network parameters and interface parameters

Introduction

Communication between networked devices requires the following parameters to be configured:

- Network parameters
  Network parameters identify the network within the system configuration, for example, using a name.
- Interface parameters
  Interface parameters define specific properties of a component capable of communication. Addresses and transmission characteristics are set automatically and are consistent with the network parameters.

Note

Network parameters and interface parameters are usually set during networking such that communication can take place for numerous applications without the parameters having to be changed. The inconsistencies possibly resulting from a modified configuration are clarified automatically or per query. In the case of consistency problems, the next free address is preferably used.

Procedure - checking or changing network parameters

Proceed as follows to check or change network parameters:

1. Enter the network view.
2. Select the subnet from the network view.
   You can see the network parameters in the "Properties" tab in the inspector window.
3. If necessary, check or modify the network parameters in the relevant group of parameters.

Procedure - checking or changing interface parameters

You can check and modify interface parameters in the network and device view.

Proceed as follows to check or change interface parameters:

1. Enter the network view or device view.
2. Select the interface.
   You can see the interface parameters in the "Properties" tab in the inspector window.
3. If necessary, check or modify the interface parameters in the relevant group of parameters.

Procedure - checking or changing interface addresses

You can edit the MPI, PROFIBUS and Ethernet addresses in the graphical network view itself.

Proceed as follows to edit MPI, PROFIBUS and Ethernet addresses in the network view:

1. Enable the display of interface addresses in the network view.
2. Select an MPI, PROFIBUS or Ethernet address to be edited.
3. Click on the selected address field or press [F2] to edit the address in the network view.

1.3.1.6 Changing networkings

Introduction

You can cancel an interface's network connection or assign it to another subnet of the same subnet type.

Consequences

Depending on the version, a distinction should be made between:

- Canceling a network connection for an interface
  The configured parameters for the interface remain unchanged.
- Assigning a network connection to another subnet
  If the addresses in the assigned subnet are not unique, in other words, they already exist, they will be changed automatically to make them unique.

Procedure - disconnecting from a network

Proceed as follows to cancel the network connection for an interface:

1. Select the networked interface.

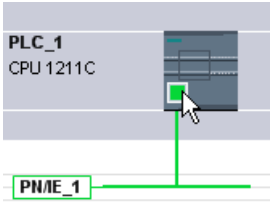

2. Select the "Disconnect from subnet" command in the shortcut menu of the interface.

The network connection is deleted, the interface addresses are, however, not changed.

Configured connections are retained; however these connections are marked red in the connection table because they are not networked. Specified connections remain specified.

See also

Networking devices in the network view (Page 70)

1.3.1.7 Copying, cutting or deleting subnets

Introduction

You can copy subnets as individual objects or copy them along with networked devices or other networks.

For example, you can create complex configurations to be used more than once in different variants within the project with no additional effort.

Effects on the copied subnet

Properties that have to be assigned explicitly within a project are re-assigned appropriately when the copied objects are copied.

For subnets, this means: The subnet ID and name are re-assigned to the copied subnet.

The configured properties are adopted in the copied subnet.

Procedure - copying a subnet

Proceed as follows to copy one or more subnets:

1. Select one or more subnets.
2. In the shortcut menu, select the "Copy" command.
3. Select the "Paste" command in the shortcut menu.

The copied subnets are shown as "orphaned" subnets in the top part of the network view.

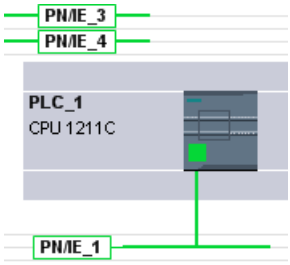

Procedure - copying a subnet with connected devices

To copy one or more subnets with networked devices, proceed as follows:

1. Select one or more subnets with the connected devices, for example by drawing a lasso around them.
2. In the shortcut menu, select the "Copy" command.
3. Select the "Paste" command in the shortcut menu.

Complete copies of the subnets and connected devices are created.

Configured connections are adopted and remain within the copied devices. Connections to devices that have not been copied are interrupted and become unspecified.

1.3.1.8 MPI network configuration

Allocating MPI addresses

Note the following for devices with an MPI interface: All devices of a subnet must have a different device address.

CPUs with MPI address ship with the default MPI address 2. Since you can only use this address once in the MPI subnet, you will have to change the default address in all other CPUs.

The following applies to devices with the article no. 6ES7 3xx-xxxxx-0AB0:

When planning the MPI addresses for several CPUs, you have to fill "MPI address gaps" for FMs and CPs with separate MPI addresses to prevent addresses being assigned twice.

Only when all the modules in a subnet have different addresses and your actual structure matches that of the network configuration produced, should you load the settings across the network.

Rules for assigning the MPI address

- Allocate the MPI addresses in ascending order.
- Reserve MPI address 0 for a programming device.
- You can link up to 126 (addressable) devices with one another in an MPI subnet; up to 8 devices at a transfer speed of 19.2 KB/s.
- All MPI addresses in an MPI subnet must be different.

You will find other rules relating to the structure of a network in the manuals for setting up automation systems.

1.3.1.9 PROFIBUS network configuration

PROFIBUS addresses

Rules for the network configuration

All the devices in a subnet must have a different PROFIBUS address.

Only when all the modules in a subnet have different addresses and your actual structure matches that of the network configuration produced, should you load the settings across the network.

You can connect devices to the PROFIBUS subnet that communicate via configured connections or that belong to a PROFIBUS DP master system.

You can find more information on configuring a DP master system in the following sections.

Requirements

The 121xC CPU is PROFIBUS compatible as of firmware version 2.0.

Rules for assigning PROFIBUS addresses

- Allocate the PROFIBUS addresses in ascending order.
- Reserve the PROFIBUS address "0" for a programming device.
- Allocate a unique PROFIBUS address between 0 and 126 for each device on the PROFIBUS network and/or for each DP master and each DP slave in the PROFIBUS network.
- There are modules with which the smallest address that can be set has to be greater than 1.
- All PROFIBUS addresses of a PROFIBUS subnet must be different.

You will find additional rules relating to the structure of a network in the manuals for setting up automation systems, for example SIMATIC S7-1200.

Note

PROFIBUS address "0"

Reserve PROFIBUS address "0" for a programming device that you will briefly connect up to the PROFIBUS network at a later date for servicing.

See also

What you need to know about PROFIBUS bus parameters (Page 80)

What you need to know about PROFIBUS bus parameters

Matching parameters to one another

The PROFIBUS subnet will only function without problem if the parameters for the bus profile are matched to one another. You should therefore only change the default values if you are familiar with how to configure the bus profile for PROFIBUS.

Note

It may be possible for the bus parameters to be adjusted depending on the bus profile. If the bus parameters cannot be adjusted, they are grayed out. The offline values of the bus parameters are always shown even if you are online and linked to the target system.

The parameters shown apply to the entire PROFIBUS subnet and are briefly explained below.

Activating cyclic distribution of the bus parameters

If the "Enable cyclic distribution of the bus parameters" check box is selected under "Bus parameters" while PROFIBUS subnet is selected in the Inspector window, the bus parameters are transferred cyclically during operation by the modules that support this function. This allows a programming device, for example, to be easily connected to the PROFIBUS in runtime.

You should deactivate this function:

- For a heterogeneous PROFIBUS subnet (or more accurately: when external devices are connected whose protocol uses the DSAP 63 for Multicast)
- When in constant bus cycle time mode (minimize bus cycle)

Bus parameters for the bus profile of PROFIBUS subnets

Bus parameter	Adjustable?	Limit values
Tslot_Init	Yes	Max. Tsdr + 15 <= Tslot_init <= 16.383 t_bit
Max. Tsdr	Yes	35 + 2*Tset + Tqui <= Max. Tsdr <= 1.023 t_bit
Min. Tsdr	Yes	11 t_bit <= Min. Tsdr <= MIN(255 t_bit, ... Max. Tsdr - 1, 34 + 2*Tset + Tqui)
Tset	Yes	1 t_bit <= Tset <= 494 t_bit
Tqui	Yes	0 t_bit <= Tqui <= MIN(31 t_bit, Min. Tsdr - 1)
GAP factor	Yes	1 <= GAP factor <= 100
Retry limit	Yes	1 <= Retry limit <= 15
Tslot	No	---
Tid2	No	Tid2 = Max. Tsdr
Trdy	No	Trdy = Min. Tsdr
Tid1	No	Tid1 = 35 + 2*Tset + Tqui
Ttr	Yes	256 t_bit <= Ttr <= 16.777.960 t_bit

Bus parameter	Adjustable?	Limit values
Ttr typical	No	This time is provided for information only and is not transferred to the nodes.
Response monitoring		10 ms <= response monitoring (watchdog) <= 650 s

If you want to create a customized bus profile, we recommend the following settings:

- Minimum target rotation time (Ttr) = 5000 x HSA (highest PROFIBUS address)
- Minimum response monitoring (watchdog) = 6250 x HSA

Recalculating

You can use the "Recalculate" button to recalculate the parameters.

See also

PROFIBUS addresses (Page 79)

Description of the bus parameters (Page 81)

Description of the bus parameters

Detailed description of PROFIBUS bus parameters

Bus parameter	Meaning
Tslot_Init	The wait-for-reception (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner. If the influence of the line components on message frame run times is entered in the "Cable Configuration" parameter group, these components must also be taken into account. The component is added to the specified Tslot_Init and the total used as Tslot.
Max. Tsdr	The maximum protocol processing time defines the latest time by which the responding node should have replied.
Min. Tsdr	The minimum protocol processing time defines the earliest time by which the responding node may reply.
Tset	The trigger time is the time which may lapse between the reception of a data message frame and the response to it in the node.
Tqui	The modulator quiet time is the time which a sending node needs after the end of the message frame to switch from sending to receiving.
GAP factor	The GAP update factor defines the number of token rotations after which a newly added, active node can be added to the logical token ring.
Retry limit	This parameter defines the maximum number of attempts (message frame repeats) made to reach a node.
Tslot	The wait-for-reception time (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner. If the influence of the bus design components on message frame run times is entered in the "Cable Configuration" parameter group, these components must also be taken into account. The component is added to the specified Tslot_Init and the total used as Tslot.

Bus parameter	Meaning
Tid2	Idle time 2 defines the earliest time by which a sending node may send the next message frame after sending a message frame that has not been acknowledged.
Trdy	The ready time specifies the earliest time by which a sending node may receive a response message frame.
Tid1	Idle time 1 defines the earliest time by which a sending node may send the next message frame after receiving a response.
Ttr	The target rotation time is the maximum time made available for a token rotation. During this time, all active nodes (DP masters etc.) have the right to send (token) once. The difference between the target rotation time and the actual token holding time of a node determines how much time is left over for the other active nodes (programming device, other DP masters etc.) to send message frames.
Ttr typical	The typical data cycle time is the average response time on the bus if all configured slaves are exchanging data with the DP master. None of the slaves report diagnostics and there is no extra message frame traffic with programming devices or other active nodes etc. on the bus.
Response monitoring	The response monitoring time is only needed for PROFIBUS-DP bus systems. It defines the latest time by which a DP slave has to be addressed by its DP master with a new data message frame. If this does not happen, the DP slave assumes that the DP master has failed and resets its outputs to a secure mode.

See also

What you need to know about PROFIBUS bus parameters (Page 80)

Bus profiles with PROFIBUS

Introduction

Depending on the device types connected and protocols used on the PROFIBUS, different profiles are available. The profiles differ in terms of the setting options and calculation of bus parameters. The profiles are explained below.

Devices with different profiles on the same PROFIBUS subnet

The PROFIBUS subnet only functions without problem if the bus parameters of all devices have the same values. For example, if both DP and FMS services are being used on a subnet, the "slower" sets of bus parameters must always be set for all devices, i.e. even the "Universal (DP/FMS)" profile for DP devices.

Profiles and transmission rates

Profiles	Supported transmission speeds in Kbits/s
DP	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000
Standard	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000
Universal (DP-FMS)	9.6 19.2 93.75 187.5 500 1500
Customized	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000

Meaning of profiles

Profile	Meaning
DP	Select the "DP" bus profile when the only devices connected to the PROFIBUS subnet are those which satisfy the requirements of standard EN 50170 Volume 2/3, Part 8-2 PROFIBUS. The bus parameter setting is optimized on these devices. This includes devices with DP master and DP slave interfaces of the SIMATIC S7 and distributed I/Os of other manufacturers.
Standard	Compared to the "DP" profile, the "Standard" profile also offers scope for devices of another project or devices which have not been configured here to be taken into account when calculating the bus parameters. The bus parameters are then calculated following a simple, non-optimized algorithm.
Universal (DP/FMS)	Select the "Universal (DP/FMS)" bus profile when individual devices on the PROFIBUS subnet use the PROFIBUS-FMS service. This includes the following devices for example: • CP 343-5 (SIMATIC S7) • PROFIBUS-FMS devices of other manufacturers As with the "Standard" profile, this profile allows you to take other devices into account when calculating the bus parameters.
Customized	The PROFIBUS subnet will only function without problem if the parameters for the bus profile are matched to one another. Select the "Customized" profile when none of the usual profiles "match" a PROFIBUS device and you need to adapt the bus parameters to your special structure. Information on this can be found in the documentation for the PROFIBUS device. You should only change the default values if you are familiar with how to configure the bus profile for PROFIBUS. Not all combinations that can be theoretically set can be used even with this bus profile. The PROFIBUS standard specifies several parameter limits that depend on other parameters. For example, a responder must not respond (Min Tsdr) before the initiator can receive the message frame (Trdy). These standard specifications are also checked in the "Customized" profile. Tip: The bus parameters last valid on the PROFIBUS subnet are always automatically set as customized. For example, if the "DP" bus profile was valid for the subnet, then the bus parameters for "DP" are set in the "Customized" bus profile. The parameters can be modified on this basis. The monitoring times are not automatically recalculated so that you do not put at risk the consistency of set values, for example with configurations in other configuration tools without realizing that you have done so. You can also have the Ttr monitoring times and target rotation time calculated on the basis of parameters you have set: Click here on the "Recalculate" button.

Note

Both mono-master mode and multi-master mode are possible with all PROFIBUS profiles.

What you need to know about PROFIBUS line configuration

Cable configuration and bus parameters

Information regarding the cable configuration can be taken into consideration when calculating the bus parameters. For this purpose, you must select the "Consider cable configuration" check box in the properties for the PROFIBUS subnet.

The remaining information then depends on the type of cable used; the following settings are available:

- Copper cable
- Fiber-optic cable/optical ring

PROFIBUS line configuration, optical ring

The calculation depends on the OLM types used. The selection is made by activating the check box (multiple activation is possible, at least one OLM type must be selected):

- OLM/P12
- OLM/G12
- OLM/G12-EEC
- OLM/G12-1300

The following bus parameter adjustments are made:

- Configuration of a node not present

  Note

  The following restrictions apply to optical rings, even for passive nodes (DP slaves for example):

  A maximum of HSA-1 nodes may be connected to the PROFIBUS network. With an HSA of 126, addresses 126 and 125 must not be used. A maximum of 125 nodes are possible on the bus (No. 0 to 124).

  If an HSA is less than or equal to 125, the addresses HSA and greater may not be used. The address HSA-1 may not be used.

- Increase the retry value to 3
- Setting of minimum slot time needed for ring mode

  Note

  Short slot time values are needed for OLM/P12, average ones for OLM/G12 and OLM/G12-EEC and long ones for OLM/G12-1300. This results in high performance for small network lengths and average to low performance for average to large network lengths.

PROFIBUS communication load

Communication load - allowing for additional network stations

The bus parameters depend on the volume of communication between the active network nodes. There are differences between cyclic communication (DP) and connection-based, acyclic communication (S7 communication, Send/Receive (FDL), FMS). Unlike DP, the volume and size of communication tasks (communication load) depends on the user program. For this reason, the communication load cannot always be calculated automatically.

To calculate the bus times you can define a network configuration in the "Additional network stations" parameter group that differs from the network configuration.

Taking the profile into account

The network configuration can be defined for the "Standard", "Universal (DP/FMS)", and "User-defined" profiles. Parameters cannot be entered in the "Additional network stations" parameter group for the "DP" profile.

Quantification of the communication load

The following settings are available in order to make allowance for the communication load.

- Information regarding the number of unconfigured network stations;
- Information on the communication load resulting from the user programs for FDL or S7 communication. Here you can selected from the following settings:
  - Low
    Typically used for DP, no great data communication apart from DP.
  - Medium
    Typically used for mixed operations featuring DP and other communication services (such as for S7 communication), when DP has strict time requirements and for average acyclic volumes of communication.
  - High
    For mixed operations featuring DP and other communication services (such as for S7 communication), when DP has loose time requirements and for high acyclic volumes of communication.

1.3.1.10 Configuring Industrial Ethernet

Rules for the network configuration

The Ethernet interfaces of the devices have a default IP address that you can change.

IP address

The IP parameters are visible if the module capable of communication supports the TCP/IP protocol. This is usually the case for all Ethernet modules.

The IP address consists of 4 decimal figures in the range of 0 to 255. The decimal figures are separated from one another by a dot.

Example: 140.80.0.2

The IP address consists of:

- Address of IP subnet
- Address of the device (generally also called host or network node)

Subnet mask

The subnet mask splits these two addresses. It determines which part of the IP address addresses the network and which part of the IP address addresses the device.

The set bits of the subnet mask determine the network part of the IP address.

Example:

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

In the example given for the above IP address, the subnet mask shown here has the following meaning:

The first 2 bytes of the IP address identify the subnet - i.e. 140.80. The last two bytes address the device, i.e. 0.2.

It is generally true that:

- The network address results from AND logic operation of the IP address and subnet mask.
- The device address results from the AND NOT logic operation of the IP address and subnet mask.

Relation between IP address and default subnet mask

An agreement exists relating to the assignment of IP address ranges and so-called "Default subnet masks". The first decimal number (from the left) in the IP address determines the structure of the default subnet mask. It determines the number of "1" values (binary) as follows:

IP address (decimal)	IP address (binary)	Address class	Default subnet mask
0 to 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 to 191	10xxxxxx.xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

Note

Range of values for the first decimal point

A value of between 224 and 255 is also possible for the first decimal number of the IP address (address class D, etc). This is not recommended, however, because there is no address check for these values.

Masking other subnets

You can use the subnet mask to add further structures and form "private" subnets for a subnet that is assigned one of the address classes A, B or C. This is done by setting other lower points of the subnet mask to "1". For each bit set to "1", the number of "private" networks doubles and the number of devices they contain is halved. Externally, the network functions like an individual network as it did previously.

Example:

You have a subnet of address class B (for example IP address 129.80.xxx.xxx) and change the default subnet mask as follows:

Masks	Decimal	Binary
Default subnet mask	255.255.0.0	11111111.11111111.00000000.00000000
Subnet mask	255.255.128.0	11111111.11111111.10000000.00000000

Result:

All devices with addresses between 129.80.001.xxx and 129.80.127.xxx are on one IP subnet, all devices with addresses between 129.80.128.xxx and 129.80.255.xxx are on another IP subnet.

Router

The task of the routers is to connect the IP subnets. If an IP datagram is to be sent to another network, it first has to be conveyed to a router. To make this possible, you have to enter the address of the router for each device in the IP subnet.

The IP address of a device in the subnet and the IP address of the router can only differ at the points at which there is a "0" in the subnet mask.

See also

Settings for interconnecting Ethernet devices (Page 87)

1.3.1.11 Settings for interconnecting Ethernet devices

The following paragraph describes the behavior of STEP 7 during interconnection of PROFINET devices and the effects of the port interconnection on the network view.

Relation between port interconnection and network view

Topology view

In the topology view, you specify the physical interconnection of Ethernet ports. You specifically determine which Ethernet port of a device is to be connected with a specific Ethernet port of another device by means of an Ethernet cable (preset topology).

Example:

You specify that port 1 of the PROFINET interface of the CPU is to be connected to port 2 of the PROFINET interface of device A by means of an Ethernet cable. You also specify the interface for devices with several PROFINET interfaces.

Example:

You specify that port 1 of PROFINET interface X2 of the CPU is to be connected to port 2 of the PROFINET interface of device A by means of an Ethernet cable. The Ethernet ports can be interconnected in a table or graphically.

Network view

In the network view, you specify which devices are to be connected with each other via an Ethernet subnet. You do not specify the Ethernet ports by which the devices are interconnected with each other (that is the task of the port interconnection).

The port interconnection has effects on the network view: When you interconnect Ethernet ports of devices with each other in the topology view, STEP 7 connects the interconnected PROFINET interfaces of the devices in the network view with an Ethernet subnet (green line). However, the course of the green line does not reflect the actual cable routing. You specify the actual wiring in the topology view.

An Ethernet subnet always has a name and an S7 subnet ID. You can set these two values in the subnet properties.

Which Ethernet subnet is used to link interconnected devices?

STEP 7 distinguishes between the following cases:

- Ethernet subnet specified (default subnet).
- No Ethernet subnet specified (no default subnet).

Ethernet subnet specified (default subnet)

The option "Connect devices that are not linked with this subnet in case of port connection" is activated in the properties of an Ethernet subnet (default). This option can be activated for exactly one Ethernet subnet. Activate the option, if necessary, for the Ethernet subnet that is to be continued when interconnecting devices that are not linked. This subnet is referred to as "default subnet" below.

Response of STEP 7:

When you interconnect the ports of two devices that are not linked in the topology view, STEP 7 links these devices with the default subnet.

Example:

Step 1: Create a subnet in the network view at PLC_4, "Connect devices that are not linked with this subnet in case of port connection" option is **enabled**.

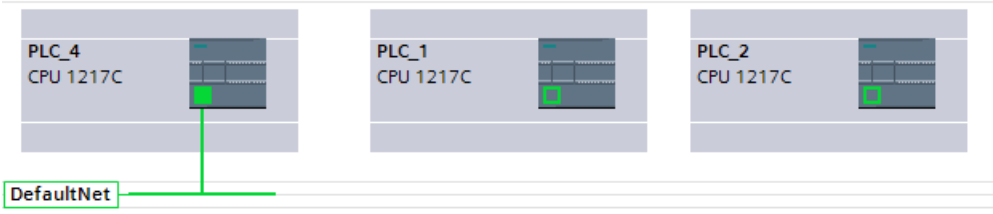

Step 2: Interconnect PLC_1 with PLC_2 (topology view).

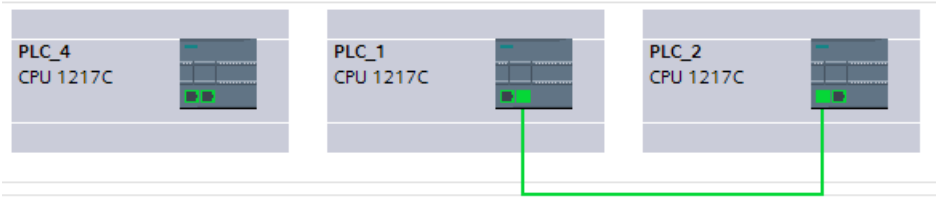

Result: All PLCs are now connected to the same default subnet (network view).

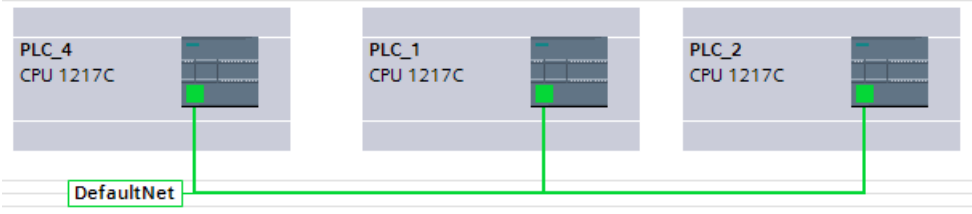

No Ethernet subnet specified (no default subnet).

This is the case when the following conditions are fulfilled:

- A subnet has been added to a PROFINET interface (this interface is referred to as "Interface A" below).
- The option "Connect devices that are not linked with this subnet in case of port connection" is deactivated in the properties of this subnet (no default subnet).
- There is no other Ethernet subnet for which this option is activated.

Response of STEP 7:

- The Ethernet subnet of interface A is only continued when you interconnect a port of interface A with a port of another device.
- When you interconnect ports of other devices that are not linked with each other, STEP 7 creates a new Ethernet subnet.

Example:

Step 1: Create a subnet in the network view at PLC_4, "Connect devices that are not linked with this subnet in case of port connection" option is **disabled**.

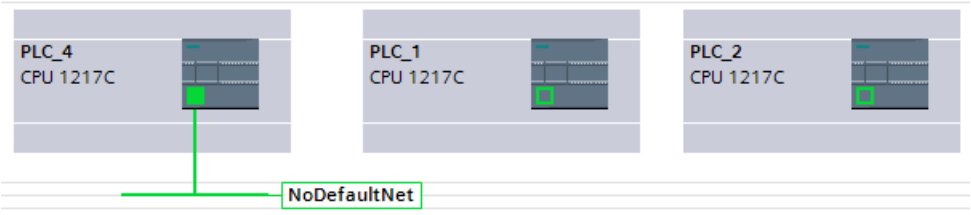

Step 2: Interconnect PLC_1 with PLC_2 (topology view).

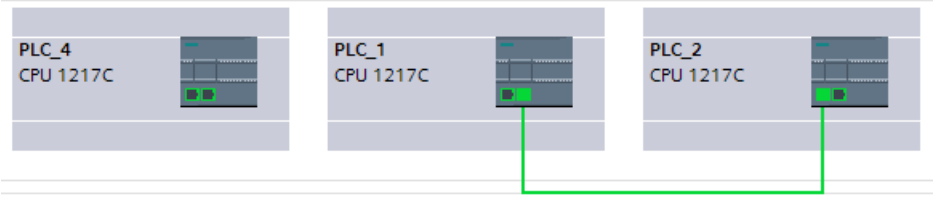

Result: PLC_1 and PLC_2 are connected to a new subnet (network view).

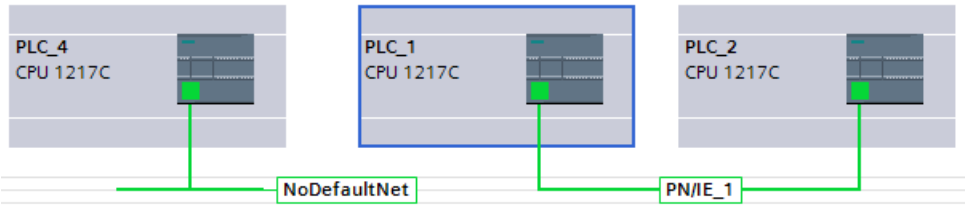

Manual adaptation of the IP addresses

It can occur that STEP 7 does not adapt the IP addresses of the interconnected devices in such a way that the resulting network can be correctly compiled, for example, for devices with multiple PROFINET interfaces. In these situations, you need to adapt the IP addresses of the devices manually.

The following rules apply:

- Devices that are to communicate with each other without a router cannot belong to different IP subnets.
- For devices with multiple PROFINET interfaces, the interfaces must be in different IP subnets.

To change the IP address of a PROFINET interface, perform the following steps:

1. Change to the network view (if not selected already).
2. Left-click on the icon of the PROFINET interface that is not to be part of the IP subnet.
3. Change the subnet part of the IP address in the properties of the PROFINET interface (in the "Ethernet addresses" area).

Example: Manual conversion of the IP subnet part for devices with multiple PROFINET interfaces in a subnet

The IP address is: "192.168.0.1".

The subnet mask is "255.255.255.0".

The first three numbers "192.168.0" form the IP subnet part of the IP address "192.168.0.1".

Change the IP subnet part, for example, to "192.168.1"

See also

Interconnecting ports in the graphic view (Page 296)

Interconnecting ports in the table view (Page 297)

Configuring Industrial Ethernet (Page 85)

1.3.2 Communication via connections

1.3.2.1 Working with connections

Using Open User Communication

Basics of Open User Communication

Introduction

Open User Communication (OUC) is the name given to a program-controlled communication process for communicating via the integrated PN/IE interface of S7-1200/1500 and S7-300/400 CPUs. Different connection types are available for this communication process.

The main feature of Open User Communication is its high degree of flexibility in terms of the data structures transferred. This allows open data exchange with any communicating devices providing they support the connection types available here. Since this communication is controlled solely by instructions in the user program, event-driven connection establishment and termination is possible. Connections can also be modified by the user program during runtime.

For CPUs with an integrated PN/IE interface, the TCP, UDP, and ISO-on-TCP connection types are available for Open User Communication. The communication partners can be two SIMATIC PLCs or a SIMATIC PLC and a suitable third-party device.

Instructions for Open User Communication

To create the connections, you have various instructions available after opening in the program editor in the "Instructions > Communication > Open User Communication" task card:

- Compact instructions for sending or receiving data via the integrated functions for establishing and terminating the connection (S7-1200/1500 only):
  - TSEND_C (connection establishment/termination, sending)
  - TRCV_C (connection establishment/termination, receiving)
- Individual instructions for sending and receiving data or for establishing or terminating connections:
  - TCON (connection establishment)
  - TDISCON (connection termination)
  - TSEND (TCP or ISO-on-TCP: Sending)
  - TRCV (TCP or ISO-on-TCP: Receiving)
  - TUSEND (UDP: Sending)
  - TURCV (UDP: Receiving)

Connection establishment

For Open User Communication, instructions for establishing and terminating the connection must exist for both communication partners. One communication partner sends its data using TSEND, TUSEND or TSEND_C while the other communication partner receives the data using TRCV, TURCV or TRCV_C.

One of the communication partners starts the connection establishment as the active partner. The other communication partner reacts by starting its connection establishment as the passive partner. If both communication partners have initiated their connection establishment, the communication connection is fully established.

Connection configuration

You can specify establishment of the connection via a connection description DB with the TCON_Param, TCON_IP_v4, or TCON_IP_RFC structure by means of parameter assignment as follows:

- Manually create, assign parameters and write directly to the instruction.
- Supported by connection configuration.

Connection configuration supports the establishment of the connection and should, therefore, be given preference over the other methods.

You specify the following in the connection configuration:

- Connection partner
- Connection type
- Connection ID

- Connection description DB
- Address details according to selected connection type

In addition, you specify here which communication partner activates the connection establishment and which partner establishes a connection passively in response to a request from its communication partner.

See also

Principle of operation of connection-oriented protocols (Page 104)

Connection configuration

Overview of connection configuration

Introduction

You can find the connection configuration in the inspector window of the program editor if you want to program Open User Communication with the communication instructions TSEND_C, TRCV_C or TCON.

Connection configuration supports the flexible functionality of communication programming: The parameters entered for the connection configuration are stored in an automatically generated global DB derived from the TCON_Param, TCON_IP_v4 or TCON_IP_RFC structure. You can modify the connection parameters in this connection description DB.

Structure of the connection configuration

The connection configuration is made up of the following components:

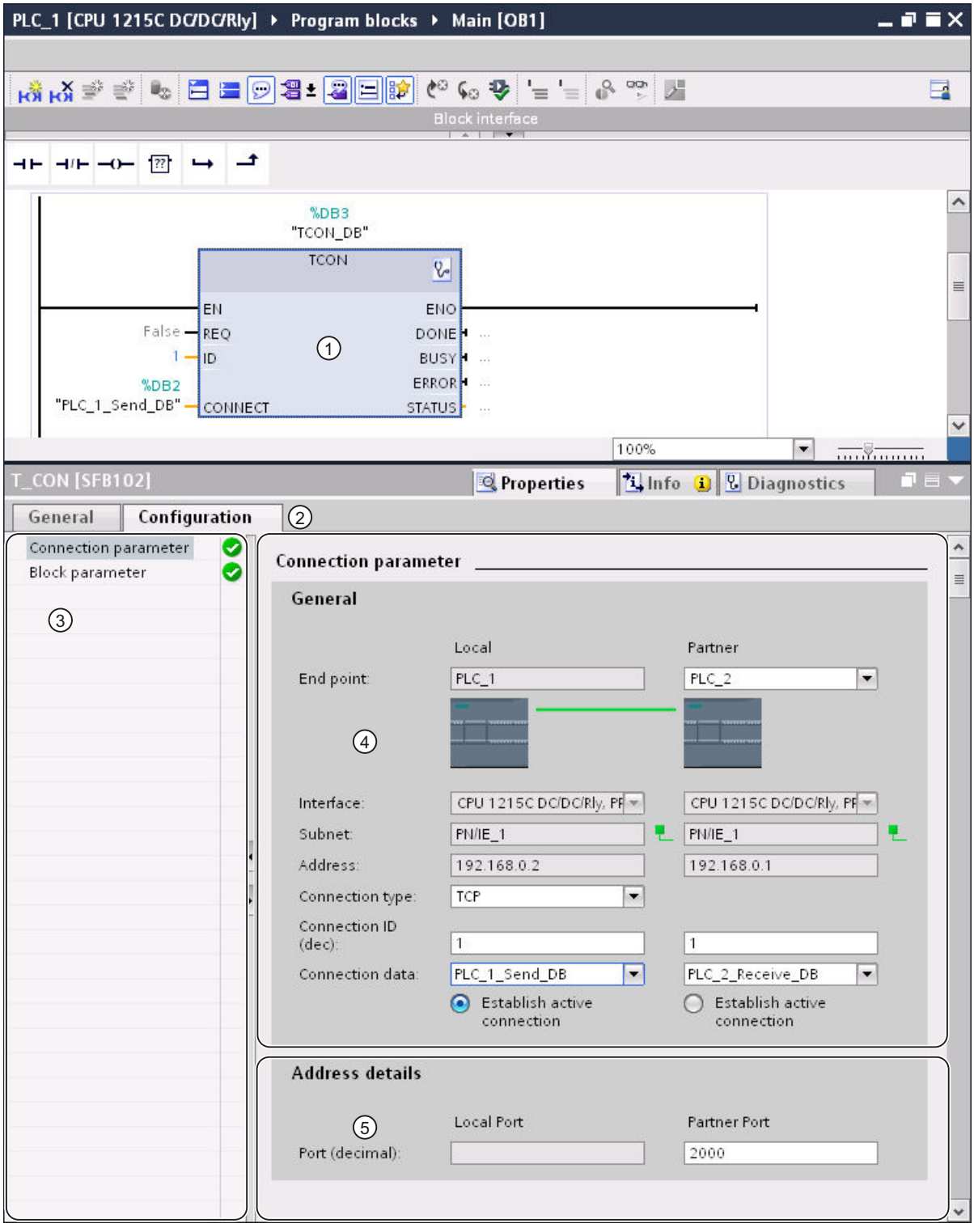

① Communication instruction for TCON, TSEND_C or TRCV_C

② "Configuration" tab in the "Properties" tab
③ Area navigation of the "Configuration" tab
④ General properties of the connection parameters
⑤ Address details of the connection parameters (for selected connection DBs)

"Configuration" tab

Enter the desired connection parameters in the "Configuration" tab. The area navigation of the "Configuration" tab includes the "Connection parameters" group. This group contains the connection configuration. Here, you can enter the parameters for the connections and the address details with system support. Here, you also connect the CONNECT (TCON, TSEND_C, TRCV_C) or ID (TCON, TSEND, TRCV, TUSEND, TURCV) block parameters of the selected communication instructions.

When all the required parameters are assigned, a check mark is set in front of the "Connection parameters" group in the area navigation.

Note

The connection parameter assignment does not check whether the connection IDs and port numbers (TCP, UDP) or TSAPs (ISO-on-TCP, ISO) are unique. When you configure Open User Communication, you should, therefore, make sure that the parameter settings are unique within a device.

See also

Connection parameters with structure according to TCON_Param (Page 106)

Connection parameters with structure according to TCON_IP_v4 (Page 109)

Connection parameters with structure according to TCON_IP_RFC (Page 114)

Description of the connection parameters

Overview

The following table shows the general connection parameters:

Parameter	Description
End point	The names of the local end point and the partner end point are shown. The local end point is the CPU for which TCON, TSEND_C or TRCV_C is programmed. The local end point is, therefore, always known. The partner end point is selected from the drop-down list. The drop-down list shows all available possible connection partners including unspecified connection partners for devices whose data is unknown in the project. For S7-1500, broadcast can be selected as the partner end point (message to all subnet devices). For S7-1500 CPs/CMs, multicast can also be selected as the partner end point (message to a group within the subnet). The connection type is converted automatically to UDP in this case. As long as no connection partner is selected, all other parameters in the mask are disabled except for the "Connection data" parameter.
Interface	The interface of the local end point is displayed. If multiple interfaces are available, e.g., by means of CPs or CMs, the interface can be selected from the drop-down list. To display or select the partner interface, a specified partner end point must first be selected.
Subnet	The subnet of the local end point is displayed, provided this exists. The partner subnet is displayed only after the partner end point has been selected. If at least one of the two connection partners is not connected with a subnet, the two connection partners are connected with each other. A connection between partners in different subnets is only possible with IP routing. The IP routing settings can be edited in the properties of the relevant Ethernet interfaces.
Address	The IP address or the PROFIBUS address of the local end point is displayed, depending on the subnet used. The corresponding address of the partner is displayed only after the partner end point has been selected. If you have selected an unspecified connection partner, the input box is empty and has a red background. In this case, you must specify a valid IP address/PROFIBUS address. The address type (IP or PROFIBUS) depends on the type of subnet that is set for the local partner. Broadcast (S7-1500 only): If "Broadcast" is set as the partner end point, a non-editable IP address with host address 255 is entered automatically for the connection partner. The network allocation corresponds to that of the sender. Example: Local IP address 192.168.0.1, partner IP address 192.168.0.255. Multicast (S7-1500 CPs/CMs only): If "Multicast" is set as the partner end point, the editable IP address 224.0.1.0 is entered automatically for the connection partner.
Connection type	Select the connection type you want to use from the "Connection type" drop-down list: • TCP • ISO-on-TCP • UDP With the S7-1500, you can also select the ISO connection type at the configuration type of the configured connections for TSEND_C and TRCV_C or TCON. The connection types can only be used for partners that support the corresponding protocol.

Parameter	Description
Connection type (for S7-1500 only)	With the S7-1500, two different configuration types can be set for TSEND_C and TRCV_C: • Programmed connections use program blocks for the connection description. • Configured connections are saved for the configuration and are only created after download to the device in runtime. You can also use the configured connection to select the connection type ISO. The specified configuration method depends on the selected connection type. If both configuration methods are possible, the programmed connection is preset. The same configuration method must be set for both connection partners.
Connection ID	Enter the connection ID in the input box. You can change the connection ID in the input boxes or enter it directly in TCON. Ensure that the connection ID assigned is unique within the device.
Connection data	The names of the connection description DBs for the connection description structured according to TCON_IP_v4, TCON_IP_RFC or TCON_Param are displayed in the drop-down lists. You can use the drop-down list to generate a new data block or to select an existing data block. This data block is filled automatically with the values from the connection configuration. The name of the selected data blocks is entered automatically in the CONNECT block parameter of the selected TSEND_C, TRCV_C or TCON instruction. From the drop-down list, you can also reference another valid data block. If a DB is referenced using the CONNECT input parameter of the TSEND_C, TRCV_C or TCON extended instructions and this does not correspond to the structure of a TCON_IP_v4, TCON_IP_RFC or TCON_Param, the drop-down list is shown with no content on a red background.
Connection name (for S7-1500 only)	If the connection type of the configured connections is set for TSEND_C and TRCV_C for the S7-1500, the "Connection data" parameter is replaced with the "Connection name" parameter. The name of the configured connection serves here as the connection data. The drop-down list is still empty after selection of a connection partner. You can use the drop-down list to generate a new connection or to select an existing connection. If needed, a data block is created and automatically filled with the values from the connection configuration. The name of the data block is entered automatically in the CONNECT block parameter of the TSEND_C or TRCV_C instruction. You can also reference an existing connection from the drop-down list.
Active connection establishment	Use the "Active connection establishment" check box to specify the active partner of the Open User Communication (only with TCP and ISO-on-TCP).
Port (only with TCP and UDP)	Address component for a TCP or UDP connection. The default after creating a new TCP connection is 2000. You can change the port numbers. The port numbers must be unique on the device!
TSAP (ISO-on-TCP only)	Address component for an ISO-on-TCP connection. The default value after creating a new ISO-on-TCP connection is E0.01.49.53.4F.6F.6E.54.43.50.2D.31 (S7-1200/1500) or E0.02.49.53.4F.6F.6E.54.43.50.2D.31 (S7-300/400). You can enter the TSAP-ID with an extension or as an ASCII TSAP. The TSAPs must be unique on the device!

Note

UDP connection for the "Broadcast" setting (S7-300/400/1200)

The parameters of the UDP connection for the "Broadcast" setting for the partner end point are stored in a connection description DB TCON_IP_v4 : With respect to UDP communication with TCON and TUSEND/TURCV , the TCON_IP_v4 is not filled with any partner parameters (value=0). However, the partner address and the partner port are necessary for sending the data and must be entered by the user in the TADDR_Param . The TADDR_Param for UDP communication is referenced by the TUSEND-/TURCV block parameter ADDR . The values for both parameters can be taken from the connection configuration.

The configuration must also be adapted for the other recipients of UDP communication. In order to receive broadcast frames, the partner port must be configured at the receiver end. For this purpose, the RemotePort parameter of the TADDR_Param must be filled at the ADDR block.

Note

Communication via TSEND_C and TRCV_C (S7-1500)

When TSEND_C and TRCV_C are used, a separate TSEND_C and TRCV_C block pair with a configured connection is required for each communication. Multiple TSEND_C and TRCV_C block pairs cannot simultaneously use the same configured connection for communication.

Additional connections for a TSEND_C or TRCV_C instruction can be created in the inspector window for the connection parameters using the appropriate button next to the connection data.

The connections configured using TSEND_C and TRCV_C are displayed in a connection table in the inspector window under "Properties > Configuration > Overview of configured connections" when the TSEND_C or TRCV_C block is selected.

See also

Assignment of port numbers (Page 120)

TSAP structure (Page 123)

Examples of TSAP assignment (Page 125)

Ability to read back connection description parameters (Page 122)

Creating and assigning parameters to connections (Page 99)

Connection parameters with structure according to TCON_Param (Page 106)

Connection parameters with structure according to TCON_IP_v4 (Page 109)

Connection parameters with structure according to TCON_IP_RFC (Page 114)

Starting connection parameter assignment

The connection configuration for Open User Communication is enabled as soon as a TCON, TSEND_C or TRCV_C instruction for communication is selected in a program block.

Requirement

- Your project must contain at least one S7-CPU.
- The program editor is open.
- A network is available.

Procedure

To insert the extended instructions for Open User Communication, proceed as follows:

1. Open the task card, pane and folder "Instructions > Communication > Open User Communication".
2. Drag one of the following instructions to a network:
   - TSEND_C
   - TRCV_C
   - TCON

   The "Call options" dialog opens.
3. Edit the properties of the instance DB in the "Call properties" dialog. You have the following options:
   - Change the default name.
   - Select the "Manual" check box to assign your own number.
   - You can also execute the DB as a multi-instance for function blocks.
4. Click "OK" to complete your entry.

Result

A corresponding instance DB is created at a single instance for the inserted instruction TSEND_C, TRCV_C or TCON. In the case of a multi-instance, the instance DB of the function block is used.

With TSEND_C, TRCV_C or TCON selected, you will see the "Configuration" tab under "Properties" in the Inspector window. The "Connection parameters" group in area navigation contains the connection parameter assignment that you can now make.

See also

Creating and assigning parameters to connections (Page 99)

Creating and assigning parameters to connections

In the connection configuration for Open User Communication, you can create and configure connections of the TCP, UDP or ISO-on-TCP type.

Requirements

A CPU exists with a TCON, TSEND_C or TRCV_C communication instruction.

Procedure

To create a connection for Open User Communication, follow these steps:

1. Select a TCON, TSEND_C or TRCV_C block of Open User Communication in the program editor.
2. Open the "Properties > Configuration" tab in the inspector window.
3. Select the "Connection parameters" group. Until you select a connection partner, only the drop-down lists for the partner end point and the local connection data are enabled. All other input options are disabled.
   The connection parameters already known are displayed:
   – Name of the local end point
   – Interface of the local end point
   – IP address (for Ethernet subnet) or PROFIBUS address (for PROFIBUS subnet) of the local end point.
4. If you want to use an existing connection description DB, go to step 6. If you want to configure the connection with a new remote connection partner, continue here:
   In the drop-down list box of the partner end point, select a connection partner. You can select an unspecified device or a CPU in the project as the communication partner. Certain connection parameters are then entered automatically.
   The following parameters are set:
   – Name of the partner end point
   – Interface of the partner end point
   – IP address (for Ethernet subnet) or PROFIBUS address (for PROFIBUS subnet) of the partner end point.

   If the connection partners are networked, the name of the subnet is displayed.
5. With the S7-1500, in the "Configuration type" drop-down list, you choose between using program blocks or configured connections.

6. Select an existing connection description DB in the "Connection data" drop-down list or for configured connections select an existing connection under "Connection name". You can also create a new connection description DB or a new configured connection. When you select an existing connection description DB, the available configuration data is applied and entered in the connection description. Any data of a remote partner that was entered previously is overwritten. Later, you can still select other configured connections or change the names of the connection description DBs in order to create new data blocks:
   - You can also see the selected data block at the interconnection of the CONNECT input parameter of the selected TCON, TSEND_C or TRCV_C instruction.
   - If you have already specified a connection description DB for the connection partner using the CONNECT parameter of the TCON, TSEND_C or TRCV_C instruction, you can either use this DB or create a new DB.
   - If you edit the name of the displayed data block in the drop-down list, a new data block with the changed name but with the same structure and content is generated and used for the connection.
   - Changed names of a data block must be unique in the context of the communication partner.
   - A connection description DB must have the structure TCON_Param, TCON_IP_v4 or TCON_IP_RFC, depending on CPU type and connection.
   - A data block cannot be selected for an unspecified partner.

   Additional values are determined and entered after the selection or creation of the connection description DB or configured connection.
   The following is valid for specified connection partners:
   - ISO-on-TCP connection type
   - Connection ID with default of 1
   - Active connection establishment by local partner
   - TSAP ID
     for S7-1200/1500: E0.01.49.53.4F.6F.6E.54.43.50.2D.31
     for S7-300/400: E0.02.49.53.4F.6F.6E.54.43.50.2D.31

   The following is valid for unspecified connection partners:
   - TCP connection type
   - Partner port 2000

   The following applies for a configured connection with a specified connection partner:
   - TCP connection type
   - Connection ID with default of 257
   - Active connection establishment by local partner
   - Partner port 2000

   The following applies for a configured connection with an unspecified connection partner:
   - TCP connection type
   - Local port 2000

7. Enter a connection ID as needed for the connection partner. No connection ID can be assigned to an unspecified partner.

   Note

   You must enter a unique value for the connection ID at a known connection partner. The uniqueness of the connection ID is not be checked by the connection parameter settings and there is no default value entered for the connection ID when you create a new connection.

8. Select the desired connection type in the relevant drop-down list. Default values are set for the address details depending on the connection type. You can choose between the following:
   - TCP
   - ISO-on-TCP
   - UDP

   For configured connections with S7-1500, ISO applies in addition.
9. You can edit the input boxes in the address details. Depending on the selected protocol, you can edit the ports (for TCP and UDP) or the TSAPs (for ISO-on-TCP and ISO).
10. Use the "Active connection establishment" check box to set the connection establishment characteristics for TCP, ISO and ISO-on-TCP. You can decide which communication partner establishes the connection actively.

Changed values are checked immediately for input errors by the connection configuration and entered in the data block for the connection description.

Note

Open User Communication between two communication partners can only work when the program section for the partner end point has been downloaded to the hardware. To achieve fully functional communication, make sure that you load not only the connection description of the local CPU on the device but also that of the partner CPU as well.

See also

Description of the connection parameters (Page 96)

Starting connection parameter assignment (Page 98)

TSAP structure (Page 123)

Assignment of port numbers (Page 120)

Connection parameters with structure according to TCON_Param (Page 106)

Connection parameters with structure according to TCON_IP_v4 (Page 109)

Connection parameters with structure according to TCON_IP_RFC (Page 114)

Deleting connections

Introduction

The data of a created connection for Open User Communication is stored in a connection description DB. You can delete the connection by deleting the data block containing the connection description.

Requirement

You have created an Open User Communication connection.

Procedure

To delete a connection, follow these steps:

1. Select a communication partner for Open User Communication in the project tree.
2. Open the "Program blocks > System blocks > Program resources" folder below the selected communication partner.
3. Select the "Delete" command from the shortcut menu of the data block with the connection parameter assignment.

Note

If you are not certain which block to delete, open the extended instruction TCON, TSEND_C or TRCV_C. You will find the name of the data block as the CONNECT input parameter or in the connection parameter assignment as the "Connection data" parameter.

If you only delete the instance DBs of the extended instructions TCON, TSEND_C or TRCV_C, the assigned connections are not deleted as well.

Note

If the connection DB is still being used by other blocks of the extended instructions, then the corresponding calls, their instance DBs, and, if present, the combination blocks TSEND_C and TRCV_C must also be deleted from the block folder, provided they are not used elsewhere.

This action prevents the program from being inconsistent.

Result

You have deleted the connection.

Note

Insert an extended instruction TCON, TSEND_C, or TRCV_C again in order to reference an existing connection description with the TCON_Param, TCON_IP_v4, or TCON_IP_RFC structure again via the "Connection data" parameter.

Connection parameters

Principle of operation of connection-oriented protocols

Introduction

Connection-oriented protocols establish a logical connection to the communication partner before data transmission is started. After the data transmission is complete, they then terminate the connection, if necessary. Connection-oriented protocols are used especially when reliable data transmission is important. Several logical connections can exist over one physical line.

Open User Communication supports the following connection types:

- TCP
- ISO-on-TCP
- ISO (S7-1500 only)
- UDP

Both communication partners must support the same connection type for a connection. If a communication partner does not support a connection of the type ISO-on-TCP, for example, use the connection type TCP instead, if it is supported.

For communication partners that cannot be configured in the TIA Portal, such as third-party devices or PCs, enter "unspecified" for the partner end point during connection parameter assignment. The required connection type for unspecified devices is listed in the respective documentation.

Note

Connections with ISO

For S7-1500 CPUs, configured connections of the type ISO can be created using the TSEND_C and TRCV_C instructions. For additional information on these connection types, refer to the general connection descriptions.

Characteristics of TCP

TCP is a streaming protocol in which the length of the data stream is transmitted to the receiver so that it can receive the data stream as individual TCP segments. This means no information about the start and end of a message is transmitted during data transmission via a TCP connection. The receiver cannot determine by the received segments of the data stream where one message in the data stream ends and the next one begins. It is therefore recommended that the number bytes to be received (parameter LEN, instruction TRCV/TRCV_C) be assigned the same value as the number of bytes to be sent (parameter LEN, instruction TSEND/TSEND_C).

If the length of the sent data and the length of the expected data do not match, the following occurs:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C copies the received data to the specified receive area (parameter DATA) only after the assigned length is reached. When the assigned length is reached, data of the subsequent job are already being received. As a result, the receive area contains data from two different send jobs. If you do not know the exact length of the first message, you are unable to recognize the end of the first message and the start of the second message.
- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C copies the number of bytes you specified in the LEN parameter to the receive data area (parameter DATA). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the value of LEN. With each subsequent call, you receive a further block of the sent data.

A receive area with fixed data length can be specified in the TRCV/TRCV_C instructions with the protocol version of the Ad-hoc mode.

Characteristics of ISO-on-TCP

ISO-on-TCP is a message-oriented protocol which detects the end of the message at the receiver end and indicates the data that belongs to the message to the user. This does not depend on the specified reception length of the message. This means that information regarding the length and the end of a message is included during data transmission via an ISO-on-TCP connection.

If the length of the sent data and the length of the expected data do not match, the following occurs:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C copies all the sent data to the receive data area (parameter DATA). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the length of the data sent.
- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C does not copy any data to the receive data area (parameter DATA), but instead supplies the following error information: ERROR=1, STATUS=W#16#8088 (destination buffer too small).

Characteristics of UDP

UDP is a message-oriented protocol which detects the end of the message at the receiver end and indicates the data that belongs to the message to the user. This does not depend on the specified reception length of the message. This means that information on the length and the end of a message is included during data transmission via a UDP connection.

If the length of the sent data and the length of the expected data do not match, the following occurs:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TUSEND/TSEND_C): TURCV/TRCV_C copies all the sent data to the receive data area (DATA parameter). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the length of the data sent.
- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TUSEND/TSEND_C): TRCV/TRCV_C copies as much data to the receive data area (parameter DATA) as the LEN parameter requests. No further error message is generated. In this case, the user has to call a T_URCV again in order to receive the remaining bytes.

See also

Basics of Open User Communication (Page 91)

Connection parameters with structure according to TCON_Param

Data block for connection description

A connection description DB with a structure according to TCON_Param is used for some S7-1200 CPUs when it comes to the assignment of parameters for TCP, UDP and ISO-on-TCP communication connections. The fixed data structure of the TCON_Param contains all the parameters that are needed to establish the connection. The connection description DB is automatically created for a new connection by the connection configuration for Open User Communication when the TSEND_C, TRCV_C or TCON instruction is used.

The CONNECT connection parameter of the instance DBs for TSEND_C, TRCV_C or TCON contains a reference to the data block used.

Structure of the connection description according to TCON_Param

Byte	Parameter	Data type	Start value	Description
0 … 1	block_length	UINT	64	Length: 64 bytes (fixed)
2 … 3	id	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.
4	connection_type	USINT	17	Connection type: • 17: TCP (17 dec = 0x11 hex) • 18: ISO-on-TCP (18 dec = 0x12 hex) • 19: UDP (19 dec = 0x13 hex)

Byte	Parameter	Data type	Start value	Description
5	active_est	BOOL	TRUE	Identifier for the type of connection establishment. FALSE always applies to UDP, since data can be sent and received via local ID. The following is valid for TCP and ISO-on-TCP: • FALSE: Passive connection establishment • TRUE: Active connection establishment
6	local_device_id	USINT	1	ID for the local PN/IE interface.
7	local_tsap_id_len	USINT	0	Length of parameter local_tsap_id used, in bytes; possible values: • 0 or 2, if connection type = 17 (TCP) Only the value 0 is permissible for the active side. • 2 to 16, if connection type = 18 (ISO-on-TCP) • 2, if connection type = 19 (UDP)
8	rem_subnet_id_len	USINT	0	This parameter is not used.
9	rem_staddr_len	USINT	4	Length of address of partner end point, in bytes: • 0: unspecified, in other words, parameter rem_staddr is irrelevant. • 4: valid IP address in the parameter rem_staddr (TCP and ISO-on-TCP only)
10	rem_tsap_id_len	USINT	2	Length of parameter rem_tsap_id used, in bytes; possible values: • 0 or 2, if connection type = 17 (TCP) Only the value 0 is permissible for the passive side. • 2 to 16, if connection type = 18 (ISO-on-TCP) • 0, if connection type = 19 (UDP)
11	next_staddr_len	USINT	0	This parameter is not used.
12 … 27	local_tsap_id	ARRAY [1..16] of BYTE	-	Local address component of connection: • TCP and UDP: local port no. (possible values: 1...49151; recommended values: 2000...5000); local_tsap_id[1] = high byte of port no. in hexadecimal notation; local_tsap_id[2] = low byte of port no. in hexadecimal notation; local_tsap_id[3-16] = irrelevant • ISO-on-TCP: local TSAP-ID: local_tsap_id[1] = B#16#E0; local_tsap_id[2] = rack and slot of local end points (bits 0 to 4: Slot number, bits 5 to 7: rack number); local_tsap_id[3-16] = TSAP extension, optional Note: Make sure that every value of local_tsap_id is unique within the CPU.
28 … 33	rem_subnet_id	ARRAY [1..6] of USINT	-	This parameter is not used.

Byte	Parameter	Data type	Start value	Description
34 … 39	rem_staddr	ARRAY [1..6] of USINT	-	TCP and ISO-on-TCP only: IP address of the partner end point, for example, for 192.168.002.003: • rem_staddr[1] = 192 • rem_staddr[2] = 168 • rem_staddr[3] = 002 • rem_staddr[4] = 003 • rem_staddr[5-6]= irrelevant
40 … 55	rem_tsap_id	ARRAY [1..16] of BYTE	-	Partner address component of connection • TCP: partner port no. (possible values: 1...49151; recommended values: 2000...5000); rem_tsap_id[1] = high byte of port no. in hexadecimal notation; rem_tsap_id[2] = low byte of port no. in hexadecimal notation; rem_tsap_id[3-16] = irrelevant • ISO-on-TCP: partner TSAP-ID: rem_tsap_id[1] = B#16#E0; rem_tsap_id[2] = rack and slot of partner end point (bits 0 to 4: Slot number, bits 5 to 7: rack number); rem_tsap_id[3-16] = TSAP extension, optional • UDP: This parameter is not used.
56 … 61	next_staddr	ARRAY [1..6] of BYTE	-	This parameter is not used.
62 … 63	spare	WORD	W#16#0000	Reserved.

Note

TCON_Param for S7-1500 CPU

The connection description DB with the structure according to TCON_Param is also supported by S7-1500 CPUs for migration reasons. However, we recommend that you use the new structures TCON_IP_v4 and TCON_IP_RFC.

Note

TCON_Param for S7-1500 R/H CPUs

The connection description DB with the structure according to TCON_Param is not supported by S7-1500 RH CPUs.

See also

Principle of operation of connection-oriented protocols (Page 104)

Description of the connection parameters (Page 96)

Ability to read back connection description parameters (Page 122)

Overview of connection configuration (Page 93)

TSAP structure (Page 123)

Assignment of port numbers (Page 120)

Connection parameters with structure according to TCON_IP_v4

Data block for connection description

A connection description DB with a structure according to TCON_IP_v4 is used for CPUs of S7-1200 V4.0 and higher and S7-1500 to assign parameters for TCP and UDP communication connections. The fixed data structure of the TCON_IP_v4 contains all parameters that are required to establish the connection. The connection description DB is automatically created for a new connection by the connection configuration for Open User Communication when the TSEND_C, TRCV_C or TCON instruction is used.

The CONNECT connection parameter of the instance DBs for TSEND_C, TRCV_C or TCON contains a reference to the data block used.

Structure of the connection description according to TCON_IP_v4

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	64	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.
4	ConnectionType	BYTE	11	Connection type: • 11: TCP (11 dec = 0x0B hex) • 19: UDP (19 dec = 0x13 hex) For compatibility and migration reasons, the CPU S7-1500 also allows the values of the connection description DB with a structure according to TCON_Param. For the connection type TCP, the entry 17 is therefore also valid (17 dec = 0x11 hex).
5	ActiveEstablished	BOOL	TRUE	Identifier for the type of connection establishment: • FALSE: Passive connection establishment • TRUE: Active connection establishment

Byte	Parameter	Data type	Start value	Description
6 … 9	RemoteAddress	ARRAY [1..4] of BYTE	-	• IP address of the partner end point, for example, for 192.168.0.1: – addr[1] = 192 – addr[2] = 168 – addr[3] = 0 – addr[4] = 1 • Multicast address of an IPv4 multicast group (for S7-1500-CPUs as of firmware version V2.0 or V2.1 and higher when using TSEND_C/TRCV_C)
10 … 11	RemotePort	UINT	2000	Port address of the remote connection partner (value range: 1 to 49151).
12 … 13	LocalPort	UINT	2000	Port address of the local connection partner (value range: 1 to 49151).

See also

Principle of operation of connection-oriented protocols (Page 104)

Description of the connection parameters (Page 96)

Ability to read back connection description parameters (Page 122)

Overview of connection configuration (Page 93)

Assignment of port numbers (Page 120)

Connection parameters according to TCON_IP_V4_SEC

Data block for connection description

To configure the communication connections for TCP over IPv4 with Secure Communication, use the following modules:

- S7-1500 as of firmware version V2.0
  With connection description DB with the structure of the SDT TCON_IP_V4_SEC
- Additionally optional: CP 1543-1 as of firmware version V2.0 or CP1543SP-1 as of firmware version V1.0

Note

Non-secure TCP or UDP connection over IPv4

You can also use SDT TCON_IP_V4_SEC for a non-secure TCP or UDP connection over IPv4.

The connection parameter CONNECT of the instance DBs for TCON contains a reference to the data block used.

Structure of the connection description according to TCON_IP_V4_SEC

Byte	Parameter	Data type	Start value	Description
0 … 15	ConnPara	TCON_IP_v4	-	SDT for the connection parameters Information about the interface_id: • If you leave the interface_id at the preset value of 0, the operating system of the CPU evaluates the remote IP address and the IP routes existing locally and then specifies an Industrial Ethernet interface of the CPU for establishing the secure OUC connection. In this case, the diagnostics data is always assigned to the first Industrial Ethernet interface of the CPU. • If you specify the hardware identifier of an Industrial Ethernet interface of the CPU or of a CP as interface_id, the secure OUC connection is established via the associated Industrial Ethernet interface.
16	ActivateSecure-Conn	BOOL	FALSE	Activation of Secure Communication for this connection If this parameter has the value FALSE, the subsequent security parameters are irrelevant, meaning that the connection is non-secure. You can set up a non-secure TCP or UDP connection in this case.
17	TLSServerReq-ClientCert	BOOL	FALSE	Only for the server side: Request for an X.509-V3 certificate from the TLS client
18 ... 19	ExtTLSCapabilities	WORD	16#0	• Bit 0: Only for the client side. A set bit means that the client validates the alternative name of the certificate subject (subjectAlternateName) in the X.509-V3 certificate of the server to check the identity of the server. The certificates are checked when the connection is established. • Bit 1 to 15: reserved for future upgrades
20 ... 23	TLSServerCertRef	UDINT	0	• Server side: ID of its own X.509-V3 certificate • Client side: ID of the X.509-V3 certificate (usually a CA certificate) that is used by the TLS client to validate the TLS server authentication. If this parameter is 0, the TLS client uses all the (CA) certificates currently loaded in the client certificate store to validate the server authentication.
24 … 27	TLSClientCertRef	UDINT	0	• Client side: ID of its own X.509-V3 certificate • Server side: ID of the X.509-V3 certificate (or a group of X.509-V3 certificates) that is used by the TLS server to validate the TLS client. If this parameter is 0, the TLS server uses all (CA) certificates currently loaded in the server certificate store to validate the client authentication.

See also

Connection parameters with structure according to TCON_IP_v4 (Page 109)

Secure OUC between two S7-1500 CPUs (Page 331)

Connection parameters in accordance with TCON_QDN

Data block for connection description

To configure the communication connections for TCP and UDP over the fully qualified domain name, use a connection description DB with a structure according to TCON_QDN for S7-1500 CPUs as of firmware version V2.0.

The connection parameter CONNECT of the instance DBs for TCON contains a reference to the data block used.

Note

TCON_DNS structure

TCON_QDN replaces TCON_DNS. Therefore use only connection description DBs with a structure according to TCON_QDN.

Structure of the connection description in accordance with TCON_QDN

Byte	Parameter	Data type	Start value	Description
0 … 1	interface_id	HW_ANY	0	irrelevant Note: As of firmware version V2.0 of the S7-1500 CPU the following applies: As long as the connection is established, it is assigned to the first PROFINET interface. As soon as the connection is established successfully, it is assigned to the used PROFINET interface.
2 … 3	id	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TCON instruction under ID.
4	connection_type	BYTE	11	Connection type: • 11: TCP (11 dec = 0x0B hex) • 19: UDP (19 dec = 0x13 hex) For compatibility and migration reasons, the CPU S7-1500 also allows the values of the connection description DB with a structure according to TCON_Param. For the connection type TCP, the entry 17 is therefore also valid (17 dec = 0x11 hex).
5	active_established	BOOL	TRUE	Identifier for the type of connection establishment: • FALSE: Passive connection establishment • TRUE: Active connection establishment
6 … 261	remote_qdn	Array of STRING [1..254]	-	Fully qualified domain name of the partner end point, which must finish with "." Please note that in a SIMATIC network, the name including the concluding dot must not exceed 254 characters.

Byte	Parameter	Data type	Start value	Description
262 … 263	remote_port	UINT	2000	Port address of the remote connection partner (value range: 1 to 49151).
264 … 265	local_port	UINT	2000	Port address of the local connection partner (value range: 1 to 49151).

Connection parameters in accordance with TCON_QDN_SEC

Data block for connection description

To configure the communication connections for TCP over the fully qualified domain name with secure communication, use a connection description DB with the structure of the SDT TCON_QDN_SEC for S7-1500 CPUs with firmware version V2.0 or higher.

Note

Non-secure TCP or UDP connection over the fully qualified domain name

You can also use the TCON_QDN_SEC SDT for a non-secure TCP or UDP connection over the fully qualified domain name.

The connection parameter CONNECT of the instance DBs for TCON contains a reference to the data block used.

Structure of the connection description according to TCON_QDN_SEC

Byte	Parameter	Data type	Start value	Description
0 … 271	ConnPara	TCON_QDN	-	Connection parameter
272	ActivateSecure-Conn	BOOL	FALSE	Activation of secure communication for this connection If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
273	TLSServerReq-ClientCert	BOOL	FALSE	Only for the server side: Request for an X.509-V3 certificate from the TLS client
274 … 275	ExtTLSCapabilities	WORD	16#0	• Bit 0: Only for the client side. A set bit means that the client validates the subjectAlternateName in the X.509-V3 certificate of the server to check the identity of the server. The certificates are checked when the connection is established. • Bit 1 to 15: reserved for future upgrades

Byte	Parameter	Data type	Start value	Description
276 .. . 279	TLSServerCertRef	UDINT	0	• Server side: ID of its own X.509-V3 certificate • Client side: ID of the X.509-V3 certificate (usually a CA certificate) that is used by the TLS client to validate the TLS server authentication. If this parameter is 0, the TLS client uses all (CA) certificates currently loaded in the client certificate store to validate the server authentication.
280 … 283	TLSClientCertRef	UDINT	0	• Client side: ID of its own X.509-V3 certificate • Server side: ID of the X.509-V3 certificate (or a group of X.509-V3 certificates) that is used by the TLS server to validate TLS client authentication. If this parameter is 0, the TLS server uses all (CA) certificates currently loaded in the server certificate store to validate the client authentication.

See also

Connection parameters in accordance with TCON_QDN (Page 112)

Secure OUC between two S7-1500 CPUs (Page 331)

Secure OUC from an S7-1500 CPU as a TLS client to an external PLC (TLS server) (Page 334)

Secure OUC from an S7-1500 CPU as TLS server to an external PLC (TLS client) (Page 336)

Secure OUC via e-mail (Page 343)

Connection parameters with structure according to TCON_IP_RFC

Data block for connection description

A connection description DB with a structure according to TCON_IP_RFC is used for CPUs of S7-1200 V4.0 and higher and S7-1500 to assign parameters to ISO-on-TCP communication connections. The fixed data structure of the TCON_IP_RFC contains all parameters that are required to establish the connection. The connection description DB is automatically created for a new connection by the connection configuration for Open User Communication when the TSEND_C, TRCV_C or TCON instruction is used.

The CONNECT connection parameter of the instance DBs for TSEND_C, TRCV_C or TCON contains a reference to the data block used.

Structure of the connection description according to TCON_IP_RFC

Byte	Parameter	Data type	Start value	Description
0 … 1	interface_id	HW_ANY	64	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	id	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.
4	connection_type	BYTE	12	Connection type 12: ISO-on-TCP (12 dec = 0x0C hex) For compatibility and migration reasons, the CPU S7-1500 also allows the values of the connection description DB with a structure according to TCON_Param. For the connection type ISO-on-TCP, the entry 18 is therefore also valid (18 dec = 0x12 hex).
5	active_established	BOOL	TRUE	Identifier for the type of connection establishment: • FALSE: Passive connection establishment • TRUE: Active connection establishment
8 … 11	remote_address	ARRAY [1..4] of BYTE	-	IP address of the partner end point, for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1
12 … 45	remote_tselector	TSelector	-	TSelector of the remote connection partner: • TSelLength = Value range 0 to 32 as UINT • TSel[1-32] = Value range each 0 to 255 in bytes
46 … 79	local_tselector	TSelector	-	TSelector of the local connection partner: • TSelLength = Value range 0 to 32 as UINT • TSel[1-32] = Value range each 0 to 255 in bytes

See also

Principle of operation of connection-oriented protocols (Page 104)
Description of the connection parameters (Page 96)
Ability to read back connection description parameters (Page 122)
Overview of connection configuration (Page 93)
TSAP structure (Page 123)

Connection parameters to TCON_Phone

System data types TCON_... for telecontrol connections

TCON_PHONE is used as a connection description for the following types of connection:

- In mobile wireless CPs of the S7-1200 for connections of Open User Communication to transfer SMS over TCON / TDISCON / TSEND or TSEND_C.
- With the CP 1242-7 for telecontrol connections to SMS clients with TC_CON

To configure establishment of a connection using one of the previously mentioned instructions, the CONNECT parameter of the instruction is used for the connection description.

The connection description is defined by the structure of the system data type (SDT) TCON_Phone. The structure of the relevant SDT contains the parameters necessary to establish the connection with the remote communications partner.

Creating a DB of the type TCON_Phone

You need to type in the data type of the DB with the keyboard. It is not displayed in a selection list. Upper or lower case is unimportant when entering the data type.

Follow these steps to create the DB:

1. Create a data block of the type "Global DB" with block access "Standard".
   To do so, deactivate the property "Optimized access" in the block properties (shortcut menu) in the "Attributes" tab.
2. Create an SDT in the parameter configuration table of the DB.
3. Enter the following name in the declaration table of the DB in the cell of the data type:
   TCON_Phone
   The SDT and its parameters are created (see below).
4. Configure the parameters described below.

Reserved bits are not displayed.

System data type TCON_Phone for SMS connections

Note

SMS text

- Programmed SMS texts for SMS messages to be sent are accessed using the DATA parameter of the TC_SEND instruction.
- The text of a received SMS message is assigned to the address area of the CPU by the DATA parameter of the TC_RECV instruction.

Table 1-1 Parameters of TCON_Phone

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceId	HW_ANY		Reference to the local interface (telecontrol interface or Ethernet interface of the CP > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the local mobile wireless connection. The ID is assigned and must be unique within the CPU. Note on telecontrol connections of the CP 1242-7: The same value as that of the parameter ID of the TC_CON instruction must be used here.
4	ConnectionType	Byte	W#16#0E	Protocol variant 14 (E_h): SMS connection
5	ActiveEstablished	Bool		Identifier for the type of connection establishment (not relevant for the CP 1242-7): • 0: Passive connection establishment (not relevant here) • 1: Active connection establishment
6...7	-	-	-	- reserviert -
8 ... 31	PhoneNumber	String[22]		Call number of the connection partner Permitted values: Plus character (+) and numbers Note the exact notation of the international dialing code of the relevant phone number assigned by the network provider ("+" character or zeros). Note on the telecontrol connections of the CP 1242-7: if the parameter PhoneNumber is not entered, no connection partner is specified and SMS messages can be received by all authorized connection partners. Without an entry, TC_RECV initially delivers the oldest received SMS messages during startup.

Connection parameters to TCON_FDL

System data block (SDT) for FDL connection description

In order to parameterize an FDL connection for the CM 1542-5, a connection description DB with an SDT to TCON_FDL is used. The fixed data structure of the STD contains all parameters that are required to establish the connection. The connection description SDT has to be created in a created DB.

The connection parameter CONNECT of the instance DBs for TSEND_C or TRCV_C contains a reference to the data block used.

The following types of FDL connections can be established:

- **Configured FDL connections**
  These connection types are configured in the STEP 7 program editor, see AUTOHOTSPOT. TSEND_C and TRCV_C are used as blocks.
  - Specified connection
    Fully configured connection between two masters
  - Unspecified connection
    Configured connection with an unspecified partner
- **Programmed FDL connections**
  These connection types cannot be configured in the STEP 7 program editor. Instead these connection types require the calling the TCON instruction together with TSEND/TRCV or TUSEND/TURCV, respectively.
  - Unspecified Layer 2 connection
    Programmed FDL connection with an unspecified partner with free Layer 2 access
  - Broadcast connection
    Connection with all connected partners
  - Multicast connection
    Connection with several defined partners

If specific settings for the individual connection types are required, this is mentioned at the subsequent parameters of the SDT.

Structure of the connection description "TCON_FDL" for configured connections

The table below describes the connection parameters for the following configured FDL connections:

- Specified connection
- Unspecified connection

Byte	Parameter	Data type	Value range / Start value	Description
0 … 1	InterfaceId	HW_ANY	0 .. 65535	Hardware identifier of the local interface
2 … 3	ID	CONN_OUC	0 .. 4095	Reference to this connection in accordance with connection configuration
4	ConnectionType	Byte	15	Connection type: 0x15 (dec. 21) for FDL
5	ActiveEstablished	Bool	FALSE	ID for the manner in which the connection is established • FALSE: Passive connection establishment • TRUE: Active connection establishment The parameter value is not processed further.

Byte	Parameter	Data type	Value range / Start value	Description
5	ServiceId	Byte	0 .. 2	Identifier of the data transfer service • 0: Automatic default with selection of the service in accordance with the receiver configured in the connection description Only SDA is used in this setting for configured connections. • 1: Send Data Acknowledged (SDA) - for active connections between two masters Use this setting for unspecified connections. • 2: Send Data Non-Acknowledged (SDN) - for multicast / broadcast
6 … 9	RemotePBAddress	Byte	0 .. 126	PROFIBUS address of the connection partner • Is used for specified connections by the connection configuration. • Has to be stated explicitly for unspecified connections.
6 … 9	LocalPBAddress	Byte	0 .. 126	Own (local) PROFIBUS address Is assigned automatically by the interface configuration.
10 … 11	RemoteLSAP	Byte	2 .. 32	LSAP (link layer service access point) of the connection partner Has to be stated explicitly for configured connections.
12 … 13	LocalLSAP	Byte	2 .. 32	Own (local) LSAP Has to be stated explicitly for configured connections.

Structure of the connection description "TCON_FDL" for programmed connections

The table below describes the connection parameters for the following programmed FDL connections:

- Unspecified Layer 2 connection
- Broadcast connection
- Multicast connection

Byte	Parameter	Data type	Value range / Start value	Description
0 … 1	InterfaceId	HW_ANY	0 .. 65535	Hardware identifier of the local interface
2 … 3	ID	CONN_OUC	0 .. 4095	Reference to this connection Reference to the value of the ID generated by TCON. Specify the value for TSEND / TRCV or TUSEND / TURCV at the parameter ID.
4	ConnectionType	Byte	15	Connection type: 0x15 (dec. 21) for FDL
5	ActiveEstablished	Bool	FALSE	ID for the manner in which the connection is established • FALSE: Passive connection establishment • TRUE: Active connection establishment The parameter value is not processed further.

Byte	Parameter	Data type	Value range / Start value	Description
5	ServiceId	Byte	0 .. 2	Identifier of the data transfer service • 0: Automatic selection of the service. Not to be used for programmed connections. • 1: Send Data Acknowledged (SDA) - for active connections between two masters • 2: Send Data Non-Acknowledged (SDN) - for multicast / broadcast "1" or "2" has to be programmed for programmed connections.
6 … 9	RemotePBAddress	Byte	0 .. 127	PROFIBUS address of the connection partner Specify the address if TSEND or TUSEND is used. The value is set by the user program for TRCV or TURCV. • For broadcast / multicast: 127
6 … 9	LocalPBAddress	Byte	0 .. 126	Own (local) PROFIBUS address Is assigned automatically by the interface configuration.
10 … 11	RemoteLSAP	Byte	2 .. 32, 63	LSAP (link layer service access point) of the connection partner Specify the address if TSEND or TUSEND is used: • For unspecified Layer 2 connections: 2 .. 32 • For broadcast: 63 • For multicast: Use the same values as "LocalLSAP". The value is set by the user program for TRCV or TURCV.
12 … 13	LocalLSAP	Byte	2 .. 32	Own (local) LSAP Is to be assigned by the program.

Assignment of port numbers

Introduction

When an Open User Communication is created, the value 2000 is automatically assigned as the port number.

Permissible values for port numbers are 1 to 49151. You can assign any port number within this range. However, because some ports may already be used depending on the system, port numbers within the range from 2000 to 5000 are recommended.

Note

Port numbers must be unique. The connection configuration or a corresponding block call is rejected with an error if the port numbers are assigned twice.

Overview of port numbers

The following table summarizes the system reactions to different port numbers.

Port no.	Description	System reaction
2000 … 5000	Recommended range	No warning, no error message on entry Port number is permitted and accepted
1 … 1999, 5001 … 49151	Can be used, but is outside the recommended range	Warning message on entry Port number is permitted and accepted
0, 20, 21, 25, 80, 102, 135, 161, 443, 465, 989, 990, 8080, 34962 … 34964	Can be used conditionally*	
53, 80, 102, 135, 161, 162, 443, 520, 9001, 34962 … 34964	Can be used conditionally**	

* Ports specified for specific functions ("well-known Ports"):
0: ANY - Port number is automatically assigned by S7-1500 CPU as value (>49151)
20: FTP data transmission
21: FTP control
25: SMTP (Simple Mail Transfer Protocol) / TMAIL_C
80: HTTP (Hypertext Transfer Protocol) / Web server
102: ISO-on-TCP (RFC1006)
135: DCE Endpoint Mapper for PROFINET
161: SNMP (Simple Network Management Protocol)
443: HTTPS (Hypertext Transfer Protocol Secure)
465: SMTPS (Simple Mail Transfer Protocol Secure)
989: FTPS data transfer
990: FTPS control
8080: HTTP alternative / internal Web
34962 ... 34964: PROFINET

Note

The user usually specifies the value 0 for the local port on the active connection end point for UDP/TCP. In this case, the CPU operating system selects the next available port above 49151. The partner port usually has the default 0 with the passive connection end point. The corresponding parameter is disabled in the connection configuration.

** These ports are disabled depending on the function scope of the CPU in use. The documentation of the respective CPUs provides the assignment of these ports.

See also

Description of the connection parameters (Page 96)

Creating and assigning parameters to connections (Page 99)

Ability to read back connection description parameters

Changing parameter values in the connection description

The connection description for exactly one connection of the Open User Communication is entered from the connection configuration in the connection description DB.

You can change the parameter values of the connection description DB outside of the connection configuration in the user program. Connection description DBs containing values you changed subsequently can be read back from the connection configuration. Under "Properties > Configuration > Connection parameters", the inspector window displays only the connection parameters stored in the connection description DB.

Note

You can only change the values in the running user program if the instructions TCON, TSEND_C or TRCV_C are not being processed and the referenced connection is not established.

The connection configuration does not support nested entries of connection descriptions in DB types that can only be found via offset referencing (for example, Global-DB).

The structure of the connection description cannot be changed.

Ability to read back individual connection parameters

For the "Address" parameter of the communication partner in a TCP or ISO-on-TCP connection, its IP address is displayed from the "rem_staddr" parameter of the connection description.

The following values can also be reloaded from the connection description:

- Connection type
- Local connection ID
- Active/passive connection establishment (only with UDP)
- Local TSAP (ISO-on-TCP only)
- Partner TSAP (ISO-on-TCP only)
- Local port (only with TCP and UDP)
- Partner port (only with TCP)

The values of the connection ID parameters of the communication partner, the connection data, as well as the connection establishment, are not included in the connection description in the local connection description DB. Consequently, these parameters cannot be displayed when the connection configuration is reopened. The connection establishment of the partner results from the local connection establishment and is therefore also displayed.

A new communication partner can be selected at any time in the "Partners" drop-down list box.

When a CPU recognized in the project is selected as a specified communication partner, the entry options for the connection ID and the connection data are shown again.

See also

Connection parameters with structure according to TCON_Param (Page 106)

Description of the connection parameters (Page 96)

TSAP structure

Introduction

For an ISO-on-TCP connection, Transport Service Access Points (TSAPs) must be assigned for both communication partners. TSAP IDs are assigned automatically after an ISO-on-TCP connection is created. To ensure the uniqueness of TSAP IDs within a device, you can change the preassigned TSAPs in the connection parameter assignment.

Structure of TSAPs

You must comply with certain rules when assigning TSAPs. A TSAP must contain a certain number of bytes, which are able to be displayed and entered as hexadecimal values (TSAP-ID) or as ASCII characters (ASCII-TSAP):

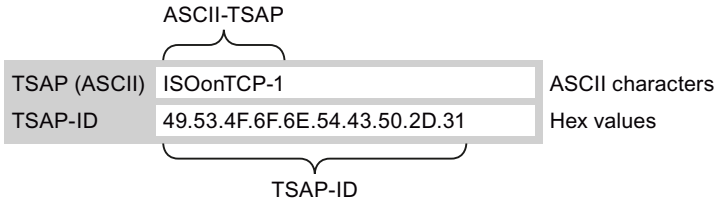

Entries or changes of the TSAP-ID or the ASCII-TSAP in the corresponding entry fields always take effect in the other display format as well.

If a TSAP contains no valid ASCII characters, the TSAP is displayed only as TSAP-ID and not as ASCII-TSAP. This is the case after a connection is created. The first two hex characters as TSAP-ID identify the communication type and the rack/slot. Because these characters are not valid ASCII characters for a CPU, the ASCII-TSAP is not displayed in this case.

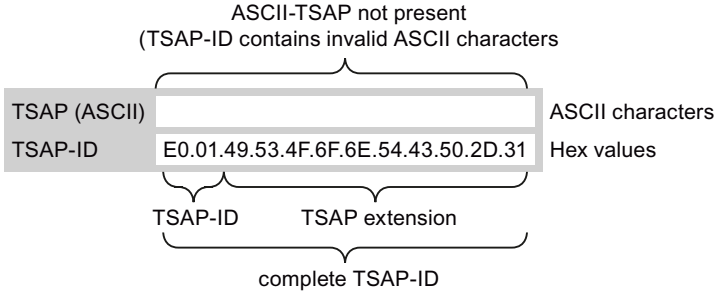

In addition to the rules for length and structure of TSAPs, you must also ensure the uniqueness of the TSAP-ID. The assigned TSAPs are not automatically unique.

Length and content of TSAPs

Structure of the TSAP ID with TSAP extension:

- Valid for CPU S7-1200 firmware V1
  Length = 2 to 16 bytes
  x_tsap_id[0] = 0xE0 (Open User Communication)
  x_tsap_id[1] (bits 0 to 4) = slot number of CPU
  x_tsap_id[1] (bits 5 to 7) = rack number of CPU
  x_tsap_id[2...15] = any characters (TSAP extension, optional)
  (x = loc (local) or x = rem (partner))
- Valid for CPU S7-1500 and S7-1200 firmware as of V2
  Length = 2 to 16 bytes
  x_tsap_id[0] = 0xE0 (Open User Communication)
  x_tsap_id[1] = 0x00 to 0xFF
  x_tsap_id[2...15] = any characters (TSAP extension, optional)
  (x = loc (local) or x = rem (partner))

Structure of the TSAP ID as ASCII TSAP:

- Valid for CPU S7-1200 firmware V1
  Length = 3 to 16 bytes
  x_tsap_id[0...2] = 3 ASCII characters (0x20 to 0x7E)
  x_tsap_id[3...15] = any characters (optional)
  (x = loc (local) or x = rem (partner))
- Valid for CPU S7-1200 firmware as of V2
  Length = 3 to 16 bytes
  x_tsap_id[0...2] for active connection = 3 ASCII characters (0x00 to 0xFF) or any bit string*
  x_tsap_id[0...2] for passive connection = 3 ASCII characters (0x20 to 0x7E) or any bit string*
  x_tsap_id[3...15] = any characters (optional)
  (x = loc (local) or x = rem (partner))
- Valid for S7-1500 CPU
  Length = 3 to 16 bytes
  x_tsap_id[0...2] = 3 ASCII characters (0x00 to 0xFF) or any bit string*
  x_tsap_id[3...15] = any characters (optional)
  (x = loc (local) or x = rem (partner))

* ASCII character strings must not start with "SIMATIC-"

The following table shows the schematic structure of various TSAP IDs:

TSAP-ID	tsap_id_len	tsap_id[0]	tsap_id[1]	tsap_id[2]	tsap_id[3..15]
...with extension (CPU S7-1200 firmware V1)	2...16 bytes	0xE0	0x01 or 0x02 or 0x00*	Extension (optional)	Extension (optional)
...with extension (CPU S7-1500, S7-1200 firmware as of V2)	2...16 bytes	0xE0	0x00...0xFF	Extension (optional)	Extension (optional)
...as ASCII-TSAP (CPU S7-1200 firmware V1)	3...16 bytes	0x20...0x7E	0x20...0x7E	0x20...0x7	Any (optional)

*An S7-1200 CPU is normally inserted on rack 0 and slot 1, and an S7-300/400 CPU on rack 0 and slot 2. For this reason, hex value 01 or 02 is valid for the second position of the TSAP ID with extension. If the connection partner is an unspecified CPU, for example, a third-party device, the hex value 00 is also permissible for the slot address.

Note

For unspecified communication partners, the local TSAP-ID and the partner TSAP-ID can have a length of 0 to 16 bytes, in which all hex values from 00 to FF are permitted.

ASCII code table for entry of ASCII TSAPs

When entering ASCII TSAPs with the hexadecimal values from 20 to 7E, only the following characters are permitted:

Code	..0	..1	..2	..3	..4	..5	..6	..7	..8	..9	..A	..B	..C	..D	..E	..F
2..		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
3..	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4..	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5..	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6..	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7..	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

See also

Examples of TSAP assignment (Page 125)

Description of the connection parameters (Page 96)

Creating and assigning parameters to connections (Page 99)

Examples of TSAP assignment

The following examples show the processing of the TSAPs for CPUs of the S7-1200/1500 (CPU on slot 1) under various points of view:

- Example 1: Creating a new connection for PLC-PLC communication
- Example 2: Entry of a local ASCII-TSAP
- Example 3: Entry of a TSAP extension in the TSAP-ID
- Example 4: Incorrect editing of the TSAP-ID
- Example 5: Entry of an ASCII-TSAP via the "TSAP-ID" entry field

Example 1: Creating a new connection for PLC-PLC communication

Once you have created a new connection with two PLCs for the Open User Communication, the TSAP extension "ISOonTCP-1" is assigned automatically.

This TSAP extension produces the TSAP-ID E0.01.49.53.4F.6F.6E.54.43.50.2D.31, which is entered automatically in the connection description DB and in the entry fields of the local and the partner TSAP. The entry fields of the ASCII-TSAPs remain empty:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	E0.01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

You can change the values in the entry fields of the TSAP-ID and the ASCII-TSAP at any time.

The entry field of the TSAP-ID shows the complete TSAP stored in the data block of the connection description. The TSAP-ID with TSAP extension, which is limited to 16 characters, is not displayed in the "TSAP (ASCII)" entry field because the character E0 does not represent a valid character for the ASCII-TSAP.

If the displayed TSAP-ID is a valid ASCII-TSAP, it is displayed in the "TSAP (ASCII)" entry field.

Changes in the entry fields for TSAP-ID and ASCII-TSAP affect the other field.

Example 2: Entry of a local ASCII-TSAP

If you have created a new connection and assigned an ASCII value for the local TSAP in the "TSAP (ASCII)" entry field, for example, "ISOonTCP-1", the resulting TSAP-ID is created automatically.

When you exit the "TSAP (ASCII)" entry field, the number of ASCII characters is checked automatically for compliance with the limit (3 to 16 characters) and the resulting TSAP-ID is entered into the corresponding entry field:

	Local TSAP	Partner TSAP
TSAP (ASCII)	ISOonTCP-1	
TSAP-ID	49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

Example 3: Entry of a TSAP extension in the TSAP-ID

If, following creation of a connection and entry of an ASCII-TSAP (see examples 1 and 2) in the entry field of the local TSAP-ID, you add the prefix "E0.01" to the TSAP value, the ASCII-TSAP will no longer be displayed when the entry field is exited.

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	E0.01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

Once you have exited the entry field of the TSAP-ID, a check is performed automatically to determine whether the first character of the TSAP-ID is a valid ASCII character. Since the character "E0" now present in the TSAP-ID is not a valid character for the ASCII-TSAP, the "TSAP (ASCII)" entry field no longer displays an ASCII-TSAP.

If a valid ASCII character is used, the check for compliance with the length specification of 2 to 16 characters follows.

Example 4: Incorrect editing of the TSAP-ID

If you remove the hex value "E0" from a TSAP-ID beginning with "E0.01", the TSAP-ID now begins with "01" and therefore no longer complies with the rules and is invalid:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

After the entry field is exited, a message is output because the TSAP-ID is neither a valid ASCII-TSAP (this would have to have a hex value in the range from 20 to 7E as the first value) or a valid TSAP-ID (this would have to have the identifier "E0" as the first value).

Example 5: Entry of an ASCII-TSAP via the "TSAP-ID" entry field

If you remove the value "01" in addition to the value "E0" from the incorrect TSAP-ID in example 4, the TSAP-ID begins with the hex value 49. This value is within the permissible range for ASCII-TSAPs:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

When you exit the entry field, the TSAP-ID is recognized as a valid ASCII-TSAP and the resulting ASCII-TSAP "ISOonTCP-1" is written to the "TSAP (ASCII)" entry field.

See also

TSAP structure (Page 123)

Description of the connection parameters (Page 96)

Configuring HMI connections

Introduction to configuring connections

Definition

A connection defines a logical assignment of two communication partners in order to undertake communication services. A connection defines the following:

- Communication partner involved
- Type of connection (e.g., HMI connection)
- Special properties (e.g., whether a connection is established permanently or whether it is established and terminated dynamically in the user program, and whether status messages are to be sent)
- Connection path

What you need to know about connection configuration

During connection configuration, a local connection name is assigned for an HMI connection as a unique local identification.

In the network view, a "Connections" tab is displayed in addition to the "Network overview" tab. This tab contains the connection table. A line in this connection table represents a configured connection, e.g., between an HMI device and PLC, along with its properties.

What you need to know about using connection resources

Introduction

Each connection requires connection resources for the end point and/or transition point on the devices involved. The number of connection resources is device-specific.

If all the connection resources of a communication partner are assigned, no new connections can be established. This situation is apparent when a newly created connection in the connection table has a red background. The configuration is then inconsistent and cannot be compiled.

HMI connections

For HMI connections via the **integrated** PN interface, one connection resource for the endpoint per HMI connection is occupied for the HMI device.

One connection resource is also required for the connection partner (PLC).

Views containing information about the configured connections

The views described below will provide you with comprehensive access to all the information and functions regarding configuring and checking communication connections.

- Connection display in the network view
- Connection table
- "Properties" tab for a connection in the inspector window

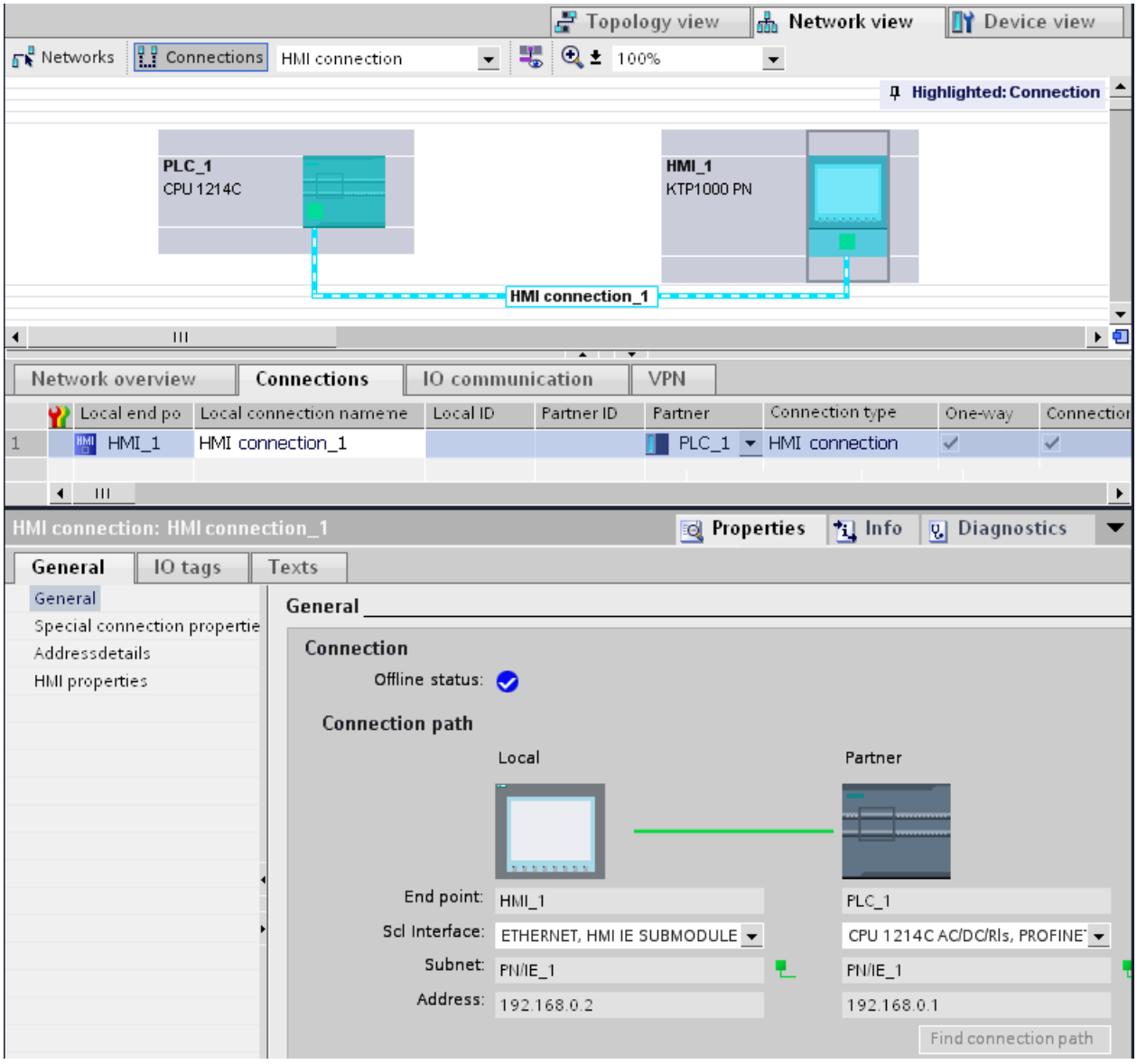

Benefits

The information shown in these views are always up-to-date in terms of the current user actions. This means:

- The connection table displays all connections created.
- If you have selected a connection in the connection table:
  - You will graphically see the connection path in the network view.
  - The "Properties" tab in the Inspector window displays the parameters of this connection.

The connection table

The connection table offers the following functions:

- List of all connections in the project
- Selection of a connection and display of connections associated with it in the network view
- Changing of connection partners
- Display showing status information

"Properties" tab for a connection in the inspector window

The properties dialog has the following meaning:

- Display for connection parameters
- Display of connection path
- Subsequent specification of connections using the "Find connection path" button

Creating a new connection

Creating a connection - alternatives

You have the following options for creating a connection in the network view:

- Graphic connection configuration
- Interactive connection configuration

You'll find the individual steps for this in the following chapters.

Requirement and result

You have created the devices with CPUs and HMI devices between which you want to configure connections in the network view.

Specifying a connection

If both partners for the connection type selected are networked on the same network, use the graphic or interactive selection of both communication partners to create a fully specified connection.

This connection is entered automatically into the connection table of the HMI device. A local connection name is assigned for this connection.

The following schematic shows a configured connection with a networked device:

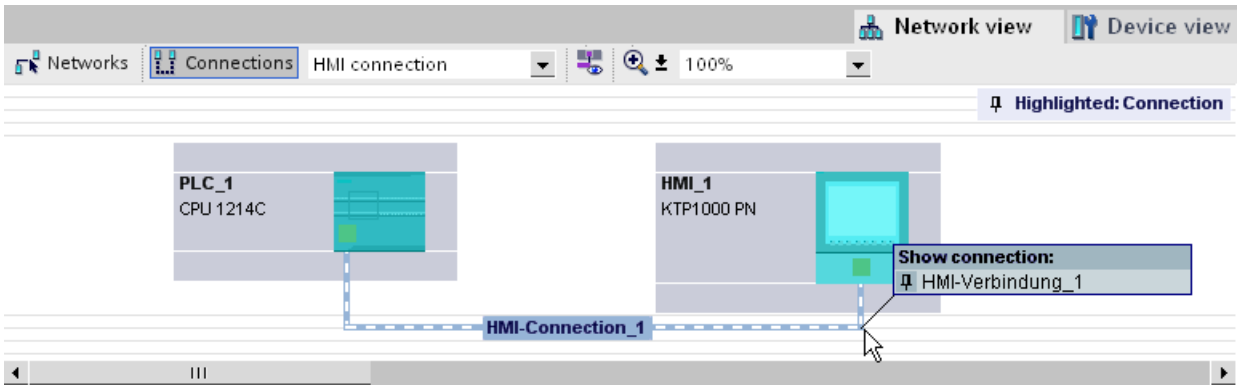

Creating a new connection graphically

Graphically configuring connections

When using the graphic connection configuration, if necessary the system asks you to define the connection path. Select the devices to be connected in the current configuration.

Automatically determining connection path

To create a connection graphically, follow these steps:

1. Click the "Connections" button.

   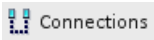

   The connection mode for the connection type you have selected is then activated.
   You will see this from the following:
   The devices that can be used for the connection type selected in your project are color-highlighted in the network view.

2. Hold down the mouse button and drag the mouse pointer from the device from which the connection will originate to the device at which the connection ends.

   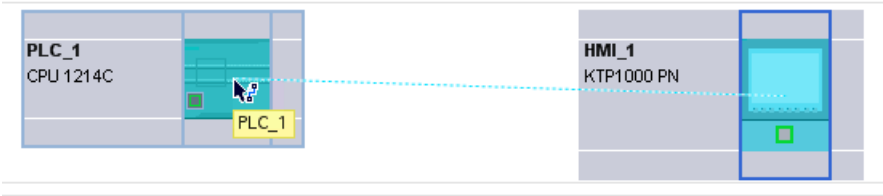

3. Release the mouse button over the destination device to create the connection between the two devices.

Result

- A specified connection is created.
- The connection path is highlighted.
- The connection is entered in the connection table.

Interactively creating a new connection

Interactively configuring connections

Define the local device and its connection partners.

Procedure

Proceed as follows to interactively create a connection:

1. Select the "Create new connection" command in the shortcut menu of a connection partner for which you want to create a connection.
   The "Create new connection" dialog is opened.
2. Select the partner endpoint.
   In the right pane of the dialog, a possible connection path fitting the selected endpoint is displayed, if available. Incomplete paths, for example, for a non-specified CPU, are marked by an exclamation mark on a red background.
3. To close the dialog, click "OK".
   To accept the configured connection and to configure additional connections to other endpoints, click "Apply".

Working in the network view

Highlighting connection path and partner in the network view

To display the connection partners for all or certain connection types in the network view, proceed as follows:

1. Click the "Connections" button.

2. Select the "Highlight connection partners" command in the shortcut menu for the HMI device whose connection partners you want to display in the network view.
3. Select "All connection partners" in the following menu.
   The local device and the CPUs of the target devices are selected. The local connection partner shows an arrow pointing right and the remote connection partners show an arrow pointing left.
4. To open a list with information on the target devices, click the arrow of the local device.
   This additional function is useful in complex network configurations in which some devices are not visible.

Note

You can display one of the connection partners which cannot be seen in the current display range of the network view. Click on the communication partner in the list that appears. Result: The display is moved such that the connection partner becomes visible.

See also

Creating a new connection graphically (Page 131)

Working with the connection table

Basic functions for tables

The connection table supports the following basic functions for editing a table:

- Changing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Changing column width

To adjust the width of a column to the content so that all texts in the lines are legible, follow these steps:

1. Position the cursor in the header of the connection table to the right of the column that you want to optimize until the cursor changes its shape to two parallel lines (as if you wanted to change the width of the column by dragging it with the cursor).
2. Double click on this point.

or

1. Open the shortcut menu on the header of the table.
2. Click on
   - "Optimize column width" or
   - "Optimize width of all columns".

For columns that are too narrow, the complete content of specific fields is shown when you pause with the cursor on the respective field.

Show / hide columns

You can use the shortcut menu of the header of the connection table to control the display of the various table columns. The shortcut menu entry "Show/hide columns" provides you with an overview of the available columns. Use the check box to control whether columns are shown or hidden.

Using cursor keys to move within the connection table

You can use the UP and DOWN cursor keys to select a connection from the connection table; the selected connection is marked and is shown highlighted in the network view.

Changing properties of connection

You can directly edit the parameters displayed in the connection table in some cases. To change the name of a connection, you do not have to navigate to the Inspector window.

Changing connection partners

You can change the connection partner of a connection as follows:

1. Select the connection.
2. Select the new connection partner from the open drop-down list in the "Partner" column.

Deleting connections

You can delete configured connections using the network view or the connection table.

In the network view you can delete one highlighted connection per action. In the connection table you can delete one or several connections per action.

Procedure

To delete a connection, follow these steps:

1. Select the connection to be deleted:
   – In the network view: Select the connection to be deleted.
   – In the connection table: Select the rows of the connections to be deleted (multiple selection possible).
2. Open the shortcut menu with a right mouse click.
3. Select the "Delete" command.

Result

The selected connection is removed completely.

Copying connections

Introduction

Connections are not copied singly but always in context along with the project or the device.

You can copy:

- Entire projects
- One or more devices within a project or from one project to another

Copying a project

When you copy a project all configured connections will also be copied. No settings whatsoever are required for the copied connections because the connections remain consistent.

Copying devices

If you copy devices for which connections have been configured (HMI devices), the connections are copied as well. To complete the connection path, you must still finalize the networking.

An S7-1200 CPU with a V.10 firmware is only a server for connections and has no connection configuration itself. Consequently, no connections are copied along with it when an S7-1200 CPU with a V1.0 firmware is copied.

Inconsistent connections - connections without assignment

With an inconsistent connection the structure of the connection data is destroyed or the connection is not functional in the project context.

Inconsistent connections cannot be compiled and loaded - operation is not possible with such a connection.

In the connection table inconsistent connections are marked in red.

Possible causes for inconsistent connections

- Deletion or change of the hardware configuration.
- Missing interface network links in the project, which are necessary for a connection.
- Connection resources are exceeded
- Errors when backing up data due to insufficient memory
- Connections to an unspecified connection partner without partner address information.

Detailed information regarding the reasons for the inconsistency can be found in the "Compile" tab following compilation (Edit > Compile).

Remedies

If the connection cannot be repaired by opening the connection properties, changing them or undoing them in the configuration, then it may be necessary to delete the connection and re-create it.

HMI connection general settings

General connection parameters

General connection parameters are displayed in the "General" parameter group under the properties of the connection; these connection parameters identify the local connection end point.

Here, you can also assign the connection path and specify all aspects of the connection partner.

Special connection properties

Display of the connection properties (cannot be changed):

- Active connection establishment
  The connection establishment always starts from the HMI device. This option is selected by default if the address of the partner is specified.
- One-way
  One-way means that the connection partner functions as a server on this connection and cannot send or receive actively.
- Sending operating mode messages
  Not relevant for HMI devices.

Address details

Displaying address details of the HMI connection. With an unspecified partner, the values for the rack and slot can be changed. All other values are obtained from the current configuration and cannot be changed.

Miscellaneous

Display of the access points for the online connection between HMI device and connection partner.

Configuring S7 connections

Introduction to configuring connections

Definition

A connection defines a logical assignment of two communication partners in order to undertake communication services. A connection defines the following:

- Communication partner involved
- Type of connection (for example, S7 connection)
- Special properties (e.g., whether a connection is established permanently or whether it is established and terminated dynamically in the user program, and whether status messages are to be sent)
- Connection path

What you need to know about connection configuration

During connection configuration, a local connection name is assigned for an S7 connection as a unique local identification.

In the network view, a "Connections" tab is displayed in addition to the "Network overview" tab. This tab contains the connection table. A row in this connection table represents a

configured connection from the viewpoint of the local communication partner with its properties, for example, between two S7-1200 CPUs.

What you need to know about using connection resources

Introduction

Each connection requires connection resources for the end point and/or transition point on the devices involved. The number of connection resources is device-specific.

If all the connection resources of a communication partner are assigned, no new connections can be established. This situation is apparent when a newly created connection in the connection table has a red background. The configuration is then inconsistent and cannot be compiled.

S7 connections

In the case of S7 connections via the PN interface, one connection resource per S7 connection is assigned for the endpoint for the S7-1200 CPU. One connection resource is also required for the connection partner.

You can find an overview of available and assigned connection resources for selected S7-1200 CPU in the Inspector window at "Properties > Connection Resources"

Views containing information about the configured connections

The views described below will provide you with comprehensive access to all the information and functions regarding configuring and checking communication connections.

- Connection display in the network view
- Connection table
- "Properties" tab for a connection in the inspector window

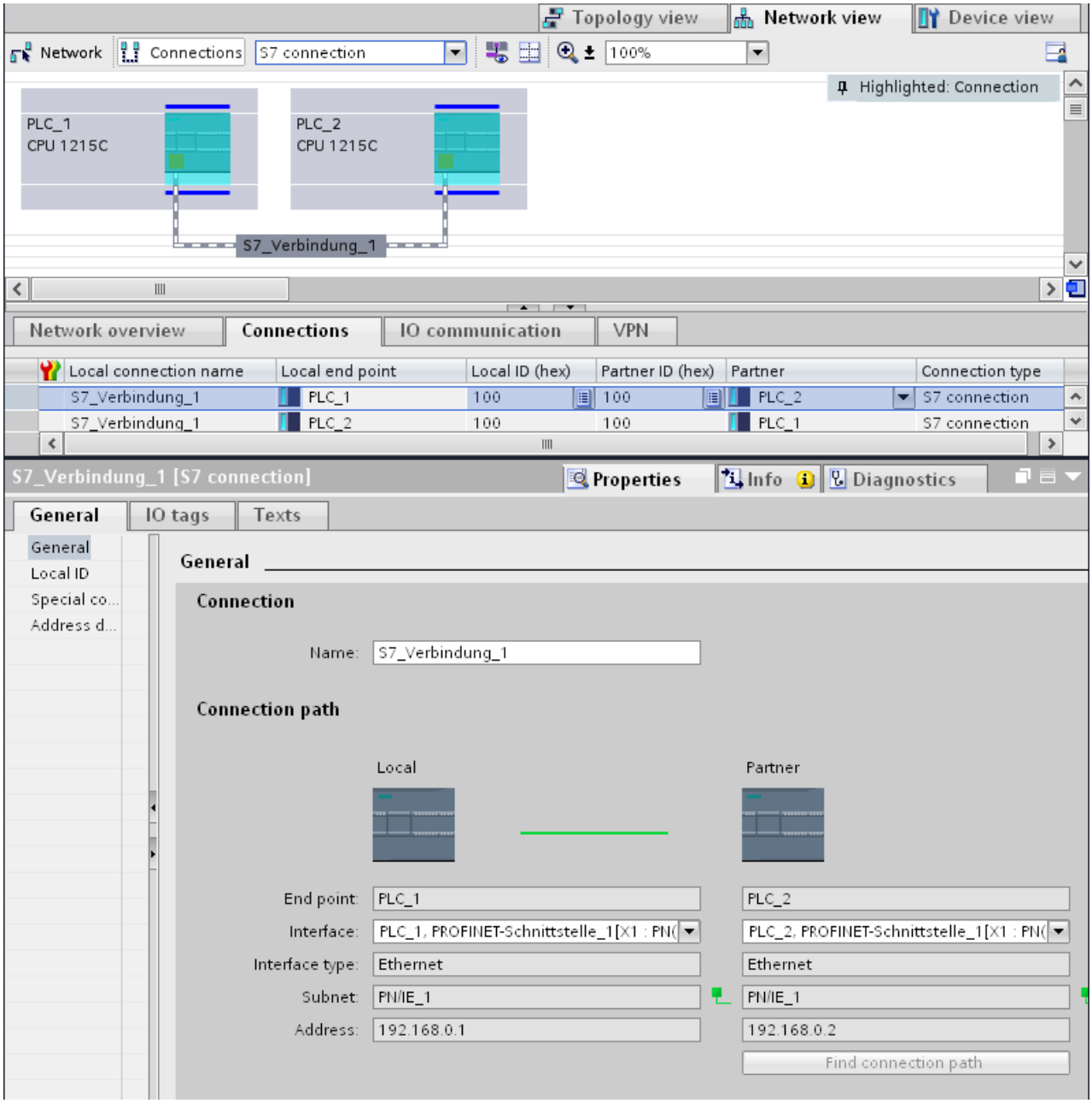

Benefits

The information shown in these views are always up-to-date in terms of the current user actions. This means:

- The connection table displays all connections created.
- If you have selected a connection in the connection table:
  - When connection mode is enabled, the connection path is highlighted in the network view.
  - The "Properties" tab in the Inspector window displays the parameters of this connection.

The connection table

The connection table offers the following functions:

- List of all connections in the project
- Selection of a connection and display of connections associated with it in the network view (when connection mode is enabled)
- Changing of connection partners
- Display showing status information

"Properties" tab for a connection in the inspector window

The properties dialog has the following meaning:

- Display for connection parameters
- Display of connection path
- Subsequent specification of connections using the "Find connection path" button

Creating a new connection

Creating a connection - alternatives

You have the following options for creating a connection in the network view:

- Graphic connection configuration
- Interactive connection configuration

You'll find the individual steps for this in the following chapters.

Requirement and result

In the network view, you have add devices between which the connections should be configured.

Specifying a connection

If both partners for the connection type selected are networked on the same network, use the graphic or interactive selection of both communication partners to create a fully specified connection.

This connection is entered automatically in the connection table of the S7-1200 CPU. A local connection name is assigned for this connection.

The following schematic shows a configured connection with a networked device:

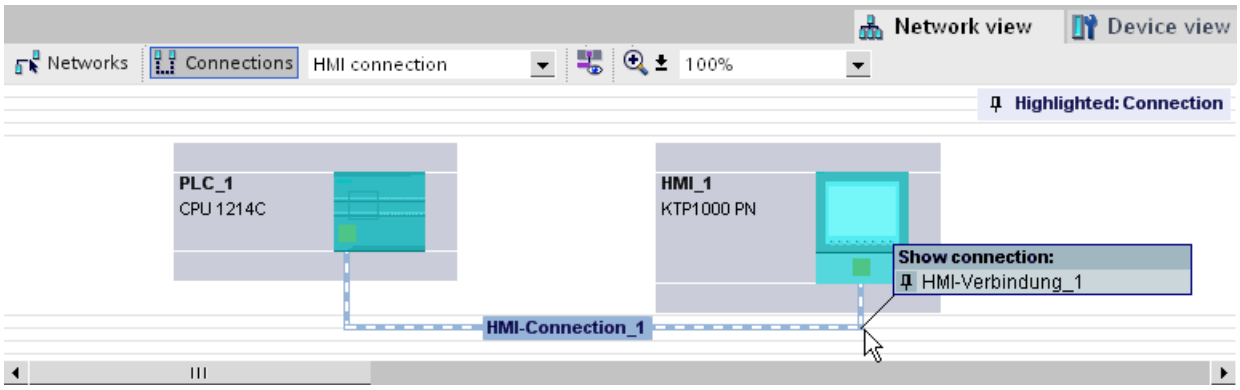

Creating a new connection graphically

Graphically configuring connections

In the case of graphic connection configuration, the connection path is automatically specified provided interfaces and resources are available. Select the devices to be connected in the current configuration.

Automatically determining connection path

To create a connection graphically, follow these steps:

1. Click the "Connections" button.

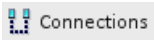

   This step activates the connection mode: You can now select the connection type you want. You will see this from the following:
   The devices that can be used for the connection type selected in your project are color-highlighted in the network view.

2. Hold down the mouse button and drag the mouse pointer from the device from which the connection will originate to the device at which the connection ends.

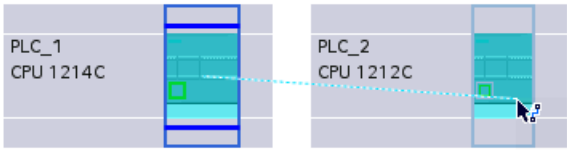

3. Release the mouse button over the destination device to create the connection between the two devices.

Result

- A specified connection is created.
- The connection path is highlighted.
- The connection is entered in the connection table.

Configuring a connection when there is no or no clear network assignment

Any networking not in place will if possible be created automatically when a connection is created. A query will be made upon completion of connection configuration if unique network assignment is not possible. You will be able to choose from the existing subnets.

Example below: A query is made upon creation of a connection between stations PLC_1 and PLC_2, which are not yet networked.

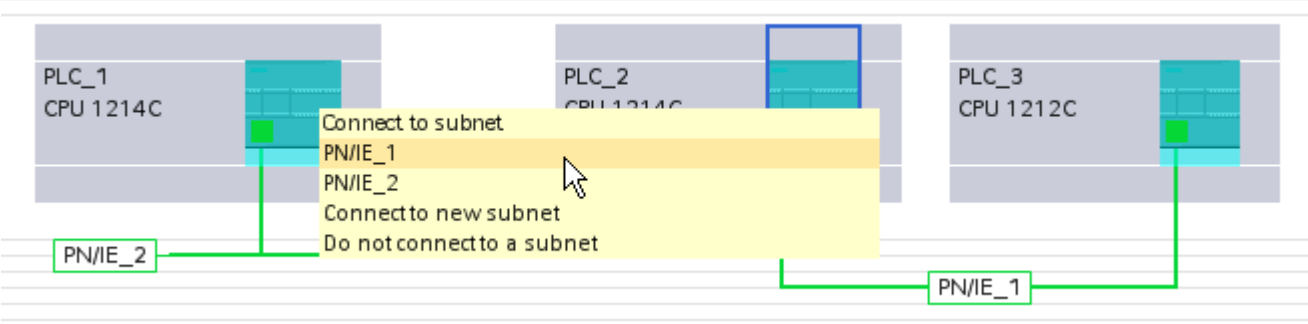

Interactively creating a new connection

Interactively configuring connections

Define the local device and its connection partners.

Procedure

Proceed as follows to interactively create a connection:

1. Select the "Add new connection" command in the shortcut menu of a connection partner for which you want to create a connection.
   The "Create new connection" dialog appears.
2. Select the partner endpoint.
   In the right pane of the dialog, a possible connection path fitting the selected endpoint is displayed, if available. Incomplete paths, for example, for a non-specified CPU, are marked by an exclamation mark on a red background.
3. To accept the configured connection and to configure additional connections to other endpoints, click "Add".
   To close the dialog, click "OK".

Working in the network view

Highlighting connection path and partner in the network view

To display the connection partners for all or certain connection types in the network view, proceed as follows:

1. Click the "Connections" button.

   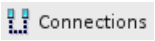

2. Select the S7-CPU for which you want to display the connection partners in the network view and then select the "Highlight connection partners" command in the shortcut menu.
3. Select "All connection partners" in the following menu.
   The local device and the CPUs of the target devices are selected. The local connection partner shows an arrow pointing right and the remote connection partners show an arrow pointing left.
4. To open a list with information on the target devices, click the arrow of the local device.
   This additional function is useful in complex network configurations in which some devices are not visible.

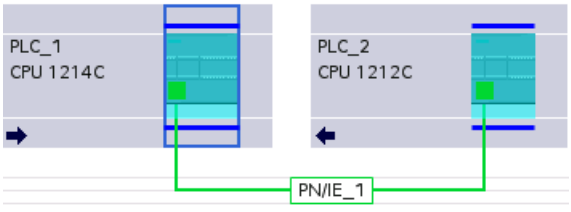

Note

You can display one of the connection partners which cannot be seen in the current display range of the network view. Click on the communication partner in the list that appears. Result: The display is moved such that the connection partner becomes visible.

Working with the connection table

Basic functions for tables

The connection table supports the following basic functions for editing a table:

- Changing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Changing column width

To adjust the width of a column to the content so that all texts in the lines are legible, follow these steps:

1. Position the cursor in the header of the connection table to the right of the column that you want to optimize until the cursor changes its shape to two parallel lines (as if you wanted to change the width of the column by dragging it with the cursor).
2. Double click on this point.

or

1. Open the shortcut menu on the header of the table.
2. Click on
   - "Optimize column width" or
   - "Optimize width of all columns".

For columns that are too narrow, the complete content of specific fields is shown when you pause with the cursor on the respective field.

Show / hide columns

You can use the shortcut menu of the header of the connection table to control the display of the various table columns. The shortcut menu entry "Show/hide columns" provides you with an overview of the available columns. Use the check box to control whether columns are shown or hidden.

If you want to store the layout, width and visibility of the table columns, click on the "Remember layout" function in the top right-hand of the network view.

Symbol	Meaning
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Using cursor keys to move within the connection table

You can use the UP and DOWN cursor keys to select a connection from the connection table; the selected connection is marked and is shown highlighted in the network view.

Changing properties of connection

You can directly edit some of the parameters displayed in the connection table. The name of the connection can, for example, only be changed in the connection table.

Changing connection partners

You can change the connection partner of a connection as follows:

1. Select the connection.
2. Select the new connection partner from the open drop-down list in the "Partner" column.

Deleting connections

You can delete configured connections using the network view or the connection table.

In the network view you can delete one highlighted connection per action. In the connection table you can delete one or several connections per action.

Procedure

To delete a connection, follow these steps:

1. Select the connection to be deleted:
   - In the network view: Select the connection to be deleted.
   - In the connection table: Select the rows of the connections to be deleted (multiple selection possible).
2. Open the shortcut menu with a right mouse click.
3. Select the "Delete" command.

Result

The selected connection is removed completely.

Copying connections

Introduction

Connections are not copied singly but always in context along with the project or the device.

You can copy:

- Entire projects
- One or more devices within a project or from one project to another

Copying a project

When you copy a project all configured connections will also be copied. No settings whatsoever are required for the copied connections because the connections remain consistent.

Copying devices

If you copy devices for which connections have been configured, the connections are copied as well. To complete the connection path, you must still finalize the networking.

An S7-1200 CPU with a V.10 firmware is merely a server for connections and has no connection configuration itself. Consequently, no connections are copied along with it when an S7-1200 CPU with a V1.0 firmware is copied.

Inconsistent connections - connections without assignment

With an inconsistent connection the structure of the connection data is destroyed or the connection is not functional in the project context.

Inconsistent connections cannot be compiled and loaded - operation is not possible with such a connection.

In the connection table inconsistent connections are marked in red.

Possible causes for inconsistent connections

- Deletion or change of the hardware configuration.
- Missing interface network links in the project, which are necessary for a connection.
- Connection resources are exceeded
- Connections to an unspecified connection partner without partner address information.

Detailed information regarding the reasons for the inconsistency can be found in the "Compile" tab following compilation (Edit > Compile).

Remedies

To assign a closed connection path to an existing open connection path, expand the device configuration in such a way that the interfaces required for the connection type are available for both partners. At "Properties > General > Interface" in the Inspector window, you can use the "Find connection path" button to create a connection to an existing partner.

S7 connection - general settings

General connection parameters

General connection parameters are displayed in the "General" parameter group under the properties of the connection; these connection parameters identify the local connection end point.

Here, you can assign the connection path and specify all aspects of the connection partner.

Local ID

The local ID of the module from which the connection is viewed is displayed here (local partner). You can change the local ID. You may need to do this if you have already programmed communication function blocks, and you want to use the local ID specified in those function blocks for the connection.

Special connection properties

Display of connection properties (can be modified depending on the components used):

- One-way
  One-way means that the connection partner functions as a server on this connection and cannot send or receive actively.
- Active connection establishment
  In the case of one-way connection, for example with a S7-1200 CPU (firmware version V1.0), a connection partner can only be provided for the active connection establishment. In the case of a two-way connection you can set which connection partner will assume the active role.
- Sending operating mode messages
  Indicates whether or not the local partner sends operating mode messages to the connection partner.

Address details

Displaying address details of the S7 connection. With an unspecified partner, the values for the rack and slot can be changed. All other values are obtained from the current configuration and cannot be changed.

S7 connection - address details

Meaning

The address details show the end points of the connection and can localize these via the specification of rack and slot.

When a connection is established, the connection-specific resources of a module are assigned permanently to this connection. This assignment requires that the connection resource can be addressed. The TSAP (Transport Service Access Point) is, as it were, the address of the resource that is formed with the help of the connection resource or, in the case of S7-1200 CPUs (firmware V2.0 or higher) with the SIMATIC-ACC (SIMATIC Application Controlled Communication).

Configuration of TSAP for S7-1200

- For S7-1200 CPU (firmare V2.0 or higher):
  "SIMATIC-ACC"<nnn><mm>
  nnn = Local ID
  mm = any value
- For S7-1200 CPU (firmare V1.0):
  <xx>.<yz>
  xx = Number of the connection resource
  y = Rack number
  z = Slot number

TSAP structure, dependent on partner

The configuration of the TSAP for S7-1200 CPUs is dependent on the respective firmware and on the remote connection partner. When a S7-1200 CPU is connected with a S7-300/400 CPU, a S7-1200 CPU also uses a TSAP configuration that includes the connection resource.

See the following examples of TSAPs of various connection configurations

- Connection between two S7-1200 CPUs (both with firmware V2.0):
  - S7-1200 CPU "A" with firmware V2.0 and local ID 100:
    TSAP: SIMATIC-ACC10001
  - S7-1200 CPU "B" with firmware V2.0 and local ID 5AE:
    TSAP: SIMATIC-ACC5AE01
- Connection between two S7-1200 CPUs (firmware V2.0 and V1.0):
  - S7-1200 CPU with firmware V2.0 and local ID 1FF:
    TSAP: SIMATIC-ACC1FF01
  - S7-1200 CPU with firmware V1.0 (rack 0, slot 1, connection resource 03):
    TSAP: 03.01
- Connection between S7-1200 CPUs (firmware V2.0) and S7-300/400 CPU:
  - S7-1200 CPU with firmware V2.0 (rack 0, slot 1, connection resource 12):
    TSAP: 12.01
  - S7-300/400 CPU (rack 0, slot 2, connection resource 11):
    TSAP: 11.02

S7 connections via CM/CP

Introduction

S7-1200 CPUs with a firmware version of V2.0 or higher support one-way and two-way S7 connections via CM/CP interfaces. This increases the number of Ethernet ports and networks that can be used for S7 connections. Even though the connection is then guided via the CM/CP, the associated S7-1200 CPU is an end point of the connection. The other end point can be any other device in the case of two-way connections. This other device must also support S7 connections.

Data volumes and configuration limit

The number of communication connections which are supported by CM/CP can be found in the manual accompanying every CM/CP. The number of connections per device can be further increased by adding other CM/CPs.

If several CM/CPs are fitted in a device, an automatic switch is made to the next CM/CP when this limit is exceeded. Specifically assign the connections using the path selection as required.

Note

Data transfer > 240 byte transfer is supported by the current CPs.

CPs with an older product version support the transfer of data with a data length of up to 240 bytes.

For more information, refer to the details provided in the Ethernet CP equipment manual.

Tasks of the Ethernet CM/CP in online mode

During data transfer via a connection, the Ethernet CM/CP takes on the following tasks:

- Receiving
  Receiving data from the Ethernet and forwarding to the user data area in the CPU
- Sending
  Transferring data from the user data area of the CPU and sending data via the Ethernet

The connection is established automatically as soon as the partner can be reached.

S7 connections using PUT and GET instructions

Basic information on communication via the PUT/GET instruction

Basic information on PUT/GET instructions

Use PUT and GET instructions to exchange data between two CPUs via an S7 connection.

The GET instruction is used to read data from a partner CPU. The PUT instruction is used to control the writing of tags by the communication partner via the user program. Apart from the PUT and GET instructions, no additional communication functions are provided for reading and writing tags.

To simplify the use of the two instructions, specify all required parameters for the connection and all block parameters in the Inspector window of the program editor.

Requirement

To be able to use the PUT and GET instructions, the following requirements must be satisfied:

- At least one S7-1200/1500 CPU or S7-300/400 CPU must be created in the project. Firmware 2.0 or higher must be installed on an S7-1200 CPU. If you have not yet created a second CPU in the project, you can initially establish the connection to an unspecified partner.
- An S7 connection must exist between the two CPUs. If you have not yet established a connection between two CPUs, a connection is automatically established during the configuration of the instructions.
- For both instructions, an instance data block is required in which all data used by the instruction is stored. The instance data block is created automatically as soon as you drag a PUT or GET instruction to a network in the program editor. For the correct execution of the program, it is essential that the instance data blocks are not changed; consequently, these data blocks are know-how protected. You only have read access to the instance data blocks.

See also

Overview of connection configuration (Page 151)

Assigning parameters to start request (Page 156)

PUT: Set parameters for write and send area (Page 157)

GET: Set parameters for read and memory area (Page 158)

Connection configuration

Overview of connection configuration

Introduction

The connection parameters for the PUT and GET instructions are assigned in the inspector window of the program editor. All parameters are saved in the corresponding instance data block.

Structure of the connection configuration

The connection configuration is made up of the following components:

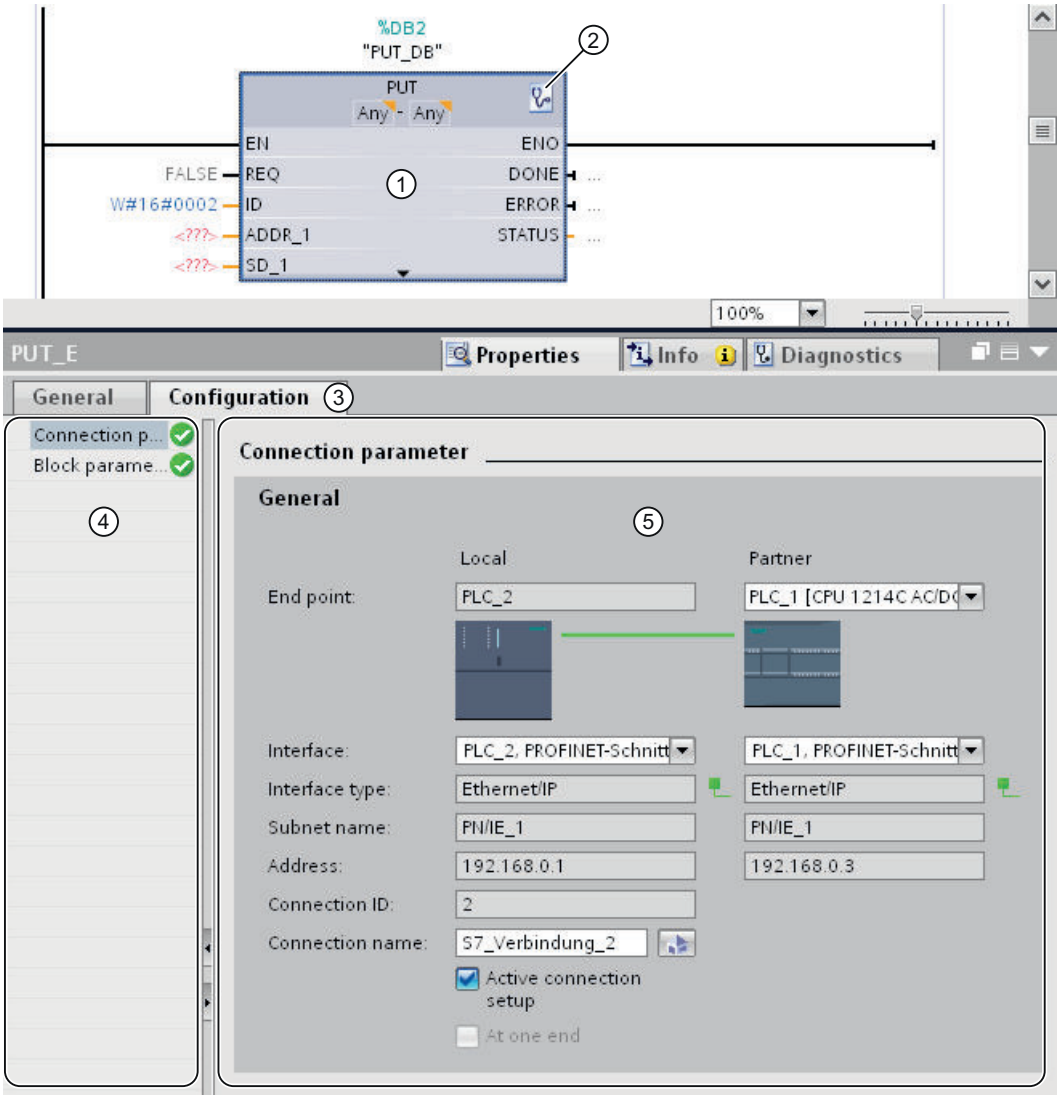

① Communication instruction for PUT or GET
② Call of online and diagnostic functions
③ "Configuration" tab in the "Properties" tab
④ Area navigation of the "Configuration" tab
⑤ General properties of the connection parameters

Display of online and diagnostic functions

If you click the icon for starting the online and diagnostic functions, the associated CPU goes online automatically. The connection table in the network view is opened. In addition, the "Diagnostics" tab and the connection information are displayed in the inspector window.

Entering the connection parameters

Enter the desired connection parameters in the "Configuration" tab. The area navigation of the "Configuration" tab includes the "Connection parameters" group. This group contains the connection configuration. Here, you can enter the parameters for the connections using system functions. When all the required parameters are assigned, a check mark is set behind the "Connection parameters" group in the area navigation.

See also

Assigning parameters to start request (Page 156)

PUT: Set parameters for write and send area (Page 157)

GET: Set parameters for read and memory area (Page 158)

Description of the connection parameters

Overview

The following table shows the general connection parameters:

Parameter	Description
End point	The names of the local end point and the partner end point are shown. • Local end point The local end point is the CPU in which the PUT or GET instruction is programmed. • Partner end point The partner end point is selected from the drop-down list. The drop-down list shows all available possible connection partners including unspecified connection partners for devices whose data is unknown in the project. As long as no connection partner is set, all other parameters in the mask are disabled.
Interface	The interface of the partner CPU is displayed. The partner interface is not displayed until a specified partner CPU has been selected.
Interface type	The type of interface via which communication is handled is displayed.
Subnet name	The subnet of the local end point is displayed, provided this exists. The partner subnet is displayed only after the partner end point has been selected. If at least one of the two connection partners is not connected with a subnet, the two connection partners are automatically connected with each other. The partner which is not connected to a network is hereby connected to the same subnet via which the other partner is already connected to a network. A connection of connection partners to different subnets is only possible with IP or S7 routing. The IP routing settings can be edited in the properties of the relevant Ethernet interfaces.
Address	The IP address of the local end point is displayed. The IP address of the partner is displayed only after the partner end point has been selected. If you have selected an unspecified connection partner, the input box is empty and has a red background. In this case, you will need to specify a valid IP address for the connection partner.
Connection ID	The currently set connection ID is displayed. You can change the connection ID in the connection table in the network view. You can also directly access the connection table while you are setting the connection parameters. To do this, click the "Create new connection" icon.

Parameter	Description
Connection name	The name of the connection which was automatically created when the PUT/GET instruction was inserted is displayed. You can change the name of the connection by entered a different name in the field. You can also create a new connection or edit existing connections by clicking the "Create new connection" icon.
Active connection establishment	Use the "Active connection establishment" option button to specify which partner starts the communication. When the connection is created, the local partner is initially specified by default for the establishment of the connection. If a device does not support active connection establishment, you have to activate active connection establishment on the other partner.
Configured at one end	If this check box is selected, the connection partner is the server for this connection. It cannot actively send or receive. This corresponds to the behavior of the PUT/GET instructions. In this case, other instructions are not possible. If the check box is not selected, other instructions can also be used for the communication.

Starting connection parameter assignment

You can assign the connection parameters for PUT and GET in the inspector window as soon as you have inserted and selected a PUT or GET instruction in a program block.

Procedure

To insert PUT/GET instructions, follow these steps:

1. Open the "Instructions" task card in the "Communication > S7 Communication" folder.
2. Drag a PUT or GET instruction to a network.
   The "Call options" dialog opens.
3. Optional: Edit the properties of the instance DB in the "Call properties" dialog. You have the following options:
   – Change the default name.
   – Select the "Manual" check box to assign your own number.
4. Click "OK".

Result

A corresponding instance data block is created for the inserted PUT or GET instruction. For S7-300 CPUs, a function block is also created in the program resources.

When PUT or GET instruction is selected, you will see the "Configuration" tab under "Properties" in the inspector window. The "Connection parameters" group in area navigation contains the connection parameter assignment that you can now make.

See also

Creating and assigning parameters to connections (Page 155)

Deleting connections (Page 156)

Creating and assigning parameters to connections

You can create S7 connections and assign the parameters for these in the connection parameter assignment of the PUT/GET instructions. Changed values are checked immediately by the connection parameter assignment for input errors.

Requirement

A CPU exists with a PUT or GET communication instruction.

Procedure

To configure an S7 connection using PUT/GET instructions, follow these steps:

1. In the program editor, select the call of the PUT or GET instruction.
2. Open the "Properties > Configuration" tab in the inspector window.
3. Select the "Connection parameters" group. Until you select a connection partner, only the empty drop-down list for the partner end point is enabled. All other input options are disabled.
   The connection parameters already known are displayed:
   – Name of the local end point
   – Interface of the local end point
   – IP address of the local end point
4. In the drop-down list box of the partner end point, select a connection partner. You can select an unspecified device or a CPU in the project as the communications partner.
   The following parameters are automatically entered as soon as you have selected the connection partner:
   – Interface of the partner end point
   – Interface of the partner end point. If several interfaces are available, you can change the interface as required.
   – Interface type of the partner end point
   – Subnet name of both end points
   – IP address of the partner end point
   – Name of the connection which is used for the communication. If no connection exists yet, it is automatically established.
5. If required, change the connection name in the "Connection name" input box. If you want to create a new connection or edit an existing connection, click on the "Create new connection" icon.

Note

The PUT and GET instructions between two communication partners can only run if both the hardware configuration and the program part for the partner end point have been loaded into the hardware. To achieve fully functional communication, make sure that you load not only the connection description of the local CPU on the device but also that of the partner CPU as well.

Deleting connections

A connection which was automatically created during the insertion of a PUT or GET instruction appears in the connection table of the network view like every standard connection. As a result, it can be deleted in the connection table.

Procedure

To delete a connection, follow these steps:

1. Open the connection table in the network view.
2. In the connection table, select the connection that you want to delete.
3. To do this, right-click the connection and select the "Delete" command from the shortcut menu.

Result

The connection is deleted. The PUT or GET instruction and the associated instance data blocks are retained and must be manually deleted if necessary.

To continue using the PUT or GET instruction, you must configure the connection again in the inspector window of the program editor, since all connection parameters were also deleted when the connection was deleted. In this case, specify a new communication partner and a suitable connection.

Block parameter assignment

Assigning parameters to start request

To start communication via the PUT or GET instruction, you have to specify an event which activates the instruction. This event is referred to as control parameter (REQ). The communication job is activated as soon as there is a positive edge at the control parameter REQ.

Please note that the control parameter REQ is assigned the default FALSE at first call.

Requirement

- The program editor is open.
- You have already inserted a PUT or GET instruction.
- A connection has been established between two communication partners.

Procedure

To define the REQ control parameter, follow these steps:

1. Select the PUT or GET instruction in the program editor.
2. Open the "Configuration" tab in the inspector window.

3. Select the "Block parameter assignment" entry in the area navigation.
4. In the "REQ" field, select a tag of the "BOOL" data type to initialize the execution of the instruction. Alternatively, you can also interconnect a previous instruction in the program editor.

See also

PUT: Set parameters for write and send area (Page 157)

GET: Set parameters for read and memory area (Page 158)

PUT: Set parameters for write and send area

For communication via the PUT instruction, you must specify the memory area of the partner CPU to which the data should be written. In addition, you must specify the memory area of the local CPU from which the data is to be read.

Requirement

- The program editor is open.
- You have already inserted a PUT instruction.
- A connection has been established between two communication partners.

Procedure

To specify the read and the memory area for the instruction, follow these steps:

1. Select the PUT instruction in the program editor.
2. Open the "Configuration" tab in the inspector window.
3. Select the "Block parameter assignment" entry in the area navigation.
4. In the "In/Outputs > Write area (ADDR_1) > Start" field, select a "REMOTE" data type pointer to the area of the partner CPU which is to be written.
   Only absolute addressing is permitted.
   Example: P#DB10.DBX5.0 Byte 10
5. In the "Length" field, enter the length of the write area and select the data type of the memory area from the drop-down list.
6. In the "In/Outputs > Send area (SD_1) > Start" field, select a pointer to the area in the local CPU which contains the data to be sent.
7. In the Length field, enter the length of the memory area to be read and select the data type from the drop-down list.
   Only the data types BOOL (for a bit array, "0" must be used as address and an integer multiple of byte must be used as length), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER are permitted.
   If the VARIANT pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 Byte 10).

See also

GET: Set parameters for read and memory area (Page 158)

GET: Set parameters for read and memory area

For communication via the GET instruction, you must specify the memory area of the local CPU to which the data should be written. In addition, you must specify the memory area of the partner CPU from which the data is to be read.

Requirement

- The program editor is open.
- You have already inserted a GET instruction.
- A connection has been established between two communication partners.

Procedure

To specify the read and the memory area for the instruction, follow these steps:

1. Select the GET instruction in the program editor.
2. Open the "Configuration" tab in the inspector window.
3. Select the "Block parameter assignment" entry in the area navigation.
4. In the "In/Outputs > Read area (ADDR_1) > Start" field, select a "REMOTE" data type pointer to the area of the partner CPU which is to be read.
   Only absolute addressing is permitted.
   Example: P#DB10.DBX5.0 Byte 10
5. In the "Length" field, enter the length of the read area and select the data type of the memory area from the drop-down list.
6. In the "In/Outputs > Memory area (RD_1) > Start" field, select a pointer to the area in the local CPU in which the read data is to be stored.
7. In the Length field, enter the length of the memory area and select the data type from the drop-down list.
   Only the data types BOOL (for a bit array, "0" must be used as address and an integer multiple of byte must be used as length), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER are permitted.

See also

PUT: Set parameters for write and send area (Page 157)

Connection resources

Configuration limits in the connection configuration

Connection resources

Some communications services require connections. During operation, communication connections utilize memory and program resources of the CPUs, CPs, and CMs involved in the communication path (e.g., memory areas in the CPU operating system). Within a controller (PLC), this applies, for example, to the CPU as connection end point and to the extensions with CPs as interfaces to the subnets. In most cases, one resource per CPU/CP/CM is assigned for a connection. In HMI communication, up to 3 connection resources are required for each HMI connection.

Available connection resources

The maximum configuration limit within the automation system is always defined by the CPU, either through the high system limit defined by the CPU (S7-1200/1500) or through the sum of usable resources of the CPU and CP (S7-300/400). In most cases, the available connection resources of a CPU/CP/CM are automatically distributed to their interfaces, as required.

Each CPU brings along reserved connection resources for programming device, HMI, and Web server communication in its configuration. In addition, there are available resources that can be used for any HMI, S7 and open communication (S7-1200/1500).

You can determine which resources are available to you based on the configuration of the devices as follows:

- Selection of the device type at connection end point (CPU)
- Number and type of additional communication modules (CP, CM)

The number of connection resources available differs for the various automation systems:

- S7-300: The used CPU defines the number of available resources for connections. The maximum number of resources for connections can be expanded by added CPs.
- S7-400: The used CPU defines the number of possible resources for S7 connections. The resources can also be used for other connection types, in which case the maximum number of resources for other connection types can be expanded by adding CPs.
- S7-1200: The used CPU defines the number of possible resources for S7 connections. This high limit can be extended by adding CPs or CMs.
- S7-1500: The CPU that is used defines the applicable high limit of the connection resources for the whole automation system. The CPU itself accounts for some of these connection resources, and additional resources are accrued via CPs and CMs. However, regardless of how many CPs and CMs are added, the total number of connection resources cannot exceed the high limit of connection resources defined by the CPU type.

Time of occupation of connection resources

Which resources are actually used in operation, depends on which communication functions are used at any specific time and which communication connections are enabled at any specific time. Therefore, only limited checks of compliance with specific configuration limits and available communication resources can be performed at the time of configuration. The connection configuration offers the greatest possible support for this.

Factors influencing the configuration limits

You determine the communication resources with the device configuration:

- Number of available S7 connection resources in the CPU (S7-300/400)
- Number of connection resources for S7 communication and open communication in the station (S7-1500)
- Number of protocol-specific connection resources in the CPs/CMs
- Number of available CPs/CMs
- Automatic, programmed, or configured connection setup

Behavior of the connection configuration

During input, the connection configuration checks the following circumstances and behaves as follows:

- During the creation of S7 connections
  Is the maximum number of available S7 connections assigned? If yes, it is not possible to configure another S7 connection via the selected interface.
  For the S7-300, it is possible to increase the configuration limit by adding CPs.
- During the creation of any connection types
  Is the maximum number of protocol-specific connection resources in the interface (CP) assigned? If yes, it is not possible to configure another connection of the selected type via the selected interface.
  For the S7-300/400, it is possible to increase the configuration limit by adding CPs.
- During the creation of any connection types with OPC (PC stations)
  In the case of connections with OPC as connection end point, more connections can generally be configured than can be operated at any one time in the CP used in the PC station. That is because the connection properties differentiate between "Maintain permanently" and "Set up on demand".
  The limit for Softnet IE CPs is 64 concurrent connections and 120 concurrent connections for Hardnet IE CPs.

Note

Number of locally configured connections greater than number of possible connections in runtime

The user can configure the maximum possible number of S7 and SEND/RECEIVE connections via OPC as "Maintain permanently" in runtime and then configure no additional connection as "Establish on demand". Alternatively, the user can configure less than the maximum possible number of S7 and SEND/RECEIVE connections via OPC as "Maintain permanently" in runtime, and accordingly configure additional S7 and SEND/RECEIVE connections as "Establish on demand".

If the user configures more than 64 S7-connections and SEND/RECEIVE connections, a warning appears during the compiling of the configuration. The warning appears independent of whether the connections are permanently ("Maintain permanently") or temporarily ("Establish on demand") configured, as even with 65 temporary connections the maximum number of simultaneous connections can theoretically be exceeded in runtime.

When the connection configuration is compiled, a warning or, in the case of a non-assignable interface, an error is output, if appropriate.

Note

S7 connection resources are exhausted

If the maximum number of S7 connection resources is reserved, no S7 connection resources additionally required for data transmission between CPU and CP are in operation or possibly available for connections. The corresponding communication jobs are then refused.

You should therefore make sure that there is a sufficient number of non-assigned S7 connection resources, depending on the total assigned communication functions.

You can find the number of reserved S7 connection resources for the selected S7-300 or S7-1200/1500 CPU in the area navigation under "Connection resources". For the S7-300, only the connection resources of the CPU are shown. For the S7-1200/1500, you also see the resources of the CPs and CMs.

Connection resources and communication type

Every connection needs connection resources for the endpoint and transition point (e.g., CP, CM) on the devices involved. The number of available connection resources is CPU/CP/CM-specific.

If all the connection resources of a communication partner are assigned, no new connections can be established. Each type of communication is considered in detail below. Any number of combinations are possible however as long as the available connection resources are taken into account.

S7 connections via integrated interface

With S7 connections via the integrated MPI/PROFIBUS DP/PN interface, one connection resource is assigned for the endpoint on the CPU per S7 connection. This applies to all S7-300/400/1200/1500 CPUs.

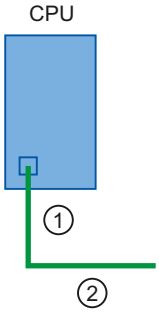

① Assigned S7 connection resource on the CPU

② Subnet (e.g., MPI, PROFIBUS-DP, or Industrial Ethernet/PROFINET)

- S7-300/400 CPU: One S7 connection resource is assigned on the CPU.
- S7-1200/1500 CPU: One S7 connection resource is assigned on the CPU and thus also one S7 connection resource on the station.

S7 connections via external interfaces

With S7 connections via an external CP interface, a distinction should be made between the following scenarios:

- S7-300 CPU
  - S7 connection configured at one end: One S7 connection resource is assigned on the CPU (for the endpoint) and one S7 connection resource on the CP (transition point) for each S7 connection.

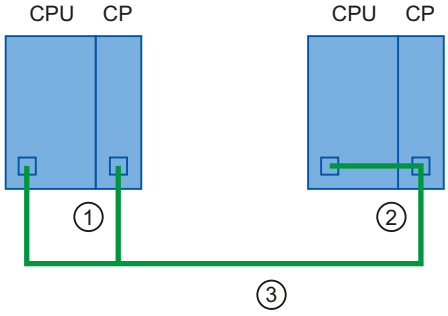

① One S7 connection resource in the CPU and one in the CP of the local end point
② Connection partner
③ Subnet, e.g., Industrial Ethernet

  - S7 connection configured at both ends: One S7 connection resource is assigned on the CP for each S7 connection.

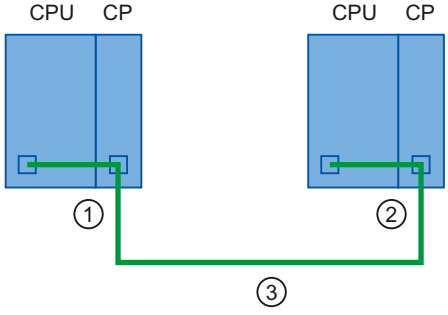

① One S7 connection resource in the CP of the local end point
② Connection partner
③ Subnet, e.g., Industrial Ethernet

- S7-400 CPU
  One S7 connection resource is assigned on the CPU (for the endpoint) and one S7 connection resource on the CP (transition point) for each S7 connection.

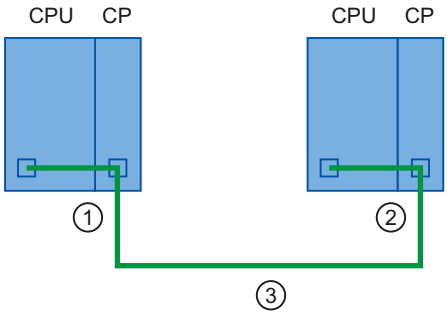

① One S7 connection resource in the CPU and one in the CP of the local end point
② Connection partner
③ Subnet, e.g., Industrial Ethernet

- S7-1200/1500 CPU
  - S7 connection configured at one end: One S7 connection resource is assigned on the CPU and thus also one S7 connection resource on the station for each S7 connection.

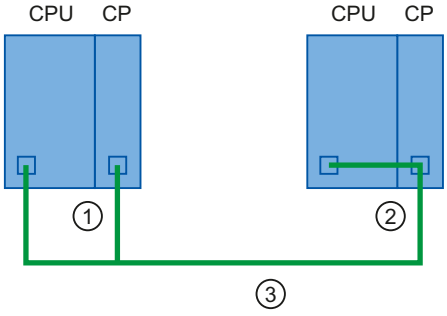

① One S7 connection resource in the CPU and one in the CP of the local end point
② Connection partner
③ Subnet, e.g., Industrial Ethernet

  - S7 connection configured at both ends: One S7 connection resource is assigned on the CPU and thus also one S7 connection resource on the station for each S7 connection.

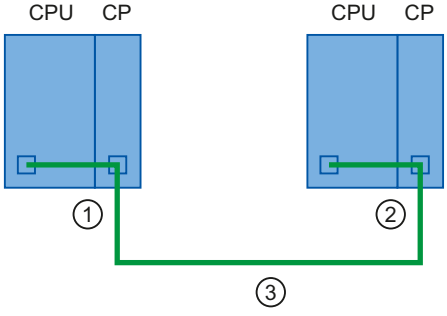

① One S7 connection resource in the CP and, thus, also one in the station
② Connection partner
③ Subnet, e.g., Industrial Ethernet

Note

The number of reserved S7 connection resources for the selected CPU for certain CPU types of the S7-300, and for the selected CPU and its connected communication modules for the S7-1200/1500, can be seen in the area navigation under "Connection resources". S7 connection resources for OP/PG and S7 basic communication can be reserved here.

Connections for open communication services (SEND/RECEIVE interface)

Communication via the SEND/RECEIVE interface is solely via CPs. For this, one protocol-specific connection resource is allocated for the endpoint per connection (i.e., FDL, ISO transport ISO on TCP, UDP and TCP connection).

Here, no connection resources are required for the connection on the CPU.

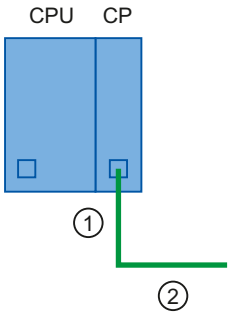

① 1 assigned connection resource in the CP

② Subnet, e.g., Industrial Ethernet

Exception: on the S7-400 CPU, connections with the property "SPEED SEND/RECEIVE" require an additional S7 connection resource for each "SPEED SEND/RECEIVE" connection on the CPU.

Connection resources and connection type

Setting up the connection

Often, you can decide between a configured setup or a programmed setup. The programmed setup makes it easier to release connection resources following data transfer. With configured setup, you receive help in managing connection resources in STEP 7.

Table 1-2 Setting up the communication service

Communication service	Automatic setup	Programmed setup	Configured setup
PG communication	X	-	-
HMI communication	X	-	X
S7 communication	-	X	X
Open communication via TCP/IP	-	X	X

Communication service	Automatic setup	Programmed setup	Configured setup
Open communication via ISO-on-TCP	-	X	X
Open communication via UDP	-	X	X
Open communication via ISO	-	X	X
Open communication via e-mail	-	X	-
Open communication via FTP	-	X	-
Open communication via FDL	-	X	X
S7 routing	-	-	X

Assignment of connection resources according to the connection setup

Connection resources are assigned in different ways, depending on the type of connection setup:

- Configured setup
  In the case of configured connections, the connection resource is assigned as soon as the hardware configuration has been downloaded to the CPU. For OPC servers, the resources are assigned, as required.
- Programmed setup
  The call of the appropriate instructions for establishing the connection (using TSEND_C or T_CON) triggers assignment of a connection resource.
- Automatic setup
  As soon as you have connected the programming device or HMI device to a CPU physically and online, the connection resources will be assigned.

Connection resources for configured setup

You set up the connection in the network view of the hardware and network editor in the context of a CPU.

Following the data transfer, the connection is not terminated, i.e., the connection resources remain permanently assigned. To terminate the connection you must delete the connection configuration and download the modified configuration to the CPU.

OPC connections are automatically established and terminated as needed.

If you have **configured** the connections, compliance with the maximum possible number of connection resources in the automation system will be monitored.

Connection resources for programmed setup

You set up the connection in the program editor in the context of a CPU by parameter assignment of instructions for communication, e.g., TSEND_C.

Once the connection is established by the TCON or TSEND_C instruction, the connection will be terminated through appropriate parameter assignment of the TSEND_C instruction or by calling the TDISCON instruction after the data transfer. If the connection is terminated, the connection resources in the CPU/CP/CM are available again. If the connection is retained and

another data transfer is performed, the processing time of the instructions will be shorter because the connection does not have to be reestablished.

If the establishment and termination of connections are **programmed** in the user program, you are responsible for ensuring compliance with the automation system limits. If necessary, you will receive a communication error at the "STATUS" output parameter of the TSEND_C or T_CON instruction and can obtain more detailed information from the connection diagnostics.

Connection resources for automatic setup

The connection will be set up automatically (e.g., programming device or HMI connection) as soon as the PG/PC interface has been physically connected to an interface of the CPU and the assignment to the interface has been made in the "Go online" dialog.

For **automatically** set up connections, compliance with the connection resources reserved in the automation system for these connections is monitored.

S7-1200 connection resources:

Display of connection resources

Connection resources in the Inspector window

You will find an overview of the reserved and available connection resources of an S7-1200 automation system in the inspector window of the hardware and network editor under the CPU properties.

Overview of connection resources

The CPU supports at most the following number of simultaneous asynchronous communication connections for PROFINET and PROFIBUS. The maximum number of connection resources assigned to each category is predefined. You cannot modify the values. The 6 freely available connections can, however, be used to increase the number of connections in each category as required by your application.

The display of connection resources distinguishes between the offline and online view. The following figure shows you the offline display of the reserved and available connection resources of a CPU 1217C:

Connection resources

	① Station resources			② Module resources
	Reserved		Dynamic ⚠	PLC_11 [CPU 1217C DC/DC/DC]
Maximum number of resources:	62		6	68
	Maximum	Configured	Configured	Configured
PG communication:	4	-	-	-
HMI communication:	12	0	0	0
S7 communication:	8	0	0	0
Open user communication:	8	0	0	0
Web communication:	30	-	-	-
Other communication:	-	-	0	0
Total resources used:		0	0	0
Available resources:		62	6	68

① Station-specific connection resources

② Module-specific connection resources

Station-specific connection resources

The columns of the station-specific connection resources provide information about the reserved and dynamically available connection resources of the station. In the example, a maximum of 68 station-specific connection resources are available for the automation system.

- 62 connection resources are reserved for specific connection types
- 6 connection resources can be used freely for connection types

The warning triangle in the column of the dynamic station resources is displayed because the sum of the maximum available connection resources of the CPU has reached the station limit of 68.

Note

Exceeding the available connection resources

STEP 7 signals the reaching of the station-specific connection resources with a warning. If you require more connection resources, you either have to use a CPU with a higher maximum number of available station-specific connection resources or reduce the number of communications connections.

Module-specific connection resources

The columns of the module-specific connection resources provide information about the allocation of resources on the CPUs, CPs and CMs of an automation system: The display is per module and not per interface. In the example, the CPU provides a maximum of 68 connection resources.

Note

If you add CM/CP modules, the total number of S7-1200 communication resources is not increased.

Online view

If you are connected to the CPU online, you can also see how many resources are currently being used in the "Connection information" tab of the "Diagnostics" section in the Inspector window.

The online view of the "Connection resources" table not only displays the reserved and dynamic connection resources of the configured connections, but also includes columns containing the connection resources of the programmed and automatically set-up connections that are currently allocated in the station. Thus, the online view displays **all** the assigned connection resources in the automation system, regardless of how the connection was set up. The "Other communication" row displays connection resources assigned for communication with external devices and communication via data record routing.

The table is updated automatically. If there are inconsistencies between the configured hardware configuration and the hardware configuration available online, a corresponding message will be output.

Resource requirements of the connections

The communication connections of open user communication, S7 connection, HMI, programming device and Web server (HTTP) can utilize several connection resources on the basis of the functions used.

The following number of connections are available per device on the basis of the assigned connection resources:

	Programming device (PG)	HMI devices	GET/PUT client/ server	Open user communication	Web browser
Maximum number of connection resources of a CPU 1217C	4 (guaranteed support of a programming device connection)	12 (guaranteed support of four HMI connections)	8	8	30 (guaranteed support of three HTTP connections)

Programming device communication

For example, four connection resources are available for programming device communication. Depending on the currently used programming device functions, the programming device occupies up to three connection resources for a connection. With the S7-1200, at least one programming device connection is guaranteed at all times.

HMI communication

A further example is the number of HMI devices as shown in the following figure. 12 connection resources are available for HMI devices. Depending on the type or model of your HMI device and the HMI functions used, the HMI device can require up to three connection resources for an HMI connection. For example, a Basic Panel requires up to two connection resources in the CPU for an HMI connection, while a Comfort Panel requires up to three connection resources in the CPU. Depending on the number of required connection resources, at least four HMI connections can be configured simultaneously.

An HMI device can use its available connection resources for the following functions:

- Reading
- Writing
- Interrupts and diagnostics

This is only an example. The actual number of connections can differ depending on the type and version of the HMI device.

Example	HMI 1	HMI 2	HMI 3	HMI 4	HMI 5	Total number of available connection resources
Used connection resources	2	2	2	3	3	12

Information regarding the availability and assignment of connection resources for HMI connections is available in the offline view in the HMI device context (under "Properties > General > Connection resources" in the Inspector window):

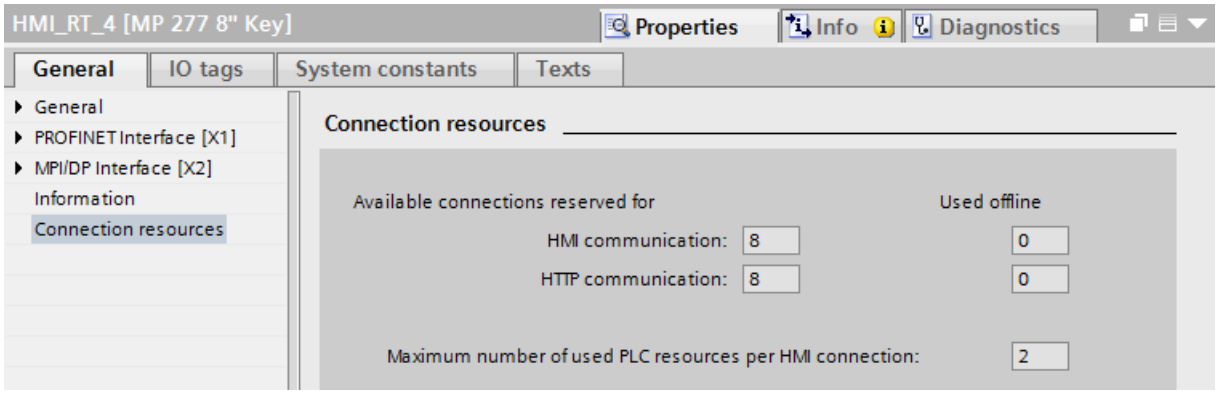

The following is displayed in the connection resources area:

- Number of available connections on the HMI reserved for HMI communication and HTTP communication
- Number of connection resources for HMI communication and HTTP communications used offline in the HMI
  If the maximum number of available connection resources for an HMI device is exceeded, a corresponding alarm is output by STEP 7.
- Maximum number of used PLC resources per HMI connection. This parameter is a factor that is to be multiplied by the number of HMI connections used offline. The product is the number of HMI resources occupied on the CPU.

If the maximum number of available connection resources for an HMI device is exceeded, a corresponding alarm is output.

Web communication

The CPU offers Web server (HTTP) connections for several Web browsers. The number of browsers that can be supported simultaneously by the CPU depends on the number of connections that a Web browser requests/utilizes.

Connection resources S7-1500

Connection resources in the automation system

Introduction

Some communications services require connections. Connections occupy resources in the automation system. The connection resources are made available by the CPUs, communications processors (CPs) and communications modules (CMs).

Connection resources of a station

The connection resources available depend on the CPUs, CPs and CMs and may not exceed a maximum number per station. The maximum number of resources of a station is determined by the CPU.

Each CPU has reserved connection resources for PG, HMI and Web server communication. This ensures, for example, that a PG can always establish at least one online connection with the CPU regardless of how many other communications services are already using connection resources. In addition, there are dynamic resources. The difference between the maximum number of connection resources and the number of reserved connection resources is the maximum number of dynamic connection resources. The communication services PG communication, HMI communication, S7 communication, Open User Communication, Web communication and other communication (such as OPC UA) use the dynamic connection resources from the pool.

The figure below shows an example of how individual components make connection resources available to an S7-1500 station.

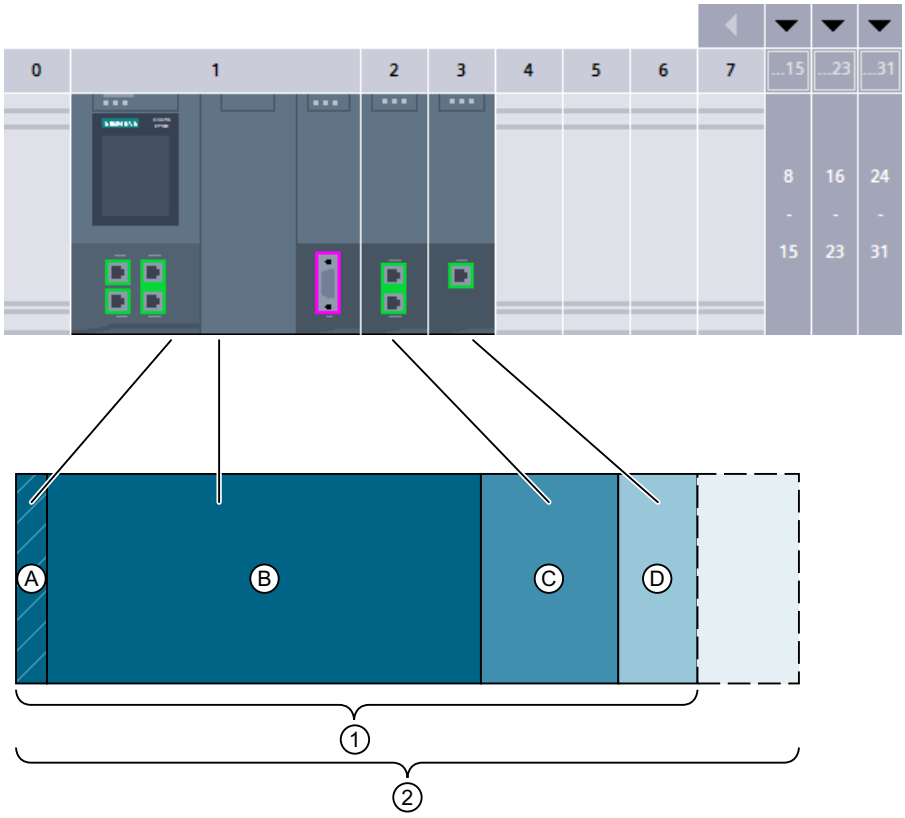

① Available connection resources of the station, of which

A	Reserved connection resources of the station
A + B	Connection resources of CPU 1518
C	Connection resources of communications module CM 1542-1
D	Connection resources of communications processor CP 1543-1

② Maximum communications resources of the station using the example of a configuration from CPU 1518, CM 1542-1 and CP 1543-1

Number of connection resources of a station

The following table lists the maximum number of available connection resources for some CPU types:

Connection resources of a station	1511 1511C	1512C 1513	1515	1516	1517	1518
Maximum connection resources of the station	96	128	192	256	320	384
of which reserved	10					
of which dynamic	86	118	182	246	310	374
Connection resources of the CPU	64	88	108	128	160	192
Max. additionally usable connection resources by plugging in CMs/CPs	32	40	84	128	160	192

Connection resources of a station	1511 1511C	1512C 1513	1515	1516	1517	1518
Additional connection resources CM 1542-1	64					
Additional connection resources CP 1543-1	118					
Additional connection resources CM 1542-5	40					
Additional connection resources CP 1542-5	16					

The number of connection resources supported by a CPU or a communication module is described in the technical specifications section of the manuals.

Example

You have configured a CPU 1518-4PN/DP with a communications module CM 1542-1 and a communications processor CP 1542-5.

- Maximum connection resources of the station: **384**
- Available connection resources:
  - CPU 1518-4 PN/DP: 192
  - CM 1542-1: 64
  - CP 1542-5: 16
  - Total: **272**

The setup provides 272 connection resources. By adding further communication modules the station can support a maximum of 112 additional connection resources.

Reserved connection resources

10 connection resources are reserved for stations with S7-1500 CPU, ET 200SP CPU and ET 200pro CPU based on S7-1500:

- 4 for PG communication required by STEP 7, for example, for test and diagnostics functions or downloading to the CPU
- 4 for HMI communication which are occupied by the first HMI connections configured in STEP 7
- 2 for communication with the Web server

Allocation of connection resources

Overview - occupation of connection resources

The following figure shows how different connections occupy the resources of the S7-1500:

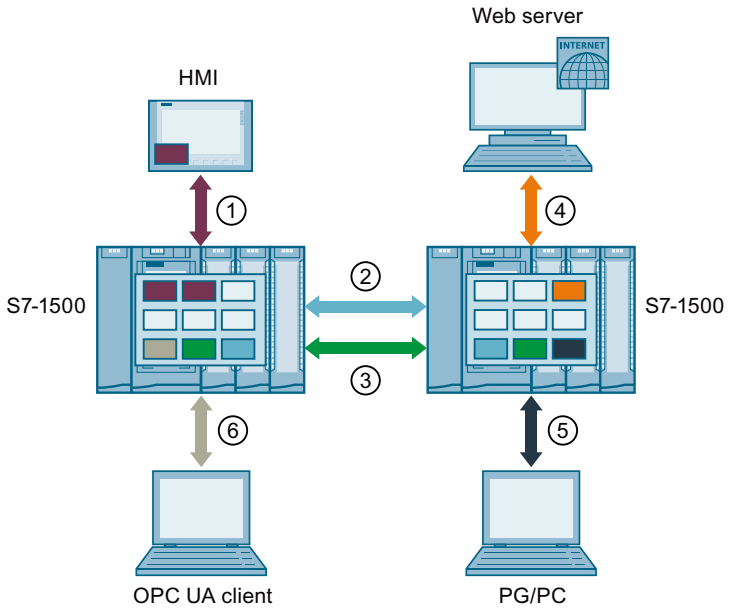

① HMI communication: See below.

② Open User Communication: Connections of Open User Communication occupy a connection resource in every end point.

③ S7 communication: Connections of S7 communication occupy a connection resource in every end point.

④ Web communication: The Web server connection occupies at least one connection resource in the station. The number of occupied connections depends on the browser.

⑤ PG communication: The PG connection occupies one connection resource in the station.

⑥ OPC UA communication: Each session that the OPC UA server of the CPU establishes with an OPC UA client usually occupies one connection resource (other communication) in the station.

Connection resource for HMI communication

Connection resource for Open User Communication

Connection resource for S7 communication

Connection resource for Web communication

Connection resource for PG communication

Connection resource for other communication (for example OPC UA)

Connection resources for HMI communication

With HMI communication, the occupation of connection resources in the station depends on the HMI device being used. The following table lists the maximum number of occupied connection resources for an HMI connection depending on the various HMI devices:

HMI device	Maximum number of allocated connection resources of the station for each HMI connection
Basic Panel	1
Comfort Panel	2[1]
RT Advanced	2[1]
RT Professional	3

[1] If you do not use system diagnostics or alarm configuration, the station occupies only one connection resource per HMI connection.

Example: You have configured the following HMI connections for a CPU 1516-3 PN/DP:

- Two HMI connections to an HMI TP700 Comfort (2 connection resources each)
- One HMI connection to an HMI KTP1000 Basic (1 connection resource)

A total of 5 connection resources are allocated for HMI communication in the CPU.

Connection resources for routing

S7 connections beyond network boundaries (S7 routing)

To transfer data beyond S7 subnets ("S7 routing"), an S7 connection is established between two CPUs. The S7 subnets are connected via gateways known as S7 routers. The router is the transition point of an S7 connection and is capable of establishing S7 connections. CPUs, CMs and CPs in S7-1500 are S7 routers.

The following applies for a routed S7 connection:

- A routed connection occupies one connection resource each in both end points. STEP 7 shows these connection resources in the "Connection resources" table.
- On the S7 router, two special connection resources are occupied for S7 routing. STEP 7 does not display the special connection resources for S7 routing in the "Connection resources" table. The number of resources for S7 routing depends on the CPU. You will find the resources for S7 routing in the technical specifications of the CPU in "Number of S7 routing connections".

The following figure shows the allocation of the connection resources at S7 routing:

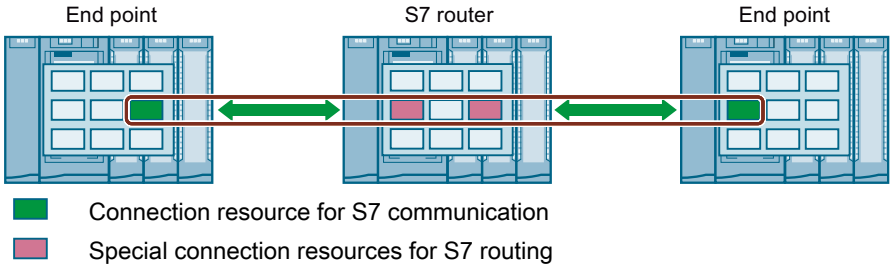

Data record routing

Data record routing also enables transfer of data beyond S7 subnets from an engineering station connected to PROFINET to various field devices via PROFIBUS.

With data record routing, as with S7 routing, two of the special connection resources for S7 routing are also occupied at the network transitions in every record router. At the end point of communication (CPU), connection resources for data record routing are displayed in the online view of the "Connection resources" table in the "Other communication" row.

Note

Connection resources with data record routing

With data record routing, on the data record router, two special connection resources for S7 routing are occupied. Neither the data record connection nor the allocated connection resources are displayed in the table of connection resources. You are responsible for observing the limits during configuration.

When are connection resources occupied?

The time for the occupation of connection resources depends on how the connection is set up.

- Programmed setup of a connection
  As soon as an instruction to establish a connection is called in the user program of the CPU (TSEND_C/TRCV_C or TCON), a connection resource is occupied. With suitable parameter assignment of the CONT parameter of the TSEND_C/TRCV_C instructions or by calling the TDISCON instruction, the connection can be terminated following data transfer and the connection resource is available again. If the connection is terminated, the connection resources on the CPU/CP/CM are available again.
- Configured connections (e.g. HMI connection)
  If you have configured a connection in STEP 7, the connection resource is occupied as soon as the hardware configuration is loaded to the CPU. After using a configured connection for data transfer, the connection is not terminated. The connection resource is permanently occupied. To release the connection resource again, you need to delete the configured connection in STEP 7 and download the modified configuration to the CPU.
- Programming device connection
  As soon as you have connected the programming device to a CPU online in STEP 7, connection resources are occupied.
- Web server
  As long as you have opened the Web server of the CPU in a browser, connection resources are occupied in the CPU.
- OPC UA server
  As long as a session exists between the OPC UA server of the CPU and an OPC UA client, a connection resource is occupied in the CPU.

Monitoring the maximum possible number of connection resources

Offline

During configuration of connections, STEP 7 monitors the occupation of the connection resources. If the maximum possible number of connection resources is exceeded, STEP 7 signals this with a suitable warning.

Online

The CPU monitors the use of connection resources in the automation system. If you establish more connections in the user program than those provided by the automation system, the CPU acknowledges the instruction to establish the connection with an error.

Display of the connection resources

Connection resources in the Inspector window

You will find an overview of the reserved available connection resources of an S7-1500 automation system in the Inspector window of the hardware and network editor under the CPU properties.

Display of the connection resources in STEP 7 (offline view)

The display of connection resources distinguishes between the offline and online view. The following figure shows you the offline display of the reserved and available connection resources of an S7-1500 CPU with a CP and a CM:

Connection resources

	① Station resources			② Module reso...	Module reso..	Module reso...
	Reserved		Dynamic ⚠	PLC_1 [CPU ...	CP 1543-1_...	CM 1542-5_...
Maximum number of resources:		10	246	128	118	48
	Maximum	Configured	Configured	Configured	Configured	Configured
PG communication:	4	-	-	-	-	-
HMI communication:	4	4	6	6	4	0
S7 communication:	0	-	23	2	0	21
Open user communication:	0	-	52	39	13	0
Web communication:	2	-	-	-	-	-
Other communication:	-	-	0	0	0	0
Total resources used:		4	81	47	17	21
Available resources:		6	165	81	101	27

① Station-specific connection resources

② Module-specific connection resources

Station-specific connection resources

The columns of the station-specific connection resources provide information about the reserved and dynamically available connection resources of the station. In the example, a maximum of 256 station-specific connection resources are available for the automation system.

- 10 reserved connection resources, of which 4 are already in use and a further 6 still available.
  The occupied 4 resources are used for HMI communication.
- 246 dynamic connection resources, of which 81 are already in use and a further 165 still available.
  The configured resources are divided up as follows:
  - 6 resources for HMI communication
  - 23 resources for S7 communication
  - 52 resources for Open User Communication

The warning triangle in the column of the dynamic station resources is therefore displayed because the sum of the maximum available connection resources of CPU, CP and CM (= 294 connection resources) exceeds the station limit of 256.

Note

Exceeding the available connection resources

STEP 7 signals the exceeding of the station-specific connection resources with a warning. To make full use of the connection resources from the CPU, CP and CM, either use a CPU with a higher maximum number of available station-specific connection resources or reduce the number of communications connections.

Module-specific connection resources

The columns of the module-specific connection resources provide information about the allocation of resources on the CPUs, CPs and CMs of an automation system: The display is per module and not per interface. In the example, the CPU makes a maximum of 128 connection resources available, of which 47 are already in use and a further 81 still available. The configured resources are divided up as follows:

- 6 resources for HMI communication
- 2 resources for S7 communication
- 39 resources for Open User Communication

The CP makes 118 resources and the CM 48 resources available, of which 17 resources and 21 resources respectively are used.

Display of the connection resources in STEP 7 (online view)

If you are connected to the CPU online, you can also see how many resources are currently being used in the "Connection information" tab of the "Diagnostics" section in the Inspector window. The following figure shows you the online display of the reserved, available and used connection resources of an S7-1500 CPU:

PLC_2 [CPU 1516-3 PN/DP] | Properties | Info | Diagnostics

Device information | Connection information | Alarm display

Connection resources

	Station resources					Module resources	
		Reserved		Dynamic		PLC_2 [CPU 1516-3 PN/DP]	
Maximum number of resources:		10	10	194	194	86	86
	Maximum	Configured	Used	Configured	Used	Configured	Used
PG communication:	4	-	4	-	0	-	0
HMI communication:	4	4	4	4	4	8	8
S7 communication:	0	0	0	72	68	34	34
Open user communication:	0	0	0	118	118	45	45
Web communication:	2	-	2	-	0	-	0
Other communication:	-	-	0	-	0	-	0
Total resources used:		4	10	194	190	82	82
Available resources:		6	0	0	4	4	4

The online view of the "Connection resources" table not only displays the reserved and dynamic connection resources of the configured connections, but also includes columns containing the connection resources of the programmed and automatically set-up connections that are currently allocated in the station. Thus, the online view displays **all** the assigned connection resources in the automation system, regardless of how the connection was set up. The "Other communication" row displays connection resources assigned for communication with external devices and communication via data record routing.

The table is updated automatically. If there are inconsistencies between the configured hardware configuration and the hardware configuration available online, a corresponding message will be output.

Note

If a routed S7 connection goes through a CPU, the required connection resources of the CPU do not appear in the table of connection resources!

Display of the connection resources for HMI

For HMI communication, connection resources in the CPU are assigned according to the HMI device used and the applications in which the HMI device is used. For example, a Basic Panel requires up to two connection resources in the CPU for an HMI connection, while a Comfort Panel requires up to three connection resources in the CPU.

Information regarding the availability and allocation of connection resources for HMI connections is available in the offline view in the HMI device context (under "Properties > General > Connection resources" in the Inspector window).

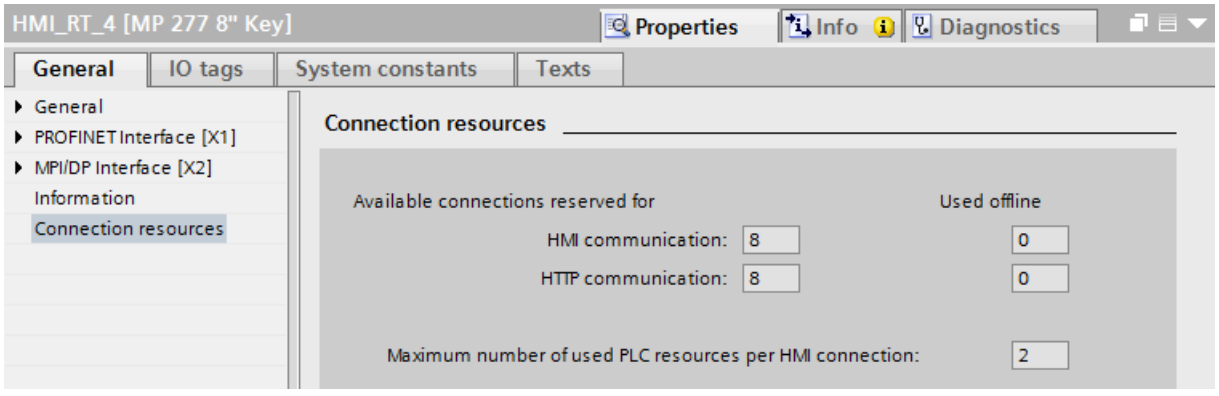

The following is displayed in the connection resources area:

- Number of available connections on the HMI reserved for HMI communication and HTTP communication
- Number of connection resources for HMI communication and HTTP communications used offline in the HMI
  If the maximum number of available connection resources for an HMI device is exceeded, a corresponding alarm is output by STEP 7.
- "Maximum number of used PLC resources per HMI connection". This parameter is a factor that is to be multiplied by the number of HMI connections used offline. The product is the number of HMI resources occupied on the CPU.

If the maximum number of available connection resources for an HMI device is exceeded, a corresponding alarm is output.

Displaying the connection resources in the Web server

You can display the connection resources not only in STEP 7, but also with a browser that displays the relevant page of the Web server. You will find information on displaying connection resources in the Web server in the function manual for the Web server.

Example for calculating the connection resources

Introduction

This example for the table displaying connection resources shows you how the individual table entries are put together.

Requirement

- 2x CPU 1516-3 PN/DP each with one CP 1543-1 and one CM 1542-5as connection partners for the purpose of extending the configuration limit.
- 1 x PC system WinCC RT Professional with CP 1616 onboard as connection partner for HMI connections
- The devices are networked together.

Creating the connections

Once the connection partners (CPUs, CPs, CMs, WinCC RT Professional) have been created and networked in the network view of the hardware and network editor, follow these steps:

1. Switch to connection mode in the network view.
2. Create the following connections:
   - S7 connections: 2x via CPU, 21x via CM
   - TCP connections: 2x via CPU, 4x via CM
   - ISO connection: 6x via CP
   - ISO-on-TCP: 3x via CPU, 4x via CM
   - UDP connections: 4x via CPU, 106x via CP
   - HMI connections: 1x via CPU, 2x via CP

Result

The display of connection resources shows you not only the available resources but also the resources required for the created connections.

The available resources of devices and modules are highlighted in the following figure.

	Station resources			Module res...	Module res...	Module res...
	Reserved		Dynamic ③ ⚠	PLC_2 [CPU ..	CP 1543-1_...	CM 1542-5_...
Maximum number of resources:	① 10		246	② 128	118	118
	Maximum	Configured	Configured	Configured	Configured	Configured
PG communication:	4	-	-	-	-	-
HMI communication:	4	4	5	3	6	0
S7 communication:	0	-	23	2	0	21
Open user communication:	0	-	129	9	112	8
Web communication:	2	-	-	-	-	-
Other communication:	-	-	-	-	-	-
Total resources used:		4	157	14	118	29
Available resources:		6	89	114	0	89

① Depending on the CPU, at most 256 connections are available for this station. Ten (10) resources are reserved for particular connection types and 246 resources are dynamically available.

② The CPU provides 128 connection resources, and the CP and CM each provide another 118 resources.

③ CPU, CP, and CM altogether provide 128 + 118 + 118 = 364 resources. Because more connection resources are provided via the devices and modules than the station can operate with its maximum limit of 256 connection resources, a warning icon is displayed.

The following figure highlights the resources utilized by the created connections:

	Station resources			Module res...	Module res...	Module res...
	Reserved		Dynamic	PLC_2 [CPU ..	CP 1543-1_...	CM 1542-5_..
Maximum number of resources:		10	246	128	118	118
	Maximum	Configured	Configured	Configured	Configured	Configured
PG communication:	4	-	-	-	-	-
HMI communication:	4	4	5	3	6	0
S7 communication:	0	-	23	2	0	21
Open user communication:	0	-	129	9	112	8
Web communication:	2	-	-	-	-	-
Other communication:	-	-	-	-	-	-
Total resources used:		4	157	14	118	29
Available resources:		6	89	114	0	89

① These are the resources assigned via the HMI connections. In this example, one HMI connection utilizes 3 connection resources. For the whole station, 3 + 6 = 9 connection resources are required for the HMI communication.

② Four (4) resources are reserved for the HMI communication.

③ The 9 resources required for HMI communication are broken down as follows: For the first 4 resources, the resources already reserved for HMI communication will be used. The other 5 required resources will be taken from the dynamic resources.

④ These are the resources assigned via the S7 connections. An S7 connection occupies 1 connection resource. For the whole station, 2 + 21 = 23 connection resources are required.

⑤ For the S7 communication, there are no reserved resources. All 23 required resources come from the dynamically available resources.

⑥ These are the resources of the Open User Communication (for TCP, FDL, ISO, ISO-on-TCP, and UDP). For the whole station, 9 + 112 + 8 = 129 connection resources are required.

⑦ For the Open User Communication, there are no reserved resources. All 129 required resources come from the dynamically available resources.

Calculation of connection resources

Based on the available and utilized resources of individual devices and modules, the resources that are still free and the total resources used are calculated:

	Station resources			Module res...	Module res...	Module res...
	Reserved		Dynamic	PLC_2 [CPU ..	CP 1543-1_...	CM 1542-5_..
Maximum number of resources:		10	246	128	③ 118	118
	Maximum	Configured	Configured	Configured	Configured	Configured
PG communication:	4	-	-	-	-	-
HMI communication:	4	4	5	3	6	0
S7 communication:	0	-	23	① 2	0	21
Open user communication:	0	-	129	9	112	8
Web communication:	2	-	-	-	-	-
Other communication:	-	-	-	-	-	-
Total resources used:		4	157	② 14	④ 118	29
Available resources:		6	89	114	⑤ 0	89

① All required resources for the respective communication types of the CPU are displayed here. Three (3) resources are required for HMI communication, 2 for S7 communication, and 9 for the connection types of the Open User Communication.

② The total number of resources required is listed below the required resources for the communication types. For the CPU, therefore, 3 + 2 + 9 = 14 resources are required.

③ The CP 1543-1 has 118 available connection resources. If these are utilized, no more connections can be created for the CP.

④ For the CP 1543-1, 118 connection resources are already utilized by the 2 HMI connections (3 resources each = 6 resources) and 112 resources by connections of the open communication.

⑤ The CP 1543-1 has no more free resources. No more connections can be created for the CP 1543-1.

The number of resources still available and the total number of resources used is also displayed for the whole station:

	Station resources			Module res...	Module res...	Module res...
	Reserved		Dynamic	PLC_2 [CPU ..	CP 1543-1_...	CM 1542-5_..
Maximum number of resources:		10	246	128	118	118
	Maximum	Configured	Configured	Configured	Configured	Configured
PG communication:	4	-	-	-	-	-
HMI communication:	4	4	5	3	6	0
S7 communication:	0	① -	23	2	0	21
Open user communication:	0	-	129	9	112	8
Web communication:	2	-	-	-	-	-
Other communication:	-	-	-	-	-	-
Total resources used:		② 4	157	14	118	29
Available resources:		③ 6	89	114	0	89

① All connection resources used (configured), including both reserved and dynamic connection resources, for the individual communication types for the entire station. The values represent the sum of all resources utilized for devices and modules (CPU, CP, CM).

② All connection resources used (configured), including both reserved and dynamic connection resources, for all communication types for the entire station.

③ Total number of reserved and dynamic connection resources still available. The values are obtained by subtracting the number of resources used from the maximum number of resources that can be configured.

Data consistency

Definition

Data consistency is important for data transfer and you need to take this into account when configuring the communication task. Otherwise, malfunctions may occur.

A data area which cannot be modified by concurrent processes is called a consistent data area. This means that a data area which belongs together and which is larger than the maximum size of the consistent data area can consist in part of new and of old data at the same time.

An inconsistency can occur when an instruction for communication is interrupted, for example by a hardware interrupt OB with higher priority. This interrupts the transfer of the data area. If the user program in this OB now changes the data that has not yet been processed by the communication instruction, the transferred data originates from different times:

The following figure shows a data area that is smaller than the maximum size of the consistent data area. In this case, when transferring the data area, it is ensured that there is no interruption by the user program during data access so that the data is not changed.

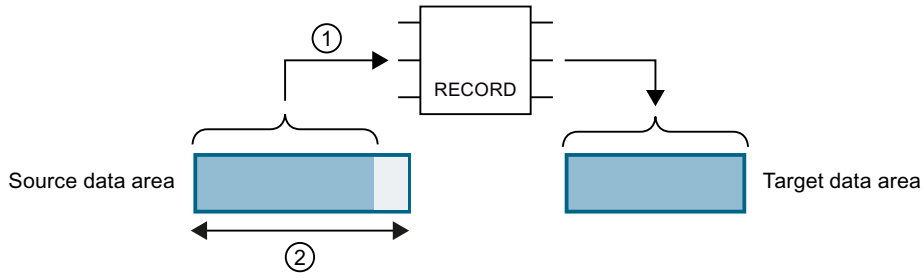

① The source data area is smaller than the maximum size of the consistent data area (②). The instruction transfers the data together to the destination data area.

② Maximum size of the consistent data area

Figure 1-1 Consistent transfer of data

The following figure shows a data area that is larger than the maximum size of the consistent data area. In this case, the data can be changed during an interruption of the data transfer. An interruption also occurs if, for example, the data area needs to be transferred in several parts. If the data is changed during the interruption, the transferred data originates from different times.

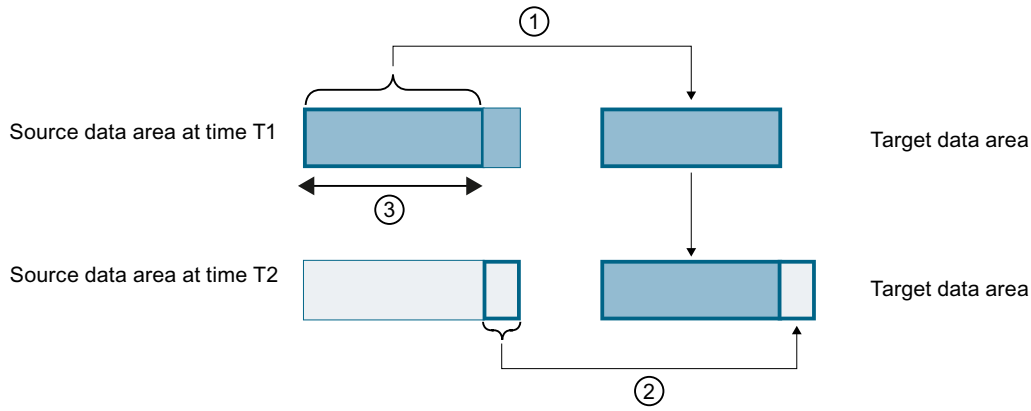

① The source data area is larger than the maximum size of the consistent data area (③). At time T1, the instruction only transfers as much data from the source data area to the destination data area as fit in the consistent data area.

② At time T2, the instruction transfers the rest of the source data area to the destination data area. After the transfer, data from different points in time exist in the destination data area. If the data in the source data area has changed in the meantime, an inconsistency may result.

③ Maximum size of the consistent data area

Figure 1-2 Transfer of data larger than the maximum consistency area

Example of an inconsistency

The figure below shows an example of changing data during the transfer. The destination data area contains data from different points in time.

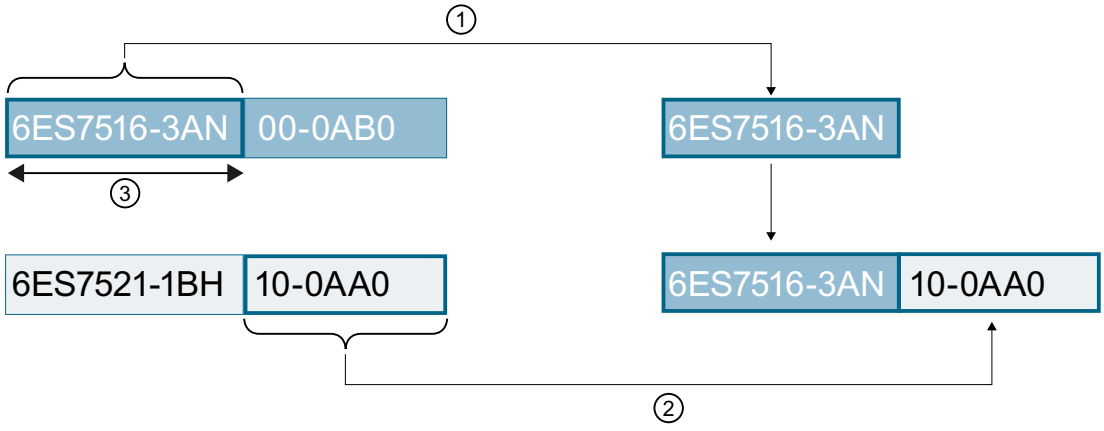

Figure 1-3 Example: Changing data during the transfer

System-specific maximum data consistency for S7-1500:

No inconsistency occurs if the system-specific maximum size of the consistent data is kept to. With an S7-1500, communication data is copied consistently into or out of the user memory in blocks of up to 512 bytes during the program cycle. Data consistency is not ensured for larger data areas. Where defined data consistency is required, the length of communication data in the user program of the CPU must not exceed 512 bytes. You can then access these data areas consistently, for example from an HMI device by reading/writing tags.

If more data than the system-specific maximum size needs to be transferred consistently, you yourself must ensure the data consistency with suitable measures in the user program.

Ensuring data consistency

Use of instructions for access to common data:

If the user program contains instructions for communication that access common data, for example TSEND/TRCV, you can coordinate access to this data area yourself, for example using the "DONE" parameter. The data consistency of the data areas that are transferred with an instruction for communication can therefore be ensured in the user program.

Note

Measures in the user program

To achieve data consistency, you can copy transferred data to a separate data area (for example global data block). While the user program continues to work with the original data, you can transfer the data saved in the separate data area consistently with the communication instruction.

For the copying, use uninterruptible instructions such as UMOVE_BLK or UFILL_BLK. These instructions ensure data consistency up to 16 KB.

Use of PUT/GET instructions or Write/Read via HMI communication:

In S7 communication with the PUT/GET instructions or Write/Read via HMI communication, you need to take into account the size of the consistent data areas during programming or configuration. In the user program of an S7-1500 as server, there is no instruction available that can coordinate the data transfer in the user program. The data exchanged using PUT/GET instructions updates the S7-1500 while the user program is running. There is no point in time within the processing of the cyclic user program at which the data is exchanged consistently. The length of the data area to be transferred should be smaller than 512 bytes.

Additional information

- You will also find the maximum amount of consistent data in the CPU, CM or CP manuals in the Technical Specifications.
- You will find further information on data consistency in the description of the instructions in the STEP 7 online help.

1.3.2.2 Connections - Purpose and selection (S7-300, S7-400, S7-1500)

Introduction to configuring connections (S7-300, S7-400, S7-1500)

Why set up connections?

A communication connection, or connection, defines a logical assignment of two communication partners for performing communication services.

You can configure communication connections and under certain circumstances even set them up in a program-controlled manner.

This chapter shows you how to define connections, which special aspects you need to take into account, and which communication instructions you can use in the user program.

Definition

A connection defines the following:

- Communication partner involved
- Type of connection (for example S7, ISO-on-TCP, ISO-Transport, FDL, UDP, e-mail or TCP connection)
- Special properties, such as
  - Setup characteristics
    A connection can remain permanently enabled or, if necessary, can be enabled and disabled.
  - Sending operating mode messages
- Communication path
  - within the devices (CPU - CP)
  - in the network (subnets / router)

What you need to know about connection configuration

During connection configuration, one unique local ID is issued per connection. Only this local ID is needed when configuring the communication modules. The local ID is clearly assigned to the addresses of the local and remote connection partners.

If a connection endpoint refers to the application of a PC station, the connection name is used as "Local ID" for connection identification.

After the project is opened, the project-wide connection table is displayed. If you deselect the "Display all connections" check box in the shortcut menu of the connection table, only object-specific connections of selected objects are displayed. The created connections can be displayed separately for the following objects:

- Programmable modules that can be the endpoint of a connection
- Communications modules (CPs) used on the connection path
- Subnets

The addresses of the local and remote connection partners are set when the devices are networked and assigned to the connections during configuration. The following connections are an exception to this rule:

- Available UDP connection
  For the free UDP connection, the address at the communications interface is assigned in the user program.
- Programmed communication connection

Note

The term "connection" is used under SIMATIC communication also with UDP. This is because the communication partners are assigned to one another and therefore logically "connected" during configuration. With the UDP, there is no explicit establishing of a connection between the communication partners.

Networking and the path selection determine the configuration status

Depending on whether the connection partners are present in the project and networked, the address parameters of the connection endpoints can be fully or only partly specified.

- Fully specified
  - The addresses and network parameters of the local and remote communication partners are defined.
  - The communications connection is ready for operation after the connection parameters have been loaded to the device.
- Unspecified (partly specified)
  Only the local communication partner is defined. The partner for the selected connection type is either disconnected, is outside the project or is not defined because it requires special handling.
  Application case:
  - The complete networking of the devices is not yet possible, as the hardware configuration is not yet complete.
  - A ready-to-receive status has to be established for any (unspecified) communication partners.

  The communications connection is provisionally ready for operation after the connection parameters have been loaded to the device.

Programmed communication connections

In certain application areas it is not beneficial to establish the communication connections statically via the the configuration. Alternatively, you can set up certain connection types in a program-controlled manner via a specific application, and therefore dynamically if necessary.

Points to note about configuring connections (S7-300, S7-400, S7-1500)

The network view provides you with a clearly structured platform for configuring communication connections between configured devices.

Essential functions

- Graphic connection configuration in the network view
  You can create a connection graphically with only a few mouse clicks.
- Highlight possible connection partners
  You select a communication type - the possible communication partners are automatically highlighted.
- Automatic search for connection paths
  You arrange for the automatic assignment of the best possible connection path or the selection of alternative connection paths.
- Automatic transfer of configurations
  If you configure or network modules later on, connections that have already been configured and are supported by the module are automatically assigned.
- User-friendly selection of connection display
  If you select an object in the network view, for example a subnet or the interface for a module capable of communication, the associated configured connection link is displayed in the connection table.
- Taking into account available resources
  If connection resources are required for a connection type, any limitation specified during the creation of the connection will be taken into account. The maximum number of connection resources varies for both the CPU device types and for the CP device types.

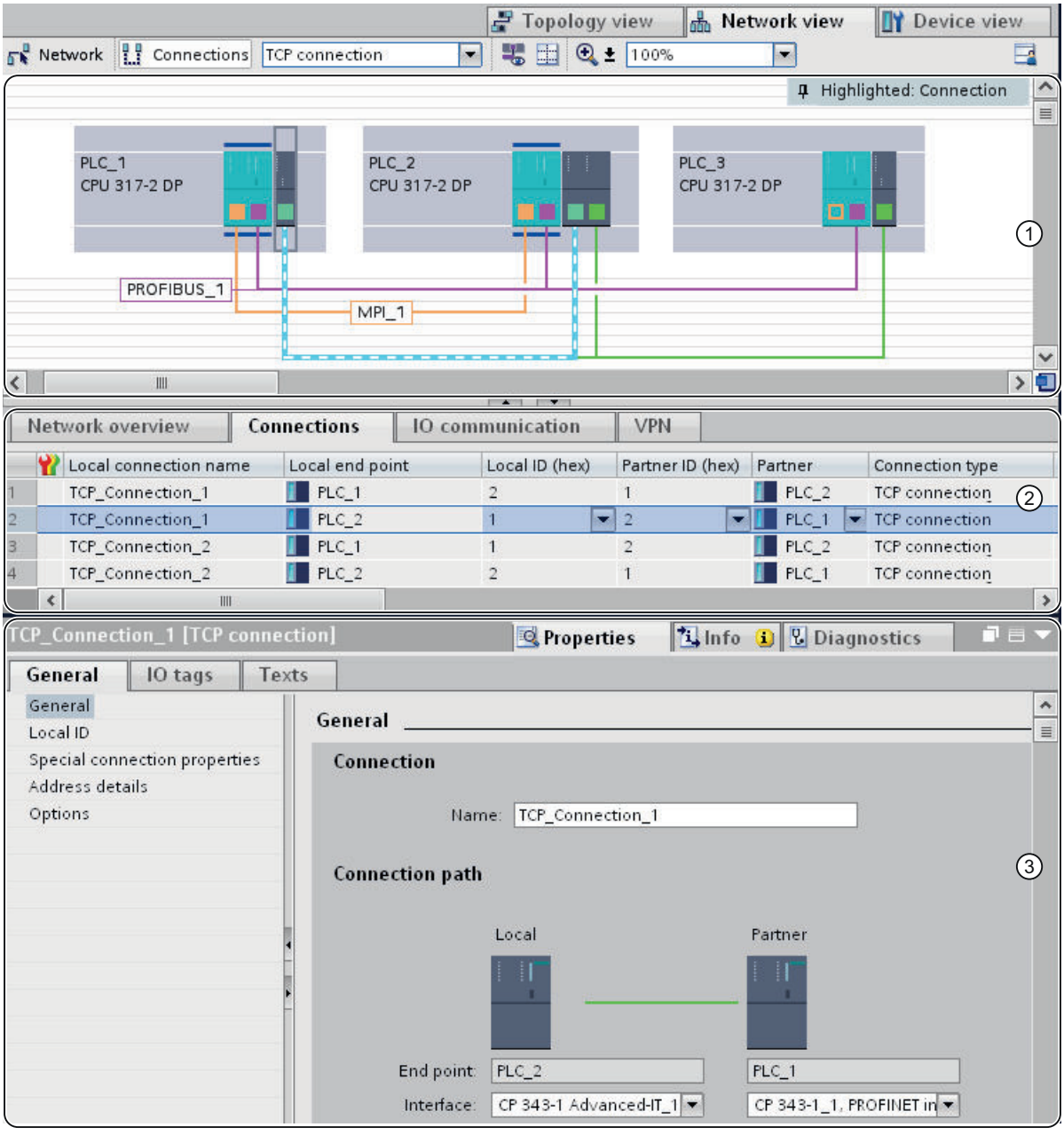

① Connection display in the network view

② Connection table

③ "Properties" tab for a connection in the inspector window

See also

Overview of connection diagnostics (Page 3360)

Range of applications of connection types (S7-300, S7-400, S7-1500)

Introduction

The following sections provide a brief overview of the connection types that you can use for communication to specific device types and areas of application.

Additional information

For more information on possible forms of communication in SIMATIC and the intended use of the connection types, refer the manual on communication with SIMATIC.

S7 connections

The features of S7 connections include:

- Connection type can be used in all S7 devices
- Can be used on the following subnet types: MPI, PROFIBUS, Industrial Ethernet
- When using BSEND/BRCV SFBs: secure transfer of data between SIMATIC S7-400/1500 stations; e.g., exchange of data block contents (up to 64 Kbytes)
- With CPU 317-2 PN/DP and with CPU 31x and a CP and when using BSEND/BRCV FBs: secure transfer of data to S7-300, S7-400, and S7-1500
- When using USEND/URCV SFBs: rapid, unsecured transfer of data regardless of the time at which the communication partner processes it; for example for operating and maintenance messages
- Acknowledgement of data transfer from communication partner at layer 7 of the ISO reference model

FDL connection

PROFIBUS-FDL (Fieldbus Data Link) is characterized by the following features:

- Transfer of data to a communication partner (for example SIMATIC S5 or PC), which supports sending and receiving according to the SDA function (Send Data with Acknowledge)
- Receipt of data is confirmed by the communication partner's FDL service using an acknowledgement
- Usable subnet type: PROFIBUS
- Corresponds to standard EN 50170 Vol.2 PROFIBUS
- Allows layer 2 of the ISO reference model to be assigned
- Interface in the S7 user program: SEND/RECEIVE;
- FDL services available as C functions on the PC

ISO transport connection

The ISO transport connection is characterized by the following features:

- "Data blocking" suited to large volumes of data
- Communication to a partner (for example SIMATIC S5 or PC) which supports sending and receiving data in accordance with ISO transport
- Usable subnet type: Industrial Ethernet
- Receipt of data is confirmed by the communication partner's ISO transport service using an acknowledgement
- Interface in the S7 user program: SEND/RECEIVE and FETCH/WRITE;
- ISO transport service (ISO 8073 class 4) corresponds to layer 4 of the ISO reference model
- ISO transport services available as C functions on the PC

ISO-on-TCP connection

The ISO-on-TCP transport connection is characterized by the following features:

- Corresponds to the standard TCP/IP (Transmission Control Protocol/Internet Protocol) with the RFC 1006 extension in accordance with layer 4 of the ISO reference model. RFC 1006 describes how the services from ISO layer 4 can be mapped on TCP
- Communication to a partner (for example PC or external system) which supports sending and receiving data in accordance with ISO-on-TCP
- Usable subnet type: Industrial Ethernet
- The receipt of data is confirmed using an acknowledgement
- Interface in the STEP 7 user program: SEND/RECEIVE and FETCH/WRITE;
- ISO-on-TCP transport services available as C functions on the PC

TCP connection

The TCP connection is characterized by the following features:

- Corresponds to the standard TCP/IP (Transmission Control Protocol/Internet Protocol):
- Communication with a partner (for example PC or third-party system) that supports sending and receipt of data in compliance with TCP/IP
- Usable subnet type: Industrial Ethernet
- Interface in the STEP 7 user program: SEND/RECEIVE and FETCH/WRITE;
- TCP/IP implementation of the PC operating system can usually be used

UDP connection

The UDP connection (User Datagram Protocol) is characterized by the following features:

- Usable subnet type: Industrial Ethernet (TCP/IP protocol)
- Unsecured transmission of related data fields between two nodes
- Interface in the S7 user program: SEND/RECEIVE.

E-mail connection

The e-mail connection is characterized by the following features:

- Usable subnet type: Industrial Ethernet (TCP/IP protocol)
- Enables for example the sending of for example process data from data blocks via e-mail using a CP with IT functionality (IT-CP)
- The mail server with which all e-mails sent by an IT-CP are served is defined by the e-mail connection.

FTP connection (specially set up TCP connection)

You use the FTP connection to process FTP order sequences.

When using the file transfer functions (FTP), CPs with IT functionality allow you to transfer files to and from the S7 devices.

HMI connection

AUTOHOTSPOT

Selecting connection types (S7-300, S7-400)

Selecting the connection type

Which type of connection you select depends on the required transmission properties, as well as on the subnet and transmission protocol by means of which the connection is established. In addition, the device type at each connection endpoint limits the connection types.

Which instructions or services you use at the interface (connection endpoints), depends on the connection type and the device type.

The following table shows these dependencies:

Connection type	Subnet type / protocol	Device type: Connection between SIMATIC ...	Interface
S7 connection	MPI, PROFIBUS, Industrial Ethernet	S7 - S7, S7 - PG/PC, S7 - PG/PC with WinCC, PG/PC - PG/PC S7 - partner in another project (S7, PG/PC with WinCC)	**Instructions for S7:** USEND, URCV, BSEND, BRCV, GET, PUT, START, STOP, RESUME, STATUS, USTATUS **PG/PC:** OPC services or programming interfaces
FDL connection	PROFIBUS (FDL protocol)	S7 - S7, S7 - S5, S7 - PC/PG, S7 - external device, PG/PC - PG/PC S7 - partner in another project (S7, S5, PG/PC, external device)	**Instructions for S7:** AG_SEND, AG_RECV, AG_LSEND, AG_LRECV **PG/PC:** OPC services
ISO transport connection	Industrial Ethernet (ISO transport protocol)	S7 - S7, S7 - S5, S7 - PC/PG, S7 - external device, PG/PC - PG/PC S7 - unspecified S7 - partner in another project (S7, S5, PG/PC, external device, unspecified)	**Instructions for S7:** AG_SEND, AG_RECV, AG_LSEND, AG_LRECV AG_LOCK AG_UNLOCK **PG/PC:**OPC services
ISO-on-TCP connection	Industrial Ethernet (TCP/IP protocol)	S7 - S7, S7 - S5, S7 - PC/PG, S7 - external devices, PG/PC - PG/PC, S7 - unspecified S7 - partner in another project (S7, S5, PG/PC, external device, unspecified)	**Instructions for S7:** AG_SEND, AG_RECV, AG_LSEND, AG_LRECV AG_LOCK AG_UNLOCK **PG/PC:**OPC services
TCP connection	Industrial Ethernet (TCP/IP protocol)	S7 - S7, S7 - S5, S7 - PC/PG, PG/PC - PG/PC S7 - external device, S7 - unspecified S7 - partner in another project (S7, S5, PG/PC, external device, unspecified)	**Instructions for S7:** AG_SEND*, AG_RECV*, AG_LSEND**, AG_LRECV** AG_LOCK AG_UNLOCK **PG/PC:**OPC services
UDP connection	Industrial Ethernet (TCP/IP protocol)	S7 - S7, S7 - S5, S7 - PC/PG, S7 - external device, S7 - unspecified S7 - partner in another project (S7, S5, PG/PC, external device, unspecified)	**Instructions for S7:** AG_SEND, AG_RECV, AG_LSEND, AG_LRECV

Connection type	Subnet type / protocol	Device type: Connection between SIMATIC ...	Interface
E-mail connection	Industrial Ethernet (TCP/IP protocol)	S7 - unspecified (S7 - mail server), S7 - PC/PG (S7 - mail server)	**Instructions for S7:** AG_SEND, AG_LSEND
FTP connection (specially set up TCP connection)	Industrial Ethernet (TCP/IP protocol)	S7 - unspecified	**Instructions for S7:** FTP_CONNECT, FTP_STORE, FTP_RETRIEVE, FTP_DELETE, FTP_QUIT

Connection to broadcast and multicast nodes

Note

For special connection types, there is the option of selecting not just one connection partner but several (broadcast and multicast nodes). You can select the connection partner "Broadcast to all..." or "... multicast nodes" in the dialog box for entering a new connection.

- You can set up a connection to "all broadcast nodes" (sending to all broadcast recipients at the same time) for connections of the FDL and UDP connection type.
- You can set up a connection to "all multicast nodes" (sending to several nodes at the same time) for connections of the FDL and UDP connection type.

Selection of connection types (S7-1500)

Selection of connection type

You must decide on a communication service according to your automation task. The selection of communication service affects the following:

- The functionality that will be available
- The time at which the connection will be established
- The activities you will have to perform (e.g., configuring of connections, programming of instructions)

The table below provides an overview of the available communication services.

Communication service	Functionality	Via interface:		
		PN	DP	Serial
PG communication	For commissioning, test, diagnostics	X	X	-
HMI communication	For operator control and monitoring	X	X	-
I/O communication	Data exchange between e.g.: • IO controllers and IO devices • DP master and DP slave	X	X	-

Communication service	Functionality	Via interface:		
		PN	DP	Serial
S7 communication	Data exchange between client and server or between clients: Configuration of connection required Using PUT/GET, BSEND/BRCV, or USEND/URCV instructions	X	X	-
Open communication via TCP/IP	Data exchange via PROFINET with TCP/IP protocol With connection configuration: TSEND_C/TRCV_C or TSEND/TRCV instructions By means of user program only: TCON, TSEND, T_RCV, T_DISCON, or TSEND_C and TRCV_C instructions	X	-	-
Open communication via ISO-on-TCP	Data exchange via PROFINET with ISO-on-TCP protocol With connection configuration: TSEND_C/TRCV_C or TSEND/TRCV instructions By means of user program only: TCON, TSEND, T_RCV, T_DISCON, or TSEND_C and TRCV_C instructions	X	-	-
Open communication via UDP	Data exchange via PROFINET with UDP protocol With connection configuration: TSEND_C/TRCV_C or TSEND/TRCV instructions By means of user program only: TCON, TSEND, T_RCV, T_DISCON, or TSEND_C and TRCV_C instructions	X	-	-
Open communication via ISO (CPs with PN interface only)	Data exchange via PROFINET with ISO protocol With connection configuration: TSEND_C/TRCV_C or TSEND/TRCV instructions By means of user program only: TCON, TSEND, T_RCV, T_DISCON, or TSEND_C and TRCV_C instructions	X	-	-
Open communication via e-mail	Data exchange via e-mail By means of user program only: TMAIL_C instruction	X	-	-
Open communication via FTP (CPs with PN interface only)	Data exchange via FTP (File Transfer Protocol) By means of user program only: FTP_CMD instruction	X	-	-
Open communication via FDL (CMs with DP interface only)	Data exchange via PROFIBUS with FDL protocol With connection configuration: TSEND_C/TRCV_C or TSEND/TRCV instructions By means of user program only: TCON, TSEND, T_RCV, T_DISCON, or TSEND_C and TRCV_C instructions	-	X	-
Point-to-point connection	Data exchange via point-to-point with Freeport, 3964 (R), USS, or Modbus protocol Using SEND_P2P/RCV_P2P or MB_COMM_LOAD, MB_MASTER, MB_SLAVE instructions	-	-	X
S7 routing	Data exchange across network boundaries, e.g., for parameter assignment, test, and diagnostic purposes	X	X	-

Communication service	Functionality	Via interface:		
		PN	DP	Serial
Data record routing	Extension of S7 routing, e.g., for parameter assignment and diagnostics of field devices on PROFIBUS by an engineering system with a PROFINET interface.	X	X	-
Web server	Data exchange via the Internet, e.g., for diagnostics	X	-	-
SNMP (Simple Network Management Protocol)	Data exchange via SNMP standard protocol, for network diagnostics and parameter assignment	X	-	-
Time-of-day synchronization	Via NTP (Network Time Protocol); CPU is client	X	-	-
	CPU is time master or times slave	-	X	-

Connection to broadcast and multicast nodes

Note

For special connection types, there is the option of selecting not just one connection partner but several (broadcast and multicast devices). You can select the connection partner "To all broadcast..." or "... multicast devices" in the dialog box for entering a new connection.

- You can set up a connection to "all broadcast devices" (sending to all broadcast recipients at the same time) for FDL and UDP connection types.
- You can set up a connection to "all multicast devices" (sending to several devices at the same time) for FDL and UDP connection types.

Instructions for communication tasks (S7-300, S7-400, S7-1500)

Interface to user program

Special instructions are available for your communication interface in the SIMATIC S7-300/400/1500 user program. Here is an overview of the instructions for the individual connection types.

The PUT/GET instructions are used for the S7-1500. See the following tables for the instructions of the S7-300/400.

Instructions for S7 connections

The system function blocks assigned to the instructions are integrated in the CPUs of the S7-400.

For S7-300 with more recent CPUs and CPs, you have the option of running the S7 communication actively (i.e. as client) via the CP's interface. The instructions have the same designation as the S7-400 SFBs, but must be called cyclically in the S7-300 CPU user program.

The CP must support the client function for S7 communication.

An S7-300 CPU with integrated PROFINET interface can also be configured as a client for S7 communication. The same blocks from the above scenario using S7-300 and CP are used. The client functionality is only available at the PROFINET interface.

Name	Brief description
USEND URCV	Uncoordinated exchange of data via a send and receive SFB Max. length SFB 8/9: 440 bytes split into 4x100 bytes Max. length FB 8/9: 160 bytes
BSEND BRCV	Exchange of data blocks of variable lengths between a send SFB and a receive SFB Max. length SFB 12/13: 64 KB Max. length FB 12/13: 32 KB
GET	Read data from a remote device Max. length SFB 14: 400 bytes split into 4x100 bytes Max. length FB 14: 160
PUT	Write data to a remote device Max. length SFB 15: 400 bytes split into 4x100 bytes Max. length FB 15: 160
START	Run a restart (warm start) in a remote device
STOP	Put a remote device in "STOP" mode
RESUME	Run a resume process in a remote device
STATUS	Specific querying of the status of a remote device
USTATUS	Receipt of status messages from remote device
CONTROL	Query the status of the connection which belongs to an SFB instance
C_CNTRL	Query the status of a connection (for S7-300 CPUs)

Instructions for S7 communication

Instructions for point-to-point connections

You can use the BSEND, BRCV, GET, PUT and STATUS SFBs for connections of the point-to-point connection type (see above table).

The PRINT SFB can also be used:

Name	Brief description
PRINT	Send data to a printer

Instructions for FDL, ISO-on-TCP, UDP and TCP connections, as well as e-mail connection

Name	Brief description
AG_SEND	Sends data via a configured connection to the communication partner (with older CPs and with CPs for S7-400, the following limitation applies to this FC: <= 240 bytes)
AG_RECV	Receives data via a configured connection from the communication partner (with older CPs and with CPs for S7-400, the following limitation applies to this FC:<= 240 bytes, not e-mail)
AG_LSEND	Sends data via a configured connection to the communication partner
AG_LRECV	Receives data via a configured connection to the communication partner (not e-mail)
AG_LOCK	Lock external data access using FETCH/WRITE (not with UPD, e-mail)

Name	Brief description
AG_UNLOCK	Unlock external data access using FETCH/WRITE (not with UPD, e-mail)
AG_CNTRL	Diagnose connections via the user program

AUTOHOTSPOT

AUTOHOTSPOT

Instructions for FTP services via TCP connections (FTP connection)

Name	Brief description
FTP_CONNECT	Establish FTP connection
FTP_STORE	Transfer data block (file DB) from FTP client to FTP server
FTP_RETRIEVE	Transfer file from FTP server to FTP client
FTP_DELETE	Delete file from FTP server
FTP_QUIT	Quit the FTP connection stated by the ID

AUTOHOTSPOT

Instructions for programmed open communication services with ISO-on-TCP, UDP and TCP connections, as well as for e-mail connection via CPs

Name	Brief description
IP_CONFIG	Set up program controlled communication connections and IP configuration of the CP.

AUTOHOTSPOT

Instructions for programmed open communication services (Open User Communication) with ISO-on-TCP, UDP and TCP connections via CPU or CP*)

*) Note: With these instructions, communication via CP is only possible using ISO-on-TCP.

Name	Brief description
TCON	Connection establishment
TDISCON	Disconnection
TSEND	Sending data
TRCV	Receiving data
TUSEND	Sending of data; connectionless protocol UDP in accordance with RFC 768
TURCV	Receiving of data; connectionless protocol UDP in accordance with RFC 768

AUTOHOTSPOT

Communication connections via CPs (S7-300, S7-400, S7-1500)

Data volumes and configuration limit

The number of communication connections which are supported by the respective CP can be found in the equipment manual accompanying every CP. The number of connections per device can be increased by adding other CPs.

If several CPs are fitted in a device, an automatic switch is made to the next CP when this limit is exceeded. Specifically assign the connections using the path selection as required.

Ethernet CP

The Ethernet CP can transfer the following data volumes via one connection per order:

	ISO transport	ISO-on-TCP	TCP	UDP
Sending	8192 bytes	8192 bytes	8192 bytes	2048 bytes
Receiving	8192 bytes	8192 bytes	8192 bytes	2048 bytes

Note

Data transfer > 240 byte transfer is supported by the current CPs.

CPs with an older product version support the transfer of data with a data length of up to 240 bytes.

For more information, refer to the details provided in the Ethernet CP equipment manual.

Tasks of the Ethernet CP during online mode

During the transfer of data via a connection, the Ethernet CP takes on the following tasks:

- Receiving
  Receiving data from the Ethernet and forwarding to the user data area in the CPU
- Sending
  Transferring data from the user data area of the CPU and sending data via the Ethernet

The connection is established automatically as soon as the partner can be reached.

With a free UDP/FDL connection, the following also applies:

- When receiving
  Entering the sender who sent the message in the order header
- When sending
  Evaluating the order header and addressing the partner

PROFIBUS-CP

The PROFIBUS CP can transfer the following data volumes via one connection per order:

FDL connection type	Specified	Unspecified - free layer 2 access	Broadcast	Multicast
Sending	240 bytes	236 bytes [1]	236 bytes [1]	236 bytes [1]
Receiving	240 bytes	236 bytes [1]	236 bytes [1]	236 bytes [1]

1) The order header also assigns 4 bytes alongside the user data stated

Tasks of the PROFIBUS CP during online mode

When processing the transfer of data via an FDL connection, the PROFIBUS-CP takes on the following tasks:

- With specified connections
  - Receiving
    Receiving data from the PROFIBUS and forwarding to the user data area in the CPU
  - Sending
    Transferring data from the user data area of the CPU and sending data via the PROFIBUS
- With unspecified connections also
  - When receiving
    Entering the sender and FDL service in the order header
  - When sending
    Evaluating the order header and addressing the partner; performing the selected FDL service

Communication connections to PC stations (S7-300, S7-400, S7-1500)

Connection endpoints of a PC station

On PC stations, you configure connections with one of the following components as connection endpoint:

- OPC server as central communication service provider
  Depending on the available CP, you can use the following connection types:
  - S7 connection
  - ISO connection
  - ISO-on-TCP connection
  - TCP connection
  - FDL connection
- Standard application
  With the S7 connection type, communication is possible via Ind. Ethernet or PROFIBUS, depending on the available CP.

Note

CPU 1515SP PC as router for S7 connections

If you assign the "PROFINET onboard [X2]" interface to the CPU 1515SP PC of the SIMATIC PC station, the CPU 1515SP PC can be used as a router for S7 connections. With the setting "None, or a different Windows setting" on the CP interface "PROFINET onboard [X2]" , it is not possible to use the CPU 1515SP PC as a router for routed S7 connections.
An existing S7 connection routed by the CPU 1515SP PC becomes invalid if the assignment of the interface of the CPU 1515SP PC is changed from "SIMATIC PC station" to "None, or a different Windows setting". Since the PLC now no longer handles routing functions for this connection, when the CPU 1515SP PC is compiled, no message relating to the invalid connection is displayed. The invalid routed S7 connection is displayed only when the end points of the connection are compiled.
The interfaces required for routed S7 connections must remain explicitly assigned on the CPU 1515SP PC . You can edit the assignment of the interface of the CPU 1515SP PC in the properties under "PROFINET onboard [X2] > Interface assignment".

Special features of the connection configuration with OPC server

In the case of connections with OPC as connection endpoint, more connections can generally be configured than can be operated at any time in the CP used in the PC stations.

That is because by the connection properties differentiate between "Maintain permanently" and "Set up on demand".

Accordingly, the number of connections with the property "Maintain permanently" cannot exceed the maximum number of CP connection resources. If it proves possible to establish additional connections on demand, you have to keep free the CP's corresponding connection resources. Therefore, configure a smaller number connections with the property "Maintain permanently" than the maximum number of CP connection resources permits.

A warning may possibly be indicated during the compiling of the connection configuration.

Additional information

Industrial communication with the PG/PC Band 1 under the entry ID: 42783968 (http://support.automation.siemens.com/WW/view/en/42783968)

Industrial communication with the PG/PC Band 2 under the entry ID: 42783660 (http://support.automation.siemens.com/WW/view/en/42783660)

Commissioning PC stations - Manual under the entry ID: 13542666 (http://support.automation.siemens.com/WW/view/en/13542666)

Unspecified connections (S7-300, S7-400, S7-1500)

Unspecified connections with Industrial Ethernet

Connections to an as yet unknown device (for example a diagnostics device) are configured as "unspecified" connections. The unspecified connection can be used in different ways as explained below based on the example of an ISO-on-TCP connection; on an ISO transport or TCP connection, the procedure is analogous.

- **Declaring ready for communication - passive connection establishment**
  Connection establishment must be set to "passive".
  The following then applies to the address setting for an ISO-on-TCP connection:
  the remote IP address and the remote TSAP are empty, in other words, they are not relevant for the CP. During connection establishment, every partner is then accepted (partner = connection name that addresses the CP with the correct IP address and TSAP).
  There is also the option of partial specification, in other words, the communication is permitted with any partner that matches the specified TSAP.
- **Connection to a specific station in any project.**
  The following then applies to the address setting for an ISO-on-TCP connection:
  You can specify the remote IP address and the port for any target station. The target station can be inside or outside of the current STEP 7 project.
  Use this variant if you have not created the partner station in the current project.
- **Connection without specifying a port**
  TCP connections are unspecified in the following situations:
  - The local port is not specified (active connection establishment).
  - The remote port is not specified (no active connection establishment).
- **IP addressing using DHCP**
  If you select the option of IP addressing using DHCP, it is not initially possible to create a fully specified connection in the project because the local IP address is unknown. You will therefore have to select "Unspecified" without active connection establishment as the connection type.

The following table summarizes the possibilities for setting the "remote" address parameters:

Significance of the connection establishment	IP address / MAC address (remote)	TSAP / port (remote)	Active connection establishment
By either partner	Empty	Empty	no
By either partner via a specific TSAP	Empty	Specified	no
To or from a specific partner	Specified	Specified	yes The local port can remain unspecified (but does not need to).
		Unspecified	no

The free UDP connection represents a further variant. With this type of connection, the address of the connection partner remains open in the configuration. The communication stations are determined by address information in the communication job of the user program.

Unspecified connections on PROFIBUS

An unspecified FDL connection with open layer 2 access enables program-controlled addressing of the communication partner and communication between two stations on PROFIBUS with the following properties:

The local station is specified by configuration. The remote node is entered by the user program in the job header of the job buffer with the AG_SEND instruction call. This means that each node on PROFIBUS (PROFIBUS addresses from 0 to 126) can be reached.

Additional information

You will find further information below in the sections describing specific connections.

Free UDP connection (Page 261)

FDL unspecified FDL connection (free layer 2 access) (Page 270)

Programmed communication connections via CP (S7-300, S7-400, S7-1500)

Range of application

In certain applications it is advantageous not to set up the communication connections or the IP configuration of the CP via the configuration interface, but rather to set up these connections in a program-controlled manner using a specific application.

You have the option of freely configuring the communication connections in a program-controlled manner via industrial Ethernet CPs. Keep in mind the system performance of the CP type you are using.

Typical application cases are, for example, with manufacturers of standard machines. To be able to offer most comfortable interfaces to their clients, these manufacturers adapt the communication services to the respective operator inputs. The end users require no knowledge about how to configure connections.

The IP_CONFIG instruction is available for these applications, which permits the flexible transfer of data blocks with configuration data to an Ethernet CP.

Interaction of programming and configuration

Connections are either configured or configured by the user program as the S7 device is running. A mixed form of these variants is not possible within one CP!

Principle

Configuration data for communication connections are transferred to the CP via the IP_CONFIG instruction in the user program.

The configuration DB can be loaded in this manner in the CP at any time. The previous connections and configuration data (IP address, subnet mask, default router, NTP time server and other parameters) are then overwritten.

The Ethernet CP uses the configuration data to detect whether the communication connections are to be set up using the user program.

Scale

A maximum of 64 connections can be specified in the IP_CONFIG instruction. However, the maximum number of connections supported by the CP type you are using is decisive.

Special features/constraints

- Consistency check only with configured connection
  The connection configuration is linked to consistency checks which are not possible or are only partly possible with the programmed configuration.
- Connection configuration by partner needed
  When configuring specified connections, the connection for the partner is implicitly created during configuration. This is not possible with a programmed configuration. In this case, appropriate consistent connections must be configured for the partners.
- Configuration of IP access protection
  IP access protection can be used to limit communication via the CP of the local S7 device to partners with very specific IP addresses. This parameter assignment also applies to programmed communication connections.
- DHCP/DNS is supported
  IP addressing is also possible via DHCP/DNS for the programmed configuration.

Connections with Open User Communication via CPU and CP (S7-300, S7-400, S7-1500)

Open User Communication via Industrial Ethernet - programmed communication connections

Open User Communication provides Industrial Ethernet services that you use via the instruction interface in the user program. Communication connections are set up via the instruction interface in the user program; it is not necessary to configure the connections.

Services via the CPU's interface

The following Open User Communication services are available with S7-1500 for data exchange via the integrated Ethernet interface of a CP:

- TCP as per RFC 793
- ISO-on-TCP as per RFC 1006
- UDP as per RFC 768 (broadcast or unicast, no support for multicast)

Services via the CPU's interface

The following Open User Communication services are available with S7-300/400/1500 for data exchange via the Ethernet interface of a CP:

- TCP as per RFC 793
- ISO as per RFC 905
- ISO-on-TCP as per RFC 1006
- UDP as per RFC 768 (broadcast or unicast, no support for multicast)

1.3.2.3 Configuring connections (S7-300, S7-400, S7-1500)

Creating a new connection - overview (S7-300, S7-400, S7-1500)

Creating a connection - alternatives

You have the following options for creating a connection in the network view:

- Graphic connection configuration
  You drag a connection line to the connection partner or assign a new connection to the previously selected connection partners. This is a simple method for creating individual connections in the network view.
  Missing networking is completed automatically or interactively.
- Interactive connection configuration
  Select the connecting partner you need from the list. This method is better than the graphic method if a large number of connections are to be configured.

You'll find the individual steps for this in the following chapters.

Assigning connection path

Depending on the application, a distinction should be made between the following procedures:

- Selecting the connection endpoint in the source and target device
  The software automatically identifies the best connection path to the target device. This connection can then be called the "standard connection path". The software considers the available connection resources when selecting the path.
  The endpoint can be a CPU or a PC application.
  Note: This variant is only possible with graphic connection configuration.
- Selecting connection paths in the source and target devices
  If intended for the purpose of splitting load between several components (CPs) and possibly also different subnets, you can also specifically configure the path of a connection. You specify a connection path via the connection endpoint and the interface to the subnet.

Requirement and result

You have created the devices with CPUs and PC applications and, if applicable, modules with communication capability (CPs) in the network view.

The result of configuration depends on the current configuration of the devices and networking in the project. You will therefore achieve the following results with the methods described in the following sections:

- Connection specified
  If both partners selected are networked in the same subnet, create a fully-specified connection using graphic or interactive selection.
  This fully-specified connection is automatically entered in the connection table of both the local device and that of the partner. The local ID and the partner ID are assigned for this connection. You will need these IDs when programming the communication instructions (value for the block parameter "ID") and the PC applications.
  Exceptions:
  - Unilateral S7 connections
    The connection partner acts as a server on an S7 connection configured at both ends. There is therefore no entry in the connection table for the partner. A connection ID is only issued for the local partner.

  The figure below shows a specified connection:

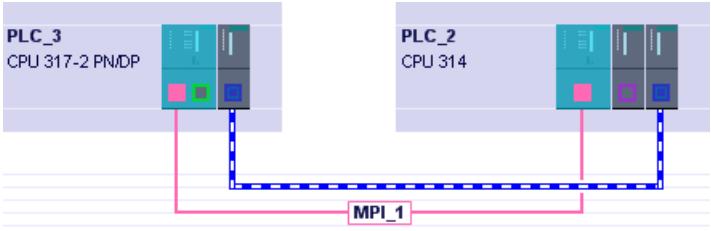

- Connection unspecified
  Create unspecified connections in the following cases by only selecting the local connection partner graphically or interactively:
  - If certain connection properties need an unspecified connection. Examples: E-mail connection, FTP connection, FDL connection with free layer 2 access.
  - If one of the two partners for the selected connection type is not (yet) available, but you want to reserve a connection resource.

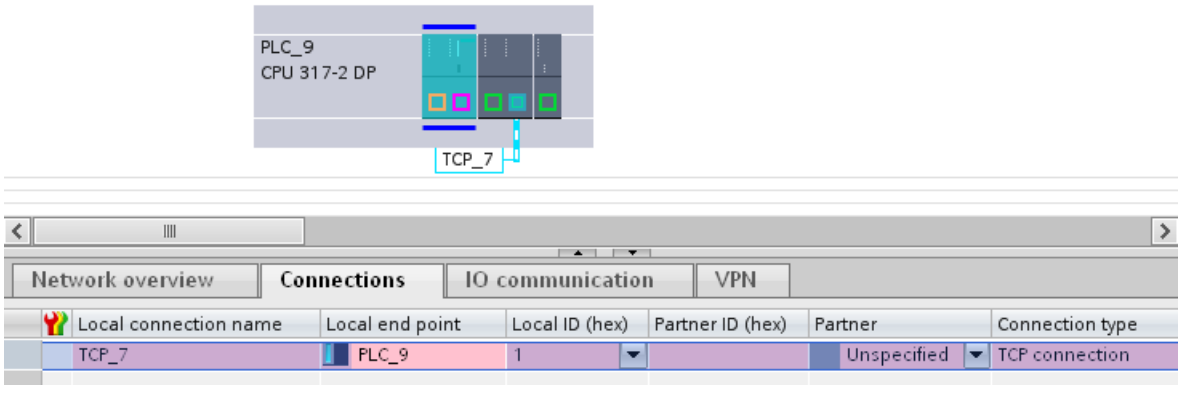

Influence of the networking - response when networking is missing

Networking the connection partners it is necessary to ensure the uniqueness and consistency of the assigned connection parameters. If one or both partners is not networked on the same subnet, the following reactions in the graphic connection configuration must be distinguished.

Note

Dialog-guided connection configuration - no automatic networking

In dialog-guided connection configuration, there is no automatic networking.

- Interfaces of the partners you want to connect are not networked, or do not have the same subnet
  The "Connect to subnet" dialog that appears provides the following options for selection:
  - Assign existing subnet
    You select a subnet from the list of existing subnets. The connection can be created consistently and can be specified.
  - "Connect to new subnet"
    The interfaces of the connection partners are connected via a new subnet. The connection can be created consistently and can be specified.
  - "Do not connect to a subnet"
    The connection is created as a specified connection, due to the missing networking the consistency cannot be assured which is why the entry appears red in the connection table. Connections are shown as dashed lines in the network view;
- Multiple interfaces / CPs on the partner - at least one of the available interfaces on the partner is not networked
  The following applies depending on how you create the connection in this case:
  - If you specify the connection only by selecting the CPU / OPC server as the connection endpoint, the connection path is specified regardless of the networking. This can lead to the response described in Case b and the selection dialog is displayed.
  - If you deliberately select the free interface on the partner that is not yet networked as the connection endpoint, the response is as described in Case a.
    The interfaces of the connection partners are connected via a new subnet. The connection can be created consistently and can be specified.

Views containing information about the configured connections (S7-300, S7-400, S7-1500)

The views described below will provide you with comprehensive access to all the information and functions regarding configuring and checking communication connections.

The view is accessed via the project navigation under "Devices & Networks".

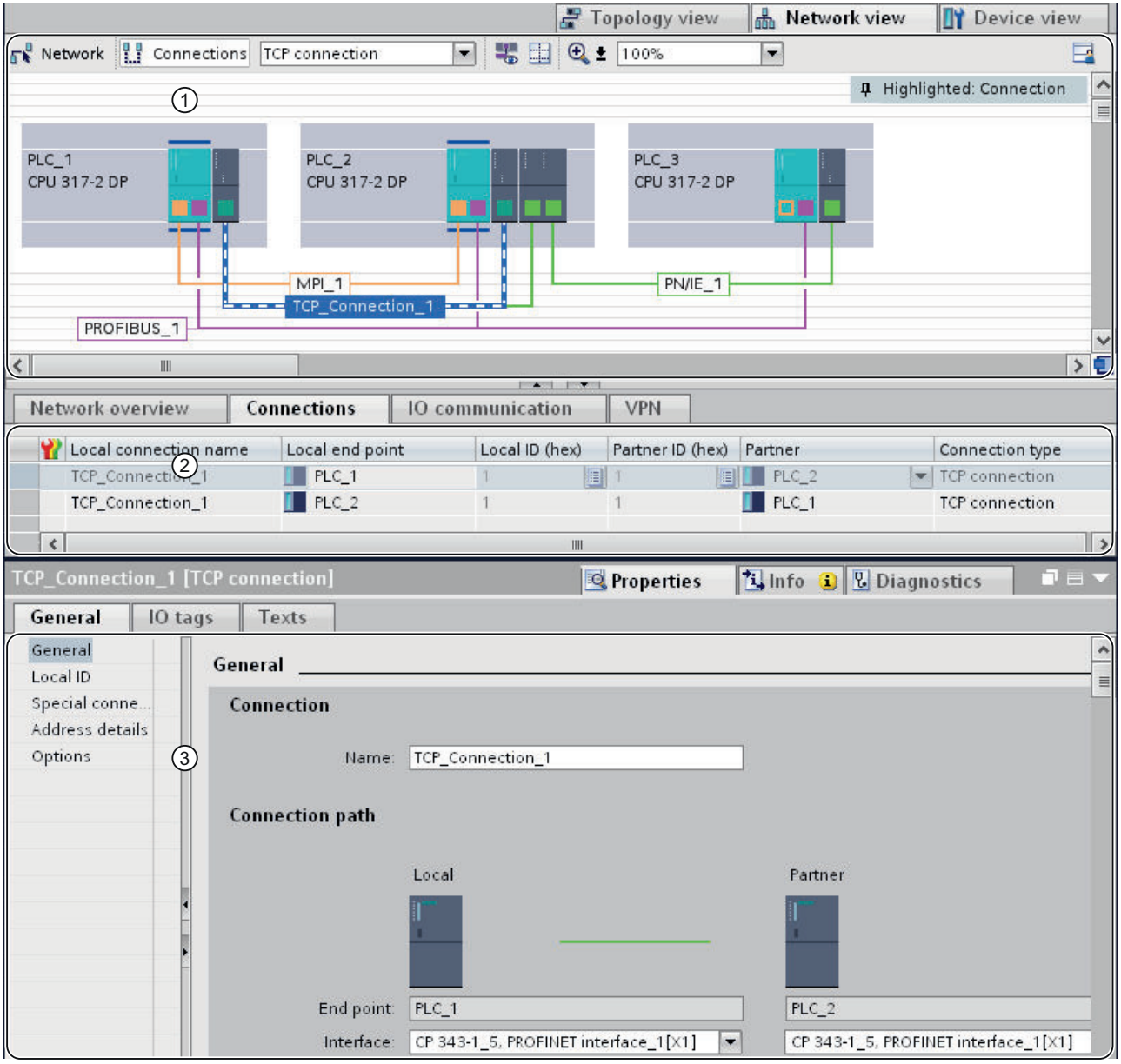

① Connection display in the network view

② Connection table

③ "Properties" tab for a connection in the inspector window

Benefits

The information shown in these views are always up-to-date in terms of the current user actions. This means:

- If you have selected a CPU, for example in the network view, the connection table shows all the connections created for this CPU.
  Note: You can use the connection table shortcut menu to choose between displaying all connections in the project and displaying the connections for a selected component.
- If you have selected a connection in the connection table:
  - The connection path in the network view will be shown in a graphic display when connection mode is activated.
  - The properties view in the inspector window displays the parameters of this connection.

Connection display in the network view

The network view shows devices and their components as well as networks and communications connections.

In relation to connections, in the network view you see:

- The devices between which connections have been created.
- The devices that have unspecified connections.

The network view offers the following functions:

- Creating connections graphically
- Selecting connection paths when creating graphically

The connection table

The connection table offers the following functions:

- Listing of the connections to an object selected in the network view (network, CPU, CP)
- Listing of all connections in the project
- Selecting a connection and the associated display of the connection in the network view and the display of the properties of a selected connection in the Inspector window;
- Changing connection partners
- Option of entering central parameters
- Display of status information;
- Starting point for interactive creation of new connections (possible if the "Display all connections" option is disabled).

"Properties" tab for a connection in the inspector window

The properties dialog has the following meaning:

- Display and option of setting all connection parameters
- Setting the connection path automatically or manually
- Later specification of unspecified connections

Creating a new connection graphically (S7-300, S7-400, S7-1500)

Graphically configuring connections

- In the simplest case, you only need to select the devices you want to connect. A possible connection path in the current configuration is selected and marked as "Standard connection path". The connection path is then selected such that the resources available are used as best as possible. The definition is based on the following behavior:
  - Non-networked components are networked when possible automatically to be able to create fully specified connections immediately.
  - The resources of one component are fully used up before switching to another component.
  - With connection types independent of networks (S7 connections), network types are subject to the following rule of preference: Industrial Ethernet before PROFIBUS.
- Alternatively, you can define the specific connection path - connection endpoint and interface to the subnet.

Note

Creating S7 connections on the CPU 1515SP PC graphically

If you want to create an S7 connection on a CPU 1515SP PC , use the right interface of the CPU 1515SP PC "PROFINET onboard [X1]" when creating the connection graphically. This interface is connected directly to the CPU 150xS .

The left interface of the CPU 1515SP PC "PROFINET onboard [X2]" is connected to the SIMATIC PC station or a Windows application and cannot be used during graphic creation of an S7 connection.

Automatically determining connection path

To graphically create a connection via a default connection path, proceed as follows:

1. On the toolbar, click the "Connections" icon. This activates connection mode.

2. Select the required connection type in the drop-down list next to the icon.
   You will then see the following response:
   – The components which can be used in your project as connection endpoints in the selected connection type are highlighted in color in the network view. These components may be CPUs or PC applications.

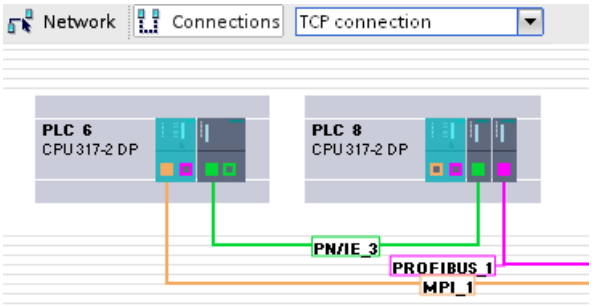

3. Select the component for the origin of the connection.
4. There are two options when creating the connection:
   – Click on the component from which the connection will originate and drag the mouse cursor to the target components in connection mode. Confirm the connection endpoint with another mouse click.

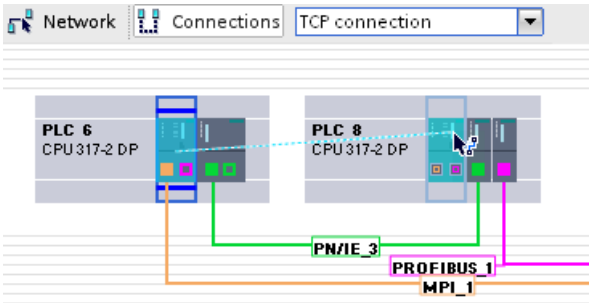

   – While holding down the shift button, select the target component and select the "Add new connection" command with the right mouse button.

Note

If you wish to exit connection mode before, press <Esc>, right-click the mouse or double-click into the background of the network view.

Result

- A specified connection is created.
- The connection path is highlighted. The connection path has been set automatically in line with the available connection resources and interfaces.
- The connection is entered in the connection tables for the components (CPU/PC application).

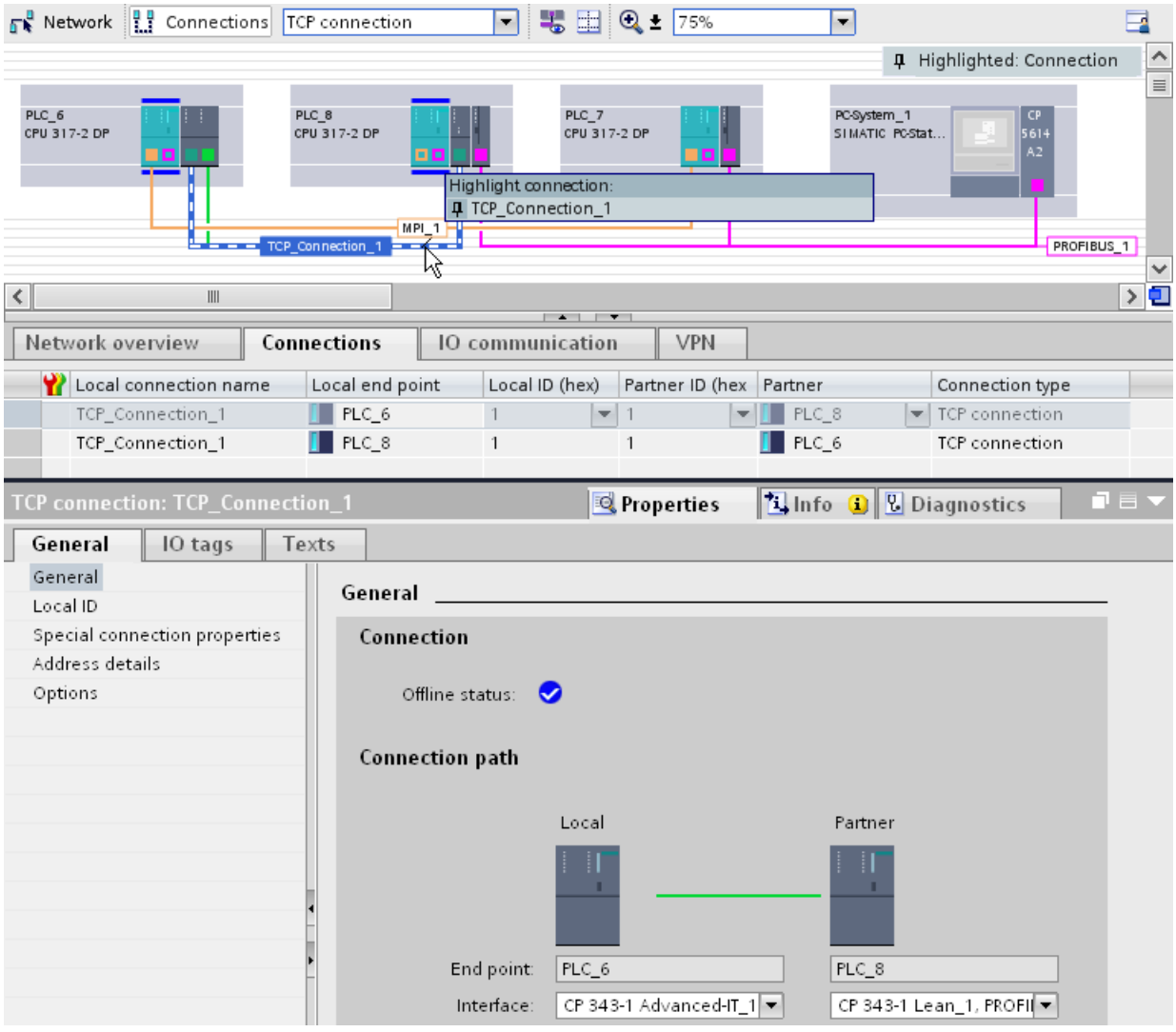

Selecting specific connection path

Proceed as follows to create a connection with an individually selected connection path:

1. On the toolbar, click the "Connections" icon. This activates connection mode.

2. Select the connection type you need from the drop-down list.

3. Click on the subnet interface in the device from which the connection is to be created.
4. In connection mode, hold down the mouse button, drag the cursor to the relevant interface in the target device and then release the mouse button.

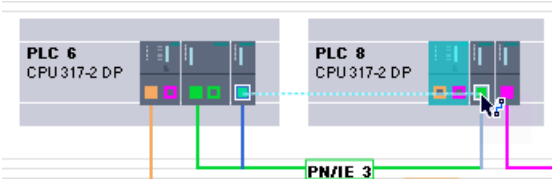

Note

If you wish to exit connection mode before, press <Esc>, right-click the mouse or double-click into the background of the network view.

Result:

– A specified connection is created.
– The connection created between the endpoints will use the selected interfaces for its connection path.
– The connection path is highlighted.
– The connection is entered in the connection tables for the devices (CPU/PC application).

Configuring a connection when there is no or no clear network assignment

Any networking not in place will if possible be created automatically when a connection is created. A query will be made upon completion of connection configuration if unique network assignment is not possible. You will be able to choose from the existing subnets.

Example below: A query is made upon creation of a connection between stations PLC2 and PLC3, which are not yet networked.

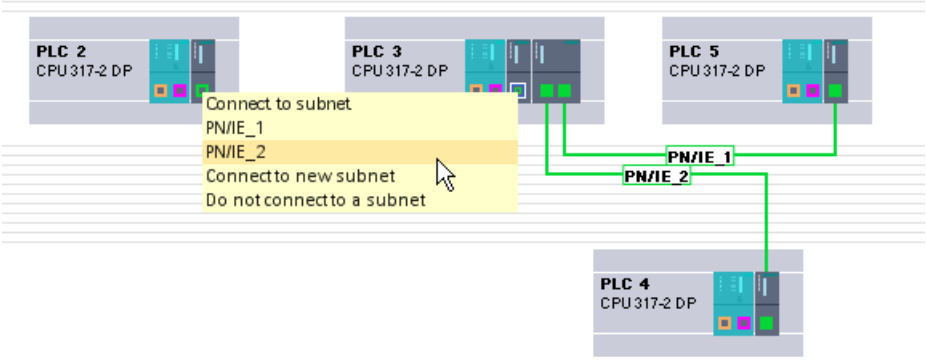

Interactively creating a new connection (S7-300, S7-400, S7-1500)

Interactively configuring connections

Interactive connection configuration is a useful way to create multiple connections in one process within a station.

On the one hand, you have complete control over the selection of the connection path as the system displays all interface variants. At the same time, you have the option of leaving evaluation of available resources and thus selection of the connection path to the system.

Note

Dialog-guided connection configuration - no automatic networking

In dialog-guided connection configuration, there is no automatic networking.

The "Create new connection" dialog

The structure of the "Create new connection" dialog is shown below. The example shows networking between several stations. Connection mode is active and the first station in the list has been selected.

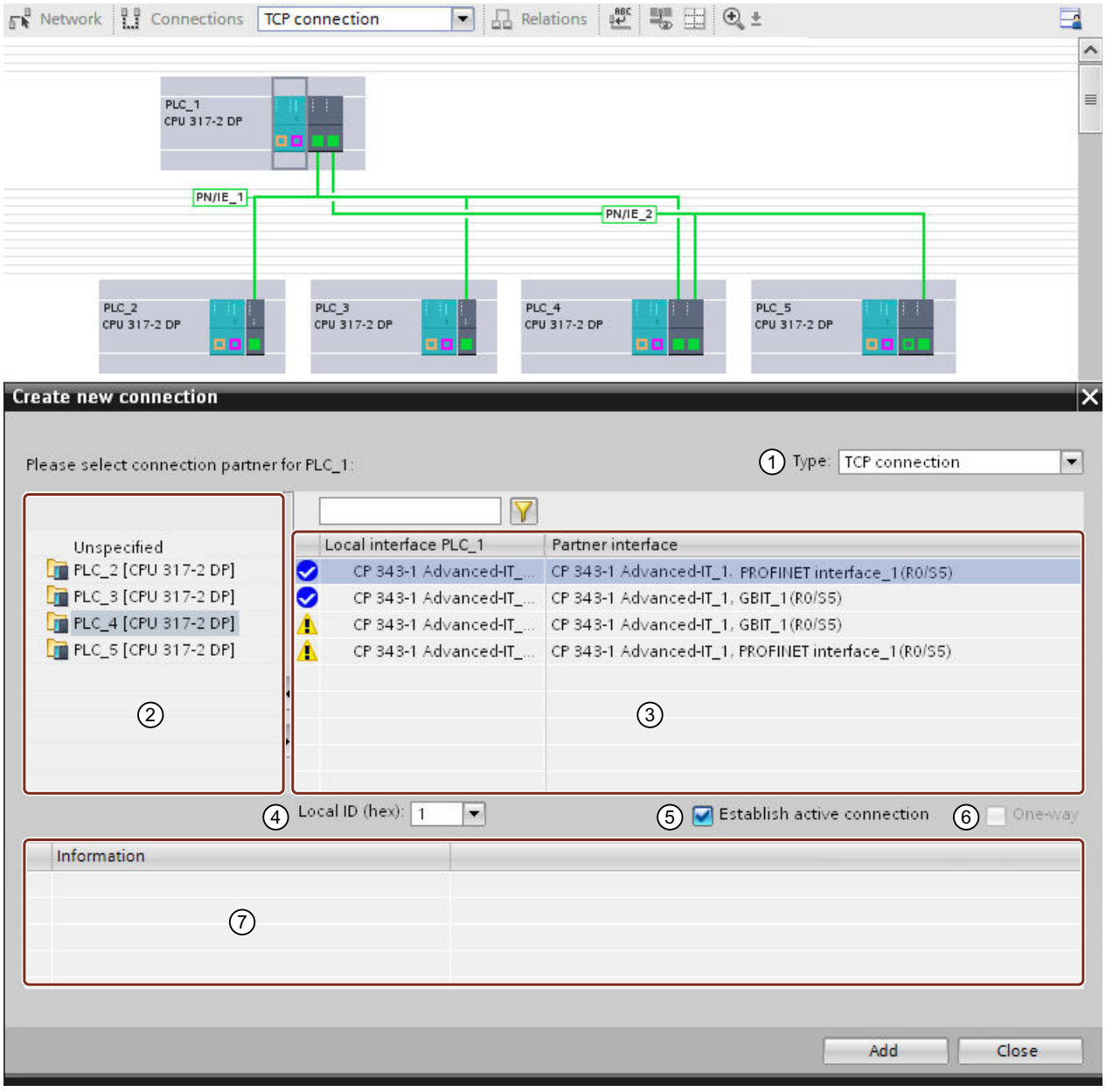

① Drop-down list for selecting the connection type.

② Shows the possible connection partners for the connection type selected.

Choose from the following options:

- A special connection configuration in line with the connection type:
  - "unspecified"
  - "Broadcast"
  - "Multicast"
- The connection partner

③ Content area for selecting interfaces.

You will see interface combinations suitable for the configuration. Shows only those interfaces possible for the connection type selected. The status display provides additional information on whether the connection can be set up as a function of networking and the available resources. You can still create a connection even if the interfaces are not networked.

See also AUTOHOTSPOT

Example shown:

Between PLC1 and PLC4, the Gbit ports are networked, as are the PROFINET interfaces. Only these two combinations are therefore shown as completely assignable.

④ Drop-down list for selecting or entering the local ID.

⑤ Check box for display of the active connection establishment.

When you create a connection, the check box for active connection establishment is always enabled for the local side. Not every device is capable of handling the connection establishment. In this case, the partner must take over connection establishment. The check box always indicates the capability of the selected connection.

⑥ Check box for display of a connection configured at one end.

A connection configured at one end refers to a client-server relationship. The client knows the server whereas the server has no information whatsoever about the client. On a connection configured at both ends, information on the partner is available on both end points. The check box always indicates the information about the selected connection.

⑦ Shows the confirmation for setting up a new connection.

The type of connection and the two connection partners are indicated. Yellow icons indicate that the connection has been established but is not consistent. In this case, check the connection parameters.

Status display

Icon	Meaning
	Interfaces are networked and connection resources are available for the selected partners.
	Interfaces are disconnected or not connected to the same subnet.

Procedure

Proceed as follows to interactively create a connection:

1. On the toolbar, click the "Connections" icon. This activates connection mode.

2. Select the command "Add connection" in the shortcut menu of the component which is to be the connection endpoint (CPU or PC application).
   or
   With the "Display all connections" option disabled and the CPU or application selected, select the "Add connection" command in the shortcut menu of the connection table.
   The "Create new connection" dialog opens.
3. Select the connection type you want, for example "FDL connection" from the "Type" drop-down list.
4. Check the default setting for the other connection parameters displayed in the footer and correct them in line with your requirements.

   Note

   You can also change these parameters later on in the connection table or in the properties for the connection.

5. Select the connection partner (device or CPU) in the left of the dialog.
   The interface combinations possible for the configuration together with status information will be displayed in the content area. The first interface combination to which a connection of the selected type can be assigned will be selected.

   Note

   Depending on the connection type selected, special connection types such as "unspecified", "broadcast" or "multicast" may be available in addition to the possible connection partners.

6. Leave the selection as it is or select the connection path you require.
7. Confirm the selection with the "Add" button.
   A connection is created and can be seen in the connection table.
8. If you have not yet closed the dialog, you have the alternative of continuing as follows:
   - You can create another connection of the same type to the same target device by clicking the "Add" button again.
   - Repeat the steps described above by selecting another connection type from the drop-down list or another connection partner.

Configuring connection without networking

Any networking not in place will if possible be created automatically when a connection is created. The relevant interface will be displayed as networked in the network view once the dialog is completed provided unique network assignment is possible.

Networking will not be displayed in the case of interactive connection configuration if unique network assignment is not possible. Connections created in this way are therefore not fully specified and are highlighted in red in the connection table.

Working in the network view (S7-300, S7-400, S7-1500)

Meaning of the network view in connection mode

The network view shows devices and their components with interfaces to the subnets. In connection mode, you can make the communication paths of configured connections visible.

The following information about connections is then available:

- The devices between which connections have been created
- Which connection types have been created for a device
- The devices that have unspecified connections

The network view offers the following functions in connetion mode:

- Graphically creating connections; see also Creating a new connection graphically (Page 211)
- Highlighting selected connections of a subnet;
- Highlighting connection partners.

Highlighting connection path and partner in the network view

To display the connection partners for all or certain connection types in the network view, proceed as follows:

1. Select the "Highlight connection partners" command in the shortcut menu of the connection endpoint (CPU or PC application) whose connection partners you want to display in the network view.
2. In the following menu, select either "All connection partners" or "Select connection type..." The local endpoint and the endpoints of the target devices relevant for all or the selected connection type are selected. In addition, the devices display an arrow.
3. You can open a list with information on the target devices by right-clicking on the arrow in the footer of the local device. This additional function is useful in complex network configurations in which some devices are not visible.

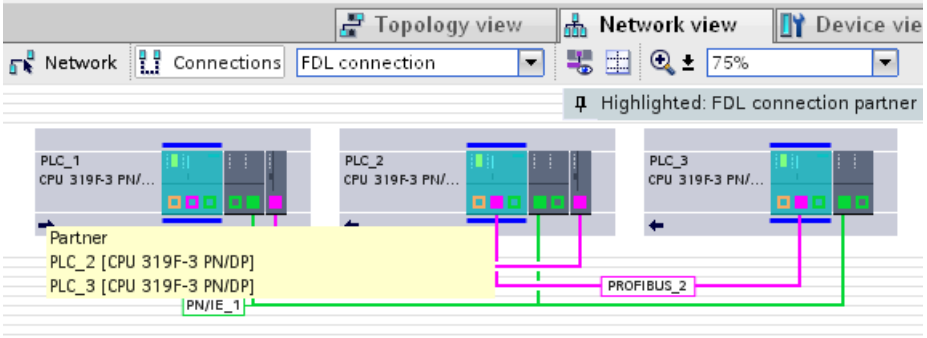

Note

You can display one of the connection partners which cannot be seen in the current display range of the network view. Click on the communication partner in the list that appears. Result: The display is moved such that the connection partner becomes visible.

Highlighting/selecting the connection

You can select and highlight individual connections. This means:

- The connection is highlighted in the connection table where you can set certain properties.
- The connection is displayed in the inspector window together with all parameter assignment options.

Follow these steps:

1. Place the cursor on the subnet.
2. Select and highlight the relevant connection in the "Highlight connection" dialog that appears.
   The connection path will be displayed as a light blue dotted line.

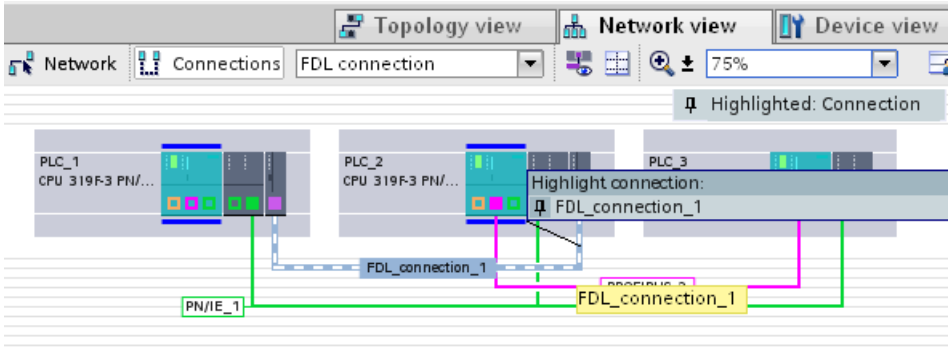

3. Click on the dotted line.
   The color of the line will change to dark blue and the connection will be displayed in the connection table and in the inspector window.

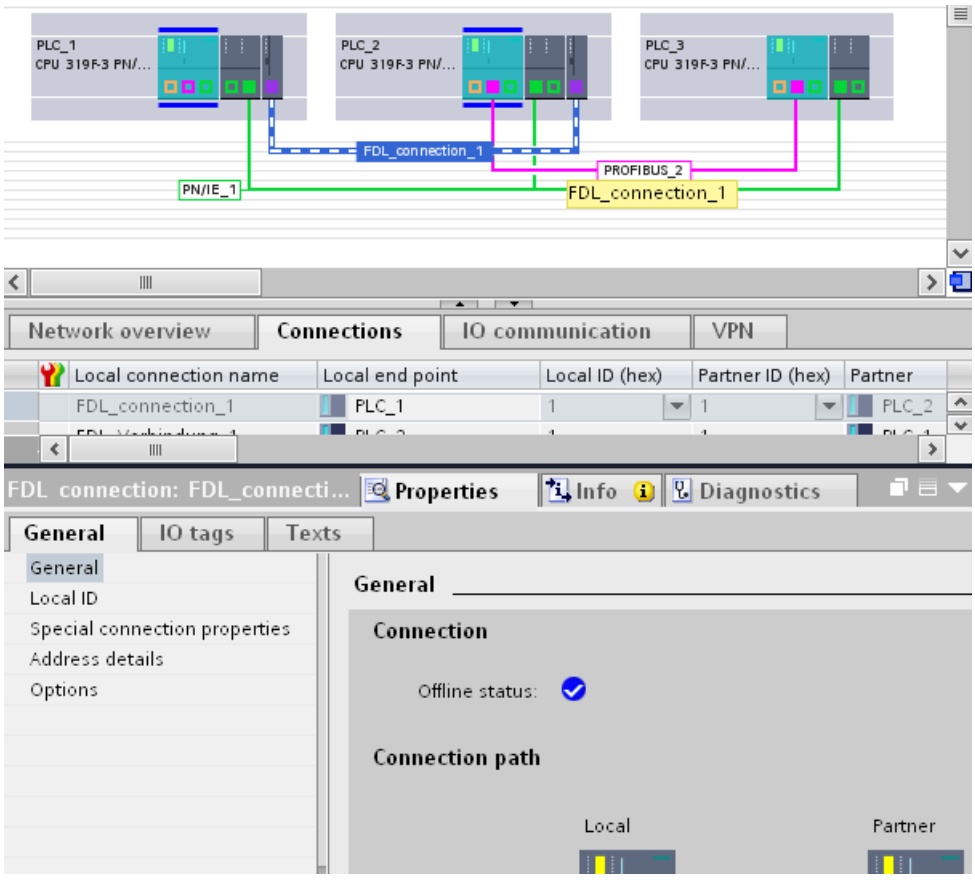

Working with the connection table (S7-300, S7-400, S7-1500)

Functions

As well as allowing you to create new connections interactively, the connection table provides other functions:

- List of connections to an object selected in the network view (network, CPU, CP)
- List of all connections in the project
- Selection of a connection and display of connections associated with it in the network view
- Changing connection partners
- Option for entering central parameters
- Display showing status information

Creating new connections interactively

Interactively creating a new connection (Page 215)

Content of the connection table

You can use the connection table to display the details for all communication connections created in the project:

- Show all connections (default setting - can be selected via the shortcut menu)
  The connection table displays all connections created for the project.
- Show selected connections (recommended setting for complex projects)
  To access this display, clear the "Show all connections" option in the shortcut menu.
  The connection table shows the connections belonging to a selected subnet or selected CPU and/or CP. In other words, it's empty when none of these named objects are selected in the network view.
  Multiple objects of the same type (multiple subnets, multiple CPUs/CPs or multiple interfaces) can also be selected at the same time. The connection table then shows you all associated connections.

Basic functions for tables

The connection table supports the following basic functions for editing a table:

- Showing and hiding specific table columns
  Select the "Show/hide columns" command in the table header in the shortcut menu.
- Changing column width manually
  Move the dividing line in the header on the right of the column.
- Optimizing column width automatically
  Select the shortcut menu command "Optimize column width" in the table header.
- Showing column meaning using tooltips.

If you want to store the layout, width and visibility of the table columns, click on the "Remember layout" function in the top right-hand of the network view.

Icon	Meaning
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Using cursor keys to move within the connection table

You can use the UP and DOWN cursor keys to select a connection from the connection table; the selected connection will be marked and highlighted in the network view when connection mode is activated.

Changing properties of connection

You can directly edit some of the parameters displayed in the connection table. The name of the connection can, for example, only be changed in the connection table.

Changing connection partners

You can change the connection partner of a connection as follows:

1. Select the connection.
2. Select the new connection partner from the open drop-down list in the "Partner" column.

See also

Overview of connection diagnostics (Page 3360)

Displaying the connection status using icons (Page 3361)

Detailed connection diagnostics (Page 3362)

Changing the connection partner (S7-300, S7-400, S7-1500)

Follow the procedure below if you need to change the connection partner for a connection already configured. The local ID and the connection type will be maintained.

Changes can only be made to connection endpoints in the connection table or in the inspector window.

Changing connections interactively

To reach a different connection partner on a default connection path proceed as follows:

1. In the network view, select the local endpoint to which you wish to assign a new connection partner.
   The endpoint connections will be displayed in the connection table.
2. Select the desired connection in the connection table.
3. Select the new target device for the connection from the drop-down list in the "Partner" table column.
   You can alternatively select the new target device under "Connection path" in the inspector window.

Note

Effect on previously configured properties

Note that when changing the connection partner the parameter assigned properties of the connection must be reset to the pre-select.

Use the entry possibilities in the connection table or in the Inspector window if you want to change the "Properties" of the connection.

Note

Effect on networking

Subnets will not automatically be created in the absence of networking to the new connection endpoint in procedures for changing the connection partner.

Setting or changing the connection path (S7-300, S7-400, S7-1500)

Depending on the components and interfaces available within a device, connections can take different paths within the device.

Connections in the configuration will have one of the following configuration statuses in line with the status of the device configuration and will be displayed accordingly:

- Connection path closed
  Both endpoints (CPU or PC application) and the interfaces required for communication (local interface in the CPU or additional CPU) are available.

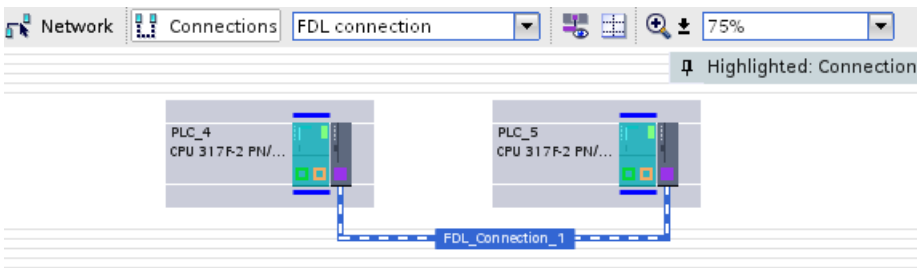

- Connection path open
  - The connection path is not fully assignable as interfaces are missing.
  - The connection path is only specified at one endpoint because it is an unspecified connection.

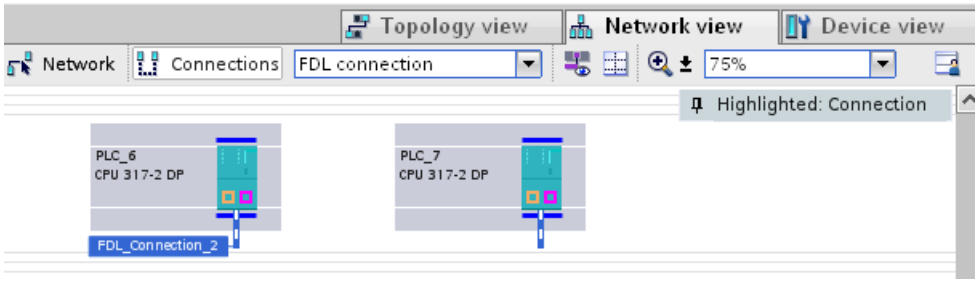

The procedure for changing or reassigning a configured connection path in a local or partner device is described below.

You can set or change the connection path in the following views:

- Inspector window - "Properties > General"

Changing the connection path in the Inspector window

To assign a different connection path to an existing connection:

1. In the network view, select the endpoint (CPU or PC application) for which a connection is to be changed.
2. In the connection table select the connection for which you want to change the connection path.
   Hint: The current connection path is highlighted in the network view if you have set the "Highlight connection" view.
3. Select the new interface in the Inspector window under " Properties > General > Interface". The partner is offered the interfaces that can be considered in the context of the local interface (Note: S7 connections can be established via different subnet types).
   In the network view the new connection path is highlighted.

Assigning connection path

To assign a closed connection path to an existing open connection path:

1. Expand the device configuration to that the interfaces required for the connection type are available for both partners.
2. In the connection table, select the connection for which you want to change the connection path.
   Hint: The current connection path will be highlighted in the network view if you have set the "Highlight connection" view accordingly.
3. Activate the "Find connection path" button under "Properties > General > Interface" in the inspector window.
   In the network view the new connection path is highlighted.

Creating an unspecified connection (S7-300, S7-400, S7-1500)

Create unspecified connections in the following cases by only selecting the local connection partner graphically or interactively:

- When certain service or connection properties require an unspecified connection (examples: E-mail connection, FTP connection, FDL connection with the "Free layer 2 access" property).
- If you want to set up a connection to be available for communication with any connection partner.
- If one of the two partners for the selected connection type is not (yet) available, but you want to reserve a connection resource.
- If the connection partner is located outside of the project (e.g. third-party stations).
- UDP connection with broadcast / multicast:
  - See UDP with broadcast and multicast (Page 258)
  - See Free UDP connection (Page 261)

Requirement

You must be in the network view.

Configuring an unspecified connection graphically

To create an unspecified connection graphically, proceed as follows:

1. In connection mode, select the required connection type in the drop-down list box, for example "TCP connection".

   The CPUs available for the selected connection type are selected and highlighted.
2. Click on the CPU from which a connection will be set up.
3. Move the mouse cursor within the CPU or on the interface of the CPU and confirm the connection endpoint by double-clicking the mouse.

Result

- An unspecified connection is created. Depending on the selected endpoint - CPU or CP - the following applies for path selection:
  - endpoint CPU: The best possible local connection path will be selected automatically. See also Creating a new connection graphically (Page 211)
  - endpoint CP interface: The connection path is determined by the entry.
- The unspecified connection is highlighted in the display by a short dotted line.

- The unspecified connection is entered in the connection table of the device and "Unknown" is entered in the "Partner" column.

Configuring the unspecified connection interactively

To interactively create an unspecified connection, proceed as follows:

1. Select the "Add connection" menu command in the context menu of the device (CPU) for which you want to create a connection.
   or
   With the "Display all connections" option disabled, select the "Add connection" command in the shortcut menu of the connection table.
   Result: The "Create new connection" dialog will open.
2. In the "Create new connection" dialog box select the desired connection type, e.g. "FDL connection".
3. In the top of the dialog box select the entry "Unspecified".
4. If several local interfaces are available, you can select the required interface from a list. The first row of the list displays the interface the system detected as the most suitable and this is selected.
5. Confirm the selection with the "Add" button.
   An unspecified connection will be created and will be visible in the connection table.
6. Repeat the process or exit the dialog via "OK".

Result

- One or more unspecified connections are created.
- The unspecified connections are entered in the connection table.

Note

Note that you can still assign parameters for special properties for each connection.

Creating UDP / FDL connections with broadcast / multicast (S7-300, S7-400, S7-1500)

Meaning

Multicast and broadcast are special connection options that are supported by Industrial Ethernet CPs on UDP connections and by PROFIBUS on FDL connections and can be configured.

Creating a connection interactively - procedure

To create a multicast / broadcast connection interactively, follow these steps:

1. Select the "Add connection" command in the shortcut menu of the component that will be the connection endpoint.
   or
   With the "Display all connections" option disabled, select the "Add connection" command in the shortcut menu of the connection table.
   The "Create new connection" dialog opens.
2. Select the connection type UDP connection / FDL connection" from the "Type" drop-down list.
3. Select the connection type broadcast or multicast in the left-hand part of the dialog.
   The local interfaces possible for the configuration together with status information are displayed in the content area. The first interface to which a connection of the selected type can be assigned is selected.
4. Leave the selection as it is or select the connection path you require.
5. Confirm the selection with the "Add" button.
   A connection is created and can be seen in the connection table.
6. If you have not yet completed the dialog, you have the alternative of continuing as follows:
   – You can create another connection of the same type to the same target device by clicking the "Add" button again.
   – Repeat the steps described above by selecting another connection type from the drop-down list or another connection partner.

Creating a connection graphically - procedure

1. First, create an unspecified UDP connection / FDL connection using the graphic connection configuration tool; Creating a new connection graphically (Page 211)
2. Now make the address settings for the connection partner according to the requirements for multicast or broadcast. As an alternative, follow these steps:
   – In to the connection table, go to the "Partner" column and select the required connection type from the drop-down list; address parameters are assigned automatically.
   or
   – For UDP, enter the IP address or port directly in "General > Address details". For FDL, set the PROFIBUS station address 63.

See also

UDP with broadcast and multicast (Page 258)

FDL connection with broadcast (Page 270)

FDL connection with multicast (Page 272)

Specifying an unspecified connection (S7-300, S7-400, S7-1500)

Characteristics of the unspecified connection

For an unspecified connection depending on the connection type either some or all of the parameters that are necessary for local access to the network and for addressing the partner, will be missing.

An unspecified connection is shown as follows:

- Network view:
  The unspecified connection will be displayed in the network view as a dotted line once it has been created or selected in the connection table.
- Connection table:
  - Display in the "Partner" column: "Unknown"

An unspecified connection occurs as described below: See also Creating an unspecified connection (Page 226)

Below are the main two different scenarios when specifying the connection later:

- The connection partner is in the same project
  You specify the connection by the subsequent selection of components available in the project.
- The connection partner is not in the project
  You specify the connection by entering the address parameters of the communication partner available outside the project.

Automatic supplementing of missing connection parameters

If the missing resources have been supplemented by adding components then connections will be assigned automatically if possible, and if necessary to an unspecified connection.

Requirement

- The components of both devices have free resources for the connection type.

Procedure in the connection table

Proceed as follows to provide an unspecified connection with the address parameters of a target device already in the project:

1. Select the CPU for which you want to specify a connection.
   In the connection table, all unspecified connections configured for the CPU have the entry "Unknown" in the "Partner" column.
2. Select the required target device in the "Partner" column of the connection table.

Result

- The connection is specified with the relevant address details for the connection partner.
- The new connection path is highlighted in the network view as a dotted line.

The figure below shows a specified connection:

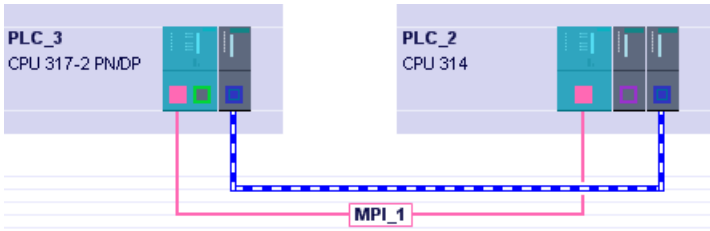

Deleting a connection (S7-300, S7-400, S7-1500)

You can delete configured connections via the network view or the connection table.

In the network view you can delete a highlighted connection. In the connection table you can delete one or more connections.

Procedure in the network view

You can delete individual connections in the network view via the shortcut menu. Follow these steps:

1. Switch to connection mode.
2. Place the cursor on the subnet.
   The "Highlight connection" dialog will show the connections currently configured.
3. Select the connection to be deleted.
4. Place the cursor on the subnet again.

5. Open the shortcut menu with a right mouse click.
6. Select the "Delete" command.
   The selected connection will be deleted in full. If the partner for a specified connection is within the project, the connection will also be deleted from the connection table of the partner.

Note

Downloading connection data to a module

Download the connection table with the deleted connection to the programmable modules affected by it.

Procedure in the connection table / network view

You can delete one or more selected connections in the connection table via the shortcut menu. Follow these steps:

1. In the connection table, select the connections that you want to delete.
2. Open the shortcut menu with a right mouse click.
3. Select the "Delete" command.
   Each connection selected will be deleted in full. If the partner for a specified connection is within the project, the connection will also be deleted from the connection table of the partner.
4. Load the connection table with the deleted connections into the appropriate programmable module.

Note

Downloading connection data to a module

Download the connection table with the deleted connection to the modules affected by it.

Note

To delete all connections of the module you must load an empty connection table.

Copying a connection (S7-300, S7-400, S7-1500)

Introduction

Connections are not copied singly but always in context along with the project or the device.

You can copy:

- Entire projects
- One or more devices within a project or from one project to another

Copying a project

When you copy a project all configured connections will also be copied. No new settings are required for the copied connections as the connections remain consistent.

Copying devices

- Copying a device within a project
  Copying individual devices produces unspecified connections and you will need to reassign the connection partners to the local node.
  If you copy a group of devices, the connections within the group will also be copied.
  Duplicating will however also produce duplicate connection names. Change the duplicate connection names after copying.
  You will need to network the copied devices again.
- Copying devices to another project
  You can copy one or more devices to another project. To continue editing the other project, restart the TIA Portal.
  Connections configured between the copied devices remain in the target project. You need only reassign networking in the target project.
  Copying the devices to the target project individually will produce temporarily unspecified connections. However, the communication partners will be assigned again if this is possible when the next device is added.

Additional information

See also Specifying an unspecified connection (Page 229)

Inconsistent connections - connections without assignment (S7-300, S7-400, S7-1500)

An inconsistent connection is one in which the connection data is incomplete or inconsistent; the connection is not functional in the context with the project.

Inconsistent connections cannot be loaded - operation is not possible with such connections.

In the connection table inconsistent connections are marked in red.

Note

Unspecified connections

Unspecified connections are not necessarily inconsistent although they are always highlighted in red in the connection table.

Creating an unspecified connection (Page 226)

Possible causes for inconsistent connections

- Deletion or change of the hardware configuration.
- Missing interface network links in the project, which are necessary for a connection.

- Connection resources are exceeded
- Connections to an unspecified connection partner without partner address information.

Detailed information about the cause of the inconsistency is provided in the inspector window in the properties page of a connection; inconsistent parameters are shown in red. You can get supplemental information about these parameters by placing the cursor on the parameter.

Examples of and remedies for inconsistencies associated with typical configuration actions

Actions are explained below that can cause configured connections to lose their assignment or be deleted.

Note

Note that as opposed to the S7 connections, a CP-dependent ID is assigned to the connections of the SEND/RECEIVE interface. Consequently the actions described below may require customizing of the ID, which means that the interface must also be customized in the user program.

If a CP is replaced by a different CP then it must provide at least the same services and have at least the same version. This is the only way to ensure that the connections configured via the CP remain consistent and can be used.

Table 1-3 Actions for a CP interface that can cause changes in configured connections.

Action	Consequence for the connections	What you must do to restore the connection
Delete the CP (module).	The connections remain intact in the connection table without assignment to a CP. The assignment to the CPU / PC application remains.	After you have placed and networked a CP in the hardware configuration: 1. Assign the CP to the connection; 2. Check the module start address LADDR and customize if necessary in the user program. 3. Customize the connection IDs in the user program. 4. Reload the connection configuration into the CP.
Delete the device.	All the connections to this device are canceled in the project. • Connections remain as unspecified connections on the partner. • Connections without a connection partner are completely deleted.	Reconfigure the device and connections. Either reassign unspecified connections or delete them.
Change the subnet assignment of the CP.	The connections that were assigned via the CP remain but may be inconsistent.	Where necessary, reassign connections.

1.3.2.4 Connection types (S7-300, S7-400, S7-1500)

S7 connection (S7-300, S7-400, S7-1500)

S7 connection - general settings (S7-300, S7-400, S7-1500)

General connection parameters

General connection parameters are displayed in the "General" parameter group under the properties of the connection; these connection parameters identify the local connection endpoint.

Here, you can assign the connection path and specify all aspects of the connection partner.

Local ID

The local ID of the module from which the connection is viewed is displayed here (local partner). You can change the local ID. You may need to do this if you have already programmed communication function blocks, and you want to use the local ID specified in those function blocks for the connection.

Special connection properties

Display of connection properties (can be modified depending on the components used):

- Configured at one end
  Configured at one end means that the connection partner functions as a server on this connection and cannot send or receive actively.
  Communications instructions used at both ends (with "pairs of blocks") cannot be used on connections configured at one end; for example USEND and URCV, BSEND and BRCV or USTATUS.
- Active connection establishment
  Use this option to specify whether the connection is established from this device.
  If you selected "Unspecified" as the connection partner when you created the connection, the option is deselected by default. If you select this option, you must specify the address of the partner in the "Addresses" tab.
- Send operating mode messages
  Indicates whether or not the local partner sends operating mode messages to the connection partner.

Address details

Displaying address details of the S7 connection. With an unspecified partner, the values for the rack and slot can be changed. All other values are obtained from the current configuration and cannot be changed.

S7 connection - address details (S7-300, S7-400, S7-1500)

Meaning

A connection resource such as shown under "Properties > Address details" is part of the TSAP (Transport Service Access Point) of the local module or partner.

When a connection is established, the connection-specific resources of a module are assigned permanently to this connection. This assignment requires that the connection resource can be addressed. The TSAP (Transport Service Access Point) is, as it were, the address of the resource that is formed with the help of the connection resource or, in the case of S7-1500 CPUs, with SIMATIC-ACC (SIMATIC Application Controlled Communication).

The ability to use connection resources is subject to rules and restrictions. For example, not every connection resource can be used for every connection type.

Structure of the TSAP

- For S7-1500 CPU:
  "SIMATIC-ACC"<nnn><mm>
  nnn = Local ID
  mm = any value
- For S7-300/400 CPU:
  <xx>.<yz>
  xx = Number of the connection resource
  y = Rack number
  z = Slot number

TSAP structure, dependent on partner

The configuration of the TSAP for S7-1500 CPUs is dependent on the remote connection partner. When an S7-1500 CPU is connected to an S7-300/400 CPU, an S7-1500 CPU also uses a TSAP configuration that includes the connection resource.

See the following examples of TSAPs of various connection configurations

- Connection between two S7-1500 CPUs:
  - S7-1500 CPU "A" with local ID 200:
    TSAP: SIMATIC-ACC20001
  - S7-1500 CPU "B" with local ID 3B2:
    TSAP: SIMATIC-ACC3B201
- Connection between S7-1500 CPU and S7-300/400 CPU:
  - S7-1500 CPU (rack 0, slot 1, connection resource 10):
    TSAP: 10.01
  - S7-300/400 CPU (rack 0, slot 2, connection resource 11):
    TSAP: 11.02

Value ranges of connection resource

The following table provides information on the meaning of the values and the type of connection. Depending on the connection partner and connection type, the range of values is automatically limited to valid values or the value of the connection resource is assigned permanently.

Connection resource (1st byte of TSAP)	Meaning	Type	Meaning
0x01 ("PG")	Programming device connection	Free connection (not configured)	At least one resource per CPU is reserved for programming device connections. However, for certain S7-300 CPUs it is possible to reserve multiple resources in the CPU properties.
0x02 ("OP")	OP connection	Free connection (not configured)	At least one resource per CPU is reserved for OP connections. However, for certain S7-300 CPUs it is possible to reserve multiple resources in the CPU properties.
0x03	Other	Free connection (configured, unspecified connection)	This connection resource can operate multiple connections. Use: Connection configured at one end with unspecified connection partner! The connection partner does not have to be configured if the connection resource 0x03 is addressed.
0x10..0xDF	Connections with static or dynamic connection setup	Configured unspecified connection	One of these connection resources can operate one (and only one) connection. Use: Connection configured at both ends with unspecified connection partner!

Free S7 connections

Free S7 connections are set up dynamically during runtime; the following connections are free:

- Programming device connections (0x01)
  S7 connections that are typically set up from a programming device or from a PC (with ES functionality). This type of connection is used to configure and program the addressed station/module as well as to test and commission it; afterwards, the connection is typically cleared again. This connection resource allows both read and write access (e.g., monitoring and loading).
- OP connections (0x02)
  S7 connections that are typically set up from an OP or from a PC (with OS functionality). This type of connection is used to monitor the addressed station/module with regard to the process that is being controlled.

- Other (0x03)
  Use is not specified. For example, this resource is used automatically when an S7 connection configured at both ends is configured from an S7-400 to an S7-300.
- S7 basic communication (0xFD)
  Connections that are typically set up from a CPU to another module (CPU, FM, etc.) within a subnet. The connection setup is initiated by the application program, in which a connection configuration does not exist. This type of connection allows process data to be exchanged between the modules. For certain S7-300 CPUs, it is possible to reserve resources for S7 basic communication.

Recommendation for selection of the connection resource

STEP 7 always suggests a free connection resource for the local partner. We recommend that you accept it.

ISO transport connection (S7-300, S7-400, S7-1500)

ISO transport – special connection properties (S7-300, S7-400, S7-1500)

Relevance

You can specify additional properties of the local connection endpoint in the "Properties > General > Special connection properties" parameter group.

Establishing an active connection

Use this option to specify whether the connection is established from this device. This is the default option if the address of the partner is specified.

- ON: Connection is established actively.
- OFF: Connection is established by the partner.

If you selected "Unspecified" as the connection partner when you created the connection, the option is deselected by default. If you select the option you must specify the address of the partner in the "Addresses" tab.

Note

Effect on the mode

Note the effect on the mode.

- S7-300 / S7-400
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), disable the "Active connection establishment" option.
- OPC server
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), enable the "Active connection establishment" option.

Name

A connection endpoint name is proposed here when you create the connection. This name contains a connection number as suffix.

Use this field to identify the partner on unspecified connections.

ISO OPC establishment details (S7-300, S7-400, S7-1500)

Relevance

Connection configuration for OPC in the parameter group "Properties > General > Establishment details".

Options for connection establishment

Entries in this area are possible only if you have selected active connection establishment in "Special connection properties".

Choose from the two following alternatives:

- Keeping the connection established permanently
  The connection is established permanently when you start up the PC station and is re-established following an interruption on the connection.
- Establishing the connection on request (access to tag)
  You can also specify how long the connection should stay up after access.

Monitoring connection establishment

- Timeout: Error wait time during the connection establishment
  If, during this wait time, the connection cannot be established, the connection is reset and a pending request acknowledged with an error.
  Range of values: 0 to 99999 ms
  Default: 15000 ms

  Note

  The connection establishment monitoring that can be set here, has the following relationship with the time monitoring that can be set in the module configuration: The lower value is valid.

  Time monitoring can be configured in the module configuration in "Properties Ethernet interface".

ISO transport address details (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Address parameter

An ISO transport connection is specified by the local and remote connection endpoint.

- Local addresses:
  Local MAC address and local TSAP (Transport Service Access Point)
- Remote addresses:
  Remote MAC address and remote TSAP

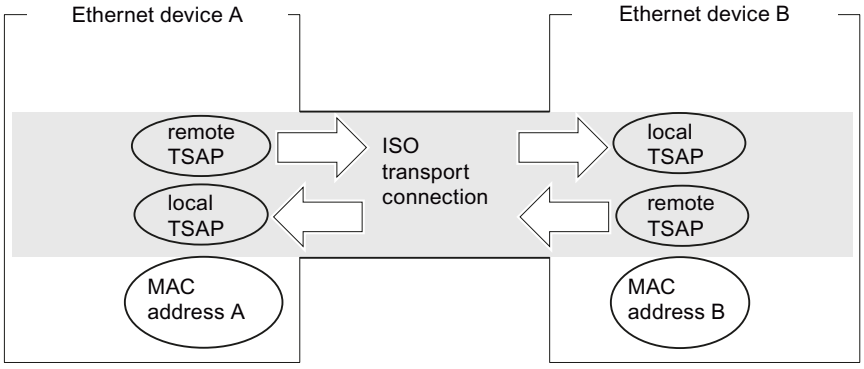

Figure 1-4 Sending and receiving via one ISO transport connection each

TSAP format

ISO transport connections have a TSAP length of 1 to 16 bytes. For the entry the actual length is displayed automatically (visible display: 16 ASCII characters). Local and remote TSAPs can be entered as a hexadecimal value or as an ASCII string. For ASCII input, the entered characters are also displayed in hexadecimal. With hexadecimal input, printable characters are shown as an ASCII value (8 hexadecimal characters are visible). If non-printable characters are entered, the ASCII display is greyed out (ASCII input is no longer possible) and the non-printable characters are shown as periods.

Local and remote TSAPs

Remote and local TSAPs can be identical because the connection through the different MAC addresses is unique. If more than one connection is to be set up between two stations, the TSAPs must differ.

- Specified connection
  The TSAPs of an ISO transport connection must agree as follows:
  - Remote TSAP (on the Ethernet CP) = local TSAP (on the target station)
  - Local TSAP (on the Ethernet CP) = remote TSAP (on the target station)
- Unspecified connection

Some address parameters are not specified: Unspecified connections (Page 203)

Default TSAPs

Default values (changeable) are recommended for configuring local and remote TSAPs (for example ISO-1 for the first connection between two partners). If new connections are configured between the same partners, the default values will be incremented automatically (for example ISO-2, etc.). For a new connection to a new partner, the default value will start with ISO-1 again.

ISO transport options (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Options".

Operating modes

- SEND/RECV:
  This is the standard mode with the AG_SEND/AG_LSEND, and AG_RECV/AG_LRECV FCs.
- FETCH/WRITE:
  System memory areas in the SIMATIC S7 can be accessed from the following devices if you select one of the operating modes below for the ISO transport connection:
  - SIMATIC S5
  - SIMATIC PC stations
  - Third-party devices

  The connection can then only be used for this mode. It will not be possible to send or receive via the AG_SEND/AG_LSEND or AG_RECV/AG_LRECV instructions.
  - FETCH
    Selecting the operating mode Fetch enables direct read access to data areas in the SIMATIC S7. Access can, for example, be from a SIMATIC S5 station or a third-party station.
  - WRITE
    Selecting the operating mode Write enables direct write access to data areas in the SIMATIC S7. Access can, for example, be from a SIMATIC S5 station or a third-party station.
- SPEED SEND/RECV:
  Selecting the SPEED SEND/RECV mode allows you to use the AG_SSEND and AG_SRECV instructions.
  AG_SSEND/AG_SRECV (SPEED SEND/RECEIVE) provide faster and therefore optimized transmission within a S7-400 station. Check the device documentation of the CP you are using to see whether this operating mode is supported by the CP and in conjunction with the selected CPU.

FETCH/WRITE requirements: Mode of connection establishment

Please select one of the following options in the parameter group "Properties > General > Special connection properties" in line with the station type for connection establishment.

- SIMATIC S7 station: PASSIVE connection establishment
- PC station: Active connection establishment

"S7 addressing mode" option (for FETCH/WRITE only)

You can select the addressing mode when you configure the FETCH ACTIVE/WRITE ACTIVE mode.

- S7 addressing mode: Byte address
- S5 addressing mode: Word address

This allows applications to access S5 or S7 stations without modifying the addresses. This is particularly useful for existing S5 applications, for example, that can now be used unchanged when accessing S7 stations.

The addressing mode for accessing SIMATIC S7 (option selected) is set by default.

See also

Connection configuration for FETCH/WRITE (Page 265)

Use of FETCH/WRITE (Page 263)

ISO transport - dynamics (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Dynamics".

The relevant timers and counters for this connection are displayed. You can accept these default values.

Where necessary (for example when linking to third-party systems), you can also set the timers and counters individually to achieve the required dynamic response of the connection.

ISO OPC properties (S7-300, S7-400, S7-1500)

Relevance

Connection configuration for OPC in the parameter group "Properties > General > OPC".

Here, you set connection-specific properties for an ISO transport connection used by the OPC server.

You can set the following parameters:

Timeout for job

If a pending job cannot be processed during the wait time set here, the connection is reset and the job acknowledged with an error.

Range of values: 0 to 99999 ms

- Default setting: 15000 ms

Send buffer size

Default size of the default send buffer. Additional send buffers of any length (up to 65535 bytes) may be used at runtime.

Range of values:

- Default setting: 256 bytes

- Range of values: 65535 bytes maximum

"Optimization of direct access for tag services" option

If you use this option, the OPC server attempts to combine direct accesses to several tags to minimize the number of jobs that need to be sent.

This option is enabled by default.

"Immediate response when interrupted connection is detected" option

If you select this option, you specify that the application is notified that a communication job (FETCH, or WRITE) cannot be executed when a connection is unavailable. The code is sent regardless of whether or not a current attempt to establish the connection succeeds.

If you do not select this option, the application is informed of the nonavailability of a connection only when the connection times out.

The option described here is relevant and available only when the two modes listed below are used:

- The connection is configured as a permanent connection.
  Connections can be configured as permanently established connections (in "Details of connection establishment"). This means that jobs can be forwarded to the partner with the shortest possible reaction time without needing to establish the connection first.
  If there is a disruption on the connection, the system automatically attempts to re-establish the connection. In this situation, there may be delays before a message can be forwarded to the application.
- You are using the FETCH or WRITE job types (can be selected in "Options").

Only with these job types is an immediate reaction necessary in certain situations if there is no connection.

ISO-on-TCP connection (S7-300, S7-400, S7-1500)

ISO-on-TCP - special connection properties (S7-300, S7-400, S7-1500)

Relevance

You can specify additional properties of the local connection endpoint in the "Properties > General > Special connection properties" parameter group.

Active connection establishment

Use this option to specify whether the connection is established from this device. This is the default option if the address of the partner is specified.

- ON: Connection is established actively.
- OFF: Connection is established by the partner.

If you selected "Unspecified" as the connection partner when you created the connection, the option is deselected by default. If you select the option you must specify the address of the partner in the "Addresses" tab.

Note

Effect on the mode

Note the effect on the mode.

- S7-300 / S7-400
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), disable the "Active connection establishment" option.
- OPC server
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), enable the "Active connection establishment" option.

Name

A connection endpoint name is proposed here when you create the connection. This name contains a connection number as suffix.

Use this field to identify the partner on unspecified connections.

ISO-on-TCP OPC establishment details (S7-300, S7-400, S7-1500)

Relevance

Connection configuration for OPC in the parameter group "Properties > General > Establishment details".

Options for connection establishment

Entries in this area are possible only if you have selected active connection establishment in "Special connection properties".

Choose from the two following alternatives:

- The connection is established permanently
  The connection is established permanently when you start up the PC station and is re-established following an interruption on the connection.
- The connection is established when it is required (access to tag)
  You can also specify how long the connection should stay up after access.

Connection establishment monitoring

Timeout: Error wait time during the connection establishment

If, during this wait time, the connection cannot be established, the connection is reset and a pending request acknowledged with an error.

Range of values: 0 to 99999 ms

Default: 15000 ms

Note

The connection establishment monitoring that can be set here, has the following relationship with the time monitoring that can be set in the module configuration: The lower value is valid.

Time monitoring can be configured in the module configuration in "Properties Ethernet interface".

ISO-on-TCP address details (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Address parameter

A connection is specified through the local and remote connection endpoint.

- Local addresses:
  Local IP address and local TSAP (Transport Service Access Point)
- Remote addresses:
  Remote IP address and remote TSAP

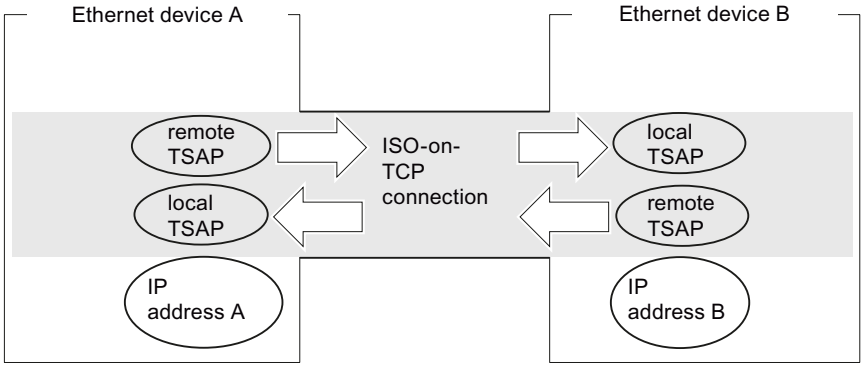

Figure 1-5 Sending and receiving via an ISO-on-TCP transport connection

Addresses tab

In the Addresses tab the relevant local and remote address information is displayed as recommended values. You have the possibility of setting the TSAPs individually.

TSAP format

ISO-on-TCP connections have a TSAP length of 1 to 16 bytes. For the entry the actual length is displayed automatically (visible display: 16 ASCII characters). Local and remote TSAPs can be entered as a hexadecimal value or as an ASCII string. For ASCII input, the entered characters are also displayed in hexadecimal. With hexadecimal input, printable characters are shown as an ASCII value (8 hexadecimal characters are visible). If non-printable characters are entered, the ASCII display is greyed out (ASCII input is no longer possible) and the non-printable characters are shown as periods.

Local and remote TSAPs

Remote and local TSAPs can be identical because the connection through the different IP addresses is unique. If more than one connection will be set up between two stations then the TSAPs must also differ.

- Specified connection
  In the configuration of the Ethernet CP and the Ethernet target Station, the TSAPs of an ISO-on-TCP connection must be crossed over.
  - Remote TSAP (on the Ethernet CP) = local TSAP (on the target station)
  - Local TSAP (on the Ethernet CP) = remote TSAP (on the target station)
- Unspecified connection
  Some address parameters are not specified: Unspecified connections (Page 203)

Default TSAPs

For configuration of local and remote TSAPs there is a default value "TCP-1" for the first connection between both partners (changeable). For a new connection between the same partners the standard value "TCP-2" is suggested. For a new connection to a new partner TCP-1 is started again.

ISO-on-TCP general (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Options".

Operating modes

- SEND/RECV:
  This is the standard mode with the AG_SEND/AG_LSEND, and AG_RECV/AG_LRECV FCs.
- FETCH/WRITE:
  System memory areas in the SIMATIC S7 can be accessed from the following devices if you select one of the operating modes below for the ISO transport connection:
  - SIMATIC S5
  - SIMATIC PC stations
  - Third-party devices

  The connection can then only be used for this mode. It will not be possible to send or receive via the AG_SEND/AG_LSEND or AG_RECV/AG_LRECV instructions.
  - FETCH
    Selecting the operating mode Fetch enables direct read access to data areas in the SIMATIC S7. Access can for example be from a SIMATIC S5 station or a third-party station.
  - WRITE
    Selecting the operating mode Write enables direct write access to data areas in the SIMATIC S7. Access can for example be from a SIMATIC S5 station or a third-party station.
- SPEED SEND/RECV:
  Selecting the operating mode SPEED SEND/RECV allows you to use the instructions AG_SSEND and AG_SRECV.
  AG_SSEND/AG_SRECV (SPEED SEND/RECEIVE) provide faster and therefore optimized transmission within a S7-400 station. Check the device documentation to see whether this operating mode is supported by the CP used.

FETCH/WRITE requirements: Mode of connection establishment

Please select one of the following options in the parameter group "Properties > General > Special connection properties" in line with the station type for connection establishment.

- SIMATIC S7 station: PASSIVE connection establishment
- PC station: Active connection setup

"S7 addressing mode" option (for FETCH/WRITE only)

You can select the addressing mode when you configure the FETCH ACTIVE/WRITE ACTIVE mode.

- S7 addressing mode: Byte address
- S5 addressing mode: Word address

This allows applications to access S5 or S7 stations without modifying the addresses. This is, for example, an interesting option for existing S5 applications that are now to be used unchanged to access S7 stations.

The addressing mode for accessing SIMATIC S7 (option selected) is set by default.

See also

Connection configuration for FETCH/WRITE (Page 265)

Use of FETCH/WRITE (Page 263)

ISO-on-TCP OPC properties (S7-300, S7-400, S7-1500)

Relevance

Connection configuration for OPC in the parameter group "Properties > General > OPC".

Here, you set connection-specific properties for an ISO-on_TCP connection used by the OPC server.

You can set the following parameters:

Timeout for job

If a pending job cannot be processed during the wait time set here, the connection is reset and the job acknowledged with an error.

Range of values: 0 to 99999 ms

- Default setting: 15000 ms

Send buffer size

Default size of the default send buffer. Additional send buffers of any length (up to 65535 bytes) may be used at runtime.

Range of values:

- Default setting: 256 bytes

- Range of values: 65535 bytes maximum

"Optimization of direct access for tag services" option

If you use this option, the OPC server attempts to combine direct accesses to several tags to minimize the number of jobs that need to be sent.

This option is enabled by default.

"Immediate response when interrupted connection is detected" option

If you select this option, you specify that the application is notified that a communication job (FETCH, or WRITE) cannot be executed when a connection is unavailable. The code is sent regardless of whether or not a current attempt to establish the connection succeeds.

If you do not select this option, the application is informed of the nonavailability of a connection only when the connection times out.

The option described here is relevant and available only when the two modes listed below are used:

- The connection is configured as a permanent connection.
  Connections can be configured as permanently established connections (in "Details of connection establishment"). This means that jobs can be forwarded to the partner with the shortest possible reaction time without needing to establish the connection first.
  If there is a disruption on the connection, the system automatically attempts to re-establish the connection. In this situation, there may be delays before a message can be forwarded to the application.
- You are using the FETCH or WRITE job types (can be selected in "Options").
  Only with these job types is an immediate reaction necessary in certain situations if there is no connection.

TCP connection (S7-300, S7-400, S7-1500)

TCP - special connection properties (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Special connection properties".

Connection parameters specify additional properties for the local connection endpoint.

Establishing an active connection

Use this option to specify whether the connection is established from this device. This is the default option if the address of the partner is specified.

- ON: Connection is established actively.
- OFF: Connection is established by the partner.

If you selected "Unspecified" as the connection partner when you created the connection, the option is deselected by default. If you select the option you must specify the address of the partner in the "Addresses" tab.

Note

Effect on the mode

Note the effect on the mode.

- S7-300 / S7-400
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), disable the "Active connection establishment" option.
- OPC server
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), enable the "Active connection establishment" option.

FTP mode (with S7-300 / S7-400 - connection unspecified)

Selecting this option has the following results:

- The TCP connection is now used as an FTP connection.
- Parameter group "Address details": The addresses are specified automatically (port = 21; note: This setting is not visible under "Address details").
- Parameter group "Options": The mode is permanently set to FTP.

The "Active connection establishment" option is irrelevant and is not available.

Requirement: This option can only be selected for certain CPs with S7-300 / S7-400 when using an unspecified TCP connection.

TCP OPC establishment details (S7-300, S7-400, S7-1500)

Relevance

Connection configuration for OPC in the parameter group "Properties > General > Establishment details".

Options for connection establishment

Entries in this area are possible only if you have selected active connection establishment in "Special connection properties".

Choose from the two following alternatives:

- The connection is established permanently
  The connection is established permanently when you start up the PC station and is re-established following an interruption on the connection.
- The connection is established when it is required (access to tag)
  You can also specify how long the connection should stay up after access.

Connection establishment monitoring

- Timeout: Error wait time during the connection establishment
  If, during this wait time, the connection cannot be established, the connection is reset and a pending request acknowledged with an error.
  Range of values: 0 to 99999 ms
  Default: 15000 ms

Note

The connection establishment monitoring that can be set here, has the following relationship with the time monitoring that can be set in the module configuration: The lower value is valid.

Time monitoring can be configured in the module configuration in "Properties Ethernet interface".

TCP address details (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Address parameters and connection types

With TCP, the communication partners are addressed through the local and remote endpoint as follows.

- Local addresses: Local IP address and local port
- Remote addresses: Remote IP address and remote port

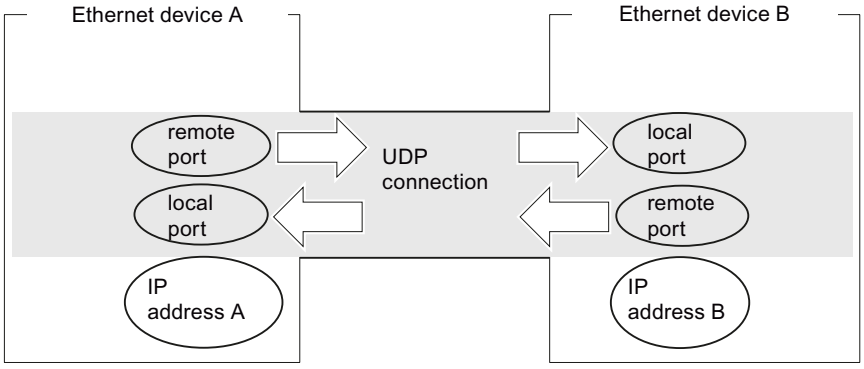

Ports

The ports or the port addresses define the access point to the user program within the station/CPU. They must be unique within the station/CPU!

- Specified connection
  In the configuration of the Ethernet CP and the Ethernet target Station, the ports of a TCP connection must be crossed over.
  - Remote port (on the Ethernet CP) = local port (on the target station)
  - Local port (on the Ethernet CP) = remote port (on the target station)
- Unspecified connection
  Some address parameters are not specified: Unspecified connections (Page 203)

Port range of values - see also

Port settings (Page 254)

TCP options (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Options".

Operating mode

- SEND/RECV:
  This is the standard mode with the AG_SEND/AG_LSEND, and AG_RECV/AG_LRECV FCs.
- FETCH/WRITE:
  System memory areas in the SIMATIC S7 can be accessed from the following devices if you select one of the operating modes below for the TCP connection:
  - SIMATIC S5
  - SIMATIC PC stations
  - Third-party devices

  The connection can then only be used for this mode. It is then not possible to send or receive using the AG_SEND/AG_LSEND or AG_RECV/AG_LRECV FCs.
  - FETCH
    Selecting this operating mode enables direct read access to data areas in the SIMATIC S7. Access can, for example, be from a SIMATIC S5 station or a third-party station.
  - WRITE
    Selecting this operating mode enables direct write access to data areas in the SIMATIC S7. Access can, for example, be from a SIMATIC S5 station or a third-party station.
- FTP
  If you have selected the "Use for FTP protocol" option in "Properties > General > Special connection properties", "FTP" will be displayed here and the TCP connection is then used as an "FTP connection"; no other options can be selected.
  Requirement: The option can only be selected for an unspecified TCP connection.
- SPEED SEND/RECV
  Selecting the SPEED SEND/RECV mode allows you to use the AG_SSEND and AG_SRECV instructions.
  The AG_SSEND/AG_SRECV (SPEED SEND/RECEIVE) instructions provide faster and therefore optimized transmission within an S7-400 station. Check the device documentation to see whether this operating mode is supported by the CP you are using.

FETCH/WRITE requirements: Mode of connection establishment

Select one of the following options for connection establishment in the "General" tab as suitable for the station type.

- SIMATIC S7 station: PASSIVE connection establishment
- PC station: Active connection establishment

"S7 addressing mode" option (for FETCH/WRITE only)

You can select the addressing mode here when you configure for FETCH/WRITE mode.

- S7 addressing mode: Byte address
- S5 addressing mode: Word address

This allows applications to access S5 or S7 stations without modifying the addresses. This is particularly useful for existing S5 applications, for example, that can now be used unchanged when accessing S7 stations.

The addressing mode for accessing SIMATIC S7 (option selected) is set by default.

See also

Connection configuration for FETCH/WRITE (Page 265)

Use of FETCH/WRITE (Page 263)

TCP OPC properties (S7-300, S7-400, S7-1500)

Relevance

Connection configuration for OPC in the parameter group "Properties > General > OPC".

Here, you set connection-specific properties for a TCP connection used by the OPC server.

You can set the following parameters:

Job timeout

If a pending job cannot be processed during the wait time set here, the connection is reset and the job acknowledged with an error.

Range of values:

- Default setting: 15000 ms

Send buffer size

Default size of the default send buffer. Additional send buffers of any length (up to 65535 bytes) may be used at runtime.

Range of values:

- Default setting: 256 bytes
- Range of values: 65535 bytes maximum

Optimization of direct access for tag services

If you use this option, the OPC server attempts to combine direct accesses to several tags to minimize the number of jobs that need to be sent.

This option is enabled by default.

Use miniprotocol

With this option, you specify whether or not the data is transferred as individual fields with length information about the field size (2 bytes in the job buffer). The recipient then reads the data out according to the length information. This property must be configured identically for both communications partners.

As default, the option is disabled; the recipient then reads the data out according to the set receive buffer size.

- Receive buffer size
  If you do not use a miniprotocol, you can set the size of the receive buffer here.
  Range of values: 0 to 65535 bytes
  - Default setting: 256 bytes

Port settings (S7-300, S7-400, S7-1500)

Port settings

The ports or the port addresses define the access point to the user program within the station/CPU. They must be unique within the station/CPU! Please see the section below on the features of unspecific connections to redundant partners.

The following table provides information about the value range:

Port addresses		Use/note
0		Permanently assigned; must not be used!
1..1023		Assigned by default, should not be used (well-known ports)
1024..49151	Ports for application-specific protocols	
	from 2000 to 5000	Range in which the configuration tool searches for and allocates a free port address. You can set the port address in this range individually.
	5001..49151	The port addresses above 5000 are used by the system! Note: If you want to use these port addresses, please contact your system administrator to verify availability.
49152..65535		Dynamically assigned ports The use of these ports is not recommended.

The following local port numbers are reserved; you should not use them for any other purpose during connection configuration.

Port number	Protocol	Service
20, 21	TCP	FTP
25	TCP	SMTP
80	TCP	HTTP
102	TCP	RFC1006
123	UDP (TCP)	NTP *
135	TCP	RPC-DCOM

Port number	Protocol	Service
161	UDP	SNMP_REQUEST
34964	UDP	PN IO
65532	UDP	NTP
65533	UDP	NTP
65534	UDP	NTP
65535	UDP	NTP

* Port 123 is reserved for NTP in the case of CP 443-1 and CP 443-1 Advanced Firmware Version V3 and higher.

Port numbers for unspecific connections to redundant partners

If you create unspecific connections to a redundant partner, for example from an S7-400 with CP to a redundant IEC master, you can use the same local port number in CP 443-1 (Advanced) for the two unspecific TCP connections.

UDP connection (S7-300, S7-400, S7-1500)

UDP address details (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Address parameters and connection types

For UDP the communication partners are addressed through the local and remote endpoint as follows.

- Local addresses:
  Local IP address and local port
- Remote addresses:
  Remote IP address and remote port

Note

The term "connection" is also used here for UDP, because during configuration – as for TCP as well – the communication partners are assigned to each other and thus are logically "connected". In actuality, there is no explicit connection setup between communication partners during station operation in UDP.

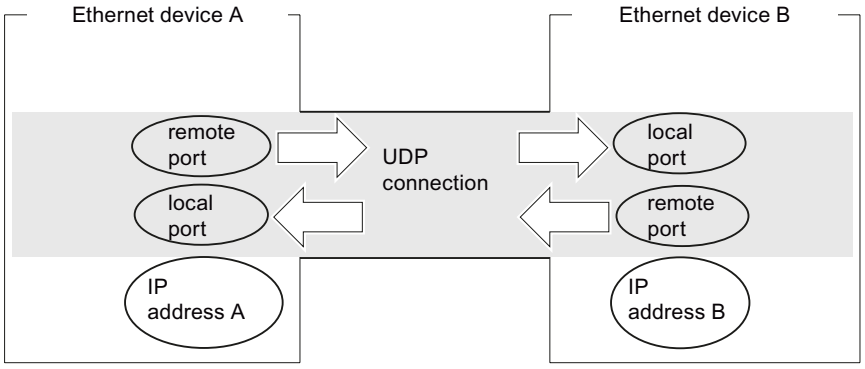

Figure 1-6 Sending and receiving over a UDP connection

Depending on the desired connection type, the remote address parameters are either specified or left open in the configuration.

- Specified UDP connection
  You have specified the connection of a target station when creating the new connection.
- UDP connection specified as broadcast
  You have specified "Broadcast" as the partner when creating the new connection (requires the connection to be created interactively or initially as an unspecified connection).
- UDP connection specified as multicast
  You have specified "Multicast" as the partner when creating the new connection (requires the connection to be created interactively or initially as an unspecified connection).
- Unspecified UDP connection
  You have selected "Unspecified" as the connection type when creating the connection.

Unspecified UDP connection

You can use the unspecified UDP connection in two ways:

- Available UDP connection
  To configure an available UDP connection, select the option "Address assignment in the block" option. In this case, entries can no longer be made in the input fields for the remote IP address and the remote port because the target addresses are now specified by the user program.
- Connection to a "third-party station" in a different project.
  You can specify the remote IP address and the port for any target station. The target station can be inside or outside of the current project.

Note

Partner addresses must be specified

Since there is no connection setup in UDP (datagram service), communication via the configured UDP connection is only possible if the partner addresses (IP address and port) are also specified.

See also

Unspecified connections (Page 203)

Connection to all broadcast devices

If you select "All broadcast devices" for the connection partner, you are specifying that UDP frames will be sent to all accessible broadcast devices.

In the "Address details" parameter group under the IP address (IP) for the partner, a valid broadcast address in the network is suggested for the partner.

Under PORT, you must enter a port address that is appropriate for all partners to be accessed.

Port settings - see also

Port settings (Page 262)

UDP with broadcast and multicast (S7-300, S7-400, S7-1500)

Application

When selecting the connection partner for UDP connections, you also have the two options:

- Connection to all broadcast devices
  If you select "All broadcast devices" for the partner, you specify are specifying that UDP frames are sent to all accessible broadcast devices.

  Note

  With broadcast, S7-CPs can only be used to send; an S7-CP does not allow receipt of user data on broadcast connections, (see below).

- Connection to all multicast devices
  If you select "All multicast devices" for the partner, you are specifying that UDP frames are sent to all accessible broadcast devices.
  Multicast and broadcast are special connection options that are supported by Industrial Ethernet CPs on UDP connections and can be configured.

The frames are sent without acknowledgement because the UDP protocol does not provide for any acknowledgements. This is intended to prevent a data flood of acknowledgements. For example, if frames are sent to 100 partners, then 100 acknowledgements (1 per partner) would arrive at the same time. Such a flood of data could not be analyzed by the sender module.

See also

Creating UDP / FDL connections with broadcast / multicast (Page 228)

When do I use multicast instead of broadcast?

To enable simultaneous sending of a frame to a number of partners, the connection option "Multicast for UDP connections" was introduced.

As opposed to the broadcast connection option, this connection type can also receive frames that are sent to multiple devices in the multicast group.

By specifying a specific receiver group (multicast group), you can prevent extra load on recipients for which the data is not intended. For this reason, multicast is always a better solution than broadcast if frames are sent to groups of partner stations.

Why does an S7-CP not allow user data to be received on broadcast connections?

S7-CPs do not receive any broadcast frames for user data transfer. Only usable frames, such as an ARP request, are forwarded beyond the LAN controller and evaluated. However, S7 CPs can send broadcast frames to the network.

Reason:

Often there is a requirement to send frames to a number of partner devices from one device. In this regard, the important thing is that the frames are sent at the same time and arrive at the partner devices at virtually the same time. Consequently, sending and receiving of

broadcast frames is always required. With a broadcast message, the frame is also actually received by all devices in the network.

A typical application is a situation where broadcast frames are required to find a MAC address for an IP address (ARP request).

Therefore a communication module must always pick up broadcast frames and analyze them from the software side. A significant disadvantage in this regard is that if there are too many broadcast frames on the network, performance drops significantly. This is due to the fact each individual module must process all broadcast frames in order to determine whether the frame was destined for it.

To avoid the disadvantages cited above, the S7-CPs behave as follows relative to broadcast:

- After receipt, the broadcast frames are filtered out with high priority for all Ethernet CPs. In other words, all unusable frames are discarded immediately. Only usable frames, such as an ARP request, are forwarded beyond the LAN controller and evaluated. This prevents the possible negative influence of broadcast frames on the other connections.

Parameter group "Address details" - Connection to all multicast devices

By selecting the "All multicast nodes" for the connection partner, you are specifying the following:

- Sent UDP frames are delivered to all accessible multicast devices of the multicast group.
- The local device for multicast frames is ready to receive in the specified multicast group.

The multicast group is specified via the IP address and the port addresses.

Under IP address (IP) for the partner, an valid IP address in the network is suggested for the multicast group. With multicast, the partner is always a group of recipients (multicast group).

Enter a PORT address under PORT for all partners that you want to access. It must apply to all of these partners.

In principle, it is possible to address multiple multicast groups under one IP address. To do this you can create multiple UDP connections with the same IP address but different PORT addresses.

Note

You should assign identical port addresses for the local port and for the partner port within a multicast group. This is the only way that frames can be both sent and received by the CP within the one multicast group!

Note the following example for three devices in the multicast group:

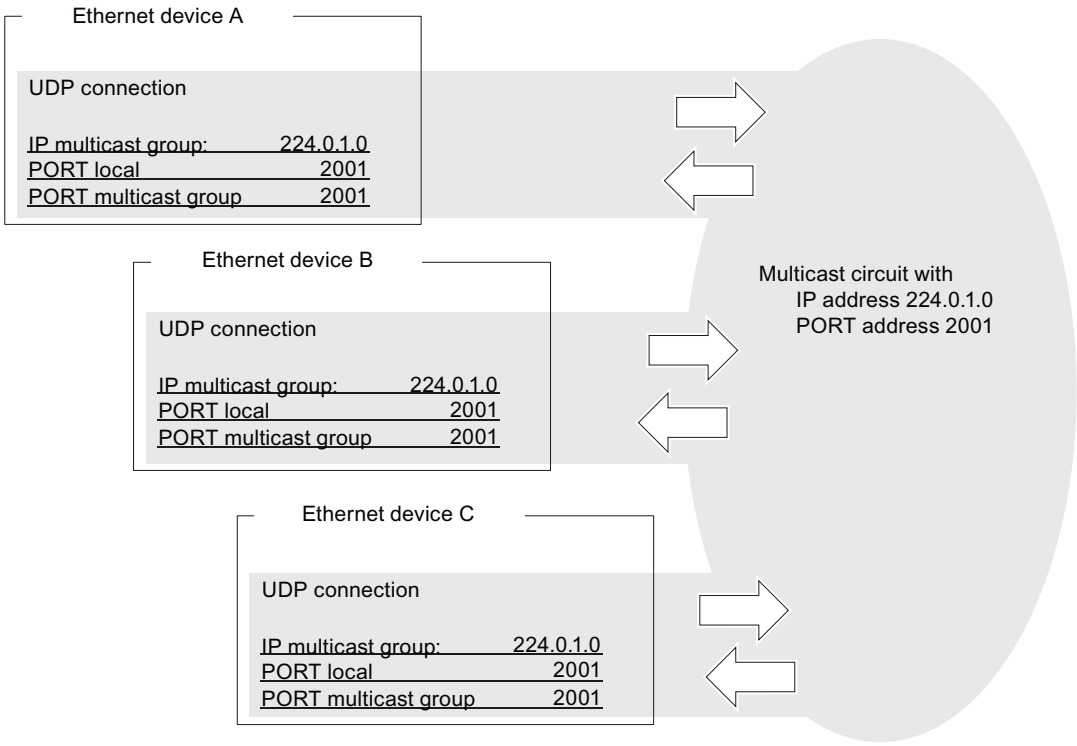

Figure 1-7 Sending and receiving in a multicast group via identical port addresses

IP addresses for IP multicast

- Value range
  For IP multicast, the IP addresses from 224.0.0.0 to 239.255.255.255 can be used. Because the IP addresses up to 224.0.0.255 are reserved for special purposes, we recommend using the IP addresses starting from 224.0.1.0 (default setting) for IP multicast.
- Identification of the multicast group
  A multicast group is not identified via the entire IP address; rather, the first address byte and the most significant bit of the second address byte are ignored. This is an important feature, because it means that apparently different IP addresses can address the same multicast group.
  Example:
  The following IP addresses each address the same multicast group.

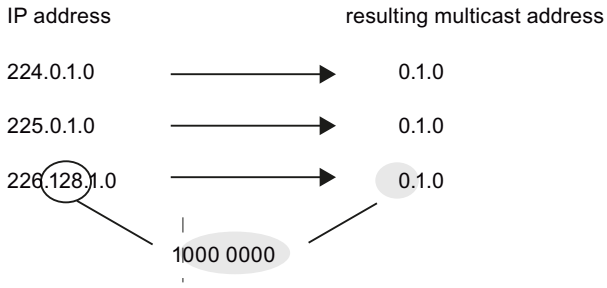

Free UDP connection (S7-300, S7-400, S7-1500)

Program-controlled addressing

A free UDP connection enables program-controlled addressing of the communication peer. Communication between two devices in Industrial Ethernet has the following characteristics:

- Data transfer is bidirectional, i.e. you can send and receive simultaneously on the UDP connection.
- The local station is specified by configuration. The remote device is entered by the user program in the request header of the request buffer with the AG_SEND call. Thus, any device can be accessed on the Ethernet/LAN/WAN.
- The sender's IP address and port can be read from the request header of the AG_RECV.

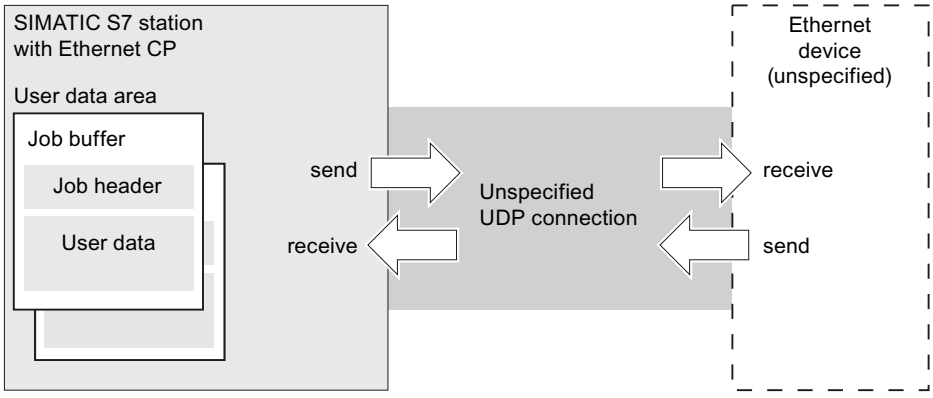

Figure 1-8 Sending and receiving via an unspecified UDP connection - Addressing via program

Data volumes and configuration limit

See the manual provided with the Ethernet CP for the number of UDP connections that the respective Ethernet CP can support. The number of connections per station can be increased by accepting additional CPs .

Each job buffer can transfer up to 2042 bytes of user data. The job header reserves an additional 6 bytes.

UDP options (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Options".

Operating mode

- **SEND/RECV:**
  This is the standard mode with the AG_SEND/AG_LSEND, and AG_RECV/AG_LRECV FCs.
- **SPEED SEND/RECV**
  Selecting the SPEED SEND/RECV mode allows you to use the AG_SSEND and AG_SRECV instructions.
  The AG_SSEND/AG_SRECV (SPEED SEND/RECEIVE) instructions provide faster and therefore optimized transmission within an S7-400 station. Check the device documentation to see whether this operating mode is supported by the CP you are using.

Port settings (S7-300, S7-400, S7-1500)

Port settings

The ports or the port addresses define the access point to the user program within the station/CPU. They must be unique within the station/CPU! Please see the section below on the features of unspecific connections to redundant partners.

The following table provides information about the value range:

Port addresses		Use/note
0		Permanently assigned; must not be used!
1..1023		Assigned by default, should not be used (well-known ports)
1024..49151	Ports for application-specific protocols	
	from 2000 to 5000	Range in which the configuration tool searches for and allocates a free port address. You can set the port address in this range individually.
	5001..49151	The port addresses above 5000 are used by the system! Note: If you want to use these port addresses, please contact your system administrator to verify availability.
49152..65535		Dynamically assigned ports The use of these ports is not recommended.

The following local port numbers are reserved; you should not use them for any other purpose during connection configuration.

Port number	Protocol	Service
20, 21	TCP	FTP
25	TCP	SMTP
80	TCP	HTTP
102	TCP	RFC1006
123	UDP (TCP)	NTP *
135	TCP	RPC-DCOM
161	UDP	SNMP_REQUEST
34964	UDP	PN IO

Port number	Protocol	Service
65532	UDP	NTP
65533	UDP	NTP
65534	UDP	NTP
65535	UDP	NTP

* Port 123 is reserved for NTP in the case of CP 443-1 and CP 443-1 Advanced Firmware Version V3 and higher.

Port numbers for unspecific connections to redundant partners

If you create unspecific connections to a redundant partner, for example from an S7-400 with CP to a redundant IEC master, you can use the same local port number in CP 443-1 (Advanced) for the two unspecific TCP connections.

FETCH/WRITE mode (S7-300, S7-400)

Use of FETCH/WRITE (S7-300, S7-400)

FETCH/WRITE

You can use the FETCH/WRITE services to access system memory areas of the S7 CPU.

- FETCH: Read data directly
- WRITE: Write data directly

Access is possible from the following devices:

- SIMATIC S5
- SIMATIC PC stations
- Non-SIMATIC devices

Connection types

The FETCH/WRITE services can be configured and used in SIMATIC S7 with the following connection types:

- ISO transport connections
- ISO-on-TCP connections
- TCP connections

System memory

You can use FETCH or WRITE to access the following operand ranges in the SIMATIC S7 system memory:

- Data blocks (DB)
  (Note the following limit for DB access: the highest DB no. is 255)
- Bit memory (M)
- Process image input (PII)
- Process image output (PIQ)
- I/O area inputs (PIW, PID, PIB)
- I/O area outputs (PQW, PQD, PQB)
- Counters (C)
- Timers (T)

Link to non-SIMATIC systems

In principle, FETCH and WRITE can be used to access S7 system memory areas from any non-SIEMENS device.

To implement this access for PC applications, for example, you must become acquainted with the PDU structure for the jobs. Refer to the device documentation for this purpose.

Alarms in the diagnostic buffer

FETCH/WRITE access may give rise to negative acknowledgements from the S7 CPU and corresponding connection-specific entries in the diagnostic buffer.

Table 1-4 Alarm coding in the diagnostic buffer at FETCH/WRITE

Coding	Meaning
01_H	Hardware error
03_H	Object access is not allowed.
05_H	Invalid address (syntax ID, range, type, bit number)
06_H	Data type is not supported.
07_H	Data type is inconsistent.
$0A_H$	The object does not exist, or the range limit was exceeded.
FF_H	Internal protocol error

Additional information

The PDU structure needed to access non-SIMATIC systems is described in the documentation for S7 CPs.

Connection configuration for FETCH/WRITE (S7-300, S7-400)

Configuring the operating mode to suit the type of device

The following operating modes can be configured for the communication endpoint, according to the type of station:

- SIMATIC S7 station: FETCH PASSIVE/WRITE PASSIVE
  If you select either FETCH PASSIVE or WRITE PASSIVE mode for the connection, the system memory areas in SIMATIC S7 can be accessed directly from a SIMATIC S5 station or a non-SIMATIC station (unspecified connection).
  The connection can then only be used for this mode. It is then not possible to send or receive using the AG_SEND/AG_LSEND or AG_RECV/AG_LRECV FCs.
  The connection is established passively, which means that only the partner station (SIMATIC S5 station, a PC station, or a non-SIMATIC station) can establish the connection. You make the necessary settings under "Properties > Options". Once you have selected the operating mode, you will not be able to change the connection setup type.
- SIMATIC PC station: FETCH ACTIVE/WRITE ACTIVE
  If you select either FETCH ACTIVE or WRITE ACTIVE mode, the system memory areas in the SIMATIC S7 or SIMATIC S5 station can be accessed directly from the PC station. The connection is established actively, which means that the partner station must await the connection to be established (passive connection setup for the partner). You make the necessary settings under "Properties > Options". Once you have selected the operating mode, you will not be able to change the connection setup type.

"S7 addressing mode" option

You can select the addressing mode when you configure FETCH ACTIVE/WRITE ACTIVE mode. This defines how the address information in the FETCH/WRITE call in the SIMATIC S7 station will be interpreted when DBs are accessed:

- S7 addressing mode: Byte address
- S5 addressing mode: Word address

This allows applications to access S5 or S7 stations without modifying the addresses. This is particularly useful for existing S5 applications, for example, that can now be used unchanged when accessing S7 stations.

The addressing mode for accessing SIMATIC S7 (option selected) is set by default.

FDL connection (S7-300, S7-400)

FDL connections - overview (S7-300, S7-400)

Area of application

Data transmission over configured FDL connection is suitable for transmission of contiguous data blocks between two or more PROFIBUS stations.

There is a distinction between

- Specified FDL connection
  The communication stations are uniquely specified by the connection configuration.
  The connection partner can be inside or outside of the STEP7 project.
- Unspecified FDL connection (free layer 2 access)
  The address of the connection partner remains open in the configuration. The communication stations are determined by address information in the communication job of the user program. This means up to 126 stations can be reached via one configured unspecified FDL connection, if these stations support FDL connections.
  The connection partner can be inside or outside of the project.
- FDL connection with broadcast
  All stations ready to receive broadcasts on the PROFIBUS are reached.
- FDL connection with multicast
  All stations belonging to the multicast group on the PROFIBUS are reached.

Note

Configuration prerequisite

If FDL connections will be used, the CP mode of the PROFIBUS CP **should not be set** to **DP slave passive**.

Note

Priority of the telegrams

Note that the PROFIBUS-CPs for SIMATIC S7 send the telegrams in "LOW" priority.

Partner stations (SIMATIC S5, S7, or third-party stations) must also use "LOW" priority, otherwise a connection willl not be established.

FDL address details (S7-300, S7-400)

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Address parameters for FDL connections

An FDL connection is specified by the local and remote connection endpoint. This includes:

- PROFIBUS address of the node that is to be reached.
- Local LSAP (Link Service Access Point):
  The local LSAP controls the readiness to receive of the PROFIBUS-CP. The receive resources for data receive in the PROFIBUS-CP are provided on the FDL connection for the LSAP.
- Remote LSAP (Link Service Access Point):
  The remote LSAP controls the send mode of the PROFIBUS-CP. The PROFIBUS-CP sends to the nodes on the FDL connection via the LSAP. The target node must be ready to receive for this SAP.

Specified FDL connection

The relevant local and remote address information is displayed as recommended values for a specified FDL connection. You have the option to set the LSAP addresses individually, if necessary.

Unspecified FDL connection

You can use the unspecified FDL connection in two ways:

- Connection to a "third-party station" in a different project.
  You can specify the remote PROFIBUS address and the LSAP for any target station. The target station can be inside or outside of the current STEP 7 project.
  If the remote address is unspecified, then communication via the FDL connection is not possible.
- Free layer-2 access
  Click the relevant check box to configure a free layer 2 access. The input fields for the remote PROFIBUS address and the remote LSAP can no longer be specified in this case because the target addresses are now specified by the user program.

FDL connection with broadcast

The remote address parameters for FDL connection with broadcast are permanently set. All broadcast nodes can be reached via the PROFIBUS address 127. Receive data is received via LSAP 63 from all broadcast nodes.

The local address parameters are entered in the job header of the message for sending and delivered to the recipient. The user program for the remote partner can thus determine the sender of the broadcast frame.

Note

CP receives only on a broadcast connection

If you use an FDL connection with broadcast, you cannot use any other broadcast connection on the CP.

> **CAUTION**
>
> **"Activate cyclic distribution of bus parameters" function**
>
> If you select this option for the subnet in "General > Bus parameters", the bus parameters will be sent cyclically during operation as broadcast frames. To avoid frame handling conflicts in the user program that receives broadcast frames, you should take the following action:
>
> - ignore all frames that are sent with an LSAP >56
> or
> - disable the option under "General > Bus parameters".

FDL connection with multicast

The remote address parameters for FDL connection with multicast are permanently set. All multicast nodes can be reached via the PROFIBUS address 127. Sending and receiving data is handled by all nodes in the multicast group via the same LSAP (range 1 to 56). This means the value of the LSAP can only be selected locally and is automatically transferred to the remote LSAP.

The local address parameters are entered in the job header of the message for sending and delivered to the recipient. The user program for the remote partner can use this information to determine the sender of the multicast frame.

FDL options (S7-300, S7-400)

Relevance

Connection configuration in the parameter group "Properties > General > Options".

Operating mode

- SEND/RECV:
  This is the standard mode with the AG_SEND/AG_LSEND, and AG_RECV/AG_LRECV FCs. This mode is the default on FDL connections and this cannot be modified.

FDL OPC properties

Relevance

Connection configuration for OPC in the parameter group "Properties > General > OPC".

Here, you set connection-specific properties for an FDL connection used by the OPC server.

You can set the following parameters:

Timeout

If a pending job cannot be processed during this wait time, the connection is reset and the job acknowledged with an error.

Max number of send retries

Here, specify how many retries there will be for a job if, for example, the connection partner cannot be reached due to a temporary lack of resources.

Send buffer size

The size of the default send buffer is set here.

This setting applies to both acknowledged and unacknowledged sending.

Additional send buffers of any length (maximum 244 or 246 bytes) can be used at runtime.

Default setting:

244 bytes or 246 bytes (with default SAPs, if the default SAP 255 is set for both the local and remote SAP)

Maximum number of parallel send jobs

Maximum number of simultaneous send jobs on the connection configured here.

Number of resources for received jobs

Specify how many receive jobs the OPC server will be able to receive simultaneously on the FDL connection configured here.

Select a value suitable for the expected amount of traffic; in other words,

- For a low total number of connections: Select a higher value to achieve a fast reaction time.
- For a high total number of connections: Select a lower value to avoid resource conflicts.
  - Default setting: 3

Send with higher priority

Specify whether the FDL frames on the connection configured here are sent via the LAN with priority (option enabled).

FDL unspecified FDL connection (free layer 2 access) (S7-300, S7-400)

Properties

An unspecified FDL connection with open layer 2 access enables program-controlled addressing of the communication partner and communication between two stations on PROFIBUS with the following properties:

- Data transfer is bidirectional, which means you can send and receive data simultaneously on the FDL connection.
- The local station is specified by configuration. The remote node is entered by the user program in the job header of the job buffer with the AG_SEND instruction call. This means that each node on PROFIBUS (PROFIBUS addresses from 0 to 126) can be reached.
- The PB address, the LSAP, and the sender service can be read from the job header of the AG_RECV.

The figure below illustrates program-controlled addressing in the job buffer:

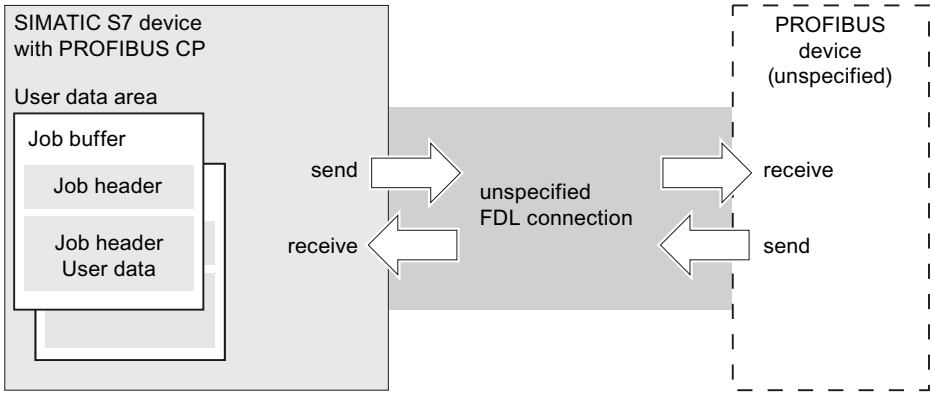

Data volume and configuration limits

Each job buffer can transfer up to 236 bytes of user data. The job header reserves an additional 4 bytes.

FDL connection with broadcast (S7-300, S7-400)

Properties

A broadcast connection sends a message to multiple recipients with **one** job. This means messages can be received on the same broadcast connection that are simultaneously received by other stations on the PROFIBUS.

The properties can be summarized as follows:

- Data transfer is bidirectional, which means you can send and receive data simultaneously on the broadcast connection.
- Sending and receiving takes place via the FDL service SDN (Send Data with No Acknowledge).

- The AG_SEND call needs a job buffer for sending. The job header of this buffer must be reserved.
- The PB address, the LSAP, and service of the broadcast sender can be read from the job header of the AG_RECV.
- The LSAP range from 1 to 56 is reserved for sending. LSAP 63 is reserved for all broadcast stations for receiving.

The figure below illustrates sending and receiving via an FDL connection with program-controlled broadcast addressing:

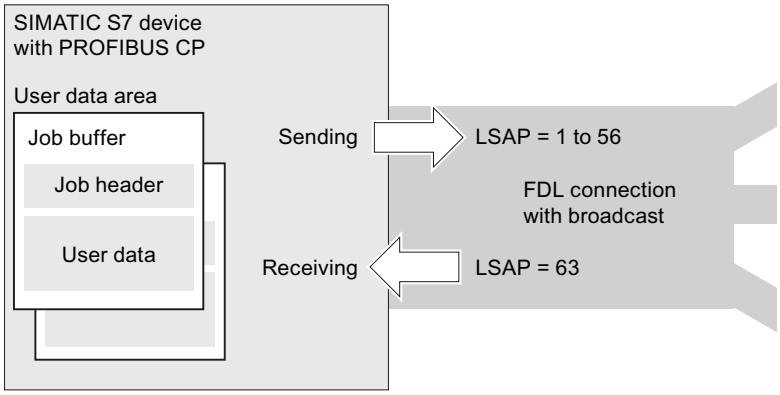

Configuring an FDL connection with broadcast

When creating the FDL connection select "All broadcast stations" as connection partner/station.

See also

Creating UDP / FDL connections with broadcast / multicast (Page 228)

Data volume and configuration limits

PROFIBUS-CP only supports **one** broadcast connection.

Each job buffer can transfer up to 236 bytes of user data. The job header reserves an additional 4 bytes.

Note

CP receives only on a broadcast connection

If you use an FDL connection with broadcast, you cannot receive any additional broadcast connection messages on the respective CP, neither on any FMS connection with broadcast.
Reason:
The receive LSAP for broadcast (63) is reserved with a broadcast connection.

FDL connection with multicast (S7-300, S7-400)

Properties

An FDL connection with multicast sends a message to multiple recipients of a multicast group with one job.

The properties can be summarized as follows:

- Data transfer is bidirectional, which means you can send and receive data simultaneously on the FDL connection with multicast.
- Sending and receiving takes place via the FDL service SDN (Send Data with No Acknowledge).
- The system sends via an LSAP (range from 1 to 56) that is uniform for the multicast group.
- The AG_SEND call needs a job buffer for sending. The job header of this buffer must be reserved.
- The PB address, the LSAP, and service of the multicast sender can be read from the job header of the AG_RECV.

The picture below illustrates sending and receiving via an FDL connection with program-controlled multicast addressing:

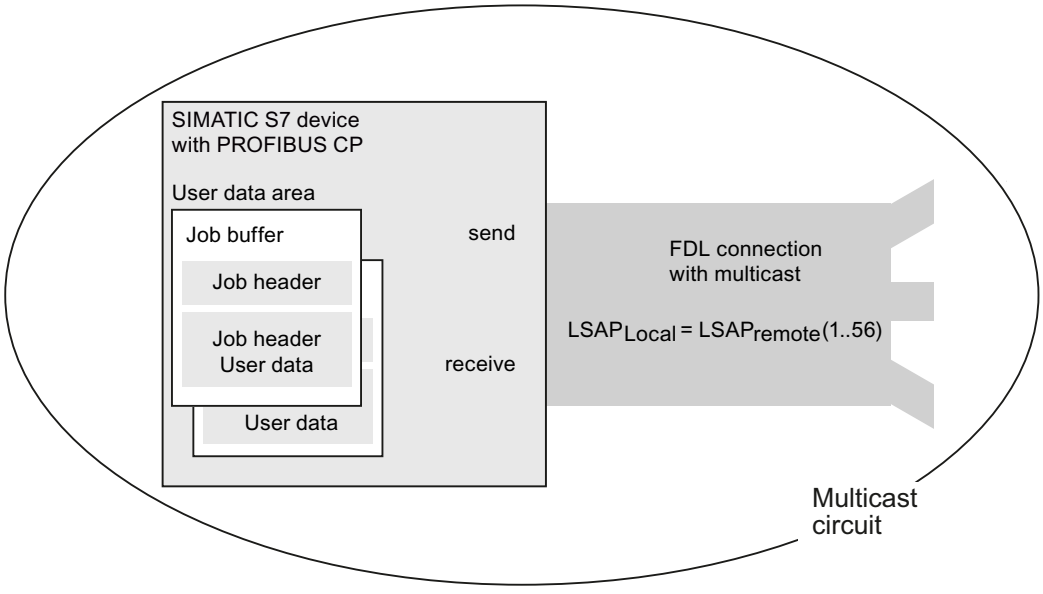

Configuring an FDL connection with multicast

When creating the FDL connection select "All multicast nodes" as connection partner/station.

See also

Creating UDP / FDL connections with broadcast / multicast (Page 228)

Data volume and configuration limits

See the device documentation for PROFIBUS-CP for the number of FDL connections that the respective PROFIBUS-CP supports. The number of connections per station can be increased by accepting additional CPs .

Each job buffer can transfer up to 236 bytes of user data. The job header reserves an additional 4 bytes.

Setting up communication over FDL (S7-300, S7-400)

Requirement

- Configuration software: STEP 7 Professional V14
- End point of the connection: CPU S7-1500 firmware version V2.0 or higher with communication module CM 1542-5 with firmware version V2.0

Setting up a configured FDL connection

Proceed as follows to set up a configured FDL connection in STEP 7:

1. Create a TSEND_C instruction in the program editor.
2. Select the TSEND_C instruction and go to "Properties" > "General" > "Connection parameters" in the Inspector window.
3. Under End point, select the partner end point. Use one of the two partner end points below:
   - CPU S7-1500 with CM 1542-5
   - Unspecified
4. Select "Use configured connection" under Configuration type.
5. Select "FDL" under Connection type.
6. Select the following interfaces under Interfaces:
   - Local: PROFIBUS interface of CM 1542-5
   - Specified partner: PROFIBUS interface of CM 1542-5
7. Select the setting <new> under Connection data.

The figure below shows a fully configured FDL connection in STEP 7.

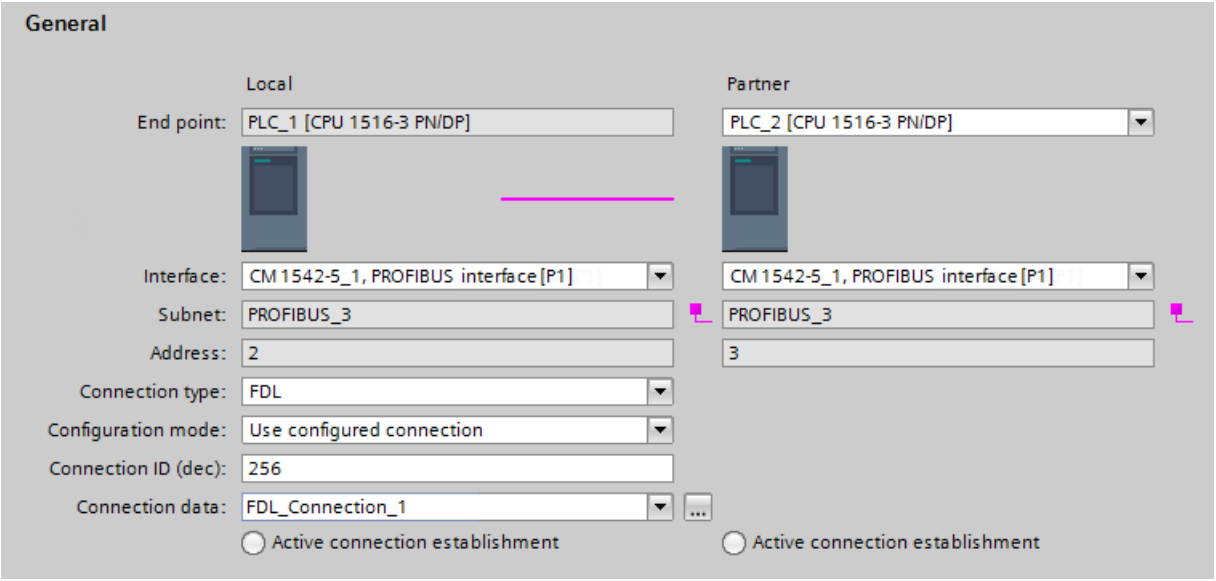

Figure 1-9 Configuring the FDL connection

Setting up an FDL connection in the user program

For communication via FDL, you need to create the data block of the TCON_FDL system data type yourself in each case, assign parameters and call it directly at the instruction. Follow these steps:

1. Create a global data block in the project tree.
2. In the global data block, define a tag of the data type TCON_FDL.
   The example below shows the global data block "FDL_connection" in which the tag "FDL_connection" of the data type TCON_FDL is defined.

FDL_connection

	Name	Data type	Start value	R...	A...	W...	V...	Comment
1	Static							
2	FDL_connection	TCON_FDL			✓		✓	
3	InterfaceId	HW_ANY	0		✓		✓	HW identifier of PB interface submodule
4	ID	CONN_OUC	16#0		✓		✓	connection reference / identifier
5	ConnectionType	Byte	16#15		✓		✓	type of connection: 21= FDL connection
6	ActiveEstablished	Bool	false		✓		✓	active/passive connection establishment
7	ServiceId	Byte	16#0		✓		✓	service id: 0 – default, 1 – SDA, 2 – SDN
8	RemotePBAddress	Byte	16#0		✓		✓	remote ProfiBus partner address
9	LocalPBAddress	Byte	16#0		✓		✓	local ProfiBus partner address
10	RemoteLSAP	Byte	16#0		✓		✓	remote PB link-layer service access point
11	LocalLSAP	Byte	16#0		✓		✓	local PB link-layer service access point

Figure 1-10 Programming an FDL connection

3. Program the parameters of the FDL connection (e.g. the PROFIBUS addresses) in the tag of the data type TCON_FDL.

4. Create a TCON instruction in the program editor.
5. Interconnect the CONNECT parameter of the TCON instruction with the tag of the data type TCON_FDL.
   In the example below, the CONNECT parameter of the TCON instruction is interconnected with the tag "FDL_Connection" (data type TCON_FDL).

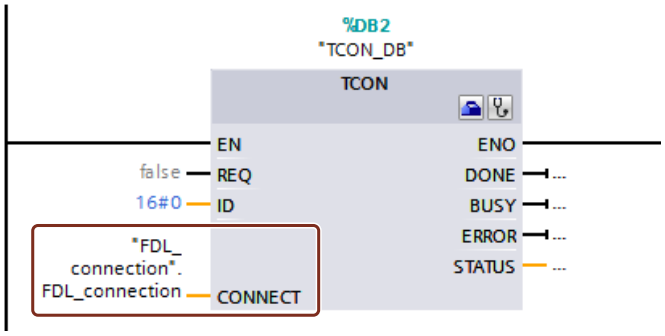

Figure 1-11 Example: TCON instruction for FDL connection

E-mail connection (S7-300, S7-400, S7-1500)

Setting up an email connection (S7-300, S7-400, S7-1500)

Emails can be sent from the automation system, but they cannot be received.

To send email in the user program of the S7-CPU, use the send call of the SEND/RECEIVE interface (FC AG_SEND / AG_LSEND).

Configuration overview

Set up exactly one e-mail connection per CP for sending e-mails. This e-mail connection specifies the mail server that delivers all e-mails sent by the CP.

An email connection can be set up as follows:

- Connection configuration (standard application)
  This application is described below.
- User program with FB CP_CONFIG and configuration data block
  There are application ranges for which it is sensible to set up the communication connections with a program-controlled procedure via specific applications rather than via the configuration interface.

Procedure - example of graphic configuration

Configuration of an email connection is described below. The procedure is basically the same as when setting up an unspecified connection: Creating an unspecified connection (Page 226).

Select the connection type, email connection, in the drop-down list box.

This step activates the connection mode: The CPUs available for the selected connection type are selected and highlighted.

1. Click on the CPU from which a connection will be set up.
2. Move the cursor within the CPU and confirm the connection endpoint with another click of the mouse.
   An unspecified connection will be created.
3. In "Properties > Address details" in the Inspector window, enter the address parameters according to the information in the address details:
   E-mail address details (Page 276)

After downloading the configuration data the user program can send e-mails over this e-mail connection.

Using an OPC server as an SMTP server

Instead of an unspecified connection, you can select an OPC server as an SMTP server as the connection partner. This means that the IP address of the SMTP interface of the PC station it is entered in the partner's address details.

Note

The PC interface must support the SMTP protocol

The e-mail connection via the OPC server is only created consistently when an interface with SMTP protocol support is enabled on the PC station. Read the documentation of the module your using.

See also

DNS configuration (Page 3063)

E-mail address details (S7-300, S7-400, S7-1500)

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Parameter

Enter the parameters required for the delivery of the e-mail. You can set the following parameters:

Parameters	Description	Example
E-mail server (SMTP)	Address of the mail server via which the e-mails are sent. The IP address can be specified absolutely or symbolically. The following applies to symbolic names: A valid name has 1 to 64 characters of which at least one character must be a letter. If one or more "." characters are used, the following constraint also applies: A valid name has 1 to 3 characters behind the last "."; at least one of these characters must be a letter. A symbolic name can only be used if the Internet CP knows the address of the Domain Name System (DNS). Make an appropriate entry when configuring the Internet CP in the DNS configuration. A maximum of 64 characters can be entered.	absolute: 140.80.0.4 symbolic name: t-online.de
Default sender name	Specification of an address that is always inserted in the email as sender address, if the sender specification (FROM parameter) is empty in the header of the email. This information is normally irrelevant for delivery of the mail. It is simply information for the mail recipient. Since some mail servers do not forward jobs if there is no sender information, it is advisable to enter information here! A maximum of 126 characters can be entered. **Note:** Remember that a default sender name must be specified if you want a test mail to be initiated by special diagnostics.	Station2.CPU412@xy.facto-ry2.de

See also

DNS configuration (Page 3063)

Sending email (S7-300, S7-400, S7-1500)

Procedure

Proceed as follows to send an e-mail:

1. Make the e-mail data available in a data block.
2. Use the AG_SEND or AG_LSEND instruction in the user program.

Requirement

You can send email if the email connection has been set up via the connection configuration. You must use the specified ID to call the FC AG_SEND/AG_LSEND for the connection configuration.

Data block

The entire email, meaning the address information and the message itself, will be set up in any data block. The example in STL notation below shows the appropriate information for the required DB structure.

Use the programming editor to create and enter the DB data.

The following table contains information about email data blocks in STL notation:

Address	Name	Type	Start value	Comment	Entry
0.0		STRUCT			
+0.0	TO[1)]	STRING[40]	"TO:name.name@t-online.de;"	Recipient	Mandatory
+42.0	CC[1)]	STRING[40]	"CC:name.name@t-online.de;"	CC recipient	Optional
+84.0	FROM	STRING[40]	"FROM:plant.werk2@xyz-on-line.de;"	Sender	Optional
+126.0	SUB	STRING[40]	'SUB:Status Station 7;'	Subject	Optional
+168.0	Text	STRING[100]	'TXT:Fault in plant section 2;'	Mail text	Mandatory
+270.0	Attach-ment	STRING[4]	'BNY:'	Here the attachment is initiated[3)]	Optional
+276.0	Value1	BYTE	**B#16#27**[2)]	Attachment/binary val-ue[3)]	Optional
+277.0	Value2	BYTE	**B#16#03**[2)]	Attachment/binary val-ue[3)]	Optional
=278.0		END_STRUCT			

1) Multiple recipients can be specified. The information must then be separated by a comma.

2) The information in bold will be delivered to the recipient as attachment

3) Data can also be supplied dynamically.

Comments on the table

- Structure and syntax of the data in the email DB
  The structure suggested here with multiple STRINGs is one of several variants. Crucial are the entries in the column "Initial value" with the IDs contained therein (TO:, SUB:, CC:, FROM:, TXT:, BNY:) which must be used in the DB in precisely this notation to identify the mail content! All entries must end with a semicolon; only the last entry does not need a semicolon.
  The string length is only shown in the table as an example; it can be customized to the actual character count (exception: The string length for identifying the attachment must be specified with [4]).
  For example, another variant would be using only one STRING in total, and assigning the entire text to this string with the identifiers.
- If you encounter a problem entering the "@" symbol, try the input ALT+64 instead.

- Attachments
  The user data entered in the email DB can also be delivered to the recipient entirely or partially as attachment. For this the sender must give the data the identifier "BNY:" .
  The data specified after this identifier will then be delivered to the recipient as attachment. In the table the attachment is 2 bytes; this is only an example however. You can enter attachments that are as complex as you like.
- Data length
  The data length specified in the call AG_SEND/AG_LSEND must at least include the length of the data in the DB; note the specifications in the address column of the programming editor (Note: The specification corresponds to the number of bytes).

Sending an email with AG_SEND/AG_LSEND [1)]

Use the FC AG_SEND instruction to send an e-mail or with data lengths >240 bytes, use AG_LSEND. A detailed description of call parameters is available in the online help for the instructions.

See also AUTOHOTSPOT

Example:

STL	Explanation
call fc 50	//AG_LSEND block call
ACT := M 10.0	//Bit for job trigger
ID := MW 12	//Connection ID (connection configuration)
LADDR := W#16#0100	//Module address 256Dec. in hardware configuration
SEND := P#db99.dbx10.0 byte 278,	//Address of the data block; DB length
LEN := MW 14	//Length of the data range to be sent
DONE := M 10.6	//Address for return parameter DONE
ERROR := M 10.7	//Address for return parameter ERROR
STATUS := MW 16	//Address for return parameter STATUS

Point-to-point link (S7-300, S7-400, S7-1500)

Point-to-point communication (S7-300, S7-400, S7-1500)

Types of communication over PtP links

When communicating over a PtP connection, two different types of communication are distinguished:

- Homogeneous communication
- Heterogeneous communication

Support of PtP communication means that external devices such as printers can be accessed.

Homogeneous PtP communication

The PtP CP forwards data between modules that support the S7 protocol. Users have the full range of communications functions available if these are provided by the end devices connected via PtP (for example CPU with CPU or PG with CPU).

On these connections, the PtP CPs are not the endpoint of the connection but take over the role of a router.

With a homogeneous link, all the communications functions defined in S7 can be used if they are supported by both connection partners. Protocol handling for specific communication instruction calls is therefore not necessary on the CP.

Heterogeneous PtP communication

The PtP CP operates like a gateway using other protocols such as RK512, 3964, open drivers and special protocols. Selected blocks serve as a user interface synchronized with the user program. At the receiving end, the user program acting as server can be accessed via the heterogeneous PtP link once again using communications functions synchronized with the program.

From a communications engineering perspective, the PtP CP is the endpoint of the S7 connection in heterogeneous communication. Communication between a CPU and CP is therefore based on S7 protocol elements. The CP handles the conversion of these protocol elements to the PtP protocol and vice versa.

For heterogeneous communication, the following communication instructions can be used on the CPU:

- BSEND
- BRCV
- PUT
- GET

On the PtP CP, the suitable S7 protocol elements must be implemented for the BSEND, BRCV, PUT and GET instructions as client (CP is active) and PUT and GET instructions as server (CP is passive).

Instructions	Client	Server
BSEND	Yes	no
BRCV	Yes	no
PUT	Yes	Yes
GET	Yes	Yes

Communications functions

Heterogeneous communication can have two different qualities depending on the application:

- Controlled communications functions:
  With controlled communications functions, it is possible to control the transfer. This means that the transfer can not only be started at the active end but can be enabled at the passive end by the control and the end of the transfer can be detected.
  This functionality is implemented for the communications instructions in S7 with the parameters "EN_R" (ready to receive) and "NDR" (receipt of data complete).
- Non-controlled communications functions:
  With non-controlled communications functions, the transfer is handled by the active partner without the passive partner being able to take any action.

Assigning parameters for a point-to-point link (S7-300, S7-400, S7-1500)

Hardware for configuring PtP

To establish a PtP connection, you require a CP 441. Communications modules of the type CP 441 have slots allowing the use of one (CP 441-1) or two (CP 441-2) PtP interface modules.

You will find PtP interface modules of the type IF963 in the hardware catalog of the SIMATIC S7-400 under the communications modules for PtP. The IF963 modules are available for the following interface types:

- RS-232C
- TTY
- RS-422/485

Depending on the interface type, the IF963 modules support at most the following communications protocols:

- ASCII
- 3964(R)
- RK512
- Printers
- Modbus master
- Modbus slave

If you use a PtP interface in a CP 441, a PtP subnet is automatically displayed and an unspecified PtP connection is added.

Once you have selected the CP 441 or selected the required interface directly, you can select and configure the protocol in the Inspector window.

PtP connection to an S7 device as the connection partner

You can enter a maximum of 8 connections in the connection table for the local partner. For the connection partner, you enter only **one** connection in the connection table. The connection partner can receive data from the 8 connections via this one connection.

Example: Settings for the local partner:

- 3964(R) protocol
- Communication direction 1: Local --> partner
- max. 8 connections, local IDs: 1000; 1001, ..., 1007

Settings for the connection partner:

- 3964(R) protocol
- Communication direction 2: Partner --> local
- max. 1 connection, local ID: 1000

Number of connections

The maximum number of configurable connections per device (CPU/FM) depends on the resources of the module you are using.

PtP communication direction (S7-300, S7-400, S7-1500)

Communications direction

For PtP connections, you can specify different communications directions. To do this, select the corresponding check box:

- Local → partner:
  Connections via which you want to send frames.
- Partner → local:
  Connections via which you want to receive frames.
- Local ↔ partner:
  Connections via which you want to send and receive frames.

Note

You will find more detailed information on specifying the communications direction in the documentation of your module.

PtP link path (S7-300, S7-400, S7-1500)

Setting the connection path

In "General > Connection path". you can set the path taken by the connection from the local endpoint to the connection partner in the settings relating to the communications direction. You can select the path depending on the configured PtP CPs. Example: The local partner contains two PtP CPs. In this case, you can select the CP via which the connection will run.

Special feature: In contrast to the S7 connections, when configuring PtP connections to an unspecified partner, networking of the local partner is not essential. The communications

partners only need to be networked in your real system before you put the connection into operation.

Selection options for the connection partners

- Via CP:
  For the local partner, you can select the PtP CP via which the connection will run (name of the CP along with the rack and slot).
- Interface:
  SIMATIC S7 CPs have several channels (interfaces) via which the PtP connections can be established. The channel and the protocol used for the channel are displayed (channel/protocol). You configured the protocol with the special configuration software of the CP.
- Type:
  The Interface type " PtP" is displayed here.

"Connection selected via RK512 CPU no.:" (local)

Automation systems capable of multicomputing (for example SIMATIC S7-400) can contain several CPUs. For this reason, you need to specify a CPU number with which the partner can access the connection.

If you have selected **Partner -> Local** or **Local <-> Partner** as the communications direction and want to receive frames from the partner using an SFB, enter the number of the CPU (1 to 4) of the **local partner** here.

The CPU number specified here must match the CPU number entered in the command frame header of the CP (for the structure of the command frame, refer to the documentation of the CP you are using). If they match, the partner accesses the connection using this CPU number.

Note

If you want to send and receive frames using SFBs, complete the two boxes below "Local" and "Partner". You will find the maximum number of connections over which you can send/receive frames in the documentation of the CP you are using.

"RK512 CPU no.:" (partner)

Automation systems capable of multicomputing (for example SIMATIC S7-400) can contain several CPUs. For this reason, you need to specify a CPU number to which the connection will go.

If you have selected **Local -> Partner** or **Local <-> Partner** as the communications direction and want to send frames to the partner using an SFB, enter the number of the CPU (1 to 4) of the **connection partner** here.

The CP enters the CPU no. you have specified in the command frame header of the frame to be sent. Using this CPU number, one of the 4 CPUs of the connection partner will be accessed.

When transferring to third-party devices, it is also possible to specify "0" as the CPU no. This makes broadcast access possible on the partner station. If a CP 441 is the partner, CPU 1 is then always accessed.

Note

If you want to send and receive frames using SFBs, complete the two boxes below "Local" and "Partner". You will find the maximum number of connections over which you can send/receive frames in the documentation of the CP you are using.

PtP - special connection properties (S7-300, S7-400, S7-1500)

Relevance

You can specify additional properties of the local connection endpoint in the "Properties > General > Special connection properties" parameter group.

Configured at one end

Configured at one end means that the connection partner functions as a server on this connection and cannot send or receive actively.

Establishing an active connection

Use this option to specify whether the connection is established from this device. This is the default option if the address of the partner is specified.

- ON: Connection is established actively.
- OFF: Connection is established by the partner.

If you selected "Unspecified" as the connection partner when you created the connection, the option is deselected by default. If you select the option you must specify the address of the partner in the "Addresses" tab.

Note

Not every node is capable of handling the connection establishment. In this case, the partner must initiate the establishment; the check boxes are grayed out and cannot be changed.

Send status messages

The connection partners can also exchange their status messages via an existing connection.

Select the check box if you want the local device to send its status messages to the partner. The status messages can then be received on the partner using the USTATUS instruction (SFB 23).

1.3.3 Displaying and configuring topology

1.3.3.1 Overview of the topology view

Functions of the topology view

The topology view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Displaying the Ethernet topology
  - Displaying all the PROFINET devices and passive Ethernet components of the project along with their ports
  - Displaying interconnections between the ports
  - Displaying corresponding logical networks
  - Display diagnostic information of all ports
- Configuring the Ethernet topology
  - Creating, modifying and deleting interconnections of the ports
  - Renaming stations, devices, interfaces or ports
  - Adding PROFINET devices and passive Ethernet components to the project from the hardware catalog
- Identifying and minimizing differences between the desired and actual topology
  - Running an offline/online comparison of Ethernet modules, ports and port interconnections
  - Adopting existing online topology information in the offline project

---

Note

Devices without valid IP address

Topology information (LLDP) cannot be read from a device without valid IP address.

To prevent devices from having an invalid IP address, specify in the settings of the TIA Portal for hardware configuration that devices without valid IP address are temporarily assigned an IP address.

Note

Subnet membership

Before the topology determination, you have to set for the local interface of your PG/PC the network address that the Ethernet components that are to be determined also have in the real plant. If you do not meet this requirement, no further topology information (information about ports and neighborhood relationships) can be determined.

Differences between network view and topology view

- The network view shows all the logical subnets of the project.
  The topology view shows all Ethernet components of the project. These include passive components such as switches, media converters and cables.

  Note

  Stations with non-Ethernet components are also displayed if the station has a least one Ethernet component.

- The position of a device in the network view and its position in the topology view are initially independent of one other; in other words, the same device generally appears at different locations in the two views.
  If you want to align the devices in the topology view as in the network view, click on the following icon in the toolbar of the topology view.

Icon	Meaning
	Changes the position of the devices in the topology view so that the new position is as close as possible to the position in the network view.

- If you open the hardware catalog in the topology view, you only see devices with an Ethernet interface.

Structure of the topology view

The topology view (Page 20) essentially consists of a graphic area (called the graphic view below) and a table area (called the table view below).

Which functions are there in the graphic view and which functions are there in the table view?

- Displaying the Ethernet topology

Function	Graphic view	Table view
Displaying all the PROFINET devices and passive Ethernet components of the project along with their ports	yes	yes
Display interconnections between the ports (including type of medium)	yes	yes
Displaying corresponding logical networks	no	yes
Displaying properties of the cables between the ports	no	yes
Display diagnostic information of all ports	yes	yes

- Configuring the Ethernet topology

Function	Graphic view	Table view
Creating, modifying and deleting interconnections of the ports	• Create: yes • Modify: yes • Delete: yes	• Create: yes • Modify: yes • Delete: yes
Renaming stations, devices, interfaces or ports	no	yes
Adding PROFINET devices and passive Ethernet components to the project from the hardware catalog	yes	no

- Identifying and minimizing differences between the desired and actual topology

Function	Graphic view	Table view
Running an offline/online comparison of Ethernet modules, ports and port interconnections	no	yes
Adopting existing online topology information in the offline project	no	yes

Structure of the table view

The table view consists of the following two tabs:

- "Topology overview" tab
- "Topology comparison" tab

The "Topology overview" tab provides the following functions:

- Display of the configured topology with hierarchical structures (from the station to the port)
- Configuration of the Ethernet topology
- The display of the diagnostics status of hardware components

The "Topology comparison" tab provides the following functions:

- Identifying and minimizing differences between the desired and actual topology

1.3.3.2 Starting the topology view

Requirement

The device or network view is open in the hardware and network editor.

Procedure

To start the topology view of your project, follow these steps:

1. Click on the "Topology view" tab.

Or:

1. Open the network view of the hardware editor.
2. Select a PROFINET device or a PROFINET module.
3. Select the "Go to topology view" command in the shortcut menu.

Result

The topology view is started. If you opened the topology view using the shortcut menu, the selected component remains selected after the change of view.

1.3.3.3 Displaying topology

Displaying the graphic view of the configured topology

What is shown?

The graphic view of the configured topology shows the following:

- Configured PROFINET devices and passive Ethernet components along with their ports
- Configured stations with non-Ethernet components if there is at least one Ethernet component in the station
- Configured interconnections between the ports

Type of display

The way in which the graphic view of the topology view and the network view are displayed is very similar:

- Compared with the device view, components are shown in a simplified form.
- The interconnections between ports are shown as horizontal and vertical lines. These are dashed when an interconnection between a tool changer port and its possible partner ports is involved.

Displaying the table view of the configured topology

Requirement

- The table view of the topology is open.
- You are in the "Topology overview" tab

What is shown?

The table view of the configured topology shows exactly the same content as the graphic view except for the logical PROFINET subnets:

- All the configured PROFINET devices and passive Ethernet components along with their ports
- All the configured stations with non-Ethernet components if there is at least one Ethernet component in the station
- Configured interconnections between the ports
  For each port with the "Alternating partner port" property, there are as many completed rows as there are potential partner ports plus one empty row.

Type of display

As the name implies, the table view of the topology view consists of a table, the topology overview table. It displays the configured topology with hierarchical structures (from the station to the port) It is structured like the network overview table. It consists of the following columns:

- Device / port
  This is the most important column of the table. The entries in this column have a hierarchical structure with the PROFINET ports being the last element in the hierarchy. You can expand and collapse the hierarchical entries. For a CPU, for example, an entry consists of the following elements:
  - Station name
  - Device name
  - Name of the PROFINET interface
  - Names of the ports

  Note: All the other columns only have entries in the rows containing the port names.
- Slot
  Indicates the number of the slot in a rack in which the relevant module is installed. For interfaces, the designation of the interface is also displayed.
- Index (by default, this column is not displayed)
  This column is only relevant for PC stations. The index is the virtual slot address of the component.
- Device type (by default, this column is not displayed)
  Shows what type of station, device or interface the table row relates to or whether it belongs to a port.
- Article no. (by default, this column is not displayed)
  Article no. of the device
- Subnet (as default, this column is not displayed)
  Configured subnet to which the interface belongs
- Master/IO system (as default, this column is not displayed)
  Shows whether or not the interface belongs to a PROFIBUS DP master system or a PROFINET IO system.

- IP address (by default, this column is not displayed)
  Configured address of the interface in the subnet
- Partner station
  Name of the station that contains the partner port
- Partner device
  Name of the device that contains the partner port
- Partner interface
  Interface to which the partner port belongs
- Partner port
- Cable data
  Contains the cable length and the signal delay of the cable connecting the ports

Basic functions for tables

The topology overview table supports the following basic functions for editing a table:

- Displaying and hiding table columns
  Note: The columns relevant for configuration cannot be hidden.
- Optimizing column width
- Saving current representation
- Displaying the meaning of a column, a row or cell using tooltips.

Displaying the diagnostics status of ports and cables in the graphic view

Requirements

The graphic view of the topology view is open.

Procedure

To determine the diagnostics status of the port, follow these steps:

1. Go online with the required component or components.

Result

The following icons are displayed:

- The corresponding diagnostics icon is displayed for each device.
- If there is an error in at least one lower-level component, the diagnostics icon "Error in lower-level component" is also displayed in the left-hand lower corner of the diagnostics icon.
- The corresponding diagnostics icon is displayed for each port.
- Every cable between two ports that are online has the color that matches its diagnostics status.

You will find the possible diagnostics icons for ports and the color coding of Ethernet cables in the description of hardware diagnostics. See: Displaying diagnostics status and comparison status using icons (Page 3305)

Showing the diagnostics status of hardware components in the table view

Requirement

- The table view of the topology view is open.
- You are in the "Topology overview" tab

Procedure

To obtain the diagnostics status of hardware components of the topology overview table, follow these steps:

1. Go online with the required components.

Result

The following icons are displayed at the left-hand edge of the topology overview table in each row that belongs to the component involved:

- The diagnostics icon belonging to the hardware component is displayed.
- If the hardware component has lower-level components and if there is an error in at least one of the lower-level components, the diagnostics icon "Error in lower-level component" is also displayed in the left-hand lower corner of the diagnostics icon of the hardware component.

For the possible diagnostics icons for hardware components, refer to the description of hardware diagnostics. See: Displaying diagnostics status and comparison status using icons (Page 3305)

Note

The display of the diagnostics status of hardware components in the topology overview table and the network overview table is identical.

Running an offline/online comparison and displaying the results

Requirements

- The table view of the topology view is open.
- You are in the "Topology comparison" tab
- There may be an online connection to one or more devices, but this is not actually necessary.

Procedure

To find the differences between the configured and the actual topology, follow these steps:

1. Click the "Compare offline/online" button in the toolbar.

Result

The actual topology is identified and compared with the configured topology.

Devices identified online are automatically assigned to configured devices as far as this is possible. See Automatic assignment of devices by offline/online comparison (Page 300)

The columns on the right-hand side of the table are filled or updated:

- Columns for the result of the offline/online comparison: "Status" and "Action"
- Columns for the topology identified online

Note

Stopping a running online/offline comparison

If you click the "Compare offline/online" button while topology discovery is active, the current topology discovery is stopped.

The following buttons are enabled in the toolbar of the table:

Button	Name	Meaning
	Update	The detection of the existing online topology is started again.
	Synchronize	• Adopting the port interconnections identified online in the project (Page 301) • Adopt the devices identified online in the project (Page 301)

Note

If you have not configured your own programming device or your own PC offline, the gray filled circle is displayed as a diagnostic symbol.

You can also configure a second and third programming device or a second and third PC offline.

Columns for the result of the offline/online comparison

The following columns are displayed:

- "Status"
  The result of the offline/online comparison is shown here in the form of diagnostics icons. The following icons are possible:

Diagnostics icon	Meaning
	Differing topology information in at least one lower-level component
	Identical topology information
	Topology information only available offline or device is disabled
	Topology information only exists online
	Differing topology information
	No port interconnection configured, no port interconnection discovered online
	If a device does not support topology functions, the "Status" column remains empty.

- "Action"
  The possible actions are shown here. The following actions are possible:
  - No action
  - Apply

Note

Separating columns

To separate the offline and online columns of the topology comparison table optically, there is a separating column to the left of the "Status" column and to the right of the "Action" column. The separating columns cannot be hidden and their width cannot be changed.

Columns for the topology identified online

The following columns are displayed:

- "PROFINET device name"
- "Article no." (by default, this column is not displayed)
- "Device family" (by default, this column is not displayed)
- "IP address" (by default, this column is not displayed)
- "Port"
- "Interconnection"
  - Green line if the port is interconnected
  - Gray line if the port is not interconnected
- "Partner port"
- "Partner IP address"
- "Partner PROFINET device name"

- "Partner article number" (by default, this column is not displayed)
- "Partner device family" (by default, this column is not displayed)

Basic functions for tables

The topology comparison table supports the following basic functions for editing a table:

- Displaying and hiding table columns
  Note: The columns relevant for configuration cannot be hidden.
- Optimizing column width
- Saving current representation
- Displaying the meaning of a column, a row or cell using tooltips.

Running an advanced offline/online comparison and displaying the results

Overview

With the standard offline/online comparison, a search method based on the DCP protocol is used.

With the advanced offline/online comparison, ICMP is also used to be able to detect devices that do not support DCP. These include, for example, PG/PC stations.

Requirement

- The table view of the topology view is open.
- You are in the "Topology comparison" tab
- The devices to be detected with ICMP are located in the same subnet as the network adapter of your PG/PC you are currently using.

Procedure

Follow these steps:

1. In the toolbar, click on the arrow in the "Advanced comparison" button.
2. With the input boxes "Start address" and "End address", select an IP address range in which accessible Ethernet nodes will be searched for with ICMP. Confirm this, for example by clicking the button with the green check mark.
3. Click on the icon in the left-hand part of the "Advanced comparison" button.

Result

The Ethernet nodes detectable with DCP and ICMP are discovered.

The columns on the right-hand side of the table are filled or updated:

1.3.3.4 Configuring topology

Interconnecting ports

Overview

Interconnecting ports in the topology view

In the topology view, you have the following options for interconnecting ports:

- in the graphic view (Page 296)
- in the graphic view of a tool changer (Page 298)
- in the table view (Page 297)
- in the table view of a tool changer (Page 298)
- by adopting port interconnections identified online (Page 301)

Note

Interconnecting an electric with an optical port

If you want to interconnect an electric and an optical port, you have to decide between RT and IRT communication:

- With RT communication, it is not necessary to configure a media converter.
- With IRT communication, you have to make the interconnection via a media converter.

What effects does the interconnection of ports have on the network view?

Note

In the properties of a subnet in the network view, you can specify that this subnet is used when a port interconnection is created between two devices that are not networked.

When you create an interconnection between two ports, the following effects are possible in the network view:

- If the corresponding interfaces are disconnected: If you have specified a default subnet, this is used. Otherwise a new subnet is created to connect the two interfaces.
- If one (and only one) of the two interfaces involved is networked: The non-networked interface is connected to the same subnet as the already networked interface.
- In all other cases: The corresponding interfaces are not connected to a logical subnet.

See also

Interconnecting ports (Page 2865)

Settings for interconnecting Ethernet devices (Page 87)

Interconnecting ports in the graphic view

Requirement

You are in the graphic view of the topology view.

Procedure – Creating a new interconnection between two ports

To interconnect a port of a device with a port of another device, follow these steps:

1. Place the mouse cursor on the port you want to interconnect.
2. Click with the left mouse button and hold it down.
3. Move the mouse pointer.
   The pointer now shows the networking symbol to indicate "Interconnect" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear when the pointer is on a valid target.
4. Now drag the mouse cursor to the target port. You can do this while holding down or after releasing the mouse button.
5. Now release the left mouse button or press it again (depending on your previous action).

Result: A new port interconnection is created.

Note

Creating a ring for S7-300, S7-400, and S7-1500 CPUs

If you create a ring using port interconnections for S7-300, S7-400, or S7-1500 CPUs, an MRP domain is created automatically.

Procedure – Changing an existing port interconnection without deleting it first

To do this, follow these steps:

1. Place the mouse cursor on the port of an existing interconnection that is to receive a new partner port.
2. Drag the port to the new partner port.

Result: The existing port interconnection is deleted. The new port interconnection is created.

Alternative procedure:

1. Place the mouse cursor on a port without an interconnection that is to be connected with an already interconnected port.
2. Drag the port to the already interconnected port.

Result: The existing port interconnection is deleted. The new port interconnection is created.

Procedure – Interconnecting two interconnected ports with each other without first deleting the two existing port interconnections

To do this, follow these steps:

1. Place the mouse cursor on the interconnected port that is to receive a new partner port.
2. Drag the port to the new partner port that has also already been interconnected.

Result: The two existing port interconnections are deleted. The new port interconnection is created.

Interconnecting ports in the table view

Which actions are possible with port interconnections in the table view?

The following actions are possible with port interconnections in the table view:

- Creating a new port interconnection
- Changing an existing port interconnection
- Deleting an existing port interconnection

Requirement

The row with the port whose interconnection you want to create, modify or delete is visible in the "Topology overview" tab.

Procedure

To create the interconnection of a port for the first time, to modify it or delete it, follow these steps:

1. Move the mouse pointer to the "Partner port" column in the row of the source port.
2. Click the drop-down list there.
3. Select the required partner port (when creating or changing a port interconnection) or the "Any partner" entry (when deleting a port interconnection).

Result

The required action is performed. The new partner port (after creating or modifying a port interconnection) or the "Select port" entry (after deleting a port interconnection) is displayed in the "Partner port" column.

Interconnecting a port with more than one partner port in the graphic view

Requirement

- You have configured a port of a PROFINET device with the "Alternative partners" property and have specified its possible partner ports.
- The graphic view of the topology view is open.

Procedure

1. Interconnect this port (referred to hereafter as source port) with one of the partner ports you have specified (referred to hereafter as target port).
2. Interconnect the source port with an additional target port.
   You can do this in several ways:
   – Drag the mouse pointer from a partner port that is already interconnected to a target port.
   – Drag the mouse pointer from an interconnection that has already been created to a target port.
   – Drag the mouse pointer from a target port to a partner port that is already interconnected.
   – Drag the mouse pointer from a target port to an already created interconnection.
3. If necessary, repeat the step above one or more times.

Result

An interconnection is created between the source port and the alternative partner ports. This is indicated by a dashed line.

Interconnecting a port with more than one partner port in the table view

Which actions are possible with port interconnections to several partner ports in the table view?

When working with a tool changer, the following actions can be performed with port interconnections to multiple partner ports in the table view:

- Creating a new port interconnection
- Changing an existing port interconnection
- Deleting an existing port interconnection

Requirement

- You have configured a port of a PROFINET device with the "Alternative partners" property and have specified its possible partner ports.
- The row with the port whose interconnection you want to create, modify or delete is visible in the "Topology overview" tab.

Procedure

To create the interconnection of a port to one or more partner ports for the first time, to modify it, or to delete it, follow these steps:

1. Move the mouse pointer to the "Partner port" column in the row of the source port.
2. Click the drop-down list there.
3. Select the required partner port (when creating or changing a port interconnection) or the "Any partner" entry (when deleting a port interconnection).

Result

The required action is performed:

- If you are creating an interconnection, a new row is inserted in the topology overview table. The new partner port is displayed there in the "Partner port" column.
- If you change an interconnection, the new partner port is displayed in the "Partner port" column.
- If you delete an interconnection, the row with the previous port interconnection is deleted.

Note

With a tool changer, there are normally several rows for a port with port interconnections to more than one partner port. The last row is always an empty row. The first row can be edited, all other rows are read-only.

Renaming stations, devices, interfaces or ports

Rename a station, a device, an interface or a port

Requirement

- The table view of the topology view is open.
- You are in the "Topology overview" tab

Procedure

To rename a station, a device, an interface or a port, proceed as follows:

1. Click twice in the relevant field of the topology overview table (the second click starts the editing mode).
2. Enter the new name and then press the ENTER key (this closes editing mode).

Result

The object is renamed.

Offline/online comparison

Automatic assignment of devices by offline/online comparison

Overview

During the offline/online comparison, the configured topology is compared with the actual existing topology. Devices identified online are automatically assigned to configured devices as far as this is possible.

Start of availability detection

You start the availability detection the first time by clicking the "Compare offline/online" button in the toolbar of the "Topology comparison" tab.

You restart availability detection by clicking the "Update" button.

Note

The availability detection can take several seconds. During this time, no user input is possible.

Automatic assignment of a PNIO device

A PNIO device identified online is automatically assigned to a configured device if the following properties of the two devices match up:

- Article no.
- Device type or device family
- PROFINET device name

No automatic assignment

In the following situations, no automatic assignment is possible:

- No device can be identified online to match a configured device. (This means that the corresponding columns in the "Online topology" area of the topology comparison table are empty.)
  In this case, you should add the already configured device to your system or delete the configured device from the configuration.
- A device identified online can be assigned to a configured device, but there are differences in the port interconnections.
  In this case, you can apply the port interconnections identified online manually to the project (Page 301).
- A device identified online cannot be assigned to a configured device. (In this case, the corresponding columns in the "Offline topology" area of the topology comparison table are empty.)
  In this case, you can include the device identified online manually in the project (Page 301).

Apply the port interconnections identified online manually to the project

Requirement

You have run an offline/online comparison in the topology view. The result of this is that at least one device identified online was automatically assigned to a configured device, but that there are differences relating to the interconnection.

Procedure

To adopt one more port interconnections identified online in the project manually, follow these steps:

1. Select the row belonging to the port interconnection.
2. If applicable, select further roles using multi-selection.
3. Select "Apply" > "Use selected" in the shortcut menu.
   The content of the corresponding table cells in the "Action" column changes to "Apply".
4. If you have mistakenly prepared too many port interconnections to be included in the project:
   Select the rows that belong to the port interconnections you have mistakenly prepared for inclusion in the project using multi-selection.
   Select "Reset" > "Reset selected" in the shortcut menu.
   The content of the corresponding table cells in the "Action" column change to "No action".
5. Click the "Synchronize" button.

Result

The port interconnections identified online for the corresponding devices are included in the project. Successful adoption is indicated by the diagnostics icon "Identical topology information" for each port.

Note

If other port interconnections are recognized for a device identified online and these differ from those that exist in the project, adopting these in the project means that the port interconnections that were previously in the project are replaced by those identified online. If no port interconnections are detected for a device identified online, adopting in the project means that all the port interconnections of this device are deleted in the project.

Include the devices identified online manually in the project

Requirements

You have run an offline/online comparison in the topology view. The result of this is that at least one device identified online could not be assigned to any configured device.

Procedure

To adopt one more devices identified online in the project manually, follow these steps:

1. For a configured device without an online partner, move the mouse pointer to the "Device/port" column of the online topology.
2. Select the device you want to assign to the configured device from the drop-down list of this box.
3. Repeat the previous steps if necessary for other configured devices without an online partner.

Result

The selected device that was identified online is moved up from the end of the table. Following this, it is in the row of the configured device to which you have just assigned it.

Automatic assignment of devices by advanced offline/online comparison

Overview

With the advanced offline/online comparison, ICMP is also used alongside DCP to be able to detect devices that do not support DCP.

Automatic assignment of devices detected by ICMP

With devices detected by ICMP, no type is available.

With passive devices, no article number is available. For this reason, passive devices can only be assigned automatically if you have not assigned an article number in the configured data and the offline and online IP addresses match.

With switches, automatic assignment is possible if the offline and online article number, IP address and PROFINET device name match.

1.3.4 Secure Communication

1.3.4.1 Basics of Secure Communication

Basics of Secure Communication

For STEP 7 (TIA Portal) as of V14 and for S7-1500 CPUs as of firmware V2.0, the options for secure communication have been broadened considerably.

"S7-1500 CPUs" also refers to CPU versions S7-1500F, S7-1500T, S7-1500C as well as S7-1500pro CPUs and ET200SP CPUs.

Introduction

The attribute "secure" is used for the identification of communication mechanisms that are based on a Public Key Infrastructure (PKI) (for example RFC 5280 for Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List Profile). A Public Key Infrastructure (PKI) is a system that can issue, distribute and check digital certificates. The digital certificates issued are used within the PKI to secure computer-based communication. If a PKI uses asymmetric key cryptography, the messages in a network can be digitally signed and encrypted.

Components that you have configured in STEP 7 (TIA Portal) for secure communication use an asymmetric key encryption scheme with a public key (Public Key) and a private key (Private Key). TLS (Transport Layer Security) is used as the encryption protocol. TLS is the successor for the SSL (Secure Sockets Layer) protocol.

Objectives of secure communication

Secure communication is used to achieve the following objectives:

- Confidentiality
  i.e. the data is secret/cannot by read by eavesdroppers.
- Integrity
  i.e. the message that reaches the recipient is the same message, unchanged, that the sender sent. The message has not been altered on the way.
- End point authentication
  i.e. the end point communication partner is exactly who it claims to be and the party who is to be reached. The identity of the partner is verified.

In the past, these objectives were primarily relevant to IT and networked computers. Today, industrial machinery and control systems with sensitive data are at equally high risk, as they are also networked, and consequently pose high security requirements for data exchange.

Protection of the automation cell by means of the cell protection concept through firewall, or via connection through VPN, for example with the security module, was common in the past and remains so.

However, it is becoming increasingly necessary to also transfer data to external computers in encrypted form via Intranet or public networks.

Common principles of secure communication

Independent of the context, secure communication is based on the Public Key Infrastructure (PKI) concept and contains the following components:

- An asymmetric encryption scheme. This scheme allows the following:
  - Encryption or decryption of the messages using public or private keys.
  - The verification of signatures in messages and certificates.
    The messages / certificates are signed by the sender / certificate owner with their private key. The recipient / verifier checks the signature with the public key of the sender / certificate owner.
- Transport and storage of the public key using X.509 certificates.
  - X.509 certificates are digitally signed data that allow public key authentication in terms of the bound identity.
  - X.509 certificates can also contain information that describes in more detail or restricts use of the public key. For example the effective date of a public key in a certificate and when it expires.
  - X.509 certificates also contain information on the issuer of the certificate in secure form.

The following paragraphs give an overview of these basic concepts which are required for managing certificates in STEP 7 (TIA Portal), for example, and for programming the communication instructions for secure Open User Communication (sOUC).

Secure communication in STEP 7

STEP 7 as of V14 provides the required PKI for the configuration and operation of secure communication.

Examples:

- The Hypertext Transfer Protokoll (HTTP) becomes Hypertext Transfer Protokoll Secure (HTTPS) by means of the TLS (Transport Layer Security) protocol. Since HTTPS is a combination of HTTP and TLS, it is called "HTTP over TLS" in the corresponding RFC. You can see in the browser that HTTPS is being used. This is indicated by the URL scheme "https://" instead of "http://" in the address bar of the browser. Most browsers highlight such secure connections.
- Open User Communication becomes secure Open User Communication. The underlying protocol is also TLS.
- E-mail providers also offer access over the "Secure SMTP over TLS" protocol to increase the security of e-mail communication.

The figure below shows the TLS protocol in the context of communication layers.

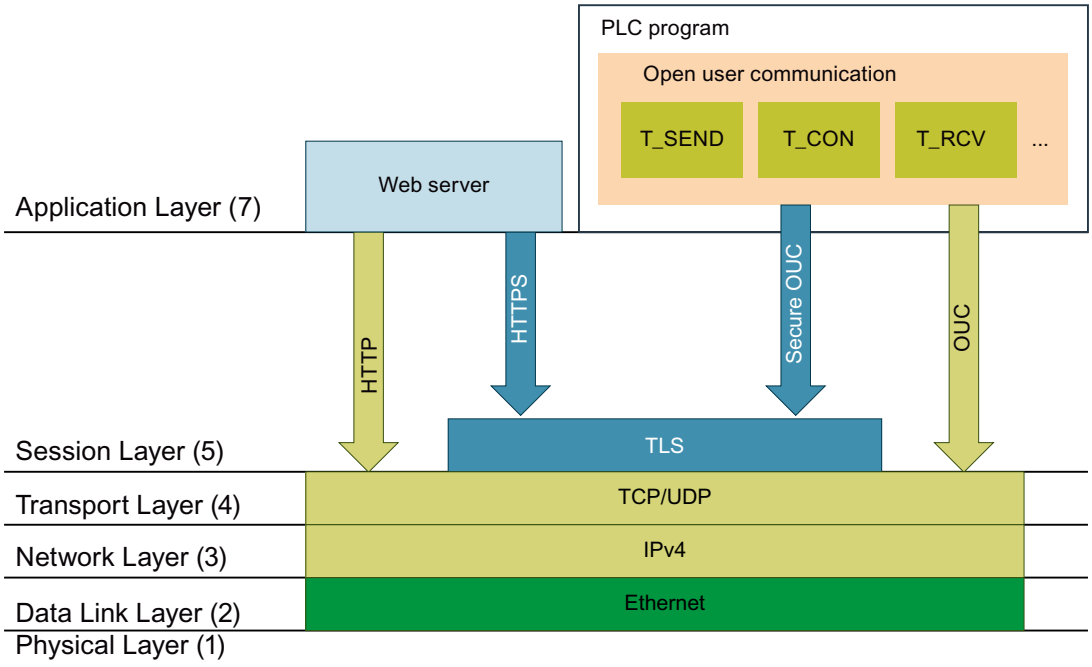

Secure communication with OPC UA

An OPC UA server is implemented in S7-1500 CPUs as of firmware V2.0. OPC UA Security also covers authentication, encryption and data integrity with digital X.509 certificates and also uses a Public Key Infrastructure (PKI). Depending on the requirements of the application you can select various security levels for the end point security. The description of the OPC UA server functionality is covered in a separate section.

See also

Setting options for security functions (Page 982)

Confidentiality through encryption

Message encryption is an important element of data security. When encrypted messages are intercepted by third parties during communication, these potential eavesdroppers cannot access the information they contain.

There is a wide range of mathematical processes (algorithms) for encrypting messages.

All algorithms process a "key" parameter to encrypt and decrypt messages.

- Algorithm + key + message => encrypted message
- Encrypted message + key + algorithm => (decrypted) message

Symmetric encryption

The central aspect of symmetric encryption is that both communication partners use the same key for message encryption and decryption, as shown in the figure below. Bob uses the same key for encryption as Alice uses for decryption. In general, we also say that the two sides share a secret, the secret key, with which they encrypt or decrypt a message.

① Bob encrypts his message with the symmetric key

② Alice decrypts the encrypted message with the symmetric key

The process can be compared to a briefcase to which the sender and recipient have the same key, which both locks and opens the case.

- Advantage: Symmetric encryption algorithms (for example AES, Advanced Encryption Algorithm) are fast.
- Disadvantages: How can the key be sent to a recipient without getting into the wrong hands? This is a key distribution problem. If enough messages are received, the key can also be worked out and must therefore be changed regularly.

If there is a large number of communication partners, there is also a large number of keys to distribute.

Asymmetric encryption

Asymmetric encryption works with a pair of keys consisting of one public key and one private key. In connection with a PKI it is also called a Public Key process or only PKI process: A communication partner, Alice in the figure below, has a private key and a public key. The public key is provided to the public, in other words any potential communication partner. Anyone with the public key can encrypt messages for Alice. In the figure below, this is Bob.

Alice's private key, which she must not disclose, is used by Alice to decrypt an encrypted message addressed to her.

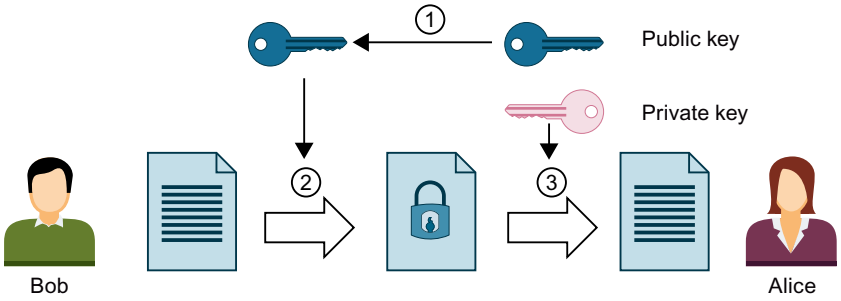

① Alice provides Bob with her public key. No precautionary measures are required to this purpose: Anyone can use the public key for messages to Alice if they are sure that it is actually Alice's public key.

② Bob encrypts his message with Alice's public key.

③ Alice decrypts the encrypted message from Bob with her private key. As only Alice has the private key and never discloses it, only she can decrypt the message. With her private key, she can decrypt any message encrypted with her public key - not only messages from Bob.

The system can be compared to a mailbox into which anyone can put a message, but from which only the person with the key can remove messages.

- Advantages: A message encrypted with a public key can only be decrypted by the owner of the private key. As another (private) key is required for decryption, it is also much harder to work out the decryption key on the basis of large numbers of encrypted messages. This means that the public key does not have to be kept strictly confidential, unlike with symmetric keys.
  Another advantage is easier distribution of public keys. No specially secured channel is required in asymmetric cryptography to transfer the public key from the recipient to the sender encrypting the messages. Less effort is thus required in the management of the keys than with the symmetric encryption process.
- Disadvantages: Complex algorithm (e.g. RSA, named after the three mathematicians Rivest, Shamir and Adleman), and therefore poorer performance than with symmetric encryption.

Encryption processes in practice

In practice, for example with a CPU Web server and Secure Open User Communication, the TLS protocol is used below the relevant application layer. Application layers are HTTP or SMTP, for example, as detailed above.

TLS (Transport Layer Security) uses a combination of asymmetric encryption and symmetric encryption (hybrid encryption) for secure data transfer, for example, over the Internet, and uses the following subprotocols:

- TLS Handshake Protocol, responsible for authentication of communication partners and negotiation of the algorithms and keys to be used for subsequent data transfer on the basis of asymmetric encryption
- TLS Record Protocol, responsible for encryption of user data with symmetric encryption and data exchange.

Both asymmetric and symmetric encryption are considered secure encryption schemes - there is basically no difference in security between the two procedures. The degree of security depends on parameters such as the selected key length.

Improper use of encryption

You cannot tell what identity is assigned to a public key from the bit string. A fraud could provide their public key and claim to be someone else. If a third party then uses this key thinking that they are addressing their required communication partner, confidential information could end up with the fraud. The fraud then uses their private key to decrypt the message that was not intended for them, and sensitive information falls into the wrong hands.

To prevent this type of abuse, the communication partners must be confident that they are dealing with the right communication partner. This trust is established by using digital certificates in a PKI.

Authenticity and integrity through signatures

Attacks from programs that intercept communication between the server and client and act as if they themselves were client or server, are called man-in-the-middle attacks. If the false identity of these programs is not detected, they can, for example, obtain important information via the S7 program or set values in the CPU and attack a machine or plants. Digital certificates are used to avoid such attacks.

Secure communication uses digital certificates that meet the standard X.509 of the International Telecommunication Union (ITU). This allows the identity of a program, a computer or an organization to be checked (authenticated).

How certificates establish trust

The main role of X.509 certificates is to bind an identity with the data of a certificate subject (for example e-mail address or computer name) to the public key of the identity. Identities can be people, computers or machines.

Certificates are issued by certificate authorities (Certificate Authority, CA) or by the owner of a certificate itself. PKI systems specify how users can trust the certificate authorities and the certificates that they issue.

The certificate process:

1. Anyone wishing to own a certificate submits a certificate application to a registration authority linked to the certificate authority.
2. The certificate authority assesses the application and applicant on the basis of set criteria.
3. If the identity of the applicant can be clearly established, the certificate authority confirms that identity by issuing a signed certificate. The applicant has now become the certificate holder.

The figure below is a simplified overview of the process. It does not show how Alice can check the digital signature.

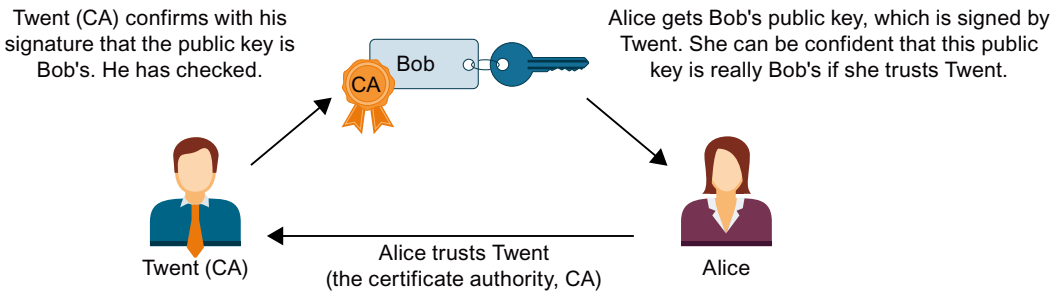

Self-signed certificates

Self-signed certificates are certificates whose signature comes from the certificate owner and not from an independent certificate authority.

Examples:

- You can create and sign a certificate yourself, for example, to encrypt messages to a communication partner. In the example above, Bob (instead of Twent) could himself sign his certificate with his private key. Alice can use Bob's public key to check that the signature and public key from Bob match. This procedure is sufficient for simple internal plant communication that is to be encrypted.
- A root certificate is, for example, a self-signed certificate, signed by the certificate authority (CA), that contains the public key of the issuer.

Features of self-signed certificates

The attributes "CN" (Common Name of Subject) for the certificate holder and "Issuer" (issuer) of the self-signed certificates are identical: You have signed your certificate yourself. The field "CA" (Certificate Autority) for the certificate authority must be set to "False"; the self-signed certificate should, after all, not be used to sign other certificates.

Self-signed certificates are not embedded in a PKI hierarchy.

Certificate content

A certificate to the X.509 V3 standard, the same standard that is used by STEP 7 and the S7-1500 CPUs, consists primarily of the following elements:

- Public key
- Details of the certificate holder (meaning the holder of the key). This is, for example, the Common Name (CN) of Subject
- Attributes such as serial number and validity period
- Digital signature from the certificate authority (CA) confirming that the information is correct.

There are also extensions, for example:

- Specification of what the public key may be used for (Key Usage), for example, signing or key encryption.
  When you create a new certificate with STEP 7, for example in the context of Secure Open User Communication, select the correct entry from the list of possible usages, e.g. "TLS".
- Specification of a "Subject Alternative Name" ("SAN"), which is used in secure communication with Web servers (HTTP over TLS), for example, to ensure that the certificate in the address bar of the Web browser also belongs to the Web server specified in the URL.

How signatures are generated and verified

Asymmetric key usage ensures that certificates can be verified: The example of the certificate "MyCert" illustrates the "Sign" and "Verify signature" processes.

Generating a signature:

1. The issuer of the "MyCert" certificate generates a hash value from the certificate data using a specific hash function (for example SHA-1, Secure Hash Algorithm).
   The hash value is a bit string of a constant length. The advantage of the constant length of the hash value is that it always takes the same amount of time to sign.
2. Using the hash value generated in this way and the private key, the issuer of the certificate then generates a digital signature. The RSA signature scheme is often used.
3. The digital signature is saved in the certificate. The certificate is now signed.

Verifying a signature:

1. The authenticator of the "MyCert" certificate obtains the certificate of the certificate issuer and thus also the public key.
2. A new hash value is formed from the certificate data with the same hash algorithm that was used for signing (for example SHA-1).
3. This hash value is then compared with the hash value that was determined by means of the public key of the certificate issuer and the signature algorithm to verify the signature on the certificate.
4. If the signature verification supplies a positive result, both the identity of the certificate holder as well as the integrity, meaning the authenticity and genuineness of the certificate contents, is proven. Anyone who has the public key, meaning the certificate from the certificate authority, can check the signature and thus prove that the certificate was actually signed by the certificate authority.

The figure below shows how Alice uses the public key in the certificate from Twent (who represents the certificate authority, CA) to verify the signature on Bob's public key. All that is required for verification is therefore the availability of the certificate from the certificate authority at the time of verification. The validation itself is executed automatically in the TLS session.

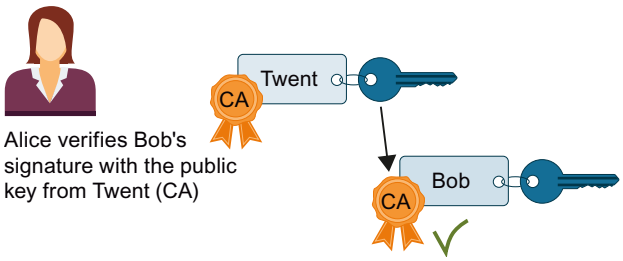

Signing messages

The method described above for signing and verifying also uses the TLS session for signing and verifying messages:

If a hash value is generated by a message and this hash value is signed with the private key of the sender and attached to the original message, the recipient of the message is able to check the integrity of the message. The recipient decrypts the hash value with the public key of the sender, puts together the hash value from the message received and compares the two values. If the values are not the same, the message has been tampered with on the way.

Chain of certificates to root certificate

The certificates of a PKI are often organized hierarchically: The top of the hierarchy is formed by root certificates. These are certificates that are not signed by a higher-level certificate authority. The certificate holder and certificate issuer of root certificates are identical. Root certificates enjoy absolute trust. They are the "anchor" of the trust and must therefore be known at the receiver as absolute certificates. They are stored in an area provided for trusted certificates.

The function of root certificates is to sign certificates from lower-level certificate authorities, so-called intermediate certificates. This transfers the trust from the root certificate to the intermediate certificate. An intermediate certificate can sign a certificate just like a root certificate; both are therefore referred to as "CA certificates".

This hierarchy can be continued over multiple intermediate certificates until the end-entity certificate. The end-entity certificate is the certificate of the user who is to be identified.

The validation process runs through the hierarchy in the opposite direction: As described above, the certificate issuer is established and the signature checked with the issuer's public key, then the certificate of the higher-level certificate issuer is established along the entire chain of trust to the root certificate.

Conclusion: The chain of intermediate certificates to the root certificate, the certificate path, must be available in every device that is to validate an end-entity certificate of the communication partner, irrespective of the type of secure communication that you configure.

Managing certificates with STEP 7

STEP 7 as of version V14 together with the S7-1500-CPUs as of FW version 2.0 support the Internet PKI (RFC 5280) in as far as an S7-1500 CPU is able to communicate with devices that similarly support the Internet PKI.

The use of X.509 certificates, for example to verify certificates as described in the preceding sections, is a result of this.

STEP 7 as of V14 uses a PKI similar to the Internet PKI. Certificate Revocation Lists (CRLs), for example, are not supported.

Create or assign certificates

For devices with security properties, such as an S7-1500 CPU as of firmware V2.0, you create certificates in STEP 7 for various applications.

The following areas in the Inspector window of the CPU allow the creation of new certificates or the selection of existing ones:

- "Protection & Security > Certificate manager" - for the generation or assignment of all types of certificates. The default setting for the creation of certificates is TLS certificates for Secure Open User Communication.
- "Web server > Security" - for the generation and assignment of Web server certificates.
- "OPC UA > Server > Security" - for the generation or assignment of OPC UA certificates

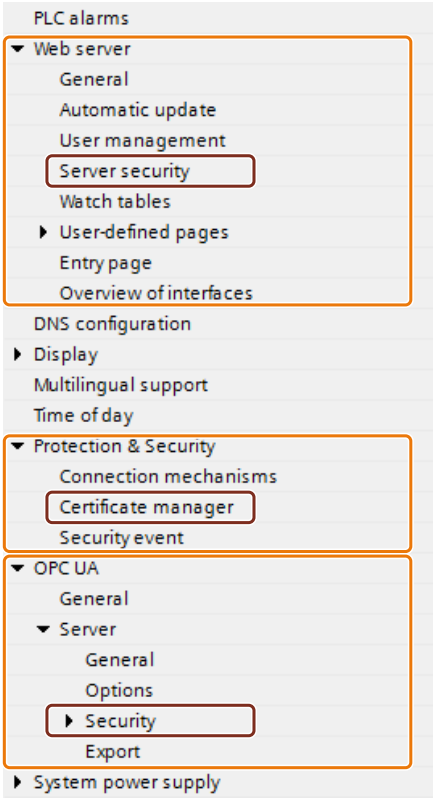

Special feature of the area "Protection & security > Certificate manager"

Only in this area of the Inspector window do you switch between the global, meaning the project-wide, certificate manager and the local, meaning the device-specific, certificate manager (option "Use global security settings for certificate manager"). The option decides whether you have access to all the certificates in the project or not.

- If you do **not** use the certificate manager in the global security settings, you only have access to the local certificate memory of the CPU. You do not have any access, for example, to imported certificates or root certificates. Without these certificates only limited functionality is available. You can, for example, only create self-signed certificates.
- If you do not use the certificate manager in the global security settings and you are logged on as an administrator, you have access to the global, project-wide certificate memory. You can, for example, assign imported certificates to the CPU or create certificates that are issued and signed by the project CA (certification authority of the project).

The figure below shows how the "Global security settings" are displayed in the project tree after activation of the option "Use global security settings for certificate manager" in the Inspector window of the CPU.

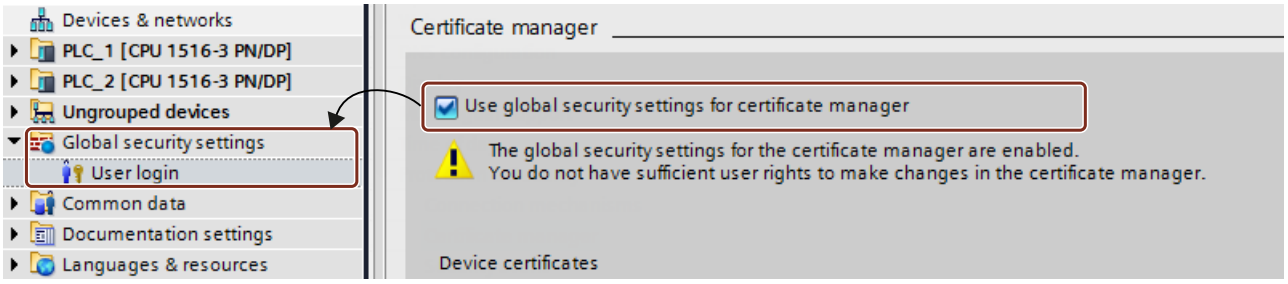

When you double-click "User login" in the project tree below the global security settings and log in, a line "Certificate manager" is displayed there among other things.

With a double-click on the "Certificate manager" line you receive access to all the certificates in the project, divided into the tabs "CA" (certificate authorities), "Device certificates" and "Trusted certificates and root certification authorities".

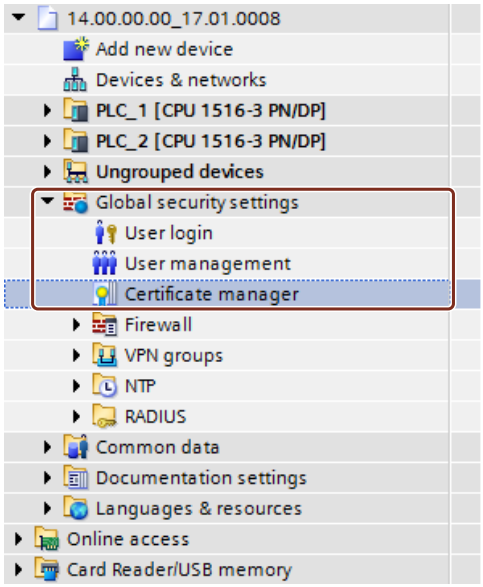

Private keys

STEP 7 generates private keys when generating device certificates or server certificates (end-entity certificates). It depends on the use of the global security settings for the certificate manager where the private key is stored encrypted:

- If you use global security settings, the private key is stored encrypted in the global (project-wide) certificate memory.
- If you do not use global security settings, the private key is stored encrypted in the local (CPU-specific) certificate memory.

The existence of the private key that is required, for example, to decrypt data, is displayed in the "Private key" column in the "Device certificates" tab of the certificate manager in the global security settings.

When the hardware configuration is downloaded, the device certificate, the public key as well as the private key are downloaded to the CPU.

NOTICE
The option "Use global security settings for certificate manager" influences the previously used private keys: If you have already created certificates without using the certificate manager in the global security settings and then change the option to the use of the certificate manager, the private keys are lost and the certificate ID can change. A warning draws your attention to this fact. Therefore specify at the beginning of the configuration which option is required for the certificate management.

Examples for the management of certificates

As explained in the preceding sections, certificates are required for every type of secure communication. The following section shows as an example how you handle the certificates with STEP 7 so that the requirements for Secure Open User Communication are met.

In the following section a distinction is made as to which devices the participating communication partners are. The respective steps for supplying the communication participants with the required certificates are described for each. An S7-1500 CPU or an S7-1500 software controller as of firmware version 2.0 is required.

The following generally applies:

During the establishment of a secure connection ("handshake") the communication partners as a rule only transfer their end-entity certificates (device certificates).

Therefore the CA certificates required to verify the transferred device certificate must be located in the certificate memory of the respective communication partner.

Note

The current date/time must be set in the CPU.

When using secure communication (for example, HTTPS, secure OUC, OPC UA), make sure that the corresponding modules have the current time of day and the current date. Otherwise, the modules evaluate the used certificates as invalid and the secure communication does not work.

Secure Open User Communication between two S7-1500 CPUs

Two S7-1500 CPUs PLC_1 and PLC_2 should interchange data over Secure Open User Communication.

You generate the required device certificates with STEP 7 and assign them to the CPUs as described below.

STEP 7 project certification authorities (CA of the project) are used to sign the device certificates.

The certificates are to be referenced by their certificate ID in the user program (TCON communication instruction in combination with the associated system data type, for example TCON_IPV4_SEC). STEP 7 assigns the certificate ID automatically when generating or creating certificates.

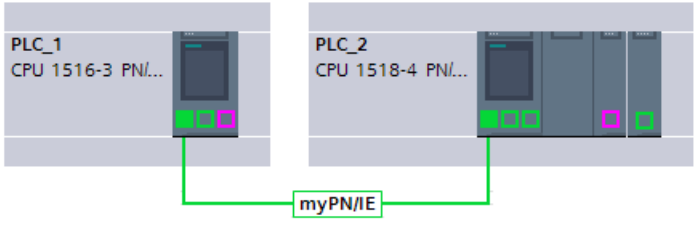

Procedure

STEP 7 automatically downloads the required CA certificates together with the hardware configuration to the participating CPUs so that the requirements for certificate verification exist for both CPUs. You therefore only have to generate the device certificates for the respective CPU - everything else is done by STEP 7 for you.

1. Select PLC_1 and activate the "Use global security settings for certificate manager" option in the "Protection & Security" area.
2. Log in as a user in the project tree in the "Global security settings" area. The "Administrator" role is the default for the first login to a new project.
3. Return to the PLC 1 in the "Protection & Security" section. Click in an empty line in the "Certificate holder" column in the "Device certificates" table to add a new certificate.
4. Click the "Add" button in the drop-down list to select a certificate.
   The "Generate new certificate" dialog opens.
5. Leave the default settings in this dialog. They are tailored to the usage of Secure Open User Communication (usage: TLS).
   Tip: Extend the preset name of the certificate holder - in this case the CPU name. In order to differentiate better, leave the default CPU name in case you have to manage a large number of device certificates.
   Example: PLC_1/TLS becomes PLC_1-SecOUC-Chassis17FactoryState.
6. Compile the configuration.
   The device certificate and the CA certificate are part of the configuration.
7. Repeat the steps described for PLC_2.

In the next step you have to create the user programs for the data exchange and download the configurations together with the program.

Use self-signed certificates instead of CA certificates

When creating device certificates you can select the "Self-signed" option. You can create self-signed certificates without having to be logged in for the global security settings. This procedure is not recommended because the certificates created by this method do not exist in the global certificate memory and can therefore not be assigned directly to a partner CPU.

As described above you should select the name of the certificate holder with care so that the right certificate can be assigned to a device without any doubt.

A verification with the CA certificates of the STEP 7 project is not possible with self-signed certificates. To ensure that self-signed certificates can be verified, you have to include the self-signed certificate of the communication partner in the list of trusted partner devices for each CPU. To do so, you must have activated the option "Use global security settings for certificate manager" and be logged in as a user in the global security settings.

Proceed as follows to add the self-signed certificate of the communication partner of the CPU:

1. Select PLC_1 and navigate to the table "Certificates of partner devices" in the "Protection & Security" area.
2. Click in an empty line of the "Certificate holder" column to open the drop-down list for adding or selecting certificates.
3. From the drop-down list select the self-signed certificate of the communication partner and confirm the selection.

In the next step you have to create the user programs for the data exchange and download the configurations together with the program.

Secure Open User Communication between an S7-1500 CPU as a TLS client and an external device as a TLS server

Two devices are to interchange data via a TLS connection or TLS session, for example to exchange recipes, production data or quality data:

- An S7-1500 CPU (PLC_1) as the TLS client; the CPU uses Secure Open User Communication
- An external device (such as a Manufacturing Execution System (MES)) as the TLS server

As the TLS client the S7-1500 CPU establishes the TLS connection / session to the MES system.

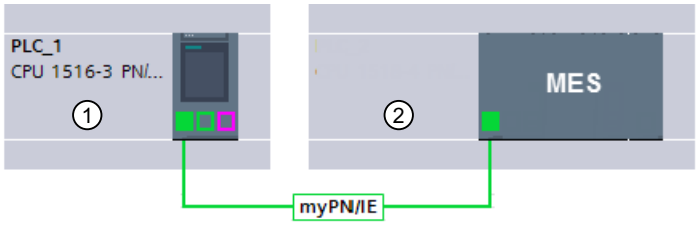

① TLS client
② TLS server

In order to authenticate the TLS server, the S7-1500 CPU requires the CA certificates of the MES system: The root certificate and if appropriate the intermediate certificates for verifying the certificate path.

You have to import these certificates into the global certificate memory of the S7-1500 CPU.

Proceed as follows to import certificates of the communication partner:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the matching table (trusted certificates of root certification authorities) for the certificate to be imported.
3. Right-click in the table to open the shortcut menu. Click "Import" and import the required certificate or the required CA certificates.
   Through the import the certificate receives a certificate ID and can be assigned to a module in the next step.

4. Select PLC_1 and navigate to the table "Certificates of partner devices" in the "Protection & Security" area.
5. Click in an empty line of the "Certificate holder" column to add the imported certificates.
6. From the drop-down list select the required CA certificates of the communication partner and confirm the selection.

Optionally the MES system can also request a device certificate of the CPU to authenticate the CPU (meaning the TLS client). In this case the CA certificates of the CPU must be available to the MES system. Prerequisite for importing the certificates into the MES system is a preceding export of the CA certificates from the STEP 7 project of the CPU. Proceed as follows:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the matching table (CA certificates) for the certificate to be exported.
3. Right-click to open the shortcut menu for the selected certificate.
4. Click "Export".
5. Select the export format of the certificate.

In the next step you have to create the user programs for the data exchange and download the configurations together with the program.

Secure Open User Communication between an S7-1500 CPU as TLS server and an external device as TLS client

If the S7-1500 CPU acts as a TLS server and the external device, for example an ERP system (Enterprise Resource Planning System), establishes the TLS connection / session, you require the following certificates:

- For the S7-1500 CPU you generate a device certificate (server certificate) with a private key and download it with the hardware configuration. When generating the server certificate you use the option "Signed by certificate authority".
  The private key is required for the key exchange as explained in the figure for the example "HTTP over TLS".
- For the ERP system you have to export the CA certificate of the STEP 7 project and import / download it to the ERP system. With the CA certificate, the ERP system checks the server certificate of the S7-1500 that was transmitted from the CPU to the ERP system during the establishment of the TLS connection/session.

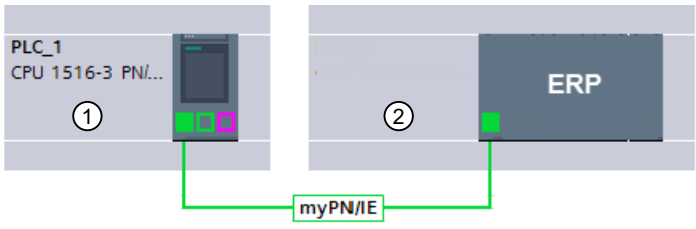

① TLS server
② TLS client

The description of the required actions is provided in the preceding sections.

Secure Open User Communication via S7-1543-1 CP as an e-mail client (SMTP over TLS)

An S7-1500 CPU can establish a secure connection to an e-mail server via the CP 1543-1 with the communication instruction TMAIL-C.

The system data types TMail_V4_SEC, TMail_V6_SEC and TMail_QDN_SEC allow you to specify the partner port of the e-mail server and thus to reach the e-mail server via "SMTP over TLS".

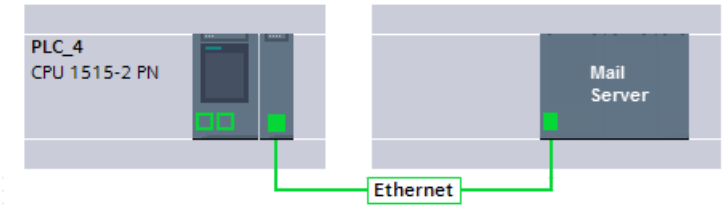

Prerequisite for a secure e-mail connection is an import of the root certificate and of the intermediate certificates from the e-mail server (provider) into the global certificate memory of the STEP 7 project. When these certificates have been assigned to the CP, the CP can check the server certificate of the e-mail server that is sent by the e-mail server during the establishment of the TLS connection / session by means of this certificate.

Follow these steps:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the matching table (trusted certificates of root certification authorities) for the certificate to be imported.
3. Right-click in the table to open the shortcut menu. Click "Import" and import the required certificate or the required CA certificates.
   Through the import the certificate receives a certificate ID and can be assigned to the CP in the next step.
4. Select the CP 1543-1 and navigate to the table "Certificates of partner devices" in the "Security" section.
5. Click in an empty line of the "Certificate holder" column to add the imported certificates.
6. From the drop-down list select the required CA certificates of the e-mail server and confirm the selection.

In the next step you have to create the user programs for the e-mail client function of the CPU and download the configurations together with the program.

See also

Secure OUC between two S7-1500 CPUs (Page 331)

Secure OUC from an S7-1500 CPU as a TLS client to an external PLC (TLS server) (Page 334)

Secure OUC from an S7-1500 CPU as TLS server to an external PLC (TLS client) (Page 336)

Secure OUC via e-mail (Page 343)

Secure OUC via CP interface (Page 339)

Example: HTTP over TLS

The following paragraphs show how the mechanisms described are used to create secure communication between a Web browser and the Web server of an S7-1500 CPU.

First the changes for the option "Permit access only through HTTPS" in STEP 7 are described. As of STEP 7 V14 you have the option of influencing the server certificate of the Web server of an S7-1500 CPU as of firmware V2.0: The server certificate is generated with STEP 7 as of this version.

In addition the processes that are implemented when a website of the Web server of the CPU is called up with a Web browser of a PC via an encrypted HTTPS connection are displayed.

Using Web server certificates for S7-1500 CPUs as of FW V2.0

For S7-1500 CPUs with a firmware version before V2.0, you were able to set "Permit access only with HTTPS" when setting the Web server properties, without specific requirements applying.

You did not have anything to do with the handling of certificates for these CPUs; the CPU automatically generates the certificates required for the Web server.

For S7-1500 CPUs as of firmware V2.0, STEP 7 generates the server certificate (end-entity certificate) for the CPU. You assign a server certificate to the Web server in the properties of the CPU (Web server > Server security).

Since a server certificate name is always preset, nothing changes in the simple configuration of the Web server: You activate the Web server and activate the option "Permit access only with HTTPS" - STEP 7 generates a server certificate with the preset name during compiling.

Irrespective of whether you use the certificate manager in the global security settings or not: STEP 7 has all the information needed to generate the server certificate.

In addition you have the option of determining the properties of the server certificate, for example, the name or the duration of validity.

Note

The current date/time must be set in the CPU.

When using secure communication (for example, HTTPS, secure OUC, OPC UA), make sure that the corresponding modules have the current time of day and the current date. Otherwise, the modules evaluate the used certificates as invalid and the secure communication does not work.

Downloading the Web server certificate

The server certificate generated by STEP 7 is automatically also loaded when the hardware configuration is downloaded to the CPU.

- If you use the certificate manager in the global security settings, the certificate authority of the project (CA certificate) signs the server certificate of the Web server. When downloading the CA certificate of the project is automatically loaded as well.
- If you do not use the certificate manager in the global security settings, STEP 7 generates the server certificate as a self-signed certificate.

When you address the Web server of the CPU via the IP address of the CPU, a new server certificate (end-entity certificate) must be generated and loaded with each change of the IP address of an Ethernet interface of the CPU. This is necessary because the identity of the CPU changes with the IP address – and the identity requires a signature in accordance with the PKI rules.

You can avoid this problem by addressing the CPU with a domain name instead of its IP address, e.g. "myconveyer-cpu.room13.myfactory.com". To do so, you need to manage the domain names of your CPUs over a DNS server.

Supplying a Web browser with a CA certificate of the Web server

In the Web browser, the user who accesses the websites of the CPU through HTTPS should install the CA certificate of the CPU. If no certificate is installed, a warning is output recommending that you do not use the page. To view this page, you must explicitly "Add an exception".

The user receives the valid root certificate (Certification Authority) for download from the "Intro" web page of the CPU Web server under "Download certificate".

STEP 7 offers a different possibility: Export the CA certificate of the project with the certificate manager into the global security settings in STEP 7. Subsequently import the CA certificate into the browser.

Course of Secure Communication

The figure below shows, in simplified terms, how communication is established ("handshake"), focusing on the negotiation of keys used for data exchange (here via HTTP over TLS).

However, the course can theoretically be applied to all communication options that are based on the use of TLS, i.e. also for Secure Open User Communication (see Basics of secure communication).

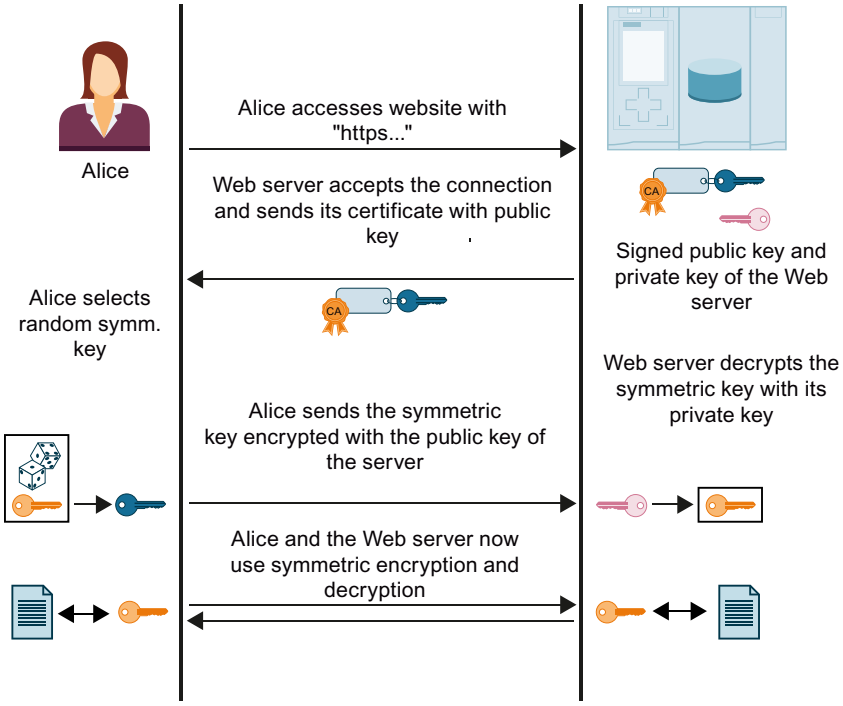

The figure does not show the measures taken at Alice's end (browser) to verify the certificate sent by the Web server. Whether Alice can trust the transferred Web server certificate and thus the identity of the Web server and allow the data exchange depends on a positive outcome of the verification.

The steps for verifying the authenticity of the Web server are:

1. Alice must know the public keys of all relevant certificate authorities, i.e. she requires the complete certificate chain to be able to verify the Web server certificate (i.e. the end-entity certificate of the Web server).
   Alice will generally have the required root certificate in her certificate memory. When a Web browser is installed, a whole range of trusted root certificates is also installed. If she does not have the root certificate, she has to download it from the certificate authority and install it in the certificate memory of the browser. The certificate authority can also be the device on which the Web server is located.
   You have the following options for obtaining the intermediate certificates:
   - The server itself sends the required intermediate certificates along with its end-entity certificate to Alice – in the form of a signed message so that Alice can check the integrity of the certificate chain.
   - The certificates often contain the URLs of the certificate issuer. Alice can load the required intermediate certificates from these URLs.

   When you handle certificates in STEP 7, it is always assumed that you have imported the required intermediate certificates and the root certificate into the project and have assigned them to the module.
2. Alice validates the signatures in the certificate chain with the public keys of the certificates.

3. The symmetric key has to have been generated and transferred to the Web server.
4. If the Web server is addressed by its domain name, Alice verifies the identity of the Web server in accordance with the PKI rules defined in RFC 2818. She is able to do this because the URL, in this case the "Fully Qualified Domain Name (FQDN) of the Web server, is saved in the end-entity certificate of the Web server. If the certificate entry in the "Subject Alternative Name" field corresponds to the entry in the address bar of the browser, everything is fine.

The process continues with the exchange of data with the symmetric key, as shown in the figure above.

See also

Certificates for the Web server (Page 989)

1.3.4.2 Global security settings

Global security certificate manager

Introduction

In the global certificate manager, you have an overview of all the certificates, for example, CA certificates, used in the project with information about the certificate holder, issuer, validity, use and the existence of a private key. With the corresponding authorization you can also manage certificates for the project there.

Activation of the global security settings

The global security settings are not visible until they have been activated in the project for at last one device. The "Use global security settings for certificate manager" check box is available to this end in the local CPU-specific certificate manager:

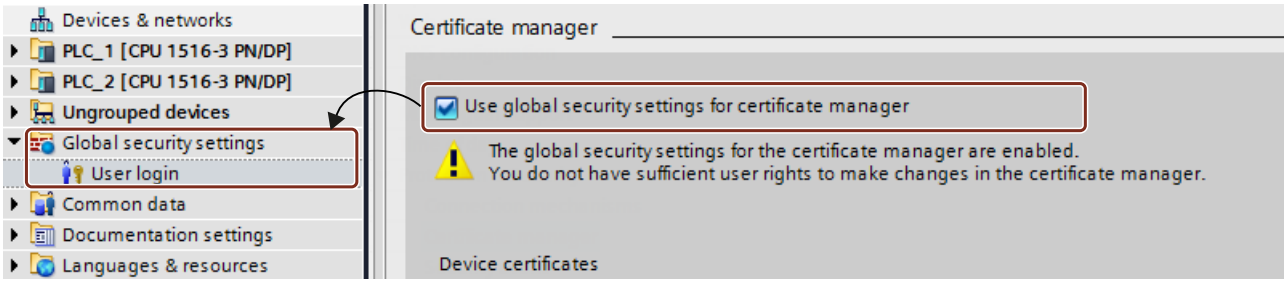

The check box is located in the Inspector window in the general settings of a device with security functions under "Protection & Security > Certificate manager".

Principle

The CA certificate is issued by a certificate authority from which the device certificates are derived.

Certificate authorities can be:

- STEP 7: If the "certificate holder" and "issuer" are the same, this is a self-signed certificate; in other words a certificate issued by STEP 7.
- Higher-level certificate authority: These external certificates are external to the project and are imported and stored in the certificate memory of STEP 7.

Certificates created by one of the two certificate authorities always have a private key so that the device certificates can be derived from them.

Functions of the global security certificate manager

The following functions are available for selection in the global security certificate manager:

- Import of new certificates and certificate authorities.
- Import of SSL certificates (only CP x43-1 Adv.), for example, for FTP communication.
- Export of the certificates and certificate authorities used in the project.
- Renewal of expired certificates and certificate authorities.
- Replacement of existing certificate authorities.
- Adding trusted certificates and certification authorities.

Deleting manually imported certificates.

Note

Downloading the configuration

After replacing or renewing certificates the configuration has to be downloaded to the corresponding devices. After replacing or renewing CA certificates the configuration has to be downloaded to the corresponding devices.

Note

Current date and current time of day on the security modules

When using secure communication (for example, HTTPS), make sure that the corresponding devices have the current time of day and the current date. Otherwise the certificates used are not evaluated as valid and the secure communication does not work.

See also

Local CPU-specific certificate manager (Page 983)

Managing users and roles

Introduction

After you have activated the global security settings for the certificate manager, you have to log in to the global security settings or you require administrator rights. Otherwise you do not have sufficient user rights to edit the global certificates. You cannot access the global certificate manager without a login or if you do not have sufficient rights.

Roles and rights

You can configure the security settings if the predefined roles "Administrator" or "Standard" are assigned to you or if a different defined role with the associated authorization is assigned. You can view the roles in the project tree under "Global security settings > User administration > Roles" or, if you have corresponding rights, edit them.

Assigned role	Diagnosing	Configuring	Managing users and roles
Administrator	Yes	Yes	Yes
Standard	Yes	Yes	No
Diagnostics	Yes	No	No

You can create new roles and users as an administrator and assign users and roles to each other in the user administration.

Requirement

You have to be logged in as an administrator via the user login to the global security settings.

Procedure

Proceed as follows to edit users and roles:

1. In the project navigator go to user administration in the global security settings.
2. Select the role to be changed or create a new role.
3. Edit the rights of the selected role in the rights list.

Importing and exporting certificates

Introduction

You can import and export certificates and the associated private keys. Certificates can only be imported with the global certificate manager. Certificates can also be exported via the local CPU-specific certificate manager.

You can import or export the certificates into the following file types:

- Certificates without private key to CER, DER, CRT or PEM
- Certificates with their private key to P12 (PKCS12).

Requirement

During import of certificates with private key you may require a password for access.

Importing certificates

To import a certificate follow these steps:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the matching table (CA certificates, device certificates, trusted certificates of root certification authorities) for the certificate to be imported.
3. Right-click in the table to open the shortcut menu.
4. Click "Import".
5. Select the import format of the certificate:
   – CER, DER, CRT or PEM for certificates without private keys.
   – P12 (PKCS12 archive) for certificates with private key.
6. Click "Open" to import the certificate.

The imported certificate has to be assigned manually to the desired device or module.

Exporting certificates

To export a certificate, follow these steps:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the matching table (CA certificates, device certificates, trusted certificates of root certification authorities) for the certificate to be exported.
3. Right-click to open the shortcut menu for the selected certificate.
4. Click "Export".
5. Select the export format of the certificate:
   – CER, DER, CRT or PEM for certificates without private keys.
   – CRL for certificate block list encoded to DER
   – P12 (PKCS12 archive) for certificates with private key.
6. Click "Save" to export the certificate.

You can also export certificates directly from the local CPU-specific certificate manager.

Selectable file formats

X.509 certificates to the certificate standard X.509 V3 from the RFC 5280 can be imported or exported in various formats with different file name extensions:

File name extension	PEM
File type	Privacy Enhanced Mail Security Certificate
Usage	Base64-encoded certificate
MIME type	application/x-pem-file
Meaning	Certificate in PEM/X509 format (PEM = Privacy-Enhanced Mail) with an encoding in ASCII (Base64). Format of the certificate authorities often used for the certificate transfer via e-mail to the applicant of the certificate. Instead of the file name extension PEM, a Base64-enoded certificate can also have the extensions CER or CRT.
Usage	In TIA Portal the import and export is supported with PEM encoding. An export with PEM encoding can be effected in the file format PEM or CRT.

File name extension	DER
File type	DER Encoded X509 Certificate
Usage	Binary encoded certificate
MIME type	application/x-x509-ca-cert, application/pkix-cert
Meaning	Certificate in DER/X.509 format (DER = Distinguished Encoding Rules) with binary encoding. The binary DER format is the basis for the ASCII-encoded PEM format. The DER format is used in particular in the Microsoft Windows environment and on Java platforms. Instead of the file name extension DER a binary-encoded certificate can also have the extensions CER or CRT.
Usage	In TIA Portal the import and export is supported with DER encoding. An export with DER encoding can be effected in the file format DER or CER.

File name extension	CER
File type	Internet Security Certificate File
Usage	After PEM or DER encoded certificate
MIME type	application/x-x509-ca-cert, application/pkix-cert, application/keychain_access
Meaning	Alternative file name extension CER (Canonical Encoding Rules) for a binary encoded certificate in the DER format or for a Base64/ASCII-encoded certificate in PEM format.
Usage	In TIA Portal the import and export is supported in the file format CER. When importing a certificate in the file format CER the encoding can be to PEM or DER. The DER encoding is used for export in the file format CER.

File name extension	CRT
File type	Internet Security Certificate File
Usage	After PEM or DER encoded certificate
MIME type	application/x-x509-ca-cert, application/pkix-cert, application/keychain_access

File name extension	CRT
Meaning	Alternative file name extension CRT for a binary encoded certificate in the DER format or for a Base64/ASCII-encoded certificate in PEM format.
Usage	In TIA Portal the import and export is supported in the file format CRT. When importing a certificate in the file format CRT the encoding can be to PEM or DER. The PEM encoding is used for export in the file format CRT.

File name extension	CRL
File type	Certificate Revocation List
Usage	Certificate block list encoded to DER
MIME type	application/pkix-crl, application/x-pkcs7-crl
Meaning	The CRL (Certificate Revocation List) is an X.509 V3 extension and contains the list of blocked or revoked certificates for a specific root certificate. Certificate authorities use the option to revoke the authorization of certificates already issued. This is the case when certificates are compromised or if their content turns out to be faulty. Blocked certificates can be unblocked again with the CRL, for example, when the certificate turns out not to be compromised after all or if the content was correct after all. Revoked certificates remain invalid forever and cannot be authorized again. The certificate authority updates its CRLs as soon as their validity has expired or the content has changed. The source of supply of a CRL is entered in the respective certificate.
Usage	The TIA Portal does not support a certificate revocation via CRL. Since some external clients must have certificate block lists for the validation of the certificates, an export of the CRL is however supported. But the list is empty and solely serves the purpose of making the validation of an output certificate by certain external clients possible.

File name extension	P12
File type	Personal Information Exchange File
Usage	As PKCS12-archive encoded certificates, optionally with private key
MIME type	application/keychain_access, application/x-pkcs12
Meaning	File types in the format P12 can also contain the private key in addition to the server certificate and intermediate certificates. A password can be assigned when storing.
Usage	In TIA Portal the import and export is supported in the file format P12. The option for exporting into a PKCS12 archive only exists if the certificate disposes of a private key. During export a password can be assigned for the PKCS12 archive. Analog to the password assignment during exporting, the entry of a password may be required to access the certificate during importing.

Renewing certificates

Introduction

You can renew certificates, for example, to update a compromised certificate by an imported or new certificate.

Procedure

To renew a certificate, follow these steps:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the matching table (CA certificates, device certificates) for the certificate to be renewed.
3. Right-click to open the shortcut menu for the selected certificate.
4. Click "Renew".
5. You can edit the data of the selected certificate in the dialog for new certificates.
6. When you click "OK", the certificate is renewed with the changed values.

When you renew a certificate, the old certificate is replaced by the certificate with the new values. Through the renewal of certificates no signature query to a certificate authority is generated.

Replacing certificates

Introduction

You can replace an existing certificate by a different certificate.

Procedure

To replace a certificate, follow these steps:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the matching table (CA certificates, device certificates, trusted certificates of root certification authorities) for the certificate to be replaced.
3. Right-click to open the shortcut menu for the selected certificate.
4. Click "Replace".
5. In the dialog for changing the certificate authority you replace the existing CA certificate of the project or the CA group certificate with a new one.

All certificates listed in the "Certificates involved" table are derived once again.

Show certificates

Introduction

You can have an overview of all the certificate data displayed via the Windows certificate dialog

Procedure

Proceed as follows to display the certificate data:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the matching table (CA certificates, device certificates, trusted certificates of root certification authorities) for the certificate to be shown.
3. Right-click to open the shortcut menu for the selected certificate.
4. Click "Show".

Result

Different information about the selected certificate is displayed in a window:

- General information about the certificate
- Detailed information about the certificate
- Information about the certification path

Deleting certificates

Introduction

You can delete certificates that are no longer required or are compromised.

Procedure

To delete a certificate, follow these steps:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the matching table (device certificates, trusted certificates of root certification authorities) for the certificate to be deleted.
3. Right-click to open the shortcut menu for the selected certificate.
4. Click "Delete".

Note

CA certificates reserved for the project and certificates that are assigned automatically to a CP cannot be deleted.

You can also delete certificates directly in the local CPU-specific certificate manager. The certificate then remains in the global security manager for the project, but is no longer used for the device.

1.3.4.3 Secure Open User Communication

Secure OUC between two S7-1500 CPUs

The following section describes how you can set up a Secure Open User Communication over TCP between two S7-1500 CPUs In the process one S7-1500 CPU acts as a TLS client (active connection establishment) and the other S7-1500 CPU as a TLS server (passive connection establishment).

Setting up a secure TCP connection between two S7-1500 CPUs

In order to establish secure TCP communication between two S7-1500 CPUs you have to create a data block with the system data type TCON_IPv4_SEC in each CPU, carry out the parameter assignment and call it directly at the instruction. The TCON instruction supports the system data types TCON_QDN_SEC and TCON_IP_V4_SEC. As of firmware version V2.5, the TSEND_C and TRCV_C instructions also support the TCON_QDN_SEC and TCON_IP_V4_SEC system data types.

Requirements:

- Both S7-1500 CPUs have a minimum firmware version of V2.0
- TLS client and TLS server have all the required certificates.

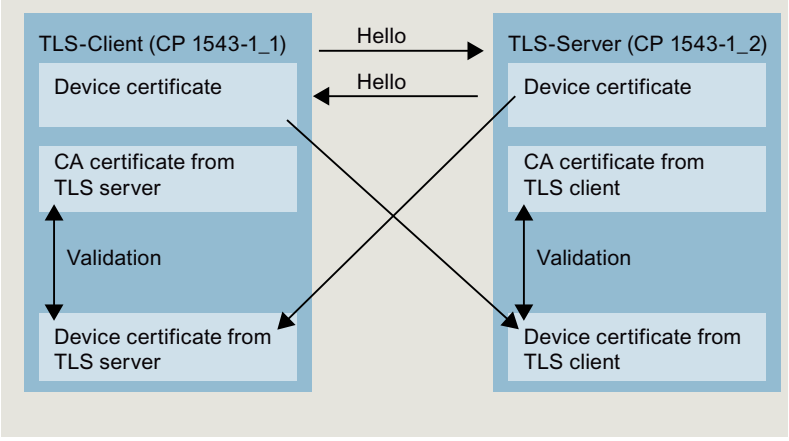

Settings at the TLS client

To set up a secure TCP connection in the TLS client, follow these steps:

1. Create a global data block in the project tree.
2. Define a tag of the data type TCON_IP_4_SEC in the global data block. To do so, enter the string "TCON_IP_V4_SEC" in the "Data type" field.
   The example below shows the global data block "Data_block_1" in which the tag "SEC connection 1 TLS Client" of the data type TCON_IP_V4_SEC is defined.

Data_block_1

	Name	Data type	Start value	Comment
1	Static			
2	SEC connection 1 TLS-Client	TCON_IP_V4_SEC		
3	ConnPara	TCON_IP_v4		parameter of the TCP connection
4	InterfaceId	HW_ANY	72	HW-identifier of IE-interface submodule
5	ID	CONN_OUC	10	connection reference / identifier
6	ConnectionType	Byte	11	type of connetion: 11=TCP/IP, 19=UDP (17=TCP/IP)
7	ActiveEstablished	Bool	true	active/passive connection establishment
8	RemoteAddress	IP_V4		remote IP address (IPv4)
9	ADDR	Array[1..4] of Byte		IPv4 address
10	ADDR[1]	Byte	192	IPv4 address
11	ADDR[2]	Byte	168	IPv4 address
12	ADDR[3]	Byte	1	IPv4 address
13	ADDR[4]	Byte	100	IPv4 address
14	RemotePort	UInt	4711	remote UDP/TCP port number
15	LocalPort	UInt	4711	local UDP/TCP port number
16	ActivateSecureConn	Bool	true	activate the security functionality of that connection in general
17	TLSServerReqClientCert	Bool	false	Just for server side: The TLS server requests a client certificate
18	ExtTLSCapabilities	Word	16#0	Bit 0: Just for client side: validate given IPv4 address against th
19	TLSServerCertRef	UDInt	1	for Server side: Reference to own X.509 V3 server certificate; fo
20	TLSClientCertRef	UDInt	5	for Client side: add id of own X.509 V3 client certificate; for Sen

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the IPv4 address of the TLS server for "RemoteAddress".
4. Set the parameters for Secure Communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerCertRef": ID of the own X.509-V3 certificate.
   - "TLSClientCertRef": Enter, for example, the value 1. This references the CA certificate of the TIA Portal project (SHA1).
5. Create a TCON instruction in the program editor.
6. Interconnect the CONNECT parameter of the TCON instruction with the tag of the data type TCON_IP_V4_SEC.

Settings at the TLS server

To set up a secure TCP connection in the TLS server, follow these steps:

1. Create a global data block in the project tree.
2. In the global data block, define a tag of the data type TCON_IP_V4_SEC.
   The example below shows the global data block "Data_block_1" in which the tag "SEC connection 1 TLS Server" of the data type TCON_IP_V4_SEC is defined.

Data_block_1

	Name	Data type	Start value	Comment
1	Static			
2	SEC connection 1 TLS-Server	TCON_IP_V4_SEC		
3	ConnPara	TCON_IP_v4		parameter of the TCP connection
4	InterfaceId	HW_ANY	120	HW-identifier of IE-interface submodule
5	ID	CONN_OUC	10	connection reference / identifier
6	ConnectionType	Byte	11	type of connetion: 11=TCP/IP, 19=UDP (17=TCP/IP)
7	ActiveEstablished	Bool	false	active/passive connection establishment
8	RemoteAddress	IP_V4		remote IP address (IPv4)
9	ADDR	Array[1..4] of Byte		IPv4 address
10	ADDR[1]	Byte	192	IPv4 address
11	ADDR[2]	Byte	168	IPv4 address
12	ADDR[3]	Byte	1	IPv4 address
13	ADDR[4]	Byte	10	IPv4 address
14	RemotePort	UInt	4711	remote UDP/TCP port number
15	LocalPort	UInt	4711	local UDP/TCP port number
16	ActivateSecureConn	Bool	true	activate the security functionality of that connection in general
17	TLSServerReqClientCert	Bool	true	Just for server side: The TLS server requests a client certificate
18	ExtTLSCapabilities	Word	16#0	Bit 0: Just for client side: validate given IPv4 address against the
19	TLSServerCertRef	UDInt	6	for Server side: Reference to own X.509 V3 server certificate; fo
20	TLSClientCertRef	UDInt	1	for Client side: add id of own X.509 V3 client certificate; for Serv

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the IPv4 address of the TLS client for "RemoteAddress".
4. Set the parameters for Secure Communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerReqClientCert ": Request for an X.509-V3 certificate from the TLS client Enter the value "true".
   - "TLSServerCertRef": ID of the own X.509-V3 certificate.
   - "TLSClientCertRef": Enter, for example, the value 1. This references the CA certificate of the TIA Portal project (SHA1).
5. Create a TCON instruction in the program editor.
6. Interconnect the CONNECT parameter of the TCON instruction with the tag of the data type TCON_IP_V4_SEC.

See also

Connection parameters according to TCON_IP_V4_SEC (Page 110)

Connection parameters in accordance with TCON_QDN_SEC (Page 113)

Examples for the management of certificates (Page 314)

Secure OUC from an S7-1500 CPU as a TLS client to an external PLC (TLS server)

The following section describes how you can set up a Secure Open User Communication over TCP from an S7-1500 CPU as TLS client to a TLS server.

Setting up a secure TCP connection from an S7-1500 CPU as TLS client to a TLS server

S7-1500 CPUs as of firmware version V2 support Secure Communication with addressing via a Domain Name System (DNS).

In order to establish secure TCP communication via the domain name you have to create a data block yourself with the system data type TCON_QDN_SEC, carry out the parameter assignment and call it directly at the instruction. The TCON instruction supports the system data type TCON_QDN_SEC.

Requirements:

- At least one DNS server exists in your network.
- You have configured at least one DNS server for the S7-1500 CPU.
- TLS client and TLS server have all the required certificates.

To set up a secure TCP connection to a TLS server, follow these steps:

1. Create a global data block in the project tree.
2. In the global data block, define a tag of the data type TCON_QDN_SEC.
   The example below shows the global data block "Data_block_1" in which the tag "DNS ConnectionSEC" of the data type TCON_QDN_SEC is defined.

Data_block_1

	Name	Data type	Start value	Comment
1	Static			
2	DNS Connection SEC	TCON_QDN_SEC		
3	ConnPara	TCON_QDN		parameter of the TCP connection
4	InterfaceId	HW_ANY	0	not relevant
5	ID	CONN_OUC	5	connection reference / identifier
6	ConnectionType	Byte	11	type of connection: 16#0B=11=TCP/IP, 16#13
7	ActiveEstablished	Bool	true	active/passive connection establishment
8	RemoteQDN	String[254]	'plc_1.factory127.'	fully or partially qualified domain name of rem
9	RemotePort	UInt	4000	remote UDP / TCP port number
10	LocalPort	UInt	0	local UDP / TCP port number
11	ActivateSecureConn	Bool	true	activate the security functionality of that conn
12	TLSServerReqClientCert	Bool	false	Just for server side: The TLS server requests a
13	ExtTLSCapabilities	Word	16#0	Bit 0: Just for client side: validate given IPv4 a
14	TLSServerCertRef	UDInt	7	for Server side: Reference to own X.509 V3 se
15	TLSClientCertRef	UDInt	0	for Client side: add id of own X.509 V3 client c

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the fully qualified domain name (FQDN) of the TLS server for "RemoteQDN".
4. Set the parameters for Secure Communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "ExtTLSCapabilities": When you enter the value 1, the client validates the subjectAlternateName in the X.509-V3 certificate of the server to check the identity of the server. The certificates are checked when the connection is established.
   - "TLSServerCertRef": ID of the X.509-V3 certificate (usually a CA certificate) that is used by the TLS client to validate the TLS server authentication. If this parameter is 0, the TLS client uses all (CA) certificates currently loaded in the client certificate store to validate the server authentication. The certificate ID of the CA certificate is entered in the TLS server.

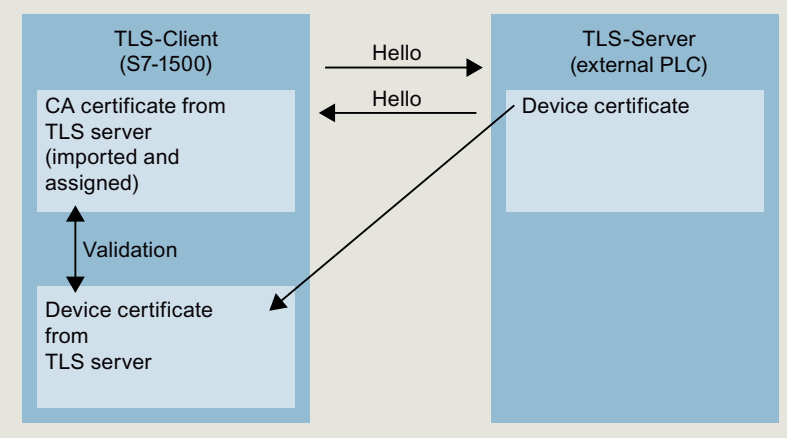

   - "TLSClientCertRef": ID of the own X.509-V3 certificate.

5. Create a TCON instruction in the program editor.
6. Interconnect the CONNECT parameter of the TCON instruction with the tag of the data type TCON_QDN_SEC.
   In the example below, the CONNECT parameter of the TCON instruction is interconnected with the tag "DNS connectionSEC" (data type TCON_QDN_SEC).

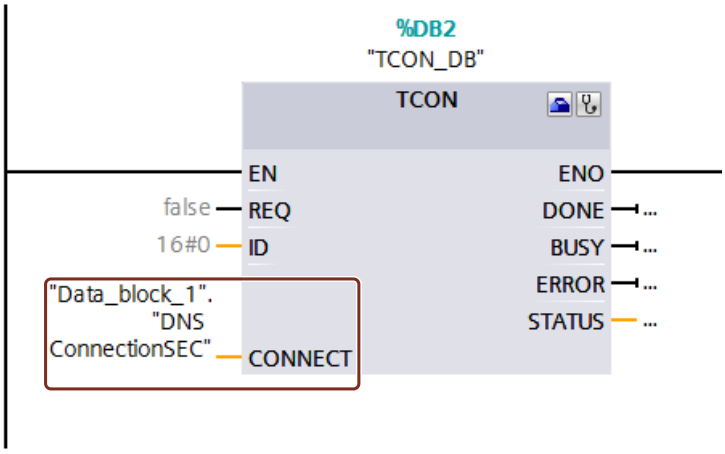

See also

Connection parameters in accordance with TCON_QDN_SEC (Page 113)

Examples for the management of certificates (Page 314)

Secure OUC from an S7-1500 CPU as TLS server to an external PLC (TLS client)

The following section describes how you can set up a Secure Open User Communication over TCP from one S7-1500 CPU as TLS server to a TLS client.

Setting up a secure TCP connection via the domain name of the communication partner

S7-1500 CPUs as of firmware version V2 support Secure Communication with addressing via a Domain Name System (DNS).

In order to establish secure TCP communication via the domain name you have to create a data block yourself with the system data type TCON_QDN_SEC, carry out the parameter assignment and call it directly at the instruction. The TCON instruction supports the system data type TCON_QDN_SEC.

Requirements:

- At least one DNS server exists in your network.
- You have configured at least one DNS server for the S7-1500 CPU.
- TLS client and TLS server have all the required certificates.

To set up a secure TCP connection to a TLS client, follow these steps:

1. Create a global data block in the project tree.
2. In the global data block, define a tag of the data type TCON_QDN_SEC.
   The example below shows the global data block "Data_block_1" in which the tag "DNS ConnectionSEC" of the data type TCON_QDN_SEC is defined.

Data_block_1

	Name	Data type	Start value	Comment
1	Static			
2	DNS Connection SEC2	TCON_QDN_SEC		
3	ConnPara	TCON_QDN		parameter of the TCP connection
4	InterfaceId	HW_ANY	0	not relevant
5	ID	CONN_OUC	8	connection reference / identifier
6	ConnectionType	Byte	11	type of connection: 16#0B=11=TCP/IP, 16#13
7	ActiveEstablished	Bool	false	active/passive connection establishment
8	RemoteQDN	String[254]	''	fully or partially qualified domain name of rem
9	RemotePort	UInt	0	remote UDP / TCP port number
10	LocalPort	UInt	2010	local UDP / TCP port number
11	ActivateSecureConn	Bool	true	activate the security functionality of that conn
12	TLSServerReqClientCert	Bool	false	Just for server side: The TLS server requests a
13	ExtTLSCapabilities	Word	16#0	Bit 0: Just for client side: validate given IPv4 a
14	TLSServerCertRef	UDInt	5	for Server side: Reference to own X.509 V3 se
15	TLSClientCertRef	UDInt	0	for Client side: add id of own X.509 V3 client c

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the local ID of the TCP connection in "ID".

4. Set the parameters for Secure Communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerReqClientCert": Request for an X.509-V3 certificate from the TLS client
   - "TLSServerCertRef": ID of the own X.509-V3 certificate.

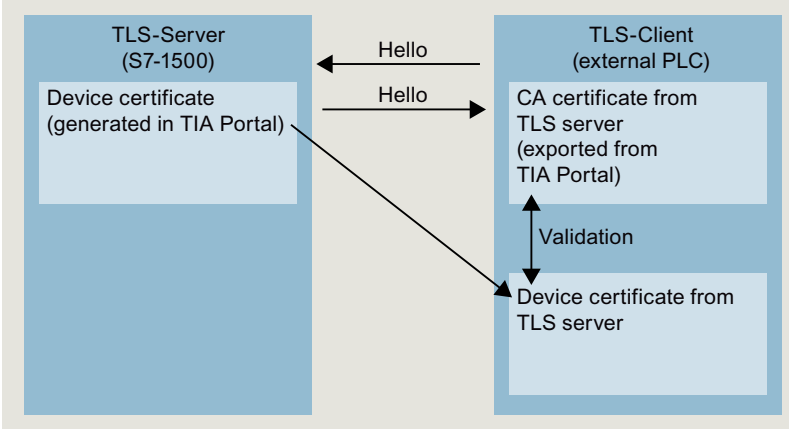

   - "TLSClientCertRef": ID of the X.509-V3 certificate (or a group of X.509-V3 certificates) that is used by the TLS server to validate TLS client authentication. If this parameter is 0, the TLS server uses all (CA) certificates currently loaded in the server certificate store to validate the client authentication.
5. Create a TCON instruction in the program editor.
6. Interconnect the CONNECT parameter of the TCON instruction with the tag of the data type TCON_QDN_SEC.
   In the example below, the CONNECT parameter of the TCON instruction is interconnected with the tag "DNS connectionSEC" (data type TCON_QDN_SEC).

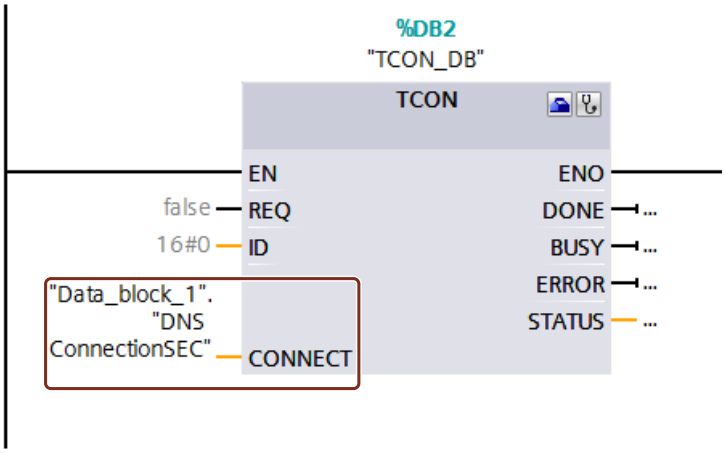

See also

Connection parameters in accordance with TCON_QDN_SEC (Page 113)

Examples for the management of certificates (Page 314)

Secure OUC via CP interface

The following sections describes the particular points to be taken into consideration in the case of Secure Open User Communication via a CP interface. At least one station is an S7-1500 station with the following modules:

- S7-1500 CPU as of firmware version V2.0 (with the exception of S7-1500 Software Controller)
- CP 1543-1 as of firmware version V2.0 or CP 1543SP-1 as firmware version V1.0

The CP acts in an S7-1500 station as a TLS client (active connection establishment) or a TLS server (passive connection establishment).

The fundamental procedure and the concept for using secure communication via a CP interface is similar to that of secure communication via the interfaces of the S7-1500 CPUs. Essentially, you have to assign the certificates to the CPU in the role of a TLS server or TLS client and not to the CPU. Other rules and procedures therefore apply. These are described below.

Handling certificates for CPs

The following applies in general: You have to be logged on at the certificate manager in the global security settings. The generation of self-signed certificates also requires logon for the global security settings. You have to have sufficient rights as a user (administrator or user with the "Standard" role with the right to "Configure security").

The starting point for the generation or assignment of certificates at the CP is the section "Security > Security properties". In this section, you log on for the global security settings.

Procedure:

1. In the network view of STEP 7, mark the CP and select the section "Security > Security properties" in the Inspector window.
2. Click on the "User logon" button.
3. Log on using your user name and password.
4. Enable the "Activate security functions" option.
   The security properties are initialized.
5. Click in the first line of the "Device certificates" table to generate a new certificate or select an existing device certificate.
6. If the communication partner is also an S7-1500 station, you also have to assign a device certificate to the communication partner with STEP 7 as described here or for the S7-1500 CPU.

Example: Setting up a secure TCP connection between two S7-1500 CPUs via CP interfaces

In order to establish secure TCP communication between two S7-1500 CPs, you have to create a data block with the system data type TCON_IPv4_SEC in each CPU, carry out the parameter assignment and call it directly at the instruction.

Requirements:

- Both S7 1500 CPUs have at least firmware version V2.0. If you use the CP 1543SP-1: Firmware version as of V1.0.
- Both CPs (for example CP 1543-1) must have at least firmware version V2.0
- TLS client and TLS server have all the required certificates.
  - A device certificate (end-entity certificate) for the CP must be generated and be located in the certificate memory of the CP. If a communication partner is an external device (for example an MES or ERP system), a device certificate also has to exist for this device.
  - The root certificate (CA certificate) with which the device certificate of the communication partner is signed must also be located in the certificate memory of the CP or in the certificate memory of the external device. If you use intermediate certificates, you have to ensure that the complete certificate path exists in the validating device. A device uses these certificates to validate the device certificate of the communication partner.
- The communication partner must always be addressed via its IPv4 address, not via its domain name.

The following figure shows the different certificates in the devices for the case that both communication partners communicate via a CP 1543-1. In addition, the figure shows the transfer of the device certificates during establishment of the connection ("Hello").

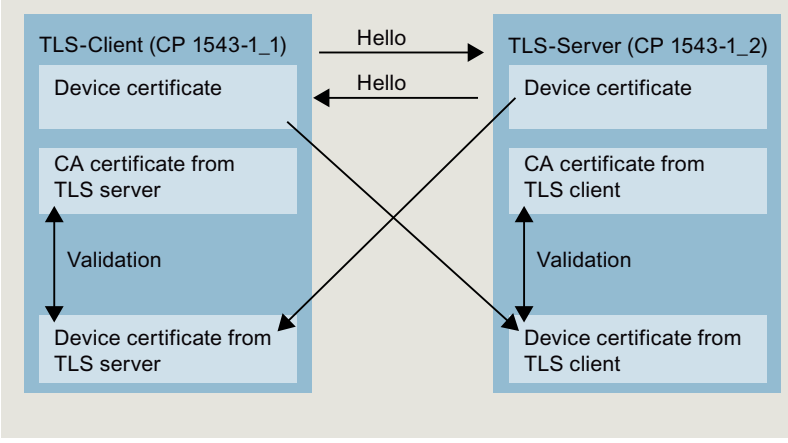

Settings at the TLS client

To set up a secure TCP connection in the TLS client, follow these steps:

1. Create a global data block in the project tree.
2. In the global data block, define a tag of the data type TCON_IP_V4_SEC. To do so, enter the string "TCON_IP_V4_SEC" in the "Data type" field.
   The example below shows the global data block "Data_block_1" in which the tag "SEC connection 1 TLS Client" of the data type TCON_IP_V4_SEC is defined.
   The Interface ID has the value of the HW identifier of the IE interface of the local CP (TLS client).

Data_block_1

	Name	Data type	Start value	Comment
1	Static			
2	SEC connection 1 TLS-Client	TCON_IP_V4_SEC		
3	ConnPara	TCON_IP_v4		parameter of the TCP connection
4	InterfaceId	HW_ANY	258	HW-identifier of IE-interface submodule
5	ID	CONN_OUC	10	connection reference / identifier
6	ConnectionType	Byte	11	type of connetion: 11=TCP/IP, 19=UDP (17=TCP/IP)
7	ActiveEstablished	Bool	true	active/passive connection establishment
8	RemoteAddress	IP_V4		remote IP address (IPv4)
9	ADDR	Array[1..4] of Byte		IPv4 address
10	ADDR[1]	Byte	192	IPv4 address
11	ADDR[2]	Byte	168	IPv4 address
12	ADDR[3]	Byte	1	IPv4 address
13	ADDR[4]	Byte	100	IPv4 address
14	RemotePort	UInt	4711	remote UDP/TCP port number
15	LocalPort	UInt	4711	local UDP/TCP port number
16	ActivateSecureConn	Bool	true	activate the security functionality of that connection in general
17	TLSServerReqClientCert	Bool	false	Just for server side: The TLS server requests a client certificate
18	ExtTLSCapabilities	Word	16#0	Bit 0: Just for client side: validate given IPv4 address against th
19	TLSServerCertRef	UDInt	1	for Server side: Reference to own X.509 V3 server certificate; fo
20	TLSClientCertRef	UDInt	5	for Client side: add id of own X.509 V3 client certificate; for Serv

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the IPv4 address of the TLS server for "RemoteAddress".
4. Set the parameters for Secure Communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerCertRef": Enter the value 2 (reference to the CA certificate of the TIA Portal project (SHA256) or the value 1 (reference to the CA certificate of the TIA Portal project (SHA1)).
   - "TLSClientCertRef": ID of the own X.509-V3 certificate.
5. Create a TCON instruction in the program editor.
6. Interconnect the CONNECT parameter of the TCON instruction with the tag of the data type TCON_IP_V4_SEC.

Settings at the TLS server

To set up a secure TCP connection in the TLS server, follow these steps:

1. Create a global data block in the project tree.
2. In the global data block, define a tag of the data type TCON_IP_V4_SEC.
   The example below shows the global data block "Data_block_1" in which the tag "SEC connection 1 TLS Server" of the data type TCON_IP_V4_SEC is defined.
   The interface ID has the value of the HW identifier of the IE interface of the local CP (TLS server).

Data_block_1

	Name	Data type	Start value	Comment
1	Static			
2	SEC connection 1 TLS-Server	TCON_IP_V4_SEC		
3	ConnPara	TCON_IP_v4		parameter of the TCP connection
4	InterfaceId	HW_ANY	260	HW-identifier of IE-interface submodule
5	ID	CONN_OUC	10	connection reference / identifier
6	ConnectionType	Byte	11	type of connetion: 11=TCP/IP, 19=UDP (17=TCP/IP)
7	ActiveEstablished	Bool	false	active/passive connection establishment
8	RemoteAddress	IP_V4		remote IP address (IPv4)
9	ADDR	Array[1..4] of Byte		IPv4 address
10	ADDR[1]	Byte	192	IPv4 address
11	ADDR[2]	Byte	168	IPv4 address
12	ADDR[3]	Byte	1	IPv4 address
13	ADDR[4]	Byte	10	IPv4 address
14	RemotePort	UInt	4711	remote UDP/TCP port number
15	LocalPort	UInt	4711	local UDP/TCP port number
16	ActivateSecureConn	Bool	true	activate the security functionality of that connection in general
17	TLSServerReqClientCert	Bool	true	Just for server side: The TLS server requests a client certificate
18	ExtTLSCapabilities	Word	16#0	Bit 0: Just for client side: validate given IPv4 address against the
19	TLSServerCertRef	UDInt	6	for Server side: Reference to own X.509 V3 server certificate; fo
20	TLSClientCertRef	UDInt	1	for Client side: add id of own X.509 V3 client certificate; for Serv

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the IPv4 address of the TLS client for "RemoteAddress".
4. Set the parameters for Secure Communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerReqClientCert ": Request for an X.509-V3 certificate from the TLS client Enter the value "true".
   - "TLSServerCertRef": ID of the own X.509-V3 certificate.
   - "TLSClientCertRef": Enter the value 2 (reference to the CA certificate of the TIA Portal project (SHA256) or the value 1 (reference to the CA certificate of the TIA Portal project (SHA1)).
5. Create a TCON instruction in the program editor.
6. Interconnect the CONNECT parameter of the TCON instruction with the tag of the data type TCON_IP_V4_SEC.

Upload device as new station

When you upload a configuration with certificates and configured secure Open User Communication as a new station into your STEP 7 project, the certificates of the CP are not uploaded, in contrast to the certificates of the CPU. After the device has been loaded as a new station, no more certificates are contained in the corresponding tables of the CPs for the device certificates.

You have to perform configuration of certificates again after the upload. Otherwise, renewed loading of the configuration results in the certificates that originally exist in the CP being deleted so that secure communication does not function.

Secure OUC connections via CPU and CP interfaces - similarities

- Connection resources:
  No differences between OUC and secure OUC. A programmed secure OUC connection uses a connection resource just like an OUC connection, irrespective of which IE/ PROFINET interface communicates with the station.
- Connection diagnostics:
  No differences between OUC and secure OUC connection diagnostics.
- Loading of projects with secure OUC connections into the CPU:
  Only possible in STOP of the CPU, if certificates are loaded as well.
  Recommendation: Load to device > Hardware and software. Reason: Ensuring the consistency between the program with secure OUC, hardware configuration and certificates.
  Certificates are loaded with the hardware configuration - therefore loading requires a stop of the CPU. The reloading of blocks that utilize further secure OUC connections is only possible in RUN if the certificates required for this purpose are already located on the module.

See also

Connection parameters according to TCON_IP_V4_SEC (Page 110)

Creating certificates (Page 985)

Examples for the management of certificates (Page 314)

Secure OUC via e-mail

The following section describes how to establish a secure connection (SNMP over TLS) to a mail server with the communication instruction TMAIL_C.

Setting up a secure connection to a mail server

In order to establish a secure communication to a mail server you have to create a data block yourself with one of the system data types TMAIL_V4_SEC, TMAIL_V6_SEC or TMAIL_QDN_SEC, carry out the parameter assignment and call it directly at the TMAIL_C instruction.

Requirements:

- S7-1500 CPU as of firmware version V2.0 with communication module CP 1543-1 as of firmware version V2.0
- ET 200SP CPU as of firmware version V2.0 with communication module CP 1542SP-1 (IRC) as of firmware version V1.0
- You have assigned all CA certificates of the mail server (TLS server) to the CP (TLS client) and have downloaded the configuration to the CPU.

To set up a secure mail server connection via the IPv4 address of the mail server, follow these steps:

1. Create a global data block in the project tree.
2. In the global data block, define a tag of the data type TMAIL_V4_SEC.
   The example below shows the global data block "MailConnDB" in which the tag "MailConnectionSEC" of the data type TMAIL_V4_SEC is defined.

MailConnDB

Name	Data type	Start value	Comment
Static			
MailConnectionSEC	TMail_V4_SEC		
InterfaceId	HW_ANY	"Local~CP_1543-1_1~Ethernet-Schnittstelle_1"	Use HW-identifier of the IE-interface to specify th
ID	CONN_OUC	100	connection reference / identifier
ConnectionType	Byte	16#20	type of connection 16#20=32=TMail_V4 or TMai
ActiveEstablished	Bool	true	active / passive connection establishment
WatchDogTime	Time	T#5000ms	watchdog time to monitor SMTP server associat
MailServerAddress	IP_V4		IPv4 address of mail server
ADDR	Array[1..4] of Byte		IPv4 address
ADDR[1]	Byte	144	IPv4 address
ADDR[2]	Byte	145	IPv4 address
ADDR[3]	Byte	2	IPv4 address
ADDR[4]	Byte	20	IPv4 address
UserName	String[254]	'myName'	user name which is necessary to login into the
PassWord	String[254]	'myPW'	user password which is necessary to login into
From	EMAIL_ADDR		source mail address
LocalPartPlusAt...	String[64]	'Mustermann@'	local part of e-mail address plus "@" sign
FullQualifiedD...	String[254]	'siemens.com'	full qualified domain name part of e-mail addre
RemotePort	UInt	587	remote TCP port number
ActivateSecureConn	Bool	TRUE	activate the security functionality of that conne
ExtTLSCapabilities	Byte	16#0	for further capability extensions of the TLS hand
TLSServerCertRef	UDInt	7	Reference to the X.509 V3 (CA-) certificate of the

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the IPv4 address of the mail server in "MailServerAddress".

4. Set the parameters for Secure Communication in the "Start value" column. For example, enter the certificate ID of the CA certificate of the communication partner in "TLSServerCertRef".
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerCertRef": Reference to the X.509 V3 (CA) certificate of the mail server which is used by the TLS client to validate the authentication of the mail server.

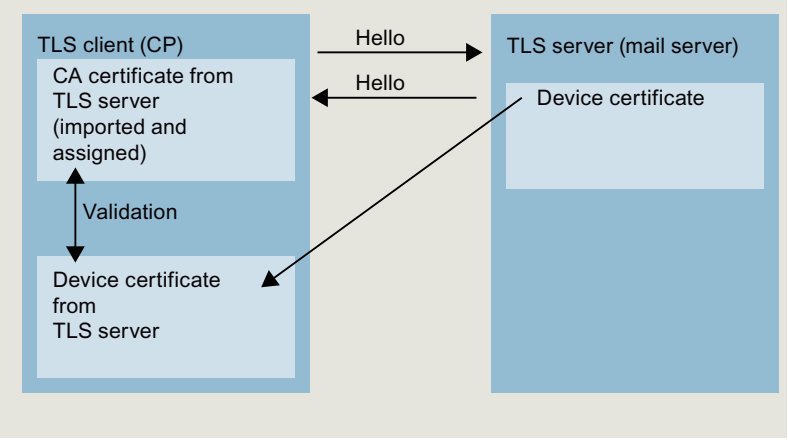

5. Create a TMAIL_C instruction in the program editor.
6. Interconnect the MAIL_ADDR_PARAM parameter of the TMAIL_C instruction with the tag of the data type TMAIL_V4_SEC.
   In the following example the MAIL_ADDR_PARAM parameter of the TMAIL_C instruction is interconnected with the "MAILConnectionSEC" tag (data type TMAIL_V4_SEC).

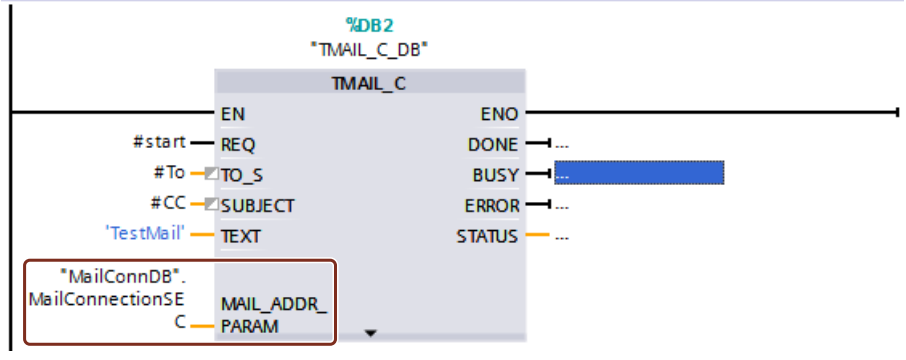

See also

Connection parameters in accordance with TCON_QDN_SEC (Page 113)

Examples for the management of certificates (Page 314)

1.3.5 Industrial Ethernet Security

1.3.5.1 Configuring security

General

Supported devices

Supported devices

The security functions described in this help section are supported y the following products;

- SCALANCE S-600 V3:
  - S602 V2*/V3/V4
  - S612 V2*/V3/V4
  - S613 V2*
  - S623 V3/V4
  - S627-2M V4
- SOFTNET Security Client:
  - SOFTNET Security Client V4
- S7 CPs / TIM: CP 343-1 GX31 Advanced, CP 443-1 GX30 Advanced, CP 443-1 OPC UA, CP 1543-1, CP 1542SP-1 IRC, CP 1543SP-1, CP 1243-1 BX30, CP 1243-1 PCC, CP 1242-7 KX31, CP 1243-7 LTE, CP 1243-8 IRC, TIM 4R-IE, TIM 4R-IE DNP3, TIM 1531 IRC
- PC CP: CP 1628
- Mobile wireless router: SCALANCE M875

* SCALANCE S V2 devices cannot be inserted in a STEP 7 project. SCALANCE S V2 devices from projects of older STEP 7 versions are simply displayed with their properties. The properties of these devices can, however, not be changed. Downloading and compilation and diagnostics of SCALANCE S V2 devices is also not possible. All the functions described for SCALANCE S devices relate to the firmware versions as of V3.

When opening a STEP 7 project with existing SCALANCE S V2 devices global firewall rules can be created from the existing local firewall rules for the device.

General terminology "security module"

In this section of the information system the name "security module" is used instead of the product names named above. You will find the functional differences between the products in the Siemens Industry Mall.

Use of the terms "interface" and "port"

In this documentation, the ports of SCALANCE S modules are named as follows:

- "External interface": The external port of the SCALANCE S602 / S612 / S613 / S623 or an external port of the SCALANCE S627-2M (marked red)
- "Internal interface": The internal port of the SCALANCE S602 / S612 / S613 / S623 or an internal port of the SCALANCE S627-2M (marked green)
- "DMZ interface": The DMZ port of the SCALANCE S623 / S627-2M (marked yellow)

The term "port" itself is used when the focus of interest is a special port of an interface.

Structure of this section of the help system

Topics that are relevant to several module types can be found in the "General" section. Information that is only relevant to a certain module type can be found in the sections relating specifically to these module types.

Overview - Scope of performance and method of operation

General use of the term "STEP 7"

Configuration of security functions is supported as of STEP 7 V12. For this reason, in this section of the information system, the name "STEP 7" will be used for all versions of STEP 7 V12 or higher.

Scope of performance

You can use the following security functions in STEP 7:

- Configuration of the security modules
- Creating VPN configuration data for SOFTNET Security Client V4
- Creating VPN configuration data for SCALANCE M875
- Creating VPN configuration files for VPN devices from third-party manufacturers
- Test and diagnostics functions, status displays

Offline configuration view and online diagnostics view

The security functions are configured in two views:

- Offline configuration view
  In the offline configuration view, you create the configuration data. Prior to downloading, there must already be a connection to the security modules.
- Online diagnostics view
  The online/diagnostics view is used for diagnostics of the security module and, among other things, allows you to run firmware updates.

How it works - security and consistency

- Access only for authorized users
  The security functions of every project are only available when project protection is active. This protects the security functions from accidental or unauthorized changes.
- Consistent project data
  Consistency checks are running even while you make the entries in the dialogs. In addition, cross-dialog, project-wide consistency checks are carried out.
  Only consistent project data can be downloaded to the security modules.
- Protecting project data by encryption
  The project and configuration data relevant to security are protected by encryption. Depending on the security module, data can be stored in the project and/or on the C-PLUG.

User interface for security functions

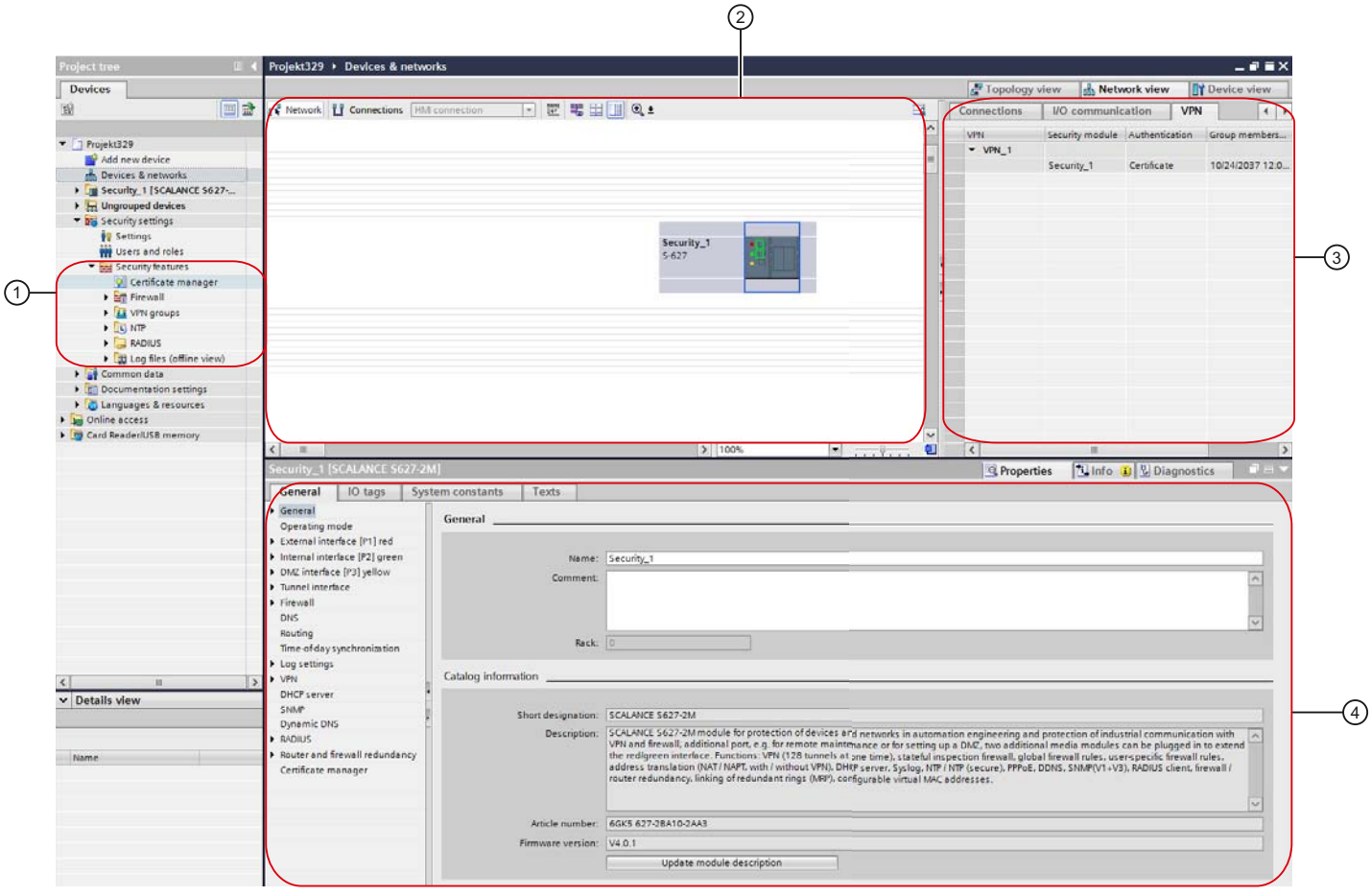

① Security functions

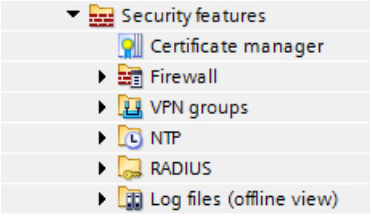

The global security functions are located in the project navigation. The global security functions can be configured independently of the module and subsequently assigned to individual security modules as required. Changes to the global security functions must be loaded on all security modules involved. This also applies to the settings of redundancy relationships.

To display the global security functions completely in the project navigation the project protection must be activated and there must be at least one security module in the project. If the first security module is a CP, the local security settings of the CP must also be activated.

The following main folders and entries are available in the global security functions:

- Certificate manager
  In the certificate manager, you see an overview of all the certificates used in the project. You can, for example, import new certificates as well as export, renew or replace existing certificates.

- Firewall
  Under the "Firewall" entry, you can define global IP and MAC firewall rule sets and user-specific IP rule sets (SCALANCE S modules only) and assign security modules. Global firewall rule sets can also be exported from STEP 7 and imported into STEP 7. IP and MAC service definitions are used to define the IP and MAC firewall rules compactly and clearly.
- VPN groups
  All created VPN groups are contained in this folder. You can create new VPN groups here and assign security modules to these VPN groups. You can also adapt VPN group properties of VPN groups that have already been created.
- NTP
  Here, you can create NTP servers and assign them to one or more security modules. This ensures that time synchronization is performed through the assigned NTP server. Unsecured NTP servers can only be configured in the local security settings.
- RADIUS
  Here, you can create RADIUS servers and assign them to one or more security modules. With this, you ensure that authentication queries from users who log on to the selected security module to activate user-specific IP rule sets are forwarded to the assigned RADIUS server.
- Log files (offline view)
  Opens logged system, audit and packet filter events that you saved as a file in the online diagnostics.

② Work area

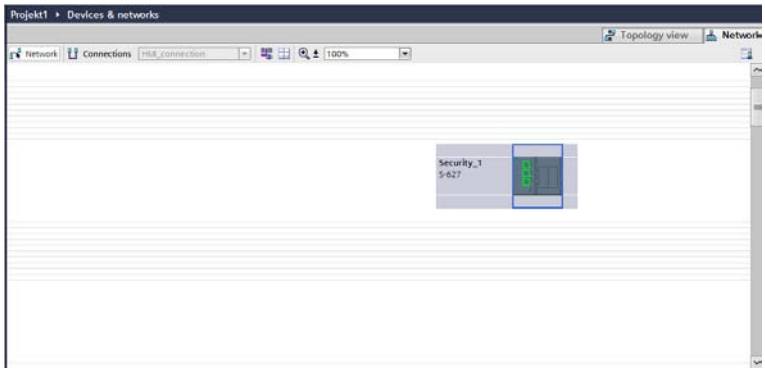

Once you have selected a SCALANCE M module or a security module in the work area, you can configure its local security settings in "Properties" > "General". If the selected object is in a VPN group, related information is displayed in the VPN tab.

③ VPN tab

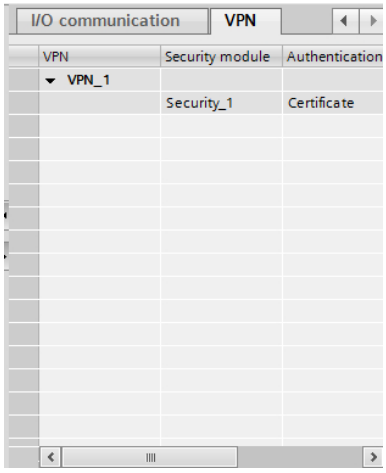

This tab displays information about all the VPN groups to which the security module that was selected in the working area belongs. Information about the respective participants of a VPN group can be displayed and hidden.

④ Local security settings

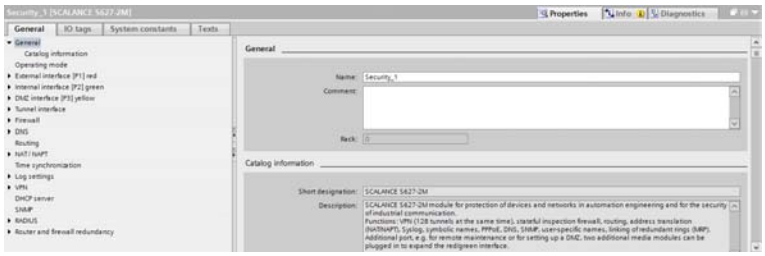

Local security settings are configured for a specific security module / SCALANCE M module / security device. After one of these objects has been selected in the working area, its local security settings are available in the Inspector window under "Properties" > "General". To display the local security settings, project protection must must be activated.

For CPs in the Inspector window, the "Activate security features" check box must be activated n the "Properties" > "General" tab, "Security" entry. The local security settings are then displayed below the "Security" entry. When the check box is selected, the some of the CP settings (assuming they were enabled) are migrated automatically to the local security settings.

Depending on the particular security module, additional security functions are also available such as NTP (secure), SNMPv3, FTPS.

In addition, firewall rules that enable a connection to be established are created automatically for configured connections. Log settings are available to record blocked packets.

Secure and non-secure configuration areas

The user interface can be divided into secure and non-secure configuration areas.

The secure areas are areas in which configuration is possible only after activating project protection. These areas are encrypted and therefore only available to persons authorized in the user management even if the project is accessible to a wider circle of people.

Functions from the non-secure areas, on the other hand, can be configured without activating project protection. The correctness of the settings must be checked before downloading the project to the plant components if a wider circle of people can make modifications to the project.

Below, you will find a list of the configuration areas of the user interface showing which areas are secure and which are non-secure. To some extent, this depends on the security module for which the configuration is created.

- All settings from the global security functions are secure.
- Secure and non-secure configuration areas for SCALANCE S modules:
  - The settings under the entry "General" in the local security settings are non-secure.
  - Higher-level settings (e.g. MRP settings such as MRP manager etc.) that are not configured on the security module itself but may affect the security module are non-secure. This does not relate to the global security functions.
  - The other settings are protected.
- Secure and non-secure configuration areas for CPs:
  - All settings outside the "Security" entry are non-secure.
  - Higher-level settings (e.g. MRP settings such as MRP manager, PROFINET settings, connections etc.) that are not configured on the security module itself but may affect the security module are non-secure. This does not relate to the global security functions.
  - All the settings for the interfaces and ports, in particular IP addresses, are non-secure.
  - All settings below the "Security" entry are secure.

Restrictions in the use of libraries

If a master copy is created in the global library for a security module, the configuration of this security module is lost in the master copy. You need to log in again if you use a master copy.

If a master copy is created in the project library for a security module, the configuration of this security module is retained, apart from the cross-module information for VPN and redundancy relationships.

IP addresses for configurable servers

When configuring servers, the (alias) IP addresses of the security module must not be used. For all configurable servers, there is a consistency check when the IP address is assigned.

Running a consistency check

Overview

The following consistency checks are available:

- Local consistency checks
- Project-wide consistency checks

In the individual dialog descriptions in this help system, the rules you need to take into account for your entries are listed under the keyword "Consistency check".

Local consistency checks

A consistency check is local when it can be performed directly within a dialog. With the following actions, local consistency checks are made:

- After exiting a box
- After exiting a row in a table
- When you confirm a dialog with OK.

Project-wide consistency checks

Project-wide consistency checks provide you with information indicating whether or not project data is correctly configured. With the following actions, there is a consistency check through the entire project:

- When compiling a configuration
- When downloading a configuration

Note

You can only download configured data to a security module if the project-wide consistency check for the security module was successful.

Replacing a security module

Module-specific function

SCALANCE S modules as of V3 can be replaced by SCALANCE S modules as of V3, refer to the section:
Replacing a security module (Page 472)

In addition to this CP 1543-1 and TIM 1531 IRC can be replaced by modules with higher firmware versions. To do this, the local security settings of the modules must be activated.

Requirement

To be able to replace security modules, their module descriptions must be up to date. To update the module description of security modules, follow the steps below:

1. Select the security module to be edited.
2. In the local security settings, click on the entry "General" > "Catalog information".
3. Click the "Update module description" button.

How to access this function

1. Select the security module to be edited in the topology or network view.
2. Right-click on the security module and select the shortcut menu command "Change device...".

Managing certificates

Overview of certificates

Certificate management

The certificate manager in the global security functions contains an overview of all the certificates used in the project, for example, CA certificates with information about the certificate owner, issuer, validity, use and the existence of a private key.

A CA certificate is a certificate issued by a certificate authority from which the device certificates are derived. The device certificates include SSL certificates. SSL certificates are required for authentication during secure communication between a network node and a security module. Further device certificates include the VPN group certificates of security modules located in VPN groups and OPC UA client/server certificates of CP 443-1 OPC UA modules.
Certificate authorities can be:

- The STEP 7 project itself. If the "certificate owner" and "issuer" are the same, this is a self-signed certificate; in other words, issued by the STEP 7 project.
- A higher ranking (commercial) certificate authority. These third-party certificates are external to the project and are imported and stored in the certificate store of STEP 7.

Certificates created by one of the two certificate authorities always have a private key so that the device certificates can be derived from them.

Every certificate is assigned an ID via which the certificate can be referenced in the program blocks. The IDs are generated consecutively when certificates are created. Due to the system some IDs are not used.

Note

Downloading the configuration

After replacing or renewing certificates, the configuration must be downloaded to the relevant security modules.

After replacing or renewing CA certificates, the configuration must be downloaded to all security modules.

Note

Current date and current time of day on the security modules

When using secure communication (for example, HTTPS, VPN...), make sure that the security modules involved have the current time of day and the current date. Otherwise the certificates used are not evaluated as valid and the secure communication does not work.

Note

Character set of certificates

Certificates are always stored in the ASCII character set. If a certificate contains characters that do not exist in the ASCII character set, the certificate is stored with the UTF-8 character set. This is not possible for SCALANCE S V2 modules.

Global certificate manager

The following functions are available in the certificate manager in the global security functions:

- Import of new certificates and certificate authorities as well as certificate chains.
- Import of SSL certificates (CP 343-1 Adv. / CP 443-1 Adv. only), e.g. for FTP communication.
- Export of the certificates and certificate authorities used in the project.
- Assignment of a certificate to one or more modules, see section Assigning certificates (Page 360).
- For information on certificates, for example when the validity has expired or for updating signatures, refer to the section Renewing certificates (Page 358).
- For information on replacing certificates, for example for updating signatures, refer to the section Replacing certificates (Page 359).
- Addition of trusted certificates and certificate authorities.
- Deleting manually imported certificates.

Double-click on the "Certificate manager" entry in the global security functions.

In the individual tabs, you have the following commands available in the shortcut menu:

Command	Meaning
Import	Import of certificates. The following formats are permitted: *.der: DER-coded certificate (without private key) *.cer: DER-coded certificate (without private key) *.crt: PEM-coded certificate (without private key) *.pem: PEM-coded certificate (without private key) *.p12: In addition to the device or CA certificate, the format can also contain all certificates of the certification path as well as the associated private key.
Export	Export of certificates. The following formats are permitted: *.der: DER-coded certificate (without private key) *.cer: DER-coded certificate (without private key) *.crt: PEM-coded certificate (without private key) *.pem: PEM-coded certificate (without private key) You can also select the following formats for CA and device certificate: *.crl: Certificate Revocation List, DER-coded *.p12 PKSC12-Archive: Certificate with associated private key *.p12 PKSC12-Certificate-Chain: Certificate with associated private key and all certificates of the certification path
Displays	Opens the certificate dialog of Windows where you see an overview of all certificate data. For the certificate to be displayed as valid, install it via the "Install certificate..." button of the "General" tab in the Windows certificate store. Appropriate Windows user rights are required to install the certificate.
Renew (only in the "Certificate authority (CA)" and "Device certificates" tabs)	Opens the "Renew certificate" dialog in which you can have a new certificate created by STEP 7 when necessary, for example with compromised certificates.
Replace	Opens the "Replace certificate" dialog in which you can replace an existing certificate with a new one.
Assign	Opens the "Assign certificate" dialog in which you can assign an existing certificate to one or more devices.
Delete	Deletes a certificate in the "Trusted certificates and root certification authorities" tab.

Importing and renewing *.p12 certificates

If you import or renew a file of the type *.p12, the system detects automatically which certificates are part of the file.

For example, this may be a certificate chain which consists of the following contents:

- Device certificate A which was issued by certificate authority 1 with private key
- CA certificate B of the certificate authority 1 without private key
- CA certificate C of the certificate authority 2 which issued certificate B without private key

When the file is imported, the following certificates are stored:

- Device certificate A in the "Device certificates" tab; private key: Yes
- CA certificate B of the certificate authority in the tab "Certificate authority (CA)"; private key: No
- CA certificate C of certificate authority 2 in the tab "Certificate authority (CA)"; private key: No

To be able to generate new device certificates with renewal as certificate authority 1, proceed as follows:

- Import a certificate of certificate authority 1 with private key.
  Result:
  - Certificate B of the certificate authority in the "Certificate authority (CA)" tab is not overwritten because it has already been imported.
  - The status of the private key is switched from "No" to "Yes".
  - New device certificates can now be signed during renewal.

Local certificate manager

Certificates that were imported via the certificate manager in the global security functions are not automatically assigned to the corresponding modules. Imported certificates must be included in the list of trustworthy partner certificates manually via the entry "Certificate manager" in the local security settings. Certificates generated by STEP 7 such as SSL certificates or VPN group certificates are automatically assigned to the corresponding modules and do not need to be assigned using the local security settings.

Before certificates can be referenced in program blocks for "Secure Communication", these certificates must be assigned to the security module as device certificates via the local certificate manager. After selecting a table cell, you can either create a new certificate using the "Add" button or use an existing certificate of the project using the check mark symbol. For access to existing certificates of the project and for signing a new certificate by a certification authority, the check box "Use global security functions for the certificate manager" must be activated. To make the assigned certificate known to the partner module, this certificate must be entered in the list of trustworthy partner certificates after import.

Certificate authorities

"CA" tab

The following certification authorities are displayed

- CA certificates of a project: When you create a new project, CA certificates are generated for the project that use different hash algorithms for the certificate signature. CA certificates for projects are always loaded on the security modules belonging to them. The SSL certificates and OPC UA client/server certificates for the individual security modules are derived from the CA certificates.
- CA group certificates: When you create a new VPN group, a CA certificate is generated for the VPN group. The VPN group certificates of VPN group subscribers are derived from this certificate..

Device certificates

"Device certificates" tab

Display of the device-specific certificates generated by the certification authorities for modules. These include:

- SSL certificate of a module: An SSL certificate that is derived from a CA certificate of the project is generated for each module that you create. SSL certificates are used for authentication during secure communication between PG/PC and module when downloading the configuration.
- OPC UA client/server certificate of the module: Depending on the configured OPC UA client/server function of the CP 443-1 OPC UA, an OPC UA client/server certificate is generated that is used for authentication with the relevant communications partner.
- VPN group certificate of a module: A VPN group certificate is also generated for each module for each VPN group in which it is located.

Trusted certificates and root certification authorities

"Trusted root certification authorities" tab

Display of the third-party certificates imported into STEP 7. Server certificates can be imported for example from external FTP servers, project certificates from other STEP 7 projects and certificates required for the external services such as dyn.DNS.

After the import, certificates must be assigned to the corresponding modules using the local security settings. For further information, refer to the section Overview of certificates (Page 354).

Renewing certificates

Meaning

In this dialog, you renew CA certificates and device certificates.

How to access this function

1. Right-click on a list entry in the certificate manager.
2. Select the "Renew" entry in the shortcut menu.
3. Decide whether or not the new certificate will be self-signed or signed by a certificate authority.

4. If the certificate is to be signed by a certificate authority, select the certificate authority to be used with the "Select" button. Only certificate authorities stored in the certificate store of the current project can be selected. This can also include certificate chains. See section Overview of certificates (Page 354).
5. Depending on the certificate, enter the following parameters in the corresponding input fields:

Parameter	Meaning
Common name of subject	Holder of the certificate. The common name of subject can contain no more than 35 characters. The displayed ID is not included in this count.
Signature	Algorithm used for signing the certificate.
Valid from	The validity period of the certificate
Valid to	
Usage	Value that is entered in the "KeyUsage" and "ExtendedKeyUsage" extensions of the standard X. 509 V3 for digital certificates.
Subject Alternative Name (SAN)	Depending on the certificate to be renewed, the alternative name of the certificate owner is displayed with an adaptable name type and value.

Example: Updating the device certificate

Renewing certificates is, for example, sensible with device certificates whose signature should be updated from SHA-1 to SHA-256. To do this, follow the steps outlined below:

1. Select the device certificate to be renewed.
2. Select the "Renew" shortcut menu command.
3. In the "Renew certificate" dialog click the "Select" button and select a certification authority (CA) with a SHA-256 signature.
   In the certificate parameters of the "Renew certificate" dialog SHA-256 is now displayed as the signature of the device certificate.

Replacing certificates

Meaning

In the "Replace certificate" dialog replace an existing certificate with a new one.

Please note that a CA certificate can only be replaced by a CA certificate and a device certificate can only be replaced by a device certificate.

How to access this function

1. Right-click on the list entry of a certificate.
2. Select the "Replace" entry in the shortcut menu.
3. The "Replace certificate" dialog opens.

All certificates listed in the "Certificates involved" table are derived once again. This means that the CA group certificate of an already configured VPN group can be replaced in the project

by the CA group certificate of a different project. The VPN group certificates for the VPN group members are therefore derived from the same CA group certificate in both projects.

After making changes in the certificate manager, download the configuration to all security modules affected.

Assigning certificates

Meaning

In the "Assign certificate to module" dialog, you assign an existing certificate to one or more modules.

How to access this function

1. Right-click on the list entry of a certificate.
2. Select the "Assign" command from the shortcut menu.
3. The "Assign certificate to module" dialog opens.
4. Select one or more modules to which the certificate should be assigned as trusted certificate.

"Device certificates" tab

In the "Device certificates" tab, select in addition whether the certificate should be assigned as device certificate or as trusted certificate. If the certificate has already been assigned, the column is highlighted in color.

A certificate can be assigned to modules as device certificate and to others as trusted certificate.

Special features of user management for security functions

Access to security functions

You can only access security functions after activating project protection and after a user has successfully logged on. The logged on user must also have the required engineering and runtime rights.

You will find basic information on user management in STEP 7 in the "Using user management" section of the information system.

Engineering rights for security modules

These engineering rights allow diagnostics and configuration of security modules:

- Security: Security modules with read rights: Display of the global security functions and the local security settings of the security modules.
- Security: Security modules with write rights: Configuration of the global security functions and the local security settings of the security modules and perform diagnostics of the security modules.

Roles with runtime rights for security modules

As default, the system-defined roles with the prefix "NET" are assigned runtime rights for security modules. Runtime rights provide the rights to execute certain tasks on security modules, for example for diagnostics or updating firmware. To be able to perform these tasks, you must assign your user a suitable system-defined role with the prefix "NET" or a user-defined role with the relevant runtime rights. Just as with engineering rights you cannot change runtime rights for system-defined roles. For user-defined roles you can change the runtime rights as required. You can specify the runtime rights of a user-defined role for each security module.

There must be at least one user assigned a role with full runtime rights for a security module. e.g. the system-defined role "NET Administrator".

The system-defined roles with the prefix "NET" are assigned engineering rights:

Right	NET Administrator	NET Standard	NET Diagnose
Open the project read only	x	x	x
Open the project with write rights	x	x	-
Edit security settings: "Settings" and "Users and roles"	x	-	-
Security: Security modules with read rights	x	x	x
Security: Security modules with write rights	x	x	-

Runtime rights for security modules

Depending on the security module, the following runtime rights are available for selection:

Table 1-5 Runtime rights CP 343-1 Advanced / CP 443-1 Advanced

Right	NET Administrator	NET Standard	NET Diagnose
FTP: Read files from the CP file system	x	x	x
FTP: Write files to the CP file system	x	x	-
Web server: Format CP file system *	x	-	-
Applets: Read values via PLC tags *	x	x	x
Applets: Read values via addresses *	x	x	x
Applets: Write values via addresses *	x	x	-
FTP: Read files (DBs) from the S7 CPU **	x	x	x
FTP: Write files (DBs) to the S7 CPU ***	x	x	-
Applets: Read status of the modules in the rack *	x	x	x

Right	NET Administrator	NET Standard	NET Diagnose
Applets: Read article numbers of the modules in the rack *	x	x	x
Applets: Write values via PLC tags *	x	x	-
SNMP: Read MIB-II	x	x	x
SNMP: Write MIB-II	x	x	-
SNMP: Read automation MIB	x	x	x
SNMP: Read LLDP MIB	x	x	x
TIA Portal: Perform diagnostics of the security module	x	x	x
Web server: Expand IP access control list *	x	-	-
Web server: Access Web diagnostics and CP file system	x	x	x
Web server: Send test e-mails *	x	x	x
Web server: Update firmware *	x	x	-
SNMP: Read MRP MIB	x	x	x
SNMP: Write MRP MIB	x	x	-
SNMP: Read SNMPv2 MIB	x	x	x
Web server: Load diagnostics texts later *	x	x	-

Table 1-6 Runtime rights CP 443-1 OPC UA

Right	NET Adminis-trator	NET Standard	NET Diagnose
SNMP: Read MIB-II	x	x	x
SNMP: Write MIB-II	x	x	-
SNMP: Read automation MIB	x	x	x
SNMP: Read LLDP MIB	x	x	x
TIA Portal: Perform diagnostics of the security module	x	x	x
Web server: Update firmware	x	x	-
SNMP: Read SNMPv2 MIB	x	x	x
Web server: Load diagnostics texts later	x	x	-
OPC UA: Read tags	x	x	x
OPC UA: Write tags	x	x	-
Web server: Diagnostics	x	x	x

Table 1-7 Runtime rights CP 1628

Right	NET Adminis-trator	NET Standard	NET Diagnose
SNMP: Read MIB-II	x	x	x
SNMP: Write MIB-II	x	x	-
SNMP: Read automation MIB	x	x	x
TIA Portal: Perform diagnostics of the security module	x	x	x
SNMP: Read SNMPv2 MIB	x	x	x

Table 1-8 Runtime rights SCALANCE S-600 V3

Right	NET Administrator	NET Standard	NET Diagnose
SNMP: MIB-II read	x	x	x
SNMP: MIB-II write	x	x	-
SNMP: Read automation MIB	x	x	x
SNMP: Read LLDP MIB	x	x	x
Download the configuration files	x	x	-
TIA Portal: Perform diagnostics of the security module	x	x	x
Web server: Update firmware	x	x	-
SNMP: MRP-MIB read (only for SCALANCE S627-2M)	x	x	x
SNMP: MRP-MIB write (only for SCALANCE S627-2M)	x	x	-
SNMP: Read SNMPv2 MIB	x	x	x

Table 1-9 Runtime rights CP 1543-1

Right	NET Administrator	NET Standard	NET Diagnose
FTP: Read files from the CP file system	x	x	x
FTP: Write files to the CP file system	x	x	-
FTP: Read files (DBs) from the S7 CPU **	x	x	x
FTP: Write files (DBs) to the S7 CPU ***	x	x	-
SNMP: Read MIB-II	x	x	x
SNMP: Write MIB-II	x	x	-
SNMP: Read automation MIB	x	x	x
SNMP: Read LLDP MIB	x	x	x
TIA Portal: Perform diagnostics of the security module	x	x	x
SNMP: Read SNMPv2 MIB	x	x	x
SNMP: Read IPv6 MIB	x	x	x

Table 1-10 Runtime rights CP 1542SP-1 IRC

Right	NET Administrator	NET Standard	NET Diagnose
SNMP: Read MIB-II	x	x	x
SNMP: Write MIB-II	x	x	-
SNMP: Read automation MIB	x	x	x
TIA Portal: Perform diagnostics of the security module	x	x	x
SNMP: Read SNMPv2 MIB	x	x	x
SNMP: Read IPv6 MIB	x	x	x
Use TeleService	x	-	-

Table 1-11 Runtime rights CP 1543SP-1

Right	NET Administrator	NET Standard	NET Diagnose
SNMP: Read MIB-II	x	x	x
SNMP: Write MIB-II	x	x	-
SNMP: Read automation MIB	x	x	x
SNMP: Read LLDP MIB	x	x	x
TIA Portal: Perform diagnostics of the security module	x	x	x
SNMP: Read SNMPv2 MIB	x	x	x
SNMP: Read IPv6 MIB	x	x	x

Table 1-12 Runtime rights CP 1243-1

Right	NET Administrator	NET Standard	NET Diagnose
SNMP: Read MIB-II	x	x	x
SNMP: Write MIB-II	x	x	-
SNMP: Read automation MIB	x	x	x
TIA Portal: Perform diagnostics of the security module	x	x	x
SNMP: Read SNMPv2 MIB	x	x	x
SNMP: Read IPv6 MIB	x	x	x
Use TeleService	x	-	-

Table 1-13 Runtime rights CP 1242-7

Right	NET Administrator	NET Standard	NET Diagnose
Use TeleService	x	-	-

Table 1-14 Runtime rights CP 1243-7 LTE

Right	NET Administrator	NET Standard	NET Diagnose
TIA Portal: Perform diagnostics of the security module	x	x	x
Use TeleService	x	-	-

Table 1-15 Runtime rights CP 1243-8 IRC

Right	NET Administrator	NET Standard	NET Diagnose
SNMP: Read MIB-II	x	x	x
SNMP: Write MIB-II	x	x	-
SNMP: Read automation MIB	x	x	x

Right	NET Administrator	NET Standard	NET Diagnose
TIA Portal: Perform diagnostics of the security module	x	x	x
SNMP: Read SNMPv2 MIB	x	x	x
SNMP: Read IPv6 MIB	x	x	x

* To be able to use the function, the runtime right "Web server: Access Web diagnostics and CP file system" must be enabled as well.

** To be able to use the function, the runtime right "FTP: Read files from CP file system" must be enabled as well.

*** To be able to use the function, the runtime right "FTP: Write files to CP file system" must be enabled as well.

Configuration limits for users and roles on security modules

- Maximum number of users on one security module:
  The maximum number of users assigned a role with runtime rights for a security module is 33 (32 + 1 administrator when creating the project).
- Maximum number of roles on one security module:
  The maximum number of roles that are assigned runtime rights for a security module and that are assigned to users is 37.

Parameters for user-specific IP rule sets

When using user-specific IP rule sets, note the following parameters of user management:

- Authentication method "RADIUS" (only for SCALANCE S as of V4):
  The authentication of the user is performed by a RADIUS server when user-specific IP rule sets are activated. The password of the user is not configured in STEP 7 when using this authentication method but must be stored on the RADIUS server. Only use this authentication method for users that only need to log on to the Web page of a security module. A user with the "RADIUS" authentication method cannot log on to STEP 7 projects.
- Maximum time of the session (only for SCALANCE S V3 or higher):
  Entry of the time after which a user logged on to the Web page for user-specific IP rule sets of SCALANCE S modules is automatically logged off. The time entered here starts after the logon and after renewing the session on the Web page of the security module.
  - Default setting: 30 minutes
  - Minimum value: 5 minutes
  - Maximum value: 480 minutes

Authentication using a RADIUS server

Module-specific function

This function is available only for SCALANCE S V4 modules or higher, refer to:

AUTOHOTSPOT.

Generating configuration data for SCALANCE M modules

Meaning

You can generate the VPN information for the assignment of parameters to a SCALANCE M using STEP 7. For this to be possible, the module must be in at least one VPN group with a security module or a SOFTNET Security Client. With the generated files, you can then configure the SCALANCE M using the Web Based Management of the device.

Generated files

The following file types are generated:

- Export file with the configuration data
  - File type: *.txt file in ASCII format
  - Contains the exported configuration information for the SCALANCE M including information on the additionally generated certificates.
- VPN group certificates of the module
  - File type of the private key: *.p12 file
  - The file contains the module certificate and the key material.
  - Access is password protected.
- CA certificates of VPN groups
  - File type: *.cer file

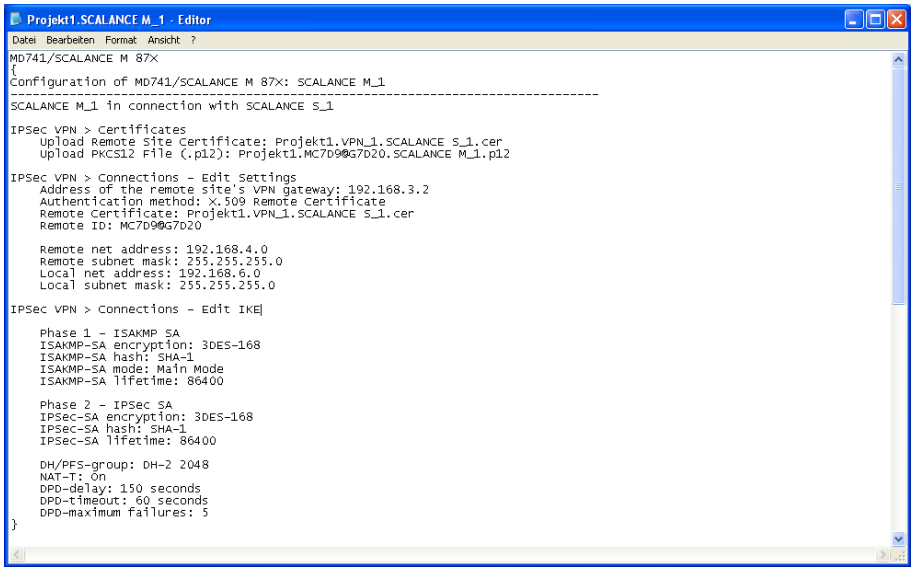

Figure 1-12 SCALANCE M configuration file

Note

No transfer to the SCALANCE M module

Configuration files are not transferred to the SCALANCE M module. An ASCII file is generated with which you can configure the VPN-relevant properties of the SCALANCE M. To allow this, the SCALANCE M must be located in at least one VPN group with another security module or a SOFTNET security client.

Note

Protecting exported configuration files from unauthorized access

Configuration files for SCALANCE M exported from STEP 7 can contain security related information. You should therefore make sure that these files are protected from unauthorized access. This is particularly important when passing on the files.

Follow the steps below

1. Select the module of the type "SCALANCE M".
2. In the local security settings, select the entry "SCALANCE M configuration".
3. Select the "Generate SCALANCE M files" check box and select a storage location for the configuration files.
4. Specify a password for the encryption of the VPN group certificate by using the project name as the password or assigning a password of your own.
5. Compile the configuration of the SCALANCE M module.

Result: The files (.txt file and certificates) are stored in the folder you specify.

Generating configuration data for security devices

Meaning

A security device serves as substitute for a device from another manufacturer.

V1.0: The device can only be operated as a VPN device.

V1.1 or higher: You can choose whether the device is operated as VPN device or Internet gateway.

Security device as VPN device

You can generate the VPN information for the assignment of parameters to a VPN device using STEP 7. For this to be possible, the VPN device must be in at least one VPN group with a security module. With the generated files, you can then configure the VPN device.

Note

Protecting exported configuration files from unauthorized access

Configuration files for VPN devices exported from STEP 7 can contain security related information. You should therefore make sure that these files are protected from unauthorized access. This is particularly important when passing on the files.

To use the security device as VPN device, proceed as follows:

1. Select the security device to be edited.
2. Select the "Settings" entry in the local security settings.
3. V1.1: Select the "VPN device" mode from the drop-down list. "VPN device" is preset for V1.0.
4. Select the "VPN device" entry in the local security settings.
5. Select the "Generate files of the VPN device" check box.
6. Select the mode of the VPN device from the "VPN mode" drop-down list. This selection is only possible when the VPN device is located in a VPN group.
   - VPN client: The VPN device can adopt the following role:
     Initiator The VPN device actively attempts to establish a VPN connection to the partner.
   - VPN server: The VPN device can adopt the following roles:
     Initiator/Responder: The VPN device actively attempts to establish a VPN connection to the partner. The receipt of requests for VPN connection establishment is also possible. This role is selected as default in the VPN connection properties.
     Responder: The receipt of requests for VPN connection establishment is possible.
   - NCP VPN client for Android (as of version 2.1): The VPN device with installed software NCP Secure VPN Client for Android actively attempts to establish a VPN connection to the partner.
   - SOFNET Security Client V5: The VPN device with installed SSC software actively attempts to establish a VPN connection to a partner.
   - SINEMA RC client: The VPN device with installed SINEMA RC client software actively attempts to establish a VPN connection to a partner.
7. Select a storage location for the VPN configuration files.
8. Specify a password for the encryption of the VPN group certificate by using the project name as the password or assigning a password of your own.

If you have configured the VPN device as a "VPN client":

1. Specify the corresponding file types in the "Exported certificates and keys" entry.
2. In the entry "Subnets reachable through tunnel", enter the subnets and/or IP addresses for which tunnel communication will be enabled.

If you have configured the VPN device as a "VPN server":

1. In the "VPN server address (WAN IP address)" box, enter the WAN IP address at which the VPN server can be reached by the initiator of the VPN connection.
2. Specify the corresponding file types in the "Exported certificates and keys" entry.
3. In the entry "Subnets reachable through tunnel", enter the subnets and/or IP addresses for which tunnel communication will be enabled.

If you have configured the VPN device as an "NCP VPN client for Android (as of version V2.1)":

1. In the "NCP VPN client for Android settings" entry, enter the storage path on the NCP VPN client.

Note

Releasing subnets and subscribers with a VPN connection between NCP secure VPN client and SCALANCE S

The local internal subnet of SCALANCE S modules is released for tunnel communication as default. For this reason, this is always entered as the first released subnet in the file generated for the NCP VPN client. Because the NCP VPN client supports a maximum of 5 entries for released subnets / subscribers, the local internal network of the SCALANCE S module and the first 4 table entries from the "Nodes" entry in the NCP VPN client are released by default. To release other subnets and subscribers for tunnel communication, the file generated by STEP 7 must be suitably adapted.

If you have configured the VPN device as a SOFTNET Security Client V5:

- No additional settings are required.

If you have configured the VPN device as a SINEMA RC client:

1. Select the file to start the SINEMA RC client via the "Search" button.
2. You open the SINEMA RC client with the "Start SINEMA RC client" button.
   No VPN configuration file is created for the SINEMA RC client.

Security device as Internet gateway

To use the security device as Internet gateway, proceed as follows:

1. Select the security device to be edited.
2. Select the "Settings" entry in the local security settings.
3. Select the "Internet gateway" mode from the drop-down list.
4. Select the "Internet gateway" entry in the local security settings.
5. Enter the WAN IP address or the FQDN.
6. Via the "Assigned modules" entry, you can assign modules located in a VPN group to the Internet gateway.

Configuring interfaces for SCALANCE S modules

Overview

You will find information on configuration interfaces of SCALANCE S modules in the section AUTOHOTSPOT. The information contained here deals with the following configuration options:

- Mode (bridge mode / routing mode / ghost mode (only for SCALANCE S602 as of V3.1)): Setting the operating mode (Page 474)
- IP address parameters: Configuring IP address parameters (Page 475)
- Port settings (only for SCALANCE S as of V3): Configuring port mode (Page 476)
- Settings for the Internet Service Provider (ISP) if one of the interfaces is operated using PPPoE (only for SCALANCE S as of V3): Configuring an Internet connection (Page 477)
- DNS (only for SCALANCE S as of V3): Configuring DNS (Page 479)
- Dynamic DNS (only for SCALANCE S as of V3): Configuring dynamic DNS (Page 479)
- LLDP (only for SCALANCE S as of V4 in routing mode): Configuring LLDP (Page 482)
- Media redundancy in ring topologies (e.g. MRP client), (only for SCALANCE S627-2M as of V4 in routing mode): AUTOHOTSPOT

For more information on special features of the Ghost mode, refer to the following section: Special features of the ghost mode (Page 485)

The configuration of the interfaces of CPs is described in the sections dealing with the CPs.

Setting up a firewall

Overview of the firewall

Meaning

The firewall functionality of the security modules is intended to protect networks and stations from third-party influence and interference. This means that only certain, previously specified communications relations are permitted. The firewall discards invalid frames without sending a response.

To filter the data traffic, IP addresses, IP subnets, services or MAC addresses can be used. You can also set a limit fro the transmission speed.

The firewall functionality can be configured for the following protocol levels:

- IP firewall with stateful packet inspection (layer 3 and 4)
- Firewall also for Ethernet "non-IP" frames according to IEEE 802.3 (layer 2)

With security modules capable of VPN, the firewall can also be used for the encrypted data traffic (IPsec tunnel). With the SCALANCE S602 Security module, the firewall can only be used for unencrypted data traffic.

Firewall rules

Firewall rules describe which packets in which direction are permitted or forbidden. IP rules affect all IP packets of layer 3 or higher. MAC rules only affect frames lower than layer 3.

Types of firewall rules

- Global firewall rule sets: Global firewall rule sets can be assigned to several security modules at the same time. Global firewall rule sets are configured in the global security functions.
- Local firewall rules: Local firewall rules are configured in the local security settings of a security module.
- User-specific IP rule sets (only for SCALANCE S as of V3): User-specific IP rule sets can be assigned to individual or several security modules at the same time. User-specific IP rule sets are configured in the global security functions and assigned there to one or more users.
  SCALANCE S as of V4 (RADIUS): User-specific IP rule sets can be assigned individual or multiple users as well as individual or multiple roles.

Service definitions

With the aid of service definitions, you can also define firewall rules clearly in a compact form. Service definitions are configured in the global security functions and can be used in the global, local and user-specific firewall rules.

Adapting standard rules for IP services

For SCALANCE S modules V3 or higher, you can adapt service-specific firewall rules that are set as default for the interfaces of the security modules. You will find information on configuring these firewall rules in the section Adapting standard rules for IP services (Page 501).

Automatically generated firewall rules for CP connections

For connections that were configured using CPs, STEP 7 automatically creates firewall rules that allow communication with the partner of the CP in the specified direction (CP active/ passive). The connection establishment directions are taken into account. To display these firewall rules, if the advanced firewall mode is enabled, the "Update connection rules" button needs to be clicked. The firewall rules are then displayed in advanced firewall mode.

Which firewall rules are generated automatically is described in the following sections.

- For S7-300 CPs/S7-400 CPs/PC CPs: Connection-related automatic firewall rules (Page 533) in the section "Security for S7-300 / S7-400 / PC CPs".
- For S7-1200-/S7-1500-CPs: Connection-related automatic firewall rules (Page 546) in the section "Security for S7-1200 / S7-1500 CPs".

Enabling the firewall

Firewall functionality for a specific security module is controlled in the local security settings using the "Enable firewall" check box. If the check box is enabled, the firewall can be configured and is effective following the download. For a security module that is a member of a VPN group, the "Enable firewall" check box is activated by default and cannot be deactivated. After switching to advanced firewall mode, it is not possible to switch back to standard mode. For more information about the standard and advanced firewall modes, refer to the section:

Overview of local firewall rules (Page 393).

Global firewall rule sets

Application

Global firewall rule sets are configured independently of the module in the global security functions. A firewall rule set consists of one or more firewall rules and is assigned to the individual security modules.

In the global firewall rule sets, a distinction is made between the following:

- IP rule sets
- MAC rule sets

The following schematic illustrates the relationship between globally defined rule sets and locally used rule sets.

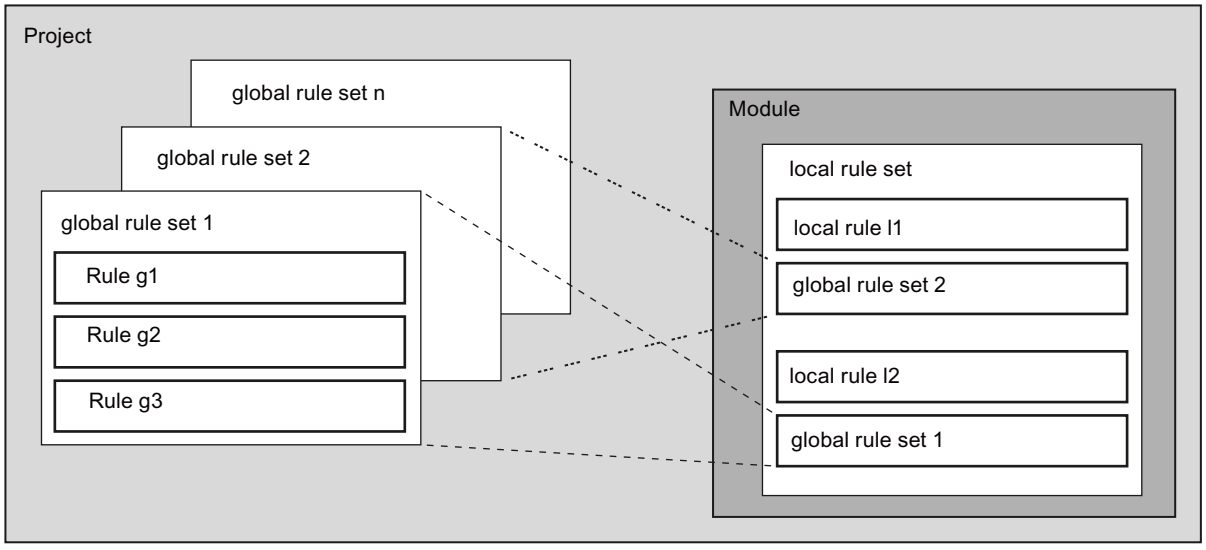

Configuring

When configuring global firewall rules, you can make detailed firewall settings. You can allow individual services for a single node or all services for the node for access to the station or network.

When are global IP and MAC firewall rules useful?

Global firewall rules are useful if you want to define identical filter criteria for communication with several security modules.

Note

Assigning firewall rule sets with incompatible firewall rules

For a security module, only the firewall rules from firewall rule sets are adopted correctly if the security module supports them. A firewall rule contained in a global firewall rule set with the direction "From: External" or "To: Any" is, for example not assigned to a CP 1628. The other firewall rules of the global firewall rule set are adopted if the CP 1628 supports them.

Global firewall rule sets - conventions

Global firewall rule sets are used locally

The following conventions apply when creating a global set of firewall rules and when assigning it to a module:

- Configuration view
  Global firewall rule sets are configured in the global security functions.
- Priority
  As default, locally defined rules have a higher priority than the global IP and MAC firewall rule sets. Newly assigned global IP and MAC firewall rule sets are therefore initially entered at the bottom of the local rule list.
  The priority can be changed by changing the position in the rule list.
- Entering, changing or deleting rule sets
  Global firewall rule sets cannot be edited in the local rule list of the firewall rules in the module properties. They can only be displayed there and positioned according to the required priority.
  It is not possible to delete a single rule from an assigned rule set in the local security settings. Only the entire rule set can be removed from the local list of rules. The firewall rule sets in the global security functions remain unaffected.

Creating global firewall rule sets

How to access this function

1. In the global security functions, select the entry "Firewall" > "Global firewall rule sets" > "IP rule sets" or "MAC rule sets".
   Result: The previously created IP rule sets or MAC rule sets are displayed under the selected entry.
2. Double-click on the entry "Add new IP rule set" or "Add new MAC rule set".
   Result: The created firewall rule set is displayed with an automatically assigned number below the entry "IP rule sets" or "MAC rule sets".

3. Double-click on the created firewall rule set.
   Result: In the working area the assignment dialog for firewall rule sets is displayed. The selected firewall rule set is selected in the "Rule set" drop-down list. The required security modules can be assigned to this, see section Assigning global firewall rule sets (Page 374). In the local security settings, the configurable properties of the firewall rule set are displayed.
4. Enter the following data in the "General" entry in the local security settings:
   – Name: Project-wide, unique name of the rule set. The name appears in the local rule list of the security module after the rule set is assigned.
   – Description (optional): Enter a description of the global rule set.
5. In the local security settings in the entry "Firewall rule set" enter the firewall rules in the list.
   Note the parameter description in the following sections:
   For IP rule sets: Defining IP packet filter rules (Page 394)
   For MAC rule sets: Defining MAC packet filter rules (Page 399)
   To switch over to the local security settings of another firewall rule set, select this from the "Rule set" drop-down list of the assignment dialog or double-click on the corresponding entry in the global security functions.

Result

You have created the global firewall rule set and can now assign this to the required security modules.
Note the descriptions in the following section:

Assigning global firewall rule sets (Page 374)

Assigning global firewall rule sets

Requirement

You have enabled the advanced firewall mode for the security modules you want to assign to a firewall rule set.

Procedure

1. In the global security functions, select the entry "Firewall" > "Global firewall rule sets" > "Assign module to a firewall rule set".
2. From the "Rule set" drop-down list, select the rule set to which you want to assign the security module.
   In the right-hand table, you will see the security modules that you can assign to the selected firewall rule set. In the left-hand table, you will see the security modules already assigned to the selected firewall rule set.
3. In the "Available modules" area, select the security modules you want to assign to the selected rule set.
4. Click the "<<" button to assign the selected modules to the selected rule set.

Result

The global rule set is used by the assigned security modules as a local rule set and appears automatically at the end of the list of firewall rules in the local security settings.

Exporting and importing global firewall rule sets

Meaning and function

Global IP rule sets and global MAC rule sets can be exported from STEP 7 in the XLSX format and imported into STEP 7. Per firewall rule set an XLSX file is created when you export.

The export and import provides the option of exchanging firewall rule sets between different STEP 7 projects. Exported firewall rule sets can also be edited in Microsoft Excel® and then imported into STEP 7 again. This simplifies the making of mass changes.

Firewall rule sets exported from STEP 7 are largely compatible with the Security Configuration Tool (SCT). Firewall rule sets exported from from the Security Configuration Tool are fully compatible with STEP 7.

Note

Restrictions when importing firewall rule sets from STEP 7 into the Security Configuration Tool

In SCT, IPv6 firewall rules and ICMPv6 services are not supported. During import firewall rules identified as IPv6 firewall rules and ICMPv6 services are ignored.

As the transmission speed limit a maximum of 100 Mbps are permitted in SCT. During import firewall rules with more than 100 Mbps are ignored.

Exporting firewall rule sets from STEP 7

1. In the global security functions, select the entry "Firewall" > "Global firewall rule sets".
2. Double-click on the "Overview" entry.
   Result: The IP rule sets and MAC rule sets that exist in STEP 7 are displayed in the working area.
3. If you do not want to export all firewall rule sets, enable the check box of the firewall rule sets to be exported.
4. Click the button.
5. In the "Export firewall rule sets" dialog, select whether all firewall rule sets, only the selected firewall rule sets or all firewall rule sets supported by SCT will be exported. With the third option firewall rule sets containing firewall rules containing communication directions not supported by SCT are excluded from the export.
6. Enter a path on which the firewall rule sets will be saved and click the "Export" button.
   Result: Per firewall rule set an XLSX file was created. The XLSX files were named with the name of the corresponding firewall rule set and according to the date and time of the export.

Importing firewall rule sets in STEP 7

Exported firewall rule sets can be edited prior to importing them. Refer to the information in the section "Requirements for firewall rule sets to be imported".

Follow the steps outlined below to import firewall rule sets in STEP 7:

1. In the global security functions, select the entry "Firewall" > "Global firewall rule sets".
2. Double-click on the "Overview" entry.
   Result: The IP rule sets and MAC rule sets that exist in STEP 7 are displayed in the working area.
3. Click the button.
4. If firewall rule sets of the same name are to be overwritten, select the "Overwrite existing firewall rule sets" check box.
   Note: For the comparison between the firewall rule sets existing in STEP 7 and the XLSX files, only the name of the firewall rule sets is used. The information about the date and time of the XLSX files is ignored.
5. Select the XLSX files to be imported. Multiple selection is possible.
6. Click the "Import" button.
   Result: The firewall rule sets of the selected XLSX files have been imported in STEP 7.

Structure of exported IP rule sets

For every IP rule set exported from STEP 7, an XLSX file is created with the following table sheets:

- IP Ruleset: Contains the IP packet filter rules of the IP rule set
- IP Services: Contains the IP services used in packet filter rules of the IP rule set. IP services in the IP service groups used are also displayed.
- ICMP Services: Contains the ICMP services used in packet filter rules of the IP rule set. ICMP services in the IP service groups used are also displayed.
- IP Service Groups: Contains the IP service groups used in IP packet filter rules of the IP rule set. The corresponding IP and ICMP services are displayed for every IP service group.

Structure of exported MAC rule sets

For every MAC rule set exported from STEP 7, an XLSX file is created with the following table sheets:

- MAC Ruleset: Contains the MAC packet filter rules of the MAC rule set
- MAC Services: Contains the MAC services used in MAC packet filter rules of the MAC rule set. MAC services in the MAC service groups used are also displayed.
- MAC Service Groups: Contains the MAC service groups used in MAC packet filter rules of the MAC rule set. The corresponding MAC services are displayed for every MAC service group.

Conventions for rule sets to be imported

The following conventions apply to firewall rule sets to be imported in STEP 7:

- The files have the file extension *.XLSX and the corresponding Office Open XML format.
- If the name of the XLSX file is longer than 128 characters, the rule set will be created in STEP 7 with a standard name.
- The names of the columns within the table sheets must not be changed, the order of the columns may, however, be changed.
- The names of the table sheets must not be changed, the order of the table sheets may, however, be changed.

Permitted parameter values for IP rule sets to be imported

The following values may be entered in the table columns of XLSX files for IP rule sets. Upper and lower case characters are not taken into account when the notation is checked.

Table 1-16 Permitted parameter values for table sheet "IP Ruleset"

Action	From	To	IPv6*	Source IP address	Destination IP Address	Service	Bandwidth	Logging	Stateful	Comment
• Accept • Drop • Reject	External	• Internal • DMZ • Tunnel • Any • Station • Backplane	• True • False	[free text with max. 255 characters]	[free text with max. 255 characters]	[free text with max. 128 characters]	[Possible values: 0.001… 1000]	• True • False	• True • False	[free text with max. 255 characters]
	Internal	• External • Internal • DMZ • Tunnel • Any • Station								
	DMZ	• External • Internal • Tunnel • Any								
	Tunnel	• External • Internal • DMZ • Tunnel • Any • Station • Backplane								
	Any	• External								

Action	From	To	IPv6*	Source IP address	Destination IP Address	Service	Bandwidth	Logging	Stateful	Comment
		• Internal • DMZ • Backplane								
	Station	• External • Internal • Tunnel • Backplane								
	Backplane	• Any • External • Station • Tunnel								

* This column is optional. If the column does not exist when importing into STEP 7, the default value "False" is used in STEP 7.

Table 1-17 Permitted parameter values for table sheet "IP Services"

Name	Protocol	Source Port	Destination Port
[free text with max. 128 characters]	• UDP • TCP • All	[free text with max. 32 characters]	[free text with max. 32 characters]

Table 1-18 Permitted parameter values for table sheet "ICMP Services"

Name	ICMPv6*	Type	Code
[free text with max. 128 characters]	True	• PacketToBig • EchoRequest • EchoReply • MulticastListenerQuery • MulticastListenerReport • V2MulticastListenerReport • MulticastListenerDone • RouterSolicitation • RouterAdvertisement • NeighbourSolicitation • NeighbourAdvertisement • RedirectMessage • InverseDiscoverySolicitation • InverseDiscoveryAdvertisement • HomeDiscoveryRequest • HomeDiscoveryReply • MobilePrefixSolicitation • MobilePrefixAdvertisement • CertificationPathSolicitation • CertificatePathAdvertisement • IcmpMessagesUtilizedBySeamoby • MulticastRouterAdvertisement • MulticastRouterSolicitation • MulticastRouterTermination • FmIpV6 • RplControl	NoCode
		RouterRenumbering	• RouterRenumberingCommand • RouterRenumberingResult • SequenceNumberReset
		IcmpNodeInformationQuery	• ContainsIpV6

Name	ICMPv6*	Type	Code
			• ContainsName • ContainsIpV4
		IcmpInformationResponse	• Successful • RefuseToSupplyAnswer • UnknownQtypeOfQuery
		TimeExceeded	• HopLimitExceededInTransit • FragmentReassemblyTimeExceeded
		ParameterProblem	• ErroneousHeaderFieldEncountered • UnrecognizedNextHeaderTypeEncountered • UnrecognizedIpV6OptionEncountered
	False	• EchoReply • SourceQuench • AlternateHostAddress • EchoRequest • RouterSolicitation • TimestampReply • TimestampRequest • InformationRequest • InformationReply • AddressMaskReply • AddressMaskRequest • MobileHostRedirect • IpV6WhereAreYou • IpV6IAmHere • MobileRegistrationReply • MobileRegistrationRequest • Skip • Star	NoCode
		DestinationUnreachable	• Star • NetUnreachable • HostUnreachable • ProtocolUnreachable • PortUnreachable • FragmentationNeededAndDontFragmentWasSet • SourceRouteFailed

Name	ICMPv6*	Type	Code
			• DestinationNetworkUnknown • DestinationHostUnknown • SourceHostIsolated • CommunicationWithDestinationNetworkIsAdministrativelyProhibited • CommunicationWithDestinationHostIsAdministrativelyProhibited • DestinationNetworkUnreachableForTypeOfService • DestinationHostUnreachableForTypeOfService • CommunicationAdministrativelyProhibited • HostPrecedenceViolation • PrecedenceCutOffInEffect
		Redirect	• Star • DatagramForNetwork • DatagramForTheHost • DatagramForTheTypeOfServiceAndNetwork • DatagramForTheTypeOfServiceandHost
		RouterAdvertisement	• Star • NormalRouterAdvertisement • DoesNotRouteCommonTraffic
		TimeExceeded	• Star • TimeToLiveExceededInTransit • FragmentReassemblyTimeExceeded
		ParameterProblem	• Star • TheIpHeaderIsInvalid • ArequiredOptionIsMissing
		Traceroute	• Star • OutboundPacketSuccessfullyForwarded

Name	ICMPv6*	Type	Code
			• NoRouteForOutboundPacketThePacketWasDiscarded
		ConversionError	• Star • UnknownOrUnspecifiedError • DontConvertOptionPresent • UnknownMandatoryOptionPresent • KnownUnsupportedOptionPresent • UnsupportedTransportProtocol • OverallLengthExceeded • IpHeaderLengthExceeded • TransportProtocolGreaterThen255 • PortConversionOutOfRange • TransportHeaderLengthExceeded • BitRolloverMissingAndAckSet • UnknownMandatoryTransportOptionPresent
		Photuris	• Star • BadSpi • AuthenticationFailed • DecompressionFailed • DecryptionFailed • NeedAuthentication • NeedAuthorization

* This column is optional. If the column does not exist when importing into STEP 7, the default value "False" is used in STEP 7.

Table 1-19 Permitted parameter values for table sheet "IP Service Groups"

Name	Description	IP Services
[free text with max. 128 characters]	[free text with max. 255 characters]	[free text, services are separated from each other with commas]

Permitted parameter values for MAC rule sets to be imported

The following values may be entered in the table columns of XLSX files for MAC rule sets. Upper and lower case characters are not taken into account when the notation is checked.

Table 1-20 Permitted parameter values for table sheet "MAC Ruleset"

Action	From	To	Source MAC Address	Destination MAC Address	Service	Bandwidth	Logging	Comment
• Accept • Drop • Reject	External	• Internal • Tunnel • Any • Station	[MAC address in the correct format: xx-xx-xx-xx-xx-xx]	[MAC address in the correct format: xx-xx-xx-xx-xx-xx]	[free text with max. 128 characters]	[Possible values: 0.001… 1000]	• True • False	[free text with max. 255 characters]
	Internal	• External • Tunnel • Any						
	Tunnel	• External • Internal • Station						
	Any	• External • Internal						
	Station	• External • Tunnel						

Table 1-21 Permitted parameter values for table sheet "MAC Services"

Name	Protocol	DSAP	SSAP	CTRL	OUI	OUI Type
[free text with max. 128 characters]	• SNAP • PROFINET IO • ISO • [hexadecimal values beginning with 0x]	[hexadecimal values with max. 2 characters]	[hexadecimal values with max. 2 characters]	[hexadecimal values with max. 2 characters]	[hexadecimal values with max. 6 characters]	[hexadecimal values with max. 4 characters]

Table 1-22 Permitted parameter values for table sheet "MAC Service Groups"

Name	Description	MAC Services
[free text with max. 128 characters]	[free text with max. 255 characters]	[free text, services are separated from each other with commas]

IP services

Defining IP services

How to access this function

In the global security functions, select the entry "Firewall" > "Services" > "Define services for IP rules".

Procedure

Using the definition of IP services, you can define succinct and clear firewall rules for specific services. You select a name and assign the service parameters to it.

These services defined in this way can also be grouped together under a group name.

When you configure the packet filter rules, you then use this name.

Parameters for IP services

You define the IP services using the following parameters:

Table 1-23 IP services: Parameter

Parameter	Meaning/comment	Available options / ranges of values
Name	Name for the service that is used as identification in the rule definition or in the group. The names of pre-defined services cannot be changed.	• Name must start with a letter. • Name must not contain any special characters. • Name must not be redundant.
Protocol	Selection of the protocol type	• TCP • UDP • TCP+UDP • All

Parameter	Meaning/comment	Available options / ranges of values
Source port	Filtering is based on the port number specified here; this defines the service access at the frame sender.	With the protocol selection "TCP+UDP", it is not possible to specify a port. Examples: *: Port is not checked 20 or 21: FTP service
Destination port	Filtering is based on the port number specified here; this defines the service access at the frame recipient.	With the protocol selection "TCP+UDP", it is not possible to specify a port. Examples: *: Port is not checked TCP 80: Web HTTP service TCP 102: S7 protocol

Defining ICMP services

How to access this function

1. In the global security functions, select the entry "Firewall" > "Services" > "Define services for IP rules".
2. Select the "ICMP" tab.

Procedure

Using the definition of ICMP services, you can define succinct and clear firewall rules for specific services. You select a name and assign the service parameters to it.

These services defined in this way can be grouped together under a group name.

When you configure the packet filter rules, you then use this name.

Parameters for ICMP services

Parameter	Meaning/comment	Available options / ranges of values
Name	User-definable name for the service that is used as identification in the rule definition or in the group. The names of predefined ICMPv6 services cannot be modified.	• Name must not occur more than once
ICMPv6	If you enable this check box, the ICMP service is declared as an ICMPv6 service and you can select an ICMPv6-specific type and code for the service. An ICMPv6 service can only be used in the firewall rule of a security module that supports IPv6.	• Enabled • Disabled (default)

Parameter	Meaning/comment	Available options / ranges of values
Type	Type of the ICMPv4 or ICMPv6 message.	If the "ICMPv6" check box is deselected, ICMPv4-specific types can be selected. If the check box is selected, ICMPv6-specific types can be selected.
Code	Code of the ICMP type.	Values depend on the selected type.

Creating service groups

How to access this function

1. In the global security functions, select the entry "Firewall" > "Services" > "Define services for IP rules".
2. Select the "Service groups" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name. IPv4 and IPv6 services can be collected in the same service group.

Create groups in the open "Service groups" tab. You can then assign services to a group in the "Group management" tab.

Follow the steps below

1. First, create groups in this tab with names to suit your requirements and add a description if required.
2. Select the "Group management" tab. You can assign the previously specified IP services to the groups defined in this tab.

Managing service groups

How to access this function

1. In the global security functions, select the entry "Firewall" > "Services" > "Define services for IP rules".
2. Select the "Group management" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name. IPv4 and IPv6 services can be collected in the same service group.

Use the "Group management" tab to assign services to a selected group you created previously in the "Service groups" tab.

Follow the steps below

1. Select a group created previously in the "Service groups" tab from the "Service groups" drop-down list in this tab.
2. Then assign the required services from the right-hand "Available services" list box to the group.

MAC services

Define MAC services

How to access this function

In the global security functions, select the entry "Firewall" > "Services" > "Define services for MAC rules".

Meaning

Using the definition of MAC services, you can define succinct and clear firewall rules for specific services. You select a name and assign the service parameters to it.

These services defined in this way can be grouped together under a group name.

When you configure the global or local packet filter rules, you use this name.

Parameters for MAC services

A MAC service definition is formed using protocol-specific MAC parameters:

Table 1-24 MAC services - parameters

Parameter	Meaning/comment	Available options / ranges of values
Name	User-definable name for the service that is used as identification in the rule definition or in the group.	• Name must start with a letter. • Name must not contain any special characters. • Name must not be redundant.
Protocol	Name of the protocol type: • ISO ISO identifies frames with the following properties: Lengthfield <= 05DC (hex), DSAP= userdefined SSAP= userdefined CTRL= userdefined • SNAP SNAP identifies frames with the following properties: Lengthfield <= 05DC (hex), DSAP=AA (hex), SSAP=AA (hex), CTRL=03 (hex), OUI=userdefined, OUI-Type=userdefined • PROFINET IO • As an alternative, a protocol number can be entered. The protocol entries 0800 (hex) and 0806 (hex) are not accepted since these values apply to IP or ARP frames.	• ISO • SNAP • PROFINET IO • 0x (entry of the protocol number)
DSAP	Destination Service Access Point: LLC recipient address	
SSAP	Source Service Access Point: LLC sender address	
CTRL	LLC control field	
OUI	Organizationally Unique Identifier (the first 3 bytes of the MAC address = vendor ID)	
OUI type	Protocol type/identification	

Note

Processing for S7-CPs

Only settings for ISO frames with DSAP=SSAP=FE (hex) are processed. Other frame types are not relevant for S7 CPs and are therefore discarded even before processing by the firewall.

Special settings for SIMATIC NET services

To filter special SIMATIC NET services, please use the following protocol settings:

- DCP:
  PROFINET IO
- SiClock :
  OUI= 08 00 06 (hex) , OUI-Type= 01 00 (hex)

Creating service groups

How to access this function

1. In the global security functions, select the entry "Firewall" > "Services" > "Define services for MAC rules".
2. Select the "Service groups" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name.

Create groups in the open "Service groups" tab. You can then assign services to a group in the "Group management" tab.

Follow the steps below

1. First, create groups in this tab with names to suit your requirements and add a description if required.
2. Select the "Group management" tab. You can assign the previously specified MAC services to the groups defined in this tab.

Managing service groups

How to access this function

1. In the global security functions, select the entry "Firewall" > "Services" > "Define services for MAC rules".
2. Select the "Group management" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name.

Use the "Group management" tab to assign services to a selected group you created previously in the "Service groups" tab.

Follow the steps below

1. Select a group created previously in the "Service groups" tab from the "Service groups" drop-down list in this tab.
2. Then assign the required services from the right-hand "Available services" list box to the group.

Overview of local firewall rules

Meaning

Local firewall rules are configured in the local security settings of a security module and apply only to this security module. After enabling the firewall functionality you can either use predefined firewall rules or define firewall rules in the advanced firewall mode.

Using predefined firewall rules

Here, you use simple, predefined rules. You can only enable service-specific rules. The enabled services are allowed for all nodes in the specified direction. You can find detailed information on the definition of firewall rules in this dialog in the following module-specific sections:

- For SCALANCE S: AUTOHOTSPOT
- For S7-300/S7-400/PC CPs: AUTOHOTSPOT
- For S7-1200/S7-1500 CPs: AUTOHOTSPOT

The communication direction "Station" that can be configured for CPs, includes access to the CP and access to the CPU via the CP.

Defining firewall rules in advanced firewall mode

In advanced firewall mode, you can make detailed firewall settings. You can allow individual services for a single node or all services for the node for access to the station or network. You enable the advanced firewall mode using the "Activate firewall in advanced mode" check box. You open the editors for configuration of IP rules and MAC rules in the advanced firewall mode using the corresponding buttons in "Firewall > IP rules". For the configuration options available here in the editors refer to the following section:

For IP packet filter rules: Defining IP packet filter rules (Page 394)

For MAC packet filter rules: Defining MAC packet filter rules (Page 399)

Note

Deactivating the advanced firewall mode not possible

Once you have activated the advanced firewall mode, you can no longer deactivate it.

Adopting predefined firewall rules in the advanced firewall mode

When you switch over to the advanced firewall mode, enabled predefined firewall rules and logging settings are adopted. If, for example, the "Allow S7 protocol" check box is selected in the direction "From External to Internal" for a SCALANCE S module, an accept rule with this communication direction and the service "S7" are created automatically in the advanced firewall mode.

If predefined logging settings exist for S7-1200 / S7-1500 CPs, in advanced mode these are not shown as firewall rules but under the entry “Firewall > Log settings” where they can be edited.

Configuration limits

Number of firewall rules (advanced firewall mode)	
SCALANCE S as of V3	Maximum of 256
S7-1200/S7-1500 CPs	Maximum of 256
CP 343-1 Adv. / 443-1 Adv.	Maximum of 226
CP 1628	Maximum of 226

Defining IP packet filter rules

Meaning

With IP packet filter rules, you filter according to IP frames, such as TCP, UDP and ICMP.

Within a packet filter rule, you can also fall back on the definitions of the IP services.

Assigned global firewall rules and automatically generated firewall rules for NAT/NAPT are shown as rule sets.

Operator controls of the menu bar

The operator controls of the menu bar have the following functions:

Operator control element	Function
	Enabling the firewall Enables the editor for IP rules. If the editor is enabled, the corresponding rules can be configured and are transferred to the module during loading.
	Insert rule above Inserts a rule above the selected rule / the selected rule set. To insert above a rule set, its first line must be selected.
	Insert rule below Inserts a rule below the selected rule / the selected rule set. To insert below a rule set, its first line must be selected.
	Move up Moves the selected rule / the selected rule set one line up. To move a rule set, its first line must be selected.

Operator control element	Function
	Move down Moves the selected rule / the selected rule set one line down. To move a rule set, its first line must be selected.
	Expand rule sets Shows existing rule sets expanded.
	Collapse rule sets Shows existing rule sets collapsed.
	Separate Converts the selected line into a separate rule or rules. If the first line of a rule set is selected, all rules of the rule set are converted into separate rules. If one rule contained in a rule set is selected, this is converted into a separate rule. In addition to this, the following rules of the rule set are converted into a separate rule set.

Entering IP packet filter rules

After enabling the editor the IP rules can be configured with the following parameters:

Table 1-25 IP rules: Parameter

Parameter	Meaning/comment	Available options / ranges of values
Action	Allow/disallow (enable/ block)	• Accept Allow frames according to definition. • Drop Block frames according to definition. • Reject The frames are rejected, and the sender receives a corresponding message. For firewall rules generated automatically as result of configuring a connection and subsequently adapted manually: • Accept* • Drop* If you change automatically created connection rules, when the "*" option is selected they are not recreated and overwritten by STEP 7.
From / To	Selection of the communications directions for which the rule will apply.	Described in separate sections. • For SCALANCE S modules: IP packet filter directions SCALANCE S (Page 500) • For S7-300/S7-400/PC CPs: IP packet filter directions S7-300-/S7-400-/ PC-CPs (Page 531) • For S7-1200/S7-1500 CPs: IP packet filter directions S7-1200 / S7-1500 CPs (Page 544) The communication direction "Station" includes access to the CP and access to the CPU via the CP. The communication direction "Any" includes all the firewall directions supported by the security module at the time of loading. The communication directions "Station", "Backplane bus" (only for CP 1543-1 as of V2.1 with the option "IP routing between communications modules") and "Tunnel" are classified as trusted by default. For communication between trustworthy directions no firewall rules need to be configured.

Parameter	Meaning/comment	Available options / ranges of values
Source IP address	The firewall rule is applied to the frames with a sender that has the IP address specified here. If you do not specify an IP address, the firewall rule applies to all nodes in the communications direction you selected in the "From" column.	You can find additional information about IP addresses in the IP addresses in IP packet filter rules (Page 402) section. **Configuration options in ghost mode (only for SCALANCE S S602 as of V3.1):** If ghost mode is activated, the IP address of the internal node is dynamically determined by the security module at runtime. Depending on the selected direction, you can select one of the following options in the column "Source IP address" (for direction "From internal to external") or in the column "Destination IP address" (for direction "From external to internal"): • IP address of the internal node: The IP address of the internal node is inserted in the firewall rule by the SCALANCE S. • Limited broadcast: The broadcast IP address 255.255.255.255 is inserted in the firewall rule by the SCALANCE S. • Directed broadcast: The broadcast IP address of the SCALANCE S network is inserted in the firewall rule by the SCALANCE S. A directed broadcast can also be forwarded into the destination network via routers. • Multicast: The multicast address band 224.0.0.0 /24 is inserted in the firewall rule by the SCALANCE S. After selecting this option, as an alternative a specific multicast IP address from the multicast address band can be specified.
Destination IP address	The firewall rule is applied to the frames with a recipient that has the IP address specified here. If you do not specify an IP address, the firewall rules applies to all nodes in the communications direction you selected in the "To" column.	
Service	Selection of the IP/ICMP service or the service group to be used. Note: Before using a predefined service make sure that its predefined parameters meet your requirements. Pay particular attention to the predefined source and destination ports.	The drop-down list box displays the predefined and configured IP services and service groups you can select. These are displayed in the global security functions under "Firewall" > "Services" > "Define services for IP rules".
Transmission speed (Mbps)	Limitation of the transmission speed Can only be entered if "Accept" is selected for the action. A packet passes through the firewall if the Accept rule applies and the permitted transmission speed for this rule has not yet been exceeded.	CP 343-1 Adv. / CP 443-1 Adv., S7-1200-/S7-1500-CPs, CP 1628: 0.001 ... 100 Mbps SCALANCE S: 0.001 ... 1000 Mbps For global and user-specific rules: 0.001 ... 1000 Mbps For IP rules with the direction "Backplane bus": 0.001 ... 1 Mbps Note: If the direction "From tunnel to station" is configured in a firewall rule for the S7-1200/S7-1500 CPs, no transmission speed limitation can be specified. For CP 1543-1 V2.1 only one transmission speed can be specified if "Any" is configured for one of the two directions. This does not apply to the combination of directions "Any" and "Station".
Logging	Enable and disable logging for this rule. If logging is enabled, the settings for the packet filter logging configured in the local security settings apply.	• Enabled • Disabled (default)

Parameter	Meaning/comment	Available options / ranges of values
Number	Automatically assigned number for the rule. The numbers are recalculated when rules are moved.	
Stateful	If this check box is cleared for an IP rule with the "Accept" action, no firewall states are generated by packets to which the Accept rule applies. Firewall states automatically allow the responses to allowed packets. Can only be adapted if "Accept" is selected for the action. Configuration of IP rules without firewall states is only possible for SCALANCE S modules as of firmware V3. If the responses to packets that have passed the firewall according to such IP rules should also be allowed, additional IP rules need to be configured for these responses.	For Accept rules of SCALANCE S ≥ V3: • enabled (default) • disabled For Accept rules of SCALANCE S < V3 and CPs: • enabled (default) Note: The check box is not present for CPs; however, the activated option is always used implicitly for Accept rules of CPs. For drop rules: • Disabled (default)
Comment	Space for your own explanation of the rule	If a comment is marked with "AUTO", it was created for an automatic connection rule. For rules you have created, entry of a comment is optional.

Table 1-26 Meaning of the entries in the shortcut menu

Entry in the shortcut menu	Meaning
Insert rule above	Use this command to insert a rule above the selected rule / the selected rule set.
Insert rule below	Use this command to insert a rule below the selected rule / the selected rule set.
Delete	This deletes the selected rule or the selected rule set. Notes on removing a globally defined and locally assigned rule set: if you delete a rule set here, this only cancels the assignment to the security module.
Save as global rule set (only for local firewall rules)	Copies the selected firewall rule(s) and inserts them as a global firewall rule set in the global security functions. The firewall configuration currently configured for the security module remains unaffected by this procedure.
Move up	Use this button to move the selected rule or selected rule set up one position in the list. As an alternative, you can move the selected rule or the selected rule set by dragging it with the mouse. You can select multiple entries. The rule / rule set you moved is therefore handled with higher priority.
Move down	Use this button to move the selected rule or selected rule set down one position in the list. As an alternative, you can move the selected rule or the selected rule set by dragging it with the mouse. You can select multiple entries. The rule / rule set you moved is therefore handled with lower priority.
Define service for IP rules	This opens the dialog in which you can manage the IP services and service groups.

Advanced IP rule editor

After selecting an IP rule, an advanced IP rule editor appears in the Inspector window in "Properties > General" and is used for simplified configuration of IP rules. Changes made in this editor are adopted in the selected IP rule.

Configuring IP rules

Address ranges can be configured in the text boxes of the "From" and "To" columns. If several address ranges are configured for an IP rule, the number of addresses must match in each case. After configuring address ranges, STEP 7 creates an IP rule set from the selected IP rule that contains a corresponding IP rule for each of the addresses.

You make the other settings described in the "IP Rules: Parameters" table under "Additional settings".

Copying IP rules and IP rule sets

You can also copy selected IP rules and IP rule sets. To copy an IP rule set, its first line must be selected.

Defining MAC packet filter rules

Meaning

With the MAC packet filter rules, you filter according to MAC frames.

Within a packet filter rule, you can also fall back on the definitions of the MAC services.

Assigned global firewall rules are shown as rule sets.

Operator controls of the menu bar

The operator controls of the menu bar have the following functions:

Operator control element	Function
	Enabling the firewall Enables the editor for MAC rules. If the editor is enabled, the corresponding rules can be configured and are transferred to the module during loading.
	Insert rule above Inserts a rule above the selected rule / the selected rule set. To insert above a rule set, its first line must be selected.
	Insert rule below Inserts a rule below the selected rule / the selected rule set. To insert below a rule set, its first line must be selected.
	Move up Moves the selected rule / the selected rule set one line up. To move a rule set, its first line must be selected.

Operator control element	Function
	Move down Moves the selected rule / the selected rule set one line down. To move a rule set, its first line must be selected.
	Expand rule sets Shows existing rule sets expanded.
	Collapse rule sets Shows existing rule sets collapsed.

Entering MAC packet filter rules

After enabling the editor the MAC rules can be configured with the following parameters:

Table 1-27 MAC rules: Parameter

Parameter	Meaning/comment	Available options / ranges of values
Action	Allow/disallow (enable/block)	• Accept Allow frames according to definition. • Drop Block frames according to definition. • Reject The frames are rejected, and the sender receives a corresponding message. For firewall rules generated automatically as result of configuring a connection and subsequently adapted manually: • Accept* • Drop* If you change automatically created connection rules, when the "*" option is selected they are not recreated and overwritten by STEP 7.
From / To	Selection of the communications directions for which the rule will apply.	Described in separate sections. • For SCALANCE S modules: MAC packet filter directions SCALANCE S (Page 500) • For S7-300/S7-400/PC CPs: MAC packet filter directions S7-300-/S7-400-/PC-CPs (Page 531) • For S7-1200/S7-1500 CPs: MAC packet filter directions S7-1200 / S7-1500 CPs (Page 545) The communication direction "Station" includes access to the CP and access to the CPU via the CP. As default, the communication directions "Station" and "Tunnel" are classified as being trustworthy. For communication between trustworthy directions no firewall rules need to be configured.

Parameter	Meaning/comment	Available options / ranges of values
Source MAC address	The firewall rule is applied to the frames with a sender that has the MAC address specified here. If you do not specify a MAC address, the firewall rule applies to all nodes in the communications direction you selected in the "From" column.	MAC address in the correct format
Destination MAC address	The firewall rule is applied to the frames with a recipient that has the MAC address specified here. If you do not specify a MAC address, the firewall rule applies to all nodes in the communications direction you selected in the "To" column.	
Service	Selection of the MAC service or the service group to be used. Note: Before using a predefined service make sure that its predefined parameters meet your requirements.	The drop-down list box displays the predefined and configured services and service groups you can select. These are displayed in the global security functions under "Firewall" > "Services" > "Define services for MAC rules".
Transmission speed (Mbps)	Limitation of the transmission speed Can only be entered if "Accept" is selected for the action. A packet passes through the firewall if the Accept rule applies and the permitted transmission speed for this rule has not yet been exceeded.	CP 343-1 Adv. / CP 443-1 Adv., S7-1200-/S7-1500-CPs: 0.001 ... 100 Mbps CP 1628 and SCALANCE S: 0.001 ... 1000 Mbps For global rules: 0.001 ... 1000 Mbps Note: If the direction "From tunnel to station" is configured in a firewall rule for the S7-1200/S7-1500 CPs, no transmission speed limitation can be specified.
Logging	Enable and disable logging for this rule. If logging is enabled, the settings for the packet filter logging configured in the local security settings apply.	• Enabled • Disabled (default)
Number	Automatically assigned number for the rule. The numbers are recalculated when rules are moved.	
Comment	Space for your own explanation of the rule	If a comment is marked with "AUTO", it was created for an automatic connection rule. For rules you have created, entry of a comment is optional.

Table 1-28 Meaning of the menu commands

Button	Meaning
Delete	This deletes the selected rule or the selected global rule set. Notes on removing a globally defined and locally assigned rule set: if you delete a rule set here, this only cancels the assignment to the security module.
Save as global rule set (only for local firewall rules)	Copies the selected firewall rule(s) and inserts them as a global firewall rule set in the global security functions. The firewall configuration currently configured for the security module remains unaffected by this procedure.

Button	Meaning
Move up	Use this button to move the selected rule or selected global rule set up one position in the list. As an alternative, you can move the selected rule or the selected rule set by dragging it with the mouse. You can select multiple entries. The rule / rule set you moved is therefore handled with higher priority.
Move down	Use this button to move the selected rule or selected global rule set down one position in the list. As an alternative, you can move the selected rule or the selected rule set by dragging it with the mouse. You can select multiple entries. The rule / rule set you moved is therefore handled with lower priority.
Define service for MAC rules	This opens the dialog in which you can manage the MAC services and service groups.

IP addresses in IP packet filter rules

Entering IP addresses in IP packet filter rules

In IP packet filter rules, you have the following options for entering IP addresses:

- Nothing specified
  The rule applies to all IP addresses.
- An IP address
  The rule applies specifically to the specified address.
- Address range
  The rule applies to all the IP addresses covered by the address range.
  An address range is defined by specifying the number of valid bit places in the IP address in the format: [IP address]/[number of bits to be taken into account]
  - [IP address]/24 therefore means that only the most significant 24 bits of the IP address are included in the filter rule. These are the first three octets of the IP address.
  - [IP address]/25 means that only the first three octets and the highest bit of the fourth octet of the IP address are included in the filter rule.
- Address area
  For the source IP address, an address range can be entered in the following format:
  [Start IP address]-[End IP address]

IPv4 addresses

An IPv4 address consists of 4 decimal numbers in the range from 0 to 255 separated from each other by a dot.

Table 1-29 Examples of address ranges in IPv4 addresses

Source IP address or destination IP address	Address range		Number of addresses*)
	from	to	
192.168.0.0/**16**	192.168.0.0	192.168.255.255	65.536
192.168.10.0/**24**	192.168.10.0	192.168.10.255	256
192.168.10.0/**25**	192.168.10.0	192.168.10.127	128

Source IP address or destination IP address	Address range		Number of addresses*)
	from	to	
192.168.10.0/**26**	192.168.10.0	192.168.10.63	64
192.168.10.0/**27**	192.168.10.0	192.168.10.31	32
192.168.10.0/**28**	192.168.10.0	192.168.10.15	16
192.168.10.0/**29**	192.168.10.0	192.168.10.7	8
192.168.10.0/**30**	192.168.10.0	192.168.10.3	4
*) Note: Note that the network address and the broadcast address are not available as IP addresses of network nodes in an address range.			

IPv6 addresses

IPv6 addresses consist of 8 fields each with four hexadecimal numbers (128 bits in total). The fields are separated by a colon.

Example: fd00:ffff:ffff:ffff:ffff:ffff:2f33:8f21

Rules / simplifications:

- Leading zeros within a field can be omitted.
  Example: Instead of 2001:0db8:2426:08d3:1457:8a2e:0070:7344 it is also possible to use the notation 2001:db8:2426:8d3:1457:8a2e:70:7344.
- If one or more fields have the value 0 (or 0000), a shortened notation is possible.
  Example: Instead of 2001:0db8:0:0:0:0:1428:57ab it is also possible to use the notation 2001:db8::1428:57ab.
  To ensure uniqueness, this shortened form can only be used once within the entire address.
- Decimal notation with periods
  The last 2 fields or 4 bytes can be written in the normal decimal notation with periods.
  Example: The IPv6 address fd00::ffff.125.1.0.1 is equivalent to fd00::ffff:7d01:1.
- Address range notation in IP packet filter rules: In the same way as IPv4 addresses, IPv6 addresses can also be noted in the form of address ranges.
  Example: The entry "2001:0db8:85a3:08d3:1319:8a2e:0:0/96" covers all IPv6 addresses from 2001:0db8:85a3:08d3:1319:8a2e:0:0 to 2001:0db8:85a3:08d3:1319:8a2e:ffff:ffff.

Carrying out module-specific log settings

Log settings - overview

Module-specific function

Some logging functions are only available for certain security modules.

Log settings in the configuration

The log settings made here are loaded on the module with the configuration and take effect when the security module starts up.

You can restrict the configured packet filter log settings in the online functions if necessary. For example, you can use the online functions to specify that merely IP logging is displayed if you have configured IP and MAC logging.

Logging procedures and event classes

Here, you can specify which data should be logged. As a result, you enable logging as soon as you download the configuration to the security module.

During configuration, you also select one or both of the possible logging procedures:

- Local logging
- Network Syslog

In both logging procedures, the security module recognizes the three following types of events:

- Packet filter events
- Audit events
- System events

Configuring local logging

How to access this function

1. Select the module to be edited.
2. Select the "Log settings" > "Local log memory" entry in the local security settings.

Configuring local logging

Table 1-30 Local logging - settings for log events

Log event	Meaning	Comments
Packet filter log (firewall)	The packet filter log records certain packets of the data traffic. Only those data packets for which a configured packet filter rule (firewall) applies or to which the basic protection reacts (corrupt or invalid packets) are logged. This is only possible when logging is enabled for the packet filter rule. In the "Packets to be logged" drop-down list, you can specify the data packets to be logged: • "All packets": The data packets that are logged are those to which a configured firewall rule applies. In addition to this, all the response packets to such packets that have passed the firewall according to a configured Accept rule are recorded. • "Status generating packets" (only for SCALANCE S as of V3): The only data packets that are logged are those to which a configured firewall rule (standard mode or advanced mode) applies and that initially generate a status in the firewall. Data packets that pass through the firewall by using this firewall status are not logged.	Packet filter log data is not retentive The data is stored in volatile memory on the security module and is therefore no longer available after the power supply has been turned off. For retentive storage, you can also save the log data displayed in the "Online & diagnostics" dialog in a file.
Audit log	The logging of audit events is always enabled. The logged information is always stored in the ring buffer. The audit log automatically records security-relevant events, for example user actions such as activating or deactivating packet logging or downloading configurations to the security module.	Audit log data is retentive The data is stored in a retentive memory of the security module and is therefore still available after turning off the power supply. Note for CPs: The audit log files are not retentive on CPs. You should therefore use a Syslog server to backup this data.
System log	The system log automatically logs successive system events, for example the start of a process or the failed login attempt of a user. Select the "Log settings" > “System events" entry to configure the event filter and cable diagnostics functions.	System log data is not retentive The data is stored in volatile memory on the security module and is therefore no longer available after the power supply has been turned off. For retentive storage, you can also save the log data displayed in the "Online & diagnostics" dialog in a file.

Table 1-31 Local logging - storage of recorded data

Storage	Meaning
Ring buffer	At the end of the buffer, the recording continues at the start of the buffer and overwrites the oldest entries.
One-shot buffer	Recording stops when the buffer is full.

Configuring system events

How to access this function

1. Select the module to be edited.
2. Select the "Log settings" > "System events" entry in the local security settings.

Filtering of the system events

In this dialog, you set a filter level for the system events. As default, the following values are set:

- SCALANCE S: Level 3 (Error)
- CP: Level 3 (Error)

The system events are recorded that have the same priority as the selected filter level and that have a higher priority than the selected filter level. The lower the value of the filter level, the higher its priority and therefore less events are recorded.

The priority of the selected filter level must be the same or lower than the severity set for the cable diagnostics; see the "Setting parameters for line diagnostics" table (not for CPs).

Recommendation: Select "Error" as the filter level or a filter level with a higher priority to prevent recording of general non-critical events.

Note for CP

For CPs, select only level 3 or level 6 since only events of these levels are generated for CPs.

- Level 0 to level 3 error messages are output if level 3 is selected.
- If you select level 6, the error messages of levels 0 to 6 are output.

Properties of the system events - line diagnostics (only for SCALANCE S)

Line diagnostics generates a special system event. A system event is generated when a selectable percentage of bad frames is reached. This system event is assigned the severity and facility set in this dialog.

Table 1-32 Setting parameters for line diagnostics

Function / option / parameter	Meaning
Enable	Enabling and disabling logging
Limit	Selectable percentage of faulty frames representing the limit at which a system event is triggered.
Facility	Select a facility from the drop-down list that identifies the system event to be logged.
Severity	Using the severity, you weight the system events of line diagnostics relative to the severity of the other system events.

Note

Severity of the system events of line diagnostics

The system events of the line diagnostics must not have a lower severity than you set for the filter. At a lower severity, these events cannot pass through the filter and will not be logged.

Configuring the network Syslog

How to access this function

1. Select the module to be edited.
2. Select the "Log settings" > "Network Syslog" entry in the local security settings.

Configuring the network Syslog

Table 1-33 Network Syslog - basic settings

Option / parameter	
Enable network Syslog	Enables and disables the transfer of logging events to the Syslog server.
Syslog server	Here, enter the IP address of the Syslog server. As an alternative, an FQDN can be entered for SCALANCE S modules as of V4. The Syslog server must be accessible from the security module at the specified address and, if applicable, using the router configuration under the "Routing" entry of the local security settings. If the Syslog server cannot be accessed no Syslog messages are sent. You can recognize this operating state on the basis of corresponding system alarms. To activate the sending of the Syslog information again, you may have to update the routing information and initiate a restart of the security module.

Option / parameter	
Enabling event classes	Enable the classes of events to be transferred to the Syslog server. You can classify packet filter and audit events according to the Severity and using the Facility according to their origin.
Module name	The module name is displayed and cannot be changed at this point.

Note

Non-secure transfer of logging events

Logging events are transferred to Syslog servers in plain language. This should be taken into account when using Syslog servers.

Table 1-34 Network Syslog - settings for log events

Log event	Configuring	Comments
Packet filter events (firewall)	The packet filter log records certain packets of the data traffic. Only those data packets for which a configured packet filter rule (firewall) applies or to which the basic protection reacts (corrupt or invalid packets) are logged. This is only possible if logging is activated for the packet filter rule. Syslog alarms can be classified according to their origin and their severity level by setting Facility and Severity. This assignment is carried out with drop-down lists. The severity and facility that you set here is assigned to every event.	The value you select here depends on the evaluation in the Syslog server. If you retain the "default" value setting, the security module specifies the combination of facility and severity with which the event is displayed.
Audit events	The audit log automatically records security-relevant events, for example user actions such as activating or deactivating packet logging or downloading configurations to the security module. Assignment of the severity and facility is carried out with drop-down lists. The severity and facility that you set here is assigned to every event.	The value you select here for the severity and facility depends on the evaluation in the Syslog server. If you retain the "default" value setting, the security module specifies the combination of facility and severity with which the event is displayed.
System events	The system log automatically logs successive system events, for example the start of a process or the failed login attempt of a user.	Select the "Log settings" > “System events" entry in the local security settings to configure the event filter and cable diagnostics functions.

Security module as router

Overview of the routing settings

Meaning

If you operate the security module in routing mode, the networks connected to the internal and external interface are transformed into separate subnets. The DMZ interface (SCALANCE S623/S627-2M only) is connected in routing mode regardless of the mode. In routing mode, the frames intended for an existing IP address in the subnet are forwarded. The firewall rules for the direction of transmission also apply.

You have the following additional options:

- Setting specific routes - can be set in the local security settings under "Routing" (SCALANCE S only), see Specifying routes (Page 502).
- Use standard router - can be set in the local security settings with "External interface [P1] red", "Internal interface [P2] green" or "DMZ interface [P3] yellow" (SCALANCE S623/ S627-2M only) see Configuring IP address parameters (Page 475).
  A maximum of one standard router can be used per security module.
- NAT/NAPT routing - can be set in the local security settings in "NAT / NAPT" (only for SCALANCE S and CP 343-1 Adv. / CP 443-1 Adv.).

Enabling routing mode (required for SCALANCE S modules only)

For this mode, you configure an internal IP address and an internal subnet mask for addressing the router in the internal subnet in the local security settings. All network requests that do not belong to a subnet are forwarded by the security module to a different subnet.

Note: In contrast to the bridge mode of the security module, VLAN tags are lost in routing mode.

1. Select the "Routing mode" option under "Mode" in the local security settings.
2. In the local security settings, enter an internal IP address and an internal subnet mask for addressing the router on the internal subnet in the input boxes under "Internal interface [P2] green" > Ethernet addresses".

Overview of NAT/NAPT

Requirements

- The security module is in routing mode or the DMZ interface is activated (SCALANCE S623 / S627-2M only).
- Since firewall rules that enable communication in the configured address translation direction are generated automatically for NAT / NAPT rules, the advanced firewall mode and the firewall in the editor for IP rules must be enabled for the security module. For more detailed information, refer to section Relationship between NAT/NAPT router and firewall (Page 423)

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "NAT / NAPT".
3. Call the appropriate editor with the "Open editor" button.
4. When required, configure address translation according to NAT (Network Address Translation) or NAPT (Network Address Port Translation).

Address translation with NAT (Network Address Translation)

NAT is a procedure for address translation between two address spaces.
The main task is to translate private addresses into public addresses; in other words into IP addresses that are used and even routed on the Internet. As a result, the IP addresses of the internal network are not known to the outside in the external network. The internal nodes are only visible to the external network by means of external IP address that is specified in the address translation list (NAT table). If the external IP address is not the address of the security module and if the internal IP address is unique, this is known as 1:1 NAT. With 1:1 NAT, the internal address is translated to this external address without port translation. Otherwise, n:1 NAT is being used.

Address translation with NAPT (Network Address Port Translation)

The address translation with NAPT changes the target address and the target port to a communication relation (port forwarding).

Frames coming from the external network or DMZ network and intended for the IP address of the security module are translated. If the destination port of the frame is identical to one of the values specified in the "Source port" column, the security module replaces the destination IP address and the destination port as specified in the corresponding row of the NAPT table. With the reply, the security module uses the values for the destination IP address and destination port as contained in the initial frame as the source IP address and the source port.

The difference to NAT is that with this protocol ports can also be translated. There is no 1:1 translation of the IP address. There is now only a public IP address that is translated to a series of private IP addresses with the addition of port numbers.

Address translation in VPN tunnels

Address translations with NAT/NAPT can also be performed for communications relations established via a VPN tunnel. This is supported for connection partners of the type SCALANCE S612 / S623 / S627-2M as of V4.

You will find further information on address translations in VPN tunnels in the following sections:

- NAT/NAPT routing (Page 411)
- Address translation with NAT/NAPT in VPN tunnels (Page 423)

Consistency check - these rules must be adhered to

Among other things, remember the following rules to obtain consistent entries:

- The IP address of the internal interface must not be used in the NAT / NAPT table.
- An IP address used in the NAT/NAPT address conversion list must not be a multicast or broadcast address.
- The external ports assigned for the NAPT translation are in the range > 0 and ≤ 65535. Port 123 (NTP), port 443 (HTTPS), port 514 (Syslog), port 161 (SNMP), ports 67+68 (DHCP) and ports 500+4500 (IPsec) are excluded if the relevant services are activated on the security module.
- The external IP address of the security module or the IP address of the DMZ interface may only be used in the NAT table for the action "Source NAT".
- Checking for duplicates in the NAT table
  An external IP address or an IP address in the DMZ network used in the direction "Destination NAT", "Source NAT +Destination NAT" or "Double NAT" may only be used once in each specified direction.
- Checking for duplicates in the NAPT table: A source port number may only be entered once for each interface.
- Internal NAPT ports can be in the range > 0 and ≤ 65535.

See also

Overview of the routing settings (Page 409)

NAT/NAPT routing

Address translation with NAT

The following control elements are available in the menu bar:

Operator control element	Function
	Enabling NAT Enables the editor for NAT rules. If the editor is enabled, the corresponding rules can be configured and are transferred to the module during loading. Since IP rules that enable communication in the configured address translation direction are generated automatically for NAT rules, the editor for NAT rules can only be enabled when advanced firewall mode is enabled with an active IP firewall.
	Insert rule above Inserts a rule above the selected rule / the selected rule set. To insert above a rule set, its first line must be selected.
	Insert rule below Inserts a rule below the selected rule / the selected rule set. To insert below a rule set, its first line must be selected.

Operator control element	Function
	Move up Moves the selected rule / the selected rule set one line up. To move a rule set, its first line must be selected.
	Move down Moves the selected rule / the selected rule set one line down. To move a rule set, its first line must be selected.
	Expand rule sets Shows existing rule sets expanded.
	Collapse rule sets Shows existing rule sets collapsed.
	Separate Converts the selected line into a separate rule or rules. If the first line of a rule set is selected, all rules of the rule set are converted into separate rules. If one rule contained in a rule set is selected, this is converted into a separate rule. In addition to this, the following rules of the rule set are converted into a separate rule set.

After creating NAT rules, the corresponding firewall rules are generated and displayed in advanced firewall mode, see section:

Relationship between NAT/NAPT router and firewall (Page 423)

If PPPoE is activated for the external interface or the DMZ interface, with the action "Destination NAT" the external interface or the DMZ interface cannot be selected as a towards direction. When configuring the action "Source NAT", the IP address cannot be entered in the "Source translation" input box because this is obtained dynamically during runtime.

Possible address translations for NAT

The following tables show the input options for address translation with NAT.

Action "Destination NAT" - "Redirect"

The action "Destination NAT" can be performed in the following direction:

- External to internal

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Destination NAT" can also be performed in the following directions:

- External to DMZ
- DMZ to internal
- DMZ to external

If the SCALANCE S module (only SCALANCE S612/S623/S627-2M as of V4) is in a VPN group and the tunnel interface is enabled, the "Destination NAT" action can also be performed in the following directions:

- Tunnel to internal
- Tunnel to external
- Tunnel to DMZ (only if the DMZ interface is activated)

The following applies, for example for the direction "external to internal": The destination IP address of a frame coming from the external network is checked to see whether it matches the IP address specified in the "Destination IP address" input box. If it matches, the frame is forwarded into the internal network by replacing the destination IP address of the frame with the IP address specified in the "Destination translation" input box. Access from external to internal using the external address is possible.

The following table shows the input required for the action "Destination NAT".

Box	Possible entries	Meaning
Source IP address	Not relevant for this action.	-
Source translation	Not relevant for this action.	-
Destination IP address	IP address in the source network	Destination IP address in the source network with which an IP address in the destination network will be accessed. The destination IP address must not match the IP address of the security module in the source network. If the destination IP address in a frame matches the address entered, the address is replaced by the corresponding IP address in the destination network. The specified destination IP address becomes the alias address. This means that the specified IP address is also registered as an IP address on the selected interface. Make sure that the alias address does not cause an IP address conflict in the network. The alias IP addresses of a security module are displayed under the entry "Alias IP addresses" of the relevant interface.
Destination translation	IP address in the destination network	The destination IP address is replaced by the IP address specified here.
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAT rule.

Action "Source NAT" - "Masquerading"

The action "Source NAT" can be performed in the following direction:

- Internal to external

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Source NAT" can also be performed in the following directions:

- Internal to DMZ
- External to DMZ
- DMZ to external

If the SCALANCE S module (only SCALANCE S612/S623/S627-2M as of V4) is in a VPN group and the tunnel interface is enabled, the "Source NAT" action can also be performed in the following directions:

- Internal to tunnel
- External to tunnel
- DMZ to tunnel (only if the DMZ interface is activated)

The following applies, for example for the direction "internal to external": The source IP address of a frame coming from the internal network is checked to see whether it matches the IP address specified in the "Source IP address" input box. If it matches, the frame with the external IP address specified in the "Source translation" input box is forwarded to the external network as a new source IP address. In the external network, the external IP address is effective.

The following table shows the input required for the action "Source NAT".

Box	Possible entries	Meaning
Source IP address	IP address in the source network	The source IP address of the specified node is replaced by the IP address specified in the "Source translation" input box.
	IP address range / IP address band in the source network	The IP addresses of the IP address range / IP address band are replaced by the IP address specified in the "Source translation" input box.
	*	The IP addresses of all nodes in the source network are replaced by the IP address specified in the "Source translation" input box.
Source translation	IP address in the destination network	Entry of the IP address that will be used as the new source IP address. If the IP address entered here is not the IP address of the security module, this becomes an alias address. This means that the specified address is also registered as an IP address on the selected interface. Make sure that the alias address does not cause an IP address conflict in the network. The alias IP addresses of a security module are displayed under the entry "Alias IP addresses" of the relevant interface.
Destination IP address	Not relevant for this action.	-
Destination translation	Not relevant for this action.	-
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAT rule.

Note

You can configure an address translation to the module IP address in the destination network for all frames going from a source network to a destination network. The security module also assigns a port number for each frame. This is an n:1 NAT address translation in which multiple IP addresses of the source network are translated to one IP address of the destination network.

Enter, for example, the following parameters for the direction "internal to external":

- Action: "Source NAT"
- From: "Internal"
- To "External"
- Source IP address: "*"
- Source translation: External IP address of the security module

Action "Source NAT + Destination NAT" - "1:1-NAT"

The action "Source NAT + Destination NAT" can be performed in the following direction:

- Internal to external

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Source NAT + Destination" can also be performed in the following directions:

- Internal to DMZ
- External to DMZ
- DMZ to external

If the SCALANCE S module (only SCALANCE S612/S623/S627-2M as of V4) is in a VPN group and the tunnel interface is enabled, the "Source-NAT + Destination-NAT" action can also be performed in the following directions:

- External to tunnel
- Internal to tunnel
- DMZ to tunnel (only if the DMZ interface is activated)

The following applies, for example for the direction "internal to external": When accessing from internal to external, the action "Source NAT" is performed. When accessing from external to internal, the action "Destination NAT" is performed.

The following table shows the input required for the action "Source NAT + Destination NAT":

Box	Possible entries	Meaning
Source IP address	IP address in the source network	The configuration is always specified in the source NAT direction. The IP addresses of the destination NAT direction are then inserted automatically by STEP 7.
Source translation	IP address in the destination network	
Destination IP address	Not relevant for this action.	
Destination translation	Not relevant for this action.	
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rules generated by STEP 7 for the NAT rule.

Action "Double NAT"

The action "Double NAT" can be performed for SCALANCE S modules in the following directions:

- Internal to external
- External to internal

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Double NAT" can also be performed in the following directions:

- Internal to DMZ
- External to DMZ
- DMZ to internal
- DMZ to external

In every direction, Source and Destination NAT always take place at the same time.
The following applies, for example for the direction "external to internal": When accessing from external to internal, the source IP address of the external node is replaced (Source NAT). Access to the internal network also uses the external IP address specified in the "Destination IP address" input box (Destination NAT).

You can, for example, use this action if a standard router other than the security module is entered for a device to be accessed using Destination NAT. Response frames from this device are then not sent to the entered standard router but to the corresponding interface of the security module.

The following table shows the input required for the action "Double NAT":

Box	Possible entries	Meaning
Source IP address	IP address in the source network	IP address of the node in the source network
Source translation	-	The Source NAT address translation is always to the IP address of the security module in the destination network. For this reason, the "Source translation" input box cannot be configured.

Box	Possible entries	Meaning
Destination IP address	IP address in the source network	Destination IP address in the source network with which an IP address in the destination network will be accessed. If the destination IP address in a frame matches the IP address entered, the IP address is replaced by the IP address specified in the "Destination translation" input box. If the IP address entered here is not the IP address of the security module, this becomes an alias address. This means that the specified address is also registered as an IP address on the selected interface. Make sure that the alias address does not cause an IP address conflict in the network. The alias IP addresses of a security module are displayed under the entry "Alias IP addresses" of the relevant interface.
Destination translation	IP address in the destination network	The destination IP address is replaced by the IP address specified here.
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAT rule.

Advanced NAT editor

After selecting a NAT rule, an advanced NAT editor appears in the Inspector window in "Properties > General" and is used for simplified configuration of address ranges for ports and IP addresses. Changes made in this editor are adopted in the selected NAT rule. In addition, selected NAT rules and NAT rule sets can be copied in the advanced NAT editor. To copy a NAT rule set, its first line must be selected. You can specify which of the copied NAT rules is used by adapting the IP rules generated for the NAT rules. Based on examples, the following section illustrates the configuration of address ranges in the advanced NAT editor.

Configuring address ranges in the advanced NAT editor

Address ranges can be configured in the text boxes of the "From" and "To" columns. If several address ranges are configured for a NAT rule, the number of addresses must match in each case. After configuring address ranges, STEP 7 creates a NAT rule set from the selected NAT rule that contains a corresponding NAT rule for each of the addresses.

In the text boxes for ports in the "To" column, the placeholder "*" can be used as an alternative. In this case, the port ranges and NAT rules affected are formed according to the other NAT parameters.

Address translation with NAPT

The following control elements are available in the menu bar:

Operator control element	Function
	Activate NAPT Enables the editor for NAPT rules. If the editor is enabled, the corresponding rules can be configured and are transferred to the module during loading. After selecting a NAPT rule, the editor for advanced NAPT settings is available In the "Properties" > "General" tab of the Inspector window.
	Insert rule above Inserts a rule above the selected rule / the selected rule set. To insert above a rule set, its first line must be selected.
	Insert rule below Inserts a rule below the selected rule / the selected rule set. To insert below a rule set, its first line must be selected.
	Move up Moves the selected rule / the selected rule set one line up. To move a rule set, its first line must be selected.
	Move down Moves the selected rule / the selected rule set one line down. To move a rule set, its first line must be selected.
	Expand rule sets Shows existing rule sets expanded.
	Collapse rule sets Shows existing rule sets collapsed.
	Separate Converts the selected line into a separate rule or rules. If the first line of a rule set is selected, all rules of the rule set are converted into separate rules. If one rule contained in a rule set is selected, this is converted into a separate rule. In addition to this, the following rules of the rule set are converted into a separate rule set.

After creating NAPT rules, the corresponding firewall rules are generated and displayed in advanced firewall mode, see section:
Relationship between NAT/NAPT router and firewall (Page 423)

The IP address translation with NAPT can be performed in the following direction:

- External to internal

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the IP address translation with NAPT can also be performed in the following directions:

- External to DMZ
- DMZ to internal
- DMZ to external

If the SCALANCE S module (only SCALANCE S612/S623/S627-2M as of V4) is in a VPN group and the tunnel interface is enabled, the IP address translation with NAPT can also be performed in the following directions:

- Tunnel to internal
- Tunnel to external
- Tunnel to DMZ (only if the DMZ interface is activated)

The following applies, for example for the direction "external to internal": Frames intended for the external IP address of the security module and for the port entered in the "Virtual port" column are forwarded to the specified destination IP address in the internal network and to the specified destination port.

The IP address from which the frames of forwarded into the target network is displayed in the column "Virtual IP address". With a direction from external this is the external IP address of the security module and with a direction from DMZ the IP address of the DMZ interface of the security module.

The following table shows the input required for address translation with NAPT in the "NAPT" tab.

Box	Possible entries	Meaning
Virtual port	TCP/UDP port or port range Example of entering a port range: 78:99	A node in the source network can send a frame to a node in the destination network by using this port number.
Destination IP address	IP address in the destination network	Frames intended for the IP address of the security module in the source network and the TCP/UDP port specified in the "Virtual port" box are forwarded to the IP address specified here.
Destination port	TCP/UDP port	Port number to which the frames coming from the source network are forwarded.
Protocol	• TCP+UDP • TCP • UDP	Selection of the protocol family for the specified port numbers
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAPT rule.

Advanced NAPT editor

After selecting a NAPT rule an advanced NAPT editor appears in the Inspector window in "Properties > General" and is used for simple configuration of address ranges for ports and IP addresses. Changes made in this editor are adopted in the selected NAPT rule. In addition to this, selected NAPT rules and NAPT rule sets can be copied in the advanced NAPT editor. To copy a NAPT rule set its first line must be selected. Which of the copied NAPT rules is used, can be specified by adapting the IP rules generated for the NAPT rules. Based on examples, the following section illustrates the configuration of address ranges in the advanced NAPT editor.

Configuration of address ranges in the advanced NAPT editor.

Address ranges can be configured in the text boxes of the "From" and "To" columns. If several address ranges are configured for a NAPT rule, the number of addresses must match in each case. After configuring address ranges, STEP 7 creates a NAPT rule set from the selected NAPT rule that contains a corresponding NAPT rule for each of the addresses.

The following example shows the configuration of address ranges for the virtual port, for the destination IP address and for the destination port.

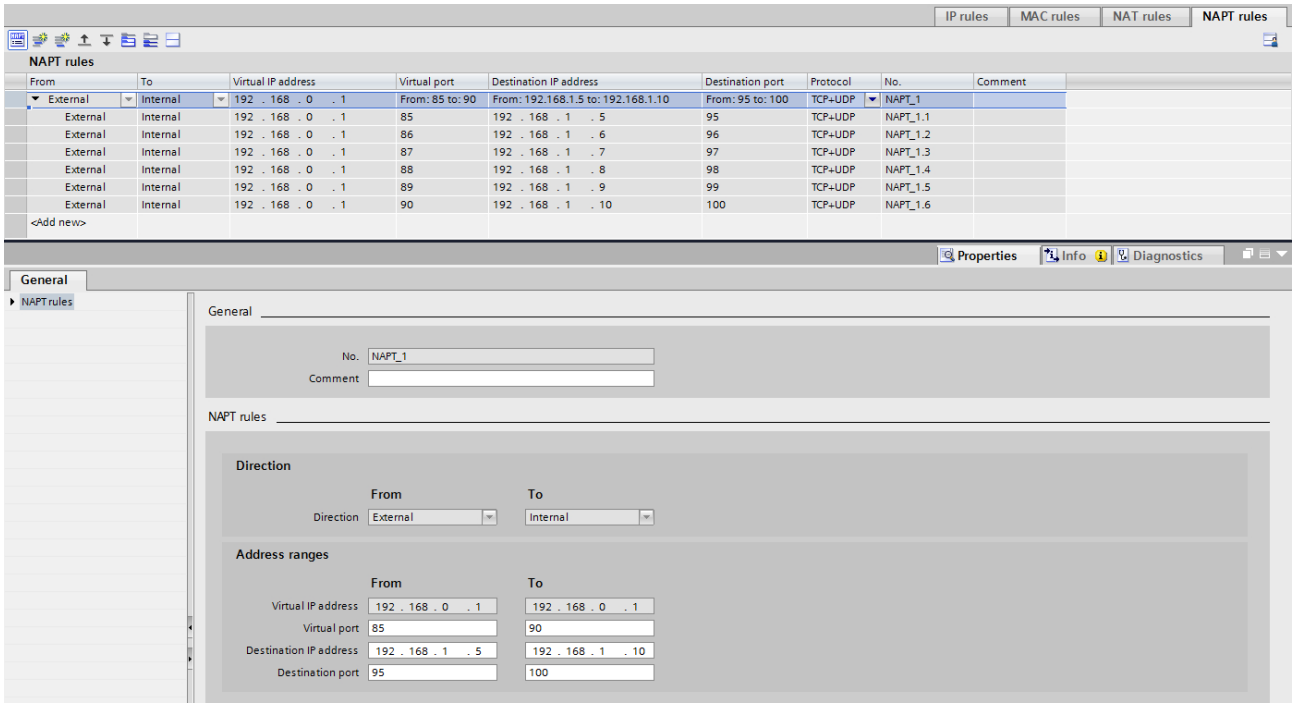

Figure 1-13 Advanced NAPT editor: Example 1

In the text boxes for ports in the "To" column, the placeholder "*" can be used as an alternative. In this case, the port ranges and NAPT rules affected are formed according to the other NAPT parameters.

In the following example the port ranges for the virtual port and the destination port are determined by STEP 7:

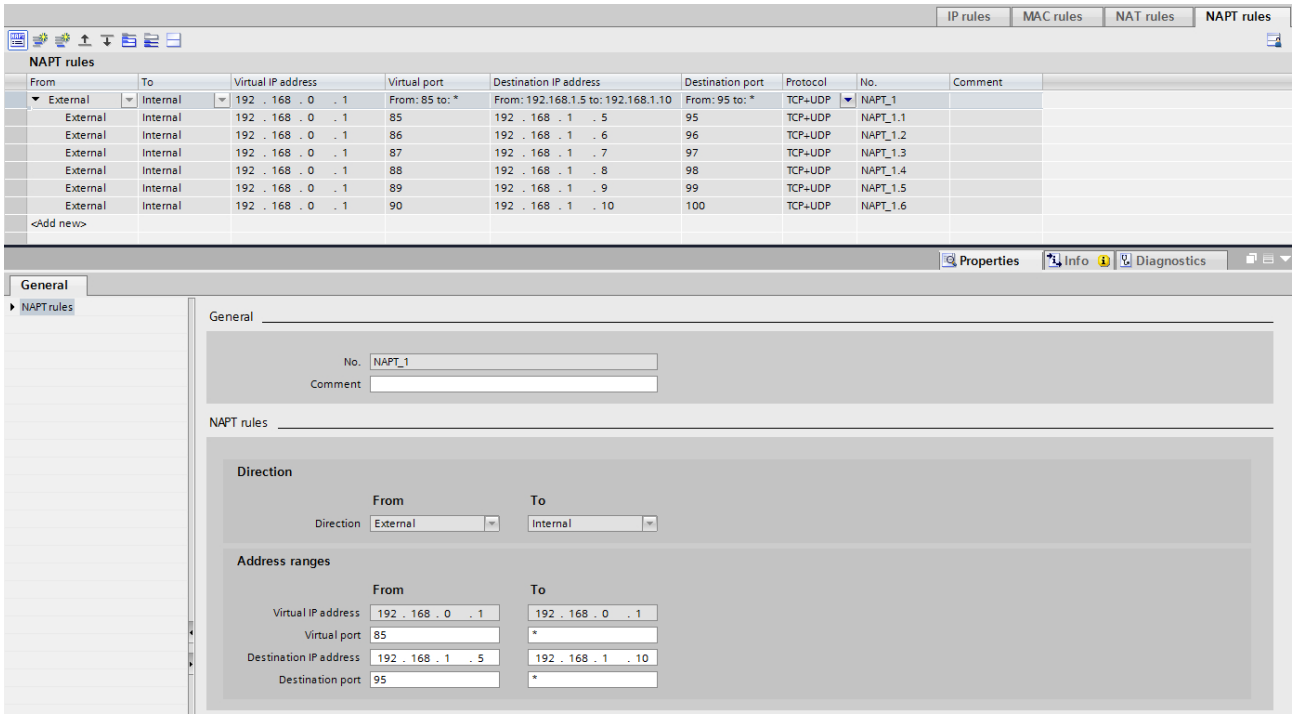

Figure 1-14 Advanced NAPT editor: Example 2

To configure the same port range for every NAPT rule of a NAPT rule set, you can specify the start and end port of the port range for the virtual port in the "From" column separated by a colon. For the destination port, such a port range cannot be specified. A placeholder "*" after the colon is interpreted as the maximum value 65535.

In the following example a port range with the explained notation is used for every NAPT rule of the NAPT rule set:

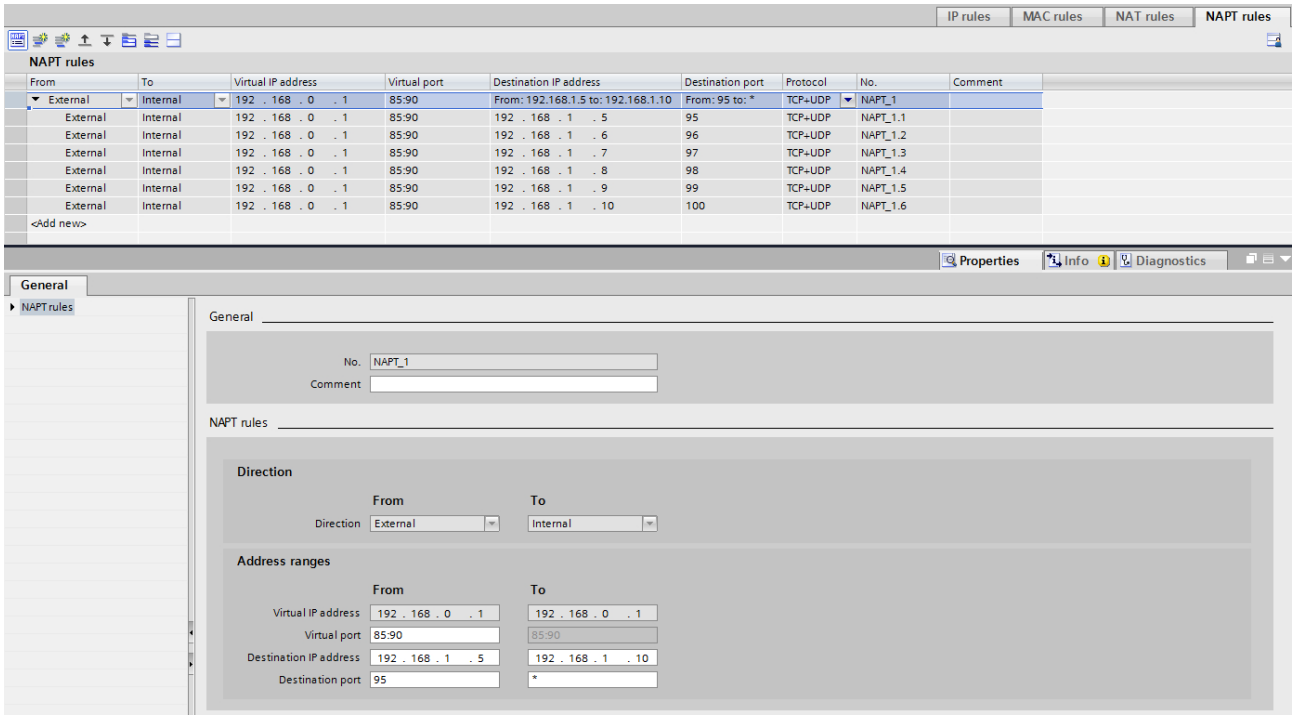

Figure 1-15 Advanced NAPT editor: Example 3 (NAPT rules)

In example 3, the virtual IP address and the virtual port are identical for every NAPT rule. In this case which NAPT rule is used is decided by the priority of the firewall rules, created by STEP 7 for the NAPT rules. The firewall rule that was created for the NAPT rule with the number NAPT_1.1 has the highest priority.

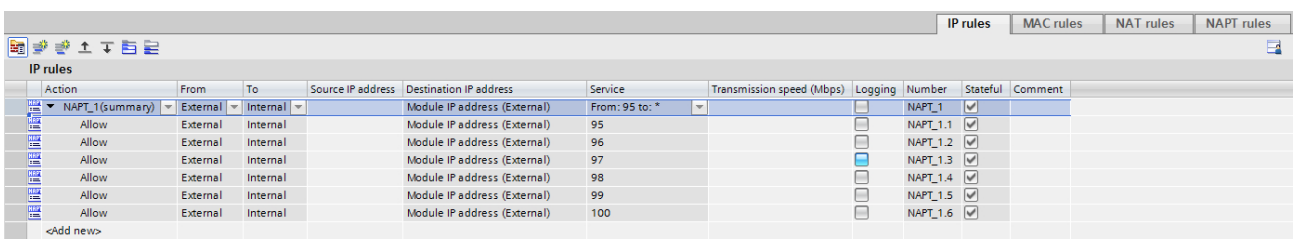

Figure 1-16 Advanced NAPT editor: Example 3 (firewall rules)

For this reason all frames intended for the IP address 192.168.0.1 and for a port in the range 85...90 are forwarded to destination IP address 192.168.1.5 and to destination port 95.

To uniquely define NAPT rules with the same virtual IP address and same virtual port / port range, source IP addresses can be configured in the relevant IP rules. The source IP addresses then decide which NAPT rule is used.

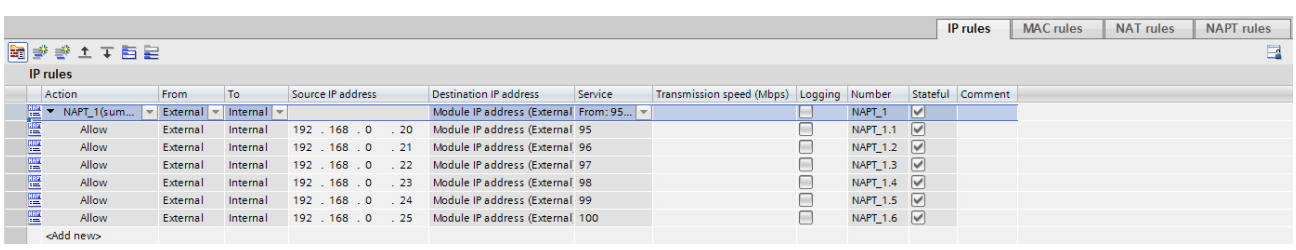

Figure 1-17 Advanced NAPT editor: Supplement to example 3

Address translation with NAT/NAPT in VPN tunnels

Module-specific function

Address translation with NAT/NAPT in VPN tunnels is only available for SCALANCE S612/S623/S627-2M modules as of V4, refer to the section:
Address translation with NAT/NAPT in VPN tunnels (Page 503)

Relationship between NAT/NAPT router and firewall

Meaning

After creating NAT/NAPT rules, STEP 7 automatically generates firewall rules that enable communication in the configured address translation direction. The generated firewall rules are visible as firewall rule sets in advanced firewall mode and can, if necessary, be expanded (additional IP address / IP address range / IP address band, services, transmission speed). In addition to this, the automatically generated firewall rules should be checked in terms of their priority and their position. If there are also manually configured firewall rules in the rule list that have higher priority than the automatically generated firewall rules, under certain circumstances no NAT / NAPT will be performed.

If there are several identical NAT / firewall pairs of rules, the priority in the firewall rule list decides which rule is used.

Firewall parameters generated by STEP 7 cannot be adapted. After deactivating NAT/NAPT, the firewall rules generated by STEP 7 are disabled.

To clarify the relationship between the NAT/NAPT rules and the corresponding firewall rules, they are identified by corresponding, consecutive numbers.

In firewall rules that were generated automatically for NAT/NAPT rules, the “Stateful” check box cannot be disabled.

The following table shows the system behind the firewall rules generated for NAT rules for SCALANCE S modules.

Table 1-35 NAT address translation and corresponding firewall rules for SCALANCE S modules

NAT action	Created firewall rule				
	Action	From	To	Source IP address	Destination IP address
Destination NAT	Accept	Source network	Destination network	-	IP address specified in the "Destination IP address" input box
Source NAT	Accept	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	-

NAT action	Created firewall rule				
	Action	From	To	Source IP address	Destination IP address
Source NAT + Destination NAT	Accept	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	-
	Accept	Destination network	Source network	-	IP address that was inserted in the "Destination IP address" input box by STEP 7
Double NAT	Accept	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	IP address specified in the "Destination IP address" input box
	Accept	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	IP address of the node specified in the "Destination translation" input box

The following table shows the system behind the firewall rules generated for NAT rules for CP 343-1 Adv. / CP 443-1 Adv.

Table 1-36 NAT address translation and corresponding firewall rules for CP 343-1 Adv. / CP 443-1 Adv.

NAT action	Created firewall rule				
	Action	From	To	Source IP address	Destination IP address
Destination NAT	Drop	External	Station	-	IP address of the security module in the external network
	Accept	External	Any	-	IP address of the node specified in the "Destination translation" input box
Source NAT	Accept	Any	External	IP address specified in the "Source translation" input box	-

NAT action	Created firewall rule				
	Action	From	To	Source IP address	Destination IP address
Source NAT + Destination NAT	Accept	Any	External	IP address specified in the "Source translation" input box	-
	Drop	External	Station	-	IP address of the security module in the external network
	Accept	External	Any	-	IP address of the node that was inserted in the "Destination translation" input box by STEP 7

The following table shows the system behind the firewall rules generated for NAT rules for SCALANCE S modules.

Table 1-37 NAPT translation and firewall rules created for SCALANCE S modules

Created firewall rule					
Action	From	To	Source IP address	Destination IP address	Service
Accept	Source network	Destination network	-	IP address of the security module in the source network	[Service_NAPT_rule]

The following table shows the system behind the firewall rules generated for NAPT rules for CP 343-1 Adv. / CP 443-1 Adv.

Table 1-38 NAPT translations and created firewall rules for CP 343-1 Adv. / CP 443-1 Adv.

Created firewall rules					
Action	From	To	Source IP address	Destination IP address	Service
Drop	External	Station	-	IP address of the security module in the external network	[Service_NAPT_rule]
Accept	External	Any	-	IP address of the security module in the external network	[Service_NAPT_rule]

Stateful packet inspection

The firewall and NAT/NAPT router supports the "Stateful Packet Inspection" mechanism. As a result, reply frames can pass through the NAT/NAPT router and firewall without it being necessary for their addresses to be included extra in the firewall rule and the NAT/NAPT address translation.

Relationship between NAT/NAPT router and user-specific firewall

Module-specific function

The configuration of NAT/NAPT rules in the user-specific firewall is only available for SCALANCE S modules as of V3, refer to the section:

Relationship between NAT/NAPT router and user-specific firewall (Page 505)

Configuring time-of-day synchronization

Procedure for time-of-day synchronization

Meaning

For the time synchronization of the transfer of productive data, the date and time are maintained on the security module to verify the validity (time) of a certificate and for time stamping log entries. For the Security functions the time of day of the security modules of a project must be synchronized.

Note

Configuring the firewall for communication with NTP servers

If the NTP server cannot be reached by the security module, you will need to allow the frames from the NTP server explicitly in the firewall (UDP, port 123).

Note

Time-of-day synchronization relates solely to the security module and cannot be used to synchronize devices in the internal network of the security module.

Some CPs can forward the time of day to other module of the station.

The CP 443-1 OPC UA can only receive the time-of-day in the SIMATIC mode from the station but cannot forward it.

Note

Before the security functions of a CP (time-of-day slave) are used, the CP must receive a valid time-of-day frame from the time master.

Configuring time-of-day synchronization for a security module

How to access this function

1. Select the module to be edited.
2. Select the "Time-of-day synchronization" entry in the local security settings.
3. Select the "Activate time-of-day synchronization" check box.

Procedure for time-of-day synchronization

Depending on the security module being used, the following methods can be configured:

Procedure	Meaning
Set time with each download	The module time is set automatically to the PC time when a configuration is downloaded.
SIMATIC	Time-of-day synchronization by received MMS time-of-day messages (MMS = Manufacturing Message Specification) For the CP select whether the CP is to adopt or forward the time of day. The CP 1628 always forwards the time of day. The CP 443-1 OPC UA uses the SIMATIC mode if no time-of-day synchronization using NTP is configured. It can only receive the time-of-day from the station but cannot forward it. Available directions: • Automatic The CP receives the time from the station or from the LAN and forwards it to the station or to the LAN. If several CPUs are being operated in the station, the default setting can lead to collisions. To avoid this, you should specifically define the forwarding direction. • From station The CP forwards the time of day to the LAN. • From LAN The CP forwards the time of day from the LAN to the station. If forwarding of the time of day is enabled, you can use the "Use corrected time" check box to specify whether or not a correction factor included in the time-of-day frame is used. For the CP 1628, this option is is the default and it cannot be disabled.

Procedure	Meaning
Time of day from partner	The time is obtained from the relevant communications partner. Synchronization cycle: Specifies the cycle for time-of-day synchronization. For the synchronization cycle of the CP, and individual hour or minute interval can be specified.
NTP / NTP (secure)	Time-of-day synchronization by an NTP server / NTP server of the type "NTP (secure)". • Time zone: In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred. This corresponds to GMT (Greenwich Mean Time). The time offset from UTC can be set by configuring the local time zone. • Update interval in seconds: Specifies the interval between the time queries in seconds. For SCALANCE S as of V3, the time interval for querying the NTP server is specified automatically. If the "Enable security functions" check box is enabled in the local security settings of a CP, the setting for the update interval is transferred from the CP's local settings into the CP's local security settings. • Time synchronization on the full minute: With this option, you specify that the time of day is forwarded to the communications bus on the full minute. This option is required for certain special applications. • Accept time from non-synchronized NTP servers Here you can specify whether the security module also accepts the time-of-day from non-synchronized NTP servers. • Forward time of day to station: Disable this option if the CPU requests the time separately from an NTP server. This prevents the time on the CPU obtained directly from the NTP server from being overwritten by the time detected in the CP. The accuracy may be reduced slightly due to forwarding via the CP. • NTP server: Creating NTP servers in the local security settings is described in the section Configuring NTP servers (Page 429).

Configuring NTP servers

Creating NTP servers of the type "NTP (secure)" in the global security functions

In the global security functions only NTP servers of the type "NTP (secure)" can be created and assigned to CPs or SCALANCE S modules as of V4.

1. In the global security functions, select the entry "NTP".
   Result: The previously created NTP servers (secure) are displayed under the selected entry.
2. Double-click on the "Add new NTP server" entry.
   Result: The created NTP server (secure) is displayed with an automatically assigned number below the entry "NTP".
3. Double-click on the created NTP server (secure).
   Result: In the working area the assignment dialog for NTP servers (secure) is displayed. The selected NTP server (secure) is selected in the "NTP server" drop-down list. The required security modules can be assigned to this, see section Assigning the security module to an NTP server (secure) (Page 431). In the local security settings, the configurable properties of the NTP server (secure) are displayed.

4. Enter the following data in the "General" entry in the local security settings:
   – Name: A maximum of 25 characters can be used for the name of the NTP server (secure).
   – IP address / FQDN: Address of the NTP server (secure).
5. Enter the following encryption parameters in the "NTP key" entry in the local security settings:

Property	Meaning
Key ID	Numeric value between 1 and 65534.
Hash algorithm	Select the hash algorithm. Observe the length for the NTP key depending on the selection.
Hex/ASCII	Select the format for the NTP key.
Key	Enter the NTP key. SCALANCE S-600 V3 / S7-CPs Selection MD5 or SHA-1: • Hex: 10 ... 40 characters • ASCII: 5 ... 20 characters TIM, CP 1628 Selection MD5 or SHA-1: • Hex: 22 ... 40 characters • ASCII: 11 ... 20 characters SCALANCE SC-600 Selection MD5: • ASCII: 16 ... 128 characters Selection SHA1: • ASCII: 20 ... 128 characters Selection DES: • ASCII: 8 characters The numbers of characters listed cover all key lengths supported by the security modules.

To switch over to the local security settings of another NTP server (secure), select this from the "NTP server" drop-down list of the assignment dialog or double-click on the corresponding entry in the global security functions.

Result

You have created the NTP server (secure) and can now assign it to the required security modules.
Note the descriptions in the following section:
Assigning the security module to an NTP server (secure) (Page 431)

Creating non-secure NTP servers in the local security settings

Non-secure NTP servers for SCALANCE S modules below V4 and CPs need to be created in the local security settings.

1. Select the module to be edited.
2. Select the "Time-of-day synchronization" entry in the local security settings.
3. Select the synchronization method..
4. Enter a name and the IP address of the NTP server. A maximum of 25 characters can be used for the name. If you have selected the "NTP (secure)" synchronization mode, you can select an NTP server (secure) that you created in the global security functions in the "Name" column.

Configuration limits for NTP servers

You can assign a maximum of 4 NTP servers to one security module.

Conventions for names of NTP servers

The characters 0-9, a-z, A-Z, -._ can be used for the names of NTP servers. A name must start with a letter and end with a letter or number. Before the '.' character only letters and numbers are permitted, after the '.' character only letters. Before the '_' or '-* character only letters and numbers are permitted.

Importing/exporting NTP servers (secure)

Using the "Import" or "Export" commands in the shortcut menu, you can export the key list of the currently selected NTP server (secure) in the global security functions and import the file into an NTP server (secure) or vice versa.

Assigning the security module to an NTP server (secure)

Requirement

- You have defined an NTP server (secure) in the global security functions.
- "NTP" or "NTP (secure)" is selected as the time-of-day synchronization mode in the local security settings of the security module that you want to assign to an NTP server (secure).

Procedure

1. Double-click on the "NTP" entry in the global security functions.
2. Double-click the entry "Assign module to an NTP server".
3. From the "NTP Server" drop-down list, select the NTP server (secure) to which you want to assign a security module.

4. In the "Available modules" section, select the security module that you want to assign to the selected NTP server (secure).
5. Click the "<<" button to assign the selected security module to the selected NTP server (secure).

Result

You have assigned the security module to the NTP server (secure). The NTP server (secure) is displayed automatically in the local security settings in the list of NTP servers.

Security module as DHCP server

Module-specific function

The use of the security module as a DHCP server is only possible with SCALANCE S modules, see section:
AUTOHOTSPOT

Configuring SNMP

Overview of SNMP

What is SNMP?

The security module supports the transfer of management information using the Simple Network Management Protocol (SNMP). For this purpose, an SNMP agent that receives and responds to SNMP requests is installed on the security module. The information on the properties of SNMPcompliant devices is entered in MIB files (MIB = Management Information Base) for which the user must have the required rights.

In SNMPv1, the "community string" is also sent. The "community string" is like a password that is transmitted along with the SNMP request. If the community string is correct, the security module replies with the required information. If the string is incorrect, the security module discards the query and does not reply. The community string is transmitted via SNMPv1 without encryption.

SNMPv3 lets you transmit encrypted data.

Configuring SNMP - "SNMP" entry

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "SNMP".
3. Activate the "Activate SNMP" check box.

4. Select one of the following SNMP protocol versions:

   Note
   Encrypted data transmission with SNMPv3
   You should use SNMPv3 to transmit data in encrypted form in order to enhance security.

   – SNMPv1
     The security module uses the following default values for the community strings to control the access rights in the SNMP agent. These default values should be adapted to increase security.
     For read access: public
     For read and write access: private
     To enable write access using SNMP, select the "Allow write access" option.
   – SNMPv3
     Select either only an authentication algorithm or an authentication algorithm and an encryption algorithm.
     Authentication algorithm: none, MD5, SHA-1
     Encryption algorithm: none, AES-128, DES

     Note
     Preventing the use of DES
     DES is an insecure encryption algorithm. Therefore, it should only be used for reasons of down compatibility.

     Note
     When using SNMPv3 no RADIUS authentication is possible.

5. If SNMPv3 is to be used, assign a user a role with corresponding activated SNMP rights to enable access to the module via SNMP. An overview of SNMP rights is available in the section:
   Special features of user management for security functions (Page 360).
6. In the "Advanced settings" area, for SCALANCE S modules configure module-specific information about the author, location and e-mail address overwriting the information from the project properties.
   The following applies to values written to the security module by an SNMP tool using an SNMP-SET command:
   If you select the "Keep values written by SNMP set" check box, the values are not overwritten when there is another download of a STEP 7 configuration to the security module.

Configuring proxy ARP

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Configuring proxy ARP (Page 509)

Activate Web server on security module

Module-specific function

This function is only available for CP 343-1 Advanced / CP 443-1 Advanced and CP 443-1 OPC UA see section:
Activating Web server on security module (Page 535)in the section "Security for S7-300 / S7-400 / PC CPs".

IPsec VPN tunnel: Creating and assigning VPN groups

How to create an IPsec tunnel with VPN groups

Requirement

Note

Current date and current time of day on the security modules

When using secure communication (for example, HTTPS, VPN...), make sure that the security modules involved have the current time of day and the current date. Otherwise the certificates used are not evaluated as valid and the secure communication does not work.

Creating a VPN group

1. In the global security functions, select the entry "VPN groups".
   Result: The previously created VPN groups are displayed under the selected entry.
2. Double-click on the "Add new VPN group" entry.
   Result: The created VPN group is displayed with an automatically assigned number below the entry "VPN groups".
   As an alternative, in the network view you can right click on an interface of a module with VPN capability and create a VPN group with the shortcut menu command "Add new VPN group" (not possible for SOFTNET Security Client). The selected security module is automatically assigned to the created VPN group.
3. Double-click on the created VPN group.
   Result: In the working area the assignment dialog for VPN groups sets is displayed. The selected VPN group is selected in the "VPN" drop-down list. Security modules can be assigned to this in the relevant assignment dialog, see section Assigning a security module to a VPN group (Page 437). In the local security settings, the VPN group properties of the selected VPN group can be adapted, see section Configuring VPN group properties (Page 439).

Conventions for names of VPN groups

The characters 0-9, a-z, A-Z, -._ can be used for the names of VPN groups. A name must start with a letter and end with a letter or number. Before the '.' character only letters and numbers are permitted, after the '.' character only letters. Before the '_' or '-* character only letters and numbers are permitted.

Display of the VPN groups with their properties

If you select a security module that is in one or more VPN group(s), the properties of the VPN group(s) of which the security module is a member are shown in the "Network data" area.

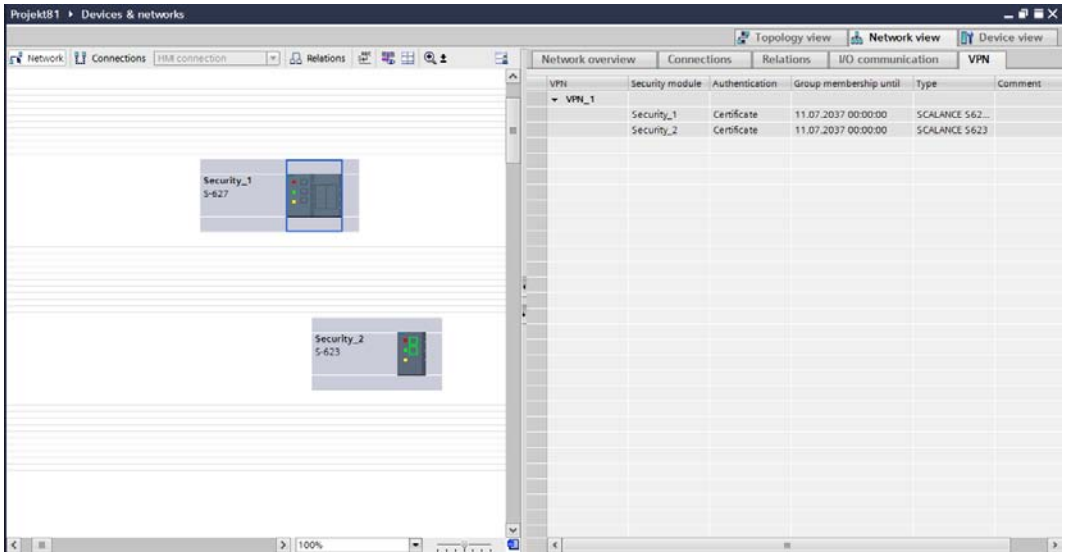

The following properties of the VPN groups are displayed in columns in the "VPN" tab of the "Network data" area:

Property/column	Meaning
VPN	Names of the VPN groups of which the selected security module is member
Security module	Names of the assigned security modules
Authentication	Type of authentication: Pre-shared key or certificate
Group membership until	Date and time up to which the VPN group certificate of the security module is valid
Type	Model numbers of the assigned security modules
Comment	Comment

Setting the life of certificates

Open the dialog in which you can set the expiry date of the certificate as follows:

1. Select the VPN group you want to edit in the "VPN" tab.
2. In the "Properties" > "General" tab of the Inspector window, select the entry "Authentication".

Note

Expiry of a certificate

Communication through the VPN tunnel continues after the certificate has expired until the tunnel is terminated or the SA lifetime expires. For more information on certificates, refer to section:
AUTOHOTSPOT.

Configuration limits

Number of the VPN tunnel	
SCALANCE S612 V3/V4	Maximum of 128
SCALANCE S623 V3/V4	Maximum of 128
SCALANCE S627-2M V4	Maximum of 128
CP 343-1 Advanced / CP 443-1 Advanced	Maximum of 32
CP 1628	Maximum of 64
S7-1200 CPs/ET 200SP CPs	Maximum of 8
S7-1500 CPs	Maximum of 16

For the number of communications relations between subnets of the security module and subnets of other security modules that can be reached via VPN tunnel the following configuration limits apply:

Number of subnet relations	
SCALANCE S612 V3/V4	Maximum of 128
SCALANCE S623 V3/V4	Maximum of 128
SCALANCE S627-2M V4	Maximum of 128
CP 343-1 Advanced / CP 443-1 Advanced	Maximum of 32
CP 1628	Maximum of 64
S7-1200 / S7-1500 CPs	Maximum of 8

Assigning a security module to a VPN group

Requirements

- You have defined a VPN group in the global security functions. Before assigning security modules to VPN groups, note the rules for this in the section Modes of VPN groups (Page 449).
- Before assigning the following modules to VPN groups, VPN must first be enabled and the VPN connection type "Internet Key Exchange (IKE)" must be selected:
  - CP 1243-1 as of V3.1
  - CP 1243-8 IRC as of V3.1
  - CP 1243-7 LTE as of V3.1
  - CP 1543SP-1 as of V2

Procedure

1. Double-click on the "VPN groups" entry in the global security functions.
2. Double-click on the "Assign module to a VPN group" entry.
3. From the "VPN" drop-down list, select the VPN group to which you want to assign a security module.
4. In the "Available modules" section, select the security module that you want to assign to the selected VPN group.
5. Click the "<<" button to assign the selected security module to the selected VPN group.
   If the security module doe not support one or more properties of the VPN group box, the module symbol in the area of the assigned modules is displayed with an error symbol .
   As an alternative to the procedure , in the network view you can assign a security module to an existing VPN group by right clicking on an interface of a module with VPN capability and assign this to the VPN group using the shortcut menu "Assign module to a VPN group" (not possible for SOFTNET Security Client).

Result

You have assigned the security module to the VPN group.

Authentication methods

The following methods are available:

The authentication method is specified per VPN group and decides the type of authentication used.

The following key-based or certificate-based authentication methods are supported:

- Pre-shared key
  Authentication is achieved using a previously specified character string that is distributed to all modules in the group.
  Enter a pre-shared key in the "Key" field under "Authentication" > "General" in the VPN group properties dialog. If you click the "Renew preshared key" button, STEP 7 renews the pre-shared key to be used.
- Certificate
  Certificate-based authentication "Certificate" is the default setting. The procedure is as follows:
  - When a group is created, a CA certificate is generated as a root certificate.
  - Each security module that is a member of the VPN group also receives a group certificate signed with the key of the CA certificate.

  If you click the "Renew certificate " button, STEP 7 renews the CA certificate of the VPN group.
  All certificates are based on the ITU standard X.509v3 (ITU, International Telecommunications Union).
  The certificates are generated by a certificate authority contained in STEP 7.

Note

Restriction in VLAN operation

No VLAN tagging is transmitted in IP frames via the VPN tunnel of the security module. The VLAN tags included in IP frames are lost when they pass through the security modules because IPsec is used to transfer the IP frames.

As default, no IP broadcast or IP multicast frames can be transferred with IPsec through a layer 3 VPN tunnel. Through a layer 2 VPN tunnel of the security module, IP broadcast or IP multicast frames are packaged just like MAC packets including the Ethernet header in UDP and transferred. With these packets, the VLAN tagging is therefore retained.

VPN group properties

VPN connection establishment by Known Peers and Unknown Peers

The properties of a VPN group are made up of the selected authentication method and the configured parameters for IKE phase 1 and 2 and during loading are transferred to all VPN group nodes. On the responder the IP addresses of the security modules that may initiate the VPN connection establishment with the configured properties of the VPN group are stored in addition. When a security module attempts to establish a VPN connection to the responder, this first checks whether the IP address of the initiator is known to it. If this is the case, the initiator is a Known Peer. If the VPN group properties proposed by this Known Peer match those stored on the responder for the Known Peer, the responder allows the VPN connection establishment. In addition to the properties of a VPN group, fallback VPN profiles can be assigned to a VPN group. VPN profiles contain predefined properties for the authentication method, IKE phase 1 and 2. Fallback VPN profiles can be used for SCALANCE S-600 V3, CP 343-1 Adv. / CP 443-1 Adv. and CP 1628 as alternative configurations for VPN connection establishment. These come into play when VPN properties of the initiator do not match the VPN properties of the responder. In such cases, fallback VPN profiles ensure the VPN connection establishment. The configured fallback VPN profiles just as the properties of the VPN group are transferred to all VPN group nodes. The priority for the use of the fallback VPN profiles is specified by their order in the configuration. The fallback VPN profiles and the properties of the VPN group must not differ in the authentication method.

If the IP address of the initiator is not known to the responder because the initiator in productive mode for example is connected via a NAT translation, the initiator is an Unknown Peer (Road Warrior). As default the responder allows connection establishment by Unknown Peers. In the module-specific VPN properties of the responder you can specify whether and with which VPN group properties connection establishment by Unknown Peers is allowed by the responder as long as it is a module of the type SCALANCE S-600 V3, CP 343-1 Adv. / CP 443-1 Adv. or CP 1628, see section Configuring module- and connection-specific VPN settings (Page 453).

Configuring VPN group properties

VPN group properties

Note

Knowledge of IPsec necessary

To be able to set these parameters, you require IPsec experience. If you do not make or modify any settings, the defaults apply.

The following settings can be configured in the properties of a VPN group:

- Automatically set compatible VPN configuration (entry: "Settings")
- Authentication method (entry: "General")
- IKE settings (entry: "Advanced settings phase 1")
- IPsec settings (entry: "Advanced settings phase 2")
- Fallback VPN profiles

If the VPN group properties you have configured are less secure than the recommended properties, STEP 7 makes this known to you with an appropriate message text.

How to access this function

1. In the global security functions select the VPN group with properties you want to configure under the entry "VPN groups".
2. Select "Open" in the shortcut menu of this entry.
   Result: The VPN group properties appear in the area of the local security settings under "Authentication".
3. Under the "Settings" entry, select whether the compatible VPN configuration should be set automatically. When the function is activated, the settings of the individual group members according to the supported device functions are changed so that they correspond to the optimum security requirements.
4. Under the "General" entry, select whether to use a pre-shared key or certificate for authentication. For more detailed information, refer to section:
   Authentication methods (Page 437).
5. Select the IKE version. Depending on the selection, various options are available in the "Advanced settings".

Advanced settings phase 1

Phase 1: IKE negotiation of the Security Association (SA) for phase 2:

Here you set the parameters for negotiating the security parameters to be used in phase 2:

The following VPN group properties are supported by the individual modules:

Module	IKE version	Phase 1				
		IKE mode	DH group	SA lifetime	Encryption	Authentication
CP 1543-1 (as of V2.1) CP 1543SP-1 (V2.0) CP 1243-1 BX30 (as of V3.0) CP 1243-7 LTE BX30 (as of V3.0) CP 1243-8 IRC BX30 (as of V3.0)	IKEv1	Main	Group 1 Group 2 Group 5 Group 14 Group 15	Time: 1440 … 2500000 minutes	AES-256, 192, 128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-256: Secure Hash Algorithm 256 SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5
CP 1543-1 (up to 2.0) CP 1543SP-1 (V1.0) CP 1243-1 BX30 (up to V2.1) CP 1243-7 LTE BX30 (V2.1) CP 1243-8 IRC BX30 (V2.1)	IKEv1	Main	Group 1 Group 2 Group 5 Group 14	Time: 1440 … 2500000 minutes	AES-256, 192, 128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5
CP 343-1 GX31 Advanced CP 443-1 GX30 Advanced CP 443-1 OPC UA CP 1628 SCALANCE S-600 as of V3	IKEv1	Main	Group 1 Group 2 Group 5 Group 14	Time: 1440 … 2500000 minutes	AES-256, 192, 128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5
SOFTNET Security Client V4, V5 (WinXP)	IKEv1	Main	Group 2 Group 14	Time: 1440 … 2879 minutes	3DES-168**: Triple DES*** DES	SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5
Security device configured as SOFTNET Security Client V5 (Win7)	IKEv1	Main	Group 2 Group 14	Time: 1440 … 2879 minutes	AES-256, 192, 128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5
MD741-1 V1.1, M875 V2.0	IKEv1	Main	Group 2	Time: 1440 … 2500000 minutes	3DES-168**: Triple DES***	SHA-1***: Secure Hash Algorithm 1

Module	IKE version	Phase 1				
		IKE mode	DH group	SA lifetime	Encryption	Authentication
Security device configured as VPN client VPN server	IKEv1	Main Aggressive	Group 1 Group 2 Group 5 Group 14 Group 15	Time: 10 ... 1666666 minutes	AES-256, 192, 128: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-512: Secure Hash Algorithm 512 SHA-384: Secure Hash Algorithm 384 SHA-256: Secure Hash Algorithm 256 SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5
	IKEv2	Main	Group 1 Group 2 Group 5 Group 14 - Group 18	Time: 10 ... 1666666 minutes	AES-256, 192, 128*: Advanced Encryption Standard (CBC, CTR, CCM 16 or GCM 16 mode) 3DES-168**: Triple DES***	SHA-512: Secure Hash Algorithm 512 SHA-384: Secure Hash Algorithm 384 SHA-256: Secure Hash Algorithm 256 SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5
Security device configured as NCP VPN client	IKEv1	Main Aggressive	Group 1 Group 2 Group 5 Group 14 Group 15	Time: 10 … 2500000 minutes	AES-256, 192, 128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-512: Secure Hash Algorithm 512 SHA-384: Secure Hash Algorithm 384 SHA-256: Secure Hash Algorithm 256 SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5

Module	IKE version	Phase 1				
		IKE mode	DH group	SA lifetime	Encryption	Authentication
SCALANCE SC-600 SCALANCE S615	IKEv1	Main Aggressive	Group 1 Group 2 Group 5 Group 14 Group 15	Time: 10 … 2500000 minutes	AES-256, 192, 128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-512: Secure Hash Algorithm 512 SHA-384: Secure Hash Algorithm 384 SHA-256: Secure Hash Algorithm 256 SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5
	IKEv2	Main	Group 1 Group 2 Group 5 Group 14 - Group 18	Time: 10 … 2500000 minutes	AES-256, 192, 128*: Advanced Encryption Standard (CBC, CTR, CCM 16 or GCM 16 mode) 3DES-168**: Triple DES***	SHA-512: Secure Hash Algorithm 512 SHA-384: Secure Hash Algorithm 384 SHA-256: Secure Hash Algorithm 256 SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5

* 256 bit, 192 bit or 128 bit key length

** 168 bit key length

*** The setting is classified as less secure. It is recommended when possible that you use a more secure setting.

VPN group properties for phase 2

Phase 2: IKE negotiation of the Security Association (SA) for IPsec data exchange:

Here, you set the parameters for negotiating the security parameters used for the IPsec data exchange with ESP (Encapsulating Security Payload) and AH (Authentication Header). Communication in phase 2 is already encrypted.

The following VPN group properties are supported by the individual modules:

Module	IKE version	Phase 2			
		SA lifetime	Encryption	Authentication	Perfect Forward Secrecy
CP 1543-1 (as of V2.1) CP 1543SP-1 (V2.0) CP 1243-1 BX30 (as of V3.0) CP 1243-7 LTE BX30 (as of V3.0) CP 1243-8 IRC BX30 (as of V3.0)	IKEv1	Time: 60 … 16666666 minutes Limit: 2000 ... 500000 MB	AES-256, 192, 128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-256: Secure Hash Algorithm 256 SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.
CP 1543-1 (up to 2.0) CP 1543SP-1 (V1.0) CP 1243-1 BX30 (up to V2.1) CP 1243-7 LTE BX30 (V2.1) CP 1243-8 IRC BX30 (V2.1)	IKEv1	Time: 60 … 16666666 minutes Limit: 2000 ... 500000 MB	AES-256, 192, 128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.
CP 343-1 GX31 Advanced CP 443-1 GX30 Advanced CP 443-1 OPC UA CP 1628 SCALANCE S-600 as of V3	IKEv1	Time: 60 … 16666666 minutes Limit: 2000 ... 500000 MB	AES-192, 128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.
SOFTNET Security Client V4, V5 (WinXP)	IKEv1	Time: 60 … 2879 minutes	3DES-168**: Triple DES*** DES	SHA-1***: Secure Hash Algorithm 1	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.

Module	IKE version	Phase 2			
		SA lifetime	Encryption	Authentication	Perfect Forward Secrecy
Security device configured as SOFTNET Security Client V5 (Win7)	IKEv1	Time: 60 … 2879 minutes	AES-128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.
MD741-1 V1.1, M875 V2.0	IKEv1	Time: 60 … 16666666 minutes	3DES-168**: Triple DES***	SHA-1***: Secure Hash Algorithm 1	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.
Security device configured as VPN client VPN server	IKEv1	Time: 10 ... 16666666 minutes Limit: 10 ... 4096 MB	AES-128*: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-512: Secure Hash Algorithm 512 SHA-384: Secure Hash Algorithm 384 SHA-256: Secure Hash Algorithm 256 MD5***: Message Digest Algorithm 5	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.
	IKEv2	Time: 10 ... 16666666 minutes Limit: 10 ... 4096 MB	AES-256, 192, 128*: Advanced Encryption Standard (CBC, CTR, CCM 16 or GCM 16 mode) 3DES-168**: Triple DES***	SHA-512: Secure Hash Algorithm 512 SHA-384: Secure Hash Algorithm 384 SHA-256: Secure Hash Algorithm 256 MD5***: Message Digest Algorithm 5	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.

Module	IKE version	Phase 2			
		SA lifetime	Encryption	Authentication	Perfect Forward Secrecy
Security device configured as NCP VPN client	IKEv1	Time: 10 ... 16666666 minutes Limit: 10 ... 4096 MB	AES-128: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES*** DES	SHA-512: Secure Hash Algorithm 512 SHA-384: Secure Hash Algorithm 384 SHA-256: Secure Hash Algorithm 256 MD5***: Message Digest Algorithm 5	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.
SCALANCE SC-600 SCALANCE S615	IKEv1	Time: 10 ... 16666666 minutes Limit: 10 ... 4096 MB	AES-128: Advanced Encryption Standard (CBC mode) 3DES-168**: Triple DES***	SHA-512: Secure Hash Algorithm 512 SHA-384: Secure Hash Algorithm 384 SHA-256: Secure Hash Algorithm 256 SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.
	IKEv2	Time: 10 ... 16666666 minutes Limit: 10 ... 4096 MB	AES-256, 192, 128*: Advanced Encryption Standard (CBC, CTR, CCM 16 or GCM 16 mode) 3DES-168**: Triple DES (CBC mode)***	SHA-512: Secure Hash Algorithm 512 SHA-384: Secure Hash Algorithm 384 SHA-256: Secure Hash Algorithm 256 SHA-1***: Secure Hash Algorithm 1 MD5***: Message Digest Algorithm 5	ON: Diffie Hellman public key values are exchanged for recalculation of the keys. OFF***: The values already exchanged in phase 1 are used for recalculation of the keys.

* 256 bit, 192 bit or 128 bit key length

** 168 bit key length

*** The setting is classified as less secure. It is recommended when possible that you use a more secure setting.

Configuring fallback VPN profiles

In addition to the properties of a VPN group, fallback VPN profiles can be assigned to a VPN group. VPN profiles contain predefined properties for the authentication method, IKE phase 1 and 2. Fallback VPN profiles can be used as alternative configurations for VPN connection establishment. These come into play when VPN properties of the initiator do not match the VPN properties of the responder. In such cases, fallback VPN profiles ensure the VPN connection establishment. The configured fallback VPN profiles just as the properties of the VPN group are transferred to all VPN group members. The priority for the use of the fallback VPN profiles is specified by their order in the configuration. The fallback VPN profiles and the properties of the VPN group must not differ in the authentication method.

The following fallback VPN profiles are available:

Table 1-39 VPN profile 1

Parameter	Setting
Authentication method	Certificate
IKE mode	Main
Phase 1 DH group	Group 14
Phase 1 encryption	AES-256
Phase 1 SA lifetime	SA lifetime proposed by the initiator: 480 minutes Range permitted by the responder for the SA lifetime: 480 … 2880 minutes
Phase 1 authentication	SHA-1
Phase 2 SA lifetime type	Time
Phase 2 encryption	AES-128
Phase 2 SA lifetime	SA lifetime proposed by the initiator: 240 minutes Range permitted by the responder for the SA lifetime: 60 … 2880 minutes
Phase 2 authentication	SHA-1
Perfect Forward Secrecy	OFF

Table 1-40 VPN profile 2

Parameter	Setting
Authentication method	Certificate
IKE mode	Main
Phase 1 DH group	Group 2
Phase 1 encryption	AES-256
Phase 1 SA lifetime	SA lifetime proposed by the initiator: 480 minutes Range permitted by the responder for the SA lifetime: 480 … 2880 minutes
Phase 1 authentication	SHA-1
Phase 2 SA lifetime type	Time
Phase 2 encryption	3DES-168
Phase 2 SA lifetime	SA lifetime proposed by the initiator: 2880 minutes Range permitted by the responder for the SA lifetime: 60 … 2880 minutes
Phase 2 authentication	SHA-1
Perfect Forward Secrecy	OFF

Table 1-41 VPN profile 3

Parameter	Setting
Authentication method	Certificate
IKE mode	Main

Parameter	Setting
Phase 1 DH group	Group 2
Phase 1 encryption	3DES-168
Phase 1 SA lifetime	SA lifetime proposed by the initiator: 480 minutes Range permitted by the responder for the SA lifetime: 480 … 2880 minutes
Phase 1 authentication	SHA-1
Phase 2 SA lifetime type	Time
Phase 2 encryption	3DES-168
Phase 2 SA lifetime	SA lifetime proposed by the initiator: 2880 minutes Range permitted by the responder for the SA lifetime: 60 … 2880 minutes
Phase 2 authentication	SHA-1
Perfect Forward Secrecy	OFF

Table 1-42 VPN profile 4

Parameter	Setting
Authentication method	Certificate
IKE mode	Main
Phase 1 DH group	Group 2
Phase 1 encryption	DES
Phase 1 SA lifetime	SA lifetime proposed by the initiator: 480 minutes Range permitted by the responder for the SA lifetime: 480 … 2880 minutes
Phase 1 authentication	MD5
Phase 2 SA lifetime type	Time
Phase 2 encryption	3DES-168
Phase 2 SA lifetime	SA lifetime proposed by the initiator: 2880 minutes Range permitted by the responder for the SA lifetime: 60 … 2880 minutes
Phase 2 authentication	SHA-1
Perfect Forward Secrecy	OFF

Table 1-43 VPN profile 5

Parameter	Setting
Authentication method	Pre-shared key
IKE mode	Main
Phase 1 DH group	Group 2
Phase 1 encryption	3DES-168
Phase 1 SA lifetime	SA lifetime proposed by the initiator: 480 minutes Range permitted by the responder for the SA lifetime: 480 … 2880 minutes

Parameter	Setting
Phase 1 authentication	SHA-1
Phase 2 SA lifetime type	Time
Phase 2 encryption	3DES-168
Phase 2 SA lifetime	SA lifetime proposed by the initiator: 2880 minutes Range permitted by the responder for the SA lifetime: 60 … 2880 minutes
Phase 2 authentication	SHA-1
Perfect Forward Secrecy	OFF

After opening a STEP 7 project, that was last stored with an older version than V14 SP1, all existing VPN groups with the authentication method "Certificate" are assigned the predefined fallback VPN profile 2 - 5 as default.

Special features for the SOFTNET Security Client and SCALANCE M

Configured fallback VPN profiles are not included in the configuration files for the Road Warrior modules SOFTNET Security Client and SCALANCE M For the SOFTNET Security Client and SCALANCE M modules the following applies:

- Select the VPN profile to be used for VPN connections of SOFTNET Security Client / SCALANCE M modules to SCALANCE S / CPs in the responder settings, see section Configuring module- and connection-specific VPN settings (Page 453).
- For VPN connections from SCALANCE M modules to SCALANCE M modules, the configured VPN group settings are used.
- For VPN connections from SOFTNET Security Client to SCALANCE M modules the settings of VPN profile 3 are always used.

Modes of VPN groups

VPN modes

Depending on the mode of the security modules that were added to a VPN group, different modes of VPN groups are distinguished. The mode of a VPN group provides information about the security modules and their modes that can be added to the VPN group.

Rules for forming groups

Remember the following rules if you want to create VPN groups:

- For SCALANCE S612 / S613 / S623 / S627-2M / SCALANCE M / VPN device
  The first module assigned in a VPN group decides which other modules can be added to it.
  If the first SCALANCE S module to be added is in routing mode or if the first module is a SCALANCE M module or a VPN device, only additional SCALANCE S modules with routing activated or SCALANCE M modules cor VPN devices an be added because SCALANCE M modules and VPN devices always operate in routing mode. If the first SCALANCE S module added is in bridge mode, you can only add other SCALANCE S modules in bridge mode. If you want to change the mode of a VPN group, you have to remove all the modules contained in the group and add them again. A CP and an SSC can be added to a group with a SCALANCE S in bridge or routing mode.
- For CP / SSC
  If a CP / SSC is the first module in a VPN group, security modules can be added in any mode. The next module that decides the mode then also decides the mode of the VPN group. A CP / SSC can be assigned to several VPN groups at the same time and use different modes. The CP / SSC is then operated in mixed mode.
- It is not possible to add a SCALANCE M module to a VPN group that contains a module in bridge mode.

Refer to the following table to see which modules can grouped together in a VPN group:

Table 1-44 Security modules and VPN modes

Module	Can be included in a VPN group in...	
	Bridge mode	Routing mode
SCALANCE S612 / S613 / S623 / S627-2M in bridge mode	x	-*
SCALANCE S612 / S613 / S623 / S627-2M in routing mode	-	x
CP 343-1 Adv. / CP 443-1 Adv.	x	x
S7-1200/S7-1500 CPs	x	x
CP 1628	x	x
SOFTNET Security Client V4.0	x	x
SCALANCE M875 / VPN device	-	x

* SCALANCE S623/S627-2M modules in bridge mode can be inserted in a VPN group in routing mode if their DMZ interface is activated (not at the same time).

Including security module in configured VPN group

The configured VPN group properties are adopted for security modules added to in an existing VPN group.

Procedure after including a security module in a configured VPN group

Depending on whether or not you have changed the VPN group properties since the last download, you must make a distinction between the following:

- **Case a:** If you have not changed the VPN group properties and the module to be added actively establishes the connection to modules of the type SCALANCE S, CP 343-1 Adv. / CP 443-1 Adv. or CP 1628:

1. Add the new security module to the VPN group.
2. Download the configuration to the new module.

- **Case b:** If you have changed the VPN group properties or the module being added does not actively establish the connection to the already configured modules:

1. Add the new security module to the VPN group.
2. Download the configuration to all modules that belong to the VPN group.

In case a, it is not necessary to reconfigure and load the already commissioned security modules. Active communication is not influenced or interrupted.

Procedure after removing an active member from a VPN group

If you remove an active node from an existing VPN group, this can still establish a connection to the group members even if you have downloaded the project to all members of the VPN group again.

If you do not want the removed active node to be able to establish the connection any longer, renew the CA group certificate and download the project again to the members of the VPN group. The certificate can be renewed in the group properties of the VPN group or in the "CA" tab of Certificate Manager.

Configuring network nodes reachable through a tunnel

Overview

Configuring network nodes reachable through a tunnel

Each security module must know the network nodes reachable via a VPN tunnel to be able to recognize the authenticity of a frame.

The security module must know both its own nodes as well as the nodes of the security modules in the same VPN group. This information is used on a security module to decide which data packet will be transferred in which tunnel.

By adding a security module to a VPN group, the local internal nodes/subnets of the security module are automatically made known to the security module. To allow communication via the VPN tunnel with other subnets or nodes of another subnet (routed internal network; DMZ network when the VPN tunnel is on the external interface and vice versa), these subnets or nodes must be enabled for VPN tunnel communication in the configuration.

SCALANCE S modules allow network nodes to be learned automatically or configured statically. The mode of the security module also decides the options available for learning internal network nodes.

SCALANCE S in bridge mode

In bridge mode, you can configure the IP/MAC nodes and the subnets or alternatively can allow automatic learning of nodes by the SCALANCE S.

SCALANCE S in routing mode

In routing mode there is no automatic learning mode available. Instead, enter entire subnets here that need to be released for tunnel communication.

CP 343-1 Advanced / CP 443-1 Advanced and CP 1628

- CP 343-1 Advanced / CP 443-1 Advanced
  Decide whether or not tunnel communication to the CP (Gbit interface) and/or to the internal subnet (PROFINET subnet) is permitted for VPN connection partners in routing mode (SCALANCE S / M).
- CP 1628
  Enter the NDIS nodes you want to be reachable through the tunnel of VPN connection partners in routing mode (SCALANCE S / M).

CP 1543-1 as of V2.1 with activated IP routing between communications modules

Enter the subnets or IP addresses reachable via the backplane bus of the station and that need to be released for tunnel communication.

Automatic learning of internal network nodes

Module-specific function

SCALANCE S modules in bridge mode provide a learning mode with which the internal network nodes can be learned automatically during operation. For more detailed information, refer to the section:
Using the learning mode to learn internal nodes (Page 510)

Configuring IP network nodes manually for SCALANCE S

Module-specific function

How to configure IP network nodes for SCALANCE S modules manually is explained in:
Configuring IP network nodes manually (Page 511).

Configuring MAC network nodes manually for SCALANCE S

Module-specific function

How to configure MAC nodes for SCALANCE S modules manually is explained in:
Configuring MAC network nodes manually (Page 512).

Configuring internal subnets for SCALANCE S manually

Module-specific function

How to configure the internal subnets for SCALANCE S modules is explained in
Configuring internal subnets manually (Page 512).

Allow access to S7-300 / S7-400 CPs for VPN connection partners

Module-specific function

How to allow access to S7-300 / S7-400 CPs for VPN connection partners is explained in
Allow access to S7-300 / S7-400 CPs for VPN connection partners (Page 536) in the section "Security for S7-300 /S7-400 /PC CPs".

Configuring NDIS nodes for PC CPs that can be reached through the tunnel

Module-specific function

How you configure NDIS nodes for PC CPs that can be reached through the tunnel is explained in
Configuring NDIS nodes manually for PC CPs that can be reached through the tunnel (Page 536) in the section "Security for S7-300 / S7-400 / PC CPs".

Configuring subnets reachable via the backplane bus for CP 1543-1

Module-specific function

For information on how to configure subnets and network nodes reachable via the backplane bus of the station for CP 1543-1 as of V2.1, see section
Configuring subnets reachable via the backplane bus for CP 1543-1 (Page 547).

Configuring module- and connection-specific VPN settings

Requirement

The module is a member of a VPN group.

Module- and connection-specific settings

With module- and connection-specific settings, you can configure specific VPN settings. Module-specific settings are configured specifically for a security module while connection-specific settings are configured specifically for a security module in a certain VPN group.

You can configure the following **module-specific** properties in the local security settings with the "VPN" entry:

- Dead peer detection
- Permission to initiate connection establishment
- Public address (IP address / FQDN) for communication via Internet gateways
- Responder settings for Road Warrior connections
- Nodes that need to be enabled for tunnel communication

If you select a security module in the list of VPN groups in the "Network data" area, the following **connection-specific** VPN settings can be seen and can be configured:

- Permission to initiate connection establishment
- Partner modules to which tunnel connections exist
- Type of transferred packets
- Selection of the local interface of the selected security module that will serve as the tunnel endpoint
- Selection of the partner interface that will serve as the tunnel endpoint

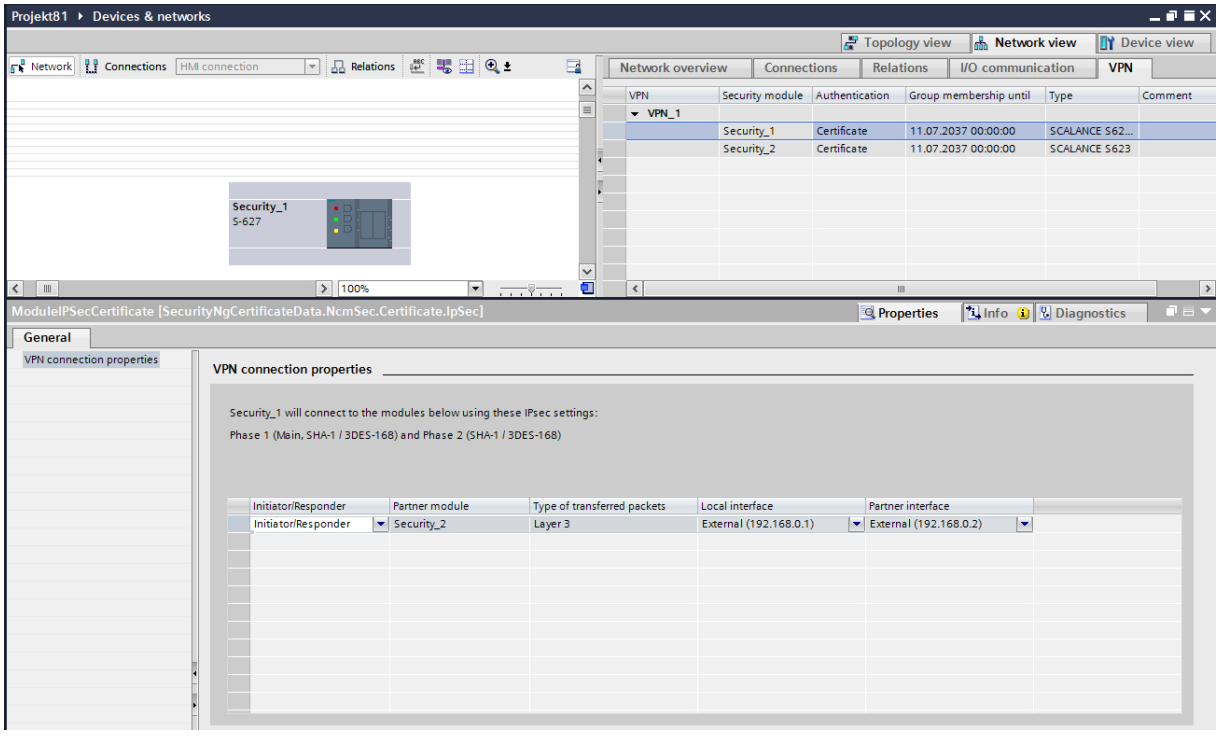

Dead peer detection (DPD)

As default, DPD is enabled.

When DPD is enabled, the modules exchange additional messages at selectable intervals if no communication occurs at these points in time. This means that it is possible to recognize whether the IPsec connection is still valid or possibly needs to be re-established. If there is no longer a connection, the security associations (SA) of phase 2 are terminated prematurely. If DPD is disabled, the SA is ended only after the SA lifetime has expired. For information on setting the SA lifetime, see section
Configuring VPN group properties (Page 439).

Permission to initiate connection establishment

You can restrict the permission for initiating the VPN connection establishment to certain modules in the VPN.

The decisive factor for the setting of the parameter described is the assignment of the address for the gateway of the module you are configuring. If a static IP address is assigned, the module can be found by the partner. If the IP address is assigned dynamically and therefore changes constantly, the partner cannot establish a connection as things stand.

Mode	Meaning
Start connection to partner (initiator/responder) (default)	If this option is selected, the module is "active", in other words, it attempts to establish a connection to the partner with a fixed IP address. The reception of requests for VPN connection establishment is also possible. This option is recommended when you obtain a dynamic IP address from the provider for the gateway of the security module you are configuring. The partner is addressed using its configured WAN IP address, its configured external module IP address or the configured DNS name.
Wait for partner (responder)	If this option is selected, the module is "passive", in other words, it waits for the partner to initiate the connection. This option is recommended when you have been assigned a static IP address by the provider for the gateway of the security module you are configuring. It means attempts to establish a connection can only be initiated by the partner. This partner can, for example, have a dynamic WAN IP address.

Note

Make sure that you do not set all the modules in a VPN group to "Wait for connection from remote VPN gateway" otherwise no connection is established.

WAN IP address / FQDN - addresses of the modules and gateways in a VPN over Internet

When operating a VPN with IPsec tunnel over the Internet, additional IP addresses are generally required for the Internet gateways such as DSL routers. The individual security or SCALANCE M modules must know the external IP addresses of the partner modules in the VPN.

Note

To use a WAN as an external public network, enter the IP address that you received from the provider as the external IP address, through which the security module is then reachable in the WAN (Internet). To allow the security module to send packets via the WAN, you need to enter your DSL router as "Standard router".

If you use a DSL router as Internet gateway, the following ports (at least) must be opened on it as described in the relevant documentation:

- Port 500 (ISAKMP)
- Port 4500 (NAT-T)

To allow this, in the module-specific VPN settings, you have the option of assigning an IP address as a "WAN IP address". When you download the module configuration, the group members are then informed of the WAN IP addresses of the partner modules. As an alternative to a WAN IP address, you can also enter an FQDN. Depending on the existing addresses, VPN endpoints are used as default according to the following priorities:

1. WAN address
2. FQDN of the primary dynamic DNS service
3. FQDN of the secondary dynamic DNS service
4. External IP address / DMZ IP address of the security module

Note: After removing an existing WAN address, the external IP address / DMZ IP address is always used.

In the interface selection of the connection-specific VPN settings, you yourself can decide which address the partner will be informed of. In these settings, you can also specify the interface to be used for communication by the nodes of a VPN group and the security module that is authorized to set up connections.

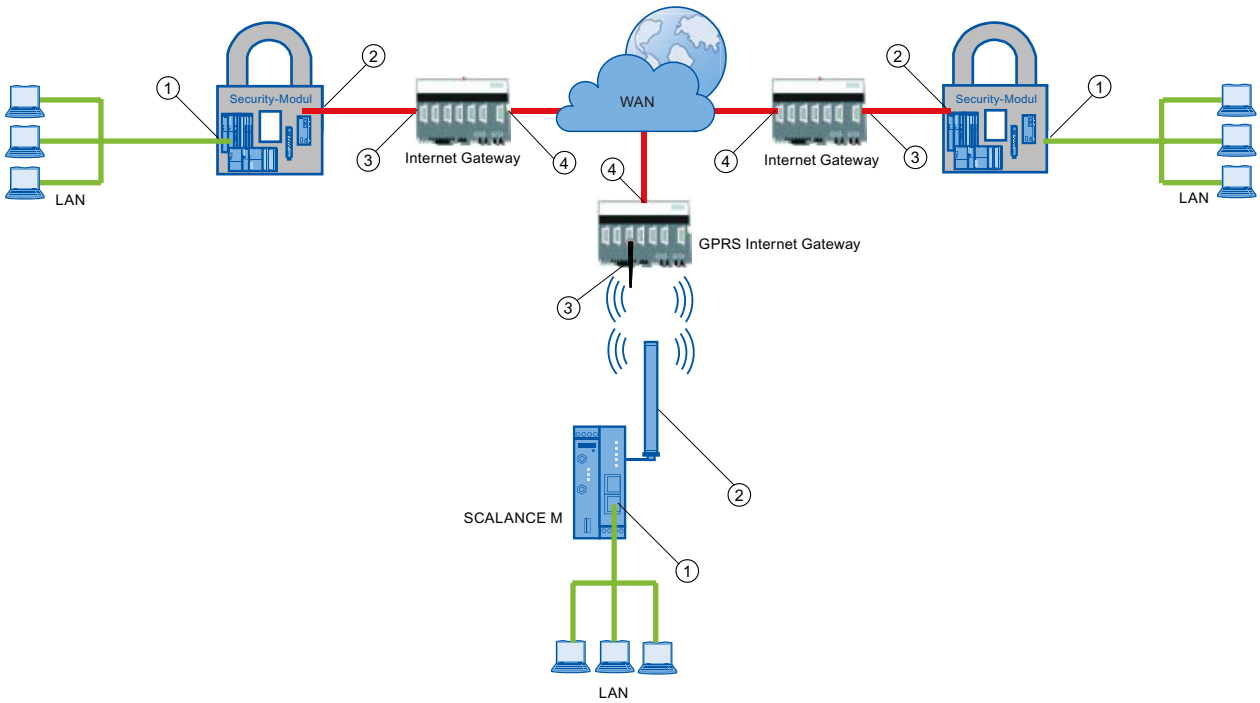

① Internal IP address - of a module
② External IP address - of a module
③ IP address of an Internet gateway (for example, GPRS gateway)
④ IP address (WAN IP address) of an Internet gateway (for example, DSL router)

Responder settings for Road Warrior connections

The VPN connection establishment by Unknown Peers (Road Warriors) is allowed by the responder as default with the secure VPN properties of VPN profile 1. You can allow or block the VPN connection establishment by Unknown Peers and the properties used by selecting a VPN group profile and by selecting VPN profiles.

Operator control element	Meaning
Allow VPN connection establishment by Road Warriors	The responder allows VPN connection establishment by Unknown Peers.
VPN profile 1 - 5 in the selection dialog for VPN profiles	The responder allows VPN connection establishment by unknown peers with the properties of the predefined VPN profiles 1 - 5. The properties of these VPN profiles correspond to the properties of the fallback VPN profiles that you can select for VPN groups, see section Configuring VPN group properties (Page 439).
VPN group profiles in the selection dialog for VPN profiles	The responder allows VPN connection establishment by Unknown Peers with the properties of the VPN group selected in the selection dialog for VPN profiles with a check mark. The VPN groups can be selected in which the responder is located. A maximum of one VPN group can be selected and used as a VPN group profile. With the "Add" button, new VPN groups can be created to which the responder is automatically assigned. These VPN groups can then be selected for the formation of a VPN group profile. If VPN connection establishment by Unknown Peers with the properties of the VPN group in which the responder and the Unknown Peers are located is to be allowed, this VPN group must be selected in the selection dialog.

After opening a STEP 7 project last saved with a version older than V14 SP1, the check box "Allow VPN connection establishment by Road Warriors" is activated s default. In addition to this, the responder is assigned the predefined VPN profiles 2 - 5.

For VPN connections from SOFTNET Security Client / SCALANCE M-800 modules to SCALANCE S / CPs the selected VPN profile with the highest priority is always used and read out. The authentication method of the selected VPN profile must match the authentication method of the corresponding VPN group. The priority of the VPN profiles is as follows:

1 Selected VPN group profile
2. VPN profile 1
3. VPN profile 2
4. VPN profile 3
5. VPN profile 4
6. VPN profile 5

For VPN connections from SCALANCE M875 modules to SCALANCE S / CPs, VPN profile 3 must be selected if the authentication method of the corresponding VPN group is "Certificate". If the authentication method of the corresponding VPN group is "Preshared key", VPN profile 5 must be selected.

Configuring internal network nodes

The configuration of internal network nodes is described in the following section: AUTOHOTSPOT

See also

How to create an IPsec tunnel with VPN groups (Page 434)

Establishing an OpenVPN tunnel to the SINEMA Remote Connect server

Module-specific function

This function is only available for the following modules:

- CP 1243-1 as of V3.1
- CP 1243-8 IRC as of V3.1
- CP 1243-7 LTE as of V3.1
- CP 1542SP-1 IRC as of V2
- CP 1543SP-1 as of V2

You will find detailed information in the section Automatic OpenVPN configuration and OpenNPN tunnel establishment (Page 548) of the chapter "Security for S7-1200-/S7-1500-CPs".

Configuring router and firewall redundancy

Module-specific function

This function is available only for SCALANCE S623/S627-2M as of V4, see section AUTOHOTSPOT.

Online functions - diagnostics and logging

Overview of diagnostics and logging in

For test and monitoring purposes, the security module has diagnostic and logging functions.

- Diagnostic functions
  These include various system and status functions that you can use for an existing network connection to the security module.
- Logging functions
  This involves the logging of system and security events and data packets.

Recording events with logging functions

You select the events to be logged in the log settings for the relevant security module.

You can configure the following variants for logging:

- Local logging
  In this variant, you log events in the local buffer of the security module. You can then access these logs, display them and archive them on the service station in the "Online & diagnostics" dialog. The evaluation of the buffer areas of the security module is only possible if there is a network connection to the selected security module.
- Network Syslog
  With Network Syslog, you use a Syslog server that exists in the network. This logs the events according to the configuration in the log settings for the relevant security module.

Archiving log data and reading in from a file

You can save the logged events for archiving in a log file and open this later even without a network connection to the security module. You will find the function for opening the log files in the global security functions.

Diagnostics in ghost mode (only for SCALANCE S S602 as of V3.1)

When you operate the security module in ghost mode, the external interface of the security module takes over the IP address of the internal node at runtime. Before you can run diagnostics of a security module in ghost mode, you need to establish the connection to the security module via the IP address that the security module obtained from the internal node at runtime.

To find out which IP address the security module currently has, you can search for accessible nodes in STEP 7 using the "Online" > "Accessible devices".

Protecting exported log files from unauthorized access

Log files exported from STEP 7 can contain security related information. You should therefore make sure that these files are protected from unauthorized access. This is particularly important when passing on the files.

Function overview online dialog

Functions of the "Online & diagnostics" dialog

In STEP 7, the security module provides the following functions in the "Online & diagnostics" dialog:

Entry in the online dialog	Function
Status	Display of status information about the selected security module, such as the current IP addresses of the interfaces and the current time and date.
Interface settings	Overview of the settings of the individual interfaces.
Dynamic DNS	Overview of the settings for dynamic DNS.
System log	Display of logged system events, starting and stopping logging (only if there is an online connection to SCALANCE S modules) and starting and stopping the reading out of log data from the local buffer of the security module.
Audit log	Display of logged security events, starting and stopping reading out of log data from the local buffer of the security module.
Packet filter log	Display of logged data packets, starting and stopping logging (only if there is an online connection to SCALANCE S modules) and starting and stopping the reading out of log data from the local buffer of the security module.
ARP table	Display of the ARP table of the security module.
Logged-on users	Shows the users logged in to the Internet page for user-specific IP rule sets.
Communications status	Display of status information about VPN tunnel connections and members of VPN groups to which the selected security module belongs.
SINEMA RC - automatic VPN configuration	Display of the status of the OpenVPN configuration and of OpenVPN connections to the SINEMA RC Server.
Internal nodes	Display of the learned or configured internal network nodes of the security module.
Dynamically updated firewall rules	Display of the IP addresses that were released dynamically over HTTP or HTTPS, or loaded by a user.
Firewall blacklist	Display of the IP addresses that were entered in the blacklist of the firewall.
Ghost mode	Dialog for the ghost mode of the SCALANCE S602 with information on the address parameters of the internal node (identical to the external IP address of the security module) and on IP address changes of the internal node.
Date and time of day	Date and time setting.
Firmware update	Updating the firmware.

Requirements for access

The following requirements must be met to use the online functions of a security module:

- There is a network connection to the selected module
- The project with which the module was configured is open
- a user with the required rights must be logged in to the project
- For CPs the diagnostics access must be opened in the firewall (CP 343-1 Adv. / 443-1 Adv. and CP 1628: TCP 443; S7-1200/S7-1500 CPs: TCP 8448)

Note

Requirement for online diagnostics in ghost mode (only for SCALANCE S S602 as of V3.1)

Online diagnostics is only available in ghost mode if the security module has learned the IP address of the internal node and has adopted this for its external interface. After this, the security module can be reached via the IP address of the external interface.

How to access this function

1. Right-click on the module to process.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. If there is no online connection established to the security module, click the "Connect online" button in the "Diagnostics" entry.

Online settings are not saved in the configuration

Settings that you make in online mode (for example settings for the logging memory) are not stored in the configuration on the security module. This is why the configuration settings are always applied at the restart of the module.

Functions of the online diagnostics

Status information of the security module - "Status" entry

Meaning

Display of the status of the security module selected in the project.

Table 1-45 Online & diagnostics: - "Status" entry

System and status functions	Meaning
Overview	
Hardware type	The type of the security module.

System and status functions	Meaning
External IP address	The external IP address of the security module. For S7-1200/S7-1500 CPs, CP 1628: The IP address of the Industrial Ethernet interface. For CP 343-1 Adv. / 443-1 Adv.: The IP address of the Gbit interface.
Internal IP address	The internal IP address of the security module. For CP 1628: The IP address of the NDIS interface. If there is more than one NDIS address, only one is displayed. For CP 343-1 Adv. / 443-1 Adv.: The IP address of the PROFINET interface.
DMZ IP address (SCALANCE S623 / S627-2M only)	The DMZ IP address of the security module.
Tunnel IP address (only SCALANCE S612/S623/ S627-2M as of V4)	The first alias internal IP address of the security module in the VPN tunnel.
Serial number	The serial number of the security module.
Article number	The article number of the security module that is used when ordering.
Firmware version	The firmware version of the security module.
Operating mode	Mode of the security module (bridge mode / routing mode)
External MAC address	The external MAC address of the security module. For S7-1200/S7-1500 CPs, CP 1628: The MAC address of the Industrial Ethernet interface. For CP 343-1 Adv. / 443-1 Adv.: The MAC address of the gigabit interface.
Internal MAC address	The internal MAC address of the security module. For CP 1628: The MAC address of the NDIS interface. For CP 343-1 Adv. / 443-1 Adv.: The MAC address of the PROFINET interface.
DMZ MAC address (SCALANCE S623 / S627-2M only)	The DMZ MAC address of the security module
Hardware release	The hardware product version of the security module.
C-PLUG	Shows whether or not a C-PLUG is inserted.
Alias IP addresses (only for SCALANCE S as of V4)	
IP address	Alias IP address that was registered on an interface of the security module by a NAT rule.
Corresponding Interface	Interface of the security module on which the alias IP address was registered.
Local time	

System and status functions	Meaning
Current time	Date and time that is displayed on the security module. Format for "German" user interface language: dd.mm.yyyy (date) hh:mm:ss (time) Format for "English" user interface language: mm/dd/yyyy (date) hh:mm:ss AM/PM (time) Format for "French", "Italian" and "Spanish" user interface language: dd/mm/yyyy (date) hh:mm:ss (time) Format for "Chinese" user interface language: yyyy/mm/dd (date) hh:mm:ss **Note (not for CPs)** You set the local time on the SCALANCE S module in "Functions" > "Date and time".
Operating time	Time since the last restart of the security module. Format with the user interface language "German", "English", "French", "Italian", "Spanish" and "Chinese": dddd.hh:mm:ss
Time-of-day source	The source from which the date and time are obtained.
Configuration	
Created	Date and time when the project was first created. Format with the user interface language "German", "English", "French", "Italian", "Spanish" and "Chinese": dd.mm.yyyy (date) hh:mm:ss (time)
Name	File name of the project last loaded.
Author	Name of the user who created the project. Is adopted from the project properties.
Compiled	Date and time when the project was last compiled. Format with the user interface language "German", "English", "French", "Italian", "Spanish" and "Chinese": dd.mm.yyyy (date) hh:mm:ss (time)
Storage location	Specifies the location (e.g. town) that was entered in the properties of an SCT project.
File system (not for CPs)	
RAM	Indicates how much RAM and flash is occupied in the file system.
Flash	

Overview of the individual interfaces - "Interface settings" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Overview of the individual interfaces - "Interface settings" entry (Page 518)

Overview of the Dyn. DNS settings - "Dynamic DNS" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Overview of the Dyn. DNS settings - "Dynamic DNS" entry (Page 519)

Display of the ARP table - "ARP table" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Display of the ARP table - "ARP table" entry (Page 521)

Users logged in to the Web page - "Logged in users" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Users logged in to the Web page - "Logged in users" entry (Page 521)

VPN connections of the security module - "Communication status" entry

Meaning

Display of the communication status of the following network components:

- Other security modules of the VPN group to which the selected security module belongs
- Internal network nodes of these security modules

Table 1-46 Online & diagnostics: "Communication Status" entry

System and status functions	Meaning
Known security devices or modules	Display of the nodes with which the selected security module is in a VPN group. This also shows whether the tunnel status is active or passive. To obtain additional information on one of the nodes, select this in the list. **Note:** Configured, inactive tunnels are indicated for CPs only.
Endpoints	Display of information on the internal network nodes of the security module that you selected in the "Known security devices or modules" table. For each internal network node, this shows whether it was learned or configured. It also shows the subnet in which the internal network node is located. With SCALANCE S modules, the subnet of network nodes is only displayed in bridge mode.
Tunnel properties	Display of properties of the VPN tunnel established to the security module that you selected in the "Known security devices or modules" table.

Automatic OpenVPN configuration via SINEMA RC Server

Module-specific function

This function is only available for the following modules:

- CP 1243-1 as of V3.1
- CP 1243-8 IRC as of V3.1
- CP 1243-7 LTE as of V3.1
- CP 1542SP-1 IRC as of V2
- CP 1543SP-1 as of V2

You will find detailed information in the section Diagnostics of the automatic OpenVPN configuration and OpenNPN connections (Page 549).

Found internal network nodes - "Internal nodes" entry

Meaning

Display of all learned and configured network nodes. This also displays whether or not the learning mode of the security module is enabled.

Updated firewall rules - "Dynamically updated firewall rules" entry

Module-specific function

This function is only available for CP 343-1 Adv. / 443-1 Adv., see section: AUTOHOTSPOTin the section "Security for S7-300 / S7-400 / PC CPs".

See also

Updated firewall rules - "Dynamically updated firewall rules" entry (Page 536)

Display of the firewall blacklist - "Firewall blacklist" entry

Module-specific function

This function is available only for SCALANCE S V4 modules or higher, see section Display of the firewall blacklist - "Firewall blacklist" entry (Page 521).

Setting the date and time - "Date and Time" entry

Module-specific function

This function is available only for SCALANCE S, see section Setting the date and time - "Date and time" entry (Page 522).

Diagnostics in ghost mode - "Ghost mode" entry

Module-specific function

This function is available only for SCALANCE S602 as of V3.1, see section Diagnostics in ghost mode - "Ghost mode" entry (Page 522).

Logging functions

Logging system events - "System log" entry

Meaning

Display of logged system events and starting and stopping reading system events from the local memory of the security module.

The system log automatically logs successive system events, for example the start of a process. The logging can be scaled based on event classes.

System and status functions	Meaning
Start/stop logging (not for CPs)	Starts/stops recording of system events. The method and the event classes that are logged are configured in the local security settings.
Start/stop reading	Starts/stops reading of system events from the local memory of the security module. If you select the "Save log file" check box, the recorded log data is also saved as file. Select the storage location and enter a file name. **Note** If you select the "Save log file" check box after starting reading, the data read out up to then can no longer be saved in a log file.
Clear display	Deletes the log data shown in the table.

You can open log files with saved system events in the global security functions in "Log files (offline view)".

Logging security events - "Audit log" entry

Meaning

Display of logged security events and starting and stopping reading of security events from the local memory of the security module.

The audit log automatically logs successive security-relevant events. This includes, for example, user actions such as enabling and disabling packet logging.

System and status functions	Meaning
Start/stop reading	Starts/stops reading of security events from the local memory of the security module. If you select the "Save log file" check box, the recorded log data is also saved as file. Select the storage location and enter a file name. **Note** If you select the "Save log file" check box after starting reading, the data read out up to then can no longer be saved in a log file.
Clear display	Deletes the log data shown in the table.

You can open log files with saved audit events in the global security functions in "Log files (offline view)".

Logging data packets - "Packet filter log" entry

Meaning

Display of logged data packets and starting and stopping reading of packet filter events.

The packet filter log records certain packets of the data traffic. Data packets are only logged if they match a configured packet filter rule (firewall) or to which the basic protection reacts (corrupt or invalid packets). This is only possible when logging is enabled for the packet filter rule.

For information about activation of the logging, refer to section AUTOHOTSPOT.

As well as reading the log data from the buffer and transferring it to the display, it can also be saved in a file for archiving.

System and status functions	Meaning
Start/stop logging (not for CPs)	Starts/stops logging of data packets. The method with which the data is logged is configured in the local security settings.
Start/stop reading	Starts/stops reading out of logged data packets from the local memory of the security module. If you select the "Save log file" check box, the recorded log data is also saved as file. Select the storage location and enter a file name. **Note** If you select the "Save log file" check box after starting reading, the data read out up to then can no longer be saved in a log file.
Clear display	Deletes the log data shown in the table.
Log category	Select the data packets for which the logging will be displayed. The selection depends on the settings you configured offline in the local security settings. Only the data packets for which logging was enabled are logged. If you select a category for which logging was not enabled, no data will be logged for this category.

You can open log files with saved packet filter events in the global security functions in "Log files (offline view)".

Download functions

Special features when downloading security configurations

Security configurations can influence the reachability of the security module for the configuration PC. This, for example, is the case if there is a tunnel connection configured for one security module to another security module in a configuration and this configuration is downloaded from the configuration PC to the security module. After the download by the configuration PC, the security module can no longer be reached and the reachability test performed as default by STEP 7 following the download of the configuration fails. The error message output by STEP 7 relates solely to the reachability test; the actual download of the configuration is ensured if the project data is consistent and the IP address relationship between the security module and the configuration PC is correct.

Special features when downloading configurations and firmware to SCALANCE S modules are described in the following section:
AUTOHOTSPOT

Uploading configurations to the engineering station

Uploading a configuration from a SCALANCE S module or from the CP 1628 to an engineering station is not possible.

Uploading a configuration from an S7 CP that supports security to an engineering station is also possible if security features were included in the configuration. Configured security functions are not, however, transferred to the engineering station. In the transferred configuration on the engineering station, the "Activate security features" check box is also deselected.

SOFTNET Security Client

Using the SOFTNET Security Client

Area of application - access over VPN

With the SOFTNET Security Client (SSC) PC software, secure remote access is possible from PGs/PCs to automation systems protected by a security module via public networks. You require SOFTNET Security Client V4.0 HF1 for S7-300/S7-400 CPs and for the PC CP 1628. These CPs are not approved for operation with SOFTNET Security Client ≤ V4.0. The SOFTNET Security Client has not been released for other CPs.

With the SOFTNET Security Client, a PG/PC is configured automatically so that it can establish a secure IPsec tunnel communication in the VPN (Virtual Private Network) with one or more security modules.

This IPsec tunnel communication makes it possible for PG/PC applications such as NCM diagnostics to securely access devices or networks that are located in an internal network protected by the security module.

How does the SOFTNET Security Client work?

The SOFTNET Security Client reads the configuration that was created by STEP 7 and, if necessary, determines from the file which certificates are to be imported.

The CA certificate and the private key are imported when applicable and stored on the local PG/PC.

Following this, security settings are made based on the data from the configuration so that applications can access IP addresses downstream from the security modules.

If the learning mode is enabled for the internal nodes or programmable controllers, a security policy is first set for secure access to security modules. Then, the SOFTNET Security Client addresses the security modules in order to obtain the IP addresses of the relevant internal nodes.

The SOFTNET Security Client enters these IP addresses in special filter lists belonging to this security policy. Applications can then communicate with the programmable controllers via VPN.

Note

Additional security measures when using the SOFTNET Security Client

The SOFTNET Security Client provides a solution for secure communication with automation cells via VPN. For self-protection of the PC/PG and the corresponding automation cell, it is advisable to use additional measures such as a virus scanner and the Windows firewall.

In Windows 7, the firewall of the operating system must be enabled so that VPN tunnel establishment works.

Creating a configuration file in STEP 7

Configuring a SOFTNET Security Client module in the project

The SOFTNET security client is created as a module in the project. In contrast to the other security modules, you do not need to configure any further properties.

Assign the SSC module to the VPN group or groups at which an IPsec tunnel to the PG/PC is to be set up.

Note

Refer to the information on parameters in section
Including security module in configured VPN group (Page 450).

Note

If you create several SOFTNET Security Clients within a group, no tunnels are set up between these clients but only from the relevant client to the security modules.

Configuration files for the SOFTNET Security Client

The interface between STEP 7 as configuration tool and the SOFTNET Security Client is controlled by configuration files.

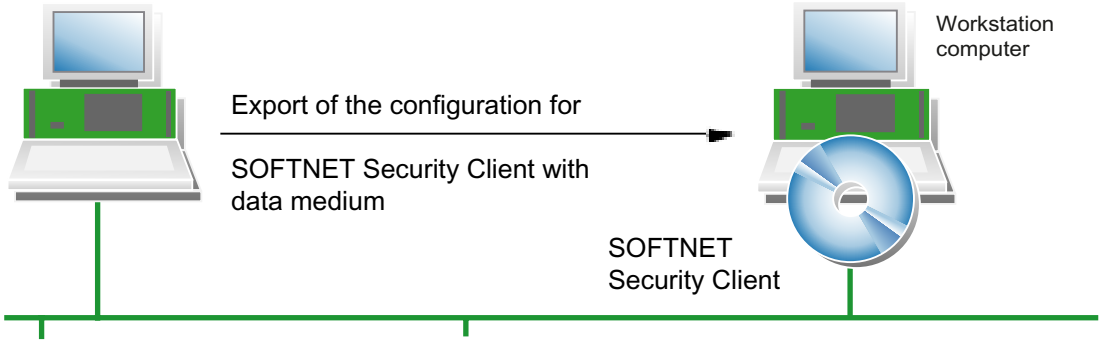

The configuration is stored in the file types "*.dat", "*.p12" and "*.cer".

Procedure

To generate the configuration files, perform the following steps in STEP 7:

1. In the "Devices & networks" view, select the "Topology view" or the "Network view" tab.
2. Insert a "Security device" into the selected tab from the hardware catalog via the selection "Network components > Industrial Security".
3. Select the SOFTNET Security Client V5 in the local security settings under "VPN device > VPN mode".
4. Assign the SOFTNET Security Client to the VPN groups in which the PG/PC will communicate over IPsec tunnels.
5. Make sure that the "Generate SSC files" check box is selected under the "Configuration of the SOFTNET Security Client" entry in the local security settings of the SOFTNET Security Client.
6. Select the storage location for the configuration files.
7. Compile the configuration of the SOFTNET Security client to export the configuration file.
8. If you selected certificate as the authentication method, specify a password for the certificate of the VPN configuration. If you do not assign a password, the project name (not the project password) is used as the password.
   Result: Export of the configuration files is completed.
9. Adopt the files of the type *.dat, *.p12, *.cer on the PG/PC on which you want to operate the SOFTNET Security Client.

Note

Protecting exported configuration files from unauthorized access

Configuration files for SOFTNET Security Client exported from STEP 7 can contain security related information. You should therefore make sure that these files are protected from unauthorized access. This is particularly important when passing on the files.

SCALANCE S-600 V3

Replacing a security module

Requirement

To be able to replace security modules, their module descriptions must be up to date. To update the module description of security modules, follow the steps below:

1. Select the security module to be edited.
2. In the local security settings, click on the entry "General" > "Catalog information".
3. Click the "Update module description" button.

How to access this function

1. Select the security module to be edited in the topology or network view.
2. Right-click on the security module and select the shortcut menu command "Change device...".
   Based on the following table, you can see which security modules you can replace without data loss and which could involve a possible data loss.

Initial module	Possible module replacement									
	S602 V3	S602 V3.1	S602 V4	S612 V3	S612 V4	S623 V3	S623 V4	S623 V4.0.1	S627-2 M V4	S627-2 M V4.0.1
S602 V3	-	x	x	x	x	x	x	x	x	x
S602 V3.1	!	-	x	!	!	!	!	!	!	!
S602 V4	!	!	-	!	!	!	!	!	!	!
S612 V3	!	!	!	-	x	x	x	x	x	x
S612 V4	!	!	!	!	-	!	x	x	x	x
S623 V3	!	!	!	!	!	-	x	x	x	x
S623 V4	!	!	!	!	!	!	-	x	x	x
S623 V4.0.1	!	!	!	!	!	!	!	-	!	x
S627-2M V4	!	!	!	!	!	!	!	!	-	x
S627-2M V4.0.1	!	!	!	!	!	!	!	!	!	-

x Without losses

! Possibly with losses

- The module type and the firmware version are not changed.

Configuring interfaces for SCALANCE S modules

Overview

Configuring the mode

With the mode you specify how interface routing is handled (external/internal). The DMZ interface of the security module (SCALANCE S623/S627-2M only) is always connected in routing mode. For more detailed information, refer to the section
Configuring IP address parameters (Page 475)

Configuring the interfaces

If the external interface, the DMZ interface (SCALANCE S623/S627-2M only) or the tunnel interface (only SCALANCE S612/S623/S627-2M as of V4 in VPN group(s)) of a security module needs to be configured, it must be activated using the "Activate interface" check box. Set the IP address information for each interface and settings for the individual ports. The following options are available with which you can assign an IP address for the external interface and for the DMZ interface in the "General" entry (SCALANCE S623/S627-2M only):

- Static IP address with subnet mask. For more detailed information, refer to the section
Configuring IP address parameters (Page 475)
- Address assignment using PPPoE. For more detailed information, refer to the section
Configuring an Internet connection (Page 477)
The internal interface and the tunnel interface can only be configured using a static IP address.

If alias IP addresses were registered on the interfaces of the security module due to configuring NAT rules, these are displayed in the "Alias IP addresses" entry.

Note

External interface and DMZ interface as Internet access

The simultaneous operation of PPPoE on the external interface and on the DMZ interface (dual ISP) is not possible.

Point to Point Protocol over Ethernet (PPPoE)

To allow Internet/WAN access directly via a DSL modem, the IP address on the external interface or on the DMZ interface is assigned using PPPoE. PPPoE is a dial-in protocol for obtaining IP addresses from an Internet service provider (ISP). SCALANCE S is operated here in routing mode.

To use this IP address assignment mode, specify the ISP in the "PPPoE" entry. The IP address, the subnet mask, the standard router and the DNS server of the interface are specified by the ISP.

Note

A configured standard router is not taken into account when using PPPoE. This is assigned dynamically to the module by the ISP.

Note

No network components between SCALANCE S and DSL modem

If the interface of a SCALANCE S module is operated using PPPoE, there must be no other network components between this interface and the connected DSL modem, otherwise the dial-in data of the Internet Service Provider may be transferred unencrypted over this link. When using the "CHAP" authentication protocol, the data is transferred encrypted.

Configuring media modules

In addition to the functions of the SCALANCE S623, the S627-2M has two media module slots in which an electrical or optical media module with two ports can be inserted. This expands both the external and internal interface by up to two ports. In routing mode, the additional ports of the security module can be used to link the external and internal interface to ring topologies.

To integrate media modules in the SCALANCE S627-2M, select the security module and change to the device view. Then select the required media modules from the hardware catalog.

For ports with the port type "Copper", you can set the transmission speed and the duplex method manually using the port mode. For ports with the port type "Optical", the port mode is fixed by the media module used or by the SFP transceiver used and cannot be adapted.

You will find information on connecting the media module ports to MRP rings in the following section:
AUTOHOTSPOT

Setting the operating mode

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Mode".

Operating mode - possible selections

You can change the operating mode in this dialog if the security module is not included in a VPN group. If the security module is in a VPN group, the mode cannot be changed.

The selection applies to interface routing between the external and internal interface. The DMZ interface (SCALANCE S623 and S627-2M only) is always connected in routing mode.

Bridge mode	For operation in flat networks. External and internal interface are in the same IP subnet. For S623 / S627-2M: External and internal interface are in the same IP subnet, the DMZ interface is in a different IP subnet or is deactivated.
Routing mode	All interfaces are in different IP subnets. If you have activated the routing mode, you must configure an internal IP address and subnet mask for the internal interface of the security module. **Note** If you have enabled the routing mode for the SCALANCE S module, no MAC firewall rules can be defined.
Ghost mode (only for SCALANCE S S602 as of V3.1)	In operation, the security module adopts the IP address of the node connected to the internal interface of the security module for the external interface. The IP address data specified for the external interface is only used for downloading the configuration prior to operation in ghost mode.

Configuring IP address parameters

Meaning

Specify network parameters such as the IP address and subnet mask for the interface(s) of the security module.

How to access this function

1. Select the module to be edited.
2. Select "External interface [P1] red" or "DMZ interface [P3] yellow" in the local security settings. The virtual tunnel interface can only be configured when the security module (only SCALANCE S612/S623/S627-2M as of V4) is in at least one VPN group, refer to the section "Meaning of the tunnel IP address".

   Note

   Configuration of the internal interface in routing mode

   If you have selected the "Routing" mode for the security module, you must also configure an internal IP address and subnet mask for the internal interface of the security module. You can access this function in the local security settings under "Internal interface [P2] green" > Ethernet addresses".

3. If applicable, enable the interface using the "Activate interface" check box.
4. Select the "Ethernet addresses" entry.
5. Complete the settings specified in the following table.

Parameter	Meaning
IP address	IP address for the external interface. The IP address consists of four decimal numbers from 0 to 255, with each number being separated by a period, for example, 141.80.0.16.
Subnet mask	The subnet mask consists of four decimal numbers that are separated by period, for example, 255.255.0.0
Use router (not possible for the tunnel interface)	Select this check box if you want to use a standard router and enter its IP address in the "Router address" input box.

Note
Networking the physical interfaces

Network the physical interfaces of the security module with suitable subnets to avoid IP address conflicts.

Meaning of the tunnel IP address

If you use the function "NAT/NAPT in the VPN tunnel" for a SCALANCE S612/S623/S627-2M module as of V4, you need to assign a tunnel IP address for the security module. This ensures the reachability of the security module via the VPN tunnel and provides a configuration and diagnostics option. The configured tunnel IP address can be expanded with alias tunnel IP addresses using suitable NAT / NAPT rules. The subnet mask is fixed at 32 bits for the tunnel IP address and cannot be changed by the configuration. The tunnel IP address can only be configured if the security module is in at least one VPN group.

You will find further information on address translation with NAT/NAPT in VPN tunnels in the following section:
Address translation with NAT/NAPT in VPN tunnels (Page 503)

Special features with a standard router

- If the IP assignment configured is via "PPPoE", a configured standard router is ignored because the standard route is always automatically via the PPPoE interface.
- If the address assignment configured is with "Static address" and if the security module is connected to the Internet via a DSL (NAPT) router, the DSL router must be entered as the standard router.
- For security modules in ghost mode (SCALANCE S602 as of V3.1 only), no standard routers can be configured since these are identified during runtime. Specific routes cannot be configured for security modules in ghost mode.

Configuring port mode

Meaning

The port mode specifies the transmission speed and the duplex method. The same parameters should be set for the ports involved in communication.

Configurable port modes

For SCALANCE S V3 and higher, the following port modes can be configured for the permanently installed ports:

Port mode	Meaning
Autonegotiation	The transmission speed and the duplex setting are selected automatically. **Note** A transmission speed of 1000 Mbps and the autocrossing function are supported only if autonegotiation is selected.
10 Mbps, half and full duplex	Transmission speed of 10 Mbps
100 Mbps, half and full duplex	Transmission speed of 100 Mbps

Deactivating a port is possible only for external ports and for the DMZ port of SCALANCE S623/S627-2M. The port modes for media module ports are configured in the device view and are based on the range of functions of the media module in question.

Configuring an Internet connection

Requirement

The "PPPoE" entry is only displayed in the local security settings if the IP assignment method "PPPoE" is configured for one of the interfaces.

How to access this function

1. Select the security module to be edited
2. In the local security settings, select the entry "PPPoE".

Meaning

In this entry, you make settings related to the Internet Service Provider (ISP) if a connection using PPPoE is set for one of the interfaces of the security module.

Table 1-47 Settings for the ISP account

Function	Description
Authentication protocol	Select none or one of the following authentication protocols: • PAP (Password Authentication Protocol) • CHAP (Challenge Handshake Authentication Protocol) **Note** Both communications partners have to use the same authentication method otherwise no connection can be established.
User name	Enter the name for logging in with the ISP account.

Function	Description
Password	Enter the password for logging in with the ISP account.
Repeat password	Enter the password for logging in with the ISP account again.

Table 1-48 Rules for user names and passwords

Permitted characters	The following characters from the ANSI X 3.4-1986 character set are permitted: 0123456789 A...Z a...z !#$%&()"*'+`,-./:;<=>?@ [\]_{|}~^
Length of the user name	1 to 120 characters
Length of the password	1 to 31 characters

Table 1-49 Settings for the connection

Function	Description
Permanent connection	Permanent Internet connection. After the connection has been terminated by the provider, the connection is automatically restored even if there are currently no packets to be sent.
On-demand connection	The Internet connection is established automatically if packets need to be sent to the Internet. In this setting, delays in the sending of packets are possible.
Maximum idle time (only with the setting "on-demand connection")	If no packets are sent during a certain time, the Internet connection is automatically terminated. In the "Maximum idle time" box, enter the time in seconds after which the connection will be terminated. • Default setting: 300 • Minimum value: 10 • Maximum value: 3600
Forced disconnection (only with the "Permanent connection" setting)	Select the check box to be able to adapt the time of the forced disconnection by the security module.
Forced disconnection time (only with the "Permanent connection" setting)	The provider terminates the Internet connection automatically after a certain period. If you enter a time of day in this box, the security module terminates the Internet connection itself at this time. This allows disconnection of the Internet connection by the provider to be delayed under certain circumstances. A self-initiated forced disconnection is only possible with an existing permanent connection. • Default setting: 00.00 • Permitted entries: 00:00 ... 23:59

Configuring DNS

Module-specific function

DNS servers for resolving FQDNs specified in the entry "Dynamic DNS" can be used by SCALANCE S modules as of V3. DNS servers for all other functions can be used by SCALANCE S modules as of V4.

DNS

To address network notes using FQDNs the security module must know the IP address of a DNS server that converts the DNS queries to the corresponding IP addresses of the network nodes.

Setting up DNS - Follow the steps below:

1. Select the security module to be edited.
2. In the local security settings, select the entry "DNS".

Option	Meaning
Obtain DNS server address automatically	The address of the DNS server can be obtained automatically using PPPoE if the security module is connected to the Internet via a DSL modem. Can only be set for the external interface and the DMZ interface. If PPPoE is set for one of these interfaces and no DNS server is configured, the option "Obtain DNS server address automatically" is set by STEP 7.
Use the following DNS server address:	Enter the address of the preferred and of the alternative DNS server manually.

Configuring dynamic DNS

Dynamic DNS

With dynamic DNS, you can access a constantly changing IP address with a permanently defined name (FQDN). This is necessary, for example, if you want to access a server that can be reached via a public IP address that changes.

The security module signals the current WAN IP address via which the security module can be reached to a provider for dynamic DNS (for example DynDNS.org, no-ip.com). The provider ensures that DNS queries sent to the FQDN of the security module are replied to with the current WAN IP address of the security module.

Dynamic DNS is permitted on the following interfaces:

- External interface
- DMZ interface

Requirement

An account has been created with a provides of dynamic DNS and an FQDN has been registered.

Setting up dynamic DNS - Follow the steps below:

1. Select the security module to be edited.
2. Select the "Dynamic DNS" entry in the local security settings.

3. Activate the "Activate service" check box in the "Primary dynamic DNS service" area and make the following settings:

Setting	Meaning
Provider	Choose the provider with which you have set up an account for dynamic DNS. With the predefined providers (DynDNS.org and No-IP.com), the provider update URL and the check IP service URL are already completed. If you select the option "User defined" you can specify the two URLs manually.
Service type	Select whether the HTTP or HTTPS protocol will be used for the provider update URL of the predefined provider.
FQDN	Enter the host name (e.g. mysecuritymodule) and the domain name (e.g. dyndns.org) that is registered with the provider separated by a period. The FQDN can function as a VPN endpoint and differ from the FQDN in the "VPN" entry. You can configure which VPN endpoint the VPN partner is informed of in the connection-specific VPN settings. The syntax of an FQDN must comply with DNS. Maximum number of characters: 99
Ignore errors when checking the server certificate	To protect the authentication data the certificate of the update server is checked. If the certificate check fails, the HTTPS connection is terminated as default and the account data is not transferred. If you activate the check box, the HTTPS connection is not terminated if the check fails (e.g. with an expired certificate) and the account data is transferred. You are advised not to ignore the check and not to select the check box. The check box can only be activated if the service type "DDNS-HTTPS" was selected.
User	Enter the user name that you specified when you created the account. The syntax of a user name must comply with DNS. Maximum number of characters: 24
Password	Enter the password that you specified when you created the account. The syntax of the password must comply with DNS. Maximum number of characters: 24
Monitor IP address change on DSL router	If the security module is connected to the Internet via a DSL router, enabling this function activates the function of the Check IP service. The security module periodically sends queries to determine the current IP address of the DSL router and to detect an IP address change on the DSL router. The IP address specified in this way is sent to the provider with each change ID.

Setting	Meaning
Period	Specify the interval at which the Check IP service is called. • Default setting: 20 minutes • Minimum value: 10 minutes • Maximum value: 1440 minutes
Provider update URL	Enter the URL with which the security module reports the public IP address to the provider. The placeholder texts <FQDN> and <CurrentWanIP> need to be placed at the correct positions in the URL. The syntax of the URL must comply with DNS. Maximum number of characters: 127
Check IP service URL	Enter the URL that the security module will use to determine its public IP address. The syntax of the URL must comply with DNS. Maximum number of characters: 127

4. In case the primary provider fails, create a second provider in the "Secondary dynamic DNS service" area (optional setting).

Configuring LLDP

Module-specific function

This function is available only for SCALANCE S V4 modules or higher.

Requirement

The security module is in routing mode.

Meaning

LLDP (Link Layer Discovery Protocol) is a protocol used to discover network topologies. A device capable of LLDP can send information about itself to neighboring devices at regular intervals and at the same time receive information from neighboring devices. The received information is stored on every device with LLDP capability in an LLDP MIB file. Network management systems can access these LLDP MIB files using SNMP and therefore recreate the existing network topology.

Configurable parameters

The degree of activity of the security module in terms of LLDP can be configured under the "LLDP mode" entry of the relevant interface.

Parameter	Description
Name	Name of the port for which the setting is configured.
LLDP mode	Configured LLDP mode: • RxTx: LLDP frames can be sent and received • Off: No LLDP frames can be sent or received

Media redundancy in ring topologies

Media redundancy with MRP

Module-specific function

This function is available only for SCALANCE S627-2M modules.

Meaning

The term "media redundancy" groups together various methods for increasing availability in Industrial Ethernet networks in which devices can be reached over different paths. This might be achieved by meshing networks, arranging parallel transmission paths or by closing a linear bus topology to form a ring.

Media redundancy method MRP

Media redundancy within a ring topology is available with SIMATIC NET products among other things with the MRP method (Media Redundancy Protocol).

With this method, one of the nodes is configured as the redundancy manager. The other nodes are redundancy clients. SCALANCE S627-2M modules can only adopt the role of an MRP client. Using test frames, the redundancy manager checks the ring to make sure it is not interrupted. The redundancy clients forward the test frames. If the test frames of the redundancy manager no longer arrive at the other ring port of the redundancy manager due to an interruption, the redundancy manager switches through its two ring ports and informs the redundancy clients of the change immediately.

The time the SCALANCE X switches need to switch through their ring ports as redundancy manager is 200 ms.

Note on the use of MRP

- MRP is supported in ring topologies with up to 100 devices. Exceeding this number of devices can lead to a loss of data traffic.
- It is recommended that you set the ring ports involved to full duplex and 100 Mbps. Otherwise there may be a loss of data traffic.

Possible uses of MRP on media module ports

MRP is supported only on the media module ports of the SCALANCE S627-2M. The following table shows the possible uses of MRP on the media module ports of a SCALANCE S627-2M:

Ring ports	Media module 1		Media module 2	
	P4	P5	P6	P7
MRP Client	-	-	-	-
	Ring 1	Ring 1	-	-
	-	-	Ring 2	Ring 2
	Ring 1	Ring 1	Ring 2	Ring 2

With two lower-layer rings per SCALANCE S module, layer 3 communication is possible between the rings.

Configuring MRP for the security module

Requirement

- The security module is in routing mode.
- Media modules are configured for the interfaces to be connected to MRP rings.
- The interfaces of the security module to be connected to rings are networked with the relevant ring managers.

How to access this function

1. Select the security module to be edited.
2. In the settings of the required interface, select the "Media Redundancy" entry.

Configurable parameters

Parameter	Meaning	Possible selections
MRP domain (only if the "MRP client" media redundancy role is selected)	The members of an MRP ring are specified with the help of MRP domains. The same MRP domain must be selected for the interfaces of all modules to be connected to the same MRP ring.	Display of the MRP domain used for the interface.
Media redundancy role	Selection of the media redundancy protocol or disabling of media redundancy for the interface.	• Not a node in the ring • MRP Client
Ring port 1 (only when the media redundancy role "MRP client" is selected)	Name of the first ring port of the selected interface if the media redundancy role "MRP client" was selected for it.	-

Parameter	Meaning	Possible selections
Ring port 2 (only when the media redundancy role "MRP client" is selected)	Name of the second ring port of the selected interface if the media redundancy role "MRP client" was selected for it.	-
Domain settings	Using the domain settings, you can add new MRP domains, edit the names of existing MRP domains and delete existing MRP domains.	-
Alternative media redundancy protocol	Select this check box to enable the interface of the security module for other media redundancy protocols.	• Enable interface for other media redundancy protocols • Disable interface for other media redundancy protocols (default setting)
Passive listening	Enable this check box if the selected interface will be connected to third-party networks in which STP/RSTP (Spanning Tree Protocol/ Rapid Spanning Tree Protocol) is used.	• Activate passive listening (default) • Deactivate passive listening

Special features of the ghost mode

Module-specific function

This function is available only for SCALANCE S602 as of V3.1.

Meaning

In ghost mode, the security module has no IP address of its own, neither on the internal nor on the external interface. Instead, the security module obtains the IP address for its external interface during runtime from a node connected to the internal interface of the security module whose IP address parameters may be unknown at the time of configuration. It is possible to change an IP address of the internal node and a corresponding IP address at the external interface. Since the internal node is identified based on its MAC address, IP address changes are made only for the learned MAC address. No IP address is configured or obtained at the internal interface of the security module.

As regards the MAC addresses, the security module replaces the MAC address of the internal node with the MAC address of the security module in all outgoing packets on the external interface (responses from the internal node).

Activating ghost mode

1. Select the module to be edited.
2. In the local security settings, select the entry "Mode".
3. Select the "Ghost mode" option.

Configurable module properties

In ghost mode, the following module properties can be configured in the local security settings:

- External interface [P1] red
- Firewall
- Time-of-day synchronization
- Log settings
- SNMP

Since no DNS servers can be configured in ghost mode, no FQDN resolution is possible.

Requirement for identifying an internal node

The security module can only obtain the IP address of the internal node if the internal node initiates data communication with a communications partner of the external network.
In addition to this, the security module does not provide any server services while obtaining the IP address. The security module can only reply to queries from external after data packets have been sent to the security module by the internal node.

Port assignment for incoming and outgoing data connections

Since the external interface of the security module and the internal node have the same IP address, the network components must be addressed explicitly via the TCP/UDP ports. For this reason, the ports are either assigned to the security module or the internal node. The assignments of the ports to the relevant devices are shown in the following tables for incoming and outgoing data connections:

Table 1-50 Port assignment for incoming connections (from external to security module)

Service	Port	Protocol	Comment
Web services, configuration and diagnostics access	443	TCP	The HTTPS port is permanently activated for configuration and diagnostics access using STEP 7 and cannot be changed.
SNMP	161	TCP UDP	Once SNMP is activated in STEP 7, incoming SNMP queries are transmitted via UDP port 161. Transfer via TCP port 161 is also possible, for example, to be able to reach the internal node. **Note** After activating SNMP, the SNMP port is permanently assigned to the security module. If SNMP is not activated, the internal node can be accessed using SNMP with the aid of a firewall rule.

Table 1-51 Port assignment for outgoing connections (from security module to external)

Service	Port	Protocol	Comment
Syslog	514	UDP	If the Syslog service is activated in STEP 7, Syslog messages are transferred via UDP port 514 by the security module. This port assignment cannot be changed.
NTP	123	UDP	If NTP servers are used for time-of-day synchronization, NTP queries are transferred via UDP port 123. This port assignment cannot be changed.

Recognizable IP addresses and subnet masks

The security module only recognizes internal nodes with IP addresses in the network class ranges A, B or C. The subnet mask is identified by the security module based on the network class (refer to the table "Network classes and corresponding subnet masks"). To allow the subnet mask to be determined correctly, a standard router must be entered for the internal node.

Nodes with IP addresses in the network classes D and E are rejected by the security module.

Network class	IP addresses		Subnet mask
	Low limit	High limit	
A	0.0.0.0	127.255.255.255	255.0.0.0
B	128.0.0.0	191.255.255.255	255.255.0.0
C	192.0.0.0	223.255.255.255	255.255.255.0
D	224.0.0.0	239.255.255.255	Rejected by the security module
E	240.0.0.0	255.255.255.255	Rejected by the security module

Configuration limits

A maximum of one internal node is recognized by the security module. If several internal nodes exist, the security module reacts as follows:

- The first device the security module recognizes in the internal network obtains access to the external network segment if the firewall is suitably configured.
- The data traffic of any additional nodes in the internal network area is blocked in the outgoing direction at level 2 (MAC layer) based on the sender address.

Loading configurations and diagnostics after commissioning

After obtaining an IP address from the internal node, the security module has an IP address on the external interface that can differ from the IP address with which the security module was initially configured. To load a configuration or to run diagnostics, in STEP 7 you need to specify the IP address for the connection to the external interface that the security module obtained from the internal node during runtime. This is possible in the local security settings or directly in the "Advanced download" or "Connect online" dialogs. For more detailed information on establishing online connections, refer to the section: Downloading a configuration (Page 523)

Routing information for hierarchical networks on the external interface

If there are hierarchical networks with subnet transitions on the external interface of the security module, the security module needs to obtain the relevant routing information from the internal node. To achieve this, the internal node must respond to ICMP queries sent to it. Responding to ICMP broadcasts is not necessary.

Authentication using a RADIUS server

Overview

Module-specific function

This function is available only for SCALANCE S V4 modules or higher.

Meaning

RADIUS (Remote Authentication Dial-In User Service) is a protocol for authenticating users by servers on which user data can be stored centrally. The use of RADIUS servers can increase the protection of user names, assigned roles and passwords.

Scenario for the use of RADIUS servers

Authentication by RADIUS servers can be performed when activating user-specific IP rule sets.

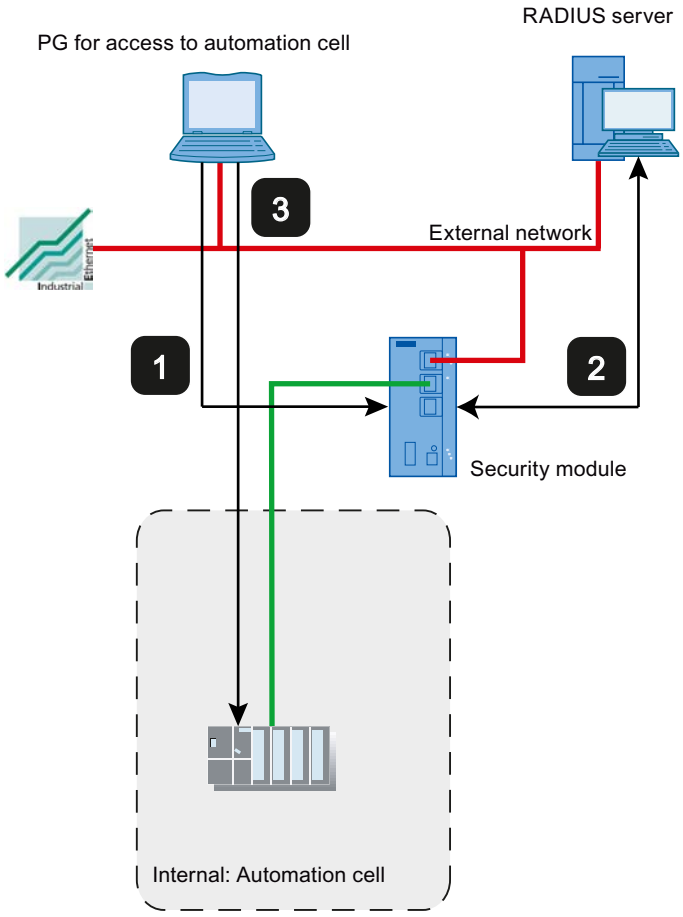

1 Entry of the user data on the Web page of the security module
2 Authentication by RADIUS server and activation of the user-specific IP rule set
3 Access to an automation cell

The network setup shown above is simply an example. The RADIUS server can also be located in the internal network or in the DMZ network of the security module.

For the configuration options described below, it is assumed that a RADIUS server is configured in STEP 7 and was assigned to the relevant security module. In addition to this, one user or role must be configured with the "RADIUS" authentication method. For more detailed information, refer to the following sections:

- Defining a RADIUS server (Page 492)
- Assigning a RADIUS server to a security module (Page 493)

For general information on user-specific IP rule sets, refer to the following section:

- AUTOHOTSPOT

Configuration options

To authenticate the user using a RADIUS server, there are two configuration options available:

- The user and the user's role are known on the security module, only the password management for the user is performed on the RADIUS server. The user and the password are configured on the RADIUS server.
  - A user with the "RADIUS" authentication method is configured.
  - The user is assigned to the user-specific IP rule set.

  Result:
  - When a user logs on to the Web page of the security module, the authentication query is forwarded to the RADIUS server.
  - The RADIUS server runs a password check and signals the result back to the security module.
  - If the password check is passed successfully, the user-specific IP rule set is activated.
- The role is known on the security module, user management is via the RADIUS server. The user and the password are configured on the RADIUS server.
  - A user-defined role or a system-defined role is assigned to the user-specific IP rule set.
  - Under the entry "RADIUS" > "RADIUS settings" in the local security settings of the security module, the check box "Allow RADIUS authentication of unconfigured users" and the check box "Filter ID is required for authentication" are enabled.

  Result:
  - When a user logs on to the Web page of the security module, the authentication and authorization query is forwarded to the RADIUS server.
  - The RADIUS server runs a password check and signals the result back to the security module.
  - Case a: If, in addition to this, the role name is configured on the RADIUS server:
    The RADIUS server returns the role name assigned to the user to the security module.
  - Case b: If the role name is not configured on the RADIUS server:
    The security module assigns the user the system-defined role "radius".
  - If the password check is passed successfully, the user-specific IP rule set is activated.

Conventions for RADIUS servers

- The RADIUS servers can be in any network connected to the security module.
- A maximum of two RADIUS servers can be configured per security module. During operation only one of the RADIUS servers is active.
- When defining a RADIUS server, an FQDN can also be used instead of IP an address.
- A maximum of 25 characters can be used for the name of a RADIUS server.

Defining a RADIUS server

Meaning

Before authentication by a RADIUS server is possible, this first needs to be stored in the STEP 7 project. Following this, you assign the defined RADIUS server to the security module for which the RADIUS server will handle user authentication.

Procedure

1. In the global security functions, select the entry "RADIUS".
   Result: The previously created RADIUS servers are displayed under the selected entry.
2. Double-click on the "Add new RADIUS server" entry.
   Result: The created RADIUS server is displayed with an automatically assigned number below the entry "RADIUS".
3. Double-click on the created RADIUS server.
   Result: In the working area the assignment dialog for RADIUS servers is displayed. The selected RADIUS server is selected in the "RADIUS server" drop-down list. The required security modules can be assigned to this, see section Assigning a RADIUS server to a security module (Page 493). In the local security settings, the configurable properties of the RADIUS server are displayed.
4. Enter the following data in the "General" entry in the local security settings:

Parameter	Meaning
Name	Name of the RADIUS server. The name can consist of a maximum of 25 characters. The characters 0-9, a-z, A-Z, -._ can be used. A name must start with a letter and end with a letter or number. Before the '.' character only letters and numbers are permitted, after the '.' character only letters. Before the '_' or '-* character only letters and numbers are permitted.
IP address / FQDN	IP address or FQDN of the RADIUS server.
Port	UDP port via which the RADIUS server can be reached. As default, authentication data is received at port 1812.
Shared secret	Entry of the password that will be used when transferring the logon data between the RADIUS server and security modules for encryption. The following characters from the ANSI X 3.4-1986 character set are permitted: 0123456789 A...Z a...z !#$%&()"*'+`,-./:;<=>?@ [\]_{|}~^ Length of the shared secret: 1 to 32 characters
Repeat shared secret	Confirmation of the password
Authentication method	Display of the method used to check the user data. Only the "PAP" method (Password Authentication Protocol) is supported.
Comment	Entry of freely selectable, optional comments.

Result

You have defined a RADIUS server and can now assign this to the required security modules.
Note the descriptions in the section below:
Assigning a RADIUS server to a security module (Page 493)

Assigning a RADIUS server to a security module

Requirement

You have defined a RADIUS server.

Procedure

1. Select the module to be edited.
2. Select the "RADIUS" entry in the local security settings.
3. Select the "Enable RADIUS authentication" check box.

   Note

   Changing the method of authentication with the Web server on the security module

   If RADIUS authentication is enabled on the security module, the method for authentication with the Web server is changed from "Digest Access Authentication" to "Basic Access Authentication".

4. In the "RADIUS timeout" input box, enter the maximum time in seconds that the security module will wait for a response from the RADIUS server.
5. In the "RADIUS repetitions" input box, enter the number of connection establishment attempts with the RADIUS server.
6. Select the "Allow RADIUS authentication of unconfigured users" check box if the user-specific IP rule to be activated was assigned a role instead of a user.
7. Select the "Filter ID is required for authentication" check box if the assigned role is a user-defined role.
8. Under the entry "RADIUS server", select the RADIUS server you want to assign to the security module from the "Name" drop-down list.
   As an alternative, you can assign security modules for which RADIUS authentication is enabled to RADIUS servers in the global security functions. You will find general information on authentication by the RADIUS server in the following section:
   AUTOHOTSPOT

Setting up a firewall

Local firewall rules for SCALANCE S modules

Configuring a firewall with predefined firewall rules

Configuring a firewall using predefined firewall rules

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" entry in the local security settings.

Firewall enabled as default

The "Enable firewall" check box is enabled by default. The firewall is therefore activated automatically and all access from external to the security module is blocked. By clicking the relevant check box, enable the firewall rules for the specific direction.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Enable firewall in advanced mode" check box. For more detailed information about the advanced firewall mode, refer to the section
Overview of local firewall rules (Page 393)

Firewall configuration with VPN

If the security module is in a VPN group, the "Tunnel communication only" check box is enabled as default. This means that only encrypted IPsec data transfer is permitted via the external interface or via the DMZ interface. Only HTTPS access to the module (TCP port 443) remains allowed untunneled.

If you deselect the check box, tunneled communication and also the types of communication selected in the other boxes are permitted.

Table 1-52 Available firewall rules and directions

Service	From internal to external	From external to internal	From internal to DMZ	From DMZ to internal	Enabled ports	Meaning
Allow IP communication	x	x	x	x	-	IP communication for the selected communication directions is allowed.
Allow S7 protocol	x	x	x	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.

Service	From internal to external	From external to internal	From internal to DMZ	From DMZ to internal	Enabled ports	Meaning
Allow FTP/ FTPS (explicit mode)	x	x	x	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	x	x	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	x	x	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow DNS	x	x	x	x	TCP port 53 UDP port 53	Communication connection to a DNS server is allowed.
Allow SNMP	x	x	x	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	x	x	x	TCP port 25	For sending e-mails via an SMTP server.
Allow NTP	x	x	x	x	UDP port 123	For synchronization of the time of day.
Allow DHCP	x	x	x	x	UDP port 67 UDP port 68	Only in bridge mode Communication connection with a DHCP server is permitted.

Table 1-53 Logging

Option	Action when activated
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.
Log blocked incoming packets	All incoming IP packets that are discarded are logged.
Log blocked outgoing packets	All outgoing IP packets that are discarded are logged.

You can view the logged packets at the "Packet filter log" entry in the "Online & Diagnostics" dialog. For more detailed information, refer to the section Logging data packets - "Packet filter log" entry (Page 468).

Configuring a firewall with predefined MAC rules

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" entry in the local security settings.

Firewall enabled as default

The "Enable firewall" check box is enabled by default. The firewall is therefore activated automatically and all access from external to the security module is blocked. By clicking the relevant check box, enable the firewall rules for the specific direction.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Enable firewall in advanced mode" check box. For more detailed information about the advanced firewall mode, refer to the section Overview of local firewall rules (Page 393).

Firewall configuration with VPN

If the security module is in a VPN group, the "Tunnel communication only" check box is enabled as default.

If you deselect the check box, tunneled communication and also the types of communication selected in the other boxes are permitted.

Available MAC rules and directions

Service	From internal to external	From external to internal	Meaning
Allow MAC communication	x	x	MAC traffic for the selected communication directions is allowed.
Allow ISO protocol	x	x	ISO traffic for the selected communication directions is allowed.
Allow SiClock	x	x	SiClock time-of-day frames for the selected communication directions are allowed.
Allow DCP	x	x	DCP traffic for assignment of IP addresses is allowed for the selected communications directions.

Table 1-54 Logging

Option	Action when activated
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.

User-specific IP rule sets

Overview

Meaning

Initially, individual or multiple users are assigned to user-specific IP rule sets. The user-specific IP rule sets are then assigned to individual or multiple security modules. This makes it possible, to allow user-specific access. If, for example all access to the networks downstream from a security module is blocked, certain nodes can be allowed for a user based on their IP addresses. This means that access is allowed for this user but access remains blocked for other users.

User logon via the Internet

The user can log in to the external interface or the DMZ interface via the Web page of the security module. If authentication is successful, the IP rule set defined for the user for the IP address of the device from which the login is made is enabled.

The connection to the Web page of the security module is via HTTPS using the IP address of the connected port and taking into account the valid routing rules:

Example:

External interface: 192.168.10.1

Call up of the login page with: https://192.168.10.1/

Users can log on with every role as long as the user or the role is assigned to a user-specific firewall rule set.

Options for authenticating the user

Depending on the authentication method selected when the user who will log in to the security module was created, the authentication is handled by different instances:

- Authentication method "Password": Authentication is handled by the security module.
- Authentication method "RADIUS": Authentication is handled by a RADIUS server (SCALANCE S as of V4 only).

Assignment of roles to user-specific IP rule sets

On SCALANCE S modules as of V4, it is also possible to assign user-specific IP rule sets that are assigned to roles. This makes it possible to enable a group of users for access to certain IP addresses.

If a RADIUS server is used for user authentication and a role is assigned to the user-specific IP rule set, users can also be authenticated by the RADIUS server although they are not configured on the security module. These users must be stored on the RADIUS server or on a separate database where they need to be assigned to the role assigned to the user-specific IP rule set in STEP 7. This procedure has the advantage that all user data is stored exclusively on the RADIUS server.

You will find more information on authentication by the RADIUS server in the following section:

AUTOHOTSPOT

User-specific IP rule sets are used locally - conventions

The same conventions as described in the following section apply:
Global firewall rule sets - conventions (Page 373)

Creating and assigning user-specific IP rule sets

Creating user-specific IP rule sets

1. In the global security functions, select the entry "Firewall" > "User-specific IP rule sets" > "IP rule sets".
2. Double-click on the "Add new IP rule set" entry to create a user-specific IP rule set.
   Result: The created user-specific IP rule set is shown in the entry.
3. Double-click on the created user-specific IP rule set.
   Result: In the "Properties" > "General" tab of the Inspector window, the configurable properties of the user-specific IP rules set are displayed.
4. In the Inspector window, click on the "General" entry and enter the following data:
   – Name: Project-wide, unique name of the rule set. The name appears in the local rule list of the security module after the rule set is assigned.
   – Description (optional): Enter a description of the user-specific IP rule set.
5. Click on the "IP rules" entry and enter the firewall rules one after the other in the list.
   No IP address can be entered in the “Source IP address” box. This is entered automatically when the node logs on to the security module.
   The “Stateful” check box cannot be disabled for IP rules of user-specific IP rule sets
   Note the parameter description in the following sections:
   Defining IP packet filter rules (Page 394)
   Note the special features of firewall rules generated automatically by STEP 7 for NAT/NAPT rules:
   Relationship between NAT/NAPT router and user-specific firewall (Page 505)

Assigning user-specific IP rule sets

1. Click on the "Users and roles" entry in the Inspector window.
2. In the "Available users and roles" area, select the users you want to assign to the user-specific rule set.

3. Click the "<<" button to assign the selected users or roles to the user-specific rule set. The assignment of roles to user-specific IP rule sets is possible only for SCALANCE S modules as of V4.
4. Assign the created user-specific IP rule set to the required security modules using the entry "Assign user-specific IP rule set" in the global security functions. For this, advanced firewall mode must be enabled for the security modules.

Note

Assignment of user-specific IP rule sets

- A security module can only be assigned one user-specific rule set per user.
- Due to the assignment, the right to log in to the security module is activated implicitly for all users and roles assigned to the IP rule set.

Result

- The user-specific rule set is used by the assigned security modules as a local rule set and automatically appears in the local list of firewall rules.
- The user can log on with the security module. Authentication of the user is performed depending on the selected authentication method either by the security module or a RADIUS server.

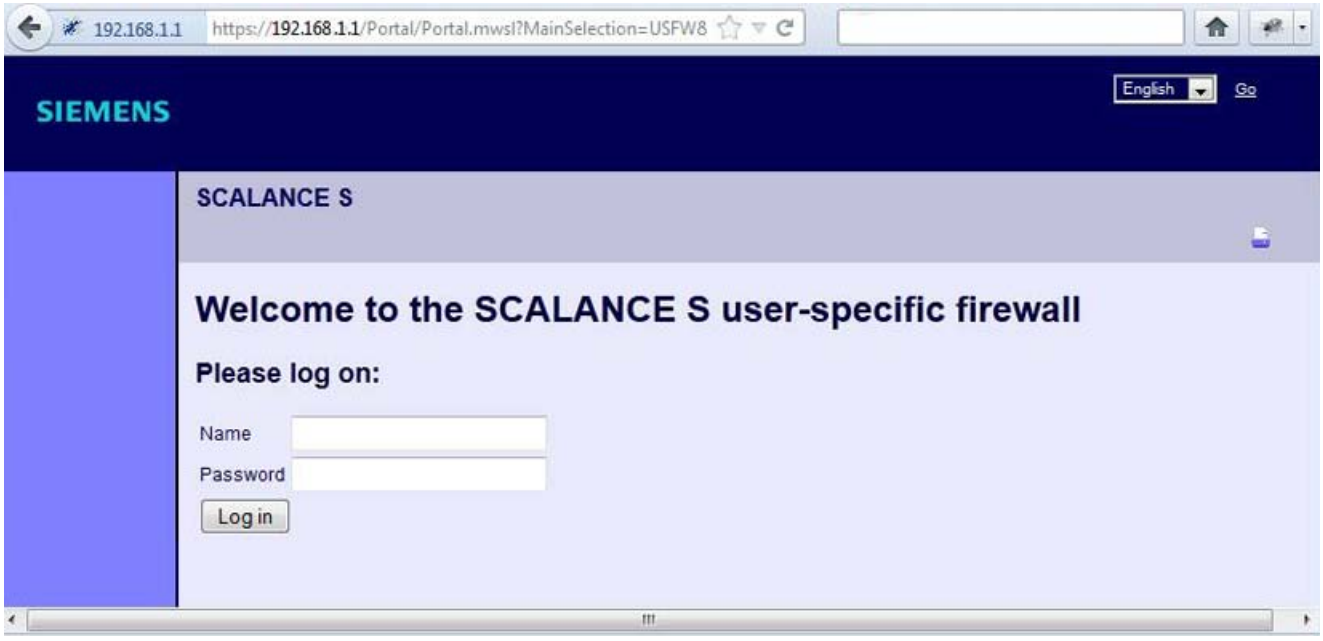

Range of values for the maximum session time

The time after which the user is automatically logged out can be specified when creating or editing a user; the default is 30 minutes. On the Web page of the security module, the session time can be extended to the value assigned to the user.

IP packet filter directions SCALANCE S

Meaning

Possible selections for the communication directions "From" and "To" in the IP rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		
From	To	S602	S61x	S623 / S627-2M
Internal	External	x	x	x
	Tunnel	-	x	x
	Any	-	x	x
	DMZ	-	-	x
	Internal	x	x	x
External	Internal	x	x	x
	Any	-	-	x
	Tunnel	-	-	x
	DMZ	-	-	x
Tunnel	Internal	-	x	x
	External	-	x	x
	DMZ	-	-	x
Any	Internal	-	x	x
	External	-	-	x
	DMZ	-	-	x
DMZ	Internal	-	-	x
	External	-	-	x
	Any	-	-	x
	Tunnel	-	-	x

x = communication direction is configurable

- = communication direction is not configurable

MAC packet filter directions SCALANCE S

Meaning

Possible selections for the communication directions "From" and "To" in the MAC rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		
From	To	S602	S61x	S623 / S627-2M
Internal	External	x	x	x
	Tunnel	-	x	x
	Any	-	x	x
External	Internal	x	x	x
	Any	-	-	x
	Tunnel	-	-	x
Tunnel	Internal	-	x	x
	External	-	x	x
Any	Internal	-	x	x
	External	-	-	x

x = communication direction is configurable

- = communication direction is not configurable

Adapting standard rules for IP services

Requirement

This function is available only in advanced firewall mode.

How to access this function

1. Select the security module to be edited.
2. In the local security settings, select the entry "Firewall" > "Standard rules for IP services".

Meaning of the advanced settings

Parameter	Meaning when activated
Use advanced status options	The permitted number of connections and firewall statuses per time unit is limited. If a network node exceeds this limit, its IP address is entered in the IP blacklist of the security module. The node can then no longer communicate via the security module. The IP blacklist of the security module can be viewed in online mode.
Log all activated rules	Packets that are allowed according to the default rules for IP services are logged.
Enable ICMP test for interfaces	Ping queries coming in on an interface of the security module can be forwarded to other interfaces. This means that, for example, ping queries can be made from the external network to the internal interface of the security module.

Default firewall rules for SCALANCE S

The following table lists the default firewall rules for SCALANCE S modules. In some cases, the firewall rules are only set if the relevant service is used by the security module (e.g. SNMP).

Service	Direction	X1 interface (red)	Interface X2 (green)	Interface X3 (yellow) (only for S623, S627-2M)	Tunnel interface* (not for S602)
Interface rerouting	outgoing	-	x	-	-
HTTPS		x	x*	x	x
ICMP	incoming	-	x	-	x
ICMP pathfinder (only for SCALANCE S S602 as of V3.1 in ghost mode)	outgoing	-	x	-	-
SNMP	incoming	x	x	x	x
Syslog	outgoing	x	x	x	x
NTP	outgoing	x	x	x	x
DNS	outgoing	x	x	x	x
HTTP	outgoing	x	-	x	-
VPN (IKE)		x	-	x	-
VPN (NAT Traversal)		x	-	x	-
BootP Server	incoming	-	x	x	-
BootP Client	outgoing	-	x	x	-
RADIUS	outgoing	x	x	x	x
CARP (only for SCALANCE S62x as of V4)	outgoing	x*	x*	-	-
Pfsync (only for SCALANCE S62x as of V4)	outgoing	-	-	x*	-

x enabled as default

- disabled as default

* cannot be adapted

SCALANCE S module as router

Specifying routes

Meaning

Specifies routes for addressing subnets that cannot be reached directly via the security module.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the "Routing" entry.
3. Double-click "Add new" in the table to add a route.
4. Enter the following values:

Parameter	Function	Example of a value
Network ID	Requests to nodes of the subnet with the network ID and the subnet mask specified here are forwarded to the subnet via the specified router IP address. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet. The specified network ID must not be located in the same subnet as the IP address of the security module.	192.168.11.0
Subnet mask	The subnet mask determines the network structure. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	255.255.255.0
Router IP address	IP address of the router that connects to the subnet. As an alternative, an FQDN can be entered for SCALANCE S modules as of V4. The specified router IP address must be in the same subnet as the IP address of the security module.	192.168.10.2
Activate rerouting (only for SCALANCE S as of V3)	Select this check box if the frames of the entered route will enter and leave on the same interface of the security module (rerouting). Rerouting is only supported on the internal interface of the security module.	

Address translation with NAT/NAPT in VPN tunnels

Module-specific function

Address translation with NAT/NAPT in VPN tunnels is only available for SCALANCE S612/S623/S627-2M modules as of V4.

Meaning

Address translations with NAT/NAPT can also be performed for communications relations established via a VPN tunnel.

Requirements

The following requirements apply generally to a SCALANCE S module that will perform an address translation with NAT/NAPT in a VPN tunnel:

- The SCALANCE S module is in a VPN group.
- The SCALANCE S module is in routing mode and/or the DMZ interface of the SCALANCE S module is activated.

- The tunnel interface is enabled.
- The advanced firewall mode is enabled.
- The firewall is enabled in the editor for IP rules.

Supported address translation directions

The address translation directions described in the following section are supported:
NAT/NAPT routing (Page 411)

Supported address translation actions

With tunneled communications relations, the following address translation actions are supported:

- Destination NAT ("Redirect")
- Source NAT ("Masquerading")
- Source and Destination NAT ("1:1-NAT")
- NAPT ("Port forwarding")

You will find basic information on these address translation actions in the following section:
NAT/NAPT routing (Page 411)

Supported VPN links

In conjunction with NAT/NAPT, the following VPN links are supported:

VPN link		VPN link is initiated by	Address translation is performed by
SCALANCE S (a)	SCALANCE S (b)	SCALANCE S (a) or SCALANCE S (b)	SCALANCE S (a) and/or SCALANCE S (b)
SCALANCE S	CP 343-1 Adv. / 443-1 Adv. / PC-CP	SCALANCE S or CP 343-1 Adv. / 443-1 Adv. / PC-CP	SCALANCE S
SCALANCE S	SCALANCE M	SCALANCE M	SCALANCE S and/or SCALANCE M*
SOFTNET Security Client	SCALANCE S	SOFTNET Security Client	SCALANCE S

* Only 1:1 NAT is supported.

SCALANCE S modules of the type SCALANCE S623 V4 and SCALANCE S627-2M V4 that have a VPN endpoint on the external interface and on the DMZ interface can perform address translations on both interfaces at the same time.

Address translation characteristics when involved in several VPN groups

If a SCALANCE S module is a member of several VPN groups, the address translation rules configured for the tunnel interface of the SCALANCE S module apply for all VPN connections of this SCALANCE S module.

Please note: Once you have configured a NAT address translation to or from the direction of the tunnel, only the IP addresses involved in the NAT address translation rules can be reached via the VPN tunnel.

Relationship between NAT/NAPT router and user-specific firewall

Module-specific function

The configuration of NAT/NAPT rules in the user-specific firewall is only available for SCALANCE S modules as of V3.

Meaning

After creating NAT/NAPT rules, STEP 7 automatically generates a user-specific IP rule set in the user-specific firewall that enables communication in the configured address translation direction. You can then assign this user-specific IP rule set to individual or multiple users and/or individual or multiple roles (only for SCALANCE S modules as of V4).

The generated firewall rules can, if necessary, be moved and expanded (additional IP address, services, transmission speed). Firewall parameters generated by STEP 7 cannot be adapted. If the user-specific IP rule set is assigned to a security module with NAT/NAPT deactivated, the NAT/NAPT rules from the user-specific firewall are also applied to this security module.

Note

The address translation actions "Source-NAT + Destination-NAT" and "Double-NAT" are not supported in conjunction with the user-specific firewall.

How to access this function

"NAT" or "NAPT" entry in the editor for user-specific IP rules sets, refer to the following section: Creating and assigning user-specific IP rule sets (Page 498)

Supported address translation directions for the action "Source NAT"

The action "Source NAT" can be performed in the following directions:

- External to DMZ
- DMZ to external

No IP address can be entered in the "Source IP address" box. This is entered automatically when the node logs on to the security module.

Supported address translation directions for the action "Destination NAT"

The action "Destination NAT" can be performed in the following directions:

- External to internal
- External to DMZ

- DMZ to internal
- DMZ to external
- Tunnel to internal (only SCALANCE S612/S623/S627-2M as of V4)
- Tunnel to external (only SCALANCE S612/S623/S627-2M as of V4)
- Tunnel to DMZ (only SCALANCE S612/S623/S627-2M as of V4)

Supported address translation directions for NAPT

The address translation with NAPT can be performed in the following directions:

- External to internal
- External to DMZ
- DMZ to internal
- DMZ to external
- Tunnel to internal (only SCALANCE S612/S623/S627-2M as of V4)
- Tunnel to external (only SCALANCE S612/S623/S627-2M as of V4)
- Tunnel to DMZ (only SCALANCE S612/S623/S627-2M as of V4)

NAT/NAPT address translation and corresponding user-specific IP rule sets

In the firewall rules for user-specific IP rule sets generated based on NAT/NAPT rules, no IP address can be entered in the "Source IP address" box. This is entered automatically when the node logs on to the security module. The remaining properties generated locally for individual security modules are identical. Refer to the section
Relationship between NAT/NAPT router and firewall (Page 423)

Security module as DHCP server

Overview DHCP server

Overview

You can operate the SCALANCE S module on the internal network and on the DMZ network as DHCP server (DHCP = Dynamic Host Configuration Protocol). This allows IP addresses to be assigned automatically to the devices connected to the internal network or the DMZ network.

Simultaneous DHCP server operation on both interfaces is possible (SCALANCE S623/S627-2M only).

The IP addresses are either distributed dynamically from an address range you have specified or you can select a specific IP address and assign it to a particular device. If devices on the internal interface or on the DMZ interface should always be assigned the same IP address for firewall configuration, the address assignment must only be static based on the MAC address or based on the client ID.

Requirements

You configure the devices on the internal network or on the DMZ network so that they obtain the IP address from a DHCP server.

Depending on the mode, the security module sends an IP address of the standard router to the nodes in the relevant subnet or you need to inform the nodes in the subnet of a router IP address.

- Router IP address will be transferred
  In the following situations, the DHCP protocol of the security module will inform the nodes of the router IP address:
  - The node is connected to the DMZ interface (SCALANCE S623/S627-2M only)
    In this case, the security module sends its own IP address as the router IP address.
  - The node is connected to the internal interface and the security module is configured for router mode
    In this case, the security module sends its own IP address as the router IP address.
  - The node is connected to the internal interface and the security module is not configured for router mode, there is, however, a standard router specified in the configuration of the security module.
    In this case, the security module transfers the IP address of the standard router as the router IP address.
- Router IP address will not be transferred
  In the following situations, enter the router IP address manually on the node:
  - The node is connected to the internal interface and the security module is not configured for router mode. There is also no standard router specified in the configuration of the security module.

See also

Configuring a DHCP server (Page 507)

Configuring a DHCP server

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "DHCP server".

3. Select the interface for which you want to make the DHCP settings.
4. Make the address assignment. You have the following configuration options:
   – Static address assignment
     Devices with a specific MAC address or client ID are assigned the specified IP addresses. You specify these addresses by entering the devices in the address list in the "Static address assignment" group box.
   – Dynamic address assignment
     Devices whose MAC address or whose client ID was not specified specifically, are assigned a random IP address from the specified address range. For this purpose, activate the "Dynamic IP address pool" check box. Set the address range in the "Dynamic IP address pool" input area.

Note

Dynamic address assignment - reaction after interrupting the power supply

Please note that dynamically assigned IP addresses are not saved if the power supply is interrupted. On return of the power, you must therefore make sure that the nodes request an IP address again.

You should therefore only use dynamic address assignment for the following nodes:

- Nodes that are used temporarily in the subnet (such as service devices).
- Nodes that have been assigned an IP address and send this as the "preferred address" the next time they request an address from the DHCP server (for example PC stations).

For permanent nodes you should preferably use static address assignment by specifying a client ID or the MAC address.

Consistency check - these rules must be adhered to

Remember the following rules when making the entries.

- The IP addresses assigned in the address list in the "Static address assignments" group box must not be in the range of the dynamic IP addresses.
- IP addresses, MAC addresses and client IDs may only occur once in the "Static address assignment" table (related to the security module).
- For the statically assigned IP addresses, you must specify either the MAC address or the client ID (computer name).
- The client ID is a string with a maximum of 63 characters. Only the following characters may be used: a-z, A-Z, 0-9 and - (dash).
  Note
  In SIMATIC S7, a client ID can be assigned to the devices on the Ethernet interface to allow them to obtain an IP address using DHCP.
  With PCs, the procedure depends on the operating system being used; it is advisable to use the MAC address here for the assignment.
- For the statically assigned IP addresses, you must specify the IP address.

- The following IP addresses must not be in the range of the free IP address range (dynamic IP addresses):
  - All router IP addresses in the "Routing" entry
  - Syslog server
  - Standard router
  - Security module address(es)
- DHCP is supported by the security module on the interface to the internal subnet and on the interface to the DMZ network. The following additional requirements for IP addresses in the range of the dynamic address assignments result from this operational behavior of the security module:
  - Bridge mode
    The range must be within the network defined by the security module.
  - Routing mode
    The range must be within the internal subnet defined by the security module.
    Note
    The DMZ network always represents a separate subnet. When using DHCP on the DMZ interface, make sure that the free IP address range (dynamic IP addresses) is within the DMZ subnet.
- The free IP address range must be fully specified by entering the start address and the end address. This end address must be higher than the start address.
- The IP addresses you enter in the address list in the "Static address assignments" input area must be in the address range of the internal subnet or in the DMZ subnet of the security module.

See also

Running a consistency check (Page 352)

Configuring proxy ARP

Overview

Proxy ARP allows routers to respond to ARP queries for hosts. The hosts are in networks separated by routers but use the same IP address range.

If PC1 sends an ARP request to PC2, it receives an ARP response and the hardware address of the interface (MAC address of the port of the security module) on which the query was received from the security module located in between and not from PC2. The querying PC1 then sends its data to the security module that then forwards it to PC2.

How to access this function

This function is only available for the internal interface of a security module that is a member of a VPN group and is in bridge mode.

1. Select the security module to be edited.
2. In the local security settings, select the entry "Proxy ARP".
3. If the security module is to respond to an ARP query from its own LAN as a substitute for the specific connection partner, enter the corresponding IP address.

IPsec tunnel: Creating and assigning groups

Configuring internal network nodes

Using the learning mode to learn internal nodes

Finding nodes for tunnel communication automatically

One great advantage when configuring and operating tunnel communication is that SCALANCE S modules in bridge mode can find the nodes on the internal interface automatically. New nodes are detected by the security module during operation. The detected nodes are signaled to the security modules belonging to the same VPN group. This allows data exchange within the tunnels of a group in both directions at any time.

Detectable nodes

The following nodes are detected:

- Network nodes with IP capability
  Network nodes with IP capability are found when they transmit an ICMP response to the ICMP subnet broadcast.
  IP nodes downstream from routers can be found if the routers pass on ICMP broadcasts.
- ISO network nodes
  You can also teach-in network nodes without IP capability that can be addressed by means of ISO protocols.
  This is only possible if they reply to XID or TEST packets. TEST and XID (Exchange Identification) are auxiliary protocols for exchanging information on layer 2. By sending these packets with a broadcast address, these network nodes can be located.
- PROFINET nodes
  Using DCP (Discovery and basic Configuration Protocol), it is possible to find PROFINET nodes.

Network nodes that do not meet these conditions must be configured manually.

Subnets located on the other side of internal routers also need to be configured manually.

How to access the function

1. Select the module.
2. In the local security settings, select the entry "VPN" > "Nodes".

Enabling/disabling the learning mode

The learning function is enabled in the configuration as default for every security module.

Learning can also be disabled completely for SCALANCE S. In this case, you will need to configure all internal network nodes participating in the tunnel communication manually.

When is it useful to disable the automatic learning mode?

The default settings for the security module assume that internal networks are always secure; in other words, in a normal situation, no network node is connected to the internal network if it is not trustworthy.

Disabling the learning mode can be useful if the internal network is static; in other words, when the number of internal nodes and their addresses do not change.

If the learning mode is disabled, this reduces the load on the medium and the nodes in the internal network resulting from the learning packets. The performance of the security module is also slightly improved since it does not need to process the learning packets.

Note: In the learning mode, all nodes in the internal network are detected. The information relating to VPN configuration limits relates only to nodes that communicate over VPN in the internal network.

Note

If more than 128 internal nodes are being operated, the permitted configuration limits are exceeded and an illegal operating status results. Due to the dynamics in the network traffic, this causes internal nodes that have already been learned to be replaced by new previously unknown internal nodes.

See also

Configuring internal subnets manually (Page 512)

Configuring IP network nodes manually

Meaning

As an alternative to the learning mode that you enable using the "Enable learning of internal nodes" check box and that allows the security module to learn the internal network nodes dynamically, you can enter the network nodes to be learned manually in the "Internal IP nodes" entry and in doing so enable them for VPN tunnel communication. The MAC address of a network node can be specified as an option.

Requirement

- The security module is in bridge mode.
- The security module is a member of a VPN group.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN" > "Nodes" > "Internal IP nodes".

Configuring MAC network nodes manually

Meaning

As an alternative to the learning mode that you enable using the "Enable learning of internal nodes" check box and that allows the security module to learn the internal network nodes dynamically, you can enter the network nodes to be learned manually in the "Internal MAC nodes" entry and in doing so enable them for VPN tunnel communication.

Requirement

- The security module is in bridge mode.
- The security module is a member of a VPN group.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN" > "Nodes" > "Internal MAC nodes".

Configuring internal subnets manually

Requirement

- The security module is a member of a VPN group.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN" > "Nodes" > "Internal subnets".

Security module in bridge mode - "Internal subnets" entry

To be able to enable internal subnets for VPN tunnel communication manually, you need to enter the following address parameters:

Parameter	Function	Example of a value
Network ID	Network ID of the subnet to be enabled for VPN tunnel communication. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet. Must not be located in the same subnet as the IP address of the security module.	192.168.11.0
Subnet mask	The subnet mask determines the network structure. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	255.255.255.0
Router IP address	IP address of the router via which the subnet you are allowing is reached. Must be located in the same subnet as the IP address of the security module.	192.168.10.2

Security module in routing mode - "Subnets reachable through tunnel" entry

In routing mode, entire subnets are always tunneled. To be able to enable internal subnets reachable via routers, the external subnet or the DMZ subnet for VPN tunnel communication manually, you need to enter the following address parameters:

Parameter	Function	Example of a value
Network ID	Network ID of the subnet to be enabled for VPN tunnel communication. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	192.168.11.0
Subnet mask	The subnet mask determines the network structure. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	255.255.255.0
Comment	Entry of additional, optional comments.	

Router and firewall redundancy

Overview

Meaning

Failures of the security modules SCALANCE S623 as of V4 and SCALANCE S627-2M as of V4 can be automatically compensated by routers and firewall redundancy during operation. To do this, group two security modules of the type SCALANCE S623 or SCALANCE S627-2M in a redundancy relationship by activating routers and firewall redundancy for both security modules. Following this, you decide which security module of the redundancy relationship is passive in normal mode (secondary module). You make this setting for the security module of the redundancy relationship that is active in normal operation (primary module). If the primary module fails during operation, the secondary module automatically adopts the function as firewall and (NAT/NAPT) router. To ensure the identical configuration of both security modules, these are connected together via their DMZ interfaces and their configurations are synchronized during operation. In this case, the DMZ interfaces of the security modules involved cannot be used for other purposes.

Address redundancy

In addition to their module IP addresses, the two security modules share a common IP address on the external and on the internal interface so that if one of the security modules fails, the IP addresses do not need to be changed. To do this, you need to configure an IP address for the external and for the internal interface of the redundancy relationship.

Effects of redundancy relationships on security modules

When you create redundancy relationships between security modules, some properties of these security modules are automatically adapted to establish compatibility with the redundancy relationship. The following properties are affected by this adaptation:

Module property	Effect on the module property
Operating mode	Where necessary, the mode is changed to the "Routing mode" option.
Membership of VPN groups	Where necessary, the security module is removed from VPN groups.
Interface configuration	Where necessary, the external interface and the DMZ interface of the security module are enabled. Where necessary, the IP assignment method "Static address" is configured for all interfaces.

Configuration of security modules in redundancy relationships

After router and firewall redundancy have been enabled and the primary module of the redundancy relationship selected, some of the module properties are configured only using the primary module. The properties configured for the primary module then apply to the redundancy relationship and cannot be configured for the secondary module. The following properties can be configured for the redundancy relationship:

- Basic settings of the redundancy relationship (secondary module, network parameters)
- Firewall (standard rules for IP services are configured separately for the individual security modules).
- Routing
- NAT/NAPT routing (not 1:1 NAT)

The values of the properties listed above are initially adopted from the primary module for the redundancy relationship.

The settings listed below remain active for the individual security modules even after including them in a redundancy relationship. Configuring these properties for the primary module therefore has no effect on the secondary module.

- Interface configuration (disabling interfaces and changing the VIP assignment method "Static address" is not possible).
- Standard rules for IP services (firewall)
- DDNS
- Time-of-day synchronization
- Log settings
- SNMP
- RADIUS

Note

Loading a configuration on security modules of a redundancy relationship (only SCALANCE S623/S627-2M as of V4)

The properties of a redundancy relationship configured for the primary module must be loaded both on the primary module and on the secondary module. To load the configuration, the physical IP address via which your engineering station can reach the security module must be used. The virtual IP addresses of the redundancy relationship cannot be used for loading.

Note

Configuring routing when using routers and firewall redundaancy

In a redundancy relationship the only routing information that is synchronized between the primary and secondary module is that that was configured statically in the local security settings of the primary module in the “Routing” entry. Routing entries resulting dynamically due to using standard routers are not synchronized. When using routers and firewall redundancy it is therefore recommended that you configure all known routers statically.

Creating redundancy relationships between security modules

Requirement

The security modules SCALANCE S623/S627-2M as of V4 are not assigned to any other redundancy relationship.

Procedure

1. Select the security module that will be the active security module in normal operation (primary module).
2. In the local security settings, select the entry "Router and firewall redundancy".
3. Select the "Router and firewall redundancy" check box.
4. In the "Secondary module" drop-down list, select the security module that will be the passive security module in normal operation.

Result: You have created a redundancy relationship between the security modules.

Configuring redundancy relationships

How to access this function

1. Select the primary module of the redundancy relationship.
2. In the local security settings, select the entry "Router and firewall redundancy".

Configuring network parameters of the redundancy relationship

Configurable parameter	Meaning
IP address	IP address of the virtual external or internal interface of the redundancy relationship. The IP address must be located in the external or internal subnet of the primary module.
Subnet mask	Subnet mask of the virtual external or internal interface of the redundancy relationship
MAC address (adaptable only for SCALANCE S623/ S627-2M as of V4.0.1)	MAC address of the virtual external or internal interface of the redundancy relationship

For general information on configuring network parameters, refer to the following section:
Configuring IP address parameters (Page 475)

Configuring the firewall

IP packet filter rules for redundancy relationships are configured on the primary module. The communications directions "From external to internal" and "From internal to external" are available.

For general information on configuring IP packet filter rules in advanced firewall mode, refer to the following section:
Defining IP packet filter rules (Page 394)

Configuring address translation with NAT/NAPT

Address translation with NAT/NAPT for the redundancy relationship is configured on the primary module. For redundancy relationships, only Source NAT and NAPT can be configured. With Source NAT, source IP addresses in the internal subnet can only be replaced with the virtual external IP address of the redundancy relationship. No alias IP addresses can be registered on the external interface of the redundancy relationship. With NAPT, only the "External to internal" address translation direction can be configured.

For general information on configuring address translations with NAT/NAPT, refer to the following section:
Overview of NAT/NAPT (Page 409)

Configuring routing

Routes for the redundancy relationship are configured on the primary module. Standard routers must be specified in the "External interface [P1] red" or "Internal interface [P2] green" entry and must be identical per interface.

For general information on configuring routing, refer to the following section:
Specifying routes (Page 502)

Online functions - diagnostics and logging

Overview of the individual interfaces - "Interface settings" entry

Meaning

Table 1-55 Online diagnostics: "Interface settings" entry

System and status functions	Meaning
Interfaces	Table above: General overview of the interfaces of the security module. Table below: Information on the interface operated via PPPoE. • Status: Indicates whether or not a connection was established to the Internet Service Provider (ISP). • Current IP address: Current IP address of the interface • Gateway: IP address of the gateway • Primary dynamic DNS service: IP address of the primary dynamic DNS service • Secondary dynamic DNS service: IP address of the secondary dynamic DNS service • Error code (numeric): Error information if no connection could be established to the ISP.
CARP (only SCALANCE S623/S627-2M as of V4)	• CARP interface: Display of the virtual CARP interface • Physical interface: Physical interface on which the virtual CARP interface is operated (external / internal). • Status: Displays which of the modules of the redundancy relationship is active. • MAC address: MAC address of the virtual CARP interface • Preferred: Displays which of the modules of the redundancy relationship is configured as the primary module.

System and status functions	Meaning
Media redundancy (SCALANCE S627-2M only)	• Interface: Interface connected to the MRP ring. • Protocol: Protocol used (MRP) • Ring port 1: Name of the first media module port of the interface connected to the MRP ring. • Ring port 2: Name of the second media module port of the interface connected to the MRP ring. • Domain name: Name of the MRP domain. • Discrepancy: Display whether the domain of the client differs from that of the redundancy manager. • Domain UUID: UUID of the MRP domain.
Media modules (SCALANCE S627-2M only)	• Port: Port ID(s) of the media module port(s) • Name: Name of the media module • MLFB: MLFB of the media module • Revision: Version of the media module • Discrepancy: Displays whether there are differences between the configuration data in STEP 7 and the media modules.

Overview of the Dyn. DNS settings - "Dynamic DNS" entry

Meaning

Table 1-56 Online diagnostics: "Dynamic DNS"

System and status functions	Meaning
Client status	Indicates whether or not a connection was established to a dyn. DNS server.
Current IP address	WAN IP address via which the security module can currently be reached.
Defined IP address	IP address currently assigned to the FQDN.
Current time	The current time-of-day.
Force update	The security module obtains the current IP address of its Internet access and sends an update query to the configured update server(s). This publishes the current IP address on the Internet. The status is displayed in the primary or secondary dynamic DNS service fields. This means, for example that it is possible to check whether configured data such as the user name and password of the dynamic DNS account are correct.
Cancel suspension	Cancels the suspension (blocking of IP address updates from the dynamic DNS provider on the security module), for example after the dynamic DNS provider password has been corrected or an error has been eliminated.
Primary and secondary dynamic DNS service	
FQDN	Fully Qualified Domain Name registered with the provider.
Server IP Address	IP address of the update server used.
Successful update	Successful update in the dynamic DNS update service.
Last update attempt	Last update attempt for DynDNS update service.

System and status functions	Meaning
Last failed update	Last update error in the dynamic DNS update service.
Error code	Error status of the last unsuccessful update attempt during a dynamic DNS update.

What is the meaning of the messages?

The messages of the last DDNS update attempt have the following meaning:

Message	Meaning
Success	
DDNS_OK	The update query was successful.
Connection-related status messages	
DDNS_E_CON_UDP_SRV_RESOLV_ERR	DNS name of the update server unknown, FQDN cannot be resolved using the known DNS server.
DDNS_E_CON_UDP_SRV_UNREACHABLE	Update server unreachable ("Timeout").
Security-related status messages (errors)	
DDNS_E_CERT_SUBJECT_INVALID	The common name of the subject in the certificate does not match the domain name of the update server or its IP address.
DDNS_E_CERT_UNABLE_TO_GET_ISSUER_CERT	Issuer certificate not found. The certificate chain could not be followed back to the root CA because an issuer certificate was not found. The trust chain is incomplete.
DDNS_E_CERT_SIGNATURE_INVALID	The signature of a certificate could not be read or is invalid.
DDNS_E_CERT_NO_TRUST	A certificate in the trust chain is invalid, in other words: • Not yet valid or ready expired • V3 extensions invalid • Critical V3 extension is not supported
DDNS_E_CERT_DEPTH_ZERO_SELFSIGNED_CERT	The update server has supplied a self-signed certificate and the certificate is not in the certificate store for trustworthy root CA certificates.
DDNS_E_CERT_SELF_SIGNED_CERT_IN_CHAIN	The certificate chain could be established using non trustworthy certificates but no suitable root CA certificate was found in the certificate store for trustworthy certificates.
DDNS_E_CERT_CHAIN_TOO_LONG	The certificate chain exceeds the maximum supported verification depth.
DDNS_E_CERT_INVALID_CA	A CA certificate is invalid, in other words expired, not yet valid or the V3 extensions are not suitable for the intended purpose (for example CA not set to TRUE for CA certificates).
DDNS_E_CERT_KEYUSAGE_UNSUITED	The V3 extensions key usage or extended key usage set in a certificate of the trust chain are not suitable for the use of the certificate.
DDNS_E_CERT_EXTENSION_UNSUPPORTED	A certificate in the trust chain used an extension marked as critical that is not supported.

Message	Meaning
Agent-related status messages (errors)	
DDNS_E_AGT_BAD_AGENT	• The update request does not correspond to the structure required by cRSP, for example URL parameters missing. • The update request was sent to an illegal URL on the update server. • The configured update string contains errors.

Display of the ARP table - "ARP table" entry

Meaning

Display of the ARP table of the security module.

Table 1-57 Online diagnostics: "ARP table" tab

System and status functions	Meaning
ARP table	Display of the static (proxy ARP) and dynamic entries of the ARP table on the security module. The "Publication type" tab specifies whether the entry is configured statically or learnt.

Users logged in to the Web page - "Logged in users" entry

Meaning

Shows the users logged in to the Internet page for user-specific IP rule sets.

System and status functions	Meaning
User name	Name of the logged on user.
Source IP address	IP address with which the user logged on.
Time remaining	Remaining time before the user is automatically logged off.
Maximum time of the session	Configured total time of the session.
Log off	The selected user is logged off.

Display of the firewall blacklist - "Firewall blacklist" entry

Meaning

Displays the IP addresses of nodes that have exceeded the permitted number of connections and firewall statuses per time unit. These nodes are entered in the IP blacklist of the firewall.

The number of connections and firewall statuses per time unit is only limited if the "Use extended status options" check box is selected in the "Standard rules for IP services" entry in the local security settings.

If you click the "Delete all" button, the displayed IP addresses are removed from the firewall blacklist of the security module. The IP addresses are also no longer displayed.

Setting the date and time - "Date and time" entry

How to access this function

1. Select the security module whose time and date you want to check or set.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. In online diagnostics, select the "Functions" > "Date and time" entry.

Setting the local time on the security module

In this area, you can read out and set the time and date of the security module. When you click the "Apply" button, the security module is assigned the time and date currently entered in the "Date" and "Time" input boxes.

Setting the local time on PC

This area shows the current time and current date of the PC on which STEP 7 is installed. If you click the "Adopt for module" button, the security module is assigned the current time and current date of the PC.

Diagnostics in ghost mode - "Ghost mode" entry

Module-specific function

This function is available only for SCALANCE S602 as of V3.1.

Meaning

Display of address information as well as information on IP address changes of the internal node.

System and status functions	Meaning
Status of SCALANCE S602	Display of the status of the security module in relation to operation in ghost mode.
IP address	IP address of the internal node (identical to the external IP address of the security module).
Subnet mask	Subnet mask of the security module.
MAC address	MAC address of the internal node.

System and status functions	Meaning
Node found on	Displays when the internal node was recognized by the security module or when an IP address change was made on the internal node.
Number of IP address changes	Number of IP address changes detected by the security module.
IP address	IP address in the external network for which the security module requires route information.
Standard router	Standard router for the IP address in the external network.

Download functions

Downloading a configuration

Downloading the configuration / establishing an online connection

1. From the "PG/PC interface" drop-down list of the "Advanced download" dialog or "Connect online", select the network adapter via which you can reach the module.
2. If the module is in the factory settings status, follow the steps below:
   - From the "Connection to interface/subnet" drop-down list, with which your engineering station is connected and for which the IP address to be assigned is configured in the local security settings.
   - Select the "Display all compatible nodes" from the drop-down list.
   - Click the "Start search" button.
   - Result: The module is displayed in the "Select target device" table with its detected MAC address.
   - Select the module entry in the table and click the "Assign IP address" button.
   - Result: The module is assigned the IP address configured in the local security settings for the selected interface.
3. If the module is not in the factory settings status, follow the steps below:
   - From the "Connection with interface/subnet", select the interface / the FQDN address / WAN address of the module via which your engineering station can reach the module. STEP 7 then uses the address configured in the local security settings for the selected component for access to the module.
   - If you select the option "Display devices with the same address" from the drop down list, the nodes will be displayed whose address matches the selected component. If you select the entry "Show all compatible devices", the devices whose module type matches that of the selected module are shown.
   - Click the "Start search" button.
   - Result: The "Address" column of the "Compatible devices in the target subnet" displays the detected IP address / FQDN address of the module.
   - Select the address entry in the table and click the "Load" or "Go online" button.

Ideally, you should configure the modules of a group via the common external network of these modules (interface X1). If the engineering station is located in an internal network, you will need to enable the IP addresses of the other modules of the group explicitly in the firewall of this SCALANCE S and configure this module first.

Note

Downloading the configuration when operating in ghost mode (only SCALANCE S602 as of V3.1)

When you operate the security module in ghost mode, the external interface of the security module takes over the IP address of the internal node at runtime. Before you can download a new configuration via the external interface to the security module, you need to specify the IP address for downloading a configuration that the security module obtained from the internal node during runtime.
To find out the current IP address of the security module, you can search for reachable nodes in STEP 7 with the menu command "Online" > "Accessible devices".

Note

Loading a configuration on security modules of a redundancy relationship (only SCALANCE S623/S627-2M as of V4)

The properties of a redundancy relationship configured for the primary module must be loaded both on the primary module and on the secondary module. To load the configuration, the physical IP address via which your engineering station can reach the security module must be used. The virtual IP addresses of the redundancy relationship cannot be used for loading.

Specifying a different address

In the "Compatible devices in target subnet" dialog area, you have the option of specifying an IP address / an FQDN address that differs from the IP address / FQDN address in the local security settings. To do this, enter the IP address / FQDN address of the module in the editable cell in the "Address" table column. After loading, the security module is reachable via the IP address / the FQDN address from the local security settings.

Firmware version

The configuration of a SCALANCE S module can also be downloaded to a SCALANCE S module whose firmware version is higher than the firmware version of the SCALANCE S module in STEP 7.

Operating mode

Configurations can be downloaded while the SCALANCE S modules are operating. Restart the SCALANCE S module to activate your configuration changes.

Note

Special characteristics

- As long as a module has not yet set IP parameters (in other words, prior to the first configuration), there must be no router between the module and the configuration computer.
- If you swap a PC from the internal to the external interface of the SCALANCE S, access from this PC to the SCALANCE S is blocked for approximately 20 minutes.

Configuration status

Prior to each download, the existing configuration on the security module is checked and compared with the configuration to be downloaded from the STEP 7 project. If the configuration of the module originates from the STEP 7 project currently to be downloaded and there are differences between these configurations, it is possible to download only files with differences between the module and project configuration to the security module. In some situations, this can speed up the download.

Transferring firmware

This needs to be taken into consideration before transferring new firmware

To transfer new firmware to a security module, the following conditions must be met:

- You have the rights required to transfer firmware; refer to section Special features of user management for security functions (Page 360).
- The security module is configured with an IP address.

The transfer is secure

The firmware is transferred over a secure connection and can therefore also be transferred from the unprotected network.

The firmware itself is signed and encrypted. This ensures that only authentic firmware can be downloaded to the SCALANCE S module.

Restart necessary following transfer

Newly downloaded firmware only becomes active after the SCALANCE S module has been restarted. If the transfer is disturbed and aborted, the module starts up again with the old firmware version.

Security for S7-300 /S7-400 / PC CPs

Setting up a firewall

Local firewall rules for S7-300 /S7-400 / PC CPs

Overview S7-300 CPs / S7-400 CPs /PC CPs

Enabling packet filter rules

If you enable the security function for the CPs in the local security settings, initially all access to and via the CP is permitted. To enable individual packet filter rules, select the "Activate firewall" check box. Then enable the required services. Firewall rules created automatically due to a connection configuration have priority over rules set manually.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Activate firewall in advanced mode" check box.

Firewall configuration with VPN

If the security module is added to a VPN group, the firewall is enabled by default. In addition, the "Tunnel communication only" check box is enabled. This means that only encrypted IPsec data transfer is permitted via the external interface. External data traffic is blocked.

If you deselect the check box, tunneled communication and also the types of communication selected in the other boxes are permitted.

Updating connection rules

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button. The modified firewall rules are then displayed in advanced firewall mode.

Configuring a firewall with predefined firewall rules - CP 343-1 Adv. / 443-1 Adv.

Configuring a firewall with predefined IP rules - CP 343-1 Adv. / 443-1 Adv.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Firewall" > "Predefined IP rules".

Table 1-58 Available services and directions

Service	From station/ internal to external	External to internal	From external to station	Enabled ports	Meaning
Allow IP communication	x	x	x	All	IP traffic for the selected communication directions is allowed.
Allow S7 protocol	x	x	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/ FTPS (explicit mode)	x	x	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	x	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	x	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow DNS	x	x	-	TCP port 53 UDP port 53	Communication connection to a DNS server is allowed.
Allow SNMP	x	x	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	x	-	TCP port 25	For sending e-mails via an SMTP server.
Allow NTP	x	x	-	UDP port 123	For synchronization of the time of day.

Table 1-59 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Accept	Station	Tunnel
		Accept	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules - CP 343-1 Adv. / 443-1 Adv.

How to access this function

1. Select the module to be edited.
2. Select the entry "Firewall" > "Predefined MAC rules".

Table 1-60 Available services and directions

Service	From station to external	From external to station	Meaning
Allow MAC communication	x	x	The MAC traffic from station to external and vice versa is allowed.
Allow ISO protocol	x	x	The ISO traffic from station to external and vice versa is allowed.

Table 1-61 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.	Accept	Station	Tunnel
		Accept	Tunnel	Station
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined firewall rules - CP 1628

Configuring a firewall with predefined IP rules - CP 1628

How to access this function

1. Select the module to be edited.
2. Select the "Security" > "Firewall" > "Predefined IP rules" entry.

Table 1-62 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/FTPS (explicit mode)	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow DNS	x	TCP port 53 UDP port 53	Communication connection to a DNS server is allowed.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	TCP port 25	For sending e-mails via an SMTP server.
Allow NTP	x	UDP port 123	For synchronization of the time of day.

Table 1-63 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Accept	Station	Tunnel
		Accept	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules - CP 1628

How to access this function

1. Select the module to be edited.
2. Select the entry "Security" > "Firewall" > "MAC rules".

Table 1-64 Available services and directions

Service	From station to external	From external to station	Meaning
Allow MAC level communication	x	x	The MAC traffic from external to the station and vice versa is allowed.
Allow ISO communication	x	x	ISO traffic from external to the station and vice versa is allowed.
Allow SiClock	x	x	SiClock time-of-day frames from external to the station and vice versa are allowed.

Table 1-65 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.	Accept	Station	Tunnel
		Accept	Tunnel	Station
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

IP packet filter directions S7-300-/S7-400-/PC-CPs

Meaning

Possible selections for the communication directions "From" and "To" in the IP rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		Meaning
From	To	CP 343-1 Adv. / 443-1 Adv.	CP 1628	
Internal	Station	x	-	Access from the internal network to the station.
	Any	x	-	Access from internal to the external network, VPN tunnel partner and the station.
External	Station	x	x	Access from the external network to the station.
	Any	x	-	Access from external to the internal network and the station.
Station	Internal	x	-	Access from the station to the internal network.
	External	x	x	Access from the station to the external network.
	Tunnel	x	x	Access from the station to the VPN tunnel partner.
Tunnel	Station	x	x	Access via the VPN tunnel partner to the station.
	Any	x	-	Access from VPN tunnel partners to the internal network and the station.
Any	External	x	-	Access from the internal network and the station to the external network.

MAC packet filter directions S7-300-/S7-400-/PC-CPs

Meaning

Possible selections for the communication directions "From" and "To" in the MAC rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		Meaning
From	To	CP 343-1 Adv. / 443-1 Adv.	CP 1628	
External	Station	x	x	Access from the external network to the station.
Station	External	x	x	Access from the station to the external network.
	Tunnel	x	x	Access from the station to the VPN tunnel partner.
Tunnel	Station	x	x	Access via the VPN tunnel partner to the station.

Configuring the access list

Module-specific function

This function is not available for the CP 1628.

Meaning

You set access protection for certain IP addresses using the IP access lists. List entries that have already been created and the corresponding rights are displayed in the local settings of the CP in the entry "Firewall" > "IP rules" (advanced firewall mode).

Note

Changed behavior after activation of security

- Once you have activated the security function for a CP, access protection will only apply to the external interface. You can apply access protection to the internal interface as well, by configuring suitable firewall rules in advanced firewall mode.
- The CP also responds to ARP requests from IP addresses that have not been released (layer 2).
- If the IP access list of a CP contains no entries and you activate security for the CP, the firewall will be activated and prevent access to the CP from external locations. Configure the corresponding firewall rules in advanced firewall mode so that the CP can be reached.

Effect of IP access list entries at activation of security

If security is enabled in the local settings of a CP, the corresponding rules are created in the advanced firewall mode. A firewall rule "Accept" > "External" > "Station" is created for an IP address you specified in the address list. The IP address from the IP access list is used accordingly as source IP address. IP addresses from a defined IP address range are also integrated into corresponding firewall rules.

Requirements for editing

Before you can edit created firewall rules, the following condition must be met:

- for editing using STEP 7: "Configure security" configuration right
- for editing using a Web server: Module right "Web: Expand IP access control list"

The requirements for editing the IP access lists outside the local security settings are described in the sections on the specific CPs.

Connection-related automatic firewall rules

Meaning

For connections that were configured using CPs, STEP 7 automatically creates firewall rules that allow communication with the partner of the CP in the specified direction (CP active/ passive). The connection establishment directions are taken into account. To display these firewall rules, if the advanced firewall mode is enabled, the "Update connection rules" button needs to be clicked. The firewall rules are then displayed in advanced firewall mode.

Note

Releasing UDP multicast connections manually

No automatic firewall rules are created for UDP multicast connections. To enable the connections, add the relevant firewall rules manually in advanced firewall mode.

Depending on how the connection establishment is configured, the following level 3 firewall rules are created. If the security module is in a VPN group, the direction "External" changes to "Tunnel". This applies only to CPs that support VPN.

The IP address of the connection partner is entered in the "Source IP address" or "Destination IP address" column of these firewall rules.

CP->external	Action	From	To
active	Accept	Station	External
	Drop	External	Station
passive	Drop	Station	External
	Accept	External	Station
active and passive	Accept	External	Station
	Accept	Station	External

CP->internal	Action	From	To
active	Accept	Station	Internal
	Drop	Internal	Station
passive	Drop	Station	Internal
	Accept	Internal	Station
active and passive	Accept	Internal	Station
	Accept	Station	Internal

For level 2 connections, "Accept" rules are created for both directions. If the security module is in a VPN group, the direction "External" changes to "Tunnel".

The MAC address of the connection partner is entered in the "Source MAC address" or "Destination MAC address" column of these firewall rules.

CP->external	Action	From	To
active, passive, active and passive	Accept	Station	External
	Accept	External	Station

Note

Changing the connection configuration

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button.

Conventions for automatically created firewall rules

- Priority
  The rules have highest priority and are therefore inserted at the top in the local rule list.
- Deleting rules
  The rules cannot be deleted. Logging can be enabled and services can be assigned. You can also insert a transmission speed and a comment.
- Changing the action
  If you set the action from "Accept" to "Drop" or vice versa, this is overwritten again during renewed system synchronization. If you want to retain your changes, select or "Accept*" or "Drop*" as action. In this case, only the IP address is synchronized and the action and direction remain as set. Settings for logging, service, transmission speed and comment are also retained after a renewed system synchronization even without changing the action to "Accept*" or "Drop*". If the configured connection is deleted, the corresponding rules are removed from the list.

Security module in VPN group

As default, the "Tunnel communication only" check box is enabled. If you deselect the check box, in addition to tunnel communication between tunnel partners, communication is also possible with network nodes to which there is no tunnel.

- Communication is untunneled if the partner address belongs to a station known in STEP 7 for which no VPN tunnel is configured.
- Communication is through the tunnel if the partner address is a VPN endpoint.
- If it is not clear whether connection should bypass or run through the VPN tunnel, the connection is assigned to the VPN tunnel and a message to this effect is displayed. The assignment can be adapted in advanced firewall mode, for example, by changing the "From" direction "Tunnel" to "External". To avoid this change being overwritten by the next system synchronization, the "Accept*" or "Drop*" action must be selected.

Note

If you want to ensure that only communication through the tunnel is possible, you will need to create suitable firewall rules in advanced firewall mode, for example, for internal nodes or NDIS addresses.

To allow only tunneled communication for a CP, add a rule with the following settings:

- "Action": "Drop"
- "From": "Any"
- "To": "External"

For the CP 1628, add a rule with the following settings:

- "Action": "Drop"
- "From": "Station"
- "To": "External"

In addition to this, you need to remove existing firewall rules that allow untunneled communication.

Activating Web server on security module

Module-specific function

This function is only available for CP 343-1 Adv. / 443-1 Adv. and CP 443-1 OPC UA:

Meaning

After activating the Web server, you have access to the Web pages of the module. In the local security settings, you can restrict access to these Web pages for the CP 343-1 Adv. / 443-1 Adv.to the HTTPS protocol. This access is controlled using the "Allow access only using HTTPS" check box. In addition, you must configure the firewall accordingly. For the CP 443-1 OPC UA access to the Web server is only permitted using HTTPS.

IPsec tunnel: Creating and assigning groups

Configuring internal network nodes - "Nodes" entry

Allow access to S7-300 / S7-400 CPs for VPN connection partners

Possible selections

Decide whether or not the VPN connection partner can have access to the CP and/or the internal subnet of the CP in routing mode (SCALANCE S / M).

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Node".
3. Configure access for the VPN connection partner of the CP in routing mode (SCALANCE S / M):
   – Allow connection to the CP (Gbit interface)
   – Allow connection to the internal subnet (PROFINET subnet)

Configuring NDIS nodes manually for PC CPs that can be reached through the tunnel

Configuring NDIS nodes that can be reached through the tunnel

The internal nodes are learned and assigned to the routes dynamically. This concerns the NDIS IP addresses of the Windows PC.

Follow the steps below

1. Select the module to be edited.
2. In the local security settings, select the entry "Nodes" > "NDIS nodes reachable via tunnel".
3. Enter the NDIS IP addresses.

Online functions - Debug / Diagnostics and Logging

Updated firewall rules - "Dynamically updated firewall rules" entry

Module-specific function

This function is only available for CP 343-1 Adv. / 443-1 Adv.

Meaning

Display of the IP addresses or IP address ranges that were enabled dynamically using HTTP or HTTPS, or loaded by a user. The rights assigned for accessing the S7 CP are displayed for the enabled IP addresses. An update of the IP addresses in this tab can only be triggered by the following events:

- Extension/modification of the IP access control list
- Update of firewall rules
- Dynamic extensions transmitted to the CP at runtime, for example, PROFINET IO devices

Since only the dynamically updated firewall rules are displayed here, you also need to take into account the firewall rules that were configured offline and downloaded to the station for a full picture of the current firewall status of the module.

Security for S7-1200-/S7-1500-CPs

Setting up a firewall

Local firewall rules for S7-1200 / S7-1500 CPs

Overview of the local firewall rules for S7-1200 / S7-1500 CPs

Enabling packet filter rules

If you enable the security function for the CPs in the local security settings, initially all access to and via the CP is permitted. To enable individual packet filter rules, select the "Activate firewall" check box. Then enable the required services. Firewall rules created automatically due to a connection configuration have priority over rules set manually.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Activate firewall in advanced mode" check box.

Updating connection rules

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button. The modified firewall rules are then displayed in advanced firewall mode.

Configuring a firewall with predefined firewall rules - CP 1543-1 / CP 1543SP-1

Configuring a firewall using predefined IP rules

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Firewall" > "Predefined IP rules".

Table 1-66 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/FTPS* (explicit mode)	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow security diagnostics	x	TCP port 8448	Allow security diagnostics.

*not available for CP 1543SP-1

Table 1-67 IP routing via backplane bus (only for the CP 1543-1 as of V2.1)

Service	Enabled ports	Meaning
Allow IP communication via backplane bus	All	IPv4 traffic is allowed via the backplane bus. If the option "IP routing between communications modules" is activated group, this check box is enabled and cannot be deselected. If the option is deactvated, this check box is deactivated and cannot be activated.

Table 1-68 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Accept	Station	Tunnel
		Accept	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall using predefined IPv6 rules

Meaning

With the predefined IPv6 rules, you have the option of configuring the firewall for services in which IPv6 is used. By enabling a predefined IPv6 rule in the local security settings of the CP 1543-1 V1.1, the system-defined ICMPv6 services that can be seen in the global security functions in the "ICMP" tab in "Firewall" > "Services" > "Define services for IP rules" are also enabled in the firewall. The firewall of the CP 1543-1 V1.0 allows ICMPv6 packets through even without enabling a predefined IPv6 rule.

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" > "Pre-defined IPv6 rules" item in the local security settings.

Table 1-69 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/FTPS* (explicit mode)	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.

*not available for CP 1543SP-1

Table 1-70 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings made in "Predefined IPv6 rules" have no effect on firewall rules that were automatically generated as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules

How to access this function

1. Select the module to be edited.
2. Select the entry "Firewall" > "Predefined MAC rules".

Table 1-71 Available services and directions

Service	From station to external	From external to station	Enabled ports	Meaning
Allow MAC communication	x	x	-	The MAC traffic from external to the station and vice versa is allowed.
Allow ISO protocol*	x	x	-	ISO traffic from external to the station and vice versa is allowed.
Allow DCP	x	x	-	DCP traffic from external to the station and vice versa is allowed.
Allow LLDP*	x	x	-	LLDP traffic from external to the station and vice versa is allowed.

*not available for CP 1543SP-1

Table 1-72 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.	Accept	Station	Tunnel
		Accept	Tunnel	Station

Option	Action when activated	Relevant firewall rule		
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined firewall rules - S7-1200 CPs

Configuring a firewall using predefined IP rules

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Firewall" > "Predefined IP rules".

Table 1-73 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow HTTP	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow security diagnostics	x	TCP port 8448	Allow security diagnostics.

Table 1-74 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Accept	Station	Tunnel
		Accept	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall using predefined IPv6 rules

Meaning

With the predefined IPv6 rules, you have the option of configuring the firewall for services in which IPv6 is used. By enabling a predefined IPv6 rule in the local security settings of an S7-1200 CP capable of IPv6, the system-defined ICMPv6 services that can be seen in the global security functions in the "ICMP" tab in "Firewall" > "Services" > "Define services for IP rules" are also enabled in the firewall.

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" > "Predefined IPv6 rules" item in the local security settings.

Table 1-75 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.

Table 1-76 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings made in "Predefined IPv6 rules" have no effect on firewall rules that were automatically generated as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules

How to access this function

1. Select the module to be edited.
2. Select the entry "Firewall" > "Predefined MAC rules".

Table 1-77 Available services and directions

Service	From station to external	From external to station	Enabled ports	Meaning
Allow MAC communication	x	x	-	The MAC traffic from external to the station and vice versa is allowed.
Allow DCP	x	x	-	DCP traffic from external to the station and vice versa is allowed.

Table 1-78 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.	Accept	Station	Tunnel
		Accept	Tunnel	Station

Option	Action when activated	Relevant firewall rule		
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

IP packet filter directions S7-1200 / S7-1500 CPs

Meaning

Possible selections for the communication directions "From" and "To" in the IP rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Meaning
From	To	
External	Station	Access from the external network to the station.
	Backplane bus*	Access from the external network to the backplane bus of the station.
	Any**	Access from the external for all supported directions.
Station	External	Access from the station to the external network.
	Tunnel	Access from the station to the VPN tunnel partner.
	Backplane bus*	Access from the station to its backplane bus.
	Any**	Access from the station to all supported directions.
Tunnel	Station	Access by VPN tunnel partners to the station.
	Backplane bus*	Access from VPN tunnel partners to the backplane bus of the station.
	Any**	Access from NPN tunnel partners to all supported directions.

Available options / ranges of values		Meaning
Backplane bus*	External	Access from backplane bus of the station to the external network.
	Tunnel	Access from backplane bus of the station to NPN tunnel partners.
	Station	Access from the backplane bus to the station.
	Any**	Access from the backplane bus to all supported directions.
Any**	External	Access from all supported directions to the external network.
	Tunnel	Access from all supported directions to VPN tunnel partners.
	Station	Access from all supported directions to the station.
	Backplane bus*	Access from all supported directions to the backplane bus of the station.

* only for CP 1543-1 as of V2.1 with IPv4 communication. if the option "IP routing between communications modules" is activated, IP routing via the backplane bus an be allowed / blocked via this direction.

* Only for CP 1543-1 as of V2.1

MAC packet filter directions S7-1200 / S7-1500 CPs

Meaning

Possible selections for the communication directions "From" and "To" in the MAC rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Meaning
From	To	
External	Station	Access from the external network to the station.
Station	External	Access from the station to the external network.
	Tunnel	Access from the station to the VPN tunnel partner.
Tunnel	Station	Access by VPN tunnel partners to the station.

Connection-related automatic firewall rules

Meaning

For connections that were configured using CPs, STEP 7 automatically creates firewall rules that allow communication with the partner of the CP in the specified direction (CP active/passive). The connection establishment directions are taken into account. To display these firewall rules, if the advanced firewall mode is enabled, the "Update connection rules" button needs to be clicked. The firewall rules are then displayed in advanced firewall mode.

Note

Enabling UDP multicast and UDP broadcast connections manually

No automatic firewall rules are created for UDP multicast and UDP broadcast connections. To enable the connections, add the relevant firewall rules manually in advanced firewall mode.

Depending on how the connection establishment is configured, the following level 3 firewall rules are created. If the CP is in a VPN group, the direction "External" changes to "Tunnel".

The IP address of the connection partner is entered in the "Source IP address" or "Destination IP address" column of these firewall rules.

CP->external	Action	From	To
active	Drop	External	Station
	Accept	Station	External
passive	Drop	Station	External
	Accept	External	Station
active and passive	Accept	External	Station
	Accept	Station	External

For level 2 connections, "Accept" rules are created for both directions. If the CP is in a VPN group, the direction "External" changes to "Tunnel".

The MAC address of the connection partner is entered in the "Source MAC address" or "Destination MAC address" column of these firewall rules.

CP->external	Action	From	To
active, passive, active and passive	Accept	Station	External
	Accept	External	Station

Note

Changing the connection configuration

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button.

Conventions for automatically created firewall rules

- Priority
  The rules have highest priority and are therefore inserted at the top in the local rule list.
- Deleting rules
  The rules cannot be deleted. Logging can be enabled and services can be assigned. You can also insert a transmission speed and a comment.
- Changing the action
  If you set the action from "Accept" to "Drop" or vice versa, this is overwritten again during renewed system synchronization. If you want to retain your changes, select or "Accept*" or "Drop*" as action. In this case, only the IP address is synchronized and the action and direction remain as set. Settings for logging, service, transmission speed and comment are also retained after a renewed system synchronization even without changing the action to "Accept*" or "Drop*". If the configured connection is deleted, the corresponding rules are removed from the list.

Security module in VPN group

As default, the "Tunnel communication only" check box is enabled. If you deselect the check box, in addition to tunnel communication between tunnel partners, communication is also possible with devices to which there is no tunnel.

- Communication is untunneled if the partner address belongs to a station known in STEP 7 for which no VPN tunnel is configured.
- Communication is through the tunnel if the partner address is a VPN endpoint.

Note

If you want to ensure that only communication through the tunnel is possible, you will need to create suitable firewall rules in advanced firewall mode.

To allow only tunneled communication for a CP, add a rule with the following settings:

- "Action": "Drop"
- "From": "Station"
- "To": "External"

In addition to this, you need to remove existing firewall rules that allow untunneled communication.

Configuring subnets reachable via the backplane bus for CP 1543-1

Requirement

- The security module CP 1543-1 as of V2.1 is a member of a VPN group and the option "IP routing between communications modules" is activated

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN" > "Nodes" " Subnets reachable through tunnel".

Configuring subnets reachable via the backplane bus

VPN communication with network nodes and subnets reachable via the backplane bus of a station is only possible if this is released for VPN communication. In the entry "Subnets reachable through tunnel" a maximum of 16 addresses or subnets can be specified. IP addresses must be specified with a 32 bit subnet mask, subnets with a permitted subnet mask in the range 1... 31 bits. Broadcast addresses must not be specified.

Automatic OpenVPN configuration via SINEMA Remote Connect Server:

Automatic OpenVPN configuration and OpenNPN tunnel establishment

Supported modules

The function is available for several security modules and telecontrol communications modules.

Function

The module can establish an OpenVPN connection to the application "SINEMA Remote Connect Server". The OpenVPN connection parameters used for this are specified by the SINEMA Remote Connect Server. The module can call up these connection parameters cyclically and therefore automatically obtains the OpenVPN configuration.

Procedure

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN".
3. Select the check box "Activate VPN" and select the VPN connection type "Automatic OpenVPN configuration via SINEMA Remote Connect Server".
4. Set the following parameters for the OpenVPN connection:

Parameter	Meaning
SINEMA RC address	IPv4 / IPv6 address, FQDN or PROFINET device name of the SINEMA Remote Connect server from which the module calls up the connection parameters for the OpenVPN connection.
SINEMA RC port	Port for the connection to the SINEMA Remote Connect server that is used to call up the connection parameters for the OpenVPN connection.

Parameter	Meaning
CA Certificate	The module authenticates itself with the SINEMA Remote Connect server with a CA certificate that was downloaded from the server and selected for the module. Only certificates can be selected that were assigned to the module via its local certificate manager. For the authentication of the SINEMA Remote Connect server, the module checks the CA certificate of the server. At the same time a check is made to establish whether the current time of day on the module is in the validity period of the certificate.
Fingerprint	The module authenticates itself with the SINEMA Remote Connect server with the "Fingerprint" property of the CA certificate of the SINEMA Remote Connect server.
Device ID	The device ID is assigned by SINEMA Remote Connect server for every module that is configured on the SINEMA Remote Connect server. The device ID serves to identify a module.
Device password	Password configured for the device ID on the SINEMA Remote Connect server.
Update interval	Interval in minutes at which the module calls up the OpenVPN connection parameters to be used from the SINEMA Remote Connect server. Whether and with which connection parameters OpenVPN connections can be established is specified by the SINEMA Remote Connect server. With the value "0" the updating is disabled. Minimum value for updating: 10 minutes Maximum value for updating: 10080 minutes
Connection type	The connection type specifies when the module establishes an OpenVPN connection to the SINEMA Remote Connect Server: • Permanent: The module establishes an OpenVPN connection after reading the OpenVPN connection parameters. The OpenVPN connection is retained until the connection parameters are changed by the SINEMA Remote Connect Server. • PLC trigger: The module establishes an OpenVPN connection as soon as the selected PLC tag has the value "TRUE".
PLC tag for connection establishment	Selection of a tag of the station of the type "BOOL". When the tag has the value "TRUE" the module establishes an OpenVPN connection to the SINEMA Remote Connect server: If the tag has the value "FALSE" the OpenVPN connection is terminated.

Diagnostics of the automatic OpenVPN configuration and OpenNPN connections

Meaning

The diagnostics page shows the status of the automatic OpenVPN configuration and the OpenVPN connections.

The status of the automatic OpenVPN configuration indicates whether the module was able to call up the configuration file with the OpenVPN connection parameters to be used from the SINEMA Remote Connect Server.

For OpenVPN connections to the SINEMA Remote Connect Server, the connection status and its causes are displayed.

1.3.5.2 Configuring security for SCALANCE S-600

Supported devices

Supported devices

The security functions described in this help section are supported y the following products:

- SCALANCE S615 V5
- SCALANCE SC-600:
  - SCALANCE SC632-2C V1
  - SCALANCE SC636-2C V1
  - SCALANCE SC642-2C V1
  - SCALANCE SC646-2C V1

General terminology "device"

In this section of the information system the term "device" is used instead of the product names listed above. You will finf the functional differences in the section Hardware equipment and system functions (Page 550) Depending on the selected device, different parameters are available on the individual configuration pages.

Hardware equipment and system functions

Availability of hardware

The following table shows the hardware of the devices.

We reserve the right to make technical changes.

	SCALANCE S615	SCALANCE SC632-2C SCALANCE SC642-2C	SCALANCE SC636-2C SCALANCE SC646-2C
Total usable ports	5 RJ-45 ports	2 RJ-45 ports	6 RJ-45 ports
SFP transceiver slots	-	2	2
Electrical connectors	5	2	6
Combo ports	-	2	2
PLUG slot	1	1	1

Availability of the system functions

The following table shows the availability of the system functions on the devices.

We reserve the right to make technical changes.

		SCALANCE S615	SCALANCE SC-600
Information	Versions	✓	✓
	I&M	✓	-
	ARP table	✓	✓
	Log tables	✓	✓
	Error	✓	✓
	SNMP	✓	✓
	LLDP	✓	✓
	Routing table	✓	✓
	IPsec VPN	✓	✓ (only SCALANCE SC64x-2C)
	SINEMA RC	✓ (only with KEY-PLUG)	✓
	OpenVPN Client	✓	-
	Redundancy	✓	-
	Security	✓	✓
System	Configuration	✓	✓ (no Telnet server, no SMTP client)
	General	✓	✓
	DNS	✓	✓
	TFTP	✓	✓
	SFTP	✓	-
	Events	✓	✓
	SMTP client	✓	-
	PLUG	✓	-
	DHCP client	✓	✓ (only IPv4)
	DHCP server	✓	✓
	SNMP	✓	✓
	System time	✓	✓
	Auto logout	✓	✓
	Button	✓ (no fault mask)	✓
	Syslog client	✓	✓
	Ports	✓ (under "Interfa-ces")	✓
	Fault Monitoring	✓ (only Link Change)	✓
	Port diagnostics (only availa-ble online)	-	✓
	cRSP / SRS	✓	✓
	Proxy Server	✓	✓
	SINEMA RC	✓ (only with KEY-PLUG)	✓

		SCALANCE S615	SCALANCE SC-600
Interfaces	Ethernet	✓	-
	PPP	✓	-
Layer 2	Port Based VLAN	✓	✓
	Dynamic MAC aging	✓	✓
	LLDP	✓	✓
	Spanning Tree	✓	-
Layer 3	Static routes	✓	✓
	Subnets	✓	✓
	NAT	✓	✓
Security	User	✓ (only in the WBM)	✓
	AAA (Authentication, Authorization, Accounting)	✓	-
	Certificates	✓	✓
	Firewall	✓	✓
	IPsec VPN	✓	✓ (only SCALANCE SC64x-2C)
	OpenVPN	✓	-

Legal notice

Qualified personnel

The product/system belonging to this documentation may only be handled by qualified personnel for the intended purpose taking into account the documentation relating to the intended purpose and, in particular, the safety and warning notices it contains. Due to training and experience, **qualified personnel** is capable of recognizing risks and avoiding possible dangers when handling these products/systems.

Useful information

IPv4 address, subnet mask and address of the gateway

Range of values for IPv4 address

The IPv4 address consists of four decimal numbers with the range from 0 to 255, each number separated by a period; example: 141.80.0.16

IPv4 address format - notation

An IPv4 address consists of 4 bytes. Each byte is represented in decimal, with a dot separating it from the previous one.

XXX.XXX.XXX.XXX

XXX stands for a number between 0 and 255

The IPv4 address consists of two parts:

- The address of the (sub) network
- The address of the node (generally also called end node, host or network node)

Range of values for subnet mask

The subnet mask consists of four decimal numbers with the range from 0 to 255, each number separated by a period; example: 255.255.0.0

The binary representation of the 4 subnet mask decimal numbers must contain a series of consecutive 1s from the left and a series of consecutive 0s from the right.

The 1s specify the network number within the IPv4 address. The 0s specify the host address within the IPv4 address.

Example:

Correct values:

255.255.0.0 D = 1111 1111.1111 1111.0000 0000.0000 0000 B

255.255.128.0 D = 1111 1111.1111 1111.1000 0000.0000 0000 B

255.254.0.0 D = 1111 1111.1111 1110.0000 0000.0000.0000 B

Incorrect value:

255.255.1.0 D = 1111 1111.1111 1111.0000 0001.0000 0000 B

Relationship between the IPv4 address and subnet mask

The first decimal number of the IPv4 address (from the left) determines the structure of the subnet mask with regard to the number of "1" values (binary) as follows (where "x" is the host address):

First decimal number of the IPv4 address	Subnet mask
0 to 127	255.x.x.x
128 to 191	255.255.x.x
192 to 223	255.255.255.x

Classless Inter-Domain Routing (CIDR)

CIDR is a method that groups several IPv4 addresses into an address range by representing an IPv4 address combined with its subnet mask. To do this, a suffix is appended to the IPv4 address that specifies the number of bits of the network mask set to 1. Using the CIDR notation, routing tables can be reduced in size and the available address ranges put to better use.

Example:

IPv4 address 192.168.0.0 with subnet mask 255.255.255.0

The network part of the address covers 3 x 8 bits in binary representation; in other words 24 bits.

This results in the CIDR notation 192.168.0.0/24.
The host part covers 1 x 8 bits in binary notation. This results in an address range of 2 to the power 8, in other words 256 possible addresses.

Value range for gateway address

The address consists of four decimal numbers taken from the range 0 to 255, each number being separated by a period; example: 141.80.0.1

Relationship between IPv4 address and gateway address

The only positions of the IPv4 address and gateway address that may differ are those in which "0" appears in the subnet mask.

Example:

You have entered the following: 255.255.255.0 for the subnet mask; 141.30.0.5 for the IPv4 address and 141.30.128.0 for the gateway address. The Ipv4 address and gateway address may only be different in the 4th decimal number. In the example, however, the 3rd position is different.

You must, therefore, change one of the following in the example:

The subnet mask to: 255.255.0.0 or

the IPv4 address to: 141.30.128.1 or

the gateway address to: 141.30.0.1

ICMP

The acronym ICMP stands for Internet Control Message Protocol (RFC792) and is used to exchange error and information messages.

- Error message
  Informs the sender of the IP frame that when forwarding the frame an error or a parameter problem occurred.
- Information message
  Can contain information about the time measurement, the address mask, the reachability of the destination or for finding the router.

Structure of the ICMP data packet

0	4	8	12	16	20	24	28	31
ICMP packet type Type of message		Code Further details of the message		Checksum				
Data (optional)								

- **ICMP packet type**
  The most important ICMP packet types are as follows:
  - Redirect
    The router informs the host in one of its subnets that there is a better route to the destination. This ICMP packet type is dealt with in more detail in the following description.
  - Destination Unreachable
    IP frame cannot be delivered.
  - Time Exceeded
    Time limit exceeded
  - Echo-Request
    Echo request, better known as ping.
- **Code**
  The code describes the ICMP packet type in greater detail. The selection depends on the selected ICMP packet type. With "Destination Unreachable,", for example "Code 1" host cannot be reached.

You will find a full list of the ICMP packet types and codes on the website of IANA (http://www.iana.org/assignments/icmp-parameters).

ICMP packet type 5 - Redirect

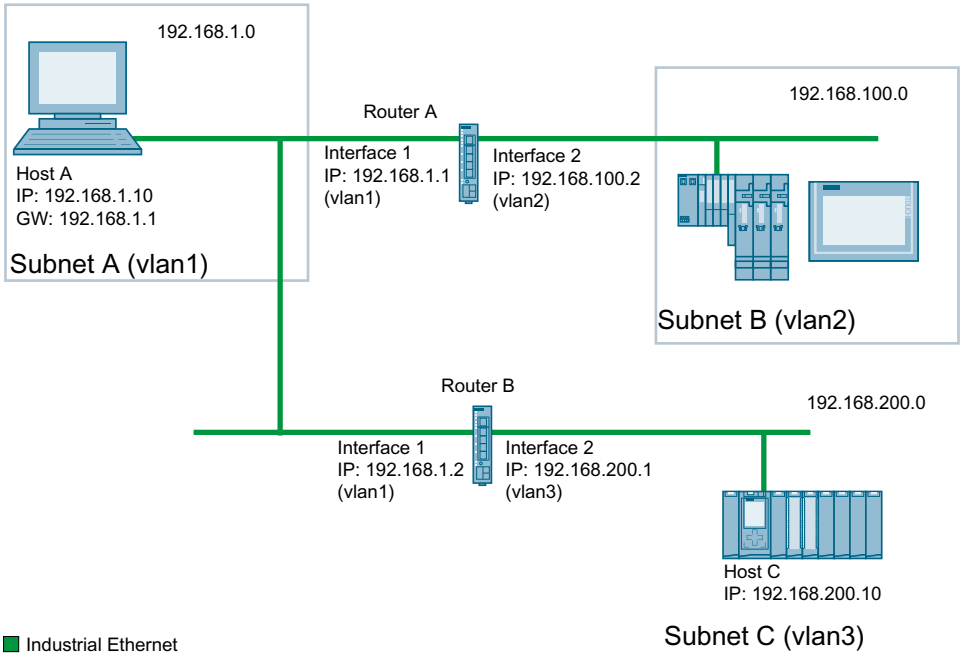

Host A wants to send an IP frame to host C. Host C is not located in the same subnet as host A. For this reason host A sends the IP frame to its default gateway. The default gateway of host A is interface 1 of router A. Router A cannot forward the IP frame because it does not know the destination network. Via its routing table, however, router A knows that subnet C is reachable via router B. Router B connects subnet A with subnet C. Router A sends a redirect message to host A. In this, router A instructs host A in future to send IP frames to host C via

router B whose IP address is contained in the redirect message. The initial IP frame is sent by router A directly to router B that forwards it to Host C.

Conditions for sending redirect messages

- The IP frame is received and sent via the same interface of router A.
- The source IP address (host A) is from the same subnet as the next hop address (router B) in the routing table.
- The IP frame is not affected by a source NAT rule (masquerading, source NAT or NETMAP).
- So that router A forwards the initial IP frame to router B, a firewall rule vlanX → vlanX is required.

VLAN

VLAN

Network definition regardless of the spatial location of the nodes

VLAN (Virtual Local Area Network) divides a physical network into several logical networks that are shielded from each other. Here, devices are grouped together to form logical groups. Only nodes of the same VLAN can address each other. Since multicast and broadcast frames are only forwarded within the particular VLAN, they are also known as broadcast domains.

The particular advantage of VLANs is the reduced network load for the nodes and network segments of other VLANs.

To identify which packet belongs to which VLAN, the frame is expanded by 4 bytes, refer to VLAN tagging (Page 558). This expansion includes not only the VLAN ID but also priority information.

Options for the VLAN assignment

There are various options for the assignment to VLANs:

- Port-based VLAN
  Each port of a device is assigned a VLAN ID. You configure port-based VLAN in "Layer 2 > VLAN > Port-based VLAN (Page 657)".

VLAN assignment on the device

VLAN assignment on the device

In the factory settings, the following assignments are made:

Table 1-79 SCALANCE S615

P1 to P4	vlan1 For access from the local network (LAN) to the device
P5	vlan2 For access from the external network (WAN) to the device

Table 1-80 SCALANCE SC6x2-2C

P0.1	vlan1 For access from the local network (LAN) to the device
P0.2	vlan2 For access from the local network (LAN) to the device

Table 1-81 SCALANCE SC6x6-2C

P0.1 - P0.4	vlan1 For access from the local network (LAN) to the device
P0.5 - P0.6	vlan2 For access from the local network (LAN) to the device

You can change the assignment in "Layer 2 > VLAN > General (Page 654)".

The VLANs are in different IP subnets. To allow these to communicate with each other, the route and firewall rule must be configured on the device.

VLAN tagging

Expansion of the Ethernet frames by four bytes

For CoS (Class of Service, frame priority) and VLAN (virtual network), the IEEE 802.1Q standard defined the expansion of Ethernet frames by adding the VLAN tag.

Note

The VLAN tag increases the permitted total length of the frame from 1518 to 1522 bytes. The end nodes on the networks must be checked to find out whether they can process this length / this frame type. If this is not the case, only frames of the standard length may be sent to these nodes.

The additional 4 bytes are located in the header of the Ethernet frame between the source address and the Ethernet type / length field:

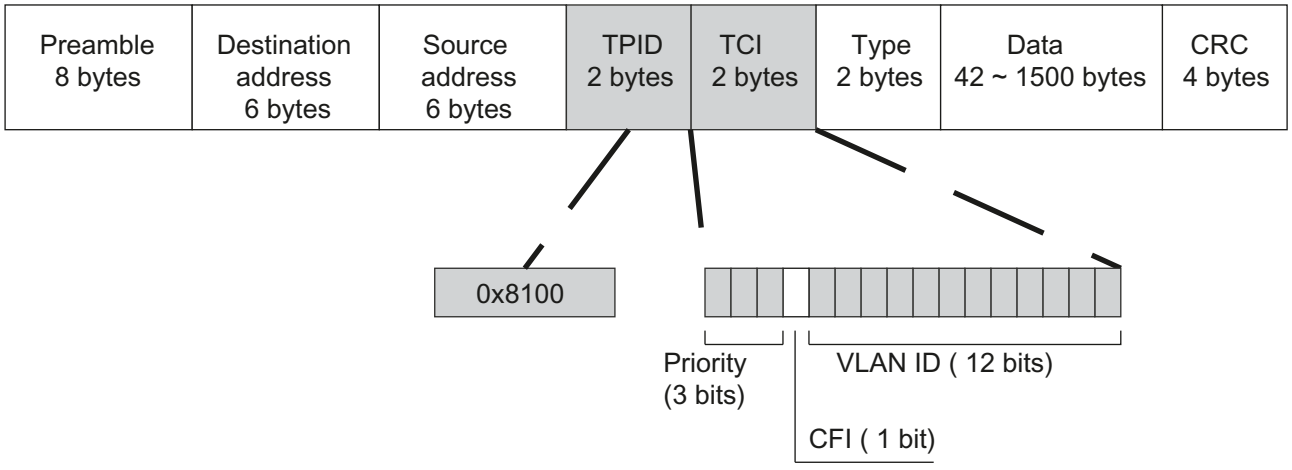

Figure 1-18 Structure of the expanded Ethernet frame

The additional bytes contain the tag protocol identifier (TPID) and the tag control information (TCI).

Tag protocol identifier (TPID)

The first 2 bytes form the Tag Protocol Identifier (TPID) and always have the value 0x8100. This value specifies that the data packet contains VLAN information or priority information.

Tag Control Information (TCI)

The 2 bytes of the Tag Control Information (TCI) contain the following information:

QoS Trust

The tagged frame has 3 bits for the priority that is also known as Class of Service (CoS), see also IEEE 802.1Q.

CoS bits	Priority	Type of the data traffic
000	0 (lowest)	Background
001	1	Best Effort
010	2	Excellent Effort
011	3	Critical Applications
100	4	Video, < 100 ms delay (latency and jitter)
101	5	Voice (language), < 10 ms delay (latency and jitter)
110	6	Internetwork Control
111	7 (highest)	Network Control

The prioritization of the data packets is possible only if there is a queue in the components in which they can buffer data packets with lower priority.

The device has multiple parallel queues in which the frames with different priorities can be processed. As default, first, the frames with the highest priority are processed. This method ensures that the frames with the highest priority are sent even if there is heavy data traffic.

Canonical Format Identifier (CFI)

The CFI is required for compatibility between Ethernet and the token Ring.
The values have the following meaning:

Value	Meaning
0	The format of the MAC address is canonical. In the canonical representation of the MAC address, the least significant bit is transferred first. Standard-setting for Ethernet switches.
1	The format of the MAC address is not canonical.

VLAN ID

In the 12-bit data field, up to 4096 VLAN IDs can be formed. The following conventions apply:

VLAN ID	Meaning
0	The frame contains only priority information (priority tagged frames) and no valid VLAN identifier.
1- 4094	Valid VLAN identifier, the frame is assigned to a VLAN and can also include priority information.
4095	Reserved

SNMP

SNMP

Introduction

With the aid of the Simple Network Management Protocol (SNMP), you monitor and control network components from a central station, for example routers or switches. SNMP controls the communication between the monitored devices and the monitoring station.

Tasks of SNMP:

- Monitoring of network components
- Remote control and remote parameter assignment of network components
- Error detection and error notification

In versions v1 and v2c, SNMP has no security mechanisms. Each user in the network can access data and also change parameter assignments using suitable software.

For the simple control of access rights without security aspects, community strings are used.

The community string is transferred along with the query. If the community string is correct, the SNMP agent responds and sends the requested data. If the community string is not correct, the SNMP agent discards the query. Define different community strings for read and write permissions. The community strings are transferred in plain text.

Standard values of the community strings:

- public
  has only read permissions
- private
  has read and write permissions

Note

Because the SNMP community strings are used for access protection, do not use the standard values "public" or "private". Change these values following the initial commissioning.

Further simple protection mechanisms at the device level:

- Allowed Host
  The IP addresses of the monitoring systems are known to the monitored system.
- Read Only
  If you assign "Read Only" to a monitored device, monitoring stations can only read out data but cannot modify it.

SNMP data packets are not encrypted and can easily be read by others.

The central station is also known as the management station. An SNMP agent is installed on the devices to be monitored with which the management station exchanges data.

The management station sends data packets of the following type:

- GET
  Request for a data record from the SNMP agent
- GETNEXT
  Calls up the next data record.
- GETBULK (available as of SNMPv2c)
  Requests multiple data records at one time, for example several rows of a table.
- SET
  Contains parameter assignment data for the relevant device.

The SNMP agent sends data packets of the following type:

- RESPONSE
  The SNMP agent returns the data requested by the manager.
- TRAP
  If a certain event occurs, the SNMP agent itself sends traps.

SNMPv1/v2c/v3 use UDP (User Datagram Protocol) and use the UDP ports 161 and 162. The data is described in a Management Information Base (MIB).

SNMPv3

Compared with the previous versions SNMPv1 and SNMPv2c, SNMPv3 introduces an extensive security concept.

SNMPv3 supports:

- Fully encrypted user authentication
- Encryption of the entire data traffic
- Access control of the MIB objects at the user/group level

With the introduction of SNMPv3 you can no longer transfer user configurations to other devices without taking special action, e.g. by loading a configuration file or replacing the C-PLUG.

According to the standard, the SNMPv3 protocol uses a unique SNMP engine ID as an internal identifier for an SNMP agent. This ID must be unique in the network. It is used to authenticate access data of SNMPv3 users and to encrypt it.

Depending on whether you have enabled or disabled the “SNMPv3 User Migration” function, the SNMP engine ID is generated differently.

Restriction when using the function

Use the "SNMPv3 User Migration" function only to transfer configured SNMPv3 users to a substitute device when replacing a device.
Do not use the function to transfer configured SNMPv3 users to multiple devices. If you load a configuration with created SNMPv3 users on several devices, these devices use the same SNMP engine ID. If you use these devices in the same network, your configuration contradicts the SNMP standard.

Compatibility with predecessor products

You can only transfer SNMPv3 users to a different device if you have created the users as migratable users. To create a migratable user the "SNMPv3 User Migration" function must be activated when you create the user.

Security functions

User management (WBM, CLI)

Overview of user management

Access to the device is managed by configurable user settings. Set up users with a password for authentication. Assign a role with suitable rights to the users.

The authentication of users can either be performed locally by the device or by an external RADIUS server. You configure how the authentication is handled on the "Security > AAA > General" page.

Local logon

The local logging on of users by the device runs as follows:

1. The user logs on with user name and password on the device.
2. The device checks whether an entry exists for the user.
   → If an entry exists, the user is logged in with the rights of the associated role.
   → If no corresponding entry exists, the user is denied access.

Login via an external RADIUS server

RADIUS (Remote Authentication Dial-In User Service) is a protocol for authenticating and authorizing users by servers on which user data can be stored centrally.

Depending on the RADIUS authorization mode you have selected on the "Security > AAA > RADIUS Client" page, the device evaluates different information of the RADIUS server.

RADIUS authorization mode "Standard"

If you have set the authorization mode "conventional", the authentication of users via a RADIUS server runs as follows:

1. The user logs on with user name and password on the device.
2. The device sends an authentication request with the login data to the RADIUS server.
3. The RADIUS server runs a check and signals the result back to the device.
   - The RADIUS server reports a successful authentication and returns the value "Administrative User" to the device for the attribute "Service Type".
     → The user is logged in with administrator rights.
   - The RADIUS server reports a successful authentication and returns a different or even no value to the device for the attribute "Service Type".
     → The user is logged in with read rights.
   - The RADIUS server reports a failed authentication to the device:
     → The user is denied access.

RADIUS authorization mode "SiemensVSA"

Requirement

For the RADIUS authorization mode "Siemens VSA" the following needs to be set on the RADIUS server:

- Manufacturer code: 4196
- Attribute number: 1
- Attribute format: Character string (group name)

Procedure

If you have set the authorization mode "SiemensVSA", the authentication of users via a RADIUS server runs as follows:

1. The user logs on with user name and password on the device.
2. The device sends an authentication request with the login data to the RADIUS server.
3. The RADIUS server runs a check and signals the result back to the device.
   Case A: The RADIUS server reports a successful authentication and returns the group assigned to the user to the device.
   - The group is known on the device and the user is not entered in the table "External User Accounts"
     → The user is logged in with the rights of the assigned group.
   - The group is known on the device and the user is entered in the table "External User Accounts"
     → The user is assigned the role with the higher rights and logged in with these rights.
   - The group is not known on the device and the user is entered in the table "External User Accounts"
     → The user is logged in with the rights of the role linked to the user account.
   - The group is not known on the device and the user is not entered in the table "External User Accounts"
     → The user is logged in with the rights of the role "Default".

   Case B: The RADIUS server reports a successful authentication but does not return a group to the device.
   - The user is entered in the table "External User Accounts":
     → The user is logged in with the rights of the linked role "".
   - The user is not entered in the table "External User Accounts":
     → The user is logged in with the rights of the role "Default".

   Case C: The RADIUS server reports a failed authentication to the device:
   - The user is denied access.

Firewall

Firewall

The security functions of the device include a stateful inspection firewall. This is a method of packet filtering or packet checking.

The IP packets are checked based on firewall rules in which the following is specified:

- The permitted protocols
- IP addresses and ports of the permitted sources
- IP addresses and ports of the permitted destinations

If an IP packet fits the specified parameters, it is allowed to pass through the firewall. The rules also specify what is done with IP packets that are not allowed to pass through the firewall.

Simple packet filter techniques require two firewall rules per connection.

- One rule for the query direction from the source to the destination.
- A second rule for the response direction from the destination to the source

Stateful Inspection Firewall

You only need to specify one firewall rule for the query direction from the source to the destination. The second rule is added implicitly. The packet filter recognizes when, for example, computer "A" is communicating with computer "B" and only then does it allow replies. A query by computer "B" is therefore not possible without a prior request by computer "A".

You configure the firewall in "Security > Firewall".

Note

IP packets via layer 2 (within the same VLAN)

If the IP packets from the device are sent via a switch port (layer 2), these IP packets are not checked based on firewall rules. The firewall has no effect on packets forwarded at the layer 2 level.

Communication directions

from	to	Meaning
vlan x	vlan x	Access from IP subnet vlan x to IP subnet vlan x. Example: vlan1 (INT) → vlan2 (EXT) Access from the local IP subnet to the external IP subnet.
	ppp2	Access from the IP subnet to the WAN interface of the device.
	Device	Access from the IP subnet to the device.
	SINEMA RC	Access from the IP subnet to the SINEMA RC connection.
	IPsec (all) IPsec <Connection Name> OpenVPN (all) OpenVPN <Connection Name>	Access from the IP subnet to the VPN tunnel partners that can be reached via all VPN connections (all) or via a certain VPN connection <Connection Name>.

from	to	Meaning
Device	vlan x	Access from the device to the IP subnet.
	ppp2	Access from the device to the WAN interface of the device.
	SINEMA RC	Access from the device to the SINEMA RC connection.
	IPsec (all) IPsec <Connection Name> OpenVPN (all) OpenVPN <Connection Name>	Access from the device to the tunnel partners that can be reached via all VPN connections (all) or via a certain VPN connection (<Connection Name>).
SINEMA RC	vlan x	Access from SINEMA RC connections to the IP subnet.
	ppp2	Access from the IP subnet to the WAN interface of the device.
	Device	Access from SINEMA RC connections to the device.
	IPsec (all) IPsec <Connection Name> OpenVPN (all) OpenVPN <Connection Name>	Access from the SINEMA RC server to the VPN tunnel partners that can be reached via all VPN connections (all) or via a certain VPN connection <Connection Name>.
IPsec (all) IPsec <Connection Name> OpenVPN (all) OpenVPN <Connection Name>	vlan x	Access via VPN tunnel partners to the IP subnet.
	ppp2	Access from the IP subnet to the WAN interface of the device.
	Device	Access via VPN tunnel partners to the device.
	SINEMA RC	Access via VPN tunnel partners to the SINEMA RC connection.
ppp0/usb	vlan x	Access from the mobile wireless interface to the IP subnet.
	Device	Access from the mobile wireless interface to the device.
	SINEMA RC	Access from the mobile wireless interface to the SINEMA RC connection.
	IPsec (all) IPsec <Connection Name> OpenVPN (all) OpenVPN <Connection Name>	Access from the mobile wireless interface to the VPN tunnel partners that can be reached via all VPN connections (all) or via a certain VPN connection <Connection Name>.

Default firewall setting

Firewall factory setting

Service	Access	
	from internal (vlan1) to the device	from external (vlan2) to the device
HTTP	yes	no
HTTPS	yes	no
TFTP	yes	no

Service	Access	
	from internal (vlan1) to the device	from external (vlan2) to the device
DNS	yes	no
SNMP	yes	no
Telnet	yes	no
IPsec VPN	no	yes
SSH	yes	no
DHCP	yes	yes (for the DHCP client function)
Ping	yes	no

NAT

NAT (Network Address Translation) is a method of translating IP addresses in data packets. With this, two different networks (internal and external) can be connected together.

A distinction is made between source NAT in which the source IP address is translated and destination NAT in which the destination IP address is translated.

You will find information on NAT scenarios that are implemented with the device at the following address:

IP masquerading

IP masquerading is a simplified source NAT. With each outgoing data packet sent via this interface, the source IP address is replaced by the IP address of the interface. The adapted data packet is sent to the destination IP address. For the destination host it appears as if the queries always came from the same sender. The internal nodes cannot be reached directly from the external network. By using NAPT, the services of the internal nodes can be made reachable via the external IP address of the device.

IP masquerading can be used if the internal IP addresses cannot or should not be forwarded externally, for example because the internal network structure should remain hidden.

You configure masquerading in "Layer 3" > "NAT" > "IP Masquerading (Page 668)".

NAPT

NAPT (Network Address and Port Translation) is a form of destination NAT and is often called port forwarding. This allows the services of the internal nodes to be reached from external that are hidden by IP masquerading or source NAT.

Incoming data packets are translated that come from the external network and are intended for an external IP address of the device (destination IP address). The destination IP address is replaced by the IP address of the internal node. In addition to address translation, port translation is also possible.

The options are available for port translation:

from	to	Response
a single port	the same port	If the ports are the same, the frames will be forwarded without port translation.
a single port	a single port	The frames are translated to the port.
a port range	a single port	The frames from the port range are translated to the same port (n: 1).
a port range	the same port range	If the port ranges are the same, the frames will be forwarded without port translation.
a port range	another port range	The frames are translated to any free port from the target range. With individual connection, they are normally translated to the first port in the target range. If there are connections at the same time, the round robin method is used to translate to a free port in the target range.
a single port	a port range	The frames are translated to any free port from the target range. With individual connection, they are normally translated to the first port in the target range. If there are connections at the same time, the round robin method is used to translate to a free port in the target range.

Port forwarding can be used to allow external nodes access to certain services of the internal network e.g. FTP, HTTP.

You configure NAPT in "Layer 3" > "NAT" > "NAPT (Page 669)".

Source NAT

As with masquerading, in source NAT the source address is translated. In addition to this, the outgoing data packets can be restricted. These include limitation to certain IP addresses or IP address ranges and limitation to certain interfaces.

Source NAT can be used if the internal IP addresses cannot or should not be forwarded externally, for example because a private address range such as 192.168.x.x is used.

You configure source NAT in "Layer 3" > "NAT" > "Source NAT (Page 671)".

NETMAP

With NETMAP it is possible to translate complex subnets to a different subnet. In this translation, the subnet part of the IP address is changed and the host part remains. For translation with NETMAP only one rule is required. NETMAP can translate both the source IP address and the destination IP address. To perform the translation with destination NAT and source NAT, numerous rules would be necessary. NETMAP can also be applied to VPN connections.

You configure NETMAP in "Layer 3" > "NAT" > "NETMAP (Page 672)".

NAT and firewall

The firewall and NAT router support the "Stateful Inspection" mechanism. If the IP data traffic from internal to external is enabled, internal notes can initiate a communications connection into the external network.

The reply frames from the external network can pass through the NAT router and firewall without it being necessary for their addresses to be included extra in the firewall rule and the NAT address translation. Frames that are not a reply to a query from the internal network are discarded without a matching firewall rule.

NAT translation and firewall rules

You will find an example of NAT translations on the Internet pages of Siemens Industry Online Support.

Link: (https://support.industry.siemens.com/cs/ww/en/view/109744660)

Certificates

Certificate types

The device uses different certificates to authenticate the various nodes.

Certificate		Is used in...
CA certificate	The CA certificate is a certificate issued by a Certificate Authority from which the server, device and partner certificates are derived. To allow a certificate to be derived, the CA certificate has a private key signed by the certificate authority. The key exchange between the device and the VPN gateway of the partner takes place automatically when establishing the connection. No manual exchange of key files is necessary.	IPsec VPN
Server certificate	Server certificates are required to establish secure communication (e.g. HTTPS, VPN...) between the device and another network participant. The server certificate is an encrypted SSL certificate. The server certificate is derived from the oldest valid CA, even if this is "out of service". The crucial thing is the validity date of the CA.	SINEMA RC
Device certificate	Certificates with the private key (key file) with which the device identifies itself.	IPsec VPN
Partner certificate	Certificates with which the VPN gateway of the partner identifies itself with the device.	IPsec VPN

File types

File type	Description
*.crt	File that contains the certificate.
*.p12	In the PKCS12 certificate file, the private key is stored with the corresponding certificate and is password protected. The CA creates a certificate file (PKCS12) for both ends of a VPN connection with the file extension ".p12". This certificate file contains the public and private key of the local station, the signed certificate of the CA and the public key of the CA.
*.pem	Certificate and key as Base64-coded ASCII text.

VPN

Overview

The device supports the following VPN systems

- IPsec VPN (only for SCALANCE S615 and SCALANCE SC64x-2C)
- OpenVPN (only for SCALANCE S615)

IPsec VPN

You configure the IPsec connections in "Security" > " IPsec VPN".

With IPsec VPN, the frames are transferred in tunnel mode. To allow the device to establish a VPN tunnel, the remote network must have a VPN gateway as the partner.

For the VPN connections, the device distinguishes two modes:

- Roadwarrior mode
  In this mode either the address of the partner is fixed or an IP range is entered from which the connections are taken. The device learns the reachable remote subnets from the partner.
- Standard mode
  In this mode the address of the partner or the remote subnet is entered permanently. The device can either establish the connection actively as a VPN client or wait passively for connection establishment by the partner.

The IPsec method

The device uses the IPsec method in the tunnel mode for the VPN tunnel. Here, the frames to be transferred are completely encrypted and provided with a new header before they are sent to the VPN gateway of the partner. The frames received by the partner are decrypted and forwarded to the recipient.

To provide security, the IPsec protocol suite uses various protocols:

- The IP Authentication Header (AH) handles the authentication and identification of the source.
- The Encapsulation Security Payload (ESP) encrypts the data.

- The Security Association (**SA**) contains the specifications negotiated between the partners, e.g. about the lifetime of the key, the encryption algorithm, the period for new authentication etc.
- Internet Key Exchange (**IKE**) is a key exchange method. The key exchange takes place in two phases:
  - Phase 1
    In this phase, no security services such as encryption, authentication and integrity checks are available yet since the required keys and the IPsec SA still need to be created. Phase 1 serves to establish a secure VPN tunnel for phase 2. To achieve this, the communications partners negotiate an ISAKMP Security Association (ISAKMP SA) that defines the required security services (algorithms, authentication methods used). The subsequent messages and phase 2 are therefore secure.
  - Phase 2
    Phase 2 serves to negotiate the required IPsec SA. Similar to phase 1, exchanging offers achieves agreement about the authentication methods, the algorithms and the encryption method to protect the IP packets with IPsec AH and IPsec ESP.
    The exchange of messages is protected by the ISAKMP SA negotiated in phase 1. Due to the ISAKMP SA negotiated in phase 1, the identity of the nodes is known and the method for the integrity check already exists.

Authentication method

- CA certificate, device and partner certificate (digital signatures)
  The use of certificates is an asymmetrical cryptographic system in which every node (device) has a pair of keys. Each node has a secret, private key and a public key of the partner. The private key allows the device to authenticate itself and to generate digital signatures.
- Pre-shared key
  The use of a pre-shared key is a symmetrical cryptographic system. Each node has only one secret key for decryption and encryption of data packets. The authentication is via a common password.

Local ID and remote ID

The local ID and the remote ID are used by IPsec to uniquely identify the partners (VPN end point) during establishment of a VPN connection.

Encryption methods

The following encryption methods are supported. The selection depends on the phase und the key exchange method (IKE)

	Phase 1		Phase 2	
	IKEv1	IKEv2	IKEv1	IKEv2
3DES	x	x	x	x
AES128 CBC	x	x	x	x
AES192 CBC	x	x	x	x
AES256 CBC	x	x	x	x

	Phase 1		Phase 2	
	IKEv1	IKEv2	IKEv1	IKEv2
AES128 CTR	-	x	x	x
AES192 CTR	-	x	x	x
AES256 CTR	-	x	x	x
AES128 CCM 16	-	x	x	x
AES192 CCM 16	-	x	x	x
AES256 CCM 16	-	x	x	x
AES128 GCM 16	-	x	x	x
AES192 GCM 16	-	x	x	x
AES256 GCM 16	-	x	x	x

x: is supported

-: is not supported

Default Ciphers

During connection establishment a preset list can be transferred to the VPN connection partners. The list contains combinations of the three algorithms (Encryption, Authentication, Key Derivation). To establish a VPN connection, the VPN connection partner must support at least one of these combinations. The combinations depend on the phase und the key exchange method IKE).

Combination			Phase 1		Phase 2	
Encryption	Authentica-tion	Key derivation	IKEv1	IKEv2	IKEv1	IKEv2
AES128	SHA1	DH Group 14	x	x	x	x
AES256	SHA512	DH Group 16	x	x	x	x
AES128 CCM 16	SHA256	DH Group 14	-	x	x	x
AES256 CCM 16	SHA512	DH Group 16	-	x	x	x
AES128	SHA1	none	-	-	x	x
AES256	SHA512	none	-	-	x	x
AES128 CCM 16	SHA256	none	-	-	x	x
AES256 CCM 16	SHA512	none	-	-	x	x

x: Combination is part of the default cipher

-: Combination is not part of the default cipher

none: For phase 2, no separate keys are exchanged. This means that Perfect Forward Secrecy (PFS) is disabled.

Requirements of the VPN partner

The VPN partner must support IPsec with the following configuration to be able to establish an IPsec connection successfully:

- Authentication with partner certificate, CA certificates or pre-shared key
- IKEv1 or IKEv2
- Support of at least one of the following DH groups: Diffie-Hellman group 1, 2, 5 and 14 - 18

- 3DES or AES encryption
- MD5, SHA1, SHA256, SHA384 or SHA512
- Tunnel mode

If the VPN partner is downstream from a NAT router, the partner must support NAT-T. Or, the NAT router must know the IPsec protocol (IPsec/VPN passthrough).

NAT traversal (NAT-T)

There may be a NAT router between the device and the VPN gateway of the remote network. Not all NAT routers allow IPsec frames to pass through. This means that it may be necessary to encapsulate the IPsec frames in UDP packets to be able to pass through the NAT router.

Dead peer detection

This is only possible when the VPN partner supports DPD. DPD checks whether the connection is still operating problem free or whether there has been an interruption on the line. Without DPD and depending on the configuration, it may be necessary to wait until the SA lifetime has expired or the connection must be reinitiated manually. To check whether the IPsec connection is still problem-free, the device itself sends DPD queries to the VPN partner station. If the VPN partner station does not reply after a certain time has elapsed, the connection to the VPN partner station will be declared invalid. You configure the settings for DPD in phase 1.

OpenVPN

With OpenVPN, virtual private networks (VPN) can be established. As an OpenVPN client, the device can establish a VPN connection to a remote network.

You configure the OpenVPN client in "Security" > " OpenVPN Client".

The VPN connection is established via virtual device drivers, the TAP and TUN device. During this, virtual network interfaces are created that act like a physical interface of the device and represent the endpoint of the VPN tunnel.

The device supports the following:

- TUN device: Routing mode
  The LAN Interface and the virtual network interface are located in different IP subnets. The virtual tunnel interface is assigned a virtual IP address from a devised subnet by the OpenVPN server. The IP packets (layer 3) are routed between the virtual tunnel interface and the LAN interface.

Authentication method

- Certificates: CA certificate and device certificate
  The use of certificates is an asymmetrical cryptographic system. Each node (device) has a secret, private key and a public key of the partner. The private key allows the device to authenticate itself and to generate digital signatures.
- User name / password
  Access is restricted by a user name and a password.

Encryption methods

The device also supports the following methods:

- BF CBC
- AES128 CBC
- AES192 CBC
- AES256 CBC
- DES EDE3

VPN connection establishment

The device supports the following options for establishing a VPN connection.

- OpenVPN: Security > OpenVPN > Connections
- IPsec VPN: Security > IPsec VPN > Connections (Page 696)
- SINEMA RC: System > SINEMA RC (Page 647)

Options	Use			Description
	OpenVPN	IP-sec VPN	SINE-MA RC [1]	
start	x	x	-	The device is "active", in other words, it attempts to establish a connection to a partner. The partner is addressed using its configured WAN IP address or the configured FQDN.
wait	-	x	-	The device is "passive", in other words, it waits for the partner to initiate the connection.
on demand	-	x	-	The device attempts to establish a connection to a partner when necessary. The receipt of requests for VPN connection establishment is also possible. For the configured local and remote subnets, an entry is created in the routing table. If a node attempts to send data packets via the VPN tunnel from one of the networks, the VPN connection is established. The settable timeout has the effect that after this time without any further data packets the VPN tunnel is terminated again.
start on DI	x	x	-	Connection establishment is controlled via the digital input (DI).
wait on DI	-	x	-	
Digital input	-	-	x	
Auto	-	-	x	The device adopts the settings of the SINEMA RC server. You configure the settings on the SINEMA RC Server in "Remote Connections > Devices". You will find further information on this topic in the operating instructions "SINEMA RC Server".
Permanent	-	-	x	The device establishes a VPN connection to the SINEMA RC Server. The VPN tunnel is established permanently

[1] For SCALANCE S615: KEY-PLUG SINEMA REMOTE CONNECT required

Digital input (DI)

The establishment of the VPN tunnel can also be controlled via the digital input, e.g. using a button. When the button is closed, voltage is applied to the digital input and the LED of the digital input lights up. The lit LED indicates that signal 1 (TRUE / HIGH) is applied. Signal 1 triggers an event on the device with which the establishment of the VPN tunnel is controlled. You will find information on connecting and the maximum current load in the operating instructions of the devices.

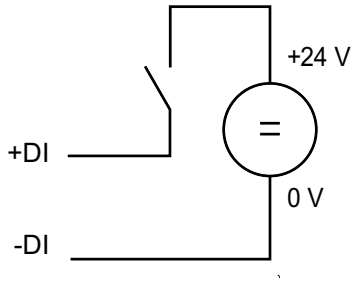

Requirement

- In "System > Events > Configuration" for the "Digital Input" event "VPN Tunnel" is activated. If this setting is not activated, the event is not passed on to the VPN connection.

Options

The device supports the following options for controlling the VPN tunnel via the digital input:

- start on DI
  If the event "Digital Input" occurs, the device becomes "active". The device attempts to establish a VPN connection (OpenVPN, IPsec) to a partner.
- wait on DI
  If the event "Digital Input" occurs, the device becomes "passive". The device waits for the partner to initiate the connection.
- Digital input
  The settings of the SINEMA RC server are ignored. If the event "Digital In" occurs, the device becomes "active". The device attempts to establish a VPN connection to the SINEMA RC server.

Notification options

If the status of the digital input or a VPN tunnel (IPsec, OpenVPN, SINEMA RC) changes, the device provides several options for notification on the "Events" page.

Type of notification	Digital In	VPN tunnel	Behavior if there is a status change
E-mail	x	x	The device sends an e-mail. The e-mail contains the identification of the sending device, a description of the cause of the alarm in plain language, and a time stamp. Requirement: • An SMTP server is set up. • In "System > SMTP Client" the function is activated, a recipient and the IP address of the SMTP server are configured.
Trap	x	x	The device sends an SNMP trap. Requirement: • "SNMPv1 traps" is enabled in "System > Configuration". • In "System > Configuration > Traps" a recipient is configured to which the device sends the SNMP traps.
Log table	x	x	The device writes an entry in the event log table. The content of the event log table is displayed in "Information > Log Table".
Syslog	x	x	The device writes an entry to the Syslog server. Requirement: • A Syslog server has been set up. • In "System > Syslog Client" the function is activated and the IP address of the Syslog server is configured.
Fault LED	x	-	The fault LED lights up on the device.

Type of notification	Digital In	VPN tunnel	Behavior if there is a status change
Digital iInput	x	x	Controls the digital output or signals the status change with the "DO" LED. A consumer can be connected to the digital output. You will find information on connecting in the operating instructions of the devices. The consumer signals a status change. 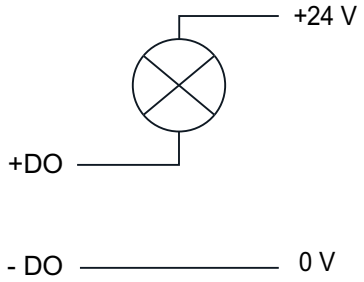 **Note** You can control the digital output directly via CLI or SNMP. In the WBM and CLI, you can configure the use of the digital output in "Events". Do not control the digital output directly when you use this in the WBM and CLI.
Read out the status of the MIB variable	x	-	Using the private MIB variable snMspsDigitalInputLevel, you can read out the status of the digital input. • OID of the private MIB variable snMspsDigitalInputLevel: `iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).siemens(4329).industrialComProducts(20).iComPlatforms(1).simaticNet(1).snMsps(1).snMspsCommon(1).snMspsDigitalIO(39).snMspsDigitalIOObjects(1).snMspsDigitalInputTable(2).snMspsDigitalInputEntry(1).snMspsDigitalInputLevel(6)` • values of the MIB variable – 1: Signal 0 at the digital input (DI) – 2: Signal 1 at the digital input (DI)

Special features of user management for security functions

Access to security functions

You can only access security functions after activating project protection and after a user has successfully logged on. The logged on user must also have the required engineering and runtime rights.

You will find basic information on user management in STEP 7 in the "Using user management" section of the information system.

Engineering rights for security devices

These engineering rights allow diagnostics and configuration of security devices:

- Security: Security modules with read rights: Display of the global security functions and the local security settings of the security devices.
- Security: Security modules with write rights: Configuration of the global security functions and the local security settings of the security devices and perform diagnostics of the security devices.

Roles with runtime rights for SCALANCE S 600

As default, some system-defined roles with the prefix "NET" are assigned runtime rights for SCALANCE S-600. Runtime rights provide the rights to execute certain tasks on the devices. Just as with engineering rights you cannot change runtime rights for system-defined roles. For user-defined roles you can change the runtime rights as required. You can specify the runtime rights of a user-defined role for each device.

There must be at least one user assigned a role with full runtime rights for a security module. e.g. the system-defined role "NET Administrator".

The system-defined roles with the prefix "NET" are assigned engineering rights:

Right	NET Administrator	NET Standard	NET Diagnose	NET Remote Access	NET Administrator Radius	NET Radius
Open the project read only	x	x	x	-	-	-
Open the project with write rights	x	x	-	-	-	-
Edit security settings: "Settings" and "Users and roles"	x	-	-	-	-	-
Security: Security modules with read rights	x	x	x	-	x	x
Security: Security modules with write rights	x	x	-	-	x	-

Runtime rights for SCALANCE S 600

The following runtime rights are available:

Right	NET Administrator	NET Standard	NET Diagnose	NET Remote Access	NET Administrator Radius	NET Radius
Download the configuration files	x	x	-	-	x	-
TIA Portal: Perform diagnostics of the security module	x	x	x	-	x	x

Certificates for security functions

During configuration only certificates generated by the TIA Portal can be used, for example VPN group certificates. Certificates not generated by the TIA Portal. for example CA certificates for SINEMA Remote Connect, cannot be used. You need to add such certificates later via the WBM and download them to the device. You will find more information in the relevant WBM configuration manual.

Editing properties and parameters

Editing options

You have the following options for editing properties and parameters:

- Hardware and network editor
  Once you have inserted the network component, you can edit the properties and parameters, for example the device name. You will find more detailed information in "Hardware and network editor".
- Web Based Management (WBM)
  The parameters and properties are accessible using the supplied HTML pages (WBM pages). Each WBM page has its own help page that describes the properties and parameters. For further information, refer to the configuration manuals of the devices. You will find these on the pages of Siemens Industry Online Support.
- Command Line Interface
  All the configuration settings for the device can be made using the CLI. The CLI provides the same options as Web Based Management (WBM). For further information, refer to the configuration manuals of the devices. You will find these on the pages of Siemens Industry Online Support.

Operating modes

The following operating modes are available:

- Offline configuration mode
  In this mode only the settings that require no connection to the device are available.
- Online diagnostics mode
  If there is an online connection to the device, additional pages are available in "Online & Diagnostics". The documentation of these pages includes the note "This page is only available if there is an online connection to the device". With some settings, the additional information "only available online" is included.

Creating and deleting an entry

As an example, an entry will be created and deleted on the Syslog client. The procedure is valid on all pages.

Creating a new entry

1. Select the device in the network or device view.
2. Open the properties of the device in the Inspector window.

3. In the Inspector window go to System > Syslog client.
4. Enter the IP address for the Syslog server.
5. Click in the table.
6. Select the entry "New entry" in the shortcut menu.
   A new entry is created in the table.

Deleting an entry

1. Select the required entry in the table.
2. Select the "Delete" entry in the shortcut menu.

Buttons you require often

- **Refresh the display with "Refresh"**
  Pages that display current parameters have a "Refresh" button at the lower edge of the page. Click this button to request up-to-date information from the device for the current page.
- **Save settings for all ports with "Transfer to table"**
  Pages on which you can configure several ports have 2 tables. In the first table, you can make the settings for all ports and transfer these to the second table. In the last column of the first table, there is the "Transfer to table" button. Click this button to save the settings you have made for all ports.

Showing information

Versions

This page shows the versions of the hardware and software of the device.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Hardware**
  - Basic Device
    Shows the basic device
  - PX.X
    X.X = port in which the SFP module is inserted.
  - SlotX
    "X" = slot number: Module plugged into this slot.
- **Name**
  Shows the name of the device or module.

- **Revision**
  Shows the hardware version of the device.
- **Article number**
  Shows the article number of the device or described module.
- **Software**
  - Firmware
    Shows the current firmware version. If a new firmware file was downloaded and the device has not yet restarted, the firmware version of the downloaded firmware file is displayed here. After the next restart, the loaded firmware is activated and used.
  - Bootloader
    Shows the version of the boot software stored on the device.
  - Firmware_Running
    Shows the firmware version currently being used on the device.
- **Description**
  Shows the short description of the software.
- **Version**
  Shows the version number of the software version.
- **Date**
  Shows the date on which the software version was created.

I&M

Note

The page is only available if there is an online connection to the device.

This page contains information about device-specific vendor and maintenance data such as the article number, serial number, version number etc. You cannot configure anything on this page.

Description of the displayed values

The table has the following rows:

- **Manufacturer ID**
  Shows the manufacturer ID.
- Article number
  Shows the article number.
- **Serial number**
  Shows the serial number.
- **Hardware revision**
  Shows the hardware version.
- **Software version**
  Shows the software version.

- Revision counter
  Shows the revision counter: Counter for revisions since the initial commissioning
- Revision Date
  Date and time of the last revision
- Function tag
  Shows the function tag (plant designation) of the device. The plant designation (HID) is created during configuration of the device with HW Config of STEP 7.
- Location tag
  Shows the location tag of the device. The location identifier (LID) is created during configuration of the device with HW Config of STEP 7.
- Date
  Shows the date created during configuration of the device with HW Config of STEP 7.
- Description
  Shows the description created during configuration of the device with HW Config of STEP 7.

ARP table

Note

The page is only available if there is an online connection to the device.

Assignment of MAC address and IP address

With the Address Resolution Protocol (ARP), there is a unique assignment of MAC address to IPv4 address. This assignment is kept by each network node in its own separate ARP table. The page shows the ARP table of the device.

Description of the displayed values

The table has the following columns:

- Interface
  Shows the interface via which the row entry was learnt.
- MAC Address
  Shows the MAC address of the destination or source device.
- IP address
  Shows the IP address of the target device.
- Media Type
  Shows the type of connection.
  - Dynamic
    The device recognized the address data automatically.
  - Static
    The addresses were entered as static addresses.

Log tables

Event log

Logging events

Note

The page is only available if there is an online connection to the device.

The page shows the system events that have occurred in the form of a table. Some of the system events can be configured in "System > Events", for example if the connection status of a port has changed.

The content of the table is retained even when the device is turned off. The event log file can be downloaded using HTTP, TFTP or SFTP.

Description of the displayed values

- **Severity Filters**
  You can filter the entries in the table in the WBM according to severity. To display all the entries, enable or disable all parameters.
  - Info
    Informative
  - When this parameter is enabled, all entries of the category "Info" are displayed.
  - Warning
    Warning
    When this parameter is enabled, all entries of the category "Warning" are displayed.
  - Critical
    Critical
    When this parameter is enabled, all entries of the category "Critical" are displayed.

  The table has the following columns:
  - **Restart**
    Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.
  - **System Up Time**
    Shows the time the device has been running since the last restart when the described event occurred.
  - **System Time**
    If the system time is set, the date and time at which the event occurred are also displayed. If no system time is set, the box displays "Date/time not set".
  - **Severity**
    Sorting of the entry into the categories above.
  - **Log Message**
    Displays a brief description of the event that has occurred.

Note

The number of entries in this table is restricted to 1200. The table can contain 400 entries for each severity. When this number is reached, the oldest entries of the relevant severity are discarded. The table remains permanently in memory.

Security log

Note

The page is only available if there is an online connection to the device.

The page shows the events that occurred during communication via a secure VPN tunnel in the form of the table.

Description of the displayed values

- Severity Filters
  You can filter the entries in the table in the WBM according to severity. To display all the entries, enable or disable all parameters.
  - Info
    Informative
    When this parameter is enabled, all entries of the category "Info" are displayed.
  - Warning
    Warning
    When this parameter is enabled, all entries of the category "Warning" are displayed.
  - Critical
    Critical
    When this parameter is enabled, all entries of the category "Critical" are displayed.

The table has the following columns:

- Restart
  Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.
- System Up Time
  Shows the time the device has been running since the last restart when the described event occurred.
- System Time
  If the system time is set, the date and time are also displayed at which the event occurred. If no system time is set, the box displays "Date/time not set".
- Severity
  Sorts the entry into the categories above.
- Log Message
  Displays a brief description of the event that has occurred.

Buttons

"Clear" button

Click this button to delete the content of the event log file. All entries are deleted regardless of what you have selected in "Severity Filters".

The display is also cleared. The restart counter is only reset after you have restored the device to the factory settings and restarted the device.

Note

The number of entries in this table is restricted to 1200. The table can contain 400 entries for each severity. When this number is reached, the oldest entries of the relevant severity are discarded. The table remains permanently in memory.

Firewall log

Note

The page is only available if there is an online connection to the device.

The firewall log logs the events that occurred on the firewall. When you create firewall rules, you can specify the event severity with which they are logged.

Description of the displayed values

- Severity Filters
  You can filter the entries in the table in the WBM according to severity. To display all the entries, enable or disable all parameters.
  - Info
    Informative
    When this parameter is enabled, all entries of the category "Info" are displayed.
  - Warning
    Warning
    When this parameter is enabled, all entries of the category "Warning" are displayed.
  - Critical
    Critical
    When this parameter is enabled, all entries of the category "Critical" are displayed.

The table has the following columns:

- Restart
  Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.
- System Up Time
  Shows the time the device has been running since the last restart when the described event occurred.

- System Time
  If the system time is set, the date and time are also displayed at which the event occurred. If no system time is set, the box displays "Date/time not set".
- Severity
  Sorts the entry into the categories above.
- Log Message
  Displays a brief description of the event that has occurred.

Description of the buttons and input boxes

"Clear" button

Click this button to delete the content of the event log file. All entries are deleted regardless of what you have selected in "Severity Filters".

The display is also cleared. The restart counter is only reset after you have restored the device to the factory settings and restarted the device.

Note

The number of entries in this table is restricted to 1200. The table can contain 400 entries for each severity. When this number is reached, the oldest entries of the relevant severity are discarded. The table remains permanently in memory.

Fault

Error status

If an error occurs, it is shown on this page. On the device errors are signaled by the fault LED lighting up.

Internal errors of the device and errors that you configure on the following pages are indicated:

- "System > Events"
- "System > Fault Monitoring"

Errors of the "Cold/Warm Start" event can be deleted by a confirmation.

The calculation of the time of an error always begins after the last system start.

If there are no errors present, the fault LED switches off.

Note

The page is only available if there is an online connection to the device.

Displayed values

- **No. of Signaled Faults**
  Number of errors displayed since the last startup.
- **"Reset counters" button**
  Click "Reset Counters" to reset the counter. The counter is reset when there is a restart.

The table contains the following columns:

- **Fault Time**
  Shows the time the device has been running since the last restart when the described fault occurred.
- **Fault Description**
  Displays a brief description of the error that has occurred.
- **Clear Fault State**
  If the "Clear Fault State" button is enabled, you can delete the error.

SNMP

Note

The page is only available if there is an online connection to the device.

This page displays the created SNMPv3 groups. You configure the SNMPv3 groups in "System" > SNMP"..

Description of the displayed values

The table has the following columns:

- **Group Name**
  Shows the group name.
- **User Name**
  Shows the user that is assigned to the group.

LLDP

Note

The page is only available if there is an online connection to the device.

Status of the neighborhood table

This page shows the current content of the neighborhood table. This table stores the information that the LLDP agent has received from connected devices. You set the interfaces via which the LLDP agent receives or sends information in the following section: "Layer 2 > LLDP".

Description of the displayed values

The table contains the following columns:

- System name
  System name of the connected device.
- Device ID
  Device ID of the connected device. The device ID corresponds to the device name assigned via PST (STEP 7). If no device name is assigned, the MAC address of the device is displayed.
- Local interface
  Port at which the device received the information
- Hold Time
  An entry remains stored on the device for the time specified here. If the device does not receive any new information from the connected device during this time, the entry is deleted.
- Capability
  Shows the capabilities of the connected device:
  - Router
  - Bridge
  - Telephone
  - DOCSIS Cable Device
  - WLAN Access Point
  - Repeater
  - Station
  - Other
- Port ID
  Port of the device with which the device is connected.

Routing

Introduction

Note

The page is only available if there is an online connection to the device.

This page shows the routes currently being used.

Description of the displayed values

The table has the following columns:

- Destination Network
  Shows the destination address of this route.
- Subnet Mask
  Shows the subnet mask of this route.
- Gateway
  Shows the gateway for this route.
- Interface
  Shows the interface for this route.
- Metric
  Shows the metric of the route. The higher value, the longer packets require to their destination.
- Routing Protocol
  Shows the routing protocol from which the entry in the routing table originates. The following entries are possible:
  - Connected: Connected routes
  - Static: Static routes

IPsec VPN

The page shows the status of the activated VPN connections.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

This table contains the following columns:

- Name
  Shows the name of the VPN connection.
- Local Host
  Shows the IP address of the device.
- Local DN
  Shows the Distinguished Name (DN) of the device that was signaled to the remote station during connection establishment. The entry is adopted from the "Local ID" box, the device certificate or the IP address of the device.
- Local Subnet
  Shows the local subnet.
- Remote Host
  Shows the IP address or the host name of the remote device.

- Remote DN
  Shows the Distinguished Name (DN) signaled by the remote device during connection establishment.
- IRemote Subnet
  Shows the remote subnet.
- Rekey Time
  Shows when the validity of the key expires.
- Status
  Shows the status of the VPN connection.

SINEMA RC

Note

The page is only available if there is an online connection to the device.

Shows information on SINEMA RC Server.

Note

For SCALANCE S615 this function can only be used with a KEY-PLUG.

Description of the displayed values

- Status
  Shows the status of the connection to SINEMA RC Server.
- Device Name
  If configured, the name of the device is displayed.
- Device Location
  If configured, the location of the device is displayed.
- GSM Number
  If configured, the phone number of the device is displayed.
- Vendor
  If configured, the entry is displayed.
- Comment
  If configured, the comment is displayed.
- Type of Connection (Server)
  Shows which type of connection is set on the SINEMA RC Server.
- Type of Connection (device)
  Shows which type of connection is set on the device.
- Fingerprint
  Shows the fingerprint of the server certificate. Is only displayed when the fingerprint is used for verification.

- Remote Address
  Shows the IP address of the SINEMA RC Server.
- Connected Local Subnet (s)
  Shows the IP addresses of the local subnets. Is only displayed when the option "Connected local subnets" is enabled on the SINEMA RC Server. You will find further information on this in the Operating Instructions of the SINEMA RC Server.
- Connected Local Host (s)
  Shows the destination IP address of the hosts that can be reached.
- Tunnel Interface Address
  Shows the IP address of the virtual tunnel interface.
- Connected Remote Subnet (s)
  Shows the subnets of the SINEMA RC Server that are reachable for the device. Which subnets are reachable for the device depends on the communications relations on the SINEMA RC Server. You will find further information on this in the Operating Instructions of the SINEMA RC Server.

OpenVPN Client

The page shows the status of the activated OpenVPN connections.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

This table contains the following columns:

- Name
  Shows the name of the OpenVPN connection.
- Remote Server
  Shows the IP address or the hostname of the OpenVPN server.
- Tunnel Interface IP
  Shows the IP address of the virtual tunnel interface.
- Exported Subnets
  Shows the IP address of the local subnets.
- Routed Subnets
  Shows the subnets of the OpenVPN server.
- Status
  Shows the status of the OpenVPN connection.

Redundancy

Overview

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The page contains the following boxes:

- **Bridge Priority / Root Priority**
  Which device becomes the root bridge is decided by the bridge priority. The bridge with the highest priority (in other words, with the lowest value for this parameter) becomes the root bridge. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames.
- **Bridge Address / Root Address**
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Port**
  Shows the port via which the switch communicates with the root bridge.
- **Root Cost**
  The path costs from this device to the root bridge.
- **Topology Changes / Last Topology Change**
  The entry for the device shows the number of reconfiguration actions due to the spanning tree mechanism since the last startup. For the root bridge, the time since the last reconfiguration is displayed as follows:
  - Seconds: Supplement "sec" after the number
  - Minutes: Supplement "min" after the number
  - Hours: Supplement "hr" after the number
- **Bridge hello time [s] / Root hello time [s]**
  Each bridge sends configuration frames (BPDUs) regularly. The interval between two such frames is the Hello time.
- **Bridge Forward Delay / Root Forward Delay**
  New configuration information is not used immediately by a bridge but only after the forwarding delay specified in the parameter. This ensures that operation is only started with the new topology after all the bridges have the required information. The
- **Bridge Max Age / Root Max Age**
  When the max age timer elapses the received BPDU is discarded to be accepted as valid by the switch. The default value is 20s.

- **Bridge Max Hop Count**
  This parameter specifies how many STP nodes a BPDU may pass through. If an MSTP BPDU is received and has a hop count that exceeds the value configured here, it is discarded.
- **Root Hop Count**
  The number of nodes that need to be run through on the way to the root bridge.

Spanning Tree

Note

The page is only available if there is an online connection to the device.

Introduction

The page shows the current information about the spanning tree and the settings of the root bridge.

Description of the displayed values

The following fields are displayed:

- **Spanning Tree Mode**
  Shows the set mode. You specify the mode in "Layer 2 > Configuration" and in "Layer 2 > Spanning Tree > General".
  The following values are possible:
  - '-'
  - RSTP
- **Bridge Priority / Root Priority**
  Which device becomes the root bridge is decided by the bridge priority. The bridge with the highest priority (in other words, with the lowest value for this parameter) becomes the root bridge. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames. The value for the bridge priority is a whole multiple of 4096 with a range of values from 0 to 32768.
- **Bridge Address / Root Address**
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root switch.
- Root Cost
  Shows the path costs from the device to the root bridge.
- Bridge Status
  Shows the status of the bridge, e.g. whether or not the device is the root bridge.

The table has the following columns:

- **Port**
  Shows the interfaces via which the device communicates.
- **Role**
  Shows the status of the port. The following values are possible:
  - Disabled
    The port was removed manually from the spanning tree and will no longer be taken into account by the spanning tree.
  - Designated
    The ports leading away from the root bridge.
  - Alternate
    The port with an alternative route to a network segment
  - Backup
    If a switch has several ports to the same network segment, the "poorer" Port becomes the backup port.
  - Root
    The port that provides the best route to the root bridge.
  - Master
    This port points to a root bridge located outside the MST region.
- **Status**
  Shows the current status of the interface. The values are only displayed. The parameter depends on the configured protocol.
  - Discarding
    The port receives BPDU frames. Other incoming or outgoing frames are discarded.
  - Listening
    The port receives and sends BPDU frames. The port is involved in the spanning tree algorithm. Other outgoing and incoming frames are discarded.
  - Learning
    The port actively learns the topology; in other words, the node addresses. Other outgoing and incoming frames are discarded.
  - Forwarding
    Following the reconfiguration time, the port is active in the network. The port receives and sends data frames.
- **Oper. Version**
  Shows the compatibility mode of Spanning Tree used by the port.
- **Priority**
  If the path calculated by the spanning tree is possible over several ports of a device, the port with the highest priority (in other words the lowest value for this parameter) is selected. A value between 0 and 240 can be entered for the priority in steps of 16. If you enter a value that cannot be divided by 16, the value is automatically adapted. The default is 128.

- **Path Cost**
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected. If several ports of a device have the same value, the port with the lowest port number is selected.
  If the value in the "Cost Calc" field is "0", the automatically calculated value is displayed. Otherwise, the value of the "Cost Calc" field is displayed.
  The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000
- **Edge Type**
  Shows the type of the connection. The following values are possible:
  - Edge Port
    There is an end device at this port.
  - No Edge Port
    There is a spanning tree or rapid spanning tree device at this port.
- **P.t.P. Type**
  Shows the type of point-to-point link. The following values are possible:
  - P.t.P.
    With half duplex, a point-to-point link is assumed.
  - Shared Media
    With a full duplex connection, a point-to-point link is not assumed.

VRRPv3 statistics

Introduction

This page shows the statistics of the VRRPv3 protocol and all configured virtual routers.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The following fields are displayed:

- VRID Errors
  Shows how many VRRPv3 packets containing an unsupported VRID were received.
- Version Errors
  Shows how many VRRPv3 packets containing an invalid version number were received.
- Checksum Errors
  Shows how many VRRPv3 packets containing an invalid checksum were received.

The table has the following columns:

- Interfaces
  Interface to which the settings relate.
- VRID
  Shows the ID of the virtual router. Valid values are 1 ... 255.
- Type
  Shows the version of the IP protocol.
- Become Master
  Shows how often this virtual router changed to the "Master" status.
- Advertisements Received
  Shows how many VRRPv3 packets were received.
- Advertisement Interval Errors
  Shows how many bad VRRPv3 packets were received whose interval does not match the value set locally.
- IP TTL Errors
  Shows how many bad VRRPv3 packets were received whose TTL (Time to live) value in the IP header is incorrect.
- Prio 0 received
  Shows how many VRRPv3 packets with priority 0 were received. VRRPv3 packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- Prio 0 sent
  Shows how many VRRPv3 packets with priority 0 were sent. Packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- Invalid Type
  Shows how many bad VRRPv3 packets were received whose value in the "Type" field of the IP header is invalid.
- Address List Errors
  Shows how many bad VRRPv3 packets were received whose address list does not match the locally configured list.
- Packet Length Errors
  Shows how many bad VRRPv3 packets were received whose length is not correct.

Security

Overview

This page shows the security settings and the local and external user accounts.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the rights of the logged-on user.

Description of the displayed values

Services

The "Services" list shows the security settings.

- **Telnet Server**
  You configure the setting in "System > Configuration".
  - Enabled: Unencrypted access to the CLI.
  - Disabled: No unencrypted access to the CLI.
- **SSH Server**
  You configure the setting in "System > Configuration".
  - Enabled: Encrypted access to the CLI.
  - Disabled: No encrypted access to the CLI.
- **Web Server**
  You configure the setting in "System > Configuration".
  - HTTP/HTTPS: Access to the WBM is possible with HTTP and HTTPS.
  - HTTPS: Access to the WBM is now only possible with HTTPS.
- **SNMP**
  You can configure setting in "System > SNMP > General".
  - "-" (SNMP disabled)
    Access to device parameters via SNMP is not possible.
  - SNMPv1/v2c/v3
    Access to device parameters is possible with SNMP versions 1, 2c or 3.
  - SNMPv3
    Access to device parameters is possible only with SNMP version 3.

- **Login Authentication**
  You configure the setting in "Security > AAA > General".
  - Local
    The authentication must be made locally on the device.
  - RADIUS
    The authentication must be handled via a RADIUS server.
  - Local and RADIUS
    The authentication is possible both with the users that exist on the device (user name and password) and via a RADIUS server.
    The user is first searched for in the local database. If the user does not exist there, a RADIUS query is sent.
  - RADIUS and fallback local
    The authentication must be handled via a RADIUS server.
    A local authentication is performed only when the RADIUS server cannot be reached in the network.
- **Password Policy**
  Shows which password policy is currently being used.

Local and external user accounts

You configure local user accounts and roles in "Security > User Accounts"

When you create a local user account an external user account is generated automatically. Local user accounts involve users each with a password for logging in on the device.

In the table "External User Accounts" a user is linked to a role, for example the user "Observer" with the role "user" In this case, the user is defined on a RADIUS server. The roll is defined locally on the device. When a RADIUS server authenticates a user, the corresponding group however is unknown or does not exist, the device checks whether or not there is an entry for the user in the table "External User Accounts". If an entry exists, the user is logged in with the rights of the associated role. If the corresponding group is known on the device, both tables are evaluated. The user is assigned the role with the higher rights.

Note

The table "External User Accounts" is only evaluated if you have set "Vendor Specific" in the RADIUS Authorization Mode".

With CLI you can access external user accounts.

The table "Local User Accounts" has the following columns:

- **User Account**
  Shows the name of the local user.
- **Role**
  Shows the role of the user. You can obtain more information on the function rights of the role in "Information > Security > Roles".

Supported Function Rights

The page shows the function rights available locally on the device.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

- **Function Right**
  Shows the number of the function right. Different rights relating to the device parameters are assigned to the numbers.
- **Description**
  Shows the description of the function right.

Roles

The page shows the roles valid locally on the device.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

This table contains the following columns:

- **Role**
  Shows the role.
- **Function Right**
  Shows the function rights of the role:
  - 1
    Users with this role can read device parameters but cannot change them.
  - 15
    Users with this role can both read and change device parameters.
  - 0
    This is a role that the device assigns internally when the user could not be authenticated. The user is denied access to the device.
- **Description**
  Shows a description of the role.

Groups

This page shows which group is linked to which role. The group is defined on a RADIUS server. The roll is defined locally on the device.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Group**
  Shows the name of the group. The name matches the group on the RADIUS server.
- **Role**
  Shows the role. Users who are authenticated with the linked group on the RADIUS server receive the rights of this role locally on the device.
- **Description**
  Shows a a description for the link.

Configuring system functions

Configuration

System configuration

The page contains the configuration overview of the access options of the device.

Specify the services that access the device. With some services, there are further configuration pages on which more detailed settings can be made.

Description

The page contains the configuration overview of the access options of the device

- **Telnet Server**
  Enable or disable the Telnet server service for unencrypted access to the CLI.
- **SSH Server**
  Enable or disable the SSH server service for encrypted access to the CLI.

- **HTTP Services**
  Specify how the WBM is accessed:
  - HTTPS
    Access to the WBM is only possible with HTTPS.
  - HTTP/HTTPS
    Access to the WBM is only possible with HTTP and HTTPS.
  - Redirect HTTP to HTTPS
    Access via HTTP is automatically diverted to HTTPS.
- **SMTP Client**
  Enable or disable the SMTP client. You can configure other settings in "System > SMTP Client".
- **Syslog Client**
  Enable or disable the Syslog client. You can configure other settings in "System > Syslog Client".
- **DCP Server**
  Specify whether or not the device can be accessed with DCP (Discovery and Configuration Protocol):
  - "-" (disabled)
    DCP is disabled. Device parameters can neither be read nor modified.
  - Read/Write
    With DCP, device parameters can be both read and modified.
  - Read Only
    With DCP, device parameters can be read but cannot be modified.
- **Time**
  Select the required setting. The following settings are possible:
  - Manual
    The system time is set manually. You can configure other settings in "System > System Time > Manual Setting".
  - SIMATIC Time
    The system time is set using a SIMATIC time transmitter. You can configure other settings in "System > System Time > SIMATIC Time Client".
  - SNTP Client
    The system time is set via an SNTP server. You can configure other settings in "System > System Time > SNTP Client".
  - NTP Client
    The system time is set via an NTP server. You can configure other settings in "System > System Time > NTP Client".

- **SNMP**
  Select the required protocol. The following settings are possible:
  - "-" (SNMP disabled)
    Access to device parameters via SNMP is not possible.
  - SNMPv1/v2c/v3
    Access to device parameters is possible with SNMP versions 1, 2c or 3. You can configure other settings in "System > SNMP > General".
  - SNMPv3
    Access to device parameters is possible only with SNMP version 3. You can configure other settings in "System > SNMP > General".
- **SNMPv1/v2 Read Only**
  Enable or disable write access to SNMP variables with SNMPv1/v2c.
- **DHCP Client**
  Enable or disable the DHCP client. You can configure other settings in "System > DHCP".
- **SNMPv1 Traps**
  Enable or disable the sending of SNMPv1 traps (alarm frames). You can configure other settings in "System > SNMP > Traps".
- **Configuration Mode**
  Select the required mode. The following modes are possible:
  - Automatic Save
    Automatic backup mode. Approximately 1 minute after the last parameter change or before you restart the device, the configuration is automatically saved.
    In addition to this, the following message appears in the display area "Changes will be saved automatically in x seconds. Click 'Write Startup Config' to save the changes immediately."

    Note

    Interrupting the save

    Saving starts only after the timer in the message has elapsed. How long saving takes depends on the device.
    - Do not switch off the device immediately after the timer has elapsed.
  - Trial
    Trial mode. In Trial mode, although changes are adopted, they are not saved in the configuration file (startup configuration).
    To save changes in the configuration file, use the "Write startup config" button. The display area also shows the message "Trial Mode Active – Press the "Write Startup Config" button to make your settings persistent" as soon as there are unsaved modifications. This message can be seen on every WBM page until the changes made have either been saved or the device has been restarted.

General

Device

This page contains the general device information.

Description

The page contains the following boxes:

- **Current System Time**(Only available online)
  Shows the current system time. The system time is either set by the user or by a time-of-day frame: either SINEC H1 time-of-day frame, NTP or SNTP. (readonly)
- **System Up Time**(only available online)
  Shows the operating time of the device since the last restart. (readonly)
- **Device Type**(only available online)
  Shows the type designation of the device. (readonly)
- **System Name**
  You can enter the name of the device. The entered name is displayed in the selection area. A maximum of 255 characters are possible. The system name is also displayed in the CLI input prompt. The number of characters in the CLI input prompt is limited. The system name is truncated after 16 characters.
- **System Contact**
  You can enter the name of a contact person responsible for managing the device. A maximum of 255 characters are possible.
- **Location**
  You can enter the location of the device. The entered installation location is displayed in the selection area. A maximum of 255 characters are possible.

Note

Permitted characters

The following printable ASCII characters (0x20 to 0x7e) are permitted in the input boxes **"System Name"**, **"System Contact"** and **"System Location"**:

- 0123456789
- A...Z a...z
- !"#$%&'()*+,-./:;<=>?@ [\]_{|}~^`

Coordinates

On this page, you can enter information about the geographic coordinates (latitude, longitude and the height above the ellipsoid according to WGS84).

Getting the coordinates

Use suitable maps for obtaining the geographic coordinates of the device.

The geographic coordinates can also be obtained using a GPS receiver. The geographic coordinates of these devices are normally displayed directly and only need to be entered in the input boxes of this page.

Description

The page contains the following input boxes with a maximum length of 32 characters.

- **"Latitude" input box**
  Enter the north or south latitude for the location of the device.
  E. G. +49° 1´ 31.67" means that the device is located at 49 degrees, 1 arc minute and 31.67 arc seconds north latitude.
  A south latitude is shown by a preceding minus character.
  You can also append the letters N (north latitude) or S (south latitude) to the numeric information (49° 1´ 31.67" N).
- **"Longitude" input box**
  Enter the eastern or western longitude for the location of the device.
  E. G. +8° 20´ 58.73" means that the device is located at 8 degrees, 20 arc minutes and 58.73 arc seconds eastern longitude.
  A western longitude is indicated by a preceding minus sign.
  You can also add the letter E (eastern longitude) or W (western longitude) to the numeric information (8° 20´ 58.73" E).
- **"Height" input box**
  Enter the value of the height above sea level in meters.
  For example, 158 m means that the device is located at a height of 158 m above sea level.
  Heights below sea level (for example the Dead Sea) are indicated by a preceding minus sign.

Load & save

TFTP

Loading and saving data via a TFTP server

On this page, you can configure the TFTP server and the file names. You can also store device data in an external file on a TFTP server or load such data from an external file from the TFTP server to the devices. This means, for example, that you can also load new firmware from a file located on a TFTP server.

On this page, the certificates required to establish a secure VPN connection can also be loaded.

Firmware

The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.

Note

Incompatibility with previous firmware versions with/without PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed.

In this situation, if a PLUG is inserted in the device, following the restart, this has the status "Not Accepted" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data. If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using the page "System > PLUG".

CLI script file

You can download existing CLI configurations (RunningCLI) and upload your own CLI scripts (Script).

Note

The downloadable CLI script is not intended to be uploaded again unchanged.

Description

- **TFTP Server Address**
  Enter the IP address or the FQDN of the TFTP server with which you exchange data.
- **TFTP Server Port**
  Here, enter the port of the TFTP server via which data exchange will be handled. If necessary, you can change the default value 69 to your own requirements.

The table has the following columns:

- **File type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.

- File name
  A file name is preset here for every file type.

  Note
  Changing the file name

  You can change the file name preset in this column. After loading on the device, the changed file name can also be used with the Command Line Interface.

- Action (only available online)
  Select the required action. The selection depends on the selected file type, for example you can only save the log file.
  The following actions are possible:
  - Save file
    With this selection, you save a file on the TFTP server.
  - Load file
    With this selection, you load a file from the TFTP server.

SFTP

Loading and saving data via an SFTP server

SFTP (SSH File Transfer Protocol) transfers the files encrypted. On this page, you configure the access data for the SFTP server.

You can also store device data in an external file on your client PC or load such data from an external file from the PC to the devices. This means, for example, that you can also load new firmware from a file located on your Admin PC.

On this page, the certificates required to establish a secure VPN connection can also be loaded.

Firmware

The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.

CLI script file

You can download existing CLI configurations (RunningCLI) and upload your own CLI scripts (Script).

Note

The downloadable CLI script is not intended to be uploaded again unchanged.

Description

The page contains the following boxes:

- **SFTP Server Address**
  Enter the IP address or the FQDN of the SFTP server with which you exchange data.
- **SFTP Server Port**
  Enter the port of the SFTP server via which data exchange will be handled. If necessary, you can change the default value 22 to your own requirements.
- **SFTP User**
  Enter the user for access to the SFTP server. This assumes that a user with the corresponding rights has been created on the SFTP server.
- **SFTP Password**
  Enter the password for the user
- **SFTP Password Confirmation**
  Confirm the password.

The table has the following columns:

- **Type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.
- **Filename**
  A file name is preset here for every file type.

  Note

  Changing the file name

  You can change the file name preset in this column. After loading on the device, the changed file name can also be used with the Command Line Interface.

- **Actions**
  Select the required action. The selection depends on the selected file type, for example you can only save the log file.
  The following actions are possible:
  - **Save file**
    With this selection, you save a file on the SFTP server.
  - **Load file**
    With this selection, you load a file from the SFTP server.

Passwords

Password for files

There are files to which access is password protected. To load the file on the device, enter the password specified for the file on the page.

Description

- **Type**
  Shows the file type
- **Description**
  Shows the short description of the file type.
- **Enabled**
  When selected, the file is used. Can only be enabled if the password is configured.
- **Password**
  Enter the password for the file.
- **Password Confirmation**
  Confirm the new password.
- **Status** (only available online)
  Shows whether the password matched the file on the device.
  - Valid
    The "Enabled" check box is selected and the password matches the file.
  - Invalid
    The "Enabled" check box is selected but the password does not match the file or no file has been loaded yet.
  - '-'
    The password cannot be evaluated or is not yet being used. The "Enabled" check box is not selected.

Events

Configuration

Selecting system events

On this page, you specify which system events are logged and how.

The following messages are always entered in the event log table and cannot be deselected:

- Changing the admin password
- Starting the device
- Operational status of the device, e.g. whether or not a PLUG is inserted
- Status of errors not yet dealt with

For SCALANCE S615 you can send these messages to a Syslog server as well. To do this enable the "Syslog" check box for the event "System General Logs".

Description

Table 1 has the following columns:

- **All Events**
  Shows that the settings are valid for all events of table 2.
- **E-mail / Trap / Log Table / Syslog / Fault / Digital Out / VPN Tunnel**
  Enable or disable the required type of notification for all events. If "No Change" is selected, the entries of the corresponding column in table 2 remain unchanged.
- **Copy To Table**
  If you click the button, the setting is adopted for all events of table 2.

Table 2 has the following columns:

- **Event**
  The "Event" column contains the following:
  - Cold/Warm Start
    The device was turned on or restarted by the user. In the error memory of the device a new entry is generated with the type of restart performed.
  - Link Change
    This event occurs only when the port status is monitored and has changed, see "System > Fault Monitoring > Link Change".
  - Authentication Failure
    This event occurs when access is attempted with an incorrect password.
  - Power Change
    This event occurs only when power supply lines 1 and 2 are monitored. It indicates that there was a change to line 1 or line 2. See "System > Fault Monitoring > Power Supply".
  - Fault State Change
    The fault status has changed. The fault state can relate to the activated port monitoring, the response of the signaling contact or the power supply monitoring.
  - Security Logs
    An entry is made in the security log if the IPsec method is used for VPN or a SINEMA RC connection is activated.
  - Firewall Logs
    Each time individual firewall rules are applied, this is recorded in the firewall log. To do this, the LOG function must be enabled for the various firewall functions.
  - DDNS Client Logs
    The event occurs when the DDNS client synchronizes the assigned IP address with the hostname registered at the DDNS provider.
  - Digital Input
    The event occurs when the status of the digital input has changed.
  - System Connection Status
    The connection status has changed.
  - System General Logs
    Connection establishment, change to the configuration.
  - Digital In
    The event occurs when the status of the digital input has changed.
  - VPN Tunnel
    The event occurs when the status of VPN (IPsec, OpenVPN, SINEMA RC) has changed.
  - Secure NTP
    The event occurs when the status of the secure NTP server has changed.
- **E-mail**
  The device sends an e-mail. This is only possible if the SMTP server is set up and the "SMTP client" function is enabled.
- **Trap**
  The device sends an SNMP trap. This is only possible if "SNMPv1 Traps" is enabled in "System > Configuration".

- Log Table
  The device writes an entry in the event log table, see "Information > Log Table"
- Syslog
  The device writes an entry to the system log server. This is only possible if the system log server is set up and the "Syslog client" function is enabled.
- Faults
  The device triggers an error. The fault LED lights up
- Digital Out
  Controls the digital output or signals the status change with the "DO" LED.
- VPN Tunnel
  Controls the forwarding of an event to a VPN connection (IPsec, OpenVPN, SINEMA RC). As long as the event is present, the VPN connection is switched to active.

Severity Filters

On this page, you configure the severity for the sending of system event notifications.

Description

The table has the following columns:

- Client Type
  Select the client type for which you want to make settings:
  - E-mail
    Sending system event messages by e-mail.
  - Log Table
    Entry of system events in the log table.
  - Syslog
    Entry of system events in the Syslog file
- Severity
  Select the required level. The following settings are possible:
  - Info
    The messages of all levels are sent or logged.
  - Warning
    The message of this level and the "critical" level are sent or logged.
  - Critical
    Only the messages of this level are sent or logged.

SMTP client

The device provides the option of automatically sending an e-mail if an alarm event occurs (for example to the network administrator). The e-mail contains the identification of the sending device, a description of the cause of the alarm in plain language, and a time stamp. This allows centralized network monitoring to be set up for networks with few nodes based on an E-mail system. When an e-mail event message is received, the WBM can be started by the browser using the identification of the sender to read out further diagnostics information.

On this page, you can configure up to three SMTP servers and the corresponding e-mail addresses.

Description

The page contains the following boxes:

- **SMTP Client**
  Enable or disable the SMTP client.
- **Sender Email Address**
  Enter the name of the sender to be included in the e-mail, for example the device name. This setting applies to all configured SMTP servers.
- **Send Test Mail**
  Send a test e-mail to check your configuration.
- **SMTP Port**
  Enter the port via which your SMTP server can be reached.
  Factory settings: 25
  This setting applies to all configured SMTP servers.
- **SMTP Server Address**
  Enter the IP address, the FQDN (Fully Qualified Domain Name) or the host name of the SMTP server.

This table contains the following columns:

- **Select**
  Select the check box in a row to be deleted.
- **SMTP Server Address**
  Shows the IP address, the FQDN (Fully Qualified Domain Name) or the host name of the SMTP server.
- **Receiver Email Address**
  Enter the e-mail address to which the device sends an e-mail if a fault occurs.

SNMP

General

On this page, you make the basic settings for SNMP. Enable the options according to the function you want to use.

Description

- **SNMP**
  Select the SNMP protocol. The following settings are possible:
  - "-" (disabled)
    SNMP is disabled.
  - SNMPv1/v2c/v3
    SNMPv1/v2c/v3 is supported.

    Note

    Note that SNMP in versions 1 and 2c does not have any security mechanisms.

  - SNMPv3
    Only SNMPv3 is supported.
- **SNMPv1/v2 Read-Only**
  If you enable this option, SNMPv1/v2c can only read the SNMP variables.

  Note

  Community String

  For security reasons, do not use the standard values "public" or "private". Change the community strings following the initial installation.

  The recommended minimum length for community strings is 6 characters.

- **SNMPv1/v2c Read Community String**
  Enter the community string for access of the SNMP protocol.
- **SNMPv1/v2c Read/Write Community String**
  Enter the community string for read and write access of the SNMP protocol.
- **SNMPv1 Traps**
  Enable or disable the sending of SNMPv1 traps (alarm frames). In the "Trap" entry, specify the IP addresses of the devices to which SNMPv1 traps will be sent.
- **SNMPv1/v2c Trap Community String - Trap**
  Enter the community string for sending SNMPv1/v2c messages.
- **SNMPv3 User Migration**
  - **Enabled**
    If the function is enabled, an SNMP engine ID is generated that can be migrated. You can transfer configured SNMPv3 users to a different device.
    If you enable this function and load the configuration of the device on another device, configured SNMPv3 users are retained.
  - **Disabled**
    If the function is disabled, a device-specific SNMP engine ID is generated. To generate the ID, the agent MAC address of the device is used. You cannot transfer this SNMP user configuration to other devices.
    If you load the configuration of the device on another device, all configured SNMPv3 users are deleted.
- **SNMP Engine ID** (only available online)
  Shows the SNMP engine ID.

Traps

If an alarm event occurs, a device can send SNMP traps (alarm frames) to up to ten different management stations at the same time. Traps are only sent if the events specified in "Events" occur.

Note

SNMP traps are sent only when the "SNMPv1 Traps" setting was selected in "SNMP > General" or in "System > Configuration".

Description

- Trap Receiver Address
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the station to which the device sends SNMP traps. You can specify up to ten different recipients.

The table has the following columns:

- Trap Receiver Address
  If necessary, change the IP addresses, the FQDN (Fully Qualified Domain Name) of the stations.
- Trap
  Enable or disable the sending of SNMP traps. Stations that are entered but not selected do not receive SNMP traps.

v3 Groups

Security settings and assigning permissions

SNMP version 3 allows permissions to be assigned, authentication, and encryption at protocol level. The security level and read/write permissions are defined according to groups. The settings automatically apply to every member of a group.

Description

- Group Name
  Enter the name of the group. The maximum length is 32 characters.
- Security Level
  Select the security level (authentication, encryption) valid for the selected group. The available options are as follows:
  - no Auth/no Priv
    No authentication enabled / no encryption enabled.
  - Auth/no Priv
    Authentication enabled / no encryption enabled.
  - Auth/Priv
    Authentication enabled / encryption enabled.

The table has the following columns:

- Group Name
  Shows the defined group names.

  Note

  Once a group name and the security level have been specified, they can no longer be modified after the group is created. If you want to change the group name or the security level , you will need to delete the group and recreate it and reconfigure it with the new name.

- Security Level
  Shows the configured security level.
- Read
  Enable or disable read access for the required group.
- Write
  Enable or disable write access for the required group.

  Note

  For write access to work, you also need to enable read access.

- Persistence
  Shows whether or not the group is assigned to an SNMPv3 user. If the group is not assigned to an SNMPv3 user, no automatic saving is triggered and the configured group is deleted after restarting the device.
  - Yes
    The group is assigned to an SNMPV3 user.
  - No
    The group is not assigned to an SNMPV3 user.

v3 users

User-specific security settings

On the page, you can create new SNMPv3 users and modify or delete existing users. The user-based security model works with the concept of the user name; in other words, a user ID is added to every frame. This user name and the applicable security settings are checked by both the sender and recipient.

Description

- User name
  Enter a freely selectable user name. After you have entered the data, you can no longer modify the name.

The table has the following columns:

- User name
  Shows the created users.
- Group Name
  Select the group to which the user will be assigned.
- Authentication Protocol
  Specify the authentication protocol for which a password will be stored.
  The following settings are available:
  - None
  - MD5
  - SHA
- Encryption Protocol
  Specify whether or not a password should be stored for encryption with the DES algorithm.
  Can only be enabled when an authentication protocol has been selected.
- Authentication Password
  Enter the authentication password. This password must have at least 1 character, the maximum length is 32 characters.

  Note
  Length of the password

  As an important measure to maximize security, we recommend that the password has a minimum length of 6 characters and that it contains special characters, uppercase/ lowercase letters, numbers.

- Confirm authentication password
  Confirm the password by repeating the entry.
- Encryption Password
  Enter your encryption password. This password must have at least 1 character, the maximum length is 32 characters.

  Note
  Length of the password

  As an important measure to maximize security, we recommend that the password has a minimum length of 6 characters and that it contains special characters, uppercase/ lowercase letters and numbers.

- Confirm Encryption Password
  Confirm the encryption password by repeating the entry.
- Persistence
  Shows whether or not the user is assigned to an SNMPv3 group. If the user is not assigned to an SNMPv3 group, no automatic saving is triggered and the configured user is deleted after restarting the device.
  - Yes
    The user is assigned to an SNMPv3 group.
  - No
    The user is not assigned to an SNMPv3 group.

System time

Overview

There are different methods that can be used to set the system time of the device. Only one method can be active at any one time.

If one method is activated, the previously activated method is automatically deactivated.

Manual setting

Manual setting of the system time

On this page, you set the date and time of the system. For this setting to be used, enable "Manual time setting".

Note

The page is only available if there is an online connection to the device.

Description

- Time Manually
  Enable or disable the manual time setting. If you enable the option, the "System Time" input box can be edited.
- System Time
  Enter the date and time in the format "MM/DD/YYYY HH:MM:SS".
- Use PC Time
  Click the button to use the time setting of the PC.
- Last Synchronization Time
  Shows when the last time-of-day synchronization took place.
  If no time-of-day synchronization was possible, the box displays "Date/time not set".

- **Last Synchronization Mechanism**
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame.
  - PTP
    Automatic time-of-day synchronization with PTP
- **Daylight Saving Time**
  Shows whether the daylight saving time changeover is active.
  - active (offset + 1 h)
    The system time was changed to daylight saving time; in other words an hour was added. You can see the current system time at the top right in the selection area of the WBM. The set time continues to be displayed in the "System Time" box.
  - inactive (offset +0 h)
    The current system time is not changed.

SNTP client

There are different methods that can be used to set the system time of the device. Only one method can be active at any one time. If one method is activated, the previously activated method is automatically deactivated.

SNTP (Simple Network Time Protocol) is used for synchronizing the time in the network. The appropriate frames are sent by an SNTP server in the network.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Description

- **SNTP Client**
  Enable or disable automatic time-of-day synchronization using SNTP.
- **Current System Time** (only available online)
  Shows the current date and current normal time received by the device. If you specify a time zone, the time information is adapted accordingly.
- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place.

- **Last Synchronization Mechanism** (only available online)
  Shows how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time.
  The time in the "Current System Time" box is adapted accordingly.
- **SNTP Mode**
  Select the synchronization mode. The following types of synchronization are possible:
  - Poll
    If you select this protocol type, the input boxes "SNTP Server Address", "SNTP Server Port" and "Poll Interval[s]" are displayed to allow further configuration. With this type of synchronization, the device is active and sends a time query to the SNTP server.
  - Listen
    With this type of synchronization, the device is passive and receives SNTP frames that deliver the time of day. For this mode create the following firewall rule manually:
    Action: Accept
    From: vlanX
    To: Device
    Source port: *
    Destination port: 123 (UDP)
- **Poll Interval[s]**
  Here, enter the interval between two-time queries. In this box, you enter the query interval in seconds. Possible values are 16 to 16284 seconds.
- **SMTP Server Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the SNTP server.
- **SNTP Server Port**
  Enter the port of the SNTP server.
  The following ports are possible:
  - 123 (standard port)
  - 1025 to 36564
- **Primary**
  The check mark is set for the SNTP server that you create first. If several SNTP servers have been created, the primary server is queried first.

NTP client

If you require time-of-day synchronization using NTP, you can make the relevant settings here.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Description

- **NTP Client**
  When enabled, the device receives the system time from an NTP server.
- **Secure NTP Client only**
  When enabled, the device receives the system time from a secure NTP server. The setting applies to all server entries.
  To enable the secure NTP client, the parameters for authentication (key ID, hash algorithm, key) must be configured. These entries must be present on the secure NTP server.
- **Current System Time** (only available online)
  Shows the current date and current normal time received by the device. If you specify a time zone, the time information is adapted accordingly.
- **Last Synchronization Time** (only available online)
  Shows when the last time-of-day synchronization took place.
- **Last Synchronization Mechanism** (only available online)
  Shows how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time.
  The time in the "Current System Time" box is adapted accordingly.
- **NTP Server Index**
  Select the index of the NTP server. The server with the lowest index is queried first.
- **NTP Server Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the NTP server.

- **NTP Server Port**
  Enter the port of the NTP server.
  The following ports are possible:
  - 123 (standard port)
  - 1025 to 36564
- **Poll Interval**
  Specify the interval between two-time queries. The greater the interval, the less accurate the time of the device. Possible values are 64 to 2592000 seconds (30 days).
- **Key ID**
  Enter the ID of the authentication key.
- **Hash Algorithm**
  Specify the format for the authentication key.
- **Key**
  Enter the authentication key. The length depends on the hash algorithm.
  - DES: ASCII 8 characters
  - MD5: ASCII 16 – 128 characters
  - SHA1: ASCII 20 – 128 characters

SIMATIC time client

Time setting via SIMATIC time client

On this page, you configure time synchronization with the SIMATIC Time Client.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Description

- **SIMATIC Time Client**
  Select this check box to enable the device as a SIMATIC time client.
- **Current System Time** (only available online)
  Shows the current system time.

- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place.
- **Last Synchronization Mechanism** (only available online)
  Shows how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP

NTP server

On this page, you configure the device as an NTP server. The other devices can call up the current time via this NTP server. This means that the supplied devices are not dependent on a connection to an external time server.

Requirement

To allow NTP frames to be received you need to create the following firewall rule:

Action: Accept

From: vlanX

To:Device

Source port *

Destination port: 123 (UDP)

Description

The page contains the following boxes:

- **NTP Server**
  Enable or disable the service NTP server.

Table 1 has the following columns

- At all interfaces
  Shows that the settings are valid for all interfaces of table 2.
- Listen
  In the drop-down list, select the setting for all interfaces. If "No Change" is selected, the entries of the corresponding column in table 2 remain unchanged.
- Copy to Table
  If you click the button, the setting is adopted for all interfaces of table 2.

Table 2 has the following columns

- Interface
  Via this interface the time is transferred using NTP.
- Listen
  When enabled, the other devices can call up the time via this interface.

Auto logout

On this page, set the times after which there is an automatic logout from WBM or the CLI following user in activity.

If you have been logged out automatically, you will need to log in again.

Note

No automatic logout from the CLI

If the connection is not terminated after the set time, check the setting of the "keepalive" mechanism on the Telnet client.

If the interval is shorter than the configured time, the connection is kept alive although no user data is transferred. You have set, for example, 300 seconds for the automatic logout and the "keepalive" function is set to 120 seconds. In this case, a packet is sent every 120 seconds that keeps the connection up.

- Disable the keepalive mechanism (interval time = 0)
  or
- Set the interval high enough so that the underlying connection is terminated when there is inactivity.

Description

- Web Based Management [s]
  Enter the time in seconds for the automatic logout from WBM. If you enter the value 0, the automatic logout is disabled.
- CLI (TELNET, SSH) [s]
  Enter the time in seconds for the automatic logout from the CLI. If you enter the value 0, the automatic logout is disabled.

Button

Functionality SCALANCE S615

The SET button is used for:

- Restart
- Loading new firmware,
- Resetting to factory settings.

You will find a detailed description of the functions in the device operating instructions.

On this page, the functionality of the button can be restricted.

Description SCALANCE S615

The following functionality is possible:

- **Restart / Restore Factory Defaults**
  When disabled, the SET button cannot be used for a restart or to restore factory defaults.

  CAUTION

  Button function "Restart / Restore Factory Defaults" active during startup

  If you have disabled this function in your configuration, disabling is only valid during operation. When restarting, for example after power down, the function is active until the configuration is loaded so that the device can inadvertently be reset to the factory settings. This may cause unwanted disruption in network operation since the device then needs to be reconfigured. An inserted PLUG is also deleted and returned to the status as shipped.

Functionality SCALANCE SC63x-2C/SC64x-2C

The SET button is used for:

- Resetting to factory settings,
- Defining the fault mask and the LED display.

You will find a detailed description of the functions in the device operating instructions.

On this page, the functionality of the button can be restricted.

Description SCALANCE SC63x-2C/SC64x-2C

The following functionality is possible:

- **Restore Factory Defaults**
  When disabled, the SET button cannot be used to restore factory defaults.

⚠ CAUTION
Button function "Restore Factory Defaults" active during startup If you have disabled this function in your configuration, disabling is only valid during operation. When restarting, for example after power down, the function is active until the configuration is loaded so that the device can inadvertently be reset to the factory settings. This may cause unwanted disruption in network operation since the device then needs to be reconfigured. An inserted PLUG is also deleted and returned to the status as shipped.

- **Set Fault Mask**
  Enable or disable the function "Define fault mask via the LED display" with the SELECT/SET button.

Syslog Client

System event agent

Syslog according to RFC 3164 is used for transferring short, unencrypted text messages over UDP in the IP network. This requires a Syslog server.

Requirements for sending log entries

- The Syslog function is enabled on the device.
- The Syslog function is enabled for the relevant event.
- There is a Syslog server in your network that receives the log entries. Since this is a UDP connection, there is no acknowledgment to the sender.
- The IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server is entered in the device.

Description

The page contains the following boxes:

- **Syslog Client**
  Enable or disable the Syslog function.
- **Syslog Server Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server.

The table contains the following columns:

- Syslog Server Address
  Shows the IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server.
- Server Port
  Enter the port of the Syslog server being used.

Ports

Overview (SC-600)

The page shows the configuration for the data transfer for all ports of the device. You cannot configure anything on this page.

Description

The table has the following columns:

- Port
  Shows the configurable ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- Port Name
  Shows the name of the port.
- Port Type (only with routing)
  Shows the type of the port. The following types are possible:
  - Switch Port VLAN Hybrid
  - Switch Port VLAN Trunk
- Combo Port Media Type (only available online)
  This column contains a value only with combo ports.
  Shows the mode of the combo port:
  - auto
  - rj45
  - sfp
- Status
  Shows whether the port is on or off. Data traffic is possible only over an enabled port.
- Operational State (only available online)
  Displays the current operational status. The operational status depends on the configured "Status" and the "Link". The available options are as follows:
  - Up
    You have configured the status "enabled" for the port and the port has a valid connection to the network.
  - Down
    You have configured the status "disabled" or "Link down" for the port or the port has no connection.

- **Mode** (only available online)
  Shows the transfer parameters of the port.
- **MTU (Maximum Transmission Unit)**
  Shows the packet size.
- **Negotiation**
  Shows whether the automatic configuration is enabled or disabled.
- **Flow Ctrl. Type**
  Shows whether flow control is enabled or disabled for the port.
- **Flow Ctrl.**
  Shows whether or not flow control is working on this port.
- **MAC Address** (only available online)
  Shows the MAC address of the port.
- **Blocked by** (only available online)
  Shows why the port is in the "blocked" status:
  - -
    The port is not blocked.
  - Admin down
    The status "disabled" is configured for the port, see "System > Ports > Configuration".
  - Link down
    The status "enabled" is configured for the port but there is no connection, see "System > Ports > Configuration".
  - Power down
    The status “Link down" is configured for the port, see "System > Ports > Configuration".

Deviating display of the transmission parameters with combo ports

In the connection status "down", the displayed transmission parameters do not match the actual values of the combo port. In the connection status "up", the correct values are displayed.

Initial situation

A pluggable transceiver is plugged into the combo port with the following settings:

- Combo Port Media Type: auto
- Status: enabled
- Link: down

Display of the transmission parameters

With 100 Mbps pluggable transceivers

- Actual response: Mode: 100M HD
- Expected response: Mode: 100M FD

With 1 Gbps pluggable transceivers

- Actual response: Mode: 1G HD
- Expected response: Mode: 1G FD

Configuration (SC-600)

Configuring ports

With this page, you can configure all the ports of the device.

Description

- **Port**
  Select the port to be configured from the drop-down list.
- **Status**
  Specify whether the port is enabled or disabled.
  - enabled
    The port is enabled. Data traffic is possible only over an enabled port.
  - disabled
    The port is disabled but the connection remains.

    Note

    Turn off unused ports.

  - Link down
    The port is disabled and the connection to the partner device is terminated.
- **Port Name**
  Here, enter a name for the port.
- **MAC Address** (only available online)
  Shows the MAC address of the port.
- **Mode Type**
  From this drop-down list, select the transmission speed and the transfer mode of the port. If you set the mode to "Auto negotiation", these parameters are automatically negotiated with the connected end device or network component. This must also be in the "auto negotiation" mode.

  Note

  Before the port and partner port can communicate with each other, the settings must match at both ends.

  Note

  "Mode Type" with combo ports

  To be able to set the "Mode Type" of a combo port, change the "Combo Port Media Type" to "rj45". If "auto" is set for the "Combo Port Media Type" and the RJ-45 port is used, you cannot set the "Mode Type".

- **Mode** (only available online)
  Shows the transmission speed and the transmission mode of the port. The following settings are possible:
  - 10 Mbps full duplex (FD) or half duplex (HD)
  - 100 Mbps full duplex (FD) or half duplex (HD)
  - 1000 Mbps (full duplex)
- **Negotiation**
  Shows whether the automatic configuration of the connection to the partner port is enabled or disabled.

  Note

  Turning flow control on/off with auto negotiation

  Flow control can only be enabled or disabled if the "auto negotiation" function is turned off. The function can be enabled again afterwards.

- **Flow Ctrl. Type**
  Enable or disable flow control for the port.
- **MTU**
  Enter the packet size.
- **Flow Ctrl.**
  Shows whether flow control is working on this port.
- **Port Type**
  Select the type of port from the drop-down list.
  - Switch Port VLAN Hybrid
    The port sends tagged and untagged frames. It is not automatically a member of a VLAN.
  - Switch-Port VLAN Trunk
    The port only sends tagged frames and is automatically a member of all VLANs.

- **Combo Port Media Type**
  Specify the mode of the combo port:
  - auto
    If you select this mode, the pluggable transceiver port has priority.
    As soon as a pluggable transceiver is plugged in, an existing connection at the fixed RJ-45 port is terminated. If no Pluggable transceiver is plugged in, a connection can be established via the fixed RJ-45 port.
  - rj45
    If you select this mode, the fixed RJ-45 port is used regardless of the pluggable transceiver port.
    If a pluggable transceiver is plugged in, it is disabled and the power turned off.
  - sfp
    If you select this mode, the pluggable transceiver port is used regardless of the fixed RJ-45 port.
    If an RJ-45 connection is established, it is terminated because the power of the RJ-45 port is turned off.

  The factory setting for the combo ports is the auto mode.

  Note

  Automatic adaptation due to PROFINET configuration

  When establishing a PROFINET connection, the setting of the combo port media type is adapted automatically:
  - If a pluggable transceiver is configured, the combo port media type will be set to "sfp".
  - If the fixed RJ-45 port is configured, the combo port media type will be set to "rj45".

  So that the automatic adaptation can be made, the combo port media type must be set to "auto".

  Configure the combo port media type accordingly using the WBM or CLI.

- **Operational State** (only available online)
  Displays the current operational status. The operational status depends on the configured "Status" and the "Link". The available options are as follows:
  - Up
    You have configured the status "enabled" for the port and the port has a valid connection to the network.
  - Down
    You have configured the status "disabled" or "Link down" for the port or the port has no connection.
  - not present
    With modular devices, this status is displayed when, for example, no media module is inserted.

- **Blocked by** (only available online)
- Shows why the port is in the "blocked" status:
  - -
    The port is not blocked.
  - Admin down
    The status "disabled" is configured for the port, see "System > Ports > Configuration".
  - Link down
    The status "enabled" is configured for the port but there is no connection, see "System > Ports > Configuration".
  - Power down
    The status “Link down" is configured for the port, see "System > Ports > Configuration".

Changing the port configuration

Click the appropriate box to change the configuration.

Note

Optical ports always work with the full duplex mode and at maximum transmission speed. As a result, the following settings cannot be made for optical ports:

- Automatic configuration
- Transmission speed
- Transmission technique

Note

With various automatic functions, the device prevents or reduces the effect on other ports and priority classes (Class of Service) if a port is overloaded. This can mean that frames are discarded even when flow control is enabled.

Port overload occurs when the device receives more frames than it can send, for example as the result of different transmission speeds.

Fault Monitoring

Power supply

Settings for monitoring the power supply

Configure whether or not the power supply should be monitored by the messaging system. With a redundant power supply, configure the monitoring separately for each individual feed-in line.

A fault is then signaled by the message system when there is no power on a monitored connection (supply 1 or supply 2) or when the applied voltage is too low.

Note

You will find the permitted operating voltage limits in the compact operating instructions of the device.

A fault causes the signaling contact to trigger and the fault LED on the device to light up and, depending on the configuration, can trigger a trap or an entry in the event log table.

Select the check box in front of the connection name you want to monitor to enable or disable the monitoring function.

Link Change

Configuration of fault monitoring of status changes on connections

On this page, you configure whether or not an error message is triggered if there is a status change on a network connection.

If connection monitoring is enabled, an error is signaled

- when there should be a link on a port and this is missing.
- or when there should not be a link on a port and a link is detected.

An error causes the signaling contact to trigger and the fault LED to light up on the device. Depending on the configuration, the error may trigger a trap, an e-mail or an entry in the event log table.

Description of the displayed boxes

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Setting**
  Select the required setting. The available options are as follows:
  - "-" (disabled)
  - Up
  - Down
  - No Change: The setting in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns

- **Port**
  Shows the available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Setting**
  Select the setting. You have the following options:
  - Up
    Error handling is triggered when the port changes to the active status.
    (From "Link down" to "Link up")
  - Down
    Error handling is triggered when the port changes to the inactive status.
    (From "Link up" to "Link down")
  - "-" (disabled)
    The error handling is not triggered.

PLUG

Configuration

Note

If there is no online connection to the device only the check box "Firmware on PLUG" (see below) is available.

NOTICE

Do not remove or insert a C-PLUG / KEY-PLUG during operation!

A PLUG may only be removed or inserted when the device is turned off.

The device checks whether or not a PLUG is inserted at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid PLUG was inserted in the device, the device changes to a defined error state following the restart.

If the device was configured at some time with a PLUG, the device can no longer be used without this PLUG. To be able to use the device again, reset the device to the factory settings.

Information about the configuration of the C-PLUG

This page provides detailed information about the configuration stored on the C-PLUG. It is also possible to reset the PLUG to factory defaults or to load it with new contents.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Description

The table has the following rows:

- **Status**
  Shows the status of the PLUG. The following are possible:
  - ACCEPTED
    There is a PLUG with a valid and suitable configuration in the device.
  - NOT ACCEPTED
    Invalid or incompatible configuration on the inserted PLUG.
  - NOT PRESENT
    There is no C-PLUG or KEY-PLUG inserted in the device.
  - FACTORY
    PLUG is inserted and does not contain a configuration. This status is also displayed when the PLUG was formatted during operation.
  - MISSING
    There is no PLUG inserted. Functions are configured on the device for which a license is required.
- **Device Group**
  Shows the SIMATIC NET product line that used the C-PLUG or KEY-PLUG previously.
- **Device Type**
  Shows the device type within the product line that used the C-PLUG or KEY-PLUG previously.
- **Configuration Revision**
  The version of the configuration structure. This information relates to the configuration options supported by the device and has nothing to do with the concrete hardware configuration. This revision information does not therefore change if you add or remove additional components (modules or extenders), it can, however, change if you update the firmware.

- File System
  Displays the type of file system on the PLUG.
- File System Size [bytes]
  Shows the maximum storage capacity of the file system on the C-PLUG.
- File System Usage
  Displays the storage space in use in the file system of the C-PLUG.
- Firmware on PLUG
  When enabled, the firmware will be stored on the PLUG. This means that automatic firmware updates/downgrades can be made with the PLUG.
- Info String
  Shows additional information about the device that used the PLUG previously, for example, article number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.
  If a PLUG was configured as a PRESET PLUG this is shown here as additional information in the first row.
- Modify PLUG
  Select the required setting. You have the following options for changing the configuration on the C-PLUG or KEY-PLUG:
  - Write current configuration to PLUG
    This option is available only if the status of the PLUG is "NOT ACCEPTED" or FACTORY. The configuration in the internal flash memory of the device is copied to the PLUG.
  - Erase PLUG to factory default
    Deletes all data from the C-PLUG and triggers low-level formatting.

License

Note

The page is only available if there is an online connection to the device.

NOTICE
Do not remove or insert a C-PLUG / KEY-PLUG during operation! A PLUG may only be removed or inserted when the device is turned off. The device checks whether or not a PLUG is present at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid PLUG was inserted in the device, the device changes to a defined error state following the restart. If the device was configured at some time with a PLUG, the device can no longer be used without this PLUG. To be able to use the device again, reset the device to the factory settings.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Information about the license of the KEY-PLUG

A C-PLUG can only store the configuration of a device. In addition to the configuration, a KEY-PLUG also contains a license that enables certain functions of your SIMATIC NET device.

This page provides detailed information about the license on the KEY-PLUG.

Description

- **Status**
  Shows the status of the KEY-PLUG. The following are possible:
  - ACCEPTED
    The KEY-PLUG in the device contains a suitable and valid license.
  - NOT ACCEPTED
    The license of the inserted KEY-PLUG is not valid.
  - NOT PRESENT
    No KEY-PLUG is inserted in the device.
  - MISSING
    There is no KEY-PLUG or a C-PLUG with the status "FACTORY" inserted in the device. Functions are configured on the device for which a license is required.
  - WRONG
    The inserted KEY-PLUG is not suitable for the device.
  - UNKNOWN
    Unknown content of the KEY-PLUG.
  - DEFECTIVE
    The content of the KEY-PLUG contains errors.
- **Article number**
  Shows the article number of the KEY-PLUG. The KEY-PLUG is available for various functional enhancements and for various target systems.

- **Serial Number**
  Shows the serial number of the KEY-PLUG.
- **Info String**
  Shows additional information about the device that used the KEY-PLUG previously, for example, article number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.

Note

When you save the configuration, the information about whether or not a KEY-PLUG was inserted in the device at the time is also saved. This configuration can then only work if a KEY-PLUG with the same article number / license is inserted.

Port diagnostics

SFP diagnostics

On this page, you run independent error diagnostics for each individual SFP port. This test is performed without needing to remove the cable, connect a cable tester or install a loopback module at the other end.

Note

The page is only available if there is an online connection to the device.

Description

The page contains the following boxes:

- **Port**
  Select the required port from the drop-down list.

The values are shown in the following boxes:

- **Name**
  Shows the name of the interface.
- **Model**
  Shows the type of interface.
- **Revision**
  Shows the hardware version of the SFP.
- **Serial**
  Shows the serial number of the SFP.
- **Nominal Bit Rate [Mbps]**
  Shows the nominal bit rate of the interface.

- **Max. Link (50.0/125um) [m]**
  Shows the maximum distance in meters that is possible with this medium.
- **Max. Link (62.5/125um) [m]**
  Shows the maximum distance in meters that is possible with this medium.

The following boxes show the values of the SFP transceiver used in this port:

- **Temperature [°C]**
  Shows the temperature of the interface.
- **Voltage [V]**
  Shows the voltage applied to the interface [V].
- **Current [mA]**
  Shows the current consumption of the interface [mA].
- **Rx Power [µW]/Rx Power [dBm]**
  Shows the receive power of the interface [µW]/[dBm].
- **Tx Power [µW]/Tx Power [dBm]**
  Shows the transmit power of the interface [µW]/[dBm].
- **Range Current**
  Shows the current value.
- **Range Low**
  Shows the lowest value.
- **Range High**
  Shows the highest value.

DNS

DNS client

The DNS server (Domain Name System) assigns a domain name to an IP address so that a device can be uniquely identified.

If this setting is enabled, the device can communicate with a DNS server as a DNS client.

Description

- **DNS client**
  Enable or disable depending on whether the device should operate as a DNS client.
- **DNS server used**
  - learned only
    The device uses only the DNS servers assigned by DHCP.
  - manual only
    The device uses only the manually configured DNS servers. The DNS servers must be connected to the Internet.
  - all
    The device uses all available DNS servers.
- **DNS Server Address**
  Enter the IP address of the DNS server.

The table has the following columns:

- **DNS Server Address**
  Shows the IP address of the DNS server.
- **Origin**
  This shows whether the DNS server was configured manually or was assigned by DHCP.

DNS proxy

The device provides a DNS server for the local network. If you enter the IP address of the device in the local application as a DNS server, then the device answers the DNS requests from its cache.

If the device does not know the IP address for a domain address, it forwards the query to an external DNS server. How long the device keeps a domain address in the cache depends on the host being addressed. In addition to the IP address, a DNS request to an external DNS server also supplies the life span of this information.

Description

The page contains the following boxes:

- **Enable DNS Proxy**
  Enable or disable the proxy of the DNS server.
- **Cache Name Errors (NXDOMAIN)**
  Enable or disable the caching of NXDOMAIN replies. If you enable the option, the domain names that were unknown to the DNS server remain in the cache.

DDNS client

The DDNS (Dynamic Domain Name System) is an Internet service that allows a fixed hostname to be set up as a pseudonym for a dynamically changing IP address.

The DDNS client synchronizes the assigned IP address with the hostname registered at the DDNS provider. This means that the device can always be reached using the same hostname.

Requirement:

- User name and password that gives you the right to use the DDNS service.
- Registered hostname, e.g. example.no-ip.com
- UDP port 53 for DNS is enabled and is not used for NAT.

Description

The table has the following columns:

- Service
  Shows which providers are supported.
- Enabled
  When enabled, the device logs on to the DDNS server.
- Host
  Enter the host name that you have agreed with your DDNS provider for the device, e.g. example.no-ip-com.
- User Name
  Enter the user name with which the device logs on to the DDNS server.
- Password
  Enter the password assigned to the user.
- Password Confirmation
  Confirm the password.

DHCP

DHCP client

If the device is configured as a DHCP client, it starts a DHCP request. As the reply to the query the device receives an IPv4 address from the DHCP server. The server manages an address range from which it assigns IPv4 addresses. It is also possible to configure the server so that the client always receives the same IPv4 address in response to its request.

Description

The page contains the following boxes:

- **DHCP Client Configuration Request (Opt. 66, 67)**
  When enabled, the DHCP client uses the options to download the configuration file (option 67) from the TFTP server (option 66). After the restart, the device uses the data from the configuration file.

  Note

  Configuration file and firmware version

  The configuration file is used to store and read in configuration data within a firmware version, e.g. 4.3. Configuration files created with a firmware version <4.2 cannot be read in to a device with a firmware version 4.3.

  Note

  If a configuration file is downloaded, this can trigger a system restart. If the currently running configuration and the configuration in the downloaded configuration file differ, the system restarts.

  Make sure that the option "DHCP Client Configuration Request (Opt. 66, 67)" is no longer set.

- **DHCP Mode**
  Specify the type of identifier with which the DHCP client logs on with its DHCP server.
  - via MAC Address
    Identification is based on the MAC address.
  - via DHCP Client ID
    Identification is based on a freely defined DHCP client ID.
  - via System Name
    Identification is based on the system name. If the system name is 255 characters long, the last character is not used for identification.
  - via Iaid and Duid
    With this the DHCP client can log on with DHCP servers that support parallel operation of IPv4 and IPv6.
    The identification is via the IAID and the DUID and identifies precisely one IP interface of the device.
    IAID (Interface Association Identifier): At least one IAID is generated for each IP interface The IAID remains unchanged when the DHCP client restarts
    DUID (DHCP Unique Identifier): Uniquely identifies server and clients and applies to all IP interfaces of the device. The DUID remains unchanged when there is a restart. Unless the user changes this.

- **DUID-Type**
  Specify which DUID type will be used. The DUID types are defined in RFC 3315.
  - DUID-LLT
    DUID is based on the link layer address of the interface and a time stamp
  - DUID-EN
    DUID is assigned by the vendor (EN = enterprise number)
  - DUID-LL
    DUID is based on the link layer address of the interface
- **Link-layer Address Plus Time (LLT)**
  The value is based on the link layer address of the interface and a time stamp. The value is regenerated each time the factory settings are restored. When necessary the value can be changed.
- **Vendor Enterprise Number (EN)**
  The value is based on the enterprise number specific to the vendor. The value is regenerated each time the factory settings are restored. When necessary the value can be changed.
- **Link-layer address (LL)**
  The link layer address is based on the MAC address. The value is regenerated each time the factory settings are restored. When necessary the value can be changed.

The table has the following columns:

- **Interface**
  Interface to which the setting relates.
- **DHCP**
  Enable or disable the DHCP client for the relevant interface.
- **IAID Value**
  Value with which the interface (DHCP client) identifies itself with the DHCP server.

DHCP server

You can operate the device as a DHCP server. This allows IP addresses to be assigned automatically to the connected devices. The IP addresses are either distributed dynamically from an address band (pool) you have specified or a specific IP address is assigned to a particular device.

On this page, specify the address band from which the device receives any IP address. You configure the static assignment of the IP addresses in "Static Leases".

Note

Deleting DHCP server bindings

If you deactivate or delete an IPv4 address band or turn the DHCP server off and on again, the DHCP server bindings are deleted see "Information > DHCP Server".

The structure of this page depends on how many VLAN IP interfaces the device has.

Requirement

The connected devices are configured so that they obtain the IP address from a DHCP server.

Devices with one VLAN IP interface

On this page, you specify tIPv4 addresses that are assigned via certain ports.

Description

The page contains the following boxes:

- DHCP Server
  Enable or disable the DHCP server on the device.

  Note

  To avoid conflicts with IPv4 addresses, only one device may be configured as a DHCP server in the network.

- Probe address with ICMP echo before offer
  When selected, the DHCP server checks whether or not the IP address has already been assigned. To do this the DHCP server sends ICMP echo messages (ping) to the IPv4 address. If no reply is received, the IPv4 address is assigned.

  Note

  This check is not made with static assignments.

  Note

  If there are devices in your network on which the echo service is disabled as default, there may be conflicts with the IPv4 addresses. To avoid this, assign these devices an IPv4 address outside the IPv4 address band.

The table has the following columns:

- Pool ID
  Shows the number of the IPv4 address band. If you click the "Create" button, a new row with a unique number is created (pool ID).
- Interface
  Select a VLAN IP interface. The IPv4 addresses are assigned dynamically via this interface. The requirement for the assignment is that the IPv4 address of the interface is located in the subnet of the IPv4 address band. If this is not the case, the interface does not assign any IPv4 addresses.
- Enable
  Specify whether or not this IPv4 address band will be used.

  Note

  If you enable the IPv4 address band, its settings in this and the other DHCP tabs are grayed out and can no longer be edited.

- **Subnet**
  Enter the network address range that will be assigned to the devices. Use the CIDR notation.
- **Lower IP Address**
  Enter the IPv4 address that specifies the start of the dynamic IPv4 address band. The IPv4 address must be within the network address range you configured for "Subnet".
- **Upper IP address**
  Enter the IPv4 address that specifies the end of the dynamic IPv4 address band. The IPv4 address must be within the network address range you configured for "Subnet".
- **Lease Time (sec)**
  Specify for how many seconds the assigned IPv4 address remains valid. When half the period of validity has elapsed. the DHCP client can extend the period of the assigned IPv4 address. When the entire time has elapsed, the DHCP client needs to request a new IPv4 address.

Devices with several VLAN IP interfaces

On this page, specify the address band from which the connected device receives any IP address. You configure the static assignment of the IP addresses in "Static Leases".

DHCP options

On this page you specify which DHCP options the DHCP server supports. The various DHCP options are defined in RFC 2132.

Description

The page contains the following boxes:

- **Pool ID**
  Select the required address band.
- **Option Code**
  Enter the number of the required DHCP option.

  Note

  DHCP options supported

  SCALANCE S615: 1, 2, 3. 4. 5, 6, 42, 66, 67

  SCALANCE SC-600: 1, 3, 6, 12, 66, 67

  The DHCP options 1, 3, 6, 66 and 67 are created automatically when the IPv4 address band is created. With the exception of option 1, the options can be deleted.

The table has the following columns:

- **Select**
  Select the check box in the row to be deleted
- **Pool ID**
  Shows the number of the address band.
- **Option Code**
  Shows the number of the DHCP option.

- **Description**
  Text describing the value of the option.
- **Use Interface IP**
  Specify whether or not the internal IP address of the device will be used.
- **Value**
  Enter the DHCP parameter that is transferred to the DHCP client. The content depends on the DHCP option.

Value	Option name		
1	Subnet Mask	The subnet mask is entered automatically.	Option cannot be deleted.
2	Offset time	Offset time to the coordinated universal time UTC.	Enter the offset time in seconds in hexadecimal format.
3	Router	The IPv4 address for router in the subnet of the DHCP client. If the device itself is the router, the IPv4 address of the interface is used.	You can specify several IPv4 addresses separated by commas.
4	Time server	The IPv4 address of the time server available to the DHCP client.	
5	Name server	The IPv4 address of the name server available to the DHCP client.	
6	DNS Server	The IPv4 address of the DNS server available to the DHCP client. If the device itself is the DNS server, the IPv4 address of the interface is used.	
12	Host name	Enter the host name in the string format.	
42	NTP Server	The IPv4 address of the NTP server available to the DHCP client.	
66	TFTP server	The IPv4 address or the hostname of the TFTP server available to the DHCP client.	Enter the address of the TFTP server.
67	Name of the bootfile	The name of the bootfile that the client downloads from the TFTP server.	Enter the name of the boot file in the string format.

Static Leases

On this page you specify that certain devices will be assigned a certain IP address. The address assignment is made based on the MAC address, the client ID or the DUID.

Description

The page contains the following boxes:

- **Pool ID**
  Select the required address band.
- **Client Identification Method**
  Select the method according to which a client is identified.
  - Ethernet MAC
    Identification is based on the MAC address. Enter the MAC address in "Value". A MAC address consists of six byes separated by hyphens in hexadecimal notation, e.g. 00-ab-1d-df-b4-1d.
  - Client ID
    Identification is based on a freely defined DHCP client ID. Enter the required designation in "Value".
  - DUID
    Identification is based on the DUID and IAID. Enter the required designation in "Value" e.g. 00-00-01-C2-00-01-00-01-00-00-00-72-00-1B-1B-B6-32-9D.
- **Value**
  Enter the required value. The entry depends on the selected identification method of the client.

  Note

  A maximum of 20 entries are possible.

The table has the following columns:

- **Select**
  Select the check box in the row to be deleted.
- **Pool ID**
  Shows the number of the address band.
- **Identification Method**
  Shows the method with which the client identifies itself with the DHCP server.
- **Value**
  Shows the MAC address or client ID or DUID of the client.
- **IP Address**
  Specify the IPv4 address that will be assigned to the client. The IPv4 address must be within the address band.

cRSP/SRS

Note

Common Remote Service Platform (cRSP) / Siemens Remote Service (SRS) is a remote maintenance platform via which remote maintenance access is possible.

To use the platform, additional service contracts are necessary and certain constraints must be kept to. If you are interested in cRSP / SRS, call your local Siemens contact or visit Web page (http://www.industry.siemens.com/topics/global/en/service/remote-service/seiten/home.aspx).

On this page, you configure the access data for the SRS / cRSP acc. to URI syntax. The Uniform Resource Identifier (URI) is defined in RFC 3986.

Description

The page contains the following boxes:

- **Enable DDNS for cRSP / SRS**
  Enable or disable the use of cRSP / SRS.
- **Update Interval**
  Enter the time interval.
- **Validate Server Certificate**
  When enabled, the device checks the validity of the received server certificate.

The table has the following columns:

- **Index**
  The number of the entry.
- **Scheme**
  Identifies the access method and the resource type.
  https: Secure access to a Web page.
- **Authority**
  Contains the address of the destination server
- **Path**
  Contains the target path to the resource. The target path can correspond to a directory name or file name.
- **Query**
  A query can contain parameter values for an application.
- **Frag**.
  Addresses local parts of the resource, e.g. the anchor attribute of a Web page.
- **Status** (only available online)
  Shows the status of the last cRSP / SRS access of the entry.
- **Enabled**
  When enabled, this entry is used.

Proxy Server

On this page, you configure the proxy server that is used by various components, for example SINEMA RC.

Description

- **Proxy Name**
  Enter a name for the proxy server.

The table has the following columns:

- **Name**
  Shows the name of the proxy server.
- **Address**
  Enter the IPv4 address of the proxy server.
- **Type**
  Specify the type of the proxy server.
  - HTTP: Proxy server only for access using HTTP.
  - SOCKS: Universal proxy server
- **Port**
  Enter the port on which the proxy service runs.
- **Auth. Method**
  Specify the authentication method.
  - None
    Without authentication
  - Basic
    Standard authentication. User name and password are sent unencrypted.
  - NTML (NT LAN Manager)
    Authentication according to the NTML standard (Windows user logon)
- **User Name**
  Enter the user name for access to the proxy server.
- **Password**
  Enter the password for access to the proxy server.
- **Password Confirmation**
  Enter the password again to confirm it.

SINEMA RC

On this page, you configure the access to the SINEMA RC Server.

Note

For SCALANCE S615 this function can only be used with a KEY-PLUG.

Description

The page contains the following:

- Enable SINEMA RC
  - Enabled:
    A connection to the configured SINEMA RC Server is established. These boxes cannot be edited.
  - Disabled:
    The boxes can be edited. Any existing connection is terminated.

"Server settings" area

- SINEMA RC Address
  Enter the IPV4 address or the FQDN (Fully Qualified Domain Name) of the SINEMA RC Server.
- SINEMA RC Port
  Enter the port via which the SINEMA RC Server can be reached.

"Server Verification" area

- Verification Type
  - Fingerprint: The identity of the server is verified based on the fingerprint.
  - CA certificate: The identity of the server is verified based on the CA certificate.
- Fingerprint
  Only necessary with the setting "Fingerprint". Enter the fingerprint of the device. The fingerprint is assigned during commissioning of the SINEMA RC Server. Based on the fingerprint, the device checks whether the correct SINEMA RC Server is involved. You will find further information on this in the Operating Instructions of the SINEMA RC Server.
- CA Certificate
  Only necessary with the setting "CA Certificate". Select the CA certificate of the server used to sign the server certificate. Only loaded CA certificates can be selected.

"Device Credentials" area

- Device ID
  Enter the device ID. The device ID is assigned when configuring the device on the SINEMA RC Server. You will find further information on this in the Operating Instructions of the SINEMA RC Server.
- Device Password
  Enter the password with which the device logs on to the SINEMA RC Server. The password is assigned when configuring the device on the SINEMA RC Server. You will find further information on this in the Operating Instructions of the SINEMA RC Server.
- Device Password Confirmation
  Repeat the password to confirm it.

"Optional Settings" area

- Auto Firewall/NAT Rules
  - Enabled
    The firewall and NAT rules are created automatically for the VPN connection. The connections between the configured exported subnets and the subnets that can be reached via the SINEMA RC Server are allowed. The NAT settings are implemented as configured in the SINEMA RC Server.
  - Disabled
    You will need to create the firewall and NAT rules yourself.
- Type of connection
  Specify the type of VPN connection. For more detailed information, refer to the section "VPN connection establishment".
  - Auto
    The device adopts the settings of the SINEMA RC Server. You configure the settings on the SINEMA RC Server in "Remote connections > Devices". You will find further information on this topic in the operating instructions "SINEMA RC Server".
  - Permanent
    The settings of the SINEMA RC Server are ignored. The device establishes a VPN connection to the SINEMA RC Server. The VPN tunnel is established permanently
  - Digital Input
    The settings of the SINEMA RC Server are ignored. If the "Digital In" event occurs, the device attempts to establish a VPN connection to the SINEMA RC Server. This is on condition that the event "Digital Input" is forwarded to the VPN connection. To do this in "System > Events> Configuration" activate "VPN Tunnel" for the "Digital In" event.
- Use Proxy
  Specify whether the connection to the defined SINEMA RC Server is established via a proxy server. Only the proxy servers can be selected that you configured in "System > Proxy Server".
- Autoconfiguration Interval [min]
  Specify the period of time in minutes after which queries are sent to the SINEMA RC Server. With this query, the device checks whether there is a newer firmware file on the SINEMA RC server or whether the configuration data has changed.
  If you enter the value 0, this function is disabled.

Configuring interfaces

Ethernet

Overview (S615)

The page shows the configuration for the data transfer for all ports of the device.

Description

- **Port**
  Shows the configurable ports.
- **Port name**
  Shows the name of the port.
- **Port Type** (only with routing)
  Shows the type of the port. The following types are possible:
  - Router port
  - Switch port VLAN hybrid
  - Switch port VLAN trunk
- **Status**
  Shows whether the port is on or off. Data traffic is possible only over an enabled port.
- **Operational State** (only available online)
  Displays the current operational status. The operational status depends on the configured "Status" and the "Link".
  The available options are as follows:
  - up
    You have configured the status "enabled" for the port and the port has a valid connection to the network.
  - down
    You have configured the status "disabled" or "Link down" for the port or the port has no connection.
- **Link** (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - up
    The port has a valid link to the network, a link integrity signal is being received.
  - down
    The link is down, for example because the connected device is turned off.
- **Mode** (only available online)
  Shows the transfer parameters of the port
- **Negotiation**
  Shows whether the automatic configuration is enabled or disabled.
- **MAC Address** (only available online)
  Shows the MAC address of the port.

Configuration (S615)

On this page, you configure the Ethernet ports of the device.

Description

- **Port**
  Select the port to be configured.
- **Status**
  Specify whether the port is enabled or disabled.
  - enabled
    The port is enabled. Data traffic is possible only over an enabled port.
  - disabled
    The port is disabled but the connection remains.
  - Link down
    The port is disabled and the connection to the partner device is terminated.
- **Port name**
  Enter a name for the port.
- **MAC Address** (only available online)
  Shows the MAC address of the port.
- **Mode Type**
  Shows the transmission speed and the transmission method of the port. If you set the mode to "Auto negotiation", these parameters are automatically negotiated with the connected end device or network component. This must also be in the "auto negotiation" mode.

  Note

  Before the port and partner port can communicate with each other, the settings must match at both ends.

- **Mode** (only available online)
  Shows the transmission speed and the transmission mode of the port. The display depends on the set "Mode Type".
- **Negotiation**
  Shows whether the automatic configuration of the connection to the partner port is enabled or disabled.
- **Port Type** (only with routing)
  Select the type of the port:
  - Router Port
    The port is an IP interface. It does not support layer 2 functions.
  - Switch-Port VLAN Hybrid
    The port sends tagged and untagged frames. It is not automatically a member of a VLAN.
  - Switch-Port VLAN Trunk
    The port only sends tagged frames and is automatically a member of all VLANs.

- Operational State (only available online)
  Displays the current operational status. The operational status depends on the configured "Status" and the "Link".
  The available options are as follows:
  - up
    You have configured the status "enabled" for the port and the port has a valid connection to the network.
  - down
    You have configured the status "disabled" or "Link down" for the port or the port has no connection.
- Link (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - Up
    The port has a valid link to the network, a link integrity signal is being received.
  - Down
    The link is down, for example because the connected device is turned off.

PPP

Overview

The page shows the status of the PPP connection.

Description

This table contains the following columns:

- Interface
  Shows the PPP interface. The entry is a link. If you click on the link, the corresponding configuration page is opened.
- Name
  Shows the name of the PPP interface.
- Type
  Shows the protocol of the PPP connection.

- Operation
  Shows whether the PPP connection is activated or deactivated.
- Status
  Shows the status of the PPP connection.
  - Ready
    The PPP connection can be configured and enabled.
  - connecting
    The PPP connection is configured, enabled and the connection is being established.
  - Connected
    The PPP connection is established.
  - Error
    Error status in which operator intervention is required, e.g. wrong password.

Configuration

On this page, you configure the PPP connection. The point-to-point protocol (PPP) allows the connection of an external ADSL modem to an Ethernet interface and via this then a connection to the Internet. The interface is also called PPP interface.

The device acts as a router and logs in with the user name and password. All connected devices can use the PPP connection.

Description

The page contains the following:

- Interface
  Select the PPP interface to be configured.
- Name
  Shows the name of the PPP interface. You can change the name in "Layer 3 > Subnets".
- Type
  Specify the protocol for the PPP connection.
  - PPPoE (Point-to-Point over Ethernet)
    The PPP data is encapsulated in an Ethernet frame.
- Operation
  Specify whether the PPP connection is activated or deactivated.
- L2 Interface
  Specify the interface via which the PPP connection is established. Only VLANs with a configured subnet can be selected.
- User Name
  Enter the user name. You will receive the user name from the DSL provider.
- Password
  Enter the password. You will receive the password from the DSL provider.
- Password Confirmation
  Repeat the password.

- **Forced Disconnect**
  After a certain time, the DSL provider terminates the connection. Enable this option if you want to shift the forced disconnect of your provider to a specific time of day, for example at night outside normal office hours.
- **Time for Forced Disconnect**
  Specify the time of day to which you want to shift the forced disconnect of the DSL provider. This is only possible if the correct system time is set on the device.
  Input format: HH:MM

Configuring layer 2 functions

Configuration

Configuring layer 2

On this page, you configure passive listening.

Description

- **Passive Listening**
  When enabled the function ensures that the BPDUs from the RSTP network are forwarded transparently and return again. If this was not the case, loops would form at the connection point between RSTP and the ring.

VLAN

General

On this page you specify whether or not the device forwards frames with VLAN tags transparently (IEEE 802.1D/VLAN-unaware mode) or takes VLAN information into account (IEEE 802.1Q/VLAN-aware mode). If the device is in the "802.1Q VLAN Bridge" mode, you can define VLANs and specify the use of the ports .

Note

Changing the agent VLAN ID

If the configuration PC is connected directly to the device via Ethernet and you change the agent VLAN ID, the device is no longer reachable via Ethernet following the change.

Description

The page contains the following boxes:

- **Base Bridge-Mode**
  Select the required mode. The following modes are possible:

  Note

  Changing Base bridge mode

  Note the section "Changing Base bridge mode". This section describes how a change affects the existing configuration.

  – 802.1 D Transparent Bridge
    Sets the mode "VLAN-unaware" for the device. In this mode, VLAN tags are not changed but are forwarded transparently. The VLAN priority is evaluated for CoS. In this mode, you cannot create any VLANs. Only a management VLAN is available: VLAN 1.

  – 802.1 Q VLAN Bridge
    Sets the mode "VLAN-aware" on the device. In this mode, VLAN information is taken into account.

- **VLAN ID**
  Enter the VLAN-ID (a number between 1 and 4094).

The table has the following columns:

- **VLAN ID**
  Shows the VLAN ID. The VLAN ID (a number between 1 and 4094) can only be assigned once when creating a new data record and can then no longer be changed. To make a change, the entire data record must be deleted and created again.

- **Name**
  Enter a name for the VLAN. The name only provides information and has no effect on the configuration.

- **Status**
  Shows the status type of the entry in the internal port filter table. Here, "Static" means that the VLAN was entered statically by the user. The entry GVRP means that the configuration was registered by a GVRP frame. This is, however, only possible if GVRP was enabled for the device.
- **List of ports**
  Specify the use of the port. The following options are available:
  - "-"
  - The port is not a member of the specified VLAN.
    With a new definition, all ports have the identifier "-".
  - M
    The port is a member of the VLAN. Frames sent in this VLAN are forwarded with the corresponding VLAN tag.
  - R
    The port is a member of the VLAN. A GVRP frame is used for the registration.
  - U (upper case)
    The port is an untagged member of the VLAN. Frames sent in this VLAN are forwarded without the VLAN tag. Frames without a VLAN tag are sent from this port.
  - u (lower case)
    The port is an untagged member of the VLAN, but the VLAN is not configured as a port VLAN. Frames sent in this VLAN are forwarded without the VLAN tag.
  - F
    The port is not a member of the specified VLAN and it is not possible for the VLAN to be registered dynamically at this port using GVRP. You can configure other settings in "Layer 2 > VLAN > Port-based VLAN".
  - T
    This option is only displayed and cannot be selected.
    This port is a trunk port making it a member in all VLANs.

Changing Base bridge mode

VLAN-unaware (802.1D transparent bridge) → VLAN-aware (802.1Q VLAN bridge)

If you change the Base Bridge mode from VLAN-unaware to VLAN aware, this has the following effects

- All static and dynamic unicast entries are deleted.

VLAN-aware (802.1Q VLAN bridge) → VLAN-unaware (802.1D transparent bridge)

If you change the Base Bridge mode from VLAN-aware to VLAN-unaware, this has the following effects:

- All VLAN configurations are deleted.
- A management VLAN is created: VLAN 1.
- All static and dynamic unicast entries are deleted.

802.1Q VLAN Bridge: Important rules for VLANs

Make sure you keep to the following rules when configuring and operating your VLANs:

- Frames with the VLAN ID "0" are handled as untagged frames but retain their priority value.
- As default, all ports on the device send frames without a VLAN tag to ensure that the end node can receive these frames.
- With SCALANCE W devices, the VLAN ID "1" is the default on all ports.
- If an end node is connected to a port, outgoing frames should be sent without a tag (static access port). If, however, there is a further switch at this port, the frame should have a tag added (trunk port).

Port-based VLAN

Processing received frames

On this page, you specify the configuration of the port properties for receiving frames.

Description of the displayed boxes

Table 1 has the following columns:

- All ports
  Shows that the settings are valid for all ports of table 2.
- Priority / Port VID / Acceptable Frames / Ingress Filtering
  In the drop-down list, select the setting for all ports. If "No Change" is selected, the entries of the corresponding column in table 2 remain unchanged.
- Transfer to table
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- Port
  Shows the available ports and link aggregations. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- Priority
  Select the priority assigned to untagged frames.
  The CoS priority (Class of Service) used in the VLAN tag. If a frame is received without a tag, it will be assigned this priority. This priority specifies how the frame is further processed compared with other frames. There are a total of eight priorities with values 0 to 7, where 7 represents the highest priority (IEEE 802.1p Port Priority).
- Port VID
  Select the VLAN ID. Only VLAN IDs defined on the "VLAN > General" page can be selected. If a received frame does not have a VLAN tag, it has a tag with the VLAN ID specified here added to it and is sent according to the rules at the port.

- Acceptable Frames
  Specify which types of frames will be accepted. The following alternatives are possible:
  - Tagged Frames Only
    The device discards all untagged frames. Frames tagged with "0" are treated like untagged frames. The device forwards all tagged frames. Otherwise, the forwarding rules apply according to the configuration.
  - All
    The device forwards all frames
- Ingress Filtering
  Specify whether the VID of received frames is evaluated. The following options are available:
  - Enabled
    The VLAN ID of received frames decides whether they are forwarded: To forward a VLAN tagged frame, the receiving port must be a member in the same VLAN. Frames from unknown VLANs are discarded at the receiving port.
  - Disabled
    All frames are forwarded.

Dynamic MAC aging

The device automatically learns the source addresses of the connected nodes.

This information is used to forward data frames to the nodes specifically involved. The network load for the other nodes is reduced.
If a device does not receive a frame whose source address matches a learnt address within a certain time, the learnt address is deleted. This mechanism is known as aging. Aging prevents frames being forwarded incorrectly, for example when an end device (for example a programming device) is connected to a different port. If the check box is not enabled, a device does not delete learned addresses automatically.

Description

- Dynamic MAC Aging
  Enable or disable the function for automatic aging of learned MAC addresses:
- Aging Time[s]
  Enter the time in seconds in steps of 15. After this time, a learned address is deleted if the device does not receive any further frames from this sender address.

Note

Rounding of the values, deviation from desired value

When you enter the Aging Time, note that it is rounded to correct values. If you enter a value that cannot be divided by 15, the value is automatically rounded down.

Spanning Tree

General

This is the basic page for spanning tree. As default, Rapid Spanning Tree is enabled.

Description

The page contains the following boxes:

- **Spanning Tree**
  Enable or disable spanning tree.
- **Protocol Compatibility**
  The following setting is available:
  – RSTP

ST general

The page consists of the following parts.

- The left-hand side of the page shows the configuration of the device.

Description

The page contains the following boxes:

- **Bridge Priority / Root Priority** (only available online)
  Which device becomes the root bridge is decided by the bridge priority. The bridge with the highest priority (in other words, with the lowest value for this parameter) becomes the root bridge. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames. The value for the bridge priority is a whole multiple of 4096 with a range of values from 0 to 32768.
- **Bridge Address** (only available online)/ **Root Address** (only available online)
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Port** (only available online)
  Shows the port via which the switch communicates with the root bridge.
- **Root Cost** (only available online)
  The path costs from this device to the root bridge.

- **Topology Changes** (only available online) **/ Last Topology Change** (only available online)
  The entry for the device shows the number of reconfigurations due to the spanning tree mechanism since the last startup. For the root bridge, the time since the last reconfiguration is displayed as follows:
  - Seconds: Unit "sec" after the number
  - Minutes: Unit min after the number
  - Hours: Unit hr after the number
- **Bridge hello time [s] / Root hello time [s]** (only available online)
  Each bridge sends configuration frames (BPDUs) regularly. The interval between two configuration frames is the "Hello Time".
- **Bridge Forward Delay[s] / Root Forward Delay[s]** (only available online)
  New configuration data is not used immediately by a bridge but only after the period specified in the "Forward Delay" parameter. This ensures that operation is only started with the new topology after all the bridges have the required information.
- **Bridge Max Age[s] / Root Max Age[s]** (only available online)
  If the BPDU is older than the specified "Max Age" it is discarded.
- **Reset Counter** (only available online)
  Click this button to reset the counters on this page.

ST Port

When the page is called, the table displays the current status of the configuration of the port parameters.

To configure them, click the relevant cells in the port table.

Description

Table 1 has the following columns:

- **All ports**
  Shows that the settings are valid for all ports of table 2.
- **Spanning Tree Status**
  In the drop-down list, select the setting for all ports. If "No Change" is selected, the entries of the corresponding column in table 2 remain unchanged.
- **Copy to Table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Spanning Tree Status**
  Specify whether or not the port is integrated in the spanning tree.

  Note

  If you disable the "Spanning Tree Status" option for a port, this may cause the formation of loops. The topology must be kept in mind.

- **Priority**
  Enter the priority of the port. The priority is only evaluated when the path costs are the same. The value must be divisible by 16. If the value that cannot be divided by 16, the value is automatically adapted.
  Range of values: 0 - 240.
  The default is 128.
- **Cost Calc.**
  Enter the path cost calculation. If you enter the value "0" here, the automatically calculated value is displayed in the "Path costs" box.
- **Path Cost** (only available online)
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected as the path. If several ports of a device have the same value for the path costs, the port with the lowest port number is selected.
  If the value in the Cost Calc." is "0", the automatically calculated value is shown. Otherwise, the value of the "Cost Calc." box is displayed.
  The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000

  The values can, however, also be set individually.

- **Status** (only available online)
  Displays the current status of the port. The values are only displayed and cannot be configured. The "Status" parameter depends on the configured protocol. The following values are possible:
  - Disabled
    The port only receives and is not involved in STP, MSTP and RSTP.
  - Discarding
    In the "Discarding" mode, BPDU frames are received. Other incoming or outgoing frames are discarded.
  - Listening

    In this status, BPDUs are both received and sent. The port is involved in the spanning tree algorithm.
  - Learning
    Stage prior to the "Forwarding" status, the port is actively learning the topology (in other words, the node addresses).
  - Forwarding
    Following the reconfiguration time, the port is active in the network; it receives and forwards data frames.
- **Fwd. Trans** (only available online)
  Specifies the number of changes from the "Discarding" status to the "Forwarding" status.
- **Edge Type**
  Specify the type of "edge port". You have the following options:
  - "-"
    Edge port is disabled. The port is treated as a "no Edge Port".
  - Admin
    Select this option when there is always an end device on this port. Otherwise a reconfiguration of the network will be triggered each time a connection is changed.
  - Auto
    Select this option if you want a connected end device to be detected automatically at this port. When the connection is established the first time, the port is treated as a "no Edge Port".
  - Admin/Auto
    Select these options if you operate a combination of both on this port. When the connection is established the first time, the port is treated as an "Edge Port".

- **Edge** (only available online)
  Shows the status of the port.
  - Enabled

    An end device is connected to this port.
  - Disabled
    There is a Spanning Tree or Rapid Spanning Tree device at this port.

  With an end device, a switch can change over the port faster without taking into account spanning tree frames. If a spanning tree frame is received despite this setting, the port automatically changes to the "Disabled" setting.
- **P.t.P. Type**
  Select the required option from the drop-down list. The selection depends on the port that is set.
  - "-"
    Point to point is calculated automatically. If the port is set to half duplex, a point-to-point link is not assumed.
  - P.t.P.

    Even with half duplex, a point-to-point link is assumed.
  - Shared Media
    Even with a full duplex connection, a point-to-point link is not assumed.

    Note

    Point-to-point connection means a direct connection between two devices. A shared media connection is, for example, a connection to a hub.
- **P.t.P.** (only available online)
  - Enabled
    Shows that a point-to-point link exists.
  - Disabled
    Shows that no point-to-point link exists.

LLDP

Identifying the network topology

LLDP (Link Layer Discovery Protocol) is defined in the IEEE 802.AB standard.

LLDP is a method used to discover the network topology. Network components exchange information with their neighbor devices using LLDP.

Network components that support LLDP have an LLDP agent. The LLDP agent sends information about itself and receives information from connected devices at periodic intervals. The received information is stored in the MIB.

Applications

PROFINET uses the LLDP protocol for topology diagnostics. In the default setting, LLDP is enabled for all ports; in other words, LLDP frames are sent and received on all ports.

The information sent is stored on every device with LLDP capability in an LLDP MIB file. Network management systems can access these LLDP MIB files using SNMP and therefore recreate the existing network topology. In this way, an administrator can find out which network components are connected to each other and can localize disruptions.

On this page, you have the option of enabling or disabling sending and/or receiving per port.

Description of the displayed boxes

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Setting**
  Select whether or not the port will send or receive LLDP frames. The following options are available:
  - Rx
    This port can only receive LLDP frames.
  - Tx
    This port can only send LLDP frames.
  - Rx & Tx
    This port can receive and send LLDP frames.
  - "-" (disabled)
    This port can neither receive nor send LLDP frames.

Configuring layer 3 functions for IPv4

Static routes

Static route

On this page you specify the routes via which a data exchange can take place between the various subnets. Dynamic routing protocols are not supported, for example RIP, OSPF.

Description of the displayed boxes

The page contains the following boxes:

- Destination Network
  Enter the network address of the destination that can be reached via this route.
- Subnet Mask
  Enter the corresponding subnet mask.
- Gateway
  Enter the IPv4 address of the gateway via which this network address is reachable.
- Interface
  Select the interface of the route.
- Administrative Distance
  Enter the administrative distance for the route. The administrative distance corresponds to the quality of a connection, for example speed, costs. If there are several equal routes, the route with the lowest value is used.
  If you do not enter anything, "not used" is entered automatically. The administrative distance can be changed later.
  Range of values: 1 - 254 or -1 for "not used". Here, 1 is the value for the best possible route.

The table has the following columns:

- Destination Network
  Shows the network address of the destination.
- Subnet Mask
  Shows the corresponding subnet mask.
- Gateway
  Shows the IPv4 address of the next gateway.
- Interface
  Shows the Interface of the route.
- Administrative Distance
  Enter the administrative distance for the route.
- State (only available online)
  Shows whether or not the route is active.

Subnets

Overview

The page shows the subnets for the selected interface. A subnet always relates to an interface and is created in the "Configuration" tab.

Description

The page contains the following box:

- Interface
  Select the interface on which you want to configure another subnet.

The table has the following columns:

- Interface
  Shows the interface.
- TIA Interface
  Shows the selected TIA interface.
- Interface Name
  Shows the name of the interface.
- MAC Address (only available online)
  Shows the MAC address.
- IP Address
  Shows the IPv4 address of the subnet.
- Subnet Mask
  Shows the subnet mask.
- Address Type
  Shows the address type. The following values are possible:
  - Primary
    The first IPv4 address that was configured on an IPv4 interface.
  - Secondary
    All other IPv4 addresses that were configured on an IPv4 interface.
- IP Assignment Method
  Shows how the IPv4 address is assigned. The following values are possible:
  - Static
    The IPv4 address is static. You enter the settings in "IP Address" and "Subnet Mask".
  - Dynamic (DHCP)
    The device obtains a dynamic IPv4 address from a DHCPv4 server.

- **Address Collision Detection Status** (only available online)
  If new IPv4 addresses become active in the network, the "Address Collision Detection" function checks whether this can result in address collisions. The allows IPv4 addresses that would be assigned twice to be detected.

  Note

  The function does not run a cyclic check.

  This column shows the current status of the function. The following values are possible:

  – Idle
    The interface is not enabled and does not have an IPv4 address.
  – Starting
    This status indicates the start-up phase. In this phase, the device initially sends a query as to whether the planned IPv4 address already exists. If the address is not yet been assigned, the device sends the message that it is using this IP address as of now.
  – Conflict
    The interface is not enabled. The interface is attempting to use an IPv4 address address that has already been assigned.
  – Defending
    The interface uses a unique IPv4 address. Another interface is attempting to use the same IPv4 address.
  – Active
    The interface uses a unique IPv4 address. There are no collisions.
  – Not supported
    The function for detection of address collisions is not supported.
  – Disabled
    The function for detection of address collisions is disabled.
- **MTU**
  Shows the packet size.

Configuration

On this page, you configure the subnet for the interface.

Description

The page contains the following:

- **Interface (Name)**
  Select the interface from the drop-down list.
- **Interface Name**
  Enter the name of the interface.
- **MAC Address** (only available online)
  Displays the MAC address of the selected interface.

- DHCP / Dynamic
  Specify whether the interface obtains the IP address dynamically.
  - DHCP: The VLAN interface obtains the IPv4 address from a DHCP server
  - Dynamic: The PPP interface obtains the IP address from the provider.

  Note

  If you want to operate the device as a router with several interfaces, disable DHCP on all interfaces.

- IP Address
  Enter the IPv4 address of the interface. The IPv4 addresses must not be used more than once.
- Subnet Mask
  Enter the subnet mask of the subnet you are creating. Subnets on different interfaces must not overlap.
- Broadcast IP Address
  If a specific IP address is to be used as the broadcast IP address of the subnet, enter this. Otherwise the last IP address of the subnet will be used.
- Address Type
  Shows the address type. The following values are possible:
  - Primary
    The first subnet of the interface.
  - Secondary
    All further subnets of the interface.
- TIA Interface
  Select whether or not this interface should become the TIA Interface.
  The following conditions apply to the TIA interface:
  - Only interfaces with the address type "Primary" can be enabled as the TIA interface.
  - There must only ever be one TIA interface.
  - There can only be one TIA interface.
  - A TIA interface is always a VLAN interface.
- MTU
  MTU (Maximum Transmission Unit) specifies the maximum size of the packet. If packets are longer than the set MTU, they are fragmented.

NAT

Masquerading

On this page, you enable the rules for IP masquerading.

Description

The table has the following columns:

- **Interface**
  Interface to which the setting relates. Only interfaces with a configured subnet are available.
- **Enable Masquerading**
  When enabled, with each outgoing data packet sent via this interface, the source IP address is replaced by the IP address of the interface.

NAPT

On this page, you configure the port forwarding.

Description

The page contains the following boxes:

- **Source Interface**
  Select the interface on which the queries will arrive.
- **Traffic Type**
  Specify the protocol for which the address assignment is valid.
- **Use Interface IP from Source Interface**
  When enabled, the IP address of the selected interface is used for "Dest IP Address".
- **Destination IP Address**
  Enter the destination IP address. The frames are received at this IP address. Can only be edited if "Use Interface IP from Source Interface" is disabled.
- **Destination Port**
  Enter the destination port. Incoming frames with this port as the destination port are forwarded. If the setting is intended to apply to a port range, enter the range with start port "-" end port, for example 30 - 40.

- Translated Destination IP
  Enter the IP address of the node to which this frame will be forwarded.
- Translated Destination Port
  Enter the number of the port. This is the new destination port to which the incoming frame will be forwarded. If the setting is intended to apply to a port range, enter the range with start port "-" end port, for example 30 - 40.
  If the "Destination Port" and the "Translated Destination Port" are the same, the frames will be forwarded without port translation.

  Note

  If the port is already occupied by a local service, for example Telnet, a warning is displayed. Make sure that you avoid using the following ports:
  - Telnet / SSH: TCP port 23
  - HTTP: TCP port 80
  - HTTPS: TCP port 443
  - SNMP: UDP port 161
  - ISAKMP: UDP port 500
  - IPsec Nat-T: UDP port 4500
  - NTP: UDP port 123

The table has the following columns:

- Source Interface
  Shows the interface from which the packets need to come. Only these packets are considered for port forwarding.
- Traffic Type
  Shows the protocol for which the address assignment applies.
- Interface IP
  Shows whether the IP address of the interface is used.
- Destination IP
  Shows the destination IP address. The frames are received at this IP address.
- Destination Port
  Shows the destination port. Incoming frames with this port as the destination port are forwarded.
- Translated Destination IP
  Shows the IP address of the node to which the packets will be forwarded.
- Translated Destination Port
  Shows the destination port to which the packets are translated.

Source NAT

On this page, you configure the rules for source NAT.

Note

Firewall rules with source NAT

If you create a firewall rule for a source NAT rule use the entry from "Source IP Subnet" in "IP Rules" for the "Source (Range)". And for "Destination (Range)" use the entry from "Destination IP Subnet".

Description

- **Source Interface / Destination Interface**
  Specify the direction of the connection establishment. Only connections established in this specified direction are taken into account.
  The virtual interfaces of VPN connections can also be selected:
  - VLANx: VLANs with configured subnet
  - SINEMA RC: Connection to SINEMA RC Server
  - IPsec:Either all IPsec VPN connections (all) or a specific IPsec VPN connection

  Note

  When you configure a NAT address translation to or from the direction of the VPN tunnel, only the IP addresses involved in the NAT address translation rules can be reached via the VPN tunnel.

- **Source IP Address(es)**
  Specify the source IP addresses for which this source NAT rule is valid. Only the packets that correspond to the addresses entered are taken into account.
  The following entries are possible:
  - IP address: Applies precisely to the specified IP address.
  - IP address range: Applies to a certain IP address range: Start IP address "-" End IP address, e.g. 192.168.100.10 - 192.168.100.20
  - IP subnet: Applies to several IPv4 addresses grouped together to form an IP address range: IP address/number of bits of the network part (CIDR notation)
- **Use Interface IP from Destination Interface**
  When enabled, the IP address of the selected destination interface is used in "Translated Source IP Address".

- **Translated Source IP Address**
  Enter the IP address with which the IP address of the sender is replaced. Can only be edited if "Use Interface IP from Destination Interface" is disabled.
- **Destination IP Address(es)**
  Specify the destination IP addresses for which this source NAT rule is valid. Only the packets whose destination IP address is in the range of entered addresses are taken into account.
  - IP address: Applies precisely to the specified IP address.
  - IP address range: Applies to a certain IP address range: Start IP address "-" End IP address, e.g. 192.168.100.10 - 192.168.100.20
  - IP subnet: Applies to several IPv4 addresses grouped together to form an IP address range: IP address/number of bits of the network part (CIDR notation)

The table has the following columns:

- Source Interface
  Shows the source interface.
- Destination Interface
  Shows the destination interface.
- Source IP Address(es)
  Shows the IP addresses of the senders for which address translation is required.
- Translated Source IP Address
  Shows the IP address with which the IP address of the sender is replaced.
- Destination IP Address(es)
  Shows the IP addresses of the recipients for which address translation is required.

NETMAP

On this page, you specify the rules for NETMAP. NETMAP is static 1:1 mapping of network addresses in which the host part is retained. For more information, refer to the section "NAT and firewall".

Note

Firewall rules with source NAT

If you create a firewall rule for a source NAT rule use the entry from "Source IP Subnet" in "IP Rules" for the "Source (Range)". And for "Destination (Range)" use the entry from "Destination IP Subnet".

Firewall rules with destination NAT

If you create a firewall rule for a destination NAT rule use the entry from "Translated Source IP Subnet" in "IP Rules" for the "Source (Range)". And for "Destination (Range)" use the entry from "Translated Destination IP Subnet".

Description

- **Type**
  Specify the type of address translation.
  - Source: Replacement of the source IP address
  - Aim: Replacement of the destination IP address
- **Source Interface**
  Specify the source interface.
  - VLANx: VLANs with configured subnet
  - SINEMA RC: Connection to SINEMA RC Server
  - IPsec: Either all IPsec VPN connections (all) or a specific IPsec VPN connection
  - ppp2: WAN interface
  - OpenVPN: Either all OpenVPN connections (all) or a specific OpenVPN connection
- **Destination Interface**
  Specify the destination interface.
  - VLANx: VLANs with configured subnet
  - SINEMA RC: Connection to SINEMA RC Server
  - IPsec: Either all IPsec VPN connections (all) or a specific IPsec VPN connection
  - ppp2: WAN interface
  - OpenVPN: Either all OpenVPN connections (all) or a specific OpenVPN connection
- **Source IP Subnet**
  Enter the subnet of the sender.
  The subnet can also be a single PC or another subset of the subnet. Use the CIDR notation.
- **Translated Source IP Subnet**
  Enter the subnet with which the subnet of the sender will be replaced. Can only be edited in the "SourceNAT" settings.
  The subnet can also be a single PC or another subset of the subnet. Use the CIDR notation.
- **Destination IP Subnet**
  Enter the subnet of the recipient.
  The subnet can also be a single PC or another subset of the subnet. Use the CIDR notation.
- **Translated Destination IP Subnet**
  Enter the subnet with which the subnet of the recipient will be replaced. Can only be edited in the "DestinationNAT" settings.
  The subnet can also be a single PC or another subset of the subnet. Use the CIDR notation.

The table has the following columns:

- Type
  Shows the direction of the address translation.
- Source Interface
  Shows the source interface.
- Destination Interface
  Shows the destination interface.

- Source IP Subnet
  Shows the subnet of the sender.
- Translated Source IP Subnet
  Shows the subnet of the sender with which the subnet of the sender is replaced.
- Destination IP Subnet
  Shows the subnet of the recipient.
- Translated Destination IP Subnet
  Shows the subnet of the recipient with which the subnet of the recipient is replaced.

VRRPv3

Router

Introduction

On this page, you create new virtual routers. You can configure other parameters in "Layer 3 (IPv4) > VRRPv3 > Configuration".

Note

- You can use VRRPv3 on VLAN interfaces. Router ports are not supported.

Requirement

For the incoming VRRP packets to be forwarded to the device, you must configure the following firewall rule:

Security > Firewall > IP Protocol:

- Protocol Name: "VRRP"
- Protokollnummer: 112

Security > Firewall > IP Rules:

- Protocol: IPv4
- Action: Accept
- From: <Interface>
- To: Device
- Source (Range): 0.0.0.0/0
- Destination (Range): 224.0.0.18/32
- Services: VRRP

Description

- VRRPv3
  Enable or disable routing using VRRPv3.
- Reply to pings on virtual interfaces
  When enabled, the virtual IPv4 addresses also reply to the ping.
- Track
  Enable or disable VRID tracking. When enabled, all VRRP instances are monitored.
  If the status of a VRRP instance changes to "Initialize", the priority of all VRRP instances is reduced to the value "1".
  If the status of a VRRP instance changes, the original priority of all VRRP instances is restored.
- Interface
  Select the required VLAN interface operating as virtual router.
- VRID
  Enter the ID of the virtual router. This ID defines the group of routers that form a virtual router (VR). In the group, this is the same. It can no longer be used for other groups.

The table has the following columns:

- Interface
  Shows the IPv4 interface that functions as the virtual router.
- VRID
  Shows the ID of the virtual router.
- Virtual MAC Address (only available online)
  Shows the virtual MAC address of the virtual router.
- Primary IP Address
  Shows the numerically lowest IPv4 address in this VLAN. The entry 0.0.0.0 means that the "Primary" address on this VLAN is used. Otherwise all IPv4 addresses configured on this VLAN in the "Layer 3 > Subnets" menu are valid values.
- Router State (only available online)
  Shows the current status of the virtual router. Possible values are:
  - Master
    The router is the master router and handles the routing functionality for all assigned IPv4 addresses.
  - Backup
    The router is the backup router. If the master router fails, the backup router takes over the tasks of the master router.
  - Initialize
    The virtual router has just been turned on. It will soon change to the "Master" or "Backup" state.
- Master IP Address (only available online)
  Shows the IPv4 address of the master router.

- **Priority** (only available online)
  Shows the priority of the virtual router.
  If an IPv4 address is assigned to the VRRP router that is also actually configured on the local IPv4 interface, the value 255 is entered automatically. All other priorities can be distributed freely among the VRRP routers. The higher the priority, the earlier the VRRP router becomes "Master".
- **Advert. Interval**
  Shows the interval at which the master router sends VRRPv3 packets.
- **Preempt**
  Shows the precedence of a router when changing roles between backup and master.
  – yes
    This router has precedence when changing roles.
  – no
    This router does not have precedence when changing roles.

VRRP and DHCP server

If you want to operate a DHCP server on the devices of a VRRP group, the DHCP server must be configured on the master router. Backup routers do not react to DHCP queries. Make sure that the master router is statically configured and that after a failure, becomes the master of the VRRP group again.

Configuration

Introduction

On this page, you configure the virtual router.

Description

The page contains the following:

- **Interface / VRID**
  Select the ID of the virtual router to be configured.
- **Primary Address**
  Select the primary IPv4 address. If the router becomes master router, the router uses this IPv4 address.

  Note

  If you only configure one subnet on this VLAN, no entry is necessary. The entry is then 0.0.0.0.
  If you configure more than one subnet on the VLAN and you want a specific IPv4 address to be used as the source address for VRRP packets, select the IPv4 address. Otherwise, the numerically lowest IPv4 address will be used.

- Master
  If enabled, the numerically lowest IPv4 address is entered for "Associated IP Address". This means that the numerically lowest IPv4 address of the VRRP router is used as the virtual IP address of the virtual master router. The backup routers in this group must disable the option and use the IPv4 address of the router in "Associated IP address".
- Priority
  Enter the priority of this virtual router. Valid values are 1-254.
  If an IPv4 address is assigned to the VRRPv3 router that is also actually configured on the local IPv4 interface, the value 255 is entered automatically. All other priorities can be distributed freely among the VRRPv3 routers. The higher the priority, the earlier the VRRPv3 router becomes "Master".
- Advertisement interval
  Enter the interval in seconds after which a master router sends a VRRPv3 packet again.
- Preempt lower priority Master.
  Allow precedence when changing roles between backup and master based on the selection process.
- VRRP Compatible Mode
  When enabled, the VRRPv3 router also sends and receives VRRPv2 frames in addition to VRRPv3 frames for configured IPv4 addresses. Only necessary when some VRRP routers do not support VRRPv3
- Track ID
  Select a track ID.
- Decrement Priority
  Enter the value by which priority of the VRRP interface will be reduced.
- Current Priority
  Shows the priority of the VRRP interface after the monitored interface has changed to the "down" status.

Addresses Overview

Overview

This page shows which IPv4 addresses the virtual router monitors. Each virtual router can monitor a maximum of 10 IPv4 addresses.

Settings

The table has the following columns:

- Interface
  Shows the Interface that functions as the virtual router.
- VRID
  Shows the ID of this virtual router.

- Number of Addresses
  Shows the number of IPv4 addresses.
- Associated IP Address (1) ...Associated IP Address (10)
  Shows the router IPv4 addresses monitored by this virtual router. If a router takes over the role of master, the routing function is taken over by this router for all these IPv4 addresses.

Addresses Configuration

Creating or changing the monitored IP addresses

On this page, you can create, modify or delete the IPv4 addresses to be monitored.

Settings

The page contains the following:

- Interface / VRID
  Select the ID of the virtual router.
- Associated IP Address
  Enter the IPv4 address that the virtual router will monitor.

The table has the following columns:

- Associated IP Address
  Shows the IPv4 addresses that the virtual router monitors.

Interface Tracking

Introduction

On this page, you configure the monitoring of interfaces.

When the link of a monitored interface changes from "up" to "down", the priority of the assigned VRRP interface is reduced. You configure the value by which the priority is reduced on the page "Layer 3 > VRRP/VRRPv3 > Configuration".

When the link of the interface changes back from "down" to "up", the original priority of the VRRP interface is restored.

Description

The page contains the following boxes:

- Interface
  Select the required interface to be monitored.
- Track ID
  Enter a track ID.

- Track ID
  Select a track ID.
- Track Interface Count
  Enter how many monitored interfaces need to change to the "down" status, before the priority is changed.

The table has the following columns:

- Track ID
  Shows the track ID.
- Interface
  Shows the interface that is being monitored.

Configuring security functions

Users

Local users

Note

The page is only available if there is an online connection to the device.

Local users

On this page, you create local user accounts for WBM and CLI with the corresponding rights.

When you create or delete a local user this change is also made automatically in the table "External User Accounts". If you want to make change explicitly for the internal or external user table, use the CLI commands.

Note

The values displayed depend on the rights of the logged-on user.

Description

The page contains the following:

- User Account
  Enter the name for the user. The name must meet the following conditions:
  - It must be unique.
  - It must be between 1 and 250 characters long.
  - The following characters must not be included: § ? " ; :
  - The following user names are not allowed: admin, service, debug

  Note
  User name cannot be changed
  After creating a user, the user name can no longer be modified.
  If a user name needs to be changed, the user must be deleted and a new user created.

  Note
  User names: admin, service, debug
  When shipped the following user names are predefined in the factory. admin, service, debug
  - admin: You can configure the device with this user name. If you log on the first time or log on after a "Restore Factory Defaults and Restart", you will be prompted to change the predefined password "admin".
  - service, debug: These user names are reserved for service purposes.

- Password Policy
  Shows which password policy is being used.
  - High
    Password length: at least 8 characters, maximum 128 characters
    at least 1 upper case letter
    at least 1 special character
    at least 1 number
  - Low
    Password length: at least 6 characters, maximum 128 characters
- Password
  Enter the password. The strength of the password depends on the set password policy.
- Password Confirmation
  Enter the password again to confirm it.
- Role
  Select a role.
  You can choose between default and self-defined roles, refer to the page "Security > Users > Roles.".

The table contains the following columns:

Note

The preset users as well as logged in users cannot be deleted or changed.

- **User Account**
  Shows the user name.
- **Role**
  Shows the role of the user.
- **Description**
  Displays a description of the user account. The description text can be up to 100 characters long.

Roles

Note

The page is only available if there is an online connection to the device.

Roles

On this page, you create roles that are valid locally on the device.

Note

The values displayed depend on the rights of the logged-in user.

Description of the displayed boxes

The page contains the following:

- **Role Name**
  Enter the name for the role. The name must meet the following conditions:
  – It must be unique.
  – It must be between 1 and 64 characters long.

  Note

  Role name cannot be changed

  After creating a role, the name of the role can no longer be changed.

  If a name of a role needs to be changed, the role must be deleted and a new role created.

This table contains the following columns:

Note
Deleting roles

Predefined roles and assigned roles cannot be deleted or modified.

- **Role**
  Shows the name of the role.
- **Function Right**
  Select the function rights of the role.
  - 1
    Users with this role can read device parameters but cannot change them. Users with this role can change their own password.
  - 15
    Users with this role can both read and change device parameters.

  Note
  Function right cannot be changed

  If you have assigned a role, you can no longer change the function right of the role.

  If you want to change the function right of a role, follow the steps outlined below:
  1. Delete all assigned users.
  2. Change the function right of the role:
  3. Assign the role again.

- **Description**
  Enter a description for the role. With predefined roles a description is displayed. The description text can be up to 100 characters long.

Groups

Note

The page is only available if there is an online connection to the device.

User Groups

On this page you link a group with a role.

In this example the group "Administrators" is linked to the "admin"role: The group is defined on a RADIUS server. The role is defined locally on the device. When a RADIUS server

authenticates a user and assigns the user to the "Administrators" group, this user is given rights of the "admin" role.

Note

The values displayed depend on the rights of the logged-on user.

Description of the displayed boxes

The page contains the following:

- **Group Name**
  Enter the name of the group. The name must match the group on the RADIUS server. The name must meet the following conditions:
  - It must be unique.
  - It must be between 1 and 64 characters long.

The table contains the following columns:

- **Group**
  Shows the name of the group.
- **Role**
  Select a role. Users who are authenticated with the linked group on the RADIUS server receive the rights of this role locally on the device. You can choose between system-defined and self-defined roles, refer to the page "Security > Users > Roles.".
- **Description**
  Enter a description for the link of the group.to a role. The description text can be up to 100 characters long.

AAA

General

Login of network nodes

The designation "AAA" used in the menu stands for "Authentication, Authorization, Accounting". This feature is used to identify and allow network nodes and to make the corresponding services available to them.

On this page, you configure the login.

Description of the displayed boxes

The page contains the following boxes:

Note

To be able to use the login authentication "RADIUS", "Local and RADIUS" or "RADIUS and fallback Local" a RADIUS server must be stored and configured for user authentication.

- **Login Authentication**
  Specify how the login is made:
  - Local
    The authentication must be made locally on the device.
  - RADIUS
    The authentication must be handled via a RADIUS server.
  - Local and RADIUS
    The authentication is possible both with the users that exist on the device (user name and password) and via a RADIUS server. The user is first searched for in the local database. If the user does not exist there, a RADIUS query is sent.
  - RADIUS and fallback Local
    The authentication must be handled via a RADIUS server. A local authentication is performed only when the RADIUS server cannot be reached in the network.

RADIUS client

Authentication over an external server

The concept of RADIUS is based on an external authentication server.

Each row of the table contains access data for one server. In the search order, the primary server is queried first. If the primary server cannot be reached, secondary servers are queried in the order in which they are entered. If no server responds, there is no authentication.

Description of the displayed boxes

The page contains the following boxes:

- **RADIUS Authorization Mode**
  For the login authentication, the RADIUS authorization mode specifies how the rights are assigned to the user with a successful authentication.
  - conventional
    In this mode the user is logged in with administrator rights if the server returns the value "Administrative User" to the device for the attribute "Service Type". In all other cases the user is logged in with read rights.
  - Vendor specific
    In this mode the assignment of rights depends on whether and which group the server returns for the user and whether or not there is an entry for the user in the table "External User Accounts".

The table has the following columns:

- Auth. Server Type
  Select which authentication method the server will be used for.
  - Login
    The server is used only for the login authentication.
  - 802.1X
    The server is used only for the 802.1X authentication.
  - Login & 802.1X
    The server is used for both authentication procedures.
- RADIUS Server Address
  Enter the IPv4 address or the FQDN (Fully Qualified Domain Name) of the RADIUS server.
- Server Port
  Here, enter the input port on the RADIUS server. As default, input port 1812 is set. The range of values is 1 ... 65535.
- Shared Secret
  Enter your access ID. The range of values is 1 ... 128 characters.
- Confirm Shared Secret
  Enter your access ID again as confirmation.
- Max. Retrans
  Here, enter the maximum number of retries for an attempted request. The initial connection attempt is repeated the number of times specified here before another configured RADIUS server is queried or the login counts as having failed. As default 3 retries are set, this means 4 connection attempts. The range of values is 1 ... 5.
- Primary Server
  Specify whether or not this server is the primary server. You can select one of the options "yes" or "no".
- Test
  With this button, you can test whether or not the specified RADIUS server is available. The test is performed once and not repeated cyclically.
- Test Result
  Shows whether or not the RADIUS server is available:
  - Failed, no test packet sent
    The IP address is not reachable.
    The IP address is reachable, the RADIUS server is, however, not running.
  - Reachable, key not accepted
    The IP address is reachable, the RADIUS server does not, however accept the shared secret.
  - Reachable, key accepted
    The IP address is reachable, the RADIUS server accepts the specified shared secret.

  The test result is not automatically updated. To delete the test result click the "Refresh" button.

Certificates

Overview

The loaded files (certificates and keys) are shown on this page. You have the following options for loading files on the device:

- System > Load&Save > HTTP
- System > Load&Save > TFTP
- System > Load&Save > SFTP

Description

- **Type**
  Shows the type of the loaded file.
  - CA Cert
    The CA certificate is signed by a CA (Certification Authority).
  - Machine certificate
  - Key File
  - Remote Cert
    Partner certificate
- **Filename**
  Shows the file name.
- **Status**
  Shows whether the certificate is valid or has already expired.
- **Subject DN**
  Shows the name of the applicant.
- **Issuer DN**
  Shows the name of the certificate issuer.
- **Issue Date**
  Shows the start of the period of validity of the certificate
- **Expiry Date**
  Shows the end of the period of validity of the certificate.
- **Used**
  Shows which function uses the certificate.

Certificates

The format of the certificate is based on X.509, a standard of the ITU-T for creating digital certificates. This standard describes the schematic structure of X509 certificates. You will find further information on this on the Internet at "http://www.itu.int".

On this page, the content of the following structure elements can be displayed. If the structure element does not exist or is not completed in the selected certificate, nothing is shown in the box on the right. Certain entries can only be edited if they are supported.

Description

- **Filename**
  Select the required certificate.
- **Type**
  Shows the type of the loaded file.
  - CA Cert
    The CA certificate is signed by a CA (Certification Authority).
  - Machine certificate
  - Key File
  - Remote Cert
    Partner certificate
- **Subject DN**
  Shows the name of the applicant.
- **Issuer DN**
  Shows the name of the certificate issuer.
- **Subject Alternate Name**
  If it exists, an alternative name of the applicant is displayed.
- **Issue Date**
  Shows the start of the period of validity of the certificate
- **Expiry Date**
  Shows the end of the period of validity of the certificate.
- **Serial Number**
  Shows the serial number of the certificate.
- **Used**
  Shows which function uses the certificate.
- **Crypto Algorithm**
  Shows which cryptographic method is used.
- **Key Usage**
  Shows the purpose that the key belonging to the certificate is used for, e.g. to verify digital signatures.
- **Extended Key Usage**
  Shows whether the purpose is additionally restricted, e.g. only to verify signatures of the CA certificate.
- **Key File**
  Shows the key file.
- **Certificate Revocation List 1st URL**
  Enter the URL with which the revocation list can be called up. Can only be edited if supported by the certificate.
- **Certificate Revocation List 2nd URL**
  Enter an alternative URL. If the revocation list cannot be called up using the 1st URL, the alternative URL is used. Can only be edited if supported by the certificate.

- **Certificate**
  Shows the name of the certificate.
- **Passphrase**
  Enter the password for the certificate. Can only be edited if the encrypted file is password protected.
- **Passphrase Confirmation**
  Enter the password again. Can only be edited if the encrypted file is password protected.

Firewall

General

On this page, you activate the firewall.

Note

Please remember that if you disable the firewall, your internal network is unprotected.

Description

The page contains the following:

- **Activate Firewall**
  When enabled, the firewall is active.
- **TCP Idle Timeout [s]**
  Enter the required time in seconds. If no data exchange takes place, the TCP connection is terminated automatically when this time has elapsed.
  The range of values is 1 to 21474836.
  Default setting: 86400 seconds
- **UDP Idle Timeout [s]**
  Enter the required time in seconds. If no data exchange takes place, the UDP connection is terminated automatically when this time has elapsed.
  The range of values is 1 to 21474836.
  Default setting: 300 seconds
- **ICMP Idle Timeout [s]**
  Enter the required time in seconds. If no data exchange takes place, the ICMP connection is terminated automatically when this time has elapsed.
  The range of values is 1 to 21474836.
  Default setting: 300 seconds

Predefined IPv4 rules

The page contains predefined IP packet filter rules. If you create your own IP packet filter rules, these have a higher priority than the predefined IP packet filter rules.

Here, you can set which services of the device should be reachable from which interface/ subnet.

Description

- Interface
  Interface to which the setting relates. The list of interfaces/subnets is dynamic and is based on the settings from "Layer 3 > Subnet".
  - VLANx: Allows access from the IP subnet to the device.
- Access to the following IPv4 services is permitted:
  - All
    All IPv4 services
  - HTTP
    For access to Web Based Management.
  - HTTPS
    For secure access to Web Based Management.

    Note

    HTTP and HTTPS deactivated

    If you disable HTTP and HTTPS, the WBM of the device can no longer be reached.

    HTTPS disabled

    When you disable HTTPS, you can only access the WBM using HTTP. This assumes that "HTTP & HTTPS" is set in "System > Configuration > HTTP Services". If for example "Redirect HTTP to HTTPS" is set, access via HTTP cannot be redirected to HTTPS. This means that the WBM of the device can no longer be reached.
  - DNS
    DNS queries to the device. Necessary only if the "DNS-Relay" function is enabled on the device.
  - SNMP
    Incoming SNMP connections. Required, for example, to access the SNMP information of the device using a MIB browser.
  - Telnet
    For unencrypted access to the CLI.
  - IPSec VPN
    Allows IKE (Internet Key Exchange) data transfer from the external network to the device. Necessary if an IPsec VPN remote station needs to establish a connection to this device.
  - SSH
    For encrypted access to the CLI.
  - DHCP
    Access to the DHCP server or the DHCP client
  - Ping
    Access to the ping function.
  - System time
    Access to NTP and SNTP

IP Services

On this page, you define the IP services. Using the IP service definitions, you can define firewall rules for specific services. You select a name and assign the service parameters to it. When you configure the IP rules, you simply use this name.

Description

The page contains the following:

- Service Name
  Enter the name of the IP service. The name must be unique.

This table contains the following columns:

- Service Name
  Shows the name of the IP service.
- Transport
  Specify the protocol type.
  - UDP
    The rule applies only to UDP frames.
  - TCP
    The rule applies only to TCP frames.
- Source Port (Range)
  Enter the source port. The rule applies specifically to the specified port.
  - If the rule is intended to apply to a port range, enter the range with start port "-" end port, for example 30 - 40.
  - If the rule is intended to apply to all ports, enter "*".
- Destination Port (Range)
  Enter the destination port. The rule applies specifically to the specified port.
  - If the rule is intended to apply to a port range, enter the range with start port "-" end port, for example 30 - 40.
  - If the rule is intended to apply to all ports, enter "*".

ICMP Services

On this page, you define ICMP services. Using the ICMP service definitions, you can define firewall rules for specific services. You select a name and assign the service parameters to it. When you configure the IP rules, you simply use this name.

Description

The page contains the following:

- Service Name
  Enter a name for the ICMP service. The name must be unique.

This table contains the following columns:

- Service Name
  Shows the name of the ICMP service.
- Protocol
  Shows the version of the ICMP protocol.
- Type
  Specify the ICMP packet type. A few examples are shown below:
  - Destination Unreachable
    IP frame cannot be delivered.
  - Time Exeeded
    Time limit exceeded
  - Echo-Request
    Echo request, better known as ping.
- Code
  The code describes the ICMP packet type in greater detail. The selection depends on the selected ICMP packet type.
  With "Destination Unreachable", for example "Code 1" host cannot be reached.

IP Protocols

On this page, you can configure user-defined protocols, e.g. IGMP for multicast groups. You select a protocol name and assign the service parameters to it. When you configure the IP rules, you simply use this protocol name.

Description

The page contains the following:

- Protocol Name
  Enter a name for the protocol.

The page contains the following check boxes:

- Protocol Name
  Shows the protocol name.
- Protocol Number
  Enter the protocol number, for example 2. You will find list of the protocol numbers on the Internet pages of iana.org

IP Rules

On this page you specify your own IP packet filter rules for the firewall.

The IP packet filer rules set here have priority:

- over the predefined IP packet filter rules (predefined IPv4) and
- over the IP packet filter rules created automatically due to a connection configuration (SINEMA RC).

Description of the displayed boxes

This table contains the following columns:

- **Select**
  Activate the check box in the row to be deleted.
- **Protocol**
  Shows the version of the IP protocol.
- **Action**
  Select how incoming IP packets are handled:
  - "Accept" - The data packets can pass through.
  - "Reject" – The data packets are rejected, and the sender receives a corresponding message.
  - "Drop" – The data packets are discarded without any notification to the sender.
- **From / To**
  Specify the communications direction of the IP rule.
  - VLANx: VLANs with configured subnet
  - Device
  - SINEMA RC: Connection to SINEMA RC Server
  - IPsec: Either all IPsec VPN connections (all) or a specific IPsec VPN connection
  - ppp2: WAN interface
  - OpenVPN: Either all OpenVPN connections (all) or a specific OpenVPN connection
- **Source (Range)**
  Enter the IP address or an IP range that is allowed to send IP packets.
  - If the rule is intended to apply to an IP range, enter the range with start address "-" end address , for example 192.168.100.10 - 192.168.100.20.
  - If the rule is intended to apply to all IP addresses , enter " 0.0.0.0/0".
- **Destination (Range)**
  Enter the IP address or an IP range that is allowed to receive IP packets.
  - If the rule is intended to apply to an IP range, enter the range with start address "-" end address , for example 192.168.100.10 - 192.168.100.20.
  - If the rule is intended to apply to all IP addresses , enter " 0.0.0.0/0".
- **Service**
  Select the service or the protocol name for which this rule is valid.

- **Log**
  Specify whether or not there should be a log entry every time the rule comes into effect and specify the severity of the event.
  The following settings are available:
  - none
    The rule coming into effect is not logged.
  - info / warning / critical
    The rule coming into effect is logged with the selected event severity. The log file is displayed in "Information" > "Log Tables" > "Firewall Log".
- **Precedence**
  Specify the precedence of the rule.

Note

Opening WBM via VPN tunnel

To open the WBM via a VPN tunnel, create the following IP rule:

Action: Accept

From: IPsec

To: Device

Service: HTTPS

IPsec VPN

Overview

In "Security > IPsec VPN" you can configure the parameters for establishing IPsec VPN connections manually.

Alternative configuration via VPN groups

As an alternative, you can configure IPsec connections via VPN groups, see section AUTOHOTSPOT. During the configuration via VPN groups the remote addresses and remote subnets are determined by STEP 7 based on the interface names:

- Interface name "INT": STEP 7 handles the interface as an internal interface.
- Interface name "EXT": STEP 7 handles the interface as an external interface.

When adding devices to VPN groups STEP 7 only creates VPN connections between the external interfaces. Only the local subnets on the internal interfaces are therefore connected together via VPN. Additional VPN connections need to be configured manually.

Points to note when configuring via VPN groups

Compared with SCALANCE S-600 V3 devices, the following special features apply:

- You specify permission to initiate connection establishment with the assignment to VPN groups.
- In the connection-specific VPN settings the device is displayed as a connection partner but cannot be edited. When the device is selected, the interface can be adapted to other partner devices.
- In terms of the VPN mode, the device always behaves as a routing device.
- The properties of a VPN group are adopted in the local properties of VPN group members. Related to the following parameters, the properties of a VPN group can be overwritten locally.
  - Phase 1: Keying Tries, DPD, DPD Period, DPD Timeout
  - Phase 2: Lifebytes, Protocol, Port (Range), Auto Firewall Rules

General

On the page, you configure the basic settings for VPN.

Description

The page contains the following:

- Activate IPsec VPN
  Enable or disable the IPsec protocol for VPN.
- Enforce strict CRL Policy
  When enabled, the validity of the certificates is checked based on the CRL (Certificate Revocation List). The certificate revocation list lists the certificates issued by the certification authority that have lost their validity before the set expiry date. You configure the certificate revocation list to be used on the page "Certificates (Page 686)".
- NAT Keep Alive Time Interval
  Specify the interval at which sign of life frames (keepalives) are sent. If there is a NAT device between two VPN endpoints, when there is inactivity, the connection is deleted from its dynamic NAT table. To prevent this, keepalives are sent.

Remote End

On this page, you configure the counterpart station (VPN end point).

Description

The page contains the following:

- Remote End Name
  Enter the name of the partner station and click "New Entry" to create a new partner station.

This table contains the following columns:

- Name
  Shows the name of the partner.
- Remote Mode
  Specify the role the remote stations will adopt.
  - Roadwarrior
    The reachable remote addresses are entered. The reachable remote subnets are learned from the partner.
  - Standard
    The reachable remote address and the reachable remote subnets are entered permanently.
- Remote Type
  Specify the type of remote station address.
  - Manual
    The address of the partner is known. The device can either establish the VPN connection actively as a VPN client or wait passively for connection establishment by the partner.
  - Any
    Accepts the connection from remote stations with any IP address address. The device can only wait for VPN connections but cannot establish a VPN tunnel as the active partner.
- Remote Address
  Can only be edited with the remote type "Manual".
  - In standard mode, enter the WAN IP address or the DDNS host name of the partner. The network mask is always 32
  - In Roadwarrior mode, you can specify either the address of the partner or enter an IP range from which connections will be accepted.
- Remote Subnet
  - In standard mode, enter the remote subnet of the remote station. Use the CIDR notation.
  - In Roadwarrior mode, the remote address informs the device of its reachable subnets and the device learns them.
- Virtual IP Mode
  Specify whether or not the remote station is offered a virtual IP address.
  The following options are available:
  - User defined IPv4
    The virtual IP address is from the band specified in "Virtual IP".
  - None
    No virtual IP address. The VPN tunnel is established dynamically to the internal IP address of the remote station.
- Virtual IP
  Specify the subnet (CIDR) from which the remote station is offered a virtual IP address.
  Can only be edited if "user defined IPv4" is selected in "Virtual IP Mode".

Connections

On this page, you configure the basic settings for the VPN connection. With these settings, the device (local endpoint) can establish a secure VPN tunnel to the partner. You specify the security settings on the WBM page "Authentication".

Note

If you use "NETMAP"

- only auto firewall rules are supported
- for "Operation" the setting "on demand" cannot be selected.

Note

Several IPsec VPN connections via the same VPN endpoint

If you have created IPsec VPN connections to different remote subnets via the same VPN endpoint, the first VPN connection configured (lowest index) is the main connection (parent).

Via the main connection all other IPsec VPN connections (children) ate created and established. If all VPN tunnels are now established and the main connection is terminated, all child connections are interrupted. When the DPD timeout has elapsed, all IPsec VPN connections are re-established via the main connection.

If only one child connection is terminated, the main connection and the other child connections are retained.

Description

The page contains the following boxes:

- **Connection name**
  Enter a name for the VPN connection and click "Create" to create a new connection.

This table contains the following columns:

- **Select**
  Select the check box in the row to be deleted.
- **Name**
  Shows the name of the VPN connection.

- Operation
  Specify who establishes the VPN connection.
  - Disabled
    The VPN connection is disabled.
  - start
    The device attempts to establish a VPN connection to the partner.
  - wait
    The device waits for the remote station to initiate the connection establishment.
  - on demand
    The VPN connection is established when necessary.
  - start on DI
    If the event "Digital In" occurs the device attempts to establish a VPN connection to the remote station.
    This is on condition that the event "Digital In" is forwarded to the VPN connection. To do this in "System > Events> Configuration" activate "VPN Tunnel" for the "Digital In" event.
  - wait on DI
    If the event "Digital In" occurs, the device waits for the remote station to initiate connection establishment.
    This is on condition that the event "Digital In" is forwarded to the VPN connection. To do this in "System > Events> Configuration" activate "VPN Tunnel" for the "Digital In" event.
- Keying Protocol
  Specify whether IKEv2 or IKEv1 will be used.
- Remote End
  Select the required remote station. Only partners can be configured that have been configured on the "Remote End" WBM page.
- Local Subnet
  Enter the local subnet. Use the CIDR notation. The local network can also be a single PC or another subset of the local network.
- Request Virtual IP
  When enabled, a virtual IP address is requested from the remote station during connection establishment.
- Timeout [sec]
  Only necessary with the “on demand" setting. Enter the interval after which the VPN connection will be terminated. If no packets are sent during this time, the VPN connection is automatically terminated.

Authentication

On this page, you specify how the VPN connection partners authenticate themselves with each other.

Description

This table contains the following columns:

- **Name**
  Shows the name of the VPN connection to which the settings relate.
- **Authentication**
  Select the authentication method. For the VPN connection, it is essential that the partner uses the same authentication method.
  - Disabled
    No authentication method is selected. Connection establishment is not possible.
  - Remote Cert
    The remote certificate is used for authentication. You specify the certificate in "Remote Certificate"
  - CA Cert
    The certificate of the certification authority is used for authentication. You specify the certificate in "CA Certificate". If you want to establish an IPsec connection to SINEMA RC, you need to add the certificate later via the WBM and download it to the device.
  - PSK
    A key is used for authentication. You configure the key in "PSK".
- **CA Certificate**
  Select the certificate. Only loaded certificates can be selected.
- **Local Certificate**
  Select the machine certificate. Only the device certificates generated by the TIA Portal can be selected.
- **Local ID**
  Enter the local ID from the partner certificate. Only when you use the partner certificate can you leave the box empty. The box is automatically filled with the value from the partner certificate.
- **Remote Certificate**
  Select the remote station certificate. Only the partner certificates generated by the TIA Portal can be selected.
- **Remote ID**
  Enter the "Distinguished Name" or "Alternate Name" from the partner certificate. Only when you use the partner certificate can you leave the box empty. The box is automatically filled with the value from the partner certificate.
- **PSK**
  Enter the key.
- **PSK Confirmation**
  Repeat the key.

Phase 1

Phase 1: Encryption agreement and authentication (IKE = Internet Key Exchange)

On this page, you set the parameters for the protocol of the IPsec key management. The key exchange uses the standardized IKE method for which you can set the following protocol parameters.

Description

This table contains the following columns:

- **Name**
  Shows the name of the VPN connection to which the settings relate.
- **Default Ciphers**
  When enabled, a preset list is transferred to the VPN connection partner during connection establishment. The list contains a combination of the three algorithms (Encryption, Authentication, Key Derivation). To establish a VPN connection, the VPN connection partner must support at least one of the combinations. The selection depends on the key exchange method. Further information can be found in the section "IPsec VPN (Page 569)"
- **Encryption**
  For phase 1, select the required encryption algorithm. Can only be selected if "Default Ciphers" is disabled.
  The selection depends on the key exchange method. Further information can be found in the section "IPsec VPN (Page 569)".

  Note

  The AES modes CCM and GCM contain separate mechanisms for authenticating data. If you use a mode AES x CCM for "Encryption", this is also used for authentication. Then only the pseudo random function will be derived from the "Authentication" parameter. So that a VPN connection can be established, all devices need to use the same settings.

- **Authentication**
  Specify the method for calculating the checksum. Can only be selected if "Default Ciphers" is disabled.
  The following methods are supported:
  - MD5
  - SHA1
  - SHA512
  - SHA256
  - SHA384

- **Key derivation**
  Select the required Diffie-Hellmann group (DH) from which a key will be generated. Can only be selected if "Default Ciphers" is disabled.
  The following DH groups are supported:
  - DH group 1
  - DH group 2
  - DH group 5
  - DH group 14
  - DH group 15
  - DH group 16
  - DH group 17
  - DH group 18
- **Keying Tries**
  Enter the number of repetitions for a failed connection establishment. If you enter the value 0, the connection establishment will be attempted endlessly.
- **Lifetime [min]**
  Enter a period in minutes to specify the lifetime of the authentication. When the time has elapsed, the VPN endpoints involved must authenticate themselves with each other again and generate a new key
- **DPD**
  When enabled DPD is used. Using DPD, it is possible to find out whether the VPN connection still exists or whether it has aborted.

  Note

  Sending DPD queries increases the amount of data sent and received. This can lead to increased costs.

- **DPD Period [sec]**
  Enter the period after which DPD requests are sent. These queries test whether or not the remote station is still available
- **DPD Timeout [sec]**
  Enter a period. If there is no response to the DPD queries, the connection to the remote station is declared to be invalid after this time has elapsed.
- **Aggressive Mode**
  - Disabled:
    Main Mode is used.
  - Enabled
    Aggressive Mode is used

  The difference between main and aggressive mode is the "identity protection" used in main mode. The identity is transferred encrypted in main mode but not in aggressive mode.

Phase 2

Phase 2: Data exchange (ESP = Encapsulating Security Payload)

On this page, you set the parameters for the protocol of the IPsec data exchange. The entire communication during this phase is encrypted using the standardized security protocol ESP for which you can set the following protocol parameters.

Description

This table contains the following columns:

- Name
  Shows the name of the VPN connection to which the settings relate.
- Default Ciphers
  When enabled, a preset list is transferred to the VPN connection partner during connection establishment. The list contains a combination of the three algorithms (Encryption, Authentication, Key Derivation). To establish a VPN connection, the VPN connection partner must support at least one of the combinations. Further information can be found in the section "IPsec VPN (Page 569)".
- Encryption
  For phase 2, select the required encryption algorithm. Can only be selected if "Default Ciphers" is disabled.
  Further information can be found in the section "IPsec VPN (Page 569)".

  Note

  The AES modes CCM and GCM contain separate mechanisms for authenticating data. If you use a mode AES x CCM or AES x GCM for "Encryption", this will also be used for authentication. Then only the pseudo random function will be derived from the "Authentication" parameter.

- Authentication
  Specify the method for calculating the checksum. Can only be selected if "Default Ciphers" is disabled.
  The following methods are supported:
  - MD5
  - SHA1
  - SHA512
  - SHA256
  - SHA384

- Key Derivation
  Select the required Diffie-Hellmann group (DH) from which a key will be generated. Can only be selected if "Default Ciphers" is disabled.
  The following DH groups are supported:
  - None: For phase 2, no separate keys are exchanged. This means that Perfect Forward Secrecy (PFS) is disabled.
  - DH group 1
  - DH group 2
  - DH group 5
  - DH group 14
  - DH group 15
  - DH group 16
  - DH group 17
  - DH group 18

  Note

  So that a VPN connection can be established, all devices need to use the same settings or provide compatible key procedures..

- Lifetime [min]
  Enter a period in minutes to specify the lifetime of the agreed keys. When the time expires, the key is renegotiated.
- Lifebytes
  Enter the data limit in bytes that specifies the lifetime of the agreed key. When the data limit is reached, the key is renegotiated.
- Protocol
  Specify the protocol for which the VPN connection is valid e.g. UDP, TCP, ICMP. If the setting is intended to apply to all protocols, enter "*".
- Port (Range)
  Specify the port via which the VPN tunnel can communicate. The setting applies specifically to the specified port
  - If the setting is intended to apply to a port range, enter the range with start port "-" end port, for example 30 - 40.
  - If the setting is intended to apply to all ports, enter "*".

  The setting is only effective for port-based protocols.
- Auto Firewall Rules
  - enabled
    The firewall rules are created automatically for the VPN connection.
  - disabled
    You will need to create the firewall rules yourself.

OpenVPN Client

General

On this page, you activate the OpenVPN client.

Description

The page contains the following:

- Activate OpenVPN Client
  Enable or disable the OpenVPN client.

Connections

On this page, you configure the basic settings for the OpenVPN connection. You specify the security settings on the WBM page "Authentication".

Description

- Connection name
  Enter a unique name for the OpenVPN connection and click "Create" to create a new connection.

This table contains the following columns:

- Select
  Select the check box in the row to be deleted.
- Name
  Shows the name of the OpenVPN connection.
- Operation
  Specify how the connection is established. You will find more detailed information in "VPN connection establishment".
  - start
    The device attempts to establish a VPN connection to the partner.
  - start on DI
    If the event "Digital In" occurs the device attempts to establish a VPN connection to the remote station.
    This is on condition that the event "Digital In" is forwarded to the VPN connection. To do this in "System > Events> Configuration" activate "VPN Tunnel" for the "Digital In" event.
  - Disabled
    The VPN connection is disabled.

- Encryption
  Select the required encryption algorithm.
  - AES-128-CBC (Default)
  - AES-192-CBC
  - AES-256-CBC
  - DES-EDE3
  - BF-CBC
- Authentication
  Specify the method for calculating the checksum.
  - SHA256 (default)
  - SHA384
  - SHA512
  - SHA224
  - SHA1
  - MD5
- LZO Comp.
  - disabled (-)
    The compression is disabled. The server can no longer enable compression.
  - No
    The compression is disabled as default. The server can enable compression
  - Yes
    The compression is enabled as default. The server can disable compression
  - Self-adjusting
    As default, compression is enabled adaptively. Only when there is data that is good for compression, otherwise the compression is disabled for a certain time.
- Auto Firewall Rules
  - Enabled
    The firewall rules are created automatically for the VPN connection.
  - Disabled
    You will need to create the suitable firewall rules yourself.
- Enable NAT
  With this setting, you enable automatic IP masquerading for this interface. The local devices are not directly reachable from the outside, but only via the IP address of the interface. The local devices can, however, connect to the devices downstream from the OpenVPN server. You will find more information on NAT in "NAT"

Remote

On this page, you configure the counterpart station (OpemVPN end point). Per connection, you can specify several OpenVPN partners. The device tries all configured OpenVPN partners one after the other until a connection is successfully established.

Description

The page contains the following:

- Remote Name
  Enter a name for the OpenVPN partner.

This table contains the following columns:

- Name
  Shows the name of the Open VPN partner.
- Connection
  Select the corresponding connection. Only connections can be configured that you configured on the "Connections" page.
- Remote Address
  Enter the WAN IP address or the DNS host name of the OpenVPN partner.
- Port
  Specify the port via which the OpenVPN tunnel can communicate. The setting applies specifically to the specified port.
- Protocol
  Specify the protocol for which the OpenVPN connection will be used.

Authentication

On this page, you specify how the VPN connection partners authenticate themselves with each other.

Description

This table contains the following columns:

- Name
  Shows the name of the VPN connection to which the settings relate.
- Method
  Select the authentication method. For the VPN connection, it is essential that the partner uses the same authentication method.
  - Disabled
    No authentication method is selected. Connection establishment is not possible.
  - Certificates
    Certificates are used for the authentication. The certificates need to be added later via the WBM and downloaded to the device.
  - Username/Password
    The user name/password are used for the authentication. The certificate needs to be added later via the WBM and downloaded to the device.
- CA Certificate
  Select the CA certificate via the WBM and load it onto the device.
- Machine certificate
  Select the machine certificate via the WBM and download it to the device.

- User Name
  Specify the user name.
- Password
  Enter the password.
- Password Confirmation
  Confirm the password.

1.4 Creating configurations

1.4.1 Configuring automation systems

1.4.1.1 Functional description of S7-300/400 CPUs (S7-300, S7-400)

Operating modes (S7-300, S7-400)

Principles of the operating modes of S7-CPUs (S7-300, S7-400)

Operating modes describe the behavior of the CPU. You can query and evaluate the operating modes when programming the STARTUP, when testing the controller and when performing error diagnostics at any time.

S7 CPUs from the various different CPU families such as S7-300, S7-400, ET 200S, ET 200Pro or WinLC (PC-based controller) or embedded controller can have the following operating modes:

- STOP
- STARTUP
- RUN
- HOLD

The CPU can communicate via its interfaces (e.g. MPI interface) in these operating modes.

"STOP" operating mode

In "STOP" operating mode the CPU checks whether all configured modules and all modules used by the standard addressing are available. Provided the result is good, the CPU then sets the I/O to a pre-defined default condition. The user program is not executed when the CPU is in "STOP" operating mode.

See also: STOP mode (Page 709)

"STARTUP" operating mode

A distinction is made in "STARTUP" operating mode between the "warm restart", "hot restart" and "cold restart" start-up modes.

See also: "STARTUP" operating mode (Page 710)

"RUN" operating mode

In "RUN" operating mode the CPU executes the user program, updates the inputs and outputs and processes interrupts and error messages.

See also: RUN mode (Page 718)

"HOLD" operating mode

In "HOLD" operating mode, execution of the user program is halted. The "HOLD" operating mode is used only when using the programming device for testing.

See also: HOLD mode (Page 718)

Other operating modes

If the CPU is not ready for operation, it is in one of two operating modes:

- Deenergized, i.e. the supply voltage is switched off.
- Defective, i.e. a non-recoverable error has occurred.
  Check that the CPU is truly defective. To do this, set the CPU to "STOP" and switch the power switch off and on again. To analyze the error, read the diagnostics buffer when the CPU starts up. If the CPU does not start up, replace it.

Operating mode transitions (S7-300, S7-400)

Overview

The following figure shows the operating modes and the operating mode transitions for the S7-300 CPUs:

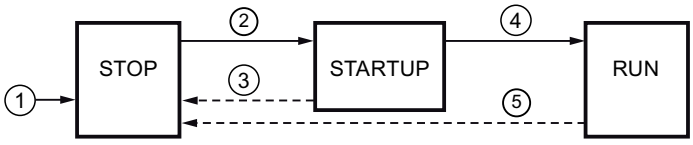

The following table shows the conditions under which the operating modes will change:

No.	Operating mode transition	Conditions
①	STOP	After you turn on the power supply, the CPU is in "STOP" operating mode. It then determines the required type of startup and changes to the next operating mode.
②	STOP → STARTUP	In the following cases the CPU switches to "STARTUP" operating mode: • The CPU is set to "RUN" either by the operating mode selector or from the programming device. • After automatic triggering of a STARTUP operating mode by "POWER-ON". • The communications function "RESUME" or "START" is performed. In the latter two cases the operating mode selector must be set to "RUN".
③	STARTUP → STOP	In the following cases the CPU switches back to STOP operating mode: • An error is detected during the STARTUP. • The CPU is set to "STOP" either by the operating mode selector or from the programming device. • A STOP command is executed in the STARTUP OB. • The communications function "STOP" is performed.

No.	Operating mode transition	Conditions
④	STARTUP → RUN	If the STARTUP is successful, the CPU switches to "RUN".
⑤	RUN → STOP	In the following cases the CPU switches back to STOP operating mode: • An error is detected in RUN operating mode and the associated OB is not loaded. • The CPU is set to "STOP" either by the operating mode selector or from the programming device. • A STOP command is executed in the user program. • The communications function "STOP" is performed.

Priority of the operating modes (S7-300, S7-400)

Function

If several operating mode changes are requested at the same time, the CPU switches to the operating mode with the highest priority. If, for example, the operating mode selector is set to "RUN" and an attempt is made via the programming device to switch the CPU to "STOP", the CPU will switch to "STOP", because this operating mode has the highest priority.

Priority	Operating mode
Highest	STOP (Page 709)
	HOLD (Page 718)
	STARTUP (Page 710)
Lowest	RUN (Page 718)

STOP mode (S7-300, S7-400)

Function

In "STOP" operating mode the user program is not executed. All outputs are set to substitute values and the process being controlled is thereby brought to a safe operating mode. The CPU checks the following points:

- Hardware, for example whether are all modules are available.
- Whether the default settings for the CPU are applicable or parameter sets are present.
- Whether the general conditions for the programmed STARTUP behavior are correct.

In STOP mode you can receive global data. In addition passive one-way communications via the communications system function blocks for configured connections and via communications system functions for unconfigured connections can be performed.

See also

Principles of the operating modes of S7-CPUs (Page 707)

"STARTUP" operating mode (S7-300, S7-400)

Principles of the STARTUP mode (S7-300, S7-400)

Function

After the CPU has been switched on, it executes a STARTUP program before starting to execute the user program. By suitably programming the STARTUP OB, you can use the startup program to specify certain default settings for your cyclic program.

The following STARTUP operating modes are available:

- Warm restart
- Hot restart
- Cold restart

Special characteristics

Note the following points regarding the "STARTUP" operating mode:

- The program in the STARTUP OB will be executed. This means OB 100 for a "warm restart", OB 101 for a "hot restart", and OB 102 for a "cold restart".
- No time-based or interrupt-based program execution can be performed.
- The timers are updated.
- The runtime meter is running.
- The outputs on the modules are disabled.

See also

Principles of the operating modes of S7-CPUs (Page 707)

Hot restart (Page 711)

Cold restart (Page 710)

Startup organization blocks (OB 100, OB 101 and OB 102) (Page 867)

Complete overview of the startup (Page 716)

Cold restart (S7-300, S7-400)

Cold restart

During a cold restart, all data (process image, bit memory, timers, counters and data blocks) are reset to the start values stored in the program (load memory), irrespective of whether they were configured as retentive or non-retentive.

Program execution is restarted at the beginning (startup OB or OB 1).

Warm restart (S7-300, S7-400)

Permitted warm restart

You can perform a warm restart in the following cases:

- After a memory reset.
- After loading the user program in CPU STOP operating mode.

You cannot run a warm restart if the system has requested a memory reset.

Manual restart (warm restart)

A manual restart ("warm restart") is triggered by:

- The operating mode selector.
- A menu command from the programming device or via a communications function, if the operating mode selector is set to RUN.

Automatic restart (warm restart)

An automatic restart ("warm restart") can by triggered by "POWER ON" in the following cases:

- When the "POWER OFF" occurred, the CPU was not in "STOP" operating mode.
- The operating mode selector is set to "RUN".
- The CPU was interrupted by a power failure during a warm restart. This is irrespective of the assigned STARTUP operating mode.

Automatic restart (warm restart) without battery backup

If your CPU is running without a backup battery, e.g. if maintenance-free operation is required, after switching on or on restoration of power after "POWER OFF" the CPU is automatically subjected to a memory reset and then a "warm restart". The user program must exist on flash EPROM (memory card) so that the user program can be copied to work memory again.

Hot restart (S7-300, S7-400)

Hot restart

This start-up mode is only available with S7-400. The features of hot restart are as follows:

- When a hot restart is performed, all data and the process image retain their last valid value.
- Program execution is resumed at the breakpoint.
- The outputs do not change their status until the current cycle is completed.
- After a power supply interruption, the hot restart function is only available in backup mode.

S7-400 CPUs execute an initialization routine and then automatically a hot restart. During a hot restart, the user program is continued at the location at which the processing was

interrupted. The part of the user program that was no longer processed before the power failure is referred to as remaining cycle. The remaining cycle can also contain time and interrupt controlled program parts.

As a rule a hot restart is only permissible if the user program was not modified in STOP state (for example, by reloading a modified block), or if a warm restart is not required for other reasons. It is necessary to distinguish between a manual and an automatic restart.

Manual hot restart

A manual hot restart is only possible if the appropriate parameters are set for the CPU and after the following STOP causes:

- The mode selector was changed from RUN to STOP
- By user programmed STOPs, STOPs after calling of OBs that are not loaded
- STOP state was caused by the programming device or by communication functions

A manual restart can be triggered via menu command from programming device or via communication functions (if the mode selector is on RUN or RUN-P). A requirement for this is that the restart is not disabled by operation (startup parameter).

- If the parameter set of the CPU's manual restart was configured

Automatic restart

An automatic restart was triggered by POWER ON if the CPU was not in STOP or HOLD at POWER OFF. A requirement for this is that the restart is not disabled by operation (startup parameter).

Retentive behavior after loss of power (S7-300, S7-400)

Retentive data areas after power failure

To avoid loss of data in the event of a power failure, you can specify the retentivity of bit memory, timers, counters and areas in data blocks. Providing that the STARTUP operating mode is "warm restart" (default), an "automatic restart with memory" will be performed when power is restored.

S7-300 and S7-400 CPUs react differently to power return after power failure.

S7-300 CPUs know only the "warm restart" restart mode To avoid loss of data in the event of a power failure, you can specify with STEP 7 the retentivity of bit memory, timers, counters and areas in data blocks. An "Automatic warm restart with memory" is executed when power returns.

Depending on the set parameters, S7-400 CPUs react to power return with either a warm restart (after buffered on unbuffered POWER ON) or with hot restart (only possible after buffered POWER ON).

Retentive behavior for CPUs with battery back-up

The following table shows the retentive behavior in the work memory of the S7-300 and S7-400 CPUs with buffering at "warm restart" and "cold restart":

Data	Warm restart	Cold restart	Hot restart
Blocks in the load memory	X	X	X
DB in the work memory	X	0	X
Bit memory, timers, counters configured as retentive	X	0	X
Bit memory, timers, counters, configured as non-retentive	0	0	X
X: Data is retained 0: Data is reset or cleared (content of DBs)			

Retentive behavior for CPUs without battery back-up

No hot restart is permitted with S7-400 CPUs!

The following table shows the retentive behavior in the work memory of the S7-300 CPUs without battery backup at "warm restart" and "cold restart":

Data	Warm restart		Cold restart	
	S7-300	S7-400	S7-300	S7-400
Blocks in the load memory	VC	VC	VC	VC
DB in the work memory configured as retentive	VX	not possible	V	not possible
DB in the work memory configured as non-retentive	V	V	V	V
Bit memory, timers, counters configured as retentive	X	0	0	0
Bit memory, timers, counters, configured as non-retentive	0	0	0	0
X: Data is retained 0: Data is reset or cleared (content of DBs) V: Data is set to the default value from the EPROM VC: Code block in the EPROM is retained, any overloaded code blocks are lost VX: A data block is only retained if it is available in the load EPROM; retentive data is transferred the from NV-RAM (data blocks loaded or generated in RAM are lost)				

Startup activities (S7-300, S7-400)

Overview

The following table shows which activities the CPU performs at STARTUP:

Activities in execution sequence	At warm restart	At cold restart	At hot restart
Clear the interrupt and block stack	X	X	0
Clear non-retentive bit memory, timers, counters	X	0	0
Clear all bit memory, timers, counters	0	X	0
Clear the process image output	X	X	Can be set
Reset the signal module outputs	X	X	Can be set
Discard the process interrupts	X	X	0
Discard the time-delay interrupts	X	X	0
Discard the diagnostics interrupts	X	X	X
Update the system status list (SSL)	X	X	X
Evaluate the module parameters and transfer them to modules or transfer default values	X	X	X
Execute the respective STARTUP OB	X	X	X
Execute the outstanding cycle (part of the user program that because of the "POWER OFF" was not executed	0	0	X
Update the process image input	X	X	X
Enable digital outputs (cancel the OD (Output Disabled) signal) after operating mode transition to RUN	X	X	X
X: Activity is performed 0: Activity is not performed			

See also

Complete overview of the startup (Page 716)

Cancellation of a startup (S7-300, S7-400)

Reasons for cancellation

If errors occur during startup, the startup is canceled and the CPU switches to or remains in "STOP" operating mode. This has the following effects on the "warm restart" operating mode:

- A "warm restart" that is canceled must be repeated.
- Both a warm restart and a hot restart are possible in the event of a canceled hot restart.
- Under the following conditions a warm restart or hot restart will not be executed or will be canceled:
  - If the CPU operating mode selector is set to the "STOP" operating mode.
  - If a memory reset is requested.
  - If a memory card whose application code is not permissible is plugged in.
  - If more than one CPU is plugged in in single processor operation.
  - If the user program contains an organization block that the CPU does not recognize or has been disabled.
  - If, after the power is switched on, the CPU detects that not all modules that are listed in the configuration created with STEP 7 are actually plugged in (target-actual difference in configuration is not permitted).
  - If errors occur during the assignment of the module parameters.
- Under the following conditions, a hot restart will not be performed or will be canceled:
  - If the CPU memory was previously reset (after the memory reset, only a warm restart is permitted).
  - If the interruption limit is exceeded (the interruption time is the time that expires after leaving RUN mode until the startup OB incl. remaining cycle time was processed).
  - If the module configuration was changed (e.g. module exchange).
  - If the only warm restart configuration is permitted.
  - If blocks were loaded, deleted or changed in STOP mode.

Complete overview of the startup (S7-300, S7-400)

Sequence

The following figure shows the activities of the CPU in "STOP", "STARTUP", and "RUN" operating modes.

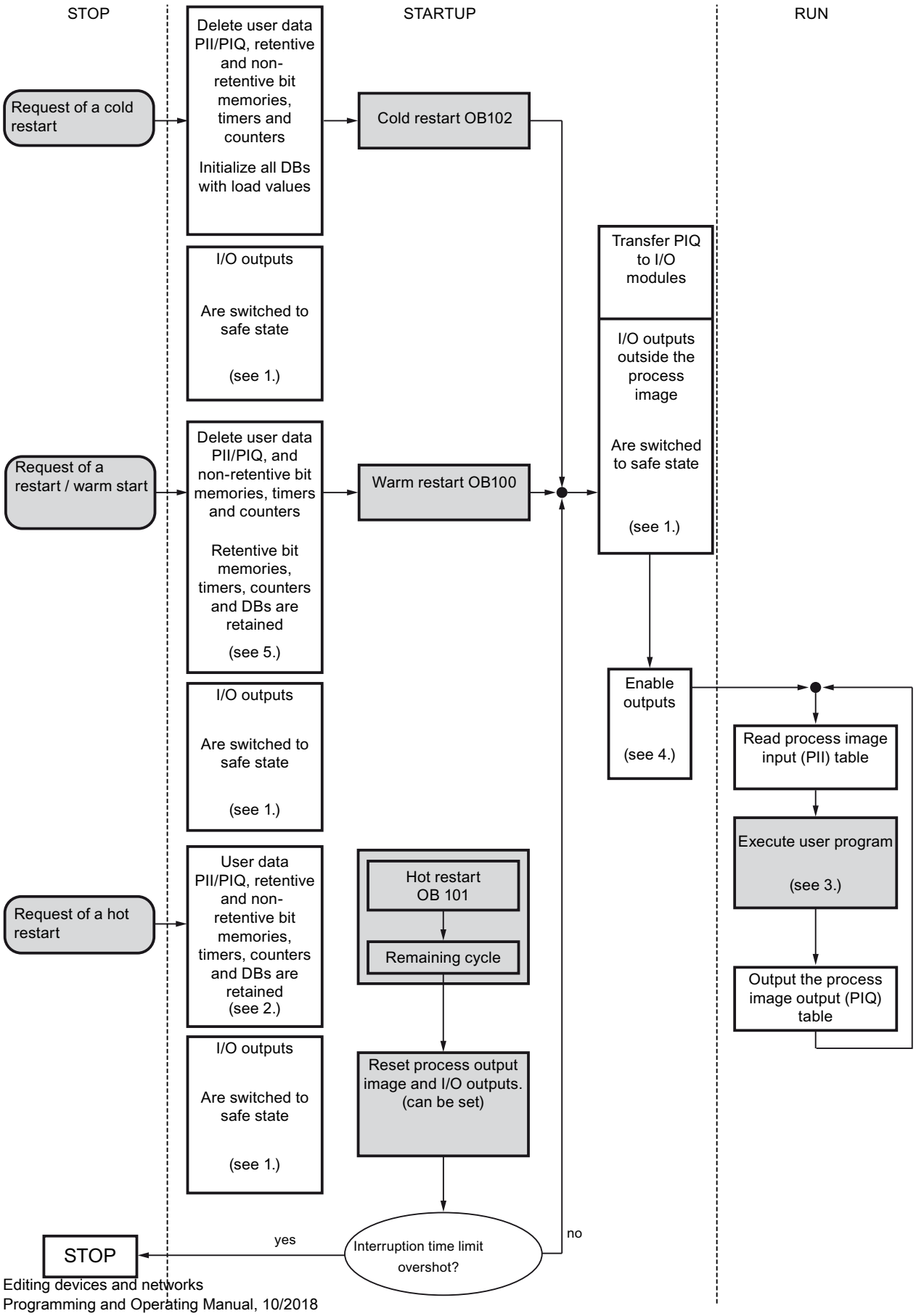

= behavior can be controlled by means of user functions

Legend for figure:

1. All peripheral outputs are switched by the peripheral modules (hardware side) to safe state (default value = "0"). This is regardless of whether it is used in the user program inside or outside the process image area.
   If signal modules with substitute value capability are used, the behavior of the outputs can be assigned, e.g. retain last value.
2. Necessary for the processing of the remaining cycle.
3. The interrupt OBs are also available during first calling of a current process image of the inputs.
4. You can use the following measures to specify the state of the local and distributed peripheral outputs in the first cycle of the user program:
   – Use configurable output modules to be able to output substitute values or to retain the last value.
   – On restart: Enable CPU startup parameter "Reset outputs on restart" in order to output a "0" (corresponds to default setting).
   – Pre-assign outputs in the startup OB (OB 100, OB 101, OB 102).
5. In S7-300 systems only the DB areas that are configured as retentive are retained.

RUN mode (S7-300, S7-400)

Function

In "RUN" operating mode the cyclic, time-driven, and interrupt-driven program execution is performed:

- The process image input is read in.
- The user program is executed.
- The process image output is read out.

The active exchange of data between CPUs via global data communication (global data table), via communication instructions for configured connections and via communication instructions for non-configured connections is only possible in RUN mode.

See also

Principles of the operating modes of S7-CPUs (Page 707)

HOLD mode (S7-300, S7-400)

Function

"HOLD" mode is a special case, since it is used only for tests in "STARTUP" or "RUN" operating mode.

"HOLD" mode:

- All timers are frozen. Timers and runtime meters are not processed, monitoring timers and the basic clock cycles of the time-driven levels are halted.
- The real-time clock still runs.
- Outputs are not enabled, they can however be enabled for test purposes.
- Inputs and outputs can be controlled.
- When power is restored after a power failure, CPUs with battery back-ups switch from "HOLD" to "STOP" operating mode and do not perform a "warm restart". On restoration of power, CPUs without battery back-ups automatically perform a "warm restart".
- They can receive global data. In addition passive one-way communication via the communications SFBs for configured connections and via communications SFCs for unconfigured connections can be performed.

See also

Principles of the operating modes of S7-CPUs (Page 707)

Basics of a memory reset (S7-300, S7-400)

Function

A CPU memory reset can be performed in STOP mode. A memory reset can be performed manually using the mode selector (MRES) or from the programming device, e.g. before loading a user program.

A memory reset resets the CPU to the "initial condition". This means that:

- The entire user program in the work memory and in the RAM load memory and all operand areas are cleared.
- The system parameters together with the CPU and module parameters are reset to the default setting. The MPI parameters set before the memory reset remain unchanged.
- When a memory card (flash EPROM) or micro memory card is plugged in, the CPU will copy the user program from the memory card to the work memory. If the respective configuration data is also present on the memory card, the CPU and module parameters are also copied.
- The diagnostics buffer, the MPI parameters, the time of day, and the runtime meter are not reset.

Memory areas (S7-300, S7-400)

What you should know about micro memory cards for S7-300 (S7-300, S7-400)

How the memory card functions

The SIMATIC micro memory card is a memory module for a S7-300-CPU. You can configure the micro memory card as a load memory or as removable media.

STEP 7 V11 supports use as load memory.

To use it you have to insert the SIMATIC micro memory card in the CPU.

NOTICE
Formatting the micro memory card If you format the micro memory card with Windows, for example with a commercially available card reader, you make the micro memory card unusable as a storage medium for an S7-CPU.

What can be saved on a SIMATIC Micro Memory Card

You can save the user program (blocks and system data) on a micro memory card.

Alternatively, you can use the micro memory card for firmware updates or for backup of the CPU firmware.

Transferring objects from the project to a micro memory card

When the micro memory card is inserted in the programming device or an external prommer, you can transfer the following objects from project navigation to the micro memory card:

- Individual blocks (multiple selection possible)
  In this case a consistent transfer is available, as the dependencies of the blocks to each other is taken into account with block selection.
- PLC
  In this case, all objects relevant to processing are transferred, such as blocks and the hardware configuration on the micro memory card, just as with downloading.

To perform a transfer, you can move the objects by dragging and dropping, or use the command "Write to memory card" in the "Project" menu.

Transferring objects from the micro memory card to the project

You can transfer Individual blocks (multiple selection is possible) by dragging them to the project. A hardware configuration cannot be transferred from the memory card to the project.

Backing up firmware on the micro memory card

We recommend that your create a backup copy of the firmware for emergencies.

To do so, you must insert a blank micro memory card in the CPU and adhere to a specific sequence of steps. You will find them described in the CPU documentation.

Documentation for S7-300 (http://support.automation.siemens.com/WW/view/de/13008499/0/en)

Updating firmware with a micro memory card

You can get the latest firmware on the internet as *.UPD files from the Service & Support pages:

http://support.automation.siemens.com (http://support.automation.siemens.com)

Save the firmware files on the hard drive.

In order to store the files on the micro memory card, select the command "Create firmware update memory card" in the "Project" menu.

Then follow the instructions in the Service & Support portal for performing a firmware update with your CPU.

See also

Memory areas of the S7-300 CPUs (Page 724)

What you should know about memory cards for S7-400 (S7-300, S7-400)

How the memory card functions

You can insert a memory card in all S7-400 CPUs. They act as an expansion of the load memory. Depending on the type of memory card used, the user program remains saved on the memory card, even if the module has no voltage.

Types of memory cards

- RAM card: only for expanding the integrated load memory.
- FLASH card: for saving the user program, even if there is no voltage, without buffering or external to the CPU.

Transferring objects from the project to a memory card

When the memory card is inserted in the programming device or an external prommer, you can transfer the following objects from project navigation to the memory card:

- Individual blocks (multiple selection possible)
  In this case a consistent transfer is available, as the dependencies of the blocks to each other is taken into account with block selection.
- PLC
  In this case, all objects relevant to processing are transferred, such as blocks and the hardware configuration on the memory card, just as with downloading.

To perform a transfer, you can move the objects by dragging and dropping, or use the command "Write to memory card" in the "Project" menu, and if applicable, follow the instructions in the dialog.

Transferring objects from the memory card to the project

You can transfer Individual blocks (multiple selection is possible) by dragging them to the project. A hardware configuration cannot be transferred from the memory card to the project.

Downloading the user program to the memory card

The command "Download user program to memory card" in the menu "Online" saves the entire user program to the FLASH card. The FLASH card is in the CPU. Before the download, all existing data on the FLASH card is deleted.

- The CPU must be in STOP mode.
- You can only download to FLASH cards with these menu commands

Updating firmware with a memory card

You can get the latest firmware on the internet as *.UPD files from the Service & Support pages:

http://support.automation.siemens.com (http://support.automation.siemens.com)

Save the firmware files on the hard drive.

To store the file on the memory card, select the command "Create firmware update memory card" in the "Project" menu.

Then follow the instructions in the Service & Support portal for performing a firmware update with your CPU.

See also

Memory areas of the S7-400-CPUs (Page 726)

Organization of Memory Areas (S7-300, S7-400)

Distribution

The memory of the S7 CPUs can be divided into three areas:

- Load memory (depending on the CPU and memory card type executed as RAM, EPROM or flash memory)
- Work memory (integrated RAM)
- System memory (integrated RAM)

Memory areas of the S7-300 (S7-300, S7-400)

Memory areas of the S7-300 CPUs (S7-300, S7-400)

The three memory areas of your CPU

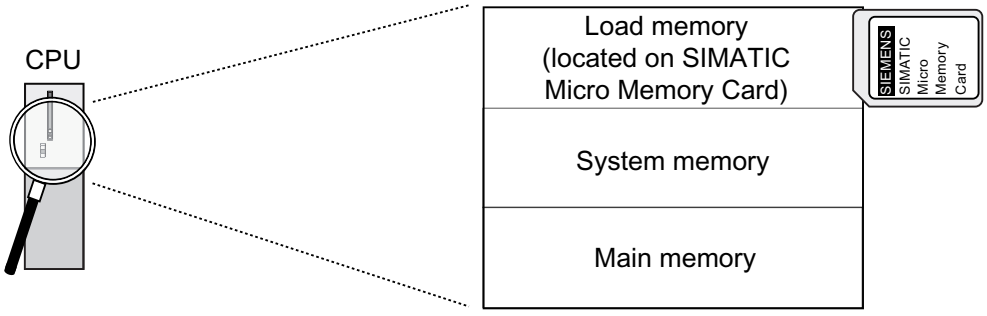

Load memory

The load memory is located on the SIMATIC Micro Memory Card. The size of the load memory corresponds exactly to the size of the SIMATIC Micro Memory Card. It is used to store code blocks, data blocks, and system data (configuration, connections, module parameters, etc.). Blocks that are identified as not relevant for execution are stored exclusively in the load memory. You can also store all the configuration data for your project on the SIMATIC Micro Memory Card.

Note

You must insert a SIMATIC Micro Memory Card into the CPU to enable loading of user programs and operation of the CPU.

System memory

The system memory is integrated in the CPU and cannot be expanded.

It contains

- The address areas for bit memories, timers, and counters
- The process images of the inputs and outputs
- Local data

Work memory

The work memory is integrated in the CPU and cannot be extended. It is used to execute the code and process user program data. Programs only run in the work memory and system memory.

See also

What you should know about micro memory cards for S7-300 (Page 720)

Retentivity of load memory, system memory, and main memory (S7-300, S7-400)

Your CPU is equipped with a maintenance-free retentive memory, i.e. no back-up battery is required for its operation. Due to the retentivity, the content of the retentive memory is retained even during a POWER OFF and restart (warm restart).

Retentive data in the load memory

Your program in the load memory is always retentive: It is stored on the SIMATIC Micro Memory Card, where it is protected against power failures or memory resets

Retentive data in the system memory

For bit memory, timers and counters, you decide via configuration (CPU properties) which part should be retentive and which should be initialized to "0" at warm restart.

The diagnostic buffer, MPI address and transmission speed (baud rate), as well as the runtime meter, are generally stored in the retentive memory area on the CPU. Retentivity of the MPI address and transmission speed ensures that your CPU can continue to communicate, even after a power failure, memory reset or loss of communication parameters (e.g. due to removal of the SIMATIC Micro Memory Card or deletion of communication parameters).

Retentive data in the work memory

The contents of retentive DBs are always retentive at restart and POWER ON/OFF. Retentive data blocks can be uploaded to the work memory in accordance with the maximum limit allowed by the work memory.

Non-retentive DBs are initialized from the load memory with their initial values at restart or POWER ON/OFF. Non-retentive data blocks and code blocks can be loaded in accordance with the maximum work memory limit.

You can find the size of the CPU retentive work memory (for retentive data blocks) in the technical specifications of the respective CPU.

Memory areas of the S7-400 (S7-300, S7-400)

Memory areas of the S7-400-CPUs (S7-300, S7-400)

Organization of Memory Areas

The memory of the S7-400-CPU can be divided into the following areas:

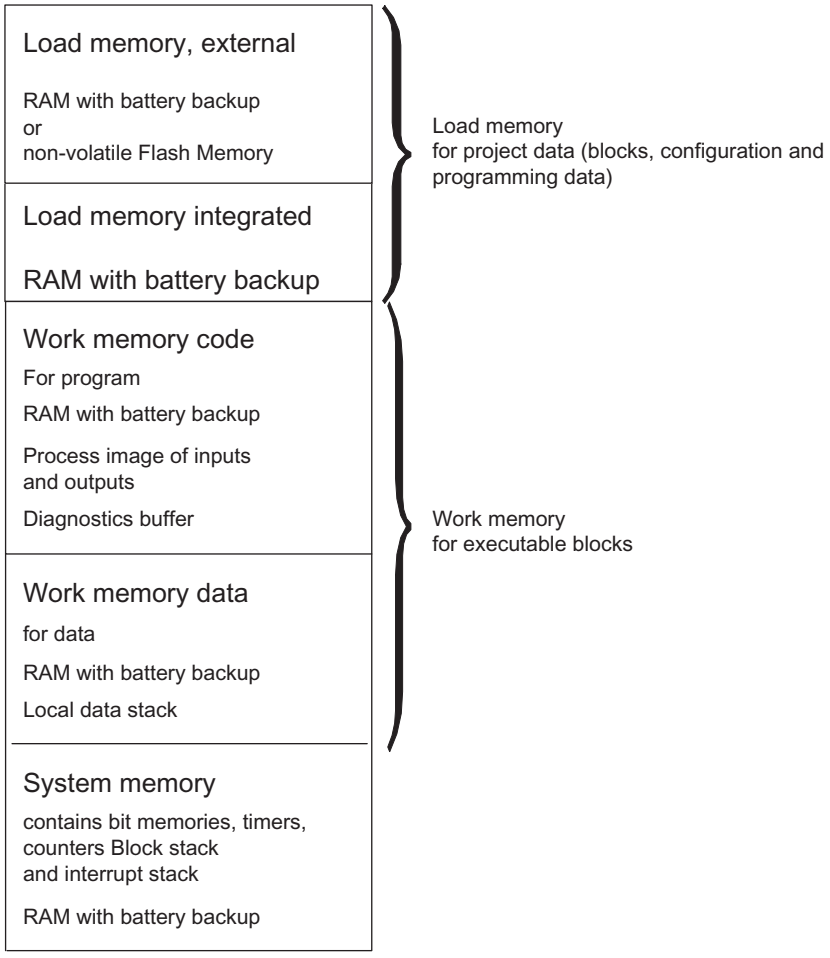

Memory Types in S7-400 CPUs

- Load memory for the project data, e.g. blocks, configuration and parameter settings.
- Work memory for the runtime-relevant blocks (logic blocks and data blocks).

- System memory (RAM) contains the memory elements that each CPU makes available to the user program, such as bit memory, timers and counters. System memory also contains the interrupt stack
- System memory of the CPU also makes temporary memory available (local data stack, diagnostic buffer and communication resources) that is assigned to the program for the temporary data of a called block. These data is only valid as long as the block is active. By changing the default values for the process image, local data, diagnostic buffer and communications resources (CPU properties for which you can set parameters) you can influence the work memory available for the process-relevant blocks.

Note

Please note the following if you expand the process image of a CPU. Reconfigure modules whose addresses have to be over the highest address of the process image so that the new addresses are still over the highest address of the expanded process image.

Important note for CPUs after the parameter settings for the allocation of RAM have been changed

If you change the work memory allocation by modifying parameters, this work memory is reorganized when you load system data into the CPU. The result of this is that data blocks that were created with SFC are deleted, and the remaining data blocks are assigned initial values from the load memory.

The usable size of the working memory for logic or data blocks is changed when loading the system data if you change the following parameters:

- Size of the process image (byte-oriented; in the "Cycle/Clock Memory" tab)
- Communication resources (S7-400 only; "Memory" tab)
- Size of the diagnostic buffer ("Diagnostics/Clock" tab)
- Number of local data for all priority classes ("Memory" tab)

Basis for Calculating the Required Working Memory

To ensure that you do not exceed the available amount of working memory on the CPU, you must take into consideration the following memory requirements when assigning parameters:

Table 1-82 Memory requirements

Parameters	Required working memory	In code/data memory
Size of the process image (inputs)	12 bytes per byte in the process image input as of V6.0: 20 bytes per byte in the process image input	Code memory
Size of the process image (outputs)	12 bytes per byte in the process image input as of V6.0: 20 bytes per byte in the process image input	Code memory
Communication resources (communication jobs)	72 bytes per communication job	Code memory
Size of the diagnostic buffer	32 bytes per entry in the diagnostic buffer	Code memory
Quantity of local data	1 byte per byte of local data	Data memory

Flexible memory capacity

- Work memory:
  The capacity of the work memory is determined by selecting the appropriate CPU from the graded range of CPUs.
- Load memory:
  The integrated load memory is sufficient for small and medium-sized programs.
  The load memory can be increased for larger programs by inserting the RAM memory card. Flash memory cards are also available to ensure that programs are retained in the event of a power failure even without a backup battery. Flash memory cards (8 MB or more) are also suitable for sending and carrying out operating system updates.

Backup

- The backup battery provides backup power for the integrated and external part of the load memory, the data section of the working memory and the code section.

See also

What you should know about memory cards for S7-400 (Page 722)

Contents of work and system memory (S7-300, S7-400)

Address areas of the CPUs (S7-300, S7-400)

Overview

Memory elements that each CPU makes available to the user program, such as the process image input and output, bit memory, timers and counters. The work or system memory of the S7-CPUs is divided into operand areas. Through use of appropriate operations in your user program, you address the data directly in the respective operand area.

The following table shows the operand areas:

Operand area	Access via units of the following size:	S7 notation	Description
Process image input	Input (bit)	I	At the start of every cycle, the CPU reads the inputs from the input modules and saves the values to the process image input.
	Input byte	IB	
	Input word	IW	
	Input double word	ID	

Operand area	Access via units of the following size:	S7 notation	Description
Process image output	Output (bit)	Q	During its cycle, the program calculates the values for the outputs and places them in the process image output. At the end of the cycle, the CPU writes the calculated output values to the output modules.
	Output byte	QB	
	Output word	QW	
	Output double word	QD	
Bit memory	Bit memory (bit)	M	This area provides storage for intermediate results calculated in the program.
	Memory byte	MB	
	Memory word	MW	
	Memory double word	MD	
Timer	Timer (T)	T	This area provides storage for timers.
Counter	Counter (C)	C	This area provides storage for counters.
Data block	Data block, opened with "OPN DB":	DB	Data blocks store information for the program. They can either be defined so that all code blocks can access them (global DBs) or assigned to a specific FB or SFB (instance DB).
	Data bit	DBX	
	Data byte	DBB	
	Data word	DBW	
	Data double word	DBD	
	Data block, opened with "OPN DI":	DI	
	Data bit	DIX	
	Data byte	DIB	
	Data word	DIW	
	Data double word	DID	

Operand area	Access via units of the following size:	S7 notation	Description
Local data	Local data bit	L	This area contains the temporary local data of a block while the block is being processed. The L stack also provides memory for transferring block parameters and for saving intermediate results from LAD networks.
	Local data byte	LB	
	Local data word	LW	
	Local data double word	LD	
I/O area: Inputs	Peripheral input byte	PIB	The peripheral input and output areas permit direct access to central and distributed input and output modules.
	Peripheral input word	PIW	
	Peripheral input double word	PID	
I/O area: Outputs	Peripheral output byte	PQB	
	Peripheral output word	PQW	
	Peripheral output double word	PQD	

process image input/output (S7-300, S7-400)

Basic principles of process images (S7-300, S7-400)

Function

When the user program addresses the input (I) and output (O) operand areas, it does not query or change the signal states on the digital signal modules. Instead, it accesses a memory area in the system memory of the CPU. This memory area is referred to as the process image.

Advantages of the process image

Compared with direct access to input and output modules, the main advantage of accessing the process image is that the CPU has a consistent image of the process signals for the duration of one program cycle. If a signal state on an input module changes during program execution, the signal state in the process image is retained until the process image is updated again in the next cycle. The process of repeatedly scanning an input signal within a user program ensures that consistent input information is always available.

Access to the process image also requires far less time than direct access to the signal modules since the process image is located in the internal memory of the CPU.

Updating the process images (S7-300, S7-400)

Sequence

The process image inputs/outputs (OB 1 process image) are updated in the following order:

1. The internal tasks of the operating system are performed.
2. The process image output (PIQ) table is written to the outputs of the module.
3. The status of the inputs is read into the process image input (PII) table.
4. The user program is executed with all the blocks that are called in it.

The operating system automatically controls the writing of the process image output to the outputs of the modules and the reading of the process image input.

I/O access error during process image updating

S7-300 CPUs respond to an error during the update of the process image as follows:

- No diagnostics buffer entry, no OB call. The respective input bytes are set to "0" and remain at "0" until the error is resolved.

S7-400 CPUs respond to an error during process image update as follows:

- Diagnostics buffer entry and start of OB 85 with every instance of I/O access each time the corresponding process image is updated. The defective input bytes are set to "0" each time the process image is accessed.

You can set the reaction to I/O access errors by selecting one of the following alternatives:

- The CPU creates a diagnostics buffer entry and starts OB 85 each time the I/O is accessed.
- The CPU creates a diagnostics buffer entry and starts OB 85 only for incoming and outgoing I/O access errors. Before OB 85 is called, the faulty input bytes are set to "0" and are no longer overwritten by the operating system until the outgoing I/O access error.
- The CPU exhibits the default behavior of the S7-300. OB 85 is not called, the corresponding input bytes are set to "0" and are no longer overwritten by the operating system until the error is resolved.

Frequency of OB 85 starts

In addition to the assigned reaction to I/O access errors, the address space of a module also influences how often OB 85 starts:

- For a module with an address space up to a double word, OB 85 starts once, for example, for a digital module with up to 32 inputs or outputs, or for an analog module with two channels.
- For modules with a larger address space, OB 85 starts as often as access has to be made to it with double word commands, for example, twice for an analog module with four channels.

Local data stack (S7-300, S7-400)

Function

When you create organization blocks, you can declare temporary tags (TEMP) that are only available when the block is executed and are then overwritten again. Each organization block also requires 20 bytes of local data for its start information.

The CPU has a limited amount of memory for the temporary tags of the local data of blocks currently being executed. The size of this local memory, called the local data stack or L stack, depends on the particular CPU. By default, the local data stack is divided equally among the priority classes. This means that each priority class has its own area in the local data stack. This ensures that high-priority classes and their OBs have space available for their local data.

Before the local data stack is accessed for the first time, the local data must be initialized.

The local data stack stores the following data:

- Temporary tags of the local data of blocks
- Start information of the organization blocks
- Information regarding the transfer of parameters
- Intermediate results of the logic in ladder logic programs

The following figure shows the assignment of local data to the priority classes:

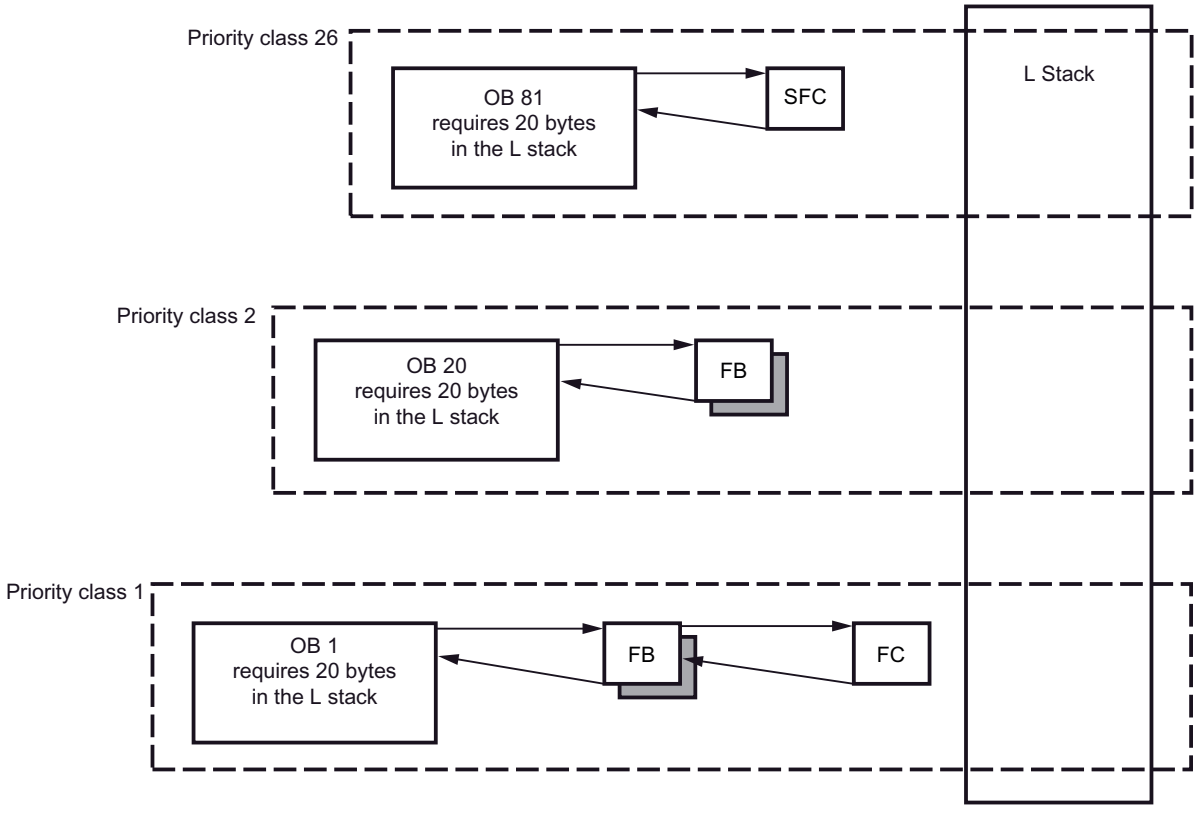

In the local data stack, OB 1 is first interrupted by OB 20, which in turn is interrupted by OB 81.

CAUTION

All temporary tags (TEMP) of an OB and its associated blocks are stored in the local data stack. If you use a large number of nesting levels when executing your blocks, the local data stack can overflow.

If you exceed the permissible local data stack size for a program, the S7-CPU changes to STOP operating mode.

You can get an overview of the required local data with the function "Show program information".

Don't forget the local data requirements of the synchronous error OBs.

Assigning local data to priority classes

For S7-300: The maximum amount of local data per priority class and per block is limited for the S7-300 (see the technical data of the CPU).

For S7-400: The maximum amount of local data in total is limited (see the technical data of the CPU). You distribute the available local data freely to the various priority classes within the framework of the CPU parameter assignment. The maximum amount of local data must not be exceeded in the process.

Interrupt stack (S7-300, S7-400)

Function

If the program execution is interrupted by an organization block with higher priority, the operating system saves the following data in the interrupt stack (I stack):

- The current contents of the accumulators and address registers
- The number and size of the open data blocks

Once the new organization block has been executed, the operating system loads the information from the interrupt stack. The execution of the interrupted block then resumes at the place where it was interrupted.

Block stack (S7-300, S7-400)

Function

If processing of a block is interrupted by the call of another block or by a higher priority class, such as an interrupt or error handling, the operating system saves the following data in the block stack (B stack):

- Number, type (OB, FB, FC, SFB, SFC) and return address of the block that was interrupted.
- Numbers of the data blocks from the DB and DI registers that were open when the block was interrupted.

Using this data, the user program can be resumed after the interruption.

You can read the block stack in "STOP" mode on the programming device with the program editor open and using the test function "Call hierarchy". The block stack lists all the blocks that had not been completely executed when the CPU switched to STOP operating mode.

The following figure shows the order in which the blocks are called:

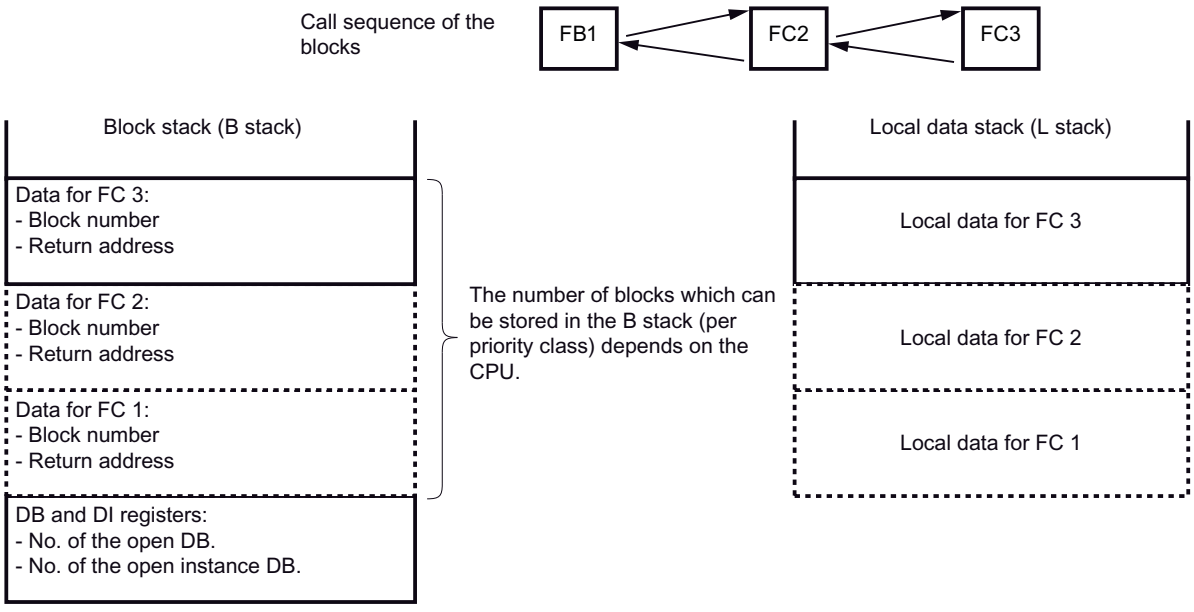

Data block register

Two data block registers are available. They contain the numbers of the data blocks that have been opened:

- The DB register contains the number of the opened global data block.
- The DI register contains the number of the opened instance data block.

Diagnostics buffer (S7-300, S7-400)

Function

The diagnostics buffer contains system- and user-defined diagnostics events, entered in the order in which they occur. The first entry contains the latest event. The number of entries in the diagnostics buffer depends on the respective module and its current operating mode.

Possible diagnostics events:

- Faults on a module
- Faults in the process wiring
- System errors in the CPU
- Operating mode transitions of the CPU
- Errors in the user program
- User-defined diagnostics events using system function SFC 52

The information that is entered in the diagnostics buffer when a system diagnostics event occurs is identical to the start information that is transferred to the respective organization block.

The entries in the diagnostics buffer cannot be deleted; the content of the diagnostics buffer is retained even after a memory reset.

Advantages of the diagnostics buffer

The diagnostics buffer offers the following advantages:

- If the CPU changes to "STOP" mode, you can evaluate the last events leading up to the STOP and thus determine the cause of the STOP.
- You can recognize the causes of errors more quickly and thus increase the availability of the system.
- You can evaluate and optimize the dynamic system response.

Organization of the diagnostics buffer

The diagnostics buffer is designed as a ring buffer. The maximum number of entries depends on the module; for example, the S7-300 CPU 314 can have 100 entries. When this maximum number of entries is reached, the next diagnostics buffer event causes the oldest entry to be deleted. All entries then move back on place. This means that the newest entry is always the first entry in the diagnostics buffer.

The number of entries displayed in the diagnostics buffer depends on the module and its current operating mode. With some CPUs, it is possible to assign the length of the diagnostics buffer.

Evaluation of the diagnostics buffer

The contents of the diagnostics buffer can be accessed as follows:

- Online via the diagnostics view
- In the user program via SFC 51 "RDSYSST".

By default, the CPU automatically sends the last diagnostics buffer entry to a monitoring device (programming device, OP, or TD) before it changes from "RUN" to "STOP" mode. This allows you to quickly find out why the operating mode changed and rectify the cause. You can change this default.

Note

To make the best use of the time stamp information on the diagnostics buffer entries, set the date and time of day on the module and check the settings occasionally.

Timers and counters (S7-300, S7-400)

Definition

Timers and counters are memory areas of the system memory. You specify the function of a timer or counter in the user program, for example the on-delay or the initial count value. The number of timers and counters available depends on the CPU.

Note

Note the following information on timers:

- If you use more timers and counters in your user program than the CPU permits, a synchronous error is signaled and organization block "OB 121" is started.

Access to the I/O data area (S7-300, S7-400)

I/O data area (S7-300, S7-400)

Structure of the distributed I/O data area (S7-300, S7-400)

I/O data area

The I/O data area can be divided into:

- User data
- Diagnostics and parameter data

Both areas have an input area with read-only access and an output area with write-only access.

User data (S7-300, S7-400)

Definition

User data can be:

- Digital and analog input and output signals from signal modules
- Control and status information from function modules
- Information for point-to-point and bus connections from communications modules

Access to user data

The user data in digital signal modules are addressed via the byte address of the input or output area.

For analog signal modules, the word address of the input or output area is used for addressing.

You can access user data via load and transfer commands or communications functions (operator control and monitoring) or via the process image transfer.

With load and transfer commands, it is possible to achieve a data consistency of 1, 2 and 4 bytes. Data from the process image are always consistent.

If you want to achieve data consistency of 3 or more than 4 bytes, you have to use the instruction "DPRD_DAT" (SFC 14) or "DPWR_DAT" (SFC 15).

To transfer 4 contiguous and unmodified bytes, use the instruction "Transfer double word". If you use four separate "transfer input byte" instructions, the consistency of the bytes cannot be maintained and unwanted results may occur.

Diagnostic and parameter data (S7-300, S7-400)

Data records of the diagnostics and parameter data

The diagnostics and parameter data for a module cannot be addressed individually. Rather, the data is always transferred in the form of complete data records. This means that consistent diagnostics and parameter data is always transferred.

Diagnostics and parameter data is addressed via the start address of the respective module and the data record number. Data records are divided into input and output data records. Input data records can only be read and output data records can only be written. You can access data records using system functions or communications functions.

The following table shows the assignment of data records to diagnostics and parameter data:

Data	Description
Diagnostics data	For modules with diagnostics capability, you obtain the diagnostics data of the module by reading data records 0 and 1.
Parameter data	For modules with parameters, you transfer the parameters to the module by writing data records.

Accessing the data records of a module (S7-300, S7-400)

Instructions for accessing data records

You can use the information in the data records to reassign parameters for modules with parameters and to read out diagnostics information for modules with diagnostics capability.

The following table shows which functions you can use to access data records:

Instruction	Application
Assigning module parameters	
WR_PARM	Transfers the modifiable parameters (data record 1) to the addressed signal module
WR_DPARM	Transfers the individual parameter data records from SDBs 100 to 129 to the addressed signal module
PARM_MOD	Transfers all parameters from SDBs 100 to 129 to the addressed signal module
WR_REC	Transfers any data record to the addressed signal module
Reading out diagnostics information	
RD_REC	Reads out the diagnostics data

Note

Data records must not be read or written by the I/O modules in the user program with instruction "WR_REC/RD_REC" or instructions "WRREC/RDREC" if a DPV1 slave is configured using a GSD file (GSD from rev. 3) and the DP interface of the DP master (S7-400 CPU) is set to "S7-compatible". In this case the DP master would address the wrong slot, i.e. the configured slot +3. To avoid this problem, change the DP master interface to "DPV1".

Module start address (S7-300, S7-400)

The module start address is the lowest byte address of a module. It represents the start address of the user data area in the module and is used in many cases to represent the entire module.

The module start address is entered in the start information of the appropriate organization block when, for example, process interrupts, diagnostics interrupts, insert/remove module interrupts, and power supply errors occur and thus identifies the interrupt-generating module.

Configuring module start addresses

The addresses used in the user program and the module start addresses are coordinated when the modules are configured.

You can change the start addresses automatically assigned when a module is inserted in the rack in the module properties ("Addresses" group).

User data area (S7-300, S7-400)

Access to the I/O addresses (S7-300, S7-400)

User data in the I/O operand area

The user data represent the I/O operand area. The following areas exist:

- Peripheral inputs PI
- Peripheral outputs PQ

If the user data lie in the process image, the CPU automatically undertakes the data exchange between module and process image area when updating the process images.

If the user data do not lie in the process image, you must access them from the user program by direct I/O access to the data (for example, L %IB500:P).

The addresses of analog input and analog output modules do not normally lie in the process image area. However, you can place the addresses of analog modules in this area.

See also

Process images (input and output memory area) (Page 812)

Direct access to the I/O (PI and PQ memory areas) (Page 813)

Address areas (S7-300, S7-400)

I/O address areas

There is a separate address area for inputs and outputs. This means that the address of an I/O area must not only include the byte or word information but also the "I" identifier for inputs and the "Q" identifier for outputs.

The following table lists the available I/O address areas:

Operand area	Access via units of the following size:	S7 notation
I/O area: Inputs	Peripheral input byte Peripheral input word Peripheral input double word	PIB PIW PID
I/O area: Outputs	Peripheral output byte Peripheral output word Peripheral output double word	PQB PQW PQD

See also

Direct access to the I/O (PI and PQ memory areas) (Page 813)

Setting the operating behavior (S7-300, S7-400)

Introduction to tuning the operational performance (S7-300, S7-400)

Influencing the properties of automation systems

Properties of automation systems that are not preset can be influenced as follows:

- By setting the system parameters
- By using system functions (SFCs) in extended instructions

Changing module parameters (S7-300, S7-400)

Default settings

In their as-delivered state, all modules with parameters have default settings suitable for standard applications. These default values allow the modules to be used immediately without making any additional settings.

You can, however, modify the behavior and the properties of the modules to suit the requirements and circumstances of your application. Modules for which you can set parameters include CPUs, FMs, CPs and some analog input and analog output modules and digital input and output modules.

Module parameters

Module parameters are grouped to improve clarity; for example, the following groups exist for CPUs:

- Startup
- Cycle
- PROFINET interface or other interface
- Diagnostics
- Retentive address area
- Clock memory
- Interrupts
- Protection
- Clock
- Connection resources

Settings and loading parameters

You can set the module parameters. When you save a device configuration with its parameters, system data is created and loaded to the CPU with the user program and then transferred to the modules during startup.

Assigning parameters with instructions

In addition to assigning parameters, you can also use advanced instructions in the program to modify module parameters.

The following table shows which instructions can change which module parameters:

System function	Application
WR_PARM	Transfers the modifiable parameters (data record 1) to the addressed signal module.
WR_DPARM	Transfers individual parameter data records from the corresponding SDBs to the addressed signal module.
PARM_MOD	Transfers all parameters from the corresponding SDBs to the addressed signal module.
WR_REC	Transfers any data record to the addressed signal module

General CPU parameters (S7-300, S7-400)

General properties of modules

The "General" parameter group specifies the type and location of the module as well as information relevant to the network, e.g. the device address, in the case of modules with interfaces.

Short designation, article no./ firmware, name, plant designation, location designation

The short designation, the information below it, and the article no./firmware are identical to the information in the hardware catalog.

The "Name" field contains the name of the module. You can change this as required. If you change the name, the changed name appears both in the device view and in the network view.

CPUs allow you to specify a specific plant designation. The higher-level item designation is used to identify parts of the plant uniquely according to their function. It is hierarchically structured in accordance with IEC 1346-1. You can also enter a location identifier of up to 32 characters. This is part of the item designation that identifies the exact location of, for example, a process tag or process tag group within a process (similar to a street on a city map).

The identifier entered in the "higher level item designation" or "location designation" field can be evaluated in the user program of the CPU. Use the option of entering a higher level item designation or location designation if you want to run the user program as a function of the higher level item designation of location designation.

Comment

Each module has a comments field. You can specify the purpose of the module in this field, for example.

Startup (S7-300, S7-400)

Startup parameters (S7-300, S7-400)

Startup behavior of the CPU

The parameters that determine the startup behavior of the S7-CPU are described below.

Startup when expected and actual configurations do not match

The preset configuration is represented by the configuration loaded to the CPU (system data blocks). The actual configuration is the actual configuration of the automation system.

If the "Startup when expected and actual configurations do not match" check box is cleared, the CPU changes to "STOP" in the following situations:

- If one or more modules are not inserted in their configured slot.
- If a module type that differs from the configured module type is inserted, e.g. during commissioning.
- If the distributed I/O is missing.

If additional modules (central or distributed) beyond those configured are present, they are disregarded when comparing the preset and actual configurations.

If the "Startup when expected and actual configurations do not match" check box is selected, the CPU will startup even in the cases listed above.

The I/O structure (central and distributed modules, central rack and DP slaves) is not checked by the CPU. PROFIBUS-DP interface modules are an exception. Because of their close connection to the CPU, configured modules of this type must always be inserted in order for the CPU to startup.

The "Startup when expected and actual configurations do not match" option is selected by default for the CPU. This means that the CPU will startup even if the preset and actual configurations are different.

Monitoring time for the "ready message" from the modules

Maximum wait time for the "ready" message of all configured modules after POWER ON. If after expiration of this monitoring time any module has not sent a "ready" message to the CPU, the actual configuration is different from the preset one. The CPU reacts to this based on the setting of the "Startup when expected and actual configurations do not match" parameter.

Within this monitoring time, for example, the various power supplies for a multi-tier configuration or for devices connected to the CPU via the DP master interface can be switched on in any order.

Monitoring time for the transfer of parameters to the modules

Maximum wait time for parameters to be "distributed" to modules with parameters (monitoring time starts after "Ready message from modules"). For CPUs with a DP master interface, you also use this parameter to specify the power-up time monitoring for the DP slaves. In other words,within this time the DP slaves must power-up and be assigned parameters by the CPU (as DP master).

If this monitoring time expires before all of the module/DP slave parameters have been assigned, the actual configuration is different from the preset one. The CPU reacts to this based on the setting of the "Startup when expected and actual configurations do not match" parameter.

Startup parameters only for S7-400 CPUs (S7-300, S7-400)

Reset outputs at hot restart (S7-300, S7-400)

Reset outputs at hot restart

S7-400 CPUs reset the outputs (and therefore also the process image of the outputs PIO) upon hot restart .

See also

Hot restart (Page 711)

Disable hot restart at startup by means of operator control (S7-300, S7-400)

Disable hot restart at startup by means of operator control (for example, programming device) or communication job (for example, from MPI stations)

You have the option of disabling "hot restart" startup mode for the S7-400 if it would otherwise be triggered by programming device operation or communication job. Hot restart can then only be executed locally.

- Select this check box if the startup mode triggered by operator control or communication job is to be limited to warm restart (=manual restart) or cold restart.
- All startup modes are possible if the check box is not selected.

Hot restart upon operator control or communication job (=manual hot restart) will be triggered under the following conditions:

- The "Disable hot restart ..." check box has not been selected and this parameter assignment has been transferred to the CPU
- There is nothing to prevent hot restart (battery ok, no user program change in STOP mode ...)
- Hot restart operating mode is selected in the programming device or another device triggers hot restart with a communication job.

See also

Hot restart (Page 711)

Cyclic behavior (S7-300, S7-400)

Settings for the cycle behavior (S7-300, S7-400)

Update OB1 process image cyclically

If this option is selected, the OB1 process image is updated cyclically (default). For S7-400-CPUs the OB1 updating can be switched off with these parameters.

Scan cycle monitoring time

The scan cycle monitoring time is set in ms.

If the cycle time exceeds the scan cycle monitoring time, the CPU switches to "STOP" mode. You can prevent "STOP" by programming organization block OB 80.

Possible causes for exceeding the scan cycle monitoring time include:

- Communications processes
- Accumulation of interrupt events
- CPU program error

Control of cycle loading by communications processes

The "Cycle load due to communication" parameter allows you to restrict the time taken by communications processes, which inevitably also extend the cycle time. Examples of communications processes include:

- Transferring data to another CPU
- Loading blocks (initiated via a programming device)

Test functions performed with the programming device are hardly influenced by this parameter, they can however significantly extend the cycle time. The time available for test functions can be restricted in process operation.

How the parameter works

The operating system of the CPU constantly provides the configured percentage of the overall CPU processing capacity for communications (time sharing). If the allocated processing capacity is not required for communications, it is made available to other processes.

Effect on the actual cycle time

In the absence of additional asynchronous events, the OB 1 cycle time is extended by a factor calculated by the following formula:

$$\frac{100}{100 - \text{"Cycle load due to communication"}}$$

Setting the size of the process image

For CPUs with variable process image size, the size of the process image (in bytes) can be set. The process image area always starts at input or output byte 0.

For S7-300-CPUs with variable process image size, there are special characteristics that must be observed during programming when accessing the process image area. These special characteristics are described in the documentation for the CPU.

Example: The variable setting of the process image currently takes effect only when the process image is updated at the cycle control point. The PII is updated only up to the set PII size, and only the set PIQ area is output to the output modules. User program accesses to the area between the assigned process image size and maximum process image size do not trigger a synchronous access error.

See also

Examples of cycle loading by communications (Page 745)

Examples of cycle loading by communications (S7-300, S7-400)

Below are two examples of how communications can affect the cycle loading.

Example of cycle load due to communications without additional asynchronous events

When you set the cycle load due to communication to 50%, the OB 1 cycle time can be doubled.

At the same time, the OB 1 cycle time is also influenced by asynchronous events (such as hardware interrupts or cyclic interrupts). From a statistical point of view, even more asynchronous events occur within the OB1 cycle because of the extension of the cycle time by the communication portion. This further extends the OB 1 cycle. This extension depends on how many events occur per OB 1 cycle and the duration of the event processing.

Example of cycle load due to communications with additional asynchronous events

For a pure OB 1 cycle time of 500 ms, a communications load of 50% can result in an actual cycle time of up to 1000 ms, assuming that the CPU has always enough communications jobs to process. If, parallel to this, a cyclic interrupt with 20 ms processing time is executed every 100 ms, this cyclic interrupt would extend the cycle by a total of 5*20 ms = 100 ms without communication load. That is, the actual cycle time would be 600 ms. Because a cyclic interrupt also interrupts communications, it affects the cycle time by adding 10 * 20 ms at 50% communication load. That is, in this case, the actual cycle time amounts to 1200 ms instead of 1000 ms.

Note

Check the effects of changing the value of the "Cycle load due to communication" parameter while the system is running.

The communications load must always be taken into account when setting the minimum cycle time; otherwise, time errors will occur.

OB 85 call on a distributed I/O access error (DAE) (S7-300, S7-400)

Settings for the OB 85 call

You can change the default reaction of the CPU to I/O access errors that occur while the system updates the process image.

Recommendation: If you want to activate the OB 85 call for I/O access errors, select the "only for incoming and outgoing errors" option, so that the CPU cycle time is not extended by repeated OB 85 calls.

In their default settings, S7-300-CPUs do not call OB 85 nor do they make an entry in the diagnostics buffer in reaction to I/O access errors.

Retentive behavior (S7-300, S7-400)

Retentive settings for bit memory, timers, and counters (S7-300, S7-400)

You use the retentivity parameters to specify the memory areas that are to be retained after a power failure or after a change from "STOP" to "RUN".

Number of memories bytes starting at MB 0

Enter the number of retentive memory bytes starting at memory byte 0.

Example

If you enter "2", memory bytes MB 0 and MB 1 are retentive.

Number of S7 timers starting at T0

Enter the number of retentive S7 timers (T) starting at T 0.

Each S7 timer occupies 2 bytes.

Example

If you enter "2", S7 timers T 0 and T 1 are retentive.

Number of S7 counters starting at C0

Enter the number of retentive S7 counters (C) starting at C 0.

Example

If you enter "2", S7 counters C 0 and C 1 are retentive.

Retention settings for data blocks (S7-300, S7-400)

Function

In the retentivity properties, you can specify which memory areas of the CPU should be retained after a power failure and after a transition from "STOP" to "RUN", even without a battery backup in the backup memory of the CPU. Whether retentive areas can be configured and, if so, how many depends on the CPU.

If a backup battery is installed, data blocks are always retentive.

Retentive areas (relevant only for S7-300 without battery backup or micro memory card)

Retentive areas are data areas of data blocks.

If you data areas in the work memory of the CPU are to be retentive, enter the following:

- DB no.
  Numbers of the data blocks that are to be retentive.
- Byte address (start address with the length of the data bytes)
  Byte addresses of the data blocks that are to be retentive.
- Number of bytes
  Number of the retentive data bytes starting with the byte address. For example, if you enter "1", then data byte "0" is retentive.

Possible settings for interrupts (S7-300, S7-400)

Settings for hardware interrupts (S7-300, S7-400)

Hardware interrupts

The higher the priority selected, the higher the priority will be for processing the associated hardware interrupt. Priority "0" deselects the interrupt OB. The default priority cannot be changed in S7-300 CPUs.

Process image partitions can be assigned to a hardware interrupt, provided the CPU supports process image partitions. At the start of the hardware interrupt OB, the assigned process image partition is updated so that the OB uses current process values. If the setting is "---", a process image is not assigned.

See also

Hardware interrupt organization blocks (OB 40 to OB 47) (Page 826)

Settings for time-delay interrupts (S7-300, S7-400)

Time-delay interrupts

Time-delay interrupt OBs allows parts of the user program to be delayed in their execution.

The higher the priority selected, the higher the priority with which the associated time-delay interrupt will be executed. Priority "0" deselects the interrupt OB. The default priority cannot be changed in S7-300 CPUs.

Process image partitions can be assigned to a time-delay interrupt, provided the CPU supports process image partitions. At the start of the hardware interrupt OB, the assigned process image partition is updated so that the OB uses current process values. If the setting is "---", a process image is not assigned.

Time-delay interrupts are handled using the following instructions:

- SRT_DINT Start time-delay interrupt
- CAN_DINT Cancel time-delay interrupt
- QRY_DINT Query time-delay interrupt status

See also

Time-delay interrupt organization blocks (OB 20 to OB 23) (Page 823)

Settings for asynchronous error interrupts (S7-300, S7-400)

Asynchronous error interrupts

The default priority cannot be changed in S7-300 CPUs.

The priority of asynchronous error OBs can be changed so that the CPU can remains in "RUN" mode even under the following circumstances:

- Asynchronous errors accumulate, all triggering the same interrupt OB (e.g. diagnostics interrupt, call OB 82).
- In addition another asynchronous event occurs, triggering another interrupt OB (e.g. program execution error, call OB 85).
- Only one alarm should be issued (e.g. in the case of station failure, which station has failed). No process response is necessary. In this case. OB 86 can be assigned a low priority, so as not to increase the reaction time to other interrupts.

Example of why a priority change is necessary

Severing a string a signal cables could trigger a diagnostics interrupt (call OB 82) for every signal module that is affected - which would immediately fill the start information buffer for OB 82.

So that an event that calls another asynchronous error OB (such as a program execution error, OB 85) does not cause the CPU to go the STOP mode, you should assign, for example, OB 85 a higher priority than OB 82.

See also

Power supply error organization block (OB 81) (Page 840)

Diagnostic interrupt OB (OB 82) (Page 843)

Insert/remove module interrupt organization block (OB 83) (Page 846)

CPU hardware fault organization block (OB 84) (Page 850)

Priority class error organization block (OB 85) (Page 851)

Rack failure organization block (OB 86) (Page 856)

Communication error organization block (OB 87) (Page 862)

Settings for time-of-day interrupts (S7-300, S7-400)

Priority

The priority of time-of-day interrupts determines the interrupt behavior of the interrupt OBs. For S7-300 the priority cannot at present be changed.

Active

Select the check box of the time-of-day interrupt OB if this OB should be started automatically when a restart occurs. You can alternatively activate the time-of-day interrupt in your user program using the OB_NR parameter in the ACT_TINT instruction.

Execution

Select how often the interrupts should be executed.

The intervals **every minute** to **every year** are available.

The intervals relate to the settings for start date and time of day.

Note

For **execution = end of month** the day of the start date is irrelevant (always the last day of the month).

Start time

Enter the date and time at which the interrupt is to be executed for the first time.

Example

Start date	Time-of-day
30.03.2006	10:15

Depending on the "Execution" parameter, the CPU will create further time-of-day interrupts at the specified intervals. The start time always relates to the base time of the CPU.

Note

If you selected "monthly" execution, the start date cannot be the 29th, 30th, or 31st, because then no time-of-day interrupt would occur in the month of February.

Instead, select "End of month" execution! In this case the day of the start date is not relevant (always the last day of the month!).

Process image partition

A process image partition can be assigned to a time-of-day interrupt, provided the CPU supports process image partitions. At the start of the interrupt OB, the assigned process image partition is updated so that the OB uses current process values. If the setting is "---", a process image partition is not assigned.

Special case: the one-time start time configured is in the past

The following configuration exists:

- Execution: once
- Active: Yes
- Start date/Time-of-day: is in the past (in relation to the CPU real-time clock)

Behavior of the CPU: After a cold restart or warm restart, the corresponding time-of-day interrupt OB is **called once** by the operating system.

See also

Time-of-day interrupt organization blocks (OB 10 to OB 17) (Page 820)

Setting time interrupts (S7-300, S7-400)

A cyclic interrupt is a periodic signal that leads to the automatic calling of a cyclic interrupt OB. The available settings are detailed below.

Priority

The priority determines the interrupt behavior of the interrupt OB. In the S7-300 CPUs it cannot be changed.

Execution

Enter the time intervals in the units shown (e.g. milliseconds) for execution of cyclic interrupt OBs.

The time cycle starts when the operating mode changes from "STOP" to "RUN".

Phase offset

If the phase offset is adjustable: Enter the time in milliseconds (ms) by which the actual execution time of the cyclic interrupt should be delayed.

Note

A phase offset is sensible if several cyclic interrupts are active. Without phase offset, cyclic interrupts would be called simultaneously and executed in succession depending on their priority.

The phase offset allows the execution time of the cyclic interrupts to be distributed across the cycle.

Unit

Refers to the "Execution" and "Phase offset" parameters. This conversion is available only on special CPUs that support cyclic interrupt settings in microseconds.

Process image partition

A process image partition can be assigned to a cyclic interrupt, provided the CPU supports process image partitions. At the start of the cyclic interrupt OB, the assigned process image partition is updated so that the OB works with current process values. If the setting is "---", a process image is not assigned.

See also

Cyclic interrupt organization blocks (OB 30 to OB 38) (Page 825)

Settings for the DPV1 interrupts (S7-300, S7-400)

Interrupts for DPV1

Interrupts can be originated by a DPV1 slave to ensure that the master CPU handles the interrupt-triggering event. The interrupt data are evaluated in the CPU even in STOP mode (update of diagnostics buffer and module status). However an OB is not executed in "STOP" mode.

In addition to the normal SIMATIC interrupts (such as diagnostics interrupt for ET 200M), interrupts such as status interrupts, update interrupts, and manufacturer-specific interrupts are supported.

See also

Status interrupt OB (OB 55) (Page 829)

Update interrupt OB (OB 56) (Page 831)

OB for manufacturer-specific alarms (OB 57) (Page 834)

Setting the isochrone mode interrupts (S7-300, S7-400)

Properties of isochronous mode interrupts

Isochronous mode interrupt OBs are executed synchronously with the equidistant clock cycle.

DP master system no.

Select the DP master system whose DP slaves should be assigned to the interrupt OB. The inputs and outputs of these DP slaves can then be synchronized with the equidistant clock cycle and the user program in the interrupt OB.

Process image partition(s)

Enter here the numbers of the process image partitions that the interrupt OB should access. The assigned distributed I/O are updated synchronously with the equidistant clock cycle via the process image partitions.

The assigned process image partitions must be updated in the synchronous cycle interrupt OB using the instructions SYNC_PI" and "SYNC_PO". The instruction "SYNC_PI" updates the input process image partition and the instruction "SYNC_PO" the output process image partition.

If the distributed I/O addresses are assigned to different process image partitions, the numbers of the process image partitions used must be entered separated by commas (e.g. 1,2,3).

Delay time

The delay time is the time between the global control telegram and the start of OB 61 (or OB 62 to OB 64). During this time, the DP master performs the cyclic data exchange with the DP slaves.

If the delay time = 0 is, OB 6x starts with the global control telegram. It is in this case extremely likely that access to the isochronous I/O via "SYNC_PI" at the start of the OB will fail as the I/O data is not yet available.

If the delay time is larger than the default value, then the OB runtime is "given away", since the clock-synchronized I/O could have been accessed earlier.

See also

Synchronous cycle interrupt OBs (OB 61 to OB 64) (Page 836)

Constant and isochronous modes (Page 2790)

Time-of-day functions (S7-300, S7-400)

Basic principles of time of day functions (S7-300, S7-400)

All S7-300/400 CPUs are equipped with either a real-time clock or a software clock. The clock can act in the automation system as both a time-of-day master and as a slave with external synchronization. This allows the use of time-of-day interrupts and run-time meters.

Time-of-day format

The clock always shows the time of day with a minimum resolution of 1 second and the date including the day of the week. Some CPUs also permit the display of milliseconds.

Setting and reading the time of day (S7-300, S7-400)

Setting the time of day

You can set and start the time of day and date of the CPU clock using the following instructions:

- "SET_CLK"
- "SET_CLKS" if this is available in the CPU

Note

To avoid time-of-day display differences in HMI systems, set the CPU to standard time or use central time-of-day synchronization.

Reading the time of day

You read the current date and time of day of the CPU with the instruction "READ_CLK" or via a menu command from the programming device.

Time synchronization SIMATIC procedure (S7-300, S7-400)

Reference

Parameter assignment of the interface properties in the following parameter groups:

- For the CPU: "Properties > General > Time"
- For the DP interface: "Properties > General > DP interface > Time synchronization".

SIMATIC procedure

With the SIMATIC procedure, the time of day is set according to MMS time-of-day messages (MMS - Manufacturing Message Specification). The MMS time-of-day messages come from a SIMATIC time-of-day transmitter or a CPU that is configured as a time master.

With a CP, the time-of-day messages can either be received by the local CPU or via LAN. It is possible to set whether or not the CP forwards the received time-of-day.

Compared with an NTP method that may be available, the SIMATIC mode provides higher accuracy.

Clock parameters

The clock parameters allow you to make the following settings:

- Synchronization type
  You can set the synchronization properties for the CPU internally (PLC) and externally (MPI).
  If you have set the synchronization mode to "As master", you can then select the time interval for automatic synchronization.
  If more than one module with a clock is present within a network, you need to specify which CPU will act as the master and which as the slave for time synchronization purposes. There can only be one time master.
- Time interval
  The time interval defines the interval between time queries.
- Correction factor
  The correction factor is used for the CPU to correct any variation of the clock within a 24 hour period.
  You can enter positive or negative value in ms.
  Example: If in 24 hours the clock has lost 4 seconds, you must enter a correction factor of "+4000 ms".

Instructions for synchronizing the time

To ensure that the time is consistent in all modules in the network, the time slaves are updated by the system program at regular configurable intervals. The instruction "SNC_RTCB" allows the date and time to be transferred from the time master to the time slave independently of the configured time interval.

Time-of-day synchronization in NTP mode (S7-300, S7-400)

Reference

Assigning parameters for the interface properties in the "Properties > General > PROFINET interface > Time synchronization" parameter group. The "Enable time synchronization via NTP server" option is selected.

NTP mode (NTP: Network Time Protocol)

In NTP mode, the device sends time queries at regular intervals (in client mode) to the NTP server in the subnet (LAN). Based on the replies from the server, the most reliable and most accurate time is calculated and the time on the station is synchronized.

The advantage of this mode is that it allows the time to be synchronized across subnets.

The IP addresses of up to four NTP servers need to be configured. The update interval defines the interval between the time queries (in seconds). The value of the interval ranges between 10 seconds and one day.

In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred; this corresponds to GMT (Greenwich Mean Time).

Time synchronization parameters in NTP mode

The clock parameters allow you to make the following settings:

- Enable time synchronization via NTP server
  Select this check box if you want the internal clock to be synchronized using the NTP synchronization mode.
- Server 1-4
  The IP addresses of up to four NTP servers need to be configured.
- Update interval
  The update interval defines the interval between time queries.

Using an operating hours counter (S7-300, S7-400)

Definition

An run-time meter counts the operating hours of connected equipment or the total runtime of the CPU. A CPU can have up to 8 run-time meters. The numbering starts at 0.

In "STOP" mode, the run-time meter is stopped. Its count value is retained even after a memory reset. During a "warm restart", the run-time meter must be restarted by the user program.

Programming run-time meters

You can program the runtime meters using the following instructions:

- Use instruction "SET_RTM" to set the runtime meter to an initial value.
- Use instruction "CTRL_RTM" to start or stop the runtime meter.
- Use the instruction "READ_RTM" to read out the current number of operating hours and the status of the counter - either "stopped" or "counting".

Use of clock bit memories (S7-300, S7-400)

Clock memory

A clock memory is a bit memory that changes its binary status periodically in the pulse-no-pulse ratio of 1:1.

You decide which memory byte of the CPU will become the clock memory byte when assigning the clock memory parameters.

Benefits

You can use clock memory, for example, to activate flashing indicator lamps or to initiate periodically recurring operations such as recording of actual values.

Available frequencies

Each bit of the clock bit memory byte is assigned a frequency. The following table shows the assignment:

Bit of the clock memory byte	7	6	5	4	3	2	1	0
Period (s)	2.0	1.6	1.0	0,8	0.5	0,4	0.2	0.1
Frequency (Hz)	0.5	0.625	1	1.25	2	2.5	5	10

Note

Clock memory runs asynchronously to the CPU cycle, i.e. the status of the clock memory can change several times during a long cycle.

The selected memory byte cannot be used for intermediate storage of data.

Web settings (S7-300, S7-400)

In this parameter group you can select:

- CPU Webserver start
- The languages in which texts are to be loaded (language-dependent texts are for example, diagnostics buffer entries and alarms).

Activating the Web server on this module

If the check box is selected, after the configuration data has been loaded, the Web server of the CPU is started and information can be read from the CPU via a Web browser.

Further settings relate to the update of web pages displayed. You can if required set the update time.

Only allow access via HTTPS

Enable this radio button for the encrypted transfer of web pages from the web server to the web browser.

Note

Access via HTTPS will only work if the time has been set in the CPU!

Selecting the languages

Select the national languages for language-dependent texts to be loaded to the CPU. The number of languages available is CPU-dependent.

Display and selection of CPU-specific languages

The display shows the languages which can be set for web access to the CPU. You can select a number of project languages for this "web server language" selection for a given CPU. Texts of diagnostics buffer entries and event texts are loaded into the CPU for the selected languages. For these languages, the information is also displayed on the Web browser in the corresponding language during operation.

If, during operation, you select a language in the web browser that has not yet been loaded, the diagnostics buffer entries and event texts will be shown in hexadecimal.

Note that each language requires memory space on the CPU.

User administration

You can define users for access to the CPU via the web browser and assign these users rights and passwords.

Rights can also be defined for the default user who does not log on to the web browser with a password ("everybody").

Monitoring tables

You can give users access via the web browser to the monitoring tables you define. Select the existing tables from the drop-down list and specify in the drop-down in the second column whether read-only or write access is to be possible.

Display classes of the messages

The selected message display classes are subsequently displayed on the CPU web pages.

You can reduce the memory occupied by the web SDBs by selecting only those display classes which are actually required.

The display class is set in the CPU properties "Message attributes" under "System diagnostics" in the inspector window.

User-defined web pages

The section "User-defined web pages" allows you to load your own web pages to the CPU and therefore make your own web applications available via the web browser.

See also

Information about the web server (Page 1244)

Creating and loading user pages (Page 1257)

Reservation of communications resources (S7-300, S7-400)

Assigning connection resources

Each communications connection requires a connection resource on the S7 CPU as a management element for the duration of the communications connection. As listed in the technical specifications, each S7 CPU has a certain number of connection resources available, which are assigned to the various communications services (programming device/OP communication, S7 communication or S7 basic communication).

Assignment of connection resources according to chronological order

When communications services log on, the connection resources are assigned according to the order in which they logged on.

Manual assignment of communications resources

In order to avoid allocation of the connection resources being dependent only on the chronological order in which various communication services are requested, connection resources can be reserved for the following services:

- Programming device communication and OP communication
- S7 basic communication

At least one connection resource is reserved by default for programming device communication and for OP communication. Smaller values are not possible.

Other communications services such as S7 communication with the extended instructions "PUT" and "GET" cannot occupy this connection resource, even if they establish their connection ahead of time. Instead, they are assigned other available connection resources, i.e. those that have not been specially reserved for a service.

Information on the maximum number and already occupied connection resources

The CPU properties display the following additional information regarding reservation of connection resources (read-only):

- Number of resources already occupied by S7 communication. These are resources such as for one-way S7 connections from another CPU to this CPU or for two-way S7 connections. These connections have already been configured in the network view.
- Maximum number of connection resources for this CPU.

Test, commissioning and routing (S7-300, S7-400)

Test, commissioning, routing

Select the check box if, for example, you want to use PG functions needed for commissioning and testing regularly over this interface.

If you select the "Test, commissioning, routing" option, the interface becomes the active device on PROFIBUS (in other words, the interface is included in the token rotation of PROFIBUS). The following functions are then possible:

- Programming (for example loading)
- Testing (status/modify)
- S7 routing (I-slave as gateway)

This can extend the token rotation time. In time-critical applications and when S7 routing and client functionality are not required for communication, this option should therefore not be enabled.

If the "Test, commissioning, routing" check box is not selected, the interface functions only as a server for communication services.

Server functionality means that only the partner (for example PG, OP or automation system) can initiate communication. Note the following features when the "Test, commissioning, routing" option is not enabled:

- S7 routing is not possible
- Programming and testing are possible but slow (no extension of the bus token rotation time)

Settings for system diagnostics (S7-300, S7-400)

The diagnostics system is the recording, evaluation and reporting of errors within the automation system.

Examples of errors

- CPU program error
- Failures of modules
- Wire break for sensors and actuators

Report cause of STOP

Select the check box to display the reason why a CPU changed to "STOP" mode on the display device (programming device or OP).

The alarm is also entered in the diagnostics buffer.

Alarms triggered by acknowledgment of the ALARM extended instruction

This parameter is relevant to CPUs that are used for process control systems. When this function is enabled, the extended instructions "ALARM", "ALARM_8" and "ALARM_8P" (SFBs 33 to 35) only report a signal change again if the previous signal change (i.e., the previous incoming alarm) was acknowledged. This prevents the accumulation of alarms that cannot be processed.

Setting options for the level of protection (S7-300, S7-400)

Protection level

This section explains how to use the individual protection levels of the S7 CPUs and what influence the type of S7 CPU has on the parameter assignment and use of the respective protection level.

Default behavior

There is no password assignment in the default protection level behavior. This level of protection means "No protection".

A password can only be given if you have selected the option "Removable with password". This option gives you write access to the CPU in RUN mode if the CPU has been write-protected with the "PROTECT" instruction in the user program.

Note

If the default protection level of your CPU is active and your CPU supports the instruction "PROTECT", you can use this instruction to switch between the default protection level "No protection" and the protection level "Write protection".

Additional password-protected protection levels

A password is required for the following protection levels:

- Write protection: With this protection level, only read-only access is available without entering the correct password, regardless of the keyswitch setting. To enable this protection level a password is required.
- Write/read protection: With this protection level, neither read nor write access is available without entering the correct password, regardless of the keyswitch setting. A password must be assigned before this protection level can be selected.

Write and read access is possible if the password is known regardless of the level of protection set.

Behavior of a password-protected module during operation

The CPU protection takes effect after the settings are downloaded in the CPU.

Before an online function is executed, the necessary permission is checked and, if necessary, the user is prompted to enter a password.

Example: The module was assigned a protection level and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on with a password.

Access authorization to the protected data is in effect for the duration of the online connection, or until the access authorization is manually rescinded with "Online > Delete access rights".

Note

You can not restrict functions for process control, monitoring, and communications. Thus, for example, the "Set time of day/date" function cannot be locked with a password.

Test mode and process mode (S7-300, S7-400)

Introduction

For some CPU models you can set the mode for the program test.

In process operation, test functions such as program status or monitor/modify tags are restricted so that the set permissible cycle time extension is not exceeded. Testing using breakpoints and step-by-step program execution cannot be performed.

In test operation, all test functions via a programming device/PC – even if they increase the cycle time significantly – can be used.

Possible settings for operating behavior

You can set the following modes for testing with the program status in the inspector window under "Properties > Mode":

- Test mode
  All test functions can be used without restrictions. The CPU cycle time can be extended considerably since, for example, the status of instructions in programmed loops is checked each time they are run through.
- Process mode
  The program status test function is restricted to ensure that the cycle time load is kept as low as possible:
  - For example, no call conditions are permitted.
  - The status display of a programmed loop is stopped at the return point.
  - The "HOLD" and step-by-step program execution test functions cannot be used.

Note

If the CPU is in test operation you must ensure that the CPU and the process can handle significant cycle time increases.

The "process operation" option allows you to prevent changes to the call environment being made in the program editor.

Parameters of the compact CPU (S7-300, S7-400)

Counting function (S7-300, S7-400)

Properties of Counting (S7-300, S7-400)

Description

If you select the CPU with integrated count and measurement function in the network view or the device view, you can configure the counting function for each of the channels listed in the inspector window under "Properties > General > Count".

If you decide to use channel 0 or channel 1 as the counting channel you can no longer use "Positioning" technology.

Note

The number of channels you can utilize as counting channels depends on the type of CPU you are using.

In the "Operating Mode" selection box, you can determine how you want the CPU to count for each channel. There are various options for this. Default is "not configured".

See also

Documentation on technological functions of the CPU (http://support.automation.siemens.com/WW/view/de/12429336/0/en)

Interrupt selection (S7-300, S7-400)

You can select several interrupts for the "Counting" function.

Requirement

- The CPU is selected.
- Interrupt selection is located at "Properties > General > Counting > Interrupt selection".

Interrupt selection (basic parameters)

In the "Select interrupt" drop-down list, select which interrupts the CPU should trigger. You have the following options:

- None
- Diagnostics
- Process
- Diagnostics and hardware

Default selection is "None".

Select the events you want to use for hardware interrupt triggering, dependent on the set operating mode.

Counting

Events that can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)
- HW gate closing (with opened SW gate)
- On reaching (activating) the comparator
- On a counting pulse
- Overflow (when the high count limit is exceeded)
- Underflow (when the low count limit is fallen below)

Frequency measurement

Events that can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)
- HW gate closing (with opened SW gate)
- End of measurement
- High limit violated
- Low limit violated

Pulse-width modulation

Events that can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)

Channel-specific parameters (S7-300, S7-400)

General

All channel-specific parameters are set under "Properties > General > Counting > Channel 0 (... Channel n)". There are six options available for this operating mode: "Not configured", "Count continuously", "Count once", "Count periodically", "Measure frequency" and "Pulse-width modulation".

"Not configured" is set as the default.

Count (S7-300, S7-400)

Operating parameters for counting (S7-300, S7-400)

Main counting direction

In "Count once" and "Count periodically" modes, you can specify from the "Main count direction" drop-down list whether or not you want the counter to have a main count direction and, if so, whether the main count direction is up or down. Default selection is "None".

The "Main counting direction" drop-down list is not shown in the "Count continuously" operating mode.

Gate Function

In the "Gate function" drop-down list, you can select whether the count operation is to be canceled or interrupted when the gate is closed.

Here, if you select "Cancel count", the count is resumed with the load value after the gate is closed and opened again. Here, if you select "Stop count", the count is resumed with the last actual counter value after the gate is closed and opened again.

Default selection is "Cancel count".

Initial value / End value

In the Initial value / End value input field, enter a value for the respective counting limit, dependent on which main count direction you have selected in the respective drop-down list.

The main count direction and this entry are related as follows:

- If the main count direction is up, enter an end value here.
- If the main count direction is down, enter an initial value here.

This input field is not shown if you have not selected a main count direction.

The default setting always has the value "2147483647" (only visible if you have selected a main counting direction).

Compare Value

In the "Comparison value" input field, enter a value that is assigned to the digital output, to the comparator status bit STS_CMP and to a hardware interrupt. The count value is compared with the comparison value. Dependent on the result of this comparison, the output and the comparator status bit STS_CMP can be set and the hardware interrupt for "Reached comparison value" can be generated.

In the output group, customize how the output is actually set dependent on the counter value.

Default comparison value is "0".

Hysteresis

In the input field "Hysteresis" enter a hysteresis range of 0 to 255 (0 and 1 means: hysteresis is switched off).

An encoder might stop at a certain position and then "swing" up / down in this position. As a result, the counter value fluctuates around a certain value. Thus, the associated output would be switched on / off, dictated by the rhythm of those fluctuations if, for example, a comparison value lies within this fluctuation range. In order to prevent a switching action as a result of small fluctuations, the CPU offers a configurable hysteresis.

You can modify this hysteresis in a current operation via the user program.

The hysteresis influences the output and the comparator status bits STS_CMP, the overflow STS_OFLW, underflow STS_UFLW and the zero crossing STS_ZP.

Default is "0".

Time base

You can specify whether the period is to be measured in units of 125 ns or 1 µs at a maximum counting frequency of 1 kHz. If the maximum counting frequency is greater than 1 kHz, the period is not measured. You can measure the period of the counting signal up to a maximum counting frequency of 1 kHz.

At a maximum counting frequency of 1 kHz, the time between two consecutive counting edges is always measured.

With a time base of 1 µs, periods of up to 4294 s (71 min, 34 s) can be measured and with a time base of 125 ns, periods of up to 178 ms (2 min, 58 s) can be measured.

If the counting edges are further apart in time, the measured period is incorrect because a overflow has not been taken into account.

Input 0 (count) (S7-300, S7-400)

Signal evaluation

In the "Signal evaluation" drop-down list, select which signals the connected encoder is to deliver. You can choose from the following options:

- Pulse and direction
- Rotary encoder single
- Rotary encoder double
- Rotary encoder quadruple

The default setting is "Pulse and direction".

HW gate

The "HW gate" check box allows you to select whether or not a hardware gate is to be used.

This check box is not selected by default.

Count Direction Inverted

The "Count direction inverted" check box allows you to select whether or not the count direction is to be inverted.

This check box is not selected by default.

Output (count) (S7-300, S7-400)

Output function course

The "Output function course" drop-down list offers you the following alternative options:

- No comparison
  The output will be switched in the same way as a normal output.
  The SFB input parameters CTRL_DO and SET_DO are not active.
  The status bit STS_DO and STS_CMP (status comparator in the IDB) remain reset.
- Set when count ≥ comparison value
  The output is set when the counter value reaches the compare value. It stays set as long as the counter value is greater than / equal to the compare value.
- Set when count ≤ comparison value
  The output is set when the counter value reaches the compare value. It stays set as long as the counter value is less than / equal to the compare value.
- Pulse when comparison value is reached
  You can specify a pulse period for the adaptation to the actuators you are using. The output is set for the specified pulse period when the counter value reaches the compare value. The output is only set when it reaches the compare value of the main counting direction, if you have configured a main counting direction.

Default is "No comparison".

Pulse Period

If you have selected the "Pulse when comparison value is reached" alternative in the "Output function course" drop-down list, you must enter the pulse period for the output signal in ms in the "Pulse period" input field. The maximum length is 510 ms. You can only enter even values.

The pulse period starts when the respective digital output is set. The pulse period inaccuracy is lower than 2 ms.

The output stays set until the compare conditions do not apply anymore if you have entered a 0-ms pulse period.

Default is "0 ms".

Note

The bit STS_CMP "Comparator Status" is set together with the output when the compare value is reached.

Max. frequency (S7-300, S7-400)

Counting signals / HW gate

In this drop-down list you can set the maximum counting frequency for the track A / pulse, track B / direction and hardware gate signals in fixed intervals. The maximum value depends on the CPU in use:

CPU	Range of values	Default selection
CPU 312C	10, 5, 2, 1 kHz	10 kHz
CPU 313C CPU 313C-2 DP CPU 313C-2 PtP	30, 10, 5, 2, 1 kHz	30 kHz
CPU 314C-2 DP CPU 314C-2 PN/DP CPU 314C-2 PtP	60, 30, 10, 5, 2, 1 kHz	60 kHz

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Latch

In this drop-down list you can set the maximum counting frequency for the latch signal in fixed intervals. The maximum value depends on the CPU in use:

CPU	Range of values	Default selection
CPU 312C	10, 5, 2, 1 kHz	10 kHz
CPU 313C CPU 313C-2 DP CPU 313C-2 PtP	30, 10, 5, 2, 1 kHz	10 kHz
CPU 314C-2 DP CPU 314C-2 PN/DP CPU 314C-2 PtP	60, 30, 10, 5, 2, 1 kHz	10 kHz

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Hardware interrupt at (count) (S7-300, S7-400)

Description

The corresponding event triggers a hardware interrupt when you select the respective button. The following events can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)
- HW gate closing (with opened SW gate)
- On reaching (activating) the comparator
- On a counting pulse
- Overflow (when the high count limit is exceeded)
- Underflow (when the low count limit is fallen below)

You must enable this in the Basic Parameters tab to ensure that the CPU generates a hardware interrupt.

No option is selected by default.

Triggering a hardware interrupt on each counting edge results in high CPU utilization at higher counting frequencies. If the hardware interrupts in the "Counting" submodule occur faster than they can be processed in the hardware interrupt OB (OB 40), this produces the diagnostic "Hardware interrupt lost), provided the diagnostic interrupt is enabled.

Assigning input data (S7-300, S7-400)

Assigning input data

You can select whether the count value or the period can be read at a maximum counting frequency of 1 kHz in the input data (I data) of the "Counting" submodule. If the maximum counting frequency is greater than 1 kHz, only "Count value" is possible.

You can measure the period of the counting signal up to a maximum counting frequency of 1 kHz.

At a maximum counting frequency of 1 kHz, the time between two consecutive counting edges is always measured. If the maximum counting frequency is greater than 1 kHz, the period is not measured and the value is 0.

Frequency measurement (S7-300, S7-400)

Measure Frequency (S7-300, S7-400)

Description

In this operating mode the CPU counts the pulses it receives with default time intervals (integration time) and calculates the frequency via the number of pulses and integration time.

The calculated frequency value is supplied in "mHz" units.

Operating parameters for frequency measurement (S7-300, S7-400)

Description

In this operating mode the CPU counts the pulses it receives within a configured integration time.

The measurement is carried out within this integration time. The measured value is updated when the integration time has expired.

If at least one rising edge does not occur during the integration time, a value of 0 or the averaged value is returned as the measured value, depending on the parameter assignment.

Integration time

In the "Integration time" input field, enter an integration time between 10 ms and 10000 ms.

Default is "100 ms".

Low limit and high limit

In the "Low limit" and "High limit" input fields, enter the values that the determined frequency is compared with, respectively, after the integration time has expired. The following are permissible ranges for limit value monitoring, dependent on the type of CPU:

- Low limit
  - CPU 312C: 0 to 9 999 999 mHz
  - CPU 313C: 0 to 29 999 999 mHz
  - CPU 314C: 0 to 59 999 999 mHz
- High limit
  - CPU 312C: 1 mHz to 10 000 000 mHz
  - CPU 313C: 1 mHz to 30 000 000 mHz
  - CPU 314C: 1 mHz to 60 000 000 mHz

  Note

  At the time of input it is verified whether the high limit > low limit.

Default for the low limits is 0 mHz respectively. CPU-dependent default high limits are:

- CPU 312C: 10 000 000 mHz
- CPU 313C: 30 000 000 mHz
- CPU 314C: 60 000 000 mHz

Max. counting frequency / HW gate

In this drop-down list you can set the maximum counting frequency for the track A / pulse, track B / direction and hardware gate signals in fixed intervals. The maximum value depends on the CPU in use:

CPU	Range of values	Default selection
CPU 312C	10, 5, 2, 1 kHz	10 kHz
CPU 313C CPU 313C-2 DP CPU 313C-2 PtP	30, 10, 5, 2, 1 kHz	30 kHz
CPU 314C-2 DP CPU 314C-2 PN/DP CPU 314C-2 PtP	60, 30, 10, 5, 2, 1 kHz	60 kHz

Output measured value

You can select whether the measured value is to be output directly or as an averaged value. To do this, select the "Directly" or "Averaged" option in the "Output measured value" drop-down list.

The measured frequency is displayed at the end of an integration time within which at least one positive edge has occurred.

If at least one positive edge does not occur during the assigned integration time:

- for direct frequency value, "0" is returned at the end of the integration time.
- for averaged frequency value, the last value is exported to subsequent measurement intervals without positive edge (f ≥ 1 mHz). This operation represents an extension of the integration time. Here, the last measured value is divided by the number of measurement intervals without positive edge.
  Example: If the last measured value was 12 000 mHz, the value 4000 mHz is output after three measurement intervals.

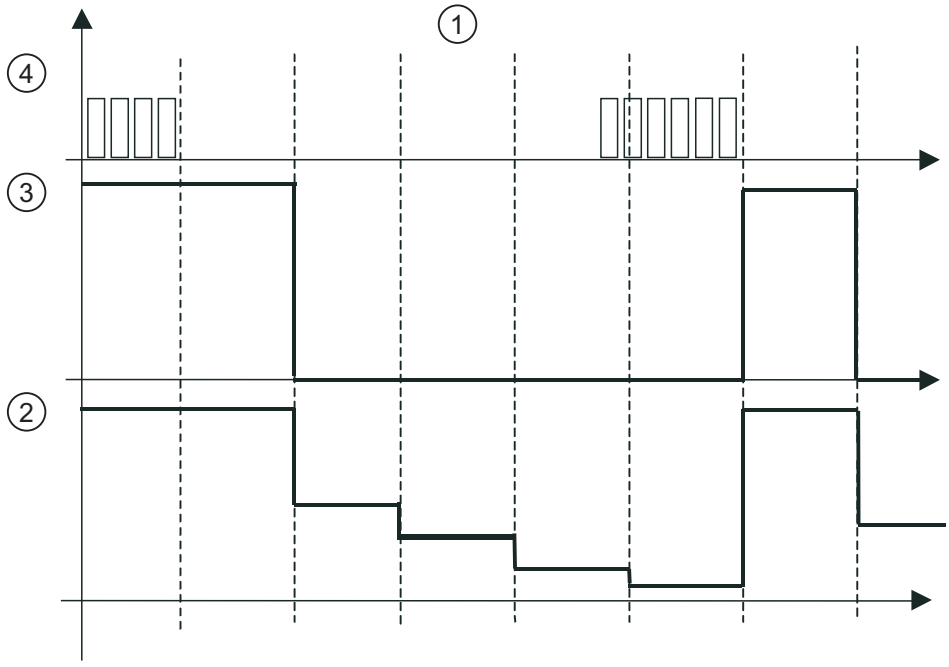

① Time interval
② Frequency averaged
③ Frequency direct
④ Pulses

The "Direct" option is selected as the default in the drop-down list.

Input 0 (frequency measurement) (S7-300, S7-400)

Signal evaluation

In the "Count direction inverted" drop-down list, select which signals the connected encoder is to supply. You can choose from the following options:

- Pulse and direction
- Rotary encoder single

The default setting is "Pulse and direction".

HW gate

The "HW gate" check box allows you to select whether or not a hardware gate is to be used.

This check box is not selected by default.

Count Direction Inverted

The "Count direction inverted" check box allows you to select whether or not the count direction is to be inverted.

This check box is not selected by default.

Output (frequency measurement) (S7-300, S7-400)

Output function course

The "Output function course" drop-down list offers you the following alternative options:

- No comparison
  The output will be switched in the same way as a normal output.
  The SFB input parameters MAN_DO and SET_DO are not active.
  The status bit STS_DO remains reset.
- Set when outside the limits
  The output is set when the measured frequency is higher than the upper limit or lower than the lower limit.
- Set when lower limit is violated
  The output is set when the measured frequency drops below the lower limit.
- Set when upper limit is violated
  The output is set when the measured frequency exceeds the upper limit.

Default is "No comparison".

Hardware interrupt at (frequency measurement) (S7-300, S7-400)

Description

The corresponding event triggers a hardware interrupt when you select the respective check box. The following events can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)
- HW gate closing (with opened SW gate)
- End of measurement
- High limit exceeded
- Low limit violated

You must enable this in the Basic Parameters tab to ensure that the CPU generates a hardware interrupt.

No option is selected by default.

Pulse-width modulation (S7-300, S7-400)

Pulse-Width Modulation (S7-300, S7-400)

Description

The output value you have specified is converted by the CPU into a pulse sequence of the respective pulse/pulse ratio (pulse-width modulation).

This pulse sequence is output on digital output DO (output sequence) after the parameterized on-delay has expired.

Operating parameters for pulse width modulation (S7-300, S7-400)

Output format

In the"Output format" drop-down list, select the value range of the output value by selecting the "ppm" or "S7 analog value" option.

If your output value lies between 0 and 1,000 check the "ppm" option.

If your output value is a SIMATIC S7 analog value (between 0 and 27,648), check the "S7 analog value" option.

The "ppm" button is selected by default.

Time base

In the "Time base" drop-down list, select the time base valid for the resolution, value range of the period, minimum pulse width and for the on-delay.

If you select the "1 ms" option, you can set times with a resolution of 1 ms.

Select "0.1 ms" option to configure times with a resolution of 0.1 ms.

The "0.1 ms" option is selected by default.

On delay

In the "On delay" input field, enter a value for the delay time between the start of an output sequence and the output of the pulse.

Enter a value as in the resolution configured above.

If you select the "1 ms" option in the "Time base" drop-down list, you can set times from 0 ms to 65535 ms with a resolution of 1 ms.

If you select the "0.1 ms" option in the "Time base" drop-down list, you can set times from 0 ms to 6553.5 ms with a resolution of 0.1 ms.

Default is "0".

This pulse sequence is output on digital output DO (output sequence) after the on-delay has expired.

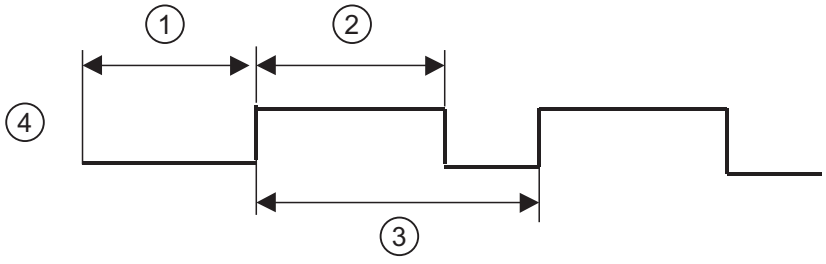

① On delay
② Pulse Period
③ Period
④ Output DO

Default setting is "0".

Period

The pulse / pause period defines the length of the output sequence.

In the "Period" input field, enter a value as in the resolution configuration above.

If you select the "1 ms" option in the "Time base" drop-down list, you can set times from 1 ms to 65535 ms at a resolution of 1 ms.

If you select the "0.1 ms" option in the "Time base" drop-down list, you can set times from 0.4 ms to 6553.5 ms with a resolution of 0.1 ms.

Default setting is "20000".

Minimum pulse time

You can set a minimum pulse time for the attenuation of short output pulses/pauses. This suppresses all pulses or pauses shorter than the minimum pulse time.

In the "Minimum pulse time" input field, enter a value as in the resolution configuration above.

If you select the "1 ms" option in the "Time base" drop-down list, enter period values from 0 to /2.

If you select the "0.1 ms" option in the "Time base" drop-down list, enter period values from 2 to /2.

The minimum pulse width is internally set to 0.2 ms if you have selected 1 ms for the time base and a minimum pulse time of 0.

Default setting is "2".

Input 0 (pulse-width modulation) (S7-300, S7-400)

HW gate

The "HW gate" check box allows you to select whether or not a hardware gate is to be used.

This check box is not selected by default.

Start of pulse-width modulation with a hardware gate is only possible if you first open the software gate and then generate a positive edge on digital input "hardware gate".

Pulse width modulation can only be stopped with a negative edge at the software gate. The hardware gate status has no effect.

Filter frequency (S7-300, S7-400)

HW gate

This drop-down list is only available if the "HW gate" option is selected under "Input". Here you can set the maximum frequency of the hardware gate signal in fixed intervals. The maximum value depends on the CPU in use:

CPU	Range of values	Default selection
CPU 312C	10, 5, 2, 1 kHz	10 kHz
CPU 313C CPU 313C-2 DP CPU 313C-2 PtP	30, 10, 5, 2, 1 kHz	30 kHz
CPU 314C-2 DP CPU 314C-2 PN/DP CPU 314C-2 PtP	60, 30, 10, 5, 2, 1 kHz	60 kHz

Hardware interrupt at (pulse-width modulation) (S7-300, S7-400)

Description

The corresponding event triggers a hardware interrupt when you select the respective check box. The following events can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)

You must enable this in the Basic Parameters tab to ensure that the CPU generates a hardware interrupt.

No option is selected by default.

Continuous Counting (S7-300, S7-400)

Description

The "Count continuously" operating mode is defined as follows:

After the counter reaches its upper limit it jumps to the lowest counting limit at the next count pulse and resumes the count from there.

After the counter reaches its lower limit it jumps to the upper counting limit at the next count pulse and resumes the count from there.

Those counting limits are set fixed to the maximum counting range.

In this operating mode, counting is started and stopped via the gate function.

Count once (S7-300, S7-400)

Description

When the counter reaches the end value 1, in the up-counting main counting direction mode, the counter jumps to the counter range start at the next pulse and stops, even if there are additional count pulses coming in.

When the counter reaches the end value 1, in the down-counting main counting direction mode, the counter jumps to the counter range start at the next pulse and stops, even if there are additional count pulses coming in.

The counting limits are set to the maximum counting range if there is no master counting direction configuration.

In this operating mode, counting is started and stopped via the gate function.

When the gate opens, the counter can receive counting impulses and the count starts. When the gate is closed, the counter can no longer receive count pulses and the count stops.

Count periodically (S7-300, S7-400)

Description

If the main count direction of the counter is up and the counter has reached the end value - 1, the arrival of an additional count pulse causes the counter to jump to the load value from where it continues counting.

If the main count direction of the counter is down and the counter has reached the count value = 1, the arrival of an additional count pulse causes the counter to jump to the start value (load value) from where it continues counting.

The count limits are set to the maximum counting range if a main count direction is not configured.

In this operating mode, counting is started and stopped via the gate function.

When the gate opens, the counter can receive count pulses and the count starts. When the gate is closed, the counter can no longer receive count pulses and the count stops.

Main counting direction (S7-300, S7-400)

Description

In your parameters you can declare "up" or "down". This determines the counting limit for the start value or end value for the operating modes Single Counting and Periodic Counting.

For main counting direction "up", the counter starts at "0" or at the load value and counts up to the limit value you have declared in your parameters.

For the main count direction "down", the counter starts at the start value you have declared in your parameters and then counts down to "0".

Even though you may have declared "down" counts in your parameters for the main counting direction, you must still set a respective direction signal for reverse counting or you must declare "Invert direction of count" in your parameters.

Gate Function (S7-300, S7-400)

Software gate and Hardware gate

The CPU has two logical AND linked gates:

A **software gate** (SW gate), controlled via the user program.

The software gate can only be opened on a 0-1 edge transition of the input parameters SW_GATE on the SFB. A reset of this parameter closes the gate.

A **hardware gate** (HW gate), controlled via the digital input "Hardware gate". In the "Count" tab you can specify whether or not you want to use the hardware gate. This gate opens at a 0-1 transition and closes at a 1-0 transition on the digital input.

Positioning function (S7-300, S7-400)

Interrupt selection (S7-300, S7-400)

You can select interrupts for the "Positioning" function.

Requirement

- The CPU is selected.
- Interrupt selection is located at "Properties > General > Positioning > Interrupt selection".

Interrupt selection

In the "Interrupt Selection" drop-down list, select whether or not the CPU should trigger diagnostic interrupts. You have the following options:

- None
- Diagnostics

Default selection is "None".

You must carry out supplementary configurations to ensure that the corresponding events are really going to trigger diagnostic interrupts.

Enable the required monitoring under "Properties > Positioning > Channel 0" with operating mode "Positioning with analog output" selected:

- Under "Drive"
  Actual value, target approach and target range can be monitored.
- Under "Axis"
  For a linear axis, you can specify here whether or not to monitor the working range. The traversing range is always monitored.
- Under "Encoder"
  For encoders with zero mark, it is verified whether the encoder always supplies the same number of pulses between two consecutive zero marks.

Select which events are to trigger a diagnostic interrupt under "Channel 0 - Diagnostics".

For a diagnostic interrupt, you must configure the parameters in the following settings:

- Interrupt selection under "Properties > General > Positioning > Channel 0"
- The corresponding monitoring in the appropriate "Drive", "Axis" or "Encoder" section.
- The respective enable signal in the "Diagnostics" section.

Operating mode (S7-300, S7-400)

Operating mode

Use the "Operating mode" parameter to determine how you want to position. You have the following options:

- Analog outputs: Positioning with analog output
- Digital outputs: Positioning with digital outputs

Default selection is "not configured".

The operating mode is selected under "Properties > General > Positioning > Channel 0".

Note

You cannot configure the "Positioning" circuit if you have configured channel 0 or 1 for the counting circuit.

You can only assign an analog output to positioning if you have disabled output 0 in section AI5/AO2.

I/O channels configured for positioning can no longer be addressed by the user program.

See also

Documentation on technological functions of the CPU (http://support.automation.siemens.com/WW/view/de/12429336/0/en)

Positioning with analog output (S7-300, S7-400)

Drive (S7-300, S7-400)

Controlling the drive and control mode (S7-300, S7-400)

Controlling the drive and control mode

In "Properties > Position > Channel 0" in the "Channel 0 - Drive" section, specify the parameters for the drive and axis for positioning using an analog output.

Drive description

The drive is controlled via two fixed assigned analog outputs, with either a voltage of ±10 V (Pin 16) or a current of ±20 mA (Pin 17).

Note

You can only assign an analog output to positioning if you have disabled output 0 in section AI5/AO2 of the area navigation.

Description of control mode

In the drop-down list you have the option of 2 steer modes. The control mode describes how your connected power unit is controlled.

- Voltage ±10 V or current ±20 mA:

A positive voltage or current is generated when traveling in the plus direction (forward). A negative voltage or current is generated when traveling in the minus direction (back).

- Voltage 0 to 10 V or current from 0 to 20 mA and direction signal:

A positive voltage or current is generated when traveling in the plus direction (forwards) and the CONV_DIR digital output is switched off.

A positive voltage or current is generated when traveling in the minus direction (back) and the CONV_DIR digital output is switched on.

The default setting is "±10".

Max. frequency (S7-300, S7-400)

Position feedback

In this drop-down list you can set the maximum counting frequency for the position feedback signals (encoder signals A, B, N) in fixed levels.

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Default setting here is "60 kHz".

Accompanying signals

In this drop-down list you can set the maximum counting frequency for the length measurement and reference point switch signals in fixed levels.

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Default setting here is "10 kHz".

Target approach (analog) (S7-300, S7-400)

Target range

In the "Target range" input field, enter an even-numbered value between 0 and 2 x 10 pulses. An odd value is rounded off internally. The target range lies symmetrically around the target. Here, if you enter the value 0, the target must be reached with the accuracy of a pulse. The target range monitoring is switched off and the relevant check box in the "Monitoring" area is unavailable.

Default is "50" pulses.

Monitoring time

In the "Monitoring time" input field, enter a time between 0 and 100000 ms. The CPU rounds this value to a 4 ms execution cycle.

The CPU uses the monitoring time to monitor the actual value and the target approach.

If you enter the value 0, the actual value monitoring and target approach monitoring are switched off and the relevant check boxes in the "Monitoring" area are then unavailable.

Default selection is "2000 ms".

Maximum speed

In the "Maximum speed" input field, enter a value between 10 and 1 000 000 pulses/s. This corresponds to a level of 10 V or 20 mA at the analog output.

Default is "1000 pulses/s".

Creep speed / reference speed

In the "Creep-/referencing speed" input field, enter a value between 10 pulses/s and the assigned maximum speed.

The drive is decelerated to creep speed after it reaches the braking point.

In the reference point approach mode, the drive is decelerated to reference speed when it reaches the reference point switch.

Default is "100" pulses/s.

Enable power section (S7-300, S7-400)

Off delay

In the "Off delay" input field, enter a time in milliseconds. The CPU rounds this value to a 4 ms execution cycle. Output CONV_EN is reset after a delay of this amount of time when a run is canceled (power section enable is removed).

If you actuate a brake via the digital output, the delay time ensures that the axis, at the time the output is reset, is slow enough for the brake to handle the kinetic energy.

Default is "1000 ms".

Monitoring (analog) (S7-300, S7-400)

Description

Here you can switch process value, target approach and target range monitoring on and off.

Target range

Once you have selected the "Target range" button to monitor whether the drive remains in the approached target position or drifts off after it has reached this target position.

You cannot switch on target range monitoring if you have customized a 0 ms monitoring time for "Target approach".

An external error message is returned when the monitoring unit is triggered. You must acknowledge this external error with ERR_A. Monitoring is switched off. Monitoring is not switched on until a new run starts.

This monitoring function is disabled by default.

Target approach

Once you have selected the "Target approach" button, it is monitored whether the axis has approached the target range within the monitoring time after it has reached the switch-off point.

You cannot switch on process value monitoring if you have customized a 0 ms monitoring time for "Target approach".

This monitoring function is disabled by default.

Process value

Axis movement is monitored for the run start to the switch-off point if you have selected the "Process value" button. The run is interrupted if the axis does not move by at least one pulse into the specified direction within the monitoring time.

You cannot switch on process value monitoring if you have customized a 0 ms monitoring time for "Target approach".

The CPU does not recognize the failure of a digital input. Process value monitoring lets you detect missing encoder signals. This error can have the following causes:

- Digital input failure
- Wire break
- Defective encoder
- Faulty power section

This monitoring feature is switched on by default.

Axis (S7-300, S7-400)

Axis type (analog) (S7-300, S7-400)

Description

Specify the parameters for the axis under "Properties > Positioning > Channel 0" in the "Channel 0 - Axis " section.

Axis type

In "Axis type" select whether to control a linear axis or a rotary axis.

Default selection is "Linear axis".

Start and end of software limit switch

The working area of a linear axis is between the start of the software limit switch and the end of the software limit switch. The software limit switches belong to the working area. Enter values between -5 x 10^8 and +5 x 10^8 pulses in the relevant boxes. The start of the software limit switch must always be less than the end of the software limit switch.

The software limit switches are monitored, provided the axis is synchronized and working range monitoring is enabled.

Default is "-100000000 pulses" for the start of the software limit switch and "100000000 pulses" for the end of the software limit switch.

End of rotary axis

In the "End of rotary axis" box, enter a value between 1 and 10^9 pulses.

The "End of rotary axis" value is the theoretically largest value that the actual value can reach. Physically, it has the same position as the start of the rotary axis (0).

Default is "100000" pulses.

Length measurement (analog) (S7-300, S7-400)

Description

The "Length measurement" function allows you to measure a distance during a run operation. The start and end of the length measurement are edge-triggered via the digital input "Length measurement". Select the type of edge in the "Detection" drop-down list.

You have the following options:

- Off
- Start/end on rising edge
  A rising edge on the digital input starts the length measurement, the next rising edge ends it.

- Start/end on falling edge
  A falling edge on the digital input starts the length measurement, the next falling edge ends it.
- Start on rising edge, end on falling edge
  A rising edge on the digital input starts the length measurement, the next falling edge ends it.
- Start with falling edge, end with rising edge
  A falling edge on the digital input starts the length measurement, the next rising edge ends it.

Default is "off".

Reference point (analog) (S7-300, S7-400)

Coordinate

After the CPU performs a STOP-RUN transition, the axis is set to the value of the reference point coordinate while the axis has not yet synchronized.

After a reference point run the value of the reference point coordinate is assigned to the reference point.

For a linear axis, the value of the reference point coordinate must lie within the working range (including the start / end of the software limit switches).

For a rotary axis, the value of the reference point coordinate must lie within the range from 0 to "End of rotary axis - 1".

Default is "0" pulses.

Reference Point Location for Reference Point Switch

In the "Reference point location for reference point switch" drop-down list, select whether the reference point lies in the plus direction (increasing actual values) or minus direction (decreasing actual values) relative to the reference point switch.

Default is "in plus direction".

Axis monitoring (analog) (S7-300, S7-400)

Working range

For a linear axis, you can specify here whether or not to monitor the working range. If you select the "Working range" button, a check is made to determine whether the actual position value is out of range of the software limit switches. Monitoring is only active on a synchronized axis.

The coordinates of the software limit switches as such belong to the working range.

If monitoring is triggered the run is aborted.

This monitoring function is enabled by default.

Traversing Range

You use traversing range monitoring to check whether the value is outside the permitted traversing range of - 5 x 10^8 to + 5 x 10^8. The traversing range is always monitored. The monitoring cannot be disabled (always enabled in the "Monitoring" parameter).

If monitoring is triggered the run is aborted.

Linear axis (S7-300, S7-400)

Description

The **linear axis** is an axis whose physical traversing range is restricted.

The working area on the linear axis is limited by the software limit switches.

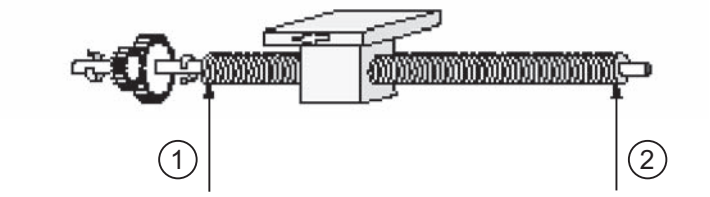

① Physical start
② Physical end

Rotary axis (S7-300, S7-400)

Description

The **rotary axis** is an axis the traversing range of which is not restricted by mechanical stops.

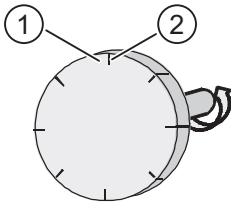

① Largest display value = end of rotary axis - 1
② Start of rotary axis (coordinate 0) = end of rotary axis

A rotary movement of the rotary axis starts at the coordinate "Zero" and ends at the coordinate "End of rotary axis". Physically, the "zero" coordinate is identical with "End of rotary axis" (= 0). The actual position value is toggled at this point. The display of the actual position is always positive.

Encoder (S7-300, S7-400)

Increments per encoder revolution

In the "Increments per encoder revolution" input field, enter the number of increments your encoder should output per 360 degrees. You can find this value in the encoder description.

The CPU evaluates the increments 4 times, in other words, an increment corresponds to four pulses.

Default setting is "1000" increments.

Count direction

In "Count direction", adapt the direction of the position feedback to the axis motion direction.

In the "count direction" drop-down list, select the "Normal" option if rising count pulses correspond to increasing actual position values.

In the "count direction" drop-down list, select the "Inverted" option if rising count pulses correspond to decreasing actual position values.

You should consider all the directions of rotation of the transmission elements, such as clutches and gears.

The "Normal" option is selected by default.

Monitoring

In "Monitoring" you can determine whether the encoder is to be monitored for missing pulses. This is only possible for encoders where the value of "Increments per encoder revolution" is divisible by 10 or 16.

If you select the "Missing pulse" check box, a check is performed to determine whether the encoder always supplies the same number of pulses between two consecutive zero marks.

If you use an encoder without zero mark, you must disable the monitoring for missing pulses.

This monitoring function is disabled by default.

Diagnostics (S7-300, S7-400)

Description

In "Properties > Position > Channel 0" in section "Channel 0 - Diagnostics", select the events for which the diagnostics interrupt is to be generated. No check box is selected by default.

Missing pulse

If you select the "Missing pulse" check box, a diagnostic interrupt is generated as soon as the encoder does not supply the same number of pulses between two consecutive zero marks. You must enable the corresponding monitoring function in the "Channel 0 - Encoder" section.

Traversing range

If you select the "Traversing range" check box, a diagnostic interrupt is generated immediately when the axis exits the traversing range. The corresponding monitoring function is always enabled.

Working range

If you select the "Working range" check box, a diagnostic interrupt is generated if the actual position value is out of range of the software limit switches. You must enable the corresponding monitoring function in the "Channel 0 - Axis" section.

You cannot use this diagnostic interrupt for positioning on a rotary axis. The respective options box is grayed out.

Actual value

If you selected the "Actual value" check box, a diagnostic interrupt is generated if during a traversing motion, the axis does not move by at least one pulse in the specified direction within the monitoring time. The run is cancelled.

The actual value is monitored from the start of travel until the switch-off point is reached.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

The CPU does not recognize the failure of a digital input. Enabling the actual value monitoring allows you to detect missing encoder signals.

This check box is selected by default.

Target approach

If you select the "Target approach" check box, a diagnostic interrupt is generated if the axis does not reach the target range within the monitoring time after it has reached the switch-off point.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

Target range

If you selected the "Target range" check box, a diagnostic interrupt is generated if the drive does not stop on an approached target position after the target range is reached or if the drive drifts away from the approached target position.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

Positioning with digital outputs (S7-300, S7-400)

Drive (S7-300, S7-400)

Controlling the drive (S7-300, S7-400)

In "Properties > Position > Channel 0" in the "Channel 0 - Drive" section, specify the parameters for the drive and axis for positioning with digital inputs.

Description

The drive is controlled via four permanently allocated 24 V digital outputs.

In the drop-down list you have the option of 4 steer modes. The control mode describes how the 4 digital outputs (DO+1.0 to DO+1.3) control the power section.

Below lists the individual output functions under the various control modes:

Output	Control mode 1	Control mode 2	Control mode 3	Control mode 4
DO + 1.0	Rapid traverse	Rapid / creep traverse	Rapid traverse	Rapid traverse plus
DO + 1.1	Creep traverse	Position reached	Creep traverse	Creep traverse plus
DO + 1.2	Drive plus	Drive plus	Drive plus	Rapid traverse minus
DO + 1.3	Drive minus	Drive minus	Drive minus	Creep traverse minus

Max. frequency (S7-300, S7-400)

Position feedback

In this drop-down list you can set the maximum counting frequency for the position feedback signals (encoder signals A, B, N) in fixed levels.

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Default setting here is "60 kHz".

Accompanying signals

In this drop-down list you can set the maximum counting frequency for the length measurement and reference point switch signals in fixed levels.

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Default setting here is "10 kHz".

Target approach (digital) (S7-300, S7-400)

Target range

Enter an even-numbered value between 0 and 2 x 10^8 pulses in the "Target range" input field. An odd value is rounded off internally. The target range lies symmetrically around the target. Here, if you enter the value 0, the target must be reached with the accuracy of a pulse. In this case, the target range monitoring is switched off and the relevant check box in the "Monitoring" area is unavailable.

Default is "50" pulses.

Monitoring time

Enter a time between 0 and 100000 ms in the "Monitoring time" input field. The CPU rounds this value to a 4 ms execution cycle.

The CPU uses the monitoring time to monitor the actual value and the target approach.

If you enter the value 0, the actual value monitoring and target approach monitoring are switched off and the relevant check boxes in the "Monitoring" area are then unavailable.

The default setting is "2000 ms".

Monitoring (digital) (S7-300, S7-400)

Description

Here you can switch actual value, target approach and target range monitoring on and off.

Target approach

If you select the "Target approach" check box, the axis is monitored to determine whether it reaches the target range within the monitoring time after the switch-off point is reached.

You cannot switch on actual value and target approach monitoring if you have set a 0 ms monitoring time for "Target approach". The respective fields are grayed out.

This monitoring function is disabled by default.

Target range

If you select "Target range" check box, the drive is monitored to determine whether it stops on an approached target position after the target range is reached or if the drive drifts away from the target range.

An external error is returned when the monitoring is triggered. You must acknowledge this external error with ERR_A. Monitoring is switched off. Monitoring is not switched on again until a new run starts.

This monitoring function is disabled by default.

Actual value

If you select the "Axis value" check box, the axis movement is monitored from the start of the run up until the switch-off point is reached. The run is canceled if the axis does not move by at least one pulse into the specified direction within the monitoring time.

You cannot switch on actual value and target approach monitoring if you have set a 0 ms monitoring time for "Target approach". The respective fields are grayed out.

The CPU does not recognize the failure of a digital input. Actual value monitoring lets you detect missing encoder signals. This error can have the following causes:

- Digital input failure
- Wire break
- Defective encoder
- Faulty power section

This monitoring function is enabled by default.

Axis (S7-300, S7-400)

Axis type (digital) (S7-300, S7-400)

Description

In "Axis type" select whether to control a linear axis or a rotary axis.

Default selection is "Linear axis".

Start and end of software limit switch

The working area of a linear axis is between the start of the software limit switch and the end of the software limit switch. The software limit switches belong to the working area. Enter values between -5 x 10^8 and +5 x 10^8 pulses in the relevant boxes. The start of the software limit switch must always be less than the end of the software limit switch.

The software limit switches are monitored, provided the axis is synchronized and working range monitoring is enabled.

Default is "-100000000 pulses" for the start of the software limit switch and "100000000 pulses" for the end of the software limit switch.

End of rotary axis

In the "End of rotary axis" box, enter a value between 1 and 10^9 pulses.

The "End of rotary axis" value is the theoretically largest value that the actual value can reach. Physically, it has the same position as the start of the rotary axis (0).

Default is "100000" pulses.

Length measurement (digital) (S7-300, S7-400)

Description

The "Length measurement" function allows you to measure a distance during a run operation. The start and end of the length measurement are edge-triggered via the digital input "Length measurement". Select the type of edge in the "Detection" drop-down list.

You have the following options:

- Off
- Start/end on rising edge
  A rising edge on the digital input starts the length measurement, the next rising edge ends it.
- Start/end on falling edge
  A falling edge on the digital input starts the length measurement, the next falling edge ends it.
- Start on rising edge, end on falling edge
  A rising edge on the digital input starts the length measurement, the next falling edge ends it.
- Start with falling edge, end with rising edge
  A falling edge on the digital input starts the length measurement, the next rising edge ends it.

Default is "off".

Reference point (digital) (S7-300, S7-400)

Coordinate

After the CPU performs a STOP-RUN transition, the axis is set to the value of the reference point coordinate while the axis has not yet synchronized.

After a reference point run the value of the reference point coordinate is assigned to the reference point.

For a linear axis, the value of the reference point coordinate must lie within the working range (including the start / end of the software limit switches).

For a rotary axis, the value of the reference point coordinate must lie within the range from 0 to "End of rotary axis - 1".

Default is "0" pulses.

Reference Point Location for Reference Point Switch

In the "Reference point location for reference point switch" drop-down list, select whether the reference point lies in the plus direction (increasing actual values) or minus direction (decreasing actual values) relative to the reference point switch.

Default is "in plus direction".

Axis monitoring (digital) (S7-300, S7-400)

Working range

For a linear axis, you can specify here whether or not to monitor the working range. If you select the "Working range" button, a check is made to determine whether the actual position value is out of range of the software limit switches. Monitoring is only active on a synchronized axis.

The coordinates of the software limit switches as such belong to the working range.

If monitoring is triggered the run is aborted.

This monitoring function is enabled by default.

Traversing range

You use traversing range monitoring to check whether the value is outside the permitted traversing range of - 5 x 10^8 to + 5 x 10^8. The traversing range is always monitored. The monitoring cannot be disabled (always enabled in the "Monitoring" parameter).

If monitoring is triggered the run is aborted.

Linear axis (S7-300, S7-400)

Description

The **linear axis** is an axis whose physical traversing range is restricted.

The working area on the linear axis is limited by the software limit switches.

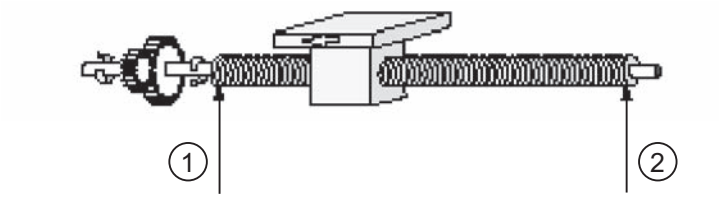

① Physical start
② Physical end

Rotary axis (S7-300, S7-400)

Description

The **rotary axis** is an axis the traversing range of which is not restricted by mechanical stops.

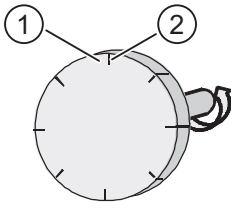

① Largest display value = end of rotary axis - 1

② Start of rotary axis (coordinate 0) = end of rotary axis

A rotary movement of the rotary axis starts at the coordinate "Zero" and ends at the coordinate "End of rotary axis". Physically, the "zero" coordinate is identical with "End of rotary axis" (= 0). The actual position value is toggled at this point. The display of the actual position is always positive.

Encoder (S7-300, S7-400)

Increments per encoder revolution

In the "Increments per encoder revolution" input field, enter the number of increments your encoder should output per 360 degrees. You can find this value in the encoder description.

The CPU evaluates the increments 4 times, in other words, an increment corresponds to four pulses.

Default setting is "1000" increments.

Count direction

In "Count direction", adapt the direction of the position feedback to the axis motion direction.

In the "count direction" drop-down list, select the "Normal" option if rising count pulses correspond to increasing actual position values.

In the "count direction" drop-down list, select the "Inverted" option if rising count pulses correspond to decreasing actual position values.

You should consider all the directions of rotation of the transmission elements, such as clutches and gears.

The "Normal" option is selected by default.

Monitoring

In "Monitoring" you can determine whether the encoder is to be monitored for missing pulses. This is only possible for encoders where the value of "Increments per encoder revolution" is divisible by 10 or 16.

If you select the "Missing pulse" check box, a check is performed to determine whether the encoder always supplies the same number of pulses between two consecutive zero marks.

If you use an encoder without zero mark, you must disable the monitoring for missing pulses.

This monitoring function is disabled by default.

Diagnostics (S7-300, S7-400)

Description

In "Properties > Position > Channel 0" in section "Channel 0 - Diagnostics", select the events for which the diagnostics interrupt is to be generated. No check box is selected by default.

Missing pulse

If you select the "Missing pulse" check box, a diagnostic interrupt is generated as soon as the encoder does not supply the same number of pulses between two consecutive zero marks. You must enable the corresponding monitoring function in the "Channel 0 - Encoder" section.

Traversing range

If you select the "Traversing range" check box, a diagnostic interrupt is generated immediately when the axis exits the traversing range. The corresponding monitoring function is always enabled.

Working range

If you select the "Working range" check box, a diagnostic interrupt is generated if the actual position value is out of range of the software limit switches. You must enable the corresponding monitoring function in the "Channel 0 - Axis" section.

You cannot use this diagnostic interrupt for positioning on a rotary axis. The respective options box is grayed out.

Actual value

If you selected the "Actual value" check box, a diagnostic interrupt is generated if during a traversing motion, the axis does not move by at least one pulse in the specified direction within the monitoring time. The run is cancelled.

The actual value is monitored from the start of travel until the switch-off point is reached.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

The CPU does not recognize the failure of a digital input. Enabling the actual value monitoring allows you to detect missing encoder signals.

This check box is selected by default.

Target approach

If you select the "Target approach" check box, a diagnostic interrupt is generated if the axis does not reach the target range within the monitoring time after it has reached the switch-off point.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

Target range

If you selected the "Target range" check box, a diagnostic interrupt is generated if the drive does not stop on an approached target position after the target range is reached or if the drive drifts away from the approached target position.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

Point-to-point connection function (S7-300, S7-400)

ASCII protocol (S7-300, S7-400)

PtP transmission (ASCII) (S7-300, S7-400)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "ASCII" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "ASCII" there are 4 parameter groups: transmission, end delimiter, data reception and signal assignment.

This section deals with PtP transmission and is divided into 3 sub-sections.

Transmission speed

Select the data transmission rate in bps (baud) from the "Transmission speed" drop-down list. You can select a value between 300 and 38400 baud. In full duplex mode, the maximum transmission speed is 19200 baud.

The default setting is "9600 baud".

Character frame

The data are transmitted via the serial interface in a character frame. Two data formats are available for each character frame.

- Data bits
  In the "Data bits" drop-down list, select the number of bits onto which a character is to be mapped.
  The default setting is "8 bits".
- Stop bits
  In the "Stop bits" drop-down list, select the number of bits appended to every transmitted character. The stop bit identifies the end of a character.
  The default setting is "1 bit".
- Parity
  In the "Parity" drop-down list, select whether the data transmission reliability is to be increased by using a parity bit. A parity of "None" means that no parity bit is transmitted.
  The setting "None" is not possible in a 7-data-bit configuration.
  A sequence of information bits can be extended to include another bit, the parity bit. The addition of its value ("0" or "1") brings the value of all the bits up to a defined state – either even or odd.
  Default selection is "Even".

Data flow control transmission

- Data flow control
  Here you can only specify parameters if you have selected "XON/XOFF" data flow control.
  Default selection is "None".
- XON character
  Enter a code in the "XON character" input field for the XON character. Enter a hex format code here.
  Default selection is "11".
- XOFF character
  Enter a code in the "XOFF character" input field for the XOFF character. Enter a hex format code here.
  Which values you can enter depends on whether you transmit your data with 7 or 8 data bits:
  - 7 data bits: 0 to 7FH
  - 8 data bits: 0 to FFH
    Default selection is "13".
- Wait for XON after XOFF
  In the "Wait for XON after XOFF" input field, specify a time the module has to wait after sending until it receives the XON character from the communication partner.
  You can enter a value in steps of 10 ms, from 20 ms to 65530 ms.
  Default selection is "20000 ms".

PtP end delimiter (ASCII) (S7-300, S7-400)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "ASCII" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "ASCII" there are 4 PtP areas: transmission, end delimiter, data reception and signal assignment.

This section deals with PtP end delimiter and is divided into 2 sub-sections.

Detection of received frame

Select the required option from the "Detection of end of received frame" drop-down list. Three options are available: "After character delay time elapses", "Using end delimiter" and "After receiving a fixed number of characters".

- After character delay time elapses
  Select "After character delay time elapses" from the drop-down list to specify a configured time interval after which the end of a frame is recognized. The character delay time defines the permissible maximum interval between 2 consecutive received characters.
  This option is selected by default.
- Character delay time
  In the "Character delay time" input field you must enter this time interval in milliseconds between 1 and 65,535.
  The shortest character delay that you can select depends on the transmission speed.

Transmission speed	Shortest character delay time
300 bps	130 ms
600 bps	65 ms
1200 bps	32 ms
2400 bps	16 ms
4800 bps	8 ms
9600 bps	4 ms
19200 bps	2 ms
38400 bps	1 ms

The default setting is "4 ms".

- Using end delimiter
  Select "Using end delimiter" from the drop-down list to specify that the end of a message frame is recognized by its end characters.
- Monitoring time for missing end delimiter
  In the "Monitoring time for missing end delimiter" input field you can set the character delay time that can be used as monitoring time for missing end delimiter. This applies to the end delimiter for the following settings:
  - after receiving a fixed number of characters
  - after receiving the end delimiter(s)
    You can also modify the monitoring time here. The same marginal conditions as for the character delay time are applied.
    The calculation when sending and evaluating the block check character must be carried out in the user program itself.
    The default selection is "4 ms".
- End delimiter
  Select the "Using end delimiter" option of the "Detection of end of received frame" parameter to specify the use of one or two defined end delimiters for designating the end of a frame. You can work with one or two end delimiters. Optionally, one or two additional characters can be received after the end delimiter detection. You can use these characters, for example, to include a block check character (BCC) in your transmission.
  Select one of the following options in the corresponding drop-down list:
  - 1st character
  - 1st character with 1 BCC
  - 1st character with 2 BCC
  - 1st AND 2nd character
  - 1st AND 2nd character with 1 BCC
  - 1st AND 2nd character with 2 BCC
    If you have selected this option, the default setting is "1st character".
- Receiver end delimiter
  In the "Receiver 1st end delimiter" input field, and, if required, in the "Receiver 2nd end delimiter" input field, enter a code for the respective end delimiter.
  Which values you can enter depends on whether you transmit your data with 7 or 8 data bits:
  - 7 data bits: 0 to 7FH
  - 8 data bits: 0 to FFH
    The default setting is "3" for the first end delimiter and "0" for the second end delimiter.
- After receiving a fixed number of characters
  Select "After receiving a fixed number of characters" from the drop-down list to specify a fixed number of characters after which the end of a frame is recognized. The length of the received frames is always identical.
  This option is selected by default.

- Pause in transmission between frames (same length as monitoring time)
  If you have selected the "Pause in transmission between frames (same length as monitoring time)" option, a pause equal to the length of the monitoring time (for missing end delimiter) is observed so that the partner detects the receipt of the frame and can synchronize itself accordingly.
  This check box is not selected by default.
- Frame length
  In the "Frame length" input field, you can customize the number of bytes a frame should consist of. You can set a value between 1 and 1024 bytes.
  The default setting is "200 bytes".

Send with end delimiter

You can transmit with an end delimiter after you have specified an end delimiter as the end delimiter of a received frame.

Default selection is "Send up to and including end delimiter".

- Send up to and including end delimiter
  Select the "End criteria > Send up to and including end delimiter" drop-down list to specify whether the end delimiter should be included in the data to be transmitted. Only data up to and including the end-of-message character is transmitted, even if a greater data length is specified in the SFB.
- Send up to length set in the block
  Select the "End criteria > Send up to length set in the block" drop-down list to determine that data are transmitted up to the length declared in the SFB parameters. The last character must be the end delimiter.
- Send with automatically append the end delimiter
  Select the "End criteria > Automatically append the end delimiter" drop-down list to determine that data are transmitted up to the length declared in the SFB parameters. The end delimiter(s) are automatically appended, that is, the end delimiters must not be contained in the data to be transmitted. Depending on the number of end identifiers, either one or two characters more than specified in the SFB (max. 1024 bytes) are transmitted to the partner.
- Sender end delimiter
  If you are using an end delimiter you can now specify its hex code. You can also select whether you want to use a second end delimiter and specify its hex code also. In the "Sender 1st end delimiter" input field, and, if required, in the "Sender 2nd end delimiter" input field, enter a code for the respective end delimiter.
  The default setting is "3" for the first end delimiter and "0" for the second end delimiter.

PtP - data reception (ASCII) (S7-300, S7-400)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "ASCII" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "ASCII" there are 4 PtP areas: transmission, end delimiter, data reception and signal assignment.

This section deals with PtP data reception.

Reception buffering on the CPU

- Clear CPU receive buffer during startup
  Select the "Clear CPU receive buffer during startup" check box to clear the receive buffer of the technological function after power up and after the CPU transition from STOP to RUN.
  This check box is not selected by default.
- Prevent overwriting
  Select the "Prevent overwriting" check box to prevent overwriting data in the receive buffer when the buffer is full.
  This check box is selected by default.
- Use entire buffer
  Select the "Use entire buffer" check box to utilize the complete input buffer of 2048 bytes. When you use the complete buffer, the number of received frames is only dependent on the frame length.
  This check box is selected by default.
- Maximum number of buffered received frames
  In the "Maximum number of buffered received frames" input field, you can enter the number of received frames you want to have buffered in the receive buffer of the technological function. To do this, deactivate the "Use entire buffer" check box. You can select a value between 1 and 10.
  Default setting is "10".

  Note

  If you specify "1" here and deactivate the "Prevent overwrite" parameter **and** cyclically read the received data from the user program, a current message frame will always be sent to the CPU.

PtP - signal assignment (ASCII) (S7-300, S7-400)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "ASCII" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "ASCII" there are 4 PtP areas: transmission, end delimiter, data reception and signal assignment.

This section deals with PtP signal assignment and is divided into 2 sub-sections.

Operating mode

Select one of the three operating modes.

Default is "Full duplex (RS 422) four-wire point-to-point mode".

Full duplex (RS 422) four-wire mode point-to-point

Check the "Full duplex (RS422) four-wire PtP mode" option button to select the suitable operating mode for a PtP connection in four-wire mode. Here, you have the following options to preset the receive line:

- None
- Signal R(A) 5 volts, Signal R(B) 0 volts
- Signal R(A) 0 volts, Signal R(B) 5 volts

Full duplex (RS 422) four-wire multipoint master mode

If the CPU is the master, check the "Full duplex (RS422) four-wire multipoint master mode" option button to select a suitable operating mode for a four-wire connection that is capable of multipoint master operation. Here, you have the following options to preset the receive line:

- Signal R(A) 0 volts, Signal R(B) 5 volts

Half duplex (RS 485) two-wire mode

Check the "Half duplex (RS485) two-wire mode" option button to select a suitable operating mode for a PtP or multipoint connection in two-wire mode. The CPU can be master or slave. Here, you have the following options to preset the receive line:

- None
- Signal R(A) 0 volts, Signal R(B) 5 volts

Receive line initial state

- None
  Check the "None" option button to select a configuration that is significant for bus-compatible special drives only.
- Signal R(A) 5 volts, Signal R(B) 0 volts
  Select the "Signal R(A) 5 volts, Signal R(B) 0 volts" option button to select a configuration for break recognition. "Not available for full duplex (RS422) four-wire multipoint master and half duplex (RS485) two-wire operation)
- Signal R(A) 0 volts, Signal R(B) 5 volts
- Select the "Signal R(A) 0 volts, Signal R(B) 5 volts" option button to select a configuration that represents the idle state (no sending active). Break recognition is not possible with this configuration.
- Default is "Signal R(A) 5 volts, Signal R(B) 0 volts".

3964(R) protocol (S7-300, S7-400)

PtP - transmission (3964(R)) (S7-300, S7-400)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "3964(R)" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "3964(R)" there are 3 PtP areas: transmission, data reception and signal assignment.

This section deals with PtP transmission and is divided into 3 sub-sections.

Transmission speed

Select the data transmission rate in bps (baud) from the "Transmission speed" drop-down list. You can select a value between 300 and 38400 baud.

Default selection is "9600 baud".

Character frame

The data are transmitted via the serial interface in a character frame. Two data formats are available for each character frame.

- Data bits
  In the "Data bits" drop-down list, select the number of bits onto which a character is to be mapped.
  The default setting is "8 bits".
- Stop bits
  In the "Stop bits" drop-down list, select the number of bits appended to every transmitted character. The stop bit identifies the end of a character.
  The default setting is "1 bit".

- Parity
  In the "Parity" drop-down list, select whether the data transmission reliability is to be increased by using a parity bit. A parity of "None" means that no parity bit is transmitted.
  The setting "None" is not possible in a 7-data-bit configuration.
  A sequence of information bits can be extended to include another bit, the parity bit. The addition of its value ("0" or "1") brings the value of all the bits up to a defined state – either even or odd.
  Default selection is "Even".
- Priority
  In the "Priority" drop-down list, select whether the CPU should have high or low priority. A partner has high priority if its send request takes precedence over the send request of the other partner. A partner has low priority if its send request must wait until the send request of the other partner has been dealt with. For the 3964(R) procedure, both communication partners must be assigned different priorities, in other words, one partner is assigned high and the other low priority.
  Default selection is "High".

Frame parameters

- With block check
  Click the "With block check" check box to specify whether to transmit frames with or without block check.
  - Without block check
    The protocol parameters can be assigned by the user.
    If the CPU recognizes the string DLE ETX, it stops receiving and sends a DLE to the communication partner if the block was received without errors or an NAK if the block was received but with errors.
  - With block check
    The protocol parameters are set to default values.
    If the CPU recognizes the string DLE ETX BCC, it stops receiving. It compares the received block check character BCC with the internally generated longitudinal parity. If the BCC is correct and no other receive error has occurred, it returns the character string DLE (the character string NAK is returned to the communication partner if an error has occurred).

  The "With block check" check box is selected by default.
- Use defaults
  Select the "Use defaults" check box if you want to assign the default values to the protocol parameters.
  The module terminates receiving when it detects the character string DLE ETX and returns a DLE (or NAK) string to the communication partner to confirm that it has received an error-free (of faulty) block.
  You can specify the following parameters individually should you decide not to use defaults.
  The "Use defaults" check box is selected by default.
- Character delay time
  In the "Character delay time" input field, you specify the maximum permissible time interval between two received characters within a frame. You can enter a value in 10 ms increments between "20 ms" and "65530 ms".
  The shortest character delay that you can select depends on the transmission speed.

Transmission speed	Shortest character delay time
300 bps	60 ms
600 bps	40 ms
1200 bps	30 ms
2400 bps	20 ms
4800 bps	20 ms
9600 bps	20 ms
19200 bps	20 ms
38400 bps	20 ms

The default setting is "220 ms".

- Acknowledgment delay time
  In the "Acknowledgment delay time" input field, specify the maximum permissible time period until the acknowledgment of the partner when a connection is established (time between STX and DLE acknowledgment of the partner) or closed (time between DLE ETX and DLE acknowledgment of the partner). You can enter a value in 10 ms increments between "20 ms" and "65530 ms".
  The shortest acknowledgment delay time that you can select depends on the transmission speed.

Transmission speed	Shortest acknowledgment delay time
300 bps	60 ms
600 bps	40 ms
1200 bps	30 ms
2400 bps	20 ms
4800 bps	20 ms
9600 bps	20 ms
19200 bps	20 ms
38400 bps	20 ms

Default is "2000 ms".

- Connection retries
  In the "Connection retries" list, specify the maximum number of tries the module has to establish a connection. You can select a value between "1" and "255".
  Default selection is "6".
- Transmission retries
  In the "Transmission retries" list, specify the maximum number of tries the module is allowed to transmit a frame (including the first frame) in case of error. You can select a value between "1" and "255".
  Default selection is "6".

PtP - data reception (3964(R)) (S7-300, S7-400)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "3964(R)" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "3964(R)" there are 3 PtP areas: transmission, data reception and signal assignment.

This topic deals with PtP data reception.

Reception buffer on CPU

- Clear CPU receive buffer during startup
  Select the "Clear CPU receive buffer during startup" check box to clear the receive buffer after power up and after the CPU transition from STOP to RUN.
  This check box is not selected by default.
- Prevent overwriting
  Select the "Prevent overwriting" check box to prevent overwriting data in the receive buffer when the buffer is full.
  This check box is selected by default.
- Use entire buffer
  Select the "Use entire buffer" check box if you want to use the entire 2048 bytes of the CPU receive buffer. When you use the complete buffer, the number of received frames is only dependent on the frame length.
  This check box is selected by default.
- Maximum number of buffered received frames
  In the "Maximum number of buffered received frames" input field, enter the number of received frames to be buffered in the CPU receive buffer. Enter a value between 1 and 10. Default setting is "10".

  Note

  If you specify "1" here and deactivate the "Prevent overwrite" parameter **and** cyclically read the received data from the user program, a current message frame will always be sent to the CPU.

PtP - signal assignment (3964(R)) (S7-300, S7-400)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "3964(R)" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "3964(R)" there are 3 PtP areas: transmission, data reception and signal assignment.

This section deals with PtP signal assignment.

Receive line initial state

With the 3964(R) protocol the X27 (RS 422/485) interface is operated in full duplex (RS 422) four-wire mode. This operating mode is suitable for four-wire PtP connections.

The following options are available as defaults for the receive line:

- None
  Select the "None" option to select a configuration that is only advisable for bus-compatible special drives.
- Signal R(A) 5 V, signal R(B) 0 V (break detection):
  With this option you select a setting that allows break detection.
- Signal R(A) 0 V, signal R(B) 5 V
  With this option you select a setting that corresponds to idle state (no sender active). No break detection is possible with this configuration.

"Signal R(A) 5 V, signal R(B) 0 V (break detection)" is selected as default.

RK512 protocol (S7-300, S7-400)

PtP - transmission (RK512) (S7-300, S7-400)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "RK512" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "RK512" there are 2 PtP areas: transmission and signal assignment.

This section deals with PtP transmission and is divided into 3 sub-sections.

Transmission speed

Select the data transmission rate in bps (baud) from the "Transmission speed" drop-down list. You can select a value between 300 and 38400 baud.

Default selection is "9600 baud".

Character frame

The data are transmitted via the serial interface in a character frame. Two data formats are available for each character frame.

- Data bits
  The "Data bits" drop-down list is grayed out as one character is always mapped onto 8 bits in RK512 transmission.
- Stop bits
  In the "Stop bits" drop-down list, select the number of bits appended to every transmitted character. The stop bit identifies the end of a character.
  The default setting is "1 bit".
- Parity
  In the "Parity" drop-down list, select whether the data transmission reliability is to be increased by using a parity bit. A parity of "None" means that no parity bit is transmitted. A sequence of information bits can be extended to include another bit, the parity bit. The addition of its value ("0" or "1") brings the value of all the bits up to a defined state – either even or odd.
  Default selection is "Even".
- Priority
  In the "Priority" drop-down list, select whether the CPU should have high or low priority. A partner has high priority if its send request takes precedence over the send request of the other partner. A partner has low priority if its send request must wait until the send request of the other partner has been dealt with. With RK512, both communication partners must be assigned different priorities, that is, one partner is assigned high and the other is assigned low priority.
  Default selection is "High".

Frame parameters

- With block check
  Click the "With block check" check box to specify whether to transmit frames with or without block check.
  - Without block check
    The protocol parameters can be assigned by the user.
    If the CPU recognizes the string DLE ETX, it stops receiving and sends a DLE to the communication partner if the block was received without errors or an NAK if the block was received but with errors.
  - With block check
    The protocol parameters are set to default values.
    If the CPU recognizes the string DLE ETX BCC, it stops receiving. It compares the received block check character BCC with the internally generated longitudinal parity. If the BCC is correct and no other receive error has occurred, it returns the character string DLE (the character string NAK is returned to the communication partner if an error has occurred).

  The "With block check" check box is selected by default.
- Use defaults
  Select the "Use defaults" check box if you want to assign default values to the protocol parameters.
  If the CPU recognizes the string DLE ETX, it stops receiving and sends a DLE to the communication partner if the block was received without errors or an NAK if the block was received but with errors.
  You can specify the following parameters individually should you decide not to use defaults.
  The "Use defaults" check box is selected by default.
- Character delay time
  In the "Character delay time" input field, you specify the maximum permissible time interval between two received characters within a frame. You can enter a value in 10 ms increments between "20 ms" and "65530 ms".
  The shortest character delay that you can select depends on the transmission speed.

Transmission speed	Shortest character delay time
300 bps	60 ms
600 bps	40 ms
1200 bps	30 ms
2400 bps	20 ms
4800 bps	20 ms
9600 bps	20 ms
19200 bps	20 ms
38400 bps	20 ms

The default setting is "220 ms".

- Acknowledgment delay time
  In the "Acknowledgment delay time" input field, specify the maximum permissible time period until the acknowledgment of the partner when a connection is established (time between STX and DLE acknowledgment of the partner) or closed (time between DLE ETX and DLE acknowledgment of the partner). You can enter a value in 10 ms increments between "20 ms" and "65530 ms".
  The shortest acknowledgment delay time that you can select depends on the transmission speed.

Transmission speed	Shortest acknowledgment delay time
300 bps	60 ms
600 bps	40 ms
1200 bps	30 ms
2400 bps	20 ms
4800 bps	20 ms
9600 bps	20 ms
19200 bps	20 ms
38400 bps	20 ms

Default selection is "2000 ms".

- Connection retries
  In the "Connection retries" list, specify the maximum number of tries the module has to establish a connection. You can select a value between 1 and 255.
  Default selection is "6".
- Transmission retries
  In the "Transmission retries" list, specify the maximum number of attempts the module is allowed to transmit a frame. You can select a value between "1" and "255".
  Default selection is "6".

PtP - signal assignment (RK512) (S7-300, S7-400)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "RK512" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

There are 2 PtP areas for "RK512": transmission and signal assignment.

This section deals with PtP transmission and is divided into 3 sub-sections.

Receive line initial state

The X27 (RS 422/485) interface is operated in full duplex (RS 422) four-wire operating mode. This operating mode is suitable for four-wire PtP connections.

The following options are available as defaults for the receive line:

- None
  Select the "None" option to select a configuration that is only advisable for bus-compatible special drives.
- Signal R(A) 5 V, signal R(B) 0 V (break detection):
  With this option you select a setting that allows break detection.
- Signal R(A) 0 V, signal R(B) 5 V
  With this option you select a setting that corresponds to idle state (no sender active). No break detection is possible with this configuration.
- "Signal R(A) 5 V, signal R(B) 0 V (break detection)" is selected as default.

CPU information (S7-300, S7-400)

CPU registers (S7-300, S7-400)

Overview

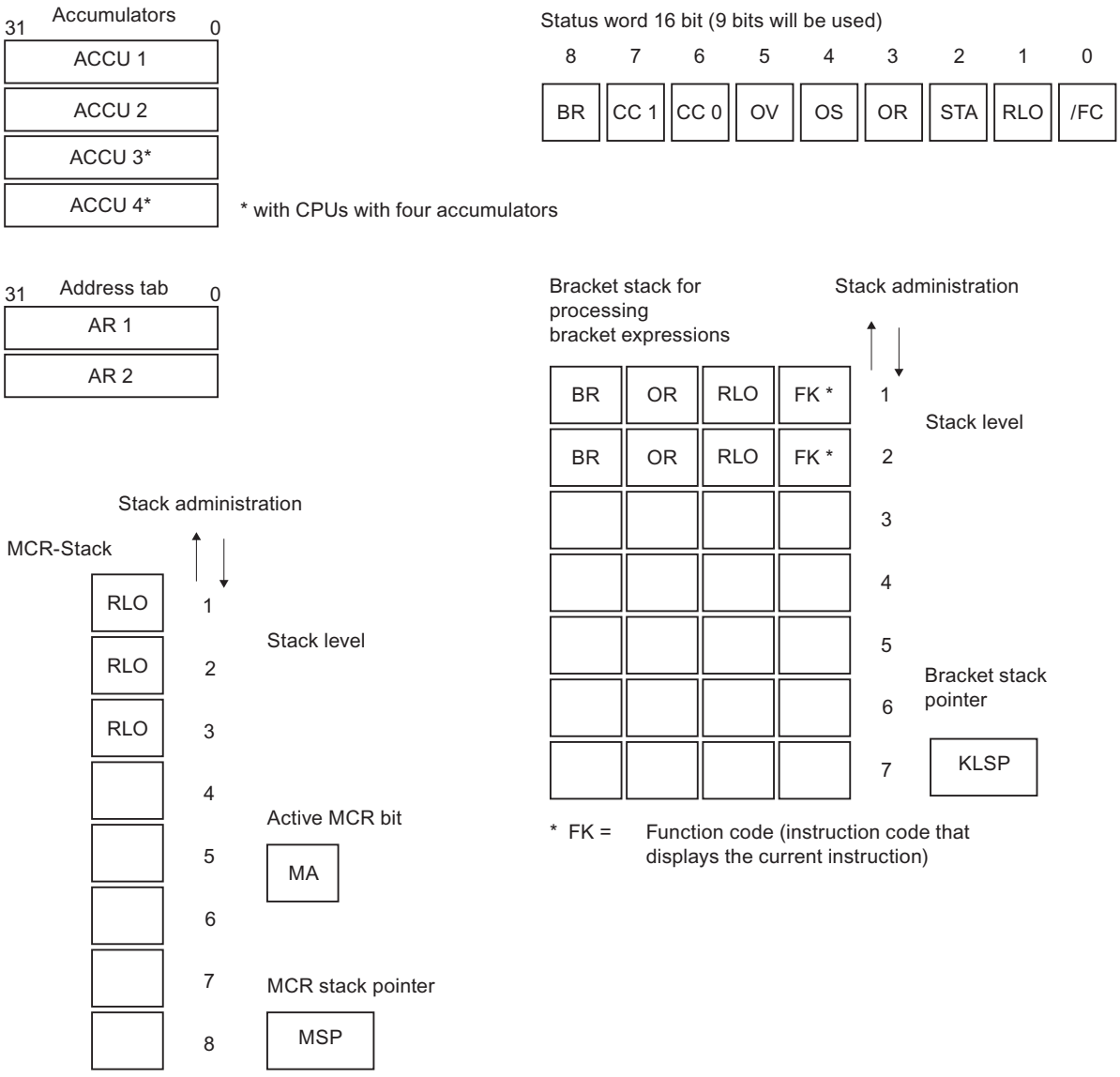

Accumulators (S7-300, S7-400)

Description

The 32-bit accumulators are general-purpose registers for processing bytes, words, and double words. For this purpose, the operands are loaded into the ACCUs and linked there. The result of the operation is always located in ACCU 1 from where it can be transferred to a memory location.

The stack mechanism for managing the accumulators works as follows:

- A load operation only affects ACCU 1 and stores the old content in ACCU 2.
- A transfer operation (copy operation) does not change the contents of the accumulators.
- The **TAK** operation swaps the contents of ACCU 1 and ACCU 2.
- The result of logic operations between ACCU 1 and ACCU 2 (math operations, comparison operations, AND, OR, etc.) is always stored in ACCU 1.

Definition of bytes and words in a 32-bit register (based on the example of ACCU 1)

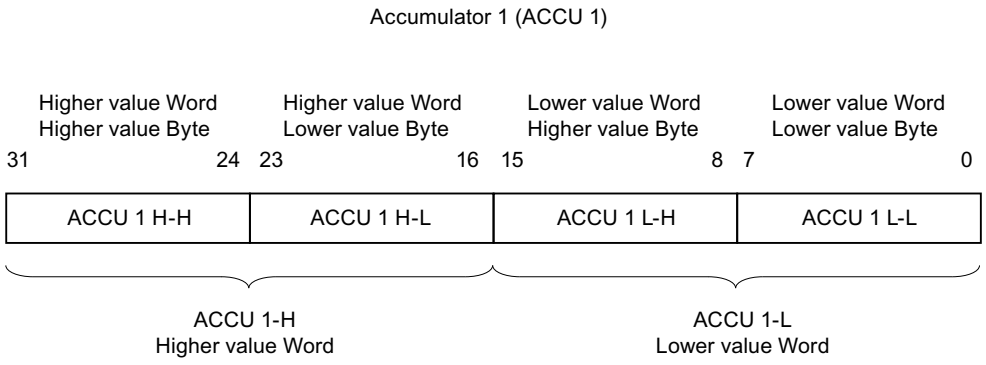

Addressing (S7-300, S7-400)

Process images (input and output memory area) (S7-300, S7-400)

Memory area of the process images

The OB-process image (OB1 PI) is cyclically updated by the operating system.

This process image is saved by the I/O hardware during each cycle of the instruction list. This means that the bits of the memory area I in the process image do not change during the access to the instructions. Even though the bits of the memory area Q in the process image can have different states during the program cycle, only the last signal state is sent to the output module.

See also: User data (Page 737) and Updating the process images (Page 731)

Exceptions

The following cases do not use the process image to access the I/O:

- Direct I/O addressing of the inputs and outputs peripheral memory area (P area).
- Using functions that process the I/O data immediately.

Example of a normal I/O access via the process image

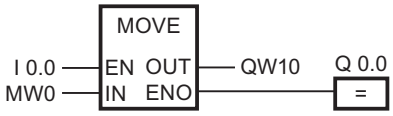

Update QW10 in the peripheral output with the value from MW0.

See also

Access to the I/O addresses (Page 739)

Direct access to the I/O (PI and PQ memory areas) (S7-300, S7-400)

Direct access to the I/O

If your project needs to update the data of the I/O more quickly than the process image makes this available, you can access the signal modules directly in the peripheral inputs and outputs (PI, PQ). The current status of the input modules can be read from the PI memory area area. Output modules can be set by writing to the PQ memory area.

CAUTION

Single bits (byte. bit) cannot be addressed in the I/O area area. To set single bits, you address bytes, words or double words in logic operations. When you write an updated bit (within a byte, word, or double word) to the PQ area, all bits at the address of this I/O area will be updated.

Refer to the **Process image** entry, for more information on the connection between the I/O memory area (PI, PQ memory area) and the process image (I and Q memory area). The entry also explains how the process image saves (stabilizes) what has been written from the signal modules or sent to them.

Example of direct access to the I/O via the I/O memory area

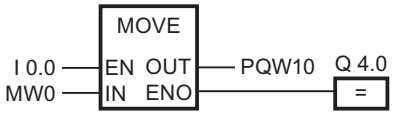

Update PQW10 in the peripheral output with the value from MW0.

See also

Access to the I/O addresses (Page 739)
Address areas (Page 739)

Full syntax for entering constants (S7-300, S7-400)

Overview

Short form	Long form
B#	BYTE#
W#	WORD#
DW#	DWORD#
D#	DATE#
T#	TIME#
TOD#	TIME_OF_DAY#
S5T#	S5TIME#
DT#	DATE_AND_TIME#
P#	POINTER#

Summary of bit, byte, word, and double word identifiers (S7-300, S7-400)

Overview

CPU memory area	Operand ID			
	Bit	Byte	Word	Double word
Process image input	I	IB	IW	ID
Process image output	Q	QB	QW	QD
Bit memory	M	MB	MW	MD
Local data	L	LB	LW	LD
Direct peripheral input area		PIB	PIW	PID
Direct peripheral output area		PQB	PQW	PQD
Data block [OPN] DB [OPN] DI	 DBX DIX	 DBB DIB	 DBW DIW	 DBD DID
Timer (T)	T		T	
Counter (C)	C		C	

Note

The two operands **IB0** and **IB1** (memory area I, format: byte, address bytes 0, 1) address the same two bytes as the operand **IW0** (memory area I, format: word, 2 bytes, incrementing of byte 0 to address bytes 0, 1).

Identifiers of elements and blocks (S7-300, S7-400)

Overview

T	Timer
DB	Global data block
DI	Instance data block
FC	Function
FB	Function block
OB	Organization block
C (or Z)	Counter (depending on the mnemonics)
SDB	System data block
SFC	System function
SFB	System function block

Organization blocks (S7-300, S7-400)

English designations of the OB types (S7-300, S7-400)

English designation of the OB types

When an OB is created in the "Add new block" dialog, the OB types can only be displayed in English for technical reasons – irrespective of the set user interface language. In order to facilitate handling, the following table lists the English terms and their corresponding designations in the set user interface language. The table applies to the superset of S7-300 and S7-400 CPUs.

English designation of the OB type in the "Add new block" dialog.			Designation in the information system
Time interrupts			Time interrupts
	Time of day		Time-of-day interrupts
		TOD_INT0 [OB10] bis TOD_INT7 [OB17]	Time-of-day interrupt OBs

English designation of the OB type in the "Add new block" dialog.			Designation in the information system
	Time delay		Time-delay interrupts
		DEL_INT0 [OB20] bis DEL_INT3 [OB23]	Time-delay interrupt OBs
	Cyclic		Cyclic interrupts
		CYC_INT0 [OB30] bis CYC_INT8 [OB38]	Cyclic interrupt OBs
Hardware Interrupts			Hardware interrupts
		HW_INT0 [OB40] bis HW_INT7 [OB47]	Hardware interrupt OB
Startup			Startup
		BACKGROUND [OB90]	Background OB
		COMPLETE RESTART [OB100]	Warm restart
		RESTART [OB101]	Hot restart
		COLD RESTART [OB102]	Cold restart
Alarming			
		DP: STATUS ALARM [OB55]	Status interrupt OB
		DP: UPDATE ALARM [OB56]	Update interrupt OB
		DP: MANUFACTURE ALARM [OB57]	Vendor-specific interrupt OB
		MULTI_INT [OB 60]	Multicomputing interrupt OB
		SYNC_1 [OB61] bis SYNC_4 [OB64]	Isochronous mode interrupt OBs
Fault Interrupts			Fault
		CYCL_FLT [OB 80]	Time error OB
		PS_FLT [OB81]	Power supply error OB
		I/O_FLT1 [OB82]	Diagnostic interrupt OB
		I/O_FLT2 [OB83]	Pull/plug interrupt OB
		CPU_FLT [OB84]	CPU hardware fault OB
		OBNL_FLT [OB85]	Program execution error OB
		RACK_FLT [OB86]	Rack failure OB
		COMM_FLT [OB87]	Communications error OB
		BREAKUP ERROR [OB88]	Processing abort OB
		PROG_ERR [OB121]	Programming error OB
		MOD_ERR [OB122]	I/O access error OB

Overview of organization blocks (S7-300, S7-400)

Organization blocks

Organization blocks (OBs) are the interface between the CPU operating system and the user program. OBs are used to execute specific program sections:

- At CPU startup
- In cyclic or clocked execution

- Whenever errors occur
- Whenever hardware interrupts occur.

Organization blocks are executed according to the priority they have been allocated.

Available OBs

Not all CPUs can process all OBs available in S7. See the operation list of your CPU to determine which OBs are available to you.

Start events and default priority class

The following table contains the start event belonging to each OB as well as the default priority class.

OB	Start event	Default priority class	Explanation
OB 1	End of startup or end of OB 1	1	Free cycle
OB 10	Time-of-day interrupt 0	2	No default time specified
OB 11	Time-of-day interrupt 1	2	
OB 12	Time-of-day interrupt 2	2	
OB 13	Time-of-day interrupt 3	2	
OB 14	Time-of-day interrupt 4	2	
OB 15	Time-of-day interrupt 5	2	
OB 16	Time-of-day interrupt 6	2	
OB 17	Time-of-day interrupt 7	2	
OB 20	Time-delay interrupt 0	3	No default time specified
OB 21	Time-delay interrupt 1	4	
OB 22	Time-delay interrupt 2	5	
OB 23	Time-delay interrupt 3	6	
OB 30	Cyclic interrupt 0 (default: 5 s sampling time)	7	Cyclic interrupts
OB 31	Cyclic interrupt 1 (default: 2 s sampling time)	8	
OB 32	Cyclic interrupt 2 (default: 1 s sampling time)	9	
OB 33	Cyclic interrupt 3 (default: 500 ms sampling time)	10	
OB 34	Cyclic interrupt 4 (default: 200 ms sampling time)	11	
OB 35	Cyclic interrupt 5 (default: 100 ms sampling time)	12	
OB 36	Cyclic interrupt 6 (default: 50 ms sampling time)	13	
OB 37	Cyclic interrupt 7 (default: 20 ms sampling time)	14	
OB 38	Cyclic interrupt 8 (default: 10 ms sampling time)	15	

OB	Start event	Default priority class	Explanation
OB 40	Hardware interrupt 0	16	Hardware interrupts
OB 41	Hardware interrupt 1	17	
OB 42	Hardware interrupt 2	18	
OB 43	Hardware interrupt 3	19	
OB 44	Hardware interrupt 4	20	
OB 45	Hardware interrupt 5	21	
OB 46	Hardware interrupt 6	22	
OB 47	Hardware interrupt 7	23	
OB 55	Status interrupt	2	DPV1 interrupts
OB 56	Update interrupt	2	
OB 57	Manufacturer-specific interrupt	2	
OB 61	Isochronous mode interrupt 1	25	Isochronous mode interrupts
OB 62	Isochronous mode interrupt 2	25	
OB 63	Isochronous mode interrupt 3	25	
OB 64	Isochronous mode interrupt 4	25	
OB 80	Time error	26, 28 [1]	Asynchronous error interrupts
OB 81	Power supply error	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 82	Diagnostic interrupt	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 83	Pull/plug interrupt	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 84	CPU hardware fault	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 85	Priority class error	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 86	Failure of an expansion device, of a DP master system or a device for distributed I/O	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 87	Communication error	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 88	Processing interrupt	28	
OB 90	Warm restart or cold restart or deleting of a block being executed in OB 90, or loading of an OB 90 into the CPU or end of OB 90	29 [2)]	Background cycle
OB 100	Warm restart	27 [1]	Startup
OB 101	Hot restart	27 [1]	
OB 102	Cold restart	27 [1]	

OB	Start event	Default priority class	Explanation
OB 121	Programming error	Priority class of the OB that has caused the error	Synchronous error interrupts
OB 122	I/O access error	Priority class of the OB that has caused the error	
[1]Priority classes 27 and 28 are valid in the priority class model of the startup. [2]Priority class 29 corresponds to priority 0.29. This means that the background cycle has lower priority than does the free cycle.			

Cyclic program (OB 1) (S7-300, S7-400)

Description

The operating system of the S7 CPU executes OB 1 cyclically. When OB 1 has been executed, the operating system starts executing OB 1 again. Cyclic execution of OB 1 is started after the startup has been completed. You can call instructions in OB 1.

Function of OB 1

OB 1 has the lowest priority of all OBs that are monitored for runtime. With the exception of OB 90 all other OBs can interrupt execution of OB 1. The following events cause the operating system to call OB 1:

- End of startup processing
- The execution of OB 1 (the previous cycle) has finished.

When OB 1 has been executed, the operating system sends global data. Before restarting OB 1, the operating system writes the process image output to the output modules, updates the process image input, and receives any global data for the CPU.

S7 monitors the maximum cycle time, ensuring a maximum response time. The value for the maximum cycle time is preset to 150 ms. You can set a new value or you can restart the time monitoring anywhere within your program with the "RE_TRIGR" instruction. If your program exceeds the maximum cycle time for OB 1, then the operating system calls OB 80 (time error OB). If OB 80 has not been programmed, the CPU changes to STOP mode.

Apart from monitoring the maximum cycle time, it is also possible to guarantee a minimum cycle time. The operating system will delay the start of a new cycle (writing of the process image output to the output modules) until the minimum cycle time has been reached.

The configuration is used to change the maximum and minimum cycle time parameters.

Local data for OB 1

The following table describes the temporary (TEMP) tags for OB 1. The tag names are the default names of OB 1.

Tag	Data type	Description
OB1_EV_CLASS	BYTE	Event class and identifiers: B#16#11: OB 1 is active
OB1_SCAN_1	BYTE	• B#16#01: Completion of the warm restart • B#16#02: Completion of the hot restart • B#16#03: Completion of the free cycle • B#16#04: Completion of the cold restart • B#16#05: First OB 1 cycle of the new master CPU after master-reserve switchover and STOP of the previous master
OB1_PRIORITY	BYTE	Priority class: 1
OB1_OB_NUMBR	BYTE	OB number (01)
OB1_RESERVED_1	BYTE	Reserved
OB1_RESERVED_2	BYTE	Reserved
OB1_PREV_CYCLE	INT	Runtime of last scan cycle (ms).
OB1_MIN_CYCLE	INT	Minimum cycle time (ms) since the last startup
OB1_MAX_CYCLE	INT	Maximum cycle time (ms) since the last startup
OB1_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Time-of-day interrupt organization blocks (OB 10 to OB 17) (S7-300, S7-400)

Description

S7 provides up to eight OBs (OB 10 to OB 17) which can be started once or periodically. You can assign parameters for your CPU in such a manner that these OBs are processed in the following intervals:

- Once
- Every minute
- Hourly
- Daily
- Weekly
- Monthly
- Yearly
- End of month

Note

For monthly execution of a time-of-day interrupt OB, only the days 1, 2, to 28 can be used as a starting date.

Function of time-of-day interrupt OBs

To start a time-of-day interrupt, you must first set and then activate the interrupt. The three following start possibilities exist:

- Automatic start of the time-of-day interrupt. This occurs if you have both set and activated the time-of-day interrupt per configuration. The following table shows the principle possibilities when activating a time-of-day interrupt per configuration.
- You set the time-of-day interrupt per configuration and then activate it by calling the "ACT_TINT" instruction in your program.
- You set the time-of-day interrupt by calling the "SET_TINT" instruction and then activate it by the "ACT_TINT" instruction.

Interval	Description
Not activated	The time-of-day interrupt OB is not executed, even when it is loaded in the CPU. You can activate the time-of-day interrupt by calling the "ACT_TINT" instruction.
Activated once only	The time-of-day interrupt OB is canceled automatically after it runs once as specified. Your program can use the "SET_TINT" instruction to reset the time-of-day interrupt and the "ACT_TINT" instruction to reactivate it.
Activated periodically	When the time-of-day interrupt occurs, the CPU calculates the next start time for the time-of-day interrupt based on the current time of day and the period.

Note

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed once, the date and time must not be in the past (relative to the real-time clock of the CPU).

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed periodically, but the start date and time are in the past, then the time-of-day interrupt will be processed the next time it is due. This is illustrated in the following figure which shows the initial processing of a time-of-day interrupt OB if the start time is in the past, and if periodic activation has been set.

You can use the "DIS_IRT" instruction to disable the call of time-of-day interrupt OBs, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

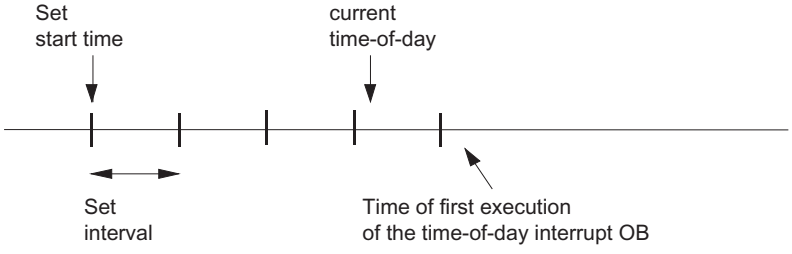

Conditions that effect the time-of-day interrupt OBs

As a time-of-day interrupt occurs only at specific intervals, certain conditions can affect the function of the associated OBs during execution of your program. The following table shows some of these conditions and describes how these affect the processing of the time-of-day interrupt OB.

Condition	Result
Your program calls the "CAN_TINT" instruction and cancels a time-of-day interrupt.	The operating system deletes the start event (date and time) of the time-of-day interrupt. You have to reset and activate the start event if the OB is to be called again.
Your program tried to activate a time-of-day interrupt OB that was not loaded in the CPU at the time of activation.	The operating system then calls OB 85. If OB 85 has not been programmed (loaded to the CPU), then the CPU switches to STOP mode.
By synchronizing or correcting the system clock of the CPU you preset the time, and the start event, the date or the time for a time-of-day interrupt OB is skipped.	The operating system calls OB 80 and codes the number of the time-of-day interrupt OB and the start event information in OB 80. The operating system then processes the time-of-day interrupt OB once, regardless of how often this OB actually should have been processed. The start event information of OB 80 shows the date and the time at which the time-of-day interrupt OB was initially skipped.
By synchronizing or correcting the system clock of the CPU you reset the time, and the start event, the date or the time for a time-of-day interrupt OB is repeated.	S7-400-CPUs: If the time-of-day interrupt OB was already activated before the time was reset, it is not called again for the times that have already passed. S7-300-CPUs: The time-of-day interrupt OB is carried out.
The CPU carries out a warm restart or a cold restart.	Each time-of-day interrupt OB that was configured by an instruction then adopts the configuration again that you have specified. If you have configured a time-of-day interrupt in such a way that the associated OB is started once and have set and activated it by means of configuration, then the OB will be called once after a warm restart or cold restart of the operating system as long as the configured starting time lies in the past (referring to the real time clock of the CPU).
A time-of-day interrupt OB is still being processed while the start event for the next interval is already occurring.	The operating system then calls OB 80. If OB 80 has not been programmed, the CPU changes to STOP mode. Otherwise, the requested OB execution takes place after processing of OB 80 and the time-of-day interrupt OB is complete.

Local data of the time-of-day interrupt OB

The following table includes the temporary (TEMP) tags of a time-of-day interrupt OB. The default names of OB 10 are selected as tag names.

Tag	Data type	Description
OB10_EV_CLASS	BYTE	Event class and identifiers: B#16#11: Interrupt is active
OB10_STRT_INFO	BYTE	B#16#11: Start request for OB 10 (B#16#12: Start request for OB 11) : : (B#16#18: Start request for OB 17)
OB10_PRIORITY	BYTE	Assigned priority class; default value: 2
OB10_OB_NUMBR	BYTE	OB number (10 to 17)
OB10_RESERVED_1	BYTE	Reserved
OB10_RESERVED_2	BYTE	Reserved
OB10_PERIOD_EXE	WORD	The OB is processed at the specified interval: W#16#0000: Once W#16#0201: Every minute W#16#0401: Hourly W#16#1001: Daily W#16#1201: Weekly W#16#1401: Monthly W#16#1801: Yearly W#16#2001: End of month
OB10_RESERVED_3	INT	Reserved
OB10_RESERVED_4	INT	Reserved
OB10_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Time-delay interrupt organization blocks (OB 20 to OB 23) (S7-300, S7-400)

Description

S7 provides up to four OBs (OB 20 to OB 23) which are executed after a specified delay. Every time-delay interrupt OB is started by calling the "SRT_DINT" instruction. The delay time is an input parameter of the instruction.

When your program calls the "SRT_DINT" instruction, you provide the OB number, the delay time, and a user-specific identifier. After the specified delay, the associated OB starts. You can also cancel execution of a time-delay interrupt that has not yet started.

Function of time-delay interrupt OBs

After the delay time has expired, whose value in milliseconds you have provided to the "SRT_DINT" instruction together with an OB number, the operating system will start the corresponding OB.

To use the time-delay interrupts, you must perform the following tasks:

- You must call the "SRT_DINT" instruction.
- You must download the time-delay interrupt OB to the CPU as part of your program.

Time-delay interrupt OBs are executed only when the CPU is in RUN mode. A warm restart or a cold restart clears any start events for a time-delay interrupt OB. If a time-delay interrupt has not yet been activated, you can use the "CAN_DINT" instruction to cancel its execution.

The delay time is measured with a precision of 1 ms. A delay time can be immediately started after your sequence. You can query the status of a time-delay interrupt using the "QRY_DINT" instruction.

The operating system calls an asynchronous error OB if one of the following events occur:

- If the operating system attempts to start an OB that is not loaded and you have specified its number when calling the "SRT_DINT" instruction.
- If the next start event for a time-delay interrupt occurs before the corresponding time-delay interrupt OB has been completely executed.

You can use the "DIS_IRT" instruction to disable the call of time-delay interrupt OBs, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Local data for time-delay interrupt OBs

The following table shows the temporary (TEMP) tags of a time-delay interrupt OB. The tag names are the default names of OB 20.

Tag	Data type	Declaration	Description
OB20_EV_CLASS	BYTE	TEMP	Event class and identifiers: B#16#11: Interrupt is active
OB20_STRT_INF	BYTE	TEMP	• B#16#21: Start request for OB 20 • (B#16#22: Start request for OB 21) • (B#16#23: Start request for OB 22) • (B#16#24: Start request for OB 23)
OB20_PRIORITY	BYTE	TEMP	Assigned priority class: Default values 3 (OB 20) to 6 (OB 23) Default value for S7-1500 CPUs: 3
OB20_OB_NUMBR	BYTE	TEMP	OB number (20 to 23)
OB20_RESERVED_1	BYTE	TEMP	Reserved
OB20_RESERVED_2	BYTE	TEMP	Reserved
OB20_SIGN	WORD	TEMP	User ID: Input parameter SIGN from the call of the "SRT_DINT" instruction
OB20_DTIME	TIME	TEMP	Assigned delay time in ms
OB20_DATE_TIME	DATE_AND_TIME	TEMP	Date and time when the OB was called

Cyclic interrupt organization blocks (OB 30 to OB 38) (S7-300, S7-400)

Description

S7 provides up to nine cyclic interrupt OBs (OB 30 to OB 38). You can use it to start programs after equidistant time phases. The following table shows the default values for the time frame and priority classes for the cyclic interrupt OBs.

Cyclic interrupt OB	Default value for time frame	Default value for the priority class
OB 30	5 s	7
OB 31	2 s	8
OB 32	1 s	9
OB 33	500 ms	10
OB 34	200 ms	11
OB 35	100 ms	12
OB 36	50 ms	13
OB 37	20 ms	14
OB 38	10 ms	15

Function of the cyclic interrupt OBs

The equidistant start times of the cyclic interrupt OBs are determined by the cycle clock and the phase offset.

Note

You must make sure that the run time of each cyclic interrupt OB is significantly shorter than its interval. If a cyclic interrupt OB has not been completely executed before it is due for execution again because the interval has expired, the time error OB (OB 80) is started. The cyclic interrupt that caused the error is executed later.

You can use the "DIS_IRT" instruction to disable the call of the cyclic interrupt OBs, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Refer to the specifications of your specific CPU for the range of the parameters cycle clock, priority class, and phase offset. You can change parameter settings through configuration.

Local data for cyclic interrupt OBs

The following table shows the temporary (TEMP) tags of a cyclic interrupt OB.

Tag	Data type	Description
OB3x_EV_CLASS	BYTE	Event class and identifiers: B#16#11: Interrupt is active
OB3x_STRT_INF	BYTE	• B#16#30: Special start request for a cyclic interrupt OB in the H-system (assigned special handling on changeover to the "Redundant" system state) • B#16#31: Start request for OB 30 … • B#16#36: Start request for OB 35 … • B#16#39: Start request for OB 38 • B#16#3A: Start request for cyclic interrupt OBs (OB 30 to OB 38) with cyclic interrupt cycle less than one millisecond
OB3x_PRIORITY	BYTE	Assigned priority class; default values: 7 (OB 30) to 15 (OB 38) Default values for S7-1500 CPUs: 8 to 14; 16 to 17
OB3x_OB_NUMBR	BYTE	OB number (30 to 38)
OB3x_RESERVED_1	BYTE	Reserved
OB3x_RESERVED_2	BYTE	Reserved
OB3x_PHS_OFFSET	INT	for OBs 30 to 34 and 36 to 38: • If OB3x_STRT_INF=B#16#3A: phase offset in µs • In all other cases: phase offset in ms
OB35_PHASE_OFFSET	WORD	for OB 35: • If OB35_STRT_INF=B#16#3A: phase offset in µs • In all other cases: phase offset in ms
OB3x_RESERVED_3	INT	Reserved
OB3x_EXC_FREQ	INT	• If OB3x_STRT_INF=B#16#3A: Cycle clock in µs • In all other cases: Cycle clock in milliseconds
OB3x_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called.

Hardware interrupt organization blocks (OB 40 to OB 47) (S7-300, S7-400)

Description

S7 provides up to eight independent hardware interrupts each with its own OB.

Per configuration you specify which channels will trip a hardware interrupt,

- under which supplementary condition for each signal module.
- Which hardware interrupt OB is assigned to the individual groups of channels (as default, all hardware interrupts are processed by OB 40).

With CPs and FMs, you must use the appropriate software for the module for this.

You select the priority classes for the individual hardware interrupt OBs per configuration.

Function of the hardware interrupt OBs

After the module triggers a hardware interrupt the operating system identifies the slot and determines the associated hardware interrupt OB. If this has a higher priority than the priority class active at the moment then it will be started. The channel-specific acknowledgment is sent after this hardware interrupt OB has been executed.

If another event that triggers a hardware interrupt occurs on the same module during the time between identification and acknowledgment of a hardware interrupt, the following applies:

- If the event occurs on the channel that previously triggered the hardware interrupt, then the associated interrupt is lost. The following figure illustrates the connection between a process signal and the execution of the associated hardware interrupt OB based on the example of a channel of a digital input module. The triggering event is the rising edge. The associated hardware interrupt OB is OB 40.

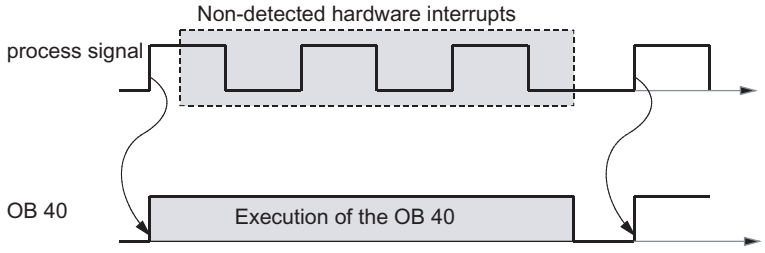

- If the event occurs on a different channel of the same module, then no hardware interrupt can be triggered at that moment. However, it is not lost, it is then triggered after the currently active hardware interrupt has been acknowledged.

If a hardware interrupt is triggered the OB of which is currently active on account of a hardware interrupt of another module, then the new request is registered and the OB is worked off at the specified time.

You can use the "DIS_IRT" instruction to disable the call of the hardware interrupt OBs, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

You can use the "WR_PARM", " WR_DPARM", and "PARM_MOD" instructions to assign the hardware interrupt parameters for a module.

Local data of the hardware interrupt OBs

The following table shows the temporary (TEMP) tags of a hardware interrupt OB. The default names of OB 40 have been selected as tag names.

Tag	Data type	Description
OB40_EV_CLASS	BYTE	Event class and identifiers: B#16#11: Interrupt is active
OB40_STRT_INF	BYTE	• B#16#41: Interrupt via interrupt line 1 • B#16#42: Interrupt via interrupt line 2 (for S7-400 only) • B#16#43: Interrupt via interrupt line 3 (for S7-400 only) • B#16#44: Interrupt via interrupt line 4 (for S7-400 only) • B#16#45: WinAC: Interrupt triggered by PC Note: The interrupt lines 1, ... 4 are assigned in the multicomputing mode of the CPUs 1, ... 4.
OB40_PRIORITY	BYTE	Assigned priority class: Default values: 16 (OB 40) to 23 (OB 47) Default value for S7-1500 CPUs: 16
OB40_OB_NUMBR	BYTE	OB number (40 to 47)
OB40_RESERVED_1	BYTE	Reserved
OB40_IO_FLAG	BYTE	• B#16#54: Input module • B#16#55: Output module • B#16#00: Not available
OB40_MDL_ADDR	WORD	Logical start address of the module that has triggered the interrupt
OB40_POINT_ADDR	DWORD	• In the case of digital modules: Bit array containing the inputs on the module that have triggered the hardware interrupt Refer to the respective module description for information on the assignment of bits of OB40_POINT_ADDR to the module channels. • In the case of analog modules:bit field with the information as to which channel has exceeded which thresholds (refer to the module description for the precise structure). • In the case of CPs or IMs: Interrupt status of the module (not user-relevant)
OB40_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Note

If you are using a DPV1-capable CPU, you can use the "RALRM" instruction to obtain additional information about the interrupt that goes beyond the start information of the OB. This also applies if the DP master is operated in S7 compatible mode.

Status interrupt OB (OB 55) (S7-300, S7-400)

Description

The CPU operating system calls OB 55 if a status interrupt was triggered by the slot of a DPV1 slave or of an IO device. This might be the case if a component (module or rack) of a DPV1 slave or of an IO device changes its operating mode, for example from RUN to STOP. For precise information on events that trigger a status interrupt, refer to the documentation provided by the manufacturer of the DPV1 slave or the IO device.

Note

If S7-400 CPUs or S7-400 PROFIBUS CPs are set to the "S7 compatible" DP mode, no status interrupt OB (OB 55) can be used.

Local data of the status interrupt OB

The following table shows the temporary (TEMP) tags of the status interrupt OB. The tag names are the default names of OB 55.

Tag	Data type	Description
OB55_EV_CLASS	BYTE	Event class and identifiers: B#16#11 (incoming event)
OB55_STRT_INF	BYTE	• B#16#55: Status interrupt for DP • B#16#58: Status interrupt for PROFINET IO
OB55_PRIORITY	BYTE	Assigned priority class, default value: 2 or 4 (for S7-1500 CPUs)
OB55_OB_NUMBR	BYTE	OB number (55)
OB55_RESERVED_1	BYTE	Reserved
OB55_IO_FLAG	BYTE	Input module: B#16#54 Output module: B#16#55
OB55_MDL_ADDR	WORD	Logical start address of the interrupt triggering component (module)
OB55_LEN	BYTE	Length of the data block supplied by the interrupt
OB55_TYPE	BYTE	ID for interrupt type "status interrupt"
OB55_SLOT	BYTE	Slot number of the component triggering the interrupt (module)

Tag	Data type	Description
OB55_SPEC	BYTE	Specifier: • Bit 0 to 1: Interrupt specifier • Bit 2: Add_Ack • Bit 3 to 7: Seq. no.
OB55_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Note

The meaning of OB55_LEN, OB55_TYPE, OB55_SLOT, OB55_SPEC indicated in the table above applies only to a status interrupt for DP. If the status interrupt is for PROFINET IO, you must organize the local tags as indicated in the next table.

Note

You can obtain complete supplemental information on the interrupt from the telegram by calling the "RALRM" instruction within OB 55.

If you want to program OB55 dependent on start events, we recommend that you organize the local tags as follows:

Tag	Data type
OB55_EV_CLASS	BYTE
OB55_STRT_INF	BYTE
OB55_PRIORITY	BYTE
OB55_OB_NUMBR	BYTE
OB55_RESERVED_1	BYTE
OB55_IO_FLAG	BYTE
OB55_MDL_ADR	WORD
OB55_Z2	WORD
OB55_Z3	WORD
OB55_DATE_TIME	DATE_AND_TIME

Depending on the start event, the tags OB55_Z2 and OB55_Z3 contain different information. This is explained in greater detail below.

Meaning of OB55_Z2

OB55_STRT_INF	Meaning of OB55_Z2
B#16#55	• Low byte: ID for interrupt type "status interrupt" • High byte: Length of the data block supplied by the interrupt
B#16#58	ID for interrupt type: • W#16#0005: Status interrupt

Meaning of OB55_Z3

OB55_STRT_INF	Meaning of OB55_Z3
B#16#55	• Low byte: Specifier – Bits 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bits 3 to 7: Sequence number • High byte: Slot of component triggering the interrupt (module)
B#16#58	Interrupt specifier • Bits 0 to 10: Sequence number (value range 0 to 2047) • Bits 11 to 15: 0

Update interrupt OB (OB 56) (S7-300, S7-400)

Description

The CPU operating system calls OB 56 if an update interrupt was triggered by the slot of a DPV1 slave or of an IO device. This can be the case if you have changed the parameters for the slot of a DPV1 slave or of an IO device (via local or remote access). For precise information on events that trigger an update interrupt, refer to the documentation provided by the manufacturer of the DPV1 slave or the IO device.

Note

If S7-400 CPUs or S7-400 PROFIBUS CPs are set to the "S7 compatible" DP mode, no update interrupt OB (OB 56) can be used.

Local data of the update interrupt OB

The following table shows the temporary (TEMP) tags of the update interrupt OB. The tag names are the default names of OB 56.

Tag	Data type	Description
OB56_EV_CLASS	BYTE	Event class and identifiers: B#16#11 (incoming event)
OB56_STRT_INF	BYTE	• B#16#56: Update interrupt for DP • B#16#59: Update interrupt for PROFINET IO
OB56_PRIORITY	BYTE	Assigned priority class, default value: 2 or 4 (for S7-1500 CPUs)
OB56_OB_NUMBR	BYTE	OB number (56)
OB56_RESERVED_1	BYTE	Reserved
OB56_IO_FLAG	BYTE	Input module: B#16#54 Output module: B#16#55
OB56_MDL_ADDR	WORD	Logical start address of the interrupt triggering component (module)
OB56_LEN	BYTE	Length of the data block supplied by the interrupt
OB56_TYPE	BYTE	ID for the interrupt type "Update interrupt"
OB56_SLOT	BYTE	Slot number of the component triggering the interrupt (module)
OB56_SPEC	BYTE	Specifier: • Bit 0 to 1: Interrupt specifier • Bit 2: Add_Ack • Bit 3 to 7: Seq. no.
OB56_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Note

The meaning of OB56_LEN, OB56_TYPE, OB56_SLOT, OB56_SPEC indicated in the table above applies only to an update interrupt for DP. If the update interrupt is for PROFINET IO, you must organize the local tags as indicated in the next table.

Note

You can obtain complete supplemental information on the interrupt from the telegram by calling the "RALRM" instruction within OB 56.

If you want to program OB 56 dependent on start events, we recommend that you organize the local tags as follows:

Tag	Data type
OB56_EV_CLASS	BYTE
OB56_STRT_INF	BYTE
OB56_PRIORITY	BYTE
OB56_OB_NUMBR	BYTE
OB56_RESERVED_1	BYTE
OB56_IO_FLAG	BYTE
OB56_MDL_ADR	WORD
OB56_Z2	WORD
OB56_Z3	WORD
OB56_DATE_TIME	DATE_AND_TIME

Depending on the start event, the tags OB56_Z2 and OB56_Z3 contain different information. This is explained in greater detail below.

Meaning of OB56_Z2

OB56_STRT_INF	Meaning of OB56_Z2
B#16#56	• Low byte: ID for the interrupt type "Update interrupt" • High byte: Length of the data block supplied by the interrupt
B#16#59	ID for the interrupt type • W#16#0006: Update interrupt • W#16#001E: Upload-and-Retrieval-Alarm In the Siemens Industry Online Support you can find the function block IPARSERV which serves as a application example that you can use to save additional configuration data when required by a PROFINET device or a module used within it to a DB and restore it following maintenance or repair. A PROFINET device sends a Upload-and-Retrieval-Alarm to the IO controller and uses the Update-Alarm OB for this. https://support.industry.siemens.com/cs/ww/en/view/45841087 (https://support.industry.siemens.com/cs/ww/en/view/45841087)

Meaning of OB56_Z3

OB56_STRT_INF	Meaning of OB56_Z3
B#16#56	• Low byte: Specifier – Bits 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bits 3 to 7: Sequence number • High byte: Slot of component triggering the interrupt (module)
B#16#59	Interrupt specifier: • Bits 0 to 10: Sequence number (value range 0 to 2047) • Bits 11 to 15: 0

OB for manufacturer-specific alarms (OB 57) (S7-300, S7-400)

Description

The CPU operating system calls OB 57 if a manufacturer-specific interrupt was triggered by the slot of a DPV1 slave or of an IO device.

Note

If S7-400 CPUs or S7-400 PROFIBUS CPs are set to the "S7 compatible" DP mode, no manufacturer-specific interrupt OB (OB 57) can be used.

Local data of the manufacturer-specific interrupt OB

The table below contains the temporary (TEMP) tags of the manufacturer-specific interrupt OB. Selected tag names are the default names of OB 57.

Tag	Data type	Description
OB57_EV_CLASS	BYTE	Event class and identifiers: B#16#11 (incoming event)
OB57_STRT_INF	BYTE	• B#16#57: Manufacturer interrupt for DP • B#16#5A: Manufacturer interrupt for PROFINET IO • B#16#5B: IO: Profile-specific interrupt
OB57_PRIORITY	BYTE	Assigned priority class, default value: 2 or 4 (for S7-1500 CPUs)
OB57_OB_NUMBR	BYTE	OB number (57)
OB57_RESERVED_1	BYTE	Reserved
OB57_IO_FLAG	BYTE	Input module: B#16#54 Output module: B#16#55
OB57_MDL_ADDR	WORD	Logical start address of the interrupt triggering component (module)
OB57_LEN	BYTE	Length of the data block supplied by the interrupt
OB57_TYPE	BYTE	ID for the interrupt type "Vendor-specific interrupt"

Tag	Data type	Description
OB57_SLOT	BYTE	Slot number of the component triggering the interrupt (module)
OB57_SPEC	BYTE	Specifier: • Bit 0 to 1: Interrupt specifier • Bit 2: Add_Ack • Bit 3 to 7: Seq. no.
OB57_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Note

The meaning of OB57_LEN, OB57_TYPE, OB57_SLOT, OB57_SPEC indicated in the table above applies only to a manufacturer-specific interrupt for DP. If the manufacturer-specific interrupt is for PROFINET IO, you must organize the local tags as indicated in the next table.

Note

You can obtain complete supplemental information on the interrupt from the telegram by calling the "RALRM" instruction within OB 57.

If you want to program OB 57 dependent on start events, we recommend that you organize the local tags as follows:

Tag	Data type
OB57_EV_CLASS	BYTE
OB57_STRT_INF	BYTE
OB57_PRIORITY	BYTE
OB57_OB_NUMBR	BYTE
OB57_RESERVED_1	BYTE
OB57_IO_FLAG	BYTE
OB57_MDL_ADR	WORD
OB57_Z2	WORD
OB57_Z3	WORD
OB57_DATE_TIME	DATE_AND_TIME

Depending on the start event, the tags OB57_Z2 and OB57_Z3 contain different information. This is explained in greater detail below.

Meaning of OB57_Z2

OB57_STRT_INF	Meaning of OB57_Z2
B#16#57	• Low byte: ID for the interrupt type "Vendor-specific interrupt" • High byte: Length of the data block supplied by the interrupt
B#16#5A	ID for interrupt type: • W#16#0020 to 007F: Manufacturer-specific interrupt

Meaning of OB57_Z3

OB57_STRT_INF	Meaning of OB57_Z3
B#16#57	• Low byte: Specifier – Bits 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bits 3 to 7: Sequence number • High byte: Slot of component triggering the interrupt (module)
B#16#5A	Interrupt specifier • Bits 0 to 10: Sequence number (value range 0 to 2047) • Bits 11 to 15: 0

Synchronous cycle interrupt OBs (OB 61 to OB 64) (S7-300, S7-400)

Description

Isochronous mode interrupts give you the option of starting programs isochronously with the DP cycle or PN send clock. OB 6y, 1 <= y <= 4, serves as an interface OB to the isochronous mode interrupt TSAL y. You can set the priority for OB 61 to 64 between 0 (OB deselected) and from 2 to 26.

CAUTION

Direct access

In the case of direct access with L or T commands (e.g., L PIB, T PQB) or with use of the "DPRD_DAT" and "DPWR_DAT" instructions, avoid accessing I/O areas to which process image partitions with OB6x connection (isochronous mode interrupts) are assigned.

Local data for the isochronous mode interrupt OB

The following table describes the temporary (TEMP) tags of the isochronous mode interrupt OB. The default names of OB 61 have been selected as tag names.

Tag	Data type	Description
OB61_EV_CLASS	BYTE	Event class and identifiers: B#16#11: Interrupt is active
OB61_STRT_INF	BYTE	B#16#64: Start request for OB 61 : B#16#67: Start request for OB 64
OB61_PRIORITY	BYTE	Assigned priority class; default value: 25 or 21 (for S7-1500 CPUs)
OB61_OB_NUMBR	BYTE	OB number: 61 ... 64
OB61_RESERVED_1	BYTE	Reserved
OB61_RESERVED_2	BYTE	Reserved
OB61_GC_VIOL	BOOL	GC violation with PROFIBUS DP
OB61_FIRST	BOOL	First use after startup or stop status
OB61_MISSED_EXEC	BYTE	Number of failed starts of OB 61 since last execution of OB 61
OB61_DP_ID	BYTE	DP master system ID of isochronous DP master system (1 to 32) or PROFINET IO System ID of isochronous PNIO system (100 to 115)
OB61_RESERVED_3	BYTE	Reserved
OB61_RESERVED_4	WORD	Reserved
OB61_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Time error OB (OB 80) (S7-300, S7-400)

Description

The CPU operating system calls OB 80 whenever an error occurs while executing an OB, such errors include:

- Exceeding the cycle time
- Acknowledgement error in the execution of an OB
- Setting time forward (time skip) to start an OB

If for example a start event occurs for a cyclic interrupt OB, before the previous execution of the same OB has been completed, then the operating system calls OB 80.

If OB 80 has not been programmed, the CPU changes to STOP mode.

You can use the "DIS_IRT" instruction to disable the call of the time error OB, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Note

If OB 80 is called twice during the same cycle due to the cycle time being exceeded, then the CPU changes to STOP mode. You can prevent this by calling the "RE_TRIGR" instruction at a suitable point.

Note on use of "Report System Errors"

"Report System Errors" delays the processing of interrupts that occur during its execution time.

If the **Execution time of "Report System Errors" > 50% of cycle time** condition is met, this effect is greater (here, the execution time of "Report System Errors" is understood to mean the sum of all processing times of "Report System Errors" within the cycle time). This can lead to the call of the time error OB and possibly to CPU STOP (if the time error OB is not loaded).

The following measures are available to prevent this:

- If you are using isochronous mode interrupt OBs (OB 61 to OB 64), you must ensure that these are called less frequently per time unit than before. You achieve this by increasing the associated "Application cycle factor" parameter for each isochronous mode interrupt OB.
- Increase the ratio (cycle time - execution time of "Report System Errors")/(execution time of "Report System Errors"). The following two options are available for this:
  - If you call "Report System Errors" in cyclic interrupt OBs (OB 30 to OB 38), then increase the cycle clocks for the associated OBs.
  - In OB 1, increase the execution time of the program section independent of "Report System Errors". You achieve this, for example, by using the "WAIT" instruction.

Local data for the time error OB

The following table lists the temporary (TEMP) tags of the time error OB. The default names of OB 80 have been selected as tag names.

Tag	Data type	Description
OB80_EV_CLASS	BYTE	Event class and identifiers: B#16#35
OB80_FLT_ID	BYTE	Error code (possible values: B#16#01, B#16#02, B#16#05, B#16#06, B#16#07, B#16#08, B#16#09, B#16#0A, B#16#0B)
OB80_PRIORITY	BYTE	Priority class: OB 80 runs with priority class 26 in RUN mode, and in the event of an overflow of the OB request buffer with priority class 28 For S7-1500 CPUs: Priority 22
OB80_OB_NUMBR	BYTE	OB number (80)
OB80_RESERVED_1	BYTE	Reserved
OB80_RESERVED_2	BYTE	Reserved
OB80_ERROR_INFO	WORD	Error information: Depends on the error code

Tag	Data type	Description
OB80_ERR_EV_CLASS	BYTE	Class of the event that caused the error
OB80_ERR_EV_NUM	BYTE	Number of the event that caused the error
OB80_OB_PRIORITY	BYTE	Error information: Depends on the error code
OB80_OB_NUM	BYTE	Error information: Depends on the error code
OB80_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The error code-dependent tags have the following meaning:

Error code	Tag	Bit	Meaning
B#16#01			Cycle time exceeded
	OB80_ERROR_INFO		Runtime of last scan cycle (ms)
	OB80_ERR_EV_CLASS		Class of the event that triggered the interrupt
	OB80_ERR_EV_NUM		Number of the event that triggered the interrupt
	OB80_OB_PRIORITY		Priority class of the OB that was being processed when the error occurred
	OB80_OB_NUM		Number of the OB that was active when the error occurred
B#16#02			The requested OB is still being processed.
	OB80_ERROR_INFO		The corresponding temporary tag of the requested OB. This tag is determined by OB80_ERR_EV_CLASS and OB80_ERR_EV_NUM.
	OB80_ERR_EV_CLASS		Class of the event that triggered the interrupt
	OB80_ERR_EV_NUM		Number of the event that triggered the interrupt
	OB80_OB_PRIORITY		Priority class of the OB that caused the error (e.g.: "7" for OB 30/priority class 7, which should have been started but was not able to be started)
	OB80_OB_NUM		Number of the OB that caused the error (e.g.: "30" for OB 30, which should have been started but was not able to be started)
B#16#05			Expired time interrupt due to time jump
	OB80_ERROR_INFO	Bit 0 set	For the time interrupt 0, the starting time lies in the past
		...	
		Bit 7 set	For the time interrupt 7, the starting time lies in the past
		Bit 8 to 15	Not used
	OB80_ERR_EV_CLASS		Not used
	OB80_ERR_EV_NUM		Not used
	OB80_OB_PRIORITY		Not used
	OB80_OB_NUM		Not used
B#16#06	Same as for error code B#16#05		Expired time interrupt upon re-entry into RUN after HOLD

Error code	Tag	Bit	Meaning
B#16#07	Same as for error code B#16#02		Overflow of the OB request buffer for the current priority class (each OB start request for a priority class is entered in the associated OB request buffer; the entry is deleted again after the end of the OB. If a priority class has more OB start requests than the maximum possible number of entries in the associated OB request buffer, OB 80 is called with error code B#16#07.)
B#16#08	Same as for error code B#16#02		Isochronous mode interrupt - time error
B#16#09	Same as for error code B#16#02		Interrupt loss due to high interrupt load
B#16#0A			Resume RUN after CiR
	OB80_ERROR_INFO		CiR synchronization time in ms
B#16#0B			Technology synchronization interrupt - time error
	OB80_ERR_EV_NUM		Number of the event that triggered the interrupt: W#16#116A
	OB80_OB_PRIORITY		Priority class of the OB that was being processed when the error occurred
	OB80_OB_NUM		Number of the OB that was active when the error occurred: 65
B#16#0C			(only for S7-1500 R/H CPUs) cycle time exceeded for the second time
	OB80_ERROR_INFO		Runtime of last scan cycle (ms)
	OB80_ERR_EV_CLASS		Class of the event that triggered the interrupt
	OB80_ERR_EV_NUM		Number of the event that triggered the interrupt
	OB80_OB_PRIORITY		Priority class of the OB that was being processed when the error occurred
	OB80_OB_NUM		Number of the OB that was active when the error occurred

Power supply error organization block (OB 81) (S7-300, S7-400)

Description

The CPU operating system calls OB 81 whenever an event occurs that is triggered by an error or fault related to the power supply (only on an S7-400) or the back-up battery (for either an incoming or outgoing event).

In S7-400, OB 81 is only called in the event of a battery fault if the battery test function has been activated with the BATT.INDIC switch.

The CPU does not change to STOP mode if OB 81 is not programmed.

You can use the "DIS_IRT" instruction to disable the call of the power supply error OB, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Local data for the power supply error OB

The following table lists the temporary (TEMP) tags of the power supply error OB. The default names of OB 81 have been selected as tag names.

Tag	Data type	Description
OB81_EV_CLASS	BYTE	Event class and identifiers: • B#16#38: Outgoing event • B#16#39: Incoming event
OB81_FLT_ID	BYTE	Error code (possible values: B#16#21, B#16#22, B#16#23, B#16#25, B#16#26, B#16#27, B#16#31, B#16#32, B#16#33)
OB81_PRIORITY	BYTE	Priority class: Can be set during configuration e.g., values 2 to 26 can be set for RUN mode.
OB81_OB_NUMBR	BYTE	OB number (81)
OB81_RESERVED_1	BYTE	Reserved
OB81_RESERVED_2	BYTE	Reserved
OB81_RACK_CPU	WORD	• Bits 0 to 7: B#16#00 • Bits 8 to 15: – For a standard CPU: B#16#00 – For a H-CPU: Bits 8 to 10: Rack no., bit 11: 0=Reserve CPU, 1=Master CPU, bits 12 to 15: 1111
OB81_RESERVED_3	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_RESERVED_4	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_RESERVED_5	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_RESERVED_6	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The tags OB81_RESERVED_i, 3 ≤ i ≤ 6 indicate the expansion racks on which the battery backup (error code B#16#31), the back-up voltage (error code B#16#32) or the 24-V power supply (error code B#16#33) has failed or returned. The following table shows which bit is assigned to which expansion rack in the tags OB81_RESERVED_i, 3 ≤ i ≤ 6.

	OB81_RE-SERVED_6	OB81_RE-SERVED_5	OB81_RE-SERVED_4	OB81_RESERVED_3
Bit 0	Reserved	8. expansion rack	16. expansion rack	Reserved
Bit 1	1. expansion rack	9. expansion rack	17. expansion rack	Reserved
Bit 2	2. expansion rack	10. expansion rack	18. expansion rack	Reserved
Bit 3	3. expansion rack	11. expansion rack	19. expansion rack	Reserved
Bit 4	4. expansion rack	12. expansion rack	20. expansion rack	Reserved
Bit 5	5. expansion rack	13. expansion rack	21. expansion rack	Reserved

	OB81_RE-SERVED_6	OB81_RE-SERVED_5	OB81_RE-SERVED_4	OB81_RESERVED_3
Bit 6	6. expansion rack	14. expansion rack	Reserved	Reserved
Bit 7	7. expansion rack	15. expansion rack	Reserved	Reserved

The bits in the tags OB81_RESERVED_i have the following meaning (for the expansion rack concerned):

When the event occurs, the expansion racks are marked (the corresponding bits are set) on which at least one battery or back-up voltage or the 24 V power supply has failed. Expansion racks on which at least one battery or back-up voltage or the 24 V power supply failed earlier are no longer indicated. When the event is eliminated and the backup is restored on at least one expansion rack, this is signaled (the corresponding bits are set).

The following table shows the event that started OB81:

OB81_EV_CLASS	OB81_FLT_ID	Meaning
B#16#39/B#16#38	B#16#21	At least one back-up battery of the central rack is exhausted/problem eliminated (BATTF) **Note:** The incoming event occurs only if one of the two batteries (in the case of redundant buffer batteries) fails. If the other battery then fails, the event does not occur again.
B#16#39/B#16#38	B#16#22	Buffer voltage in the central device is missing / remedied (BAF).
B#16#39/B#16#38	B#16#23	Failure of the 24-V supply in the central device / remedied.
B#16#39/B#16#38	B#16#25	At least one buffer battery in at least one redundant central device is empty / remedied (BATTF).
B#16#39/B#16#38	B#16#26	Buffer voltage in at least one redundant central device is missing / remedied (BAF).
B#16#39/B#16#38	B#16#27	Failure of the 24-V supply in at least one redundant central device / remedied.
B#16#39/B#16#38	B#16#31	At least one buffer battery in at least one expansion unit is empty / remedied (BATTF).
B#16#39/B#16#38	B#16#32	Buffer voltage in at least one expansion unit is missing / remedied (BAF).
B#16#39/B#16#38	B#16#33	Failure of the 24-V supply in at least one expansion unit / remedied.

Sample program for OB 81

A sample STL program will show you how you can read the error code in OB 81.

The program is structured as follows:

The error code in OB 81 (OB81_FLT_ID) is read and compared with the identifier for the "At least one back-up battery of the central rack is exhausted" (B#16#21) and "Back-up voltage in the central rack is not present" (B#16#22) events.

If the error code corresponds to one of the indicated events, the program jumps to the BF label. Otherwise, the block is terminated.

Starting at the BF label, the program sets the *Battery error* tag if the event is an incoming event. If the event is an outgoing event, the program resets this tag.

```
L    B#16#21                                  //Identifier for the "At least one back-up battery
                                              of the central rack is exhausted" event
L    #OB81_FLT_ID                             //Error code in OB 81
==I                                           //If the same,
JC    BF                                      //then jump to BF
L    B#16#22                                  //Identifier for the "Back-up voltage in central
                                              rack is not present" event
==I                                           //If the same as the error code in OB 81,
JC    BF                                      //then jump to BF
BEU                                           //No alarm about battery error

BF:    L    B#16#39                           //Identifier for incoming event
L    #OB81_EV_CLASS                           //Event class and identifiers for OB 81 call
==I                                           //If the same,
S    Battery error                            //then set the battery error (tag from the tag ta-
                                              ble)
L    B#16#38                                  //Identifier for outgoing event
==I                                           //If the same as the event class and identifiers
                                              for the OB 81 call,
R    Battery error                            //then reset the battery error (tag from the tag
                                              table)
```

Diagnostic interrupt OB (OB 82) (S7-300, S7-400)

Description

If a module with diagnostics capability for which you have enabled the diagnostic interrupt detects a change in its diagnostic state, it sends a diagnostic interrupt request to the CPU:

- There is a problem or a component requires maintenance or both (incoming event).
- There is no problem and no further components require maintenance (outgoing event).

The operating system then calls OB 82.

The local tags of OB 82 contain the logical start address as well as four bytes of diagnostics data of the defective module (see the following table).

If OB 82 has not been programmed, the CPU changes to STOP mode.

You can use the "DIS_IRT" instruction to disable the call of the diagnostic interrupt OB, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Note

For PROFINET IO controllers, you can specify during configuration whether or not communication interrupts are to trigger a call of the diagnostic interrupt OB. These interrupts involve diagnostic events of the PROFINET interface.

Default setting: these events do not trigger an OB 82 call.

Local data for diagnostic interrupt OB

The following table lists the temporary (TEMP) tags of the diagnostic interrupt OB. The default names of OB 82 have been selected as the tag names.

Tag	Data type	Description
OB82_EV_CLASS	BYTE	Event class and identifiers: • B#16#38: outgoing event: module error-free (all errors have been corrected.) • B#16#39: incoming event: module will be or remain faulty (with change in the number of its pending errors)
OB82_FLT_ID	BYTE	Error code (B#16#42)
OB82_PRIORITY	BYTE	• Priority class; can be set via configuration • For S7-1500 CPUs: priority, default value: 5
OB82_OB_NUMBR	BYTE	OB number (82)
OB82_RESERVED_1	BYTE	Reserved
OB82_IO_FLAG	BYTE	• For S7-300/S7-400: – Input module: B#16#54 – Output module: B#16#55 • For S7-1500: zero
OB82_MDL_ADDR	WORD	• For S7-300/S7-400: Logical start address of the module where the fault occurred • For S7-1500: Hardware identifier of the hardware object that triggered the diagnostic interrupt
OB82_MDL_DEFECT	BOOL	Module is defective
OB82_INT_FAULT	BOOL	Internal fault
OB82_EXT_FAULT	BOOL	External fault
OB82_PNT_INFO	BOOL	Channel fault
OB82_EXT_VOLTAGE	BOOL	External auxiliary voltage missing
OB82_FLD_CONNCTR	BOOL	Front panel connector not plugged in
OB82_NO_CONFIG	BOOL	Module parameters not assigned
OB82_CONFIG_ERR	BOOL	Incorrect parameters on module

Tag	Data type	Description
OB82_MDL_TYPE	BYTE	• Bit 0 to 3: Module class • Bit 4: Channel information exists • Bit 5: User information exists • Bit 6: Diagnostic interrupt from substitute • Bit 7: Maintenance required
OB82_SUB_MDL_ERR	BOOL	Application module is missing or has an error
OB82_COMM_FAULT	BOOL	Communication problem
OB82_MDL_STOP	BOOL	Operating mode (0: RUN, 1: STOP)
OB82_WTCH_DOG_FLT	BOOL	Watchdog timer responded
OB82_INT_PS_FLT	BOOL	Internal power supply failed
OB82_PRIM_BATT_FLT	BOOL	Battery exhausted
OB82_BCKUP_BATT_FLT	BOOL	Entire backup failed
OB82_RESERVED_2	BOOL	Maintenance demanded
OB82_RACK_FLT	BOOL	Expansion rack failure
OB82_PROC_FLT	BOOL	Processor failure
OB82_EPROM_FLT	BOOL	EPROM fault
OB82_RAM_FLT	BOOL	RAM fault
OB82_ADU_FLT	BOOL	ADC/DAC error
OB82_FUSE_FLT	BOOL	Fuse tripped
OB82_HW_INTR_FLT	BOOL	Hardware interrupt lost
OB82_RESERVED_3	BOOL	Reserved
OB82_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Note

If you are using a DPV1-capable CPU, you can use the "RALRM" instruction to obtain additional information about the interrupt that goes beyond the start information of the OB. This also applies if the DP master is operated in S7 compatible mode.

Programming OB 82

You must add OB 82 to the blocks of your CPU. Write the program that is to be executed in OB 82 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 82, for example, as follows:

- You evaluate the start information in OB 82.
- You carry out detailed diagnostics of the error that occurred.

If a diagnostic interrupt is triggered, the faulty module automatically enters 4 bytes of diagnostics data and their start address in the start information of the diagnostic interrupt OB

and in the diagnostics buffer. This provides you with information about when an error occurred and on which module.

With a suitable program in OB 82, you can evaluate additional diagnostics data for the faulty module (which channel the error occurred on, which error has occurred). You can use the "RDSYSST" instruction to read out module diagnostics data and the "WR_USRMSG" instruction to enter this information in the diagnostics buffer. You can also send a user-defined diagnostics alarm to a logged-on monitoring device.

Insert/remove module interrupt organization block (OB 83) (S7-300, S7-400)

Description

The CPU operating system calls OB 83 after a configured module has been pulled or plugged in.

If you have not programmed OB 83, the CPU changes to STOP mode.

You can use the "DIS_IRT" instruction to disable the pull/plug interrupt OB call, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Removing and inserting modules

Every removal and insertion of a configured module (not allowed: power supply modules, CPUs, adapter casings and IMs) in RUN, STOP, or STARTUP mode results in a pull/plug interrupt. This interrupt causes an entry in the diagnostics buffer and in the system status list for the CPU involved. The pull/plug OB is also started if the CPU is in RUN mode. If this OB has not been programmed, the CPU changes to STOP mode.

The removal and insertion of S7-400 modules is monitored within the system every second. To enable the CPU to detect the removal and insertion of an S7-400 module, a minimum time interval of two seconds must elapse between removal and insertion. This minimum time is slightly higher for other modules.

If you remove a configured module in RUN mode, OB 83 is started. An access error may be detected first if the module is accessed directly or when the process image is updated.

If you insert a module in a configured slot in RUN mode, the operating system checks whether the type of the module inserted matches the configuration. OB 83 is then started and parameters are assigned if the module types match.

Points to note about S7-300, ET 200S and ET 200Pro

- The removal and insertion of central IO devices is not permitted in S7-300.
- With S7-300 CPUs, there is a pull/plug interrupt only for 31x PN/DP CPUs, and then only for PROFINET IO components.
- On an ET 200Pro, for example with an IM 154-4-8 CPU, there is a pull/plug interrupt only for central IO devices.
- On an ET 200S, for example with an IM151-8 PN/DP CPU, there is a pull/plug interrupt only for central IO devices.

Local data for pull/plug OB

The following table lists the temporary (TEMP) tags of the pull/plug OB. The default names of OB 83 have been selected as the tag names.

Tag	Data type	Description
OB83_EV_CLASS	BYTE	Event class and identifiers: • B#16#32: End of module parameter reassignment • B#16#33: Start of module parameter reassignment • B#16#38: Module inserted • B#16#39: Module removed or not responding, or end of parameter reassignment
OB83_FLT_ID	BYTE	Error code (possible values: B#16#51, B#16#54, B#16#55, B#16#56, B#16#57, B#16#58, B#16#61, B#16#63, B#16#64, B#16#65, B#16#66, B#16#67, B#16#68)
OB83_PRIORITY	BYTE	Priority class; can be set via configuration For S71500 CPUs: priority, default value: 6
OB83_OB_NUMBR	BYTE	OB number (83)
OB83_RESERVED_1	BYTE	Identification of module or submodule/interface module
OB83_MDL_TD	BYTE	Range: • B#16#54: I/O range of inputs (PI) • B#16#55: I/O range of outputs (PQ)
OB83_MDL_ADDR	WORD	• Central or distributed PROFIBUS DP: Logical start address of the module affected. If it is a mixed module, it is the smallest logical address used in the module. If the I and O addresses in the mixed block are equal, the logical start address is the one that receives the event identifier. • Distributed PROFINET IO: Logical start address of the module/submodule
OB83_RACK_NUM	INT	• If OB83_RESERVED_1 =B#16#A0: number of the submodule/interface submodule (low byte) • If OB83_RESERVED_1 = B#16#C4: – central: rack number – distributed PROFIBUS DP: number of the DP station (low byte) and DP master system ID (high byte) – Distributed PROFINET IO: physical address: identifier bit (bit 15, 1 = PROFINET IO), IO system ID (bits 11 to 14) and device number (bits 0 to 10)

Tag	Data type	Description
OB83_MDL_TYPE	WORD	For S7-300/S7-400: • Central or distributed PROFIBUS DP: Module type of the module affected (X: not relevant for user). Module types not listed here are documented in the manual of the relevant module. – W#16#X5XX: analog module – W#16#X8XX: function module – W#16#XCXX: CP – W#16#XFXX: digital module – W#16#8340: substitution type ID for input module – W#16#9340: substitution type ID for output module – W#16#A340: substitution type ID for combination module (I/O) – W#16#F340: substitution type ID for empty module or module that cannot be uniquely identified (e.g. with packed addresses) – Module types not listed here are documented in the manual of the relevant module. • Distributed PROFINET IO: – W#16#8101: module type of the inserted module is the same as the module type of the removed module – W#16#8102: module type of the inserted module is not the same as the module type of the removed module
		For S7-1500: • W#16#8101: For all pull events; for all plug events except for the insertion of a module with incorrect module type • W#16#8102: module type of the inserted module is not the same as the module type of the removed module
OB83_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The following table shows which event caused the start of OB 83.

OB83_EV_CLASS	OB83_FLT_ID	Meaning
B#16#39	B#16#51	• PROFINET IO module removed • For the CPUs 1510SP-1 PN and 1512SP-1 PN: IO module removed
B#16#39	B#16#54	• PROFINET IO submodule removed • For the CPUs 1510SP-1 PN and 1512SP-1 PN: IO submodule removed

OB83_EV_CLASS	OB83_FLT_ID	Meaning
B#16#38	B#16#54	• PROFINET IO submodule inserted and matches configured submodule • For the CPUs 1510SP-1 PN and 1512SP-1 PN: IO submodule inserted and matches configured submodule
B#16#38	B#16#55	• PROFINET IO submodule inserted, but does not match configured submodule • For the CPUs 1510SP-1 PN and 1512SP-1 PN: IO submodule inserted, but does not match configured submodule
B#16#38	B#16#56	• PROFINET IO submodule inserted, but error in module parameter assignment • For the CPUs 1510SP-1 PN and 1512SP-1 PN: IO submodule inserted, but error in module parameter assignment
B#16#38	B#16#57	• PROFINET IO submodule or module inserted, but with a problem or maintenance • For the CPUs 1510SP-1 PN and 1512SP-1 PN: IO submodule or module inserted, but with fault or maintenance
B#16#38	B#16#58	• PROFINET IO submodule, access error corrected • For the CPUs 1510SP-1 PN and 1512SP-1 PN: IO submodule, access error corrected
B#16#39	B#16#61	Module removed or not responding OB83_MDL_TYPE: Actual module type
B#16#38	B#16#61	Module inserted, module type OK OB83_MDL_TYPE: Actual module type
B#16#38	B#16#63	Module inserted but module type is incorrect OB83_MDL_TYPE: Actual module type
B#16#38	B#16#64	Module inserted but faulty (module ID cannot be read) OB83_MDL_TYPE: Configured module type
B#16#38	B#16#65	Module inserted but there is an error in module parameter assignment OB83_MDL_TYPE: Actual module type
B#16#39	B#16#66	Module not responding, load voltage error
B#16#38	B#16#66	Module responds again, load voltage error corrected
B#16#33	B#16#67	Start module parameter reassignment
B#16#32	B#16#67	End of module parameter reassignment
B#16#39	B#16#68	Module parameter reassignment terminated with error

Note

If you are using a DPV1 or PROFINET capable CPU, you can use the "RALRM" instruction to obtain additional information about the interrupt that goes beyond the start information of the OB. This also applies if the DP master is operated in S7 compatible mode.

Programming OB 83

You must add OB 83 to the blocks of your CPU. Write the program that is to be executed in OB 83 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 83, for example, as follows:

- You evaluate the start information in OB 83.
- You then assign parameters for the newly inserted module using the "WR_PARM", "WR_DPARM", "PARM_MOD", "WR_REC", and "RD_REC" instructions.

CPU hardware fault organization block (OB 84) (S7-300, S7-400)

Description

The CPU operating system calls OB 84 in the following cases:

- After memory errors have been detected and corrected
- For WinAC RTX: Errors in the operating system (e.g., "blue screen")

If you have not programmed OB 84, the CPU does not change to STOP mode

You can use the "DIS_IRT" instruction to disable the CPU hardware error OB, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Local data for the CPU hardware fault OB

The following table includes the temporary (TEMP) tags of the CPU hardware fault. The default names of OB 84 have been selected as the tag names.

Tag	Data type	Description
OB84_EV_CLASS	BYTE	Event class and identifiers: • B#16#38: outgoing event • B#16#35, B#16#39: incoming event
OB84_FLT_ID	BYTE	Error code (B#16#82, B#16#83, B#16#85, B#16#86, B#16#87)
OB84_PRIORITY	BYTE	Priority class; can be set via configuration (hardware configuration)

Tag	Data type	Description
OB84_OB_NUMBR	BYTE	OB number (84)
OB84_RESERVED_1	BYTE	Reserved
OB84_RESERVED_2	BYTE	Reserved
OB84_RESERVED_3	WORD	Reserved
OB84_RESERVED_4	DWORD	Reserved
OB84_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The following table shows which event caused the start of OB 84:

OB84_EV_CLASS	OB84_FLT_ID	Start event of OB 84
B#16#35	B#16#82	Memory error in operating system detected and corrected
B#16#35	B#16#83	Accumulation of detected and corrected memory errors
B#16#35	B#16#85	Error in PC operating system
B#16#39	B#16#86	Performance of an H-Sync link affected
B#16#35	B#16#87	Multi-bit memory error detected and corrected

Programming OB 84

You must add OB 84 to the blocks of your CPU. Write the program that is to be executed in OB 84 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 84, for example, as follows:

- You evaluate the start information in OB 84.
- You send an alarm to the diagnostics buffer using the "WR_USMSG" instruction.

Priority class error organization block (OB 85) (S7-300, S7-400)

Description

The CPU operating system calls OB 85 whenever one of the following events occurs:

- Start event for a non-loaded OB (except OB 80, OB 81, OB 82, OB 83, and OB 86)
- Error when the operating system accesses a block
- I/O access error during update of the process image by the system (if the OB 85 call was not suppressed due to the configuration).

Note

If OB 85 has not been programmed, the CPU changes to STOP mode when one of these events is detected.

You can use the "DIS_IRT" instruction to disable the priority class error OB, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Local data for the priority class error OB

The following table lists the temporary (TEMP) tags of the priority class error OB. The default names of OB 85 have been selected as the tag names.

Tag	Data type	Description
OB85_EV_CLASS	BYTE	Event class and identifiers: B#16#35, B#16#38 (only for error code B#16#B3 and B#16#B4), B#16#39 (only for error code B#16#B1, B#16#B2, B#16#B3 and B#16#B4)
OB85_FLT_ID	BYTE	Error code (possible values: B#16#A1, B#16#A2, B#16#A3, B#16#A4, B#16#B1, B#16#B2, B#16#B3, B#16#B4)
OB85_PRIORITY	BYTE	Priority class; can be set via configuration
OB85_OB_NUMBR	BYTE	OB number (85)
OB85_RESERVED_1	BYTE	Reserved
OB85_RESERVED_2	BYTE	Reserved
OB85_RESERVED_3	INT	Reserved
OB85_ERR_EV_CLASS	BYTE	Class of the event that caused the error
OB85_ERR_EV_NUM	BYTE	Number of the event that caused the error
OB85_OB_PRIOR	BYTE	Priority class of the OB that was being processed when the error occurred (only for some error codes; for more details, see below).
OB85_OB_NUM	BYTE	Number of the OB that was being processed when the error occurred (only for some error codes; for more details, see below).
OB85_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

If you want to program OB 85 dependent on the possible error codes, we recommend that you organize the local tags as follows:

Tag	Data type
OB85_EV_CLASS	BYTE
OB85_FLT_ID	BYTE
OB85_PRIORITY	BYTE
OB85_OB_NUMBR	BYTE
OB85_DKZ23	BYTE
OB85_RESERVED_2	BYTE
OB85_Z1	WORD
OB85_Z23	DWORD
OB85_DATE_TIME	DATE_AND_TIME

The following table shows which event caused OB 85 to start and how the error code-dependent tags are assigned.

OB85_EV_CLASS	OB85_FLT_ID	Meaning
B#16#35	B#16#A1	Your program or the operating system (based on your configuration) creates a start event for an OB that is not loaded on the CPU. • OB85_Z1: The corresponding temporary tag of the requested OB. This is determined by OB85_Z23. • OB85_Z23: – High word: Class and number of the event causing the OB call – Low word, high byte: Program level active at the time of the error Low word, low byte: Active OB
B#16#35	B#16#A2	Your program or the operating system (based on your configuration) creates a start event for an OB that is not loaded on the CPU. OB85_Z1 and OB85_Z23 as for OB85_FLT_ID=B#16#A1
B#16#35	B#16#A3	Error when the operating system accesses a block • OB85_Z1: Error ID of the operating system – High byte: 1=integrated function, 2=IEC timer – Low byte: 0=no error resolution, 1=block not loaded, 2=area length error, 3=write-protect error • OB85_Z23: – High word: Block number – Low word: Relative address of the MC7 command causing the error. The block type must be taken from OB 85_DKZ23 (B#16#88: OB, B#16#8C: FC, B#16#8E: FB, B#16#8A: DB).
B#16#35	B#16#A4	PROFINET interface DB cannot be addressed
B#16#34	B#16#A4	PROFINET interface DB can be addressed again

OB85_EV_CLASS	OB85_FLT_ID	Meaning
B#16#39	B#16#B1	I/O access error when updating the process image of the inputs • OB85_DKZ23: ID of the type of process image transfer during which the I/O access error occurred – B#16#10: Byte access – B#16#20: Word access – B#16#30: DWord access – B#16#56 or B#16#57: Transmitting a configured consistency range • OB85_Z1: Reserved for internal use by the CPU: Logical start address of the module If OB85_RESERVED_2 has the value B#16#76, then OB85_Z1 contains the return value of the instruction involved ("DPRD_DAT", "DPWR_DAT", "UPDAT_PI", or "UPDAT_PO"). • OB85_Z23: – Byte 0: If OB85_DKZ23=B#16#57: Process image partition no. – Byte 1: Irrelevant, if OB85_DKZ23=B#16#10, 20, or 30; length of the consistency range in bytes, if OB85_DKZ23=B#16#57 – Byte 0 and byte 1, if OB85_DKZ23=B#16#56: Length of the consistency range in bytes – Bytes 2 and 3: The I/O address causing the I/O access error, if OB85_DKZ23=B#16#10, 20, or 30; Logical start address of the consistency range, if OB85_DKZ23=B#16#57
B#16#39	B#16#B2	I/O access error when transferring the process image output to the output modules OB85_DKZ23, OB85_Z1 and OB85_Z23 as for OB85_FLT_ID=B#16#B1
You obtain the error codes B#16#B1 and B#16#B2, if you have configured the repeated OB 85 call for I/O access errors for the system-side process image update.		

OB85_EV_CLASS	OB85_FLT_ID	Meaning
B#16#39/B#16#38	B#16#B3	I/O access error when updating the process image input, incoming/outgoing • OB85_DKZ23: ID of the type of process image transfer during which the I/O access error occurred – B#16#10: Byte access – B#16#20: Word access – B#16#30: DWord access – B#16#56 or B#16#57: Transmitting a configured consistency range • OB85_Z1: Reserved for internal use by the CPU: Logical start address of the module If OB85_RESERVED_2 has the value B#16#76, then OB85_Z1 contains the return value of the instruction involved ("DPRD_DAT", "DPWR_DAT", "UPDAT_PI", or "UPDAT_PO"). • OB85_Z23: – Byte 0: If OB85_DKZ23=B#16#57: Process image partition no. – Byte 1: Irrelevant, if OB85_DKZ23=B#16#10, 20, or 30; length of the consistency range in bytes, if OB85_DKZ23=B#16#57 – Byte 0 and byte 1, if OB85_DKZ23=B#16#56: Length of the consistency range in bytes – Bytes 2 and 3: The I/O address causing the I/O access error, if OB85_DKZ23=B#16#10, 20, or 30; Logical start address of the consistency range, if OB85_DKZ23=B#16#57
B#16#39/B#16#38	B#16#B4	I/O access error when updating the process image of the outputs incoming/outgoing OB85_DKZ23, OB85_Z1, OB85_Z23 as for OB85_FLT_ID=B#16#B3
You obtain the error codes B#16#B3 and B#16#B4 if you only configured the OB 85 call for incoming and outgoing I/O access errors for system side process image table updating. After a cold or warm restart, all access to non-existing inputs and outputs will be reported as incoming I/O access errors during the next process image table update.		

Programming OB 85

You must add OB 85 to the blocks of your CPU. Write the program that is to be executed in OB 85 in the generated block, and load the block to the CPU as part of your user program.

You can use OB 85, for example, as follows:

- To evaluate the start information of OB 85 and to determine which module is missing or defective (information regarding the module start address).
- You determine the slot of the module involved using the "LGC_GADR" instruction.

Rack failure organization block (OB 86) (S7-300, S7-400)

Description

The CPU operating system calls OB 86 in the following cases:

- Failure of a central expansion unit (not for S7-300) is detected (both if incoming or outgoing event).
- Failure of a DP master system is detected (both if incoming or outgoing event).
- Failure of a distributed I/O device (PROFIBUS DP or PROFINET IO) is detected (both if incoming or outgoing event).
- You have deactivated a distributed I/O device (PROFIBUS DP or PROFINET IO) with the "D_ACT_DP" instruction using MODE=4.
- You have activated a distributed I/O device (PROFIBUS DP or PROFINET IO) with the "D_ACT_DP" instruction using MODE=3.
- The failure of a PROFINET IO system or a PROFINET IO station or some of the submodules of a PROFINET I-device has been detected.

If OB 86 has not been programmed, the CPU changes to STOP mode when this type of error is detected.

You can use the "DIS_IRT" instruction to disable the rack failure OB, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Local data for the rack failure OB

The following table lists the temporary (TEMP) tags of the rack failure OB. The default names of OB 86 have been selected as the tag names.

Tag	Data type	Description
OB86_EV_CLASS	BYTE	Event class and identifiers: • B#16#32: Activation of a device with the "D_ACT_DP" instruction using MODE=3 • B#16#33: Deactivation of a device with the "D_ACT_DP" instruction using MODE=4 • B#16#38: outgoing event • B#16#39: incoming event
OB86_FLT_ID	BYTE	Error code (possible values: B#16#C1, B#16#C2, B#16#C3, B#16#C4, B#16#C5, B#16#C6, B#16#C7, B#16#C8, B#16#C9, B#16#CA, B#16#CB, B#16#CC, B#16#CD, B#16#CE, B#16#CF, B#16#F8, B#16#F9)
OB86_PRIORITY	BYTE	Priority class; can be set via configuration For S7-1500 CPUs: priority, default value: 6
OB86_OB_NUMBR	BYTE	OB number (86)
OB86_RESERVED_1	BYTE	Reserved

Tag	Data type	Description
OB86_RESERVED_2	BYTE	Reserved
OB86_MDL_ADDR	WORD	Depends on the error code
OB86_RACKS_FLTD	ARRAY [0 ..31] OF BOOL	Depends on the error code
OB86_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

If you want to program OB 86 depending on the possible error codes, we recommend that you organize the local tags as follows:

Tag	Data type
OB86_EV_CLASS	BYTE
OB86_FLT_ID	BYTE
OB86_PRIORITY	BYTE
OB86_OB_NUMBR	BYTE
OB86_RESERVED_1	BYTE
OB86_RESERVED_2	BYTE
OB86_MDL_ADDR	WORD
OB86_Z23	DWORD
OB86_DATE_TIME	DATE_AND_TIME

The following table shows which event caused the start of OB 86.

OB86_EV_CLASS	OB86_FLT_ID	Meaning
B#16#39	B#16#C1	Expansion rack failure • OB86_MDL_ADDR: Logical start address of the IM • OB86_Z23: contains a bit for each possible expansion rack: Each expansion rack that caused a call of OB 86 is reported as having failed (the respective bits are set). Expansion racks that previously failed are no longer shown. – Bit 0: always 0 – Bit 1: 1. expansion rack – : – Bit 21: 21. expansion rack – Bit 22 to 29: always 0 – Bit 30: Failure of at least one expansion rack in the SIMATIC S5 area – Bit 31: always 0
B#16#38	B#16#C1	Expansion rack operational again OB86_MDL_ADDR as for OB86_FLT_ID=B#16#C1. The expansion racks that are operational again are reported in OB86_Z23 (the respective bits are set).

OB86_EV_CLASS	OB86_FLT_ID	Meaning
B#16#38	B#16#C2	Expansion rack operational again (expansion rack failure with discrepancy between expected and actual configuration) • OB86_MDL_ADDR: Logical start address of the IM • OB86_Z23: Contains one bit for every possible expansion rack, see OB86_FLT_ID B#16#C1. Meaning of a set bit: In the affected expansion rack: – modules with incorrect module IDs exist. – Configured modules missing. – At least one module is defective.
B#16#39	B#16#C3	Distributed I/O: Failure of a DP master system Only an incoming event causes OB 86 start with error code B#16#C3. An outgoing event causes OB 86 start with error code B#16#C4 and event class B#16#38: The return of each lower-level DP device has an OB 86 start as consequence.) • OB86_MDL_ADDR: Logical start address of the DP master • OB86_Z23: DP master system ID: – Bit 0 to 7: Reserved – Bit 8 to 15: DP master system ID – Bit 16 to 31: Reserved
B#16#39/B#16#38	B#16#C4	Failure of a DP device • OB86_MDL_ADDR: Logical start address of the DP master • OB86_Z23: Address of the affected DP slave: – Bit 0 to 7: No. of the DP device – Bit 8 to 15: DP master system ID – Bit 16 to 30: Logical start address, if an S7 slave is used, or diagnostics address if a DP standard slave is used. – Bit 31: I/O identifier
B#16#39/B#16#38	B#16#C5	Return of a DP device, but device faulty OB86_MDL_ADDR and OB86_Z23 as with FLT_ID=B#16#C4

OB86_EV_CLASS	OB86_FLT_ID	Meaning
B#16#38	B#16#C6	Expansion unit return, but error in module parameter assignment • OB86_MDL_ADDR: Logical start address of the IM • OB86_Z23: contains a bit for each possible expansion rack: – Bit 0: always 0 – Bit 1: 1. expansion rack – … – Bit 21: 21. expansion rack – Bit 22 to 30: Reserved – Bit 31: always 0 • Meaning of the set bit: In the affected expansion rack, – modules with incorrect module IDs exist. – Modules with missing or incorrect parameters exist.
B#16#38	B#16#C7	Return of a DP device, but error in module parameter assignment • OB86_MDL_ADDR: Logical start address of the DP master • Address of the affected DP slave: – Bit 0 to 7: No. of the DP device – Bit 8 to 15: DP master system ID – Bit 16 to 30: Logical start address of the DP slave – Bit 31: I/O identifier
B#16#38	B#16#C8	Return of a DP device, however discrepancy in configured and actual configuration • OB86_MDL_ADDR: Logical start address of the DP master • OB86_Z23: Address of the affected DP slave: – Bit 0 to 7: No. of the DP device – Bit 8 to 15: DP master system ID – Bit 16 to 30: Logical start address of the DP slave – Bit 31: I/O identifier
B#16#32/B#16#33	B#16#C9	Activation/deactivation of a DP slave with the "D_ACT_DP" instruction using MODE=3/MODE=4 • OB86_MDL_ADDR: Logical start address of the DP master • OB86_Z23: Address of the affected DP slave – Bit 0 to 7: No. of the DP device – Bit 8 to 15: DP master system ID – Bit 16 to 30: Logical start address, if an S7 slave is used, or diagnostics address if a DP standard slave is used. – Bit 31: I/O identifier

OB86_EV_CLASS	OB86_FLT_ID	Meaning
B#16#39	B#16#CA	PROFINET IO system failure • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: 0 (station number) – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 31: 0
B#16#39/38	B#16#CB	PROFINET IO device failure/device return • OB86_RESERVED_1: B#16#C4 • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier Note: Before you check in your program whether a PROFINET IO station failure is the cause for the OB 86 start, you should check whether a PROFINET IO system failure occurred.
B#16#/38	B#16#CC	PROFINET IO station return with problem or maintenance • OB86_RESERVED_1: B#16#C4 • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier
B#16#38	B#16#CD	PROFINET IO device return, expected and actual configuration differ • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier

OB86_EV_CLASS	OB86_FLT_ID	Meaning
B#16#38	B#16#CE	PROFINET IO station return, error in module parameter assignment • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier
B#16#32/B#16#33	B#16#CF	Activation/deactivation of a PROFINET IO device with the "D_ACT_DP" instruction using MODE=3/MODE=4 • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier
B#16#39/B#16#38	B#16#F8	Failure/return of some of the submodules of a PROFINET I-device • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier
B#16#38	B#16#F9	Return of some of the submodules of a PROFINET I-device with a device configuration difference • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier

Note

If you are using a DPV1 or PROFINET capable CPU, you can use the "RALRM" instruction to obtain additional information about the interrupt that goes beyond the start information of the OB. This also applies if the DP master is operated in S7 compatible mode.

Programming OB 86

You must add OB 86 to the blocks of your CPU. Write the program that is to be executed in OB 86 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 86, for example, as follows:

- To evaluate the start information of OB 86 and to determine which rack is defective or missing
- You enter an alarm in the diagnostic buffer using the "WR_USMSG" instruction and send the alarm to a monitoring device.

Communication error organization block (OB 87) (S7-300, S7-400)

Description

The CPU operating system calls OB 87 when an event occurs that was triggered by a communication error.

If you have not programmed OB 87 and a start event for OB 87 occurs, your CPU will react as follows:

- An S7-300 CPU changes to STOP mode.
- An S7-400 CPU does not change to STOP mode.

You can use the "DIS_IRT" instruction to disable the communication error OB, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Local data for communication error OB

The following table lists the temporary (TEMP) tags of the communication error OB. The default names of OB 87 have been selected as the tag names.

Tag	Data type	Description
OB87_EV_CLASS	BYTE	Event class and identifiers: B#16#39
OB87_FLT_ID	BYTE	Error code (possible values: B#16#D2, B#16#D3, B#16#D4, B#16#D5, B#16#E1, B#16#E2, B#16#E3, B#16#E4, B#16#E5, B#16#E6)

Tag	Data type	Description
OB87_PRIORITY	BYTE	Priority class; can be set via configuration
OB87_OB_NUMBR	BYTE	OB number (87)
OB87_RESERVED_1	BYTE	Reserved
OB87_RESERVED_2	BYTE	Reserved
OB87_RESERVED_3	WORD	Depends on the error code
OB87_RESERVED_4	DWORD	Depends on the error code
OB87_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The error code-dependent tags have the following meaning:

Error code	Tag	Meaning
B#16#D2		It is not possible to send the diagnostics entries at present.
B#16#D3		Synchronization message frames cannot be sent (master).
B#16#D4		Illegal time jump through time synchronization
B#16#D5		Error in adopting the synchronization time (slave)
	OB87_RESERVED_3	Includes no extra information
	OB87_RESERVED_4	Includes no extra information
B#16#E1		Incorrect message frame identifier in global data communication
B#16#E2		GD package status cannot be entered in a DB
	OB87_RESERVED_3	DB number
	OB87_RESERVED_4	• High word: Includes no extra information
		• Low word: GD circle number (high byte), GD packet number (low byte)
B#16#E3		Message frame length error in global data communication
B#16#E4		Illegal GD package number received
	OB87_RESERVED_3	Interface identifier (0: K bus, 1: MPI)
	OB87_RESERVED_4	• High byte: GD loop number • Low byte: Includes no extra information
B#16#E5		Error accessing DB when exchanging data via communication function blocks
	OB87_RESERVED_1	Block type: B#16#88: OB, B#16#8A: DB, B#16#8C: FC, B#16#8E: FB
	OB87_RESERVED_3	Reserved for internal use by the CPU
	OB87_RESERVED_4	• High word: Number of the block with the MC7 instruction that has caused the error • Low word: Relative address of the MC7 instruction that has caused the error
B#16#E6		GD package status cannot be entered in a DB
	OB87_RESERVED_3	DB number
	OB87_RESERVED_4	Includes no extra information

Programming OB 87

You must add OB 87 to the blocks of your CPU. Write the program that is to be executed in OB 87 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 87, for example, as follows:

- You evaluate the start information in OB 87.
- You create a data block when the data block for the status information of the shared data communication is missing.

Processing interrupt OB (OB 88) (S7-300, S7-400)

Description

The CPU operating system calls OB 88 after a program block execution is been aborted. Possible causes of the abort are:

- Nesting depth exceeded for synchronous errors
- Nesting depth exceeded for block calls (U stack)
- Error during allocation of local data

If you have not programmed OB 88 and processing is interrupted, the CPU goes to STOP mode (event W#16#4570).

If program block execution is aborted in priority class 28, the CPU goes into STOP mode.

You can use the "DIS_IRT" instruction to disable the processing interrupt OB, the "EN_IRT" instruction to re-enable it, and the "DIS_AIRT" and "EN_AIRT" instructions to delay it.

Local data of the processing interrupt OB

The following table lists the temporary (TEMP) tags of the processing interrupt OB. The default names of OB 88 have been selected as the tag names.

Tag	Data type	Description
OB88_EV_CLASS	BYTE	Event class and identifiers: B#16#35
OB88_SW_FLT	BYTE	Error code Possible values: • B#16#71: Nesting depth of nesting levels exceeded • B#16#72: Nesting depth of Master Control Relay exceeded • B#16#73: Nesting depth exceeded for synchronous errors • B#16#74: Nesting depth of block calls (U stack) exceeded • B#16#75: Nesting depth of block calls (B stack) exceeded • B#16#76: Error in local data allocation • B#16#78: Unknown instruction • B#16#7A: Jump instruction with destination outside of the block Please refer to the operation list for your CPU to determine which error codes apply to your CPU.
OB88_PRIORITY	BYTE	Priority class: 28
OB88_OB_NUMBR	BYTE	OB number (88)
OB88_BLK_TYPE	BYTE	Type of block in which the error occurred: • B#16#88: OB • B#16#8C: FC • B#16#8E: FB • B#16##00: Could not determine interrupt source
OB88_RESERVED_1	BYTE	Reserved
OB88_FLT_PRIORITY	BYTE	Priority class of the OB that has caused the error
OB88_FLT_OB_NUMBR	BYTE	Number of the OB that has caused the error
OB88_BLK_NUM	WORD	Number of the block with the MC7 instruction that has caused the error
OB88_PRG_ADDR	WORD	Relative address of the MC7 instruction that has caused the error
OB88_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Programming OB 88

You must add OB 88 to the blocks of your CPU. Write the program that is to be executed in OB 88 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 88, for example, as follows:

- You evaluate the start information in OB 88.
- You enter the cause of the error in an alarm data block.

Background organization block (OB 90) (S7-300, S7-400)

Description

With S7 you can monitor a maximum cycle time and guarantee a minimum cycle time. If the execution time of OB 1 including all the nested interrupt processing and system activities is less than the minimum cycle time that you have specified, the operating system reacts as follows:

- It calls the background OB (providing it exists on the CPU).
- It delays the next OB 1 start (if OB 90 does not exist on the CPU).

Function of the background OB

OB 90 has the lowest priority of all OBs. It is interrupted by any system activity and any interrupt (even by OB 1 after the minimum cycle time has elapsed) and is only resumed if the selected minimum cycle time has not yet been reached. The only exception is the execution of instructions that are started in OB 90. These are executed with the priority of OB 1 and are therefore not interrupted by OB 1. There is no time monitoring of OB 90.

The user program in OB 90 is processed starting with the first instruction in the following situations:

- Following a warm, cold, or hot restart
- After loading or deleting a block
- After downloading OB 90 to the CPU in RUN mode
- After terminating the background cycle

Note

In configurations in which the minimum cycle time and the maximum cycle time are almost the same, the cycle time may be exceeded unexpectedly when instructions are called in the background OB.

Local data for the background OB

The following table describes the temporary (TEMP) tags of OB 90. The tag names are the default names of OB 90.

Tag	Data type	Description
OB90_EV_CLASS	BYTE	Event class and identifiers: B#16#11: Active
OB90_STRT_INF	BYTE	• B#16#91: Warm restart/cold restart/hot restart • B#16#92: Deletion of a block • B#16#93: Downloading of OB 90 to the CPU in RUN mode • B#16#95: Termination of the background cycle
OB90_PRIORITY	BYTE	Priority class: 29 (corresponds to priority 0.29)
OB90_OB_NUMBR	BYTE	OB number (90)
OB90_RESERVED_1	BYTE	Reserved
OB90_RESERVED_2	BYTE	Reserved
OB90_RESERVED_3	INT	Reserved
OB90_RESERVED_4	INT	Reserved
OB90_RESERVED_5	INT	Reserved
OB90_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Startup organization blocks (OB 100, OB 101 and OB 102) (S7-300, S7-400)

Startup types

A distinction is made between the following types of startup

- Hot restart (S7-400)
- Warm restart
- Cold restart

In the following table, you can see which OB is called by the operating system during startup.

Startup type	Corresponding OB
Hot restart	OB 101
Warm restart	OB 100
Cold restart	OB 102

Startup events

The CPU executes a startup as follows:

- After POWER ON
- After a request by a communication function (via a menu command from the programming device or by calling the "START" or "RESUME" instruction on a different CPU)

Depending on the start event, the particular CPU, and its parameters, the appropriate startup OB (OB 100, OB 101, or OB 102) is called. With suitable programming, you can make certain default settings for your cyclic program in these blocks.

Local data for startup OBs

The following table shows the temporary (TEMP) tags of a startup OB. The default names have been selected as tag names.

Tag	Data type	Description
OB10x_EV_CLASS	BYTE	Event class and identifiers:B#16#13: active
OB10x_STRTUP	BYTE	Startup request: • B#16#81: Manual warm restart request • B#16#82: Automatic warm restart request • B#16#83: Request for manual hot restart • B#16#84: Request for automatic hot restart • B#16#85: Request for manual cold restart • B#16#86: Request for automatic cold restart • B#16#87: Master: Request for manual cold restart • B#16#88: Master: Request for automatic cold restart • B#16#8A: Master: Request for manual warm restart • B#16#8B: Master: Request for automatic warm restart
OB10x_PRIORITY	BYTE	Priority class: 27 or 1 (for S7-1500 CPUs)
OB10x_OB_NUMBR	BYTE	OB number (100, 101, or 102)
OB10x_RESERVED_1	BYTE	Reserved
OB10x_RESERVED_2	BYTE	Reserved
OB10x_STOP	WORD	Number of the event that caused the CPU to stop
OB10x_STRT_INFO	DWORD	Supplementary information about the current start-up
OB10x_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The following table shows the assignment of the tags OB10x_STR_INFO.

Bit no.	Meaning	Possible binary values	Explanation
31-24	Startup information	0000 xxxx	Rack number 0 (H CPUs only)
		0100 xxxx	Rack number 1 (H-CPUs only)
		1000 xxxx	Rack number 2 (H-CPUs only)
		0001 xxxx	Multicomputing (S7-400 only)
		0010 xxxx	Operation of more than one CPU in the segmented rack (S7-400 only)
		xxxx xxx0	No difference between expected and actual configuration (S7-300 only)
		xxxx xxx1	Difference between expected and actual configuration (S7-300 only)
		xxxx xx0x	No difference between expected and actual configuration
		xxxx xx1x	Difference between expected and actual configuration
		xxxx x0xx	Not an H CPU
		xxxx x1xx	H CPU
		xxxx 0xxx	Clock for time stamp not battery-backed at last POWER ON
		xxxx 1xxx	Clock for time stamp battery-backed at last POWER ON

Bit no.	Meaning	Possible binary values	Explanation
23-16	Startup type just performed Note: These bits have no meaning for S7-1500 CPUs.	0000 0001	Warm restart for multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 0011	Warm restart triggered by mode selector
		0000 0100	Warm restart triggered by PG operator input
		0000 0101	Cold restart in multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 0111	Cold restart triggered with mode selector
		0000 1000	Cold restart triggered by PG operator input
		0000 1010	Hot restart in multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 1011	Hot restart triggered with mode selector (S7-400 only)
		0000 1100	Hot restart triggered by PG operator input (S7-400 only)
		0001 0000	Automatic warm restart after battery-backed POWER ON
		0001 0001	Cold restart after battery-backed POWER ON according to parameter assignment
		0001 0011	Warm restart triggered with mode selector; last POWER ON battery-backed
		0001 0100	Warm restart triggered by PG operator input; last POWER ON battery-backed
		0010 0000	Automatic warm restart after non-battery-backed POWER ON (with memory reset by system)
		0010 0001	Cold restart after non-battery-backed POWER ON (with memory reset by system)
		0010 0011	Warm restart triggered with mode selector; last POWER ON not battery-backed
		0010 0100	Warm restart triggered by PG operator input; last POWER ON not battery-backed
		1010 0000	Automatic hot restart after battery-backed POWER ON according to parameter assignment (S7-400 only)
15-12	Permissibility of automatic startup Note: These bits have no meaning for S7-1500 CPUs.	0000	Automatic startup illegal, memory reset requested
		0001	Automatic startup illegal, parameter modifications, etc. necessary
		0111	Automatic warm restart permitted
		1111	Automatic restart (warm/hot) permitted (S7-400 only)

Bit no.	Meaning	Possible binary values	Explanation
11-8	Permissibility of manual startup Note: These bits have no meaning for S7-1500 CPUs.	0000	Startup illegal, memory reset requested
		0001	Startup illegal, parameter modifications, etc. necessary
		0111	Warm restart permitted
		1111	Warm restart and hot restart permitted (S7-400 only)
7-0	Last valid intervention or setting of the automatic startup at POWER ON	0000 0000	No startup
		0000 0001	Warm restart in multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 0011	Warm restart triggered by mode selector
		0000 0100	Warm restart triggered by PG operator input
		0000 0101	Cold restart in multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 0111	Cold restart triggered with mode selector
		0000 1000	Cold restart triggered by PG operator input
		0000 1010	Hot restart in multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 1011	Hot restart triggered with mode selector (S7-400 only)
		0000 1100	Hot restart triggered by PG operator input (S7-400 only)
		0001 0000	Automatic warm restart after battery-backed POWER ON
		0001 0001	Cold restart after battery-backed POWER ON according to parameter assignment
		0001 0011	Warm restart triggered with mode selector; last POWER ON battery-backed
		0001 0100	Warm restart triggered by PG operator input; last POWER ON battery-backed
		0010 0000	Automatic warm restart after non-battery-backed POWER ON (with memory reset by system)
		0010 0001	Cold restart after non-battery-backed POWER ON (with memory reset by system)
		0010 0011	Warm restart triggered with mode selector; last POWER ON not battery-backed
		0010 0100	Warm restart triggered by PG operator input; last POWER ON not battery-backed
		1010 0000	Automatic hot restart after battery-backed POWER ON according to parameter assignment (S7-400 only)

See also

Principles of the STARTUP mode (Page 710)

Programming error organization block (OB 121) (S7-300, S7-400)

Description

The CPU operating system calls OB 121 whenever an event occurs that is caused by an error related to the processing of the program. For example, if your program calls a block that has not been loaded on the CPU, OB 121 is called.

Function of the programming error OB

OB 121 is executed in the same priority class as the interrupted block.

If OB 121 is not programmed, the CPU changes the operating mode from RUN to STOP.

S7 provides the following instructions for use in masking and unmasking start events of OB 121 during the execution of your program:

- The "MSK_FLT" instruction masks certain error codes.
- The "DMSK_FLT" instruction unmasks the error codes that were masked by the "MSK_FLT" instruction.
- The "READ_ERR" instruction reads the event status register.

Local data for the programming error OB

The following table lists the temporary (TEMP) tags of the programming error OB. The default names of OB 121 have been selected as the tag names.

Tag	Data type	Description
OB121_EV_CLASS	BYTE	Event class and identifiers: B#16#25
OB121_SW_FLT	BYTE	Error code (possible values: B#16#21, B#16#22, B#16#23, B#16#24, B#16#25, B#16#26, B#16#27, B#16#28, B#16#29, B#16#30, B#16#31, B#16#32, B#16#33, B#16#34, B#16#35, B#16#3A, B#16#3C, B#16#3D, B#16#3E, B#16#3F)
OB121_PRIORITY	BYTE	Priority class of the OB where the error occurred For S7-1500 CPUs: priority, default value: 7
OB121_OB_NUMBR	BYTE	OB number (121)
OB121_BLK_TYPE	BYTE	Type of block where the error occurred (no valid value is entered here in case of S7-300): B#16#88: OB, B#16#8A: DB, B#16#8C: FC, B#16#8E: FB
OB121_RESERVED_1	BYTE	Reserved
OB121_FLT_REG	WORD	Source of the error (depends on error code); for example: • Register in which the conversion error occurred • Incorrect address (read/write error) • Incorrect timer/counter/block number • Incorrect memory area

Tag	Data type	Description
OB121_BLK_NUM	WORD	Number of the block with the MC7 command that caused the error (no valid number is entered here for an S7-300)
OB121_PRG_ADDR	WORD	Relative address of the MC7 instruction that has caused the error FB) (no valid number is entered here for an S7-300)
OB121_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The error code-dependent tags have the following meaning:

Error code	Tag	Meaning
B#16#21		BCD conversion error
	OB121_FLT_REG	Identifier for the affected tab (W#16#0000: battery 1)
B#16#22		Area length error when reading
	OB121_RESERVED_1	• Bit 7 to 4 type of access: – 0: Bit access, – 1: Byte access, – 2: Word access, – 3: DWord access • Bit 3 to 0 memory area: – 0: I/O area – 1: Process image input – 2: Process image output – 3: Bit memory – 4: Global DB – 5: Instance DB – 6: own local data – 7: Local data of the caller
B#16#23		Area length error when writing
	OB121_RESERVED_1	Same as for error code B#16#22
B#16#28		Read access to a byte, word or double word with a pointer, the bit address of which is not 0
	OB121_RESERVED_1	Same as for error code B#16#22
B#16#29		Write access to a byte, word or double word with a pointer, the bit address of which is not 0 Faulty byte address. Refer to OB121_RESERVED_1 for the data area and the type of access
	OB121_RESERVED_1	Same as for error code B#16#22
B#16#24		Area error when reading
	OB121_FLT_REG	The low byte contains the identifier for the unauthorized area (B#16#86 own local data area)
B#16#25		Area error when writing

Error code	Tag	Meaning
	OB121_FLT_REG	The low byte contains the identifier for the unauthorized area (B#16#86 own local data area)
B#16#26		Error in timer number
	OB121_FLT_REG	Unauthorized number
B#16#27		Error in counter number
	OB121_FLT_REG	Unauthorized number
B#16#30		Write access to a write-protected global DB
	OB121_FLT_REG	Unauthorized DB number
B#16#31		Write access to a write-protected instance DB
	OB121_FLT_REG	Unauthorized DB number
B#16#32		DB number error when accessing a global DB
	OB121_FLT_REG	Unauthorized DB number
B#16#33		DB number error when accessing an instance DB
	OB121_FLT_REG	Unauthorized DB number
B#16#34		Number error during FC call
	OB121_FLT_REG	FC number
B#16#35		FB number error during FC call
	OB121_FLT_REG	FB number
B#16#3A		Access to a DB that is not loaded; the DB number is located in the permitted area
	OB121_FLT_REG	DB number
B#16#3C		Access to an FC that is not loaded; the FC number lies in the permissible area
	OB121_FLT_REG	FC number
B#16#3D		Access to an instruction that is not available; the SFC number is located in the permitted area
	OB121_FLT_REG	SFC number
B#16#3E		Access to an FB that is not loaded; the FB number lies in the permissible area
	OB121_FLT_REG	FB number
B#16#3F		Access to an SFB that is not available; the SFB number lies in the permissible area
	OB121_FLT_REG	SFB number

Programming OB 121

You must add OB 121 to the blocks of your CPU. Write the program that is to be executed in OB 121 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 121, for example, as follows:

- You evaluate the start information in OB 121.
- You enter the cause of the error in an alarm data block.

I/O access error organization block (OB 122) (S7-300, S7-400)

Description

The CPU operating system calls OB 122 whenever an error occurs while accessing data on a module. For example, if the CPU detects a read error when accessing data on a signal module, the operating system calls OB 122.

Function of the I/O access error OB

OB 122 is executed in the same priority class as the interrupted block. If OB 122 is not programmed, the CPU changes the operating mode from RUN to STOP.

S7 provides the following instructions for use in masking and unmasking start events of OB 122 during the execution of your program:

- The "MSK_FLT" instruction masks certain error codes.
- The "DMSK_FLT" instruction unmasks the error codes that were masked by the "MSK_FLT" instruction.
- The "READ_ERR" instruction reads the event status register.

Local data for the I/O access error OB

The following table lists the temporary (TEMP) tags of the I/O access error OB. The default names of OB 122 have been selected as the tag names.

Tag	Data type	Description
OB122_EV_CLASS	BYTE	Event class and identifiers: B#16#29
OB122_SW_FLT	BYTE	Error code: • B#16#42: I/O access error, reading. • B#16#43: I/O access error, writing.
OB122_PRIORITY	BYTE	Priority class of the OB where the error occurred For S7-1500 CPUs: priority, default value: 7
OB122_OB_NUMBR	BYTE	OB number (122)
OB122_BLK_TYPE	BYTE	Type of block where the error occurred (B#16#88: OB, B#16#8C: FC, B#16#8E: FB) (no valid number is entered here for an S7-300)
OB122_MEM_AREA	BYTE	Memory area and access type: • Bit 7 to 4: Access type – 0: Bit access – 1: Byte access – 2: Word access – 3: DWord access • Bit 3 to 0: memory area – 0: I/O area – 1: Process image input – 2: Process image output

Tag	Data type	Description
OB122_MEM_ADDR	WORD	Memory address where the error occurred
OB122_BLK_NUM	WORD	Number of the block with the MC7 command that caused the error (no valid number is entered here for an S7-300)
OB122_PRG_ADDR	WORD	Relative address of the MC7 command that caused the error (no valid value is entered here for an S7-300)
OB122_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Sample program for OB 122

The following sample program will provide a substitute value in the "REPL_VAL" instruction. If an input module fails, the processing of the L PEB0 instruction generates a synchronous error and starts OB 122. The load command reads in the value 0 by default. However, you can use the "REPL_VAL" instruction to specify any substitute value suitable for the process. This instruction replaces the ACCU content with the specified substitute value.

```
STL                                  Description
L    B#16#42                         //Identifier for "I/O access error reading" event
L    #OB122_SW_FLT                   //Error code of OB 122
==I                                  //If the same,
SPB    Qfeh                          //then jump to Qfeh
L    B#16#43                         //Identifier for "I/O access error writing" event
<>I                                  //If not the same as the error code in OB 122,
SPB    STOP                          //then jump to STOP
Qfeh:
CALL    "REPL_VAL"                   //Call the "REPL_VAL" instruction
    VAL:= DW#16#2912                 //Substitute value that is loaded in ACCU 1
    RET_VAL:= #error                 //Save the return value in #error
L    #error                          //Return value of the "REPL_VAL" instruction
L    0
==I                                  //If the same as 0 (no error has occurred),
BEC                                  //then end the block
STOP:
CALL    "STP"                        //Call the "STP" instruction: The CPU switches to STOP mode.
```

Events (S7-300, S7-400)

Events and event ID (S7-300, S7-400)

Event

All events are numbered within the SIMATIC S7. This allows you to relate a message text to an event.

Event ID

An event ID is assigned to every event. The event ID is structured as follows:

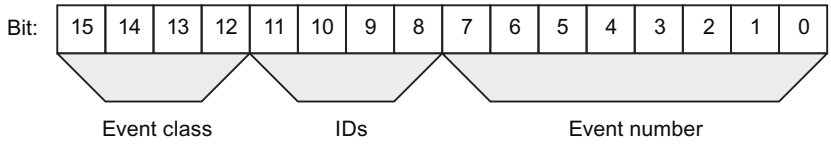

Figure 1-19 Structure of the event ID.

Event class

The event classes are as follows:

Number	Event class
1	Standard OB events
2	Synchronous errors
3	Asynchronous errors
4	Mode transitions
5	Runtime events
6	Communication events
7	Events for fail-safe and fault-tolerant systems
8	Standardized diagnostics data on modules
9	Predefined user events
A, B	Freely definable events
C , D, E	Reserved
F	Reserved for modules not in central rack (for example, CPs or FMs)

Identifier

The identifier is used to distinguish the type of events. The four bits have the following significance:

Bit no. in the event ID	Meaning
8	= 0: outgoing event = 1 incoming event
9	= 1: Entry in diagnostics buffer
10	= 1: internal error
11	= 1: external error

Event class 1 - standard OB events (S7-300, S7-400)

Description of the event IDs

Event ID	Event
W#16#1381	Manual warm restart request
W#16#1382	Automatic warm restart request
W#16#1383	Request for manual hot restart
W#16#1384	Request for automatic hot restart
W#16#1385	Request for manual cold restart
W#16#1386	Request for automatic cold restart
W#16#1387	Master CPU: Request for manual cold restart
W#16#1388	Master CPU: Request for automatic cold restart
W#16#138A	Master CPU: Manual warm restart request
W#16#138B	Master CPU: Automatic warm restart request
W#16#138C	Standby CPU: Request for manual restart
W#16#138D	Standby CPU: Request for automatic restart

Event class 2 - synchronous errors (S7-300, S7-400)

Description of the event IDs

Event ID	Event	OB
W#16#2521	BCD conversion error	OB 121
W#16#2522	Area length error when reading	OB 121
W#16#2523	Area length error when writing	OB 121
W#16#2524	Area error when reading	OB 121
W#16#2525	Area error when writing	OB 121
W#16#2526	Timer number error	OB 121
W#16#2527	Counter number error	OB 121
W#16#2528	Alignment error when reading	OB 121
W#16#2529	Alignment error when writing	OB 121
W#16#2530	Write error when accessing the DB	OB 121
W#16#2531	Write error when accessing the DI	OB 121
W#16#2532	Block number error when opening a DB	OB 121
W#16#2533	Block number error when opening a DI	OB 121
W#16#2534	Block number error when calling an FC	OB 121
W#16#2535	Block number error when calling an FB	OB 121
W#16#253A	DB not loaded	OB 121
W#16#253C	FC not loaded	OB 121
W#16#253D	SFC not loaded	OB 121
W#16#253E	FB not loaded	OB 121
W#16#253F	SFB not loaded	OB 121

Event ID	Event	OB
W#16#2942	I/O access error, reading	OB 122
W#16#2943	I/O access error, writing	OB 122

Event class 3 - synchronous errors (S7-300, S7-400)

Description of the event IDs

Event ID	Event	OB
W#16#3501	Cycle time exceeded	OB 80
W#16#3502	User interface (OB or FRB) request error	OB 80
W#16#3505	Time-of-day interrupt(s) skipped due to new clock setting	OB 80
W#16#3506	Time-of-day interrupt(s) skipped when changing to RUN after HOLD	OB 80
W#16#3507	Multiple OB request errors caused internal buffer overflow	OB 80
W#16#3508	Isochronous mode interrupt - time error	OB 80
W#16#3509	Loss of interrupt due to high interrupt load	OB 80
W#16#350A	Resume RUN after CiR	OB 80
W#16#350B	Technology synchronization interrupt - time error	OB 80
W#16#3921/3821	BATTF: failure on at least one backup battery of the central rack/ problem eliminated Note: the incoming event only occurs if one of the backup battery fails (if there are redundant backup batteries). If the other battery then fails, the event does not occur again.	OB 81
W#16#3922/3822	BAF: failure of backup voltage on central rack/ problem eliminated	OB 81
W#16#3923/3823	24 volt supply failure on central rack / problem eliminated	OB 81
W#16#3925/3825	BATTF: failure on at least one backup battery in a redundant central rack/ problem eliminated	OB 81
W#16#3926/3826	BAF: failure of backup voltage on redundant central rack/ problem eliminated	OB 81
W#16#3927/3827	24 volt supply failure on redundant central rack / problem eliminated	OB 81
W#16#3931/3831	BATTF: failure of at least one backup battery of the expansion rack/ problem eliminated	OB 81
W#16#3932/3832	BAF: failure of backup voltage on expansion rack/ problem eliminated	OB 81

Event ID	Event	OB
W#16#3933/3833	24 volt supply failure on at least one expansion rack/ problem eliminated	OB 81
W#16#3942	Diagnostic error interrupt (module/sub-module fault or maintenance required or both)	OB 82
W#16#3842	Diagnostic error interrupt (module/sub-module OK)	OB 82
W#16#3951	PROFINET IO module removed	OB 83
W#16#3954	PROFINET IO interface sub-module/sub-module removed	OB 83
W#16#3854	PROFINET IO interface sub-module/sub-module and matches the configured sub-module/sub-module	OB 83
W#16#3855	PROFINET IO interface sub-module/sub-module inserted, but does not match the configured sub-module/sub-module	OB 83
W#16#3856	PROFINET IO interface sub-module/sub-module inserted, but error in module parameter assignment	OB 83
W#16#3857	PROFINET IO submodule/module inserted, but with problem or maintenance	OB 83
W#16#3858	PROFINET IO submodule, access error corrected	OB 83
W#16#3861	Module/interface module inserted, module type OK	OB 83
W#16#3961	Module/interface module removed, cannot be addressed	OB 83
W#16#3863	Module/interface module inserted, but wrong module type	OB 83
W#16#3864	Module/interface module inserted, but causing problem (type ID unreadable)	OB 83
W#16#3865	Module inserted, but error in module parameter assignment	OB 83
W#16#3866	Module responds again, load voltage error corrected	OB 83
W#16#3966	Module not responding, load voltage error	OB 83
W#16#3367	Start module parameter reassignment	OB 83
W#16#3267	End of module parameter reassignment	OB 83
W#16#3968	Module parameter reassignment terminated with error	OB 83
W#16#3571	Nesting depth of nesting levels exceeded	OB 88
W#16#3572	Nesting depth of Master Control Relay exceeded	OB 88
W#16#3573	Nesting depth exceeded for synchronous errors	OB 88
W#16#3574	Nesting depth of block calls (U stack) exceeded	OB 88
W#16#3575	Nesting depth of block calls (B stack) exceeded	OB 88
W#16#3576	Error during allocation of local data	OB 88
W#16#3578	Unknown instruction	OB 88
W#16#357A	Call statement with destination outside of the block	OB 88
W#16#3981	Interface error, incoming	-
W#16#3881	Interface error, outgoing	-
W#16#3582	Memory error in operating system detected and corrected	OB 84
W#16#3583	Accumulation of detected and corrected memory errors	OB 84

Event ID	Event	OB
W#16#3585	Error in the PC operating system (only with Win AC controllers)	OB 84
W#16#3986	Performance of an H-Sync link affected	OB 84
W#16#3587	Multi-bit memory error detected and corrected	OB 84
W#16#35A1	User interface (OB or FRB) not found	OB 85
W#16#35A2	OB not loaded (started by SFC or operating system due to configuration)	OB 85
W#16#35A3	Error when operating system accesses a block	OB 85
W#16#35A4	PROFINET Interface DB cannot be addressed	OB 85
W#16#34A4	PROFINET Interface DB can be addressed again	OB 85
W#16#39B1	I/O access error when updating the process image of the inputs	OB 85
W#16#39B2	I/O access error when transferring the process image to the output modules	OB 85
W#16#39B3/38B3	I/O access error when updating the process image of the inputs	OB 85
W#16#39B4/38B4	I/O access error when updating the process image of the outputs	OB 85
W#16#38C1	Expansion rack operational again (1 to 21), outgoing	OB 86
W#16#39C1	Expansion rack failure (1 to 21), incoming	OB 86
W#16#38C2	Expansion rack operational again but mismatch between setpoint and actual configuration	OB 86
W#16#39C3	Distributed I/O: master system failure incoming	OB 86
W#16#39C4	Distributed I/O: Station failure, incoming	OB 86
W#16#38C4	Distributed I/O: Station failure, outgoing	OB 86
W#16#38C5	Distributed I/O: Station return with problem	OB 86
W#16#38C6	Expansion unit return, but error in module parameter assignment	OB 86
W#16#38C7	DP: Station return, but error in module parameter assignment	OB 86
W#16#38C8	DP: Station return, but difference between configured and actual configuration	OB 86
W#16#39CA	PROFINET IO system failure	OB 86
W#16#39CB	PROFINET IO station failure	OB 86
W#16#38CB	PROFINET IO station return	OB 86
W#16#38CC	PROFINET IO station return with problem or maintenance	OB 86
W#16#39CD	PROFINET IO station return, but difference between configured and actual configuration	OB 86
W#16#39CE	PROFINET IO station return, error in module parameter assignment	OB 86
W#16#33CF/32CF	Failure/return of a PROFINET IO device resulting from deactivation/activation with the D_ACT_DP instruction	OB 86

Event ID	Event	OB
W#16#39F8/38F8	Failure/return of some of the submodules of a PROFINET I-device	OB 86
W#16#38F9	Return of some of the submodules of a PROFINET I-device with a device configuration difference	OB 86
W#16#35D2	Diagnostics entries cannot be sent at present	OB 87
W#16#35D3	Synchronization frames cannot be sent	OB 87
W#16#35D4	Illegal time jump resulting from synchronization	OB 87
W#16#35D5	Error adopting the synchronization time	OB 87
W#16#35E1	Incorrect message frame ID in GD	OB 87
W#16#35E2	GD packet status cannot be entered in DB	OB 87
W#16#35E3	Frame length error in GD	OB 87
W#16#35E4	Illegal GD packet number received.	OB 87
W#16#35E5	Error while accessing DB in communication instructions for configured S7 connections	OB 87
W#16#35E6	GD total status cannot be entered in DB	OB 87

Event class 4 - stop events and other mode changes (S7-300, S7-400)

Description of the event IDs

Event ID	Event
W#16#4300	Backed-up power on
W#16#4301	Mode transition from STOP to STARTUP
W#16#4302	Mode transition from STARTUP to RUN
W#16#4303	STOP caused by stop switch being activated
W#16#4304	STOP caused by STOP operation on programming device or because of STOP instruction
W#16#4305	HOLD: breakpoint reached
W#16#4306	HOLD: breakpoint exited
W#16#4307	Memory reset started by PG operation
W#16#4308	Memory reset started by switch setting
W#16#4309	Memory reset started automatically (power on not backed up)
W#16#430A	HOLD exited, transition to STOP
W#16#430D	STOP caused by other CPU in multicomputing
W#16#430E	Memory reset executed
W#16#430F	STOP on the module due to STOP on a CPU
W#16#4510	STOP violation of the CPU's data range
W#16#4318	Start of CiR
W#16#4319	CiR completed
W#16#4520	DEFECT: STOP not possible
W#16#4521	DEFECT: failure of instruction processing processor
W#16#4522	DEFECT: failure of clock chip

Event ID	Event
W#16#4523	DEFECT: failure of clock pulse generator
W#16#4524	DEFECT: failure of timer update function
W#16#4525	DEFECT: failure of multicomputing synchronization
W#16#4926	DEFECT: failure of the watchdog for I/O access
W#16#4527	DEFECT: failure of I/O access monitoring
W#16#4528	DEFECT: failure of scan time monitoring
W#16#4530	DEFECT: memory test error in internal memory
W#16#4931	STOP or DEFECT: memory test error in memory sub-module
W#16#4532	DEFECT: failure of core resources
W#16#4933	Checksum error
W#16#4934	DEFECT: memory not available
W#16#4935	DEFECT: cancelled by watchdog/processor exceptions
W#16#4536	DEFECT: mode switch defective
W#16#4540	STOP: memory expansion of the internal work memory is not seamless. First memory expansion too small or missing.
W#16#4541	STOP caused by priority class sequence control system
W#16#4542	STOP caused by object management system
W#16#4543	STOP caused by test functions
W#16#4544	STOP caused by diagnostics system
W#16#4545	STOP caused by communication system
W#16#4546	STOP caused by CPU memory management
W#16#4547	STOP caused by process image management
W#16#4548	STOP caused by I/O management
W#16#4949	STOP caused by continuous hardware interrupt
W#16#454A	STOP caused by configuration: An OB deselected per configuration was loaded into the CPU during STARTUP
W#16#494D	STOP caused by I/O error
W#16#494E	STOP caused by power failure
W#16#494F	STOP caused by configuration error
W#16#4550	DEFECT: internal system error
W#16#4555	No hot restart possible, monitoring time elapsed
W#16#4556	STOP: memory reset request from communication system / due to data inconsistency
W#16#4357	Module watchdog started
W#16#4358	All modules are ready for operation
W#16#4959	One or more modules not ready for operation
W#16#4562	STOP caused by programming error (OB not loaded or not possible)
W#16#4563	STOP caused by I/O access error (OB not loaded or not possible)
W#16#4567	STOP caused by H-event
W#16#4568	STOP caused by time error (OB not loaded or not possible)
W#16#456A	STOP caused by diagnostic error interrupt (OB not loaded or not possible)
W#16#456B	STOP caused by removing/inserting module (OB not loaded or not possible)

Event ID	Event
W#16#456C	STOP caused by CPU hardware error (OB not loaded or not possible, or no FRB)
W#16#456D	STOP caused by program sequence error (OB not loaded or not possible)
W#16#456E	STOP caused by communication error (OB not loaded or not possible)
W#16#456F	STOP caused by rack failure OB (OB not loaded or not possible)
W#16#4570	STOP caused by hardware interrupt (OB not loaded or not possible)
W#16#4571	STOP caused by nesting stack error
W#16#4572	STOP caused by master control relay stack error
W#16#4573	STOP caused by exceeding the nesting depth for synchronous errors
W#16#4574	STOP caused by exceeding interrupt stack nesting depth in the priority class stack
W#16#4575	STOP caused by exceeding block stack nesting depth in the priority class stack
W#16#4576	STOP caused by error when allocating the local data
W#16#4578	STOP caused by unknown opcode
W#16#457A	STOP caused by code length error
W#16#457B	STOP caused by DB not being loaded on on-board I/O
W#16#497C	STOP caused by integrated technology
W#16#457D	Reset/clear request because the version of the internal interface to the integrated technology was changed.
W#16#457F	STOP caused by STOP command
W#16#4580	STOP: back-up buffer contents inconsistent (no transition to RUN)
W#16#4590	STOP caused by overloading the internal functions
W#16#49A0	STOP caused by parameter assignment error or non-permitted difference between configured and actual configuration: Start-up disabled.
W#16#49A1	STOP caused by parameter assignment error: memory reset request
W#16#49A2	STOP caused by error in parameter modification: Start-up disabled.
W#16#49A3	STOP caused by error in parameter modification: memory reset request
W#16#49A4	STOP: inconsistency in configuration data
W#16#49A5	STOP: Distributed I/O: inconsistency in the loaded configuration information
W#16#49A6	STOP: Distributed I/O: invalid configuration information
W#16#49A7	STOP: Distributed I/O: no configuration information
W#16#49A8	STOP: error indicated by the interface module for the distributed I/O
W#16#43B0	Firmware update / backup was successful
W#16#49B1	Firmware update data incorrect
W#16#49B2	Firmware update: hardware version does not match firmware
W#16#49B3	Firmware update: module type does not match firmware
W#16#43B4	Error in firmware backup
W#16#43B6	Firmware updates canceled by redundant modules
W#16#43D0	LINK-UP aborted due to violation of coordination rules
W#16#43D1	LINK-UP/UPDATE sequence aborted

Event ID	Event
W#16#49D2	Standby CPU changed to STOP due to STOP on the master CPU during link-up
W#16#43D3	STOP on standby CPU
W#16#49D4	STOP on a master, since partner CPU is also a master (link-up error)
W#16#43D5	LINK-UP rejected due to mismatched CPU memory configuration of the sub-PLC
W#16#43D6	LINK-UP rejected due to mismatched system program of the sub-PLC
W#16#43D7	LINK-UP rejected due to change in user program or in configuration
W#16#49D8	STOP/trouble shooting mode/DEFECT: hardware fault detected due to other error
W#16#49D9	STOP due to SYNC module error
W#16#49DA	STOP due to synchronization error between H-CPUs
W#16#43DC	Abort during link-up with switchover
W#16#43DD	LINK-UP rejected due to running test or other online functions
W#16#43DE	Updating aborted due to monitoring time being exceeded during the n-th attempt, new update attempt initiated
W#16#43DF	Updating aborted for the last time due to monitoring time being exceeded after completing the maximum amount of attempts. User intervention required.
W#16#43E0	Change from solo mode after link-up
W#16#43E1	Change from link-up after updating
W#16#43E2	Change from updating to redundant mode
W#16#43E3	Master CPU: change from redundant mode to solo mode
W#16#43E4	Standby CPU: change from redundant mode after error-search mode
W#16#43E5	Standby CPU: change from error-search mode after link-up or STOP
W#16#43E6	Link-up aborted on the standby CPU
W#16#43E7	Updating aborted on the standby CPU
W#16#43E8	Standby CPU: change from link-up after startup
W#16#43E9	Standby CPU: change from startup after updating
W#16#43F1	Reserve-master switchover
W#16#43F2	Coupling of incompatible H-CPUs blocked by system program
W#16#42F3	Checksum error detected and corrected by the operating system
W#16#43F4	Standby CPU: Connecting/updating disabled by means of the H_CTRL instruction in the master CPU

Event class 5 - status runtime events (S7-300, S7-400)

Description of the event IDs

Event ID	Event
W#16#530D	New startup information in STOP mode
W#16#510F	A problem has occurred with WinLC. This problem caused the CPU to go into STOP mode or has caused a fault in the CPU.
W#16#5311	Startup despite missing completed message from module(s)
W#16#5545	Start of parameter reassignment as part of a plant modification during operation
W#16#5445	End of parameter reassignment as part of a plant modification during operation

Event ID	Event
W#16#5961	Parameter assignment error
W#16#5962	Parameter assignment error with startup blocked
W#16#5963	Parameter assignment error with memory reset request
W#16#5966	Parameter assignment error when switching
W#16#5967	Parameter assignment error: Unknown configuration data
W#16#5968/5868	Emergency address of an Ethernet interface activated/deactivated
W#16#5969	Parameter assignment error with startup blocked
W#16#596A	PROFINET IO: IP address of an IO device already present
W#16#596B	IP address of an existing Ethernet interface
W#16#596C	Name of an existing Ethernet interface
W#16#596D	The existing network configuration does not mach the system requirements or configuration.
W#16#5371	Distributed I/O: end of the synchronization with a DP master
W#16#5979/5879	Diagnostics alarm from DP interface: EXTF LED on/off
W#16#597C	DP Global Control command failed or moved
W#16#5380	Diagnostics buffer entries from interrupt and asynchronous error events disabled
W#16#5581	One or more licenses for runtime software are missing.
W#16#5481	All licenses for runtime software are complete again.
W#16#558A	Difference between the Machine-Readable Product Code of the configured and inserted CPU
W#16#558B	Difference between the firmware version of the configured and inserted CPU
W#16#5966	Parameter assignment error when switching
W#16#597C	DP Global Control command failed or moved
W#16#5395	Distributed I/O: Reset of a DP master
W#16#5598	Start of potential inconsistency with DP master systems due to CiR
W#16#5498	End of potential inconsistency with DP master systems due to CiR
W#16#59A0	Interrupt in the CPU cannot be assigned
W#16#59A1	Configuration error in the integrated technology
W#16#53A2	Download of technology firmware successful
W#16#59A3	Error when downloading the integrated technology
W#16#53A4	Download of technology DB not successful
W#16#55A5	Version conflict: internal interface with integrated technology
W#16#55A6	The maximum number of technology objects has been exceeded.
W#16#55A7	A technology DB of this type is already present.
W#16#53FF	Reset to factory setting

Event class 6 - communication events (S7-300, S7-400)

Description of the event IDs

Event ID	Event
W#16#6500	Connection ID exists twice on module
W#16#6501	Connection resources inadequate
W#16#6502	Error in the connection description
W#16#6905/6805	Resource problem with permanently configured connections/corrected
W#16#6510	CFB structure error detected in instance DB when evaluating EPROM
W#16#6514	GD packet number exists twice on the module
W#16#6515	Inconsistent length specifications in GD configuration information
W#16#6316	Interface error at power up of the AS
W#16#6521	No memory sub-module and no internal memory available
W#16#6522	Illegal module: replace sub-module and reset memory
W#16#6523	Memory reset request due to error accessing module
W#16#6524	Memory reset request due to error in block header
W#16#6526	Memory reset request due to memory replacement
W#16#6527	Memory replaced, therefore hot restart not possible
W#16#6528	Object handling function in STOP/HOLD mode, no hot restart possible
W#16#6529	No startup possible during the "load user program" function
W#16#652A	No startup because block exists twice in user memory
W#16#652B	No startup because block is too long for sub-module - replace sub-module Module replacement necessary
W#16#652C	No startup due to illegal OB on sub-module
W#16#6532	No startup because illegal configuration information on sub-module
W#16#6533	Memory reset request because of invalid sub-module content
W#16#6534	No startup: block exists more than once on module
W#16#6535	No startup: not enough memory to transfer block from module
W#16#6536	No startup: module contains an illegal block number
W#16#6537	No startup: module contains a block with an illegal length
W#16#6538	Local data or write-protection ID (for DB) of a block illegal for CPU
W#16#6539	Illegal command in block (detected by compiler)
W#16#653A	Memory reset request because local OB data on module too short
W#16#6543	No startup: illegal block type
W#16#6544	No startup: attribute "relevant for processing" illegal
W#16#6545	Source language illegal
W#16#6546	Maximum number of blocks of one block type reached
W#16#6547	Parameter assignment error when assigning parameters to modules (not on I/O bus, cancel download)
W#16#6548	Plausibility error during block check
W#16#6549	Structure error in block
W#16#6550	A block has an error in the CRC
W#16#6551	A block has no CRC

Event ID	Event
W#16#6353	Firmware update: Start of firmware download over the network
W#16#6253	Firmware update: End of firmware download over the network
W#16#6560	SCAN overflow
W#16#6981	Interface error, incoming
W#16#6881	Interface error, outgoing
W#16#6390	Formatting of a Micro Memory Card executed

Event class 8 - diagnostic events for modules (S7-300, S7-400)

Description of the event IDs

Event ID	Event	Module type
W#16#8x00	Module fault/OK	Any
W#16#8x01	Internal error	
W#16#8x02	External error	
W#16#8x03	Channel fault	
W#16#8x04	External auxiliary voltage missing	
W#16#8x05	Front panel connector not plugged in	
W#16#8x06	No parameter assignment	
W#16#8x07	Incorrect parameters on module	
W#16#8x30	User module is missing or has an error	
W#16#8x31	Communication problem	
W#16#8x32	RUN/STOP mode (STOP: incoming, RUN: outgoing)	
W#16#8x33	Time monitoring responded (watchdog)	
W#16#8x34	Internal power supply failed	
W#16#8x35	BATTF: battery exhausted	
W#16#8x36	Entire backup failed	
W#16#8x40	Expansion rack failure	
W#16#8x41	Processor failure	
W#16#8x42	EPROM fault	
W#16#8x43	RAM fault	
W#16#8x44	ADC/DAC error	
W#16#8x45	Fuse tripped	
W#16#8x46	Hardware interrupt lost	
W#16#8x50	Configuration/parameter assignment error	Analog input
W#16#8x51	Common mode error	
W#16#8x52	P short circuit	
W#16#8x53	M short circuit	
W#16#8x54	Wire break	
W#16#8x55	Reference channel error	
W#16#8x56	Measuring range undershoot	
W#16#8x57	Measuring range exceeded	

Event ID	Event	Module type
W#16#8x60	Configuration/parameter assignment error	Analog output
W#16#8x61	Common mode error	
W#16#8x62	P short circuit	
W#16#8x63	M short circuit	
W#16#8x64	Wire break	
W#16#8x66	No load voltage	
W#16#8x70	Configuration/parameter assignment error	Digital input
W#16#8x71	Ground fault	
W#16#8x72	P short circuit (sensor)	
W#16#8x73	M short circuit	
W#16#8x74	Wire break	
W#16#8x75	No sensor power supply	
W#16#8x80	Configuration/parameter assignment error	Digital output
W#16#8x81	Ground fault	
W#16#8x82	P short circuit	
W#16#8x83	M short circuit	
W#16#8x84	Wire break	
W#16#8x85	Fuse tripped	
W#16#8x86	No load voltage	
W#16#8x87	excessive temperature	
W#16#8xB0	FM-count: Signal A defective	FM
W#16#8xB1	FM-count: Signal B defective	
W#16#8xB2	FM-count: Signal N defective	
W#16#8xB3	FM-count: Transfer defective value between the channels	
W#16#8xB4	FM-count: Sensor supply 5.2 V defective	
W#16#8xB5	FM-count: Sensor supply 24 V defective	

Event class 9 - standard user events (S7-300, S7-400)

Description of the event IDs

Event ID	Event
W#16#9001	Automatic mode
W#16#9101	Manual mode
W#16#9x02	OPEN/CLOSED, ON/OFF
W#16#9x03	Manual command enable
W#16#9x04	Unit protective command (OPEN/CLOSED)
W#16#9x05	Process enable
W#16#9x06	System protection command
W#16#9x07	Process value monitoring responded
W#16#9x08	Manipulated variable monitoring responded

Event ID	Event
W#16#9x09	System deviation greater than permitted
W#16#9x0A	Limit position error
W#16#9x0B	Runtime error
W#16#9x0C	Command execution error (sequence control)
W#16#9x0D	Operating status running > OPEN
W#16#9x0E	Operating status running > CLOSED
W#16#9x0F	Command blocking
W#16#9x11	Process status OPEN/ON
W#16#9x12	Process status CLOSED/OFF
W#16#9x13	Process status intermediate position
W#16#9x14	Process status ON via AUTO
W#16#9x15	Process status ON via manual
W#16#9x16	Process status ON via protective command
W#16#9x17	Process status OFF via AUTO
W#16#9x18	Process status OFF via manual
W#16#9x19	Process status OFF via protective command
W#16#9x21	Function error on approach
W#16#9x22	Function error on return
W#16#9x31	Actuator (DE/WE) limit position OPEN
W#16#9x32	Actuator (DE/WE) limit position not OPEN
W#16#9x33	Actuator (DE/WE) limit position CLOSED
W#16#9x34	Actuator (DE/WE) limit position not CLOSED
W#16#9x41	Illegal status, tolerance time elapsed
W#16#9x42	Illegal status, tolerance time not elapsed
W#16#9x43	Interlock error, tolerance time = 0
W#16#9x44	Interlock error, tolerance time > 0
W#16#9x45	No reaction
W#16#9x46	Final status exited illegally, tolerance time = 0
W#16#9x47	Final status exited illegally, tolerance time > 0
W#16#9x50	Upper limit of signal range USR
W#16#9x51	Upper limit of measuring range UMR
W#16#9x52	Lower limit of signal range LSR
W#16#9x53	Lower limit of measuring range LMR
W#16#9x54	Upper alarm limit UAL
W#16#9x55	Upper warning limit UWL
W#16#9x56	Upper tolerance limit UTL
W#16#9x57	Lower tolerance limit LTL
W#16#9x58	Lower warning limit LWL
W#16#9x59	Lower alarm limit LAL
W#16#9x60	GRAPH 7 step entering/leaving
W#16#9x61	GRAPH 7 interlock error
W#16#9x62	GRAPH 7 execution error
W#16#9x63	GRAPH 7 error noted

Event ID	Event
W#16#9x64	GRAPH 7 error acknowledged
W#16#9x70	Trend exceeded in positive direction
W#16#9x71	Trend exceeded in negative direction
W#16#9x72	No reaction
W#16#9x73	Final state exited illegally
W#16#9x80	Limit value exceeded, tolerance time = 0
W#16#9x81	Limit value exceeded, tolerance time > 0
W#16#9x82	Below limit value, tolerance time = 0
W#16#9x83	Below limit value, tolerance time > 0
W#16#9x84	Gradient exceeded, tolerance time = 0
W#16#9x85	Gradient exceeded, tolerance time > 0
W#16#9x86	Below gradient, tolerance time = 0
W#16#9x87	Below gradient, tolerance time > 0
W#16#9190/9090	User parameter assignment error incoming/outgoing
W#16#91F0	Overflow
W#16#91F1	Underflow
W#16#91F2	Division by 0
W#16#91F3	Illegal calculation operation

Event classes A and B - free user events (S7-300, S7-400)

Description of the event IDs

Event ID	Event
W#16#Axyz	Events available for user
W#16#Bxyz	Events available for user

Reserved event classes (S7-300, S7-400)

Reserved

The following event classes are reserved for later expansions:

- C
- D
- E
- F reserved for modules not in central rack (e.g., CPs or FMs)

Symbolic and numerical names of instructions (S7-300, S7-400)

Description

The instructions from the task card are comprised of functions (FC), function blocks (FB), system functions (SFC) and system function blocks (SFB) that are identified internally by numbers.

The following tables show the assignment of numerical and symbolic names.

Function blocks (FBs)

Numerical name	Symbolic name
FB 2	S_RCV
FB 3	S_SEND
FB 4	S_VSTAT
FB 5	S_VSET
FB 6	S_XON
FB 7	S_RTS
FB 8	S_V24
FB 8 (1)	USEND_S
FB 9 (1)	URCV_S
FB 14 (1)	GET_S
FB 15 (1)	PUT_S
FB 40 (1)	FTP_CMD
FB 46	PG_DIAL
FB 47	AS_DIAL
FB 48	SMS_SEND
FB 49	AS_MAIL
FB 52 (1)	PNIO_RW_REC
FB 53	PE_DS3_Write_ET200S
FB 54 (1)	PNIO_ALARM
FB 55	IP_CONFIG
FB 56 (1)	LOGICAL_TRIGGER
FB 60	SET_SW
FB 61	SET_SW_S
FB 62	TIMESTMP
FB 80	LEAD_LAG
FB 81	DCAT
FB 81	S_MODB
FB 82	MCAT
FB 83	IMC
FB 84	SMC
FB 85	DRUM
FB 86	PACK

Numerical name	Symbolic name
FB 210	FW_TCP
FB 220	FW_IOT
FB 450 (2)	RED_IN_MGP
FB 451 (2)	RED_OUT_MGP
FB 452 (2)	RED_DIAG_MGP
FB 453 (2)	RED_STATUS_MGP
FB 815	PE_Start_End_Pause
FB 816	PE_CMD
FB 817 (1)	PE_I_DEV
(1) for S7-300 (2) for S7-400 (3) not for WinAC (4) for S7-400, not for WinAC (5) for WinAC	

Functions (FCs)

Numerical name	Symbolic name
FC 0 (1)	PE_Error_RSP
FC 1 (1)	PE_Start_RSP
FC 1 (1)	DP_SEND
FC 2	CONCAT
FC 2 (1)	PE_End_RSP
FC 2 (1)	DP_RECV
FC 3 (1)	PE_List_Modes_RSP
FC 3 (1)	DP_DIAG
FC 4	DELETE
FC 4 (1)	PE_Get_Mode_RSP
FC 4 (1)	DP_CTRL
FC 5 (1)	PE_PEM_Status_RSP
FC 5 (1)	AG_SEND
FC 6 (1)	PE_Identify_RSP
FC 6 (1)	AG_RECV
FC 7	Asi_3422
FC 7 (1)	PE_Measurement_List_RSP
FC 7 (1)	AG_LOCK
FC 8 (1)	PE_Measurement_Value_RSP
FC 8 (1)	AG_UNLOCK
FC 10 (1)	AG_CNTRL
FC 11	FIND
FC 11 (1)	PNIO_SEND
FC 12 (1)	PNIO_RECV

Numerical name	Symbolic name
FC 17	INSERT
FC 17	S_USST
FC 18	S_USSR
FC 19	S_USSI
FC 20	LEFT
FC 21	LEN
FC 22	LIMIT
FC 25	MAX
FC 26	MID
FC 27	MIN
FC 31	REPLACE
FC 32	RIGHT
FC 36	ENCO
FC 36	SEL
FC 37	DECO
FC 60	LOC_TIME
FC 61	BT_LT
FC 62	LT_BT
FC 62 (1)	C_CNTRL
FC 63	S_LTINT
FC 74 (3)	I_ABORT
FC 80	TONR
FC 82	RSET
FC 83	SET
FC 84	ATT
FC 85	FIFO
FC 86	TBL_FIND
FC 87	LIFO
FC 88	TBL
FC 89	TBL_WRD
FC 90	WSR
FC 91	WRD_TBL
FC 92	SHRB
FC 93	SEG
FC 98	BCDPL
FC 99	BITSUM
FC 100	RSETI
FC 101	SETI
FC 102	DEV
FC 103	CDT
FC 104	TBL_TBL
FC 105	SCALE
FC 106	UNSCALE

Numerical name	Symbolic name
FC 450 (2)	RED_INIT_MGP
FC 451 (2)	RED_DEPA_MGP
(1) for S7-300 (2) for S7-400 (3) not for WinAC (4) for S7-400, not for WinAC (5) for WinAC	

System function blocks (SFBs)

Numerical name	Symbolic name
SFB 0	CTU
SFB 1	CTD
SFB 2	CTUD
SFB 3	TP
SFB 4	TON
SFB 5	TOF
SFB 16 (4)	PRINT
SFB 19 (2)	START
SFB 20 (2)	STOP
SFB 21 (4)	RESUME
SFB 22 (2)	STATUS
SFB 23 (2)	USTATUS
SFB 29 (1)	HS_COUNT_300C
SFB 30 (1)	FREQ_MES_300C
SFB 31 (2)	NOTIFY_8P
SFB 32	DRUM_X
SFB 33 (2)	ALARM
SFB 34 (2)	ALARM_8
SFB 35 (2)	ALARM_8P
SFB 36 (2)	NOTIFY_8P
SFB 37 (2)	AR_SEND
SFB 38 (1)	HSC_A_B_300C
SFB 39 (1)	POS_300C
SFB 44 (1)	ANALOG_300C
SFB 46 (1)	DIGITAL_300C
SFB 47 (1)	COUNT_300C
SFB 48 (1)	FREQUENC_300C
SFB 49 (1)	PULSE_300C
SFB 52	RDREC
SFB 53	WRREC
SFB 54	RALRM

Numerical name	Symbolic name
SFB 60 (1)	SEND_PTP_300C
SFB 61 (1)	RCV_PTP_300C
SFB 62 (1)	RES_RCVB_300C
SFB 63 (1)	SEND_RK_300C
SFB 64 (1)	FETCH_RK_300C
SFB 65 (1)	SERVE_RK_300C
SFB 73	RCVREC
SFB 74	PRVREC
SFB 75	SALRM
SFB 81	RD_DPAR
SFB 104	IP_CONF
SFB 65001 (5)	CREA_COM
SFB 65002 (5)	EXEC_COM
SFB 65002 (5)	ASYN_COM
(1) for S7-300 (2) for S7-400 (3) not for WinAC (4) for S7-400, not for WinAC (5) for WinAC	

System functions (SFCs)

Numerical name	Symbolic name
SFC 2	SET_RTM
SFC 3	CTRL_RTM
SFC 4	READ_RTM
SFC 5	GADR_LGC
SFC 7 (3)	DP_PRAL
SFC 9 (2)	EN_MSG
SFC 10 (2)	DIS_MSG
SFC 11	DPSYC_FR
SFC 12	D_ACT_DP
SFC 13	DPNRM_DG
SFC 14	DPRD_DAT
SFC 17	ALARM_SQ
SFC 18	ALARM_S
SFC 19	ALARM_SC
SFC 20	BLKMOV
SFC 21	FILL
SFC 22	CREAT_DB
SFC 23	DEL_DB
SFC 24	TEST_DB

Numerical name	Symbolic name
SFC 26 (2)	UPDAT_PI
SFC 27 (2)	UPDAT_PO
SFC 28	SET_TINT
SFC 29	CAN_TINT
SFC 30	ACT_TINT
SFC 31	QRY_TINT
SFC 32	SRT_DINT
SFC 33	CAN_DINT
SFC 34	QRY_DINT
SFC 35 (4)	MP_ALM
SFC 36	MSK_FLT
SFC 37	DMSK_FLT
SFC 38	READ_ERR
SFC 39	DIS_IRT
SFC 40	EN_IRT
SFC 41	DIS_AIRT
SFC 42	EN_AIRT
SFC 43	RE_TRIGR
SFC 44	REPL_VAL
SFC 46	STP
SFC 47	WAIT
SFC 48 (4)	SNC_RTCB
SFC 49	LGC_GADR
SFC 50	RD_LGADR
SFC 51	RDSYSST
SFC 52	WR_USMSG
SFC 54 (2)	RD_DPARM
SFC 55	WR_PARM
SFC 56	WR_DPARM
SFC 57	PARM_MOD
SFC 58	WR_REC
SFC 59	RD_REC
SFC 60 (4)	GD_SND
SFC 61 (4)	GD_RCV
SFC 62 (2)	CONTROL
SFC 64	TIME_TCK
SFC 64	RD_SINFO
SFC 65 (3)	X_SEND
SFC 66 (3)	X_RCV
SFC 67 (3)	X_GET
SFC 68 (3)	X_PUT
SFC 69 (3)	X_ABORT
SFC 70	GEO_LOG

Numerical name	Symbolic name
SFC 71	LOG_GEO
SFC 72 (3)	I_GET
SFC 73 (3)	I_PUT
SFC 75	SET_ADDR
SFC 78 (2)	OB_RT
SFC 79	SETP
SFC 80	RESETP
SFC 81	UBLKMOV
SFC 82 (1) (6)	CREA_DBL
SFC 83 (1) (6)	READ_DBL
SFC 84 (1) (6)	WRIT_DBL
SFC 85 (6)	CREA_DB
SFC 87 (2)	C_DIAG
SFC 90 (4)	H_CTRL
SFC 99	WWW
SFC 100 (2)	SET_CLKS
SFC 101	RTM
SFC 102 (1)	RD_DPARA
SFC 103 (3)	DP_TOPOL
SFC 104 (2)	CIR
SFC 105	READ_SI
SFC 106	DEL_SI
SFC 107	ALARM_DQ
SFC 108	ALARM_D
SFC 109	PROTECT
SFC 112	PN_IN
SFC 113	PN_OUT
SFC 114	PN_DP
SFC 126	SYNC_PI
SFC 127	SYNC_PO
SFC 154	DPWR_DAT

(1) for S7-300
(2) for S7-400
(3) not for WinAC
(4) for S7-400, not for WinAC
(5) for WinAC

1.4.1.2 Mode of operation of S7-1500 CPUs

Operating modes

Basics of the operating modes of S7-CPUs

Introduction

Operating modes describe the states of the CPU. The following operating modes are possible:

- STARTUP
- RUN
- STOP
- HOLD

In these operating modes, the CPU can communicate via the PN/IE interface, for example.

Other operating modes

If the CPU is not ready for operation, it is in one of following two operating modes:

- Off-voltage, in other words, line power is switched off.
- Defective, which means that an internal error has occurred.
  If the "Defective" status is caused by a firmware error, this state is indicated by the status LEDs of the CPU (refer to the description of the CPU in the manuals). Proceed as follows to identify the cause:
  - Turn the power supply switch off and on again.
  - Read out the diagnostics buffer when the CPU starts up and send the data for analysis to Customer Support.

  If the CPU does not start up, replace it.

Operating mode transitions

Overview

The following figure shows the operating modes and the operating mode transitions of S7-1500 CPUs:

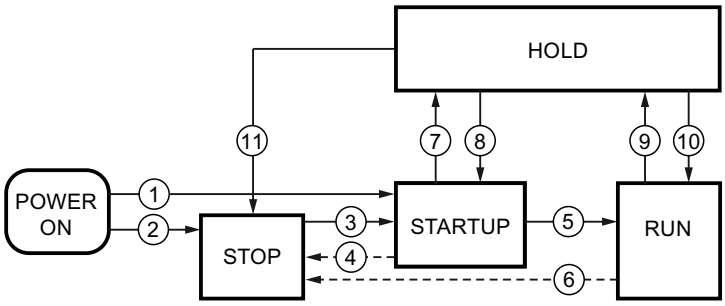

The following table shows the conditions under which the operating modes will change:

No.	Operating mode transition	Conditions
①	POWER ON → STARTUP	After switching on, the CPU goes to "STARTUP" mode if: • "Warm restart" startup type is set, and • the hardware configuration and the program blocks are consistent. Non-retentive memory is cleared and the contents of non-retentive DBs are reset to the initial values of the load memory. Retentive memory and retentive DB contents are retained.
②	POWER ON → STOP	When startup type "No startup" is set, the CPU goes to "STOP" mode after the supply voltage is switched on. Non-retentive memory is cleared and the contents of non-retentive DBs are reset to the initial values of the load memory. Retentive memory and retentive DB contents are retained.
③	STOP → STARTUP	The CPU switches to "STARTUP" mode if: • CPU is set to "RUN" from the programming device, and • the hardware configuration and the program blocks are consistent.
④	STARTUP → STOP	The CPU returns to the "STOP" mode in the following situations: • Error detected during startup. • The CPU is set to "STOP" from the programming device. • A STOP command is processed in the STARTUP OB.
⑤	STARTUP → RUN	If the STARTUP is successful, the CPU switches to "RUN".
⑥	RUN → STOP	The CPU returns to the "STOP" mode in the following situations: • An error is detected that prevents continued processing. • The CPU is set to "STOP" from the programming device. • A STOP command is processed in the user program.
⑦	STARTUP → HOLD	Breakpoint in the start-up routine reached.
⑧	HOLD → STARTUP	Breakpoint in the start-up routine exited by setting to "RUN" from the programming device.
⑨	RUN → HOLD	Breakpoint reached.

No.	Operating mode transition	Conditions
⑩	HOLD → RUN	Breakpoint exited by setting to "RUN" from the programming device.
⑪	HOLD → STOP	Operation switch/display or setting the programming device to "STOP".

See also

"RUN" operating mode (Page 905)

"STOP" operating mode (Page 905)

"HOLD" operating mode (Page 906)

"STARTUP" operating mode

Basics of "STARTUP" operating mode

Function

After turning on the CPU, it executes a startup program before starting to execute the cyclic user program.

By suitably programming startup OBs, you can specify certain initialization variables for your cyclic program in the startup program. There is no rule in terms of the number of startup OBs. That is, you can set up one or several startup OBs in your program or none at all.

Parameter settings for startup characteristics

You can specify whether the CPU remains in "STOP" mode or whether a warm restart is performed. Over and above this, you can set the response during startup ("RUN" or previous mode) in the "Startup" group of the CPU properties.

Special characteristics

Note the following points regarding the "STARTUP" mode:

- The outputs on the modules are disabled.
- The process image is initialized.
- The process image is not updated.
  - In order to read the current state from inputs during "STARTUP", you can access inputs via direct I/O access.
  - In order to initialize outputs during STARTUP, values can be written via the process image or via direct I/O access. The values are output at the outputs during the transition to the "RUN" mode.
- The non-retentive bit memories, timers and counters are initialized.
- The non-retentive tags in data blocks are initialized.

- During startup, no cycle time monitoring is running yet.
- The startup OBs are executed. Regardless of the selected type of startup, all programmed startup OBs are executed.
  The following OBs can be started during startup, if a corresponding event occurs:
  - OB 82: Diagnostic interrupt
  - OB 83: Removal/insertion of modules
  - OB 86: Rack error
  - OB 121: Programming error (only for global error handling)
  - OB 122: I/O access error (only for global error handling)

  All other OBs cannot be started until the transition into the "RUN" operating mode.

Warm restart

Function

During a warm restart, all non-retentive bit memory is deleted and non-retentive DB contents are reset to the initial values from load memory. Retentive bit memory and retentive DB contents are retained.

Program execution begins at the call of the first startup OB.

Triggering a warm restart

You can trigger a "Warm restart" using a corresponding menu command on your programming device in the following situations:

- The CPU must be in "STOP" mode.
- After a memory reset
- After downloading a consistent program and a consistent hardware configuration in the "STOP" mode of the CPU.

"POWER ON" triggers a "warm restart" if you have set the following parameters for the startup response:

- Startup type "warm restart - RUN" (regardless of the CPU operating mode prior to POWER OFF).
- "Warm restart - mode prior to POWER OFF" (depending on the CPU operating mode prior to POWER OFF. The CPU must have been in RUN mode prior to this.)

STARTUP activities

Overview

The following table shows which activities the CPU performs at STARTUP:

Activities in execution sequence	At warm restart
Clear non-retentive bit memories	yes
Clear all bit memories	no
Clear the process image output	yes
Processing startup OBs	yes
Update the process image input	yes
Enable outputs after changing to "RUN" mode	yes

Sequence

The following figure shows the activities of the CPU in "STOP", "STARTUP", and "RUN" operating modes.

You can use the following measures to specify the state of the I/O outputs in the first cycle of the user program:

- Use assignable output modules to be able to output substitute values or to retain the last value.
- Set default values for outputs in startup OBs.

During the startup, all interrupt events are entered in a queue so that they can be processed later during RUN mode. In RUN mode, hardware interrupts can be processed at any time.

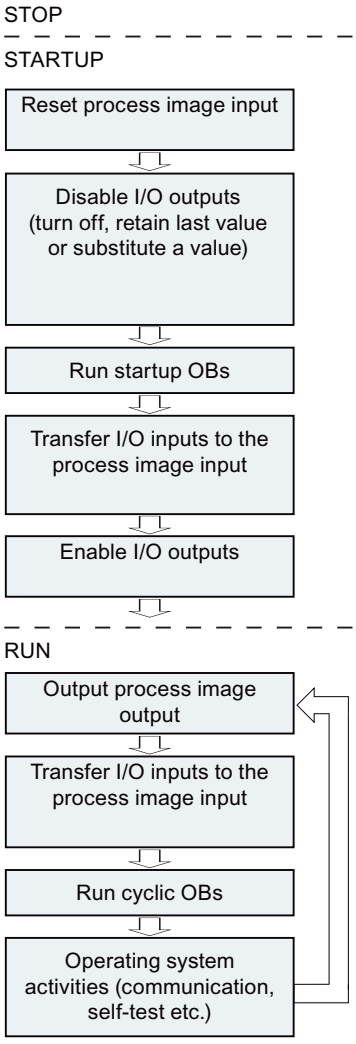

Special features during startup

Response when expected and actual configurations do not match

The expected configuration is represented by the engineering configuration loaded on the CPU. The actual configuration is the actual configuration of the automation system.

If the preset configuration and actual configuration deviate from one another, the CPU's characteristics are specified by the "Supported hardware compatibility" parameter.

Canceling a STARTUP

If errors occur during startup, the startup is canceled and the CPU remains in "STOP" mode.

Under the following conditions, a startup is not performed or canceled:

- If no SIMATIC memory card is inserted or an invalid SIMATIC memory card is inserted.
- If no hardware configuration has been downloaded.

"RUN" operating mode

Function

In "RUN" mode the cyclic, time-driven and interrupt-driven program sections execute. Addresses that are in the "Automatic Update" process image are automatically updated in each program cycle.

- The process image output is read out.
- The process image input table is read.
- The user program is executed.

Active data exchange between S7-1200 CPUs by means of Open User Communication is only possible in "RUN" mode.

Running the user program

Once the CPU has read the inputs, the cyclic program runs from the first to the last instruction.

If you have configured a minimum cycle time, the CPU does not end the cycle until this minimum cycle time is up even if the user program is completed sooner.

A maximum cycle time is set which you can adjust according to your requirements. This ensures that the cyclic program is completed within a specified time. The system responds with a time error if the cyclic program is not completed within this time.

Other events such as hardware and diagnostic interrupts can interrupt the cyclic program flow and prolong the cycle time.

See also

Operating mode transitions (Page 900)

"STOP" operating mode

Function

In "STOP" mode, the user program is not executed. All outputs are disabled or react according to the parameter settings: They provide a substitute value as set in the parameters or retain the last value output and bring the controlled process to a safe status.

The CPU checks the following points:

- Whether the default settings for the CPU are applicable or parameter sets are present
- Whether the general conditions for the programmed startup characteristics are correct

See also

Operating mode transitions (Page 900)

"HOLD" operating mode

Function

The "HOLD" operating mode is reached through a breakpoint in a block that you are testing in single step mode. The single step mode test function is available in the STL and SCL programming languages.

In "HOLD" operating mode, events are not initiated and the user program is not executed.

All outputs are disabled or react according to the parameter settings: Outputs supply a configured substitute value or keep the last value output and bring the controlled process to a safe operating state.

Timers and watchdogs are stopped. The timers for time-driven events are stopped.

In HOLD operating state, hardware interrupt OBs are not started by external interrupt triggers (e.g. hardware interrupt input).

The system time continues running.

At a POWER OFF > POWER ON transition while in HOLD mode, the CPU goes to STOP mode and remains in this state. In this case, the CPU does not perform an automatic startup.

When the user program reaches a breakpoint, the cycle monitoring time is stopped. When the breakpoint is exited again, the monitoring time restarts.

Errors that trigger a STOP of the CPU in RUN mode also do so in HOLD mode.

See also

Operating mode transitions (Page 900)

Basics of a memory reset

Function

A memory reset on the CPU is possible only in "STOP" mode.

When memory is reset, the CPU is changed to an "initial status". This means:

- An existing online connection between the PG/PC and the CPU is disconnected.
- The content of the work memory and the retentive and non-retentive data are deleted.
- The diagnostic buffer, time of day, and IP address are retained.
- The CPU is then initialized with the loaded project data (hardware configuration, code and data blocks, force jobs). These data are copied from the load memory to the work memory. Result:
  - If an IP address was set in the hardware configuration ("Set IP address in the project" option), this IP address is valid after the memory reset.
  - Data blocks no longer have current values but rather their configured start values.
  - Force jobs remain active

Memory areas

What you need to know about SIMATIC memory cards

Function of the SIMATIC memory card

The SIMATIC memory card for an S7-1500 is an SD memory card, preformatted by Siemens for the CPU user program, that is compatible with the Windows operating system.

NOTICE
Formatting the SIMATIC memory card You may only delete files and folders. If you format the SIMATIC memory card with Windows tools, for example with a commercially available card reader, you will render the memory card unusable as a storage medium for an S7 CPU.

Setting the card type

You can use the SIMATIC memory card as a program card or as a firmware update card.

To set the card type, insert the SIMATIC memory card into the card reader of the programming device and select the "Card reader/USB memory" folder from the project tree. In the properties of the selected memory card, designate the card type:

- Program
  You use a program card as an external load memory for the CPU. It contains the entire user program for the CPU. The user program is transferred to the work memory and runs there. If the SIMATIC memory card with the user program is removed, the program is no longer available.
  Directory: SIMATIC.S7S
- Firmware card
  Firmware for the S7-1500 modules can be stored on a SIMATIC memory card. It is therefore possible to perform a firmware update with the help of a specifically prepared SIMATIC memory card.
  Directory: FWUPDATE.S7S

Folders and files on the SIMATIC memory card

The following folders and files can be found on the SIMATIC memory card:

Folder	Description
FWUPDATE.S7S	Firmware update files for CPU and I/O modules
SIMATIC.S7S	User program, meaning all blocks (OBs, FCs, FBs, DBs) and system blocks, project data of the CPU
SIMATIC.HMI	HMI-relevant data
DataLogs	DataLog files
Recipes	Recipe files
UserFiles	User data
Backups	Files for backing up and restoring via the display

The content of the "DataLogs", "Recipes" and "UserFiles" folders is also loaded when the device is loaded as a new station into the STEP 7 project (as of TIA Portal V15.1). The files are located in the project path under "AdditionalFiles/MemoryCards/<Name of the PLC>". You can find further information on DataLogs and recipe files in the chapter "Recipes and Data Logging".

The following data structure exists on the SIMATIC memory card:

File type	Description
S7_JOB.S7S	Job file
SIMATIC.HMI\Backup*.psb	Panel backup files
SIMATICHMI_Back-ups_DMS.bin	Protected file (required to use panel backup files in STEP 7)
__LOG__	Protected system file (required in order to use the card)
crdinfo.bin	Protected system file (required in order to use the card)
DUMP.S7S	Service data file
*.pdf, *.txt, *.csv, ...	Additional files with different formats that you can also store in folders of the SIMATIC memory card. Only if you have saved these files in a folder by the name "UserFiles", are they stored when loading the device as a new station in the STEP 7 project.

Transferring objects from the project to a SIMATIC memory card

When the SIMATIC memory card is inserted in the programming device or in an external card reader, you can transfer the following objects from the project tree to the SIMATIC memory card:

- Individual blocks (multiple selection possible)
  In this case a consistent transfer is available, as the dependencies of the blocks to each other is taken into account with block selection.
- PLC
  In this case, all objects relevant to processing, such as blocks and the hardware configuration, are transferred to the SIMATIC memory card, just as with downloading.

To perform the transfer, you can move the objects with drag-and-drop or use the command "Card reader/USB memory > Write to memory card" in the "Project" menu.

Saving user data on a SIMATIC memory card

You transfer user files, such as PDF files for project documentation, onto the SIMATIC memory card by manually creating a folder called "UserFiles" on the memory card and saving your files in this folder. Only files in this folder (*.pdf, *.txt, *.csv, etc) are also loaded into the STEP 7 project when loading the device as a new station. The "UserFiles" folder can be used for restoring in the case of a memory card or CPU failure.

After being loaded from the memory card, the "UserFiles" folder with the files contained in it is saved in the project tree in the "AdditionalFiles > MemoryCards > [Name of the CPU]" directory.

If the folder "[Name of the CPU]" already exists, as is the case in reloading from the memory card, the folder in the project tree is overwritten by the data from the memory card.

Transferring objects from the SIMATIC memory card to the project

You transfer objects by selecting the SIMATIC memory card symbol in the project tree and then dragging it to a CPU in the project. The contents of the SIMATIC memory card must correspond to the CPU type in the project tree. If blocks existed beforehand, these are deleted prior to the transfer.

Updating firmware with a SIMATIC memory card

You can get the latest firmware data on the Internet from the Service & Support pages:

http://support.automation.siemens.com

Save the firmware files on the hard disk and plug the SIMATIC Memory Card into the card reader of your programming device.

To store the file on the SIMATIC memory card, select the memory card in the "Card Reader/USB memory" folder in the project tree. Select the shortcut menu "Card Reader/USB memory > Create firmware update memory card".

Then follow the instructions for performing the firmware update in the Service & Support portal or in the S7-1500 system manual.

Updating the firmware provides the module, for example, CPU or I/O module, with a new firmware version. If you have used the module in the project, you must replace the already configured module with the new firmware version offline by means of module replacement and adapt and then load the program or configuration.

Diagnosing manipulation of SIMATIC memory cards

If a CPU has been out of operation for a time, it may not be possible to rule out manipulation of the SIMATIC memory card. The program may function differently to before the CPU was taken out of operation.

S7-1500 CPU as of firmware version 2.0 are capable of diagnosing the following manipulation, and enter these events during startup as a security event in the diagnostics buffer:

- The project on the SIMATIC memory card has changed (the SIMATIC memory card remains the same)
- The SIMATIC memory card was replaced by another memory card

Service life of SIMATIC memory cards

The service life of SIMATIC memory cards is limited and depends on the following factors, among others:

- Capacity of the SIMATIC memory card
- Frequency of writing operations
- Amount of data written to the SIMATIC memory card.

Write or delete processes, particularly cyclic write/delete processes via the user program to the SIMATIC memory card, reduce its service life.

Cyclic execution of the instructions "CREATE_DB" (with "Create DB in load memory" attribute), "DataLogWrite", "RecipeExport", "RecipeImport" and setting the time zone "SET_TIMEZONE" reduce the service life of the memory card depending on the number or write processes and data.

In addition to the cyclic write/delete processes, writing or deleting very large amounts of data also has a negative impact on the service life of the SIMATIC memory card.

You can find a description of how to estimate the service life of SIMATIC memory cards for S7-1500 CPUs, ET 200SP CPUs and ET 200Pro CPUs in the following entry:

SIMATIC S7-1500 structure and use of the CPU memory (https://support.industry.siemens.com/cs/ww/en/view/59193101)

For S7-1200 CPUs, you can find the information under the following entry:

SIMATIC S7 S7-1200 Automation System (https://support.industry.siemens.com/cs/ww/en/view/109478121)

You have the option of configuring a service life percentage in the CPU properties so that the CPU generates a diagnostic interrupt and a diagnostic buffer entry (maintenance event) after expiration of the configured service life percentage. This will give you ample time to replace the SIMATIC memory card before it becomes unusable because the maximum number of write access operations has been reached.

Use the instruction "GetSMCinfo" to retrieve information about the inserted SIMATIC memory card. You select the information you wish to retrieve with the "Mode" parameter, for example:

- Mode = 2: Percentage of service life used so far (in %)
- Mode = 3: Configured percentage of service life after which the CPU enters a maintenance event in the diagnostic buffer and turns on the Maintenance LED.

Force jobs on the SIMATIC memory card

Please remember that a force job stored on the SIMATIC memory card is executed on every CPU into which this memory card is inserted. If you do not want this to happen, terminate the force job before you insert the SIMATIC memory card into a different CPU.

See also

What you should know about group alarms for security events (Page 979)

CPU memory areas

Memory areas of the S7-1500 CPUs

The following figure shows the CPU memory areas and the load memory on the SIMATIC memory card. In addition to the load memory, the SIMATIC memory card may contain other data, such as recipes, Data Logs, HMI backups.

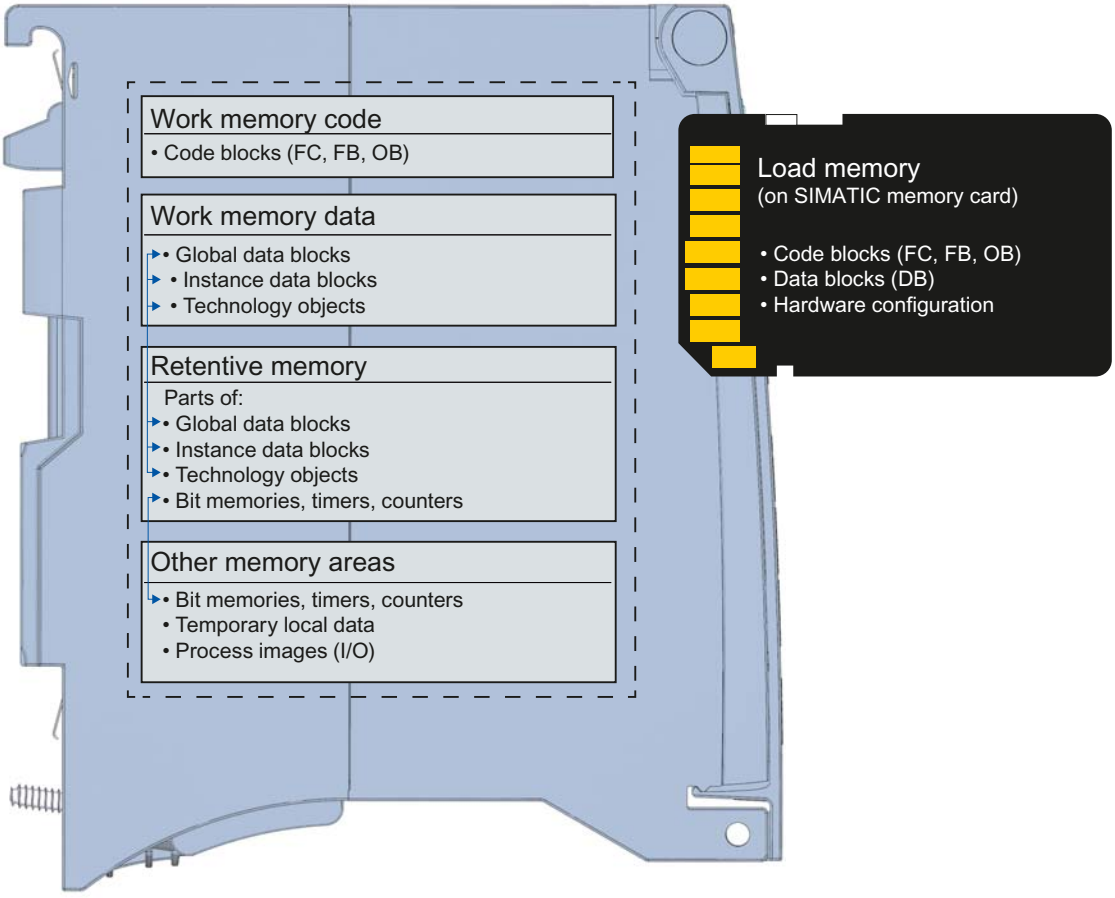

Load memory

The load memory is a non-volatile memory for program code, data blocks and hardware configuration. When these objects are loaded to the CPU, they are first stored in the load memory. This memory is located on the SIMATIC memory card. A SIMATIC memory card must be inserted in order for the CPU to operate.

Work memory

The work memory is volatile memory that contains the code and data blocks. The work memory is integrated in the CPU and cannot be extended.

The work memory in the S7-1500 is divided into two areas:

- Code work memory for runtime-relevant parts of the program code.
- Data work memory for runtime-relevant parts of data blocks and technology objects. At the operating mode transitions POWER ON → startup and STOP → startup, tags from global data blocks, instance data blocks, and technology objects are initialized with their start values. Retentive tags retain their actual values that were saved in the retentive memory.

Retentive memory

Retentive memory is a non-volatile memory for storing a limited amount of data in the event of a power failure. The tags and operand areas that have been defined as retentive are saved in the retentive memory.

This data is retained after a power-off or power failure. All other tags are lost and are set to their start values during the operating mode transitions startup after POWER ON and startup after STOP.

The contents of the retentive memory are deleted by the following actions:

- Memory reset
- Reset to factory settings

Specific tags of technology objects are also stored in the retentive memory. These tags are not deleted during memory reset.

Other memory areas

The CPU has other memory areas in addition to those described for the user program and data.

These areas include the following runtime-relevant areas:

- Bit memory, timers, counters
- Temporary local data
- Process images

The relevant CPU-specific variables can be found in the CPU's technical specifications.

Information on memory areas

You can view information offline and online via the memory areas of your S7-1500 CPU.

Offline:

During the creation or modification of a program, you can determine whether or not it is too large for a specific CPU. You can find this information, for example, under "Program information" in the project tree. In the "Resources" tab, you will find information on the overall size of the memory areas and on the memory space already allocated. At this location, you can also find information on the inputs and outputs that are already assigned.

For the S7-1500, you can specify the total size of the load memory (program information; the Load memory - total box is a drop down list for selecting the load memory size).

Online:

Start "Online & Diagnostics" in the project navigation. The "Memory" function provides information on the fill level of all the CPU's memories.

Additional information on your CPU's resources can be found in the "Assignment list" and "Call structure" tabs of the program information as well as in the PLC tag table.

See also

Showing fill levels of all types of memory on a CPU (Page 3322)

Retentive memory areas

Introduction

The S7-1500 CPUs have a retentive memory for storing retentive data in the event of a POWER OFF. The size of the retentive memory is documented in the CPU's technical specifications.

The utilization of the retentive memory of the configured CPU can be found offline in the project tree under "Program information > Resources" or online in the project tree under "Online & Diagnostics > Diagnostics > Memory".

If you define data as retentive, their content is retained after a power failure, during CPU startup, and when a modified program is loaded.

The following data and objects can be defined as retentive:

- Bit memory, timers, counters
- Tags of global data blocks
- Tags of instance data blocks of a function block

Certain tags of technology objects are always retentive, for example, calibration values of absolute encoders.

Bit memory, timers, counters

In the PLC tag table, you use the "Retain" button to define the number of retentive bit memories, timers and counters.

Tags of a global data block

Depending on the access settings, you can define either individual tags or all tags of a global data block as retentive in a global data block:

- Optimized block access enabled: retentivity can be set for each individual tag.
- Optimized block access disabled: The retain setting applies to all tags of the DB; either all or none of the tags are retentive.

Instance data block (function block) tags

You can define the tags of a function block's instance data block as retentive. Depending on the access to the instance data block, you can define the following for individual tags or for all tags of the block jointly:

- Optimized block access enabled: You can define individual tags as retentive in both the interface of the function block and in the assigned instance data block.
- Optimized block access disabled: You can define retentivity for all tags together in the instance data block.

Creating a data block in the user program

The instruction "Create_DB" is used to create a data block. Depending on the selection for the ATTRIB parameter, the generated data block either has the property "retentive" or "non-retentive". In this case, it is not possible to set the retentivity for individual tags.

Technology object tags

Certain tags of technology objects are also retentive, for example, calibration values of absolute encoders. STEP 7 manages the retentivity of the technology objects automatically. Hence, it is not necessary to configure retentivity for these. These tags are not affected by a memory reset and can only be deleted by resetting to factory settings.

Tip: Detect loss of retentivity in the startup OB and cyclic OBs

All tags (bit memories, timers, counters, DB content) of a user program lose their retentivity in the following case:

1. You plug the SIMATIC memory card with project (user program) into the card reader of your programming device.
2. You change the size of data blocks in the associated project or add data blocks with retentive tags to the associated project.
3. You transfer the changed project back to the SIMATIC memory card in the card reader.
4. You start the CPU with the changed project.

All tags (bit memories, timers, counters, DB content) of a user program lose their retentivity due to the offline changes. The tags are assigned their initial values after the STOP-RUN transition.

To react to a loss of retentivity in the user program, evaluate the following local tags of the start information:

- Startup OB: LostRetentive (true in case of loss of retentivity)
- Cyclic OBs Remanence (true if retentive data are available)

Summary of retentive behavior

Retentive characteristics of the memory objects

In addition to the retentive memory areas described, there are other objects with retentive characteristics, for example, the diagnostic buffer. These objects do not occupy retentive memory.

The following table shows the retentive characteristics of the memory objects:

- in the operating mode transitions STOP → Startup and POWER OFF → Startup
- in the memory-influencing functions Memory reset and Reset to factory settings

Table 1-83 Retentive characteristics of memory objects ("x" = content is retained; "-" = content is initialized)

Memory object	Operating mode transitions		Memory reset	Reset to factory settings
	STOP → STARTUP	POWER OFF → STARTUP		
Actual values of the data blocks, instance data blocks	Can be set in the properties of the DB.		-	-
Bit memories, timers and counters - configured as retentive	x	x	-	-
Bit memories, timers and counters - configured as non-retentive	-	-	-	-
Certain retentive tags from technology objects (for example, adjustment values of absolute encoders)	x		x	-
Diagnostic buffer (retentive area)	x	x	x	-
Diagnostic buffer entries (non-retentive area)	x	-	-	-
Runtime meters	x	x	x	-
Time-of-day	x	x	x	-

Diagnostics buffer

Part of the diagnostic buffer of S7-1500 CPUs is retentive. The most recent entries in the diagnostic buffer are retained after a power failure or memory reset. The retentive part of the diagnostic buffer can only be deleted by resetting to factory settings. The number of retentive diagnostic buffer entries depends on the CPU. The entries in the diagnostic buffer do not require retentive memory.

Runtime meters

The runtime meter of S7-1500 CPUs is retentive and not affected by a memory reset. After a reset to factory settings, it is set to zero.

Time-of-day

The time-of-day of S7-1500 CPUs is retentive and not affected by memory reset. After a reset to factory settings, the time-of-day is set to 01/01/2012 00:00:00.

Memory behavior when loading software changes

Introduction

In the STOP and RUN modes of the CPU, the S7-1500 CPU offers you the option of loading changes to the software without affecting the actual values of already loaded tags during the download operation.

In STEP 7, you load changes to the software under "Download to device > Software (changes only)".

Impact of software changes on PLC tags

You can load the following software changes, **without** affecting the actual values of tags that have already been loaded:

- Changing names
- Changing comments
- Adding new tags
- Deleting tags
- Changing retentivity settings for bit memories, times, and counters.

The actual values are affected by loading the following software changes:

- Changing data types
- Changing addresses

Impacts of software changes on data blocks with memory reserve

If you are using memory reserve for data blocks ("Optimized block access" attribute and "Download without reinitialization" button activated), you can load the following software changes **without** the actual values of already loaded tags being reinitialized.

- Changing start values
- Changing comments
- Adding new tags

To load the following software changes, you must deactivate the "Download without reinitialization" button. During the next load, all actual values of the data block will be reinitialized:

- Changing names
- Changing data types
- Changing retentivity settings

- Deleting tags
- Changes to the memory reserve settings

Impacts of software changes on data blocks without memory reserve

If you are not using memory reserve, you can load the following software changes, without affecting the actual values of tags that have already been loaded:

- Changing start values
- Changing comments

When loading the following software changes, all actual values of the data block will be reinitialized:

- Changing names
- Changing data types
- Changing retentivity settings
- Adding new tags
- Deleting tags

Memory reserve of data blocks

Each function block or data block with the activated attribute "Optimized block access" contains a memory reserve by default which you may use for subsequent interface changes. The memory reserve is initially not used. When you have compiled and loaded the block, and then observe that you want to reload interface changes, activate the memory reserve. All tags that you subsequently declare will be placed in the memory reserve. During the next load the new tags will be initialized to their start values; the previously loaded tags will not be reinitialized.

The setting for memory reserve can be found in the data block properties under "Download without reinitialization".

Memory requirements when loading software changes in RUN

For the consistent and atomic handling of the entire load procedure, the CPU requires adequate free memory space on the SIMATIC memory card. The files affected by loading the software changes on the CPU are only deleted after the new files have been created. This SIMATIC memory card therefore requires memory space corresponding approximately to the space required for all program objects on the memory card.

You will find information and tips on handling software changes in the manual "Structure and use of the CPU memory".

I/O data areas

Access to the I/O addresses

I/O addresses

If you insert a module in the device view, its user data is located in the process image of the S7-1500-CPU (default). The CPU handles the data exchange between the module and the process image area automatically during the update of the process images.

Append the suffix ":P" to the I/O address if you want the program to access the module directly instead of using the process image.

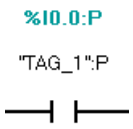

Direct I/O access does not improve performance for S7-1500 CPUs, however. Where possible, you should therefore use process images or process image partitions to address the I/O.

Start address of a module

Definition

The start address is the lowest byte address of a module. It acts as the initial address of the module user data area.

Configuring module start addresses

The addresses used in the user program and the module start addresses are coordinated when the modules are configured.

In the module properties ("I/O addresses" group), you can change the start addresses that were assigned automatically when the modules were inserted.

You can also set whether the addresses are located in the process image or in which process image partition they are located.

In addition to the process image partition, you can also select the organization block to which the process image partition is assigned. With the exception of the isochronous interrupt OB, the associated process image partition is updated when the assigned OB is called. Immediately before the beginning of the OB start, the process image partition of the inputs is updated. Immediately after the end of the OB, the process image partition of the outputs is transferred to the modules.

Select the "Automatic update" process image if the process image in which the addresses of the module are located is to be updated automatically at the cycle control point.

Evaluating the value status

Introduction

Value status (QI, quality information) refers to diagnostic information of I/O channels that is made available to the user program via the process image inputs (PII). The value status is transferred synchronously with the user data.

Each bit of the value status is assigned to a channel and provides information on the validity of the value (0 = value is incorrect).

For example, if there is a wire break at a digital input, the user data signal is logically "0". The module also sets the associated bit in the value status to 0 due to the wire break diagnostics, which means you have the option to determine that the value is invalid by querying the value status.

The assignment of the value status bytes in the process image inputs depends on the module used. Detailed information can be found in the manual for the relevant I/O module.

Summary: Possible cause for value status = 0:

- The supply voltage L+ is missing at the terminals or is not sufficient
- A channel has been deactivated
- Outputs are inactive (e.g. CPU in STOP)
- An output channel uses technology functions (HSC, PWM or PTO); see below

Enable value status

If you want to evaluate the value status of the channel, select the "Value status" check box in the properties of the I/O module. By activating the value status, additional addresses are assigned for the value status in the process image inputs (PIIs). For input modules, STEP 7 assigns the input addresses directly following the user data; for output modules, the next available input addresses are assigned.

Special features for modules with MSI/MSO

Requirement:

- GSD file for I/O modules with MSI/MSO
- IO device supports MSI/MSO (e.g., IM 155-5 PN ST firmware version V2.0.0 or higher)

The value status has the following meaning for modules with diagnostic capability for which you have configured a module-internal Shared Input (MSI) or module-internal Shared Output (MSO):

MSI:

For the 1st submodule (=basic submodule), the value status 0 indicates that the value is incorrect.

For the 2nd to 4th submodule (=MSI submodule), the value status 0 indicates that the value is incorrect or the basic submodule has not yet been assigned parameters (not ready).

MSO:

For the 1st submodule (=basic submodule), the value status 1 indicates that the output value specified by the user program is actually output at the module terminal.

Possible causes for value status = 0:

- Value is incorrect, for example, because the supply voltage is missing.
- IO controller of basic submodule is in STOP.

For the 2nd to 4th submodule (=MSO submodule), the value status 1 indicates that the output value specified by the user program is actually output at the module terminal.

Possible causes for value status = 0:

- Value is incorrect, for example, because the supply voltage is missing.
- IO controller of basic submodule is in STOP.
- The basic submodule parameters have not yet been assigned.

Behavior of the value status at the output channels for technology functions with compact CPUs, such as CPU 1511C-1 PN

The CPU 1511C-1 PN supports the value status (QI) for the channels of the analog and digital on-board I/O.

The value status is not supported by the output channels you are using for technology functions (HSC, PWM or PTO).

Therefore, the QI bits always output the value status 0 ("Bad") at the output channels for technology functions. It does not matter in this context whether the output value is incorrect or not.

See also

Cyclic OBs (Page 1000)

Example for evaluating the value status (Page 920)

System diagnostics (http://support.automation.siemens.com/WW/view/en/68052815)

Module-internal shared input/shared output (MSI/MSO) (http://support.automation.siemens.com/WW/view/en/68052815)

Example for evaluating the value status

The DI 32x24VDC HF module of the S7-1500 can be configured in STEP 7 (device view) for various applications. Depending on the configuration, additional/different addresses are assigned in the process image input.

This example shows various configurations with their respective assigned value status and describes the effect on the address assignment.

Configuration options of DI 32x24VDC HF

The following configurations are possible:

- 1 x 32-channel without value status
- 1 x 32-channel with value status
- 4 x 8-channel without value status
- 4 x 8-channel with value status
- 1 x 32-channel with value status for module-internal shared input (MSI) with up to 4 submodules

An additional bit for the value status is assigned to each channel in the configurations with value status. The value status bit indicates if the read in digital value is valid. (0 = value is incorrect).

Address space for configuration as 32-channel DI 32x24VDC HF with value status

The preset DI configuration is retained initially (32 channels, neither division into submodules nor module-internal shared input (MSI) configuration); however, the value status is to be enabled.

- Select the "Enable value status" check box in the "DI configuration" area in the module properties (Inspector window).

The following figure shows the assignment of the address space for the configuration as a 32-channel module with value status. You can freely assign the start address for the module. The addresses of the channels are derived from the start address.

Example: Start address 2 (input byte 2), end address 9 (input byte 9).

STEP 7 automatically reserves the addresses for the input values (4 bytes) and the subsequent addresses for the value status (also 4 bytes) so that the following address assignment results:

Assignment in the process image input (PII)

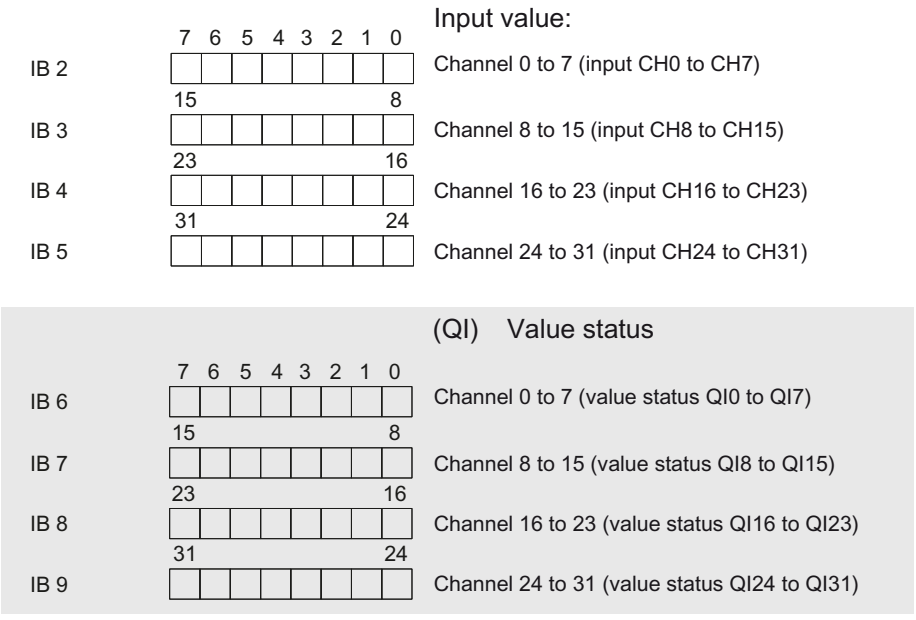

QI=0: read-in value at the channel is incorrect

Address space for configuration as 4 x 8-channel DI 32x24VDC HF with value status

The DI configuration is to be adapted so that the 32 channels of the DI 32x24VDC HF module of the S7-1500 are divided among four submodules with 8 channels each. The submodules can be assigned to different IO controllers when the module is used in a shared device. The value status is also to be enabled in this case.

- Select the submodule configuration "4 submodules with 8 digital inputs" in the "DI configuration" area in the module properties (Inspector window).
- Select the "Enable value status" check box.

The 4 submodules appear with their own name (e.g., DI 32x24VDC BA_1) in the area navigation of the Inspector window.

Contrary to the 1 x 32-channel module configuration, each of the four submodules has a freely assignable start address.

Example: The start addresses are assigned as follows:

- 1. Submodule: 10
- 2. Submodule: 20
- 3. Submodule 30
- 4. Submodule 40

STEP 7 automatically reserves a byte for the value status for each submodule. The byte for the value status has the address "Start address+1" so that the following address assignment results:

Assignment in the process image input (PII)

Input value:

	7 6 5 4 3 2 1 0		
IB 10		Channel 0 to 7 (input CH0 to CH7)	Submodule 1
	15 … 8		
IB 20		Channel 8 to 15 (input CH8 to CH15)	Submodule 2
	23 … 16		
IB 30		Channel 16 to 23 (input CH16 to CH23)	Submodule 3
	31 … 24		
IB 40		Channel 24 to 31 (input CH24 to CH31)	Submodule 4

(QI) Value status

	7 6 5 4 3 2 1 0		
IB 11		Channel 0 to 7 (value status QI0 to QI7)	Submodule 1
	15 … 8		
IB 21		Channel 8 to 16 (value status QI8 to QI15)	Submodule 2
	23 … 16		
IB 31		Channel 16 to 23 (value status QI16 to QI23)	Submodule 3
	31 … 24		
IB 41		Channel 24 to 31 (value status QI24 to QI31)	Submodule 4

QI = 0: read-in value at the channel is incorrect

Address space for configuration as 1 x 32-channel DI 32x24VDC HF with MSI and value status

The configuration shown here uses the "Module-internal shared input (MSI)" function. This function allows simultaneous reading of the input values by various IO controllers (CPUs) when the module is used in a shared device (ET 200MP).

The channels 0 to 31 of the module are copied in up to 4 submodules with configuration 1 x 32-channel module with module-internal shared input (MSI). Channels 0 to 31 are then available with identical input values in different submodules. These submodules can be assigned to up to four IO controllers when the module is used in a sShared device. Each IO controller has read access to the same channels ("listening in").

The number of usable submodules (= copies of the inputs) depends on the interface module used. Observe the information regarding this in the device manual of the particular interface module.

- Select the following options in the "DI configuration" area in the module properties (Inspector window):
  - For "Subdivision of module": None
  - For "Copies of module": One additional copy of the inputs
    This setting enables one additional IO controller to read the inputs of the module.

The value status is enabled automatically for MSI configurations (cannot be changed). The reason for this setting is that an IO controller of the shared device that is "listening in" can determine from the value status whether or not the module is ready for operation and providing valid values.

Below, the submodules that result from the above-described settings are named as follows:

- Basic submodule: That is the submodule assigned to the parameter-assigning IO controller. Only the parameter-assigning IO controller has access to module diagnostics and receives hardware interrupts.
- MSI submodule: That is the submodule assigned to an IO controller that is "listening in". The IO controller that is listening in does not have access to module diagnostics and only receives information on the validity of the input values by the value status.

Value status (Quality Information, QI)

The meaning of the value status depends on the submodule to which it pertains.

For the first submodule (=basic submodule), the value status 0 indicates that the value is incorrect.

For the 2nd to 4th submodule (=MSI submodule or copy of inputs), the value status 0 indicates that the value is incorrect or that the basic submodule parameters have not yet been assigned (not ready for operation).

Example: The DI 32x24VDC HF module is configured as described above (an additional copy of the inputs). A basic submodule and one MSI submodule result.

The basic submodule receives the start address 10, and the MSI submodule receives the start address 100 in this example. The address of the MSI submodule is not relevant for the IO controller because this submodule will be assigned to another IO controller later in the shared device configuration process and will disappear from the address space of the parameter-assigning IO controller as a result.

The following figure shows the assignment of the address space with submodules 1 and 2 and the value status. All submodules are initially assigned to the IO controller.

Assignment in the process image input (PII) for submodules 1 and 2

Input value of submodule 1 (basic submodule):

	7	6	5	4	3	2	1	0	
IB 10									Channel 0 to 7 (input CH0 to CH7)
	15							8	
IB 11									Channel 8 to 15 (input CH8 to CH15)
	23							16	
IB 12									Channel 16 to 23 (input CH16 to CH23)
	31							24	
IB 13									Channel 24 to 31 (input CH24 to CH31)
	7	6	5	4	3	2	1	0	
IB 14									Channel 0 to 7 (value status QI0 to QI7)
	15							8	
IB 15									Channel 8 to 15 (value status QI8 to QI15)
	23							16	
IB 16									Channel 16 to 23 (value status QI16 to QI23)
	31							24	
IB 17									Channel 24 to 31 (value status QI24 to QI31)

Input value of submodule 2 (MSI submodule):

	7	6	5	4	3	2	1	0	
IB 100									Channel 0 to 7 (input CH0 to CH7)
	15							8	
IB 101									Channel 8 to 15 (input CH8 to CH15)
	23							16	
IB 102									Channel 16 to 23 (input CH16 to CH23)
	31							24	
IB 103									Channel 24 to 31 (input CH24 to CH31)
	7	6	5	4	3	2	1	0	
IB 104									Channel 0 to 7 (value status QI0 to QI7)
	15							8	
IB 105									Channel 8 to 15 (value status QI8 to QI15)
	23							16	
IB 106									Channel 16 to 23 (value status QI16 to QI23)
	31							24	
IB 107									Channel 24 to 31 (value status QI24 to QI31)

QI = 0: read-in value at the channel is incorrect (basic submodule) or module is not ready for operation (MSI submodule)

Additional information

Additional information and a programming example for the evaluation of the value status are available here (http://support.automation.siemens.com/WW/view/en/59192926).

See also

Evaluating the value status (Page 919)

Basics of program execution

English designations of the OB types

English designation of the OB types

When an OB is created in the "Add new block" dialog, the OB types can only be displayed in English for technical reasons – irrespective of the set user interface language. In order to facilitate handling, the following table lists the English terms and their corresponding designations in the set user interface language.

English designation of the OB type in the "Add new block" dialog.	Designation in the information system
Program cycle	Program cycle OB, OB for cyclic program processing, cyclic program
Startup	Startup OB
Time delay interrupt	Time-delay interrupt OB
Cyclic interrupt	Cyclic interrupt OB
Hardware interrupt	Hardware interrupt OB
Time error interrupt	Time error OB
Diagnostic error interrupt	Diagnostic interrupt OB
Pull or plug of modules	Pull/plug interrupt OB
Rack or station failure	Rack failure OB
Programming error	Programming error OB
IO access error	I/O access error OB
Time of day	Time-of-day interrupt OB
MC-Interpolator	MC-Interpolator OB
MC-Servo	MC-Servo OB
MC-PreServo	MC-PreServo OB
MC-PostServo	MC-PostServo OB
MC-Transformation	MC-Transformation OB
Synchronous cycle	Isochronous mode interrupt OB
Status	Status interrupt OB
Update	Update interrupt OB
Profile	OB for vendor or profile-specific interrupt

Events and OBs

OB start events

An OB start event causes the following reaction when it occurs:

- If the event comes from an event source to which you have assigned an OB, it triggers the execution of the assigned OB. This means that the event is positioned in a queue according to its priority.
- If the event comes from an event source to which you have not assigned an OB, the default system reaction is executed.

Note

Some event sources, such as startup, pull/plug, exist even if you do not configure them.

The following table provides an overview of the OB start events, including the possible values for OB priority, possible OB numbers, default system reaction and number of OBs. The table is sorted based on the default OB priority. Priority class 1 is the lowest.

Types of event sources	Possible priorities (default priority)	Possible OB numbers	Default system reaction	Number of supported OBs
Startup	1	100, >= 123	Ignore	100
Cyclic program	1	1, >= 123	Ignore	100
Time-of-day interrupt	2 to 24 (2)	10 to 17, >= 123	not applicable	20
Status interrupt	2 to 24 (4)	55	Ignore	1
Update interrupt	2 to 24 (4)	56	Ignore	1
Manufacturer- or profile-specific interrupt	2 to 24 (4)	57	Ignore	1
Time-delay interrupt	2 to 24 (3)	20 to 23, >= 123	not applicable	20
Cyclic interrupt	2 to 24 (8 to 17, depending on the cycle time)	30 to 38, >= 123	not applicable	20
Hardware interrupt	2 to 26 (16)	40 to 47, >= 123	Ignore	50
Isochronous mode interrupt	16 to 26 (21)	61 to 64, >= 123	Ignore	20 (one per isochronous interface)
CPU redundancy error (for R/H-CPUs only)	2 to 26 (26)	72	Ignore	1
MC-Interpolator	16 to 25 (24)	92	Not applicable	1
MC-Servo	17 to 26 (26)	91	Not applicable	1
MC-PreServo	Corresponds to MC-Servo	67	Not applicable	1
MC-PostServo	Corresponds to MC-Servo	95	not applicable	1
MC-Transformation	17 to 25 (25)	98	not applicable	1
Time error [1)]	22	80	Ignore	1
Cycle monitoring time violated once			STOP	
Diagnostic interrupt	2 to 26 (5)	82	Ignore	1
Removal/insertion of modules	2 to 26 (6)	83	Ignore	1

Types of event sources	Possible priorities (default priority)	Possible OB numbers	Default system reaction	Number of supported OBs
Rack error	2 to 26 (6)	86	Ignore	1
Programming error (only for global error handling)	2 to 26 (7)	121	STOP	1
I/O access error (only for global error handling)	2 to 26 (7)	122	Ignore	1

[1] How does the R/H system behave in the case of cycle timeouts? See "Redundant system S7-1500R/H System Manual".

Assignment between event source and OBs

The type of OB determines where you make the assignment between OB and event source:

- With hardware interrupts and isochronous mode interrupts, the assignment is made during the configuration of the hardware or when the OB is created.
- STEP 7 automatically assigns OBs 91/92 to the MC Servo interrupt and MC-Interpolator interrupt, as soon as a technology object is added to "S7-1500 Motion Control".
- For all other types of OB, the assignment is made when the OB is created, possibly after you have configured the event source.

For hardware interrupts, you can change an assignment which has already been made during runtime with the instructions ATTACH and DETACH. In this case, only the actually effective assignment is changed, not the configured assignment. The configured assignment is effective after loading and for every startup.

Hardware interrupts to which no OB is assigned (because they were configured in this way or after the DETACH instruction is called) are ignored. The check as to whether an OB is assigned to an event does not take place when the associated event occurs, but only when the hardware interrupt actually has to be executed.

OB priority and runtime characteristics

S7-1500 CPUs support the priority classes 1 (lowest) to 26 (highest). An OB is assigned the priority of its start event.

The OBs are executed exclusively on a priority basis. This means that the OB with the highest priority is executed first when multiple OB requests occur at the same time. If an event with higher priority than the currently active OB occurs, this OB is interrupted. Events of the same priority are processed in order of occurrence.

CPU overload behavior

Principle of CPU overload behavior

The sequence of the user program for S7-1500 CPUs is based on events of different priorities. An event triggers the execution of an OB. The OBs are executed exclusively on a priority basis. This means that the OB with the highest priority is executed first when multiple OB requests occur at the same time. If an event with higher priority than the currently active OB occurs, this OB is interrupted. Events of the same priority are processed in order of occurrence.

For the event scenarios considered in the following section, it is assumed that you have assigned an OB to each event source and that these OBs have the same priority. The second condition, in particular, is only for the sake of a simplified representation.

When an event occurs, the execution of the associated OB is triggered. This means that the event is positioned in a queue according to its priority. Depending on the OB priority and the current processor load, there is usually a time delay before the OB is executed. The same event can therefore occur once or several times before the OB belonging to the preceding event has been processed. Such a situation is handled by the CPU as follows: In the order of their occurrence, the operating system positions the events into the queue according to their priority.

To control temporary overload situations, you can limit the number of pending events that are linked from the same source. The next event is discarded as soon as the maximum number of pending start events of a specific cyclic interrupt OB, for example, is reached.

An overload occurs when events which originate from the same source occur faster than they can be processed.

More detailed information is provided in the following sections.

Discarding similar events or fetching them later

Below, the term "similar events" refers to events from a single source, such as start events for a specific cyclic interrupt OB.

The OB parameter "Events to be queued" is used to specify how many similar events the operating system places in the associated queue and therefore post-processes. If this parameter has the value 1, for example, exactly one event is stored temporarily.

Note

Post-processing of cyclic events is often not desirable, as this can lead to an overload with OBs of the same or lower priority. Therefore, it is generally advantageous to discard similar events and to react to the overload situation during the next scheduled OB processing. If the value of the "Events to be queued" parameter is low, this ensures that an overload situation is mitigated rather than aggravated.

If, for example, the maximum number of start events is reached in the queue for a cyclic interrupt OB, each additional start event is only counted and subsequently discarded. During the next scheduled execution of the OB, the number of discarded start events is made available (in optimized start information) in the "event_count" input parameter. You can then react appropriately to the overload situation. The CPU then sets the counter for lost events to zero.

If the CPU, for example, first discards a start event for a cyclic interrupt OB, its further behavior depends on the OB parameter "Report event overrun into diagnostic buffer": If the check box is selected, the CPU enters the event DW#16#0002:3507 once in the diagnostic buffer for the overload situation at this event source. Additional diagnostic buffer entries of the event DW# 16 #0002:3507 that refer to this event source are suppressed until all events from this source have been post-processed.

Threshold mechanism for time error OB request

The OB parameter "Enable time error" is used to specify whether the time error OB should be called when a specific overload level is reached for similar events.

If yes (check box selected), you use the OB parameter "Event threshold for time error" to specify the number of similar events in the queue at which the time error OB should be called. If this parameter has the value 1, for example, the event DW#16#0002:3502 is entered once in the diagnostic buffer and the time error OB is requested when the second event occurs. Additional diagnostic buffer entries of the event DW# 16 #0002:3502 that refer to this event source are suppressed until all events from this source have been post-processed.

In the event of an overload, you therefore have the option of programming a reaction well before the limit is reached for similar events and thus before the events are discarded.

The following value range applies to the "Event threshold for time error" parameter: 1 <= "Event threshold for time error" <= "Events to be queued"

Example of a hardware interrupt event

The functional principle of event-oriented program execution in the S7-1500 CPU is described based on the example of a module triggering a hardware interrupt.

Process events are events which are triggered by the I/O (e.g. a digital input) and initiate a call of the assigned OB in the S7-1500 CPU. OBs assigned to a process event are called hardware interrupt OBs.

Procedure

To configure a hardware interrupt, proceed as follows:

1. Configure an S7-1500 with CPU and a hardware interrupt-compatible module, e.g. DI 16x24VDC HF.
2. Select the hardware interrupt-compatible module and navigate in the area navigation to the area "Inputs > Channel 0".
3. Activate the option "Enable rising edge detection".
4. Assign a meaningful name to the event.

5. If necessary, change the priority of this event.
   In the PLC tags ("System constants" tab), a system constant of the data type Event_HwInt is created for the event.

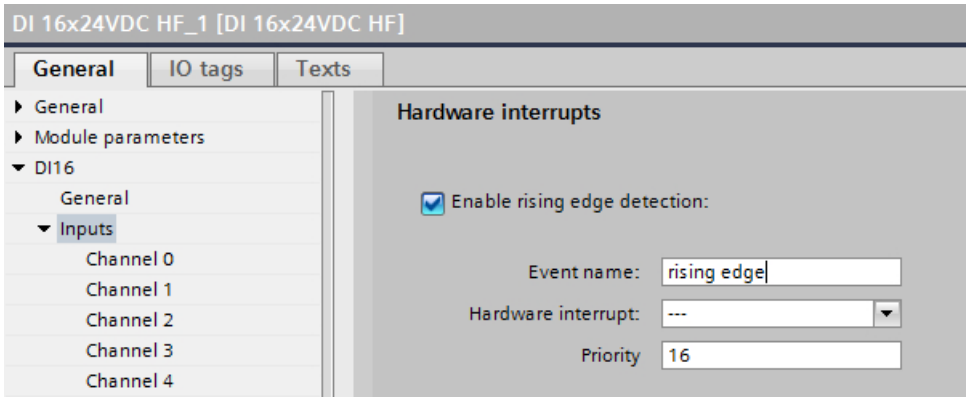

6. Click the "Hardware interrupt" drop down list to create a new hardware interrupt OB. If a hardware interrupt OB already exists, it can be selected.

7. Click the "Add object" button to create a new hardware interrupt OB.

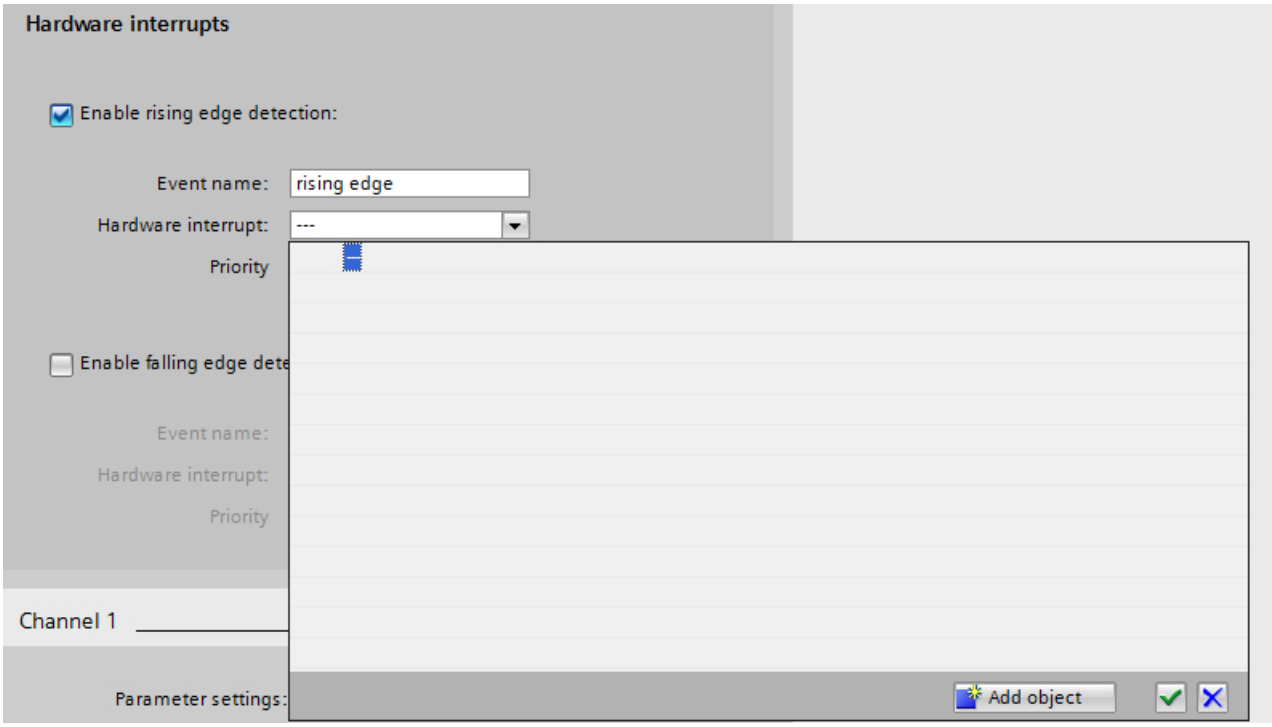

8. You can specify in the selection dialog whether the programming editor should start immediately with the opened hardware interrupt OB ("Add new and open" option).

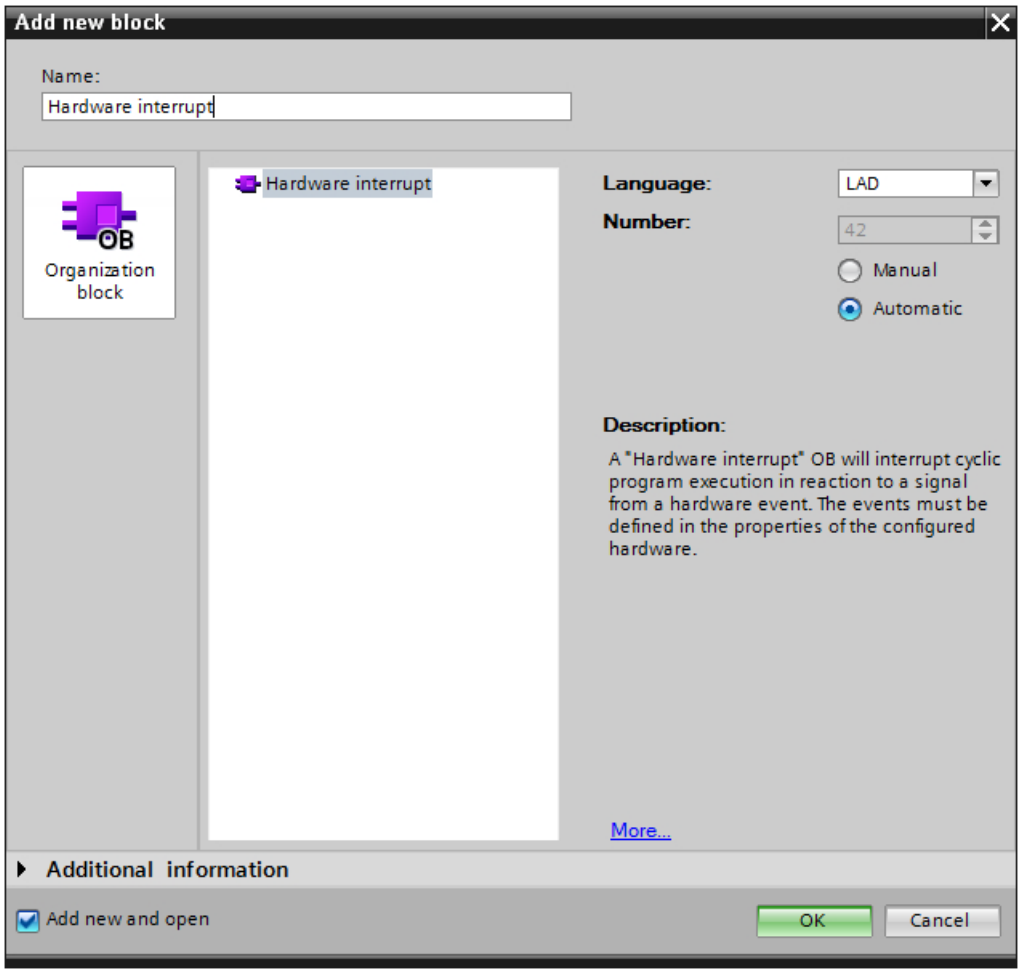

Hardware interrupt OB properties

The properties of the hardware interrupt OB contain the corresponding information about the start event for each interrupt-triggering event:

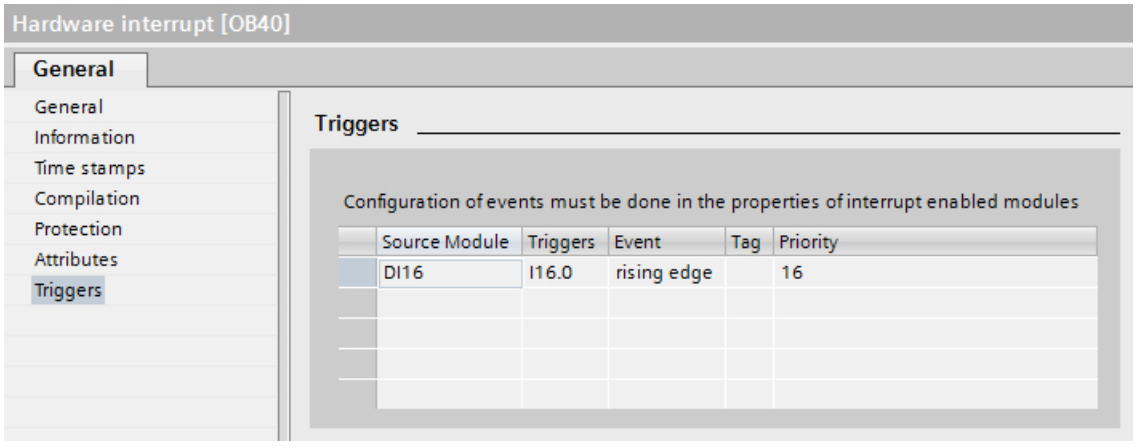

Hardware interrupt OB program

In the user program, you can use the start information of the hardware interrupt OB to identify and evaluate the hardware interrupt event.

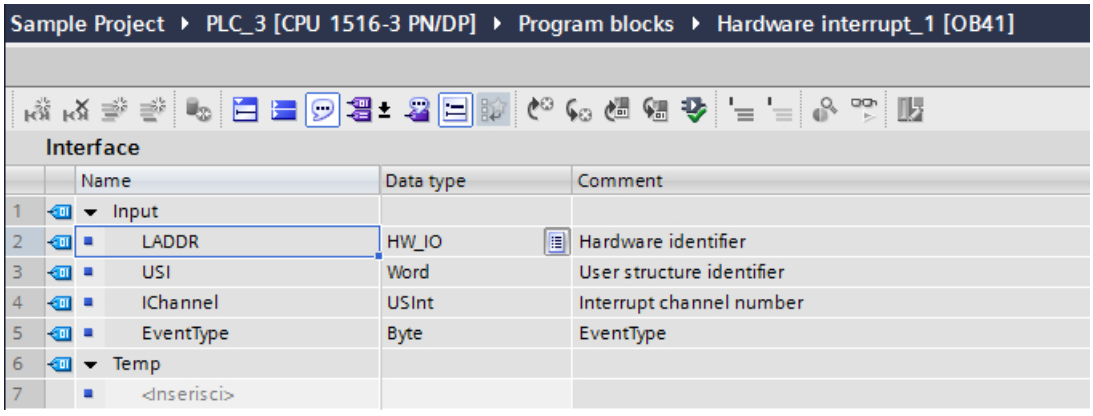

See also

Assigning parameters to hardware interrupt OBs (Page 1032)

Hardware interrupt OBs (Page 1010)

Setting the operating behavior

CPU properties

Overview of the CPU properties

Overview

The following table provides you with an overview of the CPU properties:

Group	Properties	Description
General	Project information	General information to describe the inserted CPU. Except for the slot number, you can change this information.
	Catalog information	Read-only information from the hardware catalog for this CPU.
	Identification & Maintenance	For saving application-specific information such as the name of the plant and the installation location.
	Checksums	To check the identity or integrity of PLC programs. Blocks in the block folder and text lists are automatically marked with unique checksums when they are compiled. You can easily establish whether the program currently running on the CPU is the same program that you loaded a long time ago or whether the program has been changed in the meantime. The "GetChecksum" instruction is available to read out the checksum while the program is running. See: Comparison of PLC programs based on checksums

Group	Properties	Description
PROFINET interface	General	Name and comment for this PROFINET interface. The name is limited to 110 characters.
	Ethernet addresses	Select whether the PROFINET interface is networked. If subnets have already been created in the project, they are available for selection in the drop-down list. If not, you can create a new subnet with the "Add new subnet" button. Information on the IP address, subnet mask and IP router usage in the subnet is available in the IP protocol. If an IP router is used, the information about the IP address of the IP router is necessary. See: Assigning addresses and names to PROFINET devices (Page 2840)
	Time synchronization	Settings for time synchronization in the NTP time format. The NTP (network time protocol) is a general mechanism for synchronizing system clocks in local and global area networks. In NTP mode, the interface of the CPU sends time queries (in client mode) at regular intervals to NTP servers on the subnet (LAN) and the addresses must be set in the parameters here. Based on the replies from the server, the most reliable and most accurate time is calculated and synchronized. The advantage of this mode is that it allows the time to be synchronized across subnets. The accuracy depends on the quality of the NTP server being used.
	Operating mode	Selection of operation of CPU as IO controller or IO device including the necessary settings. As IO device (I-device), you must assign an IO controller and specify if the PROFINET interface is configured by the higher-level IO controller or if the local IO controller configures the PROFINET interface. The prioritized startup can also be enabled for the role IO device if the higher-level IO controller supports this function. If you have configured the CPU as IO device, the table for configuration of the transfer areas is displayed. See: Configuring the I-device
	Advanced options	Name, comment and additional setting options of the Ethernet interface and the ports. This includes the following settings: • Interface options such as the reaction of the user program in case of communication errors, if device replacement is possible without exchangeable medium or automatic commissioning, setting regarding IEEE-compliant LLDP mode (PROFINET V2.3), time lag between sending of Keep Alives for connections. • Media redundancy settings, such as the media redundancy role (client, manager (Auto) or devices not participating in the media redundancy ring), display of the ring ports and if diagnostic interrupts are generated in case of redundancy errors. A "Domain settings" interface is the link to the domain settings of the connected subnet (MRP domain and Sync domain). • Real-time settings such as send clock and synchronization properties with a "Domain settings" interface which is the link to the domain settings of the connected subnet (MRP domain

Group	Properties	Description
		and Sync domain). In the Sync domain you set the send clock for isochronous IO systems and specify the RT classes (RT and IRT) as well as the synchronization roles for IRT. • Ports: For ports you set the interconnection with neighboring devices (alternative: use the topology view), if necessary. Port interconnections are required for isochronous applications (IRT) as well as for automatic commissioning ("Device replacement without exchangeable medium" option is selected). You also set the port options, for example, if the port is used (you can block unused ports). The port options include the connection settings (transmission rate/duplex) with the options for monitoring and autonegotiation. You also set the boundaries for the end of detecting accessible devices, for the end of topology detection and for the end of the sync domain. See: Setting the send clock (Page 2867) Interconnecting ports (Page 2865) Setting the port options (Page 2871) What you should know about media redundancy Introduction: Isochronous Realtime Ethernet Enabling device replacement without exchangeable medium (Page 2876)
	Web server access	Allows access to the web server function.
PROFIBUS interface	General	Name and comment for this PROFIBUS interface. The name is limited to 110 characters.
	PROFIBUS address	Selection of whether the PROFIBUS interface is networked. If subnets have already been created in the project, they are available for selection in the drop-down list. If not, you can create a new subnet with the "Add new subnet" button. The PROFIBUS address is assigned and the highest address and the transmission speed are displayed.
	Operating mode	Selection of CPU operating mode as DP master or DP slave. A DP master can be assigned to a DP slave and the I-slave communication parameters can be set here.
	Time synchronization	The time can be synchronized. A default can be set for the time source and synchronization frequency.
	SYNC/FREEZE	The DP master can send the SYNC and/or FREEZE control command to a group of DP slaves to synchronize these. For this purpose, the DP slaves are assigned to SYNC/FREEZE groups.

Group	Properties	Description
Startup	Startup after POWER ON	Setting the startup characteristics after a POWER OFF/POWER ON transition. See: Basics of "STARTUP" operating mode (Page 901)
	Comparison preset to actual configuration	Specifies the startup characteristics for situations in which the actual configuration of the S7-1500 station does not correspond to the preset configuration: • Startup of the CPU only if compatible • Startup of the CPU even if there are differences With the "Startup CPU only if compatible" setting, a module in a configured slot must be compatible with the configured module. Compatible means that the module that is present matches the number of inputs and outputs and must match with respect to its electrical and functional properties. A compatible module must be able to completely replace a configured module; it may be more capable, but not less capable. Example for the "Startup CPU only if compatible" setting: A CPU can be a compatible replacement for a CPU of the same type with a lower firmware version. An input module with 32 digital inputs can be a compatible replacement for a signal module with 16 digital inputs. The CPU starts up when the configured module or a compatible module is plugged in. The CPU does not start up if an incompatible module is inserted. Example of the "Startup CPU even if mismatch" setting: Instead of a configured digital input module, an analog output module is plugged in or no module is present in this slot. Although the configured inputs cannot be accessed, the CPU starts up. Note in this case that the user program cannot function correctly and take the appropriate measures. Note: For S7-1500 modules and ET 200SP modules, you can configure the comparison between preset module and actual module individually for each slot. The default is "From CPU" (in other words, the CPU's setting applies to the slot). For I/O modules from other I/O series, generally the CPU's setting applies (cannot be changed).
	Configuration time	Specifies a maximum time period (default: 60000 ms) in which the central and distributed I/O must be ready for operation. The CMs and CPs are supplied with voltage and communication parameters during the CPU startup. The configuration time limits the period during which I/O modules connected to the CM or CP must be ready for operation. If the central and distributed I/O is not ready for operation within the configuration time, the startup characteristics of the CPU depends on the setting of the "Supported hardware compatibility" parameter.

Group	Properties	Description
Clock	-	Definition for synchronizing the clock via the backplane bus. The setting options depend on the CPU. • As slave: The clock is synchronized by another clock. • As master: The clock synchronizes other clocks. • None: No synchronization takes place. The time intervals for the synchronization can be selected. See: AUTOHOTSPOT
Cycle	Maximum cycle time (cycle monitoring time) and minimum cycle time	Specification of a maximum cycle time (cycle monitoring time) or a fixed minimum cycle time. If the cycle time exceeds the maximum cycle time, the CPU calls organization block OB 80 "time error". If OB 80 is not available, the CPU changes to STOP mode. If the maximum cycle time is exceeded twice in one cycle, the CPU changes to STOP mode even if OB 80 exists. Causes of a violation: • Communication processes • Accumulation of interrupt events • CPU program error. If the cycle time is shorter than the minimum cycle time you have entered, the CPU waits until the minimum cycle time has been reached. The CPU processes the OB 90 (background OB) in this additional program processing time, if it is loaded. See: Cycle time and maximum cycle time (cycle monitoring time) (Page 944)
Communication load	Cycle load due to communication	Controls the duration of communication processes that always also extend the cycle time, within certain limits. Examples of communication processes include: Transferring data to another CPU or loading blocks (initiated via the PC). See: Cycle load due to communication (Page 945)
System and clock memory	System memory bits and clock memory bits	Setting of a byte for the system memory functions and setting of a byte for the clock memory functions (switches each bit on and off at a specified frequency). You use system memory bits for the following queries: • Is the current cycle the first since POWER OFF/POWER ON? • Have there been any diagnostics state changes since the previous cycle? • Query for "1" (high) • Query for "0" (low) Clock memory bits change their values periodically at specified intervals. See: Enabling system memory (Page 969) See: Using clock memory (Page 970)
SIMATIC memory card	-	Enabling of diagnostics for SIMATIC memory card service life.

Group	Properties	Description
System diagnostics	General	System diagnostics is the recording, evaluation and reporting of errors within the automation system. Examples of errors: • CPU program error • Failure of modules • Wire break to sensors and actuators Default alarm texts are available for determining the system diagnostics. If necessary, existing alarm texts can be changed and new ones added. The settings are saved with the project and are effective only after the compilation and loading of the hardware configuration to the relevant components. For S7-1500, the system diagnostics is automatically activated (cannot be deactivated).
PLC alarms	General Central alarm management in the PLC	Activates the central alarm management in the CPU. An alarm is completely assembled in the CPU and then sent to the HMI device. Advantage: There is no need to download the alarm texts to the HMI devices. See: Central alarm management in the CPU (Page 953)

Group	Properties	Description
Web server	General	Activates and configures the web server function. See: AUTOHOTSPOT
	Automatic update	Sends the requested web page with current CPU data periodically to the web browser. Enter the period duration under "Update interval". Automatic update can only be activated if the web server is enabled. See: AUTOHOTSPOT
	User Management	Here you enter the user names with their access rights. • Everyone: Pre-defined user whose access rights you assign using a drop-down list. The "Everyone" user is a user who does not log on with a password for web access. • User#: Editable name for a user for whom you assign access rights and a password. As soon as you click in a line below "Everyone", a preset name is displayed that you can change. Each user can be assigned different access rights via a drop-down list. There are access rights that are connected to each other. Example: When you activate the access right "Perform firmware update", the access rights "Change operating mode" and "... query diagnostics" are activated automatically. These two rights are required to perform a firmware update from the web server. See: AUTOHOTSPOT
	Security	Allows the use of a certificate for the identification of the server.
	Watch tables	Allows you to create watch tables and defines access to those tables.
	User-defined web pages	Allows access to freely-designed web pages of the CPU via a web browser. See: Creating and loading user pages (Page 1257)
	Entry page	Allows you to select the entry page.
	Overview of interfaces	Tabular representation of all modules with their Ethernet interfaces providing web server functionality for this device. Here you can permit or deny access to the web server by means of the respective interface for each Ethernet interface of the device (CPU, CP, CM).
DNS configuration	-	Configuration of the DNS server address. A DNS server may be required if the CPU or a communication partner should be accessible via the host name (FQDN). See: AUTOHOTSPOT

Group	Properties	Description
Display	General	The display remains dark in standby mode and is re-activated as soon as one of the display keys is pressed. You can also change standby mode in the display menu of the display. In energy-saving mode, the display shows the information with reduced brightness. Energy-saving mode is switched off as soon as any display key is pressed. You can also change energy-saving mode in the display menu of the display. Display language: the language is changed immediately after the hardware configuration is loaded with the set standard language. You can also change the language in the display menu of the display.
	Automatic update	Input of the time interval at which the display is updated.
	Password	Setting to enable or disable write-access via the CPU display. If you have enabled write-access, you can also enable display protection. The display is then protected by a configurable password. You can set the time until automatic logoff after input of the password on the display.
	Watch tables	Configured watch tables are provided for selection. You can use the selected watch tables on the display during operation.
	User-defined logo	You can select a user-defined logo and load it to the CPU with the hardware configuration.
Multilingual	-	If you want to use translations of project texts for the Web server/for the CPU display, you need to assign the project language to the languages for the Web server/for the CPU display (number depends on the CPU). With this, you specify the project languages that are displayed when the Web server is accessed or in the CPU display. Example: You assign the project language "German (Luxembourg)", for example, to a German language of the Web server/the CPU display by selecting it from the drop-down list. Enable the project language under "Languages & resources > Project languages" in the project tree to make it available for selection in the drop-down list. See: Multilingual (Page 947)
Time	Local time and daylight saving time	Selection of the time zone in which the CPU operates and setting for standard/daylight saving time. See also: Determining and setting the time of day on a CPU (Page 3334) Setting and reading the time of day (Page 967)
Protection & Security	Access level	Setting of the read/write protection and the password for access to the CPU. See: Setting options for the protection (Page 970) See: Configuring access levels (Page 975)
	Connection mechanisms	Enabling of access via PUT/GET communication.
	Certificate manager	Enabling of global security settings for the certificate manager and management of the certificates for the device.
	Security event	Enabling of group alarms for security events and specification of a monitoring time (interval). See: Summarizing security events (Page 979)

Group	Properties	Description
OPC UA	General	Shows the name of the OPC UA application. This is used for the OPC UA server on this CPU.
	Server	Enabling and configuring of the OPC UA server. Besides the general settings for assignment of port numbers, send interval and sampling interval, you can configure the security settings here and perform the export of the PLC/DB tags available in the OPC UA address space to an XML file. You have the following options for the security settings: • Secure channel: Select between use of global security settings or use of the local CPU-specific certificate manager with limited functionality. Create or select the server certificate here, select the suitable security policy and manage the list of trusted clients. • User authentication: Create or select new users and specify the authentication options.
	Client	Enable the OPC UA client for this CPU.
System power supply	General	Specifies that the CPU is connected to the power grid. This CPU power therefore goes in to the power supply/consumption ratio as positive.
	Power segment overview	In the power segment overview, the power provided by the CPU and the power supply modules is compared with the power required by the signal modules. If the supply/consumption ratio is negative, in other words the power requirements are greater than the power provided by the CPU and the power supply modules, this is indicated in the table.
Configuration control	-	Enabling of the configuration control. Enables a configuration change in the user program within certain limits. See: Configuration control
Connection resources	-	Provides an overview of the assigned reserved and dynamic resources for the CPU connections. The resources currently used are also displayed in the online view. See: Connection resources and communication type (Page 162)
Address overview	-	Table of all addresses used by the CPU for integrated inputs/outputs and for the inserted modules. Addresses that are not used by any module are shown as gaps. The view can be filtered by • Input addresses • Output addresses • Address gaps
Runtime licenses	CPU applications that require an RT license, e.g. OPC UA, ProDiag, Energy Suite	Depending on the CPU application, select the number of licenses to be used here. Purchasing these licenses is a requirement for their use. Exception: If you use these CPU applications in the context of simulations with PLCSim Advanced, it is not necessary to purchase licenses. In this case, simply select the number of licenses you require. Example of license selection: Setting ProDiag licenses

See also

Assigning the device name and IP address (Page 2842)

CPU settings on the display (Page 944)

Rules for CPUs with several PROFINET interfaces (Page 955)

IEEE-compliant LLDP (Page 959)

CPU settings on the display

The S7-1500 CPUs have a display with operator control buttons. The display can be used to read out status information about the configured modules, for diagnostic purposes, and also to adapt some CPU settings.

Example: Setting the IP address and the PROFINET device name on the display

If you want to assign the IP address parameters (IP address, subnet mask and router settings) on the display, you must select the following option in the relevant project for the PROFINET interface of the CPU:

- Set IP address on the device

The same applies accordingly to the PROFINET device name: If you want to assign the PROFINET device name on the display, then you must select the following option in the relevant project for the PROFINET interface of the CPU:

- Set PROFINET device name on the device

Cycle time and maximum cycle time (cycle monitoring time)

Function

The cycle time is the time that the operating system requires to execute the cyclic program and all the program sections that interrupt this cycle. The program execution can be interrupted by:

- Time errors and 2xMaxCycleTime errors
- System activities, e.g., process image updating

The cycle time (Tcyc) is therefore not the same for every cycle.

The following schematic shows an example of different cycle times (TZ1 ≠ TZ2) for S7-1500 CPUs:

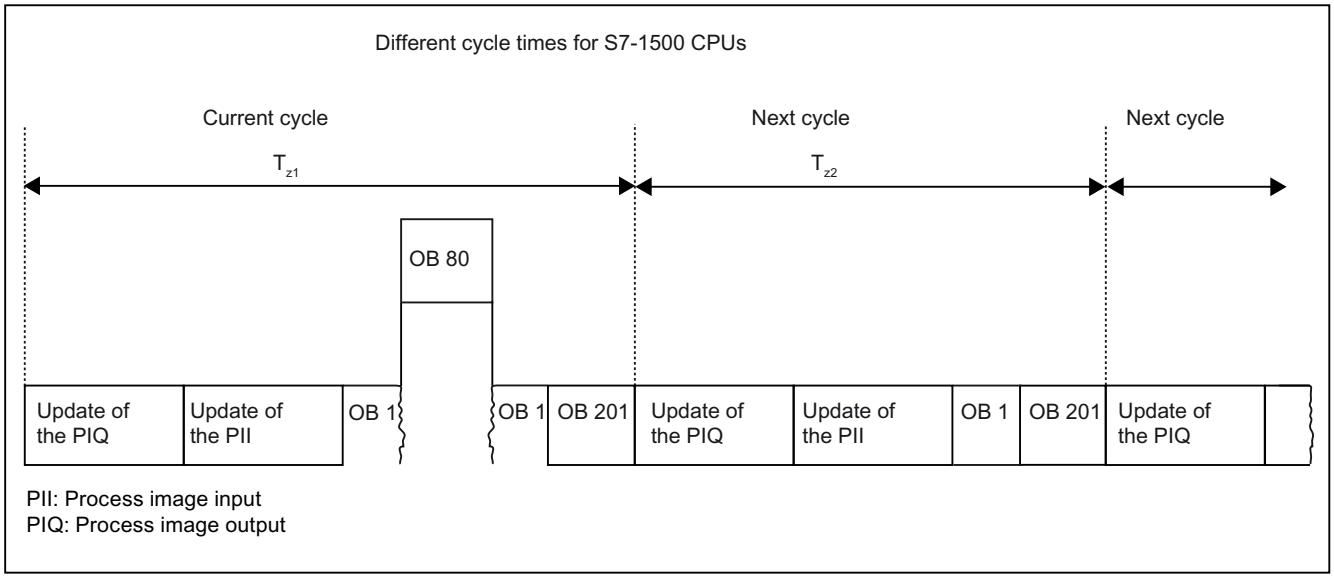

In the current cycle, the cyclic OB used here (e.g. OB 1) will be interrupted by a time error (e.g. OB 80). Following the cyclic OB, the next cycle OB 201 is processed.

Maximum cycle time (cycle monitoring time)

The operating system monitors the execution time of the cyclic program for a configurable upper limit known as the maximum cycle time. You can restart this time monitoring at any point in your program by calling the RE_TRIGR instruction.

If the cyclic program exceeds the maximum cycle time, the operating system attempts to start the time error OB (OB 80). If the OB is not present, the CPU switches to STOP mode.

In addition to monitoring the runtime for overshooting of the maximum cycle time, adherence to a minimum cycle time is guaranteed. To do this, the operating system delays the start of the new cycle until the minimum cycle time has been reached. During this waiting time, new events and operating system services are processed.

If the maximum cycle time is exceeded a second time, for example while the time error OB is being processed (2xMaxCycleTime error), the CPU changes to STOP mode.

Cycle load due to communication

Function

The cycle time of the CPU can be extended due to communication processes. These communication processes include for example:

- Transferring data to another CPU
- Loading of blocks initiated by a programming device
- Certain motion control functions

You can control the duration of these communication processes to some extent using the CPU parameter "Cycle load due to communication".

In addition to communication processes, test functions also extend the cycle time. The "Cycle load due to communication" parameter can be used to influence the duration.

The motion control functions commissioning of technology objects, restart of technology objects and interpolation of cam discs are also processed as communications processes. While these functions are being processed, they extend the cycle time as communications load.

How the parameter works

You use the "Cycle load due to communication" parameter to enter the percentage of the overall CPU processing capacity that can be available for communication processes.

This processing capacity can be used for program execution when not required for communication.

Note

Which effects can the specification of a minimum cycle time have?

When you have specified a minimum cycle time and the user program does not need it completely, the remaining time can be used once again for communication processes. The result may be that communication processes take up a larger percentage of the CPU processing capacity than you have specified in the CPU parameter "Cycle load due to communication".

For the priority classes **lower than or equal to 15**, the processing power for communication is continuously assigned by the CPU.

Priority classes **greater than 15** are not by interrupted by communication.

By assigning the priority of an event or an OB, you thus influence the interruptibility of the OB and the blocks called by the OB. If program sections are made uninterruptible in this manner in order to minimize cycle times, the online functions of STEP 7 are delayed!

Effect on the actual cycle time

The "Cycle load due to communication" parameter can be used to extend the cycle time of the cyclic organization block (e.g., OB 1) by a factor calculated according to the following formula:

$$\frac{100}{100 - \text{"Cycle load due to communication"}}$$

The formula does not take into account the effect of asynchronous events such as hardware interrupts or cyclic interrupts on the cycle time.

If the cycle time is extended due to communication processes, more asynchronous events may occur within the cycle time of the cyclic organization block. This extends the cycle still further. The extension depends on how many events occur and how long it takes to process them.

Example 1 – no additional asynchronous events:

If the "Cycle load due to communication" parameter is set to 50%, this can cause the cycle time of the cyclic organization block to increase by up to a factor of 2.

Example 2 – additional asynchronous events:

For a pure cycle time of 500 ms, a communication load of 50% can result in an actual cycle time of up to 1000 ms, provided that the CPU always has enough communication jobs to process.

If, parallel to this, a cyclic interrupt (priority class lower than or equal to 15) with 20 ms processing time is executed every 100 ms, this cyclic interrupt would extend the cycle by a total of 5*20 ms = 100 ms without communication load. In other words, the actual cycle time would be 600 ms.

Since communication at 50% communication load extends the time required for both the cyclic program and the cyclic interrupt, the actual cycle time amounts to 1200 ms instead of 1000 ms.

If the cyclic interrupt has a priority greater than 15, the processing time of the cyclic interrupt OB is not extended by communication, but the actual cycle time remains the same (1200 ms).

Note

Observe the following:

- Check the effects of changing the value of the "Cycle load due to communication" parameter while the system is running.
- You must always consider the communication load when setting the minimum cycle time; otherwise, time errors occur.

Recommendations

- Increase this value only if the CPU is used primarily for communication purposes and the user program is not time critical.
- In all other situations you should only reduce this value.
- If you do not want interrupt OBs to be interrupted by communication, you must set their priority to a value greater than 15.

Multilingual

When you configure a CPU, texts of different categories result, e.g.

- Object names (names of blocks, modules, tags, ...)
- Comments (for blocks, networks, watch tables, ...)
- Alarm texts and diagnostics texts

Texts are made available by the system (e.g. diagnostics buffer texts) or are created during the configuration (e.g. alarms).

Depending on the specific element, texts are available in one language or they can be translated into other languages and can then be present in the project in multiple languages. You can maintain project texts in all languages available to you in the project tree (Languages & resources > Project texts).

See: Basics of project texts.

The texts produced during the configuration can be loaded onto the CPU. Due to the limited resources (e.g. load memory) only a limited number of languages can be loaded.

The following description shows the uses of multilingual texts on the CPU, which texts are loaded and how you decide in which languages the texts are loaded onto the CPU.

You will also learn the influence the language assignment that you assign in the CPU properties (multilanguage) has and how this setting affects operation.

Uses of multilingual project texts

As the creator of the project, you create texts such as comments and alarms in your native language. For an original equipment manufacturer (OEM customer), you translate the texts into the language of the OEM customer, for example English, and into the language of the end customer. You can load a maximum of three languages onto the CPU.

The customers have the advantage that the relevant texts e.g. comments and alarms, exist in their native language even when the STEP 7 project is not available on the PG/PC. Loading the device as a new station also loads the languages, and the project can be analyzed and adapted on site.

A further advantage is that certain multilingual texts are available to an operator even without STEP 7.

Using the language selection of the Web server or the language selection of the CPU display, for example alarms, diagnostics buffer entries and status messages about the module status are visible in the set language.

Multilingual texts can also be displayed on HMI devices. You can display tag comments and step comments, for example, in the loaded languages in the Graph or Code viewer.

Requirements

- The project languages that are to be assigned must be activated.
- The corresponding texts (translations) must be available in the project.
  The project language selection can be found in the project tree under "Languages & Resources".

Setting the available languages for CPU display and Web server

The settings for multiple languages can be found in the CPU parameters: You set the following here:

- Which project languages are assigned to the languages of the CPU display/the Web server
- Which project languages are loaded onto the CPU

Follow these steps:

1. Navigate to the "Multiple languages" area ("Properties > General > Multiple languages") with selected CPU in the Inspector window.
   This area contains a table in which you assign languages: The project languages can be selected in the left-hand column. In the right-hand column, all the languages are listed that can be selected on the CPU display or for the Web server.
2. Assign a project language to the languages for the CPU display/Web server.
   With these assignments, existing project languages (as replacement languages) are displayed for languages that do not exist on the CPU.
   Note that these language settings only relate to the texts loaded from the configuration. Both the Web server and the CPU display have many more user interface languages so that you can have static texts displayed in the Web browser or on the CPU display in many more languages.

Note

The number of loadable languages is limited for specific CPUs. If, for example, two languages are possible, assign the corresponding project language to your native language in the "Languages of the CPU display/Web server" column. In the remaining rows, you assign the same substitute language in the languages of the CPU display/Web server.

Example

Existing project languages in STEP 7 (PG): English, French, German, Italian.

Due to the assignment in the "Multilingual support" area of the CPU parameters, the following project languages are loaded during the next load procedure.

German, English.

The project language German is set automatically in display language German. The project language English is displayed with all other European display languages.

No language is assigned to the other display languages.

On the display, you can only select the language for the display; due to the language assignment from the configuration, the assigned project language is set automatically with the language for the display:

- If you select English as the language for the display, the project language English is set.
- If you select French as the language for the display, the project language English is also set.
- If you select Chinese as the language for the display, no project texts are displayed.

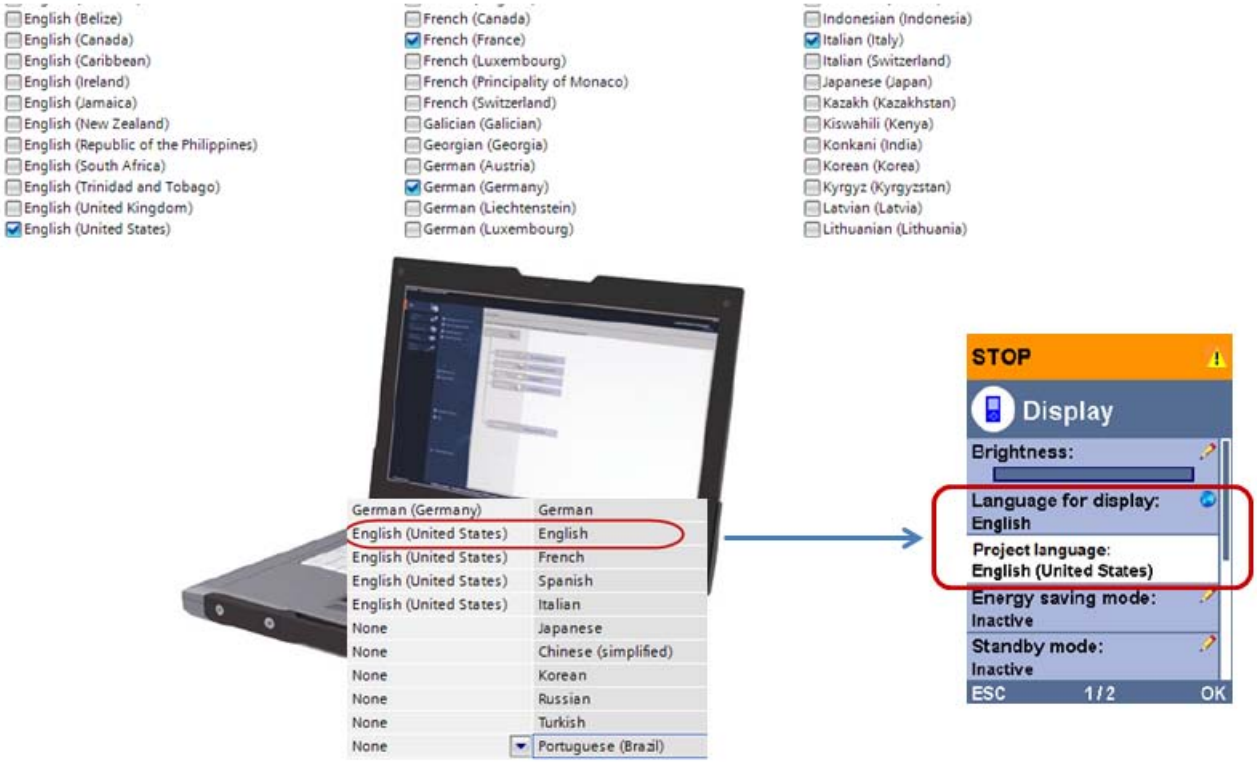

Which project texts are loaded with the selected languages?

The following texts are loaded in the selected languages with the project data onto the CPU and used by the Web server/CPU display:

- Diagnostics buffer texts (cannot be changed)
- Status texts for the module status (cannot be changed)
- Alarm texts with corresponding text lists
- Tag comments and step comments for Graph and PLC Code Viewer
- Comments in watch tables

The following texts are also loaded in the selected languages with the project languages on the CPU but are not used by the Web server/CPU display:

- Comments in tag tables (for tags and constants)
- Comments in global data blocks
- Comments of elements in block interfaces of FBs, FCs, DBs and UDTs
- Network titles in blocks written in LAD, FBD or STL
- Block comments
- Network comments
- Comments of LAD and FBD elements

The following figure once again shows the relationship between multilingual project texts and the required memory space on a SIMATIC memory card.

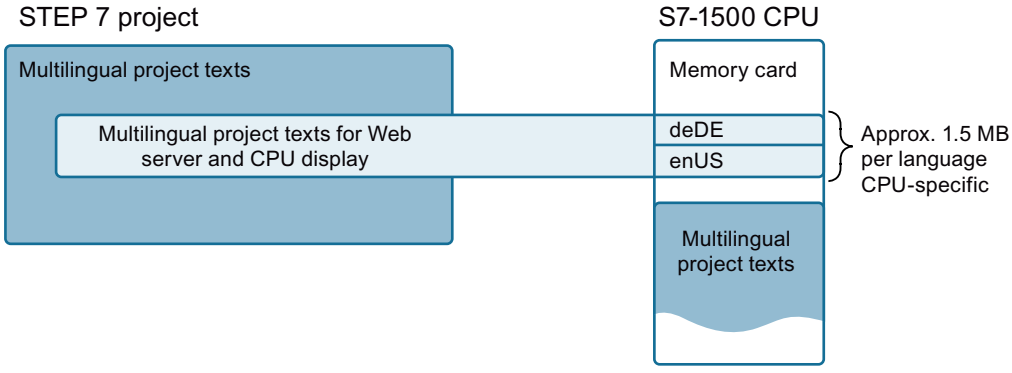

Special considerations: Module comments

Module comments are only loaded in one language.

The following applies to loading module comments:

- Requirement: In the project settings (Options > Settings "Hardware configuration" area), the option "Download module comment" must be enabled (default).
- Module comments are loaded in the language set as the editing language. You change the editing languages in "Languages & Resources > Project languages".

Special considerations: Diagnostics buffer texts and status texts for the module status

The predefined texts for diagnostics buffer entries are not available in all languages that you can select as project languages in the languages & resources in STEP 7. With the assignment between the project language and language for the Web server/CPU display, the "closest" language is therefore loaded to the CPU.

Example: The texts mentioned above are in the German language, there is only one German language variant. If German (Luxemburg) or German (Austria) is selected as project language, the language German is always loaded.

Adding or removing languages - consequences for loading onto the CPU

If the number of project languages for the CPU display/Web server has changed, loading must be performed in STOP mode.

In this case, STEP 7 runs a complete load operation (hardware and software).

This applies regardless of the objects you have selected for loading: Loading a block, normally possible in RUN, also requires a CPU STOP whenever the number of languages for the CPU display/Web server has changed since the last load operation.

Even if you deselect an added project language again, a CPU STOP is required for the loading. You can only load the CPU in RUN when you undo the selection of an additional project language with the "Undo" command in the "Edit" menu and have not changed anything else that makes a STOP necessary.

In principle, the following applies:

Before each load operation, STEP 7 checks whether the number of languages for the CPU display/Web server has changed. If the number has changed (fewer or more languages selected since the last load operation), a CPU STOP is necessary.

Special features when loading the device as a new station (hardware and software)

If you have established an online connection to a device and you load this device (CPU with all assigned modules) into your project, the following rules apply:

- The project languages in the project are updated; in other words, when additional project languages exist on the device, they also exist in the project after loading (enabled in "Languages & Resources" in the project tree). The "Offline and online texts" are also compared after the loading; in other words, the version of the texts in the project corresponds to the version on the device.
- The project languages already existing in the project remain unaffected by the loading operation, no project language is deleted.
- After loading, the assignments "Project languages of the CPU/Web server" (multilingual) in the project correspond to the assignments loaded on the CPU. Loading to the device again loads the same languages that exist on the CPU.

Special features when loading from the device (software)

If you load the software from a CPU in your project, for example individual blocks or the entire program, the following rules apply:

- The project languages in the project are updated; in other words, when additional project languages exist on the device, they also exist in the project after loading (enabled in "Languages & Resources" in the project tree). The "Offline and online texts" of the loaded components are compared after the loading; in other words, the version of the texts in the project corresponds to the version on the device.
- The assignments "Project language - languages of the CPU display/Web server" (multilingual) in the project **are retained**; in other words, they do not change even if, for example, a new project language was used in the loaded block.
  Note that in this case (new language only available in the CPU accessible online), you need to adapt the settings under "Multilingual support" in the properties of the CPU if the new language is to be loaded again to the CPU.
- There is a risk that existing translations in the project may be **deleted** when loading from the device in the following situation:
  - A language exists offline (e.g. in a block in the project)
  - This language does **not** exist online
    In this case, when you load from the device, you lose the language that exists online.
- Team engineering: You should only change the settings for multiple languages in the master project.

Special features with team engineering (joint commissioning)

In the case of joint commissioning, you have the option of recognizing missing translations with the synchronization dialog.

Recognizing missing languages during the online/offline comparison

In the online/offline comparison (comparison editor), you can recognize differences in the existing project languages for the following objects:

- Blocks
- Tag tables

Differences in the existing project languages for alarms and text lists are not taken into account by the comparison editor.

If know-how protected blocks are also included in the comparison: It is not necessary to enter passwords for these blocks for the comparison.

The following are used for the comparison:

- Online: The languages that actually exist on the CPU
- Offline: The languages selected in the "Assigned project languages" table in the "Multilingual support" area and therefore also taken into account when loading onto the CPU. These are not the languages in which, for example, the comments of the blocks actually exist.

The offline/offline comparison compares the activated project languages.

See also

Overview of the CPU properties (Page 935)

Central alarm management in the CPU

S7-1500 CPUs with firmware version V2.0 or higher support central alarm management.

The CPU parameter "Central alarm management in the PLC" is available under "PLC alarms > General".

Default: Central alarm management is enabled.

Effect of the CPU parameter "Central alarm management"

When central alarm management is enabled, the CPU transfers the entire alarm text, which can consist of multiple text list entries, to the HMI device. The CPU is responsible for compiling the entire alarm text. As soon as the triggering event for an alarm is pending, the CPU sends the entire alarm text with associated values, if any, to the HMI device through the existing HMI connection.

The enabled central alarm management makes it unnecessary to download the texts of the connected HMI devices.

If the central alarm management is **not** enabled, you must download the texts to the HMI device via the engineering system (WinCC) so that the alarm texts are displayed during operation.

When the triggering event for an alarm is pending, the CPU sends only rudimentary information to the HMI device (e.g. alarm ID, alarm version and associated values). The HMI device is responsible for alarm management: The complete alarm text is compiled and prepared from the downloaded and transmitted data. This task does not require computing power of the CPU.

Any change to the message texts requires renewed downloading to the connected HM devices.

Notes and rules for central alarm management

- Required version of HMI Runtime: V14 or higher
- Required CPU: S7-1500 as of firmware version 2.0 (also S7-1500SP CPU, S7-1500Pro CPU and S7-1500 software controller).
- Alarms for which central alarm management is possible: Controller alarms (system diagnostics, process diagnostics, program alarms).
- "Automatic update" must be activated in the runtime settings of the HMI device ("Control alarms" section in the "Alarms" area).
- Modified texts that you load to the CPU have no effect on alarm texts that were already transferred to the HMI device. The updated alarm text will only be displayed with a new triggering event.
- The following applies to alarms that are displayed in the alarm view or alarm buffer of the HMI device:
  - You must download the languages in which the alarms are to be displayed on the HMI device to the CPU (max. 3 languages, see Multilingual (Page 947)).
  - The languages on the HMI device must be the languages that were loaded to the CPU. Otherwise the HMI device only indicates that alarm texts are missing during operation. For more than three languages, you must load the HMI device with the texts (central alarm management is not possible).
- The following applies to alarms that are displayed in the alarm archive:
  - The alarm archive includes alarms in one language only. This language is set during configuration of the HMI device under "Archives > General > Archive language". The archive language also has to be one of the languages that was downloaded to the CPU.

Note

Display of faults

If faults occur, a text is displayed as follows: #TextlistID, TextID (e.g. #123, 456)

Example: You have forgotten an entry (Range) in the used text list. Then the first number is the text list ID (column is grayed out by default, but can be made visible). The second number shows the value for which the entry is missing. Text list ID = 123, value = 456.

If Range is defined without text, this does not generate a fault.

See also

Overview of the CPU properties (Page 935)

CPUs with several PROFINET interfaces

Rules for CPUs with several PROFINET interfaces

The S7-1500 family of CPUs includes CPUs with two or three PROFINET interfaces, for example, CPU 1516-3 PN/DP.

One PROFINET interface can be used for PROFINET IO communication while the others are being used for communication with a higher-level network (backbone, router or Internet) or for communication with a different machine or automation cell, for example.

Please note that some settings can be changed only at interface X1 and are read-only at the other interfaces (for example, X2), see below.

CPUs as of firmware V2.0 with multiple PN IO interfaces

Central processing units as of CPU 1515 with firmware version V2.0 and higher have two PROFINET IO interfaces:

- you can operate two separate PROFINET IO systems as each of the PN IO interfaces supports the IO controller function.
- The PROFINET interface X1 supports real-time classes IRT and RT, and PROFINET interface X2 supports RT.
- Both PROFINET interfaces support the I-device function.
- The 2nd or the 3rd PROFINET interface has restrictions compared to the 1st PROFINET interface (X1):
  - Establishment of redundant ring structures not possible
  - IRT not possible (RT at the PROFINET interface X2 possible, but selection of the send clocks and the number of connectable IO devices restricted)

  Details about the performance capability of the individual interfaces are available in the respective manuals for the individual CPUs.

The following section shows how two CPUs operate their machine networks at the PROFINET interface X1. Operation of the machine networks also works with isochronous real time (IRT).

For coordination purposes, each PROFINET X2 interface is connected to a plant network that is separate from the local machine network: multiple "Sub_Machines" are connected to a "Main_Machine" with the I-device function.

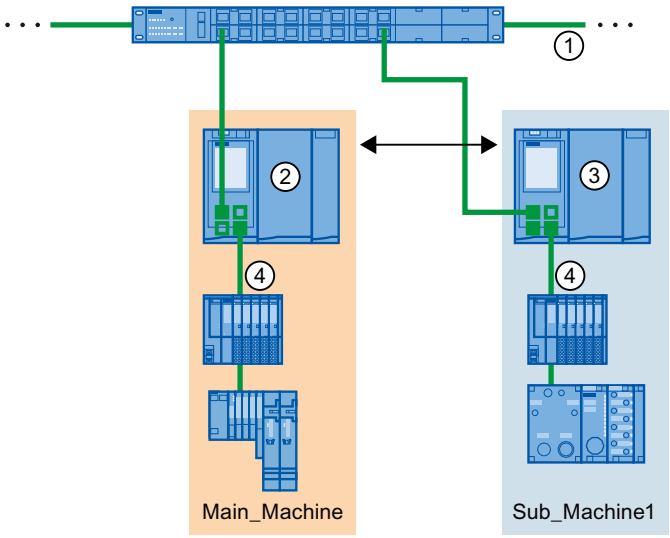

① Plant network

② The CPU is both the IO controller in the plant network (interface X2) and the IO controller in the machine network (interface X1). The machine network can be operated with IRT.

③ The CPU is both the I-device in the plant network (interface X2) and the IO controller in the machine network (interface X1).

④ Machine network

Rules

The following applies to an S7-1500 CPU with several PROFINET interfaces:

- For CPUs with multiple PROFINET interfaces **with firmware version V2.0 or higher**, the second PROFINET interface (X2) also supports IO functionality with real-time class RT, in other words the roles of IO controller and IO device.
  In addition, the second and all further PROFINET interfaces (for example X2) supports HMI communication, communication with the configuration system, communication with a higher-level network (backbone, router, Internet) and communication with another machine or automation cell. In addition, the following is supported: Send topology via LLDP protocol, manage LLDP MIBs, reply to and manage SNMP requests as well as the "Accessible devices" function (DCP protocol).
- The following applies for CPUs with **firmware versions < V2.0**: The second or all other PROFINET interfaces (for example X2) do not support any IO functionality, meaning no IO controller/IO device role. HMI communication, communication with the configuration system, communication with a higher-level network (backbone, router, Internet) and communication with another machine or automation cell are supported. In addition, the following is supported: Send topology via LLDP protocol, manage LLDP MIBs, reply to and manage SNMP requests as well as the "Accessible devices" function (DCP protocol).

- The IP subnets of all interfaces must be different. This means that the subnet part of all interface IP addresses must be different.
  Advantages regarding communication across several PROFINET interfaces only apply when the interfaces are in different IP subnets.
  The subnet mask which you also configure during the IP addressing co-determines whether the IP subnets of the two interfaces are different or not.
  When the CPU addresses an Ethernet station, it decides over which interface the respective data packages are sent or received based on the rules described in RFC 4632.
  Example: Both IP addresses are in different subnets
  - Subnet mask of both interfaces: 255.255.255.0
  - IP address of the first interface: 192.168.**0**.10
  - IP address of the second interface: 192.168.**1**.10

  Negative example: Both IP addresses are in the same subnet
  - Subnet mask of both interfaces: 255.255.0.0
  - IP address of the first interface: 192.168.0.10
  - IP address of the second interface: 192.168.1.10
- The interfaces can be connected to the same physical network (a physical network with two logical subnets, which means with two IPv4 networks.)
- Only one router may be effective at any time. The IP address of the router must be located in the IP subnet of the PROFINET interface on which the router is configured.
- The name of the interface and the name of the device (CPU) are used in the automatic assignment of the device name. Example: plc_1.profinet interface_1
- Time-of-day synchronization: The activation of the synchronization and the settings for the NTP servers can only be made on the PROFINET interface X1. The settings are applied for the interface X2 and additional interfaces (read-only there).
- Keep-alive option: For the activation of the advanced option "Send keepalives for connections ... ", the same applies as for the time-of-day synchronization: Can be activated only at interface X1, read-only at interface X2 and additional interfaces.

Go online

If you want to use a PG/PC with STEP 7 to access a CPU with two or three PROFINET interfaces online, the following activities then run in the background:

- While establishing the online connection, STEP 7 checks whether one of the PROFINET interfaces connected to the S7 subnet is located in the same IP subnet as the IP address (= IP address of the online access of the PG/PC) and then uses this IP address/PROFINET interface to go online.
- If none of the PROFINET interfaces is located in the same IP subnet, STEP 7 goes online via the PROFINET interfaces for which a router is configured.
- If these requirements are not met, STEP 7 prompts the user to change the IP subnet address of the PG/PC interface so that the PG lies in the same IP subnet.

Limitation of the data infeed into the network

Limiting the data infeed into the network on PROFINET interfaces

The "Limit data infeed into the network" function limits the network load of standard Ethernet communication which is fed into the network by the interface to a maximum value. This does not apply to cyclic real-time communication (RT/IRT).

With a line topology, critical network loads can occur quickly with standard Ethernet communication. Devices in a line topology should support the limitation of the data infeed into the network.

Devices that feed in a lot of "Standard Ethernet communication" to the network should not be positioned in a line topology but should be placed directly behind a switch.

In line with the interface, you can enable or disable the function "Limit data infeed into the network". If you use the X1 interface of an S7-1500 CPU as the IO controller or I-device, the "Limit data infeed into the network" function is always enabled. If you are not using the X1 interface of an S7-1500 CPU as IO controller or I-device, you can enable or disable the function. For other interfaces, you can always enable or disable the "Limit data infeed into the network" function.

Uses of the limitation of the data infeed to the network

- Fair division of the remaining bandwidth:
  In PROFINET networks, cyclic and non-cyclic communication share the same network. This means that only a limited bandwidth remains for non-cyclic communication. The limitation of the data infeed ensures that the remaining bandwidth for non-cyclic communication is divided fairly between the devices in the network. The limitation of the data infeed does not solve any global overload problems in the network but it avoids an individual device occupying the entire bandwidth.
- Smoothing peaks in the data infeed:
  The limitation of the data infeed smoothes peak loads of non-cyclic data (e.g. from Open User Communication, access by the Web server).
- Limitation of problems in data infeed:
  Devices are not permitted to feed too much network load into a PROFINET network. If applications in a device generate too much data, this data is not forwarded into the PROFINET network. Negative effects (e.g. data loss, communication breakdown) remain limited to the device and its communications partner. Other nodes are not disturbed.

Setting limitation of the data infeed into the network for a CPU

To set the limitation of the data infeed into the network, follow these steps:

1. In the network view of STEP 7, select the interface of the CPU.
2. In the Inspector window, go to "Properties" > "General" > "Advanced options" > "Interface options".
3. Select or clear the "Limit data infeed into the network" check box.

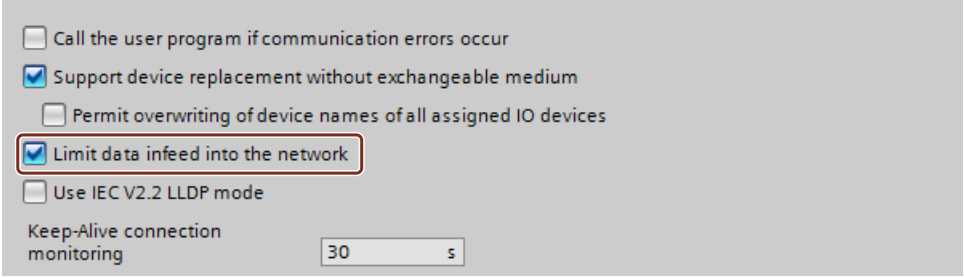

IEEE-compliant LLDP

LLDP stands for "Link Layer Discovery Protocol"; a manufacturer-independent protocol that is defined in the standard IEEE-802.1AB.

Ethernet devices use LLDP to send information about themselves to their neighboring devices in regular intervals. The neighboring devices save this information.

SNMP tools read this data and use it to create the network topology (SNMP stands for "Simple Network Management Protocol").

LLDP and IEC 61158

The fieldbus standard IEC 61158 V2.2, or "IEC V2.2" for short, implements LLDP for devices with PROFINET interface. SNMP tools use it to recognize the topology of Ethernet networks to which devices with PROFINET interface are connected.

Devices with several PROFINET interfaces require a new version of this standard, IEC V2.3. Siemens tools and third-party tools use it to detect all interfaces of a device with several PROFINET interfaces.

All networked PROFINET interfaces must be set to the same mode (either IEC V2.3 or IEC V2.2).

When you configure all devices in the PROFINET subnet in one project, STEP 7 automatically sets the correct mode and you do not have to worry about the settings. Below you will find out about the rules STEP 7 uses to automatically make the settings and the cases in which you have to set the mode manually.

How do you determine which mode is supported by a PROFINET interface?

You can see which mode is supported by a PROFINET interface in the properties of the PROFINET interface for a selected device or selected interface in the device view or network view. You can find the option "Use IEC V2.2 LLDP mode" in the area "PROFINET interface > Advanced options".

- If the option "Use IEC V2.2 LLDP mode" is selected and cannot be changed, the PROFINET interface only supports V2.2 mode.
- If the option "Use IEC V2.2 LLDP mode" is disabled and can be changed, the PROFINET interface supports V2.2 mode as well as V2.3 mode.

Automatic setting of the PROFINET interfaces

STEP 7 sets the mode of the PROFINET interfaces according to the following rules:

- If the devices support IEC V2.3 mode, STEP 7 sets all PROFINET interfaces on the subnet consistently to mode IEC V2.3.
- If only one device supports IEC V2.2 mode, STEP 7 sets all PROFINET interfaces on the subnet to IEC V2.2 mode.

Example 1: Automatic setting

In this example, an I-device was created in project 1 and then inserted in project 2 as GSD file (two projects).

Only one device (3) in project 1 supports mode V2.2. STEP 7 therefore set all PROFINET interfaces in this project to mode V2.2 and noted this setting in the GSD file.

The I-device project was then exported in a GSD file (2) and inserted into project 2.

STEP 7 automatically set all PROFINET interfaces in project 2 also to mode V2.2 due to the entry in the GSD file.

In this example, the higher-level IO controller configures the lower-level IO system (1).

SNMP tools detect only one interface for each device in this example. Tools from Siemens detect all PROFINET interfaces.

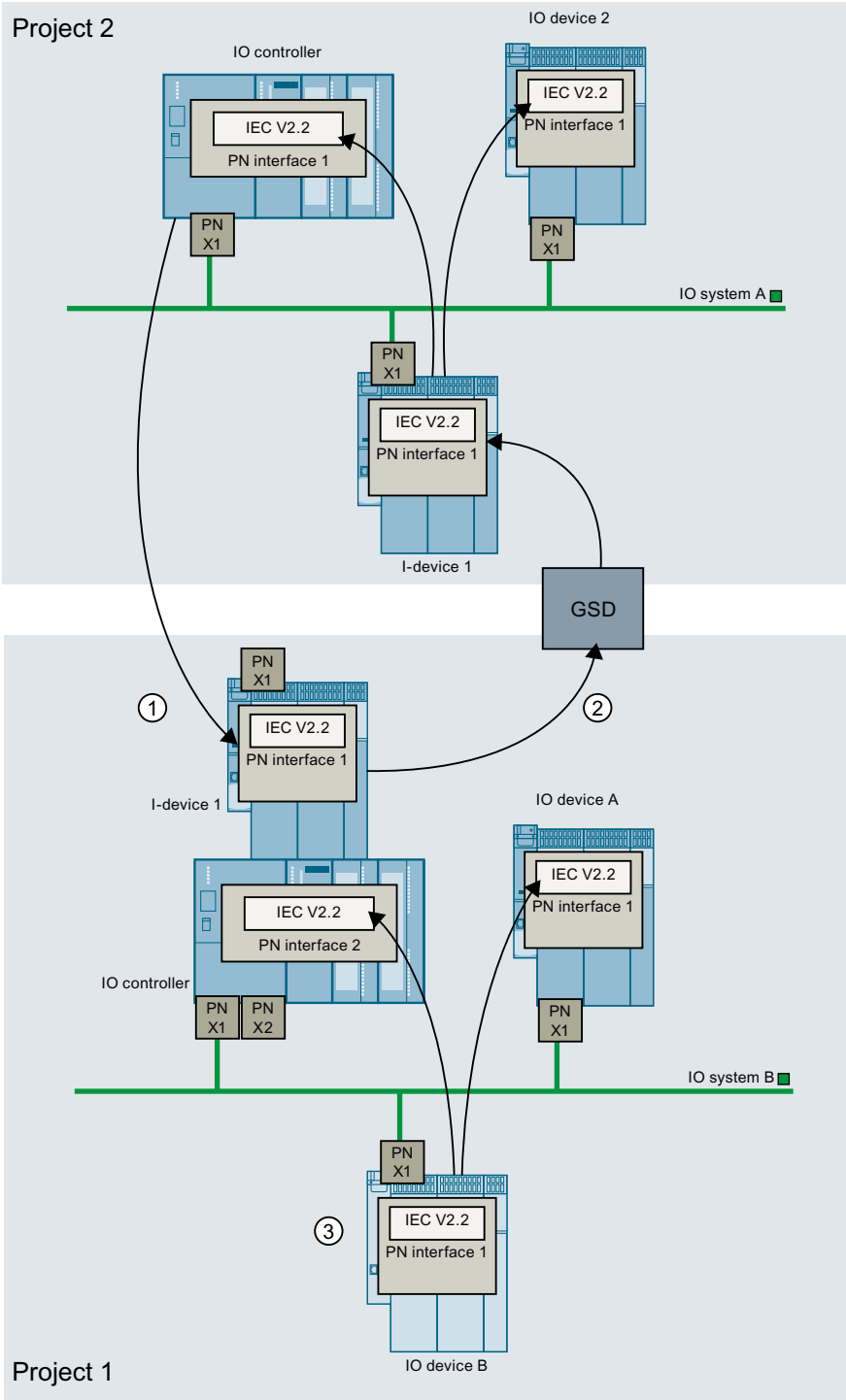

① Higher-level IO controller configures the PROFINET interface of the I-device in the IO system A.

② PROFINET GSD file has entry that only IEC V2.2 mode is supported.

③ Devices with IEC V2.2 mode force IEC V2.2 mode for all PROFINET interfaces in project 1.

Manual setting of the PROFINET interfaces

The manual setting of a PROFINET interface is required in the following cases:

- You configure a system in two or more projects. A device that only supports V2.2 mode is used in one of the projects (project 1). Only devices in V2.3 mode are used in another project (project 2). In this case, you must set one of the devices in project 2 manually to V2.2 mode.
- You are using two projects. The first project includes an I-device. The second project includes a higher-level IO system in which the I-device is integrated. This case is explained in greater detail below:

Setting the IEC mode manually

If the following conditions apply, you must manually set a PROFINET interface to IEC V2.2 mode:

- You configure an I-device and export the project as GSD file.
- You use the GSD file in another project
- You insert the I-device into a higher-level IO system.
- The project with the I-device only uses devices which support V2.3 mode. This mode is set automatically.
- The higher-level IO system includes a device which only supports the standard IEC V2.2.

You force a PROFINET interface of a device in the lower-level IO system into V2.2 mode by executing the following steps.

1. Open the I-device project in STEP 7.
2. Select any device.
3. Enable the option "Use IEC V2.2 LLDP mode" in the area "Interface options".

Then, generate the GSD file once again. Use the new GSD file in the higher-level IO system.

Download the new I-device configuration to the I-device CPU.

Example 2: Setting the IEC mode manually

Project 1: You have created an I-device in the project.

All PROFINET interfaces support V2.3 mode.

This means STEP 7 did not change the mode of the PROFINET interfaces and noted mode V2.3 in the GSD file (2).

Project 2: I-device from project 1 is inserted (GSD file).

One device only supports V2.2 mode.

This means STEP 7 sets all PROFINET interfaces in this project to V2.2 mode.

The PROFINET interfaces in project 1 and 2 have different settings.

This means you have to open the I-device project (project 1) in STEP 7 again and manually set a PROFINET interface to V2.2 mode.

You set a PROFINET interface of a device in the lower-level IO system to V2.2 mode by executing the following steps.

1. Open the I-device project in STEP 7.
2. Select any device.
3. Enable the option "Use IEC V2.2 LLDP mode" in the area "Interface options".

Then, generate a new GSD file. Insert the new GSD file in the higher-level IO system (project 2).

Download the project to the I-device CPU once again.

SNMP tools detect only one interface for each device in this example. Tools from Siemens detect all PROFINET interfaces.

If you do not change the setting of the PROFINET interface, each interface sends a diagnostics alarm during runtime (during operation).

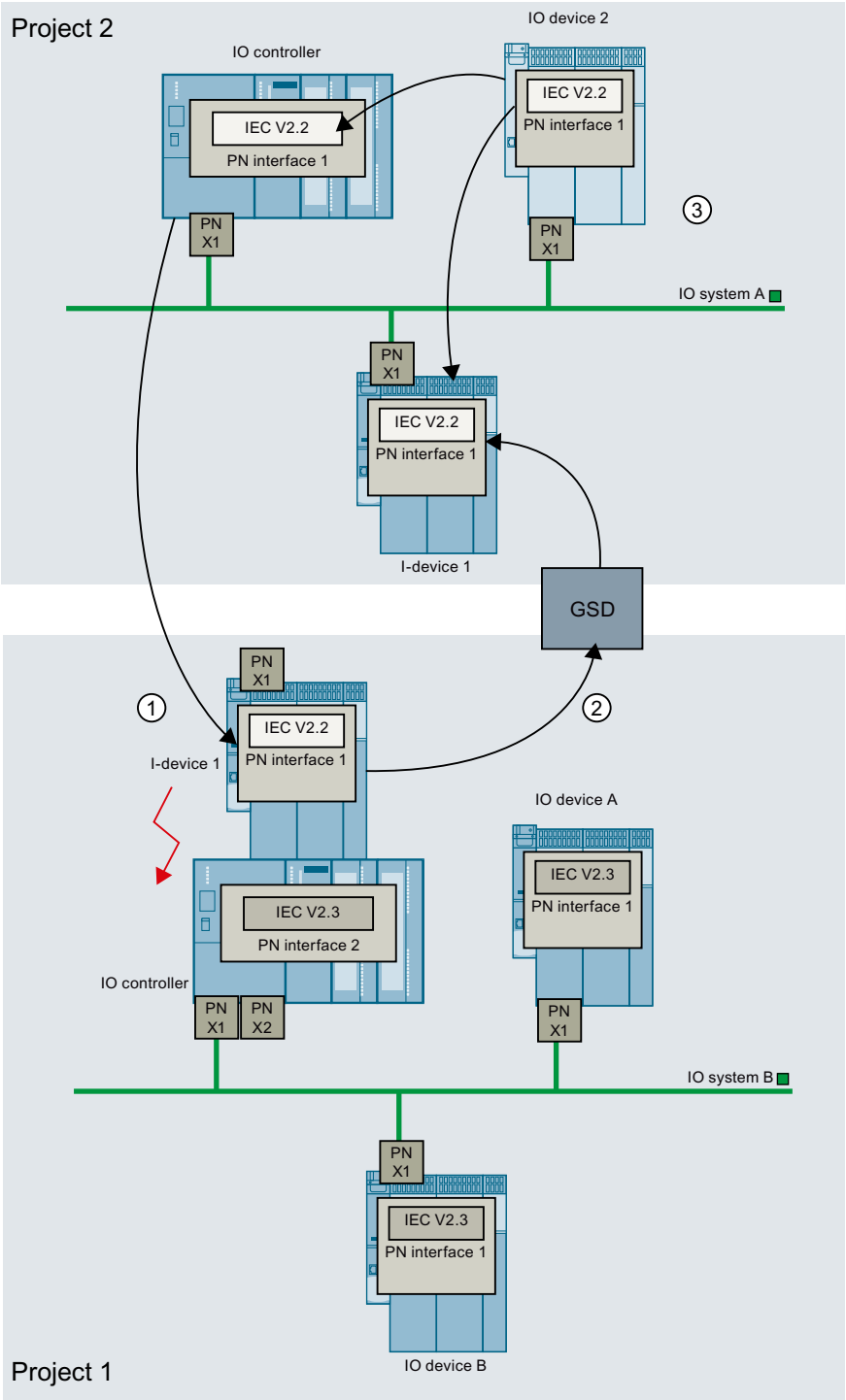

① The PROFINET interfaces of the I-device have different settings (the PROFINET interface in IO system A is set to V2.2 mode; the PROFINET interface in the IO system B is set to V2.3 mode).

② GSD file has entry that the I-device only supports IEC V2.3 mode.

③ IO device 2 only supports V2.2 mode. This means STEP 7 sets all other PROFINET interfaces in project 2 to this mode (including the PROFINET interface of the I-device in IO system A).

Example 3: Setting the IEC mode manually

This example is largely the same as example 2.

Only difference: The settings of the PROFINET interfaces are stored in the I-device (no parameter assignment by the higher-level IO controller as in example 2).

The PROFINET interfaces have different settings in IO system A (I-device set to V2.3, IO controller set to V2.2, IO device 2 only supports V2.2).

This means you must set a PROFINET interface in IO system B to V2.2 mode.

You set a PROFINET interface of a device in the lower-level IO system to V2.2 mode by executing the following steps.

1. Open the I-device project in STEP 7.
2. Select any device.
3. Enable the option "Use IEC V2.2 LLDP mode" in the area "Interface options".

Then generate a new GSD file and use the new GSD file.

Download the new I-device configuration to the I-device CPU.

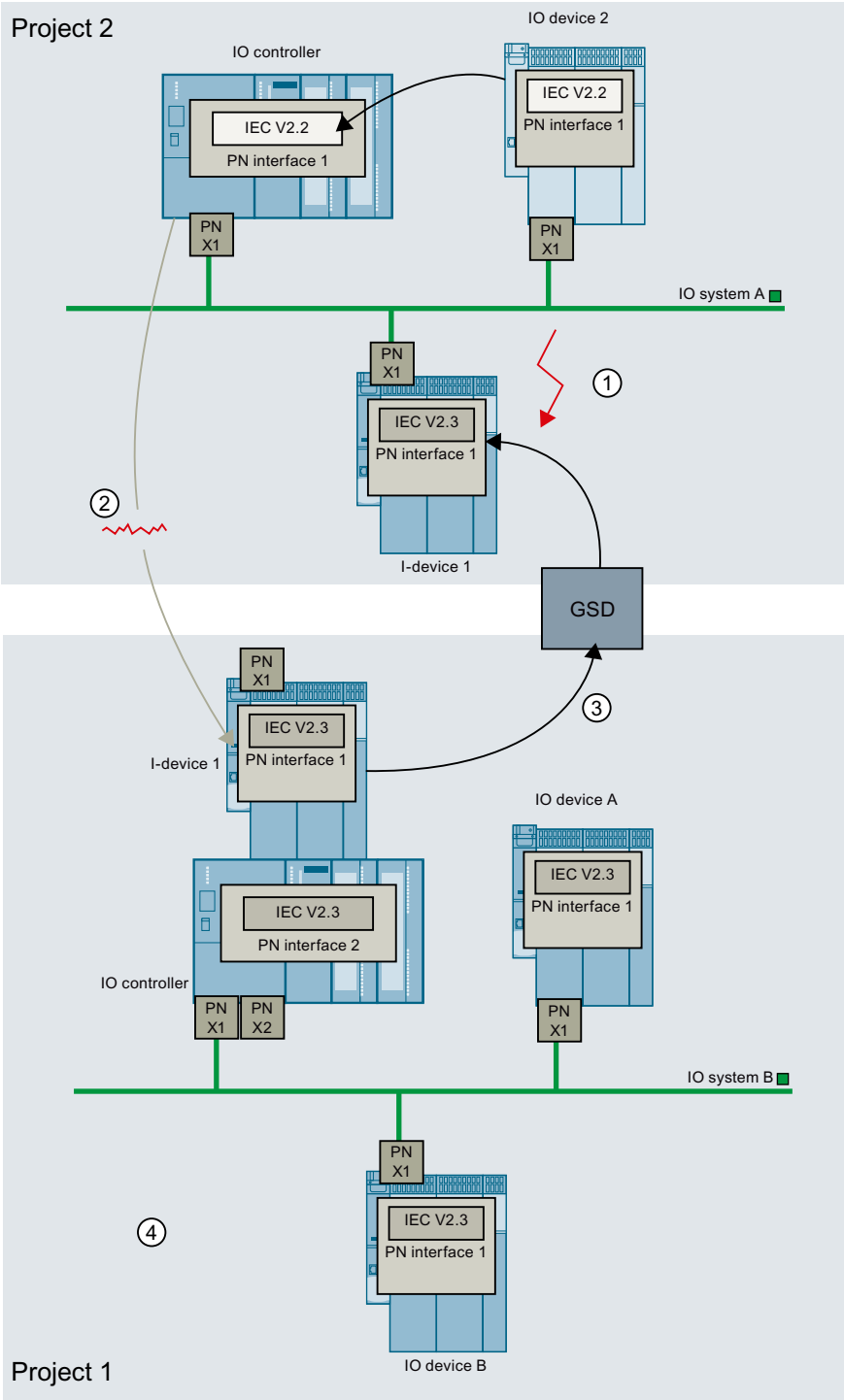

① The PROFINET interfaces have different settings in IO system A (I-device set to V2.3, IO controller set to V2.2, IO device 2 only supports V2.2).

② The settings of the PROFINET interfaces are stored in the I-device (no parameter assignment by the higher-level IO controller).

③ GSD file has entry that IEC V2.3 mode is set.

④ All devices are automatically set to V2.3 mode: This means you must set a PROFINET interface in project 1 manually to V2.2 mode.

Time-of-day functions

Basic principles of time of day functions

All S7-1500 CPUs are equipped with an internal clock. The backup supports the display of the correct time for up to 10 hours if the power supply is interrupted.

Time-of-day format

The clock always shows the time-of-day with a resolution of 1 millisecond and the date including the day of the week. The time adjustment for daylight saving time is also taken into account.

See also

Setting and reading the time of day (Page 967)

Setting and reading the time of day

Setting and reading the time with instructions

You can set, start and read the time and date on the CPU clock with the following instructions in the user program:

- Set the time: "WR_SYS_T"
- Read time "RD_SYS_T"
- Read local time "RD_LOC_T"
- Write local time "WR_LOC_T"

Manual setting

You can also read and set the time manually in the online and diagnostics view under "Functions > Set time".

Time synchronization SIMATIC procedure

Reference

Assigning parameters for the interface properties in the "Properties > General > DP interface > Time synchronization" parameter group.

SIMATIC procedure

With the SIMATIC procedure, the time of day is set according to MMS time-of-day messages (MMS - Manufacturing Message Specification). The MMS time-of-day messages come from a SIMATIC time-of-day transmitter or a CPU that is configured as a time master.

With a CP, the time-of-day messages can either be received by the local CPU or via LAN. It is possible to set whether or not the CP forwards the received time-of-day.

Compared with an NTP method that may be available, the SIMATIC mode provides higher accuracy.

Clock parameters

The clock parameters allow you to make the following settings:

- Synchronization type
  If you have set the synchronization mode to "As master", you can then select the time interval for automatic synchronization.
  If more than one module with a clock is present within a network, you need to specify which CPU will act as the master and which as the slave for time synchronization purposes. There can only be one time master.
- Time interval
  The time interval defines the interval between time queries.

Instructions for synchronizing the time

To ensure that the time is consistent in all modules in the network, the time slaves are updated by the system program at regular configurable intervals. The instruction "SNC_RTCB" allows the date and time to be transferred from the time master to the time slave independently of the configured time interval.

Time-of-day synchronization in NTP mode

Reference

Assigning parameters for the interface properties in the "Properties > General > PROFINET interface > Time synchronization" parameter group. The "Enable time synchronization via NTP server" option is selected.

NTP mode (NTP: Network Time Protocol)

In NTP mode, the device sends time queries at regular intervals (in client mode) to the NTP server in the subnet (LAN). Based on the replies from the server, the most reliable and most accurate time is calculated and the time on the station is synchronized.

The advantage of this mode is that it allows the time to be synchronized across subnets.

The IP addresses of up to four NTP servers need to be configured. The update interval defines the interval between the time queries (in seconds). The value of the interval ranges between 10 seconds and one day.

In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred; this corresponds to GMT (Greenwich Mean Time).

Time synchronization parameters in NTP mode

The clock parameters allow you to make the following settings:

- Enable time synchronization via NTP server
  Select this check box if you want the internal clock to be synchronized using the NTP synchronization mode.
- Server 1-4
  The IP addresses of up to four NTP servers need to be configured.
- Update interval
  The update interval defines the interval between time queries.

Enabling system memory

System memory

A system memory is a bit memory with defined values.

You decide which memory byte of the CPU will become the system memory byte when assigning the system memory parameters.

Benefits

You can use system memory in the user program, for example to run program segments in only the first program cycle after start-up. Two system memory bits are constant 1 or constant 0.

Bits of the system memory bytes

The following table shows the meaning of the system memory:

Bit of the system memory bytes	7	6	5	4	3	2	1	0
Meaning	Reserved (=0)	Reserved (=0)	Reserved (=0)	Reserved (=0)	=0	=1	=1 with change to the diagnosis status	=1 during the startup and in first program cycle after startup, otherwise 0

Note

The selected memory byte cannot be used for intermediate storage of data.

Using clock memory

Clock memory

A clock memory is a bit memory that changes its binary status periodically in the pulse-no-pulse ratio of 1:1.

You decide which memory byte of the CPU will become the clock memory byte when assigning the clock memory parameters.

Benefits

You can use clock memory, for example, to activate flashing indicator lamps or to initiate periodically recurring operations such as recording of actual values.

Available frequencies

Each bit of the clock bit memory byte is assigned a frequency. The following table shows the assignment:

Bit of the clock memory byte	7	6	5	4	3	2	1	0
Period (s)	2.0	1.6	1.0	0.8	0.5	0.4	0.2	0.1
Frequency (Hz)	0.5	0.625	1	1.25	2	2.5	5	10

Note

Clock memory runs asynchronously to the CPU cycle, i.e. the status of the clock memory can change several times during a long cycle.

The selected memory byte cannot be used for intermediate storage of data.

Protection

Setting options for the protection

Access levels

The following section describes how to use the various access levels of the S7-1500 CPUs.

S7-1500 CPUs provide various access levels to limit the access to specific functions.

The individual access levels as well as their associated passwords are specified in the object properties of the CPU. You assign parameters for the access level in a table.

Access level	Access			Access permission
	HMI	Read	Write	Password
Full access (no protection)	✓	✓	✓	
Read access	✓	✓		
HMI access	✓			
No access (complete protection)				

The green checkmarks in the columns to the right of the respective access level specify which operations are possible without knowing the password of this access level.

If you want to use the functions of fields that are not selected in the "Access" column, a password has to be entered:

Example:

You set the access level "Read access". You can see from the table that write access is not possible during operation without entering a password.

The table also shows that full access is necessary for the "Write" function.

To use a function requiring write access during operation, the password for full access must therefore be entered.

NOTICE
Configuring an access level does not replace know-how protection Configuring access levels offers a high degree of protection against unauthorized changes to the CPU by restricting download privileges to the CPU. However, blocks on the memory card are not write- or read-protected. Use know-how protection to protect the code of blocks on the memory card.

Default characteristics

The default access level is "Full access (no protection)". Every user can read and change the hardware configuration and the blocks. A password is not set and is also not required for online access.

The access levels in detail

Below you can find an explanation of the existing access levels and which functions are possible at the respective access level.

- Full access (no protection):
  The hardware configuration and the blocks can be read and changed by all users.
- Read access for F-blocks (F-CPUs only):
  F-blocks of the safety program cannot be changed without authorization by the password of this access level or a higher level.
  Further information is available in the programming and operating manual *SIMATIC Safety - Configuring and Programming.*
- Read access:
  With this access level, read-only access to the hardware configuration and the blocks is possible without entering a password, which means you can upload hardware configuration and blocks to the programming device. In addition, HMI access and access to diagnostics data, display of offline/online comparison results, changing the operating state (RUN/STOP), and setting time-of-day is possible.
  You cannot download blocks or a hardware configuration into the CPU. Moreover, no test functions and firmware updates are possible.
- HMI access:
  Only HMI access and access to diagnostics data possible.
  Example: With the HMI access access level, you can go online and view the diagnostics icons for the states of objects.
  Tags can be read and written via a HMI device.
  At this access level, you can neither load blocks and the hardware configuration into the CPU nor load blocks and the hardware configuration from the CPU into the programming device.
  In addition, the following is **not** possible: Test functions, changing the operating mode (RUN/STOP), firmware update and display of online/offline comparison results.
- No access (complete protection):
  Only identification data can be read, via "Accessible devices", for example.
  Neither read nor write access to the hardware configuration and the blocks is possible. HMI access is also not possible. The server function for PUT/GET communication is disabled in this access level (cannot be changed).
  Legitimation with a configured password provides you with access in accordance with the associated protection level.

Behavior of functions at different access levels

The table below describes which online functions are possible in the various protection levels.

Function	Full access	Read access	HMI access	No access
Identification of the device, via "Accessible devices", for example	Yes	Yes	Yes	Yes
Assigning an emergency address (Emergency IP)	Yes	Yes	Yes	Yes
HMI diagnostics view	Yes	Yes	Yes	No

Monitoring tags (M, I, Q, DB content) via HMI device	Yes	Yes	Yes	No
Modifying tags (M, I, Q, DB content) via HMI device	Yes	Yes	Yes	No
Diagnostics display (for example, device information, connection display, alarm display, diagnostic buffer)	Yes	Yes	Yes	No
Reading cycle time statistics (Online & Diagnostics)	Yes	Yes	Yes	No
Reading information from the hardware configuration (Online & Diagnostics)	Yes	Yes	Yes	No
Reading time-of-day	Yes	Yes	Yes	No
Executing online functions within the hardware configuration (Online & Diagnostics)	Yes	Yes	Yes	No
Acknowledging alarms	Yes	Yes	Yes	No
Receiving alarms	Yes	Yes	Yes	No
Enabling/disabling alarms	Yes	Yes	No	No
Reading tags via test function (STEP 7, tag table or watch table)	Yes	Yes	No	No
Requesting operating state change online (RUN/STOP/ warm restart)	Yes	Yes	No	No
Uploading data blocks, code blocks, hardware configuration to PG/PC	Yes	Yes	No	No
Set time-of-day	Yes	Yes	No	No
Display online/offline comparison results	Yes	Yes	No*	No
Deleting data blocks, code blocks, hardware configuration in the CPU	Yes	No	No	No
Downloading individual data blocks, code blocks, hardware configurations to the CPU	Yes	No	No	No
Loading PLC program to the device and resetting	Yes	No	No	No
Firmware update of CPUs or I/O modules	Yes	No	No	No
Modifying tags via test function (STEP 7, watch table)	Yes	No	No	No
Reading tags in the program status	Yes	No	No	No
Online editing of blocks	Yes	No	No	No
Modifying outputs in STOP mode	Yes	No	No	No

* As of STEP 7 V14, the "HMI access" access level is not sufficient for the display of online/offline comparison results, as they are for example displayed when going online with a project for blocks, PLC data types or PLC tags. You must have read access in order to have the online/offline comparison results displayed. Reason: The equivalence or non-equivalence of online/offline data is determined by STEP 7 through a comparison of the checksums for software and text lists. Reading these checksums requires the "read access" access level.

Behavior of a password-protected module during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Before an online function is executed, the necessary permission is checked and, if necessary, the user is prompted to enter a password.

Going online with a password-protected CPU

Going online with a password-protected CPU requires read access as of STEP 7 V14. When going online, you must therefore enter the password for read access or, if no password was configured here, the password for full access.

If you only have a password for HMI access with a completely protected CPU, cancel the query for the read access password. You are then prompted to enter the password for HMI access. The authorization for HMI access is, however, not sufficient for the online/offline comparison - you require the read access right for this purpose.

Examples:

- A module was configured with read access and you want to execute the "Modify tags" function. Since this requires write access to a test function, the configured password must be entered to execute the function (password for full access).
- A module was configured with HMI access and you want to go online with STEP 7 for this module. Since the data from which STEP 7 determines the equivalence between online and offline objects (checksum) requires read access, you are prompted to enter the password for read access when going online. If you abort the dialog for entering the password, an online connection to the module is nevertheless established. However, only question marks are displayed in the column for comparison icons in an online/offline comparison.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on.

Access authorization to the protected data is in effect for the duration of the online connection. If the online connection is restored after an interruption, you do not have to enter the access data again. To manually cancel access authorization, click "Online > Delete access rights".

Each access level allows unrestricted access to certain functions without entering a password, for example, identification using the "Accessible devices" function.

Access to a password-protected S7-1500 CPU can be restricted locally on the display. The restriction is only in effect when the operating mode switch is set to RUN.

See also

Disabling online access to the CPU via the display (Page 981)

Restriction of communication services (Page 978)

Configuring access levels (Page 975)

Configuring access levels

The following section describes how to configure an access level and enter passwords for an S7-1500 CPU.

For an S7-1500 CPU, you can enter multiple passwords and thereby set up different access rights for various user groups.

The passwords are entered in a table in such a way that exactly one access level is assigned to each password.

The effect of the password is given in the "Access level" column.

Example

You select the "No access (complete protection)" access level for a standard CPU (in other words, not an F-CPU) and enter a separate password for each of the access levels that lie above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the set passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (Full access (no protection)) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (read access) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (HMI access) allows access as if the CPU were read/write protected. For users who know this password, only HMI access is possible.

Procedure

To configure the access levels of an S7-1500 CPU, follow these steps:

1. Open the properties of the module in the inspector window.
2. Open the "Protection" entry in the area navigation.
   A table with the possible access levels appears in the inspector window.

Access level	Access			Access permission
	HMI	Read	Write	Password
Full access (no protection)	✓	✓	✓	
Read access	✓	✓		
HMI access	✓			
No access (complete protection)				

3. Select the required access level in the first column of the table. The green checkmarks in the columns to the right of the respective protection level show you which operations are still available without entering the password.
4. If you have selected an access level other than "Full access":
   Specify a password for full access in the first row of the "Password" column.
   In the "Confirmation" column, enter the selected password again to protect against incorrect entries.
   Ensure that the password is sufficiently secure, in other words, that is does not follow a machine-recognizable pattern!
   You must enter a password in the first row "Full access (no protection)". This enables unrestricted access to the CPU for those who know the password, regardless of the selected protection level.
5. Assign additional passwords as needed to other access levels if the selected access level allows you to do so.
6. Download the hardware configuration so that the access level takes effect.

Result

The hardware configuration and the blocks are protected against unauthorized online access according to the set access level. If an operation cannot be executed without a password due to the set access level, a dialog for entering a password is displayed.

See also

Setting options for the protection (Page 970)

Password transfer upon module replacement

If you assign a protection level that requires the input of a password, the CPU converts this input using a special algorithm and, put in simplified terms, saves a special checksum for that password.

STEP 7 never saves the password itself, nor is a password saved in the CPU in a form from which it could be reconstructed. If the password is lost, a new CPU configuration must therefore always be loaded; see the section below.

The algorithm for forming or encrypting the checksum from the password is constantly being developed to offer maximum protection from unauthorized access. This has no impact on the procedure for configuration, with the exception of module replacement. Depending on the firmware version of the CPUs affected (the CPU to be replaced and the replacement CPU), you are either offered an update to the latest algorithm or prompted to assign a new password because the replacement CPU cannot use the existing password configuration.

If the CPU to be replaced and the replacement CPU are identical in terms of the algorithm used, no action is required: the password configuration and the other parameter settings are transferred.

The same applies to the configured users of the CPU Web server: If you create users with specific rights, you must configure a password for each user. In this case too, you are informed if the algorithms are not compatible in the event of module replacement, and may then need to create new users and passwords.

Forgotten your password - loading new CPU configuration

If you forget the password protecting the CPU, you cannot go online with the CPU to load a new configuration (with a new password).

Proceed as follows to load a new configuration (with new password) in this case:

1. Plug the SIMATIC memory card into the card reader of your programming device.
2. In the project tree, drag and drop the CPU folder with the new configuration to the symbol for the SIMATIC memory card in the card reader.
3. In the load dialog, confirm that the configuration and program of the existing password-protected CPU configuration are to be overwritten.

Behavior of a password-protected CPU during module replacement

- You are replacing an S7-1500 CPU (**firmware < 2.0**) with an S7-1500 (**firmware ≥ 2.0**): You are informed that a more up-to-date algorithm is available for password encryption. You have the option of using the new algorithm by clicking on the button or - on the grounds of compatibility - retaining the previous algorithm.
- You are replacing an S7-1500 CPU (**firmware ≥ 2.0**) with an S7-1500 (**firmware < 2.0**): You are informed that the previous passwords will be lost and that new passwords must be configured.

Behavior of a CPU with configured Web server users during module replacement

To create users and their specific rights, you must configure a password for each user. Similar rules apply for module replacement as for a password-protected CPU:

- You are replacing an S7-1500 CPU (**firmware < V2.0**) with an S7-1500 (**firmware ≥ V2.0**): You are informed that a more up-to-date algorithm is available for password encryption. You have the option of using the new algorithm by clicking on the button or - on the grounds of compatibility - retaining the previous algorithm.
- You are replacing an S7-1500 CPU (**V1.7 ≤ firmware ≤ V1.8**) with an S7-1500 (**firmware ≤ 1.6**) or
- You are replacing an S7-1500 CPU (**firmware ≥ 2.0**) with an S7-1500 (**firmware < 2.0**): You are informed that all configured users are deleted and that the user "Everyone" is reset to default user rights.
  In this case, you must create the users again with rights and passwords.

Restriction of communication services

Introduction

The CPU can be the server for a number of communication services. This means that other communication participants can access CPU data even if you have not configured and programmed connections for the CPU.

The local CPU as a server thus does not have the possibility to control communication to the clients.

The parameter "Connection mechanisms" in the "Protection" area of the CPU parameters is used to specify whether this type of communication is permitted or not for the local CPU during operation.

Permit access with PUT/GET communication from remote partners

By default, the "Permit access with PUT/GET communication from remote partners (...)" option is disabled. In this case, read and write access to CPU data is only possible for communication connections that require configuration or programming both for the local CPU and for the communication partner. Access through BSEND/BRCV instructions is possible, for example.

Connections for which the local CPU is only a server (meaning that no configuration/ programming of the communication with the communication partner exists at the local CPU), are therefore not possible during operation of the CPU, for example,

- For PUT/GET, FETCH/WRITE or FTP access via communication modules
- For PUT/GET access from other S7 CPUs
- For HMI access that is realized via PUT/GET communication

If you want to allow access to CPU data from the client side, meaning that you do not want to restrict the communication services of the CPU, activate the "Permit access with PUT/GET communication from remote partners" option.

Group alarms for security events

What you should know about group alarms for security events

Security events in the diagnostics buffer

The CPU stores information on important actions and events in the diagnostics buffer.

The following security events (event types) lead to an entry in the diagnostics buffer of the S7-1500:

- Going online with the correct or incorrect password.
- Manipulated communications data detected.
- Manipulated data detected on memory card.
- Manipulated firmware update file detected.
- Changed protection level (access protection) downloaded to the CPU.
- Password legitimization restricted or enabled (by instruction or CPU display).
- Online access denied due to the possible number of simultaneous access attempts being exceeded.
- Timeout when an existing online connection is inactive.
- Logging in to the Web server with the correct or incorrect password.
- Creating a backup of the CPU.
- Restoring the CPU configuration.
- During startup:
  - Project on the SIMATIC memory card has changed (the SIMATIC memory card remains the same)
  - SIMATIC memory card was replaced

Group alarm for security events

To prevent the diagnostics buffer being "swamped" by large numbers of identical security events, as of STEP 7 V13 service pack 1, you can set parameters so that these events are entered in the diagnostics buffer as a group alarm. Per interval (monitoring time), the CPU then only generates one group alarm per event type.

Example: Brute-Force attacks

With Brute-Force attacks, attackers attempt to gain access to the CPU by systematically trying out a large number of the possible password combinations. As a result, the CPU receives a large number of login requests within seconds and the inrush of identical single entries in the diagnostics buffer leads to the loss of important diagnostics information. Group alarms help here.

Principle of operation of group alarms

The CPU enters the first three events of an event type in the diagnostics buffer. It then ignores all subsequent security events of this type.

At the end of the monitoring time (interval), the CPU generates a group alarm in which it enters the event type and the frequency of the event during the elapsed interval.

If these security events also occur in the intervals that follow, the CPU only generates one group alarm per interval.

An attack leaves the following pattern in the diagnostics buffer: Three individual alarms followed by a series of group alarms. This series can consist of two, three or more group alarms depending on the selected monitoring time and duration of the attack.

Since the alarms in the diagnostics buffer have a time stamp, the duration of the attack can be determined.

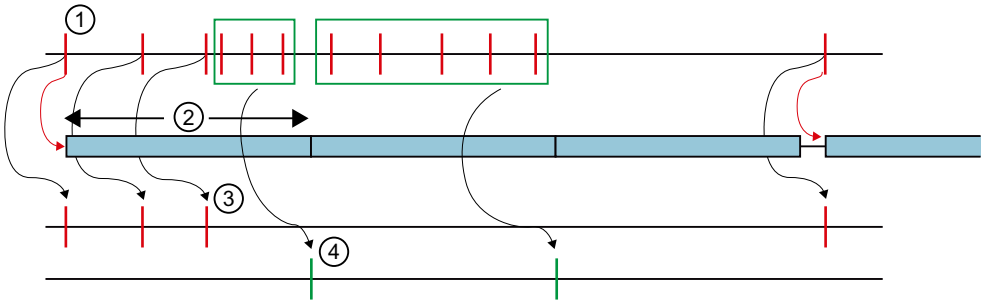

① Security events of one type, for example login attempts with an invalid password.

② Interval (monitoring time): When a security event of a particular type occurs the first time (or reoccurs), the monitoring time is started (or restarted).

③ Single alarms: The first three security events of a type are entered in the diagnostics buffer immediately. As of the fourth security event of this type, the CPU only creates group alarms.

If a security event of this type occurs after a pause of at least one interval, the CPU enters a single alarm in the diagnostics buffer and restarts the monitoring time.

④ Group alarms: After three security events, the CPU only generates a group alarm as a summary of all other security events in this interval. If these security events also occur in the intervals that follow, the CPU only generates one group alarm per interval.

See also

Configuring a group alarm for security events (Page 980)

What you need to know about SIMATIC memory cards (Page 907)

Configuring a group alarm for security events

Requirements

- STEP 7 V13 Service Pack 1 or higher
- S7-1500 CPU with firmware version 1.7 or higher

Procedure

To configure group alarms for security events, follow these steps:

1. Click on the CPU symbol in the network view.
   The properties of the CPU are displayed in the Inspector window.
2. Go to the "Protection" > "Security event" area.
3. Click on "Security event".
4. Select the option "Summarize security events in case of high message volume" to enable group alarms for security events.
5. Set the duration of an interval (monitoring time); the default is 20 seconds.

See also

What you should know about group alarms for security events (Page 979)

Disabling online access to the CPU via the display

You can block access to a password-protected CPU on the display of an S7-1500 CPU (local block).

The access block only functions if the operating mode switch is set to RUN.

The access block is effective irrespective of the password protection. This means that even if someone accesses the CPU via a connected programming device and has entered the correct password, access to the CPU remains disabled.

The access block can be set separately for each protection level on the display, so that, for example, read access is allowed locally, but write access is not allowed locally.

In addition, a password can be configured for the display in the properties of the CPU so that the local access protection is protected by a local password.

Procedure

Proceed as follows to set the local access protection for an S7-1500 CPU on the display:

1. Select the menu Settings > Protection on the display.
2. Confirm with "OK" and set for each protection level whether access in RUN mode is permitted:
   Allow: Access to the CPU is possible with password input.
   Block in RUN: If the operating mode switch is set to RUN, no user can log on to the CPU with the rights of this protection level, even when the relevant password is known. In STOP mode, access is possible with password input.

See also

Setting options for the protection (Page 970)

Security

Setting options for security functions

Introduction

Digital certificates play an important role in the secure data transfer between two devices. When digital certificates are used, a client can only establish a secure connection to a server when the server accepts the digital certificate of the client, classifies it as trustworthy and trusts it. The client must also check and trust the certificate of the server.

CPUs of the S7-1500 as of firmware V2.0 support the use of certificates.

Device-specific and project-wide certificate manager

With the local CPU-specific certificate manager, certificates can be generated and managed for the respective device. These can then be used by the following objects:

- OPC UA server of the device
- Web server of the device
- Any additional system feature that requires certificates and is referenced by a certificate ID (in TIA Portal V14 these are sOUC and TMAIL).

Note

Each certificate in the certificate manager receives an ID for unique referencing in specific SDTs. The certificate ID is assigned by the certificate manager when a certificate is created or imported. The certificate ID is unique within the project and cannot be changed.

Certificates of the local CPU-specific certificate manager refer solely to the device. If you want to use certificates project-wide, you have to activate the global security certificate manager in the device.

In the global security certificate manager you can utilize additional functions for the management of your certificates:

- Import of new certificates and certificate authorities.
- Export of the certificates and certificate authorities used in the project.
- Renewal of expired certificates and certificate authorities.
- Replacement of existing certificate authorities.
- Adding trusted certificates and certification authorities.
- Deleting manually imported certificates.

The global security settings are not visible until they have been activated in the project for at last one device. The "Use global security settings for certificate manager" check box is available to this purpose in the local CPU-specific certificate manager: It is located in the Inspector window in the general settings of a device with security functions under "Protection

& Security > Certificate manager". Depending on this setting either only the local CPU-specific certificate manager with limited functionality is used or the global security certificate manager.

NOTICE
When you activate the global security certificate manager for the project, the contents of the local CPU-specific certificate manager are lost: • The private keys cannot be restored. • Export your certificates if you want to keep using them in the global security certificate manager. • Update your configuration or your program, because the ID of the certificate can change.

After activation of the global security certificate manager the local CPU-specific certificate manager acts like a temporary memory of the CPU-specific global certificates. When you create a new certificate in the local CPU-specific certificate manager, it is entered in the global security certificate manager. You can also select certificates in the global security certificate manager in the local CPU-specific certificate manager as a certificate for the CPU.

See also

Certificates for the Web server (Page 989)

Certificates for OPC UA server (Page 990)

Local CPU-specific certificate manager (Page 983)

Basics of Secure Communication (Page 302)

Local CPU-specific certificate manager

Introduction

With the local CPU-specific certificate manager you can use certificates for the OPC UA server and the Web server without access to the global security certificate manager.

Functions of the local certificate manager

In comparison to the global certificate manager, the local CPU-specific certificate manager only has a limited functionality. With the local certificate manager you can solely generate, assign, export or delete certificates.

The settings for the local CPU-specific certificate manager are located in the Inspector window in the general settings of the device under "Protection & Security > Certificate manager".

- Option for activating the global certificate manager
- Table for device certificates: Display and management of the certificates that you use, for example, for secure OUC (TLS server / TLS client), the Web server and the OPC UA server of the device.
- Table for certificates of the partner devices: Display of the certificates that you have assigned as trusted users or clients.

Device certificates

Device certificates are valid for the device and its associated components. The certificates displayed here were, for example, also created for the Web server or for OPC UA and are displayed here. Similarly, certificates with the settings for the Web server or for OPC UA can be created here - but have to be used at the corresponding location.

When you create a new certificate, the following values are entered as the default in the corresponding dialog for the device:

- Certificate holder: <Name of the CPU>/TLS. The set of characters is limited so that the specifications to ASN.1 can be observed.
- Type of signature: sha256RSA or sha1RSA, depending on the security policy in use.
- Validity period: The start and end of the validity period is entered in the two fields for the validity period in dependence on the system time.
- Usage: TLS. The extensions "KeyUsage" and "ExtendedKeyUsage" for certificates of the X.509 V3 standard are filled suitably for OPC UA.
- Alternative name of the certificate holder (SAN): IP address of the used interfaces.

You can overwrite the defaults with your own values.

Certificates of the partner devices

For the certificates of the partner devices you can select device certificates of the local certificate manager. Device certificates are used without their keys. You can also select the certificates of other CPUs or CPs for secure communication within the project. The certificates of other CPUs or CPs are loaded to the CPU without private keys. In as far as the chain of certificates via the global certificate manager is available, the chain of certificates belonging to a CPU or CP is still not interrupted and the corresponding certificate authority and intermediate certificates remain assigned to the CPU or the CP.

When you delete a certificate in the table of certificates of the partner devices, only the link of the certificate to the global certificate manager is deleted. The certificate is therefore no longer available to the device, but remains in the global certificate manager. If required, the certificate can therefore be used again in the device if it is selected as a new certificate from the global certificate manager. If the certificate is to be deleted for the entire project, you have to do so in the shortcut menu of the global certificate manager.

See also

Setting options for security functions (Page 982)

Creating certificates (Page 985)

Assigning certificates (Page 986)

Exporting certificates (Page 987)

Deleting certificates (Page 988)

Global security certificate manager (Page 323)

Creating certificates

Introduction

You can create a new certificate for a device. When you create a new certificate with deactivated global security certificate manager, the certificate only applies for this device. With exclusive use of the local CPU-specific certificate manager, only self-signed certificates can be created.

You can furthermore create new certificates directly on the Web server or with OPC UA.

Requirement

The device must offer security functions with certificate manager. This applies for S7-1500 CPUs as of firmware V2.0.

Procedure

To create a new certificate, follow these steps:

1. In the Inspector window, go to "Protection & Security > Certificate manager" in the properties of the device.
2. Click "Add" in the table of certificates. A new row is entered in the table.
3. Click in the new row. The selection for new certificates is opened.
4. Click the "Add" button. The corresponding dialog window opens in which you can enter the data for the new certificate.
5. Select how the new certificate is to be signed:
   - Self-signed: If you want to assign the signature yourself.
   - Signed by a certificate authority (CA): Select a certificate authority from the drop-down list.
6. Enter the parameters for the certificate.
   - Certificate holder: The certificate holder is the device for which the certificate applies.
   - Valid from: The certificate is valid from the specified date.
   - Valid until: The certificate is no longer valid after the specified date has expired.
   - Usage: This value is entered in digital certificates of the X.509 V3 standard in the extensions "KeyUsage" and "ExtendedKeyUsage".
   - Alternative name of the certificate holder (SAN).
7. When you click "OK", the new certificate is created and entered in the table.

Note

Self-signed certificates

With a self-signed certificate you have already created an encryption for the transport path, but the recipient of a message does not yet have a confirmation whether the named sender is the real sender. A message that was signed by a certificate authority identifies the sender as "real" towards the recipient. However, you can create a self-signed certificate for the device when you still have to wait for a certificate signed by a certificate authority. Through the global security certificate manager in the project tree you can then either replace the temporarily used self-signed certificate or renew it by specifying the certificate authority after you have received the "official" certificate from the certificate authority (CA). You can naturally also retain a self-signed certificate in the project and forgo the usage of an "official" CA certificate.

Result

A new certificate is created for the device. For an S7-1500 CPU "TLS" is entered automatically as the usage. This is the case when a new self-signed certificate is issued or when a self-created certificate is issued by the root certificate authority of the TIA Portal via the activated global security certificate manager.

Certificate ID

Each certificate in the certificate manager receives an ID. This ID is used for identification of the certificates in the following cases:

- OPC UA server certificate
- List of trusted OPC UA clients
- List of trusted OPC UA users
- Web server certificate
- Secure OUC instructions

The certificate ID is assigned by the certificate manager when a certificate is created or imported. The certificate IDs are required to uniquely assign the certificates to specific system data types, for example, to the SDT TCON_IP_v4_Sec. The certificate ID is unique within the project and cannot be changed.

See also

Local CPU-specific certificate manager (Page 983)

Assigning certificates

Introduction

You can assign available certificates to the CPUs and CPs in the Inspector window.

Requirement

The device must offer security functions with certificate manager. This applies for S7-1500 CPUs as of firmware V2.0.

Procedure

To assign an available certificate to a device, follow these steps:

1. In the Inspector window, go to "Protection & Security > Certificate manager" in the properties of the device.
2. Click "Add" in the table of certificates. A new row is entered in the table.
3. Click in the new row. The selection for new certificates is opened.
4. Select an available certificate. If only the local CPU-specific certificate manager is active, you do not see the globally available certificates. You can only see and select globally available certificates when the global security certificate manager is activated.
5. Click the green check mark to enter the selection in the table of certificates.

Result

The selected certificate was assigned to the device.

See also

Local CPU-specific certificate manager (Page 983)

Exporting certificates

Introduction

You can export available certificates via the certificate manager.

Requirement

The device must offer security functions with certificate manager. This applies for S7-1500 CPUs as of firmware V2.0.

Procedure

To export a certificate, follow these steps:

1. In the Inspector window, go to "Protection & Security > Certificate manager" in the properties of the device.
2. Right-click in the row with the certificate to be exported to open the shortcut menu.
3. Click "Export certificate".
4. Export the certificate into one of the selectable file formats: CER, DER.

Private keys cannot be exported since they are encrypted asymmetrically and cannot be restored.

Result

The selected certificate has been exported and can now be backed up or be used in other projects.

See also

Local CPU-specific certificate manager (Page 983)

Deleting certificates

Introduction

You can delete available certificates via the certificate manager.

Requirement

The device must offer security functions with certificate manager. This applies for S7-1500 CPUs as of firmware V2.0.

Procedure

To delete a certificate, follow these steps:

1. In the Inspector window, go to "Protection & Security > Certificate manager" in the properties of the device.
2. Right-click in the row with the certificate to be deleted to open the shortcut menu.
3. Click "Delete certificate".

Result

The selected certificate was deleted in the device. With an activated global security certificate manager the certificate is retained for the project, but is no longer used for the device.

See also

Local CPU-specific certificate manager (Page 983)

Certificates for the Web server

Introduction

You can create or assign a server certificate for the Web server in the Inspector window under "Web server > Server security".

Local Web server certificate

The following restrictions apply when you create Web server certificates locally without using the global security settings for the certificate manager:

- The certificate only applies to the configured CPU and is not available throughout the project.
- Only self-signed certificates are created. Self-signed certificates are not embedded in a "Public Key Infrastructure" (PKI) and cannot sign other certificates.
- Private keys of the certificate cannot be exported.
- Import and renewal of certificates is only possible in the global security settings for the certificate manager.

Default settings for Web server certificates

When you create a new certificate, the following values are entered as the default in the corresponding dialog for a Web server:

- Certificate holder: <Name of the CPU>/Web server. The set of characters is limited so that the specifications to ASN.1 can be observed.
- Type of signature: sha256RSA or sha1RSA, depending on the security policy in use.
- Validity period: The start and end of the validity period is entered in the two fields for the validity period in dependence on the system time.
- Usage: Web server. The extensions "KeyUsage" and "ExtendedKeyUsage" for certificates of the X.509 V3 standard are filled suitably for the Web server.
- Alternative name of the certificate holder (SAN): IP address of the used interfaces of the CPU and CP.

You can overwrite the defaults with your own values.

NOTICE
SAN with reserved IP address When the alternative name of the certificate holder contains one or more reserved IP addresses, the certificate does not fulfill the guideline "Baseline Requirements Certificate Policy for the Issuance and Management of Publicly-Trusted Certificates". This can result in incompatibility with your own Web browsers. To meet the guideline you should replace the IP addresses of the alternative name of the certificate holder with domain names. This requires a DNS server in the network that dissolves the IP addresses of the devices.

See also

Setting options for security functions (Page 982)

Example: HTTP over TLS (Page 320)

Certificates for OPC UA server

Introduction

You can create or assign a server certificate for OPC UA in the Inspector window under "OPC UA > Server > Security".

Local server certificates for OPC UA

The following restrictions apply when you create server certificates locally for OPC UA without using the global security settings for the certificate manager:

- The certificate only applies to the configured CPU and is not available throughout the project.
- Only self-signed certificates are created. Self-signed certificates are not embedded in a "Public Key Infrastructure" (PKI) and cannot sign other certificates.
- Private keys of the certificate cannot be exported.
- Import and renewal of certificates is only possible in the global security settings for the certificate manager.

Default settings for server certificates

When you create a new certificate, the following values are entered as the default in the corresponding dialog for a OPC UA:

- Certificate holder: <Name of the CPU>/OPCUA. The set of characters is limited so that the specifications to ASN.1 can be observed.
- Type of signature: sha256RSA or sha1RSA, depending on the security policy in use.
- Validity period: The start and end of the validity period is entered in the two fields for the validity period in dependence on the system time.
- Usage: OPC UA. The extensions "KeyUsage" and "ExtendedKeyUsage" for certificates of the X.509 V3 standard are filled suitably for OPC UA.
- Alternative name of the certificate holder (SAN): IP address of the used interfaces and the URI of the OPC UA server.

You can overwrite the defaults with your own values.

See also

Setting options for security functions (Page 982)

Test, commissioning and routing

Test, commissioning, routing

Select the check box if, for example, you want to use PG functions needed for commissioning and testing regularly over this interface.

If you select the "Test, commissioning, routing" option, the interface becomes the active device on PROFIBUS (in other words, the interface is included in the token rotation of PROFIBUS). The following functions are then possible:

- Programming (for example loading)
- Testing (status/modify)
- S7 routing (I-slave as gateway)

This can extend the token rotation time. In time-critical applications and when S7 routing and client functionality are not required for communication, this option should therefore not be enabled.

If the "Test, commissioning, routing" check box is not selected, the interface functions only as a server for communication services.

Server functionality means that only the partner (for example PG, OP or automation system) can initiate communication. Note the following features when the "Test, commissioning, routing" option is not enabled:

- S7 routing is not possible
- Programming and testing are possible but slow (no extension of the bus token rotation time)

Deactivating SNMP

The network management protocol SNMP (Simple Network Management Protocol) is a protocol that uses various services and tools for detection and diagnostics of the network topology. SNMP uses the transport protocol UDP. SNMP recognizes two network components, the SNMP manager and the SNMP client. The SNMP manager monitors the network nodes: The SNMP clients collect the various network-specific information in the individual network nodes and store it in a structured form in the MIB (Management Information Base). Various services and tools can run detailed network diagnostics with the help of this data.

Under certain conditions, it may be useful to disable SNMP. Examples:

- The security guidelines in your network do not allow the use of SNMP.
- You use your own SNMP solution, e.g. with your own communications instructions.

If you disable SNMP for a device, various options for diagnostics of the network topology (e.g. using the PRONETA tool or the Web server of the CPU) are no longer available to you.

Requirements

- CPU S7-1200 V4.1 and higher
- CPU S7-1500 V1.8 and higher

Disabling SNMP

To disable SNMP, follow these steps:

1. In STEP 7, create a data block that contains the structure of data record $F003_H$. The following table shows the structure of the data record $F003_H$.

Byte	Element	Code	Explanation
0-1	BlockID	$F003_H$	Header The data record length is counted starting at byte 4 "Version".
2-3	BlockLength	8	
4	Version	01_H	
5	Subversion	00_H	
6-7	Reserved	-	-
8-11	SNMP controller	Disable/enable SNMP	If you want to disable SNMP, enter the value 0. If you want to enable SNMP, enter the value 1.

2. Transfer the data record $F003_H$ in the startup OB (OB100) to the CPU with the WRREC instruction (write data record). Use an integrated interface of the CPU as the hardware ID.

Example: disabling SNMP for a CPU 1516-3 PN/DP

Introduction

As the security guidelines in your network do not allow SNMP, you want to disable SNMP for a CPU 1516-3 PN/DP.

Requirement

- CPU 1516-3 PN/DP with firmware version V2.0
- STEP 7 V14

Disabling SNMP

To disable SNMP for the CPU 1516-3 PN/DP, follow these steps:

1. Create a data block that contains the structure of data record $B071_H$. The figure below shows the data block "Deactivate SNMP". The data block "Deactivate SNMP" contains not only the data record $B071_H$, but additional tags that you use to transfer the data record. The tag "snmp_deactivate" is used to trigger the job for WRREC. Place this tag in the retentive memory area so that the value is also available in the startup OB (OB100).

Deactivate SNMP

	Name	Data type	Start value	R...	A...	W...	V...	Comment
1	Static							
2	snmp_deactivate	Bool	TRUE	☑	☑	☐	☑	Tag for deactivation
3	snmp_record	Struct		☑	☑	☐	☑	Data record 16#B071
4	BlockID	UInt	16#F003	☑	☑	☐	☑	
5	BlockLenght	UInt	8	☑	☑	☐	☑	
6	Version	USInt	1	☑	☑	☐	☑	
7	Subversion	USInt	0	☑	☑	☐	☑	
8	Reserved	UInt	0	☑	☑	☐	☑	
9	SNMPControl	UDInt	0	☑	☑	☐	☑	
10	snmp_done	Bool	false	☐	☑	☐	☑	
11	snmp_error	Bool	false	☐	☑	☐	☑	
12	snmp_status	DWord	16#0	☐	☑	☐	☑	

2. Transfer the data record $B071_H$ in the startup OB (OB100) to CPU 1516-3 PN/DP with the WRREC instruction (write data record). In the following program code, the data record $B071_H$ is transferred with the WRREC instruction in a REPEAT UNTIL loop.

```
ORGANIZATION_BLOCK "Startup"
TITLE = "Complete Restart"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
BEGIN
 REPEAT
 "WRREC_DB_1"
 (REQ := "Deactivate
SNMP".snmp_deactivate,
  //Transfer data record
 INDEX:=16#b071,
  //Data record number for SNMP deactivation
 ID:="Local~PROFINET_interface_1",
  //any integrated PROFINET Interface
 DONE => "Deactivate SNMP".snmp_done,
 ERROR => "Deactivate SNMP".snmp_error,
 STATUS => "Deactivate SNMP".snmp_status,
 RECORD := "Deactivate
SNMP".snmp_record)
  //Data record
 UNTIL "Deactivate SNMP".snmp_done OR "Deactivate SNMP".snmp_error
 END_REPEAT;
END_ORGANIZATION_BLOCK
```

Re-enabling SNMP

With small changes, you can use the program code used above to enable SNMP. In the user program, assign the "Deactivate SNMP".snmp_record.SNMPControl tag the value "1":

```
"Deactivate SNMP".snmp_record.SNMPControl := 1;
```

SNMP will then be enabled again the next time the CPU is started.

Deactivating SNMP - complete example

Introduction

The example applies to S7-1500 and S7-1200.

Follow these steps to apply the program code to your project:

1. Copy the entire program code to the clipboard.
2. Open a text editor (e.g. "Editor").
3. Paste the content of the clipboard to the text editor with Ctrl+V. Save the document as scl file, for example, SNMP_DEACT.scl.
4. Open your project in STEP 7.
5. Import the scl file as an external source.
6. Generate the startup OB and the data blocks.
7. Right-click the scl file and select "Generate block from source" in the shortcut menu.

Example program with WRREC call in OB 100

```
DATA_BLOCK "Deactivate_SNMP"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
NON_RETAIN
 VAR
   snmp_deactivate : Bool;
   snmp_record : Struct
     BlockID : UInt;
     BlockLength : UInt;
     "Version" : USInt;
     Subversion : USInt;
     Reserved : UInt;
     SNMPControl : UDInt;
   END_STRUCT;
   snmp_done : Bool;
   snmp_error : Bool;
   snmp_Status : DWord;
 END_VAR

BEGIN
  snmp_deactivate := true;
  snmp_record.BlockID := 16#f003;
  snmp_record.BlockLength := 8;
```

```
  snmp_record."Version" := 1;
END_DATA_BLOCK

ORGANIZATION_BLOCK "Startup"
TITLE = "Complete Restart"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
BEGIN
  REPEAT
    "WRREC_DB_1"(REQ := "Deactivate_SNMP".snmp_deactivate,
                 ID := "Local~PROFINET-Schnittstelle_1",
                 INDEX := 16#b071,
                 DONE => "Deactivate_SNMP".snmp_done,
                 ERROR => "Deactivate_SNMP".snmp_error,
                 STATUS => "Deactivate_SNMP".snmp_Status,
                 RECORD := "Deactivate_SNMP".snmp_record);
   UNTIL "Deactivate_SNMP".snmp_done OR "Deactivate_SNMP".snmp_error
   END_REPEAT;
END_ORGANIZATION_BLOCK

DATA_BLOCK "WRREC_DB_1"
{OriginalPartName := 'WRREC';
 VersionGUID := 'bc169451-58cd-44a3-855b-3f78cc0623c8';
 S7_Optimized_Access := 'TRUE' }
AUTHOR : SIMATIC
FAMILY : DP
NAME : WRREC
VERSION : 1.0
NON_RETAIN
WRREC
BEGIN
END_DATA_BLOCK
```

Isochronous mode (central)

Configuring isochronous mode for central I/O in S7-1500

When you configure the isochronous mode of a module centrally in an S7-1500 station, such as an analog input module, disable it in the same way as when configuring a module that is configured in distributed manner:

- In the properties of the I/O modules you must enable isochronous mode, assign the addresses of the module to a process image partition, and assign this process image partition to the isochronous mode interrupt OB.
- In the properties of the isochronous mode interrupt OB, adjust the application cycle and delay time (recommendation: leave automatic setting).
- In the properties of the CPU, also enable isochronous mode and adjust the settings for send clock and Ti/To values.
  A new feature is that you can select a synchronization type: Here you can set whether the CPU should use a local send clock or whether the send clock should be synchronized to the send clock of the PROFINET interface X1 for the centrally configured modules. In the latter case the send clock of the PROFINET interface is leading.

Requirement

- S7-1500 CPU firmware V2.6 or higher (without compact CPUs and without S7-1500R/H CPUs)
- STEP 7 (TIA Portal) V15.1 or higher
- Centrally configured isochronous modules
- No PROFINET IO and PROFIBUS DP systems on centrally configured communications processors (CPs) or communication modules (CMs)
- No communications processors (CPs) or communication modules (CMs) as I-devices or I-slaves in the central configuration

Basic procedure

Configuring I/O and isochronous mode interrupt OB

To create an isochronous connection between the I/O and user program, follow these steps:

1. Select the S7-1500 CPU in the network view of STEP 7. Change to the device view.
2. Insert an I/O module that can be operated isochronously (e.g. AI 8 x U/I HS).
3. Go to the "I/O addresses" area in the Inspector window of the selected I/O module.

4. Make the following settings in the I/O addresses area:
   - Select the "Isochronous mode" option.
   - Select a process image partition, e.g., process image partition 1.
   - Click on the "Organization block" drop-down list and click the "Add new object" button or select an already existing OB. A dialog box for selecting organization blocks opens.
   - Select the "Synchronous Cycle" OB. Confirm the selection with "OK".
     In the case of automatic number assignment, OB 61 will be generated and opened. The "Isochronous mode" area is selected in the Inspector window, and you can continue directly with the setting of the application cycle and delay time and start the programming of the OB in the instruction section.
5. If, due to the utilized modules, the isochronous system only operates with a certain send clock (e.g. 1 ms) but the process values must be sampled faster, use the oversampling function. For modules that support this function, scan the process values for channels, for example, of an analog input module, at shorter intervals.
6. If required, insert additional I/O modules in the device view. Adapt the configuration and the settings for isochronous mode.

Configuring the CPU

1. To check the settings for isochronous mode, select the CPU in the device view of STEP 7.
2. Navigate to the "Isochronous mode" area in the Inspector window of the selected CPU
3. In the "Isochronous mode for local modules" area, check to determine if:
   - Isochronous mode is selected
   - "Local send clock" is selected as synchronization type when the clock synchronization is only to gain access for the centrally configured module
     or as synchronization type "Use send clock of PROFINET interface", if the SEND clock for the centrally configured module is to be synchronized to the send clock of the PROFINET interface X1.

   If necessary, you can adapt the send clock and times T_i/T_o for the isochronous read-in/output of data.
4. In the "Detail overview" area, you see all modules of the configuration you can operate isochronously. Select or deselect isochronous mode for the desired modules as appropriate.

Time sequence of synchronization in the central configuration

From reading-in of input data to outputting of output data

The basic time sequence of all components involved in synchronization is explained in the following:

① Measured value acquisition in the process

② Isochronous read-in of measured values (input data) from the central I/O

③ Transport of input data to the CPU

④ Further processing in the input data in the isochronous application of the CPU (OB 61)

⑤ Transport of output data to the central I/O

⑥ Isochronous output of output data

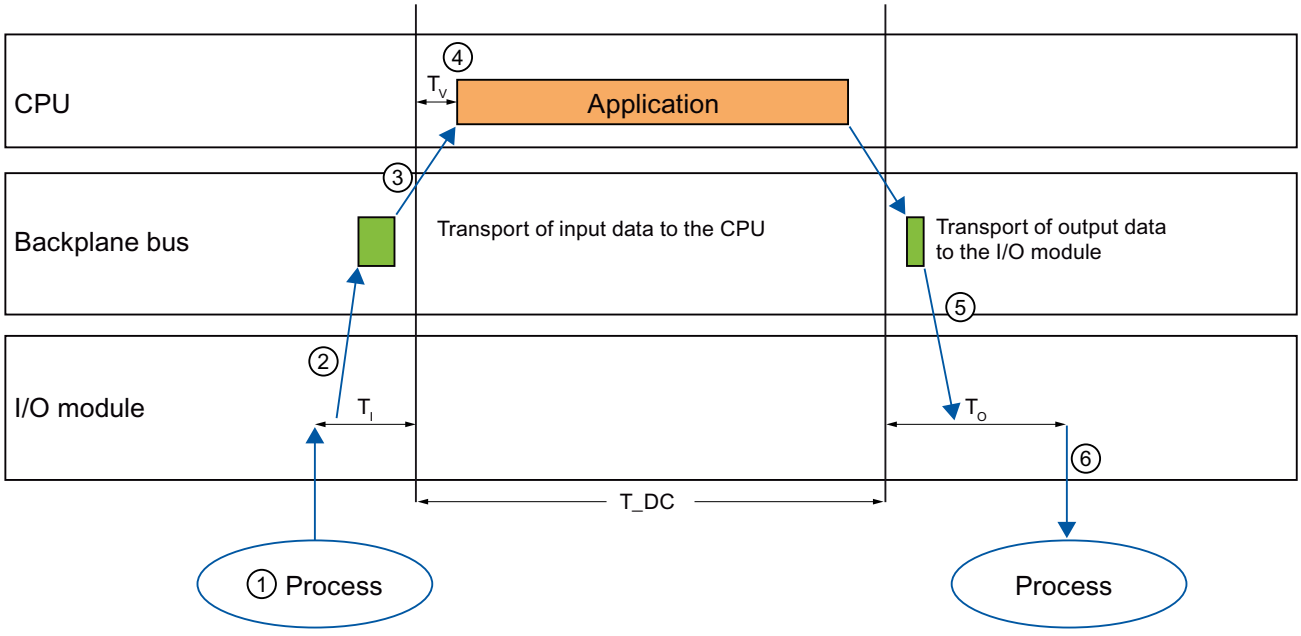

So that all input data is ready for transport to the CPU at the next start of the cycle, the start of the I/O read-in cycle is advanced by the amount of lead time T_I. T_I is the "flashbulb" for the inputs. After that, all synchronized inputs are read in. T_I is necessary in order to compensate for analog-to-digital conversion and the like. The lead time T_I can be configured by STEP 7 or by you. Let the lead time T_I be assigned automatically by STEP 7. With the default setting, STEP 7 ensures that a common, minimum T_I is set.

The backplane bus transports the input data to the CPU. The application is started synchronized to the cycle. That is, the isochronous mode interrupt OB is called after a configurable delay time T_V. The user program in the isochronous mode interrupt OB defines the process response and provides the output data in time for the start of the next data cycle. The length of the data cycle (send clock) is always configured by you.

Within time T_O, the data is:

- Transported to the I/O module over the backplane bus
- Processed in the I/O module, e.g. converted to an analog value

After expiration of time T_O, the data is output to the process.

Time T_O can be configured by STEP 7 or by you. Let time T_O be assigned automatically by STEP 7. STEP 7 automatically calculates a common, minimum T_O.

Organization blocks

OB start information

Structure of the OB start information

When you configure the properties of an OB for S7-1500 CPUs, you also specify the structure of the OB start information:

- As for S7-300 and S7-400 CPUs
- Optimized start information

OB start information as for S7-300 and S7-400 CPUs

The OB start information is part of the local data and has the same structure as for S7-300 and S7-400 CPUs.

Optimized start information

The optimized start information is a set of input tags.

It only provides the information that is necessary to identify the OB start event.

Semantically identical information is displayed for various OBs under the same tag name within the optimized start information.

Description of all types of organization blocks

Startup OBs

Description

The operating system calls each startup OB once at the transition from "STOP" to "RUN" mode. If several startup OBs are available, they are called in the order of their OB numbers, starting with the OB that has the lowest OB number.

The entirety of all startup OBs is referred to as the startup routine.

You can define default settings for your cyclic program in the startup routine.

All values of the process image input have the value zero here. There is no time limit for executing the startup routine. Time-driven or interrupt-driven organization blocks cannot be used.

After the startup routine has been completed, the operating system reads in the process image input and starts the cyclic program.

Structure of the start information

A startup OB has the following start information:

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - OB_NUMBR=B#16#FF, if the OB number > 254
- Optimized start information:

Name	Data type	Meaning
LostRetentive	BOOL	=TRUE, if the contents of retentive data areas have been lost
LostRTC	BOOL	=TRUE, if the time of the real-time clock has been lost

See also

Events and OBs (Page 927)

OB start information (Page 999)

Cyclic OBs

Description

The essential basis of the user program is the cyclic program. Its associated program parts are processed cyclically: When the processing has been completed, the operating system starts processing it again.

One or more cycle OBs belong to the cyclic program. These have priority 1. This is the lowest priority of all OBs. The cyclic program can be interrupted by events of any other event class.

When you have created several program cycle OBs, these are called one after the other in the order of their OB numbers. The program cycle OB with the lowest OB number is called first.

The following events start the cyclic program:

- End of startup processing
- End of the processing of the previous run of the cyclic program

After the cyclic program is complete, the operating system updates the process images as follows:

1. It writes the values from the process image output to the output modules.
2. It reads the inputs at the input modules and transfers these to the process image input.

Cycle time monitoring

Cycle time refers to the runtime of the cyclic program, including the runtime of all nested program parts of higher-priority OBs. If you have created several program cycle OBs, each program cycle OB contributes to the cycle time.

The operating system monitors whether the cycle time remains smaller than the configured maximum cycle time. If the cycle time exceeds the maximum cycle time, OB 80 is called (start event W#16#0002:3501). You can prevent this by restarting time monitoring at an appropriate point in program by calling the RE_TRIGR instruction. If the maximum cycle time is exceeded a second time within a cycle (without restarting time monitoring), the CPU switches to STOP mode (event W#16#0002:3500).

Minimum cycle time

Apart from monitoring the maximum cycle time, it is also possible to guarantee a minimum cycle time. To do this, the operating system delays the start of a new cycle until the minimum cycle time has been reached.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - SCAN_1=B#16#01 at the first call and B#16#03 at each further call
  - OB_NUMBR=B#16#FF, if the OB number > 254
- Optimized start information:

Name	Data type	Meaning
Initial_Call	BOOL	=TRUE in the first call of this OB at: • Transition from STOP or HOLD to RUN • After reloading
Retentivity	BOOL	=TRUE, if retentive data is available

See also

Events and OBs (Page 927)

Evaluating the value status (Page 919)

OB start information (Page 999)

Time-of-day interrupt OBs

Description

The S7-1500 CPUs provide time interrupt OBs, which can be started as follows:

- Once at a specific time (date with time)
- Periodically (the following time intervals are possible in this case: every minute, hourly, daily, weekly,
  monthly, end of month, yearly)

Starting a time interrupt OB

The following requirements must be met in order for the CPU to start a time interrupt OB:

- You must have previously set and activated the associated time interrupt. (Setting a time interrupt means specifying its start time and its duration.)
- You must have loaded the time interrupt OB to your CPU.

To set and activate a time interrupt, you have the following three options:

- You set and activate the time interrupt with STEP 7.
- You set the time interrupt with STEP 7 and activate it by calling the ACT_TINT instruction from your program.
- You set the time interrupt by calling the SET_TINTL instruction and activate it by calling the ACT_TINT instruction from within your program.

Rules for time interrupts

- If you set a time interrupt in such a way that the corresponding OB is to be processed once, the start time must not be in the past (relative to the real-time clock of the CPU).
- If you set a time interrupt in such a way that the corresponding OB is to be processed periodically, but the start time is in the past, then the time interrupt OB is processed the next time it is due according to the current time.
- The date of periodic time interrupts must correspond to a real date. For example, the monthly repetition of a time interrupt OB with start date 1/31 is therefore not possible. In this case, an OB is only started in the months that have 31 days.
- A time interrupt activated during startup is not executed until the startup has been completed.
- A startup deletes all time interrupts that were set and activated by an instruction in the user program.
- In the event of a transition from HOLD to RUN, the operating system checks whether there are elapsed time interrupts. If so, it calls the time error OB. If you have selected the same start information for it as for S7-300 and S7-400 CPUs, it enters whether time interrupts have been missed for the OBs 10 to 17 (and only for these). After the processing of OB 80 has been completed, the first missed OB is executed subsequently.

Behavior of time interrupts when time is adjusted ahead or back

The characteristics of the time interrupts depend on the time difference by which the time is set ahead or back. This is explained in more detail in the following table.

Changing the time	Effect on time interrupt
< 20 s	• When the time is set ahead: Every skipped time interrupt is executed subsequently. The start information tag "CaughtUp" is set. OB 80 is not called. • When the time is adjusted back: A time interrupt that was already executed in the reset time range or is currently active is not repeated.
> > = 20 s	• When the time is set ahead: If one or more time interrupts were skipped, OB 80 is called. For each priority, the start information provides information on whether at least one time interrupt was skipped. If you do not delete the time interrupt in OB 80, the first skipped time interrupt is executed subsequently and the start information tag "CaughtUp" is set. • When the time is adjusted back: The time interrupts in the range of the reset time are repeated. If the time correction is greater than the duration of a time interrupt, the operating system calculates the time of the first time interrupt to be repeated. In the case of a repeated time interrupt, the start information tag "SecondTime" is set.
Changing from standard to daylight saving time and vice versa	This change is only effective if you use local time. • When there is a change from standard to daylight saving time, a skipped time interrupt is executed subsequently and the start information tag "CaughtUp" is set. OB 80 is not called. • When there is a change from daylight saving time to standard time: the time interrupts in the range of the reset time are repeated. In the case of a repeated time interrupt, the start information tag "SecondTime" is set.
Changing the time zone (results in change of the local time)	The start time of the active time interrupts is recalculated. • When the local time is adjusted ahead, a skipped time interrupt is executed subsequently and the start information tag "CaughtUp" is set. OB 80 is not called. • When the local time is adjusted back, time interrupts that have already been executed may be repeated. In the case of repeated time interrupts, the start information tag "SecondTime" is set.

Time monitoring

When working with time interrupts, the following events can occur due to time monitoring by the operating system:

- W#16#0002:3504
  The time was adjusted ahead by the change from standard to daylight saving time. The start time of time interrupts was skipped during this process. OB 80 is called.
- W#16#0002:3505
  The time was adjusted ahead. The start time of time interrupts was skipped during this process. OB 80 is called.
- W#16#0002:3507
  Buffer overflow for OB start events. An entry is made in the diagnostic buffer.
- W#16#0002:3502
  Number of pending OB start events violates a preset limit. OB 80 is called.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  – STRT_INF=B#16#1F, if the OB number > 122
  – OB_NUMBR=B#16#FF, if the OB number > 254
- Optimized start information:

Name	Data type	Meaning
CaughtUp	BOOL	=TRUE if an OB call was executed subsequently because time was adjusted ahead
SecondTime	BOOL	=TRUE if an OB was called a second time because time was adjusted back. Note: SecondTime is set only once in the cases where time is adjusted back.

See also

Parameter assignment for time-of-day interrupt OBs (Page 1030)

Events and OBs (Page 927)

OB start information (Page 999)

Status interrupt OB

Description

When it receives a status interrupt, the operating system of the S7-1500 CPU calls the status interrupt OB. This may be the case if a module of a slave changes its operating state, for example, from "RUN" to "STOP". For more detailed information on events that trigger a status interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame: • PROFIBUS: – Bit 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bit 3 to 7: Seq. no. – Bit 8 to 15: Reserved • Central and PNIO: – Bits 0 to 10: Sequence number (value range 0 to 2047) – Bits 11 to 15: 0

See also

Events and OBs (Page 927)

OB start information (Page 999)

Update interrupt OB

Description

When it receives an update interrupt, the operating system of the S7-1500 CPU calls the update interrupt OB. This may be the case if you changed a parameter on a slot of a slave or device. For more detailed information on events that trigger an update interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame: • PROFIBUS: – Bit 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bit 3 to 7: Seq. no. – Bit 8 to 15: Reserved • Central and PNIO: – Bits 0 to 10: Sequence number (value range 0 to 2047) – Bits 11 to 15: 0

See also

Events and OBs (Page 927)

OB start information (Page 999)

OB for manufacturer- or profile-specific interrupt

Description

When it receives a manufacturer- or profile-specific interrupt, the operating system of the S7-1500 CPU calls OB 57. For more detailed information on events that trigger this type of interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  – STRT_INF=B#16#67
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame: • PROFIBUS: – Bit 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bit 3 to 7: Seq. no. – Bit 8 to 15: Reserved • Central and PNIO: – Bits 0 to 10: Sequence number (value range 0 to 2047) – Bits 11 to 15: 0

See also

Events and OBs (Page 927)

OB start information (Page 999)

Time-delay interrupt OBs

Description

A time-delay interrupt OB is started after a configurable time delay of the operating system. The time delay begins to elapse when the "SRT_DINT" instruction is called (this is the start of the associated time-delay interrupt).

Before a time-delay interrupt OB can be called by the operating system, the following conditions must be met:

- The time-delay interrupt must be started by calling the "SRT_DINT" instruction.
- The time-delay interrupt OB must not be deselected during configuration.

The time-delay interrupt OB must exist in the CPU.

Influencing time-delay interrupts with instructions

You can use the "CAN_DINT" instruction to cancel a time-delay interrupt which has already been started and whose associated time-delay interrupt OB has not yet been called by the operating system and use the "SRT_DINT" instruction to restart it (and thereby assign a different delay time to it). In addition, you can use the "QRY_DINT" instruction to query the current status of a time-delay interrupt.

You can disable and re-enable time-delay interrupts using the "DIS_AIRT" and "EN_AIRT" instructions.

Note

If you block this interrupt with "DIS_AIRT" after the execution of "SRT_DINT", processing is only continued after enabling with "EN_AIRT". This means that the delay time is extended accordingly.

Calling error OBs in connection with time-delay interrupt OBs

If the delay time of a time-delay interrupt expired before the processing of the associated time-delay interrupt OB was completed, the operating system generates a time error (calls OB 80).

Processing of time-delay interrupts depending on operating mode

"SRT_DINT" can be called and the delay time can elapse in both startup and in RUN mode. Time-delay interrupt OBs, however, are only processed in RUN mode. If a time-delay interrupt starts in a startup OB and the delay time has expired before the CPU reaches RUN mode, the call of the corresponding time-delay interrupt OB is delayed until the CPU is in RUN mode.

A warm or cold restart clears all start events for a time-delay interrupt OB.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - STRT_INF=B#16#2F, if the OB number > 122
  - OB_NUMBR=B#16#FF, if the OB number > 254
- Optimized start information:

Name	Data type	Meaning
Sign	WORD	User ID: Input parameter SIGN from the call of the "SRT_DINT" instruction

See also

Events and OBs (Page 927)

OB start information (Page 999)

Cyclic interrupt OBs

Description

You can start program parts at equidistant time intervals with the help of a cyclic interrupt OB.

The start times of a cyclic interrupt OB result as follows from its time interval and its phase offset:

Start times = n * time interval + phase offset (n = 0, 1, 2 ...)

- The time interval is the period of time between two calls. It is an integer multiple of the basic clock cycle of 1 µs.
- The phase offset is the time interval by which the start times are offset compared to the multiple of the time interval.
  A phase offset may be advisable when you are using multiple cyclic interrupt OBs. When their time intervals have a common multiple, for example, you can use a phase offset to prevent simultaneous start times.

Example of the use of a phase offset

You are using two cyclic interrupt OBs in your program: one OB with a time interval of 20 ms and one with a time interval of 100 ms. Use a phase offset to ensure that these are not called simultaneously at integer multiples of 100 ms.

Call of error OBs in connection with cyclic interrupt OBs

The runtime of a cyclic interrupt OB must be significantly smaller than its time interval. Otherwise, it could still happen that the time interval of a cyclic interrupt has expired but the corresponding cyclic interrupt OB is still being processed. In this case, the operating system generates a time error (calls OB 80). The cyclic interrupt that caused the error is executed later or discarded. However, only one cyclic interrupt OB for each priority can be executed later.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - STRT_INF=B#16#3F, if the OB number > 122 and the phase offset and cycle time are specified in ms
  - OB_NUMBR=B#16#FF, if the OB number > 254
- Optimized start information

Name	Data type	Meaning
Initial_Call	BOOL	=TRUE in the first call of this OB • At the transition from STOP or HOLD to RUN • After reloading
Event_count	INT	Number of discarded start events since the last start of this OB

See also

Assigning parameters to cyclic interrupt OBs (Page 1031)

Events and OBs (Page 927)

OB start information (Page 999)

Hardware interrupt OBs

Description

You can use hardware interrupt OBs to react to those events of modules with hardware interrupt capability which trigger a hardware interrupt. You define which events these are by means of configuration.

Note

Use of hardware interrupts

If possible, use hardware interrupts only to react to sporadically occurring events of modules with hardware interrupt capability.

The usage of hardware interrupts to react to events recurring often or very often does not make sense and can result in timeouts on your CPU under unfavorable conditions.

You can assign only one hardware interrupt OB to each event that triggers a hardware interrupt. Multiple events, however, can be assigned to one hardware interrupt OB (in extreme cases, all events of a module triggering a hardware interrupt).

For S7-1500 modules, individual channels can trigger hardware interrupts (e.g., a binary input on a rising edge).

Functionality of a hardware interrupt OB

After a module or submodule has triggered a hardware interrupt, the operating system determines the associated event and the associated hardware interrupt OB. If the currently active OB has a lower priority than this hardware interrupt OB, the hardware interrupt OB is started. Otherwise, the hardware interrupt OB is positioned in the queue that corresponds to its priority. The hardware interrupt is acknowledged after the completion of the corresponding hardware interrupt OB.

If another process event occurs on the same module or submodule during the time between identification and acknowledgment of a hardware interrupt, the following applies:

- If the event occurs on the channel that triggered the current hardware interrupt, the relevant hardware interrupt is lost. Another hardware interrupt can only be triggered by this channel when the current hardware interrupt is acknowledged.
- If the event occurs on a different channel, a hardware interrupt is triggered.

There may be additional restrictions for distributed I/O: for example, that a module or submodule can trigger only one interrupt of an interrupt type until the current interrupt has been acknowledged.

Hardware interrupt OBs are called only in RUN mode of the CPU.

Structure of the start information

- As for S7-300 and S7-400 CPUs: The USI, IChannel and EventType tags (explanation can be found in the optimized start information) are copied to "OB40_POINT_ADDR".
  See AUTOHOTSPOT
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - STRT_INF=B#16#41
  - OB_NUMBR=B#16#FF, if the OB number > 254
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the module that triggers the hardware interrupt
USI	WORD	Identifier for future extensions (not user-relevant)
IChannel	USINT	Number of the channel that triggered the hardware interrupt
EventType	BYTE	Identifier for the event type to which the event that triggered the interrupt belongs (e.g., rising edge) This identifier can be found in the description of the respective module.
PointAddr (Version V14.0 or higher of the TIA Portal)	DWORD	• In the case of digital modules: Bit array containing the inputs on the module that have triggered the hardware interrupt. Refer to the respective module description for information on which bit of Point_Addr is assigned to which module channel. • In the case of analog modules: Bit field with the information as to which channel has exceeded which threshold. Refer to the module description for the precise structure. • In the case of CPs or IMs: Interrupt status of the module (not user-relevant)

Note

USI has the value 0xFFFF

If you have configured the affected module in HW Config with GSDML resources, USI has the value 0xFFFF. In this case, IChannel and EventType do not get valid values. To determine the event that triggered the interrupt, proceed as follows:

You either use the structure of the start information as for S7-300 and S7-400 CPUs,

or you use the optimized start information and call the RALRM instruction in the hardware interrupt OB.

Opening, compiling and loading of hardware interrupt OBs

Because the optimized start information of the hardware interrupt OB in version 14.0 or higher of the TIA Portal includes an additional tag, constellations may result in which opening, compiling and loading of the hardware interrupt OBs to the CPU is no longer possible. These constellations are listed below:

- You copy a hardware interrupt OB with a new interface to an S7-1500 CPU with firmware version < V2.0 or to an S7-1200-CPU in the engineering system. The OB can then no longer be opened, compiled and loaded.
- You replace an S7-1500 CPU with firmware version >= V2.0 that includes a hardware interrupt OB with new interface with an S7-1500 CPU with firmware version < V2.0 in the engineering system. The OB can then no longer be opened, compiled and loaded.

However, no problems arise during opening, compiling and loading in the following constellations:

- You copy a hardware interrupt OB with old interface to an S7-1500 CPU with firmware version >= V2.0 in the engineering system. The hardware interrupt OB retains the old interface.
- You replace an S7-1500 CPU with firmware version < V2.0 that includes a hardware interrupt OB with old interface with an S7-1500 CPU with firmware version >= V2.0 in the engineering system. The hardware interrupt OB retains the old interface.

See also

Assigning parameters to hardware interrupt OBs (Page 1032)

Events and OBs (Page 927)

Example of a hardware interrupt event (Page 930)

OB start information (Page 999)

Isochronous mode interrupt OBs

Description

Isochronous mode interrupt OBs are used to start subprograms isochronously to the DP cycle or PN send clock.

You update the associated process image partition of the inputs and the associated process image partition of the outputs by calling the instructions SYNC_PI and SYNC_PO in the isochronous mode interrupt OB.

CAUTION
Direct access In cases of direct access, avoid access to I/O areas to which process image partitions with isochronous mode interrupt OB connection are assigned.

Note

Effectiveness of the OB parameter "Report event overflow into diagnostic buffer"

When you select the check box "Report event overflow into diagnostic buffer" for the OB parameters, this does not become effective until you have loaded the OB properties to the CPU and the CPU has then performed a STOP-RUN transition.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - STRT_INF=B#16#6F, if the OB number > 122
  - OB_NUMBR=B#16#FF, if the OB number > 254
- Optimized start information:

Name	Data type	Meaning
Initial_Call	BOOL	=TRUE in the first call of this OB during transition from STOP to RUN
PIP_Input	BOOL	TRUE: The associated process image input is up-to-date Note: For S7-1500 CPUs, this variable always has the value FALSE, since updating the corresponding process image partition of the inputs must be carried out by calling the SYNC_PI instruction.
PIP_Output	BOOL	TRUE: The associated process image output was transferred to the outputs in good time after the last cycle Note: For S7-1500 CPUs, this variable always has the value FALSE, since updating the corresponding process image partition of the outputs must be carried out by calling the SYNC_PO instruction.
IO_System	USINT	Number of the distributed I/O system triggering the interrupt
Event_Count	INT	• = n: Number of lost cycles • = -1: An unknown number of cycles has been lost (e.g., because cycle has changed).
SyncCycleTime	LTIME	Calculated cycle time

See also

Assigning parameters to isochronous mode interrupt OBs (Page 1033)

Events and OBs (Page 927)

OB start information (Page 999)

Time error OB

Description

The S7-1500 CPU operating system calls the time error OB if one of the following errors occurs:

- First violation of the cycle time within a cycle

  Note

  If the cycle time is violated twice in the same cycle, the CPU goes to STOP mode. You can prevent this by calling the "RE_TRIGR" instruction at a suitable point.

- The number of requested but not completely processed OB requests has reached the configured warning limit. (For migrated programs, this warning limit set to 1.)
  If you have set the warning limit to 1, this means: If a start event, e.g., for a cyclic interrupt OB, occurs before the previous processing of the same OB has been completed, the operating system calls OB 80.
  This applies to all events whose associated OB can cause a time error.
- Expired time interrupt on re-entry into RUN mode after the CPU was, for example, in HOLD
- Skipped time interrupt due to the time being adjusted more than 20 s ahead (also in the case of changeover from standard time to daylight saving time)

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
- Optimized start information

Name	Data type	Meaning
Fault_ID	BYTE	Error code. Possible values: • B#16#01: Cycle time exceeded • B#16#02: The requested OB is still being processed. • B#16#05: Expired time interrupt due to time jump • B#16#06: Expired time interrupt upon re-entry into RUN after HOLD • B#16#07: Overflow of the OB request buffer for the current priority class • B#16#08: Isochronous mode interrupt - time error • B#16#09: Interrupt loss due to high interrupt load • B#16#0B: Technology synchronization interrupt - time error • b#16#0C: (only for S7-1500 R/H CPUs) cycle time exceeded for the second time
Csg_OBnr	OB_ANY	Number of the OB that was being processed when the time error occurred
Csg_Prio	UINT	Priority of the OB that was being processed when the time error occurred

Note

Csg_OBnr=0 and Csg_Prio=0

If both the Csg_OBnr and Csg_Prio (or OB80_ERR_EV_CLASS and OB80_ERR_EV_NUM, if you have selected the S7-300- und S7-400-CPUs as structure of the start information) had the value zero, the CPU performed background activities as the time error occurred.

See also

Events and OBs (Page 927)

OB start information (Page 999)

Diagnostic interrupt OB

Description

The operating system of the S7-1500 CPU calls the diagnostic interrupt OB in the following cases:

- A diagnostics-capable S7-1500 module detects a change in its diagnostic status and sends a diagnostic interrupt request to the CPU.
- A diagnostics-capable S7-300 or S7-400 module for which you have enabled the diagnostic interrupt detects a change in its diagnostic status and sends a diagnostic interrupt request to the CPU.
- An event occurs that was triggered by an error in the power supply or the battery backup.

  Note

  With the S7-300 and S7-400 CPUs, such an event causes the power supply error OB to be called.

- The operating system has detected memory errors.

  Note

  With the S7-300 and S7-400 CPUs, such an event causes the CPU hardware error OB to be called.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
- Optimized start information:

Name	Data type	Meaning
IO_state	WORD	Status of the hardware object: • Bit 0: Good • Bit 1: Deactivated • Bit 2: Maintenance required • Bit 3: Maintenance demanded • Bit 4: Error • Bit 5: Not available • Bit 6: Qualified • Bit 7: Not available • Bits 8 to 14: Reserved (always "0") • Bit 15: Network / hardware fault If bit 4 = 1 or bit 5 = 1: – Bit 15 = 0: Network error – Bit 15 = 1: Hardware error
LADDR	HW_ANY	Hardware identifier of the hardware object that triggered the diagnostic interrupt
Channel	UINT	Channel number
MultiError	BOOL	=TRUE, if there is more than one error

See also

Events and OBs (Page 927)

OB start information (Page 999)

Pull/plug interrupt OB

Description

The S7-1500 CPU operating system calls the pull/plug interrupt OB if a configured and non-disabled distributed I/O module or submodule is removed or inserted.

Note

The removing or inserting of a central module leads to STOP of the CPU, except for CPUs 1510SP-1 PN and 1512SP-1 PN.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the affected module or submodule
Event_Class	BTYE	• B#16#38: (Sub)module plugged • B#16#39: (Sub)module pulled or not responding
Fault_ID	BYTE	Error code (possible values: B#16#51, B#16#54, B#16#55, B#16#56, B#16#57, B#16#58, B#16#61, B#16#63, B#16#64, B#16#65, B#16#66)

The following table shows which event caused the start of the pull/plug interrupt OB.

ev_class B#16# ...	fault_id B#16# ...	Meaning
39	51	IO module removed
39	54	IO submodule removed
38	54	IO submodule inserted and matches configured submodule
38	55	IO submodule inserted, but does not match configured submodule
38	56	IO submodule inserted, but error in module parameter assignment
38	57	IO submodule or module inserted, but with fault or maintenance
38	58	IO submodule, access error corrected
39	61	Module removed or not responding
38	61	Module inserted

See also

Events and OBs (Page 927)

OB start information (Page 999)

Rack error OB

Description

The S7-1500 CPU operating system calls the OB 86 in the following cases:

- The failure of a DP master system or of a PROFINET IO system is detected (in the case of either an incoming or an outgoing event).
- The failure of a DP slave or of an IO device is detected (in the case of either an incoming or an outgoing event).
- Failure of some of the submodules of a PROFINET I-device is detected.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_Device	Hardware identifier of the defective hardware object
Event_Class	BYTE	• B#16#38: outgoing event • B#16#39: incoming event
Fault_ID	BYTE	Error code (possible values: B#16#C3, B#16#C4, B#16#C9, B#16#CA, B#16#CB, B#16#CC, B#16#CD, B#16#CE, B#16#CF, B#16#F8, B#16#F9)

The following table shows which event caused the start of OB 86.

ev_class B#16# ...	fault_id B#16# ...	Meaning
39	C3	Failure of a DP master system
39/38	C4	Failure/return of a DP slave
39	CA	Failure of a PROFINET IO system
39/38	CB	Failure/return of a PROFINET IO device
38	CC	Return of a PROFINET IO device with fault or maintenance
38	CD	Return of a PROFINET IO device, deviation between preset/actual configuration
38	CE	Return of a PROFINET IO device, error in module configuration
39/38	F8	Failure/return of some of the submodules of a PROFINET I-device
38	F9	Return of some of the submodules of a PROFINET I-device with a device configuration difference

See also

Events and OBs (Page 927)

OB start information (Page 999)

Programming error OB

Requirement for calling the programming error OB

Global error handling must be set if you want to enable the S7-1500 CPU to call the programming error OB.

Description

The S7-1500 CPU operating system calls the programming error OB if a programming error occurs during the processing of an instruction of the user program. The programming error OB is processed with the priority set for it.

Note

If you have set local error handling, the programmed error handling in the corresponding block becomes effective when a programming error occurs.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
- Optimized start information:

Name	Data type	Meaning
BlockNr	UINT	Number of the block in which the programming error occurred
Reaction	USINT	• 0: Ignore errors • 1: Replace incorrect value • 2: Skip command • 3: Programmed error handling, triggered by an array access with invalid index, for example, or by an error during parameter supply of an FC or FB
Fault_ID	BYTE	Error code (possible values: B#16#00, B#16#03, B#16#04, B#16#05, B#16#20, B#16#21, B#16#22, B#16#23, B#16#24, B#16#25, B#16#26, B#16#27, B#16#28, B#16#29, B#16#2C, B#16#30, B#16#31, B#16#32, B#16#33, B#16#34, B#16#35, B#16#38, B#16#39, B#16#3A, B#16#3B, B#16#3C, B#16#3D, B#16#3E, B#16#3F, B#16#50, B#16#51, B#16#75, B#16#76, B#16#A1, B#16#A2)
BlockType	USINT	Type of block in which the error occurred: • OB: 1 • FC: 2 • FB: 3 • SFC: 4 • SFB: 5 • DB: 6

Name	Data type	Meaning
Area	USINT	Area in which the incorrect access occurred: • Local data: B#16#40 to 4E, 86, 87, 8E, 8F, C0 to CE • Process image input: B#16#01 • Process image output: B#16#02 • Technology DB: B#16#04 • I: B#16#81 • Q: B#16#82 • M: B#16#83 • DB: B#16#84, 85, 8A, 8B
DBNr	Block_DB	DB no. if AREA = DB or DI
Csg_OBNr	OB_ANY	OB number (121)
Csg_Prio	USINT	OB priority
Width	USINT	Type of access during which the error occurred: • Bit: – B#16#00 for access to the standard memory area – B#16#01 for access to the optimized memory area • Byte: B#16#01 • Word: B#16#02 • DWord: B#16#03 • LWord: B#16#04

The following table shows the meaning of the error codes:

fault_ID B#16# ...	Meaning
00	Maximum nesting depth of block calls exceeded
03	You have used a NULL pointer to address an operand.
04	Unknown instruction
05	Stop instruction
20	The addressed string has incorrect length information.
21	BCD conversion error
22	Area length error when reading
23	Area length error when writing
24	Area error when reading
25	Area error when writing
26	Error in timer no.
27	Error in counter no.
28	Read access to a byte, word or double word with a pointer whose bit address is not zero
29	Read access to a byte, word or double word with a pointer, whose bit address is not zero
2C	You have used a NULL pointer to address an operand.
30	Write access to a write-protected global DB

fault_ID B#16# ...	Meaning
31	Write access to a write-protected instance DB
32	DB number error when accessing a global DB
33	DB number error when accessing an instance DB
34	Number error during FC call
35	Number error during FB call
38	The type-safe data block is not loaded.
39	The data block is not defined as type-safe.
3A	Access to a DB that is not loaded; the DB number is located in the permitted area
3B	DB does not exist
3C	Access to an FC that is not loaded; the FC number lies in the permissible area
3D	Access to an SFC that is not loaded; the SFC number lies in the permissible area
3E	Access to an FB that is not loaded; the FB number lies in the permissible area
3F	Access to an SFB that is not available; the SFB number lies in the permissible area
50	The type-safe data block is not loaded.
51	The data block is not defined as type-safe.
75	The maximum nesting depth for block calls was exceeded.
76	The maximum of available local data was exceeded.
A1	Write access to a write-protected tag
A2	Access to a tag that contains an invalid numerical value

See also

Events and OBs (Page 927)

OB start information (Page 999)

I/O access error OB

Requirement for call of the I/O access error OB

Global error handling must be set to enable the S7-1500 CPU to call the I/O access error OB.

Description

The S7-1500-CPU operating system calls the I/O access error OB if an error occurs during direct access to the I/O data while an instruction of the user program is being executed. This, for example, is the case if a read error occurs during direct access to data of an input module. The I/O access error OB is processed with the priority assigned to it.

Note

If local error handling is set, the error handling characteristics programmed in the associated block take effect if an I/O access error occurs.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See AUTOHOTSPOT
- Optimized start information:

Name	Data type	Meaning
BlockNr	UINT	Number of the block in which the I/O access error occurred
Reaction	USINT	0: Ignore error, 1: Replace incorrect value, 2: Skip command
Fault_ID	BYTE	Error code: • B#16#42: I/O access error, reading • B#16#43: I/O access error, writing
BlockType	USINT	Type of block in which the error has occurred: • OB: 1 • FC: 2 • FB: 3 • SFC: 4 • SFB: 5 • DB: 6
Area	USINT	Identifier for the range in which the incorrect access occurred: • B#16#01: Direct access to input • B#16#02: Direct access to output • B#16#81: Access to process image input • B#16#82: Access to process image output
DBNr	Block_DB	not user-relevant
Csg_OBNr	OB_ANY	OB number causing the I/O access error
Csg_Prio	USINT	OB priority that causes the I/O access error
Width	USINT	Type of access during which the error occurred: • Bit: B#16#00 • Byte: B#16#01 • Word: B#16#02 • DWord: B#16#03 • LWord: B#16#04

See also

Events and OBs (Page 927)

OB start information (Page 999)

ProDiag OB

Description

With ProDiag, you assign a ProDiag function block to each configured supervision. Supervisions for a ProDiag FB are, however, not enabled until the block is called. The call can be made either in the ProDiag organization block or at any other point in your program.

- If you explicitly call a ProDiag FB in your program, it is not called in the ProDiag OB.
  If you explicitly call all ProDiag FBs in your program, no ProDiag OB is created.
- If you do not want to manage calling one or more ProDiag FBs and do not call them at any point in your program, this is detected when your project is compiled. In this case, a ProDiag OB is automatically created and the ProDiag FBs are called in it.
  If you do not call any ProDiag FB of your project explicitly in your program, all ProDiag FBs are called in the ProDiag OB.

Note

Creating a ProDiag OB

The ProDiag OB is the only organization block that is created automatically.

You can read out the information which ProDiag FBs are called in the ProDiag OB with the help of the cross-references.

Motion Control OBs (S7-1500, S7-1500T)

MC-servo OB (S7-1500, S7-1500T)

Description

When you create a technology object for S7-1500 Motion Control, the organization block "MC-Servo [OB91]" for processing the technology objects is created automatically. The motion control functionality of the technology objects creates its own priority class and is called in the SIMATIC S7-1500 execution system according to the application cycle.

The MC-Servo OB is write-protected. The contents cannot be changed.

The position control algorithms of all technology objects configured for motion control on the CPU are calculated within the MC-Servo OB.

You can set the application cycle and the priority of the organization block in accordance with your requirements for control quality and system load.

Application cycle

You can set the application cycle in which the MC-Servo OB is called in the properties of the organization block:

- Synchronous to the BUS
  The MC-Servo OB is called synchronously with a bus system. You set the send clock in the properties of the selected bus system. You can select the following bus systems:
  - Isochronous PROFIBUS DP
  - Isochronous PROFINET IO

  You cannot call the MC-Servo OB synchronously with a bus system that is connected to the CPU via a communication processor/communication module (CP / CM).
- Cyclical
  The MC-Servo OB is called cyclically with the specified application cycle.

The selected application cycle must be long enough to allow all the technology objects for motion control to be processed in one cycle. If the application cycle is not observed, overflows occur. The CPU switches to STOP mode in the case of an overflow of the MC-Servo OB.

Clock reduction (CPU V1.5 and higher)

You can reduce the application cycle of the MC-Servo OB relative to the send clock of an isochronous PROFINET IO system. You can set an integer multiple of the send clock as the factor. Values up to 14 times the send clock (maximum 32 ms) are possible for the application cycle.

If you call a synchronous mode interrupt OB and the MC-Servo OB synchronously with the same PROFINET IO system, you must set the same application cycle for both organization blocks.

See also

Assigning the MC-Servo OB parameters (Page 1024)

MC-PreServo OB (Page 1026)

MC-PostServo OB (Page 1027)

MC-interpolator OB (Page 1026)

MC-Transformation OB (Page 1028)

Assigning the MC-Servo OB parameters (S7-1500, S7-1500T)

Procedure for setting the parameters

To assign the parameters of an MC-Servo OB, follow these steps:

1. Open the "Properties" dialog associated with the MC-Servo OB in question.
2. Click on the "Cycle time " group in the area navigation.

Overview of the parameters that can be set

In the "Cycle time" group, select how the MC-Servo OB will be called in the execution system of the SIMATIC S7-1500:

- Cyclical
- Synchronous to the bus

You can set the following parameters according to the option selected:

- Application cycle
- Distributed I/O
- Factor

"Cyclical" option

The MC-Servo OB is called cyclically with the specified application cycle.

You set the cycle in which the MC-Servo OB is called in the "Application cycle" field.

"Synchronous to the bus" option

The MC-Servo OB is called synchronously with a bus system. You set the send clock in the properties of the selected bus system.

The following parameters are available for the "Synchronous to the bus" option:

- Distributed I/O
  In the drop-down list, select the bus system for which the MC-Servo OB will be called isochronously. You can select the following bus systems:
  - Isochronous PROFINET IO
  - Isochronous PROFIBUS DP

  You cannot call the MC-Servo OB synchronously with a bus system that is connected to the CPU via a communication processor/communication module (CP / CM).
- Send clock (ms)
  The send clock or DP cycle that is set in the selected PROFINET IO system or DP master system is displayed in this field.
- Factor
  You can reduce the application cycle of the MC-Servo OB relative to the send clock of an isochronous PROFINET IO system. You set an integer multiple of the send clock in the "Factor" field. Values up to 14 times the send clock (maximum 32 ms) are possible for the application cycle.
- Application cycle (ms)
  An application cycle that results from the send clock and the specified factor is displayed in this field.

See also

MC-servo OB (Page 1023)

MC-interpolator OB (S7-1500, S7-1500T)

Description

When you create a technology object for S7-1500 Motion Control, the organization block "MC-Interpolator [OB91]" for processing the technology objects is created automatically. The motion control functionality of the technology objects creates its own priority class and is called according to the application cycle of the MC-Servo OB (Page 1023) in the execution system of the SIMATIC S7-1500.

The MC-Interpolator OB is write-protected. The contents cannot be changed.

The evaluation of the motion control instructions as well as the monitoring and the setpoint generation for all technology objects configured on the CPU (motion control) is performed within the MC-Interpolator OB.

The MC-Interpolator OB is called at the end of processing of the MC-Servo OB. The clock ratio between MC-Interpolator OB and MC-Servo OB is always 1:1.

Process response

You must select the application cycle of the MC-Servo OB long enough to allow all the technology objects for motion control to be processed in one application cycle. If the application cycle is not observed, overflows occur. Note the following response to overflows:

- The CPU tolerates a maximum of three consecutive overflows of the MC-Interpolator OB.
- The processing of an MC-Interpolator OB can be interrupted by the calling of the MC Servo OB, at most.

If more overflows or interruptions occur, the CPU switches to STOP mode.

See also

Assigning the MC-Servo OB parameters (Page 1024)

MC-Transformation OB (Page 1028)

MC-PreServo OB (S7-1500, S7-1500T)

Description

The organization block MC-PreServo [OB67] can be programmed and is called in the application cycle configured at the MC-Servo OB (Page 1023). MC-PreServo [OB67] is called directly before MC-Servo [OB91].

Via the organization block, you can read out the configured application cycle (information in µs).

Structure of the start information

Optimized start information:

Name	Data type	Meaning
Initial_Call	BOOL	=TRUE in the first call of this OB during transition from STOP to RUN
PIP_Input	BOOL	TRUE: The associated process image input is up-to-date
PIP_Output	BOOL	TRUE: The associated process image output was transferred to the outputs in good time after the last cycle
IO_System	USINT	Number of the distributed I/O system triggering the interrupt
Event_Count	INT	• = n: Number of lost cycles • = -1: An unknown number of cycles has been lost (e.g., because cycle has changed).
Synchronous	BOOL	TRUE: The MC-Servo OB is called synchronously with a bus system.
CycleTime	UDINT	Display of the application cycle configured for the MC-Servo OB in ns

MC-PostServo OB (S7-1500, S7-1500T)

Description

The organization block MC-PostServo [OB95] can be programmed and is called in the application cycle configured for the MC-Servo OB (Page 1023). MC-PostServo [OB95] is called directly after MC-Servo [OB91].

Via the organization block, you can read out the configured application cycle (information in µs).

Structure of the start information

Optimized start information:

Name	Data type	Meaning
Initial_Call	BOOL	=TRUE in the first call of this OB during transition from STOP to RUN
PIP_Input	BOOL	TRUE: The associated process image input is up-to-date
PIP_Output	BOOL	TRUE: The associated process image output was transferred to the outputs in good time after the last cycle
IO_System	USINT	Number of the distributed I/O system triggering the interrupt
Event_Count	INT	• = n: Number of lost cycles • = -1: An unknown number of cycles has been lost (e.g., because cycle has changed).
Synchronous	BOOL	TRUE: The MC-Servo OB is called synchronously with a bus system.
CycleTime	UDINT	Display of the application cycle configured for the MC-Servo OB in ns

MC-Transformation OB (S7-1500, S7-1500T)

Description

You program the user transformation of the Cartesian coordinates and the axis-specific setpoints for the user-defined kinematics in the organization block MC-Transformation [OB98]. This programming includes the transformation of the positions and the dynamic values (velocity, acceleration, jerk).

When you add the MC-Transformation [OB98] in the TIA Portal, the system data block "TransformationParameter" is automatically created under "Program blocks > System blocks > Program resources". In the properties of the organization block, the MC-Transformation [OB98] indicates the number of the data block "TransformationParameter" under "General > Transformation".

In the user program, supply the input parameters of the MC-Transformation [OB98] with the required start information. You write the axis-specific parameters or the Cartesian parameters in the "TransformationParameter" data block. The MC-Transformation [OB98] is called with this start information in the execution system of the CPU.

Note

Only one MC-Transformation [OB98] per CPU

You program all required user transformations in an organization block MC-Transformation [OB98]. If you require multiple transformations or multiple instances of a transformation, you need to identify the transformations or the instances in your user program.

Block call and priority

The MC-Transformation [OB98] is called according to the configured priority in the execution system of the CPU.

You configure the priority of the MC-Transformation [OB98] in the properties of the organization block under "General > Attributes > Priority". For the priority you can set values from 17 to 25 (default setting 25):

- The priority of MC-Transformation [OB98] must be at least one level lower than the priority of MC-Servo [OB91].
- The priority of MC-Transformation [OB98] must be at least one level higher than the priority of MC-Interpolator [OB92].

Parameter

The following table shows the parameters of the "MC-Transformation [OB98]":

Parameter	Declaration	Data type	Default value	Description	
KinematicsObject	INPUT	DB_ANY	-	Kinematics technology object for which the MC-Transformation [OB98] calculates the transformation when called.	
ExecutionContext	INPUT	DINT	-	Processing context of the MC-Transformation OB	
				0	MOTION_EXECUTION Calculation of the axis setpoints in the motion execution in the interpolator. The calculated values are necessary for the current motion control. Torque precontrol values can be calculated and preset.
				1	NON_MOTION_EXECUTION Calculation of axis setpoints from IPO task and other OBs are possible. The transformation is necessary for the motion planning.
TransformationType	INPUT	DINT	-	Calculation called for	
				0	Forward transformation Calculation of the Cartesian parameters from the axis positions
				1	Inverse transformation Calculation of the axis-specific parameters from the Cartesian parameters
TransformationParameters	InOut	VARIANT	-	Pointer to the data block "TransformationParameter"	
FunctionResult	OUTPUT	DINT	-	Return value of the MC-Transformation [OB98] to the kinematics technology object	
				0	Calculation performed and parameters output
				< 0	Error during calculation If an error occurs during the calculation, the kinematics technology object stops the motion. The kinematics technology object outputs a technology alarm with the error ID as an accompany value and deletes the job sequence.
				If an error occurs during the calculation, the kinematics technology object stops the motion. The kinematics technology object outputs a technology alarm with the error ID as an accompany value and deletes the job sequence.	

See also

MC-servo OB (Page 1023)

MC-interpolator OB (Page 1026)

Setting parameters of organization blocks

Parameter assignment for time-of-day interrupt OBs

Procedure for setting the parameters

To set the parameters of a time-of-day interrupt OB, proceed as follows:

1. Open the "Properties" dialog belonging to the time-of-day interrupt OB in question.
2. Click the "Time-of-day interrupt" group in the area navigation.

Overview of the parameters that can be set

You can set the following parameters:

- Execution
- Start date and time-of-day
- Option buttons "Local time" and "System time"

"Execution" parameter

Use the drop-down list "Execution" to define the periods at which the time-of-day interrupt OB is to be executed. The time intervals relate to the settings for "Start date" and "Time-of-day".

The following values are possible for "Execution":

- Never
- Once
- Every minute
- Hourly
- Daily
- Weekly
- Monthly
- Yearly
- End of month

Note

With the value "End of month", the value specified under "Start date" is irrelevant.

"Start date" and "Time-of-day" parameter

Here, you specify the time at which the time-of-day interrupt is to be executed for the first time.

Example: Start date = 01/05/2013, time =11:16

Depending on the value of the "Execution" parameter, the CPU generates additional time-of-day interrupts periodically. Depending on the setting, the start time relates either to the local time or to the coordinated universal timeUTC.

Note

If you set the "Execution" parameter to "Monthly", the start date cannot be set to the 29th, the 30th, or the 31st. If you want the time-of-day interrupt OB to start at the end of the month, you should set the parameter "Execution" to "End of month".

"Local time" or "System time"

Here, you decide which time the start time of the time-of-day interrupt OB relates to:

- "Local time": The start time relates to the time zone set for the CPU.
- "System time": The start time relates to the coordinated universal time UTC (Universal Time Coordinated).

See also

Time-of-day interrupt OBs (Page 1001)

Assigning parameters to cyclic interrupt OBs

Procedure for setting the parameters

To set the parameters of a cyclic interrupt OB, proceed as follows:

1. Open the "Properties" dialog belonging to the cyclic interrupt OB in question.
2. Click the "Cyclic interrupt" group in the area navigation.

Overview of the parameters that can be set in the "Cyclic interrupt" group

In the "Cyclic interrupt" group, you can set the following parameters:

- Cycle time
- Phase offset

"Cycle time" parameter

Use the parameter "Cycle time" to set the period of time between two calls of the cyclic interrupt OB. It is an integer multiple of 1 µs.

"Phase offset" parameter

Here, you set the time period by which the start times are shifted as compared to the multiple of the cycle time.

A phase offset may be advisable when you are using multiple cyclic interrupt OBs. If their cycle times have common multiples, you can use a phase offset to prevent simultaneous start times.

Priority of the cyclic interrupt OB

When you add a new cyclic interrupt OB to your project, you enter its cycle time. The priority of the cyclic interrupt OB is set in accordance with the following table.

Cycle time	Default priority
0.25 ms <= cycle time <= 2 ms	17
2 ms < cycle time <= 5 ms	16
5 ms < cycle time <= 10 ms	14
10 ms < cycle time <= 100 ms	13
100 ms < cycle time <= 200 ms	12
200 ms < cycle time <= 500 ms	11
500 ms < cycle time <= 1000 ms	10
1000 ms < cycle time <= 2000 ms	9
Cycle time > 2000 ms	8

You can change the default priority as follows:

1. Open the "Properties" dialog belonging to the cyclic interrupt OB in question.
2. Click the "Attributes" group in the area navigation.
3. In the "Priority" area, enter the desired value in the input field "Priority".

See also

Cyclic interrupt OBs (Page 1008)

Assigning parameters to hardware interrupt OBs

Procedure for configuration of the start events of a hardware interrupt OB

Each start event for a hardware interrupt OB is configured in the properties of the corresponding module. To do this, follow these steps:

1. Open the "Properties" dialog related to the corresponding module.
2. In the area navigation, click the channel that is to trigger a hardware interrupt.
3. Select those check boxes whose associated events are to trigger a hardware interrupt. Enter associated limit values, if any.
4. Enter an event name, for example, "Rising edge 0".

5. Select an existing hardware interrupt OB from the drop-down list "Hardware interrupt" or create a new hardware interrupt OB.
6. If necessary, enter the priority of the hardware interrupt OB.

A system constant of the data type "Event_HwInt" is created automatically for the event identified by its unique event name. This is displayed in the standard tag table in the "System constants" tab.

Overview of the start events of a hardware interrupt OB

To display the currently configured start events of a hardware interrupt OB, proceed as follows:

1. Open the "Properties" dialog associated with the hardware interrupt OB in question.
2. Click the "Start events" group in the area navigation.

See also

Hardware interrupt OBs (Page 1010)

Example of a hardware interrupt event (Page 930)

Assigning parameters to isochronous mode interrupt OBs

Procedure for setting the parameters

To set the parameters of an isochronous mode interrupt OB, proceed as follows:

1. Open the "Properties" dialog associated with the isochronous mode interrupt OB in question.
2. In the area navigation, click the "Isochronous mode" group .

Overview of the parameters that can be set

You can set the following parameters:

- Application cycle
- Automatic setting
- Delay time

"Application cycle" parameter

- For PROFINET: Integer multiple (possible values: 1, ... 14) of the send clock
- For PROFIBUS DP: DP cycle

Check box "Automatic setting"

- If the check box is cleared, you specify the delay time manually.
- If the check box is selected, the delay time is calculated automatically. The associated input box is then grayed out.

"Delay time" parameter

Note

You can only set the parameter "Delay time" if the check box "Automatic setting" is cleared.

The delay time is the time interval between the send clock or the global control frame and the start of the isochronous mode interrupt OB. In this period of time, the PROFINET IO controller or the DP master handles the cyclic data exchange with the IO devices or the DP slaves.

If the delay time is zero, the isochronous mode interrupt OB starts with the send cycle clock or the global control frame. In this case, it is extremely likely that access to the isochronous I/O via the extended instruction "SYNC_PI" at the start of the OB will fail as the I/O data is not yet available.

If the delay time is greater than the default value, then the OB runtime is "given away", since the clock-synchronized I/O could have been accessed earlier.

Overview of the displayed parameters

The following parameters are displayed:

- Distributed I/O
- Process image partition

"Distributed I/O"

Here, the PROFINET IO system or DP Master system containing the IO devices or DP slaves assigned to the isochronous mode interrupt OB are displayed.

The definition of a PROFINET IO system or DP master system takes place during the assignment of the process image partition to the isochronous mode interrupt OB.

"Process image partition"

Here, the process image partition assigned to the isochronous mode interrupt OB is displayed. This assignment has been made during hardware configuration.

See also

Isochronous mode interrupt OBs (Page 1012)

Useful information on CPU firmware versions and STEP 7 versions

Below you will find information on the compatibility of the CPU firmware versions and their dependency on different STEP 7 versions.

It is possible to program a CPU with new firmware (e.g. firmware V1.7, which can be configured as of STEP 7 V13 SP1) with a non-current STEP 7 version (e.g. STEP 7 V12, which only recognizes the firmware V1.0). In this case, the functionality of the CPU with firmware version V1.7 is restricted to the functionality of firmware version V1.0.

The S7-1500 CPUs are downward compatible with their direct predecessors.

The S7-1500 CPUs are not compatible to comparable S7-1500T CPUs. Example: A CPU 1511 is compatible to a CPU 1511T.

Conversely, compatibility only exists if you do not use any extended Motion Control functions.

Examples of downwards compatibility

- CPU 1516 V2.0 is compatible with all firmware versions of the CPU 1516.
  The CPU 1516 V2.0 is supplied with a new order number but is compatible without restrictions to the predecessor CPUs with the previous order number. You do not need to note anything.
- CPU 1516 V1.7 is compatible with the CPUs 1516 V1.6, V1.5, V1.1 and V1.0, but not compatible with all other S7-1500 CPUs, e.g. CPU 1518-4DPN/DP.
- CPU 1513 V1.5 is compatible with the CPUs 1513 V1.1 and V1.0, but not compatible with all other S7-1500 CPUs.
- CPU 1511 V1.0 is not compatible with all other S7-1500 CPUs.
- CPU 1518F V1.7 is compatible with CPU 1518F V1.6.

Note

Compatibility of F-CPUs

F-CPU versions are not compatible with non-F-CPUs. For example, CPU 1516F V1.6 is not compatible with CPU 1516 V1.5 and vice versa.

1.4.1.3 Principle of operation for S7-1500R/H CPUs (S7-1500)

Operating principle of the redundant system S7-1500R/H (S7-1500)

Special features of the redundant system S7-1500R/H

Redundant automation systems are used to reach higher availability. For high availability systems, the probability of production losses is reduced by the parallel operation of two systems. The redundant system S7-1500R/H has the following special features compared to the S7-1500 automation system:

- For the redundant system S7-1500R/H, the CPUs are duplicated, meaning they are redundant. If one of the two CPUs fails, the other CPU takes over the process control at the interruption point. The failure of the CPU has no effect on the controlled process. To ensure takeover of the process control, the process-controlling CPU (primary CPU) synchronizes itself via a redundant synchronization connection (redundancy connection) with the non-process-controlling CPU (backup CPU).
  You can find information on the configuration of the S7-1500R/H system in the section "Configuration of the redundant system S7-1500R/H (Page 1038)".
- The I/O is connected in an S7-1500R/H system to the CPUs via a PROFINET ring and is configured as a media redundancy ring.
  You can find additional information on the media redundancy ring in the section "Configuring media redundancy (Page 2943)".
- There are new operating and system states in the S7-1500R/H systems.
  You can find additional information in the section "Operating and system states (Page 1045)".
- There are special instructions and organization blocks (OBs) for the S7-1500R/H CPUs.
  You can find additional information in the section "Basics of program execution (Page 1051)".
- Project data can be downloaded to the primary CPU or to the backup CPU.
  You can find additional information in the section "Loading an S7-1500R/H CPU (Page 1055)".
- In addition to supporting device IP addresses, redundant systems support a system IP address.
  You can find additional information in the section "Communication with the R/H System (Page 1059)".
- New diagnostic options are available for the redundant system S7-1500R/H.
  You can find information in the section "Diagnostics of the redundant system S7-1500R/H (Page 1069)".

The complete documentation on the S7-1500R/H redundant system is available in the system manual of the S7-1500R/H redundant system and in the linked documents.

Restrictions and special features (hardware)

When using the redundant system S7-1500R/H there are restrictions for the hardware compared to the S7-1500 automation system. Below you can find examples for some restrictions:

- No support for central I/O and central communication modules in the hardware configuration of the redundant system S7-1500R/H
  Exceptions: Load current supply (PM), system power supply (PS)
- No support of fail-safe modules in the redundant system S7-1500R/H
- Longer cycle and reaction times

A complete overview of all restrictions for the redundant system S7-1500R/H compared to the S7-1500 automation system can be found in the section "Operational planning > Restrictions" in the "Redundant System S7-1500R/H" system manual.

Restrictions and special features (software)

When using the redundant system S7-1500R/H there are restrictions for the software compared to the S7-1500 automation system. Below you can find examples for some restrictions:

- No support or restrictions for specific instructions
- Temporary local data for functions (FCs): The system initialization for standard access is identical for S7-1500R/H-CPUs as for optimized access, for the description of the system initialization for optimized access see AUTOHOTSPOT.
- No support of IRT, MRPD
- No support of isochronous operation

A complete overview of all restrictions for the redundant system S7-1500R/H compared to the S7-1500 automation system can be found in the section "Operational planning > Restrictions" in the "Redundant System S7-1500R/H" system manual.

Communication restrictions

- Programming device communication:
  - Is not possible to access both CPUs at the same time. You can access either the primary CPU either backup CPU.
  - No loading of objects (e.g. blocks) possible in redundant mode (RUN-Redundant state); neither a "Download to device" nor an "Upload from device" is possible. Furthermore, working with snapshots for tags of a data block is not possible in this system state.
  - The function "Upload device as new station" is not supported.
- Open User Communication:
  - No configured connections
  - Secure Open User Communication:
    No support because no certificate management is possible for the S7-1500R/H CPUs:
    If you have enabled Secure OUC, you can compile and load the user program, but not add any certificates to the S7-1500R/H.
    Due to the missing certificate management, email connections are only non-secure (this means the instruction TMAIL_C can only be used up to and including V4.1)
  - No FDL connections
  - Connection establishment (instruction TCON) not possible with connection description DB with a structure to TCON_Param
- No OPC UA
- No S7 communication as client
- No web server
- No S7 routing between the PROFINET interface X1 and the PROFINET interface X2 of the CPUs

See also

System IP addresses (Page 1059)

Configuration of the redundant system S7-1500R/H (S7-1500)

Creating R/H CPUs (S7-1500)

You can configure the redundant system S7-1500R/H in different versions. A PROFINET ring is mandatory for all configuration versions.

For the configuration versions of the S7-1500R/H system the redundancy exists for the following components:

- R/H CPUs
- Synchronization interfaces
- Network in the PROFINET ring

This section describes a possible configuration version. You can find a description of additional configuration versions in the section "Configuration versions" in the "Redundant System S7-1500R/H" system manual.

Creating R/H CPUs

In the network view, you select an R/H-CPU from the hardware catalog and create it together with a rack. Drag the desired modules from the hardware catalog to this device; the modules are arranged automatically on the rack.

To select an R/H CPU from the hardware catalog, follow these steps:

1. In the hardware catalog navigate to the folder with the desired R/H CPUs.
2. Open the folder with the R/H CPU type you are looking for. You will see all article numbers for the selected R/H CPU type.
3. Click an R/H CPU article number to get information about the selected R/H CPU in the "Information" pane.
4. Set the matching firmware version of your R/H CPU in the "Version" drop-down list.
5. Set up the R/H CPU and a rack. You have the following options:
   - Drag-and-drop the R/H CPU from the hardware catalog into the network view.
   - Double-click the R/H CPU item in the hardware catalog.

Representation in the network view

STEP 7 automatically creates both R/H CPUs of the redundant system. STEP 7 displays both R/H CPUs in the network view graphically:

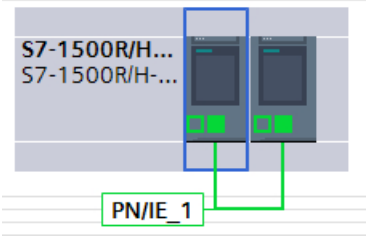

Representation in the project tree

In the STEP 7 project tree, each of the two R/H CPUs of the redundant system is displayed in its own tree. The project tree contains all elements and editors of the project.

S7-1500R/H-System_1 [S7-1500R/H-Station] Device configuration Online & diagnostics	For each redundant system in the project there is a separate folder which has an internal project name. Below this folder you find the device configuration and diagnostic options that apply to the overall system.
PLC_1 [CPU 1515R-2 PN] Device configuration Online & diagnostics Program blocks Technology objects External source files PLC tags PLC data types Watch and force tables Online backups Traces Device proxy data Program info PLC supervisions & alarms PLC alarm text lists Local modules PLC_1 [CPU 1515R-2 PN] Distributed I/O PROFINET IO-System (100): PN/IE_1 IO-Device_1 [IM 155-6 PN HF]	There is one folder each for both CPUs of the redundant system, each having an internal project name. The CPU displayed in the project tree above has the redundancy ID "1". Below the CPU its properties are displayed. In addition this contains further properties of the redundant system, the user program and additional system-relevant project items. In the "Distributed I/O" folder you find links to the distributed I/O devices assigned to the CPU.
PLC_2 [CPU 1515R-2 PN] Device configuration Online & diagnostics Local modules PLC_2 [CPU 1515R-2 PN]	The CPU displayed in the project tree below has the redundancy ID "2". Below the CPU its properties are displayed.
Ungrouped devices IO-Device_1 [IM 155-6 PN HF] Device configuration Online & diagnostics IO-Device_1 [IM 155-6 PN HF] DI 8x24VDC HF_1 DQ 8x24VDC/0.5A HF_1	All distributed I/O devices in the project are collected together in the "Ungrouped devices" folder.

You can find additional information in the section "Function and structure of the project tree" of the information system.

Redundant IDs and IP addresses

Each CPU of the redundant system has a redundancy ID. With the redundancy ID, a project tree is assigned to the real CPU in STEP 7. The top CPU in the tree (in this example: PLC_1) is always the CPU with the redundancy ID "1". The CPU below this (in this example: PLC_2) has the redundancy ID "2".

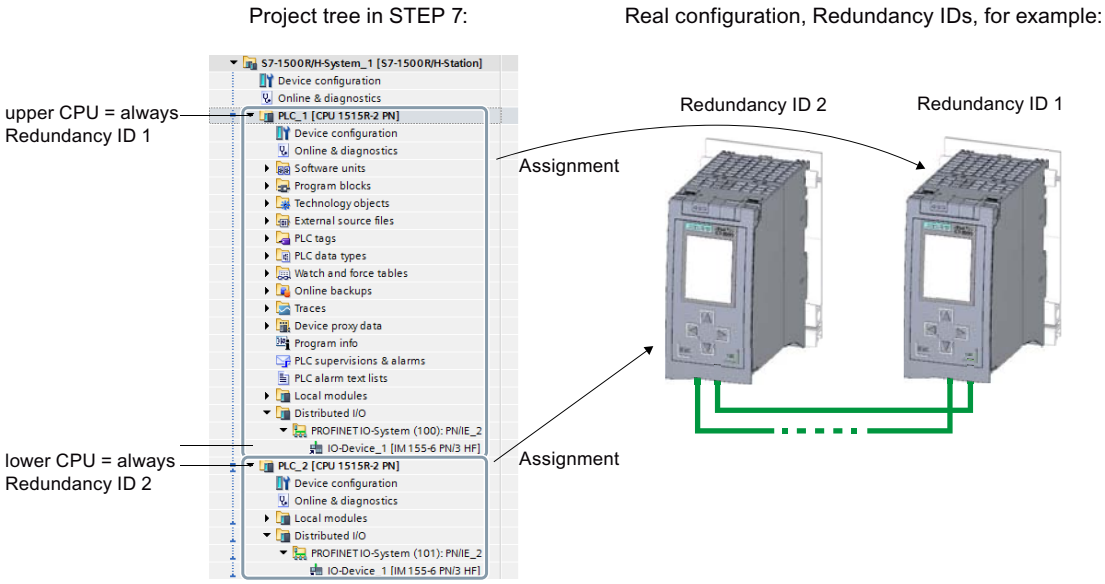

If a CPU has a valid hardware configuration and you change the redundancy ID of this CPU, you also change the name and IP addresses of this CPU. You can find additional information in the section "Redundancy ID" of the "Redundant System S7-1500R/H" system manual.

Configuring R/H CPUs (S7-1500)

Assigning an IP address

STEP 7 automatically assigns an IP address to each PROFINET interface of a CPU. You can also assign the IP addresses manually, for example in case of a conflict.

For the PROFINET interface X1 of the CPUs, the IP addresses must be located in the same subnet.

The IP address is displayed in the CPU properties, in the "PROFINET interface [X1]" area of the "IP protocol" section.

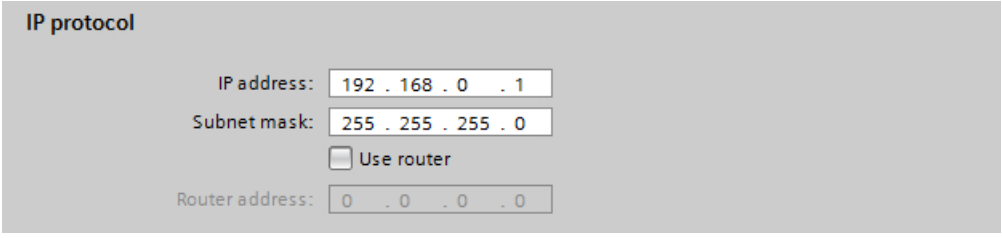

Assigning system IP address

In addition to the device IP addresses of the CPUs, you can also assign system IP addresses for the redundant system S7-1500R/H; for each PROFINET interface you can assign a system IP address for switched communication in addition to the "normal" (devices) IP addresses.

You use the system IP address for communication with other devices (for example, HMI devices, CPUs, PG/PC). The devices always communicate over the system IP address with the primary CPU of the redundant system. This ensures, for example, that the communication partner can communicate with the new primary CPU (previously backup CPU) in the RUN-Solo system state after failure of the original primary CPU in redundant operation.

Follow these steps to activate the system IP address for the PROFINET interfaces X1 of both CPUs:

1. Select a CPU in the network view. Select the "Properties" tab in the Inspector window.
2. Select the area "PROFINET interface [X1]" and the section "System IP address for switched communication" in the area navigation.
3. Make sure that the check box "Enable the system IP address for switched communication" is selected for the interface X1. Apply or assign the system IP address in the "IP address" field.
   The subnet mask cannot be modified and corresponds to the subnet mask of the IP protocol.
4. Then apply or assign a virtual MAC address to the system IP address.
   The virtual MAC address is six bytes long. The assignment of the bytes is hexadecimal.

   Note

   Virtual MAC address

   Ensure that all MAC addresses stored in the Ethernet broadcast domain are unique. This applies in particular to systems with third-party devices consisting of VRRP and redundant systems that are configured through several STEP 7 projects.

5. The other CPU applies the settings automatically.

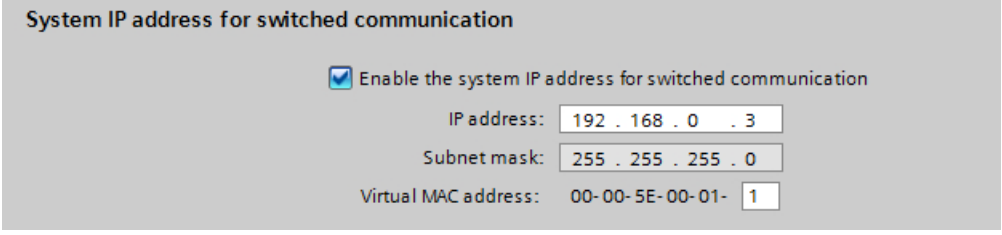

For additional information on the system IP address, refer to the "Communication function manual".

Setting the cycle monitoring time

STEP 7 assigns default values for the minimum and maximum scan cycle times.

The default values are displayed in the "Cycle" area of the CPU properties.

Note

Setting the cycle time as large as possible

Select the minimum scan cycle time and the maximum cycle time as large as your process allows.

- The time for the ongoing synchronization of the two CPUs in redundant operation is included in the cycle time.
- At a system state transition SYNCUP → RUN Redundant, a temporary increase of the cycle time can occur.

If only one CPU controls the process (RUN-Solo system status), the cycle time is significantly shorter than during redundant operation.

Creating IO devices (S7-1500)

Add IO devices with system redundancy S2 to the R/H CPUs. To do so, follow these steps:

1. In the hardware catalog navigate to the folder with the desired IO devices.
2. Open the folder with the required IO device type. You see all article numbers of the selected IO device type.
3. Click an IO device article number to get information about the selected IO device in the "Information" pane.
4. Set the matching firmware version of your IO device in the "Version" drop-down list.
5. Create the IO device and a rack. You have the following options:
   - Drag-and-drop the IO device from the hardware catalog into network view.
   - Double-click the IO device item in the hardware catalog.
6. Drag-and-drop the desired modules on the respective slots of the IO device.
7. If necessary, add additional IO devices.

Assigning IO devices to the redundant system (S7-1500)

To assign IO devices to the redundant system S7-1500R/H, connect every IO device to each CPU.

To do so, follow these steps:

1. Drag-and-drop a line between the PROFINET interface of the IO device and the PROFINET interface X1 of the left CPU.
2. Drag-and-drop a line between the PROFINET interface of the IO device and the PROFINET interface X1 of the right CPU.
3. Assign further IO devices to both CPUs in the same way.
   The IO devices are connected system-redundantly to the S7-1500R/H system.

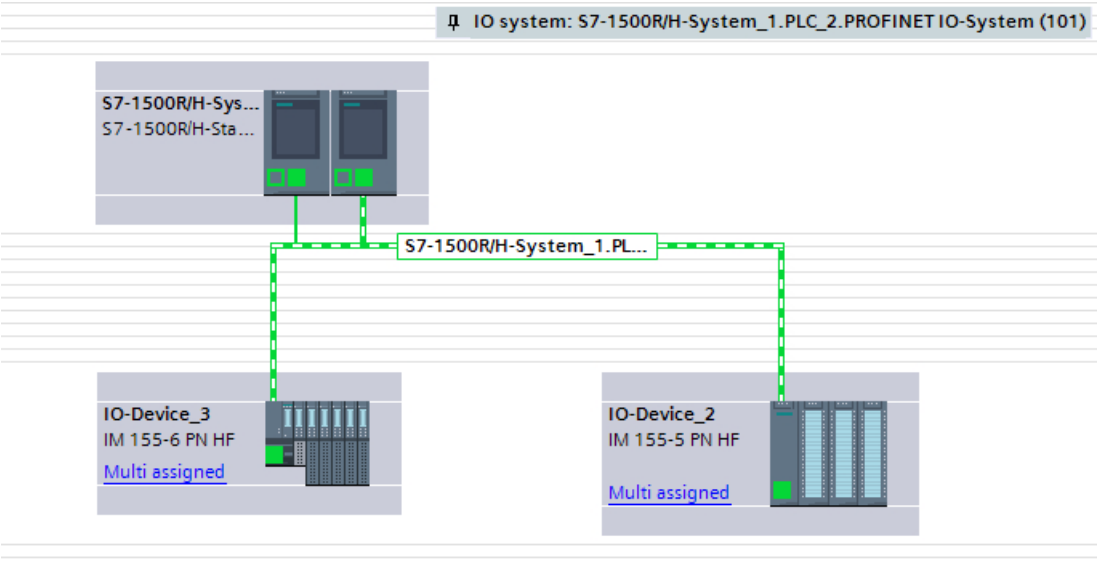

Creating an MRP ring configuration (S7-1500)

Setting the MRP role of the R/H CPU

As soon as you create a redundant system S7-1500R/H in STEP 7, STEP 7 automatically assigns the MRP role "Manager (auto)" for the PROFINET interfaces X1 of both CPUs.

Defining the MRP role for additional stations of the ring

Follow these steps to determine the media redundancy for additional stations in the ring:

1. In the network view of STEP 7, select the PROFINET interface X1 of one of the two CPUs of the redundant system S7-1500R/H.
2. In the Inspector window navigate to "Properties" > "General" > "Advanced options" > "Media redundancy".

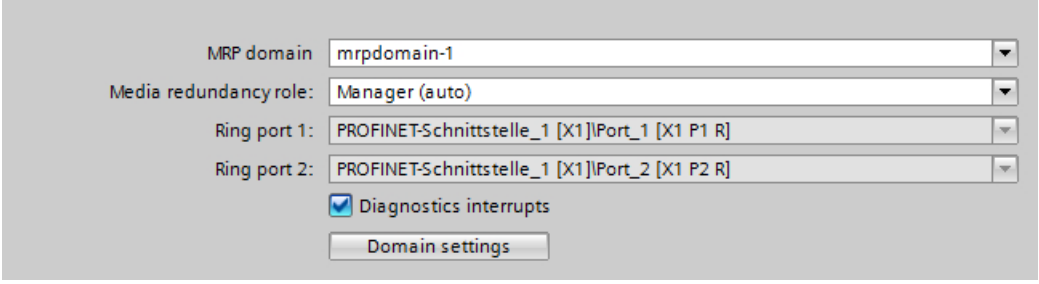

3. Click the "Domain settings" button.
   In the Inspector window STEP 7 displays the properties of the MRP domain in which the PROFINET interface X1 of the CPU is located.
4. In the "MRP role" column of the "Devices" table assign the MRP role "Client" to all additional stations.

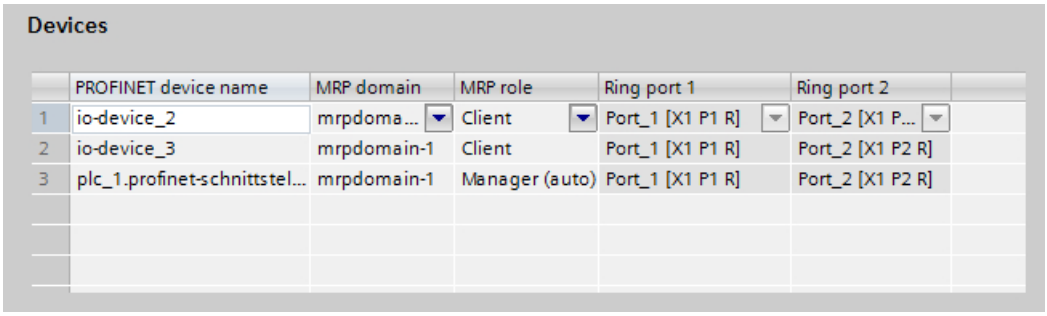

Devices

	PROFINET device name	MRP domain	MRP role	Ring port 1	Ring port 2
1	io-device_2	mrpdoma...	Client	Port_1 [X1 P1 R]	Port_2 [X1 P...
2	io-device_3	mrpdomain-1	Client	Port_1 [X1 P1 R]	Port_2 [X1 P2 R]
3	plc_1.profinet-schnittstel...	mrpdomain-1	Manager (auto)	Port_1 [X1 P1 R]	Port_2 [X1 P2 R]

Parameterizing stations outside the STEP 7 project

Set the MRP role "Client" for the stations of the ring that are not located in STEP 7.
For a switch, for example, set the MRP role "Client" via the web interface of the switch.

Operating modes and system states (S7-1500)

Overview (S7-1500)

The S7-1500R/H CPUs have the same operating modes as S7-1500 standard CPUs and additional operating modes.

System states map a state that results from the operating modes of the two CPUs in the redundant system.

You can find additional information in the section "Operating modes and system states" in the "Redundant system S7-1500R/H" system manual.

Operating modes

Operating modes describe the behavior of a single CPU at a specific time. The knowledge about the operating modes of the CPUs is useful for programming the startup, the test and the error diagnostics. The status LEDs on the front of the CPU and the CPU display indicate the current operating mode.

Like the S7-1500 standard CPUs, the S7-1500R/H CPUs have the operating modes STOP, STARTUP and RUN. For operation as redundant system, one of the two CPUs can have on an additional operating mode, SYNCUP, for synchronizing the two subsystems. The operating mode RUN is divided into RUN, RUN-Syncup and RUN-Redundant for redundant systems.

System states

The system states enable the direct assessment of the behavior of a redundant system. They result from the combination of the operating modes of the individual CPUs.

Overview of operating modes and system states

The following figure shows the possible operating modes of the CPUs and the resulting system states.

In general, both CPUs have equal status so that each CPU can be either primary or backup CPU. The CPU that you first switched fromSTOP to RUN becomes the primary CPU.

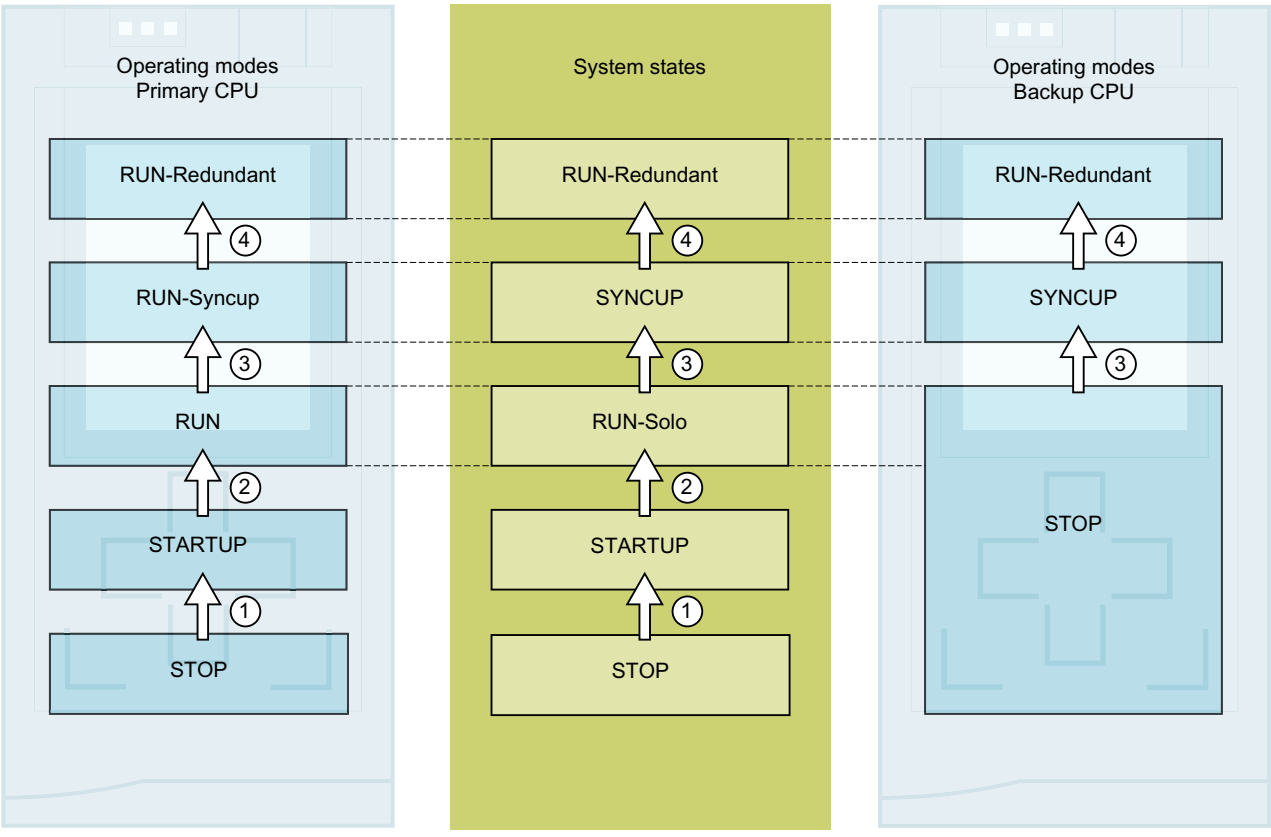

The following table provides you with an overview of how the redundant system powers up and at the same time runs through the various operating modes and system states. The following starting situation and the action steps are an example.

The operating modes and system states are described in detail in the following sections.

Table 1-84 Startup of the redundant system

No. in the screen	Primary CPU	System state	Backup CPU
Initial situation: Both CPUs are in STOP mode. The mode selector is also in STOP position.			
1st step: Switch the mode selector on the CPU which is to be the **primary CPU** from STOP to RUN.			
①	The CPU goes into STARTUP and processes the startup OB 100 and any additional startup OBs.	STOP → STARTUP	The CPU remains in STOP.

No. in the screen	Primary CPU	System state	Backup CPU
②	After successful STARTUP the CPU switches to RUN. The CPU runs in RUN, like a standard CPU and processes your user program.	STARTUP → RUN-Solo	The CPU remains in STOP.
2nd step: Switch the mode selector on the **Backup CPU** from STOP to RUN.			
③	RUN → RUN-Syncup	RUN-Solo → SYNCUP	STOP → SYNCUP
	Both user programs are synchronized for redundant operation. The primary CPU copies the load memory and work memory content to the backup CPU. The backup CPU syncs up with the user program processing of the primary CPU. After successful synchronization, both CPUs have identical memory content.		
④	RUN-Syncup → RUN-Redundant	SYNCUP → RUN-Redundant	SYNCUP → RUN-Redundant
	After the SYNCUP CPUs go to RUN-Redundant mode. Both CPUs process the user program synchronously.		

STARTUP mode (S7-1500)

Startup processing (the primary CPU only)

The STARTUP is only executed by the primary CPU.

In STARTUP mode, the CPU behaves like an S7-1500 standard CPU.

Behavior

Before the CPU starts to execute the cyclic user program, a startup program is executed.

In STARTUP the CPU compares the existing I/O configuration with the hardware configuration that you created with STEP 7.

By suitably programming startup OBs, you can specify initialization tags for your cyclic program in the startup program. You have the option of programming no Startup OB, one Startup OB, or several Startup OBs.

Special features

- All outputs are disabled or react according to the parameter settings for the respective module: They provide a substitute value as set in the parameters or retain the last value output and bring the controlled process to a safe operating state.
- The process image is initialized.
- The process image is not updated.
  In order to read the current state of inputs during STARTUP, you can access inputs by direct I/O access.
  In order to initialize outputs during STARTUP, you can write the values via the process image or via direct I/O access. The values are output at the outputs during the transition to the RUN mode.

- The CPU always starts up in a warm restart.
  - If you define data as retentive, its content is retained for the startup of a program after STOP or a voltage failure.
  - The non-retentive bit memory, timers and counters are initialized.
  - The non-retentive tags in data blocks are initialized.
- During STARTUP no cycle time monitoring is running yet.
- The CPU processes the startup OBs in the order of the startup OB numbers. The CPU processes all programmed startup OBs irrespective of the selected startup mode (see figure below "Setting the startup behavior").
- If a corresponding event occurs, the CPU can start the following OBs in startup:
  - OB 82: Diagnostics interrupt
  - OB 83: Removal/insertion of modules
  - OB 86: Rack error
  - OB 121: Programming error (only for global error handling)
  - OB 122: I/O access error (only for global error handling)

  The CPU starts all other OBs after the transition to the RUN mode.

See also

Overview (Page 1045)

STOP mode (S7-1500)

Behavior

CPUs of the redundant system behave in STOP exactly as S7-1500 standard CPUs.

The CPU does not execute the user program in STOP mode.

If both CPUs are in STOP mode, all outputs are disabled or react according to the parameter settings for the respective module: They provide a substitute value as set in the parameters or retain the last value output and thus hold the controlled process in a safe operating state.

Special features

In STOP mode, the backup CPU does not establish any connections to the IO devices.

In STOP mode, the primary CPU establishes connections to the IO devices. The primary CPU also enables the system IP address in STOP mode, provided the system IP address has been configured.

If both CPUs are in STOP and you load a configuration in the CPU, make sure to note the following:

- You must start the CPU in which you have loaded the configuration first, in order that it is/ remains the primary CPU.
- When you load to the primary CPU, connected IO devices are also parameterized according to the loaded hardware configuration in STOP mode.

See also

Overview (Page 1045)

SYNCUP operating mode (S7-1500)

Behavior

In the SYNCUP operating mode, the operating system synchronizes the backup CPU with the primary CPU. The primary CPU is in the RUN-Syncup operating mode and controls the process.

In contrast to the primary CPU, the backup CPU does not go through the STARTUP operating mode.

You can find additional information in the section SYNCUP system status (Page 1051).

See also

Overview (Page 1045)

RUN operating modes (S7-1500)

RUN operating modes

The primary CPU goes through multiple operating modes on the way to the system status RUN-Redundant:

- RUN
- RUN-Syncup
- RUN-Redundant

The backup CPU only has the RUN-Redundant operating mode.

RUN operating mode

In the RUN operating mode, the primary CPU behaves like an S7-1500 standard CPU. It performs the cyclic, time-driven and interrupt-driven program execution on its own.

Addresses in the "Automatic update" process image are automatically updated in each program cycle. You can find additional information in the section "Process image and process image partitions" of the "Redundant system S7-1500R/H" system manual.

Once the CPU has written the outputs and has read the inputs, the cyclic program is executed from the first instruction to the last instruction. Events with higher priority, such as hardware interrupts, diagnostic interrupts and communication, can interrupt the cyclic program flow and prolong the cycle time.

If you have configured a minimum cycle time, the CPU does not terminate the cycle until after this minimum cycle time has expired, even if the user program is completed sooner.

The operating system monitors the runtime time of the cyclic program for a configurable upper limit known as the maximum cycle time. You can restart this time monitoring at any point in your program by calling the RE_TRIGR instruction.

If the cyclic program exceeds the cycle monitoring time, the timeout OB (OB 80) may be started. You can find additional information in the section "Start events" of the "Redundant system S7-1500R/H" system manual.

Special features in the RUN operating mode

During non-redundant operation the CPUs are independent of each other. They can have different projects.

RUN-Syncup operating mode

In the RUN-Syncup operating mode the backup CPU synchronizes with the primary CPU. The SYNCUP that temporarily has an effect on the primary CPU (for example delay of asynchronous services, cycle time extension through compilation of the load memory contents) runs simultaneously in the backup CPU. You can find additional information in the section "System status SYNCUP" of the "Redundant system S7-1500R/H" system manual.

The operating system synchronizes the work memory contents of the primary CPU consistently at a cycle control point. During synchronization the program cycle in the primary CPU is extended depending on the amount of data to be synchronized and in addition to the other cycle time of the application.

RUN-Redundant operating mode

Both CPUs operate the user program synchronously in RUN-Redundant operating mode.

In RUN-Redundant operating mode it is not possible to load objects (e.g. blocks); neither a "Download to device" nor an "Upload from device" is possible. Furthermore, working with snapshots for tags of a data block is not possible in this system state.

See also

Overview (Page 1045)

SYNCUP system status (S7-1500)

SYNCUP system state

In the SYNCUP system status the operating system synchronizes the user programs in both CPUs for redundant operation. In the SYNCUP system state, the operating system synchronizes the backup CPU with the primary CPU. The backup CPU is in SYNCUP mode; the primary CPU is in RUN-Syncup mode and controls the process. With the SYNCUP system status the redundant system changes from RUN-Solo to the RUN-Redundant system status. Afterwards, both CPUs synchronously process the same user program. In the case of a failure of the primary CPU in redundant operation, the backup CPU takes over control of the process as the new primary CPU at the interruption point.

Properties

- An existing online connection between the primary CPU and the PG/PC at the beginning of the system state SYNCUP is terminated during the SYNCUP. The SYNCUP is not automatically re-established after it is terminated.
- If you call the instructions RDREC and WRREC during the SYNCUP system state, the error codes 0x7001 and 0x7002 are always returned. In the case of error as well, no other error code is returned even if, for example, the data record source or the data record target is not available. In this case, the correct error code is only returned after SYNCUP has been terminated.
  If you call the instructions RDREC and WRREC during the SYNCUP system state, no data record job is initiated. The data record job is not initiated until you call the instruction again after the termination of the SYNCUP. It is therefore recommended to always call RDREC and WRREC cyclically.

You can find additional properties and effects of the SYNCUP in the "SYNCUP system state" section of the "SIMATIC S7-1500R/H" system manual.

See also

Overview (Page 1045)

Basics of program execution (S7-1500)

Programming of S7-1500R/H CPUs (S7-1500)

User program for the redundant system S7-1500R/H

During the design and programming of the user program the same rules apply for the redundant S7-1500R/H system as for the S7-1500 automation system.

The user program is stored identically in both CPUs during redundant operation. Both CPUs process the user program event-synchronously.

From the point of view of the user program execution, the redundant S7-1500R/H system behaves like the S7-1500 automation system. The synchronization is integrated into the operating system and runs automatically and hidden between the primary and backup CPU.

Specific instructions and blocks for the redundant system S7-1500R/H

Specific instructions and OBs are available for the redundant system S7-1500R/H.

Use the instruction "RH_CTRL" to influence the processes for R/H systems. With the firmware version V2.6, you disable the SYNCUP system state or enable SYNCUP system state again. The goal is to only permit the SYNCUP system status in less critical process phases (see the section "Enabling/disabling the SYNCUP system status with the instruction RH_CTRL" in the "Redundant system S7-1500R/H" system manual).

You use the instruction "RH_GetPrimaryID" in the user program to read out which CPU is currently the primary CPU. You can find additional information in the section "Using the instruction 'RH_GetPrimaryID' to detect the primary CPU" in the "Redundant system S7-1500R/H" system manual.

In addition to the OBs of the S7-1500 CPU, you can use OB 72. With OB 72 you handle, for example, redundancy errors.

Special features during program execution

- You create the user program for the redundant system S7-1500R/H in the top CPU (e.g. PLC_1) of the STEP 7 project tree.
- The redundant system S7-1500R/H does not support some of the instructions the S7-1500 CPUs. Instructions that are not supported by the S7-1500R/H redundant system are grayed out in STEP 7 in the "Instructions" task card.
  STEP 7 shows the instructions that are not supported in the program code in red. If you compile with instructions that are not supported, STEP 7 issues a corresponding error message.
  The instructions that are not supported are specified in the section "Basics of program execution > Restrictions" in the "Redundant System S7-1500R/H" system manual.
- In the case of instructions with the "LADDR" block parameter, you use this parameter to determine which of the two CPUs is the target of this instruction.
  Example: To read out the I & M data of the CPU with the redundancy ID 1, enter the hardware identifier 65149 (or system constant "Local1") at the "LADDR" block parameter of the Get_IM_Data instruction.
- Behavior of asynchronous instructions
  - Normally, asynchronous instructions delay the SYNCUP process until their execution is completed.
  - If an asynchronous instruction is started while the SYNCUP process is running, it is put into a waiting state. It remains in this waiting state until the SYNCUP process is completed. When the SYNCUP process is completed, the asynchronous instruction goes into the active state.

- Response of S7-1500R/H redundant system when the maximum cycle time is exceeded, if you have not created any time error interrupt OB
  The first time the maximum cycle time is exceeded in the Run Redundant and SYNCUP system states leads to a loss of redundancy: The primary CPU goes to Run mode and the backup CPU goes to Stop mode.
  The second high-limit violation within the same program cycle causes the primary CPU to go to Stop mode.
  You can find additional information in the "Basics of program execution" section of the S7-1500 R/H system manual.
- Response of S7-1500R/H redundant system when the maximum cycle time is exceeded, if you have created a time error interrupt OB
  When the maximum cycle time is first exceeded in the Run Redundant and SYNCUP system states, the R/H system remains in its system state; the OB80 is called and the cycle time is reset.
  The second time the maximum cycle time is exceeded within the same program cycle leads to redundancy loss: The primary CPU goes to Run mode and the backup CPU goes to Stop mode. OB 80 is called and the cycle time is reset.
  The third time the maximum cycle time is exceeded within the same program cycle causes the primary CPU to go to stop mode.
  You can find additional information in the "Basics of program execution" section of the S7-1500 R/H system manual.

See also

Organization blocks (Page 1053)

Organization blocks (S7-1500)

CPU redundancy error OB (S7-1500)

Description

The operating system of each CPU of an R/H system calls up the CPU redundancy error OB (OB72) when one of the following events occurs:

- Loss of redundancy due to a CPU failure
- Loss of redundancy due to a CPU transition to STOP mode (triggered by the user or the system)
- R/H system has gone into the RUN-Redundant system status

Structure of the start information

A CPU redundancy error OB has the following start information:

- As for S7-400 CPUs:

Tag	Data type	Description
OB72_EV_CLASS	BYTE	Event class and identifiers: • B#16#73: Incoming event
OB72_FLT_ID	BYTE	Error code Possible values: • B#16#01: Loss of redundancy due to a CPU failure • B#16#02: Loss of redundancy due to a CPU transition to STOP mode • B#16#03: R/H system has gone into the RUN-Redundant system status
OB72_PRIORITY	BYTE	Priority class
OB72_OB_NUMBER	BYTE	OB number (72)
OB72_RESERVED_1	WORD	• High byte: B#16#00 • Low byte: Reserved
OB72_INFO_1	WORD	W#16#0000
OB72_INFO_2	WORD	W#16#0000
OB72_INFO_3	WORD	W#16#0000
OB72_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

- Optimized start information:

Name	Data type	Meaning
LADDR	HW_ANY	HW identifier of the defective hardware object
Event_Class	BYTE	Identical to the tag OB72_EV_CLASS in the table above
Fault_ID	BYTE	Identical to the tag OB72_FLT_ID in the table above

Start events of OB72

OB72_EV_CLASS or Event_Class	OB72_FLT_ID or Fault_ID	Start event
B#16#73	B#16#01	Loss of redundancy due to a CPU failure
B#16#73	B#16#02	Loss of redundancy due to a CPU transition to STOP mode
B#16#73	B#16#03	R/H system has gone into the RUN-Redundant system status

Loading an S7-1500R/H CPU (S7-1500)

Downloading project data to the CPUs (S7-1500)

Project data can be downloaded via an online connection from the PG/PC or HMI device into the CPUs of the S7-1500R/H system. The project data can either be downloaded into the primary CPU or the backup CPU, simultaneous access to both CPUs is not possible.

By default, the project data is downloaded to the primary CPU. In the SYNCUP system state, the project data is then transferred from the primary CPU to the backup CPU.

Project data can also be downloaded to the backup CPU. This makes sense if the backup CPU is to be the primary CPU at a restart.

The complete project data (all configuration data and the complete user program) can only be downloaded to a CPU in STOP operating state.

In the RUN-Solo system status the user program can also be loaded into the primary CPU that is in the RUN state. During downloading, the primary CPU maintains control of the process. There is no standstill of the plant.

See also

Downloading project data to the primary CPU (Page 1055)

Starting the primary CPU after a download (Page 1056)

Downloading project data to the backup CPU (Page 1057)

Loading a user program to the primary CPU (Page 1058)

Downloading project data to the primary CPU (S7-1500)

By default, the project data is downloaded to the primary CPU. To load the complete project data (all configuration data and the complete user program), the CPU must be in the operating state STOP. In the SYNCUP system state, the project data is then transferred from the primary CPU to the backup CPU.

Requirements

- The IP addresses have been assigned.
- The primary CPU is in the STOP operating state.

Procedure

To download the project data to the primary CPU, follow these steps:

1. Select the S7-1500R/H system in the project navigation.
2. Right-click to call up the shortcut menu.
3. Select the "Download to device > Hardware and software (only changes)" command from the shortcut menu.
   The "Extended download to device" dialog opens.

4. Select the subnet from the "Type of the PG/PC interface" drop-down list.
5. Select the adapter used from the "PG/PC interface" drop-down list:
6. Select the interface to which the PG/PC is connected from the "Connection to interface/ subnet" drop-down list.
   Alternatively, select the entry "Try all interfaces".
7. Click the "Start search" button.
   The "Choose target device" table shows the CPUs in the S7-1500R/H system and their roles. As a default the primary CPU is selected.
8. Click "Load".

Result

If necessary, the project data is compiled before the download.

The "Results of loading" dialog displays the results of the loading process.

Starting the primary CPU after a download (S7-1500)

After downloading the project data start the primary CPU.

Requirements

- It has been compiled without errors, consistent project data loaded.
- The primary CPU mode selector is in the RUN position.

Procedure

⚠ WARNING
Primary CPU startup with user program with errors Before starting the primary CPU ensure that a potentially faulty user program cannot damage property or people and cannot cause dangerous states.

To start the primary CPU after loading, follow these steps:

1. Select the "Start module" entry in the "Action" column in the "Results of the load process" dialog.
2. Click "Finish".
   The primary CPU switches to the RUN operating state.
3. Switch the backup CPU to the RUN operating state.

Note

Role change during loading

Beware of a possible role change between primary and backup CPU shortly before, during or after loading.

A role change can occur during loading if the primary CPU fails (power failure, hardware defect) or is in the STOP operating state and one of the following conditions occurs:

- You switch the backup CPU to RUN operating state during this time.
- You switch the backup CPU to POWER ON during this time and the startup characteristics were configured in STEP 7 so that the CPU automatically goes into the RUN operating state after the POWER ON.

In the event of a role change, the new primary CPU starts up with the old project data. The new project data in the current backup CPU is overwritten with the old project data during synchronization of the two CPUs for redundant operation.

Result

After successful SYNCUP, the S7-1500R/H system switches to redundant mode.

Downloading project data to the backup CPU (S7-1500)

By default, the project data is downloaded to the primary CPU. You can also download the project data to the backup CPU. This makes sense if the backup CPU is to be primary CPU with its project data upon a restart.

While the project data is downloaded the primary CPU continues to control the process.

After downloading you must switch the primary CPU to the STOP operating state and the backup CPU to the RUN operating state. This means the previous backup CPU becomes the new primary CPU.

Note

Loading to the backup CPU

If the project uses retentive data, the backup CPU runs with data which may be outdated.

Requirement

The backup CPU is in STOP operating state.

Downloading project data to the backup CPU

To download project data to the backup CPU, follow these steps:

1. Right-click to select the S7-1500R/H system in the project navigation.
2. Select "Hardware and software (only changes)" under "Download to backup CPU".
   The backup CPU is now selected instead of the primary CPU in the "Extended download" dialog.
3. Select the subnet from the "Type of the PG/PC interface" drop-down list.
4. Select the adapter used from the "PG/PC interface" drop-down list:
5. Select the interface to which the PG/PC is connected from the "Connection to interface/ subnet" drop-down list.
6. Alternatively, select the entry "Try all interfaces".
7. Click the "Start search" button.
   The "Choose target device" table shows the CPUs in the S7-1500R/H system and their roles. By default the backup CPU is selected.
8. Click "Load".
   The "Results of loading" dialog displays the results of the loading process.

Switching primary and backup CPU

After downloading the project data into the backup CPUs switch the CPUs so that the backup CPU becomes the new primary CPU. To do this, follow these steps:

1. Switch the primary CPU to the STOP operating state.
2. Switch the backup CPU to the RUN operating state.
   The previous backup CPU becomes the new primary CPU and controls the process on its own in the RUN-Solo system status.

Result

The S7-1500R/H system switches back to the RUN-Redundant system status after SYNCUP with the modified user program.

Loading a user program to the primary CPU (S7-1500)

The user program can be downloaded into a CPU in the operating state RUN. During downloading, the primary CPU maintains control of the process. There is no standstill of the plant.

Requirement

The primary CPU is in the RUN operating state.

Procedure

Proceed as follows to download a changed user program into the primary CPU:

1. Switch the backup CPU to the STOP operating state.
   The S7-1500R/H system switches from the RUN-Redundant system state to the RUN-Solo system state.
2. Download the modified user program to the primary CPU with "Download to device" > "Software (only changes)".
   The primary CPU continues to control the process.
3. Switch the backup CPU to the RUN operating state.

Result

The primary CPU remains in the RUN operating state and synchronizes the modified user program with the backup CPU in SYNCUP.

The S7-1500R/H system switches back to the RUN-Redundant system status with the modified user program.

Communication with the R/H system (S7-1500)

System IP addresses (S7-1500)

For information on the limitations of the R/H system with regard to communication functions, see Operating principle of the redundant system S7-1500R/H (Page 1036).

The following explanations describe the possibilities of maintaining the communication to the R/H system, even if, for example, the CPU that is controlling the process fails.

System IP addresses of the redundant system S7-1500R/H

In addition to the "normal" IP addresses of the PROFINET interfaces. the redundant system S7-1500R/H also supports **system IP addresses**.

System IP addresses are used to communicate with other devices (for example, HMI devices, CPUs, PG/PC). These devices always communicate over the system IP address with the primary CPU of the redundant system. This ensures, for example, that the communication partner can communicate with the new primary CPU (previously backup CPU) in the RUN-Solo system state after failure of the original primary CPU in redundant operation.

To differentiate between "normal" and "system" IP address, the normal IP address is referred to as "devices IP address" – only in connection with R/H systems.

The following system IP addresses exist:

- System IP address for the PROFINET interfaces X1 of both CPUs (system IP address X1)
- System IP address for the PROFINET interfaces X2 of both CPUs (system IP address X2)

A virtual MAC address belongs to each system IP address.

Advantages of the system IP addresses compared to the device IP addresses

- Communication partners communicate specifically with the primary CPU.
- The communication of the redundant system S7-1500R/H via system IP address continues to work even after the failure of the primary CPU (primary-backup switchover).

Applications

Use the system IP addresses for the following locations:

- HMI communication with the redundant system S7-1500R/H: With a HMI you control and monitor the process on the redundant system S7-1500R/H.
- Open User Communication with the redundant system S7-1500R/H:
  - Another CPU or an application on the PC is accessing the data of the redundant system S7-1500R/H.
  - The redundant system S7-1500R/H accesses another device.

  TCP, UDP and ISO-on-TCP connections are possible.

Requirements

- The communication partner in the PROFINET interfaces of both CPUs are located in the same subnet.
- Communication partner is connected to both CPUs, in each case be the same interface (for example, X2).
- The system IP address is enabled.

Communication via the system IP address X2

If the CPUs of the redundant system S7-1500R/H have two PROFINET interfaces, it is recommended to use PROFINET interface X2 four communication with other devices.

Following figure shows a configuration in which the communication partners are connected via the respective PROFINET interfaces X2 the CPUs of the redundant system S7-1500R/H.

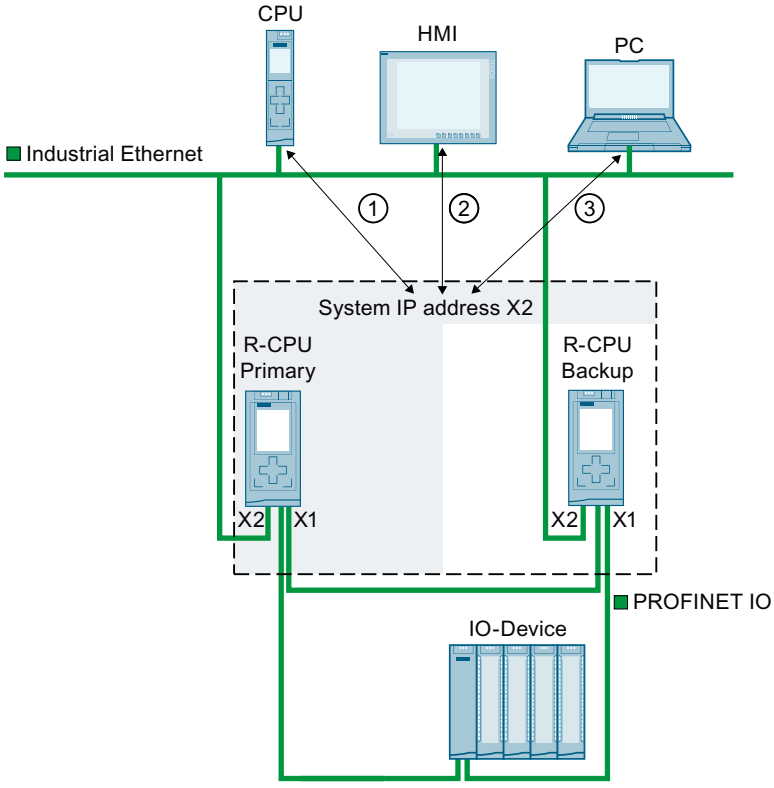

① Open User Communication between another CPU and the redundant system S7-1500R/H

② HMI communication with the redundant system S7-1500R/H

③ Open User Communication between the redundant system S7-1500R/H and a PC

Communication via the system IP address X1

The following figure shows a configuration in which the communication partners are connected with a switch to the PROFINET ring of the redundant system S7-1500R/H. The PROFINET ring connects the connection partners with the respective PROFINET interfaces X1 of both CPUs.
Since CPU 1513R only has one PROFINET interface, the connection via the PROFINET ring the only option to connect via the system IP address X1.

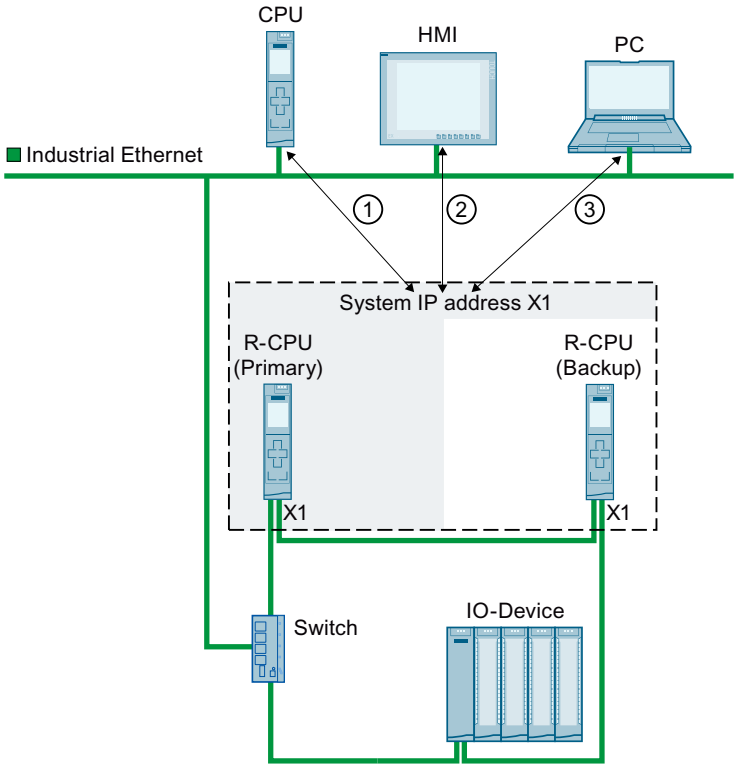

① Open User Communication between the redundant system S7-1500R/H and another CPU.

② HMI communication with the redundant system S7-1500R/H

③ Open User Communication between the redundant system S7-1500R/H and a PC

Communication via the system IP addresses X1 and X2

If the CPUs of the redundant system S7-1500R/H have two PROFINET interfaces (X1 and X2), you can use respective system IP addresses from both PROFINET interfaces. PROFINET devices which are connected with the interfaces X1 of the CPUs, communicate via the system IP address X1. PROFINET devices which are connected with the interfaces X2 of the CPUs, communicate via the system IP address X2.

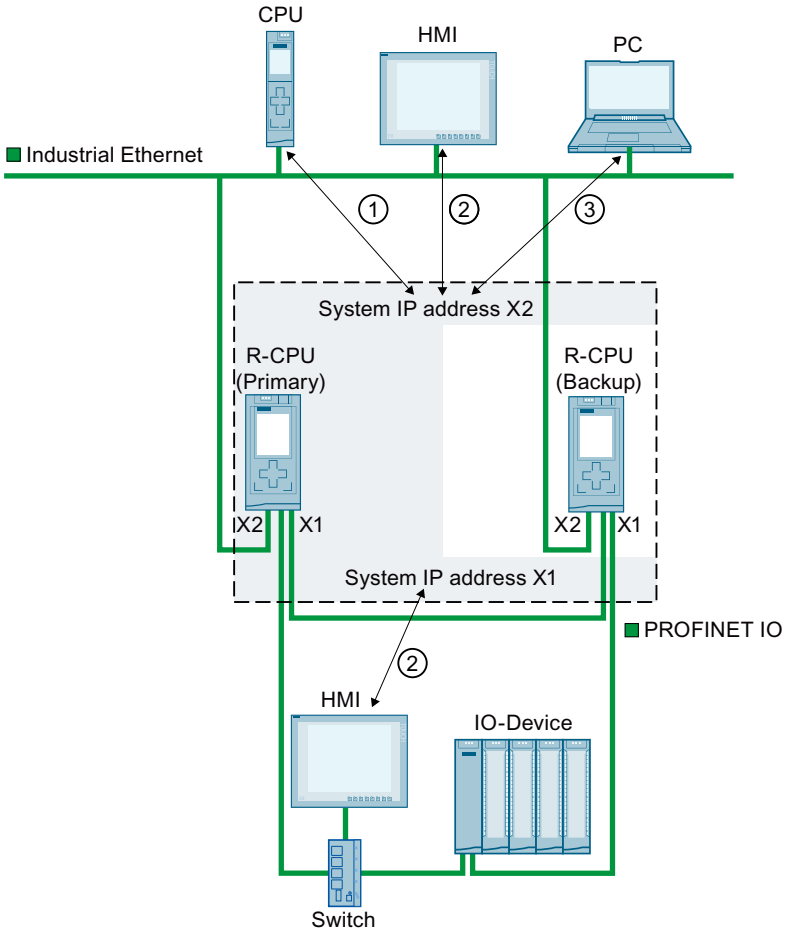

① Open User Communication between the redundant system S7-1500R/H and another CPU.

② HMI communication with the redundant system S7-1500R/H

③ Open User Communication between the redundant system S7-1500R/H and a PC

Enabling system IP addresses

If the CPUs of the redundant system S7-1500R/H have two PROFINET interfaces (X1 and X2), you can use a system IP address from both PROFINET interfaces. The section below describes how to enable the system IP address for the interface X1.

To enable the system IP address for your redundant system S7-1500R/H, proceed as follows:

1. In the network view of STEP 7, select the interface X1 of one of the two CPUs.
2. In the Inspector window, go to "Properties" > "General" > "Ethernet addresses" in the area "System IP address for switched communication".

3. Select the check box "Enable the system IP address for switched communication". STEP 7 automatically creates a system IP address.
4. If necessary, adjust the system IP address.
5. If required, adjust the virtual MAC address. To do this, assign a project-wide unique value (value range 01_H bis FF_H) for the last byte at "Virtual MAC address".

Note

Uniqueness of the virtual MAC address

The redundant system S7-1500R/H uses the Virtual Router Redundancy Protocol (VRRP) for the system IP address and associated virtual MAC address.
If you use other devices VRRP, for example switches, make sure that the MAC addresses are unique within an Ethernet broadcast domain.

See also

Behavior of connections at primary-backup switchover (Page 1069)

Setting up HMI connection via the system IP address (S7-1500)

Requirements

- Redundant system S7-1500R/H, for example, CPU 1513R-1PN is positioned in the network view
- System IP address is enabled
- HMI device with PROFINET interface is located in the network view

Procedure

To set up a HMI connection to a redundant system S7-1500R/H, follow these steps:

1. In the network view of STEP 7, select a PROFINET interface of the HMI device.
2. Using drag-and-drop operation, drag a line between the PROFINET interface of the HMI device and a PROFINET interface of the redundant system S7-1500R/H.
   The HMI device and the redundant system S7-1500R/H are interconnected.
3. On the toolbar, click the "Connections" icon. This activates connection mode.
4. Using drag-and-drop operation, drag a line between the HMI device and a CPU of the redundant system S7-1500R/H.
   The "Connection partners" list opens.
5. Select the redundant system S7-1500R/H from the "Connection Partners" list.

Result: You have set up a HMI connection between the HMI device and the redundant system S7-1500R/H. The HMI connection uses the system IP address.

Converting HMI connection to the devices IP address

To convert the HMI connection permanently to the selected CPU, clear the check box "Use system IP address for switched communication" in the properties of the HMI connection. The HMI connection then uses the devices IP address of the PROFINET interface. If this CPU fails, then the HMI connection to this CPU fails permanently.

Note

Automatic setup of HMI connection

As soon as you drag-and-drop a tag from the redundant system S7-1500R/H to a HMI screen or to a HMI tag table, STEP 7 automatically sets up an HMI connection. By default, this HMI connection exists between the PROFINET interface the HMI device and the PROFINET interface X1 of the CPU with redundancy ID 1. Connection users the devices IP address of the PROFINET interface X1.

The properties of the HMI connection you can convert HMI connection to a system IP address.

Setting up Open User Communication via the system IP address (S7-1500)

The redundant system S7-1500R/H can communicate with other devices via Open User Communication.

- Setup the connections the user program, for example, with the instruction "TSEND_C".

  Note

  Configured connections do not support the redundant system S7-1500R/H.

- You can set up the connections either via the devices IP addresses or via the system IP addresses of the PROFINET interfaces.
  - When you set up connection via a system IP address, communication is always routed through the primary CPU.
    Recommendation: Always use the system IP address for Open User Communication.
  - In redundant mode, the redundant system can establish or terminate IP address connections and send or receive data using any device.
    If you set up the connection via a device IP address, communication runs via the corresponding CPU. If the CPU fails, then the entire communication via the device IP addresses of this CPU fails.

Usable protocols / connection types

In addition to the TCP protocol explained below as example (set up by means of the connection parameterization for the instructions TSEND_C and TRCV_C), you can also use the ISO-ON-TCP, UDP and Modbus TCP protocols with the corresponding system data types and instructions, see table.

The system data type TCON_Param is not supported.

Protocol	System data type	instructions
TCP	• TCON_IP_v4 • TCON_QDN • TCON_IP_v4_SEC* • TCON_QDN_SEC*	Establish connection and send/receive data via: • TSEND_C/TRCV_C or • TCON, TSEND/TRCV or • TCON, TUSEND/TURCV (connection can be terminated via TDISCON)
ISO-on-TCP	• TCON_IP_RFC	
UDP	• TCON_IP_v4 • TCON_IP_QDN • TADDR_Param • TADDR_SEND_QDN • TADDR_RCV_IP • TCON_QDN_SEC* • TCON_IP_v4_SEC*	Establish connection and send/receive data via: • TSEND_C/TRCV_C • TUSEND/TURCV/TRCV (connection can be terminated via TDISCON)
Modbus TCP	• TCON_IP_v4 • TCON_QDN • TCON_IP_v4_SEC* • TCON_QDN_SEC*	• MB_CLIENT • MB_SERVER

* The redundant system S7-1500R/H does not yet support Open User Communication. If you use the system data types TCON_IP_v4_SEC or TCON_QDN_SEC in the redundant system S7-1500R/H, their ActivateSecureConn parameter must have the value FALSE. This means that the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.

Example: Setting up connection via the system IP address

The following section describes how to set up the connection from the redundant system S7-1500R/H to another CPU via the system IP address.

- In the user program of the redundant system S7-1500R/H, establish the connection with the instruction TSEND_C.
- In the user program of the partners CPU, create a corresponding instruction TRCV_C.

The procedure is described using the example of a TCP connection between the redundant system S7-1500R/H and a CPU 1516-3 PN/DP.

Requirements

- Local connection partners: Redundant system S7-1500R with two CPUs 1513-1 PN System IP address of the PROFINET interface X1 is enabled.
- Remote connection partner: CPU 1516-3 PN/DP
- PROFINET interfaces X1 of the CPUs 1513R and the PROFINET interface X2 of the CPU 1516-3 PN/DP are located in the same subnet.

Setting up a TCP connection in the redundant system S7-1500R/H (send)

To set up a TCP connection to another CPU, proceed as follows:

1. Create an instruction TSEND_C in the user program.
2. Select the instruction "TSEND_C".
3. In the Inspector window, go to "Properties" > "Configuration" > "Connection parameters". On the left side you see the redundant system S7-1500R/H is local end point of the connection:
   - "Interface:": The interface X1 is preset.
   - "Subnet:": If the interface X1 is assigned to an S7 subnet, in STEP 7 displays the name of the S7 subnet.
   - **The option "Use System IP address" is enabled**.
4. For "Partner", select the CPU 1516-3 PN/DP as communication partner under "End point:".
5. For "Partner", select the PROFINET interface X2 of the CPU 1516-3 PN/DP under "Interface:".
6. For "Local", select the setting <new> under "Connection data".
   STEP 7 creates a data block for the connection data in the user program of the redundant system S7-1500R/H.
7. For "Partner", select the setting "TCP" under "Connection type:".
   STEP 7 creates a data block for the connection data in the user program of the other CPU.

Setting up a TCP connection in the connection partner CPU 1516 (receive)

Create an instruction TRCV_C in the user program of the CPU 1516-3PN/DP and configure this suitably ("complementary") for the connection partner.

Setting up connection via the device IP address

To set up an OUC connection via a device IP address of one of the two CPUs:

1. Select a suitable PROFINET interface of the redundant system S7-1500R/H.
2. Disable the option "Use address of the H system".

Connection resources of the redundant system S7-1500R/H (S7-1500)

Maximum number of connection resources of the redundant system S7-1500R/H

The redundant S7-1500R/H system supports a maximum number of connection resources.

The CPU used defines the maximum number of resources for an S7-1500R/H station:

- CPU 1513R: max. 88 connection resources
- CPU 1515R: max. 108 connection resources
- CPU 1517H: max. 160 connection resources

Assignment of connection resources

Communication connections occupy connection resources in the redundant system S7-1500R/H.

Each communication connection to the redundant system S7-1500R/H occupies connection resources in the S7-1500R/H station.

Depending on the IP address used, a communication connection also occupies resources in one or both CPUs of the S7-1500R/H redundant system.

The following table shows in which CPU the communication connection occupies connection resources in dependence on the IP address used.

Connection via...	Connection resources of the station	Connection resources of CPU with redundancy ID 1	Connection resources of CPU with redundancy ID 2
A system IP address	X	X	X
A device IP address of the CPU with redundancy ID 1	X	X	-
A device IP address of the CPU with redundancy ID 2	X	-	X

Display of the assigned connection resources in STEP 7

Requirement: Online connection to the S7-1500R/H redundant system

You will find the connection resources in the Inspector window in the properties of the CPUs. STEP 7 always indicates the connection resources the selected CPU and the S7-1500R/H station.

Verbindungsressourcen

	Ressourcen der Station					Ressourcen des Moduls	
	Reserviert			Dynamisch		CPU 1515R-2 PN (R0/S1)	
Maximale Anzahl der Ressourcen:		10	10	98	98	108	108
	Maximum	Konfigurierte	Verwendet	Konfigurierte	Verwendet	Konfigurier...	Verwendet
PG-Kommunikation:	4	-	2	-	0	-	2
HMI-Kommunikation:	4	0	0	0	0	0	0
S7-Kommunikation:	0	-	0	0	0	0	0
Open User Communication:	0	-	0	0	1	0	1
Web-Kommunikation:	2	-	0	-	0	-	0
Sonstige Kommunikation:	-	-	0	0	0	0	0
Insgesamt verwendete Ressourcen:		0	2	0	1	0	3
Verfügbare Ressourcen:		10	8	98	97	108	105

Behavior of connections during Syncup (S7-1500)

Behavior of communication connections via the system IP address in SYNCUP system state

- HMI, PG and S7 connections are closed temporarily. For a short period during the SYNCUP is not possible to establish connections to the redundant system S7-1500R/H.
- All existing connections of Open User Communication are terminated:
  - Connections with the CPUs of the redundant system have established as active connection partner re-established after the SYNCUP.
  - The redundant system S7-1500R/H sets up new connection end points the passive connection establishment after the SYNCUP.
- The processing of running instances of the instructions TSEND and TRCV is stopped. The block parameter STATUS returns $80C4_H$ (temporary communication error).

Behavior of connections at primary-backup switchover (S7-1500)

Behavior of communication connections via the system IP address during the primary-backup switchover

- Running instances of the instructions TSEND and TRCV are stopped and return the status $80C4_H$ (temporary communication error).
- The new primary CPU re-establishes the connections which the redundant system S7-1500R/H had successfully actively established.
- The new primary CPU sets up new connection end points for the passive connection establishment.

See also

System IP addresses (Page 1059)

Diagnostics of the redundant system S7-1500R/H (S7-1500)

Obtaining an overview of the current state of the R/H system (S7-1500)

Introduction

You obtain an overview of the current status of the R/H system in the project tree.

The following sections explain how you determine this information from the symbols displayed there:

- The current system status of the R/H system
- The current operating modes of the CPUs of the R/H system

- The current roles of the CPUs of the R/H system
- To which of the two CPUs an online connection currently exists

Information about the system states of the R/H systems as well as the operating modes and roles of the CPUs of an R/H system is available in the "S7-1500 Redundant System S7-1500R/H" system manual.

Determining the current system status of the R/H system

The current system status of the R/H system is indicated by the two square overlay icons located on the right-hand side of the R/H system folder icon.

The following table contains the available folder icons and their meaning.

Folder icon of the R/H system	System status of the R/H system
	STOP
	RUN-Solo
	RUN-Redundant

Note

Folder icon and CPU roles

You cannot determine the roles of the individual CPUs from the folder icon for the system states RUN-Solo and RUN-Redundant.

Determining the current operating modes of the CPUs of the R/H system

The current operating mode of a CPU of the R/H system is indicated by the overlay icon that is located at the top right in the folder icon of this CPU.

The following table contains the available folder icons and their meaning using the primary CPU as an example.

Folder symbol of the primary CPU	Primary CPU operating mode
	STARTUP
	STOP
	RUN

Note

Dependency of the operating mode display of a CPU on the existing online connection

In the system states STOP and RUN-Solo you can only determine the current operating mode of the respective CPU for which an online connection currently exists.

The CPU to which an online connection does not currently exist does not have an overlay icon in its CPU folder icon at the top right.

The fact that a CPU currently does not have an online connection to the programming device / PC is indicated by the following icon in the second to last column of the CPU row:

	"Online via partner": The other CPU of the R/H system is connected online to the programming device / PC.

Determining the current roles of the CPUs of the R/H system

The current role of a CPU in the R/H system is indicated by the overlay icon that is located at the bottom right in the folder icon of the CPU. It either consists of the character "P" or the character "B":

- "P" means that the CPU in the R/H system is a primary CPU.
- "B" means that the CPU in the R/H system is a backup CPU.

Determining to which CPU an online connection currently exists

- In the system states STOP and RUN-Solo the following icon is displayed in the second to last column of the CPU row for the respective CPU to which no online connection currently exists:

	"Online via partner": The other CPU of the R/H system is connected online to the programming device / PC.

  You therefore know that an online connection to the other CPU exists.
- In the RUN-Redundant system state the "Online via partner" icon is not displayed. It is superfluous because in RUN-Redundant mode each CPU has all the diagnostic information of the other CPU.

Extensions of the online and diagnostics view for the R/H system (S7-1500)

Overview

There are three online and diagnostics views:

- One for the S7-1500R/H system
- One each for the two CPUs of the S7-1500R/H system

Online and diagnostics view of the S7-1500R/H system

This online and diagnostics view has been newly introduced for the R/H system.

It consists of the following elements:

- "Online access" group with the "Online access" area
- "Diagnostics" group with the "Diagnostics" area

In "Online access" the following extensions exist compared to the online and diagnostics view of a standard CPU:

- In the case of an existing online connection between the programming device /PC and one of the CPUs, the "Status" also indicates the role of the CPU connected online in the R/H system:
  - "Online (via primary CPU)" means the CPU connected online is the primary CPU
  - "Online (via backup CPU)" means the CPU connected online is the backup CPU
- The "Information" provides general information about the R/H system and information to which CPU an online connection is being established when going online.

In "Diagnostics" you get an overview of the current status of the R/H system. The following are displayed:

- The system state of the R/H system
- The pairing state of the two CPUs
- The operating modes of the primary and backup CPU
- The names of the primary and backup CPU (preset or assigned by you) This way you know which CPU is the primary CPU and which CPU is the backup CPU.

Online and diagnostics views of both CPUs of the S7-1500R/H system

First the dependency of the display from the system status is discussed:

- In the RUN-Redundant system state, both the online and diagnostics view of the primary CPU as well the the online and diagnostics view of the backup CPU diagnostics data is displayed. This applies regardless of which CPU has an online connection.
- In all other system states, diagnostics data is only displayed in the online and diagnostics view of the CPU connected online.

The online and diagnostics view of a CPU of the R/H system corresponds almost completely to the display for standard CPUs. The only difference is in the "Online access" group: In case of an existing online connection, the "Status" also indicates the role of the CPU connected online in the R/H system: Primary CPU or backup CPU

Dependencies of the diagnostics views in the project tree and in the online and diagnostics view (S7-1500)

Introduction

The following factors determine which diagnostics are displayed in the project tree and in the online and diagnostics view:

- What is the role of the CPU to which the programming device / PC is connected online? Primary CPU (shortened to "P" in the following table) or backup CPU (shortened to "B" in the following table)?
- What is the system state of the R/H system?

Detailed description of the dependencies

When an online connection exists between the programming device / PC and primary CPU, the following applies:

System state of the R/H system (and operating modes of the CPUs)	Possible diagnostics/diagnostics displays
STOP (P=STOP, B=STOP)	• Diagnostics of the primary CPU is possible. • Diagnostics of the backup CPU is not possible. • Diagnostics of the distributed I/Os is not possible.
RUN-Solo (P=RUN, B=STOP)	• Diagnostics of the primary CPU is possible. • Diagnostics of the backup CPU is not possible. • Diagnostics of the distributed I/Os is possible.
RUN-Redundant (P=RUN-Redundant, B=RUN-Redundant)	• Diagnostics of the primary CPU is possible. • Diagnostics of the backup CPU is possible. • Diagnostics of the distributed I/Os is possible.

If an online connection exists between the programming device / PC and the backup CPU, the following applies:

System state of the R/H system (and operating modes of the CPUs)	Possible diagnostics/diagnostics displays
STOP (P=STOP, B=STOP)	• Diagnostics of the primary CPU is not possible. • Diagnostics of the backup CPU is possible. • Diagnostics of the distributed I/Os is not possible.
RUN-Solo (P=RUN, B=STOP)	• Diagnostics of the primary CPU is not possible. • Diagnostics of the backup CPU is possible. • Diagnostics of the distributed I/Os is not possible.
RUN-Redundant (P=RUN-Redundant, B=RUN-Redundant)	• Diagnostics of the primary CPU is possible. • Diagnostics of the backup CPU is possible. • Diagnostics of the distributed I/Os is possible.

Summary

The dependencies detailed above can be summarized in the following rules:

- In the STOP and RUN-Solo system states only the CPU to which an online connection exists can be diagnosed.
- In the system status RUN-Redundant both CPUs can be diagnosed, regardless to which CPU an online connection exists.
- In the STOP system state the distributed I/Os cannot be diagnosed.

- In the RUN-Solo system state the distributed I/Os can only be diagnosed when an online connection to the primary CPU exists.
- In the RUN-Redundant system status the distributed I/Os can always be diagnosed, regardless to which CPU an online connection exists.

Extensions of the "Online tools" task card for the R/H system (S7-1500)

Overview

The "Online tools" task card shows information of the S7-1500R/H system if an online connection exists to one of the two CPUs.

It consists of the following panes:

- R/H system operator panel
- One CPU operator panel each for both CPUs
- Cycle time
- Memory

R/H system operator panel

The "R/H system operator panel" provides an overview of the current state of the R/H system. The following information is displayed:

- The name of the R/H system
- The current states of the LEDs of the primary CPU
- The system state of the R/H system
- The names and operating modes of the primary CPU and backup CPU

If you click the "STOP R/H-System" button, the effect depends on the current system status:

- In the system states RUN Redundant and RUN-Solo, clicking results in a change to the STOP system state.
- In other system states this button is grayed out.

The "RUN R/H system" button is currently grayed out and therefore without a function.

CPU operator panel for both CPUs

There is an individual CPU operator panel for each of the two CPUs of the R/H system.

First the operator panel of the CPU with redundancy ID 1 is displayed (corresponds to the top CPU in the project tree), then the operator panel of the CPU with redundancy ID 2 is displayed (corresponds to the bottom CPU in the project tree).

In the operator panel of each CPU the following is displayed:

- The name of the CPU and its role in the R/H system (primary or backup CPU)
- The current states of the LEDs of the CPU
- The position of the mode selector of the CPU

The "RUN" and "STOP" buttons are used to switch the CPU operating mode.

Note

Switchover of the operating mode

A switchover of the CPU operating mode using the "RUN" and "STOP" buttons results in the system status changing.

The "MRES" button is used to perform a full memory reset of the CPU.

In contrast to the operator panel of a standard CPU the following extension is available: The role of the CPU in the R/H system (primary or backup CPU) is displayed additionally after the CPU name.

Cycle time

The "Cycle time" pane corresponds to the pane of the standard CPUs of the same name. It shows the values of the CPU to which an online connection currently exists.

The extension compared to a standard CPU is solely that the role of the CPU connected online is displayed in square brackets right at the top in the pane behind the "Cycle time": "Primary" or "Backup"

Memory

The "Memory" pane corresponds to the pane of standard CPUs of the same name. It shows the values of the CPU to which an online connection currently exists.

The extension compared to a standard CPU is solely that the role of the CPU connected online is displayed in square brackets right at the top in the pane behind the "Memory": "Primary" or "Backup"

Connection diagnostics in the R/H system (S7-1500)

Display of the connection resources of a connection which the system IP address of the R/H system uses

Connections to an R/H system that use its system IP address behave differently than in the case of standard connections to exactly one CPU of the R/H system with regard to the used connection resources: Connections via the system IP address use resources in each of the two CPUs and in the R/H station.

The display of the connection resources ("Diagnostics" > "Connection information" of the Inspector window, "Connection resources" group) behaves accordingly.

For more information, please refer to the "Communication" function manual.

Comparison functions for an R/H system (S7-1500)

You can perform an offline/offline comparison and a offline/online comparison for this purpose.

The special features when comparing in the connection with R/H systems are described below.

Compare offline/offline (software)

With an R/H system you can only use an R/H CPU for the comparison (software) with another CPU.

The execution of the offline/offline comparison is basically no different from the offline/offline comparison with standard CPUs.

Since SW objects are only displayed for the R/H-CPU with the redundancy ID "1" (top CPU in the project tree), you can also only use this CPU for a comparison.

If you drag the icon for the R/H system (both CPUs) from the project tree to the compare editor, STEP 7 automatically selects the CPU with the redundancy ID "1".

An offline/offline comparison (software) of the two R/H CPUs (comparison of primary CPU with backup CPU) is **not** possible.

Offline/offline comparison (hardware)

You can perform a hardware comparison between two S7 CPUs.

Examples:

- Comparison of two complete R/H systems:
  Drag the respective R/H stations to the compare editor to place them there. The hardware structure of both stations including the local modules and distributed I/O devices is compared.
- Comparison of the two CPUs of the same R/H system
- Comparison of first or second CPU of two R/H systems in each case
- Any CPU of an R/H system with any other S7-1500 CPU

For example, you cannot compare an R/H CPU with PC systems (other design).

Offline/online comparison

The execution of the offline/online comparison is basically no different from the offline/online comparison with standard CPUs:

As a starting point for this comparison, you can either select the R/H station or one or both CPUs of the R/H station in the project tree and then select the shortcut menu command "Compare > Offline/Online" as usual. STEP 7 compares with the online CPU that has the role "Primary".

There is no separate menu command for the offline/online comparison with the backup CPU. If you want to compare an offline CPU (R/H system) with the backup CPU as the target device, you must first establish an online connection to the backup CPU using the shortcut menu command "Online connection to backup CPU" and then start the offline/online comparison.

R/H system is offline

If both R/H CPUs of the system are offline, the comparison of the R/H system with the primary CPU starts.

STEP 7 then establishes a connection to the primary CPU. If no online path has yet been set, the Online Connect dialog opens and suggests the primary CPU as the target device. After acknowledging the dialog ("Connect" button), the compare editor starts and shows the differences between offline and online devices of the primary CPU.

R/H system is online

If one of the two R/H CPUs is online, STEP 7 uses the corresponding target device, to which the online connection exists, for the offline/online comparison.

Where can you find additional information? (S7-1500)

You will find more information on the diagnostics of S7-1500R/H redundant systems in the "Diagnostics" function manual.

1.4.1.4 Configuring central racks

In the chapters below you will learn how to configure central racks. For this you will also find information on addressing modules.

Overview for configuring modules

Step	Description
1	Select a central module (Page 1088).
2	Specify the properties of the modules (Page 1090).
3	Specify the input and output addresses (Page 1097).

See also

Addressing modules (Page 1097)

Addressing modules (Page 1096)

Rules for modules (Page 1077)

Rules for modules (Page 1078)

1.4.1.5 Rules for modules

Introduction

Specific slot rules apply for S7-300 modules.

When you select a module in the device view of the hardware catalog, all possible slots for the selected module are marked in the rack. You can only drag modules to marked slots.

If you insert, move or swap a module, the slot rules are also applied.

Slot rules for S7-300

In addition to the general slot rules, the following rules also apply for S7-300:

- You can use up to 4 racks.
  Rack 0:
  - Slot 1: Only power supply (e.g. 6ES7 307-...) or empty
  - Slot 2: Only central rack (e.g. 6ES7 314-...)
  - Slot 3: Interface module (e.g. 6ES7 360-.../365-...) or empty
  - Slot 4 to 11: Signal modules or function modules, communication processors, or empty

  Racks 1 to 3:
  - Slot 1: Only power supply module (e.g. 6ES7 307-...) or empty
  - Slot 2: Empty
  - Slot 3: Interface module (e.g. 6ES7 361-.../365-...)
  - Slot 4 to 11: Signal module or function module, communication processors, (depending on the interface module used) or empty
- In rack 3 when using a CPU with integrated I/O you cannot use slot 11.
- You must mount modules side-by-side without gaps.

Note

Slot 3 in S7-300 is reserved for the interface module. If you use a single rack, this slot will not be assigned because you do not have to plug an interface module. You do not have a gap in the real structure because otherwise the backplane bus would be interrupted!

An ET200m with active backplane bus can have gaps in the allocation.

Special rules for DM 370 dummy module

You can insert the DM 370 dummy module (6ES7 370-0AA01-0AA0) in place of a module that will be used later.

The DM 370 dummy module reserves address space of 1 byte for a module in the modular DP slave (e.g. for a digital I/O module).

1.4.1.6 Rules for modules

Introduction

Specific slot rules apply for S7-400 modules. The rules for the assignment of components in an S7-400 rack depend on the type of rack used.

When you select a module in the device view of the hardware catalog, all possible slots for the selected module are marked in the rack. You can only drag modules to marked slots.

If you insert, move or swap a module, the slot rules are also applied.

Slot rules of the S7-400 for central racks

In addition to the general slot rules, the following rules for components on a central rack apply for S7-400:

- Only insert power supply modules in slot 1 (exception: redundant power supply modules)
- insert a max. of 6 interface modules (transmission IMs); max of 2 with power transmission
- connect a max. of 21 expansion racks to the central racks through interface modules
- connect a max. of 1 expansion rack **with power transmission** to an interface of the transmission IM (IM 460-1 with IM 461-1)
- max. of 4 expansion racks **without power transmission** (IM 460-0 with IM 461-0 or IM 460-3 with 461-3)

Slot rules of the S7-400 for expansion racks

In addition to the general slot rules, the following rules apply for components on extension racks for S7-400:

- Only insert power supply modules in slot 1 (exception: redundant power supply modules)
- insert the interface module (recipient IM) only on the outer right slot (slot 9 and slot 18)
- Insert K-bus modules only in expansion racks with a number no larger than 6 (otherwise it cannot be accessed)

Special rules for redundant power supply modules

Redundant power supply modules can be inserted into a rack twice. You can find out if the power transmission module is redundant by looking in the "Information" palette on the "Hardware Catalog" task card.

The following rules must be observed:

- You can only insert redundant power supply modules into the designated racks (recognizable from the informational text in the "Hardware Catalog" task card).
- The redundant power transmission modules must be inserted into slot 1 and the connecting slot (no gaps allowed!).
- Redundant and non-redundant power transmission modules may not be inserted in the same rack (this means that no "mixed operation" is possible).

1.4.1.7 Rules for modules

Use of S7-1500 modules

Introduction

The CPUs of the S7-1500 automation system are modular controllers for medium- and high-performance automation system solutions for discrete and continuous processes.

Application area

The S7-1500 automation system is designed for system solutions in process and manufacturing automation. As a universal automation system, the S7-1500 automation system represents the optimal solution for centralized and decentralized design applications.

The scalable structure allows you to align your control system's design exactly to the specific needs of your location.

The S7-1500 automation system complies with IP 20 degree of protection and is intended for installation in a control cabinet.

Structure

The S7-1500 automation system is installed on a mounting rail and can comprise up to 32 modules. Bus connectors are used to interconnect the modules. Power is supplied via the backplane bus; the CPU itself can also supply power to several modules. If the power consumed by the modules exceeds the amount supplied by the CPU, you can insert up to three additional power supply modules.

Compact design

The S7-1500 automation system is also available in a compact design. The CPU is then already equipped with integrated inputs and outputs.

The compact CPUs can be expanded with all modules of the S7-1500 automation system. The entire expansion is a total of 32 modules with the CPU and the modules integrated in the CPU (refer to the manuals CPU 1511C-1 PN (http://support.automation.siemens.com/WW/view/en/109478675)and CPU 1512C-1 PN (http://support.automation.siemens.com/WW/view/en/109478676)).

See also

Slot rules for S7-1500 modules (Page 1081)

Configuration of an S7-1500 station (Page 1081)

Power budget calculation (Page 1084)

Checking the power budget (Page 1084)

Slot rules for S7-1500 modules

Specific slot rules apply to S7-1500 modules. When you select a module in the device view of the hardware catalog, all possible slots for the selected module are marked in the rack. You can only drag modules to marked slots. The slot rules apply whenever you insert, move or replace a module.

Slot rules for S7-1500

The following slot rules apply to S7-1500:

- In addition to the CPU, you can also insert up to 31 modules on one rack.
- You can insert a maximum of three power supply modules. Together with the CPU, these modules supply power to all downstream devices on the rack. The power consumption of the individual modules determines how many power supply modules are required, at a minimum.
  - One load current supply module PM or one system power supply module PS can be inserted to the left of the CPU (slot 0).
  - Up to two additional standard system power supply modules (PS) can be inserted to the right of the CPU (slots 2 to 31).

Note

Power segments

A power supply module and modules supplied by it form a power segment. A power supply module always supplies power to the modules installed on its right. The power segment ends at the next power supply module and a new power segment begins for the next power supply module and the modules installed on its right. The modules located on the left of the power supply module are either supplied from the upstream power supply module or the CPU.

See also

Power budget calculation (Page 1084)

Configuration of an S7-1500 station (Page 1081)

Using extended retentive data area with PS 60W 24/48/60VDC HF (Page 1085)

Configuration of an S7-1500 station

The individual S7-1500 CPUs can be expanded with a variety of I/O modules. Besides analog and digital input and output modules, you can use technology modules, communication modules and special modules. In spite of their high performance, S7-1500 CPUs are subject to limits on the number of I/O modules and configured IO systems in situations where the hardware configuration is very large. These limits vary according to the CPU used. When you configure your hardware you should therefore observe the limits described in the technical specifications sheet for the respective CPU.

Maximum configuration of a station

When the hardware configuration is compiled, a check is made to determine whether the station configuration exceeds maximum limits. However, you are advised to make sure that any quantity limitations are met when you are configuring the S7-1500 station. In particular, when creating your hardware configuration you should check the technical specifications for your devices for the following criteria:

- Maximum number of communication modules for PROFINET, PROFIBUS and Industrial ETHERNET as well as communication processors
  In the case of an S7-1513 (6ES7 513-1AL00-0AB0), for example, the maximum number allowed is six.
- Maximum number of distributed PROFINET IO systems and distributed PROFIBUS DP systems.
  IO systems that are connected via an I-device do not have to be counted toward the maximum number. In this case, the rules for the maximum configuration of the I-device apply.
- Maximum number of distributed I/O devices (IO devices and DP slaves)
  The number includes all distributed I/O devices that are assigned to the CPU, regardless of the interface to which they are connected (integrated interface or CP/CM interface). The CPU is only included in the IO devices if it is configured as an I-device.
  If I-devices are connected to the PROFINET IO system, the IO devices on an I-device's lower-level PROFINET IO system are not counted.
- Maximum number of modules/submodules
  The maximum number of pluggable modules is very large and is only rarely reached in conventional hardware configurations of an S7-1500 automation system.
  If modules with submodules are used in very large installations, the CPU's available resources for modules/submodules can be exceeded.
  Simple I/O modules, which have no submodules, require a single resource.
  Complex modules can have multiple submodules and require one resource for each submodule. A module with submodules also requires an additional resource (which represents the whole module).
  Example:
  An IM 155-5PNHF interface module requires five resources based on the following method of counting:
  Two resources for the PROFINET interface's two ports, one resource for the interface of the PROFINET interface, one resource for the submodule for the interface module's parameters. As explained above, an additional resource is added because the module has submodules.

Example

The following figures show an example of a hardware configuration and illustrate the counting method described above:

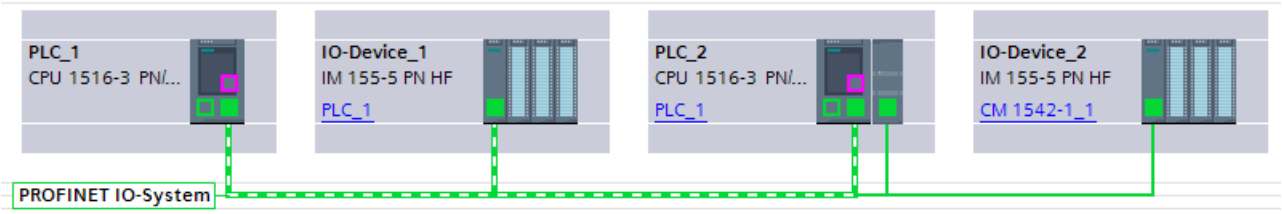

Figure 1-20 In this figure, PLC_1 is configured as the IO controller. IO-Device_1 and PLC_2 are devices on the IO system of PLC_1. Although IO-Device_2 is connected to the same network, it is not a part of the IO system. The IO system of PLC_1 therefore has two devices.

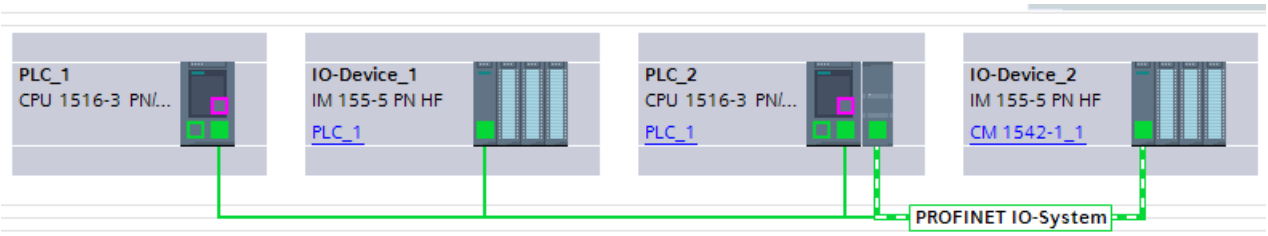

Figure 1-21 The second figure shows that while PLC_2 is an I-device in the IO system of PLC_1, it is simultaneously also an IO controller. A lower-level IO system is spanned via the communication module and this IO system is connected to IO device_2. In total, two devices are counted for PLC_2.

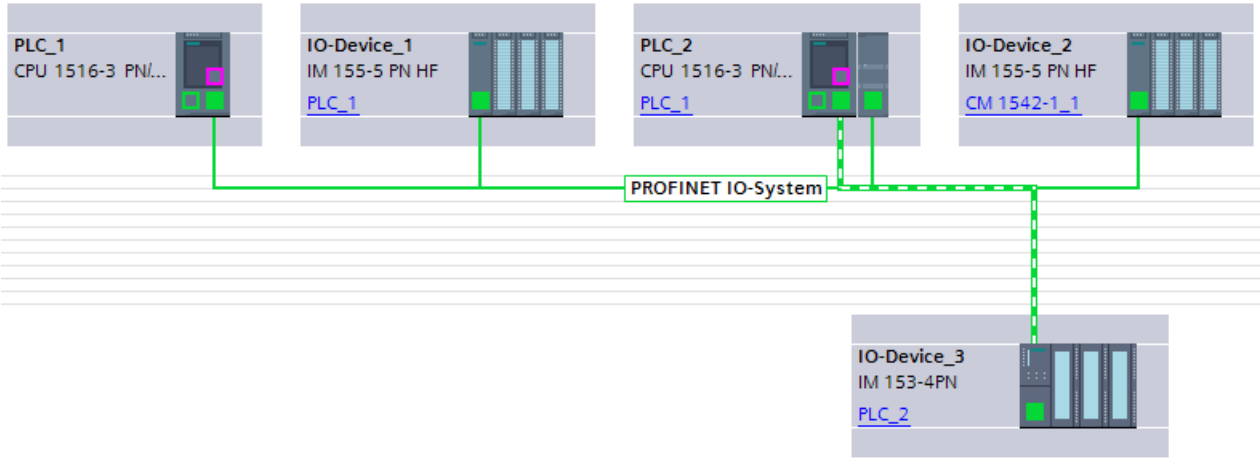

Figure 1-22 In the third figure, an additional IO system is assigned to PLC_2. PLC_2 now has two IO systems, one connected to the internal interface and one connected to the communication module. Another device must be counted for the new IO system. In total, three devices are counted for this hardware configuration for PLC_2 - one I-device (PLC_2 is assigned to PLC_1) and two I/O devices (assigned to PLC_2).

See also

Slot rules for S7-1500 modules (Page 1081)

Power budget calculation (Page 1084)

Power budget calculation

Introduction

All I/O modules draw the power needed to control the electronics from the backplane bus. This power is fed from the CPU and, if required, also by additional system power supplies to the backplane bus.

Power supply/consumption ratio check while configuring

STEP 7 helps you to check the power supply/consumption ratio. All the inserted I/O modules and their maximum power consumption are listed in a table. The power consumption is offset by the plugged system power supply modules. Thus, you see at a glance whether sufficient power is available for all modules.

You can find the table in the "System power supply" section of the Inspector window after you have selected the CPU and then selected the "Properties" tab.

In addition, a check is made during the configuration to determine if sufficient power is available. If the I/O modules consume more power from the backplane bus than is supplied, the corresponding table cells are colored red and you receive a corresponding warning after the hardware configuration is compiled.

Power supply/consumption ratio check by the CPU

During operation, the CPU monitors compliance with the power supply/consumption ratio:

- At every Power ON
- At every change of the installed hardware

If the CPU identifies a backplane bus overload, it switches off all I/O modules that are supplied by the CPU or the system power supply and enters an alarm in the diagnostic buffer. The CPU does not start up while the supply/consumption ratio is negative.

Causes of overload

An overload may be caused by a mismatch between the installed and configured hardware. For example, the system power supply units required for operation are not inserted or are not switched on or there are more modules inserted than configured.

See also

Checking the power budget (Page 1084)

Slot rules for S7-1500 modules (Page 1081)

Configuration of an S7-1500 station (Page 1081)

Checking the power budget

You can see whether sufficient power is available for all I/O modules as you are configuring a station. To do so, you must first specify whether or not the CPU is supplied with voltage.

Procedure

To activate the power supply of the CPU and evaluate the power supply/consumption ratio, follow these steps:

1. Select the S7-1500 CPU, for example, in the network view.
2. Open the "Properties" tab in the inspector window.
   The properties of the CPU are displayed.
3. Select the "System power supply" item in the area navigation.
4. Select the "Connection to supply voltage L+" option if the CPU is supplied with voltage. The option is activated by default. As a result, the power consumption of the CPU is included in the power supply/consumption ratio.
5. Check the "Power segment overview" table to find out if the power budget is positive. If the power supply/consumption ratio is negative, the underpowered modules is marked in red.

See also

Power budget calculation (Page 1084)

Using extended retentive data area with PS 60W 24/48/60VDC HF

If you are using the system power supply PS 60W 24/48/60VDC HF, you can use the extended retentive data area for an S7-1500 CPU.

You use the system power supply PS 60W 24/48/60VDC HF for processes that require a large amount of retentive data for a seamless transition after a power failure.

The values for the extended retentivity depend on the CPU you use. The values can be found in the technical specifications of the CPU manuals or in the corresponding Product Information.

Link to the Product Information:

https://support.industry.siemens.com/cs/ww/en/view/68052815 (https://support.industry.siemens.com/cs/ww/en/view/68052815)

Requirement

Requirements for use of the system power supply PS 60W 24/48/60VDC HF to use the extended retentive data area:

- CPU S7-1500 firmware version V2.1 and higher
- Compatible hardware function level of the CPU S7-1500
  With the following CPUs you can only use the extended retentive data area **as of hardware function level FS03**:
  - CPU 1517-3 PN/DP, CPU 1517F-3 PN/DP
  - CPU 1518-4 PN/DP, CPU 1518F-4 PN/DP

  With the following CPU you can only use the extended retentive data area **as of hardware function level FS02**:
  - CPU 1518F-4 PN/DP ODK

  The listed CPUs do not permit download of the user program (software) when you use the extended retentive data area.
  Remedy: Use a CPU with the matching HW function level.
- STEP 7 as of V14 Service Pack 1

Rules for configuration of an S7-1500 with PS 60W 24/48/60VDC HF

- The system power supply PS 60W 24/48/60VDC HF must be inserted to the left of the CPU (slot 0).
  For the distributed configuration of an ET 200MP the system power supply may only be inserted to the left of the interface module. The power supply does not extend the retentive data area of the CPU in case of distributed use.
  STEP 7 shows the permissible slots for placement of modules with a blue frame so that an incorrect configuration is not possible.
  The figure shows slot 100 for a system power supply module (PM) to the left of slot 0. This slot is only visible when a CPU as of firmware version 2.1 and higher is inserted.

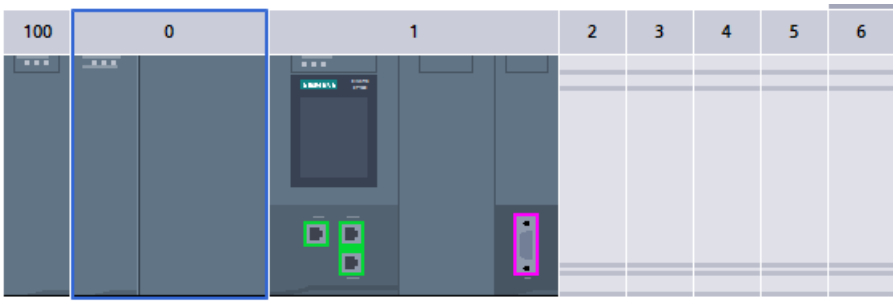

- A system power supply PS 60W 24/48/60VDC HF cannot be inserted to the right next to the CPU or to the right of an interface module (ET 200MP). If you need additional system power supplies to increase the capacity for the backplane bus, use a standard system power supply as of slot 2.

- If you are using the PS 60W 24/48/60VDC HF, the 24 V supply of the CPU is not taken into account in the power balance calculation. This means you should not connect the 24 V DC to the CPU. The CPU parameter "System power supply" therefore must be set to the option "No connection to supply voltage L+".

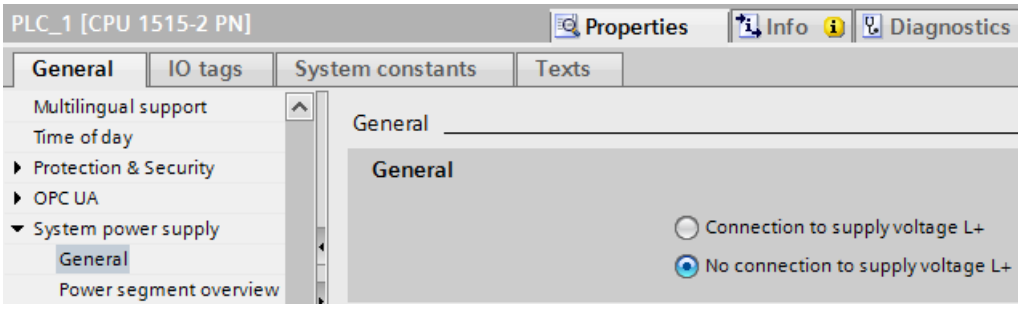

- The parameter "Comparison preset to actual module" must be set to the value "Startup CPU only if compatible".
  Reason: The retentivity of the entire CPU work memory (data) is only guaranteed when a PS 60W 24/48/60VDC HF is inserted.
  When you insert the PS 60W 24/48/60VDC HF, STEP 7 automatically sets this value for the slot.

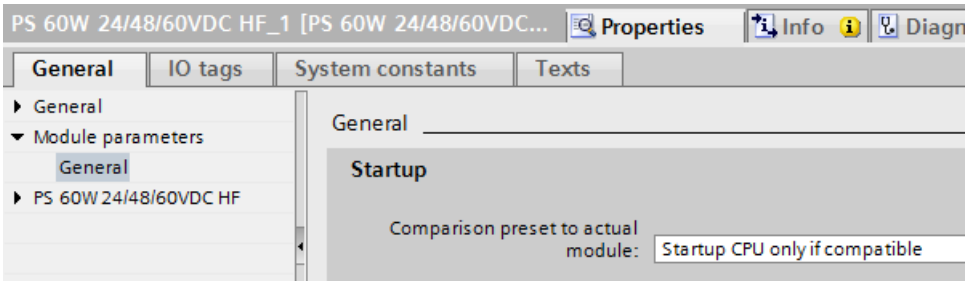

STEP 7 checks the rules listed above during compilation or loading of the user program to the CPU.

Replacement parts scenario

You can use the PS 60W 24/48/60VDC HF system power supply as replacement for a configured PS 60W 24/48/60VDC (standard PS) in slot 0. In this case you do not benefit from the extended retentive data area.

However, you can **not** use the standard PS as replacement for a configured PS 60W 24/48/60VDC HF.

Missing diagnostics of the PS 60W 24/48/60VDC HF system power supply

In case of POWER OFF, saving the extended retentive data is most important. The CPU as of FW V2.1 and higher does **no longer** output the following diagnostics of the PS 60W 24/48/60VDC HF:

- Supply voltage fault
- Switch position Off

See also

Slot rules for S7-1500 modules (Page 1081)

1.4.1.8 Selecting modules

Introduction

Select a CPU from the hardware catalog and place it, together with a rack, in the network view. On this device drag the desired modules from the hardware catalog; they are arranged automatically on the rack.

Selecting the components in the hardware catalog

Each hardware component is displayed as a folder in the hardware catalog. When you open this folder, you see the different versions of the selected hardware component with its respective article numbers.

There will be an example of how to set up a CPU with a rack in network view.

Requirement

- The hardware catalog is open.
- You must be in the network view.

Procedure

To select a CPU from the hardware catalog, proceed as follows:

1. In the hardware catalog navigate to the folder with the desired CPUs.
2. Open the folder with the desired CPU type; you will see all article numbers for the selected CPU type.

3. Click on a CPU article number to get information about the selected CPU in the "Information" pane.

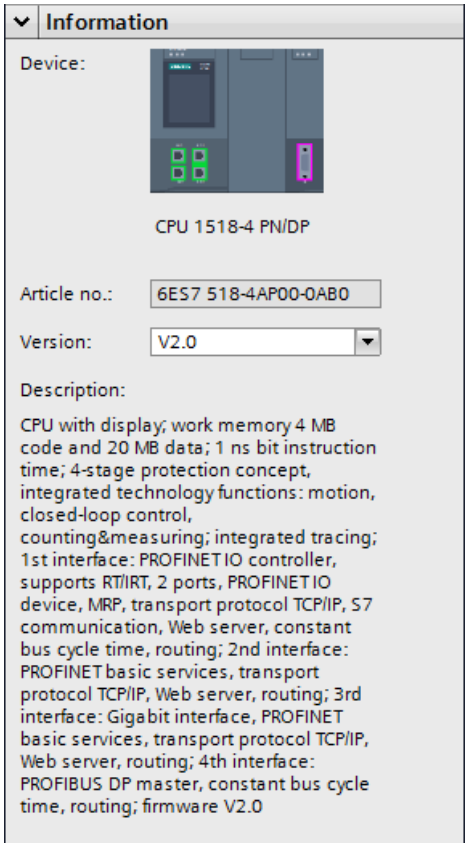

4. Set the matching firmware version of your CPU in the drop-down list box "Version".
5. Set up the CPU and a rack. You have the following options:
   - Use drag-and-drop to drag the CPU from the hardware catalog into network view.
   - Double-click the CPU entry in the hardware catalog.

1.4.1.9 Selecting fail-safe modules

Introduction

Fail-safe automation systems (F systems) are used to control processes with a safe state that can be achieved immediately on shutdown. F systems control processes that can be shut down immediately without endangering human beings or the environment.

Selecting the F components in the hardware catalog

Each hardware component is displayed as a folder in the hardware catalog. Fail-safe hardware components are displayed with yellow symbols in the hardware catalog in contrast to the blue hardware components that are not fail-safe.

The following figure shows excerpts from the hardware catalog with two CPUs and two digital output modules, normal in blue and F in yellow:

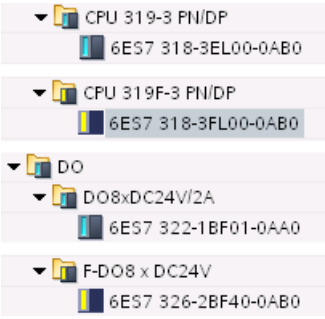

Based on this color coding, you can identify F components in the hardware catalog simply and quickly.

Display of the F components in the hardware and network editor

If you use fail-safe devices and "normal" devices in the graphic views of the hardware and network editor, you can recognize them simply by the yellow color-coding of the devices.

The following figure shows two CPUs. The F CPU is shown in yellow:

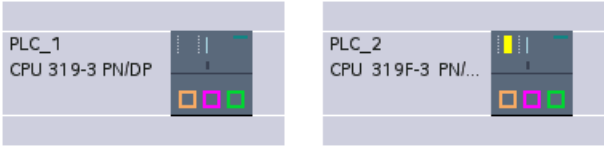

Fail-safe modules in racks or devices can also be recognized in the device view due to their yellow color coding. Since modules that are not capable of networking are not displayed in the network or topology view, devices with F components without networking capability are represented by a special symbol in the network and topology view.

The following figure shows a "normal" CPU and an F-CPU in the network view that both contain F components:

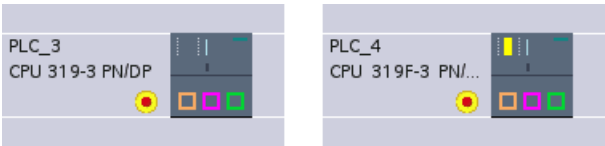

Based on the color coding of the fail-safe devices and modules, you can also identify the F components quickly in the graphic views of the hardware and network editor.

1.4.1.10 Changing the properties of CPUs

Each hardware component (module, interface, or interface module) has pre-selected properties, e.g. for analog entry modules pre-selected measurement types and measurement ranges.

Properties of CPUs

The properties of the CPUs have special significance for system behavior. For example for a CPU you can set:

- Startup characteristics
- Local data storage areas
- Priorities for interrupts
- Memory areas
- Magnetic latching
- Clock memory
- Protection level
- Password

The entry possibilities specify what is adjustable and in which value ranges. Fields that cannot be edited are disabled or are not shown in the properties window.

You can assign parameters for an integrated MPI interface or PROFIBUS DP interface in the properties of an interface, for example, and connect them with an existing subnet.

Requirement

You have already arranged the hardware components for which you want to change properties, on a rack.

Procedure

To change the properties and parameters of the hardware components, proceed as follows:

1. In the device or network view, select the hardware components that you want to edit. You can also select individual parts of a hardware component, e.g.:
   - Interface
   - Input
   - Output
2. Edit the settings for the selected object:
   - For example in the device view you can edit addresses and names.
   - In the Inspector window additional setting possibilities are available.

You do not need to confirm your entries, the changed values will be applied immediately.

Additional possibilities for parameter assignment

For the S7-300/400 automation systems, for example, you have the possibility of setting parameters for analog modules in the user program. To do this, open the system functions WR_PARM (SFC 55), DP_DPARM (SFC 56) and PARM_MOD (SFC 57) in the user program. However these settings will be lost at start-up or restart (warm restart).

Additional information

For additional information about system functions and standard functions for S7-300/400, refer to the system software reference manual for S7-300/400.

1.4.1.11 Linking racks

Introduction

Depending on the device used, the option exists to link racks using interface modules for some S7-300 CPUs.

Linking racks

To link racks through an interface module, another rack must be moved from the hardware catalog into the device view. Then a sender module is plugged into the central rack and a receiver module into the new rack. A link is automatically established between sender and receiver modules.

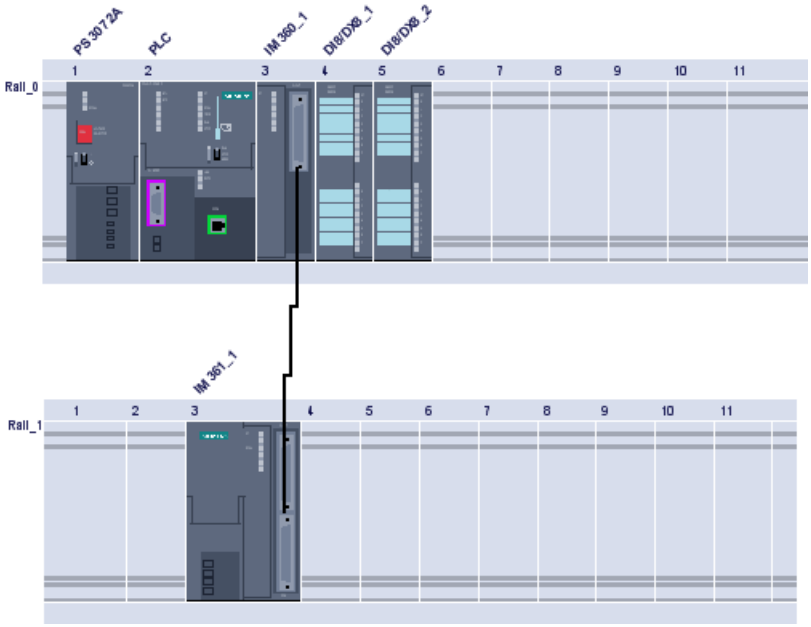

With S7-300, for example, you can link up to four racks. The interface modules are always plugged in slot 3.

Note

If your configuration does not include an expansion, slot 3 will always be identified as empty even if the module in slot 4 is actually connected directly to the CPU and there is no gap.

Use the following interface modules:

- To expand by one rack precisely:
  Racks 0 and 1: IM 365
- To expand by up to 3 racks:
  Rack 0: IM 360
  Racks 1 to 3: IM 361

If you prefer a manual link, first delete the connection line representing the link. Then press and hold down the mouse button and draw a new connection line between the interfaces of the sender and receiver modules.

Expandable racks

Depending on the device used, several CPUs (e.g. IM 151-1) have expandable racks whose groups of slots can be expanded and collapsed using the arrow symbols above the expandable slot. When the groups of slots are collapsed, the first and last of the group's slot numbers are displayed.

The following diagram shows a section of an expanded rack:

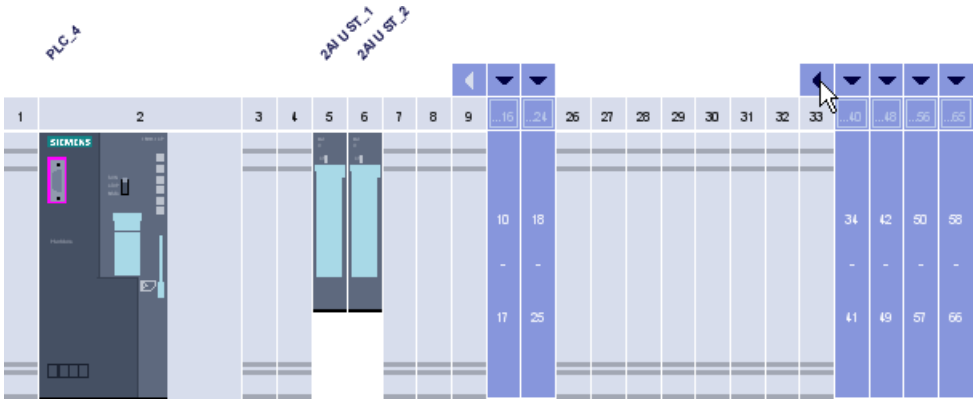

Groups of slots into which modules have already been plugged cannot be collapsed. The first group of slots and device cannot be collapsed. You can tell from the grayed out arrow symbol if a group of slots cannot be collapsed.

1.4.1.12 Linking racks

Expanding a central rack by adding an expansion rack

Configuring expansion racks in SIMATIC 400

In SIMATIC 400, the expansion options are more complex due to be various racks and interface modules.

All expansion racks connected to an interface of the IM send module of the central rack form a chain.

The following figure shows two groups of three expansion racks, with each group connected to an interface of the IM send module.

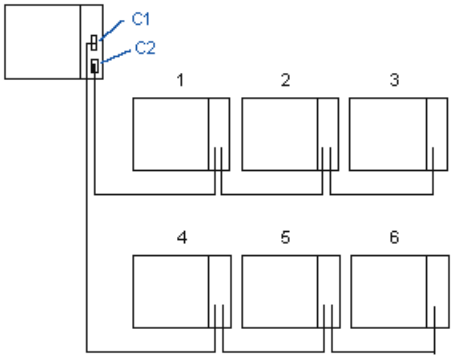

Rules for linking expansion racks

Linking expansion racks

If you link expansion racks (SIMATIC 400) to an interface of the interface module (IM send module) of the central rack, the following properties of the IM send module and IM receive module must match:

- Power transfer (with/without)
- Type of link (central/distributed)
- K bus transfer (with/without interrupt transfer)

Slot rules of the S7-400 for expansion racks

In addition to the general slot rules, the following rules apply for components on extension racks for S7-400:

- Insert power supply modules in slot 1
- Insert the interface module (receive IM) only in the outer right-hand slot (slot 9 or slot 18)
- Insert K-bus modules only in expansion racks with a number no larger than 6 (otherwise it cannot be accessed)

Arrangement of the expansion rack

Basic procedure

To arrange expansion racks and drag connections between the interface modules in the racks, follow these steps:

1. Select a CPU rack (for example CPU 417) from the "hardware catalog".
2. Drag the controller to the device view.
3. Assign the controller modules (for example, power supply and IM 460) to the rack.

4. Select suitable expansion racks (for example ER1) from the "hardware catalog".
5. Drag the expansion racks one after the other to the device view.
6. Assign modules (for example, power supply or IM 461) to each rack.
   Important: The interface modules must be inserted in all racks to allow you to link racks together.
7. Connect the interface of the IM send module with the iterface of the IM receive module of the required rack.
   Connection lines then show the assignment of the racks among themselves.

The following figure shows groups of two expansion racks, with each group connected via a receive IM to an interface of the send IM.

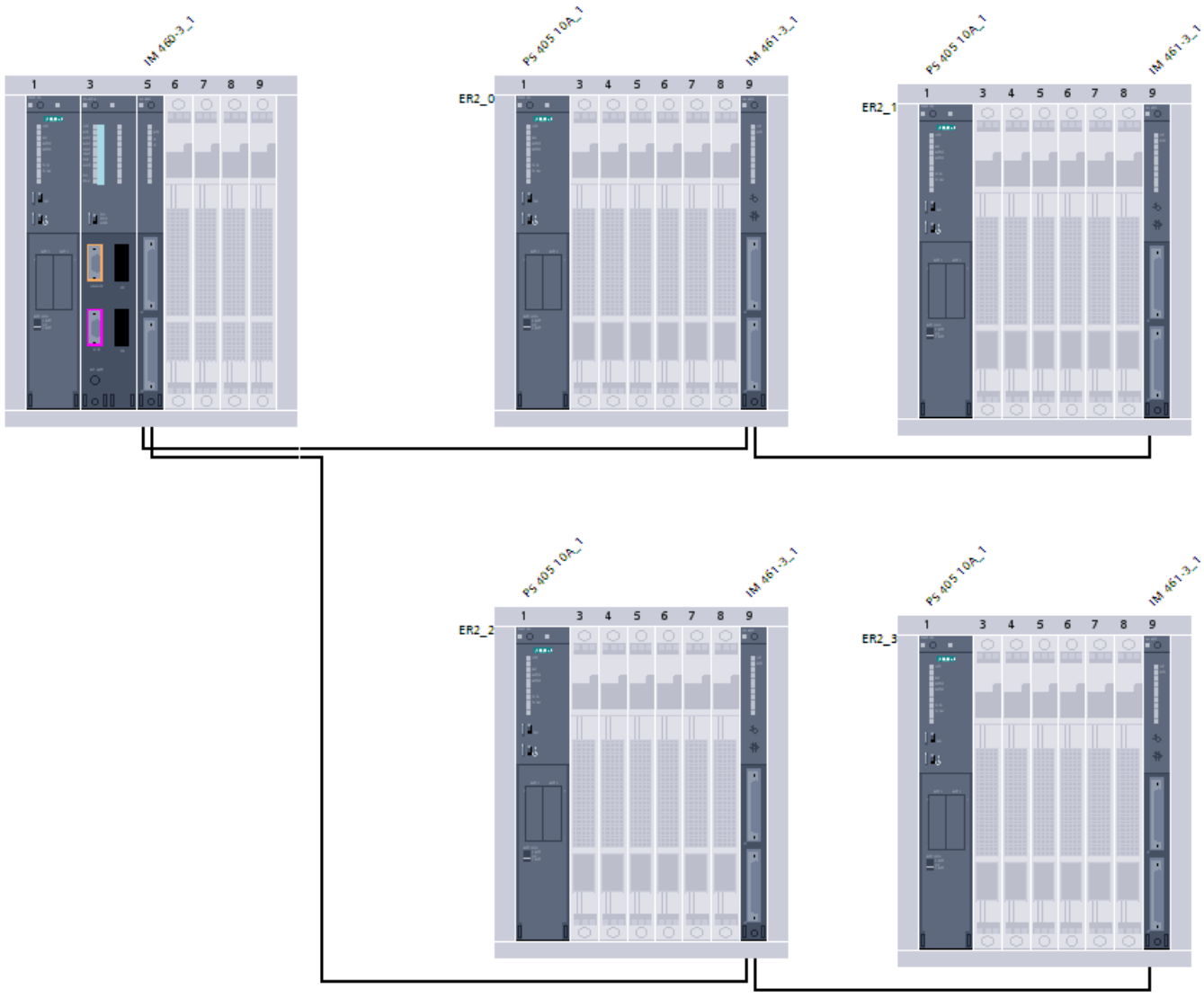

1.4.1.13 Addressing modules

Addressing modules

Introduction

In the device overview, you see the addresses or address ranges of the modules in the I address and Q address columns.

Input and output addresses are assigned automatically when modules are inserted in the rack. The address of the first channel is the start address of a module. The addresses of the other channels are derived from this start address. The end address is derived from the module-specific address length.

Special feature: Input addresses with asterisk (*)

In the device view certain addresses from the input address range are marked with an asterisk (*). In this case these are diagnostics addresses, e.g. interface modules that do not address user data, but rather that are used for system diagnostics.

Difference between device addresses and I/O addresses

When assigning addresses a distinction is made between device addresses and I/O addresses (input/output addresses):

- Device addresses are addresses of programmable modules (MPI addresses, PROFIBUS addresses, Industrial Ethernet addresses). They are required to address the different stations of a subnet, for example, to download a user program to a CPU.
- I/O addresses are used to read inputs or set outputs in the user program.

Special feature: MPI addresses of FMs and CPs (S7-300)

For CPs and FMs with own MPI address, the default MPI address is set by STEP 7.

The MPI address can be changed subsequently (in the inspector window via "Properties > MPI interface").

For CPs and FMs without own MPI address, the CPU addresses these modules via the K bus using the module rack and slot number.

See also

Specifying input and output addresses (Page 1097)

Assigning addresses to a location in the program (Page 1110)

Addressing modules

Introduction

In the device overview, you see the addresses or address ranges of the modules in the I address and Q address columns. There are other addresses as well, which are explained below.

I/O address

I/O addresses (input/output addresses) are required to read inputs and set outputs in the user program.

Input and output addresses are assigned automatically when modules are inserted in the rack. The address of the first channel is the start address of a module. The addresses of the other channels are derived from this start address. The address end is obtained from the module-specific address length.

Device address (e.g., Ethernet address)

Device addresses are addresses of programmable modules (Industrial Ethernet addresses). They are required to address the different stations of a subnet, for example, to download a user program to a CPU.

Hardware identifier for identifying modules and submodules

In addition to the I addresses and Q addresses, a hardware identifier (HW ID) is assigned automatically and is used to address and identify the module. Submodules (units of a module), such as integrated counters, also receive a hardware identifier. The hardware identifier consists of an integer and is output by the system with diagnostics alarms to allow the faulty module or the faulty submodule to be localized. In addition, the hardware identifier is used for a number of instructions to address the corresponding module. The hardware identifier cannot be changed.

See also

What you need to know about HW identifiers (Page 1099)

Specifying input and output addresses

The input and output addresses are indeed preset automatically, however you can change the addressing retroactively.

Requirement

- You are in the device view.
- The module is a compact DP slave or is inserted in one of the following objects:
  - Central rack
  - Expansion rack
  - DP slave
- The CPU allows free address assignment.

Procedure

To change the preset address range proceed as follows:

1. In the device view, click on the module for which you want to set the start address.
2. Click on "I/O Addresses" under "Properties" in the Inspector window.
3. Under "Start address" enter the required start address.
4. Press <Return> or click on any object to accept a modified value.

If you have entered an invalid address a message will be shown in which you can accept a suggested address or enter a different address.

Note

You can also change the addresses directly in the device overview. If address specification is not enabled by the CPU, the address will be disabled in the display.

For modules within a local bus segment, formed by an FM (S7-300) or for special FMs (S7-400), additionally assign another start address. In addition to the start address for the CPU, the module will then also have a start address for the FM. In the device overview in this case the start address is always displayed from the perspective of the FM.

Information about diagnostic addresses

Introduction

Diagnostic addresses are used to address components that do not have user data, for example interface modules or ports.

Input addresses as diagnostic addresses

Use diagnostic addresses to address components that do not have user data. STEP 7 assigns an e-address automatically to the components "from the top", i.e. from the highest e-address downward. This e-address is named diagnostic address.

This input address is not used to read in input values (process values). The diagnostic address is used, for example, to address components in the instructions "DPNRM_DG" or "RDREC".

Examples

To read the diagnostic data record of an DP slave, use the diagnostic address of the interface module of this DP slave with the instruction "DPNRM_DG" ("LADDR" parameter).

To read the data record containing the neighborhood relationships of a port, use the diagnostic address of this port for the instruction "RDREC" ("ID" parameter).

Addressing with input and output groups

To address the input module, use the start address with the instructions "RDREC" or "WRREC". If the addresses 5 and 6 are assigned to an input module, 5 is the start address. Set the value "DW#16#0005" for the "ID" parameter.

To address the output module, use the start address also with the instructions "RDREC" or "WRREC". Set the "ID" parameter to the value "DW#16#8005" for the start address 5. For an output module, bit 15 (value "1") must be set. Therefore, with the start address 5, the value of DW#16#8005 results for the "ID" parameter.

To address a hybrid module (input and output module), enter the smaller of the two start addresses in the "ID" parameter. If the input and output area have the same start address, then use the identifier for an input address (bit 15 of the address is not set, "0").

See also

Reading out topology data of a PROFINET system (http://support.automation.siemens.com/WW/view/en/38566021)

What you need to know about HW identifiers

Introduction

Every hardware object, e.g., a module, interface or port of a module, or even transfer areas of I-devices, has a HW identifier that STEP 7 assigns automatically when a hardware component is configured.

In S7-1500 and S7-1200 systems, these hardware identifiers have a similar function to the diagnostic addresses of HW components without user data, or the start addresses of HW objects with user data in S7-300/400 systems. The hardware identifiers of the modules are insensitive to changes to the I/O addresses and are therefore retained. When I/O addresses are changed, the instructions that use a hardware identifier do not have to be adapted.

HW identifiers in the "System constants" tab in the Inspector window

When you select a hardware object in the device view and select the "System constants" tab in the Inspector window, all HW identifiers of the hardware object are displayed as system constants with name and type.

The names of the HW identifiers are structured hierarchically. They consist of a maximum of four hierarchical level, each of which is separated by a tilde "~". Based on the name, you can therefore recognize the "path" to the relevant hardware module.

The display of the hardware identifiers depends on the object that you have selected in the graphic view. If you select an entire device in the network view or device view, all hardware identifiers of all objects in the corresponding device are displayed in the system constants (modules, interfaces, ports, etc.). If you only select an individual module or port, only the hardware identifier of the selected object is displayed in the system constants.

Example

The system constant with the name "Local~DI_16x24VDC_HF_1" identify the HW identifier for the digital input module with the name "DI_16x24VDC_HF_1" in the local ("Local") CPU. The name of the digital module matches the editable name in the properties of the component (inspector window). If you only select this module, only the hardware identifier for this module is displayed.

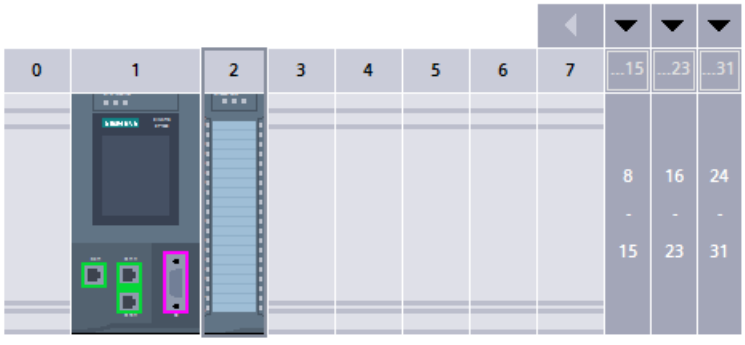

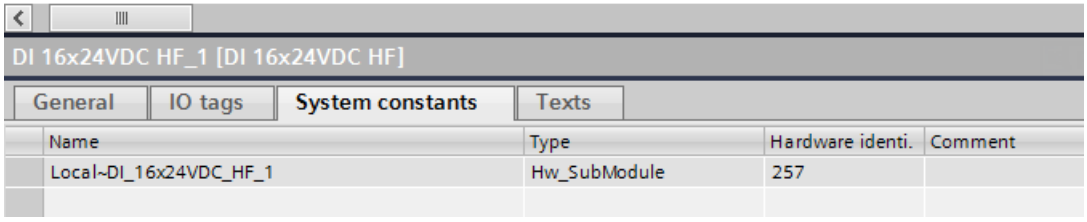

Note

To facilitate identification of HW objects, assign the most descriptive names possible for HW objects when configuring the hardware. The names should include the association of the HW object to a plant unit or device.

HW identifiers in the system constants of the PLC tag table

STEP 7 adds entries to the PLC tags ("System constants" tab) for each configured hardware component. These entries are read-only. In addition to the integer value, a name and a HW data type are also specified.

Example:

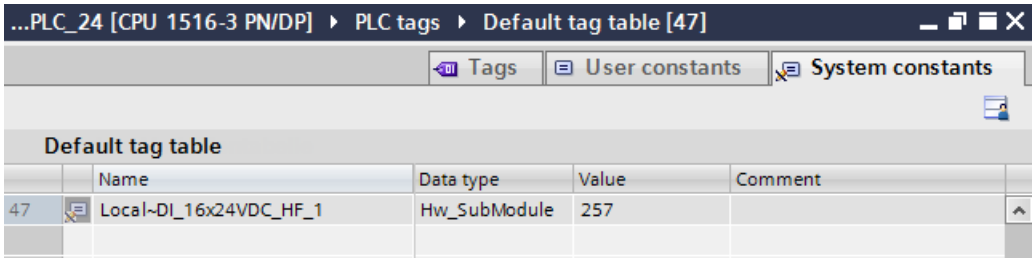

Hardware identifiers in programming

If you address a HW component in an instruction with the "ID" or "LADDR" input parameter, for example, you can double-click this input parameter to display a list of all relevant HW components for selection.

In the start information of a hardware interrupt OB, for example, you can identify the interrupt-triggering module through the hardware identifier.

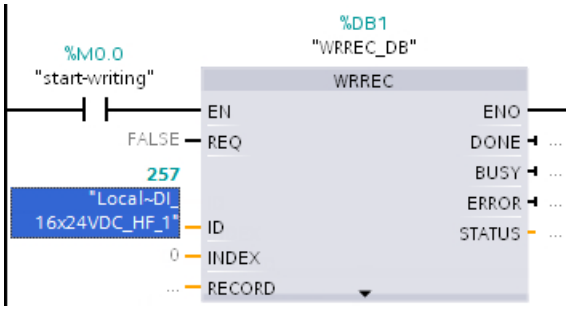

Additional information

You can find information on converting address information (IO addresses, slot, HW identifier) here.

See also

Example for the use of hardware identifiers (Page 1101)

Example for the use of hardware identifiers

You use hardware identifiers to address hardware objects such as modules, submodules or devices. The hardware identifiers have different hardware data types, depending on the instruction. The permitted hardware data type is listed for all configured components in the default tag table, "System constants" tab.

Below, you will find an explanation of the hardware identifiers that have to be used for selected instructions.

Note

Information on the respective hardware identifier

For each extended instruction that addresses hardware objects, the hardware object that has to be used is documented in the help. You access the help for the selected instruction with the <F1> key.

Recommendation

You can enter the name of the hardware object or the numerical hardware identifier of the hardware object in the Program editor.

Recommendation: Assign the most descriptive names possible for hardware objects, for example, for the modules. Read out the name and HW identifier from the "System constants" tab in the device view and enter it in the Program editor at the corresponding instruction. If you have assigned a descriptive name, you will find the name more easily in the drop-down list for selecting the HW identifier at the block parameter "ID".

Additional options for input of the HW identifier

You can drag the HW identifier from the tag table to the block parameter "ID" with drag-and-drop.

Advantage: The hardware data type is checked by the Program editor and may be rejected if an invalid hardware object was selected.

No check is possible with a numerical input.

If the permitted hardware object is an IO device, for example, select the entire IO device in the network view and read out the hardware identifier under "Properties" ("General" tab) in the inspector window. If you do not select the IO device in this case but rather the interface module in the IO device, the hardware identifier visible in the inspector window would be invalid for this instruction. You will not notice this error during programming with numerical input of the HW identifier.

Example of entering a hardware identifier

The following name of a module may be listed in the default tag table, "System constants" tab, for example:
`myModule[AI]`

1. Open the program editor
2. Open the default tag table ("PLC tags" folder)

3. Place the editor window of the programming editor and the default tag table next to each other.
   To do so, click the "Split editor space vertically" button in the toolbar.

4. Drag the "myModule[AI]" from the "System constants" tab and drop it in the "ID" or "LADDR" box of the respective instruction with drag-and-drop as seen in the figure below.

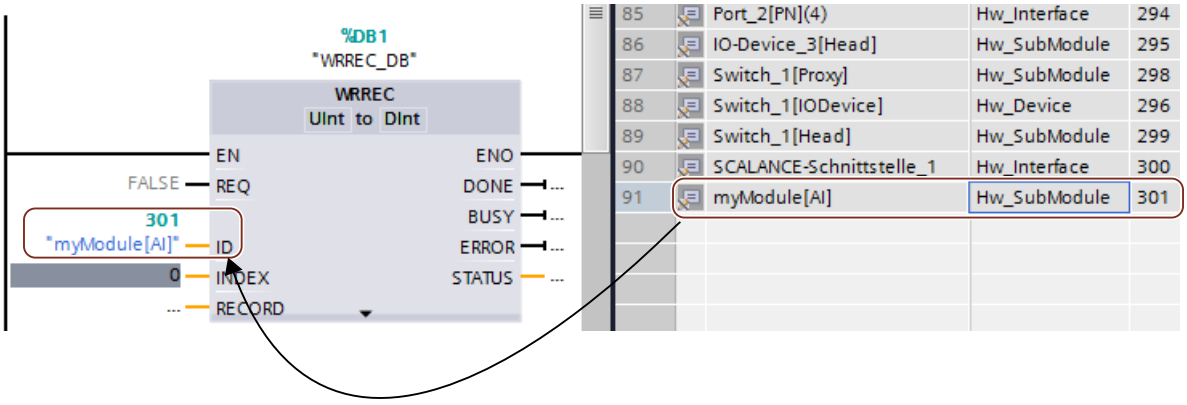

Example for addressing with hardware identifier

Extended instruction D_ACT_DP: Deactivate/activate the DP slave or PROFINET IO device.

The hardware data type is Hw_DpSlave or Hw_Device. DP slaves or IO devices are permitted hardware objects for the input parameter "LADDR".

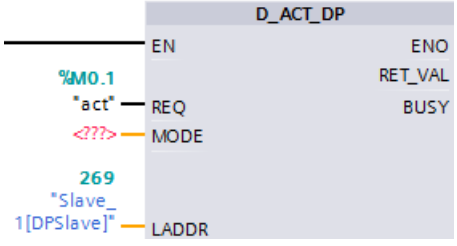

Extended instruction DPSYC_FR: Synchronize/freeze DP slaves.

The hardware data type is Hw_Interface. The DP master interface is the permitted hardware object for the input parameter "LADDR".

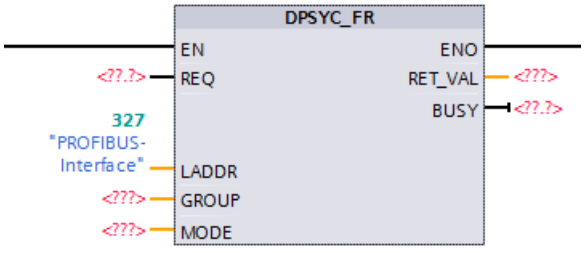

Extended instruction RDREC: Read data record.

The hardware data type is Hw_SubModule. Modules or submodules from which the data record is to be read are permitted hardware objects for the input parameter "ID".

You also use this HW data type for the extended instruction DPRD_DAT.

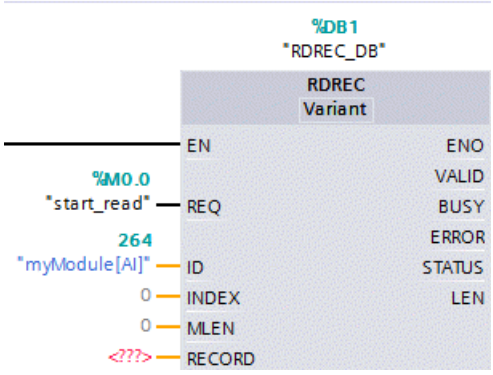

Extended instruction WRREC: Write data record. The instruction is used, for example, to write data records to a module or submodule.

The hardware data type is also Hw_SubModule. Permitted hardware objects for the input parameter "ID" are modules or submodules to which the data record is to be transferred.

You also use this HW data type for the extended instruction DPWR_DAT.

Notes:

- When dividing an I/O module into several submodules (e.g., 1 x 8 AI => 8 x 1AI), use the hardware identifier of a submodule. Do not use the hardware identifier of the module in this case.
- Use the hardware identifier of the head for interface modules of a distributed I/O system to which a data record is to be transferred for configuration control (option handling), see figure.

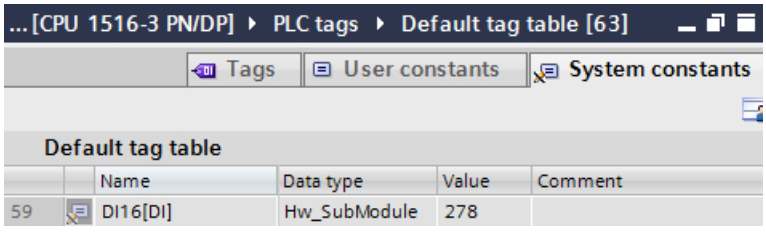

Extended instruction Station_Info: Reading information of an IO device.

The hardware data type is Hw_Device. An IO device is the permitted hardware object for the input parameter "ID".

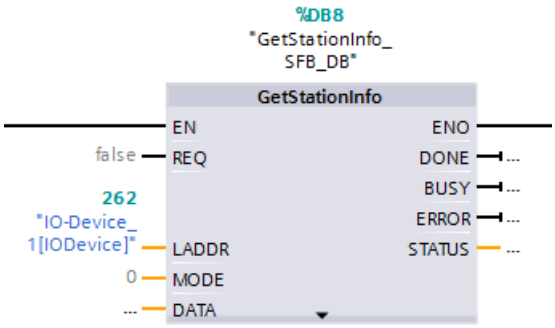

Additional information

You can find additional information on converting address information (IO addresses, slot, HW identifier) here.

See also

What you need to know about HW identifiers (Page 1099)

Input and output addresses in the address overview

Introduction

The currently used input and output addresses can be displayed in the address overview in a table form. The address overview can be found in the inspector window under "Properties" of the CPU.

Design of the address overview

With the different check boxes, you can set which objects should be displayed in the address overview:

- Inputs: Display of the input addresses
- Outputs: Display of the output addresses
- Address gaps: Display of open address spaces
- Slot: Display of the slot number

The following information is typically shown in the address overview:

Table header	Meaning
Type	Indicates whether the area is an input address area or an output address area.
Address from	Start address in the address range.
Address to	End address in the address range.
Module	Modules using the address area.

Table header	Meaning
PIP	Number of the process image partition. Shows if the cyclical OB in the process image partition.
OB	Assigned organization block
DP	Number of the master system. You can use the number to determine which slaves are assigned to a master. The value in brackets specifies the PROFIBUS address of the hardware component.
PN	Number of the IO system. The value in brackets stands for the device number of the hardware component.
Racks	Rack number on which the hardware component is inserted.
Slot	Slot number on the rack in which the hardware component is inserted.

Assign process image/process image partition

Introduction

For S7-300/400, if input addresses and output addresses are contained in a process image, they can either be in the OB1 process image (OB1-PI) or in a process image partition (PIP).

A process image offers the advantage that you can access a consistent image of process signals during cyclic program execution. For S7-300/400, you can choose between "OB1-PI" and "PIP 1" to max. "PIP 15".

Each input/output address that you have assigned to a process image partition with STEP 7 no longer belongs to the OB 1 process image input/output! Input/output addresses can be assigned only once across the OB 1 process image and all process image partitions.

You define the process image partition during address assignment (which input/output addresses of modules are managed in which process image partition). The process image partition is updated either by the user with the help of instructions or automatically by the system by connection to an OB. Exception: process image partitions of isochronous mode interrupt OBs are not updated by the system, even though they are connected to an OB (OB 61 to OB 64).

You can update each process image partition in the user program with special instructions. Use "UPDAT_PI" (SFC 26) for the process image partition of inputs and "UPDAT_PO" (SFC 27) for the process image partition of outputs.

You assign the connection of a process image partition to an OB in the properties of the CPU ("Interrupts" area).

Note

The option can only be selected for modules with inputs or outputs. In addition, the CPU must support process image partitions. The number of available process image partitions also depends on the CPU.

Requirement

- You are in the device view.
- The module is a compact DP slave or is inserted in one of the following objects:
  - Central rack
  - Expansion rack
  - DP slave
- The CPU allows free address assignment.

Procedure

To change the preset process image range, proceed as follows:

1. In the device view, select the module whose start address you want to set.
2. Click on "I/O Addresses" under "Properties" in the Inspector window.
3. Under "Process image," select the desired process image.

Note

If the entry "- - -" is visible in the "Process image" field (not changeable), this means that a process image is not available for the specified address range.

- For digital input or output modules, select a lower start address.
- If an analog module is to be addressed in a process image, then also select a lower start address. Shifting of the start address to the area of the address range for process images must be supported by the CPU.
- If the size of the process image can be specified for a CPU (via parameter assignment), then you can enlarge the process image (cycle properties of the CPU), so that the entered address is again in the process image. The parameter is called "Size of the process image...".

Assign process image/process image partition

Introduction

A process image offers the advantage that you can access a consistent image of process signals during cyclic program execution. For S7-1500, you can choose between "Update automatically" and "PIP 1" to max. "PIP 31".

Assigning addresses to the automatically updating process image area

You define the process image during address assignment (inspector window, "I/O addresses" area). The following setting should be selected if the addresses of a module are to lie in the process image area that is updated automatically by the system in each program cycle:

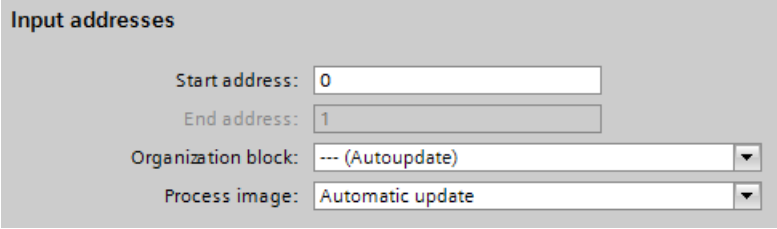

For comparison: For S7-300/400 modules, this setting is called "OB1 process image".

If the addresses should not lie in the process image area, select "None" as the process image.

Assigning an address to a process image partition

You define the process image partition during address assignment (inspector window, "I/O addresses" area). The process image partition is updated either by the user with the help of instructions or automatically by the system by connection to an OB.

Connecting a process image partition to an OB

You also connect a process image partition to an OB in the I/O addresses area ("Organization block" parameter).

For OBs of the type "Cyclic Interrupt", for example, the next free process image partition is assigned automatically; this can be changed, however. After the OB has been started, the assigned process image partition of inputs is updated by the system. At the end of the OB, the outputs of the assigned process image partition are written to the I/O outputs by the system. The process image partitions are excluded from the automatic update.

Example for the coupling of the process image partition 2 (PIP 2) to a cyclic interrupt:

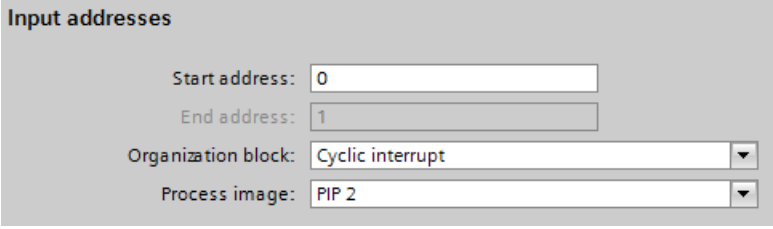

Exception: process image partitions of isochronous mode interrupt OBs are not updated by the system, even though they are connected to an OB (OB 61 to OB 64).

The assigned process image or process image partition is displayed in the properties of the assigned OB (cannot be changed).

Updating a process image partition in the user program

You can update each process image partition in the user program with special instructions. Use "UPDAT_PI" for the process image partition input and "UPDAT_PO" for the process image partition output.

Requirement

- You are in the device view.
- The module or submodule has an I/O address.

Procedure

To change the preset process image range, proceed as follows:

1. In the device view, select the module whose start address you want to set.
2. In the inspector window, click on the "I/O Addresses" area under "Properties".
3. Under "Process image," select the desired process image.

Using process image partitions in instructions

Introduction

Instructions such as "UPDAT_PI" for updating process image partition inputs and "UPDAT_PO" for outputting process image partition outputs to I/O outputs use the "PART" input parameter for inputting the process image partition.

In the following section we show you how to define a user constant for a process image partition and how to use it for inputting the process image partition in the user program. If you use different process image partitions for your I/O configuration, the symbols makes programming easier.

Requirement

You have configured the I/O, assigned addresses and process image partitions, and created the required blocks.

The programming language is set to LAD for this example.

Procedure

Follow these steps:

1. Open the tag table and enter constants of data type "Pip" (process image partition) in the "User constants" tab. For each process image partition used, assign a name and, if necessary, a comment.

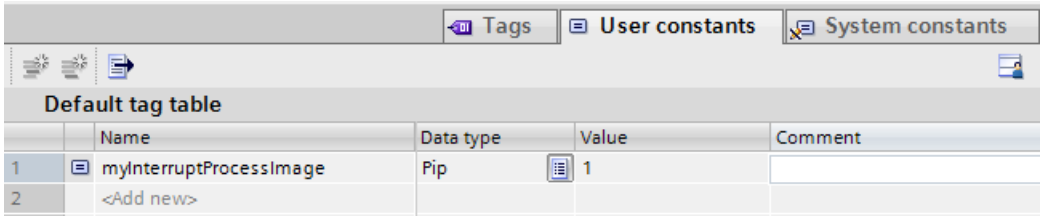

2. Open the program editor (e.g., by double-clicking an OB in the project tree).
3. Use a drag-and-drop operation to move the UPDAT_PI instruction to the network.
4. Double-click the PART input parameter to open the tag selection.
   Only process image partition constants will be listed in the tag selection.
5. Select the desired process image partition.

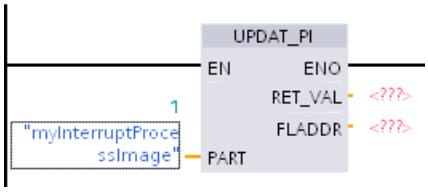

Assigning addresses to a location in the program

You can assign addresses of the I/O channels of modules directly to the points of use in the program or to a tag table.

Requirement

- The device view of the hardware and network editor is open.
- The zoom level in the device view must be set to at least 200% to allow you to see the individual I/O channels.
- The instruction window of the programming editor or a tag table is open.

Procedure

To assign I/O channels of modules to the points of use in the program or to a tag table, follow these steps:

1. In the device view, navigate to the module with the desired I/O channel.
2. Click and hold down the mouse button to drag the desired I/O address to the corresponding point of use of the block or to the tag table.

The address of the module is assigned to the point of use in the program or entered as a tag in the tag table.

Note

The tag for an input or output of a block can also be dragged to the input or output of a module in order to link the tag to the I/O channel of the module.

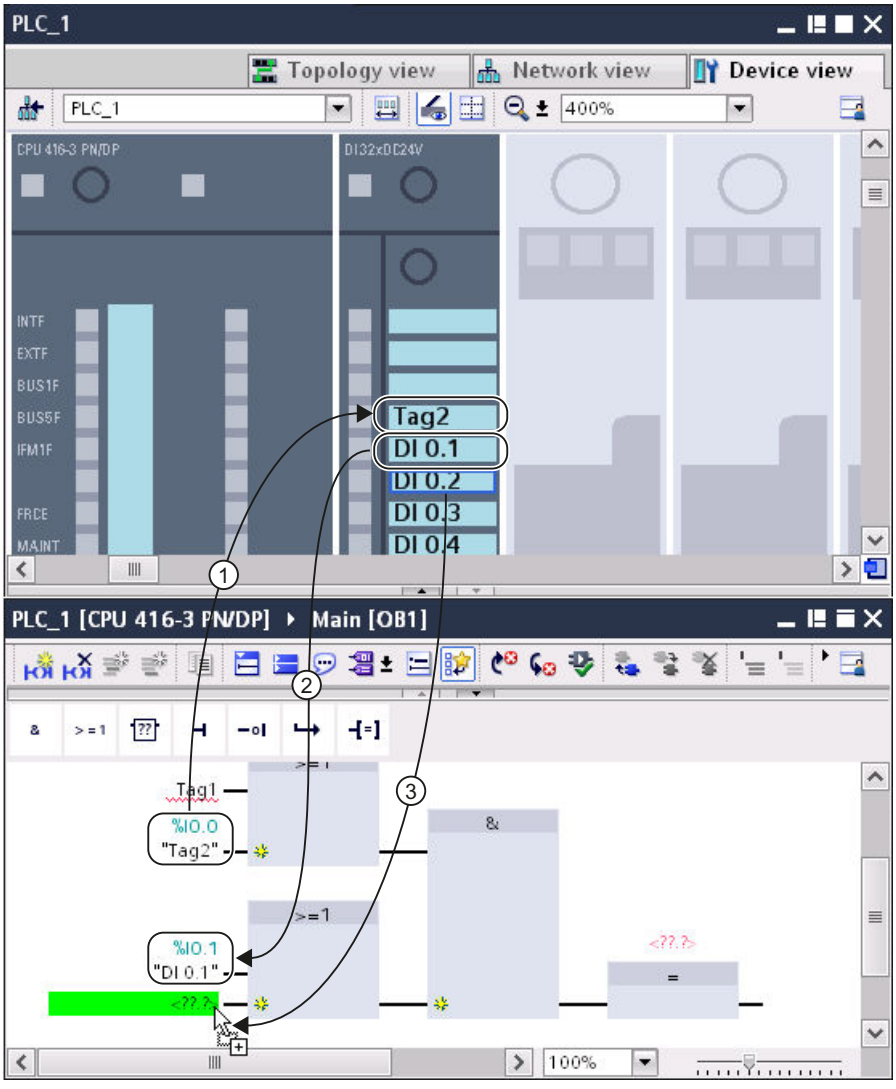

① The location of a block was dragged to an I/O channel.

② The name of an I/O channel was dragged to a point of use in the program.

③ Hold down the mouse button and drag from the name of an I/O channel to the input of a block.

1.4.1.14 Flexible automation concepts

Configuration control (option handling) for devices

Configuration control (option handling) for central S7-1500

Important information regarding configuration control (option handling)

Operating principle

You can use the configuration control to configure the structure of a control system (here: central configuration of an S7-1500) and to work with variants (options) that differ from this configuration.

- All modules that are needed in a set of similar plant units or machines are configured in a master project with a maximum configuration (station master).
- Provision is made in the user program of the master project for various station options for various plant units or machines as well as the selection of a station option. A station option uses, for example, only some of the configured modules and these modules are inserted in varying order.
- An operator selects a station option for a specific plant on-site. The operator does not have to modify the project and thus also does not have to download a modified configuration for this.

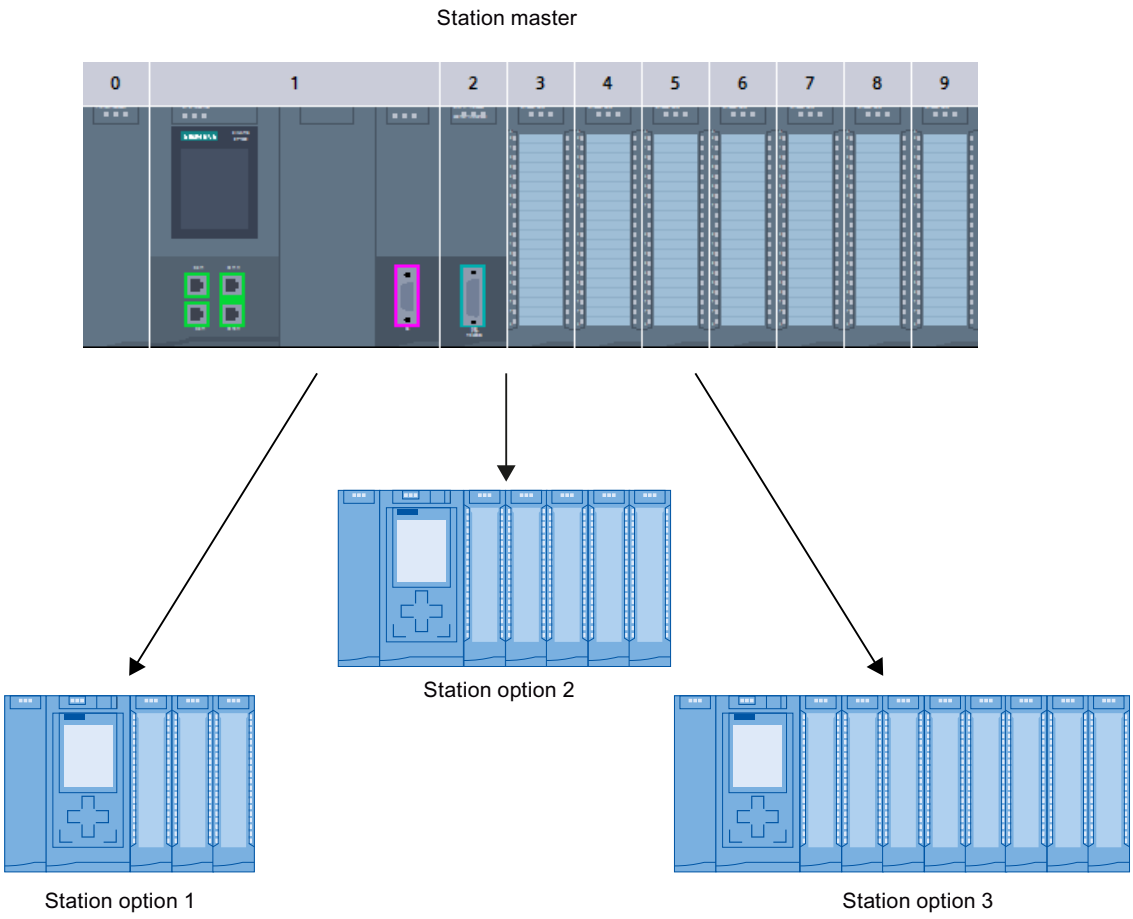

A control data record you have programmed in the startup program notifies the CPU as to which modules are missing or located in different slots as compared to the preset configuration. The configuration control has no effect on the parameter assignment of the modules.

Configuration control gives you the flexibility to vary the central installation as long as the real configuration can be derived from a preset maximum configuration.

Below you will find a description of how to activate configuration control (CPU parameter assignment) and how to structure the required data record.

Requirements

- STEP 7 Professional Version V13 or higher
- CPU S7-15XX Firmware Version 1.5 or higher Modules that support the "Configuration control" function also have the "Configuration control" entry in the description (info text) of the hardware catalog.
- Recommendation: Before you load a new program with a modified control data record, perform a reset to factory settings. This action will prevent inconsistent states that may result from the presence of incompatible control data records.
- The startup parameter "Compare preset to actual configuration" is set to "Startup CPU even if mismatch" (default setting).

Required steps

1. Enable the "Allow to reconfigure the device via the user program" parameter when configuring the CPU ("Configuration control" area).

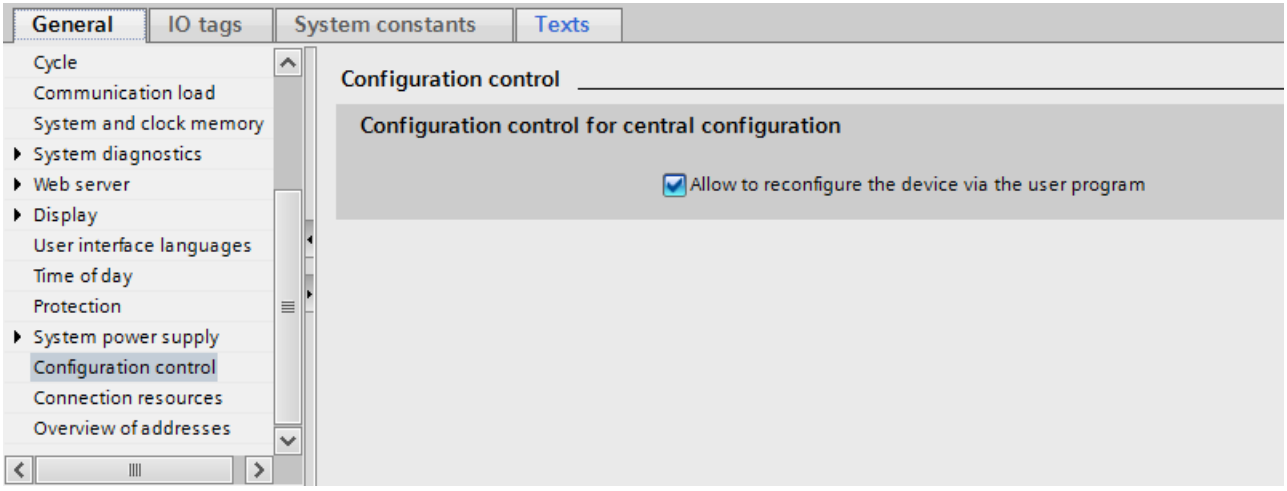

2. Create a control data record (e.g., in a data block) according to the current configuration based on the sample described below for the control data record. The control data record has the number 196. Note that you must first create a PLC data type containing the structure of the control data record and base the data block on this PLC data type.

ConfDB

	Name	Data type	Start value	...	...	...	...	Comment
1	Static							
2	ConfigControl	Struct			✓	✓		
3	Block_length	USInt	9		✓	✓		Length of control data record, including header
4	Block_ID	USInt	196		✓	✓		Data record number
5	Version	USInt	4		✓	✓		
6	Subversion	USInt	0		✓	✓		
7	slot_0	USInt	16#FF		✓	✓		configured slot_0 / assigned "real" slot
8	slot_1	USInt	16#FF		✓	✓		configured slot_1 / assigned "real" slot
9	slot_2	USInt	16#FF		✓	✓		configured slot_2 / assigned "real" slot
10	slot_3	USInt	16#FF		✓	✓		configured slot_3 / assigned "real" slot
11	slot_4	USInt	16#FF		✓	✓		configured slot_4 / assigned "real" slot

3. Transfer the control data record to the CPU in the startup program.
   The configuration control for the centrally inserted modules takes effect only after an operating mode change of the CPU from STOP to RUN. For this reason, call the extended WRREC (Write data record) instruction in the startup OB, and transfer the created control data record to the CPU; see next section.
   If a valid control data record is not transferred in the startup OB, the control is not ready for operation. The CPU returns from startup to STOP in this case.

Transferring a control data record in the startup program

The CPU processes the WRREC instruction for asynchronous transfer of the control data record. For this reason, you must call WRREC in the startup OB repeatedly in a loop until the output parameter "BUSY" or "DONE" indicate that the data record has been transferred.

Tip: Use the SCL programming language with the REPEAT ... UNTIL instruction for programming the loop.

```
REPEAT
    "WRREC_DB"(REQ := "start_config_control",
               ID := 33,
               INDEX := 196,
               LEN := "conf_LEN",
               DONE => "conf_DONE",
               BUSY => "conf_BUSY",
               ERROR => "conf_ERROR",
               RECORD := "ConfDB".ConfigControl,
               STATUS => "conf_STATUS");
UNTIL NOT "conf_BUSY"
END_REPEAT;
```

Below, you will find explanations for individual block parameters that you must supply with specific values in the configuration control context. For the remaining block parameters, see also WRREC:

Parameter	Explanation
ID	HW identifier; is always 33 (decimal) for configuration control for centrally arranged modules.
INDEX	Data record number; is always 196 (decimal) for configuration control for centrally arranged modules.
RECORD	Control data record to be transferred. See the section "Control data record" for the structure of the control data record. Tip: The "RECORD" block parameter of the WRREC instruction (V1.1 and higher) is of the "VARIANT" data type and therefore requires a tag with data type. If you store the control data record in a data block, this data block must thus have a data type. The data block created must not be of the "Global-DB" type. Rather, it must be derived from a user data type. Proceed as follows: 1. First, create a new PLC data type (user data type) with the structure of the control data record and name it, for example, "ConfDB". 2. Create a new data block. Select the newly created user data type, for example, "ConfDB", as type for this data block.

In graphical programming languages, you realize the loop using instructions for program control.

Example in FBD: Use the LABEL (jump label) and JMP (jump at RLO=1) instructions to program a loop.

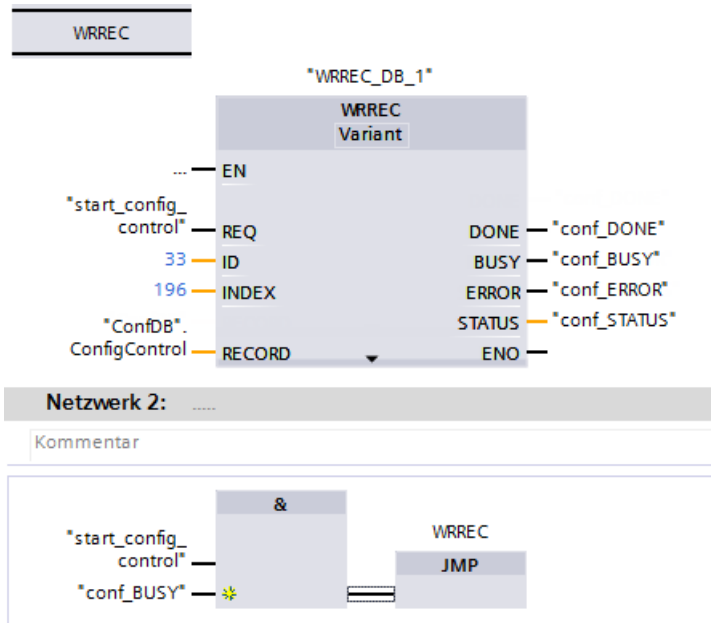

Arrangement of the modules

The following table shows the slot number assignment:

Slot	Modules	Note
0	Power supply module (optional)	Slot to the left of the CPU
1	CPU	Slot 1 is always the CPU
2 - 31	I/O modules/system power supply modules, depending on the configuration variant	Slots to the right of the CPU

Control data record

A control data record 196 containing a slot assignment is defined for configuration control.

Byte	Element	Code	Explanation
0	Block length	4 + number of slots	Header
1	Block ID	196	
2	Version	4 (for central I/O)	
3	Subversion	0	

Byte	Element	Code	Explanation
4	Assignment of configured slot 0	Real slot 0	Control element Describes in each element which real slot in the device is assigned to the configured slot. The structure of a control element is described in the following section.
5	Assignment of configured slot 1	Real slot 1 (always 1, because the CPU is always in slot 1)	
6	Assignment of configured slot 2	Real slot or 16#FF	
7	Assignment of configured slot 3	Real slot or 16#FF	
...	...	...	
4 + (max. slot number)	Assignment of configured maximum slot	Real slot or 16#FF	

Structure of a control element

A control element contains the information on which module is inserted in which slot.

The byte numbers represent the configured slots in ascending order (see above):

- Byte 4 stands for the configured slot 0
- Byte 5 stands for the configured slot 1
- Byte 6 stands for the configured slot 2
- etc.

The value that you need to enter in the corresponding byte depends on the following rule:

- If the module is present in the real configuration, enter the real slot number of the module.
  - Example 1: The module configured in slot 2 is located in slot 2.
    Enter value 2 (= actual slot) in byte 6 (= configured slot 2).
  - Example 2: The module configured in slot 3 is located in slot 2.
    Enter value 2 (= actual slot) in byte 7 (= configured slot 3).
- If the module does not exist in the real structure, enter 16#FF (255) in the byte for the configured slot.

Rules

Observe the following rules:

- If you have enabled configuration control, the CPU is not ready for operation without a control data record. The CPU returns from startup to STOP if a valid control data record is not transferred in the startup OB. The central I/O is not initialized in this case. The cause for the STOP mode is entered in the diagnostics buffer.
- For addressing the WRREC instruction, use the HW identifier 33 (decimal, for the ID block parameter) to write the control data record.
- The control data record is saved retentively in the CPU, which means that it is not necessary to write the control data record 196 again at a restart if the configuration is unchanged. Prior to commissioning, we recommend that a memory reset be performed for the CPU in order to delete any control data record that is present.

- Slot entries in the control data record outside the configured preset configuration are ignored by the CPU.
- You can shorten the control data record. It must contain the entries up to the last slot of the current preset configuration.
- Each real slot may only be present once in the control data record.
- A real slot may only be assigned to one configured slot.
- Using CPs/CMs:
  - As regards the configuration control, point-to-point CPs/CMs behave like digital and analog modules (no restrictions).
  - CPUs with firmware version V1.7 or higher:
    If the central configuration includes CPs/CMs, for example a CM 1542-5 (DP master or DP slave), these CMs/CPs are not affected by the configuration control. Therefore, these modules must remain in their configured slots and their configured slot numbers entered in the control data record ("real slot = configured slot") Maximum flexibility is achieved by inserting CMs/CPs to the immediate right of the CPU.
  - CPUs with firmware version V1.6 or higher:
    As a matter of principal, CMs and CPs cannot be used for configuration control.
- System power supply modules (PS) can also be subject to configuration control. Note the information on this topic in the S7-1500 system manual (http://support.automation.siemens.com/WW/view/en/59191792). In particular for a system power supply module (PS) on slot 0, we recommended that you avoid reconfiguration.

Note

Modified configuration

The writing of a control data record with a modified configuration leads automatically to a restart with this modified configuration.

As a result of this reaction, the retentively saved original data record 196 is deleted and the new data record 196 is saved retentively.

Behavior during operation

Effect of the discrepancy between the configured configuration and real configuration:

- For the online display and for the display in the diagnostics buffer (module OK or module faulty), the hardware configuration is always used and not the differing real configuration. Example: A module supplies diagnostic information. This module is configured in slot 4, but is really inserted in slot 3 (missing module; see example in the next section). In the online view, a configured slot 4 is indicated as faulty. In the real configuration, the module in slot 3 indicates an error via an LED display.

If modules are entered as missing in the control data record, the automation system behaves as follows:

- Modules designated as not present in the control data record do not supply diagnostics and their status is always OK. The value status is OK
- Direct write access to the outputs or write access to the process image of outputs that are not present: Remains without effect; no access error is signaled.

- Direct read access to the inputs or read access to the process image of inputs that are not present: Value "0" is supplied; no access error is signaled.
- Write data record to module that is not present: Remains without effect; no error is signaled.
- Read data record from module that is not present: An error is signaled because a valid data record cannot be returned.

Error messages

The following error messages are returned if an error occurs during writing of the control data record:

Table 1-85 Error messages

Error code	Meaning
16#80B1	Invalid length; the length information in data record 196 is not correct.
16#80B5	Configuration control parameters not assigned.
16#80E2	Data record was transferred in the wrong OB context. The data record must be transferred in the startup program.
16#80B8	Parameter error; module signals invalid parameters.

Additional information and examples

You will find a library of the data records and application examples here: Templates for the data records (http://support.automation.siemens.com/WW/view/en/29430270)

See also

S7-1500 system manual (http://support.automation.siemens.com/WW/view/en/59191792)

Example of a configuration control

A configuration consisting of a system power supply, CPU, and 3 I/O modules is configured in the following section.

The module at slot 3 is not present in the first configuration expansion and is "hidden" by the configuration control.

In the second configuration expansion, the module that was initially hidden is located in the last slot. The added slot is made known to the CPU by a modified control data record.

Actual configuration with missing module

The specified configuration contains all modules that can be present in a final expansion stage.

The module that is inserted in slot 3 in the specified configuration is missing in the real expanded configuration. For this reason, slot 3 must be coded in the control data record accordingly with "FF_H" (= not present).

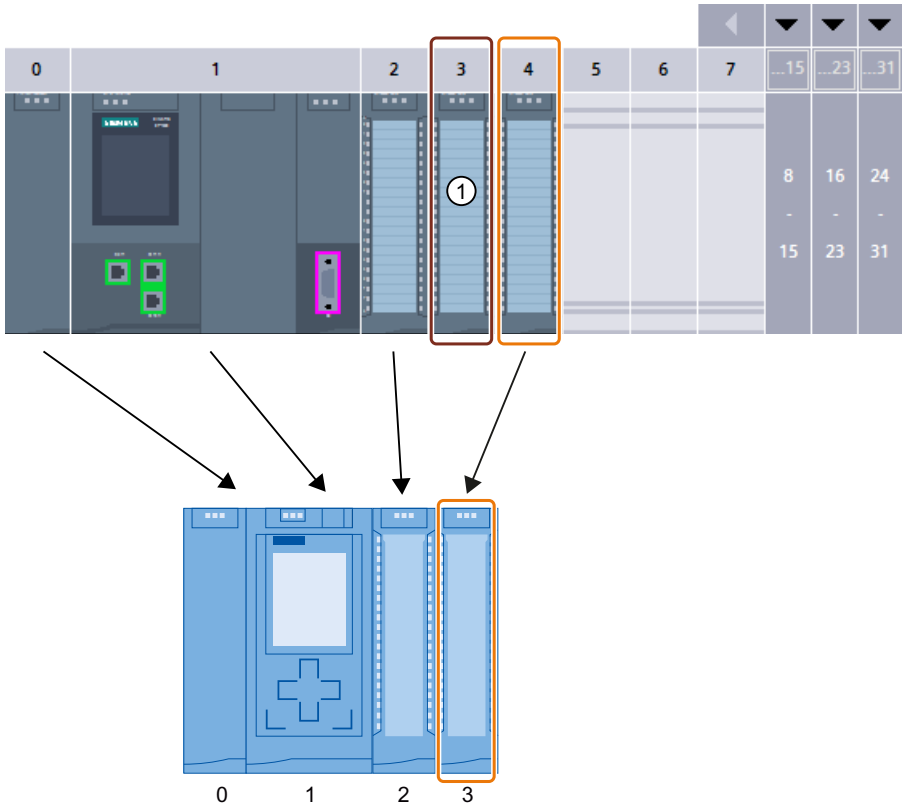

ConfDB

		Name	Data type	Start value	...	...	...	...	Comment
1		Static							
2		ConfigControl	Struct			✓	✓		
3		Block_length	USInt	9		✓	✓		4+number of slots
4		Block_ID	USInt	196		✓	✓		
5		Version	USInt	4		✓	✓		
6		Subversion	USInt	0		✓	✓		
7		slot_0	USInt	0		✓	✓		configured slot_0 / assigned "real" slot 0
8		slot_1	USInt	1		✓	✓		configured slot_1 / assigned "real" slot 1
9		slot_2	USInt	2		✓	✓		configured slot_2 / assigned "real" slot 2
10		slot_3	USInt	16#FF		✓	✓		configured slot_3 / not used ①
11		slot_4	USInt	3		✓	✓		configured slot_4 / assigned "real" slot 3

① Module is missing in the real configuration

Actual configuration with subsequently added module

The module present in slot 3 in the specified configuration is added to the back of the real configuration by inserting it as the last module in slot 4.

The control data record is adapted accordingly.

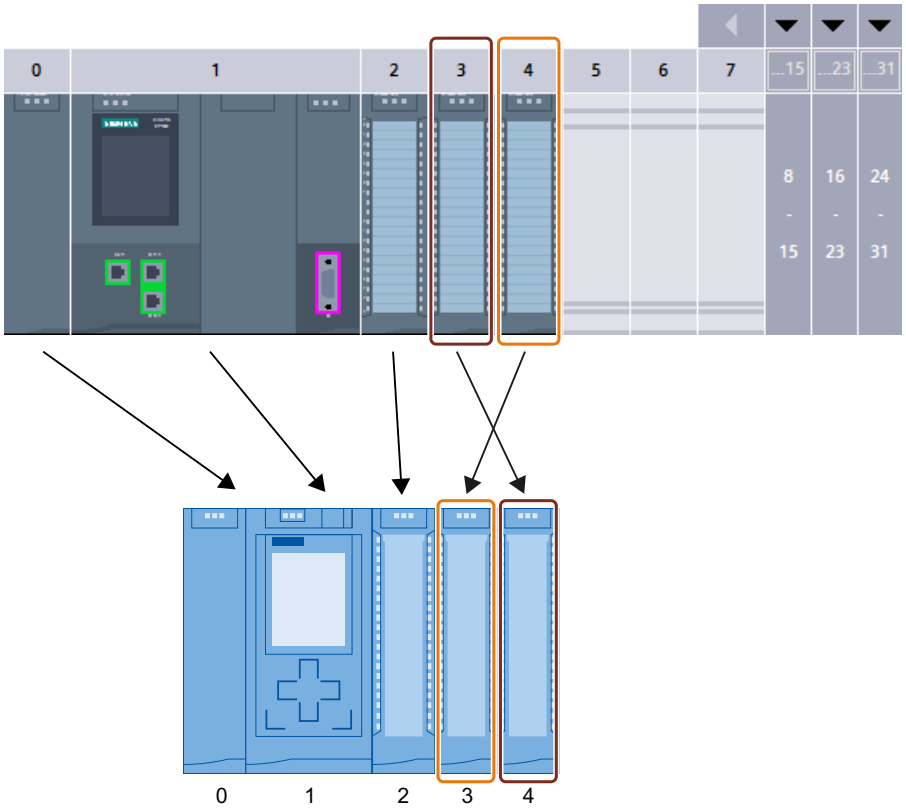

ConfDB

		Name	Data type	Start value	...	...	...	...	Comment
1		Static							
2		ConfigControl	Struct			✓	✓		
3		Block_length	USInt	9		✓	✓		4+number of slots
4		Block_ID	USInt	196		✓	✓		
5		Version	USInt	4		✓	✓		
6		Subversion	USInt	0		✓	✓		
7		slot_0	USInt	0		✓	✓		configured slot_0 / assigned "real" slot 0
8		slot_1	USInt	1		✓	✓		configured slot_1 / assigned "real" slot 1
9		slot_2	USInt	2		✓	✓		configured slot_2 / assigned "real" slot 2
10		slot_3	USInt	4		✓	✓		configured slot_3 / assigned "real" slot 4
11		slot_4	USInt	3		✓	✓		configured slot_4 / assigned "real" slot 3

Control data record for an ET 200SP CPU

Slot assignment

The following table shows the possible slots for the various modules for an ET 200SP CPU:

Table 1-86 Slot assignment

Modules	Possible slots	Note
CPU	1	Slot 1 is always the CPU
I/O modules	2 - 65	Downstream of CPU
Server module	2 - 66	The server module completes the configuration of the ET 200SP station after the CPU/the last I/O module.

Control data record

For the configuration control of an ET 200SP CPU, you define a control data record 196 V2.0, which includes a slot assignment. The maximum slot corresponds to the slot of the server module.

The table below shows the structure of a control data record with explanations of the individual elements.

Table 1-87 Configuration control: Structure of control data record 196

Byte	Element	Code	Explanation
0	Block length	4 + (number of slots × 2)	Header
1	Block ID	196	
2	Version	2	
3	Version	0	

Byte	Element	Code	Explanation
4	Slot 1 of the station master	Slot assignment 1 in the station option (always 1, because the CPU is always in slot 1)	**Control element** Contains the information on which module is inserted in which slot. The value that you need to enter in the corresponding byte depends on the following rule: • If the module exists in the station option, enter the slot number of the module. • If the module exists as empty slot (with BU cover), enter the slot number of the module + 128. (Example: module as empty slot on slot 3: Enter 131 in the control element) • If the module does not exist in the station option, enter 0. **Additional function** Contains information on whether a new potential group will be opened in the station option - by replacing a dark-colored BaseUnit with a light-colored BaseUnit. • If you replace a dark-colored BaseUnit with a light-colored BaseUnit, enter 1 as additional function. • If you accept the BaseUnit from the station master, enter 0 as additional function.
5	Additional function for slot 1		
6	Slot 2 of the station master	Slot assignment in the station option	
7	Additional function for slot 2		
8	Slot 3 of the station master	Slot assignment in the station option	
9	Additional function for slot 3		
:	:	:	
4 + ((max. slot - 1) × 2)	Server module slot	Server module slot assignment in the station option*	
4 + ((max. slot - 1)× 2) + 1	Additional function for server module slot		

* The server module must be present in the station option and must not be marked as empty slot (BU cover).

Block library "OH_S71x00_Library"

The block library OH_S71x00_Library (https://support.industry.siemens.com/cs/#document/29430270?lc=en-WW) is available for download from the Internet. The block library contains data types with the structure of the control data records for the ET 200SP distributed I/O system. You can implement your flexible automation solution economically with the help of these data types.

See also

ET 200SP system manual (https://support.industry.siemens.com/cs/ww/en/view/58649293)

Configuration control (option handling) for IO systems

Using standard machine projects

Standard machine projects can be a useful solution, for example, when:

- The automation solution (machine) is used multiple times by a customer.
- The machine is used in different plants by various customers.

Variability in the configuration through configuration control of IO devices

You obtain additional flexibility for a system or machine by configuring optional IO devices and by configuring ports that are prepared for interconnection by the user program.

Optional IO devices can be added to the configuration via the user program without having to download the project again. In addition, the interconnection of the ports of IO devices and thus the order of the IO devices can be adapted via the user program, enabling control of the configuration required in each case - without an engineering system!

Advantages of standard machine projects

- A centrally maintained project (configuration and program) can be loaded to various machines of the same type without changes.
- Only a few easy adaptations are needed locally to integrate the machine into an existing network infrastructure.
- Modular machines can be easily adapted for various applications by using the configuration control to notify the PROFINET IO system of the current real configuration of the IO system.

Additional information

For detailed information on standard machine projects, refer to the PROFINET with STEP 7 (http://support.automation.siemens.com/WW/view/en/49948856) manual here (Page 2982).

1.4.1.15 Using OPC UA communication

What you need to know about OPC UA

OPC UA and Industrie 4.0

Uniform standard for information and data exchange

Industry 4.0 stands for the intensive utilization, evaluation and analysis of the large volumes of data from production in IT systems at the enterprise level. With Industry 4.0, data exchange between the production and enterprise levels is rapidly increasing. However, a prerequisite for success is a uniform standard for the information and data exchange.

The OPC UA (OPC Unified Architecture) standard is particularly suitable for data exchange across different levels thanks to its independence from specific operating systems, its secure transfer procedures and the semantic description of data. OPC UA makes available not only data but also information about the data (data types), at the same time making possible machine-interpretable access to the data.

OPC UA for S7-1500 CPUs

In OPC UA, one system operates as a server and provides data the existing information to other systems (clients).

OPC UA clients, for example, have read and write access to data on an OPC UA server. OPC UA clients call methods on the OPC UA server.

You can access this data online with a client, including e.g. information on performance and diagnostics. In OPC UA terminology, this function is called "Browsen". The "Subscription" function saves the regular reading of a tag - the client only allows values to be sent from the server when a change has occurred.

A system can be both a client and a server.

OPC UA server of the S7-1500 CPU

As of firmware version 2.0, an S7-1500 CPU is equipped with an OPC UA server.

The following sections describe how you configure the OPC UA server of the S7-1500 CPU to make data and methods available for OPC UA clients so that clients have read or write access to PLC tags on the CPU and can call server methods.

The following sections also set out how to integrate companion specifications into the address space of the OPC UA server.

OPC UA client of the S7-1500 CPU

As of firmware version V2.6, an S7-1500 CPU is additionally equipped with an OPC UA client.

The following sections shows how to use standardized instructions (PLCopen function blocks) to create a user program that, as an OPC UA client, reads data from an OPC UA server or writes data to an OPC UA server or calls methods from an OPC UA server.

STEP 7 (TIA Portal) assists you in creating user programs by providing an editor for client interfaces and a parameter assignment for OPC UA connections:

The OPC UA instructions for an S7-1500 CPU as client are described in detail in the help to the instructions (Instructions > Communication > OPC UA).

OPC UA client for test purposes

The following description uses various different OPC UA clients to illustrate the use of OPC UA clients:

- "UaExpert" of Unified Automation. An extensive client that can be used free of charge: Link for downloading UaExpert (https://www.unified-automation.com/downloads/opc-ua-clients.html)
- "UA Sample Client" of the OPC Foundation. This client is available free of charge for users who are registered with the OPC Foundation :
  Link for downloading the example client of the OPC Foundation (https://opcfoundation.org)

Application example in Industry online support

Siemens Industry Online Support provides an application example with a client API free of charge. .NET developers can use the functions of this interface to access the OPC UA server of an S7-1500. The client API is based on the .NET OPC UA stack of the OPC Foundation.

The application example shows how to establish connections between servers and clients, for example. It also demonstrates the reading and writing of PLC tags.

Link to download: OPC UA .NET client for the SIMATIC S7-1500 OPC UA Server (http://support.automation.siemens.com/WW/view/en/109737901)

General features of OPC UA

The key features of OPC UA

- OPC UA does not depend on a specific operating system platform.
  OPC UA can, for example, be used with Windows, Linux, Mac OS X, a real-time operating system or a mobile operating system (such as Android).
- OPC UA is implemented in various different programming languages.
  The OPC Foundation has implemented the OPC UA standard in several programming languages: Stacks for ANSI C, .NET and Java are available.
- The OPC Foundation offers the Java stack and .NET stack as well as example programs as open source software. See Github (https://github.com/opcfoundation).
- A number of companies offer Software Development Kits (SDK) containing the OPC Foundation stacks and other functions to facilitate the development of solutions.
  Advantages of using SDKs:
  - Support from the supplier
  - Tested software
  - Detailed documentation
  - Clear license conditions (important for selling of solutions)
- Scalability
  OPC UA can be used in sensors, embedded systems, controllers, PC systems and smartphones, as well as in servers running MES and ERP applications.
  OPC UA and PROFINET can also be used together. The two protocols use the same network infrastructure.

- Simple client-server principle
  An OPC UA server provides information within a network and an OPC UA client retrieves this information.
- Integrated security mechanisms
  OPC UA uses security mechanisms at various levels:
  - A secure connection can only be established between an OPC UA server and an OPC UA client if the client and server can register with X.509-v3 certificates and accept each other's certificates (security at the application level). Various security policies are possible, including a non-secure connection between server and client (Security Policy: "No security").
  - Before allowing access, a server can request the following information from the user:
    - A certificate (cannot be configured in STEP 7)
    - User name and password
    - No user authorization

  The security mechanisms are optional and configurable.
- Independence of a specific transport layer
  The following transport mechanisms are currently supported by OPC UA:
  - The transfer of messages as a binary stream directly via TCP/IP
  - The transfer of messages with XML over TCP/IP and HTTP. This transport mechanism allows only a slow transfer and is therefore almost never used. The S7-1500 CPUs do not support this transport mechanism.

  Binary data exchange is supported by all OPC UA applications (required in OPC UA specification).
- PLC tag mapping
  The information of the OPC UA server (for example the PLC tags) is modeled as nodes connected to one another via references. This makes it possible to browse from node to node with an OPC UA client and find out what content can be read, monitored or written.
- Information
  OPC UA servers provide a lot of information, for example about the CPU, the OPC UA server itself, the data and the data types.
- Instance concept
  OPC UA is based on a type instance concept. Both instances and their type definitions are available in runtime.
- Differentiation of the functionality with profiles: S7-1500 supports the Embedded UA Server Profile, for example

See also

Unified Automation (https://www.unified-automation.com)

OPC Foundation (https://opcfoundation.org)

From the Classic OPC interface to OPC UA

Standardized interface

Classic OPC only runs on Windows operating systems.

To get around this restriction, the OPC Foundation developed the OPC UA standard.

The standard is not platform-specific and uses an optimized, TCP-based binary protocol for high-performance applications.

Different systems are therefore able to exchange data, for example:

- Controllers with MES and ERP systems
- Siemens controllers with controllers from other manufacturers
- Smartphones with controllers
- Embedded systems with controllers
- Smart sensors with controllers

Addressing nodes

Nodes in the OPC UA address space are uniquely identified by a NodeId (Node ID or Node Identifier).

The NodeId consists of an identifier, identifier type and a namespace index. Namespaces are used to avoid naming conflicts. The OPC Foundation has defined a wide range of nodes that provide information about the given OPC UA server. These nodes can be found in the namespace of the OPC Foundation and have the index 0.

The OPC Foundation also defines data types and tag types.

Namespace

All the tags or methods of an S7-1500 are contained in the namespace (Namespace) of the standard server interface "http://www.siemens.com/simatic-s7-opcua". By default this namespace has the Index 3. The index may change later if additional namespaces are inserted into the server or if existing ones are deleted. The current index of the namespace therefore needs to be requested from the server before values are read or written.

The following figure shows an example of the result of such a request.

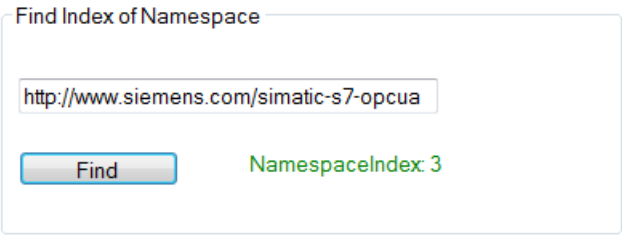

Identifier

The Identifier corresponds to the name of the PLC tag in quotation marks. The quotation mark is the only sign that is not permitted as part of a name in STEP 7. Quotation marks avoid naming conflicts.

The following example reads the value of the "StartTimer" tag:

The Identifier can consist of several components. The individual components are then separated by a dot. The following example reads the "MyDB" array data block completely. This data block contains an array with ten integer values. All ten values should be read in one pass. Therefore, "0:9" is entered at the array range.

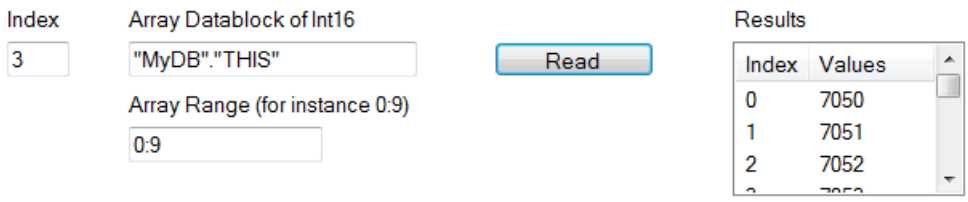

PLC tags in the address space of the OPC UA server

The figure below shows where the PLC tags in the example are located in the address space of the OPC UA server (excerpt from UA client):

The "MyDB" data block is a global data block. The data block is therefore located below the node "DataBlocksGlobal". "StartTimer" is a memory tag and is therefore stored below the "Memory" node.

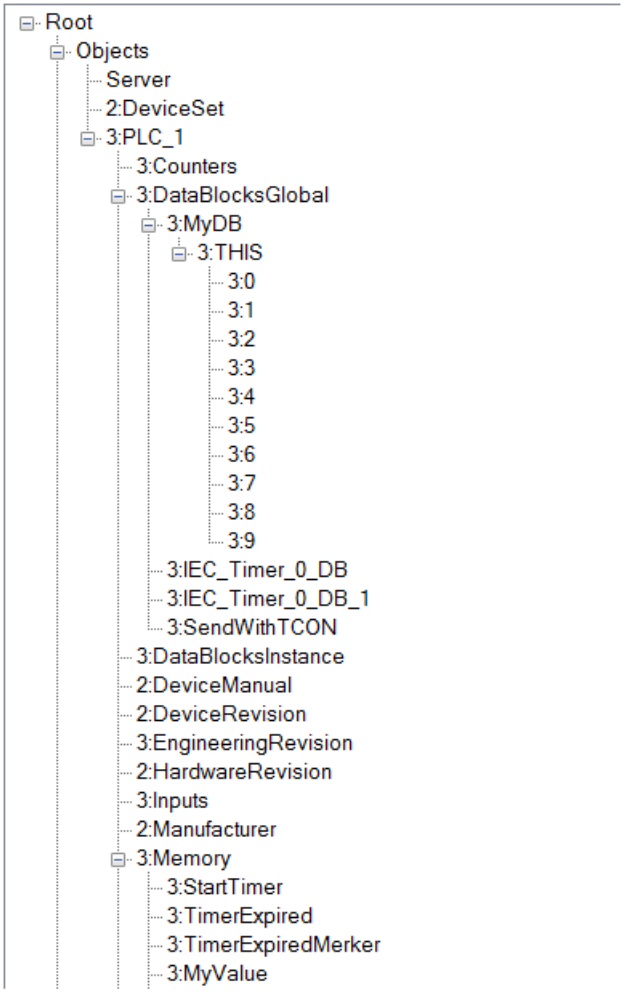
Root
Objects
Server
2:DeviceSet
3:PLC_1
3:Counters
3:DataBlocksGlobal
3:MyDB
3:THIS
3:0
3:1
3:2
3:3
3:4
3:5
3:6
3:7
3:8
3:9
3:IEC_Timer_0_DB
3:IEC_Timer_0_DB_1
3:SendWithTCON
3:DataBlocksInstance
2:DeviceManual
2:DeviceRevision
3:EngineeringRevision
2:HardwareRevision
3:Inputs
2:Manufacturer
3:Memory
3:StartTimer
3:TimerExpired
3:TimerExpiredMerker
3:MyValue

Methods in the address space of the OPC UA server

If you implement a method via your user program, this takes the following form in the address space of the OPC UA Server (see AUTOHOTSPOT):

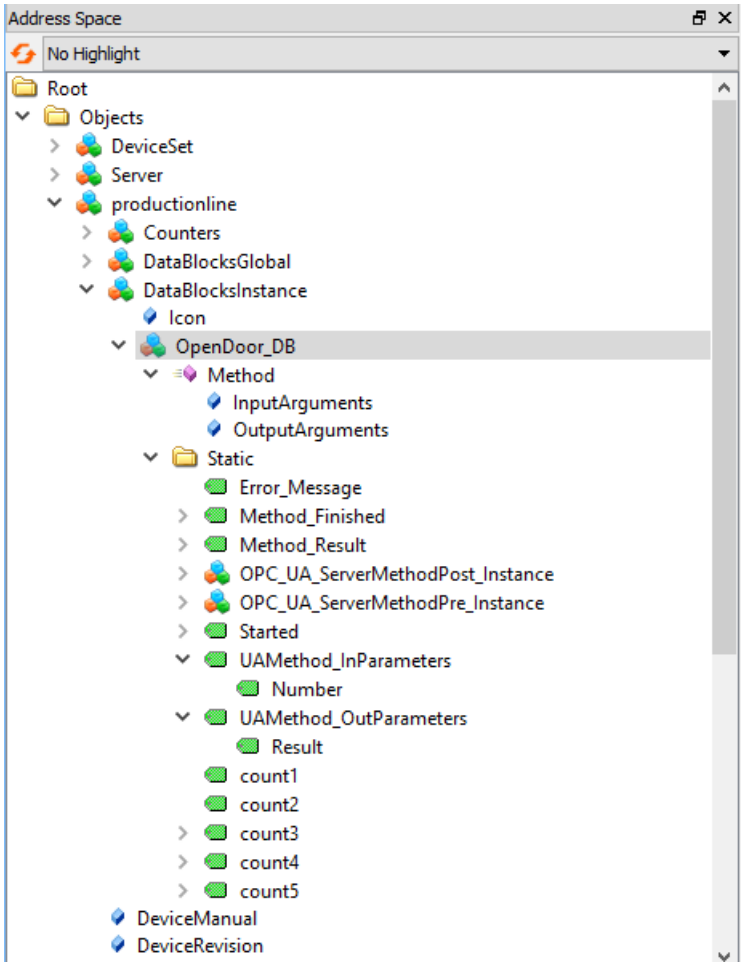

See also

S7-1500 Communication function manual (https://support.industry.siemens.com/cs/ww/en/view/59192925)

Mapping of data types

SIMATIC and OPC UA data types

SIMATIC data types do not always correspond with OPC UA data types.

S7-1500 CPUs provide SIMATIC tags (with SIMATIC data types) to their own OPC UA server as OPC UA data types so that OPC UA clients can access these tags over the server interface with OPC UA data types.

A client can read the attribute "DataType" from such a tag and reconstruct the original data type in SIMATIC.

Example

A tag has the SIMATIC data type "COUNTER". You read COUNTER → UInt16 in the table. You now know that you do not need to convert; the COUNTER value is sent over the line as a UInt16 data type.

The client detects from the attribute "DataType" that the tag is actually the SIMATIC data type "COUNTER". With this knowledge, the client reconstructs the data type.

Table 1-88 SIMATIC and OPC UA data types

SIMATIC data type	OPC UA data type
BOOL	Boolean
BYTE	BYTE → Byte
WORD	WORD → UInt16
DWORD	DWORD → UInt32
LWORD	LWORD → UInt64
SINT	SByte
INT	Int16
DINT	Int32
LINT	Int64
USINT	Byte
UINT	UInt16
UDINT	UInt32
ULINT	UInt64
REAL	Float
LREAL	Double
S5TIME	S5TIME → UInt16
TIME	TIME → Int32
LTIME	LTIME → Int64
DATE	DATE → UInt16
TIME_OF_DAY (TOD)	TOD → UInt32
LTIME_OF_DAY (LTOD)	LTOD → UInt64

SIMATIC data type	OPC UA data type
DATE_AND_TIME (DT)	DT → Byte[8]
LDT	DateTime
DTL Special note: You can only describe the structure completely with an OPC UA client. You have read-only access individual elements of this structure (e.g. "YEAR")	mapped as structure
CHAR	CHAR → Byte
WCHAR	WCHAR → UInt16
STRING (Code page 1252 or Windows-1252)	STRING → String
WSTRING (UCS-2; Universal Coded Character Set)	String
TIMER	TIMER → UInt16
COUNTER	COUNTER → UInt16

Arrays

A read or write job with OPC UA is always an array access, which means that it always has an index and length. A single tag is a special case of an array (index 0 and length 1). The data type is simply sent repeatedly on the line. For the tags, the "DataType" attribute indicates the basic data type. The attributes "ValueRank" and "ArrayDimensions" show whether or not you are dealing with an array and how large the array is.

Data types based on arrays

There are SIMATIC data types for which an OPC UA value is mapped to an array of bytes. An array of these data types is then mapped to a two-dimensional array.

Example: The SIMATIC data type DATE_AND_TIME (DT) is mapped on the OPC UA side to an 8-byte array (Byte[8]), see table above. When you define an array of the SIMATIC data type DATE_AND_TIME (DT), then it is considered as two-dimensional array.

This fact affects the use of system data types such as OPC_UA_NodeAdditionalInfo and OPC_UA_NodeAdditionalInfoExt:

For the data types described above, you must use the system data type OPC_UA_NodeAdditionalInfoExt for multidimensional arrays instead of OPC_UA_NodeAdditionalInfo.

Structures

Structures are transferred as ExtensionObject. The S7-1500 server uses binary representation for transmission of the ExtensionObjects over the line; the individual structure elements come one after the other. At the front is the NodeId of the data type; this is used by the client to establish the structure.

For the OPC UA specification <= V1.03, a client must read, decode and interpret the complete DataTypeDictionary for this purpose (unless it has already done so offline with an XML import).

Additional information

More details on mapping of basic data types, arrays and structures can be found in the OPC UA Specification Part 6, "Mappings" (see OPC UA BINARY there).

What you need to know about OPC UA clients

Basics of OPC UA clients

OPC UA clients are programs that do the following:

- Access the information from an OPC UA server (for example an S7-1500 CPU): read, write, subscriptions
- Execute methods through the OPC UA server

However, OPC US clients can only access data that is enabled for this purpose (see "Managing write and read rights (Page 1156)").

You need the endpoint of the server to establish a connection to an OPC UA server (see "Endpoints of the OPC UA servers (Page 1150)").

Reading out information from the OPC UA server

When a connection to an end point of the server exists, you can use the navigation function of the client: You navigate starting from a defined starting point (from the "root" node) through the address space of the server.

The following information is provided in the process:

- Enabled PLC tags, data blocks and data block components
- Namespace index and identifiers of these PLC tags, data blocks and DB components
- Data types of the PLC tags and DB components
- Number of components in arrays (required for reading and writing arrays)

In addition, you receive information about the OPC UA server itself as well as information about the S7-1500 based on the "OPC UA for Devices" standard of the OPC Foundation (for example, serial number, firmware version).

Reading data from the server and writing to the server

You now know the namespace, identifier and data type of PLC tags. This means that you can now specifically read individual PLC tags and DB components as well as complete arrays and structures. Examples for reading Boolean tags and array data blocks are available under Addressing nodes (Page 1128).

Rules for access to structures are available here (Page 1229).

With the information that you obtain while navigating through the address space of the server (index, identifier and data type), you can also transfer values to the S7-1500 with the OPC UA client. The following example overwrites the first three values in the array data block "MyDB".

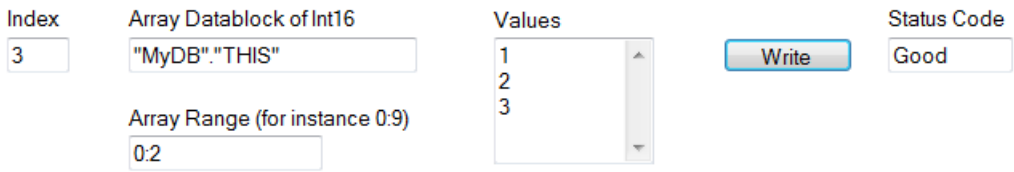

For "Array Range" you specify which components of the array you want to overwrite. The "Good" status code indicates that the values were transferred successfully. However, you can only write the values to the S7-1500 but not the time stamps of these values. The time stamps can only be read.

Faster access through registration

The examples up to now use strings as Identifier, for example, "MyBD2"."THIS". If, however, an Identifier is used as a numeric Node ID instead of a String Node ID, access is much faster. If you access specific tags regularly, you should use the functions "RegisteredRead" and "RegisteredWrite":

In this case, your client first logs in the PLC tag on the server., The server then sends back an Identifier that the client uses for the actual access. This Identifier applies solely to the current session and has to be queried again when the session connection is terminated/lost.

In the following example, the "StartTimer" tag is first registered on the server. Afterwards, the rapid function "RegisteredWrite" is used for setting the value.

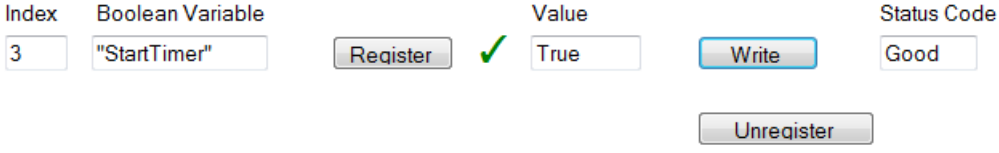

In accordance with the same scheme, the "RegisteredRead" function can also be used, which is particularly useful for recurring data readouts. Take into account, however, that depending on the application it may be advisable to use a Subscription instead.

Recommendation: It is best to place registrations in the startup program of the OPC UA client, since the registration takes up time.

Please note that you can set the maximum number of registered nodes in the properties of the S7-1500 CPU and that the Clients have to respect this number, see General settings of the OPC UA server (Page 1164).

Subscription

The term "Subscription" is used for a function in which only those tags for which an OPC UA client has registered at the OPC UA server are transferred. The OPC UA server only sends a message to the OPC UA client for these registered tags (Subscriptions) when a value has changed. The monitoring of these tags makes constant sampling by the OPC UA client (Polling)superfluous, which reduces the network load.

You have to create a Subscription to use this function. For this purpose, you specify the "Publishing Interval" at the UA client and click the "Create" button. The publishing interval is the time interval in which the server sends new values to the client in a notification (data change notification).

In the following example a subscription has been created: The client receives a message with the new values (publishing interval 50 ms) every 50 milliseconds here.

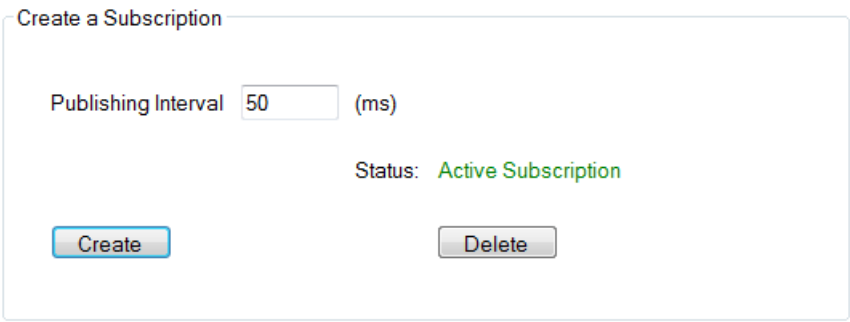

Preventing server overload

You can set the OPC UA server of the S7-1500 CPU by means of the "Minimum publishing interval" so that it does not serve extremely short send intervals requested by the client, see Settings of the server for subscriptions (Page 1165).

Example: A client wants to be operated at a publishing interval of 50 ms as detailed above. Such a short publishing interval would, however, result in a high network and server load. You should therefore set 1000 ms as the "Minimum publishing interval" for the server. Clients whose subscription requires shorter publishing intervals are "slowed" to 1000 ms and the server is protected from overload.

Sampling and transmission (Sampling & Publishing) within the scope of a subscription are communication processes which, like other communication processes (TCP/UDP/Web server communication...), are processed by the CPU with priority 15. OBs with higher priority interrupt the communication. If you set the sampling and transmission intervals too short, this setting causes a high communication load. Therefore, select intervals as large as possible, which are still sufficient for the application.

For information about the consistency of tags, refer to Consistency of CPU tags (Page 1209).

Monitoring of PLC tags

When the Subscription has been created, you inform the server which tags are to be monitored with it. In the following example, the "Voltage" tag was added to the subscription.

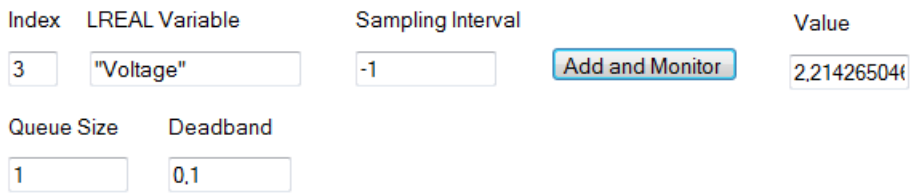

The "Voltage" tag contains the value of a voltage that is detected by an S7-1500 CPU.

The sampling interval ("Sampling Interval") contains a negative value (-1). This determines that the default setting of the OPC UA server is used for the sampling interval. The default setting is defined by the transmission interval ("Publishing Interval") of the subscription. If you want to set the smallest possible sampling interval, select the value "0".

In this example, the length of the queue is set to "1": Only one value is read from the CPU at an interval of 50 milliseconds and subsequently sent to the OPC UA client when the value has changed.

The "Deadband" parameter in this example is "0.1": Changes in value have to amount to 0.1 Volt; only then does the sender send the new value to the client. The server does not send smaller changes in value. You can use this parameter, for example, to disable signal noise: Slight changes in a process variable which do not have a real meaning.

Security at OPC UA

Security settings

Addressing risks

OPC UA allows the exchange of data between different systems, both within the process and production levels and to systems at the control and enterprise level.

This possibility also entails security risks. That is why OPC UA uses a range of security mechanisms:

- Verification of the identity of OPC UA server and clients.
- Checking of the identity of the users.
- Signed/encrypted data exchange between OPC UA server and clients.

These security policies should only be bypassed in cases where it is absolutely necessary:

- During commissioning
- In stand-alone projects without external Ethernet connection

If you have selected the endpoint "None" for "UA Sample Client" of the OPC Foundation, for example, the program issues a clear warning:

Warning: Selected Endpoint has no security.

When STEP 7 compiles your project it also checks whether you have considered the setting options for the protection and warns you of possible risks. This also includes an OPC UA security policy with the setting "no security", which corresponds to the end point "None".

Note

Disabling security policies you do not want

If you have selected all security policies (default setting) in the secure channel settings of the S7-1500 OPC UA server – in other words the end point "None" (no security) – non-secure data traffic (neither signed nor encrypted) between the server and client is also possible. The OPC UA server of the S7-1500 CPU also sends its public certificate to the client at "None" (No security). And some clients check this certificate. However, the client is not forced to send a certificate to the server. The identity of the client may possibly remain unknown. Each OPC UA client can then connect to the server irrespective of any subsequent security settings.

When configuring the OPC UA server, make sure that only security policies that are compatible with the security concept for your machine or plant are selected. All other security policies should be disabled.

Recommendation: Use the setting "Basic256Sha256 - Sign and Encrypt", which means that the server only accepts Sha256 certificates. The security policy "Basic128Rsa15" is disabled by default and should not be used as an end point. Select end points with a higher security policy.

Additional security rules

- Only use the end point "None" in exceptional cases.
- Only use the "guest authentication" of the user in exceptional cases.
- Only allow access to PLC tags and DB components via OPC UA if it is genuinely necessary.
- Use the list of trusted clients in the settings of the S7-1500 OPC UA client to allow access to certain clients only.

Certificates pursuant to ITU X.509

Security mechanisms are integrated in several layers in OPC UA. Digital certificates have an important role here. An OPC UA client can only establish a secure connection to an OPC UA server when the server accepts the digital certificate of the client and classifies it as trusted.

See section "Configuring the OPC UA server of the S7-1500 CPU".

The client must also check and trust the certificate of the server. The server and client must show their identities and prove that they are what they claim to be: They must prove their identity. Mutual authentication of client and server, for example, prevents man-in-the-middle attacks.

Man-in-the-middle attacks

A "man-in-the-middle" could have positioned itself between server and client. A man-in-the-middle is a program that intercepts communication between server and client and claims to be a client or server, and is thus able to obtain information about the S7 program or to set values in the CPU and attack a machine or plant.

OPC UA uses digital certificates that meet standard X.509 of the International Telecommunication Union (ITU).

This allows the identity of a program, a computer or an organization to be proven (authenticated).

X.509 certificates

An X.509 certificate includes the following information:

- Version number of the certificate
- Serial number of the certificate
- Information on the algorithm used by the certificate authority to sign the certificate.
- Name of the certificate authority
- Start and end of the validity period of the certificate
- Name of the program, person or organization for which/whom the certificate has been signed by the certificate authority.
- The public key of the program, person or organization.

An X509 certificate thus links an identity (name of a program, person or an organization) to the public key of the program, person or organization.

Check during connection establishment

When a connection is being established between the client and server, the devices check all information from the certificate that is required to establish integrity, for example signature, validity, application name (URN) and as of firmware V2.5 also the IP addresses of the client in the client certificate.

The validity period stored in the certificate is also checked. The CPU clock must therefore be set and date/time must be within the validity period, otherwise no communication takes place.

Signing and encryption

To allow you to check whether a certificate has been manipulated, certificates are signed.

There are various possible procedures here:

- Within the TIA Portal you have the possibility to generate and sign certificates. If you have protected your project and are logged in as a user with the function right to make security settings, you can use the global security settings. The global security settings allow access to the certificate manager and therefore to the certificate authority (CA) of the TIA Portal.
- Additional options are available for creating and signing certificates. In the TIA Portal, you can import certificates into the global certificate manager.
  - You contact a certificate authority (CA) and have your certificate signed.
    In this case, the certificate authority checks your identity and signs your certificate with the private key of the certificate authority. For this purpose you send a CSR (Certificate Signing Request) to the certificate authority. The process of creating a CSSR with the OpenSSL tool yourself is described here. (Page 1142)
  - You yourself create a certificate and sign it.
    To this purpose you use, for example, the "Opc.Ua.CertificateGenerator" program of the OPC Foundation. The procedure is described here. Or use OpenSSL: Instructions are available under Generating PKI key pairs and certificates yourself (Page 1143).

Useful information: Certificate types

- Self-signed certificate:
  Each device generates and signs its own certificate. Application examples: Static configuration with limited number of communication nodes.
  No new certificates can be derived from a self-signed certificate. However, you need to load all self-signed certificates from partner devices to the CPU (STOP required).
- CA certificate:
  All certificates are generated and signed by a certificate authority. Application examples: Dynamically growing plants.
  You only need to download the certificate from the certificate authority to the CPU. The certificate authority can generate new certificates (partner devices can be added without CPU STOP).

Signing

The signature makes it possible to prove the integrity and source of a message as detailed below.

Signing starts with the sender creating a hash value from the plain text (plain text message). The sender then encrypts the hash value with its private key and subsequently transfers the plain text together with the encrypted hash value to the recipient. To verify the signature, the recipient needs the public key of the sender (this is contained in the X509 certificate of the sender). The recipient uses the sender's public key to decrypt the hash value received. The recipient then forms the hash value themselves from the plain text received (the hash process is contained in the sender's certificate). The recipient compares the two hash values:

- If the two hash values are identical, the plain text message has reached the receiver unchanged and has not been manipulated.
- If the two hash values do not match, the plain text message has not reached the receiver unchanged. The plain text message has been manipulated or has been distorted during transfer.

Encryption

Encrypting data prevents unauthorized parties from reading the content. X509 certificates are not encrypted; they are public and can be viewed by anyone.

Encryption involves the sender encrypting the plain text message with the public key of the recipient. To do so, the sender requires the recipient's X509 certificate, as it contains the public key of the recipient. The recipient decrypts the message with their private key. Only the recipient can decrypt the message: They alone hold the private key. The private key must therefore never be disclosed.

Secure channel

OPC UA uses the private and public key of client and server to establish a secure connection, the secure channel. Once the secure connection has been established, the client and server generate an internal key only known to them which they both use for signing and encrypting messages. This symmetric process (a shared key) is much faster than asymmetric processes (private and public key).

Certificates with OPC UA

Usage of X509 certificates with OPC UA

OPC UA uses three types of X.509 certificates for establishing connections from client to server:

- OPC UA application certificates
  Such X.509 certificates identify the software instance, the installation of client or server software. For the "Organization name" attribute, you enter the name of the company that uses the software.

  Note

  The OPC UA server of the S7-1500 uses application certificates also for the security setting "None" (No security). This ensures compatibility to OPC UA V1.1 and earlier versions.

- OPC UA software certificates
  This X-509 certificate identifies a specific version of the client or server software. These certificates contain attributes that describe which tests this version of the software has passed during certification by the OPC Foundation (or recognized test laboratories). For the "Organization name" attribute, you enter the name of the company that has developed or markets the software.

  Note

  Software certificates are not supported in STEP 7.

- OPC UA user certificates
  This X.509 certificate identifies the specific user who, for example, accesses process data from the OPC UA server of an S7-1500 CPU. This certificate is not required if the user can authenticate themselves with a password, or if anonymous access is configured.

  Note

  User certificates are not supported in STEP 7.

These certificates are end-entity certificates: They identify, for example, a person, an organization, a company or an instance (installation) of a software.

Creating self-signed certificates

Using the client's certificate generator

Many OPC UA client applications or SDKs are integrated in a sample application that allows you to generate certificates for the client from this application.

The description for certificate generation can generally be found in the context for describing the OPC UA client application.

Example client from the online support

The OPC UA .NET client for the SIMATIC S7-1500 OPC UA server (https://support.industry.siemens.com/cs/ww/en/view/109737901) creates a self-signed software certificate of the client application in the Windows Certificate Store during the first program

start. The documentation for this example describes the procedure for handling these certificates.

Using the certificate generator of the TIA Portal

If you use an OPC UA client that does not generate a client certificate, self-signed certificates can be created with STEP 7.

To do this, follow these steps:

1. In the properties of the CPU, double-click "<Add new>" under "Protection & Security > Certificate manager > Device certificates".
2. Click "Add".
3. In the "Create a new certificate" dialog, select the "OPC UA client" option for "Usage".
4. Click "OK".

In the field "Subject Alternative Name" STEP 7 automatically enters the URI for the generated certificate. In the program-specific certificate generation by means of the .NET stack of the OPC Foundation, the field is called, for example, "ApplicationUri" - it can have a different name in other tools for certificate generation.

See also

Handling of the client certificates of the S7-1500 CPU (Page 1234)

Generating PKI key pairs and certificates yourself

This section is only relevant if you want to use an OPC UA client that cannot itself create a PKI key pair and a client certificate. In this case, you generate a private and a public key using OpenSSL, generate an X.509 certificate, and sign the certificate yourself.

Using OpenSSL

OpenSSL is a tool for Transport Layer Security that you can use to create certificates. You can also use other tools, for example XCA, a type of key management software with a graphical user interface for an improved overview of certificates issued.

To work with OpenSSL under Windows, follow these steps:

1. Install OpenSSL under Windows. If you are using a 64-bit version of the operating system, install OpenSSL in the "C:\OpenSSL-Win64" directory, for example. You can obtain OpenSSL-Win64 as a download from various providers for open source software.
2. Create a directory, for example "C:\demo".
3. Open the command line (cmd.exe). To do so, click "Start" and enter "cmd" in the search field. Right-click "cmd.exe" in the results list and run the program as an administrator. Windows then opens the command line (DOS prompt)
4. Change to the "C:\demo" directory. To do this, enter the following command: "cd C:\demo".

5. Set the following network variables:
   - set RANDFILE=c:\demo\.rnd
   - set OPENSSL_CONF=C:\OpenSSL-Win64\bin\openssl.cfg

   The figure below shows the command line with the following commands:

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten.

C:\demo>set RANDFILE=C:\demo\.rnd

C:\demo>set OPENSSL_CONF=C:\OpenSSL-Win64\bin\openssl.cfg

C:\demo>
```

6. Now start OpenSSL. If OpenSSL has been installed in the C:\OpenSSL-Win64 directory, enter the following: C:\OpenSSL-Win64\bin\openssl.exe The figure below shows the command line with the following command:

```
C:\WINDOWS\system32\cmd.exe - C:\OpenSSL-Win64\bin\openssl.exe

C:\demo>C:\OpenSSL-Win64\bin\openssl.exe
OpenSSL>
```

7. Generate a private key. Save the key to the "myKey.key" file. The key in this example is 1024 bits long; for greater RSA security, use 2048 bits in practice. Enter the following command: "genrsa -out myKey.key 2048" ("genrsa -out myKey.key 1024" in the example). The figure below shows the command line with the command and the output of OpenSSL:

```
C:\WINDOWS\system32\cmd.exe - C:\OpenSSL-Win64\bin\openssl.exe

C:\demo>C:\OpenSSL-Win64\bin\openssl.exe
OpenSSL> genrsa -out myKey.key 1024
Generating RSA private key, 1024 bit long modulus
.....++++++
........++++++
e is 65537 (0x10001)
OpenSSL>
```

8. Generate a CSR (Certificate Signing Request). To do this, enter the following command: "req -new -key myKey.key -out myRequest.csr". During execution of this command, OpenSSL queries information about your certificate:
   - Country name: for example "DE" for Germany, "FR" for France
   - State or province name: for example "Bavaria".
   - Location Name: for example "Augsburg".
   - Organization Name: Enter the name of your company.
   - Organizational Unit Name: for example "IT"
   - Common Name: for example "OPC UA client of machine A"
   - Email Address:

Important: The IP address of the client program has to be stored in the "Subject Alternative Name" field of the created certificate; otherwise, the CPU will not accept the certificate.

The information you enter is added to the certificate. The figure below shows the command line with the command and the output of OpenSSL:

```
C:\WINDOWS\system32\cmd.exe - C:\OpenSSL-Win64\bin\openssl.exe
OpenSSL> req -new -key myKey.key -out MyRequest.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
```

The command creates a file in the C:\demo directory containing the Certificate Signing Request (CSR); in the example, this is "myRequest.csr".

Using the CSR

There are two ways to use a CSR:

- You send the CSR to a certificate authority (CA): Read the information of the respective certification authority. The certificate authority (CA) checks your information and identity (authentication) and signs the certificate with the private key of the certificate authority. You receive the signed X.509 certificate and use this certificate for OPC UA, HTTPS or Secure OUC (secure open user communication), for example. Your communication partners use the public key of the certificate authority to check whether your certificate was really issued and signed by that CA (i.e. that the certificate authority has confirmed your information).
- You sign the CSR yourself: Using your private key. This option is shown in the next step.

Signing the certificate yourself

Enter the following command so that you can generate and sign your certificate (self-signed certificate) yourself: "x509 -req -days 365 -in myRequest.csr -signkey myKey.key -out myCertificate.crt".

The figure below shows the command line with the command and OpenSSL:

```
C:\WINDOWS\system32\cmd.exe - C:\OpenSSL-Win64\bin\openssl.exe
OpenSSL> x509 -req -days 365 -in myRequest.csr -signkey myKey.key -out myCertifi
cate.crt
Signature ok
subject=/C=DE/ST=Bayern/L=Augsburg/O=MyCompany/OU=IT/CN=MyName/emailAddress=MyEM
ail@me.de
Getting Private key
OpenSSL>
```

The command generates an X.509 certificate with the attributes that you transfer with the CSR (in the example "myRequest.csr"), for example with a validity of one year (-days 365). The command also signs the certificate with your private key ("myKey.key" in the example). Your communication partners can use your public key (contained in your certificate) to check that you are in possession of the private key that belongs to this public key. This also prevents your public key from being misused by an attacker.

With self-signed certificates, you yourself confirm that the information in your certificate is correct. There is no independent body that checks your information.

See also

Handling of the client certificates of the S7-1500 CPU (Page 1234)

Secure transfer of messages

Establishing secure connections with OPC UA

OPC UA uses secure connections between client and server. OPC UA checks the identity of the communication partners. OPC UA uses certificates in accordance with X.509-V3 from the ITU (International Telecommunication Union) for client and server authentication. Exception: A secure connection is not established with the "No security" security policy.

Message security mode

OPC UA uses the following security policies to protect messages:

- No security
  All messages are unsecured. In order to use this security policy, establish a connection to a None end point of a server.
- Signing
  All message are signed. This allows the integrity of the messages received to be checked. Manipulations are detected. In order to use this security policy, establish a connection to a Sign end point of a server.
- Sign & Encrypt
  All messages are signed and encrypted. This allows the integrity of the messages received to be checked. Manipulations are detected. What is more, no attacker can read the content of the message (protection of confidentiality). In order to use this security policy, establish a connection to a "SignAndEncrypt" end point of a server.

The security policies are also named according to the algorithms used. Example: "Basic256Sha256 - Sign & Encrypt" means: Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.

Layers required

The figure below shows the three layers that are always required for establishing a connection: the transport layer, the secure channel and the session.

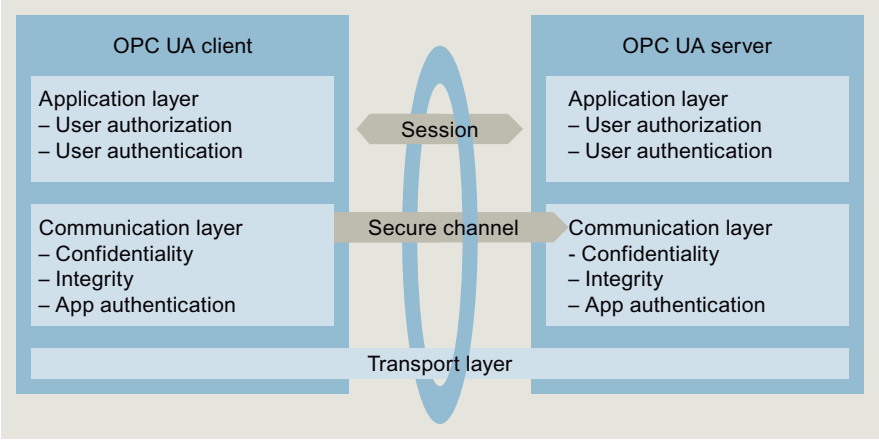

Figure 1-23 Necessary layers: transport layer, secure channel and session

- Transport layer:
  This layer sends and receives messages. OPC UA uses an optimized TCP-based binary protocol here. The transport layer is the basis for the subsequent secure channel.
- Secure channel
  The secure channel receives the data received from the transport layer, and forwards that data to the session. The secure channel forwards data of the session that is to be sent to the transport layer.
  In "Sign" security mode, the secure channel signs the data (messages) that is sent. When a message is received, the secure channel checks the signature to detect any manipulations.
  With a "SignAndEncrypt" security policy, the secure channel signs and encrypts the send data. Data received is decrypted by the secure channel, and the secure channel then checks the signature.
  With the "No security" security policy, the message packages pass the secure channel unchanged (the messages are received and sent in plain text).
- Session
  The session forwards the messages from the secure channel to the application, or receives from the application the messages that are to be sent. The application uses the process values or provides the values.

Establishing the secure channel

The secure channel is established as follows:

1. The server starts establishing the secure channel when it receives a request to this effect from the client. This request is signed or signed and encrypted, or the message is sent in plain text (security mode of the selected server end point). With "Sign" and "Sign & Encrypt", the client sends a "secret" (random number) with the request.
2. The server validates the client certificate (contained in the request, unencrypted) and checks the identity of the client. If the server trusts the client certificate,
   - it decrypts the message and checks the signature ("Sign & Encrypt"),
   - checks the signature only ("Sign"),
   - or leaves the message unchanged ("No security")
3. The server then sends a response to the client (same level of security as the request). The server secret is contained in the response. The client and server calculate a symmetric key from the client and server secret. The secure channel is now established.

The symmetric key (instead of the private and public key of client and server) is now used for signing and encrypting messages.

Establishment of the session

The session is executed as follows:

1. The client starts establishing the session by sending a CreateSessionRequest to the server. This message contains a Nonce, a random number that is only used once. The server must sign this random number (Nonce) to prove that it is the owner of the private key. The private key belongs to the certificate that the server uses to establish the secure channel. This message (and all subsequent messages) is secured in line with the security policies for the selected server endpoint (selected security policies).
2. The server responds with the CreateSession Response. This message contains the public key of the server and the signed Nonce. The client checks the signed Nonce.
3. If the server passes the test, the client sends a SessionActivateRequest to the server. This message contains the information that is required for user authentication:
   - Either the user name and password
   - Or the X.509 certificate of the user (not supported in STEP 7 V15)
   - Or no data (if anonymous access is configured).
4. If the user has the necessary rights, the server returns a message to the client (ActivateSessionResponse). This activates the session.

The secure connection between the OPC UA client and server has been established.

Establishing a connection to PLCopen function block

The PLCopen specification defines a range of IEC 61131 function blocks for OPC UA clients. The instruction UA_Connect initiates both a secure channel and a session following the pattern described above.

Using the S7-1500 as an OPC UA server

Interesting information about the S7-1500 CPU OPC UA server

The OPC UA server of the S7-1500 CPUs

The S7-1500 CPUs as of firmware V2.0 are equipped with an OPC UA server. Apart from the Standard-S7-1500 CPUs this applies to the variants S7-1500F, S7-1500T, S7-1500C, S7-1500pro CPUs, ET 200SP CPUs, SIMATIC S7-1500 SW controllers and PLCSIM Advanced.

Convention: "S7-1500 CPUs" also includes the above-mentioned CPU variants.

S7-1500 CPU OPC UA server basics

Access to the OPC UA server of the CPU is possible via all integrated PROFINET interfaces of the S7-1500 CPU.

Direct access to the OPC UA server of the CPU over the backplane bus of the automation system is not possible via a CP or CM.

For access by clients, the server saves the enabled PLC tags and other information in the form of nodes (see AUTOHOTSPOT). These nodes are interconnected and form a network. OPC UA defines access points to this network (well-known nodes) that enable navigation to subordinate nodes.

With an OPC UA client you can read, observe or write values of tags of the PLC program as well as call methods that are available to the server. As of firmware version 2.5 you can implement methods, see Important facts about server methods (Page 1183).

Node classes

OPC UA servers provide information in the form of nodes. A node can be, for example, an object, a tag, a method or a property.

The example below shows the address space of the OPC UA server of an S7-1500 CPU (extract from the OPC UA client "UaExpert" from Unified Automation).

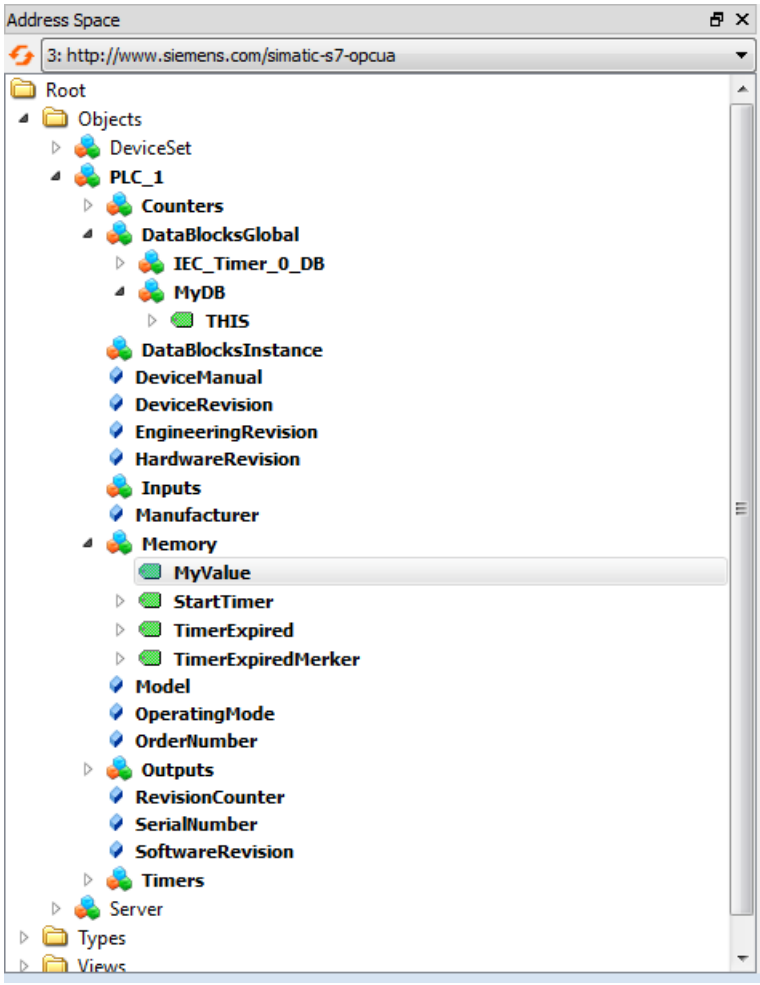

Figure 1-24 Example of the address space of the OPC UA server of an S7-1500 CPU

In the figure above, the "MyValue" tag is selected (highlighted in gray).

This tag is located below the "Memory" node, which has the node class "Object".

"Memory" is below the "PLC_1" node (also an Object).

Address space

The nodes are linked over references, for example, the reference "HasComponent, which represents a hierarchical relationship between a node and its subordinate nodes. With their references, the nodes form a network that can, for example, take the form of a tree.

A network of nodes is also called an address space. Starting from the root, all nodes can be reached in the address space.

End points of the OPC UA server

The end points of the OPC UA server define the security level for a connection. Depending on the purpose of use or desired security level, you have to carry out the corresponding settings for the connection at the end point.

Different security settings

Before establishing a secure connection, OPC UA clients ask the server with which security settings connections are possible. The server returns a list with all the security settings (endpoints) that the server offers.

Structure of end points

End points consist of the following components:

- Identifier for OPC: "opc.tcp"
- IP address: 192.168.178.151 (in the example)
- Port number for OPC UA: 4840 (standard port)
  The port number can be configured.
- Security setting for messages (Message Security Mode): None, Sign, SignAndEncrypt.
- Encryption and hash procedures (Security Policy): None, Basic128Rsa15, Basic256, Basic256Sha256 (in the example).

The following figure shows the "UA Sample Client" of the OPC Foundation.

The client has established a secure connection to the OPC UA server of an S7-1500 CPU to the end point "opc.tcp://192.168.178.151:4840 - [SignAndEncrypt: Basic128Rsa15:Binary]". The security settings "SignAndEncrypt:Basic128Rsa15" are contained in the end point.

Note

Select an endpoint with as strict as possible a security policy

Select an application-appropriate security policy for the end point and disable the less strict security policy at the OPC UA server.

A Sha256 certificate is required for the most secure end points (Basic256Sha256) of the S7-1500 CPU OPC UA server.

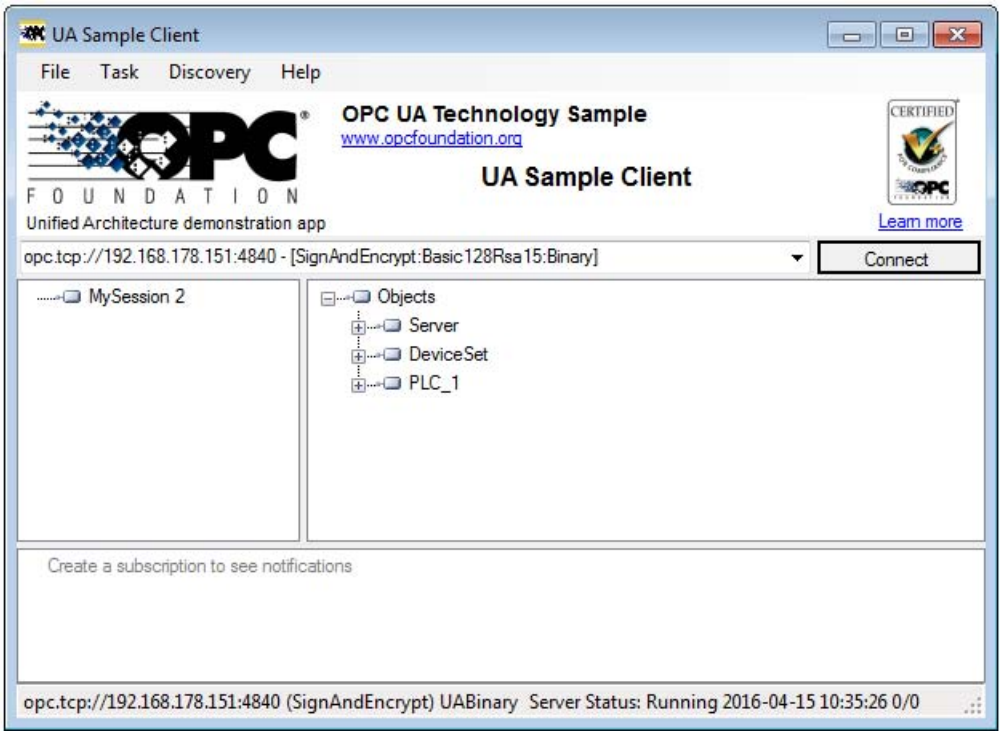

Figure 1-25 "UA Sample Client" program of the OPC Foundation

A connection to a server end point is only established if the OPC UA client complies with the security policies of that end point.

Through the information provided by the OPC UA server

OPC UA servers provide a wide range of information:

- The values of PLC tabs and DB components which clients may access.
- The data types of these PLC tags and DB components.
- Information on the OPC UA server itself and on the CPU.

This gives clients an overview and allows them to read out specific information. Previous knowledge of the PLC program and the CPU data is not required. You do not need to ask the developer of the PLC program when PLC tags are to be read. All necessary information is stored on the server itself (for example, the data types of the PLC tags).

Display of the information of the OPC UA server

You have the following options:

- Online: You have all the available information displayed during the runtime of the OPC UA server. To do so, navigate (browse) the address space of the server.
- Offline: You export an XML file that is based on the XML schemes of the OPC Foundation. Server methods created by the user (FB instance that can be called by an OPC UA client) are not exported as of STEP 7 V15.1), see AUTOHOTSPOT.

- Offline with the Openness API: In your program, you use the API (Application Programming Interface) of the TIA Portal to access the function for exporting all PLC tags that can be read by OPC UA. This requires .NET Framework 4.0; see TIA Portal Openness, Automating SIMATIC projects with scripts (https://support.industry.siemens.com/cs/ww/en/view/109477163).
- If you already know the syntax and the PLC program, you can access the OPC UA server without first researching the information.

Runtime behavior of the OPC UA server

OPC UA server in operation

The OPC UA server of the S7-1500 CPU starts when you activate the server and download the project to the CPU.

How to activate the OPC UA server is described here.

Response to CPU STOP

An activated OPC UA server remains in operation even if the CPU switches to "STOP". The OPC UA server continues to respond to requests from OPC UA clients.

Server response in detail:

- If you request the values of PLC tags, you will get what were the latest values before the CPU switched to or was set to "STOP".
- If you write values to the OPC UA server, the OPC UA server will accept those values. However, the CPU will not process the values because the user program is not executed in "STOP" mode.
  An OPC UA client can nonetheless read the values written at STOP from the OPC UA server of the CPU.
  During restart, the CPU overwrites the values written at STOP with the start of the PLC tags.
- If you call a server method, the error message 16#00AF_0000 (BadInvalidState) is output because the server method (user program) is not running.

Server restart

The OPC UA server is stopped upon each download to the CPU (for example after a configuration or block is downloaded) and then restarts. The duration of the restart depends on the scope of the data structure.

Reading CPU operating mode over OPC UA server

The OPC UA server allows you to read out the CPU mode, see figure below:

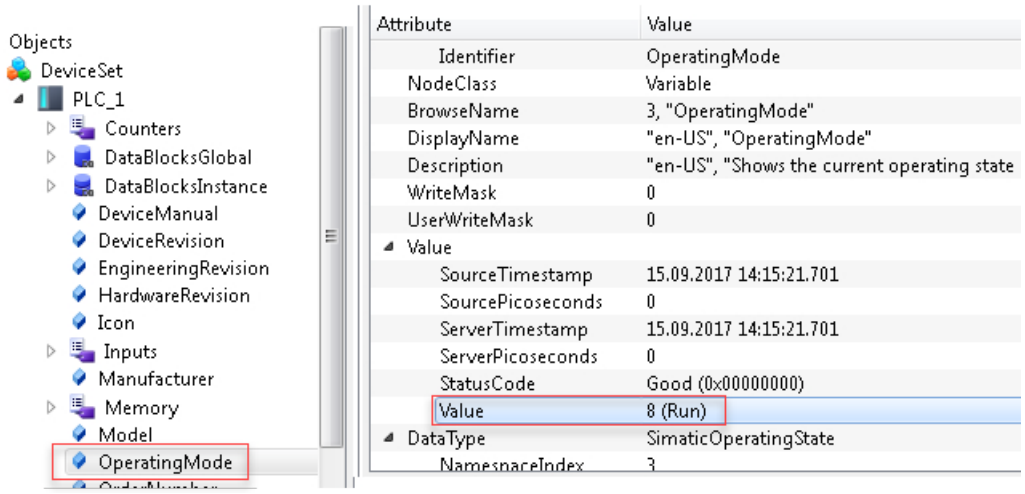

Figure 1-26 Reading CPU operating mode over OPC UA server

In addition to the operating mode of the CPU you can, for example, read out the diagnostics status of the CPU (Good).

Diagnostics of the OPC UA server

Online diagnostics of the OPC UA server

The S7-1500 CPU OPC UA server can be diagnosed online with standard OPC UA clients, such as UaExpert.

The diagnostic information is subdivided into the following areas:

- Server Diagnostics
- Sessions Diagnostics
- Subscriptions Diagnostics

In the address space of the server, for example, the following nodes are available with diagnostic information:

- **ServerDiagnosticsSummary**: Server diagnostics summary
  - CurrentSessionCount: Number of active sessions
- **SessionsDiagnosticsSummary**: Session diagnostics summary
  - Session timeout: Set time that a session lasts, e.g. in the event of disconnection
  - SecurityRejectedSessionCount: Number of sessions rejected due to mismatching end point security settings between client and server
- **SubscriptionsDiagnosticsArray**: ARRAY with one element per subscription for each session

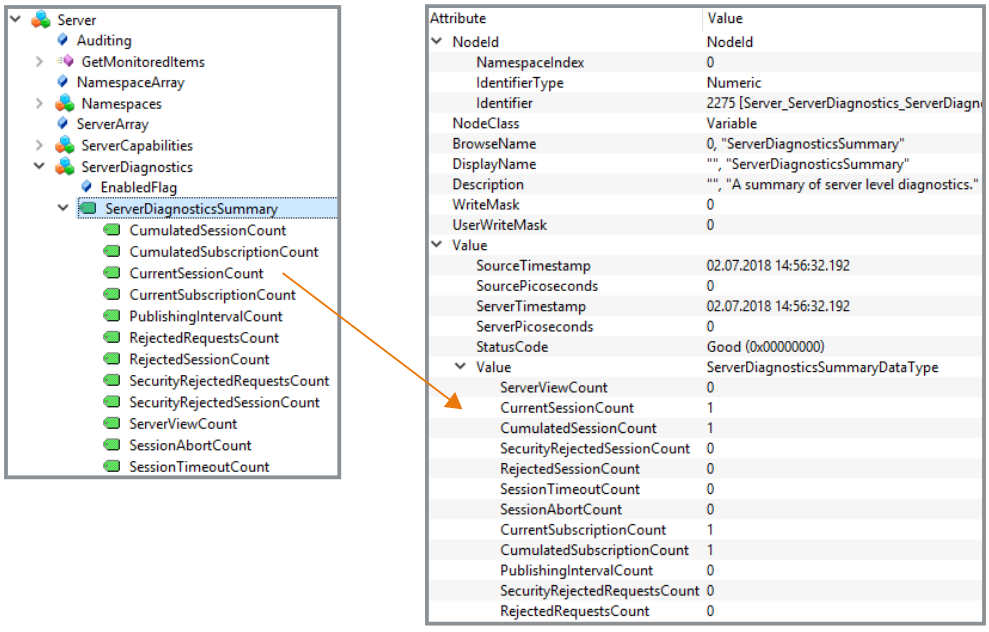

The SessionsDiagnosticsSummary node also shows the properties of the client application accessing the server within the session.

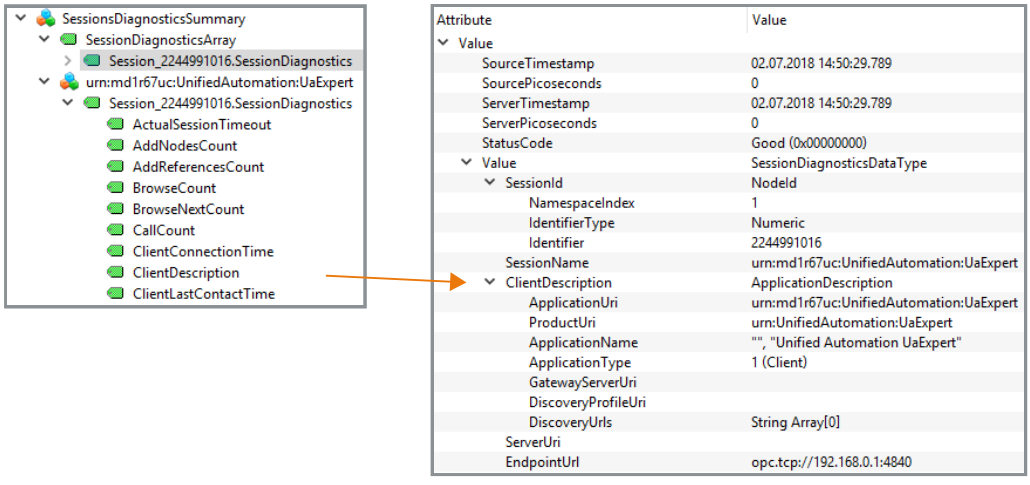

Diagnostics of the connection between client and server

To diagnose the status of the connection during program runtime in the client, use the following instruction:

OPC_UA_ConnectionGetStatus: Read connection status.

Configuring access to PLC tags

Managing write and read rights

Enabling PLC tags and DB tags for OPC UA

OPC UA clients can have read and write access to PLC tags and DB tags if the tags are enabled for OPC UA (default setting). For an enabled tag the check box "Accessible from HMI/OPC UA" is activated.

The following example shows an array data block:

MyDB

Name	Data type	Accessible from HMI/OPC UA	Writable from HMI/OPC UA	Visible in HMI engineering
MyDB	Array[0..9] o...	☑	☑	☑
MyDB[0]	Int	☑	☑	☑
MyDB[1]	Int	☑	☑	☑
MyDB[2]	Int	☑	☑	☑
MyDB[3]	Int	☑	☑	☑
MyDB[4]	Int	☑	☑	☑
MyDB[5]	Int	☑	☑	☑
MyDB[6]	Int	☑	☑	☑
MyDB[7]	Int	☑	☑	☑
MyDB[8]	Int	☑	☑	☑
MyDB[9]	Int	☑	☑	☑

Figure 1-27 Enabling PLC tags and DB tags for OPC UA

This array can be read completely in one pass by OPC UA clients (see Addressing nodes (Page 1128)). The check boxes at "Accessible from HMI/OPC UA" and "Writable from HMI/OPC UA" are activated for all the components of the array.

Result: OPC UA clients can both read and write these components.

Removing write rights

If you want to write-protect a tag, deselect the "Writable from HMI/OPC UA" option for that tag. This removes the write right for the OPC UA clients and HMI devices.

Result: Only read access by OPC UA clients and HMI devices is possible. OPC UA clients cannot assign values to this tag and therefore cannot influence execution of the S7 program.

Removing write and read rights

To write-protect and read-protect a tag, disable the "Accessible from HMI/OPC UA" option for that tag (checkbox not selected). This makes the OPC UA server remove the tag from its address space. OPC UA clients can no longer see that CPU tag.

Result: OPC UA clients and HMI devices can neither read nor write the tag.

Write and read rights of structures

If you remove the read or write right for the component of a structure, the structure or the data block cannot be written or read as a whole.

If you remove read and write rights for individual components of a PLC data type (UDT), the rights will also be removed from any data block based on that data type!

Visible in HMI engineering

The option "Visible in HMI Engineering" applies to Siemens engineering tools. If you disable the option "Visible in HMI Engineering" (check mark not set), you can no longer configure the tag in WinCC (TIA Portal).

The option does not have any effect on OPC UA.

Rules

- Only allow read access to PLC tags and tags of data blocks in STEP 7 if this is necessary for communication with other systems (controllers, embedded systems or MES).
  You should not enable other PLC tags.
- Only allow write access over OPC UA if write rights are genuinely necessary for specific PLC tags and tags of data blocks.
- If you have reset the "Accessible from HMI/OPC UA" option for all elements of a data block, the data block for an OPC UA client is no longer visible in the address space of the OPC UA server of the S7-1500 CPU.
- You can also prevent access to an entire data block centrally (see Managing write and read rights for a complete DB (Page 1157)). This setting "overrules" the settings for the components in the DB editor.

Managing write and read rights for a complete DB

Hiding DBs or DB contents for OPC UA clients

As of STEP 7 V15, you have the option of easily preventing access to a complete data block by an OPC UA client.

With this option, the data of the corresponding DB, including instance DBs of function blocks, remains hidden for OPC UA clients.

In the default setting, data blocks can be read and written from OPC UA clients.

Procedure

Proceed as follows to completely hide a data block for OPC UA clients or to protect a data block from write access from OPC UA clients:

1. Select the data block to be protected in the project tree.
2. Select the "Properties" shortcut menu.

3. Select the "Attributes" area.
4. Select/clear the "DB accessible from OPC UA" checkbox as required.

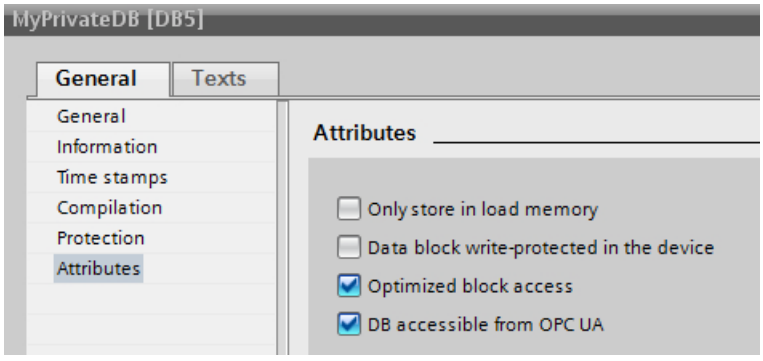

Figure 1-28 Hiding DBs or DB contents for OPC UA clients

Note

Effect on settings in the DB editor

If you hide a DB using the DB attribute described here, the settings for the components in the DB editor are no longer relevant; individual components can no longer be accessed or written.

Accessing OPC UA server data

High performance in line with application

OPC UA is designed for the transfer of a high volume of data within a short period of time. You can increase the performance significantly if you do not access individual PLC tags, but rather read and write arrays and structures as a whole.

It is fastest to access arrays. Therefore, you should combine the data for OPC UA clients into arrays.

Recommendations for access to the OPC UA server by the OPC UA client

- For one-off or infrequent data access, use standard read/write access.
- For cyclic access to small amounts of data (up to ca. every 5 seconds), use subscriptions. Optimize the settings for the smallest publishing interval and the smallest sampling interval at the OPC UA server.
- If you access specific tags regularly (recurring access), you should use the functions "RegisteredRead" and "RegisteredWrite".

Allow a greater communication load for the PLC by increasing the value for "Cycle load due to communication". Make sure that your application still works properly with the changed settings.

Procedure for creating an array DB

You can create arrays for example in global data blocks, in the instance data block of a function block or as an array DB . The following sections describe how to create an Array-DB.

To create a data block with an array (array data block), follow these steps:

1. Select the CPU with the OPC UA server in the project tree.
2. Double-click "Program blocks".
3. Double-click "Add new block".
4. Click "Data block".
5. Select a unique name for the data block and accept the name that is already entered.
6. Select the "Array DB" entry from the drop-down list for "Type".
7. Select the data type for the individual components of the array from the drop-down list for "Array data type".
8. Enter the high limit of the array for "Array limit".
9. Click "OK".

Export OPC UA XML file

Generating an OPC UA export file

OPC UA specifies an information model XML scheme with a standard syntax, with which manufacturers can define their information model of a system component, for example, and make it available in machine-readable form. A file that uses this scheme is also known as a node set file.

With STEP 7 (TIA Portal) you can easily export the standard SIMATIC information model of the S7-1500 CPU as a server to an OPC UA XML file (node set file); including all PLC variables and methods you have enabled for OPC UA.

You use the OPC UA XML file for the offline configuration of an OPC UA client; it is structured according to the OPC UA specification and acts as a standard SIMATIC server interface.

To create and export the OPC UA XML file, follow these steps:

1. Select the CPU. Click on the CPU symbol (for example in the network view).
2. Click "General > OPC UA > Server > Export" in the properties of the CPU.
3. Click "Export OPC UA XML file".
4. Select the directory in which you want to save the export file.
5. Select a new name for the file or keep the name that is already entered.
6. Click "Save".

Note

As of STEP 7 (TIA Portal) V15.1, server methods are contained in the OPC UA export file (node set) together with their input and output parameters.

Exporting all array elements separately

If the "Export all array elements as separate nodes" option is selected in the CPU properties under "OPC UA > Server > Export", the OPC UA XML file contains all elements of arrays as individual XML elements. In addition, the arrays themselves are each described in an XML element in the XML file.

If an array contains many array elements, the XML file can be very large.

Tip

The following FAQ contains a converter with which you can convert the export file into CSV format. You then obtain a list of the tags of the CPU that can be accessed by OPC UA.

You can find the FAQ on the Internet (https://support.industry.siemens.com/cs/ww/en/view/109742903).

Configuring the OPC UA server of the S7-1500 CPU

Enabling the OPC UA server

Requirement

- You have acquired a Runtime license for the operation of the OPC UA server; see Licenses for the OPC UA server (Page 1183).
- When using certificates for secure communication (for example, HTTPS, secure OUC, OPC UA), make sure that the corresponding modules have the current time of day and the current date. Otherwise, the modules will evaluate the certificates used as invalid and secure communication will not work.

Commissioning an OPC UA server

By default, the OPC UA server of the CPU is not enabled for reasons of security: OPC UA clients have neither write nor read access to the S7-1500 CPU.

Follow these steps to activate the OPC UA server of the CPU:

1. Select the CPU. Click on the CPU symbol (for example in the network view).
2. Click "OPC UA > Server" in the properties of the CPU.
3. Activate the OPC UA server of the CPU.
4. Confirm the security notes.

5. Go to the CPU properties, select "Runtime licenses" and set the runtime license acquired for the OPC UA server.
6. Compile the project.
7. Download the project to the CPU.
   The OPC UA server of CPU now starts.

Settings remain stored

If you have already enabled the server and made settings, those settings are not lost if the server is disabled. The settings are saved as before and are available when you enable the server again.

Application name

The application name is the name of the OPC UA application and applies to the server and the client. The name is displayed under "OPC UA > General":

- The default for the application name is: "SIMATIC.S7-1500.OPC-UA.Application:PLC_1".
- The default consists of "SIMATIC.S7-1500.OPC-UA.Application:" and the name of the CPU selected under "General > Product information > Name", in this case "PLC_1".
- The OPC UA server uses this application name to identify itself to a communication partner (OPC UA client), for example, when an OPC UA client uses the discovery service to detect accessible servers.
- The displayed application name uses the OPC UA client of the CPU when connecting to an OPC UA Server. This means that the CPU enters this application name automatically as "ApplicationName" for the instruction "OPC_UA_Connect" (tag of type "OPC_UA_SessionConnectInfo" at the parameter "SessionConnectInfo" of the instruction "OPC_UA_Connect").
  When you program the instruction "OPC_UA_Connect" you must therefore assign an empty string to the "ApplicationName". You can use the application name, for example, to identify the client and its sessions (SessionNames) for diagnostic purposes.

If you have activated the server, you can also use a different name that is meaningful in your project and which fulfils the requirements of your project, e.g. according for worldwide uniqueness.

The example below originates from UaExpert:

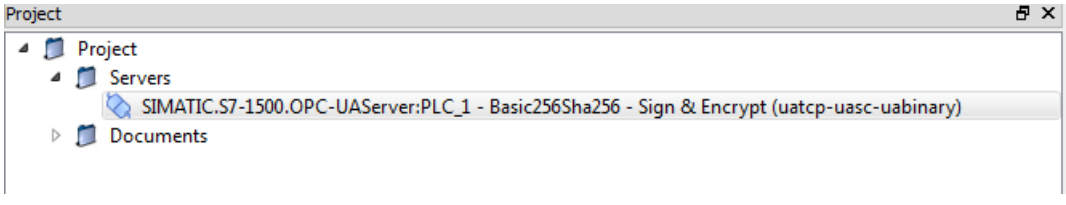

Changing the application name

To change the application name, follow these steps:

1. Select the CPU. Click on the CPU symbol (for example in the network view).
2. Click "OPC UA > General" in the properties of the CPU.
3. Enter a meaningful name.

Please note that the application name is also entered on the certificate (Subject Alternative Name) and you may have to generate an existing certificate again after changing the application name.

Access to the OPC UA server

Server addresses

The OPC UA server of the S7-1500 CPU can be accessed over all internal PROFINET interfaces of the CPU (as of firmware V2.0), but not over the PROFINET interfaces of CP/CM.

With SIMATIC S7-1500 SW controllers, access to the OPC UA server is only possible via PROFINET interfaces that are assigned to the software PLC.

In the example, connections to the OPC UA server of the CPU can be established over the following URLs (Uniform Resource Locator):

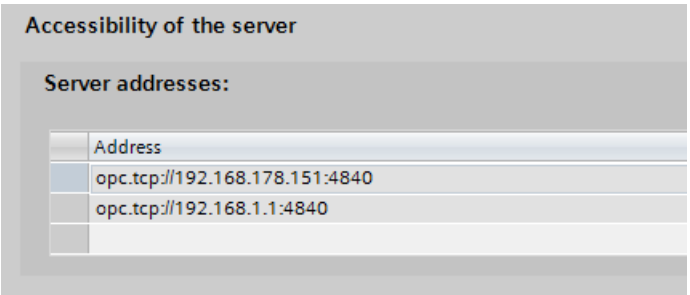

The URLs are structured as follows:

- Protocol identifier "opc.tcp://"
- IP address
  - 192.168.178.151
    The IP address at which the OPC UA server can be accessed from the Ethernet subnet 192.168.178.
  - 192.168.1.1
    The IP address at which the OPC UA server can be accessed from the Ethernet subnet 192.168.1.
- TCP Port number
  - Default: 4840 (standard port)
    The port number can be changed under "OPC > UA > Server > Port".

Dynamic IP addresses

In the example below, the IP address for the PROFINET interface [X2] has not yet been specified.

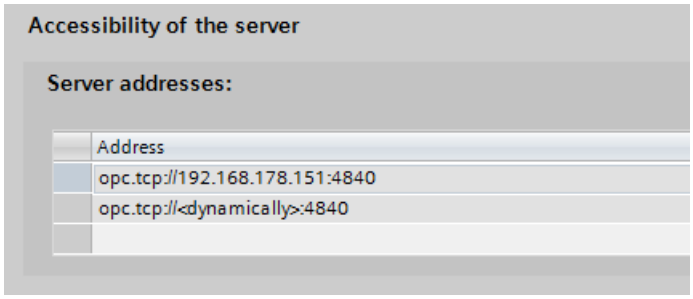

The placeholder "<dynamically>" appears in the table.

The IP address of this PROFINET interface is set later on the device, e.g. via the display of the CPU.

Activating the standard SIMATIC Server interface

If the "Activate Standard SIMATIC server interface" option is selected, the OPC UA server of the CPU provides the enabled PLC tags and server methods to the clients, as is specified in the OPC UA specification.

This option is selected in the default setting.

Leave the option selected so that OPC UA clients can automatically connect to the OPC UA server of the CPU and exchange data.

If you do not select this option, you must add the server interface by entering the "OPC UA communication" entry in the project tree. This interface is then used as OPC UA server interface, see AUTOHOTSPOT.

Backward compatible data type definitions according to OPC UA specification ≤ V1.03

The OPC UA specification (<= V1.03) defines mechanisms in order to read out data type definitions, for example for user-defined structures (UDTs), from a server by means of the TypeDictionaries.

In the OPC UA server properties of the CPU, you can set whether the CPU generates these backward compatible data type definitions according to the OPC UA specification ≤ V1.03 for the standard SIMATIC server interface or not.

Because TypeDictionaries are complex and result in large OPC UA XML files (server interfaces) which the client has to interpret, a simpler solution was introduced with OPC UA Specification V1.04 (attribute "DataTypeDefinition" at the data type node). If your client supports the OPC UA specification V1.04 or higher, then disable the option.

General settings of the OPC UA server

TCP port for OPC UA

By default, OPC UA uses TCP port 4840. You can, however, select a different port. Entries from 1024 to 49151 are possible. You must, however, make sure that there are no conflicts with other applications. OPC UA clients must use the selected port when establishing a connection.

In the example below, port 48400 is selected:

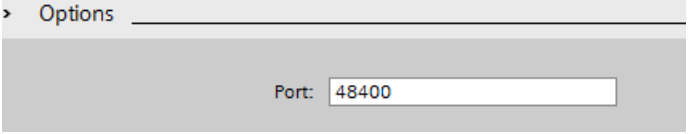

Settings for sessions

- Maximum timeout for sessions
  In this field, you specify the maximum time period before the OPC UA server closes a session without data exchange.
  Possible values between 1 and 600000 seconds.
- Maximum number of OPC UA sessions
  In this field, you specify the maximum number of sessions the OPC UA server starts and simultaneously operates.
  The maximum number of sessions is dependent on the performance capability of the CPU. Each session ties up resources.

Maximum number of registered nodes

In this field, you specify the maximum number of nodes the OPC UA server registers.

The maximum number of registered nodes depends on the capacity of the CPU and is displayed when you configure the field content (place cursor in field). Each registration ties up resources.

Note

No error message following attempt to register more than the configured maximum number of registrable nodes

If a client tries to register more nodes during runtime than the configured maximum number, the server of the S7-1500 CPU only registers the configured maximum number. From the configured maximum number of registrable nodes, the server sends the client the regular string node IDs so the speed advantage gained by registration for these nodes is lost. The client does not receive an error message.

When configuring, make sure you have a sufficient reserve by taking into account the maximum number of nodes that can be registered (for example, using the technical data of the CPU).

Additional information

Details on which ports are used by the various services for data transfer via TCP and UDP, and what are the points to note when using routers and firewalls can be found in the FAQ (https://support.industry.siemens.com/cs/ww/en/view/8970169).

Settings of the server for subscriptions

Subscription instead of cyclic queries

An alternative to cyclic queries for a PLC tag (polling) is to monitor this value. Use a Subscription: The server informs the client if the value of PLC tags changes. See "The OPC UA client".

One server usually monitors a large number of PLC values. At regular intervals, the server therefore sends the client messages (notifications) containing the new values of the PLC tags.

How frequently does the server send notifications?

When a Subscription is set up, the OPC UA client specifies the intervals at which it wants to be sent the new values in the event of changes. To limit the communication load through OPC UA, set a minimum interval for the messages. For this purpose, use the parameters for the minimum publishing interval and the minimum sampling interval.

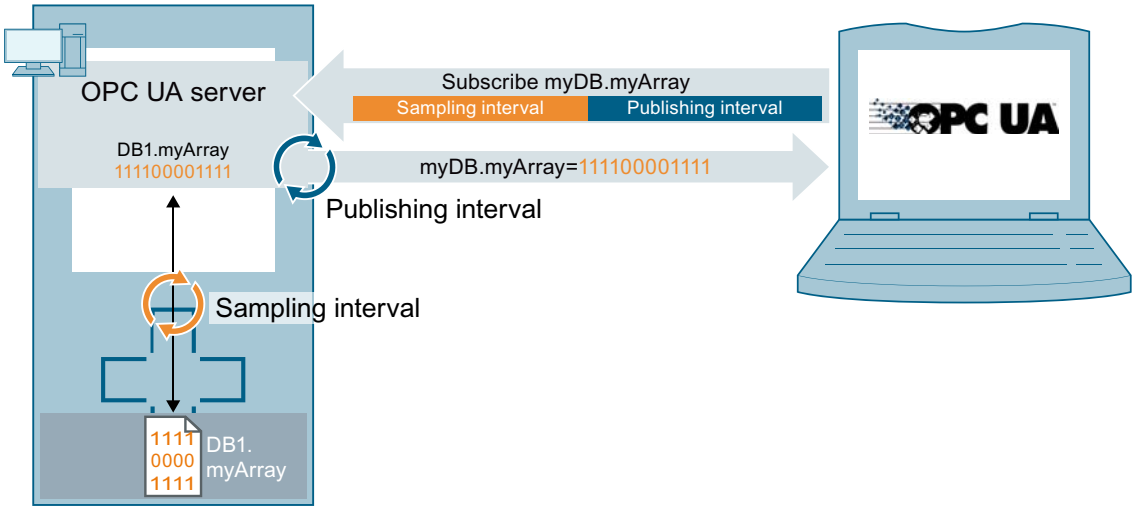

Minimum publishing interval

With "Minimum publishing interval", you set the time intervals at which the server sends a message to the client with the new values in the event of changes.

250 ms is used as the "Minimum sampling interval" in the figure below. The value 200 ms is entered as the "Minimum publishing interval".

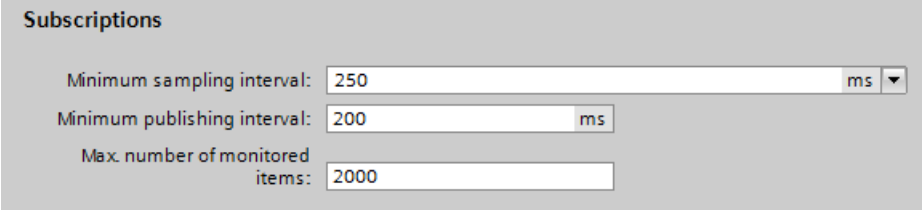

In the example, following a value change the OPC UA server will send a new message every 200 ms if the OPC UA client requests an update.

If the OPC UA client requests an update every 1000 ms, the OPC UA server will only send a message with the new values once every 1000 ms (one second).

If the OPC UA client requests an update every 100 ms, the server will still only send a message every 200 ms (minimum publishing interval).

Minimum sampling interval

With "Minimum sampling interval", you set the time intervals at which the OPC UA server records the value of a CPU tag and compares it with the previous value to detect any changes.

If the sampling interval is selected smaller than the publishing interval and an OPC UA client requests a high sampling rate for certain PLC tags, two or more values may be measured during each publishing interval.

In this case, the OPC UA server writes the value changes into the queue and sends all value changes to the client after the completion of the publishing interval. If more value changes occur in the publishing interval than fit in the queue, the OPC UA server overwrites the oldest values. (depending on the set "Discard Policy", the option "Discard Oldest" has to be activated in this case). The most recent values are sent to the client.

Maximum number of monitored elements (monitored items)

In this field, you specify the maximum number of elements that the OPC UA server of the CPU simultaneously monitors for a value change.

The monitoring ties up resources. The maximum number of monitored elements is dependent on the utilized CPU.

Additional information

Information about the system limits of the OPC UA server of the S7-1500 CPUs (firmware V2.0 and V2.1) regarding subscriptions, sampling intervals and publishing intervals can be found in the following FAQ (https://support.industry.siemens.com/cs/ww/en/view/109755846).

When using subscriptions, certain status codes of errors provide information on the error that occurred. For information on causes and remedies for status codes of OPC UA client that appear, see the list of error codes in the online help of Step 7 (TIA Portal) or in the following FAQ (https://support.industry.siemens.com/cs/ww/en/view/109755860).

Handling client and server certificates

A secure connection between the OPC UA server and an OPC UA client is only established when the server can prove its identity to the client. This is done with the server certificate.

Certificate of the OPC UA server

When you have activated the OPC UA server and have confirmed the security prompts, STEP 7 automatically generates the certificate for the server and saves it in the local certificate directory of the CPU. You can view and manage this directory with the local certificate manager of the CPU (exporting or deleting certificates).

The figure below shows the local certificate manager of the CPU with the automatically generated certificate for the OPC UA server:

Figure 1-29 Local certificate manager of the CPU

Alternatively, you can also generate a server certificate yourself.

The certificate of the server is transferred from the server to the client during establishment of a connection. The client checks the certificate.

The client user decides whether the server certificate is to be trusted.

The user at the client side now has to decide whether the server certificate is to be trusted. If the user trusts the server certificate, the client stores the server certificate in its directory containing the trusted server certificates.

The following example shows a dialog of the client "UA Sample Client". When the user clicks the "Yes" button, the client trusts the server certificate:

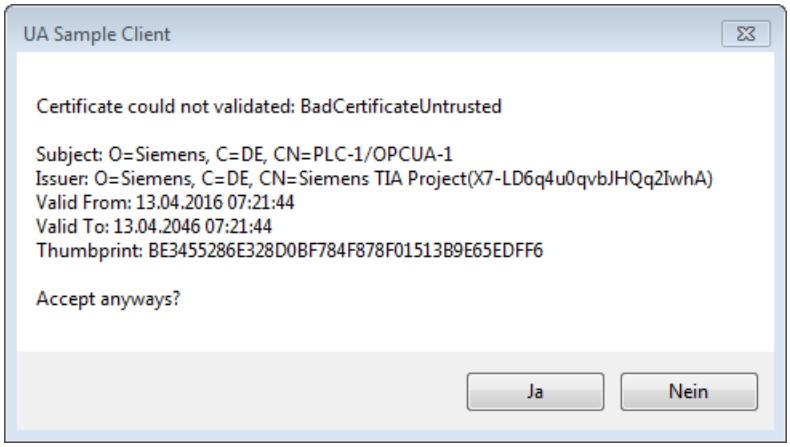

Figure 1-30 Dialog of the client "UA Sample Client"

See also

Generating server certificates with STEP 7 (Page 1173)

Secured transferring of messages

Where does a client certificate come from?

When you use UA clients from manufacturers or the OPC Foundation, a client certificate is generated automatically during installation or upon the first program call. You have to import these certificates via the global certificate manager in STEP 7 and use them for the corresponding CPU (as shown above).

When you program an OPC UA client yourself, you can have the certificates generated by the program; see the section "Instance certificate for the client". Alternatively, you can generate certificates with tools, for example with OpenSSL or the certificate generator of the OPC Foundation:

- The procedure for OpenSSL is described here: "Generating PKI key pairs and certificates yourself".
- Working with the certificate generator of the OPC Foundation is described here: "Creating self-signed certificates".

Announcing client certificates to the server

You need to send client certificates to the server to allow a secure connection to be established.

To do this, follow these steps:

1. Select the "Use global security settings for certificate manager" option in the local certificate manager of the server. This makes the global certificate manager available.
   You will find this option under "Protection & Security > Certificate manager" in the properties of the CPU that is acting as server.
   If the project is not yet protected, select "Security settings > Settings" in the STEP 7 project tree, click the "Protect this project" button and log on.
   The "Global security settings" item is now displayed under "Security settings" in the STEP 7 project tree.
2. Double click "Global security settings".

3. Double click "Certificate manager".
   STEP 7 opens the global certificate manager.
4. Click on the "Trusted certificates" tab.
5. Right-click in the tab on a free area (not on a certificate).
6. Select the "Import" command from the shortcut menu.
   The dialog for importing certificates is displayed.
7. Select the client certificate that the server is to trust.
8. Click "Open" to import the certificate.
   The certificate of the client is now contained in the global certificate manager.
   Note the ID of the client certificate just imported.
9. Click the "General" tab in the properties of the CPU that is acting as server.
10. Click "OPC UA > Server > Security > Secure Channel".
11. Scroll down in the "Secure Channel" dialog to the section "Trusted clients".
12. Double-click in the table on the empty row with "<add new>". A browse button is displayed in the row.
13. Click this button.
14. Select the client certificate that you have imported.
15. Click the button with the green check mark.
16. Compile the project.
17. Load the configuration onto the S7-1500 CPU.

Result

The server now trusts the client. If the server certificate is also considered trusted, the server and client can establish a secure connection.

Accepting client certificates automatically

When you select the option "Automatically accept all client certificates during runtime" (below the "Trusted clients" list), the server automatically accepts all client certificates.

NOTICE
Setting after commissioning In order to avoid security risks, deactivate the "Automatically accept client certificates during runtime" option again after commissioning.

Configuring security settings of the server

The figure below shows the available server security settings for signing and encrypting messages.

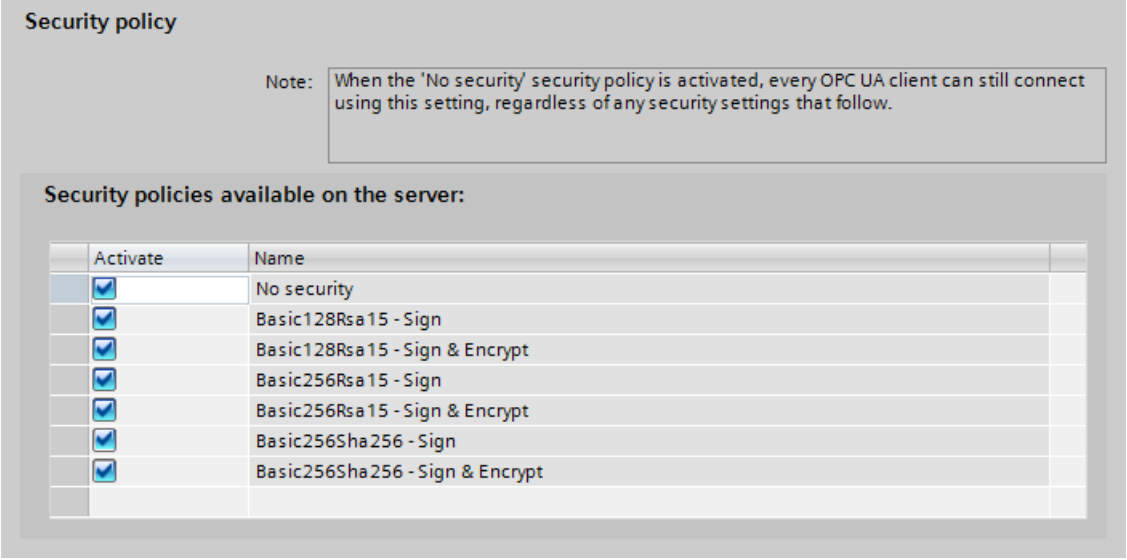

Figure 1-31 Configuring security settings of the server

By default, a server certificate is created that uses SHA256 signing. The following security policies are released:

- None
  Insecure end point

  Note

  Disabling security policies you do not want

  If you have selected all security policies (default setting) in the secure channel settings of the S7-1500 OPC UA server – in other words the end point "No Security" – non-secure data traffic (neither signed nor encrypted) between the server and client is also possible. The identity of the client remains unknown with "No security". Each OPC UA client can then connect to the server irrespective of any subsequent security settings.

  When configuring the OPC UA server, make sure that only security policies that are compatible with the security concept for your machine or plant are selected. All other security policies should be disabled.

  Recommendation: If possible, use the setting "Basic256Sha256".

- Basic128Rsa15 -Sign
  Insecure end point, supports a series of algorithms that use the hash algorithm RSA15 and 128-bit encryption.
  This endpoint protects the integrity of the data through signing.
- Basic128Rsa15 -Sign & Encrypt
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 128-bit encryption.
  This endpoint protects the integrity and confidentiality of the data through signing and encrypting.

- Basic256Rsa15 -Sign
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 256-bit encryption.
  This endpoint protects the integrity of the data through signing.
- Basic256Rsa15 -Sign & Encrypt
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 256-bit encryption.
  This end point protects the integrity and confidentiality of the data through signing and encrypting.
- Basic256Sha256 - Sign
  Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.
  This endpoint protects the integrity of the data through signing.
- Basic256Sha256 - Sign & Encrypt
  Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.
  This endpoint protects the integrity and confidentiality of the data through signing and encryption.

To enable the security setting, click the check box in the relevant line.

Note

If you use the settings "Basic256Sha256 -Sign" and "Basic256Sha256 -Sign & Encrypt", the OPC UA server and OPC UA clients must use "SHA256"-signed certificates.

For the settings "Basic256Sha256 -Sign" and "Basic256Sha256 -Sign & Encrypt", the certificate authority of STEP 7 automatically signs the certificates with "SHA256".

"No Security" security policy and authentication via user name and password

You can set the following combination:

Security policy = "No Security" and authentication via user name and password.

- The OPC UA server of the S7-1500 supports this combination. OPC UA clients can connect and encrypt the authentication data or not.
- OPC UA client of the S7-1500 CPU also supports this combination: However, in runtime it only connects if it can send the authentication data encrypted via cable!

Handling of the client certificates of the S7-1500 CPU

Where does the client certificate come from?

If you are using the OPC UA client of an S7-1500 CPU (OPC UA client enabled), you can create certificates for these clients with STEP 7 V15 and higher as described in the following sections.

When you use UA clients from manufacturers or the OPC Foundation, a client certificate is generated automatically during installation or upon the first program call. You have to import these certificates with the global certificate manager in STEP 7 and use them for the respective CPU.

When you program an OPC UA client yourself, you can have the certificates generated by the program; see the section "Instance certificate for the client". Alternatively, you can generate certificates with tools, for example with OpenSSL or the certificate generator of the OPC Foundation:

- The procedure for OpenSSL is described here: "Generating PKI key pairs and certificates yourself".
- Working with the certificate generator of the OPC Foundation is described here: "Creating self-signed certificates".

Certificate of the OPC UA client of the S7-1500 CPU

A secure connection between the OPC UA server and an OPC UA client is only established if the server classifies the certificate of the client as trusted.

Therefore you have to make the client certificate known to the server.

The following sections describe how you can initially generate a certificate for the OPC UA Client of the S7-1500 CPU and then make it available to the Server.

1. Generate and export a certificate for the client

For a secure connection you have to generate a client certificate and export the certificate.

To do this, follow these steps:

1. In the "Project tree" area, select the CPU you want to use as a client.
2. Double-click "device configuration".
3. In the properties of the CPU click "Protection & Security > Certificate manager".
4. Double-click "<add>" in the "Device certificates" table.
   STEP 7 opens a dialog.
5. Click "Add".
6. Select the "OPC UA client" entry from the list at the "Usage".
   Attention:
   Under "Alternative name of the certificate holder (SAN)" the IP addresses under which the CPU in your system can be accessed has to be entered.
   You must therefore configure the IP interface of the CPU before you have generate a Client certificate.
7. Click "OK".
   STEP 7 now lists the client certificate in the "Device certificates" table.
8. Right-click this row and select the "Export certificate" entry from the shortcut menu.
9. Select a directory in which you store the Client certificate.

2. Announcing the Client certificate to the server

You have to make the Client certificate available to the server to allow a secure connection to be established.

To do this, follow these steps:

1. Select the "Use global security settings for certificate manager" option in the local certificate manager of the server. This makes the global certificate manager available.
   You will find this option under "Protection & Security > Certificate manager" in the properties of the CPU that is acting as server.
   If the project is not yet protected, select "Security settings > Settings" in the STEP 7 project tree, click the "Protect this project" button and log on.
   The "Global security settings" item is now displayed under "Security settings" in the STEP 7 project tree.
2. Double click "Global security settings".
3. Double click "Certificate manager".
   STEP 7 opens the global certificate manager.
4. Click on the "Trusted certificates" tab.
5. Right-click in the tab on a free area (not on a certificate).
6. Select the "Import" command from the shortcut menu.
   The dialog for importing certificates is displayed.
7. Select the client certificate that the server is to trust.
8. Click "Open" to import the certificate.
   The certificate of the client is now contained in the global certificate manager.
   Note the ID of the client certificate just imported.
9. Click the "General" tab in the properties of the CPU that is acting as server.
10. Click "OPC UA > Server > Security > Secure Channel".
11. Scroll down in the "Secure Channel" dialog to the section "Trusted clients".
12. Double-click in the table on the empty row with "<add new>". A browse button is displayed in the row.
13. Click this button.
14. Select the client certificate that you have imported.
15. Click the button with the green check mark.
16. Compile the project.
17. Load the configuration onto the S7-1500 CPU.

Result

The server now trusts the client. If the server certificate is also considered trusted, the server and client can establish a secure connection.

Generating server certificates with STEP 7

The description below shows the procedure for generating new certificates with STEP 7 and applies in principle to various uses of the certificates. STEP 7 sets the appropriate purpose - in this case "OPC UA Client & Server" - depending on which area of the CPU properties is used to start the following dialog.

Recommendation: To use the full functionality for the security of the OPC UA server, use the global security settings.

The global security settings are enabled in the CPU properties under "Protection & Security > Certificate manager".

Customizing server certificates

STEP 7 automatically generates a certificate for the OPC UA server of the S7-1500 when you activate the server (see "Activating the OPC UA server (Page 1160)"). In the process STEP 7 uses the default values for the parameters of the certificate. If you want to change the parameters, follow these steps:

1. Click the Browse button under "General > OPC UA > Server > Security > Secure channel > Server certificate" in the properties of the CPU. A dialog is displayed that shows the certificates available locally.
2. Click the "Add" button.

3. The dialog for generating new certificates is displayed (figure below). The values for an example are already entered:

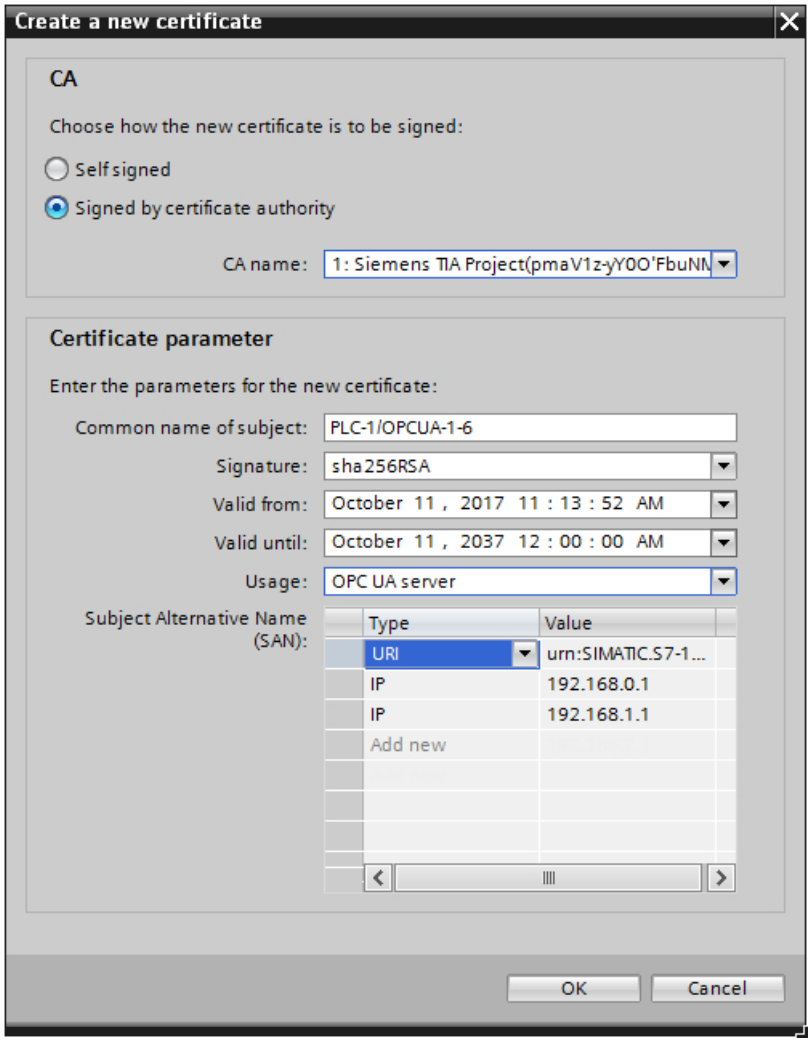

Figure 1-32 Customizing server certificates

4. Use other parameters if this is necessary in accordance with the security specifications in your company or your customer.

Explanation of fields for certificate generation

- CA
  Select whether the certificate is to be self-signed or signed by one of the CA certificates of the TIA Portal. The certificates are described under "Certificates with OPC UA". If you want to generate a certificate that is to be signed by one of the CA certificates of the TIA-Portal, the project must be protected and you must be logged in as a user with all the required function rights. Further information can be found under "Basics of user administration in the TIA Portal".
- Certificate holder
  The default setting always consists of the name of the project and "\OPCUA-1". In the example, the project name is "PLC1". In the properties of the CPU set the project name under "General > Project information" > Name". Keep the default or enter a different name that is more meaningful for the OPC-UA server under "Certificate holder".
- Signature
  Here you select the hash and encryption process that is to be used when signing the server certificate. The following entries are available:
  - "sha1RSA",
  - "sha256RSA".
- Valid from
  Here you enter the date and time for the beginning of the validity of the server certificate.
- Valid until
  Here you enter the date and time for the end of the validity of the server certificate. Ensure that the certificate is valid not only for one year or a few years. In the example the certificate is valid for 30 years. However, for reasons of security you should renew the certificate at much shorter intervals. The long period of validity gives you the opportunity to decide when a suitable moment would be, for example, when the system is being serviced.
- Usage
  The default is "OPC UA client & server". Keep this default for the OPC UA server. The "Generate new certificate" dialog can be called from several points in STEP 7. If, for example, you call this dialog for the Web server of the CPU, "Web server" is entered under "Application". The following entries are available in the Usage drop-down list:
  - "OPC UA client"
  - "OPC UA client & server"
  - "OPC UA server"
  - "TLS"
  - "Web server"
- Alternative name of the certificate holder
  The following is entered in the example above: "URI:urn:SIMATIC.S7-1500.OPC-UAServer:PLC1,IP:192.168.178.151,IP:192.168.1.1". The following entry would also be valid: "IP: 192.168.178.151, IP: 192.168.1.1". It is important that the IP addresses over which the OPC UA server of the CPU can be accessed are entered here (see "Access to the OPC UA server (Page 1162)"). This allows OPC UA clients to verify whether a connection to the OPC UA server of the S7-1500 is really to be established or whether in fact an attacker is trying to send manipulated values from another PC to the OPC UA client.

Editing the security settings of the OPC UA server.

OPC UA uses secure connections between client and server. OPC UA checks the identity of the communication partners. OPC UA uses certificates in accordance with X.509-V3 from the ITU (International Telecommunication Union) for client and server authentication.

Exception: A secure connection is not established with the "No security" security policy.

Configuring security settings of the server

The figure below shows the available server security settings for signing and encrypting messages.

The security policies are named according to the algorithms used. Example: "Basic256Sha256 - Sign & Encrypt" means: Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.

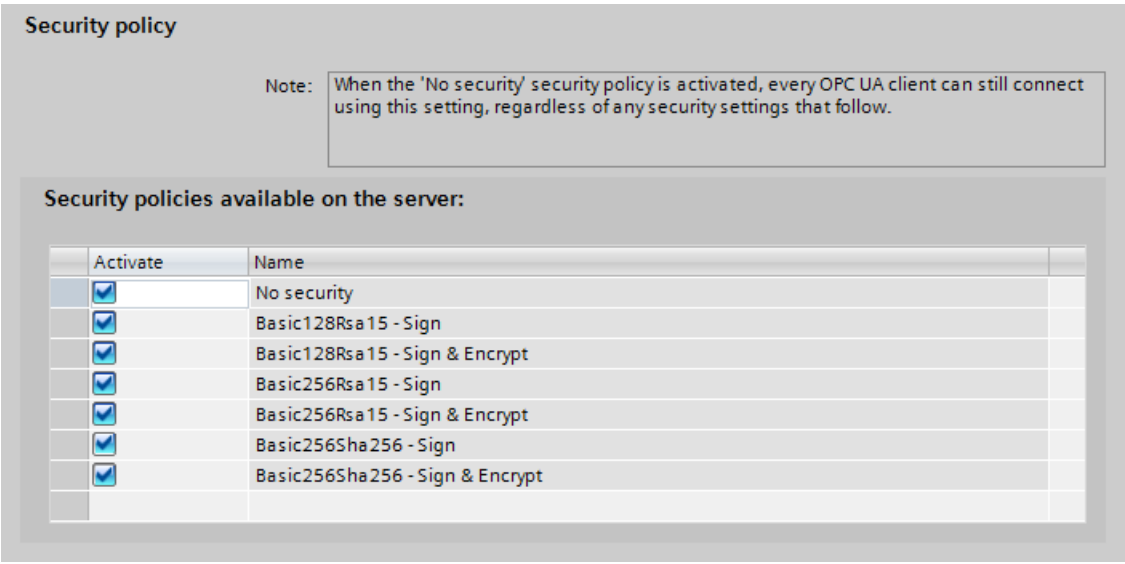

Figure 1-33 Configuring security settings of the server

By default, a server certificate is created that uses SHA256 signing. The following security policies are released:

- None
  Insecure end point

  Note

  Disabling security policies you do not want

  If you have selected all security policies (default setting) in the secure channel settings of the S7-1500 OPC UA server – in other words the end point "No Security" – non-secure data traffic (neither signed nor encrypted) between the server and client is also possible. The identity of the client remains unknown with "No security". Each OPC UA client can then connect to the server irrespective of any subsequent security settings.

  When configuring the OPC UA server, make sure that only security policies that are compatible with the security concept for your machine or plant are selected. All other security policies should be disabled.

  Recommendation: If possible, use the setting "Basic256Sha256".

- Basic128Rsa15 -Sign
  Insecure end point, supports a series of algorithms that use the hash algorithm RSA15 and 128-bit encryption.
  This endpoint protects the integrity of the data through signing.
- Basic128Rsa15 -Sign & Encrypt
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 128-bit encryption.
  This endpoint protects the integrity and confidentiality of the data through signing and encrypting.
- Basic256Rsa15 -Sign
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 256-bit encryption.
  This endpoint protects the integrity of the data through signing.
- Basic256Rsa15 -Sign & Encrypt
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 256-bit encryption.
  This end point protects the integrity and confidentiality of the data through signing and encrypting.
- Basic256Sha256 - Sign
  Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.
  This endpoint protects the integrity of the data through signing.
- Basic256Sha256 - Sign & Encrypt
  Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.
  This endpoint protects the integrity and confidentiality of the data through signing and encryption.

To enable the security setting click the check box in the relevant line.

Note

If you use the settings "Basic256Sha256 -Sign" and "Basic256Sha256 -Sign & Encrypt", the OPC UA server and OPC UA clients must use "SHA256"-signed certificates.

For the settings "Basic256Sha256 -Sign" and "Basic256Sha256 -Sign & Encrypt", the certificate authority of STEP 7 automatically signs the certificates with "SHA256".

"No Security" security policy and authentication via user name and password

You can set the following combination:

Security policy = "No Security" and authentication via user name and password.

- The OPC UA server of the S7-1500 supports this combination. OPC UA clients can connect and encrypt the authentication data or not.
- OPC UA client of the S7-1500 CPU also supports this combination: However, in runtime it only connects if it can send the authentication data encrypted via cable!

See also

User authentication (Page 1179)

User authentication

Types of user authentication

For the OPC UA server of the S7-1500, you can set what authentication is required for a user of the OPC UA client wishing to access the server.

You have the following options:

- **Guest authentication**
  The user does not have to prove their authorization (anonymous access). The OPC UA server does not check the authorization of the client user
  If you want to use this type of user authentication, select the "Enable guest authentication" option under "OPC UA > Server > Security > User authentication".

  Note

  To increase security, you should only allow access to the OPC UA server with user authentication.

- **User name and password authentication**
  The user has to prove their authorization (no anonymous access). The OPC UA server checks whether the client user is authorized to access the server. Authorization is given by the user name and the correct password.
  If you want to use this type of user authentication, select the "Enable user name and password authentication" option under "OPC UA > Server > Security > User authentication".
  Deactivate the guest authentication.
  Enter the user in the "User management" table.
  To do so, click the "<Add new user>" entry. A new user is created with an automatically assigned name. You can edit the user name and enter the password for the user name.
  You can add a maximum of 21 users.
- **Additional user administration via the security settings of the project**
  If you select this option, the user management for the open project will also be used for user authentication for the OPC UA server: The same user names and passwords are then valid in OPC UA as in the current project.
  Proceed as follows to activate user management for the project:
  - Click "Security settings > Settings" in the project tree.
  - Click the "Protect this project" button.
  - Enter your user name and your password.
  - Enter additional users under "Security settings > Users and roles".
- If you configure an additional OPC UA server in your project, also select the option "Enable additional user administration via the security settings of the project". Repeated input of user names and passwords is then unnecessary.

See also

Editing the security settings of the OPC UA server. (Page 1177)

User authentication (Page 1237)

Users and roles with OPC UA function rights (Page 1181)

Users and roles with OPC UA function rights

The following options for user authentication use central project settings for project users:

- For the server:
  For configuration of CPU properties (OPC UA > Server > Security > User authentication). Option: "Enable additional user administration via the security settings of the project"
- For the client:
  For configuration of client interface ("Configuration" tab, "Security"). Option: "User (TIA Portal - security settings)"

Requirement

Before you can edit the security settings, the project must be protected and you must be logged on with sufficient rights, for example as administrator.

Settings in the project tree > "Security settings"

You access the central user settings and roles in the protected project in the project tree under "Security settings". This is where you centrally define users with user name, password and function rights. You can simply use these settings elsewhere.

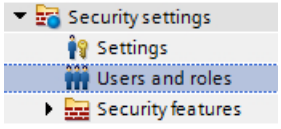

Reusing central security settings

Examples for reusing elsewhere:

- User selection for user authentication for OPC UA server
  With this setting, you tell the server which client (user) with which user name and which password is allowed to access the server.
- User selection for OPC UA client authentication
  With this setting, you tell the client the user name and password that it is to use for client authentication for the server.

The settings for the client and server must correspond: The user name and password used by the client to log on must have been set up on the server and assigned the required authorizations.

Function rights for server and client

The corresponding function rights for the client or the server must also be enabled for users of the client function and users of the server function on an S7-1500 CPU. It is not enough simply to save the user name and password centrally.

Here is an example to illustrate this type of rights use.

1. Under "Security settings > Users and roles", you define a new role in the "Roles" tab with the name "PLC-opcua-role-all-inclusive", for example.
   Tip: The tab may be covered by an information window ("The current status has not yet been checked..."). In this case, first close the information window.
2. In the "Categories of function rights" section, you navigate to the runtime rights and then to the CPU function rights, and select the CPU whose function rights you want to set, for example PLC_2.
3. You will find the following function rights in the "Function rights" section:
   – OPC UA server access
     This function right apples on the OPC UA server of the S7-1500 CPU. Only when this option is selected does a user of the CPU PLC_2 server who has been assigned the role "PLC-opcua-role-all-inclusive" have the following right: For the establishment of a session with the server, the user requires client authentication with one of the user names and corresponding passwords that have been centrally defined (and loaded to the CPU).
   – User authentication of the OPC UA client
     This function right apples on the OPC UA client of the S7-1500 CPU (with client instructions). Only when this option is selected can the user of the client of CPU PLC_2 who has been assigned the role "PLC-opcua-role-all-inclusive" use the user name and password for authentication to establish a session with a server.

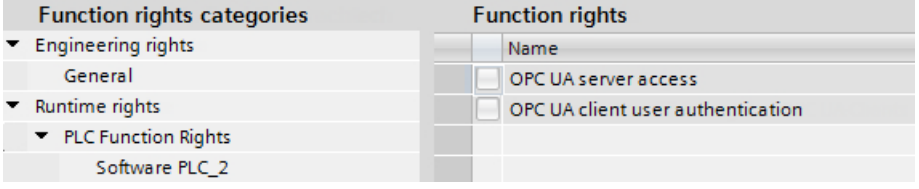

4. The role "PLC-opcua-role-all-inclusive" still needs to be assigned to the relevant users ("Users" tab under "Security settings" in the project tree).

See also

User authentication (Page 1179)

User authentication (Page 1237)

Licenses for the OPC UA server

Runtime licenses

A license is required to run the OPC UA server of the S7-1500 CPU. The type of license required depends on the performance of the respective CPU. The following license types are differentiated:

- SIMATIC OPC UA S7-1500 small (required for CPU 1511, CPU 1512, CPU 1513, ET 200SP CPUs, CPU 1515SP PC)
- SIMATIC OPC UA S7-1500 medium (required for CPU 1515, CPU 1516, Software Controller CPU 1507, CPU 1516pro-2PN)
- SIMATIC OPC UA S7-1500 large (required for CPU 1517, CPU 1518)

The required license type is displayed under "Properties > General > Runtime licenses > OPC-UA > Type of required license":

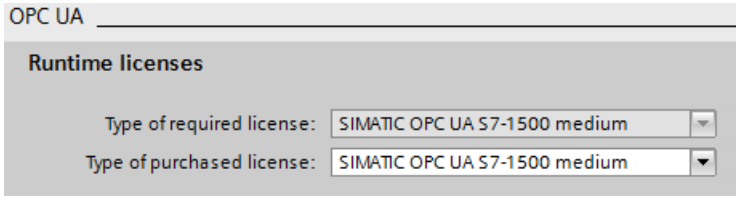

To confirm purchase of the required license, follow these steps:

1. Click "Runtime licenses > OPC UA" in the properties of the CPU.
2. Select the required license from the "Type of purchased license" drop-down list.

Providing methods on the OPC UA server

Important facts about server methods

Providing user program for server methods

On the OPC UA server of an S7-1500 CPU (as of firmware V2.5), you have the option of providing methods via your user program. These methods can be used by OPC UA clients, for example to start a manufacturing job using the method call of the S7-1500 CPU.

OPC UA methods, an implementation of "Remote Procedure Calls", provide an efficient mechanism for interactions between different communication nodes. The mechanism provides both job confirmation and feedback values so you no longer have to program handshaking mechanisms.

Using OPC UA methods, you can transfer data consistently without trigger bits/handshaking, for example, or trigger specific actions on the controller.

How does an OPC UA method work?

An OPC UA method in principle operates like a know-how protected function block that is called by an external OPC UA client in runtime.

The OPC UA client only "sees" the defined inputs and outputs. The content of the function block, the method or algorithm, remains hidden to the external OPC UA client. The OPC UA client receives feedback on successful execution and values returned by the function block (method), or an error message if execution has not been successful.

As the programmer, you have full control over and responsibility for the program context in which the OPC UA method runs.

Rules for programming a method and runtime behavior

- Make sure that the values returned by the OPC UA method are consistent with the input values provided by the OPC UA client.
- Follow the rules on assigning name and the structure of parameters, and the permitted data types (see description of the OPC UA server instructions).
- Behavior during runtime: The OPC UA server accepts **one** call per instance. The method instance is not available for other OPC UA clients until the call has been processed by the user program or has timed out.

The basic procedure for implementing a user program as a server method is set out below.

Implementing a server method

A program (function block) for implementing a server method is structured as follows:

1. **Querying the server method call with OPC_UA_ServerMethodPre**
   You first call the "OPC_UA_ServerMethodPre" instruction in your user program (i.e. in your server method).
   This instruction has the following tasks:
   - With this instruction you ask the OPC UA server of the CPU whether your server method was called from an OPC UA client.
   - If the method was called and the server method has input parameters, your server method now receives the input parameters.
     The input parameters of the server method come from the calling OPC UA client.
2. **Editing the server method**
   In this section of the server method, you provide the actual user program.
   You have the same options as in any other user program (for example, access to other function blocks or global data blocks).
   If the server method uses input parameters, these parameters are available to you.
   This section of the server method should only be executed if an OPC UA client has called the server method.
   After successful execution of the method, you set the output parameters of the server method if the method has output parameters.
3. **Responding to server method with OPC_UA_ServerMethodPost**
   To complete the server method, call the "OPC_UA_ServerMethodPost" instruction.
   Use the parameters to notify the "OPC_UA_ServerMethodPost" instruction whether or not the user program has been processed.
   If the user program has been successfully executed, the OPC UA server is notified via the relevant parameters. The OPC UA server then sends the output parameters of the server method to the OPC UA client.

Always call the instructions "OPC_UA_ServerMethodPre" and "OPC_UA_ServerMethodPost" as a pair irrespective of whether the user program is processed by both instructions or continued in the next cycle.

You will find an example of a server method implementation in the STEP 7 online help.

Integrating the server method

The diagram below shows how an OPC UA client (A) calls the server method "Cool":

The CPU executes the instance "Cool1" of the server method "Cool" in the cyclic user program ⑥.

The CPU first uses the instruction "OPC_UA_ServerMethodPre" to query ④ whether an OPC UA client has called the server method "Cool" ①.

- If the server method has not been called, program execution returns directly to the cyclic user program over ④ and ⑥. The CPU resumes the cyclic user program after "Cool1".
- If the server method has been called, this information is returned to the server method "Cool" over ④. The actual functionality is now executed in the Cool server method, see"<Method Functionality>" in the graphic.
  The server method then uses the instruction "OPC_UA_ServerMethodPost" ⑤ to notify the firmware (B) that the instruction has been executed ③.
  The firmware returns this information over ② to the calling OPC UA client (A).
  The CPU resumes the cyclic user program after "Cool1".

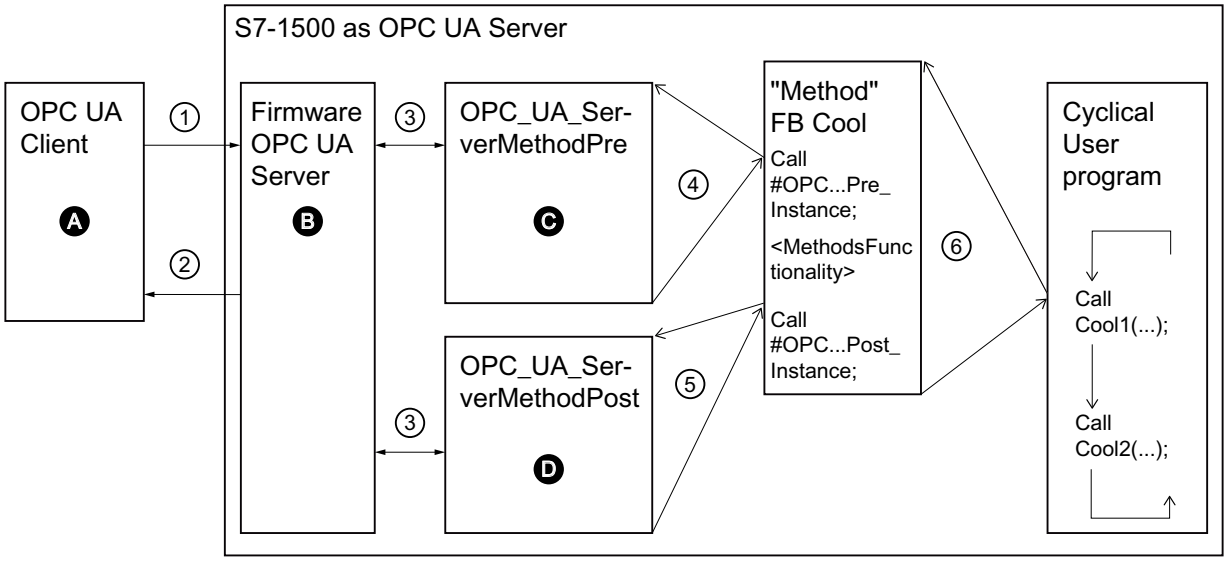

A	Call of the server method and management of the "Done" information (method complete)
①	Asynchronous call of the server method
②	Asynchronous "Done" information for the method called (method complete)
B	Wait for OPC UA client calls, management of calls in the queue, forwarding of "Done" information from the cyclic user program to the OPC UA client
③	Data transfer from the OPC UA server to the method instances of the user program and vice versa
C	Check whether method has been called. If it has, forwarding of input data from the OPC UA server to the method instance of the user program and feedback to the method instance that the method has been called ("called")
④	Synchronous call of the instruction OPC_UA_ServerMethodPre as a multi-instance stating the storage area for the input data from the OPC UA server. The return value indicates whether or not the method has been called by the OPC UA client.
⑤	Check whether the method has been completed or is still active ("busy").
D	Check whether the method has been completed. If it has, the output data of the method instance is forwarded to the OPC UA server and the method instance is notified that the method has been completed. The OPC UA server is notified.
⑥	Call of the method FB (in this case: FB Cool) with the required instance and the process parameters

Information about server instructions

The "OPC_UA_ServerMethodPre" and "OPC_UA_ServerMethodPost" are described in detail in the help to the Instructions > Communication > OPC UA > OPC UA server.

See also

Boundary conditions for using server methods (Page 1187)

Boundary conditions for using server methods

Permitted data types

If you provide server methods, observe the following rule:

- Assign the data types as shown below (SIMATIC data type - OPC UA data type). Other assignments are not permitted.

STEP 7 does not check the observance of this rule and does not prevent an incorrect assignment. You are responsible for the rule-compliant selection and assignment of the data types.

You can also use the listed data types, for example, as elements of structures/UDTs for input and output parameters of self-created server methods (UAMethod_InParameters and UAMethod_OutParameters).

SIMATIC data type	OPC UA data type
BOOL	Boolean
SINT	SByte
INT	Int16
DINT	Int32
LINT	Int64
USINT	Byte
UINT	UInt16
UDINT	UInt32
ULINT	UInt64
REAL	Float
LREAL	Double
LDT	DateTime
WSTRING	String
DINT	Enumeration (Encoding Int32) and all derived data types
User-defined data type required (UDT, user-defined data type) The user-defined data type must be created with the prefix "Union_", for example "Union_MyDatatype". The first element (Selector) in this UDT must have the data type "UDINT".	UNION and all derived data types

Number of implementable server methods and number of arguments

If you implement server methods via your user program, the number of usable methods is limited depending on the CPU type, see the following table.

Technical specification value	CPU 1510SP (F) CPU 1511 (C/F/T/TF) CPU 1512C CPU 1512SP (F) CPU 1513 (F)	CPU 1505 (S/SP/SP F/SP T/SP TF) CPU 1515 (F/T/TF) CPU 1515 SP PC (F/T/TF) CPU 1516 (F/T/TF)	CPU 1507S (F) CPU 1517 (F/T/TF) CPU 1518 (F)
Maximum number of usable server methods or max. number of server method instances (OPC_UA_ServerMethodPre, OPC_UA_ServerMethodPost instructions)	20	50	100
Maximum number of arguments per method (More than the specified number of arguments can be configured and loaded into the CPU, but an OPC UA client cannot call the method).	20	20	20

Error message when exceeded

If the maximum number of server methods is exceeded, the OPC_UA_ServerMethodPre or OPC_UA_ServerMethodPost instructions report the error code 0xB080_B000 (TooManyMethods).

Supply of structured data types with nested arrays

If a structured data type (Struct/UDT) contains an array, the OPC UA server does not provide information about the length of this array.

If you use such a structure as the input or output parameter of a server method, for example, you must ensure that the nested array is supplied with the correct length when the method is called.

If you do not adhere to this rule, the method fails with the error code "BadInvalidArgument".

OPC UA server interface configuration

What is a server interface?

Definition

A server interface combines related nodes of an OPC UA address space of a CPU into a unit, so that a specific view of this CPU is provided for OPC UA clients.

1st example:

A CPU controls an injection molding machine.

In this example, a server interface contains all OPC UA nodes of the CPU,

- which you can read with an OPC UA client, to get information about this injection molding machine (from readable PLC tags),
- which you can write with an OPC UA client write, to transfer values to the injection molding machine (in writable PLC tags),
- which you can call with an OPC UA client to start injection molding machine functions (via server methods).

This server interface allows a view of a CPU which can be used to control an injection molding machine.

For injection molding machines, the companion specification "Euromap" defines a whole series of OPC UA nodes that you can combine in a server interface.

Other OPC UA nodes of the CPU are not included in this view.

This provides a better overview.

The section "Create server interface (Page 1189)" shows how to create a server interface and add related OPC UA nodes to this server interface.

The "Euromap77" serves as an example for related OPC UA nodes.

2. Example

Plant-specific or customer-specific tasks

You can also define a server interface which, for example, meets the requirements in a customer-specific project.

Specify all required OPC UA nodes in an XML file.

Use a tool for this purpose, for example the free program "SiOME" from Siemens; you can find the explanations and download link here (https://support.industry.siemens.com/cs/ww/en/view/109755133).

Create a server interface

Create server interfaces

To create an Server interface, follow these steps:

1. Select the project view in the TIA Portal.
2. Select the CPU that you want to use as an OPC UA server.
3. Click "OPC UA communication > Server interfaces".
4. Double-click "Add new server interface".
   STEP 7 creates a new server interface, names it "Server interface_1" and opens the editor for server interfaces.
5. Change the name of the new server interface so that it is descriptive in your project.

6. Click the "Import companion specification" button in the top right of the Server Interfaces view.
   STEP 7 displays the "Import" dialog.
7. Select an XML file which contains an instance (or multiple instances) of a companion specification.
   The "Using OPC UA companion specification (Page 1192)" section describes how to create such an XML file.
   STEP 7 imports the XML file and displays the new server interface in the editor.
   The following figure shows a section of a server interface which uses an instance of the "Euromap" companion specification:

Export interface | Consistency check | Update interface | Import companion specification

OPC UA server interface

	Name of the node	Node type	Access level	Description		Local data
1	IMM_manufacturer_01234	Object				
2	InjectionUnits	Object				
3	InjectionUnit_1	Object				
4	TemperatureZones	Object				
5	BarrelTemperatureZone_1	Object				
6	ActualTemperature	Double	RD	The ActualTemperat...		"InjectionUnits"."i.
7	EngineeringUnits	EUInformation	RD	Show Placeholde		
8	EURange	Range	RD	Range for this control		
9	NominalTemperature	Double	RD	The nominal Placeh...		"InjectionUnits"."i.
10	EngineeringUnits	EUInformation	RD			
11	EURange	Range	RD			
12	Index	UInt32	RD	The Index Property g...		"InjectionUnits"."i.
13	IsPresent	Boolean	RD	TRUE if present		"InjectionUnits"."i.
14	Name	String	RD	The Name Property r...		"InjectionUnits"."i.
15	TemperatureConfiguration	TemperatureConfig...	RD	The TemperatureCo...		"InjectionUnits"."i.

Information on the server interface

The editor for server interfaces is structured as table and provides a range of information about the imported node set file (OPC UA-XML file).

The tables provides the following information:

- Name of the node
  The top node is named "IMM_manufacturer_01234" in the example.
  This name stands for the injection molding machine as a whole. It is the name of the instance of the "Euromap" companion specification that is used here.
  According to the companion specification, the instance name should begin with "IMM", followed by the name of the manufacturer of the injection molding machine; the serial number of the machine is added to the end. This allows a unique identification of the machine.
  The name of this node is assigned by the user.
  The names of all other (lower-level) nodes are defined by the specification (in the example above by the "Euromap" companion specification). These node names must not be changed by the user. This ensures a uniform view of all injection molding machines, which complies with the specification.
- Node type
  Type of the node. The type is specified by the companion specification that is used.
  STEP 7 shows a node type in red in the table when the imported XML file contains no type definition.
  In this case, you can find the referenced but missing namespaces in the properties of the server interface under "Namespaces".
  Import the referenced XML files that contain the missing namespaces (and all data types defined in them) into a server interface to be created.
- Access level
  Nodes (tags) can only be readable (RD) or readable and writeable (RD/WR).
  The server methods of a specification can always be called. If they were not callable, they would not have been included in the specification.
- Description
  The imported XML file may contain descriptions of the nodes in several languages.
  If you have imported such a file, STEP 7 displays the description in the project language.
  If no description is available in the XML file in the project language, then the English description is displayed.
- Local data
  STEP 7 displays the data block which is assigned to the node in the CPU (mapping).
  The CPU reads the value of the OPC UA node from this data block.
  The OPC UA server then makes this value available for OPC UA clients.
  If a data block is displayed in red, then the specified data block is not available in the CPU.
  In this case you have to create the missing data block in the CPU (of user program) and supply it with a value.
  Alternative: Change the mapping of the OPC UA node to a local data block (or to a tag of a tag table of your user program) in the XML file.
  The "Using OPC UA companion specification (Page 1192)" section describes how to change mapping.
- Data type
  This is the data type of the data block in the CPU, from which the value of an OPC UA node is read, or to which a value is written.

Consistency check

You have the option of checking the server interface. STEP 7 hereby checks whether the nodes (e.g. tags) are mapped correctly in the imported node set file (OPC UA XML file).

1. Select the nodes that you want to check.
2. Click "Consistency check".

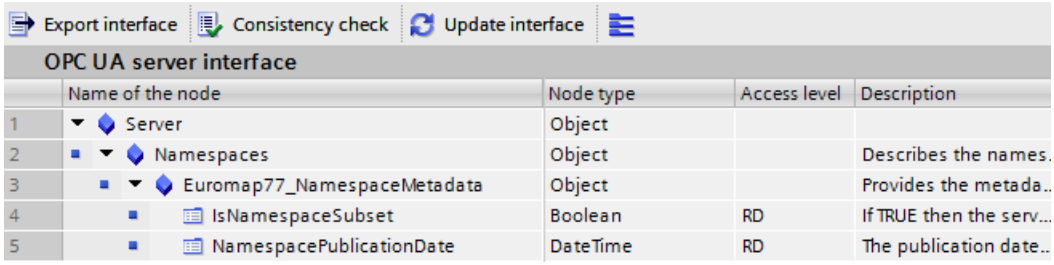

The consistency check checks the mapping of the local data against the displayed server interface:

– Do the names of the PLC tags in the CPU program correspond to those of the imported server interface?

– Do the data types of the PLC tags correspond to the corresponding nodes in the server interface? STEP 7 displays corresponding warnings and errors if the above conditions are not met.

Using OPC UA companion specifications

Introduction

OPC UA is universally applicable: The standard itself does not, for example, specify how PLC tags are to be named. It is also up to the individual user (application developer) to program and name server methods that can be called over OPC UA.

Information modeling and standardization for devices and sectors

For applications of the same kind, it is worth standardizing your device or machine interface with the "OPC UA toolkit".

Many different bodies and working groups have driven forward standardization and developed a range of companion specifications.

These specifications define:

- The objects, methods and tags with which a typical device or machine is to be described.
- The namespace intended for the specified objects.

Machines are typically structured in functional or technological units, and these units are then standardized.

Companion specifications offer machine and plant operators the benefits of a standardized interface. For example, all RFID readers that comply with the AutoID specifications can be integrated in the same way. This means that all RFID readers that comply with the AutoID

specifications can be addressed by OPC UA clients in the same way irrespective of manufacturer.

Another example of companion specifications is the Euromap 77 Companion Specifications from the injection molding machinery sector.

The following section uses the example of Euromap 77 to detail how to apply companion specifications in STEP 7 (TIA Portal) and create the necessary PLC tags.

Example: Euromap 77

Euromap 77 standardizes the exchange of data between injection molding machines and the higher-level MES (manufacturing execution system). This allows the MES to connect all lower-level injection molding machines in the same way.

The standardized data interface facilitates the integration of injection molding machines into a plant.

Using companion specifications: Overview

Euromap 77 is described in the OPC UA XML file "Opc_Ua.EUROMAP77.NodeSet2.xml".

Note

Euromap 77, Euromap 83 and OPC UA for Devices (DI)

With Release Candidate 2, some of the Euromap definitions have been transferred from Euromap 77 to Euromap 83. You will therefore also need to import the OPC UA server interface of Euromap 83.

"OPC UA for Devices" is a generally applicable information model for the configuration of hardware and software components. The information model also serves as the basis for other companion standards and is therefore also imported.

The OPC UA XML files are available here:

Euromap77 (http://www.euromap.org/euromap77)

Euromap83 (http://www.euromap.org/euromap83)

OPC UA for Devices (https://opcfoundation.org/UA/schemas/DI/)

The Euromap OPC UA XML files define the interface of the OPC UA server of an injection molding machine that complies with the Euromap 77 companion standard.

Proceed as follows to model the OPC UA server of the S7-1500 CPU in accordance with the Euromap 77 standard:

1. Generate an XML file by creating an instance of the type "IMM_MES_InterfaceType".
   "IMM_MES_InterfaceType" is the highest-level node in Euromap 77: This data type is directly derived from the OPC UA data type BaseObjectType".
   All Euromap 77 (83) tags and methods are defined under "IMM_MES_InterfaceType".
   "Step 1" sets out the procedure in detail.
2. Assign PLC tags and FB instances (server methods) from your S7-1500 CPU to the tags and methods of Euromap 77 (the Euromap 77 information model).
   "Step 2" sets out the procedure in detail.

3. Import the XML file as "Server interface".
   "Step 3" sets out the procedure in detail.
   Compile the STEP 7 project.
   Download the project to the CPU that acts as the controller for an injection molding machine.
4. In your STEP 7 project, create the PLC tags and server methods to which you have assigned Euromap 77 tags and methods under 2.
   The PLC tags must have compatible data types; see "Mapping data types".
   Enable read and write access to these PLC tags for OPC UA clients in accordance with Euromap 77.
   Save the PLC tags in a data block, for example.
   "Step 4" sets out the procedure in detail.

Result: The Euromap 77 tags and server methods are available for OPC UA clients in the address space of the OPC UA server of your CPU.

Step 1: Creating an instance

The following section describes how to use the free program "SiOME", the "Siemens OPC UA Modelling Editor".

With SiOME, you can create an OPC UA XML file, which describes the server interface (an information model).

To load and publish the new interface in the OPC UA server of an S7-1500 CPU, import the server interface into the STEP 7 project, see section "Create a server interface (Page 1189)".

When the project is loaded into the CPU, the new server interface is available for OPC UA clients.

Other tools for designing information models

You can also work with other tools to design information models to create a server interface, such as the program "UaModeler" from Unified Automation. You can download the "UaModeler" program from here (https://www.unified-automation.com/downloads/opc-ua-development.html).

The tools for designing information models are constantly being honed and improved. You should therefore always use the documentation provided by the manufacturer.

SiOME

With the help of SiOME (Siemens OPC UA Modeling Editor), a tool for implementing OPC UA companion specifications, you can design information models / address spaces for your OPC UA server and can create new types and instances of OPC UA nodes.

You can also use SiOME to map UA tags to PLC tags and UA methods to PLC function blocks (instances).

Download link and explanations about SIOME are available here (https://support.industry.siemens.com/cs/ww/en/view/109755133).

Proceed as follows to create an XML file with an instance of "IMM_MES_InterfaceType":

1. Download the files "Opc_Ua.EUROMAP77.NodeSet2.xml" and "Opc_Ua_EUROMAP83_NodeSet2.xml" from the Euromap website (see above).
2. Download the file "Opc.Ua.Di.NodeSet2.xml" from the OPC Foundation website. Euromap77/83 is an OPC UA Companion specification, this means that it is based on OPC UA and uses data types defined in the file "Opc.Ua.Di.NodeSet2.xml".
3. Start SiOME.
4. First, import the namespace "http://opcfoundation.org/UA/DI/".
   To do so, click the ""Import XML" button in the "Information model" area.

   Select the file "OPCUaDiNodeSet2 "xml" and click "Open" to import the file.
   SiOME imports the XML file and shows the namespace "http://opcfoundation.org/UA/DI/" in the "Namespaces" area.
   The standard namespace "http://opfoundation.org/UA/" is always available in SiOME and does not have to be imported.
5. Now import the namespace "http://www.euromap.org/euromap83/"
   To do so, click the ""Import XML" button in the "Information model" area.
   Select the file "Opc_Ua.EUROMAP83.NodeSet2.xml".
   SiOME imports the XML file and shows the namespace "http://www.euromap.org/euromap83/" in the "Namespaces" area.
6. Now import the namespace "http://www.euromap.org/euromap77/"
   To do so, click the ""Import XML" button again in the "Information model" area.
   Select the file "Opc_Ua.EUROMAP77.NodeSet2.xml".
7. **Defining your own namespace and creating an instance**
   So far, you have imported Euromap77.
   Now use the companion specification for an injection molding machine, i.e. you create an instance of Euromap77 (a usage).
   In the "Namespaces" area, right-click "Models" and select "Add Model".
   SiOME opens the "Add Model" dialog.
8. Click "New Model", to create a new model.
   SiOME now shows in green the switch in front "New Model":

   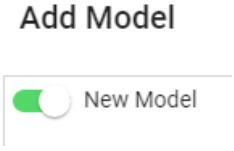

9. Add your own namespace to the XML file.
   Use a unique name for this purpose.
   In the example, enter "YourComany.com":

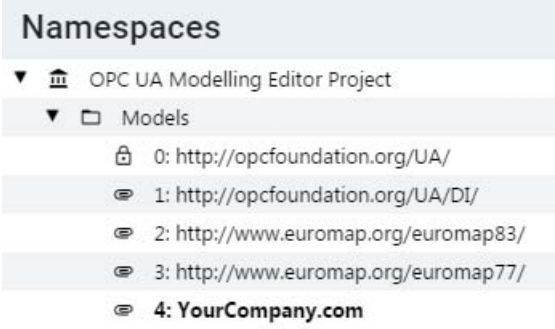

10. **Create new sequence**
    Now create now a new instance of Euromap77. This allows you to use the Euromap77 for an injection molding machine.
    Right-click on "Objects" in the "Information model" area.
    SiOME shows a shortcut menu.

11. Click "Add Instance" to add a new instance.
    SiOME displays the "Add Instance" dialog.
    At "Name" insert a descriptive name for your instance, i.e. for the machine you are using for Euromap77.
    In the example, enter "IMM_Manufacturer_01234".
    For "TypeDefinition", select "IMM_MES_InterfaceType".
    Click "OK".
    SiOME shows the new instance in the "Information model" area under "Objects":

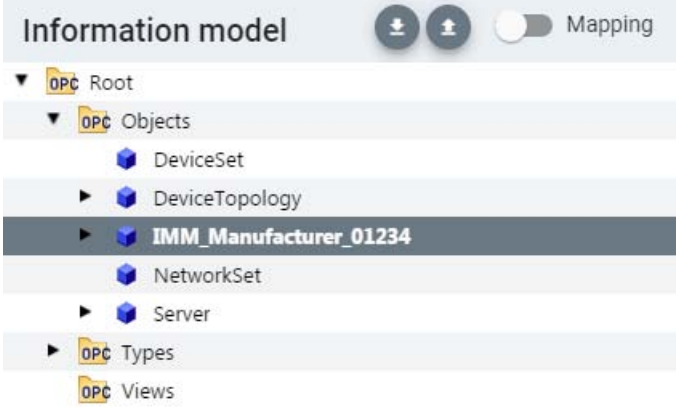

12. Click the black arrow in front of "IMM_Manufacturer_01234", to open the instance:

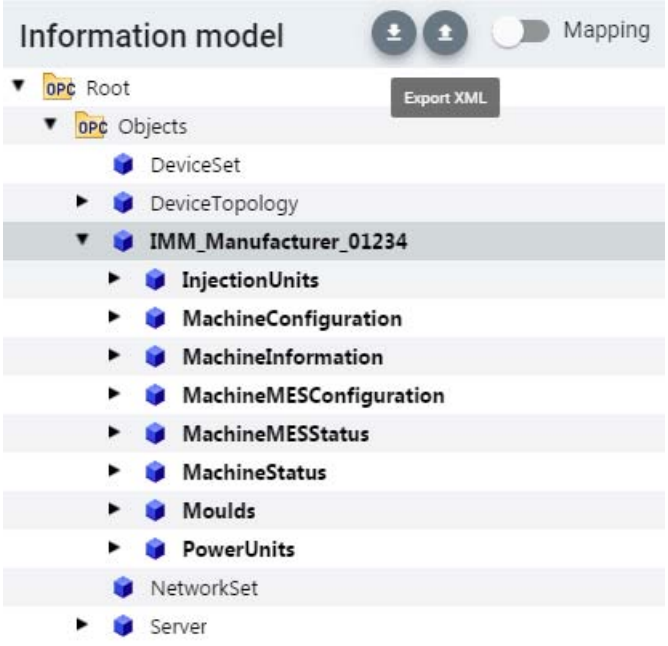

13. Now create a new instance for "InjectionUnits".
    Right-click "InjectionUnits".
    SiOME displays the "Add Instance" dialog.
    At "Name", insert a descriptive name for your instance.
    In the example, enter "InjectionUnit_1.
    For "TypeDefinition", select "InjectionUnitType".
    Click "OK".

14. Now create a new instance for "Molds" and "PowerUnits", as described above for "InjectionUnits".

15. Now save the XML file.
    Export the file for this purpose.
    Click the ""Export XML" button in the "Information model" area.

    SiOME shows the "Export XML" dialog.

16. Leave the "Include mappings" option unselected, because the file still contains no mappings (UAtags on PLC tags, UAmethods on server methods of the CPU).
    The imported namespaces are required for Euromap77. Therefore, SiOME displays these in grey.
    Click "OK".
    SiOME shows a dialog.

17. Select a descriptive name for the XML file and save the file.

Result:

You have now created an XML file containing the companion specification "Euromap77" and the instances you added. Euromap77 is used for this purpose.

Step 2: Assigning PLC tags

The description below uses the example of a tag to show how you assign specific PLC tags and server methods to the tags and methods of Euromap 77.

The easy way

We recommend using the SiOME tool here as well. In SiOME you have to enable mapping:

In the "TIA Portal" area, select the data blocks whose tags you want to assign ("map") to the OPC UA tags of the information model.

The tags from the TIA Portal project can easily be connected to the OPC UA tags using drag-and-drop operation.

To link methods, you must use a drag-and-drop operation to link the instance data blocks of the corresponding TIA Portal function blocks to the methods of the information model.

When exporting, select the "Include mappings" option, since the information model for an OPC UA server is required here.

This option is not required if you need the information model for an OPC UA client.

A detailed description can be found in the online support application example (SiOME (https://support.industry.siemens.com/cs/ww/en/view/109755133)).

Assigning PLC tags manually

If you do not assign the data with SiOME, proceed as follows:

1. Use an editor to open the file "demo.xml" that you created in "Step 1".
   In the XML file, search for the UA tag with
   NodeId="ns=1;s=MyIMM_MES_Interface.InjectionUnits.NodeVersion".
   These tags are assigned a PLC tag in the next step.
2. To assign this OPC UA tag a PLC tag, add an extension to the XML element of the OPC UA tag.
   In turn, you then add an element of the type "<si:VariableMapping>" to this extension:

```
<UAVariable DataType="String" ParentNodeId="ns=1;s=MyIMM_MES_Interface.InjectionUnits" NodeId=
"ns=1;s=MyIMM_MES_Interface.InjectionUnits.NodeVersion" BrowseName="NodeVersion">
    <DisplayName>NodeVersion</DisplayName>
    <References>
        <Reference ReferenceType="HasProperty" IsForward="false">
        ns=1;s=MyIMM_MES_Interface.InjectionUnits</Reference>
        <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
    </References>
    <Extensions>
        <Extension>
            <si:VariableMapping>"MyIMM_MES_Interface"."InjectionUnits.NodeVersion"
            </si:VariableMapping>
        </Extension>
    </Extensions>
    <Value>
</UAVariable>
```

3. The XML element "<si:VariableMapping>" is defined in the XML namespace "http://www.siemens.com/OPCUA/2017/SimaticNodeSetExtensions".
   You therefore need to assign this namespace to the XML element "<UANodeSet>", for example in a one-off operation with the following code line at the start of the XML file:

   ```
   <UANodeSet xmlns:si="http://www.siemens.com/OPCUA/2017/SimaticNodeSetExtensions"
   ```

   If you do not add the namespace, the element "<si:VariableMapping>" will not be known in the file.
4. Save the "demo.xml" file and close the editor.

Useful information: Assigning PLC methods

As well as assigning tags, you can also assign methods to an FB instance (user program or function block as representation of the method).

To assign an OPC UA method to an FB instance, you also need to add an extension to the OPC UA XML file following the procedure outlined below. Please note that the suffix ".Method" without quotation marks must be added to the instance name.

You add an element of the <si:MethodMapping> type to the extension:

The properties of an OPC UA method with the BrowseName "InputArguments" and "OutputArguments" are OPC UA tags and are **not** assigned.

```
<UAMethod ParentNodeId="ns=1;i=5003" NodeId=
"ns=1;s=MyIMM_MES_Interface.MachineConfiguration.SetServerTime" BrowseName="4:SetServerTime"
MethodDeclarationId="ns=2;i=7005">
    <DisplayName>SetServerTime</DisplayName>
    <References>
        <Reference ReferenceType="HasProperty">
        ns=1;s=MyIMM_MES_Interface.MachineConfiguration.SetServerTime.InputArguments</Reference>
        <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=5003</Reference>
    </References>
    <Extensions>
        <Extension>
            <si:MethodMapping>"MachineConfiguration.SetServerTime_IDB".Method</si:MethodMapping>
        </Extension>
    </Extensions>
</UAMethod>
```

Step 3: Importing server interfaces

Proceed as follows to import an OPC UA XML file as "Server interface":

1. Open the STEP 7 project.
2. Click "OPC UA communication > Server interfaces".
3. Double-click on "Import server interface".
4. In the "Import" dialog, select the file that you want to import as server interface.
   In the example, this is the file "demo.xml".

5. Click the "Import" button.
6. Also import the files "Opc.Ua.Di.NodeSet2.xml", "Opc_Ua.EUROMAP77.NodeSet2.xml" and "Opc_Ua.EUROMAP83.NodeSet2.xml".
   The "demo.xml" file relates to the files just specified.
   Always use the latest versions of these files as STEP 7 does not compile OPC UA XML files with errors.

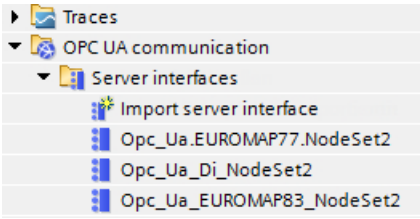

The above image of the machine generated in accordance with Euromap 77 specifications is now available in the address space of the OPC UA server.

Step 4: Creating PLC tags and server methods in the STEP 7 project

In the example, you have assigned the PLC tag "MyIMM_MES_Interface"."InjectionUnits.NodeVersion" to the UA tag with NodeId="ns=1;s=MyIMM_MES_Interface.InjectionUnits.NodeVersion"; see "Step 2: Assigning PLC tags".

Proceed as follows to create the required PLC tag with STEP 7 (TIA Portal):

1. Generate a "MyIMM_MES_Interface" data block.
2. Generate the DB element "InjectionUnits.NodeVersion". Use the SIMATIC data type compatible with the OPC UA data type. The UA tag in the example has the OPC UA data type "String" (DataType="String"). The compatible SIMATIC data type is WSTRING.

3. Compile the project.
4. Download the project to the CPU.

Importing exported OPC UA XML files to an S7-1500 CPU

Please note the following information when importing server interfaces that come from the OPC UA XML export of an S7-1500.

Note

Import blocked for namespace "http://www.siemens.com/simatic-s7-opcua"

You cannot import server interfaces with the namespace "http://www.siemens.com/simatic-s7-opcua" to an S7-1500 CPU because this namespace is reserved for S7-1500 CPUs (standard SIMATIC server interface) and is not available for imports.

If you want to import a server interface with the namespace "http://www.siemens.com/simatic-s7-opcua", open the server interface to be imported (OPC UA XML file) and change the namespace in the relevant places. The file thus changed can then be imported.

Integrity of the OPC UA XML files

OPC UA XML files represent the server address space. These files are, for example, imported by you in the context of OPC UA Companion specifications as a server interface after adaptation to the application, loaded with the hardware configuration into the S7-1500 CPU and tested.

WARNING

No checking of imported OPC UA XML files

Protect these OPC UA XML files against unauthorized manipulation since STEP 7 does not check the integrity of these files.

Recommendation

To minimize risks in the case of an extension or adaptation of the server address space, follow these steps:

1. Protect the project (project navigation: Security settings > Settings).
2. Export the corresponding server interface before the extension or adaptation.
3. Revise this OPC UA XML file.
4. Import the file again as a server interface.

Mapping of data types

The table below shows the compatible SIMATIC data type for each OPC UA data type.

Assign the data types as shown below (SIMATIC data type - OPC UA data type). Other assignments are not permitted. STEP 7 does not check the observance of this rule and does not prevent an incorrect assignment. You are responsible for the rule-compliant selection and assignment of the data types.

You can also use the listed data types, for example, as elements of structures/UDTs for input and output parameters of self-created server methods (UAMethod_InParameters and UAMethod_OutParameters).

Table 1-89 Mapping of data types

SIMATIC data type	OPC UA data type
BOOL	Boolean
SINT	SByte
INT	Int16
DINT	Int32
LINT	Int64
USINT	Byte
UINT	UInt16
UDINT	UInt32
ULINT	UInt64
REAL	Float
LREAL	Double
LDT	DateTime
WSTRING	String
DINT	Enumeration (Encoding Int32) and all derived data types
User-defined data type required (UDT, user-defined data type) The user-defined data type must be created with the prefix "Union_", for example "Union_MyDatatype"; see example below the table. The first element (Selector) in this UDT must have the data type "UDINT".	UNION and all derived data types

User-defined data type for UNION required

The figure below shows the tag "MyVariable", which has the data type "Union_MyDatatype".

This SIMATIC data type corresponds to an OPC UA tag with the data type UNION.

The figure shows an example of the declaration: When Selector = 1, Union accepts one ByteString; when Selector = 2, Union accepts one WString.

Name	Data type
Static	
Selector	UDInt
ByteArray	Array[0..1] of Byte
WString	WString[42]

Using other OPC UA basic data types

Apart from the OPC UA data types listed in the section "Mapping of data types" and their correspondences on the SIMATIC side, there are the following OPC UA basic data types which you can also use:

- OpcUa_NodeId
- OpcUa_QualifiedName
- OpcUa_Guid
- OpcUa_LocalizedText
- OpcUa_ByteString
- OpcUa_XmlElement

Requirement for the use of the basic data types listed above as variables in the application program: The basic data types have to exist as complex data types that are structured exactly like the corresponding OPC UA basic data types.

- OpcUa_NodeId and OpcUa_QualifiedName exist as system data types; that's why you can use these data types not only for single variables but also as elements of a structure.
- For the remaining basic types you have to create a PLC data type in accordance with the OPC UA specification and subsequently use it as an element in a structure so that the data types of the elements can be resolved. What each PLC data type must look like is described below for every single basic data type. "EUInformation" is an example of a data structure in which, for example, the UDT "LocalizedText" is used. EUInformation contains information on EngineeringUnits. You can find an example of the implementation of the EUInformation data structure at the end of the PLC data type descriptions.

System data type "OPC_UA_NodeId"

For the OPC UA basic data type "OpcUa_NodeId", please refer to the following table for the meaning of the parameters. Use OPC_UA_NodeId for the identification of a node in the OPC UA server.

Parameter	S7 data type	Meaning
NamespaceIndex	UINT	Namespace index of the node in the OPC UA server. A node can, for example, be a tag.
Identifier	WSTRING[254]	The designation of the node (object or tag) depends on the identifier type: • Numeric identifier: The node is labeled with a number, for example "12345678". • String identifier: The node is labeled with a name, for example "MyTag". No distinction is made between upper and lower case.
IdentifierType	UDINT	Type of identifier • 0: Numeric identifier • 1: String identifier • 2: GUID • 3: Opaque

System data type "OPC_UA_QualifiedName"

See the following table for the structure of the system data type "OPC_UA_QualifiedName":

Name	S7 data type	Meaning
NamespaceIndex	UINT	The namespace index of the name.
Name	WSTRING[64]	Name of the node or tag.

UDT "Guid"

For the basic data type "Guid", create the following PLC data type. The default values used as examples can also be set differently.

Guid

	Name	Data type	Default value
1	Data1	UDInt	16#11223344
2	Data2	UInt	16#5566
3	Data3	UInt	16#7788
4	Data4	ULInt	16#99AABBCCDDEEFF11

UDT "LocalizedText"

For the basic data type "LocalizedText", create the following PLC data type:

LocalizedText

	Name	Data type	Default value
1	EncodingByte	Byte	16#3
2	Locale	WString[5]	WSTRING#'de-DE'
3	Text	WString[255]	WSTRING#'Text'

The EncodingByte indicates which fields (Locale or Text) are available:

EncodingByte	Meaning
0	The fields Locale and Text are empty
1	The field Locale has content, the field Text is empty
2	The field Locale is empty, the field Text has content
3	The fields Locale and Text have content

UDT "ByteString"

For the basic data type "ByteString", create the following PLC data type; in this case, for example, a ByteString array with 12 elements:

ByteString			
	Name	Data type	Default value
1	ByteString	Array[0..11] of Byte	
2	ByteString[0]	Byte	16#0
3	ByteString[1]	Byte	16#0
4	ByteString[2]	Byte	16#0
5	ByteString[3]	Byte	16#0
6	ByteString[4]	Byte	16#0
7	ByteString[5]	Byte	16#0
8	ByteString[6]	Byte	16#0
9	ByteString[7]	Byte	16#0
10	ByteString[8]	Byte	16#0
11	ByteString[9]	Byte	16#0
12	ByteString[10]	Byte	16#0
13	ByteString[11]	Byte	16#0

UDT "XmlElement"

An XmlElement is a serialized XML fragment (UTF-8 string)

For the basic data type "XmlElement", create the following PLC data type:

XmlElement			
	Name	Data type	Default value
1	XmlElement	WString	WSTRING#''

Example: Structure of EUInformation with UDT "LocalizedText"

Velocity	"EUInformation"		
NamespaceUri	WString[255]	WSTRING#'http://yourorganization.org'	Identifies the organization (company, standards organization) that defines the EUInformation
UnitId	DInt	1	Identifier for programmatic evaluation. –1 is used if a unitId is not available.
DisplayName	"LocalizedText"		
Encoding...	Byte	16#3	
Locale	WString[5]	WSTRING#'en-EN'	
Text	WString[255]	WSTRING#'m/s'	
Description	"LocalizedText"		
Encoding...	Byte	16#3	
Locale	WString[5]	WSTRING#'en-EN'	
Text	WString[255]	WSTRING#'meter per second'	

MinimumSamplingInterval attribute of tags

In addition to "Value", "DataType" and "AccessLevel", you can also set the "MinimumSamplingInterval" attribute for a tag in the XML file that represents the server address space.

The attribute specifies how fast the server can sample the tag value.

The OPC UA server of the S7-1500 CPU handles the values for MinimumSamplingInterval as follows:

- Negative values and values greater than 4294967 are set to -1; this means: Default setting of the server should be used for the sampling interval for sampling. This is defined by the publishing interval of the subscription, see OPC UA specification part 4.
- Decimal numbers are rounded to three decimal places.

See also

Unified_Automation_Download (https://www.unified-automation.com/downloads/opc-ua-development.html)

Missing namespaces

Introduction

A node set file (OPC UA XML file) may contain references to other OPC UA XML files that provide type definitions in other namespaces. You must import all required definitions for the information model in order for the server interface to work properly.

Example for list of namespaces (NamespaceUris) in an OPC UA XML file

```
<NamespaceUris>
    <Uri>http://www.siemens.com/SimaticIdent/RF600R/</Uri>
    <Uri>http://opcfoundation.org/UA/DI/</Uri>
    <Uri>http://opcfoundation.org/UA/AutoID/</Uri>
</NamespaceUris>
```

Importing additional namespaces

If namespaces are still missing after the import of an OPC UA XML file, STEP 7 generates an error message.

In this case, import the OPC UA XML files with the missing namespaces into a new server interface. You must create a new server interface, otherwise the additional imported XML file would overwrite the current companion specification (the current information model).

The missing namespaces are displayed in red in the properties of the server interface.

After importing the XML files, you must click the "Refresh interface" button to refresh the view.

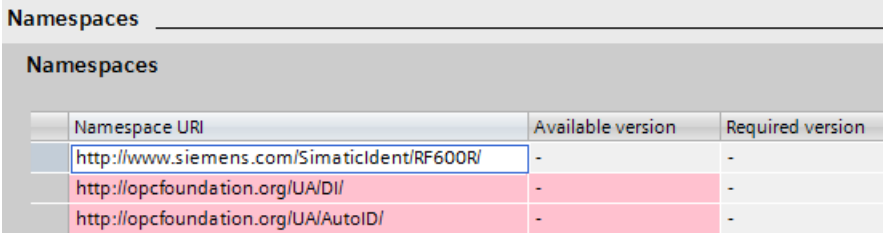

Coordinating write and read rights for CPU tags

Definition of write and read rights in the information model (OPC UA XML)

In the OPC UA information model, the attribute "AccessLevel" regulates access to tags.

AccessLevel is defined bit by bit:

Bit 0 = CurrentRead and Bit 1 = CurrentWrite. The meaning of the bit combinations is as follows:

- AccessLevel = 0: no access
- AccessLevel = 1: read only
- AccessLevel = 2: write only
- AccessLevel = 3: read+write

Example of the assignment of write and read rights (read+write)

```
<UAVariable NodeId="ns=3;s=&quot;Data_block_2&quot;.&quot;Static_1&quot;"
BrowseName="3:Static_1"
ParentNodeId="ns=3;s=&quot;Data_block_2&quot;"
DataType="INT"
AccessLevel="3">
    <DisplayName>Static_1</DisplayName>
```

Definition of write and read rights in STEP 7

When you define tags, you specify the access rights using the properties "Accessible from HMI/OPC UA" and "Writable from HMI/OPC UA".

Example of the assignment of write and read rights

Name	Data type	Accessible from HMI/OPC UA	Writable from HMI/OPC UA
Static_1	Int	☑	☑
<Add new>		☑	☑

Interaction between write and read rights

If you have imported an OPC UA server interface and AccessLevel attributes are set in this OPC UA XML file, the write and read rights are defined by the following rule: The least extensive access rights for each setting apply.

Example

- AccessLevel = 1 (read only) in the OPC UA server interface
- Both "Accessible from HMI/OPC UA" and "Writable from HMI/OPC UA" are selected in the PLC tag table

Result: The tag is only read.

Rules

If write rights are required:

- AccessLevel = 2 oder 3
- "Writable from HMI/OPC UA" enabled

If read rights are required:

- AccessLevel = 1 (AccessLevel 3 is also possible, but misleading. The settings suggests that an OPC UA client has write and read rights)
- "Accessible from HMI/OPC UA" enabled, "Writable from HMI/OPC UA" disabled

If neither read nor write rights are to be granted (no access):

- AccessLevel = 0
- "Accessible from HMI/OPC UA" disabled
  Only one of the two conditions needs to be met to block all access. In this case, review whether the tag in the OPC UA server interface is actually necessary at all.

Access table

"Accessible from HMI/OPC UA" must be set if access over OPC UA is to be possible at all. "Writable from HMI/OPC UA" must be set to allow an OPC UA client to write a tag / DB element.

Please see the table for the resulting access right.

Table 1-90 Access table

OPC UA XML	STEP 7 (TIA Portal), for example tag table		
AccessLevel	Accessible from HMI/ OPC UA	Writable from HMI/OPC UA	Resulting access right
0	x	x	No access
x	0	x	No access
1	Enabled	x	Read only
2	Enabled	Disabled	No access
3	Enabled	Disabled	Read only
2	Enabled	Enabled	Write only
3	Enabled	Enabled	Read+write

(x = don't care)

See also

Consistency of CPU tags (Page 1209)

Consistency of CPU tags

"AccessLevelEx" attribute extends access properties

As of firmware version V2.6, the OPC UA server of the S7-1500 CPU supports not only the attribute "AccessLevel" (see Coordinating write and read rights for CPU tags (Page 1207)) but also the attribute "AccessLevelEx" which, in addition to the already explained bits for read access and write access, provides information on the consistency of a OPC UA tag. The new attribute was introduced with version V1.04 of the OPC UA specification (Part 3, Address Space Model).

Reading consistency properties

In the OPC UA information model of the OPC UA server, the attribute "AccessLevel" defines the access.

AccessLevelEx is defined bit by bit; in this case the relevant bits are:

- Bit 0 = CurrentRead
- Bit 1 = CurrentWrite
- Bits 2 to 7 are not relevant for the OPC UA server of an S7-1500 CPU

The meaning of the bit combinations is explained in the section on read and write rights.

The following bits for consistency are also added:

- Bit 8 = NonatomicRead; the bit is set if the tag cannot be read consistently. For read consistency of tags, bit 8=0.
- Bit 9 = NonatomicWrite; is set if the tag cannot be written consistently. For write consistency of tags, or if no write access is granted, bit 9=0.

Examples

An OPC UA tag (structure) is readable and writable; but inconsistent for reading and writing access.

Consequently: Bits 0, 1, 8 and 9 are set: AccessLevelEx = "771" (1+2+256+512).

Another structure is read-only.

Consequently: Bits 0 and 8 are set, bit 1 and bit 9 are not set: AccessLevelEx = "257" (1+0+256+0).

Handling of the attribute in the server

The "AccessLevelEx" attribute is only available in the OPC UA server. The attribute is not present in a node set file (XML export file).

However, the attribute "AccessLevel", which is exported, includes the information from "AccessLevelEx", see next section.

Export

With XML export of the standard Simatic server interface, the server sets the attribute "AccessLevel", which was extended to 32 Bit Advanced in V1.04 compared to V1.03, to the value of attribute "AccessLevelEx".

Import

When importing a node set file (e.g. from an export of a server interface), the S7-1500 CPU sets the attribute "AccessLevelEx" according to its own estimate of the consistency of the imported data type, see next section. The imported value is ignored.

Consistency of data types at the server interface

The consistency of tags ("atomicity" in the language usage of OPC UA) within a program cycle of an S7-1500 CPU is ensured at the nodes of the server interface for the following data types:

- BOOL, BYTE, WORD, DWORD, LWORD
- SINT, INT, LINT, DINT, USINT, UINT, ULINT, UDINT
- REAL, LREAL
- DATE, LDT, TIME, LTIME, TIME_OF_DAY, LTIME_OF_DAY, S5TIME
- CHAR, WCHAR
- System data types and hardware data types that are based on the above-mentioned data types are also consistent.
  Example: HW_ANY, derived from UINT (UInt16).

Tip: If you browse in the address space of the S7-1500 CPU (e.g. with the OPC UA Client UaExpert), you can find the consistent data types under Types > BaseDataType > Enumeration/Number/String.

Tags of the following data types are **not** consistent ("nonatomic" in the language usage of OPC UA):

- SIMATIC structures are generally not consistent. This means that all tags which, for example, have unknown structures or a UDT data type are not consistent.
- System data types such as DTL, IEC_Counter, IEC_TIMER, etc. are data types that are derived from structures.

Tip: If you browse in the address space of the S7-1500 CPU (e.g. with the OPC UA Client UaExpert), you can find data types based on structures under Types > BaseDataType > Structure.

Notes on configuration limits when using server interfaces

When you use OPC UA server interfaces, you must comply with limits for the following objects in line with the S7-1500 CPU performance class:

- Number of server interfaces
- Number of OPC UA nodes

- Load object data volume
- If you have implemented methods: Number of server methods or server method instances

Configuration limits for OPC UA server interfaces and methods

The table below sets out the configuration limits for S7-1500 CPUs; these must also be taken into account when you compile and load a configuration.

A violation of configuration limits results in an error message.

Table 1-91 Configuration limits for OPC UA server interfaces

Technical specification value	CPU 1510SP (F) CPU 1511 (C/F/T/TF) CPU 1512C CPU 1512SP (F) CPU 1513 (F)	CPU 1505 (S/SP/SP F/SP T/SP TF) CPU 1515 (F/T/TF) CPU 1515 SP PC (F/T/TF) CPU 1516 (F/T/TF)	CPU 1507S (F) CPU 1517 (F/T/TF) CPU 1518 (F)
Use of imported companion specifications (information models)			
• Maximum number of OPC UA server interfaces	10	10	10
• Maximum number of OPC UA nodes in user-defined server interfaces	1000	5000	30000
• Maximum size of loadable OPC UA server interfaces	1024 KB	5120 KB	15360 KB
Provision of methods			
Maximum number of usable server methods or max. number of server method instances (instructions OPC_UA_ServerMethodPre, OPC_UA_ServerMethodPost)	20	50	100

Using the S7-1500 CPU as an OPC UA client

Overview and requirements

With STEP 7 (TIA Portal) Version 15.1 and higher, you can configure and program an OPC UA client with your user program that can read PLC tags in an OPC UA server. Furthermore it is possible to transfer new values for PLC tags to an OPC UA server. In addition, you can call methods that an OPC UA server provides in your user program. You use the new instructions for OPC UA clients for this purpose in your user program.

The instructions of the OPC UA client are based on the standard "PLCopen OPC UA Client for IEC61131-3".

PLCopen specification

With these standardized instructions, you can develop an OPC UA client functions in your user program that can be executed in an S7-1500 CPU.

In addition, it is possible with just a few adaptations to run this user program in controllers of other manufacturers if these manufacturers also implemented the OPC UA Specification "PLCopen OPC UA client for IEC61131-3".

Convenient editors in STEP 7

For the parameter assignment of the instructions for OPC UA clients, a convenient editor is available in the TIA Portal – the connection parameter assignment (Page 1160).

As of Version 15.1, STEP 7 also features an editor for client interfaces (Page 1217).

This section describes how you work with these editors.

First, you will be shown how to create and configure a new interface with the interface editor, because you need this type of interface for the subsequent connection parameter assignment.

The description uses an example for better comprehensibility, see Description of the example (Page 1215).

Requirement

To use the client of the S7-1500 CPU, you must enable it:

1. Select the area "OPC UA > Client" in the properties of the CPU.
2. Select the "Enable OPC UA client" option.

If you do not enable the client, the connection is not established. You receive a corresponding error message at the instructions, for example "OPC_UA_Connect".

For information about the application name, which also applies to the server and the client, see here.

Overview

To use the editor and the connection parameter assignment, follow these steps:

1. First, specify a client interface. Add to this the PLC tags and PLC methods interface that you want to access ("First step (Page 1217)").
2. Next, configure the connection to the OPC UA server (Second step (Page 1230)).
3. Finally, use the configured connection for the OPC UA client instructions (Third step (Page 1238)).

Useful information about the client instructions

With the standardized OPC UA client instructions you are able to control communication for the following tasks with the S7-1500 CPU as an OPC UA client:

- Read/write tags of the OPC UA server
- Call methods in the OPC UA server

Optional instructions can be used to determine the status of the connection between the OPC UA client and OPC UA server or to determine nodes along a know hierarchy.

Standardized sequence of OPC UA communication

The sequence of the communication, and thus the order of the instructions, follows a pattern that is illustrated in the following.

Run sequence for a read or write operation

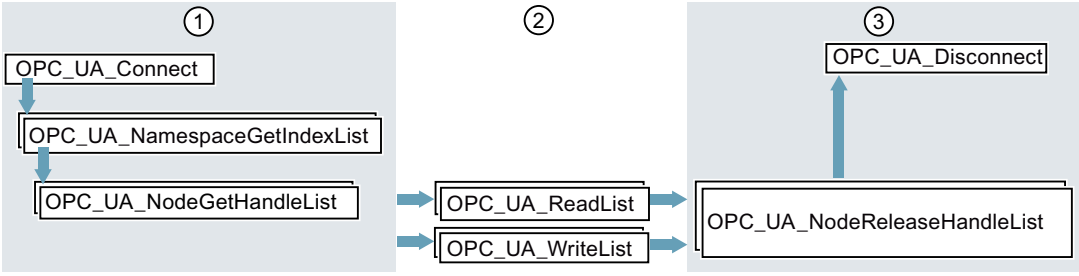

① Instructions for preparation of read and write operations
② Read and write instructions
③ Instructions for "clean-up" after a completed read or write operation

Run sequence for a method call in the OPC UA server

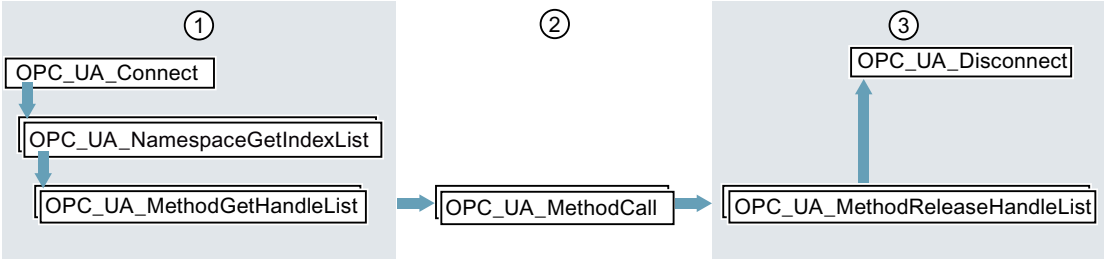

① Instructions for preparation of method calls
② Method calls
③ Instructions for "clean-up" after completed method calls

Optional instructions for reading out the status of a connection between the OPC UA client and OPC UA server and for reading out complete paths from the OPC UA server.

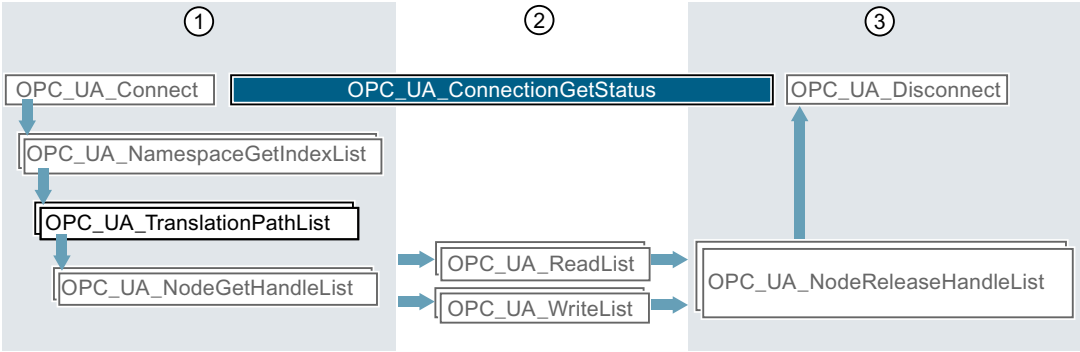

① Instructions for preparation of read and write operations with inserted instruction for requesting, for example, the NodeIDs of nodes of the OPC UA server.

② You can determine the connection status between the establishment and termination of the connection in parallel with other instructions.

③ Instructions for "clean-up"

Convenient editors in STEP 7

The OPC UA client instructions are described in detail in the reference part (STEP 7 information system). For parameter assignment of the instructions, a convenient editor is available in the TIA Portal – the connection parameter assignment (Page 1230).

We recommend starting with the connection parameter assignment for the first program draft and using additional instructions and manually optimizing the program as required.

Information about the client instructions

The client instructions are described in detail in the help to the Instructions > Communication > OPC UA > OPC UA client.

Number of client instructions that can be used simultaneously

SIMATIC error codes for OPC UA client instructions

The following limits apply to the simultaneous use of OPC UA client instructions:

Table 1-92 Quantity structures for OPC UA client instructions

OPC UA instruction	Maximum number for CPU 1510SP (F) CPU 1511 (C/F/T/TF) CPU 1512C CPU 1512SP (F) CPU 1513 (F)	Maximum number for CPU 1505 (S/SP/SP F/SP T/SP TF) CPU 1515 (F/T/TF) CPU 1515 SP PC (F/T/TF) CPU 1516 (F/T/TF)	Maximum number for CPU 1507S (F) CPU 1517 (F/T/TF) CPU 1518 (F)
OPC_UA_Connect	4	10	40
OPC_UA_NamespaceGetIndexList	4	10	40
OPC_UA_NodeGetHandleList	4	10	40
OPC_UA_MethodGetHandleList	4	10	40
OPC_UA_TranslatePathList	4	10	40
OPC_UA_ReadList	20 in total (max. 5 per connection, see OPC_UA_Connect)	50 in total (max. 5 per connection, see OPC_UA_Connect)	200 in total (max. 5 per connection, see OPC_UA_Connect)
OPC_UA_WriteList	20	50	200
OPC_UA_MethodCall	20	50	200
OPC_UA_NodeReleaseHandleList	4	10	40
OPC_UA_MethodReleaseHandleList	4	10	40
OPC_UA_Disconnect	4	10	40
OPC_UA_ConnectionGetStatus	4	10	40

Maximum number of usable OPC UA client interfaces

If you create OPC UA client interfaces using the connection parameter assignment, the number of client interfaces is limited to 40.

Create the OPC UA client interfaces by double-clicking the "Add new client interface" symbol in the project tree of the "OPC UA communication" area.

The maximum number of OPC UA client interfaces is independent of whether you also use the CPU as OPC UA server.

Example configuration for OPC UA

The following sections describe how you can use the client interfaces editor and the connection parameter assignment.

The description is based on a specific example: Two S7-1500 CPUs operate in the system: One CPU serves as the OPC UA client and the other as the OPC UA server.

You can, of course, also use controllers, sensors and IT systems of other manufacturers as OPC UA clients or servers. In particular, the data exchange between different systems (interoperability) is a major advantage of OPC UA.

Connection parameter assignment using an example:

The plant produces blanks in a production line.

The following controllers are used:

1. An S7-1511 CPU serves as the controller of the production line.
   The controller is named "**Productionline**" in the example.
   The OPC UA server of the controller is enabled.
   The CPU has the IP address 192.168.1.1 in the example.
   This CPU publishes the values of following tags via the OPC UA server:
   - **NewProduct**
     The tag has the data type "BOOL".
     When this PLC tag has the value TRUE, the production line has processed a blank.
     The blank is ready for pick-up.
   - **ProductNumber**
     This tag contains the identification number of the blank.
     The tag has the data type "Int".
   - **Temperature**
     This tag contains temperature values recorded during the production of the blank.
     The tag is an array with elements of the "Real" data type.

   In addition, this CPU provides the following writable tag:
   - **ProductionEnabled**
     The tag is set by the OPC UA client.
     The tag has the data type "BOOL".
     If the value is set to TRUE, the production line is released and may produce blanks.

   In addition, this CPU provides the following method via the OPC UA server:
   - **OpenDoor**.
     OPC UA clients can hereby arrange for an access door to be opened to the production line.
2. An S7-1516 CPU controls the interaction with other production lines.
   This CPU is named "**Supervisor**" in the example.
   The OPC UA client of this CPU is enabled.
   Using OPC UA, this CPU can read the NewProduct and ProductNumber tags, set the ProductionEnabled tag and call the OpenDoor method.
   The CPU has the IP address 192.168.1.2 in the example.

The following figure shows the example in the network view of the TIA Portal:

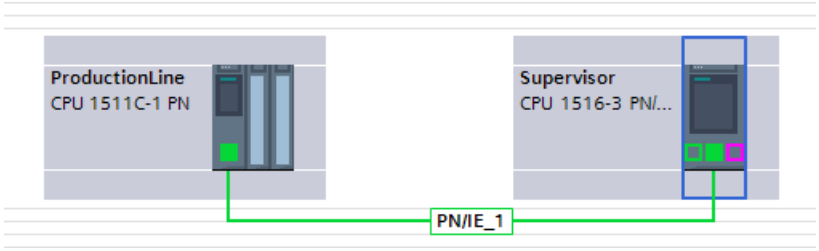

Creating client interfaces

As of Version 15.1, the TIA Portal has an editor for client interfaces.

You group all PLC tags that you want to read or write from an OPC UA server in a client interface.

In addition, the client interface contains all methods that the OPC UA server provides and that you want to call with your user program (that acts as an OPC UA client).

If you create a client interface, STEP 7 also creates data blocks for the parameter assignment of the connection to the OPC UA server from which you want to read data or to which you want to write data.

Maximum number of client interfaces

You can create a maximum of 40 client interfaces.

Editor for client interfaces

To create a client interface, follow these steps:

1. Select the project view in the TIA Portal.
2. In the "Devices" area, select the CPU you want to use as an OPC UA client.
3. Click "OPC UA communication > Client interfaces".

4. Double-click "Add new client interface".
   STEP 7 creates a new client interface and display in the editor.

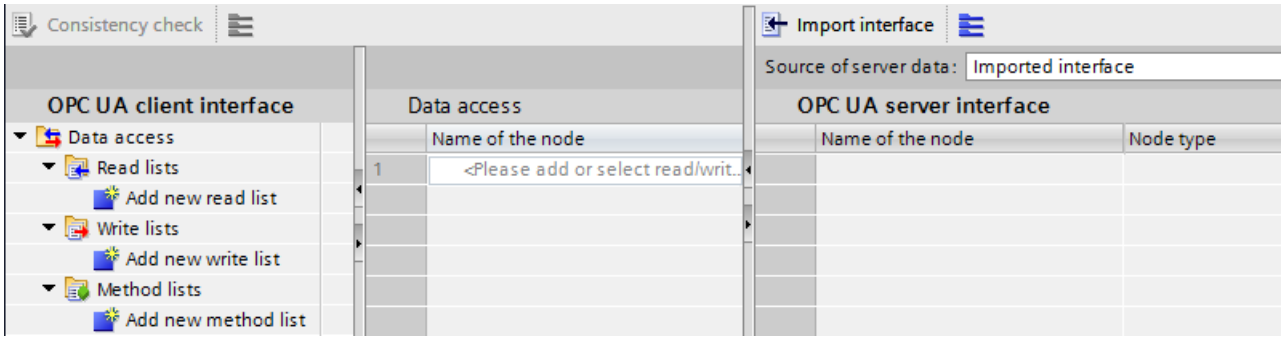

   STEP 7 names the new interface "Client interface_1". If a "Client interface_1" already exists, the new interface receives the designation "Client interface_2" etc.
   In addition, STEP 7 creates the following data blocks:

   – Client_Interface_1_Configuration
     The data block already contains all system data types that are needed for the instructions of the OPC UA client.
     This data block is filled when you configure the connection to the OPC UA server.
     You configure a connection in the properties of the client interface, see: Example configuration for OPC UA (Page 1215).

   – Client_Interface_1_Data
     A data block for the PLC tags that you want to read or write from an OPC UA server.
     You use this data block in your user program.
     This data block is currently still empty.

5. Select a descriptive name for the new client interface.
   Select "Productionline" in the example.
   This also changes the names of the associated data blocks to:

   – Productionline_Data

   – Productionline_Configuration

6. To import an OPC UA server interface, click the "Import interface" button in the top right of the editor.
   This allows you to import an XML file which describes the server interface of an OPC UA server.
   Alternative: To determine online the server interface of a connected OPC UA server, see: Determine server interface online (Page 1224).

7. STEP 7 displays a dialog with which you can select an XML file.
   This XML file describes a server interface (address space of an OPC UA server).
   A server interface is the grouping of all PLC tags and server methods which an OPC UA server publishes.
   OPC UA clients can access the server interface:
   - Read PLC tags
   - Write PLC tags
   - Calling Server Methods
   For information on how to create a server interface for an OPC UA server, see Create a server interface (Page 1189).

8. Create a read list in this client interface.
   To do this, follow these steps:
   - Click "Add new read list" in the left section of the editor.
     STEP 7 adds a new list named "ReadList_1".
     For the example, change the name to "ReadProduct"
   - Now add the new read list of the PLC tags that you want to read from the OPC UA server.
     In the example the "NewProduct" and "ProductNumber" tags are added to the "ReadProduct" read list.
     Select the "NewProduct" tag in the right-hand field of the editor ("OPC UA Server interface"). Drag the "NewProduct" tag to the "ReadProduct" read list in the middle field of the editor. Follow the same procedure with the "ProductNumber" tag.

   The figure below shows the right field of the editor.

OPC UA server interface

Name of the node	Node type	Access level	Node ID
Productionline	Object		http://www.sie...
DataBlocksGlobal	Folder		http://www.sie...
Data_for_OPC_UA_Clients	Object		http://www.sie...
NewProduct	Boolean	RD	http://www.sie...
ProductNumber	Int16	RD	http://www.sie...
Temperature	Array of Float	RD	http://www.sie...
Data_from_OPC_UA_Clients	Object		http://www.sie...
ProductionEnabled	Boolean	RD/WR	http://www.sie...
DataBlocksInstance	Folder		http://www.sie...
OpenDoor_DB	Object		http://www.sie...
InOuts	Folder		http://www.sie...
Static	Folder		http://www.sie...
Method	Method		http://www.sie...

Alternative:
You can also select a new read list by dragging the right field of the editor ("OPC UA Server interface") to a node of the type Object or Folder and then dragging it to "Add new read list" in the left field of the editor. The new read list then contains all PLC tags of the node that has been moved.
In the example, select the object "Data_for_OPC_UA_Clients", which contains the tags "NewProduct" and "ProductNumber". STEP 7 generates the new write list "Data_for_OPC_UA_Clients". In addition, the object contains the tag "Temperature". Delete the "Temperature" tag from the read list. The OPC UA server should not query the "Temperature" tag.
Change the name of the read list in "ReadProduct".
The following figure shows the content of the read list:

ReadProduct

Name of the node	Node type	Access level	Node ID
NewProduct	BOOL	RD	http://www.siemen...
ProductNumber	INT	RD	http://www.siemen...

9. If you want assign new values to PLC tags, create a write list in this client interface.
   To do this, follow these steps:
   - Click "Add new write list" in the left section of the editor.
     STEP 7 adds a new list with the name "ReadList_1".
     For the example, change the name to "WriteStatus".
   - Now add the new write list of all OPC UA server tags to which you want to assign new values.
     In the example, add the "WriteStatus" tag to the write list "ProductionEnabled".
     Select the Tag of right field of the editor ("OPC UA Server interface"). Drag the tag to the write list in the middle field of the editor.

   Alternative:
   You can also create a new write list by selecting a node of the type Object or Folder in the right field of the editor ("OPC UA server interface") and then dragging to "Add new write list" in the left field of the editor.
   The new write list then contains all tags of the relevant node.
   In the example, select the object "Data_from_OPC_UA_Clients", which contains the tag "ProductionEnabled". STEP 7 generates the new write list "Data_from_OPC_UA_Clients".
   Change the name in "WriteStatus".
   The following figure shows the content of the write list:

   WriteStatus

Name of the node	Node type	Access level	Node ID
ProductionEnabled	BOOL	RD/WR	http://www.siemen...

10. If you want to call a method of this OPC UA server, generate a new method list.
    To do this, follow these steps:
    - In the left section of the editor, click "Add new method list".
      STEP 7 adds a new list with the name "Method list_1".
      For the example, change the name to "CallOpenDoor".
    - Now add a method of the OPC UA server to the new method list.
      In this example, add the method "OpenDoor" to the method list "CallOpenDoor".
      Select the method of right field of the editor ("OPC UA Server interface"). Drag the method to the method list in the middle field of the editor.

    Alternative:
    You can also generate a new method list by selecting a method (node of the type Object) in the right field of the editor (OPC UA Server interface) and then dragging it to "Add new method list" in the left field of the editor. The new method list then contains the method of the relevant node.
    The following figure shows the content of the method list:

    CallOpenDoor

Name of the node	Node type	Access level	Node ID
Method			http://www.siemen...

    If you want to call another method of the OPC UA server, you must create a new method list. Each method list contains only one method.

11. Compile the project.
    To do so, select the project and click the following button in the toolbar:

STEP 7 compiles the project and updates the data blocks that belong to the "ProductionLine" client interface.

Note

During compilation, STEP 7 overwrites all data in the data blocks belonging to the client interface. For this reason, you should neither add to nor correct these data blocks manually.

Note

Renaming nod names (DisplayNames)

In read lists, write lists and method lists you can rename the name of a node by means of the shortcut menu. This is the "DisplayName" in the OPC UA language usage.

If you rename the name of a method list node and the node is already used in a programmed block for the method call "OPC_UA_MethodCall", the compilation of the project leads to consistency errors: During the compilation the UDTs of the method are generated with the changed name. The references to the method used in the program are then no longer correct.

To correct the consistency errors, you can either undo the name change of the method in the client interface or navigate to the method call and assign the relevant parameters again there under "Properties > Block parameters" ("Configuration" tab).

Data blocks of client interface

The following data blocks belong to the "Productionline" client interface:

- **Productionline_Configuration**
  A data block for the configuration.
  In the example, this data block is called "Productionline_Configuration".
  The data block already contains all system data types that are needed for the instructions of the OPC UA client.
  In addition, the data block contains general default values for parameter assignment of the connection to an OPC UA server.
  If you are working with connection parameter assignment, this data block will be filled.
- **ProductionLine_Data**
  A data block for the PLC tags that you have entered in the client interface editor.
  In the example, this data block is called "Productionline_Data".
  The figure below shows the data block.

Static	
ReadProduct	Struct
Variable	"Productionline.ReadProduct"
NewProduct	Bool
ProductNumber	Int
NodeStatusList	Array[0..1] of DWord
NodeStatusList[0]	DWord
NodeStatusList[1]	DWord
TimeStamps	Array[0..1] of LDT
TimeStamps[0]	LDT
TimeStamps[1]	LDT
WriteStatus	Struct
Variable	"Productionline.WriteStatus"
ProductionEnabled	Bool
NodeStatusList	Array[0..0] of DWord
NodeStatusList[0]	DWord
CallOpenDoor	Struct
MethodStatusList	Array[0..0] of DWord
MethodStatusList[0]	DWord
Method	Struct

Use the "ProductionLine_Data" data block in your user program and access the read values of the "NewProduct" and "ProductNumber" PLC tags. This is explained in the following section using an example.

Reading and writing PLC tags of the client interface

Example: Reading the "ProductNumber" value

For example, you write in an SCL program:

```
#MyLocalVariable :=
"ProductionLine_Data".ReadProduct.Variable.ProductNumber;
```

You use this, for example, to assign the number of the blank that was just produced in the production line to the local tag "#MyLocalVariable".

Requirements:

- A connection exists to the OPC UA server of the CPU, which controls the production line.
- The OPC UA client has read the current values.

For this reason you check whether a read value is valid:

- Check whether the value in "ProductionLine_Data".Product.NodeStatusList[1]" is equal to 0.
- Optional: Check when this value was sent from the OPC UA server. This value is in "ProductionLine_Data".Product.TimeStamps[1]. If no time stamp is requested, the communication load is reduced.

Example: Writing the "ProductEnabled" value

Transfer the new values for PLC tags, in the example for the "ProductionEnabled" tag, to the OPC UA server using the data block.

With the following assignment, you enable the production line in the example plant:
`"ProductionLine_Data".WriteStatus.Variable.ProductionEnabled := TRUE;`

This is only successful, however, if the following requirements are met:

- A connection exists to the OPC UA server of the CPU, which controls the production line.
- Current values are being written via the OPC UA client

Consistency check

Finally, check the consistency of the read/write list or method list.

1. Select the list that you want to check.
2. Click the "Consistency check" button above the "OPC UA client interface" area.

A green check mark indicates an error-free assignment of the tags or methods to the corresponding elements of the server interface.

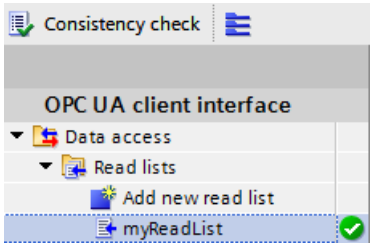

You can assume that the data exchange between client and server and method calls operate without problem in runtime.

In the event of an error a list appears in the Inspector window. From this list you can jump to the respective error.

During the consistency check, STEP 7 checks:

- Whether all elements that you use in the respective list are also present in the server.
- Do the data types used match?
- For methods: Do the number, name, order, and data types of method arguments match?

Determine server interface online

With STEP 7 (TIA Portal) you can determine the interface of an OPC UA server online. This provides information on which tags of a connected OPC UA server you can read or set (write) with OPC UA clients. It also provides information on which server methods of the OPC UA server are available for OPC UA clients.

If you are work offline you can create the interface of the OPC UA server by means of an OPC UA XML file. The address space of the server is described in the OPC UA XML file, see: Export OPC UA XML file (Page 1159).

Determine online server interfaces

To determine a server interface online, follow these steps:

1. In the STEP 7 project tree, select the CPU which is configured as OPC UA client (Supervisor in the example).
2. Select the client interface (in the example, OPC UA Communication > Client interfaces > Productionline).
   If no client interface has been created, double-click "Add new client interface".
3. Double-click the selected client interface.
   The editor for client interfaces is displayed.

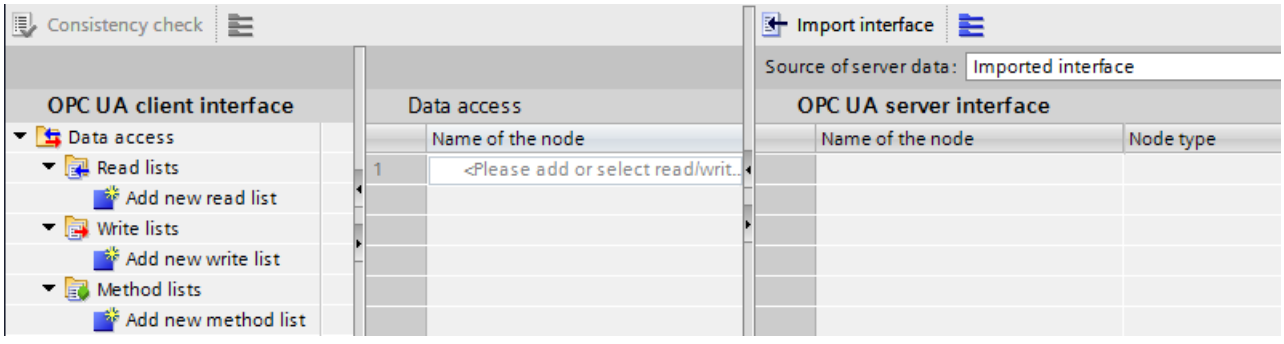

4. In the left section of the editor, click "Add new read list", "Add new write list", or "Add new method list".
5. In the right field of the editor, select "Online[]" as data source for "Source of server data":

6. Click the "Online Access" button.
   STEP 7 displays the "Connect to OPC UA server" dialog.

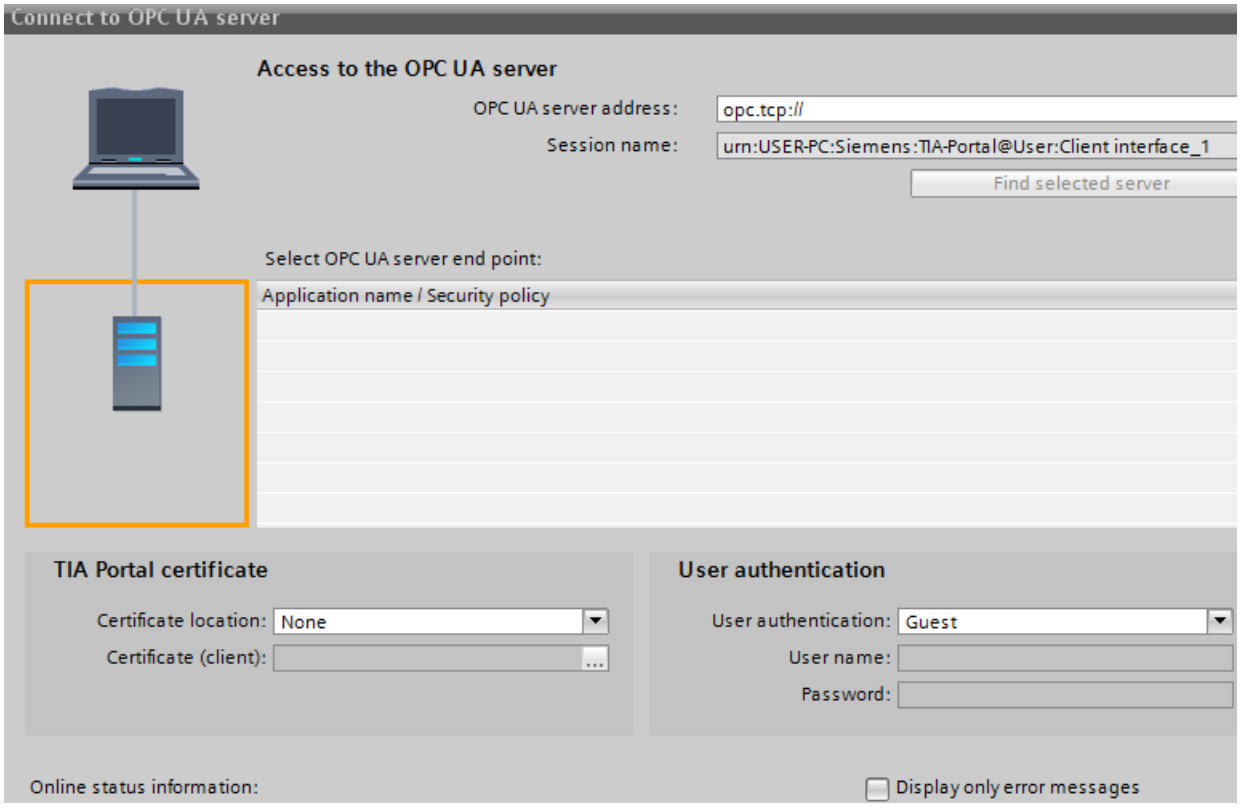

   Tip: When you are establishing an online connection to the OPC UA server for the first time, use the "Online access" button. When reconnecting after a disconnection, select the "Connect to Online Server" button next to the "Online" selection field.
   In the top right, enter the IP address of the OPC UA server whose server interface you want to determine online (in the example, 192.168.1.1).

7. Click "Find selected server".
   STEP 7 establishes a connection to the OPC UA server and determines all security settings (server endpoints) which the server provides.
   STEP 7 displays the end points as list:

8. Click on the end point you want to use for a connection of step 7 to the OPC UA server.

9. Do you want to use a secure connection?
   - If you have selected a secure end point, then select the entry "TIA Portal" for the "Certificate location".
     And under "Certificate (Client)", select a client certificate for your PC on which STEP 7 (TIA Portal) is currently running.
     If a client certificate does not yet exist for your PC, you can generate a client certificate here in the TIA Portal.
     Proceed as follows to generate a certificate for your PC:
     - Click on the button in the "Certificate (Client)" input field.
     - Click "Add".
     - For "Certificate owners" enter "STEP 7 (TIA Portal)".
     - Select the "OPC UA client" entry at "Usage".
     - For "Subject Alternative Name (SAN)", enter the IP address of your PC, on which you are currently running STEP 7 (TIA Portal), under "Value". Overwrite the already entered IP address.
     - If your PC uses a second IP address, enter this address as well. If your PC does not use a second IP address, delete the second IP address already entered.
     - Click "OK".
   - If you have not selected a secure end point, then keep the default ("None").
10. How do you want log on?
    - If you want to log onto the OPC UA server as guest, then apply the default with "User authentication".
    - If you want to log on with user name and password, select "User name and password". Use the user name and password which was stored during the configuration of the OPC UA server in the properties of the CPU under "General > OPC UA > Server > Security > User authentication > User management".
11. Click on the "Go online" button.
    When a secure connection is established, a message appears that you must accept the server certificate for the secure connection to be established. In the message window, you can display further details about the server certificate via a link.
    This standard Windows window only provides information about the server certificate. If you click on the button to install the server certificate but the server certificate is not included in the certificate memory of the TIA Portal, i.e. the next time you connect you will be prompted to accept the server certificate again.
    STEP 7 then establishes a connection to the OPC UA server and again displays the editor for client interfaces.
    In the right field of the editor, STEP 7 displays the uppermost level of the address space of the OPC UA server:

12. Click on the small black triangle next to "Objects".
    STEP 7 now also displays the level below Objects.

13. Click on the small black triangle next to "Productionline".
    STEP 7 now also displays the level below Productionline.
14. Now open additional lower-level folders:

OPC UA server interface

Name of the node	Node type	Access level
Server	Object	
DeviceSet	Object	
Productionline	Object	
Counters	Object	
DataBlocksGlobal	Object	
Icon	ImagePNG	RD
Data_from_OPC_UA_Clients	Object	
ProductionEnabled	Boolean	RD/WR
Data_for_OPC_UA_Clients	Object	
ProductNumber	Int16	RD
Temperature	Array of Float	RD
NewProduct	Boolean	RD
DataBlocksInstance	Object	
Icon	ImagePNG	RD
OpenDoor_DB	Object	
DeviceManual	String	RD

Alternative:
To open the server interface in one stop, click the following symbol:

STEP 7 now displays the fully expanded server interface.
If you click on this symbol again, STEP 7 again displays only the uppermost level of the server address space.

Using multilingual texts

In the client interface editor, you are also importing texts that can be displayed in different languages with the OPC UA XML files (information models). Multilingualism is optional, and each node can be defined differently regarding the languages it offers.

In the XML file, these are the following fields that can be prepared for different languages:

- Display name
- Description

Example for multilingual texts in an OPC UA XML file

In the XML file below, the display name and the description, for example, are entered with a "default" text and multiple localizable texts.

- Default text is the first entry without localization information.
- Localized text is the text after "Locale=" followed by a language code, e.g. "it-IT" for Italian

```
<UAVariable NodeId="ns=3;i=6070" BrowseName="3:EngineeringRevision" ParentNodeId="ns=3;i=1002"
DataType="String">
    <DisplayName>EngineeringRevision</DisplayName>
    <DisplayName Locale="en-US">EngineeringRevision</DisplayName>
    <DisplayName Locale="de-DE">Revisionsstand</DisplayName>
    <Description>Revision Level of the engineering environment.</Description>
    <Description Locale="en-US">Revision Level of the engineering environment.</Description>
    <Description Locale="de-DE">Revisionsstand der Engineeringumgebung.</Description>
    <Description Locale="fr-FR">Niveau de révision de l'environnement d'ingénierie.</Description>
    <Description Locale="it-IT">Livello di revisione dell'ambiente di ingegneria.</Description>
<References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
    <Reference ReferenceType="HasModellingRule">i=78</Reference>
</References>
</UAVariable>
```

Display of multilingual texts

When importing a server interface, the available multilingual texts are saved internally and downloaded to a CPU together with the project.

The client editor displays the text from the OPC UA XML file in the columns "Name of the node" (corresponds to "DisplayName") and "Description" (corresponds to "Description").

The following cascading rules determine which language is shown for a node:

- When the node contains text in the currently used editing language, the text is also displayed in the editing language.
  (Setting the editing language: In the project tree, select the area "Languages & resources > Project languages")
- When the node does not contain text in the editing language but a default text is defined there (without language code), the default text is displayed.
- "Name of the node" column: If no default text is defined either but a text in any other language exists, the DisplayName text is displayed in the first available language. This rule does not apply to description texts (Description).
- If none of the conditions listed above is met, no text is displayed.

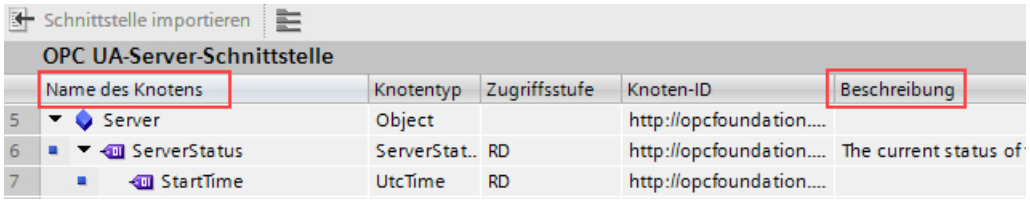

When you change the editing language, the multilingual text in the imported interface will also change according to the rules explained above.

You can then apply the nodes in the corresponding lists (read list, write list, method list) with drag and drop.

You cannot change the language in the lists (read list, write list, method list).

Applying the displayed description texts as comment in PLC data types

When you compile the program, STEP 7 automatically creates PLC data types (UDTs) for each read list, for each write list and for inputs or outputs of each method. These UDTs each have one element for each node.

The UDTs apply the description text as comment according to the rules stated above. STEP 7 creates the comment in only one language, just like the texts in the OPC UA server interface can only be displayed in one language.

Rules for the access to structures

The rules for the access to structures are explained below. Note these rules when reading and writing values of complete structures provided by an OPC UA server.

How the client of the S7-1500 CPU accesses structures

The OPC UA client of the S7-1500 CPU uses neither TypeDictionaries nor DataTypeDefinition attributes, which a server offers for the resolution of these structures.

These options of the OPC UA client for checking structural elements in runtime are limited in the client.

Rules for the access to structures

If you use the client interfaces to configure the read and write lists (connection parameterization) and assign the PLC data types to the imported or online determined address model of the server, the read and write accesses to structures operate trouble-free in runtime.

The configuration by means of client interface automatically ensures that the sequence and the data type of the structural elements are coordinated on client and server side.

Recommendation: Update an S7-1500 CPU (as server) to the current firmware version (e.g. V2.0 > V2.5.2 or higher).

In runtime the OPC UA client only checks the total length of the transmitted value; more detailed checks are not possible.

Mapping rules apply to the assignment of OPC UA structures to PLC tags or DB tags (see Mapping of data types (Page 1131)). Non-executed data type (such as OPC UA byte strings) are not supported.

Example of an error-free assignment of the structure elements

In the imported node set file (XML export), the structure is defined as follows:

opcUaStruct	Object
allOk	Boolean
myOPCstruct1	myOPCUAstruct
varA	Int64
VarB	Byte
nestedStructZ	"myOPCUAstruct"."nestedStructZ"
varD	Float
varE	Double
varC	Double
DeviceManual	String

The structure mapped in the read list matches, both in the order and in the assigned data types, the corresponding nodes of the node set file.

myOPCstruct1	myOPCUAstruct
varA	LINT
VarB	USINT
nestedStructZ	"myOPCUAstruct"."nestedStructZ"
varD	REAL
varE	LREAL
varC	LREAL

If the structure now changes on the server, for example tagA and tagB are swapped, and the read list remains the same in the client, the assignment is no longer correct:

- The total length of the data remains the same (only the order has changed)
- The configuration of the structure is different for client and server!

WARNING

No error message in the case of different structure configuration between client and server

If the structures of client and server do not match, this rule violation will possibly not generate any error during compilation and also not in runtime.

Make sure not to change the configured assignments for structures in runtime. If required, reconfigure the assignment in the read and write lists!

Using connection parameter assignment

Creating and configuring connections

With the instructions for OPC UA clients, you create a user program that exchanges data with an OPC UA server. A series of system data types are required for this.

To simplify your work with these system data types, a connection parameter assignment for OPC UA clients is available starting in STEP 7 (TIA Portal) Version 15.1.

Use of the connection parameter assignment is optional and not mandatory. You can also manually create the required system data types.

We use an example to make the description easier to follow, see description of the example (Page 1215).

Opening the connection parameter assignment

To configure the connection to an OPC UA server, follow these steps:

1. In the "OPC UA communication" area, double-click the client interface whose parameters you want to assign in the project tree.
   For the example configuration: Double-click the "ProductionLine" client interface.

   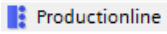

   The section "Create client interface (Page 1217)" describes how to create a client interface.
2. Click the "Properties" tab (Inspector window) if the tab is not already displayed.
   STEP 7 now displays the connection parameter assignment for the instructions of the OPC UA client.
   The "General" tab is open.
3. Click the "Configuration" tab.

Setting the connection parameters

You set the connection to the OPC UA server in the "Configuration" tab.

1. Select a descriptive name for the session. For the example, select the name "OPC UA Connection to ProductionLine".
2. In the "Address" field, enter the IP address of the OPC UA server to which your user program (that operates as an OPC UA client) is to establish a connection.
   In the example configuration, the CPU that controls the production line has the IP address "192.168.1.1". A connection to the OPC UA server of this CPU is to be established. For this reason, you enter this IP address in the "Address" field.
3. If the OPC UA server is not using the standard port 4840, you must insert the port number here.
   For example, enter the number 48040 in the field, if the OPC UA server to which you want to establish a connection uses this port number.
   For the example, we accept the default setting "4840" because the OPC UA server of the example is accessible via port 4840.

4. Enter a path within the OPC UA server to restrict access to this path.
   When you specify a path, it is automatically entered at the "ServerEndpointUrl" entry in the configuration DB for the client interface. The entry then consists of the components "OPC Schematic Prefix", "IP address", "Port number" and "Server path", for example: "opc.tcp://192.168.0.10:4840/Beispiel/Pfad".
   The information is optional. However, some servers only establish a connection if a server path is specified.
5. In addition, you accept the default settings for session timeout (30 seconds) and monitoring time (5 seconds).
   The following figure shows the connection parameters of the example:

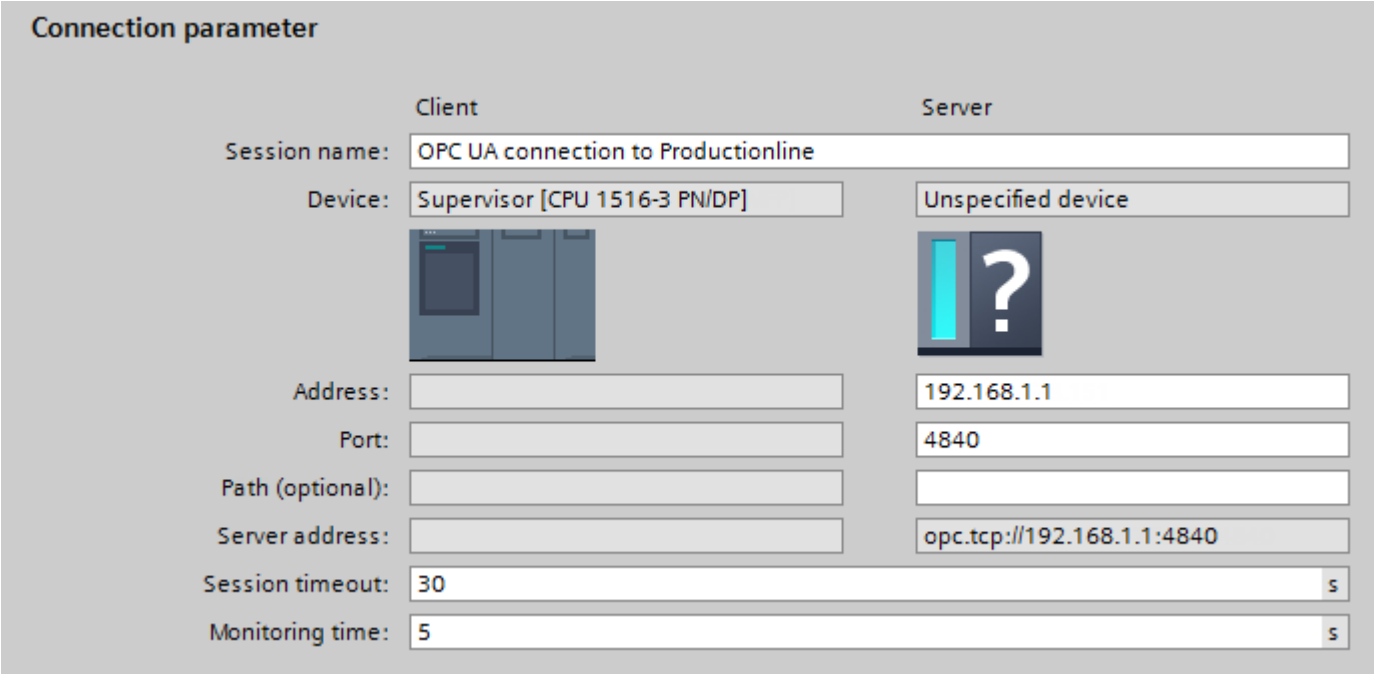

Setting the security parameters

1. Click the "Security" area in the "Configuration" tab.
   This area contains all security settings for the connection to the OPC UA server.

The following settings are possible:

"General" area

Security mode:

Select the security mode that the connection to the OPC UA server must meet from the drop-down list.

If the server does not meet the selected mode, a session is not established.

The following settings are available:

- No security: No secure connection!
- Sign: OPC UA server and OPC UA client sign the data transmission (all messages): Manipulations can thus be detected.
- Sign & Encrypt: OPC UA server and OPC UA client sign and encrypt the data transmission (all messages):

Security policy:

Set the encryption techniques for the signing and encryption of messages.

The following settings are possible:

- No security
- Basic128Rsa15
- Basic256
- Basic256Sha256

To configure a secure connection, you must observe the following items:

- A certificate is required for the client for a secure connection.
- You have to make the client certificate known to the server.

To find out how to proceed, see the section "Handling client and server certificates (Page 1167)" under "Certificate of the OPC UA client".

"Certificates" area

Client certificate:

The certificate confirms the authenticity of the OPC UA client.

To select a certificate, click the following symbol

STEP 7 displays a list of certificates.

Select the certificate that you have made known to the server.

Click the symbol with the green check mark:

Or, create a new certificate. To do so, click the "Add" symbol.

If you create a new certificate, you must make this certificate known to the server.

"User authentication" area

The following settings are possible for user authentication:

- Guest
- User name and password
- Users (TIA Portal - Security Settings)

For more information, see Users and roles with OPC UA function rights (Page 1181).

Setting languages

UA tags of the String type can be localized with OPC UA, that is, texts (values for the UA tag) can be available in different languages for the server. For example, localized texts can be available for DisplayName (Name of the node) and Description (Description).

In the "Languages" area of the "Configuration" tab you can, for example, influence the language of the texts returned by the server as follows:

In the "Languages" area, enter a number of languages that the server transfers to the client during connection setup.

The language or the local ID ("language code") associated with it that you enter in the first line is the language preferred by the client.

- If the server can provide the UA tag in the requested language, it is transferred to the client.
- If the server cannot provide the UA tag in the requested language, it checks whether it can provide the UA tag in the language you have entered in the second line (first substitute language).
- The server works its way down the list, and when it can provide neither the requested language nor a substitute language, it will provide the default language.

See also

Handling of the client certificates of the S7-1500 CPU (Page 1234)

Handling of the client certificates of the S7-1500 CPU

Where does the client certificate come from?

If you are using the OPC UA client of an S7-1500 CPU (OPC UA client enabled), you can create certificates for these clients with STEP 7 V15.1 and higher as described in the following sections.

When you use UA clients from manufacturers or the OPC Foundation, a client certificate is generated automatically during installation or upon the first program call. You have to import these certificates with the global certificate manager in STEP 7 and use them for the respective CPU.

If you program an OPC UA client yourself, you can generate certificates through the program. Alternatively, you can generate certificates with tools, for example with OpenSSL or the certificate generator of the OPC Foundation:

- The procedure for OpenSSL is described here: "Generating PKI key pairs and certificates yourself (Page 1143)".
- Working with the certificate generator of the OPC Foundation is described here: "Creating self-signed certificates (Page 1142)".

Certificate of the OPC UA client of the S7-1500 CPU

A secure connection between the OPC UA server and an OPC UA client is only established if the server classifies the certificate of the client as trusted.

Therefore you have to make the client certificate known to the server.

The following sections describe how you can initially generate a certificate for the OPC UA client of the S7-1500 CPU and then make it available to the server.

1. Generate and export a certificate for the client

For a secure connection you have to generate a client certificate and - if the server and client are located in different projects - export the certificate.

If client and server are in the same project, exporting the client certificate and subsequent import are not necessary.

Requirements

The IP interface of the CPU is configured, an IP address is available.

Background: The IP address under which the CPU can be accessed in your system is entered under "Subject Alternative Name (SAN)".

Creating an OPC UA Client certificate

The easiest way to generate a client certificate for an S7-1500 CPU is to configure a client interface.

The configuration of the client interface provides for the selection or generation of a client certificate, see Creating and configuring connections (Page 1230).

Alternatively, you can create the client as follows:

1. In the "Project tree" area, select the CPU you want to use as a client.
2. Double-click "Device configuration".
3. In the properties of the CPU click "Protection & Security > Certificate manager".
4. Double-click "<add>" in the "Device certificates" table.
   STEP 7 opens a dialog.
5. Click "Add".
6. Select the "OPC UA client" entry from the list at the "Usage".
7. Click "OK".
   STEP 7 now lists the client certificate in the "Device certificates" table.
8. If the server is in another project: Right-click this line and select "Export certificate" from the shortcut menu.
9. Select a directory in which you store the client certificate.

2. Announcing the client certificate to the server

You have to make the client certificate available to the server to allow a secure connection to be established.

To do this, follow these steps:

1. If the client was configured in another project and you created and exported the client certificate there:
   - Select the "Use global security settings for certificate manager" option in the local certificate manager of the server. This makes the global certificate manager available. You will find this option under "Protection & Security > Certificate manager" in the properties of the CPU that is acting as server.
   - If the project is not yet protected, select "Security settings > Settings" in the STEP 7 project tree, click the "Protect this project" button and log on.
     The "Global security settings" item is now displayed under "Security settings" in the STEP 7 project tree.
   - Double click "Global security settings".
   - Double click "Certificate manager".
     STEP 7 opens the global certificate manager.
   - Click the "Device certificates" tab.
   - Right-click in the tab on a free area (not on a certificate).
   - Select the "Import" shortcut menu.
     The dialog for importing certificates is displayed.
   - Select the client certificate that the server is to trust.
   - Click "Open" to import the certificate.
     The certificate of the client is now contained in the global certificate manager. Note the ID of the client certificate just imported.
2. Click the "General" tab in the properties of the CPU that is acting as server.
3. Click "OPC UA > Server > Security > Secure Channel".
4. Scroll down in the "Secure Channel" dialog to the section "Trusted clients".
5. Double-click in the table on the empty row with "<add new>". A browse button is displayed in the row.
6. Click this button.
7. Select the prepared client certificate.
8. Click the button with the green check mark.
9. Compile the project.
10. Load the configuration onto the S7-1500 CPU (server).

Result

The server now trusts the client. If the server certificate is also considered trusted, the server and client can establish a secure connection.

User authentication

In the OPC UA client interface of the S7-1500, you can set what authentication is required for a user of the OPC UA client wishing to access the server. To do so, you must select the corresponding client interface in the project tree of the requested S7-1500 CPU under "OPC UA communication > Client interfaces" and select the type of user authentication in the Inspector window under "Properties > Configuration > Security".

Types of user authentication

The following options are available for user authentication:

- **Guest**
  The user does not need to verify authorization (anonymous access). The CPU creates an anonymous session for the user, and the OPC UA server does not check the authorization of the client user.
- **User name and password**
  The user must prove authorization (no anonymous access). The OPC UA server checks whether the client user is authorized to access the server. Authorization is given by the user name and the correct password. These inputs cannot be checked by the client interface, which means all values are accepted as being valid.

  Note

  STEP 7 stores user name and password unencrypted in the data block/instance data block. Recommendation: Use the user authentication "User (TIA Portal - Security Settings)".

- **Users (TIA Portal - Security Settings)**
  You can enter a user name from the list of users entered in the project for authentication. The names of the registered users for the current project are available in the user administration in the project tree under "Security Settings > Users and roles". There you can also enter additional users.
  You can also enter a name that is not listed in the user administration of the project or leave the field blank. This is necessary when the corresponding user name is only provided by a different source during runtime, for example, via HMI or from a different OPC UA client.

"No Security" security policy and authentication via user name and password

You can set the following combination:

Security policy = "No Security" and authentication via user name and password.

- The OPC UA server of the S7-1500 supports this combination. OPC UA clients can connect and encrypt the authentication data or not.
- OPC UA client of the S7-1500 CPU also supports this combination: However, in runtime it only connects if it can send the authentication data encrypted via cable!

Result: With the following configuration, not connection can be established in runtime.

- S7-1500 as OPC UA client
- OPC UA server which supports no encryption of authentication data when "No Security" (="none") is set as security policy.

See also

Users and roles with OPC UA function rights (Page 1181)

Using a configured connection

Introduction

This section shows you how to use a configured connection for OPC UA instructions (third step).

Requirements

- You have created a client interface and added PLC tags and PLC methods to this interface, see ("First step (Page 1217)").
- You have configured a connection to an OPC UA server (Second step (Page 1230)).

Overview

To read data from an OPC UA server or write data to an OPC UA server, use the following instructions:

- OPC_UA_Connect
- OPC_UA_NamespaceGetIndexList
- OPC_UA_NodeGetHandleList
- OPC_UA_ReadList or OPC_UA_WriteList
- OPC_UA_NodeReleaseHandleList
- OPC_UA_Disconnect

Order of the OPC UA instructions

The following figure shows the order in which the OPC UA instructions are called in a user program in order to use these instructions to read or write PLC tags:

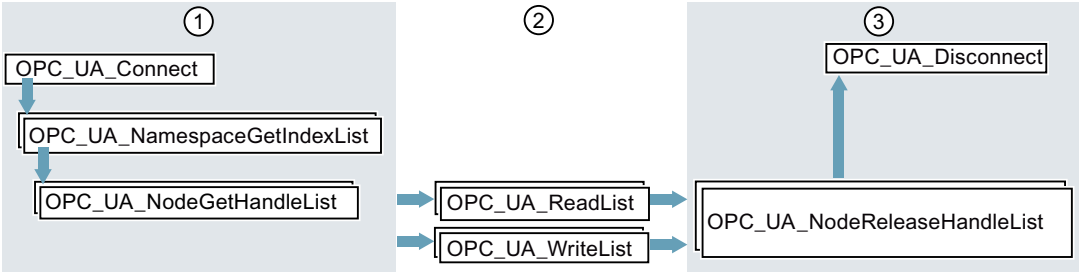

① Instructions for preparation of read and write operations

② Read and write instructions

③ Instructions for "clean-up" after a completed read or write operation
The "OPC_UA_NodeReleaseHandleList" instruction can be omitted if "OPC_UA_Disconnect" is called immediately afterwards.

STEP 7 (TIA Portal) automatically supplies the parameters of these instructions if you are using a client interface and a configured connection to an OPC UA server.

The procedure is shown in the following section.

Using a client interface and configured connection

To use a configured OPC UA connection, follow these steps:

1. Open your user program in the TIA Portal.
2. Using drag-and-drop, move the "**UA_Connect**" instruction into the program editor.
   You will find the instruction under "Instructions > Communication > OPC UA" in the TIA Portal.

3. Select a call option for the instruction
   The example uses a multi-instance.
   STEP 7 displays the instruction in the program editor.
   The editor for the Function Block Diagram (FBD) programming language uses the following display:

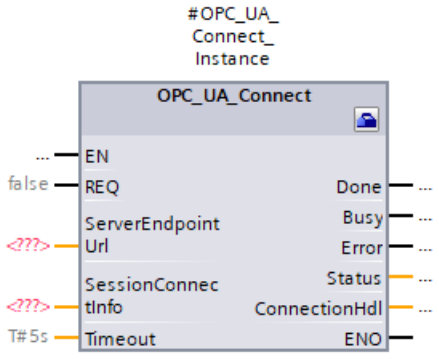

   The editor for the Ladder Logic (LAD) programming language displays the instruction similarly.

4. Click the toolbox symbol in the editor for FBD or LAD.
   The symbol is located in the heading of the instruction:

   If you are using the editor for STL or SCL: Click the small green rectangle below the first character of the instance name:

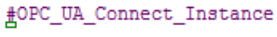

   The example (Page 1215) uses "#OPC_UA_Connect_Instance" as the instance name.
   STEP 7 displays the properties of the instruction in a separate dialog.

5. For "Client interface" select the client interface that you want to use for the instruction.
   We select the "ProductionLine" client interface in the example.
   STEP 7 now interconnects the "ProductionLine" client interface with the parameters of the OPC_UA_Connect instruction:

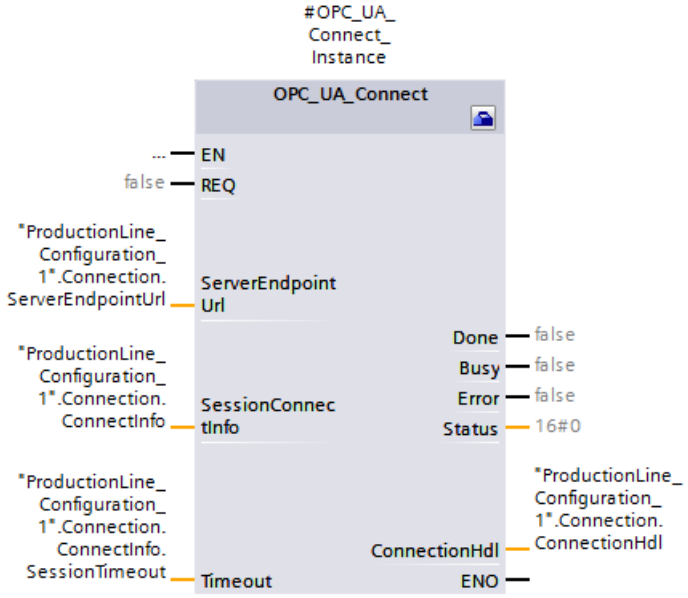

   "ProductionLine" is the interface that the OPC UA client of the example (Page 1215) uses for data exchange with the OPC UA server "ProductionLine".

6. Using drag-and-drop, move the "**UA_NamespaceGetIndexList**" instruction into the program editor.
   You will find the instruction under "Instructions > Communication > OPC UA" in the TIA Portal.
   Select the "Multi-instance" call option.
   Click the toolbox symbol (LAD and FBD) or the small green box below the instance name (STL and SCL) if the editor is not already open.
   Select the client interface that you want to use ("ProductionLine" in the example).
   STEP 7 now automatically interconnects all parameters of the "OPC_UA_NamespaceGetIndexList" instruction:

7. Using drag-and-drop, move the "**UA_NodeGetHandleList**" instruction into the program editor.
   Select the "Multi-instance" call option.
   Click the toolbox symbol (LAD and FBD) or the small green box below the instance name (STL and SCL) if the editor is not already open.
   Select the client interface that you want to use. The example uses the "ProductionLine" client interface.
   Under "Data access > Read/Writelist" select the read list that you want to use (in the example the read list "Product").
   STEP 7 now automatically interconnects all parameters of the "OPC_UA_NodeGetHandleList" instruction:

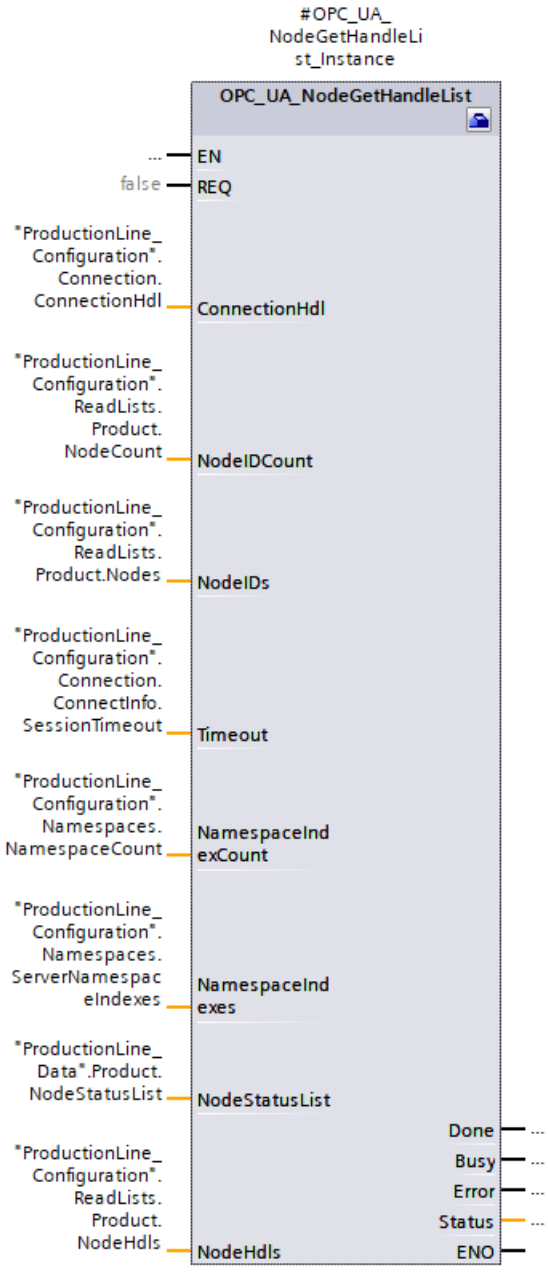

If you want to write data to an OPC UA server, select the write list you want to use under "Data access > Read/Writelist" (the "ProductionStatus" write list in the example).

8. Using drag-and-drop, move the "**UA_ReadList**" instruction into the program editor.
   Select the "Multi-instance" call option.
   Click the toolbox symbol (LAD and FBD) or the small green box below the instance name (STL and SCL) if the editor is not already open.
   Select the client interface that you want to use. The example uses the "ProductionLine" client interface.
   Under "Data access > Read/Writelist" select the read list that you want to use (in the example the "Product" read list).
   STEP 7 now automatically interconnects all parameters of the "OPC_UA_ReadList" instruction.
   If you want to write data to an OPC UA server, use the "**OPC_UA_Write**" instruction and select the list of tags you want to send to the server under "Data access > Writelist" ("ProductionStatus" write list in the example).
9. If you use different read lists or write lists as program-controlled lists in your user program, move the "**UA_NodeReleaseHandleList**" instruction to the program editor using drag-and-drop operation.
   Select the client interface that you want to use.
   Now select a read list or write list that you want to release: Only release read or write lists that you rarely use, since re-registering is time-consuming.
   Then, repeat the steps starting with step 7 with the "UA_NodeGetHandleList" instruction.
10. Using drag-and-drop, move the "**UA_Disconnect**" instruction into the program editor.
    Select the "Multi-instance" call option.
    Click the toolbox symbol (LAD and FBD) or the small green box below the instance name (STL and SCL) if the editor is not already open.
    Select the client interface that you want to use. The example uses the "ProductionLine" client interface.
    STEP 7 now automatically interconnects all parameters of the "OPC_UA_Disconnect" instruction.

Supported instructions

For the following instructions, STEP 7 automatically supplies the parameters if you are using a client interface and a configured connection to an OPC UA server:

- OPC_UA_Connect
- OPC_UA_NamespaceGetIndexList
- OPC_UA_NodeGetHandleList
- OPC_UA_MethodGetHandleList
- OPC_UA_MethodReleaseHandleList
- OPC_UA_ReadList
- OPC_UA_WriteList
- OPC_UA_MethodCall
- OPC_UA_NodeReleaseHandleList
- OPC_UA_Disconnect

1.4.1.16 Configuring the Web server

Information about the web server

Introduction

The Web server lets you monitor and administer the CPU through authorized users by means of a network. This permits evaluation and diagnostics over long distances. All you need is a web browser.

Alarms and status information are visualized on HTML pages.

Web browser

You need a web browser to access the HTML pages of the CPU.

You will find information on the Web browsers and Web browser versions with which the Web server was tested in the manuals listed and linked below.

Note

If you access the Web server of the CPU using a communications processor (CP), ensure that the cache (temporary Internet files) is enabled in your browser. Choose the "Automatically" option in the cache settings of your browser.

If the cache is disabled or if a setting other than "Automatically" is made in your browser, this may result in slow access times and incomplete display.

Reading information via the Web server

The following information can be read from the CPU. The availability of the respective web pages depends on the CPU and its firmware version.

Documentation for S7-300 (http://support.automation.siemens.com/WW/view/en/12996906)

Documentation for S7-400 (http://support.automation.siemens.com/WW/view/en/44444467)

Documentation for S7-1500 (http://support.automation.siemens.com/WW/view/en/59193560)

Below you will find an overview of the information available on the individual Web pages of the Web server of the CPU S7-1500.

Information	Description
Start page with general CPU information	The start page provides an overview of general information on the CPU, the name of the CPU, the type of CPU and basic information on the current operating state.
Diagnostics: Identification information	Displays the static identification information such as serial number, article number and version numbers.
Diagnostics: Memory	Current values on the memory in use.
Content of the diagnostic buffer	Displays the content of the diagnostics buffer with the most recent entries first.
Status of the modules	Displays whether the components of a station are in proper order, if, for example, there are maintenance requirements or components cannot be reached.
Firmware update	As a user with suitable access rights, you have the option of updating the firmware of CPUs, of the display or individual central or distributed modules with an update file.
Alarms	Displays project-oriented alarm text with filters and sorting options.
Information about communication	Display of the communication connections, display of the resources as well as interface parameters and port statistics.
Topology	Display of the topology with the graphic, table views and status overview for networked PROFINET participants.
Status of the tags	Display of the status of user program operands for observing the values. Tip: When the tag status page is exited, the entries made on it are not saved. If you want to monitor the same entered tag again later on, create a bookmark in your Web browser for the "Tag status" page. Otherwise, you will have to enter the tag again when the page is re-opened.
Watch tables	Display of the watch tables created with STEP 7 with the "Web server" option.
User pages (if such Web pages/Web applications have been configured and loaded)	The user pages deliver a list of Web pages with customer-specific Web applications. With the first call of the "WWW" instruction in the user program, the link to the user page is displayed on this Web page . A click on the link opens the user page in a new window.

Information	Description
File browser	S7-1500: The file browser lists files and directories that are located on the SIMATIC Memory Card. Files can be downloaded, deleted, renamed or uploaded regardless of the file type. Directories can only be created, deleted or renamed. The Filebrowser only grants you read access to the "DataLogs" and "Backups" directories. System files are not displayed in the file browser and can therefore not be changed or deleted. System files are the order file and all the special directories including their content which are referenced by the order file.
Data logs	Display and call up of the data logs generated by the user program. Note: If you open large data log files via this Web page, the processing times of the data log instructions that process this data log file can increase considerably. Although the processing time of the priority class (OB in which the data log instruction is called) does not increase, the output parameter BUSY is set for the duration of the processing.
Online backup	S7-1500 (firmware V2.0 or higher): Backing up and restoring the CPU configuration. You can back up a CPU configuration using the Web server with the corresponding access rights. If necessary, you can restore this configuration at a later time. You can create as many backups as you want and store a variety of configurations for a CPU.
Motion diagnostics	S7-1500 (firmware V2.0 or higher): Displays the status, errors, technology interrupts and the current values of technology objects (TOs), for example, speed axis (TO_SpeedAxis), positioning axis (TO_PositioningAxis), synchronous axis (TO_SynchronousAxis), external encoder (TO_ExternalEncoder), measuring input (TO_MeasuringInput), cam (TO_OutputCam), cam track (TO_CamTrack). The "Motion diagnostics" Web page provides detailed information on the configured technology objects in the Diagnostics and Service overview views.
Trace	Display of measurements stored on the SIMATIC memory card.

Web access to the CPU via PG/PC

Proceed as follows to access the Web server:

1. Connect the client (PG/PC) to the CPU via the PROFINET interface.
   If you do not access the web via the PROFINET interface of the CPU but via the interface of a CP (e.g., CP 1543-1), the web pages load much more slowly than with the connection via the interface of the CPU. Make sure that the cache settings (temporary Internet files) of the browser are set to "Automatic" in this case.
2. Open the web browser.
   Enter the IP address of the CPU in the "Address" field of the Web browser in the format http://ww.xx.yy.zz (example: http://192.168.3.141).
   The start page of the CPU is opened. You can navigate from the start page to additional information.
   If the start page does not appear, make sure that the Web server was activated in the configuration and that the configuration was loaded.

Web access to an S7-300/400 CPU via HMI devices and mobile devices

The Web server also supports the Windows Terminal Service so that in addition to the operation of programming devices and PCs, thin client solutions with mobile devices (e.g. PDA, MOBIC T8) and rugged local stations (e.g. SIMATIC MP370 with ThinClient/MP option) with Windows CE are possible.

Proceed as follows to access the Web server:

1. Connect the client (HMI device, PDA) with the CPU via the PROFINET interface.
2. Open the web browser.
   Enter the IP address of the CPU in the "Address" field of the Web browser in the format http://ww.xx.yy.zz /basic (example: http://192.168.3.141/basic).
   The start page of the CPU is opened. From the start page, you can navigate to further information.

HMI devices operating with the Windows CE operating system V 5.x or earlier process CPU information in a browser specially developed for Windows CE. The information appears in a simplified format in this browser.

Note

Using SIMATIC Micro Memory Card with Web server (S7-300)

The configuration data for the Web server is stored on the SIMATIC Micro Memory Card. That is why we recommend using a SIMATIC Micro Memory Card with at least 512 KB.

You can also use the Web server without SIMATIC Micro Memory Card. The CPU must have been assigned an IP address to permit operation.

- The content of the diagnostic buffer is displayed in hexadecimal code.
- Start page, identification and communication information and tag status are displayed as plain text.
- Following displays remain empty:
  - Topology
  - Alarms
  - Status of the modules
- The automatic page refresh is also activated without SIMATIC Micro Memory Card and without configuration.

Web access to an S7-1500 CPU via HMI devices and mobile devices

HMI devices operating with the preinstalled Windows CE operating system (V 5.x or earlier) process CPU information in a browser specially developed for Windows CE. The information appears in a simplified format in this browser.

The Web server offers web pages with reduced complexity which have been optimized for devices with small screens and low computing power.

Requirement for access with mobile devices:

- WLAN connection for the CPU (WLAN Access Point, for example, SCALANCE W788-1RR or SCALANCE W784-1)
- Activated Web server on the CPU
- WLAN has to be activated for the mobile device and a connection to the access point has to be established.

The procedure for a mobile device is identical to that for other devices (programming device/PC). In the browser, you enter the IP address of the CPU interface, download the certificate and install it subsequently.

Security

Make sure that you protect the CPU from being compromised through the use of different methods, for example, by limiting network access and using firewalls.

The Web server offers the following security functions:

- Access via the secure transmission protocol HTTPS
- Configurable user authorization via user list

See also

Requirements for web access (Page 1249)

Reading out service data

The web server of an S7-1500 CPU gives you the option of saving service data. In addition to the contents of the diagnostic buffer, service data includes additional information on the internal status of the CPU. If you should encounter a problem with the CPU that you cannot resolve, you have the option of submitting the service data to the Service&Support team.

Procedure

1. Enter the following address in the address bar of your web browser:
   "http://<CPU IP address>/save_service_data.html", e.g.,
   "http://192.168.3.141/save_service_data.html"
2. Your screen displays the service data page with a button for saving the service data.

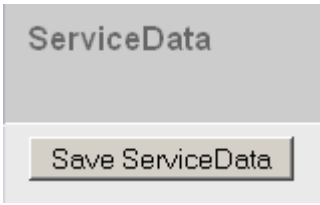

3. Save the service data locally on your display device by clicking "Save ServiceData".

Result

The data is saved into a .dmp file with the following naming convention: "<MLFB><serial number><time stamp>.dmp". The file name can be changed subsequently.

Standard websites

Requirements for web access

In the following, the requirements for accessing standard CPU websites are explained, and the effects of missing or existing configuration information are illustrated.

Requirement

The web server must be started.

The web server does not start until you have enabled it in the "Web server" area of the CPU properties.

Note the following:

- The websites are transmitted by default through a non-secure connection and are not protected from third party access. If you want to transmit the websites in an encrypted fashion, then enable the "Only permit access through HTTPS" option in the CPU characteristics. Make sure that the URL of the CPU begins with https:// in this case.
- Access through HTTPS only works if the time was set in the CPU.
- For further access protection, create a user in the "Web server" area of the CPU properties.
- The full functionality of the web pages in relation to module state, topology and alarms requires system diagnostic data to be generated and loaded (Report System Error) for S7-300/400 CPUs. After the web server has been activated, a query appears asking if the system diagnostics is to be enabled. You should acknowledge this query with "Yes".

Logon

To access the standard websites, no logon is required. A user must be logged on to execute certain actions like changing the operating mode of the CPU or for write access. The input fields for logging on are located in the upper left corner on each standard website.

If you logon as a user, you have to enter the user name and password here. The user name and password are set in the "Web server" area of the CPU properties.

Web access without project information

Without configuration, only basic information is available on the standard websites. This information is available in different languages:

- Diagnostic buffer entries are not displayed as clear text without configuration information, but rather only as a hexadecimal code.
- If the web server supports the CPU tag tables (watch tables):
  The tag tables page does not display any tags without configuration. You can always open up a tag status.

Web access with configuration information

Texts are contained in the selected languages in the configuration information:

- Text for diagnostic buffer entries as short/long text
- For S7-300/400 CPUs: Alarm texts; for example, alarm component AlarmS/SQ, AlarmD/DQ
- If tag tables (watch tables) are supported: Tag tables with the symbols and comments contained in them

The texts for diagnostic buffer entries and alarms are shown during operation in the language that you set in your web browser.

See also

Settings for operation (Page 1251)

Settings for operation

Settings for operation

To use a CPU web server and set the behavior of the web server, you have to highlight the CPU in the network view, device view or topology view.

Execute the following settings in the Inspector window under "Properties > General > Web server":

- Enable the web server
- Limiting web access to secure Hypertext transmission protocol (HTTPS)
- Activate automatic update
- Multilingual (set project languages for Web server and CPU display)
- Expanding user lists
- For S7-300/400 CPUs: Selecting the display classes of the alarms

Every PROFINET interface of an S7-1500 CPU or of a PROFINET CP or a PROFINET-CM can also be enabled or disabled for "external" access to the web server. It is simplest to manage this access centrally in the properties of the CPU. To do so, click the "Web server > Overview of interfaces" area in the "Properties" tab of the inspector window and activate the interfaces through which access is allowed.

See also

Requirements for web access (Page 1249)

Enable the web server

Enable the web server

The web server is disabled by default. The corresponding option box must be enabled to have the CPU show websites.

This setting can be found in the Inspector window under "Properties > General > Web server"; optional box "Enable web server for this module".

If the web server was activated, a prompt is displayed for S7-300/400 CPUs asking whether the system diagnostics (Report System Errors) should be enabled. Click on "Yes" if the required modules and texts should be generated with "Report System Errors" so that the alarm texts are available in the web browser during operation. For S7-1200/1500 CPUs, system diagnostics is always enabled (cannot be disabled).

In the case of S7-1500 CPUs with several PROFINET interfaces or when CPs with PROFINET interfaces are configured, you additionally have to activate the interfaces through which web

server access is to be allowed. This setting is located under "Properties > General > Web server > Overview of interfaces".

Activate automatic update

Activate automatic update

With the "Automatic updating" check box activated, the web sites are automatically updated in the time intervals set under the "Update interval". The "Identification" website is excluded from the updating.

Proceed as follows to activate the automatic updating:

1. Select the check box "Activate automatic update" under "Properties > General > Web server > Automatic update".
2. Enter the updating interval.

Note

Update time

The set activation interval is the shortest updating time. Larger amounts of data or several HTTP/HTTPS connections increase the update time.

Set the regional web language

Set the regional web language

To select the regional languages for language-dependent texts which will be downloaded to the CPU, proceed as follows:

1. Navigate to the "Multilingual" area in the Inspector window.
2. Activate the required languages for display in the web browser in the first column; the number of selectable languages is specific to the CPU.
3. Assign the desired project language to the languages of the display / Web server.
   The project languages that are assigned must be activated and the corresponding texts (translations) must be in the project. The project language selection can be found in the project navigation under "Languages & Resources".
   Through these assignments, for example, display texts not available in the CPU will be downloaded in the selected language (with language and country code) when loading the CPU.

Project language	Languages of CPU display/Web server
English (United States)	German
English (United States)	English
English (United States)	French
English (United States)	Spanish
English (United States)	Italian
English (United States)	Japanese
English (United States)	Chinese (simplified)
English (United States)	Korean
English (United States)	Russian
English (United States)	Turkish
English (United States)	Portuguese (Brazil)

Note

If you activate the web server and do not assign a language, messages and diagnostic information are displayed in a hexadecimal code.

User administration

User Management

The User Management offers the following possibilities:

- Create users
- Set execution rights
- Assign passwords

With this assignment, users are exclusively entitled to options that are assigned to the execution rights.

- If a user is configured, this user can access the web pages according to his access authorizations after logging on with his password.
- Depending on the CPU and firmware used, you can assign different user rights. Rights that your CPU does not support cannot be selected.
- A user called "Everybody" with minimal access rights is created by default in the user list: Read access to the intro and start page. The "Everybody" user is defined without assignment of a password.
- As a non-logged-in user, you always access the Web server as the "Everybody" user by default. It does not matter in this case whether you have configured additional users. However, you can assign all access authorizations available in STEP 7 to the "Everybody" user. Because the "Everybody" user is defined in STEP 7 without assignment of a password, pay close attention to the access authorizations you assign to this user. Individual authorizations, such as the option to change the operating mode, may pose a safety risk.
  For assigning of safety-related authorizations, we recommend the following: Configure a user and always assign a password!

Recommendations for password assignment

- Assign secure passwords to users during configuration.
  An example of a secure password is one which is only used for a single application, is more than 8 characters long, and consists of lower-case and upper-case letters as well as special characters and numbers (?!+%$1234...).
- Do not use commonly used character sequences on the computer keyboard or words from the dictionary.
- Change the password at regular intervals. If possible, select the "Permit access only with HTTPS" option as soon as you have assigned a password to at least one user.

Server security

A Web server certificate has to be generated and assigned to the CPU for SIMATIC S7-1500 CPUs as of firmware V2.0 to use the HTTPS transmission protocol.

The Web server certificate is also prerequisite for user administration of the Web server (password protection).

Requirements

You have opened the properties of the CPU (Inspector window).

Procedure

The settings for generation of a Web server certificate are located in the CPU properties in the "Web server > Security" area.

You can generate or select certificates in this area. A distinction is made as to whether or not you are using the certificate manager in the global security settings.

For additional information of the certificate manager and the procedure involved refer to "See also".

Generation of Web server certificates depending on the CPU firmware version

With SIMATIC S7-1500 CPUs as of firmware V2.0, you have to create the certificate for the Web server of the CPU with STEP 7 yourself and assign it to the Web server in the CPU properties. This certificate is also downloaded to the CPU automatically when the hardware configuration is downloaded.

The STEP 7 project itself has a root certificate (CA, Certification Authority) with which the Web server certificate can be signed. Logging in for the certificate manager in the global security settings is required to this purpose.

Exception:

When you update the firmware of an S7-1500 CPU or ET 200SP with firmware version < V2.0 to a firmware version ≥ V2.0, a valid server certificate is automatically generated by the CPU and used. The same applies to the replacement parts scenario in which a newer CPU replaces a CPU with firmware version < V2.0.

In these cases the Web server certificate of STEP 7 is not used.

Access only through HTTPS

Access only through HTTPS

HTTPS is used for the encryption and authentication of the communication between the browser and web server.

The web pages are transmitted by default via non-secure connection and are not protected from attacks by third parties. If you always want to transmit web pages and login information to the browser in secure form, enable the "Permit access only with HTTPS" option in the CPU properties. Note that the URL of the CPU starts with "https://" in this case.

Note: Prerequisite for the use of the secure transmission protocol "HTTPS" is a valid Web server certificate in the CPU.

Requirements for access via HTTPS

For error-free HTTPS access from your Web browser to the Web server of the CPU, the following conditions must be met:

- The current time is set on the CPU.
  The following generally applies: When using secure communication (for example, HTTPS, secure OUC, OPC UA), make sure that the corresponding modules have the current time of day and the current date. Otherwise, the modules evaluate the used certificates as invalid and the secure communication does not work.
- The IP address of the CPU must be set.
- A valid root certificate offered by the CPU should be installed in the Web browser.
  If no certificate is installed, a warning is output recommending that you do not use the page. To view this page, you must explicitly "Add an exception".
  A valid root certificate (certification authority) is available for download on the "Intro" web page of the Web server under "Download certificate".
  You can find instructions for installing the certificate in the help system of your Web browser and in the FAQ (https://support.industry.siemens.com/cs/ww/en/view/103528224) with the entry ID 103528224 on the Service&Support Internet page.

Note

To protect against manipulation from the outside, download the certificate only in an environment that is guaranteed not to be compromised.

Installation of the root certificate has to be carried out once for each display device you wish to use.

Example of access to web pages with HTTPS protocol

To access web pages of the CPU with HTTPS protocol, enter the URL in the following format: https://ww.xx.yy.zz, with ww.xx.yy.zz being the IP address of the CPU.

Example entry: https://192.168.3.141

You can recognize an encrypted connection by a lock symbol in the status bar of the web page.

Access protection

The certificate establishes an encrypted connection that prevents tapping or distortion of the communication but does not provide access protection. This means you have to protect your CPU from unauthorized access with the corresponding configuration in the user management.

Display classes of the messages

Display classes of the messages

All display classes of the messages are activated in the basic configuration. The messages for the selected display classes are displayed during operation on the "Messages" website. Messages for display classes that are not selected are shown as hexadecimal code and not as plain text.

How to configure the display classes:

- for "Report System Errors" in the Inspector window under "Properties > General > System diagnostics"
- for module-related messages in the message editor

Note

Reducing memory requirements

You can reduce the memory requirements for the web server configuration by selecting only the display classes of the messages that should be used.

Creating and loading user pages

What you need to know about user pages

Concept

The concept of user pages or of user-defined web pages allows you to access freely-designed web pages of the CPU with a web browser. The Web server of the CPU provides this function.

You are not dependent on special tools for the design and functionality of the user-defined web pages. You can adapt the pages in the layout with CSS, provide dynamic content with JavaScript or use any framework to produce web pages. The totality of files processed by the Web server is also referred to as the "web application".

Web application and user program

Using HTML code in user-defined web pages (AWP commands), you can also transmit data via a web browser to the user program of the CPU for further processing and can display data from the operand area of the CPU in the web browser.

You can use script instructions (such as Javascript) to optimize your web pages, for example to dynamically change contents or validate user entries.

To synchronize between the user program and the Web server, but also to initialize, you must call the WWW (SFC 99) instruction in the user program.

- If no interaction is required between the web application and the user program, for example, if a web page only provides static information, only initialization in the user program is required.
- If a simple data exchange is necessary between PLC tags and tags in web applications, to display the contents of PLC tags or write a value in a PLC tag for example, the syntax for reading and writing tags has to be observed. In this case only an initialization is required in the user program, for example in the startup OB.
- If a further interaction is required between the web application and the user program, you must handle status and control information from the Web Control DB in addition to the synchronization between Web server and user program. This is the case, for example, when user entries are transmitted via the web browser to the Web server for evaluation by the CPU. Unlike simple data exchange, the user program directly influences the time at which the requested web page is relayed back to the web browser. In this case, you must be acquainted with the concept of manual fragments and the structure of the Web Control DB.

Integration of the HTML sources via the TIA Portal

The parameters for an integration of the HTML sources in the TIA Portal are located in the properties of the respective CPU with Web server:

Parameter	Meaning
HTML directory	Directory of the HTML sources for the web applications. You can input the path directly or navigate to the directory by using the button next to the entry field. It is advantageous if you integrate your Web pages with relative paths. To do this you have to insert the directory in which you store the HTML sources for the Web application in the STEP 7 project directory, for example, in a new directory "Webpages". You then specify the relative path name as the HTML directory; in the example ".\Webpages". Advantages of using relative paths: • The user-defined Web pages are archived together with the STEP 7 project. • The Web pages are copied to the new path with the menu command "Project > Save as...". • When copying the project to another path using the Windows Explorer, it is no longer necessary to adapt the HTML directory path to generate new Web DBs. • Several control systems in a project that use the same HTML pages also have the same HTML directory path information.
Start HTML page	Path to the Start HTML page. You can input the path directly or navigate to the HTML page with the button next to the entry field. The start HTML page is the HTML page to be opened at the start of the web application.
Application name	Optional name for the application. This name is used to further divide or group the web pages. When an application name already exists, the URL is displayed in the following format:"http://a.b.c.d/awp/<Application name>/<Page name>.html"

Initialization

User-defined web pages are "packaged" in data blocks for processing by the CPU. You must generate appropriate data blocks from the source data (HTML files, images, JavaScript files, etc.) during configuration to be able to download the web application into the CPU. The Web Control DB takes on a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. Data blocks that contain coded web pages are termed "Fragment DBs".

When the data block is downloaded into the CPU, the CPU does not "know" that user-defined web pages are coded inside it. The "WWW" (SFC 99) instruction, for example, in the Startup OB informs the CPU which DB is the Web Control DB. The user-defined web pages can be accessed via a web browser after this initialization.

Synchronization

If the user program is to exchange data with the user-defined web pages, the WWW (SFC 99) instruction must be used in the cyclic program section.

Examples of interaction between user program and web page:

- Check received data
- Assemble and send back data to the web browser making the request

In this case, the status information must be able to be evaluated and control information must be transmitted to the Web server, for example, to release a requested web page.

See also

Additional information and sample applications (Page 1288)

S7-1500 web server (http://support.automation.siemens.com/WW/view/en/59193560)

Procedure overview

Basic

In this chapter, the basic procedure is explained step-by-step to create, load and use user-defined websites in the operating phase.

The following image displays the procedure for the creation and display of user-defined websites:

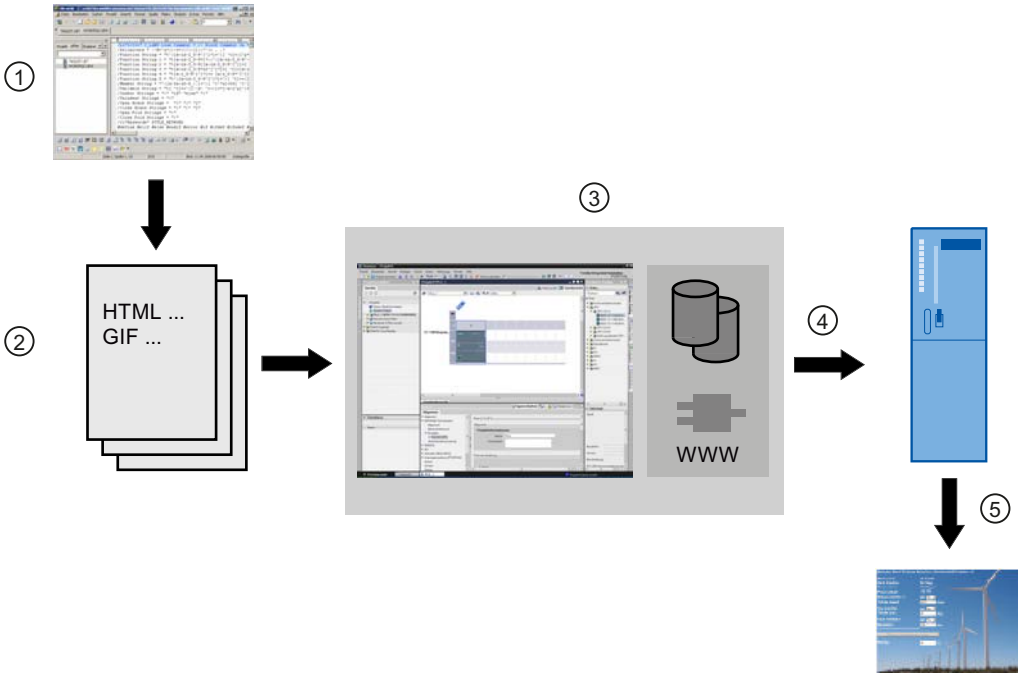

① Programming a web application (using suitable tools when required and AWP commands for dynamic pages when applicable).

② Web application consists of individual source files, for example, *.html, *.gif, *.js, ...

③ With STEP 7:

- Generating the data blocks (web-control-DB and fragment-DBs) from source files. The DBs contain meta information and the complete web application including the images and the dynamic and static parts of the web application. The DBs are stored in the project navigation under "System blocks".
- Open the "WWW" (SFC 99) instruction in the user program. The instruction initializes the web server of the CPU for a web application.
- If needed, program out interaction between web server and user program

④ Load the modules in the CPU.

⑤ Open website in the browser. The opening of websites from the CPU occurs through the Entry of the IP address of the CPU.

See also

Additional information and sample applications (Page 1288)

Creating web pages

Web design tools from various companies can be used to create user-defined web pages. As a rule, the web pages should be programmed and designed compliant to the conventions of the W3C (World Wide Web Consortium). No check is made for compliance to W3C criteria in the web server of the CPU.

Rules

- The tool must be able to directly edit the HTML code so that the AWP command can be inserted into the HTML page.
  Only the AWP commands are parsed in the CPU and, for example, replaced by values from the user program/process image of the CPU.
- Files containing AWP commands must be coded in UTF-8. In the metadata of the HTML page, therefore, set the attribute charset to UTF-8 and save the file UTF-8 coded.
- Files containing AWP commands must not contain the following sequence:]]
- Files containing AWP commands must not contain the following sequence outside of the "Tag read ranges" (:=<Tag name>:): :=
  Tip: Replace the first character of a prohibited sequence with its character coding; for the colon, for example, :.

A small example for a custom web page should make clear the basic design.

Requirement

- The CPU must have a web server and the web server of the CPU must be activated.
- To be able to access PLC tags with write access as a user, you must be logged on as "admin".
- For the example below, PLC tags must be defined for those PLC tags that are to be shown on the web page. This is shown here for the first tab used, "Tank_below_max".

		Name	Data type	Address
1		Tank_below_max	Bool	%I0.0

Creating user-defined web pages

The following code for an example web page reads values from the process image and provides them in a table.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/
TR/html4/strict.dtd">
<html>
  <head>
   <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
   <title>Mix</title>
  </head>
   <body>
    <h1>Mix</h1>
    <h2> Actual State </h2>
    <table border="1">
     <tr>
      <th>Variable</th>
      <th>State</th>
     </tr>
     <tr>
      <td>Tank below max</td>
      <td>:="Tank_below_max":</td>
```

```
      </tr>
      <tr>
       <td>Tank above min</td>
       <td>:="Tank_above_min":</td>
      </tr>
     </table>
    </body>
</html>
```

See also

Additional information and sample applications (Page 1288)

User page as start page and size recommendations

Defining the user page as start page

Defining the user page as start page

In addition to the default intro page, you can also define the start page of your user pages as the start page of the Web server.

Requirements

- You have configured a user in STEP 7 whom you have assigned at least the authorization "... open user-defined web pages".
- The CPU is in RUN mode.

Procedure

Proceed as follows to define the user pages in STEP 7 as start page of the Web server:

1. Select the CPU in the device configuration.
2. Open the settings in the Inspector window of the CPU under "Properties > General > Web server".
3. Select the entry "AWP1" in the area "Entry page" under "Select entry page".

If you now enter the IP address of the CPU in the browser, a connection is automatically established to your user pages.

If you want to access the web pages of your CPU again, link the web pages from your user pages, via the URL "http://a.b.c.d./Portal/Portal.mwsl?PriNav=Start" or "https://a.b.c.d/Portal/Portal.mwsl?PriNav=Start" for example. In this case, the information "a.b.c.d" represents, as an example, the IP address of the configured CPU.

Example of link in HTML:

```
<a href="/Portal/Portal.mwsl?PriNav=Start">SIMATIC web pages</a>
```

Note

If you define your user page as start page of the Web server, all direct access to the web pages of the CPU is disabled. This applies also to the bookmarks you saved for the web pages of the CPU as well as the page for reading out the service data.

Reading out service data

If you define your user page as start page of the Web server, all direct access to the web pages for reading out the service data is also disabled.
If you want to continue to read out service data via the Web server when servicing is required, here is how you can link the service data page directly to your user page.
Just as for the web pages of the CPU, link the service data page e.g. via the URL "http://a.b.c.d/save_service_data" or "https://a.b.c.d/save_service_data", the "a.b.c.d" here is an example of the IP address of the configured CPU.

Example of link in HTML:

```
<a href="/save_service_data">Service data</a>
```

Reference

You can find additional information on the topic of user page as start page in the FAQ with entry ID 67184104 on the Online Support (https://support.industry.siemens.com/cs/ww/en/view/67184104) Internet page.

Recommendations for the size of Web pages

Influence of the Web page size on the required memory and performance

Extensive HTML pages, especially those with a lot of images, take up a lot of space in the load memory. Note the following tips:

- Select a SIMATIC Memory Card with enough memory to provide sufficient load memory.
- Limit the data volume of the HTML pages to 1 MB.
  When the data volume of the HTML pages exceeds 1 MB, performance losses may occur as only 1 MB data is saved in the cache.
- Limit the size of a file of an HTML page to 512 kB.
  When a single file of an HTML page exceeds 512 kB, problems can occur when sending the file from the Web server to the browser.

Have the file sizes displayed in the file explorer of the directory in which the Web application lies. Where necessary and possible, reduce the file sizes.

AWP commands for controlling Web pages

Using AWP commands in Web pages

With AWP (Automation Web Programming) commands, you declare the interface between your user pages (Web application, e.g. a simple HTML page) and the CPU data.

To develop user pages or Web applications, you need only observe the restrictions of the Web browser. In one of the programming languages of STEP 7, control with the user program in the CPU which CPU data is displayed at what time in the Web browser of the viewer.

Use AWP commands, which you comment within the HTML page, to declare data to be used for intentional interaction between the Web application and the user program.

You insert AWP commands as HTML comments with a special syntax into HTML pages. Using AWP commands, the following features can be implemented:

- Read PLC tags
- Write PLC tag
- Read special tags
- Write special tags
- Define enum types
- Assign tags to enum types
- Defining fragments
- Import fragments

Syntax of AWP commands

An AWP command begins with "`<!-- AWP_`" and ends with "` -->`". In JavaScript files, the commands should also be enclosed by JavaScript comments ("`/*...*/`").

Notation rules for PLC tag names within an AWP command

The AWP commands "AWP_In_Variable" and "AWP_Out_Variable" contain a name attribute and optionally a use attribute. A PLC tag name is assigned to these attributes, by means of which the PLC tags in the browser are written or read. The following rules apply to handling PLC tag names in HTML code:

- PLC tags must be enclosed in quotation marks (" ... ").
- PLC tags used in AWP commands must also be enclosed by single quotation marks ("' ... ''") or with quotation marks masked by a backslash ("\" ... \"").
- If the PLC tag name contains the character \ (backslash) or * (star), this character must be designated with the escape sequence \\ as standard character of the PLC tag name. See below for examples.

- If the PLC tag name in the AWP command is also enclosed by single quotation marks and the single quotation mark (') occurs within the name, it must also be designed as normal character by the escape sequence \'.
- If an absolute address (input, output, bit memory) is used in AWP command, it is enclosed by single quotation marks.

PLC tag	PLC tag in HTML code
"Velocity"	<!-- AWP_In_Variable Name='"Velocity"' -->
	<!-- AWP_In_Variable Name="\"Velocity\"" -->
"abc\de"	<!-- AWP_In_Variable Name='"abc\\de"' -->
"abc'de"	<!-- AWP_In_Variable Name='"abc\'de"' -->
"abc'de"	<!-- AWP_In_Variable Name="abcde" Use'"abc\'de"' -->
"DB name".tag	<!-- AWP_In_Variable Name='"DB name".tag' -->
"Plc1".Data[1].typeDataS-truct.value	<!-- AWP_In_Variable Name='"Plc1".Data[1].typeDataStruct.value' -->
-	<!-- AWP_Out_Variable Name=’flag1’ Use='M0.0' -->

See also

Additional information and sample applications (Page 1288)

Reading tags

User-defined web pages can read PLC tags.

The PLC tag must be specified by a PLC tag name.

These OUT variables (direction of output as viewed from the controller) are inserted at any location within the HTML text with the syntax described in the following.

Syntax

```
:=<varname>:
```

These references are replaced when the Web server is in operation by the current values of the PLC tag in each case.

`<varname>` can be a simple, global PLC tag but also a complete tag path to a structural element.

Notation rules for PLC tag names

- PLC tags in HTML code are enclosed by quotation marks ("), if they are defined in the tag table. In the case of data block tags, the name of the data block is enclosed by quotation marks. If special characters are used in the structure elements of the data block, for example the dot (.) or blank, this part must also be enclosed by quotation marks.
- Quotation marks are not used for absolute addresses of inputs, outputs or bit memories.

PLC tag	PLC tag in HTML code
"DB_name".var_name	:="DB_name".var_name:
"DB_name"."var.name"	:="DB_name"."var.name":
"memory"	:="memory":
-	:=I0.0:
	:=Q0.0:
	:=MW100:
	:=%MW100:
"My_Data_Block".flag1	<!-- AWP_Out_Variable Name='flag1' Use='"My_Data_Block".flag1' -->
	...
	:=flag1:

- If the PLC tag name contains the character : (colon) or \ (backslash), this character must be designated with the escape sequence \: or \\ as standard character of the PLC tag name.

PLC tag	PLC tag in HTML code
"abc:de"	:="abc\:de":
"abc\de"	:="abc\\de":

- Special characters "<, &, >"
  Display problems can occur if these characters are contained in the tag name (for example, "a<b").
  Avoid expressions such as :="a<b": in the HTML page.
  To prevent display problems from occurring, use e.g. an AWP command with a use expression according to the pattern depicted below. The use attribute defines the PLC tag with the problematic character, the name attribute defines the name without problematic character, as it is used in the HTML page.

PLC tag	PLC tag in HTML code
"a<b"	<!-- AWP_Out_Variable Name='simplename' Use='"a<b"' -->
	...
	:=simplename:

See also

Additional information and sample applications (Page 1288)

Particular points for PLC tags of the String and Character type

Reading tags of the String and Character type

Below, these types of quotation marks are used in the explanation: single quotation marks ('), double quotation marks (").

As of firmware V1.6, with the "Read PLC tags" function, the CPU outputs tags of the String or Character type enclosed in single quotation marks to the browser.
For example:

- "Varname".MyString = ABC string tag
- You read the tag in HTML using the function :="Varname".MyString:
- The Web server outputs the character string 'ABC' to the browser

Using String or Character tags in expressions

On your HTML page, you use an expression in which the character string for reading a tag is enclosed in quotation marks, for example in forms.

Possible HTML code used:
```
<input type="text" name="appfield" value="myvalue">
```

If you read the displayed value for the "value" attribute from a PLC tag in this expression, the HTML code appears as follows:
```
<input type="text" name="appfield" value=":="Varname".MyString:">
```

By reading the PLC tag, the Web server outputs the value 'ABC'. In HTML, the code is then represented as follows:
```
<input type="text" name="appfield" value=" 'ABC' ">
```

If you have used single quotation marks instead of double quotation marks in your HTML code to enclose the attributes, the Web server passes on the content of the tags enclosed in two single quotation marks to the browser. As a result of this, the browser does not output the content of the String or Character tag, since two consecutive single quotation marks each form a closed sequence. The values to be read are located between these sequences and are not output by the browser.

In this context, note in particular that the character string with double quotation marks is not identical to two single quotation marks even if they appear to be identical.

Note

The code is not adapted automatically during an update to firmware as of V1.6.
Adapt your HTML code if you have used single quotation marks to enclose attributes for the "Read PLC tags" function.

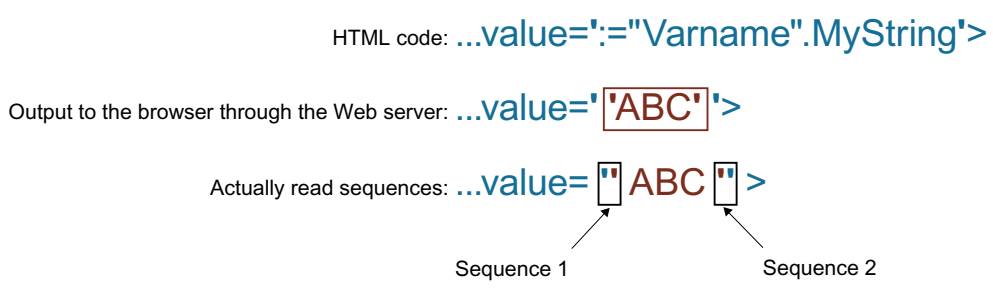

Writing tags

Custom web pages can write data into the CPU.

This requires an AWP command that identifies the PLC tag to be written.

The PLC tag must also be specified by a PLC tag name.

The IN tags (direction of input as viewed from the controller) are placed on the browser page. This can be done, for example, in a form.

The tags are either set in the HTTP header (by cookie or POST method) or in the URL (GET method) by the browser and are then written by the web server into the respective PLC tag.

Syntax

To allow the IN tags to be written to the CPU, the tags must first be defined by an explicit AWP instruction:

```
<!-- AWP_In_Variable Name='"<Varname1>"' Name='"<Varname2>"'
Name='"<Varname3>"' -->
```

Several tags can be defined in an instruction - such as that shown above.

In cases where the name of the tag that you use for the web application is not identical to the name of the PLC tag, the "Use" parameter can be used to assign to a PLC tag.

```
<!-- AWP_In_Variable Name='<Varname_Webapp>' Use='<PLC_Varname>'
```

Example

The "AWP_In_Variable" AWP command is indispensable when handling forms.

```
<form method="post" action="/awp/appl/x.html">
 <p>
  <input name='"var1"' type="text">
  <input value="set" name='"Button1"' type="submit">
 </p>
</form>
```

In the form defined above, tags "var1" and "Button1" are transferred to the web server by "post". The user places the "var1" tag in the form field. The "Button1" tag has the fixed value "set". To allow the "var1" and "Button1" tags to be written to the CPU, the following instruction must be included in the same fragment:

```
<!-- AWP_In_Variable Name='"var1"' Name='"Button1"' -->
```

Because global tags are enclosed in quotation marks, the name must be enclosed in the AWP command by apostrophes or by masked quotation marks (\").

```
<!-- AWP_In_Variable Name='"Info".par1' -->
```

Conditions for write access during operation

The following requirement has to be met in order for a user to be able to write to PLC tags from a user-defined web page.

- A user was set up and is logged on.

This rules applies to all writing access to web pages on a CPU.

See also

User administration (Page 1253)

Additional information and sample applications (Page 1288)

Special tags

Special tags are mainly HTTP tags set in the definition of the World Wide Web Consortium (W3C) . Special tags are also used for cookies and server tags.

The AWP command to read and write special tags differ only in that they have additional parameters than the AWP command used to read and write normal tags.

Reading a special tag

The Web server can read PLC tags and transfer these to special tags in the HTTP Response Header. You can, for example, read a URL for a diversion to another web page and transfer to the special tag HEADER:Location using the special tag HEADER:Location.

The following special tags can be read:

Name	Description
COOKIE_VALUE:name	Value of cookie with name: "name"
COOKIE_EXPIRES:name	Execution time of cookie with name: "name" in seconds (must be set beforehand).
HEADER:Status	HTTP status code (if no other value has been set, status code 302 is returned).
HEADER:Location	Path for forwarding to another page. Status code 302 must be set.
HEADER:Retry-After	Anticipated time in which the service is not available. Status code 503 must be set.
HEADER: …	All other header tags can also be forwarded in this way.

Use the AWP command "AWP_Out_Variable" to specify which PLC tags are to be transferred in the HTTP header to the web browser.

Basic structure:

```
<!-- AWP_Out_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description

- Name: Type and name of special tag
- Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, parameter "Use" can be used to assign to a PLC tag.

Example:

```
<!-- AWP_Out_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

Writing a special tag

In principle, all HTTP tags written in the HTTP header by the web browser can be evaluated by the user program of the CPU. Examples of tag types:

Name	Description
HEADER:Accept-Language	Accepted or preferred language
HEADER:Authorization	Proof of authorization for a requested resource
HEADER:Host	Host and port of the requested resource
HEADER:User-Agent	Information on the browser
HEADER: …	All other header tags can also be forwarded in this way
SERVER:current_user_id	Indicates whether a user is logged on (current_user_id=0: no user logged on)
SERVER:current_user_name	User name of the user logged on
SERVER:GET	Request method is GET
SERVER:POST	Request method is POST
COOKIE_VALUE:name	Value of cookie with name: "name"

The AWP command "AWP_In_Variable" is used to define which special tags are to be evaluated in the user program of the CPU.

Basic structure:

```
<!-- AWP_In_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description:

Name: Type and name of special tag

Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, the parameter Use can be used to assign to a PLC tag.

Examples:

```
<!-- AWP_In_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use . The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='COOKIE_VALUE:siemens' Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use. The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='"COOKIE_VALUE:siemens"' -->
```

The HTTP-header variable is written in the same-name PLC variable.

See also

Additional information and sample applications (Page 1288)

Enumeration types

Enumeration types (enums)

Numerical values from the PLC program can be converted into text and vice versa using enums. The numerical values can also be assigned for several languages.

Creating enums

Enter an AWP command using the following syntax at the start of the HTML file:

```
<!-- AWP_Enum_Def Name="<Name of the enum type>"
Values='0:"<Text_1>", 1:"<Text_2>", ... , x:"<Text_x>"' -->
```

For example, for German values to be saved as an HTML file in the "de" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"an", 1:"aus", 2:"Störung"' -->
```

For example, for English values, to be saved as an HTML file in the "en" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"on", 1:"off", 2:"error"' -->
```

Assigning enums

Tags are assigned from the user program to the individual enum texts using a special AWP command:

```
<!-- AWP_Enum_Ref Name="<VarName>" Enum="<EnumTypeName>" -->
```

<VarName> is thereby the symbolic name from the user program and <EnumTypeName> is the previously set name of the enum type.

Note

In each fragment in which enum texts are referenced by a PLC tag, this PLC tag must be assigned by the appropriate AWP command of the enum type name.

Ensure that no AWP command for importing fragments is positioned between an enum assignment and enum usage because this import can result in the enum assignment lying in a different fragment than the enum usage.

Example

Enum type "state" is defined with values "0" and "1". "0" means "off", "1" means "on":

```
<!-- AWP_Enum_Def Name="state" Values='0:"off", 1:"on"' -->
```

The following code is contained in the HTML code of the web page to be output:

```
<!-- AWP_Enum_Ref Name="operating state" Enum="state" -->
:=operating state:
```

Depending on the value of the "operating state" tag, the ´result displayed is no longer "0" or "1", but "off" or "on".

Simplified use of enumeration types

At S7-1200 CPUs as of firmware version 4 it is possible to use enumerations directly in AWP commands to read and write PLC variables.

You create enums as described in the previous section, and you can then utilize the values with user program read and write commands.

Creating enums

```
<!-- AWP_Enum_Def Name="<Name des Enum Typs>" Values='0:"<Text_1>",
1:"<Text_2>", ... , x:"<Text_x>"' -->
```

Utilizing enums in the user program read and write commands

```
<!-- AWP_In_Variable Name='<Varname>' Enum="<EnumType>" -->
<!-- AWP_Out_Variable Name='<Varname>' Enum="<EnumType>" -->
```

Example of reading PLC tags

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' --><!-- AWP_Out_Variable Name='"Alarm"'
Enum="AlarmEnum" -->...<p>The current value of "Alarm"
is :="Alarm":</p>
```

If the value of "Alarm" in the CPU is "2", the following text will be displayed on the HTML page:

'The current value of "Alarm" is Tank is empty' because the enum definition assigns the string "Tank is empty" to the numerical value 2.

Example of writing PLC tags

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' --><!-- AWP_In_Variable Name='"Alarm"'
Enum='AlarmEnum' -->...
<form method="POST">
<p><input type="hidden" name='"Alarm"' value="Tank is full" /></p>
<p><input type="submit" value='Set Tank is full' /></p>
</form>
```

Because the enum definition assigns the string "Tank is full" to the numerical value "1", the value "1" is written to the PLC tag "Alarm".

Using arrays

The Web server provides the user program commands AWP_Start_Array and AWP_End_Array for accessing all values of an array.

Only one-dimensional arrays are supported.

Multidimensional arrays of form array[x][y] are not supported.

Syntax

```
<!-- AWP_Start_Array Name='"<DB name>".<array name>' -->
... Content of the array, utilized keywords: ArrayIndex and value..
<!-- AWP_End_Array -->
```

Parameter

<Name>	Defines the name of the array whose elements you want to access. You require the DB name and the name of the array corresponding to the data block structure defined in STEP 7. The name must be within single or double quotation marks. The DB name is within double quotation marks.
<ArrayIndex>	Index of an array element
<value>	Value of an array element

Example

The example reads all elements of the "MyArray" structure in the "DB_Name" data blocks of the CPU and displays the index and the values of the tags on the user-defined web page.

DB_Name

	Name	Data type	Start value
1	Static		
2	myArray	Array [0..2] of Int	
3	myArray[0]	Int	42
4	myArray[1]	Int	43
5	myArray[2]	Int	44

```
<!-- AWP_Start_Array Name='"DB_Name".MyArray' -->
Index: :=ArrayIndex: Value: :=value:
<!-- AWP_End_Array -->
```

The code indicated above generates the following display:

Index: 1 Value: 42

Index: 2 Value: 43

Index: 3 Value: 44

Representation of arrays of the data type BOOL with S7-1500

The output of arrays of the type BOOL is always filled to the next full 8 bits. This particular feature only occurs with BOOL arrays.

Example:

```
"DB_1".bitArray is a BOOL array with 5 elements.
<!-- AWP_Start_Array Name='"DB_1".bitArray' -->
:=ArrayIndex: -> :=value:
<!-- AWP_End_Array -->
```

Output:

0 -> value from "DB_1".bitArray[0]

1 -> value from "DB_1".bitArray[1]

2 -> value from "DB_1".bitArray[2]

3 -> value from "DB_1".bitArray[3]

4 -> value from "DB_1".bitArray[4]

5 > 0

6 > 0

7 > 0

See also

Additional information and sample applications (Page 1288)

Using structures

The Web server provides user program commands for accessing structures in order to access the values of a PLC tag of data type STRUCT.

Syntax

```
<!-- AWP_Start_Struct Name='"<DB name>".<struct name>' -->
... Content of structure ...
<!-- AWP_End_Struct -->
```

Parameter

<Name>	Defines the name of the structure whose elements you want to access. You require the DB name and the name of the structure corresponding to the data block structure defined in STEP 7. The name must be within single or double quotation marks. The DB name is within double quotation marks.

Example

The example reads elements of the "MyStruct" structure in the "DB_Name" data blocks of the CPU and displays the value of the tag on the user-defined web page.

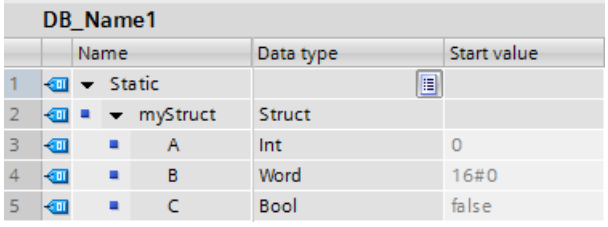

DB_Name1

	Name	Data type	Start value
1	Static		
2	myStruct	Struct	
3	A	Int	0
4	B	Word	16#0
5	C	Bool	false

```
<!-- AWP_Start_Struct Name='"DB_Name".MyStruct' -->
:=A:
:=B:
:=C:
<!-- AWP_End_Struct -->
```

The code indicated above corresponds to the following commands:

```
:="DB_Name".MyStruct.A:
:="DB_Name".MyStruct.B:
:="DB_Name".MyStruct.C:
```

Rules for arrays and structures

Arrays and structures can be nested.

Rules

AWP_Start_Array and AWP_End_Array commands must be used in pairs. They must not be overlapped.

Permissible construction:

AWP_Start_Array

AWP_Start_Struct

AWP_End_Struct

AWP_End_Array

Nonpermissible construction (overlapping):

AWP_Start_Array

AWP_Start_Struct

AWP_End_Array

AWP_End_Struct

Definition of fragments

Fragments

Fragments are "logical sections" of a web page to be processed by the CPU individually.

Fragments are usually complete pages but can also be individual elements such as files (for example, images) or complete documents.

Defining fragments

```
<!-- AWP_Start_Fragment Name="<Name>" [Type="<Type>"] [ID="<Id>"]
[Mode=<Mode>]-->
```

The start of a fragment is specified by this command. A fragment runs to the start of the next fragment or to the end of the file.

- <Name> Indicates the name of the fragment.
  The name must start with a letter [a-zA-Z] or an underscore (_). Letters, underscores or numbers [0-9] can follow after this first character.
- <Type> Indicates the type of the fragment.
  - "manual" The user program is informed of the request for a fragment; the web page to be returned can be changed by the user program.
  - "automatic" The page is automatically processed (default).

- <id> A numeric ID can be stipulated for the fragment. If no ID is assigned, the fragment is automatically assigned an ID. For manual pages (<Type>=manual) , the fragment can be addressed in the user program of the CPU by this ID.

  Note

  Keep the ID low because the highest ID influences the size of the Web Control DB.

- <Mode> Fragments support the visible and hidden modes.
  - "visible" The fragment is a part of the web page. This mode is preset and can also be omitted.
  - "hidden" The fragment is not part of the web page. However, the fragment will be saved in the Web DB and is available to the user program for inserting in a requested web page. You use an exchange of the fragment ID (Web-Control-DB.fragment_index tag) to insert a "hidden" fragment in the requested web page.

The input document is completely divided into fragments by the "AWP_Start_Fragment" command. "AWP_End_Fragment" is therefore unnecessary.

Without a start fragment command, a file is mapped as a fragment; the fragment name is derived from the file name. If a file is divided into several fragments (by "AWP_Start_Fragment"), the file must begin with the "AWP_Start_Fragment" command.

See also

Additional information and sample applications (Page 1288)

Importing fragments

You can declare a fragment in an HTML page and import this fragment into other web pages.

Example

A company logo is to be displayed on all web pages of a web application.

There is only one instance of the HTML code for the fragment that displays the company logo. You can import the fragment as often and into as many HTML files as required.

Syntax

```
<!-- AWP_Import_Fragment Name = "<name>"-->
```

- <name> is the name of the fragment to be imported.

Example

HTML code within a web page that declares a fragment:

```
<!-- AWP_Start_Fragment Name = "My_Company_Logo" -->
<p><img src = "compay_logo.jpg"></p>
```

Example

HTML code within another web page that imports the declared fragment:

```
<!-- AWP_Import_Fragment Name = "My_Company_Logo" -->
```

Creating and loading a data block

Requirement

- All source files required for the web application (*.html, *.js, *.png, ...) have been created.
- The source files are located in one folder, but only those source files that are required for the web application. No other files may be located in this folder.

Note

Length of file names and tag names

If you have a comprehensive web application with many files and directories, the generation of the web data blocks may possibly fail. If this happens, the generation is aborted with the message "Text list overflow...". The cause is system-internal size limitations for management information saved in the web data block.

Remedy: Use short file names and short tag names.

Procedure

To create data blocks from the source files for user-defined web pages in STEP 7, proceed as follows:

1. Select the CPU, for example, in the device configuration.
2. Select the properties for user-defined web pages in the inspector window under "Properties > General > Web server".
3. As "HTML source", select the folder that contains the source files for the web application.
4. Enter the HTMP page to be opened on starting the web application as the start HTML page.
5. Enter a name for the application if required.

6. You can supplement a range of file name extensions as "Files with dynamic content" if necessary. Only enter those file name extensions that also contain AWP commands.
7. The number for the Web Control DB and for the fragment DB start number can be kept as long as they are not already being used by your user program.
8. Click on the "Generate" button to create DBs from the source files.
   The generated data blocks are saved in the project navigation in the "System block" folder (in the "Web server" subfolder).
9. In the CPU, select the network view to be loaded and then select the "Download to device" command in the "Online" menu to download the blocks. Compilation of the blocks is implicitly initiated before downloading.
   If errors are reported during this process, you must correct these errors before you can download the configuration.

Structure of the PLC program

Your user program must call the " WWW" instruction to even allow the web application, for example, the user-defined web pages, to be available to the CPU on the standard web pages and to allow them to be called up there.

The Web Control DB you have created from the source files is the input parameter (CTRL_DB) for the "WWW" instruction. The Web Control DB references the content of the user-defined web pages coded in the fragment DB and then receives status and control information.

Calling the "WWW" instruction in the startup program

If you do not want the user program to influence requested web pages, it is sufficient to only call the "WWW" instruction once in a startup OB. This instruction initializes communication between the web server and the CPU.

Calling the "WWW" instruction in the cyclic program

The "WWW" instruction can also be called in an OB processed in cycles (for example, OB 1). This has the advantage of being able to respond to web server requests from within the user program. Manual fragments must be used for this.

In this case, you must evaluate information from the Web Control DB in order to identify the requested web page or the requested fragment. On the other hand, you must set a bit in the user program in order to explicitly release the web page to be returned by the web server after processing the web page request.

The structure of the Web Control DB is described in the following section.

Web Control DB

The Web Control DB (DB 333 by default) is created by STEP 7 and contains information on the structure of user pages, the status of communication and any errors that occur.

Additional fragment DBs are also created as well as the Web Control DB. These fragment DBs (there may also only be one fragment DB) are referenced in the Web Control DB. The fragment DBs contain the web pages and media data coded in fragments, for example, images. The

content of the fragment DB cannot be changed by the user program. It is created automatically and is only for data management.

The status and control tags of the Web Control DB are accessed via symbols.

The following lists the tags of the Web Control DB required for status evaluation and to control interaction.

The Web Control DB provides two types of information:

- Global status information: Not bound to a concrete web page request.
- Request status and control information: Information about queued requests.

Global status information

"WEB-Control_DB".commandstate.init	Activates and initializes the web application.
"WEB-Control_DB".commandstate.deactivate	Deactivates the web application.
"WEB-Control_DB".commandstate.inititializing	The web application is initialized (read Web Control DB, etc.).
"WEB-Control_DB".commandstate.error	Web application could not be initialized. The reason is coded in "WEB-Control_DB".commandstate.last_error .
"WEB-Control_DB".commandstate.deactivating	The web application is closed.
"WEB-Control_DB".commandstate.initialized	The web application has been initialized and is ready.
"WEB-Control_DB".commandstate.last_error	Refer to the next table for a value table of possible errors.

Last_error	Description
1	Fragment DB is inconsistent (does not match the Web Control DB).
2	A web application already exists with this name.
3	Memory problem initializing in the web server.
4	Inconsistent data in the Web Control DB.
5	A fragment DB is not available (not loaded).
6	No AWP ID for a fragment DB.
7	The enum fragment is not available (contains the texts and information on the enum types).
8	An action requested via the command flag in the Web Control DB is prohibited in the current state.
9	Web application is not initialized (if there is no reinitializing after disabling).
10	Web server is disabled.
...	Last_error is reset once the web application has been successfully initialized.

Request status information

Request status information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].idle	Nothing need be done.
"WEB-Control_DB".requesttab[x].waiting	The user program must react to a request from a manual fragment and explicitly initiate further processing in the web browser.
"WEB-Control_DB".requesttab[x].sending	The web server is occupied with processing the request/fragment.
"WEB-Control_DB".requesttab[x].aborting	The TCP connection is closed by the web server.

Request control information

Request control information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].continue	Releases the fragment being processed for transmission. Processing of the next fragment is initiated.
"WEB-Control_DB".requesttab[x].repeat	Releases the fragment being processed for transmission. The fragment is then processed again.
"WEB-Control_DB".requesttab[x].abort	Closes the TCP connection.
"WEB-Control_DB".requesttab[x].finish	Releases the fragment being processed for transmission. Stops further processing of requests (terminates the request).

Example:

The tag for the DB is: "WEB-Control_DB". Whether errors have occurred during initialization of the web application can be determined by requesting bit "WEB-Control_DB".commandstate.error in the user program.

If an error has occurred you can analyze it using the "WEB-Control_DB".commandstate.last_error value.

Interaction with the user program

With the help of manual fragments, you can make sure that the user program reacts synchronously to browser entries so that the returned website can be prepared by the user program.

Fragment type

To react directly to the received data in the user program, the "manual" fragment type (for "manual sites") must be used for the fragment that the data writes:

```
<!-- AWP_Start_Fragment Name="testfrag" ID="1" Type="manual" -->
```

The values are always transferred to the Web server of the CPU for automatic and manual pages in the same way:

Example:

```
<form method="POST" action="">
<p>
<input type="submit" value="Set new value">
<input type="text" name='"Velocity"' size="20">
</p>
</form>
```

User program for manual fragments

In the CPU user program, the "WWW" (SFC 99) instruction must be cyclically opened with the usage of manual sites.

To react to the values entered in the browser, it is required to evaluate the request that the manual site places for the Web server in the user program. The web-control-DB (for example, DB 333) must investigate pending requests to do this. The array that manages four requests is contained in the "requesttab" section of the Web Control DB. Each element of the array contains information about the respective request in a structure.

A simple AWL program example shows how pending requests are checked provided the variables from the web-control-DB.

```
      U     "WEB-Control_DB".requesttab[1].waiting
      JCN R2
      L     "WEB-Control_DB".requesttab[1].fragment_index
      L     1
      JU     PdRq                        // request pending
R2:   U     "WEB-Control_DB".requesttab[2].waiting
      JCN R3
      L     "WEB-Control_DB".requesttab[2].fragment_index
      L     2
      JU     PdRq                        // request pending
R3:   U     "WEB-Control_DB".requesttab[3].waiting
      JCN R4
      L     "WEB-Control_DB".requesttab[3].fragment_index
      L     3
      JU     PdRq                        // request pending
R4:   U     "WEB-Control_DB".requesttab[4].waiting
      JCN  NoRq
      L     "WEB-Control_DB".requesttab[4].fragment_index
      L     4
      JU     PdRq                        // request pending
NoRq: L     0
      T     #frag_index
      T     #req_index
      JU     Exit
PdRq: T     #req_index
      POP
      T     #frag_index
Exit: BE
```

If there is a request present, this AWL program section writes the fragment ID in the variable #frag_index and the request number (value range 1-4) in the variable #req_index.

With this information, the information transferred in the request can now be processed in the AWL program for each fragment ID separately (for example, plausibility check).

If the processing of the request by the AWL program is complete, the request must be answered and the corresponding entry under "requesttab" from the web-control DB (for example, DB 333) must be reset.

A simple AWL program example for answering requests:

```
SET
       L     #req_index
      SPL   Con0
      JU     Con0
      JU     Con1
      JU     Con2
      JU     Con3
      JU     Con4
Con0: JU     Exit
Con1: =     "WEB-Control_DB".requesttab[1].continue
      JU     Exit
Con2: =     "WEB-Control_DB".requesttab[2].continue
      JU     Exit
Con3: =     "WEB-Control_DB".requesttab[3].continue
      JU     Exit
Con4: =     "WEB-Control_DB".requesttab[4].continue
      JU     Exit
Exit: BE
```

Principle sequence of a browser request with interaction from the user program

The following figure shows the simplified, principle sequence of the web browser request on the effects of Web Control DB content and the actions required from the user program until the processed web page is returned (response).

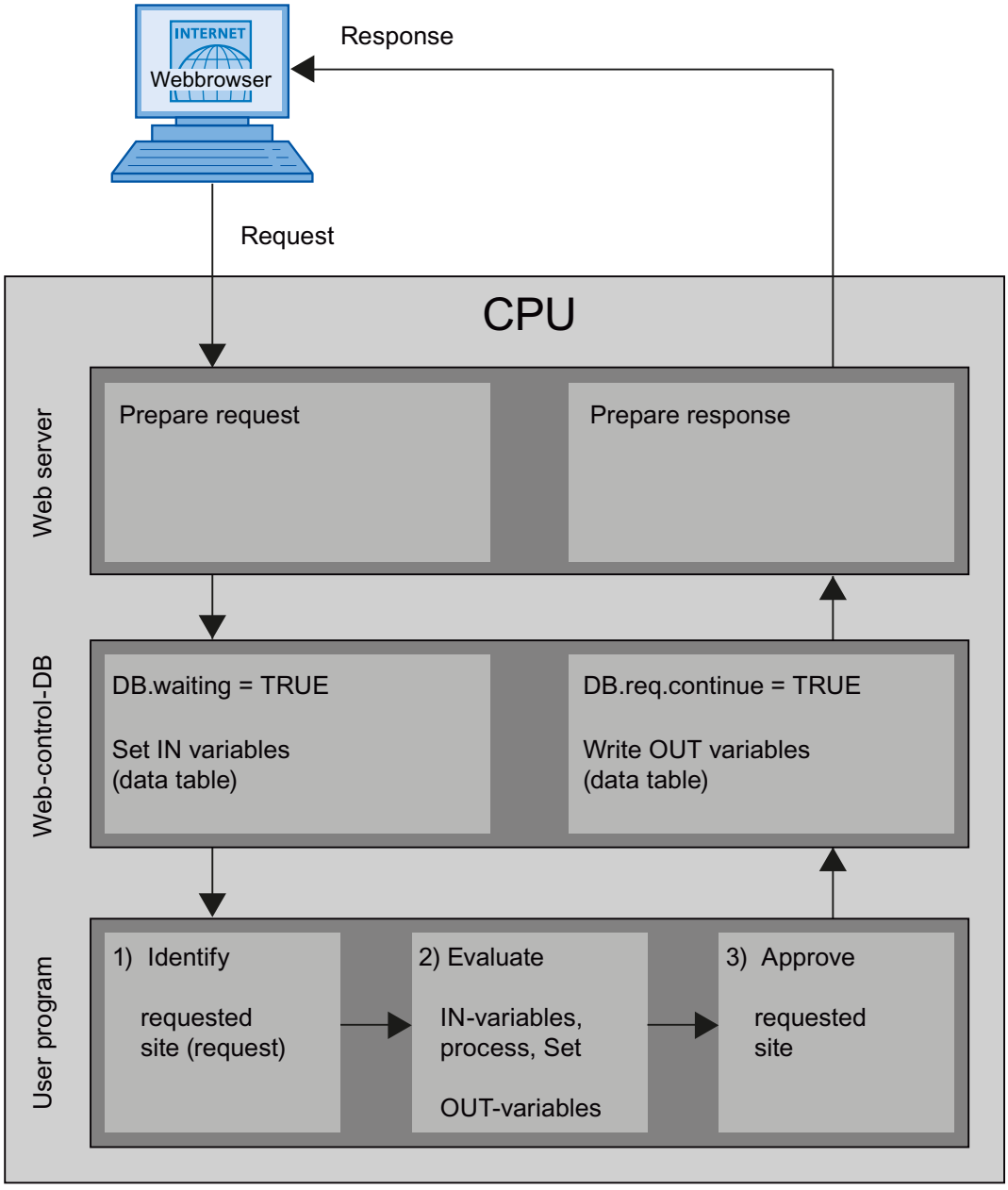

See also

Additional information and sample applications (Page 1288)

Displaying custom web pages in the browser

Display web pages in browser

Web pages are called from the standard web pages of the web browser.

In addition to the other links in the navigation bar, the standard web pages also have a link to "user pages".

Click on the "user pages" link to open the web browser you have configured as the default HTML page.

See also

Additional information and sample applications (Page 1288)

Create multilingual user defined websites

You have the possibility to provide your user-defined websites in the different languages.

Requirements

The language-dependent HTML sites will be filed in a folder structure that contains the following language abbreviations:

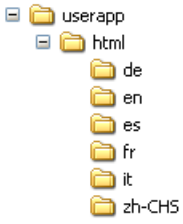

Defined language abbreviations

The language abbreviations "de", "en", "fr", "es", "it" and "zh-CHS" are defined. Other language folders or other language folders labeled differently are not supported.

If needed, you can create more folders in the same folder hierarchy for other files; for example, an "img" folder for images and a "script" folder for Java script files.

See also

Additional information and sample applications (Page 1288)

Language switching for custom web pages

Requirements

The HTML pages are contained in the predefined language folders, for example, HTML pages with German text are in the "de" folder, HTML pages with English text are in the "en" folder.

Language switching concept

Language switching is based on a predefined cookie named "siemens_automation_language". If the cookie is set to value "de", at the next web page request or web page update, the web server switches to the web page from the "de" folder.

Similarly, the web server switches to the web page from the "en" folder when the cookie is set to "en".

See also

Additional information and sample applications (Page 1288)

Example of a language switch

The example is designed as follows:

- In both language folders "de" and "en", you will find the language-dependent HTML files with the same names, for example, "langswitch.html". The text displayed within the two files is in German or English corresponding with the folder name.
- Additionally, there is a folder "script" in the folder structure in which the Java script file "lang.js" is located. Functions are stored in this file that are required for the language switch.

Design of the "langswitch.html" file ("de" folder)

The meta data "Content-Language", charset and path to the Java script file are set in the header of the file.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="de">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Language Switch German Site</title>
<script type="text/javascript" src="script/lang.js" ></script>
</head>
```

In the body of the file, the language selection occurs with the help of the HTML element "select". The select element activates a selection list and contains the options "de", labeled with "German" and "en", labeled with "English"; "de" is preselected.

With the help of the event handler "onchange", the function "DoLocalLanguageChange(this)" is opened. The "this" parameter transfers the select-object with the selected option to this function. "onchange" executes the option to open the function upon each change.

```
<!-- Language Selection -->
```

```
<table>
    <tr>
        <td align="right" valign="top" nowrap>
         <!-- change language immediately on change of the selection
-->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" selected >German</option>
                <option value="en" >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Design of the file "langswitch.html" (folder "en")

The header of the HTML file with English text is designed analog to the HTML file with German text.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="en">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Language switching english page</title>
<script type="text/javascript" src="script/lang.js" ></script>
```

In the body of the file, the language selection occurs with the help of the HTML element "select". In contrast to the German HTML file, the English option is preselected and the text and labels are in English.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
         <!-- change language immediately on change of the selection
-->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" >German</option>
                <option value="en" selected >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Design of the "lang.js" file (in the "script" folder)

The "DoLocalLanguageChange" function is defined in the Java script file, which opens the "SetLangCookie" function with the language selection value. SetLangCookie combines the cookie name and cookie value and sets the cookie through the corresponding document.cookie property. To make the web server react to the setting of cookies with the display of the desired language, the website must be reloaded (top.window.location.reload).

```
function DoLocalLanguageChange(oSelect) {
        SetLangCookie(oSelect.value);
        top.window.location.reload();
    }

 function SetLangCookie(value) {
        var strval = "siemens_automation_language=";
            // this is the cookie by which the web server
            // detects the desired language
            // this name is required by the web server
        strval = strval + value;
        strval = strval + "; path=/ ;";
            // set path to the application, since otherwise
           // path would be set to the requesting page would not get
the cookie
            /*
            use expiration if this cookie should live longer
            than the current browser session
            var now      = new Date();
            var endttime = new Date(now.getTime() + expiration);
            strval = strval + "; expires=" + endttime.toGMTString()
+ ";";
            */
        document.cookie = strval;
    }
```

See also

Additional information and sample applications (Page 1288)

Additional information and sample applications

Information and examples for user-defined pages

Examples and further explanations are available in the Siemens Industry online support.

- You can find the function manual for the Web server of an S7-1500 CPU here (http://support.automation.siemens.com/WW/view/en/62046437?Datakey=47071380).
- An example for accessing the Web server of an S7-1200 using an iPad and for integrating the jQuery library is available here (http://support.automation.siemens.com/WW/view/en/62046437?Datakey=47071380).

- An application example for the creation of a user-defined web page is available here (http://support.automation.siemens.com/WW/view/en/58862931).
- A description of the options for updating user-defined web pages with HTML5 code or JavaScript is available here (http://support.automation.siemens.com/WW/view/en/62543256).
- An overview of the topic of industrial security is available here (http://support.automation.siemens.com/WW/view/en/50203404).

Simple examples for special applications

For the features described above, we provide you with documented examples with sample HTML pages and - depending on the application - with corresponding TIA project/user program; see here (https://support.industry.siemens.com/cs/ww/en/view/68011496).

Apart from basic descriptions and projects, you will also find simple examples for the Web server of SIMATIC S7-1200/S7-1500.

See also

S7-1500 web server (http://support.automation.siemens.com/WW/view/en/59193560)

Using source files for web server user-defined pages in STEP 7 (TIA Portal)

During migration of STEP 7 projects, the sources for the web server user-defined pages, such as HTML pages and graphics, are not automatically applied to the new project in the TIA Portal.

After the migration, you must integrate the sources of your user-defined pages in your project. Proceed as follows:

1. Copy the source data to a directory that can be accessed by the TIA Portal.
2. Enter the paths to the source files in the CPU properties under "Web server > User-defined pages".
3. Then compile the hardware configuration.

1.4.2 Configuring automation systems

1.4.2.1 Functional description of S7-1200 CPUs (S7-1200)

Operating modes (S7-1200)

Principles of the operating modes of S7-CPUs (S7-1200)

Introduction

Operating modes describe the behavior of the CPU. The following operating modes are possible:

- STARTUP
- RUN
- STOP

In these operating modes, the CPU can communicate via the PN/IE interface, for example.

Other operating modes

If the CPU is not ready for operation, it is in one of following two operating modes:

- Deenergized, i.e. the supply voltage is switched off.
- Defective, which means an internal error has occurred.
  If the "Defective" status is caused by a firmware error, this state is indicated by the status LEDs of the CPU (refer to the description of the CPU). To find out the cause, follow these steps:
  - Turn the power supply switch off and on again.
  - Read out the diagnostics buffer when the CPU starts up and send the data for analysis to Customer Support.

  If the CPU does not start up, replace it.

See also

STOP mode (Page 1295)

RUN mode (Page 1295)

"STARTUP" operating mode (Page 1292)

Operating mode transitions (S7-1200)

Overview

The following figure shows the operating modes and the operating mode transitions of S7-1200 CPUs:

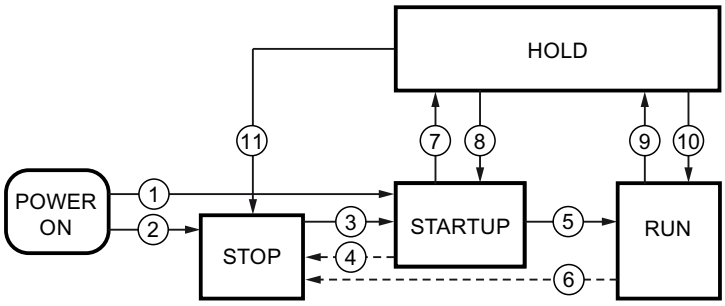

The following table shows the conditions under which the operating modes will change:

No.	Operating mode transition	Conditions
①	POWER ON → STARTUP	After switching on, the CPU goes to "STARTUP" mode if: • "Warm restart" startup type is set, and • the hardware configuration and the program blocks are consistent. Non-retentive memory is cleared and the contents of non-retentive DBs are reset to the initial values of the load memory. Retentive memory and retentive DB contents are retained.
②	POWER ON → STOP	When startup type "No startup" is set, the CPU goes to "STOP" mode after the supply voltage is switched on. Non-retentive memory is cleared and the contents of non-retentive DBs are reset to the initial values of the load memory. Retentive memory and retentive DB contents are retained.
③	STOP → STARTUP	The CPU switches to "STARTUP" mode if: • CPU is set to "RUN" from the programming device, and • the hardware configuration and the program blocks are consistent.
④	STARTUP → STOP	The CPU returns to the "STOP" mode in the following situations: • Error detected during startup. • The CPU is set to "STOP" from the programming device. • A STOP command is processed in the STARTUP OB.
⑤	STARTUP → RUN	If the STARTUP is successful, the CPU switches to "RUN".
⑥	RUN → STOP	The CPU returns to the "STOP" mode in the following situations: • An error is detected that prevents continued processing. • The CPU is set to "STOP" from the programming device. • A STOP command is processed in the user program.

"STARTUP" operating mode (S7-1200)

Principles of the STARTUP mode (S7-1200)

Function

After turning on the CPU, it executes a startup program before starting to execute the cyclic user program.

By suitably programming startup OBs, you can specify certain initialization variables for your cyclic program in the startup program. There is no rule in terms of the number of startup OBs. That is, you can set up one or several startup OBs in your program, or none at all.

Parameter settings for startup characteristics

You can specify whether the CPU remains in STOP mode or whether a warm restart is run. Over and above this, you can set the response during startup (RUN or previous mode) in the "Startup" group of the CPU properties.

Special characteristics

Note the following points regarding the "STARTUP" mode:

- The startup OBs are executed. All startup OBs you have programmed are executed, regardless of the selected startup mode.
- No time-based program execution can be performed.
- Interrupt controlled program execution limited to:
  - OB 82 (diagnostics interrupt)
- The outputs on the modules are disabled.
- The process image is not updated; direct I/O access to inputs is possible.

See also

Principles of the operating modes of S7-CPUs (Page 1290)

Startup OBs (Page 1353)

Warm restart (Page 1292)

Warm restart (S7-1200)

Function

During a warm restart, all non-retentive bit memory is deleted and non-retentive DB contents are reset to the initial values from load memory. Retentive bit memory and retentive DB contents are retained.

Program execution begins at the call of the first startup OB.

Triggering a warm restart

You can trigger a "Warm restart" using a corresponding menu command on your programming device in the following situations:

- The CPU must be in "STOP" mode.
- After a memory reset
- After downloading a consistent program and a consistent hardware configuration in the "STOP" mode of the CPU.

"POWER ON" triggers a "warm restart" if you have set the following parameters for the startup response:

- Startup type "warm restart - RUN" (regardless of the CPU operating mode prior to POWER OFF).
- "Warm restart - mode prior to POWER OFF" (depending on the CPU operating mode prior to POWER OFF. The CPU must have been in RUN mode prior to this.)

See also

Retentive memory areas (Page 1300)

Startup activities (S7-1200)

Overview

The following table shows which activities the CPU performs at STARTUP:

Activities in execution sequence	At warm restart
Clear non-retentive bit memories	Yes
Clear all bit memories	No
Clear the process image output	Yes
Processing startup OBs	Yes
Update the process image input	Yes
Enable outputs after changing to "RUN" mode	Yes

Sequence

The following figure shows the activities of the CPU in "STOP", "STARTUP", and "RUN" modes.

You can use the following measures to specify the state of the I/O outputs in the first cycle of the user program:

- Use assignable output modules to be able to output substitute values or to retain the last value.
- Set default values for outputs in startup OBs.

During the startup, all interrupt events are entered in a queue so that they can be processed later during RUN mode. In RUN mode, hardware interrupts can be processed at any time.

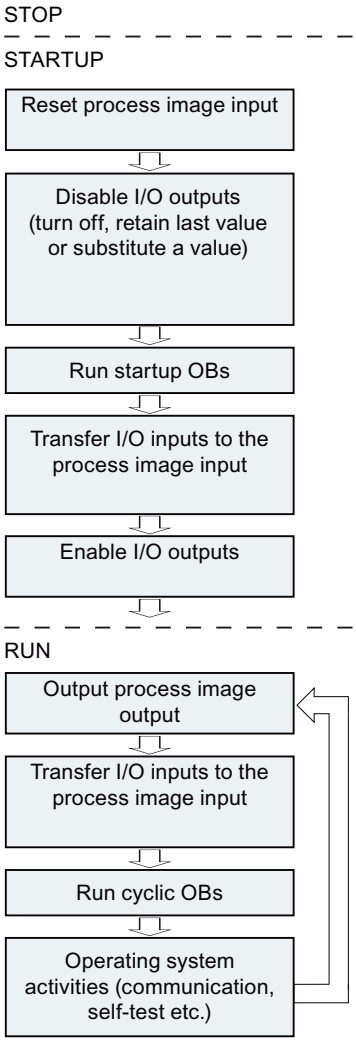

Special features during startup (S7-1200)

Response when expected and actual configurations do not match

The expected configuration is represented by the engineering configuration loaded on the CPU. The actual configuration is the actual configuration of the automation system.

If the expected configuration and actual configuration differ, the CPU nevertheless initially changes to RUN.

Canceling a STARTUP

If errors occur during startup, the startup is canceled and the CPU remains in "STOP" mode.

Under the following conditions, a startup will not be performed or will be canceled:

- If an invalid SD card is inserted.
- If no hardware configuration has been downloaded.

See also

Overview of the CPU properties (Page 1314)

RUN mode (S7-1200)

Function

In "RUN" mode the cyclic, time-driven, and interrupt-driven program sections execute:

- The process image output is read out.
- The process image input table is read.
- The user program is executed.

Active data exchange between S7-1200 CPUs by means of Open User Communication is only possible in "RUN" mode.

Running the user program

Once the CPU has read the inputs, the cyclic program runs from the first to the last instruction.

If you have configured a minimum cycle time, the CPU will not end the cycle until this minimum cycle time is up even if the user program is completed sooner.

A maximum cycle time is set which you can adjust according to your requirements. This ensures that the cyclic program is completed within a specified time. The system will respond with a time error if the cyclic program is not completed within this time.

Other events such as hardware and diagnostic interrupts can interrupt the cyclic program flow and prolong the cycle time.

See also

Principles of the operating modes of S7-CPUs (Page 1290)

Events and OBs (Page 1305)

STOP mode (S7-1200)

Function

In "STOP" mode, the user program is not executed. All outputs are disabled or react according to the parameter settings: They provide a substitute value as set in the parameters or retain the last value output and bring the controlled process to a safe status.

The CPU checks the following points:

- Hardware, for example whether are all modules are available
- Whether the default settings for the CPU are applicable or parameter sets are present
- Whether the general conditions for the programmed startup behavior are correct

See also

Principles of the operating modes of S7-CPUs (Page 1290)

Basics of a memory reset (S7-1200)

Function

A memory reset on the CPU is possible only in STOP mode.

When memory is reset, the CPU is changed to an "initial status". This means:

- An existing online connection between your programming device/PC and the CPU is terminated.
- The content of the work memory and the retentive and non-retentive data are deleted.
- The diagnostic buffer, the time, the IP address, the hardware configuration, and active force jobs are retained.
- The load memory (code and data blocks) is then copied to work memory. As a result, the data blocks no longer have current values but their configured initial values.

Memory areas (S7-1200)

What you need to know about SIMATIC memory cards (S7-1200)

Function of the SIMATIC memory card

The SIMATIC memory card for an S7-1200 is an SD memory card, preformatted by Siemens for the CPU user program, that is compatible with the Windows operating system.

NOTICE
Formatting the SIMATIC memory card You may only delete files and folders. If you format the SIMATIC memory card with Windows tools, for example with a commercially available card reader, you will render the memory card unusable as a storage medium for an S7 CPU.

Setting the card type

You can use the SIMATIC memory card as a transfer card, a program card or a firmware update card.

To set the card type, insert the SIMATIC memory card into the card reader of the programming device and select the "Card reader/USB memory" folder from the project tree. In the properties of the selected memory card, designate the card type:

- Program
  If the SIMATIC memory card is used as a program card, you can load the user program to the card. In this case, the internal load memory of the device is replaced by the SIMATIC memory card and the internal load memory is erased. The user program is then fully executable from the SIMATIC memory card. If the SIMATIC memory card with the user program is removed, there is then no program.
- Transfer
  If the SIMATIC memory card is used as a transfer card, you can transfer the user program from the card to the internal load memory of the CPU. You can then remove the SIMATIC memory card.
- Firmware card
  Firmware for the S7-1200 modules can be stored on a SIMATIC memory card. It is therefore possible to perform a firmware update with the help of a specifically prepared SIMATIC memory card. A backup copy of firmware for a module can also be stored on the SIMATIC memory card.

Transferring objects from the project to a SIMATIC memory card

When the SIMATIC memory card is inserted in the programming device or in an external card reader, you can transfer the following objects from the project tree to the SIMATIC memory card:

- Individual blocks (multiple selection possible)
  In this case a consistent transfer is available, as the dependencies of the blocks to each other is taken into account with block selection.
- PLC
  In this case, all objects relevant to processing, such as blocks and the hardware configuration, are transferred to the SIMATIC memory card, just as with downloading.

To perform the transfer, you can move the objects with drag-and-drop or use the command "Card reader/USB memory > Write to memory card" in the "Project" menu.

Transferring objects from the SIMATIC memory card to the project

You can transfer Individual blocks (multiple selection is possible) by dragging them to the project. A hardware configuration cannot be transferred from the SIMATIC memory card to the project.

Updating firmware with a SIMATIC memory card

You can get the latest firmware data on the Internet from the Service & Support pages:

http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/34143537)

Save the firmware files on the hard disk and plug the SIMATIC Memory Card into the card reader of your programming device.

To store the file on the SIMATIC memory card, select the SIMATIC memory card in the "Card Reader/USB memory" folder in the project tree. Select the shortcut menu "Card Reader/USB memory > Create firmware update memory card".

Then follow the instructions in the Service & Support portal for performing a firmware update with your CPU.

Updating the firmware changes the CPU firmware status. If you have used the CPU in the project, you will have to update the CPU already configured to the CPU with the new firmware status by changing devices offline, and adapt and then load the program or configuration.

See also

Useful information on CPU firmware versions and STEP 7 versions (Page 1375)

Load memory (S7-1200)

Function

Each CPU has an internal load memory. The size of this internal load memory depends on the CPU used.

This internal load memory can be replaced by using external memory cards. If there is no memory card inserted, the CPU uses the internal load memory; if a memory card is inserted, the CPU uses the memory card as load memory.

Copying from the internal to the external load memory

If you do not select the setting "Disable copying from internal load memory to external load memory" in the protection properties of the device configuration of the CPU, the CPU determines which type of memory card you are inserting:

- Empty memory card: An empty memory card has no job file (S7_JOB.S7S). If you insert an empty memory card, the CPU adds a program job file. It then copies the internal load memory to the external load memory (the program file on the memory card) and deletes the internal load memory.
- Empty program card: An empty program card contains a program job file, which is empty. In this case, the CPU copies the internal load memory to the external load memory (the program file on the memory card) and deletes the internal load memory.

If you have selected the setting "Disable copying from internal load memory to external load memory" in the protection properties of the device configuration of the CPU, the CPU reacts as follows:

- Empty memory card: An empty memory card has no job file (S7_JOB.S7S). If you insert an empty memory card, the CPU does not execute anything. It creates no program job file and it does not copy the internal load memory to the external load memory (the program file on the memory card). It does not delete the internal load memory.
- Empty program card: An empty program card contains a program job file, which is empty. In this case the CPU executes no action. It does not copy the internal load memory to the external load memory (the program file on the memory card). It does not delete the internal load memory.

If you insert a program card, transfer card or a card with a firmware update into the CPU, the configuration setting "Disable copying from internal load memory to external load memory" does not affect how the CPU evaluates the memory card.

Work memory (S7-1200)

Function

Work memory is a non-retentive memory area for storing elements of the user program that are relevant for program execution. The user program is executed exclusively in work memory and system memory.

System memory (S7-1200)

System memory areas (S7-1200)

Function

System memory contains the memory elements that each CPU makes available to the user program, such as the process image and bit memory.

By using appropriate operations in your user program, you address the data directly in the relevant operand area.

The following table shows the operand areas of the system memory:

Operand area	Description	Access via units of the following size:	S7 notation
Process image output	The CPU writes the values from the process image output table to the output modules at the start of the cycle.	Output (bit)	Q
		Output byte	QB
		Output word	QW
		Output double word	QD
Process image input	The CPU reads the inputs from the input modules and saves the values to the process image input table at the start of the cycle.	Input (bit)	I
		Input byte	IB
		Input word	IW
		Input double word	ID

Operand area	Description	Access via units of the following size:	S7 notation
Bit memory	This area provides storage for intermediate results calculated in the program.	Bit memory (bit)	M
		Memory byte	MB
		Memory word	MW
		Memory double word	MD
Data block	Data blocks store information for the program. They can either be defined so that all code blocks can access them (global DBs) or assigned to a specific FB or SFB (instance DB). Requirement: The block attribute "Optimized block access" is not enabled.	Data bit	DBX
		Data byte	DBB
		Data word	DBW
		Data double word	DBD
Local data	This area contains the temporary data of a block while the block is being processed. Requirement: The block attribute "Optimized block access" is not enabled. Recommendation: Access local data (temp) symbolically.	Local data bit	L
		Local data byte	LB
		Local data word	LW
		Local data double word	LD
I/O input area	The I/O input and output areas permit direct access to central and distributed input and output modules.	I/O input bit	<tag>:P
		I/O input byte	
		I/O input word	
		I/O input double word	
I/O output area		I/O output bit	
		I/O output byte	
		I/O output word	
		I/O output double word	

See also

Diagnostics buffer (Page 1303)

Basic principles of process images (Page 1301)

Access to the I/O addresses (Page 1304)

Retentive memory areas (S7-1200)

Retentive memory areas

Data loss after power failure can be avoided by marking certain data as retentive. This data is stored in a retentive memory area. A retentive memory area is an area that retains its content following a warm restart, in other words, after cycling the power when the CPU changes from STOP to RUN.

The following data can be assigned retentivity:

- Bit memory: The precise width of the memory can be defined for bit memory in the PLC tag table or in the assignment list.
- Tags of a function block (FB): You can define individual tags as retentive in the interface of an FB if you have enabled optimized block access. Retentivity settings can be defined only in the assigned instance data block if optimized block access has not been activated for the FB.
- Tags of a global data block: You can define retentivity either for individual or for all tags of a global data block depending on the settings for access.
  - Block with optimized access: retentivity can be set for each individual tag.
  - Block with standard access: The retentivity setting applies to all tags of the DB; either all tags are retentive or no tag is retentive.

See also

Warm restart (Page 1292)

process image input/output (S7-1200)

Basic principles of process images (S7-1200)

Function

When the user program addresses the input (I) and output (O) operand areas, it does not query or change the signal states on the digital signal modules. Instead, it accesses a memory area in the system memory of the CPU. This memory area is referred to as the process image.

Advantages of the process image

Compared with direct access to input and output modules, the main advantage of accessing the process image is that the CPU has a consistent image of the process signals for the duration of one program cycle. If a signal state on an input module changes during program execution, the signal state in the process image is retained until the process image is updated again in the next cycle. The process of repeatedly scanning an input signal within a user program ensures that consistent input information is always available.

Access to the process image also requires far less time than direct access to the signal modules since the process image is located in the internal memory of the CPU.

Updating the process images (S7-1200)

Sequence

The operating system updates the process images at cyclic intervals unless defined otherwise in your configuration. The process image input/output is updated in the following order:

1. The internal tasks of the operating system are performed.
2. The process image output (PIQ) table is written to the outputs of the module.
3. The status of inputs is read to the process image input (PII) table.
4. The user program is executed with all the blocks that are called in it.

The operating system automatically controls the writing of the process image output to the outputs of the modules and the reading of the process image input.

Special characteristics

You have the option of accessing inputs or outputs directly using direct I/O access.

- If an instruction accesses an output directly and the output address is located in the process image output, the process image of the relevant output is updated.
- If an instruction accesses an output directly and the output address is **not** located in the process image output, the process image of the relevant output is **not** updated.

Example of normal I/O access by way of the process image

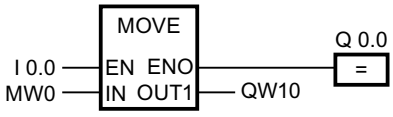

Update QW10 in the I/O output area with the value from MW0.

I/O access error during process image updating

If an error occurs during process image updating (I/O access error), the CPU reacts with the default system reaction "Ignore".

See also

Start address of a module (Page 1303)

Access to the I/O addresses (Page 1304)

Startup activities (Page 1293)

Diagnostics buffer (S7-1200)

Function

The diagnostics buffer is part of the system memory of the CPU. It contains the errors detected by the CPU or modules with diagnostics capability. It includes the following events:

- Every mode change of the CPU (for example, POWER UP, change to STOP mode, change to RUN mode)
- Every diagnostics interrupt

The diagnostics buffer of the S7-1200-CPU has a capacity of 50 entries of which the last (most recent) 10 entries are retained following power cycling.

Those entries can only be cleared by restoring the CPU to factory defaults.

You can read the content of the diagnostics buffer with the help of the Online and Diagnostics view.

See also

Basic information on the diagnostics buffer (Page 3351)

I/O data area (S7-1200)

Start address of a module (S7-1200)

Definition

The start address is the lowest byte address of a module. It acts as the initial address of the module user data area.

Configuring module start addresses

The addresses used in the user program and the module start addresses are coordinated when the modules are configured.

In the module properties ("I/O addresses" group), you can change the start addresses that were assigned automatically after the modules were inserted.

You can also make a setting that decides whether or not the addresses are located in the process image.

Access to the I/O addresses (S7-1200)

I/O addresses

If you insert a module in the device view, its user data is located in the process image of the S7-1200 CPU (default). The CPU handles the data exchange between the module and the process image area automatically during the update of the process images.

Append the suffix ":P" to the I/O address if you want the program to access the module directly instead of using the process image.

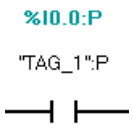

This could be necessary, for example, during execution of a time-sensitive program which also has to control the outputs within the same cycle.

Basics of program execution (S7-1200)

English designation of the OB types (S7-1200)

English designation of the OB types

When an OB is created in the "Add new block" dialog, the OB types can only be displayed in English for technical reasons – irrespective of the set user interface language. In order to facilitate handling, the following table lists the English terms and their corresponding designations in the set user interface language.

English designation of the OB type in the "Add new block" dialog.	Designation in the information system
Program cycle	Cycle OB, OB for cyclic program processing, cyclic program
Startup	Startup OB
Time delay interrupt	Time-delay interrupt OB
Cyclic interrupt	Cyclic interrupt OB
Hardware interrupt	Hardware interrupt OB
Time error interrupt	Time error OB
Diagnostic error interrupt	Diagnostic interrupt OB
Pull or plug of modules	Pull/plug interrupt OB
Rack or station failure	Rack failure OB
Time of day	Time-of-day interrupt OB
Status	Status interrupt OB
Update	Update interrupt OB
Profile	OB for vendor or profile-specific interrupt

English designation of the OB type in the "Add new block" dialog.	Designation in the information system
MC-Interpolator	MC-Interpolator OB
MC-Servo	MC-Servo OB
MC-PreServo	MC-PreServo OB
MC-PostServo	MC-PostServo OB

Events and OBs (S7-1200)

Events and OBs

The operating system of S7-1200-CPUs is based on events. There are two types of events:

- Events which can start an OB
- Events which cannot start an OB

An event which can start an OB triggers the following reaction:

- It calls the OB you possibly assigned to this event. The event is entered in a queue according to its priority if it is currently not possible to call this OB.
- The default system reaction is triggered if you did not assign an OB to this event.

An event which cannot start an OB triggers the default system reaction for the associated event class.

The user program cycle is therefore based on events, the assignment of OBs to those events, and on the code which is either contained in the OB, or called in the OB.

The following table provides an overview of the events which can start an OB, including the associated event classes and OBs. The table is sorted based on the default OB priority. Priority class 1 is the lowest.

Event class	OB no.	Number of OBs	Start event	OB priority (default)
Cyclic program	1, >= 123	>= 1	Starting or end of the last program cycle OB	1
Startup	100, >= 123	>=0	STOP to RUN transition	1
Time-of-day interrupt	>= 10	Max. 2	Start time has been reached	2

Event class	OB no.	Number of OBs	Start event	OB priority (default)
Time-delay interrupt	>= 20	Max. 4	Delay time expired	• 3 • Firmware version V4.0 or higher: 3 for OB20, 4 for OB21, 5 for OB22, 6 for OB23, 3 for OB123 to 32767
Cyclic interrupt	>= 30		Constant bus cycle time expired	• 8 • Firmware version V4.0 or higher: 8 for OB30, 9 for OB31, 10 for OB32, 11 for OB33, 12 for OB34, 13 for OB35, 14 for OB36, 16 for OB37, 17 for OB38, 7 for OB123 to 32767
Hardware interrupt	>= 40	Max. 50 (more can be used with DETACH and AT-TACH)	• Positive edge (max. 16) • Negative edge (max. 16)	18
			• HSC: Count value = reference value (max. 6) • HSC: Count direction changed (max. 6) • HSC: External reset (max. 6)	18
Status interrupt	55	0 or 1	CPU has received status interrupt	4
Update interrupt	56	0 or 1	CPU has received update interrupt	4
Manufacturer- or profile-specific interrupt	57	0 or 1	CPU has received manufacturer-specific or profile-specific interrupt	4
Diagnostic error interrupt	82	0 or 1	Module has detected an error	5
Pull/plug interrupt	83	0 or 1	Removal/insertion of modules of distrib-uted I/O	6
Rack error	86	0 or 1	Error in the I/O system of the distributed I/O	6
Time error	80	0 or 1	• Maximum cycle time exceeded • Called OB is still being executed • Time-of-day interrupt missed • Time-of-day interrupt missed during STOP • Queue overflow • Interrupt loss due to high interrupt load	22

The following table describes events which do not trigger an OB start, including the corresponding reaction of the operating system. The table is sorted based on event priority.

Event class	Event	Event priority	System reaction
Insert/remove central modules	Insert/remove a module	21	STOP
I/O access error during process image update	I/O access error during process image update	22	Ignore
Programming error	Programming error in a block for which you use system reactions provided by the operating system (note: the error handling routine in the block program is executed if you activated local error handling).	23	RUN
I/O access error	I/O access error in a block for which you use system reactions provided by the operating system (note: the error handling routine in the block program is executed if you activated local error handling).	24	RUN
Maximum cycle time exceeded twice	Maximum cycle time exceeded twice	27	STOP

Assignment between OBs and events

With the exception of the cyclic program and startup program and event can only be assigned to one OB. However, in certain event classes such as hardware interrupts one and the same OB can be assigned to several events.

The assignment between OBs and events is defined in the hardware configuration. Defined assignments can be changed at runtime by means of ATTACH and DETACH instructions.

OB priority and runtime behavior

S7-1200-CPUs support the priority classes 1 (lowest) to 27 (highest). An OB is assigned the priority of its start event.

OBs are always executed on a priority basis: The OBs with the highest priority are executed first. Events of the same priority are processed in order of occurrence.

As of firmware version V4.0 of the S7-1200 CPUs you can specify in the device configuration, under properties of the CPU, if the OBs are interruptible or not. This parameter assignment has an effect on all OBs with exception of the cycle OBs which are always interruptible.

The following applies to S7-1200 CPUs with firmware version < V4.0:

- Any OB with priority >= 2 will interrupt cyclic program execution.
- An OB of priority 2 to 25 cannot be interrupted by any event of priority group 2 to 25. This rule also applies to events of a priority higher than that of the currently active OB. Such events are processed later.
- A time error (priority 26) will interrupt any other OB.

The following applies to S7-1200 CPUs as of firmware version V4.0:

If you do not configure the OBs as interruptible, an OB is always processed completely even if an event of a higher priority occurs during its runtime. Specifically, this means:

- Any OB with priority >= 2 will interrupt cyclic program execution.
- An OB of priority 2 to 25 cannot be interrupted by any event. This rule also applies to events of a priority higher than that of the currently active OB, which also includes a time error. Such events are processed later.

If you do configure the OBs as interruptible and an event of a higher priority occurs during the runtime of an OB, the running OB is interrupted and the OB associated with the occurring event is processed. Once this OB has been completed, processing of the interrupted OB continues. Specifically, this means:

- Any OB with priority >= 2 will interrupt cyclic program execution.
- An OB of priority 2 to 25 can be interrupted by any event whose priority is higher than that of the running OB. This is also true for time errors: A time error (priority 26) will interrupt any OB.

OB start information

Certain OBs have start information, while others do not. This is explained in greater detail in the description of the relevant OB.

See also

Event-based program execution (Page 1308)

Event-based program execution (S7-1200)

OB priority and runtime behavior

S7-1200-CPUs support the priority classes 1 (lowest) to 27 (highest). An OB is assigned the priority of its start event.

Interrupt OBs can only be interrupted by time error interrupts. This rule also applies to events of a priority higher than that of the currently active OB. That is, only one interrupt OB can be active, with exception of the time error interrupt OB.

Any further event of generated while an interrupt OB is being executed is added to a queue in accordance with its priority. Start events within a queue are processed later based on the chronological order of their occurrence.

Program execution on the CPU

Cyclic OBs are interrupted by interrupt OBs.

Reactions to events which start an interrupt OB:

- For CPUs up to firmware version V3: Interrupt OBs can only be interrupted by time error interrupt OBs.
- For CPUs as of firmware version V4: Interrupt OBs can be interrupted by interrupt OBs of a higher priority.

The figure below shows the basic procedure in case interrupt OBs cannot be interrupted (behavior up to firmware version V3):

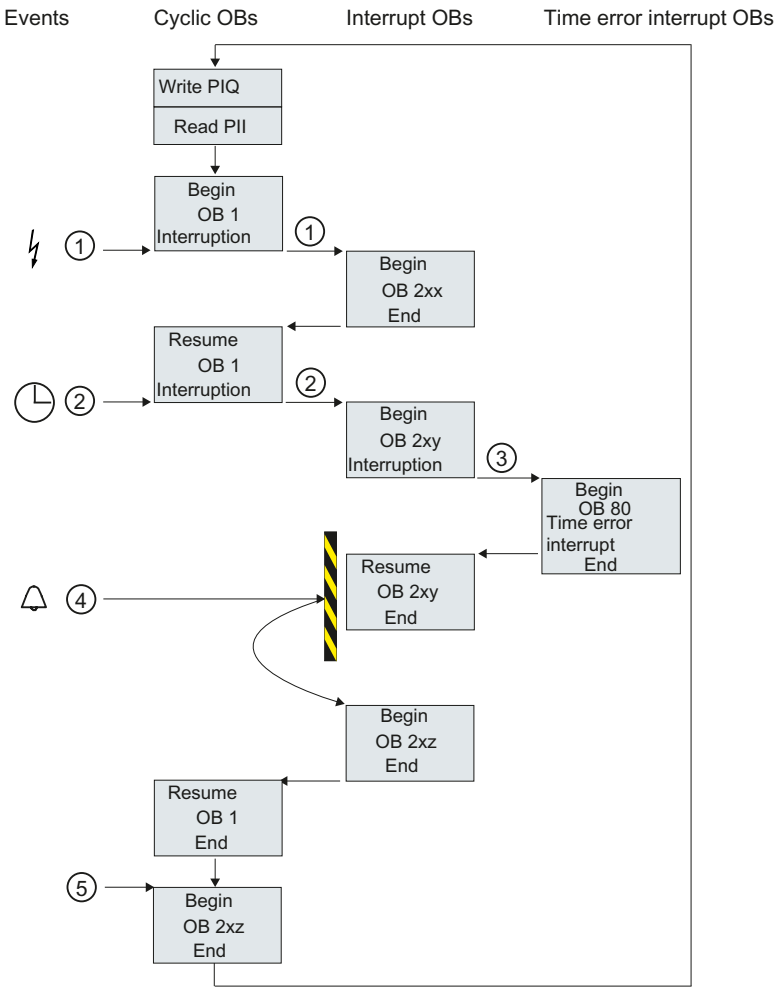

① and ② An event (e.g. a hardware interrupt) calls its associated OB.

A called OB is executed without interruption, including all of its nested blocks. Execution of the cyclic OB is resumed on completion of interrupt processing, provided the queue does not contain any events which trigger an OB start.

③ An interrupt OB can only be interrupted by a time error interrupt OB (OB 80).

④ An new alarm-triggering event occurs during interrupt processing.

Reaction for CPUs up to and including firmware version V3:

This new event is added to a queue. The queued events successively call their corresponding OBs only after execution of the current interrupt OBs was completed and according to the following rules:

- Events are processed in the order of their priority (starting at the highest priority).
- Events of the same priority are processed in chronological order.

Reaction for CPUs as of firmware version V4:

You use a CPU parameter to set the interruptibility for CPUs as of firmware version V4. Default behavior: OBs are interruptible. In this case: If the new event has a higher priority than the currently running OB, the OB started by the new event interrupts the currently running OB.

If you disable the option, the interrupt OBs cannot be interrupted.

⑤ The cyclic OBs are processed one after the other.

Notes on queues

- Every priority class (OBs of the same priority to be called) is assigned a separate queue. The size of those queues is set by default.
- Any new event leading to the overflow of a queue is discarded and therefore lost. A "time error interrupt event" is generated simultaneously. Information identifying the OB that caused the error is included in the start information of the time error interrupt OB (OB 80). A corresponding reaction such as an alarm trigger can be programmed in the time error interrupt OB.

Example of a hardware interrupt event (S7-1200)

The function principle of event-oriented program execution in the S7-1200 CPU is described based on the example of a hardware interrupt-triggering module.

Process events and their priority

Process events are triggered by the I/O (e.g. at a digital input) and initiate a call of the assigned OB in the S7-1200 CPU. OBs assigned to a process event are called hardware interrupt OBs.

Examples of process events and their priority:

- Process events "rising edge" or "falling edge" at an interrupt-triggering module: The hardware interrupt OB started by such an event is always assigned priority 5.
- Process events from a high-speed counter
  - Count value corresponds to the reference value
  - Change count direction
  - External reset of the high-speed counter
    The hardware interrupt OB started by this event is always assigned priority 6.

The figure below shows the chronological sequence of hardware interrupt execution: In the case of two hardware interrupt events in immediate succession, the second hardware interrupt triggering event is held back in the queue until the first OBx has been processed. The next hardware interrupt triggering event can only start the associated OBx when the OBx has been processed. Additional hardware interrupt triggering events are lined up in the queue according to this principle.

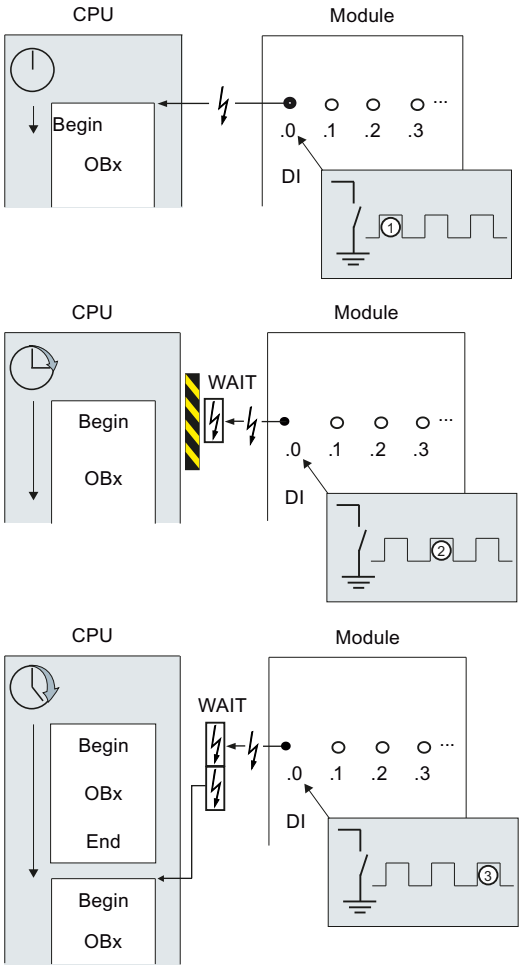

Hardware interrupt execution

① A hardware interrupt-triggering event such as a rising edge at the input calls the OB to which it is assigned.

② If a new event occurs that triggers a hardware interrupt while the OB is executing, this event is entered in a queue.

③ The new event that triggers a hardware interrupt starts the hardware interrupt OB assigned to the event.

Assigning the interrupt-triggering event

The interrupt-triggering event is assigned to an OB in the input properties of the device view.

- An interrupt-triggering event can only be assigned to a single OB.
- OBs, however, can be assigned to several interrupt-triggering events.
  This means, for example, that you can assign both the rising and the falling edge to the same interrupt OB in order to trigger the same reaction to any change of the input signal.

- The started OB can interrupt a cycle OB at every instruction. Consistent data access is secured up to dword size.
- You can parameterize module-specific interrupt-triggering events such as a rising and the falling edge at the input.
- Assign the interrupt-triggering event and the OB to be started in the configuration of the interrupt-triggering module. However, within the started hardware interrupt OB you can override this assignment using the DETACH instruction, or assign the same event to a different OB using the ATTACH instruction. This functionality allows a flexible reaction to external process signals.

Setting the operating behavior (S7-1200)

Changing properties of the modules (S7-1200)

Default settings

When they leave the factory, all hardware components with parameters have default settings suitable for standard applications. These default values allow the hardware components to be used immediately without making any additional settings.

You can, however, modify the behavior and the properties of the hardware components to suit the requirements and circumstances of your application. Hardware components with settable parameters include, for example, communications modules and several analog and digital modules.

Setting and loading parameters

When you have selected a hardware component in the device or network view, you can set the properties in the Inspector window. When you save a device configuration with its parameters, data is generated that needs to be loaded on the CPU. This data is transferred to the relevant modules during startup.

Properties of the CPUs

The properties of the CPUs have special significance for system behavior. For example for a CPU you can set:

- Interfaces
- Inputs and outputs
- High-speed counters
- Pulse generators
- Startup behavior
- Time-of-day
- Protection level
- Bit memory for system and clock

- Cycle time
- Communications load

The entry possibilities specify what is adjustable and in which value ranges. Fields that cannot be edited are disabled or are not shown in the properties window.

Requirement

You have already arranged the hardware components for which you want to change properties on a rack.

Procedure

To change the properties and parameters of the hardware components, follow these steps:

1. In the device or network view, select the hardware component or interface that you want to edit.
2. Edit the settings for the selected object:
   – For example in the device view you can edit addresses and names.
   – In the Inspector window additional setting possibilities are available.

You do not need to confirm your entries, the changed values will be applied immediately.

CPU properties (S7-1200)

Overview of the CPU properties (S7-1200)

Overview

The following table provides you with an overview of the CPU properties:

Group	Properties	Description
General	Project information	General information to describe the inserted CPU. Except for the slot number, you can change this information.
	Catalog information	Read-only information from the hardware catalog for this CPU.
	Identification & Maintenance	For saving application-specific information such as the name of the plant and the installation location.
	Checksums	To check the identity or integrity of PLC programs. Blocks in the block folder and text lists are automatically marked with unique checksums when they are compiled. You can easily establish whether the program currently running on the CPU is the same program that you loaded a long time ago or whether the program has been changed in the meantime. The "GetChecksum" instruction is available to read out the checksum while the program is running. See: Comparison of PLC programs based on checksums

Group	Properties	Description
PROFINET interface	General	Name and comment for this PROFINET interface. The name is limited to 110 characters.
	Ethernet addresses	Select whether the PROFINET interface is networked. If subnets have already been created in the project, they are available for selection in the drop-down list. If not, you can create a new subnet with the "Add new subnet" button. Information on the IP address, subnet mask and IP router usage in the subnet is available in the IP protocol. If an IP router is used, the information about the IP address of the IP router is necessary. See: Assigning addresses and names for PROFINET devices
	Advanced options	Name, comment and additional setting options of the Ethernet interface port.
	Time synchronization	Settings for time synchronization in the NTP time format. The NTP (network time protocol) is a general mechanism for synchronizing system clocks in local and global area networks. In NTP mode, the interface of the CPU sends time queries (in client mode) at regular intervals to NTP servers on the subnet (LAN) and the addresses must be set in the parameters here. Based on the replies from the server, the most reliable and most accurate time is calculated and synchronized. The advantage of this mode is that it allows the time to be synchronized across subnets. The accuracy depends on the quality of the NTP server being used.
DI#/DO#	General	Name of and comment on the integrated digital inputs of the CPU.
	Digital inputs	Input delays can be set for digital inputs. The input delays can be set in groups (in each case for 4 inputs). The detection of a positive and a negative edge can be enabled for each digital input. A name and a hardware interrupt can be assigned to this event. Depending on the CPU, pulse catches can be activated at various inputs. When the pulse catch is activated, even pulse edges that are shorter than the cycle time of the program are detected.
	Digital outputs	The reaction to a mode change from RUN to STOP can be set for all digital outputs: The state can either be frozen (corresponds to retain last value) or you set a substitute value ("0" or "1")
	I/O addresses	The address space of the input and output addresses is specified as is the process image.

Group	Properties	Description
AI#	General	Name of and comment on the integrated analog inputs of the CPU.
	Analog inputs	During noise reduction, the specified integration time suppresses interference frequencies at the specified frequency (in Hz). The channel address, measurement type, voltage range, smoothing and overflow diagnostics must be specified in the "Channel #" group. The measurement type and voltage range are set permanently to voltage, 0 to 10 V. Smoothing analog values provides a stable analog signal for further processing. Smoothing analog values can be useful with slow measured value changes, for example, in temperature measurement. The measured values are smoothed with digital filtering. Smoothing is achieved by the module forming mean values from a specified number of converted (digitalized) analog values. The selected level (slight, medium, strong) decides the number of analog signals used to create the mean value. If overflow diagnostics is enabled, a diagnostics event is generated if an overflow occurs.
	I/O addresses	The address space of the input addresses is specified as is the process image.
High-speed counter (HSC)	High-speed counter (HSC)#	High-speed counters are typically used to drive counting mechanisms. See: Configuring high-speed counters (Page 1327)
Pulse generators (PTO/PWM)	PTO#/PWM#	A pulse generator is activated and can be initialized with project information. For the configuration of an activated pulse generator, specify the usage as PWM (Pulse Width Modulation) or as PTO (Pulse Train Output). Specify the output source, time base, pulse width format, cycle time and initial pulse width for PWM. A pulse output is specified as the hardware output. The PWM output is controlled by the CTRL_PWM instruction, see CTRL_PWM. Specify the output source for PTO. A pulse output and a direction output are specified as the hardware outputs. A PTO is operated together with a high-speed counter in the "axis of motion" count mode and controlled by the Motion Control technology object (see keyword "Motion Control S7-1200") . The hardware ID is displayed in the I/O-diagnostics addresses and, if the PWM function is selected, the address space of the output addresses and the process image can be selected.

Group	Properties	Description
Startup	Startup after POWER ON	Setting the startup characteristics after cycling power. See: Principles of the STARTUP mode (Page 1292)
	Comparison of preset configuration and actual configuration	Specifies the startup characteristics of the CPU for situations in which the actual configuration of the S7-1200 station does not correspond to the preset configuration: • Startup of the CPU only if compatible • Startup of the CPU even if there are differences With the "Startup CPU only if compatible" setting, a module in a configured slot must be compatible with the configured module. Compatible means that the module that is present matches the number of inputs and outputs and must match with respect to its electrical and functional properties. A compatible module must be able to completely replace a configured module; it may be more capable, but not less capable. Example for the "Startup CPU only if compatible" setting: A CPU can be a compatible replacement for a CPU of the same type with a lower firmware version. An input module with 32 digital inputs can be a compatible replacement for a signal module with 16 digital inputs. The CPU starts up when the configured module or a compatible module is plugged in. The CPU does not start up if an incompatible module is inserted. Example of the "Startup CPU even if mismatch" setting: Instead of a configured digital input module, an analog output module is plugged in or no module is present in this slot. Although the configured inputs cannot be accessed, the CPU starts up. Note in this case that the user program cannot function correctly and take the appropriate measures.
	Configuration time for central and distributed I/O	Specifies a maximum period (standard: 60000 ms) in which the central I/O and distributed I/O must start up. (The CMs and CPs are supplied with voltage and communication parameters during the CPU startup. This configuration time provides a time period during which I/O modules connected to the CM or CP must start up.) The CPU switches to RUN as soon as the central I/O and the distributed I/O has started and is ready for operation, regardless of the "Parameter assignment time for central and distributed I/O" parameter. If the central I/O and distributed I/O has not started up during this period, the CPU switches to RUN without the central I/O and distributed I/O.
Cycle	Maximum cycle time and minimum cycle time.	Specification of a maximum cycle time or a fixed minimum cycle time. If the cycle time exceeds the maximum cycle time, the CPU goes to STOP mode. See: Cycle time and maximum cycle time (Page 1320)
Communication load	Maximum allocation of the cycle for communication (as a percentage)	Controls the duration of communication processes that always also extend the cycle time, within certain limits. Examples of communication processes include: Transferring data to another CPU or loading blocks (initiated via the PC). See: Cycle loading by communications (Page 1321)

Group	Properties	Description
System and clock memory	System memory bits and clock memory bits	You use system memory bits for the following scans: • Is the current cycle the first since cycling power? • Have there been any diagnostics state changes compared with the previous cycle? • Scan for "1" (high) • Scan for "0" (low) Clock memory bits change their values at specified periodic intervals. See: Enabling system memory (Page 1338) See: Using clock memory (Page 1338)
Web server	-	Enables and configures the Web server function. See: Enabling the Web server (Page 1251)
	Automatic update	Sends the requested web page with current CPU data periodically to the web browser. Enter the period duration under "Update interval". Automatic update can only be activated if the web server is enabled. See: Activate automatic update (Page 1252)
	User Management	You enter the user names with their access rights here. • Everyone: pre-defined user whose access rights you assign using a drop-down list. The "Everyone" user is a user who does not log on with a password for Web access. • User#: Editable name for a user for whom you assign access rights and a password. As soon as you click in a line below "Everyone", a preset name is displayed that you can change. Each user can be assigned different access rights via a drop-down list. There are access rights that are connected to each other. Example: When you activate the access right "Perform firmware update", the access rights "Change operating mode" and "... query diagnostics" are activated automatically. These two rights are required to perform a firmware update from the Web server. See: User list (Page 1253)
	Watch tables	Allows you to create watch tables and defines access to those tables.
	User-defined web pages	Allows access to freely-designed web pages of the CPU via a web browser. See: What you need to know about user pages (Page 1257)
	Entry page	Allows you to select the entry page.
	Overview of interfaces	Table of all modules with their Ethernet interfaces providing Web server functionality for this device. Here, you can permit or deny access to the Web server over the interface for each Ethernet interface of the device (CPU, CP or CM).

Group	Properties	Description
User interface languages	-	If you want to use translations of project texts for the Web server, you need to assign the project language to the languages for the Web server (number depends on the CPU). This specifies the project languages that are displayed when the Web server is accessed. Example: You assign the project language "German (Luxembourg)" to the Web server language "German" by selecting it from the drop-down list. Enable the project language under "Languages & resources > Project languages" in the project tree to make it available for selection in the drop-down list.
Time	Local time and daylight saving time	Setting of the time zone in which the CPU is operated and setting of the daylight-saving/standard time changeover.
Protection & Security	Protection and password for read/write access	Setting the read/write protection and the password for access to the CPU. See: Setting options for the protection level (FW V1 to V3) (Page 1339) See: Setting options for the protection (FW as of V4) (Page 1340)
	Connection mechanisms	Enabling access over PUT/GET communication.
	Security event	Enabling group alarms for security events and specifying a monitoring time (interval).
	External load memory	Disables copying from the internal to the external load memory (Page 1298).
Configuration control	-	Enables configuration control. Allows a configuration change in the user program within certain limits. See: Configuration control
Connection resources	-	Provides an overview of the assigned reserved and dynamic resources for the CPU connections. The resources currently used are also displayed in the online view. See: Use of connection resources
Address overview	-	Tabular representation of all addresses used by the CPU for integrated inputs/outputs as well as for the inserted modules. Addresses that are not used by any module are represented as gaps. The view can be filtered according to • Input addresses • Output addresses • Address gaps

See also

Assigning parameters to hardware interrupt OBs (Page 1373)

Access to the I/O addresses (Page 1304)

Special features during startup (Page 1294)

Cycle time and maximum cycle time (S7-1200)

Function

The cycle time is the time that the operating system requires to execute the cyclic program and all the program sections that interrupt this cycle. The program execution can be interrupted by:

- Time errors and 2xMaxCycleTime errors
- System activities, e.g., process image updating

The cycle time (Tcyc) is therefore not the same for every cycle.

The following schematic shows an example of different cycle times (TZ1 ≠ TZ2) for S7-1200 CPUs:

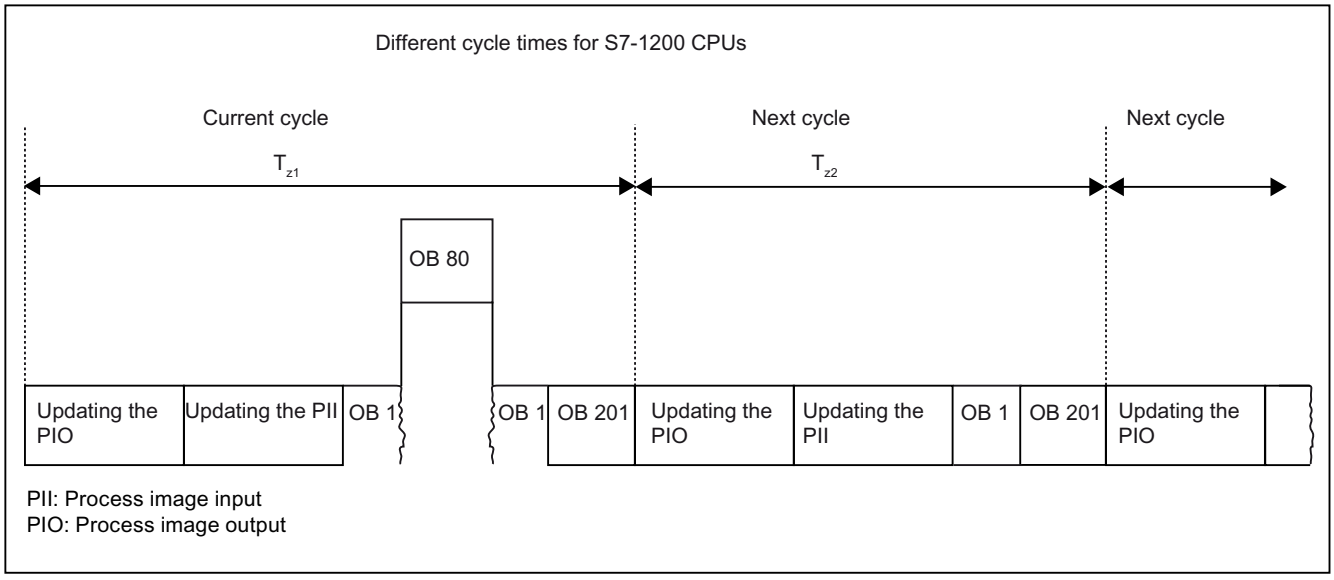

In the current cycle, the cyclic OB used here (e.g. OB 1) will be interrupted by a time error (e.g. OB 80). Following the cyclic OB, the next cycle OB 201 is processed.

Maximum cycle time

The operating system monitors the execution time of the cyclic program for a configurable upper limit known as the maximum cycle time. You can restart this time monitoring at any point in your program by calling the RE_TRIGR instruction.

If the cyclic program exceeds the maximum cycle time, the operating system will attempt to start the time error OB (OB 80). If the OB is not available, the S7-1200 CPU reacts as follows:

- CPUs with firmware version V1.x to V3.x: The CPU remains in RUN mode.
- CPUs with firmware version as of V4.x: The CPU changes to STOP mode

In addition to monitoring the runtime for overshooting of the maximum cycle time, adherence to a minimum cycle time is guaranteed. To do this, the operating system delays the start of

the new cycle until the minimum cycle time has been reached. During this waiting time, new events and operating system services are processed.

If the maximum cycle time is exceeded a second time, for example while the time error OB is being processed (2xMaxCycleTime error), the CPU changes to STOP mode.

See also

Time error OB (Page 1364)

Cycle loading by communications (S7-1200)

Function

The cycle time of the CPU can be extended due to communications processes. These communications processes include for example:

- Transferring data to another CPU
- Loading of blocks initiated by a programming device

You can control the duration of these communications processes to some extent using the CPU parameter "Cycle load due to communication".

In addition to communications processes, test functions also extend the cycle time. The "Cycle load due to communication" parameter can be used to influence the duration.

How the parameter works

You use the "Cycle load due to communication" parameter to enter the percentage of the overall CPU processing capacity that can be available for communications processes. This CPU processing capacity is now available at all times for communication. This processing capacity can be used for program execution when not required for communication.

Effect on the actual cycle time

The "Cycle load due to communication" parameter can be used to extend the cycle time of the cyclic organization block (e.g., OB 1) by a factor calculated according to the following formula:

$$\frac{100}{100 - \text{"Cycle load due to communication"}}$$

The formula does not take into account the effect of asynchronous events such as hardware interrupts or cyclic interrupts on the cycle time.

If the cycle time is extended due to communication processes, more asynchronous events may occur within the cycle time of the cyclic organization block. This extends the cycle still further. The extension depends on how many events occur and how long it takes to process them.

Example 1 – no additional asynchronous events:

If the "Cycle load due to communication" parameter is set to 50%, this can cause the cycle time of the cyclic organization block to increase by up to a factor of 2.

Example 2 – additional asynchronous events:

For a pure cycle time of 500 ms, a communication load of 50% can result in an actual cycle time of up to 1000 ms, provided that the CPU always has enough communications jobs to process. If, parallel to this, a cyclic interrupt with 20 ms processing time is executed every 100 ms, this cyclic interrupt would extend the cycle by a total of 5*20 ms = 100 ms without communication load. That is, the actual cycle time would be 600 ms. Because a cyclic interrupt also interrupts communications, it affects the cycle time by adding 10 * 20 ms at 50 % communication load. That is, in this case, the actual cycle time amounts to 1200 ms instead of 1000 ms.

Note

Observe the following:

- Check the effects of changing the value of the "Cycle load due to communication" parameter while the system is running.
- You must always consider the communication load when setting the minimum cycle time as time errors will otherwise occur.

Recommendations

- Increase this value only if the CPU is used primarily for communication purposes and the user program is not time critical.
- In all other situations you should only reduce this value.

Time-of-day functions (S7-1200)

Basic principles of time of day functions (S7-1200)

All S7-1200 CPUs are equipped with an internal clock. The backup supports the display of the correct time for up to 10 hours if the power supply is interrupted.

Time-of-day format

The clock always shows the time of day with a resolution of 1 millisecond and the date including the day of the week. The time adjustment for daylight-saving time is also taken into account.

Setting and reading the time of day (S7-1200)

Setting and reading the time with instructions

You can set, start and read the time-of-day and date on the CPU clock with the following instructions in the user program:

- Set the time-of-day: "WR_SYS_T"
- Read time of day "RD_SYS_T"
- Read local time "RD_LOC_T"
- Set time zone "SET_TIMEZONE"

Manual setting

You can also read and set the time-of-day manually in the online and diagnostics view under "Functions > Set time-of-day".

Assigning the clock parameters (S7-1200)

Clock parameters

The clock parameters allow you to make the following settings:

- Enable time synchronization via NTP server
  Select this check box if you want the internal clock to be synchronized using the NTP synchronization mode.
- Network time server
  The IP addresses of up to four NTP servers need to be configured.
- Update interval
  The update interval defines the interval between time queries.

See also

NTP method - configuration (Page 3054)

High-speed counters (S7-1200)

General information on high-speed counters (S7-1200)

Introduction

High-speed counters are typically used to drive counting mechanisms in which a shaft turning at a constant speed is equipped with an incremental step encoder. The incremental step encoder ensures a certain number of count values per rotation and a reset pulse once per rotation. The clock memory bit(s) and the reset pulse of the incremental step encoder supply the inputs for the high-speed counter.

The various S7-1200 CPUs have differing numbers of high-speed counters available:

S7-1200 CPU	Number of HSCs	HSC designation
CPU 1211C	3 (with digital signal board 4)*	HSC1…3 (and HSC5)*
CPU 1212C	4 (with digital signal board 5)*	HSC1…4 (and HSC5)*
CPU 1214C CPU 1215C CPU 1217C	6	HSC1…6

* with DI2/DO2 signal board

How it works

The first of several default values is loaded on the high-speed counter. The required outputs are enabled for the time during which the current value of the counter is lower than the default value. The counter is set up so that an interrupt occurs if the current value of the counter is equal to the default value or when the counter is reset.

If the current value is equal to the default value and an interrupt event results, a new default value is loaded and the next signal state is set for the outputs. If an interrupt event occurs because the counter is reset, the first default value and the first signal states of the outputs are set and the cycle repeated.

Since the interrupts occur much less frequently than the high-speed counter counts, a precise control of the fast operations can be implemented with only a slight influence on the overall cycle of the automation system. Since you can assign specific interrupt programs to interrupts, each new default can be loaded in a separate interrupt program allowing simple control of the state.

Note

You can also process all interrupt events in a single interrupt program.

Count algorithms of the various counters

All counters work in the same way, however some high-speed counters do not support all count algorithms. There are four basic count algorithms:

- Single-phase counter with internal direction control
- Single-phase counter with external direction control
- 2-phase counter with 2 clock inputs
- A/B counter

Each high-speed counter can be used with or without a reset input. If the reset input is activated, this resets the current value. The current value remains reset until the reset input is deactivated.

See also

Configuring high-speed counters (Page 1327)

Interdependency of the counter mode and counter inputs (Page 1325)

Interdependency of the counter mode and counter inputs (S7-1200)

General information on counter mode and counter inputs

You can assign not only the counter modes and counter inputs to the high-speed counters but also functions such as clock pulse generator, direction control, and reset. The following rules apply:

- An input cannot be used for two different functions.
- If an input is not required by the current counter mode of the defined high-speed counter, it can be used other purposes.

If, for example, you set HSC1 to counter mode 1, in which inputs I0.0 and I0.3 are required, you can use I0.1 for edge interrupts or for HSC2.

If, for example, you set HSC1 and HSC5, inputs I0.0 (HSC1) and I1.0 (HSC5) are always used with the counting and frequency counter modes. As a result, these two inputs are not available for any other functions when counters are operated.

You have additional inputs available if you use a digital signal board.

Overview of the interdependency of counter mode and counter inputs

Counter mode	Description	Inputs		
	HSC1	I0.0 (CPU) I4.0 (signal board)	I0.1 (CPU) I4.1 (signal board)	I0.3 (CPU) I4.3 (signal board)
	HSC2	I0.2 (CPU) I4.2 (signal board)	I0.3 (CPU) I4.3 (signal board)	I0.1 (CPU) I4.1 (signal board)
	HSC3*	I0.4 (CPU)	I0.5 (CPU)	I0.7 (CPU)
	HSC4 (CPU 1212/14/15/17C only)	I0.6 (CPU)	I0.7 (CPU)	I0.5 (CPU)
	HSC5 (CPU 1214/15/17C only)**	I1.0 (CPU) I4.0 (signal board)	I1.1 (CPU) I4.1 (signal board)	I1.2 (CPU) I4.3 (signal board)
	HSC6 (CPU 1214/15/17C only)**	I1.3 (CPU)	I1.4 (CPU)	I1.5 (CPU)
Counting / frequency	Single-phase counter with internal direction control	Clock pulse generator	-	-
Counting				Resetting
Counting / frequency	Single-phase counter with external direction control	Clock pulse generator	Direction	-
Counting				Resetting
Counting / frequency	2-phase counter with 2 clock inputs	Clock pulse generator forwards	Clock pulse generator backwards	-
Counting				Resetting
Counting / frequency	A/B counter	Clock pulse generator A	Clock pulse generator B	-
Counting				Resetting
Motion axis	Pulse generators PWM/PTO	HSC1 and HSC2 support the motion axis count mode for the PTO1 and PTO2 pulse generators: • For PTO1, HSC1 evaluates the Q0.0 outputs for the number of pulses. • For PTO2, HSC2 evaluates the Q0.2 outputs for the number of pulses. Q0.1 is used as the output for the motion direction.		

* HSC3 can only be used for CPU 1211 without a reset input

** HSC5 can be also be used for CPU 1211/12 if a DI2/DO2 signal board is used

See also

General information on high-speed counters (Page 1324)

Configuring high-speed counters (Page 1327)

Configuring high-speed counters (S7-1200)

Requirement

An S7-1200 CPU has been inserted in the hardware configuration.

Procedure

To configure a high-speed counter, follow these steps:

1. Select an S7-1200 CPU in the device or network view.
2. Click on the required high-speed counter under "Properties > High-speed counter (HSC)" in the Inspector window:
   - CPU 1211C: HSC1 to HSC3 (also HSC5 with a DI2/DO2 signal board)
   - CPU 1212C: HSC1 to HSC4 (also HSC5 with a DI2/DO2 signal board)
   - CPU 1214C / 1215C / 1217C: HSC1 to HSC6
3. Enable the high-speed counter in the "General" parameter group using the relevant check box.

   Note

   If you use a CPU 1211C or CPU 1212C with a DI2/DO2 signal board, you can also enable the high-speed counter HSC5.

   Note

   When you activate the pulse generators and operate them as PTO (to control stepper motors), high-speed counters are required.

   With firmware versions as of V3.0, the CPU uses internal (additional) HSCs.

   With firmware versions lower than V3.0, the CPU uses HSC1 for PTO1 and HSC2 for PTO2. The counters are then no longer available for other counting tasks.

   In "Project information" you can enter a name and a comment for the counter.

4. Define the functionality of the counter in the "Function" parameter group:
   – Count mode: Select one of the following functions of the counter from the drop-down list:
     - Counting (decides the number of pulses)
     - Period (measurement of the period of pulses)
     - Frequency (measurement of the frequency of pulses)
     - Motion Control (use counter for motion control).
   – Operating phase: Select from the drop-down list how the counter will be operated:
     - Single phase (one clock input for counting, up and down can be set via the direction input)
     - Two-phase (one clock input for counting up, one clock input for counting down)
     - A/B counter (two clock inputs with phase-shifted pulses, single count value)
     -A/B counter fourfold (two clock inputs with phase-shifted pulses, quadruple count value)
   – Input source (only in firmware versions lower than 4.0): As the input source for the count pulses, select a CPU input or an input of a signal board.
   – Count direction is specified by: If the operating phase "Single phase" is set, you can choose whether the user program decides the count direction (instruction CTRL_HSC, parameters DIR and NEW_DIR) or a digital input.
   – Initial count direction: If the user program is set as the direction control for the count direction, you can select the count direction at the start of counting.
   – Frequency meter period: If frequency or period is set as the count mode, you can select the duration of the frequency measurement periods in the drop-down list.
5. Specify the initial values and reset condition of the counter in the "Reset to initial values" parameter group:
   – Initial counter value: Enter a start value for the counter.
   – Initial reference value: Enter a reference value.
     The current count is compared with this reference value: When the current counter value is equal to the reference value, an event is triggered (if enabled in "Event configuration").

   Here, you can also set whether the counter uses a reset input and the counter resets the current count at a high or low level at the reset input.
6. The following events can be enabled in the "Event configuration" parameter group:
   – Counter value equal to reference value
   – External reset
   – Direction change

   You can assign a hardware interrupt OB to each of these events under "Hardware interrupt". Alternatively, assign a hardware interrupt OB to every event in the user program ("ATTACH" instruction).
   Result: When an event is triggered, the CPU executes the assigned hardware interrupt OB (according to the configured priority).
   The names of the events and the hardware interrupts can be changed.

7. In the parameter group "Hardware inputs", hardware inputs are already entered. You can accept these. Alternatively, click the "..." button beside "Clock generator up", "Clock generator down" or "Reset input" and select an input.
   Note: With earlier firmware versions (lower than V4.0), the hardware inputs were preset and could not be changed.
8. In the "I/O addresses" area, the following parameters can be set:
   - Start address: The CPU writes the counter status to the address area with the set start address (data type DInt).
   - Organization block: Assignment of this address area to an organization block.
   - Process image: Assignment of the address area to a process image partition (PIP). The default setting is automatic update (PIP0).

Result

You have now adapted the parameters of the high-speed counter to the requirements of your project.

See also

General information on high-speed counters (Page 1324)

Interdependency of the counter mode and counter inputs (Page 1325)

Point-to-point communication (S7-1200)

Overview of point-to-point communication (S7-1200)

PtP communication is communication via a serial interface that uses standardized UART data transmission (Universal Asynchronous Receiver/Transmitter). The S7-1200 uses communications modules with an RS-232 or RS-485 interface to establish PtP communication.

Functions of point-to-point communication

Point-to-point communication (PtP) allows numerous applications:

- Direct transmission of information to an external device, for example a printer or a barcode reader
- Reception of information from external devices such as barcode readers, RFID readers, cameras and third-party optical systems as well as many other devices.
- Exchange of information with third-party devices, for example GPS devices, radio modems and many others

The Freeport protocol

The S7-1200 supports the Freeport protocol for character-based serial communication. Using Freeport communication, the data transmission protocol can be configured entirely by the user program.

Siemens provides libraries with Freeport communication functions that you can use in your user program:

- USS Drive protocol
- Modbus RTU Master protocol
- Modbus RTU Slave protocol

See also

Configuring a communications port (Page 1330)

Using RS-232 and RS-485 communications modules (S7-1200)

Communications modules with RS-232 and RS-485 interfaces

In an S7-1200 CPU, you can use two different communications modules:

- RS-232 communications module
- RS-485 communications module

The communications modules can be connected to the S7-1200 CPU via the I/O channel on the left. You can plug in up to three different modules.

Properties of the communications modules

The communications modules have the following features:

- Support of the Freeport protocol
- Configuration by the user program with the aid of expanded instructions and library functions

Configuring a communications port (S7-1200)

Configuring a communications port

After you have inserted a communications module with an RS-232 or RS-485 interface, you then set the interface parameters. You set the parameters for the interface either in the properties of the interface or you control the interface parameters from the user program using the PORT_CFG instruction. The following description relates to the graphic configuration.

Note

If you use the user program to change the port setting, the settings of the graphic configuration are overwritten.

You should also keep in mind that the settings made by the user program are not retained if there is a power down.

Requirement

- A communications module is already plugged in.
- You are in the device view.

Procedure

To configure the communications port, proceed as follows:

1. Select the interface in the graphic representation in the device view.
   The properties of the interface are displayed in the Inspector window.
2. Select the "Port configuration" group in the area navigation of the Inspector window.
   The settings of the port are displayed.
3. From the "Transmission speed" drop-down list, select the speed for the data transmission.
   With user-programmed communication, remember the influence of the transmission speed on the changeover time.
4. From the "Parity" drop-down list, select the type of detection of bad information words.
5. Using the "Data bits" drop-down list, decide whether a character consists of eight or seven bits.
6. From the "Stop bit" drop-down list, select how many bits will identify the end of a transmitted word.
7. From the "Flow control" drop-down list, select the method for ensuring a trouble-free data stream between sender and receiver. This parameter can only be set for the RS-232 interface.
   - Enter a HEX value in the "XON character" box that will cause the transmission of the message to be continued when it is detected. This parameter can only be set for software-controlled data flow control.
   - Enter a HEX value in the "XOFF character" box that will cause the transmission of the message to be suspended for the set wait time. This parameter can only be set for software-controlled data flow control.
8. In the "Wait time" box, enter a wait time in ms that must be kept to after the end of the message before the next transmission can start.

Note

You can configure the interface in the network view as well. To do so, you must first select the communication module in the tabular network view and then select the interface in the Inspector window. Then you can continue as described above.

See also

Setting data flow control (Page 1332)

Setting data flow control (S7-1200)

Data flow control

Data flow control is a method that ensures a balanced send and receive behavior. In the ideal situation, the intelligent control does not allow data to be lost. It ensures that a device does not send more information than the receiving partner can process.

There are two methods of data flow control:

- Hardware-controlled data flow control
- Software-controlled data flow control

With both methods, the DSR signals of the communications partners must be active at the beginning of transmission. If the DSR signals are inactive, the transmission is not started.

The RS-232 communications module can handle both methods. The RS-485 communications module does not support data flow control.

Hardware-controlled data flow control

Hardware-controlled data flow control uses the request to send (RTS) and clear to send (CTS) signals. With the RS-232 communications module, the RTS signal is transmitted via the output by pin 7. The CTS signal is received via pin 8.

If hardware-controlled data flow control is enabled, the RTS signal is then always set to activated when data is sent. At the same time, the CTS signal is monitored to check whether the receiving device can accept data. If the CTS signal is active, the module can transfer data until the CTS signal becomes inactive. If the CTS signal is inactive, the data transfer must be suspended for the set wait time. If the CTS signal is still inactive after the set wait time, the data transfer is aborted and an error is signaled back to the user program.

Data flow control using hardware handshaking

If data flow control is controlled by hardware handshaking, the sending device sets the RTS signal to active as default. A device such as a modem can then transfer data at any time. It does not wait for the CTS signal of the receiver. The sending device itself monitors it own transmission by sending only a limited number of frames (characters), for example to prevent overflow of the receive buffer. If there is nevertheless an overflow, the transferring device must hold back the message and signal an error back to the user program.

Software-controlled data flow control

Software-controlled data flow control uses certain characters within the messages and these control the transfer. These characters are ASCII characters selected for XON and XOFF.

XOFF indicates when a transmission must be suspended. XON indicates when a transmission can be continued.

If the sending device receives the XOFF character, it must suspend sending for the selected wait time. If the XON character is sent after the selected wait time, the transfer is continued. If no XON character is received after the wait time, an error is signaled back to the user program.

Software data flow control requires full duplex communication because the receiving partner needs to send the XON character during the ongoing transfer.

See also

Configuring a communications port (Page 1330)

Configuration of message transfer (S7-1200)

User-programmed communication

You can control the data traffic between a communications module and a device connected externally via the serial interface using your own mechanisms. If you want to do this, you will need to define a communications protocol yourself. In freely programmable communication, ASCII and binary protocols are supported for message transfer.

Within the communications protocol, you will need to specify the criteria by which the start and end of a transferred message can be recognized in the data stream.

User-programmed communication can only be activated in RUN mode. If there is a change to STOP mode, the user-programmed communication is stopped.

Specifying the communications protocol

You can specify the communications protocol as follows:

- With the user program
  - The behavior when sending data is controlled by the SEND_CFG instruction.
  - The behavior when receiving data is controlled by the RCV_CFG instruction.
- Using parameter settings set graphically in the Inspector window

Note

If you change the communications protocol from the user program, the settings of the graphic configuration are overwritten.

You should keep in mind that the settings made by the user program are not retained if there is a power down.

See also

User-programmed communication with RS-232 devices (Page 1334)

Making the settings for sending (Page 1335)

Specifying the start of the message (Page 1336)

Specifying the end of the message (Page 1337)

User-programmed communication with RS-232 devices (S7-1200)

RS-232/PIP multi-master cable and user-programmed communication with RS-232 devices

Using the RS-232/PIP multi-master cable and user-programmed communication, you can connect a wide variety of RS-232-compliant devices to the communications modules of the S7-1200. The cable must, however, be set to the "PIP/user-programmed communication" mode.

Settings on the cable

The switches on the cable must be set as follows:

- Switch 5 must be set to 0
- Switch 6 sets either the local mode (DCE) or the remote mode (DTE):
  - Switch set to 0 for the local mode
  - Switch set to 1 for the remote mode

Changing over between send and receive mode

The RS-232/PIP multi-master cable is in send mode when data is sent from the RS-232 interface to the RS-485 interface. The cable is in receive mode when it is idle or when data is sent from the RS-485 interface to the RS-232 interface. The cable changes from receive to send mode immediately when it detects characters on the RS-232 send line.

Supported transmission speeds

The RS-232/PIP multi-master cable supports transmission rates between 1200 baud and 115.2 kbaud. The RS-232/PIP multi-master cable can be set to the required transmission rate using the DIP switch on the PC/PIP cable.

The following table shows the switch settings for the various transmission speeds:

Transmission speed	Switchover time	Settings (1 = up)
115200 bps	0.15 ms	110
57600 bps	0.3 ms	111
38400 bps	0.5 ms	000
19200 bps	1.0 ms	001
9600 bps	2.0 ms	010
4800 bps	4.0 ms	011
2400 bps	7.0 ms	100
1200 bps	14.0 ms	101

The cable returns to receive mode when the RS-232 send line is idle for a certain time that is defined as the changeover time of the cable. The set transmission speed influences the changeover time as shown in the table.

Influence of the changeover time

When working with an RS-232/PIP multi-master cable in a system in which user-programmed communication is used, the program must take into account the changeover time for the following reasons:

- The communications module reacts to messages sent by the RS-232 device.
  Once the communications module has received a request from the RS-232 device, it must delay the reaction message for a period that is equal to or longer than the changeover time of the cable.
- The RS-232 device reacts to messages sent by the communications module.
  Once the communications module has received a reaction message from the RS-232 device, it must delay the next request message for a period that is equal to or longer than the changeover time of the cable.

In both situations, the RS-232-PIP multi-master cable has enough time to change from send to receive mode so that the data can be sent from the RS-485 interface to the RS-232 interface.

See also

Configuration of message transfer (Page 1333)

Making the settings for sending (Page 1335)

Specifying the start of the message (Page 1336)

Specifying the end of the message (Page 1337)

Making the settings for sending (S7-1200)

Sending messages

You can program pauses between individual messages.

The following table shows which pauses can be set:

Parameter	Definition
RTS ON delay	You can set the time that must elapse after the send request RTS (request to send) before the actual data transfer starts.
RTS OFF delay	You can set the time that must elapse after the complete transfer before RTS signal is deactivated.
Send pause at the start of the message	You can specify that a pause is sent at the start of every message transfer when the RTS ON delay has elapsed. The pause is specified in bit times.
Send Idle Line after a pause	You can make a setting so that following a selected pause at the start of the message, the "Idle Line" signal is output to signal that the line is not in use. To enable the parameter, "Send pause at message start" must be set. The duration of the "Idle Line" signal is specified in bit times.

See also

Specifying the start of the message (Page 1336)

Specifying the end of the message (Page 1337)

User-programmed communication with RS-232 devices (Page 1334)

Specifying the start of the message (S7-1200)

Recognizing the start of the message

To signal to the receiver when the transfer of a message is completed and when the next message transfer starts, criteria must be specified in the transmission protocol to identify the end and start of a message.

If a criterion is met that indicates the start of a message, the receiver starts searching the data stream for criteria that mean the end of the message.

There are two different methods for identifying the start of a message:

- Starting with any character:
  Any character can defined the start of a message. This is the default method.
- Starting with a specific condition:
  The start of a message is identified based on selected conditions.

Conditions for detecting the start of a message

The following table shows the various options for defining the start of a message:

Parameter	Definition
Recognize start of message by line break	The receiver recognizes a line break when the received data stream is interrupted for longer than one character. If this is the case, the start of the message is identified by the line break.
Recognize start of message by idle line	The start of a message is recognized when the send transmission line is in the idle state for a certain time (specified in bit times) followed by an event such as reception of a character.
Recognize start of message with individual characters	The start of a message is recognized when a certain character occurs. You can enter the character as a HEX value.
Recognize start of message by a character string	The start of a message is detected when one of the specified character sequences arrives in the data stream. You can specify up to four character sequences each with up to five characters.

The individual conditions can be logically linked in any way.

See also

Making the settings for sending (Page 1335)

User-programmed communication with RS-232 devices (Page 1334)

Specifying the end of the message (S7-1200)

Recognizing the end of the message

To signal to the receiver when the transfer of a message is completed and when the next message transfer starts, criteria must be specified in the transmission protocol to identify the end and start of a message.

In total, there are six different methods of recognizing the end of a message and these can all be logically linked in any way. The following table shows the various possible setting options:

Parameter	Definition
Recognize end of message by message timeout	The end of a message is recognized automatically when a selected maximum duration for a message is exceeded. Values from 0 to 65535 ms can be set.
Recognize end of message by reply timeout	The end of a message is recognized when there is no reply within a set time after transferring data. Values from 0 to 65535 ms can be set.
Recognize end of message by timeout between characters	The end of a message is detected when the time between two characters specified in bit times is exceeded. Values from 0 to 2500 bit times can be set. The S7-1200 CPU only accepts a maximum time of eight seconds even if the value that is set results in a duration of more than eight seconds.
Recognize end of message by maximum length	The end of a message is recognized when the maximum length of a message is exceeded. Values from 1 to 1023 characters can be set.
Read message length from message	The message itself contains information about the length of the message. The end of a message is reached when the value taken from the message is reached. Which characters are used for the evaluation of the message length is specified with the following parameters: • Offset of the length field in the message The value decides the position of the character in the message that will be used to indicate the message length. Values from 1 to 1022 characters can be set. • Size of the length field This value specifies how many characters starting at the first evaluation position will be used to indicate the message length. Values of 0, 1, 2 and 4 characters can be set. • The data following the length field (does not belong to the message length) The value specifies the number of bytes after the length field that must be ignored in the evaluation of the message length. Values from 0 to 255 characters can be set.
Recognize message end with a character sequence	The end of a message is detected when the specified character sequence arrives in the data stream. You can define up to five characters in the character string that are to be checked. If the specified characters appear at the correct location in the message, the message end is recognized. To recognize the message end when character 1 and character 3 have a certain value, for example, you have to activate the check box for character 1 and character 3 and enter a character value.

See also

Making the settings for sending (Page 1335)

User-programmed communication with RS-232 devices (Page 1334)

Enabling system memory (S7-1200)

System memory

A system memory is a bit memory with defined values.

You decide which memory byte of the CPU will become the system memory byte when assigning the system memory parameters.

Benefits

You can use system memory in the user program, for example to run program segments in only the first program cycle after start-up. Two system memory bits are constant 1 or constant 0.

Bits of the system memory bytes

The following table shows the meaning of the system memory:

Bit of the system memory bytes	7	6	5	4	3	2	1	0
Meaning	Reserved (=0)	Reserved (=0)	Reserved (=0)	Reserved (=0)	=0	=1	=1 with change to the diagnosis status	=1 during the startup and in first program cycle after startup, otherwise 0

Note

The selected memory byte cannot be used for intermediate storage of data.

Using clock memory (S7-1200)

Clock memory

A clock memory is a bit memory that changes its binary status periodically in the pulse-no-pulse ratio of 1:1.

You decide which memory byte of the CPU will become the clock memory byte when assigning the clock memory parameters.

Benefits

You can use clock memory, for example, to activate flashing indicator lamps or to initiate periodically recurring operations such as recording of actual values.

Available frequencies

Each bit of the clock bit memory byte is assigned a frequency. The following table shows the assignment:

Bit of the clock memory byte	7	6	5	4	3	2	1	0
Period (s)	2.0	1.6	1.0	0.8	0.5	0.4	0.2	0.1
Frequency (Hz)	0.5	0.625	1	1.25	2	2.5	5	10

Note

Clock memory runs asynchronously to the CPU cycle, i.e. the status of the clock memory can change several times during a long cycle.

The selected memory byte cannot be used for intermediate storage of data.

Protection (S7-1200)

Setting options for the protection level (FW V1 to V3) (S7-1200)

Protection level

The following section describes how to use the various protection levels of the S7-1200 CPUs V1 to V3.

Effects of the protection level setting

You can choose between the following protection levels:

- No protection: This corresponds to the default behavior. You cannot enter a password. Read and write access is always permitted.
- Write protection: Only read-only access is possible. You cannot change any data on the CPU and cannot load any blocks or a configuration. HMI access and communication between CPUs are excluded from the write protection. Assignment of a password is required to select this protection level.
- Write/read protection: No write or read access is possible in the "Accessible devices" area or in the project for devices that are switched online. Only the CPU type and the identification data can be displayed in the project tree under "Accessible devices". Display of online information or blocks under "Accessible devices", or in the project for devices interconnected online, is possible.
  HMI access and communication between CPUs are excluded from the write protection. Assignment of a password is required to select this protection level.

Behavior of a password-protected CPU during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Validity is checked before the online function is executed. If password protection is in place, you are prompted to enter a password.

Example: The module was assigned write protection and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on with a password.

Access authorization to the protected data is in effect for the duration of the online connection or until the access authorization is manually rescinded with "Online > Delete access rights". Access authorization will also expire when the project is closed.

Note

You cannot restrict functions for process control, monitoring, and communications.

However, some functions are protected due to their use as online data. RUN/STOP in the "Online tools" task card or "Set time of day" in the online and diagnostics view are therefore write-protected.

Setting options for the protection (FW as of V4) (S7-1200)

Protection level

The following section describes how to use the various access levels of the S7-1200 CPUs as of V4.

S7-1200 CPUs provide various access levels to limit the access to specific functions.

The parameters for the access levels are assigned in a table. The green checkmarks in the columns to the right of the respective access level specify which operations are possible without knowing the password of this access level. If you want to use the functions of check boxes that are not selected, a password has to be entered.

NOTICE
Configuring an access level does not replace know-how protection Configuring access levels prevents unauthorized changes to the CPU by restricting download privileges. However, blocks on the memory card are not write- or read-protected. Use know-how protection to protect the code of blocks on the memory card.

Default characteristics

The default access level is "Full access (no protection)". Every user can read and change the hardware configuration and the blocks. A password is not set and is also not required for online access.

The access levels in detail

With an S7-1200 CPU, you can configure the following access levels:

- Full access (no protection): The hardware configuration and the blocks can be read and changed by all users.
- Read access: With this access level, only read access to the hardware configuration and the blocks is possible without entering a password - meaning that you can load the hardware configuration and blocks into the programming device. In addition, HMI access and access to diagnostics data is possible.
  You cannot load blocks or a hardware configuration into the CPU without entering the password. Moreover, writing test functions and firmware updates are **not** possible without a password.
- HMI access: With this access level, only HMI access and access to diagnostics data is possible without entering the password.
  Without entering the password, you can neither load blocks and hardware configuration into the CPU, nor load blocks and hardware configuration from the CPU into the programming device. In addition, the following is **not** possible without a password: Writing test functions, changing the operating state (RUN/STOP) and firmware updates.
- No access (complete protection): When the CPU is completely protected, no read or write access to the hardware configuration and the blocks is possible. HMI access is also not possible. The server function for PUT/GET communication is disabled in this access level (cannot be changed).
  Authorization with the password again provides you full access to the CPU.

Behavior of a password-protected module during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Validity is checked before the online function is executed. If password protection is in place, you are prompted to enter a password.

Example: The module was configured with read access and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on.

Access authorization to the protected data is in effect for the duration of the online connection or until the access authorization is manually rescinded with "Online > Delete access rights".

Each access level allows unrestricted access to certain functions without entering a password, for example, identification using the "Accessible devices" function.

Configuring access levels (S7-1200)

The following section describes how to configure an access level and enter passwords for an S7-1200 CPU as of V4.

For an S7-1200 CPU, you can enter multiple passwords and thereby set up different access rights for individual user groups.

The passwords are entered in a table in such a way that exactly one access level is assigned to each password.

The effect of the password is given in the "Access level" column.

Example

You select the "No access (complete protection)" access level for a standard CPU (in other words, not an F-CPU) and enter a separate password for each of the access levels that lie above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the set passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (Full access (no protection)) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (read access) allows access as if the CPU were write-protected. Users who know this password have read-only access to the CPU.
- The password in row 3 (HMI access) allows access as if the CPU were write-protected and read-protected so that only HMI access is possible for users who know this password.

Procedure

To configure the access levels of an S7-1200 CPU, follow these steps:

1. Open the properties of the module in the inspector window.
2. Open the "Protection" entry in the area navigation.
   A table with the possible access levels appears in the inspector window.

Protection level	Access			Access permission	
Protection	HMI	Read	Write	Password	Confirmation
Full access (no protection)	✓	✓	✓		
Read access	✓	✓			
HMI access	✓				
No access (complete protection)					

3. Activate the required access level in the first column of the table. The green checkmarks in the columns to the right of the respective protection level show you which operations are still available without entering the password.
4. In the "Password" column, specify a password for full access in the first row. In the "Confirmation" column, enter the selected password again to protect against incorrect entries.
   Ensure that the password is sufficiently secure, in other words, that is does not follow a machine-recognizable pattern!
   You must enter a password in the first row "Full access (no protection)". This enables unrestricted access to the CPU for those who know the password, regardless of the selected protection level.
5. Assign additional passwords as needed to other access levels if the selected protection level allows you to do so.
6. Download the hardware configuration so that the access level takes effect.

Result

The hardware configuration and the blocks are protected against unauthorized access according to the set access level. If an operation cannot be executed without a password due to the set access level, a dialog for entering a password is displayed.

Restriction of communication services (S7-1200)

Introduction

The CPU can be the server for a number of communication services. This means that other communication participants can access CPU data even if you have not configured and programmed connections for the CPU.

The local CPU as a server thus does not have the possibility to control communication to the clients.

The parameter "Connection mechanisms" in the "Protection" area of the CPU parameters is used to specify whether this type of communication is permitted or not for the local CPU during operation.

Permit access with PUT/GET communication from remote partners

By default, the "Permit access with PUT/GET communication from remote partners (...)" option is disabled. In this case, read and write access to CPU data is only possible for communication connections that require configuration or programming both for the local CPU and for the communication partner. Access through BSEND/BRCV instructions is possible, for example.

Connections for which the local CPU is only a server (meaning that no configuration/ programming of the communication with the communication partner exists at the local CPU), are therefore not possible during operation of the CPU, for example,

- For PUT/GET, FETCH/WRITE or FTP access via communication modules
- For PUT/GET access from other S7 CPUs
- For HMI access that is realized via PUT/GET communication

If you want to allow access to CPU data from the client side, meaning that you do not want to restrict the communication services of the CPU, activate the "Permit access with PUT/GET communication from remote partners" option.

Group alarms for security events (S7-1200)

What you should know about group alarms for security events

Security events in the diagnostics buffer

The CPU stores information on important actions and events in the diagnostics buffer.

The following security events (event types) lead to an entry in the diagnostics buffer of the S7-1200:

- Going online with the correct or incorrect password.
- Manipulated communications data detected.
- Manipulated data detected on memory card.
- Manipulated firmware update file detected.
- Changed protection level (access protection) downloaded to the CPU.
- Password legitimization restricted or enabled (by instruction or CPU display).
- Online access denied due to the possible number of simultaneous access attempts being exceeded.
- Timeout when an existing online connection is inactive.
- Logging in to the Web server with the correct or incorrect password.
- Creating a backup of the CPU.

- Restoring the CPU configuration.
- During startup:
  - Project on the SIMATIC memory card has changed (the SIMATIC memory card remains the same)
  - SIMATIC memory card was replaced

Group alarm for security events

To prevent the diagnostics buffer being "swamped" by large numbers of identical security events, as of STEP 7 V13 service pack 1, you can set parameters so that these events are entered in the diagnostics buffer as a group alarm. Per interval (monitoring time), the CPU then only generates one group alarm per event type.

Example: Brute-Force attacks

With Brute-Force attacks, attackers attempt to gain access to the CPU by systematically trying out a large number of the possible password combinations. As a result, the CPU receives a large number of login requests within seconds and the inrush of identical single entries in the diagnostics buffer leads to the loss of important diagnostics information. Group alarms help here.

Principle of operation of group alarms

The CPU enters the first three events of an event type in the diagnostics buffer. It then ignores all subsequent security events of this type.

At the end of the monitoring time (interval), the CPU generates a group alarm in which it enters the event type and the frequency of the event during the elapsed interval.

If these security events also occur in the intervals that follow, the CPU only generates one group alarm per interval.

An attack leaves the following pattern in the diagnostics buffer: Three individual alarms followed by a series of group alarms. This series can consist of two, three or more group alarms depending on the selected monitoring time and duration of the attack.

Since the alarms in the diagnostics buffer have a time stamp, the duration of the attack can be determined.

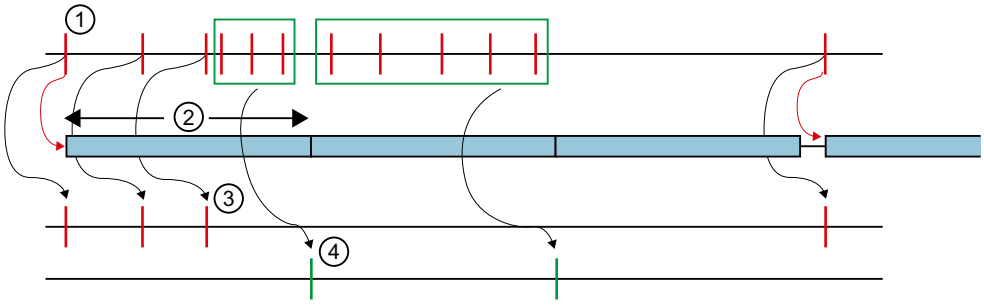

① Security events of one type, for example login attempts with an invalid password.

② Interval (monitoring time): When a security event of a particular type occurs the first time (or reoccurs), the monitoring time is started (or restarted).

③ Single alarms: The first three security events of a type are entered in the diagnostics buffer immediately. As of the fourth security event of this type, the CPU only creates group alarms.

If a security event of this type occurs after a pause of at least one interval, the CPU enters a single alarm in the diagnostics buffer and restarts the monitoring time.

④ Group alarms: After three security events, the CPU only generates a group alarm as a summary of all other security events in this interval. If these security events also occur in the intervals that follow, the CPU only generates one group alarm per interval.

Configuring a group alarm for security events (S7-1200)

Requirements

- STEP 7 V14
- S7-1200-CPU as of firmware V4.2

Procedure

To configure group alarms for security events, follow these steps:

1. Click on the CPU symbol in the network view.
   The properties of the CPU are displayed in the Inspector window.
2. Go to the "Protection" > "Security event" area.
3. Click on "Security event".
4. Select the option "Summarize security events in case of high message volume" to enable group alarms for security events.
5. Set the duration of an interval (monitoring time); the default is 20 seconds.

See also

What you should know about group alarms for security events (Page 1344)

Deactivating SNMP (S7-1200)

The network management protocol SNMP (Simple Network Management Protocol) is a protocol that uses various services and tools for detection and diagnostics of the network topology. SNMP uses the transport protocol UDP. SNMP recognizes two network components, the SNMP manager and the SNMP client. The SNMP manager monitors the network nodes: The SNMP clients collect the various network-specific information in the individual network nodes and store it in a structured form in the MIB (Management Information Base). Various services and tools can run detailed network diagnostics with the help of this data.

Under certain conditions, it may be useful to disable SNMP. Examples:

- The security guidelines in your network do not allow the use of SNMP.
- You use your own SNMP solution, e.g. with your own communications instructions.

If you disable SNMP for a device, various options for diagnostics of the network topology (e.g. using the PRONETA tool or the Web server of the CPU) are no longer available to you.

Requirements

- CPU S7-1200 V4.1 and higher
- CPU S7-1500 V1.8 and higher

Disabling SNMP

To disable SNMP, follow these steps:

1. In STEP 7, create a data block that contains the structure of data record $F003_H$. The following table shows the structure of the data record $F003_H$.

Byte	Element	Code	Explanation
0-1	BlockID	$F003_H$	Header The data record length is counted starting at byte 4 "Version".
2-3	BlockLength	8	
4	Version	01_H	
5	Subversion	00_H	
6-7	Reserved	-	-
8-11	SNMP controller	Disable/enable SNMP	If you want to disable SNMP, enter the value 0. If you want to enable SNMP, enter the value 1.

2. Transfer the data record $F003_H$ in the startup OB (OB100) to the CPU with the WRREC instruction (write data record). Use an integrated interface of the CPU as the hardware ID.

Deactivating SNMP - complete example (S7-1200)

Introduction

The example applies to S7-1500 and S7-1200.

Follow these steps to apply the program code to your project:

1. Copy the entire program code to the clipboard.
2. Open a text editor (e.g. "Editor").
3. Paste the content of the clipboard to the text editor with Ctrl+V. Save the document as scl file, for example, SNMP_DEACT.scl.
4. Open your project in STEP 7.
5. Import the scl file as an external source.
6. Generate the startup OB and the data blocks.
7. Right-click the scl file and select "Generate block from source" in the shortcut menu.

Example program with WRREC call in OB 100

```
DATA_BLOCK "Deactivate_SNMP"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
NON_RETAIN
 VAR
   snmp_deactivate : Bool;
   snmp_record : Struct
     BlockID : UInt;
     BlockLength : UInt;
     "Version" : USInt;
     Subversion : USInt;
     Reserved : UInt;
     SNMPControl : UDInt;
   END_STRUCT;
   snmp_done : Bool;
   snmp_error : Bool;
   snmp_Status : DWord;
 END_VAR

BEGIN
  snmp_deactivate := true;
  snmp_record.BlockID := 16#f003;
  snmp_record.BlockLength := 8;
  snmp_record."Version" := 1;
END_DATA_BLOCK

ORGANIZATION_BLOCK "Startup"
TITLE = "Complete Restart"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
BEGIN
  REPEAT
    "WRREC_DB_1"(REQ := "Deactivate_SNMP".snmp_deactivate,
                 ID := "Local~PROFINET-Schnittstelle_1",
                 INDEX := 16#b071,
                 DONE => "Deactivate_SNMP".snmp_done,
                 ERROR => "Deactivate_SNMP".snmp_error,
```

```
                 STATUS => "Deactivate_SNMP".snmp_Status,
                 RECORD := "Deactivate_SNMP".snmp_record);
   UNTIL "Deactivate_SNMP".snmp_done OR "Deactivate_SNMP".snmp_error
   END_REPEAT;
END_ORGANIZATION_BLOCK

DATA_BLOCK "WRREC_DB_1"
{OriginalPartName := 'WRREC';
 VersionGUID := 'bc169451-58cd-44a3-855b-3f78cc0623c8';
 S7_Optimized_Access := 'TRUE' }
AUTHOR : SIMATIC
FAMILY : DP
NAME : WRREC
VERSION : 1.0
NON_RETAIN
WRREC
BEGIN
END_DATA_BLOCK
```

Symbolic and numerical names of instructions (S7-1200)

Description

The instructions from the task card are comprised of functions (FC), function blocks (FB), system functions (SFC) and system function blocks (SFB) that are identified internally by numbers.

The following tables show the assignment of numerical and symbolic names.

Function blocks (FBs)

Numerical name	Symbolic name
FB 105	TC_CONFIG
FB 110	Port_Config
FB 111	Send_Config
FB 112	Receive_Config
FB 113	Send_P2P
FB 114	Receive_P2P
FB 115	Receive_Reset
FB 116	Signal_Get
FB 117	Get_Features
FB 118	Set_Features
FB 163	TC_SEND
FB 164	TC_RECV
FB 165	TC_CON
FB 166	TC_DISCON
FB 804	SET_TIMEZONE

Numerical name	Symbolic name
FB 1030	TSEND_C
FB 1031	TRCV_S
FB 1071	USS_DRIVE
FB 1080	MB_COMM_LOAD
FB 1081	MB_MASTER
FB 1082	MB_SLAVE
FB 1084	MB_CLIENT
FB 1085	MB_SERVER
FB 1100	MB_Halt
FB 1101	MC_Home
FB 1102	MC_MoveAbsolute
FB 1103	MC_MoveJog
FB 1104	MC_MoveRelative
FB 1105	MC_MoveVelocity
FB 1107	MC_Power
FB 1108	MC_Reset
FB 1110	MC_MoveInterrupt
FB 1111	MC_ChangeDynamik
FB 1112	MC_CommandTable
FB 1113	MC_MoveLinearAbs_2D
FB 1114	MC_MoveLinearRel_2D
FB 1115	MC_MoveCircular_2D
FB 1130	PID_Compact
FB 1134	PID_3Step
FB 1140	HSC
FB 2040	RecipeCreate
FB 2041	RecipeOpen
FB 2042	RecipeRead
FB 2043	RecipeWrite
FB 2044	RecipeAppend
FB 2045	RecipeClose

Functions (FCs)

Numerical name	Symbolic name
FC 2 (1)	CONCAT
FC 4 (1)	DELETE
FC 11 (1)	FIND
FC 17 (1)	INSERT
FC 20 (1)	LEFT
FC 21 (1)	LEN
FC 22 (1)	LIMIT

Numerical name	Symbolic name
FC 25 (1)	MAX
FC 26 (1)	MID
FC 27 (1)	MIN
FC 31 (1)	REPLACE
FC 32 (1)	RIGHT
FC 36 (1)	ENCO
FC 36 (1)	SEL
FC 37	DECO
FC 800	LED
FC 801	IM_DATA
FC 802	DeviceStates
FC 803	ModuleStates
FC 1070	USS_PORT
FC 1072	USS_RPM
FC 1073	USS_WPM
(1) MC7+ instruction	

System data types (SDTs)

Numerical name	Symbolic name
SDT 99	WWW_CDB
SDT 513	CONDITIONS
SDT 581	Send_Conditions
SDT 582	Receive_Conditions

System function blocks (SFBs)

Numerical name	Symbolic name
SFB 0 (1)	CTU
SFB 1 (1)	CTD
SFB 2 (1)	CTUD
SFB 3 (1)	TP
SFB 4 (1)	TON
SFB 5 (1)	TOF
SFB 27	START_OB
SFB 52	RDREC
SFB 53	WRREC
SFB 54	RALRM
SFB 105	T_CONFIG
SFB 106	TDIAG
SFB 107	TRESET
SFB 110	PORT_CFG

Numerical name	Symbolic name
SFB 111	SEND_CFG
SFB 112	RCV_CFG
SFB 113	SEND_PTP
SFB 114	RCV_PTP
SFB 115	SGN_GET
SFB 116	SGN_SET
SFB 117	RCV_RST
SFB 120	CTRL_HSC
SFB 122	CTRL_PWM
SFB 124	CTRL_HSC_EXT
SFB 140	DataLogCreate
SFB 141	DataLogOpen
SFB 142	DateLogWrite
SFB 143	DataLogClear
SFB 144	DataLogClose
SFB 145	DataLogDelete
SFB 146	DataLogNewFile

System functions (SFCs)

Numerical name	Symbolic name
SFC 7	DP_PRAL
SFC 11	DPSYC_FR
SFC 13	DPNRM_DG
SFC 14	DPRD_DAT
SFC 16	RD_OBINF
SFC 23	DEL_DB
SFC 28	SET_TINT
SFC 29	CAN_TINT
SFC 30	ACT_TINT
SFC 31	QRY_TINT
SFC 32	SRT_DINT
SFC 33	CAN_DINT
SFC 34	QRY_DINT
SFC 41	DIS_AIRT
SFC 42	EN_AIRT
SFC 43	RE_TRIGR
SFC 45	D_ACT_DP
SFC 46	STP
SFC 82	CREA_DBL
SFC 83	READ_DBL
SFC 84	WRIT_DBL

Numerical name	Symbolic name
SFC 86	CREATE_DB
SFC 89	RST_EVOV
SFC 99	WWW
SFC 101	RTM
SFC 117	GET_DIAG
SFC 124	ATTR_DB
SFC 140	IO2MOD
SFC 143	RD_ADDR
SFC 154	RD_LOC_T
SFC 154	DPWR_DAT
SFC 161	WR_LOC_T
SFC 180	ID2LOG
SFC 181	LOG2ID
SFC 182	ID2GEO
SFC 190	SET_CINT
SFC 191	QRY_CINT
SFC 192	ATTACH
SFC 193	DETACH
MC7+ Anweisung	GET_ERROR
MC7+ Anweisung	GET_ERR_ID

Organization blocks (S7-1200)

Startup OBs (S7-1200)

Description

You can determine the boundary conditions for the startup characteristics of your CPU, for example, the initialization values for "RUN". To do this, write a startup program. The startup program consists of one or more startup OBs (OB numbers 100 or >= 123).

The startup program is executed once during the transition from "STOP" mode to "RUN" mode. Current values from the process image of the inputs are not available for startup program, nor can these values be set.

After the complete execution of the startup OBs, the process image of the inputs is read in and the cyclic program is started.

There is no time limit for executing the startup routine. Therefore the scan cycle monitoring time is not active. Time-driven or interrupt-driven organization blocks cannot be used.

Start information

A startup OB has the following start information:

Tag	Data type	Description
LostRetentive	BOOL	= 1, if retentive data storage areas have been lost
LostRTC	BOOL	= 1, if realtime clock has been lost

See also

"STARTUP" operating mode (Page 1292)

Events and OBs (Page 1305)

Cyclic OBs (S7-1200)

Introduction

For the program execution to start, at least one program cycle OB must be present in the project. The operating system calls this program cycle OB once in each cycle and thereby starts the execution of the user program. You can use multiple OBs (OB numbers >= 123). When multiple program cycle OBs are used, these are called in one after the other in the order of their OB numbers. The program cycle OB with the lowest OB number is called first.

The program cycle OBs have the priority class 1. This corresponds to the lowest priority of all OBs. The cyclic program can be interrupted by events of any other event class.

Programming cyclic program execution

You program cyclic program execution by writing your user program in the cycle OBs and the blocks that they call.

The first cyclic program execution begins as soon as the startup program has ended without errors. The cycle restarts after the end of each cyclic program execution.

Sequence of cyclic program execution

One cycle of the program execution encompasses the following steps:

1. The operating system starts the maximum cycle time.
2. The operating system writes the values from the process image output to the output modules.
3. The operating system reads out the state of the inputs of the input modules and updates the process image input.
4. The operating system processes the user program and executes the operations contained in the program.
5. At the end of a cycle, the operating system executes any tasks that are pending, for example, loading and deleting blocks or calling other cycle OBs.
6. Finally, the CPU returns to the start of the cycle and restarts the scan cycle monitoring time.

See also: process image input/output (Page 1301)

Options for interrupting

Cyclic program execution can be interrupted by the following events:

- Interrupt
- A STOP command, triggered by
  - Operation of the programming device
  - "STP" instruction
- Supply voltage failure
- Occurrence of a device fault or program error

Start information

- None
- Optimized start information:

Name	Data type	Meaning
first_scan	BOOL	= TRUE in the first call of this OB: • Transition from STOP or HOLD to RUN • After reloading
retentivity	BOOL	= TRUE, if retentive data are available

See also

Events and OBs (Page 1305)

Organization blocks for interrupt-driven program execution (S7-1200)

Time-of-day interrupt OBs (S7-1200)

Function

Organization blocks for time-of-day interrupt (OB number >=123) can be processed as follows:

- One time to a preset time (date with time of day)
- Periodically with preset start time and the following intervals:
  - Every minute
  - Hourly
  - Daily
  - Weekly
  - Monthly
  - Yearly
  - End of month

Time-of-day interrupt-OBs are therefore used to run parts of the user program on a time-controlled basis.

Status for time-of-day interrupt

The following table contains the possible status of a time-of-day interrupt as well as it's meaning.

Status	Meaning
Cancelled	The one-time processing has already taken place, or the start event of an not yet processed time-of-day interrupt has been deleted with the extended instruction CAN_TINT.
Set	You have scheduled the time or start time for processing.
Activated	You have scheduled whether processing takes place one time or periodically, and in the case of periodic processing, you have scheduled the interval.

Rules for time-of-day interrupts

The following rules apply to the use of time-of-day interrupts:

- A time-of-day interrupt can only be processed if it is set and activated, and a corresponding organization block exists in the user program.
- The start times of periodic time-of-day interrupts must correspond to a real date. For example, it is not possible to repeat an organization block monthly which first occurs on January 31st. In this case, on OB will only be started in the months that have 31 days.
- A time-of-day interrupt activated during startup by extended instruction call ACT_TINTwill not be executed until the startup is complete.
- After each CPU startup, you must reactivate previously set time-of-day interrupts

Setting and activating a time-of-day interrupt OB

Before a time-of-day interrupt can be deleted and the corresponding time-of-day interrupt OB can be processed from the operating system, you must set and activate the interrupt. You have the following options:

Set time-of-day interrupt	Activate time-of-day interrupt
Means of configuring	Means of configuring
Means of configuring	By extended instruction call ACT_TINT
By extended instruction call SET_TINTL	By extended instruction call ACT_TINT

Note

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed once, the start time must not be in the past (relative to the real-time clock of the CPU).

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed periodically, but the start time is in the past, then the time-of-day interrupt will be processed the next time it is due.

Time-of-day interrupt status query

In order to query the status of the time-of-day interrupt, call the extended instruction QRY_TINT.

Cancelling a time-of-day interrupt

You can cancel a time-of-day interrupt which has not yet been processed with the extended instruction CAN_TINT.

You can reset cancelled time-of-day interrupts with the extended instruction SET_TINTL and activate them with the extended instruction ACT_TINT.

Conditions that effect the time-of-day interrupt OBs

Since a time-of-day interrupt occurs only at specific intervals, certain conditions can affect the function of the associated OBs during execution of your program. The following table shows some of these conditions and describes how these affect the processing of a time-of-day interrupt OB.

Condition	Result
The extended instruction CAN_TINT is called in the user program.	The operating system deletes the start event (date and time) of the time-of-day interrupt. You must reset and reactivate the time-of-day interrupt if you are going to call the corresponding time-of-day interrupt OB again.
By synchronizing or correcting the CPU system clock, the time of day will be set ahead. With this, the start time for a time-of-day interrupt OB will be bypassed.	The operating system calls the time error interrupt OB (OB 80) and records the start event, the number, and the priority of the first bypassed time-of-day interrupt OB in its start information. After completion of the OB 80, the operating system processes the bypassed time-of-day interrupt OB only once.
By synchronizing or correcting the CPU system clock, the time of day will be set back. The corrected clock time is before the start time of an already processed time-of-day interrupt OB.	The time-of-day interrupt OB is repeated.
A time-of-day interrupt OB is still being processed when the start event for its next execution occurs.	The operating system then calls time error interrupt OB 80. The requested OB is processed only after the processing and further execution of the current time-of-day interrupt OB has completed.

Start information

A time-of-day interrupt OB has the following start information:

Tag	Data type	Description
CaughtUp	BOOL	=1, if the OB call is executed because clock time is turned ahead.
SecondTime	BOOL	=1 if the OB is called a second time because clock time is turned back (more exactly, if the planned time of the current OB processing is earlier than or the same time as the planned time for the previous OB processing). Note: SecondTime is only set once.

Status interrupt OB (S7-1200)

Description

When it receives a status interrupt, the operating system of the S7-1200 CPU calls the status interrupt OB from a DP master or IO controller. This may be the case if a module of a slave changes its operating mode, for example, from "RUN" to "STOP". For more detailed information on events that trigger a status interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

The status-interrupt OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware address of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame

See also

Events and OBs (Page 1305)

Update interrupt OB (S7-1200)

Description

When it receives a status interrupt, the operating system of the S7-1200 CPU calls the update interrupt OB from a DP master or IO controller. This may be the case if you changed a parameter on a slot of a slave or device. For more detailed information on events that trigger an update interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

The update interrupt OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware address of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame

See also

Events and OBs (Page 1305)

OB for vendor- or profile-specific interrupts (S7-1200)

Description

When it receives a manufacturer-specific or profile-specific interrupt from a DP master or IO controller, the operating system of the S7-1200 CPU calls OB 57. For more detailed information on events that trigger this type of interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

The OB for manufacturer-specific and profile-specific interrupts has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware address of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame

See also

Events and OBs (Page 1305)

Time-delay interrupt OBs (S7-1200)

Description

A time-delay interrupt OB is started after a configurable time delay of the operating system. The delay time starts after the SRT_DINT instruction is called.

You can use up to four time-delay interrupt OBs or cyclic interrupt OBs in your program. If, for example, you are already using two cyclic interrupt OBs, you can insert a maximum of two further time-delay interrupt OBs in your program.

You can use the CAN_DINT instruction to prevent the execution of a time-delay interrupt that has not yet started.

Function of time-delay interrupt OBs

The operating system starts the corresponding OB after the delay time, which you have transferred with an OB number and an identifier to the SRT_DINT instruction.

To use a time-delay interrupt OB, you must execute the following tasks:

- You must call the instruction SRT_DINT.
- You must download the time-delay interrupt OB to the CPU as part of your program.

The delay time is measured with a precision of 1 ms. A delay time can immediately start again after it expires.

Time delay interrupt OBs are executed only when the CPU is in the "RUN" mode. A warm restart clears all start events of time-delay interrupt OBs.

The operating system calls the time-delay interrupt OB if one of the following events occurs:

- If the operating system attempts to start an OB that is not loaded and you specified its number when calling the SRT_DINT instruction.
- If the next start event for a time-delay interrupt occurs before the time delay OB has completely executed.

You can disable and re-enable time-delay interrupts using the DIS_AIRT and EN_AIRT instructions.

Note

If you disable an interrupt with DIS_AIRT after executing SRT_DINT, this interrupt executes only after it has been enabled with EN_AIRT. The delay time is extended accordingly.

Start information

- None
- Optimized start information:

Name	Data type	Meaning
sign	WORD	User ID: Input parameter SIGN from the call of the "SRT_DINT" instruction

Note

"sign" tag

The "sign" tag exists only since firmware version V4.1 of the S7-1200 CPUs. See Useful information on CPU firmware versions and STEP 7 versions (Page 1375)

See also

Events and OBs (Page 1305)

Cyclic interrupt OBs (S7-1200)

Description

Cyclic interrupt OBs serve to start program in periodic time intervals independently of the cyclic program execution. The start times of a cyclic interrupt OB are specified using the time base and the phase offset.

The time base defines the intervals at which the cyclic interrupt OB is started and is an integer multiple of the basic clock cycle of 1 ms. The phase offset is the time by which the start time is offset compared with the basic clock cycle. If several cyclic interrupt OBs are being used, you can use this offset to prevent a simultaneous start time if the time bases of the cyclic interrupt OBs have common multiples.

You can specify a time period between 1 ms and 60000 ms as the time base.

You can use up to four cyclic interrupt OBs or time-delay interrupt OBs in your program. If, for example, you are already using two time-delay interrupt OBs, you can insert a maximum of two further cyclic interrupt OBs in your program.

Note

The execution time of each cyclic interrupt OB must be noticeably smaller than its time base. If a cyclic interrupt OB has not been completely executed but execution is again pending because the cycle clock has expired, the time error OB is started. The cyclic interrupt that caused the error is executed later or discarded.

Example of the use of phase offset

You have inserted two cyclic interrupt OBs in your program:

- Cyclic interrupt OB1
- Cyclic interrupt OB2

For cyclic interrupt OB1, you have set a time base of 20 ms and for cyclic interrupt OB2 a time base of 100 ms. After expiration of the time base of 100 ms, the cyclic interrupt OB1 reaches the start time for the fifth time, cyclic interrupt OB2 for the first time. To nevertheless execute the cyclic interrupt OBs offset, enter a phase offset for one of the two cyclic interrupt OBs.

Start information

- None
- Optimized start information:

Name	Data type	Meaning
first_scan	BOOL	= TRUE in the first call of this OB • At the transition from STOP or HOLD to RUN • After reloading
event_count	INT	Number of lost start events since the last start of this OB

See also

Assigning parameters to cyclic interrupt OBs (Page 1372)

Events and OBs (Page 1305)

Hardware interrupt OBs (S7-1200)

Description

You can use hardware interrupt OBs to react to specific events. You can assign an event that triggers an alarm to precisely one hardware interrupt OB. A hardware interrupt OB in contrast can be assigned to several events.

Hardware interrupts can be triggered by high-speed counters and input channels. For each high-speed counter and input channel that should trigger a hardware interrupt, the following properties need to be configured:

- The process event that should trigger the hardware interrupt (for example, the change of a count direction of a high-speed counter)
- The number of the hardware interrupt OB which is assigned to this process event

You can use up to 50 hardware interrupt OBs (OB numbers >= 123) that are independent of each other in your program.

Functionality of a hardware interrupt OB

After triggering a hardware interrupt, the operating system identifies the channel of the input or the high-speed counter and determines the assigned hardware interrupt OB.

If no other interrupt OB is active, the determined hardware interrupt OB is called. If a different interrupt OB is already being executed, the hardware interrupt is placed in the queue of its priority class. The hardware interrupt is acknowledged after the completion of the assigned hardware interrupt OB.

If another event that triggers a hardware interrupt occurs on the same module during the time between identification and acknowledgement of a hardware interrupt, the following applies:

- If the event occurs on the channel that previously triggered the hardware interrupt, then no additional hardware interrupt is triggered. Another hardware interrupt can only be triggered if the current hardware interrupt is acknowledged.
- If the event occurs on a different channel, a hardware interrupt is triggered.

Hardware interrupt OBs are called only in the CPU's "RUN" mode.

Start information

- None
- Optimized start information:

Name	Data type	Meaning
Laddr	HW_IO	Hardware identifier of the module that triggers the hardware interrupt
USI	WORD	Identifier for future extensions (not user-relevant)
IChannel	USINT	Number of the channel that triggered the hardware interrupt
EventType	BYTE	Identifier for the event type associated with the event triggering the interrupt (e.g., positive edge) This identifier can be found in the description of the respective module.

See also

Assigning parameters to hardware interrupt OBs (Page 1373)

Events and OBs (Page 1305)

Time error OB (S7-1200)

Description

The operating system calls the time error OB (OB 80) if one of the following events occurs:

- The cyclic program exceeds the maximum cycle time.
- The OB called is currently still being executed (possible for time-delay interrupt OBs and cyclic interrupt OBs).
- A time-of-day interrupt was missed because the clock time was set forward by more than 20 seconds.
- A time-of-day interrupt was missed during a STOP.
- An overflow has occurred in an interrupt OB queue.
- An interrupt was lost due to the excessive interrupt load.

If you have programmed no time error OB, the S7-1200 CPU reacts as follows:

- CPUs with firmware version V1.x to V3.x: The CPU remains in RUN mode.
- CPUs with firmware version as of V4.x:
  - The CPUs goes to STOP mode when the maximum cycle time is exceeded.
  - With all other start events of the time error OB, the CPU remains in RUN mode.

Violation of the maximum cycle time twice does not lead to the calling of an OB, but to the STOP of the CPU. You can avoid the second violation by restarting the cycle monitoring of the CPU with the RE_TRIGR instruction.

You can use only one time error OB in your program.

Start information

The time error OB has the following start information:

Tag	Data type	Description
fault_id	BYTE	• 0x01: Maximum cycle time exceeded • 0x02: Called OB is still being executed • 0x05: Expired time-of-day interrupt due to time jump • 0x06: Expired time-of-day interrupt on return to RUN mode • 0x07: Queue overflow • 0x09: Interrupt loss due to high interrupt load
csg_OBnr	OB_ANY	Number of the OB being executed at the time of the error
csg_prio	UINT	Priority of the OB being executed at the time of the error

See also

Events and OBs (Page 1305)

Cycle time and maximum cycle time (Page 1320)

Diagnostic interrupt OB (S7-1200)

Description

You can enable the diagnostic error interrupt for diagnostics-capable modules so that the module detects changes in the I/O status. As a result, the module triggers a diagnostic error interrupt in the following cases:

- A fault is present (incoming event)
- A fault is no longer present (outgoing event)

If no other interrupt OB is active, then the diagnostic interrupt OB (OB 82) is called. If another interrupt OB is already being executed, the diagnostic error interrupt is placed in the queue of its priority group.

You can use only one diagnostic interrupt OB in your program.

Start information

The diagnostic interrupt OB has the following start information:

Tag	Data type	Description
IO_state	WORD	Contains the I/O status of the diagnostics-capable module.
laddr	HW_ANY	HW-ID
Channel	UINT	Channel number
multi_error	BOOL	= 1, if there is more than one error

IO_state tag

The following table shows the possible I/O states that the IO_state tag can contain:

IO_state	Description
Bit 0	Configuration correct: = 1, if the configuration is correct = 0, if the configuration is no longer correct
Bit 4	Error: = 1, if an error is present, e.g., a wire break = 0, if the error is no longer present
Bit 5	Configuration not correct: = 1, if the configuration is not correct = 0, if the configuration is correct again
Bit 7	I/O cannot be accessed: = 1, if an I/O access error has occurred In this case, laddr contains the hardware identifier of the I/O with the access error. = 0, if the I/O can be accessed again
Bits 8 to 15	Reserved (always 0)

See also

Events and OBs (Page 1305)

Pull/plug interrupt OB (S7-1200)

Description

The S7-1200 CPU operating system calls the pull/plug interrupt OB (OB 83) if a configured and non-disabled distributed I/O module or submodule (PROFIBUS, PROFINET, AS-i) is removed or inserted.

Note

The removal or insertion of a central module leads to the STOP of the CPU.

Start information

The pull/plug interrupt OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the affected module or submodule
Event_Class	BYTE	• B#16#38: (Sub)module plugged • B#16#39: (Sub)module pulled or not responding
Fault_ID	BYTE	Error code (possible values: B#16#51, B#16#54, B#16#55, B#16#56, B#16#57, B#16#58)

The following table shows which event caused the start of the pull/plug interrupt OB.

ev_class (B#16# ...)	fault_id (B#16# ...)	Meaning
39	51	Module pulled
39	54	Submodule pulled
38	54	Submodule plugged, which conforms to the parameterized submodule
38	55	Submodule inserted, which does not conform to the submodule parameter assignment
38	56	Submodule inserted, but error in module parameter assignment
38	57	Submodule or module inserted, but with a fault or maintenance
38	58	Submodule access error remedied

See also

Events and OBs (Page 1305)

Rack error OB (S7-1200)

Description

The S7-1200 CPU operating system calls the OB 86 in the following cases:

- The failure of a DP master system or of a PROFINET IO system is detected (in the case of either an incoming or an outgoing event).
- The failure of a DP slave or of an IO device is detected (in the case of either an incoming or an outgoing event).
- Failure of some of the submodules of a PROFINET I-device is detected.

Structure of the start information

The rack fault OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the defective hardware object
Event_Class	BYTE	• B#16#32: Activation of a DP slaves or an IO device • B#16#33: Deactivation of a DP slaves or an IO device • B#16#38: outgoing event • B#16#39: incoming event
Fault_ID	BYTE	Error code (possible values: B#16#C3, B#16#C4, B#16#C5, B#16#C6, B#16#C7, B#16#C8, B#16#C9, B#16#CA, B#16#CB, B#16#CC, B#16#CD, B#16#CE, B#16#CF, B#16#F8, B#16#F9)

The following table shows which event caused the start of OB 86.

Ev_class B#16# ...	Fault_id B#16# ...	Meaning
39	C3	Failure of a DP master system
39/38	C4	Failure/return of a DP slave
38	C5	Return of a DP slave; however, slave is faulty
38	C6	Expansion unit return, but error in module parameter assignment
38	C7	Return of a DP device, but there is error in module configuration
38	C8	Return of a DP device, but discrepancy between preset/actual configuration
32/33	C9	Activation/deactivation of a DP slave with the "D_ACT_DP" instruction
39	CA	Failure of a PROFINET IO system
39/38	CB	Failure/return of a PROFINET IO device
38	CC	Return of a PROFINET IO device with fault or maintenance
38	CD	Return of a PROFINET IO device, deviation between preset/actual configuration
38	CE	Return of a PROFINET IO device, error in module configuration
32/33	CF	Activation/deactivation of an IO device with the "D_ACT_DP" instruction
39/38	F8	Failure/return of some of the submodules of a PROFINET I-device
38	F9	Return of some of the submodules of a PROFINET I-device with a device configuration difference

See also

Events and OBs (Page 1305)

MC-servo OB (S7-1200)

Description

When you create a technology object with drive connection using MC-Servo [OB91]/create an analog output, the organization block PROFIdrive is created automatically. The Motion Control functionality of the technology objects creates its own execution level, and is called according to the Motion Control application cycle.

The MC-Servo OB is write-protected. The content cannot be changed.

The position control algorithms of all technology objects configured for motion control on the CPU are calculated within the MC-Servo OB.

You can set the application cycle and the priority of the organization block in accordance with your requirements for control quality and system load.

Application cycle

You can set the application cycle in which the MC-Servo OB is called in the properties of the organization block.

The MC-Servo OB is called cyclically with the specified application cycle.

The selected application cycle must be long enough to allow all the technology objects to be processed in one cycle. If the processing time of the technology objects is longer than the application cycle, overflows will occur.

To avoid disruptions in the program execution on the CPU, set the application cycle depending on the number of axes used as follows:

Application cycle = number of axes x 2 ms

The following table shows the resulting application cycle as an example according to the number of axes:

Number of axes	Application cycle
1	2 ms
2	4 ms
4	8 ms
8	16 ms

The SINAMICS G120 drive updates the drive process image every 4 ms. To improve control, set the application cycle of the MC-Servo OB to 4 ms or to a multiple of 4 ms.

See also

MC-PreServo OB (Page 1369)

MC-PostServo OB (Page 1369)

MC-interpolator OB (Page 1370)

Assigning the MC-Servo OB parameters (Page 1375)

MC-PreServo OB (S7-1200)

Description

The organization block MC-PreServo [OB67] can be programmed and is called in the application cycle configured at the MC-Servo OB (Page 1368). MC-PreServo [OB67] is called directly before MC-Servo [OB91].

Via the organization block, you can read out the configured application cycle (information in µs).

Structure of the start information

Optimized start information:

Name	Data type	Meaning
Initial_Call	BOOL	=TRUE in the first call of this OB during transition from STOP to RUN
PIP_Input	BOOL	TRUE: The associated process image input is up-to-date
PIP_Output	BOOL	TRUE: The associated process image output was transferred to the outputs in good time after the last cycle
IO_System	USINT	Number of the distributed I/O system triggering the interrupt
Event_Count	INT	• = n: Number of lost cycles • = -1: An unknown number of cycles has been lost (e.g., because cycle has changed).
Synchronous	BOOL	Reserved
CycleTime	UDINT	Display of the application cycle configured for the MC-Servo OB in µs

MC-PostServo OB (S7-1200)

Description

The organization block MC-PostServo [OB95] can be programmed and is called in the application cycle configured for the MC-Servo OB (Page 1368). MC-PostServo [OB95] is called directly after MC-Servo [OB91].

Via the organization block, you can read out the configured application cycle (information in µs).

Structure of the start information

Optimized start information:

Name	Data type	Meaning
Initial_Call	BOOL	=TRUE in the first call of this OB during transition from STOP to RUN
PIP_Input	BOOL	TRUE: The associated process image input is up-to-date
PIP_Output	BOOL	TRUE: The associated process image output was transferred to the outputs in good time after the last cycle
IO_System	USINT	Number of the distributed I/O system triggering the interrupt
Event_Count	INT	• = n: Number of lost cycles • = -1: An unknown number of cycles has been lost (e.g., because cycle has changed).
Synchronous	BOOL	Reserved
CycleTime	UDINT	Display of the application cycle configured for the MC-Servo OB in µs

MC-interpolator OB (S7-1200)

Description

When you create a technology object with drive connection using MC-Interpolator [OB91]/ create an analog output, the organization block PROFIdrive is created automatically. The Motion Control functionality of the technology objects creates its own execution level, and is called according to the Motion Control application cycle of the MC-Servo OB (Page 1368).

The MC-Interpolator OB is write-protected. The content cannot be changed.

The evaluation of the motion control instructions as well as the monitoring and the setpoint generation for all technology objects configured on the CPU (motion control) is performed within the MC-Interpolator OB.

The MC-Interpolator OB is called at the end of processing of the MC-Servo OB. The clock ratio between MC-Interpolator OB and MC-Servo OB is always 1:1.

Process response

You must select the application cycle of the MC-Servo OB long enough to allow all the technology objects for motion control to be processed in one application cycle. If the application cycle is not observed, overflows occur. Note the following response to overflows:

- The CPU tolerates a maximum of three consecutive overflows of the MC-Interpolator OB.
- The processing of an MC-Interpolator OB can be interrupted by the calling of the MC Servo OB, at most.

Block parameters of organization blocks (S7-1200)

Basics of block parameters (S7-1200)

Introduction

Several organization blocks (OBs) have properties with which you can control their behavior or their assignment to specific events. You can influence these properties by assigning parameters.

Overview

You can assign parameters to the properties for the following organization blocks:

- Time-of-day interrupt OBs
- Cyclic interrupt OBs
- Hardware interrupt OBs
- MC-Servo OB

See also

Assigning parameters to hardware interrupt OBs (Page 1373)

Assigning parameters to cyclic interrupt OBs (Page 1372)

Parameter assignment for time-of-day interrupt OBs (S7-1200)

Procedure for setting the parameters

To set the parameters of a time-of-day interrupt OB, proceed as follows:

1. Open the "Properties" dialog belonging to the time-of-day interrupt OB in question.
2. Click the "Time-of-day interrupt" group in the area navigation.

Overview of the parameters that can be set

You can set the following parameters:

- Execution
- Start date and time-of-day
- Option buttons "Local time" and "System time"

"Execution" parameter

Use the drop-down list "Execution" to define the periods at which the time-of-day interrupt OB is to be executed. The time intervals relate to the settings for "Start date" and "Time-of-day".

The following values are possible for "Execution":

- Never
- Once
- Every minute
- Hourly
- Daily
- Weekly
- Monthly
- Yearly
- End of month

Note

With the value "End of month", the value specified under "Start date" is irrelevant.

"Start date" and "Time-of-day" parameter

Here, you specify the time at which the time-of-day interrupt is to be executed for the first time.

Example: Start date = 07/05/2013, time =11:16

Depending on the value of the "Execution" parameter, the CPU generates additional time-of-day interrupts periodically. Depending on the setting, the start time relates either to the local time or to the coordinated universal timeUTC.

Note

If you set the "Execution" parameter to "Monthly", the start date cannot be set to the 29th, the 30th, or the 31st. If you want the time-of-day interrupt OB to start at the end of the month, you should set the parameter "Execution" to "End of month".

"Local time" or "System time"

Here, you decide which time the start time of the time-of-day interrupt OB relates to:

- "Local time": The start time relates to the time zone set for the CPU.
- "System time": The start time relates to the coordinated universal time UTC (Universal Time Coordinated).

Assigning parameters to cyclic interrupt OBs (S7-1200)

Introduction

You can use cyclic interrupt OBs to start programs at regular time intervals. To do so you must enter a scan time and a phase shift for each cyclic interrupt OB used.

You can use up to four cyclic interrupt OBs or time-delay OBs (OB numbers >= 200) in your program. If, for example, you are already using two time-delay interrupt OBs, you can insert a maximum of two further cyclic interrupt OBs in your program.

Note

If you assign multiple cyclic OBs, make sure that you assign a different cycle time or phase offset to each cyclic interrupt OB to avoid them executing at the same time or having to queue. When you create a cyclic interrupt OB, the cycle time 100 and the phase offset 0 are entered as start value.

Procedure

To enter a scan time and a phase shift for a cyclic interrupt OB, proceed as follows:

1. Open the "Program blocks" folder in the project tree.
2. Right-click on an existing cyclic interrupt OB.
3. Select the "Properties" command in the shortcut menu.
   The "<Name of the cyclic interrupt OB>" dialog box opens.
4. Click the "Cyclic interrupt" group in the area navigation.
   The text boxes for the scan time and the phase shift are displayed.
5. Enter the scan time and the phase shift.
6. Confirm your entries with "OK".

See also

Basics of block parameters (Page 1371)

Cyclic interrupt OBs (Page 1361)

Assigning parameters to hardware interrupt OBs (S7-1200)

Introduction

You must select the corresponding event and assign the following parameters for every input channel and high-speed counter that should trigger a hardware interrupt:

- Event name
- Number of the hardware interrupt OB that is assigned to this process event

The parameters of the hardware interrupt are assigned in the properties of the corresponding device. You can assign parameters for up to 50 hardware interrupt OBs.

You can create the hardware interrupt OB to be assigned parameters either before or during activation of an event.

Procedure

To configure a hardware interrupt event, follow these steps:

1. Double-click the "Devices & Networks" command in the project tree.
   The hardware and network editor opens in the network view.
2. Change to the device view.
3. If the Inspector window closed in the device view, select the "Inspector window" check box in the "View" menu.
   The Inspector window opens.
4. Click the "Properties" tab.
5. In the device view, select the module for which you want to a assign a hardware interrupt.
6. Select the corresponding event that will trigger a hardware interrupt, e.g., a positive edge.

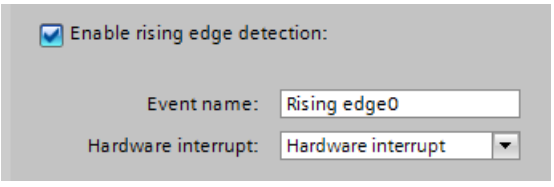

7. Enter an event name.
8. Select an existing hardware interrupt OB from the "Hardware interrupt" drop-down list or create a new hardware interrupt OB. If you have not previously created any hardware interrupt OBs, you can click the "Add new block" button in the drop-down list.
   The start information of the corresponding hardware interrupt OB, including all specifications for the interrupt-triggering event, is updated.

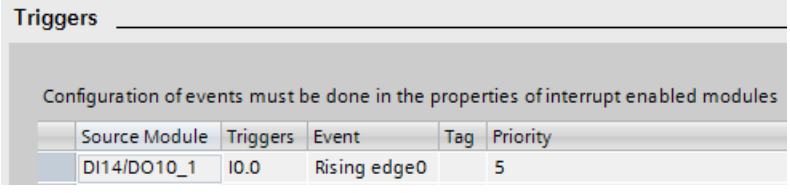

9. If you want to assign further hardware interrupts, repeat steps 5 to 8.

A system constant of data type Event_HwInt is created automatically for the event identified by the explicit event name. The system constants are displayed in the standard tag table.

See also

Basics of block parameters (Page 1371)

Hardware interrupt OBs (Page 1362)

Events and OBs (Page 1305)

Assigning the MC-Servo OB parameters (S7-1200)

Procedure for setting the parameters

To assign the parameters of an MC-Servo OB, follow these steps:

1. Open the "Properties" dialog associated with the MC-Servo OB in question.
2. Click on the "Cycle time " group in the area navigation.

Overview of the parameters that can be set

In the "Cycle time" group, select the application cycle with which the MC-Servo OB will be called cyclically in the execution system of the SIMATIC S7-1200

You set the cycle in which the MC-Servo OB is called in the "Application cycle" field.

See also

Basics of block parameters (Page 1371)

MC-servo OB (Page 1368)

Useful information on CPU firmware versions and STEP 7 versions (S7-1200)

CPUs and engineering software for configuring the CPUs is constantly developed further to improve performance and security. New versions that have some special features with respect to the interaction of the components are created in this way. The sections below describe the special features of the S7-1200 CPUs with firmware version V4 as compared to firmware versions V1 to V3.

For a detailed comparison of the range of functions, read the S7-1200 System Manual (description of new instructions, new organization blocks and advanced configuration options).

Required engineering software

S7-1200 CPUs V4 can be configured with STEP 7 as of V13.

Compatibility between memory card content and firmware version of the CPU

Memory cards (transfer cards or program cards) with configuration and program for an S7-1200 CPU V1, V2 or V3 do not work with an S7-1200 CPU V4.
Memory cards with configuration and program for an S7-1200 CPU V4 do not work with an S7-1200 CPU V1, V2 or V3.

You must change an S7-1200 CPU V1 to V3 configuration to an S7-1200 CPU V4 configuration (device replacement) and then download it to the CPU. A gradual device replacement is required for an S7-1200 CPU V1-V2 (see below).

When you insert the memory card into a CPU with an incompatible firmware version, the CPU does not start up. When you insert a memory card for a CPU V1, V2 or V3 into an S7-1200 CPU V4, this CPU outputs a version error.

Going online and loading

When you have configured an S7-1200 CPU with firmware version V1, V2 or V3 with STEP 7, the CPU to which you want to go online or which you want to download must have one of these firmware versions. You cannot go online to an S7-1200 CPU V4 with a configured S7-1200 CPU V1, V2 or V3.

On the other hand, you cannot go online to an S7-1200 CPU V1, V2 or V3 with a configured S7-1200 CPU V4 or download this CPU.

Replacing an existing CPU

You can replace a configured S7-1200-CPU V1, V2 or V3 with a new CPU with a firmware version greater than or equal to 4. In case of an existing S7-1200 CPU V1 or V2, you must start by replacing the device with an S7-1200 CPU V3: It is not possible to directly replace it with an S7-1200 CPU V4.

1. S7-1200 CPU V1 (V2) > S7-1200 CPU V3
2. S7-1200 CPU V3 > S7-1200 CPU V4

As long as you do not download the configuration, you can undo the device replacement ("Undo" command in the "Edit" menu). When you download the configuration for the new CPU firmware version ("V4 configuration") to the CPU, you can no longer return to version V3.

This means you should save the existing project, for example, with a V3 configuration as project archive so that you can access it later.

Special notes for device replacement (V3 > V4):

- The interrupt behavior of interrupt OBs remains the same; they are not configured as interruptible. This is also the default behavior of S7-1200 CPUs V1-V3. The interrupt behavior of the interrupt OB can be configured for S7-1200 V4 CPUs. If you drag an S7-1200 CPU V4 directly from the hardware catalog into the network view, this option is activated (interrupt OBs are interruptible).
- The behavior for PUT/GET access of remote partners remains the same; access is permitted. This is also the default behavior of S7-1200 CPUs V1-V3. The access via PUT/GET communication by means of remote partners can be configured for S7-1200 V4 CPUs ("Protection" area of CPU parameters). If you drag an S7-1200 CPU V4 directly from the hardware catalog into the network view, access is not permitted and must be explicitly enabled.
- The wording for the protection levels changes but the effect of the settings remains the same. You can also select the access level "No access (complete protection)".
- The web server settings for activation of the web server and HTTP/HTTPS settings are applied. You also have the option to create users and assign them specific rights (Web server area > User management of CPU parameters). Web server users only have access to standard websites when you do not configure users. An S7-1200 CPU V4 no longer supports the user "admin" and his/her password.
- For reasons of compatibility, the interfaces of used OBs are not changed by the device replacement. You can therefore continue to use used OBs. If you want to use OB properties that have been added or changed in the new firmware version of the S7-1200 CPU, you will need to recreate the corresponding OB.

Communication with HMI devices

When you connect an HMI device to an S7-1200 CPU V4, make sure that you use the appropriate Runtime software version of the HMI device.

You may have to transfer the latest HMI Runtime version by means of the WinCC engineering software.

You must compile the HMI configuration once again and download it to the HMI device for the CPU-HMI communication to work.

See also

What you need to know about SIMATIC memory cards (Page 1296)

1.4.2.2 Addressing modules

Addressing modules

Introduction

In the device overview, you see the addresses or address ranges of the modules in the I address and Q address columns. There are other addresses as well, which are explained below.

I/O address

I/O addresses (input/output addresses) are required to read inputs and set outputs in the user program.

Input and output addresses are assigned automatically when modules are inserted in the rack. The address of the first channel is the start address of a module. The addresses of the other channels are derived from this start address. The address end is obtained from the module-specific address length.

Device address (e.g., Ethernet address)

Device addresses are addresses of programmable modules (Industrial Ethernet addresses). They are required to address the different stations of a subnet, for example, to download a user program to a CPU.

Hardware identifier for identifying modules and submodules

In addition to the I addresses and Q addresses, a hardware identifier (HW ID) is assigned automatically and is used to address and identify the module. Submodules (units of a module), such as an integrated counter, also receive such a hardware identifier.

The hardware identifier consists of an integer and is output by the system with diagnostics alarms to allow the faulty module or the faulty submodule to be localized.

In addition, the hardware identifier is used for a number of instructions to address the corresponding module.

The hardware identifier cannot be changed.

Example: Identifying high-speed counters of the S7-1200 CPU

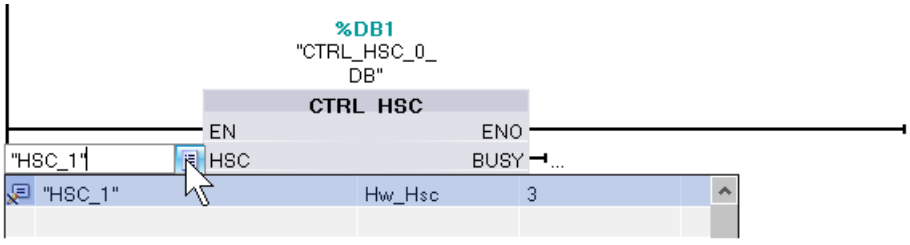

The hardware identifier is assigned automatically when components are inserted in the device or network view and in the PLC tags. A name is also assigned automatically for the hardware identifier. The system constants of the PLC tags cannot be changed either.

See also

Specifying input and output addresses (Page 1378)

Assigning addresses to a location in the program (Page 1379)

Introduction to loading a configuration (Page 3030)

Inspector window (Page 23)

Specifying input and output addresses

Default input and output addresses are set automatically. You can, however, change the address assignment later.

All addresses of modules are located in the process image area. The process image is automatically updated cyclically.

Requirement

You are in the device view.

Procedure

To change the preset address range proceed as follows:

1. In the device view, click on the module for which you want to set the start address.
2. Go to "I/O addresses" in "Properties" in the inspector window.
3. Under "Start address" enter the required start address.
4. Press <Return> or click on any object to accept a modified value.

If you have entered an invalid address, a message indicating the next available address is displayed.

Note

You can also change the addresses directly in the device overview.

See also

Editing properties and parameters (Page 61)

Input and output addresses in the address overview (Page 62)

Assigning addresses to a location in the program

You can assign addresses of the I/O channels of modules directly to the points of use in the program or to a tag table.

Requirement

- The device view of the hardware and network editor is open.
- The zoom level in the device view must be set to at least 200% to allow you to see the individual I/O channels.
- The instruction window of the programming editor or a tag table is open.

Procedure

To assign I/O channels of modules to the points of use in the program or to a tag table, follow these steps:

1. In the device view, navigate to the module with the desired I/O channel.
2. Click and hold down the mouse button to drag the desired I/O address to the corresponding point of use of the block or to the tag table.

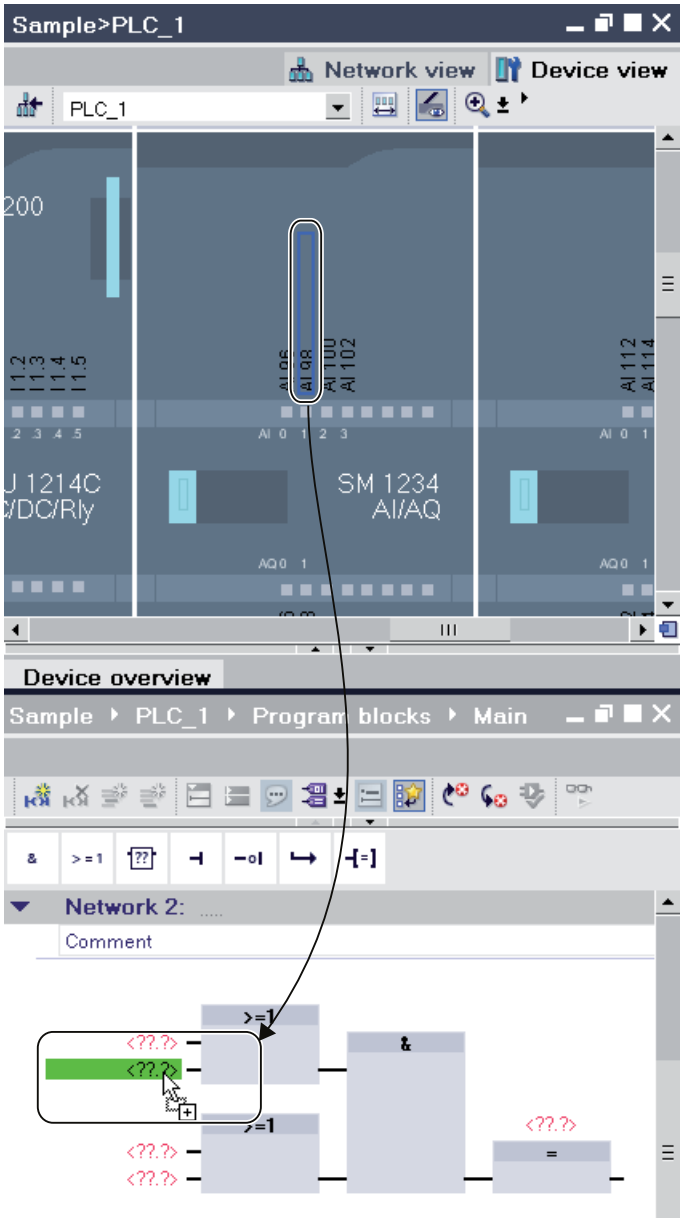

The address of the module is assigned to the point of use in the program or entered as a tag in the tag table.

Note

The tag for an input or output of a block can also be dragged to the input or output of a module in order to link the tag to the I/O channel of the module.

1.4.2.3 Signal board

Inserting a signal board in a CPU

Introduction

Signal boards allows you to increase the number of the S7-1200 CPU's own inputs and outputs. Just like all other hardware components, you will find signal boards in the hardware catalog. However, you do not insert signal boards in the rack like other modules but directly in a slot of the CPU itself.

Note the following points when using a signal board:

- Each CPU can have only one signal board inserted in it.
- A signal board can only be inserted when the slot in the CPU is free.

There are various ways of inserting a signal board in a CPU:

- Double click on a signal board in the hardware catalog when there is a free slot in the CPU
- Drag from the hardware catalog to a free slot in the CPU
- Shortcut menu of a signal board in the hardware catalog for copying and pasting

Requirement

- The hardware catalog is open.
- The S7-1200 CPU has a free slot for the signal board.

Inserting a signal board in a CPU

To insert a signal board in a CPU, proceed as follows:

1. Go to the required signal board in the hardware catalog.
2. Select the signal board you want to use.
3. Drag the signal board to the free slot in the CPU.

You have now inserted a signal board in the slot of the CPU.

If you are in the network view, you can also drag a signal board to a device. If the CPU has a an empty slot for a signal board, the signal board is inserted automatically into this slot.

1.4.2.4 Configurations for Web server

Information about the web server

Introduction

The Web server allows you to monitor the CPU via the Internet or the intranet of your company. This permits evaluation and diagnostics over long distances.

Alarms and status information are visualized on HTML pages.

Web browser

You need a web browser to access the HTML pages of the CPU.

The information with which Web browsers and Web browser versions the Web server was tested is available in the corresponding manual (http://support.automation.siemens.com/WW/view/en/36932465).

Reading information via the Web server

The following information can be read from the CPU. The availability of the respective web pages depends on the CPU and its firmware version.

As of firmware version 4 the web pages are available in several languages.

Page/information	Description
Introduction	Entry page for the standard web pages
Start page with general CPU information	The start page provides an overview of general information on the CPU, the name of the CPU, the type of CPU and basic information on the current operating state.
Diagnostics	Display of the detailed information about the CPU including the serial number, article number and version numbers. Displays the content of the diagnostics buffer with the most recent entries first.
Module information	Display of the identification information whether the centrally inserted components of a station are OK, whether there are maintenance requirements or components cannot be reached, for example. As of firmware version 4 a firmware update is possible via this Web page.
Communication	Displays the communication connections during open communication (OUC); displays resources and address parameters.
Diagnostic buffer	Display of the diagnostic buffer content
Tags	Displays the status of operands of the user program to monitor and change the values.
File browser (as of firmware version 4)	Data logs in CSV format to transfer to the hard disk of the programming device. The data logs are created with data log instructions in the user program and filled with data. As of firmware version 4 you have access to files of the internal load memory and of the external load memory (Memory Card) via this web page, for example, to the content of the directories "DataLogs" and "Recipes".
Login	Login as a user or logoff.
User pages (if user-defined web pages have been configured and loaded)	The user web pages deliver a list of web pages with customer-specific web applications.

Web access to the CPU via PG/PC

Proceed as follows to access the Web server:

1. Connect the client (PG/PC) to the CPU via the PROFINET interface.
2. Open the web browser.
   Enter the IP address of the CPU in the "Address" field of the Web browser in the format http://ww.xx.yy.zz (example: http://192.168.3.141).
   The start page of the CPU is opened. From the start page, you can navigate to further information.

Standard web pages

Requirements for web access

The requirements for access to standard CPU web pages are explained in the following, as well as the effects of missing or existing configuration information.

Requirement

The web server must be started.

The web server only starts when it has been activated in the properties of the CPU in the "Web server" section.

Note the following:

The web pages are normally transmitted via an non-secure connection and are not secured against hacker attacks. If you want to transfer the web pages in encrypted form to the browser, use the URL https://, followed by the IP address of the CPU.

Logon

No logon is required to access the standard web pages read-only. A user must be logged on to execute certain actions like changing the operating mode of the CPU or for write access.

For S7-1200 CPUs up to FW version V3:

You must be logged on as "admin" for the actions listed above. The logon input boxes are on the top left of each standard web page.

If you log on as "admin", you must enter the user name and password there.

Name: admin.

Password: configured CPU password (for password-protected CPU).

For S7-1200 CPUs as of FW version 4:

You can freely select the names of users and the passwords (CPU parameter "Web server", area "User management").

You assign rights to the users, for example the right to query diagnostics or to update the firmware.

JavaScript and cookies

The standard web pages use JavaScript and cookies. You must enable both in your web browser.

If JavaScript is not enabled, the following limitations apply:

- Data from standard web pages is not automatically updated.
- You cannot log on as user.
- Fields cannot be sorted (module information)

If cookies are not enabled, you cannot log on.

See also

Access for HTTPS (Page 1386)

Settings for operation

Settings for operation

To be able to use the web server of an S7-1200-CPU, you must select the CPU in the network view or the device view and make the following settings in the inspector window under "Properties > General > Web server":

- Enable the web server
- Restricting access to the CPU to HTTPS transmission protocol (encrypted transmission) Access via port 80 is then blocked. Communication is only possible via port 443.

- Enabling automatic update of web pages
  The update interval is set by default and cannot be changed. The CPU updates web pages with changing content ((for example, status information or diagnostics information) at regular intervals.
- Creating and managing users
  Users are exclusively entitled to options that are assigned to the access rights. Depending on the CPU and firmware used, you can assign different user rights. Rights that your CPU does not support cannot be activated. A user called "Everybody" with minimal access rights, that you can, however, extend, is set by default in the user list. A user who uses the Web server without entering a password has "Everybody" user rights.
  You have the possibility of configuring further users with different access rights. These users have to log on with the configured user name and password.

WARNING

Unauthorized access to the CPU by means of the Web server

Unauthorized access to the CPU or the changing of PLC variables to invalid values can result in interruptions of the processes controlled by the CPU and cause death, serious bodily harm or material damage!

Since the activation of the Web server allows authorized persons for example to change operating modes, to write-access CPU data or to update the firmware, we recommend that the following security measures be taken:

- If possible limit access to the HTTPS protocol.
- Create users with secure passwords. An example of a secure password is one which is only used for a single application, is more than 8 characters long, and consists of lower- and upper-case letters as well as symbols and numbers (?!+%$1234...). In addition, passwords based on common keyboard sequences or words from the dictionary should be avoided.
  Change the password at regular intervals.
- Do not extend the rights of the "Everybody" user.
- Check the PLC variables in the user program and limit the range of values to permissible ranges since users can set invalid values via the Web server.

Access for HTTPS

Access via HTTPS

HTTPS is used for encrypting and authentication of communication between the browser and web server.

To transfer data between the browser and the CPU using the HTTPS protocol, enter the URL as https://ww.xx.yy.zz in the address line of your browser, whereby ww.xx.yy.zz stands for the IP address of the CPU.

You require a valid, installed certificate for error-free HTTPS access to the CPU.

If no certificate is installed a warning is displayed with a recommendation not to use this page. To view the page you must explicitly "Add exception".

You can receive a valid certificate (Certification Authority) "SIMATIC CONTROLLER" as a download from the "Intro" web page under "Download certificate". The help function for your respective web browser provides information on how to install a certificate.

Accessing data of the CPU memory

You can access data in the internal or external CPU load memory by means of a standard web page.

- Use the web page "Data logs" for S7-1200 CPUs up to and including FW version 3. From this web page, you transfer the data logs from the CPU to a drive on your PC.
- Use the web page "File Browser" for S7-1200 CPUs as of FW version 4. From this web page, you transfer data from the folders "Data logs" or "Recipes", for example, to a drive on your PC.

Depending on the file type and the access permissions you have configured for the web server user, you can download, delete, rename or upload the files. The actual directories can only be created, deleted or renamed.

Example: Data logs

To open a data log, click on the link of the desired data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

Special note: Data logs are saved in U.S. American CSV format. You can only open the file directly using the U.S. version of Microsoft Excel. If you are using another national version of Microsoft Excel, you must import the file, selecting "comma" in the import assistant as the delimiter.

Downloading a data log

To download a data log, click on the download icon of the desired data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

Downloading and clearing or deleting a data log

For a CPU with FW version up to V3.0:

To download and delete the current entries of the data log, you must be logged on. To do this, click on the "Download and delete" icon of the required data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

For a CPU as of FW version V4.0:

To reset the data log, follow these steps:

1. Open the CSV file, for example with Excel.
2. Delete the rows between the headline and the row with the entry "//END", if this row exists.

3. Save the file to a drive on your PC.
4. On the web page "File Browser", delete the data log (which means the CSV file) and load the prepared CSV file to the CPU with the "Upload file" button of the "File Browser" web page.

Additional information is available in the S7-1500 CPU system manual.

Creating and loading user pages

What you need to know about user pages

Concept

The concept of user pages or of user-defined web pages allows you to access freely-designed web pages of the CPU with a web browser. The Web server of the CPU provides this function.

You are not dependent on special tools for the design and functionality of the user-defined web pages. You can adapt the pages in the layout with CSS, provide dynamic content with JavaScript or use any framework to produce web pages. The totality of files processed by the Web server is also referred to as the "web application".

Web application and user program

Using HTML code in user-defined web pages (AWP commands), you can also transmit data via a web browser to the user program of the CPU for further processing and can display data from the operand area of the CPU in the web browser.

You can use script instructions (such as Javascript) to optimize your web pages, for example to dynamically change contents or validate user entries.

To synchronize between the user program and the Web server, but also to initialize, you must call the WWW (SFC 99) instruction in the user program.

- If no interaction is required between the web application and the user program, for example, if a web page only provides static information, only initialization in the user program is required.
- If a simple data exchange is necessary between PLC tags and tags in web applications, to display the contents of PLC tags or write a value in a PLC tag for example, the syntax for reading and writing tags has to be observed. In this case only an initialization is required in the user program, for example in the startup OB.
- If a further interaction is required between the web application and the user program, you must handle status and control information from the Web Control DB in addition to the synchronization between Web server and user program. This is the case, for example, when user entries are transmitted via the web browser to the Web server for evaluation by the CPU. Unlike simple data exchange, the user program directly influences the time at which the requested web page is relayed back to the web browser. In this case, you must be acquainted with the concept of manual fragments and the structure of the Web Control DB.

Integration of the HTML sources via the TIA Portal

The parameters for an integration of the HTML sources in the TIA Portal are located in the properties of the respective CPU with Web server:

Parameter	Meaning
HTML directory	Directory of the HTML sources for the web applications. You can input the path directly or navigate to the directory by using the button next to the entry field. It is advantageous if you integrate your Web pages with relative paths. To do this you have to insert the directory in which you store the HTML sources for the Web application in the STEP 7 project directory, for example, in a new directory "Webpages". You then specify the relative path name as the HTML directory; in the example ".\Webpages". Advantages of using relative paths: • The user-defined Web pages are archived together with the STEP 7 project. • The Web pages are copied to the new path with the menu command "Project > Save as...". • When copying the project to another path using the Windows Explorer, it is no longer necessary to adapt the HTML directory path to generate new Web DBs. • Several control systems in a project that use the same HTML pages also have the same HTML directory path information.
Start HTML page	Path to the Start HTML page. You can input the path directly or navigate to the HTML page with the button next to the entry field. The start HTML page is the HTML page to be opened at the start of the web application.
Application name	Optional name for the application. This name is used to further divide or group the web pages. When an application name already exists, the URL is displayed in the following format:"http://a.b.c.d/awp/<Application name>/<Page name>.html"

Initialization

User-defined web pages are "packaged" in data blocks for processing by the CPU. You must generate appropriate data blocks from the source data (HTML files, images, JavaScript files, etc.) during configuration to be able to download the web application into the CPU. The Web Control DB takes on a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. Data blocks that contain coded web pages are termed "Fragment DBs".

When the data block is downloaded into the CPU, the CPU does not "know" that user-defined web pages are coded inside it. The "WWW" (SFC 99) instruction, for example, in the Startup OB informs the CPU which DB is the Web Control DB. The user-defined web pages can be accessed via a web browser after this initialization.

Synchronization

If the user program is to exchange data with the user-defined web pages, the WWW (SFC 99) instruction must be used in the cyclic program section.

Examples of interaction between user program and web page:

- Check received data
- Assemble and send back data to the web browser making the request

In this case, the status information must be able to be evaluated and control information must be transmitted to the Web server, for example, to release a requested web page.

Procedural overview

Basic information

This section provides a step-by-step explanation of the basic procedure used to create and download custom web pages and to use them in the operating phase.

The following graphic provides a simplified representation of the process used in creating and displaying custom web pages:

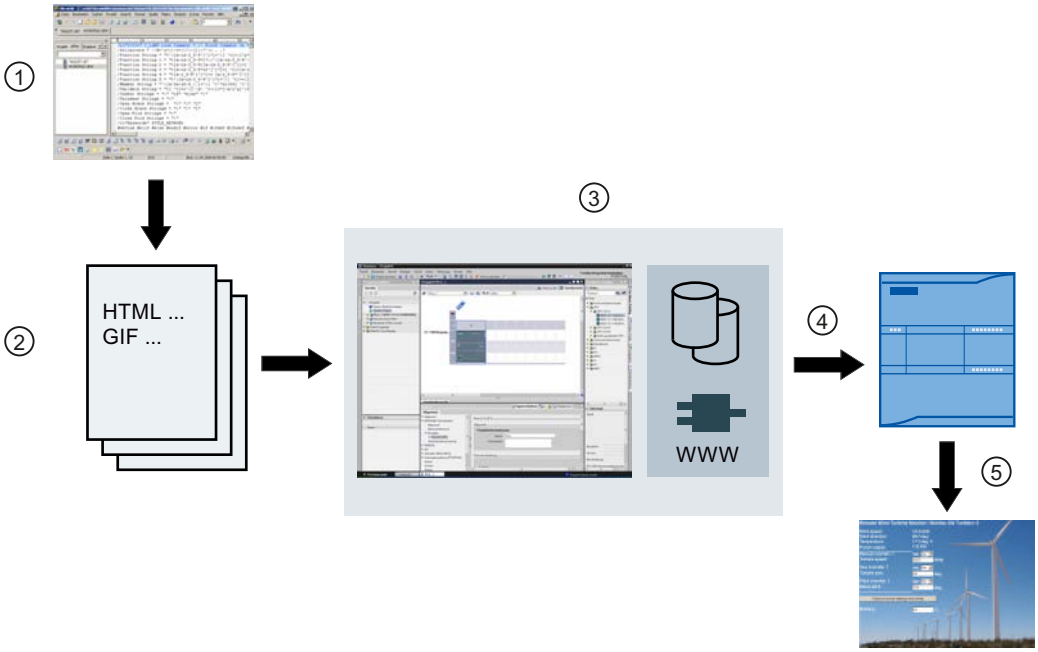

① Programming a web application (using suitable tools when required and AWP commands for dynamic pages when applicable).

② The web application is comprised of single source files, for example, *.html, *.gif, *.js, etc.

③ Using STEP 7:

- Generate the data blocks (Web Control DB and fragment DBs) from source files. The DBs contain meta information and the complete web application, including the images and the dynamic and static parts of the web application. The DBs are stored under "System blocks" in the project tree.
- Call the "WWW" instruction in the user program. This instruction initializes the web server of the CPU for a web application.
- If required, complete final programming for interaction between the web server and user program

④ Downloading the blocks to the CPU.

⑤ Call the web page in the browser. The web pages of the CPU are called by entering the IP address of the CPU.

Additional information

You can find additional information and examples relating to the S7-1200 web server on the Internet (http://support.automation.siemens.com/WW/view/en/36932465).

Creating web pages

Web design tools from various companies can be used to create user-defined web pages. As a rule, the web pages should be programmed and designed compliant to the conventions of the W3C (World Wide Web Consortium). No check is made for compliance to W3C criteria in the web server of the CPU.

Rules

- The tool must be able to directly edit the HTML code so that the AWP command can be inserted into the HTML page.
  Only the AWP commands are parsed in the CPU and, for example, replaced by values from the user program/process image of the CPU.
- Files containing AWP commands must be coded in UTF-8. In the metadata of the HTML page, therefore, set the attribute charset to UTF-8 and save the file UTF-8 coded.
- Files containing AWP commands must not contain the following sequence:]]
- Files containing AWP commands must not contain the following sequence outside of the "Tag read ranges" (:=<Tag name>:): :=
  Tip: Replace the first character of a prohibited sequence with its character coding; for the colon, for example, :.

A small example for a custom web page should make clear the basic design.

Requirement

- The CPU must have a web server and the web server of the CPU must be activated.
- To be able to access PLC tags with write access as a user, you must be logged on as "admin".
- For the example below, PLC tags must be defined for those PLC tags that are to be shown on the web page. This is shown here for the first tab used, "Tank_below_max".

		Name	Data type	Address
1		Tank_below_max	Bool	%I0.0

Creating user-defined web pages

The following code for an example web page reads values from the process image and provides them in a table.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/
TR/html4/strict.dtd">
<html>
  <head>
   <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
```

```
   <title>Mix</title>
  </head>
   <body>
    <h1>Mix</h1>
    <h2> Actual State </h2>
    <table border="1">
     <tr>
      <th>Variable</th>
      <th>State</th>
     </tr>
     <tr>
      <td>Tank below max</td>
      <td>:="Tank_below_max":</td>
     </tr>
     <tr>
      <td>Tank above min</td>
      <td>:="Tank_above_min":</td>
     </tr>
    </table>
   </body>
</html>
```

AWP commands for controlling Web pages

Using AWP commands in Web pages

With AWP (Automation Web Programming) commands, you declare the interface between your user pages (Web application, e.g. a simple HTML page) and the CPU data.

To develop user pages or Web applications, you need only observe the restrictions of the Web browser. In one of the programming languages of STEP 7, control with the user program in the CPU which CPU data is displayed at what time in the Web browser of the viewer.

Use AWP commands, which you comment within the HTML page, to declare data to be used for intentional interaction between the Web application and the user program.

You insert AWP commands as HTML comments with a special syntax into HTML pages. Using AWP commands, the following features can be implemented:

- Read PLC tags
- Write PLC tag
- Read special tags
- Write special tags
- Define enum types
- Assign tags to enum types
- Defining fragments
- Import fragments

Syntax of AWP commands

An AWP command begins with "`<!-- AWP_`" and ends with "` -->`". In JavaScript files, the commands should also be enclosed by JavaScript comments ("`/*...*/`").

Notation rules for PLC tag names within an AWP command

The AWP commands "AWP_In_Variable" and "AWP_Out_Variable" contain a name attribute and optionally a use attribute. A PLC tag name is assigned to these attributes, by means of which the PLC tags in the browser are written or read. The following rules apply to handling PLC tag names in HTML code:

- PLC tags must be enclosed in quotation marks (" ... ").
- PLC tags used in AWP commands must also be enclosed by single quotation marks ('" ... "') or with quotation marks masked by a backslash ("\" ... \"").
- If the PLC tag name contains the character \ (backslash) or * (star), this character must be designated with the escape sequence \\ as standard character of the PLC tag name. See below for examples.
- If the PLC tag name in the AWP command is also enclosed by single quotation marks and the single quotation mark (') occurs within the name, it must also be designed as normal character by the escape sequence \'.
- If an absolute address (input, output, bit memory) is used in AWP command, it is enclosed by single quotation marks.

PLC tag	PLC tag in HTML code
"Velocity"	<!-- AWP_In_Variable Name='"Velocity"' -->
	<!-- AWP_In_Variable Name="\"Velocity\"" -->
"abc\de"	<!-- AWP_In_Variable Name='"abc\\de"' -->
"abc'de"	<!-- AWP_In_Variable Name='"abc\'de"' -->
"abc'de"	<!-- AWP_In_Variable Name="abcde" Use'"abc\'de"' -->
"DB name".tag	<!-- AWP_In_Variable Name='"DB name".tag' -->
"Plc1".Data[1].typeDataS-truct.value	<!-- AWP_In_Variable Name='"Plc1".Data[1].typeDataStruct.value' -->
-	<!-- AWP_Out_Variable Name=’flag1’ Use='M0.0' -->

See also

Reading tags (Page 1393)

Writing tags (Page 1395)

Special tags (Page 1396)

Reading tags

User-defined web pages can read PLC tags.

The PLC tag must be specified by a PLC tag name.

These OUT variables (direction of output as viewed from the controller) are inserted at any location within the HTML text with the syntax described in the following.

Syntax

```
:=<varname>:
```

These references are replaced when the Web server is in operation by the current values of the PLC tag in each case.

`<varname>` can be a simple, global PLC tag but also a complete tag path to a structural element.

Notation rules for PLC tag names

- PLC tags in HTML code are enclosed by quotation marks ("), if they are defined in the tag table. In the case of data block tags, the name of the data block is enclosed by quotation marks. If special characters are used in the structure elements of the data block, for example the dot (.) or blank, this part must also be enclosed by quotation marks.
- Quotation marks are not used for absolute addresses of inputs, outputs or bit memories.

PLC tag	PLC tag in HTML code
"DB_name".var_name	:="DB_name".var_name:
"DB_name"."var.name"	:="DB_name"."var.name":
"memory"	:="memory":
-	:=I0.0:
	:=Q0.0:
	:=MW100:
	:=%MW100:
"My_Data_Block".flag1	<!-- AWP_Out_Variable Name='flag1' Use='"My_Data_Block".flag1' -->
	...
	:=flag1:

- If the PLC tag name contains the character : (colon) or \ (backslash), this character must be designated with the escape sequence \: or \\ as standard character of the PLC tag name.

PLC tag	PLC tag in HTML code
"abc:de"	:="abc\:de":
"abc\de"	:="abc\\de":

- Special characters "<, &, >"
  Display problems can occur if these characters are contained in the tag name (for example, "a<b").
  Avoid expressions such as :="a<b": in the HTML page.
  To prevent display problems from occurring, use e.g. an AWP command with a use expression according to the pattern depicted below. The use attribute defines the PLC tag with the problematic character, the name attribute defines the name without problematic character, as it is used in the HTML page.

PLC tag	PLC tag in HTML code
"a<b"	<!-- AWP_Out_Variable Name='simplename' Use='"a<b"' --> ... :=simplename:

See also

Using AWP commands in Web pages (Page 1392)

Writing tags

Custom web pages can write data into the CPU.

This requires an AWP command that identifies the PLC tag to be written.

The PLC tag must also be specified by a PLC tag name.

The IN tags (direction of input as viewed from the controller) are placed on the browser page. This can be done, for example, in a form.

The tags are either set in the HTTP header (by cookie or POST method) or in the URL (GET method) by the browser and are then written by the Web server into the respective PLC tag.

Syntax

To allow the IN tags to be written to the CPU, the tags must first be defined by an explicit AWP instruction:

```
<!-- AWP_In_Variable Name='<PLC_Varname1>' Name='<PLC_Varname2>'
Name='<PLC_Varname3>' -->
```

Several tags can be defined in an instruction - such as that shown above.

The specific PLC tag name is hereby written in double quotation marks; for example <PLC_Varname1> = "myVar".

In cases where the name of the tag that you use for the web application is not identical to the name of the PLC tag, the "Use" parameter can be used to assign to a PLC tag:

```
<!-- AWP_In_Variable Name=’<Webapp_Varname>’ Use=’<PLC_Varname>’
```

Example

The "AWP_In_Variable" AWP command is indispensable when handling forms.

```
<form method='post' action='/awp/appl/x.html'>
 <p>
  <input name='"var1"' type='text'>
  <input value='set' name='Button1' type='submit'>
 </p>
</form>
```

In the form defined above, the HTTP request method "post" is used to transfer the tag "var1" to the Web server. The user places the "var1" tag in the form field. The tag 'Button1' has the

value 'set', but is not required for the CPU. To allow the "var1" tag to be written to the CPU, the following instruction must be included in the same fragment:

```
<!-- AWP_In_Variable Name='"var1"' -->
```

Since PLC tags are enclosed in double quotation marks ("), the name in the AWP command must be enclosed in single quotation marks (') or in masked quotation marks (\"). To avoid the numerous escape sequences, we recommend the use of single quotation marks.

```
<!-- AWP_In_Variable Name=’"Info".par1’ -->
<!-- AWP_In_Variable Name="\"Info".par1\"" -->
```

Conditions for write access during operation

The following requirement has to be met in order for a user to be able to write to PLC tags from a user-defined web page:

The user must have rights to change tags. The Web server ignores the commands if the user has no change rights.

This rules applies to all writing access to web pages on a CPU.

See also

Requirements for web access (Page 1384)

Using AWP commands in Web pages (Page 1392)

Special tags

Special tags are mainly HTTP tags set in the definition of the World Wide Web Consortium (W3C) . Special tags are also used for cookies and server tags.

The AWP command to read and write special tags differ only in that they have additional parameters than the AWP command used to read and write normal tags.

Reading a special tag

The Web server can read PLC tags and transfer these to special tags in the HTTP Response Header. You can, for example, read a URL for a diversion to another web page and transfer to the special tag HEADER:Location using the special tag HEADER:Location.

The following special tags can be read:

Name	Description
COOKIE_VALUE:name	Value of cookie with name: "name"
COOKIE_EXPIRES:name	Execution time of cookie with name: "name" in seconds (must be set beforehand).
HEADER:Status	HTTP status code (if no other value has been set, status code 302 is returned).
HEADER:Location	Path for forwarding to another page. Status code 302 must be set.

Name	Description
HEADER:Retry-After	Anticipated time in which the service is not available. Status code 503 must be set.
HEADER: …	All other header tags can also be forwarded in this way.

Use the AWP command "AWP_Out_Variable" to specify which PLC tags are to be transferred in the HTTP header to the web browser.

Basic structure:

```
<!-- AWP_Out_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description

- Name: Type and name of special tag
- Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, parameter "Use" can be used to assign to a PLC tag.

Example:

```
<!-- AWP_Out_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

Writing a special tag

In principle, all HTTP tags written in the HTTP header by the web browser can be evaluated by the user program of the CPU. Examples of tag types:

Name	Description
HEADER:Accept-Language	Accepted or preferred language
HEADER:Authorization	Proof of authorization for a requested resource
HEADER:Host	Host and port of the requested resource
HEADER:User-Agent	Information on the browser
HEADER: …	All other header tags can also be forwarded in this way
SERVER:current_user_id	Indicates whether a user is logged on (current_user_id=0: no user logged on)
SERVER:current_user_name	User name of the user logged on
SERVER:GET	Request method is GET
SERVER:POST	Request method is POST
COOKIE_VALUE:name	Value of cookie with name: "name"

The AWP command "AWP_In_Variable" is used to define which special tags are to be evaluated in the user program of the CPU.

Basic structure:

```
<!-- AWP_In_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description:

Name: Type and name of special tag

Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, the parameter Use can be used to assign to a PLC tag.

Examples:

```
<!-- AWP_In_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use . The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='COOKIE_VALUE:siemens' Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use. The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='"COOKIE_VALUE:siemens"' -->
```

The HTTP-header variable is written in the same-name PLC variable.

See also

Using AWP commands in Web pages (Page 1392)

Enumeration types

Enumeration types (enums)

Numerical values from the PLC program can be converted into text and vice versa using enums. The numerical values can also be assigned for several languages.

Creating enums

Enter an AWP command using the following syntax at the start of the HTML file:

```
<!-- AWP_Enum_Def Name="<Name of the enum type>"
Values='0:"<Text_1>", 1:"<Text_2>", ... , x:"<Text_x>"' -->
```

For example, for German values to be saved as an HTML file in the "de" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"an", 1:"aus", 2:"Störung"' -->
```

For example, for English values, to be saved as an HTML file in the "en" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"on", 1:"off", 2:"error"' -->
```

Assigning enums

Tags are assigned from the user program to the individual enum texts using a special AWP command:

```
<!-- AWP_Enum_Ref Name="<VarName>" Enum="<EnumTypeName>" -->
```

<VarName> is thereby the symbolic name from the user program and <EnumTypeName> is the previously set name of the enum type.

Note

In each fragment in which enum texts are referenced by a PLC tag, this PLC tag must be assigned by the appropriate AWP command of the enum type name.

Ensure that no AWP command for importing fragments is positioned between an enum assignment and enum usage because this import can result in the enum assignment lying in a different fragment than the enum usage.

Example

Enum type "state" is defined with values "0" and "1". "0" means "off", "1" means "on":

```
<!-- AWP_Enum_Def Name="state" Values='0:"off", 1:"on"' -->
```

The following code is contained in the HTML code of the web page to be output:

```
<!-- AWP_Enum_Ref Name="operating state" Enum="state" -->
:=operating state:
```

Depending on the value of the "operating state" tag, the ´result displayed is no longer "0" or "1", but "off" or "on".

Simplified use of enumeration types

At S7-1200 CPUs as of firmware version 4 it is possible to use enumerations directly in AWP commands to read and write PLC variables.

You create enums as described in the previous section, and you can then utilize the values with user program read and write commands.

Creating enums

```
<!-- AWP_Enum_Def Name="<Name des Enum Typs>" Values='0:"<Text_1>",
1:"<Text_2>", ... , x:"<Text_x>"' -->
```

Utilizing enums in the user program read and write commands

```
<!-- AWP_In_Variable Name='<Varname>' Enum="<EnumType>" -->
<!-- AWP_Out_Variable Name='<Varname>' Enum="<EnumType>" -->
```

Example of reading PLC tags

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' --><!-- AWP_Out_Variable Name='"Alarm"'
Enum="AlarmEnum" -->...<p>The current value of "Alarm"
is :="Alarm":</p>
```

If the value of "Alarm" in the CPU is "2", the following text will be displayed on the HTML page:

'The current value of "Alarm" is Tank is empty' because the enum definition assigns the string "Tank is empty" to the numerical value 2.

Example of writing PLC tags

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' --><!-- AWP_In_Variable Name='"Alarm"'
Enum='AlarmEnum' -->...
<form method="POST">
<p><input type="hidden" name='"Alarm"' value="Tank is full" /></p>
<p><input type="submit" value='Set Tank is full' /></p>
</form>
```

Because the enum definition assigns the string "Tank is full" to the numerical value "1", the value "1" is written to the PLC tag "Alarm".

Definition of fragments

Fragments

Fragments are "logical sections" of a web page to be processed by the CPU individually.

Fragments are usually complete pages but can also be individual elements such as files (for example, images) or complete documents.

Defining fragments

```
<!-- AWP_Start_Fragment Name="<Name>" [Type="<Type>"] [ID="<Id>"]
[Mode=<Mode>]-->
```

The start of a fragment is specified by this command. A fragment runs to the start of the next fragment or to the end of the file.

- <Name> Indicates the name of the fragment.
  The name must start with a letter [a-zA-Z] or an underscore (_). Letters, underscores or numbers [0-9] can follow after this first character.
- <Type> Indicates the type of the fragment.
  - "manual" The user program is informed of the request for a fragment; the web page to be returned can be changed by the user program.
  - "automatic" The page is automatically processed (default).
- <id> A numeric ID can be stipulated for the fragment. If no ID is assigned, the fragment is automatically assigned an ID. For manual pages (<Type>=manual) , the fragment can be addressed in the user program of the CPU by this ID.

  Note

  Keep the ID low because the highest ID influences the size of the Web Control DB.

- <Mode> Fragments support the visible and hidden modes.
  - "visible" The fragment is a part of the web page. This mode is preset and can also be omitted.
  - "hidden" The fragment is not part of the web page. However, the fragment will be saved in the Web DB and is available to the user program for inserting in a requested web page. You use an exchange of the fragment ID (Web-Control-DB.fragment_index tag) to insert a "hidden" fragment in the requested web page.

The input document is completely divided into fragments by the "AWP_Start_Fragment" command. "AWP_End_Fragment" is therefore unnecessary.

Without a start fragment command, a file is mapped as a fragment; the fragment name is derived from the file name. If a file is divided into several fragments (by "AWP_Start_Fragment"), the file must begin with the "AWP_Start_Fragment" command.

Importing fragments

You can declare a fragment in an HTML page and import this fragment into other web pages.

Example

A company logo is to be displayed on all web pages of a web application.

There is only one instance of the HTML code for the fragment that displays the company logo. You can import the fragment as often and into as many HTML files as required.

Syntax

```
<!-- AWP_Import_Fragment Name = "<name>"-->
```

- <name> is the name of the fragment to be imported.

Example

HTML code within a web page that declares a fragment:

```
<!-- AWP_Start_Fragment Name = "My_Company_Logo" -->
<p><img src = "compay_logo.jpg"></p>
```

Example

HTML code within another web page that imports the declared fragment:

```
<!-- AWP_Import_Fragment Name = "My_Company_Logo" -->
```

Creating and loading a data block

Requirement

- All source files required for the web application (*.html, *.js, *.png, ...) have been created.
- The source files are located in one folder, but only those source files that are required for the web application. No other files may be located in this folder.

Note

Length of file names and tag names

If you have a comprehensive web application with many files and directories, the generation of the web data blocks may possibly fail. If this happens, the generation is aborted with the message "Text list overflow...". The cause is system-internal size limitations for management information saved in the web data block.

Remedy: Use short file names and short tag names.

Procedure

To create data blocks from the source files for user-defined web pages in STEP 7, proceed as follows:

1. Select the CPU, for example, in the device configuration.
2. Select the properties for user-defined web pages in the inspector window under "Properties > General > Web server".
3. As "HTML source", select the folder that contains the source files for the web application.
4. Enter the HTMP page to be opened on starting the web application as the start HTML page.
5. Enter a name for the application if required.
6. You can supplement a range of file name extensions as "Files with dynamic content" if necessary. Only enter those file name extensions that also contain AWP commands.
7. The number for the Web Control DB and for the fragment DB start number can be kept as long as they are not already being used by your user program.
8. Click on the "Generate" button to create DBs from the source files.
   The generated data blocks are saved in the project navigation in the "System block" folder (in the "Web server" subfolder).
9. In the CPU, select the network view to be loaded and then select the "Download to device" command in the "Online" menu to download the blocks. Compilation of the blocks is implicitly initiated before downloading.
   If errors are reported during this process, you must correct these errors before you can download the configuration.

Creating and loading a data block

Requirement

- All source files required for the web application (*.html, *.js, *.png, ...) have been created.
- The source files are located in one folder, but only those source files that are required for the web application. No other files may be located in this folder.

Note

Length of file names and tag names

If you have a comprehensive web application with many files and directories, the generation of the web data blocks may possibly fail. If this happens, the generation is aborted with the message "Text list overflow...". The cause is system-internal size limitations for management information saved in the web data block.

Remedy: Use short file names and short tag names.

Procedure

To create data blocks from the source files for user-defined web pages in STEP 7, proceed as follows:

1. Select the CPU, for example, in the device configuration.
2. Select the properties for user-defined web pages in the inspector window under "Properties > General > Web server".
3. As "HTML source", select the folder that contains the source files for the web application.
4. Enter the HTMP page to be opened on starting the web application as the start HTML page.
5. Enter a name for the application if required.
6. You can supplement a range of file name extensions as "Files with dynamic content" if necessary. Only enter those file name extensions that also contain AWP commands.
7. The number for the Web Control DB and for the fragment DB start number can be kept as long as they are not already being used by your user program.
8. Click on the "Generate" button to create DBs from the source files.
   The generated data blocks are saved in the project navigation in the "System block" folder (in the "Web server" subfolder).
9. In the CPU, select the network view to be loaded and then select the "Download to device" command in the "Online" menu to download the blocks. Compilation of the blocks is implicitly initiated before downloading.
   If errors are reported during this process, you must correct these errors before you can download the configuration.

Structure of the PLC program

Your user program must call the " WWW" instruction to even allow the web application, for example, the user-defined web pages, to be available to the CPU on the standard web pages and to allow them to be called up there.

The Web Control DB you have created from the source files is the input parameter (CTRL_DB) for the "WWW" instruction. The Web Control DB references the content of the user-defined web pages coded in the fragment DB and then receives status and control information.

Calling the "WWW" instruction in the startup program

If you do not want the user program to influence requested web pages, it is sufficient to only call the "WWW" instruction once in a startup OB. This instruction initializes communication between the web server and the CPU.

Calling the "WWW" instruction in the cyclic program

The "WWW" instruction can also be called in an OB processed in cycles (for example, OB 1). This has the advantage of being able to respond to web server requests from within the user program. Manual fragments must be used for this.

In this case, you must evaluate information from the Web Control DB in order to identify the requested web page or the requested fragment. On the other hand, you must set a bit in the user program in order to explicitly release the web page to be returned by the web server after processing the web page request.

The structure of the Web Control DB is described in the following section.

Web Control DB

The Web Control DB (DB 333 by default) is created by STEP 7 and contains information on the structure of user pages, the status of communication and any errors that occur.

Additional fragment DBs are also created as well as the Web Control DB. These fragment DBs (there may also only be one fragment DB) are referenced in the Web Control DB. The fragment DBs contain the web pages and media data coded in fragments, for example, images. The content of the fragment DB cannot be changed by the user program. It is created automatically and is only for data management.

The status and control tags of the Web Control DB are accessed via symbols.

The following lists the tags of the Web Control DB required for status evaluation and to control interaction.

The Web Control DB provides two types of information:

- Global status information: Not bound to a concrete web page request.
- Request status and control information: Information about queued requests.

Global status information

"WEB-Control_DB".commandstate.init	Activates and initializes the web application.
"WEB-Control_DB".commandstate.deactivate	Deactivates the web application.
"WEB-Control_DB".commandstate.inititializing	The web application is initialized (read Web Control DB, etc.).
"WEB-Control_DB".commandstate.error	Web application could not be initialized. The reason is coded in "WEB-Control_DB".commandstate.last_error .
"WEB-Control_DB".commandstate.deactivating	The web application is closed.

"WEB-Control_DB".commandstate.initialized	The web application has been initialized and is ready.
"WEB-Control_DB".commandstate.last_error	Refer to the next table for a value table of possible errors.

Last_error	Description
1	Fragment DB is inconsistent (does not match the Web Control DB).
2	A web application already exists with this name.
3	Memory problem initializing in the web server.
4	Inconsistent data in the Web Control DB.
5	A fragment DB is not available (not loaded).
6	No AWP ID for a fragment DB.
7	The enum fragment is not available (contains the texts and information on the enum types).
8	An action requested via the command flag in the Web Control DB is prohibited in the current state.
9	Web application is not initialized (if there is no reinitializing after disabling).
10	Web server is disabled.
...	Last_error is reset once the web application has been successfully initialized.

Request status information

Request status information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].idle	Nothing need be done.
"WEB-Control_DB".requesttab[x].waiting	The user program must react to a request from a manual fragment and explicitly initiate further processing in the web browser.
"WEB-Control_DB".requesttab[x].sending	The web server is occupied with processing the request/fragment.
"WEB-Control_DB".requesttab[x].aborting	The TCP connection is closed by the web server.

Request control information

Request control information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].continue	Releases the fragment being processed for transmission. Processing of the next fragment is initiated.
"WEB-Control_DB".requesttab[x].repeat	Releases the fragment being processed for transmission. The fragment is then processed again.

"WEB-Control_DB".requesttab[x].abort	Closes the TCP connection.
"WEB-Control_DB".requesttab[x].finish	Releases the fragment being processed for transmission. Stops further processing of requests (terminates the request).

Example:

The tag for the DB is: "WEB-Control_DB". Whether errors have occurred during initialization of the web application can be determined by requesting bit "WEB-Control_DB".commandstate.error in the user program.

If an error has occurred you can analyze it using the "WEB-Control_DB".commandstate.last_error value.

Interaction with the user program

With the help of manual fragments, you can make sure that the user program reacts synchronously to browser entries so that the returned website can be prepared by the user program.

Fragment type

To react to the received data in the user program the "manual" fragment type must be used for the fragment writing the data (for "manual pages"):

```
<!-- AWP_Start_Fragment Name="testfrag" ID="1" Type="manual" -->
```

The values are always transferred to the Web server of the CPU for automatic and manual pages in the same way:

Example:

```
<form method="POST" action="">
 <p>
  <input type="submit" value="Set new value">
  <input type="text" name='"Velocity"' size="20">
 </p>
</form>
```

User program for manual fragments

When using manual pages, the "WWW" instruction must be called in cycles in the user program of the CPU.

To react to values entered in the browser, the request - which is made by the manual page to the Web server - must first be evaluated in the user program. The web-control-DB (for example, DB 333) must investigate pending requests to do this. The array that manages four requests is contained in the "requesttab" section of the Web Control DB. Each element of the array contains information about the respective request in a structure.

A simple programming example shows how queued requests are checked based on the tags of the Web Control DB.

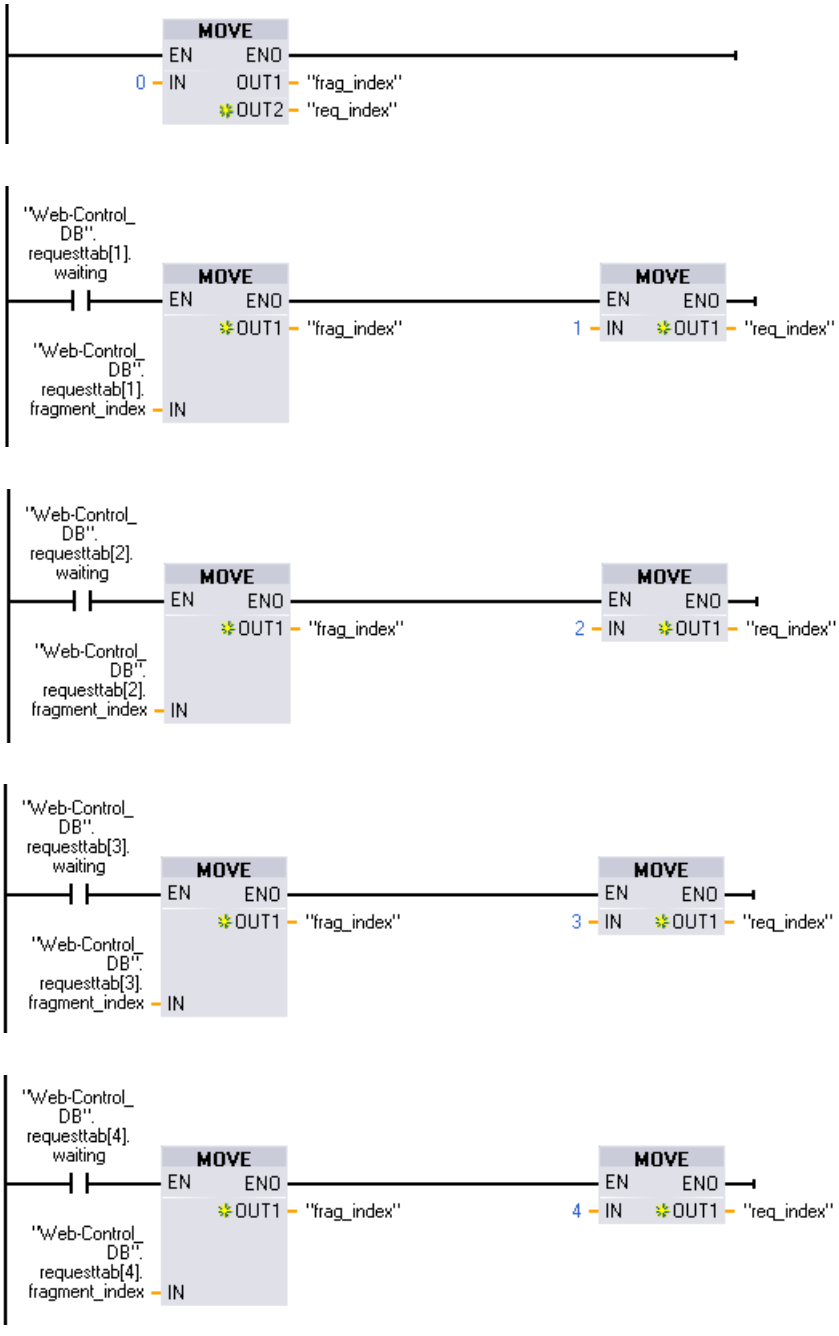

In cases where a request has been made, this program section writes the fragment ID in the #frag_index tag and the request no. (value range 1-4) in the #req_index tag.

Using the information from this, the information transferred in the request can now be processed separately for each fragment ID in the program (for example, plausibility check).

Once processing of the request has been completed by the program, the request must be answered and the appropriate entry is once more reset under"requesttab" of the Web Control DB (for example, DB 333).

A simple programming example for replying to requests:

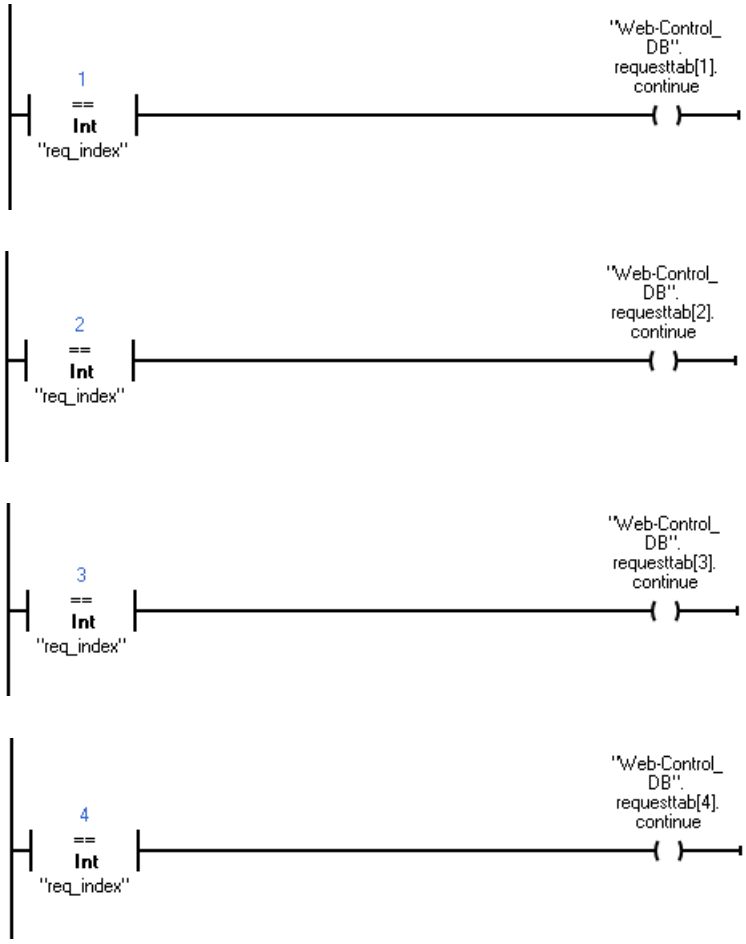

Principle sequence of a browser request with interaction from the user program

The following figure shows the simplified, principle sequence of the web browser request on the effects of Web Control DB content and the actions required from the user program until the processed web page is returned (response).

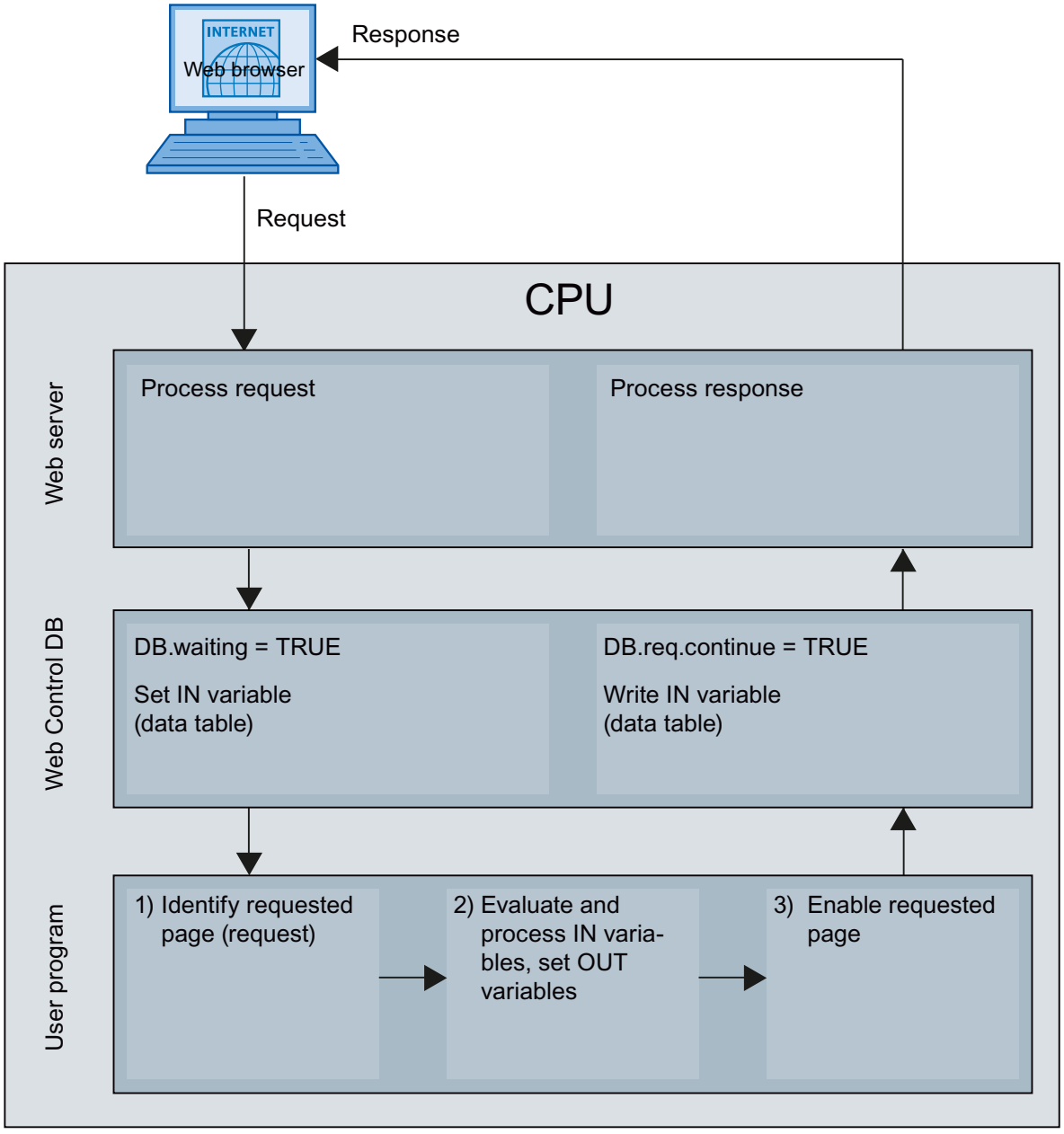

Displaying custom web pages in the browser

Display web pages in browser

Web pages are called from the standard web pages of the web browser.

In addition to the other links in the navigation bar, the standard web pages also have a link to "user pages".

Click on the "user pages" link to open the web browser you have configured as the default HTML page.

Creating custom web pages in several languages

You can make each of your custom web pages available in various languages.

Requirements

The language-dependent HTML; pages must be stored in a folder structure containing folders with the respective language abbreviations:

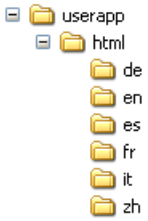

Specified language abbreviations

Language abbreviations "de", "en", "fr", "es", "it" and "zh" are fixed. Additional language folders or other designated language folders are not supported.

Additional folders within the same folder hierarchy for other files can be created as required; for example, an "img" folder for images and a "script" folder for JavaScript files.

Selecting an entry page

Defining the user page as the start page

In addition to the default intro page, you can also define the start page of your user pages as the start page of the Web server.

Follow these steps to define the user pages in STEP 7 as the start page of the Web server:

1. Select the CPU in the device configuration.
2. Open the settings in the Inspector window of the CPU under "Properties > General > Web server".
3. Select the entry "AWP1" in the area "Entry page" under "Select entry page".

If you now enter the IP address of the CPU in the browser, a connection to your user pages is automatically established.

If you want to access the Web pages of your CPU again, link the Web pages from your user pages, e.g. via the URL "http://a.b.c.d./Portal/Portal.mwsl?PriNav=Start" or "https://a.b.c.d/

Portal/Portal.mwsl?PriNav=Start". In this case, the information "a.b.c.d" represents, as an example, the IP address of the configured CPU.

Example of the link in HTML:
```
<a href="/Portal/Portal.mwsl?PriNav=Start">SIMATIC Web pages</a>
```

Note

If you define your user page as start page of the Web server, all direct access to the Web pages of the CPU is blocked. This applies also to the bookmarks you saved for the Web pages of the CPU as well as the page for reading out the service data.

Reading out service data

If you define your user page as the start page of the Web server, all direct access to the Web pages for reading out the service data is also blocked.
If you want to continue to read out service data via the Web server in a service situation, here is how you can link the service data page directly to your user page.
Just as for the web pages of the CPU, link the service data page e.g. via the URL "http://a.b.c.d/save_service_data" or "https://a.b.c.d/save_service_data"; the "a.b.c.d" here is an example of the IP address of the configured CPU.

Example of the link in HTML:
```
<a href="/save_service_data">Service data</a>
```

Language switching for custom web pages

Requirements

The HTML pages are contained in the predefined language folders, for example, HTML pages with German text are in the "de" folder, HTML pages with English text are in the "en" folder.

Language switching concept

Language switching is based on a predefined cookie named "siemens_automation_language". If the cookie is set to value "de", at the next web page request or web page update, the web server switches to the web page from the "de" folder.

Similarly, the web server switches to the web page from the "en" folder when the cookie is set to "en".

Example of language switching

The example is structured as follows:

- The language-dependent HTML files with the same name, for example, "langswitch.html" are located in both language folders "de" and "en". The text to be displayed within the two files are German or English, corresponding to the name of the folder.
- There is an additional "script" folder in the folder structure containing the JavaScript file "lang.js". Functions required for language switching are stored in this file .

Structure of the "langswitch.html" file ("de" folder)

Meta data "content language", charset and path to JavaScript file are set in the file header.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="de">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Switch language to German page</title>
<script type="text/javascript" src="script/lang.js" ></script>
<head>
```

Language selection is implemented in the body of the file by the "select" HTML element. The select element initiates a list box and contains the "de" option, labeled as "German" and "en", labeled as "English"; "de" is the default.

The "DoLocalLanguageChange(this)" function is called using the "onchange" event handler. The "this" parameter transmits the select object with the selected option to this function. "onchange" calls the function each time the option is changed.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
         <!-- change language immediately on change of the selection
-->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" selected >Deutsch</option>
                <option value="en" >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Structure of the "langswitch.html" file ("en" folder)

The header of the HTML file with English text is structured similarly to the HTML file with German text.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="en">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Language switching english page</title>
<script type="text/javascript" src="script/lang.js" ></script>
```

Language selection is also implemented in the body of the file by the "select" HTML element. In contrast to the German HTML file, the English option is already selected as a default and the test or the labels are in English.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
```

```
         <!-- change language immediately on change of the selection
-->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" >German</option>
                <option value="en" selected >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Structure of "lang.js" file (in the "script" folder)

The " DoLocalLanguageChange" function is defined in the Java script file and calls the "SetLangCookie" function with the language selection value. SetLangCookie combines the cookie name and cookie value and then sets the cookie by means of the corresponding document.cookie property. The web page must then be reloaded (top.window.location.reload) to allow the web server to react to the setting of the cookie by displaying the required language.

```
function DoLocalLanguageChange(oSelect) {
        SetLangCookie(oSelect.value);
        top.window.location.reload();
    }

 function SetLangCookie(value) {
        var strval = "siemens_automation_language=";
            // this is the cookie by which the web server
            // detects the desired language
            // this name is required by the web server
        strval = strval + value;
        strval = strval + "; path=/ ;";
            // set path to the application, since otherwise
            // path would be set to the requesting page
            // would not get the cookie.
            // The path for user defined applications follows this
sample:
            // path=/awp/<application name>/<pagename>
            // example: path=/awp/myapp/myappstartpage.htm
            //(where myapp is the name of the web application
            // entered in the web server properties of the cpu)
            /*
            use expiration if this cookie should live longer
            than the current browser session
            var now      = new Date();
            var endttime = new Date(now.getTime() + expiration);
            strval = strval + "; expires=" + endttime.toGMTString()
+ ";";
            */
        document.cookie = strval;
    }
```

Additional examples

Filling bottles with different drinks

This example describes a bottling plant (http://support.automation.siemens.com/WW/view/en/94681612) that can be monitored and controlled via a user-defined Web page.

It is possible to decide the type of drinks to be bottled using a recipe.

For the control and visualization of the plant, no panel (HMI) is necessary.

A Web browser serves to display the Web page.

The Web page is regularly supplied with current process values.

This is done with an invisible inline frame (iframe) that is re-sent regularly by the Web server. Current process values reach the browser with the inline frame. Using JavaScript, the process values are read from the inline frame and inserted in the main page.

In addition to this, users can call functions on the CPU via the Web page, for example starting or stopping bottling or importing or exporting recipes (refer to the link Bottling plant application (http://support.automation.siemens.com/WW/view/en/94681612)).

Filling and emptying a tank

This example describes a tank for liquids (http://support.automation.siemens.com/WW/view/en/58862931) that can be monitored and controlled via a user-defined Web page. Using buttons, the user controls the inflow into the tank and the outflow from the tank.

The current fill level is displayed along with the last ten fill levels with time information

The pages "Overview" and "Data" read in the current process values at fixed intervals whereby the Web server sends the pages completely new to the browser.

The "DataOpti" page, on the other hand, uses an inline frame simply to receive the current fill levels of the tank from the Web server (see Tank simulation application (http://support.automation.siemens.com/WW/view/en/58862931)).

1.4.2.5 Configuration control (option handling) for devices

Important information regarding configuration control (option handling)

Operating principle

With S7-1200 Firmware Version 4.1 or higher, the configuration control enables you to configure the structure of a control system and to work with variants (options) that differ from this configuration.

- All modules that are needed in a set of similar plant units or machines are configured in a master project with a maximum configuration (station master).
- Provision is made in the user program of the master project for various station options for various plant units or machines as well as the selection of a station option. A station option uses, for example, only some of the configured modules and these modules are inserted in varying order.
- An operator selects a station option for a specific plant on-site. The operator does not have to modify the project and thus also does not have to download a modified configuration for this.

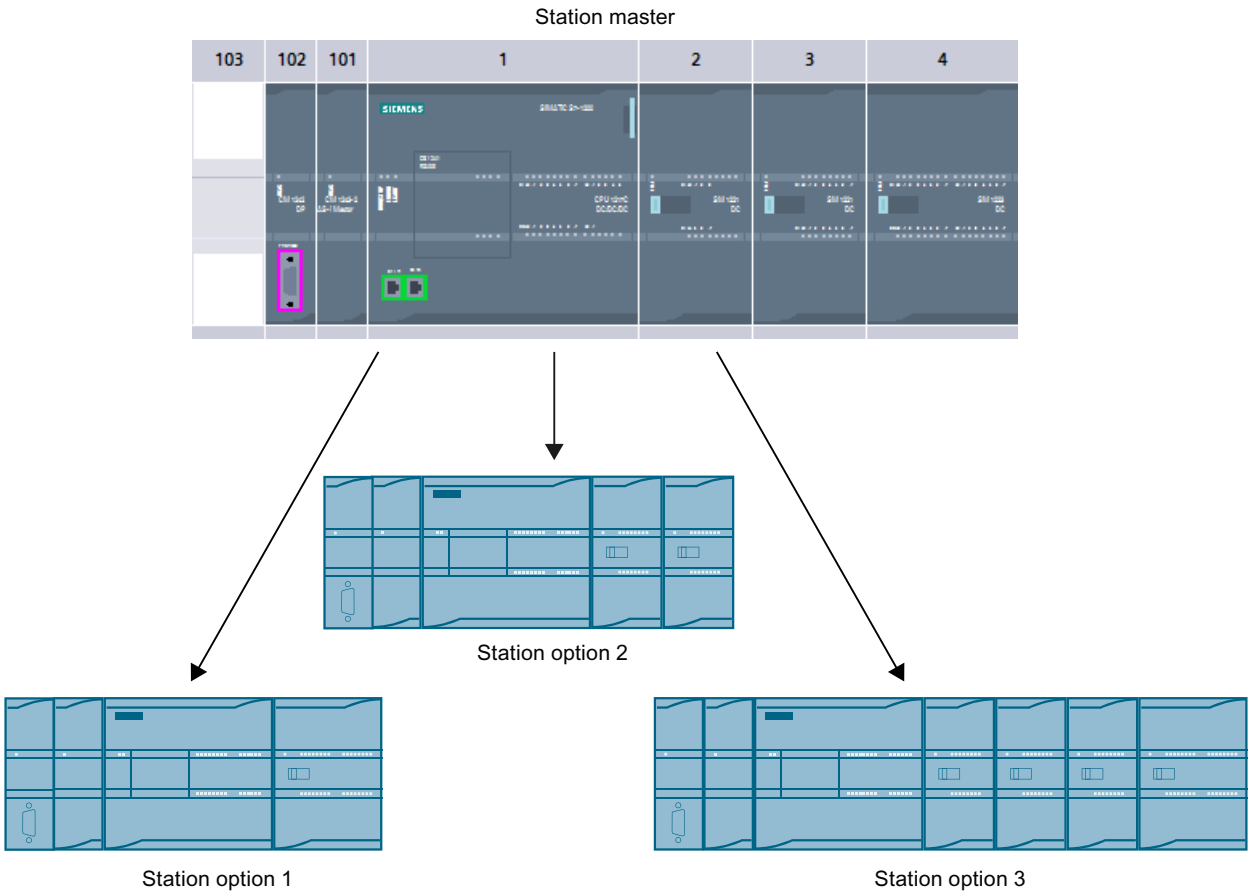

A control data record you have programmed in the startup program notifies the CPU as to which modules are missing or located in different slots as compared to the preset configuration. The configuration control has no effect on the parameter assignment of the modules.

Configuration control gives you the flexibility to vary the central installation as long as the real configuration can be derived from a preset maximum configuration.

Below you will find a description of how to activate configuration control (CPU parameter assignment) and how to structure the required data record.

Requirements

- STEP 7 Version V13 SP1 or higher
- CPU S7-12XX Firmware Version V4.1 or higher Modules that support the "Configuration control" function also have the "Configuration control" entry in the description (info text) of the hardware catalog.
- Recommendation: Before you load a new program with a modified control data record, perform a memory reset. This action will prevent inconsistent states that may result from the presence of incompatible control data records.
- The startup parameter "Compare preset to actual configuration" is set to "Startup CPU even if mismatch" (default setting).

Required steps

1. Enable the "Allow to reconfigure the device via the user program" parameter when configuring the CPU ("Configuration control" area).

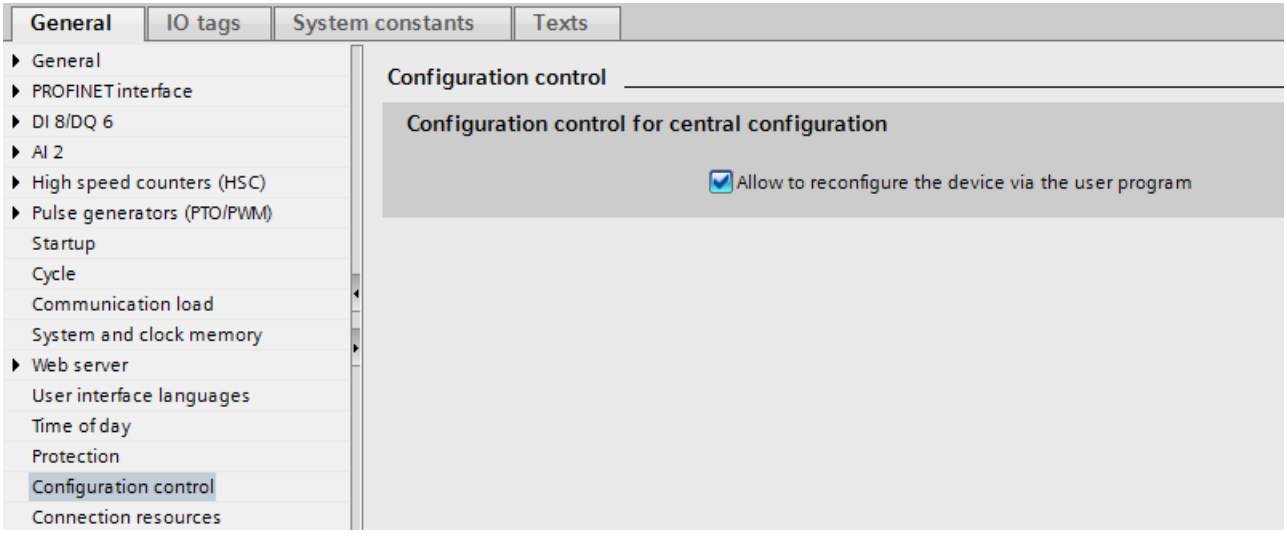

2. Create a control data record (e.g., in a data block) according to the current configuration based on the sample described below for the control data record. The control data record has the number 196. If you want to transfer the control data record as a whole block to the WRREC instruction (input parameter RECORD), note that you must first create a PLC data type containing the structure of the control data record and base the data block on this PLC data type.

ConfDB

	Name	Data type	Start value	...	...	...	...	Comment
1	Static							
2	ConfigControl	Struct			✓	✓		
3	Block_length	USInt	9		✓	✓		Length of control data record, including header
4	Block_ID	USInt	196		✓	✓		Data record number
5	Version	USInt	5		✓	✓		
6	Subversion	USInt	0		✓	✓		
7	Slot_1	USInt	16#FF		✓	✓		Assignment for CPU annex card/Actual annex card
8	Slot_2	USInt	16#FF		✓	✓		Configured slot 2/ Assigned "real" slot
9	Slot_3	USInt	16#FF		✓	✓		Configured slot 3 / Assigned "real" slot
10	Slot_4	USInt	16#FF		✓	✓		Configured slot 4 / Assigned "real" slot
11	Slot_5	USInt	16#FF		✓	✓		Configured slot 5 / Assigned "real" slot
8	Slot_6	USInt	16#FF		✓	✓		Configured slot 6 / Assigned "real" slot
9	Slot_7	USInt	16#FF		✓	✓		Configured slot 7 / Assigned "real" slot
10	Slot_8	USInt	16#FF		✓	✓		Configured slot 8 / Assigned "real" slot
11	Slot_9	USInt	16#FF		✓	✓		Configured slot 9 / Assigned "real" slot
8	Slot_101	USInt	16#FF		✓	✓		Configured slot 101 / Assigned "real" slot
9	Slot_102	USInt	16#FF		✓	✓		Configured slot 102 / Assigned "real" slot
10	Slot_103	USInt	16#FF		✓	✓		Configured slot 103 / Assigned "real" slot

3. Transfer the control data record to the CPU in the startup program.
The configuration control for the centrally inserted modules takes effect only after an operating mode change of the CPU from STOP to RUN. For this reason, call the extended WRREC (Write data record) instruction in the startup OB, and transfer the created control data record to the CPU; see next section.

If a valid control data record is not transferred in the startup OB, the control is not ready for operation. The CPU returns from startup to STOP in this case.

Transferring a control data record in the startup program

The CPU processes the WRREC instruction for asynchronous transfer of the control data record. For this reason, you must call WRREC in the startup OB repeatedly in a loop until the output parameter "BUSY" or "DONE" indicate that the data record has been transferred.

Tip: Use the SCL programming language with the REPEAT ... UNTIL instruction for programming the loop.

```
REPEAT
    "WRREC_DB"(REQ := "start_config_control",
               ID := 33,
               INDEX := 196,
               LEN := "conf_LEN",
               DONE => "conf_DONE",
               BUSY => "conf_BUSY",
               ERROR => "conf_ERROR",
               RECORD := "ConfDB".ConfigControl,
               STATUS => "conf_STATUS");
UNTIL NOT "conf_BUSY"
END_REPEAT;
```

Below, you will find explanations for individual block parameters that you must supply with specific values in the configuration control context. For the remaining block parameters, see also WRREC:

Parameter	Explanation
ID	HW identifier; is always 33 (decimal) for configuration control for centrally arranged modules.
INDEX	Data record number; is always 196 (decimal) for configuration control for centrally arranged modules.
RECORD	Control data record to be transferred. See the section "Control data record" for the structure of the control data record. Tip: The "RECORD" block parameter of the WRREC instruction (V1.1 and higher) is of the "VARIANT" data type and therefore requires a tag with data type. If you store the control data record in a data block, this data block must thus have a data type. The data block created must not be of the "Global-DB" type. Rather, it must be derived from a user data type. Proceed as follows: 1. First, create a new PLC data type (user data type) with the structure of the control data record and name it, for example, "ConfDB". 2. Create a new data block. Select the newly created user data type, for example, "ConfDB", as type for this data block.

In graphical programming languages, you realize the loop using instructions for program control.

Example in FBD: Use the LABEL (jump label) and JMP (jump at RLO=1) instructions to program a loop.

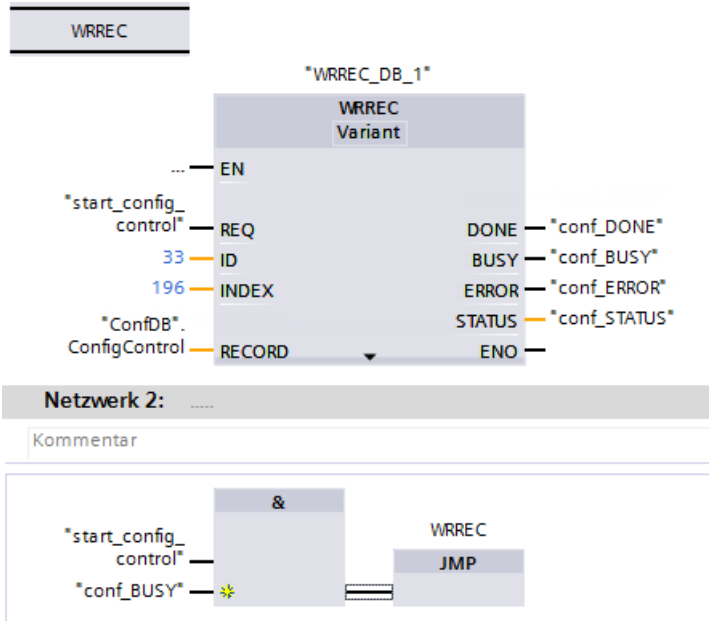

Arrangement of the modules

The following table shows the slot number assignment:

Slot	Modules	Note
1	Signal board, communication board, battery board	Slot on front of the CPU
2 - 9	Signal modules	Slots to the right of the CPU
101 - 103	Communications modules	Slots to the left of the CPU

Control data record

A control data record 196 containing a slot assignment is defined for configuration control.

The following codes apply:

0	Module is included in the hardware configuration but is not used in the current configuration.
1 to 9, 101 to 103	Current slot of the module
16#FF (255)	A module is not located in this slot in the hardware configuration.

Byte	Element	Code	Explanation
0	Block length	4 + number of slots	Header
1	Block ID	196	
2	Version	5 (for central I/O)	
3	Subversion	0	
4	Assignment of the CPU expansion board	Expansion board, 0 or 16#FF	Control element Describes in each element which real slot in the device is assigned to the configured slot. The structure of a control element is described in the following section.
5	Assignment of configured slot 2	Real slot, 0 or 16#FF	
...	...	...	
12	Assignment of configured slot 9	Real slot, 0 or 16#FF	
13	Assignment of configured slot 101	Real slot or 16#FF	In contrast to signal modules, the real slot of the communications modules must correspond to the configured slot.
14	Assignment of configured slot 102	Real slot or 16#FF	
15	Assignment of configured slot 103	Real slot or 16#FF	

Structure of a control element

A control element contains the information on which module is inserted in which slot.

The byte numbers represent the configured slots in ascending order (see above):

- Byte 4 stands for the configured slot of the expansion board
- Byte 5 to byte 9 stand for the configured slots 2 to 9
- Byte 13 to byte 15 stand for the configured slots 101 to 103

The value that you need to enter in the corresponding byte depends on the following rule:

- If the module is present in the real configuration, enter the real slot number of the module.
  - Example 1: The module configured in slot 2 is located in slot 2.
    Enter value 2 (= actual slot) in byte 5 (= configured slot 2).
  - Example 2: The module configured in slot 3 is located in slot 2.
    Enter value 2 (= actual slot) in byte 6 (= configured slot 3).
- If the module is configured but is missing in the real structure, enter 0 in the byte for the configured slot.
- If a module is not located in this slot in the hardware configuration, enter 16#FF (255) in the byte for the configured slot.

Rules

Observe the following rules:

- The configuration control does not support the repositioning of communications modules. The slot entries in the control data record for slots 101 to 103 must correspond to the real positions of the modules or be defined as not present in the hardware configuration by entering 16#FF (255).
- Slot gaps in the configuration are not allowed. For example, if a signal module is inserted in slot 4 in the real configuration, slots 2 and 3 of the real configuration must also be occupied. The same applies to the slots 101 to 103. If a communications module is inserted in slot 102 in the real configuration, slot 101 must also have a communications module inserted in the real configuration.
- If you have enabled configuration control, the CPU is not ready for operation without a control data record. The CPU returns from startup to STOP if a valid control data record is not transferred in the startup OB. The central I/O is not initialized in this case. The cause for the STOP mode is entered in the diagnostics buffer.
- For addressing the WRREC instruction, use the HW identifier 33 (decimal, for the ID block parameter) to write the control data record.
- The control data record is saved retentively in the CPU, which means that it is not necessary to write the control data record 196 again at a restart if the configuration is unchanged. Prior to commissioning, we recommend that a memory reset be performed for the CPU in order to delete any control data record that is present.
- Slot entries in the control data record outside the configured preset configuration are ignored by the CPU.
- Each real slot may only be present once in the control data record.
- A real slot may only be assigned to one configured slot.

Note

Modified configuration

The writing of a control data record with a modified configuration triggers the following automatic reaction by the CPU:

Memory reset with subsequent startup with this modified configuration.

As a result of this reaction, the retentively saved original data record 196 is deleted and the new data record 196 is saved retentively.

Behavior during operation

Effect of the discrepancy between the configured configuration and real configuration:

- For the online display and for the display in the diagnostics buffer (module OK or module faulty), the hardware configuration is always used and not the differing real configuration. Example: A module supplies diagnostic information. This module is configured in slot 4, but is really inserted in slot 3 (missing module; see example in the next section). In the online view, a configured slot 4 is indicated as faulty. In the real configuration, the module in slot 3 indicates an error via an LED display.

If modules are entered as missing in the control data record, the automation system behaves as follows:

- Modules designated as not present in the control data record do not supply diagnostics and their status is always OK. The value status is OK
- Direct write access to the outputs or write access to the process image of outputs that are not present: Remains without effect; no access error is signaled.
- Direct read access to the inputs or read access to the process image of inputs that are not present: Value "0" is supplied; no access error is signaled.
- Write data record to module that is not present: Remains without effect; no error is signaled.
- Read data record from module that is not present: An error is signaled because a valid data record cannot be returned.

Error messages

The following error messages are returned if an error occurs during writing of the control data record:

Table 1-93 Error messages

Error code	Meaning
16#80B1	Invalid length; the length information in data record 196 is not correct.
16#80B5	Configuration control parameters not assigned.
16#80E2	Data record was transferred in the wrong OB context. The data record must be transferred in the startup program.
16#80B8	Parameter error; module signals invalid parameters.

See also

S7-1200 System Manual (http://support.automation.siemens.com/WW/view/en/89851659)

Example of a configuration control

A configuration consisting of a CPU and 3 signal modules is configured in the following.

The module at slot 3 is not present in the first configuration expansion and is "hidden" by the configuration control.

In the second configuration expansion, the module that was initially hidden is located in the last slot. The added slot is made known to the CPU by a modified control data record.

Actual configuration with missing module

The specified configuration contains all modules that can be present in a final expansion stage.

The module that is inserted in slot 3 in the specified configuration is missing in the real expanded configuration. For this reason, slot 3 must be coded in the control data record accordingly with "FF_H" (= not present).

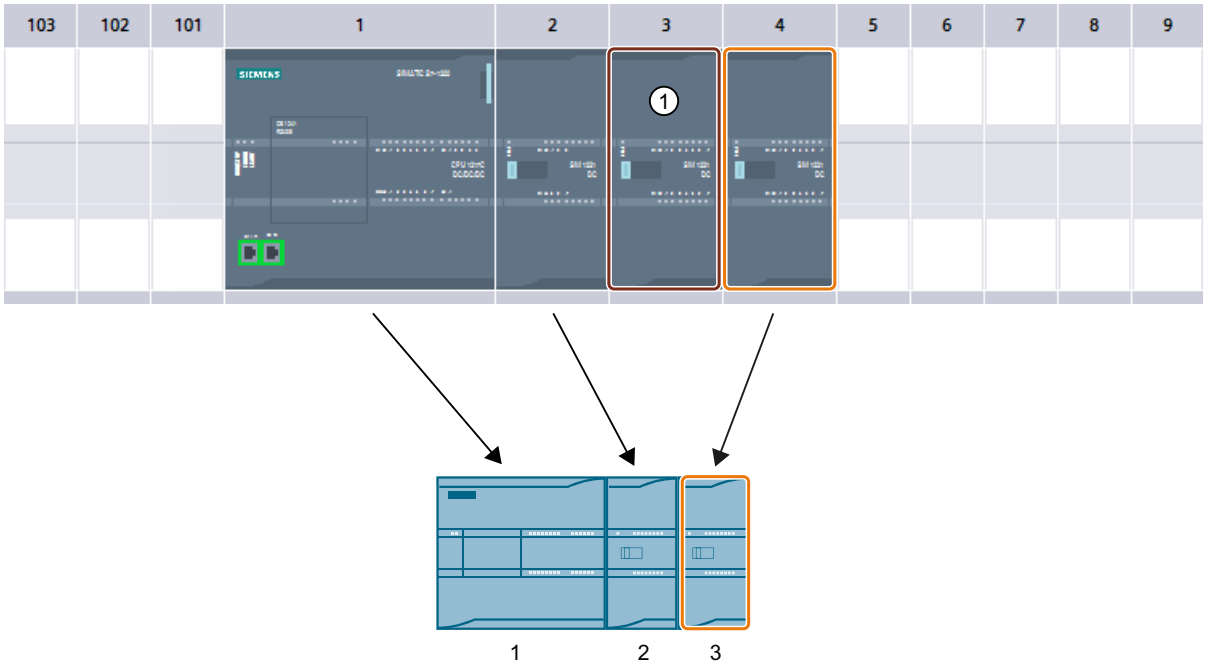

ControlDataRecord

	Name	Data type	Start value	Comment
1	Static			
2	ConfigControl	Struct		
3	Block_length	USInt	16	Length of control data record, including header
4	Block_ID	USInt	196	Data record number
5	Version	USInt	5	
6	Subversion	USInt	0	
7	Slot_1	USInt	255	Assignment for CPU annex card/Actual annex ca...
8	Slot_2	USInt	2	Configured slot 2 / Assigned ① al" slot
9	Slot_3	USInt	0	Configured slot 3 / Assigned "real" slot
10	Slot_4	USInt	3	Configured slot 4 / Assigned "real" slot
11	Slot_5	USInt	255	Configured slot 5 / Assigned "real" slot
12	Slot_6	USInt	255	Configured slot 6 / Assigned "real" slot
13	Slot_7	USInt	255	Configured slot 7 / Assigned "real" slot
14	Slot_8	USInt	255	Configured slot 8 / Assigned "real" slot
15	Slot_9	USInt	255	Configured slot 9 / Assigned "real" slot
16	Slot_101	USInt	255	Configured slot 101 / Assigned "real" slot
17	Slot_102	USInt	255	Configured slot 102 / Assigned "real" slot
18	Slot_103	USInt	255	Configured slot 103 / Assigned "real" slot

① Module is missing in the real configuration

Actual configuration with subsequently added module

The module present in slot 3 in the specified configuration is added to the back of the real configuration by inserting it as the last module in slot 4.

The control data record is adapted accordingly.

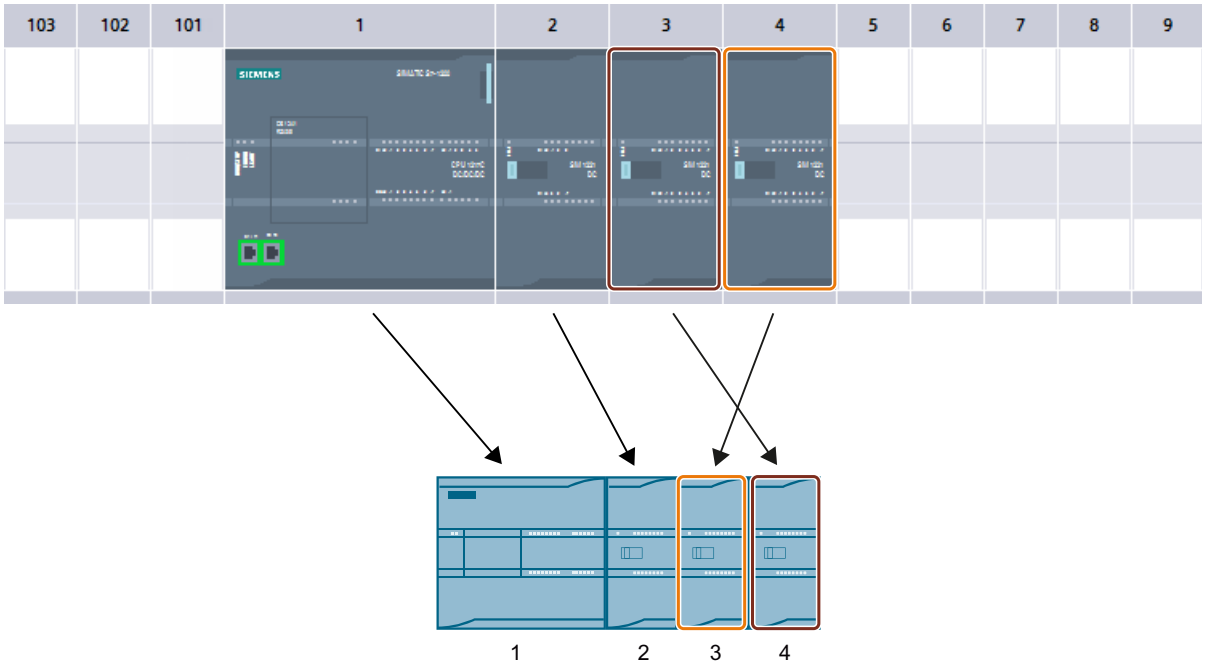

ControlDataRecord

		Name	Data type	Start value	Comment
1		Static			
2		ConfigControl	Struct		
3		Block_length	USInt	16	Length of control data record, including header
4		Block_ID	USInt	196	Data record number
5		Version	USInt	5	
6		Subversion	USInt	0	
7		Slot_1	USInt	255	Assignment for CPU annex card/Actual annex card
8		Slot_2	USInt	2	Configured slot 2/ Assigned "real" slot
9		Slot_3	USInt	4	Configured slot 3 / Assigned "real" slot
10		Slot_4	USInt	3	Configured slot 4 / Assigned "real" slot
11		Slot_5	USInt	255	Configured slot 5 / Assigned "real" slot
12		Slot_6	USInt	255	Configured slot 6 / Assigned "real" slot
13		Slot_7	USInt	255	Configured slot 7 / Assigned "real" slot
14		Slot_8	USInt	255	Configured slot 8 / Assigned "real" slot
15		Slot_9	USInt	255	Configured slot 9 / Assigned "real" slot
16		Slot_101	USInt	255	Configured slot 101 / Assigned "real" slot
17		Slot_102	USInt	255	Configured slot 102 / Assigned "real" slot
18		Slot_103	USInt	255	Configured slot 103 / Assigned "real" slot

1.4.2.6 Additional configurations

Configuring additional functions

The S7-1200 automation system has numerous additional functions that are useful as integrated CPU functions or available via plug-in modules (for example, communication modules). You can find the description via the following links.

1.4.3 Configuring PC systems

1.4.3.1 Overview

Important information about PC systems and PC stations

Definition of PC systems

PC systems are available in a wide range of designs, which can flexibly be expanded with software and hardware components. In this case, we are describing PC systems with a software controller that makes a PC (e.g. IPC) into a controller - similar to a modular controller such as a CPU S7-300/400 or a CPU S7-1500.

PC Station and possibly other components are installed in a PC system.

Definition of PC Station

PC Station is a software component that manages SIMATIC software products and the use of interfaces on a PC. In addition, it controls communication among the SIMATIC products on a PC.

PC Station is required by the SIMATIC software products SIMATIC NET PC software, WinAC RT or the SIMATIC S7-1500 software controllers and installed on the target PC.

SIMATIC HMI products such as WinCC Advanced RTX can work with different versions of PC Station.

As of version V2.x, PC Station can also be installed without the software controller.

Selecting the version of PC Station

During configuration, the version of your PC Station is automatically adapted by the software components configured in the PC system.

The following scenario is an example of a version change for PC Station:

- You have selected an IPC from the hardware catalog that conforms to PC Station V2.x:
  - In the initial state of the IPC, PC Station V1.0 is the default.
  - PC Station V1.0 remains installed as long as components that are compatible with PC Station V1.0 are inserted in the PC system.
  - The version of the PC Station only adjusts automatically to V2.x when the S7-1500 version of the software controller is inserted in an IPC.
  - If you **only** insert WinCC Runtime, the display changes from "PC Station" to "not installed". No installation of PC Station is required on the PC.
  - If you insert WinCC Runtime and a software controller is already inserted, the version of PC Station remains unchanged.

Note

When you include incompatible software products in the PC system, you receive an error message when trying to compile the station.

The properties of PC Station indicate which version of PC Station is installed: For the selected PC Station, you can find the settings in the Inspector window in the "PC Station" area.

Reference

You can find more information on operating a software controller with a PC Station in the SIMATIC ET 200SP Open Controller CPU 1515SP PC (https://support.industry.siemens.com/cs/de/en/view/109248384) manual, in the SIMATIC S7-1500 CPU 150xS (https://support.industry.siemens.com/cs/de/en/view/109249299) operating manual and in the SIMATIC NET PC software Commissioning PC Stations (https://support.industry.siemens.com/cs/de/en/view/109488960) configuration manual.

Software components of a PC station

A PC station contains a range of configurable software components, which are applied during loading to the PC station. There are different applications which have different effects on the generation and handling of the configuration data.

- PC station V2.x:
  - WinCC RT Advanced, as of V13 SP1
  - S7-1500 Software Controller
- PC station V1.0:
  - Standard application
  - OPC server as a central communication service provider
  - WinCC RT Advanced
  - WinCC RT Professional
  - WinCC V7
  - WinAC RTX
  - Communications processors
- Not installed:
  - WinCC RT Advanced
  - WinCC RT Professional
  - WinCC Client

You configure the specific properties of these components according to the utilized services.

WinCC RT Advanced

WinCC RT Advanced is a PC-based HMI solution for single-user systems directly at the machine level. The WinCC RT software is integrated in the SIMATIC HMI devices (e.g. Basic/Comfort/Mobile Panel) and permits data exchange over HMI connections.

If you want to set up several HMI connections to a CPU, use for example:

- The PROFINET and PROFIBUS DP interfaces of the PC station
- CPs and CMs with the relevant interfaces

Other communication options:

- OPC UA client or as server (all HMI versions)
- Communication between HMI systems over the Intranet/Internet
  - Read and write access to tags

S7-1500 Software Controller

The SIMATIC S7-1500 Software Controller is a PC-based controller.

- It offers the same functionality as CPUs of the SIMATIC S7-1500 automation system.
- Software Controllers offer the functionality of a programmable logic controller (PLC) in a PC-based real-time environment.
- In order to develop your user program, use the same programming languages, program structure, and programming interface (STEP 7) with the PC-based controller as those used for hardware controllers.
- For the SIMATIC S7-1500 Software Controller, you can use the same user program as for a hardware controller. The Software Controller CPU has a display application that runs on the PC. The CPU's operating mode can be shown with the display application. With the display application - similar to the display of a hardware CPU - you can execute diagnostics and commissioning tasks.
- The Software Controller makes use of the interfaces of the PC for PROFINET and PROFIBUS.
  Windows-independent use of PROFINET or PROFIBUS for operating distributed I/O.
  Depending on the interface hardware used, the following functions are possible:
  - PROFINET IO RT or PROFINET IO IRT
  - PROFIenergy
  - I-device
  - PROFIBUS DP master
- Communication (SIMATIC communication, Open User Communication, OPC UA) with Windows applications or external devices via the Windows interfaces of the PC.
  The communications services are based on the module in use.

The Software Controller based on S7-1500 is available in the following versions:

- As SIMATIC ET 200SP Open Controller CPU 1515SP PC and as CPU 1515SP PC including HMI in ET 200SP design, with pre-installed and pre-configured SIMATIC S7-1500 Software Controller. Can be centrally expanded using ET 200SP I/O modules.
- As CPU 1507S and CPU 1508S Software Controller for PC-based automation. The SIMATIC S7-1500 Software Controller provides the benefits of the standard SIMATIC S7-1500 controller on high-performance industrial PCs.

Standard application

With standard applications, you communicate with other applications and devices via your PC module.

You can configure the following communication services depending on the PC module in use:

- S7 connections
- DP services

OPC server as central communication service provider

OPC clients use the interface to an OPC server to communicate with an automation device (such as a SIMATIC S7-1500).

Configure its properties to adapt the OPC server. The set or changed parameters take effect after the configuration data is downloaded to the PC station and the OPC server is started.

You can use the following communication services depending on the PC module in use:

- All connection types
- DP services (DPV0)

WinAC RTX

The SIMATIC WinAC (Windows Automation Center) is a PC-based controller and provides the same functionality as SIMATIC S7-300/400 CPUs (hardware controllers). WinAC RTX is used as a solution for typical PC applications (for example, MS Office and user programs) and automation tasks in a system.

Distributed I/O is controlled via:

- PROFIBUS
  - The connection of I/Os via PROFIBUS DP takes place over the integrated DP interface of the SIMATIC PCs or over a communications processor.
- PROFINET
  - The connection of I/O devices via PROFINET takes place over an integrated Ethernet interface of the SIMATIC IPC or over a communications processor.

Optimum access to the data, e.g. for image processing, measured value acquisition, etc., takes place via the WinAC option Open Development Kit (ODK).

You can use the following communication services depending on the PC module in use:

- PG/OP communication
- S7 communication
- Open User Communication (OUC)
- Process data access via OPC

Communications processors

Connecting the SIMATIC controllers via communications processors enables direct integration of controllers in industrial networks.

You can use the following communication services depending on the PC module in use:

- OPC UA client or as server
- Open communication (TCP/IP and UDP): Multicast with UDP
- PG/OP communication: cross-network via S7 routing
- S7 communication (client, server)
- IT communication (HTTP(S), e-mail)

- PROFINET IO controller with real-time properties (RT and IRT)
- IP address assignment via DCP
- Shared device
- PROFIBUS DP
- PG/OP communication

WinCC RT Professional

WinCC RT Professional is a PC-based control and monitoring system for visualizing and operating processes, production flows, machines and plants – from simple single-user systems to multi-user systems and multi-site solutions with Web clients.

The WinCC RT software is integrated in the SIMATIC HMI devices (e.g. Basic/Comfort/Mobile Panel) and permits data exchange over HMI connections.

If you want to set up several HMI connections to a CPU, use for example:

- The PROFINET and PROFIBUS DP interfaces of the PC station
- CPs and CMs with the relevant interfaces

Other communication options:

- OPC server
- OLE DB server
- WinCC WebNavigator
- WinCC DataMonitor

WinCC Client

Depending on requirements, a WinCC single-user system can be expanded into a high-performance client/server system. In this way, several coordinated HMI stations can be operated in a single group with networked automation systems.

Reference

You can find more information on WinCC RT in the WinCC online help.

You can find additional information on the S7-1500 Software Controller and WinAC RTX in the section What you need to know about PC systems (Page 1427).

You can find additional information about the meaning and handling of PC applications and the OPC server in the SIMATIC NET - Industrial Communication (https://support.industry.siemens.com/cs/de/en/view/https://support.industry.siemens.com/cs/de/de/view/61630799) configuration manual and the SIMATIC NET - Commissioning PC Stations (https://support.industry.siemens.com/cs/de/en/view/109488960) configuration manual.

You can find more information on communication processors on theService&Support (http://w3.siemens.com/mcms/industrial-communication/en/ie/system-interfacing/advanced-controller/Pages/CPs-fuer-advanced-controller.aspx) Web pages.

You can find application examples for changing to the new Software Controller generation in the Guide for Migrating PC-based Controllers: from SIMATIC WinAC RTX to the SIMATIC S7-1500 Software Controller and TIA Portal (https://support.industry.siemens.com/cs/document/109478804/guide-for-migrating-pc-based-controllers%3A-from-simatic-winac-rtx-to-the-simatic-s7-1500-software-controller-and-tia-portal?dti=0&lc=en-WW).

Improvements of PC Station V2.x compared to PC Station V1.0

The common features and improvements of the two PC Stations are listed briefly below.

Common features of PC Station V2.x and V1.0

PC Station V2.x and PC Station V1.0 have the following common features:

- Simple installation of all required software components via a setup (for example, S7-1500 Software Controller, WinCC Advanced RT, etc.)
- PC systems with pre-installed software controller or pre-installed Runtime Advanced can also be ordered.

Improvements of PC Station V2.2 compared to PC Station V2.1

PC Station V2.2 offers the following improvements compared to PC Station V2.1:

- Loading project data in a configuration file

Improvements of PC Station V2.1 compared to PC Station V2.0

PC Station V2.1 offers the following improvements compared to PC Station V2.0:

- Access possible via a Web server (Page 1449)

Improvements of PC Station V2.0 compared to PC Station V1.0

Compared to PC Station V1.0, PC Station V2.x provides the following improvements:

- Loading the PC Station configuration is easy, similar to a modular S7-1500 controller; no prior configuration of the PC system setup is needed on the IPC.
- Integrated PC hardware diagnostics is integrated into the system diagnostics, for example, to diagnose the state of PC components such as fan or hard disk.

Rules for the configuration

Rules and special features for configuring a PC system

Due to the numerous possible combinations of software and interface cards with various IPC versions, there are complex dependencies.

We recommend simply configuring PC Station according to your requirements and then compiling. STEP 7 performs most of the required adjustments, such as setting the station type.

If a combination cannot be compiled, an error message indicates that a rule has been violated and you receive information about how to rectify the error.

The following sections provide information on the basic rules:

- If an IPC is suitable for an S7-1500 Software Controller, it can also be inserted in a WinAC RTX. Applies to most IPCs.
- The following components are suitable or not suitable for the S7-1500 controller software:
  - The open controller (CPU 1515SP PC) is not suitable for the S7-1500 Software Controller.
  - The IPC2x7D and Embedded Controller (S7-mEC 31) are not suitable for the S7-1500 controller software.
- Device replacement between S7-1500 Software Controller and WinAC RTX is not possible.
- You can only insert one software controller in a PC system in each case.
- Neither an OPC server nor an application (user application) can be used in combination with an S7-1500 Software Controller. An installed PC Station V2.x prevents installation of the OPC server or the application on an IPC.
  Both components are used for a PC Station V1.0.
- Assigning interfaces: Assigning "None, or a different Windows setting" to an interface is only practical for onboard interfaces. The interfaces of the configured CPs/CMs are initially assigned to the PC station, because they are normally used by the configured components of the PC station.
- The graphic view of a PC Station in STEP 7 is identical in the network view and the device view.
  The "slots" on which the components of a PC system are located are not numbered. When you insert components by dragging them from the hardware catalog, STEP 7 places them automatically: The hardware components (e.g. CPs) are on the left side, the software components (e.g. a software controller) are on the right.
  Detailed information, for example, about the order number (i.e. the localization information similar to a slot in the PC system) is available only in the device overview table of the device view.
  - In the device overview table, index 125 is permanently assigned for the PC station (it is also referred as the "Station Manager" for PC Station V1.0). This index is displayed in the first row of the device overview table.
  - For the open controller CPU 1515SP PC, there are the columns "Rack", "Slot" and "I address" and "Q address" in addition to the "Index" column. These columns contain the corresponding information for inserted ET 200SP modules.
  - Inserted software components and CPs obtain their index in the order in which they are inserted in the device view. You can force a specific index only by dragging it from the hardware catalog to the corresponding row in the device overview table. This has no impact on the representation in the graphic device view.
  - You can find onboard interfaces to which no software components have been assigned starting at index 100. As soon as you have assigned an onboard interface, it is assigned the next free index, for example 4. In this way, assigned interfaces always appear at the start of the device overview table.

IPCs, CPs and software components for PC Station V2.x

The following IPCs are suitable for PC Station V2.x:

- IPC227E, IPC277E
- IPC427D, IPC477D
- IPC427E, IPC477E
- As of IPC547G (without Software Controller CPU 1505SP, CPU 1507S or CPU 1508S)
- IPC627D, IPC647D, IPC677D
- IPC827D, IPC847D (not with fail-safe versions of the Software Controller)

The following CPs are suitable for PC Station V2.x (can only be used as an interface of the S7-1500 Software Controller, CPU 1507S):

- IE generally for PROFINET IO with RT
- CP 1625 for PROFINET IO RT
- CP 1625 for PROFINET IO IRT
- CP5622 for PROFIBUS DP
- CP5623 for PROFIBUS DP

The following software components are suitable for PC Station V2.x:

- CPU 1507S Software Controller
- CPU 1505SP Software Controller
- CPU 1507S Software Controller
- CPU 1508S Software Controller
- WinCC RT Advanced as of V13 SP1

Procedure for configuring a PC system

Introduction

The following section describes how to select a PC system from the hardware catalog in the network view. Drag the required modules from the hardware catalog to this PC system; they will be arranged automatically.

Selecting the components in the hardware catalog

Each component is displayed as a folder in the hardware catalog. When you open this folder, you see the different versions of the selected component with their respective article numbers.

The following provides an example of how to create a PC system in the network view.

Requirement

- The hardware catalog is open.
- You are in the network view.

Procedure

To configure a PC system with a required software controller, follow these steps:

1. Navigate to the "PC Systems" folder in the hardware catalog.
2. Open the folder with the required industrial PC type. You will see all types of the selected IPC type.
3. Click the chosen IPC type to select it.
4. Drag the IPC from the hardware catalog and drop it into the network view.
5. Navigate to the "SIMATIC Controller Application" folder in the hardware catalog.
6. Select the desired software controller:
   - CPU 1507S or CPU 1508S
   - SIMATIC WinAC RTX
7. Drag the desired software controller and drop it onto this PC system.
8. Configure the properties of the PC system and the Software Controller.

Note

Requirements for loading

In order to load the hardware configuration, the software controller or PC station must be assigned to an interface.

Refer the following sections to learn how to assign interfaces: Important information about assigning interfaces (Page 1472) and Assigning interfaces for the communication (Page 1474).

Installation requirements depending on configured components

The configured components of a PC system in STEP 7 determine which installation requirements apply to the target system, for example, to an IPC. These installation requirements must be met in order to successfully load the configuration and enable you to access the PC system with STEP 7 online. STEP 7 shows the applicable (minimum) installation requirements for a selected PC system in the "PC System" area of the Inspector window.

Example: Only WinCC RT configured

If only WinCC RT is configured, "not installed" must be shown in the PC system in the sense of "no PC Station installed".

Therefore, STEP 7 displays "not installed".

Example: WinAC RTX configured

If the WinAC RTX Software Controller is configured, SIMATIC WinAC RTX 2010 must be installed (referred to as "PC Station V1.0" below) on the PC system. SIMATIC WinAC RTX 2010 includes PC Station or S7-RTM, referred to as "PC Station V1.0" in the following, as an installation component.

STEP 7 automatically sets this version of PC Station once a WinAC RTX has been configured.

Example: CPU 1507S Software Controller configured

If the CPU 1507S Software Controller is configured, the SIMATIC S7-1500 Software Controller must be installed on the PC system. This includes PC Station installation component, referred to as "PC Station V2.x" below.

STEP 7 automatically sets this version of PC Station once a SIMATIC S7-1500 software controller has been configured.

- PC Station V2.4 (for Software Controller V2.6 as of TIA Portal V15 SP1)
- PC Station V2.3 (for Software Controller V2.5 as of TIA Portal V15)
- PC Station V2.2 (for Software Controller V2.1 as of TIA Portal V14 SP1)
- PC Station V2.1 (for Software Controller V2.0 as of TIA Portal V14)
- PC Station V2.0 (for Software Controller V1.x as of TIA Portal V13 SP1)

Reference

You can find additional information about the installation requirements of the relevant PC Station version in the section Loading with STEP 7 (Page 1439).

Special features of the diagnostics in the PC System V2.x

Special features of the diagnostics in the PC System V2.x

The diagnostics concept for a PC System V2.x has the same hierarchical structure as that of a modular CPU. You can see at a glance whether the station is free from faults or if, for example, an underlying component has a fault.

There is only one difference: They have one hierarchical level more than modular CPUs; the PC system is on a level higher than the Software Controller. The PC system has its own hardware status (separate diagnostics).

The following figure shows the diagnostics of the complete PC system in the project tree. The PC station itself is OK (green check mark), but has an underlying component with a fault (exclamation mark in a green check mark). The symbol provides preliminary information about disrupted or missing modules in the underlying structure.

To obtain a more detailed description of the error, double-click on the "Online & Diagnostics" entry in the project tree.

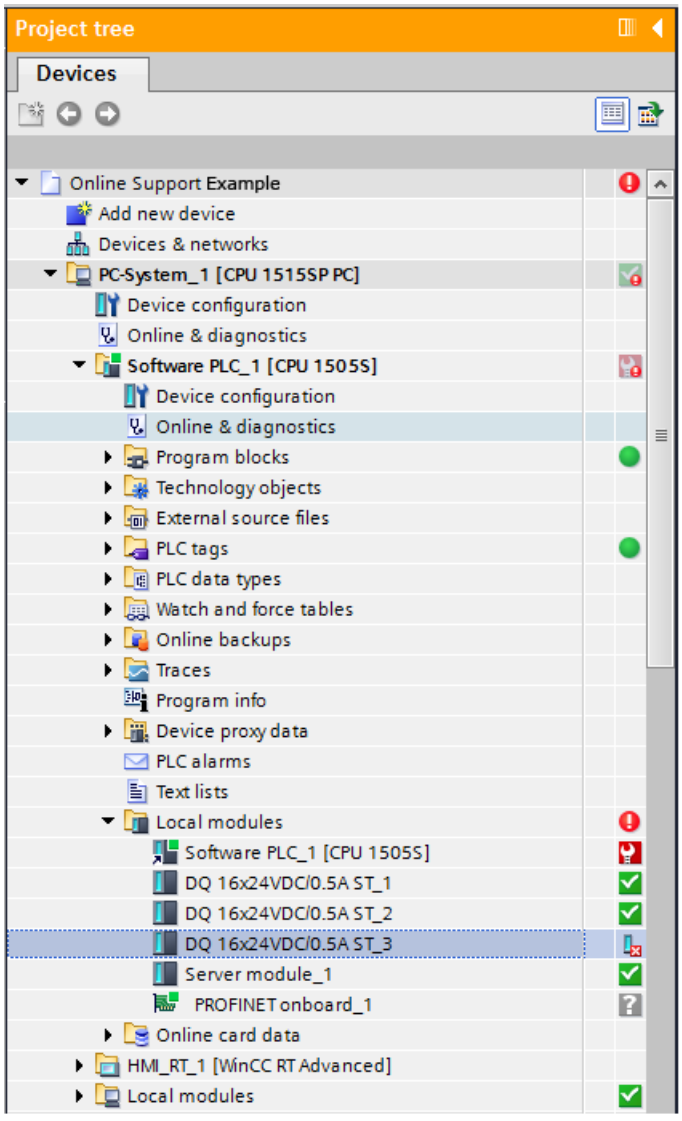

Figure 1-34 Displaying faults in the project tree

The following figure shows the two entries in the project tree for starting the online and diagnostics view of the PC system. For details, refer to the figure's legend.

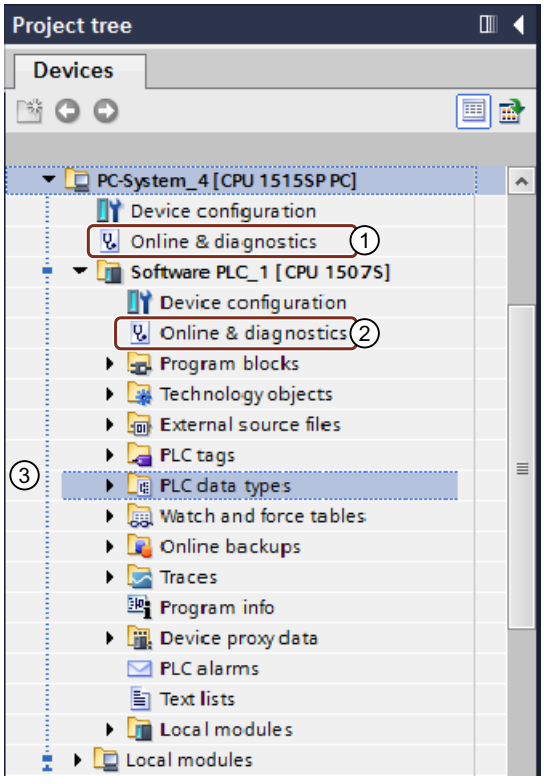

① Hardware diagnostics of the PC system (for example, fans, temperatures, voltages for IPCs)

② Standard diagnostics (as for S7-1500 CPU)

② Visual brackets for the components of the PC station

Figure 1-35 "Online & Diagnostics" project navigation

Additional information

You can find additional information on "Diagnostics" and the legend with the meaning of each symbol in the Diagnostics (http://support.automation.siemens.com/WW/view/en/59192926) function manual.

Loading with STEP 7

Loading PC station V2.x with STEP 7

The special features for loading a PC System V2.x with STEP 7 are listed below.

Requirement

- The corresponding software components are installed on the PC system (e.g. SIMATIC S7-1500 CPU 1507S).
- If needed, also install the corresponding version of the Setup DVD for SIMATIC WinCC Runtime Advanced on the PC system.

- The SIMATIC IPC hardware component is physically connected via Ethernet to the PC on which STEP 7 is installed.
- The Software Controller is started.

Other settings on the PC station are not required, because PC System V2.x is fully configured by loading the project.

Note

Recommended interfaces for download

- Use a "PN/IE" interface to load a SIMATIC IPC, which is not to be assigned to the Software Controller as a PROFINET interface.
  The "X1" interface is recommended.
- Use the "X2" interface to load a CPU 1515SP PC.

Procedure

To download the STEP 7 project, follow these steps:

1. Select the PC system in the device view.
2. Select the "Download to device" shortcut menu command.
   The "Extended download to device" dialog opens.
3. Configure the settings for the interface.
4. Click the "Download" button to start the download.
   If the PC system is not executed in the "Extended download to device" dialog, this may be due to the lack of an interface assignment.

Result

The project is downloaded step-by-step. A dialog shows the download progress.

- If you have made changes to the PC system configuration (for example, interface assignment, LED, NVRAM or index), then the PC system configuration is downloaded first. If a restart is required after this, STEP 7 displays a corresponding message. You can stop the download at this point, for example, to first configure the Software Controller and then perform the download later. Configurations already loaded are only reloaded if they have changed.
- If you wait instead of stopping the download after the download of the PC system configuration, the PC system reboots and automatically loads the remaining components, such as the configuration of the Software Controller and WinCC Advanced.

Checking the result of the download

After successful completion of the download, the CPU link appears in the Windows Start menu with the name you have assigned in the CPU settings in STEP 7.

The name assigned in STEP 7 is also visible in the CPU display.

Loading PC station V1.0 with STEP 7

The special features for loading a PC System V1.0 with STEP 7 are listed below.

Requirement

- The SIMATIC WinAC RTX DVD is installed on the PC system.
- Each device to which you want to load is accessible via online access.
- The PC station (runtime PC) is linked to the configuration station over a network. The PC station can be reached as communication station, for example, by setting the parameters in the initial configuration.
- An existing connection is checked, for example, with the "Accessible devices" function.
- The target PC station must be configured with the Station Configuration Editor.

Transfer of the configuration

You have the following options for transferring a created configuration to the PC station:

- Online operation
  - Networked (engineering station networked with Runtime PC); access point: "S7ONLINE (STEP 7)"
  - Local (configuration station same as PC station); access point: "PC internal (local)"
- Offline operation (separate engineering station and runtime PC) - Create and import XDB file and WinAC program file (*.wld)

Online operation

Online operation enables direct loading of the configuration data:

- Loading over a PC station connected to a network (MPI, PROFIBUS or Ethernet)
- Loading to the local PC station that is also used as configuration station

Procedure: Online operation - networked

1. Select the station to be loaded in STEP 7.
2. Load the configuration data with "Online" > "Download to device".

Note

For programming device operation

Make sure that you set the correct interface with "Set PG/PC Interface" (access point "S7ONLINE (STEP 7)").

Procedure: Online operation - local

The configuration data is transferred over an internal PC connection in this case.

1. Select the station to be loaded in STEP 7.
2. Load the configuration data with "Online" > "Download to device".

Offline mode

A file of the type XDB is required for data transfer in this operating mode. The configuration system generates an XDB file for each configured PC station.

The storage location of the configuration file is available in the properties of the "PC Station" object in the configuration system. This file can be imported with the "Station Configuration Editor" on the PC station.

You can find more information on offline information in section "Loading PC station V1.0 via storage medium (Page 1442)".

Reference

Additional information can be found in the "NET PC software Commissioning PC Stations - Manual and Quick Start (https://support.industry.siemens.com/cs/de/en/view/109488960)".

Load via file

Load PC Station V1.0 via file

The particularities of loading a PC Station V1.0 via storage medium are listed below.

To load the complete configuration, you have two options:

- Option 1: Create and import a configuration using an XDB file (Page 1442)
- Option 2: Create the program with the help of the memory card and load it to the Software Controller (Page 1443)

Create and import a configuration using an XDB file

Note

WinAC RTX cannot be installed.

Procedure

Follow these steps to configure a PC station offline using an XDB file and then importing it:

1. Drag a device for a PC station into the network view from the hardware catalog under "PC systems".
2. Select the PC station.
3. Change the name of the PC station to suit your needs under "Properties > General" in the Inspector window.

4. Drag-and-drop the additional components required into the PC station from the hardware catalog. An added component is inserted on a new index.

   Note

   You can change the index of a selected component in the Inspector window under "Properties > General > Position number". There may be gaps in index numbering for this reason. These gaps are not displayed in the graphic representation of the device view. However, gaps are displayed in the tabular area of the device view.

5. Select the PC station.
6. Under "Properties > XDB configuration" in the Inspector window, select the "Generate XDB file" check box.
   The connection data and addresses for CPs and applications are saved in the XDB file.
7. Specify the storage path under "XDB file path".
8. Select the PC station.
9. Select the shortcut menu command "Compile > ...".
   When the configuration of a PC station is saved and compiled, system data and the XDB configuration file are generated. This can then be loaded into the target system or be installed.

   Note

   The location of the configuration file is set on the target PC station via the program "Set PG-PC interface" (in the "STEP 7 Configuration" tab).

10. Copy the XDB file to a storage medium that can be accessed by the target station.
11. Import the XDB file to your target PC station with the Station Configuration Editor.

Create the program with the help of the memory card and load it to the Software Controller

Procedure

The configuration data, the current values of the DBs and the STEP 7 user program can be archived, reused and passed on in wld files. The log file (*.wld) does not save the configuration of the PC station in the Station Configuration Editor.

The controller must be in STOP mode for this to take place.

To create a log file, follow these steps:

1. Select the "New..." command in the "Project > Memory card file" menu.
   A dialog opens.
2. Select the directory in which you want to create the file.
3. Enter the file name and the path.
4. Confirm with "Create".

   Note

   You can also create a log file in WinAC with the menu command "File > Archive from CPU".

5. Select the "Open..." command in the "Project > Memory card file" menu to display the created log file in the project tree.
6. Select the directory in which the log file with the file extension (*.wld) is located.
7. The log file is displayed in the project tree under "Card Reader/USB memory".
8. Copy the wld file to a storage medium that can be accessed by the target station.
9. To restore the log file, select the menu command "File > Download to CPU".
10. Select the log file with the extension *.wld that you want to restore.
11. Click "OK" to confirm. The STEP 7 user program and the configuration for the Software Controller are loaded again.

Loading PC Station V2.x with file

The possibility to save and transport the system configuration of the PC system in a configuration file, offers the following advantages:

- Update of large plants without the TIA Portal
- Simple provision of program and configuration updates
- Plant-level update no longer necessary
- No special software required

Create configuration file

Create configuration file

The entire configuration of your PC station is saved in a configuration file from the TIA Portal. The data may be re-used and distributed. The configuration file has the extension *.psc.

To create a configuration file, follow these steps:

1. Select the menu command "Project > PC Station configuration file > New > PC Station configuration file".
2. Enter the file name in the "Create memory card file" dialog that opens. To avoid error messages, make sure that the entries are correct:
   - Use short, unique name
   - Name may not contain more than 255 characters
   - Name may not contain spaces
   - Only use permitted characters; these are letters and digits, and the special characters '-' and '_'.
3. Select the desired directory in which you want to create the file. To avoid error messages, also make sure that the entries are correct, as in 2.
4. Confirm with "Create".

NOTICE
Secure data against access by third parties The customer is fully responsible for the secure transport of data.

Result

The "Memory card file" folder is created in the project tree under "Card Reader/USB memory" with the following structure:

- PC system configuration file
  This file contains the PC system configuration file. The information indicates the file name and path information, for example: Drive:\PC-SystemConfiguration01.psc
  - "Display PC system info" icon
    Double-click on the icon to display all project, device and module information about the configuration loaded. If more data is loaded, you can use the "Update" button to display the latest metadata.
  - Folder with station name already assigned in the project navigation, for example, PC-System_1.
    This folder contains the configuration of the PC system.

Loading project data to configuration file

To load data into the PC system configuration file, you have the following options:

- Load project data to a memory card using Drag&Drop or Copy&Paste
- Write project data to a memory card

Requirement

- A PC system with a PC station V2.2 or higher is configured in the STEP 7 project.
  For Failsafe, a PC system with a PC station V2.3 or higher is configured.
- A *.psc file is created and opened in the project tree.
- If you use a CPU1515SP PC2, Microbox, Nanobox "E", IPC547G or an IPC xxxD in your configuration, this is only possible with a PC station version ≥ V2.2.

  Note

  Special feature

  An individual software CPU cannot be copied to the *psc file.

Please also note the information on configuration file import in the Failsafe manual "Configuring and programming Safety" (https://support.industry.siemens.com/cs/ww/en/view/54110126).

You will find an application example for "Loading project data to configuration file" on the Internet (https://support.industry.siemens.com/cs/ww/en/view/109759142).

Loading project data to a memory card file

To load project data to a memory card file, follow these steps:

1. Drag the project data that you want to load from the project tree to the memory card.
   If necessary, the project data is compiled.
2. Then, the "Load preview" dialog opens. This dialog displays messages and recommends actions necessary for loading.
3. Check the messages and enable the actions in the "Action" column if necessary. As soon as loading is possible, the "Load" button is enabled.
4. Click the "Load" button.
   The loading is performed.

or:

1. Select the "PC system" folder in the project tree.
2. Right-click on the selection and select the "Copy" command from the shortcut menu. Alternatively, you can also use the shortcut <Ctrl+V>.
3. Right-click on the "*.psc" file level or "PC system" folder in the memory card file and select the "Paste" shortcut menu command. Alternatively, you can also use the shortcut <Ctrl+V>. All other levels are locked. If necessary, the project data is compiled.
4. Then, the "Load preview" dialog opens. This dialog displays messages and recommends actions necessary for loading.
5. Check the messages and enable the actions in the "Action" column if necessary. As soon as loading is possible, the "Load" button is enabled.
6. Click the "Load" button.
   The loading is performed.

or:

1. Select the "PC system" folder in the project tree.
2. In the "Project" menu, select the command "Card Reader / USB memory > Write to memory card".
   The "Select memory card Select" dialog opens.
3. Select a memory card.
   A button with a green check mark is enabled in the lower part of the dialog.
4. Click on the button with the green check mark.
   If necessary, the project data is compiled.
5. Then, the "Load preview" dialog opens. This dialog displays messages and recommends actions necessary for loading.
6. Check the messages and enable the actions in the "Action" column if necessary. As soon as loading is possible, the "Load" button is enabled.
7. Click the "Load" button.
   The loading is performed.

Result

The psc file contains the configuration for all components in corresponding subfolders. The name of the folder is changed to the current PC system name.

Note

Check that file is complete

Check the psc file in your TIA Portal to make sure it is complete, for the file can only be edited in the TIA Portal.

Open existing configuration files

Open configuration file

To view a configuration file in the project tree, follow these steps:

1. Select the menu command "Project > PC Station configuration file > Open > PC Station configuration file".
2. Select the directory containing the psc file.

The memory card file appears with the mentioned content under "Card Reader / USB memory" in the project tree.

Import configuration file on the target system

The following lists the tools with which you can commission the target system without TIA Portal:

- Command line tool
- PC Station display in the information area

Requirements

- You have administrator rights.
- A Software Controller, version 2.5 or higher, is installed.
- Access to the Open Controller CPU 1515 SP PC2 and/or an IPC is possible.
- The psc file is located in a local directory or is available on a medium such as a USB flash drive.
- For Failsafe: You are a member of the Windows "Failsafe Operators" user group.

Overview of the supported command line commands

The following table provides an overview of the supported command line commands:

Command	Explanation
PCSystem_Control.exe /Help	Displays the help text in the command line editor.
PCSystem_Control.exe /HelpExitCode	Displays the help text for the error codes in the command line editor.
PCSystem_Control.exe /PrintConfig <file.psc*>	Displays the information about the components; output format: Standard
PCSystem_Control.exe /PrintConfig <file.psc*> /xml	Displays the information about the components; output format: XML format
PCSystem_Control.exe /ImportConfig <file.psc*>	The call starts the full import of the psc file.
PCSystem_Control.exe /GetStatus /ImportConfig	Displays the current status of all the components, the import process; output format: XML format
PCSystem_Control.exe /GetStatus /SimaticComponents	Displays information on all installed Simatic components; output format: XML format

* The <file.psc > parameter stands for the full path and file name, for example, C:\Data\Test.psc

Result

The data is imported in the background.

If a restart is configured, this is performed automatically once the import is completed. If software CPU is contained in the imported configuration, it remains in STOP after the restart.

Procedure for error messages

If an error occurs with a command, you can determine the bit number of the error and to obtain an accurate description of it.

Proceed as follows:

1. To obtain the error code, enter the command "echo% errorlevel%" in the command line tool.
2. This command gives you the following result:
   - If error code = 0, there is no error.
   - If error code > 0, there is an error and it is output as a decimal number.
   - For Failsafe: Error code 20899 or 0x51A3, import successful.
3. The decimal number must be converted into a binary number.
4. To get an overview of the bit number describing the error, enter the command line command "HelpExitCode".

Example

Example of conversion of number systems:

- Decimal number: 288
- Binary number: 100100000

Bit number with the corresponding error description:

Result: You can read the error at bit number 5 "Err_Net_45_Full=1" and bit number 8 "Err_IIS_Running=1".

Web server

Settings of the Web server on the target system

The Web server functionality is available as of PC Station V2.1. Below you will find the settings and special features for using the Web server.

Requirement

The following software requirements are necessary to use the full range of functions offered by the Web server:

- PC Station ≥ V2.3:
  - Windows 10 IoT Enterprise (64-bit) is installed
- PC Station ≤ V2.2:
  - Windows 7 Embedded is installed
  - Windows Ultimate 7 including the Internet Information Services (IIS) ≥ V7.0 is installed (see section Installing the Internet Information Service (Page 1450))
- .NET version ≥ 4.5 is installed

Note

The software must be installed BEFORE installing the PC station.

Principle

The Web server is a Web application and part of the Microsoft Internet Information Server installation. During installation, the Web server integrates itself into the standard website under the pre-installed IIS with access via port 80.
The Web server is used when the URL address contains /simatic or /simatic/.

Example

Examples of how to reach the Web server:

- http://IP address/simatic/ or http://IP address/simatic
- http://computer name/simatic/ or http://computer name/simatic

Reference

You can find additional information on the Web server in the Web server function manual (https://support.industry.siemens.com/cs/de/en/view/59193560).

Enabling of Internet Information Services for Windows 7 and Windows 10

Enabling of Internet Information Services (IIS) for Windows 7

Below we will show you how to enable the Internet Information Services (IIS) and the required settings.

Note

The additional installation of the IIS is only necessary when Windows 7 Ultimate is installed on your computer.

Requirement

Windows 7 Ultimate is installed.

Procedure

Follow these steps to enable the Internet Information Services (IIS):

1. Click on the "Start" menu icon.
2. Select "System Control > Programs and Features > Turn Windows features on or off".
3. Confirm the message for user account control with "Yes".
4. Navigate to the menu command: "Internet Information Services" and make the following settings:

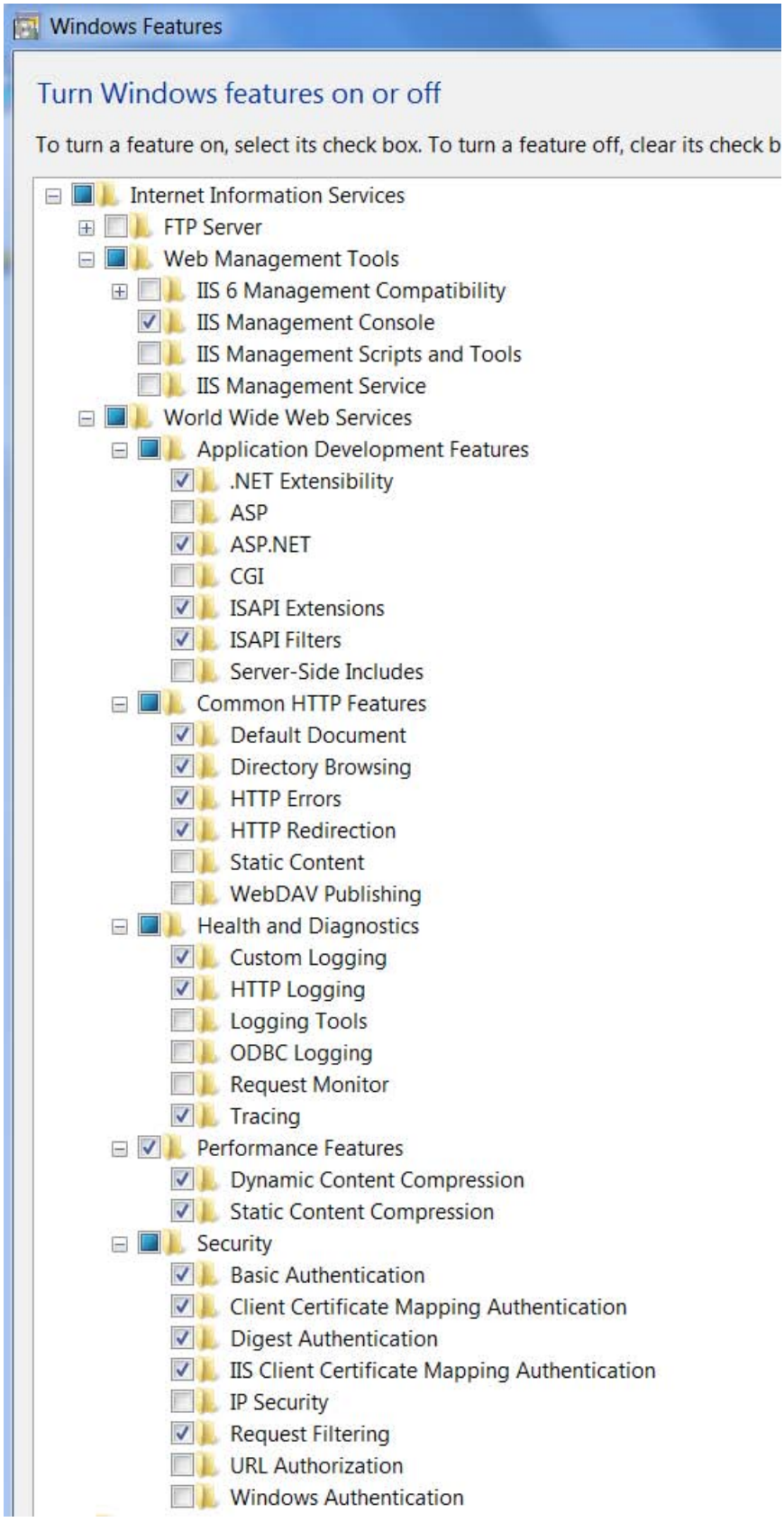
Windows Features
Turn Windows features on or off
To turn a feature on, select its check box. To turn a feature off, clear its check b
Internet Information Services
FTP Server
Web Management Tools
IIS 6 Management Compatibility
IIS Management Console
IIS Management Scripts and Tools
IIS Management Service
World Wide Web Services
Application Development Features
.NET Extensibility
ASP
ASP.NET
CGI
ISAPI Extensions
ISAPI Filters
Server-Side Includes
Common HTTP Features
Default Document
Directory Browsing
HTTP Errors
HTTP Redirection
Static Content
WebDAV Publishing
Health and Diagnostics
Custom Logging
HTTP Logging
Logging Tools
ODBC Logging
Request Monitor
Tracing
Performance Features
Dynamic Content Compression
Static Content Compression
Security
Basic Authentication
Client Certificate Mapping Authentication
Digest Authentication
IIS Client Certificate Mapping Authentication
IP Security
Request Filtering
URL Authorization
Windows Authentication

Figure 1-36 Settings for Internet Information Services

Problems when accessing the Web server

You can find possible causes and their remedies here:

- Internet Information Services were enabled after installation of .NET
- A 64-bit version of Windows is run on the computer

To remedy the problem, you must register a .NET version afterwards in the Internet Information Services (IIS). To do so, execute the following command in the Windows CMD window: C:\Windows\Microsoft.NET\Framework64\v4.0.30319>aspnet_regiis.exe –ir

Enabling of Internet Information Services (IIS) for Windows 10

Below we will show you how to enable the Internet Information Services (IIS) and the required settings.

Note

The additional installation of the IIS is only necessary when Windows 10 IoT Enterprise (64-bit) is installed on your computer.

Requirement

Windows 10 IoT Enterprise (64-bit) is installed.

Procedure

Follow these steps to enable the Internet Information Services (IIS):

1. Right-click on the "Start" menu icon.
2. Select "Programs and Features > Turn Windows features on or off".

3. Confirm the message for user account control with "Yes".
4. Navigate to the menu command: "Internet Information Services" and make the following settings:

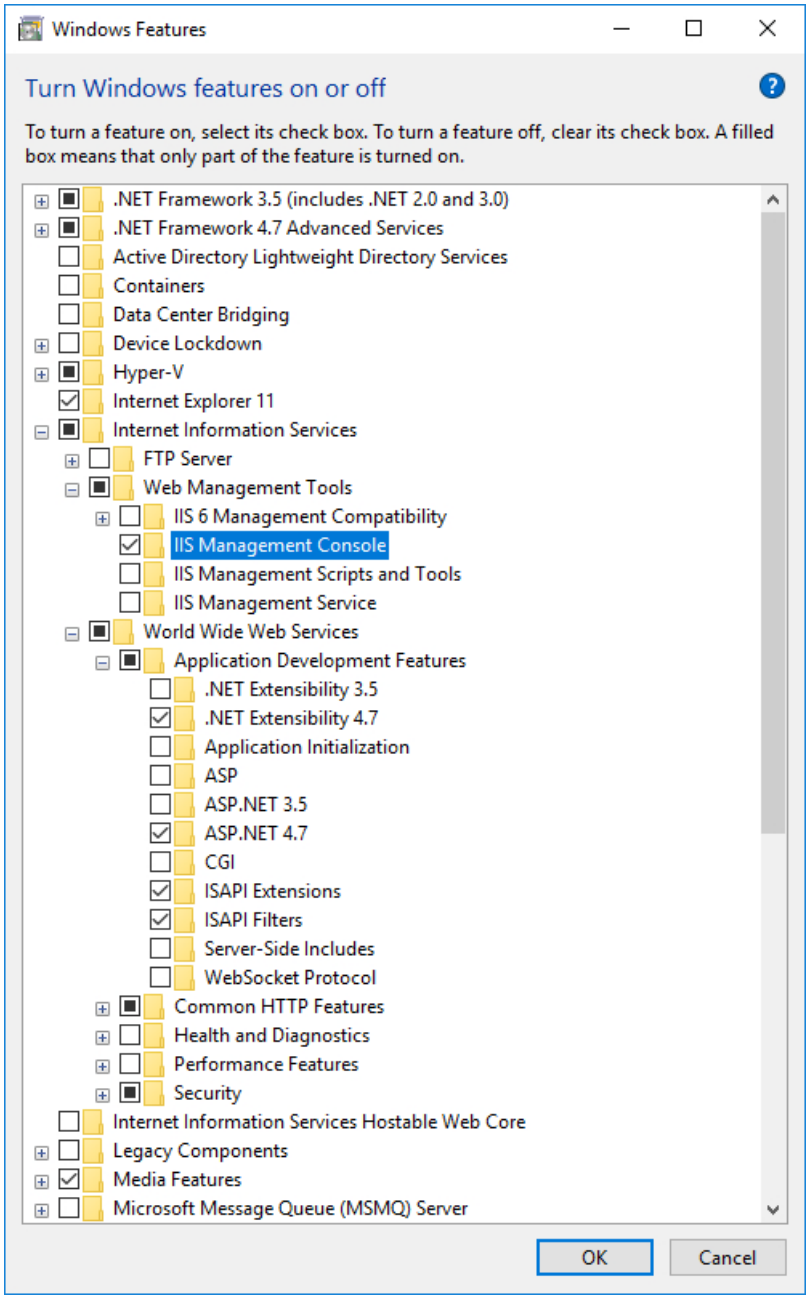

Problems when accessing the Web server

You can find possible causes and their remedies here:

- Internet Information Services were enabled after installation of .NET
- A 64-bit version of Windows is run on the computer

To remedy the problem, you must register a .NET version afterwards in the Internet Information Services (IIS). To do so, execute the following command in the Windows CMD window: C:\Windows\Microsoft.NET\Framework64\v4.0.30319>aspnet_regiis.exe –ir

Configuration of the Web server in the TIA Portal

The procedure for configuring the Web server in the TIA Portal is described below.

Procedure

To use the Web server, follow these steps:

1. Select the desired PC system.
2. Select the following settings in the Inspector window under "Properties > General > Web server":
   - Activate web server on this module
   - Enable automatic update
   - Create and manage users

Reference

You can find additional information on the Web server in the Web server function manual (https://support.industry.siemens.com/cs/de/en/view/59193560).

Setting up the Web server using command line commands

The following command line tool is installed when installing the PC system: PCSystem_Control.exe.

The tool enables you to query information over the Web server and perform the following tasks:

- Query and check the Web server registration
- Register a Web server
- Revoke Web server registration
- Subsequently change the Web server settings

Requirement

The user has administrator rights.

Overview of the supported command line commands

The following table provides an overview of the supported command line commands:

Command	Explanation
PCSystem_Control.exe/Help	Displays the help text in the command line editor.
PCSystem_Control.exe /HelpExitCode	Displays the help text for the error codes in the command line editor.
PCSystem_Control.exe/Help/RegWebserver	Displays the help text for this call.
PCSystem_Control.exe/Help/UnregWebserver	
PCSystem_Control.exe/Help/VerifyWebserver	
PCSystem_Control.exe/RegWebserver/	The Dispatcher Web server registers itself in the Internet Information Services (IIS).
PCSystem_Control.exe/RegWebserver/HTTP	The call expects an existing connection for the HTTP protocol to the standard port "80".
PCSystem_Control.exe/RegWebserver/HTTPS	The call expects an existing connection for the HTTPS protocol to the standard port "443".
PCSystem_Control.exe/RegWebserver/HTTP/HTTPS	The call expects an existing connection for the HTTP protocol to the standard port "80" and for the HTTPS protocol to the standard port "443".
PCSystem_Control.exe/RegWebserver/Sitename"..."	The call expects an existing website in the IIS with the transferred name.
PCSystem_Control.exe/UnregWebserver	The registration of the Web server in the IIS is undone.
PCSystem_Control.exe/VerifyWebserver	Displays whether the user can register the Web server in the IIS or if prerequisites have to be met.

Procedure for error messages

If an error occurs with a command, you can determine the bit number of the error and obtain an accurate description of it.

Proceed as follows:

1. To obtain the error code, enter the command "echo% errorlevel%" in the command line tool.
2. This command gives you the following result:
   - If the error code = 0, there is no error
   - If the error code > 0, there is an error and it is output as a decimal number
3. The decimal number must be converted into a binary number.
4. To get an overview of the bit number describing the error, enter the command line command "HelpExitCode".

Example

Here is an example for a conversion and a bit number associated with the corresponding error description:

- Decimal number: 288
- Binary number:100100000

Result: You can read the error at bit number 5 "Err_Net_45_Full=1" and bit number 8 "Err_IIS_Running=1".

Access via HTTPS

Requirements

The requirements for error-free HTTPS access are as follows:

- The current time-of-day is set.
- The firewall is configured appropriately.
- A valid certificate is installed.
  - If no certificate is installed, a recommendation is displayed that you should not use the page. You also have the option of creating your own certificate.

Input and example

To access the Web pages with the HTTPS protocol, enter the URL in the following format:

- https://ww.xx.yy.zz, where ww.xx.yy.zz
  Example input: https://192.168.3.141

You can recognize an encrypted connection by a lock symbol in the status bar of the Web page.

Reference

You can find additional information on how to create your own certificate in the section Creating your own certificate (Page 1456).

Creating your own certificate

If your company does not have a Certification Authority (CA), you can use the procedure described in this section. The key files are created with the "OpenSSL" program. If you have not yet installed OpenSSL on your PC, you can download and install the program for free from the following website: http://openssl.org/ (http://openssl.org/)

Procedure

To create your own certificate, follow these steps:

1. Open the "Internet Information Services manager" application.
2. Click on the computer name.
3. Double-click "Server Certificates" in the "Features" view.
   The "Server Certificates" directory opens.
4. Click "Create Self-Signed Certificate" in the "Actions" area.
5. On the "Create Self-Signed Certificate" page, enter a display name for the certificate in the "Specify a friendly name for the certificate" field.
6. Click "OK".
   The new certificate is displayed in the "Certificate Server" overview.
7. Switch back to the "Internet Information Services Manager".
8. In the "Pages" directory, select the website in which the new HTTPS connection shall be created.
9. On the right side, click on "Edit Page > Binding".
   The dialog box with the available Web pages opens.
10. Select the connection type "https" and click "Add new".
    The dialog with the connection properties opens.
11. Select the protocol type "HTTPS" and the port number 443; insert the certificate you created in the "SSL Certificates" selection box.
12. Click "OK" to create a new connection.

1.4.3.2 Integrating IPCs into the TIA Portal

In addition to control and visualization tasks, PC systems can also be used for the following applications:

- Data gateway to higher-level MES
- Plant-level data server
- Hardware platform for SCADA systems
- Industrial workstations (for example, at test stations)
- Process & data analysis on site

Integrating SIMATIC IPCs in TIA Portal

Integrating the SIMATIC PC station (IPC in TIA Portal) integrates a PC system in a plant configuration (TIA project), with the following advantages:

- Fast configuration
  - Configuration via drag-and-drop from the hardware catalog
  - Detailed display of all interfaces – suitable for configuration of SIMATIC IPC
  - Network-spanning view including display of IP addresses
  - Fine-grained status diagnostic network topology
- Simplified commissioning
  - Central overview of all IP addresses
    Excludes conflicts due to duplicate assignments
    Easy to find SIMATIC IPC (including host name and IP address) via "Accessible devices"
  - Status diagnostics within the TIA Portal
    Diagnostics of the Ethernet/PROFINET connection to the SIMATIC IPC from the configuration
    Simple and fast error localization, for example, topology errors in the network, inaccessible devices
    Port-granular diagnostics (including the ports assigned in the topology on the SCALANCE switch)
  - Overview of the entire network
    Comprehensive information on topology, structures and hierarchies
- Integration of hardware diagnostics for IPCs in the TIA Portal
  - Fine-grained status diagnostics of the network topology, for example, topology errors in the network, inaccessible devices
  - Preventive maintenance, for example, of hard disks or fans
  - Early reaction to violation of high/low limit of the permitted operating temperature
  - Central overview of all SIMATIC IPCs in a plant
  - Display of hardware diagnostics in the WinCC Diagnostic Viewer
    Display of PC diagnostics in the running plant
    Preventive maintenance

System requirements

To install a PC station in an IPC, the following requirements must be met:

Requirements

- You have administrator rights on your computer.
- A minimum of 400 MB space is available.

Hardware requirements

The following IPCs are supported:

- SIMATIC IPC2x7E
- SIMATIC IPC4x7D
- SIMATIC IPC4x7E
- SIMATIC IPC547G
- SIMATIC IPC6x7D
- SIMATIC IPC8x7D

Note

Make sure that the IPCs you use in your configuration are compatible.

If you ignore the error message about your configuration, diagnostics errors in TIA Portal can result.

Always select the appropriate IPC for your configuration.

Software requirements

- The SIMATIC IPC software can be used with the following operating systems:
  - Windows 10 Enterprise IoT (for CPU 1515SP PC2, all IPCs; exception: IPC4x7D)
  - Windows Embedded Standard 7 E/P SP1 32-bit
  - Windows Embedded Standard 7 E/P SP1 64-bit
  - Windows 7 SP1 32-bit
  - Windows 7 SP1 64-bit
- The following diagnostic software is installed:
  - DiagBase >= V1.5.0 or
  - DiagMonitor >= V4.5.0

Restrictions

- The following filters are disabled:
  - UWF (for CPU 1515SP PC2, all IPCs; exception: IPC4x7D)
  - EWF
  - FBWF
- The following components may not be installed:
  - PC Station classic (S7-RTM)
  - CPU 150xS
  - SIMATIC NET
  - WinAC RTX

Installation

Storage media for the installation

Setup can be installed from the following media:

- USB stick
- DVD
- Hard disk
- Network drive

Installation

Setup is started by running "start.exe" in the Setup folder.

Proceed as follows for this:

1. Double-click on "start.exe".
2. Select the desired product properties for the installation.
3. Follow the instructions on the screen during the installation.

Result

After a successful installation, the following components have been installed:

- PC Station V2.x
- S7 device drivers
- SIMATIC NET CP Manager Plus
- PC Station Web server

In addition, an icon for the PC Station service is also displayed in the notification area of the Windows task bar during operation of the CPU.

Reference

Additional information on the installation requirements of configured components is available in the section Procedure for configuring a PC system (Page 1435).

PC station display in the notification area

An icon for the PC Station service is also displayed in the notification area of the Windows task bar during operation of the CPU. Among other things the icon indicates the current state of the PC Station service, and provides you with the opportunity to do configurations.
Right clicking the icon in the notification area opens the PC Station shortcut menu.

States of the notification area icon

The state of the icon for the PC Station service in the notification area of the task bar changes as soon as the mode of the PC station changes.
The notification area icon can display the following states:

RUN	STOP

Configuration options using the "Station Manager" service's icon

The icon for the PC Station service in the task bar's notification area gives you the following configuration options via the shortcut menu:

- Import > Import Configuration
  This shortcut menu command opens and displays the metadata of a selected PC system configuration file. The user also starts the import of the psc file with this. The feature is not supported for F-CPUs.
- Configuration
  - Delete existing configuration
    Administrative rights are required for this configuration option.
    This shortcut menu command causes the configuration to be deleted.
    This function is disabled by default in an F-CPU. The "Failsafe Operators" user group must be added explicitly authorized in order to provide them with permission on an individual basis. Furthermore, the configured passwords for an F-CPU are retained after the configuration is deleted. These passwords are required later again for downloading. To reset the password, load a project without passwords.
  - Change the storage path of configuration data
    Administrative rights are required for this configuration option.
    If you are protecting a partition with an enhanced write filter (EWF), for example, the configuration and diagnostic data is also thereby write-protected.
    Save the diagnostic data in an area of the hard disk that is not write-protected. You can also save the configuration data containing the configuration in a section of the hard disk that is not write-protected.

- Restart all of the PC Station services
  Administrative rights are required for this configuration option.
  This shortcut menu command causes all of the PC Station services to be restarted.
- Exit
  This shortcut menu command causes the PC Station panel to be closed. The icon for the PC Station service in the notification area of the task bar is hidden. Restart the PC Station panel using the following entry in the Windows Start menu: **Siemens Automation > SIMATIC > PC Station > "PC Station"**

Repairing/uninstalling

You have the option to repair or uninstall the installed program.

Repairing

To repair the program, proceed as follows:

1. Navigate to the path where your Setup is installed.
2. Double-click on "start.exe".
3. Select "TIA integration for SIMATIC IPC" by placing a check mark and click "Next".
4. Select the "Repair" button.
5. Click "Next" and follow the instructions on the screen.

Uninstalling

To uninstall the program, you have two options.

Via the Control Panel

1. In the Windows taskbar, select **Start > Control Panel > Run Advertised Programs**
2. Select the program "TIA integration for SIMATIC IPC".
3. Click "Uninstall".
4. Follow the on-screen instructions during the uninstall process.

Via "start.exe":

1. Navigate to the path where your Setup is installed.
2. Double-click on "start.exe".
3. Select "TIA integration for SIMATIC IPC" by placing a check mark and click "Next".
4. Select the "Uninstall" button.
5. Click "Next" and follow the instructions on the screen.

Restrictions for cancellation

When the setup is aborted, all installation steps must be carried out again.

1.4.3.3 Basic configuration of the Open Controller

Creating the configuration

You have created a new project in the TIA Portal.

To create the configuration in the TIA Portal, follow these steps:

1. Double-click "Add new device" in the project tree.
2. Select "CPU 1515SP PC (+HMI)" under "PC Systems > Open Controller".
3. Select the appropriate version.
   The configured Open Controller is displayed in the device view.
4. The following components can be seen in the Open Controller:
   – Onboard interface X2 (GB Ethernet Windows interface) that is assigned directly to the PC station (1 port)
   – Exchangeable BusAdapter X1 that is assigned directly to the Software Controller (2 ports)
   – On the right side (behind the plugged modules in the rack): the configured CPU 1505S Software Controller and WinCC Runtime Advanced.
5. Insert the server module.
   The server module at the right end of the configuration forms the termination of the CPU with the I/O modules.

Setting the IP addresses:

In the Inspector window under "Properties":

- Onboard interface X2: The configured, preset IP address is identical to the Windows IP address that is set on the Open Controller during initial commissioning.
- BusAdapter X1: The configured, preset IP address is identical to the IP address that was set in the panel of the Software Controller (display application) during initial commissioning.

Important properties of the Software Controller

If necessary, you can change the properties in the Inspector window under "Properties".

- Selecting startup type
- System diagnostics
- Setting the storage location for retentive data
- Setting up copy protection
- Using the LEDs of the hardware
- Configuring the Web server
- Assigning interfaces for the communication

Establish HMI connection

1. Right-click the WinCC RT Advanced in the device view.
2. Start the HMI device wizard.

3. Apply all default settings.
   The wizard creates system images for you with the corresponding navigation.
4. Switch to the connection view in the network view.
5. Click on WinCC RT Advanced.
6. Draw a line from the WinCC RT Advanced software to the S7-1500 Software Controller (e.g. CPU 1505SP) while keeping the mouse button pressed.
   A network connection is established between the two devices.

Downloading the project to the target system

The complete PC station is downloaded the first time you download the configuration.

1. Select the complete PC system in the device view.
2. Click the "Download" button.
3. Make the following settings:
   - Type of connection
   - Specify the interface of the programming device
   - Specify the X2 interface on the Open Controller

Note

The first TIA Portal download **must** take place over the "X2" interface.

Only ASCII characters are permitted in the name of the interface X2 PN/IE (LAN) in the TIA Portal, e.g. PROFINET_2.

Download and compile the project. The hardware configuration and the first download are now completed.

WinCC RT Advanced

WinCC Runtime Advanced contains all the essential functions for operator control and monitoring of machinery or plants.

Reference

You can find additional information on configuration and downloading in the ET 200SP Open Controller CPU 1515SP PC (https://support.industry.siemens.com/cs/de/en/view/109248384) manual.

You can find more information on WinCC RT Advanced in the WinCC Advanced V13.0 SP1 (https://support.industry.siemens.com/cs/de/en/view/109091876) system manual.

1.4.3.4 Parameter assignment

Startup of the Software Controller (startup parameter)

Configuration of the startup type of the CPU

With a CPU 150xS, you have the option of assigning parameters for the startup characteristics in addition to the standard startup parameters, for example "Startup after POWER ON". The CPU can be started (POWER ON) in two different ways:

- Manual start via the "Power" button on the CPU display
- Automatic start during PC start (recommended)

Using the "Automatic start after booting the PC" option in the properties of the software controller, you determine how the CPU starts.

We recommend you enable the option "Automatic start after booting the PC", otherwise the software controller does not automatically boot after the PC system starts. A software controller can be only loaded after it has started.

If this option is disabled, you must start the CPU manually before a download procedure via the CPU display.

Note

BIOS memory test on SIMATIC IPCs

PCs provide the option of a memory test. Some hardware tests, such as the memory test, are disabled by default in the BIOS setup program and are skipped during startup of the PC. Booting is accelerated as a result.

If you are using the CPU on a SIMATIC IPC or CPU 1515SP PC, the BIOS memory test must not be enabled.

The default startup parameters are no different from the startup parameters of S7-1500 CPUs.

Setting the startup type

To set the startup type, follow these steps:

1. Start STEP 7.
2. Open your project.
3. Change to the project view.
4. Open the device view.
5. Select the CPU.
6. On the "Properties" > "General" tab of the Inspector window, select the "Startup" area.

7. Configure the startup characteristics of your CPU.

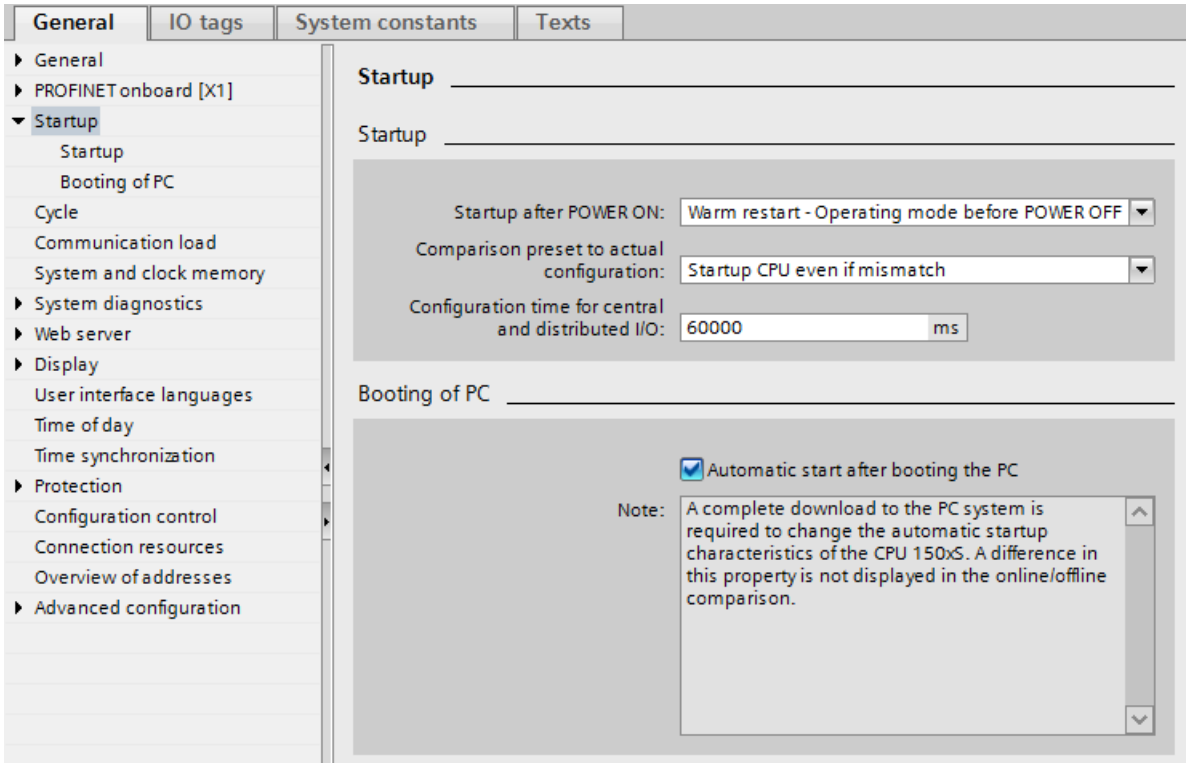

8. Download the project to the CPU, by selecting the PC system before you start the download.

Result

If, in addition to the settings for the startup type, you also change the "Automatic start after booting the PC" option, the CPU is stopped automatically before the download. At the beginning of the download, the CPU starts again in STOP mode.

The project is downloaded. The new settings for the startup type are active.

Special features in the parameter assignment of the access protection

Assigning access protection parameters in STEP 7

The access protection parameters are assigned using the properties of the PC system in which the software controller is inserted.

The access levels are read-only in the properties of the CPU. To access the properties of the PC system, click the arrow. Here you can define the access levels of the PC system and overwrite them for the entire application.

Note

Assigning access protection parameters for the entire PC system

Unlike for a hardware CPU, parameter assignment for access protection is not done directly in the CPU's properties. This ensures that consistent protection level passwords are configured for all of a PC system's components.

Unlike a hardware CPU of the S7-1500 automation system, a separate display password cannot protect the CPU against unauthorized access . Because the CPU can also be controlled by remote access, it uses the access protection passwords from STEP 7 to ensure access protection for the display.

Reference

For additional information on the protection concept in conjunction with display access, refer to theSIMATIC S7-1500 CPU 150xS (https://support.industry.siemens.com/cs/de/en/view/109249299) operating manual.

Advanced configuration for Software Controllers

If you select an S7-150xS software controller in the device view or network view, click on the "Advanced configuration" area in the Inspector window. The parameters under "Advanced configuration" are explained below or you can find references to explanations for these parameters.

PC communication interface

The PC communication interface offers access, for example, to an office network via a Windows-controlled interface independent of a PROFIBUS or PROFINET subnet. Your user program uses this interface to communicate with other devices or fetch information via the web server of the software controller:

Accessing the web server

Similar to the hardware interface, you also need to enable access via the PC communication interface to be able to access the web server of the controller software through this interface.

The following figure shows the web server access from the web browser of the PC system via the PC communication interface.

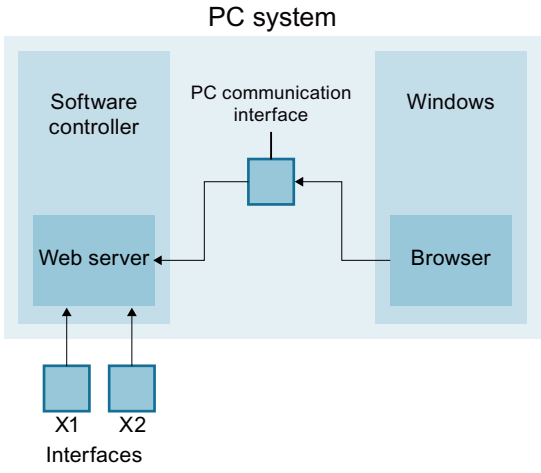

Communication via Open User Communication (OUC)

You can also use Ethernet interfaces on the PC via the PC communication interface. Basically, only the name of the host interface and the hardware ID is displayed for the configuration.
In the properties of a communication instruction (e.g. TSEND_C), you can access the properties of the PC communication interface and use the "Connection data" area to access the address parameters for the programmed OUC communication. The hardware ID of the PC communication interface is entered in the connection data.

Index

The index of a component in the PC system corresponds to the position of the component (for example, a hardware resource as an interface). It is a sort of "slot address on the PC-internal subnet" (Softbus).

The index of a software or hardware component is not displayed in the graphical view.

When a software controller is selected in the network view or device view, you can read or change the index in the properties of the software controller under "Advanced configuration". You can also change the index in the device overview table in the device view.

Reference

You can find information on using the hardware LEDs here (Page 1470).

You can find information about setting the storage location for retentive data here (Page 1469).

Setting the storage location for retentive data

The S7-1500 Software Controller (CPU) provides the option of storing data retentively in the PC mass storage or in the integrated NVRAM when the CPU is stopped or a power failure occurs. You set the type of data storage in the CPU properties.

Note

Data loss when changing the storage type

When you change the storage type, the current retentive data and the contents of the diagnostic buffer are deleted when the changed configuration is loaded.

Requirement

The project view is open.

The device view is open.

Procedure

To configure the type of storage, follow these steps:

1. Select the CPU.
2. Select the "Advanced configuration" area on the "Properties" tab of the Inspector window.
   - Select the "PC mass storage" option button to store the retentive data in the mass storage of your PC.
   - Select the "NVRAM of PC platform" option button to save the retentive data in the integrated NVRAM of your PC.

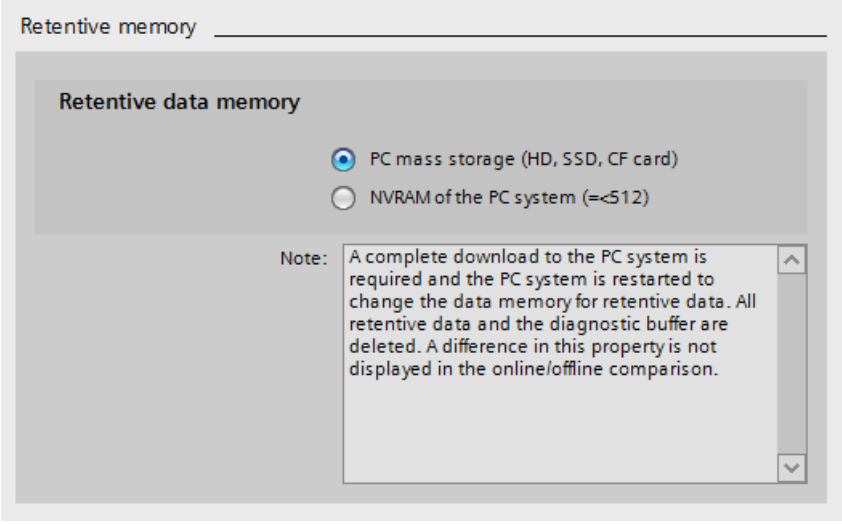

3. To change the type of data storage in STEP 7 at a later time, download the project to the PC system again.

Note

Downloading the change of configuration

When you change the storage type for storage of retentive data in STEP 7, you must download the entire PC system for this configuration change. When you change the storage type, the CPU reboots during the download process. Without a reboot the storage type is not changed despite the download process.

To ensure the required restart of the CPU, you have to load the PC and then restart it.

See also

Advanced configuration for Software Controllers (Page 1467)

Using the LEDs of the hardware

The S7-1500 Software Controller (CPU) can display its status on the LEDs of the hardware platform on which it is installed. You set this functionality in the CPU properties in STEP 7.

Note

Simultaneous access by multiple components

Take care that multiple competing components (for example DiagBase and CPU) do not make simultaneous access to the hardware LED.

Requirement

Your project is open.

You are in the project view.

Procedure

To use the LEDs of the hardware platform for the CPU, follow these steps:

1. Open the network view or device view.
2. Select the CPU.

3. Select the "Advanced configuration > Hardware LED" area on the "Properties" tab of the Inspector window.

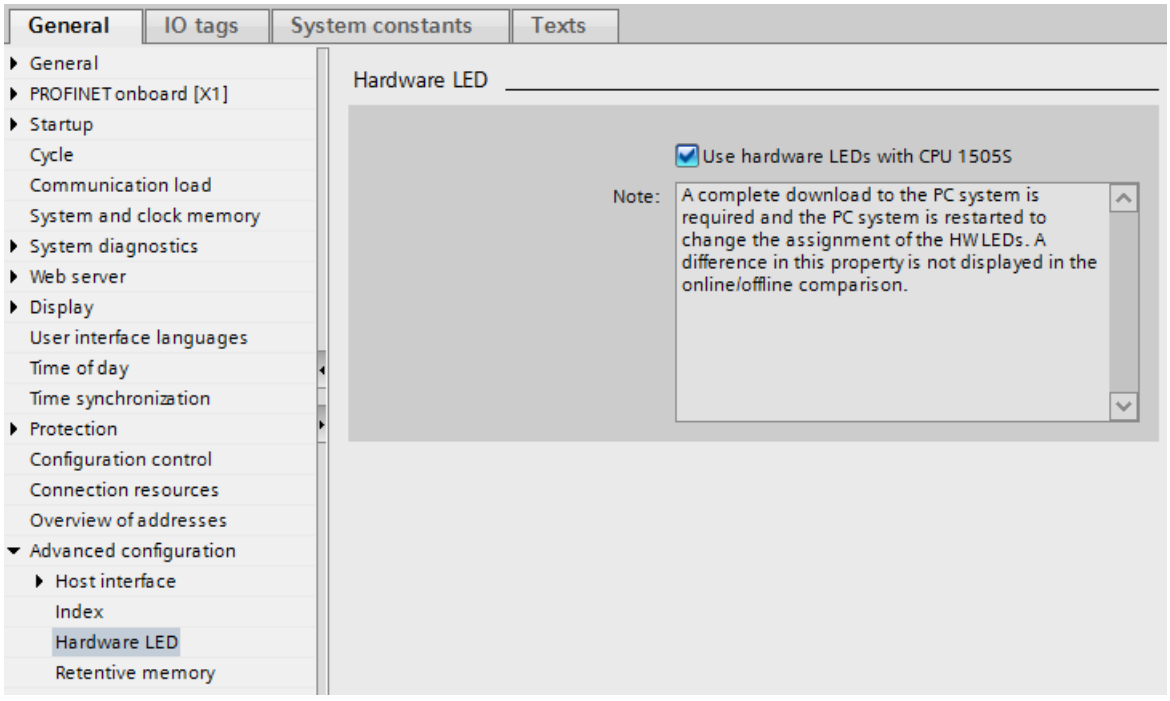

4. Select the "Use hardware LEDs with CPU 150xS" option.

As described in the note below the option, you then have to fully download the PC system and reboot it.

Result

The PC is shut down and the CPU is stopped. The project is downloaded.

The PC is restarted automatically. The CPU starts in STOP mode. The download is continued. After completion of the download, the CPU is put into RUN mode.

Special feature

If there is an online/offline difference with regard to the "Use hardware LEDs" property, this difference is not shown in the online/offline comparison.

See also

Advanced configuration for Software Controllers (Page 1467)

Important information about assigning interfaces

As of STEP 7 V13 SP1, relationships between a software component, such as a Software Controller, and its interfaces to the outside world, e.g. a PROFINET interface, are represented by reference lines.

Representation of relationships

If nothing was selected, all interfaces and the software components are connected by a gray line. The lines have a square connector at the end. These lines show the relationship between software components (e.g. S7-1500 Software Controller, WinCC RTX or OPC server) and interfaces (PROFINET, PROFIBUS, ...).

When you select an interface, the assigned software components can be recognized by the blue line that is clearly distinguishable from the gray lines.

Interfaces without relation lines or with gray reference lines are not assigned.

Example 1: Assigning an interface to the Software Controller (CPU)

A blue reference line between a selected interface and an S7-1500 Software Controller means that the interface is assigned to the S7-1500 Software Controller and can only be used by it. In this case, the Software Controller, as IO controller, can operate a PROFINET IO system on this interface.

Only with this type of interface assignment is it possible to configure the IP address and the device name of the PROFINET interface and load it in the PC system.

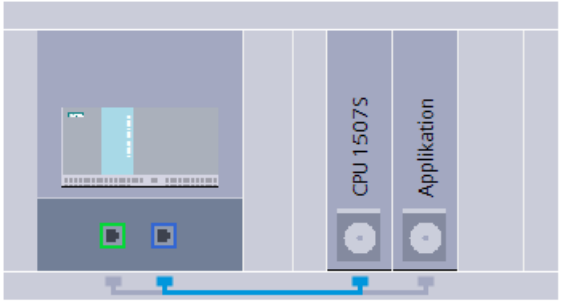

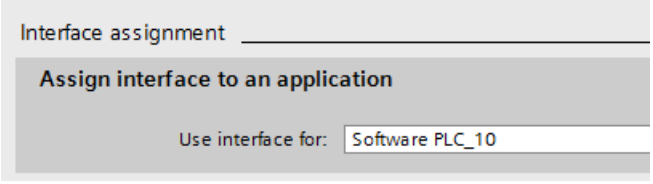

Example 2: Assigning the interface to the SIMATIC PC station

A blue reference line from a selected onboard interface to several software components means that this interface is assigned to the PC station and can therefore be used by all software components. Programming device communication, for example, can be performed through this interface.

Although you can edit the IP address of the interface with this type of interface assignment, this setting is not loaded in the PC system. The IP address must be set in Windows on the PC system. It still makes sense to synchronize IP addresses, for example, to be offered exactly this PC station in the "Extended download" dialog (filter function).

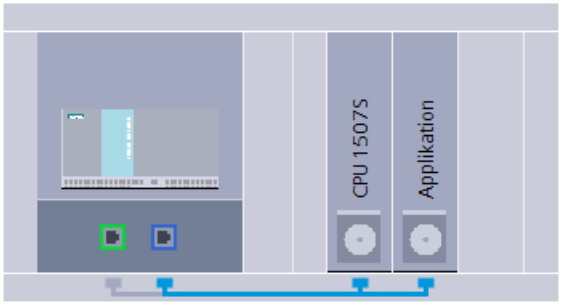

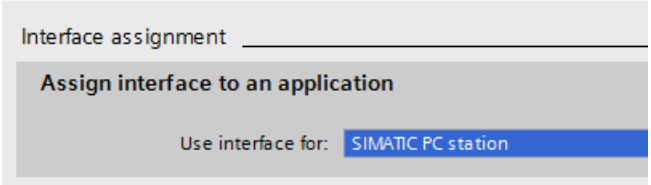

Example 3: None, or a different Windows setting

If you assign the "None, or a different Windows setting" option to an onboard interface, no relation to the inserted software components is possible and therefore also not visible. Only Windows applications can use this interface; however, they are not visible in the configuration. The IP address of the interface is not visible in the configuration as well.

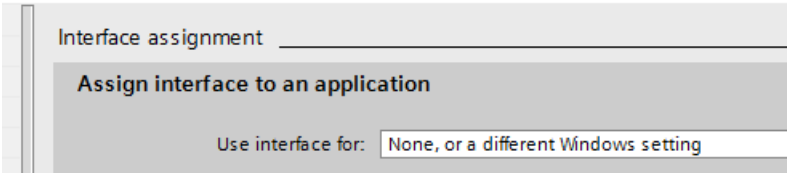

Displaying the position of the interface

The position of the interface is displayed below the "Interface assignment" area in the Inspector window. Depending on the assignment to the PC station or to the Software Controller, the index or the hardware resource/interface type is displayed.

Assigning interfaces for communication of the PC Station V2.x

Interface types

The PC station uses the Ethernet interfaces on the PC for communication. The following communication services are available through these Ethernet interfaces:

- Engineering access to the PC station and the installed components, for example, the S7-1500 Software Controller or WinCC RT Advanced
- HMI connections to the PC station or to the Software Controller
- Data communication (S7 communication of the Software Controller)
- HMI connections, for example, between WinCC RT Advanced and external controllers
- S7 connections routed through the PC

Communication between devices

To enable configured communication between components on the PC and other SIMATIC devices, you need to assign the interface to the PC station. The following Ethernet interfaces can be assigned to PC Station V2.x:

- On-board Ethernet interfaces, for example, X1 and X2, of the SIMATIC IPC (not: on-board CP 1616)
- On-board X2 interface of CPU 1515SP PC

A maximum of 2 on-board interfaces can be assigned.

Engineering functions are possible with all Ethernet interfaces of the PC as long as they are operated under Windows, even if they are not assigned to the PC station.
However, for simplified commissioning, at least one interface must be assigned:

- SIMATIC PC: X1 interface
- CPU 1515SP PC: X2 interface
- No use of external CPs

Note

Interfaces for exclusive use can also be assigned to the S7-1500 Software Controller. These are then not available for the PC station.

Procedure

To assign the interfaces for the communication to the PC station, follow these steps:

1. Select the PC system.
2. Select the desired integrated interface in the device view.
3. In the Inspector window, select the area "General > Interface assignment".
4. Assign the interface to the "SIMATIC PC station".

5. Compile the project with "Edit > Compile".
6. Download the hardware configuration of the PC system to the target device.

Result

After downloading, the interfaces can be used for configured communication services.

Reference

You can find more information on the special aspects of configuring interfaces with the S7-1500 Software Controller in the SIMATIC S7-1500 CPU 150xS operating manual (https://support.industry.siemens.com/cs/de/en/view/109249299).

SIMATIC NET version

Information on the SIMATIC Net version is available at Service&Support under the Entry ID: 63098071 (https://support.industry.siemens.com/cs/document/63098071?lc=en-WW).

SIMATIC NET version for onboard interfaces

The SIMATIC NET version of the onboard interfaces can be found as follows:

1. Select the onboard interface.
2. In the Inspector window, select the area "General > Catalog information".
   - For a **V2.x PC system,** STEP 7 automatically sets the version of the PC station once a SIMATIC S7-1500 Software Controller has been configured. This setting should not be changed. SIMATIC NET cannot be used or configured.
   - For a **V1.0 PC system**, the latest version of the SIMATIC NET CD from the hardware catalog is displayed, for example V8.2, see the description text. STEP 7 does not know the SIMATIC NET version actually used because you are installing and configuring PC System V1.0 with S7-RTM (Station Manager) from the SIMATIC NET CD.

SIMATIC NET version for PC Station

If you select a PC system in the network view and select the "PC Station" area in the Inspector window, the various PC system versions can be selected in the drop-down list.

- Select "PC System V1.0" option: **No** SIMATIC NET version displayed because STEP 7 does not know the SIMATIC NET version actually used, see above.
- Select the "PC-System V2.x" option: SIMATIC NET cannot be used or configured.

1.4.3.5 Special features of the S7 Modular Embedded Controller

Special features of the S7 Modular Embedded Controller

Recommendation

For new projects we recommend switching to the new product generation with S7-1500 Software Controllers.

Reference

You can find additional information on the S7-modular Embedded Controller in the SIMATIC Embedded Automation S7 Modular Embedded Controller (https://support.industry.siemens.com/cs/de/en/view/37971572) operating instructions.

Introduction and basic procedure

Introduction

S7-modular Embedded Controllers (S7-mEC) are PCs in S7-300 design with preinstalled and preconfigured software:

- Windows Embedded Standard operating system
- WinAC RTX (F)
- SIMATIC NET
- WinCC Runtime (only with EC31-HMI/RTX)

Ethernet interfaces of the S7-modular Embedded Controller

An S7-modular Embedded Controller has two integrated Industrial Ethernet interfaces:

- The X1 PN LAN P1 / X1 PN LAN P2 interface can be configured as a PROFINET interface.
- The X2 P1 PN/IE LAN interface is assigned to the PC station in index 3 at IE General and is preconfigured for Industrial Ethernet communication.

Note

If accessing S7-mEC via Remote Desktop Protocol, ensure that S7-mEC and the Engineering PC/programming device are connected to the same Ethernet subnet.

To access S7-mEC via Remote Desktop protocol, you have to use the IE General network connection (usually X2 P1 PN/IE LAN interface).

Basic configuration and programming of the S7-mEC

Since an S7-mEC is a preconfigured system, follow these steps:

1. Create a project.
2. Insert a PC station.
3. Configure the hardware.
4. Create an S7 program.
5. Configure visualization in WinCC for the EC31-HMI/RTX.
6. Download the configuration data to the WinLC RTX EC.

EC31-HMI/RTX: Configuring visualization

For EC31-HMI/RTX only

Create the visualization project for the EC31-HMI/RTX on the engineering PC, using SIMATIC WinCC.

Requirement

- WinCC is installed on the engineering PC.
- A WinCC RT object had been inserted in the PC station.

Procedure

In order to create a visualization project for EC31-HMI/RTX, follow these steps:

1. Open the device view of the PC station of the EC31-HMI/RTX.
2. Select WinCC RT object.
3. Select the "Open object" command.
   WinCC is started.
4. Create and configure the necessary elements: Pictures, alarms, tags, etc., using the WinCC editors.
5. Save and test the visualization.
6. Transfer the project to the EC31-HMI/RTX target system.

Downloading the configuration data to the target system

Download options

You can download the following configuration data from STEP 7 to the embedded controller:

- The entire PC station (including, for example, IE General)
- Only the configuration data of the WinLC RTX / WinLC RTX F
- Only the S7 program of the WinLC RTX / WinLC RTX F

For an EC31-HMI/RTX, you will need to transfer the configuration data from WinCC.

Requirement

- The enhanced write filter (EWF) must be deactivated on the Embedded Controller if the entire PC station is to be downloaded.
- The embedded controller is connected to the programming device / PC via Industrial Ethernet. The PG/PC interface must be set to TCP/IP.
- **EC31-HMI/RTX only:** The visualization project has been created in WinCC.
- Before the following functions become available, the embedded controller must be connected to the engineering system via a port of the X1 PN LAN interface:
  - Assign PROFINET IO device names
  - PROFINET CBA: Interconnection download and online functions

Procedure

1. Deactivate EWF on the EC31-RTX / EC31-RTX F / EC31-HMI/RTX (if not already deactivated). After EWF has been deactivated, turn off and restart the embedded controller.
2. Start WinLC.
3. Activate the PC station in the Station Configuration Editor ("Enable station" button).
4. Change the embedded controller to STOP mode.
5. Download the hardware configuration, the S7 program and the configuration data to the target system.

Making ready for operation

To make the EC31-RTX / EC31-HMI/RTX ready for operation, follow these steps:

1. Set the enhanced write filter for drive C:\ to the "activated" state.

NOTICE
Flash memory failure If you leave the enhanced write filter in "deactivated" state, you risk an early failure of your flash memory due to the persistent write access of the operating system. When the enhanced write filter is activated, the number of write accesses to the operating system partition is minimized, thereby increasing the life of the flash memory.

2. Record all special settings (such as changes to the IP addresses) made in the course of application development. Store them along with your project.
3. Create a backup copy of your settings.

1.4.4 Configuring CPs and applications for PC stations

1.4.4.1 CPs for PC station 1.0

PC station

SIMATIC NET PC software for PC communications processors

SIMATIC NET PC software for PC communications processors

The use of the SIMATIC NET PC software versions listed below is recommended for the following communications processors depending on the operating system:

- CP 1613, CP 1613 A2, CP 1623, CP 1628
- CP 5613, CP 5613 A2, CP 5614, CP 5614 A2, CP 5623, CP 5624

You should use the following versions of the SIMATIC NET PC depending on the operating systems:

- Windows 8 (Professional, Enterprise), 32- and 64-bit and Windows Server 2012
  DVD "SIMATIC NET PC Software" V12
- Windows 7 (Professional, Enterprise, Ultimate), 32- and 64-bit and Windows Server 2008 R2
  DVD "SIMATIC NET PC Software" V12
- Windows XP, Windows Server 2003 and Windows Server 2008, 32-bit
  CD "SIMATIC NET, PC Software", Edition 2008 + SP6, version number V7.1.6

Slot rules for a PC station

Introduction

Components such as CPs, PC-based user programs for connection configuration and OPC servers can be plugged into slots (index) 1 to 32. Certain slot rules apply for PC stations.

Inserting an OPC server

In PC stations, OPC servers can be inserted for OPC clients that access remote automation systems via configured connections:

- A maximum of one OPC server can be inserted in a PC station for OPC clients.
- If an OPC server is plugged into a PC station, a CP can be assigned to it.

Other standard applications can be used instead of the OPC server or in addition to an OPC server.

Plugging components

The following rules apply to index numbering and plugging components into the PC station:

- The index numbers 1 to 32 can be assigned.
- The first plugged component always gets index 1.
- Additional plugged components get the next free index number.
- If a component is deleted, the index numbers of the other modules remain unchanged. There may be gaps in index numbering for this reason.
- Inserting components between existing components is permissible.
- The following applies for inserting components when components are already present:
  - If free index numbers are available between two components, the first free index will be assigned.
  - If no free index numbers are available between two components, the index of all subsequent components will be incremented by one number until the first free index. A component with index 32 can, in this way, be shifted to the area of unplugged modules. You can therefore expand the PC station without losing components with important settings in the event of over-allocation.

Inserting submodules

WinLC RTX supports as a sub-module

- a maximum of one CP 5611/21 card or an integrated CP 5611/21 interface
- a maximum of one PROFINET interface (CP 1616, CP 1604, among others)
- a maximum of four sub-modules altogether. From the four submodules, there may be any number of CP 5613 communication interfaces.

The following rules apply when a component has submodules:

- A controller contains a fixed number of free slots.
- Free slots are shown in the graphic and tabular device view.
- Submodules can only be inserted on free sub slots.
- A subslot is freed when a submodule is deleted.
- Subslots do not reserve an index number.

Ethernet PC CPs

Subnet addressing with CP 1613, CP 1623 and CP 1628

CP 1613, CP 1623 and CP 1628 are communications modules with a microprocessor. For the secure handling of communications connections these are processed on the module. For diagnostic purposes (SNMP, DCP) the protocol stack in your PC is used. To allow both protocol stacks to communicate with the same partners (IE and NDIS access), we recommend that you place both protocol stacks of a module in the same subnet.

Pulling and plugging Ethernet PC CPs

Pulling and plugging Ethernet PC CPs

Ethernet modules, for example USB network adapters, must already be plugged and activated when the system starts. If Ethernet modules are deactivated or do not exist when the system starts or they are pulled and plugged again during operation, the "Available devices" functionality does not display all the nodes in STEP 7. This also applies to the change of docking stations when the Ethernet module is not plugged into the notebook but rather into the docking stations.

Activate the deactivated Ethernet modules by using "Control Panel > Network Connection" and reboot the system. Also reboot the system if Ethernet modules were plugged during operation or if the docking station with the Ethernet module was changed during operation.

OPC configuration (Ethernet CP)

Reference

Setting the OPC configuration for the PC Ethernet CP in the "Properties > General > OPC configuration" parameter group.

The parameter group exists if an OPC server is configured in the PC station.

"Access via the OPC server" option

Enable the module for operation via an OPC server configured on the device.

"Automatic mode of the PROFINET IO controller" option

If you select this option, the operating mode of the PROFINET IO controller is set automatically to OPERATE.

In the OPERATE mode, the PROFINET IO controller has access to the connected PN IO system.

The option can be selected if the option "Access via the OPC server" was selected.

- Option enabled (default setting)
  If you select this option, the PROFINET IO controller is changed to the OPERATE mode as soon as there is access to an IO device. This access can either be over the OPC Item interface or the IO Base user programming interface.
  No prior calls are required to initialize the operating state.
- Option disabled
  Prior to data access, the PROFINET IO controller needs to be set to the OPERATE mode. To do this, use suitable initialization calls for an interface in the user program.
  The interface in the user program can be an OPC item interface or an IO-Base user programming interface.

Ethernet port

Time synchronization

Time synchronization

Here, you decide whether or not the CP processes time frames.

The following two options are available depending on where the time master is located

- Master (time transmitter)
  Time messages can come from the PC station and be forwarded to the subnet (LAN). A PC application that performs the time master function transfers the time information to the CP in a job.
- Slave (time recipient)
  Time messages can come from the subnet (LAN) and be forwarded to the station. A PC application can read the transferred time using a suitable job.

If you are running several CPs in your station, you should consider the flow of time frames in relation to the time-of-day master since you can pass time frames from one network to another network using the function described here.

If there are several CPs that are connected to the same network in a station, only **one** of these CPs may forward the time synchronization messages.

Advanced options

Keep-Alive connection monitoring

Reference

Making special settings for the Ethernet interface in the "PROFINET/Ethernet interface > Advanced options > Interface options" parameter group

Keep-Alive connection monitoring

If the keepalive mechanism is activated, keepalives are sent for connections via TCP/IP. This ensures that connections are terminated if one of the communication partners fails and that connection resources are released again.

Depending on the module type, there are two options for the setting:

- Enable/disable connection monitoring with a fixed interval time
  You can enable and disable the mechanism for connection monitoring. An interval time of 30 seconds is specified for the module; this is the interval at which the keepalives will be sent to the partner of a communication connection.
  - Keepalive disabled: 0 seconds
  - Keepalive enabled: Interval time permanently set to 30 seconds
- Configurable interval time for monitoring frames
  Here, you can set the interval time in seconds with which keepalives are sent to the communication partner:
  - Range of values: 0 .. 999999
  - Function switched off: 0
  - Default: 30

Notes / recommendations:

Note that the keepalive mechanism can cause subordinate connections (for example, an ISDN telephone connection) to be maintained even though user data is not actually being transferred. If this is not desired, follow these steps:

- Disable the monitoring mechanism (interval time = 0)
  or
- Set the interval so high that the underlying connection is terminated before a keepalive frame is sent when there is no data traffic.

Note

Modules of other manufacturers

When using modules of other manufacturers, refer to their information regarding keepalive settings.

Notes on connection release monitoring

Relationship to connection configuration

The possible time monitoring setting in this case relates to the time monitoring that can be set in the connection configuration as follows: The lower value is valid.

Time monitoring can be configured in the connection configuration in "Properties > Details on connection setup".

Web

Web server on the PC CP

Reference

Configuration of access to the Web server on the PC CP in the "Web server" parameter group

"Enable Web server on this module" option

The CP provides you with the functionality of a Web server for access via port 80 using a Web browser. Certain HTML pages with CP information and diagnostic functions are stored in a memory area of the CP for this.

Enable this option in order to be granted access to these HTML pages.

- Option enabled
  Enable this check box so that the Web server of the CP is started after downloading the configuration data. You can then read out all the information available on the CP Web pages using a Web browser.
- Option disabled
  If the check box is disabled, access to information provided by the CP Web pages is restricted. Disabling may be suitable in certain operating situations to keep additional resources free on the CP.
  The following pages are not displayed:
  - Diagnostics buffer
  - Module information
  - Topology

Automatic update

If the check box is selected, the Web pages are updated automatically at the interval set in "Update interval".

The Identification Web page is excluded from the automatic update.

Selecting languages

Select the languages for language-dependent texts to be downloaded to the CP. The number of selectable languages is CP dependent. The texts in the diagnostics buffer, for example, are language dependent.

You should also remember that the languages you select here must be installed in the project. You will find the selection of project languages in the project tree in "Languages & Resources"

If the languages you select were not previously installed, the Web server can only display texts in your default language.

Options for setting protection

Protection level

Below, you will see how to use the individual access levels of the CP.

CPs such as the CP 1626 provide various access levels to restrict access to certain functions.

The parameter assignment for the access levels is made in the table. The green check mark in the columns on the right of the particular access level indicate the maximum possible operations without knowing the password for this access level. If you want to use the functions in the check boxes without a check mark, you will need to enter a password.

NOTICE
Configuring an access level does not replace the know-how protection Setting the parameters for access levels prevents unauthorized changes to the CP by restricting the download rights. Blocks on the memory card are, however, neither write nor read protected. To protect the code of blocks on the memory card, use the know-how protection.

Default reaction

The default access level is "Full access (no protection)". Every user can read and modify the hardware configuration and the blocks. No password has been set and no password is required for online access.

The individual access levels

With a CP, you can set the following access levels:

- Full access (no protection): The hardware configuration and the blocks can be read and modified by anybody.
- Read access: With this access level, without entering the password, only read access to the hardware configuration and the blocks is possible; in other words, you cannot download the blocks or hardware configuration to the CP without entering the password. Without the password, writing test functions and firmware updates are also not possible.
- No access (complete protection): If the CP is completely protected, neither read nor write access to the hardware configuration and blocks is possible.

With the legitimization provided by using the password, you once again have full access to the CP.

Reaction of a password-protected CP during operation

Protection of the CP is effective after the settings have been loaded on the CP.

Before an online function is executed, a check is made to establish whether or not it is permitted. If there is password protection, you will be prompted to enter the password.

Example: The module parameters were set for read access and you want to use the "Modify tags" function. Since this is write access, the set password must be entered before the function can be executed.

The functions protected by the password can only be executed by one PG/PC at any one time. Another PG/PC cannot log on.

The access rights to the protected data apply for the duration of the online connection or until the access rights are canceled again with "Online > Delete access rights".

Each access level allows unrestricted access to certain functions even without entering a password, for example identification using the "Accessible devices" function.

Setting parameters for access protection

Below you will see how to set an access level for a CP and enter passwords.

You can enter several passwords for a CP setting up different access rights for different user groups.

The passwords are entered in the table so that precisely one access level is assigned to each password.

How the password takes effect is shown in the "Access level" column.

Example:

You select the access level "No access (complete protection)" and enter your own password for each of the access levels higher up the table.

For users that do not know any of the passwords, the CP is completely protected.

For users who know one of the set passwords the effect depends on the table row in which the password is located:

- The effect of the password in row 1 (full access (no protection)) is as if the CP was unprotected. Users that know this password have unrestricted access to the CP.
- The effect of the password in row 2 (read access) is as if the CP was write-protected. Despite knowing the password, users that know this password only have read access to the CP.
- The effect of the password in row 3 (complete protection) is as if the CP was write and read protected. Users that know this password only have read access to the CP.

Procedure

Follow the steps below to set the parameters for the access levels of a CP:

1. Open the properties of the CP in the Inspector window.
2. Open the "Protection" entry in the navigation panel.
   A table with the possible access levels is displayed in the Inspector window.
3. Select the required access level in the first column of the table. The green check mark in the columns on the right of the particular access level indicate which operations are still possible without entering the password.
4. If you have selected an access level other than "full access":
   - Assign a password for full access in the "Password" column in the first row (full access).
   - Repeat the selected password in the "Confirm password" column to protect against incorrect entries.
   - Make sure that the password is adequately secure; in other words, that it does not include a pattern that can be machine read!
   - The entry of the password in the first row "Full access (no protection)" is obligatory and allows a user who knows the password unrestricted access to the CP regardless of the selected access level.
5. As necessary, assign other passwords to the required access levels if the selected access level permits this.
6. Download the hardware configuration so that the access level takes effect.

Result

The hardware configuration and the blocks are protected from unauthorized online access according to the set access level. If an operation cannot be executed without a password due to the set access level, a dialog appears prompting entry of a password.

Customer-specific adaptations for CP 1626 as of V1.0

General

With the "Customer-specific adaptations" a device manufacturer can replace the SIEMENS-specific information with any self-selected information. This information is also exported to the GSDML file.

PROFINET vendor ID and PROFINET device ID

These parameters are applied for from the PI (PROFIBUS International) for the specific manufacturer and can then be entered here.

Article number

In this input box, you can enter any article number.

Product family, vendor name, product name and software version

These parameters are also used in WEB user interfaces and in the SNMP protocol and can be entered here.

GSD bitmap file

In this input box, you can specify the name of the GSD bitmap export file that you create with the input. The bitmap format is preset as default and does not need to be entered.

Assigning RT or IRT functionality

As of version V1.1 you can select between the two options "RT" and "IRT". The selected option is exported with the GSDML file regardless of the setting in "Parameter assignment of PN interface by higher-level IO controller" in the dialog "Operating mode".

Customer-specific adaptations for CP 1604/CP 1616 as of V2.7

General

With the "Customer-specific adaptations" a device manufacturer can replace the SIEMENS-specific information with any self-selected information. This information is also exported to the GSDML file.

PROFINET vendor ID and PROFINET device ID

These parameters are applied for from the PI (PROFIBUS International) for the specific manufacturer and can then be entered here.

Article number

In this input box, you can enter any article number.

Product family, vendor name, product name and software version

In this input box, you can enter any names.

GSD bitmap file

In this input box, you can specify the name of the GSD bitmap export file that you create with the input. The bitmap format is preset as default and does not need to be entered.

Assigning RT or IRT functionality

As of version V2.8 you can select between the two options "RT" and "IRT". The selected option is exported with the GSDML file regardless of the setting in "Parameter assignment of PN interface by higher-level IO controller" in the dialog "Operating mode".

PROFIBUS PC CPs

Operating PROFIBUS CPs in computers with STEP 7 V5.5

Operating PROFIBUS CPs in computers with STEP 7 V5.5

If both STEP 7 V5.5 and STEP 7 Professional are installed on a computer and STEP 7 V5.5 is removed, the setup repair function of STEP 7 Professional must be run before the following communications processors can be used as network adapters for downloading firmware:

- CP 561x, CP 5611 A2, CP 5612, CP 5613 A2, CP 5614 A2
- CP 5621, CP 5622, CP 5623, CP 5624
- CP 5711

Firmware download for S7-300/400 PROFIBUS CPs

Firmware download for S7-300/400 PROFIBUS CPs via CP 5612 or CP 5622

If the required interface parameter assignment "FWL" or "FWL_FAST_LOAD" is not available for the firmware download, follow these steps:

1. Open the "Set PG/PC interface" program. You can find the program in the Control Panel of your Windows operating system.
2. Select the interface parameter assignment depending on the operating system and then copy this:
   - For 32-bit operating systems:
     "CP5612(PROFIBUS)" / "CP5622(PROFIBUS)"
   - For 64-bit operating systems:
     "CP5612.PROFIBUS.1" / "CP5622.PROFIBUS.1"
3. Depending on your requirements, give the copy one of the following names:
   - For 32-bit operating systems:
     CP5612(FWL) / CP5622(FWL)
     CP5612(FWL_FAST_LOAD) / CP5622(FWL_FAST_LOAD)
   - For 64-bit operating systems:
     "CP5612.FWL.1" / "CP5622.FWL.1"
     "CP5612.FWL_FAST_LOAD.1" / "CP5622.FWL_FAST_LOAD.1"
4. Open the properties of the entry and make the following settings in the "PROFIBUS" tab:
   - Select the check box for "PG/PC is the only master on the bus".
   - Assign the value "0" to the address.
   - Set the value 187.5 Kbps (or the value 1.5 Mbps for FWL_FAST_LOAD) as the transmission speed.
   - Change the "Highest device address" to the value 126.
   - Change the profile to "Universal (DP/FMS)".
5. Exit the "Set PG/PC interface" program.

Result:

The configured interface parameter assignment is created and is then available in the firmware loader.

LSAP configuration

Reference

Setting the LSAP configuration for the PC PROFIBUS CP in the "Properties > General > LSAP configuration" parameter group.

Reserving LSAPs

Reserve LSAPs (Local Service Access Point) for other protocols used on the CP.

Example FDL connections: The FDL protocol uses LSAPs that must not be used by other protocols at the same time.

To reserve an LSAP, select the number belonging to the LSAP in the displayed list.

FDL connections

LSAPs that have already been used in FDL connections are shown as being disabled.

Display the local LSAP and the partner LSAP in the connection table to be able to check which connection uses which LSAP.

As default, these values are not displayed. To show them, open the shortcut menu in the header bar of the connection table. Click the Show/hide” menu item and select in the list “Local LSAP” and “Partner LSAP”.

The LSAPs are also shown in the connection properties under “Address details”. Here, you can, when necessary, also change the LSAP used.

OPC configuration (PROFIBUS CP)

Reference

Setting the OPC configuration for the PC PROFIBUS CP in the "Properties > General > OPC configuration" parameter group.

The parameter group exists if an OPC server is configured in the PC station.

"Access via the OPC server" option

Enable the module for operation via an OPC server configured on the device.

"Set DP master to OPERATE automatically" option

If you select this option, DP mode starts automatically when the OPC server starts up.

Activity monitoring of the CP enabled - "DP watchdog" option

If the OPC server no longer accesses the DP interface (due to an error), the module reacts as follows if the watchdog is activated: Data with the value 0 (substitute values) is sent to all DP slaves assigned to the OPC server.

Selectable parameter:

- Timeout of the DP watchdog
  If the watchdog is activated, the substitute values are transferred if there is no further access to the DP interface within the time set here.
  Default: 6000 ms

"Diagnostics monitoring" option enabled

If diagnostics monitoring is enabled, the OPC server itself monitors the diagnostics flag when inputs and outputs are accessed and evaluates the diagnostics data of DP standard slaves. If errors or problems are detected in the diagnostics data, access to the inputs and outputs of the DP slaves is prevented for the lock time set here; read and write jobs are therefore acknowledged with an error.

Selectable parameter:

- Lock time for Read / Write
  Default: 500 ms

Error wait time for DP C1 jobs

Here, you specify an error wait time for the DPC1 services (acyclic transfer as DP master class 1).

Default: 15000 ms

OPC server functions for DP master class 2

Here, you can enable the OPC server functions for DP master class 2. The settings here relate to the data record functions according to DP master class 2.

A 2-level option is available:

- Enable data record functions
  With the data record functions, device-specific information can be read out or parameters can be written. With this option, you enable the OPC server that supports this function.
- Operate in PROFIdrive mode
  If you select this extra option, the PROFIdrive bus server is enabled. As a result the data record functions can be used (only) in the special PROFIdrive mode.

Selectable parameter:

- Connection establishment timeout
  If the request to establish a connection is not responded to by the communications partner within the time set here, OPC jobs are aborted with an error message.
  Since due to the structure of the network some time may be necessary before a connection can be established successfully, the connection establishment timeout should be longer than the job timeout (refer to the error wait time for asynchronous jobs).
  Default: 15000 ms
- Error wait time for asynchronous jobs
  If a read or write job sent to a partner device is not responded to within the time set here, the pending OPC jobs are aborted with an error message.
  Default: 15000 ms

Configuring the LDB file

Reference

Setting the LDB configuration for the PC PROFIBUS CP in the "Properties > General > LDB configuration" parameter group.

Note

Version of the SIMATIC NET PC software

The LDB file can only be used for CPs with the SIMATIC NET PC software up to the version May 2000.

Generating LDB file (applies only to CP is with firmware < V6.0 SP5 or up to SIMATIC NET PC software May 2000)

The configuration of the DP master system must be exported to an LDB file (loadable database) when saving and compiling. This file contains the DP project information. It is required by the DP system attachments for PG/PC such as SOFTNET or CP 5614 in the PC stations.

Note

LDB file and identifying the input/output addresses of the DP modules

The addresses of the input/output areas of the DP slaves are acquired automatically in ascending order based on the PROFIBUS address of the DP slaves and on the basis of the inserted DP modules and entered in the LDB file.

Address information or address changes to the input/output addresses of the DP slaves in the DP master system configured here therefore have no effect on the address information stored in the LDB file.

Networked PC configurations

Configuration dialog box

Reference

You select the " Online > Configure PC station online" menu command for a selected PC station.

You transfer the selected configuration for a PC station or display its current configuration with the highlighted dialog.

Requirements

The "SIMATIC NET PC Software" DVD must be installed on your target computer and the configuration computer.

If more than one communication module is present on the configuring computer or the target computer, you must first specify the particular module for the configuration operation. The SIMATIC Shell program is available for this purpose.

If you have specified the module for the configuring computer or if only one communication module is present, the local network connection is already displayed and cannot be changed.

Provided you have selected the target computer in your STEP 7 project and there is an explicit assignment to the accessible computers, the "Accessible target computer" and "Target computer" fields are also preassigned accordingly and do not have to be changed.

Note

SIMATIC Shell

The "SIMATIC Shell" program is installed with the SIMATIC NET PC software products. Note the documentation on the SIMATIC NET PC software.

Follow these steps:

1. From the drop down list box, select the network connection to be used to address the target computer.
2. Click the "Update" button to determine the accessible computers again.
3. In the "Accessible computers" list, select the desired target computer or enter the name of the target computer directly in the "Target computer" field. The designation of the target computer is the same as the computer name in Windows.
4. Now select one of the following procedures:
   - Display the current configuration of the selected target computer: Click the "Display..." button for this purpose.
   - Download the configured configuration to the target computer: Click the "Configure..." button for this purpose.
5. Note any messages that appear following configuration.

Assigning the network connection and target computer

If you do not want to use any preassignments or there is no preassignment, start by following these steps:

1. If necessary, disable the "Use configured target computer" option.
2. From the drop down list box, select the network connection to be used to address the target computer.

3. Click the "Update" button to determine the accessible computers again.
4. In the "Reachable computers" list, select the required target computer. As an alternative, enter the name of the target computer directly in the "Target computer" box. The designation of the target computer is the same as the computer name in Windows.

Configuring the target computer

Continue as follows:

1. Choose between the following procedures:
   – Display the current configuration of the selected target computer: Click the "Display..." button for this purpose.
   – Download the configured configuration to the target computer: Click the "Configure..." button for this purpose.
2. Note any messages that appear following configuration.

Tip:

You can also access the functions described for the buttons by right-clicking.

General notes

Note that after the start of the configuration dialog, changes will not be detected on the local computer. The settings or configuration data detected during the startup are valid. This affects the following actions:

- Enabling or disabling the network adapters.
- Changes in the STEP 7 configuration of the PC station.

"Target computer configuration" dialog box

This dialog box shows the current configuration including modules and applications.

Here, you can compare a current configuration with a configured one.

Station

The station name is shown here.

The station name can be the same as the computer name of the target computer if the target station has not yet been configured with a new name after installation of the SIMATIC NET PC software. Upon booting, the computer name is initially transferred as the station name.

See also

Status symbols during configuring (Page 1497)

Configuration dialog box (Page 1493)

"Configure: selected target computer" dialog box

This dialog box shows the configuration, as currently configured, for the selected target computer including modules and applications.

You can initiate the transfer for the initial configuration or configuration change from here.

When the dialog box is opened, a check is made to determine whether the configuration, as configured, is valid for transfer to the target computer and, thus, can be transferred.

Station

The station name is shown here.

The station name can be the same as the computer name of the target computer if the target station has not yet been configured with a new name after booting. Upon booting, the computer name is initially transferred as the station name.

Checking and adopting the configuration, as configured

Here, you can check the configuration, as configured, before carrying out the final data transfer.

The information bar below the configuration list indicates whether or not the configuration can be performed. For this purpose, the existing physical configuration must match the configuration, as configured.

Note the symbols that indicate special circumstances.

After you have checked the displays and initiated the transfer (OK button), you will be prompted again to confirm the transfer.

When the configuration is rejected

The configuration cannot be transferred in the following cases:

- Configured modules are not present in the PC station.
- Configured modules cannot be explicitly assigned (see assignment rules below).
- The configuration, as configured, exceeds the permissible quantity framework for the PC station.
- A connection to the PC station cannot be established.
  This can be due to the fact that a firewall configured in the PC station blocks access to Port 102, which is used by STEP 7. Enable Port 102 in the PC station for TCP.

Review the notes or messages in the "Error" and/or "Status" fields! For example, you receive warnings if the assignment would cause a change in the previous configuration (example: an already configured module is moved to a different index).

If you double-click the message text, a window with further information regarding the message is displayed.

Assignment rules

During the import, a check is made to determine whether the modules can be assigned and, thus, transferred.

- The specified modules must be present.
- If multiple modules (e.g., PROFIBUS CPs) are found, an attempt is made to make an assignment:
  - Based on the parameter assignment string assigned to the module (assigned in STEP 7)
  - Based on the address (PB address, MAC/IP address)
  - Based on compatibility (range of functions)

Note

Softnet-CPs

Note that only 1 Softnet-CP (PROFIBUS or Industrial Ethernet) at a time may be present in the configuration!

See also

Status symbols during configuring (Page 1497)

Configuration dialog box (Page 1493)

Status symbols during configuring

"Status" column

Note the additional information in the "Causes" column.

Error: Component cannot be assigned/transferred or is not installed

Warning: Assignment was adapted or is not unique.

Information bar below the configuration list

A group message containing information about the import operation is output in the information bar:

Configuration cannot be performed

Configuration can be performed, but with warnings: Assignment of individual components was adapted or is not unique.

Configuration cannot be performed

See also

"Configure: selected target computer" dialog box (Page 1496)

PC CP settings that do not depend on the protocol or the network

XDB configuration

Reference

Setting the XDB configuration for the a PC station in the "Properties > General > XDB configuration" parameter group.

XDB configuration

- Compatibility - "S7RTM is installed" option
  When you configure a new PC station, the check box is enabled. In this case, a runtime station manager (S7RTM) is required on the target PC station. The target PC station must be configured with the Station Configuration Editor (SIMATIC NET PC software as of approximately 7/2001).
  If the check box is disabled, configuration files are generated that can be interpreted by "older" PC stations.
- Storage location of the configuration file
  The configuration with the connection descriptions for the SIMATIC PC station is stored in a configuration file (*.XDB) when you save and compile. You then install this file on the relevant PC station.
  With the "Browse" button, you can change the default path for the configuration file.
  The relative path information (.\...) relates to the folder in which the current project is stored (<Install dir>\S7proj\<project>\XDB). If you enter an absolute path (for example C:\...), remember that the access only works as long as the project remains in the current environment.

Setting the interface parameter assignment

Checking the interface parameter assignment

When you take the PC communications card from the hardware catalog and place it in a rack, it is assigned a suitable interface parameter assignment.

The valid parameter assignment can be recognized based on the information on the module type in the module catalog (name of the module and number of configurable modules of the same type) and the existing already configured modules (assignment of the board number).

If several CPs of the same type are configured in the station, the module names have an additional consecutive number added to them – the board number. From the available drop-down list, you can see the permitted entries and, if necessary, select them.

In special cases, for example when you change the module arrangement of identical modules in the PC station, it may be necessary to adapt the interface assignment here; changing the arrangement means the actual hardware and not the "virtual" slot in HW Config.

Note

Interface parameter assignment on the target system (PC system)

Make sure that the same interface parameter assignment is used in the target system (PC station) as you specify here. Otherwise, you will have an interface conflict in the station after loading the configuration file.

Note

Checking on the target system

Use the Setting the PG/PC Interface function to check the settings on the target system (PC station). You will find this in the Windows Control Panel. In the "Access path" tab, you will see which interfaces are configured and whether conflicts exist.

1.4.4.2 Applications for PC station 1.0

OPC Server user application

OPC server

General protocol-specific settings

General

Cycle time

Effects of the selectable cycle time

The cycle time influences how often the OPC server updates the values of the OPC items in its cache using a new communications job and how often it checks for changes. The cycle time therefore influences the performance, the communication load, and the processor load.

If the value for the cycle time is set too low, this increases processor load unnecessarily!

Relationship between the cycle time and update interval (UpdateRate)

The update interval (UpdateRate) parameter that you can set in the user program specifies the shortest interval for checking the values of the OPC items of an active OPC group. The server checks whether or not the values of the OPC items have changed at the earliest when this interval has elapsed. This limits the communication load.

The update intervals used by the OPC server for SIMATIC NET are multiples of the cycle time specified here during configuration. The shortest possible update interval is the cycle time.

Relationship between the cycle times set for specific protocols

Since the SIMATIC NET OPC Server can use variables of different protocols at the same time,

the minimum update interval of the OPC Server is the highest value set for the cycle time for the active protocols (protocols for which connections are configured).

Example:

A cycle time of 100 ms is configured for S7. No other protocol is being used. 100 ms is used by the OPC server as the minimum update time. The DP protocol is now added and the cycle time configured for this protocol is 50 ms. Result: The OPC server now uses the value 50 ms as the minimum update time for all protocols.

Note: The response described results from the requirements of the OPC standard. The OPC standard specifies that data changes must not be signaled more quickly than specified by the minimum update rate.

Actual update interval

The actual update interval (UpdateRate) that can be achieved during operation depends on various factors including:

- The underlying communication protocol
- The communication partner
- Number and type of items

As a result, the actual update interval that can be achieved maybe longer than the minimum value configured here.

If the process value changes several times within the actual update interval, some value changes will not be forwarded to the application.

Note:

The data throughput with asynchronous and synchronous jobs does not depend on the update interval parameter (UpdateRate) and the cycle time set in the configuration.

Access protection

Enabling access protection

Reference

Settings for the OPC server / "Properties > General" parameter group; a protocol-specific node is selected.

Meaning of access permissions for OPC variables

With the OPC server, access permissions to individual variables or groups of variables can be specified for a specific protocol. By specifying access permissions, you can, for example, prevent an internal controller calculation variable from being overwritten.

- Standard rights
  For OPC items that are not specifically named, you can specify a default value here:
  - Read-write: Read and write
  - Read: Read
  - Write: Write
  - None: No access
- OPC item-specific access rights
  If you select this option, you can assign access permissions explicitly for certain OPC items in the input box.

See also

Syntax - OPC item-specific access rights (Page 1502)

Syntax - OPC item-specific access rights

You can enter access rights for one or more OPC items.

The default rights defined for the other OPC items are not affected.

The syntax for OPC item-specific access rights is defined as follows:

<OPCItem>=<rights>

where:

- <OPCItem>
  You specify one or more OPC Items according to the syntax defined in the OPC documentation. It is also possible to use aliases.
  The following placeholders can be used:
  * Any number of irrelevant characters
  ? exactly one character is irrelevant
- <rights>

Read-write	Read and write
Read	Read
Write	Write
None	No access

Rules for evaluation

- The specific rights assigned here have priority over the default rights you assigned in the input area Access protection.
- The specific rights assigned here are evaluated in the order shown here. If there is a multiple assignment for an OPC item, the last assignment is always valid.
  Example:
  DP:[CP 5611]Slave040_QB*=RW
  DP:[CP 5611]Slave040_QB1=R
  DP:[CP 5611]Slave040_QB2=W
  DP:[CP 5611]Slave040_QB1=W
  DP:[CP 5611]Slave040_QB1*=R
  Due to this entry, the following access rights are effective:
  DP:[CP 5611]Slave040_QB2=W
  DP:[CP 5611]Slave040_QB1=R
  Tip: Please read the detailed description of the syntax in the OPC documentation listed below. There, you will find a series of examples - also for applications such as PROFINET IO.
- Leading zeros in items and in the definitions made here for the access rights are always ignored.
  The following example illustrates the advantage of this:
  You configure the following access right: Slave004_QB0=R
  You have therefore implicitly assigned the access right to the following items:
  - Item 1: Slave00004_QB000
  - Item 2: Slave4_QB00,1 (note: the syntax ",1" is demanded by the OPC Scout)

  When using the wildcards "*" and "?", this means the following:
  Do not use these wildcards at positions at which a leading zero can precede an item!
  Example:
  The definition Slave0?9M06_AB1=R must be replaced by the two following definitions:
  - Slave?9M06_AB1=R
  - Slave9M06_AB1=R

Further information

Read the detailed description of the syntax in the OPC documentation. There, you will find a series of examples - also for applications such as PROFINET IO.

Leading zeros in items - example with wildcards:

You are using the following item:

- Slave00004_QB000

An access right definition that would fail in the comparison might be:

- Slave0?4_QB0=R

Reason:

In the comparison, a leading "0" is found in the item, the item is therefore reduced to: Slave4_QB

Due to the wildcard "?", the leading "0" in the definition is no longer detected as a leading "0". The definition for the access right is therefore not reduced and the comparison fails.

Remedy:

To achieve the required rights for this example, the following two definitions would be necessary:

- Slave?4_AB0=R
- Slave4_AB0=R

S7-specific settings

OPC symbols

Using symbols

Reference

Settings for OPC server / "Properties > S7 > OPC symbols" parameter group.

Using symbols (as of OPC server V6.1)

Here, you can specify whether symbols from the symbol tables configured for a CPU will be included in the system data for the OPC server. This is necessary before user applications (OPC client) can access variables symbolically via the OPC server.

The symbol tables of the CPUs to which S7 connections are configured for the OPC server are used. Only symbols that relate to data blocks (DBs) are taken into account.

The available alternatives have the following meaning:

- None
  No symbols from STEP 7 are included in the system data for the OPC Server. The access rights configured for the OPC items under Access protection apply.
- All
  All relevant symbols from STEP 7 are included in the system data for the OPC server. Making this selection sets the access rights for all symbolic access to the assigned variables to Read/Write (RW). These rights have priority over the access rights for the OPC items set under Access rights.

- Configuring
  All selected and specifically configured symbols from STEP 7 are adopted in the system data for the OPC server. With the "Configure" button, you open the "Symbol configuration" dialog in which you can select and configure the symbols.
- Array elements visible at runtime (selectable as of OPC server V8.0)
  Here, you can decide whether you want the elements of arrays to be visible on the OPC server during runtime. This relates to the arrays configured as symbols.
  As default, the option is disabled because making array elements visible causes a high load on the OPC server. With older OPC server versions (< V8.0), array elements are always visible and this setting cannot be disabled.

Note

Consistency

If you select the "All" option, the current CPU symbol tables are taken into account when you save and compile.

If you use the "Configured" option, remember that subsequent changes in the symbol tables have the following effects in the system data for the OPCserver:

- Newly added symbols are ignored
- Selected symbols that were removed later in the symbol table cannot be saved.

Note

Bit arrays

Symbols that include data of the "bit array" type are not included in the system data of the OPC server!

Note

Note on the procedure

While the symbols are being adopted as described here, no FB in which symbols are used may be open in LAD / STL / FBD. Otherwise some symbols will not be read in.

Symbol configuration

Reference

Settings for OPC server / "Properties > S7 > OPC symbols" parameter group.

Calling the symbol configuration with the "Configure symbols" button.

Overview

In this dialog, you can select and configure the symbols to be adopted in the system data of the OPC server.

The CPUs relevant for the selection of symbols are those to which S7 connections are configured for the OPC server.

In the navigation area on the left, you will see the relevant stations and CPUs along with any structure elements. In the content area on the right, you will see the symbols defined for the

selected CPU or the selected structure element. The assigned data type and the currently configured properties are shown for each symbol. These properties can be configured in the lower part of the dialog.

Note

Note that the display in the "Access" column can also contain the value "mixed(default access right)". This is displayed when the access right in a lower level element is not the same as the default access right. The default access right is specified as standard for newly created lower-level elements.

Selecting and configuring symbols

1. First select the symbol group of a CPU or a structure element in the navigation area.
2. Then select one or more symbols with the mouse cursor in the content area.
3. Configure the properties of the selected symbols in the lower part of the dialog as described below.

"Elements in active branch" display

The meaning of the display is as follows:

- Symbols: Total number of elements in the selected branch or in the selected structure element (a selected structure element is included in this number).
- Array elements: Number of elements in the array
  - Value shown on the left: Number of elements selected as visible
  - Value shown on the right: Total number of elements

Configurable properties

- "Visible" option
  If you disable this option, the selected symbol is excluded from the download to the OPC server.
- "Download only visible symbols" option
  If you select the option, only the symbols marked as "visible" are adopted in the system data of the OPC server and are therefore visible to the OPC client. The option reduces the time for the download to the OPC server considerably if large numbers of symbols are marked as "not visible".

- Access right
  From the drop-down list, you can assign the following access permissions to the selected symbols:
  - Read-write: Read and write
  - Read: Read
  - Write: Write
  - None: No access
    The access permissions selected here have priority over the access permissions configured for the OPC items in access protection.

  Note

  Note that the setting for "Access right" is not exported and therefore also not imported, but the current setting is always displayed.

- Connection
  From the drop-down list of the configured connections, select the S7 connection via which the symbolic variable access will be made.
- "Edit EU Lo / Hi" input boxes - Specify tolerance range for measured values (EU Lo and EU Hi limit values)
  Here, you specify limit values for calculating a deadband for a selected OPC item. The input boxes are enabled under the following condition:
  - You have only selected a single symbol;
  - An elementary data type is assigned to the symbol that permits a numeric representation (INT, DINT, REAL, BYTE, WORD, DWORD).
    Note:
    This entry is possible only for OPC servers as of version 6.4.
    See also Entering limit values for EU Lo and EU Hi (Page 1508)

"Access path" display

The storage path of the selected object in the current STEP 7 project is displayed in the navigation area.

Special features

- Bit arrays are not adopted
  Symbols that include data of the "bit array" type are not adopted in the system data of the OPC server. This restriction applies although this symbol is displayed for selection. This restriction applies regardless of whether you set the "visible" option (see above) for these symbols.
- The data types "Array of UDT" and "Array of Struct" are displayed differently
  With symbols of the complex data type "Array of UDT" or "Array of Struct", the structure of the "UDT" or "Struct" data type is only displayed once in the left-hand area of the dialog. The "Array" entry in the "Data type" box on the right indicates that this involves an array. Only the first array element is displayed with its addresses and therefore the address assignment of the individual elements.
  In contrast to this, other tools, such as the Symbol File Configurator display multiple array elements.
  The data structures involved are nevertheless identical.
  Note that the setting for the "Visible" option described above applies to all elements of an array including array elements that are not displayed.

Entering limit values for EU Lo and EU Hi

Entering the tolerance range for measured values (high and low limit values EU Lo and EU Hi)

Here, you specify limit values for calculating a deadband for a selected OPC item (for the meaning, see below).

You can specify numeric values as integers, floating points or in exponential notation (scientific notation)

Note: As separator use the "." character instead of the "," character.

- EU Lo (Engineering Unit Low)
  Low limit
- EU Lo (Engineering Unit Low)
  High limit

the following must apply: EU Lo <= EU Hi

If an input box is left empty it is automatically completed with "0".

Make sure that you keep to the value ranges depending on the data type. Range of values of the data types (Page 1509)

- Examples:
  - EU Lo = 10 EU Hi = 25
  - EU Lo = 4.5 EU Hi = 14.5
  - EU Lo = 1.2e-2 EU Hi = 2.5e-2
- Notes:
  - The tolerance range ("Percent Deadband") applies to a group of OPC items.
  - This entry is possible only for OPC servers as of version 6.4.

Meaning of the low limit value EU Lo and high EU Hi

Fo rmeasured values, you can specify deadbands for measured value acquisition. Within such a percent deadband, a measured value can change without the change being signaled.

To achieve this, a relative value is calculated from the specified parameters according to the following formula. This value specifies the maximum number of units by which a physical value can change up or down without a value change being signaled.

A = PD / 100 x (EU HI – EU Lo)
---

where:

A = amount of the deviation (up or down)

PD = Percent Deadband / %

EU Lo = low limit value

EU Hi = High limit value

Example

With a temperature measurement, you assume that there is a typical deadband for the measured value in the range between 30°C and 40°C. As the percent deadband, set 10%.

You would therefore select the following values:

EU Lo = 30

EU Hi = 40

PD = 10%

This results in: A = 10 / 100 x (40 – 30) = 1

A temperature change is therefore signaled when the temperature is more than 1 °C above or below the previous measured value.

See also

Symbol configuration (Page 1505)

Range of values of the data types

The following ranges of values apply to the data types used:

BYTE= 0 .. 256

INT= -32768 ... 32767

WORD= 0 ... 65535

DWORD= 0 ... 4294967295.0

DINT= -2147483648 ... 2147483647.0

REAL POS UPPER = 3.402823e+38

REAL POS UNDER = 1.175495e-38

REAL NEG UPPER = -1.175495e-38(negative high limit)

REAL NEG UNDER = -3.402823e+38 (negative low limit)

See also

Entering limit values for EU Lo and EU Hi (Page 1508)

Syntax - OPC item-specific access rights

You can enter access rights for one or more OPC items.

The default rights defined for the other OPC items are not affected.

The syntax for OPC item-specific access rights is defined as follows:

<OPCItem>=<rights>

where:

- <OPCItem>
  You specify one or more OPC Items according to the syntax defined in the OPC documentation. It is also possible to use aliases.
  The following placeholders can be used:
  * Any number of irrelevant characters
  ? exactly one character is irrelevant
- <rights>

Read-write	Read and write
Read	Read
Write	Write
None	No access

Rules for evaluation

- The specific rights assigned here have priority over the default rights you assigned in the input area Access protection.
- The specific rights assigned here are evaluated in the order shown here. If there is a multiple assignment for an OPC item, the last assignment is always valid.
  Example:
  DP:[CP 5611]Slave040_QB*=RW
  DP:[CP 5611]Slave040_QB1=R
  DP:[CP 5611]Slave040_QB2=W
  DP:[CP 5611]Slave040_QB1=W
  DP:[CP 5611]Slave040_QB1*=R
  Due to this entry, the following access rights are effective:
  DP:[CP 5611]Slave040_QB2=W
  DP:[CP 5611]Slave040_QB1=R
  Tip: Please read the detailed description of the syntax in the OPC documentation listed below. There, you will find a series of examples - also for applications such as PROFINET IO.
- Leading zeros in items and in the definitions made here for the access rights are always ignored.
  The following example illustrates the advantage of this:
  You configure the following access right: Slave004_QB0=R
  You have therefore implicitly assigned the access right to the following items:
  - Item 1: Slave00004_QB000
  - Item 2: Slave4_QB00,1 (note: the syntax ",1" is demanded by the OPC Scout)

  When using the wildcards "*" and "?", this means the following:
  Do not use these wildcards at positions at which a leading zero can precede an item!
  Example:
  The definition Slave0?9M06_AB1=R must be replaced by the two following definitions:
  - Slave?9M06_AB1=R
  - Slave9M06_AB1=R

Further information

Read the detailed description of the syntax in the OPC documentation. There, you will find a series of examples - also for applications such as PROFINET IO.

OPC alarms:

Configuring OPC alarms

Reference

Settings for OPC server / "Properties > S7 > OPC alarms" parameter group.

Language / message text language setting

For OPC alarms, you specify which language versions of the text library for alarm texts (block- and symbol-related alarms) will be downloaded to the OPC server.

Using the two check boxes, you can set the following:

- The default language (specified as the default language for display devices):
  When the default language for alarms is one of the following STEP 7 default languages: German (Germany), English (USA), French (France), Spanish (Traditional Sort), Italian (Italy). Otherwise the check box is disabled.
- English (USA)
  (if English (USA) is installed as the default language)
  If the default language for display devices is already English (USA), the check box is disabled.

You can select one or both options. If you select both options, a check is made as to whether multiple language text storage was selected in "Language for display devices" in the project. If this is the case, the relevant text libraries are downloaded when you download to the PC station.

Time stamps for alarms

In this parameter group, you decide which entity will set the time stamp of an alarm. If you leave time stamping to the CPU, you obtain alarms with the exact time of their occurrence. With the option of specifying a time offset, you can also register alarms occurring, for example, in plants with different time zones with a common reference time in the acquisition system.

- CPU time stamp
  Alarms are given the time stamp of the signaling CPU. Exception: StatepathEvents are locally generated alarms and are given the PC time (UTC).
- CPU time + offset
  Alarms are given the time stamp of the CPU sending the alarm; a selectable time offset accurate to a second is added to the time stamp. Exception: StatepathEvents are locally generated alarms and are given the PC time (UTC).
  Range of the time offset: -23:59:59 to + 23:59:59
- PC time (UTC)
  Alarms are given the time stamp (UTC) at the time of arrival at the receiving PC station (OPC server).

Assignment table for OPC alarms

Assignment table for OPC alarms

Reference

Settings for OPC server / "Properties > S7 > Assignment table OPC alarms" parameter group.

Meaning

The alarm classes configured in the project in "Common data > Alarm classes" are displayed in a table. The OPC alarm category "OffNormal" is assigned to the message classes as default. If necessary, assign a different OPC alarm category.

S7 connections for access to optimized data blocks with S7-1200/S7-1500

To allow data access to optimized data blocks of the OPC server V12 to data blocks of unsure types, bit memory, inputs, outputs, counters and timers of a SIMATIC S7-1500 and SIMATIC S7-1200, you can configure a S7 connection to the CPU.

S7 connections for access to optimized data blocks in the S7-1200/S7-1500 are available only using Industrial Ethernet.

Requirements

- STEP 7 Professional V12 SP1 or higher
- OPC server V12 or higher
- S7-1500 or S7-1200 V4.0 or higher

Connection names for OPC server V12 and S7 connections

When you create an S7 connection for an OPC server, STEP 7 proposes a valid S7 connection name. You can change the S7 connection name at any time. Some characters and connection names are used by the system and cannot be used when assigning the connection name. STEP 7 checks for invalid characters in the connection name automatically.

Connection table

In the table network view in the "Connections" tab, the connections you have created and their parameters are displayed in the connection table. In the "Partner" column, you can change the communications partner to the OPC server V12.

Parameters of special connection properties

In the Inspector window of the selected S7 connection configured at one end (one-way), go to "Properties > Configuration > Special connection properties", the "One-way" and "Active connection establishment" check boxes are selected. You cannot make any changes in the "Special connection properties" parameter group.

Address details

In the Inspector window of the selected S7 connection configured at one end in "Properties > Configuration > Address details" you will see the TSAPs of the communications partners in the connection resources. The local TSAP is structured as follows:

SNOPCCxxxxyyyyyy

- SNOPCC = SIMATIC NET OPC Configured connection
- xxxx = index of the OPC server V12
- yyyyyy = connection ID of the S7 connection

The partner TSAP is "SIMATIC-ROOT-OTH".

You cannot make any changes in the "Address details" parameter group.

OPC parameters

In the Inspector window of the selected S7 connection configured at one end in "Properties > Configuration > OPC" you have the following configuration options:

- Enabling optimized access
- Configuration of the buffer size for optimized access to tags
- Administration of the optimized access to tags by entering the access password

Buffer size:

It may be useful to change the default buffer size in specific applications with particularly large numbers of data tags on a CPU (S7-1500 or S7-1200 V4.0). Access to contiguous data tags within a data area is optimized to improve performance. Optimum results are achieved with buffer sizes that include at least the majority of the contiguous data tags that are accessed regularly and often.

Example: Many data tags between bytes 1000 and 1999 are read regularly from the data block "db10", more seldom also data tags in byte 30000. In this case, a buffer size of 1000 bytes is adequate.

Connection resources

With configured S7 connections for an OPC server V12 or higher, to allow data access to optimized data blocks on a PLC S7-1500 or S7-1200 V4.0 or higher, the connection resources required for this are displayed on the PLC page. You will find the display of the required connection resources in the properties of the relevant CPU in the Inspector window in "Properties > General > Connection resources".

For S7 connections for data access to optimized data blocks of an OPC server V12 or higher, the resources of such an S7 connection are not assigned to the "S7 communication" row on the PLC but to the rows "Other communication" (S7-1500) or "Free available connections" (S7-1200 V4.0 or higher). For a configured S7 connection for an OPC server V12 to an S7-1500 or S7-1200 V4.0 or higher, more than one connection resource is required.

Replacing a OPC server

With the "Change device" shortcut menu of a selected OPC server you have the option of replacing an OPC server V12 which allows access to optimized data blocks on S7-1500 or S7-1200 V4.0 with an OPC server < V12 and vice versa.

If S7 connections had already been configured for the OPC server you are replacing, during the replacement of the OPC server you will receive a message that the existing S7 connections will be deleted. Following replacement, these need to be recreated.

Display of an S7 connection configured at one end (one-way) between an OPC server V12 and an S7-1500 or S7-1200 V4.0 or higher

When an OPC server V12 has sent a successful job for data access to optimized data blocks on an S7-1500 or S7-1200 V4.0 or higher, this connection is displayed in the online connection table of the corresponding PLCs in STEP 7.

Such connections are also shown in the Inspector window "Connection information" of the selected online CPU: the online connection resources used on the PLC are displayed here.

Redundant OPC UA server

Redundant OPC UA server - basics

Transparent OPC UA redundancy servers

The following figure shows how the OPC UA servers are embedded in a redundancy group and configured for the transparent OPC UA redundancy solution.

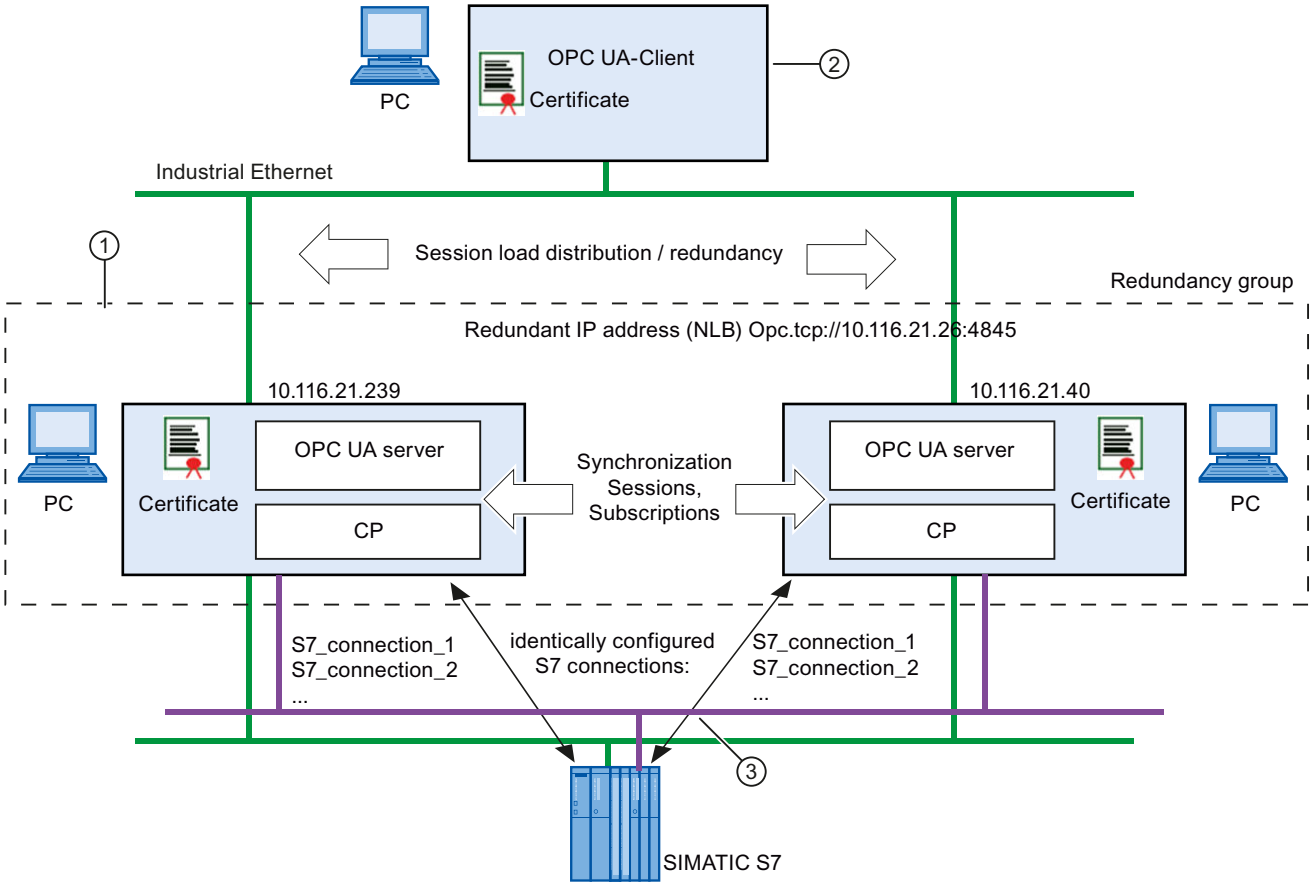

① You configure a PC station for each redundant OPC UA server. The redundant PC stations assigned to an OPC UA client form a redundancy group. This assignment is made by the uniform configuration of the "IP address of the redundancy group" for the OPC UA servers.

② The OPC UA client that is using the redundancy is located on a higher-level PC station.

The higher-level PC station does not need to be configured for use of OPC UA redundancy in the STEP 7 project.

③ The communication between the OPC UA servers and the programmable controllers is via multiple, identically configured S7 connections.

Depending on the configuration, S7 connections can be used via Ind. Ethernet or via PROFIBUS.

Setting a SIMATIC NET OPC server as S7 OPC UA server for transparent OPC UA redundancy

You can achieve transparent redundancy to be used by the higher-level OPC UA client by following the steps outlined below during configuration:

1. To operate PC stations with redundant SIMATIC NET OPC UA servers, create a component of the type OPC server >= V8.1 for each station. Enable S7 UA redundancy for these OPC servers.
2. They adopt the virtual IP address. You specified this virtual IP address previously in the network load balancing manager (NLB) for both PC stations. The virtual IP address is mapped to the real IP addresses of the OPC UA server.
3. You specify a port via which the two OPC UA servers can communicate so that they can synchronize their data;
4. You configure identical S7 connections for both OPC UA servers for communication with the connected automation systems;
5. Authentication certificates are generated automatically and assigned to the PC stations. If necessary, assign the two redundant PC stations newly generated authentication certificates.

Consistency check

When you save and compile the configuration, a consistency check is run. In relation to OPC UA redundancy, the following checks are made:

- The server-to-server addresses within a redundancy group must be unique. A common IP address of a redundancy group may also only be used once within a project.
- Each member of a redundancy group must have the same S7 connections with the same connection names and the same communications partners.
- The hardware configuration of the PC stations in the redundancy groups must be identical. In other words, the modules used for the redundancy function must be identical in the redundant PC stations.
- The parameters of the S7 connections must be identical; these include connection establishment, alarms, optimizations, PDU size, number of parallel jobs, monitoring times.
- The S7 settings of the OPC UA servers must be identical; these include cycle time, access rights, S7 alarms and S7 symbols.

Note

Recommendation: Permanently established S7 connections

We strongly recommend that you configure active, permanently established S7 connections on the OPC UA servers. This achieves shorter failover times if there is a server failure.

Procedure / recommendation

The parameter assignments of the redundant OPC UA servers must match each other. When you save and compile the configuration, a consistency check is run.

By using the STEP 7 options for copying, you reduce the risk of inconsistent information. Either copy the entire PC station or only the OPC server. If you have already configured S7

connections, these are also copied. The connections in the copied PC station must still be assigned the relevant connection partner.

See also

Configuring S7 UA redundancy tab (Page 1518)

Configuring S7 UA redundancy tab

Reference

Setting a redundant OPC UA server in the "Properties > General > S7 UA Redundancy" parameter group

Transparent OPC UA redundancy

In this parameter group, you configure the PC station as a redundant OPC UA server for operation with transparent OPC UA redundancy. Here, use the SIMATIC NET OPC server version 8.1

The redundant PC stations assigned to an OPC UA client belong to a redundancy group.

In this parameter group, follow the steps outlined below:

1. Select the "Enable redundancy" option.
2. Enter the IP address of the redundancy group. Here, you have the following options:
   - You can either enter the IP address directly or accept or modify the proposed IP address. Select this option if you want to create a new redundancy group.
   - Click the "Assign" button
     Select this option to assign the OPC server to an existing redundancy group.

     Note

     After entering or assigning, note the "IP addresses of the other members of the redundancy group" table at the end of the parameter group. You can see which OPC servers are assigned to the same redundancy group.
3. Specify the redundancy port for communication with the OPC UA client; if appropriate, accept the default setting.
4. Set the parameters for the server-to server communication.
5. If necessary check the authentication certificate assigned to the OPC server using the "Display..." button.

Parameters / input areas / buttons

- "Enable redundancy" option
  By selecting the option, you enable the parameter assignment option for the OPC UA server functionality.
- "Assign..." button
  You open the dialog in which the redundancy groups that have already been created can be selected.
- "IP address of redundancy group" parameter
  Virtual IP address via which the OPC UA client reaches the redundant OPC UA server (the redundancy group).
  You specify this virtual IP address in the network load balancing manager (NLB) for both PC stations. The virtual IP address is mapped to the real IP addresses of the OPC UA server.
- "Redundancy port" parameter
  Port via which communication with the OPC UA client is handled.
  Default: 4845
- "Server-to-server port" parameter
  Port via which the communication for synchronization with the OPC UA servers in the redundancy group is handled.
  Default: 4846
- "Server-to-server interface" parameter
  Here, select the interface to the subnet via which the synchronization with the OPC UA servers in the redundancy group is handled.
- "Certificate" input area / "Display..." button
  STEP 7 automatically generates a certificate that it is transferred to the OPC UA client during connection establishment. The OPC UA client uses this for authentication during the subsequent communication.
  The certificate is also used to transfer the key for data encryption.
  The private key remains on the server and is used for encryption. With the corresponding public key it received from the server with the certificate, the client can then encrypt messages of the server.
  You can display the currently valid certificate here and check it.
  Note:
  The certificate display does not recognize whether or not a certificate is already being displayed. Each time you click the "Display..." button, a new certificate display is opened.
- "Certificate" input area / "Update" button
  You instruct STEP 7 to create a new certificate.

IP addresses of the other members of the redundancy group

Display of the other members of the redundancy group. A redundancy group is made up of the PC stations that are assigned the same "IP address of the redundancy group".

Note

Maximum number of members in a redundancy group

In the current version of STEP 7 V12, **2** OPC UA servers are supported within a redundancy group.

See also

Redundant OPC UA server - basics (Page 1516)

Assigning an S7-UA redundancy group

Reference

Setting a redundant OPC UA server in the "Properties > General > S7 UA Redundancy > S7 UA Redundancy Master" parameter group

Assigning an existing S7-UA redundancy group

The table shows the redundancy groups available and selectable in the STEP 7 project along with their IP addresses.

If you have assigned a new IP address to the current OPC server, the following entry appears instead of this assignment:

- "(own redundancy group <IP address>)"

Follow the steps outlined below:

1. Select the required redundancy group.
2. Confirm the dialog with the "OK" button.

The IP address of the selected redundancy group is then adopted in the "S7-UA Redundancy" parameter group.

See also

Redundant OPC UA server - basics (Page 1516)

PLC alarms for OPC server

SIMATIC NET OPC server - Recommendations on the configuration of CPU messages

If you configure PLC alarms for an OPC server of a PC station, their consistent configuration will not be verified by STEP 7. To ensure error-free functioning of the PLC alarms, note the following recommendations:

- Only ever configure one S7 connection with interrupt capability per PLC for receiving alarms.
- Only use one language for alarm texts in a given project. The use of multi-language alarm texts is not supported.
- For all alarms, avoid inputting user-specific sources (additional text 1). The uniqueness of the alarms is automatically ensured if the default sources are used.

Editor for OPC program alarms

General

The editor for OPC program alarms is available as of OPC server V14. This helps you to configure OPC program alarms of an S7 connection between a PC station or OPC server and an S7-1500/S7-1500S so that the display in the user-specific view, for example corresponds to your plant topology. At the same time you can add, rename or delete areas, sources and conditions in the user-specific view and can also manage them in different langages. The structure and the names of the OPC program alarms in the default view are preset based on the tree structure from the STEP 7 project. The texts of the OPC program alarms are configured in the STEP 7 configuration of the S7-1500/S7-1500S. The created structure of the OPC program alarms in the user-specific view is displayed identically in the OPC UA client with browsing interface after you have compiled and downloaded the configuration of the PC station.

Default view

The available OPC program alarms are displayed in the default view. With a high number of OPC program alarms, coordination between the PLC configuration and OPC configuration is advisable to make management clearer.

The requirement is an S7 connection in whose OPC properties the check boxes "Maintain connection permanently" and "Receive program alarms" are enabled (select "Network view" > "Connections" tab > "S7 connection" > "General" tab > "OPC" > "Connection establishment" or "Alarms").

Note

A If after configuration in the editor you disable the check boxes "maintain connection permanently" or "Receive program alarms" in the OPC properties of the S7 connection, the OPC program alarms configured up to then are no longer displayed in the default view until the options are enabled again.

Note

If you do not configure OPC program alarms in the user-specific view and download the STEP 7 project to the PC station, the OPC program alarms in the OPC UA client will be displayed identical to the preset tree structure of the STEP 7 project.

User-specific view

In the user-specific view, you have the option of structuring OPC program alarms to meet your requirements. Using the context menu for "S7OPTAREAS:" you define the areas and sources according to your plant topology and add the conditions in the relevant sources.

Between the default view and the user-specific view, arrow symbols show you the graphic assignment of the conditions. If a condition is selected in the the user-specific view, you will see the origin and target based on the arrow symbol. Depending on which node is selected or the extent to which the tree structures are expanded or collapsed the position of the arrow symbol changes. With collapsed tree structures, the arrow symbols can e displayed merged. The following options exist for the graphic assignment of the conditions by arrow symbols:

- The tree structures are fully expanded in both views. By selecting a condition the direct 1:1 assignment between the origin and destination of the condition is diplayed.
- The area or the source in which the condition (origin) is contained is collapsed in the default view. A condition (destination) is selected in the user-specific view. From the area or the source the arrow symbol points from the default view to the condition (destination) in the user-specific view.
- The area or the source in which the configured condition (destination) is contained is collapsed in the user-specific view. A condition (origin) is selected in the default view. From the area or the source the arrow symbol points from the user-specific view to the condition (origin) in the default view.
- A condition is selected in the default view that is not configured in the user-specific view. No arrow symbol is displayed, since no assignment is made or is possible.

As soon as you have configured a condition in the user-specific view, this is grayed out in the default view. If you delete a condition from the user-specific view, this is no longer grayed out in the default view and can be used again elsewhere in the user-specific view.

Note

If you delete OPC program alarms from the S7-1500/S7-1500S, these are no longer displayed in the default view. In the user-specific view, these OPC program alarms are marked red and are inconsistent. You can no longer use the inconsistent OPC program alarms and need to delete them manually.

Note

All the OPC program alarms configured on the PLC that are not configured in the user-specific view are nevertheless also loaded on the PC station when it is loaded. On the OPC client, these OPC program alarms not configured in the editor are contained in a special folder. The name of this folder corresponds to the name in the PLC configuration.

Meaning of the icons used

In the editor for OPC program alarms there are the following icons that are used to display areas, sources and conditions in both views:

Area:

Every station name (e.g. Station1), device name (e.g. Controller1) or group (e.g. Group1) is represented in the default view as an area. The resulting tree structure is decided by the STEP 7 project structure.

The area must first be created in the user-specific view because there is no area present at the start. Several areas can be defined.

Source:
The source in the default view consists of the station name (e.g. Station1), device name (e.g. Controller1) and the symbolic name of the data block of the PLC alarm (e.g. Motor1).

A source must be added to an area in the user-specific view. Several sources can be added to an area.

Condition:
The condition in the default view consists of the name of the configured OPC program alarm. Example: alarm_motor1

Conditions are dragged from the default view to the required source within the user-specific view.

"General" Tab

The "General" tab displays the properties "Condition", "OPC severity" and "OPC event type" of the selected OPC program alarm. The input box "Condition" can be edited.

"General" > "OPC alarm instances" tab

In the "General" > "OPC alarm instances" tab, you obtain an overview of various properties of the OPC program alarm. These properties are read-only at this point and cannot be changed.

The following properties of the OPC program alarms:

- Name
  Specifies the name of the OPC program alarm
- ID
  Specifies the ID of the OPC program alarm.
- PLC name
  Specifies the name of the S7-1500/S7-1500S on which the OPC program alarms are configured.
- Message text
  Specifies the message texts configured on the S7-1500/S7-1500S.
- Condition
  Specifies the name of the condition
- OPC severity
  Specifies the OPC severity of the OPC program alarm.
- Area
  Specifies the name of the area of the OPC program alarm

"Languages & Resources" dialog box

In the "Languages & Resources" dialog box you can set the editing language and the reference language in all languages supported by "STEP 7 Professional (TIA portal)". This gives you the option, for example, of editing and saving in German and when you then later switch over to the English language, you can rename and store the names of the OPC alarms in English. Depending on which language you now set the names of the OPC program alarms will always be displayed to you in the "correct" language. New names of the OPC program alarms that have not yet been translated are enclosed in the characters "<" and ">" in the user-specific view.. In the default view the same names are used for all languages as was defined in the configuration in the PLC.

You can add further project languages with "Project tree" > "Languages & resources" > "Project languages".

In e.g. "Project tree" > "Languages & resources" > "Project texts" all project texts including the alarm texts of the conditions can be displayed and edited. In the same Inspector window, project texts can also be imported or selected project texts, here conditions, can be exported.

Note

For the translation of the names of the OPC program alarms into the various languages, for easier handling the described import or export function is recommended.

Typical procedure for the first use of the editor

Proceed as follows:

1. Create OPC program alarms in the S7-1500/S7-1500S for the STEP 7 project. With the S7 connection, the OPC properties "Maintain connection permanently" and "Receive program alarms" must be enabled.
2. Open the editor for OPC program alarms by double-clicking on "Editor for OPC program alarms" in the tree tree structure of the project tree of the relevant OPC server of the STEP 7 project.
3. In the user-specific view select the command "Add area node" from the shortcut menu of "S7OPTAREAS:"
4. Assign the newly created area any name (forbidden characters "\" and control characters). In keeping with your plant topology you can place other areas in the area.
5. Select the command "Add source" from the shortcut menu of the relevant area.
6. Assign the source any name (forbidden characters: control characters).
7. Drag the conditions from the default view to the required position in the user-specific view.
8. Assign the condition any name (forbidden characters: control characters).
9. If necessary, specify the editing and reference language in the "Languages & Resources" dialog box. Note that the untranslated names in pointed brackets (<>) will be translated.
10. Save and compile or load the new configuration in the PC station and in the PLC..
11. Open an OPC UA client that has a browsing interface.
12. Enable and monitor the configured OPC program alarms.

Note

If you select a node that has already been created, you can use the keys "x", "+" and "-" in the numeric keypad to speed up opening and closing of the lower-level tree structure.

- Multiplication kex "x": Opens the entire lower-level tree structure of the selected node.
- Addition key "+": Opens only the selected node.
- Subtraction key "-": Closes the entire lower-level tree structure of the selected node.

This functions both in the default view and in the user-specific view.

Typical procedure for further use of the editor

1. Open the editor for OPC program alarms by double-clicking on "Editor for OPC program alarms" in the tree tree structure of the project tree of the relevant OPC server of the STEP 7 project.
2. Analyze the actual situation of the existing configuration.
3. Decide what you want to reconfigure.
4. Make the changes accordingly.
5. Save and compile or load the new configuration in the PC station and in the PLC..

6. Open an OPC UA client that has a browsing interface.
7. Enable and monitor the configured OPC program alarms.

Consistency check

The configuration in the editor for OPC program alarms is checked for consistency before it is downloaded to the PC station. During this, the following compilation errors or warnings can be triggered:

Compilation error or warning	Remedy
Error: The OPC program alarms of the specified PLC are connected via several connections.	The OPC program alarms may only be connected via one S7 connection.
Error: The combination of the specified source and condition is not unique in a language.	The combination of source and condition must be configured to be unique.
Error: The OPC program alarm in the specified path contains invalid characters in the specified language.	Only permitted characters may be used:
Error: The OPC program alarm in the specified path contains at least one area node, a source or a condition with invisible ANSI code characters in the specified language.	Only permitted characters may be used:
Error: For the OPC program alarm in the specified path at least one area node, a source or a condition is not translated into the specified language.	All translations must exist for the languages used.
Error: The OPC program alarm in the specified path is no longer valid because there is no S7 connection with valid alarm properties to the specified PLC.	A suitable S7 connection needs to be created or the named OPC program alarm needs to be deleted in the user-specific view.
Error: The OPC program alarm in the specified path is no longer valid because the corresponding PLC alarm instance no longer exists.	The named OPC program alarm needs to be deleted in the user-specific view.
Warning: The specified alarm text is not translated in the specified language.	The preset text must be adapted to the specified language.
Warning: The specified OPC program alarm on the specified path is not configured in the user-specific view, but the OPC server uses at least one OPC program alarm that was configured in the user-specific view.	If necessary, configure OPC program alarm shown in the warning in the user-specific view.

Note

Displayed compilation errors must be eliminated efoe you can download the configuration to the PC station.

Compilation warnings can if applicable be ignored, in other words the configuration can be downloaded to the PC station without rectifying the warning.

1.4.5 Configurations for point-to-point connections (S7-1500)

1.4.5.1 Configuration for point-to-point links (S7-1500)

Introduction (S7-1500)

Overview of the communication modules (S7-1500)

Automation systems encompass a wide range of components. These also include communication modules. A simple possibility of data exchange is provided by serial communication via point-to-point connections.

Customizing to a wide range of communication partners is possible by setting the communication parameters at a lower layer of the OSI layer model (see section Transmission security (Page 1532)).

Communication through point-to-point connection with S7-1500, ET 200MP and ET 200SP takes place exclusively by means of communication modules (CM) with serial interfaces.

SIMATIC S7 offers a number of modules that provide the physical interface and fundamental protocol mechanisms for this purpose.

- RS232: An interface that can coordinate the communication between the partners through additional accompanying signals.
- RS422/RS485: An interface that allows longer lines through the use of differential voltages as transmission technology and also enables structures with more than 2 devices through a bus structure (RS485).

Instructions that carry out the coordination between the CPU and CM (Communication Module) are available to transfer data from the CPU to the respective modules. They inform the user program about a successful transfer or the receipt of new data (in systems without a SIMATIC CPU users have to program the function of these instructions themselves).

The function and use of the PtP communication modules is described in this function manual.

Overview of components and order numbers

Tabular overview of communication modules and their application suitability

Communication module	S7-1500	ET 200MP	ET 200SP	Article number
CM PtP RS232 BA [1]	X	X	-	6ES7540-1AD00-0AA0
CM PtP RS422/485 BA	X	X	-	6ES7540-1AB00-0AA0
CM PtP RS232 HF [2]	X	X	-	6ES7541-1AD00-0AB0
CM PtP RS422/485 HF	X	X	-	6ES7541-1AB00-0AB0
CM PtP (ET 200SP)	-	-	X	6ES7137-6AA00-0BA0

[1] BA = Basic

[2] HF = High Feature

Overview of components and interfaces

Tabular overview of communication modules and their functions.

Communication module	Interface	Protocols					Connection technology	
		Freeport	3964(R)	Modbus Master	Modbus Slave	USS-Master	D-Sub 9-pin	D-Sub 15-pin
CM PtP RS232 BA	RS232	X	X	-	-	X	X	-
CM PtP RS422/485 BA	RS422	X	X	-	-	X	-	X
	RS485	X	-	-	-	X	-	X
CM PtP RS232 HF	RS232	X	X	X	X	X	X	-
CM PtP RS422/485 HF	RS422	X	X	X	X	X	-	X
	RS485	X	-	X	X	X	-	X
CM PtP (ET 200SP)	RS232	X	X	X	X	X	ET 200SP BaseUnit 1)	
	RS422 2)	X	X	X	X	X		
	RS485	X	-	X	X	X		

1) BaseUnit with terminals instead of D-Sub; assignment depending on physical transmission properties

2) The CM PtP communication module can also be used for multipoint connection in RS422 mode

Overview of components and data transmission rates

The communication modules can send and receive data with different data transmission rates. The table below shows the assignment to the individual communication modules.

Communication module	Data transmission rate in bps										
	300	600	1200	2400	4800	9600	19200	38400	57600	76800	115200
CM PtP RS232 BA	X	X	X	X	X	X	X	-	-	-	-
CM PtP RS422/485 BA	X	X	X	X	X	X	X	-	-	-	-
CM PtP RS232 HF	X	X	X	X	X	X	X	X	X	X	X
CM PtP RS422/485 HF	X	X	X	X	X	X	X	X	X	X	X
CM PtP (ET 200SP)	X	X	X	X	X	X	X	X	X	X	X

Overview of components and receive buffer size

Each communication module has a buffer to temporarily store received frames. The table below shows the assignment of the maximum size of an individual frame as well as the size of the memory for the individual communication modules.

Module	Receive buffer size KB	Max. frame length KB	Bufferable frames
CM PtP RS232 BA	2	1	255
CM PtP RS422/485 BA	2	1	255
CM PtP RS232 HF	8	4	255
CM PtP RS422/485 HF	8	4	255
CM PtP (ET 200SP)	4	2	255

Accompanying signals and data flow control

- Software data flow control with XON/XOFF
  The Freeport protocol supports data flow control with XON/XOFF via the RS232 and RS422 interfaces.
- Hardware data flow control with RTS/CTS
  The Freeport protocol supports data flow control with RTS/CTS via the RS232 interface.
- Automatic operation of accompanying signals
  The RS232 accompanying signals can be controlled with the Freeport, Modbus master and Modbus slave protocols by means of the RS232 interface. (Only available if hardware data flow control is activated.)

Protocols of the communication modules

You may set up a communication connection with different protocols, depending on the communication modules used:

- Freeport: Transmission of ASCII character strings without specified protocol format
- 3964(R): Communication between programmable logic controllers (master/master communication)
- Modbus RTU: Communication between programmable logic controllers (master/slave communication) The communication module can be the master as well as the slave.
- USS: Communication between a programmable logic controller and a drive (master/slave communication) Communication is tailored to the drive technology requirements. The communication module can only be master.

Overview of the processing steps (S7-1500)

Point-to-point connection

The system provides various networking options for the exchange of data between two or more communication partners. The simplest form of data interchange is via a point-to-point connection between two communication partners.

The communication module (CM) forms the interface between a programmable logic controller and a communication partner. Data is sent in serial mode via point-to-point connection with the communication module.

Configuring / parameter assignment

Configuring the communication module includes the arrangement of the communication module in the device configuration of STEP 7 (TIA Portal) as well as the settings of the specific protocol parameters in the properties dialog of the communication module (static configuration).

Programming

Programming includes the program-specific connection of the communication module to the corresponding CPU by means of the user program. You program the communication module with STEP 7 (TIA Portal).

Communication between CPU, communication module and a communication partner takes place through instructions. A number of instructions are available for the S7-1500 and S7-1200 automation systems. You can use these instructions to initiate and control communication in the user program as well as influence the configuration for runtime (dynamic configuration).

For more information, please refer to Overview of instructions (Page 1530) and the STEP 7 (TIA Portal) online help.

Overview of instructions (S7-1500)

Note

CPU quantity structures

The point-to-point instructions communicate with the communication modules by reading or writing data records.
When using the instructions, you should therefore check the configuration limits of the CPU for reading and writing data records.
If multiple instructions need to read or write data records simultaneously on a CPU, there may need to be a gap between the calls of each instruction by the user program.

Overview of the instructions

The communication protocols are implemented on the communication module. The protocol is used to adapt the interface of the communication module to the interface of the communication partner.

Communication between the CPU, the communication module and a communication partner takes place by means of special instructions and protocols that support the corresponding communication modules.

The instructions form the software interface between the CPU and the communication module. They must be called cyclically from the user program. Data transmission takes place across several cycles.

The instructions are part of STEP 7 (TIA Portal). The instructions are available in the "Instructions" task card under Communication > Communication processor. They apply to all listed communication modules if they support the required function.

Table 1-94 Instructions for PtP

Instruction	Meaning
Send_P2P	The instruction Send_P2P allows you to send data to a communication partner.
Receive_P2P	The instruction Receive_P2P allows you to receive data from a communication partner.

Instruction	Meaning
Receive_Reset	The instruction Receive_Reset allows you to clear the receive buffer of the communication module.
Port_Config	The instruction Port_Config allows you to dynamically assign basic interface parameters.
Send_Config	The instruction Send_Config (send configuration) allows you to dynamically assign serial send parameters of a port.
Receive_Config	The instruction Receive_Config (receive configuration) allows you to dynamically assign serial receive parameters of a port.
P3964_Config	The instruction P3964_Config (protocol configuration) allows you to dynamically configure the parameters of the 3964(R) procedure.
Signal_Get	The instruction Signal_Get allows you to read the RS232 accompanying signals.
Signal_Set	The instruction Signal_Set allows you to set the RS232 accompanying signals.
Get_Features	The instruction Get_Features allows you to read the extended functions supported by the communication module.
Set_Features	The instruction Set_Features allows you to activate the extended functions supported by the communication module.

Table 1-95 Instructions for Modbus

Instruction	Meaning
Modbus_Master	The Modbus_Master instruction allows you to communicate as Modbus master by means of the PtP port.
Modbus_Slave	The Modbus_Slave instruction allows you to communicate as Modbus slave by means of the PtP port.
Modbus_Comm_Load	The Modbus_Comm_Load instruction allows you to configure the port of the communication module for Modbus RTU.

Table 1-96 Instructions for USS

Instruction	Meaning
USS_Port_Scan	The instruction USS_Port_Scan allows you to communicate by means of the USS network.
USS_Drive_Control	The instruction USS_Drive_Control allows you to swap data with the drive.
USS_Read_Param	The instruction USS_Read_Param allows you to read parameters from the drive.
USS_Write_Param	The instruction USS_Write_Param allows you to change parameters in the drive.

Basics of serial communication (S7-1500)

Serial data transmission (S7-1500)

During serial data transmission, the individual bits of a character of information to be transmitted are sent successively in a defined sequence.

Bidirectional data traffic - operating mode

In the context of bidirectional data traffic, we distinguish between two operating modes for the communication module:

- Half-duplex operation
  The data is exchanged between the communication partners in both directions alternately. In half-duplex operation one communication partner is sending and the other communication partner is receiving at the same time. In the process one line is alternately used for sending or receiving.
- Full-duplex operation
  The data is exchanged between one or more communication partners in both directions simultaneously, which means you can send and receive data at the same time. This process requires one line for sending and one line for receiving.

Asynchronous data transmission

Serial data transmission is performed asynchronously with communication modules. The so-called time base synchronism (a fixed timing code used in the transmission of a fixed character string) is only upheld during transmission of a character. Each character to be sent is preceded by a synchronization impulse, also called start bit. The length of the start-bit transmission determines the clock pulse. The end of character transmission is formed by one or two stop bits.

Declarations

In addition to start and stop bits, additional declarations must be made between the sending and receiving partners before serial transmission can take place. These include:

- Data transmission rate
- Frame start and end criteria (e.g., character delay time)
- Parity
- Number of data bits (7 or 8 bits/characters)
- Number of stop bits (1 or 2)

Transmission security (S7-1500)

Transmission security plays an important role in the transmission of data and in the selection of the transmission procedure. Generally speaking, the more layers of the reference model are applied, the higher the transmission security.

Classification of existing protocols

The figure below illustrates how the protocols of the communication module fit into the reference model.

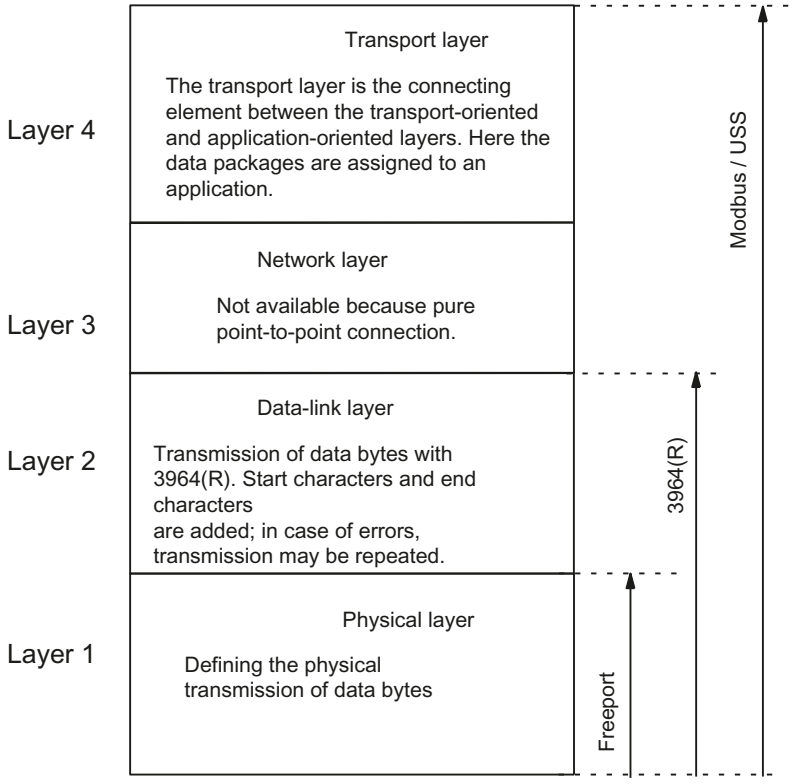

Figure 1-37 Classification of the existing protocols of the communication module in the reference model

Transmission security with Freeport

Transmission security when using Freeport:

- When data is sent with Freeport, there are no data security measures other than the use of a parity bit. This means data transmission with Freeport is very efficient, but data security is not guaranteed. A certain degree of data security can be achieved through parameter assignment of the frame start and frame end conditions.
- Using the parity bit ensures that the inversion of a bit in a character to be sent can be recognized. If two or more bits of a character are inverted, however, there is no guarantee that these errors are still detected.
- To increase transmission security, you can, for example, implement a checksum, a frame length specification, or configurable end conditions. These measures must be implemented by the user.
- A further increase in data security can be achieved by means of acknowledgment frames in response to send or receive frames. This is the case with high-grade protocols for data communication (ISO 7-layer reference model).

Transmission security with 3964(R)

The parity bit is used to increase data security; depending on the configuration, it completes the number of data bits to be transmitted to form an even or odd number.

Using the parity bit ensures that the inversion of a bit in a character to be sent can be recognized. If two or more bits of a character are inverted, however, these errors can no longer be reliably detected.

If parity is set to "none", no parity bit is transmitted. This setup reduces transmission security.

Two different procedures for data transmission can be used, either with or without a block check character:

- Data transmission without block check character: 3964
  Transmission security is achieved by means of a specified frame structure, frame breakdown, and frame repetitions.
- Data transmission with block check character: 3964R
  The high degree of transmission security is achieved by means of a specified frame structure and breakdown, frame repetitions, as well as inclusion of a block check character (BCC).

In this manual, the term 3964(R) is used when descriptions and notes refer to both data transmission modes.

Transmission integrity for Modbus and USS

The parity bit is used to increase transmission security; depending on the configuration, it completes the number of data bits to be transmitted to form an even or odd number.

Using the parity bit ensures that the inversion of a bit in a character to be sent can be recognized. If two or more bits of a character are inverted, however, this error can no longer be clearly detected.

If parity is set to "none", no parity bit is transmitted. This setup reduces transmission security.

The CRC (cyclic redundancy check) is additionally used with Modbus. With this method additional redundancy in the form of a so-called CRC value is added for each data block of the user data before data transmission. This is a check value calculated by using a specific procedure that can be used to detect any errors that may occur during transmission.

A BCC (block check character) is additionally used with USS. The block check character is formed during the receipt and is compared with the received BCC after the entire frame has been read in. If these two characters do not match, the frame is not evaluated. (If a character is incorrectly transmitted, an error is reliably detected. If an even number of characters are incorrectly transmitted, an error can no longer be reliably detected.)

RS232 mode (S7-1500)

The following communication modules support RS232 mode:

- CM PtP RS232 BA
- CM PtP RS232 HF
- CM PtP (ET 200SP)

In RS232 mode, data is sent via two lines. A separate line is available for the send direction and the receive direction. Simultaneous sending and receiving is possible (full duplex).

RS232 signals

In addition to the TXD, RXD and GND signals, the communication module provides additional RS232 signals when using RS232 hardware:

TXD	Output	Transmitted data; Interface is transmitting
RXD	Input	Received data; Interface is receiving
GND		Functional ground; isolated
DCD	Input	Data carrier detect; Carrier signal when connecting a modem. The communication partner signals that it recognizes incoming data.
DTR	Output	Data terminal ready; DTR set to "ON": Communication module switched on, ready for operation DTR set to "OFF": Communication module not switched on, not ready for operation
DSR	Input	Data set ready; DSR set to "ON": Communication partner signals readiness for operation DSR set to "OFF": Communication partner not switched on, not ready for operation
RTS	Output	Request to send; RTS set to "ON": Communication module ready to send; signals to the communication partner that there is data ready to send RTS set to "OFF": Communication module not ready to send
CTS	Input	Clear to send; Communication partner can receive data from the communication module (response to RTS = ON of the communication module) CTS set to "ON": Signals readiness to receive to the communication partner CTS set to "OFF": Signals "Not ready to receive" to the communication partner
RI	Input	Incoming call for connecting a modem (ring indicator)

After power on of the communication module, the output signals are in the OFF state (inactive).

You configure the operation of the DTR/DSR and RTS/CTS control signals in the user interface of the communication module.

The RS232 signals cannot be influenced in case of:

- configured data flow control "Hardware RTS always switched"
  (corresponds to automatic operation of the accompanying signals)
- configured data flow control "Hardware RTS always ON"
  (corresponds to hardware flow control with RTS/CTS)
- configured data flow control "Hardware RTS always ON, ignore DTR/DSR"

For more information, refer to chapter Handshake procedure (Page 1543).

Connecting cables

The following standard connecting cables of various lengths are available for connecting to a communication partner which also has a 9-pin D-sub male connector:

Article number	6ES7902-1AB00-0AA0	6ES7902-1AC00-0AA0	6ES7902-1AD00-0AA0
Product type designation	S7 connecting cable RS232		
Cable length	5 m	10 m	15 m

The table below shows the pin assignment for the 9-pin D-sub male connector of the respective communication module.

Male connector*	Pin	Designation	Input/ output	Required/optional for self-fabrication
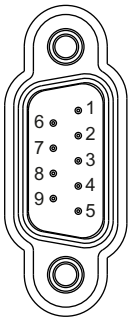	1	DCD	Input	Optional
	2	RXD	Input	Required
	3	TXD	Output	Required
	4	DTR	Output	Optional
	5	GND	—	Required
	6	DSR	Input	Optional
	7	RTS	Output	Optional
	8	CTS	Input	Optional
	9	RI	Input	Optional

* View from the front

The cable or the connector of the listed connecting cables are not available for order as separate items. If you fabricate your own connecting cables you must remember that unconnected inputs at the communication partner may have to be connected to open-circuit potential.

Note that you may only use shielded connector enclosures. A large surface area of the cable shield must be in contact with the connector enclosure on both ends.

CAUTION

Never connect cable shield with GND

Never connect the cable shield with the GND as this could destroy the interfaces. GND must always be connected on both sides (pin 5), otherwise the interface modules could be destroyed.

The figure below illustrates the cable for a point-to-point connection between a communication module and a communication partner.

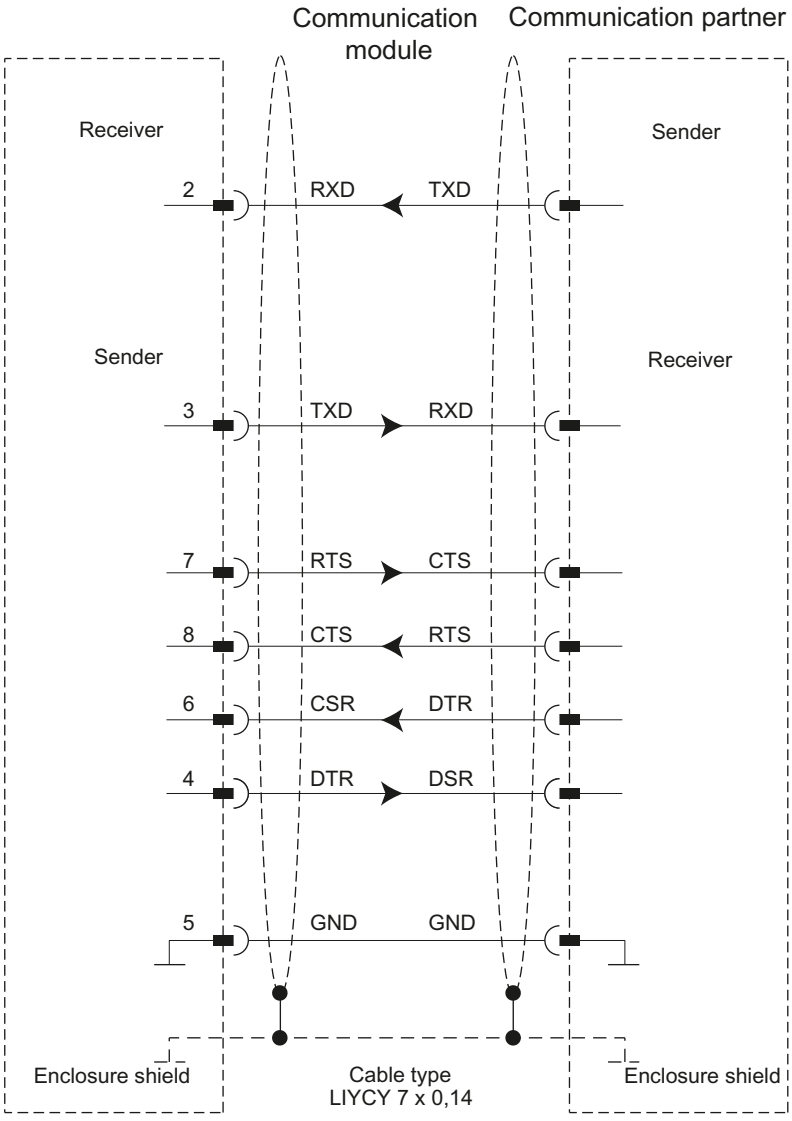

RS422 mode (S7-1500)

The following communication modules support RS422 mode:

- CM PtP RS422/485 BA
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

In RS422 mode, data is transmitted via two line pairs (four-wire operation). A separate line pair is available for the send direction and the receive direction. Simultaneous sending and receiving is possible (full duplex).

All communication partners must be capable of simultaneous operation of a sender and receiver unit.

The data can be exchanged simultaneously between two or more communication partners. In RS422 multipoint mode, only one slave may send data at any given time.

Interface operating modes

The following table is a summary of the interface operating modes for the various communication modules and protocols.

The communication module can be used in the following topologies in RS422 mode:

- Link between two nodes: Point-to-point connection
- Link between several nodes: Multipoint connection
  (only available with CM PtP (ET 200SP))

Operating mode	Description
Full duplex (RS422) four-wire operation (point-to-point connection)	Both devices have the same priority in this operating mode.
Full duplex (RS422) four-wire operation (multipoint master)	The communication module can be used as multipoint master.
Full duplex (RS422) four-wire operation (multipoint slave)	The communication module can be used as multipoint slave.

The following applies for a multipoint master/slave topology in RS422 mode:

- The sender of the master is interconnected with the receivers of all slaves.
- The senders of the slaves are interconnected with the master's receiver.
- Only the receiver of the master and the receiver of one slave have a default setting. All other slaves operate without default settings.

RS422 signals

The following signals are present on the communication module when using the RS422 hardware:

T (A) -	Output	Transmitted data
T (B) +	Output	Transmitted data
R (A) -	Input	Received data
R (B) +	Input	Received data
GND		Functional ground; isolated

Connecting cables

The following standard connecting cables of various lengths are available for connecting to a communication partner which also has a 15-pin D-sub female connector:

Article number	6ES7902-3AB00-0AA0	6ES7902-3AC00-0AA0	6ES7902-3AG00-0AA0
Product type designation	S7 connecting cable RS422		
Cable length	5 m	10 m	50 m

The table below shows the pin assignment of the 15-pin D-sub female connector of the respective communication module.

Socket*	Pin	Designation	Input/output
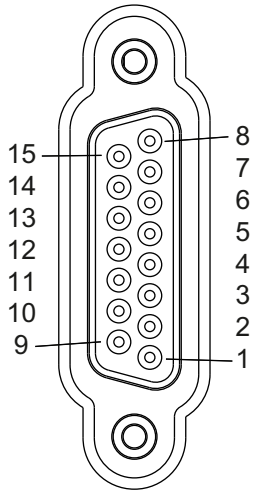	1	-	-
	2	T (A) -	Output
	3	-	-
	4	R (A) -	Input
	5	-	-
	6	-	-
	7	-	-
	8	GND	-
	9	T (B) +	Output
	10	-	-
	11	R (B) +	Input
	12	-	-
	13	-	-
	14	-	-
	15	-	-

* View from the front

The cable or the connector of the listed connecting cables are not available for order as separate items. If you fabricate your own connecting cables you must remember that unconnected inputs at the communication partner may have to be connected to open-circuit potential.

Note that you may only use shielded connector enclosures. A large surface area of the cable shield must be in contact with the connector enclosure on both ends.

CAUTION

Never connect cable shield with GND

Never connect the cable shield with the GND as this could destroy the interfaces. GND must always be connected on both ends (pin 8), otherwise the interface modules could be destroyed.

The figure below illustrates the cable for a point-to-point connection between a communication module and a communication partner.

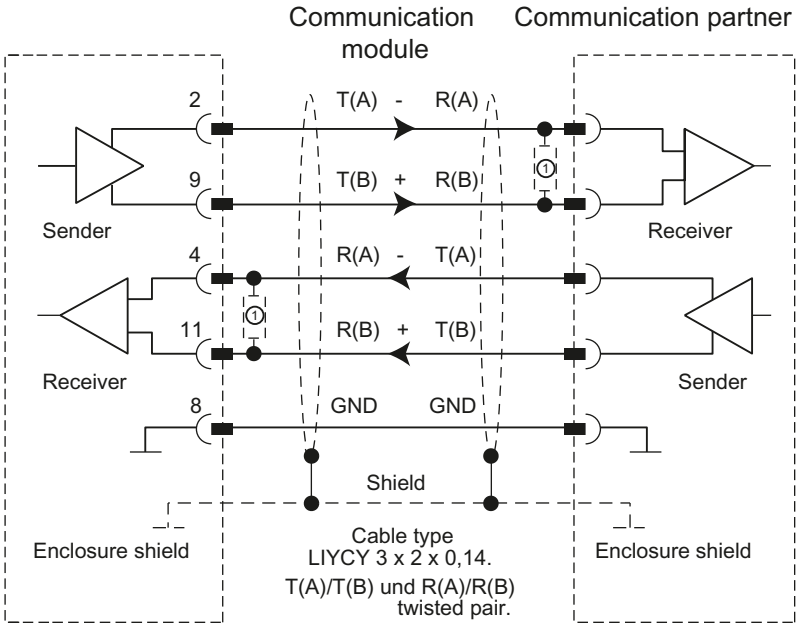

① For cable lengths longer than 50 m, you need to solder a terminating resistor of approx. 330 Ω at the receiver end to ensure interference-free data traffic.

Note

This cable type can be used in the following lengths for a communication module as communication partner: max. 1200 m at 19200 baud, max. 500 m at 38400 baud max. 250 m at 115200 Baud.

RS485 mode (S7-1500)

The following communication modules support RS485 mode:

- CM PtP RS422/485 BA
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

In RS485 mode, data is transmitted via one line pair (two-wire operation). The line pair is available alternately for the send and receive directions. It is possible to either send or receive (half duplex). On completion of a send operation, operation is immediately switched to receive mode (ready to receive). Send mode is reset again as soon as a new send job is received.

Interface operating modes

The following table is a summary of the interface operating modes for the various communication modules and protocols.

Operating mode	Description
Half duplex (RS485) two-wire operation	Operating mode for point-to-point connection or multipoint connection (multipoint) in two-wire operation. The communication module can be the master as well as the slave.

If you operate the Freeport in RS485 mode (half duplex, two-wire operation), you must make provisions in the user program to ensure that only one device sends data at any given time. If more than one device sends data at the same time, the frames are corrupted.

Modbus automatically ensures that only one device is sending.

Changeover times for RS485 communication module in half duplex mode

A maximum time of 0.1 ms is set for the changeover between sending and receiving.

RS485 signals

The following signals are present on the communication module when using the RS485 hardware:

R (A)/T (A) -	Input/output	Received/transmitted data
R (B)/T (B) +	Input/output	Received/transmitted data
GND		Functional ground; isolated

Connecting cables

The table below shows the pin assignment of the 15-pin D-sub female connector of the respective communication module.

Socket*	Pin	Designation	Input/output
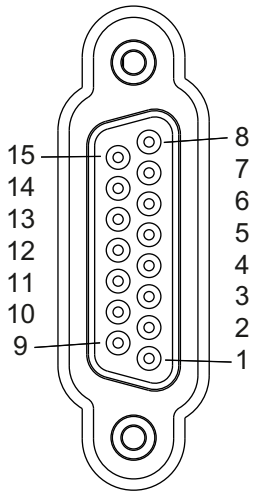	1	-	-
	2	-	-
	3	-	-
	4	R (A)/T (A) -	Input/output
	5	-	-
	6	-	-
	7	-	-
	8	GND	-
	9	-	-
	10	-	-
	11	R (B)/T (B) +	Input/output
	12	-	-
	13	-	-
	14	-	-
	15	-	-

* View from the front

When fabricating the connecting cables, you need to remember that unconnected inputs at the communication partner may have to be connected to open-circuit potential.

Note that you may only use shielded connector enclosures. A large surface area of the cable shield must be in contact with the connector enclosure on both ends.

⚠ CAUTION

Never connect cable shield with GND

Never connect the cable shield with the GND as this could destroy the interfaces. GND must always be connected on both ends (pin 8), otherwise the interface modules could be destroyed.

The figure below illustrates the cable for a point-to-point connection between a communication module and a communication partner.

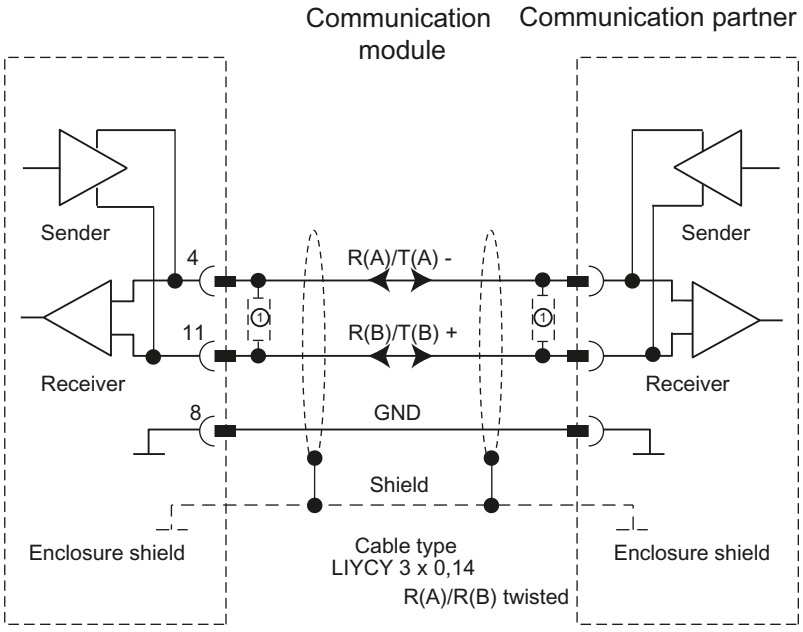

① For cable lengths longer than 50 m, you need to solder a terminating resistor of approx. 330 Ω at the receiver end to ensure interference-free data traffic.

Handshake procedure (S7-1500)

Introduction

Handshaking controls the data flow between two communication partners. The use of the handshaking method prevents data loss during transmission if the devices are operating at different speeds.

We can basically distinguish between the following methods:

Table 1-97 Overview of methods and interfaces

Method	RS232	RS422	RS485
Software data flow control XON/XOFF	X	X	-
Hardware data flow control (RTS/CTS)	X	-	-
Automatic operation of accompanying signals	X	-	-

Software data flow control

Software data flow control is implemented as follows on the communication module:

- XON/XOFF
  - As soon as the communication module has been set to the "XON/XOFF" operating mode by means of parameter assignment, it sends the XON character, thereby allowing the communication partner to send data.
  - On reaching the configured maximum number of frames, or 16 characters ahead of receive buffer overflow, the communication module sends the XOFF character, thereby requesting that the communication partners stop sending. If the communication partner nonetheless continues to send data, an error message is generated if the receive buffer overflows. Data received in the last frame is discarded.
  - As soon as a frame has been fetched by the CPU and the receive buffer is ready to receive data again, the communication module sends the XON character.
  - If the communication module receives the XOFF character during sending, it cancels the current send operation until it receives a XON again from its communication partner. If no XON is received within a specific configurable time, send operation is canceled and a corresponding error message is output.

Note

You can configure the characters for XON and XOFF (any ASCII character).

During parameter assignment of the XON/XOFF software data flow control, user data may not contain any of the configured XON or XOFF characters.

Hardware data flow control

Note

The DTR/DSR signals do not have to be wired for "Hardware RTS always ON, ignore DTR/DSR" parameter assignment.

If "Hardware RTS always ON" is configured, it is imperative that you fully wire the interface signals used. Make sure that the local RTS (out) is connected with the CTS (in) of the communication partner and the local CTS is connected with the RTS of the communication partner. Accordingly, the local DTR must be connected with the DSR of the communication partner and the local DSR with the DTR of the communication partner.

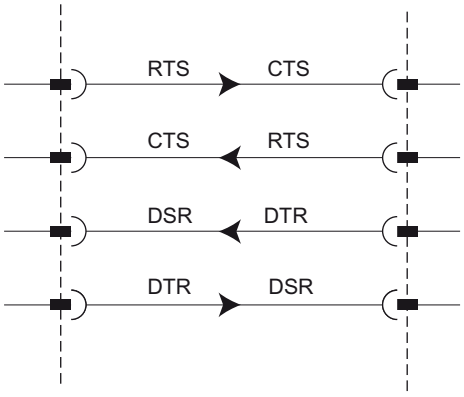

Figure 1-38 Wiring of the interface signals

- Hardware RTS always ON, ignore DTR/DSR
  - As soon as the communication module has been set to an operating mode with "Hardware RTS always ON" through parameter assignment, it outputs the RTS = ON signal to the communication partner to indicate its ready state.
  - RTS is set to OFF as soon as the configured maximum number of frames or 16 characters before buffer overflow is reached.
    If the communication partner nonetheless continues to send data, an error message is generated on overflow of the receive buffer. Data received in the last frame is discarded.
  - RTS is reset to ON as soon as the frame has been fetched by the CPU and the receive buffer is ready to receive data again.
  - If CTS switches to OFF during the send operation, the communication module interrupts the send operation until CTS is reset to ON. If CTS is not reset to ON within a specific configurable time, the send operation is canceled and a corresponding error message is output.
- Hardware RTS always ON
  The "Hardware RTS always ON" mode corresponds to the "Hardware RTS always ON, ignore DTR/DSR" mode. However, you also need to wire DTR and DSR.
  - As soon as the communication module has been set set to an operating mode with "Hardware RTS always ON" through parameter assignment, it sets DTR = ON and RTS = ON to signal its general ready state to the communication partner.
  - RTS is set to OFF as soon as the configured maximum number of frames or 16 characters before buffer overflow is reached.
    If the communication partner nonetheless continues to send data, an error message is generated on overflow of the receive buffer. Data received in the last frame is discarded.
  - RTS is reset to ON as soon as the frame has been fetched by the CPU and the receive buffer is ready to receive data again.
  - If CTS switches to OFF during the send operation, the communication module interrupts the send operation until CTS is reset to ON. If CTS is not reset to ON within a specific configurable time, the send operation is canceled and a corresponding error message is output.
  - A switch from DSR = ON to DSR = OFF cancels an active send job and triggers an error message.

Automatic operation of accompanying signals

- **Hardware RTS always switched**
  "Hardware RTS always switched" is implemented as follows on the communication module:
  - As soon as the communication module is set to the operating mode with "Hardware RTS always switched" through parameter assignment, it sets the line RTS to OFF and DTR to ON (communication module ready for operation).
    It is not possible to send frames until the DSR line is set to ON. No data is sent via the RS232C interface as long as DSR is set to OFF. A send job is canceled and a corresponding error message is generated.
  - When a send job is pending, RTS is set to ON and the configured RTS ON delay starts. On expiration of the data output time, the system checks whether the communication partner has set CTS to ON. If so, the data is sent via the RS232 interface.
  - If the CTS line is not set to ON within the RTS ON delay, or if CTS changes to OFF during transmission, the send job is aborted and an error message is generated.
  - Once the data has been sent and the configured clear RTS OFF delay has elapsed, the RTS line is set to OFF. The system does not wait for CTS to change to OFF.
  - It is always possible to receive data via the RS232 interface. There will be no reaction if there is a danger of the receive buffer of the communication module overflowing.
  - A switch from DSR = ON to DSR = OFF cancels an active send job and triggers an error message.

  Note

  Set the "RTS ON delay" in such a way that the communication partner is able to enter the ready to receive state before the time elapses.

  Set the "RTS OFF delay" in such a way that the communication partner is able to receive the last characters of the frame completely before RTS is set to OFF and the send request is canceled.

  Note

  When automatic operation of the RS232 signals is configured, RTS and DTR cannot be controlled by means of the corresponding instruction!

Time diagram

The figure below shows the chronological sequence of a send job with configured data flow control "Hardware RTS always switched":

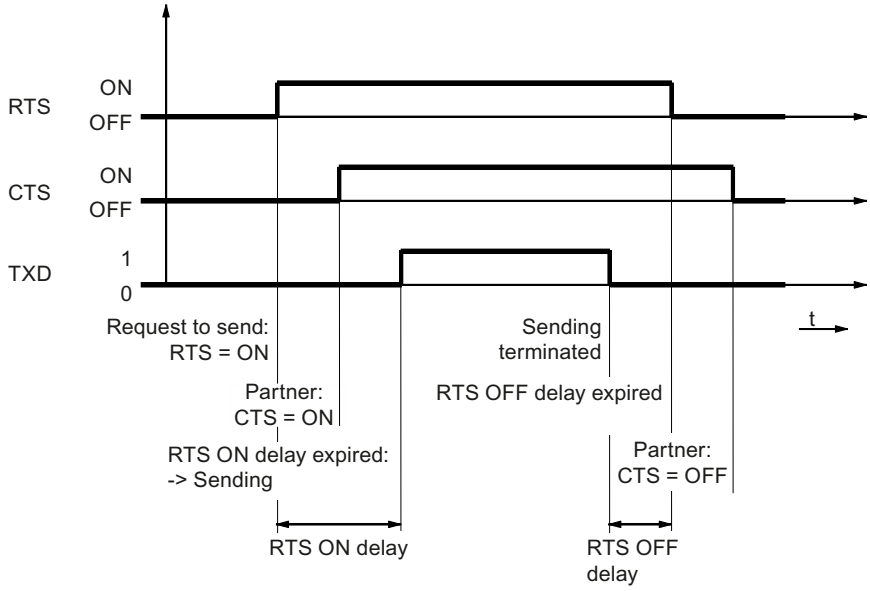

Figure 1-39 Time diagram for Hardware RTS always switched

Additional information

Note

Operation of DTR/DSR or RTS/CTS is accepted by the communication module with the following settings:

- Hardware RTS always ON, ignore DTR/DSR
- Hardware RTS always ON
- Hardware RTS always switched

Configuring / parameter assignment (S7-1500)

Configuring / parameter assignment of a communication module (S7-1500)

The following sections contain explanations on the following protocols and their parameters:

- Communication using Freeport (Page 1548)
- Communication using 3964(R) (Page 1555)
- Communication through Modbus RTU (Page 1560)
- Communication using USS (Page 1566)

This information is required to carry out the parameter assignment and subsequently programming of the communication in accordance with the used protocol.

Configuration and parameter assignment are carried out in the device view of STEP 7 (TIA Portal) and in the properties dialog of the communication module. Some configurations can also be changed during runtime by means of the corresponding "Config" instructions (Port_Config, Send_Config, Receive_Config, P3964_Config).

Procedure for setting up point-to-point communication

The procedure does not depend on the communication module used.

1. Configure an S7-1500 design with CPU and communication module in the device view of the STEP 7 (TIA Portal) hardware editor.
2. You assign the parameters of the communication module interface (protocol, protocol parameters, addresses) in the "General" area of the "Properties" tab.

Communication using Freeport (S7-1500)

Procedure for establishing a serial connection with Freeport (S7-1500)

Requirements

- The hardware is set up and there is an electrical connection to the link partner.
- The project has been created in STEP 7 (TIA Portal) and the CPU has been inserted into the hardware configuration.

Procedure - Hardware configuration

1. Insert the CM PtP communication module into the hardware configuration.
2. Set the communication parameters according to the link partner:
   For example, transmission speed, character frame, frame start and frame end
   These parameters are transferred to the CM PtP communication module every time the CPU is started.

Procedure - Programming

1. Create the data structure that is to include the data to be transferred.

Sending data

1. Insert the instructions from the PtP Communication library: Send_P2P for sending data
2. Interconnect the input and output parameters of the instruction, e.g.:
   - HWID from the system tags at the PORT input
   - Data structure with the data to be sent at the BUFFER input

   Note: During operation, each positive edge at the REQ input will send the specified data area once. The block must be called until DONE indicates that the data was transferred to the module.
   In case of an error, setting ERROR once and displaying the corresponding information in STATUS indicates that the data was not transferred.

Receiving data:

1. Insert the instructions from the PtP Communication library: Receive_P2P for sending data
2. Interconnect the input and output parameters of the instruction, e.g.:
   - HWID from the system tags at the PORT input
   - Data structure for storage of received data at the BUFFER input

   Note: A high level at the NDR output during operation indicates that new data was received and stored in the specified data area. The block must be called until NDR=TRUE. The received data can then be analyzed and the RECEIVE_P2P can be called again.

Optional additions

- Instructions that end in _Config can be used optionally to change the parameters of the hardware configuration during operation of the user program. The changes are not saved in the hardware configuration. They are overwritten at the next restart.
- The instructions Signal_Set and Signal_Get can be used to control the RS232 accompanying signals individually if automatic operation is not a suitable option.

Data transmission with Freeport (S7-1500)

Introduction

Freeport is a freely programmable frame-based protocol that is also known as ASCII protocol.

The Freeport protocol controls data transmission by means of a point-to-point connection between the communication module and a communication partner. The Freeport protocol contains the physical layer (Layer 1).

The Freeport protocol supports sending and receiving of messages with any structure (all characters from 00H through FFH (for character frames with 8 data bits) or from 00H through 7FH (for character frames with 7 data bits)).

The frame start and end criteria must be configured both for the send and the receive direction. The start and end criteria can be configured differently.

Instructions are available for communication with a communication partner (see Overview of PtP programming (Page 1569)).

Sending data with Freeport (S7-1500)

Specifying settings for sending

To send a message, the partner must be informed of the start and end of the message. These settings can be set fixed in the hardware configuration or be adjusted during runtime by using the instruction Send_CFG. You can select one of the following options or also combine them:

- Send break before frame start
  You can specify that an additional Break is sent at the beginning of each message transmission on expiration of the RTS ON delay time.
  The duration of the "Break" is specified in bit times.
  Compliance with the send break can be deactivated if other mechanisms are used for synchronization.
- Send idle line
  You can specify that an additional "Idle Line" signal is output at the start of each message transmission.
  The duration of the "Idle Line" is specified in bit times.
  Compliance with the send break can be deactivated if other mechanisms are used for synchronization.
- RTS ON delay
  You can configure the time that has to expire after the RTS (Request to send) before the actual data transmission starts (RS232 only).
- RTS OFF delay
  You can configure the time that has to expire after transmission has been completed before the RTS signal is deactivated (RS232 only).
- Send up to and including the end delimiter
  You can configure the number of end delimiters (1 or 2) and their value.
  All data up to the end delimiter(s) is sent, independent of the selected frame length. The end delimiter must be included in the data to be sent. Data is sent only up to and including the delimiter, even if the specified data length is longer.
- Number of appended characters
  Input of the number of appended characters. Sending takes place up to the configured length. The end delimiter(s) is/are appended automatically. Depending on the number of end delimiters, one to five characters more than the number specified at the instruction are sent to the partner.

Note

If you combine "Send break before frame start", "Send idle line" and "RTS ON delay", these are processed in the order "RTS ON delay", "Send break before frame start" and "Send idle line".

Receiving data with Freeport (S7-1500)

Specifying the message start

For data transmission with Freeport, you can choose between several different start criteria. The start criterion defines when a frame starts. Once a criterion that indicates the start of the message is met, the data stream is scanned for message end criteria. Here you select the settings that correspond to the properties of the sending communication partner.

Two different methods are available for detecting the message start:

- Start on any character
  Any character can be used to define the start of the message (default).
  This means that the first character sent at the start of communication, or after the frame end has been detected, will be identified as the first character of a message.
- Start on special condition
  The start of the message is detected based on the following specified conditions.
  - After detection of a line break
    The frame start is not accepted unless a break has been received beforehand, in other words, it is compulsory for the partner to send a break before sending a frame.
  - After detection of an idle line
    The frame start is not accepted until the configured idle line duration has expired. This procedure requires a minimum interval between two frames.
  - After receipt of a start character
    The frame start is detected when the configured start character is identified.
  - After detection of one or several start sequences
    The frame start is detected when the configured string with a length of up to five characters is identified. You can configure up to 4 start sequences. The start sequences that are up to 5 characters long can also contain "don't care characters".

Example:

Table 1-98 Configured start conditions

Start condition	1. character	2. character	3. character	4. character	5. character
1	0x68	xx	xx	0x68	xx
2	0x10	0xaa	xx	xx	xx
3	0xdc	0xaa	xx	xx	xx
4	0xe5	xx	xx	xx	xx
:					

The following message has been received: 68 10 aa 68 bb 10 aa 16

The evaluation of the start criteria begins with the receipt of the first character 0x68.
The 2nd and 3rd characters are free.
When the 4th character (second 0x68) is received, the first start condition is met and further evaluation of the message begins.

Specifying the message end

You can choose from several different end criteria for data transmission using the Freeport protocol. The end criterion defines the point at which a frame has been received completely.

Configurable end criteria are:

- Recognize message end by message timeout
- Recognize message end by response timeout
- After character delay time elapses (default)
- After receipt of a fixed frame length
- After receipt of a maximum number of characters
- Read message length from message
- After receipt of an end sequence

Message timeout

When data is received, the end of frame is detected on expiration of the configured time for transferring a frame. Time measurement starts after the start criterion has been met.

Response timeout

The response time is used to monitor the response behavior of the communication partner. If a valid frame start is not recognized after the completion of a send job, the send job is acknowledged with a corresponding message.

The actual end criterion has to be configured additionally.

Expiration of character delay time

When data is received, the frame end is detected when the configured maximum time between successive characters is exceeded (character delay time). The value is specified in bit times.

In this case, the character delay time must be set in such a way as to ensure that it expires between two consecutive frames. However, it should be of sufficient length to exclude incorrect identification of the end of the frame whenever the communication partner performs a transmission pause within a frame.

Fixed frame length

When data is received, the end of the frame is identified after the configured frame length has been reached.

An error message is output and the frame is discarded if the character delay time expires (if activated) before the fixed frame length has been reached.

Please note the following if the frame length of the received characters does not match the fixed configured frame length:

- All characters received after the fixed configured frame length has been reached will be discarded until a new start criterion is detected.
- An error message is output and the frame is discarded if another (activated) end criterion is met before the fixed frame length has been reached.

Maximum number of characters

When receiving data, the end of the frame is recognized after the declared number of characters have arrived.

This setting can be combined with the "Character delay time" settings. The frame received is also assessed as free of error if another end condition occurs, regardless of whether the maximum number of characters has been reached.

Please note the following if the frame length of the received characters does not match the configured maximum frame length:

- All characters received after the configured maximum number of characters has been reached will be discarded until a new start criterion (e.g. "Idle Line") is detected.
- If a different (activated) end criterion is met before the configured maximum number of characters has been reached, this "frame part" is assessed as a valid frame and the partner waits for a new start criterion. All characters received prior to fulfillment of a new start criterion are discarded.

Note

If no further end criterion is activated, the fixed frame length and maximum number of characters will respond in the same way.

Message length in the message

When data is received, the frame end is detected when the frame length sent with the received frame has been reached.

The following parameters define the characters to be used for evaluation of the message length:

- **Offset of length field in message**
  In the message, the value defines the position of the character that is to be used to determine the message length.
  You can set values from 0 to 4095 characters, depending on the buffer size.
- **Size of length field**
  This value specifies the number of characters as of the first evaluation position to be used to determine the message length.
  You can set values of 1, 2 and 4 characters.
- **Number of characters not counted in length specification**
  Number of characters appended to the frame without counting towards the frame length. This value defines the number of bytes at the end of the frame which should not be included in the evaluation of the message length.
  You can set values from 0 to 255 characters.

Example:

Parameter assignments for "Message length in the message"

Offset of length field in message:	3. byte ("2" has to be configured as offset)
Size of length field:	1 byte
Number of characters not counted in length specification:	3 bytes

Message					Number of characters not counted in length specification		
Start character	Address	Length field			Checksum	End delimiter	
Byte 1	Byte 2	Byte 3	Byte ...	Byte X	Byte X+1	Byte X+2	Byte X+3

End sequence

When data is received, the end of the frame is identified when the configured end sequence (max. 5 characters) is received. The end sequence which is up to 5 characters long can also contain "don't care characters". The received data is applied by the CPU, including the end sequence.

If you are working with the end sequence, transmission is not code-transparent and you must exclude the presence of end code in the user data.

Note

Frame end sequence

If there is only one end delimiter, the entry **must** take place in the 5th line.

If there are two end delimiters, the entries **must** take place in the 4th and 5th line (no gaps).

The same applies to the use of additional characters.

Code transparency (S7-1500)

Code transparency

Code-transparent means that any character combinations can occur in the user data without the end criterion being recognized.

The code transparency of the procedure depends on the selection of the configured end criterion and flow control:

- With specified end sequence or using XON/XOFF flow control
  - Not code-transparent
- End criterion character delay time, fixed frame length, maximum frame length, message timeout, or response timeout and message length in the message
  - Code-transparent

Receive buffer (S7-1500)

Receive buffer of the module

The communication modules have a receive buffer that stores the received frames temporarily until they are transmitted to the CPU. The receive buffer is implemented as a ring buffer, which means the frames are transmitted to the CPU in the order in which they were received until the receive buffer is full. If additional frames are received once the buffer is full, the oldest frame is overwritten. If "Prevent overwriting" was configured, a corresponding message is generated when the receive buffer is full. All further frames are rejected until the receive buffer is ready to receive new ones.

During the parameter assignment, you can specify whether the receive buffer should be deleted during startup. You can also specify the range of values (1 to 255) for the number of buffered receive frames.

The receive buffer of the module may have a size of up to 8 KB, depending on the communication module used (see chapter Introduction (Page 1527)). The frame has a maximum length of 4 KB. This means that each communication module is capable of buffering at least two frames.

If you always want to transfer the last frame received to the CPU, you must set the value "1" for the number of buffered frames and deactivate overwrite protection.

Note

If continuous reading of the received data in the user program is interrupted for a certain time, you may find when the receive data is requested again, that the communication module first sends older frames before the CPU receives the most recent one. At the time of interruption, the old frame had already been transmitted from the receive buffer of the communication module and prepared for transmission to the CPU.

Communication using 3964(R) (S7-1500)

Procedure for establishing a serial connection with 3964(R) (S7-1500)

Requirements

- The hardware is set up and there is an electrical connection to the link partner.
- The project has been created in STEP 7 (TIA Portal) and the CPU has been inserted into the hardware configuration.

Procedure - Hardware configuration

1. Insert the CM PtP communication module into the hardware configuration.
2. Set the communication parameters according to the link partner:
   For example, transmission speed, character frame, frame start and frame end
   These parameters are transferred to the CM PtP communication module every time the CPU is started.

Procedure - Programming

1. Create the data structure that is to include the data to be transferred.

Sending data:

1. Insert the instructions from the PtP Communication library: Send_P2P for sending data
2. Interconnect the input and output parameters of the instruction, e.g.:
   – HWID from the system tags at the PORT input
   – Data structure with the data to be sent at the BUFFER input

   Note: During operation, each positive edge at the REQ input will send the specified data area once. The block must be called until DONE indicates that the data was transferred to the module.
   In case of an error, setting ERROR once and displaying the corresponding information in STATUS indicates that the data was not transferred.

Receiving data:

1. Insert the instructions from the PtP Communication library: Receive_P2P for sending data
2. Interconnect the input and output parameters of the instruction, e.g.:
   – HWID from the system tags at the PORT input
   – Data structure for storage of received data at the BUFFER input

   Note: A high level at the NDR output during operation indicates that new data was received and stored in the specified data area. The block must be called until NDR=TRUE. The received data can then be analyzed and the RECEIVE_P2P can be called again.

Optional additions

- Instructions that end in _Config can be used optionally to change the parameters of the hardware configuration during operation of the user program. The changes are not saved in the hardware configuration. They are overwritten at the next restart.
- The instructions Signal_Set and Signal_Get can be used to control the RS232 accompanying signals individually if automatic operation is not a suitable option.

Data transmission with 3964(R) procedure (S7-1500)

Introduction

The 3964(R) procedure controls point-to-point data exchange between the communication module and a communication partner and contains both the physical layer (layer 1) and the link layer (layer 2).

Instructions are available for communication with a communication partner (see Overview of PtP programming (Page 1569)).

Control characters (S7-1500)

Introduction

During data transmission, the 3964(R) procedure adds control characters to the information data (link layer). The communication partner can use these control characters to check whether it has received all data completely and without errors.

Control characters of the 3964(R) procedure

The 3964(R) procedure evaluates the following control characters:

STX	Start of Text	Beginning of the character string to be transmitted	02H
DLE	Data Link Escape	Data transmission changeover	10H
ETX	End of Text	End of the character string to be transmitted	03H
NAK	Negative Acknowledge	Negative acknowledgment	15H
BCC	Block Check Character (only with 3964R)	Block check character	

BCC is formed and monitored automatically in the communication module. The block check character is not transmitted as frame content to the CPU.

Note

If the DLE character is transmitted as an information character within a frame, it is sent twice (DLE duplication) to distinguish it from the DLE control character during connection establishment and termination. The receiver reverses the DLE duplication.

Priority

With the 3964(R) procedure, one communication partner must be assigned a higher and the other a lower priority. If both partners start to establish a connection at the same time, the partner having lower priority will cancel its send job.

Block check character (S7-1500)

Block check character

With the 3964R transfer protocol, data security is enhanced by sending an additional block check character (BCC = Block Check Character).

The block check character is the even longitudinal parity (EXOR logic operation of all data bytes) of a sent or received block. Its calculation begins with the first byte of user data (first byte of the frame) after the connection establishment, and ends after the DLE ETX character at connection termination.

Note

With DLE duplication, the DLE character is included twice in the BCC calculation.

Sending data with 3964(R) (S7-1500)

Connection establishment for sending

The 3964(R) procedure sends the STX control character to set up the connection. If the communication partner responds with the DLE character before the acknowledgment delay time expires, the procedure switches to send mode.

If the communication partner answers with NAK or any other character (except for DLE or STX), or the acknowledgment delay time expires without a response, the procedure tries to set up the connection again. After the configured number of unsuccessful setup attempts, the procedure cancels the connection setup and sends the NAK character to the communication partner. The communication module outputs a corresponding error message.

Sending data

If the connection is successfully established, the user data contained in the output buffer of the communication module is sent to the communication partner with the selected transmission parameters (a DLE recognized in the user data is doubled during the send job). The communication partner monitors the time intervals between the incoming characters. The interval between two characters must not exceed the character delay time. Monitoring of the character delay time starts immediately after the connection has been established.

If the communication partner sends the NAK character during an active send operation, the procedure cancels the block and repeats it as described above, beginning with connection establishment. If a different character is sent, the procedure first waits for the character delay time to expire and then sends the NAK character to set the communication partner to idle state. Then, the procedure restarts sending with the connection setup STX.

Connection termination during sending

Once the contents of the buffer have been sent, the procedure appends the DLE and ETX characters and (only with 3964R) the block checksum BCC as the end identifier, and then waits for an acknowledgment character. If the communication partner sends the DLE characters within the acknowledgment delay time, the data block has been received without errors. If the communication partner responds with NAK, any other character (except DLE), or with a corrupted character, or if the acknowledgment delay time expires without a response, the procedure restarts sending with the connection setup STX.

After the configured number of attempts to send, the procedure stops the process and sends an NAK to the communication partner. The communication module outputs a corresponding error message.

Receiving data with 3964(R) (S7-1500)

Connection setup for receiving

In idle state, when there is no send job to be processed, the procedure waits for the communication partner to set up the connection.

A wait time is started (wait time = acknowledgment delay time - 10 ms, however, maximum of 400 ms) if no free receive buffer is available during the connection setup with STX. An error message is generated if no free receive buffer is available on expiration of this time. The procedure sends the NAK character and returns to the idle state. Otherwise, the procedure sends a DLE and receives the data as described above.

The acknowledgment delay time should be set to the same value at both communication partners.

If the procedure receives any character (except for STX or NAK) while in idle state, it waits for the character delay time (CDT) to expire and then sends the NAK character. The communication module outputs a corresponding error message.

Receiving data

After a successful connection establishment, the incoming receive characters are saved to the receive buffer. If two consecutive DLE characters are received, only one of these is saved to the receive buffer.

After connection has been established and after each receive character, the procedure waits for the next character during the character delay time. If this period expires before another character is received, an NAK is sent to the communication partner. The communication module outputs a corresponding error message. A retry is then expected.

If transmission errors occur during receiving (frame errors, parity errors, etc.), the procedure continues to receive data until the connection is terminated and then sends an NAK to the communication partner. A retry is then expected. If the block still cannot be received without errors after the specified number of transfer attempts, or if the communication partner does not start the retry within a block wait time of 4 seconds, the procedure cancels the receive operation. The communication module reports the first corrupted transfer and the final cancelation.

Connection setup for receiving

If the 3964 procedure detects a DLE ETX string, it terminates the receive operation and confirms a successfully received block by sending a DLE to the communication partner. In the case of a receive error, an NAK is sent to the communication partner. A retry is then expected.

The 3964R procedure terminates the receive operation after having detected the DLE ETX BCC string. It compares the received block check character BCC with the internally calculated longitudinal parity. If the BCC is correct and no other receive errors have occurred, the 3964R procedure sends a DLE and returns to the idle state. The communication module informs the control system that new receive data is available.

If the BCC is faulty or a different receive error occurs, an NAK is sent to the communication partner. A retry is then expected.

Communication through Modbus RTU (S7-1500)

Procedure for establishing a serial connection with Modbus RTU (S7-1500)

Requirements

- The hardware is set up and there is an electrical connection to the link partner.
- The project has been created in STEP 7 (TIA Portal) and the CPU has been inserted into the hardware configuration.

Procedure - Hardware configuration

1. Insert the CM PtP communication module into the hardware configuration.
2. Select the Modbus RTU protocol.
   Note: With Modbus RTU, most communication parameters are set using the Modbus_Comm_Load instruction during CPU start.

Procedure - Programming

1. Create the data structure that is to include the data to be transferred.
2. Integrate the Modbus_Comm_Load instruction into the cyclic sequence for parameter assignment of the communication module.
3. Interconnect the HWID from the system tags at the PORT input.
4. Call the instruction until successful execution is displayed at the DONE output. Do not call the instruction again thereafter unless you want to change the communication parameters.

Operation as Modbus master:

1. Insert the Modbus_Master instruction from the MODBUS (RTU) library:
2. Interconnect the data structure with the data to be sent at the BUFFER input.
3. Interconnect the instance DB of the Modbus_Master instruction at the MB_DB input of the Modbus_Comm_Load.
   Note: During operation, each positive edge at the REQ input will process the specified job once. The block must be called until DONE indicates that the data was transferred to the module.
   In case of an error, setting ERROR once and displaying the corresponding information in STATUS indicates that the data was not transferred.

Operation as Modbus slave:

1. Insert the Modbus_Slave instruction from the MODBUS (RTU) library.
2. Interconnect the data structure with the Modbus hold registers.
3. Enter the Modbus slave address at the MB_ADDR input.
4. Interconnect the instance DB of the Modbus_Master instruction at the MB_DB input of the Modbus_Comm_Load.
   Note: A high level at the NDR output during operation indicates that new data was received and stored in the specified data area.

Overview of modbus communication (S7-1500)

Modbus RTU communication

Modbus RTU (Remote Terminal Unit) is a standard protocol for communication in the network and uses the electrical RS232 or RS422/485 connection for serial data transmission between Modbus devices in the network.

Modbus RTU uses a master/slave network in which the entire communication is triggered by only one master device while the slaves can only respond to the request of the master. The master sends a request to a slave address and only this slave address responds to the command (exception: broadcast frames to slave address 0 which are not acknowledged by the slaves).

The procedure used is a code-transparent, asynchronous half-duplex procedure. Data transmission is carried out without handshake.

Position in the system environment

The following Modbus description refers to the use of the corresponding communication modules.

- CM PtP RS232 HF
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

Function of the coupling

With the corresponding communication modules and the related instructions, you can establish a communication connection between a remote Modbus control system and a SIMATIC S7.

The GOULD-MODBUS protocol in RTU format is used for transmission.

Function codes 01, 02, 03, 04, 05, 06, 08, 15 and 16 are used for communication between a communication module operated as a Modbus slave and a master system (see Function Codes (Page 1565)).

If a SIMATIC S7 communication module is operated as a Modbus master, function codes 11 and 12 are also available.

SIMATIC S7 as a Modbus slave

The master has the initiative for transmission, the communication module works as a slave.

frame traffic from slave to slave is not possible.

The instruction Modbus_Slave makes the data available on a SIMATIC data area in accordance with the mapping specification or stores them.

SIMATIC S7 as a Modbus master

As master, the communication module initiates transmission and, after outputting a request frame, it waits for the configured response monitoring time for a response frame from the slave. If the slave does not respond, the master repeats the request in accordance with the configuration before it outputs an error message.

frame structure

The data exchange "Master-Slave" and/or "Slave-Master" begins with the **slave address**, followed by the **function code**. Then the data are transferred. The structure of the data field depends on the function code used. The CRC check is transmitted at the end of the frame.

ADDRESS	FUNCTION	DATA	CRC-CHECK
Byte/Word	Byte	n byte	2 byte

ADDRESS	Modbus slave address • Standard address: 1 to 247 (bytes) • Extended station address: 1 to 65535 (word)
FUNCTION	Modbus Function Codes (Page 1565)
DATA	frame data: Management and net data depending on the function code
CRC-CHECK	frame checksum

Slave address

The slave address can be range from 1 to 247 (byte) or 1 to 65535 (word). The address is used to address a defined slave on the bus.

Broadcast Message

The master uses slave address 0 to address all slaves on the bus.

Broadcast messages are only permitted in conjunction with writing Function codes 05, 06, 15 and 16.

A broadcast message is not followed by a response frame from the slave.

Data Field DATA

The data field DATA is used to transfer the function code-specific data such as:

- Bytecount, Coil_Startaddress, Register_Startaddress; Number_of_Coils, Number_of_Registers, ...
  For details, see "Function Codes (Page 1565)".

CRC-Check

The end of the frame is identified by means of the CRC 16 checksum consisting of 2 bytes. It is calculated by the following polynominal: $x^{16} + x^{15} + x^2 + 1$.

The low byte is transmitted first, followed by the high byte.

End of frame

The end of frame is recognized when no transmission takes place during the time period required for the transmission of three and a half characters (3.5 times character delay time) (see Modbus Protocol Reference Guide).

This end of frame TIME_OUT therefore depends on the data transmission rate and is indicated in bit times (35 bit times are fix coded internally; further bit times can be configured in addition at the instruction).

The Modbus frame received from the connection partner is evaluated and formally checked after the end of frame TIME_OUT is received.

Exception responses

On recognition of an error in the request frame from the master, for example, register address illegal, the slave sets the highest value bit in the function code of the response frame.

This step is followed by transmission of a byte exception code that describes the cause of the error.

A detailed description of the meaning of the above-mentioned parameters is available in the "GOULD MODICON Modbus Protocol" (not part of this documentation).

Exception code frame

The exception code frame from the slave has the following structure:

- for example, slave address 5, function code 5, exception code 2

Response frame from the slave EXCEPTION_CODE_xx:

05H	Slave address
85H	Function code
02H	Exception code (1...7)
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

On receipt of an exception code frame by the driver, the current job is completed with error.

The following error codes are defined in accordance with the Modbus specification:

Error code	Meaning in accordance with Modbus specification	Cause - Short Description *
1	Illegal function	Illegal function code
2	Illegal data address	Slave has illegal data address
3	Illegal data value	Slave has illegal data value
4	Failure in Associated Device	Slave has internal error
5	Acknowledge	Function is carried out
6	Busy, Rejected message	Slave is not ready to receive
7	Negative acknowledgement	The function cannot be carried out.
* Check slave for further details.		

RS232 mode

The following communication modules support RS232 mode:

- CM PtP RS232 HF
- CM PtP (ET 200SP)

For more information on RS232 mode, see the chapter RS232 mode (Page 1534).

For information on hardware data flow control and on automatic operation of the accompanying signals, refer to the Handshake procedure (Page 1543) chapter.

RS422/485 mode

The following communication modules support RS422/485 mode:

- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

For more information on RS422/485 mode, see the chapters RS422 mode (Page 1537) and RS485 mode (Page 1540).

FAQ

For more information, see the following FAQs in the Siemens Industry Online Support:

- Entry ID 68202723 (https://support.industry.siemens.com/cs/ww/en/view/68202723)
- Entry ID 58386780 (https://support.industry.siemens.com/cs/ww/en/view/58386780)

Function Codes (S7-1500)

Used function codes

The function code defines the meaning of the frame. It also defines the structure of a frame.

The following function codes are supported by the communication module:

Function code	Function in accordance with MODBUS specification	Range
01	Read Coil Status	1 to 2000 bit/request
02	Read Input Status	1 to 2000 bit/request
03	Read Holding Registers	1 to 124/125 bit/request (124 at extended station address)
04	Read Input Registers	1 to 124/125 bit/request (124 at extended station address)
05	Force Single Coil	1 bit/request
06	Preset Single Register	1 word/request
08 *	Loop Back Test	Read slave status or reset event counter in the slave
11 *	Fetch Communications Event Counter (only master)	-
15	Force Multiple Coils	1 to 1968 bits/request
16	Preset Multiple Registers	1 to 123 bit/request

* Diagnostic information for slave communication

Modbus function code 00 sends a broadcast message to all slaves (without slave response).

Assignment of the Modbus addresses to the SIMATIC addresses

The table below shows the assignment of the Modbus addresses to the SIMATIC addresses.

Modbus				S7-1500	
FC [1]	Function	Declaration	Address area	Declaration	CPU address
01	Read bits	Output	1 - 9999	Process image of outputs	Q0.0 - Q1249.6
02	Read bits	Input	10001 - 19999	Process image of inputs	I0.0 - I1249.6
03 [2]	Read words	Holding Register	40001 - 49999 or 400001 - 465535	DW0 - DW19998 or DW0 - DW131068	The M address area depends on the CPU
04	Read words	Input	30001 - 39999	Process image of inputs	IW0 to IW19996
05 [2]	Write bits	Output	1 - 9999	Process image of outputs	Q0.0 to Q1248.7
06	Write words	Holding Register	40001 - 49999 or 400001 - 465535	DW0 - DW19998 or DW0 - DW131068	The M address area depends on the CPU

Modbus				S7-1500	
FC [1]	Function	Declaration	Address area	Declaration	CPU address
15	Write bits	Output	1 - 9999	Process image of outputs	Q0.0 - Q1249.6
16 [2]	Write words	Holding Register	40001 - 49999 or 400001 - 465535	DW0 - DW19998 or DW0 - DW131068	The M address area depends on the CPU

1) FC = function code

2) The value of the HR_Start_Offset determines whether data areas or bit memory address areas can be addressed with the FCs 03, 05 and 16 in the SIMATIC CPU.

Communication using USS (S7-1500)

Procedure for establishing a serial connection with Modbus USS (S7-1500)

Requirements

- The hardware is set up and there is an electrical connection to the link partner.
- The project has been created in STEP 7 (TIA Portal) and the CPU has been inserted into the hardware configuration.

Procedure - Hardware configuration

1. Insert the CM PtP communication module into the hardware configuration.
2. Select the Freeport protocol and set the communication parameters.
   Note: The USS functionality is implemented by the instructions.

Procedure - Programming

1. Insert the USS_Port_Scan instruction from the USS Communication library.
2. Interconnect the HWID from the system tags at the PORT input.
3. Insert the USS_Drive_Control instruction from the USS Communication library.
4. Interconnect the USS_DB data structure in the instance DB of the USS_Drive_Control instruction to the USS_DB input of the USS_Port_Scan instruction. The data structure contains the data to be transferred for all drives.
5. Insert an additional call of the USS_Drive_Control instruction for each additional axis that is to be connected via the USS interface.
   Use the same instance DB each time. The distinction takes place with the help of the USS address that you specify at the DRIVE input of the USS_Drive_Control instruction. This means you have access to the control and feedback data at the parameters of the respective call for each drive.

Overview of USS communication (S7-1500)

Position in the system environment

The following USS description refers to the use of the corresponding communication modules.

- CM PtP RS232 BA
- CM PtP RS422/485 BA
- CM PtP RS232 HF
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

Introduction

The USS® protocol (Universal Serial Interface Protocol) is a basic serial data transmission protocol designed to meet the requirements of drive technology.

The USS protocol defines an access method based on the master-slave principle for communication via a serial bus. One master and up to 16 drives (slaves) can be connected to the bus. The individual drives are selected by the master using an address character in the frame. A drive can never send anything without first being initiated by the master. Therefore, direct data transmission between individual drives is not possible. Communication functions in half-duplex mode. The master function cannot be transferred.

Drive technology requires specific response times for the control tasks and therefore strict cyclical frame traffic:

The master continuously sends frames (job frames) to the drives and expects a response frame from each addressed drive.

A drive must send a response frame if

- it has a received a frame without errors and
- it was addressed in this frame.

A drive may not send if these conditions are not met or the drive was addressed in the broadcast.

The connection with the respective drives exists for the master once it receives a response frame from the drive after a specified processing time (response delay time).

frame structure

Each frame begins with a start character (STX), followed by the length specification (LGE) and the address byte (ADR). The data field comes after that. The frame ends with the block check character (BCC). The frame length includes the user data (quantity n), the address byte (ADR) and the data verification character (BCC).

STX	LGE	ADR	1	2	...	N	BCC

For single-word (16-bit) data, the high byte is sent first followed by the low byte. Correspondingly, with double-word data the high word is sent first, followed by the low word. The length of a frame is specified in bytes.

Data encryption

The data is encrypted as follows:

- STX: 1 byte, start of text, 02H
- LGE: 1 byte, contains the frame length as a binary number
- ADR: 1 byte, contains the slave address and frame type in binary code
- Data fields: One byte each, content depending on job
- BCC: 1 byte, block check character

Data transmission procedure

The master ensures cyclic data transmission in frames. The master addresses all slave devices one after another with a job frame. The nodes addressed respond with a response frame. In accordance with the master-slave procedure, the slave must send the response frame to the master after it has received the job frame. Only then can the master address the next slave.

Data field in the frame

The data field is divided into two areas: the parameter area (PKW) and the process data area (PZD).

STX	LGE	ADR	Parameter (PKW)	Process data (PZD)	BCC

- **Parameter area** (PKW)
  The PKW area handles parameter transmission between two communication partners (e.g., controller and drive). This involves, for example, reading and writing parameter values and reading parameter descriptions and the associated text. The PKW interface generally contains jobs for operation and display, maintenance and diagnostics.
- **Process data area** (PZD)
  The PZD area consists of signals that are required for automation:
  - Control words and setpoints from the master to the slave
  - Status words and actual values from the slave to the master

  The contents of the parameter area and process data area are defined by the slave drives. For additional information on this, refer to the drive documentation.

Overview of functions (S7-1500)

Transmission sequence

The instructions process the data transmission cyclically with up to 16 drive slaves. Only one job is active for each drive at any one time.

Performance features:

- Creation of data storage areas for communication, depending on the bus configuration
- Execution and monitoring of PKW jobs
- Monitoring of the complete system and troubleshooting
- Communication with the CPU
- Access to the drive functions
- Reading the drive parameters
- Writing the drive parameters

Programming - communication using instructions (S7-1500)

Overview of point-to-point programming (S7-1500)

Data exchange using Freeport or 3964(R) communication

You must make the send data available in data blocks or in the bit memory address area in the user program of the corresponding CPU. A receive buffer is available in the communication module for the receive data. A corresponding data block is set up in the data block.

In the user program of the CPU, the following instructions carry out the data transfer between the CPU and the communication module.

- Send_P2P
- Receive_P2P

The receive buffer can be deleted with the instruction Receive_Reset.

Dynamic configuration by means of the user program

As an alternative to or in addition to the parameter assignment of the communication module interface described in section Configuring / parameter assignment of a communication module (Page 1547), it may be advisable in certain application areas to set up the communication dynamically, i.e., program-controlled by a specific application.

All parameters assigned in the properties dialog of the communication module can also be changed during runtime by means of one of the following "Config" instructions:

Port_Config, Send_Config, Receive_Config, P3964_Config

Program calls for point-to-point communication - sequence

The figure below shows the function of the point-to-point instructions for communication between the user program and communication partner.

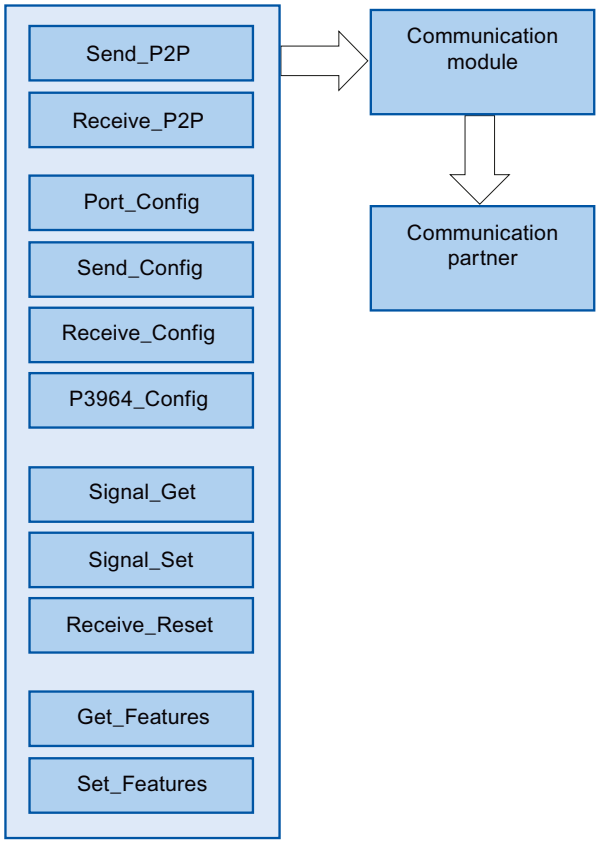

PtP instructions

Application	Instruction	Description
Data exchange between CPU, communication module and communication partner (communication)	Send_P2P	The instruction Send_P2P (send point-to-point data) can be used to send data to the communication partner. Call up the instruction Send_P2P to send data using the Freeport protocol. You have to call the instruction cyclically until you receive a corresponding acknowledgement at the output parameters of the instruction. Note: During parameter assignment of the XON/XOFF data flow control, user data may not contain any of the configured XON or XOFF characters. Default settings are DC1 = 11H for XON and DC3 = 13H for XOFF.
	Receive_P2P	The instruction Receive_P2P (receive point-to-point data) can be used to pick up the messages received in the communication module from a communication partner. Call the Receive_P2P instruction cyclically to receive data using the Freeport protocol. The instruction indicates at the NDR parameter if new received data is available. To signal the start and end of a message transmission, you need to define criteria in the Freeport protocol which identify the start and end of the message.
Deletion of the receive buffer	Receive_Reset	The instruction Receive_Reset (delete receive buffer) allows you to clear the receive buffer of the communication module.
Dynamic parameter assignment of the interface or the protocol (optional)	Port_Config	You can use the Port_Config instruction (port configuration) to configure basic interface parameters, such as the data transmission rate, parity and data flow control dynamically through your user program.
	Send_Config	With the instruction Send_Config(send configuration) you can configure serial send parameters, such as RTS ON delay / RTS OFF delay, dynamically for a point-to-point communication interface.
	Receive_Config	The instruction Receive_Config (receive parameter assignment) allows you to dynamically assign serial receive parameters to a communication module. This instruction configures the conditions that specify the start and the end of a received message.
	P3964_Config	The instruction P3964_Config(configure protocol) can be used to dynamically configure protocol parameters of the procedure 3964(R), such as character delay time, priority and block check using your program.
Operation of RS232 accompanying signals	Signal_Get	With the Signal_Get instruction (get RS232 signals) you can read the current states of the RS232 signals.
	Signal_Set	With the Signal_Set instruction (get RS232 signals), you can set the states of the RS232 signals DTR and RTS.
Enable Modbus CRC support and diagnostic interrupt	Get_Features	You can use the instruction Get_Features(get extended functions) to get information on the Modbus support and on generating diagnostic alarms.
	Set_Features	If supported by the module, you can use the instruction Set_Features(set extended functions) to activate the generation of diagnostic alarms.

Procedure for setting up Freeport or 3964(R) communication

Requirement: The configuration and parameter assignment of a CPU and a communication module in the device view and in the properties dialog of the communication module are complete.

1. In the project navigation for the CPU select the folder "Program blocks" and open the Main (OB1) in the folder by double-clicking it. The program editor opens.
2. From the "Instructions" task card, "Communication" area select the instructions Send_P2P and Receive_P2P and drag-and-drop them into a network of the Main (OB1).
3. Configure the instructions in accordance with your specifications.
4. Download the hardware configuration and the user program to the CPU.

Overview of Modbus programming (S7-1500)

Program calls for Modbus communication - sequence

The figure below shows the function of the Modbus instructions for communication between user program and Modbus device. (The instructions Send_P2P, Receive_P2P and the Config instruction are required downstream).

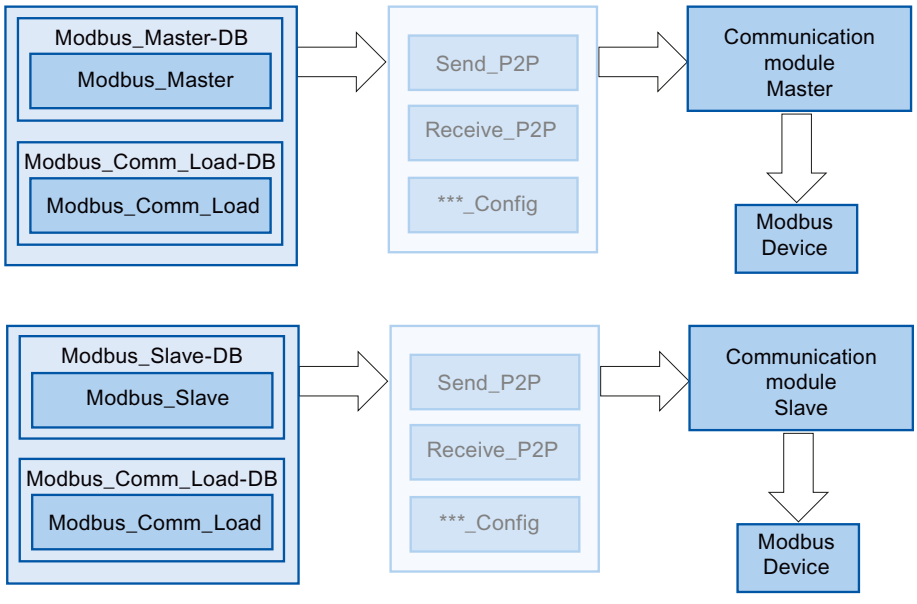

Modbus instructions

Application	Instruction	Description
Data exchange between user program and Modbus device (communication)	Modbus_Master	The Modbus_Master instruction allows you to communicate as Modbus master by means of the PtP port. The CPU can be used as Modbus RTU master device with the Modbus_Master instruction for communication with one or several Modbus slave devices.
	Modbus_Slave	The Modbus_Slave instruction allows you to communicate as Modbus slave by means of the PtP port. The CPU can be used as Modbus RTU slave device with the Modbus_Slave instruction for communication with one Modbus master device.
Parameter assignment of the interface and the protocol (mandatory)	Modbus_Comm_Load	The instruction Modbus_Comm_Load allows you to configure the port of the communication module for Modbus RTU. You have to run Modbus_Comm_Load to set up PtP port parameters, such as data transmission rate, parity and flow control. Once you have configured the interface for the Modbus RTU protocol, it can only be used by the instruction Modbus_Master or the instruction Modbus_Slave .

Note

Alternative use of Modbus_Slave and Modbus_Master

A communication module can be operated either as master or as slave.

Procedure for setting up Modbus communication

Requirement: The configuration and parameter assignment of a CPU and a communication module in the device view and in the properties dialog of the communication module are complete.

1. In the project navigation for the CPU select the folder "Program blocks" and open the Main (OB1) in the folder by double-clicking it. The program editor opens.
2. From the "Instructions" task card, "Communication" area select the instructions for Modbus communication in accordance with your task and drag-and-drop them into a network of the Main (OB1):
   - The instruction Modbus_Comm_Load configures the port of the communication module for Modbus communication.
     The Modbus_Comm_Load must be called in Main (OB1) until DONE (or ERROR) is reported.
   - The instruction Modbus_Master is used for the Modbus master functionality.
   - The instruction Modbus_Slave is used for the Modbus slave functionality.
3. Configure the instructions in accordance with your specifications.
4. Download the hardware configuration and the user program to the CPU.

See also

Overview of instructions (Page 1530)

Overview of USS programming (S7-1500)

Program calls for USS communication - sequence

The figure below shows the function of the USS instructions for communication between user program and USS drive. (The instructions Send_P2P, Receive_P2P and the Config instruction are required downstream).

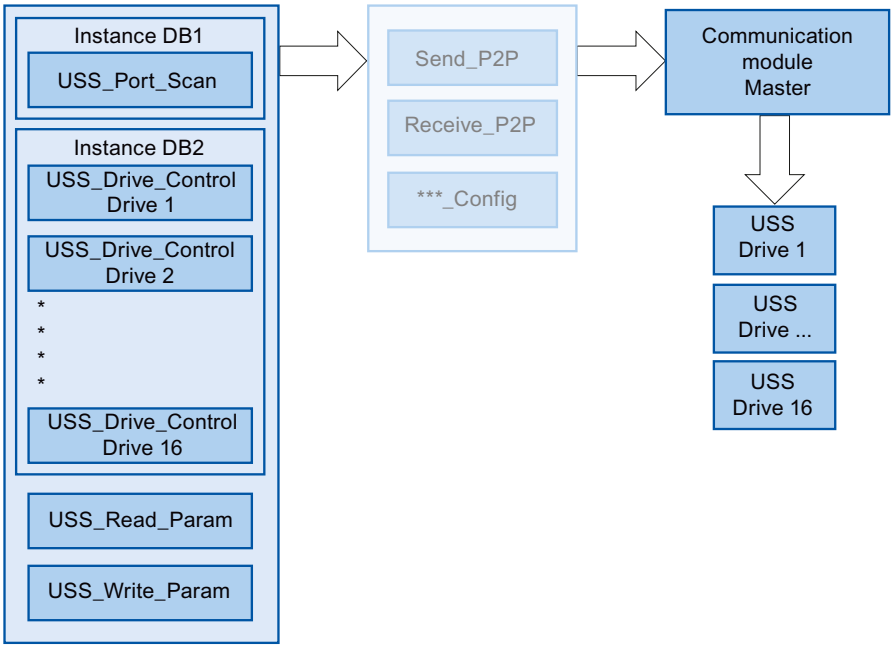

USS instructions

Application	Instruction	Description
Data communication between CPU, communication module and USS drive	USS_Port_Scan	The USS_Port_Scan instruction allows you to communicate via a communication module with up to 16 drives using a USS network (must be called cyclically). The instruction USS_Port_Scan controls the communication between CPU and the drives by means of the PtP communication port. A communication with the drive is processed every time you call this function. The instruction USS_Port_Scan is required once: Since most drives features a configurable internal function that monitors the integrity of the communication based on a timeout, the instruction USS_Port_Scan should be called from a time-controlled OB.
Exchange data with USS drive	USS_Drive_Control	The USS_Drive_Control instruction allows you to prepare the send data for a drive and to display the received data. The inputs and outputs of the instruction correspond to the states and operating functions of the drive. The USS_Drive_Control instruction must be called once for each drive. Only one common instance DB is required for all calls of the instruction USS_Drive_Control for a USS network. Interconnect all calls of the instructions USS_Drive_Control for a USS network with the same instance DB. The USS_Drive_Control instruction should be called from the cyclic Main (OB1) of the main program.
Read or modify parameters in USS drive	USS_Read_Param	The USS_Read_Param instruction allows you to read parameters from the drive. You use the USS_Read_Param instruction to read the operating parameters of the drive that controls the internal drive functions. The USS_Read_Param instruction should be called from the cyclic Main (OB1) of the main program.
	USS_Write_Param	The USS_Write_Param instruction allows you to change parameters in the drive. The USS_Write_Param instruction should be called from the cyclic Main (OB1) of the main program.

Procedure for setting up USS communication

Requirement: The configuration and parameter assignment of a CPU and a communication module in the device view and in the properties dialog of the communication module are complete.

1. In the project tree for the CPU, select the "Program blocks" folder and open the desired time-controlled OB by double-clicking it. The program editor opens.
2. From the "Instructions" task card, "Communication" area select the instruction USS_Port_Scan and drag-and-drop it into a network of a time-controlled OB.
   The instruction USS_Port_Scan allows you to communicate by means of the USS network.
3. In the project navigation for the CPU select the folder "Program blocks" and open the Main (OB1) in the folder by double-clicking it. The program editor opens.

4. From the "Instructions" task card, "Communication" area select the instructions for USS communication in accordance with your task and drag-and-drop them into a network of the Main (OB1):
   - The instruction USS_Drive_Control is used for data exchange with the drive.
   - The instruction USS_Read_Param is used for reading parameters from the drive.
   - The instruction USS_Write_Param is used for changing parameters in the drive.
5. Configure the instructions in accordance with your specifications.
6. Download the hardware configuration and the user program to the CPU.

See also

Overview of instructions (Page 1530)

Startup and Diagnostics (S7-1500)

Startup characteristics (S7-1500)

Operating mode transitions

After the communication module starts up, all data between the CPU and the communication module is exchanged by means of instructions.

CPU STOP	During a running data transmission communication module - CPU, both a send and a receive job is aborted.
CPU RUN	Send and receive operation is ensured in the RUN state of the CPU. With a corresponding configuration in the properties dialog of the communication module, you can automatically clear the receive buffer on the communication module during CPU startup.

From the view of the communication module, there are no further operating states/operating state transitions.

Diagnostic functions (S7-1500)

Introduction

The diagnostic functions of the communication module allow errors that have occurred to be located quickly. The following diagnostic options are available to you:

Diagnostics by means of the display elements of the communication module	The indicators provide information on the operating mode or the possible error states of the communication module. The indicators provide an initial overview of any internal or external errors and interface-specific errors. For more information refer to the device manual of the corresponding communication module.
Diagnostics via the STATUS output of the instructions	Instructions have a STATUS output for error diagnostics; it provides information about communication errors between the communication module and the CPU. You can evaluate the STATUS parameter in the user program (it is active for exactly one cycle).
Diagnostic error interrupt	The communication module can trigger a diagnostic error interrupt on the CPU assigned to it. The communication module makes diagnostic information available. The analysis of this information is made via the user program or by reading the CPU diagnostics buffer.

Diagnostic interrupts (S7-1500)

The diagnostics are displayed as plain text in STEP 7 (TIA Portal) in the online and diagnostics view. You can evaluate the error codes with the user program.

The following diagnostics can be signaled:

- Error (9_H)
- Parameter assignment error (10_H)
- Wire break (109_H)

1.4.6 Configurations for point-to-point connections (S7-300, S7-400)

1.4.6.1 Configuration for point-to-point links (S7-300, S7-400)

Introduction (S7-300, S7-400)

Overview of the components and interfaces (S7-300, S7-400)

Contents

Table overview of the module variations and their functions.

For CP 340/341/440/441 and ET 200S 1SI

Module	Interface	Protocols						
		ASCII	3964(R)	RK512	Printer driver	Modbus master	Modbus slave	USS master
CP 340	RS232 C	X	X	-	X	-	-	-
CP 340	20 mA TTY	X	X	-	X	-	-	-
CP 340 1)	RS422	X	X	-	X	-	-	-
	RS485	X	-	-	-	-	-	-
CP 341	RS232 C	X	X	X	X	X 2)	X 2)	-
CP 341	20 mA TTY	X	X	X	X	X 2)	X 2)	-
CP 341 1)	RS422	X	X	X	X	X 2)	X 2)	-
	RS485	X	-	-	X	X 2)	X 2)	-
CP 440	RS422	X	X	-	-	-	-	-
	RS485	X	-	-	-	-	-	-
CP 441-1	RS232 C	X	X	-	X	-	-	-
CP 441-1	20 mA TTY	X	X	-	X	-	-	-
CP 441-1 1)	RS422	X	X	-	X	-	-	-
	RS485	X	-	-	X	-	-	-
CP 441-2	RS232 C	X	X	X	X	X 2)	X 2)	-
CP 441-2	20 mA TTY	X	X	X	X	X 2)	X 2)	-
CP 441-2 1)	RS422	X	X	X	X	X 2)	X 2)	-
	RS485	X	-	-	X	-	-	-
ET 200S 1SI for ASCII and 3964(R)	RS232 C	X	X	-	-	-	-	-
	RS422	X	X	-	-	-	-	-
	RS485	X	-	-	-	-	-	-
ET 200S 1SI for modbus and USS	RS232 C	-	-	-	-	X	X	X
	RS422	-	-	-	-	X	X	-
	RS485	-	-	-	-	X	X	X
CPU 313C-2 PtP	RS422	X	X	-	-	-	-	-
	RS485	X	-	-	-	-	-	-
CPU 314C-2 PtP	RS422	X	X	X	-	-	-	-
	RS485	X	-	-	-	-	-	-

1) The difference between RS422 and RS485 is set in the configuration dialog.

2) A loadable driver with a dongle is needed for the Modbus protocol.

Accompanying signals and data flow control

The RS232C accompanying signals can be controlled with the ASCII, Modbus master and Modbus slave protocols via the RS232C interface.

Data flow control with RTS/CTS is possible with the ASCII and printer driver protocols via the RS 232C interface.

Data flow control with XON/XOFF is possible with the ASCII and printer driver protocols via the RS232C, 20mA TTY and RS422 interfaces.

Overview of the instructions (S7-300, S7-400)

Overview of the instructions

The communication between the CPU, communication module and a communication partner is performed via the instructions and the protocols of the communication module.

The instructions form the software interface between the CPU and the communication module. They must be called cyclically from the user program.

The communication protocols are implemented on the communication module. The protocol is used to adapt the interface of the communication module to the interface of the communication partner.

Table 1-99 Instructions for CP 340

Instruction	Meaning
P_SEND	The instruction P_SEND allows you to send an entire range or a partial range of a data component to a communication partner.
P_RCV	The instruction P_RCV allows you to receive data from a communication partner and to store them in a data block.
P_PRINT	The P_PRINT instruction allows you to print a message text with up to 4 variables on a printer.
P_RESET	The P_RESET instruction allows you to reset the internal receive buffer of the CP 340.
V24_STAT_340	The V24_STAT instruction allows you to read the signal states at the RS232C interface of the CP 340-RS232C.
V24_SET_340	The V24_SET instruction allows you to set/reset the outputs on the RS232C interface of the CP 340 RS232C.

Table 1-100 Instructions for CP 341

Instruction	Meaning
P_SND_RK	You can use the instruction P_SND_RK to send the entire range or a partial range of a data block to a communication partner, or you can fetch data from the communication partner (using RK512).
P_RCV_RK	You can use the P_RCV_RK instruction to receive data from a communication partner and store it in a data block or make data available to the communication partner (using RK512).
P_PRINT341	The instruction P_PRINT341 allows you to print a message text with up to 4 variables on a printer.
V24_STAT	The V24_STAT instruction allows you to read the signal states at the RS232C interface of the CP 341-RS232C.
V24_SET	The V24_SET instruction allows you to set/reset the outputs on the RS232C interface of the CP 341 RS232C.

Table 1-101 Instructions for CPU 313C-2 PTP

Instruction	Meaning
SEND_PTP	With the instruction SEND_PTP, you can send the entire range or a partial range of a data block to a communication partner.
RCV_PTP	With the instruction RCV_PTP, you can receive the data from a communication partner and store it in a data block.
RES_RCVB	You can use the RES_RCVB instruction to reset the receive buffer of the module.

Table 1-102 Instructions for CPU 314C-2 PTP

Instruction	Meaning
SEND_PTP	With the instruction SEND_PTP, you can send the entire range or a partial range of a data block to a communication partner.
RCV_PTP	With the instruction RCV_PTP, you can receive the data from a communication partner and store it in a data block.
RES_RCVB	You can use the RES_RCVB instruction to reset the receive buffer of the module.
SEND_RK	You can use the SEND_RK instruction to send the entire range or a partial range of a data block to a communication partner using the RK512 protocol.
FETCH_RK	You can use the FETCH_RK instruction to request data from the communication partner using the RK512 protocol.
SERVE_RK	You can use the SERVE_RK instruction to receive RK512 protocol data from a communication partner and store it in a data block or make data available to the communication partner.

Table 1-103 Instructions for CP 440

Instruction	Meaning
SEND_440	With the instruction SEND_440 , you can send the entire range or a partial range of a data block to a communication partner.
RECV_440	With the instruction RECV_440 , you can receive the data from a communication partner and store it in a data block.
RES_RECV	With the instruction RES_RECV , you can reset the receive buffer of the CP 440.

Table 1-104 Instructions for CP 441

Instruction	Meaning
V24_STAT_441	The V24_STAT_441 instruction allows you to read the signal states on the RS232C interface of the CP 441.
V24_SET_441	The V24_SET_441 instruction allows you to set/reset the outputs on the RS232C interface of the CP 441 .
The automation system S7-400 provides a row of instructions that activate and control the communication in the user program between CPU and the communication processor CP 441. These instructions are stored in the memory of the CPU (see below in the section "Instructions from the automation system S7-400").	

Table 1-105 Instructions for ET 200S 1SI

Instruction	Meaning
S_SEND	The instruction S_SEND allows you to send an entire range or a partial range of a data component to a communication partner.
S_RCV	The instruction S_RCV allows you to receive data from a communication partner and to store them in a data block.
S_VSTAT	The instruction S_VSTAT allows you to read the signal states on the RS 232C-interface of the component ET 200S 1SI.
S_VSET	The instruction S_VSET allows you to set/reset the outputs on the RS 232C-interface of the module ET 200S 1SI.
S_XON	With the instruction S_XON , you can set additional settings if the module was set for the flow control XON/XOFF.
S_RTS	With the instruction S_RTS , you can set additional settings if the module was set for the flow control RTS/CTS.
S_V24	With the instruction S_V24 , you can set additional settings if the module was set for the automatic operation of the V.24 signals.
S_MODB	The S_MODB instruction allows you to establish a communication connection to a Modbus master control system and implement the Modbus addresses in the SIMATIC memory areas.
S_USST	The instruction S_USST processes the transfer of the network data (PZD and PKW data) to the slaves depending on the network data structure being used.
S_USSR	The instruction S_USSR processes the reception of the network data (PZD and PKW data) from the slaves depending on the network data structure being used.
S_USSI	If this instruction S_USSI is called upon starting up the S7-system, communication processor DB, network DB and parameter DB are created, which are required for communication.

Table 1-106 Instructions for Modbus

Instruction	Meaning
MODB_341	With the instruction MODB_341 and the affiliated driver, a communication connection between a modbus master control system and the communication module CP 341 "modbus-capable" slave system is allowed for.
MODB_441	With the instruction MODB_441 and the affiliated driver, a communication connection between a modbus master control system and the communication module CP 441-2 "modbus-capable" slave system is allowed for.

Instructions from the automation system S7-400

In the following table, you will find the instructions from the automation system S7-400, which are available for communication between CPU and CP 441.

Table 1-107 Instructions from the automation system S7-400

Instruction	Meaning
BSEND	With the instruction BSEND, you can send data from a S7-data range to a communication partner with a destination.
BRCV	With the instruction BRCV, you can receive the data from a communication partner and transfer it to a S7-data range.
GET	RK512 only: With the instruction GET, you can fetch the data from a communication partner.
PUT	RK512 only: With the instruction PUT, you can send the data to the communication partner with dynamic destination addresses..
PRINT	With the instruction PRINT, you can print a message text on a printer with up to four variables.
STATUS	With the instruction STATUS, you can call the status of a communication connection.

Further Information

A detailed description of the instructions from the automation system S7-400 and their settings can be found under: Overview of the instructions for S7 communication

Basic information (S7-300, S7-400)

Serial transfer of a character (S7-300, S7-400)

Introduction

To exchange data between two or more communication partners, there are different networking possibilities available. The point-to-point connection between two communication partners is the simplest case of information exchange.

Point-to-point connection

For the point-to-point connection, the communication module forms the interface between a programmable logic controller and a communication partner. Data is transferred serially with the communication module for the point-to-point connection.

The addressing mechanisms of the selected transmission procedure areimplemented on the communication module. Therefore, the point-to-point connection is closed on the communication module and not on the communication partner, as it is with other connections.

Serial data transmission

During serial data transmission, the individual bits from a byte, which must be transferred, are transmitted sequentially in a determined sequence.

Unidirectional/bidirectional data traffic

Data transmission with the communication partner is processed independently by the communication module via the serial interface. The communication module is equipped with various drivers to accomplish this.

- unidirectional data traffic:
  - Printer driver
- bidirectional data traffic:
  - ASCII driver
  - 3964(R) procedure
  - Computer connection RK512
  - Modbus/USS

Data transmission via the serial interface is processed by the communication module depending on the physical interface and the selected driver.

Unidirectional data traffic - printout

During the printout (printer driver), n bytes of user data are output to a printer. A reception of characters does not occur. Individual control characters for data flow control (for example, XON/XOFF) are exceptions to this.

Bidirectional data traffic - operating modes

With bi-directional data traffic, there are two different operating modes for the communication module:

- Half-duplex mode (3964 (R procedure), ASCII driver, RK512, Modbus/USS)
  The data is transferred alternating in both directions between one or multiple communication partners. Half-duplex mode means that everything is sent or received at one point in time. The exceptions to this can be individual control characters for data flow control (for example, XON/XOFF), which can also be received/sent during transmission/reception operation.
- Full-duplex mode (ASCII-driver)
  The data is exchanged simultaneously between one or more communication partners; it may be sent and received at one point in time. Each communication partner must be able to simultaneously operate a transmission and reception set-up.

Only half-duplex mode can be used with an X27 interface module (RS422/485) set to RS485 (2-wire).

Asynchronous data transmission

Serial data transmission is performed asynchronously with the communication module. The so-called time grid simultaneous operation (fixed time grid for the transfer of a fixed character string) must only be maintained during the transmission of a character. Each transferred character precedes a synchronization impulse, also referred to as a start bit. The length of the start bit transmission is determined by the clock. The end of the character transfer is formed by the stop bit.

Agreements

In addition the start and stop bits, there are other agreements between the two communication partners required for serial data transmission. This includes:

- Transmission speed (baud rate)
- Character and acknowledgement delay time
- Parity
- Number of data bits
- Number of stop bits

Character frame

The data between communication module and a communication partner is transmitted via the serial interface in a 10, 11 or 12 bit character frame. Several data formats are available for each character frame. Configure the desired format of the data transmission in the properties dialog of the module. Depending on the particular communication module, not all the theoretically possible character frames (combinations) are available.

10-Bit Character Frame

In the following illustration, three data formats of the 10-bit character frame are shown as an example.

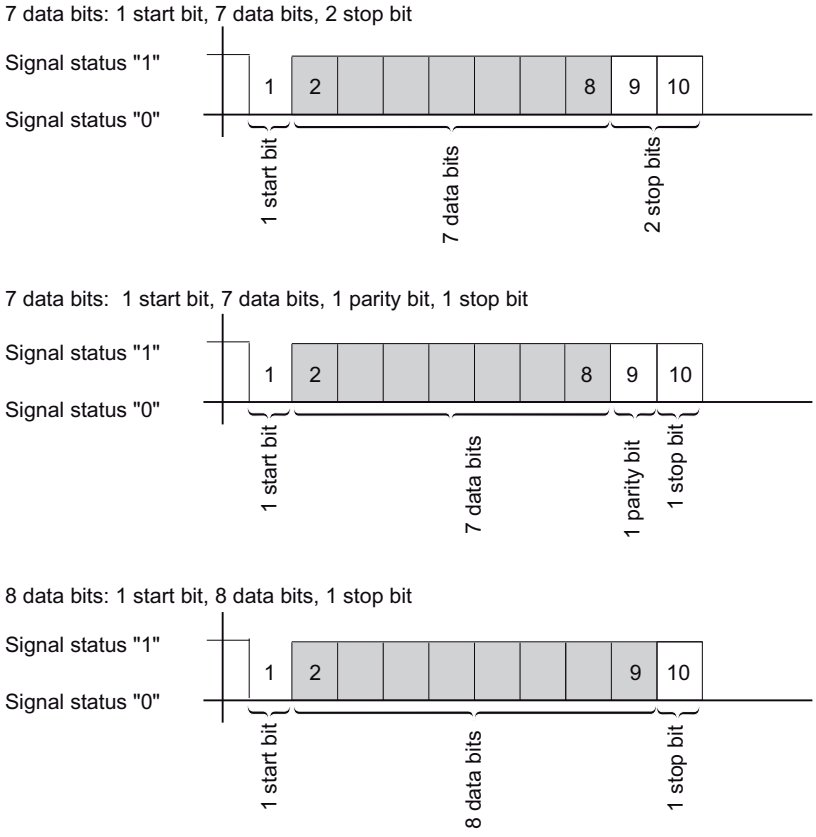

Figure 1-40 10-Bit Character Frame

11-Bit Character Frame

In the following illustration, three data formats of the 11-bit character frame are shown as an example.

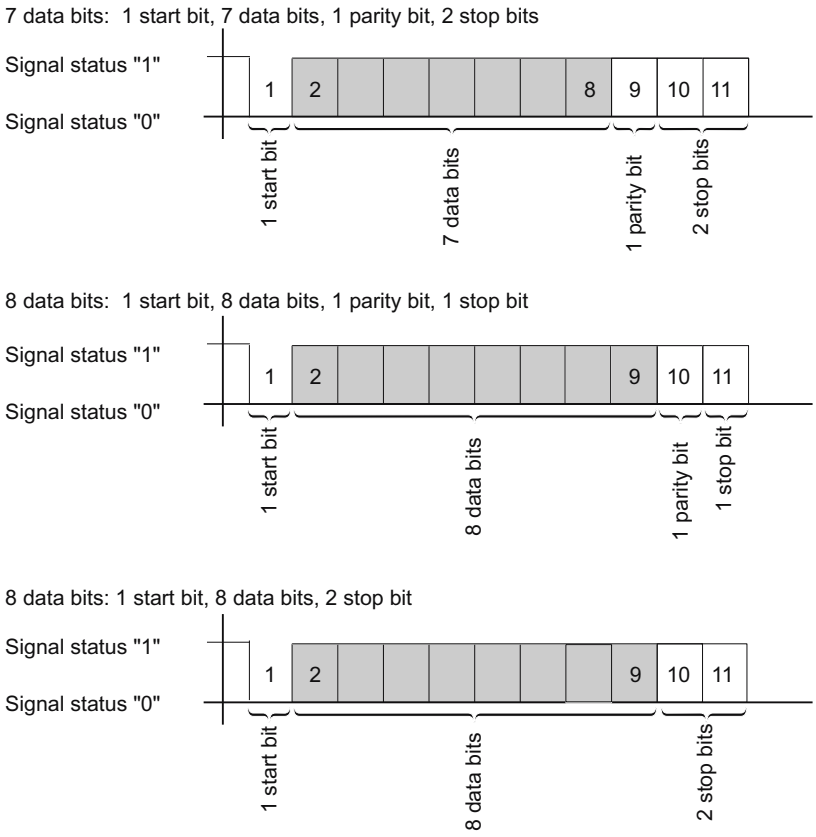

Figure 1-41 11-Bit Character Frame

Character Delay Time

The following illustration shows the maximum permissible interval between two received characters within a message frame (character delay time).

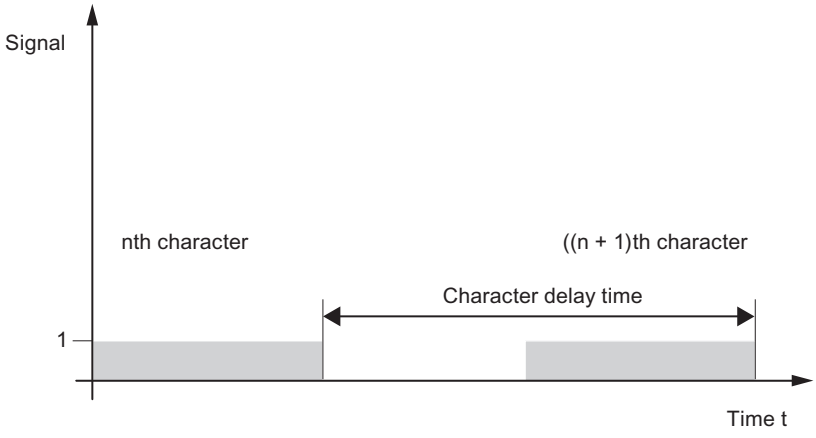

Figure 1-42 Character Delay Time

Transmission integrity (S7-300, S7-400)

Transmission integrity plays an important role in the transmission of data and in selection of the transmission procedure. Generally speaking, the more layers of the reference model are applied, the greater the transmission integrity.

Classifying the supplied protocols

The figure below illustrates how these protocols of the communication module fit into the reference model:

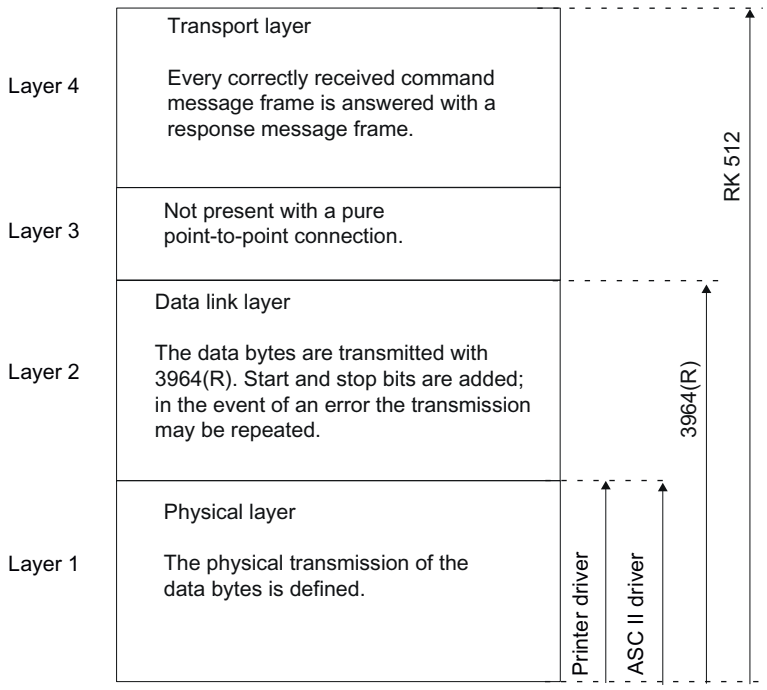

Figure 1-43 Classification of the existing protocols of the communication module in the reference model

Mapping of protocols to the communication modules

Protocol	CP 340	CP 341	CPU 313 C-2 PtP	CPU 314C-2 PtP	CP 440	CP 441	ET 200S 1SI
Printer driver	●	●				●	
ASCII driver	●	●	●	●	●	●	●
3964(R) procedure	●	●	●	●	●	●	●
Computer connection RK512		●		●		● 2)	
Modbus master		● 1)				● 1) 2)	●

Protocol	CP 340	CP 341	CPU 313 C-2 PtP	CPU 314C-2 PtP	CP 440	CP 441	ET 200S 1SI
Modbus slave		● [1]				● [1] [2]	●
USS-master							●

[1] A loadable driver with a dongle is needed for the Modbus protocol.

[2] only CP 441-2

Transmission Integrity with the printer driver

Data integrity when using the printer driver:

- No data integrity precautions are taken for data transport with the printer driver.
- To prevent data from being lost in the event of the printer receive buffer overflowing, you can work with data flow control (XON/XOFF, RTS/CTS).
- When data is output to the printer, the printer's BUSY signal is evaluated. The communication module receives the BUSY signal as a CTS signal and evaluates it in the same way (see ASCII driver). Note that, when using CTS/RTS flow control, you must set the polarity of the BUSY signal to CTS = "OFF" on the printer.

Transmission Integrity with the ASCII driver

Data integrity when using the ASCII driver:

- When data is transmitted via the ASCII driver, there are no data integrity precautions other than the use of a parity bit (can also be canceled, depending on how the character frame is set). This means that, although this type of data transport has a very efficient throughput rate, security is not guaranteed.
- Using the parity bit makes it possible to detect an inverted bit in a character that is to be transmitted. If two or more bits of a character are inverted, this error can no longer be detected.
- To increase transmission integrity, a checksum and length specification for a message frame can be employed. These measures must be implemented by the user.
- A further increase in data integrity can be achieved by means of acknowledgment message frames in response to send or receive message frames. This is the case with high-level protocols for data communication (ISO 7-layer reference model).

Transmission Integrity with 3964(R)

Enhanced data integrity through use of the 3964(R) procedure:

- The Hamming distance with the 3964(R) is 3. This measures the integrity of a data transmission.
- The 3964(R) procedure ensures high transmission integrity on the transmission line. This high transmission integrity is achieved by means of a specified message-frame setup and release as well as the use of a block check character (BCC).

Two different procedures for data transmission can be used, either with or without a block check character:

- Data transmission without a block check character: **3964**
- Data transmission with a block check character: **3964(R)**

In this manual, the designation 3964(R) is used when descriptions and notes refer to both data transmission procedures.

Transmission integrity with RK512

Very high data security using RK512:

- The Hamming distance with the RK512 and 3964R is 4. The Hamming distance is a measure of the integrity for a data transmission.
- Using the RK512 computer connection ensures high transmission integrity on the transmission line (because RK512 uses the 3964R procedure for data transport).
- Further processing at the communication partner is ensured (because the RK512 interpreter checks the additional length specification in the header and, after storing the data in the destination data area of the communication partner, generates a message frame acknowledging the success or failure of the data transport).
- The RK512 computer connection independently guarantees the correct use of the 3964R procedure and the analysis/addition of the length specification as well as the generation of the response telegram. There is no user handling! All you need to do is evaluate the positive/negative final acknowledgment.

Transmission integrity for Modbus/USS

Data security when using the Modbus/USS driver.

- The Hamming distance for Modbus/USS is 4. The Hamming distance is a measure of the integrity for a data transmission.
- Using Modbus/USS guarantees high transmission integrity on the transmission line (through a CRC16 block check for Modbus and an XOR block check for USS).
- Further processing in the communication partner is assured because both Modbus and USS evaluate additional length information in the header and generate a (time-monitored) acknowledgment message frame relating to the successful/unsuccessful data transmission.

Data transmission with ASCII driver (S7-300, S7-400)

Data Transmission with the ASCII Driver (S7-300, S7-400)

Assignment of the protocols to the communication modules, see: Overview of the components and interfaces (Page 1577)

Introduction

The ASCII driver controls data transmission via a point-to-point connection between the communication processor and a communication partner. This driver contains the physical layer (layer 1).

The structure of the message frames is left open through the S7 user passing on the complete send message frame to the communication processor. For the receive direction, the end criterion of a message must be configured. The structure of the send message frames may differ from that of the receive message frames.

The ASCII driver allows data of any structure (all printable ASCII characters as well as all other characters from 00 through FFH (with 8 data bit character frames) or from 00 through 7FH (with 7 data bit character frames) to be sent and received.

Sending Data with the ASCII Driver (S7-300, S7-400)

Sending Data

For sending, you specify the number of user data bytes to be transmitted in the length parameter when you call the respective send instruction. For more information, see the section "Overview of the instructions (Page 1579)".

When you work with the end criterion "Character Delay Time" when receiving data, the ASCII driver pauses between two message frames when sending. You can call the send instruction at any time, but the ASCII driver only begins output when the time since the last message frame was sent is longer than configured character delay time.

Note

When XON/XOFF flow control is configured, the user data must not contain the configured XON or XOFF characters. The default settings are DC1 = 11H for XON and DC3 = 13H for XOFF.

If you work with the "end delimiter" criterion, you have a choice of three options:

- Send up to and including the end delimiter
  The end delimiter must be included in the data to be sent. Data is sent only up to and including the end delimiter, even if the data length specified in the instruction is longer.
- Send up to the length configured at the instruction
  Data is sent up to the length that is configured at the instruction. The last character must be the end delimiter.
- Send up to the length configured at the instruction and automatically append the end delimiter(s)
  Data is sent up to the length that is configured at the instruction. The end delimiter is automatically appended, in other words the end delimiters must not be included in the data to be sent. 1 or 2 characters more than the number specified at the instruction are sent to the partner, depending on the number of end delimiters.

When you work with the "fixed message frame length" end criterion, the amount of data transferred in the send direction is what you have specified for the length parameter of the instruction. The amount of data transferred in the receive direction, in other words, into the

receive DB, is specified at the receiver using the "fixed message frame length" parameter in the properties dialog of the module. The two parameter settings must be identical, in order to ensure correct data traffic. A pause equal to the length of the character delay time (CDT) is inserted between two message frames when sending, to allow the partner to synchronize (recognize start of message frame).

If some other method of synchronization is used, the pause in sending can be deactivated in the properties dialog of the module.

Send operation

The figure below illustrates a send operation.

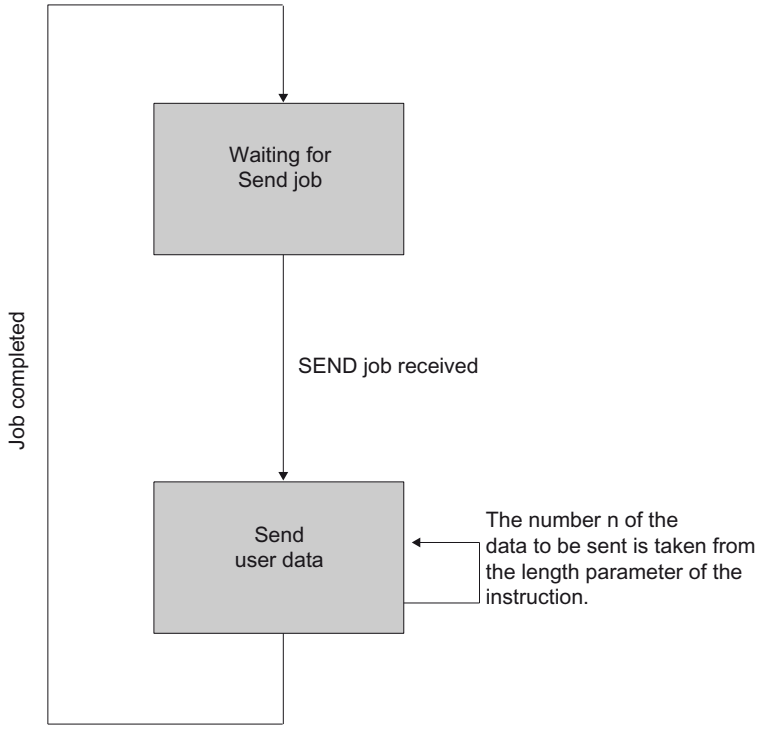

Figure 1-44 ASCII driver, sequence of sending data

Receiving Data with the ASCII Driver (S7-300, S7-400)

Selectable end criteria

For data transmission using the ASCII driver you can choose between three different end criteria. The end criterion defines when a complete message frame is received. The possible end criteria are as follows:

- After character delay time elapses
  The message frame has neither a fixed length nor a defined end delimiter; the end of the message is defined by a pause on the line (After character delay time elapses).
- On receipt of the end delimiter(s)
  The end of the message frame is marked by one or two defined end delimiters.
- On receipt of fixed number of characters
  The length of the receive message frames is always identical.

Code transparency

The code transparency of the procedure depends on the choice of configured end criterion and flow control:

- With one or two end delimiters
  - not code-transparent
- When end criterion is character delay time or fixed message frame length
  - code-transparent
- Code-transparent operation is not possible when the flow control XON/XOFF is used.

Code-transparent means that any character combinations can occur in the user data without the end criterion being recognized.

End criterion "After character delay time elapses"

When data is received, the end of the message frame is recognized when the character delay time expires. The received data is accepted from the CPU.

In this case, the character delay time must be set so that it reliably expires between two consecutive message frames. But it should be long enough so that the end of the message frame is not falsely identified whenever the partner in the connection takes a send pause within a message frame.

The figure below illustrates a receive operation with the end criterion "After character delay time elapses".

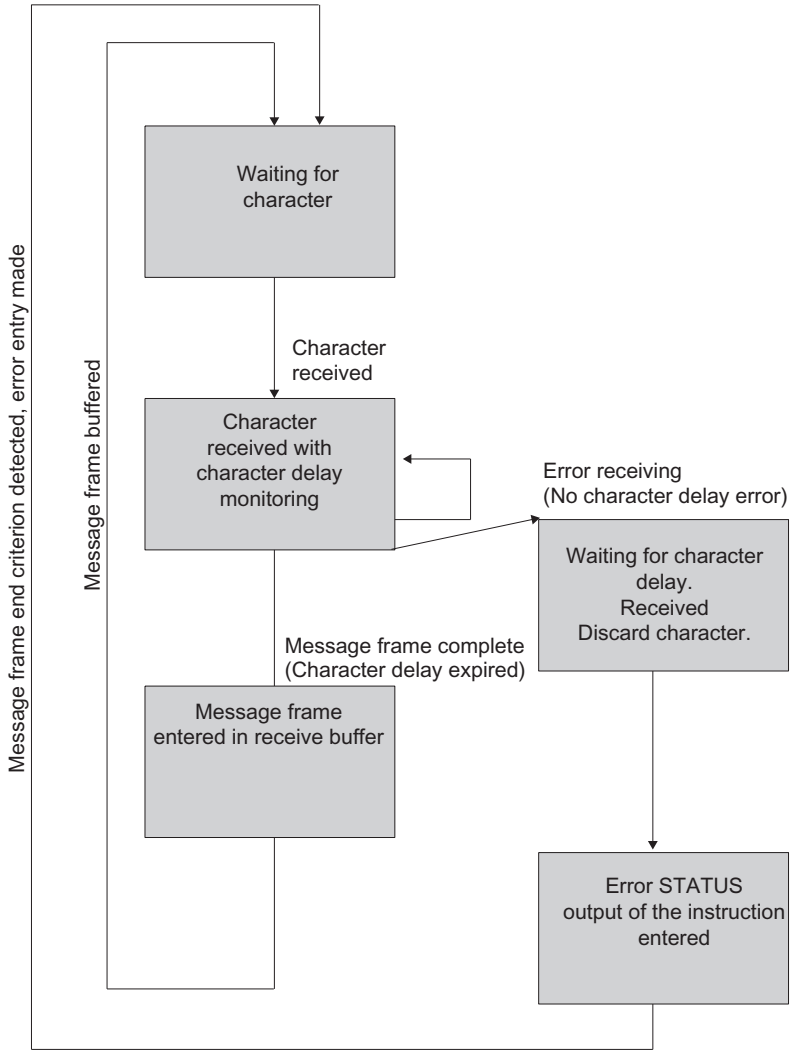

Figure 1-45 Sequence of Receive Operation with End Criterion "Expiration of Character Delay Time"

End criterion end delimiter

When data is received, the end of the message frame is recognized when the configured end delimiter(s) arrive. The received data including the end delimiter(s) is accepted from the CPU.

If the character delay time expires while the message frame is being received, the receive operation is terminated. An error message is issued and the message frame fragment is discarded.

If you are working with end delimiters, transmission is not code-transparent, and you must make sure that the end code(s) are not in the user data of the user.

Note the following when the last character in the received message frame is not the end delimiter.

- End delimiter elsewhere in the message frame:
  All characters including the end delimiter are entered in the receive DB. The characters following the end delimiter
  - is discarded if the character delay time (CDT) expires at the end of the message frame.
  - is merged with the next message frame if a new message frame is received before the character delay time expires.
- End delimiter not included in message frame:
  The message frame
  - is discarded if the character delay time (CDT) expires at the end of the message frame.
  - is merged with the next message frame if a new message frame is received before the character delay time expires.

The figure below illustrates a receive operation with the end criterion "end delimiter".

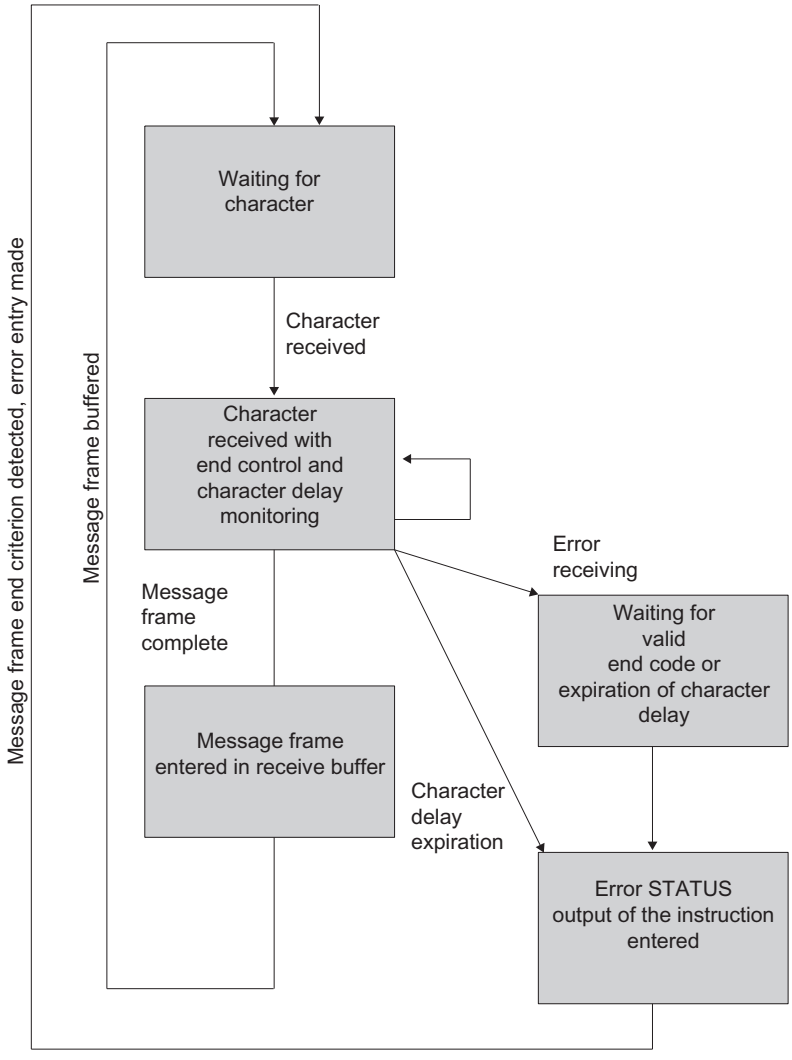

Figure 1-46 Sequence of receive operation with "End-of-Text Character" end criterion

End criterion fixed message frame length

When data is received, the end of the message frame is recognized when the configured number of characters has arrived. The received data is accepted from the CPU.

If the character delay time expires before the configured number of characters has been reached, the receive operation is terminated. An error message is issued and the message frame fragment is discarded.

Note the following if the message frame length of the received characters does not match the configured fixed message frame length:

- Message frame length of received characters greater than configured fixed message frame length:
  All characters received after the configured fixed message frame length is reached
  - is discarded if the character delay time (CDT) expires at the end of the message frame.
  - is merged with the next message frame if a new message frame is received before the character delay time expires.
- Message frame length of received characters less than configured fixed message frame length:
  The message frame
  - is discarded if the character delay time (CDT) expires at the end of the message frame.
  - is merged with the next message frame if a new message frame is received before the character delay time expires.

The figure below illustrates a receive operation with the end criterion "fixed message frame length".

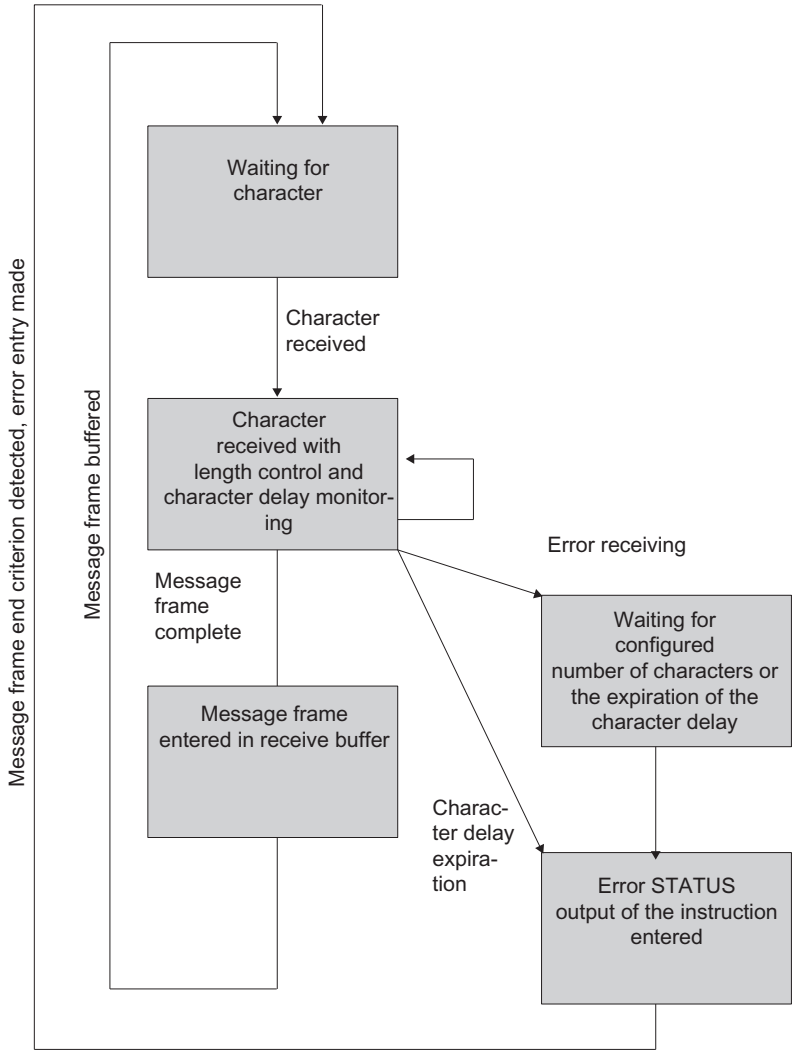

Figure 1-47 Sequence of receive operation with "Fixed Message Frame Length" end criterion

Receive buffer of the module

The receive buffer of the module can be up to up to 4096 bytes, depending on the communication module.

During configuration, you can specify if:

- CP and CPU 31xC-2 PtP
  the receive buffer should be deleted at startup or overwriting of data in the receive buffer should be prevented. You can also specify the value range (1 to 250) for the number of buffered receive message frames.
- ET 200S 1SI
  the receive buffer should be deleted on startup and if an overwriting of the data in the receive buffer should be avoided. Additionally, you can activate or block the buffering of received message frames.

The receive buffer on the module is a ring buffer:

- If several message frames are entered in the receive buffer of the module, the following applies: it is always the oldest message frame that is sent from the module to the CPU.
- If you only want to transfer the newest message frame, you must:
  - CP and CPU 31xC-2 PtP
    set "1" as the number of buffered message frames and disable the overwrite protection.
  - ET 200S 1SI
    block the dynamic message frame and disable the overwrite protection.

Note

If constant reading of the received data from the user program is interrupted for a time, you may find that when the received data is requested again, the CPU first receives old message frames from the module before it receives the most recent one.

During this interruption, the old message frames were either on the way between the module and CPU or had already been received by the instruction.

RS485 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (S7-300, S7-400)

RS485 mode

When you run the ASCII driver in RS485 mode (half-duplex, two-wire mode), you must take steps in the user program to ensure that only one user sends data at any one time. If two users send data simultaneously, the message frame is corrupted.

Switch-over Times for RS485 Module in Half-Duplex Mode

The maximum switch-over time between sending and receiving is 1 ms.

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (S7-300, S7-400)

RS232C accompanying signals

The following RS232C accompanying signals are available on the modules with the RS232C interface:

DCD	(input)	Data carrier detect
DTR	(Output)	Data terminal ready; module ready for operation
DSR	(input)	Data set ready; communication partner ready for operation
RTS	(Output)	Request to send, module ready to send

CTS	(input)	Clear to send; communication partner can receive data from the module (response to RTS = ON of the communication module)
RI	(input)	Ring indicator,

When the module is switched on, the output signals are in the OFF state (inactive).

You can configure the way in which the DTR/DSR and RTS/CTS control signals are used in the properties dialog of the module or control them via instructions in the user program.

Operate RS232C accompanying signals

The RS232C accompanying signals can be used as follows:

- When automatic use of all RS232C accompanying signals is configured
- When data flow control (RTS/CTS) is configured
- via the instructions V24_STAT and V24_SET

Note

When automatic operation of the R232C accompanying signals is configured, neither RTS/CTS data flow control nor RTS and DTR control by means of the V24_SET instruction are possible! When RTS/CTS data flow control is configured, RTS control by means of the V24_SET is not possible! On the other hand, it is always possible to read all RS232C accompanying signals using the V24_STAT instruction.

The sections that follow describe how the control and evaluation of the RS232C accompanying signals is handled.

Automatic use of accompanying signals

Automatic operation of the RS232C accompanying signals on the communication module is realized as follows:

- As soon as the module is switched by configuration to an operating mode with automatic operation of the RS232C accompanying signals, it sets RTS line to OFF and DTR line to ON (communication module ready for use).
  This prevents sending and receiving of message frames until the DTR line is set to ON. As long as DTR remains set to OFF, no data is received via the RS232C interface. If a send job is made, it is aborted with a corresponding error message.
- When a send job is made, RTS is set to ON and the configured data output waiting time starts. When the data output time elapses and CTS = ON, the data is sent via the RS232C interface.
- If the CTS line is not set to ON within the data output time so that data can be sent, or if CTS changes to OFF during transmission, the send job is aborted and an error message generated.
- Once the data has been sent and the configured clear RTS time has elapsed, the RTS line is set to OFF. The CP does not wait for CTS to change to OFF.

- Data can be received via the RS232C interface as soon as the DSR line is set to ON. If the receive buffer of the communication module threatens to overflow, there is no response from the communication module.
- If DSR changes from ON to OFF, an active send job as well as the receipt of data will be canceled with an error message. The message "DSR = OFF (automatic use of V24 signals)" is entered in the diagnostic buffer of the module .

Note

When automatic operation of the R232C accompanying signals is configured, neither RTS/CTS data flow control nor RTS and DTR control by means of the V24_SET instruction are possible!

The "Clear RTS time" must be set in the properties dialog of the module so that the communication partner can receive the last characters of the message frame in their entirety before RTS, and thus the send job, is revoked. The "data output waiting time" must be set so that the communication partner can be ready to receive before the time elapses.

Time Diagram

The figure illustrates the chronological sequence of a send job.

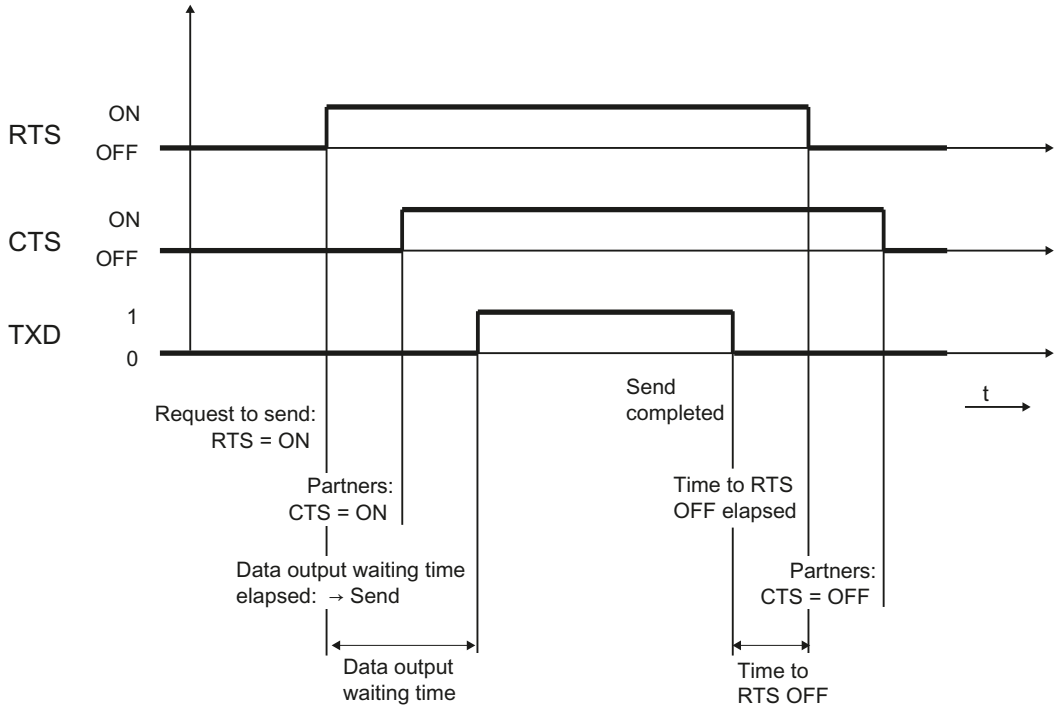

Figure 1-48 Time diagram of automatic use of the RS232C accompanying signals

Tasks of the instruction command V24_STAT/SET

The V24_STAT instruction command allows the status of each RS232C accompanying signal to be determined. The V24_SET allows the DTR and RTS output signals to be controlled.

Software/hardware handshake (S7-300, S7-400)

Introduction

Handshake protocol to control the flow of data between two communication partners. The use of the handshake procedure prevents data from getting lost during transmission when the two devices are operating at different rates.

Handshake procedure

There are essentially two types of handshaking here:

- Software handshaking (e.g. XON/XOFF)
- Hardware handshaking (e.g. RTS/CTS)

Data flow control is implemented as follows on the communication module:

- As soon as the module is switched by configuration to an operating mode with flow control, it sends the XON character or it sets the RTS line to ON.
- When the configured number of message frames is reached, or alternatively 50 characters before the receive buffer overflows (size of the receive buffer: 4096 bytes), the module sends XOFF character or sets the RTS line to OFF. If the communication partner continues to send data regardless of this, the receive buffer overflows and an error message is generated. The data received in the last message frame is discarded.
- As soon as a message frame is retrieved by the S7-CPU and the receive buffer is ready to receive, the module sends the XON character or sets the RTS line to ON.
- If the module receives the XOFF character, or the CTS control signal is set to OFF, the module interrupts the transmission. If neither an XON character is received nor CTS is set to ON after a configured time has elapsed, the transmission is aborted and an appropriate error message (0708H) is entered in the error message area of the module.

Note

When RTS/CTS data flow control is configured, you must fully wire the interface signals in the plug connection. When RTS/CTS data flow control is configured, RTS control by means of the V24_SET is not possible!

Data transmission with 3964(R) (S7-300, S7-400)

Data Transmission with the 3964(R) Procedure (S7-300, S7-400)

Assignment of the protocols to the communication modules, see: Overview of the components and interfaces (Page 1577)

Introduction

The 3964(R) procedure controls point-to-point data exchange between the communication processor and a communication partner. As well as the physical layer (layer 1), the 3964(R) procedure also incorporates the data-link layer (layer 2).

Startup

The figure below illustrates the start-up of the 3964(R) procedure.

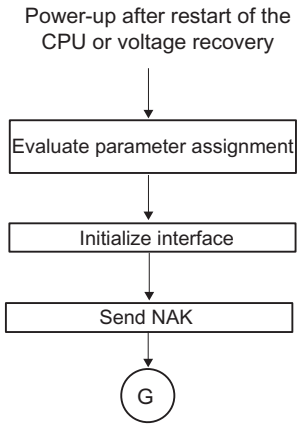

Figure 1-49 Flow diagram of the start-up of the 3964(R) procedure

Control characters (S7-300, S7-400)

Introduction

The RK 512 computer connection provides a very high degree of data integrity. During data transmission, the 3964(R) procedure adds control characters to the information data (data-connection layer). These control characters allow the communication partner to check whether the data has arrived complete and without errors.

The Control Characters of the 3964(R) Procedure

The 3964(R) procedure analyzes the following control codes:

- **STX**Start of Text; start of character string for transfer
- **DLE**Data Link Escape; data connection escape
- **ETX**End of Text; end of character string for transfer
- **BCC**Block Check Character (3964(R) only)
- **NAK**Negative Acknowledge

Note

If DLE is transmitted as an information string, it is sent twice so that it can be distinguished from the control code DLE during connection setup and release on the send line (DLE duplication). The receiver then reverses the DLE duplication.

Priority

With the 3964(R) procedure, one communication partner must be assigned a higher priority and the other partner a lower priority. If both partners begin connection setup at the same time, the partner with the lower priority will defer its send request.

Block Checksum (S7-300, S7-400)

Block Checksum

With the 3964(R) transmission protocol, data integrity is increased by the additional sending of a block check character (BCC).

Message frame:

STX	Data			DLE ETX		BCC
02H →	30H →	31H →	32H →	10H →	03H →	20H

30	=	0011	0000
31	=	0011	0001
XOR	=	0000	0001
32	=	0011	0010
XOR	=	0011	0011
10	=	0001	0000
XOR	=	0010	0011
03	=	0000	0011
XOR	=	0010	0000
BCC	→	2	0

Figure 1-50 Block Checksum

The block checksum is the even longitudinal parity (EXOR operation on all data bytes) of a sent or received block. Its calculation begins with the first byte of user data (first byte of the message frame) after the connection setup, and ends after the DLE ETX code on connection release.

Note

If DLE duplication occurs, the DLE code is accounted for twice in the BCC calculation.

Sending data with 3964(R) (S7-300, S7-400)

Process of Data Transmission when Sending

The figure below illustrates the transmission sequence when data is sent with the 3964(R) procedure.

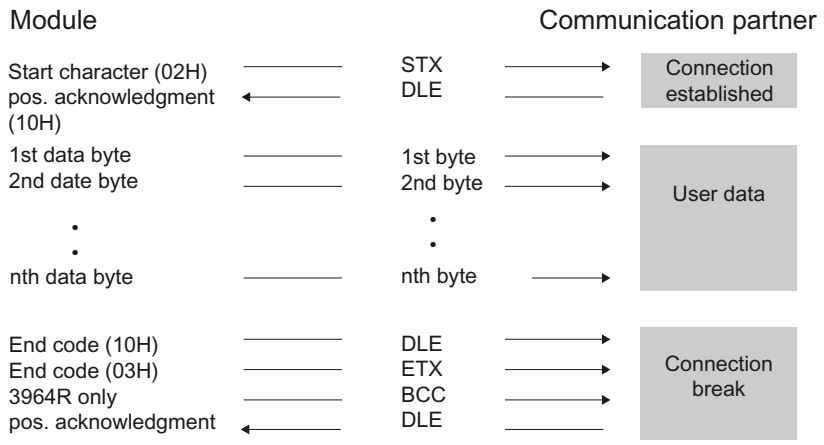

Figure 1-51 Data traffic when sending with the 3964(R) procedure

Establishing a Send Connection

To establish the connection, the 3964(R) procedure sends the control code STX. If the communication partner responds with the DLE code before the acknowledgment delay time expires, the procedure switches to send mode.

If the communication partner answers with NAK or with any other control code (except for DLE or STX), or the acknowledgment delay time expires without a response, the procedure repeats the connection setup. After the defined number of unsuccessful setup attempts, the procedure aborts the connection setup and sends the NAK code to the communication partner. The communication module enters a corresponding error number in the error message area.

Sending Data

If a connection is successfully established, the user data contained in the output buffer of the communication processor is sent to the communication partner with the chosen transmission parameters. The partner monitors the times between incoming characters. The interval between two characters must not exceed the character delay time.

If the communication partner sends the NAK control code during an active send operation, the procedure aborts its transmission of the block and tries again as described above, beginning with connection setup. If a different code is sent, the procedure first waits for the character delay time to expire and then sends the NAK code to change the mode of the communication

partner to idle. Then the procedure starts to send the data again with the connection setup STX.

Releasing a Send Connection

Once the contents of the buffer have been sent, the procedure adds the codes DLE, ETX and with the 3964R only the block checksum BCC as the end identifier, and waits for an acknowledgment code. If the communication partner sends the DLE code within the acknowledgment delay time, the data block has been received without errors. If the communication partner responds with NAK, any other code (except DLE), or a damaged code, or if the acknowledgment delay time expires without a response, the procedure starts to send the data again with the connection setup STX.

After the defined number of attempts to send the data block, the procedure stops trying and sends an NAK to the communication partner. The communication module reports the error in the error message area.

Sending with the 3964(R) Procedure

The figure below illustrates sending with the 3964(R) procedure.

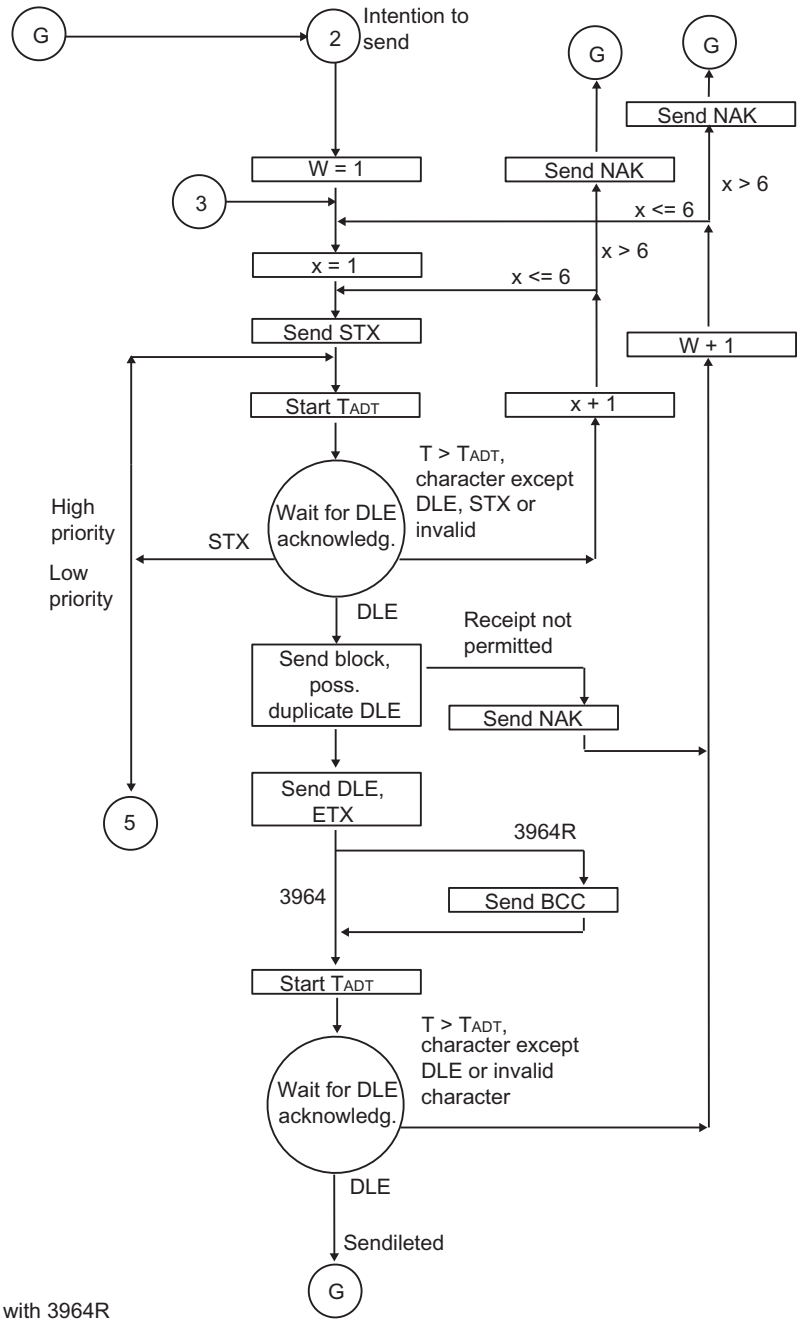

BCC only with 3964R
x = setup attempt count
T_{ADT} = 500 ms (3964R T_{ADT}=2s)
W = transmission attempt count
Immediate return to initial state at line break (BREAK)

Figure 1-52 Flow diagram of sending with the 3964(R) procedure

Receiving data with 3964(R) (S7-300, S7-400)

Process of Data Transmission when Receiving

The figure below illustrates the transmission sequence when data is received with the 3964(R) procedure.

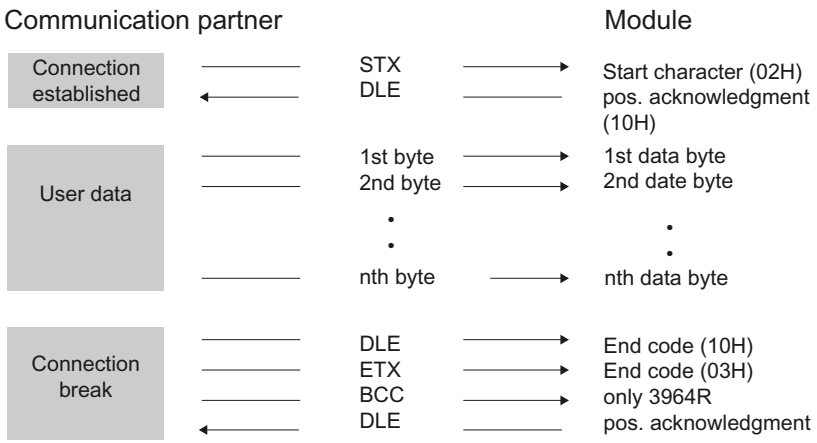

Figure 1-53 Data traffic when receiving with the 3964(R) procedure

Note

As soon as it is ready, the 3964(R) procedure sends a single NAK to the communication partner to set the latter to idle.

Establishing a Receive Connection

In idle mode, when there is no send job to be processed, the procedure waits for the communication partner to establish the connection.

If no empty receive buffer is available during a connection setup with STX, a wait time of 400 ms is started. If, after this time, there is no empty receive buffer, the system program reports the error in the error message area of the module. and the procedure sends a NAK and returns to idle mode. Otherwise, the procedure sends a DLE and receives the data as described above.

If the idle procedure receives any control code except for STX or NAK, it waits for the character delay time to expire, then sends the code NAK. The communication module reports the error in the error message area of the module.

receiving data

After a successful connection setup, the receive characters that are arrive are stored in the receive buffer. If two consecutive DLE codes are received, only one of these is stored in the receive buffer.

After each receive character, the procedure waits out the character delay time for the next character. If this period expires before another character is received, an NAK is sent to the

communication partner. The communication module reports the error in the error message area of the module. The 3964(R) procedure does not initiate a repetition.

If transmission errors occur during receiving (lost character, frame error, parity error, etc.), the procedure continues to receive until the connection is closed. NAK is then sent to the communication partner. A repetition is then expected. If the block still cannot be received after the number of transmission attempts defined in the static parameter set, or if the communication partner does not start the repetition within a block wait time of 4 seconds, the procedure aborts the receive operation. The communication module reports the first transmission error and the final abort in the error message area of the module.

Releasing a Receive Connection

When the 3964 procedure detects a DLE ETX character string, it ends the receiving operation and confirms the successfully received block by sending a DLE signal to the communication partner. When errors are found in the received data, it outputs a NAK signal to the communication partner. A repetition is then expected.

If the 3964R procedure recognizes the string DLE ETX BCC, it stops receiving and compares the received block check character with the longitudinal parity calculated internally. If the BCC is correct and no other receive errors have occurred, the CP sends the code DLE to the communication partner. If the BCC is correct and no other receive errors have occurred, the 3964R procedure sends a DLE and returns to idle mode. If the BCC is faulty or a different receiving error occurs, an NAK is sent to the communication partner. A repetition is then expected.

Receiving with the 3964(R) Procedure

The figure below illustrates receiving with the 3964(R) procedure.

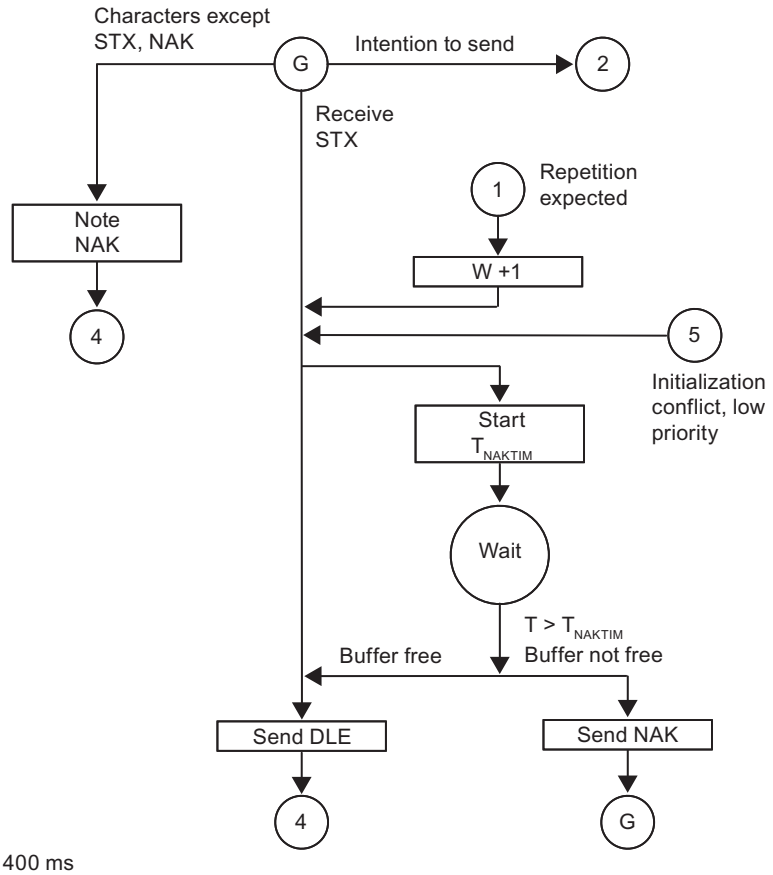

T_{NAKTIM} = 400 ms
W = transmission attempt counter
Immediate return to initial state at line break (BREAK)

Figure 1-54 Flow diagram of receiving with the 3964(R) procedure (part 1)

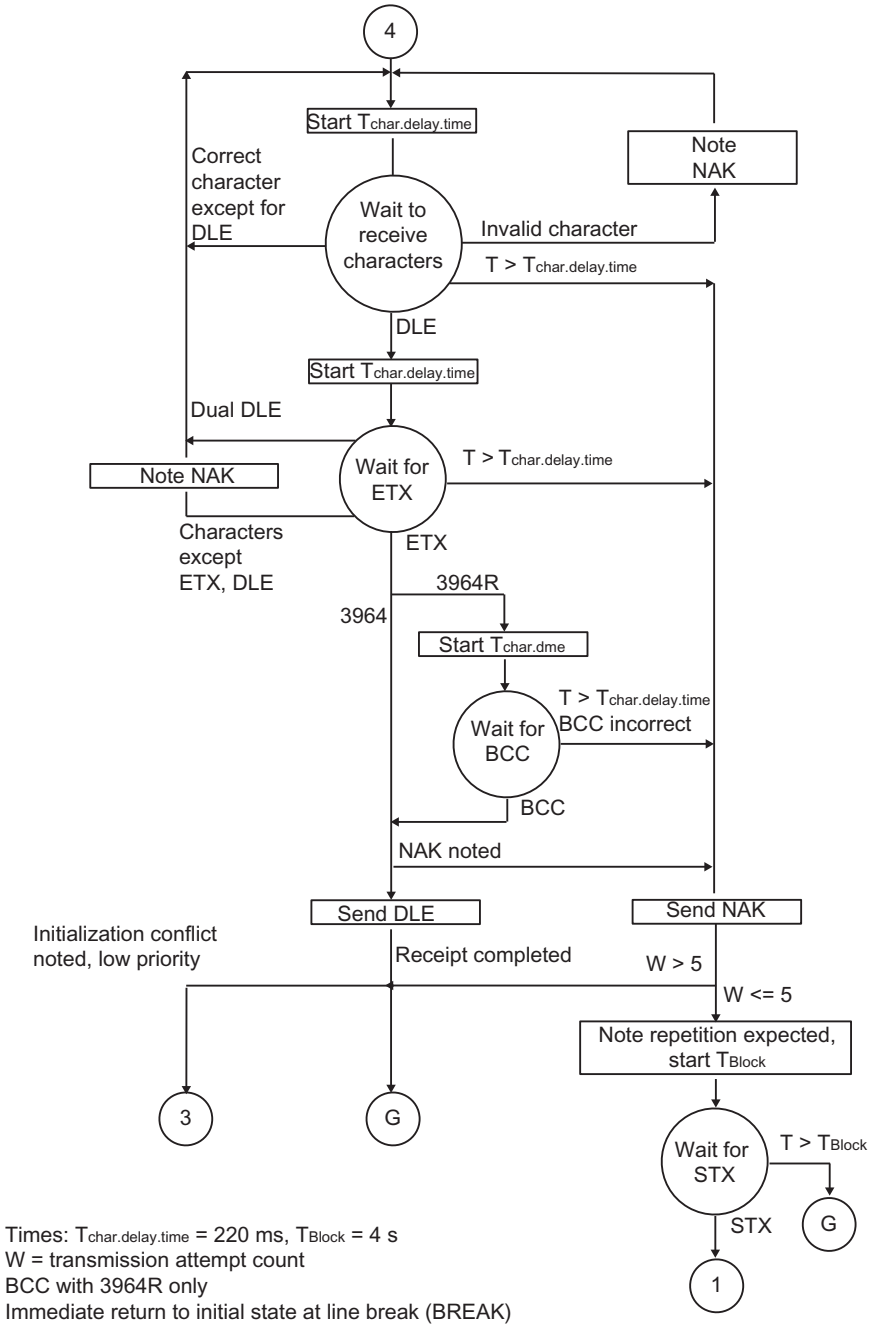

Figure 1-55 Flow diagram of receiving with the 3964(R) procedure (part 2)

Data transmission with Rk512 (S7-300, S7-400)

Data transmission with the RK512 computer connection (S7-300, S7-400)

Assignment of the protocols to the communication modules, see: Overview of the components and interfaces (Page 1577)

Introduction

The RK512 computer connection controls data transmission in a point-to-point connection between the communication module and a communication partner.

Unlike the 3964(R) procedure, the RK512 computer connection includes not only the physical layer (layer 1), and the data link layer (layer 2), but also the transport layer (layer 4) of the ISO reference model. The RK512 computer connection also offers greater data security and enhanced addressing options.

Response message frame

The RK512 computer connection answers each correctly received command message frame with a response message frame to the client (transport layer). This allows senders to check whether their data has arrived undamaged at the CPU of the communication partner or whether the data they require is available on the CPU of the communication partner.

Command message frame

Command message frames are either SEND frames or FETCH frames.

SEND frame

A SEND frame is created when the communication module sends a command message frame with user data, and the communication partner replies with a response message frame without user data.

FETCH frame

A FETCH frame is created when the communication module sends a command message frame without user data, and the communication partner replies with a response message frame with user data.

Continuation message frame

If the volume of data exceeds 128 bytes, SEND and FETCH frames are automatically sent with continuation message frames.

Message frame header

With RK512, each message frame begins with a message frame header. It can contain message frame IDs, information on the data destination and source and an error number.

Structure of the message frame header

The table below indicates the structure of the header of the command message frame.

Table 1-108 Format of the message frame header for the command message frame (RK512)

Byte	Meaning
1	Message frame ID in command message frames (00H), in continuation command message frames (FFH)
2	Message frame ID (00H)
3	'A' (41H) for SEND job with destination DB or 'O' (4FH) for SEND job with destination DX or 'E' (45H) for FETCH job
4	Data to be transmitted consist of (only 'D' possible for sending): 'D' (44H)=data block 'I' (45H) = Input bytes 'O' (41H) = output bytes 'M' (4DH)=flag bytes 'C' (5AH)= counter cells 'T' (54H) = time cells
5 and 6	Data destination of SEND/PUT job or data source of GET job e.g. byte 5 = DB no., byte 6 = DW no. (RK512 addressing describes data source and destination with word limits. Conversion to byte addresses in SIMATIC S7 is automatic.)
7 and 8	Length of high byte Length of data to be transmitted according to type in bytes or Length of low byte words
9	Byte number of the communication flag; FFH is shown here if you have not specified an interprocessor communication flag.
10	**Bit 0 to 3:** Bit number of the communication flag, The protocol enters FH here if you have not specified an interprocessor communication flag. **Bit 4 to 7:** CPU number (number from 1 to 4); If you have not specified a CPU number but you have specified an interprocessor communication flag, OH is displayed here; if you specified neither a CPU number nor an interprocessor communication flag, FH is shown here.

The letters in bytes 3 and 4 are ASCII characters.

The header of the continuation command ^message frame consists of bytes 1 to 4 only.

Response message frame

Once the command message frame has been transmitted, the RK512 expects a response message frame from the communication partner within the monitoring time. The length of the monitoring time is 20 seconds irrespective of the transmission speed (baud rate).

Depending on the PtP module used, this monitoring time can be reduced by user configuration in the properties dialog of the module. By selecting the "transmission-rate dependent" option, monitoring is performed with the following maximum waiting times:

Table 1-109 Transmission-rate dependent monitoring time for response telegram

Transmission rate	Monitoring time
• 300 baud	10 s
• 600 baud	7 s
• 1200 baud	5 s
• from 38400 baud	3 s

The "grayed" "Maximum wait time" field is used only for showing the configured monitoring time and cannot be edited!

Structure and contents of the response message frame

The response message frame consists of 4 bytes and contains information on the progress of the job.

Byte	Meaning
1	Message frame ID in response message frames (00H), in continuation response message frames (FFH)
2	Message frame ID (00H)
3	Displays 00H
4	Error number of the communication partner in the response message frame: • **00H** if transmission was error-free • **> 00H** error number The error number in the response message frame automatically causes an error number to be entered in the error message area of the module.

Sending data with RK512 (S7-300, S7-400)

Process of Data Transmission when Sending

The figure below shows the transmission sequence when sending data with a response message frame using the RK512 computer connection.

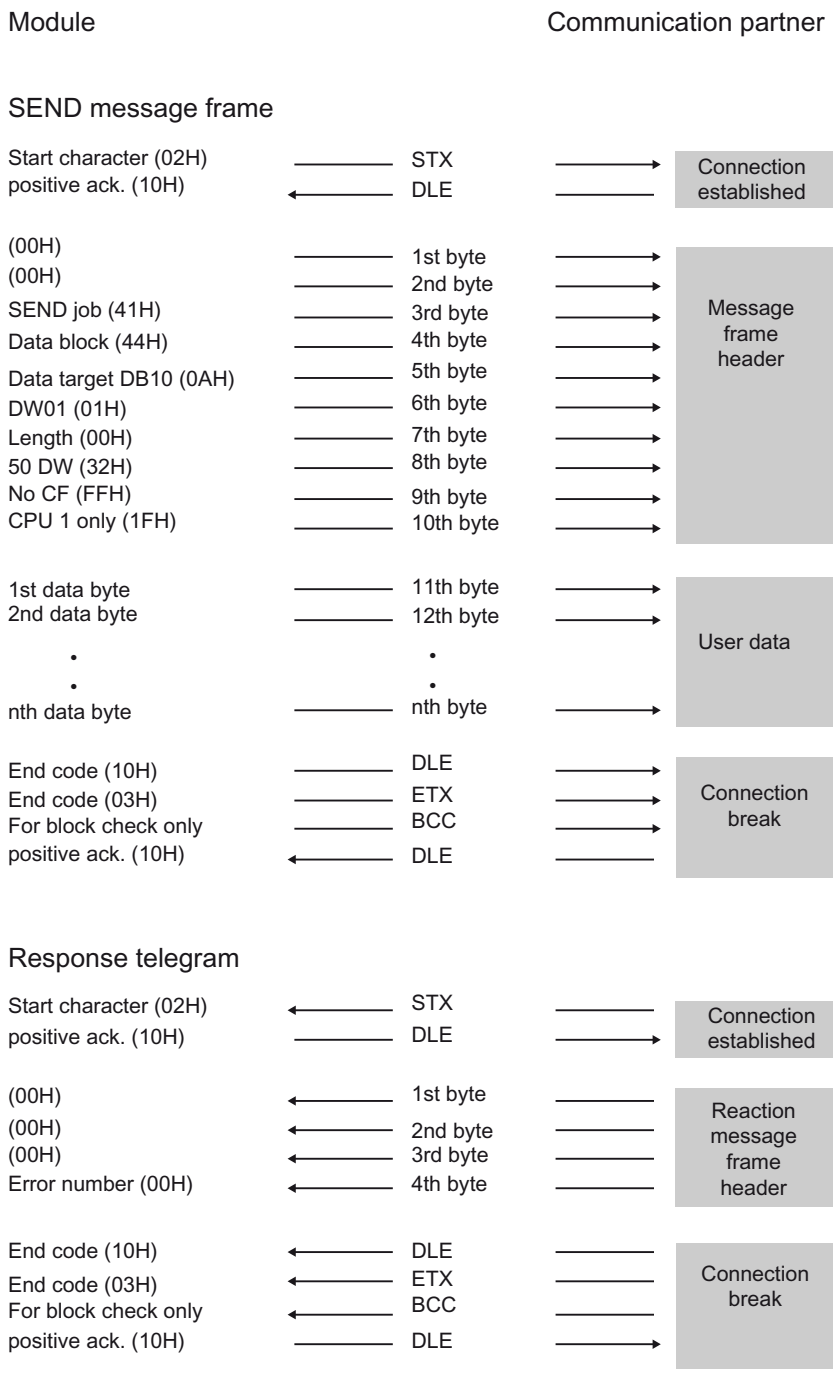

Figure 1-56 Data traffic when sending with a response frame

Sending Data

The SEND job is executed in the following sequence:

- Active partner
  Sends a SEND message frame. This contains a message frame header and data.
- Passive partner
  Receives the message frame, checks the header and the data, and acknowledges it with a response frame after passing the data on to the CPU.
- Active partner
  Receives the response message frame and terminates the SEND job.

Note

If the user data of the SEND job is not received correctly by the CPU or an error occurs in the message frame header, the communication partner enters an error number in the 4th byte of the response message frame. This does not apply when protocol errors occur.

Continuation SEND message frame

A continuation SEND job is started when the amount of data exceeds 128 bytes. The procedure is similar to the SEND job.

If more than 128 bytes are sent, the extra bytes are automatically transmitted in one or more continuation message frames. Each continuation SEND frame is acknowledged with a continuation response message frame.

The figure below shows the data transmission sequence when sending a continuation SEND message frame with a continuation response message frame.

The continuation response message frame for the last continuation SEND frame is only sent after successful transfer of the entire user data to the CPU.

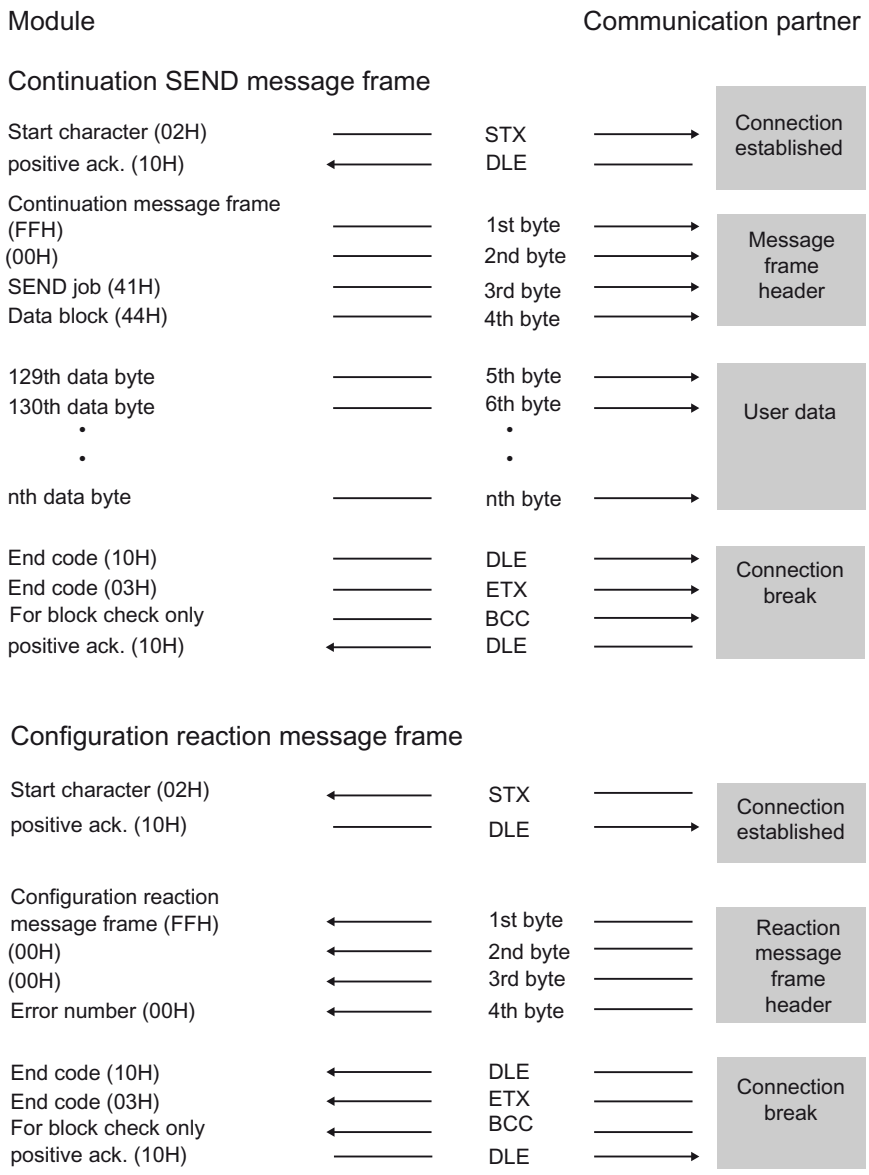

Figure 1-57 Sequence of a continuation SEND message frame with a continuation response message frame

Fetching data with RK512 (S7-300, S7-400)

Procedure for fetching data with RK512

The figure below shows the transmission sequence when fetching data with a response message frame using the RK512 computer connection.

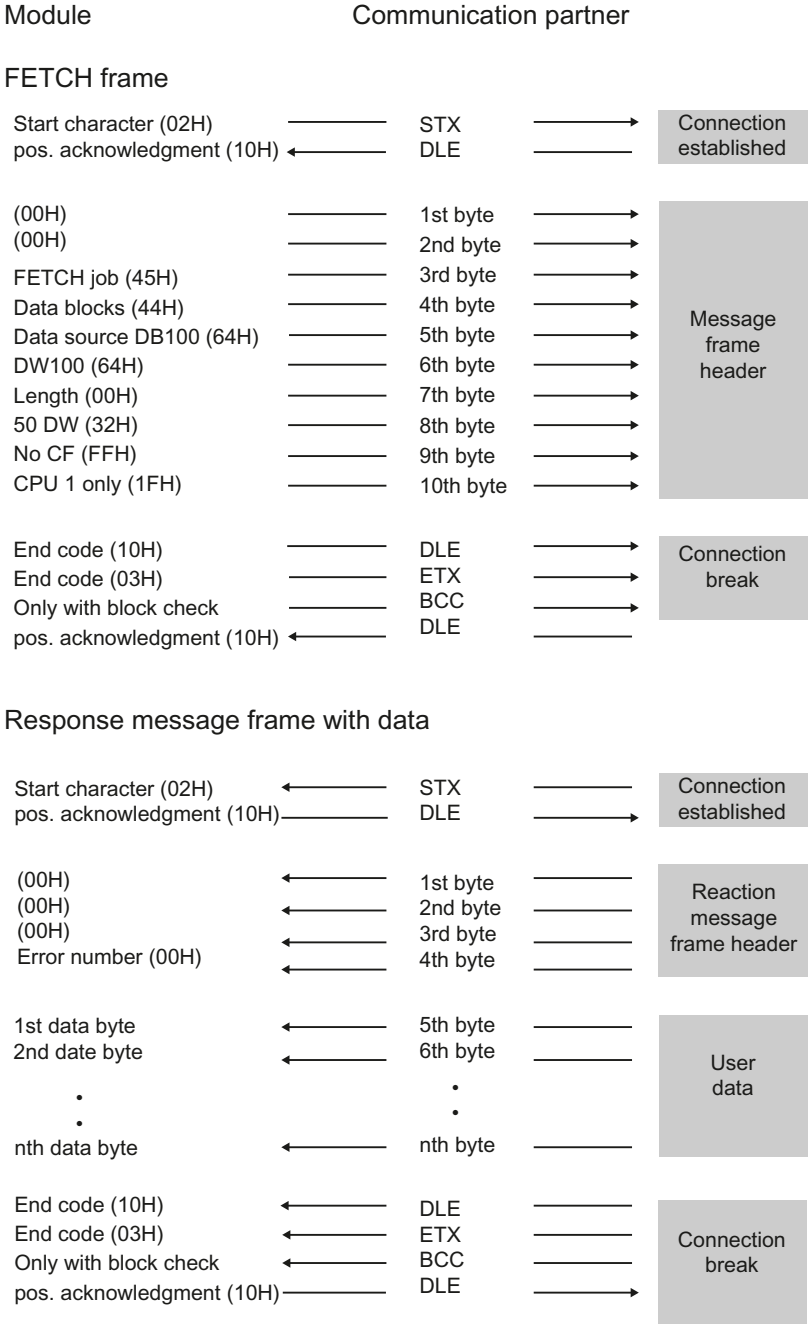

Figure 1-58 Data traffic when fetching with a response message frame

Fetching data

The FETCH job is executed in the following sequence:

- Active partner
  Sends a FETCH frame. This contains a message frame header.
- Passive partner
  Receives the message frame, checks the header, fetches the data from the CPU, and acknowledges this with a response frame containing the data. This contains the data.
- Active partner
  Receives the response message frame, passes user data received in the response message frame to the CPU, and terminates the FETCH job.

If there is an error number (not equal to 0) in the 4th byte, the response frame does not contain any data.

If more than 128 bytes are requested, the extra bytes are automatically fetched in one or more continuation message frames. Each continuation FETCH frame is acknowledged with a continuation response message frame (with more user data).

After receipt of the last continuation response message frame (with the last user data block), the entire user data are passed to the CPU and the FETCH job is completed.

Note

If the FETCH message frame is not correctly processed by the CPU or if an error occurs in the message frame header, the communication partner enters an error number in the 4th byte of the response message frame. This does not apply when protocol errors occur.

Continuation FETCH frame

The figure below shows the transmission sequence when fetching data with a continuation response message frame.

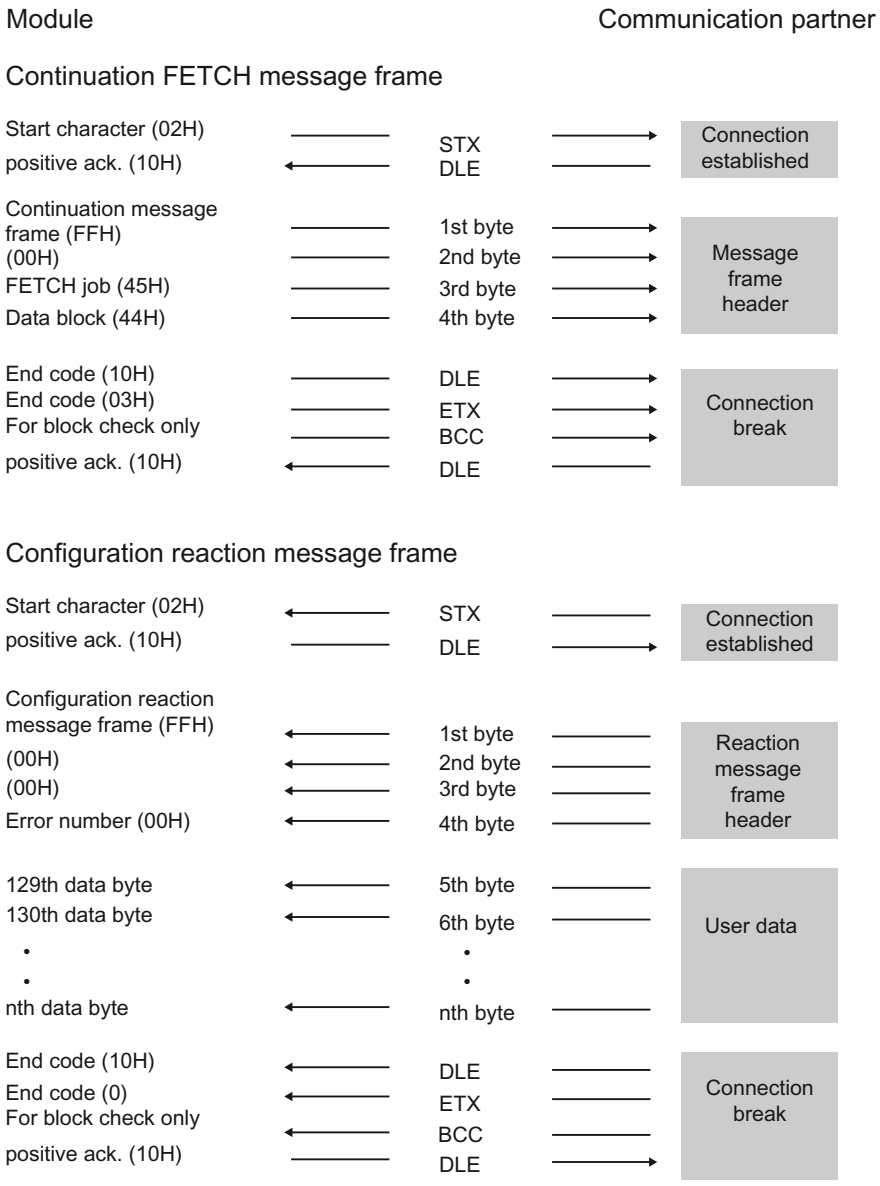

Figure 1-59 Sequence of a FETCH frame with a continuation response message frame

Quasi-full-duplex operation

Quasi-full-duplex mode means: The partners can send command and response frames at any time as long as the other partner is not sending. The maximum nesting depth for command and response frames is "1". The next command message frame, therefore, cannot be processed until the previous one has been answered with a response frame.

It is possible under certain circumstances - when both partners want to send - to transmit a SEND or FETCH message frame from the partner before the response message frame. This can occur, for example, when a SEND or FETCH message frame from the partner is entered in the output buffer of the communication module before the response message frame.

In the following figure, the continuation response message frame for the first SEND frame is only sent after the SEND frame of the partner.

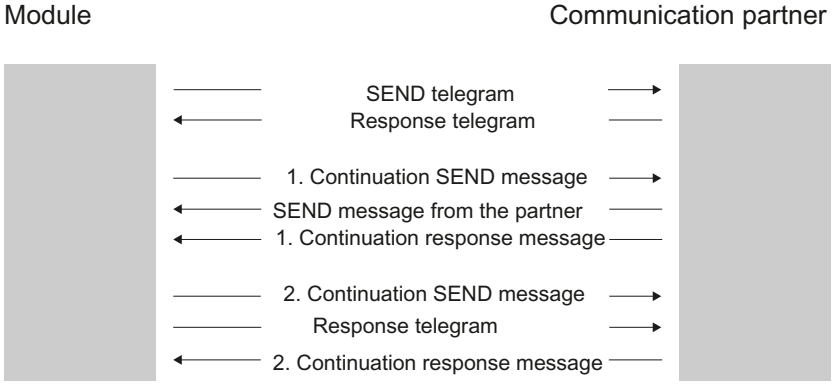

Figure 1-60 Quasi-full-duplex operation

RK512 CPU jobs

The figure below shows the sequences involved in the RK512 computer connection when CPU jobs are made.

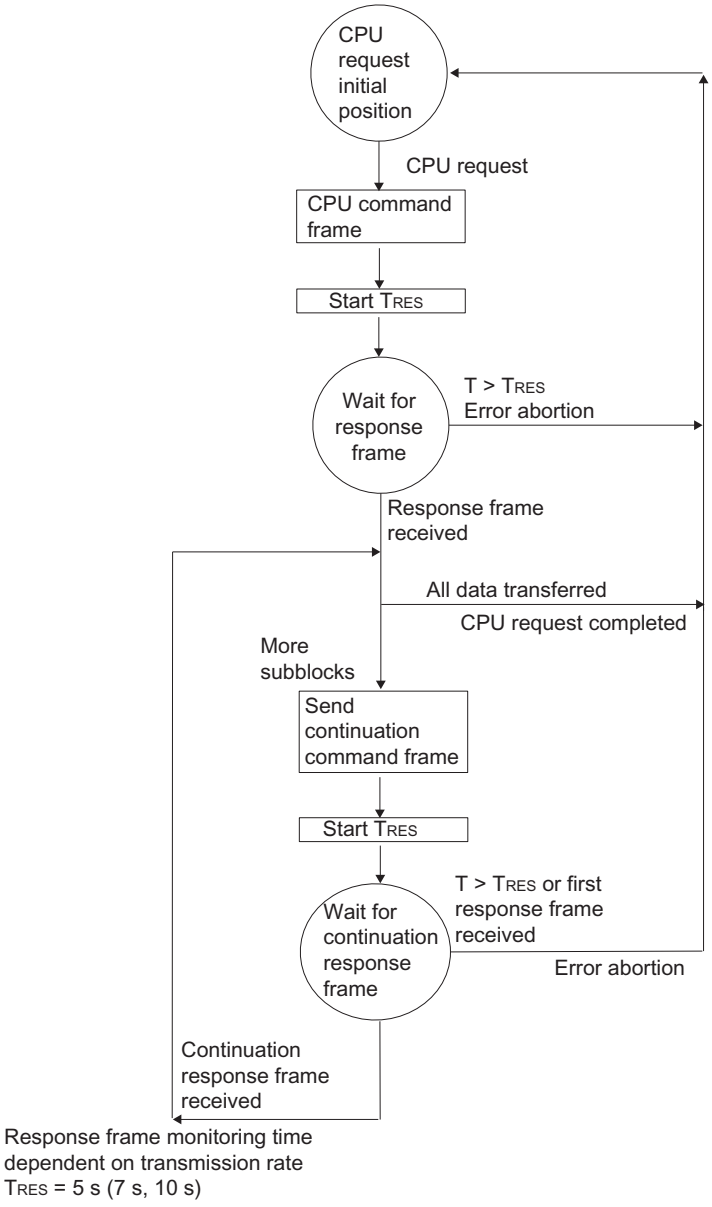

Figure 1-61 Sequence for data transmission from CPU jobs with RK512

RK512 partner jobs

The figure below shows the sequences involved in the RK512 computer connection when partner jobs are made.

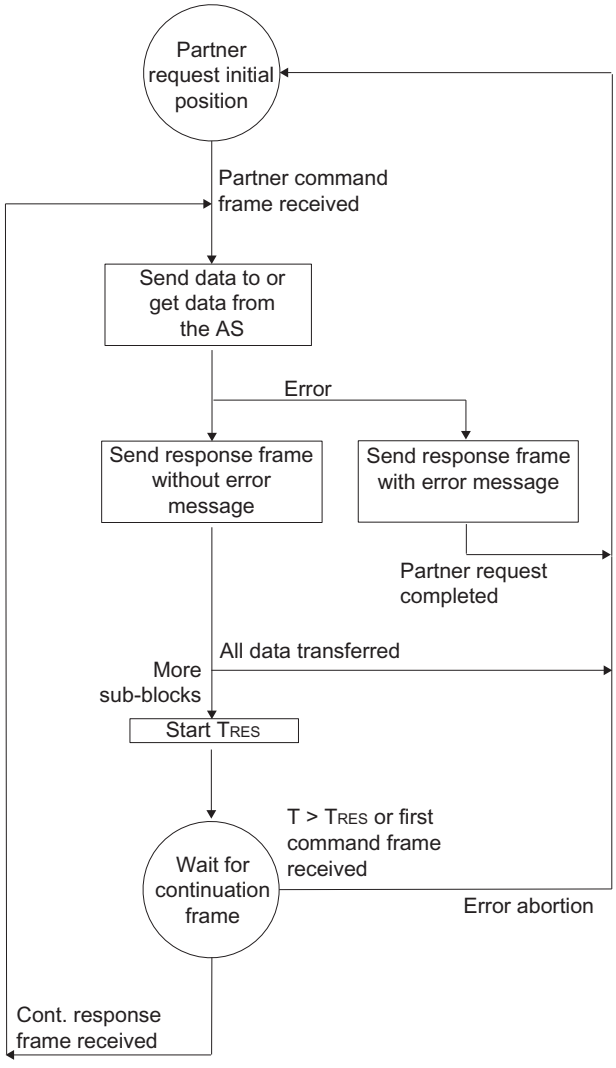

Response frame monitoring time dependent on transmission rate
TRES = 5 s (7 s, 10 s)
AS = automation system

Figure 1-62 Flow chart for data transmission from partner jobs with RK512

Data transmission with printer driver (S7-300, S7-400)

Data transmission with the printer driver (S7-300, S7-400)

Assignment of the protocols to the communication modules, see: Overview of the components and interfaces (Page 1577)

Introduction

The printer driver allows you to output message texts with the date and time to a printer. This enables you to monitor simple processes, print error or fault messages or issue instructions to the operating personnel, for example.

The printer driver contains the physical layer (layer 1).

Message texts and parameters for printout

In the properties dialog of the module, you configure message text and set the parameters (page layout, character set, control characters) for the printout. Message texts and printout parameters are transmitted to the communication module together with the module parameters when it starts up.

- Message texts:
  You can configure message texts with variables and control instructions (e.g. for bold, narrow, wide or italic type and underlining). Each message text is assigned a number during parameter assignment. The printout of a specific message text with CP 441 modules is made by specifying a reference (to the memory cell containing the message text no.) at the send parameters SD_1 to SD_4 of the instruction. With CP 34x modules, this is done by specifying the message text number in a format string when calling the instruction.
  You can also save configured message texts to an external text file (SDB name, max. 8 ASCII characters), and edit them with a suitable editor. In this way, the texts can be translated and then read back, for example.
  Structure of the message text file:
  <Versions number>
  <Number of the message>:<Content the message>
- Page layout:
  You can configure the margins, optional line breaks and headers and footers of the page layout.
- Character set:
  STEP 7 converts the ANSI character set to the printer character set using a character conversion table. You can change a character conversion table suggested for a printer type in order to include special characters required for a particular language, for example.
- Control characters:
  Using a control character table, you can change the control instructions in the message text for printer emulation to switch on and off bold, narrow, wide or italic type and underline, and to add control characters.

Variables

Up to 4 variables (3 + text message number) are displayed in a message text. Variable values can be transmitted from the CPU to the communication module. The following variables can be displayed:

- Calculated values of the user program (e.g. fill levels)
- Date and time
- Strings (string variable)
- Other message texts

A conversion instruction must be specified for each variable in the configured message text or in the format string, in which the meaning and the output format of the variable value are encrypted.

Format string

You can use the format string to define the appearance and the composition of a message text. The format string can consist of:

- Text (any printable characters, e.g. Fill level l reached at ...!)
- Conversion instructions for variables (such as %N = pointer to the message text number x, where x is a variable value (see Example 2 below))
  For each variable, exactly one conversion instruction must be present in the format string or configured message text. The conversion instructions are applied to the variables according to their order.
- Control instructions with control characters for bold, narrow, wide, italics, underline (e.g. \B = bold on) or with additional control characters you have defined

You can use additional control characters when you enter these characters into the control character table in the properties dialog of the module and configure the new communication module.

Other functions

In addition to the output of message texts, you can use the following functions for the printout. You run these functions from a format string.

- Set page number (format string = %P)
- Start new page (format string = \F)
- Print with/without line break (\x at the end of the format string)

Note that a line feed is performed by default for each printout.

Examples

Example 1: Fill level "200" l reach at "5:30 pm"!

Format string = Fill level %i l reach at %Z!
Variable 1 = time
Variable 2 = level

Example 2: The chamber pressure "drops"

Format string = %N %S
Variable 1 = 17 (message text no. 17: The chamber pressure ...)
Variable 2 = Reference to string (string variable: ... drops)

Example 3: (Set page number to 10)

Format string = %P
Variable 1 = 10 (page number: 10)

Printout

To output n bytes of user data to a printer, the format string and the variables of the message text must be specified as parameters when the instruction is called.

During output the data is edited for printing. Preparation for printing is carried out according to the parameters in the properties dialog of the module (page layout, character set, control characters, etc.).

Characters are not received during printout. The exception to this are any flow control characters that have been configured. Any characters received are not adopted.

Message text output

The figure below illustrates the sequence of operations at printout.

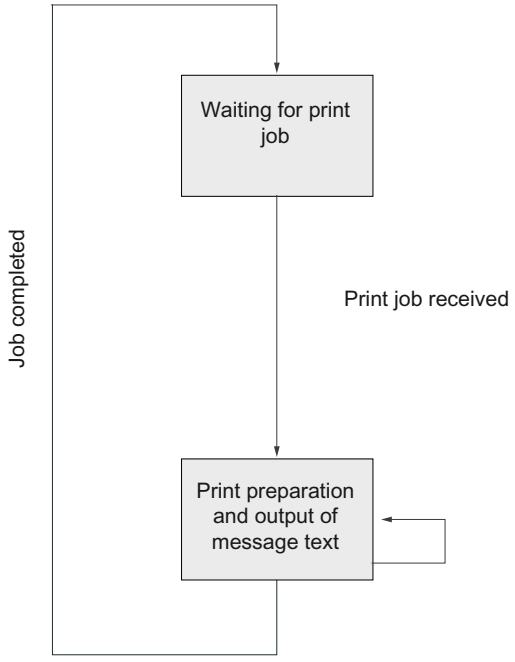

Figure 1-63 Flow chart of printout

Software/hardware handshake

Handshaking controls the data flow between two communication partners. Handshaking ensures that data is not lost in transmissions between devices that work at different speeds.

You can also send printout data with flow control. There are essentially two types of handshaking here:

- Software handshaking (e.g. XON/XOFF)
- Hardware handshaking (e.g. RTS/CTS)

Implementation of flow control on the communication module for printout is performed as follows:

- As soon as the communication module is switched by configuration to an operating mode with flow control, it sends the XON character or sets the RTS line to ON.
- If the communication module receives the XOFF character or the CTS control signal is set to OFF, the communication module interrupts the output of characters. If no XON is received or CTS is not set to ON after a configurable time, the printout is interrupted and a corresponding error message (0708H) is generated at the STATUS output of the instruction.

Note

If you configure RTS/CTS flow control, be sure the interface signals used are fully wired in making the plug connection.

BUSY signal

The communication module evaluates the "BUSY" control signal of the printer. The printer reports to CP 34x1 its readiness to receive

- With CP 34x 20mA-TTY: With power on RxD line.
- With CP 34x RS232C and CP 34x RS422/485: With CTS = "ON" signal.

Note

If you configure RTS/CTS flow control, you must set the polarity of the BUSY signal on the printer as follows:

- BUSY signal: CTS = "OFF"

Note that some printers show the BUSY signal using the DTR signal. In this case, you need to rewire the cable to the communication module correspondingly.

Indicator for end of a message freme (S7-300, S7-400)

End criteria

You can choose between three different end criteria for data transmission using the ASCII driver:

- **After character delay time elapses**
- **On receipt of the end delimiter(s)**
- **On receipt of fixed number of characters**

Protocol parameters

The following table provides a description of the protocol parameters.

Table 1-110 Protocol parameters (ASCII driver)

Parameters	Description	Value range	Default value
End detection indicator of a receive message frame	Specifying the criteria that determine the end of message frames.	• After character delay time elapses • On receipt of the end delimiter(s) • On receipt of fixed number of characters	After character delay time elapses
End delimiter 1 [1]	Code of the first end detection.	• With 7 data bits: 0 to 7FH (Hex) [2] • With 8 data bits: 0 to FFH (Hex) [2]	3 (03H = ETX)
End delimiter 2 [1]	Code of the second end detection, if selected.	• With 7 data bits: 0 to 7FH (Hex) [2] • With 8 data bits: 0 to FFH (Hex) [2]	0
[1] Only for the end delimiter end criterion. [2] Depending on whether you configure 7 or 8 data bits for the character frame.			

Further Information

A detailed description of data reception with the ASCII driver can be found in section Receiving Data with the ASCII Driver (Page 1592).

Data flow control and secondary signals (S7-300, S7-400)

Introduction

The communication module enables point-to-point communication with SIMATIC modules and with third-party products.

Supported interface functions

Different driver functions can be used depending on the interface used:

Table 1-111 Functions of the communication module based on the interface employed

	RS232C	20mA TTY	X27 (RS422/485)	
			RS422*	RS485*
Function CP 441				
3964(R) procedure	●	●	●	-
Computer connection RK512	●	●	●	-

	RS232C	20mA TTY	X27 (RS422/485)	
			RS422*	RS485*
ASCII driver:	●	●	●	●
• Operation of the RS232C accompanying signals	●	-	-	-
• Controlling/reading the RS232C accompanying signals with instructions	●	-	-	-
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-
Printer driver:	●	●	●	●
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-
Function CP 440				
3964(R) procedure			●	-
ASCII driver:			●	●
• XON/XOFF data flow control			●	-
Function CP 341				
3964(R) procedure	●	●	●	-
Computer connection RK512	●	●	●	-
ASCII driver:	●	●	●	●
• Operation of the RS232C accompanying signals	●	-	-	-
• Controlling/reading the RS232C accompanying signals with instructions	●	-	-	-
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-
Printer driver:	●	●	●	●
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-
Function CP 340				
3964(R) procedure	●	●	●	-
ASCII driver:	●	●	●	●
• Operation of the RS232C accompanying signals	●	-	-	-
• Controlling/reading the RS232C accompanying signals with instructions	●	-	-	-
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-
Printer driver:	●	●	●	●
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-

Function ET 200S 1SI				
3964(R) procedure	●		●	-
ASCII driver:	●		●	●

• Operation of the RS232C accompanying signals	•		-	-
• Controlling/reading the RS232C accompanying signals with instructions	•		-	-
• RTS/CTS data flow control	•		-	-
• XON/XOFF data flow control	•		•	-
* The difference between RS422 and RS485 is defined by configuration.				

Application options for the communication modules

Depending on the communication module used, you can establish a point-to-point connection with various Siemens modules and with third-party products:

- SIMATIC S5 through 3964(R) driver or RK512 with corresponding interface module at S5 end
- Siemens BDE-terminals ES 2-family through 3964(R) driver
- MOBY I (ASM 420/421, SIM), MOBY L (ASM 520) and capture station ES 030K through 3964R-driver
- PCs via procedure 3964(R) (development tools exist for programming on the PC: PRODAVE DOS 64R (6ES5 897-2UD11) for MS-DOS, PRODAVE WIN 64R (6ES5 897-2VD01) for Windows or the ASCII driver)
- Bar code reader via 3964(R) driver or ASCII driver
- PLC from other manufacturers via 3964(R), ASCII driver or RK512
- other devices with simple protocol structures via corresponding protocol adjustment with the ASCII driver
- Additional devices that also feature 3964(R) drivers or RK512
- Printer (HP-Deskjet, HP-Laserjet, post script, Epson, IBM)

Interface operating modes (S7-300, S7-400)

Possible interface operating modes

The following table is a summary of the interface operating modes from the various communication modules and protocols.

Operating mode	Description
Half-duplex (RS485) two-wire mode.	Operating mode for point-to-point connection or multipoint connection (multipoint) in two-wire mode. The communication module can be the master as well as the slave. During RS485-mode, the data transmission occurs through two lines (two-wire mode). The two lines are available alternatively for send and receive directions. They can either send or receive (half-duplex). After a send process, it is automatically switched to receive (maximum switch-over time of 1 ms).
Full-duplex (RS422), four-wire mode	During RS422-mode, the data transmission occurs through four lines (four-wire mode). There are always two lines available for send and receive directions. They can send and receive simultaneously (full-duplex). The data is exchanged simultaneously between one or more communication partners, i.e., it can be sent and received at the same time. Each communication partner must be able to simultaneously operate a transmission and reception set-up.
Full-duplex (RS 422) four-wire mode (point-to-point connection) [1)]	The communication module can be used in the following topologies in RS422 mode: • Link between two participants: Point-to-point connection • Connection between numerous stations: Multipoint connection The communication module can be used as both master and slave.

Operating mode	Description
Full-duplex (RS422) four-wire mode (multipoint master) [1]	The communication module can be used in the following topologies in RS422 mode: • Link between two participants: Point-to-point connection • Connection between numerous stations: Multipoint connection The communication module can be used as a multipoint master as well as a multipoint slave. In the case of a master/slave topology in the RS422 mode: • The master's sender is interconnected with the receivers of all the slaves. • The slaves' senders are interconnected with the master's receiver. • Only the receiver of the master and the receiver of one slave have a default setting. All the other slaves function without default settings.
Full-duplex (RS422) four-wire mode (multipoint slave) [1]	The communication module can be used in the following topologies in RS422 mode: • Link between two participants: Point-to-point connection • Connection between numerous stations: Multipoint connection The communication module can be used as a multipoint master as well as a multipoint slave. In the case of a master/slave topology in the RS422 mode: • The master's sender is interconnected with the receivers of all the slaves. • The slaves' senders are interconnected with the master's receiver. • Only the receiver of the master and the receiver of one slave have a default setting. All the other slaves function without default settings.

[1] CP 440 only

Initial state of receive line (S7-300, S7-400)

X27 (RS422/485) interface

The table below contains descriptions of the parameters for the X27 (RS422/485) interface submodule. RS485-mode is not possible with the 3964(R) and RK512 procedures and the printer.

Table 1-112 X27 (RS422/485) interface module

Parameters	Description	Value range	Default value
Initial state of receive line	**none**: This setting only makes sense with bus-capable special drivers.	None	Signal R(A) 5V, signal R(B) 0V (break detection) [2]
	Signal R(A) 5V, signal R(B) 0V (break detection): Break detection is possible in "Full-duplex (RS 422) four-wire mode" with this setting.	Signal R(A) 5V, signal R(B) 0V (break detection) [1]	
	Signal R(A) 0V, signal R(B) 5V: This initial state corresponds to idle (no senders active) in "Half Duplex (RS 485) Two-Wire Mode". No break detection is possible with this setting.	Signal R(A) 0V, signal R(B) 5V	

[1] Only with "Full-duplex (RS422) four-wire mode"

[2] Only with "Full-duplex (RS422) four-wire mode", with "Half-duplex (RS485) two-wire mode" the default setting is signal R (A) 0V, signal R (B) 5V

Initial state of receive line

The figure illustrates the wiring of the receiver at the X27 (RS422/485) interface:

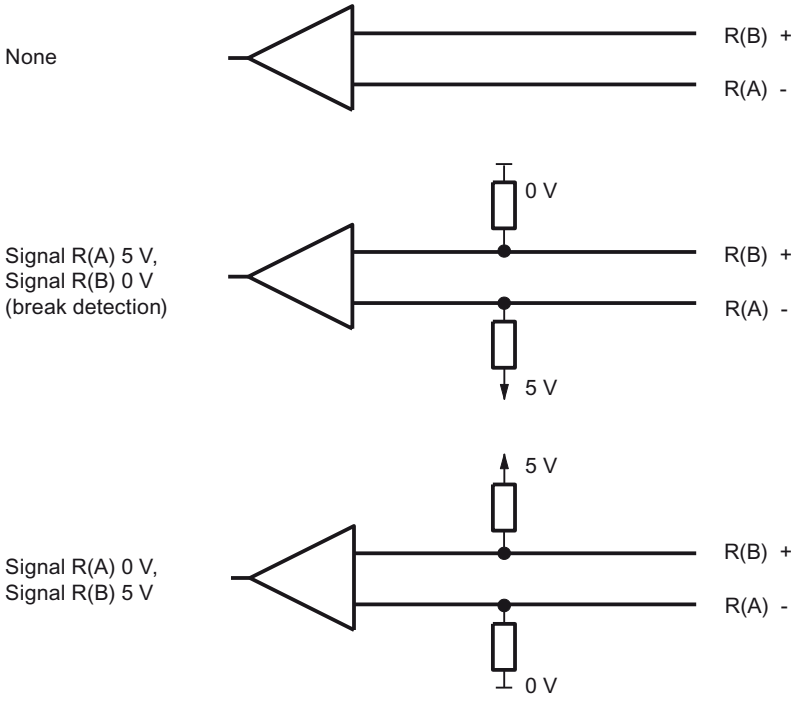

Figure 1-64 Wiring of the receiver at the X27 (RS422/485) interface

CP 440 in a multipoint topology with the RS422/485 protocols (S7-300, S7-400)

Application options

The CP 440 can be used in different topologies in the RS422 and RS485 operating modes.

Distinctions are drawn between connections with:

- Two nodes (**point-to-point**) and
- Several nodes (**multipoint**)

In these cases the CP 440 can be used as:

- **Master** or
- **Slave**.

RS422 or RS485 mode for point-to-point connections

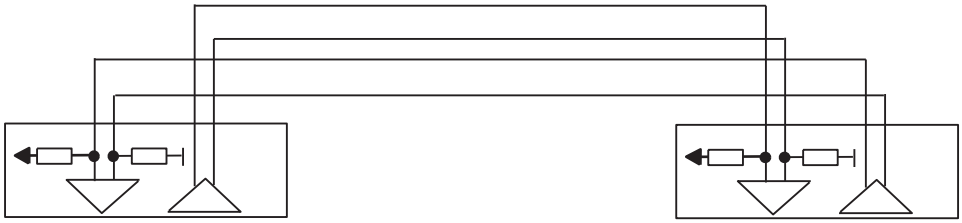

Figure 1-65 RS422 point-to-point

Figure 1-66 RS485 point-to-point

In the case of a **master/slave topology**, there must be an appropriate message frame in the user program. Example: The master sends all the slaves a message frame with address information. All the slaves listen in and compare the address with their own. If the address is the same, the addressed slave sends its answer.

The senders of all slaves must be able to switch to low impedance.

RS422 mode in multipoint connections

In the case of a master/slave topology in RS422 mode:

- The master's sender is interconnected with the receivers of all the slaves.
- The slaves' senders are interconnected with the master's receiver.
- Only the receiver of the master and the receiver of one slave have a default setting. All the other slaves function without default settings.

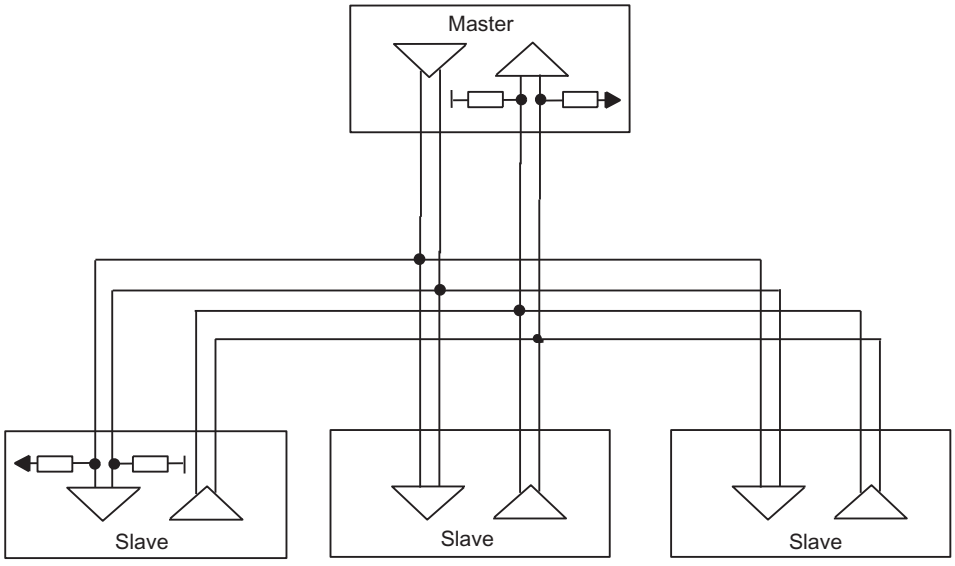

Figure 1-67 RS422 multipoint

RS485 mode in multipoint connections

In the case of a topology in RS485 operation:

- The cable pair is interconnected for the send/receive line of all the nodes.
- Only the receiver of a node has a default setting. All the other modules function without default settings.

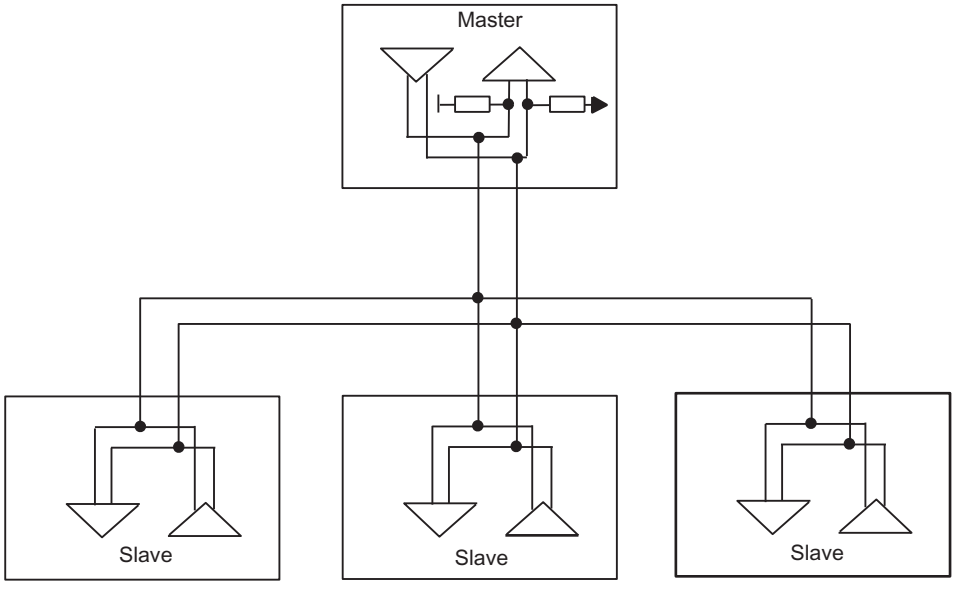

Figure 1-68 RS485 multipoint

The necessary settings are made for the different topologies in the "Interface" section of the properties dialog for the module.

Note

When you run the ASCII driver in RS422 multipoint or RS485 mode, you must take steps in the user program to ensure that only one node sends data at any one time. If two users send data simultaneously, the message frame is corrupted.

Receive buffer (S7-300, S7-400)

Receive buffer on the module

The following table describes the parameters for the module receive buffer.

Table 1-113 Receive buffer from the module - procedure 3964(R) / RK512 / ASCII driver

Parameters	Description	Value range	Default value
Clear module receive buffer on startup	The module receive buffer is not deleted at CPU start-up (STOP RUN transition).	No (fixed)	No
Use CPU receive mailbox (CP 441 only)	Here you can specify whether a receive mailbox is to be set up on the CPU. You must set up a receive mailbox if you have not programmed a BRCV instruction for the module in the user program of the CPU. If you have programmed a BRCV, you must deactivate this parameter, otherwise data will be stored in the receive mailbox defined here instead of being processed by the BRCV.	• Yes • No	No
DB number (1)	Number of the data block for the receive mailbox on the CPU.	1 to 65535 (depending on the CPU)	1
(1) Only when "Use CPU Receive Mailbox" = "Yes" (2) Only when "Buffered receive message frames" = "1"			

Table 1-114 Receive buffer from the module - additional for ASCII drivers

Parameters	Description	Value range	Default value
Buffered receive message frames	Here you can specify the number of receive message frames to be buffered in the receive buffer of the communication module. If you specify "1" here and deactivate the following parameter "prevent overwrite" and cyclically read the received data from the user program, a current message frame will always be sent to the CPU.	1 to 250	250
Dynamic message frame ET 200S 1SI	For the reception of messages, you can specify if a message should just be buffered or if the messages should be dynamically buffered. By activating dynamic message frames, the module can buffer multiple messages with different lengths. The buffer is a ring buffer. If the buffer is full, the oldest message is overwritten, unless the "Prevent overwriting buffers" parameter is enabled. In this case, the newest message is discarded. In both cases, a diagnosis alarm will display saying that data was lost.	• Activated • Blocked	Activated
Prevent overwrite	You can deactivate this parameter if the parameter "buffered receive message frames" is set to "1". This authorizes the buffered receive message frame to be overwritten.	• Yes • no (2)	Yes
(1) Only when "Use CPU Receive Mailbox" = "Yes" (2) Only when "Buffered receive message frames" = "1"			

Modbus communication (S7-300, S7-400)

Overview of modbus communication (S7-300, S7-400)

Position in the system environment

The following Modbus description refers to the use of the reloadable driver on the CP 341 (S7 300) and CP 441-2 (S7 400) communication processors, and the operation of the ET 200S 1SI module in the Modbus/USS model.

Function of the coupling

With the appropriately equipped communication modules and the related instructions, you can establish a communication connection between a remote Modbus control system (such as Modicon PLCs or Honeywell TDC3000) and a SIMATIC S7.

The GOULD-MODBUS protocol in the RTU format is used for transmission.

Function codes 01, 02, 03, 04, 05, 06, 08, 15 and 16 are used for communication between a communication module operated as a Modbus slave and a master system.

If a SIMATIC S7 communication module is operated as a Modbus master, function codes 07, 11 and 12 are additionally available.

Modbus protocol

The modbus protocol is a communication protocol based on a master/slave and client/server architecture.

The procedure used is a code-transparent, asynchronous half-duplex procedure. Data transmission is carried out without handshake.

SIMATIC S7 as a Modbus slave

The master has the initiative for transmission, the CP / S7 CPU works as a slave.

Frame traffic from slave to slave is not possible.

SIMATIC S7 as a Modbus master

As master, the communication module initiates transmission and, after outputting a request message frame, it waits for the configured response monitoring time for a response message frame from the slave.

Message frame structure

The data exchange "Master-Slave" and/or "Slave-Master" begins with the **slave address**, followed by the **function code**. Then the data are transferred. The structure of the data field depends on the function code used. The CRC check is transmitted at the end of the message frame.

ADDRESS	FUNCTION	DATA	CRC-CHECK
Byte	Byte	n byte	2 byte

ADDRESS	MODBUS slave address
FUNCTION	MODBUS function code
DATA	Message frame data: Byte_Count, Coil_Number, Data
CRC-CHECK	Message frame checksum

Slave address

The slave address can be within the range of 1 to 255. The address is used to address a defined slave on the bus.

Broadcast Message

The master uses slave address 0 to address all slaves on the bus.

Broadcast messages are only permitted in conjunction with writing **Function codes 05, 06, 15 and 16**.

A broadcast message is not followed by a response message frame from the slave.

Function code

The function code defines the meaning of the message frame. It also defines the structure of a message frame. The following function codes are supported by the communication module:

Function Code	Function in accordance with MODBUS specification
01	Read Coil Status
02	Read Input Status
03	Read Holding Registers
04	Read Input Registers
05	Force Single Coil
06	Preset Single Register
07	Read Exception Status (only master)
08	Loop Back Test
11	Fetch Communications Event Counter (only master)
12	Fetch Communications Event Log (only master)
15	Force Multiple Coils
16	Preset Multiple Registers

Data Field DATA

The data field DATA is used to transfer the function code-specific data such as:

- Bytecount, Coil_Startaddress, Register_Startaddress; Number_of_Coils, Number_of_Registers, ...
  See section "Function codes (Page 1651)".

CRC-Check

The end of the message frame is identified by means of the CRC 16 checksum consisting of 2 bytes. It is calculated by the following polynominal: $x^{16} + x^{15} + x^2 + 1$.

The low byte is transferred first, followed by the high byte.

End of Message Frame

The loadable driver recognizes the end of the message frame when no transmission takes place during the time period required for the transmission of three and a half characters (3.5 times character delay time) (see MODBUS Protocol Reference Guide).

This message frame end TIME_OUT is thus dependent on the data transmission rate.

Data transmission rate	TIME_OUT
115200 bps	0,3 ms
76800 bps	0,5 ms
57600 bps	0.7 ms
38400 bps	1 ms
19200 bps	2 ms
9600 bps	4 ms

Data transmission rate	TIME_OUT
4800 bps	8 ms
2400 bps	16 ms
1200 bps	32 ms
600 bps	64 ms
300 bps	128 ms

During "Normal operation" the Modbus message frame received by the connection partner is evaluated and checked after the end of frame for TIME_OUTs is received.

During "Interference suppression" the end of frame is recognized by correctly formatted receive frame with a correct CRC code.

Exception responses

On recognition of an error in the request message frame from the master, for example, register address illegal, the slave sets the highest value bit in the function code of the response message frame.

This is followed by transmission of one byte of error code, exception code, which describes the reason for the error.

A detailed description of the meaning of the above-mentioned parameters can be found in the “GOULD MODICON Modbus Protocol.”

Exception code message frame

The error code response message frame from the slave has the following structure:

- for example, slave address 5, function code 5, exception code 2

Response frame from the slave EXCEPTION_CODE_xx:

05H	Slave address
85H	Function code
02H	Exception code (1...7)
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

On receipt of an error code response message frame by the driver, the current job is completed with error.

An error number corresponding to the received error code (exception code 1-7) is also entered in the error message area.

There is no entry in the target data block.

The following error codes are defined in accordance with the MODBUS Specification:

Error code	Meaning in accordance with MODBUS Specification	Cause - Short Description *
1	Illegal function	Illegal function code
2	Illegal data address	Slave has illegal data address

Error code	Meaning in accordance with MODBUS Specification	Cause - Short Description *
3	Illegal data value	Slave has illegal data value
4	Failure in Associated Device	Slave has internal error
5	Acknowledge	Function is carried out
6	Busy, Rejected message	Slave is not ready to receive
7	Negative acknowledgement	The function cannot be carried out.
* Check slave for further details.		

RS232C accompanying signals

The following RS232C accompanying signals are available on the communication modules that use the RS232C interface module:

DCD	(input)	Data carrier detect	Data carrier detected
DTR	(Output)	Data terminal ready	Communication module ready
DSR	(input)	Data set ready	Communication partner ready for operation
RTS	(Output)	Request to send	Communication module ready to send
CTS	(input)	Clear to send	Communication partner can receive data from the communication module (response to RTS = ON of the CP)
RI	(input)	Ring indicator	Ring indicator

When the communication module is switched on, the output signals are in the OFF state (inactive).

You can configure the operation of the DTR/DSR and RTS/CTS control signals in the properties dialog of the module or via instructions in the user program.

Operating RS232C accompanying signals

The RS232C accompanying signals can be used as follows:

- When the automatic use of all RS232C accompanying signals is configured.
- Via the corresponding instructions. For more information, see the section "Overview of the instructions (Page 1579)".

Note

When automatic operation of the RS232C accompanying signals is configured, RTS and DTR cannot be controlled via the corresponding instruction (see section "Overview of the instructions (Page 1579)")!

On the other hand, it is always possible to read all RS232C accompanying signals using the corresponding instruction.

The following sections describe the basic procedures for control and evaluation of the RS232C accompanying signals.

Automatic use of accompanying signals

Automatic operation of the RS232C accompanying signals on the communication module is realized as follows:

- As soon as the communication module is switched by configuration to an operating mode with automatic operation of the RS232C accompanying signals, it sets the RTS line to OFF and the DTR line to ON (communication module ready for use).
- This prevents sending and receiving of message frames until the DTR line is set to ON. As long as DTR remains set to OFF, no data is received via the RS232C interface. If a send job is made, it is aborted with a corresponding error message.
- When a send job is made, RTS is set to ON and the parameterized data output waiting time starts. When the data output time elapses and CTS = ON, the data is sent via the RS232C interface.
- If the CTS line is not set to ON within the data output time so that data can be sent, or if CTS changes to OFF during transmission, the send job is aborted and an error message generated.
- Once the data has been sent and the configured clear RTS time has elapsed, the RTS line is set to OFF. The CP does not wait for CTS to change to OFF.
- Data can be received via the RS232C interface as soon as the DSR line is set to ON. If the receive buffer of the communication module threatens to overflow, there is no response from the communication module.
- If DSR changes from ON to OFF, an active send job as well as the receipt of data will be canceled with an error message. The message "DSR = OFF (automatic operation of V24 signals)" is entered in the diagnostics buffer of the communication module.

Note

When automatic operation of the RS232C accompanying signals is configured, RTS and DTR cannot be controlled via the corresponding instruction! For more information, see the section "Overview of the instructions (Page 1579)".

Note

Set "Clear RTS time" in the properties dialog of the module so that the communication partner can receive the last characters of the message frame in their entirety before RTS, and thus the send job, is revoked. The "data output waiting time" must be set such that the communication partner can be ready to receive before the time elapses.

Time Diagram

The following figure illustrates the chronological sequence of a send job.

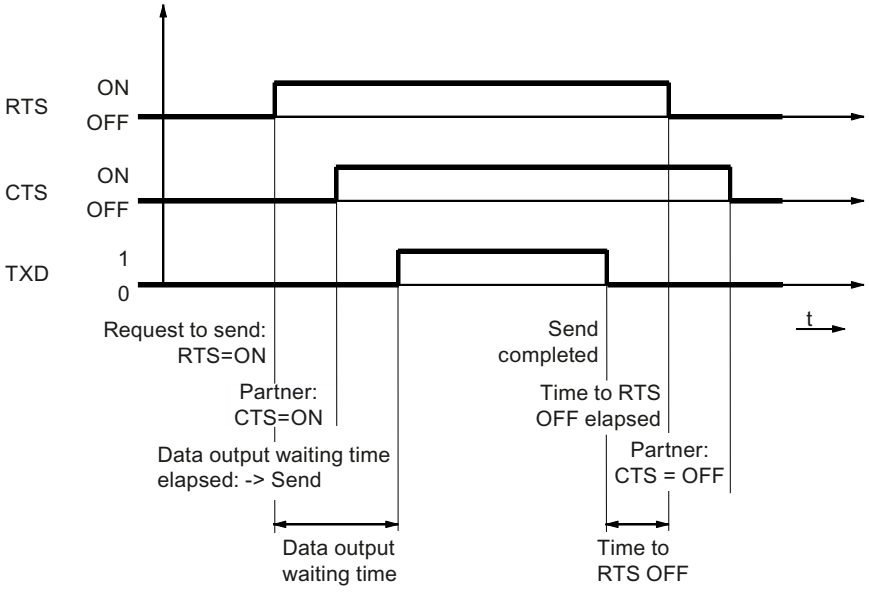

Figure 1-69 Time diagram for automatic operation of the RS232C accompanying signals

Modbus master (S7-300, S7-400)

CP 341 (S7-300, S7-400)

Data transmission from CPU to communication module with P_SND_RK (CP 341) (S7-300, S7-400)

Activation

Execution of a MODBUS function code is activated by means of the instruction **P_SND_RK** with an **edge** at the **REQ** input.

Enter ‘S’ for SEND at the SF parameter.

The logical module address is entered at LADDR.

You must enter ‘X’ for expanded data block as the area type of the partner CPU. No values must be specified for the other parameters of the partner CPU (R_...).

This ensures transfer to the driver of the parameters required for the execution of the function code.

Data source

When P_SND_RK is activated, the **source data area** specified with the **DB_NO** and **DBB_NO** parameters is transferred to the communication module with the length **LEN**.

Length Indication

The length **LEN** depends on the function code used.

Function code	Length LEN in bytes
01	6
02	6
03	6
04	6
05	6
06	6
07	2
08	6
11	2
12	2
15	>6
16	>6

If the transferred data quantities differ from those listed above for the individual function codes, the job is not carried out and P_SND_RK rejects it with an edge at output ERROR.

SEND Source DB

The parameters required for the execution of a function code must be entered as user data in the source data area.

A detailed description of each P_SND_RK source DB is available in the description of the individual function codes in the section "Function codes (Page 1651)".

Generation of Message Frames

The request message frames to the slave are generated in accordance with the transferred P_SND_RK source data and sent by the communication module.

First of all the driver checks if the length LEN specified at P_SND_RK corresponds to the length for this function code.

If not, the job is not carried out and it is completed with an edge at output ERROR of the P_SND_RK.

When using function codes other than those listed, the activated job is not carried out either and is completed with ERROR at P_SND_RK.

The “byte counter” and “CRC check” elements in the request message frame are generated by the communication module; an entry in the P_SND_RK source DB is not required.

Job Completion for Writing Functions

For writing function codes, the activated P_SND_RK is completed after a response message frame is received without error. This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the P_SND_RK.

If **errors** are detected in the message frame traffic, or if the slave sends an **error code** response message frame, this is reported by an edge at output **ERROR**.

Job Completion for Reading Functions

For reading functions, the activated P_SND_RK is completed after the response message frame is received without error **and** complete transfer of the received data to the CPU.

This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the P_SND_RK.

At this time the received data are already available in the CPU.

If **errors** are detected in the message frame traffic, or if the slave sends an **error code** response message frame, this is reported by an edge at output **ERROR**.

In this case no receive data are transferred to the CPU.

STATUS Entry on Job Completion

For those instances when a job is completed with **ERROR** at P_SND_RK, an additional error code is entered in the **STATUS** parameter.

The exact cause for the error can be determined with this error code

Data transmission from the communication module to the CPU with P_RCV_RK (CP 341) (S7-300, S7-400)

Prerequisite

All **reading** function codes require a P_RCV_RK.

Data destination

When the instruction P_RCV_RK is ready to receive, it accepts the received data from the communication module and enters it into the data destination specified in the parameters **DB_N0** and **DBB_N0**.

How Receipt of Data is Displayed

The user is informed of the receipt of data in the CPU by means of an edge at output **NDR**.

At this point the length of the received data block is displayed in the parameter **LEN**.

Completion of the entire modbus job can be recognized at output DONE of the instruction P_SND_RK.

How to Handle an Error

In the event of receive or transfer errors, no data is transferred to the CPU. In this case, the instruction P_SND_RK is completed with an edge at the output ERROR.

P_RCV_RK Destination DB

The user data received with a reading function code are entered into the P_RCV_RK destination area.

A detailed description of each P_RCV_RK destination DB can be found in section " Function codes (Page 1651)".

The length of data entered is displayed an the parameter LEN of P_RCV_RK.

CP 441-2 (S7-300, S7-400)

Data transmission from CPU to communication module with BSEND (CP 441-2) (S7-300, S7-400)

Communication connection

The **Parameter ID** describes the unique communication connection to a communication partner. You must specify the local ID from the data link configuration here

Block Relationship

The **Parameter R_ID** describes the unique block relationship within a communication connection.

With this driver, **any** values from **0..255** may be entered for R_ID on BSEND.

In the event of reading jobs, parameter assignment of the associated BRCV must have the same R_ID as BSEND.

Activation

Execution of a MODBUS function code is activated by means of the instruction **BSEND** with an **edge** at the **REQ** input.

This ensures transfer to the driver of the parameters required for the execution of the function code.

Data source

When BSEND is activated, the **source data area** specified with parameter **SD_1** is transferred to the communication module with the length **LEN**.

Length Indication

The length **LEN** depends on the function code used.

Function code	Length LEN in bytes
01	6
02	6
03	6
04	6
05	6
06	6
07	2
08	6
11	2
12	2
15	>6
16	>6

If the transferred data quantities differ from the ones listed above for the individual function codes, the job is not carried out and BSEND rejects it with an edge at output ERROR.

BSEND Source DB

The parameters required for the execution of a function code must be entered as user data in the source data area.

A detailed description of each BSEND source DB is available in the description of the individual function codes in section "Function codes (Page 1651)".

Generation of Message Frames

The request message frames to the slave are generated in accordance with the transferred BSEND source data and sent by the communication module.

First of all the driver checks if the length LEN specified at BSEND corresponds to the length for this function code.

If not, the job is not carried out and it is completed with an edge at output ERROR of BSEND.

When using function codes other than those listed, the activated job is not carried out either and is completed with ERROR at BSEND.

The “byte counter” and “CRC check” elements in the request message frame are generated by the communication module; an entry in the BSEND source DB is not required.

Job Completion for Writing Functions

For writing function codes, the activated BSEND is completed after a response message frame is received without error.

This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the BSEND.

If **errors** are detected in the message frame traffic, or if the slave sends an **error code** response message frame, this is reported by an edge at output **ERROR**.

Job Completion for Reading Functions

For reading functions, the activated BSEND is completed after the response message frame is received without error **and** complete transfer of the received data to the CPU.

This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the BSEND.

At this time the received data are already available in the CPU.

If **errors** are recognized in the data frame traffic, or if the slave sends an **error code** response message frame, this is reported by an edge at output **ERROR**.

In this case no receive data are transferred to the CPU.

Entry in the error message area at job completion

For those instances when a job is completed with **ERROR** at BSEND, an additional error code is entered in the error message area.

The exact cause for the error can be determined with this error code.

Data transmission from the communication module to the CPU with BRCV (CP 441-2) (S7-300, S7-400)

Communication connection

The **Parameter ID** describes the unique communication connection to a communication partner. You must specify the local ID from the data link configuration here.

Block Relationship

The **Parameter R_ID** describes the unique block relationship within a communication connection.

All **reading** function codes require a BRCV.
Parameter assignment of **R_ID** on **BRCV** must be the **same R_ID** as for the corresponding BSEND, which was used to activate this job (any values 0 to 255).

In this way you can program several BSEND / BRCV pairs in the SIMATIC user program.

The response message frames received from the Modbus slave are then stored in different destination areas, depending on the R_ID used for this job.

Data destination

When the BRCV instruction is ready to receive, it accepts the received data from the communication module and enters them in the data destination specified in the parameter **RD_1**. This means that the data destination is variable.

How Receipt of Data is Displayed

The user is informed of the receipt of data in the CPU by means of an edge at output **NDR**.

At this point the length of the received data block is displayed in the parameter **LEN**.

You can recognized that the entire Modbus job is completed at the DONE output of the BSEND instruction.

How to Handle an Error

In the event of receive or transfer errors, no data is transferred to the CPU. In this instance BSEND is completed with an edge at the output ERROR.

BRCV Destination DB

The user data received with a reading function code are entered into the BRCV destination area.

The detailed structure of the respective BRCV destination DB is provided in the description of the individual function codes in section "Function codes (Page 1651)".

The length of data entered is displayed at the parameter LEN of BRCV.

ET 200S 1SI (S7-300, S7-400)

Data transmission with the ET 200S 1SI Modbus master (S7-300, S7-400)

Introduction

Data transmission between the module and CPU using the S_SEND and S_RCV instructions.

S_SEND instruction: Send data to a communication partner

The S_SEND and S_RCV instructions must be activated in order to execute a Modbus master job. The S_SEND instruction is activated by an edge at the REQ input when data is to be output to the module. The S_RCV instruction is made ready to receive data from the module with EN_R=1. An S_RCV is required for all reading function codes. The figure below shows the overall behavior of the S_SEND and S_RCV parameters when a Modbus job is executed.

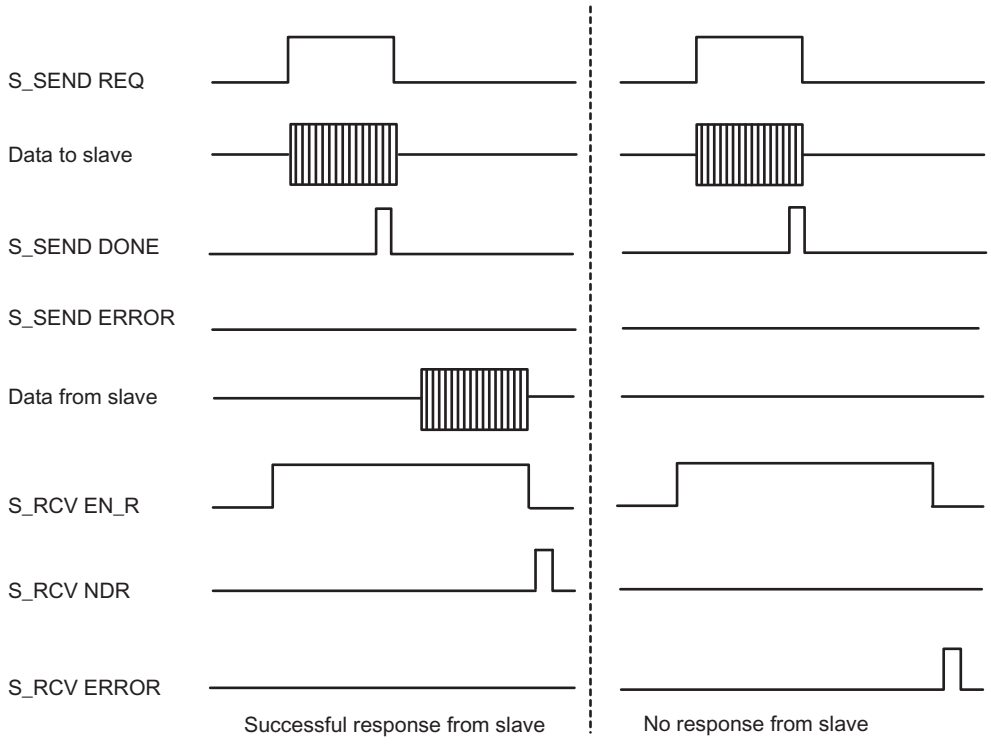

Figure 1-70 Time Sequence Chart for Modbus Request

Data transmission is initiated by a positive edge at the REQ input. Data transmission can take several calls (program cycles), depending on the data volume.

The S_SEND instruction can be called cyclically by setting the signal state at parameter input R to "1". This cancels transmission to the module and resets the S_SEND instruction to its initial state. Data that has already been received by the module is still sent to the communication partner. If the signal state remains static at "1" at the input R, sending is disabled.

The address of the ET 200S Modbus/USS serial interface module to be addressed is specified at the LADDR parameter.

The DONE output indicates "Job completed without errors". ERROR indicates error events. If an error has occurred, the corresponding event number is displayed in STATUS. If no error occurs, STATUS has the value 0. DONE and ERROR/STATUS are also output at RESET of S_SEND. The binary result is reset if an error has occurred. If the block is completed without errors, the binary result has the status "1".

Modbus master read job

As the interface between the user program and the interface module operates in half-duplex mode you must observe the following items:

After positive acknowledgment of the Modbus master read job, you first have to fetch the receive data from the interface module by calling the S_RCV instruction before starting a new Modbus master send job.

Modbus slave (S7-300, S7-400)

Initialization of the communication module (S7-300, S7-400)

Introduction

The Modbus communication instructions ,(MODB_341, MODB_441 and S_MODB) for the loadable Modbus slave driver must be called in the cyclic part of the SIMATIC S7 CPU.

The Modbus communication instruction initializes the CP and executes the Modbus functions that the driver cannot perform independently. The Modbus communication instruction must then also be called in the user program if these function codes are not used by the Modbus master system.

Communication between the CP and the instruction is performed through the CPU operating system functions and instructions BSEND (CP 441-2) or P_SND_RK and P_RCV_RK (CP 341), which are called by the instruction.

Startup, initialization

The Modbus communication instruction must be initialized after a warm restart or hot restart of the CPU.

Initialization is triggered by a rising edge at the CP_START input.

The instruction first deletes the instance DB, then it reads the address areas I, O, M, T, C from the CPU and saves them in the instance DB. This allows the write requests from the Modbus master system to be tested to determine if they exceed the range.

With a send request, the CP is informed of the number of the instance DB and the previously successful initialization.

If the initialization ends with errors, no Modbus communication is possible. All requests from the Modbus master system are answered with an exception code message frame.

Instance DB

All data relating to the Modbus communication instruction are in an instance data block. This data block is both an instance DB (multi-instances) for the various instructions used, and a work area for the Modbus communication instruction. An additional data area is not required.

The Modbus communication instruction only works with the instance DB and with local data.

The instance DB is read-only.

Monitoring time

After power on, the CP needs several seconds for the hardware and memory test until it is ready. Initialization attempts made by the Modbus communication instruction during this time end with an error. Therefore, the initialization job is repeated several times during the monitoring period.

Read time interval SYSTAT (CP 441-2 only)

Since a regular SYSTAT read each cycle or every second cycle put an unnecessarily load on the CP and K bus and thereby reduce the data throughput, a time interval can be set for reading the SYSTAT (only relevant for CP 441-2 (MODB_441)).

Function codes (S7-300, S7-400)

Modbus master (RTU) (S7-300, S7-400)

Function Code 01 – Read Output Status (S7-300, S7-400)

Function

This function enables individual bits to be read from the slave.

Start address

The **Bit start address** parameter is not checked by the driver and is sent unchanged.

Bit number

For **Bit number**, number of coils, any value between **1** and **2040** is permitted (for ET 200S 1SI Modbus up to 2008).

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#1	Function code
+2.0	bit_startadr	WORD	W#16#0040	Bit start address
+4.0	bit_number	INT	16	Bit number

Example

Request message frame FUNCTION 01:

05H	Slave address
01H	Function code
00H	Bit start address "High"
40H	Bit start address "Low"
00H	Bit number "High"
10H	Bit number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 01:

05H	Slave address
01H	Function code
02H	Byte counter
01H	<Data>
17H	<Data>
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#1701	Data

The driver enters the data of the response message frame into the destination DB **word by word**.

The 1st received byte is entered as the Low Byte of the 1st word “data[1],” the 3rd received byte as the Low Byte of the 2nd word “data[2],” etc.

If fewer than 9 bits or if only one Low Byte was read, the value **00H** is entered into the remaining High Byte of the last word.

Function code 02 - Read Input Status (S7-300, S7-400)

Function

This function enables individual bits to be read from the slave.

Start address

The **Bit start address** parameter is not checked by the driver and is sent unchanged.

Bit number

For **Bit number**, number of coils, any value between **1** and **2040** is permitted (for ET 200S 1SI Modbus up to 2008).

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#2	Function code
+2.0	bit_startadr	WORD	W#16#0120	Bit start address
+4.0	bit_number	INT	24	Bit number

Example

Request message frame FUNCTION 02:

05H	Slave address
02H	Function code
01H	Bit start address "High"
20H	Bit start address "Low"
00H	Bit number "High"
18H	Bit number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 02:

05H	Slave address
02H	Function code
03H	Byte counter
04H	<Data>
26H	<Data>
48H	<Data>
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#2604	Data
+2.0	data[2]	WORD	W#16#0048	Data

The driver enters the data of the response message frame into the destination DB **word by word**.

The 1st received byte is entered as the Low Byte of the 1st word “data[1],” the 3rd received byte as the Low Byte of the 2nd word “data[2],” etc.

If fewer than 9 bits or if only one Low Byte was read, the value **00H** is entered into the remaining High Byte of the last word.

Function Code 03 - Read Output Registers (S7-300, S7-400)

Function

This function enables you to read individual registers from the slave.

Start address

The **Register start address** parameter is not checked by the driver and is sent unchanged.

Register number

1 Up to a **maximum of 127 registers** (for ET 200S 1SI Modbus max. 125 registers) can be read (1 register = two bytes).

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#3	Function code
+2.0	reg_startadr	WORD	W#16#0040	Register start address
+4.0	reg_number	INT	2	Register number

Example

Request message frame FUNCTION 03:

05H	Slave address
03H	Function code
00H	Register start address "High"
40H	Register start address "Low"
00H	Register number "High"
02H	Register number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 03:

05H	Slave address
03H	Function code
04H	Byte counter
21H	Register address 40H data "High"
23H	Register address 40H data "Low"
25H	Register address 41H data "High"
27H	Register address 41H data "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#2123	Data
+2.0	data[2]	WORD	W#16#2527	Data

Function code 04 - Read Input Registers (S7-300, S7-400)

Function

This function enables you to read individual registers from the slave.

Start address

The **Register start address** parameter is not checked by the driver and is sent unchanged.

Register number

1 Up to a **maximum of 127 registers** (for ET 200S 1SI Modbus max. 125 registers) can be read (1 register = two bytes).

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#4	Function code
+2.0	reg_startadr	WORD	W#16#0050	Register start address
+4.0	reg_number	INT	3	Register number

Example

Request message frame FUNCTION 04:

05H	Slave address
04H	Function code
00H	Register start address "High"
50H	Register start address "Low"
00H	Register number "High"
03H	Register number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 04:

05H	Slave address
04H	Function code
04H	Byte counter
31H	Register address 50H data "High"
32H	Register address 50H data "Low"
33H	Register address 51H data "High"
34H	Register address 51H data "Low"
35H	Register address 52H data "High"
36H	Register address 52H data "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#3132	Data
+2.0	data[2]	WORD	W#16#3334	Data
+4.0	data[3]	WORD	W#16#3536	Data

Function Code 05 - Force Single Coil (S7-300, S7-400)

Function

This function serves to set or delete individual bits in the slave.

Bit address

The **bit address** parameter is not checked by the driver and is sent unchanged.

Bit Status

The following two values are permitted as the **bit status**:

- FF00H = Set bit
- 0000H = Delete bit

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#5	Function code
+2.0	bit_address	WORD	W#16#0019	Bit address
+4.0	bit_state	WORD	W#16#FF00	Bit Status

Example

Request message frame FUNCTION 05:

05H	Slave address
05H	Function code
00H	Bit address "High"
19H	Bit address "Low"
FFH	Set bit
00H	

xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 05:

05H	Slave address
05H	Function code
00H	Bit address "High"
19H	Bit address "Low"
FFH	Bit status "High"
00H	Bit status "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

The response message frame is not entered in the receive DB.

Function code 06 - Preset Single Register (S7-300, S7-400)

Function

This command enables a slave register to be overwritten with a new value.

Register address

The **Register address** parameter is not checked by the driver and is sent unchanged.

Register value

Any value can be used as the **Register value**.

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#6	Function code
+2.0	reg_address	WORD	W#16#0180	Register address
+4.0	reg_value	WORD	W#16#3E7F	Register value

Example

Request message frame FUNCTION 06:

05H	Slave address
06H	Function code
01H	Register address "High"
80H	Register address "Low"
3EH	Register value "High"
7FH	Register value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 06:

05H	Slave address
06H	Function code
01H	Register address "High"
80H	Register address "Low"
3EH	Register value "High"
7FH	Register value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

The response message frame is not entered in the receive DB.

Function Code 07 - Read Exception Status (S7-300, S7-400)

Function

This function code enables 8 event bits to be read from the connected slave.

The start bit number of the event bit is determined by the connected device and therefore does not have to be specified by the SIMATIC user program

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#7	Function code

Example

Request message frame FUNCTION 07:

05H	Slave address
07H	Function code
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 07:

05H	Slave address
07H	Function code
3EH	<Data>
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#3Exx	Data

The driver enters the individual bytes of the response message frame in the **High Byte** in the destination DB data[1].

The low byte of data[1] remains unchanged.

A value of 1 is displayed as the length in the LEN parameter of the receive instruction.

Function code 08 - Loop back Diagnostic Test (S7-300, S7-400)

Function

This function is used to check the communications connection.

Only **Diagnostic code 0000** is supported with this function code!

Diagnostic Code

The only permissible value for the parameter **Diagnostic code** is 0000.

Test Value

Any value can be used as the **Test Value**.

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#8	Function code
+2.0	diag_code	WORD	W#16#0000	Diagnostic Code
+4.0	test_value	WORD	W#16#A5C3	Test Value

Example

Request message frame FUNCTION 08:

05H	Slave address
08H	Function code
00H	Diagnostic code "High"
00H	Diagnostic code "Low"
A5H	Test value "High"
C3H	Test value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 08:

05H	Slave address
08H	Function code
00H	Diagnostic code "High"
00H	Diagnostic code "Low"
A5H	Test value "High"
C3H	Test value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

The slave must return the request message frame received from the master unchanged as an echo.

The response message frame is not entered into an RCV DB.

Function code 11 - Fetch Communications Event Counter (S7-300, S7-400)

Function

This function code is used to read a **"Status Word"** (2 bytes long) and an **"Event Counter"** (2 bytes long) from the slave.

The meaning of the above parameters is described in detail in the "GOULD MODICON Modbus Protocol".

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#0B	Function code

Example

Request frame FUNCTION 11:

05H	Slave address
0BH	Function code
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 11:

05H	Slave address
0BH	Function code
FEH	Status Word "High"
DCH	Status Word "Low"
01H	Event Counter "High"
08H	Event Counter "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#FEDC	Status Word
+2.0	data[2]	WORD	W#16#0108	Event Counter

Note

Use with the ET 200S 1SI distributed I/O

Please note that when using with the ET 200S 1SI distributed I/O, the entire response frame of the slave, i.e. with Modbus address and function code (additional 2 bytes), is stored in the RCV destination DB. This causes the displacement of the offset for the status word (Offset 2.0) and for the event counter (Offset 4.0).

Function Code 12 - Fetch Communications Event Log (S7-300, S7-400)

Function

This function code enables the following to be read from the slave.

- 2 Byte **"Status Word"**
- 2 Byte **"Event Counter",**
- 2 Byte **"Message Counter"** and
- 64 Byte **"Event Bytes"**

The meaning of the above parameters is described in detail in the “GOULD MODICON Modbus Protocol.”

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#0C	Function code

Example

Request message frame FUNCTION 12:

05H	Slave address
0CH	Function code
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 12:

05H	Slave address
0CH	Function code
46H	Byte counter

87H	Status Word "High"
65H	Status Word "Low"
01H	Event Counter "High"
08H	Event Counter "Low"
02H	Message Counter "High"
20H	Message Counter "Low"
01H	Event Byte 1
12H	Event Byte 2
:	:
C2H	Event Byte 63
D3H	Event Byte 64
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#8765	Status Word
+2.0	data[2]	WORD	W#16#0108	Event Counter
+4.0	data[3]	WORD	W#16#0220	Message Counter
+6.0	bytedata[1]	BYTE	B#16#01	Event Byte 1
+7.0	bytedata[2]	BYTE	B#16#12	Event Byte 2
:	:			:
+68.0	bytedata[63]	BYTE	B#16#C2	Event Byte 63
+69.0	bytedata[64]	BYTE	B#16#D3	Event Byte 64

Function code 15 - Force Multiple Coils (S7-300, S7-400)

Function

With this function code, up to 2040 bits (for ET 200S 1SI Modbus max. 1744 bits) can be changed in the slave

Start address

The **Bit start address** parameter is not checked by the driver and is sent unchanged.

Bit number

For **Bit number**, number of coils, any value between **1** and **2040** is permitted (for ET 200S 1SI Modbus up to 1744).

This specifies how many bits in the slave are to be overwritten.

The "Byte counter" parameter in the request message frame is generated by the driver on the basis of the transferred parameter "Bit number".

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#0F	Function code
+2.0	bit_startadr	WORD	W#16#0058	Bit start address
+4.0	bit_number	INT	10	Bit number
+6.0	coil_state[1]	WORD	W#16#EFCD	Status Coil 5FH..58H/57H..50H

Example

Request message frame FUNCTION 15:

05H	Slave address
0FH	Function code
00H	Bit address "High"
50H	Bit address "Low"
00H	Bit number "High"
0AH	Bit number "Low"
02H	Byte counter
CDH	Status Coil 50H..57H
EFH	Status Coil 58H..59H
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 15:

05H	Slave address
0FH	Function code
00H	Bit address "High"
50H	Bit address "Low"
00H	Bit number "High"
0AH	Bit number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

The response message frame is not entered in the receive DB.

Function code 16 - Preset Multiple Registers (S7-300, S7-400)

Function

Function code 16 enables you to overwrite up to **127 registers** in the slave (for ET 200S Modbus 1SI max. 109 registers) with a request message frame.

Start address

The **Register start address** parameter is not checked by the driver and is sent unchanged.

Register number

1 Up to a **maximum of 127 registers** (for ET 200S 1SI Modbus max. 109 registers) can be read (1 register = two bytes).

The "Byte counter" parameter in the request message frame is generated by the driver on the basis of the transferred parameter "number of registers".

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#10	Function code
+2.0	reg_startadr	WORD	W#16#0060	Register start address
+4.0	reg_number	INT	3	Register number
+6.0	reg_data[1]	WORD	W#16#41A1	Register data
+8.0	reg_data[2]	WORD	W#16#42A2	Register data
+10.0	reg_data[3]	WORD	W#16#43A3	Register data

Example

Request message frame FUNCTION 16:

05H	Slave address
10H	Function code
00H	Register address "High"
60H	Register address "Low"
00H	Register number "High"
03H	Register number "Low"
06H	Byte counter
41H	<reg_data[1]> "High"
A1H	<reg_data[1]> "Low"
42H	<reg_data[2]> "High"
A2H	<reg_data[2]> "Low"

43H	<reg_data[3]> "High"
A3H	<reg_data[3]> "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 16:

05H	Slave address
10H	Function code
00H	Register address "High"
60H	Register address "Low"
00H	Register number "High"
03H	Register number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

The response message frame is not entered in the receive DB.

Modbus slave (RTU) (S7-300, S7-400)

Function Code 01 – Read Coil (Output) Status (S7-300, S7-400)

Function

This function enables the MODBUS master system to read individual bits from the SIMATIC memory areas listed below.

Request message frame

ADDR	FUNC	start_address	bit_number	CRC

Response message frame

ADDR	FUNC	Byte_count n	n byte DATA	CRC

start_address

The MODBUS bit address "**start_address**" is interpreted by the driver as follows:

The driver checks that “start_address” is located within one of the areas which were specified during parameter assignment in the dialog box "**Conversion of MODBUS addressing for FC 01, 05, 15**" (from/to: memory bits, outputs, timers, counters).

If the MODBUS bit address **start_address** is located in one of these areas,	access is made to this **SIMATIC memory area**	
from aaaaa to bbbbb	commence at memory bit	M uuuuu.0
from ccccc to ddddd	commence at output	Q ooooo.0
From eeeee to fffff	commence at timer	T ttttt
From ggggg to hhhhh	commence at counter	C zzzzz

The address calculation for access, address conversion, is carried out as follows:

Access beginning with SIMATIC		Conversion formula			
Memory byte	=	((start_address	– aaaaa)	/ 8)	+ uuuuu
Output byte	=	((start_address	– ccccc)	/ 8)	+ ooooo
Timer	=	((start_address	– eeeee)	/ 16)	+ ttttt
Counters	=	((start_address	– ggggg)	/ 16)	+ zzzzz

Access to “memory bits” and “outputs”

When accessing SIMATIC areas “Memory bits” and “Outputs,” the remaining Rest Bit_Number is calculated and used to address the relevant bit within the first / last memory or output byte.

Access to "timers" and "counters"

With the address calculation, it must be possible to divide the result by 16 without having a left over value:

- (start_address - eeeee)
- (start_address - ggggg)

Only access word-by-word starting from word limit is possible.

bit_number

For **bit_number**, number of coils, values between **1** and **2040** (with ET 200S 1SI Modbus max. 1768 bits) are allowed. This number of bits is read.

Access to "timers" and "counters"

When accessing SIMATIC areas “Timers” and “Counters,” it must be possible to divide the "bit_number" by 16 (access word-by-word only).

Please note that a maximum of 16 “timers” and/or “counters” can be read if you are using the CP 341.

Note

Note the CPU-specific restrictions.

Application example

Example of parameter assignment:

Table 1-115 Conversion of MODBUS addressing for function codes FC 01, 05 and 15

MODBUS address in transmission message frame	SIMATIC memory area	
From *0* to *2047*	commence at memory bit	M *1000.0*
From *2048* to *2559*	commence at output	Q *256.0*
From *4096* to *4607*	commence at timer	T *100*
From *4608* to *5119*	commence at counter	C *200*

Request message frame FUNCTION 01:

05H	Slave address ADDR
01H	Function code FUNC
00H	start_address "High"
40H	start_address "Low"
00H	bit_number "High"
20H	bit_number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 01:

05H	Slave address ADDR
01H	Function code FUNC
04H	Byte_count
01H	<DATA 1> M 1008.0 - M 1008.7
17H	<DATA 2> M 1009.0 - M 1009.7
02H	<DATA 3> M 1010.0 - M 1010.7
18H	<DATA 4> M 1011.0 - M 1011.7
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address “start_address” 0040 Hex (64 decimal) is located in the “memory bit” area:

```
Memory byte  =  ((start_address  - aaaaa)  / 8)  + uuuuu
             =  ((64             - 0)      / 8)  + 1000
             =  1008 ;
```

The remaining rest bit_number has the following result:

```
Rest bit_no.    =  ((start_address  - aaaaa)  % 8)  [Modulo 8]
                =  ((64             - 0)      % 8)
                =  0 ;
```

Access is made starting from bit M 1008.0 up to and including M 1011.7.

Bit number:

The number of MODBUS bits "bit_number" 0020 Hex (32 decimal) means that 32 bits = 4 bytes are to be read.

Further examples

Some other access examples are listed in the table below.

All examples are based on the above area specification.

start_address		Access in SIMATIC beginning					->	with
Hex	Decimal	(Decimal)						
0000	0	Bit memory	((0	- 0)	/ 8)	+ 1000	->	M 1000.0
0021	33	Bit memory	((33	- 0)	/ 8)	+ 1000	->	M 1004.1
0400	1024	Bit memory	((1024	- 0)	/ 8)	+ 1000	->	M 1128.0
0606	1542	Bit memory	((1542	- 0)	/ 8)	+ 1000	->	M 1192.6
0840	2112	Output	((2112	- 2048)	/ 8)	+ 256	->	Q 264.0
09E4	2532	Output	((2532	- 2048)	/ 8)	+ 256	->	Q 316.4
1010	4112	Timers	((4112	- 4096)	/ 16)	+ 100	->	T 101
10C0	4288	Timers	((4288	- 4096)	/ 16)	+ 100	->	T 112
1200	4608	Counters	((4608	- 4608)	/ 16)	+ 200	->	C 200
13E0	5088	Counters	((5088	- 4608)	/ 16)	+ 200	->	C 230

Function code 02 - Read Input Status (S7-300, S7-400)

Function

This function enables the MODBUS master system to read individual bits from the SIMATIC memory areas listed below.

Request message frame

ADDR	FUNC	start_address	bit_number	CRC

Response message frame

ADDR	FUNC	Byte_count n	n byte DATA	CRC

start_address

The MODBUS bit address "**start_address**" is interpreted by the driver as follows:

The driver checks that "start_address" is located within one of the areas which were entered during parameter assignment in the dialog box "**Conversion of MODBUS addressing for FC 02**" (from/to: memory bits, inputs).

If the MODBUS bit address is located in area **start_address** ,	**SIMATIC memory area** is accessed	
from kkkkk to lllll	commence at memory bit	M vvvvv.0
from nnnnn to rrrrr	from input	I sssss.0

The address calculation for access, address conversion, is carried out as follows:

Access beginning with Conversion formula
SIMATIC

Memory byte = ((start_address – *kkkkk*) / 8) + *vvvvv*

Input byte = ((start_address – *nnnnn*) / 8) + *sssss*

Access to “memory bits” and “inputs”

When accessing SIMATIC areas “Memory bits” and “Inputs,” the remaining rest bit_number is calculated and used to address the relevant bit within the first / last memory or input byte.

bit_number

For **bit_number**, number of coils, values between **1** and **2040** (with ET 200S 1SI Modbus max. 1768 bits) are allowed. This number of bits is read.

Note

Please note the CPU-specific restrictions.

Application example

Example of parameter assignment:

Table 1-116 **Conversion of MODBUS addressing for function code FC 02**

MODBUS address in transmission message frame	**SIMATIC memory area**	
from 0 to 4095	commence at memory bit	M 2000.0
from 4096 to 5119	commence at input	I 128.0

Request message frame FUNCTION 02:

05H	Slave address ADDR
02H	Function code FUNC
10H	start_address "High"
30H	start_address "Low"
00H	bit_number "High"
18H	bit_number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 02:

05H	Slave address ADDR
02H	Function code FUNC
03H	Byte_count
12H	<DATA 1> I 134.0 - I 134.7
34H	<DATA 2> I 135.0 - I 135.7
56H	<DATA 3> I 136.0 - I 136.7
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "start_address" 1030 Hex (4144 decimal) is located in the "Inputs" area:

Input byte = ((start_address - nnnnn) / 8) + sssss
= ((4144 - *4096*) / 8) + *128*
= 134 ;

The remaining rest bit_number has the following result:

Rest bit_no. = ((start_address - nnnnn) % 8) [Modulo 8]
= ((4144 - 4096) % 8)
= 0 ;

Inputs I 134.0 up to and including I 136.7 are accessed.

Bit number:

The number of MODBUS bits "bit_number" 0018 Hex (24 decimal) means that 24 bits = 3 bytes are to be read.

Further examples

Some other access examples are listed in the table below.

All examples are based on the above area specification.

start_address		Access in SIMATIC beginning					->	with
Hex	Decimal	(Decimal)						
0000	0	Bit memory	((0	- 0)	/ 8)	+ 2000	->	M 2000.0
0071	113	Bit memory	((113	- 0)	/ 8)	+ 2000	->	M 2014.1
0800	2048	Bit memory	((2048	- 0)	/ 8)	+ 2000	->	M 2256.0
0D05	3333	Bit memory	((3333	- 0)	/ 8)	+ 2000	->	M 2416.5
1000	4096	Input	((4096	- 4096)	/ 8)	+ 128	->	I 128.0
10A4	4260	Input	((4260	- 4096)	/ 8)	+ 128	->	I 148.4

Function Code 03 - Read Output Registers (S7-300, S7-400)

Function

This function enables the MODBUS master system to read data words from a data block.

Request message frame

ADDR	FUNC	start_register	register_number	CRC

Response message frame

ADDR	FUNC	Byte_count n	n/2 register DATA (High, Low)	CRC

start_register

The MODBUS register address "**start_register**" is interpreted by the driver as follows:

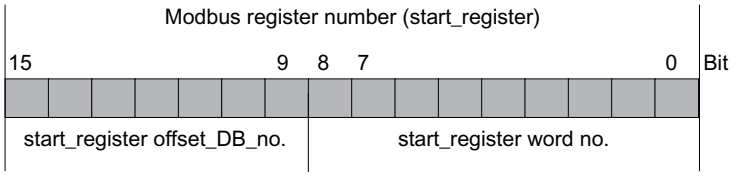

For further address generation, the driver uses the “base DB number” (commencing at DB xxxxx) entered in the dialog box "**Conversion of MODBUS addressing for FC 03, 06, 16**" during parameter assignment .

The address calculation for access, address conversion, is carried out as follows:

Access to SIMATIC		Conversion formula	
Data block DB (resulting DB)	=	(base DB number *xxxxx*	+ start_register offset_DB_No.)
Data word DBW	=	(start_register word_No.	* 2)

Calculation formula for start_register

Providing the resulting DB to be read is known, the MODBUS address **start_register** required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

This is based on even numbered data word numbers only.

register_number

For **bit_number**, number of registers, values between **1** and **127** (with ET 200S 1SI Modbus max. 110 bits) are allowed. This number of registers is read. Please observe the following rule:

$(\text{register_number})_{max}$ = 512 - start_register

Note

Note the CPU-specific restrictions.

Application example

Example of parameter assignment:

Table 1-117 Conversion of MODBUS addressing for function codes FC 03, 06 and 16

MODBUS address in transmission message frame	SIMATIC memory area	
0	commence at data block (base DB number)	DB 800

Request message frame FUNCTION 03:

05H	Slave address ADDR
03H	Function code FUNC
00H	start_register "High"
50H	start_register "Low"
00H	register_number "High"
02H	register_number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 03:

05H	Slave address ADDR
03H	Function code FUNC
04H	Byte_count
87H	<DATA 1> DBW 160 "High"
65H	<DATA 2> DBW 160 "Low"
43H	<DATA 3> DBW 161 "High"
21H	<DATA 4> DBW 161 "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "start_register" 0050 Hex (80 decimal) is interpreted as follows:

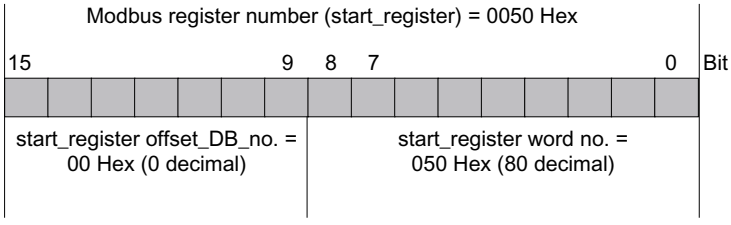

Data block DB (resulting DB)	=	(base DB number xxxx + start_register offset_DB_no.)
	=	(800 + 0)
	=	800;

Data word DBW	=	(start_register word_No. * 2)
	=	(80 * 2)
	=	160;

DB 800, data word DBW 160 is accessed.

Register number:

The number of MODBUS registers "register_number" 0002 Hex (2 decimal) means that 2 registers = 2 data words are read.

Further examples

Some other access examples are listed in the table below.

			start_register				
start_register		Base DB no.	Offset DB_no.	Word number		Resulting DB	DBW
Hex	Decimal	Decimal	Decimal	Hex	Decimal	Decimal	Decimal
0000	0	800	0	000	0	800	0
01F4	500	800	0	1F4	500	800	1000
0200	512	800	1	000	0	801	0
02FF	767	800	1	0FF	255	801	510
0300	768	800	1	100	256	801	512
03FF	1023	800	1	1FF	511	801	1022
0400	1024	800	2	000	0	802	0

Function code 04 - Read Input Registers (S7-300, S7-400)

Function

This function enables the MODBUS master system to read data words from a data block.

Request message frame

ADDR	FUNC	start_register	register_number	CRC

Response message frame

ADDR	FUNC	Byte_count n	n/2 register DATA (High, Low)	CRC

start_register

The MODBUS register address "**start_register**" is interpreted by the driver as follows:

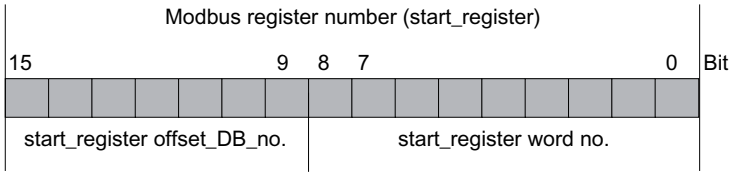

For further address generation, the driver uses the “base DB number” (commencing at DB xxxxx) specified in the dialog box "**Conversion of MODBUS addressing for FC4**" .

The address calculation for access, address conversion, is carried out in two steps as follows:

Access to SIMATIC		Conversion formula
Data block DB (resulting DB)	=	(base DB number *xxxxx* + start_register-offset_DB_no.)
Data word DBW	=	(start_register word_No. * 2)

Calculation formula for start_register

Providing the resulting DB to be read is known, the MODBUS address **start_register** required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

Only even numbered data word numbers are permissible.

register_number

For **bit_number**, number of registers, values between **1** and **127** (with ET 200S 1SI Modbus max. 110 bits) are allowed. This number of registers is read. Please observe the following rule:

$(\text{register_number})_{max}$ = 512 - start_register

Note

Note the CPU-specific restrictions.

Application example

Example of parameter assignment:

Table 1-118 Conversion of MODBUS addressing for function code FC 04

MODBUS address in transmission message frame	SIMATIC memory area	
0	commence at data block (base DB number)	DB 900

Request message frame FUNCTION 04:

05H	Slave address ADDR
04H	Function code FUNC
02H	start_register "High"
C0H	start_register "Low"

00H	register_number "High"
03H	register_number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 04:

05H	Slave address ADDR
04H	Function code FUNC
06H	Byte_count
A1H	<DATA 1> DBW 384 "High"
A2H	<DATA 2> DBW 384 "Low"
A3H	<DATA 3> DBW 385 "High"
A4H	<DATA 4> DBW 385 "Low"
A5H	<DATA 5> DBW 386 "High"
A6H	<DATA 6> DBW 386 "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "start_register" 02C0 Hex (704 decimal) is interpreted as follows:

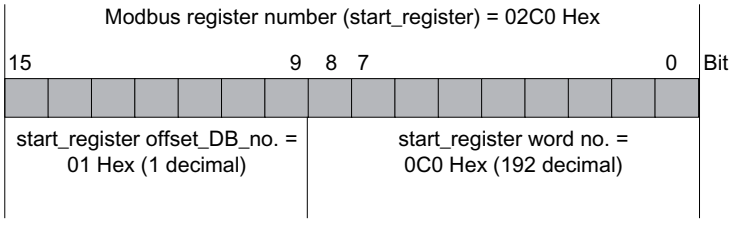

Data block DB (resulting DB)	=	(base DB number xxxx + start_register offset_DB_no.)
	=	(900 + 1)
	=	901;

Data word DBW	=	(start_register word_No. * 2)
	=	(192 * 2)
	=	384;

DB 901, data word DBW 384 is accessed.

Register number:

The number of MODBUS registers "register_number" 0003 Hex (3 decimal) means that 3 registers = 3 data words are read.

Further examples

Some other access examples are listed in the table below.

			start_register				
start_register		Base DB no.	Offset DB_no.	Word number		Resulting DB	DBW
Hex	Decimal	Decimal	Decimal	Hex	Decimal	Decimal	Decimal
0000	0	900	0	000	0	900	0
0064	100	900	0	064	100	900	200
00C8	200	900	0	0C8	200	900	400
0190	400	900	0	190	400	900	800
1400	5120	900	10	000	0	910	0
1464	5220	900	10	064	100	910	200
14C8	5320	900	10	0C8	200	910	400

Function Code 05 - Force Single Coil (S7-300, S7-400)

Function

This function enables the MODBUS master system to write a bit into the SIMATIC memory of the CPU as listed below.

Request message frame

ADDR	FUNC	coil_address	DATA on/off	CRC

Response message frame

ADDR	FUNC	coil_address	DATA on/off	CRC

coil_address

The MODBUS bit address "**coil_address**" is interpreted by the driver as follows:

The driver checks whether "coil_address" is located within one of the areas which were entered during parameter assignment in the dialog box "**Conversion of MODBUS Addressing for FC 01, 05, 15**" (from/to: memory bits, outputs, timers, counters).

If the MODBUS bit address **coil_address** is located in the area	the **SIMATIC memory area** is accessed	
from aaaaa to bbbbb	commence at memory bit	M uuuuu.0
from ccccc to ddddd	commence at output	Q ooooo.0

The address calculation for access (address conversion) is carried out as follows:

Access beginning with SIMATIC		Conversion formula			
Memory byte	=	((coil_address	- *aaaaa*)	/ 8)	+ *uuuuu*
Output byte	=	((coil_address	- *ccccc*)	/ 8)	+ *ooooo*

Access to “memory bits” and “outputs”

When accessing the SIMATIC areas "memory bits" and "outputs", the remaining rest bit_number is calculated and used to address the relevant bit within the memory or output byte.

Access to "timers" and "counters"

Access to the SIMATIC timers and counters areas is not permitted with function code FC 05 and is rejected by the driver with an error message frame.

DATA on/off

The following two values are permitted as **DATA-on/off**:

- FF00H -> Set bit.
- 0000H -> Delete bit.

Application example

Example of parameter assignment:

Table 1-119 Conversion of MODBUS addressing for function codes FC 01, 05, 15

MODBUS address in transmission message frame	SIMATIC memory area	
from 0 to 2047	commence at memory bit	M 1000.0
from 2048 to 2559	commence at output	Q 256.0

Request message frame FUNCTION 05:

05H	Slave address ADDR
05H	Function code FUNC
08H	coil_address "High"
09H	coil_address "Low" A257.1
FFH	DATA on/off "High"
00H	DATA on/off "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Reply message frame FUNCTION 05:

05H	Slave address ADDR
05H	Function code FUNC
08H	coil_address "High"
09H	coil_address "Low" A257.1
FFH	DATA on/off "High"
00H	DATA on/off "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "coil_address" 0809 Hex (2057 decimal) is located in the "outputs" area:

```
Output byte    =  ((coil_address   - ccccc)  / 8)   + ooooo
               =  ((2057           - 2048)   / 8)   + 256
               =   257 ;
```

The remaining rest bit_number has the following result:

```
Rest bit_no.   =  ((coil_address   - ccccc)  % 8)   [Modulo 8]
               =  ((2057           - 2048)   % 8)
               =   1 ;
```

Output Q 257.1 is accessed.

Further examples

For further examples of access to memory bits and outputs, refer to FC 01.

Function code 06 - Preset Single Register (S7-300, S7-400)

Function

This function enables the MODBUS master system to write a data word in a data block of the CPU.

Request message frame

ADDR	FUNC	start_register	DATA value (High, Low)	CRC

Response message frame

ADDR	FUNC	start_register	DATA value (High, Low)	CRC

start_register

The MODBUS register address "**start_register**" is interpreted by the driver as follows:

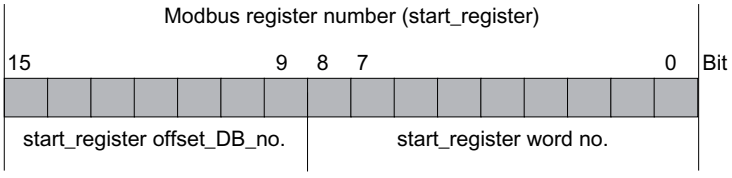

For further address generation, the driver uses the "base DB number" commencing at DB xxxxx entered in the dialog box "**Conversion of MODBUS addressing for FC 03, 06, 16**" during parameter assignment .

The address calculation for access, address conversion, is carried out in two steps as follows:

Access to SIMATIC		Conversion formula
Data block DB (resulting DB)	=	(base DB number xxxxx + start_register-offset_DB_no.)
Data word DBW	=	(start_register word_No. * 2)

Calculation formula for start_register

Providing the resulting DB to be written is known, the MODBUS address **start_register** required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

Only even numbered data word numbers are permissible.

DATA Value

Any value can be used as the DATA **value**, register value.

Application example

Example of parameter assignment:

Table 1-120 Conversion of MODBUS addressing for function codes FC 03, 06 and 16

MODBUS address in transmission message frame	SIMATIC memory area	
0	commence at data block (base DB number)	DB 800

Request message frame FUNCTION 06:

05H	Slave address ADDR
06H	Function code FUNC
01H	start_register "High"
80H	start_register "Low" DBW 768
2BH	DATA value "High"
1AH	DATA value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Reply message frame FUNCTION 06:

05H	Slave address ADDR
06H	Function code FUNC
01H	start_register "High"
80H	start_register "Low" DBW 768
2BH	DATA value "High"
1AH	DATA value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "start_register" 0180 Hex (384 decimal) is interpreted as follows:

Modbus register number (start_register) = 0180 Hex

15						9	8	7							0	Bit
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	
start_register offset_DB_no. = 00 Hex (0 decimal)							start_register word no. = 180 Hex (384 decimal)									

Data block DB (resulting DB)	=	(base DB number xxxx + start_register offset_DB_no.)
	=	(800 + 0) = 800;

Data word DBW	=	(start_register word_No. * 2)
	=	(384 * 2) = 768;

DB 800, data word DBW 768 is accessed.

Further examples

For further access examples, please refer to FC 03.

Function code 08 - Loop back Diagnostic Test (S7-300, S7-400)

Function

This function is used to check the communications connection.

It has no effect on the S7 CPU, the user programs or user data. The received message frame is returned to the master system by the driver independently.

Request message frame

ADDR	FUNC	Diagnostic code (High,Low)	Test Data	CRC

Response message frame

ADDR	FUNC	Diagnostic code (High,Low)	Test Data	CRC

Diagnostic Code

Only **Diagnostic Code 0000** is supported!

Test Data

Any16-bit value.

Application example

Request message frame FUNCTION 08:

05H	Slave address ADDR
08H	Function code FUNC
00H	Diagnostic code "High"
00H	Diagnostic code "Low"
A5H	Test value "High"
C3H	Test value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Reply message frame FUNCTION 08:

05H	Slave address ADDR
08H	Function code FUNC
00H	Diagnostic code "High"

00H	Diagnostic code "Low"
A5H	Test value "High"
C3H	Test value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Function code 15 - Force Multiple Coils (S7-300, S7-400)

Function

This function enables the MODBUS master system to write several bits to the SIMATIC memory areas listed below.

Request message frame

ADDR	FUNC	start_address	quantity	byte_count n	n DATA	CRC

Response message frame

ADDR	FUNC	start_address	quantity	CRC

start_address

The MODBUS bit address "**start_address**" is interpreted by the driver as follows:

The driver checks whether “start_address” is located within one of the areas which were entered during parameter assignment in the dialog box "**Conversion of MODBUS addressing for FC 01, 05, 15**" (from/to: memory bits, outputs, timers, counters).

If the MODBUS bit address is located in area **start_address** ,	**SIMATIC memory area** is accessed	
from aaaaa to bbbbb	commence at memory bit	M uuuuu.0
from ccccc to ddddd	commence at output	Q ooooo.0

The address calculation for access, address conversion, is carried out as follows:

Access beginning with SIMATIC		Conversion formula
Memory byte	=	((start_address - aaaaa) / 8) + uuuuu
Output byte	=	((start_address - ccccc) / 8) + ooooo

Access to “memory bits” and “outputs”

When accessing the SIMATIC areas "memory bits" and "outputs", the remaining rest bit_number is calculated and used to address the relevant bit within the memory or output byte.

Access to "timers" and "counters"

Access to the SIMATIC timers and counters areas is not permitted with function code FC 15 and is rejected by the driver with an error message frame.

Quantity

For **quantity**, bit number, any value between **1** and **2040** (with ET 200S 1SI Modbus max. 1976 bits) is allowed.

Note

Please note the CPU-specific restrictions.

DATA

Any values are contained in the **DATA field** as bit statuses.

Application example

Example of parameter assignment:

Table 1-121 Conversion of MODBUS addressing for function codes FC 01, 05 and 15

MODBUS address in transmission message frame	SIMATIC memory area	
from 0 to 2047	commence at memory bit	M 1000.0
from 2048 to 2559	commence at output	Q 256.0

Action:

The MODBUS master system wants to write the following bit statuses on to memory bits M 1144.0 ... M 1144.7 and M 1145.0 ... M 1145.3:

Memory bit M 1144	7	6	5	4	3	2	1	0	Bit
	ON	ON	OFF	OFF	ON	ON	OFF	ON	
Memory bit M 1145	7	6	5	4	3	2	1	0	Bit
	-	-	-	-	ON	OFF	OFF	ON	

Request message frame FUNCTION 15:

05H	Slave address ADDR
0FH	Function code FUNC
04H	start_address "High"
80H	start_address "Low" (M 1144.0 ...)
00H	quantity "High"
0CH	quantity "Low" (12 bits)
02H	bytecount
CDH	status coil (M 1144.0 ... M 1144.7)
09H	status coil (M 1145.0 ... M 1145.3)
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 15:

05H	Slave address ADDR
0FH	Function code FUNC
04H	start_address "High"
80H	start_address "Low"
00H	quantity "High"
0CH	quantity "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "coil_address" 0480 Hex (1152 decimal) is located in the "memory bit" area:

```
Memory byte   =  ((start_address  - aaaaa)  / 8)   + uuuuu
              =  ((1152           - 0)      / 8)   + 1000
              =   1144 ;
```

The remaining rest bit_number has the following result:

```
Rest bit_no.  =  ((start_address  - aaaaa)  % 8)  [Modulo 8]
              =  ((1152           - 0)      % 8)
              =   0 ;
```

Access is made to memory bits commencing at M 1144.0.

Further examples

For further examples of access to memory bits and outputs, refer to FC 01.

Function code 16 - Preset Multiple Registers (S7-300, S7-400)

Function

The function code enables the MODBUS master system to write several data words in a data block of the SIMATIC CPU.

Request message frame

ADDR	FUNC	start_register	quantity	byte_count n	n-DATA (High,Low)	CRC

Response message frame

ADDR	FUNC	start_register	quantity	CRC

start_register

The MODBUS register address "**start_register**" is interpreted by the driver as follows:

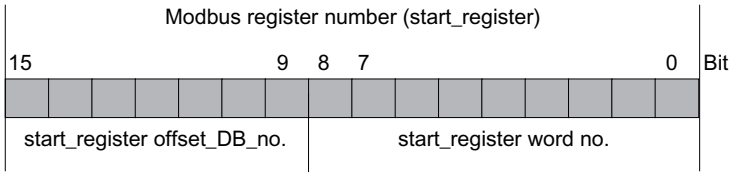

For further address generation, the driver uses the "base DB number" commencing at DB xxxxx entered in the dialog box "**Conversion of MODBUS addressing for FC 03, 06, 16**" during parameter assignment .

The address calculation for access, address conversion, is carried out in two steps as follows:

Access to SIMATIC	Conversion formula
Data block DB (resulting DB)	= (base DB number *xxxxx* + start_register-offset_DB_no.)
Data word DBW	= (start_register word_No. * 2)

Calculation formula for start_register

Providing the resulting DB to be written is known, the MODBUS address **start_register** required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

Only even numbered data word numbers are permissible.

Quantity

For **quantity**, register number, any value between **1** and **127** (with ET 200S 1SI Modbus max. 123 bits) is allowed. Please observe the following rule:

$(\text{Quantity})_{max}$ = 512 - start_register

Note

Please note the CPU-specific restrictions.

DATA (High,Low)

Any value can be used as DATA **DATA (High,Low)** (register value).

Application example

Example of parameter assignment:

Table 1-122 Conversion of MODBUS addressing for function codes FC 03, 06, 16

MODBUS address in transmission message frame	SIMATIC memory area	
0	commence at data block (base DB number)	DB 800

Action:

The MODBUS master system wants to write the values CD09 Hex, DE1A Hex and EF2B Hex to data words DBW 100, DBW 102 and DBW 104 of DB 800.

Request message frame FUNCTION 16:

05H	Slave address ADDR
10H	Function code FUNC
00H	start_register "High"
32H	start_register "Low" DBW 100
00H	quantity "High"
03H	quantity "Low" (3 registers)
06H	bytecount
CDH	register value -High (DBW100)
09H	register value -Low
DEH	register value -High (DBW102)
1AH	register value -Low

EFH	register value -High (DBW104)
2BH	register value -Low
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 16:

05H	Slave address ADDR
10H	Function code FUNC
00H	start_register "High"
32H	start_register "Low"
00H	quantity "High"
03H	quantity "Low" (3 registers)
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "start_register" 0032 Hex (50 decimal) is interpreted as follows:

Modbus register number (start_register) = 0032 Hex

15						9	8	7							0	Bit
0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	
start_register offset_DB_no. = 00 Hex (0 decimal)							start_register word no. = 32 Hex (50 decimal)									

Data block DB (resulting DB)	=	(base DB number xxxx + start_register offset_DB_no.)
	=	(800 + 0) = 800;

Data word DBW	=	(start_register word_No. * 2)
	=	(50 * 2) = 100;

DB 800, data word DBW 100 is accessed.

Further examples

For further access examples, please refer to FC 03.

Error message area (S7-300, S7-400)

Numbering scheme event class / event number

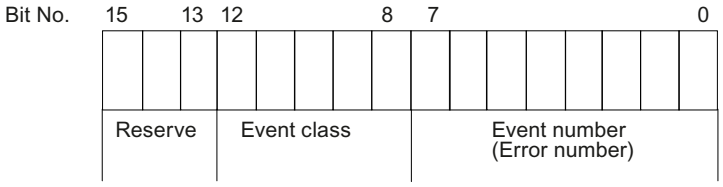

Event classes

Event class	Meaning
2	Error initializing
5	Error executing a CPU job
8	Receive errors
14 ($0E_H$)	General processing error
30 ($1E_H$)	Error during communication between serial port and the CPU.

Event class / event number

Error code(W#16 #...)	Description	Remedy
0201	No (valid) configuration available.	Provide the module with correct parameters. If necessary, ensure the system is correctly installed.
0502	Order inadmissible in this operating mode of the serial interface module ET 200S Modbus/USS (example: device interface is not configured).	Evaluate the diagnostic interrupt and rectify the error accordingly.
050E	Invalid telegram length	The send message frame is longer than 224 bytes. The send job was aborted by the ET 200S Modbus/USS module. Select a shorter telegram length.
0518	• Transmission length during sending is too large (> 4 Kbyte) • The transmission length during SEND is too small.	Check LEN parameter on SEND.
0530	Modbus master send job rejected because the response of the communication partner to a previous reading Modbus master send job has not yet been retrieved.	After a successful read Modbus master send job, you must first read the response from the communication partner from the module before you start a new Modbus master send job.
0551	Frame sequence errors in the communication between the serial interface module ET 200S Modbus/USS and the CPU. The error occurred while transmitting a received message frame of the ET 200S SI serial interface module to the CPU.	The module and the CPU have canceled the transmission. Repeat the receive job. The ET 200S Modbus/USS serial interface module sends the received message again.

Error code(W#16 #...)	Description	Remedy
0806	Character delay time ZVZ exceeded	Error by the partner device or repair disruptions on the transmission line.
080A	Overflow of the receive buffer at the master during the reception of the response message frame.	Check the protocol settings of the slave.
080C	A transmission error was recognized in a character (parity error, overrun error, stop bit error (frame)).	Check to see if the disruptions have an effect on the transmission line. If applicable, change the system structure and the cable routing. Check to see if the protocol parameters for data transmission rate, data bit number, parity, stop bit number are identically set for the communication module and connection partner.
080D	BREAK The receive line to the partner device is disrupted.	Establish the connection between the devices or switch on the partner device. During TTY mode, check to see if the line current is flowing in idle state. If applicable, check and change the default setting of the 2-wire receive line R(A), R(B) with an RS422/485 (X27) connection.
0810	Parity error: There is a break in the connection (line break) between the two communication partners.	Check the connection line of the communication partners or verify that both devices are configured with the same data transmission rate, parity and stop bit number. Change your system setup or cable wiring.
0811	Character frame error: There is a break in the connection (line break) between the two communication partners.	Check the connection line of the communication partners or verify that both devices are configured with the same data transmission rate, parity and stop bit number. Change your system setup or cable wiring.
0812	Additional characters were received after the CTS serial port was set to OFF.	Reconfigure the communications partner or fetch from the serial interface faster.
0830	The response monitoring time after sending a request message frame expired without the beginning of the response message frame being recognized.	Check if transmission line is interrupted (interface analyzer may be required). Check to see if the protocol parameters for data transmission rate, data bit number, parity, stop bit number are identically set for the communication module and connection partner. In the properties dialog of the module, check if the configured response monitoring time is set high enough. Check to see if the specified slave address is available.
0831	The first character of the response message frame from the slave is different from the slave address sent in the request message frame (with "normal" operating mode).	The wrong slave has replied. Check if transmission line is interrupted (interface analyzer may be required).
0832	Overflow of the receive buffer in the communication module upon receipt of the response message frame.	Check the protocol settings of the slave.

Error code(W#16 #...)	Description	Remedy
0833	Bit or register number for function code FC 15/16 and message frame element byte_count do not match.	Correct the bit/register number or byte_count.
0834	Illegal bit coding for "Set bit / Reset bit" recognized.	Use only the 0000Hex or FF00Hex codes for FC05.
0835	Illegal diagnostic subcode (not 0000Hex) at function code FC 08 "Loop back test" recognized.	Use only the subcode 0000Hex for FC08.
0836	The internally formed value of the CRC 16 checksum does not match the received CRC checksum.	Check the formation of the CRC checksum on the Modbus master system.
0837	Message frame sequence error: The Modbus master system sent a new request message frame before the last response message frame was transmitted by the driver.	Increase the timeout on the slave response message frame for Modbus master system.
0850	Length of the receive message frame greater than 224 bytes or greater than the configured message frame length.	Adjust the message frame length of the partner.
0E01	Error during initialization of the driver-specific SCC process.	Reassign parameters of driver and reload.
0E02	Error during startup of driver: Wrong SCC process active (SCC driver). The driver cannot function with this SCC driver.	Reassign parameters of driver and reload.
0E03	Error during startup of driver: Wrong data transmission process active (interface to instructions). The driver cannot work with this data transmission process.	Reassign parameters of driver and reload.
0E04	Error during startup of driver: Illegal interface submodule. The driver cannot run with the configured interface submodule.	Check and correct the parameter assignment.
0E05	Error with driver dongle: No dongle plugged in, or inserted dongle is faulty. The driver is not ready to run.	Check if a driver dongle is plugged into the communication module. If the inserted dongle is faulty, replace it with a functioning dongle.
0E06	Error with driver dongle: The dongle has no valid content. The driver is not ready to run.	Obtain a correct dongle from the SIEMENS office which supplied you with the driver.
0E10	Internal error procedure: Default branch in procedure automatic device.	Restart the communication module (POWER_ON)
0E11	Internal error procedure: default branch for procedure status Send / Receive.	Restart the communication module (POWER_ON)
0E12	Internal error active automatic device: Default branch	Restart the communication module (POWER_ON)
0E13	Internal error passive automatic device: Default branch	Restart the communication module (POWER_ON)
0E20	The number of data bits for this connection must be 8. The driver is not ready to run.	Correct the parameter assignment of the driver.
0E21	The multiplication factor set for the character delay time is not within the value range of 1 to 10. The driver is operating with a default setting of 1.	Correct the parameter assignment of the driver.
0E22	The operating mode set for the driver is illegal. “Normal operation” or “Interference Suppression” must be specified. The driver is not ready to run.	Correct the parameter assignment of the driver.

Error code(W#16 #...)	Description	Remedy
0E23	An illegal value for the response monitoring time has been set: Valid values are 5 to 65500ms. The driver is not ready to run.	Correct the parameter assignment of the driver.
0E24	Invalid limits were configured for write access. The driver is not ready.	Correct the parameter assignment of the driver.
0E25	An invalid "from/to" combination was configured in the specification of the "Conversion of the Modbus addresses with function code FC 01, 05, 15" area. (Memory bits, outputs, timers, counters area). The driver is not ready.	Correct the parameter assignment of the driver.
0E26	An invalid "from/to" combination was configured in the specification of the "Conversion of the Modbus addresses with function code FC 02" area. (Memory bits, inputs area). The driver is not ready.	Correct the parameter assignment of the driver.
0E27	An overlap was configured in the specification of the "Conversion of the Modbus addresses with function code FC 01, 05, 15" area. (Memory bits, outputs, timers, counters area). The driver is not ready.	Correct the parameter assignment of the driver.
0E28	An overlaps was configured in the specification of the "Conversion of the Modbus addresses with function code FC 02" area. (Memory bits, inputs area). The driver is not ready.	Correct the parameter assignment of the driver.
0E2E	An error occurred when reading the interface parameter file. The driver is not ready to run.	Restart the communication module (POWER_ON).
0E30	Internal error during data transmission to CPU: Unexpected acknowledgment Passive.	Can be ignored if it happens intermittently.
0E31	Timeout during data transmission to CPU.	Check CP-CPU interface.
0E32	Error occurred during data transmission to CPU with RCV: Exact failure reason (detailed error) is in SYSTAT before this entry.	Check CP-CPU interface.
0E33	Internal error during data transmission to CPU: Illegal status of automatic device	Check CP-CPU interface.
:		
0E38	An error has occurred accessing SIMATIC areas "Memory bit, output, timers, counters, input" with the function code FC 01 or FC 02: For example, input not available or attempt to read beyond the end of the area.	Check whether the addressed SIMATIC area exists and that no attempt was made to access beyond the end of the area.
0E39	An error has occurred accessing the SIMATIC "Data block" area with the function code FC 03, 04, 06, 16: Data block not available or too short.	Check if the addressed data block exists and is sufficiently long.
0E3A	An error has occurred executing a write job with the function code FC 05, 15: Instance data of the Modbus instruction is not available or too short.	Check if the configured instance DB for the Modbus communication instruction is available and is sufficiently long.

Error code(W#16 #...)	Description	Remedy
0E3B	Timeout (waiting for acknowledgement) in the execution of a write job by the Modbus communication instruction.	Check the connection configuration as well as CP-CPU interface (BSEND), you may need to reload the Modbus communication instruction.
0E3C	Illegal job with this driver.	Only SEND, RCV, STATUS (CP 441-2 only) permitted
0E40	Value specified for LEN parameter in the SEND instruction is too low.	Minimum length is 2 bytes.
0E41	Value specified for LEN parameter in the SEND instruction is too low. A greater length is required for the transferred function code.	The minimum length for this function code is 6 bytes.
0E42	Transferred function code is illegal.	The only function codes permitted are those listed in the section " Function codes (Page 1651) ".
0E43	Slave address 0 (= Broadcast) not permitted with this function code.	Only use slave address 0 for the suitable function codes.
0E44	The value of the transferred “Bit number” parameter is not within the range 1 to 2040.	The “Bit number” parameter must be within the range 1 to 2040.
0E45	The value of the transferred “Register number” parameter is not within the range 1 to 127.	The “Register number” parameter must be within the range 1 to 127.
0E46	Function codes 15 or 16: The value of the transferred “Bit number” or “Register number” parameter is not within the range 1 to 2040 and/or 1 to 127.	The “Bit number” or “Register number” parameter must be within the range 1 to 2040 and/or 1 to 127.
0E47	Function codes 15 or 16: The LEN parameter for the SEND instruction does not correspond to the transferred “Bit number” or “Register number” parameter, LEN parameter is too small.	Increase the LEN parameter for SEND until a sufficient amount of user data is transferred to the communication module. A larger amount of user data must be transferred to the communication module because of the “Bit number” or “Register number”.
0E48	Function code 05: The code specified in SEND source DB for “Set Bit” (FF00H) or “Delete Bit” (0000H) is wrong.	The only permitted codes are FF00H and 0000H.
0E49	Function code 08: The code specified in SEND source DB for “Diagnostic Code” is wrong.	The only permitted code is “Diagnostic Code” 0000H.
0E50	In the word-based SIMATIC timers/counters areas, the resulting residual bit number from the Modbus address is ≠ 0.	Only use Modbus addresses that result in valid bit numbers.
0E51	The received Modbus address is outside the configured "from/to" areas.	Only use addresses as address information in the request message frame that have previously been defined in the configuration.
0E52	• A SIMATIC area limit was exceeded during an access attempt by the Modbus master system: Resulting DB number < 1 • Write access to an unreleased area (configuration) • Write access to the instance DB of the communication instruction	Access area limited to valid SIMATIC memory areas.

Error code(W#16 #...)	Description	Remedy
0E53	A SIMATIC area limit was exceeded during an access attempt by the Modbus master system: For example, overflow from the formation of the resulting DB no. (> 65535)	Access area limited to valid SIMATIC memory areas.
0E54	Access beyond the configured area end or access beyond the SIMATIC area end.	Access area limited to valid SIMATIC memory areas.
0E55	Write access to this SIMATIC memory area is not allowed.	Write only to the SIMATIC memory bits, outputs data areas.
0E56	The coupling cannot be operated because the communication instruction is not running.	Call the Modbus communication instruction in the STEP 7 user program cyclically. You may need to repeat initialization of the communication instruction.
0E57	An error has occurred executing the Modbus function code in the communication instruction.	Analyze the exact cause.
0E4A	The length of this function code is greater than the maximum length.	The manual contains the maximum lengths for each function code.
0E4F	CP 441: The R_ID specified in the SEND instruction is invalid for this driver.	Only use R_IDs 0 to 255 (00 00 00 00 ... 00 00 00 FFH)
	CP 341: The R_TYPE specified at the SEND_RK instruction is invalid for this driver.	"X" has to be entered as R_TYP.
0E50	Slave address incorrect: The received slave address is different from the sent slave address.	The wrong slave has replied. Check if transmission line is interrupted (interface analyzer may be required).
0E51	Function code incorrect: The function code received in the response message frame is different from the sent function code.	Check the slave device.
0E52	Byte Underflow: Amount of characters received is less than should have resulted from the byte counter of the response message frame, or is less than expected with this function code.	Check the slave device.
0E53	Byte Overflow: Amount of characters received is more than should have resulted from the byte counter of the response message frame, or is more than expected with this function code.	Check the slave device.
0E54	Byte counter too small: The byte counter received in the response message frame is too small.	Check the slave device.
0E55	Byte counter wrong: The byte counter received in the response message frame is wrong.	Check the slave device.
0E56	Echo wrong: The data of the response message frame (amount of bits, ...) echoed from the slave are different from the data sent in the request message frame.	Check the slave device.
0E57	CRC Check incorrect: An error has occurred on checking the CRC 16 checksum of the response message frame from the slave.	Check the slave device.

Error code(W#16 #...)	Description	Remedy
0E61	Response message frame with exception code 01: Illegal function	See “Manual of Slave Device”
0E62	Response message frame with exception code 02: Illegal data address	See “Manual of Slave Device”
0E63	Response message frame with exception code 03: Illegal data value	See “Manual of Slave Device”
0E64	Response message frame with exception code 04: Failure in associated device	See “Manual of Slave Device”
0E65	Response message frame with exception code 05: Acknowledge	See “Manual of Slave Device”
0E66	Response message frame with exception code 06: Busy, Rejected message	See “Manual of Slave Device”
0E67	Response message frame with exception code 07: Negative Acknowledgement	See “Manual of Slave Device”
1E0D	"Job aborted due to warm restart, hot restart or reset"	
1E0E	Static error calling instruction DP_RDDAT. The RET_VAL return value of the instruction is provided to you in the SFCERR tag in the instance DB for evaluation.	Load the SFCERR tag from the instance DB.
1E0F	Static error calling instruction DP_WRDAT. The RET_VAL return value of the instruction is provided to you in the SFCERR tag in the instance DB for evaluation.	Load the SFCERR tag from the instance DB.
1E10	Static error calling instruction RD_LGADR. The RET_VAL return value of the instruction is provided to you in the SFCERR tag in the instance DB for evaluation.	Load the SFCERR tag from the instance DB.
1E11	Static error calling instruction RDSYSST. The RET_VAL return value of the instruction is provided to you in the SFCERR tag in the instance DB for evaluation.	Load the SFCERR tag from the instance DB.
1E20	Parameter out of range.	Enter a parameter within the range valid for the function module.
1E41	Number of bytes specified in the LEN parameter of the instruction invalid.	You must stay within a range of values of 1 to 256 bytes.

USS communication (S7-300, S7-400)

Overview of USS communication (S7-300, S7-400)

Position in the system environment

The following USS description refers to the operation of the ET 200S 1SI module in the Modbus/USS model.

Introduction

The USS protocol is a straightforward serial data transmission protocol designed to meet the requirements of drive technology.

The USS protocol defines an access method based on the master-slave principle for communication via a serial bus. One master and up to 31 slaves can be connected to the bus. Master and slaves form a USS network. The data exchanged between master and slave are referred to as network data. The network data block contains the data for the master. The individual slaves are selected by the master using an address character in the message frame. A slave can never send anything without first being initiated by the master. Therefore, direct data transmission between individual slaves is not possible. Communication functions in half-duplex mode. The master function cannot be transferred. The USS system has only one master.

Message frame structure

Each message frame begins with a start character (STX), followed by the length specification (LGE) and the address byte (ADR). The data field comes after that. The message frame ends with the block check character (BCC).

STX	LGE	ADR	1	2	...	N	BCC

For single-word (16-bit) data in the network data block, the high byte is sent first, followed by the low byte. Correspondingly, with double-word data the high word is sent first, followed by the low word.

The protocol does not identify any tasks in the data fields.

Data encryption

The data is encrypted as follows:

- STX: 1 byte, start of text, 02H
- LGE: 1 byte, contains the message frame length as a binary number
- ADR: 1 byte, contains the slave address and message frame type in binary code
- Data fields: One byte each, contents are task-dependent
- BCC: 1 byte, block check character

Data transmission procedure

The master ensures cyclic data transmission in message frames. The master addresses all slave nodes one after another with a task message frame. The nodes addressed respond with a response frame. In accordance with the master-slave procedure, the slave must send the response frame to the master after it has received the task message frame. Only then can the master address the next slave.

General structure of the network data block

The network data block is split into two areas: the parameter area (PKW) and the process data area (PZD).

STX	LGE	ADR	Parameter (PKW)	Process data (PZD)	BCC

- **Parameter area** (PKW)
  The PKW area handles parameter transmission between two communication partners (e.g., the controller and drive). This involves, for example, reading and writing parameter values and reading parameter descriptions and the associated text. The PKW interface generally contains tasks for operation and display, maintenance and diagnostics.
- **Process data area** (PZD)
  The PZD area consists of signals that are required for automation:
  – Control words and setpoints from the master to the slave
  – Status words and actual values from the slave to the master

  The contents of the parameter area and process data area are defined by the slave drives. For additional information on this, refer to the drive documentation.

USS-master (S7-300, S7-400)

Overview of functions (S7-300, S7-400)

Network data transmission sequence

The instructions handle network data transmission in cycles with up to 31 drive slaves, in accordance with the sequence specified in the polling list (configuration DB). Only one job is active for each slave at any one time. The user stores the network data for each slave in a data block (network data block) and calls it from there. As specified in the program definition in the polling list, the data is transmitted to and called from the communications processor via another data storage area (communications processor DB).

Two function calls are required for this procedure (one send and one receive instruction). Another function supports the generation and presetting of the data blocks required for communication.

Performance features:

- Creation of data storage areas for communication, depending on the bus configuration
- Presetting of the polling list
- Message frame structure in accordance with the USS specification
- Network data exchange can be parameterized in accordance with the required network data structure
- Execution and monitoring of PKW jobs
- Handling of parameter change reports
- Monitoring of the complete system and error elimination

Various network data structures can be used to send network data. In other words, the content of network data can be selected according to a given application.

Depending on the structure selected, the network data has a PZD area for process data and a PKW area for parameter processing.

In the PKW area, the master can read and write parameter values, and the slave can display parameter changes by means of parameter change reports.

The PZD area contains signals required for process control, such as control words and setpoints from the master to the slave, and status words and actual values from the slave to the master.

The correct sequence for instruction calls is: S_USST, S_SEND, S_RCV, S_USSR. This is important because the outputs of the S_SEND and S_RCV instructions are only valid in the current cycle of the automation system.

The figure below shows the data traffic between the user program and the USS slave.

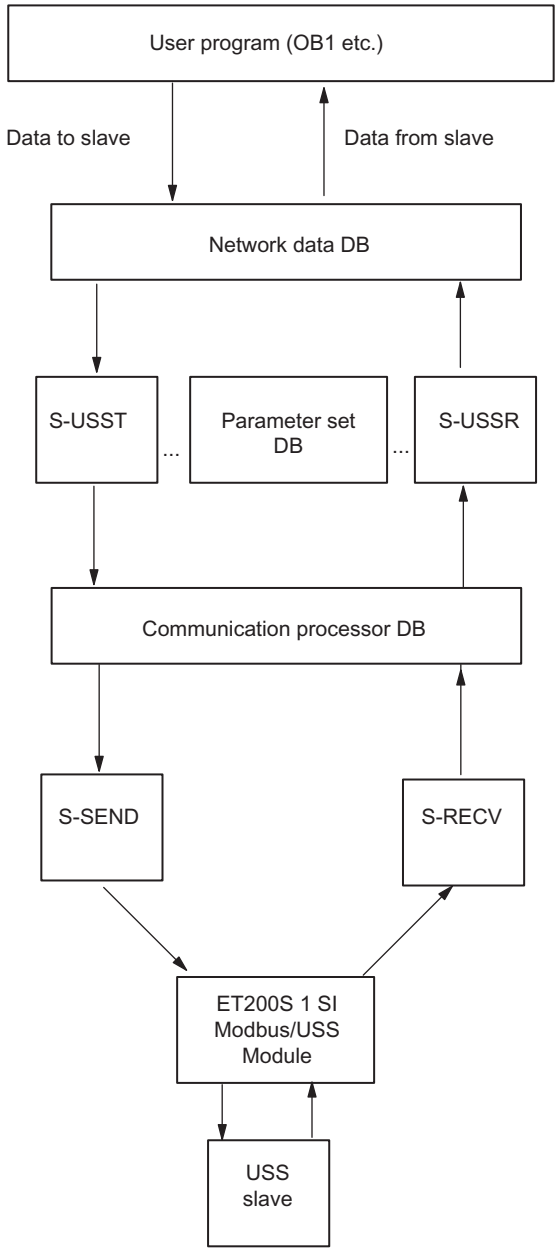

Figure 1-71 Data traffic between the user program and the USS slave

Further Information

Information about the data blocks used can be found here:

- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT

Startup and Diagnostics (S7-300, S7-400)

Startup characteristics (S7-300, S7-400)

Startup characteristics

The start of the communication module is divided into the following phases:

- Initialization (POWER ON of the communication module)
- Parameter assignment

Initialization

Once the communication module is connected to power and after the completion of a hardware test program, the firmware is prepared for the operation on the communication module.

Parameter assignment

During the parameter assignment phase, the communication module receives the module parameters assigned to the current slot. The communication module is now operational.

Operating mode transitions

After the communication module starts up, all data between the CPU and the communication module is exchanged via instructions. The operating state transition characteristics of the communication module depends on the operating mode of the CPU.

CPU STOP	Active data transmission from the communication module to the CPU and both sending and receiving jobs are stopped and a warm restart of the connection is initiated.
CPU RUN	Unlimited send and receive operation is possible in the RUN state of the CPU. The communication module and the given instruction are synchronized in the first instruction cycle following CPU warm restart,.

CPU warm restart	The jobs on the CPU are reset during a warm restart of the CPU; in other words, all active jobs between the CPU and the communication module are automatically cancelled.
Hot restart of the CPU	With a hot restart of the CPU, jobs continue to be processed. With a corresponding configuration in the properties dialog of the module, you can automatically clear the receive buffer on the communication module during CPU startup.

Diagnostic functions (S7-300, S7-400)

Introduction

The diagnostic functions of the communication module allow errors that have occurred to be located quickly. The following diagnostic options are available to you:

Diagnostics via the display elements of the communication module	The indicators provide information on the operating state or the possible error states of the communication module. The indicators provide an initial overview of any internal or external errors and interface-specific errors.
Diagnostics via the STATUS output of the instructions	Instructions have a STATUS output for error diagnostics. This output provides information about communication errors between the communication module and the CPU. You can evaluate the STATUS parameter in the user program.
Diagnostics on the SYSTAT error message area [1]	You can query the status of an interface by programming the STATUS instruction in the user program. By reading the SYSTAT, you obtain detailed information about errors/events that have occurred in communication between the communication module and the associated CPU, as well as the connected communication partner on this interface.
Diagnostics via the error numbers in the response message frame	If you are using the RK512 computer connection and an error occurs with a SEND or FETCH message frame at the communication partner, the communication partner sends a response message frame with an error number in the 4th byte.
Diagnostics via the diagnostic buffer of the communication module	All errors/events are also entered in the diagnostic buffer [2] of the communication module. Similar to the CPU diagnostic buffer, you can also display user-specific information from the CP diagnostic buffer in plain text.
Diagnostic interrupt	The communication module can trigger a diagnostic interrupt on the CPU assigned to it. The communication module provides 4 bytes of diagnostic information. The analysis of this information is made via the user program or by reading the CPU diagnostic buffer.

[1] CP 441-1 and CP 441-2 only

[2] CP 340, ET 200S 1SI modules and CPU31xC-2 PtP have no diagnostic buffer

1.4.7 TeleControl

1.4.7.1 Configuring telecontrol modules

Configuration

Communication types

In this parameter group, you enable the communication types you want to use for the particular module.

To minimize the risk of unauthorized access to the station, you need to enable the communications services that you want to use with the module individually.

You can leave all options disabled if you do not need them, for example if you only use a CP to send SMS messages or for Open User Communication.

"Communication types" parameter group

- **Enable telecontrol communication**
  In the module, enables the telecontrol communication.

  Note
  Full telecontrol functionality only with enabled security functions

  For the following functions, you need to enable the security functions:
  - Sending e-mails using the telecontrol functionality
  - Use of the "TeleControl Basic" (general) protocol
  - Use of the DNP3 security functions
  - Use of certificates

  Note
  Loss of configuration data when changing the telecontrol protocol

  If you change the protocol on a configured module, protocol-specific configuration data is lost. Among other things, this includes the following:
  - Partner configuration
  - Data point configuration
  - Configured messages (e-mails / SMS messages)
  - Configured time-of-day synchronization
  - Configured VPN partner

  Depending on the module type, the following protocols can be used:

  – TeleControl Basic *
  Enables communication with the telecontrol server.

  – ST7
  Enables communication with a control system that supports the ST7 protocol.

  – DNP3 *
  Enables communication with up to four DNP3 masters.

  – IEC 60870-5-104 *
  Enables communication with up to four IEC masters.

  * For telecontrol communication via SINEMA Remote Connect see the following section.
  For configuration of the telecontrol connections for the ST7, DNP3 and IEC protocols, see section AUTOHOTSPOT.
  You can find a description of important parameters and functions of the telecontrol protocols in the section AUTOHOTSPOT.

- **Activate online functions**
  In the module enables access to the CPU for the online functions (diagnostics, loading project data etc.). If the function is enabled, the engineering station can access the CPU via the module.
  If the option is disabled, you have no access to the CPU via the module with the online functions. Online diagnostics of the CPU with a direct connection to the interface of the CPU however remains possible.

- **Enabling S7 communication**

Enables the functions of S7 communication with the assigned SIMATIC CPU and the S7 routing.
If you configure S7 connections to the relevant station, and these run via the module, you will need to enable this option.
S7 routing is supported by the following modules:

– CP 1243-1, CP 124x-7, CP 1243-8
  As of CP firmware V2.1 with CPU ≥ V4.2
– CP 1542SP-1, CP 1542SP-1 IRC, CP 1543SP-1
  As of CP firmware V1.0 with CPU ≥ V2.0

Remote maintenance with SINEMA Remote Connect (SINEMA RC)

The application "SINEMA Remote Connect" (SINEMA RC) is available for remote maintenance purposes.

SINEMA RC uses OpenVPN for encryption of the data. The center of the communication is SINEMA RC Server via which communication runs between the subscribers and that manages the configuration of the communications system.

Usable modules and telecontrol protocols

The following CPs support communication via SINEMA RC:

- CP 1243-1, CP 1243-7, CP 1243-8
  As of firmware V3.1
- CP 1542SP-1 IRC, CP 1543SP-1
  As of firmware V2.0

VPN-capable telecontrol CPs can also handle telecontrol communication via the SINEMA RC server. The following telecontrol protocols support communication via SINEMA RC:

- TeleControl Basic
- DNP3
- IEC 60870-5-104

Applications and configuration

Make the following settings for using SINEMA Remote Connect and telecontrol communication via SINEMA RC:

1. Communication via SINEMA RC: Parameter group "Security > VPN"
   Activate VPN so that the other parameters become visible.
2. Telecontrol communication via SINEMA RC: "Communication types" parameter group
   Communication via SINEMA RC must be enabled beforehand.

The following application cases result from the combination of telecontrol communication and the use of SINEMA RC:

- (1) No telecontrol communication and no SINEMA RC CP for network separation.
- (2) CP only for remote maintenance via SINEMA RC
- (3) CP for telecontrol communication only

- (4) CP uses telecontrol communication, but SINEMA RC only for remote maintenance.
- (5) CP uses SINEMA RC for telecontrol communication and remote maintenance.

The table provides an overview of the applications with the parameter settings.

Table 1-123 Use cases and parameters to be activated

Use case	Parameter settings * (parameter abbreviated)		
	SRC	TC	TC + SRC
(1)	Off	Off	Off
(2)	On	Off	Off
(3)	Off	On	Off
(4)	On	On	Off
(5)	On	On	On

* Explanation of the parameter abbreviations:

SRC - Security > VPN (activated) > "VPN connection type":
"Automatic OpenVPN configuration via SINEMA Remote Connect Server"

TC - Communication types > Telecontrol communication enabled

TC + SRC - Communication types >
"Activate communication via SINEMA Remote Connect"

Telecontrol communication via SINEMA Remote Connect

Steps in configuration

Follow the steps below when configuring the CP for use of telecontrol communication via SINEMA RC:

1. In the "Communication types" parameter group activate telecontrol communication and select one of the protocols named above.
   The option for communication via SINEMA RC is not yet visible.
2. Change to the "Security" parameter group and enable the security functions.
   (In the "Communication types" parameter group the SINEMA RC option appears disabled and grayed out)
3. Open the "Security > VPN" parameter group and enable VPN.
4. For the parameter "VPN connection type" select the option "Automatic OpenVPN configuration via SINEMA Remote Connect Server" if this is not preset.
   (In the "Communication types" parameter group the SINEMA RC option becomes usable.)
5. Change to the "Communication types" parameter group and set the check mark for the option "Activate communication via SINEMA Remote Connect".
   The CP is now prepared for telecontrol communication via SINEMA Remote Connect.
6. Create the remaining configuration of the SINEMA RC Server in "Security > VPN".
7. Import the certificate of the SINEMA RC Server in "Security > Certificate manager".

Configuration of connection establishment

The connection establishment is configured in the "Security > VPN > Optional settings" parameter group with the parameter "Connection type".

- **Update interval**
  With this parameter you set the interval at which the CP queries the configuration on the SINEMA RC Server.
  Note that with the setting 0 (zero) changes to the configuration of the SINEMA RC Server may result in the CP no longer being capable of establishing a connection to the SINEMA RC Server.
- **"Connection type"**
  The two options of the parameter have the following effect:
  - Permanent
    The option is intended for permanent communication.
    This is, for example, the case with telecontrol communication via SINEMA RC. The CP and the communications partner of the CP must be configured in the SINEMA RC Server.
  - PLC trigger
    The option is intended for sporadic communication of the CP via the SINEMA RC Server. You can use this option when you sporadically want to establish temporary connections between the CP and the engineering station (PC with the STEP 7 project of the CP), for example for service work.
- **PLC tag for connection establishment**
  If the option "PLC trigger" is selected, the CP establishes a connection when the PLC tag changes to the value 1. During operation the PLC tag can be set when necessary, for example using an HMI panel.
  When the PLC tag is reset to 0, the connection is terminated again.

Basic settings

Configuration

The parameter group is only available for some communication modules of the SINAUT ST7 system.

For the supported modules, see section Expansion of SINAUT projects (Page 1802).

Telecontrol configuration

This is where you define the procedure for configuring the subscriber and SINAUT connection data for the communication module.

- Configure
  The above-mentioned configuration data is configured in the parameter groups of the module and in the connection editor.
- Import
  The above-mentioned configuration data is configured for a proxy module in STEP 7 V5. The STEP 7 V5 configuration data of the proxy module is imported into STEP 7 Professional for the module.
  For information on importing and the scope of the imported data, see section Import partner configuration (ST7 protocol) (Page 1714).

Time-of-day synchronization

Time-of-day synchronization for telecontrol CPs

Note

Recommendation: Time-of-day synchronization only by 1 module

Only have the time of day of the station from an external time source synchronized by a single module so that a consistent time of day is maintained within the station.

When the CPU takes the time from the CP, disable time-of-day synchronization of the CPU.

Module and synchronization method

If an external time source is used, the S7 station can obtain the current time both via the CPU and via a Telecontrol CP.

There is no forwarding of the time of day from the station to the subnet for S7-1200 / ET 200SP.

- **Time-of-day synchronization of the CPU**
  As the synchronization method, with the CPU only NTP can be selected.
  For the options for setting the CPU time via a telecontrol CP see below.
- **Time-of-day synchronization of telecontrol CPs**
  The following synchronization methods are available for the CPs:
  - **NTP**
    NTP can also be used when the telecontrol communication is disabled. In this case, the telecontrol CP is used as an extended PROFINET interface of the CPU.
    The address of the NTP server can also be entered as a URL, e.g. <ntp.server.com>.
    If the security functions are enabled, the method "NTP (secure)" can also be used.
  - **Time from partner**
    When telecontrol communication is enabled, the CP automatically takes the time-of-day from the communications partner.
  - **Time from the CPU**
    All CMs/CPs of the station are synchronized by the time of the CPU.

Forwarding the time from the CP to the CPU (S7-1200)

Note

Forwarding the time to the CPU

Depending on the firmware version of the modules involved, the time-of-day of the CP is forwarded to the CPU in different ways:

- Optional forwarding of the CP time to the CPU using a PLC tag
- Obligatory forwarding of the CP time to the CPU via the backplane bus

The forwarding of the CP time to the CPU depends on the firmware version of the CP and the CPU. Note the following behaviour.

- **CP firmware ≤ V2.1.5**
  With this firmware version the CP can make the time-of-day available to the CPU as an option via a PLC tag. When this PLC tag is read cyclically by the CPU, the CPU adopts the CP time.
  For the PLC tag, see the parameter group "Communication with the CPU" of the CP.
- **CP firmware ≥ V2.1.7 and CPU firmware ≥ V4.2**
  If both modules in the station have one of the named firmware versions, the time of day of the CP is automatically forwarded to the CPU. Since the CPU automatically adopts the CP time, you no longer require the forwarding option using the PLC tag.
  If for the CPU the option "CPU synchronizes the modules of the device" is enabled in "PROFINET interface > Time synchronization", all smart modules of the station are synchronized with the CPU time.

Time-of-day synchronization with NTP

NTP mode

In NTP mode, the CP sends time-of-day queries at regular intervals to one or more NTP servers. From the responses of the servers, the module selects the most accurate time of day.

The advantage of this mode is that it allows the time to be synchronized across subnets.

NTP generally uses UTC (Universal Time Coordinated). This corresponds to GMT (Greenwich Mean Time).

NTP server and synchronization interval

The IP address of at least one NTP server must be configured.

The synchronization interval specifies the cycle of the time-of-day queries to the NTP server. Depending on the selected module, the cycle can be selected from a drop down list or configured within a range of values between 10 seconds and 1 day (86400 seconds).

"Accept time of day from non-synchronized NTP servers" option

If the option is supported by the selected module the following behavior applies:

- Enabled
  The module also accepts the time-of-day from non-synchronized NTP servers with stratum ≥ 15.
- Disabled
  If the module receives a time-of-day frame from a non-synchronized NTP server with stratum ≥ 15, the time of day is not set according to the frame.

Ethernet interface

Reconnection delay

Note

If the partner cannot be reached, connection establishment via the mobile wireless network can take several minutes. This may depend on the particular network and current network load.

Depending on your contract, costs may result from each connection establishment attempt.

"Reconnection delay" parameter

The reconnection delay for connections in telecontrol communication is the waiting time between repeated attempts to establish the connection by the CP when the telecontrol server is not reachable or the connection has aborted. This waiting time avoids continuous connection establishment attempts at short intervals if there are connection problems.

A basic value is configured for the waiting time before the next connection establishment attempt. Starting at the basic value, the current waiting time is doubled after every 3 unsuccessful retries up to a maximum value of 900 s. Range of values for the basic value: 10 .. 300 s

Example: The basic value 20 results in the following intervals (waiting times) between the attempts to re-establish a connection.

- three times 20 s
- three times 40 s
- three times 80 s
- etc. up to max. 900 s

Note

Special characteristics with CP 1242-7 (6GK7242-7KX30-0XE0) together with TELECONTROL SERVER BASIC (V2)

If a second telecontrol server or a second router of the telecontrol server is configured, the CP attempts to connect to the second partner at the 4th attempt. If the second partner is also unreachable, the 7th time the CP attempts to connect to the first partner again.

Serial interface

Activation of the serial interface interface of the CP 1243-8 IRC (V3) for the ST7 protocol

Serial interface of the CP 1243-8 IRC

When you insert the CP from the catalog in the project, the serial interface of the CP is not visible in the network view of STEP 7.

When using the types of communication "DNP3" and "IEC" the serial interface cannot be used.

After activating thde type of communication "ST7", the serial interface is visible but cannot be configured. To use the serial interface of the CP, a TS module is required.

Follow the steps below to insert the TS module and to activate the serial interface for ST7 communication.

1. From the network view, open the device view of the CP (double-click on the module).
   The device view opens.
   At the same time the catalog view changes. Among other things the Communications modules" directory appears.
2. Expand the "Communications modules" directory:
   > Industrial Remote Communication > TS Modules
3. Expand the subdirectory of the required TS module.
4. Select the TS module.
   A vertical blue line appears on the symbol of the CP.
5. Now drag the TS module onto the blue line of the CP in the device view.
6. The TS moduell with the serial interface is inserted in the station.
   At the same time a new WAN interface is inserted in the network view.

You can configure the serial interface directly in the device view with the TS adapter or in the WAN interface of the CP in the network view.

Parameters of WAN networks (dedicated line / dial-up network)

Interface operating mode

The operating mode of the serial interface specifies the mode of the data transfer.

A maximum of one serial interface of a communications module may be operated in DMA mode.

Range of values:

- Interrupt (block)
  This operating mode applies to the transmit and receive mode. The default mode Interrupt (block) is suitable for all connections. 4 characters are transferred per block. Following this, there is an interrupt. The received characters are checked only after a complete data frame has arrived.
- DMA
  This operating mode applies to the transmit and receive mode. The DMA mode should be used for connections with a high baud rate or a lot of data traffic, however not for GSM networks.
- Interrupt (individual characters)
  This operating mode is used only in the receive direction. In the transmit direction, the block mode continues to the used. The interrupt mode is suitable for extremely bad lines. An interrupt is triggered per transmitted character and each character is analyzed immediately after it is received allowing extremely good diagnostics of transmission errors. This mode is more reliable than the block mode but is slower.

Transmission criterion

Only for interfaces with the Connection mode "Station" / "Node station".

The transmission criterion controls connection establishment for the transmission of conditional spontaneous data frames. This reduces the number of connection retries.

Range of values:

- Standard conditions
  No connection will be established due to the existence of conditional spontaneous data frames.
  Only in the following cases will a connection be established to send conditional spontaneous data frames.
  - Threatened overflow of the send buffer
  - Connection establishment by the communications partner
- Degree of filling
  The module only establishes a connection when the configured fill level of the send buffer for conditional spontaneous data frames is exceeded.
  In the input box, enter the fill level (%) of the send buffer at which when exceeded the module establishes a connection.
- Time
  The module sends conditional spontaneous data frames at a time of day, configurable with "Hours" / "Minutes".
- Time scheme
  The module sends conditional spontaneous data frames cyclically at an interval, configurable with "Hours" / "Minutes".

Polling monitoring time

Only for WAN networks of the type dedicated line

The communications module monitors the frame traffic via a WAN. If the module is not called after a certain time has elapsed, it signals its CPU that the master station is disrupted. The sequence is as follows:

- Polling monitoring time configured
  After the configured time elapses, the module sends a message to its local CPU.
- Polling monitoring time not configured
  If no monitoring time is configured, the module outputs a disruption message after the following internally calculated times.
  - At a transmission speed of 9,600 bps: After 4 seconds without message traffic
  - At a transmission speed of 1,200 bps: After 32 seconds without message traffic

Web server

The Web server of the module

The communications module makes the HTML pages of its Web server available for access by a PC with Web browser. The HTML pages of the WBM (Web Based Management) provide configuration and diagnostic information of the module.

General

- **Enable Web server on this module**
  If the option is activated, data processing is activated in the Web server of the module making access with HTTP/HTTPS possible.
  For modules with multiple interfaces you must also specify the interfaces for access (see below).
- **Allow access only using HTTPS**
  Allows access to the Web server only with the secure protocol HTTPS.
  If the option is enabled only a secure HTTPS connection can be established via the open port 443 of the communications module.
  If the option is disabled, an HTTP connection is established via port 80 of the communications module.

Automatic update

- **Enable automatic update**
  Enables automatic updating of the displayed values.
  If the option is disabled, only the values at the time of connecting to the Web server are displayed.
- **Update interval**
  Select the interval at which you require an update of the displayed values.
  Default setting: 30. Permitted range: 5...999

Overview of the interfaces

Here by selecting the check boxes you specify the interfaces for which access to the WBM of the module is allowed.

The settings for activating the interfaces for Web server access are exchanged between the two parameter groups "Web server" and "Ethernet interface > Access to the Web server".

If no interface is enabled for access, it is not possible to access the WBM of the module.

Partner stations

Import partner configuration (ST7 protocol)

The import options are only available for modules that use the ST7 protocol and that were configured earlier as a proxy in STEP 7 V5.

Selection of the "Import" configuration procedure in the "Basic settings" parameter group is required for the import of configuration data from STEP 7 V5, see section Configuration (Page 1707).

For the supported modules, see section Expansion of SINAUT projects (Page 1802).

Parameter groups

Proxy import

- Import partner configuration
  With this button you import the configuration file (SDBs) from the STEP 7 V5 project. The configuration file must be located in the file system of the engineering station.
- Reset partner configuration
  With this button, you delete all the configuration data of the module that you previously imported from a text file with the "Import partner configuration" button. With this you reset the partner configuration data to the status of an unconfigured module.
- Imported configuration file
  Shows the name of the text file with the configuration data you last loaded.
- Last import
  Shows the time at which you imported the configuration file.

Please note:

- If you import a second time, the data of the first import will be overwritten.
  If you import a second time, always check the configuration, the connections and data points of the module.
- If you change from "Import" to "Configure" following an import, all imported ST7 connections including the corresponding time data will be deleted. Configured data points lose their partners.

Partner stations

The table shows the data of the communications partner imported from STEP 7 V5 after the import:

- Subscriber number
- Subscriber ID
- Subscriber additional identifier
- Number of configured partners
- Subscriber numbers of the partners

Imported configuration data

Imported data

When importing the configuration data of a proxy module from STEP 7 V5 into the compatible module in STEP 7 Professional, the ST7-specific subscriber data and the SINAUT connections are imported.

The import file generated in STEP 7 V5 contains the data of the following SDBs:

- SDB 3202: WAN data
  Data of a serial interface cannot be adopted in a CP 1542SP-1 IRC, see also section Expansion of SINAUT projects (Page 1802).
- SDB 3203: Subscriber data
- SDB 3207: S7 connection data
- SDB 3208: Ethernet data

Data point information / SINAUT objects

The response differs depending on the TD7 version used in STEP 7 V5:

- TD7onTIM
  No information is converted to data points.
- TD7onCPU
  When importing to a TIM 1531 IRC from the information about the blocks assigned in STEP 7 V5 the TD7onCPU blocks of the CPU 1500 are created.
  If you revoke the import ("Reset partner configuration", see above) the TD7onCPU blocks are retained.

Data of time-of-day synchronization

Data of time-of-day synchronization is adopted from STEP 7 V5.

With the following modules, the time-of-day data of a serial interface can then be changed in STEP 7 Professional.

- CP 1243-8 IRC as of firmware version V3
- TIM 1531 IRC

Subscriber additional identifier

After importing the configuration data, the display includes "Subscriber additional identifier" (32-bit). Depending on the subscriber type of the partner, the subscriber additional identifier contains the following information:

- CPU (TD7)
  Status of bit 0:
  - 0: CPU does not send operating status to partner.
  - 1: CPU sends operating status to partner.

  The remaining bits are not used for the CPU.
- TIM (ST7)

Bit no.	Meaning
Bit 0	DCF77 clock • 0: Not available • 1: Available
Bit 1	Connection establishment from communication module to CPU • 0: Not permitted • 1: Permitted
Bit 2	Serial interface 1 (WAN driver) • 0: Interface not available • 1: Interface available
Bit 3	Serial interface 2 (WAN driver) • 0: Interface not available • 1: Interface available
Bit 4...15	Not used
Bit 16...19	Serial interface 1: Network type • 0: Dedicated line • 1: Spontaneous network (mobile network) • 2: Dial-up network • 3...4: Not used • 5: MSC network • 6...15: Not used
Bit 20...21	Serial interface 1: Network node type • 0: Master station • 1: Node station • 2: Station
Bit 22	Serial interface 1: Frame format • 0: FT1.2 • 1: FT2
Bit 23	Not used

Bit no.	Meaning
Bit 24...27	Serial interface 2: Network type • 0: Dedicated line • 1: Spontaneous network (mobile network) • 2: Dial-up network • 3...4: Not used • 5: MSC network • 6...15: Not used
Bit 28...29	Serial interface 2: Network node type • 0: Master station • 1: Node station • 2: Station
Bit 30	Serial interface 2: Frame format • 0: FT1.2 • 1: FT2
Bit 31	Not used

Telecontrol server (Telecontrol Basic protocol)

Addressing of the duplicate or redundant telecontrol server

- **TCSB V3: Addressing the TCSB redundancy combination**
  For CPs as of firmware version V2 with the "Telecontrol Basic" protocol
  In the LAN in the master to which the TCSB server PCs and the DSL router (e.g. SCALANCE M) are connected, the Network Load Balancing (NLB) of the computer operating system will assign a common virtual IP address to the two server PCs.
  This IP address is configured depending on the network setup:
  - If CPs without a DSL router are connected, the virtual address assigned by the NLB must be configured in the CPs as the IP address of the telecontrol server.
  - If a DSL router is used, only one IP address will be configured to address the redundant telecontrol server in the stations, the public address of the DSL router.
    Set the port forwarding on the DSL router so that the public IP address (external network) is led to the virtual IP address of the TCSB server PCs (internal network). Only the public IP address is reachable from the Internet. The station does not therefore receive any information telling it which of the two computers of the redundancy group it is connected to.

  If you configure a second IP address, you need to make sure that TCSB is reachable via the IP address of a second router.
- **TCSB V2: Addressing the main and substitute telecontrol server**
  Only for CP 1242-7 (6GK7 242-7KX30-0XE0)
  With TCSB V2, two independent server PCs can be installed. The address of the substitute telecontrol server also needs to be configured only when these are connected to the Internet via two routers.

Communication with the CPU

CPU scan cycle

Input data points are assigned to the read cycle of the CPU in the data point configuration in the "General > Read cycle" tab.

Structure of the CPU scan cycle

The cycle with which the transferring module (CP / TIM) scans the memory area of the CPU is made up of the following phases:

- High-priority read jobs
  (Fast cycle)
  For all data points with the assignment "Fast cycle" the PLC tags are read in every scan cycle.
  Normally it is adequate to assign only data to be acquired quickly such as alarms and contact changeover messages and command, setpoint and parameter objects for the 1oon check to the fast cycle.
- Write jobs
  In every cycle, the values of a certain number of unsolicited write jobs are written to the CPU.
  - Telecontrol CP / TIM 1500
    The number of tags written per cycle is specified for the transferring module in the "Communication with the CPU" parameter group with the "Max. number of write jobs" parameter. The tags whose number exceeds this value are then written in the next or one of the following cycles.
- Low-priority read jobs - proportion
  (Normal cycle)
  For data points with the assignment "Normal cycle", a proportion of the values of their PLC tags is read in every scan cycle.
  - Telecontrol CP / TIM 1500
    The number of tags read per cycle is specified for the transferring module in the "Communication with the CPU" parameter group with the "Max. number of read jobs" parameter. The tags that exceed this value and can therefore not be read in one cycle are then read in the next or one of the following cycles.
- Cycle idle time
  This waiting time between two scan cycles is used to reserve adequate time for other processes that access the CPU.

Duration of the CPU scan cycle

Since no fixed time can be configured for the cycle and since the individual phases cannot be assigned a fixed number of objects, the duration of the scan cycle is variable and can change dynamically.

Partner status and Path status

You will find the parameters described below in the following parameter group of modules with the type of communication "ST7":

"Communication with the CPU" > "Partner status"

The following statuses are monitored and written to PLC tags:

- **Partner status**
  Availability of a communication partner
- **Path status**
  Status of the connections to a communication partner

Partners / Partner name

This is where you define the communication partners of the selected module whose reachability status and path status of the telecontrol connections should be monitored. Partners can be CPUs or a control center application (for example, ST7cc).

All partners with which a telecontrol connection has been configured can be selected in the table.

PLC tags with the various TIM types

The PLC tags in which the statuses are written differ depending on the TIM type:

- **TIM 4R-IE**
  - **Partner status**
    The Partner status is written to a PLC tag of the type Bool (DB, bit memory, output). For the coding see below.
  - **Path status**
    The Path status is written to a PLC tag of the type Byte (DB, bit memory, output). For the assignment of the PLC tags, see below.
- **TIM 1531 IRC / CP 1243-8 IRC**
  - **Partner status / path status**
    The two statuses are stored in a common PLC tag of the type Word or UInt (DB, bit memory, output).
    The two statuses each occupy one byte of the PLC tag. The coding is written separately for byte 0 (partner status) and byte 1 (path status).

Partner status

- **TIM 4R-IE**
  The bit for partner status is coded as follows (range of values):
  - 0: Partner not reachable
  - 1: Partner reachable
- **TIM 1531 IRC / CP 1243-8 IRC**
  Byte 0 of the common PLC tag for partner status / path status codes information on the availability of the communications partner, on the existing connections and connection paths and on the status of the send buffer of the TIM.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Path redundancy	**Connection mode**	**Temporary connection ****	*(Reserved)*	**Frame memory ***		**Path status**	**Partner status**
0: No redundancy 1: Redundancy exists	0: Permanent 1: Temporary	0: Partner not reachable 1: Partner reachable *	-	0: Send buffer OK 1: Memory allocation > 90% 3: Overflow (memory allocation 100%)		0: Not all paths reachable 1: All paths reachable	0: Partner not reachable 1: Partner reachable

* Partners that support temporary connections are set to 'reachable' if the partner itself terminates the connection.
** For the behavior of the frame memory, refer to the section "Send buffer" in the section Process image, type of transmission, event classes (Page 1765).

Path status

Byte 1 shows the status of the connection path (configured connection) to the partner from the point of view of the local TIM.

A maximum of 2 paths (main and substitute path) to a partner can be configured, see section Creating telecontrol connections (Page 1729).

Both connection paths must start or end on a local TIM.

The PLC tag shows the following:

- The paths via which the partner can be reached.
- The path currently being used
- The TIM interface via which the main path was configured.
- The TIM interface via which the substitute path was configured.

The path of a connection is specified as a combination of the used interfaces of the TIM and the status of the path.

- **Byte assignment**
  The byte is assigned as follows:
  - Two bits for the interface of the main path
  - Two bits for the interface of the substitute path
  - Two bits for the status of the main path
  - Two bits for the status of the substitute path

Bit 6 + 7	Bit 4 + 5	Bit 2 + 3	Bit 0 + 1
Configured interface		Path status	
No. for substitute path	No. for main path	Substitute path (2nd path)	Main path (1st path)

- **Configured interface**
  In the following table the interfaces of the TIM modules are numbered through from 0 .. 3 and have the following meaning:
  - **"0"**
    TIM 4R-IE: Ethernet interface 1 (X3)
    TIM 1531 IRC: Ethernet interface 1 (X1)
    CP 1243-8 IRC: Ethernet interface (X1)
  - **"1"**
    TIM 4R-IE: Ethernet interface 2 (X4)
    TIM 1531 IRC: Ethernet interface 2 (X2)
    CP 1243-8 IRC: WAN interface (X2)
  - **"2"**
    TIM 4R-IE: WAN interface 1 (X1)
    TIM 1531 IRC: Ethernet interface 3 (X3)
  - **"3"**
    TIM 4R-IE: WAN interface 2 (X2)
    TIM 1531 IRC: WAN interface 1 (X4)

  For the CP 1243-8 IRC, the interface numbers "2" and "3" are not relevant.

Status bit 5 / bit 7	Status bit 4 / bit 6	Meaning
0	0	Coding for interface "0"
0	1	Coding for interface "1"
1	0	Coding for interface "2"
1	1	Coding for interface "3"

- **Path status**
  - Main path = 1st path, coding by bits 0 and 1
  - Substitute path = 2nd path, coding by bits 2 and 3

Status bit 1 / bit 3	Status bit 0 / bit 2	Meaning of bit 1 / bit 3	Meaning of bit 0 / bit 2
0	0	Path not current	Subscriber not reachable
0	1	Path not current	Subscriber reachable
1	0	Path current	Subscriber not reachable
1	1	Path current	Subscriber reachable

Example of coding options for the path status

The following table shows an example of the coding options of the path status for the TIM 1531 IRC.

The abbreviations for the interfaces have the following meaning:

- IE1 = Ethernet interface 1 (X1)
- IE2 = Ethernet interface 2 (X2)
- IE3 = Ethernet interface 3 (X3)
- WAN = serial interface (X4)

Same coding of the configured interface for the main and the substitute path means that there is no path redundancy (only 1 interface configured). The path status is output via the bits of the main path (1st path).

Table 1-124 Coding example for the path status.

Configured interface		Path status	
Coding for substitute path	Coding for main path	Substitute path (2nd path)	Main path (1st path)
0 0	0 0 = Coding for IE1	Irrelevant (not redundant)	Status IE1
0 0	0 1 = Coding for IE2	Status IE1	Status IE2
0 0	1 0 = Coding for IE3	Status IE1	Status IE3
0 0	1 1 = Coding for WAN	Status IE1	Status WAN
0 1	0 0	Status IE2	Status IE1
0 1	0 1	Irrelevant (not redundant)	Status IE2
0 1	1 0	Status IE2	Status IE3
0 1	1 1	Status IE2	Status WAN
1 0	0 0	Status IE3	Status IE1
1 0	0 1	Status IE3	Status IE2
1 0	1 0	Irrelevant (not redundant)	Status IE3
1 0	1 1	Status IE3	Status WAN
1 1	0 0	Status WAN	Status IE1
1 1	0 1	Status WAN	Status IE2
1 1	1 0	Status WAN	Status IE3
1 1	1 1	Irrelevant (not redundant)	Status WAN

Operator input monitoring

Operator input monitoring

The Operator input monitoring is used to monitor operator input (buttons etc.) to protect against incorrect input.

For monitored operator input a minimum and maximum duration is specified. The monitoring status is saved in the "PLC tag for operator input monitoring" (byte).

The PLC tag can be used for entries via the following data point types:

- Command output (Cmd01B_S)
- Setpoint output (Set01W_S)
- Parameter entries (Par12D_S)

The Operator input monitoring is recommended particularly when commands are entered via digital inputs, for example using buttons connected to them. This also applies to the situation when setpoint and parameter entries are transmitted as the result of a trigger and this triggering is via a digital input, in other words for example once again via a button.

Monitoring incorrect input (input time)

Using Operator input monitoring reduces the risk of incorrect input when the entries are made via digital inputs. The following times and the code bits are relevant only for operator input over digital inputs.

- Min. input time
  For these entries, a minimum input time can be specified, in other words, the button must be pressed for the minimum time. The accidental activation of a button does not then lead to an unwanted transfer. When the minimum input time has elapsed and the button can be released, the "PLC tag for operator input monitoring" shows "entry OK" in bit 0 (see below).
  A time of at least 1 second is recommended.
  If you do not require the parameter, configure 0 (zero).
- Max. input time
  Apart from the minimum duration, a maximum input time can also be set for digital inputs. This allows a button that is sticking or defective digital inputs that supply a permanent 1 signal to be detected in good time. Such errors are written as "input error" to bit 1 of the PLC tag.
  Further hardware entries are not processed as long as the "input error" bit is set.
  If you do not require the parameter, configure 0 (zero).

Monitoring for 1 out of n errors.

Generally for all operator input via digital inputs. bit memory or data blocks the PLC tag returns the error status "1 out of n error" in bit 2. The error status is set when more than 1 bit is set at the same time.

The error is output in the following situations:

- 1-out-of-n error with "Cmd01B_S"
  More than 1 bit was set in the input byte of the command data point type "Cmd01B_S". To increase reliability of command input, only 1 bit may ever be set with this object. If two or more bits are set at the same time, the command input is rejected.
- 1-out-of-n errors with all command, setpoint and parameter entries in the fast read cycle
  If increased reliability is required for the input of commands, setpoints and parameters, all objects with which this data is sent should be assigned to the fast read cycle. All command, setpoint and parameter objects in the fast cycle are then subjected to a 1-out-of-n check. At the end of the fast cycle there is a check to make sure that there is a command, setpoint or parameter entry for only one of the acquired objects. Only then is the corresponding entry processed and transferred. If there is more than one entry, the entries are rejected. A new command, setpoint or parameter is processed only when previously no entry was acquired in at least one fast cycle.

Note

Resetting the command bit

If commands are entered over a memory or data byte, or a setpoint or parameter entry is enabled by a memory or data bit (trigger signal), the set command bit or trigger signal is automatically reset to zero.

If, however, a 1-out-of-n error is detected, the command bits are not automatically reset. They must then be reset by the user program.

Assignment of the status byte of "PLC tag for operator input monitoring"

Table 1-125 Bit assignment of the status byte

Bit:	.7 *	.6 *	.5 *	.4 *	.3 *	.2	.1	.0
Status:	-	-	-	-	-	1-out-of-n error	Input error	Input OK
For value:	0	0	0	0	0	1	1	1
* Unused bits are set to 0								

Protection

Protection of the TIM 1531 IRC

Protection

Using the "Protection" table you can configure various access levels to restrict access to the configuration data of the module.

The green check mark in the "Access" column indicates which operations are released without knowing the password of this access level.

If you want to use operations that are not released, you will need to enter the configured password.

Exception
Even without entering a password, each access level allows access to the identification data (name, addresses ...) of the module using the "Accessible devices" function.

Default reaction

The default access level is "Full access (no protection)". Every user can read and modify the configuration data of the TIM. No password has been configured and no password is required for online access.

The individual access levels

You can configure the following access levels:

- Full access (no protection)
  The hardware configuration and the blocks can be read and modified by anybody.
- Read access
  Without entering the password, only read access is possible.
  Without entering the password, you cannot load any data (configuration, firmware, certificates) on the module and not perform any writing test functions.
- No access (complete protection)
  Neither read nor write access to the module is possible.

With the legitimization provided by using the password, you once again have full access to the module.

Behavior of a password-protected module during operation

Protection of the module is effective after the settings have been loaded on the module.

The functions protected by the password can only be executed by the engineering station (STEP 7 PC).

Before an online function is executed, a check is made to establish whether or not it is permitted. If there is password protection, you will be prompted to enter the password.

Example:
The module was configured for read access and you want to load new firmware. Since this is write access, the configured password must be entered before the function can be executed.

The access rights to the protected data apply for the duration of the online connection or until the access rights are canceled again manually with "Online > Delete access rights".

See also

Configuring access protection (Page 1726)

Configuring access protection

For the module you can enter several passwords and therefore set up different access rights for different user groups.

The passwords are entered in the table so that precisely one access level is assigned to each password. How the password takes effect is shown in the "Access level" column.

Example (No access):

You select the access level "No access (complete protection)" for the module and enter a separate password for each of the access levels higher up the table.

For users that do not know any of the passwords, the module is completely protected.

For users who know one of the passwords the effect depends on the table row in which the password is located:

- The effect of the password in the row "Full access (no protection)" is that the module is unprotected for users that know this password. They have unrestricted access to the module.
- The password in the row "Read access" brings about write protection for the module. Users that know this password only have read access to the module.

Procedure

Follow the steps below to configure the access levels in the "Protection" table:

1. Select the required access level in the first column of the table. The green check mark in the columns on the right of the particular access level indicate which operations are still possible without entering the password.
2. If you have selected an access level other than "Full access":
   - Assign a password for full access in the "Password" cell in the first row (Full access).
   - Repeat the selected password to protect against incorrect entries.

   Make sure that the password is adequately secure; in other words, that it does not include a pattern that can be machine read!
   The entry of the password in the row "Full access (no protection)" is obligatory and allows a user who knows the password unrestricted access to the module regardless of the selected protection level.
3. As necessary, assign other passwords to the required access levels if the selected access level permits this.
4. Download the configuration data to the module so that the access level takes effect.

Result

The configuration data is protected from unauthorized online access according to the set access levels. If an operation cannot be executed without a password due to the configured access level, a dialog appears prompting entry of the password.

Telecontrol connections

Telecontrol connections

Telecontrol connections

Telecontrol relationships between the communication modules involved are required for telecontrol communication. Depending on the module type and the firmware version, you perform the configuration in the following parameter groups:

- "Partner stations" parameter group
  or
- "Network data" editor

Configuration in the "Partner stations" parameter group

For S7-1200 telecontrol CPs up to V2.1, which only function as stations, the relationships to the master station or to the master are configured in the "Partner stations" parameter group. This is where you define the communication partners of the CPs.

All other settings required for communication with the master station are taken from the other configuration data of the CPs and do not have to be specially configured for the connections.

Note

Exception: CP 1243-8 IRC V2.1 (ST7)

As of STEP 7 V15.1, you can configure a physical CP with firmware V2.1 in the STEP 7 project with firmware version V3.0:

1. Insert a CP with firmware version V3.0 into the STEP 7 project.
2. Select the "Compatibility mode V2.1" option in the "General > Catalog Information" parameter group of the CP.

You can then use the CP in telecontrol connections that you configure using the "Network data" editor.

You can load the V3.0 configuration data into CP V2.1.

The following functions are not supported when this option is selected:

- Switching the protocol type
- Receive time from local station and via NTP
- "Communication with the CPU > CP diagnostics"
- Online & Diagnostics > "Functions > Save service data"

Configuration in the "Network data" editor

For all other telecontrol modules that do not fall into the above group, you need to configure telecontrol connections in the "Network data" editor.

The "Network data" editor is used for the following protocols and modules:

- ST7
  Modules without imported configuration data:
  - CP 1243-8 IRC as of firmware V3.0
  - CP 1542SP-1 IRC as of firmware V2.0
  - TIM 1531 IRC as of firmware V1.0
  - TIM 3V-IE / TIM 4R-IE as of firmware version V2.7
- DNP3
  - CP 1243-1 / CP 1243-8 IRC as of firmware V3.0
  - CP 1542SP-1 IRC as of firmware V2.0
  - TIM 1531 IRC as of firmware V2.0
  - TIM 3V-IE DNP3 / TIM 4R-IE DNP3 as of firmware V3.2
- IEC 60870-5
  - CP 1243-1 / CP 1243-8 IRC as of firmware V3.0
  - CP 1542SP-1 IRC as of firmware V2.0
  - TIM 1531 IRC as of firmware V2.0

Creating telecontrol connections

Opening the editor "Network data" > "Telecontrol" tab

To open the editor, follow the steps below:

1. Open the network view of the project.
   On the right you will find the collapsed "Network data" editor.

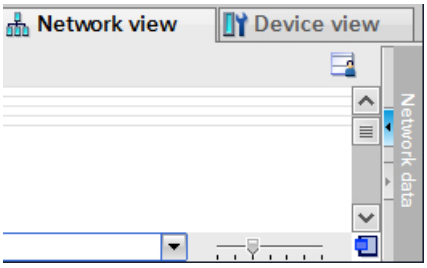

2. Open the "Network data" editor using the arrow symbol.
   The editor is displayed with several tabs, on the left the "Network overview" tab.
3. Expand the editor until the "TeleControl" tab appears.
   This tab is further divided into the following tabs:
   - ST7
   - DNP3
   - IEC

   Depending on the protocol used, select the corresponding tab to configure the telecontrol connections.

Display and show/hide columns

In the "Telecontrol connections" table, you can display or hide the columns, arrange them and optimize the column width. Right-click on a column header to access the shortcut menu.

- Arrange columns
  If you click on a column header and hold down the left mouse button, you can move the column within the table.
- Show/Hide
  You can show or hide individual columns using this function in the shortcut menu.
  This increases the legibility of the table.
- Show all columns
  Shows all columns of the table.
- Optimize width / Optimize width of all columns
  You use these shortcut menus to optimize the width of the selected column or all columns in the table.
  The column width adapts to the widest entry in this column.

Creating telecontrol connections

To create a telecontrol connection, follow these steps:

1. In the "Telecontrol connections" table, double-click in a field in the first free row.
   A new empty connection is created. The connection name is preset.
2. Select the starting point of the connection from the drop-down list in the "Starting point" field.
   The drop-down list of the field shows the possible starting points.
3. To change the connection name, click in the "Name" field.
   See also the following section.
4. In the other fields, configure the required parameters, such as interfaces and endpoint of the connection.

Some fields of the table can be edited, in others you configure the parameter via a drop-down list.

Boxes with a missing or bad configuration are shown on a red background, in the example the representation "TC_Connection_3".

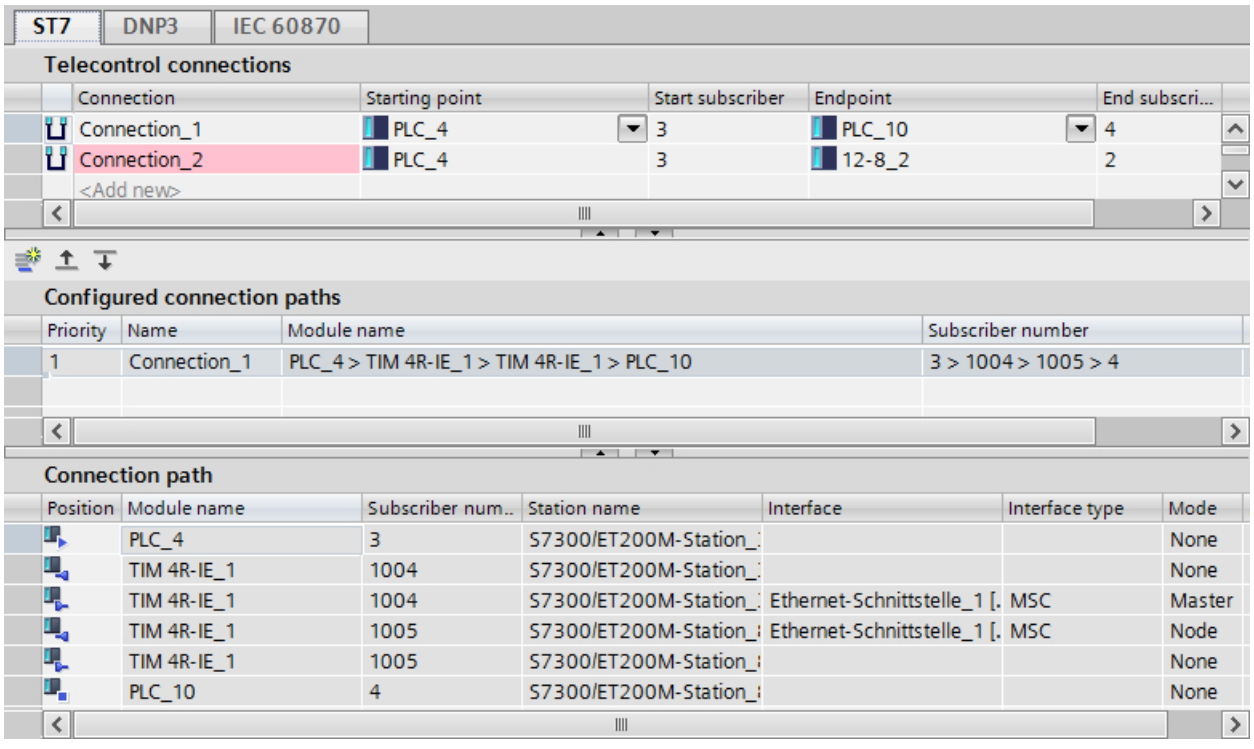

Figure 1-72 "Network data" editor, "Telecontrol > ST7" tab, example of ST7 connections

The connections are displayed somewhat differently in the "DNP3" and "IEC" tabs, see protocol-specific sections.

Names of the connections

You can adapt the default names of the connections.

A maximum of 129 characters from the following ASCII character sets (numbers decimal) are permitted:

- **No. 32..126**
  Space , ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~
- **No. 128, 130..140, 142, 145..156, 158..159, 161..172**
  € , ƒ „ … † ‡ ˆ ‰ Š ‹ Œ Ž ‘ ’ “ ” • – — ˜ ™ š › œ ž Ÿ ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬
- **No. 174..255**
  ® ¯ ° ± ² ³ ´ µ ¶ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ

Specification of the connection path

Once a connection has been created, the actual course of the connection is usually not yet defined. Especially with larger networks, several connection paths are often possible.

Specify the actual connection path of the created telecontrol connection via the icon "Add new connection path" . The "Add connection paths" dialog opens.

You can find more details in the following protocol-specific chapters of the telecontrol connections.

Symbols in the "Add connection paths" dialog

The "Connection path" table supports you in checking the configured connections. For every configured connection, the detailed connection path is shown here.

A station symbol with an identifier for the connection point is displayed in the "Position" column:

The color of the identifier indicates the validity of the connection point:

- Blue: Valid connection point
- Red: Invalid connection point

Symbol	Meaning
	Starting point
	Node - input
	Node - output
	Endpoint
	Examples of invalid connection points

Error displays

Faulty connection points, networks or parameters are highlighted in red in the tables.

Causes of faulty connections include, for example:

- Starting point and endpoint are identical.
- The connection runs through an impermissible network.
- The connection runs through an invalid subscriber.

Example of representation:

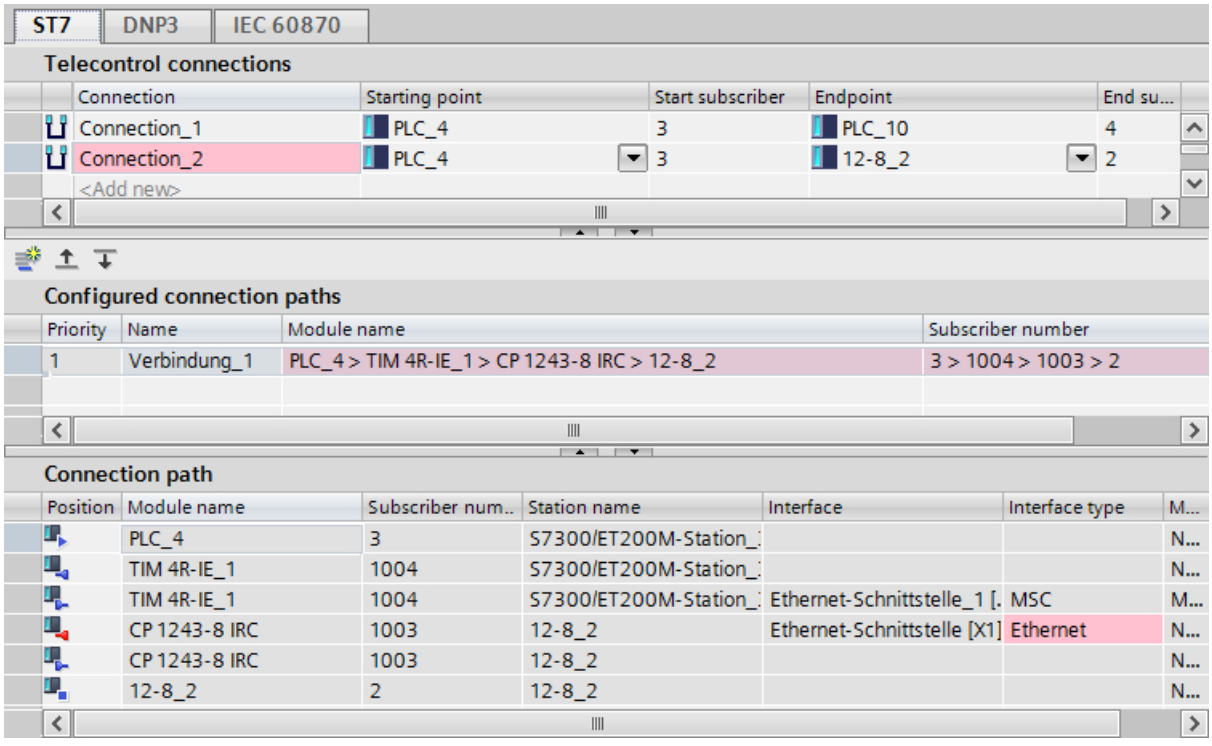

Figure 1-73 Redundant connection with incorrectly configured connection path (Route_2)

The "Route_2" path of an ST7 connection shown in the figure was created via an invalid MSC network. The connection of the PC station to the MSC network was then deleted, so that the telecontrol connection runs over subscribers that are no longer available.

General rules for connection configuration

Note the following general rules for connection configuration:

- A connection must run over a single subnet.
  If a "third-party device" is located as a partner in another subnet, the gateway between the subnets is the endpoint for connection configuration.
- All sections of a telecontrol connection must be configured using the same telecontrol protocol.
- A connection via an inconsistent network is invalid.
  Examples:
  - Subscribers with incompatible network node types (e.g.: 2 masters)
  - Connections via nodes that are not configured as node stations.
  - Subscriber with incompatible modems
  - Incompatible settings of two modems in a connection
  - Incompatible settings between modem and network parameters
  - Two connections to a partner via the same interface of a module are not allowed.
  - A connection via a network in which two analog switched network modems (e.g. MD3) communicate with each other via 1200 bps / half-duplex / AT mode is not permitted.

You can find further protocol-specific rules in the following sections.

Delete invalid or redundant connections

If there are unauthorized or unwanted redundant connections, you must delete a connection path:

1. In the "Configured connection paths" table, select the unwanted connection path.
2. Click "Delete" in the shortcut menu.

"Properties" tab of the connection

If you select a connection in the "Telecontrol connections" table in the "Network data" editor, additional parameter groups for this connection are displayed in the "Properties" tab of the Inspector window. Here you can check and, if necessary, correct the connection configuration and configure additional properties.

Parameter groups:

- General
  Shows the basic properties of the telecontrol connection.
  You can change the connection name and the subscriber number / station address of the starting point and endpoint. This affects the configuration data of the subscriber.
- Other protocol-specific parameters
  You can find the description below.

ST7 connections

ST7-specific rules for connection configuration

Note the following special rules for connection configuration:

- An MSC connection can only run via the first Ethernet interface [X1] of a TIM or via the serial interface of a module with the setting "IP-based".
- Connections between a PC application and a module can only be of the Network type "Neutral" (S7 connections).
- For connections between a redundant PC application and a module, you need to create two connections.
  For a redundant ST7cc control center, in STEP 7 create two PC stations each with an ST7cc application (see below).
  Only in this way can you configure a connection from each application.

Create ST7 connections

For communication between two ST7 stations via classic WAN networks (dedicated line, dial-up network) you need at least one telecontrol connection.

The following configurable starting points are displayed in the "Starting point" field:

- S7 stations: Assigned CPU
- PC stations: Application (e.g. ST7cc / ST7sc)

For connections between an application on the master station PC and an S7 station, always select the application as the starting point because it can only be selected in the list of starting points.

After clicking on the "Add new connection path" icon ( ), the "Add connection paths" dialog opens, where you can search for the possible connection paths. The connection paths found are displayed in the "Select a connection path ..." table.

For ST7 connections, the "Configured connection paths" table is located below the "Telecontrol connections" table. This contains the added connection path for each selected connection.

For redundant connections, both paths are displayed.

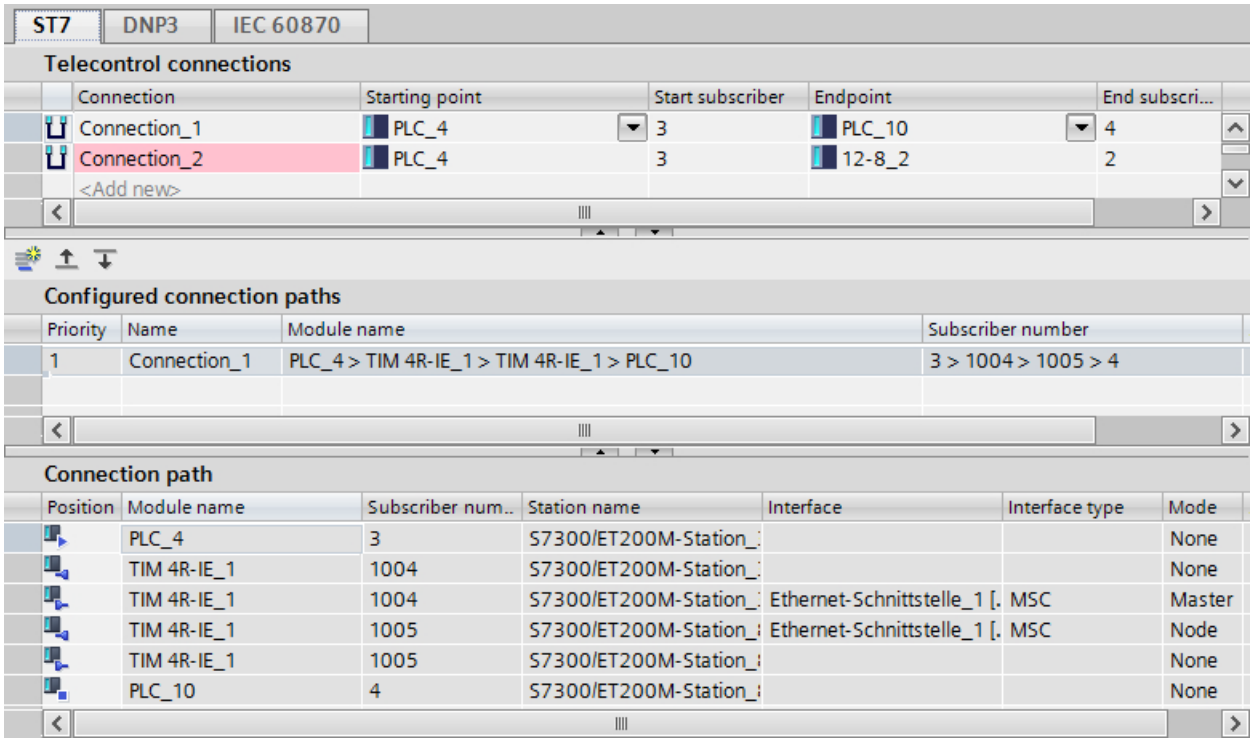

Figure 1-74 "Network data" editor, "Telecontrol" tab

The third table "Connection path" shows details of the selected path.

Specification of the connection path

Specify the actual connection path of the created telecontrol connection as follows.

1. Select a connection in the "Telecontrol connections" table.
2. Click on the "Add new connection path" icon.
   The "Add connection paths" dialog opens.
   The possible connection paths are searched for automatically, which is indicated by the progress bar at the bottom of the dialog. The status and result of the search are displayed in the "Information" field below.
   The connection paths found are displayed in the "Select a connection path ..." table.
3. Select one of the possible connection paths.
   Details about the selected connection path are shown in the "Connection path" table.
   The "Preview" table contains additional details about all connection points involved in the selected connection path.
   - If several connections are displayed in the "Select a connection path ..." table, you can choose one or two redundant connections for selecting the connection path to be used.
   - If no connection is displayed in the table, there is a configuration error in the corresponding stations or networks.
     In this case, close the dialog with the "Close" button and complete the configuration.

4. Select a connection path and click "Add" to add this path to the project as a configured connection.
   "Information" shows whether the connection has been added or whether it has already been configured.
5. If necessary, select a second connection path and click "Add" to add this path as a redundantly configured connection.
6. Close the dialog using the "Close" button if the added connection paths correspond to the project defaults.

Creating redundant ST7 connections

If different networks are created between two subscribers in the network configuration, the different connection paths are displayed:

> dialog "Add connection paths" > table "Select a connection path ..."

If you only want to use one connection path, only select the desired path and click "Add".

If you want to use two redundant connection paths for the ST7 connection, select both one after the other and click "Add" for each.

After closing the "Add connection paths" dialog, both paths for this connection are displayed in the "Configured connections" table.

The first of the two is the main path and the second the alternative path. This is displayed in the "Priority" column of the "Configured connections" table:

1. Main path
2. Substitute path

You can change the priority of the two connection paths (main/alternative path) later. The arrow keys serve this purpose:

"Properties" tab of the connections

For the "General" parameter group, see section Creating telecontrol connections (Page 1729).

- **Redundant ST7cc/sc**
  If you use a redundant control center in your project (e.g. ST7cc or ST7sc), you must first create two PC stations and the individual connections between the module and the two applications.
  Only afterwards you create a redundancy group here for the connection to the redundancy group.
  - Add redundant ST7cc/cc partner
  - Enable the option for each of the two connections between the redundant application and the module.
  - Redundant application
    Select the required application from the "Redundant application" drop-down list.
- **Destination subscriber properties**
  The parameter group relates to the destination subscriber that was configured as the endpoint of the connection.
  - **General request supervision time**
    This is the maximum time that may be required by a destination subscriber (communications module capable of ST7) to respond fully to a general request (GR). If the GR response has not arrived completely at the requesting partner when the supervision time has expired, a message is entered in the diagnostics buffer of the module.
    The general request supervision time is 900 seconds.
  - **Frames to destination subscriber with time stamp**
    Regardless of the configuration, all frames are sent to the destination subscriber with a time stamp.

DNP3 and IEC connections

Communications options

Communication paths

The following communication channels are possible:

- Connections between master and station
- Redundant connections
  If two subscribers can be reached via different paths or networks, you can create a maximum of two connections between the subscribers to establish path redundancy.
- Direct communication
  With direct communication, stations communicate directly with each other without the frames being transmitted from a master station or station. Two stations can thus communicate directly with each other.
  You define the function when configuring a data point via the "Master function" parameter. As a further requirement, the "Unsolicited reporting" option must be selected in the telecontrol connection.
  The function is supported by the following modules:
  - S7-1200 telecontrol CPs as of firmware V3.1
  - CP 1542SP-1 IRC as of firmware V2.0
  - TIM 1531 IRC as of firmware V2.0
  - TIM 3V-IE DNP3 / TIM 4R-IE DNP3 as of firmware V3.2

Configure connections

Protocol-specific rules for connection configuration

Note the following special rules for connection configuration:

- You can create connections between subscribers that are configured in the STEP 7 project. The address and interface data of both subscribers is available in the connection table.
- You can create connections between a subscriber that is configured in STEP 7 and a "third-party device" that is not configured in STEP 7.
  The device that is not configured in STEP 7 is displayed in the "Endpoint" field of the connection table as a "third-party device". You configure the address data of this subscriber using the real address data (IP address / telephone number).
- You can configure single and redundant connections to subscribers in other subnets that can be reached via a gateway. The corresponding fields in the connection table are available for this purpose.

Definition of the single or redundant connection path

Specify the actual connection path of the created telecontrol connection as follows.

1. Select a connection in the "Telecontrol connections" table.
2. Click on the "Add new connection path" icon.
   The "Add connection paths" dialog opens.
   The possible connection paths are searched for automatically, which is indicated by the progress bar at the bottom of the dialog.
   - The status and result of the search are displayed in the "Information" field below.
   - The connection paths found are displayed in the "Select a connection path ..." upper table.
   - Details for a selected connection path are shown in the "Connection path" table.
   - When selecting a connection path, the "Preview" table shows which connection points of the selected connection path are transferred to the connection editor when you click "Add".
3. Select the desired connection path(s).
   - If one or more connection paths are displayed in the upper table, select the desired connection path and click on "Add".
     "Information" shows whether the connection path has been added or whether it has already been configured.
     If you want to use a redundant connection, select a second path and click on "Add".
     Close the dialog using the "Close" button if the added connection paths correspond to the project defaults.
   - If no connection is displayed in the table, there is a configuration error in the corresponding stations or networks.
     In this case, close the dialog with the "Close" button and complete the configuration.

Interface-specific configuration of the connection segments

ST7 | DNP3 | IEC 60870

Telecontrol-Verbindungen

Connection	Starting point	Start subscriber	Start interface	Endpoint	End subscriber	Partner list
*	15_	*		*	*	
_1	15_4	1	TIM_4 - Ethernet..	15_6	3	3
_3	15_6	3	TIM_2 - Ethernet.	15_4	1	1
_5	15_4	1	TIM_4 - Ethernet..	15_3_node	2	2,3
_6	15_6	3	TIM_2 - Ethe..	15_3_n...	2	1,2
_7	15_3_node	2	TIM_3_node - Et..	15_6	3	3

Figure 1-75 "Network Data" editor, "Telecontrol > IEC..." tab, example of connection segments

The filter "15_" is set in the "Starting point" column. The meaning is described in the respective connection table (see below).

Note

Section for multiple connections

A connection segment between two subscribers used for multiple connections only appears once in the connection table.

DNP3 and IEC connections can run over multiple subscribers. A connection segment between two subscribers can also be used for multiple connections.

Since you can configure individual settings for the individual connection segments, the connection segments between the interfaces of two devices are displayed in separate lines.

In the figure, the "15_3_node" station is a node station that is located between the master station (15_4) and the station (15_6) in the plant hierarchy. The special features of node stations are described in the respective connection table (see below) for the "Partner list" parameter.

Parameters of the DNP3 connections

You can find the description of the parameters in the section Connection table - DNP3 (Page 1740).

Parameters of IEC connections

You can find the description of the parameters in the section Connection table - IEC (Page 1745).

"Properties" tab of the connection

- **General**
  See section Creating telecontrol connections (Page 1729).
- **DNP3 security options**
  See section DNP3 security options (Page 1814).
- **Options 1st Path / Options 2nd Path**
  See section Options (IEC) (Page 1749).
- **Third-party device parameters**
  See section Third-party device parameters (Page 1749).

Connection table - DNP3

Filter

The first row below the table header contains a filter function with which you can restrict the selection of configurable subscribers and connection options. Using filters reduces the number of combination possibilities and increases the clarity.

Once you have created some connection segments, enable the filter by entering a recurring name or partial name in the filter cell. The cell is given a colored background, see figure in section Configure connections (Page 1738).

Example:
You have created connections with the starting points "1200", "PLC_1" and "PLC_2".
When "PLC_" is entered in the filter cell, only the segments whose starting points start with this partial string are shown: "PLC_1" and "PLC_2"

Filters set in multiple columns multiply.

The selection "*" shows all existing connection segments.

The filter can be applied to the following columns:

- Connection name
- Starting point / Start subscriber
- Endpoint / End subscriber

Parameters

If parameters are already used by the configuration, the values are transferred to the respective columns.

- **Name**
  You can change the default name of the connection segment between two subscribers.
  See section Creating telecontrol connections (Page 1729) for information on this.
- **Starting point**
  Select the desired starting point of the connection from the drop-down list.
  - The starting point of a connection is a CPU.
- **Start subscriber**
  Station address of the starting point
- **Start interface**
  Interface of the starting point module through which the connection runs.
- **Start interface (red.)**
  Interface of the starting point module through which the redundant connection runs.
- **Endpoint**
  Select the endpoint of the connection.
  Endpoints of a connection can be:
  - A CPU
  - A third-party device
    The network node type of third-party devices is configured in the parameter group of the connection, see section Third-party device parameters (Page 1749).
- **End subscriber**
  Station address of the endpoint
- **End interface**
  Interface of the endpoint module through which the connection runs.
  For a third-party device that is not configured in the STEP 7 project, you must enter the IP address manually.

- **End interface (red.)**
  Interface of the endpoint module through which the redundant connection runs.
- **Partner list**
  When a partner (endpoint) is selected that is located in the STEP 7 project, its station address is automatically entered in the Partner list.

Note

Manual entries

You must manually enter the station addresses in the following situations:

- **Third-party device**
  For a third-party device that is not configured in the STEP 7 project, you must enter the station address manually.
- **Node station**
  The Partner list of a connection segment that transfers data from or to additional stations must contain all partners. This is the case for connection segments in which the subscriber is a node station. Here you must enter the station addresses of the additional partners manually.
  The station addresses are entered separated by commas.

For an example of a node station, see also the figure in section AUTOHOTSPOT. You configure three stations:

– Master
  Name: "15_4", station address: "1"

– Node station
  Name: "15_3_node", station address: "2"

– Station
  Name: "15_6", station address: "3"

The connection segment "_5" runs from the master to the node station. The station address "2" of the node station is automatically entered in the partner list.
Because the data from and to the station "15_6" is also supposed to be transferred over the connection segment, you must enter the station address "3" manually, see figure.
The same is true for the connection segment "_6" between station and node station over which the data is also to be transferred from and to the master.

- **Port no.**
  Relevant for third-party device (master / station)
  Number of the listener port of the partner
  For modules of the STEP 7 project, the value is taken from the configuration. It can be changed.
  You need to enter the port number for a third-party device.
  Range of values: 0 ... 65535
  Default: 20000

- **Partner monitoring**
  Relevant for all subscriber types
  If the station module does not receive a sign of life from the master on the application layer within the configured time, it classifies the connection as disrupted and terminates it.
  The master module expects a response from the station after sending data within the configured time.
  With the setting 0 (zero), the function is disabled.
  See "Partner monitoring time" parameter, section Transmission settings – DNP3 (Page 1808).
- **Transport prot.**
  Relevant for all subscriber types
  Select the transport protocol:
  - TCP
  - UDP
- **Spontaneous**
  Transfer mode for events
  For modules of the STEP 7 project, the value is taken from the configuration and cannot be changed.
  For information on the parameter, see section Settings DNP3 station (Page 1811).
- **Event polling**
  Relevant for master, third-party device (master)
  The "Event polling interval" defines the cycle in which events of the station are queried by the DNP3 master station. The interval is specified as a multiple of the master's "Polling basic interval", see section Settings DNP3 master (Page 1811).
- **Class 0 polling interval**
  Relevant for master, third-party device (master)
  The class 0 polling interval determines the cycle in which class 0 data from the image memory of the station is polled by general requests from the DNP3 master station. The interval is specified as a multiple of the parameter "Polling basic interval" of the master, see section Settings DNP3 master (Page 1811).
  The value configured for the station is transferred to the master station and stored there.
  Range of values: 0 ... 65535
  Default: 1
  With the setting 0 (zero), the function is switched off, class 0 data is not transmitted cyclically.
- **Max. polling duration**
  Relevant for master, third-party device (master)
  Specifies the maximum time period during which the master may continuously call this station. Even if data is still pending for transmission in the station after this time, the calls from the master station are canceled. This means that the master station is once again available to other stations.
  The value configured for the station is transferred to the master station and stored there.
  Range of values: 0 ... 65535
  Default: 10
  With the setting 0 (zero), the function is disabled; in other words, the calling period is unlimited.

- Polling mode
  Relevant for master, third-party device (master)
  This is where you define the mode in which the master station calls the station.
  The value configured for the station is transferred to the master station and stored there.
  Ranges of values:
  - Cyclic
    The station is called cyclically. The duration of the polling cycle is calculated from the "Class 0 polling interval" parameter, see above.
  - After startup
    The station is only called after the initial startup and after a restart.
    If no unsolicited transmission is enabled for a station, no data is transmitted during operation when this option is selected.
- Conformity level
  Relevant for all subscriber types
  Specifies the DNP3 conformity level (DNP3 implementation level) supported by the partner: Level 1, 2, 3, 4 and Level 4+
  The conformity level referred to as Level 4+ contains the functional scope of Level 4 and also the support of additional data types / variations; see also data point types in the section Datapoint types (Page 1758).
- Temporary
  Relevant for partners with temporary connections (e.g. RTU)
  Select this option if the partner supports temporary connections that it can terminate itself. The connection termination by the partner is then not interpreted as a connection termination.
  For example, the partner might be an RTU3000C on which the "Temporary connection" option is selected.
- Polling basic interval
  Relevant for master, third-party device (master)
  Base interval for the duration of the polling cycle of the central station (see "Polling mode" above) and for calculating the "Event polling interval" and "Class 0 polling interval" parameters.
  For information on the parameter, see section Settings DNP3 master (Page 1811).
- Max. number of events per query
  Relevant for master, third-party device (master)
  Maximum number of events that can be sent in the response frame of the station after being called by the master station. (0 means no limitation)
  For information on the parameter, see section Settings DNP3 master (Page 1811).

- **Form of transfer**
  Relevant for station, third-party device (station)
  Determines the send sequence of events to the DNP3 master station.
  - Type-specific
    Bundled transmission according to data type
  - Chronological
    Chronological transmission by time stamp

  For information on the parameter, see section Transmission settings – DNP3 (Page 1808).
- **Partner address**
  Relevant for third-party device (master / station)
  IP address (Ethernet) or telephone number (dial-up network) of the partner

Parameters for redundant connection paths

If redundant connection paths are configured, these are configured in the same way as the main paths.

The parameters of the redundant connection paths are distinguished by the following suffix:

- *** (red)**

The parameters of the redundant connection paths have the same functions as those for the main path. See above for the meaning.

Connection table - IEC

Filter

The first row below the table header contains a filter function with which you can restrict the selection of configurable subscribers and connection options. Using filters reduces the number of combination possibilities and increases the clarity.

Once you have created some connection segments, enable the filter by entering a recurring name or partial name in the filter cell. The cell is given a colored background, see figure in section Configure connections (Page 1738).

Example:
You have created connections with the starting points "1200", "PLC_1" and "PLC_2".
When "PLC_" is entered in the filter cell, only the segments whose starting points start with this partial string are shown: "PLC_1" and "PLC_2"

Filters set in multiple columns multiply.

The selection "*" shows all existing connection segments.

The filter can be applied to the following columns:

- Connection name
- Starting point / Start subscriber
- Endpoint / End subscriber

Parameters

If parameters are already used by the configuration, the values are transferred to the respective columns.

- **Name**
  You can change the default name of the connection segment between two subscribers. See section Creating telecontrol connections (Page 1729) for information on this.
- **Starting point**
  Select the desired starting point of the connection from the drop-down list.
  - The starting point of a connection is a CPU.
- **Start subscriber**
  Station address of the starting point
- **Start interface**
  Interface of the starting point module through which the connection runs.
- **Endpoint**
  Select the endpoint of the connection.
  Endpoints of a connection can be:
  - A CPU
  - A third-party device
    The network node type of third-party devices is configured in the parameter group of the connection, see section Third-party device parameters (Page 1749).
- **End subscriber**
  Station address of the endpoint
- **End interface**
  Interface of the endpoint module through which the connection runs.
  For a third-party device that is not configured in the STEP 7 project, you must enter the IP address manually.

- **Partner list**
  When a partner (endpoint) is selected that is located in the STEP 7 project, its station address is automatically entered in the Partner list.

  Note

  Manual entries

  You must manually enter the station addresses in the following situations:

  - **Third-party device**
    For a third-party device that is not configured in the STEP 7 project, you must enter the station address manually.
  - **Node station**
    The Partner list of a connection segment that transfers data from or to additional stations must contain all partners. This is the case for connection segments in which the subscriber is a node station. Here you must enter the station addresses of the additional partners manually.
    The station addresses are entered separated by commas.

  For an example of a node station, see also the figure in section AUTOHOTSPOT. You configure three stations:

  – Master
    Name: "15_4", station address: "1"

  – Node station
    Name: "15_3_node", station address: "2"

  – Station
    Name: "15_6", station address: "3"

  The connection segment "_5" runs from the master to the node station. The station address "2" of the node station is automatically entered in the partner list.
  Because the data from and to the station "15_6" is also supposed to be transferred over the connection segment, you must enter the station address "3" manually, see figure.
  The same is true for the connection segment "_6" between station and node station over which the data is also to be transferred from and to the master.

- **Port no.**
  Relevant for third-party device (master / station)
  Number of the listener port of the partner
  For modules of the STEP 7 project, the value is taken from the configuration. It can be changed.
  You need to enter the port number for a third-party device.
  Range of values: 0 ... 65535
  Default: 2404

- **Partner monitoring**
  Relevant for all subscriber types
  If the station module does not receive a sign of life from the master on the application layer within the configured time, it classifies the connection as disrupted and terminates it.
  The master module expects a response from the station after sending data within the configured time.
  With the setting 0 (zero), the function is disabled.
  See "Partner monitoring time" parameter, section IEC-specific parameters (Page 1818).

- **Spontaneous**
  Transfer mode for events
  For modules of the STEP 7 project, the value is taken from the configuration and cannot be changed.
  For information on the parameter, see section IEC-specific parameters (Page 1818).
- **Polling mode**
  Relevant for master, third-party device (master)
  This is where you define the mode in which the master station calls the station.
  The value configured for the station is transferred to the master station and stored there.
  Ranges of values:
  - Cyclic
    The station is called cyclically. The duration of the polling cycle is calculated from the "Class 0 polling interval" parameter, see above.
  - After startup
    The station is only called after the initial startup and after a restart.
    If no unsolicited transmission is enabled for a station, no data is transmitted during operation when this option is selected.
- **Temporary**
  Relevant for partners with temporary connections (e.g. RTU)
  Select this option if the partner supports temporary connections that it can terminate itself. The connection termination by the partner is then not interpreted as a connection termination.
  For example, the partner might be an RTU3000C on which the "Temporary connection" option is selected.

Parameters for redundant connection paths

If redundant connection paths are configured, these are configured in the same way as the main paths.

The parameters of the redundant connection paths are distinguished by the following suffix:

- *** (red)**

The parameters of the redundant connection paths have the same functions as those for the main path. See above for the meaning.

Third-party device parameters

Third-party device parameters

Only valid for partners that are not configured in the STEP 7 project.

- **Partner station address /**
  Station address (red.)
  Station address (ASDU address) of the third-party device, that is accessible via a connection or a redundant connection path.
- **Network node type third-party device /**
  Network node type third-party device (red.)
  Define the network node type of the third-party device that is accessible via a connection or a redundant connection path:
  - Master station
    (Master)
  - Node station
    For modules that act as a node station, the following applies:
    The interface in the direction of the master station is configured as a "node station".
    The interface in the direction of the lower-level network is configured as a "Master station".
  - Station

Options (IEC)

The following parameters can be found in the parameter groups "Options 1st Path" / "Options 2nd Path" of the IEC connections.

Call intervals

The following parameters define the intervals of special calls of the master for the station (cause of transmission 20 - 41).

All parameters are configured as multiples of the "polling basic interval" (see master).

- **Interval for general request**
  Defines the interval with which general requests of the master are answered.
- **Interval for group request**
  Defines the interval with which the respective group request of the master is answered.
- **Interval for counter general request**
  Defines the interval with which counter general requests of the master are answered.
- **Interval for counter group request**
  Defines the interval with which the respective counter group request of the master is answered.

You define the setting as to whether a general request is answered and the assignment to a group request for each individual data point in the data point configuration.

Data point editor

Configuring data points and messages

Preliminary overview: TIM modules with SINAUT ST7

TIM modules using the SINAUT ST7 telecontrol protocol can use the "Telecontrol ST7" (TD7onCPU) block library as an alternative to the data points (TD7onTIM).

This section applies to "TD7onTIM".

Data point-related communication with the CPU

No program blocks need to be programmed for the telecontrol modules with data point configuration to transfer user data between the station and communications partner. The data areas in the memory of the CPU intended for communication with the communications partner are configured data point-related on the module. Each data point is linked to a PLC tag or the tag of a data block.

Requirement: Created PLC tags and/or data blocks (DBs)

PLC tags or DBs must first be created on the CPU to allow configuration of the data points.

The PLC tags for data point configuration can be created in the standard tag table or in a user-defined tag table. All PLC tags intended to be used for data point configuration must have the attribute "Visible in HMI".

Address areas of the PLC tags are input, output or bit memory areas on the CPU.

The formats and S7 data types of the PLC tags that are compatible with the protocol-specific data point types of the CP can be found in the section Datapoint types (Page 1758).

Access to the memory areas of the CPU

The values of the PLC tags or DBs referenced by the data points are read and transferred to the communications partner by the CP.

Data received from the communications partner is written by the CP to the CPU via the PLC tags or DBs.

Configuring the data points and messages in STEP 7

You configure the data points in STEP 7 in the data point and message editor. You can open both editors alternatively as follows:

- Selection of the communication module in the network view or device view
  Shortcut menu "Open the data point and messages editor"
- Via the project navigation:
  Project > Directory of the station > Local modules > desired communication module

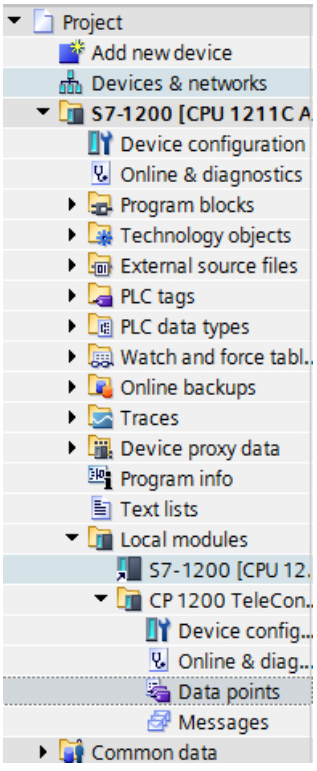

Figure 1-76 Open the data point and messages editor

By double-clicking on the entry, the data point or message editor opens.

After opening the editor window using the two entries to the right above the table, you can switch over between the data point and message editor.

Figure 1-77 Switching over between the two editors

Creating obects

With the data point or message editor open, create a new object (data point / message) by double clicking "<Add object>" in the first table row with the grayed out entry.

A preset name is written in the cell. You can change the name to suit your purposes but it must be unique within the CP.

		Name	PLC tag
1		DataPoint	"Tag_1-BI"
2		DataPoint_1	"Tag_2-BQ"
3		DataPoint_2	"Tag_1-BI"

Figure 1-78 Data point table

You configure the remaining properties of every object using the drop-down lists of the other table columns and using the parameter boxes shown at the bottom of the screen.

Assigning data points to their data source

After creating it, you assign a new data point to its data source. Depending on the data type of the data point a PLC tag can serve as the data source.

For the assignment you have the following options:

- Click on the table symbol in the cell of the "PLC tag" column.
  All configured PLC tags and the tags of the created data blocks are displayed. Select the required data source with the mouse or keyboard.
- Click the symbol ...
  A selection list of the configured PLC Tags and the blocks is displayed. From the relevant table, select the required data source.
- In the name box of the PLC tag, enter part of the name of the required data source.
  All configured PLC tags and tags of the data blocks whose names contain the letters you have entered are displayed.

  Va
  Variable_5
  UInt_Var_6
  DB_Var.Static_1

  Select the required data source.

Note

Assignment of parameter values to PLC tags

The mechanisms described here also apply if you have to assign the value of a parameter to a PLC tag. The input boxes fro the PLC tag (e.g.: PLC tag for partner status support the functions described here for selecting the PLC tag.

Arranging columns and rows, showing/hiding columns

As with many other programs, you can arrange the columns and sort the table according to your needs in the data point or message editor:

- Arrange columns
  If you click on a column header with the left mouse button pressed, you can move the column.
- Sorting objects
  If you click briefly with the left mouse button on a column header, you can sort the objects of the table in ascending or descending order according to the entries in this column. The sorting is indicated by an arrow in the column header.
  After sorting in descending order of a column the sorting can be turned off by clicking on the column header again.
- Adapting the column width
  You can reach this function with the following actions:
  - Using the shortcut menu that opens when you click on a column header with the right mouse button.
    "Optimize width", "Optimize width of all columns"
  - If you move the cursor close to the limit of a column header, the following symbol appears:

    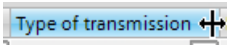

    When it does, double-click immediately on the column header. The column width adapts itself to the broadest entry in this column.
- Showing / hiding columns
  You call this function using the shortcut menu that opens when you click on a column header with the right mouse key.

Copy data points and messages

As with many other programs, you can copy and paste objects in the data point or message editor.

If you right-click in the row of an object in the table, you can access this function from the shortcut menu:

- Cut
- Copy
- Paste
  You can paste cut or copied objects within the table or in the first free row below the table. You can also paste cut or copied objects into tables of other communication modules of the same type and with the same telecontrol protocol.
- Delete

If you hold down the <Ctrl> key, you can select several rows that are not contiguous.

With the <Shift> key pressed, you can select the beginning and the end of a contiguous area.

Automatic cell filling (ST7 data points only)

Automatic cell filling is available for ST7 data points in the following function versions:

- **Overwrite**
- **Paste**

Automatic filling out is particularly useful for the following applications:

- You have copied ST7 data points.
  You want to fill the cells of a column with the same value or in ascending or descending order.
- You have created ST7 data points.
  You want to add more data points of the same type.

Proceed as follows to fill in the form automatically:

1. Select one or more cells (start cells) of a column for the function.
2. Move the cursor to the lower right corner of the selection and drag the cursor down to the target cells.

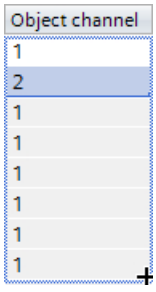

   A dialog box opens in which you select the desired function version.
3. Select the desired "Overwrite" or "Paste" option and click "OK".

Overwrite

Depending on the desired content of the target cells, proceed as follows:

- Select a single start cell if you want to overwrite the target cells with the value of the start cell.
- Select at least two start cells with numerically ascending values if you want to overwrite the target cells with numerically ascending values. This version is shown above in step 2.
  If you select two cells with different text contents (e.g. Bool[1] and DWord[1] PLC tag), then the target cells are overwritten only with the first value.
  You select multiple cells as you would for copying:
  - <Ctrl> key: Selection of multiple cells. The cells must be contiguous.
  - <Shift> key: Selection of a contiguous area
- Select multiple start cells with any value block if you want to overwrite the target cells with this value block.
  Example: You select three cells with the values 4, 12 and 7. The target cells are then continuously overwritten with blocks of 4, 12 and 7.

Exception: Special rules apply to the names of the data points; see below.

If the start cells are selected as shown in the figure above, the target cells are overwritten with the following values after dragging the cursor.

Object channel
1
2
3
4
5
6
7
8

You can also overwrite target cells in numerical descending order if the two start cells contain descending values or if you select two cells with the highest numbers and drag the cursor upwards. If overwritten in descending order, the lowest value is 1, since 0 is an invalid value.

Overwriting can be used in the following columns:

- Name
  When overwriting the names of data points, no consecutive names are assigned. However, a unique name is assigned to each data point.
- PLC tag
  Requirement: The target cells are assigned data point types that are compatible with the start cell.
- Data point type
  Requirement: The data points of the target cells use the same PLC tag.
- Type of transmission
  Requirement: The data points of the target cells use the same PLC tag and have the same transmission direction (input or output).
- Object channel
  The rules for assigning channel numbers apply, see section ST7 data points (Page 1768).
- Object number
  The rules for assigning object numbers apply.

Paste

This option inserts new rows at the end of the table.

- Select one or more cells and then drag the cursor to other cells.
  The total number of cells selected after dragging the cursor is inserted.

The function is supported in the following columns:

- Name
- PLC tag
- Data point type
- Type of transmission
- Object channel
- Object number

Exporting and importing data points

To simplify the engineering of larger plants, you can export the data points of a configured module and import them into other modules in the project. This is an advantage particularly in projects with many identical or similar stations or modules.

You can also access the export/import function:

- Using the two symbols above the data point list
- Via the shortcut menu of the module, if you select it in the network or device view.

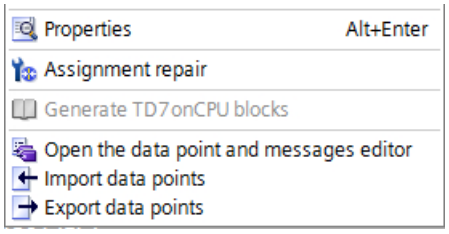

Figure 1-79 Part of the shortcut menu of a telecontrol module

The data point information of a module is saved in exactly one CSV file during export.

Export

When you call the export function, the export dialog opens. Here, you select the CPs or modules of the project whose data point information needs to be exported. When necessary, you can export the data points of all modules of the project at one time.

In the export dialog, you can select the storage location in the file directory. When you export the data of a module you can also change the preset file name.

When you export from several modules, the files are formed with preset names made up of the station name and module name.

The file itself contains the following information in addition to the data point information:

- Module name
- Module type
- CPU name
- CPU type

Editing the export files

You can edit the data point information in an exported CSV file. This allows you to use this file as a configuration template for many other stations.

If you have a project with many stations of the same type, you can copy the CSV file with the data points of a fully configured module for other as yet unconfigured stations and adapt individual parameters to the particular station. This saves you having to configure the data points for every module in STEP 7. Instead, you simply import the copied and adapted CSV file to the other modules of the same type. When you import this file into another module, the changed parameter values of the CSV file are adopted in the data point configuration of this module.

The lines of the CSV file have the following content:

- Line 1: ,Name,Type,
  This line must not be changed.
- Line 2: PLC,<CPU name>, <CPU type>,
  Meaning: PLC (designation of the station class), CPU name, CPU type
  Only the elements <CPU name> and <CPU type> may be changed.
  The CPU type must correspond exactly to the name of the CPU in the catalog.
- Line 3: Module,<module name>,<module type>,
  Meaning: Module (Designation of the module class), module type, module name
  Only the elements <module name> and <module type> may be changed.
  Be careful when changing the module names if you want to import data points into several modules (see below).
  The module type must correspond exactly to the name of the module in the catalog.
- Line 4: Parameter names (English) of the data points
  This line must not be changed.
- Lines 5..n: Values of the parameters according to line 4 of the individual data points
  You can change the parameter values for the particular station.

Importing into a module

Before importing the data points make sure that the PLC tags required for the data points have been created.

Note that when you import a CSV file all the data points existing on the module will be deleted and replaced by the imported data points.

Select a CP and select the import function from the shortcut menu of the module. The import dialog opens in which you select the required CSV file in the file directory.

If the information for assigning the individual data points to their respective PLC tags corresponds with the assignment in the original module, the data points are assigned to the respective PLC tags.

When you import data points into a CP, but some required PLC tags have not yet been created in the CPU, the corresponding data point information cannot be assigned. In this case, you can subsequently create missing PLC tags and them assign them the imported data point information. The "Assignment repair" function is available for this (see below).

If the names of the PLC tags in the module into which the import is made have different names than in the module that exported, the corresponding data points cannot be assigned to your PLC tags.

Importing into several modules

You can import the data points from several modules into the modules of a different project. To do this in the import dialog select all the required CSV files with the control key.

Before importing the data points, make sure that the respective stations have been created with CPUs of the same name, modules of the same name and PLC tags of the same name.

When you import the corresponding stations of the project are searched for based on the module names in the CSV files. If a target station does not exist in the project or the module has a different name, the import of the particular CSV file will be ignored.

Restrictions for the import of data points

In the following situations the import of data points will be aborted:

- An attribute required by the module is missing in the CSV file to be imported.
  Example: If a data point to be imported uses a time trigger, the import will be aborted if no time-of-day synchronization was configured for the module.
- The telecontrol protocol used by the module differs from that of the original module.
  Modules with the same telecontrol protocol are compatible with each other:

Only when importing into several modules:

- The import is aborted when a module or CPU name is different from the data in the CSV file.

Assignment repair

If you have named the PLC tags differently in a station to which you want to import them than in the station from which the CSV file was exported, the assignment between data point and PLC tag is lost during the import.

You then have the option to either rename the existing PLC tags appropriately or add missing PLC tags. You can then repair the assignment between unassigned data points and PLC tags. This function is available either via the shortcut menu of the CP (see above) or with the following icon to the upper left in the data point editor:

If a PLC tag with a matching name is found for a data point by the repair function, the assignment is restored. However the data type of the tag is not checked.

After the assignment repair make sure that you check whether the newly assigned PLC tags are correct.

Datapoint types

During the configuration of the user data to be transferred by the communications module, each data point is created as a protocol-specific data point type.

Afterwards the protocol-specific data point types along with the compatible S7 data types are listed.

The "Direction" column shows the transmission direction:

- "in": Monitoring direction
- "out": Control direction

With the ST7 protocol the transmission direction can be read from the object name.

Note

Effect of the change of arrays for data points

If an array is modified later, the data point must be recreated.

Data point types of the "TeleControl Basic" protocol

Table 1-126 Supported data point types and compatible S7 data types

Format (memory requirements)	Data point type	Direction	S7 data types	Operand area
Bit	Digital input	in	Bool	I, Q, M, DB
	Digital output	in	Bool	Q, M, DB
Byte	Digital input	in	Byte, Char, USInt	I, Q, M, DB
	Digital output	out	Byte, Char, USInt	Q, M, DB
Integer with sign (16 bits)	Analog input	in	Int	I, Q, M, DB
	Analog output	out	Int	Q, M, DB
Counter (16 bits)	Counter input	in	Word, UInt	I, Q, M, DB
Integer with sign (32 bits)	Analog input	in	DInt	Q, M, DB
	Analog output	out	DInt	Q, M, DB
Counter (32 bits)	Counter input	in	UDInt, DWord	I, Q, M, DB
Floating-point number with sign (32 bits)	Analog input	in	Real	Q, M, DB
	Analog output	out	Real	Q, M, DB
Floating-point number with sign (64 bits)	Analog input	out	LReal	Q, M, DB
	Analog output	out	LReal	Q, M, DB
Data block (1 .. 64 bytes)	Data	in / out	ARRAY [1)]	DB
	Data	in / out	ARRAY [1)]	DB

[1)] For the possible formats of the ARRAY data type, refer to the following section.

Data block (ARRAY)

With the ARRAY data type, contiguous memory areas up to a size of 64 bytes can be transferred. The following S7 data types are compatible components of ARRAY:

- Byte, USInt (total of up to 64 per block of data)
- Char (total of up to 64 per block of data) - CP as of firmware version 2.1.77
- Int, UInt, Word (total of up to 32 per block of data)
- DInt, UDInt, DWord (total of up to 16 per block of data)

If the array is modified later, the data point must be recreated.

Time stamp in UTC format

Time stamps are output by the OPC server applications in UTC format (48 bits) and contain milliseconds.

Data point types of the "ST7" protocol

The direction of the data transmission can be seen in the "Object" column:

- Sending objects have the suffix "_S".
- Received objects have the suffix "_R".

Table 1-127 Supported data point types and compatible S7 data types

Format (memory requirements)	Data point type	S7 data type	Operand area	Data points (channels) per object	Object
Bit	Digital input	Bool	I, Q, M, DB	8	Bin08X_S
	Digital output	Bool	Q, M, DB	8	Bin08X_R
Byte	Digital input	Byte, USInt [3]	I, Q, M, DB	4	Bin04B_S
	Digital output	Byte, USInt [3]	Q, M, DB	4	Bin04B_R
	Command output	Byte, USInt [3]	Q, M, DB	1	Cmd01B_R
	Command input	Byte, USInt [3]	Q, M, DB	1	Cmd01B_S
Integer with sign (16 bits)	Analog input	Int	I, Q, M, DB	4	Ana04W_S
	Mean value input	Int	I, Q, M, DB	4	Mean04W_S
	Analog output	Int	Q, M, DB	4	Ana04W_R
	Mean value output	Int	Q, M, DB	4	Mean04W_R
	Setpoint output [1]	Int	Q, M, DB	1	Set01W_R
	Setpoint input [1]	Int	Q, M, DB	1	Set01W_S
Counter (16 bits)	Counter input	UInt, Word [3]	I, Q, M, DB	1	Cnt01D_S
	Counter input	UInt, Word [3]	I, Q, M, DB	4	Cnt04D_S
	Counter output	UDInt, DWord [3]	I, Q, M, DB	1	Cnt01D_R
	Counter output	UDInt, DWord [3]	I, Q, M, DB	4	Cnt04D_R
Floating-point number (32 bits)	Analog input	Real [3]	M, DB	4	Ana04R_S
	Analog output	Real [3]	Q, M, DB	4	Ana04R_R
Data block (4 .. 48 bytes)	Data input	ARRAY [1...12] of DInt / UDInt / DWord / Real [2] [3]	DB	12	Dat12D_S
	Data output		DB	12	Dat12D_R
	Parameter output [1]		DB	12	Par12D_R
	Parameter input [1]		DB	12	Par12D_S

[1] See below, section "1: Mirroring back"

[2] See below, section "2: Data block"

[3] USINT, UInt, Real and UDInt are not supported by S7-300/400.

1: Mirroring (TIM 3V-IE / TIM 4R-IE)

Mirroring of the current local data to the partner can be configured for the following data point types:

- Setpoint input/output (Set01W_R / Set01W_S)
- Parameter input/output (Par12D_R / Par12D_S)

If the "Setpoint input mode of the partner" option is enabled, the local values are monitored for changes and transferred to the partner when the value changes.

You configure the function via the parameters of the "Local setpoint input" tab; see section "Local setpoint input" tab (Page 1774).

2: Data block

With the ARRAY data type, blocks of data from contiguous memory areas up to a size of 4 .. 48 bytes can be transferred.

Compatible components of ARRAY are DInt, UDint, DWord or Real. The components within an array must be of the same type.

Format of the time stamp

For information on the format of the time stamp, refer to the manual of the module.

Data point types of the "DNP3" protocol

Table 1-128 Supported data point types, DNP3 object groups, variants and compatible S7 data types

Format (memory requirements)	Data point type CP [Data point type TIM]	DNP3 object group [variations]	Direction	S7 data types		Operand area
				CP	TIM	
Bit	Binary Input	**1** [1, 2]	in	Bool	Bool	I, Q, M, DB
	Binary Input Event	**2** [1, 2]	in	Bool	Bool	I, Q, M, DB
	Binary Output [1)]	**10** [2]	out	Bool	Bool	Q, M, DB
	Binary Output Event [1)]	**11** [1, 2]	out	Bool	Bool	Q, M, DB
	Binary Command	**12** [1]	out	Bool	Byte	Q, M, DB
Integer (16 bits)	Counter	**20** [2]	in	UInt, Word	Word	I, Q, M, DB
	Frozen Counter [2)]	**21** [2, 6]	in	UInt, Word	Word	I, Q, M, DB
	Counter Event	**22** [2, 6]	in	UInt, Word	Word	I, Q, M, DB
	Frozen Counter Event [3)]	**23** [2, 6]	in	UInt, Word	Word	I, Q, M, DB
	Analog Input	**30** [2]	in	Int	Int	I, Q, M, DB
	Analog Input Event	**32** [2]	in	Int	Int	I, Q, M, DB
	Analog Output Status [4)]	**40** [2]	out	Int	Int	Q, M, DB
	Analog Output	**41** [2]	out	Int	Int	Q, M, DB
	Analog Output Event [4)]	**42** [2, 4]	out	Int	Int	Q, M, DB
Integer (32 bits)	Counter	**20** [1]	in	DWord	DWord	I, Q, M, DB
	Frozen Counter [2)]	**21** [1, 5]	in	DWord	DWord	I, Q, M, DB
	Counter Event	**22** [1, 5]	in	DWord	DWord	I, Q, M, DB
	Frozen Counter Event [3)]	**23** [1, 5]	in	DWord	DWord	I, Q, M, DB
	Analog Input	**30** [1]	in	DInt	-	Q, M, DB
	Analog Input Event	**32** [1]	in	DInt	-	Q, M, DB
	Analog Output Status [4)]	**40** [1, 3]	out	DInt	DWord	Q, M, DB
	Analog Output	**41** [1]	out	DInt	DWord	Q, M, DB
	Analog Output Event [4)]	**42** [1]	out	DInt	DWord	Q, M, DB

Format (memory requirements)	Data point type CP [Data point type TIM]	DNP3 object group [variations]	Direction	S7 data types		Operand area
				CP	TIM	
Floating-point number (32 bits)	Analog Input	30 [5]	in	Real	-	Q, M, DB
	Analog Input Event	32 [5, 7]	in	Real	-	Q, M, DB
	Analog Output Status [4]	40 [3]	out	Real	-	Q, M, DB
	Analog Output	41 [3]	out	Real	-	Q, M, DB
	Analog Output Event [4]	42 [5, 7]	out	Real	-	Q, M, DB
Floating-point number (64 bits)	Analog Input	30 [6]	in	LReal	-	Q, M, DB
	Analog Input Event	32 [6, 8]	in	LReal	-	Q, M, DB
	Analog Output	41 [4]	out	LReal	-	Q, M, DB
	Analog Output Event [4]	42 [6, 8]	out	LReal	-	Q, M, DB
Data block (1...64 bytes) [5]	Octet String / Octet String Output	110 [-]	in, out	[5]	[5]	DB
	Octet String Event [5]	111 [-]	in, out	[5]	[5]	DB

[1] This object group can be configured in the Data point editor of STEP 7 using the substitute object group 12.

[2] This object group can be configured in the Data point editor of STEP 7 using the substitute object group 20.

[3] This object group can be configured in the Data point editor of STEP 7 using the substitute object group 22.

[4] This object group can be configured in the Data point editor of STEP 7 using the substitute object group 41.

[5] With these data point types, contiguous memory areas up to a size of 64 bytes can be transferred. All S7 data types with a size between 1 and 64 bytes are compatible.

Explanation of the table footnotes [1], [2], [3], [4]: Configuring data points using substitute object groups

The initial data point types of the following object groups can be configured using the substitute object groups listed above:

- 10 [2]
- 11 [1, 2]
- 21 [1, 2, 5, 6]
- 23 [1, 2, 5, 6]
- 40 [1, 2, 3]
- 42 [1, 2, 4, 5, 6, 7, 8]

To configure the DNP3 CP, use the specified substitute object group.

Assign each data point on the master using the configurable data point index in STEP 7. The data point of the DNP3 CP is then assigned to the corresponding data point on the master.

Example of configuring the data point Binary Output (10 [2])
The data point is configured as follows:
On the DNP3 CP as Binary Command (12 [1])
On the master as Binary Output (10 [2])

With the data point types Binary Output Event (11) and Analog Output Event (42), you also need to enable mirroring; refer to the next section.

Configuration of the mirroring for output events (object groups 11 and 42)

You first create the data point types Binary Output Event (object group 11) and Analog Output Event (object group 42) as described above as data points of the object groups 12 or 41.

The local values of these two object groups can be monitored for change and the changes transferred to the master (). Changing a local value can, for example, be caused by manual operator input on site.

To allow the value resulting from local events or interventions to be transferred to the master, the data point in question requires a channel for mirroring back. You configure this mirroring back function is configured using the "Value monitoring" option in data point configuration, General tab.

Remember that to use the mirror back function, you need to interconnect the local values in the controller with the relevant PLC tag of the data point.

Time stamp of the data with the DNP3 protocol in UTC format

Time stamps are transferred in UTC format (48 bits) and contain milliseconds.

Data point types of the "IEC 60870-5" protocol

Table 1-129 Supported data point types, IEC types and compatible S7 data types

Format (memory requirements)	Data point type	IEC type	Direction	S7 data types	Operand area
Bit	Single-point information	<1>	in	Bool	I, Q, M, DB
	Single-point information with time tag CP56Time2a [1]	<30>	in	Bool	I, Q, M, DB
	Single command	<45>	out	Bool	Q, M, DB
	Single command with time tag CP56Time2a [1]	<58>	out	Bool	Q, M, DB
	Double command with time tag CP56Time2a [1]	<59>	out	Bool	DB [2]
Byte	Step position information	<5>	in	Byte, USInt	I, Q, M, DB
	Step position information with time tag CP56Time2a [1]	<32>	in	Byte, USInt	I, Q, M, DB
	Regulating step command with time tag CP56Time2a [1]	<60>	out	Byte, USInt	DB [2]
Integer (16 bits)	Measured value, normalized value	<9>	in	Int	I, Q, M, DB
	Measured value, normalized value with time tag CP56Time2a [1]	<34>	in	Int	I, Q, M, DB
	Measured value, scaled value	<11>	in	Int	I, Q, M, DB
	Measured value, scaled value with time tag CP56Time2a [1]	<35>	in	Int	I, Q, M, DB
	Set point command, normalized value	<48>	out	Int	Q, M, DB
	Set point command, scaled value	<49>	out	Int	Q, M, DB
	Set point command, normalized value with time tag CP56Time2a [1]	<61>	out	Int	Q, M, DB
	Set point command, scaled value with time tag CP56Time2a [1]	<62>	out	Int	Q, M, DB

Format (memory requirements)	Data point type	IEC type	Direction	S7 data types	Operand area
Integer (32 bits)	Bitstring of 32 bits	<7>	in	UDInt, DWord	I, Q, M, DB
	Bitstring of 32 bits with time tag CP56Time2a [1]	<33>	in	UDInt, DWord	I, Q, M, DB
	Integrated totals	<15>	in	UDInt, DWord	I, Q, M, DB
	Integrated totals with time tag CP56Time2a [1]	<37>	in	UDInt, DWord	I, Q, M, DB
	Bitstring of 32 bits	<51>	out	UDInt, DWord	Q, M, DB
	Bitstring of 32 bits with time tag CP56Time2a - control direction [1]	<64>	out	UDInt, DWord	Q, M, DB
Floating-point number (32 bits)	Measured value, short floating point number	<13>	in	Real	Q, M, DB
	Measured value, short floating point number with time tag CP56Time2a [1]	<36>	in	Real	Q, M, DB
	Set point command, short floating point number	<50>	out	Real	Q, M, DB
	Set point command, short floating point with time tag CP56Time2a [1]	<63>	out	Real	Q, M, DB
Data block (1...2 Bit) [2]	Double-point information	<3>	in	[2]	DB
	Double-point information with time tag CP56Time2a [1]	<31>	in	[2]	DB
	Double command	<46>	out	[2]	DB
	Regulating step command	<47>	out	[2]	DB
	Double command with time tag CP56Time2a [1]	<59>	out	[2]	DB
	Regulating step command with time tag CP56Time2a [1]	<60>	out	[2]	DB
Data block (1...32 Bit) [3]	Bitstring of 32 bits [3]	<7>	in	[3]	DB
	Bitstring of 32 bits with time tag CP56Time2a [1] [3]	<33>	in	[3]	DB
	Bitstring of 32 bits [3]	<51>	out	[3]	DB
	Bitstring of 32 bits with time tag CP56Time2a - control direction [1] [3]	<64>	out	[3]	DB

[1] For the format of the time stamp, see the following section.

[2] For these data point types, create a data block with an array of precisely 2 bool.

[3] With these data point types, contiguous memory areas up to a size of 32 bits can be transferred. Only the S7 Bool data type is compatible.

Time stamp of the data with the IEC protocol

Time stamps are transferred according to the IEC specification in the "CP56Time2a" format. Note that only the first 3 bytes for milliseconds and minutes are transferred.

Process image, type of transmission, event classes

Introduction

Some of the functions described below differ depending on the communication module or protocol used.

Storage of values

As a rulethe values of all data points are stored in the image memory of the module. Values in the image memory are only transferred after being called up by the master station or the DNP3 / IEC master.

Events are also stored in the send buffer and can be transferred unsolicited.

The image memory, the process image of the module

The image memory is the process image of the communications module. All the current values of the configured data points are stored in the image memory. New values of a data point overwrite the last stored value in the image memory.

The values are only transferred under the following conditions.

- After querying the communications partner, refer to "Transfer after call" in the section "Types of transmission".
- Along with a data frame from the send buffer that needs to be transferred immediately.

The send buffer

The send buffer of the communications module is the memory for the individual values of data points that are configured as an event (frame memory). You will find the size of the send buffer in the manual of the relevant module.

The capacity of the send buffer is divided up equally for all enabled partners.

If the connection to a communications partner is interrupted, the individual values of the events are retained in the buffer. When the connection returns, the buffered values are sent. The frame memory operates chronologically; in other words, the oldest data is sent first (FIFO principle).

If a data frame was transferred to the communications partner, the transferred data is deleted from the send buffer.

If data cannot be transferred for a longer period of time and the send buffer is threatening to overflow, the protocol-dependent response is as follows:

- Telecontrol Basic, ST7
  The forced image mode
  If the send buffer reaches a certain fill level, the module changes to the forced image mode. The fill level for changing to the forced image mode is:
  - Telecontrol Basic: 80% occupation of the send buffer
  - ST7: 90% occupation of the send buffer

  New values from data points configured as an event are no longer added to the send buffer in forced image mode but rather they overwrite older existing values in the image memory. When the connection to the communications partner returns, the module changes back to the send buffer mode as soon as the fill level of the send buffer has fallen below 50%.
- DNP3 / IEC
  If the fill level of the send buffer reaches 100%, no more values are saved until the fill level falls below 100% again.

Saving the data point values

As a rule, the values of data points are stored in the image memory of the moodule and transferred only when queried by the communications partner.

Events are also stored in the send buffer and can be transferred unsolicited.

Data points are configured as a static value or as an event using the "Type of transmission" parameter (see below):

- **Static value (no event)**
  Static values are entered in the image memory (process image of the CP).
  Static values correspond to the following classes:
  - DNP3: Class 0
  - IEC: Class 2
- **Event**
  The values of data points configured as an event are also entered in the image memory of the communications module. The value of the event is sent unsolicited to the communications partner if this function is enabled by the master.
  The values of events are also entered in the send buffer of the communications module.
  Events correspond to the following classes:
  - DNP3: Class 1 / 2 / 3
  - IEC: Class 1

Types of transmission

The following types of transmission are possible:

- **Transfer after call (class 0)**
  The current value of the data point is entered in the image memory of the CP. New values of a data point overwrite the last stored value in the image memory.
  After being called by the communications partner, the current value at this time is transferred.
- **Triggered (events)**
  The values of data points configured as an event are entered in the image memory and also in the send buffer of the CP.
  The values of events are saved in the following situations:
  - The configured trigger conditions are fulfilled (data point configuration > "Trigger" tab, see below)
  - The value of a status bit of the status identifiers of the data point changes; see sectionStatus IDs of the data points (Page 1771).
    Example:
    When the value of a data point configured as an event is updated during startup of the station by reading the CPU data for the first time, the status "RESTART" of this data point changes (bit status change 1 → 0). This leads to generation of an event.

  The configurable event classes of the various protocols are described in the following sections.

Event classes with the "Triggered" type of transmission

Depending on the protocol used, the following event classes are available:

- TeleControl Basic / ST7
  - Every value triggered
    Each value change is entered in the send buffer in chronological order.
  - Current value triggered
    Only the last, current value is entered in the send buffer. It overwrites the value stored there previously.
- DNP3
  The evaluation of the following classification must be handled by the master.
  - Event class 1
    Class according to DNP3 protocol: Class 1
    Each value change is entered in the send buffer in chronological order.
  - Event class 2
    Class according to DNP3 protocol: Class 2
    Each value change is entered in the send buffer in chronological order.
  - Event class 3
    Class according to DNP3 protocol: Class 3
    Only the current value at the time the trigger condition was met is entered in the send buffer and overwrites the last value stored there.
- IEC
  Both of the following event classes correspond to the user data class 1 of the IEC protocol
  - Every value triggered
    Each value change is entered in the send buffer in chronological order.
  - Current value triggered
    Only the current value at the time the trigger condition was met is entered in the send buffer and overwrites the last value stored there.

Various trigger types are available for starting event-driven transfer, see section "Trigger“ tab (Page 1776).

ST7 data points

Objects of TD7 software

The telecontrol communication of modules that use the SINAUT ST7 telecontrol protocol is handled by the TD7 software.

The data to be transferred from the SINAUT objects is alternatively configured or parameterized:

- Via data points (TD7onTIM)
- Via the program blocks of the "Telecontrol ST7" library (TD7onCPU)

For information on the two variants of the TD7 software, refer to the section The TD7 software (Page 1797).

Data objects

The sending and receiving of process data is configured with the aid of standardized data objects. According to the two transmission directions, these are divided into:

- **Send objects (.._S)**
  Data objects for acquiring and sending data
  Their names have the ending "_S" (send).
- **Receive objects (.._R)**
  The data objects for receiving and outputting data
  Their names have the ending "_R" (receive).

The names of the data objects are identical in TD7onTIM and TD7onCPU. In terms of functionality, they are compatible with each other.

Object numbers

Each data object has an object number. When new data points are created, the numbers are assigned in ascending order by default. The numbers can be changed.

Rules:

- The object numbers of a module must be unique.
- The lowest object number is 1.

Object channels

Each data object has one or more object channels, see section Datapoint types (Page 1758).

The send and receive channels of the data objects are responsible for the processing of an individual process value, for example for processing and sending an analog value or receiving and outputting a command byte.

Each data object contains one or more send or receive channels. The number and type of send and receive channels per data object cannot be modified. The data object "Ana04W_S", for example, has 4 send channels of the type "send analog value".

If objects have multiple channels of the same type, not all channels need to be activated if they are not required.

Rules:

- The channel numbers of an object must begin with 1 and be assigned consecutively without a gap.
- All channels of an object must have the same data point type.
- All channels of an object must have the same partner.

DNP3 and IEC data points

General

You will also the most important parameters in the first tab of the data point editor in the data point table.

- Data source
  - Name: Unique name of the data point
  - For the assignment of the PLC tabs, see section Configuring data points and messages (Page 1750).
  - For the data point type, see section Datapoint types (Page 1758).
  - For the data point index, see section Rules for configuring the data point index: (Page 1772).
- Master function
  - Option enabled: The values of the data point are sent as for a master:
    - Input data points are received.
    - Output data points are sent.
  - Option disabled: The data point can be used for 'direct communication' between stations:
    - Input data points can be received and sent.
    - Output data points can be sent and received.
- Value monitoring
  Only for DNP3 commands and outputs
  When this option is activated, the current value of the data point, which can be set by the master, is mirrored back from the station to the master. Furthermore, the trigger options of the data point can be enabled to configure the triggering of the transmission.
- Type of transmission
  For the type of transmission, see section Process image, type of transmission, event classes (Page 1765).
- Read cycle
  Only for inputs
  For the read cycle see section Read cycle (Page 1773).
- Response to general request
  Enables the data point for the response to a general request. If the function is disabled, the value of the data point is not sent to the communications partner following a general request.
- Assignment to group request
  Only IEC data points
  Assigns the data point to a group request.
  For group queries of the master to the respective group, the value of the data point is sent to the master.

Status IDs of the data points

Status identifiers

The status identifiers of the data points listed in the following tables are transferred along with the value in each data frame to the communications partner. They can be evaluated by the communications partner.

The CP 1243-8 IRC does not supply any status identifiers.

The status identifiers differ depending on the protocol used.

The entries in the table row "Significance" relate to the entry in the table row "Bit status".

TeleControl Basic

Depending on their status, the status bits (see table) are converted to the OPC quality code by TCSB.

- Quality = BAD
  Bit 2 or 7 = 1
- Quality = UNCERTAIN
  Bit 1 or 3 or 5 = 1
- Quality = GOOD
  Bits 1 and 2 and 3 and 5 and 6 = 0

Table 1-130 Bit assignment of status byte 0

Bit	7	6	5	4	3	2	1	0
Flag name	-	NON_ EXISTENT	Substituted	LOCAL_ FORCED	CARRY	OVER_ RANGE	RESTART	ONLINE
Meaning	-	Data point does not exist or S7 address unreachable	Substitute value	Local operator control	Counted value overflow before reading the value	Limit value of the analog preprocessing overshot / undershot	Value not yet updated after start	Value is valid
Bit status	*(always 0)*	1	1	1	1	1	1	1

DNP3

The status IDs correspond to the following elements of the specification:

OBJECT FLAGS - DNP3 Specification, Volume 6, Data Object Library - Part 1

Table 1-131 Bit assignment of the status byte

Bit	7	6	5	4	3	2	1	0
Flag name	-	-	-	LOCAL_ FORCED	DISCONTI NUITY	OVER_ RANGE	RESTART	ONLINE
Meaning	-	-	-	Local operator control	Counted value overflow before reading the value	Limit value of the analog preprocessing overshot / undershot	Value not yet updated after start	Value is valid
Bit status	*(always 0)*	*(always 0)*	*(always 0)*	1	1	1	1	1

IEC 60870-5

The status IDs correspond to the following elements of the specification:

Quality descriptor - IEC 60870 Part 5-101

Table 1-132 Bit assignment of the status byte

Bit	7	6	5	4	3	2	1	0
Flag name	-	-	SB substituted	-	CY carry	OV overflow	NT not topical	IV invalid
Meaning	-	-	Substitute value	-	Counted value overflow before reading the value	Value range exceeded, analog value	Value not updated	Value is valid
Bit status	*(always 0)*	*(always 0)*	1	*(always 0)*	1	1	1	0

Rules for configuring the data point index:

Configuration of the data point index

Below you will find the rules for configuring the data point index for the individual protocol types.

TeleControl Basic

Within a CP, the indexes of the data point classes must comply with the following rules:

- Input
  The index of a data point of the type input must be unique throughout all data point types (digital inputs, analog inputs etc.).
- Output
  - A data point of the type output can have the same index as a data point of the type input.
  - Several data points of the type output can have the same index.

Note

Data points for the inter-station communication with a CP in another S7 station

Note that for inter-station communication, the indexes of the two corresponding data points (data point pair) must be identical for the sending and receiving CP.

DNP3

On a CP, data point indexes must be unique within each of the following object groups:

- Binary Input / Binary Input Event
- Binary Output / Binary Command
- Counter / Counter Event
- Analog Input / Analog Input Event
- Analog Output
- Octet String / Octet String Event

Indexes of two data points in different object groups can be identical.

IEC 60870-5-104

The data point indexes must be unique in a CP.

Data point indexes assigned twice are indicated as errors in the consistency check and prevent the project being compiled.

Read cycle

Input data points are assigned to the read cycle of the CPU in the data point configuration in the "General > Read cycle" tab.

Structure of the CPU scan cycle

The cycle with which the transferring module (TIM) scans the memory area of the CPU is made up of the following phases:

- **High-priority read jobs**
  (Fast cycle)
  For all data points with the assignment "Fast cycle" the PLC tags are read in every scan cycle.
  As a rule, it is sufficient to assign only data to be acquired quickly, such as alarms and contact changeover messages as well as command, setpoint and parameter objects for the 1oon check, to the fast cycle.
  For information on the 1oon check, see the Glossary.
- **Write jobs**
  In every cycle, the values of a certain number of unsolicited write jobs are written to the CPU.
  The number of tags written per cycle is specified for the transferring module in the "Communication with the CPU" parameter group with the "Max. number of write jobs" parameter. The tags whose number exceeds this value are then written in the next or one of the following cycles.
- **Low-priority read jobs - proportion**
  (Normal cycle)
  For data points with the assignment "Normal cycle", a proportion of the values of their PLC tags is read in every scan cycle.
  The number of tags read per cycle is specified for the transferring module in the "Communication with the CPU" parameter group with the "Max. number of read jobs" parameter. The tags that exceed this value and can therefore not be read in one cycle are then read in the next or one of the following cycles.
- **Cycle idle time**
  This waiting time between two scan cycles is used to reserve adequate time for other processes that access the CPU.

"Local setpoint input" tab

Setpoints and their mirroring

Using the options described below, you can acquire the locally specified values from setpoint outputs and mirror back the current values to the partner upon change. A change to a local value can be caused by, for example, manual operator input on site.

The functions are available for the following modules:

- TIM 3V-IE / TIM 3V-IE Advanced
- TIM 4R-IE

You can configure the functions for the following data point types:

- Setpoint input (Set01W_S)
- Setpoint output (Set01W_R)

- Parameter input (Par12D_S)
- Parameter output (Par12D_R)

The corresponding input and output data points must be created for both communication partners for the functions. For the mirror function, move the local value in the controller to the relevant PLC tag of the data point.

The parameters for local setpoint input mode and mirroring are optional. If you do not require this, leave it disabled.

Parameter

The list shows the functions of the data point types as well as the tabs and parameter groups in which the functions are configured.

- **Setpoint/parameter input**
  "General" > "Data source" tab:
  - Send setpoint to partner
    The setpoint is stored in the PLC tag.

  "Local setpoint input" > "Receive" tab:
  - Receipt of the 'Local input' setpoint input mode sent by the partner object
    The input mode is transferred to the PLC tag "Setpoint input mode" (Bool).
  - Reception of the return value from the partner
    The value mirrored by the partner is transferred to the PLC tag "Mirrored value".
- **Setpoint/parameter output**
  "General" > "Data source" tab:
  - Setpoint received from partner
    The setpoint is transferred to the PLC tag.

  "Local setpoint input" > "Send" tab:
  - Send the 'Local input' local setpoint input mode to the partner object
    The information as to whether the locally assigned setpoint or the setpoint received from the partner is valid is read from the PLC tag "Setpoint input mode".
    - 0: The received setpoint is valid.
    - 1: The locally entered setpoint is valid.
  - Send the locally entered setpoint to the partner
    The setpoint is read from the PLC tag "Local setpoint".

Local setpoint input - Reception

The parameter group is active for sending data types (*_S).

The following parameters are available for local setpoints of the partner:

- Setpoint input mode
  Via the parameter, the feedback is received from the partner data point about its setpoint input mode.
  The information is stored in the PLC tag (Bool).
  - 0 = Data point of the partner in 'Setpoint receipt' mode.
  - 1 = Data point of the partner in 'local' setpoint input mode.

  The parameter is used only for signaling. This parameter and the corresponding parameter of the partner data point do not cause an interlock for the partner with the local setpoint specification. Interlocking must be implemented by the user program.
  After startup of the CPU or partner CPU, or after the return of the connection, a general request ensures that the current, valid status of the partner is displayed at the parameter.
- Return value
  The partner data point that receives the setpoint feeds back the currently valid setpoint if the parameter "Setpoint status" is enabled there. This mirrored back setpoint is displayed here.
  If the partner object is set to 'local' and if new input is made there, the setpoint changed there is displayed here if the Local setpoint parameter is enabled for the partner object.
  After startup of the CPU or partner CPU, or after the return of the connection, a general request ensures that the current, valid setpoint of the partner is displayed at the parameter.

Local setpoint input - Sending

The parameter group is enabled for receiving data types (*_R).

The following parameters are available for local setpoints of the own station:

- Setpoint input mode
  This input can be used to signal whether the locally specified setpoint or the setpoint sent by the partner is valid.
  The information is stored in the PLC tag (Bool).
  - 0 = The setpoint of the partner is valid.
  - 1 = The locally entered setpoint is valid.
    A setpoint sent by the partner (e.g. the master station) can also be accepted by the object if the parameter is set (True).

  The parameter is used only for information. An interlock with the remote setpoint assignment must be implemented in the user program.
- Local setpoint
  If this parameter is enabled, the locally entered setpoint is read from the PLC tag and mirrored back to the send data point of the partner.

"Trigger“ tab

Trigger

Data points are configured as a static value or as an event using the "Type of transmission" parameter.

Storing an event value in the send buffer (frame memory) is triggered via various trigger types:

- **Threshold value trigger**
  The value of the data point is saved when this reaches a certain threshold. The threshold is calculated as the difference compared with the last stored value, refer to the section Threshold value trigger (Page 1778).
- **Time trigger**
  The value of the data point is saved at configurable intervals or at a specific time of day.
- **Event trigger (Trigger tag)**
  The value of the data point is saved when a configurable trigger signal is fired. The edge change (0 → 1) of a trigger tag set by the user program is evaluated as a trigger signal. When necessary, a separate trigger tag can be configured for each data point.
  Reset of the trigger tag in the memory bit area / DB:
  If the memory area of the trigger tag is in the bit memory or in a data block, the trigger tag is reset to zero when the value is transferred. This can take up to 500 milliseconds.

Note

Fast setting of triggers

Triggers must not be set faster than a minimum interval of 500 milliseconds. This also applies to hardware triggers (input area).

Note

Hardware trigger

You need to reset hardware triggers via the user program

Transmission time

Whether the value of an event is transmitted to the communication partner immediately after activating the trigger or delayed depends on the protocol used and the settings:

- DNP3 / IEC
  The spontaneous transfer in these protocols depends on whether spontaneous sending or asymmetric communication is possible in the network.
  For DNP3 modules, this can be set with the "Transmission mode" in the "Transmission settings - DNP3" parameter group of the interface.
- TeleControl Basic / ST7
  Here you set the time of transmission with the "Transmission mode" parameter at the data point.

Transmission mode

The transmission mode of a data frame is set in the "Trigger" tab of the data point. With the option, you specify whether data frames of events are sent immediately or following a delay:

- Spontaneous
  The value is transferred immediately.
- Conditional spontaneous
  The value is transferred only when one of the following conditions is fulfilled:
  - The communications partner queries the station.
  - The value of another event with the transmission mode "Unsolicited" is transferred.
  - The fill level of the send buffer has reached 80% of its maximum capacity.

Threshold value trigger

Note

Threshold value trigger: Calculation only after "Analog value preprocessing"

Note that the analog value preprocessing is performed before the check for a configured threshold value and before calculating the threshold value.

This affects the value that is configured for the threshold value trigger.

Note

No Threshold value trigger if Mean value generation is configured

If mean value generation is configured, no threshold value trigger can be configured for the analog value event involved.

For the time sequence of the analog value preprocessing refer to the section Analog value preprocessing (Page 1780).

Threshold value trigger

Function

If the process value deviates by the amount of the threshold value, the process value is saved.

Two methods are used to calculate the threshold value deviation:

- **Absolute method**
  With binary and counter values as well as with analog values with configured mean value generation, the absolute method is used to calculate the threshold value deviation.
- **Integrative method**
  With analog values without configured mean value generation, the integrating method is used to calculate the threshold value deviation.
  In the integration threshold value calculation, it is not the absolute value of the deviation of the process value from the last stored value that is evaluated but rather the integrated deviation.

Absolute method

For each binary value a check is made to determine whether the current (possibly smoothed) value is outside the threshold value band. The current threshold value band results from the last saved value and the amount of the configured threshold value:

- Upper limit of the threshold value band: Last saved value + threshold value
- Lower limit of the threshold value band: Last saved value - threshold value

As soon as the process value reaches the upper or lower limit of the threshold value band, the value is saved. The newly saved value serves as the basis for calculating the new threshold value band.

Integrative method

The integration threshold value calculation works with a cyclic comparison of the integrated current value with the last stored value. The calculation cycle in which the two values are compared is 500 milliseconds.
(Note: The calculation cycle must not be confused with the scan cycle of the CPU memory areas).

The deviations of the current process value are totaled in each calculation cycle. The trigger is set only when the totaled value reaches the configured value of the threshold value trigger and a new process value is entered in the send buffer.

The method is explained based on the following example in which a threshold value of 2.0 is configured.

Table 1-133 Example of the integration calculation of a threshold value configured with 2.0

Time [s] (calculation cycle)	Process value stored in the send buffer	Current process value	Absolute deviation from the stored value	Integrated deviation
0	**20.0**	**20.0**	0	0
0.5		20.3	+0.3	0.3
1.0		19.8	-0.2	0.1
1.5		20.2	+0.2	0.3
2.0		20.5	+0.5	0.8
2.5		20.3	+0.3	1.1
3.0		20.4	+0.4	1.5
3.5	**20.5**	**20.5**	+0.5	**2.0**
4.0		20.4	-0.1	-0.1
4.5		20.1	-0.4	-0.5
5.0		19.9	-0.6	-1.1
5.5		20.1	-0.4	-1.5
6.0	**19.9**	**19.9**	-0.6	**-2.1**

With the changes in the process value shown in the example, the threshold value trigger configured with 2.0 fires twice:

- At the time 3.5 s: The value of the integrated deviation is at 2.0. The new process value stored in the send buffer is 20.5.
- At the time 6.0 s: The value of the integrated deviation is at 2.1. The new process value stored in the send buffer is 19.9.

In this example, if a deviation of the process value of approximately 0.5 should fire the trigger, then with the behavior of the process value shown here a threshold value of approximately 1.5 ... 2.5 would need to be configured.

Analog value preprocessing

CPs with data point configuration support analog value preprocessing. For analog value data points, some or all of the functions described below can be configured.

Requirements and restrictions

You will find the requirements for the configuration of the preprocessing options and restrictions in the section relating to the particular function.

Note

Restrictions due to configured triggers

The analog value preprocessing options "Fault suppression time", "Limit value calculation" and "smoothing" are not performed if no threshold value trigger is configured for the relevant data point.. In these cases, the read process value of the data point is entered in the image memory of the CP before the preprocessing cycle of the threshold value calculation (500 ms) elapses.

Sequence of the analog value preprocessing options

The values of analog inputs configured as an event are processed on the CPU according to the following scheme:

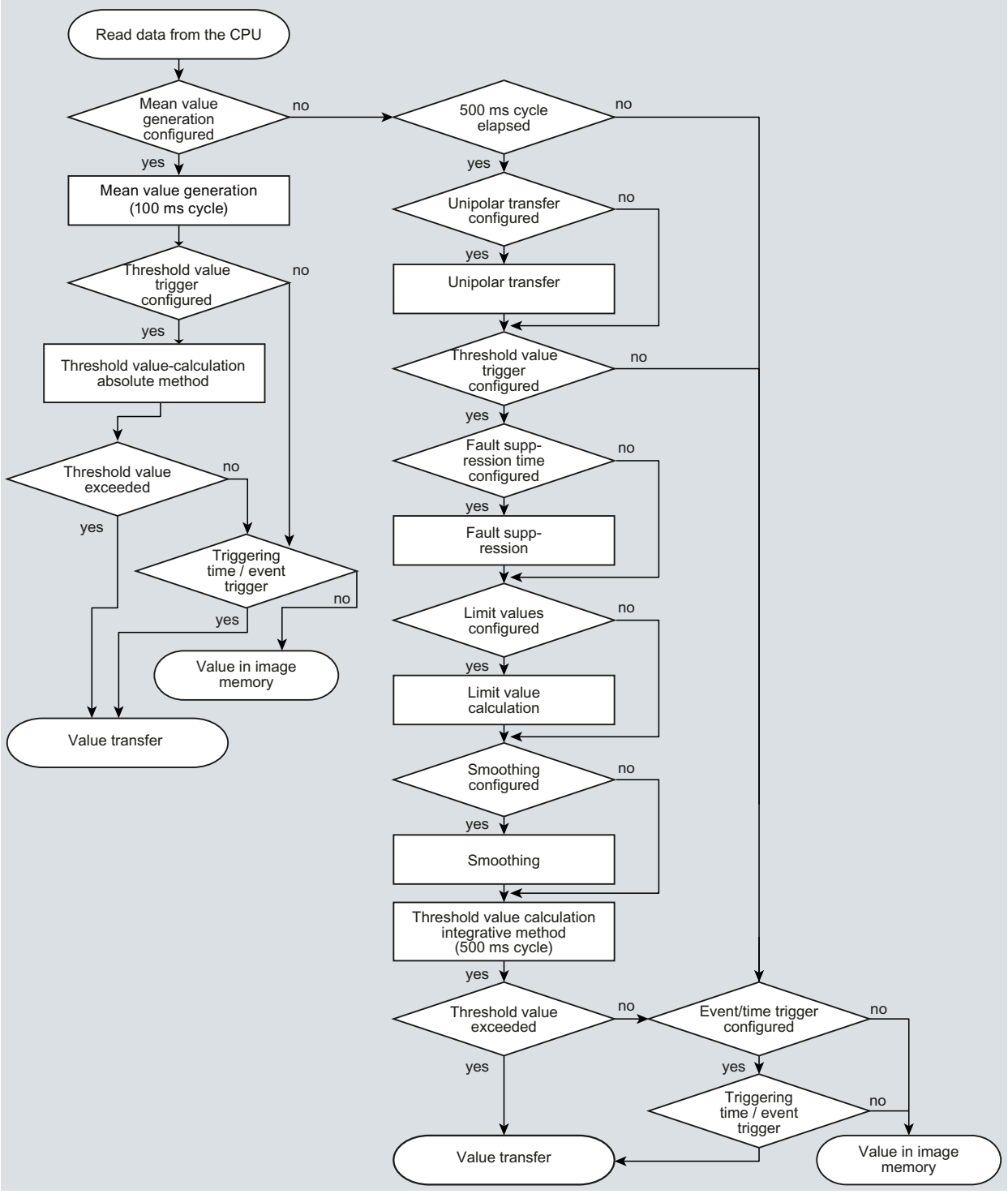

Figure 1-80 Sequence of the analog value preprocessing

The 500 millisecond cycle is started by the integrative threshold value calculation. In this cycle, the values are saved even when the following preprocessing options are enabled:

- Unipolar transfer
- Fault suppression time
- Limit value calculation
- Smoothing

Mean value generation

Note

Restricted preprocessing options if mean value generation is configured

If you configure mean value generation for an analog value event, the following preprocessing options are not available:

- Unipolar transfer
- Fault suppression time
- Smoothing

Function

With this parameter, acquired analog values are transferred as mean values.

If mean value generation is active, it makes sense to configure a time trigger..

The current values of an analog data point are read in a 100 millisecond cycle and totaled. The number of read values per time unit depends on the read cycle of the CPU and the CPU scan cycle of the CP.

The mean value is calculated from the accumulated values as soon as the transfer is triggered by a trigger. Following this, the accumulation starts again so that the next mean value can be calculated.

The mean value is formed even if the transmission of the analog value frame is triggered by a call of the communication partner. The duration of the mean value calculation period is then the time from the last transmission (for example initiated by the trigger) to the time of the call. Once again, the accumulation restarts so that the next mean value can be calculated.

Input modules: Overflow range / underflow range

As soon as a value is acquired in the overflow or underflow range, mean value generation is stopped. The value 32767 / $7FFF_h$ or -32768 / 8000_h is saved as an invalid mean value for the current mean value calculation period and sent with the next data frame.

The calculation of a new mean value is then started. If the analog value remains in the overflow or underflow range, one of the two values named is again saved as an invalid mean value and sent when the next transfer is triggered.

Note

Fault suppression time > 0 configured

If you have configured an error suppression time and then enable mean value generation, the value of the error suppression time is grayed out but no longer used. If mean value generation is enabled, the error suppression time is set to 0 (zero) internally.

Unipolar transfer

Restrictions

Unipolar transfer cannot be configured at the same time as mean value generation. Enabling unipolar transfer has no effect when mean value generation is activated.

Function

With unipolar transfer, negative values are corrected to zero. This can be desirable if values from the underrange should not be transferred as real measured values.

Exception: With process data from input modules, the value -32768 / 8000_h for wire break of a live zero input is transferred.

With a software input, on the other hand, all values lower than zero are corrected to zero.

Fault suppression time

Requirements for the function

Configuration of the threshold trigger for this data point

Restrictions

The fault suppression time cannot be configured at the same time as mean value generation. A configured value has no effect when mean value generation is activated.

Function

A typical use case for this parameter is the suppression of peak current values when starting up powerful motors that would otherwise be signaled to the control center as a disruption.

The transmission of an analog value in the overflow ($7FFF_h$) or underflow range (8000_h) is suppressed for the specified time. The value $7FFF_H$ or 8000_H is only sent after the fault suppression time has elapsed, if it is still pending.

If the value returns to the measuring range before the fault suppression time elapses, the current value is transferred.

Input modules

The suppression is adjusted to analog values that are acquired directly by the S7 analog input modules as raw values. These modules return the specified values for the overflow or underflow range for all input ranges (also for live zero inputs).

An analog value in the overflow range (32767 / $7FFF_h$) or underflow range (-32768 / 8000_h) is not transferred for the duration of the fault suppression time. This also applies to live zero inputs. The value in the overflow/underflow range is only sent after the fault suppression time has elapsed, if it is still pending.

Recommendation for finished values that were preprocessed by the CPU:

If the CPU makes preprocessed finished values available in bit memory or in a data block, suppression is only possible or useful if these finished values also adopt the values listed above 32767 / $7FFF_h$ or -32768 / 8000_h in the overflow or underflow range. If this is not the case, the parameter should not be configured for preprocessed values.

For finished values preprocess in the CPU, the limits for the overflow and underflow can be freely assigned.

Smoothing factor

Requirements for the function

Configuration of the threshold trigger for this data point

Restrictions

The smoothing factor cannot be configured at the same time as mean value generation. A configured value has no effect when mean value generation is activated.

Function

Analog values that fluctuate quickly can be evened out using the smoothing function.

The smoothing factors are calculated according to the following formula as with S7 analog input modules.

$$y_n = \frac{x_n + (k - 1)\ y_{n-1}}{k}$$

where
y_n = smoothed value in the current cycle n
y_{n-1} = smoothed value in the previous cycle n-1
x_n = value acquired in the current cycle n
k = smoothing factor

The following values can be configured for the module as the smoothing factor.

- 1 = No smoothing
- 4 = Weak smoothing
- 32 = Medium smoothing
- 64 = Strong smoothing

Set limit value 'low' / Set limit value 'high'

Requirements for the function

- Configuration of the threshold trigger for this data point
- PLC tag in the bit memory operand area or data area
  The analog value data point must be linked to a PLC tag in the bit memory or data area (data block). For PLC tags of hardware modules (input operand area) limit value configuration is not possible.

The configuration of limit values is pointless for measured values that have already been preprocessed on the CPU.

Function

In these two input boxes, you can set a limit value in the direction of the start of the measuring range or in the direction of the end of the measuring range. You can also evaluate the limit values, for example as the start or end of the measuring range.

Status identifier "OVER_RANGE" / "overflow"

With protocols that support status identifiers, if the limit value is overshot or undershot, the status identifier of the data point is set for measured range violation indicated below as the identifier "OV". This status identifiers are described in the section Status IDs of the data points (Page 1771).

The "OV" bit of the status identifier of the data point is set as follows when the relevant analog value is transferred:

- Limit value 'high':
  - If the limit value is exceeded: OV = 1
  - If the value then falls below the limit value: OV = 0
- Limit value 'low':
  - If the value falls below the limit value: OV = 1
  - If the value then exceeds the limit value: OV = 0

Configuration of the limit value

The limit value is configured as a whole decimal number. The range of values is based on the range of values of the raw value of analog input modules.

Range	Raw value (16-bit)		Module output [mA]			Measuring range [%]
	Decimal	Hexadecimal	0 .. 20 (unipolar)	-20 .. +20 (bipolar)	4 .. 20 (life zero)	
Overflow	32767	7FFF	> 23.515	> 23.515	> 22.810	> 117.593
Overrange	32511 ... 27649	7EFF ... 6C01	23.515 ... 20.001	23.515 ... 20.001	22.810 ... 20.001	117.593 ... 100.004
Nominal range (unipolar / life zero)	27648 ... 0	6C00 ... 0000	20 ... 0		20 ... 4	100 ... 0

Range	Raw value (16-bit)		Module output [mA]			Measuring range [%]
	Decimal	Hexadecimal	0 .. 20 (unipolar)	-20 .. +20 (bipolar)	4 .. 20 (life zero)	
Nominal range (bipolar)	27648 ... 0 ... -27648	6C00 ... 0000 ... 9400		20 ... 0 ... -20		100 ... 0 ... -100
Underrange (unipolar / life zero)	-1 ... -4864	FFFF ... ED00	-0.001 ... -3.518		3.999 ... 1.185	-0.004 ... -17.59
Underrange (bipolar)	-27649 ... -32512	93FF ... 8100		-20.001 ... -23.516		-100.004 ... -117.593
Undershoot / wire break	-32768	8000	< -3.518		< 1.185	< -17.593

Please note: The entry of the value 0 (zero) is interpreted as a deactivated limit value.

Recommendation for quickly fluctuating analog values:

If the analog value fluctuates quickly, it may be useful to smooth the analog value first if limit values are configured.

Masks

Masking of individual bits in the byte of binary inputs (Bin04B_S)

The "Masks" parameter group provides three options for transferring binary values. The corresponding bits are masked in hexadecimal format.

By entering hexadecimal values you can specify bit-by-bit whether certain bits do not trigger transmission or which bits are transferred in a different transmission mode.

- **Alarm mask**
  Changes in masked bits cause an unconditional spontaneous transmission. Changes from 0 to 1 and from 1 to 0 are evaluated.
  The "alarm mask" is only practical when the data point is transmitted over a dialup network and the option "Conditional spontaneous" was activated.
- **Send buffer principle mask**
  Changes in masked bits cause transmission according to the send buffer principle. Changes from 0 to 1 and from 1 to 0 are evaluated.
  The send buffer mask is only practical with data points that have the "Image memory" option enabled.
- **Disable mask**
  Masked bits are ignored when changes are checked. This means that changes to the masked bits for this data point do not trigger transmission. A masked bit always has the value 0 in the data frame.

Example of byte assignment

The following table shows the masking of the bits according to the hexadecimal value "A3" entered in the input box of the parameters.

Table 1-134 Byte assignment

Bit:	.7	.6	.5	.4	.3	.2	.1	.0
Masked	1	0	1	0	0	0	1	1
Hex mask	A				3			

The bits with the value 1 are masked; in other words bits no. 0, 1, 5 and 7 cause the relevant function in the described masks.

Command output

Processing options of command outputs

Parameter

For data points in stations that output commands from a master station or a master, the following output options can alternatively be activated:

- LATCH_ON/OFF
  The function permanently latches a command output to the value 0 or 1.
  Note:
  The latched value is only canceled by a new command. Alternatively, the command can be reset by the user program.
- PULSE_ON/OFF
  The function evaluates the number and length of the signals (pulses) of commands from a master station.

Data point types

The output options can be configured for the following data point types:

- ST7
  Command output (Cmd01B_R)
- IEC
  Single command
- DNP3
  Binary Command Output

Reference to specifications

The output options correspond to the following parts of the specification:

- IEC
  Qualifier of command - IEC 60870-5-101, Part 1
- DNP3
  Control Code, Operation Type field (OP Type) - Volume 6, Part 2, Objects

Protocol-dependent processing

ST7

- LATCH_ON/OFF
  The function latches the value of a command as described above.
- PULSE_ON/OFF
  The function evaluates the "command output time" for a single pulse (see below).

IEC

- LATCH_ON/OFF
  The function latches the value of a command as described above.
- PULSE_ON/OFF
  The function is coded by the master via a "Qualifier of command". The following codes are evaluated by the communication module in the station:
  - QU (Type 1.1) <0> no additional definition
    Corresponding parameter for the module: "Pulse control"
  - QU (Type 1.1) <1> short pulse duration
    Corresponding parameter for the module: "Short pulse duration"
  - QU (Type 1.1) <2> long pulse duration
    Corresponding parameter for the module: "Long pulse duration"
  - S/E (Type 6) <0> execute
    Corresponding parameter for the module: "Command execution mode"
  - S/E (Type 6) <1> select
    Corresponding parameter for the module: "Command execution mode"

DNP3

The two functions are encoded by the master via the Control Code of the object.

- LATCH_ON/OFF
  The function permanently latches a command output to the value 0 or 1.
  The station processes the command according to the Complementary Latch Model.
- PULSE_ON/OFF
  The function evaluates the "PULSE_ON" Control Code in command frames of a master.
  The master sends the information "Trip-Close Count", "On-time", "Off-time" and "Pulse count" with the command frame to the station.
  The station data point evaluates the information and checks it for consistency with the parameters "Max. pulse duration", "Pulse duration replacement time" and "Max. number of pulses".
  The TIM 1531 IRC processes the command according to the Activation Model.

The "Clear" field (CR) is not supported.

The station data point evaluates the coding of the control code as follows.

Table 1-135 Decoding of the control code by the command data point of the station

Received control code			Reaction of the station data point	
Control Code	Trip-Close Code	Operation Type Field	Option enabled: PULSE_ON	Option enabled: LATCH_ON / OFF
0x01	NUL	PULSE_ON	The output is set to 1 for the duration of "On-time".	The command is rejected.
0x03	NUL	LATCH_ON	• CP The command is rejected. • TIM 1531 IRC The output is set to 1 for the duration of "On-time".	The output is set to 1.
0x04	NUL	LATCH_OFF	• CP The command is rejected. • TIM 1531 IRC The output is set to 1 for the duration of "On-time".	The output is set to 0.
0x41	CLOSE	PULSE_ON	• CP The command is rejected. • TIM 1531 IRC The output is set to 1 for the duration of "On-time".	• CP The command is rejected. • TIM 1531 IRC The output is set to 1.
0x81	TRIP	PULSE_ON	• CP The command is rejected. • TIM 1531 IRC The output is set to 1 for the duration of "On-time".	• CP The command is rejected. • TIM 1531 IRC The output is set to 0.

Note

Mirroring

If you want the station data point to mirror the current value back to the master, enable the "Value monitoring" option in the "General" tab of the data point.

Parameter

Name: **Control Code**

Range of values:
- PULSE_ON
- LATCH_ON/OFF

Explanation: Output option of the command output. See above for the meaning.

Name:	**Pulse control** (Master only) • DNP3: Trip-Close Code • IEC: Qualifier of command
Range of values:	DNP3: • NUL • CLOSE • TRIP IEC: • No presettings For additional qualifiers, see the parameters "Short pulse duration / Long pulse duration" below.
Explanation:	Trip-Close Code for Control Code "PULSE_ON". The data point sends the configured code to the corresponding data point of the station. See above for the meaning.

Name:	**Pulse ON duration** (Master only)
Range of values:	0 ... 4294967295
Default:	10
Explanation:	Duration (milliseconds) for "On-time". The data point sends the configured time to the corresponding data point of the station.

Name:	**Pulse OFF duration** (Master only)
Range of values:	0 ... 4294967295
Default:	0
Explanation:	Duration (milliseconds) for "Off-time". The data point sends the configured time to the corresponding data point of the station.

Name:	**Number of pulses** (DNP3 master only)
Range of values:	1 ... 255
Default:	1
Explanation:	Number of pulses that the data point sends to the corresponding data point of the station.

Name:	**Max. number of pulses**
Range of values:	• DNP3: 0 ... 255 • ST7 and IEC: Fixed assignment with "1"
Default:	1
Explanation:	Monitors the number of pulses sent by the master (Count). If the number of pulses received from the master exceeds the value configured here, the command is discarded. If you enter 0 (zero), the monitoring is disabled.

Name:	**Max. pulse duration (s)** (DNP3 only)
Range of values:	0 ... 65535
Default:	0
Explanation:	Monitors the pulse duration transmitted by the master in the command frame (On-time). If the pulse duration exceeds the value configured here, the command is rejected by the station module, provided no "Pulse duration replacement time" is configured. When the "Max. pulse duration" is exceeded and "Pulse duration replacement time" is configured or when the received pulse duration is 0, the "Pulse duration replacement time" is used.

Name:	**Pulse duration replacement time (s)** (DNP3 only)
Range of values:	0 ... 65535
Default:	0
Explanation:	Replacement value for the pulse duration The replacement value is used in the following cases: • The pulse duration received from the master exceeds the value configured under "Max. pulse duration". • The received pulse duration is 0. If the value is 0 (zero), no replacement value is used. The command is discarded. If the "Max. pulse duration" and "Pulse duration replacement time" are both configured with zero, each command is discarded.

Name:	**Short pulse duration (s)** (IEC only)
Range of values:	0 ... 65535

Default:	0
Explanation:	Commands from the master with the Qualifier of command (QOC) = <1> (short pulse duration) are output by the communication module for the duration configured here. With the setting 0 (zero), commands with QOC = <1> are rejected by the module.

Name:	**Long pulse duration (s)** (IEC only)
Range of values:	0 ... 65535
Default:	0
Explanation:	Commands from the master with the Qualifier of command (QOC) = <2> (long pulse duration) are output by the communication module for the duration configured here. With the setting 0 (zero), commands with QOC = <2> are rejected by the module.

Name:	**Command output time (s)** (ST7 only)
Range of values:	0 ... 65535
Default:	0
Explanation:	Period for which the command is output. The duration is valid for all 8 channels of the SINAUT object. When the time expires, the output is automatically reset. With zero, a set command output is not automatically reset. In this case, you need to reset the output via the user program.

Name:	**Command execution mode** (DNP3 and IEC only)
Range of values:	• Execute directly • Select and operate

Default: Execute directly

Explanation:

- Execute directly
  The command is immediately transmitted to the CPU of the station for execution.
- Select and operate
  Procedure:
  - Trigger the command in the master module
    The "select" frame is sent from the master station to the communication module of the station.
  - The station acknowledges receipt.
  - The master data point sends the execution frame after receiving the acknowledgment from the station.
  - The station only forwards the command to the CPU when it receives the "Execution" frame from the master within the configured "Max. time between Select and Operate".
    The station must not receive any other data frame between Select and Operate.

  Note: "Max. time between Select and Operate" is configured in the transmission settings of the respective interface.
  Reference to the specifications:
  - DNP3
    "Select and Operate" is realized via the function codes 03 and 04.
    "Direct Operate" is realized via the function codes 05 and 06.
  - IEC (Select and Execute)
    The communication module acknowledges receipt of the selection ASDU with the Qualifier "S/E (Type 6) <1> select".
    After receiving the acknowledgment, the master sends the execution ASDU with the Qualifier "S/E (Type 6) <1> execute".

Message editor

Processing status of messages (e-mails / SMS messages)

Enable status identifier / External status

If the option is enabled, the CP writes a processing status about the sent e-mail / SMS to a PLC tag.

The meaning of the statuses is as follows:

Table 1-136 SMS: Meaning of the status ID output in hexadecimal format

Status	Meaning
0000	Transfer completed free of errors
0001	Error in the transfer, possible causes: • SIM card invalid. • No network • Wrong destination phone number (number not reachable)

Table 1-137 E-mails: Meaning of the processing status output in hexadecimal format

Status	Meaning
0000	Transfer completed free of errors
82xx	Other error message from the e-mail server Apart from the leading "8", the status corresponds to the three-digit error number of the SMTP protocol.
8401	No channel available Possible cause: There is already an e-mail connection via the CP. A second connection cannot be set up at the same time.
8403	No TCP/IP connection could be established to the SMTP server.
8405	The SMTP server has denied the login request.
8406	An internal SSL error or a problem with the structure of the certificate was detected by the SMTP client.
8407	Request to use SSL was denied.
8408	The client could not obtain a socket for creating a TCP/IP connection to the mail server.
8409	It is not possible to write via the connection. Possible cause: The communications partner reset the connection or the connection aborted.
8410	It is not possible to read via the connection. Possible cause: The communications partner terminated the connection or the connection was aborted.
8411	Sending the e-mail failed. Cause: There was not enough memory space for sending.
8412	The configured DNS server could not resolve specified domain name.
8413	Due to an internal error in the DNS subsystem, the domain name could not be resolved.
8414	An empty character string was specified as the domain name.
8415	An internal error occurred in the cURL module. Execution was aborted.
8416	An internal error occurred in the SMTP module. Execution was aborted.
8417	Requests to SMTP on a channel already being used or invalid channel ID. Execution was aborted.
8418	Sending the e-mail was aborted. Possible cause: Execution time exceeded.
8419	The channel was interrupted and cannot be used before the connection is terminated.
8420	Certificate chain from the server could not be verified with the root certificate of the CP.
8421	Internal error occurred. Execution was stopped.
8450	Action not executed: Mailbox not available / unreachable. Try again later.

Status	Meaning
84xx	Other error message from the e-mail server Apart from the leading "8", the status corresponds to the three-digit error number of the SMTP protocol.
8500	Syntax error: Command unknown. This also includes the error of having a command chain that is too long. The cause may be that the e-mail server does not support the LOGIN authentication method. Try sending e-mails without authentication (no user name).
8501	Syntax error. Check the following configuration data: Message configuration > Message parameters: • Recipient address ("To" or "Cc").
8502	Syntax error. Check the following configuration data: Message configuration > Message parameters: • Email address (sender)
8535	SMTP authentication incomplete. Check the "User name" and "Password" parameters in the CP configuration.
8550	SMTP server cannot be reached. You have no access rights. Check the following configuration data: • CP configuration > E-mail configuration: – User name – Password – Email address (sender) • Message configuration > Message parameters: – Recipient address ("To" or "Cc").
8554	Transfer failed
85xx	Other error message from the e-mail server Apart from the leading "8", the status corresponds to the three-digit error number of the SMTP protocol.

Protocol-specific parameters

TeleControl Basic

CP identification

Reference

CPs using the "TeleControl Basic" protocol.

"CP identification" parameter group

One project number for the entire STEP 7 project

If you change the project number for a CP in the "CP identification" parameter group, this parameter is changed for all CPs in the entire STEP 7 project using the "TeleControl Basic" protocol.

TCSB evaluates project numbers from 1 ... 2000.

Configuration of the station number

You need to configure a unique station number for each station in the STEP 7 project that uses a CP using the "TeleControl Basic" protocol.

TCSB evaluates station numbers from 1 ... 8000.

Access ID

The access ID displayed in the "CP identification" parameter group is made up of the hexadecimal values of the project number, station number and slot. The parameter of the type DWORD is allocated as follows:

- Bits 0 - 7: Slot
- Bits 8 to 20: Station number
- Bits 21 to 31: Project number

Address and authentication data

IP address of the CP

Since the CP always establishes the connection with TCSB, a dynamic IP address can be assigned to the CP by the Internet service provider.

Address and authentication data for communication with TCSB

The following information is required for the STEP 7 configuration of the CP for communication with TCSB:

- Parameters in the "Partner stations" parameter group
  - Partner IP address
    IP address or host name of the DSL router via which the telecontrol server is connected to the Internet.
    A fixed IP address is recommended.
  - Partner port (port number of the listener port of TCSB)
- Parameters in the "Security > CP identification" parameter group
  - Project number
  - Station number
  - Password (for authentication)

Acknowledgment

Acknowledgment in TeleControl Basic

The receipt of a sent frame is acknowledged as follows.

- Telecontrol communication
  Frames received from TCSB are acknowledged immediately by the CP.
  Frames sent by the CP are acknowledged by TCSB.
- Inter-station communication
  Received frames are acknowledged immediately by the CP. The acknowledgment frame is forwarded by the telecontrol server to the destination CP.
  For sent frames, this applies in the opposite direction.

SINAUT ST7

The TD7 software

Tasks of the TD7 software

In communications modules that use the telecontrol protocol SINAUT ST7, the software "SINAUT TD7" handles the communication and process data transfer between the subscribers. Only with TD7 software can data communication via WAN connections between CPs with ST7 capability and TIM modules be implemented.

TD7 provides change-driven process data transfer between the station and the control center (e.g. ST7cc), between individual stations and between the communications module and CPU. Failure of connections from CPUs or the control center are displayed. Once a problem has been corrected and when the CPUs or control center start up, the data is updated automatically.

Versions of the TD7 software

The TD7 software is available in two versions. Only one version can be alternatively used in communication modules that support both TD7 versions.

- TD7onTIM
  TD7onTIM is configured via the data points of the module. In productive operation, this runs in the communication module.
  The functions and parameters are described in the sections for configuring the data points.
  The version can be used in all ST7-capable modules.
  The advantages of this variant are:
  - Easy to configure
  - Independence of the TD7 software from the CPU program
  - Relief of the CPU when the CPU memory is low on reserves
- TD7onCPU
  TD7onCPU takes the form of a program block library. The blocks are used and configured in the CPU assigned to the TIM. The software runs during productive operation on the CPU.
  TD7onCPU can be used with the following modules:
  - TIM 1531 IRC
  - TIM 3V-IE Advanced
  - TIM 4R-IE / TIM 4R-IE Standalone

  The advantages of this variant are:
  - High speed if data changes
    Changes are detected with every CPU cycle.
  - Larger selection of data typicals for the master station and the station

The TD7onCPU libraries

TD7onCPU is available in two block libraries for the following SIMATIC families:

- S7-1500
  The blocks can run on an S7-1500.
- S7-300 / 400
  The blocks can run on an S7-300 and on an S7-400 CPU. Exceptions will be pointed out explicitly.
  Apart from process data transfer via WAN, the package is also suitable for local communication between CPUs, if they are connected via MPI. Here, as well, local connections and CPUs are constantly monitored and there is an automatic update after starting up or eliminating disruptions.

For a description of TD7onCPU, refer to the section AUTOHOTSPOT.

Communication between SINAUT objects

Communication between objects or typicals

- Object / typical
  In SINAUT ST7 the term "object" refers to the representation and handling of process variables such as messages, analog values, commands, motors, valves, controllers etc. The TD7 software always processes "objects".
  Objects in the TD7 world consist of processing instructions in the form of program blocks - known as typicals
  An object consists of the following components:
  - A function block (FB)
  - A data block (DB)
    The DB is the object data record that is assigned to the FB as an instance DB.

An object or a typical is always made up of a process part and an operator control and monitoring part (operator part) that operate in different subscribers in the SINAUT network. To be able to process their predetermined function the two parts must communicate with each other.

These two parts are used in the following subscriber types and have the following taks:

- Operator typicals
  Used in central stations (operator end)
  These blocks send setpoints, parameters, command and organizational instructions to the process typicals.
- Process typicals
  Used in stations (process end)
  These blocks manipulate process values, and return process data, alarm and status messages and organizational information to the operator typicals.

The data that describes the object is exchanged between the two communications partners. The data is located in the object data record in the data indexes 0...n. The extent and composition of this data area depend on the typical involved. It can consist of several identical or a combination of different data types. In the object data record at the process and operator end, the data structure of two typicals that belong together is identical

The data exchange does not necessarily run in both directions.

Object addressing

- Subscriber number
  Each CPU with TD7 software receives a SINAUT subscriber number that can be assigned in the range from 1 to 32000 and is unique throughout the network.
- Object number
  Each typical that is called in one of these CPUs has an instance DB whose number is identical to the SINAUT object number.

With the aid of the SINAUT subscriber number and the SINAUT object number, explicit addressing for the communication between the typicals that belong together can be implemented.

All data point typicals are function blocks (FBs). An instance DB must be specified when an FB is called. The number of this instance DB is identical to the object number of the data point object. At the send and receive ends this object number does not have to be identical.

To specify the communications relation, every typical as the following parameters

- **Partner number**
  "PartnerNo"
  Subscriber number of the partner being communicated with.
- **Partner object number**
  "PartnerObcetcNo"
  Object number (= instance DB no.) of this partner
- **Index**
  For the storage, the index number included in the data frame is taken into account.

Types of frame

After the process and operator data has been exchanged, organizational information (Org. frames) is transferred between the two ends. The following frame types (TA) are distinguished:

- TA 0: Spontaneous Org. frames
- TA 1: Requested Org. frames
- TA 2: Spontaneous data frames.
- TA 3: Requested data frames

This data flow of the Org. frames and the data areas in the object data record intended for this (Org. indexes) are not shown in the figure above.

Address parameters with the "ST7" protocol

Addressing communications partners and network nodes

Communication with the "SINAUT ST7" protocol can run via different paths and subnets. The following two parameters serve to address the individual subscribers in the ST7 network:

- Subscriber number
  The subscriber number is unique for every subscriber in a STEP 7 project. The following subscribers require a subscriber number:
  - Communications module
    ST7-capable modules (TIM, CP)
  - CPU
    The local CPU that is assigned to the communications module as the end point of a telecontrol connection receives a subscriber number via the ST7 protocol.
    With a CP and a TIM 3V-IE / TIM 4R-IE, the local CPU is assigned automatically via the rack.
    Assign the CPU to the TIM 1531 IRC in the "Subscriber numbers" parameter group of the TIM.
  - Application of the master station PC
    The application that you configure in the PC station of the master station.

  For the TIM and the assigned CPU, the subscriber number is configured in the "Subscriber numbers" parameter group of the TIM.
  For the application of the master station PC, the subscriber number is configured in the telecontrol connection: "Network data" editor > "Telecontrol" tab
- WAN address
  For every networked serial interface in a classic ST7 network a WAN address is assigned. It is unique in the respective WAN network.
  Since telecontrol connections can run via several network nodes and node stations, unique addressing of the interfaces involved is necessary throughout the network.
  The WAN address is configured in the parameter group "WAN parameters" of the serial interface.

Addressing the data

The process data to be transferred is configured as data points, see section ST7 data points (Page 1768).

Acknowledgment

Acknowledgment in ST7

- Communication with the master station
  - Data frames sent by the master station are acknowledged immediately by the CP when received.
  - Data frames sent by the communications module are acknowledged by the master station.
- Inter-station communication between stations
  Data frames are acknowledged during communication between station and master station. With inter-station communication acknowledgement frames of the receiving module are not forwarded by the master station to the sending communications module.

Expansion of SINAUT projects

Expansion of existing SINAUT projects

SINAUT projects with TIM modules for the SIMATIC families S7-300 and 400 that were configured in STEP 7 V5 can be expanded with newer modules that are configured in STEP 7 Professional.

To do this, in STEP 7 V5 projects, TIM modules are configured as placeholders (proxies) for modules that are configured in STEP 7 Professional. The configuration data (SDBs) are exported from STEP 7 V5 and imported into STEP 7 Professional.

Selection of the "Import" configuration procedure in the "Basic settings" parameter group is required for the import of configuration data from STEP 7 V5; see section Configuration (Page 1707).

The extension of SINAUT projects is supported by the following modules:

Table 1-138 Module migration from STEP 7 V5 to STEP 7 Basic / Professional (TIA Portal)

Module function for STEP 7 V5 project expansion			S7-1200/1500 module in STEP 7 Basic / Professional	
TIM (function) for expansion	Proxy to be used in the catalog		Compatible module	Required STEP 7 version
TIM 3V-IE Advanced	PROXY CP1243-8 IRC	⇒	CP 1243-8 IRC	STEP 7 Basic
TIM 3V-IE Advanced	PROXY CP1243-8 IRC	⇒	CP 1542SP-1 IRC	STEP 7 Professional
TIM 4R-IE	PROXY TIM 1531 IRC	⇒	TIM 1531 IRC	STEP 7 Professional
TIM 4R-IE Standalone	PROXY TIM 1531 IRC	⇒	TIM 1531 IRC	STEP 7 Professional

Note

TIM 4R-IE Stand-alone for S7-400 becomes TIM 1531 IRC

A TIM 4R-IE Stand-alone required in STEP 7 V5 that is assigned to a CPU-400 must be replaced by a TIM 1531 IRC for the expansion of classic SINAUT projects in STEP 7 Professional.

A TIM 4R-IE Stand-alone can only be created in new projects that are configured exclusively in STEP 7 Professional.

Note

No serial interface for CP 1542SP-1 IRC

For a CP 1542SP-1 IRC, no serial interface can be configured in STEP 7 V5 because the CP does not have a serial interface.

Requirements and module update

Requirements for the import are as follows:

- **STEP 7 V5 project**
  A configuration file from a consistent STEP 7 V5 project
- **Firmware version**
  One of the following firmware versions is required for the communication module:
  - TIM 1531 IC: V2.0
  - TIM 1531 IC: V1.1
    A TIM V1.1 can be replaced by a TIM V2.0, not vice versa.
    In STEP 7, the "Telecontrol configuration" method is set to "Configure". Existing configuration data is applied.
  - CP 1542SP-1 IRC: V2.0
    A CP V1.0 can be replaced by a CP V2.0, not vice versa.
    In STEP 7, the "Telecontrol configuration" method is set to "Configure". Existing configuration data is applied.
  - CP 1243-8 IRC: V3.1
    Transfer of data from STEP 7 V5 with modification options (see "Imported configuration data" below)
    A CP V3.0 can be replaced by a CP V3.1, not vice versa.
  - CP 1243-8 IRC: V2.1
    Adoption of the data from STEP 7 V5 without the option of making changes
    A CP V2.1 can be replaced by a CP V3.1, not vice versa. The CP used must have hardware product version 2.
    In STEP 7, the "Telecontrol configuration" method is set to "Import". Existing configuration data is applied.

    Note

    Disabling the partner configuration data during migration from STEP 7 V13 SP1

    If you migrate a CP 1243-8 IRC V2.1 from STEP 7 V13 SP1 to STEP 7 V15, the partner configuration data is disabled. You receive a message when compiling the project.

    In this case, repeat the import of the partner configuration data into the module.

  - TIM 3V... / TIM 4R...: V2.5.5

- **ST7 connections**
  For ST7 connections between a TIM 1531 IRC and a CPU:
  - The firmware of the CPU must have at least version V2.5.
  - Prior to import, the TIM and the CPU must be networked in STEP 7 Professional.
  - When configuring TD7onCPU, the CPU must not be assigned to the TIM in STEP 7 Professional.

  For ST7 connections between a TIM 1531 IRC and a PC application:
  - Prior to import, the TIM and the PC station must be networked in STEP 7 Professional.
- **CPU 1500 / PC station**
  For a CPU 1500 or a PC station with SINAUT application as an ST7 communication partner, the IP address in the STEP 7 V15 project must correspond to the address configured in STEP 7 V5.

Configuration in STEP 7 V5

For a description of the configuration, refer to the manuals of the modules involved.

DNP3

Addressing

To configure and commission the communications module, the following information is required:

Address information of the communications module

For the addressing, the following parameters must be configured for the module:

- Station address (DNP address on the data link layer)
- Address, depending on network type:
  - IP address and subnet mask; as an alternative: IP address of a DHCP server
  - Telephone number (for dial-up network)
  - WAN address (for dedicated line)
- Number of the station's own listener port (for connection requests from the master)

Address information of the master

The following information of the master is required for the configuration of the module:

- Station address (DNP address of the master on the data link layer)
- Address of the master, depending on network type:
  - IP address / subnet mask + port number of the listener port of the master
    or
    Name resolvable by DNS
    (You need the DNS server address, the DNS server must be accessible by the module.)
  - Telephone number (for dial-up network)
  - WAN address (for dedicated line)

Redundant DNP3 master

The two devices of a redundant master have an identical station address.

They only require different IP addresses or host names.

Configurations with connections over the Internet: VPN connections

For connections running via the Internet, dynamic IP addresses can be used.

To allow communication in both directions and to ensure that the data is protected during transfer, a connection with a VPN tunnel is necessary. For this the security modules of the SCALANCE S or SCALANCE M series are available.

Remember the following points when configuring:

- You configure the master IP address as normal.
- When configuring the interface of the module, configure the IP address of the router.
- You create the VPN configuration with SCALANCE S/M both for the station end and for the control center end in STEP 7.

Connection establishment

Connection establishment with DNP3

With the DNP3 protocol, the master establishes the connection (call operation / polling).

If an established connection is interrupted, a master module tries to re-establish the connection.

Note

Connection interrupted by the mobile wireless network provider

When using mobile wireless services, remember that existing connections can be interrupted by mobile wireless network providers for maintenance purposes.

Spontaneous mode

Enabling the "Unsolicited reporting" parameter in the "Unsolicited responses" parameter group in the "Settings DNP3 station" of an interface enables the unsolicited transmission of data from a station. In this case, dial-up network connections can be established by the communication module as client in order to send data of events spontaneously.

Direct communication between stations

Direct communication between two stations is possible if unsolicited mode is activated at the stations.

Connection establishment in Open User Communication and PG/OP communication

In Open User Communication in an S7 station, the CPU is the connection partner.

Connections are established as soon as the corresponding program blocks are called on the CPU.

This also applies to the situation when a different S7 station sends data. In this case, the receiver station calls the corresponding receiver modules.

Acknowledgment with the DNP3 protocol

Acknowledgment in DNP3

The basic acknowledgment mechanisms are configured for the data link layer, see section Transmission settings – DNP3 (Page 1808).

A station module acknowledges a request from the master with its response frames.

The acknowledgment of unsolicited frames that a station module sends to the master is configured in the "Settings DNP3 station" parameter group, see section Settings DNP3 station (Page 1811).

Transmission settings – DNP3

Transmission settings – DNP3

- **Max. time between Select and Operate**
  Max. time period (seconds) between command preselection (Select) and command input (Operate)
  - Master station:
    If no command input follows in the master station within the configured period after a command preselection, the command must be selected again. Having to enter the command twice reduces the risk of accidental operation in the master station.
  - Station:
    In order for a received command to be sent to the CPU and thereby take effect, the device must not receive any further command frames between Select and Operate.

  Range of values: 1..65535
  Default setting: 1
  If a master sends an object with function code 7 "DIRECT_OPERATE" to the station, the station does not evaluate the time period configured here.
- **Form of transfer**
  Defines the form of transfer of events.
  - Type-specific
    This is the bundled transfer of events according to data types typical for DNP3. First the existing binary events are sent, then all analog values and finally all counted value changes. This makes the frames somewhat more compact and transmission more efficient.
  - Chronological
    In this mode, events are transmitted strictly chronologically. The optimizing effect of grouping in blocks of the same data type as described in the DNP3 specification (see above) is lost. This mode is primarily intended for control systems that archive events strictly chronologically.

  Default: Type-specific

- **Connection monitoring time (application layer)**
  (Application layer confirmation timeout)
  Time (seconds) in which the module expects a sign of life on the application layer from its partner.
  - Master (master station):
    Time period in which the master expects a sign of life from the station.
    If the configured time is exceeded, the master sends a frame to the station. If this frame is not acknowledged by the station, the master classifies the station as not accessible, terminates the connection and then re-establishes it.
    Commands:
    If a connection with a station is interrupted or disrupted, command frames are repeated a maximum of three times before they are discarded by the master.
  - Station:
    In the station, the parameter is used to calculate the waiting times during connection establishment.

  With redundant paths, the following applies: If the acknowledgment is not received, the transmission path is classified as disrupted.
  Range of values: 0 ... 65535 s
  Default: 5
  With the setting 0 (zero), the function is disabled.
- **Partner monitoring time**
  If the station module does not receive a sign of life from the master on the application layer within the configured time, it classifies the connection as disrupted and terminates it.
  The master module expects a response from the station after sending data within the configured time.
  Range of values: 0 ... 65535
  Default: 0
  With the setting 0 (zero), the function is disabled.

Data link layer

- **Frame repetition**
  Maximum number of retries to send a frame if no acknowledgment is received from the partner on the data link layer.
  - For serial connections, repetitions can be useful if the connection quality is not good.
  - With Ethernet connections, it is recommended to configure the value 0, since the required repetitions are carried out by the protocol implementation.

  Range of values: 0 ... 255
  Default setting: 0
  With the setting 0 (zero), the function is disabled.
- **Acknowledgment**
  Condition under which the module requests an acknowledgment of receipt from the partner (sending the "CONFIRMED_USER_DATA" function code):
  - Never
    No acknowledgments are requested.
  - Always
    Acknowledgments are always requested.
  - When segmented
    Acknowledgments are only requested for data frames which are divided into several segments due to their length.

  With Ethernet connections, the DNP3 specification recommends that the acknowledgment requests are not used on the data link layer.
  On connections that are liable to disturbances, for example wireless links, the configuration of acknowledgment requests can be useful on the data link layer.
- **Acknowledgment monitoring time**
  Time period (seconds) in which a confirmation (acknowledgment) is expected from the partner on the data link layer. If no acknowledgment is received within the configured time period, data transmission is repeated, see the "frame repetition" parameter above.
  Range of values: 0...65535
  Default setting: 2
  The default value of 2 seconds usually only needs to be increased for slow serial connections.
  With the setting 0 (zero), the function is disabled.

Settings DNP3 master

Master-specific DNP3 parameters

You can find the following parameters in the "Settings DNP3 master" parameter group of the interfaces of the communication module set on the "Master station" network node type.

- **Polling basic interval**
  This is where you define the basic interval for station calls by the master station.
  Value range: 0 ... 65535 seconds
  Default: 30
  With the setting 0 (zero), the function is disabled. No cyclic polling takes place, not even for the parameters listed below, the calculation of which is based on the polling basic interval.
  Basis for further parameters
  The Polling basic interval is used to calculate the following parameters in the connection editor (editor "Network data > Telecontrol > DNP3"):
  - Event polling interval
  - Class 0 polling interval
  - Duration of the polling cycle, see "Polling mode" parameter.
- **Max. number of events per query**
  Maximum number of events that can be sent in the response frame of the station after being called by the master station.
  Range of values: 0 ... 65535
  Default: 0
  With the setting 0 (zero), the function is disabled (no limitation).

Settings DNP3 station

DNP3 parameters of the station / node station

You can find the following parameters in the "Settings DNP3 station" parameter group of the interfaces of the communication module set on the "Station" or "Node station" network node type.

Monitoring parameters

- Disturbance bit
  The station can use bit 1.6 of the Internal Indication Byte (IIN1.6) in the Application Response Header to indicate a fault in the CPU to the master.
  If this option is activated, the bit is set to "1" when the CPU is in STOP state.
- IP address check
  This parameter is used for security when establishing a connection between a station and the master.
  - Check IP address
    When this option is activated, the station checks the partner's IP address in addition to the station address when a connection is requested.
    If the IP address of the partner does not match the address of a configured partner of the station, the station rejects the connection.
  - Do not check IP address
    The station does not check IP addresses for incoming TCP connection requests.

Event properties

- Event class for image memory
  Selection of an event class in which only the last current values are stored in the send buffer. Older events are overwritten in the send buffer (image memory method).
  The parameter is not available for the TIM 3V.. / TIM 4R... These modules operate according to the "Default" setting.
  - Class x events
    Only the current value of events of the selected class is stored in the send buffer (image memory method).
  - Standard
    All values of events of classes 1 and 2 are stored in the send buffer.
    The current value is stored in the send buffer only for events of class 3 (image memory method).

  You can find details of how the image memory and send buffer work as well as the options for transferring data in the section Process image, type of transmission, event classes (Page 1765).
- Buffer for class 1 / 2 / 3 events
  For each of the three event classes, this is where you specify the number of events after which the stored events are sent to the communication partner.
  Note that the maximum size of the send buffer is divided over all connected DNP3 master stations.
  Permitted range: 1 ... 255
- Delay time class 1 / 2 / 3 events
  Here, for each of the three event classes you specify the maximum time in seconds for which the events can be stored in the send buffer before they are sent to the communications partner.
  This requires unsolicited transmission to be configured (see below).
  Permitted range: 0 ... 65535
  With the setting 0 (zero), the function is disabled.

Node station

This parameter is only available for interfaces for which the network node type is set to "Node station".

- Response without current station image
  With an interface of the "Node station" network node type, the communication module of the node station keeps the station process image in the image memory for each lower-level DNP3 station.
  - Option enabled
    The module replies to a class 0 call even if the process image in the image memory of the node station is not up to date. When a master station calls, the data of all connected stations is sent to the master station, even if the data of one or more stations is not up-to-date.
  - Option disabled
    The module replies to a class 0 call only if the process image in the image memory of the node station is up to date.

Unsolicited sending

- **Unsolicited reporting**:
  Transfer mode for events
  - Yes
    Event frames are transferred immediately.
    With this option, the station independently establishes a connection with the master.
    If unsolicited transmission in the master is disabled, the master sends the "DISABLE_UNSOLICITED" function code (unsolicited frames cannot be sent by the station). In this case, the station saves the events without sending them immediately.
  - No
    Event frames are only transmitted when data is requested from the master.

  For consistent configuration, you should select the same setting for the station and the master station.

  Note
  Setting for dedicated lines

  When connecting a serial interface to a DNP3 dedicated line with the "half-duplex" setting, you must set the parameter to "No".

  To avoid collisions, you should also use this setting when connecting to full-duplex multidrop dedicated lines.

- **Max. number of retries**
  Maximum number of repetitions of unsolicited frames if no acknowledgment is received from the master. After reaching the configured number, the station terminates the connection.
  Value range: 0...255
  Default setting: 3
- **Monitoring time for spontaneous frames**
  (Unsolicited Confirmation Timeout period)
  Time period (seconds) in which the station expects an acknowledgment of unsolicited frames from the master. If no acknowledgment is received within the configured time period, the data transmission is repeated.
  The sent frames are deleted from the send buffer only when the station receives the acknowledgment from the master.
  Value range: 1...65535
  Default setting: 5

DNP3 security options

Authentication and key exchange

When the (Secure Authentication) function is enabled, the DNP3 master and station authenticate themselves with a secret key, the pre-shared key.

With the help of the common pre-shared key, after the first connection establishment between master and station, session keys are agreed that are then renewed cyclically. Renewal of the session keys is normally initiated by the master.

The criteria for renewing the key are specified in the following parameters.

- Key exchange interval
- Authentication requests before key exchange

As soon as one of these conditions is met, the session key is renewed.

Parameters

- **Enable DNP3 security options**
  Select the option if you want to use Secure Authentication.
- **IKE mode**
  Selection of the mode for key exchange. Range of values:
  - Aggressive Mode
    The Aggressive Mode is somewhat faster but transfers the identity unencrypted.
  - Main Mode
    The Main Mode is the standard mode.

  Default setting: Aggressive Mode
- **Security statistics**
  Specifies whether the statistics of security events are sent to the master. Security events are authentication requests to the module. If this option is activated, all authentication request are stored in the module with date, time and result and sent to the master for further evaluation.
  Range of values:
  - Do not send security statistics
  - Send security statistics

  Default setting: Do not send security statistics
- **SHA-1 interlock**
  Setting to select whether the CP may use the secure hash algorithm SHA-1 if "SHA-256" was configured as the Secure hash algorithm and the master does not support SHA-256.
  Range of values:
  - SHA-1 mode not allowed
    The module must not use SHA-1. If the master does not support SHA-256, no connection will be established.
  - SHA-1 mode allowed
    The module can use SHA-1 if the master does not support SHA-256.

  Default setting: SHA-1 mode not allowed
- **Secure hash algorithm (SHA)**
  Selection of the Secure Hash Algorithm (SHA)
  Range of values:
  - SHA-1
  - SHA-256

  Default setting: 256

- **Key wrap algorithm**
  Selection of the Advanced Encryption Standard (AES)
  Range of values:
  - AES-128
  - AES-256

  Default setting: AES-128
- **Key length**
  Specifies the length of the pre-shared key in bytes.
  The following lengths are used depending on the key wrap algorithm.
  - For AES-128: 16 bytes
  - For AES-256: 32 bytes
- **Max. number of statistics queries**
  If the configured number of statistics requests of the master is exceeded within the key exchange interval, the module enters a message in the diagnostics buffer of the CPU.
  Range of values: 2...255 Default setting: 5
- **Authentication requests before key exchange**
  Maximum number of station authentication requests to the master. When this number is reached, the session key is renewed.
  Range of values: 1...10000 Default setting: 1000
  Recommendation: Set the number for the station twice as high as for the master.
- **Key exchange interval**
  Period after which the key is exchanged again between the station and the master. The interval must be matched up on both communications partners.
  Range of values: 0...65535 min. at 0 (zero), the key is never changed (function disabled).
  Default setting: 15 min.
  Recommendation: Set the key exchange interval for the station twice as high as for the master.
- **Authentication timeout**
  Maximum waiting time for the master to respond to an authentication request from the station.
  Exceeding the waiting time is considered an error by the station. In this case, the station generates a security event and sends it to the master.
  Range of values: 1... 65535 s Default setting: 5
- **Pre-shared key**
  The pre-shared key can be configured in two ways:
  - Manual configuration
    Enter the pre-shared key in STEP 7 manually as a hexadecimal value.
  - Import as file
    Import the pre-shared key from the file system of the engineering station if the pre-shared key was generated by the master or another system.

  The pre-shared key of a station module must be identical to the pre-shared key of the master.

IEC 60870-5

Addressing

To configure and commission the communications module, the following information is required:

Address information of the communications module

For the addressing, the following parameters must be configured for the module:

- ASDU address (address of the information object on the data link layer)
- Address, depending on network type:
  - IP address and subnet mask; as an alternative: IP address of a DHCP server
  - Telephone number (for dial-up network)
  - WAN address (for dedicated line)
- Number of the station's own listener port (for connection requests from the master)

Address information of the master

The following information of the master is required for the configuration of the module:

- ASDU address (address of the information object of the master on the data link layer)
- Address of the master, depending on network type:
  - IP address / subnet mask + port number of the listener port of the master
    or
    Name resolvable by DNS
    (You need the DNS server address, the DNS server must be accessible by the module.)
  - Telephone number (for dial-up network)
  - WAN address (for dedicated line)

Redundant IEC master

The two devices of a redundant master have an identical ASDU address.

They only require different IP addresses or host names.

Configurations with connections over the Internet: VPN connections

For connections running via the Internet, dynamic IP addresses can be used.

To allow communication in both directions and to ensure that the data is protected during transfer, a connection with a VPN tunnel is necessary. For this the security modules of the SCALANCE S or SCALANCE M series are available.

Remember the following points when configuring:

- You configure the master IP address as normal.
- When configuring the interface of the module, configure the IP address of the router.
- You create the VPN configuration with SCALANCE S/M both for the station end and for the control center end in STEP 7.

Acknowledgment with IEC protocol

Acknowledgment mechanisms for the protocol IEC 60870-5-104

Configuration: Interface of the module > "Advanced options" > Transmission settings - IEC 60870-5"

With each sent data frame, the RTU sends a continuous send sequence number. The data frame initially remains stored in the send buffer of the module.

Upon reception, the master sends the send sequence number from this or (if several data frames are received) the last data frame back to the module as an acknowledgment. The module stores the send sequence number returned by the master as the reception sequence number and uses it as an acknowledgment.

Data frames whose send sequence number is equal to or less than the current receive sequence number are evaluated as successful and deleted from the send buffer of the module.

Parameters:

- **k: Difference between send sequence number N(S) and receive sequence number N(R)**
  Maximum number of unacknowledged data frames (I-APDUs) as maximum difference between send sequence number N(S) and receive sequence number N(R).
  If k is reached and t_1 has not yet expired, the module does not send any data frames until all sent data frames have been acknowledged by the master.
  When k is reached and t_1 has elapsed, the TCP connection is terminated.
- **w: Max. number of unacknowledged data frames**
  Maximum number of received data frames (I-APDUs), after which the oldest data frame received from the master must be acknowledged.

Recommendations of the specification:

- w should not be greater than 2/3 k.
- Recommended value for k: 12
- Recommended value for w: 8

IEC-specific parameters

The following parameters can be found in the parameter groups for the master or station settings.

Some parameters are also displayed in the connection editor of the telecontrol connections and can be edited there in part.

IEC master

You can find the following parameters in the "Settings IEC master" parameter group of the interfaces of the communication module configured at the IEC network type and the "Master station" network node type.

- **Polling basic interval**
  This is where you define the basic interval for station calls by the master station.
  Value range: 0 ... 65535 seconds
  Default: 30
  With the setting 0 (zero), the function is disabled. No cyclic polling takes place, not even for the parameters listed below, the calculation of which is based on the polling basic interval.
  The basic interval is used for the calculation of the following parameters in the connection configuration:
  - Interval for general request
  - Interval for counter general request
  - Interval for group request
  - Interval for counter group request
- **Max. number of events per call**
  Maximum number of events that can be sent in the response frame of the station after being called by the master station.
  Range of values: 0 ... 65535
  Default: 0
  With the setting 0 (zero), the function is disabled (no limitation).

IEC station

You can find the following parameters in the "Settings IEC station" parameter group of the relevant interfaces of the communication module.

Station parameters

- Transmission mode
  This parameter determines whether ASDUs (events) of this station may be sent spontaneously to the master station.
  Range of values:
  - Spontaneous transfer
    The module can send spontaneous ASDUs (cause of transmission <3>).
  - No spontaneous transfer
    The module does not send spontaneous ASDUs.

  Default: Spontaneous

  Note

  Collisions on full duplex dedicated lines

  If you connect a serial interface to a dedicated line with the "half-duplex" connection type, you must disable spontaneous transfer.

  To avoid collisions, you should also disable Spontaneous transfer when connecting to full duplex multidrop dedicated lines.

- Polling mode
  This is where you define the mode in which the master station calls the station.
  The value configured for the station is transferred to the master station and stored there.
  Ranges of values:
  - Cyclic
    The station is called cyclically.
    The cycle time is determined by the master's "Class 0 polling interval" parameter.
  - Only after startup or connection re-establishment
    The station is only called after the initial startup and after a restart.
- Partner monitoring time
  If the communication module of the station does not receive a sign of life from the master within the configured time, it classifies the connection to the master as disrupted. The module terminates the connection and tries to re-establish it.
  Range of values: 0 ... 65535
  Default: 0
  With the setting 0 (zero), the function is disabled.

Other parameters

- General IEC-specific parameters
  See "Transmission settings - IEC 60870-5" for the respective interface of the module.
- Response to general request / Assignment to group request
  In the data point configuration, see section DNP3 and IEC data points (Page 1770).
- Call intervals
  In the connection configuration, see section Options (IEC) (Page 1749).

Additional information on mobile wireless CPs

CP 1242-7 (6GK7 242-7KX30-0XE0)

Operating modes of CP 1242-7

Modes of the CP

CP 1242-7 (6GK7 242-7KX30-0XE0) allows an S7-1200 to communicate as a GPRS station with a master or other remote networks via the GSM network. For communication using GPRS, the CP is set to one of the following operating modes:

- Telecontrol
  This operating mode of the CP allows the GPRS station to exchange data with the following partners:
  - Communication with the Telecontrol server
    This CP mode allows the GPRS station to exchange data with a telecontrol server.
    The Telecontrol server is a PC connected to the Internet with the "TCSB" application. It is generally located in the master station and is used for monitoring and control of the remote GPRS stations. Via the integrated OPC interface, data can be exchanged with the OPC client of a central control system.
    The Telecontrol server PC is not configured in STEP 7. The "TCSB" application has its own configuration user interface.
  - Communication with another remote GPRS station
    The data is transmitted via the Telecontrol server.
  - Communication with an engineering station (for TeleService)

  Communication with the Telecontrol server is performed via the GSM network and the Internet.
  This operating mode requires a SIM card with GPRS service enabled and a Telecontrol server that can be reached by the CP.
- GPRS direct
  This operating mode of the CP is used for direct communication between remote stations via the GSM network. No Telecontrol server is required.
  To allow network nodes in public wireless networks to be directly accessible, these need to be addressed using a fixed address. Here, SIM cards with a fixed IP address are used that allow the stations to address each other directly.
  The possible communications services and security functions (for example VPN) depend on what is offered by the network provider.
  Possible communications partners of the station with a CP 1242-7 in "GPRS direct" mode are:
  - A node that can be reached by the CP via an IP address (S7 station with CP 1242-7)
  - An engineering station (for TeleService)

See also

Connection establishment with a CP 1242-7 (Page 1822)

Connection establishment with a CP 1242-7

Connection modes of CP 1242-7 (6GK7 242-7KX30-0XE0)

- "GPRS direct" mode
  There are no different connection modes in the "GPRS direct" mode.
- "Telecontrol" mode
  The CP can be configured for the following connection modes.
  - "Permanent" connection mode
    There is a permanent TCP connection to the telecontrol server. Following connection establishment, there is a permanent TCP connection to the telecontrol server even if data is not transferred permanently.
  - "Temporary" connection mode
    A connection is only established to the telecontrol server when required.

If a TCP connection is established, process data is sent as soon as the telecontrol instructions are called on the CPU.

A connection is always established by the CP. If a connection established by the CP is interrupted, the CP automatically attempts to re-establish the connection.

Establishing a connection at permanent stations ("Telecontrol" mode)

In the "Telecontrol" mode, the permanent connection to the telecontrol server is established when the station starts up. If there is an interruption on the connection, the establishment of the connection can be triggered by a wake-up SMS (see below).

Establishing a connection at temporary stations ("Telecontrol" mode)

With temporary stations, connection establishment can be triggered by the following events:

- Event on the local CPU that needs to be evaluated by the program.
  In terms of the program, two situations need to be distinguished:
  - Events that lead to a single connection establishment (for example alarms or commands from the operator).
  - Expiry of an interval that leads to cyclic connection establishment (for example once daily for data transmission)
- Request by a communications partner (OPC client or S7 station)
  The request from the communications partner leads to the connection being established.
- Request for TeleService by an engineering station
  The request switched by the telecontrol server or TeleService gateway does not need to be evaluated in the program.
- Wake-up SMS of the telecontrol server
  The wake-up SMS can be triggered spontaneously on the telecontrol server. It is also possible to configure cyclic sending on the telecontrol server.

- Telephone wake-up call
  The wake-up call can be sent from a telephone that has a phone number authorized in the STEP 7 project. The telephone must support the CLIP function (transfer of its own call number).
  The connection establishment with the (main) telecontrol server is triggered.
- Telephone wake-up SMS
  The wake-up SMS can be sent from a telephone that has a phone number authorized in the STEP 7 project. The telephone must support the CLIP function (transmission of its own number) and sending of SMS.
  The connection establishment with the telecontrol server specified in the SMS is triggered.

When a temporary station is woken up, all the data is transferred if this has changed since the last data transfer.

Wake-up call / wake-up SMS in the "Telecontrol" operating mode

For wake-up call or wake-up SMS, see section Wake-up of cellular CPs (Page 1823).

Establishing a connection in "GPRS direct" mode

In "GPRS direct" mode, a connection establishment is triggered by the following events:

- Event on the local CPU that is evaluated by the program.
- Request by a communications partner (not an engineering station)
  The request from the communications partner is evaluated in the program by calling the telecontrol instructions.
- Request for TeleService by an engineering station
  The request switched by the telecontrol server or TeleService gateway does not need to be evaluated in the program.

Wake-up of cellular CPs

Right to wake-up by "Authorized phone numbers"

The CP only accepts a wake-up SMS or a wake-up call if the sending communication partner is authorized based on its phone number. These numbers are in configured for the CP in STEP 7 in the "authorized phone numbers" list.

Note

"Authorized phone numbers" in the STEP 7 project

- A phone number entered here gives the sender who transfers this phone number the right to trigger connection establishment.
- If an asterisk (*) is entered in the list, the CP accepts SMS messages from all senders.
- If the list is empty, the CP cannot be woken up for connection establishment.

Wake-up SMS

Depending on the connection type and the triggering server or intermediary TeleService server, the following text must be transferred in the wake-up SMS:

- For telecontrol connections:
  - Text for the wake-up SMS message for establishing a connection to the telecontrol server:
    `TELECONTROL`
  - Text for the wake-up SMS message for establishing a connection to the main telecontrol server:
    `TELECONTROL MAIN`
  - Text for the wake-up SMS message for establishing a connection to the substitute telecontrol server:
    `TELECONTROL BACKUP`

  The configuration of the telecontrol server for the GPRS CP is set in STEP 7 in "Telecontrol interface > Operating mode > main or substitute telecontrol server".

  Note

  Wake-up with a mobile phone

  - One of the texts listed above can be used in a wake-up SMS message.
  - With a wake-up call, the station always connects to the main telecontrol server.

- For TeleService connections:
  - Text for the wake-up SMS message for establishing a connection to the first configured TeleService server:
    `TELESERVICE`
    or
    `TELESERVICE 1`
  - Text for the wake-up SMS message for establishing a connection to the second configured TeleService server:
    `TELESERVICE 2`

  The configuration of the TeleService server for the GPRS CP is set in STEP 7 in "Telecontrol interface > TeleService authorization > 1st or 2nd TeleService server".

Preferred mobile wireless networks

Selection of the preferred mobile wireless networks

The following options are available to select the networks that the mobile wireless CP should preferably dial up:

- Automatic
  The CP dials with highest priority into the mobile wireless network of the contracted network provider set on the SIM card. If a dialing up the contract network fails, the CP dials up other mobile wireless networks with which the network provider has roaming contracts and whose access data is stored on the SIM card.
- Contract network only
  The CP only dials up the mobile wireless network of the contract network provider whose SIM card is plugged into the CP. No roaming.
- Contract network and preferred networks
  The CP dials up the contracted network with the highest priority. If dialing up the contract network is unsuccessful, the CP dials up alternative mobile wireless networks, in decreasing priority, as entered in the list of preferred network providers.
  The alternative networks are entered in the list as "Public Land Mobile Network" (PLMN).
  PLMN is a construct of Mobile Country Code (MCC) and Mobile Network Code (MNC).
  Example: 26276
  This is the PLMN for the test network of Siemens AG with MCC = 262 and MNC = 76.

Online functions

Download from device with CP 1243-8 IRC (V2.1)

Restrictions when downloading from a CP 1243-8 IRC

When you download the configuration data from a CP 1243-8 IRC using the online functions, the following data will not be fully downloaded:

- Partner station data
- Configuration and partner data of the data points
- References to PLC tags related to communication partners

In order to use the STEP 7 project after a complete download from the devices, you need to download the original export files from STEP 7 V5 with the configuration data of proxies into each CP 1243-8 IRC (Import partner configuration).

TeleService via mobile wireless

Download using TeleService

The following behavior applies to all mobile wireless CPs:

- CP 1242-7 (6GK7 242-7KX30-0XE0)
- CP 1242-7 GPRS V2 (6GK7 242-7KX31-0XE0)
- CP 1243-7 LTE-EU (6GK7 243-7KX30-0XE0)
- CP 1243-7 LTE-US (6GK7 243-7SX30-0XE0)

Connection resources for TeleService

The TeleService function occupies a connection resource on the engineering station.

The function download to device or upload from device during a TeleService session occupies a second connection resource on the engineering station.

Download to device

Only use the "Download to device" function with the mobile wireless CP via a TeleService connection as follows:

1. Select the CP in STEP 7.
2. Select the "Online" > "Download to device" menu.
3. In the "Extended download" dialog that appears, select the TeleService interface.
4. Download the project data from the "Extended download" dialog.

Download from device

The "Download from device" function via a TeleService connection is supported by the mobile wireless CPs with the following TeleService server applications:

- TeleControl Server Basic as of version V3
- TeleService Gateway as of version V3

TeleService via telecontrol

Communication path for TeleService via telecontrol

With TeleService via telecontrol, the connection between the engineering station and a remote S7 station always runs through a TeleService server as an intermediary.

The request for connection establishment is triggered by the engineering station and forwarded by the TeleService server. Depending on the available communication paths to the station, the TeleService server forwards the request to the next station via LAN or by a wake-up SMS. The telecontrol CP in the S7-1200 station establishes a connection to the engineering station via the TeleService server.

An alternative intermediary (TeleService server) is:

- A telecontrol server
  The telecontrol server can be a separate PC or installed on the engineering station in the form of the "TCSB" application.
- A TeleService gateway
  A TeleService gateway is used when no telecontrol server is available.

The TeleService server can be connected to the engineering station via LAN or Internet and the TeleService function can be called from there.

Requirements for TeleService via telecontrol

- A TeleService server
- The STEP 7 project with the required stations

Note on project engineering

The TeleService server is not configured in STEP 7.

Establish remote connection via telecontrol

Requirements for TeleService via telecontrol

For the requirements and mechanisms, see section TeleService via telecontrol (Page 1826).

Start a TeleService session via telecontrol

Start TeleService via telecontrol as follows:

1. In the project on the authorized engineering station, select the remote S7 station with which you want to establish a TeleService connection via WAN.
2. Alternatively, open the "Connect online" dialog box as follows:
   - "Connect online" button
   - "Connect online" shortcut menu (right mouse button)
   - "Online" > "Connect online" menu

   The "Connect online" dialog box opens.
3. Select the interface type "TeleService via Telecontrol" in the "Type of PG/PC interface" drop-down list.
4. In the "PG/PC interface" drop-down list, select the "TeleService board" option if it is not automatically displayed.
5. Click on the "Connect" icon next to the "PG/PC interface" drop-down list.
   The "Establish remote connection via telecontrol" dialog box opens.
6. Make the necessary settings in this dialog.
   Details of this are contained in the tooltip cascade of STEP 7.

The following details are required to successfully establish a connection:

Details required to establish a connection with the S7 station

The following information is necessary in the Establish remote connection" dialog:

- IP address or DNS name of the telecontrol server
- TCP port number of the telecontrol server or the DSL router via which the connection between the engineering station and the remote S7 station is running.
- Server password of the ES to authenticate the engineering station at the telecontrol server
  Only required if a group-specific password has been configured in the "TCSB" application.
- TeleService user name
  See CP configuration in STEP 7.
- TeleService password
  See CP configuration in STEP 7.
- Access ID of the CP
  Only required when there are several telecontrol CPs in the station. See CP configuration in STEP 7.

Status

Connection states with TeleService via telecontrol

The connection states described below can be displayed in the "Establish remote connection via telecontrol" dialog box.

When opening the dialog

- Not connected
  There is no connection to the remote S7 station. Connection establishment has not yet started.

When establishing a connection

If connection establishment was started by clicking the "Connect" button, the following states are displayed one after the other after the connection has been established:

- Connect to telecontrol server
  The engineering station connects to the telecontrol server.
- Wait for S7 station
  The wake-up SMS has been sent to the remote station. Wait for reply from station.
- Authentication at S7 station
  The S7 station has established an IP connection with the engineering station via GPRS and Internet and is checking the received logon and authentication data.
- Connected
  The station has successfully established the connection to the engineering station.

If connection establishment is unsuccessful

The following states may be displayed if the connection cannot be successfully established:

- Telecontrol server not accessible
  Possible causes:
  - The connection between the engineering station and the telecontrol server has been interrupted.
  - The telecontrol server is switched off.
- Wrong server password
  Cause: The wrong server password for logging on and authentication was entered at the telecontrol server.
- S7 station not responding
  Possible causes:
  - The communication between the telecontrol server and the station is disrupted.
  - There is a disrupted connection between the mobile wireless network and the Internet.
  - Faulty Internet connection.
  - The telecontrol server could not send a wake-up SMS.
  - The CP has not received a wake-up SMS.
  - The sender of the SMS was not configured in the list of authorized wake-up call numbers.
- Wrong TeleService user name or TeleService password
  Possible causes:
  - An incorrect TeleService user name or incorrect TeleService password was entered in the authentication dialog for the CP.
  - The TeleService user name or TeleService password was not configured in STEP 7.
- All TeleService access points are in use.
- The CP is not known to the telecontrol server.
  Cause: The CP originates from a STEP 7 project that does not match the project of the telecontrol server.
- No resources available for TeleService on the CP: Please contact the hotline.
- Protocol error
  Cause: Incompatible data or data from incompatible subscriber. Please contact the hotline.

Block library Telecontrol ST7

Creating a library

Delivery form of the block library

The program blocks of the TD7onCPU library are used as a global library for STEP 7.

The library is available archived and compressed on the Internet at the following address:
Link: (https://support.industry.siemens.com/cs/ww/en/view/109755374)

The file archive name has the following form: TD7_<version>.zip

Creating the library

To use the library in STEP 7, follow these steps:

1. Create a suitable directory in the STEP 7 installation directory of your engineering station, for example, one named "Telecontrol ST7".
2. Save the compressed and archived library (*.zip) in this directory.
3. Unpack the file archive using the utilities of the operating system.
   The library is now available as an archived global library under the name "Telecontrol ST7.zal15".
4. Open the STEP 7 project.
5. In the "Options" menu, select the command "Global libraries > Retrieve library".
   The "Retrieve archived global library" dialog opens.
6. Select the archive file.
7. Recommendation: Open the global library read-only.
   The "Open read-only" option is selected by default.
8. Click Open.
   The "Find folder" dialog opens. Select the destination directory in which the archived global library is to be unpacked.
9. Click "OK".
   The library is unpacked and opened.

You can find the library in the STEP 7 project in the library view (task card), "Global libraries" pane under the name "Telecontrol ST7_<Version>".

Structure of the library

Structure of the global library

The opened library contains the following directories:

- Types
  - PLC data types (UDT)
    Here you will find the PLC data types (UDTs) used by the libraries.
  - System data types (SDT)
    Here you will find the system data types used by the libraries.
- Master copies
  You can find the following program blocks here:
  - CPU300/400
    Program blocks for the S7-300/400 CPU
  - CPU1500
    Program blocks for the S7-1500 CPU

Master copies

The two libraries under "Master copies" for S7-300/400 and S7-1500 are divided into the following folders:

- Data point typicals
  Data point typicals (blocks) for transferring data
- Optional blocks
  Optional blocks that you can use when necessary.
- System blocks
  Blocks that are used for basic communication.
  Depending on the SIMATIC product series used, you can find the following auxiliary blocks for communication between TIM and CPU in separate folders:
  - BCom
    Blocks for processing the communication mailbox of the S7-400 and S7-1500
  - PCom
    Blocks for processing the communication mailbox of an S7-300 with P bus (CPU without partyline)
  - XCom
    Blocks for processing the communication mailbox of an S7-300 with X connections

Generating, using and calling the blocks

Generating TD7onCPU blocks

Once you have opened the TD7 library in the pane "Global libraries" under the name "Telecontrol ST7", you can generate the blocks for basic communication for one or more CPUs.

Requirements

The following requirements must be met before generating the blocks:

- The communication modules must be networked with each other.
- A TIM 1531 IRC must be networked with a CPU via Ethernet.
- A TIM 1531 IRC cannot be assigned to a CPU.
  In the "Subscriber numbers" parameter group, the "Selected CPU" box must be empty.
- No data point can be created in the TIM whose CPU is to use TD7onCPU.
- No message can have been created already in the message editor of the TIM.
- Valid telecontrol connections must be created between the communication partners involved.
- The "Telecontrol ST7" library must be open in the "Global libraries" task card.

Different requirements for modules with imported configuration data

Note the other requirements for communications modules whose SINAUT configuration was imported from a STEP 7 V5 project. With these modules, the option "Telecontrol configuration" is set to "Import" in the parameter group "Basic settings > Configuration".

These modules have different requirements for generating the blocks:

- No ST7 telecontrol connections need to be created between the communication partners involved.
  The relevant information is stored in the imported configuration data.
- The CPU of the communication module must have the same IP address as in the STEP 7 V5 project. The CPU is assigned to the communication module via its IP address.

Procedure for generating

Generate the blocks for basic communication as follows:

1. Select the desired TIM.
2. Select the shortcut menu command (right mouse button) "Generate TD7onCPU blocks".
3. In the dialog that follows, select one or more CPUs and click "Generate..."

The TD7onCPU blocks are created and compiled in the selected CPUs under the program blocks in a new "TD7onCPU" directory.

Using blocks

- System blocks
  The required blocks for basic communication are created automatically during the generation in the CPU.
  You do not need to use these blocks manually in the CPU.
  The system blocks handle central tasks such as startup, monitoring of the connections, reachability of the connection partners, entering data in the send mailbox or taking data from the receive mailbox of the communication DBs, general request, time keeping, handling communication etc.
  Communication DBs:
  - S7-300/400
    For each connection a separate communication DB is created automatically: It contains the send and receive mailbox and all the data required for controlling and monitoring this connection.
  - S7-1500
    For the CPU 1500, there is only one communication DB "BConnectData". A connection instance is stored in this DB in the "BConnection" array for each local connection.
- Data point typicals
  You need to use data point typicals on the relevant CPU according to the communications tasks.
  Drag the individual blocks with the mouse from the global library to the user program of the CPU. The corresponding instance data block is created automatically when using a data point typical.
  - Dat256D_S / Dat256D_R
    Copy also the UDT "TransmitBlock" for these typicals from the library (Types > PLC data types (UDT)) into the "PLC data types"directory of the CPU. The UDT is not automatically referenced by the typical.
- Optional blocks
  You should also use the optional blocks on the CPU depending on the communications tasks of the station.
  Drag the individual blocks with the mouse from the global library to the user program of the CPU.

Call in the user program

- System blocks
  Only the two following FCs need to be called in the user program:
  - StartUp
    Call the FC in the startup OB. The block has no parameters.
    To be able to call the FC StartUp, you must first create the startup OB (OB100).
  - BasicTask
    Call the FC in the cyclic OB.
    The DB BasicData is created automatically. The DB contains all the centrally required data including the records of all communication partners and the connections to be managed.
    For information on the various "BasicTask" FCs, see section Note on the "BasicTask_*" FCs (Page 1834).
- Data point typicals
  Depending on the communication tasks, call up all required data point typicals in the user program.
- Optional blocks
  Depending on the communication tasks, call up all required optional blocks in the user program.

See also section The cyclic SINAUT program (Page 1835) for information on calling the blocks.

Note

Error display for library conformance

In the "Compilation" parameter group of blocks, an error can be displayed under the entry "Library conformance" (call of single instances). This is caused by the call of global data blocks, for example, DB BasicData.

Since the "Telecontrol ST7" library has no "know-how protection", you can ignore these error notifications.

Note on the "BasicTask_*" FCs

Versions of the "BasicTask" FC

There are different versions of the "BasicTask" FC in the TD7 block library "Telecontrol ST7" for STEP 7 Professional.

Depending on the SIMATIC product series and the backplane bus, the FCs have a different suffix at the end of the block name:

- BasicTask_B
  For S7-1500 CPU
- BasicTask_B
  For S7-400 CPU

- BasicTask_X
  For S7-300 PU with partyline
- BasicTask_P
  For S7-300 CPU without partyline

Party line CPUs are: CPU 312.. / 313.. / 314.. / 315.. to CPU 315-2 DP and C7 devices.

For the block description, see section BasicTask_* FC (Page 1954).

"BasicTask" label for three FCs

References to the respective FC are made in many places in the following sections. Since the description of the different blocks usually applies to several SIMATIC product series, the three FCs are referred to hereafter as "BasicTask". The FC version for the relevant CPU type is meant in each context.

Structure of user program for TD7onCPU

The cyclic SINAUT program

Introduction

The structure of the TD7 blocks in the user program (OB1) is explained below.

The TD7 blocks must be proceeded in every OB1 cycle. Keep to the call sequence of the blocks unless instructed otherwise.

Other user-specific parts of the program can be linked in before or after the TD7 blocks in OB1 or, if practical, within the TD7 blocks.

You can structure the SINAUT-specific part in OB1 by calling it in lower-level FCs.

All data point typicals are FBs. An instance DB must be specified when an FB is called. The number of this instance DB is identical to the object number of the data point object.

The structure of the TD7 blocks in OB1

Cyclic OB1	
BasicTask	The respective FC BasicTask must always be called at the start. It handles basic SINAUT tasks that are always required.
Optional SINAUT basic functions	After FC BasicTask, call the following optional blocks (FCs) if you need them: • ListGenerator Creation of address lists for received frames with incomplete destination address. • TimeTask Provides the time of day.
	If necessary, call additional optional FCs in the subsequent program execution, for example: • Trigger Scheduled start of user programs and data frames • PartnerStatus Display the status of partners • PartnerMonitor Extended subscriber-specific display and control options. • PulseCounter Count pulse acquisition • PathStatus Display the status of the connection path to a partner **Note:** The following two blocks may not be called explicitly, since they are activated via an internal interface. • TestCopy • TestcopyDB
Data point typicals	In the subsequent program execution, call the data point typicals for sending and receiving data. The sequence of the individual typicals is unimportant. The number of typicals to call and the required types depend on the amount and type of data to be sent and received. **Note:** The instance DBs of these FBs are referenced using the partner object number of the communication partner.

Example of a TD7 program for a station

Cyclic OB1	
BasicTask	The respective FC BasicTask must always be called at the start of the cyclic program. Generally, data is ready for further processing after it has been received. The FC has only one parameter, "UserFC". Normally 0 can be specified. If you require user-specific processing for received data, specify here the number of an FC containing the program for this processing. You can also copy received data chronologically to an archive memory via this interface.
TimeTask	As an option, you can call the FC TimeTask immediately after FC BasicTask. The FC has no parameters. FC TimeTask TimeTask must be included if you need the time of day. This allows data frames to be time-stamped. However, you can also use the time of day to start program components at a specific point in time or to schedule the transmission of data frames. FC Trigger, described below, is then required. For this FC Time Task to be used, the CPU must be provided with the time from a local TIM module. You specify this in the configuration.
Trigger	FC Trigger can be included as an option. The FC sets its output for the duration of one OB1 cycle when the point in time or the time interval set for the FC has been reached. The FC can be inserted several times if several times or various time intervals are required. Using the FC requires that FC TimeTask must be called first in the OB1 program (see above) and the CPU time must have already been set once.
PartnerStatus	FC PartnerStatus can be included as an option. The FC shows the reachability for a maximum of 8 communications partners.
ListGenerator	• ListGenerator300 for S7-300 CPU • ListGenerator400 for S7-400 CPU • ListGenerator1500 for S7-1500 CPU The FC can be installed as an option. The FC is required if the station receives frames containing no destination address or an incomplete destination address. This happens if the configuration of the destination address has been completely or partially omitted in a partner for data point typicals. ("PartnerNo" and "PartnerObjectNo" were not specified, so transmission to all known destination subscribers takes place.)
PulseCounter PathStatus PartnerMonitor	When necessary

Cyclic OB1	
Data point typicals	Following the FCs for SINAUT basic tasks, data point typicals for sending and receiving data are called. The sequence of the individual typicals is unimportant. The number of typicals to call and the required types depend on the amount and type of data to be sent and received. The following typically applies to a station: • The following are sent: – Binary information, such as status messages and alarms – Analog values – Counted values • The following are received: – Commands – Setpoints, limit values, parameters
Bin04B_S	One or more FBs for the acquisition and transfer of binary information, such as messages, alarms, etc.
Ana04W_S	One or more FBs for the acquisition and transfer of analog values
Cnt01D_S / Cnt04D_S	One or more FBs for the acquisition and transfer of counted values A requirement for the use of the FBs is that FC PulseCounter is included in a cyclic interrupt OB, e.g. OB35. This FC is responsible for the time-driven acquisition of counted pulses in the background.
Cmd01B_R	One or more FBs for receipt and output of commands.
Set01W_R / Par12D_R	One or more FBs for receipt and output of setpoints, limit values or parameters..
Dat12D_S / Dat256D_S	One or more FBs for the acquisition and transfer of 12 data double words with any content. There is no data-specific processing and change control for these typicals. The user program is responsible for this. As an option change control can be activated that triggers a transfer with each bit change.

TD7 programs for master stations and node stations

In principle, the TD7 programs of master stations and node stations look the same as for a station. The corresponding receive typicals are used in a master station for the send typicals of a station.

In a node station, both send typicals and the corresponding receive typicals are used according to the transmission direction.

In the master station it is practical to structure the OB1 program according to stations, in other words, all data typicals belonging to the same station are called in one FC. The best overview is provided when the number of the FC is identical to the subscriber number of the station.

Cyclic interrupt OB

Introduction

Only include time-controlled TD7 blocks in a CPU if fast count pulses must be detected in this CPU which could not be reliably detected within an OB1 cycle due to an excessively long cycle time.

The count pulses are acquired via any digital input module. To acquire the pulses reliably, the digital inputs use must be queried for change at a fixed time interval. The time interval is based on the duration of the shortest count pulse. The minimum count pulse duration may be 50 ms. The same applies to the duration of the pause. This results in a maximum count frequency of 10 Hz.

The time interval in which the count pulse acquisition is performed must be approximately half the count pulse duration, i.e. at 50 ms at an interval of approximately 25 ms.

For this time-controlled count pulse acquisition, OB35 needs to be configured for an S7-300 CPU, one of the available cyclic interrupt OBs OB30 to OB38 for an S7-400 CPU, and a cyclic interrupt OB with a number from 30..38 / ≥ 123 for an S7-1500 CPU.

All cyclic interrupt OBs have a preset time interval, for OB35 for example 100 ms. This can be changed in steps of 1 ms. This makes it possible to set a cyclic interrupt OB, for example, to 25 ms.

You change the time interval for a cyclic interrupt OB in the Properties dialog (S7-400/1500), or in the Properties dialog of the CPU with S7-300.

Table 1-139 Calling FC PulseCounter in the cyclic interrupt OB

OB35	
Pulse Counter	One or more FC-PulseCounters can be included for the acquisition of count pulses. FC PulseCounter processes up to 8 pulse inputs of any digital input. The acquired count pulses are added together in programmable SIMATIC counters. These access the function blocks that put together the count value frames (FB-Cnt0x_S). User-specific cyclic interrupt functions required independently of the SINAUT program can be included at any point in the cyclic interrupt OB.

Programming error OB

Validity

Only for S7-300/400

Function

When a block that does not exist is called in an S7-300/400 CPU, the CPU normally changes to STOP. The diagnostics buffer indicates which FB, FC or DB was missing. You can then reload the missing block and restart the CPU.

If, however, you do not want the CPU to change to STOP if there is a missing block or only changes to STOP when certain block types or block numbers are missing, you can specify the reaction you require in OB121.

Even if you have loaded OB121 as an empty block on the CPU, this is enough to have the CPU continue running if a block is missing. If you want to decide more selectively when the CPU should continue running or change to STOP, include OB121 in the user program.

In conjunction with SINAUT ST7 it is possible that a CPU changes to STOP when it receives data from another CPU that it does not (or not yet) know. This is, for example, the case when you add a data point typical to a station and give it a complete destination address (destination subscriber no. plus destination object no.). The specified destination object no. can lead to a stop on the destination subscriber in the following situation:

- As soon as you have installed a new data point typical in a station, the data is transferred to the destination.
- If there is no corresponding receive typical installed on the destination CPU, the destination object no. (= instance DB of the received typical) does not exist.

The result is that the CPU changes to STOP as soon as the data is received.

To avoid this it is advisable to call FC ST7ObjectTest in OB121.

Table 1-140 Call of FC ST7ObjectTest in the programming error OB

OB121	
ST7ObjectTest	Calling FC ST7ObjectTest in OB121 prevents a CPU STOP, if the CPU receives data with an unknown destination object no. Other calls can be included at any point in OB121 regardless of the FC ST7ObjectTest call. FC ST7ObjectTest has a parameter "StopInOtherCases". Here you can specify what happens in other situations (STOP or continue running) if OB121 was called because another data block or an FB or FC is missing.

Types

The following data types are used in blocks of the S7-300/400 and S7-1500.

PLC data types (UDT)

UDTs used

The following PLC data type (UDT) is used by the block library:

- TransmitBlock
  UDT for the data transfer between the CPU and TIM
  When using the "Dat256D_S / Dat256D_R" typical, copy the UDT into the "PLC data types" directory of the respective CPU. The UDT is not automatically referenced by the typical. The UDT does not require any parameter assignment.

System data types (SDT)

SDTs used

The following system data types (SDTs) are used by the DB BasicData and stored automatically in the directory "PLC data types" of the CPU when generating/compiling the TD7 blocks:

- ConnectionDescription
  SDT is part of the "BasicData" DB and has an "Array[1..1] of ConnectionDescription" data type. It contains the following details for the connection description:
  - Number of the configuration DB
  - Connection type
  - CFB number in the case of an S7 connection
  - Subscriber number of the local TIM.
- SubscriberObject
  SDT is part of the "BasicData" DB and has an "Array[1..1] of SubscriberObject" data type. It contains the following information about the communication partner:
  - Subscriber number of the partner
  - Array index of the connection description
  - Subscriber type
  - Information whether the partner is local or remote.

Master copies (S7-300/400, S7-1500)

Validity

The following program blocks are located in the master copies of the global library:

- CPU300/400
  Program blocks for the S7-300/400 CPU
- CPU1500
  Program blocks for the S7-1500 CPU

The blocks are largely identical. Unless otherwise stated, the description applies equally to the blocks of the S7-300/400 and S7-1500.

If individual blocks cannot be used for all SIMATIC product series mentioned or if functions for the SIMATIC product series differ, this is expressly indicated.

Data point typicals

Types and overview of the data point typicals

Types of the data point typicals

Data point typicals process one or more data points of the same information type, e.g. 4 bytes of binary information, 4 analog values or 1 byte commands, etc.

When a data point typical (FB) is called the corresponding instance DB needs to be specified in which the data will be written or from which the data to be transferred will be read.

Transfer direction of the typicals

A data point typical for a specific type or amount of information always come in two versions:

- A typical for acquiring and sending
- A typical for receiving and outputting

When using data point typicals a distinction is therefore made according to the transfer direction:

- **Sending typicals**
  Sending typicals process data and send it to the remote partner.
  They have the ending "_S" Examples: Bin04B_S, Ana04W_S
- **Receiving typicals**
  Receiving typicals receive the data from their remote partner.
  They have the ending "_R" Examples: Bin04B_R, Ana04W_R

When transferring data on the two communicating partners there is always a corresponding pair of typicals involved.

Examples:

- Monitoring direction
  - The station sends binary data with the typical Bin04B_S.
  - The central station receives the data with the typical Bin04B_R.
- Control direction
  - The central station sends a command with the typical Cmd01B_S.
  - The station receives the command with the typical Cmd01B_R.

Structure of typical names

Note

No SMS blocks

Unlike the block library for STEP 7 V5, there are no blocks for sending SMS in the global library "Telecontrol ST7".

You configure the sending of e-mails and/or SMS in STEP 7 Professional in the TIA Portal via the message editor, see section Configuring data points and messages (Page 1750).

The 8-character names of the data point typicals are assigned according to the following scheme:

Table 1-141 Structure of 8-character typical names

1	2	3	4	5	6	7	8
Data point type			**Amount of data**		**Data format**	**Not used**	**S**
Bin = Binary information			(number of the type according to characters 6)		**X** = bit	(underscore)	Send function
Ana = Analog value					**B** = byte		**R**
Cnt = Count value					**W** = word		Receive function
Cmd = Command			Exception: The amount of data with "Dat256D_x" occupies three columns.		**D** = double word		**FS ***
Set = Setpoint, parameter					**R** = real		Send (Forced Send)
Par = Parameter							**FR ***
Dat = Data (any mixture of information types)							Receive (Forced Receive)

***** Blocks with the suffixes "_FS" and "_FR" are only available for the S7-1500.

Overview of the data point typicals

The following table provides an overview of the data point typicals.

Table 1-142 Overview of the data point typicals

Data format	Symbolic block name	Function
Binary value typicals		
Byte	Bin04B_S	Send 4 byte binary values
Byte	Bin04B_R	Receive 4 byte binary values
Analog value typicals		
Int	Ana04W_S	Send 4 analog values
Int	Ana04W_R	Receive 4 analog values
Real	Ana04R_S	Send 4 analog values as floating point number (CPU 1500 only)

Data format	Symbolic block name	Function
Real	Ana04R_R	Receive 4 analog values as floating point number (CPU 1500 only)
Counted value typicals		
UInt, Word *	Cnt01D_S	Send 1 counted value
UInt, Word *	Cnt01D_R	Receive 1 counted value
UInt, Word *	Cnt04D_S	Send 4 counted values
UInt, Word *	Cnt04D_R	Receive 4 counted values
Command typicals		
Byte, USInt *	Cmd01B_S	Send 1 byte commands (1-out-of-8)
Byte, USInt *	Cmd01B_FS	Send 1 byte commands (without 1-out-of-8 test)
Byte, USInt *	Cmd01B_R	Receive 1 byte commands (1-out-of-8)
Byte, USInt *	Cmd01B_FR	1 byte commands received (without 1-out-of-8 test)
Setpoint and parameter typicals		
Word	Set01W_S / Set01W_FS	Send 1 setpoint and receive current local setpoint
Word	Set01W_R	Receive 1 setpoint and send current local setpoint
ARRAY [1...12] of DInt / UDInt / DWord / Real *	Par12D_S / Par12D_FS	Send max. 12 double words with parameters and receive current local parameters
ARRAY [1...12] of DInt / UDInt / DWord / Real *	Par12D_R	Receive max. 12 double words with parameters and send current local parameters
Typicals for variable data types and amounts		
ARRAY [1...12] of DInt / UDInt / DWord / Real *	Dat12D_S	Send maximum of 12 double words with any data
ARRAY [1...12] of DInt / UDInt / DWord / Real *	Dat12D_R	Receive maximum of 12 double words with any data
DWord	Dat256D_S	Send maximum of 256 double words with any data
DWord	Dat256D_R	Receive maximum of 256 double words with any data

* USINT, UInt and UDInt are not supported at the same time by S7-300/400.

Reoccurring parameters

Description of the typical parameters

In the following description data point typicals that are identical except for the number of data points to be processed are described together.

In the tables of the data point typicals you will find the following information about the individual parameters:

- Parameter
  Name of the parameter
- Declaration
  Parameter type
  - INPUT
    Input parameter
  - OUTPUT
    Output parameter
  - IN_OUT
    In-out parameter
- Data type
  Data type supported for this parameter
- Range of values
- Default
  Preset value of the parameter
  If you do not configure individual parameters of a data point typical, the default value is used.
- Explanation
  Description of the function and the typical-specific properties of the parameter

Parameters used by many typicals

The following parameters are used by many of the typicals of the TD7onCPU library. These parameters are described here once and are not repeated in each instance in the following sections of the individual data point typicals.

Depending on the use of the typicals, some parameters are configured differently. Note the typical use described below.

PartnerNo

Parameter:	**PartnerNo**
Declaration:	INPUT
Data type:	INT
Range of values:	0 / 1 ... 32000
Default:	0
Explanation:	Subscriber number of the partner The subscriber number of the partner with which the block communicates must be specified.. • With a process typical this is normally the subscriber number of the master station or the application of the control center (e.g. ST7cc). • With an operator typical this is normally the subscriber number of a station.

Effects of the value 0 (zero) on various typical classes

- Sending process typicals
  (Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Dat12D_S, Dat256D_S)
  With the value 0 the data is sent to all subscribers to which an ST7 connection is configured. In this case, the parameter "PartnerObjectNo" is automatically sent with the value zero by the process typical.
  If PartnerNo is not found in the administration (DB BasicData), an entry to this effect is made in the diagnostics buffer (event ID B101). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.
  Notes
  - If the"PartnerObjectNo" is missing, there must be a list on the partner CPU from which the missing object number can be recognized (see ListGenerator1500/300/400 FC (Page 1927)).
  - Use of the block in a node station
    If the CPU of the node station maintains both connections to higher-level subscribers as well as to lower-level stations, a data frame with PartnerNo = 0 is transferred to all subscribers in the direction master station and in the direction stations.
- Receiving operator typicals
  (Bin04B_R, Ana04W_R, Cnt01D_R/Cnt04D_R, Dat12D_R, Dat256D_R)
  The value 0 is not permitted!
  If PartnerNo is < 1 or > 32000, an error message is entered in the diagnostic buffer (event ID B100).
  If the configured value is permitted and correct but "PartnerNo" is not found in the administration (DB BasicData), an entry is also made in the diagnostics buffer (event ID B101). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.
  If the CPU receives a data frame for this typical, a check is made to establish whether the source subscriber number in the data frame is identical to the "PartnerNo" configured here. If the two subscriber numbers are different, the received information is discarded and an error message is entered in the diagnostic buffer (Event ID B130).

- **Sending operator typicals**
  (Cmd01B_S, Set01W_S, Par12D_S)
  The value 0 is not permitted!
  If "PartnerNo" is < 1 or > 32000, an error message is entered in the diagnostic buffer (event ID B100).
  If the configured value is permitted and correct but "PartnerNo" is not found in the administration (DB BasicData), an entry is also made in the diagnostics buffer (event ID B101). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.
- **Receiving process typicals**
  (Cmd01B_R, Set01W_R, Par12D_R)
  Set the value 0 if the typical is to receive data from more than one partner, e.g. if it is to receive data from several control centers.
  If the CPU receives data for this typical and "PartnerNo" is > 0, a check is made to establish whether the source subscriber number in the data frame is identical to the "PartnerNo" configured here. If the two are different, the received information is discarded and an error message is entered in the diagnostic buffer (Event ID B130).
  This check is not made if "PartnerNo" is = 0. Regardless of the sender, each data frame addressed to the typical is passed on to the typical.
  If "PartnerNo" is > 0 and this number is not found in the administration (in DB-BasicData), an entry is made in the diagnostic buffer (event ID B101). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.
  Notes
  - If "PartnerNo = 0" is set, make sure that each partner sends the data with a complete destination address (target subscriber no. and target object no.).
  - Use of the block in a node station
    If the CPU of the node station maintains both connections to higher-level subscribers as well as to lower-level stations, a data frame with PartnerNo = 0 is transferred to all subscribers in the direction master station and in the direction stations.

PartnerObjectNo

Parameter:	**PartnerObjectNo**
Declaration:	INPUT
Data type:	INT
Range of values:	0 / 1 ... 32000
Default:	0
Explanation:	Object number of the partner The number of the object (= DB number) on the partner with which the block communicates.

Effects of the value 0 (zero) on various typical classes

- Sending process typicals
  (Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Dat12D_S, Dat256D_S)
  Setting the parameter to 0 makes sense if PartnerNo = 0 was set for the preceding parameter. If the"PartnerObjectNo" is missing, there must be a list on the partner CPU from which the missing object number can be recognized (see FC-ListGenerator).
  If the partner is an ST7cc control center specifying the "PartnerObjectNo" for this block can be omitted because in ST7cc there are no DBs as target objects. ST7cc decodes its data solely based on the source address in the data frame.
- Receiving operator typicals
  (Bin04B_R, Ana04W_R, Cnt01D_R/Cnt04D_R, Dat12D_R, Dat256D_R)
  The value 0 is not permitted!
  If the parameter assignment is incorrect (< 1 or > 32000), an error message is entered in the diagnostic buffer (event ID B102). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.
  If the CPU receives data for the object configured here, there is a check to establish whether the source object number in the data frame is identical to the "PartnerObjectNo" configured here. If they are different, the received information is discarded. An error message is entered in the diagnostic buffer (event ID B131).
- Sending operator typicals
  (Cmd01B_S, Set01W_S, Par12D_S)
  The value 0 is not permitted!
  If the parameter assignment is incorrect (< 1 or > 32000), an error message is entered in the diagnostic buffer (event ID B102). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.
- Receiving process typicals
  (Cmd01B_R, Set01W_R, Par12D_R)
  0 must be set in the following situations:
  - The partner is not an S7 CPU, i.e. there is no DB number as object. This is, for example, the case when the partner is an ST7cc control center.
  - There is more than one partner (PartnerNo = 0) from which the typical is to receive data. The corresponding objects of these partners will then generally have different numbers; in other words, no unique number can be specified here.

  If the CPU receives data for the object configured here and "PartnerObjectNo" is > 0, there is a check to establish whether the source object number in the data frame is identical to the "PartnerObjectNo" configured here. If they are different, the received information is discarded. An error message is entered in the diagnostic buffer (event ID B131).
  This check is not made if "PartnerObjectNo" is = 0. Regardless of the sender object, each data frame addressed to the object is also passed on to the receiving object.

Enabled

Parameter:	**Enabled**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	TRUE	
Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Enable block processing

- If processing is enabled, all the functions of the block execute.
- The response is different if processing has not been enabled:

Processing not enabled

- **With sending process typicals**
  (Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S,
  Set01W_R, Par12D_R,
  Dat12D_S, Dat256D_S)
  If processing is not enabled, the block can only communicate at the organizational level; in other words, Org frames can be sent and received.
  Please note: Although a query from an operator typical is answered, however the reply frame contains the data valid at the time the function was disabled.
  Note
  The response described here does not apply to Cmd01B_R, (see Command typical Cmd01B_R / Cmd01B_FR (Page 1873))!
- **With receiving operator typicals**
  (Bin04B_R, Ana04W_R, Cnt01D_R/Cnt04D_R,
  Set01W_S, Par12D_S,
  Dat12D_R, Dat256D_R)
  If processing is not enabled, the block can only communicate at the organizational level; in other words, Org frames can be sent and received.
  Please note: A request can still be sent and the answer received, the received information is, however, not output to the outputs. You will find the relevant outputs in the description of the appropriate data point typicals.
  Note
  The response described here does not apply to Cmd01B_S, see Command typical Cmd01B_S / Cmd01B_FS (Page 1871)!

ImageMemory

Parameter:	**ImageMemory**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE

Default:	TRUE	
Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Using in typicals	Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Set01W_R, Par12D_R, Dat12D_S	
Explanation:	Image memory principle for spontaneous data transmission • TRUE The data is transferred according to the image memory principle. The image memory principle reduces the memory required for storing data frames and produces as little data traffic on the WAN as possible. The default TRUE is the correct choice in most cases. • FALSE The data is transferred according to the send buffer principle. The send buffer principle is only required with data points whose individual data changes will be saved and transferred to the partner, for example alarms with time stamps.	

Conditional

Parameter:	**Conditional**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	TRUE	
Address range:	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Using in typicals	Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Set01W_R, Par12D_R, Dat12D_S	
Explanation:	Conditional spontaneous data transfer You will information on this below with the parameter "Unconditional".	

Unconditional

Parameter:	**Unconditional**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Address range:	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Using in typicals	Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Set01W_R, Par12D_R, Dat12D_S, Dat256D_S	
Explanation:	Unconditional spontaneous data transfer	

With the two parameters "Conditional" and "Unconditional" you decide whether a data frame is transferred immediately by the module if the value changes (unconditional spontaneous) or at a later point in time (conditional spontaneous).

The two parameters are configured as follows:

- **Conditional spontaneous transfer (not necessarily immediately)**
  - Conditional = TRUE
  - Unconditional = FALSE
- **Unconditional spontaneous transfer (immediately)**
  - Conditional = FALSE
  - Unconditional = TRUE

The default of the two parameters was chosen so that a data frame is not transmitted immediately.

The decision whether a data frame is transferred immediately or later only relates to dial-up networks.

In a dial-up network you must decide from case to case whether a change to the value of the data point requires immediate transfer and therefore the immediate establishment of a connection. This can, for example, be required for data points with alarms.

On a dedicated line, the transmission is always immediate even if the combination of "Conditional" and "Unconditional" is set to "not immediately". On dedicated lines, you do not need to make changes to the two parameter settings.

Permanent

Note

Parameter irrelevant

The parameter (INPUT, Bool) only appears in blocks for S7-300/400.

It is no longer supported functionally with S7-300/400, the value is always = FALSE.

TimeStamp

Parameter:	**TimeStamp**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Using in typicals	Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Dat12D_S, Dat256D_S, Set01W_R, Par12D_R
Explanation:	Time stamp • TRUE The data frame will be transferred with a time stamp. The prerequisite is that the time provided by the local TIM is available in the CPU. For more detailed information, refer to the description of FC TimeTask. **Note** With Ana04W_S remember the dependency of the time stamp on the parameter "MeanValueGeneration", see Analog value typical Ana04W_S (Page 1859). • FALSE The data frame is transferred without a time stamp. For information on the format of the time stamp, refer to the section Time stamp (Page 1853).

Note

Re-initialization

This parameter requires re-initialization of the CPU.

Recommendation:

Do not change the parameter in runtime or after the restart.

NewData

Parameter:	**NewData**	
Declaration:	OUTPUT	
Data type:	BOOL (Dat256D_R : DWORD)	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Output	Q 0.0 ... Q n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Using in typicals: Bin04B_R, Ana04W_R, Cnt01D_R/Cnt04D_R, Cmd01B_R, Set01W_S, Set01W_R, Par12D_S, Par12D_R, Dat12D_R, Dat256D_R

With Dat256D_R, the output occupies a double word with one status per segment.

Explanation: Receive new data

The NewData output is intended for user-specific further processing, for example to react in a specific way to receipt of new data.

Whenever the block has received new data and has output it to the outputs for the specific typical, NewData is set to TRUE for one OB1 cycle.

You will find the specific outputs in the description of the individual data point typicals.

With the data point typicals "Set01W_R" and "Par12D_R", NewData is also set to TRUE for one OB1 cycle if there is a new local value entered in the Local = 1 state.

If you do not require the parameter, simply leave it open.

Time stamp

Format of the SINAUT time stamp

For many data point typicals you can use the TimeStamp parameter to instruct that the data of the object should be transferred with a time stamp.

However for the receiving data point typicals there is no output parameter with which to output the received time stamp. The time stamp is only saved in the instance DB which you have specified when calling the respective receive typical. To further process the time stamp, the data must be read out of the DB by the user program.

The time stamp is saved in two data double words that have the same name in all object DBs:

Name of the double word	Contents
RecTimeStamp_1	Year, month, day and hour
RecTimeStamp_2	Minute, second, millisecond and time status

With the exception of the half byte with the time status, the date and time are coded in BCD format.

Table 1-143 Assignment of the structure of the time stamp

Name of the double word	Byte no.	Contents	
		High nibble	Low nibble
RecTimeStamp_1	0	Year * 10	Year * 1
	1	Month * 10	Month * 1
	2	Day * 10	Day * 1
	3	Hour * 10	Hour * 1

Name of the double word	Byte no.	Contents	
		High nibble	Low nibble
RecTimeStamp_2	0	Minute * 10	Minute * 1
	1	Second * 10	Second * 1
	2	Millisecond * 100	Millisecond * 10
	3	Millisecond * 1	Time status

Table 1-144 The bit assignment of the half byte "time status" (low nibble of byte 3 of RecTimeStamp_2)

Bit no.	Value	Meaning
0	0	Time is invalid
	1	Time is valid
1	0	Standard time
	1	Daylight saving time
2		Not used
3		Not used

The time double words occupy different addresses depending on the typical. Look in the instance DB or in the declaration header of the FB to find the absolute address of both double words.

It is more convenient to give the instance DBs symbolic names. You can then use the symbolic addresses to read out the information. In this case, you do not need to worry about the actual absolute addresses. These are used automatically by STEP 7. The following example clarifies this procedure.

Example

Symbolic name of instance DB: `ObjectDB27`

The STEP 7 program for reading the date and time of day and for saving in DB20 beginning with data byte 100 may appear as follows programmed in STL:

```
L "ObjectDB27".RecTimeStamp_1
T DB 20.DBD 100
L "ObjectDB27".RecTimeStamp_2
T DB 20.DBD 104
```

Analog value typical Ana04R_S

Validity

S7-1500 only

Function

Send 4 analog values as 32-bit floating-point number

Ana04R_S transfers the 4 analog values as instantaneous values. At the time of the transfer, the currently pending analog value is acquired and transferred to the partner.

Note

Common processing of the four analog values

The processing parameters such as threshold, smoothing factor etc. exist only once in a typical. These parameters apply to all 4 analog values in common; in other words, it is not possible to set the parameters for the individual analog values. For this reason, each typical should only acquire analog values that should be processed in the same way.

Parameter

Parameter: **PartnerNo**
PartnerObjectNo
Enabled
ImageMemory
Conditional
Unconditional
TimeStamp

For a description, see section Reoccurring parameters (Page 1844).

Parameter:	**ThresholdIntegration**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Threshold processing according to the integration principle With this parameter, you can specify whether the integration principle is used in threshold value processing. With the default FALSE the threshold value is calculated without integration. In this case, you can expect less data traffic on the telecontrol line and locally between CPU and TIM.

Parameter:	**ZeroLimitation**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE

Default:	TRUE
Explanation:	Zero limitation If this parameter is activated negative values are be suppressed and replaced with the value 0.

Parameter:	**TriggerInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Trigger input

With the edge change 0 → 1 of the "TriggerInput" input the transfer of the analog value frame can be triggered at a required time.

Example:
Time-driven analog value transfer with time stamp for supplying an analog value archive in the control center.

Make sure that you set the "ImageMemory" parameter to FALSE to prevent these data frames with time stamps from being overwritten when saving on the station TIM.

The FC Trigger block can be used for time-driven triggering of a transmission over "TriggerInput".

If you do not require the parameter, simply leave it open. The transfer should then be triggered based on the "ThresholdValue" and "ThresholdIntegration" threshold parameters.

"TriggerInput" actually only triggers transmission indirectly. With a 0 → 1 edge change at "TriggerInput", the data frame is put together with its current values and transferred to the local TIM. The TIM is responsible for the actual transmission to the partner. With dedicated lines or wireless networks the transfer is immediate. With a dial-up connection, it is possible that the data frame is saved first on the TIM and sent at a later point in time. The reason can, for example, be that the data frame is marked as "Conditional spontaneous", see parameter "Conditional".

Parameter:	**AnalogInput_1 ... _4**
Declaration:	INPUT
Data type:	REAL
Range of values:	See address range
Default:	0.0

Address range:	Bit memory address	MD0 ... MDn
	Data address	LD0 ... LDn DBm.DBD0 ... DBDn

Explanation: Analog input word

For each analog value to be transmitted in the data frame, you can specify from where the FB will take the analog information. Any combination of a data block and the bit memory address area can be used.

If you do not require parameters, simply leave them open. The value 0 is transferred for these analog inputs in the data frame.

Parameter: **SamplingPeriod**

Declaration: INPUT

Data type: INT

Range of values: 0 ... 32767 [ms]

Default: 500

Explanation: Acquisition interval for analog inputs in milliseconds

The acquisition interval is required for the following parameters:

- Threshold formation according to the integration principle (ThresholdIntegration)
- Smoothing the analog input value (SmoothingFactor)

The value must be selected high enough so that it is certain that a new value was acquired within the encryption time of the analog input. The interval to be specified has to be at least as long as the encoding time of the analog input module being used at the selected resolution (8 ... 15 bits).

The value must also be selected generously so that analog values are acquired even with the highest resolution and with analog modules with the highest number of inputs.

Specifying a "SamplingPeriod" that is too short may lead to an overflow of the internal accumulation counter.

Parameter: **ThresholdValue**

Declaration: INPUT

Data type: REAL

Range of values: +1.175495e-38 ... +3.402823e+38

Default: 1.0

Explanation: Threshold value

Without configuration, the default value 1.0 is used.

Point to note with "ThresholdValue" = 0: Changes are not checked based on the threshold value. The analog value frame will only be sent in the following situations:

- When there is a trigger via the "TriggerInput" input, typically a time-driven or event-driven trigger.
- When there is a general request to the station or a single request for the data frame.

Parameter: **SmoothingFactor**

Declaration: INPUT

Data type: REAL

Range of values: 1.0 (no smoothing)
4.0 (weak smoothing)
32.0 (medium smoothing)
64.0 (strong smoothing)

Default: 4.0

Explanation: Smoothing factor

Using the smoothing factor, quickly fluctuating analog values can be smoothed to a greater or lesser extent depending on the factor. This may allow a narrower threshold band to be selected (ThresholdValue).

The smoothing factors are identical to the smoothing factors that are configured for some S7 analog input modules. The smoothing is calculated using the same formula as on the input module:

$$y_n = \frac{x_n + (k-1)\ y_{n-1}}{k}$$

y_n = smoothed value in the current cycle n

y_{n-1} = smoothed value in previous cycle n-1

x_n = acquired value in the current cycle n

k = smoothing factor

Analog value typical Ana04R_R

Validity

S7-1500 only

Function

Receive 4 analog values as 32-bit floating-point number

Parameter

Parameter:	**PartnerNo** **PartnerObjectNo** **Enabled** For a description, see section Reoccurring parameters (Page 1844).

Parameter:	**AnalogOutput_1 … _4**	
Declaration:	OUTPUT	
Data type	REAL	
Range of values:	See address range	
Default:	0.0	
Address range:	Bit memory address	MD0 ... MDn
	Data address	LD0 ... LDn DBm.DBD0 ... DBDn
Explanation:	You can select where the individual analog values received by the FB are output. Addresses of a data block and in the bit memory address area can be mixed as required. If you do not require parameters, simply leave them open.	

Parameter:	**NewData** For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).
Typical-specific response:	Whenever the FB has received new data and has output it to the outputs "AnalogOutput_1" to "AnalogOutput_4", the "NewData" output is set to TRUE for one OB1 cycle.

Analog value typical Ana04W_S

Function

Send 4 analog values as 16 bit values

Ana04W_S transfers the 4 analog values alternatively:

- As instantaneous values
  At the time of the transfer, the currently pending analog value is acquired and transferred to the partner.
- As mean values
  The pending analog value is accumulated at selectable intervals. At the time of the transmission, a mean value is formed from the total value and transferred to the partner.

Note

Common processing of the four analog values

The processing parameters such as threshold, smoothing factor etc. exist only once in a typical. These parameters apply to all 4 analog values in common; in other words, it is not possible to set the parameters for the individual analog values. For this reason, each typical should only acquire analog values that should be processed in the same way.

Parameter

Parameter: **PartnerNo**
PartnerObjectNo
Enabled
ImageMemory
Conditional
Unconditional

For a description, see section Reoccurring parameters (Page 1844).

Parameter: **TimeStamp**

For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).

Typical-specific functions: With the setting TRUE the time stamp depends on the setting of the parameter "MeanValueGeneration":

- MeanValueGeneration = FALSE
  Instantaneous values are transferred in the data frame.
  The time stamp in the data frame is identical to the time of acquisition of the instantaneous values contained in the data frame.
- MeanValueGeneration = TRUE
  The data frame contains mean values.
  The time stamp is identical to the time at which the mean value calculation period was completed.
  The start of the mean value calculation period is not included in the data frame. This is, however, identical to the time stamp of the previously transferred mean value frame.

Parameter: **ThresholdIntegration**

Declaration: INPUT

Data type: BOOL

Range of values: TRUE / FALSE

Default:	FALSE
Explanation:	Threshold processing according to the integration principle With this parameter, you can specify whether the integration principle is used in threshold value processing. With the default FALSE the threshold value is calculated without integration. In this case, you can expect less data traffic on the telecontrol line and locally between CPU and TIM. When MeanValueGeneration = TRUE, (analog values are sent as mean values), the "ThresholdIntegration" parameter has no meaning.

Parameter:	**ZeroLimitation**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	TRUE
Explanation:	Zero limitation If this parameter is activated negative values are be suppressed and replaced with the value 0.

Parameter:	**TriggerInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Trigger input

With the edge change 0 → 1 of the "TriggerInput" input the transfer of the analog value frame can be triggered at a required time.

Example:
Time-driven analog value transfer with time stamp for supplying an analog value archive in the control center.

Make sure that you set the "ImageMemory" parameter to FALSE to prevent these data frames with time stamps from being overwritten when saving on the station TIM.

If the block calculates mean values, the duration of the calculation period is decided by the "TriggerInput" input. The current period is ended and a new period begun each time a transmission is triggered by this input. The interval between triggering a data frame twice determines the duration of the mean value calculation period.

The FC Trigger block can be used for time-driven triggering of a transmission over "TriggerInput".

If you do not require the parameter, simply leave it open. The transfer should then be triggered based on the "ThresholdValue" and "ThresholdIntegration" threshold parameters.

"TriggerInput" actually only triggers transmission indirectly. With a 0 → 1 edge change at "TriggerInput", the data frame is put together with its current values/ mean values and transferred to the local TIM. The TIM is responsible for the actual transmission to the partner. With dedicated lines or wireless networks the transfer is immediate. With a dial-up connection, it is possible that the data frame is saved first on the TIM and sent at a later point in time. The reason can, for example, be that the data frame is marked as "Conditional spontaneous", see parameter "Conditional".

Parameter:	MeanValueGeneration
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Explanation: Mean value generation

If the parameter is enabled the analog values to be acquired are transferred as mean values.

If you select mean value generation, the currently pending analog value is acquired cyclically and accumulated. The acquisition cycle depends on the "SamplingPeriod" parameter (for example 500 ms, see also the description of this parameter). The mean value is calculated from the accumulated values as soon as a transmission is triggered via the "TriggerInput" input. Following this, the accumulation starts again so that the next mean value can be calculated.

The mean value can also be calculated if the transmission of the analog value frame is triggered by a general or single request. The duration of the mean value calculation period is then the time from the last transmission (for example triggered via TriggerInput) to the time of the general or single request. Once again, the accumulation restarts so that the next mean value can be calculated.

If the acquired analog value is above or below the permitted range (7FFFH bzw. 8000H), this value can either be taken into account immediately in the calculation of the mean value or it can be suppressed for a specific period for the calculation of the mean value. The required response can be decided with the "FaultSuppressionTime" parameter:

- **FaultSuppressionTime = 0**
  Acquisition of a value above or below the over- or underrange results in immediate cancellation of the mean value calculation. The value $7FFF_H$ or 8000_H is saved as an invalid mean value for the current mean value calculation period and sent when the next analog value frame is triggered. The calculation of a new mean value is then started. If the analog value remains in the overshoot or undershoot range, this new value is again saved immediately as an invalid mean value and sent when the next frame is triggered.
- **FaultSuppressionTime > 0**
  If the acquired analog value is in the overshoot or undershoot range, the bad values are excluded from the calculation of the mean value for a maximum duration as defined by the FaultSuppressionTime. If this time is exceeded, the value $7FFF_H$ or 8000_H is saved as an invalid mean value and sent when the next analog value frame is triggered. The procedure is identical in each new mean value calculation period averaging period; in other words, bad values are again suppressed for the duration of the "FaultSuppressionTime". The duration of the "FaultSuppressionTime" also indirectly decides the proportion of invalid values per mean value calculation period. For example, if the mean value is calculated every 15 minutes and "FaultSuppressionTime" is set to 5 minutes, the mean value is only sent as invalid when more than 1/3 of the analog values acquired are above or below the overshoot or undershoot range in the current mean value calculation period.

Parameter:	**AnalogInput_1 ... _4**
Declaration:	INPUT
Data type:	WORD
Range of values:	See address range

Default:	0 (W#16#0)	
Address range:	I/O words	PIW0 ... PIWn
	Memory words	MW0 ... MWn
	Data words	LW0 ... LWn DBm.DBW0 ... n

Explanation: Analog input word

For each analog value to be transmitted in the data frame, you can specify from where the FB will take the analog information. I/O words from analog input modules, data words from a data block and memory words can be mixed as required.

If you do not require parameters, simply leave them open. The value 0 is transferred for these analog inputs in the data frame.

Parameter: **SamplingPeriod**

Declaration: INPUT

Data type: INT

Range of values: 0 ... 32767 [ms]

Default: 500

Explanation: Acquisition interval for analog inputs in milliseconds

The acquisition interval is required for the following parameters:

- Threshold formation according to the integration principle (Threshold Integration)
- Smoothing of the analog input value (Smoothing Factor)
- Mean value generation

The value must be selected high enough so that it is certain that a new value was acquired within the encryption time of the analog input. The interval to be specified has to be at least as long as the encoding time of the analog input module being used at the selected resolution (8 ... 15 bits).

The value must also be selected generously so that analog values are acquired even with the highest resolution and with analog modules with the highest number of inputs.

If mean values are calculated, SamplingPeriod should not be less than 500 ms. If mean values are calculated over very long periods, the time must be increased as follows:

- Mean value calculation period 12 h: SamplingPeriod = 1000 [ms]
- Mean value calculation period 24 h: SamplingPeriod = 2000 [ms]

Specifying a "SamplingPeriod" that is too short may lead to an overflow of the internal accumulation counter. The maximum value of 2 147 483 647 of a double integer must not be exceeded. When an overflow is detected, the invalid mean value of 8000_H is transferred for the current mean value calculation period.

Parameter: **ThresholdValue**

Declaration: INPUT

Data type:	INT
Range of values:	0 / 1 ... 32767
Default:	270
Explanation:	Threshold value

When specifying the threshold value, take the encryption range of the analog values into account. Raw values from S7 analog inputs are always encoded in the range from 0 ... 27648 (= 0 ... 100 %) or + 27648 (= + 100%). Depending on the resolution of the analog input, the value jumps from 128 (with 8-bit resolution) or 1 (with 15-bit resolution). If the acquired analog values have a different encoding range, specify a threshold value oriented toward this.

If the parameter is not configured, the default value of 270 is used. This corresponds to approximately 1% of the normal S7 analog raw value range.

Point to note with "ThresholdValue" = 0

Changes are not checked based on the threshold value. The analog value frame will only be sent in the following situations:

- When there is a trigger via the "TriggerInput" input, typically a time-driven or event-driven trigger.
- When there is a general request to the station or a single request for the data frame.
- When the analog value moves into the overshoot or undershoot range ($7FFF_H$ or 8000_H) (possibly after the suppression time set for "FaultSuppressionTime" has elapsed).

When MeanValueGeneration = TRUE, i.e. the analog values are sent as mean values, the "ThresholdValue" parameter has no meaning.

Parameter:	**SmoothingFactor**
Declaration:	INPUT
Data type:	INT
Range of values:	1 (no smoothing) 4 (weak smoothing) 32 (medium smoothing) 64 (strong smoothing)

Default: 1

Explanation: Smoothing factor

When MeanValueGeneration = TRUE, i.e. the analog values are sent as mean values, the "ThresholdValue" parameter has no meaning.

Using the smoothing factor, quickly fluctuating analog values can be smoothed to a greater or lesser extent depending on the factor. This may allow a narrower threshold band to be selected (ThresholdValue).

The smoothing factors are identical to the smoothing factors that are configured for some S7 analog input modules. The smoothing is calculated using the same formula as on the input module:

$$y_n = \frac{x_n + (k-1)\ y_{n-1}}{k}$$

y_n = smoothed value in the current cycle n

y_{n-1} = smoothed value in previous cycle n-1

x_n = acquired value in the current cycle n

k = smoothing factor

Parameter: **FaultSuppressionTime**

Declaration: INPUT

Data type: INT

Range of values: 0 ... 32767

Default: 0

Explanation: Fault suppression time in seconds.

Transmission of an analog value located in the overshoot or undershoot range ($7FFF_H$ or 8000_H) is suppressed for the time period specified here. The value $7FFF_H$ or 8000_H is only sent after this time has elapsed, if it is still pending. If the value returns to below $7FFF_H$ or above 8000_H again before this time elapses, it is immediately sent again as normal. The suppression time is started again for the full duration the next time $7FFF_H$ or 8000_H is acquired.

This is typically used for temporary suppression of current values that may occur when powerful pumps and motors are started. The analog input may exceed the maximum range several times under some circumstances. Suppression prevents these values from being signaled as faults in the control center system.

The suppression is adjusted to analog values that are acquired by the S7 analog input modules as raw values. These modules return the specified values for the overflow or underflow range for all input ranges (also for life-zero inputs). With provided ready values, fault suppression is only possible if these also adopt the values $7FFF_H$ or 8000_H when there is overshoot or undershoot. If this is not the case, the parameter does not need to have a value entered.

The parameter can also be used in combination with the mean value calculation for temporary suppression of the values $7FFF_H$ or 8000_H (see parameter MeanValueGeneration).

If no parameter is specified, the default of 0 seconds applies. An acquired value of $7FFF_H$ or 8000_H is then sent immediately when it is first detected or, with mean value calculation, as an invalid mean value for the current mean value calculation period.

Analog value typical Ana04W_R

Function

Receive 4 analog values as 16-bit values

Parameter

Parameter: **PartnerNo**
PartnerObjectNo
Enabled
For a description, see section Reoccurring parameters (Page 1844).

Parameter: **AnalogOutput_1 … _4**
Declaration: OUTPUT
Data type WORD
Default: TRUE
Explanation 0 (W#16#0)

Range of values:	I/O words	PQW0 ... PQWn
	Memory words	MW0 ... MWn
	Data words	LW0 ... LWn DBm.DBW0 ... n

You can select where the individual analog values received by the FB are output. I/O words from analog output modules, data words from a data block and memory words can be mixed as required.

If you do not require parameters, simply leave them open.

Parameter: **NewData**

For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).

Typical-specific response: Whenever the FB has received new data and has output it to the outputs "AnalogOutput_1" to "AnalogOutput_4", the "NewData" output is set to TRUE for one OB1 cycle.

Binary value typical Bin04B_S

Function

Send 4 bytes of binary information

Parameter

Parameter: **PartnerNo**
PartnerObjectNo
Enabled
ImageMemory
Conditional
Unconditional
TimeStamp

For a description, see section Reoccurring parameters (Page 1844).

Parameter: **InputByte_1 … _4**
Declaration: INPUT
Data type: BYTE
Range of values: See address range
Default: 0 (B#16#0)

Address range:	Input bytes	IB0 ... IBn PIB0 ... PIBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n

Explanation: Input byte

Specify the memory area (1 to 4 bytes) of the binary information to be transferred by the FB.

Input bytes from the process input image, I/O bytes directly from digital input modules, data bytes from a data block and memory bytes can be mixed.

If you do not require parameters, simply leave them open.

For unconfigured bytes, the value 0 (zero) is transferred.

Parameter: **DisableMask**

Declaration: INPUT

Data type: DWORD

Range of values: 0 ... 2147483647

- As 32 bit binary number
  2#0 ... 2#11111111_11111111_11111111_11111111
- As 32 bit hexadecimal number
  DW#16#0 ... DW#16#FFFF_FFFF

Default: 0 (2#0)

Explanation: Disable mask

- For every input to be blocked enter a 1 at the relevant position in the bit pattern.
- For the other inputs enter a 0.

A disabled input always has the value 0 (zero) during the transfer.

For the assignment of the 32 inputs from "InputByte_1" to "InputByte_4" to the 32 bits of the blocking mask, see the following table.

	InputByte_1								**InputByte_2**								**InputByte_3**								**InputByte_4**							
Bit	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0
2#	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
DW#16#	_				_				_				_				_				_				_				_			

Parameter: **InversionMask**

Declaration: INPUT

Data type: DWORD

Range of values: 0 ... 2147483647
- As 32 bit binary number
  2#0 ... 2#11111111_11111111_11111111_11111111
- As 32 bit hexadecimal number
  DW#16#0 ... DW#16#FFFF_FFFF

Default: 0 (2#0)

Explanation: Inversion mask

The inversion of input signals can, for example, be useful when using a mixture of sensors operating on the open and closed circuit principle.

- For every input to be inverted enter a 1 at the relevant position in the bit pattern.
- For the other inputs enter a 0.

For the assignment of the 32 inputs from "InputByte_1" to "InputByte_4" to the 32 bits of the inversion mask, see the following table.

	InputByte_1								InputByte_2								InputByte_3								InputByte_4							
Bit	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0
2#	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
DW#16#	_				_				_				_				_				_				_				_			

Binary value typical Bin04B_R

Function

Receive 4 bytes of binary information

Parameter

Parameter: **PartnerNo**
PartnerObjectNo
Enabled
For a description, see section Reoccurring parameters (Page 1844).

Parameter: **OutputByte_1 … _4**
Declaration: OUTPUT
Data type: BYTE
Range of values: See address range.
Default: 0 (B#16#0)

Address range:	Output bytes	QB0 ... QBn PQB0 ... PQBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n

Explanation: Output byte

Specify the memory area (1 to 4 bytes) of the binary information to be output in the binary information.

Output bytes from the process image output, I/O bytes directly to digital output modules, data bytes from a data block and memory bytes can be mixed.

If you do not require parameters, simply leave them open.

For information on reading out the time stamp received with the data using the user program, see the section Time stamp (Page 1853).

Parameter: **NewData**

Explanation: For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).

The output "NewData" is always set to TRUE for one OB1 cycle when the FB has received new data and output it to the output bytes "OutputByte_1" to "OutputByte_4"

Command typical Cmd01B_S / Cmd01B_FS

Function

Send 1 byte commands with 1-out-of-8 check

The 1-out-of-8 check is performed by the data point typical.

The 1-out-of-n check is performed by FC Safe.

Note

FC Safe required

With Cmd01B_S, data can only be transmitted when FC Safe is linked in cyclic program.

Parameter

Parameter: **PartnerNo**

PartnerObjectNo

For a description, see section Reoccurring parameters (Page 1844).

Parameter:	**Enabled**
	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).
Typical-specific response:	Enable block processing
	If processing is disabled the FB only checks to see if the disabled status has been canceled. The block cannot communicate at the organizational level in this status because it cannot send any Org. frames.

Parameter:	**CommandInputByte_HW**	
Declaration:	INPUT	
Data type:	BYTE	
Range of values:	See address range	
Default:	0 (B#16#0)	
Address range:	Input bytes	IB0 ... IBn PIB0 ... PIBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n
Explanation:	Command input byte for hardware input.	
	This command input byte is specially designed for entering commands using hardware, i.e. via digital inputs. Input using memory or data bytes is also possible, but you must then make sure that the command pending at the input byte is reset, which occurs during hardware input when the command button is released.	
	When input is detected, if no error is detected during the 1-out-of-8 and 1-out-of-n check, and if the central enable memory bit is set, the command is transferred. This is automatically set by FC Safe following a selected time delay set there (see FC Safe, "InputDelayTime" parameter).	
	If a 1-out-of-8 or 1-out-of-n error is detected, the entered command is no longer processed. A new command is only read in again if previously for the time of one OB1 cycle no hardware command was detected in the CPU at this or another command input block with a hardware input.	
	The FB enters a detected 1- out-of-8- or 1-out-of-n error in the diagnostic buffer (event ID B171 or B172). The error status is also indicated via the "InputError" output of FC Safe (see FC Safe, "InputError" parameter) and continues to be indicated as long as the error remains.	

Name:	**CommandInputByte_SW**
Declaration:	IN_OUT
Data type	BYTE
Range of values:	See address range
Default:	0 (B#16#0)

Address range:	Memory bytes	MB0 ... MBn
	Data bytes	DBm.DBB0 ... n
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation	Command input byte for software input This command input byte is specially designed for entering commands using software, i.e. by the user program or an operator panel (OP). When an input is detected and if no error is detected during the 1-out-of-8 and 1-out-of-n check, the command id reset at the input byte and transferred. The central enable memory bit is ignored here because it is only intended for command input over hardware (see "CommandInputByte_HW"). If a 1-out-of-8 or 1-out-of-n error is detected, the entered command is no longer processed. A new command is only read in again if previously for the time of one OB1 cycle no software command was detected in the CPU at this or another command input block with a software input. The FB enters a detected 1- out-of-8- or 1-out-of-n error in the diagnostic buffer (event ID B171 or B172). Appropriate error bits are also set in the central data block BasicData where they can be queried by the software. For further details, refer to the description of FC Safe. In principle it is possible to create a new command at "CommandInputByte_SW" in every OB1 cycle. However, only one command per OB1 cycle is allowed and this applies to all command input blocks with software input (1-out-of-n check). An ’empty cycle’ between two consecutive software commands is therefore not necessary.	

Note

CommandInputByte_HW / CommandInputByte_SW

If the same commands need to be entered over the hardware and software, the two command inputs "CommandInputByte_HW" and "CommandInputByte_SW" can also be used at the same time.

If a command is entered at the same time over both input bytes, this is only accepted when exactly the same command is entered over the hardware as well as the software input. The hardware input is then processed further.

In all other cases the input is rejected and an error message is entered in the diagnostic buffer (Event ID B170). The error status is also indicated via the InputError output of FC Safe. Appropriate error bits are set in the central data block BasicData where they can be queried by the software (see FC Safe).

Command typical Cmd01B_R / Cmd01B_FR

Function

Receive 1 byte commands (1-out-of-8 format)

Parameter

Parameter: **PartnerNo**
PartnerObjectNo
For a description, see section Reoccurring parameters (Page 1844).

Parameter: **Enabled**
For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).

Typical-specific response: Enable block processing
If processing is disabled the FB only checks to see if the disabled status has been canceled. Any commands that are still received are not output. The FB cannot communicate at the organizational level in this status because Cmd01B_R cannot send or receive Org. frames.

If the "Enabled" input should be made operable by a switch, this local disable means received commands are no longer output. Since the block is, however, not capable of sending Org. frames, it cannot report this local block back to the partner itself. This must be done with another typical, e.g. Bin04B_S.

Parameter: **MultipleOutput**
Declaration: INPUT
Data type: BOOL
Range of values: TRUE / FALSE
Default: FALSE
Explanation: Simultaneous output of multiple commands permitted

With this parameter, you can specify whether or not several (consecutively received) commands can be output simultaneously; in other words, you specify how the block reacts when a new command is received and the previously received command still needs to be output.

Requirement: The command output time has not yet elapsed and the user program has not yet reset this command.

- FALSE
  Multiple output is not permitted. The newly received command overwrites the output byte. Any command still pending is therefore reset to 0 unless the new command is identical to the old one.
- TRUE
  Multiple output is permitted. A newly received command is written to the current output byte. The command output time is restarted and applies to all pending commands.

Parameter: **CommandOutputTime**
Declaration: INPUT
Data type: INT

Range of values:	0 ... 500
Default:	500
Explanation:	Command output time for command outputs in milliseconds The specified time applies to all command outputs. If more than one output can be set at the same time (MultipleOutput = TRUE), the output time is restarted with each newly received command. This means retriggering for already pending commands. All the command outputs are reset at the same time onöy when the output time elapses. With the value 0 a set command output is not reset by the command typical. You need to do this via the user program.

Parameter:	**NewData**
Explanation:	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844). The output "NewData" is always set to TRUE for one OB1 cycle when the FB has received new data and output it to the output byte "CommandOuputByte".

Parameter:	**CommandOutputByte**	
Declaration:	IN_OUT	
Data type:	BYTE	
Range of values:	See address range	
Default:	0 (B#16#0)	
Address range:	(process image) output bytes	QB0 ... QBn
	Memory bytes	MB0 ... MBn
	Data bytes	DBm.DBB0 ... n
	Since the parameter is an IN_OUT parameter, direct I/O output of the command byte to PQB0 ... PQBn is not permitted. It is also difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Command output byte To allow the command outputs to be reset both by the command typical itself as well as by the user program (when output time = 0), the parameter was declared as an IN_OUT parameter.	

Counted value typicals Cnt01D_S / Cnt04D_S

Function

- Cnt01D_S: Send 1 counted value (32 bits).
- Cnt04D_S: Send 4 counted values (32 bits).
  Note that the parameter "DifferenceValue" for forming the difference value can only be activated at the same time for all four counted values.

Parameter

Parameter:	**PartnerNo** **PartnerObjectNo** **Enabled** **ImageMemory** **Conditional** **Unconditional** **TimeStamp** For a description, see section Reoccurring parameters (Page 1844).

Parameter:	**GeneralTriggerCommand**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Restore collective command The restore collective command as a central system memory bit belongs to the organizational SINAUT system commands. Set the parameter to TRUE if the counted value transfer is to be triggered by a restore collective command. • When the destination subscriber no. (PartnerNo) = 0 (transfer to all), the restore collective command is taken into account. When the restore collective command is detected, the currently accumulated counted value is transferred regardless of other triggers for transfer. The restore bit is inverted in this counted value. • If in the typical an explicit destination subscriber no. (PartnerNo > 0) is configured, the restore collective command is evaluated in the corresponding subscriber object in the central administration. You can use the parameters "GeneralTriggerCommand" and "TriggerInput" at the same time. In this case the transfer is triggered both by an edge change 0 → 1 at "TriggerInput" and when a restore collective command is received.

Parameter:	**TriggerInput**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Trigger input

With the edge change 0 → 1 of the "TriggerInput" input the triggered transfer can be triggered at a required time regardless of other criteria for a transfer.. The currently accumulated counted value is transmitted. The restore bit (see above) is inverted in this counted value.

Example:
Time-driven transmission with time stamp for supplying an archive in the control center.

You can use the parameters "GeneralTriggerCommand" and "TriggerInput" at the same time. In this case the transfer is triggered both by an edge change 0 → 1 at "TriggerInput" and when a restore collective command is received.

With the setting FALSE no restoring and no transfer via the "TriggerInput" input is triggered.

Parameter: **Counter_1 (Cnt01D_S)**
Counter_1 ... _4 (Cnt04D_S)

Declaration: INPUT

Data type: COUNTER

Range of values: 0 ... 32767

- Z0 as placeholder
  or
- Z1 ... Zn
  n depends on the CPU type.

Default: -

Explanation: Number of the SIMATIC counter

Here, you specify the SIMATIC counter in which the pulses were counted time-driven. This counting takes place in the background using FC PulseCounter that is called in a cyclic interrupt OB (for example in OB35). See also section FC PulseCounter (Page 1938) and Cyclic interrupt OB (Page 1839).

The COUNTER data type cannot be preassigned a value.

If you configure the Z0 placeholder, the corresponding counted value is not processed.

Parameter: **DifferenceValue**

Declaration: INPUT

Data type: INT

Range of values: 0 ... 31767

Default: 0

Explanation: Difference value

- When a value between 1 and 31767 is configured, the counted value is transferred as soon as the difference between the current and most recently transferred counted value reaches the value specified here.
- If the default value 0 is configured, a counted value is transferred only in the following situations:
  - On an edge change 0 → 1 at the "TriggerInput" input
  - On receipt of a restore command when "GeneralTriggerCommand" = TRUE.

Select the difference value dependent on the maximum pulse rate per second.

Do not select a value that is too low so that the counted value is not constantly transferred to the TIM. This would put load on the communications path to the CPU and the send queue of the CPU.

Note on Cnt04D_S
In the typical this processing parameter for forming the difference value only exists once. It applies to all 4 counted values together. It is not possible to set the parameter for the individual counted values. When using this parameter, each typical should therefore only acquire counted values that can be processed identically.

Counted value typicals Cnt01D_R / Cnt04D_R

Function

- Cnt01D_S: Receive 1 counted value (32 bits)
- Cnt04D_S: Receive 4 counted values (32 bits)

Parameter

Parameter: **PartnerNo**
PartnerObjectNo
Enabled
For a description, see section Reoccurring parameters (Page 1844).

Parameter: **BCD_Format**
Declaration: INPUT
Data type: BOOL
Range of values: TRUE / FALSE

Default:	TRUE
Explanation:	Counted value output in BCD format • If the parameter is activated the received counted value is output as a positive BCD value at the "CountedValueOutput_n" output. • If the parameter is deactivated the counted value is output as a positive 32-bit integer value. For the different value ranges of the two formats, see the parameter "CountedValueOutput_n". If the maximum counted value that can be represented is exceeded, the counted value starts again at 0 and counting continues in the positive numeric range. Note on Cnt04D_R The parameter exists only one in the typical and it applies to all 4 counted values together. It is not possible to make an individual setting per counted value. When using this parameter, per typical only counted values with an identical output format should be output.

Parameter:	CntValInvalid
Declaration:	OUTPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Address range:	Output	Q 0.0 ... Q n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Counted value invalid

When evaluating "CntValInvalid" remember that the bit might only be set for one OB1 cycle.

The CntValInvalid output indicates whether the last received counted value was invalid. . With "Cnt04D_R", this counts as a group display for all 4 counted values, see the note below.

The output shows the validity status of the most recently received counted value in inverted form.

The output serves the following purposes:

- Error display
- Signal for user-specific further processing
  You can, for example, react to the lack of up-to-dateness, by correcting the counted value output at "CountedValueOutput_n" with possibly lost counting pulses.

If you do not require the parameter, simply leave it open.

Note on Cnt04D_R
Although all 4 counted values have their own status bit in the data frame, for the status at the output "CntValInvalid" only the bit of the first counted value in the previously received data frame is evaluated.
This status, however, applies to all 4 counted values.
(All counted values in the data frame always have the same status.)

Parameter:	**RestoreStatus**
Declaration:	OUTPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Address range:	Output	Q 0.0 ... Q n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Status of the restore bit in the received counted value.

The "RestoreStatus" output indicates the current status of the restore bit from the last received counted value frame.

You can use the output for user-specific further processing.

Example:
You can only access the information at "CountedValueOutput_n" when a change has been detected at the "RestoreStatus" output; in other words, when the counted value has been received due to a restore, such as a local time-driven restore.

If you do not require the parameter, simply leave it open.

Note on Cnt04D_R
Although all 4 counted values have their own restore bit in the data frame, for the status at the "RestoreStatus" output only the restore bit of the first counted value in the previously received data frame is evaluated.
This status, however, applies to all 4 counted values.
(All counted values in the data frame always have the same status.)

Parameter: **NewData**

Explanation: For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).

The output "NewData" is always set to TRUE for one OB1 cycle when the FB has received new data and output it to the output / outputs "CountedValueOutput_1" to "CountedValueOutput_4"

Parameter: **CountedValueOutput_1 (Cnt01D_S)**
CountedValueOutput_1 ... _4 (Cnt04D_S)

Declaration: IN_OUT

Data type: DWord, UDInt

Range of values:	• Integer:	0 ... 2 147 483 647
	• BCD:	0 ... 9 999 999

Default: 0

Address range:	Output (DWORD)	QD0 … QDn
	Bit memory (DWORD)	MD0 ... MDn
	Data (DWORD)	DBm.DBB0 … n
	Since the parameter is an in-out parameter (declaration IN_OUT), direct I/O output of the counted value to PQD0 ...PQDn is not permitted.	
	It is also difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Counted value output	
	The counted value typical always adds the newly formed difference value (difference between the new and last received counted value) to the value currently output at the counted value output.	
	The counted value output is a double word in which the counted value is stored in BCD format or as a 32-bit integer value (depending on the "BCD_Format" parameter, see above).	
	The counted value id always output as a positive number. If the maximum counted value that can be represented is exceeded, the counted value starts again at 0 and counting continues in the positive numeric range.	
	Since the parameter is an in-out parameter (IN_OUT), the value can be reset to 0 or another value at the counted value output by the user program at any time.	

Data typical Dat12D_S

Function

Send maximum of 12 double words with any data content.

The content of each double word may be a value in double word (DWORD, DINT, REAL) format, it can also be a mixture of other data types which together form a double word, for example:

- 4 bytes
- 2 words
- 2 bytes + 1 word

Sending the data area can be triggered in two ways:

- By a change check
  The data is transferred as soon as a bit changes ("SendOnChange" = TRUE).
- By the user program
  The transfer can be triggered by an edge change 0 → 1 at the "TriggerInput" input
  For time-driven transfer FC Trigger can be used.

With "SendAll" you can also specify whether the transfer always includes all data or only the data double words that have changed.

Note

Remember double word boundaries

When changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 12 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameter

Parameter:	**PartnerNo** **PartnerObjectNo** **Enabled** **ImageMemory** **Conditional** **Unconditional** **TimeStamp** For a description, see section Reoccurring parameters (Page 1844).

Parameter:	**SendOnChange**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Send on change With the setting TRUE, the block runs a change check within the acquired data area "DataInput". The block checks whether at least one bit has changed. If a change is detected, a transfer of the data area is started automatically. You specify whether the entire area is transferred or only the changed part with the "SendAll" parameter. If the setting is FALSE, you need to trigger the transfer via the input parameter "TriggerInput".

Parameter:	**TriggerInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Trigger input

With the edge change 0 → 1 of the "TriggerInput" input, the transfer of the data frame can be triggered at a required time.

Example:
Time-driven analog value transfer with time stamp for supplying an analog value archive in the control center.

Make sure that you set the "ImageMemory" parameter to FALSE to prevent this data with time stamps from being overwritten when saving on the station TIM.

The FC Trigger block can be used for time-driven triggering of a transmission over "TriggerInput".

"TriggerInput" actually only triggers transmission indirectly. With a 0 → 1 edge change at "TriggerInput", the data frame is put together with its current values and transferred to the local TIM. The TIM is responsible for the actual transmission to the partner. With dedicated lines or wireless networks the transfer is immediate. With a dial-up connection, it is possible that the data frame is saved first on the TIM and sent at a later point in time. The reason can, for example, be that the data frame is marked as "Conditional spontaneous", see parameter "Conditional".

If you do not require the parameter, simply leave it open. You should, however, then set the "SendOnChange" parameter to TRUE so that the data is transmitted automatically at every change.

For the triggering you can also select a combination of "SendOnChange" plus "TriggerInput". This means that a transfer is triggered both when a change is detected and at every edge change from 0 to 1 at the "TriggerInput" input.

If you use neither "SendOnChange" nor "TriggerInput" to trigger data transfer, the data will only be transferred when there is a single request for this data object or within the framework of a general request.

Parameter:	**SendAll**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE

Default: TRUE

Explanation: Send all data with every transfer

With the parameter, you specify whether the block will always transfer all data of the area specified with "DataInput" or only changed data. The transfer can be triggered by the activated change check (SendOnChange = TRUE) or by "TriggerInput".

- SendAll = TRUE
  Always send all data
- SendAll = FALSE
  Send only changed data
  Exception:
  If "SendAll" is set to = FALSE, the transfer is triggered by "TriggerInput" and if no data has changed at this time, the complete area will be transferred. For this exception this corresponds to "SendAll" = TRUE.
  When only the changed data area is transferred ("SendAll" = FALSE), this area consists of the first and the last double word in which a change was detected and all words located in between, even if these have not changed.

If there is a single request for this data object or within the framework of a general request, all data words of the area specified by "DataInput" are always transferred.

Parameter: **DataInput**

Declaration: INPUT

Data type: ANY

Range of values: See address range

Default: P#P 0.0 VOID 0
(null pointer)

Address range: P#DBxx.DBX yy.0 DWORD zz

- xx: Data block number 1...32767
- yy: Byte number
- zz: Number of double words 1...12 starting at byte number yy

Example:
`P#DB20.DBX 100.0 DWORD 4`

Remember the periods and spaces when entering the pointer!

Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.

Explanation: Data input area

The ANY pointer addresses the data area in which the data to be acquired is located. This data area must be within a data block and its length can vary between 1 and 12 data double words.

For information on the content and formats, refer to the section "Function" above.

If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Data typical Dat12D_R

Function

Receive maximum of 12 double words with any data content.

The content of each double word may be a value in double word (DWORD, DINT, REAL) format, it can also be a mixture of other data types which together form a double word, for example:

- 4 bytes
- 2 words
- 2 bytes + 1 word

Dat12D_R stores the received data without further processing in the data area specified by "DataOutput". You need to evaluate and process the received data with the user program.

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameter

Parameter:	**PartnerNo** **PartnerObjectNo** **Enabled** For a description, see section Reoccurring parameters (Page 1844).

Parameter:	**DataOutput**
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)

Address range: P#DBxx.DBX yy.0 DWORD zz

- xx: Data block number 1...32767
- yy: Byte number
- zz: Number of double words 1...12 starting at byte number yy

Example:
`P#DB20.DBX 100.0 DWORD 4`

Remember the periods and spaces when entering the pointer!

Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.

Explanation: Data output area

The ANY pointer addresses the data area in which the received data is saved. This data area must be within a data block and its length can vary between 1 and 12 double words.

For information on the content and formats, refer to the section "Function" above.

Dat12D_R stores the received data without further processing in the data area specified by "DataOutput". You need to evaluate and process the received data with the user program.

When only changed data is sent by the partner object Dat12D_S, it is possible that only part of the data output area is newly written. this is the area in which the changes were detected at the acquisition end

If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Parameter: **NewData**

Explanation: Receive new data

For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).

Whenever the block has received new data values from the partner object and has output them to the output field "DataOutput", the "NewData" output is set to TRUE for one OB1 cycle.

Data typical Dat256D_S

Function

Send a maximum of 256 double words with any data content

The content of each double word can be a value in double word format (DINT, REAL etc.). A combination of other formats is permitted that together result in a double word again, for example

- 32 Bool
- 4 bytes

- 2 words
- Any combination such as 2 bytes plus 1 word etc.

Note

Remember double word boundaries

When changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 256 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

Sending the data area can be triggered in two ways:

- By a change check
  The data is transferred as soon as a bit changes ("SendOnChange" = TRUE).
- By the user program
  The transfer can be triggered by an edge change 0 → 1 at the "TriggerInput" input
  For time-driven transfer FC Trigger can be used.

With "SendAll" you can also specify whether the transfer always includes all data or only the data double words that have changed.

With S7-300 CPUs with X communication, the maximum length of a data frame is 76 bytes. 1024 bytes of net data are transferred using a serial transfer process consisting of a sequence of at least 22 data frames (segments). Each data frame apart from the last contains a segment of 48 bytes of net data of the input data area.

To ensure data consistency when the "SendAll" parameter is activated or during a general or single request, the data is transferred in consecutive segments. During the transfer process, the status is indicated by "SendAllBusy". On the recipient, the status is indicated at the "DataStatus" output.

Note

TriggerInput - SendAllBusy

If "TriggerInput" is triggered when "SendAllBusy" = TRUE, this leads to the "DataLoss" error message (status in the frame header) if the transfer is triggered again.

Only when "SendAllBusy" = FALSE is set is the edge change 0 → 1 triggered at "TriggerInput".

If the transfer is interrupted, "SendAllError" is indicated. An entry is also made in the diagnostics buffer with the event ID B14DTD7_Diagnostics.

If the transfer is incomplete, the data status at the recipient is also "invalid". This is indicated on the recipient in the DataStatus parameter. Apart from this an entry with the event ID B13BTD7_Diagnostics is written to the diagnostics buffer.

Note

Availability of the partner

If the status of the partner changes from "available" to "unavailable", the transfer of all data is stopped immediately. All object data is deleted from the TIM buffer. This can lead to loss of data.

As soon as the partner is available again, the automatic general request ensures that the data of the partner is up-to-date again for the next transfer.

Note

Dat256D_S and Dat256D_R require the UDT "TransmitBlock".

When using the typical, copy the UDT from the global library into the "PLC data types" directory of the CPU. The UDT is automatically referenced by the typical from the block directory of the CPU, but not from the global library.

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameter

Parameter: **PartnerNo**
PartnerObjectNo
Enabled
Unconditional
TimeStamp
For a description, see section Reoccurring parameters (Page 1844).

Parameter:	**SendOnChange**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE

Default:	FALSE
Explanation:	Send on change With the setting TRUE, the block runs a change check within the acquired data area "DataInput". The block checks whether at least one bit has changed. If a change is detected, a transfer of the data area is started automatically. You specify whether the entire area is transferred or only the changed part with the "SendAll" parameter. If the setting is FALSE, you need to trigger the transfer via the input parameter "TriggerInput".

Parameter:	**TriggerInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Trigger input

With the edge change 0 → 1 of the "TriggerInput" input, the transfer of the data frame can be triggered at a required time.

Example:
Time-driven analog value transfer with time stamp for supplying an analog value archive in the control center.

The FC Trigger block can be used for time-driven triggering of a transmission over "TriggerInput".

"TriggerInput" actually only triggers transmission indirectly. With a 0 → 1 edge change at "TriggerInput", the data frame is put together with its current values and transferred to the local TIM. The TIM is responsible for the actual transmission to the partner. With dedicated lines or wireless networks the transfer is immediate. With a dial-up connection, it is possible that the data frame is saved first on the TIM and sent at a later point in time. The reason can, for example, be that the data frame is marked as "Conditional spontaneous", see parameter "Conditional".

Select suitable trigger points so that the data on the TIM is not overwritten by buffer overflow (intervals too long).

If you do not require the parameter, simply leave it open. You should, however, then set the "SendOnChange" parameter to TRUE so that the data is transmitted automatically at every change.

For the triggering you can also select a combination of "SendOnChange" plus "TriggerInput". This means that a transfer is triggered both when a change is detected and at every edge change from 0 to 1 at the "TriggerInput" input.

If you use neither "SendOnChange" nor "TriggerInput" to trigger data transfer, the data will only be transferred when there is a single request for this data object or within the framework of a general request.

Do not transfer any analog values for which the "SendOnChange" parameter = TRUE is set without first preprocessing the process data. You will find more detailed information on this with the analog value typical Ana04W_S, parameter "ThresholdValue".

Parameter:	**SendAll**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE

Default: TRUE

Explanation: Send all data with every transfer

With the parameter, you specify whether the block will always transfer all data of the area specified with "DataInput" or only changed data. The transfer can be triggered by the activated change check (SendOnChange = TRUE) or by "TriggerInput".

- SendAll = TRUE
  Always send all data
- SendAll = FALSE
  Send only changed data
  Exception:
  If "SendAll" is set to = FALSE, the transfer is triggered by "TriggerInput" and if no data has changed at this time, the complete area will be transferred. For this exception this corresponds to "SendAll" = TRUE.

If there is a single request for this data object or within the framework of a general request, all data words of the area specified by "DataInput" are always transferred.

Parameter: **DataInput**

Declaration: INPUT

Data type: ANY

Range of values: See address range

Default: P#P 0.0 VOID 0
(null pointer)

Address range: P#DBxx.DBX yy.0 DWORD zz

- xx: Data block number 1...32767
- yy: Byte number
- zz: Number of double words 1...256 starting at byte number yy

Example:
`P#DB20.DBX 100.0 DWORD 200`

Remember the periods and spaces when entering the pointer!

Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.

Explanation: Data input area

The ANY pointer addresses the data area in which the data to be acquired is located. This data area must be within a data block and its length can vary between 1 and 256 data double words. For information on the possible double word formats, refer to the section "Function" above.

If the parameter assignment is incorrect (null pointer, length > 256, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Data consistency;
If a data segment to be transferred consists of a maximum of 48 bytes, data consistency during the transfer is assured.

Parameter:	**SendAllBusy**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Block being processed with "SendAll" = TRUE This output indicates that the block is currently transferring the data specified by "DataInput". The procedure is activated either by a remote single or general request or by a local internal or external trigger. If "SendAll" is set to TRUE, the transfer of all data is triggered either by internal change control (SendOnChange = TRUE) or by the external "TriggerInput" (edge change 0 → 1). The edge change 0 → 1 has no effect whatsoever with an external "TriggerInput" as long as "SendAllBusy" indicates TRUE. The edge change 0 → 1 of "TriggerInput" only takes effect when "SendAllBusy" = FALSE.

Parameter:	**SendAllError**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Error when processing "SendAll" "SendAllError" is set to TRUE when the connection is interrupted during processing of "SendAll". Apart from this an entry is written to the diagnostics buffer. "SendAllError" remains set to TRUE until it is reset by the user program or by the next CPU restart.

Data typical Dat256D_R

Function

Receive maximum of 256 double words with any data content.

The content of each double word may be a value in double word format (e.g. DINT, REAL etc.); it can also be a mixture of other formats which together form a double word, for example,

- 32 Bool
- 4 bytes
- 2 words
- Any combination such as 2 bytes plus 1 word etc.

Note

Remember double word boundaries

When changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 256 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

Dat256D_R stores the received data without further processing in the data area defined by "DataOutput". Evaluate the received data with the user program.

With S7-300 CPUs with X communication, the maximum length of a data frame is 76 bytes (net 48 bytes). 1024 bytes of net data are transferred using a serial transfer process consisting of a sequence of at least 22 data frames (segments). Each data frame apart from the last contains a segment of 48 bytes of net data of the output data area.

Each time a detected data segment is received, this is indicated by a corresponding status (bit 1 to 22) of the "NewData" output parameter.

If a change was detected in the data segment, the status bit 0 is also set to TRUE in "NewData" for one CPU cycle. This makes it possible to recognize which segment of the output data area has changed.

Note

When receiving a sequence of several data segments (data frames), the status bits 1 to 22 in the "NewData" parameter are set to TRUE one after the other and remain set to TRUE until the last segment has been received.

If a data segment (data frame) is not part of a received sequence (SendAll = FALSE), the status remains set to TRUE for only one CPU cycle.

To ensure data consistency when "SendAll"= TRUE or during a general or single request, the data area is updated in consecutive individual segments.

During receipt, the status is indicated by the "DataStatus" output byte ("SequenceState" status). If the receive sequence was completed successfully, the data output area is up-to-date and the output data is consistent. This is indicated by "DataStatus" ("DataValid" = TRUE status).

Note

The consistency of the data segments or the limit segments cannot be guaranteed if the "SendAll" parameter was set to FALSE on the sender.

Receipt of a sequence can be disrupted by the following causes:

- The receive sequence was interrupted when communication to the partner fails during an active sequence (event ID B13BTD7_Diagnostics).
- The monitoring time was exceeded. Not all segments could be received within the time set for the "MonitoringTime" parameter (event ID B13CTD7_Diagnostics).
- Other receiving errors occur (event ID B13DTD7_Diagnostics), for example:
  - A new receive sequence is registered during an active, error-free sequence.
  - A spontaneous segment (data frame) is received during an active sequence.

Note

Dat256D_S and Dat256D_R require the UDT "TransmitBlock".

When using the typical, copy the UDT from the global library into the "PLC data types" directory of the CPU. The UDT is automatically referenced by the typical from the block directory of the CPU, but not from the global library.

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameter

Parameter:	**PartnerNo** **PartnerObjectNo** **Enabled** For a description, see section Reoccurring parameters (Page 1844).

Parameter:	**SingleRequest**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	TRUE

Address range:	Input	I 0.0 … I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: A single request is sent to the partner object.

If the partner is available, you can send a single request to the partner object. If a reply is returned, the information is forwarded to the data area specified in "DataOutput".

In terms of transfer sequences, there are priorities:

- Lowest priority: TriggerInput
  An active transfer triggered, for example, by "TriggerInput" on the sender can be interrupted by a single request or a general request.
- Medium priority: Single request
  An active transfer triggered, for example, by "TriggerInput" on the sender can be interrupted by a single request or a general request.
  The interrupted or restarted request leads to a restart of the active sequences without an error message.
- Highest priority: General request
  A general request can interrupt itself or a single request.
  The interrupted or restarted request leads to a restart of the active sequences without an error message.

The request interrupted or restarted by a single or general request leads to a restart of the active sequences without an error message. If a sequence was completed successfully, the "DataValid" status of the "DataStatus" output byte remains set to TRUE.

The time taken for the response to the single request is evaluated by the "MonitoringTime" parameter.

Note:
Consistency of the data output area across segments is only assured if the receive sequence was completed successfully.

Parameter:	**MonitoringTime**
Declaration:	INPUT
Data type:	INT
Range of values:	0 (no limit) / 1 … 32000 (seconds)
Default:	0

Explanation: Maximum time for a complete response to a single request

Each time a single request starts (see SingleRequest parameter), the time specified here is started in "SingleRequest".

If a value higher than 0 is entered and the time for the response sequence is exceeded, an error is indicated via the "DataStatus" output byte (status bits "SequenceState"). Apart from this an entry is written to the diagnostics buffer (event ID B13CTD7_Diagnostics).

Each time a single request starts, "MonitoringTime" is reactivated.

Parameter:	**DataOutput**
Declaration:	INTPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...256 starting at byte number yy Example: `P#DB20.DBX 100.0 DWORD 200` Remember the periods and spaces when entering the pointer! Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.
Explanation:	Data output area The ANY pointer addresses the data area in which the received data is saved. This data area must be within a data block and its length can vary between 1 and 256 data double words. For information on the possible double word formats, refer to the section "Function" above. Dat256D_R stores the received data without further processing in the data area specified by "DataOutput". You need to evaluate and process the received data with the user program. When only changed data is sent by the partner object Dat256D_S, it is possible that only part of the data output area is newly written. this is the area in which the changes were detected at the acquisition end If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Parameter:	**NewData**
Declaration:	OUTPUT
Data type:	DWORD
Range of values:	See address range
Default:	0 (DW#16#0)

Address range:	Output (DWORD)	QD0 … QDn PQD0 … PQDn
	Bit memory (DWORD)	MD0 … MDn LD0 … LDn
	Data (DWORD)	DBm.DBB0 … n

Explanation: Receive new data

Whenever the block receives new data segments, the display of status bits 1 to 22 according to the received data segments 1 to 22 appears in "NewData".

If at least one data segment of the received data contains changes, bit 0 of "NewData" is set to TRUE for one OB1 cycle.

When receiving a sequence of several data segments (data frames), the status bits 1 to 22 in the "NewData" parameter are set to TRUE one after the other and remain set to TRUE until the last segment has been received.

If a data segment (data frame) is not part of a received sequence ("SendAll" = FALSE), the status remains set to TRUE for only one CPU cycle.

The output is intended for user-specific further processing, for example to react in a specific way to receipt of new data.

If you do not require the parameter, simply leave it open.

Parameter:	**DataStatus**	
Declaration:	OUTPUT	
Data type:	BYTE	
Range of values:	See address range	
Default:	0 (B#16#0)	
Address range:	Output	QB0 … QBn PQB0 … PQBn
	Memory bit	MB0 … MBn LB0 … LBn
	Data	DBm.DBB0 … n

Explanation: Currentness status of a received data segment

During receipt of a sequence, the current status is indicated by the "DataStatus" output byte:

- If the receipt of the sequence was completed successfully, the "DataOutput" data output area is up-to-date. The status bit "DataValid" is set to TRUE. If "SendAll" is set to TRUE on the sender, the data is consistent.
- If the receipt of a sequence is disrupted, the "DataOutput" data output area is not up-to-date and "DataStatus" indicates an error.
  - The status bit "DataValid" is set to FALSE and an entry is written to the diagnostics buffer.
  - The "SequenceState" status shows the error (see table).

Table 1-145 Bit assignment of "DataStatus"

Bit	Name	Value	Meaning
0	**DataValid**	FALSE	Data invalid
		TRUE	Data valid
1 ... 5	Reserved	- (FALSE)	Not used
7, 6	**SequenceState**	0	No data being received or receipt completed without error.
		1	First segment of a sequence received
		2	Second or higher segment of a sequence received
		3	Errors: • Transmission sequence aborted • Monitoring time exceeded • Other error receiving

Parameter typical Par12D_S / Par12D_FS

Function

Send 1 to 12 parameter values (each 1 double word) and receive the current, locally valid parameter values from the partner.

The 1-out-of-n check is performed by FC Safe.

Note

FC Safe required

With Par12D_S, data can only be transmitted when FC Safe is linked in the end of cyclic program, see Section FC Safe (Page 1915).

The content of each double word may be a value in double word (DWORD, DINT, REAL) format, it can also be a mixture of other data types which together form a double word, for example:

- 4 bytes
- 2 words
- 2 bytes + 1 word

See also the note "Note word boundaries" below.

The data area to be transferred is specified for the "ParameterInput" parameter in the form of an Any pointer. This data area must be within a data block and its length can vary between 1 and 12 double words. The data area sent to the partner or the parameter values entered locally at the partner are returned from there and output here at the "ReturnedParameter" parameter. This output area (Any pointer) must also be within a data block and its length must match that specified for "ParameterInput".

Separate data areas are normally specified for "ParameterInput" and "ReturnedParameter". This makes it easy to recognize what was most recently entered and what is locally valid. However, it is also possible to specify the same data area for both parameters. The two areas

then overlap 100% and therefore always match. In this case, you can no longer distinguish the difference between what has been entered most recently and what is locally valid. When mirrored back values are not needed, there is no need to configure "ReturnedParameter".

Even when separate areas are specified for "ParameterInput" and "ReturnedParameter", it is still possible to ensure that the "ParameterInput" input area is always synchronized with the mirrored back values of "ReturnedParameter". This can be done manually from case to case with the "ApplyRemoteParamMan" input or automatically by setting the "ApplyRemoteParamAuto" parameter to TRUE.

A parameter can also be set locally at the partner object that receives the parameter. The partner object must then be set to ’local’ at the "Local" input parameter (see block Par12D_R). The current status of the "Local" input parameter is reported by the partner object and indicated here at the "LocalOperation" output. As long as the partner object is set to 'local', no parameters are accepted there from other nodes.

The sending of the data area defined by "ParameterInput" can be triggered via the following parameters:

- EnterInput
  You should use this input parameter when the data area defined at "ParameterInput" is entered over hardware (digital and analog input modules). "EnterInput" must then be connected to a button on a console or panel via a digital input. The transmission of the entered values is then triggered by pressing this button.
  The entire data area specified by "ParameterInput" is transferred.
- ContinuousEnterFunct
  Set the parameter to TRUE if you enter the parameters using software, for example via an operator panel (OP). There is a constant check for changes. When a change is detected in the data area defined with "ParameterInput", the data double words that have changed since the last transfer are transferred.
  Only the changed data is transferred (see note "Changed data areas" below).
- Release
  Use this input parameter if you enter the parameters using software, for example via an OP. The "Release" input should then be set using a function key on the OP. Changes are checked when a 1 signal is detected at the "Release" input. The data double words from the data area specified with "ParameterInput" that have changed since the last transfer are transferred.
  Only the changed data is transferred (see note "Changed data areas" below).
- RetransmitAll
  Use this input parameter if you enter the parameters using software, for example via an OP. The "RetransmitAll" input should then be set using a function key on the OP. When a 1 signal is detected at the "RetransmitAll" input, the data area configured in "ParameterInput" is transferred without checking for changes.
  The entire data area specified by "ParameterInput" is transferred.

Note

Changed data areas

When only the changed data area is transferred, this area consists of the first and the last double word in which a change was detected and all words located in between, even if these have not changed.

Example:
The area to be read in is 10 double words long. In this case, changes were detected in the second, fifth and eighth double words. The transferred area is therefore from the 2nd to the 8th double word.

Remember word boundaries

When only changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 12 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameter

Parameter:	**PartnerNo** **PartnerObjectNo** **Enabled** For a description, see section Reoccurring parameters (Page 1844).	

Parameter:	**EnterInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Enter input

The transfer of the parameter values at the "ParameterInput" input can be triggered vis this input by a signal edge change.

A signal change at "EnterInput" is only taken into account when "ContinuousEnterFunct" = FALSE. If this condition is fulfilled, an edge change 0 → 1 causes the parameter values entered in "ParameterInput" to be adopted and transferred. There is no change check. The entire data area specified in "ParameterInput" is always transferred.

This method of triggering transfer is suitable for input vis hardware, for example via a console or control panel. For further information and related parameters, refer to the section "Function" above.

If you do not require the parameter, simply leave it open.

Data checks:

- The parameters that are read in are then transferred f no error is detected during the 1-out-of-n check, and if the central enable memory bit is set. This is automatically set by FC Safe following a selected time delay set there (see FC Safe, "InputDelayTime" parameter). The input area is then only read in again by the FB when a 0 signal was detected at EnterInput for at least one OB1 cycle.
- When a 1-out-of-n error is detected at the hardware input, the entered parameters are no longer processed. New parameters are read in again only when previously for the length of an OB1 cycle, no hardware input via a command, setpoint or parameter block was detected.
  The FB enters the detected 1-out-of-n error in the diagnostic buffer (event ID B172). As long as the error remains, the error status is indicated via the "InputError" output of FC Safe (see FC Safe, "InputError" parameter).

Parameter:	ContinuousEnterFunct
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Explanation: Continuous change check

With this parameter, you can decide whether the parameter values at the "ParameterInput" input should be continuously read in and checked for changes. The change check is made by comparison with the last values that were transferred. Only changed values are sent. If more than one change is detected, the block sends the data area in which all changed parameter values are located.

The changed data area that is transferred consists of the first and the last double word in which a change was detected and all words located in between, even if these have not changed.

A new transfer of the parameter values can be triggered via the "RetransmitAll" input (see below) even when the parameter entries have not changed.

This method of transmission triggering is suitable when the parameter values are entered in the ParameterInput area by software, but can also be used for entering the parameters from an operator panel (OP).

For further information refer to the section "Function" above.

If you do not require the parameter, simply leave it open.

Data checks:

The parameters read in are only transmitted if no error is detected during the 1-out-of-n check.

- While for hardware input (see EnterInput) an empty cycle must be detected before new parameter values can be transferred from the block.
- For software input new parameter values can be transferred in every OB1 cycle. This assumes that there is no other software entry by another block pending in this cycle. Otherwise a 1-out-of-n error is detected.
  When a 1-out-of-n error is detected during the software input, the entered parameters are no longer processed. The FB enters the detected 1-out-of-n error in the diagnostic buffer (event ID B172).
  New parameters are read in again only when previously for the length of an OB1 cycle in the CPU, no software input via a command, setpoint or parameter block was detected.

Parameter: **ApplyRemoteParamAuto**

Declaration: INPUT

Data type: BOOL

Range of values: TRUE / FALSE

Default: FALSE

Explanation: Automatic synchronization of the input area with the returned area.

With the parameter all the parameter values from the "ReturnedParameter" mirror back area are then copied to the "ParameterInput" area.

In addition to this the mirrored back parameter values are written the send mailbox of the communications DB.

Automatic synchronization is then always performed when new data is received from the partner object (Par12D_R).

If you do not require the parameter, simply leave it open.

Parameter:	**ParameterInput**
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...12 starting at byte number yy Example: `P#DB20.DBX 100.0 DWORD 4` Remember the periods and spaces when entering the pointer! Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.
Explanation:	Parameter input area. The ANY pointer addresses the data area in which the parameter values to be acquired are located. This data area must be within a data block and its length can vary between 1 and 12 data double words. For information on the content and formats, refer to the section "Function" above. If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected. How the parameters at ParameterInput are processed depends on whether they are hardware or software entries and how the transfer of this data area is triggered. You will find more information on this in the section "Function" above.

Parameter:	**ReturnedParameter**
Declaration:	OUTPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)

Address range: P#DBxx.DBX yy.0 DWORD zz

- xx: Data block number 1...32767
- yy: Byte number
- zz: Number of double words 1...12 starting at byte number yy

Example:
`P#DB20.DBX 100.0 DWORD 4`

Remember the periods and spaces when entering the pointer!

Note that the preset value (null pointer) is not permitted. A pointer with a real address must be specified.

Explanation: Parameter output area

The partner object receiving the parameter values reports back the valid parameter values there. These values are displayed at the "ReturnedParameter" output. If the partner object is set to 'local' and a new input is made there, the parameters changed locally are indicated here by "ReturnedParameter".

The ANY pointer defines the data area in which the received parameter values are output. This data area must be within a data block and its length can vary between 1 and 12 data double words. The length must be identical with the length set for ParameterInput.

After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the current, local, valid parameters are indicated at "ReturnedParameter".

If you do not require the parameter, simply leave it open.

If the parameter setting is incorrect (data area not a data block, length greater than 12 or length different from the length set for ParameterInput), an error message to this effect is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The FB is then no longer processed, however, until the error has been corrected.

Parameter: **LocalOperation**

Declaration: OUTPUT

Data type: BOOL

Range of values: TRUE / FALSE

Default: FALSE

Address range:	Output	Q 0.0 ... Q n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Return message from partner object: Object is set to local operation.

At the partner object that receives the parameters a local setpoint entry can also be made.. The partner object Par12D_R must then be set to 'local' at the "Local" input parameter. The current status of the "Local" input parameter is reported by the partner object and indicated here at the "LocalOperation" output.

After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the local currently valid status is displayed at "LocalOperation".

Parameter: **NewData**

Explanation: For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).

Whenever the block has received new data and has output it to the output "ReturnedSetpoint" or "LocalOperation", the "NewData" output is set to TRUE for one OB1 cycle.

Parameter: **Release**

Declaration: IN_OUT

Data type: BOOL

Range of values: TRUE / FALSE

Default: FALSE

Address range:	Memory bit	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Trigger input for sending the currently pending parameter values	
	You can use this input parameter when the parameter is entered by software, for example at an operator panel (OP). "Release" should then be set using a function key on the OP. You can then enter several parameters initially on the OP. The parameters are transferred only when the Release function key is activated.	
	A change check is performed only with signal 1 at the Release input. The data double words from the data area configured with "ParameterInput" that have changed since the last transfer are transferred.	
	If you always want to transfer the entire data area specified with "ParameterInput" and not only the changed parameter values, you should use the "RetransmitAll" input parameter instead of "Release".	
	The "Release" input is reset automatically. You should therefore only specify memory or data inputs as the input. The automatic reset would not work with a digital input.	
	Data checks: The same safety checks are carried out as with ContinuousEnterFunct, see above.	
	If you do not require the parameter, simply leave it open.	

Parameter:	**RetransmitAll**
Declaration:	IN_OUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	TRUE

Address range:	Memory bit	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Trigger input for retransferring the entire data area specified by "ParameterInput".	
	You can use this input parameter when the parameter is entered by software, for example at an operator panel (OP). "RetransmitAll" should then be set using a function key on the OP. When a 1 signal is detected at the "RetransmitAll" input, the entire data area specified by "ParameterInput" is transferred. A change check is not performed	
	The "RetransmitAll" input is reset automatically. You should therefore only specify memory or data inputs as the input. The automatic reset would not work with a digital input. Since there is no change check, this would lead to continuous transfer of all parameter values as long as the input has a 1 signal.	
	"RetransmitAll" can also be used as an option in addition to "Release" or "ContinuousEnterFunct" when new parameter values were entered but could not be transferred to the partner (disrupted connection or the partner object is set to ’local’). In this case you can then trigger repeated transfer of the entire data area specified by "ParameterInput" using "RetransmitAll". All changes that were previously entered but are not yet available at the partner are consistently included.	
	"RetransmitAll" can also be used as an independent transfer trigger when you always want to send all entries and not just those that have changed. In this case you can use "RetransmitAll" instead of "Release" that only sends the changed parameter values.	
	Data checks: The same safety checks are carried out as with "ContinuousEnterFunct", see above.	
	If you do not require the parameter, simply leave it open.	

Parameter:	**ApplyRemoteParamMan**
Declaration:	IN_OUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Address range:	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.
Explanation:	Trigger input for synchronization of the input area with the returned area. The input triggers a one-time synchronization of the ParameterInput input area with the "ReturnedParameter" mirror back area. All the parameter values from the "ReturnedParameter" mirror back area are then copied to the "Parameter-Input" input area. The send mailbox of the communications DB is also synchronized with the returned parameter values. "ApplyRemoteParamMan" is reset automatically. You should therefore only specify memory or data inputs as the input. The automatic reset would not work with a digital input. The result would be a constant synchronization as long as the parameter has a 1 signal. If you do not require the parameter, simply leave it open.

Parameter typical Par12D_S

Function

Receive 1 to 12 parameter values (each 1 double word) or enter locally and send back the current, locally valid parameter values to the partner.

The content of each double word may be a value in double word (DWORD, DINT, REAL) format, it can also be a mixture of other data types which together form a double word, for example:

- 4 bytes
- 2 words
- 2 bytes + 1 word

See also the note "Note word boundaries" below.

The data area to be transferred is specified for the "ParameterOutput" parameter in the form of an Any pointer. This data area must be within a data block and its length can vary between 1 and 12 double words.

You can also use the block to enter the parameter values locally. The input area for this is specified as an Any pointer with the "LocalParameterInput" parameter. It must be located within a data block and its length must be identical to the length configured in the "ParameterOutput" parameter.

The block only processes the changed data area. In response to a general or single request, on the other hand, the entire parameter set is transferred or mirrored back..

Bumpless switchover between the "Local" and "Remote" operating modes is guaranteed.

Note

Changed data areas

The changed data area consists of the first and the last double word in which a change was detected and all words located in between, even if these have not changed.

Example:
The area to be read in is 10 double words long. In this case, changes were detected in the second, fifth and eighth double words. The transferred area is therefore from the 2nd to the 8th double word.

Remember word boundaries

When only changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 12 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameter

Parameter:	**PartnerNo** **PartnerObjectNo** **Enabled** **ImageMemory** **Conditional** **Unconditional** For a description, see section Reoccurring parameters (Page 1844).

Parameter:	**Local**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Local parameter input released

Release of the local parameter input of the data area specified by "LocalParameterInput".

A setpoint sent by the partner (master station) is not accepted by the object as long as "Local" = TRUE.

The current status of the "Local" input is transferred to the partner.

Bumpless switchover:

- When there is a switchover from "Local" = 0 to 1, the last values output at the "ParameterOutput" output are retained until new parameter values are entered via the local input area "LocalParameterInput".
- When there is a switchback from Local = 1 to 0, the last values at the "ParameterOutput" output are retained until the block receives new parameter values from the partner.

Read the note on the "ContinuousEnterFunct" parameter.

Special situation:

You can also enter the parameter values during local input directly in the output area specified by "ParameterOutput". Either you do not specify an input area for "LocalParameterInput" or you specify the same data area both for "LocalParameterInput" and "ParameterOutput".

This type of parameter input cannot be prevented by the "Local" input. Regardless of the status of the "Local" parameter, the values entered in the output area are sent immediately by the block to the partner.

Local parameter entries can therefore be made regardless of the status of the "Local" input. "Local" only influences the acceptance of parameters sent by the partner:

- Local = 0
  The parameters sent by the partner are accepted and output to the "ParameterOutput" data area.
- Local = 1
  The parameters sent by the partner are rejected.

In this special situation, the "Release" and "ContinuousEnterFunct" have no function.

A status change of the "Local" parameter is always transferred by the module according to the send buffer principle, even when the parameter "ImageMemory" = TRUE. This ensures that the optional synchronization of the input and output area on the partner is always performed correctly (see Par12D_S, parameters "ApplyRemoteParamMan" and "ApplyRemoteParamAuto").

Parameter: **ContinuousEnterFunct**

Declaration: INPUT

Data type: BOOL

Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Continuous local parameter acquisition. With this parameter, you can decide whether the values in the "LocalParameterInput" input area should be continuously read in and checked for changes. The change is checked by comparing the current values at the "ParameterOutput" output. Changes in the input area are copied immediately to the output area and transferred to the partner object. Only changed values are sent. If there is more than one change, the block sends the data area in which all changed parameter values are located. The "ContinuousEnterFunct" = TRUE setting only takes effect when the following conditions are met: • An input area is defined by "LocalParameterInput" and this is not identical to the output area set for "ParameterOutput". and • there is a 1 signal at the "Local" input (= TRUE). With the setting "ContinuousEnterFunct" = TRUE, when the signal 1 is detected at the "Local" input, the values pending at "LocalparameterInput" are adopted immediately and output at the "ParameterOutput" output. The condition is that the local input values differ from the currently output values at this time. This method of acquisition of local values is suitable when the values are entered in the "LocalParameterInput" area by software. If you do not require the parameter, simply leave it open.

Parameter:	**LocalParameterInput**
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)

Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...12 starting at byte number yy Example: `P#DB20.DBX 100.0 DWORD 4` Remember the periods and spaces when entering the pointer! Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.
Explanation:	Local parameter input area The ANY pointer addresses the data area in which the parameter values to be acquired are located. This data area must be within a data block and its length can vary between 1 and 12 data double words. The length must be identical to the length specified for" ParameterOutput". For information on the content and formats, refer to the section "Function" above. If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected. If you do not require the parameter, simply leave it open.

Parameter:	**ParameterOutput**
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)

Address range: P#DBxx.DBX yy.0 DWORD zz

- xx: Data block number 1...32767
- yy: Byte number
- zz: Number of double words 1...12 starting at byte number yy

Example:
`P#DB20.DBX 100.0 DWORD 4`

Remember the periods and spaces when entering the pointer!

Note that the preset value (null pointer) is not permitted. A pointer with a real address must be specified.

Explanation: Parameter output area

The ANY pointer addresses the data area in which the locally entered parameter values or those received from the partner are output. This data area must be within a data block and its length can vary between 1 and 12 double words.

For information on the content and formats, refer to the section "Function" above.

Par12D saves the received data without further processing in the data area specified by "ParameterOutput". You need to evaluate and process the received data with the user program.

When only changed data is sent by the partner object Par12D_S, it is possible that only part of the data output area is newly written. this is the area in which the changes were detected at the acquisition end

If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Parameter: **NewData**

Explanation: For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).

Whenever the block has received new parameter values from the partner object and has output them to the output field "ParameterOutput", the "NewData" output is set to TRUE for one OB1 cycle. This also applies when there is new local input in the status "Local" = 1.

Parameter: **Release**

Declaration: IN_OUT

Data type: BOOL

Range of values: TRUE / FALSE

Default: FALSE

Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7

This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.

Explanation: Input for the acceptance of local parameter input.

The acceptance of the parameter values at the "LocalParameterInput" parameter input can be triggered via this input by a signal edge change.

A change from 0 to 1 at the "Release" input is taken into account only when the following conditions are met:

- An input area is specified by the "LocalParameterInput" parameter and this is not identical to the output area specified by "ParameterOutput" and
- The "Local" input is set to TRUE.

You can use "Release" for parameter input via software, e.g. via an operator panel (OP). The "Release" input should then be set using a function key on the OP. You can then enter several parameters initially on the OP. The parameter values are read in and checked for changes only when the Release function key is activated.

The change is checked by comparing the current parameter values at the "ParameterOutput" output. Changes in the input area are then copied immediately to the output area and transferred to the partner.

Only changed values are sent. If there is more than one change, the block sends the data area in which all changed parameter values are located.

The "Release" input is reset automatically. Instead of a memory bit or data bit, a digital input can also be specified as the input. The automatic reset would not work with a digital input. This does not, however, have negative effects. Acquisition via "Release" is edge triggered, in other words only once.

If you do not require the parameter, simply leave it open.

FC Safe

Function

The FC Safe ensures reliable input of commands and setpoints by using the 1-out-of-n check.

The FC Safe checks th einput for the following data point typicals:

- Cmd01B_S
- Set01W_S
- Par12D_S

If an input is pending, the FC checks whether there is only one entry pending in the current OB1 cycle and then enables the block reading in. As soon as two (or more) entries are pending in an OB1 cycle, both entries are ignored.

In every CPU in which commands and/or setpoints are acquired, the FC Safe needs to be called in OB1 to complete of all command and setpoint FBs.

The FC has separate monitoring for entries via different system memory areas. They are divided into hardware entries and software entries:

- Hardware entries
  - Input modules (I)
  - Peripheral inputs (PI)
- Software entries
  - Bit memory (M)
  - Data blocks (DB)
  - Local data (L)
  - Operator panels

The two types of entry "hardware entries" and "Software entries" are checked separately by the block. The FC enables the hardware and software entries separately. For each type of entry, only one command or setpoint entry may be detected.

If a single hardware entry and a single software entry are pending at the same time both entries are enabled.

For hardware entries, the following additional condition applies: The entry must be pending constantly for the duration of "InputDelayTime". Only when an entered command or setpoint is pending unchanged for this time and during this time no other command or setpoint entry is detected is the block processing enabled.

The actual putting together of the command or setpoint frame is handled by the block that read in the command or setpoint.

For the hardware entry the FC Safe provides the following two display bits:

- InputOK
  Display of the enabling of hardware commands and setpoints
- InputError
  Display of entry errors with hardware commands and setpoints

FC-Safe shows a command output error detected in a station via the following output:

- GlobalCmdOutputError
  Group message: Command output error in a station

Parameter

Name:	**InputDelayTime**
Declaration:	INPUT
Data type:	INT
Range of values:	• 0 Value for unrequired parameter • 1 ... 32000 Range of values for delay time
Explanation:	Delay time in ms for commands and setpoints that are input via hardware. A delay time of at least 1000 ms is recommended.

Name:	**MaxInputTime**
Declaration:	INPUT
Data type:	INT
Range of values:	• 0 Value for unrequired parameter • 1 ... 32000 Range of values for monitoring time
Explanation:	Monitoring time in ms for commands and setpoints that are input via hardware. A monitoring time of at least 30 s is recommended. If you do not require the parameter, specify 0 (zero).

Name:	**ResetError**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Input for resetting the output GlobalCmdOutputError. If you do not require the parameter, specify a memory or data bit that always has signal 0.	

Name:	**InputOK**	
Declaration:	• S7-300	• OUTPUT
	• S7-1500	• IN_OUT
Data type:	BOOL	
Range of values:	Output	Q 0.0 ... Q n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Display of the enabling of hardware commands and setpoints Has a 1 signal as soon as the current entry was enabled, i.e. when the hardware command or setpoint was entered correctly. The display bit goes off when the entry is reset, in other words when the command key is released or the setpoint entry key "EnterInput" input is released. If you do not require the parameter, specify a memory or data bit in the memory area of the local data.	

Name:	**InputError**	
Declaration:	• S7-300	• OUTPUT
	• S7-1500	• IN_OUT

Data type:	BOOL	
Range of values:	Output	Q 0.0 ... Q n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Display of entry errors with hardware commands and setpoints The output has a 1 signal when one of the following hardware entry errors is detected within the monitoring time "MaxInputTime": • Two or more command and/or setpoint entries were detected at the same time. • At one of the inputs a 1 signal was detected over a longer period of time (for example when the input is defective). If you do not require the parameter, specify a memory or data bit in the memory area of the local data.	

Name:	**GlobalCmdOutputError**	
Declaration:	• S7-300 • S7-1500	• OUTPUT • IN_OUT
Data type:	BOOL	
Range of values:	Output	Q 0.0 ... Q n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Group message: In a station a command output error was detected. A command output error can occur at the receive end only in the following cases: • The content of the two command bytes in the received frame is not identical. • More than one bit is set in the command byte (1-out-of-8 error). If such an error is detected, this is sent by the station with an organizational frame to the subscriber that sent the commands. FC Safe on the sending subscriber then indicates the disruption at the output GlobalCmdOutputError. When an error is detected, the output remains set to the 1 signal until you request the group signal to be reset via the "ResetError" input. If you do not require the parameter, specify a memory or data bit in the memory area of the local data.	

Setpoint typical Set01W_S / Set01W_FS

Function

Send 1 setpoint as a 16 bit value and receive local setpoint from partner

The 1-out-of-n check is performed by FC Safe.

Note

FC Safe required

With Set01W_S, data can only be transferred when the block FC Safe is linked in the end of the cyclic program, see section FC Safe (Page 1915).

Parameter

Parameter:	**PartnerNo** **PartnerObjectNo** **Enabled** For a description, see section Reoccurring parameters (Page 1844).	

Parameter:	**EnterInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Enter input for hardware setpoint The adoption of a setpoint at the "SetpointInput" can be triggered by a signal edge change. A signal change at "EnterInput" is only taken into account when "ContinuousEnterFunct" = FALSE. If this condition is fulfilled, an edge change 0 → 1 causes the setpoint entered in "SetpointInput" to be adopted and transferred. This also applies when the newly entered setpoint is identical to the last setpoint transferred. This method of setpoint adoption is suitable for input vis hardware, for example via a console or control panel. It can also be used for entering setpoints at an operator panel (OP). In this the adoption must be triggered by a function key on the OP. If you do not require the parameter, simply leave it open.	

Parameter:	**ContinuousEnterFunct**
Declaration:	INPUT
Data type:	BOOL

Range of values:	TRUE / FALSE
Default:	TRUE
Explanation:	Apply setpoint continuously for software setpoint With this parameter, you specify whether the setpoint at "SetpointInput" should be continuously read in and checked for changes. Changes are checked by comparison with the last setpoint that was transferred. This method of setpoint adoption is suitable for input via suitable software. It can, however, also be used for entering setpoints at an operator panel (OP) if the OP does not have a separate function key that can be used to trigger the input. If you do not require the parameter, simply leave it open.

Parameter:	**SetpointInput**	
Declaration:	INPUT	
Data type:	WORD	
Range of values:	See address range	
Default:	0 (W#16#0)	
Address range:	Input words	IW0 ... IWn PIW0 ... PIWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n

Explanation: Setpoint input

How a setpoint available at SetpointInput is processed depends on whether it is a hardware or software input. You specify the type of input with the "ContinuousEnterFunct" parameter:

- **ContinuousEnterFunct = FALSE (hardware input)**
  A setpoint at "SetpointInput" is only read in as long as the signal is 1 at "EnterInput". The setpoint that is read in is then transferred if no error is detected during the 1-out-of-n check, and if the central enable memory bit is set. This is automatically set by FC Safe following a selected time delay set there (see FC Safe, "InputDelayTime" parameter).
  A further setpoint is then only read in again by the FB when a 0 signal was detected at EnterInput for at least one OB1 cycle.
  When a 1-out-of-n error is detected at the hardware input, the entered setpoint is no longer processed. A new setpoint is read in again only when previously for the length of an OB1 cycle in the CPU, no hardware input via a command, setpoint or parameter block was detected.
  The FB enters the detected 1-out-of-n error in the diagnostic buffer (event ID B172). As long as the error remains, the error status is indicated via the "InputError" output of FC Safe (see FC Safe, "InputError" parameter).
- **ContinuousEnterFunct = TRUE (software input)**
  A setpoint at SetpointInput is read in continuously and checked for changes. Changes are checked by comparison with the last setpoint that was transferred. The setpoint is sent immediately every time a change occurs unless the 1-out-of-n check detects an error.
  - With hardware input (see EnterInput) an empty cycle must be detected before a new setpoint can be transferred by the block.
  - With software input a new setpoint can be transferred in every OB1 cycle. This assumes that there is no other software entry by another block pending in this cycle. Otherwise a 1-out-of-n error is detected.
    A new transfer of the software setpoint can be triggered via the "SendSoftSetpoint" input even when the software input has not changed (see below).
    When a 1-out-of-n error is detected at the software input, the entered setpoint is no longer processed.
    A new setpoint is read in again only when previously for the length of an OB1 cycle in the CPU, no software input (command or setpoint) was detected. The block enters detected 1-out-of-n errors in the diagnostics buffer (event ID B172). Appropriate error bits are also set in the central data block BasicData where they can be queried by the software. For further details, refer to the description of FC Safe.

Parameter:	**ReturnedSetpoint**
Declaration:	OUTPUT
Data type:	WORD
Range of values:	See address range
Default:	0 (W#16#0)

Address range:	Output words	QW0 ... QWn PQW0 ... PQWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n

Explanation: Output for a returned setpoint

The partner object receiving the setpoint reports back the currently valid setpoint there. This value is displayed at the "ReturnedSetpoint" output.

If the partner object is set to 'local' and a new input is made there, the setpoint changed locally is indicated here by "ReturnedSetpoint".

After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the locally valid setpoint is indicated by "ReturnedSetpoint".

If you do not require the parameter, simply leave it open.

Parameter:	**LocalOperation**	
Declaration:	OUTPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Output	Q 0.0 ... Q n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Return message from the partner object: The object is set to local operation.

A setpoint can also be set locally at the partner object that receives the setpoint. The partner object Set01W_R must then be set to ’local’ at the "Local" input parameter. The current status of the "Local" input parameter is reported by the partner object and indicated here at the "LocalOperation" output.

After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the local currently valid status is displayed at "LocalOperation".

If you do not require the parameter, simply leave it open.

Parameter: **NewData**

Explanation: For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844).

Whenever the block has received new data and has output it to the output "ReturnedSetpoint" or "LocalOperation", the "NewData" output is set to TRUE for one OB1 cycle.

Parameter:	**SendSoftSetpoint**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Memory bit	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Trigger input for resending the last software setpoint. For further details, refer to the "SetpointInput" parameter. If you do not require the parameter, simply leave it open.	

Setpoint typical Set01W_R

Function

Receive 1 setpoint in the station as a 16 bit value or enter locally and return the local setpoint to the master station

Parameter

Parameter:	**PartnerNo** **PartnerObjectNo** **Enabled** **ImageMemory** **Conditional** **Unconditional** **TimeStamp** For a description, see section Reoccurring parameters (Page 1844).

Parameter:	**Local**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Release of the local setpoint input of the data area specified by "LocalSetpointInput".

A setpoint sent by the partner (master station) is not accepted by the object as long as "Local" = TRUE.

The current status of the "Local" input is transmitted to the partner together with a copy of the setpoint which is currently being output at "SetpointOutput" (setpoint mirroring).

Bumpless switchover:

- When there is a switchover from "Local" = 0 to 1, the last values output at the "SetpointOutput" output are retained until a new setpoint is entered via the local input area "LocalSetpointInput".
- When there is a switch back from "Local" = 1 to 0, the last value output at the "SetpointOutput" output is retained until the block receives a new setpoint from the partner.

Read the note on the "ContinuousEnterFunct" parameter.

Parameter:	**EnterInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Enter input for local setpoint input.

Edge triggered, the parameter triggers adoption of the setpoint at the setpoint input "LocalSetpointInput".

A signal change at "EnterInput" is only taken into account when the value TRUE is set at the "Local" input parameter and "ContinuousEnterFunct" = FALSE. If these conditions are fulfilled, a signal change from 0 to 1 at the causes the setpoint entered at "LocalSetpointInput" to be adopted and output at "SetpointOutput".

This method of setpoint adoption is suitable for hardware input, for example via a console, control panel or an operator panel (OP). With an OP, the adoption must be triggered by a function key.

If you do not require the parameter, simply leave it open.

Parameter:	**ContinuousEnterFunct**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Continuous local setpoint acquisition When "Local" = TRUE and "ContinuousEnterFunct" = TRUE the setpoint at "LocalSetpointInput" is continuously read in and checked for changes. The setpoint read in at "LocalSetpointInput" is output at the "SetpointOutput" output if at this time the local input value differs from the last returned setpoint. If an array is detected at the setpoint input "LocalSetpointInput" this is output immediately at "SetpointOutput" (without a change check). This method of setpoint adoption is suitable for software input. It can, however, also be used for entering setpoints at an operator panel (OP) if the OP does not have a function key. If you do not require the parameter, simply leave it open.

Parameter:	**LocalSetpointInput**	
Declaration:	INPUT	
Data type:	WORD	
Range of values:	See address range	
Default:	0 (W#16#0)	
Address range:	Input words	IW0 ... IWn PIW0 ... PIWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n
Explanation:	Local setpoint input A value at "LocalSetpointInput" is only adopted if "Local" = TRUE. If this condition is met, how a pending setpoint is processed depends on whether it is a hardware or software input. The type of input is decided by the "ContinuousEnterFunct" parameter:	

- ContinuousEnterFunct = FALSE
  Hardware input
  A setpoint at "LocalSetpointInput" is only read in when a signal change 0 → 1 is detected at "EnterInput". The setpoint entered locally is output via the "SetpointOutput" output and transferred to the partner.
  A further setpoint is then only read in again by the block when a 0 signal was detected at "EnterInput" for at least one OB1 cycle.
- ContinuousEnterFunct = TRUE
  Software input
  A setpoint at "LocalSetpointInput" is read in continuously and checked for changes. The change check is achieved by comparison with the last valid setpoint; in other words, the value stored as the returned setpoint. With each change, the setpoint is output via the output set at "SetpointOutput" and transferred to the partner.
  With software input a new setpoint can be entered in every OB1 cycle.
  With hardware input an empty cycle must be detected before a new setpoint can be read in by the block.

If you do not require the parameter, simply leave it open.

Parameter:	**SetpointOutput**	
Declaration:	OUTPUT	
Data type:	WORD	
Range of values:	See address range	
Default:	0 (W#16#0)	
Address range:	Output words	QW0 ... QWn PQW0 ... PQWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n
Explanation:	Setpoint output word. The setpoint sent by the partner object or entered locally at "LocalSetpointInput" is output at the "SetpointOutput" output.	

Parameter:	**NewData**
Explanation:	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 1844). Whenever the block has received a new setpoint from the partner object and has output it to "SetpointOutput", the "NewData" output is set to TRUE for one OB1 cycle. This also applies when there is a new local setpoint input (Local = 1).

Optional blocks

ListGenerator1500/300/400 FC

The ListGenerator FC is available in three versions with the following symbolic names:

- For S7-300
  ListGenerator300
- For S7-400
  ListGenerator400
- For S7-1500
  ListGenerator1500

Function

The FC ListGenerator is required in a CPU that receives data containing either an incomplete destination address or no destination address at all. The lack of the destination object number is the most important factor here because this points to the instance DB in which the received information should be stored.

Missing or incomplete destination addresses can occur when no or incomplete parameters are set for them in the station. This is permitted for typicals that send binary information, analog values or counted values. If these typicals send data to more than one destination, no destination address is set for them. Due to the missing destination information, the send frame is automatically transmitted to all destinations for which a connection was set up during configuration. This data is therefore received without a destination address at the various destinations.

Note

Supplement to destination subscriber no.

Data frames to be sent without a destination address have the destination subscriber number added by the sending TIM, and sometimes several if there are several destinations.

The TIM enters 0 in the address field for the destination object number, since the TIM does not have the relevant information. The only destination subscribers it knows are those to which it has a configured connection.

At the receiving end, the data frame therefore contains the destination subscriber number but the destination object number is 0.

If the destination object number is not contained in the received data frame, FC Distribute, which is responsible for distributing the received frames, references an object reference list.

Using the source address (source subscriber no. + source object no.) contained in every data frame, FC Distribute searches through the list for an entry that specifies the missing destination object number for the given source address; in other words, it searches for the number of the local instance DB.

This object reference list is created by FC ListGenerator. The FC has no parameters. It is linked into the cyclic user program (OB1) following the FC BasicTask.

When creating the list, FC ListGenerator uses the addresses set in the parameters for the receiving typicals. Specification of "PartnerNo" and "PartnerObjectNo" is mandatory for these typicals. These parameters are identical to the source address in the corresponding receive frame. Since the typical also knows the number of its instance DB, it therefore knows all the addresses required for an entry in the reference list.

During startup, FC ListGenerator arranges that all receiving typicals enter the addresses from their parameter assignment with the number of the instance DB in the reference list. The object reference list therefore does not require special parameter settings, it is simply created from the existing parameters of the receive typicals and is therefore always consistent.

How it works

FC ListGenerator creates the lists after startup in three consecutive OB1 cycles:

- In the first cycle, it determines how many entries will be required in the first and, if applicable, in the second object reference list. The typicals involved only increment a counter during this run.
- In the second cycle, FC ListGenerator generates the data block for the first and, if applicable, the second object reference list with the required length and enters 0 in all the data words. During the same cycle, all typicals involved enter their addresses and the number of the corresponding instance DB in the list.
- In the third and final cycle, FC ListGenerator sorts all the entries in ascending order. Sorting speeds up the search in the list during actual operation.

When generating the data block, FC ListGenerator does the following:

If no list has yet been created, a free DB number is searched for. Starting with the number of DB BasicData, the number of the next lowest free DB is taken.

If a list already exists, FC ListGenerator checks to see whether the existing DB is long enough for the currently required number of references. If the length is adequate, 0 is entered as the content and the addresses are written again and sorted.

If the existing data block is too short, different procedures are used for the various SIMATIC product series:

- For S7-300
  A new DB is generated. The old DB remains in memory because S7-300 has no delete function for data blocks.
  With the 300 CPU, you have to delete the old DBs with the programming device.
  Note:
  Without generating a new DB, you would need to delete this DB with the programming device. If there is not enough memory on the CPU to be able to generate a new DB, you need to delete the existing DB before restarting.
- For S7-400
  The existing DB is deleted, the memory is compressed and the DB is then generated with the same number in the new length.
  With the 400 CPU, you may be able to manually compress the memory or reload the CPU.
- For S7-1500
  The existing DB is deleted and generated in the new length with the same number.
  With the 1500 CPU, the memory is automatically compressed, just like with the 300 CPU.

If the ListGenerator FC can no longer generate a DB, an error message is written to the diagnostic buffer of the CPU:

- 0xB107 "Error generating the object reference list"
  - DB[Info1]
    DB[Info1] cannot be created.
  - Cause: [Info2].
    In Info2, the return value of the SFC Create_DB function is output, refer to the description there.

FC PartnerMonitor

Function

FC PartnerMonitor has the following functions:

- It displays important status information about a SINAUT subscriber (see "PartnerStatus" parameter).
- The FC can also be used to trigger a general request of the subscriber, except to a control center (e.g. ST7cc/ST7sc).
- It can also be used to establish and terminate a permanent connection to the subscriber.

The FC can be called at any point in the cyclic user program (in OB1).

If you want to monitor and control more than one subscriber, include an appropriate number of FC PartnerMonitor in the user program.

A SINAUT subscriber (partner) can only be an ST7 CPU or an ST7cc to which a connection was configured. TIMs cannot be monitored or controlled by FC PartnerMonitor.

Note

FC PartnerMonitor in a station

FC PartnerMonitor can also be used in a station. However, the control inputs for establishing and terminating a permanent connection can then no longer be used. This only works in the master station when the local TIM is a master station TIM.

Parameter

Name:	**PartnerNo**
Declaration:	INPUT
Data type:	INT
Range of values:	1 ... 32000 [ms]

Explanation SINAUT subscriber no. of the subscriber to be monitored and controlled.

If the set "PartnerNo" is not found in the administration (DB BasicData), then (only during startup) an entry is written to the diagnostics buffer (event ID B101). The CPU does not change to STOP.

The status of a subscriber with correct parameter settings is indicated in the "PartnerStatus" output word and the control inputs are processed.

An unknown subscriber is not processed until the error is eliminated. The "PartnerStatus" output word remains set to 0 during this time.

Name: **MaxConnectTime**

Declaration: INPUT

Data type: INT

Range of values:
- 0 (no limitation)
- 1 … 480 [minutes]

Explanation Maximum duration of a permanent connection

The time specified here (> 0) is activated at the start of a permanent connection (see "PermanentCall_On" parameter).

If the time elapses before the permanent connection is reset, it is automatically disconnected. The time is retriggered as long as the signal 1 is present at the "PermanentCall_On" input.

The time specified here applies to a permanent connection in a dial-up network as well as to a permanent connection (continuous polling) on a dedicated line.

Name: **PartnerStatus**

Declaration: OUTPUT

Data type: WORD

Range of values:

Output words	QW0 ... QWn PQW0 ... PQWn
Memory words	MW0 ... MWn LW0 ... LWn
Data words	DBm.DBW0 ... n

Explanation Output word to indicate the status of the subscriber to be monitored.

If you do not require the parameter, simply leave it open.

The meaning of the status bits in the "PartnerStatus" output word:

Bit	.0	Status of the subscriber
	0	0 = Subscriber disrupted
	1	1 = Subscriber OK

Bit	.1	Status of the redundant connection
	0	0 = Redundant connection is disrupted
	1	1 = All connections OK.

Bit	.3	.2	Status of the general request (GR)
	0	0	0 = GR complete without error
	0	1	1 = GR started
	1	0	2 = GR start received
	1	1	3 = GR finished with error (GR incomplete or cannot be executed, e.g. due to fault at subscriber)

Bit	.6	.5	.4	Status of the dial-up connection
	0	0	0	0 = No connection
	0	0	1	1 = Outgoing call activated
	0	1	0	2 = Incoming call established
	0	1	1	3 = Outgoing call established
	1	0	0	4 = Permanent connection registered
	1	0	1	5 = Permanent connection established
	1	1	0	6 = Permanent connection disconnected

Bit	.7	Status of the dial-up connection
	0	0 = No dial-up connection check in background
	1	1 = Dial-up connection check in background is activated

Bit	.8	Status of continuous polling (on dedicated line)
	0	0 = No continuous polling
	1	1 = Continuous polling activated

Bit	.9	Status of the WAN connection resources
*)	0	0 = Sufficient resources on partner
	1	1 = Insufficient resources on partner

Bit	.10	Time status
	0	0 = Date/time not available / not OK on partner
	1	1 = Date/time OK on partner

Bit	.11	Time-of-day synchronization
	0	0 = The partner CPU received a plausible time during the last synchronization or no time of day has been received since startup.
	1	1 = The partner CPU has received an implausible time of day; the last valid time will continue to be used. (Display only with TimeMask > V1.6)

Note

In-out parameters

The following parameters are in/out parameters (declaration IN_OUT):

- GeneralRequest
- PermanentCall_On
- PermanentCall_Off

It is difficult to specify local bit memory with this parameter type and this should not be used.

Name:	**GeneralRequest**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation	Input for triggering a general request to the subscriber specified with PartnerNo. A general request to the subscriber is triggered with a 1 signal at this input if no request is active for this subscriber at this time. The input is then automatically reset by the FC. If an input of a digital input is specified (I 0.0 ... I n.7), you are responsible for resetting the signal at the input. Reset the signal before ending the currently running general request so that another general request is not triggered immediately.	

Name:	**PermanentCall_On**
Declaration:	IN_OUT
Data type:	BOOL

Range of values:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation

Input for triggering a permanent connection to the subscriber specified with PartnerNo.

A permanent connection to the subscriber is triggered with a 1 signal at this input if there is currently no permanent connection to this subscriber. The input is then automatically reset by the FC. If an input of a digital input is specified (I 0.0 ... I n.7), you are responsible for resetting the signal at the input at the latest before the termination of the existing permanent connection.

A 1 signal at the input "PermanentCall_On" also activates the time specified with "MaxConnectTime" if it is greater than 0.

Depending on whether the subscriber can be reached over a dial-up connection or a dedicated line, the command to establish the permanent connection is processed as follows and indicated at the "PartnerStatus" output:

- **For a dial-up connection:**
  A dial-up connection is established by the master TIM to the appropriate subscriber and, regardless of the data traffic, maintained until the terminate command is sent.
  The current status of the permanent connection is indicated in the PartnerStatus output word with the bits 4 ... 6 (see PartnerStatus parameter).
- **For a dedicated line:**
  In this case the master TIM operates in polling mode with the stations. A permanent connection is implemented in this case by ’continuous polling’ of the subscriber. This is actually an intermittent poll to the subscriber; in other words, the other subscribers on the dedicated line network are still polled but the preferred subscriber is polled again after every poll to a ’normal’ subscriber .
  The current status of the continuous polling is indicated by bit 8 in the "PartnerStatus" output parameter.

Special feature with stations:

A permanent connection cannot be established from a station. This control input cannot therefore be used when FC PartnerMonitor is used in a station.

Name:	**PermanentCall_Off**
Declaration:	IN_OUT
Data type:	BOOL

Range of values:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation

Terminating an existing permanent connection

The input serves to trigger termination of an existing permanent connection to the subscriber specified with "PartnerNo".

A permanent connection to the subscriber is terminated with a 1 signal at this input if there is currently a permanent connection to this subscriber. The input is then automatically reset by the FC. If an input of a digital input is specified (I 0.0 ... I n.7), you are responsible for resetting the signal at the input via the user program. This should be done at the latest before establishing a permanent connection again.

Depending on whether the subscriber can be reached over a dial-up connection or a dedicated line, the command to terminate the permanent connection is processed as follows and indicated at the "PartnerStatus" output:

- **For a dial-up connection:**
  The existing dial-up connection is terminated by the master TIM but only after any pending data has been sent.
  The current status of the permanent connection is indicated in the PartnerStatus output word with the bits 4 ... 6 (see PartnerStatus parameter).
- **For a dedicated line:**
  The master TIM deletes the registration for continuous polling of the corresponding subscriber. The polling cycle for all connected subscribers continues in normal mode.
  The current status of the continuous polling is indicated by bit 8 in the "PartnerStatus" output parameter.
  Continuous polling can also be canceled on a dedicated line by instructing the master TIM to start continuous polling of another subscriber. The existing job is then replaced by the new one.

Special feature with stations:

A permanent connection cannot be terminated by a station. This control input cannot therefore be used when FC PartnerMonitor is used in a station.

FC PartnerStatus

Function

The FC PartnerStatus can show the current status ’disrupted’ or ’OK’ for a maximum of 8 SINAUT subscribers.

The FC can be called at any point in the cyclic user program (in OB1).

If you want to monitor more than 8 subscribers, include the appropriate number of FCs PartnerStatus in the user program.

The partner can be an ST7 CPU or an ST7cc to which a connection was configured, or a local TIM.

One bit per subscriber is reserved in the "PartnerStatus" output byte to indicate the status of the respective subscriber:

- FALSE (0):
  - Subscriber disrupted
  - Corresponding input parameter not used (= 0)
  - Subscriber unknown
- TRUE (1): Subscriber OK

Parameter

Name:	**Partner1 ... Partner8**
Declaration:	INPUT
Data type:	INT
Range of values:	• 0 Value for unrequired parameter • 1 ... 32000 Number of the subscriber to be monitored
Explanation	SINAUT subscriber number of the subscriber to be monitored If a set subscriber number is not found in the administration (DB BasicData), then (only during startup) an entry is written to the diagnostics buffer (event ID B101). The CPU does not change to STOP. The status of a subscriber with a correct parameter assignment is indicated in the "PartnerStatus" output byte. An unknown subscriber is not processed until the error is eliminated. Their status bits are set to 0.

Name:	**PartnerStatus**	
Declaration:	OUTPUT	
Data type:	BYTE	
Range of values:	Output bytes	QB0 ... QBn PQB0 ... PQBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n
Explanation:	Indication of the status of the subscriber to be monitored. Assignment of the status bits in the "PartnerStatus" output byte depending the parameters Partner1 ... "Partner8":	

Bit	.7	.6	.5	.4	.3	.2	.1	.0
Partner	8	7	6	5	4	3	2	1

Value:

- 0 = partner disrupted, not set in the parameters or unknown
- 1 = partner OK

FC PathStatus

Function

The block (FC) shows the status of the path to a partner from the perspective of the local TIM.

A maximum of 2 paths (main and substitute path) to a partner can be configured. Both paths must begin or end on a local TIM.

The block shows the following:

- The paths via which the partner can be reached.
- The path currently being used
- The TIM interface via which the main path was configured.
- The TIM interface via which the substitute path was configured.

The path of a connection is specified as a combination of the used interfaces of the TIM and the status of the path.

In the output byte PathStatus the following bits are reserved:

- Two bits for the interface of the main path
- Two bits for the interface of the substitute path
- Two bits for the status of the main path
- Two bits for the status of the substitute path

The FC can be called at any point in the cyclic user program (OB1) after calling the FC BasicTask.

If the status of the path to more than one subscriber is to be shown, a corresponding number of FC calls need to be programmed in the user program program.

Parameter

Name:	**Partner**
Declaration:	INPUT
Data type:	INT
Range of values:	1 ... 32000 (subscriber number)
Explanation:	SINAUT subscriber no. of the partner

Name:	**PathStatus**
Declaration:	OUTPUT
Data type:	BYTE

Range of values:	Output bytes	QB0 ... QBn PQB0 ... PQBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n
Explanation:	Display of the path status of the connection to the partner	

PathStatus - Coding

Table 1-146 Coding of the status bits in the "PathStatus" output byte

Bits 6 + 7	Bits 4 + 5	Bits 2 + 3	Bits 0 + 1
Configured interface		Path status	
No. for substitute path	No. for main path	Substitute path (2nd path)	Main path (1st path)

Configured interface (bits 4..7)

The TIM interfaces in the block are consecutively numbered decimally from 0 to 3, see "No." column in the table. The table shows the coding of the bits for the different TIM types.

Table 1-147 Coding of bits 4 + 5 (interface of the main path) or bits 6 + 7 (interface of the substitute path)

Status bit 5 (7)	Status bit 4 (6)	Interfaces			
		No.	TIM 3V-IE	TIM 4R-IE	TIM 1531 IRC
0	0	0	Ethernet "IE1" (X2)	Ethernet "IE1" (X3)	Ethernet "IE1" (X1)
0	1	1	-	Ethernet "IE2" (X4)	Ethernet "IE2" (X2)
1	0	2	Serial "WAN1" (X1)	Serial "WAN1" (X1)	Ethernet "IE3" (X3)
1	1	3	-	Serial "WAN2" (X2)	Serial "WAN" (X4)

Path status (bits 0..3)

- Main path = 1. Path (bits 0 + 1)
- Substitute path = 2nd path (bits 2 + 3)

Table 1-148 Status table: Coding of bits 0 + 1 or bits 2 + 3

Status bit 1 (3)	Status bit 0 (2)	Meaning bit 1	Meaning bit 0
0	0	Path not current	Subscriber not reachable
0	1	Path not current	Subscriber reachable
1	0	Path current	Subscriber not reachable
1	1	Path current	Subscriber reachable

FC PulseCounter

Function

The FC PulseCounter is responsible for count pulse acquisition.

A maximum of 8 pulse strings are detected via digital inputs and fed to the function blocks with the aid of SIMATIC counters that put together the counted value frames (Cnt01D_S, Cnt04D_S).

The acquisition of the count pulses is time-controlled To do this the FC PulseCounter must be included in a cyclic interrupt OB, e.g. OB35. The call interval of the cyclic interrupt OB must be matched to the pulse duration of the count pulses. You will find more information on count pulse acquisition with the cyclic interrupt OB in the section Cyclic interrupt OB (Page 1839).

Parameter

Name:	**InByte**	
Declaration:	INPUT	
Data type:	BYTE	
Range of values:	Input bytes	PEB0 ... PEBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n
	If an input byte of a digital input is specified, this must be the address of the I/O byte (PIB) directly from the digital input modules. The current status of the count input can only be detected reliably by direct access. When reading out from the process image of the inputs (PII) count pulses could remain undetected.	
Explanation:	Input byte for count pulses. The parameters for inputs for count pulse acquisition can can be set in bytes.	

Name:	**EnableMask**
Declaration:	INPUT
Data type:	BYTE
Range of values:	B#16#00 ... B#16#FF

Explanation Enable mask for the counting inputs

With this parameter, it is possible to specify in the form of a bit mask at which inputs in the input byte count pulses are actually connected. The following applies to every bit in the bit mask:

- 0 = Input bit blocked for acquisition
- 1 = Input bit enabled for acquisition

The input of the mask is only permitted in hexadecimal format B#16#00 to B#16#FF.

Input as an 8-bit binary number from 2#0 to 2#1111 1111 is not possible with the data type BYTE.

The assignment of the bits in the mask to the inputs in the input byte "InByte":

InByte	.7	.6	.5	.4	.3	.2	.1	.0
EnableMask B#16#	0 … F					0 … F		

Example: EnableMask B#16#83

Enabled are: Inputs .7, .1 and .0

Blocked are: Inputs .6 to .2

Name: **CntIn_0 ... CntIn_7**

Declaration: INPUT

Data type: COUNTER

Range of values: C0 or C1 ... Cn
(n depends on the CPU)

Explanation Pulse counter

For each of the enabled counting inputs a SIMATIC counter needs to be specified for the corresponding parameter CntIn_0 ... . "CntIn_7". With each pulse acquired, the SIMATIC counter is incremented.

The counters set here must be specified as input counters (parameter Counter_1 ... _4) in the actual counted value function blocks "Cnt01D_S" and "Cnt04D_S". These function blocks out read the assigned counter and then reset it.

As the placeholder for parameters that are not required, it is recommended to specify the counter C0.

Example of the parameter assignment of "CntIn_0" ... "CntIn_7" starting at "EnableMask" = B#16#83:

CntIn_0 : = Z10
CntIn_1 : = Z11
CntIn_2 : = Z0
CntIn_3 : = Z0
CntIn_4 : = Z0
CntIn_5 : = Z0
CntIn_6 : = Z0
CntIn_7 : = Z12

FC ST7ObjectTest

Validity

S7-300/400 CPU

Function

Calling FC ST7ObjectTest in programming error OB121 prevents a CPU STOP, if the CPU receives data with an unknown destination object no.

FC ST7ObjectTest checks why OB121 was called, i.e. which block type is missing.

- If the missing block is a data block and this data block is an instance DB of a SINAUT object, the CPU does not change to STOP.
- If no SINAUT instance DB is missing but rather another block (DB, FB, FC), you can specify the reaction with the parameter "StopInOtherCases".
  - CPU changes to STOP.
  - CPU continues to run.

For more information on the programming error OB121 and background information relating to the use of FC ST7ObjectTest, see section Programming error OB (Page 1839).

Parameter

Name:	**StopInOtherCases**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Explanation:	The CPU should change to STOP in other error situations. For details of the parameter see above, section "Function". • TRUE: CPU changes to STOP. • FALSE: CPU continues to run.

FC TestCopy

Function

Using FC TestCopy, extracts of the data traffic between ST7 subscribers can be recorded or the entire traffic can be recorded. With search masks to be set in the control field of DB TestCopyData individual frame types can be filtered out and then copied from the send or receive buffer for further evaluation in the DB TestCopyData. For details, see below.

Send and receive frames are all stored in the same data block DB TestCopyData. This makes it simple to track the chronological order of the copied send and receive frames.

Requirements

To use the TestCopy function, the user program must meet the following conditions:

- The FC TestCopy function must be available on the CPU.
- DB TestCopyData must be present on the CPU and have a sufficient length.
  - To achieve this, copy DB TestCopyData (DB99) from the TD7 library to your CPU.
  - If necessary, change the length of the buffer area in the DB by increasing or decreasing the size of the array "TestCopyBuffer" in DW40, which has the default length of [0..240] WORD.
- Make the following entries in the respective communication DB of the CPU 300/400 (BComData / XComData / PComData) whose send and/or receive frames you want to write:
  - In DW32 (TestCopyDBNo) of the communication DB, enter the number of the DB TestCopyData.
  - In DW34 (TestCopyFCNo) of the communication DB, enter the number of the FC TestCopy.

  Proceed in the same way for the communication DB of CPU 1500. The tags in DB BConnectData have the same names.

Linking FC TestCopy into the user program

If the above-mentioned conditions are fulfilled, the test function is processed cyclically by the respective communication FB of the CPU.

FC TestCopy cannot be called in the user program.

Monitoring the written data

Use the ready-made watch table "TestCopyMonitor", which you can copy from the TD7 library into the "Watch and force tables" directory of the CPU.

If the settings are to be retained even after CPU startup, they can also be stored directly in the start values of the individual BConnection instances of the DB BConnectData.

DB TestCopyData

Structure of the TestCopyData DB

Areas of the TestCopyData DB

The DB for the TestCopy function is divided into the following areas (after Offset in the DB):

- **0 ... 27: User interface**
  Interface for setting the TestCopy mode and functions. The area is divided into:
  - 1 ... 13
    Filter settings for RecvCopy function and number of counted received frames
  - 15 ... 25
    Filter settings for SendCopy function and number of counted sent frames
- **28: Error display**

- **31 ... 39: Internal management pointer**
- **40 ... (Default: 523): Buffer range**
  Buffer area for storing frames that match the filter criteria.
  The buffer area must be configured as an array [0...xxxx] of WORD.

The following table shows the structure of the DB TestCopyData:

Data type / (offset)		Tag name	Format	Explanation
User interface				
DBB	0	OperationMode	BYTE	Mode
DBW	12	Recv_TgramCounter	INT	Number of copied receive frames
DBW	26	Send_TgramCounter	INT	Number of copied send frames
RecvCopy function				
DBB	1	Recv_TgrmType	BYTE	Receive filter: Message type (MT)
DBW	2	Recv_DestSubscr	INT	Receive filter: Destination subscriber no.
DBW	4	Recv_DestObject	INT	Receive filter: Destination object no.
DBW	6	Recv_SourceSubscriber	INT	Receive filter: Source subscriber no.
DBW	8	Recv_SourceObject	INT	Receive filter: Source object no.
DBW	10	Recv_StartIndex	INT	Receive filter: Start index no.
DBB	14	SpareDBB14	BYTE	*Reserve*
SendCopy function				
DBB	15	Send_TgrmType	BYTE	Send filter: Message type (MT)
DBW	16	Send_DestSubscr	INT	Send filter: Destination subscriber no.
DBW	18	Send_DestObject	INT	Send filter: Destination object no.
DBW	20	Send_SourceSubscriber	INT	Send filter: Source subscriber no.
DBW	22	Send_SourceObject	INT	Send filter: Source object no.
DBW	24	Send_StartIndex	INT	Send filter: Start index no.
Error display				
DBB	28	FC_RetVal	BYTE	Error information: 0 = No error 1 = DB TestCopyData too short 10 = Unknown mode
DBB	29	SpareDBB29	BYTE	*Reserve*
DBB	30	SpareDBB30	BYTE	*Reserve*
Internal management pointer				
DBB	31	TestCopyStatus	BYTE	Status byte for TestCopy operation
DBB	32	TestCopyCmdByte	BYTE	Command byte for TestCopy operation
DBB	33	TestCopyDelCount	BYTE	Loop counter for TestCopy delete function
DBW	34	NextFreeCopyByte	INT	Address of the next free TestCopyBuffer byte

Data type / (offset)		Tag name	Format	Explanation
DBD	36	StartTimeSFC64	DINT	SFC64 time at the start of the copy procedure
Buffer range				
DBB	40	TestCopyBuffer[0]	BYTE	Copy area, byte 0
DBB	41	TestCopyBuffer[1]	BYTE	Copy area, byte 1
DBB	42	TestCopyBuffer[2]	BYTE	Copy area, byte 2
DBB	43	TestCopyBuffer[3]	BYTE	Copy area, byte 3
DBB	n	TestCopyBuffer[n]	BYTE	Copy area, byte n

Structure of a copied message block

A frame block can contain several frames. The frames are saved in the DB TestCopyData according to the following rules:

1. The first entry indicates the time difference in milliseconds (7 decade BCD plus sign) since the last selection of an operating mode > 0.
2. This is followed by a separation sign AAAA for sent messages, EEEE for received messages.
3. Storage of the first message from the frame block.
4. Separation identifier AAAA, or EEEE:
5. Storage of the last frame from the message block.
6. Block end identifier FFFF.

Example

- All received frames will be stored in DB TestCopyData.
- Communication via X blocks, i.e. max. 76 bytes per receive block.
- The receive buffer of the DB XComData is the source for FC TestCopy.
- The current receive block contains 3 messages.

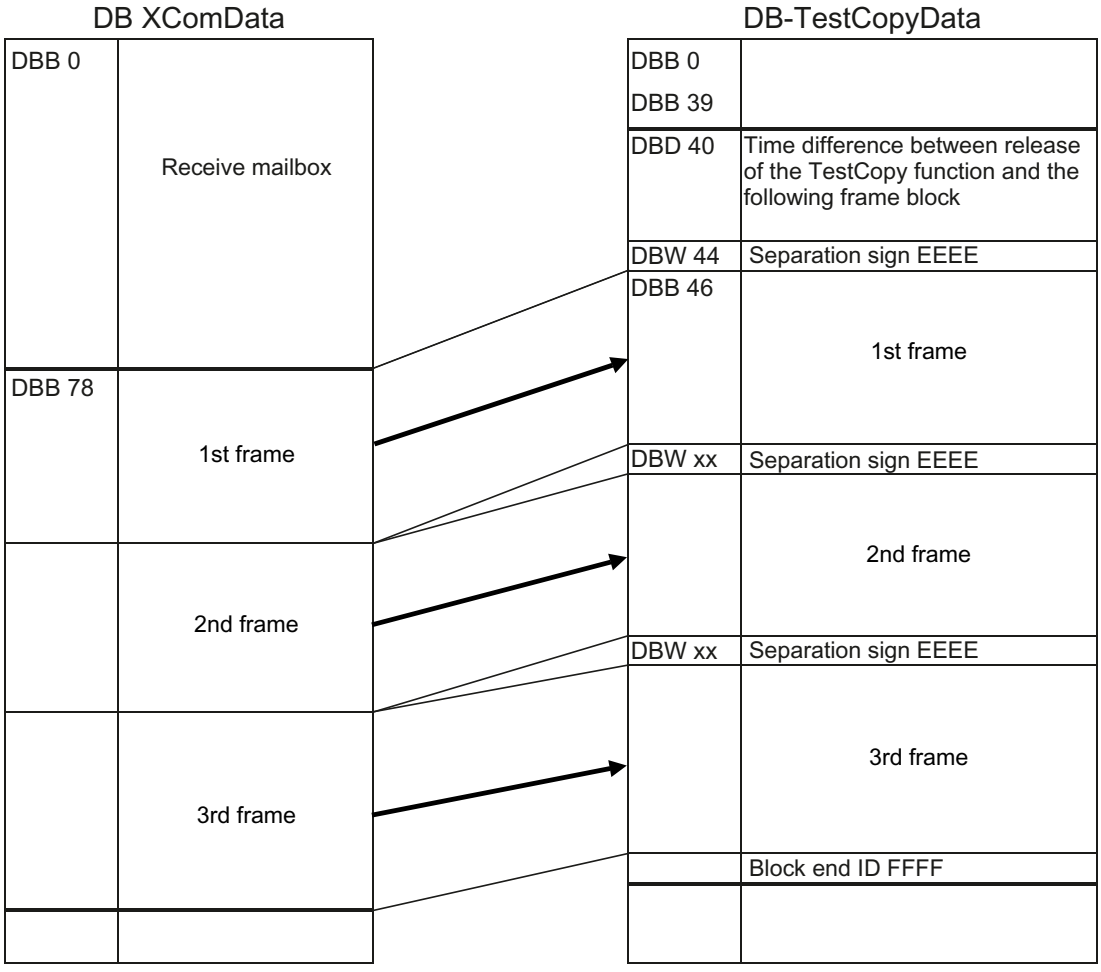

Figure 1-81 Example of filling the DB TestCopyData

Length calculation

FC TestCopy uses the following parameters for determining the minimum length for the DB TestCopyData:

Parameter	Parameter name	Length
Length of communication buffer	$Len_{ComBuffer}$	76 or 202 bytes *
Minimum frame length	$Len_{MinTgrm}$	14 bytes
Offset management area	Offset	40 bytes
Length of the time difference	Len_{dt}	4 bytes
Length of the block separators	$Len_{Trenner}$	2 bytes

* 76 with X communication, 202 with B communication

The formula used for the actual calculation is the same for X communication and B communication. The results differ only due to different lengths for the communication buffer for X and B communication:

- **Length with X communication**

Len_{Min_Xcom}	$=Len_{ComBuffer} + Offset + Len_{dt} + (Len_{ComBuffer} / Len_{MinTgrm} + 1) * Len_{Trenner}$
	= 76 + 40 + 4 + (76 / 14 + 1) * 2
	= 120 + 12 = 132 bytes minimum

- **Length with B communication**

Len_{Min_Bcom}	$=Len_{ComBuffer} + Offset + Len_{dt} + (Len_{ComBuffer} / Len_{MinTgrm} + 1) * Len_{Trenner}$
	= 202 + 40 + 4 + (202 / 14 + 1) * 2
	= 236 + 40 = 276 bytes minimum

If FC TestCopy determines that the DB TestCopyData does not have the calculated minimum length an error message to this effect is entered in data byte DBB28.

Operation of TestCopyMonitor

Operating modes and transfer directions of DB TestCopyData

In byte 0 (OperationMode) of the DB TestCopyData, the mode of the FC is coded. The values 0 ... 3 identify the mode:

- Mode 0
  Function blocked
- Mode 1
  Frame entry as of the start of DB TestCopyData
- Mode 2
  Write to DB TestCopyData endlessly as circulating buffer
- Mode 3
  Fill DB TestCopyData once, then set mode 0.

You will find the detailed coding in the table below.

Along with the mode the coding of DBB0 also includes the transfer direction of the logged data. The following assignment applies for the coding:

- Bit 0...3
  Modes (0, 1, 2, 3) for the direction "RecvCopy" (copy receive frames)
- Bit 4...7
  Modes (0, 1, 2, 3) for the direction "SendCopy" (copy send frames)

Bit	.7	.6	.5	.4	.3	.2	.1	.0
Direction	SendCopy				RecvCopy			

The functions "SendCopy" and "RecvCopy" can be activated individually or at the same time, however only a common mode is possible for both communication directions.

When a mode > 0 is set in the less significant half byte, this always applies to both communication directions.

Only if mode 0 is coded in bit 0...3 (RecvCopy), does the value in bit 4...7 apply (SendCopy).

Exception:
To delete the DB TestCopyData, DBB0 must be written with FF, 0F is not enough.

Examples of coding of DBB0:

- 00_h
  No TestCopy function enabled
- 03_h
  Only RecvCopy function (OperationMode = 3), no SendCopy function.
- 30_h
  Only SendCopy function (OperationMode = 3), no RecvCopy function.
- 33_h
  RecvCopy function and SendCopy function (OperationMode = 3)
- FF_h
  Delete content of DB TestCopyData

Filter settings via TestCopyMonitor

Run FC TestCopy via a watch table which is available in finished form as "TestCopyMonitor" in the TD7 library.

The following settings are possible using the watch table in the DB TestCopy:

Name (symbol)	Permitted values (hex.)	Meaning
OperationMode	00	Mode 0: Function blocked
	11	Mode 1: Frame entry always as of the start of DB TestCopyData
	22	Mode 2: Write to DB TestCopyData endlessly as circulating buffer
	02 20	• Write for SendCopy function • Write for RecvCopy function
	33	Mode 3: Fill DB TestCopyData once, then set mode 0.
	03 30	• Fill for SendCopy function • Fill for RecvCopy function
	FF	Delete the entire DB TestCopyData and reset to default

As the output value, FC TestCopy returns a count value in DBW12 (Recv_TgramCounter) of the DB TestCopyData for the frames received since setting mode 1, 2 or 3 that match the filter criteria.

The number of send frames is entered in DBW26 (Sendcv_TgramCounter).

In DBB28, you receive a return value that indicates the errors that occurred during processing of the FC:

- FC_RetVal = 0
  No error
- FC_RetVal = 1
  The specified DB TestCopyData is too short.
- FC_RetVal = 10d
  The mode entered in DBB0 is not defined.

In addition, the following special filters are available for the RecvCopy or SendCopy function:

Name (symbol) *	Permitted values (hex.)	Meaning
Xxxx_TgrmType	FF	Copy all frame types (TA) to DB TestCopyData
	00	Copy only spontaneous Org. frames (TA = 0)
	11	Copy only queried Org. frames (TA = 1).
	22	Copy only spontaneous data frames (TA = 2).
	33	Copy only queried data frames (TA = 3).
	01	Copy Org. frames with TA = 0 or 1
	23	Copy data frames with TA = 2 or 3
	Any combination	Copy any 0, 1, 2, 3 combinations.
Xxxx_DestSubscr	All permitted subscribers	Filter for the dest. subscriber no. in the frame
	-1	Copy all frames regardless of the dest. subscriber no.
Xxxx_DestObject	All permitted objects	Filter for the dest. object no. in the data frame.
	-1	Copy all frames regardless of the dest. object no.
Xxxx_SourceSubscr	All permitted subscribers	Filter for the source subscriber no. in the frame
	-1	Copy all frames regardless of the source subscriber no.
Xxxx_SourceObject	All permitted objects	Filter for the source object no. in the data frame
	-1	Copy all data frames regardless of the source object no.
Xxxx_StartIndex	All permitted indexes	Filter for the start index no. in the data frame
	-1	Copy all data frames regardless of the start index no.

* "Xxxx": Placeholder for "Recv" or "Send"

The desired filter tags can be easily added using the tag names of the TestCopy DB.

Notes on operator input

When changing from one mode to the next, the content of DB TestCopyData is not deleted. Only internal pointers and frame counters in the management area of the DB TestCopyData are reset. When there is a mode change it is therefore recommended to use the delete function "FF" to preset the frame buffer area with 0. The makes the copied frame blocks easier to read.

If send and receive frames are to be copied, in the left half byte of the "OperationMode" parameter, the same mode must be entered as in the right half byte.

The following scheme applies to mode 0, 1, 2 and 3:

- If the buffer is deleted, FF_h must always be entered.
- Separate deletion of the receive and send frames is not possible.

FC TimeTask

Function

The FC TimeTask keeps a continuous date and time on a CPU The FC has no parameters.

Link the FC into the cyclic user program (in OB1) following the FC BasicTask.

FC TimeTask can only be used when the CPU is synchronized by a local TIM. Activate the time-of-day synchronization for the relevant TIM modules.

After the CPU has started up, the TIM supplies the date and current time the first time with an Org. frame. After this, time-of-day synchronization is performed at the interval specified in the configuration of the TIM. For time-of-day synchronization on MPI/partyline an interval of one minute is recommended. FC TimeTask sets the clock of the CPU with the time provided by the TIM.

The FC reads out the time in every OB1 cycle. The read time is entered in the first two double words of the DB BasicData and marked as valid or invalid and with an indication whether it is daylight saving or standard time.

From DB BasicData all blocks take the current time if they need it. For example the data point typicals do this to time stamp their data or FC Trigger to check whether a time set for the FC has been reached or a preset interval has elapsed. This time of day is also available to the user program.

Table 1-149 Assignment of the data words with date, time and time status

CurrentDate	Data byte 0	Year * 10	Year * 1
	Data byte 1	Month * 10	Month * 1
	Data byte 2	Day * 10	Day * 1
	Data byte 3	Hour * 10	Hour * 1
CurrentTime	Data byte 4	Minute * 10	Minute * 1
	Data byte 5	Second * 10	Second * 1
	Data byte 6	Milliseconds * 100	Milliseconds * 10
	Data byte 7	Milliseconds * 1	Time status

Table 1-150 Assignment of the half byte "time status"

0	0 = date/time invalid 1 = date/time valid
1	0 = standard time 1 = daylight saving time
2	(not used)
3	(not used)

Apart from the time status, whether or not the date/time is valid can also be determined based on data bit 16.1 "CpuClockOk". As soon as the time on the CPU is valid, this bit is set to 1 by FC TimeTask. In the user program this bit can be queried directly under the symbolic name "BasicData.CpuClockOk".

FC Trigger

Function

The FC sets an output (memory bit, data bit or digital output) at a time that can be configured by the user or at a preset time interval.

The FC resets this output after one OB1 cycle.

The FC can be called at any point in the cyclic user program (OB1) also more than once.

If the running of a program section or a software function is to be triggered using FC Trigger we recommend that FC Trigger is called directly before execution of this function. Applications for triggering functions due to the memory bit set by FC Trigger are, for example, as follows:

- Running through a function
- Calling a block
- Triggering transfer of a counted value every 2 hours

If several functions need to be activated at the same time this can be implemented by one FC Trigger block if all functions query the same memory bit set by the FC. This works, however, only when the triggered functions do not reset this memory bit themselves.

Remedy if triggered blocks reset the memory bit:

- You call FC Trigger often, always with the same time but a different output memory bit.
- After calling FC Trigger you reproduce the set output memory bit in an appropriate number of further memory bits.

The FC accesses the SINAUT time of day in the first two data double words of the DB BasicData. These are constantly supplied if an FC TimeTask is included in the user program and this is synchronized at regular intervals by a local TIM. FC Trigger compares the time set for it with the current time only when the time in DB BasicData, data byte 7 (time status byte, bit 0 = 1) is marked as being valid.

The accuracy with which FC Trigger operates depends on the accuracy of the time and on the OB1 cycle time.

If the OB1 cycle time is lower than 1 second (this is normal) the output is set exactly at the programmed second value with the inaccuracy of the OB1 cycle time of less than 1 s.

If the OB1 cycle time is higher than 1 seconds, the FC works with a tolerance of 4 seconds. If the FC is processed too late but still within 4 seconds of the configured time, the output is still set.

The edge memory bit "Flag" configured for the FC is set at the same time as the output and reset 5 seconds after the configured time.

No placeholder parameter may be used for the edge memory bit and it must not be reset by the user program.

You can find examples of the parameter assignment for FC Trigger further below.

Parameter

Name:	**IntervalMode**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Explanation	Point in time / time interval • FALSE = point in time • TRUE = time interval You will find examples of the parameter assignment for a time or time interval after this explanation of the parameters.

Name:	**Hour_Minute**
Declaration:	INPUT
Data type:	WORD
Explanation	Specifies the values for hours and minutes. Further explanation: Refer to the parameter "Month_Year".

Name:	**Second_Day**
Declaration:	INPUT
Data type:	WORD
Explanation	Specifies the values for seconds and day. Further explanation: Refer to the parameter "Month_Year".

Name:	**Month_Year**
Declaration:	INPUT
Data type:	WORD
Range of values:	• 00 ... 99 • FF
Explanation	Specifies the values for month and year Each parameter is in two parts. Each of the two values per parameter is specified with two digits as a BCD-coded value. • The first two digits specify the value for hours, seconds or month. • The two other digits specify the value for minutes, day or year. For parts of the parameter that are not needed FF is entered. Which parameters are permitted depends on the particular parameter and on the "IntervalMode" parameter. You will find further information following the explanation of the parameters.

Name:	**TriggerOutput**	
Declaration:	OUTPUT	
Data type:	BOOL	
Range of values:	Output	Q 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation	Trigger output The output is set for the duration of one OB1 cycle when the time or time interval set for "Hour" to "Year" is reached.	

Name:	**Flag**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	Memory bit	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter and this should not be used.	
Explanation	Edge memory bit for the "TriggerOutput" output. No placeholder memory bit may be specified! The edge memory bit must not be reset by the user program.	

Examples of the parameter assignment of a time or time interval

IntervalMode = FALSE (or 0)

The FC operates according the time principle. When the time set is reached, the output "TriggerOutput" is set for one OB1 cycle.

All time parameters can be used to specify the point in time. Parameters not required should be assigned FF. They are ignored during the check for reaching the time.

Even when "IntervalMode" = 0, time intervals can be set in certain ranges, see the following examples.

Permitted values for the time parameters:

Hours	00-23	Day	01-31
Minutes	00-59	Month	01-12
Seconds	00-59	Year	00-99

Examples:

- IntervalMode = FALSE
  The output "TriggerOutput" is set once on 04.02.91 at 06:45:12:
  - Hour_Minute : W#16#0645
  - Second_Day : W#16#1204
  - Month_Year : W#16#0291
- IntervalMode = FALSE
  The output "TriggerOutput" is set every day at 06:00:00:
  - Hour_Minute : W#16#0600
  - Second_Day : W#16#00FF
  - Month_Year : W#16#FFFF
- IntervalMode = FALSE
  The output "TriggerOutput "is set on the 1st of every month at 06:00:00:
  - Hour_Minute : W#16#0600
  - Second_Day : W#16#0001
  - Month_Year : W#16#FFFF
- IntervalMode = FALSE
  The output "TriggerOutput" is set every year on October 1, at 06:00:00:
  - Hour_Minute : W#16#0600
  - Second_Day : W#16#0001
  - Month_Year : W#16#10FF

IntervalMode = TRUE (or 1)

The FC operates according the time interval principle. When the time value set or a multiple of it is reached, the output "TriggerOutput" is set for one OB1 cycle.

Only the specifications for hours, minutes and seconds are relevant. The date parameters are ignored. A time interval can also only be set in hours or in minutes or in seconds. Time parameters not required should be assigned FF.

The following time intervals are permitted:

- Hours: 01, 02, 03, 04, 06, 08, 12, 24
- Minutes: 01, 02, 03, 04, 05, 06, 10, 12, 15, 20, 30, 60
- Seconds: 10, 12, 15, 20, 30, 60

Examples:

- IntervalMode : TRUE
  The output "TriggerOutput" is set as follows:
  - Hour_Minute : W#16#06FF (every 6 hours)
  - Second_Day : W#16#FFFF (at 00:00:00, 06:00:00, 12:00:00 o'clock and ...)
  - Month_Year : W#16#FFFF (... at 18:00:00 o'clock)
- IntervalMode : TRUE
  The output "TriggerOutput" is set as follows:
  - Hour_Minute : W#16#FF30
  - Second_Day : W#16#FFFF (at 00:00:00, 00:30:00, 01:00:00 o'clock and ...)
  - Month_Year : W#16#FFFF (... at 01:30:00, 02:00:00, 02:30:00 o'clock etc.)

Error message during startup

The FC checks the parameters Hour_Minute, Second_Day und Month_Year in every cycle to ensure that they keep to the permitted range of values. What is permitted is also dependent on the "IntervalMode" parameter.

If the parameter assignment is incorrect, an error message is entered in the diagnostics buffer (event ID B113) only during startup. The CPU does not change to STOP. Afterwards, the FC checks the parameters without outputting error messages until the error has been eliminated.

The diagnostics message provides a precise identification of the incorrect parameter (continuous number of the parameter, i.e. 2, 3 or 4). The causes of the diagnostics message can be depend on the parameter "IntervalMode".

IntervalMode = FALSE (or 0)

The permitted ranges of values for the parameters Hours, Minutes, Seconds, Day, Month and Year were not kept to. Apart from FF, the following can be configured:

Hours	00-23	Day	01-31
Minutes	00-59	Month	01-12
Seconds	00-59	Year	00-99

"IntervalMode" = TRUE (or 1)

In this case the error can have two different causes:

- The permitted ranges of values for the parameters Hours, Minutes and Seconds were not kept to. Apart from FF, the following can be configured:
  - Hours: 01, 02, 03, 04, 06, 08, 12, 24
  - Minutes: 01, 02, 03, 04, 05, 06, 10, 12, 15, 20, 30, 60
  - Seconds: 10, 12, 15, 20, 30, 60
- A time interval can only be set in hours or in minutes or in seconds. The two unused parameters must have FF written to them. Even if FF was entered for all three named parameters, an error exists.

System blocks

BasicTask_* FC

Block versions

The block is available in the following versions for the different CPU types:

- **BasicTask_B**
  For S7-1500 CPU
  The block for the S7-1500 CPU cannot be used for S7-400 CPU.
- **BasicTask_B**
  For S7-400 CPU
  The block for the S7-400 CPU cannot be used for S7-1500 CPU.
- **BasicTask_X**
  For S7-300 PU with partyline
- **BasicTask_P**
  For S7-300 CPU without partyline

For information on "Partyline", see Glossary.

The tasks of the various "BasicTask_*" FCs for handling basic communication tasks are the same in the respective CPU type. The only parameter of the FC is identical in each case.

When the TD7onCPU is generated, the correct FC is automatically created in the respective CPU.

Function

The block is required in every CPU. It handles the following tasks:

- Central tasks during startup
- The processing of all communication mailboxes
- Central organizational tasks such as starting, monitoring and answering general requests.

Call FC BasicTask as the first block in OB1.

Parameter

Name:	**UserFC**
Declaration:	INPUT
Data type:	INT

Range of values:
- 1 ... 32000
  Number of the FC
  The maximum possible number depends on the CPU.
- 0
  Substitute value if there is no FC present for the specified purpose.

Explanation: Number of a user FC for specific further processing of the received data.

If an FC is specified, this FC is called automatically by the user program with all received data. At the time of the call, the receive frame is still in the receive mailbox of the communication DB.

The program in the user FC can read the receive frame from the receive mailbox and process it further in any way required, e.g. write the received data to an intermediate buffer.

Using the user program, the required information on the communication DB can be read from the DB BasicData via the "CurrentComDB" tag in DW60. CurrentComDB contains the following information:

- S7-1500
  Index of the current BConnection instance in the DB BConnectData
- S7-300/400
  Number of the current communication DB

In the opened communication DB, the start of the current receive frame in the receive mailbox can be found via the "CurrentReceivedMessage" tag in DW10.

DB BasicData

This data block provides the central data management. It contains information that must be maintained centrally for all blocks. Among other things, the data block contains the subscriber records and the connection descriptions.

DB BasicData is automatically generated in the required length preset with the subscriber and connection specific data and then saved in the block directory of the CPU.

DB BasicData exists once on every CPU.

Note

Number of DB BasicData

In the TD7 library, DB BasicData has the number DB127 and is also saved there under this number when the DB is generated for the various CPUs In principle it would be possible to change the number, but this involves a lot of work and can cause errors in the further creation of the user program.

It is therefore recommended to keep the DB number 127 free for the DB BasicData.

FC Create

Auxiliary block for putting together the data to be sent and its entry in the relevant instances of the send mailbox(es) (SendBuffer array) in the respective communication DB. These are the following DBs:

- **S7-1500**
  DB BConnectData > BConnection[n]
- **S7-400**
  DB BComDataXX
- **S7-300**
  - DB XComDataXX
    or
  - DB PComDataXX

XX is a placeholder for the sequential number (01, 02, etc.).

FC Create is required by the data point typicals for data and organizational frames and by FC BasicTask for organizational frames only.

Diagnose / Diagnostics FC

Name of the block:

- For S7-300: FC Diagnose
- For S7-1500: FC Diagnosis

Auxiliary block for entering SINAUT system messages in the diagnostics buffer of the CPU.

FC Distribute

Auxiliary block for distribution of the data located in the receive mailbox to the data point typicals responsible or to the node objects in the subscriber records.

FC Search

Auxiliary block for the following search tasks:

- Search for the initial address of a subscriber object within the subscriber records
- Search for the local object no. (Instance DB) from one of the two object reference lists for a received message with an incomplete destination address

The auxiliary block is required by almost all blocks.

FC Startup

The block is required in every CPU. It must be linked into the startup program OB100.

The block has the task of setting the startup memory bit in the DB BasicData and to reset the corresponding edge memory bit if this is still set.

The block has no parameters.

System blocks - BCom

FB BCom

Validity

- S7-1500
- S7-400

The block for the S7-400 CPU cannot be used for S7-1500 CPU and vice versa. The tasks of the two FCs are the same.

Function

Auxiliary block for FC BasicTask_B for processing a communication mailbox of the type DB BConnectData (S7-1500) or DB BComData (S7-400).

Via the DB, a configured PBK (programmed block communication) connection is handled using the SFBs "BSEND" and "BRCV".

FB BCom also ensures that received frames are distributed immediately to the receive objects responsible in the CPU. To do this FB BCom calls FC Distribute as an auxiliary block.

DB BComData

Validity

S7-400

Function

Instance data block for the communication block FB BCom. The instance DB represents the communication mailbox and contains the following, among other things:

- The receive mailbox (ReceiveBuffer)
- A send mailbox (SendBuffer)

It also contains central data required to control and manage the PBK connection that runs via this mailbox.

The data block is required in every CPU in which FB BCom is used. If the CPU has several PBK connections the DB is required more than once.

DB BComData is automatically generated in the required length preset with the connection specific data and then saved in the block directory of the CPU.

BConnect FB

Validity

S7-1500

Function

Auxiliary block for FC BasicTask for processing connections to the communication partners.

DB BConnectData

Validity

S7-1500

Function

Instance data block for the communication block FB BCom. The instance DB represents the communication mailbox and contains the following, among other things:

- The receive mailbox (ReceiveBuffer)
- A send mailbox (SendBuffer)

It also contains central data required to control and manage the PBK connection that runs via this mailbox.

The data block is required in every CPU in which FB BCom is used. If the CPU has several PBK connections the DB is required more than once.

DB BComData is automatically generated in the required length preset with the connection specific data and then saved in the block directory of the CPU.

System blocks - PCom

FB PCom

Validity

S7-300 CPU without partyline

FB PCom is used only with communication via the P bus. This affects the communication between a TIM and a CPU with P bus.

Function

Auxiliary block for FC BasicTask_P for processing a communication mailbox of the type DB PComData using the SFCs WR_REC and RD_REC.

Received frames are also distributed immediately to the receive objects responsible in the CPU. To do this FB PCom calls FC Distribute as an auxiliary block.

DB PComData

Instance data block for the communication block FB PCom. The instance DB makes the communication mailbox available and contains the following:

- The receive mailbox (ReceiveBuffer)
- A send mailbox (SendBuffer)

It also contains central data required to control and manage the connection that runs via this mailbox.

The data block is required in every CPU in which FB PCom is used. If the CPU has several corresponding connections the DB is required more than once.

System blocks - XCom

FB XCom

Validity

S7-300 CPU with partyline

Function

Auxiliary block for FC BasicTask_X for processing a communication mailbox of the type DB XComData.

Via DB BComData a configured connection (PBK connection) is handled using the SFCs "X_SEND" und "X_RCV".

The FB XCom also ensures that received frames are distributed immediately to the receiving objects responsible in the CPU. To do this the FB XCom calls the FC Distribute as an auxiliary block.

DB XComData

Instance data block for the communication block FB XCom. The instance DB makes the communication mailbox available and contains the following:

- The receive mailbox (ReceiveBuffer)
- A send mailbox (SendBuffer)

It also contains central data required to control and manage the X connection that runs via this mailbox.

The data block is required in every CPU in which FB XCom is used. If the CPU has several X connections the DB is required more than once.

DB XComData is automatically generated in the required length preset with the connection specific data and then saved in the block directory of the CPU.

1.4.8 SCALANCE X, W and M

1.4.8.1 Configuring SCALANCE X / W / M

Legal notice

Qualified personnel

The product/system belonging to this documentation may only be handled by qualified personnel for the intended purpose taking into account the documentation relating to the intended purpose and, in particular, the safety and warning notices it contains. Due to training and experience, **qualified personnel** is capable of recognizing risks and avoiding possible dangers when handling these products/systems.

Note on downloading configuration files

The password preset in the factory is not accepted for a local download to the SCALANCE device.

- Change the password preset in the factory via WBM or CLI.

Configuring SCALANCE X

System functions

Availability of the system functions

The following table shows the availability of the system functions on the IE switches. Note that all functions are described in the online help. Depending on your IE switch, some functions are not available.

We reserve the right to make technical changes.

		SCA-LANCE XB-200	SCA-LANCE XC-200	SCA-LANCE XP-200	SCA-LANCE XF-200BA	SCA-LANCE XR-300 WG	SCA-LANCE XM-400/ SCA-LANCE XR-500
Information	ARP table	✓	✓	✓	✓	✓	✓
	Log table	✓	✓	✓	✓	✓	✓
	Ethernet Statistics	✓	✓	✓	✓	✓	✓
	Diagnostics (temperature)	-	✓	✓	✓	✓	✓
System functions	DNS	-	-	-	-	-	✓
	SMTP Client	✓	✓	✓	✓	✓	✓
	DHCP Client	✓	✓	✓	✓	✓	✓
	DHCP Server	✓ 2)	✓	✓	✓	✓ 2)	✓
	SNMP	✓	✓	✓	✓	✓	✓
	Manual time setting	✓	✓	✓	✓	✓	✓
	DST	-	✓	✓	✓	-	✓
	SNTP	✓	✓	✓	✓	✓	✓
	NTP	✓	✓	✓	✓	✓	✓
	SIMATIC Time Client	✓	✓	✓	✓	✓	✓
	PTP	-	-	-	-	-	✓ 1)
	Auto logout	✓	✓	✓	✓	✓	✓
	Syslog Client	✓	✓	✓	✓	✓	✓
	Fault Monitoring	✓	✓	✓	✓	✓	✓
	PROFINET	✓	✓	✓	✓	✓	✓
	EtherNet/IP	✓	✓	✓	✓ 4)	✓	-
	PLUG	-	✓	✓	✓	-	✓
	Port Diagnostics	✓	✓	✓	✓	✓	✓

		SCA-LANCE XB-200	SCA-LANCE XC-200	SCA-LANCE XP-200	SCA-LANCE XF-200BA	SCA-LANCE XR-300 WG	SCA-LANCE XM-400/ SCA-LANCE XR-500
Layer 2 functions	Sending priorities	-	✓	✓	✓	-	-
	CoS Map	✓	✓	✓	✓	✓	✓
	DSCP Mapping	✓	✓	✓	✓	✓	✓
	QoS Trust	✓	✓	✓	✓	✓	✓
	CoS Port Remap	-	✓	✓	✓	-	-
	Rate Control	✓	✓	✓	✓	✓	✓
	GVRP	-	✓	✓	✓	-	✓
	Base Bridge Mode (VLAN)	✓	✓	✓	✓	✓	-
	Port Based VLAN	✓	✓	✓	✓ 4)	✓	✓
	Private VLAN	-	✓	✓	-	-	✓
	Provider Bridge (Bridge Mode)	-	✓	✓	✓	-	✓
	Switch-Port VLAN Trunk	-	✓	✓	✓ 4)	-	✓
	Protocol Based VLAN Group	-	-	-	-	-	✓
	Protocol Based VLAN Port	-	-	-	-	-	✓
	IPv4 Subnet Based VLAN	-	-	-	-	-	✓
	Port Based Mirroring	✓	✓	✓	✓	✓	✓
	RSPAN	-	-	-	-	-	✓
	VLAN-based mirroring	-	-	-	-	-	✓
	MAC flow mirroring	-	-	-	-	-	✓
	IP flow mirroring	-	-	-	-	-	✓
	Dynamic MAC Aging	✓	✓	✓	✓	✓	✓
	Ring redundancy	✓	✓	✓	✓	✓	✓
	MRP Interconnection	-	-	-	-	-	✓
	H-Sync support	-	✓	✓	✓ 4)	-	-
	S2 devices	-	✓	✓	✓ 4)	-	-
	CiR/H-CiR support	-	✓	✓	✓ 4)	-	-
	Standby (HRP)	✓	✓	✓	✓ 4)	✓	✓
	Observer (HRP)	-	✓	✓	✓	-	✓
	Link Check	✓	✓	-	✓ 4)	✓	✓
	Spanning Tree	✓	✓	✓	✓	✓	✓
	RSTP	✓	✓	✓	✓	✓	✓
	MSTP	-	✓	✓	-	-	✓

		SCA-LANCE XB-200	SCA-LANCE XC-200	SCA-LANCE XP-200	SCA-LANCE XF-200BA	SCA-LANCE XR-300 WG	SCA-LANCE XM-400/ SCA-LANCE XR-500
	Enhanced Passive Listening Compatibility	✓	✓	✓	✓	✓	✓
	Ring with RSTP	-	✓	✓	✓	-	✓
	RSTP+	-	-	-	-	-	✓
	Loop Detection	✓	✓	✓	✓	✓	✓
	Link Aggregation	-	✓	✓	✓ 4)	-	✓
	DCP Forwarding	✓	✓	✓	✓	✓	✓
	LLDP	✓	✓	✓	✓	✓	✓
	Unicast filter	✓	✓	✓	✓	✓	✓
	Locked Ports	✓	✓	✓	✓	✓	✓
	Unicast learning	✓	✓	✓	✓	✓	✓
	Unicast blocking	✓	✓	✓	✓	✓	✓
	Multicast groups	✓	✓	✓	✓	✓	✓
	IGMP	✓	✓	✓	✓	✓	✓
	GMRP	-	✓	✓	✓	-	✓
	Multicast blocking	✓	✓	✓	✓	✓	✓
	Broadcast blocking	✓	✓	✓	✓	✓	✓
	RMON	✓	✓	✓	✓	✓	✓
	RMON history	-	✓	✓	✓	✓	✓
Layer 3 functions	NAT/NAPT	-	✓	✓	-	-	✓ 3)
	Static routes	-	-	-	-	-	✓ 3)
	Route maps	-	-	-	-	-	✓ 3)
	DHCP relay agent	✓	✓	✓	✓	✓	✓ 3)
	DHCPv6 relay agent	-	-	-	-	-	✓ 3)
	VRRP	-	-	-	-	-	✓ 3)
	VRRPv3	-	-	-	-	-	✓ 3)
	OSPFv2	-	-	-	-	-	✓ 3)
	OSPFv3	-	-	-	-	-	✓ 3)
	RIPv2	-	-	-	-	-	✓ 3)
	RIPng	-	-	-	-	-	✓ 3)
	IGMP	-	-	-	-	-	✓ 3)
	PIM	-	-	-	-	-	✓ 3)
	MSDP	-	-	-	-	-	✓ 3)
	Single Hop Inter-VLAN Routing	-	✓	✓	-	-	-
	Common agent address	-	✓	✓	-	-	-

		SCALANCE XB-200	SCALANCE XC-200	SCALANCE XP-200	SCALANCE XF-200BA	SCALANCE XR-300WG	SCALANCE XM-400/ SCALANCE XR-500
Security functions	Roles	-	-	-	-	-	✓
	Groups	-	-	-	-	-	✓
	Passwords	✓	✓	✓	✓	✓	✓
	RADIUS authentication	✓	✓	✓	✓	✓	✓
	MAC authentication	-	✓	✓	✓	-	✓
	Guest VLAN	-	✓	✓	✓	-	✓
	802.1X reauthentication	✓	✓	✓	✓	✓	✓
	MAC ACL	-	-	-	-	-	✓
	IP ACL	-	-	-	-	-	✓
	Management ACL	✓	✓	✓	✓	✓	✓

[1] Only SCALANCE XR528-6M and SCALANCE XR552-12M as of firmware version 5.0

[2] Restricted

[3] Integrated or using a KEY-PLUG

[4] Not with DNA devices

Availability of hardware

The following table shows the hardware of the IE switches.

We reserve the right to make technical changes.

	SCALANCE XB-200	SCALANCE XC-200	SCALANCE XP-200	SCALANCE XF-200BA	SCALANCE XR-300WG	SCALANCE XM-400/ SCALANCE XR-500
Modularity	-	-	-	-	-	✓ [1]
C-PLUG	-	✓	✓	✓	-	✓
KEY-PLUG	-	-	-	-	-	✓
Combo ports	-	-	-	-	✓	✓
Pluggable transceiver slots	-	✓	-	-	-	✓
SELECT/SET button	-	✓	✓	-	-	✓
RESET button	✓	-	✓	-	✓	-
Signaling contact	-	✓	✓	✓	-	✓
Serial interface	✓	✓	✓	-	✓	✓
Out-of-band interface	-	-	-	-	-	✓
Display modes	-	✓	✓	-	-	✓

	SCA-LANCE XB-200	SCA-LANCE XC-200	SCA-LANCE XP-200	SCA-LANCE XF-200BA	SCA-LANCE XR-300WG	SCA-LANCE XM-400/ SCA-LANCE XR-500
NFC	-	✓	-	-	-	✓ [2]
Power over Ethernet	-	-	✓ [3]	-	-	✓

[1] SCALANCE XM-400 via extender, SCALANCE XR-500 via media modules

[2] Only SCALANCE XM-400

[3] "PoE" identifier in device name

Configuration limits

Configuration limits of the device

The following table lists the configuration limits for Web Based Management and the Command Line Interface of the devices.

The usability of various functions depends on the device type and whether or not a KEY-PLUG is inserted.

	Configurable function	Maximum number					
		SCALANCE XB-200	SCALANCE XR-300 WG	SCALANCE XC-200	SCALANCE XP-200	SCALANCE XF-200BA	SCALANCE XM-400/ SCALANCE XR-500
System	Maximum frame size (ingress)	1632 bytes	1632 bytes	1632 bytes	1632 bytes	1632 bytes	9194 Byte (configurable)
	DNS server manual (IPv4/ IPv6)	-	-	-	-	-	3
	DNS server learned (IPv4)	-	-	-	-	-	2
	DNS server in total	-	-	-	-	-	5
	Syslog server	3	3	3	3	3	3
	E-mail server	3	3	3	3	3	3
	DHCP pools	16 [1)]	28 [1)]	24	24	24	24
	IPv4 addresses per DHCP pool	1	1	24	24	24	24
	IPv4 addresses managed by the DHCP server (dynamic + static)	16 [1)]	28 [1)]	576	576	576	576
	DHCP static assignments per DHCP pool	-	-	24	24	24	24
	SNMP trap recipient	10	10	10	10	10	10
	SNTP server	1	1	1	1	1	2
	NTP server	1	1	1	1	1	3
	Agent/TIA interfaces [2)]	1	1	1	1	1	1

	Configurable function	Maximum number					
		SCALANCE XB-200	SCALANCE XR-300 WG	SCALANCE XC-200	SCALANCE XP-200	SCALANCE XF-200BA	SCALANCE XM-400/ SCALANCE XR-500
Layer 2	Virtual LANs (port-based; including VLAN 1)	17	17	257	257	257	257
	Protocol-based VLAN groups	-	-	-	-	-	12
	Protocol-based VLAN groups per port	-	-	-	-	-	12
	IPv4 subnet-based VLANs	-	-	-	-	-	150
	IPv6 subnet-based VLANs	-	-	-	-	-	150
	Private VLAN	-	-	1	1	-	1
	Primary PVLANs	-	-	1	1	-	1
	Secondary isolated PVLANs	-	-	24	24	-	24
	Secondary community PVLANs	-	-	256	256	-	256
	Mirroring sessions	1	1	1	1	1	7
	VLANs whose data traffic can be mirrored to a monitor port	-	-	-	-	-	255
	RSPAN sessions	-	-	-	-	-	1
	Standby ports	1	1	1	1	1	5
	Multiple Spanning Tree instances	-	-	4	4	-	16
	Link aggregations or Etherchannels each with a maximum of 8 ports per aggregation	-	-	8	8	4	8
	Ports in a link aggregation	-	-	8	8	4	8
	Unicast filtering	128	128	128	128	128	128
	Static MAC addresses in the FDB (Forward Database)	128	128	128	128	128	256
	Multicast addresses without active GMRP	512	512	512	512	512	512
	Multicast addresses with active GMRP	-	-	50	50	50	50
	VLANs whose data traffic can be mirrored to a monitor port	-	-	-	-	-	255

	Configurable function	Maximum number					
		SCALANCE XB-200	SCALANCE XR-300 WG	SCALANCE XC-200	SCALANCE XP-200	SCALANCE XF-200BA	SCALANCE XM-400/ SCALANCE XR-500
Layer 3	IP interfaces	1	1	24	24	1	127
	Entries in the hardware routing table	-	-	-	-	-	4096
	Static routes	-	-	-	-	-	100
	Possible routes to the same destination	-	-	-	-	-	8
	DHCP relay agent interfaces	1	1	24	24	1	127
	DHCP Relay Agent servers	4	4	4	4	4	4
	NAT interfaces	-	-	1	1	-	5
	VRRP router interfaces (only VLAN interfaces)	-	-	-	-	-	52
	OSPF areas per device	-	-	-	-	-	5
	OSPF area range entries per OSPF area (intra-area summary)	-	-	-	-	-	3
	OSPF interfaces	-	-	-	-	-	40
	OSPF interfaces per OSPF area	-	-	-	-	-	40
	OSPF virtual links (within an autonomous system)	-	-	-	-	-	8
	OSPFv3 neighbors	-	-	-	-	-	300
	OSPFv3 neighbors per interface	-	-	-	-	-	8
	OSPFv3 routes	-	-	-	-	-	1500
	OSPFv2 interface authentication key	-	-	-	-	-	200 (40 interfaces each with 5 keys)
	OSPFv2 virtual link authentication key	-	-	-	-	-	40 (8 virtual links each with 5 keys)
	PIM multicast routes per device [4)]	-	-	-	-	-	1000
	PIM components	-	-	-	-	-	1
	Rendezvous points	-	-	-	-	-	3
	Candidates for rendezvous points	-	-	-	-	-	3
	Static rendezvous points	-	-	-	-	-	3

	Configurable function	Maximum number					
		SCALANCE XB-200	SCALANCE XR-300 WG	SCALANCE XC-200	SCALANCE XP-200	SCALANCE XF-200BA	SCALANCE XM-400/ SCALANCE XR-500
Security	Roles	1	1	1	1	1	29
	Users	30	30	30	30	30	18
	Groups	-	-	-	-	-	32
	IP addresses from RADIUS servers	4	4	4	4	4	4
	Simultaneous MAC authentications (authenticated and blocked) per device [3]	4000	4000	4000	4000	4000	2000
	Simultaneous MAC authentications (authenticated and blocked) per port (configurable) [3]	100	100	100	100	100	100
	Management ACLs (access rules for management)	10	10	10	10	10	10
	Rules for port ACL MAC	-	-	-	-	-	128
	Ingress and egress rules for port ACL MAC (total)	-	-	-	-	-	364
	Rules for port ACL IP	-	-	-	-	-	128
	Ingress and egress rules for port ACL IP (total)	-	-	-	-	-	364
	Rules for VLAN ACL IP	-	-	-	-	-	128

[1] With the SCALANCE XB-200 and SCALANCE XR-300WG, the number of DHCP pools and manageable IPv4 addresses depends on the number of ports. The number of ports corresponds to the maximum number of DHCP pools and manageable IPv4 addresses.

[2] This is an IP interface.

[3] If the maximum number of MAC authentications per device is exceeded, all MAC authentications of the port at which the value was exceeded are reset.

[4] The maximum number of PIM multicast routes per device is made up as follows: SSM streams + SM streams + bidirectional streams (from IGMP joins) + bidirectional routes (from RP multicast groups) Depending on the structure of the PIM network (hierarchical structure) with bidirectional multicast a larger number of streams can be supported

Useful information

IP addresses

IPv4 / IPv6

What are the essential differences?

	IPv4	IPv6
IP configuration	• DHCP server • Manual	• Stateless Address Autoconfiguration (SLAAC): Stateless autoconfiguration using NDP (Neighbor Discovery Protocol) – Creates a link local address for every interface that does not require a router on the link. – Checks the uniqueness of the address on the link that requires no router on the link. – Specifies whether the global addresses are obtained via a status-free mechanism, a mechanism with status or via both mechanisms. (Requires a router on the link.) • Manual • DHCPv6 (status dependent)
Available IP addresses	32-bit: 4, 29 * 10^9 addresses	128-bit: 3, 4 * 10^{38} addresses
Address format	Decimal: 192.168.1.1 with port: 192.168.1.1:20	Hexadecimal: 2a00:ad80::0123 with port: [2a00:ad80::0123]:20
Loopback	127.0.0.1	::1
IP addresses of the interface	4 IP addresses	Multiple IP addresses • LLA: A link local address (formed automatically) fe80::/128 per interface • ULA: Several unique local unicast addresses per interface • GUA: Several global unicast addresses per interface
Header	• Checksum • Variable length • Fragmentation in the header • No security	• Checking at a higher layer • Fixed size • Fragmentation in the extension header
Fragmentation	Host and router	Only endpoint of the communication
Quality of service	Type of Service (ToS) for prioritization	The prioritization is specified in the header field "Traffic Class".
Types of frame	Broadcast, multicast, unicast	Multicast, unicast, anycast

	IPv4	IPv6
Identification of DHCP clients/ server	Client ID: • MAC address • DHCP client ID • System name • PROFINET station name • IAID and DUID	DUID + IAID(s) = exactly one interface of the host DUID = DHCP unique identifier Identifies server and clients uniquely and should not change, not even when replacing network components! IAID = Identity Association Identifier At least one per interface is generated by the client and remains unchanged when the DHCP client restarts Three methods of obtaining the DUID • DUID-LLT • DUID-EN • DUID-LL
DHCP	via UDP with broadcast	via UDP with unicast RFC 3315, RFC 3363 **Stateful DHCPv6** Status-dependent configuration in which the IPv6 address and the configuration settings are transferred. Four DHVPv6 messages are exchanged between client and server: 1. SOLICIT: Sent by the DHCPv6 client to localize DHCPv6 servers. 2. ADVERTISE The available DHCPv6 servers reply to this. 3. REQUEST The DHCPv6 client requests an IPv6 address and the configuration settings from the DHCPv6 server. 4. REPLY The DHCPv6 server sends the IPv6 address and the configuration settings. If the client and server support the function "Rapid commit" the procedure is shortened to two DHCPv6 messages SOLICIT and REPLY . **Stateless DHCPv6** In stateless DHCPv6, only the configuration settings are transferred. **Prefix delegation** The DHCPv6 server delegates the distribution of IPv6 prefixes to the DHCPv6 client. The DHCPv6 client is also known as PD router.
Resolution of IP addresses in hardware addresses	ARP (Address Resolution Protocol)	NDP (Neighbor Discovery Protocol)

IPv4 address

Structure of an IPv4 address

Address classes

IP address range	Max. number of networks	Max. number of hosts/network	Class	CIDR
1.x.x.x through 126.x.x.x	126	16777214	A	/8
128.0.x.x through 191.255.x.x	16383	65534	B	/16
192.0.0.x through 223.255.255.x	2097151	254	C	/24
224.0.0.0 - 239.255.255.255	Multicast applications		D	
240.0.0.0 - 255.255.255.255	Reserved for future applications		E	

An IP address consists of 4 bytes. Each byte is represented in decimal, with a dot separating it from the previous one. This results in the following structure, where XXX stands for a number between 0 and 255:

XXX.XXX.XXX.XXX

The IP address is made up of two parts, the network ID and the host ID. This allows different subnets to be created. Depending on the bytes of the IP address used as the network ID and those used for the host ID, the IP address can be assigned to a specific address class.

Subnet mask

The bits of the host ID can be used to create subnets. The leading bits represent the address of the subnet and the remaining bits the address of the host in the subnet.

A subnet is defined by the subnet mask. The structure of the subnet mask corresponds to that of an IP address. If a "1" is used at a bit position in the subnet mask, the bit belongs to the corresponding position in the IP address of the subnet address, otherwise to the address of the computer.

Example of a class B network:

The standard subnet address for class B networks is 255.255.0.0; in other words, the last two bytes are available for defining a subnet. If 16 subnets must be defined, the third byte of the subnet address must be set to 11110000 (binary notation). In this case, this results in the subnet mask 255.255.240.0.

To find out whether two IP addresses belong to the same subnet, the two IP addresses and the subnet mask are ANDed bit by bit. If both logic operations have the save result, both IP addresses belong to the same subnet, for example, 141.120.246.210 and 141.120.252.108.

Outside the local area network, the distinction between network ID and host ID is of no significance, in this case packets are delivered based on the entire IP address.

Note

In the bit representation of the subnet mask, the "ones" must be set left-justified; in other words, there must be no "zeros" between the "ones".

IPv6 addresses

IPv6 terms

Network node

A network node is a device that is connected to one or more networks via one or more interfaces.

Router

A network node that forwards IPv6 packets.

Host

A network node that represents an end point for IPv6 communication relations.

Link

A link is, according to IPv6 terminology, a direct layer 3 connection within an IPv6 network.

Neighbor

Two network nodes are called neighbors when they are located on the same link.

IPv6 interface

Physical or logical interface on which IPv6 is activated.

Path MTU

Maximum permitted packet size on a path from a sender to a recipient.

Path MTU discovery

Mechanism for determining the maximum permitted packet size along the entire path from a sender to a recipient.

LLA

Link local address FE80::/10

As soon as IPv6 is activated on the interface, a link local address is formed automatically. Can only be reached by nodes located on the same link.

ULA

Unique Local Address

Defined in RFC 4193. Via this address, the IPv6 interface can be reached in the LAN.

GUA

Global Unicast Address Via this address, the IPv6 interface can be reached, e.g. via the Internet.

Interface ID

The interface ID is formed with the EUI-64 method or manually.

EUI-64

Extended Unique Identifier (RFC 4291); method for forming the interface ID. In Ethernet, the interface ID is formed from the MAC address of the interface. Divides the MAC address into

the manufacturer-specific part (OUI) and the network-specific part (NIC) and inserts FFFE between the two parts.

Example:

MAC address = AA:BB:CC:DD:EE:FF

OUI = AA:BB:CC

NIC = DD:EE:FF

EUI-64 = OUI + **FFFE** + NIC = AA:BB:CC:**FF:FE**:DD:EE:FF

Scope

Defines the range of the IPv6 address.

Structure of an IPv6 address

IPv6 address format - notation

IPv6 addresses consist of 8 fields each with four-character hexadecimal numbers (128 bits in total). The fields are separated by a colon.

Example:

fd00:0000:0000:ffff:02d1:7d01:0000:8f21

Rules / simplifications:

- If one or more fields have the value 0, a shortened notation is possible.
  The address fd00:**0000:0000**:ffff:02d1:7d01:0000:8f21 can also be shortened and written as follows:
  fd00**::**ffff:02d1:7d01:0000:8f21
  To ensure uniqueness, this shortened form can only be used once within the entire address.
- Leading zeros within a field can be omitted.
  The address fd00:0000:0000:ffff:**02d1**:7d01:0000:8f21 can also be shortened and written as follows:
  fd00**::**ffff:**2d1**:7d01:0000:8f21
- Decimal notation with periods
  The last 2 fields or 4 bytes can be written in the normal decimal notation with periods.
  Example: The IPv6 address fd00::ffff.125.1.0.1 is equivalent to fd00::ffff:7d01:1

Structure of the IPv6 address

The IPv6 protocol distinguishes between three types of address: Unicast , anycast and multicast. The following section describes the structure of the global unicast addresses.

IPv6 prefix		Suffix
Global prefix: n bits	Subnet ID m bits	Interface ID 128 - n - m bits
Assigned address range	Description of the location, also subnet prefix or subnet	Unique assignment of the host in the network. The ID is generated from the MAC address.

The prefix for the link local address is always fe80:0000:0000:0000. The prefix is shortened and noted as follows: fe80::

IPv6 prefix

Specified in: RFC 4291

The IPv6 prefix represents the subnet identifier.

Prefixes and IPv6 addresses are specified in the same way as with the CIDR notation (Classless Inter-Domain Routing) for IPv4.

Design

IPv6 address / prefix length

Example

IPv6 address: 2001:0db8:1234::1111/48

Prefix: 2001:0db8:1234::/48

Interface ID: ::1111

Entry and appearance

The entry of IPv6 addresses is possible in the notations described above. IPv6 addresses are always shown in the hexadecimal notation.

PROFINET

PROFINET

PROFINET is an open standard (IEC 61158/61784) for industrial automation based on Industrial Ethernet. PROFINET uses existing IT standards and allows end-to-end communication from the field level to the management level as well as plant-wide engineering. PROFINET also has the following features:

- Use of TCP/IP
- Automation of applications with real-time requirements
  - Real-Time (RT) communication
  - Isochronous Real-Time (IRT) communication
- Seamless integration of fieldbus systems

You configure PROFINET in "System > PROFINET (Page 2136)".

PROFINET IO

Within the framework of PROFINET, PROFINET IO is a communications concept for implementing modular, distributed applications. PROFINET IO is implemented by the PROFINET standard for programmable controllers (IEC 61158-x-10).

EtherNet/IP

EtherNet/IP

EtherNet/IP (Ethernet/Industrial Protocol) is an open industry standard for industrial real-time Ethernet based on TCP/IP and UDP/IP. With EtherNet/IP, Ethernet is expanded by the Common Industrial Protocol (CIP) at the application layer. In EtherNet/IP, the lower layers of the OSI reference model are adopted by Ethernet with the physical, network and transport functions.

You configure EtherNet/IP in "System > EtherNet/IP (Page 2137)".

Common Industrial Protocol

The Common Industrial Protocol (CIP) is an application protocol for automation that supports transition of the field buses in Industrial Ethernet and in IP networks. This industry protocol is used by field buses/industrial networks such as DeviceNet, ControlNet and EtherNet/IP at the application layer as an interface between the deterministic fieldbus world and the automation application (controller, I/O, HMI, OPC, ...). The CIP is located above the transport layer and expands the pure transport services with communications services for automation engineering. These include services for cyclic, time-critical and event-controlled data traffic. CIP distinguishes between time-critical I/O messages (implicit messages) and individual query/response frames for configuration and data acquisition (explicit messages). CIP is object-oriented; all data "visible" from the outside is accessible in the form of objects. CIP has a common configuration basis: EDS (Electronic Data Sheet).

Electronic Data Sheet

Electronic Data Sheet (EDS) is an electronic datasheet for describing devices.

The EDS required for EtherNet/IP operation can be found in "System > Load&Save".

VLAN

VLAN

Network definition regardless of the spatial location of the nodes

VLAN (Virtual Local Area Network) divides a physical network into several logical networks that are shielded from each other. Here, devices are grouped together to form logical groups. Only nodes of the same VLAN can address each other. Since multicast and broadcast frames are only forwarded within the particular VLAN, they are also known as broadcast domains.

The particular advantage of VLANs is the reduced network load for the nodes and network segments of other VLANs.

To identify which packet is assigned to which VLAN, the frame is expanded by 4 bytes (VLAN tagging). This expansion includes not only the VLAN ID but also priority information.

Options for the VLAN assignment

There are various options for the assignment to VLANs:

- Port-based VLAN
  Each port of a device is assigned a VLAN ID. You configure port-based VLAN in "Layer 2 > VLAN > Port-based VLAN".
- Protocol-based VLAN
  Each port of a device is assigned a protocol group. You configure protocol-based VLAN in "Layer 2 > VLAN > Protocol-based VLAN port".
- IPv4 subnet-based VLAN
  The IPv4 address of the device is assigned a VLAN ID. You configure subnet-based VLAN in "Layer 2 > VLAN > IPv4 subnet-based VLAN".
- IPv6 prefix-based VLAN
  The IPv6 address of the device is assigned a VLAN ID. You configure prefix-based VLAN in "Layer 2 > VLAN > IPv6 Prefix Based VLAN".

processing the VLAN assignment

If more than one VLAN assignment is created on the device, the assignments are processed in the following order:

1. IPv4 subnet-based VLAN/IPv6 prefix-based VLAN
2. Protocol-based VLAN
3. Port-based VLAN

The frame is first examined for the IP address. If a rule on the "IPv4 subnet-based VLAN" or "IPv6 Prefix Based VLAN" tab applies, the frame is sent to the corresponding VLAN. If no rule applies, the protocol type of the frame is examined. If a rule on the "Protocol-based VLAN port" tab applies, the frame is sent to the corresponding VLAN. If no rule applies, the frame is sent via the port-based VLAN. The rules for the port-based VLAN are specified on the "Port-based VLAN" tab.

VLAN tagging

Expansion of the Ethernet frames by four bytes

For CoS (Class of Service, frame priority) and VLAN (virtual network), the IEEE 802.1Q standard defined the expansion of Ethernet frames by adding the VLAN tag.

Note

The VLAN tag increases the permitted total length of the frame from 1518 to 1522 bytes. The end nodes on the networks must be checked to find out whether they can process this length / this frame type. If this is not the case, only frames of the standard length may be sent to these nodes.

The additional 4 bytes are located in the header of the Ethernet frame between the source address and the Ethernet type / length field:

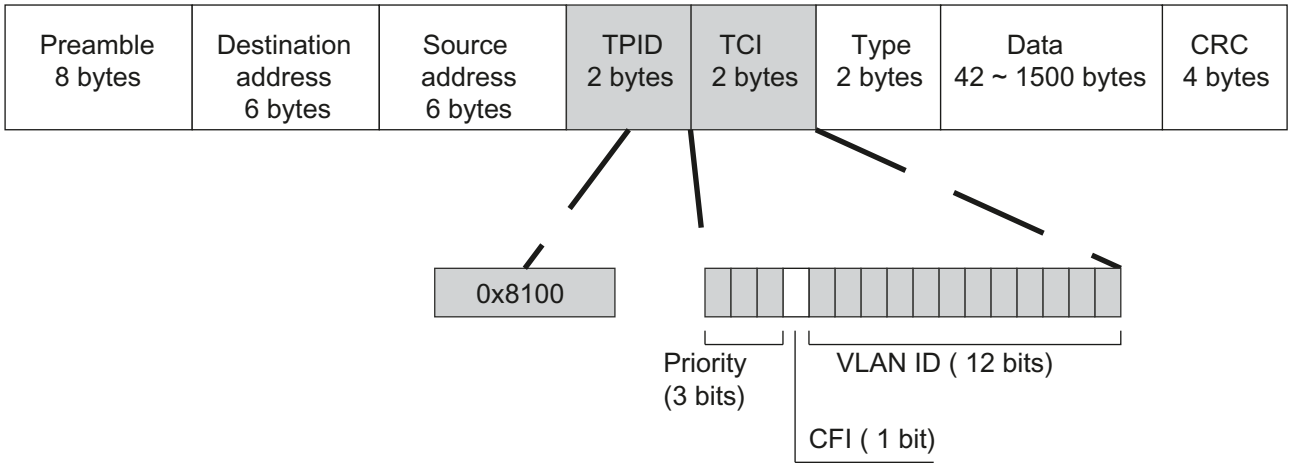

Figure 1-82 Structure of the expanded Ethernet frame

The additional bytes contain the tag protocol identifier (TPID) and the tag control information (TCI).

Tag protocol identifier (TPID)

The first 2 bytes form the Tag Protocol Identifier (TPID) and always have the value 0x8100. This value specifies that the data packet contains VLAN information or priority information.

Tag Control Information (TCI)

The 2 bytes of the Tag Control Information (TCI) contain the following information:

QoS Trust

The tagged frame has 3 bits for the priority that is also known as Class of Service (CoS), see also IEEE 802.1Q.

CoS bits	Priority	Type of the data traffic
000	0 (lowest)	Background
001	1	Best Effort
010	2	Excellent Effort
011	3	Critical Applications
100	4	Video, < 100 ms delay (latency and jitter)
101	5	Voice (language), < 10 ms delay (latency and jitter)
110	6	Internetwork Control
111	7 (highest)	Network Control

The prioritization of the data packets is possible only if there is a queue in the components in which they can buffer data packets with lower priority.

The device has multiple parallel queues in which the frames with different priorities can be processed. As default, first, the frames with the highest priority are processed. This method ensures that the frames with the highest priority are sent even if there is heavy data traffic.

Canonical Format Identifier (CFI)

The CFI is required for compatibility between Ethernet and the token Ring.
The values have the following meaning:

Value	Meaning
0	The format of the MAC address is canonical. In the canonical representation of the MAC address, the least significant bit is transferred first. Standard-setting for Ethernet switches.
1	The format of the MAC address is not canonical.

VLAN ID

In the 12-bit data field, up to 4096 VLAN IDs can be formed. The following conventions apply:

VLAN ID	Meaning
0	The frame contains only priority information (priority tagged frames) and no valid VLAN identifier.
1- 4094	Valid VLAN identifier, the frame is assigned to a VLAN and can also include priority information.
4095	Reserved

Private VLAN

With a private VLAN (PVLAN) you can divide up the layer 2 broadcast domains of a VLAN.

A private VLAN consists of the following units:

- A primary private VLAN (primary PVLAN)
  The VLAN that is divided up is called primary private VLAN.
- secondary private VLANs (secondary PVLAN)
  Secondary PVLANs exist only within a primary PVLAN. Every secondary PVLAN has a specific VLAN ID and is connected to the primary PVLAN.
  Secondary PVLANs are divided into the following types:
  - Isolated Secondary PVLAN
    Devices within an isolated secondary PVLAN cannot communicate with each other via layer 2.
  - Community Secondary PVLAN
    Devices within a community secondary PVLAN can communicate with each other directly via layer 2. The devices cannot communicate with devices in other communities of the PVLAN via layer 2.

Note

VLAN ID with secondary PVLANs

If you use the same VLAN ID for secondary PVLANs on different IE switches, the end devices in these secondary PVLANs can communicate with other via layer 2 across the different switches.

Note

Private VLAN functionality and RADIUS authentication

When VLAN assignment is activated via RADIUS authentication for one or more ports of a VLAN, you should not configure this VLAN additionally as private VLAN.

The private VLAN functionality in connection with VLAN assignment via RADIUS authentication can result in an inconsistent system state.

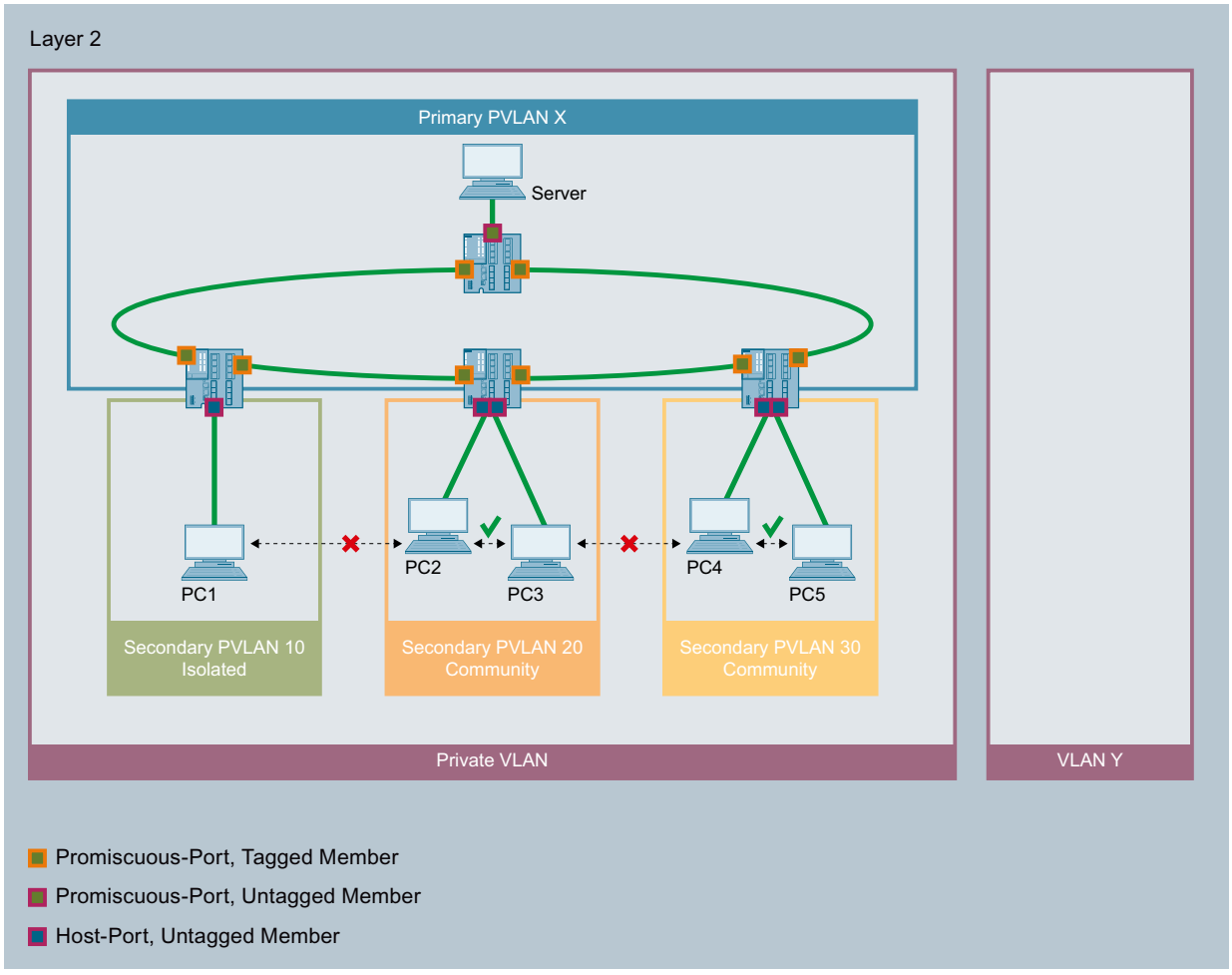

In this example, the ports of the IE switches that connect them to other IE switches are promiscuous ports. These network ports are tagged members in all PVLANs: Primary PVLAN and all secondary PVLANs.

The ports to which the PCs are connected are host ports. The host ports are all untagged members in the primary PVLAN and in their secondary PVLAN.

The port to which the server is connected is a promiscuous port. This promiscuous port ports is an untagged member in all PVLANs: Primary PVLAN and all secondary PVLANs.

In this example all PCs can communicate with the server. The server can communicate with all PCs. PC1 cannot communicate with any other PC. The PCs within a community secondary PVLAN can communicate with each other but not with the PCs in another secondary PVLAN.

VLAN tunnel

With the Q-in-Q VLAN Tunnel function it is possible to forward the data traffic from different customer networks using a VLAN tunnel via a provider network. Every customer network has the full number of possible VLANs available.

A VLAN tunnel is established between provider switches that are configured at the boundaries of a provider network. A provider switch has the following types of ports:

- Access port
  The provider switch is connected to a customer network via an access port.
  - Incoming data traffic
    The incoming data traffic at an access port is treated as if it were untagged ①. All incoming frames are expanded by a tag with the port VID of the access port ②. With frames that are already tagged, this means they are expanded by a second 802.1Q tag ③ the outer VLAN tag.

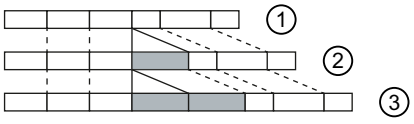

  - Outgoing data traffic
    With outgoing data traffic the outer tag is removed again at an access port.
- Core port
  The provider switch is connected to a provider network via a core port.
  Core ports are members in the port VLAN of the access port or configured with the port type "Switch-Port VLAN Trunk".

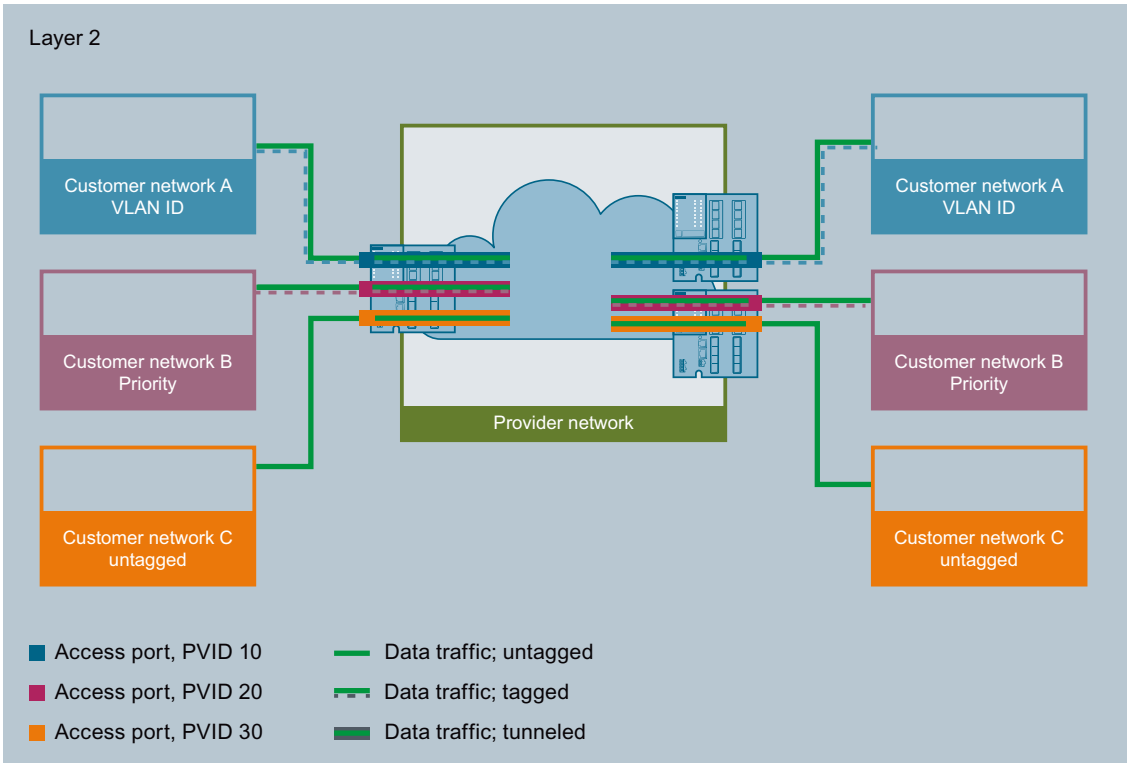

In this example the data traffic from the customer networks A, B and C is forwarded over the provider network using a VLAN tunnel. The frames from customer network A are tagged with a VLAN ID. The frames from customer network B are tagged with a priority. The frames from customer network C are untagged.

When the frames reach the relevant access port, they are expanded by a tag with the port VID of the access port and tunneled through the provider network. As soon as the frames leave the provider network, the outer VLAN tag (PVID) is removed again. The frames are forwarded in their original form. The priority of the frame is retained.

SNMP

Introduction

With the aid of the Simple Network Management Protocol (SNMP), you monitor and control network components from a central station, for example routers or switches. SNMP controls the communication between the monitored devices and the monitoring station.

Tasks of SNMP:

- Monitoring of network components
- Remote control and remote parameter assignment of network components
- Error detection and error notification

In versions v1 and v2c, SNMP has no security mechanisms. Each user in the network can access data and also change parameter assignments using suitable software.

For the simple control of access rights without security aspects, community strings are used.

The community string is transferred along with the query. If the community string is correct, the SNMP agent responds and sends the requested data. If the community string is not correct, the SNMP agent discards the query. Define different community strings for read and write permissions. The community strings are transferred in plain text.

Standard values of the community strings:

- public
  has only read permissions
- private
  has read and write permissions

Note

Because the SNMP community strings are used for access protection, do not use the standard values "public" or "private". Change these values following the initial commissioning.

Further simple protection mechanisms at the device level:

- Allowed Host
  The IP addresses of the monitoring systems are known to the monitored system.
- Read Only
  If you assign "Read Only" to a monitored device, monitoring stations can only read out data but cannot modify it.

SNMP data packets are not encrypted and can easily be read by others.

The central station is also known as the management station. An SNMP agent is installed on the devices to be monitored with which the management station exchanges data.

The management station sends data packets of the following type:

- GET
  Request for a data record from the SNMP agent
- GETNEXT
  Calls up the next data record.
- GETBULK (available as of SNMPv2c)
  Requests multiple data records at one time, for example several rows of a table.
- SET
  Contains parameter assignment data for the relevant device.

The SNMP agent sends data packets of the following type:

- RESPONSE
  The SNMP agent returns the data requested by the manager.
- TRAP
  If a certain event occurs, the SNMP agent itself sends traps.

SNMPv1/v2c/v3 use UDP (User Datagram Protocol) and use the UDP ports 161 and 162. The data is described in a Management Information Base (MIB).

SNMPv3

Compared with the previous versions SNMPv1 and SNMPv2c, SNMPv3 introduces an extensive security concept.

SNMPv3 supports:

- Fully encrypted user authentication
- Encryption of the entire data traffic
- Access control of the MIB objects at the user/group level

With the introduction of SNMPv3 you can no longer transfer user configurations to other devices without taking special action, e.g. by loading a configuration file or replacing the C-PLUG.

According to the standard, the SNMPv3 protocol uses a unique SNMP engine ID as an internal identifier for an SNMP agent. This ID must be unique in the network. It is used to authenticate access data of SNMPv3 users and to encrypt it.

Depending on whether you have enabled or disabled the “SNMPv3 User Migration” function, the SNMP engine ID is generated differently.

Restriction when using the function

Use the "SNMPv3 User Migration" function only to transfer configured SNMPv3 users to a substitute device when replacing a device.
Do not use the function to transfer configured SNMPv3 users to multiple devices. If you load a configuration with created SNMPv3 users on several devices, these devices use the same SNMP engine ID. If you use these devices in the same network, your configuration contradicts the SNMP standard.

Compatibility with predecessor products

You can only transfer SNMPv3 users to a different device if you have created the users as migratable users. To create a migratable user the "SNMPv3 User Migration" function must be activated when you create the user.

RCDP

You configure RCDP (RUGGEDCOM Discovery Protocol) on the page "System > Configuration".

If RCDP is configured accordingly on the device, you can access the device with the RUGGEDCOM EXPLORER via out-of-band interface.

RUGGEDCOM EXPLORER

With the RUGGEDCOM EXPLORER you can recognize devices in a network and make the following basic settings:

- IP Address
- Subnet Mask
- Default Gateway
- System Name
- Device Location
- System Contact

The RUGGEDCOM EXPLORER can run on every LAN-based MS Windows-PC.

You will find more information on the RUGGEDCOM EXPLORER here:

- On the Internet pages of Siemens Industry Online Support
- On the Internet pages of RUGGEDCOM products

Redundancy mode

Spanning Tree

Avoiding loops on redundant connections

The spanning tree algorithm allows network structures to be created in which there are several connections between two IE switches / bridges. Spanning tree prevents loops being formed in the network by allowing only one path and disabling the other (redundant) ports for data traffic. If there is an interruption, the data can be sent over an alternative path. The functionality of the spanning tree algorithm is based on the exchange of configuration and topology change frames.

Definition of the network topology using the configuration frames

The devices exchange configuration frames known as BPDUs (Bridge Protocol Data Units) with each other to calculate the topology. The root bridge is selected and the network topology created using these frames. BPDUs also bring about the status change of the root ports.

The root bridge is the bridge that controls the spanning tree algorithm for all involved components.

Once the root bridge has been specified, each device sets a root port. The root port is the port with the lowest path costs to the root bridge.

Response to changes in the network topology

If nodes are added to a network or drop out of the network, this can affect the optimum path selection for data packets. To be able to respond to such changes, the root bridge sends configuration messages at regular intervals. The interval between two configuration messages can be set with the "Hello Time" parameter.

Keeping configuration information up to date

With the "Max Age" parameter, you set the maximum age of configuration information. If a bridge has information that is older than the time set in "Max Age", it discards the message and initiates recalculation of the paths.

New configuration data is not used immediately by a bridge but only after the period specified in the "Forward Delay" parameter. This ensures that operation is only started with the new topology after all the bridges have the required information.

RSTP, MSTP, CIST

Rapid Spanning Tree Protocol (RSTP)

One disadvantage of STP is that if there is a disruption or a device fails, the network needs to reconfigure itself: The devices start to negotiate new paths only when the interruption occurs. This can take up to 30 seconds. Fur this reason, STP was expanded to create the "Rapid Spanning Tree Protocol" (RSTP, IEEE 802.1w). This differs from STP essentially in that the devices are already collecting information about alternative routes during normal operation and do not need to gather this information after a disruption has occurred. This means that the reconfiguration time for an RSTP controlled network can be reduced to a few seconds.
This is achieved by using the following functions:

- Edge ports (end node port)
  Edge ports are ports connected to an end device.
  A port that is defined as an edge port is activated immediately after connection establishment. If a spanning tree BPDU is received at an edge port, the port loses its role as edge port and it takes part in (R)STP again. If no further BPDU is received after a certain time has elapsed (3 x hello time), the port returns to the edge port status.
- Point-to-point (direct communication between two neighboring devices)

  By directly linking the devices, a status change (reconfiguration of the ports) can be made without any delays.

- Alternate port (substitute for the root port)

  A substitute for the root port is configured. If the connection to the root bridge is lost, the device can establish a connection over the alternate port without any delay due to reconfiguration.

- Reaction to events

  Rapid spanning tree reacts to events, for example an aborted connection, without delay. There is no waiting for timers as in spanning tree.

- Counter for the maximum bridge hops
  The number of bridge hops a package is allowed to make before it automatically becomes invalid.

In principle, therefore with rapid spanning tree, alternatives for many parameters are preconfigured and certain properties of the network structure taken into account to reduce the reconfiguration time.

Multiple Spanning Tree Protocol (MSTP)

The Multiple Spanning Tree Protocol (MSTP) is a further development of the Rapid Spanning Tree Protocol. Among other things, it provides the option of operating several RSTP instances within different VLANs or VLAN groups and, for example, making paths available within the individual VLANs that the single Rapid Spanning Tree Protocol would globally block.

Common and Internal Spanning Tree (CIST)

CIST identifies the internal instance used by the switch that is comparable in principle with an internal RSTP instance.

HRP

HRP - High Speed Redundancy Protocol

HRP is the name of a redundancy method for networks with a ring topology. The switches are interconnected via ring ports. One of the switches is configured as the redundancy manager (RM). The other switches are redundancy clients. Using test frames, the redundancy manager checks the ring to make sure it is not interrupted. The redundancy manager sends test frames via the ring ports and checks that they are received at the other ring port. The redundancy clients forward the test frames.

If the test frames of the RM no longer arrive at the other ring port due to an interruption, the RM switches through its two ring ports and informs the redundancy clients of the change immediately. The reconfiguration time after an interruption of the ring is a maximum of 300 ms.

Standby redundancy

Standby redundancy is a method with which rings each of which is protected by high-speed redundancy can be linked together redundantly. In the ring, a master/slave device pair is configured and these monitor each other via their ring ports. If a fault occurs, the data traffic is redirected from one Ethernet connection (standby port of the master or standby server) to another Ethernet connection (standby port of the slave).

Requirements

HRP

- HRP is supported in ring topologies with up to 50 devices. Exceeding this number of devices can lead to a loss of data traffic.
- For HRP, only devices that support this function can be used in the ring.
- Devices that do not support HRP must be linked to the ring using special devices with HRP capability. Up to the ring, this connection is not redundant.
- All devices must be interconnected via their ring ports. Multimode connections up to 3 km and single mode connections up to 26 km between two IE switches are possible. At greater distances, the specified reconfiguration time may be longer.
- A device in the ring must be configured as redundancy manager by selecting the "HRP Manager" setting. On all other devices in the ring, either the "HRP Client" or "Automatic Redundancy Detection" mode must be activated.
- The standby ports must be disabled in spanning tree.
- You configure HRP in Web Based Management, Command Line Interface or using SNMP.

Standby redundancy

- With standby coupling partners HRP must be set permanently.
- The ports of the standby coupling partners must be disabled in spanning tree.
- You configure standby redundancy in Web Based Management, Command Line Interface or using SNMP.

Standby

General

SCALANCE X switches support not only ring redundancy within a ring but also redundant linking of rings or open network segments (linear bus). In the redundant link, rings are connected together over Ethernet connections. This is achieved by configuring a master/slave device pair in one ring so that the devices monitor each other and, in the event of a fault, redirect the data traffic from the normally used master Ethernet connection to the substitute (slave) Ethernet connection.

Standby redundancy

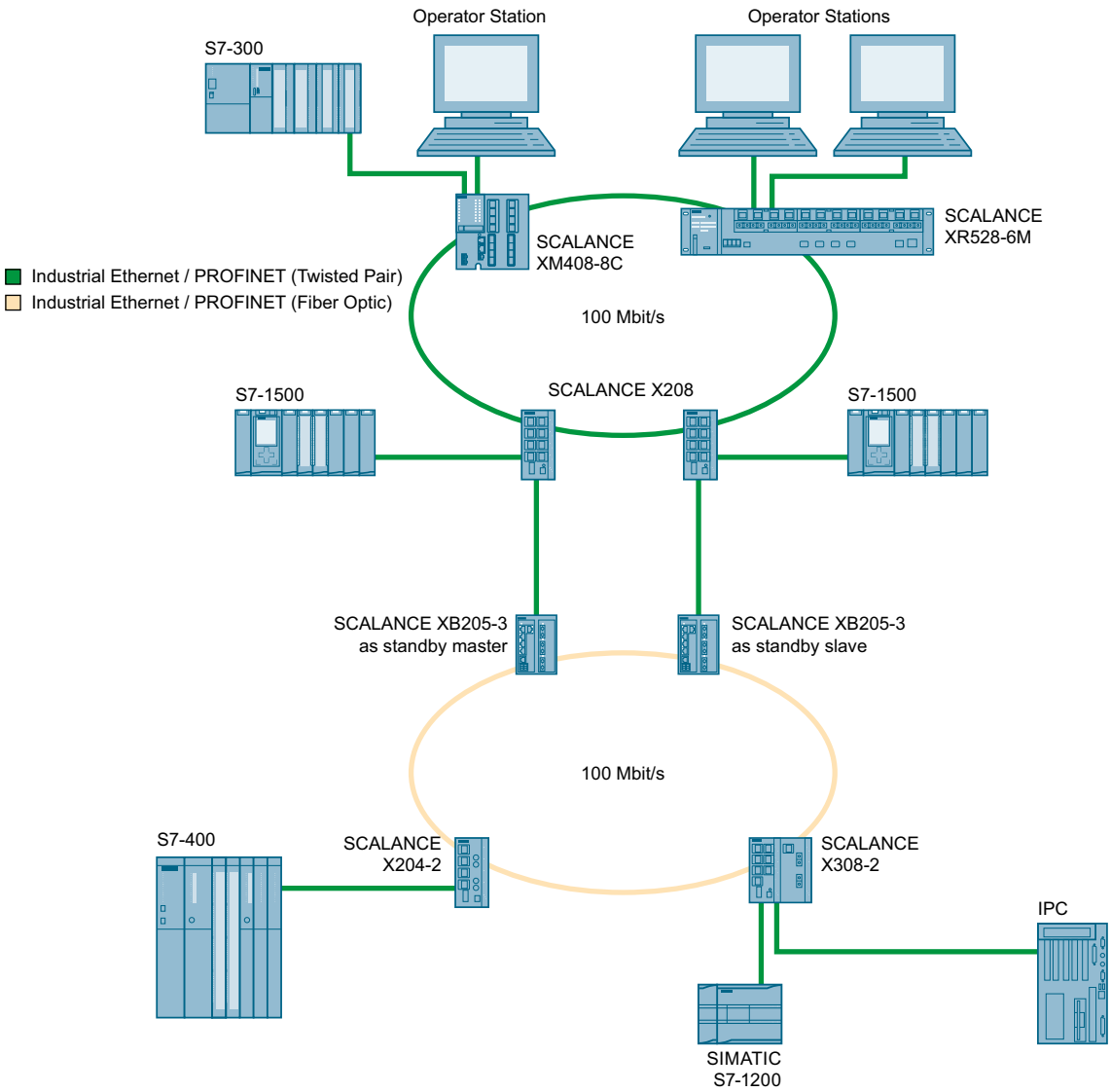

Figure 1-83 Example of a redundant link between rings

For a redundant link as shown in the figure, two devices must be configured as standby redundancy switches within a network segment. In this case, network segments are rings with a redundancy manager. Instead of rings, network segments might also be linear.

The two standby redundancy switches connected in the configuration exchange data frames with each other to synchronize their operating statuses (one device is master and the other slave). If there are no problems, only the link from the master to the other network segment is active. If this link fails (for example due to a link-down or a device failure), the slave activates its link as long as the problem persists.

Coupling of several HRP network segments

If you connect several HRP rings or links using standby redundancy, the standby master and standby slave must be located in a closed network segment. Under no circumstances may this network segment be open, i.e. a line.

Standby master and slave in a closed network segment

If the connection between the standby master and slave is interrupted, the two devices can still communicate via the redundant link of the HRP redundancy manager.

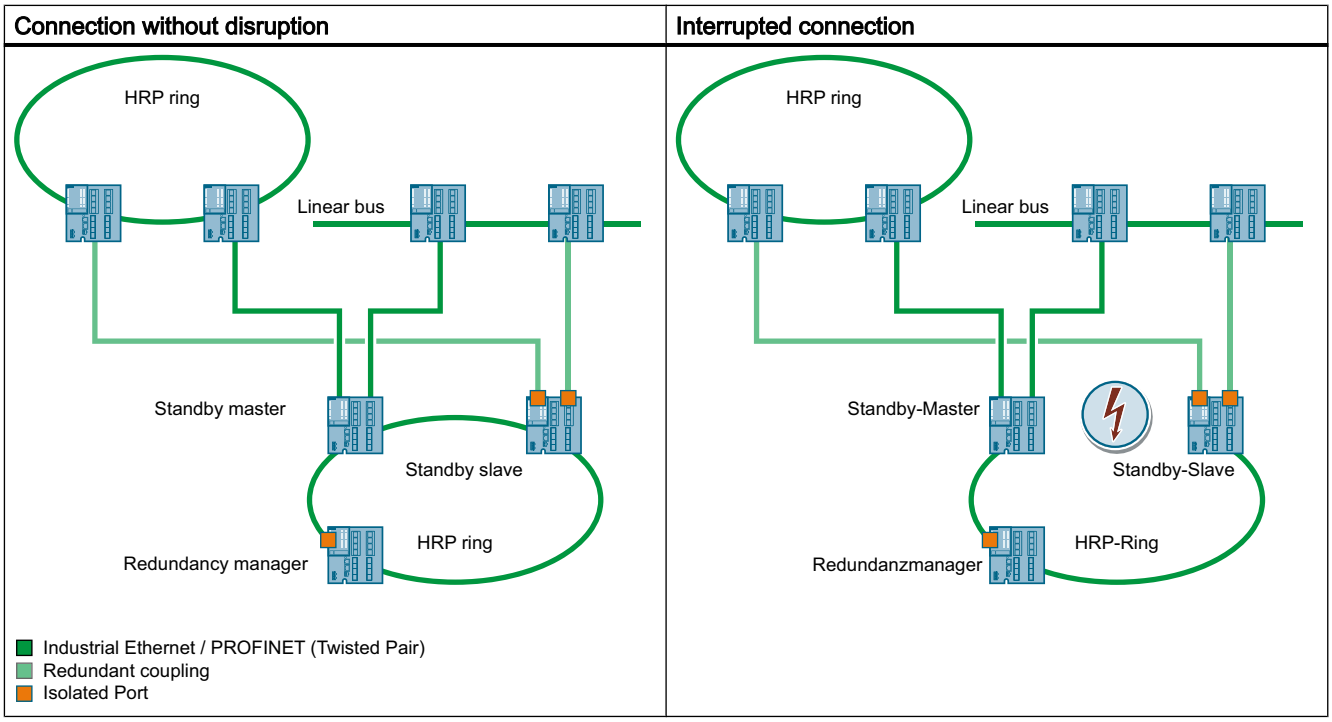

Standby master and slave in an open network segment

If the connection between the standby master and slave is interrupted, the two devices can no longer communicate. This causes a loop via the coupled network segments.

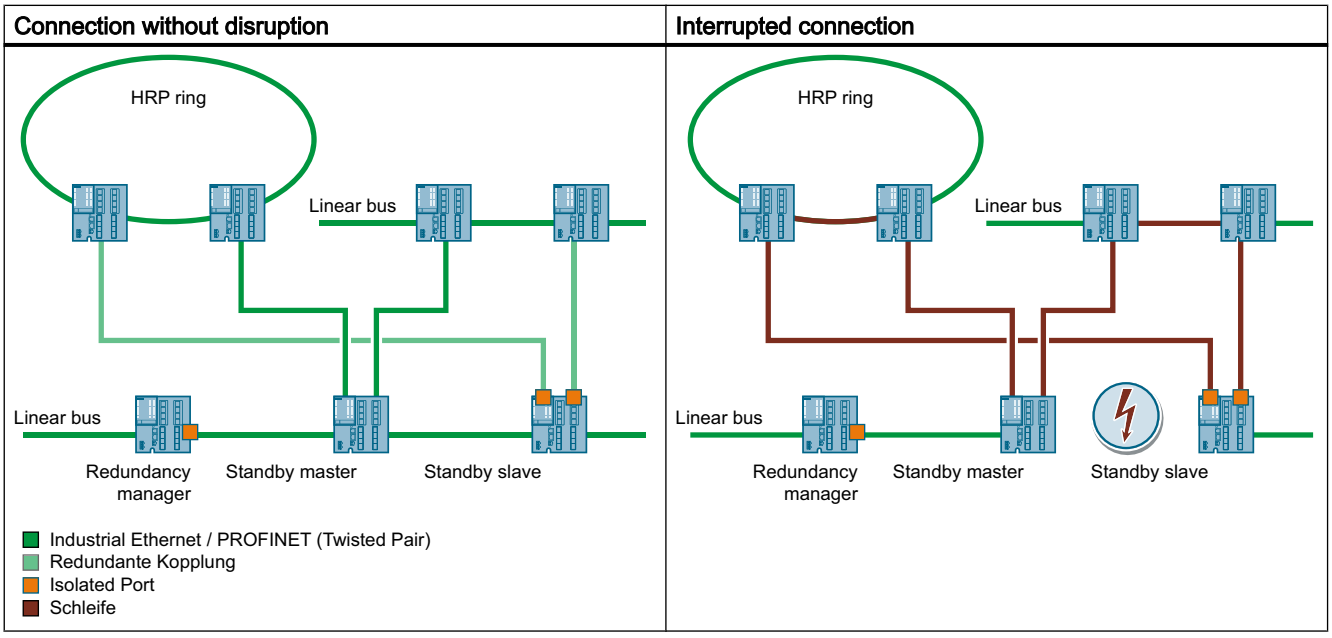

Link Check

Monitoring optical connections in the ring

On optical connections disturbances are possible in which the optical connection is not completely interrupted, but frames are lost sporadically. Such problems can, for example, be caused by defective optical cables, dirty connectors or device defects.

The redundancy manager of an HRP or MRP ring with optical connections detects a “non-recoverable ring error” with such a disturbance. The redundancy manager cannot eliminate the disturbance by closing the ring. Closing the ring in this case, would lead to circulating frames.

With the Link Check function, you can monitor the transmission quality of optical sections within an HRP or MRP ring, identify disturbed connections and under certain conditions turn them off. When the disturbed section is turned off, the redundancy manager can close the ring and restore communication.

How Link Check works

Behavior with an undisturbed connection

If you enable Link Check on two connected ring ports, the two connection partners exchange Link Check frames cyclically on these ports. The frames received by one connection partner are sent back to the other.

When the devices receive back the frames they sent from the connection partner, the connection is prepared for Link Check. The connection partners then increase the send frequency of the Link Check test frames and the actual connection monitoring is active.

Behavior with a disturbance

When connection monitoring is enabled, you can see the number of sent and received Link Check test frames on the “Information > Redundancy >Link Check” page. Based on these statistics you can recognize smaller disturbances for which the disturbance does not yet cause the transmission line to be closed down by Link Check.

Link Check recognizes a connection as being disturbed and closes it down when too many test frames are lost within a given period. Link Check uses several intervals to be able to recognize sudden occurrences of errors as well as a continuous low error rate.

A port that was turned off by Link Check must be reset to be able to communicate again. To do this you have 2 options:

- Pull out the connecting cable and plug it in again.
- Reset the function on both connection partners using the “Reset” button. This must be done on both devices within 30 s.

Note

When you use the “Reset” button, loops can form temporarily resulting in a loss of data traffic. The loop is automatically cleared again.

If this is not acceptable for your application, reset Link Check by pulling the cable and plugging it in again.

After resetting Link Check, the function is restarted on the port and the statistics are reset.

Configuring via a PROFINET IO controller

If MRP is configured via a PROFINET IO controller, you can start the Link Check function for the optical ring ports of the 1st MRP ring instance using WBM or CLI.

When a new configuration is transferred, Link Check is automatically disabled on all ports that were not configured as ring ports of the 1st MRP ring instance.

Note

Events relating to the Link Check function are reported only indirectly by PROFINET IO. If the MRP diagnostics alarms are enabled and a ring port is disabled by LinkCheck, Profinet IO generates an error message that the connection no longer exists.

MRP

MRP - Media Redundancy Protocol

The "MRP" method conforms to the Media Redundancy Protocol (MRP) specified in the following standard:

IEC 62439-2 Release 1.0 (2010-02) Industrial communication networks - High availability automation networks Part 2: Media Redundancy Protocol (MRP)

The reconfiguration time after an interruption of the ring is a maximum of 200 ms.

Topology

The following figure shows a possible topology for devices in a ring with MRP.

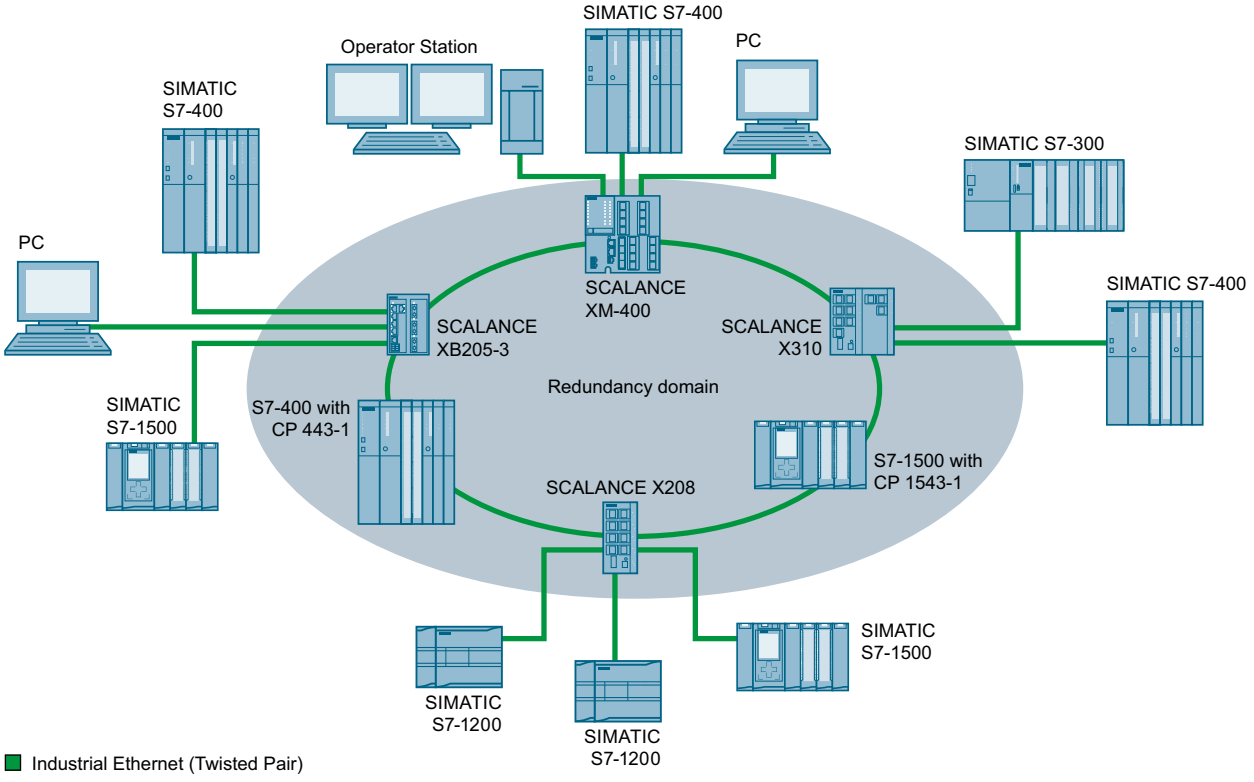

Figure 1-84 Example of a ring topology with the MRP media redundancy protocol

The following rules apply to a ring topology with media redundancy using MRP:

- All the devices connected within the ring topology are members of the same redundancy domain.
- One device in the ring is acting as redundancy manager.
- All other devices in the ring are redundancy clients.

Non MRP-compliant devices can be connected to the ring via a SCALANCE X switch or via a PC with a CP capable of MRP.

Requirements

The requirements for problem-free operation with the MRP media redundancy protocol are as follows:

- MRP is supported in ring topologies with up to 50 devices.
  Exceeding this number of devices can lead to a loss of data traffic.
- The ring in which you want to use MRP may only consist of devices that support this function. These include, for example, some of the Industrial Ethernet SCALANCE X switches, some of the communications processors (CPs) for SIMATIC S7 and PG/PC or non-Siemens devices that support this function.
- All devices must be interconnected via their ring ports.
  Multimode connections up to 3 km and single mode connections up to 26 km between two SCALANCE X IE switches are possible. At greater distances, the specified reconfiguration time may be longer.
- "MRP" must be enabled for all devices in the ring.
- The connection settings (transmission medium / duplex) must be set to full duplex and at least 100 Mbps for all ring ports. Otherwise there may be a loss of data traffic.
  - STEP 7: Set all the ports involved in the ring to "Automatic settings" in the "Options" tab of the properties dialog.
  - WBM: If you configure with Web Based Management, the ring ports are set automatically to autonegotiation.

Note

Number of devices

Except in PROFINET IO systems, topologies with up to 100 SCALANCE X-200 and SCALANCE X-300 IE switches were tested successfully.

Configuration in WBM

Role

The choice of role depends on the following use cases:

- You want to use MRP in a ring topology only with Siemens devices:
  - For at least one device in the ring select "Automatic Redundancy Detection" or "MRP Auto Manager".
  - For all other devices in the ring select "MRP Client" or "Automatic Redundancy Detection".
- You want to use MRP in a ring topology that also includes non-Siemens devices:
  - For exactly one device in the ring select the role "MRP Auto Manager".
  - For all other devices in the ring topology, select the role of "MRP client".

  Note

  The use of "Automatic Redundancy Detection" is not possible when using non-Siemens devices.

- You configure the devices in an MRP ring topology partly with WBM and partly with STEP 7:
  - With the devices you configure using WBM, select "MRP Client" for all devices.
  - With the devices that you configure using STEP 7, select precisely one device as "Manager" or "Manager (Auto)" and "MRP Client" for all other devices.

  Note

  If a device is assigned the role of "Manager" with STEP 7, all other devices in the ring must be assigned the "MRP Client" role. If there is a device with the "Manager" role and a device with the "Manager (Auto)"/"MRP Auto-Manager" in a ring, this can lead to circulating frames and therefore to failure of the network.

Configuration

In WBM, you configure MRP on the following pages:

- Configuration (Page 2152)
- Ring (Page 2181)

Configuration in STEP 7

Configuration in STEP 7

To create the configuration in STEP 7, select the parameter group "Media redundancy" on the PROFINET interface.

Set the following parameters for the MRP configuration of the device:

- Domain
- Role
- Ring port
- Diagnostic interrupts

These settings are described below.

Note

Valid MRP configuration

In the MRP configuration in STEP 7, make sure that all devices in the ring have a valid MRP configuration before you close the ring. Otherwise, there may be circulating frames that will cause a failure in the network.

One device in the ring needs to be configured as "redundancy manager" and all other devices in the ring as "clients".

Note

Note factory settings

MRP is disabled and spanning tree enabled for the following brand new IE switches and those set to the factory settings:

- SCALANCE XB-200 (Ethernet/IP variants)
- SCALANCE XC-200 (EtherNet/IP variants)
- SCALANCE XP-200 (Ethernet/IP variants)
- SCALANCE XR-300WG
- SCALANCE XM-400
- SCALANCE XR-500

To load a PROFINET configuration into one of the specified devices, first disable spanning tree on the device.

Note

Reconfiguration only when the ring is open

First open the ring before you

- change the MRP role or
- reconfigure ring ports

Note

Starting up and restarting

The MRP settings are still effective after a restart of the device or a power failure and hot restart as long as the power failure does not occur within 90 seconds after the configuration change.

Note

Prioritized startup

If you configure MRP in a ring, you cannot use the "prioritized startup" function in PROFINET applications on the devices involved.

If you want to use the "prioritized startup" function, then disable MRP in the configuration.

In the STEP 7 configuration, set the role of the relevant device to "Not a node in the ring".

Domain

Single MRP rings

If you want to configure a single MRP ring, leave the factory setting "mrpdomain 1" in the "Domain" drop-down list.

All devices configured in a ring with MRP must belong to the same redundancy domain. A device cannot belong to more than one redundancy domain in a single ring.

Multiple MRP rings

If you configure multiple single MRP rings, the nodes of the ring will be assigned to the individual rings with the "Domain" parameter. Set the same domain for all devices within a ring. Set different domains for different rings. Devices that do not belong to the same ring must have different domains.

If you want to configure MRP multiple rings, as the central redundancy manager for up to four rings select a device that is capable of multiple rings. Specify different domains for all ring instances and assign these to the corresponding ring ports of the redundancy manager. Configure the other devices as clients. The same domain must be set for all devices within a ring.

Note

Suitable devices for MRP multiple rings

You can use all products from the following product lines as redundancy manager connecting multiple rings:

- SCALANCE X-300 as of firmware version V4.0
- SCALANCE X-400 SCALANCE
  - X408-2 as of firmware version V4.0
  - X414-3E as of firmware version V3.10

Note

Suitable devices for MRP Interconnection

You can use all products from the following product lines as media redundancy interconnection manager and media redundancy interconnection client:

- SCALANCE XM-400 as of firmware version V6.2
- SCALANCE XR-500 as of firmware version V6.2

Role

Note

Reconfiguration only when the ring is open!

The choice of role depends on the following use cases.

- You want to use MRP in a topology with **one ring** only with Siemens devices and without monitoring diagnostic interrupts:
  Assign all devices to the "mrpdomain-1" domain and the role "Manager (Auto)".
  The device that actually takes over the role of redundancy manager, is negotiated by Siemens devices automatically.
- You want to use MRP in a topology with **multiple rings** only with Siemens devices and without monitoring diagnostic interrupts:
  - Assign all instances of the device that connects the rings the role of "Manager".
  - For all other devices in the ring topology, select the role of "Client".

- You want to use MRP in a ring topology that also includes non-Siemens devices or you want to receive diagnostic interrupts relating to the MRP status from a device (see "Diagnostic interrupts"):
  - Assign precisely one device in the ring the role of "Manager (Auto)".
  - For all other devices in the ring topology, select the role of "Client".
- You want to disable MRP:
  Select the option "Not node in the ring" if you do not want to operate the device within a ring topology with MRP.

Note

Role after resetting to factory settings

With brand new Siemens devices and those reset to the factory settings the following MRP role is set:

- "Manager (Auto)"
  - CPs
- "Automatic Redundancy Detection"
  - SCALANCE X-200
  - SCALANCE XB-200 (PROFINET variants)
  - SCALANCE XC-200 (PROFINET variants)
  - SCALANCE XF-200BA
  - SCALANCE XP-200 (PROFINET variants)
  - SCALANCE X-300
  - SCALANCE X-400

If you are operating a non-Siemens device as the redundancy manager in the ring, this may cause loss of the data traffic.

MRP is disabled and spanning tree enabled for the following brand new IE switches and those set to the factory settings:

- SCALANCE XB-200 (Ethernet/IP variants)
- SCALANCE XC-200 (EtherNet/IP variants)
- SCALANCE XP-200 (Ethernet/IP variants)
- SCALANCE XR-300WG
- SCALANCE XM-400
- SCALANCE XR-500

Ring port 1 / ring port 2

Here, select the port you want to configure as ring port 1 and ring port 2.

With devices with more than 8 ports, not all ports can be selected as ring port.

The drop-down list shows the selection of possible ports for each device type. If the ports are specified in the factory, the boxes are grayed out.

NOTICE
Ring ports after resetting to factory settings If you reset to the factory settings, the ring port settings are also reset. If other ports were used previously as ring ports before resetting, with the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

Note

Reconfiguration only when the ring is open

First open the ring before you reconfigure the ring ports of a ring manager.

Diagnostic interrupts

Enable the "Diagnostic interrupts" option, if you want diagnostic interrupts relating to the MRP status on the local CPU to be output.

The following diagnostic interrupts can be generated:

- Wiring or port error
  Diagnostic interrupts are generated if the following errors occur at the ring ports:
  - Connection abort on a ring port
  - A neighbor of the ring port does not support MRP.
  - A ring port is connected to a non-ring port.
  - A ring port is connected to the ring port of another MRP domain.
- Status change active/passive (redundancy manager only)
  If the status changes (active/passive) in a ring, a diagnostics interrupt is generated.

Parameter assignment of the redundancy is not set by STEP 7 (redundancy alternatives)

This option affects all SCALANCE X switches. Select this option if you want to set the properties for media redundancy using alternative mechanisms such as WBM, CLI or SNMP.

If you enable this option, existing redundancy settings are retained and are not overwritten. The parameters in the "MRP configuration" box are then reset and grayed out. The entries then have no meaning.

MRP Interconnection

The MRP Interconnection mode is an extension of MRP and enables redundant linking of two or more MRP rings in networks without real-time capability. Like MRP, MRP Interconnection is specified in the standard IEC 62439-2. MRP Interconnection allows for very fast reconfiguration; the reconfiguration time is typically less than 200 milliseconds.

Topology

The diagram below shows the redundant linking of two MRP rings. In addition, MRP Interconnection also supports additional topologies which are, however, not supported by the current firmware.

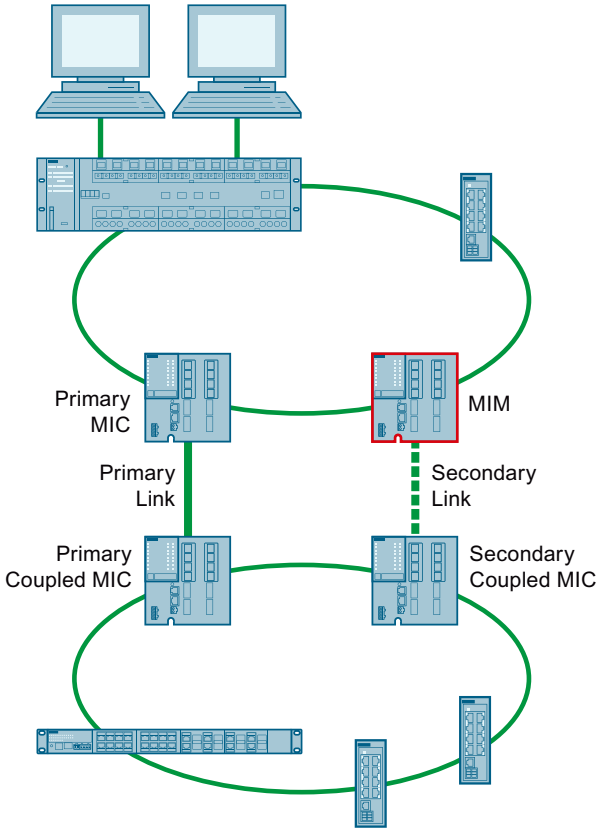

Figure 1-85 Redundant connection of two MRP rings with MRP Interconnection

Operating principle

The requirement for MRP Interconnection is that MRP is used in all rings involved. Four devices are required for the two MRP Interconnection connections:

- One Media Redundancy Interconnection Manager (MIM, shown with a red outline in the diagram)
- Three Media Redundancy Interconnection Clients (Primary MIC, Primary Coupled MIC and Secondary Coupled MIC)

Because each device is part of an MRP ring, each device also takes on one of the roles defined for MRP, i.e. MRC or MRM.

Depending on the connection status of the interconnection ports, the MIC sends status messages (Link up or Link down) to the MIM. Interconnection ports are ports that are connected over the primary or secondary connection. This means the MIM is always informed about the connection status between the Primary MIC and the Primary Coupled MIC ("primary connection") as well as its own connection to the Secondary Coupled MIC ("secondary connection"). In regular operation, the data exchange between the two rings is via primary

connection and the MIM blocks its interconnection port. If a Link down of the primary connection is signaled to MIM, it switches its interconnection port to the "Forwarding" status, and the data exchange between the two rings is via secondary connection between MIM and Secondary Coupled MIC.

Note

Suitable devices for MRP Interconnection

The Interconnection Manager, the Interconnection Client and all other devices in the rings must support MRP Interconnection. This is the case for the following devices:

- SCALANCE XM-400 as of firmware version V6.2
- SCALANCE XR-500 as of firmware version V6.2

Additional documentation

You can find additional descriptions of the configuration of MRP Interconnection in the WBM configuration manual of SCALANCE XM-400/SCALANCE XR-500.

You will find the supplementary documentation here:

- On the data medium that ships with some products:
  - Product CD / product DVD
  - SIMATIC NET Manual Collection
- On the Internet pages of Siemens Industry Online Support (https://support.industry.siemens.com/cs/ww/en/ps/15313/man)

Parallel Redundancy Protocol

Parallel Redundancy Protocol

The "Parallel Redundancy Protocol" (PRP) is a redundancy protocol for Ethernet networks. It is defined in Part 3 of the IEC 62439 standard. This redundancy method allows data communication to be maintained without interruption/reconfiguration time if there are interruptions in the network.

The PRP method is supported, for example, by the devices of the SCALANCE X-200RNA product line.

Overlong frames

When sending PRP frames, the IE switch expands the frame with a PRP trailer. With frames with the maximum length, appending the PRP trailer results in an overlong frame that exceeds the maximum permitted frame length (according to the IEEE 802.3 standard).

To prevent data loss with overlong frames, all network components located in a PRP network must support a frame length of at least 1528 bytes.

The devices described in this manual can be used in PRP networks, see also section "Configuration limits (Page 1965)".

Routing function

Introduction

The term routing describes the specification of routes for communication between different networks; in other words, how does a data packet from subnet A get to subnet B.

SCALANCE X supports the following routing functions:

- Static routing
  With static routing, the routes are entered manually in the routing table.
- Router redundancy
  With standardized VRRP (Virtual Router Redundancy Protocol), the availability of important gateways is increased by redundant routers.
  - VRRPv2 (IPv4)
  - VRRPv3 (IPv4 / IPv6)
- Dynamic routing
  The entries in the routing table are dynamic and are updated continuously. The entries are created with one of the following dynamic routing protocols:
  - OSPFv2 (IPv4)
  - OSPFv3 (IPv6)
  - RIPv2 (IPv4)
  - RIPng (IPv6)
  - MSDP
  - PIM Sparse Mode
  - BiDir PIM

VRRP

VRRPv2

Router redundancy with VRRP

With the Virtual Router Redundancy Protocol (VRRP), the failure of a router in a network can be countered.

VRRP can only be used with virtual IP interfaces (VLAN interfaces) and not with router ports.

Several VRRP routers in a network segment are put together as a logical group representing a virtual router (VR). The group is defined using the virtual ID (VRID). Within the group, the VRID must be the same. The VRID can no longer be used for other groups.

The virtual router is assigned a virtual IP address and a virtual MAC address. One of the VRRP routers within the group is specified as the master router. The master router has priority 255. The other VRRP routers are backup routers. The master router assigns the virtual IP address and the virtual MAC address to its network interface. The master router sends VRRP packets

(advertisements) to the backup routers at specific intervals. With the VRRP packets, the master router signals that it is still functioning. The master router also replies to the ARP queries.

If the virtual master router fails, a backup router takes over the role of the master router. The backup router with the highest priority becomes the master router. If the priority of the backup routers is the same, the higher MAC address decides. The backup router becomes the new virtual master router.

The new virtual master router adopts the virtual MAC and IP address. This means that no routing tables or ARP tables need to be updated. The consequences of a device failure are therefore minimized.

You configure VRRP in "Layer 3 (IPv4) > VRRP".

VRRP3

Version 3 of VRRP is based on version 2.

Note

- Enable routing to be able to use VRRPv3.
- You can only use VRRPv3 in conjunction with VLAN interfaces. Router ports are not supported.
- Simultaneous operation of VRRP and VRRPv3 is not possible.
- VRRPv3 supports IPv4 and IPv6. Both can be configured and operated at the same time with VRRP3.

You configure VRRPv3 in:

IPv4: Layer 3 (IPv4)> VRRPv3

IPv6: Layer 3 (IPv6)> VRRPv3

OSPF

OSPFv2

Dynamic routing with OSPFv2

OSPF (Open Shortest Path First) is a cost-based routing protocol. To calculate the shortest and most cost-effective route, the Short Path First algorithm by Dijkstra is used. OSPF was developed by the IETF (Internet Engineering Task Force).

You configure OSPFv2 in "Layer 3 (IPv4) > OSPFv2".

OSPFv2 divides an autonomous system (AS) into different areas.

Areas in OSPF

The following areas exist:

- Backbone
  The backbone area is area 0.0.0.0. All other areas are connected to this area. The backbone area is connected either directly or via virtual connections with other areas.
  All routing information is available in the backbone area. As a result, the backbone area is responsible for forwarding information between different areas.
- Stub Area
  This area contains the routes within its area within the autonomous system and the standard route out of the autonomous system. The destinations outside this autonomous system are assigned to the standard route.
- Totally Stubby Area
  This area knows only the routes within its area and the standard route out of the area.
- Not So Stubby Area (NSSA)
  This area can forward (redistribute) packets from other autonomous systems into the areas of its own autonomous system. The packets are further distributed by the NSSA router.

Routers of OSPF

OSPF distinguishes the following router types:

- Internal router (IR)
  All OSPF interfaces of the router are assigned to the same area.
- Area Border Router (ABR)
  The OSPF interfaces of the router are assigned to different areas. One OSPF interface is assigned to the backbone area. Where possible, routes are grouped together.
- Backbone Router (BR)
  At least one of the OSPF interfaces is assigned to the backbone area.
- Autonomous System Border Router (ASBR)
  One interface of the router is connected to a different AS, for example an AS that uses the routing protocol RIP.

Virtual connection

Each area must be connected to the backbone area. In some situations a direct physical connection is not possible. In this case, a router of the relevant area must be connected to a backbone router via a virtual connection.

LSA types

Within the autonomous system, packets are exchanged that contain information about the connections of a router and the connection status message. The packets are also known as LSAs (Link State Advertisements). The LSAs are always sent from the router to the neighbor router.

If there are changes in the network, LSAs are sent to all routers in the network. The information depends on the LSA type.

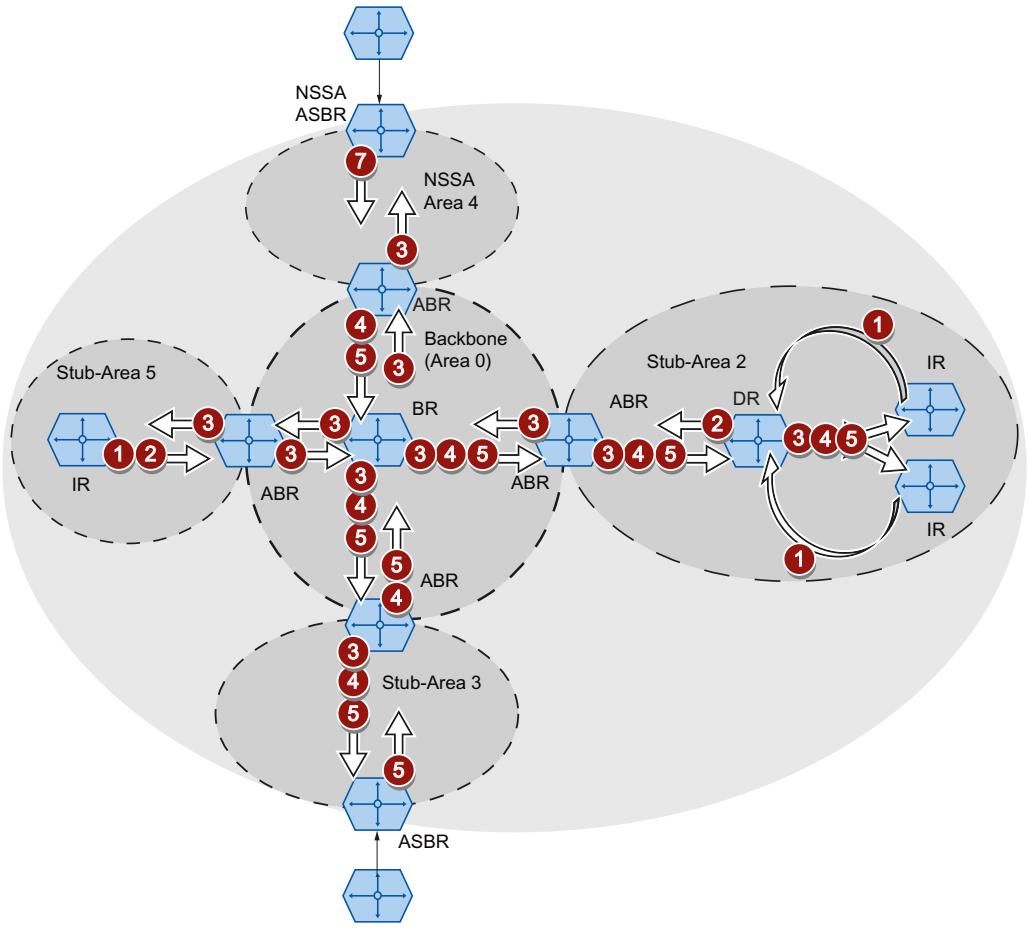

1. **Router LSA (LSA Type 1)**
   The LSA Type 1 is only sent within an area. For each active connection of the router that belongs to the area in consideration, an LSA Type 1 is generated. The LSA Type 1 contains information about the status and the costs of the connection, for example IP address, network mask, network type
2. **Network LSA (LSA Type 2)**
   The LSA Type 2 is sent only within an area. For each network that belongs to the relevant area, the router generates an LSA Type 2. If several routers are interconnected in a network, the LSA Type 2 is sent by the designated router (DR). The LSA Type 2 includes the network address, the network mask and a list of routers that are connected to the network

3 **Summary LSA (LSA Type 3 / LSA Type 4)**
The Summary LSA is generated by the area border router and sent into the area. The
4 Summary LSA contains information about routes outside the area but inside the AS. Where possible, the routes are grouped together.

- Summary LSA (LSA Type 3)
  The LSA Type 3 describes the routes to the networks and advertises the standard route to the areas.
- AS Summary LSA (LSA Type 4)
  The LSA Type 4 describes the routes to the ASBR.

5 **External LSA (LSA Type 5 / LSA Type 7)**
The External LSA is generated by the ASBR. The LSA type depends on the area.

7
- AS External LSA (LSA Type 5)
  The LSA Type 5 is sent by the AS border router into the areas of the autonomous system except the Stub and NSSA areas. The LSA contains information about routes to a network in another AS. The routes are either created manually or learned externally. The ASBR uses LSA Type 5 to distribute standard routes to the backbone area.
- NSSA External LSA (LSA Type 7)
  The LSA Type 7 is generated by the AS border router of an NSSA. The router is also known as the NSSA ASBR. The LSA Type 7 is sent only within the NSSA. If the P bit in LSA Type 7 = 1, these LSAs are converted to LSA Type 5 by the ABR and sent to the backbone area.

Establishing the neighborhood

The router runs through the following statuses to establish a connection to the neighbor router.

1. Attempt state / Init state
   The router activates OSPF and begins to send and receive Hello packets. Based on the received Hello packets, the router learns which OSPF routers are in its vicinity. The router checks the content of the Hello packet. The Hello packet also contains the list of the neighbor routers (neighbor table) of the "sender".
2. Two way state
   If, for example, the ID of the area, the area type and the settings for the times match, a connection (adjacency) can be established to the neighbor. In a point-to-point network, the connection is established directly. If several neighbor routers can be reached in a network, the designated router (DR) and the backup designated router (BDR) are identified based on Hello packets. The router with the highest router priority becomes the designated router. If two routers have the same router priority, the router with the highest router ID becomes the designated router. The router establishes a connection to the designated router.
3. Exchangestart state
   The neighbor routers decide which router starts communication. The router with the higher router ID becomes the designated router.
4. Exchange state
   The neighbor routers send packets that describe the content of their neighborhood database. The neighborhood database (link state database - LSDB) contains information on the topology of the network.

5. Loading state
   The router completes the received information. If the router still has questions relating to the status of a specific connection, it sends a link state request. The neighbor router sends a response (link state update). The response contains a suitable LSA. The router confirms receipt of the response (link state acknowledge).
6. Full State
   The information exchange with the neighbor router is completed. The neighborhood database of the neighbor router is the same. Based on the Short Path First algorithm, the router calculates a route to every destination. The route is entered in the routing table.

Check the neighborhood

The Hello packets are only used to establish the neighborhood relations. Hello packets are used to check the connection to the neighbor router by sending them cyclically. If no Hello packet is received within a certain interval (dead interval), the connection to the neighbor is marked as "down". The relevant entries are deleted.

Updating the neighborhood database

Once the neighborhood database is established, LSAs are sent to all routers in the network if there are changes in the topology.

OSPFv3

Version 3 of OSPF is based on version 2 and is only used with IPv6. A large part of the routing mechanisms was adopted. OSPFv3 is defined in the RFCs 2740 and 5340.

You configure OSPFv3 under "Layer 3 (IPv6) > OSPFv3".

The following has not changed:

- The statuses that a router runs through to establish a connection to the neighbor router.
- The areas : Backbone, Stub Area, Totally Stubby Area, Not So Stubby Area (NSSA)
- The router types: Internal Router (IR), Area Border Router (ABR), Backbone Router (BR), Autonomous System Area Border Router (ASBR), Designated Router (DR)
- The router ID, the area ID and the ID of the LSA are entered in the IPv4 address format: x.x.x.x

What has changed?

Terms

The terms network or subnet are replaced by link.

Authentication

The authentication was removed. Instead OSPFv3 uses IPsec, that is implemented in IPv6.

Neighbor routers

The neighbor routers are identified via the router ID.

Neighbor database

The neighbor database (link state database - LSDB) is divided into different areas of application:

- Link scope LSDB
  Contains the link LSA
- Area scope LSDB
  contains the following LSAs
  - Router LSA
  - Network LSA
  - Inter-area prefix LSA
  - Inter area router LSA
  - Intra area prefix LSA
- AS scope LSDB
  Contains the AS external LSA

LSA types

Two new LSA types were defined for OSPFv3.

OSPFv2		OSPFv3		Who	Within	Description
1	Router LSA	0x2001	Router LSA	every router	Area	No longer contains address information. This is contained in the new LSA type 2009.
2	Network LSA	0x2002	Network LSA	DR	Area	No longer contains address information. This is contained in the new LSA type 2009.
3	Summary LSA	0x2003	Inter-area prefix LSA	ABR	Area	Same function as in OSPFv2, simply renamed
4	AS Summary LSA	0x2004	Inter-Area Router LSA	ABR	Area	Same function as in OSPFv2, simply renamed
5	AS External LSA	0x4005	AS External LSA	ASBR	AS	Same function as in OSPFv2, simply renamed
7	NSSA External LSA	0x2007	Type 7 LSA	NSSA ASBR	NSSA	Same function as in OSPFv2, simply renamed
		0x2008	Link LSA	every router	Link	The LSA is sent by the router to every router linked to it. The LSA contains the link local address of the router and a list with IPv6 prefixes configured on the link.
		0x2009	Intra area prefix LSA	every router	Area	The LSA is sent only within an area. The LSA contains the IPv6 prefixes connected to the router or network.

In contrast to OSPFv2, OSPFv3 can forward unknown LSA types. Previously these were deleted and not distributed further.

RIP

RIPv2

Dynamic routing with RIPv2

The Routing Information Protocol (RIPv2) is used to create routing tables automatically. RIPv2 is used in autonomous systems (AS) with a maximum of 15 hops. It is based on the Distance-Vector algorithm.

RIPv2 was developed by the IETF (Internet Engineering Task Force) and is described in RFC 2453.

You configure RIPv2 in "Layer 3 (IPv4) > RIPv2".

Setting up a routing table

Since a router initially only knows its directly connected networks, it sends a request to its direct neighbor routers. As the reply, it receives the routing tables of the neighbor routers. Based on the information it receives, the router set up its own routing table.

The routing table contains entries for all possible destinations. Each entry includes the distance to the destination and the first router on the route.

The distance is also known as the metric. This indicates the number of routers to be passed through on the route to the destination (hop count). The maximum distance is 15 hops.

Updating the routing table

Once the routing table is set up, the router sends its routing table to each direct neighbor router via the UDP port 520 at intervals of 30 seconds.

The router compares new routing information with its existing routing table. If the new information includes shorter routes, the existing routes are overwritten. The router only keeps the shortest route to a destination.

Checking neighbor routers

If a router does not receive messages from a neighbor router for longer than 180 seconds, it marks the router as being invalid. The router assigns the metric 16 for the neighbor router.

RIPng

RIPng (RIP next generation) is only used with IPv6 and is defined in RFC 2080. As with RIP (IPv4), RIPng is based on the distance vector algorithm of Bellman-Ford.

In contrast to RIPv2, RIPng is activated directly at the interface (VLAN interface / router port) and not globally on the device.

RIPng uses the UDP port 521 and RIP the UDP port 520.

You configure RIPng in "Layer 3 (IPv6) > RIPng".

PIM

Protocol Independent Multicast (PIM) allows the routing of multicast packets, regardless of lower-level routing protocols such as OSPFv2 or static routing (IPv4). PIM expands the routing information of the unicast routing protocol active on the router with additional information for multicast operation.

Requirements for PIM:

- IGMP is enabled on the first hop and last hop of the routing topology.
- PIM is enabled on all routers of the routing topology.
- There is at least one rendezvous point (RP).
- In every subnet there is a designated router (DR).
- The DR must also be the IGMP querier.

PIM network

PIM operates in the sparse mode designed for networks with a low node density.

Sparse mode (SM)

If a router receives a multicast, it sends this information to the rendezvous point (RP). A device that wants to receive a multicast sends an IGMP Join to the routers in its subnet. The designated router (DR) of the network sends this request to the RP. The RP therefore has the unicast address of the sender and a unicast address for the recipient, that of the DR.

In Sparse mode between the sender and recipient the shortest path is selected based on the routing table and the information can be sent.

Bidirectional multicast and designated forwarder

The "Bidirectional multicast" functionality is a variant of the Sparse mode in which the information between sender and recipient is always sent via the rendezvous point.

In Bidirectional multicast the DR is replaced by the designated forwarder (DF). When the DF receives an IGMP Join, it sends a (*, G) message to the RP. The RP replaces the * with the source and so establishes the connection. Only the DF forwards the multicast.

Rendezvous point (RP) and bootstrap router (BSR)

The rendezvous point manages the information about the sender and recipient: You can create RPs statically or define RP candidates.

When you define RP candidates you require a bootstrap router. The BSR coordinates the RP candidates and decides on an RP. You define BSR candidates of which one becomes the bootstrap router.

BSR border

With the "BSR border" function you can divide a PIM network into domains. There is no BSR communication between the PIM domains so a BSR is selected in every domain. The BSR of the domain selects an RP for the domain.

MSDP

The RPs communicate beyond PIM domain boundaries using MSDP and synchronize the multicast groups they receive. This means that a device in domain A can receive a multicast although the information about the sender is managed by the RP in domain B.

Source specific multicast (SSM)

PIM source specific multicast is a further variant of Sparse mode that manages without rendezvous points. Via IGMPv3, the recipient directly informs the router of the source and the multicast it would like to receive.

Example

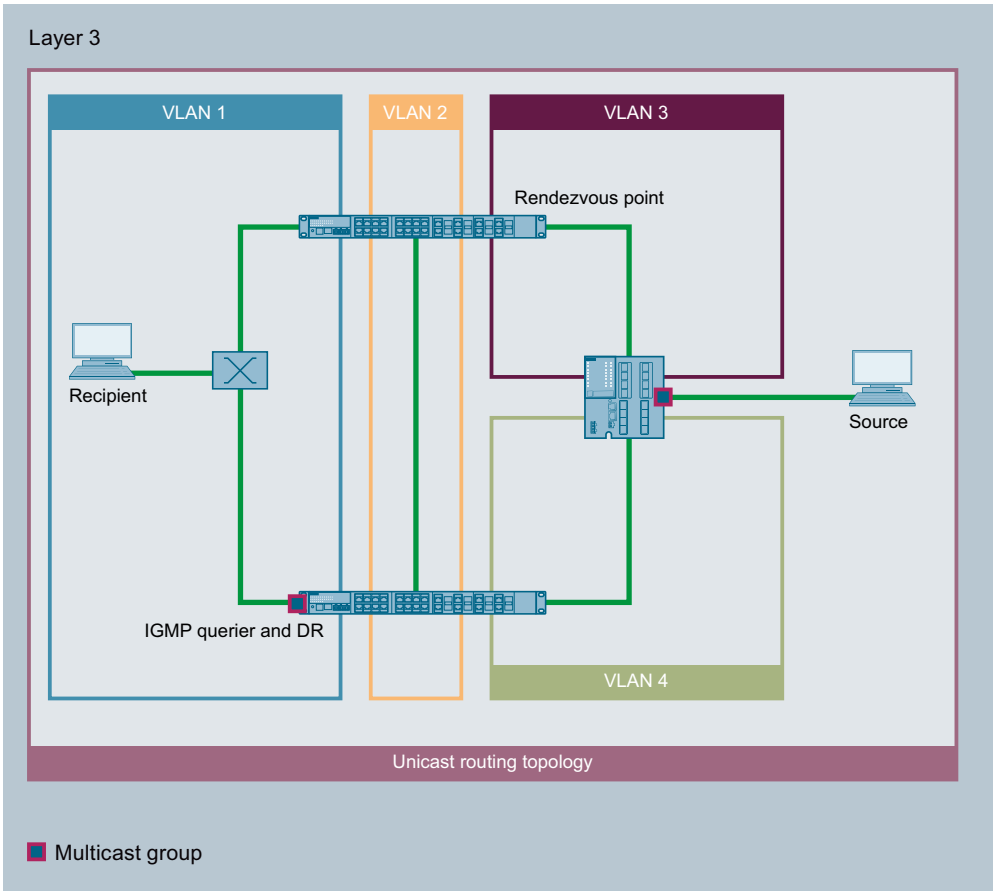

In this example the source sends a multicast to the connected router. The router sends this information to the rendezvous point.

The recipient sends an IGPM Join to its IGMP querier and designated router. The DR forwards the request to the RP.

Based on the routing table, the shortest path is selected for the multicast.

NAT/NAPT

Note

NAT/NAPT is possible only on layer 3 of the ISO/OSI reference model. To use the NAT function, the networks must use the IP protocol.

When using the ISO protocol that operates at layer 2, it is not possible to use NAT.

In Network Address Translation (NAT) IP subnets are divided into "Inside" and "Outside". The division is from the perspective of a NAT interface. All networks reachable via the NAT interface itself count as "Outside" for this interface. All networks reachable via other IP interfaces of the same device count as "Inside" for the NAT interface.

If there s routing via the NAT interface, the source or destination IP addresses of the transferred data packets are changed at the transition between "Inside" and "Outside". Whether or not the source or destination IP address is changed depends on the communications direction. It is always the IP address of the communications node that is located "Inside" that is adapted. Depending on the perspective the IP address of a communications node is always designated as "Local" or "Global".

		Perspective	
		Local	Global
Position	Inside	An actual IP address that is assigned to a device in the internal network. This address cannot be reached from the external network.	An IP address at which an internal device can be reached from the external network.
	Outside	An actual IP address that is assigned to a device in the external network. Since only "Inside" addresses are converted, there is no distinction made between outside local and outside global.	

Example

In the example two IP subnets are connected together via an IE switch. The division is from the perspective of the NAT interface 10.0.0.155. The communication of PC2 with PC1 is implemented via NAT/NAPT.

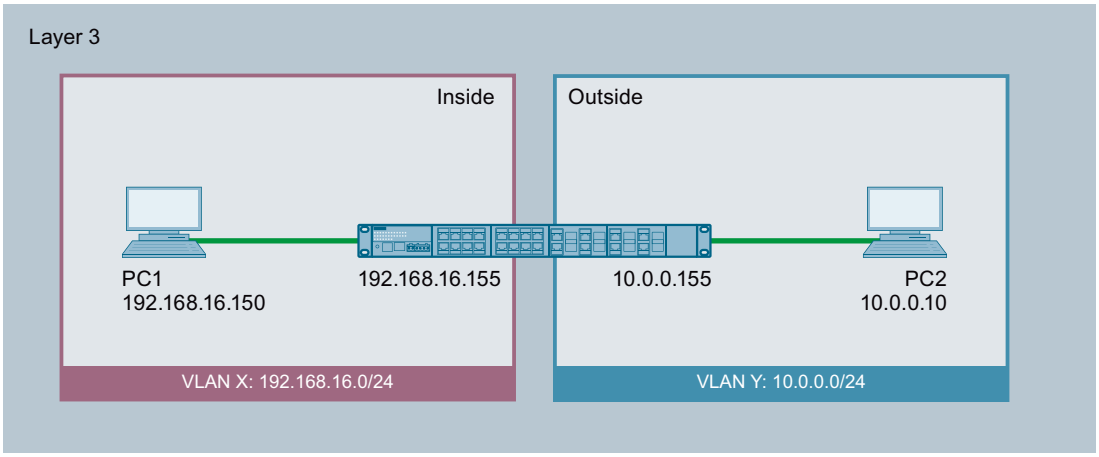

The actual IP address of PC1 (inside local) is implemented statically with NAT. For PC2, PC1 can be reached at the inside global address.

		Perspective	
		Local	Global
Position	Inside	192.168.16.150	10.0.0.7
	Outside	10.0.0.10	

The actual IP address of PC1 (inside local) is implemented with NAPT (Network Address and Port Translation).. For PC2, PC1 can be reached at the inside global address.

		Perspective	
		Local	Global
Position	Inside	192.168.16.150:80	10.0.0.7:80
	Outside	10.0.0.10:1660	

Computing capacity

Due to the load limitation of the CPU packet receipt of the device is limited to 300 packets a second. This corresponds to a maximum data through of 1.7 Mbps. This load limitation does not apply per interface but generally for all packets going the CPU.

The entire NAT communication runs via the CPU and therefore represents competition for IP communication going to the CPU, e.g. WBM and Telnet.

Note that a large part of the computing capacity is occupied if you use NAT. This can slow down access via Telnet or WBM.

NAT

With Network Address Translation (NAT), the IP address in a data packet is replaced by another. NAT is normally used on a gateway between an internal network and an external network.

With source NAT, the inside local source address of an IP packet from a device in the internal network is rewritten by a NAT device to an inside global address at the gateway.

With destination NAT, the inside global source address of an IP packet from a device in the external network is rewritten by a NAT device to an inside local address at the gateway.

To translate the internal into the external IP address and back, the NAT device maintains a translation list. The address assignment can be dynamic or static. You configure NAT in "Layer 3 (IPv4) > NAT (Page 2227)".

NAPT

In "Network Address Port Translation" (NAPT), several internal source IP addresses are translated into the same external IP address. To identify the individual nodes, the port of the internal device is also stored in the translation list of the NAT device and translated for the external address.

If several internal devices send a query to the same external destination IP address via the NAT device, the NAT device enters its own external source IP address in the header of these

forwarded frames. Since the forwarded frames have the same external source IP address, the NAT device assigns the frames to the devices using a different port number.

If a device from the external network wants to use a service in the internal network, the translation list for the static address assignment needs to be configured. You configure NAPT in "Layer 3 (IPv4) > NAT > NAPT (Page 2230)".

NAT/NAPT and IP routing

You can enable NAT/NAPT and IP routing at the same time. In this case, you need to regulate the reachability of internal addresses from external networks with ACL rules.

Single-Hop Inter-VLAN-Routing

Introduction

A physical network is divided into broadcast domains and subnets by VLANs.

Devices (hosts) within a VLAN can communicate with each other directly via layer 2. The frames are forwarded to the relevant device based on the MAC address.

Devices from different VLANs cannot communicate with each other directly via layer 2. The data traffic must be routed based on the IP address.

With the Single-Hop Inter-VLAN-Routing function it is possible that devices from different VLANs communicate with each other without a router being necessary.

Requirements

- The IE switch can manage several IP interfaces.
- The switch is a member in the VLANs to be routed.
- With the hosts, the IP address of the VLAN is entered as default gateway.

Single-Hop Inter-VLAN-Routing

The IE switch receives a frame and recognizes that it is addressed to a device in another VLAN. It forwards the frame to the corresponding port in the VLAN.

The IE switch only knows VLANs with which it is directly connected (Connected). With Single-Hop Inter-VLAN-Routing it is therefore only possible to route between two local IP interfaces.

Example

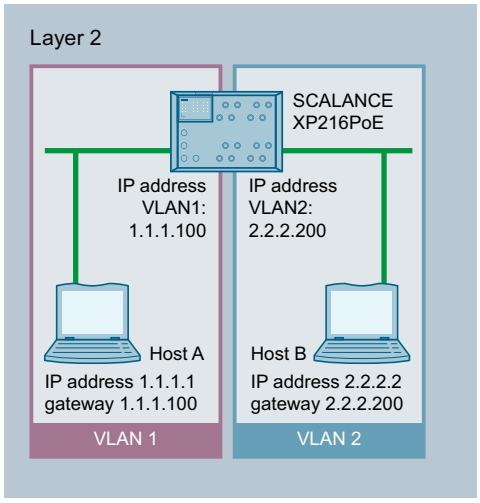

In this example, the host A is connected to the IE switch via VLAN1. Host A is connected to the IE switch via VLAN2. With host A, the IP address of VLAN 1 is entered as the default gateway. With host B the IP address of VLAN 2 is entered as the default gateway.

If the Single-Hop Inter-VLAN-Routing function is enabled on the SCALANCE XP216PoE, host A and host B can communicate with each other.

Link aggregation

Link aggregation

With link aggregation, several parallel physical connections with the same transmission speed are grouped together to form a logical connection with a higher transmission speed. This method based on IEEE 802.3ad is also known as port trunking or channel bundling.

Link aggregation works only with full duplex connections with the same transmission speed in point-to-point mode. This achieves multiplication of the bandwidth or transmission speed. If part of the connection fails, the data traffic is handled via the remaining parts of the connection.

To control and monitor, the Link Aggregation Control Layer (LACL) and the Link Aggregation Control Protocol (LACP) are used.

Quality of service

Quality of Service (QoS) is a method to allow efficient use of the existing bandwidth in a network.

QoS is implemented by prioritization of the data traffic. Incoming frames are sorted into a Queue according to a certain prioritization and further processed. This gives certain frames priority.

The different QoS methods influence each other and are therefore taken into account in the following order:

1. The switch first checks whether the incoming frame is a broadcast or agent frame.
   → When the first condition is met, the switch takes into account the priority set on the "General (Page 2156)" page.
   The switch sorts the frame into a queue according to the assignment on page "CoS Map (Page 2157)".
2. If the first condition is not met the switch checks whether the frame contains a VLAN tag.
   → If the second condition is met the switch checks the settings for the priority on the "General (Page 2156)" page. The switch checks whether a value other than "Do not force" is set for the priority.
   If the priority is set the switch sorts the frame into a queue according to the assignment on page "CoS Map (Page 2157)".
3. If the second condition is also not met the frames are further processed according to the Trust mode. You configure the Trust mode on the page "QoS Trust (Page 2159)".

See also

General (Page 2163)

Authentication

Overview of user management

Access to the device is managed by configurable user settings. Set up users with a password for authentication. Assign a role with suitable rights to the users.

The authentication of users can either be performed locally by the device or by an external RADIUS server. You configure how the authentication is handled on the "Security > AAA > General" page.

When you transfer the configuration of a device to TIA, the configured users, roles and groups are not transferred.

Compatibility with predecessor versions

With firmware version 5.1, user management was expanded by the RADIUS authorization mode "Vendor Specific". To ensure compatibility with firmware versions ≤ 5.0, the default setting was selected so that following a firmware update the earlier authentication mode "conventional" continues to be used.

Local login

The local logging in of users by the device runs as follows:

1. The user logs in with user name and password on the device.
2. The device checks whether an entry exists for the user.
   → If an entry exists, the user is logged in with the rights of the associated role.
   → If no corresponding entry exists, the user is denied access.

Login via an external RADIUS server

RADIUS(Remote Authentication Dial-In User Service) is a protocol for authenticating and authorizing users by servers on which user data can be stored centrally.

Depending on the RADIUS authorization mode you have selected on the "Security > AAA > RADIUS Client" page, the device evaluates different information of the RADIUS server.

RADIUS authorization mode "conventional"

If you have set the authorization mode "conventional", the authentication of users via a RADIUS server runs as follows:

1. The user logs in with user name and password on the device.
2. The device sends an authentication request with the login data to the RADIUS server.
3. The RADIUS server runs a check and signals the result back to the device.
   - The RADIUS server reports a successful authentication and for the "Service Type" attribute returns the value "Administrative User" to the device
     → The user is logged in with administrator rights.
   - The RADIUS server reports a successful authentication and returns a different or even no value to the device for the attribute "Service Type".
     → The user is logged in with read rights.
   - The RADIUS server reports a failed authentication to the device:
     → The user is denied access.

RADIUS authorization mode "Vendor Specific"

Requirement

For the RADIUS authorization mode "Vendor Specific" the following needs to be set on the RADIUS server:

- Manufacturer code: 4196
- Attribute number: 1
- Attribute format: Character string (group name)

Procedure

If you have set the authorization mode "Vendor Specific", the authentication of users via a RADIUS server runs as follows:

1. The user logs in with user name and password on the device.
2. The device sends an authentication request with the login data to the RADIUS server.
3. The RADIUS server runs a check and signals the result back to the device.
   Case A: The RADIUS server reports a successful authentication and returns the group assigned to the user to the device.
   - The group is known on the device and the user is not entered in the table "External User Accounts"
     → The user is logged in with the rights of the assigned group.
   - The group is known on the device and the user is entered in the table "External User Accounts"
     → The user is assigned the role with the higher rights and logged in with these rights.
   - The group is not known on the device and the user is entered in the table "External User Accounts"
     → The user is logged in with the rights of the role linked to the user account.
   - The group is not known on the device and the user is not entered in the table "External User Accounts"
     → The user is logged in with the rights of the role "Default".

   Case B: The RADIUS server reports a successful authentication but does not return a group to the device.
   - The user is entered in the table "External User Accounts":
     → The user is logged in with the rights of the linked role.
   - The user is not entered in the table "External User Accounts":
     → The user is logged in with the rights of the role "Default".

   Case C: The RADIUS server reports a failed authentication to the device:
   - The user is denied access.

Authentication method

You can configure the authentication methods "802,1X" and "MAC Authentication" as well as the "Guest VLANGuest VLAN" option separately for each port.

The functions have a hierarchical order. If all three functions are enabled, an attempt is made initially to authenticate the end device using "802.1x". If this authentication is unsuccessful, "MAC Authentication" is started. If this authentication is also unsuccessful, the end device is enabled for communication in the "Guest VLAN". "Guest VLAN" can only be used if at least one authentication method is active.

The two authentication methods depend on the end device. If the end device supports EAP (Extensible Authentication Protocol), it can be authenticated using the "802.1X" method. If the end device does not support EAP, it can be authenticated using "MAC Authentication". In this case, the IE switch adopts the role of the end device and uses the MAC address of the device as the authentication parameter.

802.1X

The "802.1X" authentication method works as follows:
An end device that supports EAP sends authentication information to the IE switch. The IE switch forwards the information to the authentication server. The authentication server checks the information and allows or denies the end device access to the network.

MAC Authentication

The "MAC Authentication" authentication method works as follows:
As soon as the IE switch receives a frame from the end device, the IE switch sends a query to the RADIUS server to allow or deny the end device access to the network.

Assignment of a VLAN via RADIUS or guest VLAN

Authentication with a change to the VLAN configuration

If during authentication a port is assigned to a VLAN dynamically using the function "RADIUS VLAN Assignment Allowed" or "Guest VLAN" the following happens:

- If the VLAN that is to be assigned has not been created on the device, the authentication is rejected.
- If the VLAN that is to be assigned has been created on the device:
  - The port becomes an untagged member in the assigned VLAN if it was not already.
  - The PVID of the port is changed to the ID of the assigned VLAN.

Note

If the port is only to be assigned to one VLAN, you need to adapt the VLAN configuration manually. As default, all ports are untagged members in "vlan 1".

If the authentication is canceled, e.g. by link down, the dynamic changes are canceled.

- The port is no longer a member in the assigned VLAN.
- The PVID of the port is reset to the value it had prior to authentication.

Authentication without a change to the VLAN configuration

If during authentication no VLAN is assigned either by the function "RADIUS VLAN Assignment Allowed" or by "Guest VLAN", the existing VLAN configuration of the port remains unchanged.

Editing properties and parameters

Editing options

You have the following options for editing properties and parameters:

- Hardware and network editor
  Once you have inserted the network component, you can edit the properties and parameters, for example the device name. You will find more detailed information in "Hardware and network editor".
- Web Based Management (WBM)
  The parameters and properties are accessible using the supplied HTML pages (WBM pages). Each WBM page has its own help page that describes the properties and parameters. For further information, refer to the configuration manuals:
  - "SCALANCE XB-200/XC-200/XP-200 Web Based Management"
    You will find the configuration manual on the Internet pages of Siemens Industry Online Support inSCALANCE XB-200 (https://support.industry.siemens.com/cs/ww/en/ps/15291/man), SCALANCE XC-200 (https://support.industry.siemens.com/cs/ww/en/ps/15291/man) and SCALANCE XP-200 (https://support.industry.siemens.com/cs/ww/en/ps/21869/man).
  - "SCALANCE XM-400/XR-500 Web Based Management"
    You will find this configuration manual on the Internet pages of Siemens Industry Online Support in SCALANCE XM-400 (https://support.industry.siemens.com/cs/ww/en/ps/15315/man) and SCALANCE XR-500 (https://support.industry.siemens.com/cs/ww/en/ps/15317/man).
- Command Line Interface
  All the configuration settings for the device can be made using the CLI. The CLI provides the same options as Web Based Management (WBM). For further information, refer to the configuration manuals:
  - "SCALANCE XB-200/XC-200/XP-200 Command Line Interface"
    You will find the configuration manual on the Internet pages of Siemens Industry Online Support inSCALANCE XB-200 (https://support.industry.siemens.com/cs/ww/en/ps/15291/man), SCALANCE XC-200 (https://support.industry.siemens.com/cs/ww/en/ps/15291/man) and SCALANCE XP-200 (https://support.industry.siemens.com/cs/ww/en/ps/21869/man).
  - "SCALANCE XM-400/XR-500 Command Line Interface"
    You will find this configuration manual on the Internet pages of Siemens Industry Online Support in SCALANCE XM-400 (https://support.industry.siemens.com/cs/ww/en/ps/15315/man) and SCALANCE XR-500 (https://support.industry.siemens.com/cs/ww/en/ps/15317/man).

Availability

The availability of the settings depends

- on the port type
- on the configuration mode

SCALANCE X distinguishes the following port types:

- Switch port
- Router port

The following configuration modes are available:

- Offline configuration
  You can configure a PC station the first time offline. In this mode only the settings that require no connection to the device are available.
- Online configuration mode
  If there is a connection to the device, there are additional pages available in the Inspector window. These pages include the note "This page is only available if there is an online connection to the device". With some settings, the additional information "only available online" is included.

Creating and deleting an entry

As an example, an entry will be created and deleted on the Syslog client. The procedure is valid on all pages.

Creating a new entry

1. Select the device in the network or device view.
2. Open the properties of the device in the Inspector window.
3. In the Inspector window go to System > Syslog client.
4. Enter the IP address for the Syslog server.
5. Click in the table.
6. Select the entry "New entry" in the shortcut menu.
   A new entry is created in the table.

Deleting an entry

1. Select the required entry in the table.
2. Select the "Delete" entry in the shortcut menu.

Buttons you require often

- **Refresh the display with "Refresh"**
  Pages that display current parameters have a "Refresh" button at the lower edge of the page. Click this button to request up-to-date information from the device for the current page.
- **Save settings for all ports with "Transfer to table"**
  Pages on which you can configure several ports have 2 tables. In the first table, you can make the settings for all ports and transfer these to the second table. In the last column of the first table, there is the "Transfer to table" button. Click this button to save the settings you have made for all ports.

Assigning the IP address

Configuration options

When shipped and after restoring to the factory settings, the device does not have an IP address.

The following options are available to assign an IP address to the device:

- DHCP (default setting)
- Primary Setup Tool
  For further information, refer to the configuration manual "Primary Setup Tool". You will find the MIB at Siemens Industry Automation and Drives Service & Support on the Internet under the entry ID 19440762 (http://support.automation.siemens.com/WW/view/en/19440762).
- STEP 7
- CLI via the serial interface
  For further information, refer to the configuration manual "SCALANCE XM-400/XR-500 Command Line Interface". You will find the configuration manual on the Internet pages of Siemens Industry Automation and Drives Service & Support (http://support.automation.siemens.com/WW/view/en/48805144/133300).

Requirement

- "Set IP address on device" is enabled in the properties of the device. You will find more detailed information in "Addressing PROFINET devices".

Configuring a SCALANCE device as an IO device

With a SCALANCE device configured as a PROFINET IO device and assigned to an IO controller, the functions "Compile" and "Download to device" only download the data to the SCALNCE device that can also be configured in Web Based Management (WBM) (System, Layer 2, Layer 3, Security).

If you want to use the "Compile" or "Download to device" functions for the PROFINET IO device data of the device, first select the assigned IO controller.

Uploading the configuration data with a PROFINET IO device

With a SCALANCE device that is assigned to a PROFINET IO controller, you must distinguish between the data of the PROFINET IO device configuration and the SCALANCE configuration:

- The PROFINET IO device configuration can only be supplied by the PROFINET IO controller.
- The SCALANCE configuration relates only to the data that is configurable in the Web Based Management (WBM) (System, Layer 2, Layer 3, Security).

Follow the steps below to load the entire configuration of a SCALANCE device:

1. Download the PROFINET IO device configuration via the PROFINET IO controller:
   - Select the PROFINET IO controller in the project tree.
   - Select the command "Online" > "Load device as new station (hardware and software)" in the toolbar.
   - See also the section "Loading project data from a device"
2. Load the SCALANCE configuration as a local upload via the device:
   - Select the SCALANCE device in the network view.
   - In the shortcut menu of the selected device select the command "SCALANCE configuration > Upload to PG/PC".
   - See also section "Configuring deevices".

Show information

Versions

Note

The page is only available if there is an online connection to the device.

This page shows the versions of the hardware and software of the device.

Description of the displayed values

Table 1 has the following columns:

- **Hardware**
  - Basic device
    Shows the basic device.
  - PX.X
    X.X = port in which the SFP module is inserted.
  - SlotX
    X = slot number: Module plugged into this slot.
- **Name**
  Shows the name of the device or module.
- **Revision**
  Shows the hardware version of the device.
- **Article number**
  Shows the order number of the device or described module.

Table 2 has the following columns:

- Software
  - Firmware
    Shows the current firmware version. If a new firmware file was downloaded and the device has not yet restarted, the firmware version of the downloaded firmware file is displayed here. After the next restart, the downloaded firmware is activated and used.
  - Bootloader
    Shows the version of the boot software stored on the device.
- Description
  Shows the short description of the software.
- Version
  Shows the version number of the software version.
- Date
  Shows the date on which the software version was created.

I&M

Note

The page is only available if there is an online connection to the device.

This page contains information about device-specific vendor and maintenance data such as the article number, serial number, version number etc. You cannot configure anything on this page.

Description of the displayed values

The table has the following rows:

- Manufacturer ID
  Shows the manufacturer ID.
- Article number
  Shows the article number.
- Serial number
  Shows the serial number.
- Hardware revision
  Shows the hardware version.
- Software version
  Shows the software version.
- Revision counter
  Shows the revision counter: Counter for revisions since the initial commissioning
- Revision Date
  Date and time of the last revision

- **Function tag**
  Shows the function tag (plant designation) of the device. The plant designation (HID) is created during configuration of the device with HW Config of STEP 7.
- **Location tag**
  Shows the location tag of the device. The location identifier (LID) is created during configuration of the device with HW Config of STEP 7.
- **Date**
  Shows the date created during configuration of the device with HW Config of STEP 7.
- **Description**
  Shows the description created during configuration of the device with HW Config of STEP 7.

ARP / neighbor table

ARP table

Note

The page is only available if there is an online connection to the device.

Assignment of MAC address and IP address

With the Address Resolution Protocol (ARP), there is a unique assignment of MAC address to IPv4 address. This assignment is kept by each network node in its own separate ARP table. The page shows the ARP table of the device.

Description of the displayed values

The table has the following columns:

- **Interface**
  Shows the interface via which the row entry was learnt.
- **MAC Address**
  Shows the MAC address of the destination or source device.
- **IP address**
  Shows the IP address of the target device.
- **Media Type**
  Shows the type of connection.
  - Dynamic
    The device recognized the address data automatically.
  - Static
    The addresses were entered as static addresses.

IPv6 Neighbor Table

Assignment of MAC address and IPv6 address

Via the IPv6 neighbor table, there is a unique assignment of MAC address to IPv6 address. This assignment is kept by each network node in its own separate neighbor table.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Interface**
  Displays the interface via which the row entry was learnt.
- **MAC Address**
  Shows the MAC address of the destination or source device.
- **IP Address**
  Shows the IPv6 address of the destination device.
- **Media Type**
  Shows the type of connection.
  - Dynamic
    The device recognized the address data automatically.
  - Static
    The addresses were entered as static addresses.

Log table

Logging events

Note

The page is only available if there is an online connection to the device.

The device allows you to log events that occur, some of which you can specify in " System > Events". This, for example, allows you to record when an authentication attempt failed or when the connection status of a port has changed. The content of the events log table is retained even when the device is turned off.

Description of the displayed values

- Severity Filters

Note

For each severity, a maximum of 400 entries in the table are possible. If the maximum number of entries is reached for a severity, the oldest entries of this severity are overwritten in the table. The table remains permanently in the memory

You can filter the entries in the table according to severity. Select the required entries in the check boxes above the table.

– Info
  Information

– When this parameter is enabled, all entries of the category "Info" are displayed.

– Warning
  Warning
  When this parameter is enabled, all entries of the category "Warning" are displayed.

– Critical
  Critical
  When this parameter is enabled, all entries of the category "Critical" are displayed.

The table has the following columns:

– Restart
  Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.

– System Up Time
  Shows the time the device has been running since the last restart when the described event occurred.

– System Time
  Shows the date and time when the described event occurred.
  If the system time is set, the time is also displayed at which the event occurred.

– Severity
  Shows the severity of the message.

– Log Message
  Displays a brief description of the event that has occurred. You will find the list of possible messages in Appendix D of the configuration manual.

Fault

Error status

Note

The page is only available if there is an online connection to the device.

if an error occurs, it is shown on this page. On the device, errors are indicated by red fault LED lighting up.

Internal errors of the device and errors that you configure on the following pages are indicated:

- System > Events"
- "System" > Fault Monitoring"

Errors of the "Cold/Warm Start" event can be deleted by a confirmation.

The calculation of the time of an error always begins after the last system start.

If there are no errors present, the fault LED switches off.

Description of the displayed values

- **No. of Signaled Faults**
  Shows the number of reported faults.
- **Reset Counter**
  Click the "Reset Counter" button to reset the counter. The counter is also reset when there is a restart.

The table contains the following columns:

- **Fault Time**
  Shows the time the device has been running since the last restart when the described error/fault occurred.
- **Fault Description**
  Displays a brief description of the fault/error that has occurred.
- **Clear Fault State**
  If the "Clear Fault State" button is enabled, you can delete the fault.
- "Reset Counters" button
  Click "Reset Counters" to reset all counters. The counters are reset by a restart.

Redundancy

Spanning Tree

Note

The page is only available if there is an online connection to the device.

The page shows the current information about the Spanning Tree and the settings of the root bridge.

Description of the displayed values

- **Spanning Tree Mode**
  Shows the set mode. You specify the mode in "Layer 2 > Spanning Tree > General".
  The following values are possible:
  - '-'
  - STP
  - RSTP
  - MSTP
- **Instance ID**
  Shows the number of the instance. The parameter depends on the configured mode.
- **Bridge Priority / Root Priority**
  Which device becomes the root bridge is decided by the bridge priority. The bridge with the highest priority (in other words, with the lowest value for this parameter) becomes the root bridge. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames. The value for the bridge priority is a whole multiple of 4096.
- **Bridge address / root address**
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Cost**
  Shows the path costs from the device to the root bridge.
- **Bridge Status**
  Shows the status of the bridge, e.g. whether or not the device is the root bridge.
- **Regional root priority** (available only with MSTP)
  For a description, see Bridge priority / Root priority.
- **Regional root address** (available only with MSTP)
  Shows the MAC address of the device.
- **Regional Root Cost** (only available with MSTP)
  The path costs from this device to the regional root bridge.

The table has the following columns:

- **Port**
  Shows the port via which the device communicates.
- **Role**
  Shows the status of the port. The following values are possible:
  - Disabled
    The port was removed manually from the spanning tree and will no longer be taken into account by the spanning tree.
  - Designated
    The ports leading away from the root bridge.
  - Alternate
    The port with an alternative route to a network segment
  - Backup
    If a switch has several ports to the same network segment, the "poorer" port becomes the backup port.
  - Root
    The port that provides the best route to the root bridge.
  - Master
    This port points to a root bridge located outside the MST region.
- **Status**
  Displays the current status of the port. The values are only displayed. The parameter depends on the configured protocol. The following statuses are possible:
  - Discarding
    The port receives BPDU frames. Other incoming or outgoing frames are discarded.
  - Listening
    The port receives and sends BPDU frames. The port is involved in the spanning tree algorithm. Other outgoing and incoming frames are discarded.
  - Learning
    The port actively learns the topology; in other words, the node addresses. Other outgoing and incoming frames are discarded.
  - Forwarding
    Following the reconfiguration time, the port is active in the network. The port receives and sends data frames.
- **Oper. Version**
  Shows which Spanning Tree protocol is used by the port.
- **Priority**
  If the path calculated by the spanning tree is possible over several ports of a device, the port with the highest priority (in other words the lowest value for this parameter) is selected. A value between 0 and 240 can be entered for the priority in steps of 16. If you enter a value that cannot be divided by 16, the value is automatically adapted.

- **Path Cost**
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected. If several ports of a device have the same value, the port with the lowest port number will be selected.
  If the value "Cost Calc." is "0", the automatically calculated value is shown. Otherwise, the value of "Cost Calc." is displayed.
  The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000.

  You configure the "Cost Calc." on the pages "Layer 2 > Spanning Tree > CIST Port" and "Layer 2 > Spanning Tree > MST Port".
- **Edge type**
  Shows the type of the connection. The following values are possible:
  - Edge Port

    An edge port is connected to this port.
  - No Edge Port
    There is a spanning tree device at this port.
- **P.t.P. Type**
  Shows the type of point-to-point link. The following values are possible:
  - P.t.P.
    With half duplex, a point-to-point link is assumed.
  - Shared Media

    With a full duplex connection, a point-to-point link is not assumed.

VRRP statistics

Introduction

Note

The page is only available if there is an online connection to the device.

This page shows the statistics of the VRRP protocol and all configured virtual routers.

Description of the displayed values

The following fields are displayed:

- VRID errors
  Shows how many VRRP frames containing an unsupported VRID were received.
- Version errors
  Shows how many VRRP frames containing an invalid version number were received.
- Checksum errors
  Shows how many VRRP frames containing an invalid checksum were received.

The table has the following columns:

- Interface
  Interface to which the settings relate.
- VRID
  Shows the ID of the virtual router.
- Change to master status
  Shows how often this virtual router changed to the "Master" role.
- Advertisements recd.
  Shows how often a VRRP frame was received that contained a bad address list.
- Advertisement Interval Errors
  Shows how many bad VRRP packets were received whose interval does not match the value set locally.
- IP TTL errors
  Shows how many bad VRRP packets were received whose TTL (Time to live) value in the IP header is incorrect.
- Prio 0 received
  Shows how many VRRP packets with priority 0 were received. VRRP packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- Prio 0 sent
  Shows how many VRRP packets with priority 0 were sent. Packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- Invalid Type
  Shows how many bad VRRP packets were received whose authentication type was not 0. Type 0 means "no authentication".
- Address List Errors
  Shows how many bad VRRP packets were received whose address list does not match the locally configured list.
- Invalid Auth. Type
  Shows how many bad VRRP packets were received whose authentication type was not type 0. Type 0 means "no authentication".

- Auth. Type Mismatch
  Shows how many bad VRRP frames were received whose authentication type does not match.
- Frame length error
  Shows how many bad VRRP frames were received whose length is not correct.

VRRP Statistics

Introduction

This page shows the statistics of the VRRPv3 protocol and all configured virtual routers.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The following fields are displayed:

- VRID Errors
  Shows how many VRRPv3 packets containing an unsupported VRID were received.
- Version Errors
  Shows how many VRRPv3 packets containing an invalid version number were received.
- Checksum Errors
  Shows how many VRRPv3 packets containing an invalid checksum were received.

The table has the following columns:

- Interfaces
  Interface to which the settings relate.
- VRID
  Shows the ID of the virtual router. Valid values are 1 ... 255.
- Type
  Shows the version of the IP protocol.
- Become Master
  Shows how often this virtual router changed to the "Master" status.
- Advertisements Received
  Shows how many VRRPv3 packets were received.
- Advertisement Interval Errors
  Shows how many bad VRRPv3 packets were received whose interval does not match the value set locally.
- IP TTL Errors
  Shows how many bad VRRPv3 packets were received whose TTL (Time to live) value in the IP header is incorrect.

- **Prio 0 received**
  Shows how many VRRPv3 packets with priority 0 were received. VRRPv3 packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- **Prio 0 sent**
  Shows how many VRRPv3 packets with priority 0 were sent. Packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- **Invalid Type**
  Shows how many bad VRRPv3 packets were received whose value in the "Type" field of the IP header is invalid.
- **Address List Errors**
  Shows how many bad VRRPv3 packets were received whose address list does not match the locally configured list.
- **Packet Length Errors**
  Shows how many bad VRRPv3 packets were received whose length is not correct.

Ring Redundancy

Note

The page is only available if there is an online connection to the device.

Automatic Redundancy Detection (ARD) is the default when the device ships. If you want to use the previous High Speed Redundancy Protocol (HRP), HRP must be configured.

- Reconfiguration time of the frame traffic following a failover in MRP: 200 ms
- Reconfiguration time of the frame traffic following a failover in HRP: 300 ms

This page informs you about the values currently set for the device.

Description of the displayed values

The following fields are displayed:

- **Redundancy Function**
  The "Redundancy Function" column shows the role of the device within the ring:
  - no Ring Redundancy
    The IE switch is operating without redundancy function.
  - HRP Client
    The IE switch is operating as an HRP client.
  - HRP manager
    The IE switch is operating as an HRP manager.
  - MRP Client
    The IE switch is operating as an MRP client.
  - MRP manager
    The IE switch is operating as an MRP manager. Using STEP 7, the role "Manager" was set for the device.
  - MRP Auto-Manager
    The IE switch is operating as an MRP manager. Using WBM or CLI the role "MRP Auto-Manager" or using STEP 7 the role "Manager (Auto)" was set.
- **RM Status**
  The "RM Status" column shows whether or not the IE switch is operating as redundancy manager and whether it has opened or closed the ring in this role.
  - Passive
    The IE switch is operating as redundancy manager and has opened the ring; in other words, the line of switches connected to the ring ports is operating problem free. The "Passive" status is also displayed if the IE switch is not operating as the redundancy manager (Redundancy manager disabled).
  - Active
    The IE switch is operating as redundancy manager and has closed the ring; in other words, the line of switches connected to the ring ports is interrupted (problem). The redundancy manager connects its ring ports through and restores an uninterrupted linear topology.
  - "-"
    The redundancy function is disabled.
- **Observer Status**
  Shows the current status of the observer.
- **Ring Port 1 and Ring Port 2**
  The "Ring Port 1" and "Ring Port 2" columns show the ports being used as ring ports.

- **No. of Changes to RM Active State**
  Shows how often the device as redundancy manager switched to the active status, i.e. closed the ring.
  If the redundancy function is disabled or the device is not the HRP/MRP manager, the text "Redundancy manager disabled" appears.
- **Max. delay of the RM test frames [ms]**
  Shows the maximum delay for test frames of the redundancy manager.
  If the redundancy function is disabled or the device is not the HRP/MRP manager, the text "Redundancy manager disabled" appears.

Standby

Note

The page is only available if there is an online connection to the device.

On this page, you obtain information about the status of the device in terms of standby redundancy. The text boxes on this page are read-only.

Note

Device with the higher MAC address becomes master

When linking HRP rings redundantly, two devices are always configured as a master/slave pair. This also applies to interrupted HRP rings = linear buses. When operating normally, the device with the higher MAC address adopts the role of master. This type of assignment is important in particular when a device is replaced. Depending on the MAC addresses, the previous device with the slave function can take over the role of the standby master.

The Standby tab shows the status of the standby function:

Description of the displayed values

The following fields are displayed:

- **Standby ports**
  Shows the standby port.
- **Standby name**
  Standby connection name

- Standby Function
  Shows whether the standby connection is activated or deactivated.
  - Master
    The device has a connection to the partner device and is operating as master. In normal operation, the standby port of this device is active.
  - Slave
    The device has a connection to the partner device and is operating as slave. In normal operation, the standby port of this device is inactive.
  - Disabled
    The standby link is disabled. The device is operating neither as master nor slave. The port configured as a standby port works as a normal port without standby function.
  - Connection establishment
    No connection has yet been established to the partner device. The standby port is inactive. In this case, either the configuration on the partner device is inconsistent (for example incorrect connection name, standby link disabled) or there is a physical fault (for example device failure, link down).
  - Connection lost
    The existing connection to the partner device has been lost. In this case, either the configuration on the partner device was modified (for example a different connection name, standby link disabled) or there is a physical fault (for example device failure, link down).
- Standby status
  Shows the status of the standby port:
  - Active
    The standby port of this device is active; in other words is enabled for frame traffic.
  - Passive
    The standby port of this device is inactive; in other words is blocked for frame traffic.
  - "-"
    The standby function is disabled.
- No. of Changes to Standby Active State
  Shows how often the IE switch has changed the standby status from "Passive" to "Active". If the connection of a standby port fails on the standby master, the IE switch changes to the "Active" status. If the standby function is disabled, the text "Standby Disabled" appears in this box.
  Following each restart on the device, the counters are automatically reset.

Link Check

Monitoring optical connections in the ring

The page shows the following information on the link check:

- The ports on which you can enable Link Check
- The current status
- The statistics of sent and received Link Check frames of the monitored connections.

Description of the displayed values

The following fields are displayed:

- **Port**
  Shows the port to which the following information relates. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Link Check**
  Shows whether the Link Check function is enabled or disabled.
- **OperState**
  Shows the status of the Link Check function. The following statuses are possible:
  - Disabled
    The function is disabled.
  - Enabled
    The function is enabled. The connection partner has not yet confirmed the monitoring.
  - Running
    The function is enabled. The connection monitoring is enabled. The outgoing and incoming test frames are counted and matched up.
  - Faults
    The function is enabled. Link Check has detected a fault on the monitored section and turned off the port.
- **Frames in**
  Shows how many Link Check test frames were received.
- **Frames out**
  Shows how many Link Check test frames were sent

MRP Interconnection

Redundant linking of rings

This page shows the configured MRP Interconnections and detailed information on the individual parameters.

Description of the displayed values

The following boxes are displayed:

- **Interconnection ID**
  The MRP Interconnection.
- **Interconnection Name**
  The name of the MRP Interconnection.
- **Interconnection Port**
  The port that is used for the MRP Interconnection.
- **Port Status**
  Shows whether the port is on or off. Data traffic is possible only over an enabled port.

- Operative Role
  Shows the role of the device. With the "Client" role, the position of the client is shown in addition. The available options are as follows:
  - Disabled
  - Manager
  - Primary Client
  - Secondary Client
- Connection Status
  The status of the MRP Interconnect domain. The available options are as follows:
  - Disabled
  - Not defined
  - Open
    The connection between the primary devices is open, the rings are connected via the secondary connection.
  - Close
    There is a connection between the primary devices.
- Open Count
  Shows how often the "Open" status has occurred since the last counter reset.
- Open Time
  The time since the last change to the connection status.
- Reset Counter
  Click "Reset Counter" to reset the counter. The counter is reset when there is a restart.

Ethernet Statistics

Interface statistics

Information from the MIB

The page shows the statistics from the interface table of the Management Information Base (MIB).

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- In Octet
  Shows the number of received bytes.
- Out Octet
  Shows the number of sent bytes.
- In Unicast
  Shows the number of received unicast frames.
- In Non Unicast
  Shows the number of received frames that are not of the type unicast.
- Out Unicast
  Shows the number of sent unicast frames.
- Out Non Unicast
  Shows the number of sent frames that are not of the type unicast.
- In Errors
  Shows the number of all possible RX errors, refer to the tab "Packet Error".

Description of the button

Reset Counter

Click the "Reset Counter" button to reset all counters. The counters are also reset by a restart.

Frame length

Frames sorted by length

This page displays how many frames of which length were sent and received at each port. You cannot configure anything on this page.

Note

The page is only available if there is an online connection to the device.

The displayed values are transferred by RMON.

On the page "Layer 2 > RMON > Statistics", you can set the ports for which values will be displayed.

Description of the displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.

  Note
  Display of frame statistics

  In the statistics relating to frame lengths, note that both incoming and outgoing frames are counted.

- **Frame lengths**
  The other columns after the port number contain the absolute numbers of frames according to their frame length.
  The following frame lengths are distinguished:
  - 64 bytes
  - 65 - 127 bytes
  - 128 - 255 bytes
  - 256 - 511 bytes
  - 512 - 1023 bytes
  - 1024 - max.

Note
Data traffic on blocked ports

For technical reasons, data packets can be indicated on blocked ports.

Description of the button

Reset Counter

Click the "Reset Counter" button to reset all counters. The counters are also reset by a restart.

Frame type

Received frames sorted by type

This page displays how many frames of the type "unicast", "multicast", and "broadcast" were received at each port.

You cannot configure anything on this page.

The displayed values are transferred by RMON. On the page "Layer 2 > RMON > Statistics", you can set the ports for which values will be displayed.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Unicast / Multicast / Broadcast**
  The other columns after the port number contain the absolute numbers of the incoming frames according to their frame type "unicast", "multicast" and "broadcast".

Description of the button

Reset Counter

Click the "Reset Counter" button to reset all counters. The counters are also reset by a restart.

Packet Errors

Received bad frames

This page shows how many bad frames were received per port.

You cannot configure anything on this page.

The displayed values are transferred by RMON. On the page "Layer 2 > RMON > Statistics", you can set the ports for which values will be displayed.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations.
- **Error types**
  The other columns after the port number contain the absolute numbers of the incoming frames according to their error type.
  In the columns of the table, a distinction is made according to the following error types:
  - CRC

    Packets whose content does not match the CRC checksum.
  - Undersize
    Packets with a length less than 64 bytes.
  - Oversize
    Frames discarded because they were too long.
  - Fragments

    Packets with a length less than 64 bytes and a bad CRC checksum.
  - Jabbers
    VLAN-tagged packets with an incorrect CRC checksum that were discarded because they were too long.
  - Collisions
    Collisions that were detected.

Description of the button

Reset Counter

Click the "Reset Counter" button to reset all counters. The counters are also reset by a restart.

History

Samples of the statistics

The page shows samples from each port with information from the statistics.

On the page "Layer 2 > RMON > History", you can set the ports for which samples will be taken.

Note

The page is only available if there is an online connection to the device.

Settings

- **Port**
  Select the port for which the history will be displayed.

Description of the displayed values

- **Entries**
  Maximum number of samples that can be saved at the same time.
- **Interval [s]**
  Interval after which the current status of the statistics will be saved as a sample.

The table has the following columns:

- **Sample**
  Number of the sample
- **Time for the Sample**
  System up time at which the sample was taken.
- **Unicast**
  Number of received unicast frames.
- **Multicast**
  Number of received multicast frames.
- **Broadcast**
  Number of received broadcast frames.
- **CRC**
  Number of frames with a bad CRC checksum.
- **Undersize**
  Number of frames that are shorter than 64 bytes.
- **Oversize**
  Number of frames discarded because they were too long.
- **Fragments**
  Number of frames that are shorter than 64 bytes and have a bad CRC checksum.
- **Jabbers**
  Number of frames with a VLAN tag that have a bad CRC checksum and will be discarded because they are too long.
- **Collisions**
  Number of collisions of received frames.
- **Utilization**
  Utilization of the port during a sample.

Unicast

This page shows the current content of the unicast filter table. This table lists the source addresses of unicast address frames. Entries can be made either dynamically when a node sends a frame to a port or statically by the user setting parameters.

Description of the displayed values

- VLAN ID
  Shows the VLAN-ID assigned to this MAC address.
- MAC Address
  Shows the MAC address of the node that the device has learned or the user has configured.
- Status
  Shows the status of each address entry:
  - Learnt
    The specified address was learned by receiving a frame from this node and will be deleted when the aging time expires if no further packets are received from this node.

    Note

    If there is a link down, learned MAC entries are deleted.
  - Static
    Configured by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the switch is restarted.
- Port
  Shows the port via which the node with the specified address can be reached. Frames received by the device whose destination address matches this address will be forwarded to this port.

Multicast

Multicast

This table shows the multicast frames currently entered in the multicast filter table and their destination ports. The entries can be dynamic (the device has learned them) or static (the user has set them).

Description of the displayed values

- VLAN ID
  Shows VLAN ID of the VLAN to which the MAC multicast address is assigned.
- MAC Address
  Shows the MAC multicast address that the device has learned or the user has configured.

- Status
  Shows the status of each address entry. The following information is possible:
  - Static
    The address was entered statically by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the device is restarted. These must be deleted by the user.
  - IGMP
    The destination port for this address was obtained by IGMP configuration.
  - MLD
    The destination port for this address was determined by MLD configuration.
  - GMRP
    The destination port for this address was registered by a received GMRP frame.
- List of ports
  There is a column for each slot. Within a column, the multicast group to which the port belongs is shown:
  - M (Member)
    Multicast frames are sent via this port.
  - R (Registered)
    Member of the multicast group, registration was by an GMRP frame.
  - I (IGMP/MLD)
    Member of the multicast group, registration was by an IGMP/MLD frame.
  - –
    Not a member of the multicast group. No multicast frames with the defined multicast MAC address are sent via this port.
  - F (Forbidden)
    Not a member of the multicast group. Moreover, this address must not be an address learned dynamically with GMRP or IGMP/MLD.

IGMP Groups

Status of the multicast filter table

This table shows the multicast groups currently entered in the filter table, configured and joined.

Description of the displayed values

This table contains the following columns:

- Interface
  Shows the IGMP interface.
- Group Address
  Shows the address of the multicast group that the device has learned or the user has configured.

- **Mode**
  Shows whether the recipient has specified that a multicast from a certain source should be received or rejected.
  - '-'
    IGMPv1 or IGMPv2 is used or the recipient has not given any source preference or excluded any source.
  - Include
    The multicast should be received from certain sources.
  - Exclude
    The multicast should not be received from certain sources.
- **Status**
  Shows the status of each entry. The following information is possible:
  - Static
    The multicast group was configured statically.
  - Dynamic
    The multicast group was learned via IGMP configuration.

IGMP Groups

Status of the multicast filter table

This table shows the multicast groups currently entered in the filter table, configured and joined.

Description of the displayed values

This table contains the following columns:

- **Interface**
  Shows the IGMP interface.
- **Group Address**
  Shows the address of the multicast group that the device has learned or the user has configured.

- **Mode**
  Shows whether the recipient has specified that a multicast from a certain source should be received or rejected.
  - '-'
    IGMPv1 or IGMPv2 is used or the recipient has not given any source preference or excluded any source.
  - Include
    The multicast should be received from certain sources.
  - Exclude
    The multicast should not be received from certain sources.
- **Status**
  Shows the status of each entry. The following information is possible:
  - Static
    The multicast group was configured statically.
  - Dynamic
    The multicast group was learned via IGMP configuration.

LLDP

Note

The page is only available if there is an online connection to the device.

Status of the neighborhood table

This page shows the current content of the neighborhood table. This table stores the information that the LLDP agent has received from connected devices. You set the interfaces via which the LLDP agent receives or sends information in the following section: "Layer 2 > LLDP".

Description of the displayed values

The table contains the following columns:

- **System name**
  System name of the connected device.
- **Device ID**
  Device ID of the connected device. The device ID corresponds to the device name assigned via PST (STEP 7). If no device name is assigned, the MAC address of the device is displayed.
- **Local interface**
  Port at which the device received the information
- **Hold Time**
  An entry remains stored on the device for the time specified here. If the device does not receive any new information from the connected device during this time, the entry is deleted.

- **Capability**
  Shows the capabilities of the connected device:
  - Router
  - Bridge
  - Telephone
  - DOCSIS Cable Device
  - WLAN Access Point
  - Repeater
  - Station
  - Other
- **Port ID**
  Port of the device with which the device is connected.

FMP

Monitoring optical links

With Fiber Monitoring, you can monitor optical links. The table shows the current status of the ports.

You set the values to be monitored on the following page: "Layer 2 > FMP".

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

- **Port**
  Shows the available optical ports.
- **Rx Power State**
  - disabled
    Fiber monitoring is disabled.
  - ok
    The value for the received power of the optical link is within the set limits
  - maint. req.
    Check the link.
    A warning is signaled.
  - maint. dem.
    The link needs to be checked.
    An alarm is signaled and the fault LED is lit.
  - link down
  - The connection is interrupted.
- **RX Power**
  Shows the current value of the received power. The value can have a tolerance of +/- 3 dB. If there is no connection (link down) or fiber monitoring is disabled, "-" is displayed. If fiber monitoring is not enabled on the partner port, the value 0.0 is displayed.
- **Power Loss State**
  To be able to monitor the power loss of the connection, the port requires a link to another port with FM enabled.
  - disabled
    FM is disabled.
  - ok
    The value for the power loss of the optical link is OK.
  - maint. req.
    Check the link.
    A warning is signaled.
  - maint. dem.
    The link needs to be checked.
    An alarm is signaled and the fault LED is lit.
  - idle
    The port has no connection to another port with fiber monitoring enabled.
    If no diagnostics information is received from the optical port of the connection partner for 5 cycles, the fiber monitoring connection is assumed to be interrupted. A cycle lasts 5 seconds.
- **Power Loss [dB]**
  Shows the current value of the power loss. The value can have a tolerance of +/- 3 dB. If there is no connection (link down), fiber monitoring is disabled or the partner port does not support fiber monitoring, "-" is displayed.

IPv4 routing

Routing table

Overview

This page shows the routes currently being used.

Description of the displayed values

The table has the following columns:

- **Destination network**
  Shows the destination address of this route.
- **Subnet mask**
  Shows the subnet mask of this route.
- **Gateway**
  Shows the gateway for this route.
- **Interface**
  Shows the interface for this route.
- **Metric**
  Shows the metric of the route. The higher value, the longer the packets require to their destination.
- **Routing protocol**
  Shows the routing protocol from which the entry in the routing table originates. The following entries are possible:
  - Connected: Connected routes
  - Static: Static routes
  - RIP: Routes via RIP
  - OSPF: Routes via OSPF
  - other: Other routes

OSPFv2 interfaces

Overview

Note

The page is only available if there is an online connection to the device.

This page shows the configuration of the OSPF interface.

Description of the displayed values

The table has the following columns:

- **IP Address**
  Shows the IPv4 address of the OSPF interface
- **Area ID**
  Shows the Area ID to which the OSPF interface belongs.
- **Interface Status**
  Shows the status of the WLAN interface:
  - Down
    The interface is not available.
  - Loopback
    Loopback interface
  - Waiting
    Startup and negotiation of the interface.
  - Point to Point
    Point-to-point connection
  - Designated Router
    The router is a designated router and generates a network LSA.
  - Backup D. Router
    The router is backup for the designated router.
  - Other D. Router
    The interface is started up. The router is neither a designated router nor a backup designated router.
- **OSPF status**
  Shows the status of OSPF.
  - Enabled: OSPF is enabled on the interface.
  - Disabled: OSPF is disabled on the interface.
- **Designated Router**
  Shows the IPv4 address of the designated router for this OSPF interface.
- **Backup Designated Router**
  Shows the IPv4 address of the designated backup router for this OSPF interface.
- **Events**
  Shows the number of status changes of OSPF.

OSPFv2 neighbors

Overview

Note

The page is only available if there is an online connection to the device.

This page shows the dynamically detected neighbor routers in the relevant networks.

Description of the displayed values

The table has the following columns:

- **IP Address**
  Shows the IPv4 address of the neighbor router in this network.
- **Router ID**
  Shows the ID of the neighbor router. The two addresses can match.
- **Status**
  Shows the status of the neighbor router. The status can adopt the following values:
  - unknown
    Status of the neighbor router is unknown.
  - down
    The neighbor router cannot be reached.
  - attemptand init
    Short-lived statuses during initialization
  - two-way
    Two-way receipt of Hello packets. Specification of the designated router and the designated backup router.
  - exchangestart, exchange and loading
    Status during exchange of the LSAs
  - full
    The database is complete and synchronized within the area. The routes can now be detected.

    Note

    Normal status

    If the partner router is a designated router or a designated backup router, the status is "full". Otherwise the status is "two-way".
- **Assoc. Area Type**
  Shows the area type via which the neighbor-neighbor relation is maintained. The following area types exist:
  - Normal
  - Stub
  - NSSA
  - Backbone
- **Priority**
  Shows the priority of the neighbor router. This is only significant when selecting the designated router on a network. For virtual neighbor routers, this information is irrelevant.
- **Hello Suppr.**
  Shows the suppressed Hello packets to the neighbor router. This field normally displays "no".

- Hello Queue
  Shows the length of the queue with Hello packets still to be transmitted.
- Events
  Shows the number of status changes.

OSPFv2 virtual neighbors

Overview

Note

The page is only available if there is an online connection to the device.

This page shows the configured virtual neighbor routers.

Description of the displayed values

The table has the following columns:

- IP Address
  Shows the IPv4 address of the virtual neighbor router in this network.
- Router ID
  Shows the router ID of the virtual neighbor router.
- Status
  Shows the status of the neighbor router. The status can adopt the following values:
  - unknown
    Status of the neighbor router is unknown.
  - down
    The neighbor router cannot be reached.
  - attemptand init
    Short-lived statuses during initialization
  - two-way
    Two-way receipt of Hello packets. Specification of the designated router and the designated backup router.
  - exchangestart, exchange and loading
    Status during exchange of the LSAs
  - full
    The database is complete and synchronized within the area. The routes can now be detected.

    Note

    Normal status

    If the partner router is a designated router or a designated backup router, the status is "full". Otherwise the status is "two-way".

- Transit Area ID
  Shows the ID of the area via which the virtual neighborhood relation exists.
- Hello Suppr.
  Shows whether there are suppressed Hello packets to the virtual neighbor router.
  – no: There are no suppressed Hello packets (default)
  – yes: There are suppressed Hello packets.
- Hello Queue
  Shows the length of the queue with Hello packets still to be transmitted.
- Events
  Shows the number of status changes.

OSPFv2 LSDB

Overview

Note

The page is only available if there is an online connection to the device.

The link state database is the central database for managing all links within an area. It consists of the link state advertisements (LSAs). The most important data of these LSAs is shown on the this WBM page.

Description of the displayed boxes

The table has the following columns:

- **Area ID**
  Shows the ID of the area to which the LSA belongs. If the LSA is an external connection, '-' is displayed.
- **LSA type**
  Shows the LSA type. The following values are possible:
  - unknown
    LSA type is unknown.
  - Router
    The router LSA (Type 1) is sent by the OSPF router within an area. The LSA contains information about the status of all router interfaces.
  - Network
    The network LSA (Type 2) is sent by the designated router within an area. The LSA contains a list of routers connected to the network.
  - NSSA External
    The NSSA external LSA (Type 7) is sent by the NSSA-ASBR within an NSSA. The NSSA-ASBR receives LSAs of Type 5 and converts the information to LSAs of Type 7. The NSSA router can forward these LSAs within an NSSA.
  - Summary
    The summary LSA (Type 3) is sent by the ABR within an area. The LSA contains information about routes to other networks.
  - AS Summary
    The AS summary LSA (Type 4) is sent by the area border router within an area. The LSA contains information about routes to other autonomous systems.
  - AS External
    The AS external LSA (Type 5) is sent by the AS border router within an autonomous system. The LSA contains information about routes from one network to another.
- **Link State ID**
  Shows the ID of the LSA.
- **Router ID**
  Shows the ID of the router that sent this LSA.
- **Sequence number**
  Shows the sequence number of the LSA. Each time an LSA is renewed, this sequence number is incremented by one.

RIPv2 statistics

Overview

Note

The page is only available if there is an online connection to the device.

This page shows the statistics of the RIP interface.

Description of the displayed values

The table has the following columns:

- **IP address**
  Shows the IP address of the RIPv2 interface
- **Bad packets**
  Number of received RIP packets that were deleted and therefore ignored.
- **Bad Routes**
  Number of routes of valid RIP packets that could not be taken into consideration.
- **Updates Sent**
  Shows how often the router has sent its routing table to its neighbor routers.

NAT Translations

Overview

This page displays the active NAT connections.

Description of the displayed values

The table has the following columns:

- **Interface**
  Shows the IP interface.
- **Inside Local Address**
  Shows the actual address of the device that should be reachable from external.
- **Inside Local Port**
  Shows the port that is assigned to the Inside Local Address.
- **Inside Global Address**
  Shows the address at which the device can be reached from external.
- **Inside Global Port**
  Shows the port that is assigned to the Inside Global Address.
- **Outside Local/Global Address**
  Shows the address of the communications partner.
- **Outside Local/Global Port**
  Displays the port of the external communications partner.
- **Last Use Time**
  Shows the time at which the last packet was transferred.

PIM Interfaces

Overview

This page shows the PIM interfaces.

Description of the displayed values

The table has the following columns:

- Interface
  Shows the PIM interface.
- Address
  Shows the IP address of the interface.
- Query Interval
  Every PIM router or every PIM interface sends Hello packets cyclically at the specified intervals. This allows every PIM router to get to know its neighbors and the designated router can be specified.
- DR Address
  Shows the IP address of the designated router.
- DR Priority
  Shows the DR priority of the interface.

PIM Neighbors

Overview

This page shows the PIM neighborhood table.

Description of the displayed values

The table has the following columns:

- Interface
  Shows the PIM interface. The PIM routers are connected together via this interface.
- Neighbor Address
  Shows the IP address of the neighbor.
- DR Priority
  Shows the DR priority of the interface.

PIM Routes

Overview

This page shows the PIM routing table.

Description of the displayed values

Select which PIM routes should be displayed.

- All
- Outgoing

Depending on your selection not all columns are displayed in the table.

The table has the following columns:

- Group Address
  Shows the address of the multicast group.
- Group Mask
  Shows the subnet mask that restricts the multicast band.
- Source Address
  - IP Address
    Shows the source IP address of the multicast packet.
  - *
    (*,G) messages contain the group and the information to pass on the message in the direction of the RP.
    The route is via the RP.
- Incoming Interface
  Shows the interface via which the multicast will be received.
- Outgoing Interface
  Shows the interface via which the multicast will be forwarded.
- Status
  Shows whether or not a multicast group is used.
  - Forwarding
    The multicast group is used.
  - Pruned
    The multicast group is not used.

PIM RPs

Samples of the statistics

This page shows information on rendezvous points.

Description of the displayed values

Select which rendezvous points should be displayed.

- Candidate
  Shows the interfaces of the device that are candidates for rendezvous points.
- Elected (Designated)
  Shows the designated router of the device.
- Static
  Shows the static rendezvous points of the device.

Depending on your selection not all columns are displayed in the table.

The table has the following columns:

- Group Address
  Shows the address of the multicast group.
- Group Mask
  Shows the subnet mask that restricts the multicast band.
- RP Address
  Shows the IP address of the RP.
- RP Priority
  Shows the priority of the interface as RP.
- Bidirectional Multicast
  Shows whether the "bidirectional multicast" functionality is activated or deactivated.

PIM BSRs

Overview

The page shows information on bootstrap routers (BSR).

Description of the displayed values

The page contains the following boxes:

- Address of the selected BSR
  Shows the IP address of the BSR that was selected within a PIM network or a PIM domain.

The table has the following columns:

- Candidate BSR Address
  Shows the interfaces of the device that are candidates for the BSR on the PIM router.
- Candidate BSR Priority
  Shows the priority of the BSR.

MSDP Cache

Overview

This page shows the MSDP partners.

Description of the displayed values

The table has the following columns:

- Source
  Shows the source of the multicast.
- Group
  Shows the multicast group.
- RP
  Shows the MSDP partner.
- Uptime
  Show the remaining time after which an entry is deleted from the SA cache if no SA cache updates are received.

IPv6 routing

Routing table

Overview

This page shows the IPv6 routes currently being used.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- Destination Network
  Shows the destination address of this route.
- Prefix Length
  Shows the prefix length of this route.
- Gateway
  Shows the gateway for this route.
- Interface
  Shows the interface for this route.

- **Metric**
  Shows the metric of the route. The higher value, the longer packets require to their destination.
- **Routing Protocol**
  Shows the routing protocol from which the entry in the routing table originates. The following entries are possible:
  - connected: Connected routes
  - Static: Static routes
  - RIPng: Routes via RIPng
  - OSPFv3: Routes via OSPFv3
  - other: Other routes

OSPFv3 Interfaces

Overview

This page shows the configuration of the OSPFv3 interface.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Interface**
  Shows the interface to which the settings relate.
- **Area ID**
  Shows the Area ID to which the OSPF interface belongs.

- **Interface Status**
  Shows the status of the interface.
  - Down
    The interface is not available.
  - Loop back
    Loop back interface
  - Waiting
    Startup and negotiation of the interface.
  - Point to Point
    Point-to-point connection
  - Designated Router
    The router is a designated router and generates network LSAs.
  - Backup D. Router
    The router is the backup router for the designated router.
  - S D. Router
    The Interface has started up. The router is neither a designated nor a designated backup router.
- **OSPF Status**
  Shows the status of OSPF.
  - Enabled: OSPF is enabled on the interface.
  - Disabled: OSPF is disabled on the interface.
- **Designated Router**
  Shows the IPv6 address of the designated router for this OSPFv3 interface.
- **Backup Designated Router**
  Shows the IPv6 address of the designated backup router for this OSPFv3 interface.
- **Events**
  Shows the number of status changes of OSPFv3.

OSPFv3 neighbors

Overview

This page shows the dynamically detected neighbor routers in the relevant networks.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Interface**
  Displays the OSPFv3 interface via which the neighbor router can be reached.
- **Router ID**
  Shows the ID of the neighbor router.
- **Status**
  Shows the status of the neighbor router. The status can adopt the following values:
  - unknown
    Status of the neighbor router is unknown.
  - down
    The neighbor router cannot be reached.
  - attempt and init
    Short-lived statuses during initialization
  - two-way
    Two-way receipt of Hello packets. Specification of the designated router and the designated backup router.
  - exchangestart, exchange and loading
    Status during exchange of the LSAs
  - full
    The database is complete and synchronized within the area. The routes can now be detected.

    Note

    Normal status

    If the partner router is a designated router or a designated backup router, the status is "full". Otherwise the status is "two-way".
- **Assoc. Area Type**
  Shows the area type via which the neighbor-neighbor relation is maintained. The following area types exist:
  - Standard
  - Stub
  - NSSA
  - Backbone
- **Priority**
  Shows the priority of the neighbor router. This is only significant when selecting the designated router on a network. For virtual neighbor routers, this information is irrelevant.
- **Hello Suppr.**
  Shows the suppressed Hello packets to the neighbor router. This field normally displays "no".

- Retrans Queue
  Shows the length of the queue with Hello packets still to be transmitted.
- Events
  Shows the number of status changes.

OSPFv3 Virtual Neighbors

Overview

This page shows the configured virtual neighbors.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- Interface
  Displays the OSPFv3 interface via which the neighbor router can be reached.
- Router ID
  Shows the ID of the virtual neighbor router.
- Status
  Shows the status of the virtual neighbor router. The status can adopt the following values:
  - unknown
    Status of the neighbor router is unknown.
  - down
    The neighbor router cannot be reached.
  - attempt and init
    Short-lived statuses during initialization
  - two-way
    Two-way receipt of Hello packets. Specification of the designated router and the designated backup router.
  - exchangestart, exchange and loading
    Status during exchange of the LSAs
  - full
    The database is complete and synchronized within the area. The routes can now be detected.

    Note

    Normal status

    If the partner router is a designated router or a designated backup router, the status is "full". Otherwise the status is "two-way".

- Trans. Area-ID
  Shows the ID of the area via which the virtual neighborhood relation exists.
- Hello Suppr.
  Shows the suppressed Hello packets to the neighbor router. This field normally displays "no".
- Hello Queue
  Shows the length of the queue with Hello packets still to be transmitted.
- Events
  Shows the number of status changes.

OSPFv3 AS-Scope LSDB

Overview

The AS-Scope LSDB consists of AS external LSAs. The most important data of these LSAs is shown on this page.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- Link State Type
  - AS External
    The AS external LSA (Type 0x4005) is sent by the AS border router within an autonomous system. The LSA contains information about routes from one network to another.
- Link State ID
  Shows the ID of the LSA.
- Router ID
  Shows the ID of the router that sent this LSA.
- Sequence Number
  Shows the sequence number of the LSA. Each time an LSA is renewed, this sequence number is incremented by one.

OSPFv3 Area-Scope LSDB

Overview

The Area-Scope LSDB is the central database for managing all links within an area. The most important data of the LSAs is shown on this page.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Area ID**
  Shows the ID of the area to which the LSA belongs. If the LSA is an external connection, '-' is displayed.
- **Link State Type**
  Shows the LSA type. The following values are possible:
  - Router
    The router LSA (Type 0x2001) is sent by the OSPF router within an area. The LSA contains information about the status of all router interfaces. However, no longer contains address information. This is contained in the new LSA type 0x2009.
  - Network
    The network LSA (Type 0x2002) is sent by the designated router within an area. The LSA contains a list of routers connected to the network. However, no longer contains address information. This is contained in the new LSA type 2009.
  - Inter-Area Prefix
    the inter-area prefix LSA (Type 0x2003) is sent by the ABR within an area. The LSA contains information about routes to other networks.
  - Inter-Area Router
    The inter-area router LSA (Type 0x2004) is sent by the area border router within an area. The LSA contains information about routes to other autonomous systems.
  - Type 7
    The Type 7 LSA (Type 0x2007) is sent by the NSSA-ASBR within an NSSA. The NSSA-ASBR receives LSAs of Type 0x4005 and converts the information to LSAs of Type 0x2007. The NSSA router can forward these LSAs within an NSSA.
  - Intra-Area Prefix
    The intra-area prefix LSA (Type 0x2009) is only sent within an area. It contains the IPv6 prefixes connected to the router or network.
- **Link State ID**
  Shows the ID of the LSA.

- Router ID
  Shows the ID of the router that sent this LSA.
- Sequence Number
  Shows the sequence number of the LSA. Each time an LSA is renewed, this sequence number is incremented by one.

OSPFv3 Link Scope LSDB

Overview

The link scope LSDB consists of the link LSAs. The most important data of these LSAs is shown on this page.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- Interface
  Shows the OSPFv3 interface to which the link LSA belongs.
- Link State Type
  - Link
    The link LSA (Type 0x2009) is sent by the router to every router linked to it. It contains the link local address of the router and a list with IPv6 prefixes configured on the link.
- Link State ID
  Shows the ID of the LSA.
- Router ID
  Shows the ID of the router that sent this LSA.
- Sequence Number
  Shows the sequence number of the LSA. Each time an LSA is renewed, this sequence number is incremented by one.

RIPng Statistics

Overview

This page shows the statistics of the RIPng interfaces.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- Interface
  Shows the RIPng interface.
- Status
  Shows the status of the RIPng interface.
  - up
  - down
- Messages Received
  Shows how often the router has received messages.
- Requests Received
  Shows how often the router has received requests.
- Responses Received
  Shows how often the router has received responses.
- Unknown Commands
  Shows how many RIPng packets were received whose value in the "Commands" field of the RIPng header is unknown. Known values are 1 for requests and 2 for responses.
- Other version
  Shows how many RIPng packets were received whose value in the "Version" field is not 1.
- Discards
  Shows how often the router has discarded RIPng packets.
- Messages Sent
  Shows how often the router has sent messages.
- Requests Sent
  Shows how often the router has sent requests.
- Responses Sent
  Shows how often the router has sent responses.
- Updates Sent
  Shows how often the router has sent its routing table to its neighbor routers.

DHCP Server

This page shows whether IPv4 addresses were assigned to the devices by the DHCP server.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

- **IP Address**
  Shows the IPv4 address assigned to the DHCP client.
- **Pool ID**
  Shows the number of the IPv4 address band.
- **Identification method**
  Shows the method according to which the DHCP client is identified.
- **Identification value**
  Shows the MAC address or he client ID of the DHCP client.
- **Assignment method**
  Shows whether the IPv4 address was assigned statically or dynamically. You configure the static entries in "System > DHCP > Static Leases".
- **Assignment status**
  Shows the status of the assignment.
  - Assigned
    The assignment is used.
  - Not used
    The assignment is not used.
  - Probing
    The assignment is being checked.
  - Unknown
    The status of the assignment is unknown.
- **Expire Time**
  Shows until when the assigned IPv4 address is still valid. Up to this time, the DHCP client must either request a new IPv4 address or extend the lease time of the assigned IPv4 address.

SNMP

Note

The page is only available if there is an online connection to the device.

This page displays the created SNMPv3 groups. You configure the SNMPv3 groups in "System" > SNMP"..

Description of the displayed values

The table has the following columns:

- **Group Name**
  Shows the group name.
- **User Name**
  Shows the user that is assigned to the group.

Security

Overview

This page shows the security settings and the local and external user accounts.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the rights of the logged-on user.

Description of the displayed values

Services

The "Services" list shows the security settings.

- **SSH Server**
  You configure the setting in "System > Configuration".
  - Enabled: Encrypted access to the CLI.
  - Disabled: No encrypted access to the CLI.
- **Web Server**
  You configure the setting in "System > Configuration".
  - HTTP/HTTPS: Access to the WBM is possible with HTTP and HTTPS.
  - HTTPS: Access to the WBM is now only possible with HTTPS.
- **SNMP**
  You can configure setting in "System > SNMP > General".
  - "-" (SNMP disabled)
    Access to device parameters via SNMP is not possible.
  - SNMPv1/v2c/v3
    Access to device parameters is possible with SNMP versions 1, 2c or 3.
  - SNMPv3
    Access to device parameters is possible only with SNMP version 3.
- **Management ACL**
  You configure the setting in "Security > Management ACL".
  - Disabled: No access restriction
    The access control is disabled.
  - Enabled: No access restriction
    The access control is enabled but no access rules have been defined.
  - Enabled: Restricted access only
    The access control is enabled and access rules have been defined.

- **Login Authentication**
  You configure the setting in "Security > AAA > General".
  - Local
    The authentication must be made locally on the device.
  - RADIUS
    The authentication must be handled via a RADIUS server.
  - Local and RADIUS
    The authentication is possible both with the users that exist on the device (user name and password) and via a RADIUS server.
    The user is first searched for in the local database. If the user does not exist there, a RADIUS query is sent.
  - RADIUS and fallback local
    The authentication must be handled via a RADIUS server.
    A local authentication is performed only when the RADIUS server cannot be reached in the network.
- **Password Policy**
  Shows which password policy is currently being used.

Local and external user accounts

You configure local user accounts and roles in "Security > User Accounts"

When you create a local user account an external user account is generated automatically. Local user accounts involve users each with a password for logging in on the device.

In the table "External User Accounts" a user is linked to a role, for example the user "Observer" with the role "user" In this case, the user is defined on a RADIUS server. The roll is defined locally on the device. When a RADIUS server authenticates a user, the corresponding group however is unknown or does not exist, the device checks whether or not there is an entry for the user in the table "External User Accounts". If an entry exists, the user is logged in with the rights of the associated role. If the corresponding group is known on the device, both tables are evaluated. The user is assigned the role with the higher rights.

Note

The table "External User Accounts" is only evaluated if you have set "Vendor Specific" in the RADIUS Authorization Mode".

With CLI you can access external user accounts.

The table "Local User Accounts" has the following columns:

- **User Account**
  Shows the name of the local user.
- **Role**
  Shows the role of the user. You can obtain more information on the function rights of the role in "Information > Security > Roles".

Supported Function Rights

The page shows the function rights available locally on the device.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the role of the logged-on user.

Description of the displayed values

- **Function Right**
  Shows the number of the function right. Different rights relating to the device parameters are assigned to the numbers.
- **Description**
  Shows the description of the function right.

Roles

The page shows the roles valid locally on the device.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the role of the logged-on user.

Description of the displayed values

This table contains the following columns:

- **Role**
  Shows the role.
- **Function Right**
  Shows the function rights of the role:
  - 1
    Users with this role can read device parameters but cannot change them.
  - 15
    Users with this role can both read and change device parameters.
  - 0
    This is a role that the device assigns internally when the user could not be authenticated. The user is denied access to the device.
- **Description**
  Shows a description of the role.

Groups

This page shows which group is linked to which role. The group is defined on a RADIUS server. The roll is defined locally on the device.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the role of the logged-on user.

Description of the displayed values

The table has the following columns:

- **Group**
  Shows the name of the group. The name matches the group on the RADIUS server.
- **Role**
  Shows the role. Users who are authenticated with the linked group on the RADIUS server receive the rights of this role locally on the device.
- **Description**
  Shows a a description for the link.

Configuring system functions

Configuration

The page contains the configuration overview of the access options of the device. Specify the services that access the device. With some services, there are further configuration pages on which more detailed settings can be made.

Description

The page contains the following boxes:

- **Telnet Server**
  Enable or disable the "Telnet Server" service for unencrypted access to the CLI.
- **SSH server**
  Enable or disable the "SSH Server" service for encrypted access to the CLI.
- **HTTPS server only**
  If this function is enabled, you can only access the device via HTTPS.
- **DNS client**
  Enable or disable depending on whether the IE switch should operate as a DNS client. You can configure other settings in "System > DNS Client".
- **SMTP Client**
  Enable or disable the SMTP client. You can configure other settings in "System > SMTP Client".

- Syslog Client
  Enable or disable the Syslog client. You can configure other settings in "System > Syslog client".
- DCP Server
  Specify whether or not the device can be accessed with DCP (Discovery and Configuration Protocol):
  - "-" (disabled)
    DCP is disabled. Device parameters can neither be read nor modified.
  - Read/write access
    With DCP, device parameters can be both read and modified.
  - Read-Only
    With DCP, device parameters can be read but cannot be modified.
- RCDP Server
  Specify whether or not the device can be accessed with RCDP (RUGGEDCOM Discovery Protocol):
  - "-" (deactivated)
    RCDP is deactivated. Device parameters can neither be read nor modified.
  - Read/Write
    With RCDP, device parameters can be both read and modified.
  - Read Only
    With RCDP, device parameters can be read but cannot be modified.
- Time
  Select the required setting. The following settings are possible:
  - Manual
    The system time is set manually. You can configure other settings in "System > System Time > Manual Setting".
  - SIMATIC Time
    The system time is set using a SIMATIC time transmitter. You can configure other settings in "System > System Time > SIMATIC Time Client".
  - SNTP Client
    The system time is set via an SNTP server. You can configure other settings in "System > System time > SNTP client".
  - NTP Client
    The system time is set via an NTP server. You can configure other settings in "System > System time > NTP client".
  - PTP Client(SCALANCE XR528-6M and XR552-12M only)
    The system time is set via a PTP server. You can configure other settings in "System > System Time > PTP Client".

- **SNMP**
  Select the required protocol. The following settings are possible:
  - "-" (SNMP disabled)
    Access to device parameters using SNMP is not possible.
  - SNMPv1/v2c/v3
    Access to device parameters is possible with SNMP versions 1, 2c or 3. You can configure other settings in "System > SNMP > General".
  - SNMPv3
    Access to device parameters is possible with SNMP version 3. You can configure other settings in "System > SNMP > General".
- **SNMPv1/v2 Read-Only**
  Enable or disable write access to SNMP variables with SNMPv1/v2c.
- **SNMPv1 traps**
  Enable or disable the sending of SNMP traps (alarm frames). You can configure other settings in "System > SNMP > Traps".
- **SINEMA configuration interface**
  If the SINEMA configuration interface is enabled, you can download configurations to the IE switch via the TIA Portal.
- **DUID Type**
  Specify which DUID type is used. The DUID types are defined in RFC 3315.
  - DUID-LLT
    DUID is based on the link layer address of the interface and a time stamp
  - DUID-EN
    DUID is assigned by the vendor (EN = enterprise number)
  - DUID-LL
    DUID is based on the link layer address of the interface
- **Link-layer Address Plus Time (LLT)**
  The value is based on the link layer address of the interface and a time stamp. The value is regenerated each time the factory settings are restored. The value can be changed if necessary.
- **Vendor Enterprise Number (EN)**
  The value is based on the enterprise number specific to the vendor. The value is regenerated each time the factory settings are restored. The value can be changed if necessary.
- **Link-layer address (LL)**
  The link-layer address is based on the MAC address. The value is regenerated each time the factory settings are restored. The value can be changed if necessary.

- **NFC** (only with SCALANCE XM-400)
  Enable or disable the "NFC" function.
  You will find further information on NFC in the "SCALANCE XM400" Operating Instructions.
- **Configuration mode**
  Select the required operating mode. The following modes are possible:
  - Automatic save
    Automatic backup mode. Approximately 1 minute after the last parameter change or when you restart the device, the configuration is automatically saved.
  - Trial
    In the "Trial" configuration mode, although changes are adopted, they are not saved in the configuration file (startup configuration).
    To save changes in the configuration file, use the "Write startup config" button. The "Write startup config" button is displayed when you set trial mode. The display area also shows the message "Trial Mode Active – Click the "Write Startup Config" button to make your settings persistent" as soon as there are unsaved modifications. This message can be seen on every page until the changes made have either been saved or the device has been restarted.

General

Device

This page contains the general device information.

Description

The page contains the following boxes:

- **Current System Time**(Only available online)
  Shows the current system time. The system time is either set by the user or by a time-of-day frame: either SINEC H1 time-of-day frame, NTP or SNTP. (readonly)
- **System Up Time**(only available online)
  Shows the operating time of the device since the last restart. (readonly)
- **Device Type**(only available online)
  Shows the type designation of the device. (readonly)
- **System Name**
  You can enter the name of the device. The entered name is displayed in the selection area. A maximum of 255 characters are possible. The system name is also displayed in the CLI input prompt. The number of characters in the CLI input prompt is limited. The system name is truncated after 16 characters.

- System Contact
  You can enter the name of a contact person responsible for managing the device. A maximum of 255 characters are possible.
- Location
  You can enter the location of the device. The entered installation location is displayed in the selection area. A maximum of 255 characters are possible.

Note

The ASCII characters 0x20 to 0x7e are used in the input boxes.

Coordinates

On this page, you can enter information on the geographic coordinates (latitude, longitude and the height above the ellipsoid according to WGS84).

Getting the coordinates

Use suitable maps for obtaining the geographic coordinates of the device.

The geographic coordinates can also be obtained using a GPS receiver. The geographic coordinates of these devices are normally displayed directly and only need to be entered in the input boxes of this page.

Description

The page contains the following input boxes with a maximum length of 32 characters.

- "Latitude" input box
  Enter the north or south latitude for the location of the device.
  E. G. +49° 1´ 31.67" means that the device is located at 49 degrees, 1 arc minute and 31.67 arc seconds north latitude.
  A south latitude is shown by a preceding minus character.
  You can also append the letters N (north latitude) or S (south latitude) to the numeric information (49° 1´ 31.67" N).
- "Longitude" input box
  Enter the eastern or western longitude for the location of the device.
  E. G. +8° 20´ 58.73" means that the device is located at 8 degrees, 20 arc minutes and 58.73 arc seconds eastern longitude.
  A western longitude is indicated by a preceding minus sign.
  You can also add the letter E (eastern longitude) or W (western longitude) to the numeric information (8° 20´ 58.73" E).
- "Height" input box
  Enter the value of the height above sea level in meters.
  For example, 158 m means that the device is located at a height of 158 m above sea level.
  Heights below sea level (for example the Dead Sea) are indicated by a preceding minus sign.

Agent IP

Configuration of the IP addresses

Note

The page is only available if there is an online connection to the device.

On this page, you specify the IP configuration for the device.

With devices with more than one IP interface, this call references the "Subnets > Configuration" menu item in the "Layer 3 (IPv4)" menu and the configuration of the TIA interface there.

Description

The page contains the following boxes:

- **IP Assignment Method**
  Shows how the IP address is assigned.
  - Static
    The IP address is static. You enter the IP settings in the input boxes "IP Address" and "Subnet Mask".
  - Dynamic (DHCP)
    The device obtains a dynamic IP address from a DHCP server.
- **IP Address**
  Enter the IP address of the device.
  After clicking the "Set Values" button, this IP address is also displayed in the address bar of the Internet browser. If this does not take place automatically, you will need to enter the IP address in the address bar of the Internet browser manually.
- **Subnet Mask**
  Enter the subnet mask of the device.
- **Default gateway**
  Enter the IP address of the default gateway to be able to communicate with devices in another subnet, for example diagnostics stations, e-mail server.
- **Agent VLAN ID**
  Select the VLAN ID from the drop-down list. You can only select VLANs that have already been configured.
  In the mode "802.1D Transparent Bridge", this drop-down list is grayed out, see also "Layer 2 > VLAN > General".

  Note

  Changing the Agent VLAN ID

  If the configuration PC is connected directly to the device via Ethernet and you change the Agent VLAN ID , the device is no longer reachable via Ethernet following the change.

- **MAC Address**
  Shows the MAC address of the device. The MAC address is linked to the hardware and cannot be modified.

DNS

DNS client

On this page, you can manually configure up to 3 DNS servers with IPv4 or IPv6 addresses. Manually configured DNS servers are each assigned an index from 1 to 3. Using DHCP the IE switch can learn 2 DNS servers with IPv4 addresses. Learned DNS servers are automatically assigned an index from 6 to 7.

If there is more than one DNS server, the order in the table specifies the order in which the servers are queried. The top server is queried first. In total 5 DNS servers can be configured on the IE switch. Manually configured DNS servers are given preference.

The DNS () server (Domain Name System) assigns a domain name to an IP address so that a device can be uniquely identified.

If this function is enabled, the IE switch can communicate with a DNS server as a DNS client. You have the option of entering names in IP address boxes.

Note

The DNS client function can only be used if there is a DNS server in the network.

Description

The page contains the following boxes:

- **DNS Client**
  Enable or disable depending on whether the IE switch should operate as a DNS client.
- **Used DNS Servers**
  Specify which DNS server the device uses:
  - learned only
    The device uses only the DNS servers assigned by DHCP.
  - manual only
    The device uses only the manually configured DNS servers. The DNS servers must be connected to the Internet. A maximum of three DNS servers can be configured.
  - all
    The device uses all available DNS servers.
- **DNS Server IP Address**
  Enter the IP address of the DNS server.

The table contains the following columns:

- **DNS Server Address**
  Shows the IP address of the DNS server.
- **Origin**
  Shows whether the DNS server was configured manually or was assigned by DHCP.

DNS domain

On this page, you can define up to 4 domain names. The primary domain name is used first to resolve a host name.

Domain names 2 to 4 can be learned or configured manually on this page. If there is more than one DNS server, the order in the table specifies the order in which the domain names are used.

If domain names are stored, you have the option of entering the host name for some of the IP address fields.

Description

The page contains the following boxes:

- Primary Domain
  Enter the name of the primary domain. This entry is used first to resolve a host name.
- Domain Name
  Enter the name of the other domain.

The table contains the following columns:

- Domain Name
  Shows the name of the other domain.
- Origin
  Shows whether the domain name was configured manually or was assigned by DHCP.

Restart

In this menu, there is a button with which you can restart the device and the option of resetting to the factory setting or reset the default settings of various profiles.

Note

The page is only available if there is an online connection to the device.

Restart

Note the following points about restarting a device:

- You can only restart the device with administrator privileges.
- A device should only be restarted with the buttons of this menu or with the appropriate CLI commands and not by a power cycle on the device.
- If the device is in "Trial" mode, configuration modifications must be saved manually before a restart. Any modifications you have made only become active on the device after clicking the "Set Values" button on the relevant WBM page.
- If the device is in "Automatic Save" mode, the last changes are saved automatically before a restart.

Reset to Factory Defaults

By resetting all the settings to the factory settings, the IP address and the passwords are also lost. Following this, the device can only be accessed via the serial interface, using the Primary Setup Tool or using DHCP.

NOTICE
Failure of the data traffic With the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

Resetting to defaults (profiles)

The profiles provide a preconfiguration for various use cases of the devices.

When you start a device with the default settings of a profile, the settings are reset to the factory settings and some parameters are set so that they are designed for a use case. In contrast to resetting to the factory settings. the users and passwords are retained after the restart. The configured IP address is lost so that device can then only be accessed via the serial interface, using the Primary Setup Tool or using DHCP.

NOTICE
Failure of the data traffic With the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

Which settings were set specially for the profile are displayed before the restart. The profiles can be used independently of the factory setting of the device.

Description

Note

Note the effects of the individual functions described in the sections above.

To restart the device, the buttons on this page provide you with the following options:

- **Restart**
  Click this button to restart the system. You must confirm the restart in a dialog box. During a restart, the device is reinitialized, the internal firmware is reloaded, and the device runs a self-test. The settings of the start configuration are retained, e.g. the IP address of the device. The learned entries in the address table are deleted. You can leave the browser window open while the device restarts. After the restart you will need to log in again.
- **Restore Factory Defaults and Restart**
  Click this button to restore the factory defaults of the device and to restart the device. You must confirm the restart in a dialog box. The factory settings depend on the device.

To restart the device with a predefined profile, the buttons on this page provide you with the following options:

- PROFINET Defaults
  Click this button to restore the default settings of the PROFINET profile and to restart the device. You must confirm the restart in a dialog box. The dialog box displays the settings specially made for operation with the PROFINET protocol.
- EtherNet/IP Defaults
  Click this button to restore the default settings of the EtherNet/IP profile and to restart the device. You must confirm the restart in a dialog box. The dialog box displays the settings specially made for operation with the EtherNet/IP protocol.
- Industrial Ethernet Defaults
  Click this button to restore the default settings of the Industrial Ethernet profile and to restart the device. You must confirm the restart in a dialog box. The dialog box displays the settings specially made for operation in the Industrial Ethernet environment.

Load & save

File types

Overview of the file types

File type	Description
Config	This file contains the start configuration. Among other things, this device contains the definitions of the users, roles, groups and function rights. The passwords are stored the file "Users".
ConfigPack	Detailed configuration information. for example, start configuration, users, certificates and WBM favorites. ZIP file consisting of the Config, Users and LSYS fle.
ConfigPackBackup	This ZIP file stores all the configuration backups you have created.
Debug	This file contains information for Siemens Support. It is encrypted and can be sent by e-mail to Siemens Support without any security risk.
DebugExt	This file contains detailed information for Siemens Support. It is encrypted and can be sent by e-mail to Siemens Support without any security risk. Saving the file may take some time.
EDS	Electronic Data Sheet (EDS) Electronic data sheets for describing devices in the EtherNet/IP mode
Firmware	The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.
GSDML	PROFINET information on the device properties

File type	Description
HTTPSCert	Default HTTPS certificates including key The preset and automatically created HTTPS certificates are self-signed. We strongly recommend that you create your own HTTPS certificates and make them available. We recommend that you use HTTPS certificates signed either by a reliable external or by an internal certification authority. The HTTPS certificate checks the identity of the device and controls the encrypted data exchange. Certificates with a different format cannot be copied in.
LogFile	File with entries from the event log table
MIB	Private MSPS MIB file
RunningCLI	Text file with CLI commands This file contains an overview of the current configuration in the form of CLI commands. Passwords are masked in this file as follows: [PASSWORD] You can download the text file. The file is not intended to be uploaded again unchanged.
RunningSINEMA-Config	You save the current device configuration in this file type for transfer to STEP 7 Basic/Professional. The file can be imported in STEP 7 Basic/Professional and installed on a device with the same article number and firmware version. Before you can save a file, you must assign a password for the "RunningSINEMAConfig" in the WBM under "System > Load&Save > Passwords". You also need this password to import the file into STEP 7 Basic/Professional. See also "SINEMAConfig"
Script	Text file with CLI commands You can upload a script file in a device. The CLI commands it contains are executed appropriately.
SINEMAConfig	You load configuration data that was exported via STEP 7 Basic/Professional for transfer to the WBM with this file type. To load a file, you must assign a password for the "SINEMAConfig" under "System > Load&Save > Passwords". You also need this password to export the file from STEP 7 Basic/Professional. See also "RunningSINEMAConfig"
StartupInfo	Startup log file This file contains the messages that were entered in the log during the last start-up.
Users	This file contains the assignment of the user names to the corresponding passwords.
WBMFav	WBM favorite pages This file contains the favorites that you created in the WBM.. You can download this file and upload it in other devices.

HTTP

Loading and saving data via HTTP

This page allows you to store device data in an external file on your client PC or to load such data from an external file from the client PC to the devices.
This means, for example, that you can also load new firmware from a file located on your client PC.

Note

The page is only available if there is an online connection to the device.
This page is available both for connections using HTTP and for connections using HTTPS.

Firmware

The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.

Note

Incompatibility with previous firmware versions with/without PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed.

In this situation, if a PLUG is inserted in the device, following the restart, this has the status "Not Accepted" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data. If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using the WBM page "System > PLUG".

Configuration files

Note

Configuration files and trial mode/automatic save mode

In Automatic Save mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In Trial mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write Startup Config" button on the "System > Configuration" WBM page to save changes in the configuration file.

CLI script file

You can download existing CLI configurations (RunningCLI) and upload your own CLI scripts (Script).

Note

The downloadable CLI script is not intended to be uploaded again unchanged.

Exchange of configuration data with STEP 7 Basic/Professional using a file

You use the two file types "RunningSINEMAConfig" and "SINEMAConfig" to exchange configuration data between a device (WBM) and STEP 7 Basic/Professional via a file.

Requirements:

- Same article number
- Same firmware version
- Password
  You assign the password in the WBM under "System > Load&Save > Passwords".

You can use the file types as follows:

- For offline diagnostics
  You can save the faulty configuration of a device as "RunningSINEMAConfig" via the WBM and import it in STEP 7 Basic/Professional. No connection to a real device is required for the diagnostics in STEP 7 Basic/Professional. You can export a corrected configuration and load it as "SINEMAConfig" again using the WBM.
- For configuration
  No connection to a real device is required to configure a device in STEP 7 Basic/Professional. You can export the configuration and load it as "SINEMAConfig" to the real device using the WBM.

Description

The table has the following columns:

- **Type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.
- **Load**
  With this button, you can upload files to the device. The button can be enabled, if this function is supported by the file type.
- **Save**
  With this button, you can save files from the device. The button can only be enabled if this function is supported by the file type and the file exists on the device.
- **Delete**
  With this button, you can delete files from the device. The button can only be enabled if this function is supported by the file type and the file exists on the device.

Note

Following a firmware update, delete the cache of your Internet browser.

TFTP

Loading and saving data via a TFTP server

On this page, you can configure the TFTP server and the file names. You can also store device data in an external file on a TFTP server or load such data from an external file from the TFTP server to the devices. This means, for example, that you can also load new firmware from a file located on a TFTP server.

Firmware

The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.

Note

Incompatibility with previous firmware versions with/without PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed.

In this situation, if a PLUG is inserted in the device, following the restart, this has the status "Not Accepted" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data. If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using the WBM page "System > PLUG".

Configuration files

Note

Configuration files and Trial mode /Automatic save

In "Automatic Save" mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In "Trial" mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write startup config" button on the "System > Configuration" WBM page to save changes in the configuration files.

CLI script file

You can download existing CLI configurations (RunningCLI) and upload your own CLI scripts (Script).

Note

The downloadable CLI script is not intended to be uploaded again unchanged.

CLI commands for saving and loading files cannot be executed with the CLI script file (Script).

Exchange of configuration data with STEP 7 Basic/Professional using a file

You use the two file types "RunningSINEMAConfig" and "SINEMAConfig" to exchange configuration data between a device (WBM) and STEP 7 Basic/Professional via a file.

Requirements:

- Same article number
- Same firmware version
- Password
  You assign the password in the WBM under "System > Load&Save > Passwords".

You can use the file types as follows:

- For offline diagnostics
  You can save the faulty configuration of a device as "RunningSINEMAConfig" via the WBM and import it in STEP 7 Basic/Professional. No connection to a real device is required for the diagnostics in STEP 7 Basic/Professional. You can export a corrected configuration and load it as "SINEMAConfig" again using the WBM.
- For configuration
  No connection to a real device is required to configure a device in STEP 7 Basic/Professional. You can export the configuration and load it as "SINEMAConfig" to the real device using the WBM.

Description

- **TFTP Server Address**
  Enter the IP address or the FQDN of the TFTP server with which you exchange data.
- **TFTP Server Port**
  Here, enter the port of the TFTP server via which data exchange will be handled. If necessary, you can change the default value 69 to your own requirements.

The table has the following columns:

- **File type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.
- **File name**
  A file name is preset here for every file type.

  Note

  Changing the file name

  You can change the file name preset in this column. After loading on the device, the changed file name can also be used with the Command Line Interface.

- **Action** (only available online)
  Select the required action. The selection depends on the selected file type, for example you can only save the log file.
  The following actions are possible:
  - Save file
    With this selection, you save a file on the TFTP server.
  - Load file
    With this selection, you load a file from the TFTP server, e.g. certificates required to establish a secure connection.

SFTP

Loading and saving data via an SFTP server

SFTP (SSH File Transfer Protocol) transfers the files encrypted. On this page, you configure the access data for the SFTP server.

You can also store device data in an external file on your client PC or load such data from an external file from the PC to the devices. This means, for example, that you can also load new firmware from a file located on your Admin PC.

On this page, the certificates required to establish a secure VPN connection can also be loaded.

Firmware

The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.

CLI script file

You can download existing CLI configurations (RunningCLI) and upload your own CLI scripts (Script).

Note

The downloadable CLI script is not intended to be uploaded again unchanged.

CLI commands for saving and loading files cannot be executed with the CLI script file (Script).

X509 certificates

The following file types can be loaded into the device:

- .crt, pem, zip: Maximum file name length 255 characters
- .p12: Maximum file name length 248 characters

Exchange of configuration data with STEP 7 Basic/Professional using a file

You use the two file types "RunningSINEMAConfig" and "SINEMAConfig" to exchange configuration data between a device (WBM) and STEP 7 Basic/Professional via a file.

Requirements:

- Same article number
- Same firmware version
- Password
  You assign the password in the WBM under "System > Load&Save > Passwords".

You can use the file types as follows:

- For offline diagnostics
  You can save the faulty configuration of a device as "RunningSINEMAConfig" via the WBM and import it in STEP 7 Basic/Professional. No connection to a real device is required for the diagnostics in STEP 7 Basic/Professional. You can export a corrected configuration and load it as "SINEMAConfig" again using the WBM.
- For configuration
  No connection to a real device is required to configure a device in STEP 7 Basic/Professional. You can export the configuration and load it as "SINEMAConfig" to the real device using the WBM.

Description of the displayed boxes

The page contains the following boxes:

- **SFTP Server Address**
  Enter the IP address or the FQDN of the SFTP server with which you exchange data.
- **SFTP Server Port**
  Enter the port of the SFTP server via which data exchange will be handled. If necessary, you can change the default value 22 to your own requirements.
- **SFTP User**
  Enter the user for access to the SFTP server. This assumes that a user with the corresponding rights has been created on the SFTP server.
- **SFTP Password**
  Enter the password for the user
- **SFTP Password Confirmation**
  Confirm the password.

The table has the following columns:

- **Type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.

- **Filename**
  A file name is preset here for every file type.

  Note

  Changing the file name

  You can change the file name preset in this column. After loading on the device, the changed file name can also be used with the Command Line Interface.

- **Actions**
  Select the action from the drop-down list. The selection depends on the selected file type, for example you can only save the log file.
  The following actions are possible:
  - **Save file**
    With this selection, you save a file on the SFTP server.
  - **Load file**
    With this selection, you load a file from the SFTP server.

Passwords

There are files to which access is password protected. To load the file on the device, enter the password specified for the file on the WBM page.

Description

The table has the following columns:

- **Type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.
- **Enabled**
  When selected, the password is used. Can only be enabled if the password is configured.
- **Password**
  Enter the password for the file.
- **Password Confirmation**
  Confirm the new password.
- **Status**
  Shows whether the current settings for the file match the device.
  - Valid
    the "Enabled" setting is selected and the password matches the certificate.
  - Invalid
    the "Enabled" setting is selected but the password does not match the certificate or no certificate has been loaded yet.
  - '-'
    The password cannot be evaluated or is not yet being used. The "Enabled" check box is not selected.

Events

Configuration

On this page, you specify how the device reacts to system events. By enabling the appropriate options, you specify how the device reacts to events. To enable or disable the options, click the relevant settings of the columns.

Description

- Signaling Contact Response
  Select the behavior of the signaling contact. The available options are as follows:
  - conventional
    Default setting for the signaling contact. An error/fault is displayed by the fault LED and the signaling contact is opened. When the error/fault state no longer exists, the fault LED goes off and the signaling contact is closed.
  - User-defined
    The way the signaling contact works does not depend on the error/fault that has occurred. The signaling contact can be opened or closed as required by user actions.
- Signaling Contact Status
  Select the status of the signaling contact from the drop-down list. The following statuses are possible:
  - close
    Signaling contact is closed.
  - open
    Signaling contact is opened.

Table 1 has the following columns:

- 1st column
  Shows that the settings are valid for all ports of table 2.
- E-mail/Trap/Log table/Syslog/Errors
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Transfer to table
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Event**
  The "Event" column contains the following values:
  - Cold/Warm Start
    The device was turned on or restarted by the user.
  - Link Change
    This event occurs only when the port status is monitored and has changed, see "System > Fault Monitoring > Link Change".
  - Authentication Failure
    This event occurs when attempting access with a bad password.
  - RMON Alarm
    An alarm or event has occurred relating to the remote monitoring of the system.
  - Power Supply Switchover
    This event occurs only when power supply lines 1 and 2 are monitored. It indicates that there was a change to line 1 or line 2, see System > Fault Monitoring > Power Supply".
  - RM status change
    The redundancy manager has recognized an interruption or restoration of the ring and has switched the line over or back.
  - Spanning Tree Change
    The STP or RSTP or MSTP topology has changed.
  - Fault State Change
    The fault state has changed. The fault state can relate to the activated port monitoring, the response of the signaling contact or the power supply monitoring.
  - Standby State Change
    A device with an established standby connection (master or slave) has activated or deactivated the link to the other ring (standby port). The data traffic was redirected from one Ethernet connection (standby port of the master) to the other Ethernet connection (standby port of the slave).
  - VRRP State Change (only with routing using VRRP)
    The status of the virtual router has changed
  - Loop Detection
    A loop was detected in the network segment.
  - Diagnostics Alarms
    The diagnostics value has fallen below or exceeded a certain limit.
  - OSPF State Change
    The status of OSPF has changed.
  - 802.1X Port Authentication State Change
    This event occurs with 802.1X authentications.
  - PoE State Change
    The status of PoE has changed.
  - FMP Status Change
    The value of the received power or the power loss has fallen below a certain limit.

- Link Check
  A disruption was detected on an optical transmission link.
- CLI Script File
  An error was detected in the CLI script file.
- NTP (secured)
  An error occurred when using Secure NTP, e.g. a key with the wrong length was specified.
- Configuration Change
  The configuration was saved retentively.
- MRP Interconnection State Change
  A switch from the primary to the secondary connection or vice versa took place in the MRP Interconnection domain.
- Service Information
  For certain events, entries are made in the log table even without configuration. For these events, you can configure additional subsequent actions here (e-mail, trap, syslog).

- **E-mail**
  The device sends an e-mail. This is only possible if the SMTP server is set up and the "SMTP client" function is enabled.
- **Trap**
  The device sends an SNMP trap. This is only possible if "SNMPv1 traps" is enabled in "System > Configuration".
- **Log table**
  The device writes an entry in the event log table, see "Information > Log Table"
- **Syslog**
  The device writes an entry to the system log server. This is only possible if the system log server is set up and the "Syslog client" function is enabled.
- **Fault**
  The device triggers a fault. The error LED lights up

Severity filter

On this page, set the threshold levels for sending system event notifications.

Settings

The table has the following columns:

- **Client Type**
  Select the client type for which you want to make settings:
  - E-mail
    Sending system event messages by e-mail
  - Log table
    Entry of system events in the log table.
  - Syslog
    Entering system events in the Syslog file
  - WLAN Authentication Log
    Forwarding of the entries from the WLAN Authentication Log to the system protocol server.
    Can only be configured with SCALANCE W770 and SCALANCE W780
- **Severity**
  Select the required level. The following settings are possible:
  - Info
    System events are processed as of the severity level "Info".
  - Warning
    System events are processed as of the severity level "Warning".
  - Critical
    System events are processed as of the severity level "Critical".

SMTP client

Network monitoring with e-mails

The device provides the option of automatically sending an e-mail if an alarm event occurs (for example to the network administrator). The e-mail contains the identification of the sending device, a description of the cause of the alarm in plain language, and a time stamp. This allows centralized network monitoring to be set up for networks with few nodes based on an E-mail system. When an e-mail error message is received, the WBM can be started by the Internet browser using the identification of the sender to read out further diagnostics information.

On this page, you can configure up to three SMTP servers and the corresponding e-mail addresses.

Description

The page contains the following boxes:

- **SMTP Client**
  Enable or disable the SMTP client.
- **Sender Email Address**
  Enter the name of the sender to be included in the e-mail, for example the device name. This setting applies to all configured SMTP servers.

- **Send Test Mail**
  Send a test e-mail to check your configuration.
- **SMTP Port**
  Enter the port via which your SMTP server can be reached.
  Factory settings: 25
  This setting applies to all configured SMTP servers.
- **SMTP Server Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the SMTP server.

This table contains the following columns:

- **SMTP Server Address**
  Shows the IP address or the FQDN (Fully Qualified Domain Name) of the SMTP server.
- **Email address of the recipient**
  Enter the e-mail address to which the device sends an e-mail if a fault occurs.

Syntax of email addresses

The following conditions apply to e-mail addresses:

- Alphanumeric characters are permitted.
- The following special characters are permitted:
  - @
  - _ (underscore)
  - - (hyphen)
  - . (period)
- A @ character must be included.
- Only one @ may be used.
- The characters "@" and the "." must not be the first or last character.

Note

Depending on the properties and configuration of the SMTP server, it may be necessary to adapt the "Sender E-Mail Address” input box. Check with the administrator of the SMTP server.

DHCPv4

DHCP client

Setting of the DHCP mode

If the device is configured as a DHCP client, it starts a DHCP query. As the reply to the query the device receives an IPv4 address from the DHCP server. The server manages an address range from which it assigns IPv4 addresses. It is also possible to configure the server so that the client always receives the same IPv4 address in response to its request.

Descriptions

- **DHCP client configuration file request (opt 66, 67)**
  Select this option if you want the DHCP client to use options 66 and 67 to download and then enable a configuration file.
- **DHCP Mode**
  Select the DHCP mode. The following modes are possible:
  - via MAC Address
    Identification is based on the MAC address.
  - via DHCP Client ID
    Identification is based on a freely defined DHCP client ID.
  - via System Name
    Identification is based on the system name. If the device name is 255 characters long, the last character is not used for identification.
  - via PROFINET Name of Station
    Identification is based on the PROFINET name of the station.
  - via IAID and DUID
    With this, the DHCP client can log on to DHCP servers that support parallel operation of IPv4 and IPv6.
    The identification is via the IAID and the DUID and identifies precisely one IP interface of the device.
    IAID (Interface Association Identifier): At least one IAID is generated for each IP interface. The IAID remains unchanged when the DHCP client restarts.
    DUID (DHCP Unique Identifier): Uniquely identifies servers and clients and applies to all IP interfaces of the device. The DUID remains unchanged when there is a restart, unless the user changes it.
- **DUID-Type**
  Specify which DUID type will be used. The DUID types are defined in RFC 3315.
  - DUID-LLT
    DUID is based on the link layer address of the interface and a time stamp
  - DUID-EN
    DUID is assigned by the vendor (EN = enterprise number)
  - DUID-LL
    DUID is based on the link layer address of the interface

- Link-layer Address Plus Time (LLT)
  The value is based on the link layer address of the interface and a time stamp. The value is regenerated each time the factory settings are restored. The value can be changed if necessary.
- Vendor Enterprise Number (EN)
  The value is based on the enterprise number specific to the vendor. The value is regenerated each time the factory settings are restored. The value can be changed if necessary.
- Link-layer address (LL)
  The link layer address is based on the MAC address. The value is regenerated each time the factory settings are restored. The value can be changed if necessary.

This table contains the following columns:

- Interface
  Interface to which the setting relates.
- DHCP
  Enable or disable the DHCP client for the relevant interface.
- IAID Value
  Value with which the interface (DHCP client) identifies itself on the DHCP server.

DHCP Server

You can operate the device as a DHCP server. This allows IP addresses to be assigned automatically to the connected devices. The IP addresses are either distributed dynamically from an address band (pool) you have specified or a specific IP address is assigned to a particular device.

Note

Deleting DHCP server bindings

If you deactivate or delete an IPv4 address band or turn the DHCP server off and on again, the DHCP server bindings are deleted see "Information > DHCP Server".

The structure of this page depends on how many VLAN IP interfaces the device has.

Requirement

The connected devices are configured so that they obtain the IP address from a DHCP server.

Devices with one VLAN IP interface

On this page, you specify tIPv4 addresses that are assigned via certain ports.

Description

The page contains the following boxes:

- **DHCP Server**
  Enable or disable the DHCP server on the device.

  Note

  To avoid conflicts with IPv4 addresses, only one device may be configured as a DHCP server in the network.

- **Probe address with ICMP echo before offer**
  When selected, the DHCP server checks whether or not the IP address has already been assigned. To do this the DHCP server sends ICMP echo messages (ping) to the IPv4 address. If no reply is received, the IPv4 address is assigned.

  Note

  This check is not made with static assignments.

  Note

  If there are devices in your network on which the echo service is disabled as default, there may be conflicts with the IPv4 addresses. To avoid this, assign these devices an IPv4 address outside the IPv4 address band.

The table has the following columns:

- **Pool ID**
  Shows the number of the IPv4 address band. If you click the "Create" button, a new row with a unique number is created (pool ID).
- **Port**
  Specify the port via which IPv4 address of this DHCP pool will be assigned.
  The requirement for the assignment is that the IPv4 address of the interface is located in the subnet of the IPv4 address band. If this is not the case, the interface does not assign any IPv4 addresses.
- **Enable**
  Specify whether or not this IPv4 address will be used.

  Note

  If you enable the IPv4 address, its settings in this DHCP tab are grayed out and can no longer be edited.

- **IP Address**
  Enter the IPv4 address that will be assigned via the specified port.
- **Subnet**
  Enter the subnet mask matching the IPv4 address. Use the CIDR notation.
- **Lease Time (sec)**
  Specify for how many seconds the assigned IPv4 address remains valid. When half the lease time has elapsed. the DHCP client can extend the period of the assigned IPv4 address. When the entire time has elapsed, the DHCP client needs to request a new IPv4 address.

Devices with several VLAN IP interfaces

On this page, specify the address band from which the connected device receives any IP address. You configure the static assignment of the IP addresses in "Static Leases".

Description

- DHCP Server
  Enable or disable the DHCP server on the device.

  Note

  To avoid conflicts with IPv4 addresses, only one device may be configured as a DHCP server in the network.

  If you want to operate a DHCP server on the devices of a VRRP group, note the information in the section "Layer 3 (IPv4/IPv6) > VRRP/VRRPv3 > Router".

- Probe address with ICMP Echo before offer
  When selected, the DHCP server checks whether or not the IP address has already been assigned. To do this the DHCP server sends ICMP echo messages (ping) to the IPv4 address. If no reply is received, the IPv4 address is assigned.

  Note

  This check is not made with static assignments.

  Note

  If there are devices in your network on which the echo service is disabled as default, there may be conflicts with the IPv4 addresses. To avoid this, assign these devices an IPv4 address outside the IPv4 address band.

The table has the following columns:

- Pool ID
  Shows the number of the IPv4 address band. If you click the "Create" button, a new row with a unique number is created (pool ID).
- Interface
  Select an IP interface or a router port. The IPv4 addresses are assigned dynamically via this interface.
- Enable
  Specify whether or not this IPv4 address band will be used.

  Note

  If you enable the IPv4 address, its settings in this DHCP tab are grayed out and can no longer be edited.

- Subnet
  Enter the subnet mask matching the IPv4 address. Use the CIDR notation.
- Lower IP Address
  Enter the IPv4 address that specifies the start of the dynamic IPv4 address band. The IPv4 address must be within the network address range you configured for "Subnet".

- **Upper IP address**
  Enter the IPv4 address that specifies the end of the dynamic IPv4 address band. The IPv4 address must be within the network address range you configured for "Subnet".
- **Lease Time (sec)**
  Specify for how many seconds the assigned IPv4 address remains valid. When half the period of validity has elapsed. the DHCP client can extend the period of the assigned IPv4 address. When the entire time has elapsed, the DHCP client needs to request a new IPv4 address.

Port-IP Address Mapping

On this page, you assign exactly one IP address to a certain port. After you have created an IPv4 address band in the "DHCP Server" tab, a new row is created in this tab. The configuration on this page has effects on the tabs "DHCP Server" and "Port Range".

Description

This table contains the following columns:

- **Pool ID**
  Shows the number of the IPv4 address band. A line is created for every address band.
- **Port**
  Select the setting from the drop-down list. You have the following setting options:
  - Px.y
    Specify the port via which IPv4 address will be assigned. You can only select ports located in the corresponding VLAN. If you select a port, only this port is enabled in the "Port Range" tab.
  - Not Selected
    With this setting, in the "Port Range" tab no ports or more than one port are selected. If you select the setting "Not Selected", all ports in the "Port Range" tab are disabled.
- **IP Address**
  Enter an IPv4 address. In the "DHCP Server" tab, the boxes "Lower IP Address" and "Upper IP Address" are filled accordingly.
- **Subnet Mask**
  Enter a corresponding subnet mask. In the "DHCP Server" tab, the "Subnet" box is filled accordingly.

Port Range

On this page, you define the ports via which the IPv4 addresses of an address band are assigned.

After you have created an IPv4 address band in the "DHCP Server" tab, a new line is created in this tab and all ports selected that are currently located in the corresponding VLAN. If you add ports to the VLAN later, the ports are not automatically enabled in this tab.

Description

This table contains the following columns:

- **Pool ID**
  Shows the number of the IPv4 address band. A line is created for every address band.
- **Interface**
  Shows the assigned VLAN IP interface.
- **All ports**
  Select the setting from the drop-down list. You have the following setting options:
  - Enabled
    The check box is enabled for all ports of the relevant VLAN.
  - Disabled
    The check box is disabled for all ports of the relevant VLAN.
  - No Change
    The table remains unchanged.
- **Px.y**
  Specify the ports via which IPv4 addresses of the address band will be assigned. You can only select ports located in the corresponding VLAN.

Note

Effects on other tabs

- If you enable precisely one port this is selected in the "Port-IP Address Mapping" tab.
- If you enable no port or more than one port in the "Port-IP Address Mapping" tab, the setting "Not Selected" is selected.

DHCP Options

On this page you specify which DHCP options the DHCP server supports. The various DHCP options are defined in RFC 2132.

The DHCP options 1, 3, 6, 66 and 67 are created automatically when the IPv4 address band is created. With the exception of DHCP option 1, the options can be deleted. With DHCP option 1 the subnet mask is set automatically that you entered for the address band in "DHCP Server". With the DHCP option 3, you can set the internal IPv4 address of the device as a DHCP parameter using a check box.

Description

The page contains the following boxes:

- **Pool ID**
  Select the required IPv4 address band.
- **Option Code**
  Enter the number of the required DHCP option. The various DHCP options are defined in RFC 2132. The supported DHCP options are listed in the following paragraph.

The table has the following columns:

- Select
  Select the check box in the row to be deleted.
- Pool ID
  Shows the number of the IPv4 address band.
- Option Code
  Shows the number of the DHCP option.
- Description
  Shows a description of the DHCP option.
- Use Interface IP
  If you enable the check box, the IPv4 address is used as the default gateway that is assigned to the VLAN IP address. If the check box is disabled, you can enter an IPv4 address.
- Value
  Enter the DHCP parameter that is transferred to the DHCP client. The content depends on the DHCP option.
  - DHCP option 3 (default gateway):
    Enter the DHCP parameter as an IPv4 address, e.g. 192.168.100.2.
  - DHCP option 6 (DNS):
    Enter the DHCP parameter as an IPv4 address, e.g. 192.168.100.2. You can specify up to three IPv4 addresses separated by commas.
  - DHCP option 12 (host name)
    Enter the host name in the string format.
  - DHCP option 66 (TFTP Server):
    Enter the DHCP parameter as an IPv4 address, e.g. 192.168.100.2.
  - DHCP option 67 (boot file name)
    Enter the name of the boot file in the string format.

DHCP options supported

The following DHCP options are supported:

- Option 1
- Option 3
- Option 6
- Option 12
- Option 66
- Option 67

Relay Agent Information

On this page you define that devices with a certain remote ID and circuit ID are assigned the IPv4 addresses from a specific address band.

If you create such an entry for an address band, the ports of the address band only react to DHCP queries via a DHCP relay agent (option 82). You can create further address bands for the same VLAN IP interfaces so that ports react to different requests.

Description

The page contains the following boxes:

- **Pool ID**
  Select the required IPv4 address band.
- **Remote ID**
  Enter the remote ID.
- **Circuit ID**
  Enter the circuit ID.

The table has the following columns:

- **Select**
  Select the check box in the row to be deleted.
- **Pool ID**
  Shows the number of the IPv4 address band.
- **Remote ID**
  Shows the remote ID.
- **Circuit ID**
  Shows the circuit ID.

Static Leases

On this page you define that DHCP clients are assigned a preset IPv4 address depending on their client ID or MAC address.

Description

The page contains the following boxes:

- **Pool ID**
  Select the required IPv4 address band.
- **Client identification method**
  Select the method according to which a client is identified.
  - Ethernet MAC
    The client is identified by its MAC address.
  - Client ID
    The client is identified by a freely defined DHCP client ID.
- **Value**
  Enter the MAC address (Ethernet MAC) or the client ID of the client.

The table has the following columns:

- Select
  Select the check box in the row to be deleted.
- Pool ID
  Shows the number of the IPv4 address band.
- Identification method
  Shows whether the client is identified by its MAC address or the client ID.
- Value
  Shows the MAC address or client ID of the client.
- IP Address
  Specify the IPv4 address that will be assigned to the client. The IPv4 address must be within the IPv4 address band.

Host options

On this page you specify which DHCP options the DHCP server supports. The various DHCP options are defined in RFC 2132.

The DHCP options 1, 3, 6, 66 and 67 are created automatically when the IPv4 address band is created. With the exception of DHCP option 1, the options can be deleted. With DHCP option 1 the subnet mask is set automatically that you entered for the address band in "DHCP Server". With the DHCP option 3, you can set the internal IPv4 address of the device as a DHCP parameter using a check box.

Description

The page contains the following boxes:

- Pool ID
  Select the required IPv4 address band.
- Client
  Select the device for which you want to set a DHCP option.
- Option Code
  Enter the number of the required DHCP option. The various DHCP options are defined in RFC 2132. The supported DHCP options are listed in the following paragraph.

The table has the following columns:

- Select
  Select the check box in the row to be deleted.
- Pool ID
  Shows the number of the IPv4 address band.
- Option Code
  Shows the number of the DHCP option.
- Description
  Shows a description of the DHCP option.

- Use Interface IP
  If you enable the check box, the IPv4 address is used as the default gateway that is assigned to the VLAN IP address. If the check box is disabled, you can enter an IPv4 address.
- Value
  Enter the DHCP parameter that is transferred to the DHCP client. The content depends on the DHCP option.
  - DHCP option 12 (host name)
    Enter the host name in the string format.
  - DHCP option 66 (TFTP Server):
    Enter the DHCP parameter as an IPv4 address, e.g. 192.168.100.2.
  - DHCP option 67 (boot file name)
    Enter the name of the boot file in the string format.

DHCPv6

DHCPv6 client

On this page, you specify whether the DHCPv6 client receives not only the configuration settings but also the IPv6 address or supports the "prefix delegation" function.

Prefix delegation

The DHCPv6 server delegates the distribution of the IPv6 prefix to the DHCPv6 client. To do this an IPv6 interface must be configured on the DHCPv6 client as a PD client. Via the PD client, the DHCPv6 client obtains an IPv6 prefix from the DHCPv6 server and stores it under a prefix name. The IPv6 interfaces of the DHCPv6 client the should adopt the prefix are configured as so-called PD sub clients.

You configure the PD sub-clients in "Layer 3 (IPv6)" > "DHCPv6 Client" > "DHCPv6 PD Sub Client".

Description of the displayed boxes

The page contains the following:

- Interface
  Select the IP interface via which the DHCPv6 client functions.
- Enable
  Enable or disable the DHCPv6 client for the relevant interface.
- Mode
  Specify the procedure.
  - Status dependent Obtains the IPv6 address and the configuration file from the DHCPv6 server
  - Stateless: Obtains only the configuration file from the DHCPv6 server
  - Prefix Delegation: The DHCPv6 client takes over the distribution of the IPv6 prefix

- **Prefix Name**
  Enter the prefix name that will be further distributed by the DHCPv6 router. Can only be edited with the "Prefix Delegation" procedure. The DHCPv6 router stores IPv6 prefix of the DHCPv6 server under this name.
- **Rapid Commit**
  When enabled the procedure for the IPv6 address assignment is shortened. Instead of 4 DHCPv6 messages (SOLICIT, ADVERTISE , REQUEST, REPLY) only 2 DHCPv6 messages (SOLICIT, REPLY) are used. You will find further information on the messages in RFC 3315.

DHCPv6 PD Sub Client

On this page you specify which IPv6 interfaces of the DHCPv6 client will adopt the prefix.

This achieved by assigning the prefix name followed by a prefix and the required prefix length to them as the IP address. Here, the first bits of the configured prefix are replaced by the values stored in the prefix name.

The PD sub client forms the IPv6 address from the prefix name, prefix and prefix length.

Example:

Under the prefix name "test", the following IPv6 prefix is stored: 2001:DB8:1234::/48

Prefix = 2002:AF9:5678::1

Prefix length = 64

From these settings the PD sub client forms the following IPv6 address: 2001:DB8:1234::1

Description of the displayed values

The page contains the following:

- **Interface**
  Select the interface that functions as the PD sub client. Only IP interfaces can be selected.
- **Prefix Name**
  Enter the prefix name that will be further distributed by the PD router. The PD router stores IPv6 prefix of the DHCPv6 server under this name.
- **Prefix**
  Specify the network identifier of the host.
- **Prefix Length**
  Enter the number of left-hand bits belonging to the prefix

This table contains the following columns:

- **Interface Name**
  Shows the name of the interface
- **Prefix Name**
  Shows the name of the prefix.
- **Prefix**
  Shows the prefix.

- **Prefix Length**
  Shows the prefix length.
- **Generated Address** (only available online)
  Shows the IPv6 address formed from the prefix name, prefix and prefix length.

SNMP

General

On this page, you make the basic settings for SNMP. Enable the options according to the function you want to use.

Description

- **SNMP**
  Select the SNMP protocol. The following settings are possible:
  - "-" (disabled)
    SNMP is disabled.
  - SNMPv1/v2c/v3
    SNMPv1/v2c/v3 is supported.

    Note

    Note that SNMP in versions 1 and 2c does not have any security mechanisms.

  - SNMPv3
    Only SNMPv3 is supported.
- **SNMPv1/v2 Read-Only**
  If you enable this option, SNMPv1/v2c can only read the SNMP variables.

  Note

  Community String

  For security reasons, do not use the standard values "public" or "private". Change the community strings following the initial installation.

  The recommended minimum length for community strings is 6 characters.

- **SNMPv1/v2c Read Community String**
  Enter the community string for access of the SNMP protocol.
- **SNMPv1/v2c Read/Write Community String**
  Enter the community string for read and write access of the SNMP protocol.
- **SNMPv1 Traps**
  Enable or disable the sending of SNMPv1 traps (alarm frames). In the "Trap" entry, specify the IP addresses of the devices to which SNMPv1 traps will be sent.
- **SNMPv1/v2c Trap Community String - Trap**
  Enter the community string for sending SNMPv1/v2c messages.

- **SNMPv3 User Migration**
  - **Enabled**
    If the function is enabled, an SNMP engine ID is generated that can be migrated. You can transfer configured SNMPv3 users to a different device.
    If you enable this function and load the configuration of the device on another device, configured SNMPv3 users are retained.
  - **Disabled**
    If the function is disabled, a device-specific SNMP engine ID is generated. To generate the ID, the agent MAC address of the device is used. You cannot transfer this SNMP user configuration to other devices.
    If you load the configuration of the device on another device, all configured SNMPv3 users are deleted.
- **SNMP Engine ID** (only available online)
  Shows the SNMP engine ID.

Traps

If an alarm event occurs, a device can send SNMP traps (alarm frames) to up to ten different management stations at the same time. Traps are only sent if the events specified in "Events" occur.

Note

SNMP traps are sent only when the "SNMPv1 Traps" setting was selected in "SNMP > General" or in "System > Configuration".

Description

- **Trap Receiver Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the station to which the device sends SNMP traps. You can specify up to ten different recipients.

The table has the following columns:

- **Trap Receiver Address**
  If necessary, change the IP addresses, the FQDN (Fully Qualified Domain Name) of the stations.
- **Trap**
  Enable or disable the sending of SNMP traps. Stations that are entered but not selected do not receive SNMP traps.

v3 Groups

Security settings and assigning permissions

SNMP version 3 allows permissions to be assigned, authentication, and encryption at protocol level. The security level and read/write permissions are defined according to groups. The settings automatically apply to every member of a group.

Description

- **Group Name**
  Enter the name of the group. The maximum length is 32 characters.
- **Security Level**
  Select the security level (authentication, encryption) valid for the selected group. The available options are as follows:
  - no Auth/no Priv
    No authentication enabled / no encryption enabled.
  - Auth/no Priv
    Authentication enabled / no encryption enabled.
  - Auth/Priv
    Authentication enabled / encryption enabled.

The table has the following columns:

- **Group Name**
  Shows the defined group names.

  Note

  Once a group name and the security level have been specified, they can no longer be modified after the group is created. If you want to change the group name or the security level , you will need to delete the group and recreate it and reconfigure it with the new name.

- **Security Level**
  Shows the configured security level.
- **Read**
  Enable or disable read access for the required group.
- **Write**
  Enable or disable write access for the required group.

  Note

  For write access to work, you also need to enable read access.

- **Persistence**
  Shows whether or not the group is assigned to an SNMPv3 user. If the group is not assigned to an SNMPv3 user, no automatic saving is triggered and the configured group is deleted after restarting the device.
  - Yes
    The group is assigned to an SNMPV3 user.
  - No
    The group is not assigned to an SNMPV3 user.

v3 users

User-specific security settings

On the page, you can create new SNMPv3 users and modify or delete existing users. The user-based security model works with the concept of the user name; in other words, a user ID is added to every frame. This user name and the applicable security settings are checked by both the sender and recipient.

Description

- User name
  Enter a freely selectable user name. After you have entered the data, you can no longer modify the name.

The table has the following columns:

- User name
  Shows the created users.
- Group Name
  Select the group to which the user will be assigned.
- Authentication Protocol
  Specify the authentication protocol for which a password will be stored.
  The following settings are available:
  - None
  - MD5
  - SHA
- Encryption Protocol
  Specify whether or not a password should be stored for encryption with the DES algorithm. Can only be enabled when an authentication protocol has been selected.
- Authentication Password
  Enter the authentication password. This password must have at least 1 character, the maximum length is 32 characters.

  Note

  Length of the password

  As an important measure to maximize security, we recommend that the password has a minimum length of 6 characters and that it contains special characters, uppercase/lowercase letters, numbers.

- Confirm authentication password
  Confirm the password by repeating the entry.

- Encryption Password
  Enter your encryption password. This password must have at least 1 character, the maximum length is 32 characters.

  Note
  Length of the password

  As an important measure to maximize security, we recommend that the password has a minimum length of 6 characters and that it contains special characters, uppercase/ lowercase letters and numbers.

- Confirm Encryption Password
  Confirm the encryption password by repeating the entry.
- Persistence
  Shows whether or not the user is assigned to an SNMPv3 group. If the user is not assigned to an SNMPv3 group, no automatic saving is triggered and the configured user is deleted after restarting the device.
  - Yes
    The user is assigned to an SNMPv3 group.
  - No
    The user is not assigned to an SNMPv3 group.

System time

There are different methods that can be used to set the system time of the device. Only one method can be active at any one time. If one method is activated, the previously activated method is automatically deactivated.

Manual time setting

Manual setting of the system time

On this page, you set the date and time of the system. For this setting to be used, enable "Manual time setting".

Note

The page is only available if there is an online connection to the device.

Description

- Time Manually
  Enable or disable the manual time setting. If you enable the option, the "System Time" input box can be edited.
- System Time
  Enter the date and time in the format "MM/DD/YYYY HH:MM:SS".

- Use PC Time
  Click the button to use the time setting of the PC.
- Last Synchronization Time
  Shows when the last time-of-day synchronization took place.
  If no time-of-day synchronization was possible, the box displays "Date/time not set".
- Last Synchronization Mechanism
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame.
  - PTP
    Automatic time-of-day synchronization with PTP
- Daylight Saving Time
  Shows whether the daylight saving time changeover is active.
  - active (offset + 1 h)
    The system time was changed to daylight saving time; in other words an hour was added.
    You can see the current system time at the top right in the selection area of the WBM.
    The set time continues to be displayed in the "System Time" box.
  - inactive (offset +0 h)
    The current system time is not changed.

DST overview

Daylight saving time switchover

On this page, you can create new entries for the daylight saving time changeover. The table provides an overview of the existing entries.

Description

- DST No.
  Shows the number of the entry.
  If you create a new entry, a new line with a unique number is created.
- Name
  Shows the name of the entry.
- Year
  Shows the year for which the entry was created.

- **Start Date**
  Shows the month, day and time for the start of daylight saving time.
- **End Date**
  Shows the month, day and time for the end of daylight saving time.
- **Recurring Date**
  With an entry of the type "Rule", the period in which daylight saving time is active is displayed consisting of week, day, month and time of day. With an entry of the type "Date" a "-" is displayed.
- **State**
  Shows the status of the entry:
  - Enabled
    The entry was created correctly.
  - Invalid
    The entry was created new and the start and end date are identical.
- **Type**
  Shows how the daylight saving time changeover is made:
  - Date
    A fixed date is entered for the daylight saving time changeover.
  - Recurring
    A rule was defined for the daylight saving time changeover.

DST configuration

Configuring the daylight saving time switchover

On this page, you can configure the entries for the daylight saving time changeover. As result of the changeover to daylight saving or standard time, the system time for the local time zone is correctly set. You can define a rule for the daylight saving time changeover or specify a fixed date.

Description

Note

The content of this page depends on the selection in the "Type" box.

The boxes "DST No.", "Type" and "Name" are always shown.

- **DST No.**
  Select the type of the entry.
- **Type**
  Select how the daylight saving time changeover is made:
  - Date
    You can set a fixed date for the daylight saving time changeover. This setting is suitable for regions in which the daylight saving time changeover is not governed by rules.
  - Recurring
    You can define a rule for the daylight saving time changeover. This setting is suitable for regions in which the daylight saving time always begins or ends on a certain weekday.
- **Name**
  Enter a name for the entry. The name can be a maximum of 16 characters long.

Settings with "Date" selected

You can set a fixed date for the start and end of daylight saving time.

- **Year**
  Enter the year for the daylight saving time changeover.
- **Start Date**
  Enter the following values for the start of daylight saving time:
  - Day
    Specify the day.
  - Hour
    Specify the hour.
  - Month
    Specify the month.
- **End Date**
  Enter the following values for the end of daylight saving time:
  - Day
    Specify the day.
  - Hour
    Specify the hour.
  - Month
    Specify the month.

Settings with "Recurring" selected

You can create a rule for the daylight saving time changeover.

- **Start Date**
  Enter the following values for the start of daylight saving time:
  - Hour
    Specify the hour.
  - Month
    Specify the month.
  - Week
    Specify the week.
    You can select the first to fourth or the last week of the month.
  - Weekday
    Specify the weekday.
- **End Date**
  Enter the following values for the end of daylight saving time:
  - Hour
    Specify the hour.
  - Month
    Specify the month.
  - Week
    Specify the week.
    You can select the 1st to 4th or the last week of the month.
  - Weekday
    Specify the weekday.

SNTP client

Time-of-day synchronization in the network

SNTP (Simple Network Time Protocol) is used for synchronizing the time in the network. The appropriate frames are sent by an SNTP server in the network.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Description

- **SNTP Client**
  Enable or disable automatic time-of-day synchronization using SNTP.
- **Current System Time** (only available online)
  Shows the current date and current normal time received by the IE switch. If you specify a time zone, the time information is adapted accordingly.

- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place.
- **Last Synchronization Mechanism** (only available online)
  Shows how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time. The time in the "Current System Time" box is adapted accordingly.
- **Daylight Saving Time**
  Shows whether the daylight saving time changeover is active.
  - active (offset +1 h)
    The system time was changed to daylight saving time; in other words an hour was added. The normal time including the time zone continues to be displayed in the "Current System Time" box.
    The normal time including the time zone continues to be displayed in the "Current System Time" box.
  - inactive (offset +0 h)
    The current system time is not changed.
- **SNTP Mode**
  Select the synchronization mode. The following types of synchronization are possible:
  - Listen
    With this mode, the device is passive and receives SNTP frames that deliver the time of day. Settings in the input boxes "SNTP Server Address" and "SNTP Server Port" have no effect in this mode. In this mode, only IPv4 addresses are supported.
  - Poll
    If you select this mode, the input box "Poll Interval[s]" is displayed to allow further configuration. In this mode the settings in the input boxes "SNTP Server Address" and "SNTP Server Port" are taken into account. With this type of synchronization, the device is active and sends a time query to the SNTP server. In this mode, IPv4 and IPv6 addresses are supported.
- **Poll Interval[s]**
  Here, enter the interval between two-time queries. In this box, you enter the query interval in seconds. Possible values are 16 to 16284 seconds.

- **SNTP Server Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the SNTP server.
- **SNTP Server Port**
  Enter the port of the SNTP server.
  The following ports are possible:
  – 123 (standard port)
  – 1025 to 36564
- **Primary**
  The check mark is set for the SNTP server that you create first. If several SNTP servers have been created, the primary server is queried first.

NTP client

Automatic time-of-day setting with NTP

If you require time-of-day synchronization using NTP, you can make the relevant settings here.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Description

- **NTP Client**
  Select this check box to enable automatic time-of-day synchronization with NTP.
- **Secure NTP Client Only**
  When enabled, the device receives the system time from a secure NTP server. The setting applies to all server entries.
  To enable the secure NTP client, you configure the parameters for authentication (key ID, hash algorithm, key).
- **Current System Time** (only available online)
  Shows the current date and current normal time received by the IE switch. If you specify a time zone, the time information is adapted accordingly.
- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place.

- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time.
  The time in the "Current System Time" box is adapted accordingly.
- **Daylight Saving Time**
  Shows whether the daylight saving time changeover is active.
  - active (offset +1 h)
    The system time was changed to daylight saving time; in other words an hour was added. The normal time including the time zone continues to be displayed in the "Current System Time" box.
  - inactive (offset +0 h)
    The current system time is not changed.
- **NTP Server Index**
  Select the index of the NTP server. The server with the lowest index is queried first.
- **NTP Server Address**
  Enter the IPv4 address of the NTP server.
- **NTP Server Port**
  Enter the port of the NTP server.
  The following ports are possible:
  - 123 (standard port)
  - 1025 to 36564
- **Poll Interval[s]**
  Specify the interval between two time queries. Possible values are 64 to 1024 seconds.
- **Key ID**
  Enter the ID of the authentication key.
- **Hash Algorithm**
  Specify the format for the authentication key.

- **Key**
  Enter the authentication key.
- **Key Confirmation**
  Enter the authentication key again to confirm it.

SIMATIC Time Client

Time setting via SIMATIC time client

On this page, you configure time synchronization with the SIMATIC Time Client.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Description

- **SIMATIC Time Client**
  Select this check box to enable the device as a SIMATIC time client.
- **Current System Time** (only available online)
  Shows the current system time.
- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place.
- **Last Synchronization Mechanism** (only available online)
  Shows how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP

PTP Client

Automatic time-of-day setting with PTP

If you require time-of-day synchronization using PTP, you can make the relevant settings here.

The following devices support time-of-day synchronization with PTP:

- SCALANCE XR528-6M
- SCALANCE XR552-12M

Description

- **PTP Client**
  Select this check box to enable automatic time-of-day synchronization with PTP.
- **Current System Time** (only available online)
  Shows the current date and current normal time obtained due to time synchronization in the network. If you specify a time zone, the time information is adapted accordingly.
- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".
- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP

- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time. The time in the "Current System Time" box is adapted accordingly.
- **Daylight Saving Time**
  Shows whether the daylight saving time changeover is active.
  - active (offset +1 h)
    The system time was changed to daylight saving time; in other words an hour was added. The normal time including the time zone continues to be displayed in the "Current System Time" box.
  - inactive (offset +0 h)
    The current system time is not changed.

NTP server

Configuring the device as NTP server

On this page, you configure the device as an NTP server. The other devices can call up the current time via this NTP server. This means that the supplied devices are not dependent on a connection to an external time server.

Note

Time synchronization

Also configure the device as NTP client so that it synchronizes the connected devices to a correct time. As NTP client, the device gets the precise time from an external time server and distributes it to its NTP clients as NTP server.

Description

- **NTP Server**
  Enable or disable the NTP server service.

  Note

  The SNTP mode "Listen" of the SNTP client and the NTP server cannot be activated at the same time.

Table 1 has the following columns:

- **At all interfaces**
  Shows that the settings are valid for all interfaces of table 2.
- **Listen**
  In the drop-down list, select the setting for all interfaces. If "No Change" is selected, the entries of the corresponding column in table 2 remain unchanged.
- **Copy to Table**
  If you click the button, the setting is adopted for all interfaces of table 2

Table 2 has the following columns

- Interface
  Via this interface the time is transferred using NTP.
- Listen
  When enabled, the other devices can call up the time via this interface.

Automatic Logout

Setting the automatic logout

On this page, set the times after which there is an automatic logout from the WBM or the CLI following user inactivity. If you have been logged out automatically, you will need to log in again.

Configuration

- Web Based Management [s]
  Enter a value of 60-3600 seconds in this input box. If you enter the value 0, the automatic logout is disabled.
- CLI (TELNET, SSH, Serial) [s]
  Enter a value of 60-600 seconds in this input box. If you enter the value 0, the automatic logout is disabled.

Button

The "SELECT/SET" button is used for the following:

- Changing the display mode,
- Resetting to factory defaults,
- Defining the fault mask and the LED display.

You will find a detailed description of the individual functions available with the buttons in the device operating instructions. On this page, the functionality of the SELECT/SET button can be restricted or fully disabled.

Descriptions

- **Restore Factory Defaults**
  Enable or disable the "Restore Factory Defaults" function with the SELECT/SET button.

  ⚠ CAUTION

  Button function "Restore Factory Defaults" active during startup

  If you have disabled this function in your configuration, disabling is only valid during operation. When restarting, for example after power down, the function is active until the configuration is loaded so that the device can inadvertently be reset to the factory settings. This may cause unwanted disruption in network operation since the device needs to be reconfigured if this occurs. An inserted PLUG is also deleted and returned to the status as shipped

  ⚠ CAUTION

  Resetting with WBM and CLI

  If you have disabled the "Restore Factory Defaults" function in your configuration, the disabling does not apply to the CLI command "Restart" and for the WBM page "System > Restart". This means that the device can inadvertently be reset to the factory defaults. This may cause unwanted disruption in network operation since the device then needs to be reconfigured. An inserted PLUG is also deleted and returned to the status as shipped.

- **Set Fault Mask**
  Enable or disable the function "Define fault mask via the LED display" with the SELECT/SET button.

Syslog client

System event agent

Syslog according to RFC 3164 is used for transferring short, unencrypted text messages over UDP in the IP network. This requires a Syslog server.

Requirements for sending log entries

- The Syslog function is enabled on the device.
- The Syslog function is enabled for the relevant event.
- There is a Syslog server in your network that receives the log entries. Since this is a UDP connection, there is no acknowledgment to the sender.
- The IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server is entered in the device.

Description

The page contains the following boxes:

- Syslog Client
  Enable or disable the Syslog function.
- Syslog Server Address
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server.

The table contains the following columns:

- Syslog Server Address
  Shows the IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server.
- Server Port
  Enter the port of the Syslog server being used.

Ports

Overview

Overview of the port configuration

The page shows the configuration for the data transfer for all ports of the device. You cannot configure anything on this page.

Description of the displayed boxes

The table has the following columns:

- Port
  Shows the configurable ports. The entry is a link. If you click on the link, the corresponding configuration page is opened. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- Port name
  Shows the name of the port.
- Port Type (only with routing)
  Shows the type of the port. The following types are possible:
  - Router Port
  - Switch Port VLAN Hybrid
  - Switch-Port VLAN Trunk
  - Switch Port VLAN Host
  - Switch-Port PVLAN Promiscuous
  - Switch Port VLAN Access

- **Combo Port Media Type**
  The values are only available with combo ports. Shows the mode of the combo port:
  - auto
  - rj45
  - sfp
- **Status**
  Shows whether the port is on or off. Data traffic is possible only over an enabled port.
- **OperState**
  Displays the current operational status. The operational status depends on the configured "Status" and the "Link". The available options are as follows:
  - up
    You have configured the status "enabled" for the port and the port has a valid connection to the network.
  - down
    You have configured the status "disabled" or "Link down" for the port or the port has no connection.
  - not present
    With modular devices, this status is displayed when, for example, no media module is inserted.
- **Link** (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - Up
    The port has a valid link to the network, a link integrity signal is being received.
  - Down
    The link is down, for example because the connected device is turned off.
- **Mode** (only available online)
  Shows the transfer parameters of the port
- **MTU** (Maximum Transmission Unit)
  Shows the maximum packet size.
- **Negotiation**
  Shows whether the automatic configuration is enabled or disabled.
- **Flow Ctrl. Type**
  Shows whether flow control is enabled or disabled for the port.
- **Flow Ctrl.**
  Shows whether or not flow control is working on this port.

- **MAC Address** (only available online)
  Shows the MAC address of the port.
- **Blocked by**
  Shows why the port is in the "blocked" status:
  - -
    The analysis of the port status Is not supported.
  - Ring redundancy
    The port belongs to a redundancy manager. When the redundancy manager is in the "Passive status, one of the ring ports is in the "blocking" status.
  - Spanning Tree
    The port has the status "Discarding" in the spanning tree. The port is part of a spanning tree, however it is located on a redundant path and is deactivated for data traffic.
  - Loop Detection
    A loop was detected and as the reaction to a loop, the status "disable" was configured for the port.
  - Link Check
    A disruption was detected on an optical transmission link and as the reaction the port status "disable" was configured.
  - Link Aggregation Member
    The port is part of a link aggregation and was deactivated by LACP.
  - Link Aggregation (LoopD)
    The port is part of a link aggregation. A loop was detected and as the reaction to a loop, the status "disable" was configured for the link aggregation.
  - Link Aggregation (STP)
    The port is part of a link aggregation. The link aggregation was switched to the status "Discarding" by the spanning tree.
  - Admin down
    The status "disabled" is configured for the port, see "System > Ports > Configuration".
  - Link down
    The status "enabled" is configured for the port but there is no connection, see "System > Ports > Configuration".
  - Power down
    The status “Link down" is configured for the port, see "System > Ports > Configuration".
  - Standby
    Standby redundancy is enabled on the device. The port is a standby port with the status "Passive".

Deviating display of the transmission parameters with combo ports

In the connection status "down", the displayed transmission parameters do not match the actual values of the combo port. In the connection status "up", the correct values are displayed.

Initial situation

A pluggable transceiver is plugged into the combo port with the following settings:

- Combo Port Media Type: auto
- Status: enabled
- Link: down

Display of the transmission parameters

With 100 Mbps pluggable transceivers

- Actual response: Mode: 1G HD
- Expected response: Mode: 100M FD

With 1 Gbps pluggable transceivers

- Actual response: Mode: 1G HD
- Expected response: Mode: 1G FD

Configuration

Configuring ports

With this page, you can configure all the ports of the device.

Description

The table has the following rows:

- **Port**
  Select the port to be configured. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Status**
  Specify whether the port is enabled or disabled.
  - enabled
    The port is enabled. Data traffic is possible only over an enabled port.
  - disabled
    The port is disabled but the connection remains.
  - Link down
    The port is disabled and the connection to the partner device is terminated.
- **Port name**
  Enter a name for the port.
- **MAC Address** (only available online)
  Shows the MAC address of the port.

- **Mode Type**(only available online)
  Select the transmission speed and the transmission method of the port. If you set the mode to "Auto negotiation", these parameters are automatically negotiated with the connected end device. This must also be in the "Auto negotiation" mode.

  Note

  Before the port and partner port can communicate with each other, the settings must match at both ends.

  Note

  "Mode Type" with combo ports

  To be able to set the "Mode Type" of a combo port, change the "Combo Port Media Type" to "rj45". If "auto" is set for the "Combo Port Media Type" and the RJ-45 port is used, you cannot set the "Mode Type".

- **Mode** (only available online)
  Shows the transmission speed and mode of the port.
  The transmission speed can be 10 Mbps, 100 Mbps, 1000 Mbps or 10 Gbps. As the transmission mode, you can configure full duplex (FD) or half duplex (HD).
- **Negotiation**
  Shows whether the automatic configuration of the connection to the partner port is enabled or disabled.
- **Flow Ctrl. Type**
  Enable or disable flow control for the port.

  Note

  Turning flow control on/off with autonegotiation

  Flow control can only be enabled or disabled if the "autonegotiation" function is turned off. The function cannot enabled again afterwards.

- **Flow Ctrl.**
  Shows whether flow control is working on this port.
- **MTU**
  Enter the packet size.

- **Port Type** (only with routing)
  Select the type of the port:
  - Router Port
    The port is a layer 3 interface. It does not support layer 2 functions.
  - Switch Port VLAN Hybrid
    The port sends tagged and untagged frames. It is not automatically a member of a VLAN.
  - Switch-Port VLAN Trunk
    The port only sends tagged frames and is automatically a member of all VLANs.
  - Switch Port VLAN Host
    Host ports belong to a secondary PVLAN. Connect devices to host ports that are only intended to communicate with certain devices of the PVLAN.
  - Switch-Port PVLAN Promiscuous
    Promiscuous ports belong to a primary PVLAN. Connect devices to promiscuous ports that are intended to communicate all devices of the PVLAN.
  - Switch Port VLAN Access
    Access ports belong to a provider switch that supports the function Q-in-Q VLAN tunnel. Connect a customer network to access ports.
- **Combo Port Media Type**
  Specify the mode of the combo port:
  - auto
    If you select this mode, the SFP transceiver port has priority. As soon as an SFP transceiver is plugged in, an existing connection at the fixed RJ-45 port is terminated. If no SFC transceiver is plugged in, a connection can be established via the fixed RJ-45 port.
  - rj45
    If you select this mode, the fixed RJ-45 port is used regardless of the SFP transceiver port. If a pluggable transceiver is plugged in, it is disabled and the power turned off.
  - sfp
    If you select this mode, the SFP transceiver port is used regardless of the built-in RJ-45 port. If an RJ-45 connection is established, it is terminated because the power of the RJ-45 port is turned off.

  The factory setting for the combo ports is the auto mode.

  Note

  Automatic adaptation due to PROFINET configuration

  When establishing a PROFINET connection, the setting of the combo port media type is adapted automatically:
  - If a pluggable transceiver is configured, the combo port media type will be set to "sfp".
  - If a the fixed RJ-45 port is configured, the combo port media type will be set to "rj45".

  So that the automatic adaptation can be made, the combo port media type must be set to "auto". Configure the combo port media type accordingly using the WBM or CLI.

- OperState
  Displays the current operational status. The operational status depends on the configured "Status" and the "Link". The available options are as follows:
  - up
    You have configured the status "enabled" for the port and the port has a valid connection to the network.
  - down
    You have configured the status "disabled" or "Link down" for the port or the port has no connection.
  - not present
    With modular devices, this status is displayed when, for example, no media module is inserted.

- **Link** (only available online)
  Shows the connection status to the network. The available options are as follows:
  - Up
    The port has a valid link to the network, a link integrity signal is being received.
  - Down
    The link is down, for example because the connected device is turned off.
- **Blocked by**
  Shows why the port is in the "blocked" status:
  - -
    The analysis of the port status Is not supported.
  - Ringredundanz
    The port belongs to a redundancy manager. When the redundancy manager is in the "Passive status, one of the ring ports is in the "blocking" status.
  - Spanning Tree
    The port has the status "Discarding" in the spanning tree. The port is part of a spanning tree, however it is located on a redundant path and is deactivated for data traffic.
  - Loop Detection
    A loop was detected and as the reaction to a loop, the status "disable" was configured for the port.
  - Link Check
    A disruption was detected on an optical transmission link and as the reaction the port status "disable" was configured.
  - Link Aggregation member
    The port is part of a Link Aggregation and was deactivated by LACP.
  - Link Aggregation (LoopD)
    The port is part of a link aggregation. A loop was detected and as the reaction to a loop, the status "disable" was configured for the link aggregation.
  - Link Aggregation (STP)
    The port is part of a link aggregation. The link aggregation was switched to the status "Discarding" by the spanning tree.
  - Admin down
    The status "disabled" is configured for the port, see "System > Ports > Configuration".
  - Link Down
    The status "enabled" is configured for the port but there is no connection, see "System > Ports > Configuration".
  - Power Down
    The status “Link down" is configured for the port, see "System > Ports > Configuration".
  - Standby
    Standby redundancy is enabled on the device. The port is a standby port with the status "Passive".

Changing the port configuration

Click the appropriate box to change the configuration.

Note

Optical ports only work with the full duplex mode and at maximum transmission rate. As a result, the following settings cannot be made for optical ports:

- Automatic configuration
- Transmission speed
- Transmission technique

Note

With various automatic functions, the device prevents or reduces the effect on other ports and priority classes (Class of Service) if a port is overloaded. This can mean that frames are discarded even when flow control is enabled.

Port overload occurs when the device receives more frames than it can send, for example as the result of different transmission speeds.

Fault Monitoring

Power supply

Settings for monitoring the power supply

Configure whether or not the power supply should be monitored by the messaging system. Depending on the hardware variant, there are one or two power connectors (Supply 1 / Supply 2). With a redundant power supply, configure the monitoring separately for each individual feed-in line.

If there is no power on one of the monitored lines (Supply 1 or Supply 2) or when the voltage is too low, a fault is signaled by the signaling system.

Note

You will find the permitted operating voltage limits in the compact operating instructions of the device.

A fault causes the signaling contact to trigger and the fault LED on the device to light up and, depending on the configuration, can trigger a trap, an e-mail, or an entry in the event log table.

Settings

- Line 1
  Enable or disable the monitoring of power connector 1.
- Line 2
  Enable or disable the monitoring of power connector 2.

Link Change

Configuration of fault monitoring of status changes on connections

On this page, you configure whether or not an error message is triggered if there is a status change on a network connection.

If connection monitoring is enabled, an error is signaled

- when there should be a link on a port and this is missing.
- or when there should not be a link on a port and a link is detected.

An error causes the signaling contact to trigger and the fault LED to light up on the device. Depending on the configuration, the error may trigger a trap, an e-mail or an entry in the event log table.

Description of the displayed boxes

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Setting**
  Select the required setting. The available options are as follows:
  - "-" (disabled)
  - Up
  - Down
  - No Change: The setting in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns

- **Port**
  Shows the available ports.
- **Setting**
  Select the setting. You have the following options:
  - Up
    Error handling is triggered when the port changes to the active status.
    (From "Link down" to "Link up")
  - Down
    Error handling is triggered when the port changes to the inactive status.
    (From "Link up" to "Link down")
  - "-" (disabled)
    The error handling is not triggered.

Redundancy

On this page, you configure whether or not an error message is triggered if there is a status change on a redundant connection.

Setting

- **Redundancy loss (HRP only)**
  Enable or disable connection monitoring. If the redundancy of the connection is lost, an error is signaled.

PROFINET

Settings for PROFINET

On this page, you configure the mode of PROFINET.

Description of the displayed boxes

The page contains the following boxes:

- **PNIO Device Diagnostics**
  Shows whether PNIO is enabled ("On") or disabled ("Off").
- **PNIO Device Diagnostics for next boot**
  Set whether PNIO will be enabled ("On") or disabled ("Off") after the next device restart.

  Note

  PNIO and EtherNet/IP

  When PNIO is turned on, EtherNet/IP is turned off. The switchover from PNIO and EtherNet/IP has no effect on DCP.

  Note

  PNIO AR Status

  If a PROFINET connection is established; in other words the PNIO AR status is "Online", you cannot disable PNIO.

- **PNIO AR Status**
  This box shows the status of the PROFINET IO connection; in other words whether the device is connected to a PROFINET IO controller "Online " or "Offline".
  Here, online means that a connection to a PROFINET IO controller exists, that this has downloaded its configuration data to the device and that the device can send status data to the PROFINET IO controller. In this status known as "in data exchange", the parameters set via the PROFINET IO controller cannot be configured.
- **PNIO Name of Station**
  This box displays the PROFINET IO device name according to the configuration in HW Config of STEP 7.

- **Restart with PROFINET Defaults**
  Click this button to restore the default settings of the PROFINET IO profile and to restart the device. You must confirm the restart in a dialog box. The dialog box displays the settings specially made for operation with the PROFINET protocol.

NOTICE
Loss of the IP address By resetting the settings to the default settings of a profile, the IP address is also lost. Following this, the device can only be accessed via the serial interface, using the Primary Setup Tool or using DHCP. With the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

- **Allow PNIO Data Exchange**(only with SCALANCE XB-200)
  Enable or disable PNIO data exchange
- **Simulate PNIO Data Exchange**(only with SCALANCE XB-200)
  Enable or disable the simulation of PNIO data exchange.

EtherNet/IP

EtherNet Industrial Protocol (EtherNet/IP)

On this page, you configure the mode of EtherNet/IP.

Description of the displayed boxes

The page contains the following boxes:

- **EtherNet/IP Device Diagnostics**
  Shows whether EtherNet/IP is enabled ("On") or disabled ("Off").
- **EtherNet/IP Device Diagnostics for next boot**
  Set whether EtherNet/IP will be enabled ("On") or disabled ("Off") after the next device restart.

  Note

  EtherNet/IP and PROFINET

  When EtherNet/IP is turned on, PROFINET is turned off. The switchover from EtherNet/IP and PROFINET has no effect on DCP.

  Note

  PROFINET AR Status

  If a PROFINET connection is established; in other words the PROFINET AR status is "Online", you cannot enable EtherNet/IP.

- **Restart with EtherNet/IP Defaults**
  Click this button to restore the default settings of the EtherNet/IP profile and to restart the device. You must confirm the restart in a dialog box. The dialog box displays the settings specially made for operation with the EtherNet/IP protocol.

NOTICE
Loss of the IP address By resetting the settings to the default settings of a profile, the IP address is also lost. Following this, the device can only be accessed via the serial interface, using the Primary Setup Tool or using DHCP. With the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

PLUG

PLUG configuration

Note

The page is only available if there is an online connection to the device.

NOTICE
Do not remove or insert a C-PLUG / KEY-PLUG during operation! A PLUG may only be removed or inserted when the device is turned off. The device checks whether or not a PLUG is present at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid KEY-PLUG was inserted in the device, the device changes to a defined error state following the restart.

Information about the configuration of the C-PLUG / KEY-PLUG

This page provides detailed information about the configuration stored on the C-PLUG or KEY-PLUG. It is also possible to reset the PLUG to "factory defaults" or to load it with new contents.

Note

The action is only executed after you click the "Set Values" button.

The action cannot be undone.

If you decide against executing the function after making your selection, click the "Refresh" button. As a result the data of this page is read from the device again and the selection is canceled.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Settings

The table has the following rows:

- **Status**
  Shows the status of the PLUG. The following are possible:
  - ACCEPTED
    There is a PLUG with a valid and suitable configuration in the device.
  - NOT ACCEPTED
    Invalid or incompatible configuration on the inserted PLUG.
  - NOT PRESENT
    There is no C-PLUG or KEY-PLUG inserted in the device.
  - FACTORY
    PLUG is inserted and does not contain a configuration. This status is also displayed when the PLUG was formatted during operation.
  - MISSING
    There is no PLUG inserted. Functions are configured on the device for which a license is required.
- **Device Group**
  Shows the SIMATIC NET product line that used the C-PLUG or KEY-PLUG previously.
- **Device Type**
  Shows the device type within the product line that used the C-PLUG or KEY-PLUG previously.
- **Configuration Revision**
  The version of the configuration structure. This information relates to the configuration options supported by the device and has nothing to do with the concrete hardware configuration. This revision information does not therefore change if you add or remove additional components (modules or extenders), it can, however, change if you update the firmware.
- **File System**
  Displays the type of file system on the PLUG.

NOTICE
New file system UBI As of firmware version 3.0, UBI is the standard file system for the C-PLUG or KEY-PLUG. If a C-PLUG with the previous file system IECP is detected in such a device, this C-PLUG will be formatted for the UBI file system and the data will be rewritten to the C-PLUG. This change in the file system also occurs following a firmware update to V3.0. A downgrade to the previous version of the firmware then presents a problem. The firmware can neither read nor write the C-PLUG or KEY-PLUG and it is not even possible to "Erase PLUG to factory default".

- **File System Size [bytes]**
  Shows the maximum storage capacity of the file system on the C-PLUG.
- **File System Usage**
  Displays the storage space in use in the file system of the C-PLUG.

- Info String
  Shows additional information about the device that used the PLUG previously, for example, article number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.
- Firmware on PLUG
  When the function enabled, the firmware will be stored on the PLUG. This means that automatic firmware updates/downgrades can be made with the PLUG. The box "Info String" shows whether or not the firmware is stored on the PLUG.
- Modify PLUG
  Select the required setting. You have the following options for changing the configuration on the C-PLUG or KEY-PLUG:
  - Write current configuration to PLUG
    This option is available only if the status of the PLUG is "NOT ACCEPTED" or FACTORY. The configuration in the internal flash memory of the device is copied to the PLUG.
  - Erase PLUG to factory default
    Deletes all data from the C-PLUG and triggers low-level formatting.

PLUG license

Note

The page is only available if there is an online connection to the device.

NOTICE

Do not remove or insert a C-PLUG / KEY-PLUG during operation!

A PLUG may only be removed or inserted when the device is turned off.
The device checks whether or not a PLUG is present at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid KEY-PLUG was inserted in the device, the device changes to a defined error state following the restart.

If the device was configured at some time with a PLUG, the device can no longer be used without this PLUG. To be able to use the device again, reset the device to the factory settings.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Information about the license of the KEY-PLUG

A C-PLUG can only store the configuration of a device. In addition to the configuration, a KEY-PLUG also contains a license that enables certain functions of your SIMATIC NET device.

This page provides detailed information about the license on the KEY-PLUG.

Description of the displayed boxes

- Status
  Shows the status of the KEY-PLUG. The following are possible:
  - ACCEPTED
    The KEY-PLUG in the device contains a suitable and valid license.
  - NOT ACCEPTED
    The license of the inserted KEY-PLUG is not valid.
  - NOT PRESENT
    No KEY-PLUG is inserted in the device.
  - MISSING
    There is no KEY-PLUG or a C-PLUG with the status "FACTORY" inserted in the device. Functions are configured on the device for which a license is required.
  - WRONG
    The inserted KEY-PLUG is not suitable for the device.
  - UNKNOWN
    Unknown content of the KEY-PLUG.
  - DEFECTIVE
    The content of the KEY-PLUG contains errors.
- Order ID
  Shows the order number of the KEY-PLUG. The KEY-PLUG is available for various functional enhancements and for various target systems.
- Serial Number
  Shows the serial number of the KEY-PLUG.
- Info String
  Shows additional information about the device that used the KEY-PLUG previously, for example, order number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.

Note

When you save the configuration, the information about whether or not a KEY-PLUG was inserted in the device at the time is also saved. This configuration can then only work if a KEY-PLUG with the same article number / license is inserted.

Ping

Reachability of an address in an IP network

Note

The page is only available if there is an online connection to the device.

With the ping function, you can check whether a certain address is reachable in the network.

Description

The page contains the following boxes:

- **Dest. address**
  Enter the IPV4, IPv6 address or the FQDN (Fully Qualified Domain Name) of the device.
- **Repeat**
  Enter the number of ping requests.
- **DNS Resolution**
  Select the IP address type in which an entered FQDN will be resolved.
  - Auto
    In this mode, the IP address type is selected automatically.
  - IPv4
    The entered FQDN will be resolved in an IPv4 address.
  - IPv6
    The entered FQDN will be resolved in an IPv6 address.
- **Out Interface for IPv6**
  This selection is only required when the destination address is a multicast or a link local address.
  - "-" (factory setting)
  - Select the relevant IPv6 interface.
- **Ping**
  Click this button to start the ping function.
- **Ping Output**
  This box shows the output of the ping function.
- **Delete**
  Click this button to delete the ping output.

DCP Discovery

Search for devices reachable via the selected interface

On this page, you can select an interface and search for devices that are reachable via the interface. The reachable devices are listed in a table. In the table, you can check and adapt the network parameters of the devices. To identify and configure the devices, the Discovery Configuration Protocol (DCP) is used.

Note

DCP Discovery

The function is only available in the VLAN associated with the TIA interface. You can configure the TIA interface under "Layer 3 > Subnets > Configuration".

Requirement:

To adapt network parameters, DCP requires write access to the device. If access is write-protected, the network parameters cannot be configured.

On the SCALANCE devices, you configure the access in "System > Configuration".

Description of the displayed values

- **Interface**
  Select the required interface.
- **Browse**
  Starts the search for devices reachable via the selected interface.
  On completion of the search, the reachable devices are listed in the table. The table is limited to 100 entries.

The table has the following columns:

- **Port**
  Shows the port via which the device can be reached.
- **MAC Address**
  Shows the MAC address of the device.
- **Device Type**
  Shows the product line or product group to which the device belongs.
- **Device Name**
  If the device supports this function, you can assign a PROFINET device name to the device.
- **IP Address**
  If necessary, adapt the IPv4 address of the device.
  The IPv4 address should be unique within your network and should match the network.
  The IPv4 address 0.0.0.0 means that no IPv4 address has yet been set.
- **Subnet mask**
  If necessary, adapt the subnet mask of the device.
- **Gateway Address**
  If necessary, specify the IPv4 address of the gateway.

- **Status Device Name**
  - Discovered: The set device name is used.
  - Configured: The device was assigned a new device name.
- **Status IP Address**
  - Discovered/IP: The device uses a static IPv4 address.
  - Discovered/DHCP: The device has obtained the IPv4 address from a DHCP server.
  - Configured: The device was assigned a new IPv4 address.
- **Timeout**
  Specify the time for flashing. When the time elapses, flashing stops.
- **Flash**
  Makes the port LEDs of the selected device flash.

PoE

General

Note

The menu item “PoE” is only displayed for devices that support PoE

Settings for Power over Ethernet (PoE)

On this page, you see information about the power that the IE switch supplies with PoE.

The SCALANCE X-500 represents a PSE (Power Sourcing Equipment). With the SCALANCE XM400, each group of four ports with PoE capability is known as a PSE. The displayed values apply only to the corresponding PSE.

Description of the displayed boxes

- **PSE** (readonly)
  Shows the number of the PSE.
- **Maximum Power [W]** (read-only)
  Maximum power that a PSE provides to supply PoE devices. The "Maximum Power" value can be set for a SCALANCE XM-400.
- **Allocated Power [W]** (read-only)
  Sum of the power reserved by the PoE devices according to the "Classification".
- **Power in Use [W]** (read-only)
  Sum of the power used by the end devices.
- **Usage Threshold [%]**
  If the power being used by the end devices exceeds the percentage shown here, an event is triggered.

Power over Ethernet with SCALANCE XM-400

With a SCALANCE XM-400, you can use the "Power over Ethernet" function via the port extender PE408PoE.

PoE power supply

The connection of the PoE power supply is external. You can connect 2 PoE power supplies with each PE408PoE port extender. Each PE408PoE therefore has 2 PSE units (Power Sourcing Equipment) each with 4 ports.

Numbering of the PSE units

To be able to differentiate between the PSE units in the configuration, they are numbered:

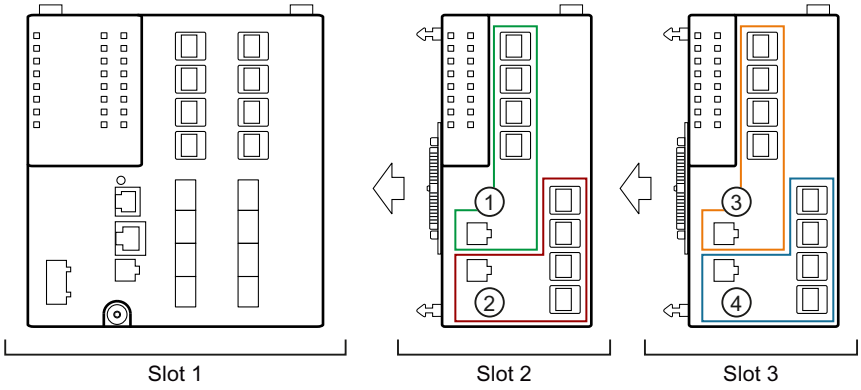

Figure 1-86 SCALANCE XM-400 with 2 PE408PoE port extenders

If a PE408PoE is inserted in slot 2, its two PSE slots have indexes 1 and 2. If a PE408PoE is inserted in slot 3, its two PSE slots have indexes 3 and 4.

The numbering of the PSE units is decided by the slots. If there is a port extender without PoE in slot 2, and there is a PE408PoE in slot 3, the PSE units in slot 3 still have the indexes 3 and 4.

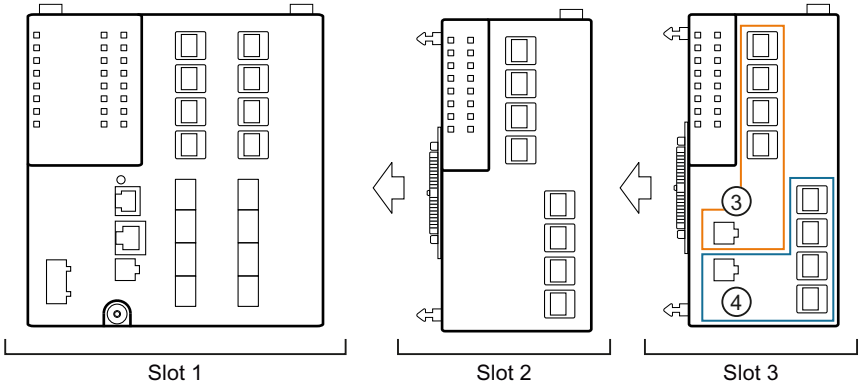

Figure 1-87 SCALANCE XM-400 with 2 port extenders (1 port extender without PoE and 1 PE408PoE)

With a SCALANCE XM416-4C with which you can only connect one port extender, the indexes of the PSE units are 1 and 2.

Power over Ethernet with SCALANCE XR528-6M and SCALANCE XR552-12M

With SCALANCE XR528-6M and SCALANCE XR552-12M only the module slots 1 to 3 can be fitted with PoE modules.

Port

Settings for the ports

For each individual PoE port, you can specify whether or not the power will be supplied via Ethernet. You can also set a priority for each connected powered device (PD). Devices for which a high priority was set, take preference over other devices for the power supply. On this page, you see detailed information about the individual PoE ports.

Description of the displayed boxes

The page contains two tables. In table 1, you can make settings and assign them to all ports at the same time. In table 2, you can make different settings for each port.

Table 1 has the following columns:

- Port
  Shows that the settings are valid for all ports.
- Setting
  Select the required setting. Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Priority
  Select the required priority of the port. Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Type
  Enter a string that describes the connected device in greater detail. The maximum length is 255 characters.
- Use Custom Maximum Power
  Select whether the custom maximum power is used. If "No Change" is selected, the entry in table 2 remains unchanged.
- Custom Maximum Power [W]
  Enter the maximum power that a port makes available to supply a connected device. This value is only taken into account when the "Use Custom Maximum Power" check box is selected. If "No Change" is entered, the entry in table 2 remains unchanged
- Copy to Table
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- Port
  Shows the configurable PoE ports. The port is made up of the port number and the slot number, for example port 0.1 is slot 0, port 1.
- Setting
  Enable the PoE power supply for this port or interrupt it.

- **Priority**
  Select which priority this port will have for the power supply. The following settings are possible, in ascending order of relevance:
  - Low
  - High
  - Critical

  If the power of the connected power supply is inadequate to supply all connected devices, devices with a higher priority are given preference. If the same priority is set for two ports, the port with the lower number will be preferred when necessary.
- **Type**
  Here, you can enter a string to describe the connected device in greater detail. The maximum length is 255 characters.
- **Use Custom Maximum Power**
  If you enable this check box for a port, the user-defined maximum power is used.
- **Custom Maximum Power [W]**
  Enter the maximum power that a port makes available to supply a connected device. This value is only taken into account when the "Use Custom Maximum Power" check box is selected. The user-defined power is compared to the range of values of the class indicated by the connected device.
  - If the user-defined power is within the class of the connected device, the user-defined value is used.
  - If the user-defined power is above the class of the connected device, the highest value of the class is used.
  - If the user-defined power is below the class of the connected device, the lowest value of the class is used.

  If the power consumption of the connected device exceeds the defined or used maximum power, the connected device is turned off.
- **Classification** (read-only)
  The classification specifies the class of the device. From this, it is possible to recognize the maximum power of the device.

- **Status** (read-only)
  Shows the current status of the port.
  The following states are possible:
  - disabled
    The PoE power supply is deactivated for this port.
  - delivering
    The PoE power supply is activated for this port and a device is connected.
  - searching
    The PoE power supply is activated for this port but there is no device connected.

  Note

  If a device is connected to a port with PoE capability, a check is made to determine whether the power of the port is adequate for the connected device. If the power of the port is inadequate, although PoE is enabled in "Setting", the port nevertheless has the status "disabled". This means that the port was disabled by the PoE power management.

- **Power [mW]** (read-only)
  Shows the power that the SCALANCE provides at this port.
- **Voltage [V]** (read-only)
  Shows the voltage applied to this port.
- **Current [mA]** (read-only)
  Shows the current with which a device connected to this port is supplied.

Port diagnostics

Cable tester

Note

The page is only available if there is an online connection to the device.

With this page, each individual Ethernet port can run independent fault diagnostics on the cable. This test is performed without needing to remove the cable, connect a cable tester and install a loopback module at the other end. Short-circuits and cable breaks can be localized to within a few meters.

Note

Please note that this test is permitted only when no data connection is established on the port to be tested. If, however, there is a data connection to the port to be tested, this is briefly interrupted. Automatic re-establishment of the connection can fail and then needs to be done manually.

Description

- **Port**
  Select the required port from the drop-down list.
- **Run Test**
  Activates error diagnostics. The result is shown in the table.

This table contains the following columns:

- **Pair**
  Shows the wire pair in the cable.

  Note

  Wire pairs
  - Wire pairs 4-5 and 7-8 of 10/100 Mbps network cables are not used.
  - In 1000 Mbps or gigabit Ethernet, all 4 wire pairs are used. The wire pair assignment - pin assignment is as follows (DIN 50173):
    - Pair 1 = pin 4-5
    - Pair 2 = pin 1-2
    - Pair 3 = pin 3-6
    - Pair 4 = pin 7-8

- **Status**
  Displays the status of the cable.
- **Distance [m]**
  Displays the distance to the cable end, cable break, or short-circuit.

SFP diagnostics

On this page, you run independent error diagnostics for each individual SFP port. This test is performed without needing to remove the cable, connect a cable tester or install a loopback module at the other end.

Note

Please note that this test is permitted only when no data connection is established on the port to be tested. If, however, there is a data connection to the port to be tested, this is briefly interrupted. Automatic re-establishment of the connection can fail and then needs to be done manually.

Description

The page contains the following boxes:

- **Port**
  Select the required port from the drop-down list.
- **Refresh**
  Refreshes the display of the values of the set port. The result is shown in the table.

The values are shown in the following boxes:

- Name
  Shows the name of the interface.
- Model
  Shows the type of interface.
- Revision
  Shows the hardware version of the SFP.
- Serial
  Shows the serial number of the SFP.
- Nominal Bit Rate [Mbps]
  Shows the nominal bit rate of the interface.
- Max. Link (50.0/125um) [m]
  Shows the maximum distance in meters that is possible with this medium.
- Max. Link (62.5/125um) [m]
  Shows the maximum distance in meters that is possible with this medium.

The following table shows the values of the SFP transceiver used in this port:

- Temperature [°C]
  Shows the temperature of the interface.
- Voltage [V]
  Shows the voltage applied to the interface [V].
- Current [mA]
  Shows the current consumption of the interface [mA].
- RX Power [mW]
  Shows the receive power of the interface [mW].
- Tx Power [mW]
  Shows the transmit power of the interface [mW].
- Current
  Shows the current value.
- Low
  Shows the lowest value.
- High
  Shows the highest value.

Configuration Backup

Configuration Package Backup

On this page, you can save backups of your configuration.

The created backups are saved in the file type "ConfigPackBackup". On the "System > Load&Save > HTTP/TFTP > " page you can save configuration backups on your client PC or load them from there.

Description

The page contains the following boxes:

- **Name**
  Enter a name for the backup.

This table contains the following columns:

- **Select**
  Select the row you want to delete.
- **Name**
  Shows the name of the backup.
- **Size [bytes]**
  The first row "Available memory" shows how much memory is available for backups on the device. When you create a backup, the available memory space is reduced accordingly. The other rows show the size of each backup.
- **Restore**
  Click the "Restore" button to load the relevant backup on the device.

Configuring layer 2 functions

Configuration

Configuring layer 2

On this page, you create a basic configuration for the functions of layer 2. On the configuration pages of these functions, you can make more detailed settings. You can also check the settings on the configuration pages.

Description of the displayed boxes

- **Protocol Based VLAN**
  Enable or disable protocol-based VLAN. You can configure other settings in "Layer 2 > VLAN".
- **Subnet Based VLAN**
  Enable or disable subnet-based VLAN. You can configure other settings in "Layer 2 > VLAN".
- **Dynamic MAC Aging**
  Enable or disable the "Aging" mechanism. You can configure other settings in "Layer 2 > Dynamic MAC Aging".

- **Redundancy Type**
  The following settings are available:
  - **"-" (disabled)**
    The redundancy function is disabled.
  - **Spanning Tree**
    If you select this option, you specify the required redundancy mode in "Redundancy Mode".
  - **Ring**
    Enables ring redundancy. You can configure other settings in "Layer 2 > Ring Redundancy > Ring".
  - **Ring with RSTP**
    If you select this option, the compatibility mode for spanning tree is set permanently to RSTP. In the "Redundancy Type" drop-down list, you specify the redundancy mode of the ring redundancy. You can change the current setting in the "Ring Redundancy" and "Spanning Tree" menus.

    Note

    Restriction relating to ports with the "Ring with RSTP" option

    If you have enabled the "Ring with RSTP" option, the following ports must not be included in the spanning tree:
    - Ring ports
    - Standby ports
    - Standby coupling ports

- **Redundancy Mode**
  If you select "Ring" or "Ring with RSTP" for the "Redundancy Type", the following options are then available:
  - Automatic Redundancy Detection
    Select this setting to create an automatic configuration of the redundancy mode.
    In the Automatic Redundancy Detection mode, the device automatically detects whether or not there is a device with the "HRP Manager" role in the ring. If there is, the device adopts the role "HRP Client" client.
    If no HRP manager is found, all devices with the "Automatic Redundancy Detection" or "MRP Auto Manager" setting negotiate among themselves to establish which device adopts the role of "MRP Manager". The device with the lowest MAC address will always become "MRP Manager". The other devices automatically set themselves to "MRP Client" mode.
  - MRP Auto-Manager
    In the "MRP Auto Manager" mode, the devices negotiate among themselves to establish which device will adopt the role of "MRP Manager". The device with the lowest MAC address will always become "MRP Manager". The other devices automatically set themselves to "MRP Client" mode.
    In contrast to the setting "Automatic Redundancy Detection", the devices are not capable of detecting whether or not an HRP manager is in the ring.

    Note

    MRP configuration in STEP 7

    If you set the role "Manager (Auto)" or "Manager" for the device in STEP 7, in both cases, "MRP Auto Manager" is displayed on this WBM page. In the display in the CLI, a distinction is made between the two roles.

  - MRP Client
    The device adopts the role of MRP client.
  - HRP Client
    The device adopts the role of HRP client.
  - HRP Manager
    The device adopts the role of HRP manager.
    When you configure an HRP ring, one device must be set as HRP manager. For all other devices, "HRP Client" or "Automatic Redundancy Detection" must be set.

  If you select "Spanning Tree" for the "Redundancy Type", the following options are then available:
  - **STP**
    Enables the Spanning Tree Protocol (STP). Typical reconfiguration times with spanning tree are between 20 and 30 seconds. You can configure other settings in "Layer 2 > Spanning Tree".

- RSTP
  Enables the Rapid Spanning Tree Protocol (RSTP). If a spanning tree frame is detected at a port, this port reverts from RSTP to spanning tree. You can configure other settings in "Layer 2 > Spanning Tree".

  Note

  When using RSTP (Rapid Spanning Tree Protocol), loops involving duplication of frames or frames being overtaken may occur briefly. If this is not acceptable in your particular application, use the slower standard spanning tree mechanism.

- MSTP
  Enables the Multiple Spanning Tree Protocol (MSTP). You can configure other settings in "Layer 2 > Spanning Tree".

- Standby
  Enable or disable the "Standby redundancy" function. You can configure other settings in "Layer 2 > Ring Redundancy > Standby".
- MRP Interconnection
  Enable or disable the MRP Interconnection function. You can find other settings in "Layer 2 > Ring Redundancy > MRP Interconnection".
- Passive Listening
  Enable or disable the "passive listening" function.
- RMON
  If you select this check box, Remote Monitoring (RMON) allows diagnostics data to be collected on the device, prepared and read out using SNMP by a network management station that also supports RMON. This diagnostic data, for example port-related load trends, allow problems in the network to be detected early and eliminated.
- Dynamic Multicast
  The following settings are possible:
  - "-" (disabled)
  - IGMP Snooping
    Enables IGMP (Internet Group Management Protocol). You can configure other settings in "Layer 2 > Multicast > IGMP".
  - MLD Snooping
    Enables MLD. You can configure other settings in "Layer 2 > Multicast > MLD".
  - IGMP and MLD Snooping
    Enables IGMP and MDL.
  - GMRP
    Enables GMRP (GARP Multicast Registration Protocol). You can configure other settings in "Layer 2 > Multicast > GMRP".

    Note

    GMRP and IGMP cannot operate at the same time.

- GVRP
  Enable or disable "GVRP" (GARP VLAN Registration Protocol). You can configure other settings in "Layer 2 > VLAN > GVRP".

- **Mirroring**
  Enable or disable port mirroring. You can configure other settings in "Layer 2 > Mirroring > Port".
- **Loop Detection**
  Enable or disable loop detection. You can configure other settings in "Layer 2 > Loop Detection".
- **PTP**
  Specify how the device will process PTP messages. You can configure other settings in "Layer 2 > PTP".
  - off
    The device does not process any PTP messages. PTP messages are, however, forwarded according to the rules of the switch.
  - transparent
    The device adopts the function of a transparent clock and forwards PTP messages to other nodes while at the same time making entries in the correction field of the PTP message.

QoS

General

Note

This page only exists with the following modules:

- SCALANCE XB-200
- SCALANCE XC-200
- SCALANCE XF-200BA
- SCALANCE XP-200
- SCALANCE XR-300WG

Transmission priorities

On this page, you can specify the priorities of different frames. In addition to this, depending on the priority you can set the method according to which the processing order of the frames is specified. You should also refer to the section "Basics (Page 2016)".

Description of the displayed values

The page contains the following boxes:

- **Broadcast Priority**
  Specify the priority of broadcast frames. The switch sorts the frame into a queue according to this prioritization . You configure the assignment of the priority to a queue on the page ""Layer 2 > QoS > CoS Map".
- **Agent Priority**
  Specify the priority of agent frames. The switch sorts the frame into a queue according to this prioritization . You configure the assignment of the priority to a queue on the page ""Layer 2 > QoS > CoS Map".
- **Scheduling Mode**
  Select the order in which the frames are processed in the queues.
  The higher the queue number, the higher the processing priority.
  - Strict Queueing
    As long as there are frames with high priority in the queue, only these high-priority frames are processed.
  - Weighted Fair Queueing
    Even if there are frames with high priority in the queue, frames with a lower priority will be processed occasionally.

CoS Map

CoS Queue

On this page, you can assign CoS priorities to different Traffic Queues.

Description of the displayed boxes

- **CoS**
  Shows the CoS priority of the incoming packets.
- **Queue**
  From the drop-down list, select the queue that is assigned to the CoS priority.
  The higher the number of the queue, the higher the processing priority.
  With queues 1 - 6 frames with a lower priority are occasionally processed even if there are frames with high priority in the queue.
  With queues 7 - 8 only frames with a high priority are processed as long as there are frames with high priority in the queue.

The service classes (CoS) are assigned to the queues as follows:

SCALANCE XB-200/XP-200	SCALANCE XM-400/XR-500
CoS 0 → Queue 2	CoS 0 → Queue 2
CoS 1 → Queue 1	CoS 1 → Queue 1
CoS 2 → Queue 1	CoS 2 → Queue 3
CoS 3 → Queue 2	CoS 3 → Queue 4
CoS 4 → Queue 3	CoS 4 → Queue 5
COo 5 → Queue 3	CoS 5 → Queue 6
CoS 6 → Queue 4	CoS 6 → Queue 7
CoS 7 → Queue 4	CoS 7 → Queue 8

DSCP Mapping

DSCP Mapping

On this page, you can assign DSCP priorities to different Queues.

Description of the displayed values

Table 1 has the following columns:

- **DSCP min**
  From the drop-down list, select the minimum value for a range of DSCP codes to which you wish to assign a queue.
- **DSCP max**
  From the drop-down list, select the maximum value for a range of DSCP codes to which you wish to assign a queue.
- **Queue**
  From the drop-down list, select the forwarding queue (send priority) that is assigned to the range of DSCP codes.
- **Copy to Table**
  When you click the button, the selected forwarding queue (send priority) is assigned to the DSCP codes in the specified range.

Table 2 has the following columns:

- **DSCP**
  Shows the DSCP priority of the incoming frames.
- **Queue**
  From the drop-down list, select the queue that is assigned to the DSCP priority.
  The higher the queue number, the higher the processing priority.
  With queues 1 - 6 frames with a lower priority are occasionally processed even if there are frames with high priority in the queue.
  With queues 7 - 8 only frames with a high priority are processed as long as there are frames with high priority in the queue.

SCALANCE XB-200/XP-200	SCALANCE XM-400/XR-500
DSCP codes 0 - 15 → Queue 1	DSCP codes 0 - 7 → Queue 2
DSCP codes 16 - 31 → Queue 2	DSCP codes 8 - 15 → Queue 1
DSCP codes 32 - 47 → Queue 3	DSCP codes 16 - 23 → Queue 3
DSCP codes 48 - 63 → Queue 4	DSCP codes 24 - 31 → Queue 4
	DSCP codes 32 - 39 → Queue 5
	DSCP codes 40 - 47 → Queue 6
	DSCP codes 48 - 55 → Queue 7
	DSCP codes 56 - 63 → Queue 8

QoS Trust

Specifying the subnet priority

On this page you can set the method according to which frames to be forwarded are prioritized port by port.

Description of the displayed values

Table 1 has the following columns:

- Port
  Shows that the setting is valid for all ports of table 2.
- Trust Mode
  Select the setting from the drop-down list. The following settings are possible:
  - No Trust
  - Trust COS
  - Trust DSCP
  - Trust COS-DSCP
  - No change
    Table 2 remains unchanged.
- Transfer to table
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the configurable PoE ports.
  The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Trust Mode**
  Select the required mode from the drop-down list:

  Note

  You configure the prioritization of the receiving port on the page "Layer 2 > VLAN > Port Based VLAN".

  You configure the assignment of the following priorities to a queue on the page ""Layer 2 > QoS > CoS Map".

  - Receiving port
  - VLAN tag
  - Broadcast and agent frame

  You configure the assignment of the DSCP prioritization to a queue on the page ""Layer 2 > QoS > DSCP Map".

  – No Trust
    The switch sorts the incoming frames into a queue according to the prioritization of the receiving port.
    If there is a DSCP value in the IP header, this is ignored. If a VLAN tag exists, it is replaced by the priority value of the receiving port.

  – Trust COS
    If an incoming frame contains a VLAN tag, the switch sorts it into a queue according to this prioritization.
    If the frame does not contain a VLAN tag, the switch sorts the frame into a queue according to the prioritization of the receiving port.
    If there is a DSCP value in the IP header, this is ignored.

  – Trust DSCP
    If an incoming frame contains a DSCP prioritization, the switch sorts it into a queue according to this prioritization.
    If the frame does not contain a DSCP prioritization, the switch sorts the frame into a queue according to the prioritization of the receiving port.
    If the frame contains a VLAN tag, this is ignored.

  – Trust COS-DSCP
    With an incoming frame, there is a sequential check to determine which prioritization it contains.
    If the frame contains a DSCP prioritization, it is handled as in the "Trust DSCP" mode.
    If the frame contains no DSCP prioritization, the switch checks whether the frame contains a VLAN tag. If it contains a VLAN tag, the switch sorts it into a queue according to this prioritization.
    If the frame contains neither a DSCP prioritization nor a VLAN tag, the switch sorts the frame into a queue according to the prioritization of the receiving port.

CoS Port Remap

Changing priority when sending

On this page depending on the priority when receiving, you can change the priority of a frame with which it is sent.

Description of the displayed boxes

The page contains the following boxes:

- **CoS Remap**
  Enable or disable frames being sent with changed priorities according to Table 2.

Table 1 has the following columns:

- **Port**
  Shows that the settings are valid for all ports of table 2.
- **Priority 0 - 7**
  The priority in the column stands for the priority with which a frame is received.
  - 0 - 7
    Select the priority with which a frame will be sent.
  - No Change
    No change in table 2.
- **Copy to Table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows all available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Priority 0 - 7**
  The priority in the column stands for the priority with which a frame is received.
  In the drop-down list select the priority with which a frame will be sent.

Rate Control

Limiting the transfer rate of incoming and outgoing data

On this page, you configure the load limitation (maximum number of data packets per second) for the individual ports. You can specify the category of frame for which these limit values will apply.

Description of the displayed values

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Limit Ingress Unicast (DLF) / Limit Ingress Broadcast / Limit Ingress Multicast / Limit Ingress Unicast**
  Select the required setting.
  - Enabled: Enables the function.
  - Disabled: Disables the function
  - No Change: The setting in table 2 remains unchanged
- **Total Ingress Rate [pkts/s]**
  Specify the maximum number of incoming packets. If "No Change" is entered, the entry in table 2 remains unchanged
- **Egress Rate [Kb/s]**
  Specify the data rate for all outgoing frames. If "No Change" is entered, the entry in table 2 remains unchanged
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows all available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Limit Ingress Unicast (DLF)**
  Enable or disable the data rate for limiting incoming unicast frames with an unresolvable address (Destination Lookup Failure).
- **Limit Ingress Broadcast**
  Enable or disable the data rate for limiting incoming broadcast frames.
- **Limit Ingress Multicast**
  Enable or disable the data rate for limiting incoming multicast frames.
- **Limit Ingress Unicast**
  Enable or disable the data rate for limiting the incoming unicast frames.
- **Total Ingress Rate [pkts/s]**
  Specify the maximum number of incoming packets.
- **Egress Rate [Kb/s]**
  Specify the data rate for all outgoing frames.

Note

Rounding of the values, deviation from desired value

When you enter values, note that the WBM rounds to correct values.

If values are configured for Total Ingress Rate and Egress Rate, the actual values in operation can deviate slightly from the set values.

VLAN

General

VLAN configuration page

On this page, you specify whether the device forwards frames with VLAN tags transparently (IEEE 802.1D/VLAN-unaware mode) or takes VLAN information into account (IEEE 802.1Q/VLAN-aware mode). If the device is in the "802.1Q VLAN Bridge" mode, you can define VLANs and specify the use of the ports.

The possible settings on this page depend on what you select in the "Base Bridge Mode" box.

Note

Changing the "Agent VLAN ID"

If the configuration PC is connected directly to the device via Ethernet and you change the "agent VLAN ID", the device is no longer reachable via Ethernet following the change.

Important rules for VLANs

Make sure you keep to the following rules when configuring and operating your VLANs:

- Frames with the VLAN ID "0" are handled as untagged frames but retain their priority value.
- As default, all ports on the device send frames without a VLAN tag to ensure that the end node can receive these frames.
- With SCALANCE X devices, the VLAN ID "1" is the default on all ports.
- If an end node is connected to a port, outgoing frames should be sent without a tag (static access port). If, however, there is a further switch at this port, the frame should have a tag added (trunk port).
- With a trunk port, the VLAN assignment is dynamic. Static configurations can only be created if, in addition to the trunk port property, the port is also entered statically as a member in the VLANs involved. An example of a static configuration is the assignment of the multicast groups in certain VLANs.

Note

With SCALANCE XB-200, XC-200, XF-200BA, XP-200 and XR-300WG, the rules for Base Bridge Mode "802.1Q VLAN Bridge" apply.

Description of the displayed boxes

The page contains the following boxes:

- **Bridge mode**
  - Customer
    If you operate the device with the "Customer" role, it behaves like a standard IE switch.
  - Provider
    If you operate the device with the "Provider" role, in addition to the properties of the "Customer" role, it provides the option of managing external VLAN tags. With this role you can use the function Q-in-Q VLAN tunnel.
- **Base Bridge Mode (not with SCALANCE XM-400/XR-500)**
  Specify the "Base Bridge Mode".

  Note
  Changing Base Bridge Mode

  Note the section "Changing Base Bridge Mode". This section describes how a change affects the existing configuration.

  - 802.1Q VLAN Bridge
    Sets the mode "VLAN-aware" for the device. In this mode, VLAN information is taken into account.
  - 802.1D Transparent Bridge
    Sets the mode "VLAN-unaware" for the device. In this mode, VLAN tags are not changed but are forwarded transparently. The VLAN priority is evaluated for CoS. In this mode, you cannot create any VLANs. Only a management VLAN is available: VLAN 1.
- **VLAN ID**
  Enter the VLAN ID in the "VLAN ID" input box. Range of values: 1 ... 4094

The table has the following columns:

- **Select**
  Select the row you want to delete.
- **VLAN ID**
  Shows the VLAN ID. The VLAN ID (a number between 1 and 4094) can only be assigned once when creating a new data record and can then no longer be changed. To make a change, the entire data record must be deleted and created again. Up to 257 VLANs can be defined.
- **Name**
  Enter a name for the VLAN. The name only provides information and has no effect on the configuration. The VLAN name can be a maximum of 32 characters long.
- **Status**
  Shows the status type of the entry in the internal port filter table. Here, static means that the address was entered as a static address by the user. The entry GVRP means that the configuration was registered by a GVRP frame. This is, however, only possible if GVRP was enabled for the device.
- **Private VLAN Type**
  Shows the type of the PVLAN.

- Primary VLAN ID
  , With secondary PVLANs shows the ID of the corresponding primary PVLAN.

- Transparent(only with SCALANCE XM-400/XR-500)
  If you enable these settings, you switch a VLAN to the transparent mode. Ports that were assigned to this VLAN as members or untagged members now become transparent ports. This means the following:
  - The port VLAN ID of the transparent port is set to the ID of this VLAN.
  - Untagged frames that are received at these ports are forwarded to all other transparent ports once again without tag as long as they are not forwarded to a standard VLAN by a protocol or subnet rule.
  - Frames tagged with VLAN ID "0" and that are received at these ports are forwarded to all other transparent ports once again tagged with VLAN ID "0" as long as they are not forwarded to a standard VLAN by a protocol or subnet rule.
  - Frames tagged with the VLAN ID of the transparent VLAN and that are received at these ports are forwarded to all transparent ports once again tagged with the VLAN ID of the transparent VLAN.
  - Other frames are forwarded according to the normal VLAN rules and a transparent port behaves like an untagged member in this VLAN.
  - All ports that were not members or untagged members in the relevant VLAN are automatically set to the "Forbidden" status.
  - As long as a VLAN is configured as a transparent VLAN, the ports belonging to this VLAN cannot be modified.
  - You can only configure one a transparent VLAN.
- List of ports
  Specify the use of the port. The following options are available:
  - "-"
    The port is not a member of the specified VLAN. With a new definition, all ports have the identifier "-".
  - M
    The port is a member of the VLAN. Frames sent in this VLAN are forwarded with the corresponding VLAN tag.
  - R
    The port is a member of the VLAN. A GVRP frame is used for the registration.
  - U (upper case)
    The port is an untagged member of the VLAN. Frames sent in this VLAN are forwarded without the VLAN tag. Frames without a VLAN tag are sent from this port.
  - u (lower case)
    The port is an untagged member of the VLAN, but the VLAN is not configured as a port VLAN. Frames sent in this VLAN are forwarded without the VLAN tag.
  - F
    The port is not a member of the specified VLAN and it is not possible for the VLAN to be registered dynamically at this port using GVRP. If a port in a VLAN has this option, it cannot become a member of this VLAN even if it is configured as a trunk port. You can configure other settings in "Layer 2 > VLAN > Port-based VLAN".

– T
This option is only displayed and cannot be selected in the WBM. This port is a trunk port making it a member in all VLANs. You configure this function in the CLI (Command Line Interface) using the "`switchport mode trunk`" command.

Changing Base Bridge Mode with SCALANCE XB-200

VLAN-unaware (802.1D transparent bridge) → VLAN-aware (802.1Q VLAN bridge)

If you change the Base Bridge Mode from VLAN-unaware to VLAN aware, this has the following effects:

- All static and dynamic unicast entries are deleted.
- All static and dynamic multicast entries are deleted.
- With spanning tree you can set the following protocol compatibility: STP, RSTP and MSTP.

VLAN-aware (802.1Q VLAN bridge) → VLAN-unaware (802.1D transparent bridge)

If you change the Base Bridge Mode from VLAN-aware to VLAN-unaware, this has the following effects:

- All static and dynamic VLAN information is deleted.
- A management VLAN is created: VLAN 1:
- All static and dynamic unicast entries are deleted.
- All static and dynamic multicast entries are deleted.
- With spanning tree you can set the following protocol compatibility: STP and RSTP.
- You cannot use GVRP.
- You cannot use guest VLAN.
- The VLAN assignment cannot be adopted from the RADIUS server.
- You cannot configure the port type.
- Defined access rules must be valid for all VLANs. On the "Security > Management ACL" page, the value "1-4094" must be defined for the parameter "VLANs Allowed".

GVRP

Configuration of GVRP functionality

Using GVRP frame, a different device can register at the port of the device for a specific VID. A different device, can, for example be an end device or a switch. The device can also send GVRP frames via this port. On this page, you can enable each port for GVRP functionality.

Description of the displayed boxes

The page contains the following boxes:

- GVRP
  Enable or disable the "GVRP" function.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **Setting**
  Select the setting. You have the following setting options:
  - Enabled
    Enables the sending of GVRP frames.
  - Disabled
    Disables the sending of GVRP frames.
  - No Change
    No change in table 2.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows all available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Setting**
  Enable or disable the sending GVRP frames.

Port-based VLAN

Processing received frames

On this page, you specify the configuration of the port properties for receiving frames.

Description of the displayed boxes

Table 1 has the following columns:

- **Port**
  Shows that the settings are valid for all ports.
- **Priority / Port VID / Acceptable Frames / Ingress Filtering**
  Select the setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports and link aggregations. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Priority**
  Select the priority assigned to untagged frames.
  The CoS priority (Class of Service) used in the VLAN tag. If a frame is received without a tag, it will be assigned this priority. This priority specifies how the frame is further processed compared with other frames. There are a total of eight priorities with values 0 to 7, where 7 represents the highest priority (IEEE 802.1p Port Priority).
- **Port VID**
  Select the VLAN ID. Only VLAN IDs defined on the "VLAN > General" page can be selected. If a received frame does not have a VLAN tag, it has a tag with the VLAN ID specified here added to it and is sent according to the rules at the port.
- **Acceptable Frames**
  Specify which types of frames will be accepted. The following alternatives are possible:
  - Tagged Frames Only
    The device discards all untagged frames. Frames tagged with "0" are treated like untagged frames. The device forwards all tagged frames. Otherwise, the forwarding rules apply according to the configuration.
  - Untagged and Priority Tagged Only
    The device discards all tagged frames. The device forwards all untagged frames and frames with a priority (Priority Tagged Frames). Otherwise, the forwarding rules apply according to the configuration. If you have configured the Bridge mode "Provider" this means that the device treats all incoming frames as untagged frames.
  - All
    The device forwards all frames
- **Ingress Filtering**
  Specify whether the VID of received frames is evaluated. The following options are available:
  - Enabled
    The VLAN ID of received frames decides whether they are forwarded: To forward a VLAN tagged frame, the receiving port must be a member in the same VLAN. Frames from unknown VLANs are discarded at the receiving port.
  - Disabled
    All frames are forwarded.

Protocol-based VLAN group

Introduction

On this page, you specify groups and assign a protocol to them.

Description of the displayed boxes

The page contains the following boxes:

- **Protocol Based VLAN**
  Enable or disable the protocol-based VLAN assignment.
- **Protocol Value**
  Enter the hexadecimal protocol value.
  A few examples are shown below:
  - PROFINET: 88:92
  - IP: 08:00
  - Novell: 81:37
  - netbios: f0:f0
  - appletalk: 80:9b
- **Group ID**
  Enter the ID of the group.

The table has the following columns:

- **Protocol Value**
  Shows the file value.
- **Group ID**
  Shows the group ID.

Protocol-based VLAN port

Introduction

On this page, you specify which protocol and which VLAN is assigned to the individual port.

Description of the displayed boxes

- **Port**
  Select the required port. All available ports and the link aggregations can be selected.
- **Group ID**
  Select the group ID from the drop-down list. Specify the ID on the WBM page "Layer 2 > VLAN > Protocol Based VLAN Group".

The table has the following columns:

- **Port**
  All available ports and the link aggregations are shown.
- **Group ID**
  Shows the group ID assigned to the port.
- **VLAN ID**
  Select the required VLAN ID to be assigned to the port.

IPv4 subnet-based VLAN

Introduction

On this page, you specify which VLAN ID is assigned to the subnet.

Description of the displayed boxes

The page contains the following boxes:

- Subnet Based VLAN
  Enable or disable the IPv4 subnet-based VLAN assignment.
- Port
  Select the port. All available ports and the link aggregations can be selected.
- Subnet Address
  Enter the IP address of the subnet.
  Example: 192.168.10.0 for the network 192.168.10.x with nodes 192.168.10.1 to 192.168.10.254.
- Subnet Mask
  Enter the subnet mask.

The table has the following columns:

- Port
  All available ports and the link aggregations are shown.
- Subnet Address
  Shows the subnet assigned to the port.
- Subnet Mask
  Shows the subnet mask assigned to the port.
- VLAN ID
  Select the VLAN ID you want to assign to the port or the subnet.

IPv6 prefix-based VLAN

Introduction

On this page, you specify which VLAN ID is assigned to the IPv6 prefix.

Description of the displayed boxes

The page contains the following boxes:

- Port
  Select the port in the drop-down list. All available ports and the link aggregations can be selected.
- Prefix Address
  Enter the IPv6 address.

- Prefix Length
  Enter the number of left-hand bits belonging to the prefix
- VLAN ID
  Select the VLAN ID you want to assign to the port or the prefix.

The table has the following columns:

- Select
  Select the row you want to delete.
- Port
  All available ports and the link aggregations are displayed.
- Prefix Address
  Shows the IPv6 address of the port.
- Prefix Length
  Shows the prefix length.
- VLAN ID
  Select the VLAN ID you want to assign to the port or the prefix.

Private VLAN

General

Private VLAN configuration page

On this page you define the types of the PVLANs and assign secondary PVLANs to a primary PVLAN.

Note

All secondary PVLANs must be known on all devices of a PVLAN. Even if a device has no host port in a secondary PVLAN, the secondary PVLAN must be known on the device.

Description of the displayed boxes

The table has the following columns:

- VLAN ID
  Shows the VLAN ID.
- Private VLAN Type
  Specify the type of PVLAN:
  - Primary
    Mit diesem Typ definieren Sie ein Primary PVLAN. In a PVLAN you can only define one primary PVLAN. The primary PVLAN uses the VLAN ID of the VLAN.
  - Isolated
    With this type, you define a secondary PVLAN. Only one device can exist in this secondary PVLAN. The secondary PVLAN has a specific VLAN ID.
  - Community
    With this type, you define a secondary PVLAN. The devices in this secondary PVLAN can communicate with each other via layer 2. The secondary PVLAN has a specific VLAN ID.
- Primary VLAN ID
  With secondary PVLANs select the VLAN ID of the primary PVLAN.

IP Interface Mapping

Private VLAN configuration page

On this page you specify from which secondary PVLANs the IP interface of the primary PVLAN will be reachable. Configure the IP interface assignment for all functions for which an end device needs to communicate from the secondary PVLAN via the IP interface of the primary PVLAN.

Examples:

- An end device in the secondary PVLAN is configured as DHCP client. A remote DHCP server is set up. A PVLAN switch is configured as DHCP relay agent. Configure an IP interface in the primary PVLAN of the DHCP relay agent. Assign the secondary PVLANs containing DHCP clients to this IP interface.
- A PVLAN switch is configured as router. Configure an IP interface in the primary PVLAN of the router. Assign the secondary PVLANs containing end devices that use the router as a gateway to this IP interface.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  Select a primary PVLAN with an IP interface.
- Secondary VLAN ID
  Select a secondary VLAN ID from which the IP interface of the primary PVLAN will be reachable.

The table has the following columns:

- Select
  Select the row you want to delete.
- Interface
  Shows the IP interface.
- Secondary VLAN-ID
  Shows the secondary VLAN-ID of the secondary PVLAN from which the IP interface of the primary PVLAN is reachable.

Provider bridge

Tunnel ports

Configuration page for tunnel ports

On this page, you enable the Q-in-Q VLAN tunnel function. Frames received by a tunnel port are expanded by an external VLAN tag, the PVID of the port.

Description of the displayed boxes

Table 1 has the following columns:

- 1st column
  Shows that the settings are valid for all ports of table 2.
- Setting
  Select the setting from the drop-down list. You have the following setting options:
  - Enabled
    Enables the Q-in-Q VLAN tunnel function on all ports.
  - Disabled
    Disables the Q-in-Q VLAN tunnel function on all ports.
  - No Change
    Table 2 remains unchanged.
- Copy to Table
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- Port
  Shows all available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- Setting
  Enable or disable the function for this port.

Mirroring

Basics

Mirroring

The device provides the option of simultaneously channeling incoming or outgoing data streams via other interfaces for analysis or monitoring. This has no effect on the monitored data streams. This procedure is known as mirroring. In this menu section, you enable or disable mirroring and set the parameters.

Mirroring ports

Mirroring a port means that the data traffic at a port (mirrored port) of the IE switch is copied to another port (monitor port). You can mirror one or more ports to a monitor port.

If a protocol analyzer is connected to the monitor port, the data traffic at the mirrored port can be recorded without interrupting the connection. This means that the data traffic can be investigated without being affected. This is possible only if a free port is available on the device as the monitor port.

General

Note

It cannot be guaranteed when mirroring the data traffic that all packets are mirrored. This depends primarily on the load on the mirrored ports and on the number of sessions. To achieve maximum precision, a limit of one session is recommended.

Mirroring in general

On this page, you can enable or disable the mirroring function and make the basic settings.

Note

Mirroring a port does not work beyond switch core boundaries. Refer to the operating instructions of the device.

Note

You need to disable port mirroring if you want to connect a normal end device to the monitor port. This does not apply to the function extender BUS ANALYZER Agent XM-400.

Note the data rate

If the maximum data rate of the mirrored port is higher than that of the monitor port, data may be lost and the monitor port no longer reflects the data traffic at the mirrored port. Several ports can be mirrored to one monitor port at the same time.

RSPAN

With RSPAN (Remote Switched Port Analyzer) you can forward the data traffic of a mirroring session to the monitor port via a VLAN. On the RSPAN VLAN, the mirrored data traffic is not disturbed by other data.

Frames addressed directly to the mirroring source cannot be mirrored on the RSPAN destination port.

Settings

- Mirroring
  Enable or disable mirroring of the data traffic.
- Monitor Barrier
  Enable or disable the setting to restrict communication via the monitor port.

  Note

  Effects of monitor barrier

  If you enable this setting, the management of the switch via the monitor port is no longer reachable. The following port-specific functions are changed:
  - DCP forwarding is turned off
  - LLDP is turned off
  - Unicast, multicast and broadcast blocking is turned on

  The previous statuses of these functions are no longer restored after disabling monitor barrier again. They are reset to the default values and may need to be reconfigured.

  You can reconfigure these functions manually even if monitor barrier is turned on. The data traffic on the monitor port is also allowed again. If you do not require this, make sure that only the data traffic you want to monitor is forwarded to the interface.

  If mirroring is disabled, the listed port-specific functions are reset to the default values. This reset takes place regardless of whether the functions were configured manually or automatically by enabling "Monitor Barrier".

  Note

  Function extender BUS ANALYZER Agent XM-400

  If the destination port is a port of the function extender BUS ANALYZER Agent XM-400 the monitor barrier option is always activated. It is activated regardless of whether the check box was disabled or enabled.

- RSPAN VLAN ID
  Select a VLAN on which the data traffic of a mirroring session is transferred without disruption.

The table for the basic settings contains the following boxes:

- **Session ID**
  The Session ID is assigned automatically when a new entry is created.
- **Session Type**
  Select the required entry:
  - '-'
    None
  - Port-based
    Port-based mirroring
  - VLAN
    VLAN-based mirroring.
  - MAC ACL
    Mirroring of the MAC Access Control List.
  - IP ACL
    Mirroring of the IP Access Control List.

  Note

  If you have created a session of the type "VLAN" "MAC ACL" or "IP ACL", you cannot create any further sessions. You can create up to 7 sessions of the type "Port-based".

  Note

  If you change the "Session Type" of an existing session, all previous configurations of this session are lost.

- **Status**
  Shows whether or not mirroring is active.
- **Hardware Index**
  If in a VLAN you select more than one source port for the port-based egress mirroring, unknown unicast and multicast frames as well as broadcast frames are forwarded only once to the destination port. With several sessions, the corresponding frames are only visible in one session. They are only mirrored on the one destination port with the lowest hardware index.
- **Dest. Port**
  Select the destination port to be mirrored to in this session.

  Note

  Function extender BUS ANALYZER Agent XM-400

  If you connect a function extender BUS ANALYZER Agent XM-400 with a SCALANCE XM-400 basic device, you can select up to four ports of the function extender BUS ANALYZER Agent XM-400 as destination ports. Function extender BUS ANALYZER Agent XM-400 supports port-based, VLAN-based, MAC ACL-based and IP ACL-based mirroring.

- **RSPAN**
  Enable or disable RSPAN for a session.

Port

Mirroring ports

You can only configure the settings on this page if you have already generated a session ID with the session type "Port-based" on the "General" tab.

Description of the displayed boxes

- Session ID
  Select the session you want to monitor. Up to 7 parallel sessions are possible and their ports must not overlap.

The table has the following columns:

- Port
  Shows the port to be monitored.
- Ingress Mirroring
  Enable or disable listening in on incoming packets at the required port.
- Egress Mirroring
  Enable or disable listening in on outgoing packets at the required port.

VLAN

VLAN sources of the port mirroring

You can only configure the settings on this page if you have already generated a session ID with the session type "VLAN" on the "General" tab.

On this page, you specify the VLAN whose incoming data traffic will be mirrored to the monitor port.

It can happen that data packets are visible on the monitor port that were not received in the defined VLAN. These data packets come from functions that are enabled on the device, e.g. SIMATIC time client. To avoid these data packets when VLAN mirroring, disable the relevant functions on the device before a recording.

Description of the displayed boxes

The page contains the following boxes:

- Session ID
  Select the session you want to monitor. Only one session is possible.
- VLAN ID
  Enter the VLAN ID in the "VLAN ID" input box. Range of values: 1 ... 4094

The table "Ingress Mirroring " has the following columns:

- VLAN-ID
  Shows the VLAN ID for which the incoming frames are mirrored. The VLAN ID can only be assigned once when you create a new data record and can then no longer be changed. To make a change, the entire data record must be deleted and created again.

MAC Flow

ACL filter for port mirroring

You can only configure the settings on this page if you have already generated a session ID with the session type "MAC ACL" on the "General" tab.

The MAC ACL filter decides which data is available at the monitor port.

Description of the displayed boxes

- Session ID
  Select the session you want to monitor. Only one session is possible.

The table has the following columns:

- ACL Filter Number
  Shows the number of the ACL filter. You configure the MAC ACL filter in "Security > MAC ACL".
- Ingress Mirroring
  Shows whether incoming packets are mirrored.

  Note

  Rules

  The selected rule only becomes active when you specify with which ACL rules the incoming packets will be filtered for at least one interface. You configure the settings in "Security > MAC ACL > Ingress Rules".

- Source MAC
  Shows the MAC address of the sender.
- Dest. MAC
  Shows the MAC address of the recipient.
- Ingress Interfaces
  Shows all interfaces to which this rule applies. The ACL filter decides which incoming data streams are mirrored on the monitor port (destination port).
- Egress Interfaces
  Shows all interfaces to which this rule applies.

IP Flow

ACL filter for port mirroring

You can only configure the settings on this page if you have already generated a session ID with the session type "IP ACL" on the "General" tab.

The ACL filter decides which data is available at the monitor port.

Description of the displayed boxes

- Session ID
  Select the session you want to monitor. Only one session is possible.

The table has the following columns:

- ACL Filter Number
  Shows the number of the ACL filter.
- Ingress Mirroring
  Shows whether incoming packets are mirrored.

Note

Rules

The selected rule only becomes active when you specify with which ACL rules the incoming packets will be filtered for at least one interface. You configure the settings in "Security > IP ACL > Ingress Rules".

- Source IP
  Shows the IPv4 address of the sender.
- Source Subnet Mask
  Shows the subnet mask of the sender.
- Dest. IP
  Shows the IPv4 address of the recipient.
- Dest. Subnet Mask
  Shows the subnet mask of the recipient.
- Ingress Interfaces
  Shows all interfaces to which this rule applies. The ACL filter decides which incoming data streams are mirrored on the monitor port (destination port).
- Egress Interfaces
  Shows all interfaces to which this rule applies. The ACL filter decides which outgoing data streams are mirrored on the monitor port (destination port).

Dynamic MAC Aging

Protocol settings and switch functionality

The device automatically learns the source addresses of the connected nodes. This information is used to forward data frames to the nodes specifically involved. The network load for the other nodes is reduced.
If a device does not receive a frame whose source address matches a learnt address within a certain time, it deletes the learnt address. This mechanism is known as aging. Aging prevents frames being forwarded incorrectly, for example when an end device (for example a programming device) is connected to a different port. If the check box is not enabled, a device does not delete learned addresses automatically.

Description of the displayed boxes

The page contains the following boxes:

- Dynamic MAC Aging
  Enable or disable the function for automatic aging of learned MAC addresses:
- Aging Time [s]
  Enter the time in seconds. After this time, a learned address is deleted if the device does not receive any further frames from this sender address. Range of values: 10 - 630 seconds

Ring Redundancy

Ring

Factory settings

The factory setting defines MSTP as the redundancy method and the following ring ports:

Devices	Factory defaults ring ports
SCALANCE XB208 and XB216	P0.1 and P0.2
SCALANCE XB205-3	P0.7 and P0.8
SCALANCE XB213-3	P0.15 and P0.16
SCALANCE XC206-2SFP, XC208, XC216 and XC224	P0.1 and P0.2
SCALANCE XC206-2	P0.7 and P0.8
SCALANCE XF-200BA	P1.1 and P2.1
SCALANCE XP208	P0.1 and P0.2
SCALANCE XP216	P0.10 and P0.12
SCALANCE XR324WG	P0.1 and P0.2
SCALANCE XR328-4C WG	P0.25 and P0.26
SCALANCE XM-400	P1.1 and P1.2
SCALANCE XR-500	P0.1 and P0.2

Enabling ring redundancy

You can enable ring redundancy as follows:

- using the WBM
- using the CLI
- using the SELECT/SET button
- using a PNIO configuration download

Settings

- Ring Redundancy
  If you enable the "Ring Redundancy" check box, you turn ring redundancy on. The ring ports set on this page are used.
- Ring redundancy mode
  Specify the ring redundancy mode. The following modes are available:
  - Automatic Redundancy Detection
    Select this setting to create an automatic configuration of the redundancy mode.
    In the "Automatic Redundancy Detection" mode, the device automatically detects whether or not there is a device with the "HRP Manager" role in the ring. If this is the case, the device adopts the role of "HRP Client". If no HRP manager is found, all devices with the "Automatic Redundancy Detection" or "MRP Auto Manager" setting negotiate among themselves to establish which device adopts the role of "MRP Manager". The device with the lowest MAC address will always become "MRP Manager". The other devices automatically set themselves to "MRP Client" mode.
  - MRP Auto Manager
    In the "MRP Auto Manager" mode, the devices negotiate among themselves to establish which device will adopt the role of "MRP Manager". The device with the lowest MAC address will always become "MRP Manager". The other devices automatically set themselves to "MRP Client" mode. In contrast to the setting "Automatic Redundancy Detection", the devices are not capable of detecting whether or not an HRP manager is in the ring.

    Note
    MRP configuration in STEP 7

    If you set the role "Manager (Auto)" or "Manager" for the device in STEP 7, in both cases, "MRP Auto Manager" is displayed on this WBM page. In the display in the CLI, a distinction is made between the two roles.

  - MRP Client
    The device adopts the role of MRP client.
  - HRP Client
    The device adopts the role of HRP client.
  - HRP Manager
    The device adopts the role of HRP manager. When you configure an HRP ring, one device must be set as HRP manager. For all other devices, "HRP Client" or "Automatic Redundancy Detection" must be set.

- **Ring ports**
  Here, you set the ports to be used as ring ports in ring redundancy. The ring port you select in the left-hand drop-down menu is the "Isolated Port" in HRP. With MRP when the link is established, it is decided which port will become the "Isolated Port".
- **Observer**
  Enable or disable the observer. The "Observer" function is only available in HRP rings. The ring port selected in the left-hand drop-down menu is connected to the "Isolated Port" of an HRP manager.
  The observer monitors malfunctions of the redundancy manager or incorrect configurations of an HRP ring.
  If the observer is enabled, it can interrupt the connected ring if errors are detected. To do this, the observer switches a ring port to the "blocking" status. When the error is resolved, the observer enables the port again.
- **Restart Observer** (only available online)
  If numerous errors occur in quick succession, the observer no longer enables its port automatically. The ring port remains permanently in the "blocking" status. This is signaled by the error LED and a message text.
  After the errors have been eliminated, you can enable the port again using the "Restart Observer" button.

Restoring factory settings

If you have restored the factory defaults, ring redundancy is disabled and the default ports are used as the ring ports. This can lead to circulating frames and failure of the data traffic if other settings were used in a previous configuration.

Changing over the status of the ring ports with the redundancy manager

If you configure a redundancy manager, set the status of the ring ports. The first ring port changes to the “blocking" status and the second ring port to the “forwarding" status. As long as ring redundancy is enabled, you cannot change the status of these ring ports.

Note

Make sure that you first open the ring so that there are no circulating frames.

Changing ring ports

You can change the ring ports without needing to open the ring. To change the ring ports, follow the steps below:

- Change to the page "Layer 2 > Spanning Tree > CIST Port".
- Disable the ports in the spanning tree you want to configure as ring ports.
- Change to the page "Layer 2 > Ring Redundancy > Ring".
- Select the new ring ports.
- Change the cable connections.

- Change to the page "Layer 2 > Spanning Tree > CIST Port".
- Enable the ports in the spanning tree that are no longer ring ports.

Standby

Redundant linking of rings

Standby redundancy allows the redundant linking of HRP rings. To establish a standby connection, configure two neighboring devices within a ring as standby master or standby slave. The standby master and the standby slave must be connected via parallel cables to two devices in another ring. In problem-free operation, messages are exchanged between the two rings via the master. If the master's line is disturbed, the slave takes over the forwarding of messages between the two rings. Enable standby redundancy for both standby partners and select the ports via which the device is connected to the rings you want to link to. For the "Standby Connection Name", a name unique within the ring must be assigned for both partners. This identifies the two devices as standby partners that belong together.

Note

To be able to use the function, HRP must be activated.

The standby manager always requires an activated HRP client or HRP manager.

Note

Standby master coupling with an optical 100 Mbps connection

While the coupling partner of a standby master with an optical 100 Mbps connection starts up again, physically disconnect the connection between the standby master and its coupling partner.

Description of the displayed boxes

- **Standby**
  Enable or disable this function.

  Note

  If two devices are linked by the standby function, the "Standby" function must be enabled on both devices.

- **Standby Port**
  Select the port to be standby port. The link to the other ring is via the standby port.
  Note

  Note

  Standby ports in the spanning tree

  Before you enable the port as the standby port, you need to disable the port in the spanning tree.

  1. Change to the page "Layer 2 > Spanning Tree > CIST Port".

  2. Disable the ports in the spanning tree you want to configure as standby ports.

  The standby port is involved in the redirection of data traffic. In there are no problems, only the standby port of the master is enabled and handles the data traffic into the connected HRP ring or HRP bus.
  If the master or the Ethernet connection (link) of one of the standby ports of the master fails, the standby port of the master will be disabled and the standby port of the slave enabled. As a result, a functioning Ethernet connection to the connected network segment (HRP ring or HRP linear bus) is restored.

- **Standby Connection Name**
  This name defines the master/slave device pair. Both devices must be located in the same ring.
  Here, enter the name for the standby connection. This must be identical to the name entered on the standby partner. You can select any name to suit your purposes, however, you can only use the name for one pair of devices in the entire network.

- **Force device to Standby Master**
  When selected, the device is configured as standby master regardless of its MAC address.
  - If this check box is not selected for either of the devices for which the standby master is enabled, then assuming that no error has occurred, the device with the higher MAC address adopts the role of standby master.
  - If the option is selected for both devices or if the "Force device to Standby Master" property is supported by only one device, the standby master is also selected based on the MAC address.

  This type of assignment is important in particular when a device is replaced. Depending on the MAC addresses, the previous device with the slave function can take over the role of the standby master.

  Note

  If the option "Force device to Standby Master" is enabled on both devices of a standby coupling, this can lead to circulating frames and therefore to failure of the data traffic. Enable the "Force device to Standby Master" option only on one device of a standby coupling.

- **Wait for Standby Partner**
  - Enabled
    A standby connection is enabled only after the standby master and the standby slave as well as their standby partners have established a connection. This ensures that the redundant connection is really available before communication via a standby connection is enabled.
  - Disabled
    A standby connection is enabled even if the standby master has not yet established a connection to the standby slave.
    This can lead to circulating frames and failure of the data traffic if another standby connection has already been enabled. Multiple standby connections can, for example, result due to configuration errors if different standby connection names were assigned to the standby master and standby slave.

Link Check

Requirements

- You cannot enable Link Check on ports with 10 Gbps.
- You can only enable the Link Check function with optical ring ports of an HRP or MRP ring.

  Note

  Replacing: media modules: Optical → electrical

  If you run Link Check on an optical port of a media module, not the following:
  - Link Check is activated on the optical port of a media module.
  - You want to replace the media module with a module without optical ports.

  1. Disable the link check on the ports of the inserted module.
  2. Replace the media module.

- Link Check must be enabled on two neighboring devices (connection partners) within an HRP or MRP ring.
- The ring ports on which you enable Link Check must be connected.

Monitoring optical connections in the ring

With the Link Check function, you can monitor the transmission quality of optical sections within an HRP or MRP ring, identify disturbed connections and under certain conditions turn them off. When the disturbed section is turned off, the redundancy manager can close the ring and restore communication.

NOTICE

Make sure that the frames used by Link Check for monitoring the optical connections are not supplanted by an overload of high priority frames in the network.

An overload of high priority frames can, for example, be caused by the following:

- Network loops that can cause duplication of the high priority frames
- Changing the priorities for forwarding frames

Note

Enable Link Check on only one of two connection partners This can lead to incorrect behavior.

Note

If Link Check is enabled on all devices of a ring at the same time, and several connections within the ring have problems, this leads to fragmentation of the ring.

1. During commissioning enable the Link Check function for one connection section after the other by enabling Link Check for the two connection partners connected to a line.
2. To ensure an error-free connection, wait 1 min. before you enable Link Check for the next connection.

Description of the displayed boxes

This table contains the following columns:

- **Port**
  Shows the available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Setting**
  With this check box, you enable or disable the Link Check function for a port.
  When connection monitoring is enabled, you can see the number of sent and received Link Check test frames on the "Information > Redundancy > Link Check” page.
- **Reset**
  After resetting Link Check, the function is restarted on the port and the statistics are reset. If you use the “Reset” button, the reset must be performed on both connection partners within 30 s.

Note

When you use the “Reset” button, loops can form temporarily resulting in a loss of data traffic. The loop is automatically cleared again.

If this is not acceptable for your application, reset Link Check by pulling the cable and plugging it in again.

MRP Interconnection

Redundant linking of rings

On this page, you create, delete and configure MRP Interconnections.

Description of the displayed boxes

- **MRP Interconnection**
  Select this check box to activate MRP Interconnection for the device. You can only enable MRP Interconnection when the following conditions are met:
  - Ring redundancy is enabled.
  - "MRP Manager" or "MRP Client" is used as ring redundancy method.
  - There is an activated MRP Interconnection.

The table has the following columns:

- **Select**
  Select the row you want to delete.
- **Interconnection-ID**
  Specify the ID of the MRP Interconnection. When specifying the ID, observe the following rules:
  - The Interconnection ID cannot be 0.
  - You need to configure the same Interconnection ID for all four devices used for linking the rings.

- Interconnection Name
  Enter any name for the MRP Interconnection. You can also define different names for the four devices used for linking the rings. The letters 'A' to 'Z' and 'a' to 'z', the numbers '0' to '9' and the '-' symbol are valid characters for this name. A hyphen cannot be used for the first or last character of the name. The interconnection name must contain at least one character and no more than 240 characters.
- Interconnection Port
  From this drop-down list, select the port that is used for the MRP Interconnection. Be aware of the following restrictions:
  - The port cannot be disabled or blocked. The "Unicast Blocking" function cannot be activated for the port.
  - The port cannot be used for a link aggregation.
  - The port cannot be a monitor port of the "Mirroring".
  - The port cannot be a spanning tree port.
  - The port cannot be a ring port.
  - The port cannot be a router port.
  - The port must be a member of "vlan1".
  - LLDP (Tx and Rx) must be activated for the port.
  - The port cannot be an 802.1X Authenticator Port.
  - The port cannot be an 802.1X Supplicant Port.
- Wait
  If this check box is selected, the configuration of all four devices of an MRP Interconnection link is awaited before data transfer starts.
- Role
  From this drop-down list, select the role and, in the case of the "Client" role, the position of the client. The following selections are possible:
  - Manager
  - Primary Client
  - Secondary Client
- State
  Select this check box to enable the MRP Interconnection. Observe the following rules:
  - If there is not at least one activated MRP Interconnection, you cannot enable the MRP Interconnection for the device.
  - A maximum of one MRP Interconnection can be activated.

Spanning Tree

General

General settings of MSTP

This is the basic page for spanning tree. Select the compatibility mode from the drop-down list. On the configuration pages of these functions, you can make further settings. Depending on the compatibility mode, you can configure the corresponding function on the relevant configuration page.

Settings

- **Spanning Tree**
  Enable or disable Spanning Tree.
- **Protocol Compatibility**
  Select the compatibility mode of MSTP. For example if you select RSTP, MSTP behaves like RSTP.
  The following settings are available:
  - STP
  - RSTP
  - MSTP
- **RSTP+**
  RSTP+ enables the linking of a network segment in which Spanning Tree is activated with an MRP ring. Make sure that the following requirements have been met before selecting this check box:
  - MRP must be activated as redundancy method.
  - If ring redundancy is activated, you need to disable the ring ports for Spanning Tree.

  When you activate RSTP+, the ring ports become both part of the MRP ring and part of the Spanning Tree network segment.
- **RSTP+ MRP Interconnection Domain ID**
  Configure the MRP interconnection domain ID for RSTP+ here. This value must not match the MRP interconnection domain ID configured for the active MRP interconnection.

See also

General information (Page 2201)

MST port (Page 2197)

MST General (Page 2196)

CIST general/ST general

The page consists of the following parts.

- The left-hand side of the page shows the configuration of the device.
- The central part shows the configuration of the root bridge that can be derived from the spanning tree frames received by a device.
- The right-hand side shows the configuration of the regional root bridge that can be derived from the MSTP frames. The displayed data is only visible if you have enabled "Spanning Tree" on the "General" page and if "MSTP" is set for "Protocol Compatibility". This also applies to the "Bridge Max Hop Count" parameter. If the device is a root bridge, the information on the left and right matches.

Description of the displayed boxes

- **Bridge Priority / Root Priority** (only available online)
  Which device becomes the root bridge is decided based on the bridge priority. The bridge with the highest priority becomes the root bridge. The lower the value, the higher the priority. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames.
- **Bridge Address / Root Address**(only available online)
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Port** (only available online)
  Shows the port via which the switch communicates with the root bridge.
- **Root Cost** (only available online)
  The path costs from this device to the root bridge.
- **Topology Changes** / **Last Topology Change** (only available online)
  The entry for the device shows the number of reconfigurations due to the spanning tree mechanism since the last startup. For the root bridge, the time since the last reconfiguration is displayed as follows:
  - Seconds: sec unit after the number
  - Minutes: min unit after the number
  - Hour: hr unit after the number
- **Bridge Hello Time [s] / Root Hello Time[s]** (only available online)
  Each bridge regularly sends configuration frames (BPDUs). The interval between two configuration frames is the "Hello Time". Factory setting: 2 seconds

  Note

  The setting of the "Bridge Hello Time" is only possible with the protocol compatibility RSTP. If the protocol compatibility MSTP is set, the "Hello Time" parameter on the page "Layer 2 > Spanning Tree > CIST Port" page is used.

- **Bridge Forward Delay [s] / Root Forward Delay [s]**
  New configuration information is not used immediately by a bridge but only after the period specified in the "Forward Delay" parameter. This ensures that operation is only started with the new topology after all the bridges have the required information.
  Factory setting 15 seconds
- **Bridge Max Age / Root Max Age** (only available online)
  If the BPDU is older than the specified "Max Age" it is discarded. Factory setting: 20 seconds
- **Regional root priority** (only available online)
  For a description, see Bridge Priority / Root Priority
- **Regional root address** (only available online)
  The MAC address of the device.
- **Regional root costs** (only available online)
  Shows the path costs from this device to the regional root bridge.
- **Bridge Max Hop Count**
  This parameter specifies how many MSTP nodes a BPDU may pass through. If an MSTP BPDU is received and has a hop count that exceeds the value configured here, it is discarded. The default for this parameter is 20.
- **Region Name**
  Enter the name of the MSTP region to which this device belongs. As default, the MAC address of the device is entered here. This value must be the same on all devices that belong to the same MSTP region.
- **Region Version**
  Enter the version number of the MSTP region in which the device is located. This value must be the same on all devices that belong to the same MSTP region.
- **Reset Counter** (only available online)
  Click this button to reset the counters on this page to zero.

CIST Port/ST Port

MSTP-CIST port configuration

When the page is called, the table displays the current status of the configuration of the port parameters.

To configure them, click the relevant cells in the port table.

Description of the displayed boxes

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Spanning Tree Status**
  Select the setting. The following settings are possible:
  - Enabled
    Port is integrated in the spanning tree.
  - Disabled
    Port is not integrated in the spanning tree.
  - No change
    Table 2 remains unchanged.
- Transfer to table
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows all available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Spanning Tree Status**
  Specify whether or not the port is integrated in the spanning tree.

  Note

  If you disable the "Spanning Tree Status" setting for a port, this may cause the formation of loops. The topology must be kept in mind.

- **Priority**
  Enter the priority of the port. The priority is only evaluated if the path costs are the same. The value must be divisible by 16. If the value that cannot be divided by 16, the value is automatically adapted.
  Range of values: 0 - 240. The default is 128.
- **Cost Calc.**
  Enter the path cost calculation. If you enter the value "0" here, the automatically calculated value is displayed in the "Path Costs" box.

- **Path Costs** (only available online)
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected as the path. If several ports of a device have the same value for the path costs, the port with the lowest port number is selected.
  If The "Cost Calc." is "0", the automatically calculated value is shown. Otherwise, the value of the "Cost Calc." box is displayed. The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000

  The values can, however, also be set individually.
- **Status** (only available online)
  Displays the current status of the port. The values are only displayed and cannot be configured. The "Status" parameter depends on the configured protocol. The following is possible for status:
  - Disabled

    The port only receives and is not involved in STP, MSTP and RSTP.
  - Discarding
    In the "Discarding" mode, BPDU frames are received. Other incoming or outgoing frames are discarded.
  - Listening

    In this status, BPDUs are both received and sent. The port is involved in the spanning tree algorithm.
  - Learning

    Stage prior to the forwarding status, the port is actively learning the topology (in other words, the node addresses).
  - Forwarding

    Following the reconfiguration time, the port is active in the network; it receives and forwards data frames.

- **Fwd. Trans** (only available online)
  Specifies the number of changes from the "Discarding" status to the "Forwarding" status
- **Edge type**
  Specify the type of edge port. You have the following options:
  - "-"
    Edge port is disabled. The port is treated as a "no edge port".
  - Admin
    Select this option when there is always an end device on this port. Otherwise a reconfiguration of the network will be triggered each time a connection is changed.
  - Auto
    Select this option if you want a connected end device to be detected automatically at this port. When the connection is established the first time, the port is treated as a "no Edge Port".
  - Admin/Auto
    Select these options if you operate a combination of both on this port. When the connection is established the first time, the port is treated as an Edge Port.
- **Edge** (only available online)
  Shows the status of the port.
  - Enabled

    An edge port is connected to this port.
  - Disabled
    There is a spanning tree or rapid spanning tree device at this port.

  With an end device, a switch can change over the port faster without taking into account spanning tree frames. If a spanning tree frame is received despite this setting, the port automatically changes to the "Disabled" setting for switches.
- **P.t.P. Type**
  Select the required option. The selection depends on the port that is set.
  - "-"
    Point to point is calculated automatically. If the port is set to half duplex, a point-to-point link is not assumed.
  - P.t.P.

    Even with half duplex, a point-to-point link is assumed.
  - Shared Media

    Even with a full duplex connection, a point-to-point link is not assumed.

    Note

    Point-to-point connection means a direct connection between two devices. A shared media connection is, for example, a connection to a hub.

- **Hello Time**
  Enter the interval after which the bridge sends configuration frames (BPDUs). As default, 2 seconds is set. Range of values: 1-2 seconds

  Note

  The port-specific setting of the Hello time is only possible with protocol compatibility MSTP. If the protocol compatibility RSTP is set, the "Bridge Hello Time" parameter on the page "Layer 2 > Spanning Tree > CIST General" page is used.

- **Restr. Role**
  If this check box is selected, the corresponding port is not selected as root port, regardless of the priority value. If the check box is selected, the port with lowest priority also does not become the root port. Only activate this option if you wish to restrict the impact of bridges outside of the administered range on the Spanning Tree topology.
- **Limited TCN**
  If this check box is selected, the corresponding port does not forward either received or detected topology changes (Topology Change Notification) to other ports. Only activate this option if you wish to restrict the impact of bridges outside of the administered range on the Spanning Tree topology.
- **Limited TCN**
  If this check box is selected, the corresponding port accepts received and detected topology changes, but does not forward topology changes to other ports. You can only select the check box in this column if the following conditions are met:
  – RSTP+ must be activated.
  – The "Limited TCN" check box must be cleared for this port.

  If the specified conditions are not met, the check box in this column is shown grayed out.

MST General

Multiple Spanning Tree configuration

With MSTP, in addition to RSTP, several VLANs can be managed in a LAN with separate RSTP trees.

Description of the displayed boxes

- **MSTP Instance ID**
  Enter the number of the MSTP instance. Permitted values: 1 - 64

The table has the following columns:

- **Select**
  Select the row you want to delete.
- **MSTP Instance ID**
  Shows the number of the MSTP instance.
- **Root Address**
  Shows the MAC address of the root bridge

- **Root Priority**
  Shows the priority of the root bridge.
- **Bridge Priority**
  Enter the bridge priority in this box. The value for the bridge priority is a whole multiple of 4096 with a range of values from 0 to 61440.
- **VLAN ID**
  Enter the VLAN ID. Here, you can also specify ranges with Start ID, "-", End ID. Several ranges or IDs are separated by ",". Permitted values: 1- 4094

MST port

Configuration of the Multiple Spanning Tree port parameters

On this page, you set the parameters for the ports of the configured multiple spanning tree instances.

Description of the displayed boxes

The page contains the following box:

- **MSTP Instance ID**
  Select the ID of the MSTP instance.

Table 1 has the following columns

- **1st column**
  Shows that the setting is valid for all ports.
- **MSTP Status**
  Select the setting. The following settings are possible:
  - Enabled
  - Disabled
  - No Change: Table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2

Table 2 has the following columns:

- **Port**
  Shows all available ports and link aggregations.
- **MSTP Instance ID**
  ID of the MSTP instance.
- **MSTP Status**
  Click the check box to enable or disable this option.
- **Priority**
  Enter the priority of the port. The priority is only evaluated if the path costs are the same. The value must be divisible by 16. If the value that cannot be divided by 16, the value is automatically adapted.
  Range of values: 0 - 240. Factory setting: 128

- **Cost Calc.**
  Enter the path cost calculation in the input box. If you enter the value "0" here, the automatically calculated value is displayed in the next box "Path Costs".
- **Path Costs** (only available online)
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected as the path. If several ports of a device have the same value for the path costs, the port with the lowest port number is selected.
  If the "Cost Calc." box has the value "0", the automatically calculated value is shown. Otherwise, the value of the "Cost Calc." box is displayed. The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000

  The values can, however, also be set individually.
- **Status** (only available online)
  Displays the current status of the port. The values are only displayed and cannot be configured. The following is possible for status:
  - Discarding
    The port exchanges MSTP information but is not involved in the data traffic.
  - Blocked
    In the blocking mode, BPDU frames are received.
  - Forwarding
    The port receives and sends data frames.
- **Fwd. Trans**(only available online)
  Specifies the number of status changes Discarding - Forwarding or Forwarding - Discarding for a port.

Enhanced Passive Listening Compatibility

Enabling the function

On this page, you can enable passive listening compatibility.

Description of the displayed boxes

The page contains the following boxes:

- **Enhanced Passive Listening Compatibility**
  Enable or disable this function for the entire device.
- **Setting**
  - Enabled
    Enables the function for all ports of the device.
  - Disabled
    Disables the function for all ports of the device.
  - No change
    No change
- **Copy to Table**
  Writes the setting made in "Setting" to the following table.

Port-specific table:

If the function is enabled for the entire device, enable or disable this function on individual ports.

- **Port**
  Displays the port of the device.
- **Setting**
  Enable or disable the function for this port.

Loop detection

With the "Loop Detection" function, you specify the ports for which loop detection will be activated. The ports involved send special test frames - the loop detection frames. If these frames are sent back to the device, there is a loop.

A "local loop" involving this device means that the frames are received again at a different port of the same device. If the sent frames are received again at the same port, there is a "remote loop" involving other network components.

Note

A loop is an error in the network structure that needs to be eliminated. The loop detection can help to find the errors more quickly but does not eliminate them. The loop detection is not suitable for increasing network availability by deliberately including loops.

Note

Note that loop detection is only possible at ports that were not configured as ring ports or standby ports.

Description of the displayed boxes

- **Loop Detection**
  Enable or disable loop detection.
- **VLAN Loop Detection**
  Enable or disable loop detection in a VLAN.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2
- **Interval [ms] / Threshold / Timeout [s]/ Remote Reaction / Local Reaction**
  Select the required settings.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Specify how the port handles loop detection frames.

  Note

  Test frames create additional network load. We recommend that you only configure individual switches, for example at branch points of the ring, as "sender" and the others as "forwarder".

  – sender
    Loop detection frames are sent out and forwarded.
  – forwarder
    Loop detection frames from other devices are forwarded.
  – blocked
    The forwarding of loop detection frames is blocked.
- **Interval [ms]**
  Specifies the send interval for loop detection frames in milliseconds.
- **Threshold**
  By entering a number, specify the number of received loop detection frames as of which a loop is assumed.
- **Timeout** [s]
  Specify the number of seconds after which the device automatically changes to the status in which it was before the loop. If the value "0" is set, you need to enable the port manually again following a loop with the "Reset" button.

- Remote Reaction
  Specify how the port will react if a remote loop occurs.
  - No Change
    A loop has no effect on the port.
  - Disabled
    The port is blocked.
- Local Reaction
  Specify how the port will react if a local loop occurs.
  - No Change
    A loop has no effect on the port.
  - Disabled
    The port is blocked.
- Status (only available online)
  This box shows whether loop detection is enabled or disabled for this port.
- Source Port (only available online)
  Shows the receiving port of the loop detection frame that triggered the last reaction
- Source VLAN (only available online)
  Shows the VLAN ID of the loop detection frame that triggered the last reaction.
  This requires that the "VLAN Loop Detection" setting is selected.
- Reset (only available online)
  After a loop in the network has been eliminated, click this button to reset the port again.

Changing the configured port status with loop detection

The configuration of the port status can be changed with the "Loop Detection" function. If, for example, the administrator has disabled a port, the port can be enabled again after a device restart with "Loop Detection”. The port status "Link down" is not changed by "Loop Detection".

Effects of configuration using STEP 7

The configuration of Spanning Tree can be changed if you configure "Loop Detection" using STEP 7. If you enable "Loop Detection" on a port with STEP 7, Spanning Tree is automatically disabled on this port. If you disable "Loop Detection" for the port again with STEP 7 Spanning Tree is not automatically enabled. Enable Spanning Tree with the WBM or CLI.

Link aggregation

General information

Bundling network connections for redundancy and higher bandwidth

The link aggregation according to IEEE 802.3ad allows several connections between neighboring devices to be bundled to achieve higher bandwidths and additional protection against failure.

Ports on both partner devices are included in link aggregations and the devices are then connected via these ports. To assign ports (in other words links) correctly to a partner device, the Link Aggregation Control Protocol (LACP) from the IEEE 802.3ad standard is used.

Up to 8 link aggregations can be defined. A maximum of 8 ports can be assigned to each link aggregation.

Note

When a port is assigned to a link aggregation but is not active (e.g. link down), the values displayed may differ from the values configured for the link aggregation. If the port in the link aggregation becomes active, individual port configurations such as DCP forwarding are overwritten with the configured values of the link aggregation.

Description of the displayed boxes

This page displays all the configured link aggregations. The table has the following columns:

- **Port**
  Shows the virtual port number of this link aggregation. This identifier is assigned internally by the firmware.
- **Link Aggregation Name**
  Shows the name of the link aggregation. This name can be specified by the user during configuration. The name is not absolutely necessary but can be useful to distinguish between the various link aggregations.
- **MAC Address**
  Shows the MAC address.
- **Status**
  Enable or disable the link aggregation.
- **MTU**
  Specify the packet size.
- **LACP**
  - On
    Enables the sending of LACP frames.
  - Off
    Disables the sending of LACP frames.
- **Frame Distribution**
  Set the type of distribution of frames on the individual links of an aggregation.
  - Destination&Source Mac
    The distribution is based on a combination of the destination and source MAC address.
  - Destination&Source IP-MAC
    The distribution is based on a combination of the destination and source IP address and MAC address.

- VLAN Mode
  Specify how the link aggregation is entered in a VLAN:
  - Hybrid
    The link aggregation sends tagged and untagged frames. It is not automatically a member of a VLAN.
  - Trunk
    The link aggregation only sends tagged frames and is automatically a member of all VLANs.
- List of ports
  Shows the ports that belong to this link aggregation. The following values can be selected from the drop-down list:
  - "-" (deactivated)
    Link aggregation is disabled.
  - "a" (active)
    The port sends LACP frames and is only involved in the link aggregation when LACP frames are received.
  - "p" (passive)
    The port is only involved in the link aggregation when LACP frames are received.
  - o" (on)
    The port is involved in the link aggregation and does not send any LACP frames.

Note

Within a "link aggregation", only ports with the following configuration are possible:
- all ports with "o"
- all ports with "a" or "p".

Note

If you add a configured port to a link aggregation, the port adopts the configuration of the link aggregation. If you take the port out of the link aggregation, the settings of the port are reset to the factory settings.

See also

General (Page 2190)

LACP timeout

Configuration of the LACP timeout

In the IEEE 802.3ad standard, two possible values are defined for the length of the timeout, "Long" (90 seconds) and "Short" (3 seconds). This value defines the interval at which LACPDUs are sent. The value "Long" is configured by default for all ports. To enable a symmetrical LACP configuration, the value "Short" can optionally be selected. For all ports of a link aggregation, select the same value for the timeout.

Description of the displayed boxes

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **Setting**
  Select the setting from the drop-down list. You have the following setting options:
  - Short
    The value for the LACP timeout is 3 seconds.
  - Long
    The value for the LACP timeout is 90 seconds.
  - No Change
    Table 2 remains unchanged.
- **Copy to Table**
  When you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows all available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Setting**
  Select the value "Short" or "Long" for this port.

DCP forwarding

Applications

The DCP protocol is used by STEP 7 and the PST Tool for configuration and diagnostics. When shipped, DCP is enabled on all ports; in other words, DCP frames are forwarded at all ports. With this option, you can disable the sending of these frames for individual ports, for example to prevent individual parts of the network from being configured with the PST Tool or to divide the full network into smaller parts for configuration and diagnostics.

Note

PNIO configuration

Since DCP is a PROFINET protocol, the configuration created here is only effective with the VLAN associated with the TIA interface.

All the ports of the device are displayed on this page. After each displayed port, there is a drop-down list for function selection.

Description of the displayed values

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Setting**
  Specify whether the port should block or forward outgoing DCP frames. You have the following options available:
  - Forward
    DCP frames are forwarded at this port.
  - Block
    No outgoing DCP frames are forwarded via this port. It is nevertheless still possible to receive via this port.

LLDP

Identifying the network topology

LLDP (Link Layer Discovery Protocol) is defined in the IEEE 802.AB standard.

LLDP is a method used to discover the network topology. Network components exchange information with their neighbor devices using LLDP.

Network components that support LLDP have an LLDP agent. The LLDP agent sends information about itself and receives information from connected devices at periodic intervals. The received information is stored in the MIB.

Applications

PROFINET uses the LLDP protocol for topology diagnostics. In the default setting, LLDP is enabled for all ports; in other words, LLDP frames are sent and received on all ports. With this function, you have the option of enabling or disabling sending and/or receiving per port.

Description

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Setting**
  Select whether or not the port will send or receive LLDP frames. The following options are available:
  - Rx
    This port can only receive LLDP frames.
  - Tx
    This port can only send LLDP frames.
  - Rx & Tx
    This port can receive and send LLDP frames.
  - "-" (disabled)
    This port can neither receive nor send LLDP frames.

FMP

Requirements

- You can only use Fiber Monitoring with transceivers capable of diagnostics. Note the documentation of the devices.
- To be able to use the Fiber Monitoring function, enable LLDP. The fiber monitoring information is appended to the LLDP packets.

Monitoring optical links

With Fiber Monitoring, you can monitor the received power and the loss of power on optical links between two switches.

If you enable FM on an optical port, diagnostics information is sent via the port. In addition to sending, there is also a check to establish whether information is being received.

Regardless of whether the IE switch receives diagnostics information from its connection partner, it monitors the received power measured at the optical port for the set limit values.

If fiber monitoring is enabled on the connection partner, the connection partner transfers the current value for the transmit power of the port to the device. The device compares the value it has received for the transmit power with the actually received power. The difference between the received power and the transmit power represents the power loss on the link. The calculated power loss is also monitored for the set limit values.

If the value of the received power or the power loss falls below or exceeds the set limit values, an event is triggered. You can set limit values in two stages for messages with the severity levels "Warning" and "Critical".

In "System > Events > Configuration", you can specify how the IE switch indicates the event.

Note

If you have enabled Fiber Monitoring and a pluggable transceiver with diagnostics capability is pulled, Fiber Monitoring is automatically disabled for this port and the set limit values and a possibly pending error status are deleted.

Description of the displayed boxes

In the table you can specify the limit values for the measured received power to be monitored and the calculated power loss.

- **Port**
  Shows the available optical ports that support fiber monitoring. This depends on the transceivers.
- **State**
  Enable or disable FM. As default, the function is enabled.
- **Rx Power [dBm] maintenance required (warning)**
  Enter the value at which you are informed of the deterioration of the received power by a message of the severity level “Warning". If you enter the value "0", the received power is not monitored. The default value depends on the relevant transceiver. The default value depends on the relevant pluggable transceiver.
- **Rx Power [dBm] maintenance demanded (critical)**
  Enter the value at which you are informed of the deterioration of the received power by a message of the severity level “Critical". If you enter the value "0", the received power is not monitored. The default value depends on the relevant transceiver.
  The default value depends on the relevant pluggable transceiver.
- **Power Loss [dB] maintenance required (warning)**
  Specify the value at which you are informed of the power loss of the connection by a message of the severity level "Warning". If you enter the value "0", the power loss is not monitored. Default: -50 dB
- **Power Loss [dB] maintenance demanded (critical)**
  Specify the value at which you are informed of the power loss of the connection by a message of the severity level "Critical". If you enter the value "0", the power loss is not monitored. Default: -55 dB

Unicast

Filtering

Address filtering

This table shows the unicast addresses entered statically by the user during parameter assignment. On this page, you also define the static unicast filters.

Description of the displayed boxes

The page contains the following boxes:

- VLAN ID
  Select the VLAN ID in which you configure a new static MAC address. If nothing is set, "VLAN1" is set as the basic setting.
- MAC Address
  Enter the MAC address here.

This table contains the following columns:

- VLAN ID
  Shows the VLAN-ID assigned to this MAC address.
- MAC Address
  Shows the MAC address of the node that the device has learned or the user has configured.
- Status
  Shows the status of each address entry:
  - Static
    Configured by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the switch is restarted.
  - Invalid
    These values are not evaluated.
- Port
  Shows the port via which the node with the specified address can be reached. Frames received by the device whose destination address matches this address will be forwarded to this port.

  Note

  You can only specify **one** port for unicast addresses.

Locked ports

Activating the access control

On this page, you can block individual ports for unknown nodes.

If the Port Lock function is enabled, packets arriving at this port from unknown MAC addresses are discarded immediately. Packets from known nodes are accepted by the port.
Since ports with the port lock function enabled cannot learn any MAC addresses, learned addresses on these ports are automatically deleted after the port lock function is enabled. The port accepts only static MAC addresses that were created previously either manually or with the "Start learning" function and the "Stop learning" function.

To enter all connected nodes automatically, there is a function for automatic learning (see "Layer 2 > Unicast > Learning").

Description of the displayed boxes

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the setting from the drop-down list. The following settings are possible:
  - Enabled
    Enables the port lock function.
  - Disabled
    Disables the port lock function.
  - No change
    Table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  This column lists all the ports available on this device.
- **Setting**
  Enable or disable access control for the port.

Learning

Starting/stopping learning

Note

The page is only available if there is an online connection to the device.

With the automatic learning function, all connected devices are automatically entered in the unicast filter table. As long as the "Start learning" function is enabled, all learned unicast addresses are created immediately as static unicast entries.
The learning process is ended only after clicking the "Stop learning" button. With this method, learning can take a few minutes or several hours in larger networks before all nodes have really been learned. Only nodes that send packets during the learning phase are found.

By subsequently enabling the Port Lock function, only packets from the nodes known after the end of the learning phase (static unicast entries) will be accepted at the relevant ports.

Note

If the Port Lock function was already active on individual ports prior to the automatic learning phase, no addresses will be learned on these ports. This makes it possible to restrict learning to certain ports. To do this, first enable the Port Lock function of the ports that are not intended to learn addresses.

Steps in configuration

Learning addresses

1. Click the "Start learning" button to start the learning phase. After starting the learning phase, the "Start learning" button is replaced by the "Stop learning" button. The device now enters the addresses of connected devices until you stop the function.
2. Click the "Stop learning" button to stop the learning phase. The button is once again replaced by the "Start learning" button. The learned entries are stored.

Deleting all static unicast addresses

1. Click the "Clear all static unicast addresses" button to delete all static entries.
   In large networks with numerous nodes, automatic learning may lead to a lot of undesired static entries. To avoid having to delete these individually, this button can be used to delete all static entries. This function is disabled during automatic learning.

   Note

   Depending on the number of entries involved, deleting may take some time.

Blocking

Blocking forwarding of unknown unicast frames

On this page, you can block the forwarding of unknown unicast frames for individual ports.

Description of the displayed values

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the setting from the drop-down list. The following settings are possible:
  - Enabled
    Blocking of unicast frames is enabled.
  - Disabled
    Blocking of unicast frames is disabled.
  - No change
    Table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  All available ports are listed in this column. Unavailable ports are not displayed.
- **Setting**
  Enable or disable the blocking of unicast frames.

Multicast

Groups

Multicast applications

In the majority of cases, a frame is sent with a unicast address to a particular recipient. If an application sends the same data to several recipients, the amount of data can be reduced by sending the data using one multicast address. For some applications, there are fixed multicast addresses (NTP, IETF1 Audio, IETF1 Video etc.).

Reducing network load

In contrast to unicast frames, multicast frames represent a higher load for the device. Generally, multicast frames are sent to all ports. There are three ways of reducing the load caused by multicast frames:

- Static entry of the addresses in the multicast filter table.
- Dynamic entry of the addresses by listening in on IGMP parameter assignment frames (IGMP Configuration).
- Active dynamic assignment of addresses by GMRP frames.

The result of all these methods is that multicast frames are sent only to ports for which an appropriate address is entered.

The "Multicast" menu item, shows the multicast frames currently entered in the filter table and their destination ports that the user set in the parameters (static).

Description of the displayed boxes

The page contains the following boxes:

- VLAN ID
  If you click on this text box, a drop-down list is displayed. Here you can select the VLAN ID of a new MAC address you want to configure.
- MAC Address
  Here you enter a new MAC multicast address you want to configure.

The table has the following columns:

- VLAN ID
  Here, the VLAN ID of the VLAN is displayed to which the MAC multicast address of this row is assigned.
- MAC Address
  Shows the MAC multicast address that the device has learned or the user has configured.
- Status
  Shows the status of each address entry. The address was entered statically by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the device is restarted. These must be deleted by the user.
- List of ports
  There is a column for each slot. Within a column, the multicast group to which the port belongs is shown. The drop-down list provides the following options:
  - M
    (Member) Multicast frames are sent over this port.
  - F
    (Forbidden) Not a member of the multicast group. Moreover, this address must not be an address learned dynamically with GMRP or IGMP/MLD.
  - -
    Not a member of the multicast group. No multicast frames with the defined multicast MAC address are sent via this port

IGMP

Function

The device supports "IGMP Snooping" and the "IGMP Querier" function. If "IGMP Snooping" is enabled, IGMP frames (Internet Group Management Protocol) are evaluated and the multicast filter table is updated with this information. If "IGMP Querier" is enabled as well, the device also sends IGMP queries that trigger responses from IGMP-compliant nodes.

IGMP Snooping Aging Time

In this menu, you can configure the aging time for IGMP Configuration. When the time elapses, entries created by IGMP are deleted from the address table if they are not updated by a new IGMP frame.

This applies to all ports; a port-specific configuration is not possible.

IGMP Snooping Aging Time depending on the querier

SCALANCE XR500 as IGMP querier:

If a SCALANCE XR500 is used as an IGMP querier, the query interval is 125 seconds. For the "IGMP Snooping Aging Time", set at least 250 seconds.

Other IGMP queriers

If a different IGMP querier is used, the value of the "IGMP Snooping Aging Time" should be at least twice as long as the query interval.

Description of the displayed boxes

The page contains the following boxes:

- IGMP Snooping
  Enable or disable IGMP snooping (Internet Group Management Protocol). The function activates IGMP snooping on all interfaces. If the function is enabled, the multicast addresses learned with IGMP snooping are entered in the multicast filter table.
- IGMP Snooping Aging Time
  Enter the value for the aging time in seconds in this box. As default, 300 seconds is set. Range of values: 130 - 1225 seconds
- IGMP Querier
  Enable or disable "IGMP Querier". The device sends IGMP queries cyclically.
- IGMP Snooping Switch IP Address
  After the "IGMP Querier" check box is selected, the IP address of the device is shown here. You have the option of entering the IP address of a different device that should take on the function of the IGMP Querier.

The table has the following columns:

- VLAN ID
  The VLAN ID for which IGMP Snooping or IGMP Querier should be activated.
- IGMP Snooping
  Select the check box in this column for the VLANs for which IGMP Snooping should be activated. The specifications in this column only become valid when you enable IGMP Snooping for the device (first check box on this page).
- IGMP Querier
  Select the check box in this column for the VLANs for which IGMP Querier should be activated. The prerequisite is that IGMP Snooping is activated for the corresponding VLAN.

Static IGMPs

Function

Static multicast addresses are entered by the user into the filter table and permanently stored there. In contrast to entries created dynamically, static entries in the filter table are not deleted after the aging time or when the device is restarted. These must be deleted by the user.

Description of the displayed boxes

The page contains the following boxes:

- VLAN ID
  Select the VLAN ID of the group to be configured from this drop-down list.
- Start Port
  From this drop-down list, select the first port of a port range for which you want to configure a multicast address.
- End Port
  The selection made in the "Start Port" drop-down list is first displayed here. From this drop-down list, select the last port of a port range for which you want to configure a multicast address. If you do not make a selection in this drop-down list, only one port is assigned to the multicast address.
- MAC address
  Specify the multicast IP address of the group.

The table has the following columns:

- Select
  Select the row you want to delete.
- VLAN ID
  The ID of the VLAN to which the multicast address of this row is assigned is displayed here.
- Group Address
  The multicast IP address of the group is displayed here.
- List of ports
  There is a column with a check box for each port. If the check box is selected, the port belongs to the corresponding group.

Configuration procedure

Creating static multicast group

Note

You cannot create any static multicast entries if GMRP is enabled.

1. Select the required "VLAN ID" from the drop-down list.
2. In the Start Port and End Port drop-down lists, enter the ports to be members of the multicast group.

3. In the "Group Address" drop-down list, enter the multicast IP address of the group.
4. Click the "Create" button. A new entry is generated in the table.
5. You can change existing groups by selecting/clearing the check boxes of the corresponding ports.

GMRP

Enabling GMRP

By selecting the check box, you specify whether or not GMRP is used for each individual port. If "GMRP" is disabled for a port, no registrations are made for it and it cannot send GMRP frames.

For GMRP to work, you need to enable the function globally and on the ports.

Description of the displayed boxes

The page contains the following box:

- **GMRP**
  Enable or disable the GMRP function.

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. The following settings are possible:
  - Enabled
    Enables the sending of GMRP frames.
  - Disabled
    Disables the sending of GMRP frames.
  - No change
    Table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  This column shows all the ports available on the device as well as the link aggregations.
- **Setting**
  With this check box, you enable or disable GMRP for the port or link aggregation.

Multicast blocking

On this page, you can block the forwarding of unknown multicast frames for individual ports.

Description of the displayed values

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. The following settings are possible:
  - Enabled
    Blocking of multicast frames is enabled.
  - Disabled
    Blocking of unknown multicast frames is disabled.
  - No change
    Table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  All available ports are listed in this column. Unavailable ports are not displayed.
- **Setting**
  Enable or disable the blocking of multicast frames.

MLD

Mulitcast Listener Discovery

Devices support "MLD snooping" and the "MLD querier" function. If "MLD snooping" is enabled, MLD frames are evaluated and the multicast filter table is updated with this information. If "IGMP querier is also enabled, IE switches also send MLD queries that trigger responses from MLD-compliant nodes.

With MLD snooping, MLD frames are only forwarded to the intended multicast listeners instead of being flooded to all ports.

For MLD snooping to work, you need to enable the function globally and in the VLANs.

MLD Snooping Aging Time

On this page, you can configure the aging time for MLD configuration. When the time elapses, entries created by MLD are deleted from the address table if they are not updated by a new MLD packet.

This applies to all VLANS; a VLAN-specific configuration is not possible.

Description of the displayed boxes

- MLD Snooping
  Enable or disable MLD on the device.
- MLD Snooping Aging Time
  Enter the time after which entries generated by MLD are deleted from the multicast filter table. Assuming that this is not updated by a new MLD packet.
  This applies to all VLANS; a VLAN-specific configuration is not possible

Table 1 has the following columns:

- 1st column
  Shows that the settings are valid for all VLANs of table 2.
- Snooping / Querier
  - Enabled
    The function is enabled.
  - Disabled
    The function is disabled.
  - No change
    Table 2 remains unchanged.
- Copy to Table
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- VLAN
  Shows the VLAN to which the settings relate
- Snooping
  Enable or disable MLD for the required VLAN. When enabled, the MLD entries are included the multicast filter table and MLD packets are forwarded.
- Querier
  Enable or disable sending of MLD queries

Broadcast

Blocking the forwarding of broadcast frames

On this page, you can block the forwarding of broadcast frames for individual ports.

Note

Some communication protocols work only with the support of broadcast. In these cases, blocking can lead to loss of data communication. Block broadcast only when you are sure that you do not need it.

Description of the displayed boxes

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. The following settings are possible:
  Enabled
  The blocking of broadcast frames is enabled.
  Disabled
  The blocking of broadcast frames is disabled.
  No change
  Table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  All available ports and the link aggregations are shown.
- **Setting**
  Enable or disable the blocking of broadcast frames.

PTP

General

The following devices support time-of-day synchronization with PTP:

- SCALANCE XR528-6M
- SCALANCE XR552-12M

IEEE 1588 with SCALANCE devices

The IEEE 1588v2 standard defines mechanisms with which highly precise time-of-day synchronization of devices in a network can be achieved. SCALANCE devices with suitable hardware support time synchronization according to IEEE 1588v2.

The functionality is disabled on these devices when they are shipped and following a "Reset to factory defaults". To be able to use PTP, enable this function and configure every port that is on the synchronization path as well as ports that are blocked due to redundancy mechanisms. PTP can also be used with redundancy mechanisms in the ring such as HRP, standby linking of rings, MRP and RSTP.

Configuration

- **1588 Mode**
  You can make the following settings:
  - **off**
    The device does not process any PTP messages. PTP messages are, however, forwarded according to the rules of the switch.
  - **transparent**
    The device adopts the function of a transparent clock and forwards PTP messages to other nodes while at the same time making entries in the correction field of the PTP message.

TC general

On this page, you specify the general settings for PTP.

1588 Transparent Clock Configuration

Delay Mechanism

Specify the delay mechanism the device will work with:

- end to end
  (Delay request response mechanism will be used)

  Note

  With end-to-end synchronization with more than 2 slaves, freak values > 100 ns can occur in the offset.

- peer to peer
  (Peer delay mechanism will be used)

Domain Number

Enter the domain number for the device here. The device ignores PTP messages with a different domain number. A SCALANCE device can only be assigned to one synchronization domain.

TC port

This page contains the port settings for PTP.

1588 Transparent Clock Port Parameters

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Setting**
  Select the setting. If "No Change" is selected, the entry in table 2 remains unchanged.

- **Transport Mechanism**
  The following settings are possible:
  - Ethernet
  - UDP IPv4
  - No Change
    If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  The port number With modular devices, the slot number and port number are displayed separated by a dot. PTP is not supported on SFP+ ports.
- **Setting**
  The port status. The following entries are possible:
  - Enabled
    The port is not involved in PTP.
  - Disabled
    The port processes PTP messages.
- **Faulty Flag**
  The error status relating to PTP.
  - true
    An error occurred.
  - false
    No error has occurred on this port.
- **Transport mechanism**
  Choose how this port will handle PTP message data traffic. You can make different settings for the ports of a device, however, the relevant communications partner must support the selected transport mechanism. The following settings are possible:
  - Ethernet
  - UDP IPv4

RMON

Statistics

Statistics

On this page you can specify the ports for which RMON statistics are displayed. The RMON statistics are shown on the page “Information > Ethernet Statistics” in “Packet Size”, “Frame Type” and “Packet Errror”.

Settings

- **RMON**
  When enabled, Remote Monitoring (RMON) allows diagnostics data to be collected on the device, prepared and read out using SNMP by a network management station that also supports RMON. This diagnostic data, for example port-related load trends, allow problems in the network to be detected early and eliminated.

  Note

  If you disable RMON, these statistics are not deleted but retain their last status.

- **Port**
  Select the ports for which statistics will be displayed.

The table has the following columns:

- **Port**
  Shows the ports for which statistics will be displayed.

History

Samples of the statistics

On this page, you can specify whether or not samples of the statistics are saved for a port. You can specify how many entries should be saved and at which intervals samples should be taken.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Entries**
  Enter the maximum number of samples to be stored at the same time. If "No Change" is entered, the entry in table 2 remains unchanged
- **Interval [s]**
  Enter the interval after which the current status of the statistics will be saved as a sample. If "No Change" is entered, the entry in table 2 remains unchanged
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the port to which the settings relate.
- **Setting**
  Enable or disable the recording of the history on the relevant port.
- **Entries**
  Enter the maximum number of samples to be stored at the same time. The maximum number of entries can be restricted by the capacity of the device.
  Range of values: 1 - 65535
  Factory setting: 24
- **Interval [s]**
  Enter the interval after which the current status of the statistics will be saved as a sample.
  Range of values: 1 - 3600
  Factory setting: 3600

Configuring layer 3 functions for IPv4

Configuration

The page contains the overview of the layer 3 functions for IPv4 of the device. On this page, you enable or disable the required layer 3 function.

The functions "Routing", "VRRP", "RIP" and "OSPF" are only available with layer 3.

Description of the displayed boxes

The page contains the following boxes:

- **Routing**(only available with devices with a layer 3 license)
  - Enabled
    IPv4 routing is enabled. You can only enable the routing function if DHCP is disabled on all configured interfaces.
  - Disabled
    IPv4 routing is disabled. If IPv6 routing is enabled on the device, this is also disabled.
- **DHCP Relay Agent**
  Enable or disable the DHCP relay agent. You can configure other settings in "Layer 3 (IPv4)> DHCP Relay Agent".
- **VRRP** (only available with devices with a layer 3 license)
  Enable or disable routing using VRRP. To use VRRP, first enable the routing function. You can configure other settings in "Layer 3 (IPv4) > VRRP".
- **OSPF** (only available with devices with a layer 3 license)
  Enable or disable routing using OSPF. To use OSPF, first enable the Routing function. You can configure other settings in "Layer 3 (IPv4) > OSPF".

- **RIP** (only available with devices with a layer 3 license)
  Enable or disable routing using RIP. To use RIP , first enable the routing function. You can configure other settings in "Layer 3 (IPv4) > RIP".
- **PIM-Routing** (only available with devices with a layer 3 license)
  Enable or disable PIM routing. You can configure other settings in "Layer 3 (IPv4) > PIM".

Subnets

Overview (IPv4)

Creating subnets

The page shows the subnets for the selected interface. If more than one subnet is available on an interface, the first entry of this interface is of the address type "Primary". All other subnets are created on this page. A subnet always relates to an interface. The interface is created on the "Configuration" tab.

Description of the displayed values

The page contains the following boxes:

- **Interface**
  Select the interface on which you want to configure another subnet.
- **Loopback**
  Enable or disable the "loopback" property on the IP interface. If the "loopback" property is to be enabled for an IP interface, the assigned VLAN must not contain any member ports. Since the IP interface is not linked to physical ports, the operating status of the IP interface is not dependent on the link status of the physical ports. The loopback interface can be reached as long as the device is connected to a suitable power supply. Via the loopback interface you can access the device.

  Note

  An interface configured as loopback must be located in its own subnet with a subnet mask of at least 30 bits.

The table has the following columns:

- **Interface**
  Shows the interface.
- **TIA Interface**
  Shows whether or not the entry is a TIA interface.
- **Interface Name**
  Shows the name of the interface.
- **MAC Address**
  Shows the MAC address.
- **IP Address**
  Shows the IPv4 address of the subnet.

- **Subnet Mask**
  Shows the subnet mask.
- **Address Type**
  Shows the address type. The following values are possible:
  - Primary
    The first IPv4 address that was configured on an IP interface.
  - Secondary
    All other IPv4 addresses that were configured on an interface.
- **IP Assignment Method**
  Shows how the IPv4 address is assigned.
  - Static
    The IP address is static. You enter the IP settings in "IP Address" and "Subnet Mask".
  - Automatic (DHCP)
    The device obtains a dynamic IPv4 address from a DHCP server.
- **Address Collision Detection Status**
  If new IPv4 addresses become active in the network, this function checks whether this can result in address collisions. With this function, IPv4 addresses that will be assigned twice can be detected.
  Shows the status of the "Detection of address collisions" function.

  Note

  The function does not run a cyclic check.

  - Idle
    The interface is not enabled and does not have an IP address.
  - Starting
    This status indicates the start-up phase. In this phase, the device initially sends a query as to whether the planned IP address already exists. If the address is not yet been assigned, the device sends the message that it is using this IP address as of now.
  - Conflict
    The interface is not enabled. The interface is attempting to use an IP address that is already been assigned.
  - Defending
    The interface uses a unique IP address. Another interface is attempting to use the same IP address.
  - Active
    The interface uses a unique IP address. There are no collisions.
  - Not supported
    The function for detection of address collisions is not supported.
  - Disabled
    The function for detection of address collisions is disabled.
- **Loopback**
  Shows whether or not the "Loopback" property is enabled.

Configuration

On this page, you configure the subnet.

Description of the displayed values

The page contains the following boxes:

- **Interface (Name)**
  Select the required interface.
- **Interface Name**
  Enter a name for the interface.
- **MAC Address** (only available online)
  Shows the MAC address of the selected interface
- **DHCP**
  Enable or disable the DHCP client for this IPv4 interface.

  Note

  If you want to operate the device as a router with several interfaces, disable DHCP on all interfaces.

- **IP Address**
  Enter the IPv4 address of the subnet. The IPv4 address must not be used more than once.
- **Subnet Mask**
  Enter the subnet mask of the subnet you are creating. Subnets on different interfaces must not overlap.
- **Address Type**
  Shows the address type. The following values are possible:
  - Primary
    The first subnet of the interface.
  - Secondary
    All other subnets of the interface.
- **Loopback**
  Shows whether or not the "Loopback" property is enabled.

- IP Source
  When the option is enabled, the loopback interface is used as source IP address in the following protocols:
  - RADIUS
  - SNMPv1 Trap
  - Syslog
  - SNTP
  - NTP
  - SMTP
  - TFTP
  - SFTP

  This option can be helpful, for example, when using a firewall or authentication over RADIUS. Because a router with multiple IP interfaces does not always specify the same IP interface as source, without this setting all IP interfaces of a router would have to be configured on a RADIUS server or in a firewall. This setting makes it possible for a router to use a loopback interface of the outgoing IP communication and that only this loopback interface must be known to the RADIUS server and the firewall.
  When the option is disabled, the outgoing interface is specified as source in the protocols.
- TIA Interface
  Enable or disable the setting.
  The following conditions apply to the TIA interface:
  - Only interfaces with the address type "Primary" can be enabled as the TIA interface.
  - There must only ever be one TIA interface.
  - There can only be one TIA interface.
  - A TIA interface is always a VLAN interface.

TIA interface and PROFINET interface

The IP address of the TIA interface is linked to the IP address located in the parameter group "Properties > General > PROFINET interface > Ethernet addresses".

Both IP addresses always have the same value. If you change the value of one IP address, the other IP address changes accordingly.

TIA Interface

The TIA interface is the interface via which all PROFINET functions of the IE switch are handled.

TIA interface is the name of the IP address for PROFINET from the perspective of the IE switch.

PROFINET interface

You make the IP setting of the PROFINET device in TIA under PROFINET interface.

PROFINET interface is the name of the IP address for PROFINET from the perspective of the TIA Portal.

Default gateway

Creating subnets

On this page, you define the default gateway.

Description of the displayed values

The page contains the following boxes:

- **Default Gateway**
  Enter the IP address of the interface that is used as the default gateway.

NAT

NAT

On this WBM page, you specify the basic settings for NAT.

Description

The page contains the following boxes:

- **NAT**
  Enable or disable NAT/NAPT for the entire device. When enabled, the device operates as a NAT router.
- **Idle Timeout [s]**
  Enter the required time. The device checks cyclically after the set period has elapsed whether the aging time of TCP and UDP connections has elapsed. The connections whose aging time has elapsed since the last check are deleted from the table "NAT Translations".
- **TCP Timeout [s]**
  Enter the required aging time for TCP connections. TCP connections are stored until no data exchange has taken place for the set period. Depending on the cyclic check when the Idle Timeout has elapsed, the connections are deleted from the table "NAT translations".
- **UDP Timeout [s]**
  Enter the required aging time for UDP connections. UDP connections are stored until no data exchange has taken place for the set period. Depending on the cyclic check when the Idle Timeout has elapsed, the connections are deleted from the table "NAT translations".

- Interface
  Select an IP interface from the drop-down list on which you want to configure NAT. As soon as you have configured an interface as a NAT interface, all other configurations are considered starting from this interface. This means for this interface that all networks reachable via the interface itself count as "Outside". All other networks are "Inside".

  Note

  If you have configured several NAT interfaces on a device, this means that a network is "Outside" from the perspective of one NAT interface and "Inside" from the perspective of another NAT interface.

- NAT
  Enable or disable NAT for an IP interface.
  An entry is created automatically in the "Pool" tab. The device can be reached from the external network using the IP address of the IP interface.
  If you disable NAT for an IP interface and there are no configurations on the NAT interface, the entry is automatically deleted from the table.
  NAPT
  Enable or disable NAPT for an IP interface.

The table has the following columns:

- Interface
  Interface to which the setting relates.
- NAT
  Shows whether NAT is enabled or disabled for the selected IP interface. NAT is only enabled, when you have enabled NAT for the entire device.
- NAPT
  Shows whether . NAPT is enabled or disabled for the selected IP interface. NAPT is only enabled if you have enabled NAT for the entire device. If you do not create any further configurations for NAPT, the dynamic port translation is enabled automatically. As default, a device in the internal network cannot be reached from an external network. If the internal device wants to communicate in an external network, the inside local address and the IP address of the IP interface have a port added and assigned to the internal device as assigned as inside local and inside global address. Using this inside global address, the internal device can be reached from the external network until the timer of the connection elapses.

Static

On this WBM page, you configure static 1:1 address translations.

You specify which inside global address the inside local address of a device will be converted to and vice versa. This variant allows connection establishment in both directions. The device in the internal network can be reached from the external network.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  Select the a NAT interface from the drop-down list for which you want to create further NAT configurations.
- Inside Local Address
  Enter the actual address of the device that should be reachable from external.
- Inside Global Address
  Enter the address at which the device can be reached from external.

The table has the following columns:

- 1st column
  Select the check box in the row to be deleted.
- Interface
  NAT interface to which the setting relates.
- Inside Local Address
  Shows the actual address of the device that should be reachable from external.
- Inside Global Address
  Shows the address at which the device can be reached from external.

Pool

On this WBM page, you configure static address translations.

As default, the device cannot be reached from an external network. If the device wants to communicate in an external network, an inside global address is assigned to it dynamically. Using this inside global address, the device can be reached from the external network until the timer of the connection elapses.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  Select the a NAT interface from the drop-down list for which you want to create further NAT configurations.
- Inside Global Address
  Enter the start address for the dynamic assignment of addresses at which devices will be reachable from external.
- Inside Local Address Mask
  Enter the address mask of the external subnet.

The table has the following columns:

- 1st column
  Select the check box in the row to be deleted.
- Interface
  Interface to which the setting relates.

- Inside Global Address
  Shows the start address for the dynamic assignment of addresses at which devices will be reachable from external.
- Inside Local Address Mask
  Shows the address mask of the external subnet.

NAPT

On this WBM page, you configure static address translations.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  Select the a NAT interface from the drop-down list for which you want to create further NAT configurations.
- Inside Local Address
  Enter the actual address of the device that should be reachable from external.
- Service
  Select the service for which the port translation is valid.
  When you select a service, the same port is entered in the Start Port and End Port boxes.
  if you change the start port, the end port is changed accordngly.
  if you select the entry "-", you can enter the start and end port freely.
- Start Port
  Enter a port.
- End Port
  Depending on your selection in the "Service" drop down list, you can enter a port or a port is displayed.
  If you enter different ports in the Start Port and End Port boxes, the same port range is entered in the Inside Global Port box. A port range can only be translated to the same port range.
  If you enter the same port in the Start Port and End Port boxes, you can enter any Inside Global Port.
- Inside Global Port
  Depending on your selection in the "Service" drop down list, you can enter a port or a port is displayed.
- Protocol
  Select the protocol for which the port translation is valid.
- Description
  Enter a description for the port translation.

The table has the following columns:

- 1st column
  Select the check box in the row to be deleted.
- Interface
  NAT interface to which the setting relates.

- Inside Local Address
  Shows the actual address of the device that should be reachable from external.
- Start Port
  Shows the start port that will be assigned to the inside local address.
- End Port
  Shows the end port that will be assigned to the inside local address.
- Protocol
  Shows the protocol for which the port translation applies.
- Inside Global Address
  Shows the address at which the device will be reached from external.
- Inside Global Port
  Shows the port that will be assigned to the Inside Global Address.
- Description
  Shows a a description for the port translation.

Static Routes (IPv4)

Static route

On this page, you create the static IPv4 routes.

Description of the displayed boxes

The page contains the following boxes:

- Destination Network
  Enter the network address of the destination that can be reached via this route.
- Subnet Mask
  Enter the corresponding subnet mask.
- Gateway
  Enter the IPv4 address of the gateway via which this network address is reachable.
- Administrative Distance
  Enter the administrative distance for the route. The administrative distance corresponds to the quality of a connection, based for example on speed or costs. If there are several equal routes, the route with the lowest value is used. As default, -1 is set. This setting means that the metric is not set.
  Range of values: 1 - 254

The table has the following columns:

- Destination Network
  Shows the network address of the destination.
- Subnet Mask
  Shows the corresponding subnet mask.
- Gateway
  Shows the IPv4 address of the next gateway.

- **Interface** (only available online)
  Shows the Interface of the route.
- **Administrative Distance**
  Enter the administrative distance for the route. When creating the route, "not used" is entered automatically (value -1). The administrative distance corresponds to the quality of a connection, based for example on speed or costs. If there are several equal routes, the route with the lowest value is used. Range of values: 1 - 254
- **State** (only available online)
  Shows whether or not the route is active.

Route maps

General (IPv4)

Route maps

With route maps, you control how routing information is further processed. You can filter routing information and specify whether the information is further processed, modified or discarded.

Route maps operate according to the following principle:

- Routing information is compared with the filters of the route maps.
- The comparison is continued until the filters of a route map match the properties of an item of information.
- The information is then processed according to the route map settings:
  - The routing information is discarded.
  - The properties of the routing information are changed.

Description of the displayed boxes

- **Name**
  Enter the name for the route map.
- **Sequence No**
  Enter a number for the route map.
  You can create several route maps with the same name but with different sequence numbers. The sequence numbers then specify the order in which the route maps are processed.

The table has the following columns:

- Name
  Shows the name of the route map.
- Sequence No
  Shows the sequence number of the route map.
- Action
  Specify what happens to the routing information that matches the settings of the route map:
  - permit
    The routing information is further processed according to the settings you make on the "Set" page.
  - deny
    The routing information is discarded.

Interface & Value Match (IPv4)

On this page, you specify whether or not the routing information for a route map is filtered according to interfaces, metric or tags.

Description of the displayed boxes

- Route Map (name/seq. no.)
  Select a route map.
  The created route maps are available to you.
- Type
  Select the basis for the filtering:
  - Interface
  - Metric
  - Tag
  - Route Type
  - metric type
- Interface
  Select an interface.
  Only active if you have selected the "Interface" entry for "Type".
- Metric
  Enter a value for the metric.
  This box is only active if you have selected the "Metric" entry for "Type".
- Tag
  Enter a value for the tag.
  This box is only active if you have selected the "Tag" entry for "Type".

- **Route Type**
  Select the type of the route.
  - Local
    The routing information for the route map is filtered according to directly connected routes (local interfaces).
  - Remote
    The routing information for the route map is filtered according to learned or statically configured routes.

  This box is active only if i you have selected the entry "Route Type in the "Type" drop-down list.
- metric type
  Select the type of the metric.
  - Inter-Area
    The routing information for the route map is filtered according to routes learned from other areas.
  - Intra-Area
    The routing information for the route map is filtered according to routes originating from the same area.
  - type-1-external
    The routing information for the route map is filtered according to routes whose path costs (metric) are made up of the external path costs and the path costs to the ASBR.
  - type-2-external
    The routing information for the route map is filtered according to routes with only external path costs (metric).

  This box is active only if you have selected the entry "Metric Type" in the "Type" drop-down list.

The table has the following columns:

- **Type**
  Shows the selected type:
  - Interface
  - Metric
  - Tag
  - Route Type
  - metric type
- **Value**
  Shows the selected interface, the value of the metric or of the tag.

Destination Match (IPv4)

On this page, you specify whether or not the routing information for a route map is filtered based on the destination IPv4 address.

Description of the displayed boxes

- Route Map (name/seq. no.)
  Select a route map.
- IP Address
  Enter the IPv4 address of the destination on which the filtering is based.
- Subnet Mask
  Enter the subnet mask of the destination on which the filtering is based.

The table has the following columns:

- IP Address
  Shows the IPv4 address of the destination.
- Subnet mask/prefix
  Shows the subnet mask of the destination.

Next Hop Match (IPv4)

On this page, you specify whether or not the filtering for a route map will be based on the router to which the routing information is sent next.

Description of the displayed boxes

- Route Map (name/seq. no.)
  Select a route map.
- IP Address
  Enter the IP address of the router to which the routing information will be sent next.

The table has the following columns:

- IP Address
  Shows the IP address of the next router.

Set (IPv4)

On this page, you specify whether or not the routing information will be changed by a route map.

If, for example, you have filtered based on a certain metric, you can change the value of the metric here. The routing information is then forwarded with the new value. You can only change the information of a "Permit" route map.

Description of the displayed boxes

- Route Map (name/seq. no.)
  Select a route map.

The table has the following columns:

- Name
  Shows the name of the route map.
- Sequence No
  Shows the sequence number of the route map.
- Metric
  Enter the new value for the metric with which the routing information will be forwarded.
- Tag
  Enter the new value for the tag with which the routing information will be forwarded.

DHCP Relay Agent

General

If the DHCP server is in a different network from the DHCP client, the client cannot reach the server. The DHCP relay agent intercedes between the DHCP server and DHCP client. If you configure option 82, the DHCP relay agent expands the packets to the DHCP server by a circuit ID and a remote ID.

You can specify up to 4 DHCP servers for the DHCP relay agent. If a DHCP server is unreachable, the device can switch to a different DHCP server.

Description of the displayed boxes

- DHCP Relay Agent
  Enable or disable the DHCP relay agent.
- Send Option 82
  Enable or disable option 82.
- Common Agent Address
  Enable or disable the common agent address. When the function is activated, in the DHCP request, the relay agent replaces the address of the receiving port with the address of the interface that you configure in "Common Agent Interface".
- Common Agent Interface
  The relay agent uses the IP address of the interface selected here as the source address (giaddr) in DHCP requests.
- Server IP Address
  Enter the IPv4 address of the DHCP server.

The table has the following columns:

- Server IP Address
  Shows the IPv4 address of the DHCP server.

Option

Parameters of the DHCP relay agent

On this page, you can specify parameters for the DHCP server, for example the circuit ID. The circuit ID describes the origin of the DHCP query, for example which port received the DHCP query. You specify the DHCP server on the "General" tab.

Description of the displayed boxes

The page contains the following boxes:

Global configuration

- **Circuit ID router index**
  Enable or disable the setting. When enabled, the router index is added to the generated circuit ID.
- **Circuit ID recipient VLAN ID**
  Enable or disable the setting. When enabled, the VLAN ID is added to the generated circuit ID.
- **Circuit ID receiving port**
  Enable or disable the setting. When enabled, the receiving port is added to the generated circuit ID.

  Note

  You need to select a least one option.

  You will find further information on the router index (Circuit ID Router Index) and port index (Circuit ID Receive Port) in the IfTable using SNMP.

- **Remote ID**
  Shows the device ID.

Interface-specific configuration

- **Interface**
  Select the required IPv4 interface.

The table has the following columns:

- **Interface**
  Shows the interface.

  Note

  If you have not created an interface-specific configuration, the global configuration with the MAC address is used as the device ID.

- **Remote ID type**
  Select the type of the device ID. You have the following options:
  - IP Address
    The IPv4 address of the device is used as the device ID.

    Note

    No automatic updating

    There is no link between the "Remote ID" box and the currently set IP address.

    If the IP address is changed, the new IP address is not entered in the "Remote ID" box automatically. Only the value of the "Remote ID Type" box changes to "Free Text". To use the current IP address again, select "IP Address" again in the "Remote ID Type" box.

  - MAC Address
    The MAC address of the device is used as the device ID.
  - Free Text
    If you use "Free Text", you can enter the device name as the device identifier in "Remote ID".
- **Remote ID**
  Enter the device name. Can only be edited if you select the entry "Free Text" for "Remote ID Type".
- **Circuit ID Type**
  Select the type of circuit ID from the drop-down list. You have the following options:
  - Predefined
    The circuit ID is created automatically based on the router index, VLAN ID or port.
  - Free Number
    If you use "Free Number", you can enter the ID for "Circuit ID".
- **Circuit ID**
  Enter the circuit ID. The box can only be edited if you select the "Free Number" entry for the "Circuit ID Type".

VRRP

Router

Using the "Create" button, you can create new virtual routers. A maximum of 52 virtual routers can be configured. You can configure other parameters on the "Configuration" tab.

Note

- VRRP is available only on layer 3.
- Select "VRRP" to configure VRRP
- Simultaneous operation of VRRP and VRRPv3 is not possible.
- VRRP can only be used in conjunction with VLAN interfaces. Router ports are not supported.

Description of the displayed boxes

The page contains the following boxes:

- **VRRP**
  Enable or disable routing using VRRP.
- **Reply to pings on virtual interfaces**
  When enabled, the virtual IPv4 addresses also reply to the ping.
- **Track**
  Enable or disable VRID tracking. When enabled, all VRRP instances are monitored. If the status of a VRRP instance changes to "Initialize", the priority of all VRRP instances is reduced to the value "1". If the status of a VRRP instance changes, the original priority of all VRRP instances is restored.
- **Interface**
  Select the required VLAN interface operating as virtual router.
- **VRID**
  Enter the ID of the virtual router in the input box. This ID defines the group of routers that form a virtual router (VR). In the group, this is the same. it can no longer be used for other groups. Valid values are 1 ... 255.

The table has the following columns:

- **Interface**
  Shows the Interface that functions as the virtual router.
- **VRID**
  Shows the ID of the virtual router.
- **Virtual MAC Address**
  Shows the virtual MAC address of the virtual router.
- **Primary IP Address**
  Shows the numerically lowest IPv4 address in this VLAN. The entry 0.0.0.0 means that the "Primary" address on this VLAN is used. Otherwise all IPv4 addresses configured in this VLAN in "Layer 3 (IPv4) > Subnets" are valid values.

- **Router Status**
  Shows the current status of the virtual router. Possible values are as follows:
  - Master
    The router is the master router and handles the routing functionality for all assigned IP addresses.
  - Backup
    The router is the backup router. If the master router fails, the backup router takes over the tasks of the master router.
  - Initialize
    The virtual router has just been turned on. It will soon change to the "Master" or "Backup" status.
- **Master IP Address** (only available online)
  Shows the IPv4 address of the master router.
- **Priority**
  Shows the priority of the virtual router.
  If an IPv4 address is assigned to the VRRP router that is also actually configured on the local IPv4 interface, the value 255 is entered automatically. All other priorities can be distributed freely among the VRRP routers. The higher the priority, the earlier the VRRP router becomes "Master".
- **Advert. Interval**
  Shows the interval at which the master router sends VRRP packets.
- **Preempt**
  Shows the precedence of a router when changing roles between backup and master.
  - yes
    This router has precedence when changing roles.
  - no
    This router does not have precedence when changing roles.

VRRP and DHCP server

If you want to operate a DHCP server on the devices of a VRRP group, the DHCP server must be configured on the master router. Backup routers do not react to DHCP queries. Make sure that the master router is statically configured and that after a failure, becomes the master of the VRRP group again.

Configuration

Introduction

On this page, you configure the virtual router.

Note

VRRP is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- **Interface / VRID**
  Select the ID of the virtual router to be configured.
- **Primary IP Address**
  Select the numerically lowest IPv4 address: If the router becomes master router, the router uses this IPv4 address.

  Note

  If you only configure one subnet on this VLAN, no entry is necessary. The entry is then 0.0.0.0.
  If you configure more than one subnet on the VLAN and you want a specific IPv4 address to be used as the source address for VRRP packets, select the IPv4 address from the drop-down list. Otherwise, the numerically lowest IPv4 address will be used.

- **Master**
  If this option is enabled, the numerically lowest IPv4 address is entered for "Associated IP Address". This means that the highest priority IPv4 address of the VRRP router is used as the virtual IP address of the virtual master router. The backup routers in this group must disable the option and use the IPv4 address of the router in "Associated IP address".
- **Priority**
  Enter the priority of this virtual router.
  If an IPv4 address is assigned to the VRRP router that is also actually configured on the local IPv4 interface, the value 255 is entered automatically. All other priorities can be distributed freely among the VRRP routers. The higher the priority, the earlier the VRRP router becomes "Master".
- **Advertisement Interval**
  Enter the interval in seconds after which a master router sends a VRRP packet again.
- **Preempt lower priority Master**
  Allow the precedence when changing roles between backup and master based on the selection process.
- **Track ID**
  Select a track ID.
- **Decrement Priority**
  Enter the value by which priority of the VRRP interface will be reduced.
- **Current Priority**
  Shows the priority of the VRRP interface after the monitored interface has changed to the "down" status.

Address overview

Overview

This page shows which IPv4 addresses the virtual router monitors. Each virtual router can monitor a maximum of 10 IPv4 addresses.

Note

VRRP is available only on layer 3.

Description of the displayed boxes

The table has the following columns:

- **Interface**
  Shows the Interface that functions as the virtual router.
- **VRID**
  Shows the ID of this virtual router.
- **Number of Addresses**
  Shows the number of IPv4 addresses.
- **Associated IP Address (1) ... Associated IP Address (4)**
  Shows the router IPv4 addresses monitored by this virtual router. If a router takes over the role of master, the routing function is taken over by this router for all these IPv4 addresses.

Address configuration

Creating or changing the monitored IPv4 addresses

On this page, you can create, modify or delete the IPv4 addresses to be monitored. A maximum of 10 IPv4 addresses can be monitored by a virtual router.

Note

VRRP is available only on layer 3.

Description of the displayed values

The following settings are possible:

- **Interface / VRID**
  Select the required virtual router.
- **Associated IP Address**
  Enter the IPv4 address that the virtual router will monitor. A maximum of 10 IPv4 addresses are possible.

The table has the following columns:

- Associated IP Address
  Shows the IPv4 addresses that the virtual router monitors.

Interface Tracking

Introduction

On this page, you configure the monitoring of interfaces.

When the link of a monitored interface changes from "up" to "down", the priority of the assigned VRRP interface is reduced. You configure the value by which the priority is reduced on the page "Layer 3 > VRRP/VRRPv3 > Configuration".

When the link of the interface changes back from "down" to "up", the original priority of the VRRP interface is restored.

Note

This function is available only with layer 3.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  From the drop-down list, select the interface to be monitored.
- Track ID
  Enter a track ID.
- Track ID
  Select a track ID.
- Track Interface Count
  Enter how many monitored interfaces need to change to the "down" status, before the priority is changed.

The table has the following columns:

- Track ID
  Shows the track ID.
- Interface
  Shows the interface that is being monitored.

VRRPv3

Router (IPv4)

Introduction

On this page, you create new virtual routers. You can configure other parameters in "Layer 3 (IPv4) > VRRPv3 > Configuration".

Note

- This function is available only with layer 3.
- Simultaneous operation of VRRP and VRRPv3 is not possible.
- Select the "VRRPv3" check box to configure VRRPv3.
- You can use VRRPv3 on VLAN interfaces. Router ports are not supported.

Description of the displayed boxes

The page contains the following:

- **VRRPv3**
  Enable or disable routing using VRRPv3. Can only be enabled when "Layer 3 (IPv6) > Configuration" "IPv6 Routing" is enabled.
- **Reply to pings on virtual interfaces**
  When enabled, the virtual IPv4 addresses also reply to the ping.
- **Track**
  Enable or disable VRID tracking. When enabled, all VRRP instances are monitored. If the status of a VRRP instance changes to "Initialize", the priority of all VRRP instances is reduced to the value "1". If the status of a VRRP instance changes, the original priority of all VRRP instances is restored.
- **Interface**
  Select the required VLAN interface operating as virtual router.
- **VRID**
  Enter the ID of the virtual router. This ID defines the group of routers that form a virtual router (VR). In the group, this is the same. It can no longer be used for other groups. Valid values are 1 ... 255.

The table has the following columns:

- **Interface**
  Shows the IPv4 interface that functions as the virtual router.
- **VRID**
  Shows the ID of the virtual router.
- **Virtual MAC Address**
  Shows the virtual MAC address of the virtual router.

- Primary IP Address
  Shows the numerically lowest IPv4 address in this VLAN. The entry 0.0.0.0 means that the "Primary" address on this VLAN is used. Otherwise all IPv4 addresses configured on this VLAN in the "Layer 3 (IPv4) > Subnets" menu are valid values.
- Router State
  Shows the current status of the virtual router. Possible values are:
  - Master
    The router is the master router and handles the routing functionality for all assigned IPv4 addresses.
  - Backup
    The router is the backup router. If the master router fails, the backup router takes over the tasks of the master router.
  - Initialize
    The virtual router has just been turned on. It will soon change to the "Master" or "Backup" state.
- Master IP Address
  Shows the IPv4 address of the master router.
- Priority
  Shows the priority of the virtual router. Valid values are 1 ... 254. If an IPv4 address is assigned to the VRRP router that is also actually configured on the local IPv4 interface, the value 255 is entered automatically. All other priorities can be distributed freely among the VRRP routers. The higher the priority, the earlier the VRRP router becomes "Master".
- Advert. Interval
  Shows the interval at which the master router sends VRRPv3 packets.
- Preempt
  Shows the precedence of a router when changing roles between backup and master.
  - yes
    This router has precedence when changing roles.
  - no
    This router does not have precedence when changing roles.

VRRP and DHCP server

If you want to operate a DHCP server on the devices of a VRRP group, the DHCP server must be configured on the master router. Backup routers do not react to DHCP queries. Make sure that the master router is statically configured and that after a failure, becomes the master of the VRRP group again.

Configuration (IPv4)

Introduction

On this page, you configure the virtual router.

Note

This function is available only with layer 3.

Description of the displayed boxes

The page contains the following:

- **Interface / VRID**
  Select the ID of the virtual router to be configured.
- **Primary Address**
  Select the primary IPv4 address. If the router becomes master router, the router uses this IPv4 address.

  Note

  If you only configure one subnet on this VLAN, no entry is necessary. The entry is then 0.0.0.0.
  If you configure more than one subnet on the VLAN and you want a specific IPv4 address to be used as the source address for VRRP packets, select the IPv4 address. Otherwise, the numerically lowest IPv4 address will be used.

- **Master**
  If enabled, the numerically lowest IPv4 address is entered for "Associated IP Address". This means that the numerically lowest IPv4 address of the VRRPv3 router is used as the virtual IP address of the virtual master router. The backup routers in this group must disable the option and use the IPv4 address of the router in "Associated IP address".
- **Priority**
  Enter the priority of this virtual router. Valid values are 1-254.
  If an IPv4 address is assigned to the VRRPv3 router that is also actually configured on the local IPv4 interface, the value 255 is entered automatically. All other priorities can be distributed freely among the VRRPv3 routers. The higher the priority, the earlier the VRRPv3 router becomes "Master".
- **Advertisement interval**
  Enter the interval in seconds after which a master router sends a VRRPv3 packet again.
- **Preempt lower priority Master**.
  Allow precedence when changing roles between backup and master based on the selection process.
- **VRRP Compatible Mode**
  When enabled, the VRRPv3 router also sends and receives VRRPv2 frames in addition to VRRPv3 frames for configured IPv4 addresses. Only necessary when some VRRP routers do not support VRRPv3

- Track ID
  Select a track ID.
- Decrement Priority
  Enter the value by which the priority of the VRRPv3 interface will be reduced.
- Current Priority
  Shows the priority of the VRRPv3 interface after the monitored interface has changed to the "down" status.

Addresses Overview (IPv4)

Overview

This page shows which IPv4 addresses the virtual router monitors. Each virtual router can monitor a maximum of 10 IPv4 addresses.

Note

This function is available only with layer 3.

Description of the displayed boxes

The table has the following columns:

- Interface
  Shows the Interface that functions as the virtual router.
- VRID
  Shows the ID of this virtual router.
- Number of Addresses
  Shows the number of IPv4 addresses.
- Associated IP Address (1) ... Associated IP Address (10)
  Shows the router IPv4 addresses monitored by this virtual router. If a router takes over the role of master, the routing function is taken over by this router for all these IPv4 addresses.

Addresses Configuration (IPv4)

Creating or changing the monitored IP addresses

On this page, you can create, modify or delete the IPv4 addresses to be monitored.

Note

This function is available only with layer 3.

Description of the displayed boxes

The page contains the following:

- Interface / VRID
  Select the ID of the virtual router.
- Associated IP Address
  Enter the IPv4 address that the virtual router will monitor. A maximum of 10 IPv4 addresses are possible.

The table has the following columns:

- Associated IP Address
  Shows the IPv4 addresses that the virtual router monitors.

Interface Tracking (IPv4)

Introduction

On this page, you configure the monitoring of interfaces.

When the link of a monitored interface changes from "up" to "down", the priority of the assigned VRRP interface is reduced. You configure the value by which the priority is reduced on the page "Layer 3 > VRRP/VRRPv3 > Configuration".

When the link of the interface changes back from "down" to "up", the original priority of the VRRP interface is restored.

Note

This function is available only with layer 3.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  From the drop-down list, select the interface to be monitored.
- Track ID
  Enter a track ID.
- Track ID
  Select a track ID.
- Track Interface Count
  Enter how many monitored interfaces need to change to the "down" status, before the priority is changed.

The table has the following columns:

- Track ID
  Shows the track ID.
- Interface
  Shows the interface that is being monitored.

OSPFv2

Configuration

Introduction

On this page, you configure routing with OSPF.

Note

OSPF is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- OSPFv2
  Enable or disable routing using OSPFv2.
- Router ID
  Enter the name of one of the OSPFv2 interfaces. The name is entered in the IP address format and does not need to match the local IP address. The router ID must be unique in the network.
- OSPFv2 RFC1583 Compatibility
  Enable the option if you still have old OSPFv2 routers in operation that are not compatible with RFC 2328.
- Border Router
  Displays the status of the OSPFv2 router. If the local system is an active member in at least 2 areas, this is an area border router.
- AS Border Router
  Specify whether the router is an AS border router. An AS border router intercedes between multiple autonomous systems, for example if you have an additional RIP network. An AS border router is also necessary to add and to distribute static routes.
- New LSA Received (only available online)
  Shows the number of received LSAs. Updates and its own LSAs are not counted.
- New LSA Configured (only available online)
  Number of different LSAs sent by this local system.

- External LSA Maximum
  To limit the number of entries of external LSAs in the database, enter the maximum number of external LSAs.
- Exit Interval [s]
  Enter the interval after which the OSPFv2 router once again attempts to come out of the overflow status. A 0 means that the OSPFv2 router attempts to exit the overflow status only following a restart.
- Inbound Filter
  Select a route map that filters inbound routes.
- Redistribute Routes
  Specify which known routes are distributed using OSPFv2. You can make different decisions for the route types Default, Connected and Static.

  Note

  The settings can only be enabled on an AS border router. Enabling the Default and Static options, in particular, can cause problems if they are enabled at too many points in the network, for example, forwarding loops.

- Route Map
  Select a route map that filters which routes are forwarded using OSPFv2.
- Protocol Preferences
  - Default Distance
    Set the administrative distance for the OSPFv2 protocol.
- Route Map Preferences
  - Distance Route Map
    Select a route map from the drop-down list for which you want to specify the administrative distance.
  - Distance
    Configure the administrative distance for the previously selected route map.

Areas

Overview

An autonomous system can be divided into smaller areas.

On this page, you can view, create, modify or delete the areas of the router.

Note

OSPF is available only on layer 3.

Description of the displayed values

- **Area ID**
  Enter the designation of the area. The database is synchronized for all routers of an area. The area ID must be unique in the network. The area identifier is a 32-bit number with the following format: x.x.x.x where x = 0 ... 255
  The area ID 0.0.0.0 is reserved for the backbone area and cannot be deleted.

This table contains the following columns:

- **Area ID**
  Shows the ID of the area.
- **Area Type**
  Select the area type from the drop-down list.
  - Normal
  - Stub
  - NSSA
  - Backbone
- **Summary**
  Specify whether summary LSAs are generated for this area.
  - Summary: Summary LSAs are sent to the area.
  - No Summary: Summary LSAs are not sent to the area.
- **Metric**
  Displays the costs for the OSPFv2 interface.
- **Updates**
  Shows the number of recalculations of the routing tables.
- **LSA Count**
  Shows the number of LSAs in the database.
- **Area BR**
  Shows the number of reachable area border routers (ABR) within this area.
- **AS BR**
  Shows the number of reachable autonomous system border routers (ASBR) in this area.

Area range

Creating a new OSPFv2 area range

On this page, networks can be grouped together under one area ID. The method is used only with area border routers. This means that an area border router only advertises one route for grouped areas to the outside.

Note

OSPF is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- **Area ID**
  Select the area ID of the area. You configure the ID in "Layer 3 > OSPFv2 > Areas".
- **Subnet Address**
  Enter the IPv4 address of the network that will be grouped.
- **Subnet Mask**
  Enter the subnet mask of the network that will be grouped.

This table contains the following columns:

- **Area ID**
  Shows the ID of the area.
- **Subnet Address**
  Shows the IPv4 address of the network that will be grouped with other networks.
- **Subnet Mask**
  Shows the subnet mask of the network that will be grouped with other networks.
- **Advertise**
  Enable this setting to advertise the grouped network.

Interfaces

Overview

On this page, you can configure OSPFv2 interfaces.

Note

OSPF is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- **IP Address**
  Select the IPv4 address of the OSPFv2 interface.
- **Area ID**
  Select the area ID of the area with which the OSPFv2 interface is connected.

  Note

  For the secondary interface select the same Area ID as for the corresponding primary interface.
  The information whether a primary or secondary interface is involved can be found in the "Address Type" column on the "Layer 3 (IPv4) > Subnets > Overview" page.

This table contains the following columns:

- **IP Address**
  Shows the IPv4 address of the OSPFv2 interface
- **Address Type**
  There are two address types:
  - Primary
  - Secondary

  Cells for secondary addresses are grayed out and show the values of the associated primary address.
- **Area ID**
  Select the area ID of the area with which the OSPFv2 interface is connected.
- **Passive Interface**
  Specify the behavior of the interface:
  - Enabled
    No OSPFv2 information (e.g. Hello frames and LSDB updates) is sent via this interface and learned.
  - Disabled
    OSPFv2 information is sent via this interface and learned.
- **Metric**
  Enter the costs for the OSPFv2 interface.
- **Priority**
  Enter the router priority. The priority is only relevant for selecting the designated router or designated border router. This parameter can be selected differently on routers within the same subnet.
  Range of values: 0 to 255
  Default setting: 1
- **Trans. Delay**
  Enter the required delay when sending a connection update.
  Range of values: 1 s to 3600 s
  Default: 1 s
- **Retrans. Delay**
  Enter the time after which an OSPFv2 packet is transferred again if no confirmation was received.
  Range of values: 1s to 3600 s
  Default: 5 s
- **Hello Interval**
  Enter the interval between two Hello packets.
  Range of values: 1 s to 65.535 s
  Default: 10 s
- **Dead interval**
  Enter the interval after which the neighbor router is marked as "failed" if no more Hello packets are received from it during this time.
  Default: 40 s

Interface authentication

Configuring the interface login

On this page, you define the authentication of the interface.

Note

OSPF is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- **OSPF Interface**
  Select the OSPFv2 interface for which you want to configure authentication.
- **Authentication type**
  Select the authentication method of the OSPF interface. You have the following options:
  - none:
    No authentication
  - simple:
    Authentication using an unencrypted password
  - MD5:
    Authentication using MD5

Section "Simple Authentication"

- **Password**
  Enter a password.
- **Confirmation**
  Confirm the entered password.

Section "MD5 Authentication"

- **Authentication Key ID**
  Enter the identifier of the MD5 authentication key. Enter the ID for MD5 authentication with which the password will be used as a key. Since the key ID is transferred with the protocol, the same key must be stored under the same key ID on all neighboring routers.

The table has the following columns:

- **Authentication Key ID**
  Can only be edited if you set the MD5 authentication method. It is only possible to use several keys there.
- **MD5 Key**
  Enter the MD5 key.

- **Confirm MD5 Key**
  Confirm the entered key.
- **Youngest Key ID** (only available online)
  Shows whether or not the MD5 key is the latest key ID.

Virtual links

Overview

Due to the protocol, each area border router must have access to the backbone area. If a router is not connected directly to the backbone area, a virtual link to it is created.

Note

OSPF is available only on layer 3.

Note

Note that when creating a virtual link both the transit area and the backbone area must already be configured.

the virtual link must be configured identically at both ends.

Description of the displayed boxes

The page contains the following note:

- **Virtual links are not effective unless the device is an ABR.** (only available online)
  This is displayed when at least one virtual link entry is configured and the device is not an area border router.

The page contains the following boxes:

- **Neighbor Router ID**
  Enter the ID of the neighbor router at the other end of the virtual connection.
- **Transit Area ID**
  Select the ID of the area that connects the two routers.

This table contains the following columns:

- **Transit Area ID**
  Shows the ID via which the two routers are connected.
- **Neighbor Router ID**
  Shows the ID of the neighbor router at the other end of the virtual connection.
- **Virt. Link Status**
  Specify the status of the virtual link. The following statuses are possible:
  - down: The virtual link is inactive.
  - point-to-point: The virtual link is active.

- **Trans. Delay**
  Enter the required delay when sending a link update packet.
  Range of values: 1 s to 3600 s
  Default setting: 1 s
- **Retrans. Delay**
  Enter the time after which a packet is transferred again if no confirmation was received.
  Range of values: 1 s to 3600 s
  Default setting: 5 s
- **Hello Interval**
  Enter the interval between two Hello packets.
  Range of values: 1 s to 65,535 s
  Default setting: 10 s
- **Dead Interval**
  Enter the interval after which the neighbor router counts as "failed" if no more Hello packets are received from it during this time.
  Default setting: 40 s

Virtual Link Authentication

Configuring the interface login

On this page, you define the authentication of the interface.

Note

OSPF is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- **Virtual Link (Area/Neighbor)**
  Select the virtual link for which you want to configure authentication.
- **Authentication type**
  Select the authentication method of the OSPF interface. You have the following options:
  - none:
    No authentication
  - simple:
    Authentication using an unencrypted password.
  - MD5
    Authentication using MD5

Section "Simple Authentication"

- Password
  Enter the password.
- Confirmation
  Confirm the entered password.

Section "MD5 Authentication"

- Authentication Key ID
  Enter the identifier of the MD5 authentication key. Enter the ID for MD5 authentication with which the password will be used as a key.
  Since the key ID is transferred with the protocol, the same key must be stored under the same key ID on all neighboring routers.

The table has the following columns:

- Authentication Key ID
  Can only be edited if you set the MD5 authentication method. It is only possible to use several keys there.
- MD5 Key
  Enter the MD5 key.
- Confirm MD5 Key
  Confirm the entered key.
- Youngest Key ID
  Shows whether or not the MD5 key is the latest key ID.

RIPv2

Configuration

On this page, you configure routing with RIP.

Note

RIPv2 is available only on layer 3.

Description of the displayed boxes

- RIPv2
  Enable or disable routing using RIPv2.
- Inbound Filter
  Select a route map that filters inbound routes.

- **Redistribute Routes**
  Specify which known routes are distributed using RIPv2.
  The following types of route exist:
  - Static Default
  - Connected
  - Static
  - OSPF
- **Route Map**
  Select a route map that filters which routes are forwarded using RIPv2.
- **Protocol Preferences**
  - **Default Distance**
    Set the administrative distance for the OSPFv2 protocol.
- **Route Map Preferences**
  - **Distance Route Map**
    Select a route map from the drop-down list for which you want to specify the administrative distance.
  - **Distance**
    Configure the administrative distance for the previously selected route map.

Interfaces

Overview

On this page, you can configure RIPv2 interfaces.

Note

RIPv2 is available only on layer 3.

Description of the displayed boxes

- **IP Address**
  Select the IPv4 address of the RIPv2 interface.

This table contains the following columns:

- **IP Address**
  Shows the IPv4 address of the RIPv2 interface
- **Send Updates**
  Select the way in which updates are sent:
  - no send
    No updates are sent.
  - RIPv1
    Updates for RIPv1 are sent.
  - RIPv1-compat.
    RIPv2 updates are sent as broadcasts according to the rules of RIPv1.
  - RIPv2
    Updates for RIPv2 are sent as multicasts.
  - RIPv1 demand/RIPv2 demand
    RIP packets are sent only as a response to an explicit query.
- **Receive Updates**
  Select the form in which received updates are accepted:
  - no receive
    No updates are received.
  - RIPv1
    Only updates of RIPv1 are received.
  - RIPv2
    Only updates of RIPv2 are received.
  - RIPv1/v2
    Updates of RIPv1 and RIPv2 are received.
- **Default Metric**
  Enter the costs for the RIPv2 interface.

IGMP

IGMP

On this page, you configure IGMP (Internet Group Management Protocol). IGMP is a network protocol used for IP multicast. When IP muticasting IP packets with one IP address are distributed to multiple clients at the same time. IGMP can manage dynamic and static multicast groups.

Description of the displayed boxes

The page contains the following boxes:

- **IGMP**
  Enable or disable IGMP (Internet Group Management Protocol) for the entire device.
- **Interface**
  Select the required interface on which you want to configure IGMP.

The table has the following columns:

- **Interface**
  Shows the interface.
- **Version**
  Select the version of IGMP.
- **Query Interval [s]**
  Define the interval at which the multicast router sends IGMP queries.
  Specified in seconds
- **Max. Response Time [1/10 s]**
  Specify the period of time in an IGMP query that the client has to respond to a query.
  Specified in tenths of seconds
- **Explicit Tracking**
  Enable or disable the "Explicit Tracking" function for the interface.
  If the "Explicit Tracking" function is enabled, the multicast router tracks precisely which clients log on to a multicast group (Join) and log off (Leave). As soon as the last client has logged off from a multicast group, the multicast router deletes the multicast group. Since the multicast router has precise information about the logged on and logged off clients, the multicast router does not need to query whether a further client wants to receive the multicast.
- **Admin Status**
  Enable or disable IGMP on the interface.
- **Operating Status**
  Shows the operating status of the interface, see also "System > Ports > Overview".
  - Up
  - Down

PIM

Configuration

On this page, you configure PIM.

Note

PIM is available only on layer 3.

Description of the displayed boxes

- **PIM Routing**
  Enable or disable PIM globally on the device.
- **Static Rendezvous Point (RP)**
  Enable or disable depending on whether the device should operate as a static rendezvous point.
- **Bidirectional Multicast**
  Enable or disable the "bidirectional multicast" functionality globally on the device.

Interface

Overview

On this page, you configure PIM interfaces.

Note

PIM is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- **Interface**
  Select a VLAN IP interface.
- **PIM**
  If you select this check box, The selected interface is assigned PIM.
- **Bootstrap Router (BSR) Candidate**
  If you enable this check box, you specify that the interface will become a candidate for the bootstrap router. A BSR coordinates the rendezvous points in a PIM network.
- **BSR Candidate Value**
  Enter the priority of the interface as BSR candidate.
  Which candidate becomes the BSR is decided based on the priority. The candidate with the highest priority becomes the BSR.
  If the same priority is set for two candidates, when necessary the decision is made based on the IP address. The interface with the higher IP address becomes BSR.
  A BSR coordinates the rendezvous points in a PIM network.

- **Designated Router (DR) Priority**
  Enter the DR priority of the interface.
  Which candidate becomes the DR is decided based on the DR priority. The candidate with the highest priority becomes the DR.
  If the same priority is set for two candidates, when necessary the decision is made based on the IP address. The interface with the higher IP address becomes DR.
  The DR forwards the IGMP Joins to the rendezvous point.
- **BSR Border**
  When enabled there is no BSR communication via this interface. As a result a PIM network is divided into PIM domains.

RP Static

Overview

On this page you specify which interfaces will become static rendezvous points.

Configure the static RP on all devices of the PIM component.

Note

PIM is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- **Group Address**
  Enter the address of the multicast group for which the interface will become RP.
- **Group Mask**
  Enter the subnet mask that restricts the multicast band.
- **RP Address**
  Enter the IP address of the interface that will become RP.
- **Bidirectional Multicast**
  Enable or disable the "bidirectional multicast" functionality for a multicast group.
  Configure the functionality for a multicast group the same on all RPs.
  If you enable the function for an RP address, the function is enabled for all existing and future entries of this RP address.

This table contains the following columns:

- **Select**
  Select the row you want to delete.
- **Group Address**
  Shows the address of the multicast group.
- **Group Mask**
  Shows the subnet mask.

- RP Address
  Shows the IP address of the RP.
- Bidirectional Multicast
  Shows whether the "bidirectional multicast" functionality is activated or deactivated for a multicast group.

RP Candidate

Overview

On this page you specify which interfaces will be candidates for the rendezvous point (RP).

Within a PIM network you can configure several candidates for the RP. The BSR coordinates the candidates and decides on the RP.

Note

PIM is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- Group Address
  Enter the address of the multicast group.
- Group Mask
  Enter the subnet mask that restricts the multicast band.
- RP Interface
  Select the interface that will become the RP candidate.
- Bidirectional Multicast
  Enable or disable the "bidirectional multicast" functionality for a multicast group.
  Configure the functionality for a multicast group the same on all RPs.
  If you enable the function for an RP interface, the function is enabled for all existing and future entries of this RP interface.

This table contains the following columns:

- Select
  Select the row you want to delete.
- Group Address
  Shows the address of the multicast group.
- Group Mask
  Shows the subnet mask.
- RP Candidate
  Shows the IP address of the RP.

- **Bidirectional Multicast**
  Shows whether the "bidirectional multicast" functionality is activated or deactivated for a multicast group.
- **RP Priority**
  Enter the priority of the interface.
  Which candidate becomes the rendezvous point is decided based on the RP priority. The candidate with the highest priority becomes the RP. The lower the value, the higher the priority.
  If the same priority is set for two candidates, when necessary the decision is made based on the IP address. The interface with the lowest IP address becomes RP.

MSDP

Configuration

Overview

On this page, you can configure MSDP.

MSDP (Multicast Source Discovery Protocol) is supported within a PIM-SM domain in combination with anycast RPs. Anycast RPs involve a redundancy concept for RPs. The RPs involved all receive the same IP address and are made known in the network. The RPs are linked together via MSDP sessions. The RPs synchronize their multicast sources via source active messages.

Configure MSDP on the rendezvous points.

You can only use MSDP when the "bidirectional multicast" functionality is disabled in the PIM network.

Note

MSDP is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- **MSDP**
  Enable or disable MSDP.
- **Originator ID**
  Select an interface for identification of the MSDP node. The Originator ID must be unique in the network.

- Cache Life Time
  Enter the time after which an entry with information of another RP is deleted from the SA cache if no SA cache updates are received. After an entry is created or updated, a timer starts and runs backwards with the defined time. SA cache updates are received cyclically and reset the timer of the entry. When the timer expires without an SA cache update being received in this time. the corresponding entry is deleted.
  If the value "0" is set, no SA cache entries are saved.
- Listener Port
  TCP port via which the device communicates with other MSDP nodes. If you change the listener port, only new MSDP connections are affected that are created after this change. Existing MSDP connections continue to use the previous listener port.

Peer

Overview

On this page, you configure MSDP partners.

Note

MSDP is available only on layer 3.

Description of the displayed boxes

The page contains the following boxes:

- Peer Address
  Enter the IP address of the MSDP partner to which the device will establish a connection.
- Connected Interface
  Select the interface via which the device communicates with the MSDP partner.

This table contains the following columns:

- Select
  Select the row you want to delete.
- Peer Address
  Shows the address of the MSDP partner.
- Local Address
  Shows the IP address of the interface via which the device communicates with the MSDP partner.
- Admin Status
  Enable or disable the connection.

- Connection Status
  Shows the status of the connection.
  - Listening
    The device is waiting for the connection.
  - Connecting
    Connection establishment
  - Established
    The connection is established.
  - Inactive
    The connection is not active.
  - Disabled
    The connection is disabled.
- Keepalive Interval[s]
  Define the interval at which the device sends keepalive messages to the MSDP partners.
- Peer Hold Time Interval[s]
  Define the time after which an MSDP connection is terminated if no keepalive messages are received from the MSDP partner.
- Connect Retry Interval[s]
  Define the interval after which an attempt is made to establish a connection. If there is no MSDP connection to a partner, when the Connect Retry Interval elapses, an attempt is made to establish a connection.
- TTL Threshold
  Define the maximum number of routers that may be passed through on the way to the MSDP partner (hop count). If the specified value used is exceeded, the frame is discarded.

Configuring layer 3 functions for IPv6

Configuration

Introduction

The page contains the overview of the layer 3 functions for IPv6. On this page, you enable or disable the required layer 3 function.

The functions "IPv6 Routing", "VRRPv3" and "OSPFv3" are only available with Layer 3 (IPv6).

Description of the displayed boxes

- **IPv6 Routing** (only available with devices with a layer 3 license)
  - Enabled
    IPv6 routing is enabled. If IPv4 routing is not enabled on the device this is also enabled with this function.
  - Disabled
    IPv6 routing is disabled.
- **VRRPv3** (only available with devices with a layer 3 license)
  Enable or disable routing using VRRPv3. To use VRRPv3 , first enable the IPv6 routing function. You can configure other settings in "Layer 3 (IPv6) > VRRPv3".
- **OSPFv3** (only available with devices with a layer 3 license)
  Enable or disable routing using OSPFv3. You can configure other settings in "Layer 3 (IPv6) > OSPFv3".

Subnets (IPv6)

Connected Subnets

On this page, you can enable IPv6 on the interface. The interface is also called an IPv6 interface. An IPv6 interface can have several IPv6 addresses.

Description of the displayed boxes

The page contains the following:

- **Interface**
  Select the IP interface for which IPv6 will be enabled.
- **IPv6 Enable**
  Enable or disable IPv6 for this interface.

  Note

  Disabling IPv6

  As soon as IPv6 is enabled on the IP interface, IPv6 can only be disabled by removing the interface.

- **IPv6 Address**
  Enter the IPv6 address. The entry depends on the selected address type.
- **Prefix Length**
  Enter the number of left-hand bits belonging to the prefix

- IPv6 Address Type
  Select the address type.
  - Unicast
  - Anycast
  - Link Local IPv6 address is only valid on the link
- Address Autoconfiguration (SLAAC)
  Enable or disable the SLAAC (Stateless Address Auto Configuration) mechanism for the address configuration.
  If SLAAC is enabled, there will be stateless auto configuration via NDP (Neighbor Discovery Protocol).
  If the option is disabled, the address configuration is status dependent. The device obtains the IPv6 address and the configuration file from a DHCPv6 server.

The table has the following columns:

- Interface Name
  Shows the name of the interface.
- IPv6 Address
  Shows the IPv6 address of the subnet.
- Prefix Length
  Shows the prefix length.
- IPv6 Address Type
  Displays the address type. The following values are possible:
  - Unicast
  - Anycast
  - Link Local

- **Loopback**
  Shows whether or not the "Loopback" property is enabled. You configure the "loopback" property on the page "Layer 3 (IPv4) > Subnets > Overview".
- **Duplicate Address Detection Status**
  In Address Autoconfiguration (SLAAC), the "Duplicate Address Detection Status" function prevents IPv6 addresses from being assigned twice. The device can only use free IPv6 addresses during autoconfiguration.
  When the function is enabled, the check via NDP takes place automatically.

  Note

  The function does not run a cyclic check.

  This column shows the current status of the function. The following values are possible:

  – Tentative
    This status indicates that the selected IPv6 address is being checked. The device sends a neighbor solicitation message to the selected IPv6 address.
  – Conflict
    This status indicates that the IPv6 address is already being used. In this case, a neighbor advertisement message with the selected IPv6 address is returned to the device. The device forms a new IPv6 address and checks it again.
  – Complete
    This status indicates that the selected IPv6 address can be used. In this case, the device did not receive feedback within a period of time and assumes that the IPv6 address is not yet assigned.
  – Down
    This status indicates that the interface is not active. No check is carried out.

Static Routes (IPv6)

On this page, you configure static IPv6 routes.

Description of the displayed boxes

The page contains the following:

- **Destination Network**
  Enter the network address of the destination that can be reached via this route.
- **Prefix Length**
  Enter the number of left-hand bits belonging to the prefix
- **Gateway**
  Enter the IPv6 address of the gateway via which this network address is reachable.

- Metric
  Enter the metric for the route. The metric corresponds to the quality of a connection, based for example on speed or costs. If there are several equal routes, the route with the lowest value is used.
  Range of values: 1 - 254
- Interface
  Specify the IPv6 interface via which the network address of the destination is reached.

This table contains the following columns:

- Destination Network
  Shows the IPv6 network address of the destination.
- Prefix Length
  Shows the prefix length.
- Gateway
  Shows the IPv6 address of the next gateway.
- Interface
  Shows the Interface of the route.
- Metric
  Enter the metric for the route. When creating the route, "not used" is entered automatically. The metric corresponds to the quality of a connection, based for example on speed or costs. If there are several equal routes, the route with the lowest value is used.
  Range of values: 1 - 254
- Status
  Shows whether or not the route is active.

Route maps

General (IPv6)

Route maps

With route maps, you control how routing information is further processed. You can filter routing information and specify whether the information is further processed, modified or discarded.

Route maps operate according to the following principle:

- Routing information is compared with the filters of the route maps.
- The comparison is continued until the filters of a route map match the properties of an item of information.
- The information is then processed according to the route map settings:
  - The routing information is discarded.
  - The properties of the routing information are changed.

Description of the displayed boxes

- **Name**
  Enter the name for the route map.
- **Sequence No**
  Enter a number for the route map.
  You can create several route maps with the same name but with different sequence numbers. The sequence numbers then specify the order in which the route maps are processed.

The table has the following columns:

- **Name**
  Shows the name of the route map.
- **Sequence No**
  Shows the sequence number of the route map.
- **Action**
  Specify what happens to the routing information that matches the settings of the route map:
  - permit
    The routing information is further processed according to the settings you make on the "Set" page.
  - deny
    The routing information is discarded.

Interface & Value Match (IPv6)

On this page, you specify whether or not the routing information for a route map is filtered according to interfaces, metric or tags.

Description of the displayed boxes

- **Route Map (name/seq. no.)**
  Select a route map.
  The created route maps are available to you.
- **Type**
  Select the basis for the filtering:
  - Interface
  - Metric
  - Tag
  - Route Type
  - metric type
- **Interface**
  Select an interface.
  Only active if you have selected the "Interface" entry for "Type".

- **Metric**
  Enter a value for the metric.
  This box is only active if you have selected the "Metric" entry for "Type".
- **Tag**
  Enter a value for the tag.
  This box is only active if you have selected the "Tag" entry for "Type".
- **Route Type**
  Select the type of the route.
  - Local
    The routing information for the route map is filtered according to directly connected routes (local interfaces).
  - Remote
    The routing information for the route map is filtered according to learned or statically configured routes.

  This box is active only if i you have selected the entry "Route Type in the "Type" drop-down list.
- metric type
  Select the type of the metric.
  - Inter-Area
    The routing information for the route map is filtered according to routes learned from other areas.
  - Intra-Area
    The routing information for the route map is filtered according to routes originating from the same area.
  - type-1-external
    The routing information for the route map is filtered according to routes whose path costs (metric) are made up of the external path costs and the path costs to the ASBR.
  - type-2-external
    The routing information for the route map is filtered according to routes with only external path costs (metric).

  This box is active only if you have selected the entry "Metric Type" in the "Type" drop-down list.

The table has the following columns:

- **Type**
  Shows the selected type:
  - Interface
  - Metric
  - Tag
  - Route Type
  - metric type
- **Value**
  Shows the selected interface, the value of the metric or of the tag.

Destination Match (IPv6)

On this page, you specify whether or not the routing information for a route map is filtered based on the destination IPv6 address.

Description of the displayed boxes

- Route Map (Name/Seq. No.)
  Select a route map.
- IP Address
  Enter the IPv6 address of the destination on which the filtering is based.
- Prefix Length
  Enter the number of left-hand bits belonging to the prefix

The table has the following columns:

- IP Address
  Shows the IPv6 address of the destination.
- Prefix Length
  Shows the prefix length.

Next Hop Match (IPv6)

On this page, you specify whether or not the filtering for a route map will be based on the router to which the routing information is sent next.

Description of the displayed boxes

- Route Map (name/seq. no.)
  Select a route map.
- IP Address
  Enter the IP address of the router to which the routing information will be sent next.

The table has the following columns:

- IP Address
  Shows the IP address of the next router.

Set (IPv6)

On this page, you specify whether or not the routing information will be changed by a route map.

If, for example, you have filtered based on a certain metric, you can change the value of the metric here. The routing information is then forwarded with the new value. You can only change the information of a "Permit" route map.

Description of the displayed boxes

- Route Map (name/seq. no.)
  Select a route map.

The table has the following columns:

- Name
  Shows the name of the route map.
- Sequence No
  Shows the sequence number of the route map.
- Metric
  Enter the new value for the metric with which the routing information will be forwarded.
- Tag
  Enter the new value for the tag with which the routing information will be forwarded.

DHCPv6 Relay Agent

Interfaces

A DHCPv6 relay agent forwards the DHCP queries of the DHCPv6 client to the DHCPv6 server. When forwarding the remote ID is inserted in the DHCP query. The remote ID is the identifier of the interface. With this, the DHCPv6 agent can send the reply of the DHCPv6 server to the required DHCPv6 client.

Description of the displayed boxes

- Remote ID (Opt. 37)
  When enabled the DHCP relay agent option 37 (Remote ID) is used. Specify the type of identifier in "Remote ID Type".
- Interface
  Select the IPv6 interface.

The table has the following columns:

- Interface
  Interface to which the settings relate.
- Current Remote ID
  Shows which remote ID is currently being used.
- Remote ID Type
  Select hat is used as the identifier for the IPv6 interface. The following options are available:
  - DUID
    DHCP Unique Identifier
  - System Name
    System name of the device
  - Management IP
    Management IPv6 address of the device
  - User Defined
    User-defined entry
- Router Index Suffix
  With selected check box, the Router Index Suffix is appended to the remote ID.

- VLAN ID Suffix
  With selected check box, the VLAN ID Suffix is appended to the remote ID.
- Receive-Port Suffix
  With selected check box, the Receive-Port Suffix is appended to the remote ID.
- DHCP Unique Identifier
  Enter the DUID (DHCP Unique Identifier).
- User Defined
  Enter a user-defined entry as the identifier.

Server Addresses

On this page you enable the DHCPv6 relay agent on the interface. You also specify which DHCPv6 server can be reached via it.

Description of the displayed boxes

- Relay Interface
  Select the interface on which the DHCPv6 relay agent accepts DHCPv6 requests.
- Server IP Address
  Enter the IPv6 address of the DHCPv6 server to which the DHCPv6 requests will be forwarded. The address can be either the unicast or the link local multicast address. With the link local multicast address you need to specify an interface, via which the DHCPv6 server can be reached. You configure the interface in "Layer 3 (IPv6) > DHCPv6 Relay Agent > Outgoing Interfaces".

The table has the following columns:

- Server IP Address
  Shows the IPv6 address of the DHCPv6 server.

Outgoing Interfaces

Outgoing Interfaces

The link local multicast address (ff02::1:2) of the DHCPv6 server can only be reached by devices located on the same link. On this page, you specify the IPv6 interface via which this DHCPv6 server can be reached.

Description of the displayed boxes

- Relay Interface
  Select the IPv6 interface on which the DHCPv6 relay agent accepts DHCPv6 requests.
- Server IP Address
  Select the IPv6 address of the DHCPv6 server.
- Outgoing Interface
  Select the interface via which the DHCPv6 server can be reached.

The table has the following columns:

- Server IP Address
  Shows the IPv6 address of the DHCPv6 server.
- Outgoing Interface
  Shows the IPv6 interface via which the DHCPv6 server can be reached.

VRRPv3

Router (IPv6)

Introduction

On this page, you create new virtual routers. A maximum of 52 virtual routers can be configured. You can configure other parameters in "Layer 3 (IPv6) > VRRPv3 > Configuration".

Note

- This function is available only with layer 3.
- Simultaneous operation of VRRP and VRRPv3 is not possible.
- Select the "VRRPv3" check box to configure VRRPv3.
- You can use VRRPv3 on VLAN interfaces. Router ports are not supported.

Description of the displayed boxes

- VRRPv3
  Enable or disable routing using VRRPv3. Can only be enabled when "Layer 3 (IPv6) > Configuration" "IPv6 Routing" is enabled.
- Track
  Enable or disable VRID tracking.
  When enabled, all interfaces of a VRID are monitored. When the link of an interface changes from "up" to "down", the priority of all VRRP interfaces with the same VRID is reduced to the value "0".
  When the link of an interface changes back from "down" to "up", the original priority of the VRRP interfaces is restored.
- Interface
  Select the required VLAN interface operating as virtual router.
- VRID
  Enter the ID of the virtual router. This ID defines the group of routers that form a virtual router (VR). In the group, this is the same. It can no longer be used for other groups. Valid values are 1 ... 255.

The table has the following columns:

- Interface
  Shows the Interface that functions as the virtual router.
- VRID
  Shows the ID of the virtual router.
- Virtual MAC Address (only available online)
  Shows the virtual MAC address of the virtual router.
- Primary IP Address
  Shows the numerically lowest IPv6 address in this VLAN. The entry "::" means that the "Primary" address is used on this VLAN. Otherwise all IPv6 addresses configured on this VLAN in the "Layer 3 (IPv6) > Subnets" menu are valid values.
- Router State (only available online)
  Shows the current status of the virtual router. Possible values are:
  - Master
    The router is the master router and handles the routing functionality for all assigned IPv4 addresses.
  - Backup
    The router is the backup router. If the master router fails, the backup router takes over the tasks of the master router.
  - Initialize
    The virtual router has just been turned on. It will soon change to the "Master" or "Backup" state.
- Master IP Address (only available online)
  Shows the IPv6 address of the master router.
- Priority (only available online)
  Shows the priority of the virtual router. Valid values are 1 ... 254.
  The current master router is automatically given 255. All other priorities can be distributed freely among the VRRPv3 routers. The higher the priority, the earlier the VRRPv3 router becomes "Master".
- Advert. Internal
  Shows the interval at which the master router sends VRRPv3 packets.
- Preempt
  Shows the precedence of a router when changing roles between backup and master.
  - yes
    This router has precedence when changing roles.
  - no
    This router does not have precedence when changing roles.
- Accept Mode
  Shows the precedence of a router when changing roles between backup and master.
  - yes
    Virtual IP addresses also reply to pings.
  - no
    Virtual IP addresses do not reply to pings.

VRRP and DHCP server

If you want to operate a DHCP server on the devices of a VRRP group, the DHCP server must be configured on the master router. Backup routers do not react to DHCP queries. Make sure that the master router is statically configured and that after a failure, becomes the master of the VRRP group again.

Configuration (IPv6)

Introduction

On this page, you configure the virtual router.

Note

This function is available only with layer 3.

Description of the displayed boxes

- **Interface / VRID**
  Select the ID of the virtual router to be configured.
- **Primary Address**
  Select the primary IPv6 address. If the router becomes master router, the router uses this IPv4 address.

  Note

  If you only configure one subnet on this VLAN, no entry is necessary. The entry is then **::**. If you configure more than one subnet on the VLAN and you want a specific IPv6 address to be used as the source address for VRRPv3 packets, select the IPv6 address. Otherwise, the IPv6 address with priority will be used.

- **Master**
  If this option is enabled, the numerically lowest IPv6 address is entered for "Associated IP Address". This means that the highest priority IPv6 address of the VRRP router is used as the virtual IPv6 address of the virtual master router. The backup routers in this group must disable the option and use the IPv6 address of the router in "Associated IP Address".
- **Priority**
  Enter the priority of this virtual router. Valid values are 1 ... 254. The current master router is always given 255. All other priorities can be distributed freely among the redundant routers. The higher the priority, the earlier the router becomes "Master".
- **Advertisement interval**
  Enter the interval in seconds after which a master router sends a VRRPv3 packet again.
- **Preempt lower priority Master**.
  Allow precedence when changing roles between backup and master based on the selection process.
- **Reply to pings on virtual interfaces (Accept Mode)**
  When enabled, the virtual IPv6 addresses also reply to the ping.

- Track ID
  Select a track ID.
- Decrement Priority
  Enter the value by which priority of the VRRP interface will be reduced.
- Current Priority
  Shows the priority of the VRRP interface after the monitored interface has changed to the "down" status.

Addresses Overview (IPv6)

Overview

This page shows which IPv6 addresses the virtual router monitors. Each virtual router can monitor a maximum of 10 IPv6 addresses.

Note

This function is available only with layer 3.

Description of the displayed boxes

The table has the following columns:

- Interface
  Shows the Interface that functions as the virtual router.
- VRID
  Shows the ID of this virtual router.
- Number of Addresses
  Shows the number of IPv6 addresses.
- Associated IP Address (1) ... Associated IP Address (10)
  Shows the router IPv6 addresses monitored by this virtual router. If a router takes over the role of master, the routing function is taken over by this router for all these IPv6 addresses.

Addresses Configuration (IPv6)

Creating or changing the monitored IP addresses

On this page, you can create, modify or delete the IPv4 addresses to be monitored. A maximum of 10 IPv4 addresses can be monitored by a virtual router.

Note

This function is available only with layer 3.

Description of the displayed boxes

- Interface / VRID
  Select the ID of the virtual router.
- Associated IP Address
  Enter the IPv6 address that the virtual router will monitor. A maximum of 10 IPv6 addresses are possible.

The table has the following columns:

- Associated IP Address
  Shows the IPv6 addresses that the virtual router monitors.

Interface Tracking (IPv6)

Introduction

On this page, you configure the monitoring of interfaces.

When the link of a monitored interface changes from "up" to "down", the priority of the assigned VRRP interface is reduced. You configure the value by which the priority is reduced on the page "Layer 3 > VRRP/VRRPv3 > Configuration".

When the link of the interface changes back from "down" to "up", the original priority of the VRRP interface is restored.

Note

This function is available only with layer 3.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  From the drop-down list, select the interface to be monitored.
- Track ID
  Enter a track ID.
- Track ID
  Select a track ID.
- Track Interface Count
  Enter how many monitored interfaces need to change to the "down" status, before the priority is changed.

The table has the following columns:

- Track ID
  Shows the track ID.
- Interface
  Shows the interface that is being monitored.

OSPFv3

Configuration

Introduction

On this page, you configure the routing using OSPFv3.

Note

This function is available only with layer 3.

Description of the displayed boxes

The page contains the following boxes:

- **OSPFv3**
  Enable or disable routing using OSPFv3.
- **Router ID**
  Enter the name of one of the OSPFv3 interfaces. The router ID is specified in IPv4 format and must be unique in the network.
- **Border Router**
  Displays the status of the OSPFv3 router. If the local system is an active member in at least 2 areas, this is an area border router.
- **AS Border Router**
  Specify whether the router is an AS border router. An AS border router intercedes between multiple autonomous systems, for example if you have an additional RIP network. An AS border router is also necessary to add and to distribute static routes.
- **New LSA Received**
  Shows the number of received LSAs. Updates and its own LSAs are not counted.
- **New LSA Configured**
  Shows the number of different LSAs sent by this local system.
- **External LSA Maximum**
  To limit the number of entries of external LSAs in the database, enter the maximum number of external LSAs.
- **Exit Interval [s]**
  Enter the interval after which the OSPFv3 router once again attempts to come out of the overflow status. A 0 means that the OSPFv3 router attempts to exit the overflow status only following a restart.
- **Inbound Filter**
  Select a route map that filters inbound routes.

- Redistribute Routes
  Specify which known routes are distributed using OSPFv3. You make different decisions for the route types Connected, Static and RIPng.

  Note

  The options can only be enabled on an AS border router. Enabling the Default and Static options, in particular, can cause problems if they are enabled at too many points in the network, for example, forwarding loops.

- Route Map
  Select a route map that filters which routes are forwarded using RIPng.
- Protocol Preferences
  - Default Distance
    Set the administrative distance for the OSPFv2 protocol.
- Route Map Preferences
  - Distance Route Map
    Select a route map from the drop-down list for which you want to specify the administrative distance.
  - Distance
    Configure the administrative distance for the previously selected route map.

Areas

Overview

An autonomous system can be divided into smaller areas.

On this page, you can view, create, modify or delete the areas of the router.

Note

This function is available only with layer 3.

Description of the displayed boxes

The page contains the following:

- Area ID
  Enter the designation of the area. The database is synchronized for all routers of an area. The area identifier must be unique in the network. The area identifier is a 32-bit number with the following format: x.x.x.x where x = 0 ... 255
  The area identifier 0.0.0.0 is reserved for the backbone area and cannot be deleted.

This table contains the following columns:

- Area ID
  Shows the ID of the area.
- Area Type
  Specify the type of area.
  - Normal
  - Stub
  - NSSA
  - Backbone
- Summary
  Specify whether summary LSAs are generated for this area.
  - Summary: Summary LSAs are generated and sent to the area.
  - No Summary: Summary LSAs are not generated and sent to the area.
- Metric
  Displays the costs for the OSPFv3 interface.
- Updates
  Shows the number of recalculations of the IPv6 routing tables.
- LSA Count
  Shows the number of LSAs in the database.
- Area BR
  Shows the number of reachable area border routers (ABR) within this area.
- AS BR
  Shows the number of reachable autonomous system border routers (ASBR) in this area.

Area Range

Creating a new OSPFv3 area range

On this page, you can group up to 4 networks under one area ID. The method is used only with area border routers. This means that an area border router only advertises one route for grouped areas to the outside.

Note

This function is available only with layer 3.

Description of the displayed boxes

- **Area ID**
  Select the ID of the area. You specify the ID in "Areas".
- **Prefix**
  Enter the IPv6 address of the network that will be grouped.
- **Prefix Length**
  Enter the number of bits belonging to the prefix that will be grouped.

This table contains the following columns:

- **Area ID**
  Shows the ID of the area.
- **Prefix**
  Shows the IPv6 address of the network that will be grouped.
- **Prefix Length**
  Enter the number of bits that will belong to the prefix.
- **Advertise**
  Enable this option to advertise the grouped network.

Interfaces

Overview

On this page, you can configure OSPFv3 interfaces.

Note

This function is available only with layer 3.

Description of the displayed boxes

- **Interface**
  Select the IPv6 interface on which OSPFv3 will be enabled.
- **Area ID**
  Select the area ID of the area with which the OSPFv3 interface is connected.

  Note

  For the secondary interface select the same Area ID as for the corresponding primary interface.
  The information whether a primary or secondary interface is involved can be found in "Layer 3 (IPv6) > Subnets".

This table contains the following columns:

- Interface
  Shows the OSPFv3 interface.
- Area ID
  Select the area ID of the area with which the OSPFv3 interface is connected.
- Passive Interface
  Specify whether or not OSPFv3 is active on the interface.
  - Enabled
    No OSPFv3 information (e.g. Hello frames and LSDB updates) is sent via this interface and learned.
  - Disabled
    OSPFv3 information is sent via this interface and learned.
- Metric
  Enter the costs for the OSPFv3 interface.
- Priority
  Enter the router priority. The priority is only relevant for selecting the designated router or designated border router. This parameter can be selected differently on routers within the same subnet.
  Range of values: 0 to 255
  Default: 1.
- Trans. Delay
  Enter the required delay when sending a connection update.
  Range of values: 1 s to 3600 s
  Default setting: 1 s
- Retrans. Interval
  Enter the time after which an OSPFv3 packet is transferred again if no confirmation was received.
  Range of values: 1 s to 3600 s
  Default setting: 5 s
- Hello Interval
  Enter the interval between two Hello packets.
  Range of values: 1 s to 65,535 s
  Default setting: 10 s
- Dead Interval
  Enter the interval after which the neighbor router is marked as "failed" if no more Hello packets are received from it during this time.
  Default setting: 60 s

Virtual links

Overview

Due to the protocol, each area border router must have access to the backbone area. If a router is not connected directly to the backbone area, a virtual link to it is created.

Note

This function is available only with layer 3.

Note

Note that when creating a virtual link both the transit area and the backbone area must already be configured.

A virtual link must be configured identically at both ends.

Description of the displayed boxes

The page contains the following:

- **Neighbor Router ID**
  Enter the ID of the neighbor router at the other end of the virtual connection.
- **Transit Area ID**
  Select the ID of the area that connects the two routers.

This table contains the following columns:

- **Transit Area ID**
  Shows the ID via which the two routers are connected.
- **Neighbor Router ID**
  Shows the ID of the neighbor router at the other end of the virtual connection.
- **Virt. Link Status**
  Specify the status of the virtual link. The following states are possible:
  - down: The virtual link is inactive.
  - point-to-point: The virtual link is active.
- **Trans. Delay**
  Enter the required delay when sending a connection update.
  Range of values: 1 s to 3600 s
  Default setting: 1 s
- **Retrans. Interval**
  Enter the time after which a packet is transferred again if no confirmation was received.
  Range of values: 1 s to 3600 s
  Default setting: 5 s

- **Hello Interval**
  Enter the interval between two Hello packets.
  Range of values: 1 s to 65,535 s
  Default setting: 10 s
- **Dead Interval**
  Enter the interval after which the neighbor router counts as "failed" if no more Hello packets are received from it during this time.
  Default setting: 60 s

RIPng

Configuration

On this page, you configure routing with RIPng.

Note

This function is available only with layer 3.

Description of the displayed boxes

- **Inbound Filter**
  Select a route map that filters inbound routes.
- **Redistribute Routes**
  Specify which known routes are distributed using RIPng.
  The following types of route exist:
  - Connected
  - Static
  - OSPFv3
- **Route Map**
  Select a route map that filters which routes are forwarded using RIPng.
- **Protocol Preferences**
  - **Default Distance**
    Set the administrative distance for the OSPFv2 protocol.
- **Route Map Preferences**
  - **Distance Route Map**
    Select a route map from the drop-down list for which you want to specify the administrative distance.
  - **Distance**
    Configure the administrative distance for the previously selected route map

Interfaces

On this page, you specify the interfaces that support RIPng.

Note

This function is available only with layer 3.

Description of the displayed boxes

This table contains the following columns:

- **Interface**
  Shows the available IPv6 interface.
- **Admin State**
  Enable or disable the RIPng routing for the interface.
- **Default Metric**
  Enter the costs for the RIPng interface.

Configuring security functions

User management (WBM or CLI)

Overview of user management

Access to the device is managed by configurable user settings. Set up users with a password for authentication. Assign a role with suitable rights to the users.

The authentication of users can either be performed locally by the device or by an external RADIUS server. You configure how the authentication is handled on the "Security > AAA > General" page.

Note

When you transfer the configuration of a device to STEP 7 (TIA Portal), the configured users, roles and groups are not transferred.

Compatibility with predecessor versions

With firmware version 5.1, user management was expanded by the RADIUS authorization mode "Vendor Specific". To ensure compatibility with firmware versions ≤ 5.0, the default setting was selected so that following a firmware update the earlier authentication mode "conventional" continues to be used.

Local logon

The local logging on of users by the device runs as follows:

1. The user logs on with user name and password on the device.
2. The device checks whether an entry exists for the user.
   → If an entry exists, the user is logged in with the rights of the associated role.
   → If no corresponding entry exists, the user is denied access.

Login via an external RADIUS server

RADIUS(Remote Authentication Dial-In User Service) is a protocol for authenticating and authorizing users by servers on which user data can be stored centrally.

Depending on the RADIUS authorization mode you have selected on the "Security > AAA > RADIUS Client" page, the device evaluates different information of the RADIUS server.

RADIUS authorization mode "Standard"

If you have set the authorization mode "conventional", the authentication of users via a RADIUS server runs as follows:

1. The user logs on with user name and password on the device.
2. The device sends an authentication request with the login data to the RADIUS server.
3. The RADIUS server runs a check and signals the result back to the device.
   - The RADIUS server reports a successful authentication and for the "Service Type" attribute returns the value "Administrative User" to the device
     → The user is logged in with read/write rights.
   - The RADIUS server reports a successful authentication and returns a different or even no value to the device for the attribute "Service Type".
     → The user is logged in with read rights.
   - The RADIUS server reports a failed authentication to the device:
     → The user is denied access.

RADIUS authorization mode "Vendor Specific"

Requirement

For the RADIUS authorization mode "Vendor Specific" the following needs to be set on the RADIUS server:

- Manufacturer code: 4196
- Attribute number: 1
- Attribute format: Character string (group name)

Procedure

If you have set the authorization mode "Vendor Specific", the authentication of users via a RADIUS server runs as follows:

1. The user logs on with user name and password on the device.
2. The device sends an authentication request with the login data to the RADIUS server.
3. The RADIUS server runs a check and signals the result back to the device.
   Case A: The RADIUS server reports a successful authentication and returns the group assigned to the user to the device.
   - The group is known on the device and the user is not entered in the table "External User Accounts"
     → The user is logged in with the rights of the assigned group.
   - The group is known on the device and the user is entered in the table "External User Accounts"
     → The user is assigned the role with the higher rights and logged in with these rights.
   - The group is not known on the device and the user is entered in the table "External User Accounts"
     → The user is logged in with the rights of the role linked to the user account.
   - The group is not known on the device and the user is not entered in the table "External User Accounts"
     → The user is logged in with the rights of the role "Default".

   Case B: The RADIUS server reports a successful authentication but does not return a group to the device.
   - The user is entered in the table "External User Accounts":
     → The user is logged in with the rights of the linked role "".
   - The user is not entered in the table "External User Accounts":
     → The user is logged in with the rights of the role "Default".

   Case C: The RADIUS server reports a failed authentication to the device:
   - The user is denied access.

Assignment of a VLAN via RADIUS or guest VLAN

Authentication with a change to the VLAN configuration

If during authentication a port is assigned to a VLAN dynamically using the function "RADIUS VLAN Assignment Allowed" or "Guest VLAN" the options are as follows:

- If the VLAN that is to be assigned has not been created on the device, the authentication is rejected.
- If the VLAN that is to be assigned has been created on the device:
  - The port becomes an untagged member in the assigned VLAN if it was not already.
    → This makes it possible for the static configuration of the port in this VLAN to be overwritten and not restored if the authentication is retracted.
  - The port VID of the port is changed to the ID of the assigned VLAN.

Note

If the port is only to be assigned to one VLAN, you need to adapt the VLAN configuration manually. As default, all ports are untagged members in "VLAN 1".

If the authentication is canceled, e.g. by link down, the dynamic changes are canceled.

- The port is no longer a member in the assigned VLAN.
- The port VID of the port is reset to the value it had prior to authentication.

Note

If the port VID corresponds to the assigned port VID prior to authentication, the port remains an untagged member in this VLAN.

Authentication without a change to the VLAN configuration

If during authentication no VLAN is assigned either by the function "RADIUS VLAN Assignment Allowed" or by "Guest VLAN", the existing VLAN configuration of the port remains unchanged.

Users

Local Users

Note

The page is only available if there is an online connection to the device.

Local users

On this page, you create local users with the corresponding rights.

When you create or delete a local user this change is also made automatically in the table "External User Accounts". If you want to make change explicitly for the internal or external user table, use the CLI commands.

Note

The values displayed depend on the rights of the logged-in user.

Description of the displayed boxes

- User Account
  Enter the name for the user. The name must meet the following conditions:
  - It must be unique.
  - It must be between 1 and 255 characters long.
  - It must not contain the following characters: | ? " ; :
    The characters for Space and Delete must also not be included.

  Note

  User name cannot be changed

  After creating a user, the user name can no longer be modified.

  If a user name needs to be changed, the user must be deleted and a new user created.

  Note

  Default user "user" set in the factory

  As of firmware version 6.0 the default user set in the factory "user" is no longer available when the product ships. If you update a device to the firmware V6.0 the default user set in the factory "user" is initially still available. If you reset the device to the factory settings ("Restore Factory Defaults and Restart") the default user set in the factory "user" is deleted. You can create new users with the role "user".

- Password Policy
  Shows which password policy is being used.
  - High
    Password length: at least 8 characters, maximum 128 characters
    at least 1 upper case letter
    at least 1 special character
    at least 1 number
  - Low
    Password length: at least 6 characters, maximum 128 characters

  You configure the password policy on the page "Security > Passwords > Options".

- Password
  Enter the password for the user. The strength of the password depends on the set password policy.

- **Password Confirmation**
  Enter the password again to confirm it.
- **Role**
  Select a role.
  You can choose between system-defined and self-defined roles, refer to the page "Security > Users > Roles.".

The table contains the following columns:

Note

The preset users as well as logged in users cannot be deleted or changed.

- **User Account**
  Shows the user name.
- **Role**
  Shows the role of the user.
- **Description**
  Displays a description of the user account. The description text can be up to 100 characters long.

Roles

Note

The page is only available if there is an online connection to the device.

Roles

On this page, you create roles that are valid locally on the device.

Note

The values displayed depend on the rights of the logged-in user.

Description of the displayed boxes

The page contains the following:

- **Role Name**
  Enter the name for the role. The name must meet the following conditions:
  - It must be unique.
  - It must be between 1 and 64 characters long.

  Note
  Role name cannot be changed

  After creating a role, the name of the role can no longer be changed.

  If a name of a role needs to be changed, the role must be deleted and a new role created.

This table contains the following columns:

Note
Deleting roles

Predefined roles and assigned roles cannot be deleted or modified.

- **Role**
  Shows the name of the role.
- **Function Right**
  Select the function rights of the role.
  - 1
    Users with this role can read device parameters but cannot change them. Users with this role can change their own password.
  - 15
    Users with this role can both read and change device parameters.

  Note
  Function right cannot be changed

  If you have assigned a role, you can no longer change the function right of the role.

  If you want to change the function right of a role, follow the steps outlined below:
  1. Delete all assigned users.
  2. Change the function right of the role:
  3. Assign the role again.

- **Description**
  Enter a description for the role. With predefined roles a description is displayed. The description text can be up to 100 characters long.

Groups

Note

The page is only available if there is an online connection to the device.

User Groups

On this page you link a group with a role.

In this example the group "Administrators" is linked to the "admin"role: The group is defined on a RADIUS server. The role is defined locally on the device. When a RADIUS server authenticates a user and assigns the user to the "Administrators" group, this user is given rights of the "admin" role.

Note

The values displayed depend on the rights of the logged-on user.

Description of the displayed boxes

The page contains the following:

- **Group Name**
  Enter the name of the group. The name must match the group on the RADIUS server. The name must meet the following conditions:
  - It must be unique.
  - It must be between 1 and 64 characters long.

The table contains the following columns:

- **Group**
  Shows the name of the group.
- **Role**
  Select a role. Users who are authenticated with the linked group on the RADIUS server receive the rights of this role locally on the device. You can choose between system-defined and self-defined roles, refer to the page "Security > Users > Roles.".
- **Description**
  Enter a description for the link of the group.to a role. The description text can be up to 100 characters long.

Passwords

Passwords

Configuration of the device passwords

Note

The page is only available if there is an online connection to the device.

Note

If you are logged in via a RADIUS server, you cannot change any local device passwords.

On this page, you can change passwords. If you are logged on with the "admin" role, you can change the passwords for all user accounts. If you are logged on with the "user" role, you can only change your own password.

Note

When you log in for the first time or following a "Restore Factory Defaults and Restart", with the preset user "admin" you will be prompted to change the password.

The factory settings for the passwords when the devices ship are as follows:

- admin: admin

Note

Changing the password in Trial mode

Even if you change the password in "Trial" configuration mode, this change is saved immediately.

Description of the displayed boxes

- **Current User**
  Shows the user that is currently logged in.
- **Current User Password**
  Enter the password for the currently logged in user.
- **User Account**
  Select the user whose password you want to change.

- **Password Policy**
  Shows which password policy is being used when assigning new passwords.
  - High
    Password length: at least 8 characters, maximum 128 characters
    at least 1 uppercase letter
    at least 1 special character
    at least 1 number
  - Low
    Password length: at least 6 characters, maximum 128 characters
- **New name for the Admin user account**
  You can change the name the default user "admin" once. Afterwards, renaming "admin" is no longer possible.
- **New Password**
  Enter the new password for the selected user. The following characters must not be included:
  – § ? " ; : –
  The space and the character for "remove" must also not be included.
- **Password Confirmation**
  Enter the new password again to confirm it.

Options

On this page you specify which password policy will be used when assigning new passwords.

Description

- **Password Policy**
  Shows which password policy is currently being used
- **New Password Policy**
  Select the required setting from the drop-down list.
  - High
    Password length: at least 8 characters, maximum 128 characters
    at least 1 uppercase letter
    at least 1 special character
    at least 1 number
  - Low
    Password length: at least 6 characters, maximum 128 characters

AAA

General

Login of network nodes

The designation "AAA" used in the menu stands for "Authentication, Authorization, Accounting". This feature is used to identify and allow network nodes and to make the corresponding services available to them.

On this page, you configure the login.

Description of the displayed boxes

The page contains the following boxes:

Note

To be able to use the login authentication "RADIUS", "Local and RADIUS" or "RADIUS and fallback Local" a RADIUS server must be stored and configured for user authentication.

- **Login Authentication**
  Specify how the login is made:
  - Local
    The authentication must be made locally on the device.
  - RADIUS
    The authentication must be handled via a RADIUS server.
  - Local and RADIUS
    The authentication is possible both with the users that exist on the device (user name and password) and via a RADIUS server. The user is first searched for in the local database. If the user does not exist there, a RADIUS query is sent.
  - RADIUS and fallback Local
    The authentication must be handled via a RADIUS server. A local authentication is performed only when the RADIUS server cannot be reached in the network.

RADIUS client

Authentication over an external server

The concept of RADIUS is based on an external authentication server.

Each row of the table contains access data for one server. In the search order, the primary server is queried first. If the primary server cannot be reached, secondary servers are queried in the order in which they are entered. If no server responds, there is no authentication.

Description of the displayed boxes

The page contains the following boxes:

- RADIUS Authorization Mode
  For the login authentication, the RADIUS authorization mode specifies how the rights are assigned to the user with a successful authentication (Page 2017).
  - conventional
    In this mode the user is logged in with administrator rights if the server returns the value "Administrative User" to the device for the attribute "Service Type". In all other cases the user is logged in with read rights.
  - Vendor specific
    In this mode the assignment of rights depends on whether and which group the server returns for the user and whether or not there is an entry for the user in the table "External User Accounts".

The table has the following columns:

- Auth. Server Type
  Select which authentication method the server will be used for.
  - Login
    The server is used only for the login authentication.
  - 802.1X
    The server is used only for the 802.1X authentication.
  - Login & 802.1X
    The server is used for both authentication procedures.
- RADIUS Server Address
  Enter the IPv4 address or the FQDN (Fully Qualified Domain Name) of the RADIUS server.
- Server Port
  Here, enter the input port on the RADIUS server. As default, input port 1812 is set. The range of values is 1 ... 65535.
- Shared Secret
  Enter your access ID. The range of values is 1 ... 128 characters.
- Confirm Shared Secret
  Enter your access ID again as confirmation.
- Max. Retrans
  Here, enter the maximum number of retries for an attempted request. The initial connection attempt is repeated the number of times specified here before another configured RADIUS server is queried or the login counts as having failed. As default 3 retries are set, this means 4 connection attempts. The range of values is 1 ... 5.
- Primary Server
  Specify whether or not this server is the primary server. You can select one of the options "yes" or "no".

- **Test**
  With this button, you can test whether or not the specified RADIUS server is available. The test is performed once and not repeated cyclically.
- **Test Result**
  Shows whether or not the RADIUS server is available:
  - Failed, no test packet sent
    The IP address is not reachable.
    The IP address is reachable, the RADIUS server is, however, not running.
  - Reachable, key not accepted
    The IP address is reachable, the RADIUS server does not, however accept the shared secret.
  - Reachable, key accepted
    The IP address is reachable, the RADIUS server accepts the specified shared secret.

  The test result is not automatically updated. To delete the test result click the "Refresh" button.

802.1x-Authenticator

Setting up network access

An end device can only access the network after the device has verified the login data of the device with the authentication server. The authentication can be via 802.1X or the MAC address. When authenticating using 802.1X both the end device and the authentication server must support the EAP protocol (Extensive Authentication Protocol).

Enabling authentication for individual ports

By selecting the check box, you specify whether or not network access protection according to IEEE 802.1x is enabled on this port.

Description of the displayed boxes

The page contains the following boxes:

- **MAC Authentication**
  Enable or disable MAC authentication for the device.
- **Guest VLAN**
  Enable or disable the "Guest VLAN" function for the device.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **802.1X Auth. Control**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.

- 802.1x Re-authentication
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- MAC Authentication
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
  MAC Auth. only on Timeout
  Select the desired setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Adopt RADIUS VLAN Assignment
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- MAC Auth. Max Permitted Addresses
  Enter how many end devices are allowed to be connected to the port at the same time. If "No Change" is entered, the entry in table 2 remains unchanged
- Guest VLAN
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Guest VLAN ID
  Enter the VLAN ID of the port. If "No Change" is entered, the entry in table 2 remains unchanged
- Guest VLAN max. Permitted Addresses
  Enter how many end devices are allowed in the guest VLAN at the same time. If "No Change" is entered, the entry in table 2 remains unchanged
- Transfer to table
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- Port
  This column lists all the ports available on this device.
- 802.1X Auth. Control
  Specify the authentication of the port:
  - Force Unauthorized
    Data traffic via the port is blocked.
  - Force Authorized
    Data traffic via the port is allowed without any restrictions. Default setting
  - Auto
    End devices are authenticated on the port with the "802.1x" method. The data traffic via the port is permitted or blocked depending on the authentication result.
- 802.1x Re-authentication
  Enable this option if you want reauthentication of an already authenticated end device to be repeated cyclically.

- MAC Authentication
  Enable this option if you want end devices to be authenticated with the "MAC Authentication" method.
  If "Auto is configured for "802.1x Auth. Control and the " MAC Authentication is enabled, the timeout for the "802.1X procedure is 5 seconds. If manual input is necessary at a port for the authentication with the 802.1X" procedure, the 5 seconds are not adequate. To be able to run authentication using "802.1X", disable the MAC authentication on this port.
- MAC Auth. only on Timeout
  If this check box is selected, MAC authentication is only possible after a 802.1X timeout, but not after a failed 802.1X authentication. When the check box is not selected, MAC authentication is possible both after a 802.1X timeout and after a failed 802.1X authentication.
- Adopt RADIUS VLAN Assignment
  The RADIUS server informs the IE switch of the VLAN to which the port will belong. Enable this option if you want the information of the server to be taken into account.
  The port can only e assigned to the VLAN, if the VLAN has been created on the device. Otherwise authentication (page 474) is rejected. If during authentication a port is assigned to a VLAN dynamically using this function, assignment using the VLAN-ID or the VLAN name is possible. Configure the following values on the RADIUS server:
  - Tunnel-Type = VLAN
  - Tunnel-Medium-Type = IEEE-802
  - Tunnel-Private-Group-Id = VLAN ID or VLAN name

  The IE switch distinguishes as follows:
  - VLAN ID: The RADIUS server transfers a numeric string for the parameter "Tunnel-Private-Group-Id".
  - VLAN name: The RADIUS server transfers an alphanumeric string for the parameter "Tunnel-Private-Group-Id".

- MAC Auth. Max Permitted Addresses
  - 1 ... 100
    Specify how many MAC addresses can communicate on the port at the same time.

    Note

    If a device uses several MAC addresses, all MAC addresses must be authenticated. Store all the MAC addresses to be authenticated on the RADIUS server. Enter the number in the "MAC Auth. Max Allowed Addresses" box.

  - 0
    You can set the value "0". This setting has the effect that after the first successful authentication of a MAC address, the port is released for all MAC addresses.
    Use case
    If you configure the value "0" for a port. connect this port to a WLAN access point. After the AP has authenticated itself successfully, all MAC addresses are released on this port. All WLAN clients connected to the AP can communicate on the port without their own authentication. Make sure that the clients are authenticated by the AP.
    Firmware compatibility
    In firmware versions < 6.0.2 the value "0" is not a valid entry. If you configure the value "0" with a firmware version ≥ 6.0.2 and copy a firmware version <6.0.2 to the device, the function MAC authentication is deactivated on the relevant ports. Reconfigure the MAC authentication on these ports:
- Guest VLAN
  Enable this option if you want the end device to be permitted in the guest VLAN if authentication fails.
  The port can only e assigned to the VLAN, if the VLAN has been created on the device. Otherwise Authentication (Page 2017) is rejected.
  This function is also known as "Authentication failed VLAN".
- Guest VLAN ID
  Enter the VLAN ID of the port.
- Guest VLAN max. Permitted Addresses
  Enter how many end devices are allowed in the guest VLAN at the same time.
- 802.1X Auth.-Status
  Shows the status of the authentication of the port:
  - Authorize
  - Do not authorize
- MAC Auth. Actual Allowed Addresses
  Shows the number of currently connected end devices.
- MAC Auth. Currently Blocked Addresses
  Shows the number of currently blocked end devices.
- Guest VLAN Currently Permitted Addresses
  Shows how many end devices are currently allowed in the guest VLAN.

MAC ACL

Rule configuration

Introduction

On this page, you specify the access rules for the MAC-based Access Control List. Using the MAC-based ACL, you can specify whether frames of certain MAC addresses are forwarded or discarded.

Description of the displayed boxes

The table has the following columns:

- Rule Number
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- Source MAC
  Enter the unicast MAC address of the source.
  Destination MAC Address
  Enter the unicast MAC address of the destination.
- Action
  Select the action. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- Ingress Interfaces
  Shows a list of all interfaces to which this rule applies.
- Egress Interfaces
  Shows a list of all interfaces to which this rule applies.

Note

Entering the MAC addresses

You can configure the access rules only for unicast MAC addresses. Only if you enter the address "00:00:00:00:00:00" for the source and/or destination MAC address, the rule created in this way applies to all source or destination MAC addresses.

Ingress Rules

Introduction

On this page, you specify the ACL rules according to which incoming frames are filtered at interfaces. You specify the ACL rules in the "Rules Configuration" tab.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  Select the required interface from the drop-down list. The available interfaces depend on your device.
- Add Rule
  Select the ACL rule to be assigned to the interface. You specify the ACL rule on the "Rules Configuration" page.
- Add
  To permanently assign the ACL rule to the interface, click the "Add" button. The configuration is shown in the table.
- Remove Rule
  Select the ACL rule to be deleted.
- Remove
  To remove the ACL rule from the interface, click the "Remove" button.

The table has the following columns:

- Rule Order
  Shows the order of the ACL rules.
- Rule Number
  Shows the number of the ACL rule.
- Source MAC
  Shows the unicast MAC address of the source.
- Destination MAC Address
  Shows the unicast MAC address of the destination.
- Action
  Shows the action:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.

Egress Rules

Introduction

On this page, you specify the ACL rule according to which outgoing frames are handled by the port. You specify the ACL rule in the "Rules Configuration" tab.

Description of the displayed boxes

The page contains the following boxes:

- **Interface**
  Select the required interface. The available interfaces depend on your device.
- **Add Rule**
  Select the ACL rule to be assigned to the interface.
- **Add**
  To permanently assign the ACL rule to the interface, click the "Add" button. The configuration is shown in the table.
- **Remove Rule**
  Select the ACL rule to be deleted.
- **Remove**
  To remove the ACL rule from the interface, click the "Remove" button.

The table has the following columns:

- **Rule Order**
  Shows the order of the ACL rules.
- **Rule Number**
  Shows the number of the ACL rule.
- **Source MAC**
  Shows the unicast MAC address of the source.
- **Destination MAC Address**
  Shows the unicast MAC address of the destination.
- **Action**
  Shows the action
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.

IP ACL

Rule configuration

Introduction

On this page, you specify the rules for the IP-based Access Control List. Using the IP-based ACL, you can specify whether frames of certain IPv4 addresses are forwarded or discarded.

Description of the displayed boxes

The table has the following columns:

- Rule Number
  Shows the number of the ACL rule. If you create a new entry, a new row with a unique number is created.
- Source IP
  Enter the IPv4 address of the source.
- Source Subnet Mask
  Enter the subnet mask in which the source is located.
- Dest. IP
  Enter the IP address of the destination.
- Dest. Subnet Mask
  Enter the subnet mask in which the destination is located.
- Action
  Select whether the frame is forwarded or rejected when it corresponds to the ACL rule.
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- Ingress Interfaces
  Shows a list of all interfaces to which this rule applies.
- Egress Interfaces
  Shows a list of all interfaces to which this rule applies.

Note

Subnet mask for individual hosts

If you create the rule for a single system (one IPv4 address), specify the subnet mask "255.255.255.255".

Protocol configuration

On this page, you specify the rules for protocols.

Description of the displayed boxes

The table has the following columns:

- Rule Number
  Shows the number of the protocol rule. When you create a rule, a new row with a unique number is created.
- Protocol
  Select the protocol for which this rule is valid.

- Protocol Number
  Enter a protocol number to define further protocols. This box can only be edited if you have set "Other Protocol" for the protocol.
- Source Port Min.
  Enter the lowest possible port number of the source port. This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Source Port Max.
  Enter the highest possible port number of the source port. This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Dest. Port Min.
  Enter the lowest possible port number of the destination port. This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Dest. Port Max.
  Enter the highest possible port number of the destination port. This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Message Type
  Enter a message type to decide the format of the message.
  This box can only be edited if you have set "ICMP" for the protocol.
- Message Code
  Enter a message code to specify the function of the message. This box can only be edited if you have set "ICMP" for the protocol.
- DSCP
  Enter a value for classifying the priority. This box cannot be edited if you have set "ICMP" for the protocol.

Ingress Rules

Introduction

On this page, you specify the ACL rules according to which incoming frames are handled by interfaces. You specify the ACL rules in the "Rules Configuration" tab.

Description of the displayed boxes

Description of the displayed boxes:

- Interface
  Select the required port: a port or a VLAN interface. The available interfaces depend on your device. To select a VLAN interface, an IP interface must be configured.

  Note

  If you use a VLAN interface, the ACL rule applies to all ports that belong to the VLAN.

- Add Rule
  Select the ACL rule to be assigned to the port.

- **Add**
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- **Remove Rule**
  Select the ACL rule to be deleted.
- **Remove**
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- **Rule Order**
  Shows the order of the ACL rules.
- **Rule Number**
  Shows the number of the ACL rule.
- **Protocol**
  Shows the protocol for which this rule is valid.
- **Protocol Number**
  Shows the protocol number.
- **Source IP**
  Shows the IP address of the source.
- **Source Subnet Mask**
  Shows the subnet mask in which the source is located.
- **Dest. IP**
  Shows the IP address of the destination.
- **Dest. Subnet Mask**
  Shows the subnet mask of the destination.
- **Action**
  Select whether the frame is forwarded or rejected when it corresponds to the ACL rule.
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- **Source Port Min.**
  Shows the lowest possible port number of the source port.
- **Source Port Max.**
  Shows the highest possible port number of the source port.
- **Dest. Port Min.**
  Shows the lowest possible port number of the destination port.
- **Dest. Port Max.**
  Shows the highest possible port number of the destination port.
- **Message Type**
  Shows a message type to decide the format of the message.

- Message Code
  Shows a message code to specify the function of the message.
- DSCP
  Shows the value for classifying the priority.

Egress Rules

Introduction

On this page, you specify the ACL rule according to which outgoing frames are handled by the port. You specify the ACL rules in the "Rules Configuration" tab.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  Select the required port: a port or a VLAN interface. The available interfaces depend on your device. To select a VLAN interface, an IP interface must be configured.

  Note

  If you use a VLAN interface, the ACL rule applies to all ports that belong to the VLAN.

- Add Rule
  Select the ACL rule to be assigned to the port.
- Add
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- Remove Rule
  Select the ACL rule to be deleted.
- Remove
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- Rule Order
  Shows the order of the ACL rules.
- Rule Number
  Shows the number of the ACL rule.
- Protocol
  Shows the protocol for which this rule is valid.
- Protocol Number
  Shows the protocol number.
- Source IP
  Shows the IP address of the source.
- Source Subnet Mask
  Shows the subnet mask of the source.

- Dest. IP
  Shows the IP address of the destination.
- Dest. Subnet Mask
  Shows the subnet mask of the source.
- Action
  Select whether the frame is forwarded or rejected when it corresponds to the ACL rule.
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- Source Port Min.
  Shows the lowest possible port number of the source port.
- Source Port Max.
  Shows the highest possible port number of the source port.
- Dest. Port Min.
  Shows the lowest possible port number of the destination port.
- Dest. Port Max.
  Shows the highest possible port number of the destination port.
- Message Type
  Shows a message type to decide the format of the message.
- Message Code
  Shows a message code to specify the function of the message.
- DSCP
  Shows the value for classifying the priority.

Management ACL

Description of configuration

On this page, you can increase the security of your device. To specify which station with which IPv4 address is allowed to access your device, configure the IPv4 address or an entire address range.

You can select the protocols and the ports with which the station is allowed to access the device. You define the VLAN in which the station may be located. This ensures that only certain stations within a VLAN have access to the device.

Description of the displayed boxes

Note

If you enable this function, note the following

A bad configuration on the "Management Access Control List" page can result in you being unable to access the device. You should therefore configure an access rule that allows access to the management before you enable the function.

The page contains the following boxes:

- **Management ACL**
  Enable or disable the function. As default, the function is disabled.

  Note

  If the function is disabled, there is unrestricted access to the management of the IE switch. The configured access rules are only taken into account when the function is enabled.

- **IP Address**
  Enter the IPv4 address or the network address for which the rule will apply. If you use the IPv4 address 0.0.0.0, the settings apply to all IPv4 addresses. If you use the IPv6 address :: the settings apply to all IPv6 addresses.
- **Subnet Mask**
  Enter the subnet mask or the prefix length. The subnet mask 255.255.255.255 is for a specific IPv4 address. If you want to allow a subnet, for example a C subnet, enter 255.255.255.0. The subnet mask 0.0.0.0 applies to all subnets.

The table has the following columns:

- **Rule Order**
  Shows the order of the ACL rules.
- **IP Address**
  Shows the IPv4 address.
- **Subnet Mask / Prefix Length**
  Shows the subnet mask or the prefix length.
- **VLANs Allowed**
  Enter the number of the VLAN in which the device is located. The station can only access the device if it is located in this configured VLAN. If this input box remains empty, there is no restriction relating to the VLANs.
- **Out-Band**
  Specify whether or not the IPv4 address can access the switch via the out-band port.
- **SNMP**
  Specify whether the station (or the IP address) can access the device using the SNMP protocol.
- **TELNET**
  Specify whether the station (or the IP address) can access the device using the TELNET protocol.

- **HTTP**
  Specify whether the station (or the IP address) can access the device using the HTTP protocol.
- **HTTPS**
  Specify whether the station (or the IP address) can access the device using the HTTPS protocol.
- **SSH**
  Specify whether the station (or the IP address) can access the device using the SSH protocol.
- **Px.y**
  Specify whether the station (or the IP address) can access this device via this port (slot.port).

Configuring SCALANCE W

Interfaces and system functions

SCALANCE W780 / W740

Availability of the interfaces

The following table shows the availability of the physical and logical interfaces. Note that in this table all interfaces are listed. Depending on the system function, some interfaces are not available. On the WBM pages you can only select the available interfaces.

We reserve the right to make technical changes.

	Client device **W748-1 M12** **W748-1 RJ-45**	**Access points** **W786-1 RJ-45** **W788-1 M12** **W788-1 RJ-45**	**Access points** **W786-2 RJ-45** **W786-2IA RJ-45** **W786-2 SFP** **W788-2 M12** **W788-2 M12 EEC** **W788-1 RJ-45** **W788-2 RJ-45**
Wireless interface (WLAN)	WLAN 1	WLAN 1	• WLAN 1 (in client mode only one WLAN interface is available) • WLAN 2
IP interface: LAN interface VLAN	 P1 ManagementVLAN	 P1 ManagementVLAN	 P1 ManagementV-LAN

	Client device W748-1 M12 W748-1 RJ-45	Access points W786-1 RJ-45 W788-1 M12 W788-1 RJ-45	Access points W786-2 RJ-45 W786-2IA RJ-45 W786-2 SFP W788-2 M12 W788-2 M12 EEC W788-1 RJ-45 W788-2 RJ-45
VAP interface [1]	-	VAP 1.Y Y = 1 ... 8	VAP X.Y X = 1 ... 2 Y = 1 ... 8
WDS interface [1]	_	WDS 1.Y Y = 1 ... 8	WDS X.Y X = 1 ... 2 Y = 1 ... 8
VLAN	24	24	24

[1] only in access point mode

Availability of the system functions

The following table shows the availability of the system functions on the devices. Note that all functions are described in this configuration manual and in the online help. Depending on the mode and the KEY-PLUG, some functions are not available.

We reserve the right to make technical changes.

			Access point mode	Access points in client mode. Client device
Information	Security	Inter AP blocking	✓ W780 iFeatures (MLFB 6GK5 907-8PA00) W700 Security (MLFB 6GK5907-0PA00)	-
	WLAN	AP overview	✓	-
		Client list	✓	-
		WDS list	✓	-
		AP overlap	✓	-
		Force roaming	✓	✓
		Client overview	-	✓
		Available APs	-	✓
		IP assignment	-	✓
		Background noise	✓	✓
	WLAN statistics	Error	✓	✓
		Management sent	✓	✓
		Management received	✓	✓
		Data sent	✓	✓
		Data received	✓	✓
	WLAN iFeatures	iREF client list	✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	-
		iREF WDS list	✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	-
		AeroScout	✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	-
System		PROFINET	✓	-✓
		EtherNet/IP	✓	✓

			Access point mode	Access points in client mode. Client device
Interfaces	WLAN	Basic	✓	-✓
		Expansions	✓	✓
		Antennas	✓	✓
		Permitted channels	✓	✓
		802.11n	✓	✓
		AP	✓	-
		AP WDS	✓	-
		AP 802.11a/b/g data rates	✓	-
		AP 802.11n data rates	✓	-
		Client 802.11a/b/g data rates	-	✓
		Client 802.11n data rates	-	✓
		Force roaming	✓	✓
		Signal recorder	-	✓
		Spectrum analyzer	✓	-
Layer 3	NAT	Basic	-	✓
		NAPT	-	✓

			Access point mode	Access points in client mode. Client device
Security	WLAN	Basic	✓	✓
		AP communication	✓	-
		AP RADIUS authenticator	✓	-
		Client RADIUS supplicant	-	✓
		Key	✓	✓
	Inter AP Blocking	Basic	✓ W780 iFeatures (MLFB 6GK5 907-8PA00) W700 Security (MLFB 6GK5907-0PA00)	-
		Allowed IP addresses	✓ W780 iFeatures (MLFB 6GK5 907-8PA00) W700 Security (MLFB 6GK5907-0PA00)	-

			Access point mode	Access points in client mode. Client device
iFeatures	iPCF		✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	✓ Access point in client mode: W780 iFeatures (MLFB 6GK5 907-8PA00 Client: W740 iFeatures (MLFB 6GK5 907-4PA00)
	iPCF-MC		✓ Only dual APs W780 iFeatures (MLFB 6GK5 907-8PA00)	✓ Access point in client mode: W780 iFeatures (MLFB 6GK5 907-8PA00 Client: W740 iFeatures (MLFB 6GK5 907-4PA00)
	iPRP		✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	✓ Access point in client mode: W780 iFeatures (MLFB 6GK5 907-8PA00 Client: W740 iFeatures (MLFB 6GK5 907-4PA00)
	iREF		✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	-
	AeroScout		✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	-

SCALANCE W770 / W730

Availability of the interfaces

The following table shows the availability of the physical and logical interfaces. Note that in this table all interfaces are listed. Depending on the system function, some interfaces are not available. On the WBM pages you can only select the available interfaces.

We reserve the right to make technical changes.

	Client device SCALANCE W734-1 RJ-45 SCALANCE W738-1 M12	Access points SCALANCE W774-1 RJ45 SCALANCE W774-1 M12 EEC SCALANCE W778-1 M12 SCALANCE W778-1 M12 EEC
Wireless interface (WLAN)	WLAN 1	WLAN 1
IP interface: LAN interface VLAN	 P1 ManagementVLAN	 P1 ManagementVLAN

	Client device SCALANCE W734-1 RJ-45 SCALANCE W738-1 M12	Access points SCALANCE W774-1 RJ45 SCALANCE W774-1 M12 EEC SCALANCE W778-1 M12 SCALANCE W778-1 M12 EEC
VAP interface [1]	-	VAP 1.Y Y = 1 ... 8
WDS interface [1]	_	WDS 1.Y Y = 1 ... 8
VLAN	16	16

[1] only in access point mode

Availability of the system functions

The following table shows the availability of the system functions on the devices. Note that all functions are described in this configuration manual and in the online help. Depending on the mode and the KEY-PLUG, some functions are not available.

We reserve the right to make technical changes.

			Access point mode	Access points in client mode Client devices
Information	Security	Inter AP blocking	✓ W780 iFeatures (MLFB 6GK5 907-8PA00) W700 Security (MLFB 6GK5907-0PA00)	-
	WLAN	Overview AP	✓	-
		Client List	✓	-
		WDS List	✓	-
		Overlap AP	✓	-
		Force Roaming	✓	-
		Overview Client	-	✓
		Available AP	-	✓
		IP Mapping	-	✓
	WLAN Statistics	Faults	✓	✓
		Management Sent	✓	✓
		Management Received	✓	✓
		Data Sent	✓	✓
		Data Received	✓	✓
	WLAN iFeatures	iREF Client List	✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	-
		iREF WDS List	✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	-
		AeroScout	✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	-
System		PROFINET	✓	-✓
		EtherNet/IP	✓	✓

			Access point mode	Access points in client mode Client devices
Interfaces	WLAN	Basic	✓	-✓
		Expansions	✓	✓
		Antennas	✓	✓
		Allowed Channels	✓	✓
		802.11n	✓	✓
		AP	✓	-
		AP WDS	✓	-
		AP 802.11a/b/g Rates	✓	-
		AP 802.11n Rates	✓	-
		Client 802.11a/b/g Rates	-	✓
		Client 802.11n Rates	-	✓
		Force Roaming	✓	✓
		Signal recorder	-	✓
		Spectrum Analyzer	✓	-
Layer 3	NAT	Basic	-	✓
		NAPT	-	✓
Security	WLAN	Basic	✓	✓
		AP Communication	✓	-
		AP RADIUS Authenticator	✓	-
		Client RADIUS Supplicant	-	✓
		Keys	✓	✓
	Inter AP Blocking	Basic	✓ W780 iFeatures (MLFB 6GK5 907-8PA00) W700 Security (MLFB 6GK5907-0PA00)	-
		Allowed Addresses	✓ W780 iFeatures (MLFB 6GK5 907-8PA00) W700 Security (MLFB 6GK5907-0PA00)	-

			Access point mode	Access points in client mode Client devices
iFeatures	iPCF	iPCFv1	✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	✓ Access point in client mode: W780 iFeatures (MLFB 6GK5 907-8PA00 Client: W740 iFeatures (MLFB 6GK5 907-4PA00)
	iPCF-MC		✓ Only dual APs W780 iFeatures (MLFB 6GK5 907-8PA00)	✓ Access point in client mode: W780 iFeatures (MLFB 6GK5 907-8PA00 Client: W740 iFeatures (MLFB 6GK5 907-4PA00)
	iPRP		✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	✓ Access point in client mode: W780 iFeatures (MLFB 6GK5 907-8PA00 Client: W740 iFeatures (MLFB 6GK5 907-4PA00)
	iREF		✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	-
	AeroScout		✓ W780 iFeatures (MLFB 6GK5 907-8PA00)	-

SCALANCE W760 / W720

Availability of the interfaces

The following table shows the availability of the physical and logical interfaces. Note that in this table all interfaces are listed. Depending on the system function, some interfaces are not available. On the WBM pages you can only select the available interfaces.

We reserve the right to make technical changes.

	Client device W721-1 RJ45 W722-1 RJ45	Access points W761-1 RJ45
Wireless interface (WLAN)	WLAN 1	WLAN 1
IP interface: LAN interface VLAN	 P1 ManagementVLAN	 P1 ManagementVLAN
VAP interface [1)]	-	VAP 1.1

	Client device W721-1 RJ45 W722-1 RJ45	Access points W761-1 RJ45
WDS interface [1]	_	WDS 1.1
VLAN	3	3

[1] only in access point mode

Availability of the system functions

The following table shows the availability of the system functions on the devices. Note that all functions are described in this configuration manual and in the online help. Depending on the mode and the KEY-PLUG, some functions are not available.

We reserve the right to make technical changes.

			Access point mode	Access points in client mode Client devices
Information	WLAN	Overview AP	✓	-
		Client List	✓	-
		WDS List	✓	-
		Overlap AP	✓	-
		Force Roaming	✓	-
		Overview Client	-	✓
		Available AP	-	✓
		IP Mapping	-	✓
	WLAN Statistics	Faults	✓	✓
		Management Sent	✓	✓
		Management Received	✓	✓
		Data Sent	✓	✓
		Data Received	✓	✓
System		PROFINET	✓	✓
		EtherNet/IP	✓	✓

			Access point mode	Access points in client mode Client devices
Interfaces	WLAN	Basic	✓	✓
		Advanced	✓	✓
		Antennas	✓	✓
		Allowed Channels	✓	✓
		802.11n	✓	✓
		AP	✓	-
		AP WDS	✓	-
		AP 802.11a/b/g Rates	✓	-
		AP 802.11n Rates	✓	-
		Force Roaming	✓	✓
		Signal recorder	-	✓
		Spectrum Analyzer	✓	-
Layer 3	NAT	Basic	-	✓
		NAPT	-	✓
Security	WLAN	Basic	✓	✓
		AP Communication	✓	-
		AP RADIUS Authenticator	✓	-
		Client RADIUS Supplicant	-	✓
		Keys	✓	✓
iFeatures	iPCF		-	✓ Only W722-1 RJ45 This can be integrated in a WLAN with iPCF, iPCF-MC.
	iPCF-HT		-	✓ Only W722-1 RJ45
	iPCF-MC		-	✓ Only W722-1 RJ45 This can be integrated in a WLAN with iPCF, iPCF-MC.
	iPRP		-	✓ Only W722-1 RJ-45

IPv6 support

Support of IPv6

The following system functions do not support IPv6 addresses:

- Inter AP blocking
- Force roaming
- IP ACL
- Management ACL

Configuration limits

SCALANCE W780 / W740

Configuration limits of the device

The following table lists the configuration limits for Web Based Management and the Command Line Interface of the device.

Depending on your device, some functions are not available.

	Configurable function		Maximum number
System	Syslog server		3
	DNS server	manual (IPv4/IPv6)	3
		learned (IPv4/IPv6)	2
		in total	7
	SMTP server		2
	SNMPv1 trap recipient		10
	SNTP server		2
	NTP server		1
	DHCP pools		1
	IPv4 addresses managed by the DHCP server (dynamic + static)		100
	DHCP static assignments per DHCP pool		20
	DHCP options		20
Interfaces	Force destination addresses for roaming		10
	Connected clients per VAP		126
Layer 2	Virtual LANs (port-based; including VLAN 1)		24
	Multiple Spanning Tree instances		16

	Configurable function	Maximum number
Security	IP addresses from RADIUS servers	• AAA: 4 • WLAN: 2
	Management ACLs (access rules for management)	10
	MAC ACL rule configuration	20
	Ingress and egress rules for MAC ACL (total)	40 per interface (20 ingress rules / 20 egress rules) • Client: 80 (P1, WLAN) • Access point: 680 (P1, WDS 1,Y, VAP 1,Y) • Dual access point: 1320 (P1, WDS X,Y, VAP X,Y)
	IP ACL rule configuration	20
	Ingress and egress rules for port ACL IP (total)	40 per interface (20 ingress rules / 20 egress rules) • Client: 120 (P1, WLAN, management VLAN) • Access point: 720 (P1, WDS 1,Y, VAP 1,Y, management VLAN) • Dual access point: 1360 (P1, WDS X,Y, VAP X,Y, management VLAN)
	User roles	28
	User groups	32
	Users	28

SCALANCE W770 / W730

Configuration limits of the device

The following table lists the configuration limits for Web Based Management and the Command Line Interface of the device.

Depending on your device, some functions are not available.

	Configurable function		Maximum number
System	Syslog server		3
	DNS server	manual (IPv4/IPv6)	3
		learned (IPv4/IPv6)	2
		in total	7
	SMTP server		3
	SNMPv1 trap recipient		10
	SNTP server		2
	NTP server		1
	DHCP pools		1
	IPv4 addresses managed by the DHCP server (dynamic + static)		100
	DHCP static assignments per DHCP pool		20
	DHCP options		20
Interfaces	Force destination addresses for roaming		10
	Connected clients per VAP		126
Layer 2	Virtual LANs (port-based; including VLAN 1)		16
	Multiple Spanning Tree instances		16
Security	IP addresses from RADIUS servers		• AAA: 4 • WLAN: 2
	Management ACLs (access rules for management)		10
	MAC ACL rule configuration		20
	Ingress and egress rules for MAC ACL (total)		40 per interface (20 ingress rules / 20 egress rules) • Client: 80 (P1, WLAN) • Access point: 680 (P1, WDS 1.Y, VAP 1.Y)
	IP ACL rule configuration		20
	Ingress and egress rules for port ACL IP (total)		40 per interface (20 ingress rules / 20 egress rules) • Client: 120 (P1, WLAN, management VLAN) • Access point: 720 (P1, WDS 1.Y, VAP 1.Y, management VLAN)
	User roles		28
	User groups		32
	Users		28

SCALANCE W760 / W720

Configuration limits of the device

The following table lists the configuration limits for Web Based Management and the Command Line Interface of the device.

Depending on your device, some functions are not available.

	Configurable function		Maximum number
System	Syslog server		3
	DNS server	manual (IPv4/IPv6)	3
		learned (IPv4/IPv6)	2
		in total	7
	SMTP server		3
	SNMPv1 trap recipient		10
	SNTP server		2
	NTP server		1
	DHCP pools		1
	IPv4 addresses managed by the DHCP server (dynamic + static)		100
	DHCP static assignments per DHCP pool		20
	DHCP options		20
Interfaces	Force destination addresses for roaming		10
	Connected clients per VAP		126
Layer 2	Virtual LANs (port-based; including VLAN 1)		3
	Multiple Spanning Tree instances		16

	Configurable function	Maximum number
Security	IP addresses from RADIUS servers	• AAA: 4 • WLAN: 2
	Management ACLs (access rules for management)	10
	MAC ACL rule configuration	20
	Ingress and egress rules for MAC ACL (total)	40 per interface (20 ingress rules / 20 egress rules) • Client: 80 (P1, WLAN) • Access point: 680 (P1, WDS 1.Y, VAP 1.Y)
	IP ACL rule configuration	20
	Ingress and egress rules for IP ACL IP (total)	40 per interface (20 ingress rules / 20 egress rules) • Client: 120 (P1, WLAN, management VLAN) • Access point: 720 (P1, WDS 1.Y, VAP 1.Y, management VLAN) • Dual access point: 1360 (P1, WDS X.Y, VAP X.Y, management VLAN)
	User roles	28
	User groups	32
	Users	28

Useful information

IP addresses

IPv4 / IPv6

What are the essential differences?

	IPv4	IPv6
IP configuration	• DHCP server • Manual	• Stateless Address Autoconfiguration (SLAAC): Stateless autoconfiguration using NDP (Neighbor Discovery Protocol) – Creates a link local address for every interface that does not require a router on the link. – Checks the uniqueness of the address on the link that requires no router on the link. – Specifies whether the global addresses are obtained via a status-free mechanism, a mechanism with status or via both mechanisms. (Requires a router on the link.) • Manual • DHCPv6 (status dependent)
Available IP addresses	32-bit: 4, 29 * 10^9 addresses	128-bit: 3, 4 * 10^{38} addresses
Address format	Decimal: 192.168.1.1 with port: 192.168.1.1:20	Hexadecimal: 2a00:ad80::0123 with port: [2a00:ad80::0123]:20
Loopback	127.0.0.1	::1
IP addresses of the interface	4 IP addresses	Multiple IP addresses • LLA: A link local address (formed automatically) fe80::/128 per interface • ULA: Several unique local unicast addresses per interface • GUA: Several global unicast addresses per interface
Header	• Checksum • Variable length • Fragmentation in the header • No security	• Checking at a higher layer • Fixed size • Fragmentation in the extension header
Fragmentation	Host and router	Only endpoint of the communication
Quality of service	Type of Service (ToS) for prioritization	The prioritization is specified in the header field "Traffic Class".
Types of frame	Broadcast, multicast, unicast	Multicast, unicast, anycast

	IPv4	IPv6
Identification of DHCP clients/ server	Client ID: • MAC address • DHCP client ID • System name • PROFINET station name • IAID and DUID	DUID + IAID(s) = exactly one interface of the host DUID = DHCP unique identifier Identifies server and clients uniquely and should not change, not even when replacing network components! IAID = Identity Association Identifier At least one per interface is generated by the client and remains unchanged when the DHCP client restarts Three methods of obtaining the DUID • DUID-LLT • DUID-EN • DUID-LL
DHCP	via UDP with broadcast	via UDP with unicast RFC 3315, RFC 3363 **Stateful DHCPv6** Status-dependent configuration in which the IPv6 address and the configuration settings are transferred. Four DHVPv6 messages are exchanged between client and server: 1. SOLICIT: Sent by the DHCPv6 client to localize DHCPv6 servers. 2. ADVERTISE The available DHCPv6 servers reply to this. 3. REQUEST The DHCPv6 client requests an IPv6 address and the configuration settings from the DHCPv6 server. 4. REPLY The DHCPv6 server sends the IPv6 address and the configuration settings. If the client and server support the function "Rapid commit" the procedure is shortened to two DHCPv6 messages SOLICIT and REPLY . **Stateless DHCPv6** In stateless DHCPv6, only the configuration settings are transferred. **Prefix delegation** The DHCPv6 server delegates the distribution of IPv6 prefixes to the DHCPv6 client. The DHCPv6 client is also known as PD router.
Resolution of IP addresses in hardware addresses	ARP (Address Resolution Protocol)	NDP (Neighbor Discovery Protocol)

IPv4 address

Structure of an IPv4 address

Address classes

IP address range	Max. number of networks	Max. number of hosts/network	Class	CIDR
1.x.x.x through 126.x.x.x	126	16777214	A	/8
128.0.x.x through 191.255.x.x	16383	65534	B	/16
192.0.0.x through 223.255.255.x	2097151	254	C	/24
224.0.0.0 - 239.255.255.255	Multicast applications		D	
240.0.0.0 - 255.255.255.255	Reserved for future applications		E	

An IP address consists of 4 bytes. Each byte is represented in decimal, with a dot separating it from the previous one. This results in the following structure, where XXX stands for a number between 0 and 255:

XXX.XXX.XXX.XXX

The IP address is made up of two parts, the network ID and the host ID. This allows different subnets to be created. Depending on the bytes of the IP address used as the network ID and those used for the host ID, the IP address can be assigned to a specific address class.

Subnet mask

The bits of the host ID can be used to create subnets. The leading bits represent the address of the subnet and the remaining bits the address of the host in the subnet.

A subnet is defined by the subnet mask. The structure of the subnet mask corresponds to that of an IP address. If a "1" is used at a bit position in the subnet mask, the bit belongs to the corresponding position in the IP address of the subnet address, otherwise to the address of the computer.

Example of a class B network:

The standard subnet address for class B networks is 255.255.0.0; in other words, the last two bytes are available for defining a subnet. If 16 subnets must be defined, the third byte of the subnet address must be set to 11110000 (binary notation). In this case, this results in the subnet mask 255.255.240.0.

To find out whether two IP addresses belong to the same subnet, the two IP addresses and the subnet mask are ANDed bit by bit. If both logic operations have the save result, both IP addresses belong to the same subnet, for example, 141.120.246.210 and 141.120.252.108.

Outside the local area network, the distinction between network ID and host ID is of no significance, in this case packets are delivered based on the entire IP address.

Note

In the bit representation of the subnet mask, the "ones" must be set left-justified; in other words, there must be no "zeros" between the "ones".

IPv6 addresses

IPv6 terms

Network node

A network node is a device that is connected to one or more networks via one or more interfaces.

Router

A network node that forwards IPv6 packets.

Host

A network node that represents an end point for IPv6 communication relations.

Link

A link is, according to IPv6 terminology, a direct layer 3 connection within an IPv6 network.

Neighbor

Two network nodes are called neighbors when they are located on the same link.

IPv6 interface

Physical or logical interface on which IPv6 is activated.

Path MTU

Maximum permitted packet size on a path from a sender to a recipient.

Path MTU discovery

Mechanism for determining the maximum permitted packet size along the entire path from a sender to a recipient.

LLA

Link local address FE80::/10

As soon as IPv6 is activated on the interface, a link local address is formed automatically. Can only be reached by nodes located on the same link.

ULA

Unique Local Address

Defined in RFC 4193. Via this address, the IPv6 interface can be reached in the LAN.

GUA

Global Unicast Address Via this address, the IPv6 interface can be reached, e.g. via the Internet.

Interface ID

The interface ID is formed with the EUI-64 method or manually.

EUI-64

Extended Unique Identifier (RFC 4291); method for forming the interface ID. In Ethernet, the interface ID is formed from the MAC address of the interface. Divides the MAC address into

the manufacturer-specific part (OUI) and the network-specific part (NIC) and inserts FFFE between the two parts.

Example:

MAC address = AA:BB:CC:DD:EE:FF

OUI = AA:BB:CC

NIC = DD:EE:FF

EUI-64 = OUI + **FFFE** + NIC = AA:BB:CC:**FF:FE**:DD:EE:FF

Scope

Defines the range of the IPv6 address.

Structure of an IPv6 address

IPv6 address format - notation

IPv6 addresses consist of 8 fields each with four-character hexadecimal numbers (128 bits in total). The fields are separated by a colon.

Example:

fd00:0000:0000:ffff:02d1:7d01:0000:8f21

Rules / simplifications:

- If one or more fields have the value 0, a shortened notation is possible.
  The address fd00:**0000:0000**:ffff:02d1:7d01:0000:8f21 can also be shortened and written as follows:
  fd00**::**ffff:02d1:7d01:0000:8f21
  To ensure uniqueness, this shortened form can only be used once within the entire address.
- Leading zeros within a field can be omitted.
  The address fd00:0000:0000:ffff:**02d1**:7d01:0000:8f21 can also be shortened and written as follows:
  fd00**::**ffff:**2d1**:7d01:0000:8f21
- Decimal notation with periods
  The last 2 fields or 4 bytes can be written in the normal decimal notation with periods.
  Example: The IPv6 address fd00::ffff.125.1.0.1 is equivalent to fd00::ffff:7d01:1

Structure of the IPv6 address

The IPv6 protocol distinguishes between three types of address: Unicast , anycast and multicast. The following section describes the structure of the global unicast addresses.

IPv6 prefix		Suffix
Global prefix: n bits	Subnet ID m bits	Interface ID 128 - n - m bits
Assigned address range	Description of the location, also subnet prefix or subnet	Unique assignment of the host in the network. The ID is generated from the MAC address.

The prefix for the link local address is always fe80:0000:0000:0000. The prefix is shortened and noted as follows: fe80::

IPv6 prefix

Specified in: RFC 4291

The IPv6 prefix represents the subnet identifier.

Prefixes and IPv6 addresses are specified in the same way as with the CIDR notation (Classless Inter-Domain Routing) for IPv4.

Design

IPv6 address / prefix length

Example

IPv6 address: 2001:0db8:1234::1111/48

Prefix: 2001:0db8:1234::/48

Interface ID: ::1111

Entry and appearance

The entry of IPv6 addresses is possible in the notations described above. IPv6 addresses are always shown in the hexadecimal notation.

VLAN

VLAN tagging

Expansion of the Ethernet frames by four bytes

For CoS (Class of Service, frame priority) and VLAN (virtual network), the IEEE 802.1Q standard defined the expansion of Ethernet frames by adding the VLAN tag.

Note

The VLAN tag increases the permitted total length of the frame from 1518 to 1522 bytes. The end nodes on the networks must be checked to find out whether they can process this length / this frame type. If this is not the case, only frames of the standard length may be sent to these nodes.

The additional 4 bytes are located in the header of the Ethernet frame between the source address and the Ethernet type / length field:

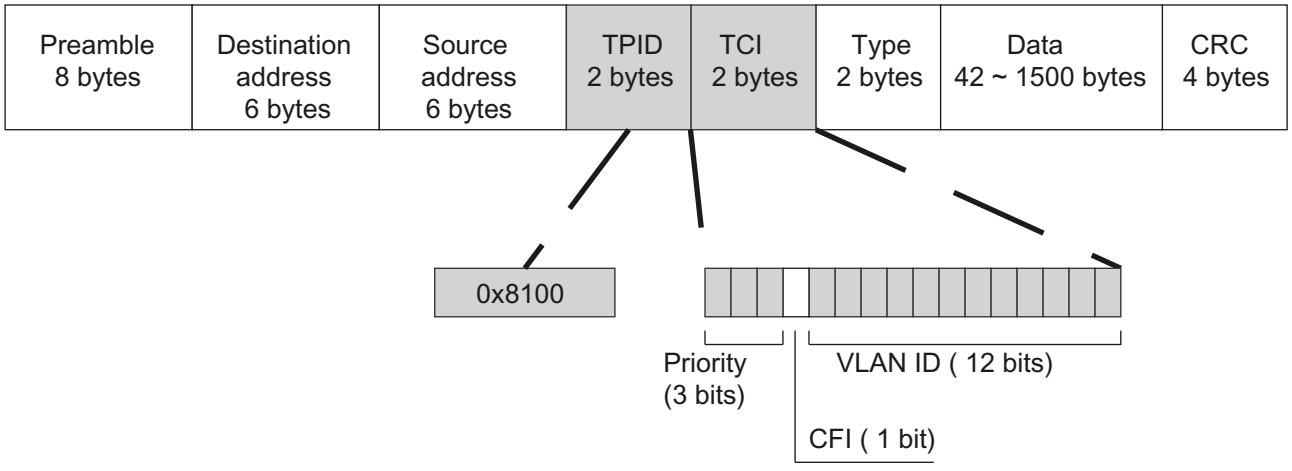

Figure 1-88 Structure of the expanded Ethernet frame

The additional bytes contain the tag protocol identifier (TPID) and the tag control information (TCI).

Tag protocol identifier (TPID)

The first 2 bytes form the Tag Protocol Identifier (TPID) and always have the value 0x8100. This value specifies that the data packet contains VLAN information or priority information.

Tag Control Information (TCI)

The 2 bytes of the Tag Control Information (TCI) contain the following information:

QoS Trust

The tagged frame has 3 bits for the priority that is also known as Class of Service (CoS), see also IEEE 802.1Q.

CoS bits	Priority	Type of the data traffic
000	0 (lowest)	Background
001	1	Best Effort
010	2	Excellent Effort
011	3	Critical Applications
100	4	Video, < 100 ms delay (latency and jitter)
101	5	Voice (language), < 10 ms delay (latency and jitter)
110	6	Internetwork Control
111	7 (highest)	Network Control

The prioritization of the data packets is possible only if there is a queue in the components in which they can buffer data packets with lower priority.

The device has multiple parallel queues in which the frames with different priorities can be processed. As default, first, the frames with the highest priority are processed. This method ensures that the frames with the highest priority are sent even if there is heavy data traffic.

Canonical Format Identifier (CFI)

The CFI is required for compatibility between Ethernet and the token Ring.
The values have the following meaning:

Value	Meaning
0	The format of the MAC address is canonical. In the canonical representation of the MAC address, the least significant bit is transferred first. Standard-setting for Ethernet switches.
1	The format of the MAC address is not canonical.

VLAN ID

In the 12-bit data field, up to 4096 VLAN IDs can be formed. The following conventions apply:

VLAN ID	Meaning
0	The frame contains only priority information (priority tagged frames) and no valid VLAN identifier.
1- 4094	Valid VLAN identifier, the frame is assigned to a VLAN and can also include priority information.
4095	Reserved

VLAN

Network definition regardless of the spatial location of the nodes

VLAN (Virtual Local Area Network) divides a physical network into several logical networks that are shielded from each other. Here, devices are grouped together to form logical groups. Only nodes of the same VLAN can address each other. Since multicast and broadcast frames are only forwarded within the particular VLAN, they are also known as broadcast domains.

The particular advantage of VLANs is the reduced network load for the nodes and network segments of other VLANs.

For the identifier which frame is assigned to which VLAN, the frame is expanded by 4 bytes (VLAN tagging). Apart from the VLAN-ID this expansion also includes priority information.

Options for the VLAN assignment

There are various options for the assignment to VLANs:

- Port-based VLAN
  Each port of a device is assigned a VLAN ID. You configure port-based VLAN in "Layer 2 > VLAN".
- Protocol-based VLAN
  Each port of a device is assigned a protocol group.
- Subnet-based VLAN
  The IP address of the device is assigned a VLAN ID.

Doubly tagged frame (Q-in-Q)

There are devices e.g. SCALANCE XR500 that support the Q-in-Q function. With the Q-in-Q function the incoming data traffic is treated as if it were untagged. With frames that are already tagged ①, this means they are expanded by a second VLAN tag, the outer VLAN tag ②.

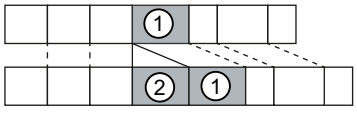

When a SCALANCE W device receives a doubly tagged frame, it uses the VLAN ID from the outer VLAN tag ②. The frame is then forwarded to the relevant VLAN.

MAC-based communication

Frames sent by the client to the access point always have the MAC address of the WLAN client as the source MAC address. In the "learning table" of the access point there is therefore only the MAC address of the WLAN client.

MAC mode "Automatic", "Manual" and "Own"

If the MAC address of a device connected to the client is adopted (Automatic) or is set manually (Manual), both the MAC-based and the IP-based frames find their destination for precisely this device. If the MAC address of the Ethernet interface of the WLAN client is used (Own), the MAC-based and IP-based frames only reach the WLAN client.

The access point checks whether the destination MAC address matches the MAC addresses of the connected clients. Since a WLAN client can only use a MAC address, communication at the MAC address level (ISO/OSI layer 2) can be to a maximum of one node downstream from the client or the client itself.

With IP Mapping, several nodes downstream from a client can be addressed based on the IP protocol. The IP packets are broken down according to an internal table and forwarded to the connected devices.

Maximum possible number of Ethernet nodes with layer 2 communication downstream from the client: 1

Notes on the "Automatic" setting:

- As long as there is no link on the Ethernet interface, the device uses the MAC address of the Ethernet interface so that it can be reached in this status. In this status, the device can be found using the Primary Setup Tool and configured with WBM or CLI.
- As soon as there is a link on the Ethernet interface, the device adopts the source MAC address of the first received frame.

Note

From the moment that the device adopts another MAC address (manually or automatically), the device no longer responds to queries of the Primary Setup Tool when the query is received over the WLAN interface. Queries of the PST over the Ethernet interface continue to be replied to.

MAC mode "Layer 2 Tunnel"

The WLAN client uses the MAC address of the Ethernet interface for the WLAN interface.

The access point is also informed of the MAC addresses connected to the Ethernet interface of the WLAN client. This makes it possible to enter the MAC addresses of these devices in the "learning table" of the access point. The access point can forward MAC-based frames for the devices downstream from the client to the appropriate client.

In much the same way as with WDS, a separate port is created for the L2T client over which the Ethernet frames are sent without changing the destination MAC address.

Maximum possible number of Ethernet nodes downstream from the client: 8

IEEE 802.11n

Overview

The standard IEEE 802.11n is an expansion of the 802.11 standard and was approved in 2009. Previous standards worked either in the 2.4 GHz frequency band (IEEE 802.11g /b) or in the 5 GHz frequency band (IEEE 802.11a). IEEE 802.11n can operate in both frequency band. In the IEEE 802.11n standard, there are mechanisms implemented in PHY and MAC layers that increase the data throughput and improve the wireless coverage.

- MIMO antenna technology
- Maximum ratio combining (MRC)
- Spatial multiplexing
- Channel bonding
- Frame aggregation

- Accelerated guard interval
- Modulation and coding scheme
- Data throughput rates up to 450 Mbps (gross)
  This is not possible on all SCALANCE W700 devices.

MIMO antenna technology

MIMO (Multiple Input - Multiple Output) is based on an intelligent multiple antenna system. The transmitter and the receiver have several spatially separate antennas. The spatially separate antennas transmit the data streams at the same time. Up to four data streams are possible. The data streams are transmitted over spatially separate paths and return over different paths due to diffraction, refraction, fading and reflection (multipath propagation). The multipath propagation means that at the point of reception a complex, space- and time-dependent pattern results as a total signal made up of the individual signals sent. MIMO uses this unique pattern by detecting the spatial position of characteristic signals. Here, each spatial position is different from the neighboring position. By characterizing the individual senders, the recipient is capable of separating several signals from each other.

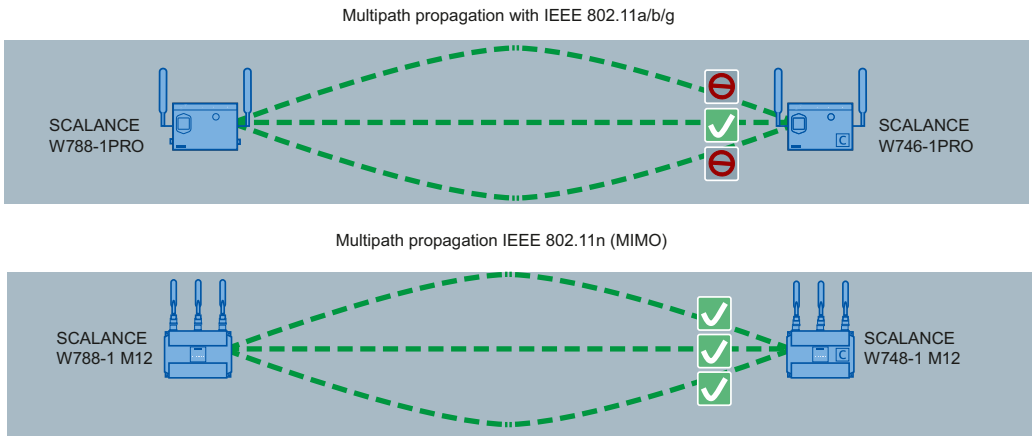

Maximum ratio combining (MRC)

In a multiple antenna system, the wireless signals are received by the individual antennas and combined to form one signal. The MRC method is used to combine the wireless signals. The MRC method weights the wireless signals according to their signal-to-noise ratio and combines the wireless signals to form one signal. The signal-to-noise ratio is improved and the error rate is reduced.

Spatial mutliplexing

With spatial multiplexing, different information is sent using the same frequency. The data stream is distributed over n transmitting antennas; in other words, each antenna sends only 1/n of the data stream. The division of the data stream is restricted by the number of antennas. At the receiver end, the signal is reconstructed.
Due to the spatial multiplexing, there is a higher signal-to-noise ratio and a higher data throughput.

Channel bonding

With IEEE 802.11n, data can be transferred via two directly neighboring channels. The two 20 MHz channels are put together to form one channel with 40 MHz. This allows the channel bandwidth to be doubled and the data throughput to be increased.
To be able to use channel bonding, the recipient must support 40 MHz transmissions. If the recipient does not support 40 MHz transmissions, the band is automatically reduced to 20 MHz. This means that IEEE 802.11n can also communicate with IEEE 802.11a/b/g devices.
The channel bundling is set on the "AP" WBM page with the "HT Channel Width [MHz]" parameter.

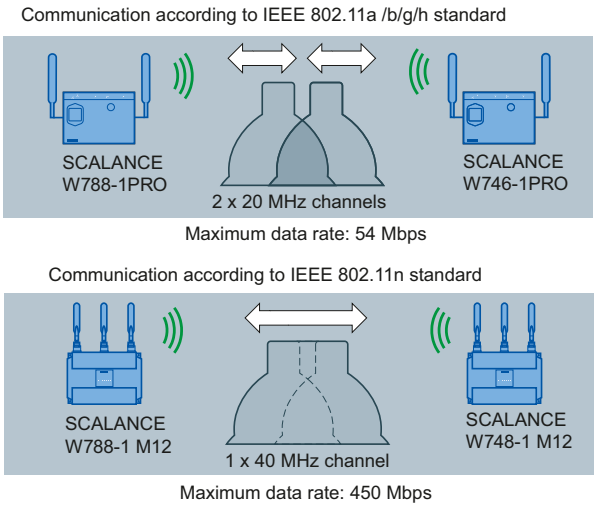

Frame aggregation

With IEEE 802.11n, it is possible to group together individual data packets to form a single larger packet; this is known as frame aggregation. There are two types of frame aggregation:

- Aggregated MAC Protocol Data Unit (A-MPDU)
  With A-MPDU, multiple MPDU data packets with the same destination address are bundled and sent as one large A-MPDU.
- Aggregated Mac Service Data Unit (A-MSDU)
  With A-MSDU, multiple MSDU data packets with the same destination address are chained together and sent.

The SCALANCE W devices support both types of frame aggregation. You make the settings on the WBM page "AP 802.11n".

Accelerated guard interval

The guard interval prevents different transmissions being mixed together. In telecommunications, this mixing is also known as intersymbol interference (ISI).
When the send time has elapsed, a send pause (guard interval) must be kept to before the next transmission begins.

The guard interval of IEEE 802.11a /b/g is 800 ns. IEEE 802.11n can use the reduced guard interval of 400 ns. You specify the guard interval on the WBM page "AP 802.11n".

Modulation and coding schemes

The IEEE 802.11n standard supports different data rates. The data rates are based on the number of spatial streams, the modulation method and the channel coding. The various combinations are described in modulation and coding schemes.

PROFINET

PROFINET

PROFINET is an open standard (IEC 61158/61784) for industrial automation based on Industrial Ethernet. PROFINET uses existing IT standards and allows end-to-end communication from the field level to the management level as well as plant-wide engineering. PROFINET also has the following features:

- Use of TCP/IP
- Automation of applications with real-time requirements
  - Real-Time (RT) communication
  - Isochronous Real-Time (IRT) communication
- Seamless integration of fieldbus systems

You configure PROFINET in "System > PROFINET (Page 2136)".

PROFINET IO

Within the framework of PROFINET, PROFINET IO is a communications concept for implementing modular, distributed applications. PROFINET IO is implemented by the PROFINET standard for programmable controllers (IEC 61158-x-10).

EtherNet/IP

EtherNet/IP

EtherNet/IP (Ethernet/Industrial Protocol) is an open industry standard for industrial real-time Ethernet based on TCP/IP and UDP/IP. With EtherNet/IP, Ethernet is expanded by the Common Industrial Protocol (CIP) at the application layer. In EtherNet/IP, the lower layers of the OSI reference model are adopted by Ethernet with the physical, network and transport functions.

You configure EtherNet/IP in "System > EtherNet/IP (Page 2137)".

Common Industrial Protocol

The Common Industrial Protocol (CIP) is an application protocol for automation that supports transition of the field buses in Industrial Ethernet and in IP networks. This industry protocol is used by field buses/industrial networks such as DeviceNet, ControlNet and EtherNet/IP at the application layer as an interface between the deterministic fieldbus world and the automation application (controller, I/O, HMI, OPC, ...). The CIP is located above the transport layer and expands the pure transport services with communications services for automation engineering. These include services for cyclic, time-critical and event-controlled data traffic. CIP distinguishes between time-critical I/O messages (implicit messages) and individual query/response frames for configuration and data acquisition (explicit messages). CIP is object-oriented; all data "visible" from the outside is accessible in the form of objects. CIP has a common configuration basis: EDS (Electronic Data Sheet).

Electronic Data Sheet

Electronic Data Sheet (EDS) is an electronic datasheet for describing devices.

The EDS required for EtherNet/IP operation can be found in "System > Load&Save".

NAT/NAPT

What is NAT?

With Network Address Translation (NAT), the IPv4 address in a data packet is replaced by another. NAT is normally used on a gateway between a private LAN and an external network with globally valid IPv4 addresses. A local IPv4 address of the internal LAN is changed to an external global IPv4 address by a NAT device at the gateway.

To translate the internal into the global IPv4 address, the NAT device maintains a translation list. The address assignment is automatic. You configure the address assignment in "Layer 3 > NAT > Basic".

What is NAPT?

In "Network Address Port Translation" (NAPT) or "Port Address Translation" (PAT), several internal source IPv4 addresses are translated into the same external source IPv4 address. To identify the individual source nodes, the port of the source device is also stored in the translation list of the NAT gateway and translated for the external address.

If several local clients send a query to the same external destination IPv4 address via the NAT gateway, the gateway enters its own external source IPv4 address in the header of these data packets. Since the forwarded data packets have the same global source IPv4 address, the NAT gateway assigns the data packets to the clients using different port number.

Note

NAT/NAPT is possible only on layer 3 of the ISO/OSI reference model. To use the NAT function, the networks must use the IP protocol.

When using the ISO protocol that operates at layer 2, it is not possible to use NAT.

If a client from the global network wants to use a service in the internal network, the translation list for the static address assignment needs to be configured. You configure the translation list for NAPT in "Layer 3 > NAT > NAPT".

iPCF / iPCF-HT / iPCF-MC

The wireless range of an IWLAN system can be expanded by using multiple access points. If a client moves from the area covered by one access point to the area covered by another access point, the wireless link is maintained after a short interruption (roaming).

If very fast update times are required, for example for PROFINET communication, access points and client modules need to be used that use the proprietary methods iPCF / iPCF-HT or iPCF-MC for fast roaming and deterministic data traffic.

iPCF / iPCF-HT / iPCF-MC can only be operated alone. A combination with each other is not possible, e.g. iPCF with iPCF-HT or iPCF-MC.

How it works

iPCF

With iPCF the access point checks all nodes in the wireless cell cyclically. At the same time, the scan includes the downlink traffic for this node. In the reply, the node sends the uplink data. The access point scans a new node at least every 5 ms.

The scan of a node is seen by all other nodes in the cell. This allows a client to detect the quality of the wireless link to the access point even when it is not communicating with the access point itself. If the client does not receive any frames from the access point for a certain time, it starts to search for a new access point.

In iPCF mode, both the search for a new access point and the registration with this access point have been optimized in terms of time. Handover times significantly below 50 ms are achieved.

Stable PROFINET communication is only possible when a WLAN client is in a wireless cell with more than 60 % or -65 dBm signal strength at all times. This can be checked by activating and deactivating the various wireless cells.

This does not mean that the client needs to change when there is a signal strength < 60 % (< -65 dBm). Make sure that access points are available with adequate signal strength.

You configure iPCF in "iFeatures > iPCF > iPCF".

iPCF-HT

If a higher data throughput is required for iPCF, iPCF-HT is used. With this you can, for example, alongside PROFINET also transfer video data. This is achieved by more effective transfer of data packets using frame-bursting (A-MPDU). The individual data packets are grouped together that are intended for the same receiver station (client) and that have the same prioritization.

You configure iPCF-HT in "iFeatures > iPCF > iPCF-HT".

iPCF-MC

For freely moving nodes that communicate independently of a RCoax cable or directional antennas, iPCF-MC should be used. With iPCF-MC, the client also searches for potentially suitable access points when it receives iPCF queries from the access point and the existing

connection to an access point is working problem-free. This means that if a change to a different access point is necessary, this is achieved extremely quickly. In contrast to iPCF, the handover times for iPCF-MC are not dependent on the number of wireless channels being used.

It is necessary to use an access point with two wireless interfaces a so-called dual access point. The one interface operates as management channel and sends short frames (beacons) with administrative information (e.g. channel setting of the data channel and SSID). The other interface (data channel) exclusively transfers the user data.

You configure iPCF-MC in "iFeatures > iPCF > iPCF-MC".

The following graphic shows a configuration example for iPCF-MC.

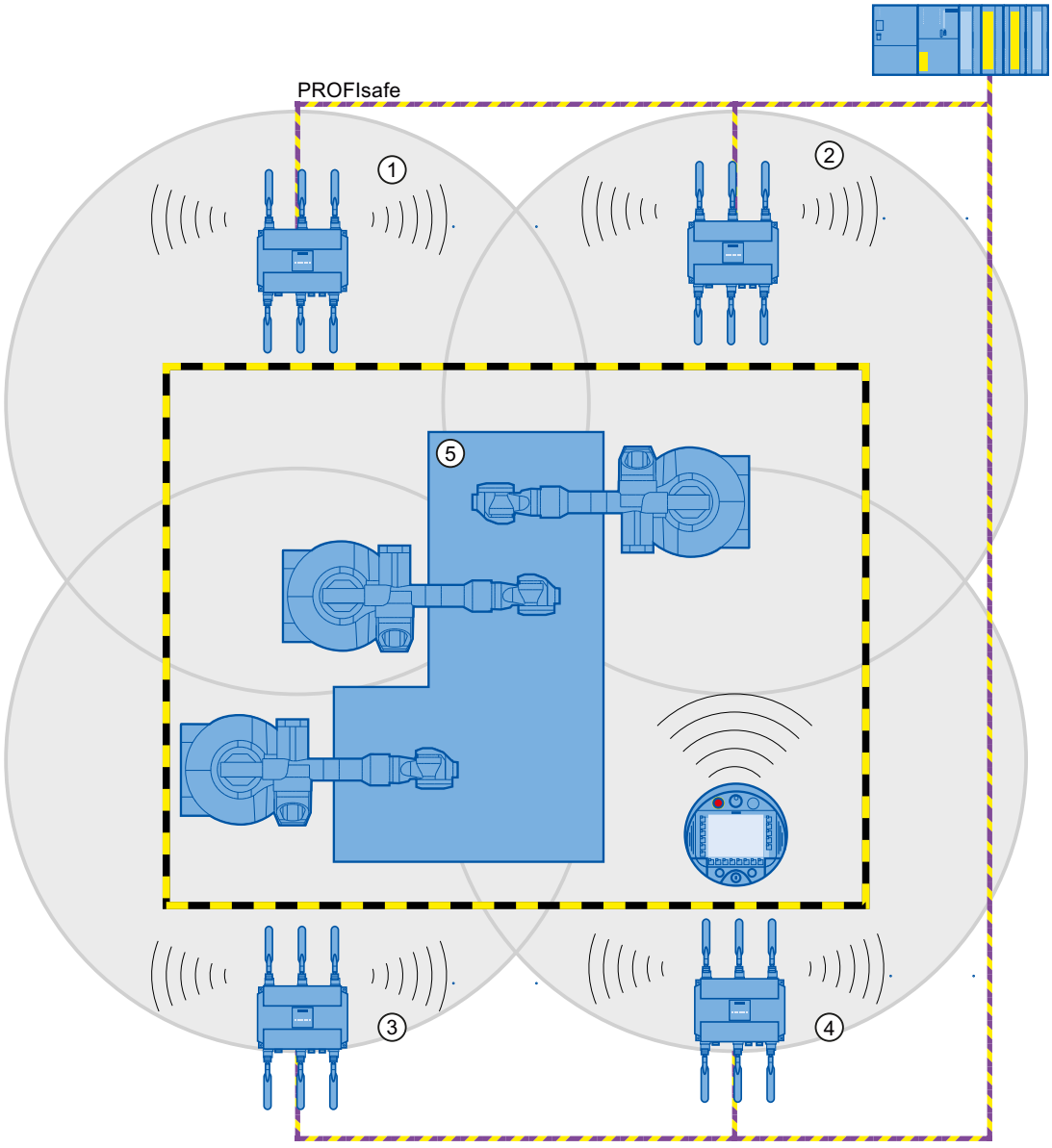

① Wireless cell of access point 1
② Wireless cell of access point 2
③ Wireless cell of access point 3
④ Wireless cell of access point 4
⑤ Plant

Restrictions

- iPCF / iPCF-HT and iPCF-MC are developments of Siemens AG and function only with nodes on which iPCF / iPCFv2 / iPCF-MC is implemented.
- With an access point with several WLAN interfaces, it is possible to use both iPCF / iPCF-HT as well as standard WLAN at the same time.

- Access points with a WLAN interface cannot take part in the iPCF-MC procedures, iPCF is, however, possible.
- iPCF-HT is available only on WLAN interface 1 and can only be used in the 5 GHz band with WLAN mode "(only) IEEE 802.11n".

Requirements for iPCF-MC

iPCF-MC uses the two wireless interface of the access point in different ways: One interface works as the management interface and sends a beacon every five milliseconds. The other interface transfers the user data.

The following requirements must be met before you can use iPCF-MC:

- Only SCALANCE W700 devices with two WLAN interfaces can be used as access points
- The data interface (WLAN1) and management interface (WLAN2) must be operated in the same frequency band and must match in terms of their wireless coverage. iPCF-MC will not work if the two wireless interfaces are equipped with directional antennas that cover different areas.
- The management interfaces of all access points to which a client can change must use the same channel. A client scans only this one channel to find accessible access points.
- Transmission based on IEEE 802.11h (DFS) cannot be used for the management interface. 802.11h (DFS) is possible for the data interface.
- A client must support this feature on its WLAN interface.

See also

iPCF (Page 2540)

iPCF-MC (Page 2545)

iREF

How it works

If an access point has several activated antennas, the transmit power is distributed equally on these antennas. The transmit power is subject to country-specific legal restrictions. The maximum permitted power depends on the gain of the connected antennas. If the connected antennas have different gains, the maximum antenna gain effectively restricts the permitted transmit power.

iREF (industrial Range Extension Function) ensures that the data traffic from the access point to each individual client is handled via the most suitable antenna. Which antenna is most suitable is determined by the access point based on the RSSI values of received packets.

Taking into account antenna gain and possible cable losses, packets are only sent via the antennas with which the maximum signal strength at the client end can be expected.

During this time the other antennas are inactive and the legally permitted transmit power is available for the selected antenna. The inactive antennas do not restrict the permitted transmit power.

In particular in applications in which MIMO cannot be used or brings no advantage, this allows data to be transmitted at the highest possible data transmission rate.

You configure iREF in "iFeatures > iREF"

without iREF

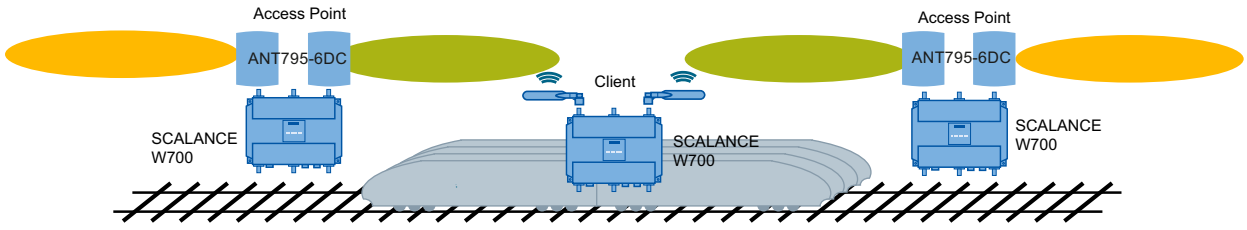

where iREF

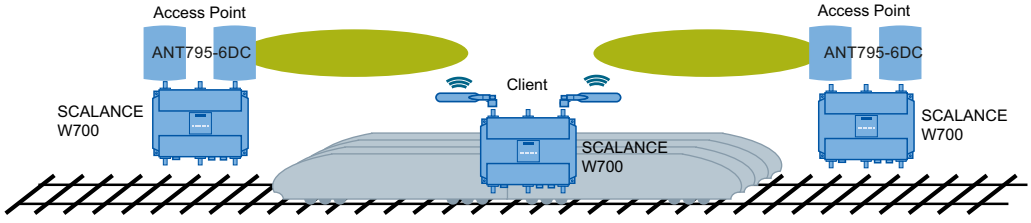

Requirement

- To be able to use iREF, the SCALANCE W700 device must have at least 2 activated antennas.

Restrictions

- A maximum data rate of only up to 150 Mbps (MCS 0 - 7 or 1 x spatial stream) is possible
- iREF cannot be used along with other iFeatures (for example iPCF or iPCF-MC)

Advantages

- Due to the directional data transmission and dynamic deactivation of antennas that do not radiate in the direction of the particular client, interference can be reduced.
- The signal strength is improved because the active antenna always has the maximum permitted transmit power available.

iPRP

The "Parallel Redundancy Protocol" (PRP) is a redundancy protocol for cabled networks. It is defined in Part 3 of the IEC 62439 standard.

With the "industrial Parallel Redundancy Protocol" (iPRP) the PRP technology can be used in wireless networks. This improves the availability of wireless communication.

How it works

A PRP network consists of two completely independent networks. If one network is disrupted, the frames are sent without interruption/reconfiguration via the parallel redundant network. To achieve this the Ethernet frames are sent to the recipient in duplicate via both networks. Devices capable of PRP have at least two separate Ethernet interfaces that are connected to independent networks.

With devices not capable of PRP a redundancy box (RedBox) is connected upstream. This allows access for so-called Single Attached Nodes (SAN) to PRP networks. The RedBox duplicates every Ethernet frame to be sent and adds a PRP trailer to the frame that among other things contains a sequence number. The RedBox simultaneously sends a copy of the frame to the PRP A and PRP B network. At the receiving end the duplicate frame is discarded by the RedBox. For this the RedBox requires certain transfer times designed for Ethernet networks. For this reason using PRP in WLAN networks results in duplicate and delayed frames.

With iPRP this problem is solved and the use of PRP in WLAN with SCALANCE W700 devices becomes possible

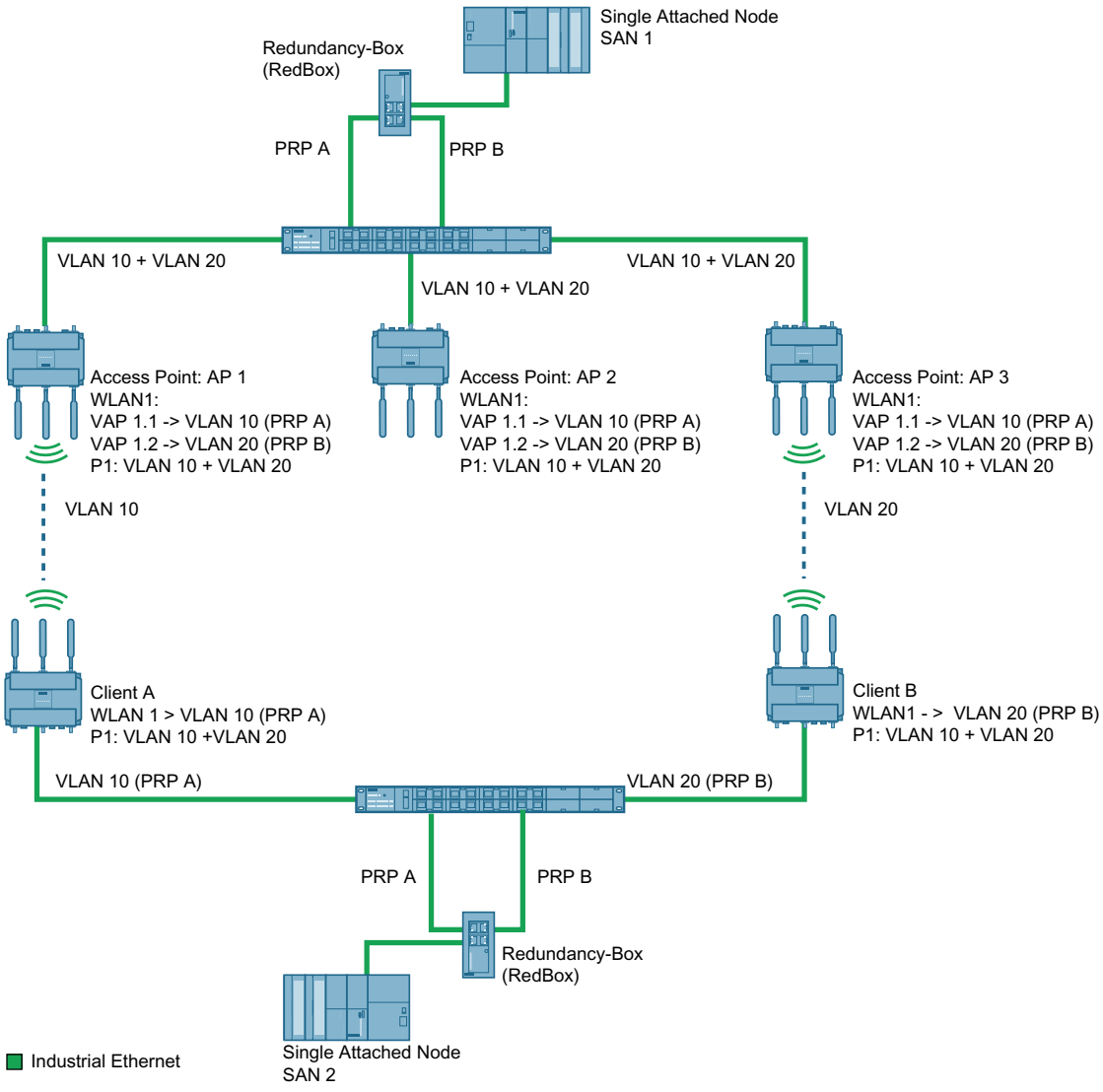

The access points (AP 1, AP 2 and AP 3) and the RedBox at the AP end are connected to each other via a switch. PRP network A und B are separated from each other via VLANs.

If SAN1 sends a frame to SAN2, the frame is duplicated by the RedBox at the AP end and the two redundant frames are transferred via the switch to the access points. Via the two different wireless paths the redundant PRP frames are transferred to the RedBox at the client end. The clients are also connected to their RedBox via a switch. This forwards the first PRP frame to arrive to SAN2 and discards the second one.

Note

On the interfaces of the switches to the SCALANCE W700 devices, only the VLANs that are also set on the VAP or WLAN interfaces of the SCALANCE W700 devices may be configured.

With iPRP the redundant partners (here: AP1 and AP3 or client A and client B) communicate with each other via a switch to prevent the two redundant PRP frames from arriving at the RedBox with too great a time difference.

If for example the communication between AP1 and client A is very slow, the slower frame is discarded at the receiving end.

You configure iPRP in "iFeatures > iPRP (Page 2547)".

Requirement

- The base bridge mode "802.1Q VLAN Bridge" is set.
- The VLANs have been created.
- Access point mode: The VAP interface is enabled.
- Client mode: In MAC mode "Layer 2 Tunnel" is set.
- Depending on the configuration the clients can communicate with every access point.

AeroScout

AeroScout tags

SCALANCE W700 devices support tags of the AeroScout company. Tags are battery-operated RFID sensors that send their data cyclically as multicast frames.

Among other things, AeroScout tags have the following features:

- **Ambient temperature**
  If a tag is fitted to a SCALANCE W700 device or material, it is possible to monitor whether a selected ambient temperature is being maintained.
- **Motion**
  Here, a tag can also supply information indicating whether it is in motion or stationary. The areas of material flow and material handling engineering represent possible applications for this function.
- **Button**
  Regardless of the frames sent cyclically, a user can also send a message by pressing a button.
- **LED**
  This provides information on the operating status of the tag.

Note

For more detailed information, please refer to the AeroScout documentation (www.aeroscout.com).

How it works

The tag sends its data as AeroScout frames. The tags and the access points communicate in the 2.4 GHz band.

If the WLAN interface of the access point receives the AeroScout frame, this is converted into a UDP datagram. The SCALANCE W700 device forwards the UDP datagram along with the

information about the signal strength (RSSI) to a PC. The AeroScout Engine runs on the PC and evaluates the received information.

Note

It is **not** advisable to use PROFINET communication and AeroScout together on one wireless interface.

Accuracy of localization

To achieve optimum precision in the localization of AeroScout Tags,

- we recommend the use of antennas with omnidirectional characteristics
- if the signals should be received by at least three access points.

SNMP

Introduction

With the aid of the Simple Network Management Protocol (SNMP), you monitor and control network components from a central station, for example routers or switches. SNMP controls the communication between the monitored devices and the monitoring station.

Tasks of SNMP:

- Monitoring of network components
- Remote control and remote parameter assignment of network components
- Error detection and error notification

In versions v1 and v2c, SNMP has no security mechanisms. Each user in the network can access data and also change parameter assignments using suitable software.

For the simple control of access rights without security aspects, community strings are used.

The community string is transferred along with the query. If the community string is correct, the SNMP agent responds and sends the requested data. If the community string is not correct, the SNMP agent discards the query. Define different community strings for read and write permissions. The community strings are transferred in plain text.

Standard values of the community strings:

- public
  has only read permissions
- private
  has read and write permissions

  Note

  Because the SNMP community strings are used for access protection, do not use the standard values "public" or "private". Change these values following the initial commissioning.

Further simple protection mechanisms at the device level:

- Allowed Host
  The IP addresses of the monitoring systems are known to the monitored system.
- Read Only
  If you assign "Read Only" to a monitored device, monitoring stations can only read out data but cannot modify it.

SNMP data packets are not encrypted and can easily be read by others.

The central station is also known as the management station. An SNMP agent is installed on the devices to be monitored with which the management station exchanges data.

The management station sends data packets of the following type:

- GET
  Request for a data record from the SNMP agent
- GETNEXT
  Calls up the next data record.
- GETBULK (available as of SNMPv2c)
  Requests multiple data records at one time, for example several rows of a table.
- SET
  Contains parameter assignment data for the relevant device.

The SNMP agent sends data packets of the following type:

- RESPONSE
  The SNMP agent returns the data requested by the manager.
- TRAP
  If a certain event occurs, the SNMP agent itself sends traps.

SNMPv1/v2c/v3 use UDP (User Datagram Protocol) and use the UDP ports 161 and 162. The data is described in a Management Information Base (MIB).

SNMPv3

Compared with the previous versions SNMPv1 and SNMPv2c, SNMPv3 introduces an extensive security concept.

SNMPv3 supports:

- Fully encrypted user authentication
- Encryption of the entire data traffic
- Access control of the MIB objects at the user/group level

With the introduction of SNMPv3 you can no longer transfer user configurations to other devices without taking special action, e.g. by loading a configuration file or replacing the C-PLUG.

According to the standard, the SNMPv3 protocol uses a unique SNMP engine ID as an internal identifier for an SNMP agent. This ID must be unique in the network. It is used to authenticate access data of SNMPv3 users and to encrypt it.

Depending on whether you have enabled or disabled the “SNMPv3 User Migration” function, the SNMP engine ID is generated differently.

Restriction when using the function

Use the "SNMPv3 User Migration" function only to transfer configured SNMPv3 users to a substitute device when replacing a device.
Do not use the function to transfer configured SNMPv3 users to multiple devices. If you load a configuration with created SNMPv3 users on several devices, these devices use the same SNMP engine ID. If you use these devices in the same network, your configuration contradicts the SNMP standard.

Compatibility with predecessor products

You can only transfer SNMPv3 users to a different device if you have created the users as migratable users. To create a migratable user the "SNMPv3 User Migration" function must be activated when you create the user.

Spanning Tree

Avoiding loops

The Spanning Tree algorithm detects redundant physical network structures and prevents the formation of loops by disabling redundant paths. It evaluates the distance and performance of a connection or bases the decisions on settings made by the user. Data is then exchanged only over the remaining connection paths.

If the preferred data path fails, the Spanning Tree algorithm then searches for the most efficient path possible with the remaining nodes.

Root bridge and bridge priority

The identification of the most efficient connection is always related to the root bridge, a network component that can be considered as a root element of a tree-like network structure. With the "Bridge Priority" parameter, you can influence the selection of the root bridge. The computer with the lowest value set for this parameter automatically becomes the root bridge. If two computers have the same priority value, the computer with the lower MAC address becomes the root bridge.

Response to changes in the network topology

If nodes are added to a network or drop out of the network, this may affect the optimum path selection for data packets. To be able to respond to such changes, the root bridge sends configuration messages (BPDUs) at regular intervals. You can set the interval between two configuration messages with the "Hello Time" parameter.

Keeping configuration information up to date

With the "Max Age" parameter, you set the maximum age of configuration information. If a bridge has information that is older than the time set in Max Age, it discards the message and initiates recalculation of the paths.

New configuration data is not used immediately by a bridge but only after the period specified in the "Forward Delay" parameter. This ensures that operation is started with the new topology only after all the bridges have the required information.

RSTP, MSTP, CIST

Rapid Spanning Tree Protocol (RSTP)

One disadvantage of STP is that if there is a disruption or a device fails, the network needs to reconfigure itself: The devices start to negotiate new paths only when the interruption occurs. This can take up to 30 seconds. Fur this reason, STP was expanded to create the "Rapid Spanning Tree Protocol" (RSTP, IEEE 802.1w). This differs from STP essentially in that the devices are already collecting information about alternative routes during normal operation and do not need to gather this information after a disruption has occurred. This means that the reconfiguration time for an RSTP controlled network can be reduced to a few seconds.
This is achieved by using the following functions:

- Edge ports (end node port)
  Edge ports are ports connected to an end device.
  A port that is defined as an edge port is activated immediately after connection establishment. If a spanning tree BPDU is received at an edge port, the port loses its role as edge port and it takes part in (R)STP again. If no further BPDU is received after a certain time has elapsed (3 x hello time), the port returns to the edge port status.
- Point-to-point (direct communication between two neighboring devices)

  By directly linking the devices, a status change (reconfiguration of the ports) can be made without any delays.
- Alternate port (substitute for the root port)

  A substitute for the root port is configured. If the connection to the root bridge is lost, the device can establish a connection over the alternate port without any delay due to reconfiguration.
- Reaction to events

  Rapid spanning tree reacts to events, for example an aborted connection, without delay. There is no waiting for timers as in spanning tree.
- Counter for the maximum bridge hops
  The number of bridge hops a package is allowed to make before it automatically becomes invalid.

In principle, therefore with rapid spanning tree, alternatives for many parameters are preconfigured and certain properties of the network structure taken into account to reduce the reconfiguration time.

Multiple Spanning Tree Protocol (MSTP)

The Multiple Spanning Tree Protocol (MSTP) is a further development of the Rapid Spanning Tree Protocol. Among other things, it provides the option of operating several RSTP instances within different VLANs or VLAN groups and, for example, making paths available within the individual VLANs that the single Rapid Spanning Tree Protocol would globally block.

Common and Internal Spanning Tree (CIST)

CIST identifies the internal instance used by the switch that is comparable in principle with an internal RSTP instance.

Configuring a SCALANCE device as an IO device

With a SCALANCE device configured as a PROFINET IO device and assigned to an IO controller, the functions "Compile" and "Download to device" only download the data to the SCALNCE device that can also be configured in Web Based Management (WBM) (System, Layer 2, Layer 3, Security).

If you want to use the "Compile" or "Download to device" functions for the PROFINET IO device data of the device, first select the assigned IO controller.

IP address assignment

Configuration options

When shipped and after restoring to the factory settings, the device does not have an IP address.

The following options are available to assign an IP address to the device:

- DHCP (default setting)
- Primary Setup Tool
  For further information, refer to the configuration manual "Primary Setup Tool". You will find the configuration manual at Siemens Industry Automation and Drives Service & Support on the Internet under the entry ID 19440762 (http://support.automation.siemens.com/WW/view/en/19440762).
- STEP 7
- CLI via the serial interface
  For further information, refer to the configuration manuals for the Command Line Interface". You will find the configuration manuals at Siemens Industry Automation and Drives Service & Support on the Internet under the following entry IDs.
  - SCALANCE W780/W740 Command Line Interface - entry ID 62515451 (http://support.automation.siemens.com/WW/view/en/62515451)
  - SCALANCE W770/W730 Command Line Interface - entry ID 89534887 (http://support.automation.siemens.com/WW/view/en/89534887)
  - SCALANCE W760/W720 Command Line Interface - entry ID 89534699 (http://support.automation.siemens.com/WW/view/en/89534699)

Requirement

- "Set IP address on device" is enabled in the properties of the device. You will find more detailed information in "Addressing PROFINET devices".

Editing properties and parameters

Editing options

You have the following options for editing properties and parameters:

- Hardware and network editor
  Once you have inserted the network component, you can edit the properties and parameters, for example the device name. You will find more detailed information in "Hardware and network editor".
- Web Based Management (WBM)
  The parameters and properties are accessible using the supplied HTML pages (WBM pages). Each WBM page has its own help page that describes the properties and parameters. For further information, refer to the configuration manuals for Web Based Management.
  - SCALANCE W780/W740 Web Based Management - entry ID: 62382125 (http://support.automation.siemens.com/WW/view/en/62382125)
  - SCALANCE W770/W730 Web Based Management - entry ID: 89535155 (http://support.automation.siemens.com/WW/view/en/89535155)
  - SCALANCE W760/W720 Web Based Management - entry ID: 89535534 (http://support.automation.siemens.com/WW/view/en/89535534)
- Command Line Interface
  All the configuration settings for the device can be made using the CLI. The CLI provides the same options as Web Based Management (WBM). For further information, refer to the configuration manuals for the Command Line Interface.
  - SCALANCE W780/W740 Command Line Interface - entry ID: 62515451 (http://support.automation.siemens.com/WW/view/en/62515451)
  - SCALANCE W770/W730 Command Line Interface - entry ID: 89534887 (http://support.automation.siemens.com/WW/view/en/89534887)
  - SCALANCE W760/W720 Command Line Interface - entry ID: 89534699 (http://support.automation.siemens.com/WW/view/en/89534699)

Availability

The availability of the settings depends

- on the device mode
- on the configuration mode
- Device

Device mode

With SCALANCE W the following device modes are available:

- Access point mode
- Client mode

Configuration mode

The following configuration modes are available:

- Offline configuration
  You can configure a PC station the first time offline. In this mode only the settings that require no connection to the device are available.
- Online configuration mode
  If there is a connection to the device and Web Based Management has started, there are additional pages available in the work area. In the online help, these pages include the note "This page is only available if there is an online connection to the device". With some settings, the additional information "only available online" is included.
  Web Based Management can be started in the following ways:
  - Select "Management" in the navigation panel and click on "Web Based Management".
  - Start the Web browser and enter the IP address or the URL of the device in the address box.

Device

Some of the settings depend on the device. The names used in the online help have the following meaning:

SCALANCE W740	The chapter described / the section described / the setting described is only relevant for the following devices: SCALANCE W748-1 RJ-45 SCALANCE W748-1 M12
SCALANCE W780	The chapter described / the section described / the setting described is only relevant for the following devices: SCALANCE W788-1 M12 SCALANCE W788-2 M12 SCALANCE W788-2 M12 EEC SCALANCE W788-1 RJ-45 SCALANCE W788-2 RJ-45 SCALANCE W786-1 RJ-45 SCALANCE W786-2 RJ-45 SCALANCE W786-2IA RJ-45 SCALANCE W786-2 SFP
SCALANCE W770	The chapter described / the section described / the setting described is only relevant for the following devices: SCALANCE W774-1 RJ-45
SCALANCE W730	The chapter described / the section described / the setting described is only relevant for the following devices: SCALANCE W734-1 RJ-45
SCALANCE W760	The chapter described / the section described / the setting described is only relevant for the following devices: SCALANCE W761-1 RJ-45
SCALANCE W720	The chapter described / the section described / the setting described is only relevant for the following devices: SCALANCE W721-1 RJ-45 SCALANCE W722-1 RJ-45

Access points	SCALANCE W788-1 M12 SCALANCE W788-2 M12 SCALANCE W788-2 M12 EEC SCALANCE W788-1 RJ-45 SCALANCE W788-2 RJ-45 SCALANCE W786-1 RJ-45 SCALANCE W786-2 RJ-45 SCALANCE W786-2IA RJ-45 SCALANCE W786-2 SFP SCALANCE W774-1 RJ-45 SCALANCE W761-1 RJ-45
Client	SCALANCE W748-1 RJ-45 SCALANCE W748-1 M12 SCALANCE W734-1 RJ-45 SCALANCE W721-1 RJ-45 SCALANCE W722-1 RJ-45

Creating and deleting an entry

As an example, an entry will be created and deleted on the Syslog client. The procedure is valid on all pages.

Creating a new entry

1. Select the device in the network or device view.
2. Open the properties of the device in the Inspector window.
3. In the Inspector window go to System > Syslog client.
4. Enter the IP address for the Syslog server.
5. Click in the table. Select the entry "New entry" in the shortcut menu. A new entry is created in the table.

Deleting an entry

1. Select the required entry in the table.
2. Select the "Delete" entry in the shortcut menu.

Buttons you require often

- **Updating the display with "Refresh"**
  Pages that display current parameters have a "Refresh" button at the lower edge of the page. Click this button to request up-to-date information from the device for the current page.
- **Save the settings for all ports with "Transfer to table"**
  Pages on which you can configure several ports have 2 tables. In the first table, you can make the settings for all ports and transfer these to the second table. In the last column of the first table, there is the "Transfer to table" button. Click this button to save the settings you have made for all ports.

Error message when downloading the configuration

Potential problems during downloading

Status messages about the progress of the download process are displayed in the output window while the configuration is downloaded to a SCALANCE device.

If problems occur during download to the device, they are usually caused by one of the following:

- Incorrect SCALANCE device in the project
- Incorrect firmware version
- The SCALANCE device is not connected to the configuration PC.

Error message

The following error message can occur during downloading when no connection can be established:

Possible errors	Possible causes	Possible remedy
Error message "The destination device currently cannot accept a download of data"	The destination device is currently in an active application relation with an IO controller.	Disable the application relation between the IO controller and the SCALANCE device.
	The TIA configuration interface on the destination device is disabled.	Enable the TIA configuration interface on the SCALANCE device via WBM or CLI.

Show information

Versions

This page shows the versions of the hardware and software of the device.

Note

The page is only available if there is an online connection to the device.

Displayed values

Table 1 contains the following columns:

- **Hardware**
  - Basic Device
    Shows the basic device.
  - WLAN X
    Shows the available WLAN interfaces. A device can have up to two WLAN interfaces.
- **Name**
  Shows the name of the device or module.
- **Revision**
  Shows the hardware version of the device.
- **Article number**
  Shows the article number of the device or described module. For the wireless card, only one version is then displayed if the WLAN interface is enabled.

Table 2 has the following columns:

- **Software**
  - Firmware
    Shows the current firmware version. If a new firmware file was downloaded and the device has not yet restarted, the firmware version of the downloaded firmware file is displayed here. After the next restart, the downloaded firmware is activated and used.
  - Bootloader
    Shows the version of the boot software stored on the device.
  - Firmware_Running
    Shows the firmware version currently being used on the device
- **Description**
  Shows the short description of the software.
- **Version**
  Shows the version number of the software version.
- **Date**
  Shows the date on which the software version was created.

I&M

Note

The page is only available if there is an online connection to the device.

This page contains information about device-specific vendor and maintenance data such as the article number, serial number, version number etc. You cannot configure anything on this page.

Description of the displayed values

The table has the following rows:

- **Manufacturer ID**
  Shows the manufacturer ID.
- Article number
  Shows the article number.
- **Serial number**
  Shows the serial number.
- **Hardware revision**
  Shows the hardware version.
- **Software version**
  Shows the software version.
- **Revision counter**
  Shows the revision counter: Counter for revisions since the initial commissioning
- **Revision Date**
  Date and time of the last revision
- **Function tag**
  Shows the function tag (plant designation) of the device. The plant designation (HID) is created during configuration of the device with HW Config of STEP 7.
- **Location tag**
  Shows the location tag of the device. The location identifier (LID) is created during configuration of the device with HW Config of STEP 7.
- **Date**
  Shows the date created during configuration of the device with HW Config of STEP 7.
- **Description**
  Shows the description created during configuration of the device with HW Config of STEP 7.

ARP / neighbor table

IPv6 Neighbor Table

Assignment of MAC address and IPv6 address

Via the IPv6 neighbor table, there is a unique assignment of MAC address to IPv6 address. This assignment is kept by each network node in its own separate neighbor table.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- Interface
  Displays the interface via which the row entry was learnt.
- MAC Address
  Shows the MAC address of the destination or source device.
- IP Address
  Shows the IPv6 address of the destination device.
- Media Type
  Shows the type of connection.
  - Dynamic
    The device recognized the address data automatically.
  - Static
    The addresses were entered as static addresses.

ARP table

Note

The page is only available if there is an online connection to the device.

Assignment of MAC address and IP address

With the Address Resolution Protocol (ARP), there is a unique assignment of MAC address to IPv4 address. This assignment is kept by each network node in its own separate ARP table. The page shows the ARP table of the device.

Description of the displayed values

The table has the following columns:

- Interface
  Shows the interface via which the row entry was learnt.
- MAC Address
  Shows the MAC address of the destination or source device.
- IP address
  Shows the IP address of the target device.
- Media Type
  Shows the type of connection.
  - Dynamic
    The device recognized the address data automatically.
  - Static
    The addresses were entered as static addresses.

Log tables

Event log

Logging events

Note

The page is only available if there is an online connection to the device.

The device allows you to log events that occur, some of which you can specify in " System > Events". This, for example, allows you to record when an authentication attempt failed or when the connection status of a port has changed. The content of the events log table is retained even when the device is turned off.

Description of the displayed values

- Severity Filters

  Note

  For each severity, a maximum of 400 entries in the table are possible. If the maximum number of entries is reached for a severity, the oldest entries of this severity are overwritten in the table. The table remains permanently in the memory

  You can filter the entries in the table according to severity. Select the required entries in the check boxes above the table.

  – Info
    Information

  – When this parameter is enabled, all entries of the category "Info" are displayed.

  – Warning
    Warning
    When this parameter is enabled, all entries of the category "Warning" are displayed.

  – Critical
    Critical
    When this parameter is enabled, all entries of the category "Critical" are displayed.

  The table has the following columns:

  – Restart
    Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.

  – System Up Time
    Shows the time the device has been running since the last restart when the described event occurred.

  – System Time
    Shows the date and time when the described event occurred.
    If the system time is set, the time is also displayed at which the event occurred.

  – Severity
    Shows the severity of the message.

  – Log Message
    Displays a brief description of the event that has occurred. You will find the list of possible messages in Appendix D of the configuration manual.

WLAN authentication log

Logging authentication attempts

This page shows a table with information on successful or failed authentication attempts.

Note

The page is only available if there is an online connection to the device.

Displayed values

- Severity Filters
  You can filter the entries in the table according to severity. To display all the entries, enable or disable all parameters.
  - Info
    Information
    When this parameter is enabled, all entries of the category "Info" are displayed.
  - Warning
    Warnings
    When this parameter is enabled, all entries of the category "Warning" are displayed.
  - Critical
    Critical
    When this parameter is enabled, all entries of the category "Critical" are displayed.

The table has the following columns:

- Restart
  Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.
- System Up Time
  Shows the time the device has been running since the last restart when the described fault occurred.
- System Time
  Shows the time at which the described error occurred.
- Severity
  Shows the severity of the message.
- Log Message
  Displays a brief description of the event that has occurred. You will find the list of possible messages in Appendix D of the configuration manual.

If the system time is set, the time is also displayed at which the event occurred.

Buttons

- Clear
  Click this button to delete the content of the log file. All entries are deleted regardless of what you have selected under "Severity Filters".
  The display is also cleared. The restart counter is only reset after you have restored the device to the factory settings and restarted the device.

  Note

  For each severity a maximum of 400 entries in the table are possible. If the maximum number of entries is reached for a severity, the oldest entries of this severity are overwritten in the table. The table remains permanently in memory.

- **Show all**
  Click this button to display all the entries on the WBM page. Note that displaying all messages can take some time.
- **Next**
  Click this button to go to the next page.
- **Prev**
  Click this button to go to the previous page.

Fault

Error status

if an error occurs, it is shown on this page. On the device errors are indicated by the fault LED lighting up.

Internal errors of the device and the errors that you configure on the following pages are indicated:

- "System > Events"
- "System" > Fault Monitoring"

The calculation of the time of an error always begins after the last system start.

If there are no errors present, the fault LED switches off.

Note

The page is only available if there is an online connection to the device.

Displayed values

- **No. of Signaled Faults**
  Indicates how often the fault LED lit up and not how many faults occurred.
- **"Reset counters" button**
  The number is reset with this button. The counter is reset when there is a restart.

The table contains the following columns:

- **Fault Time**
  Shows the time the device has been running since the last restart when the described fault occurred.
- **Fault Description**
  Displays a brief description of the error that has occurred.
- **Clear Fault State**
  If the "Clear Fault State" button is enabled, you can delete the error, e.g. error of the event "Cold/Warm Start".

Spanning Tree

Note

The page is only available if there is an online connection to the device.

The page shows the current information about the Spanning Tree and the settings of the root bridge.

- If spanning tree is turned off, only the basic information about this device is displayed.
- If Spanning Tree is turned on, the status information about the selected instance is displayed and the information about the configured ports is shown in the table. The information depends on the selected Spanning Tree mode.

Displayed values

- **Spanning Tree Mode**
  Shows the set mode. You specify the mode in "Layer 2 > MSTP > General".
  The following values are possible:
  - '-'
  - STP
  - RSTP
  - MSTP
- **Instance ID**
  Shows the number of the instance. The parameter depends on the configured mode.
- **Bridge Priority / Root Priority**
  Which device becomes the root bridge is decided based on the bridge priority. The bridge with the highest priority (in other words, with the lowest value for this parameter) becomes the root bridge. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames. The value for the bridge priority is a whole multiple of 4096 with a range of values from 0 to 32768.
- **Bridge Address / Root Address**
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Cost**
  Shows the path costs from the device to the root bridge.
- **Bridge Status**
  Shows the status of the bridge, e.g. whether or not the device is the root bridge
- **Regional root priority** (available only with MSTP)
  For a description, see Bridge Priority / Root Priority

- **Regional root address** (available only with MSTP)
  Shows the MAC address of the device.
- **Regional root costs** (available only with MSTP)
  Shows the path costs from the regional root bridge to the root bridge.

The table has the following columns:

- **Port**
  Shows the port via which the device communicates.
- **Role**
  Shows the status of the port. The following values are possible:
  - Disabled
    The port was removed manually from the spanning tree and will no longer be taken into account by the spanning tree.
  - Designated
    The ports leading away from the root bridge.
  - Alternate
    The port with an alternative route to a network segment
  - Backup
    If a switch has several ports to the same network segment, the "poorer" Port becomes the backup port.
  - Root
    The port that provides the best route to the root bridge.
  - Master
    This port points to a root bridge located outside the MST region.
- **Status**
  Displays the current status of the port. The values are only displayed. The parameter depends on the configured protocol. The following statuses are possible:
  - Discarding
    The port receives BPDU frames. Other incoming or outgoing frames are discarded.
  - Listening
    The port receives and sends BPDU frames. The port is involved in the spanning tree algorithm. Other outgoing and incoming frames are discarded.
  - Learning
    The port actively learns the topology; in other words, the node addresses. Other outgoing and incoming frames are discarded.
  - Forwarding
    Following the reconfiguration time, the port is active in the network. The port receives and sends data frames.
- **Oper. Version**
  Describes the type of spanning tree in which the port operates
- **Priority**
  If the path calculated by the spanning tree is possible over several ports of a device, the port with the highest priority (in other words the lowest value for this parameter) is selected. A value between 0 and 240 can be entered for the priority in steps of 16. If you enter a value that cannot be divided by 16, the value is automatically adapted.

- **Path costs**
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected. If several ports of a device have the same value, the port with the lowest port number will be selected.
  If the value "Cost Calc." is "0", the automatically calculated value is shown. Otherwise, the value of "Cost Calc." is displayed.
  The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000.
- **Edge type**
  Shows the type of the connection. The following values are possible:
  - Edge Port

    An edge port is connected to this port.
  - No Edge Port
    There is a spanning tree or rapid spanning tree device at this port.
- **P.t.P. Type**
  Shows the type of point-to-point link. The following values are possible:
  - P.t.P.
    With half duplex, a point-to-point link is assumed.
  - Shared Media

    With a full duplex connection, a point-to-point link is not assumed.

Note

Point-to-point connection means a direct connection between two devices.

A shared media connection is, for example, a connection to a hub.

Ethernet Statistics

Interface statistics

Information from the MIB

The page shows the statistics from the interface table of the Management Information Base (MIB).

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **In Octet**
  Shows the number of received bytes.
- **Out Octet**
  Shows the number of sent bytes.
- **In Unicast**
  Shows the number of received unicast frames.
- **In Non Unicast**
  Shows the number of received frames that are not of the type unicast.
- **Out Unicast**
  Shows the number of sent unicast frames.
- **Out Non Unicast**
  Shows the number of sent frames that are not of the type unicast.
- **In Errors**
  Shows the number of all possible RX errors, refer to the tab "Packet Error".

Description of the button

Reset Counter

Click the "Reset Counter" button to reset all counters. The counters are also reset by a restart.

Frame length

Frames sorted by length

This page displays how many frames of which length were sent and received at each port. You cannot configure anything on this page.

Note

The page is only available if there is an online connection to the device.

The displayed values are transferred by RMON.

On the page "Layer 2 > RMON > Statistics", you can set the ports for which values will be displayed.

Description of the displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.

  Note
  Display of frame statistics

  In the statistics relating to frame lengths, note that both incoming and outgoing frames are counted.

- **Frame lengths**
  The other columns after the port number contain the absolute numbers of frames according to their frame length.
  The following frame lengths are distinguished:
  - 64 bytes
  - 65 - 127 bytes
  - 128 - 255 bytes
  - 256 - 511 bytes
  - 512 - 1023 bytes
  - 1024 - max.

Note
Data traffic on blocked ports

For technical reasons, data packets can be indicated on blocked ports.

Description of the button

Reset Counter

Click the "Reset Counter" button to reset all counters. The counters are also reset by a restart.

Frame type

Received frames sorted by type

This page displays how many frames of the type "unicast", "multicast", and "broadcast" were received at each port.

You cannot configure anything on this page.

The displayed values are transferred by RMON. On the page "Layer 2 > RMON > Statistics", you can set the ports for which values will be displayed.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Unicast / Multicast / Broadcast**
  The other columns after the port number contain the absolute numbers of the incoming frames according to their frame type "unicast", "multicast" and "broadcast".

Description of the button

Reset Counter

Click the "Reset Counter" button to reset all counters. The counters are also reset by a restart.

Packet Errors

Received bad frames

This page shows how many bad frames were received per port.

You cannot configure anything on this page.

The displayed values are transferred by RMON. On the page "Layer 2 > RMON > Statistics", you can set the ports for which values will be displayed.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations.
- **Error types**
  The other columns after the port number contain the absolute numbers of the incoming frames according to their error type.
  In the columns of the table, a distinction is made according to the following error types:
  - CRC

    Packets whose content does not match the CRC checksum.
  - Undersize
    Packets with a length less than 64 bytes.
  - Oversize
    Frames discarded because they were too long.
  - Fragments

    Packets with a length less than 64 bytes and a bad CRC checksum.
  - Jabbers
    VLAN-tagged packets with an incorrect CRC checksum that were discarded because they were too long.
  - Collisions
    Collisions that were detected.

Description of the button

Reset Counter

Click the "Reset Counter" button to reset all counters. The counters are also reset by a restart.

Learning table

Address filtering

This page shows the current content of the learning table. This table lists the source addresses of unicast address frames.

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- **VLAN-ID**
  Shows the VLAN ID of the node.

  Note

  This column appears in the table only if a VLAN is configured.

- **MAC Address**
  Shows the MAC address of the node.
- **Status**
  Shows the status of each address entry:
  - Dynamic
    The specified address was learned by receiving a frame from this node and will be deleted when the aging time expires if no further frames are received from this node.
  - Invalid
    These values are not evaluated.
- **Port**
  Shows the port via which the node with the specified address can be reached. Frames received by the device whose destination address matches this address will be forwarded to this port.

Buttons

- **Show all**
  Click this button to display all the entries on the page. Note that displaying all messages can take some time.
- **Next**
  Click this button to go to the next page.
- **Prev**
  Click this button to go to the previous page.

Drop-down list for page change

From the drop-down list, select the page you want to go to.

IPv6 routing table

Overview

This page shows the IPv6 routes currently being used.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The table has the following columns:

- **Destination Network**
  Shows the destination address of this route.
- **Prefix Length**
  Shows the prefix length of this route.
- **Gateway**
  Shows the gateway for this route.
- **Interface**
  Shows the interface for this route.
- **Metric**
  Shows the metric of the route. The higher value, the longer packets require to their destination.
- **Routing Protocol**
  Shows the routing protocol from which the entry in the routing table originates. The following entries are possible:
  - connected: Connected routes
  - Static: Static routes
  - RIPng: Routes via RIPng
  - OSPFv3: Routes via OSPFv3
  - other: Other routes

DHCP Server

This page shows whether IPv4 addresses were assigned to the devices by the DHCP server.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

- **IP Address**
  Shows the IPv4 address assigned to the DHCP client.
- **Pool ID**
  Shows the number of the IPv4 address band.
- **Identification method**
  Shows the method according to which the DHCP client is identified.
- **Identification value**
  Shows the MAC address or he client ID of the DHCP client.
- **Assignment method**
  Shows whether the IPv4 address was assigned statically or dynamically. You configure the static entries in "System > DHCP > Static Leases".
- **Assignment status**
  Shows the status of the assignment.
  - Assigned
    The assignment is used.
  - Not used
    The assignment is not used.
  - Probing
    The assignment is being checked.
  - Unknown
    The status of the assignment is unknown.
- **Expire Time**
  Shows until when the assigned IPv4 address is still valid. Up to this time, the DHCP client must either request a new IPv4 address or extend the lease time of the assigned IPv4 address.

SNMP

Note

The page is only available if there is an online connection to the device.

This page displays the created SNMPv3 groups. You configure the SNMPv3 groups in "System" > SNMP"..

Description of the displayed values

The table has the following columns:

- **Group Name**
  Shows the group name.
- **User Name**
  Shows the user that is assigned to the group.

Security

Overview

This page shows the security settings and the local and external user accounts.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the rights of the logged-on user.

Description of the displayed values

Services

The "Services" list shows the security settings.

- **SSH Server**
  You configure the setting in "System > Configuration".
  - Enabled: Encrypted access to the CLI.
  - Disabled: No encrypted access to the CLI.
- **Web Server**
  You configure the setting in "System > Configuration".
  - HTTP/HTTPS: Access to the WBM is possible with HTTP and HTTPS.
  - HTTPS: Access to the WBM is now only possible with HTTPS.
- **SNMP**
  You can configure setting in "System > SNMP > General".
  - "-" (SNMP disabled)
    Access to device parameters via SNMP is not possible.
  - SNMPv1/v2c/v3
    Access to device parameters is possible with SNMP versions 1, 2c or 3.
  - SNMPv3
    Access to device parameters is possible only with SNMP version 3.
- **Management ACL**
  You configure the setting in "Security > Management ACL".
  - Disabled: No access restriction
    The access control is disabled.
  - Enabled: No access restriction
    The access control is enabled but no access rules have been defined.
  - Enabled: Restricted access only
    The access control is enabled and access rules have been defined.

- **Login Authentication**
  You configure the setting in "Security > AAA > General".
  - Local
    The authentication must be made locally on the device.
  - RADIUS
    The authentication must be handled via a RADIUS server.
  - Local and RADIUS
    The authentication is possible both with the users that exist on the device (user name and password) and via a RADIUS server.
    The user is first searched for in the local database. If the user does not exist there, a RADIUS query is sent.
  - RADIUS and fallback local
    The authentication must be handled via a RADIUS server.
    A local authentication is performed only when the RADIUS server cannot be reached in the network.
- **Password Policy**
  Shows which password policy is currently being used.

Local and external user accounts

You configure local user accounts and roles in "Security > User Accounts"

When you create a local user account an external user account is generated automatically. Local user accounts involve users each with a password for logging in on the device.

In the table "External User Accounts" a user is linked to a role, for example the user "Observer" with the role "user" In this case, the user is defined on a RADIUS server. The roll is defined locally on the device. When a RADIUS server authenticates a user, the corresponding group however is unknown or does not exist, the device checks whether or not there is an entry for the user in the table "External User Accounts". If an entry exists, the user is logged in with the rights of the associated role. If the corresponding group is known on the device, both tables are evaluated. The user is assigned the role with the higher rights.

Note

The table "External User Accounts" is only evaluated if you have set "Vendor Specific" in the RADIUS Authorization Mode".

With CLI you can access external user accounts.

The table "Local User Accounts" has the following columns:

- **User Account**
  Shows the name of the local user.
- **Role**
  Shows the role of the user. You can obtain more information on the function rights of the role in "Information > Security > Roles".

Supported Function Rights

The page shows the function rights available locally on the device.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the role of the logged-on user.

Description of the displayed values

- **Function Right**
  Shows the number of the function right. Different rights relating to the device parameters are assigned to the numbers.
- **Description**
  Shows the description of the function right.

Groups

This page shows which group is linked to which role. The group is defined on a RADIUS server. The roll is defined locally on the device.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the role of the logged-on user.

Description of the displayed values

The table has the following columns:

- **Group**
  Shows the name of the group. The name matches the group on the RADIUS server.
- **Role**
  Shows the role. Users who are authenticated with the linked group on the RADIUS server receive the rights of this role locally on the device.
- **Description**
  Shows a a description for the link.

Roles

The page shows the roles valid locally on the device.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the role of the logged-on user.

Description of the displayed values

This table contains the following columns:

- **Role**
  Shows the role.
- **Function Right**
  Shows the function rights of the role:
  - 1
    Users with this role can read device parameters but cannot change them.
  - 15
    Users with this role can both read and change device parameters.
  - 0
    This is a role that the device assigns internally when the user could not be authenticated. The user is denied access to the device.
- **Description**
  Shows a description of the role.

Inter AP blocking

the page shows a list of the devices with which the clients are allowed to communicate.

Note

The page is only available

- with SCALANCE W780 / W770
- if there is an online connection to the device.
- in access point mode.
- with KEY-PLUG inserted: W780 iFeatures (MLFB 6GK5 907-8PA00) or W700 Security (MLFB 6GK5907-0PA00)

Settings

The table has the following columns:

- Radio
  Shows the available WLAN interfaces to which the settings relate.
- Port
  Shows the VAP interface to which the settings relate.
- MAC address
  Shows the MAC address of the device with which the client may communicate.
- IP address
  Shows the IP address of the device with which the client may communicate.
- IP address resolution
  Shows the IP address with which the permitted IP address is resolved.

WLAN

AP overview

Overview of the configuration

This page shows these settings/properties of the WLAN or the WLAN interface.

Note

The page is only available

- if there is an online connection to the device.
- in access point mode.

Displayed values

Table 1 has the following columns:

- Radio
  Shows the available WLAN interfaces.
- Port
  Shows the available VAP interfaces.
- WLAN Mode
  Shows the transmission standard. If DFS is activated, the transmission standard "802.11h" is not shown additionally but only the configured transmission standard "802.11a".
- Configured Channel
  Shows the configured channel. If "Auto" is displayed, the access point searches for a free channel itself.

- **Alternative DFS Channel**
  If the DFS function is enabled, the configured alternative channel of the access point is displayed.
  If "Auto" is displayed, the access point searches for an alternative channel itself.
  If the DFS function is activated and the access point browses for primary users for 60 seconds before starting communication with the selected channel, the text "scanning ..." is displayed instead of the channel.
- **Operational channel**
  Shows the channel of the access point via which the access point communicates.
- **HT Channel Width**
  Shows the channel bandwidth.
  - 20
    Channel bandwidth 20 MHz
  - 40 up
    Channel bandwidth 40 MHz. The configured channel and the neighboring channel above it are used.
  - 40 down
    Channel bandwidth 40 MHz. The configured channel and the neighboring channel below it are used.

  Note

  Channel bandwidth 40 MHz and frequency band 2.4 GHz

  If the access point detects another access point on the configured channel or on neighboring channels, the access point changes the channel bandwidth from 40 MHz to 20 MHz. If you set a "free" channel on the access point, the access point uses the channel bandwidth 40 MHz.

- **iFeatures**
  Shows which iFeatures are used.
  - -
    iFeatures are not used.
  - iPCF
  - iPCF-MC
  - iREF
- **Status**
  Shows the status of the WLAN interface.
  - enabled
    The WLAN interface is enabled.
  - disabled
    The WLAN interface is disabled.

Table 2 has the following columns:

- Radio
  Shows the available WLAN interfaces in this column.
- Port
  Shows the port of the virtual access point.
- MAC Address
  Shows the MAC address of the virtual access point.
- SSID
  Shows the SSID.
- Security
  Shows which authentication method is used.
  If the authentication method "Open System + Encryption" or "Shared Key" is used, the "Encrypted (WEP/AES)" authentication method is displayed for both.
- Status
  Shows the status of the WLAN interface.
  - enabled
    The WLAN interface is enabled.
  - disabled
    The WLAN interface is disabled.

Client list

Logged-on clients

The page the clients logged on with the access point as well as additional information, for example status, signal strength, MAC address.

Note

The page is only available

- if there is an online connection to the device.
- in access point mode.

Displayed values

- Logged-on Clients
  Shows the number of clients logged on to the access point.

The table has the following columns:

- AID (Associated ID)
  Shows the connection ID of the client. If the client connects to the access point via the VAP interface, the client is assigned a connection ID. The connection ID is unique within a VAP interface. If two clients log on at different VAP interfaces, both clients can receive the same ID.
- Radio
  Shows the available WLAN interfaces.
- Port
  Shows the VAP interface.
- Type
  Shows the client type, for example "Sta" stands for IEEE 802.11 standard client.
- MAC Address
  Shows the MAC address of the client.
- System Name
  Shows the system name of the client if the client communicates this to the access point. Not all clients support this parameter.
- Channel
  Shows the channel over which the client communicates with the access point.
- Signal Strength [dBm]
  Shows the signal strength of the connected client in bBm.
- Signal Strength [%]
  Shows the signal strength of the connected client as a percentage.
- Age [s]
  Shows the time that has elapsed since the last client activity.
- Security
  Shows which authentication method is used.
  If the authentication method "Open System + Encryption" or "Shared Key" is used, the "Encrypted (WEP/AES)" authentication method is displayed for both.
- WLAN Mode
  Shows the transmission standard. If DFS is activated, the transmission standard "802.11h" is not shown additionally but only the configured transmission standard "802.11a".
- Maximum Data Rate [Mbps]
  Shows the maximum data rate in megabits per second.
- Status
  Shows the current status of the connection, for example connected means that the client is connected to the AP and is ready to communicate with the AP.

WDS list

Communication between access points

In normal operation, the access point is used as an interface to a network and communicates with clients. There are, however, situations in which several access points need to communicate with each other, for example to extend wireless coverage or to set up a wireless backbone. This mode is possible with WDS (Wireless Distributed System).

Note

The page is only available

- if there is an online connection to the device.
- in access point mode.

This page shows information about the WDS connections of the access point.

Displayed values

The table has the following columns:

- **Radio**
  Shows the available WLAN interfaces.
- **Port**
  Shows the port.
- **BSSID**
  Shows the MAC address of the WDS partner.
- **WDS ID**
  Shows the name of the WDS partner.
- **Channel**
  Shows the channel over which the access point communicates with the WDS partner.
- **Signal Strength [dBm]**
  Shows the signal strength of the connected access point in bBm.
- **Signal Strength [%]**
  Shows the signal strength of the connected access point as a percentage.
- **Security**
  Shows which authentication method is used.
  If the authentication method "Open System + Encryption" or "Shared Key" is used, the "Encrypted (WEP/AES)" authentication method is displayed for both.
- **Max. Data Rate [Mbps]**
  Shows the maximum data rate in megabits per second.
- **Status**
  Shows the current status of the WDS connection.

AP overlap

Overlapping channels

Note

The page is only available

- if there is an online connection to the device.
- in access point mode.

For optimum data throughput, it is important that the set wireless channel is not used by other access points. In the 2.4 GHz band (802.11b or 802.11g), there is overlapping of the channels so that an access point occupies not only the set channel but also the two or three adjacent channels. You should therefore make sure that there is adequate channel spacing to neighboring access points.

This page shows all access points that are visible on the set or adjacent channels (at 2.4 GHz). If entries exist here, the maximum data throughput of the access point and the availability of the communication link to the access point is potentially impaired.

Displayed values

Table 1 has the following columns:

- **Radio**
  Shows the available WLAN interfaces.
- **Aging Time [min]**
  Specify the life time of the entries in the list. If an access point is inactive for longer than the set time, it is removed from the list.

  Note

  Changing the aging time

  The aging time is a WLAN setting. For this reason, if a change is made, the WLAN connection is briefly interrupted to accept the new value.

The table has the following columns:

- **Radio**
  Shows the available WLAN interfaces in this column.
- **Type**
  Shows the mode of the WLAN interface.
- **SSID**
  Shows the SSID of the access point.
- **BSSID**
  Shows the MAC address of the access point.
- **System Name**
  Shows the system name of the device. The entry depends on the access point. Not all access points support this parameter.

- Channel
  Shows the channel over which the client communicates with the access point.
- Signal Strength [dBm]
  Shows the signal strength of the client in bBm.
- Signal Strength [%]
  Shows the signal strength of the client as a percentage.
- Age [s]
  Shows the time that has elapsed since the last access point activity.
- Security
  Shows which authentication method is used.
  If the authentication method "Open System + Encryption" or "Shared Key" is used, the "Encrypted (WEP/AES)" authentication method is displayed for both.
- WLAN Mode
  Shows the transmission standard. If DFS is activated, the transmission standard "802.11h" is not shown additionally but only the configured transmission standard "802.11a" or "802.11n".

Client overview

Overview of the configuration

The page shows an overview of the existing clients and their configuration.

Note

The page is only available

- if there is an online connection to the device.
- for clients or access points in client mode.

Displayed values

- Radio
  Shows the available WLAN interfaces in this column.
- WLAN Mode
  Shows the transmission standard.

- MAC Mode
  Shows how the MAC address is assigned to the interface.
  - Automatic
    The client automatically adopts the source MAC address of the first frame that it receives over the Ethernet interface.
  - Manual
    The address was entered manually
  - Own
    The client uses the MAC address of the Ethernet interface for the WLAN interface.
  - Layer 2 Tunnel
    The client uses the MAC address of the Ethernet interface for the WLAN interface. The network is also informed of the MAC addresses connected to the Ethernet interface of the client. Up to eight MAC addresses can be used.
- MAC Address
  Shows the MAC address of the WLAN interface.
- Operational channel
  Shows the channel of the access point with which the client is connected.
- HT Channel Width [MHz]
  Shows the channel bandwidth.
  - 20
    Channel bandwidth 20 MHz
  - 40 up
    Channel bandwidth 40 MHz. The configured channel and the neighboring channel above it are used.
  - 40 down
    Channel bandwidth 40 MHz. The configured channel and the neighboring channel below it are used.

    Note

    Channel bandwidth 40 MHz and frequency band 2.4 GHz

    If the access point detects another access point on the configured channel or on neighboring channels, the access point changes the channel bandwidth from 40 MHz to 20 MHz. If you set a "free" channel on the access point, the access point uses the channel bandwidth 40 MHz.
- Connected BSSID
  This box shows the MAC address of the access point with which the client is connected.
- Connected SSID
  This box shows the SSID of the access point with which the client is connected.
- Security
  Shows which authentication method is used.
  If the authentication method "Open System + Encryption" or "Shared Key" is used, the "Encrypted (WEP/AES)" authentication method is displayed for both.

- **iFeatures**
  Shows which iFeatures are used.
  - -
    iFeatures are not used.
  - iPCF
  - iPCF-MC
- **Status**
  Shows the status of the WLAN interface.
  - enabled
    The WLAN interface is enabled.
  - disabled
    The WLAN interface is disabled.

Available APs

Available access points

Note

The page is only available
- if there is an online connection to the device.
- for clients or access points in client mode.

This page shows all the access points visible to the client. The list also includes the access points to which the client cannot connect due to its configuration.

Note

Display when iPCF mode is activated

If the iPCF mode is active, the display is different. Since the client does not run a background scan in this case, only the access point with which the client is currently connected is displayed.

Displayed values

The table has the following columns:

- **Radio**
  Shows the WLAN interface visible to the access point.
- **SSID**
  Shows the SSID of the access point.
- **BSSID**
  Shows the MAC address of the access point.

- System Name
  Shows the system name of the access point. The entry depends on the access point. Not all access points support this parameter.
- Channel
  Shows the channel on which the access point transmits or communicates.
- Signal Strength [dBm]
  Shows the signal strength of the access point in bBm.
- Signal Strength [%]
  Shows the signal strength of the access point as a percentage.
- Type
  Shows the mode of the WLAN interface.
- Security
  Shows which authentication method is used.
  If the authentication method "Open System + Encryption" or "Shared Key" is used, the "Encrypted (WEP/AES)" authentication method is displayed for both.
- WLAN Mode
  Shows the transmission standard. If DFS is activated, the transmission standard "802.11h" is not shown additionally but only the configured transmission standard "802.11a" or "802.11n".
- Status
  Shows the status of the access point, for example whether or not the access point is available.

Force Roaming

Note

The page is only available

- if there is an online connection to the device.
- in access point mode.

The device monitors the connection to IP addresses cyclically. To achieve this, the device sends echo messages (pings) to the configured destination IP address at regular intervals.

This WBM page shows the current status of the connection. It also shows whether there is roaming.

Displayed values

The table has the following columns:

- **Port**
  Shows the available VAP interfaces.
- **Destination Address / State**
  Shows which destination address is monitored and the status of the connection. You configure the destination address in "Interfaces > WLAN > Force Roaming".
  - not configured: No destination address is configured.
  - idle: The configuration is incomplete.
  - up:The destination address is reachable.
  - down: The destination address is unreachable.
- **Force Roaming on IP down**
  Indicates whether roaming is currently being performed.
  - inactive: No roaming is being performed.
  - active: Roaming is being performed. None of the destination addresses is reachable.

IP mapping table

WLAN access for several devices via a client

Note

The page is only available

- if there is an online connection to the device.
- for clients or access points in client mode.

You can make WLAN access available for several devices with one client if you use IP mapping. This means that you do not need to equip every device with its own WLAN client. This is possible only if the connected devices are addressed only by IP frames. Communication at MAC address level (ISO/OSI layer 2) can

- be established with one component whose MAC address is configured on the client,
- be established with a maximum of eight components if the "Layer 2 Tunnel" MAC mode is selected.

The "Layer 2 Tunnel" setting meets the requirements of industrial applications in which MAC address-based communication takes place with several devices downstream from the client. Clients with this setting cannot connect to standard Wi-Fi devices and access points with firmware V3.0 or older.

The client maintains a table with the assignment of MAC address and IP address to be able to send incoming IP frames to the correct MAC address. The page shows the IP mapping table of the device.

Note

Display of the IP mapping table

If "Layer 2 Tunnel" is configured on the device for the MAC mode, the IP mapping table is not displayed.

Displayed values

- **MAC address**
  The MAC address of a device located downstream from the WLAN client from the perspective of the access point.
- **IP address**
  The IP address managed for this device by the WLAN client.
- **Type**
  There are two options for the type:
  - system
    The information relates to the WLAN client itself.
  - learned
    The information relates to a device downstream from the WLAN client.

MAC mode

Frames sent by the client to the access point always have the MAC address of the WLAN client as the source MAC address. In the "learning table" of the access point there is therefore only the MAC address of the WLAN client.

If there are further SCALANCE W700 devices downstream from the client, the "Automatic" option should not be enabled. In this case, the MAC address would be assigned indiscriminately to the first SCALANCE W700 device that signals over Ethernet. If there is only IP communication between the access point and the client, the default setting "Own" can be retained. If MAC address-based frames are also to be sent by SCALANCE W700 devices downstream from the client, you need to select the settings "Automatic", "Manual" or "Layer 2 Tunnel".

Background noise

The page displays the background noise of the channel.

Note

The page is only available if there is an online connection to the device.

Description

- **Connector**
  Shows the name of the relevant antenna connector.
- **Channel [dBm]**
  Shows the background noise of the set channel.
- **Extended Channel [dBm]**
  Shows the background noise of the extended channel (HT-40).

WLAN statistics

Fault

The page show how many bad frames were received or sent per WLAN interface. If an increased number of errors occurs, you should check the settings for the WLAN interface(s), the setup of the devices and the connection quality.

Note

The page is only available if there is an online connection to the device.

Displayed values

The Transmission Errors table has the following columns:

- **Interface**
  Shows the WLAN interface to which the entries apply.
- **Error types**
  The other columns after the WLAN interface contain the absolute numbers of the frames sent according to their error type.
  The columns of the table distinguish the following error types:
  - Transmission errors
    Shows the number and percentage of bad frames that were sent.
  - Discarded frames
    Shows the number and percentage of frames that were discarded. Despite all the retries, the frame could not be successfully sent.
    The frame has not yet been sent and the recipient has logged off in the meantime.
  - Retries
    Shows the number and percentage of frames sent successfully that required one or more retries.

The "Received Errors" table has the following columns:

- Interface
  Shows the WLAN interface to which the entries apply.
- Error types
  The other columns after the WLAN interface contain the absolute numbers of the frames received according to their error type.
  The columns of the table distinguish the following error types:
  - Received errors
    Shows the number and percentage of bad frames that were received.
  - Duplicate frames
    Shows the number and percentage of frames that were received twice.
  - Encryption errors
    Shows the number and percentage of incorrectly encrypted frames.
  - FCS errors
    Shows the number and percentage of frames in which the checksum was incorrect.

Sent management frames

The page shows how many frames in response to logging on or logging off were counted per VAP interface.

Note

The page is only available

- if there is an online connection to the device.
- in access point mode.

Displayed values

The table has the following columns:

- Interface
  Shows the VAP interface to which the entries apply.
- Frames
  - Management Frames
    Shows the number of management frames
  - Association Requests
    Shows the number of requesting association frames relevant for a logon.
  - Association Responses
    Shows the number of responding association frames relevant for a logon.
  - Disassociation Requests
    Shows the number of requesting disassociation frames relevant for a logoff.
  - Authentication Requests
    Shows the number of requesting Authentication frames relevant for a logon.
  - Authentication Responses
    Shows the number of responding authentication frames relevant for a logon.
  - Deauthentication Requests
    Shows the number of requesting deauthentication frames relevant for a logon.

Management Frames Received

The page shows how many frames in response to logging on or logging off were counted per VAP interface.

Note

The page is only available

- if there is an online connection to the device.
- in access point mode.

Displayed values

The table has the following columns:

- Interface
  Shows the VAP interface to which the entries apply.
- Frame
  - Management Frames
    Shows the number of management frames
  - Association Requests
    Shows the number of requesting association frames relevant for a logon.
  - Association Responses
    Shows the number of responding association frames relevant for a logon.
  - Disassociation Requests
    Shows the number of requesting disassociation frames relevant for a logoff.
  - Authentication Requests
    Shows the number of requesting Authentication frames relevant for a logon.
  - Authentication Responses
    Shows the number of responding authentication frames relevant for a logon.
  - Deauthentication Requests
    Shows the number of requesting deauthentication frames relevant for a logon.

Sent data frames

The page shows how many frames were sent per VAP interface.

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- Interface
  Shows the VAP interface to which the entries apply.
- Frame types
  The other columns after the VAP interface contain the absolute numbers of the sent frames according to the frame types.
  In the columns of the table, a distinction is made according to the following frame types:
  - Data Frames
    Shows the number of sent data frames.
  - Multicast/Broadcast Frames
    Shows the number of sent multicast and broadcast frames.
  - Unicast Frames
    Shows the number of unicast frames sent.
  - Average Data Rate
    Shows average data rate of the last data frames sent.

Received data frames

The page shows how many frames were received per VAP interface.

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- Interface
  Shows the VAP interface to which the entries apply.
- Frame types
  The other columns after the VAP interface contain the absolute numbers of the received frames according to the frame types.
  In the columns of the table, a distinction is made according to the following frame types:
  - Data Frames
    Number of received data frames.
  - Multicast/Broadcast Frames
    Shows the number of received multicast and broadcast frames.
  - Unicast Frames
    Shows the number of received unicast frames.
  - Average Data Rate
    Shows average data rate of the last data frames received.

WLAN iFeatures

iREF Client List

The page shows the antenna connector via which the clients logged on to the access point communicate. Other information such as the signal strength and the MAC address of the WLAN interface is also shown.

Note

The page is only available

- for SCALANCE W780
- for SCALANCE W780
- if there is an online connection to the device.
- in access point mode.
- with KEY-PLUG inserted: W780 iFeatures (MLFB 6GK5 907-8PA00)

Displayed values

- **Associates Stations**
  Shows the number of clients logged in to the access point.

The table has the following columns:

- **AID** (Associated ID)
  Shows the connection ID of the client. If the client connects to the access point via the VAP interface, the client is assigned a connection ID. The connection ID is unique within a VAP interface. If two clients log on at different VAP interfaces, both clients can receive the same ID.
- **Radio**
  Shows the available WLAN interfaces.
- **Port**
  Shows the VAP interface.
- **MAC Address**
  Shows the MAC address of the client.
- **System Name**
  Shows the system name of the client if the client communicates this to the access point. Not all clients support this parameter.
- **TX Chain**
  Shows the antenna connector over which the client communicates with the access point.
- **Signal Strength [dBm]**
  Shows the signal strength of the connected client in bBm.
- **Signal Strength [%]**
  Shows the signal strength of the connected client as a percentage.
- **Age [s]**
  Shows the age of the listed client.

iREF WDS List

The page shows the access points logged on to the access point via a WDS link. Information such as the antenna used and the signal strength of the WLAN interface is shown.

Note

The page is only available

- SCALANCE W780
- SCALANCE W770
- if there is an online connection to the device.
- in access point mode.
- with KEY-PLUG inserted: Access point W780 iFeatures (MLFB 6GK5 907-8PA00)

Displayed values

- **Connected WDS partners**
  Shows the number of access points logged in to the access point

The table has the following columns:

- **Radio**
  Shows the available WLAN interfaces.
- **Port**
  Shows the WDS interface.
- **BSSID**
  Shows the MAC address of the WDS partner.
- **WDS ID**
  Shows the name of the WDS partner.
- **TX Chain"**
  Shows the antenna connector over which the two access points communicate with each other.
- **Signal Strength [dBm]**
  Shows the signal strength of the connected access point in bBm.
- **Signal Strength [%]**
  Shows the signal strength of the connected access point as a percentage.

AeroScout

Note

The page is only available

- for SCALANCE W780 and SCALANCE W774-1 RJ-45
- if there is an online connection to the device.
- in access point mode.
- with KEY-PLUG inserted: Access points W780 iFeatures (MLFB 6GK5 907-8PA00)

Note

AeroScout and iPCF mode

The AeroScout function cannot be combined with other iFeatures (iPCF, iPCF-MC iREF). AeroScout can only be used in the 2.4 GHz band according to IEEE 802.11g, IEEE 802.11n and IEEE 802.11n-only.
For more detailed information, please refer to the AeroScout documentation of (www.aeroscout.com).

Displayed values

- **Tag Information forwarding**
  Shows the setting of the management program.
  In the management program that evaluates the AeroScout frames, you can specify whether or not an IWLAN device will forward frames.

  Note

  No evaluation or processing

  With a suitable configuration, the SCALANCE W device forwards AeroScout frames but does not process or evaluate them itself. This is done only in the "AeroScout System Manager" program.

- **AeroScout Status**
  Shows whether or not AeroScout is activated. You can set this for each WLAN interface of the device in "iFeatures > AeroScout". If you have enabled forwarding, "active" is displayed here, otherwise "inactive"
- **Engine Port**
  Shows the port at which the device expects UDP datagrams from the AeroScout Engine.
- **Response Port**
  Shows the port via which the device forwards the received AeroScout frames.
- **Response IP**
  Shows the IP address of the PC on which the AeroScout Engine for evaluation of the AeroScout frames is running.
- **Multicast Address**
  Shows the multicast address that the tag uses. The multicast address is configured in the AeroScout Engine.

- **Acknowledgments Sent**
  Shows the number of acknowledgments sent by the device. The number reflects the received UDP datagrams.
- **Messages Dropped**
  Shows the number of AeroScout frames that were discarded by the device. If, for example, an AeroScout tag is configured so that it sends on channel 1, the device does not forward a frame received on channel 6.

iRPR

On this page you can check whether the settings for iPRP are correct. You can, for example, see which device is the partner client.

Note

The page is only available if there is an online connection to the device.

Note

Clients or access points in client mode

- SCALANCE W740, SCALANCE W734-1 RJ45 and SCALANCE W722-1 RJ45
- The setting can only be configured with the following KEY-PLUG:
  – Client: W740 iFeatures (MLFB 6GK5 907-4PA00)

Access point mode

- SCALANCE W780 and SCALANCE W774-1 RJ45
- The setting can only be configured with the following KEY-PLUG:
  – Access point: W780 iFeatures (MLFB 6GK5 907-8PA00)

Description

The table has the following columns:

- **Radio**
  Shows the WLAN interfaces via which the client is connected to the access point
- **Port** (only in access point mode)
  Shows the VAP interface on which the iPRP clients are logged on.
- **iPRP Client**
  Shows the MAC address of the iPRP client.
- **Activity status**
  Shows whether or not iPRP is enabled.
- **Partner Client**
  Shows the MAC address of the partner client.
- **Partner BSS**
  Shows the MAC address of the access point to which the partner client is connected.

- **Delete Frames Sent**
  Shows the number of iPRP delete frames that the device (access point / client) has sent to its partner device.
- **Delete Frames Received**
  Shows the number of iPRP delete frames that the device (access point / client) has received from its partner device.
- **Frames Deleted**
  Shows the number of frames not yet sent that were deleted from the queue due to the iPRP delete frame.
- **Scanning Sync State** (in client mode only)
  So that both clients do not search for an access point and change to the scan mode at he same time they synchronize with each other.
  Synchronization can have the following statuses:
  - idle: Idling No scanning
  - requested: Query to the partner client whether scanning is possible.
  - pending: Scanning is possible. Waits for the start of scanning and then changes to the status "foreground" or "background".
  - background: Background scan is performed.
  - foreground: The client has, for example, just started up and is running a foreground scan.

Configuring system functions

Configuration

Specify the services that access the device. With some services, there are further configuration pages on which more detailed settings can be made.

Settings

The page contains the configuration overview of the access options of the device.

- **Telnet Server**
  Enable or disable the "Telnet Server" service for unencrypted access to the CLI.
- **SSH server**
  Enable or disable the "SSH Server" service for encrypted access to the CLI.
- **HTTPS Server only**
  Enable or disable access using HTTP.
- **DNS client**
  Enable or disable the DNS client. You can configure other settings in "System > DNS Client".
- **SMTP Client**
  Enable or disable the SMTP client. You can configure other settings in "System > SMTP Client".

- Syslog Client
  Enable or disable the system event client. You can configure other settings in "System > Syslog Client".
- DCP Server
  Specify whether or not the device can be accessed with DCP (Discovery and Configuration Protocol):
  - "-" (disabled)
    DCP is disabled. Device parameters can neither be read nor modified.
  - Read/write access
    With DCP, device parameters can be both read and modified.
  - Read-Only
    With DCP, device parameters can be read but cannot be modified.
- Time
  Select the required setting. The following settings are possible:
  - Manual
    The system time is set manually.
  - SIMATIC Time
    The system time is set using a SIMATIC time transmitter. You can configure other settings in "System > System Time > SIMATIC Time Client".
  - SNTP Client
    The system time is set via an SNTP server. You can configure other settings in "System > System time > SNTP client".
  - NTP Client
    The system time is set via an NTP server. You can configure other settings in "System > System Time > NTP Client".
- SNMP
  Select the required protocol. The following settings are possible:
  - "-" (SNMP disabled)
    Access to device parameters using SNMP is not possible.
  - SNMPv1/v2c/v3
    Access to device parameters is possible with SNMP versions 1, 2c or 3. You can configure other settings in "System > SNMP > General".
  - SNMPv3
    Access to device parameters is possible with SNMP version 3. You can configure other settings in "System > SNMP > General".
- SNMPv1/v2 Read-Only
  Enable or disable write access to SNMP variables with SNMPv1/v2c.
- SNMPv1 Traps
  Enable or disable the sending of SNMP traps (alarm frames). You can configure other settings in "System > SNMP > Traps".
- DHCP Client
  Enable or disable the DHCP client. You can configure other settings in "System > DHCP client".

- **DHCPv6 Client**
  Enable or disable the DHCPv6 client.
- **SINEMA configuration interface**
  If the SINEMA configuration interface is enabled, you can download configurations to the device via the TIA Portal.
- **Configuration Mode**
  Select the required mode. The following modes are possible:
  - Automatic Save
    Automatic backup mode. Approximately 1 minute after the last parameter change or when you restart the device, the configuration is automatically saved. In addition to this, the following message appears in the display area "Changes will be saved automatically in x seconds. Press 'Write Startup Config' to save immediately".

    Note

    Interrupting the save

    Saving starts only after the timer in the message has elapsed. How long saving takes depends on the device.

    During the save, the message "Saving configuration data in progress. Please do not switch off the device" is displayed.

    - Do not switch off the device immediately after the timer has elapsed.
  - Trial
    Trial mode. In Trial mode, although changes are adopted, they are not saved in the configuration file (startup configuration).
    To save changes in the configuration file, use the "Write Startup Config" button. The "Write Startup Config" button is displayed when you set trial mode. In addition to this after every parameter change the following message is displayed in the display area: "Trial Mode Active – Press "Write Startup Config" button to make your settings persistent" as soon as there are unsaved modifications. This message can be seen on every WBM page until the changes made have either been saved or the device has been restarted.

General

Device

This page contains the general device information.

Description

The page contains the following boxes:

- **Current System Time**(Only available online)
  Shows the current system time. The system time is either set by the user or by a time-of-day frame: either SINEC H1 time-of-day frame, NTP or SNTP. (readonly)
- **System Up Time**(only available online)
  Shows the operating time of the device since the last restart. (readonly)

- **Device Type**(only available online)
  Shows the type designation of the device. (readonly)
- **System Name**
  You can enter the name of the device. The entered name is displayed in the selection area. A maximum of 255 characters are possible. The system name is also displayed in the CLI input prompt. The number of characters in the CLI input prompt is limited. The system name is truncated after 16 characters.
- **System Contact**
  You can enter the name of a contact person responsible for managing the device. A maximum of 255 characters are possible.
- **Location**
  You can enter the location of the device. The entered installation location is displayed in the selection area. A maximum of 255 characters are possible.

Note

The ASCII characters 0x20 to 0x7e are used in the input boxes.

Coordinates

On this page, you can enter information on the geographic coordinates (latitude, longitude and the height above the ellipsoid according to WGS84).

Getting the coordinates

Use suitable maps for obtaining the geographic coordinates of the device.

The geographic coordinates can also be obtained using a GPS receiver. The geographic coordinates of these devices are normally displayed directly and only need to be entered in the input boxes of this page.

Description

The page contains the following input boxes with a maximum length of 32 characters.

- **"Latitude" input box**
  Enter the north or south latitude for the location of the device.
  E. G. +49° 1´ 31.67" means that the device is located at 49 degrees, 1 arc minute and 31.67 arc seconds north latitude.
  A south latitude is shown by a preceding minus character.
  You can also append the letters N (north latitude) or S (south latitude) to the numeric information (49° 1´ 31.67" N).
- **"Longitude" input box**
  Enter the eastern or western longitude for the location of the device.
  E. G. +8° 20´ 58.73" means that the device is located at 8 degrees, 20 arc minutes and 58.73 arc seconds eastern longitude.
  A western longitude is indicated by a preceding minus sign.
  You can also add the letter E (eastern longitude) or W (western longitude) to the numeric information (8° 20´ 58.73" E).
- **"Height" input box**
  Enter the value of the height above sea level in meters.
  For example, 158 m means that the device is located at a height of 158 m above sea level.
  Heights below sea level (for example the Dead Sea) are indicated by a preceding minus sign.

Agent IPv4

Configuration of the IP addresses

On this page, you configure the IPv4 address for the device.

Settings

- **IP Assign Method**
  Shows how the IPv4 address is assigned.
  - Static
    The IPv4 address is static. You enter the IP settings in "IP Address" and "Subnet Mask".
  - Dynamic (DHCP)
    The device obtains a dynamic IPv4 address from a DHCP server.
- **IP Address**
  Enter the IP address of the device. If you change the IPv4 address, the Web browser should automatically set itself to the new address. If this does not happen, enter the new address in the Web browser manually.
- **Subnet Mask**
  Enter the subnet mask of the device.
- **Default gateway**
  Enter the IPv4 address of the default gateway to be able to communicate with devices in another subnet, for example diagnostics stations, e-mail server.

- Agent VLAN ID
  Select the VLAN ID from the drop-down list. The drop-down list is only available if "802.1 Q VLAN Bridge" is set for the "Base Bridge Mode" parameter. You configure the parameter in "Layer 2 > VLAN > General". You can only select VLANs that have already been configured.

  Note

  Changing the agent VLAN ID

  If the configuration PC is connected directly to the device via Ethernet and you change the agent VLAN ID, the device is no longer reachable via Ethernet following the change.

- MAC Address (only available online)
  Shows the MAC address of the device. The MAC address is linked to the hardware and cannot be modified.

Agent IPv6

Agent IPv6

Configuration of the IP addresses

On this page, enable IPv6 on the management VAN. This VLAN interface is also called an IPv6 interface. An IPv6 interface can have several IPv6 addresses.

Description

The page contains the following:

- Interface
  Shows the VLAN interface on which IPv6 will be enabled.
- IPv6 Enable
  Enable or disable IPv6 on the interface. When you enable the setting and accept it, the link local address is created automatically.
- IPv6 Address
  Enter the IPv6 address. The entry depends on the selected address type.
- Prefix Length
  Enter the number of left-hand bits belonging to the prefix
- IPv6 Address Type
  Select the address type:
  - Unicast
  - Link Local: IPv6 address is only valid on the link.

- **Address Configuration**
  Specify the mechanism for the address configuration:
  - Automatic (default)
    The IPv6 address is created using either a stateless mechanism or a stateful mechanism.
  - DHCPv6
    Status dependent: Obtains the IPv6 address and the configuration file from the DHCPv6 server.
  - SLAAC (Stateless Address Auto Configuration)
    Stateless autoconfiguration using NDP (Neighbor Discovery Protocol)
  - Static
    Enter a static IPv6 address.
- **Rapid Commit**
  When enabled the procedure for the IPv6 address assignment is shortened. Instead of 4 DHCPv6 messages (SOLICIT, ADVERTISE , REQUEST, REPLY) only 2 DHCPv6 messages (SOLICIT, REPLY) are used. You will find further information on the messages in RFC 3315.

The table has the following columns:

- **Select**
  Select the check box in the row to be deleted.
- **Interface Name**
  Shows the name of the VLAN interface.
- **IPv6 Address**
  Shows the IPv6 address.
- **Prefix Length**
  Shows the prefix length.
- **IPv6 Address Type**
  Displays the address type. The following values are possible:
  - Unicast
  - Link Local

IPv6 default routes

On this page, you configure the IPv6 default route. The IPv6 default route is an IPv6 route, that applies to all IPv6 addresses. The device only needs to know the default gateway and sends all IPv6 packets to it.

The default gateway either knows all routes itself or has a default route to another default gateway

Description

The page contains the following:

- **Destination Network**
  The destination network (:: or 0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0) applies to all IPv6 addresses.
- **Prefix Length**
  Enter the number of left-hand bits belonging to the prefix
- **Gateway**
  Enter the IPv6 address of the gateway to which the IPv6 packets will be sent.
- **Metric**
  Enter the metric for the route. The metric corresponds to the quality of a connection, based for example on speed or costs. If there are several equal routes, the route with the lowest metric value is used.
  Range of values: 1 - 254
- **Interface**
  Specify the interface via which the network address of the destination is reached.

This table contains the following columns:

- **Select**
  Select the check box in the row to be deleted.
- **Destination Network**
  Shows the network address of the destination.
- **Prefix Length**
  Shows the prefix length.
- **Gateway**
  Shows the IPv6 address of the next gateway.
- **Interface**
  Shows the Interface of the route.
- **Administrative Distance**
  Enter the metric for the route. When creating the route, "not used" is entered automatically. The metric corresponds to the quality of a connection, based for example on speed or costs. If there are several equal routes, the route with the lowest metric value is used.
  Range of values: 1 - 254
- **Status**
  Shows whether or not the route is active.

DNS client

On this page, you can manually configure up to 3 DNS servers with IPv4 or IPv6 addresses. Manually configured DNS servers are each assigned an index from 1 to 3.

If there is more than one DNS server, the order in the table specifies the order in which the servers are queried.

The top server is queried first. The DNS server (Domain Name System) assigns a domain name to an IP address so that a device can be uniquely identified. If this function is enabled,

the device can communicate with a DNS server as a DNS client. You have the option of entering names in IP address boxes.

Note

The DNS client function can only be used if there is a DNS server in the network.

Settings

- **DNS client**
  Enable or disable depending on whether the device should operate as a DNS client.
- **DNS server used**
  - learned only
    The device uses only the DNS servers assigned by DHCP.
  - manual only
    The device uses only the manually configured DNS servers. The DNS servers must be connected to the Internet.
  - all
    The device uses all available DNS servers.
- **DNS Server Address**
  Enter the IP address of the DNS server.

The table has the following columns:

- **DNS Server Address**
  Shows the IP address of the DNS server.
- **Origin**
  This shows whether the DNS server was configured manually or was assigned by DHCP.

Restart

On this page, there is a button with which you can restart the device and various options for resetting to the device defaults.

Note

The page is only available if there is an online connection to the device.

Note

Note the following points about restarting a device:

- You can only restart the device with administrator privileges.
- A device should only be restarted with the buttons of this page or with the appropriate CLI commands and not by a power cycle on the device.
- Any modifications you have made only become active on the device after clicking the "Set Values" button on the relevant page. If the device is in "Trial Mode", configuration modifications must be saved manually before a restart. In "Automatic Save" mode, the last changes are saved automatically before a restart.

Description

The following options are available for restarting:

- **Restart**
  Click this button to restart the system. You must confirm the restart in a dialog box. During a restart, the device is reinitialized, the internal firmware is reloaded, and the device runs a self-test. The learned entries in the address table are deleted. You can leave the browser window open while the device restarts. You then need to log in again.
- **Restore Memory Defaults and Restart**
  Click on this button to restore the factory configuration settings with the exception of the following parameters and to restart:
  - IP addresses
  - Subnet mask
  - IP address of the default gateway
  - DHCP client ID
  - DHCP
  - System name
  - System location
  - System contact
  - Mode of the device
- **Restore Factory Defaults and Restart**
  Click on this button to restore the factory configuration settings. The protected defaults are also reset.
  An automatic restart is triggered.

  Note

  By resetting to the factory configuration settings, the device loses the IP address. Following this, the device can only be accessed using the Primary Setup Tool or using DHCP.

  With the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

Commit Control

Description

Note

The page is only available if there is an online connection to the device.

On this page, you specify when the WLAN settings become effective on the device.
If you change a WLAN setting and confirm the change, this change is adopted and takes effect immediately. To do this, the WLAN connection is briefly interrupted. This means that you can lose the WLAN connection to your device before it is fully configured.

With the "Manual Commit" setting, you have the opportunity of first fully configuring the device. The changes are accepted, but are not active immediately. The changes only take effect when you confirm the changes with the "Set Values" button.

Note

If you configure the device via the WLAN interface, we recommend that you use the "Manual Commit" setting. Check the parameters again before you confirm the changes with the "Set Values" button.

Settings

- **Commit Mode**
  Select the required setting.
  - Automatic Commit
    Each change in the WLAN settings is adopted and is immediately effective when you click the "Set Values" button. With its default setting, the SCALANCE W700 device is set to "Automatic Commit".
  - Manual Commit
    The changes are accepted, but do not take effect immediately. The changes only take effect when you click the "Set Values" button. The "Commit Changes" button is displayed when you set "Manual Commit". In addition to this the message “Manual Commit Mode active - Click the "Commit Changes" button to to provide current configuration to driver” appears as soon as there are WLAN changes. This message can be seen on every WBM page until the changes made have either taken effect or the device has been restarted.

    Note

    When the changes take effect, the WLAN connection to all WLAN interfaces will be interrupted for a short time. The WLAN driver is started with the new settings.

Load & save

Note

The files that can be loaded from the device depends on the role of the logged-on.

Overview of the file types

Table 1-151 HTTP

File type	Description	Down-load	Save	Delete
Config	This file contains the start configuration. Among other things, this file contains the definitions of the users, roles, groups and function rights. The passwords are stored in the "Users" file.	X	X	--
ConfigPack	Detailed configuration information. for example, start configuration, users, certificates, favorites, firmware of the device (if saved as well). For more detailed information on creating and using the ConfigPack incl. firmware, refer to the section "Maintenance".	X	X	--
CountryList	The zip file contains the country list as a csv and as a pdf file.	--	X	--
Debug	This file contains information for Siemens Support. It is encrypted and can be sent by e-mail to Siemens Support without any security risk.	--	X	X
EDS	Electronic Data Sheet (EDS) Electronic data sheets for describing devices in the EtherNet/IP mode	--	X	--
Firmware	The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.	X	X	--
GSDML	Information on the device properties (PROFINET)	--	X	--
HTTPS Cert	HTTPS certificate Maximum file size: 8192 bits	X	X	X
LogFile	File with entries from the event log table	--	X	--
MIB	Private MSPS MIB file "Scalance_w_msps.mib"	--	X	--
RunningCLI	Text file with CLI commands This file contains an overview of the current configuration in the form of CLI commands. Passwords are masked in this file as follows: [PASSWORD] You can download the text file. The file is not intended to be uploaded again unchanged.	--	X	--
Script	Text file with CLI commands	X	--	--
StartupInfo	Startup log file This file contains the messages that were entered in the log file during the last startup.	--	X	--

File type	Description	Down-load	Save	Delete
Users	File with user names and passwords	X	X	--
WBMFav	WBM favorites This file contains the favorites that you created in the WBM. You can download this file and upload it to other devices.	X	X	X
WLANAuth-log	File with entries from the WLAN Authentication Log (information on successful or failed authentication attempts)	--	X	--
WLANCert (in client mode only)	User certificate. You can specify a password for the user certificate on the WBM page "Load&Save > Password". Maximum file size: 8192 bits	X	X	X
WLANServ-Cert (in client mode only)	Server certificate Maximum file size: 8192 bits	X	X	X
WLANSi-gRec (in client mode only)	The zip file contains the following: • csv file with the measured values of the signal recorder • pdf file with the measured values and an additional graphic representation of the measured values. You will find information about the measured values and their graphic representation in the section "Signal recorder".	--	X	X
WLANSpec-trumAnalyz-er (Only in access point mode)	The Zip file contains a csv file with the measured values of the spectrum analyzer. You will find information about the measured values and their graphic representation in the section "Spectrum analyzer".	--	X	X

Table 1-152 TFTP/SFTP

File type	Description	Save	Down-load
Config	This file contains the start configuration. Among other things, this file contains the definitions of the users, roles, groups and function rights. The passwords are stored in the "Users" file.	X	X
ConfigPack	Detailed configuration information. for example, start configuration, users, certificates, firmware of the device (if saved as well). For more detailed information on creating and using the ConfigPack incl. firmware, refer to the section "Maintenance".	X	X
CountryList	The zip file contains the country list as a csv and as a pdf file.	X	--

File type	Description	Save	Down-load
Debug	This file contains information for Siemens Support. It is encrypted and can be sent by e-mail to Siemens Support without any security risk.	X	--
EDS	Electronic Data Sheet (EDS) Electronic data sheets for describing devices in the EtherNet/IP mode	X	--
Firmware	The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.	X	X
GSDML	Information on the device properties (PROFINET)	X	--
HTTPS Cert	Preset HTTPS certificates including key The preset and automatically created HTTPS certificates are self-signed. We strongly recommend that you create your own HTTPS certificates and make them available. We recommend that you use HTTPS certificates signed either by a reliable external or by an internal certification authority. The HTTPS certificate checks the identity of the device and controls the encrypted data exchange. There are files to which access is password-protected. To successfully load the file into the device, enter the password specified for the file on the WBM page "Passwords". Maximum file size: 8192 bits	X	X
LogFile	File with entries from the event log table	X	--
MIB	Private MSPS MIB file "Scalance_w_msps.mib"	X	--
RunningCLI	Text file with CLI commands This file contains an overview of the current configuration in the form of CLI commands. Passwords are masked in this file as follows: [PASSWORD] You can download the text file. The file is not intended to be uploaded again unchanged.	X	--
Script	Text file with CLI commands You can upload a script file in a device. The CLI commands it contains are executed accordingly. CLI commands for saving and loading files cannot be executed with the CLI script file.	--	X
StartupInfo	Startup log file This file contains the messages that were entered in the log file during the last startup.	X	--
Users	File with user names and passwords	X	X
WBMFav	WBM favorites This file contains the favorites that you created in the WBM. You can download this file and upload it to other devices.	X	X
WLANAuthlog	File with entries from the WLAN Authentication Log (information on successful or failed authentication attempts)	X	--

File type	Description	Save	Down-load
WLANCert (in client mode only)	User certificate. You can specify a password for the user certificate on the WBM page "Load&Save > Password". Maximum file size: 8192 bits	X	X
WLANServerCert (in client mode only)	Server certificate Maximum file size: 8192 bits	X	X
WLANSigRec (in client mode only)	The zip file contains the following: • csv file with the measured values of the signal recorder • pdf file with the measured values and an additional graphic representation of the measured values. You will find information about the measured values and their graphic representation in the section "Signal recorder".	X	--
WLANSpectrumAnalyzer (Only in access point mode)	The Zip file contains a csv file with the measured values of the spectrum analyzer. You will find information about the measured values and their graphic representation in the section "Spectrum analyzer".	X	--

Uploading and saving using HTTP

Loading and saving data via HTTP

This page allows you to store device data in an external file on your client PC or to load such data from an external file from the client PC to the devices.
This means, for example, that you can also load new firmware from a file located on your client PC.

Note

The page is only available if there is an online connection to the device.
This page is available both for connections using HTTP and for connections using HTTPS.

Firmware

The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.

Note

Incompatibility with previous firmware versions with/without PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed.

In this situation, if a PLUG is inserted in the device, following the restart, this has the status "Not Accepted" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data. If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using the WBM page "System > PLUG".

Configuration files

Note

Configuration files and trial mode/automatic save mode

In Automatic Save mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In Trial mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write Startup Config" button on the "System > Configuration" WBM page to save changes in the configuration file.

CLI script file

You can download existing CLI configurations (RunningCLI) and upload your own CLI scripts (Script).

Note

The downloadable CLI script is not intended to be uploaded again unchanged.

Exchange of configuration data with STEP 7 Basic/Professional using a file

You use the two file types "RunningSINEMAConfig" and "SINEMAConfig" to exchange configuration data between a device (WBM) and STEP 7 Basic/Professional via a file.

Requirements:

- Same article number
- Same firmware version
- Password
  You assign the password in the WBM under "System > Load&Save > Passwords".

You can use the file types as follows:

- For offline diagnostics
  You can save the faulty configuration of a device as "RunningSINEMAConfig" via the WBM and import it in STEP 7 Basic/Professional. No connection to a real device is required for the diagnostics in STEP 7 Basic/Professional. You can export a corrected configuration and load it as "SINEMAConfig" again using the WBM.
- For configuration
  No connection to a real device is required to configure a device in STEP 7 Basic/Professional. You can export the configuration and load it as "SINEMAConfig" to the real device using the WBM.

Description

The table has the following columns:

- **Type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.
- **Load**
  With this button, you can upload files to the device. The button can be enabled, if this function is supported by the file type.
- **Save**
  With this button, you can save files from the device. The button can only be enabled if this function is supported by the file type and the file exists on the device.
- **Delete**
  With this button, you can delete files from the device. The button can only be enabled if this function is supported by the file type and the file exists on the device.

Note

Following a firmware update, delete the cache of your Internet browser.

Uploading and saving using TFTP

Loading and saving data via a TFTP server

On this page, you can configure the TFTP server and the file names. You can also store device data in an external file on a TFTP server or load such data from an external file from the TFTP server to the devices. This means, for example, that you can also load new firmware from a file located on a TFTP server.

Firmware

The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.

Note

Incompatibility with previous firmware versions with/without PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed.

In this situation, if a PLUG is inserted in the device, following the restart, this has the status "Not Accepted" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data. If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using the WBM page "System > PLUG".

Configuration files

Note

Configuration files and Trial mode /Automatic save

In "Automatic Save" mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In "Trial" mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write startup config" button on the "System > Configuration" WBM page to save changes in the configuration files.

CLI script file

You can download existing CLI configurations (RunningCLI) and upload your own CLI scripts (Script).

Note

The downloadable CLI script is not intended to be uploaded again unchanged.

CLI commands for saving and loading files cannot be executed with the CLI script file (Script).

Exchange of configuration data with STEP 7 Basic/Professional using a file

You use the two file types "RunningSINEMAConfig" and "SINEMAConfig" to exchange configuration data between a device (WBM) and STEP 7 Basic/Professional via a file.

Requirements:

- Same article number
- Same firmware version
- Password
  You assign the password in the WBM under "System > Load&Save > Passwords".

You can use the file types as follows:

- For offline diagnostics
  You can save the faulty configuration of a device as "RunningSINEMAConfig" via the WBM and import it in STEP 7 Basic/Professional. No connection to a real device is required for the diagnostics in STEP 7 Basic/Professional. You can export a corrected configuration and load it as "SINEMAConfig" again using the WBM.
- For configuration
  No connection to a real device is required to configure a device in STEP 7 Basic/Professional. You can export the configuration and load it as "SINEMAConfig" to the real device using the WBM.

Description

- **TFTP Server Address**
  Enter the IP address or the FQDN of the TFTP server with which you exchange data.
- **TFTP Server Port**
  Here, enter the port of the TFTP server via which data exchange will be handled. If necessary, you can change the default value 69 to your own requirements.

The table has the following columns:

- **File type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.
- **File name**
  A file name is preset here for every file type.

  Note

  Changing the file name

  You can change the file name preset in this column. After loading on the device, the changed file name can also be used with the Command Line Interface.

- **Action** (only available online)
  Select the required action. The selection depends on the selected file type, for example you can only save the log file.
  The following actions are possible:
  - Save file
    With this selection, you save a file on the TFTP server.
  - Load file
    With this selection, you load a file from the TFTP server, e.g. certificates required to establish a secure connection.

Uploading and saving using SFTP

Loading and saving data via an SFTP server

SFTP (SSH File Transfer Protocol) transfers the files encrypted. On this page, you configure the access data for the SFTP server.

You can also store device data in an external file on your client PC or load such data from an external file from the PC to the devices. This means, for example, that you can also load new firmware from a file located on your Admin PC.

On this page, the certificates required to establish a secure VPN connection can also be loaded.

Firmware

The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.

Configuration files

Note

Configuration files and Trial mode /Automatic Save

In "Automatic Save" mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In "Trial" mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write Startup Config" button on the "System > Configuration" WBM page to save changes in the configuration files.

CLI script file

You can download existing CLI configurations (RunningCLI) and upload your own CLI scripts (Script).

Note

The downloadable CLI script is not intended to be uploaded again unchanged.

CLI commands for saving and loading files cannot be executed with the CLI script file (Script)

Description

The page contains the following boxes:

- **SFTP Server Address**
  Enter the IP address or the FQDN of the SFTP server with which you exchange data.
- **SFTP Server Port**
  Enter the port of the SFTP server via which data exchange will be handled. If necessary, you can change the default value 22 to your own requirements.
- **SFTP User**
  Enter the user for access to the SFTP server. This assumes that a user with the corresponding rights has been created on the SFTP server.

- SFTP Password
  Enter the password for the user
- SFTP Password Confirmation
  Confirm the password.

The table has the following columns:

- Type
  Shows the file type.
- Description
  Shows the short description of the file type.
- Filename
  A file name is preset here for every file type.

  Note

  Changing the file name

  You can change the file name preset in this column. After loading on the device, the changed file name can also be used with the Command Line Interface.

- Actions
  Select the required action. The selection depends on the selected file type, for example you can only save the log file.
  The following actions are possible:
  - Save file
    With this selection, you save a file on the SFTP server.
  - Load file
    With this selection, you load a file from the SFTP server, e.g. certificates required to establish a secure connection.

Passwords

Password for files

There are files to which access is password protected. To load the file on the device, enter the password specified for the file on the page.

Description

- Type
  Shows the file type
- Description
  Shows the short description of the file type.
- Enabled
  When selected, the file is used. Can only be enabled if the password is configured.
- Password
  Enter the password for the file.

- **Password Confirmation**
  Confirm the new password.
- **Status** (only available online)
  Shows whether the password matched the file on the device.
  - Valid
    The "Enabled" check box is selected and the password matches the file.
  - Invalid
    The "Enabled" check box is selected but the password does not match the file or no file has been loaded yet.
  - '-'
    The password cannot be evaluated or is not yet being used. The "Enabled" check box is not selected.

Events

Configuration

On this page, you specify how the device reacts to system events. By enabling the appropriate options, you specify how the device reacts to events. To enable or disable the options, click the relevant settings of the columns.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all events of table 2.
- **E-mail / Trap / Log Table / Syslog / Faults**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Event**
  The column contains the following values:
  - **Cold/Warm Restart**
    The device was turned on or restarted by the user.
  - **Link Change**
    This event occurs only when the port status is monitored and has changed, see "System > Fault Monitoring > Link Change".
  - Authentication error
    This event occurs when attempting access with a bad password.
  - Power Change
    This event occurs when power supply lines 1 and 2 are monitored. It indicates that there was a change to line 1 or line 2. The event occurs when the PoE power supply has failed, see "System > Fault Monitoring > Power Supply".
  - Spanning Tree Change
    The STP or RSTP or MSTP topology has changed.
  - Fault State Change
    The fault status has changed. The fault state can relate to the activated port monitoring, the response of the signaling contact or the power supply monitoring.
  - Overlap AP Detection (only in access point mode)
    This event is triggered when there is an entry in the Overlap AP list.
  - WDS (Only in access point mode)
    The connection status of a WDS link has changed.
  - DFS (Only in access point mode)
    This event occurs if a radar signal was received or the DFS scan was started or stopped.
  - WLAN General (Only in access point mode)
    This event occurs if a the channel bandwidth has changed.
  - WLAN Authentication Log
    Forwarding of the entries from the WLAN authentication log to the system protocol server.
  - WLAN De/Authentication (Only in client mode)
    With successful or failed WLAN authentication attempts.
  - iPCF Cycle Time (Only in access point mode)
    Only available when the KEY-PLUG is inserted.
    This event occurs if too many clients are logged on for the set iPCF cycle time or if some clients were not reached in one cycle.
  - iPCF Poll Size
    Only available when the KEY-PLUG is inserted.
    This event occurs if the PROFINET data size is too large for transfer.
  - Configuration Change
    This event occurs when the configuration of the device has changed.
- **E-mail**
  The device sends an e-mail. This is only possible if the SMTP server is set up and the "SMTP client" function is enabled.

- Trap
  The device sends an SNMP trap. This is only possible if "SNMPv1 Traps" is enabled in "System > Configuration".
- Log Table
  The device writes an entry in the event log table.
- Syslog
  The device writes an entry to the system log server. This is only possible if the system log server is set up and the "Syslog client" function is enabled.
- Faults
  The device triggers an error. The error LED lights up.
- E-mail
  The device sends an e-mail. This is only possible if the SMTP server is set up and the "SMTP client" function is enabled.
- Trap
  The device sends an SNMP trap. This is only possible if "SNMPv1 traps" is enabled in "System > Configuration".
- Log table
  The device writes an entry in the log table.
- Syslog
  The device writes an entry to the system log server. This is only possible if the system log server is set up and the "Syslog client" function is enabled.
- Fault
  The device triggers a fault. The error LED lights up

Severity filter

On this page, set the threshold levels for sending system event notifications.

Settings

The table has the following columns:

- **Client Type**
  Select the client type for which you want to make settings:
  - E-mail
    Sending system event messages by e-mail
  - Log table
    Entry of system events in the log table.
  - Syslog
    Entering system events in the Syslog file
  - WLAN Authentication Log
    Forwarding of the entries from the WLAN Authentication Log to the system protocol server.
    Can only be configured with SCALANCE W770 and SCALANCE W780
- **Severity**
  Select the required level. The following settings are possible:
  - Info
    System events are processed as of the severity level "Info".
  - Warning
    System events are processed as of the severity level "Warning".
  - Critical
    System events are processed as of the severity level "Critical".

SMTP client

Network monitoring with e-mails

The device provides the option of automatically sending an e-mail if an alarm event occurs (for example to the network administrator). The e-mail contains the identification of the sending device, a description of the cause of the alarm in plain language, and a time stamp. This allows centralized network monitoring to be set up for networks with few nodes based on an E-mail system. When an e-mail error message is received, the WBM can be started by the Internet browser using the identification of the sender to read out further diagnostics information.

On this page, you can configure up to three SMTP servers and the corresponding e-mail addresses.

Description

The page contains the following boxes:

- **SMTP Client**
  Enable or disable the SMTP client.
- **Sender Email Address**
  Enter the name of the sender to be included in the e-mail, for example the device name. This setting applies to all configured SMTP servers.

- **Send Test Mail**
  Send a test e-mail to check your configuration.
- **SMTP Port**
  Enter the port via which your SMTP server can be reached.
  Factory settings: 25
  This setting applies to all configured SMTP servers.
- **SMTP Server Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the SMTP server.

This table contains the following columns:

- **SMTP Server Address**
  Shows the IP address or the FQDN (Fully Qualified Domain Name) of the SMTP server.
- **Email address of the recipient**
  Enter the e-mail address to which the device sends an e-mail if a fault occurs.

Syntax of email addresses

The following conditions apply to e-mail addresses:

- Alphanumeric characters are permitted.
- The following special characters are permitted:
  - @
  - _ (underscore)
  - - (hyphen)
  - . (period)
- A @ character must be included.
- Only one @ may be used.
- The characters "@" and the "." must not be the first or last character.

Note

Depending on the properties and configuration of the SMTP server, it may be necessary to adapt the "Sender E-Mail Address” input box. Check with the administrator of the SMTP server.

DHCPv4

DHCP client

Setting of the DHCP mode

If the device is configured as a DHCP client, it starts a DHCP query. As the reply to the query the device receives an IPv4 address from the DHCP server. The server manages an address range from which it assigns IPv4 addresses. It is also possible to configure the server so that the client always receives the same IPv4 address in response to its request.

Descriptions

- **DHCP client configuration file request (opt 66, 67)**
  Select this option if you want the DHCP client to use options 66 and 67 to download and then enable a configuration file.
- **DHCP Mode**
  Select the DHCP mode. The following modes are possible:
  - via MAC Address
    Identification is based on the MAC address.
  - via DHCP Client ID
    Identification is based on a freely defined DHCP client ID.
  - via System Name
    Identification is based on the system name. If the device name is 255 characters long, the last character is not used for identification.
  - via PROFINET Name of Station
    Identification is based on the PROFINET name of the station.
  - via IAID and DUID
    With this, the DHCP client can log on to DHCP servers that support parallel operation of IPv4 and IPv6.
    The identification is via the IAID and the DUID and identifies precisely one IP interface of the device.
    IAID (Interface Association Identifier): At least one IAID is generated for each IP interface. The IAID remains unchanged when the DHCP client restarts.
    DUID (DHCP Unique Identifier): Uniquely identifies servers and clients and applies to all IP interfaces of the device. The DUID remains unchanged when there is a restart, unless the user changes it.
- **DUID-Type**
  Specify which DUID type will be used. The DUID types are defined in RFC 3315.
  - DUID-LLT
    DUID is based on the link layer address of the interface and a time stamp
  - DUID-EN
    DUID is assigned by the vendor (EN = enterprise number)
  - DUID-LL
    DUID is based on the link layer address of the interface

- **Link-layer Address Plus Time (LLT)**
  The value is based on the link layer address of the interface and a time stamp. The value is regenerated each time the factory settings are restored. The value can be changed if necessary.
- **Vendor Enterprise Number (EN)**
  The value is based on the enterprise number specific to the vendor. The value is regenerated each time the factory settings are restored. The value can be changed if necessary.
- **Link-layer address (LL)**
  The link layer address is based on the MAC address. The value is regenerated each time the factory settings are restored. The value can be changed if necessary.

This table contains the following columns:

- **Interface**
  Interface to which the setting relates.
- **DHCP**
  Enable or disable the DHCP client for the relevant interface.
- **IAID Value**
  Value with which the interface (DHCP client) identifies itself on the DHCP server.

DHCP server

Note

The page is only available for

- SCALANCE W780 / W770 / W740 / W730, W761-1 RJ-45 and W722-1 RJ-45

You can operate the device as a DHCP server. This allows IPv4 addresses to be assigned automatically to the connected devices. The IPv4 addresses are either distributed dynamically from an IPv4 address band you have specified or a specific IPv4 address (static) can be assigned to a particular device.

On this page, specify the IPv4 address band from which the device receives any IPv4 address. You configure the static assignment of the IPv4 addresses in "Static IP addresses".

Requirement

- In access point mode
  - The connected devices are configured so that they obtain the IPv4 address from a DHCP server.
- In client mode
  - The connected devices are configured so that they obtain the IPv4 address from a DHCP server.
  - NAT is enabled. You enable NAT in "Layer 3 > NAT".

Settings

- **DHCP server**
  Enable or disable the DHCP server on the device.

  Note

  To avoid conflicts with IPv4 addresses, only one device may be configured as a DHCP server in the network.

  Note

  Access point

  With an access point, the "DHCP Server" function is only possible on the VLAN assigned to the management (agent VLAN ID).

- **Prior to offering, check the address with ICMP echo**
  When selected, the DHCP server checks whether or not the IPv4 address has already been assigned. To do this the DHCP server sends ICMP echo messages (ping) to the IPv4 address. If no reply is received, the DHCP server can assign the IPv4 address.

  Note

  If there are devices in your network on which the echo service is disabled as default, there may be conflicts with the IPv4 addresses. To avoid this, assign these devices an IPv4 address outside the IPv4 address band.

The table has the following columns:

- **Pool ID**
  Shows the number of the IPv4 address band.

  Note

  Only one pool ID (ID = 1) can be created.

- **Enable**
  Specify whether or not this IPv4 address band will be used.

  Note

  If the IPv4 address band is enabled, neither DHCP options nor static PPv4 addresses can be configured.

- **Interface**
  Specify the interface via which the IPv4 addresses are dynamically assigned.
  The requirement for the assignment is that the IPv4 address of the interface is located within the IPv4 address band. If this is not the case, the interface does not assign any IPv4 addresses.
- **Subnet**
  Enter the network address range that will be assigned to the devices. Use the CIDR notation.
- **Lower IP address**
  Enter the IPv4 address that specifies the start of the dynamic IPv4 address band. The IPv4 address must be within the network address range you configured for "Subnet".

- **Upper IP address**
  Enter the IPv4 address that specifies the end of the dynamic IPv4 address band. The IPv4 address must be within the network address range you configured for "Subnet".
- **Lease Time [sec]**
  Specify for how many seconds the assigned IPv4 address remains valid. Once this period has elapsed, the device must either request a new IPv4 address or extend the lease time of the existing IPv4 address.

DHCP options

Note

The page is only available for

- SCALANCE W780 / W770 / W740 / W730, W761-1 RJ-45 and W722-1 RJ-45

On this page you specify which DHCP options the DHCP server supports. The various DHCP options are defined in RFC 2132.

Settings

- **Pool ID**
  Select the required IPv4 address band.
- **Option Code**
  Enter the number of the required DHCP option. The various DHCP options are defined in RFC 2132. The DHCP options 1, 3, 6, 66 and 67
  are created automatically when the IPv4 address band is created. With the exception of option 1, the options can be deleted.
  With the DHCP option 3, the internal IPv4 address of the device is automatically set as a DHCP parameter.

  Note

  Not supported DHCP servers

  The DHCP options 50 - 60 and 255 are not supported.

The table has the following columns:

- **Pool ID**
  Shows the number of the IPv4 address band.
- **Option Code**
  Shows the number of the DHCP option.

- **Use Interface IP**
  Specify whether or not the internal IPv4 address of the device will be used.
- **Value**
  Enter the DHCP parameter that is transferred to the DHCP client. The content depends on the DHCP option.
  - DHCP option 67 (boot file name)
    Enter the name of the boot file in the string format.
  - DHCP options 3 (Router) and 6 (DNS):
    Enter the DHCP parameter as an IPv4 address, e.g. 192.168.100.2. With DHCP option 6, you can specify several IPv4 addresses separated by commas.
  - DHCP option 66 (TFTP Server):
    Enter the DHCP parameter as an IPv4 address, e.g. 192.168.100.2 or the FQDN name.
  - All other DHCP options
    Enter the DHCP parameter in hexadecimal, e.g. the IPv4 address 192.168.100.2 corresponds to "C0A86402".

Static IP addresses

Note

The page is only available for

- SCALANCE W780 / W770 / W740 / W730, W761-1 RJ-45 and W722-1 RJ-45

On this page you specify that devices with a certain MAC address are assigned to the selected IPv4 address.

Settings

- **Pool ID**
  Select the required IPv4 address band.
- **Client Identification Method**
  Select the method according to which a client is identified.
  - Ethernet MAC
    The client is identified by its MAC address.
  - Client ID
    The client is identified by a freely defined DHCP client ID.
- **Value**
  Enter the MAC address.

The table has the following columns:

- **Pool ID**
  Shows the number of the IPv4 address band.

  Note

  Only Pool ID = 1 is supported.

- **Identification Method**
  Shows whether the client is identified by its MAC address or the client ID.
- **Value**
  Shows the MAC address to which the IPv4 address is assigned.
- **IP Address**
  Specify the IPv4 address. The IPv4 address must match the subnet of the IPv4 address band.

SNMP

General

On this page, you make the basic settings for SNMP. Enable the options according to the function you want to use.

Description

- **SNMP**
  Select the SNMP protocol. The following settings are possible:
  - "-" (disabled)
    SNMP is disabled.
  - SNMPv1/v2c/v3
    SNMPv1/v2c/v3 is supported.

    Note

    Note that SNMP in versions 1 and 2c does not have any security mechanisms.

  - SNMPv3
    Only SNMPv3 is supported.
- **SNMPv1/v2 Read-Only**
  If you enable this option, SNMPv1/v2c can only read the SNMP variables.

  Note

  Community String

  For security reasons, do not use the standard values "public" or "private". Change the community strings following the initial installation.

  The recommended minimum length for community strings is 6 characters.

- **SNMPv1/v2c Read Community String**
  Enter the community string for access of the SNMP protocol.
- **SNMPv1/v2c Read/Write Community String**
  Enter the community string for read and write access of the SNMP protocol.
- **SNMPv1 Traps**
  Enable or disable the sending of SNMPv1 traps (alarm frames). In the "Trap" entry, specify the IP addresses of the devices to which SNMPv1 traps will be sent.
- **SNMPv1/v2c Trap Community String - Trap**
  Enter the community string for sending SNMPv1/v2c messages.
- **SNMPv3 User Migration**
  - **Enabled**
    If the function is enabled, an SNMP engine ID is generated that can be migrated. You can transfer configured SNMPv3 users to a different device.
    If you enable this function and load the configuration of the device on another device, configured SNMPv3 users are retained.
  - **Disabled**
    If the function is disabled, a device-specific SNMP engine ID is generated. To generate the ID, the agent MAC address of the device is used. You cannot transfer this SNMP user configuration to other devices.
    If you load the configuration of the device on another device, all configured SNMPv3 users are deleted.
- **SNMP Engine ID** (only available online)
  Shows the SNMP engine ID.

Traps

If an alarm event occurs, a device can send SNMP traps (alarm frames) to up to ten different management stations at the same time. Traps are only sent if the events specified in "Events" occur.

Note

SNMP traps are sent only when the "SNMPv1 Traps" setting was selected in "SNMP > General" or in "System > Configuration".

Description

- **Trap Receiver Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the station to which the device sends SNMP traps. You can specify up to ten different recipients.

The table has the following columns:

- Trap Receiver Address
  If necessary, change the IP addresses, the FQDN (Fully Qualified Domain Name) of the stations.
- Trap
  Enable or disable the sending of SNMP traps. Stations that are entered but not selected do not receive SNMP traps.

v3 Groups

Security settings and assigning permissions

SNMP version 3 allows permissions to be assigned, authentication, and encryption at protocol level. The security level and read/write permissions are defined according to groups. The settings automatically apply to every member of a group.

Description

- Group Name
  Enter the name of the group. The maximum length is 32 characters.
- Security Level
  Select the security level (authentication, encryption) valid for the selected group. The available options are as follows:
  - no Auth/no Priv
    No authentication enabled / no encryption enabled.
  - Auth/no Priv
    Authentication enabled / no encryption enabled.
  - Auth/Priv
    Authentication enabled / encryption enabled.

The table has the following columns:

- Group Name
  Shows the defined group names.

  Note

  Once a group name and the security level have been specified, they can no longer be modified after the group is created. If you want to change the group name or the security level , you will need to delete the group and recreate it and reconfigure it with the new name.

- Security Level
  Shows the configured security level.
- Read
  Enable or disable read access for the required group.

- Write
  Enable or disable write access for the required group.

  Note

  For write access to work, you also need to enable read access.

- Persistence
  Shows whether or not the group is assigned to an SNMPv3 user. If the group is not assigned to an SNMPv3 user, no automatic saving is triggered and the configured group is deleted after restarting the device.
  - Yes
    The group is assigned to an SNMPV3 user.
  - No
    The group is not assigned to an SNMPV3 user.

v3 users

User-specific security settings

On the page, you can create new SNMPv3 users and modify or delete existing users. The user-based security model works with the concept of the user name; in other words, a user ID is added to every frame. This user name and the applicable security settings are checked by both the sender and recipient.

Description

- User name
  Enter a freely selectable user name. After you have entered the data, you can no longer modify the name.

The table has the following columns:

- User name
  Shows the created users.
- Group Name
  Select the group to which the user will be assigned.
- Authentication Protocol
  Specify the authentication protocol for which a password will be stored.
  The following settings are available:
  - None
  - MD5
  - SHA
- Encryption Protocol
  Specify whether or not a password should be stored for encryption with the DES algorithm. Can only be enabled when an authentication protocol has been selected.

- Authentication Password
  Enter the authentication password. This password must have at least 1 character, the maximum length is 32 characters.

  Note
  Length of the password

  As an important measure to maximize security, we recommend that the password has a minimum length of 6 characters and that it contains special characters, uppercase/ lowercase letters, numbers.

- Confirm authentication password
  Confirm the password by repeating the entry.
- Encryption Password
  Enter your encryption password. This password must have at least 1 character, the maximum length is 32 characters.

  Note
  Length of the password

  As an important measure to maximize security, we recommend that the password has a minimum length of 6 characters and that it contains special characters, uppercase/ lowercase letters and numbers.

- Confirm Encryption Password
  Confirm the encryption password by repeating the entry.
- Persistence
  Shows whether or not the user is assigned to an SNMPv3 group. If the user is not assigned to an SNMPv3 group, no automatic saving is triggered and the configured user is deleted after restarting the device.
  - Yes
    The user is assigned to an SNMPv3 group.
  - No
    The user is not assigned to an SNMPv3 group.

System time

Manual time setting

Manual setting of the system time

On this page, you set the date and time of the system. For this setting to be used, enable "Manual time setting".

Note

The page is only available if there is an online connection to the device.

Description

- **Time Manually**
  Enable or disable the manual time setting. If you enable the option, the "System Time" input box can be edited.
- **System Time**
  Enter the date and time in the format "MM/DD/YYYY HH:MM:SS".
- **Use PC Time**
  Click the button to use the time setting of the PC.
- **Last Synchronization Time**
  Shows when the last time-of-day synchronization took place.
  If no time-of-day synchronization was possible, the box displays "Date/time not set".
- **Last Synchronization Mechanism**
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame.
  - PTP
    Automatic time-of-day synchronization with PTP
- **Daylight Saving Time**
  Shows whether the daylight saving time changeover is active.
  - active (offset + 1 h)
    The system time was changed to daylight saving time; in other words an hour was added. You can see the current system time at the top right in the selection area of the WBM. The set time continues to be displayed in the "System Time" box.
  - inactive (offset +0 h)
    The current system time is not changed.

DST overview

Daylight saving time switchover

On this page, you can create new entries for the daylight saving time changeover. The table provides an overview of the existing entries.

Description

- **DST No.**
  Shows the number of the entry.
  If you create a new entry, a new line with a unique number is created.
- **Name**
  Shows the name of the entry.
- **Year**
  Shows the year for which the entry was created.
- **Start Date**
  Shows the month, day and time for the start of daylight saving time.
- **End Date**
  Shows the month, day and time for the end of daylight saving time.
- **Recurring Date**
  With an entry of the type "Rule", the period in which daylight saving time is active is displayed consisting of week, day, month and time of day. With an entry of the type "Date" a "-" is displayed.
- **State**
  Shows the status of the entry:
  - Enabled
    The entry was created correctly.
  - Invalid
    The entry was created new and the start and end date are identical.
- **Type**
  Shows how the daylight saving time changeover is made:
  - Date
    A fixed date is entered for the daylight saving time changeover.
  - Recurring
    A rule was defined for the daylight saving time changeover.

DST configuration

Configuring the daylight saving time switchover

On this page, you can configure the entries for the daylight saving time changeover. As result of the changeover to daylight saving or standard time, the system time for the local time zone is correctly set. You can define a rule for the daylight saving time changeover or specify a fixed date.

Description

Note

The content of this page depends on the selection in the "Type" box.

The boxes "DST No.", "Type" and "Name" are always shown.

- **DST No.**
  Select the type of the entry.
- **Type**
  Select how the daylight saving time changeover is made:
  - Date
    You can set a fixed date for the daylight saving time changeover. This setting is suitable for regions in which the daylight saving time changeover is not governed by rules.
  - Recurring
    You can define a rule for the daylight saving time changeover. This setting is suitable for regions in which the daylight saving time always begins or ends on a certain weekday.
- **Name**
  Enter a name for the entry. The name can be a maximum of 16 characters long.

Settings with "Date" selected

You can set a fixed date for the start and end of daylight saving time.

- **Year**
  Enter the year for the daylight saving time changeover.
- **Start Date**
  Enter the following values for the start of daylight saving time:
  - Day
    Specify the day.
  - Hour
    Specify the hour.
  - Month
    Specify the month.
- **End Date**
  Enter the following values for the end of daylight saving time:
  - Day
    Specify the day.
  - Hour
    Specify the hour.
  - Month
    Specify the month.

Settings with "Recurring" selected

You can create a rule for the daylight saving time changeover.

- **Start Date**
  Enter the following values for the start of daylight saving time:
  - Hour
    Specify the hour.
  - Month
    Specify the month.
  - Week
    Specify the week.
    You can select the first to fourth or the last week of the month.
  - Weekday
    Specify the weekday.
- **End Date**
  Enter the following values for the end of daylight saving time:
  - Hour
    Specify the hour.
  - Month
    Specify the month.
  - Week
    Specify the week.
    You can select the 1st to 4th or the last week of the month.
  - Weekday
    Specify the weekday.

SNTP client

Time-of-day synchronization in the network

SNTP (Simple Network Time Protocol) is used for synchronizing the time in the network. The appropriate frames are sent by an SNTP server in the network.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Description

- **SNTP Client**
  Enable or disable automatic time-of-day synchronization using SNTP.
- **Current System Time** (only available online)
  Shows the current date and current normal time received by the IE switch. If you specify a time zone, the time information is adapted accordingly.

- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place.
- **Last Synchronization Mechanism** (only available online)
  Shows how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time. The time in the "Current System Time" box is adapted accordingly.
- **Daylight Saving Time**
  Shows whether the daylight saving time changeover is active.
  - active (offset +1 h)
    The system time was changed to daylight saving time; in other words an hour was added. The normal time including the time zone continues to be displayed in the "Current System Time" box.
    The normal time including the time zone continues to be displayed in the "Current System Time" box.
  - inactive (offset +0 h)
    The current system time is not changed.
- **SNTP Mode**
  Select the synchronization mode. The following types of synchronization are possible:
  - Listen
    With this mode, the device is passive and receives SNTP frames that deliver the time of day. Settings in the input boxes "SNTP Server Address" and "SNTP Server Port" have no effect in this mode. In this mode, only IPv4 addresses are supported.
  - Poll
    If you select this mode, the input box "Poll Interval[s]" is displayed to allow further configuration. In this mode the settings in the input boxes "SNTP Server Address" and "SNTP Server Port" are taken into account. With this type of synchronization, the device is active and sends a time query to the SNTP server. In this mode, IPv4 and IPv6 addresses are supported.
- **Poll Interval[s]**
  Here, enter the interval between two-time queries. In this box, you enter the query interval in seconds. Possible values are 16 to 16284 seconds.

- **SNTP Server Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the SNTP server.
- **SNTP Server Port**
  Enter the port of the SNTP server.
  The following ports are possible:
  – 123 (standard port)
  – 1025 to 36564
- **Primary**
  The check mark is set for the SNTP server that you create first. If several SNTP servers have been created, the primary server is queried first.

NTP client

Automatic time-of-day setting with NTP

If you require time-of-day synchronization using NTP, you can make the relevant settings here.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Description

- **NTP Client**
  Select this check box to enable automatic time-of-day synchronization with NTP.
- **Secure NTP Client Only**
  When enabled, the device receives the system time from a secure NTP server. The setting applies to all server entries.
  To enable the secure NTP client, you configure the parameters for authentication (key ID, hash algorithm, key).
- **Current System Time** (only available online)
  Shows the current date and current normal time received by the IE switch. If you specify a time zone, the time information is adapted accordingly.
- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place.

- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time.
  The time in the "Current System Time" box is adapted accordingly.
- **Daylight Saving Time**
  Shows whether the daylight saving time changeover is active.
  - active (offset +1 h)
    The system time was changed to daylight saving time; in other words an hour was added. The normal time including the time zone continues to be displayed in the "Current System Time" box.
  - inactive (offset +0 h)
    The current system time is not changed.
- **NTP Server Index**
  Select the index of the NTP server. The server with the lowest index is queried first.
- **NTP Server Address**
  Enter the IPv4 address of the NTP server.
- **NTP Server Port**
  Enter the port of the NTP server.
  The following ports are possible:
  - 123 (standard port)
  - 1025 to 36564
- **Poll Interval[s]**
  Specify the interval between two time queries. Possible values are 64 to 1024 seconds.
- **Key ID**
  Enter the ID of the authentication key.
- **Hash Algorithm**
  Specify the format for the authentication key.

- **Key**
  Enter the authentication key.
- **Key Confirmation**
  Enter the authentication key again to confirm it.

SIMATIC Time Client

Time setting via SIMATIC time client

On this page, you configure time synchronization with the SIMATIC Time Client.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Description

- **SIMATIC Time Client**
  Select this check box to enable the device as a SIMATIC time client.
- **Current System Time** (only available online)
  Shows the current system time.
- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place.
- **Last Synchronization Mechanism** (only available online)
  Shows how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP

Auto logout

On this page, set the times after which there is an automatic logout from WBM or the CLI following user in activity.

If you have been logged out automatically, you will need to log in again.

Note

No automatic logout from the CLI

If the connection is not terminated after the set time, check the setting of the "keepalive" mechanism on the Telnet client.

If the interval is shorter than the configured time, the connection is kept alive although no user data is transferred. You have set, for example, 300 seconds for the automatic logout and the "keepalive" function is set to 120 seconds. In this case, a packet is sent every 120 seconds that keeps the connection up.

- Disable the keepalive mechanism (interval time = 0)
  or
- Set the interval high enough so that the underlying connection is terminated when there is inactivity.

Description

- **Web Based Management [s]**
  Enter the time in seconds for the automatic logout from WBM. If you enter the value 0, the automatic logout is disabled.
- **CLI (TELNET, SSH) [s]**
  Enter the time in seconds for the automatic logout from the CLI. If you enter the value 0, the automatic logout is disabled.

Syslog client

System event agent

Syslog according to RFC 3164 is used for transferring short, unencrypted text messages over UDP in the IP network. This requires a Syslog server.

Requirements for sending log entries

- The Syslog function is enabled on the device.
- The Syslog function is enabled for the relevant event.
- There is a Syslog server in your network that receives the log entries. Since this is a UDP connection, there is no acknowledgment to the sender.
- The IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server is entered in the device.

Description

The page contains the following boxes:

- **Syslog Client**
  Enable or disable the Syslog function.
- **Syslog Server Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server.

The table contains the following columns:

- **Syslog Server Address**
  Shows the IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server.
- **Server Port**
  Enter the port of the Syslog server being used.

Fault Monitoring

Power supply

Note

This page is not available on the SCALANCE W786-2 SFP.

Configure whether or not the power supply should be monitored by the messaging system. Depending on the hardware variant, there are one or two power connectors (Supply 1 / Supply 2). With a redundant power supply, configure the monitoring separately for each individual feed-in line.

If there is no power on one of the monitored lines (Supply 1 or Supply 2) or when the voltage is too low, a fault is signaled by the signaling system.

Note

You will find the permitted operating voltage limits in the operating instructions of the device.

An error causes the fault LED to light up on the device. Depending on the configuration, the error may trigger a trap, an e-mail or an entry in the event log table.

Settings

- **Line 1**
  Enable or disable the monitoring of power connector 1.
- **Line 2** (not available with W760/W720)
  Enable or disable the monitoring of power connector 2.

- **PoE** (not available with W760/W720)
  Enable or disable the monitoring of the power supply via Ethernet.
- **PROFINET redundancy** (not available with W760/W720)
  With the following devices, you can also configure which power supply will be monitored by PROFINET:
  - SCALANCE W788-x RJ-45 variants
  - SCALANCE W748-1 RJ-45
  - SCALANCE W774-1 (RJ-45 and M12 variant)
  - SCALANCE W734-1 RJ-45

Link Change

Configuration of fault monitoring of status changes on connections

On this page, you configure whether or not an error message is triggered if there is a status change on a network connection.

If connection monitoring is enabled, an error is signaled

- when there should be a link on a port and this is missing.
- or when there should not be a link on a port and a link is detected.

An error causes the signaling contact to trigger and the fault LED to light up on the device. Depending on the configuration, the error may trigger a trap, an e-mail or an entry in the event log table.

Description of the displayed boxes

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Setting**
  Select the required setting. The available options are as follows:
  - "-" (disabled)
  - Up
  - Down
  - No Change: The setting in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns

- **Port**
  Shows the available ports.
- **Setting**
  Select the setting. You have the following options:
  - Up
    Error handling is triggered when the port changes to the active status.
    (From "Link down" to "Link up")
  - Down
    Error handling is triggered when the port changes to the inactive status.
    (From "Link up" to "Link down")
  - "-" (disabled)
    The error handling is not triggered.

EtherNet/IP

EtherNet/IP

On this page, you configure the mode of EtherNet/IP.

Note

Devices with two Ethernet interfaces

On devices with two Ethernet interfaces only one of the interfaces (P1 or P2) may be used for the EtherNet/IP configuration.

This applies to the following devices:

- SCALANCE W786-2 SFP .
- SCALANCE W774-1 RJ-45
- SCALANCE W774-1 M12 EEC
- SCALANCE W734-1 RJ-45.

Description of the displayed boxes

The page contains the following boxes:

- **EtherNet/IP Device Diagnostics**
  Shows whether EtherNet/IP is enabled ("On") or disabled ("Off").
- **EtherNet/IP Device Diagnostics for next boot**
  Set whether EtherNet/IP will be enabled ("On") or disabled ("Off") after the next device restart.

Note

EtherNet/IP and PROFINET

When EtherNet/IP is turned on, PROFINET is turned off. The switchover from EtherNet/IP and PROFINET has no effect on DCP.

Note

PROFINET AR Status

If a PROFINET connection is established; in other words the PROFINET AR status is "Online", you cannot enable EtherNet/IP.

PROFINET

Settings for PROFINET

This page shows the PROFINET AR status and the device name.

Description of the displayed boxes

The page contains the following boxes:

- **PROFINET Mode**
  Shows whether PROFINET is enabled ("On") or disabled ("Off").
- **PROFINET Mode for next boot**
  Set whether PROFINET will be enabled ("On") or disabled ("Off") after the next device restart.

Note

PROFINET and EtherNet/IP

When PROFINET is turned on, EtherNet/IP is turned off. The switchover from PROFINET and EtherNet/IP has no effect on DCP.

Note

PROFINET AR Status

If a PROFINET connection is established; in other words the PROFINET AR status is "Online", you cannot disable PROFINET.

- PROFINET AR Status
  This box shows the status of the PROFINET connection; in other words whether the device is connected to a PROFINET controller "Online " or "Offline".
  Here, online means that a connection to a PROFINET controller exists, that this has downloaded its configuration data to the device and that the device can send status data to the PROFINET controller. In this status known as "in data exchange", the parameters set via the PROFINET controller cannot be configured.
- PROFINET Name of Station
  This box displays the PROFINET device name according to the configuration in HW Config of STEP 7.

Note

Devices with two Ethernet interfaces

With devices that have two Ethernet interfaces, only interface P1 should be used for the PROFINET configuration because LLPD frames can only be sent and received via interface P1. They are blocked at interface P2 and are also not forwarded between the interfaces.

This applies to the following devices:

- SCALANCE W786-2 SFP
- SCALANCE W774-1 RJ-45
- SCALANCE W774-1 M12 EEC
- SCALANCE W734-1 RJ-45

SCALANCE W700 and STEP 7

The Ethernet interface can be configured in STEP 7 if the following requirements are met:

- STEP 7 V13 Update 3 with HSP0107 or
- STEP7 version 5.5.4 with GSDML version 2.31

The diagnostics functions can also be used. The WLAN interface cannot be configured with STEP 7.

PROFINET for client devices

If a client is to be used as a PROFINET device, the MAC address of the client must be specified as follows (MAC Mode):

- Own
  In the network beyond the device, only IP communication and no PROFINET is possible.
- Layer 2 Tunnel
  The client and the devices downstream from it can be used as PROFINET devices.

  Note

  If "Automatic" or "Manual" is configured as the MAC mode for a client, this device cannot be used as a PROFINET device.

PLUG

Configuration

Note

The page is only available if there is an online connection to the device.

This page is only available for SCALANCE W780 / W 770 / W740 / W730.

NOTICE

Do not remove or insert a C-PLUG / KEY-PLUG during operation!

A PLUG may only be removed or inserted when the device is turned off.

The device checks whether or not a PLUG is inserted at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid KEY-PLUG was inserted in the device, the device changes to a defined error state following the restart. With SCALANCE W, the available wireless interfaces are deactivated in this case.

If the device was configured at some time with a PLUG, the device can no longer be used without this PLUG. To be able to use the device again, reset the device to the factory settings.

Information about the configuration of the C-PLUG / KEY-PLUG

This page provides detailed information about the configuration stored on the C-PLUG or KEY-PLUG. It is also possible to reset the PLUG to factory defaults or to load it with new contents.

Note

The action is only executed after you click the "Set Values" button.

The action cannot be undone.

If you decide against executing the function after making your selection, click the "Refresh" button. As a result the data of this page is read from the device again and the selection is canceled.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Settings

The table has the following rows:

- **Status**
  Shows the status of the PLUG. The following are possible:
  - ACCEPTED
    There is a PLUG with a valid and suitable configuration in the device.
  - NOT ACCEPTED
    Invalid or incompatible configuration on the inserted PLUG.
  - NOT PRESENT
    There is no C-PLUG or KEY-PLUG inserted in the device.
  - FACTORY
    PLUG is inserted and does not contain a configuration. This status is also displayed when the PLUG was formatted during operation.
  - MISSING
    There is no PLUG inserted. Functions are configured on the device for which a license is required.
- **Device Group**
  Shows the SIMATIC NET product line that used the C-PLUG or KEY-PLUG previously.
- **Device Type**
  Shows the device type within the product line that used the C-PLUG or KEY-PLUG previously.
- **Configuration Revision**
  The version of the configuration structure. This information relates to the configuration options supported by the device and has nothing to do with the concrete hardware configuration. This revision information does not therefore change if you add or remove additional components (modules or extenders), it can, however, change if you update the firmware.
- **File System**
  Displays the type of file system on the PLUG.

NOTICE
New file system UBI As of firmware version 2.0, UBI is the standard file system for the C-PLUG or KEY-PLUG. If a C-PLUG with the previous file system IECP is detected in such a device, this C-PLUG will be formatted for the UBI file system and the data will be rewritten to the C-PLUG. This change in the file system also occurs following a firmware update to V2.0. A downgrade to the previous version of the firmware then presents a problem. The firmware can neither read nor write the C-PLUG or KEY-PLUG and it is not even possible to "Erase PLUG to factory default".

- **File System Size [bytes]**
  Shows the maximum storage capacity of the file system on the C-PLUG.
- **File System Usage**
  Displays the storage space in use in the file system of the C-PLUG.

- **Firmware on PLUG**
  When enabled, the firmware will be stored on the PLUG. This means that automatic firmware updates/downgrades can be made with the PLUG.
- **Info String**
  Shows additional information about the device that used the PLUG previously, for example, article number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.
  If a PLUG was configured as a PRESET PLUG this is shown here as additional information in the first row. For more detailed information on creating and using a PRESET PLUG refer to the section "Maintenance (page 363)".
- **Modify PLUG**
  Select the required setting. You have the following options for changing the configuration on the C-PLUG or KEY-PLUG:
  - Write current configuration to PLUG
    This option is available only if the status of the PLUG is "NOT ACCEPTED" or FACTORY. The configuration in the internal flash memory of the device is copied to the PLUG.
  - Erase PLUG to factory default
    Deletes all data from the C-PLUG and triggers low-level formatting.

License

NOTICE
Do not remove or insert a C-PLUG / KEY-PLUG during operation! A PLUG may only be removed or inserted when the device is turned off. The device checks whether or not a PLUG is present at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid KEY-PLUG was inserted in the device, the device changes to a defined error state following the restart. With SCALANCE W, the available wireless interfaces are deactivated in this case. If the device was configured at some time with a PLUG, the device can no longer be used without this PLUG. To be able to use the device again, reset the device to the factory settings.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Information about the license of the KEY-PLUG

A C-PLUG can only store the configuration of a device. In addition to the configuration, a KEY-PLUG also contains a license that enables certain functions of your SIMATIC NET device.

Note

Information only available online

You can only view the information on this page if there is an online connection to the device.

This page is only available for SCALANCE W780 / W 770 / W740 / W730.

Displayed values

- **Status**
  Shows the status of the KEY-PLUG. The following are possible:
  - ACCEPTED
    The KEY-PLUG in the device contains a suitable and valid license.
  - NOT ACCEPTED
    The license of the inserted KEY-PLUG is not valid.
  - NOT PRESENT
    No KEY-PLUG is inserted in the device.
  - MISSING
    There is no KEY-PLUG or a C-PLUG with the status "FACTORY" inserted in the device. Functions are configured on the device for which a license is required.
  - WRONG
    The inserted KEY-PLUG is not suitable for the device.
  - UNKNOWN
    Unknown content of the KEY-PLUG.
  - DEFECTIVE
    The content of the KEY-PLUG contains errors.
- **Article number**
  Shows the article number of the KEY-PLUG. The KEY-PLUG is available for various functional enhancements and for various target systems.
- **Serial Number**
  Shows the serial number of the KEY-PLUG.
- **Info String**
  Shows additional information about the device that used the KEY-PLUG previously, for example, article number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.

Note

When you save the configuration, the information about whether or not a KEY-PLUG was inserted in the device at the time is also saved. This configuration can then only work if a KEY-PLUG with the same order number / license is inserted. This applies regardless of whether or not iFeatures are configured.

Ping

Reachability of an address in an IP network

Note

The page is only available if there is an online connection to the device.

With the ping function, you can check whether a certain address is reachable in the network.

Description

The page contains the following boxes:

- **Dest. address**
  Enter the IPV4, IPv6 address or the FQDN (Fully Qualified Domain Name) of the device.
- **Repeat**
  Enter the number of ping requests.
- **DNS Resolution**
  Select the IP address type in which an entered FQDN will be resolved.
  - Auto
    In this mode, the IP address type is selected automatically.
  - IPv4
    The entered FQDN will be resolved in an IPv4 address.
  - IPv6
    The entered FQDN will be resolved in an IPv6 address.
- **Out Interface for IPv6**
  This selection is only required when the destination address is a multicast or a link local address.
  - "-" (factory setting)
  - Select the relevant IPv6 interface.
- **Ping**
  Click this button to start the ping function.

- **Ping Output**
  This box shows the output of the ping function.
- **Delete**
  Click this button to delete the ping output.

Configuring interfaces

Ethernet

Overview

The page shows the configuration for the data transfer for all ports of the device.

Description

- **Port**
  Shows the configurable ports.
- **Port name**
  Shows the name of the port.
- **Status**
  Shows whether the port is on or off. Data traffic is possible only over an enabled port.
- **Operational State** (only available online)
  Displays the current operational status. The operational status depends on the configured "Status" and the "Link".
  The available options are as follows:
  - up
    You have configured the status "enabled" for the port and the port has a valid connection to the network.
  - down
    You have configured the status "disabled" or "Link down" for the port or the port has no connection.
- **Link** (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - up
    The port has a valid link to the network, a link integrity signal is being received.
  - down
    The link is down, for example because the connected device is turned off.
- **Mode** (only available online)
  Shows the transfer parameters of the port
- **MTU**
  Shows the maximum packet size.

- **Negotiation**
  Shows whether the automatic configuration is enabled or disabled.
- **MAC Address** (only available online)
  Shows the MAC address of the port.

Configuration

On this page, you configure the Ethernet ports of the device.

Note

SCALANCE W786-2 SFP

The two SFP ports of the SCALANCE W786-2 SFP cannot be assigned parameters or diagnosed individually.

Settings

- **Port**
  Select the port to be configured.
- **Status**
  Specify whether the port is enabled or disabled.
  - enabled
    The port is enabled. Data traffic is possible only over an enabled port.
  - disabled
    The port is disabled but the connection remains.
  - Link down
    The port is disabled and the connection to the partner device is terminated.
- **Port name**
  Enter a name for the port.
- **MAC Address** (only available online)
  Shows the MAC address of the port.

- **Mode Type**

  Note

  This parameter cannot be configured for:
  - SCALANCE W786-2 SFP
  - SCALANCE W770
  - SCALANCE W760
  - SCALANCE W730
  - SCALANCE W720

  The value is permanently set to “Auto negotiation“.

  Shows the transmission speed and the transmission method of the port. You make the settings for "Autonegotiation" and "Transmission rate" in the port options.

  Note

  Before the port and partner port can communicate with each other, the settings must match at both ends.

- **Mode** (only available online)
  Shows which transmission speed and which transmission mode is currently being used.
- **Negotiation**
  Shows whether the automatic configuration of the connection to the partner port is enabled or disabled.
- MTU ((Maximial Transmission Unit)
  Enter the packet size above which packets are fragmented.
- **OperState**
  Displays the current operational status. The operational status depends on the configured "Status" and the "Link".
  The available options are as follows:
  - up
    You have configured the status "enabled" for the port and the port has a valid connection to the network.
  - down
    You have configured the status "disabled" or "Link down" for the port or the port has no connection.
- **Link** (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - Up
    The port has a valid link to the network, a link integrity signal is being received.
  - Down
    The link is down, for example because the connected device is turned off.

Changing the port configuration

Note

Optical ports only work with the full duplex mode and at maximum transmission rate. As a result, the following settings cannot be made for optical ports:

- Automatic configuration
- Transmission speed
- Transmission technique

Note

With various automatic functions, the device prevents or reduces the effect on other ports and priority classes (Class of Service) if a port is overloaded. This can mean that frames are discarded even when flow control is enabled.

Port overload occurs when the device receives more frames than it can send, for example as the result of different transmission speeds.

WLAN

Basic

Note

To configure the WLAN interface, you must always specify the country code first. Some parameters are dependent on the country setting, for example the transmission standard.

Settings

- **Country Code**
  Specify the country in which the device will be used.
  You do not need to know the data for the specific country, the channel division and output power are set by the device according to the country you select.

  Note
  Locale setting

  The correct country setting is mandatory for operation complying with the approvals. Selecting a country different from the country of use can lead to legal prosecution.

- **Device Mode**
  Mode of the device. The mode can only be changed for access points.
  The following modes are possible:
  - AP: Access point mode
  - Client: Client mode

  Note

  After changing the mode, a message is displayed. If you confirm the message with "Yes", the configuration settings are updated in the changed mode.

- **Radio**
  Shows the available WLAN interfaces.
- **Enabled**
  Status of the WLAN interface. To enable the WLAN interface, enable the setting.

  Note
  Enabling the WLAN interface

  The WLAN interfaces are disabled when the device is supplied. The WLAN interfaces are can be enabled once the country and the antenna settings are configured.

- **Radio Mode**
  Shows the mode of the WLAN interface.
- **Frequency Band**
  Specify the frequency band. In client mode, dual-frequency operation is also possible.
  - 2.4 GHz
  - 5 GHz
  - 2.4 GHz + 5 GHz (only in client mode)

  Note
  Configuring WLAN interfaces of the W786-2IA RJ-45 for different frequency bands

  If both WLAN interfaces are configured for the same frequency band on this device, there may be mutual influence or interference. This applies in particular when there is a high data throughput.

- **WLAN Mode 2.4 GHz/WLAN Mode 5 GHz**
  Select the required transmission standard for the configured frequency band.
  The selection depends on the country setting.
  - Auto (in client mode only)
    The transmission standard is determined automatically (2.4 GHz + 5 GHz).
  - 802.11a
    The transmission standard IEEE 802.11a (5 GHz) is set.
  - 802.11g
    The transmission standard IEEE 802.11g (2.4 GHz) is set. This transmission standard is downwards compatible with IEEE 802.11b
  - 802.11n
    The transmission standard IEEE 802.11n (2.4 GHz and 5 GHz) is set. This transmission standard is downwards compatible with IEEE 802.11a and IEEE 802.11g.
  - 802.11n only
    The transmission standard IEEE 802.11n (2.4 GHz and 5 GHz) is set. This transmission standard is downwards compatible with IEEE 802.11a and IEEE 802.11g.

Note

Fragmentation length

If you select the transmission standard "802.11n", "802.11n only" or "Auto" (only in client mode), you cannot set the threshold value of the fragmentation length see "Fragmentation Length Threshold" in "Interfaces > WLAN > Advanced".

- **DFS (802.11h)**
  Enables or disables the "Dynamic Frequency Selection (DFS)" function.

Note

RCoax 5 GHz and DFS

In the USA and in countries that follow the FCC (Federal Communication Commission) when operating with DFS (Dynamic Frequency Selection), the IWLAN RCoax Cable 5 GHz may not be used. The current status of the approvals can be found on the Internet at the address (http://www.siemens.com/funkzulassungen).

  - Enabled
    If you have enabled this function, additional DFS channels from the 5 GHz band are available to you. These channels are country-specific and are subject to certain DFS regulations.
    If the access point discovers a disturbance on the current channel, it changes automatically to an alternative channel and the current channel is blocked for 30 minutes. The disturbance can originate from a primary user or radar interference.
    Before the access point starts the communication on a channel it searches 60 seconds for primary users on the channel. During this time the access point does not send beacons. If signals are found on the channel, the access point changes channel and repeats the check. Only when no signals from primary users are detected after 60 seconds does the access point send on a channel. The access point also searches for primary users during operation.
  - Disabled
    The DFS function is not used.

- **Outdoor Mode**
  - Enabled
    If you have enabled the outdoor mode, you only have the channels available that are permitted for outdoor operation.
  - Disabled
    If you have disabled the outdoor mode, you only have the channels available that are permitted for operation in a building.
- **Maximum Tx Power**
  Specify the transmit power of the device. If the transmit power is set too high the received signal at the client may be overmodulated. Check the received signal strength at the client (dBm). It may be necessary to reduce the transmit power depending on the antennas being used to avoid exceeding the maximum legal transmit power. Reducing the transmit power effectively reduces cell size.

  Note

  The maximum possible transmit power varies depending on the channel and data rate. For more detailed information on transmit power, refer to the documentation "Characteristics radio interface".

  Note

  If both interfaces of access points with two WLAN interfaces are operated in the same frequency range, this may cause wireless interference on one or both interfaces at a transmit power higher than 15 dBm.

- **Tx power check (only available online)**
  Indicates whether the settings that have been made will violate the permitted transmit power restrictions of the selected country. The following parameters influence this calculation: max. Maximum Tx Power, Antenna Gain, Additional Attenuation.
  The following displays can appear:
  - Allowed
    The channels can be used with the current settings.
  - Not Allowed (Some Channels)
    Among the channels, there are some on which the current transmit power exceeds the maximum permitted transmit power.
  - Not Allowed (All Channels)
    No permitted operation is possible. The transmit power is too high.
  - Controlled automatically by iREF.
    Controlled by the iREF function.

Advanced

You only need to adapt the settings on this page if the device cannot be used as it is intended with the default settings.

Settings

- **Radio**
  Shows the available WLAN interfaces.
- **Beacon Interval [ms]** (Only in access point mode)
  Specify the interval (40 - 1000 ms) at which the access point sends beacons. Beacons are packets that are sent cyclically by an access point to inform clients of its existence.
- **DTIM**(Only in access point mode)
  The DTIM interval (1-15) specifies the number of beacons to be sent before the access point sends the collected packets (broadcast, unicast, multicast) to the client.
  - If you enter a "1", the access point transmits broadcast, unicast and multicast packets directly after each beacon (recommended setting for normal network environments).
  - If you entered a "5", this would mean that the access point collects the packets and sends them after every fifth beacon.

  Increasing this value allows a longer sleep mode for the clients but means a greater delay for packets.
- **RTS/CTS Threshold [Bytes]**
  RTS/CTS (Request To Send/Clear To Send) is a method for avoiding collisions. The method is based on the exchange of status information prior to sending the actual data (Hidden node problem). To minimize the network load due to additional protocol traffic, this method is used only as of a certain packet size. You specify the packet size with the RTS/CTS Threshold parameter.
- **Fragmentation Length Threshold [Bytes]**
  Specify the maximum packet size transferred on the wireless link. Large packets are divided up into small packets prior to transmission and then reassembled into the original size after they have been received. This can be beneficial if the transmission quality is poor because larger packets are more difficult to transmit. However fragmentation into smaller packets means a poorer throughput.

  Note

  You can only edit this value if the you have set the transmission standard "802.11g" (2.4 GHz) or "802.11 a" (5 GHz), see "WLAN Mode" in "Interfaces > WLAN > Basic".

- **HW Retries**
  Specify the number of hardware retries. The hardware repetition is performed by the WLAN chip itself when it tries to repeat an unacknowledged packet immediately.
  If all hardware repetitions were unsuccessful, the packet is deleted.

  Note

  If you have enabled iPCF or iPCF-MC, you can set up to 16 hardware retries.

- **Multi Radar Detection** (only in access point mode)
  - Enabled
    This function is only available if you have enabled the "DFS" function on the "Basic" page. This function is suitable for systems with several access points connected via an Ethernet network and that send on the same channel. When an access point detects a radar signal it distributes this information to all connected access points. If at least one further access point verifies the radar signal within 40 ms, all connected access points are informed. All the devices sending on this channel change to a different channel. The channel is blocked for 30 minutes for the access points in the network. If you have configured "Auto" for the channel on the "Interfaces > WLAN > AP" page, the function cannot be used reliably. In this case the verification of the radar signals is only possible when at least two connected access points happen to transmit on the same channel. If only one access point detects a signal on a channel, it treats this as a valid radar signal.
  - Disabled
    The function is not used. When an access point detects a radar signal it changes to another channel.. The configured channel is no longer taken into account.
- **Prefer Configured DFS Channel** (only in access point mode)
  - Enabled
    This function is only available if you have enabled the "DFS" function on the "Basic" page. If the configured channel of a WLAN interface was blocked due to radar detection and is released again after 30 minutes the access point changes automatically to the configured channel. Before the access point starts the communication on the configured channel it searches 60 seconds for primary users on the channel. During this time the access point does not send beacons. If signals are found on the channel, the access point changes channel and repeats the check. Only when no signals from primary users are detected after 60 seconds does the access point send on the channel. If you have configured "Auto" for the channel on the "Interfaces > WLAN > AP" page, the device does not have a configured channel to which it can return.
  - Disabled
    The function is not used.

Antennas

Overview

Overview of IWLAN antennas:

Type of antenna	Frequency (GHz)	Antennas	SCALANCE W780/W740	SCALANCE W770/W730	SCALANCE W760/W720	SCALANCE W1780/W1740
Omnidirectional	2.4	ANT792-6MN	●	●	●	●
	2.4 and 5	ANT795-4MA	●	●	●	●
		ANT795-4MB	●	●	●	●
		ANT795-4MC	●	●		●
		ANT795-4MD	●	●		●
		ANT795-4MX	●	●		●
		ANT795-6MN	●	●	●	●
		ANT795-6MT	●			●
		ANT795-6MP	●	●	●	●
	5	ANT793-6MN	●	●	●	●
Sector	2.4 and 5	ANT795-6DC	●	●	●	●
	5	ANT793-6DG	●	●		●
		ANT793-6DT	●			●

Type of antenna	Frequency (GHz)	Antennas	SCALANCE W780/W740	SCALANCE W770/W730	SCALANCE W760/W720	SCALANCE W1780/W1740
Directional	2.4	ANT792-8DN	●			●
	5	ANT793-8DP	●	●	●	●
		ANT793-8DJ	●	●		●
		ANT793-8DK	●	●		●
		ANT793-8DL	●	●		●
RCoax	2.4	RCoax waveguide 2.4 GHz	●	●	●	●
		ANT792-4DN	●	●	●	●
	5	RCoax waveguide 5 GHz	●	●	●	●
		ANT793-4MN	●	●	●	●
Mobile wireless for GSM	2.4 and 5	ANT896-6MM				●

The antenna name provides information about the properties of the antennas listed in the IWLAN antenna overview:

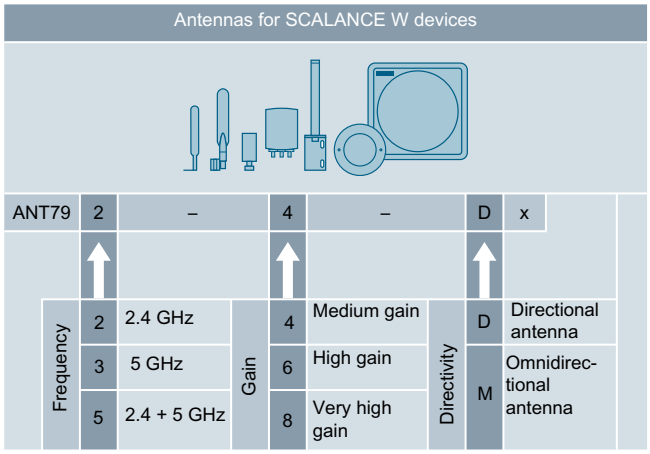

Antennas

On this page, you configure the settings for the connected external antennas.

Note

Transmission disruptions without an antenna connected

A WLAN interface can have up to three antenna connectors. A the first connector, there must be always be an antenna connected as soon as the associated WLAN Interface is turned on.

Connectors that are not used must have a 50 Ω terminating resistor fitted.

If no antenna is connected, the relevant WLAN interface must also be disabled for Rx and Tx. Otherwise, there may be transmission disruptions.

Settings

- **Connector**
  Shows the name of the relevant antenna connection.
- **Antenna Type**
  Select the type of external antenna connected to the device. If the type of your external antenna is not available, select the entry "User defined".
  If you terminate an antenna connection using a 50 Ω terminating resistor, select the entry "Not used (Connect 50 Ohm Termination)".
- **Antenna Gain**
  If you select the "User defined" entry for the "Antenna Type", enter the antenna gain manually in the "dBi" unit.
  - **Antenna Gain 2.4 GHz [dBi]**
    Here, enter the antenna gain the antenna has in the 2.4 GHz frequency band.
  - **Antenna Gain 5 GHz [dBi]**
    Here, enter the antenna gain the antenna has in the 5 GHz frequency band.

- **Cable Length [m]**
  Enter the length of the flexible antenna connecting cable between the device and the external antenna in meters.
- **Additional Attenuation [dB]**
  Here, specify the additional attenuation caused, for example by an additional splitter.
- **Antenna Mode**
  Specify the use of the antenna. For antenna connector 1 (R1 A1 and R2 A1), the entry cannot be changed.
  - TX
    For sending only
  - RX
    For receiving only
  - RX\TX
    For receiving and sending

  The following table shows which combinations are possible:

R1 A1 / R2 A1	R1 A2 / R2 A2	R1 A3 / R2 A3
RX\TX	RX\TX	RX\TX
RX\TX	RX\TX	RX
RX\TX	RX	RX
RX\TX	RX\TX	TX
RX\TX	TX	TX
RX\TX	RX\TX	-- 1)
RX\TX	TX	-- 1)
RX\TX	RX	-- 1)
RX\TX	-- 1)	-- 1)

1) Antenna type "50 Ω termination"

Permitted channels

For communication, a specific channel within a frequency band is used. You can either set this channel specifically or configure so that the channel is selected automatically.

Settings

Table 1 contains the following columns:

- **Radio**
  Shows the available WLAN interfaces.
- **Use Allowed Channels only**
  If you enable the setting, you restrict the selection of channels via which the AP or the client is allowed to establish the connection.
  In the following tables, you define the
  - channels that the AP can use to establish a wireless cell when the "Auto" channel setting is enabled.
  - the channels on which the client searches for an AP.

  The tables are divided up according to frequency bands.
  When disabled, the channels available based on the settings (country code, antennas, transmit power etc.) are used.

Above the tables for the frequency bands, you will find the following setting:

- **Select / Deselect all**
  - Enabled
    All channels are activated.
  - Disabled
    The first valid channel of the frequency band remains enabled. Enable the required channel.

The tables of the frequency bands have the following columns:

- **Radio**
  Shows the available WLAN interfaces.
- **Radio Mode**
  Shows the mode.
- **Channel number**
  To specify the valid channels for the required frequency band, select the appropriate check box for the channel number.
  The table displays the permitted channels of the country. Only the valid channels can be enabled. Invalid channels are grayed out and cannot be enabled.

  Note

  To specify the channels, the setting "Use Allowed Channels only" must be enabled.

802.11n

Properties of 802.11n

With the IEEE 802.11n standard, it is possible to put together individual data packets in one larger data packet, the A-MPDU and A-MSDU data packets. This achieves a higher data throughput.
On this page, you make the settings for the A-MPDU and A-MSDU data packets. Some of the settings depend on the set transmission standard and the selected channel width.

Settings

The table has the following columns:

- Radio
  Shows the available WLAN interfaces.
- A-MPDU
  Enables or disables the sending of A-MPDU data packets. If this setting is disabled, only A-MPDU data packets are received.
- A-MPDU Limit [frames]
  Specify the number of individual subframes grouped together in one A-MPDU data frame.
- A-MPDU Limit [Bytes]
  Specify the maximum size of the A-MPDU frame.
- A-MSDU
  Aggregated MAC Service Data Unit (A-MSDU)Enables or disables that several MSDUs with the same destination address are bundled into an A-MSDU and are sent together. This reduces the network load. Due to their shorter maximum length A-MSDUs are more suitable for the bundling of several shorter frames. If this check box is disabled, A-MSDU data packets are received but not sent.
- A-MSDUPacket Size [bytes]
  Specify the maximum size of the A-MSDU data packet.
  Range of values: 50 - 200 bytes
  Default: 100 bytes
- Guard Interval [ns](only in access point mode)
  Select the send pause that must be kept to between two transmitted OFDM symbols.
  The following settings are possible. The selection depends on the selected transmission standard.
  - 400 (short)/800 (long): The setting 400 ns is optional. Depending on the signal quality, packets can be sent with a send pause of 400 ns or 800 ns.
  - 800 (long): The send pause is 800 ns.

Access point

Note

The page is available only in access point mode.

Settings

Table 1 has the following columns:

- **Radio**
  Shows the available WLAN interfaces.
- **Channel**
  Specify the main channel.
  If you want the access point to search for a free channel itself, use "Auto". The selection of channels used by an access point when establishing a wireless cell can be restricted. For "Channel selection", enable the setting "Use Allowed Channels only". If you want to use a fixed channel, select the required channel from the drop-down list.

  Note

  SCALANCE W780/W740: Channel spacing with WLAN interfaces

  If you use a second WLAN interface, make sure that you have adequate channel spacing.

- **Alternative DFS Channel (802.11h)**
  If you have enabled the "DFS" function on the "Basic" page, specify the alternative channel here. If you want the access point to search for a free channel itself, use "Auto".
  If a primary user was detected both on the main and alternative channel, the access point automatically searches for a free channel.
  If you want to use a fixed channel, select the required channel from the drop-down list.
- **HT Channel Width [MHz]**
  You can specify the channel bandwidth only with the IEEE 802.11n transmission standard. The following settings are possible.
  - 20
    Channel bandwidth 20 MHz
  - 40 up
    Channel bandwidth 40 MHz. The configured channel and the neighboring channel above it are used.
  - 40 down
    Channel bandwidth 40 MHz. The configured channel and the neighboring channel below it are used.

    Note

    Channel bandwidth 40 MHz and frequency band 2.4 GHz

    If the access point detects another access point on the configured channel or on neighboring channels, the access point changes the channel bandwidth from 40 MHz to 20 MHz. If you set a "free" channel on the access point, the access point uses the channel bandwidth 40 MHz.

Table 2 has the following columns:

- **Radio**
  Shows the available WLAN interfaces in this column.
- **Available Channels**
  Shows the permitted channels. The display depends on the wireless approvals of the currently selected country and the settings for "Channel selection".

Table 3 has the following columns:

- Radio
  Shows the WLAN interface.
- Port
  Shows the VAP interface.
- Enabled
  To use the required VAP interface, select this check box.
- SSID
  Enter the SSID of the WLAN. The length of the character string for SSID it is 1 to 32 characters.
  The ASCII characters 0x20 to 0x7e are used for the SSID.
- Broadcast SSID
  - deactived
    The SSID is no longer sent in the beacon frame of the access point. This means that the SSID is not visible for other devices. Only clients that know the SSID of the access point and that are configured with it can connect to the access point. The "Any SSID" option must be disabled on these clients.
  - activated
    The SSID is sent in the Beacon frame of the access point and is visible for other devices. This means that clients on which the "Any SSID" setting is enabled can also connect to the access point.

  Note

  Since no encryption is used for the SSID transfer, this function can only provide basic protection against unauthorized access. The use of an authentication method (for example WPA2 (RADIUS) or WPA2-PSK if this is not possible) provides higher security. You must also expect that certain end devices may have problems with access to a hidden SSID.

- WDS only
  If you enable this setting, the access point only supports communication via WDS. In WDS mode, all access points must use the same channel.
- WDS ID
  Enter the WDS ID. The WDS ID can be a maximum of 32 characters long.
  To establish a WDS connection, enter this WDS ID for the WDS partner.
  ASCII characters 0x20 to 0x7e are used for the WDS ID.

AP WDS

In normal operation, the access point is used as an interface to a network and communicates with clients. There are, however, situations in which several access points need to communicate with each other, for example to extend wireless coverage or to set up a wireless backbone. This mode is possible with WDS (Wireless Distributed System).

Note

This tab is available only in access point mode.

Settings

- **Radio**
  Shows the available WLAN interfaces.
- **Port**
  Shows the port.
- **Port enabled**
  Enables the WDS interface.
- **Connection over**
  Specify the VAP interface via which the WDS connection is established. Both the MAC address of the VAP as well as security settings (for example WPA2) are used.
- **Partner ID Type**
  Specify the type of WDS communication.
  - MAC Address
    The MAC address is used. The "Partner WDS ID" input box is grayed out. For "Partner MAC", enter the MAC address of the WDS partner.
  - WDS ID
    The WDS ID is used. The "Partner MAC" input box is grayed out. Enter the WDS ID of the WDS partner in "Partner WDS ID". Use this option if you want to replace the access point later using the C-PLUG or KEY-PLUG.
- **Partner MAC**
  Enter the MAC address of the WDS partner. Can only be edited if "MAC Address" is set for the "Partner ID Type".
- **Partner WDS ID**
  Enter the WDS ID of the WDS partner. Can only be edited if "WDS ID" is set for the "Partner ID Type". For the WDS ID, the ASCII characters 0x20 to 0x7e are permitted.

Note

Matching security settings in WDS mode

In WDS mode, make sure that the security settings match up for all devices involved. If settings are incorrect or not compatible on the individual devices, no data exchange is possible due to incorrect authentication. Avoid the "Auto" setting in the security settings of the Basic Wizard, because with this setting, synchronization of the security settings between the access points is not possible.

Note

In WDS operation, the following restrictions apply to all access points involved:

- All access points that will communicate with each other must use the same channel, the same transmission procedure and the same data rate.
- You can select either WEP or WPA(2)-PSK as the encryption method.
  You configure the security settings in the assigned VAP interface: "Security > WLAN > Basic"
  You cannot use authentication with a RADIUS server for a WDS connection.
- In the IEEE 802.11h transmission mode, it is not practical to select the WDS mode. In WDS mode, all access points must use the same channel. If a signal from a primary user is detected by an access point, the channel is changed automatically and the existing connection is then terminated.

AP 802.11a/b/g Rates

Note

The page is only

- available in access point mode.
- Can be configured if "802.11a", "802.11g" or "802.11n" is set for the WLAN mode.

The page shows the available data transmission speeds for the WLAN mode 802.11a/b/g. If necessary, you can change the data transmission speeds. Otherwise, we recommend that you retain the default setting for data transmission speeds. The access point will then use only the selected data transmission speeds for communication with the clients.

Settings

Table 1 has the following columns:

- **Radio**
  Specifies the WLAN interface to which the information relates.
- **Use selected data rates only**
  If you select this setting, you can specify the data transmission speeds for the required WLAN interface.
  If this setting is disabled, the default values are used.

Radio
Select the WLAN interfaces displayed in Table 3 (Data rate).

Table 2 has the following columns:

- **All data rate settings**
  Shows that the setting is valid for all entries in Table 3.
- **Enabled / Basic settings**
  Select the required setting. If "No Change" is selected, the entry in table 3 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 3

Table 3 (Data Rate) has the following columns:

- **Radio**
  Specifies the WLAN interface to which the information relates.
- **Data Rate [Mbps]**
  Shows the supported data transmission speeds in megabits per second.

- **Enabled**
  Enable the setting to assign the required data transmission speed to the WLAN interface.

  Note

  You need to enable at least one data transmission speed.

- **Basic**
  Enable the option to declare the required data transmission speed as "Basic". The setting specifies that a client must be capable of the required speed to be able to connect to the access point. The setting can only be enabled if an available data rate has been selected.

  Note

  At least one data transmission speed needs to be specified as "Initial Setup".

Default Values
The "Default Values" button sets the selection of values to conform with the standard.

AP 802.11n data rates

Note

The page is only
- available in access point mode.
- Can be configured if "802.11n only" or "802.11n" is set for the WLAN mode.

The page shows the available data transmission speeds (MCS = Modulation and Coding Schemes) for the WLAN mode 802.11n. You can select any combination of these data transmission speeds. The access point will then use only the selected data transmission speeds for communication with the clients.

Settings

Table 1 has the following columns:

- **Radio**
  Specifies the WLAN interface to which the information relates.
- **Use selected data rates only**
  If you select this setting, you can specify the data transmission speeds for the required WLAN interface.
  If this setting is disabled, the default values are used. As default, the setting is disabled.

Radio
Select the WLAN interfaces displayed in Table 3 (MCS Index).

Table 2 has the following columns:

- All data rate settings
  Shows that the setting is valid for all entries in Table 3.
- Enabled
  Select the required setting. If "No Change" is selected, the entry in table 3 remains unchanged.
- Transfer to table
  If you click the button, the setting is adopted for all ports of table 3.

Table 3 (MCS Index) consists of the following columns:

- Radio
  Specifies the WLAN interface to which the information relates.
- MCS Index
  Shows the supported MCS indexes. The displayed MCS indexes depend on the settings "Antenna Type" and "Antenna Mode". You will find the settings in "Interfaces > WLAN > Antennas". If, for example, you only use one antenna, only the MCS 0 to 7 are displayed.
- Streams
  Shows the maximum possible number of parallel data streams that can be transmitted with the selected MCS index.
- Data Rate [Mbps]
  Shows the supported data transmission speeds in megabits per second. The displayed data transmission speeds depend on the settings "Guard Interval" and "HT Channel Width". The "HT Channel Width" setting can be found in "Interfaces > WLAN > Access Point". The "Guard Interval" setting can be found in "Interfaces > WLAN > 802.11n"
- Enabled
  Enable the setting to assign the required data transmission speed to the WLAN interface.

  Note

  You need to enable at least one MCS index.

Default Values
The "Default Values" button sets the selection of values to conform with the standard.

Client

Connecting to a network

On this page, you can specify how the device connects to a network as client.

Note

This page is only available for clients or access points in client mode.

Settings

Table 1 has the following columns:

- **Radio**
  Shows the available WLAN interfaces.

  Note
  WLAN interface disabled

  The WLAN interface will be disabled unless at least one SSID is configured or the setting "Any SSID" is enabled.

- **MAC Mode**
  Specify how the MAC address is assigned to the client. The following are possible:
  - Automatic
    The client automatically adopts the source MAC address of the first frame that it receives over the Ethernet interface.
  - Manual
    If you select Manual, enter the MAC address in the "MAC Address" column.
  - Own
    The client uses the MAC address of the Ethernet interface for the WLAN interface.
  - Layer 2 Tunnel
    The client uses the MAC address of the Ethernet interface for the WLAN interface. The network is also informed of the MAC addresses connected to the Ethernet interface of the client. Up to eight MAC addresses can be used.
- **MAC Address**
  If you have selected "Manual" for "MAC Mode", enter the MAC address of the client.
- **Any SSID**
  - Enabled
    In client mode the SCALANCE W700 device attempts to connect to the access point that corresponds to the security settings of security context 1. The clients can only connect to the access point on which the "Broadcast SSID" option is enabled.
  - Disabled
    The client attempts to connect to the access point from the SSID list whose security settings match one of the defined security contexts.
- **DHCP Renew After Roaming**
  - Enabled
    After changing to a different access point, a check is made to find out whether the IPv4 address of the client is still valid. If he IPv4 address is invalid, a new IPv4 address is requested from the DHCP server.
  - Disabled
    If the client changes to a different access point the IPv4 address is not checked.

- Roaming Threshold
  Specify the threshold after which the client roams to the new access point.
  - High
    Changes only at a significantly higher field strength to the AP with the stronger signal.
  - Medium
    Changes at a moderately higher field strength to the AP with the stronger signal.
  - Low
    Changes at a slightly higher field strength to the AP with the stronger signal.
- min. AP Signal Strength
  The client has signal strength preset for it. The client must receive the signal coming from the access point with at least the specified signal strength to be able to connect to this access point.
  The signal strength can fluctuate briefly, e.g. due to the client moving or other disruptive factors. To filter out these fluctuations of the signal, a hysteresis is placed around this value in which the client does not change access points before this range is unsershot. If the signal coming from the access point falls below this range, the client separates itself from the connected access point and searches for a new access point.
- Background Scan Mode
  While the client is connected to an access point, it scans for other access points in the background with which it can connect when necessary. Specify the mode for the scan. The following options are available:
  - always
    If the background scan threshold is undershot, the client searches continuously for access points.
  - idle
    If there is no data transfer for a certain time, a scan is started for further access points.
  - disabled
    As long as the client is connected, there is no scan for further access points.
- Background Scan Interval [ms]
  Specify the interval at which further access points are scanned.
- Background Scan Threshold [dBm]
  Specify the threshold. If the threshold is undershot, the client searches for further access points.

Table 2 has the following columns:

- Radio
  Shows the WLAN interface.
- Scan Channels
  Shows the channels on which the client searches for an access point. The display depends on the wireless approvals of the selected country and the settings for "Channel selection".

Table 3 has the following columns:

- Enabled
  Enables or disables the relevant SSID.
- Radio
  Shows the WLAN interface.

- **SSID**
  Enter the SSID of the access point with which the client will connect.
  For the SSID, ASCII code 0x20 to 0x7e is used.
- **Security**
  Select a security context. You create and configure a security context in "Security > WLAN > Basic".
  Default setting: Context 1

Note

iPCF or iPCF-MC mode activated

If the iPCF or iPCF-MC mode is enabled, you can only select security context 1.

Force Roaming

Note

The page is only available

- in access point mode
- if there is an online connection to the device.

The WLAN interface of the device must be enabled, otherwise the frequency ranges cannot be scanned.

If the interface is no longer available (cable break, network component faild, connector removed), a client connected over the wireless network is not aware of this. The access point can force the logged-on clients to roam by deactivating the relevant interface.

On this page you specify when roaming is performed.

- On connection abort
  If the wired Ethernet interface is no longer available the WLAN interface is turned off. The clients roam and then connect to a different access point. As soon as the first access point reaches the server again it switches its WLAN interfaces active again.
- When the destination address is unreachable.
  Roaming depends on destination addresses. To monitor the device sends pings to the configured destination addresses at regular intervals.
  - VAP interface monitored using a destination address
    If no ping response is received from this destination address, the access point turns the relevant VAP interface off.
  - VAP interface monitored using several destination addresses
    Only when no ping response is received from any of these destination addresses does the access point turn the relevant VAP interface off. As long as at least one destination address is reachable, the VAP interface remains active.

  The access point sends a disassociation frame to the WAN clients connected via this VAP interface. The WLAN clients roam and connect to a different VAP interface. If the address becomes reachable again, the connection can be established again via this VAP interface.

Settings

Table 1 has the following columns:

- **Radio**
  Shows the available WLAN interfaces.
- **Force roaming on link down**
  When enabled if there is a connection abort via the Ethernet interface, the WLAN interface is turned off.

The table "Force Roaming on IP down"has the following columns:

- **Select**
  Select the check box in the row to be deleted.
- **Destination Address**
  Enter the IPv4 address or the FQDN (Fully Qualified Domain Name) of the destination whose reachability will be checked.

  Note

  Destination address not in the agent IP subnet

  If the destination address is not in the agent IP subnet, a gateway must be entered for "Layer 2 > Agent IP".

  The Base Bridge mode "802.1Q VLAN Bridge"

  If you have configured the "Base Bridge Mode" "802.1Q VLAN Bridge" in "Layer 2 > VLAN", echo messages (ping packets) are sent into the management VLAN.

- **Interval [ms]**
  Specify the interval at which echo messages (pings) are sent.
- **Max. Lost Packets**
  Specify the maximum number of consecutive lost ping responses. When this number is reached for a destination address, this destination address counts as being unreachable (down).
- **VAP X.Y**
  Specify the VAP interface via which the address is monitored.

Signal recorder

Recording the effective user signal

The signal recorder is used to record the effective user signal between access point and client. Using this data, you can locate areas with an inadequate user signal. The signal recorder can be particularly useful when the client moves along a fixed path.

Note

The page is only available

- for clients or access points in client mode
- if there is an online connection to the device.

The WLAN interface of the device is enabled, otherwise no recording will be made.

Description

The display is divided into two areas.

- Client
  Represents the measurement of the client.
- Access point
  Displays the measurement of the access point with which the client is currently connected. This requires that the setting "Bidirectional Recording" is enabled and that a firmware version > 6.1 is installed on the access point. The access point sends its data to a maximum of 3 clients on which signal recorders are running. The access point data is not displayed on other clients.

Both areas each contain two graphics.

The first graphic contains the following elements:

- Scroll bar
  With the scroll bar, you can look through the entire measurement. To do this you can use the "<<" and ">>" buttons or the arrow keys on the keyboard.
- Bar (left)
  In the bar on the left-hand side the user signal of the client / access point is displayed in real time according to the color scheme. The gray line shows the background noise.
  If the client has an iPCF-MC connection, the user signal of the management channel is shown with a black line.
- Color scheme
  The range > -35 dBm (blue) is the overmodulation range, in other words the WLAN signal is too strong and is received overmodulated. As of approximately -60 dBm (yellow) the WLAN signal is weaker.

- x axis
  The x axis shows the course of the measurement in random samples and seconds.
- Measurement data
  - Client
    The measurement data shows the value of the effective user signal according to the color scheme shown. The gray line shows the background noise.
    If the client changes access points during a measurement (roaming) or reconnects, this is displayed by a vertical black line. On the line the new AP system name and the BSSID are shown.
    If during a measurement the client has no connection to an access point, no user signal is displayed. To make it clear that there is no connection to an access point, the BSSID is set to 00:00:00:00:00:00 and shown in red.
    If the client has an iPCF-MC connection, the user signal of the management channel is shown with an additional black line.
  - Access point
    The measurement data shows the value of the effective user signal according to the color scheme shown. The gray line shows the background noise.
    If the client changes access points during a measurement (roaming) or reconnects, this is displayed by a vertical black line.
    If the access point does not support the setting "Bidirectional Recording" no user signal is displayed.

The second graphic contains the following elements:

- Bar (left)
  In the bar on the left-hand side the transfer attempts and the data rate of the client / access point are displayed according to the color scheme.
- Color scheme
  The range > -35 dBm (blue) is the overmodulation range, in other words the WLAN signal is too strong and is received overmodulated. As of approximately -60 dBm (yellow) the WLAN signal is weaker. The individual colors are described again under the graphic.
- x axis
  The x axis shows the course of the measurement in random samples and seconds.
- Measurement data
  - Client
    The measurement data shows the transfer attempts according to the color scheme shown. The transfer attempts are shown as a bar. The data rate of the sent data packets is represented as a line. If the client changes access points during a measurement (roaming) or reconnects, this is displayed by a vertical black line.
  - Access point
    The measurement data shows the transfer attempts according to the color scheme shown. The transfer attempts are shown as a bar. The data rate of the sent data packets is represented as a line.
    If the client changes access points during a measurement (roaming) or reconnects, this is displayed by a vertical black line. If the access point does not support the setting "Bidirectional Recording" no data is displayed

Beside the graphics the following values are displayed:

- Status
  Shows whether or not the signal recorder is recording values.
- Current Sample
  The number of the current measurement
- CL RX-Signal [dBm] / AP RX-Signal [dBm]
  The effective user signal of the client / access point in dBm
- CL NF [dBm] / AP NF [dBm]
  The background noise of the client / access point in dBm
- CL Retries [%] / AP Retries [%]
  The transfer repetitions of the client / access point as a percentage.
- CL RSSI / AP RSSI
  The raw value of the RSSI (Received Signal Strength Indication) of the client / access point
- CL TX-Rate [Mbps] / AP TX-Rate [Mbps]
  The average data rate of the sent data packets during the current random test
- CL M-Signal [dBm]
  If the client has an iPCF-MC connection, the user signal of the management channel is displayed.
- Roaming Counter
  The roaming counter shows how often the client has changed access points during the recording. After 4 294 967 295 changes the counter is reset.
- Operative Channel
  The current channel or the channel on which the client is connected to the access point
- AP System Name
  The system name of the access point
- BSSID
  The BSSID (Basic Service Set Identification) of the access point.
- Connected Stations
  Number of clients connected to the access point.
- Bidirectional Status
  Shows whether the data of the access point is also being recorded.

Settings

This table contains the following columns:

- **Radio**
  Shows the WLAN interface to which the information applies. Since a client has a WLAN interface, there is only ever one row for "WLAN 1" in this table.
- **Interval [ms]**
  Specify the time interval between acquiring two measured values in milliseconds. The first measured value is displayed only after the set time interval has elapsed.
- **Samples**
  Specify how many measurements should be made.

- **Endless**
  If you enable the option check mark, the number of measurements is unlimited The "Samples" box is grayed out . The signal recorder runs until it is stopped manually or the device is reconfigured.
  You can only select this option starting at a time interval ≥ 100 milliseconds.
  If the recording contains more than 10000 measurements, the last 10000 measurements are listed in the csv file and the PDF file.
- **Bidirectional Recording**
  If you enable the setting the values of the access point are recorded as of a time interval of ≥ 10 milliseconds.
- **Start**
  Click the button in this column to start recording the wanted signal.

  Note
  - If you start a new recording, the previous recording will be overwritten.
  - If the recording has lasted less than 10 minutes and has not yet been completed (e.g. due to a restart or power down), the measured values are deleted.

  The signal recorder saves the recorded data automatically every 10 minutes. Following a restart, the recording contains all the values up to the last save action.
- **Stop**
  Click the button in this column to stop recording the wanted signal prematurely. If the specified number of measurements has been made, recording of the user data signal stops automatically.
- **Displayed Samples**
  Select how many measurements will be shown in the graphic.

Notes on usage

Note the following tips that will help you to obtain useful measurements with the signal recorder:

- Set a fixed data rate on the access point.
- If you have activated iPCF, set as low a cycle time on the access point as possible for the measurements.
- Make sure that there is enough data communication during the measurement because the statistics functions evaluate incoming data frames.
- The measurement path should be traveled 2 to 3 times with the same parameters to find out whether losses of signal strength always occur at the same position.
- Selective measurements at a fixed position should be made over a longer period of time.

Note
- If you start a new recording, the previous recording will be overwritten.
- When the device restarts, the measured values are deleted.

Change to one of the following menu items to call up the result of the recording:

- System > Load&Save > HTTP
  Click the "Save" button in the "WLANSigRec" table row to save the file "signal_recorder_SCALANCE_W700.zip" in the file system of the connected PC.
- System > Load&Save > TFTP
  If necessary, change the file name "signal_recorder_SCALANCE_W700.zip" in the "WLANSigRec" table row. In the table row "WLANSigRec", select the "Save file" entry from the drop-down list of the last column and click the "Save Values" button.

The downloaded ZIP file contains two files with the results of the recording:

- A PDF file
- A CSV file

Spectrum Analyzer

Note

The page is only available

- in access point mode
- if there is an online connection to the device.

The WLAN interface of the device must be enabled, otherwise the frequency ranges cannot be scanned.

Technical information

The frequency range depends on the configuration.

Parameters		Value
Amplitude accuracy	In 2.4 GHz	3 dBm
	In 5 GHz	7 dBm
Resolution bandwidth		330 KHz
Min. signal strength		-100 dBm
Max. signal strength		0 dBm
Analysis time	At 40 MHz	120 ms
	At 20 MHz	95 ms
Update time		1 s.

Representing signals of the frequency range

Which the spectrum analyzer you can recognize and represent the electromagnetic signals of a frequency range. You can measure the strength of all signals located in the environment of the access point.

Note

When the spectrum analyzer is started, all WLAN connections are terminated on both WLAN interfaces. The access point then also does not send any beacons.

Note

Do not enable the spectrum analyzer if the device is operating productively. This can influence the performance of the device.

Note

The functionality of the spectrum analyzer does not replace a dedicated spectrum analyzer.

Description

The page contains the following graphics:

In all graphics, the lower x axis shows the channels around the selected center frequency for which the measurements are made. The upper x axis shows the frequency range. The display of the y axis depends on the selected graphic.

- Realtime
  The y axis shows the signal strength in dBm.
  The graphic shows the strength of all signals that the access point receives in its environment in the configured frequency range.
  The red line shows the maximum values since the start of the measurement. The white line shows the current values. The green line shows the average values.
- Spectrogram
  The y axis shows the course of the measured values over time from current (0 s) to the values received before 500 s.
  The graphic shows the strength of all signals that the access point receives in its environment in the configured frequency range.
  The color depends on the setting for "Color Scheme".
- Density Chart
  The y axis shows the signal strength in dBm.
  The graphic shows how often signals occur with a certain strength in the configured frequency range.
  The color goes from the lowest value (0%) in black to the highest value (100%) in red.

The page contains the following buttons:

- Expand
  With this icon you only show one graphic type in large format on the page.
- Reduce
  With this icon you return to the view with all three graphic types.

- Color Scheme
  With this icon, you change the colours scheme for the graphic type "Spectrogram":
  - The color goes from the lowest value (-100 dBm) in black to the highest value (0 dBm) in red.
  - The color goes from the lowest value (-100 dBm) in red to the highest value (0 dBm) in black.
- Reset
  With this icon you reset the maximum and average values of the graphic type "Realtime".

This table contains the following columns:

- Radio
  Shows the WLAN interface to which the information applies.
- State
  Shows the status of the measurement. The following values are possible:
  - Stopped
    The measurement was stopped.
  - Started
    The measurement is running.
- Frequency Band
  Specify the frequency band.
- Center Frequency
  Select the center frequency.
  How many channels are displayed depends on the setting for the "HT Channel Width", refer to the page "Interfaces > WLAN > AP".
- Start
  Click the button in this column to start the measurement.

  Note

  If you start a new measurement, the previous measurement will be overwritten.

- Stop
  Click the button in this column to end the measurement.

You can save the measurement result as a CSV file with the menu “System > Load&Save > WLANSpectrumAnalyzer”.

Measurement results

CSV file

The CSV file contains information on the configuration of the device and detailed information on all individual measurements and is divided into two areas. The first area contains the configured settings:

- System Name
  The system name of the access point
- Device IP
  The IP address of the device

- Device MAC
  The MAC address of the device
- Recording Interval
  The interval between acquisition of two measured values

The second area is a table. The table contains the following for each measured value:

- Sample
  The consecutive number of the measurement
- Timestamp
  The time stamp
- The following columns show all frequencies of the selected frequency band. The cells are only filled for the frequencies for which a value was measured. The measured values show the signal strength in dBm.

Remote Capture

On these pages activate the function "Remote Capture" on the interface (Ethernet, WLAN). The function is for network diagnostics via a connected PC, e.g. to detect transfer errors.

You can also enable the function on several interfaces at the same time. When the function is enabled the interface can be linked in Wireshark. For a period Wireshark record the data traffic over the interface. Afterwards from the recording you can see the content of the frames or filter according to certain contents.

Description

This table contains the following columns:

- Interface
  The interface to which the entry relates.
- Enable
  Enable or disable the "Remote Capture" function. As default, the function is disabled.

  Note

  Performance

  Enable the function only for diagnostics purposes. The increased data traffic could influence the performance of the device.

The page contains the following boxes:

- **WLAN Capture Mode (only in access point mode)**
  Specify the recording mode for the WLAN interface:
  - Own Traffic
    In this case, the frames are recorded that were received and sent by the device. Exception: The data packets dealt with directly by the hardware are not displayed, for example hardware repetitions, acknowledgment frames.
  - All Traffic
    The access point sends no more frames but records all incoming data packets.

  Note

  No WLAN communication between access point and clients

  If the setting "All Traffic" is used, the access point is no longer reachable for other nodes and loses the connected clients.

- **Activate after System Restart**
  - Disabled
    After a restart the configuration is rest to the default settings.
  - Enabled
    The configuration is saved and retained after a restart.

Linking in the interface in Wireshark

Requirement:

- Wireshark V2.0.0 is installed on the PC.
- The PC and device must be reachable via IP (layer 3).

Procedure

To analyze the data traffic e.g. of the WLAN interface 1 in Wireshark, follow the steps below:

1. Activate the function "Remote Capture" on the device on the WLAN interface.
2. As the receive mode, select "Own Traffic".
3. Click "Set Values" to enable the function.
4. Start Wireshark.
5. Click "Options" in the "Capture" menu. The window "Wireshark - Capture Interfaces" opens.
6. Click the "Manage Interfaces..." button on the "Input" tab. In the following dialog, click on the "Remote Interfaces" tab.
7. To add the interface click on the Plus character in the "Remote Interfaces" tab.
8. In the following dialog for "Host" enter the IPv4 address of the device and for "Port" 2002.
9. Enable "Null authentication" for "Authentication" and click the "OK" button.
10. On the "Remote Interfaces" tab, the host and the interfaces on which the function "Remote Capture" was previously enabled are displayed.

11. Select the interface and click the "OK" button.
12. To start the recording click "Start". You can obtain further information about handling the program in Wireshark.

If you analyze several interfaces you can use a Wireshark instance for each interface.

Configuring layer 2 functions

VLAN

General

On this page you specify whether or not the device forwards frames with VLAN tags transparently (IEEE 802.1D/VLAN-unaware mode) or takes VLAN information into account (IEEE 802.1Q/VLAN-aware mode). If the device is in the "802.1Q VLAN Bridge" mode, you can define VLANs and specify the use of the ports .

Note

Changing the agent VLAN ID

If the configuration PC is connected directly to the device via Ethernet and you change the agent VLAN ID, the device is no longer reachable via Ethernet following the change.

Description

The page contains the following boxes:

- **Base Bridge-Mode**
  Select the required mode. The following modes are possible:

  Note
  Changing Base Bridge Mode

  Note the section "Changing Base Bridge Mode". This section describes how a change affects the existing configuration.

  – 802.1 D Transparent Bridge
    Sets the mode "VLAN-unaware" for the device. In this mode, VLAN tags are not taken into account or changed but are forwarded transparently. In this mode, you cannot create any VLANs.
  – 802.1Q VLAN Bridge
    Sets the mode "VLAN-unaware" for the device. In this mode, VLAN tags are not taken into account or changed but are forwarded transparently. In this mode, you cannot create any VLANs.
    Only a management VLAN is available: VLAN 1.

- **VLAN ID**
  Enter the VLAN-ID (a number between 1 and 4094).
  You can create up to 2 (SCALANCE W760), 8 (SCALANCE W770) or 24 (SCALANCE W780) VLANS.

The table has the following columns:

- **VLAN ID**
  Shows the VLAN ID. The VLAN ID (a number between 1 and 4094) can only be assigned once when creating a new data record and can then no longer be changed. To make a change, the entire data record must be deleted and created again.
- **Name**
  Enter a name for the VLAN. The name only provides information and has no effect on the configuration.

- **Status**
  Shows the status type of the entry in the internal port filter table. Here, static means that the address was entered as a static address by the user. The entry GVRP means that the configuration was registered by a GVRP frame. This is, however, only possible if GVRP was enabled for the device.
- **List of ports**
  Specify the use of the port. The following options are available:
  - "-"
  - The port is not a member of the specified VLAN.
    With a new definition, all ports have the identifier "-".
  - M
    The port is a member of the VLAN. Frames sent in this VLAN are forwarded with the corresponding VLAN tag.
  - R
    The port is a member of the VLAN. A GVRP frame is used for the registration.
  - U (upper case)
    The port is an untagged member of the VLAN. Frames sent in this VLAN are forwarded without the VLAN tag. Frames without a VLAN tag are sent from this port.
  - u (lower case)
    The port is an untagged member of the VLAN, but the VLAN is not configured as a port VLAN. Frames sent in this VLAN are forwarded without the VLAN tag.
  - F
    The port is not a member of the specified VLAN and it is not possible for the VLAN to be registered dynamically at this port using GVRP. You can configure other settings in "Layer 2 > VLAN > Port-based VLAN".
  - T
    This option is only displayed and cannot be selected. This port is a trunk port, making it a member in all VLANs. You configure this function in the CLI (Command Line Interface) using the "`switchport mode trunk`" command or in the WBM under "Interfaces > Ethernet > Configuration".

Changing Base Bridge Mode

VLAN-unaware (802.1D transparent bridge) → VLAN-aware (802.1Q VLAN bridge)

If you change the Base Bridge Mode from VLAN-unaware to VLAN aware, this has the following effects:

- All static and dynamic unicast entries are deleted.

VLAN-aware (802.1Q VLAN bridge) → VLAN-unaware (802.1D transparent bridge)

If you change the Base Bridge Mode from VLAN-aware to VLAN-unaware, this has the following effects:

- All VLAN configurations are deleted.
- A management VLAN is created: VLAN 1.
- All static and dynamic unicast entries are deleted.

802.1Q VLAN Bridge: Important rules for VLANs

Make sure you keep to the following rules when configuring and operating your VLANs:

- Frames with the VLAN ID "0" are handled as untagged frames but retain their priority value.
- As default, all ports on the device send frames without a VLAN tag to ensure that the end node can receive these frames.
- You can find the factory assignment of the ports in the section "VLAN".
- If an end node is connected to a port, outgoing frames should be sent without a tag (static access port). If, however, there is a further switch at this port, the frame should have a tag added (trunk port).

Port-based VLAN

Processing received frames

On this page, you specify the configuration of the port properties for receiving frames.

Description of the displayed boxes

Table 1 has the following columns:

- Port
  Shows that the settings are valid for all ports.
- Priority / Port VID / Acceptable Frames / Ingress Filtering
  Select the setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Transfer to table
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- Port
  Shows the available ports and link aggregations. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- Priority
  Select the priority assigned to untagged frames.
  The CoS priority (Class of Service) used in the VLAN tag. If a frame is received without a tag, it will be assigned this priority. This priority specifies how the frame is further processed compared with other frames. There are a total of eight priorities with values 0 to 7, where 7 represents the highest priority (IEEE 802.1p Port Priority).
- Port VID
  Select the VLAN ID. Only VLAN IDs defined on the "VLAN > General" page can be selected. If a received frame does not have a VLAN tag, it has a tag with the VLAN ID specified here added to it and is sent according to the rules at the port.

- Acceptable Frames
  Specify which types of frames will be accepted. The following alternatives are possible:
  - Tagged Frames Only
    The device discards all untagged frames. Frames tagged with "0" are treated like untagged frames. The device forwards all tagged frames. Otherwise, the forwarding rules apply according to the configuration.
  - Untagged and Priority Tagged Only
    The device discards all tagged frames. The device forwards all untagged frames and frames with a priority (Priority Tagged Frames). Otherwise, the forwarding rules apply according to the configuration. If you have configured the Bridge mode "Provider" this means that the device treats all incoming frames as untagged frames.
  - All
    The device forwards all frames
- Ingress Filtering
  Specify whether the VID of received frames is evaluated. The following options are available:
  - Enabled
    The VLAN ID of received frames decides whether they are forwarded: To forward a VLAN tagged frame, the receiving port must be a member in the same VLAN. Frames from unknown VLANs are discarded at the receiving port.
  - Disabled
    All frames are forwarded.

Dynamic MAC aging

Protocol settings and switch functionality

The device automatically learns the source addresses of the connected nodes. This information is used to forward data frames to the nodes specifically involved. The network load for the other nodes is reduced.
If a device does not receive a frame whose source address matches a learnt address within a certain time, it deletes the learnt address. This mechanism is known as aging. Aging prevents frames being forwarded incorrectly, for example when an end device (for example a programming device) is connected to a different port. If the check box is not enabled, a device does not delete learned addresses automatically.

Description of the displayed boxes

The page contains the following boxes:

- Dynamic MAC Aging
  Enable or disable the function for automatic aging of learned MAC addresses:
- Aging Time [s]
  Enter the time in seconds. After this time, a learned address is deleted if the device does not receive any further frames from this sender address. Range of values: 10 - 630 seconds

Spanning Tree

General

General settings of MSTP

This is the basic page for spanning tree. Select the compatibility mode from the drop-down list. As default, Multiple Spanning Tree is enabled.

On the configuration pages of these functions, you can make detailed settings.

Note

Client device not as root bridge

Using the configuration of priorities and path costs, make sure that a client device can never become the root bridge. If a client device becomes the root bridge the Rapid Spanning Tree function no longer works.

Depending on the compatibility mode, you can configure the corresponding function on the relevant configuration page.

Settings

- **Spanning Tree**
  Enable or disable Spanning Tree.

  Note

  Clients or access points in client mode

  If "Automatic" is set for "MAC Mode", spanning tree is not supported.

- **Protocol Compatibility**
  Select the compatibility mode of Spanning Tree. If, for example, you select RSTP, Spanning Tree behaves like RSTP.
  The following settings are available:
  - STP
  - RSTP
  - MSTP

  Note

  If iPCF mode is enabled, only the compatibility modes STP and RSTP are supported.

CIST general

The page consists of the following parts:

- The left-hand side of the page shows the configuration of the device.
- The central part shows the configuration of the root bridge that can be derived from the spanning tree frames received by a device.
- The right-hand side shows the configuration of the regional root bridge that can be derived from the MSTP frames received by a device. The displayed data is only visible if you have enabled "Spanning Tree" on the "General" page and if "MSTP" is set for "Protocol Compatibility". This also applies to the "Bridge Max Hop Count" parameter. If the device is a root bridge, the information on the left and right matches.

Settings

- **Bridge Priority / Root Priority** (only available online)
  Which device becomes the root bridge is decided based on the bridge priority. The bridge with the highest priority becomes the root bridge. The lower the value, the higher the priority. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames.
- **Bridge Address** / **Root Address**(only available online)
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Port** (only available online)
  Shows the port via which the switch communicates with the root bridge.
- **Root Cost** (only available online)
  The path costs from this device to the root bridge.
- **Topology Changes** / **Last Topology Change** (only available online)
  The entry for the device shows the number of reconfigurations due to the spanning tree mechanism since the last startup. For the root bridge, the time since the last reconfiguration is displayed as follows:
  - Seconds: "Sec" unit after the number
  - Minutes: "Min" unit after the number
  - Hour: "Hr" unit after the number
- **Bridge Hello Time / Root Hello Time** (only available online)
  Each bridge regularly sends configuration frames (BPDUs). The interval between two such frames is the Hello time. The default for this parameter is 2 seconds.
- **Bridge Forward Delay [s] / Root Forward Delay [s]**
  New configuration data is not used immediately by a bridge but only after the period specified in the parameter. This ensures that operation is only started with the new topology after all the bridges have the required information.
- **Bridge Max Age / Root Max Age** (only available online)
  Bridge Max Age defines the maximum "age" of a received BPDU for it to be accepted as valid by the switch.

The following values are available only in access point mode:

- **Regional root priority** (only available online)
  For a description, see Bridge Priority / Root Priority
- **Regional root address** (only available online)
  The MAC address of the device.
- **Regional root costs** (only available online)
  The path costs from this device to the root bridge.
- **Bridge Max Hop Count**
  This parameter specifies how many MSTP nodes a BPDU may pass through. If an MSTP BPDU is received and has a "Bridge Max Hop Count" that exceeds the value configured here, it is discarded.
- **Region Name**
  Enter the name of the MSTP region to which this device belongs. As default, the MAC address of the device is entered here. This value must be the same on all devices that belong to the same MSTP region.
- **Region Version**
  Enter the version number of the MSTP region in which the device is located. This value must be the same on all devices that belong to the same MSTP region.
- **Layer-2 Tunnel Admin Edge Port** (only available in access point mode)
  Enable this setting if there can be an end device on a layer 2 tunnel port. Otherwise a reconfiguration of the network will be triggered whenever a link to this port is modified. The L2T clients should be interconnected.
- **Layer-2 Tunnel Auto Edge Port**(Only available in access point mode)
  Enable this setting if you want to detect automatically at all layer 2 tunnel ports whether or not an end device is connected.

To reset the counters on this page the following button is available:

- **Reset Counters** (only available online)
  The values are reset with this button.

CIST port

MSTP-CIST port configuration

When the page is called, the table displays the current status of the configuration of the port parameters.

To configure them, click the relevant cells in the port table.

Description of the displayed boxes

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Spanning Tree Status**
  Select the setting. The following settings are possible:
  - Enabled
    Port is integrated in the spanning tree.
  - Disabled
    Port is not integrated in the spanning tree.
  - No change
    Table 2 remains unchanged.
- Transfer to table
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows all available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Spanning Tree Status**
  Specify whether or not the port is integrated in the spanning tree.

  Note

  If you disable the "Spanning Tree Status" setting for a port, this may cause the formation of loops. The topology must be kept in mind.

- **Priority**
  Enter the priority of the port. The priority is only evaluated if the path costs are the same. The value must be divisible by 16. If the value that cannot be divided by 16, the value is automatically adapted.
  Range of values: 0 - 240. The default is 128.
- **Cost Calc.**
  Enter the path cost calculation. If you enter the value "0" here, the automatically calculated value is displayed in the "Path Costs" box.

- **Path Costs** (only available online)
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected as the path. If several ports of a device have the same value for the path costs, the port with the lowest port number is selected.
  If The "Cost Calc." is "0", the automatically calculated value is shown. Otherwise, the value of the "Cost Calc." box is displayed. The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000

  The values can, however, also be set individually.
- **Status** (only available online)
  Displays the current status of the port. The values are only displayed and cannot be configured. The "Status" parameter depends on the configured protocol. The following is possible for status:
  - Disabled

    The port only receives and is not involved in STP, MSTP and RSTP.
  - Discarding
    In the "Discarding" mode, BPDU frames are received. Other incoming or outgoing frames are discarded.
  - Listening

    In this status, BPDUs are both received and sent. The port is involved in the spanning tree algorithm.
  - Learning

    Stage prior to the forwarding status, the port is actively learning the topology (in other words, the node addresses).
  - Forwarding

    Following the reconfiguration time, the port is active in the network; it receives and forwards data frames.

- **Fwd. Trans** (only available online)
  Specifies the number of changes from the "Discarding" status to the "Forwarding" status
- **Edge type**
  Specify the type of edge port. You have the following options:
  - "-"
    Edge port is disabled. The port is treated as a "no edge port".
  - Admin
    Select this option when there is always an end device on this port. Otherwise a reconfiguration of the network will be triggered each time a connection is changed.
  - Auto
    Select this option if you want a connected end device to be detected automatically at this port. When the connection is established the first time, the port is treated as a "no Edge Port".
  - Admin/Auto
    Select these options if you operate a combination of both on this port. When the connection is established the first time, the port is treated as an Edge Port.
- **Edge** (only available online)
  Shows the status of the port.
  - Enabled

    An edge port is connected to this port.
  - Disabled
    There is a spanning tree or rapid spanning tree device at this port.

  With an end device, a switch can change over the port faster without taking into account spanning tree frames. If a spanning tree frame is received despite this setting, the port automatically changes to the "Disabled" setting for switches.
- **P.t.P. Type**
  Select the required option. The selection depends on the port that is set.
  - "-"
    Point to point is calculated automatically. If the port is set to half duplex, a point-to-point link is not assumed.
  - P.t.P.

    Even with half duplex, a point-to-point link is assumed.
  - Shared Media

    Even with a full duplex connection, a point-to-point link is not assumed.

    Note

    Point-to-point connection means a direct connection between two devices. A shared media connection is, for example, a connection to a hub.

- **Hello Time**
  Enter the interval after which the bridge sends configuration frames (BPDUs). As default, 2 seconds is set. Range of values: 1-2 seconds

  Note

  The port-specific setting of the Hello time is only possible with protocol compatibility MSTP. If the protocol compatibility RSTP is set, the "Bridge Hello Time" parameter on the page "Layer 2 > Spanning Tree > CIST General" page is used.

- **Restr. Role**
  If this check box is selected, the corresponding port is not selected as root port, regardless of the priority value. If the check box is selected, the port with lowest priority also does not become the root port. Only activate this option if you wish to restrict the impact of bridges outside of the administered range on the Spanning Tree topology.
- **Limited TCN**
  If this check box is selected, the corresponding port does not forward either received or detected topology changes (Topology Change Notification) to other ports. Only activate this option if you wish to restrict the impact of bridges outside of the administered range on the Spanning Tree topology.
- **Limited TCN**
  If this check box is selected, the corresponding port accepts received and detected topology changes, but does not forward topology changes to other ports. You can only select the check box in this column if the following conditions are met:
  - RSTP+ must be activated.
  - The "Limited TCN" check box must be cleared for this port.

  If the specified conditions are not met, the check box in this column is shown grayed out.

MST General

Multiple Spanning Tree configuration

With MSTP, in addition to RSTP, several VLANs can be managed in a LAN with separate RSTP trees.

Description of the displayed boxes

- **MSTP Instance ID**
  Enter the number of the MSTP instance. Permitted values: 1 - 64

The table has the following columns:

- **Select**
  Select the row you want to delete.
- **MSTP Instance ID**
  Shows the number of the MSTP instance.
- **Root Address**
  Shows the MAC address of the root bridge

- **Root Priority**
  Shows the priority of the root bridge.
- **Bridge Priority**
  Enter the bridge priority in this box. The value for the bridge priority is a whole multiple of 4096 with a range of values from 0 to 61440.
- **VLAN ID**
  Enter the VLAN ID. Here, you can also specify ranges with Start ID, "-", End ID. Several ranges or IDs are separated by ",". Permitted values: 1- 4094

MST port

Configuration of the Multiple Spanning Tree port parameters

On this page, you set the parameters for the ports of the configured multiple spanning tree instances.

Description of the displayed boxes

The page contains the following box:

- **MSTP Instance ID**
  Select the ID of the MSTP instance.

Table 1 has the following columns

- **1st column**
  Shows that the setting is valid for all ports.
- **MSTP Status**
  Select the setting. The following settings are possible:
  - Enabled
  - Disabled
  - No Change: Table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2

Table 2 has the following columns:

- **Port**
  Shows all available ports and link aggregations.
- **MSTP Instance ID**
  ID of the MSTP instance.
- **MSTP Status**
  Click the check box to enable or disable this option.
- **Priority**
  Enter the priority of the port. The priority is only evaluated if the path costs are the same. The value must be divisible by 16. If the value that cannot be divided by 16, the value is automatically adapted.
  Range of values: 0 - 240. Factory setting: 128

- **Cost Calc.**
  Enter the path cost calculation in the input box. If you enter the value "0" here, the automatically calculated value is displayed in the next box "Path Costs".
- **Path Costs** (only available online)
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected as the path. If several ports of a device have the same value for the path costs, the port with the lowest port number is selected.
  If the "Cost Calc." box has the value "0", the automatically calculated value is shown. Otherwise, the value of the "Cost Calc." box is displayed. The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000

  The values can, however, also be set individually.
- **Status** (only available online)
  Displays the current status of the port. The values are only displayed and cannot be configured. The following is possible for status:
  - Discarding
    The port exchanges MSTP information but is not involved in the data traffic.
  - Blocked
    In the blocking mode, BPDU frames are received.
  - Forwarding
    The port receives and sends data frames.
- **Fwd. Trans**(only available online)
  Specifies the number of status changes Discarding - Forwarding or Forwarding - Discarding for a port.

DCP forwarding

Applications

The DCP protocol is used by STEP 7 and the PST Tool for configuration and diagnostics. When shipped, DCP is enabled on all ports; in other words, DCP frames are forwarded at all ports. With this option, you can disable the sending of these frames for individual ports, for example to prevent individual parts of the network from being configured with the PST Tool or to divide the full network into smaller parts for configuration and diagnostics.

Note

Empty table

If you have enabled NAT on the device, the table is empty or will be emptied:

Settings

- **Port**
  Shows the available ports.
- **Setting**
  Specify whether the port should block or forward outgoing DCP frames. The following options are available:
  - block
    No outgoing DCP frames are forwarded via this port. It is nevertheless still possible to receive via this port.
  - forward
    DCP frames are forwarded via this port.

LLDP

Identifying the network topology

LLDP (Link Layer Discovery Protocol) is defined in the IEEE 802.AB standard.

LLDP is a method used to discover the network topology. Network components exchange information with their neighbor devices using LLDP.

Network components that support LLDP have an LLDP agent. The LLDP agent sends information about itself and receives information from connected devices at periodic intervals. The received information is stored in the MIB.

Applications

PROFINET uses the LLDP protocol for topology diagnostics. In the default setting, LLDP is enabled for all ports; in other words, LLDP frames are sent and received on all ports. With this function, you have the option of enabling or disabling sending and/or receiving per port.

Settings

The table has the following columns:

- **Port**
  Shows the port.
- **Setting**
  Specify the LLDP functionality. The following options are available:
  - Tx
    This port can only send LLDP frames.
  - Rx
    This port can only receive LLDP frames.
  - Rx & Tx
    This port can receive and send LLDP frames.
  - "-" (disabled)
    This port can neither receive nor send LLDP frames.

Configuring layer 3 functions for IPv4

NAT

Basic

Note

The page is only available

- SCALANCE W740, SCALANCE W730 and SCALANCE W722-1 RJ45
- clients or access points in client mode.

On this page, you specify the basic settings for NAT.

Settings

- **Interface**
  Select the required interface.
- **Enable NAT**
  Enable or disable NAT for the Ethernet interface. Cannot be enabled if Layer 2 Tunnel is set for MAC Mode on the client.
- **TCP Idle Timeout [s]**
  Enter the required time in seconds. If no data exchange takes place, the TCP connection is deleted from the translation table when this time has elapsed.
  The range of values is 1 to 4294967295. Default setting: 86400 seconds
- **UDP Idle Timeout [s]**
  Enter the required time in seconds. If no data exchange takes place, the UDP connection is deleted from the translation table when this time has elapsed.
  The range of values is 1 to 4294967295. Default setting: 300 seconds
- **Local Interface IP address**
  Enter the local IP address of the Ethernet interface. This IP address is the gateway address of the local device.
- **Local Interface Subnet Mask**
  Enter the subnet mask for the local Ethernet.

- **IPv6 Transparent Mode**
  When enabled, the transparent mode is activated. IPv6 frames are forwarded unchanged between Ethernet and WLAN. This assumes that "Own" is not set for MAC mode and IPv6 is turned off. If you have set "Manual" for the MAC mode, you need to enter the MAC address of the IPv6 device that receives or sends the IPv6 frames.
- **IPv4 Multicast Forwarding**
  Specify whether or not the incoming multicast frames will be forwarded.
  – From Global to Local Interface
    The multicast frames incoming on the WLAN interface are forwarded via the Ethernet interface into the internal network.
  – From Local to Global Interface
    The multicast frames incoming on the local Ethernet interface are forwarded via the WLAN interface into the external network.

NAPT

Note

The page is only available
- SCALANCE W740, SCALANCE W730 and SCALANCE W722-1 RJ45
- clients or access points in client mode.

On this page, you define the translation list for communication from the global to the local network.

Settings

- **Interface**
  Interface to which the settings relate. Can only be selected if the device has several interfaces.
- **Traffic Type**
  Specify the protocol for which the address assignment is valid.
- **Global Port**
  Enter the global port. Incoming frames with this port as the destination port are forwarded.

  Note

  If the port is already occupied by a local service, for example Telnet, a warning is displayed. In this case, avoid using TCP port 23 (Telnet), port 22 (SSH), ports 80/443 (http/https: reachability of the client with the WBM) and UDP port 161 (SNMP) as global port.
- **Local IP address**
  Enter the IP address of the node in the local network. The frame is forwarded to this IP address.
- **Local Port**
  Enter the number of the port. This is the new destination port to which the incoming frame will be forwarded. You can enter the same port in the local area network for more than one node.

The table has the following columns:

- Select
  Select the lines to be deleted.
- Interface
  Shows the interface to which the settings relate.
- Activate
  When enabled, the entry for the address assignment is used.
- Traffic Type
  Shows the protocol for which the address assignment applies.
- Dynamic Global IP Address
  Shows whether or not dynamic address translation is used.
- Global IP Address
  Shows the global IP address to which the local IP address will be translated.
- Global Port
  Shows the global port.
- Local IP address
  Shows the IP address of the node in the local network.
- Local Port
  Shows the number of the local port.

Configuring security functions

Passwords

Passwords

Configuration of the device passwords

Note

The page is only available if there is an online connection to the device.

Note

If you are logged in via a RADIUS server, you cannot change any local device passwords.

On this page, you can change passwords. If you are logged on with the "admin" role, you can change the passwords for all user accounts. If you are logged on with the "user" role, you can only change your own password.

Note

When you log in for the first time or following a "Restore Factory Defaults and Restart", with the preset user "admin" you will be prompted to change the password.

The factory settings for the passwords when the devices ship are as follows:

- admin: admin

Note

Changing the password in Trial mode

Even if you change the password in "Trial" configuration mode, this change is saved immediately.

Description of the displayed boxes

- **Current User**
  Shows the user that is currently logged in.
- **Current User Password**
  Enter the password for the currently logged in user.
- **User Account**
  Select the user whose password you want to change.
- **Password Policy**
  Shows which password policy is being used when assigning new passwords.
  - High
    Password length: at least 8 characters, maximum 128 characters
    at least 1 uppercase letter
    at least 1 special character
    at least 1 number
  - Low
    Password length: at least 6 characters, maximum 128 characters
- **New name for the Admin user account**
  You can change the name the default user "admin" once. Afterwards, renaming "admin" is no longer possible.
- **New Password**
  Enter the new password for the selected user. The following characters must not be included:
  – § ? " ; : –
  The space and the character for "remove" must also not be included.
- **Password Confirmation**
  Enter the new password again to confirm it.

Options

On this page you specify which password policy will be used when assigning new passwords.

Description

- **Password Policy**
  Shows which password policy is currently being used
- **New Password Policy**
  Select the required setting from the drop-down list.
  - High
    Password length: at least 8 characters, maximum 128 characters
    at least 1 uppercase letter
    at least 1 special character
    at least 1 number
  - Low
    Password length: at least 6 characters, maximum 128 characters

Users

Local users

Note

The page is only available if there is an online connection to the device.

Local users

On this page, you create local users with the corresponding rights.

When you create or delete a local user this change is also made automatically in the table "External User Accounts". If you want to make change explicitly for the internal or external user table, use the CLI commands.

Note

The values displayed depend on the rights of the logged-in user.

Description of the displayed boxes

- User Account
  Enter the name for the user. The name must meet the following conditions:
  - It must be unique.
  - It must be between 1 and 255 characters long.
  - It must not contain the following characters: | ? " ; :
    The characters for Space and Delete must also not be included.

  Note

  User name cannot be changed

  After creating a user, the user name can no longer be modified.

  If a user name needs to be changed, the user must be deleted and a new user created.

  Note

  Default user "user" set in the factory

  As of firmware version 6.0 the default user set in the factory "user" is no longer available when the product ships. If you update a device to the firmware V6.0 the default user set in the factory "user" is initially still available. If you reset the device to the factory settings ("Restore Factory Defaults and Restart") the default user set in the factory "user" is deleted. You can create new users with the role "user".

- Password Policy
  Shows which password policy is being used.
  - High
    Password length: at least 8 characters, maximum 128 characters
    at least 1 upper case letter
    at least 1 special character
    at least 1 number
  - Low
    Password length: at least 6 characters, maximum 128 characters

  You configure the password policy on the page "Security > Passwords > Options".
- Password
  Enter the password for the user. The strength of the password depends on the set password policy.
- Password Confirmation
  Enter the password again to confirm it.
- Role
  Select a role.
  You can choose between system-defined and self-defined roles, refer to the page "Security > Users > Roles.".

The table contains the following columns:

Note

The preset users as well as logged in users cannot be deleted or changed.

- **User Account**
  Shows the user name.
- **Role**
  Shows the role of the user.
- **Description**
  Displays a description of the user account. The description text can be up to 100 characters long.

Roles

Note

The page is only available if there is an online connection to the device.

Roles

On this page, you create roles that are valid locally on the device.

Note

The values displayed depend on the rights of the logged-in user.

Settings

- **Role Name**
  Enter the name for the role. The name must meet the following conditions:
  – It must be unique.
  – It must be between 1 and 64 characters long.

  Note

  Role name cannot be changed

  After creating a role, the name of the role can no longer be changed.

  If a name of a role needs to be changed, the role must be deleted and a new role created.

  Deleting roles

  Predefined roles and assigned roles cannot be deleted or modified.

The table contains the following columns:

- Role
  Shows the name of the role.
- Function Right
  Select the function rights of the role.
  - 1
    Users with this role can read device parameters but cannot change them.
  - 15
    Users with this role can both read and change device parameters.

  Note

  Function right cannot be changed

  If you have assigned a role, you can no longer change the function right of the role.

  If you want to change the function right of a role, follow the steps outlined below:

  1. Delete all assigned users.
  2. Change the function right of the role:
  3. Assign the role again.

- Description
  Enter a description for the role. With predefined roles a description is displayed. The description text can be up to 100 characters long.

Groups

Note

The page is only available if there is an online connection to the device.

User Groups

On this page you link a group with a role.

In this example the group "Administrators" is linked to the "admin"role: The group is defined on a RADIUS server. The role is defined locally on the device. When a RADIUS server authenticates a user and assigns the user to the "Administrators" group, this user is given rights of the "admin" role.

Settings

- Group Name
  Enter the name of the group. The name must match the group on the RADIUS server. The name must meet the following conditions:
  - It must be unique.
  - It must be between 1 and 64 characters long.

The table contains the following columns:

- Group
  Shows the name of the group.
- Role
  Select a role. Users who are authenticated with the linked group on the RADIUS server receive the rights of this role locally on the device.
  You can choose between system-defined and self-defined roles, refer to the page "Security > Users > Roles.".
- Description
  Enter a description for the link of the group.to a role.

AAA

User management

Overview of user management

Access to the device is managed by configurable user settings. Set up users with a password for authentication. Assign a role with suitable rights to the users.

The authentication of users can either be performed locally by the device or by an external RADIUS server. You configure how the authentication is handled on the "Security > AAA > General" page.

Local logon

The local logging on of users by the device runs as follows:

1. The user logs on with user name and password on the device.
2. The device checks whether an entry exists for the user.
   → If an entry exists, the user is logged in with the rights of the associated role.
   → If no corresponding entry exists, the user is denied access.

Login via an external RADIUS server

RADIUS (Remote Authentication Dial-In User Service) is a protocol for authenticating and authorizing users by servers on which user data can be stored centrally.

Depending on the RADIUS authorization mode you have selected on the "Security > AAA > RADIUS Client" page, the device evaluates different information of the RADIUS server.

RADIUS authorization mode "Standard"

If you have set the authorization mode "conventional", the authentication of users via a RADIUS server runs as follows:

1. The user logs on with user name and password on the device.
2. The device sends an authentication request with the login data to the RADIUS server.
3. The RADIUS server runs a check and signals the result back to the device.
   - The RADIUS server reports a successful authentication and returns the value "Administrative User" to the device for the attribute "Service Type".
     → The user is logged in with administrator rights.
   - The RADIUS server reports a successful authentication and returns a different or even no value to the device for the attribute "Service Type".
     → The user is logged in with read rights.
   - The RADIUS server reports a failed authentication to the device:
     → The user is denied access.

RADIUS authorization mode "SiemensVSA"

Requirement

For the RADIUS authorization mode "Siemens VSA" the following needs to be set on the RADIUS server:

- Manufacturer code: 4196
- Attribute number: 1
- Attribute format: Character string (group name)

Procedure

If you have set the authorization mode "SiemensVSA", the authentication of users via a RADIUS server runs as follows:

1. The user logs on with user name and password on the device.
2. The device sends an authentication request with the login data to the RADIUS server.
3. The RADIUS server runs a check and signals the result back to the device.
   Case A: The RADIUS server reports a successful authentication and returns the group assigned to the user to the device.
   - The group is known on the device and the user is not entered in the table "External User Accounts"
     → The user is logged in with the rights of the assigned group.
   - The group is known on the device and the user is entered in the table "External User Accounts"
     → The user is assigned the role with the higher rights and logged in with these rights.
   - The group is not known on the device and the user is entered in the table "External User Accounts"
     → The user is logged in with the rights of the role linked to the user account.
   - The group is not known on the device and the user is not entered in the table "External User Accounts"
     → The user is logged in with the rights of the role "Default".

   Case B: The RADIUS server reports a successful authentication but does not return a group to the device.
   - The user is entered in the table "External User Accounts":
     → The user is logged in with the rights of the linked role "".
   - The user is not entered in the table "External User Accounts":
     → The user is logged in with the rights of the role "Default".

   Case C: The RADIUS server reports a failed authentication to the device:
   - The user is denied access.

General

Login of network nodes

The designation "AAA" used in the menu stands for "Authentication, Authorization, Accounting". This feature is used to identify and allow network nodes and to make the corresponding services available to them.

On this page, you configure the login.

Description of the displayed boxes

The page contains the following boxes:

Note

To be able to use the login authentication "RADIUS", "Local and RADIUS" or "RADIUS and fallback Local" a RADIUS server must be stored and configured for user authentication.

- **Login Authentication**
  Specify how the login is made:
  - Local
    The authentication must be made locally on the device.
  - RADIUS
    The authentication must be handled via a RADIUS server.
  - Local and RADIUS
    The authentication is possible both with the users that exist on the device (user name and password) and via a RADIUS server. The user is first searched for in the local database. If the user does not exist there, a RADIUS query is sent.
  - RADIUS and fallback Local
    The authentication must be handled via a RADIUS server. A local authentication is performed only when the RADIUS server cannot be reached in the network.

RADIUS client

Authentication over an external server

The concept of RADIUS is based on an external authentication server.

Each row of the table contains access data for one server. In the search order, the primary server is queried first. If the primary server cannot be reached, secondary servers are queried in the order in which they are entered. If no server responds, there is no authentication.

Description of the displayed boxes

The page contains the following boxes:

- **RADIUS Authorization Mode**
  For the login authentication, the RADIUS authorization mode specifies how the rights are assigned to the user with a successful authentication.
  - conventional
    In this mode the user is logged in with administrator rights if the server returns the value "Administrative User" to the device for the attribute "Service Type". In all other cases the user is logged in with read rights.
  - Vendor specific
    In this mode the assignment of rights depends on whether and which group the server returns for the user and whether or not there is an entry for the user in the table "External User Accounts".

The table has the following columns:

- Auth. Server Type
  Select which authentication method the server will be used for.
  - Login
    The server is used only for the login authentication.
  - 802.1X
    The server is used only for the 802.1X authentication.
  - Login & 802.1X
    The server is used for both authentication procedures.
- RADIUS Server Address
  Enter the IPv4 address or the FQDN (Fully Qualified Domain Name) of the RADIUS server.
- Server Port
  Here, enter the input port on the RADIUS server. As default, input port 1812 is set. The range of values is 1 ... 65535.
- Shared Secret
  Enter your access ID. The range of values is 1 ... 128 characters.
- Confirm Shared Secret
  Enter your access ID again as confirmation.
- Max. Retrans
  Here, enter the maximum number of retries for an attempted request. The initial connection attempt is repeated the number of times specified here before another configured RADIUS server is queried or the login counts as having failed. As default 3 retries are set, this means 4 connection attempts. The range of values is 1 ... 5.
- Primary Server
  Specify whether or not this server is the primary server. You can select one of the options "yes" or "no".
- Test
  With this button, you can test whether or not the specified RADIUS server is available. The test is performed once and not repeated cyclically.
- Test Result
  Shows whether or not the RADIUS server is available:
  - Failed, no test packet sent
    The IP address is not reachable.
    The IP address is reachable, the RADIUS server is, however, not running.
  - Reachable, key not accepted
    The IP address is reachable, the RADIUS server does not, however accept the shared secret.
  - Reachable, key accepted
    The IP address is reachable, the RADIUS server accepts the specified shared secret.

  The test result is not automatically updated. To delete the test result click the "Refresh" button.

WLAN

Basic (Access Point)

Safety levels

To make the network secure, authentication and encryption are used. On this page, you specify the security settings.

Note

Transmission standard IEEE 802.11 n

The transmission standard IEEE 802.11 n with the setting "802.11n" or "802.11 n only" supports only WPA2/ WPA2-PSK with AES in the security settings.

iPCF, iPCF-HT or iPCF-MC mode activated

If iPCF, iPCF-HT or iPCF-MC is enabled, only "Open System" with the encryption methods WEP or AES is supported in the security settings.

Settings

The table has the following columns:

- **Interface**
  Shows the available interfaces.
- **Authentication Type**
  Select the type of authentication. The selection depends on the operating mode and the transmission standard.
  - **Open System**
    There is no authentication. Encryption with a fixed (unchanging) WEP key can be selected as an option. To use the key, enable "Encryption". You define the WEP key on the "Keys" page.
    If iPCF mode is enabled, the encryption method AES with 128 bit key length is also supported.
  - **Shared Key**
    With this authentication, a fixed key is stored on the client and access point. This WEP key is then used for authentication and encryption. You define the WEP key on the "Keys" page.

    Note

    If you use "Open System" with "Encryption" or "Shared Key", Key 1 must always be set on the "Keys" page.

  - **WPA (RADIUS)**
    Wi-Fi Protected Access is a method specified by the Wi-Fi Alliance to close security gaps in WEP. Authentication using a server is stipulated (802.1x). The dynamic exchange of keys at each data frame introduces further security.
  - **WPA-PSK**
    WPA Pre Shared Key (WPA-PSK) is a weakened form of WPA. In this method, authentication is not established by a server but is based on a password. This password is configured manually on the client and server.
  - **WPA2 (RADIUS)**
    WPA2 (Wi-Fi Protected Access 2) is a further development of WPA and implements the functions of the IEEE 802.11i security standard. WPA authentication works, however, without the RADIUS server.
  - **WPA2-PSK**
    WPA2-PSK is based on the 802.11i standard. WPA authentication works, however, without a RADIUS server. Instead of this, a WPA(2) key (WPA(2) pass phrase) is stored on each client and access point. The WPA(2) pass phrase is used for authentication and further encryption.
  - **WPA/WPA2-Auto-PSK**
    Setting with which an access point can process both the "WPA-PSK" as well as the "WPA2-PSK" type of authentication. This is necessary when the access point communicates with different clients, some using "WPA-PSK" and others "WPA2-PSK". The same encryption method is set on the clients.

- WPA/WPA2-Auto
  Setting with which an access point can process both the "WPA" as well as the "WPA2" type of authentication. This is necessary when the access point communicates with different clients, some using "WPA" and others "WPA2". The same encryption method is set on the clients

- Encryption
  Encryption protects the transferred data from eavesdropping and corruption. You can only disable encryption if you have selected "Open System" for authentication. All other security methods include both authentication and encryption.

- Cipher
  Select the encryption method. The selection depends on the transmission standard.

  - AUTO
    AES or TKIP is used depending on the capability of the other station.

  - WEP
    WEP (Wired Equivalent Privacy)
    A symmetrical stream encryption method with only 40- or 104-bit long keys based on the RC4 algorithm (Ron's Code 4).

  - TKIP (Temporal Key Integrity Protocol)
    A symmetrical encryption method with the RC4 algorithm (Ron's Code 4). In contrast to the weak WEP encryption, TKIP uses changing keys derived from a main key. TKIP can also recognize corrupted data frames.

  - AES (Advanced Encryption Standard)

    Strong symmetrical block encryption method based on the Rijndael algorithm that further improves the functions of TKIP.

    Note

    To provide better protection of your data against attacks, use WPA2/ WPA2-PSK with AES.

- WPA(2) Pass Phrase
  Enter a WPA(2) pass phrase here. This WPA(2) key must be known both at the client end and on the access point and is entered by the user at both ends.
  ASCII code 0x20 to 0x7e is used for the WPA(2) key.

- WPA(2) Pass Phrase Confirmation
  Confirm the entered WPA(2) key.

- Default Key
  Specify the WEP key used to encrypt the data. You define the WEP key on the "Keys" page.

Options

Which settings are possible depends on the "Authentication Type" you have selected.

Authentication type	Encryption	Cipher	Encryption key source
Open System	disabled	--	--
Open System	Enabled	WEP	Default Key
Open System	Enabled	AES[1)]	Default Key (128-bit)

Authentication type	Encryption	Cipher	Encryption key source
Shared Key	Enabled	WEP	Default Key
WPA (RADIUS)	Enabled	Auto/TKIP/AES	RADIUS Server
WPA-PSK	Enabled	Auto/TKIP/AES	WPA(2) Pass Phrase
WPA2 (RADIUS)	Enabled	Auto/TKIP/AES	RADIUS Server
WPA2-PSK	Enabled	Auto/TKIP/AES	WPA(2) Pass Phrase
WPA/WPA2-AutoPSK	Enabled	Auto/TKIP/AES	WPA(2) Pass Phrase
WPA/WPA2-Auto (RADI-US)	Enabled	Auto/TKIP/AES	RADIUS Server

1) available only with iPCF, iPCF-HT or iPCF-MC

Basic (Client)

Safety levels

To make the network secure, authentication and encryption are used. On this page, you specify the security settings.

Note

Transmission standard IEEE 802.11 n

The transmission standard IEEE 802.11 n with the setting "802.11n" or "802.11 n only" supports only WPA2/ WPA2-PSK with AES in the security settings.

iPCF, iPCF-HT or iPCF-MC mode activated

If iPCF, iPCF-HT or iPCF-MC mode is enabled, only "Open System" with the encryption method AES is supported in the security settings with Security Context 1.

Settings

The table has the following columns:

- Security context
  Shows the number of the entry. If you create a new entry, a new row with a unique number is created.
  You can create up to 8 security contexts. The security context 1 cannot be deleted.
- Authentication Type
  Select the type of authentication. The selection depends on the operating mode and the transmission standard.
  - Open System
    There is no authentication. Encryption with a fixed (unchanging) WEP key can be selected as an option. To use the key, enable "Encryption". You define the WEP key on the "Keys" page.
    If iPCF or iPCF-MC mode is enabled, only the encryption method AES with 128 bit key length is supported.
  - Shared Key
    In Shared Key authentication, a fixed key is stored on the client and access point. This WEP key is then used for authentication and encryption. You define the WEP key on the "Keys" page.
  - WPA (RADIUS)
    Wi-Fi Protected Access is a method specified by the Wi-Fi Alliance to close security gaps in WEP. Authentication using a server is stipulated (802.1x). The dynamic exchange of keys at each data frame introduces further security.

    Note

    Make the relevant RADIUS settings initially on the page "Security > WLAN > Client Radius Supplicant".

  - WPA-PSK
    WPA Pre Shared Key (WPA-PSK) is a weakened form of WPA. In this method, authentication is not established by a server but is based on a password. This password is configured manually on the client and server.
  - WPA2 (RADIUS)
    WPA2 (Wi-Fi Protected Access 2) is a further development of WPA and implements the functions of the IEEE 802.11i security standard. WPA authentication works, however, without the RADIUS server.

    Note

    Make the relevant RADIUS settings initially on the page "Security > WLAN > Client Radius Supplicant".

  - WPA2-PSK
    WPA2-PSK is based on the 802.11i standard. WPA authentication works, however, without a RADIUS server. Instead of this, a WPA(2) key (WPA(2) pass phrase) is stored on each client and access point. The WPA(2) pass phrase is used for authentication and further encryption.

- WPA/WPA2-Auto-PSK
  Setting with which an access point can process both the "WPA-PSK" as well as the "WPA2-PSK" type of authentication. This is necessary when the access point communicates with different clients, some using "WPA-PSK" and others "WPA2-PSK". The same encryption method is set on the clients.
- WPA/WPA2-Auto
  Setting with which an access point can process both the "WPA" as well as the "WPA2" type of authentication. This is necessary when the access point communicates with different clients, some using "WPA" and others "WPA2". The same encryption method is set on the clients

- Encryption
  Encryption protects the transferred data from eavesdropping and corruption. You can only disable encryption if you have selected "Open System" for authentication. All other security methods include both authentication and encryption.
- Cipher
  Select the encryption method. The selection depends on the transmission standard.
  - AUTO
    AES or TKIP is used depending on the capability of the other station.
  - WEP
    WEP (Wired Equivalent Privacy)
    A symmetrical stream encryption method with only 40- or 104-bit long keys based on the RC4 algorithm (Ron’s Code 4).
  - TKIP (Temporal Key Integrity Protocol)
    A symmetrical encryption method with the RC4 algorithm (Ron’s Code 4). In contrast to the weak WEP encryption, TKIP uses changing keys derived from a main key. TKIP can also recognize corrupted data frames.
  - AES (Advanced Encryption Standard)

    Strong symmetrical block encryption method based on the Rijndael algorithm that further improves the functions of TKIP.

    Note

    To provide better protection of your data against attacks, use WPA2/ WPA2-PSK with AES.

- WPA(2) Pass Phrase
  Enter a WPA(2) pass phrase here. This WPA(2) key must be known on both the client and the access point and is entered by the user at both ends.
  For a key with 8 to 63 characters, you can only use the following readable ASCII characters: 0x20 - 0x7e.
  For a key with precisely 64 characters, you can use the following ASCII characters: 0 - 9, a - f and A - F.
- WPA(2) Pass Phrase Confirmation
  Confirm the entered WPA(2) key.
- Default Key
  Specify the WEP key used to encrypt the data. You define the WEP key on the "Keys" page.

AP communication

Communications options

On this page, you specify the type of communication allowed by the access point.

Note

The page is available only in access point mode.

Settings

Table 1 has the following columns:

- **Port**
  Shows that the settings are valid for all ports of Table 2.
- **within own VAP / with other VAPs / with Ethernet**
  Select the setting you require. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Radio**
  Shows the available WLAN interfaces.
- **Port**
  Shows the VAP interface.
- **within own VAP**
  - Enabled
    Clients logged on to the same VAP interface of an access point can communicate with each other.
  - Disabled
    Option is disabled.

- **with other VAPs**
  Only available if the device has at least 2 VAP interfaces.
  - Enabled
    Clients logged on to different VAP interfaces of an access point can communicate with each other.
  - Disabled
    Option is disabled.

  Note

  For an access point, "with other VAPs" must be enabled on all WLAN interfaces or on all VAPs to allow communication between clients logged on at different VAP interfaces of the access point.

- **with Ethernet**
  - Enabled
    Clients can communicate via the Ethernet interface of the access point.
  - Disabled
    Option is disabled.

Client RADIUS Supplicant

Client Supplicant

On this page, you configure the settings for the RADIUS authorization of the client.

Note

This page is only available for clients or access points in client mode.

Note

Dot1X user name and Dot1X user password

With WPA (RADIUS), WPA2 (RADIUS), EAP-TLS, EAP-TTLS and PEAP the Dot1X user name and the Dot1X user password must be configured. With the setting "Auto" either the certificate must be loaded or the Dot1X user name and the Dot1X user passport must be configured.

Settings

The table has the following columns:

- **Security context**
  Shows the security context.
- **Dot1x User Name**
  Enter the user name with which the client will log on with the RADIUS server.

- **Dot1x User Password**
  Enter the password for the user name selected above. The client logs on with the RADIUS server using this combination.
  For password assignment, ASCII characters 0x20 to 0x7e are used.
- **Dot1x User Password Confirmation**
  Confirm the password.
- **Dot1X Server Certificate**
  Specify whether or not the RADIUS server identifies itself to the client using a certificate.
- **Dot1x EAP Types**
  Specify the authentication method. The following methods are available:
  - Auto
    Client offers RADIUS server all methods.
  - EAP-TLS
    Extensible Authentication Protocol - Transport Layer Security:
    Uses certificates for authentication.
  - EAP-TTLS
    Extensible Authentication Protocol - Tunnel Transport Layer Security:
    After setting up the TLS tunnel, MS-CHAPv2 is used for internal authentication.
  - PEAP
    Protected Extensible Authentication Protocol
    Alternative draft protocol of IETF for EAP-TTL.

AP RADIUS Authenticator

Configuration of the RADIUS server

On this page, you define the RADIUS servers and the RADIUS authentication of the access point. You can enter data for two RADIUS servers.

Note

The page is available only in access point mode.

Settings

- **Reauthentication Mode**
  Specify who sets the time after which the clients are forced to reauthenticate.
  - '-' Disabled
    Reauthentication mode is disabled.
  - Server
    Enables time management on the server.
  - Local
    Enables local time management. In "Reauthentication Interval", specify the time of validity.
- **Reauthentication Interval [s]**
  If time management is local, enter the period of validity of the authentication in seconds. The minimum time is 1 minute (enter 60), the maximum time is 12 hours (enter 43200). The default is one hour (3,600 seconds).

The table has the following columns:

- **Server IP Address**
  Here, enter the IP address of the RADIUS server.
- **Server Port**
  Here, enter the input port on the RADIUS server.
- **Shared Secret**
  Enter the password of the RADIUS server.
  For the password, ASCII code 0x20 to 0x7e is used.
- **Confirm Shared Secret**
  Confirm the password.
- **Max. Retransmissions**
  Enter the maximum number of connection attempts.
- **Primary Server**
  Specify whether or not this server is the primary server.
  - yes: Primary server
  - no: Backup server.
- **Status**
  With this setting, you can enable or disable the RADIUS server

Key

Specifying the WEP key

To allow you to enable the encryption for the "Open System" and "Shared Key" authentication methods, you must first enter at least one key in the key table.

Settings

The table has the following columns:

- **Key 1 - 4**
  Enter the WEP or AES key. For the WEP key, characters of the ASCII code from 0x20 to 0x7E or hexadecimal characters from 0x00 to 0xFF are permitted.
  If iPCF or iPCF-MC mode is enabled, only the encryption method AES with 128-bit key length is supported.
  You can choose between the following key lengths:
  - 5 or 13 ASCII or 10 or 26 hexadecimal characters (40/104 bits)
  - 16 ASCII or 32 hexadecimal characters (128 bits)

  Note

  The hexadecimal characters are entered without being preceded by "0x". One hexadecimal character codes four bits. The entries "ABCDE" (ASCII characters) and "4142434445" (hexadecimal characters) are therefore the same because the ASCII character "A" has hexadecimal code "0x41".

- **Key 1 - 4 Confirmation**
  Confirm the WEP key.

MAC ACL

Rule configuration

Introduction

On this page, you specify the access rules for the MAC-based Access Control List. Using the MAC-based ACL, you can specify whether frames of certain MAC addresses are forwarded or discarded.

Description of the displayed boxes

The table has the following columns:

- **Rule Number**
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- **Source MAC**
  Enter the unicast MAC address of the source.
  Destination MAC Address
  Enter the unicast MAC address of the destination.

- **Action**
  Select the action. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- **Ingress Interfaces**
  Shows a list of all interfaces to which this rule applies.
- **Egress Interfaces**
  Shows a list of all interfaces to which this rule applies.

Note

Entering the MAC addresses

You can configure the access rules only for unicast MAC addresses. Only if you enter the address "00:00:00:00:00:00" for the source and/or destination MAC address, the rule created in this way applies to all source or destination MAC addresses.

Ingress Rules

Introduction

On this page, you specify the ACL rules according to which incoming frames are filtered at interfaces. You specify the ACL rules in the "Rules Configuration" tab.

Description of the displayed boxes

The page contains the following boxes:

- **Interface**
  Select the required interface from the drop-down list. The available interfaces depend on your device.
- **Add Rule**
  Select the ACL rule to be assigned to the interface. You specify the ACL rule on the "Rules Configuration" page.
- **Add**
  To permanently assign the ACL rule to the interface, click the "Add" button. The configuration is shown in the table.
- **Remove Rule**
  Select the ACL rule to be deleted.
- **Remove**
  To remove the ACL rule from the interface, click the "Remove" button.

The table has the following columns:

- Rule Order
  Shows the order of the ACL rules.
- Rule Number
  Shows the number of the ACL rule.
- Source MAC
  Shows the unicast MAC address of the source.
- Destination MAC Address
  Shows the unicast MAC address of the destination.
- Action
  Shows the action:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.

Egress Rules

Introduction

On this page, you specify the ACL rule according to which outgoing frames are handled by the port. You specify the ACL rule in the "Rules Configuration" tab.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  Select the required interface. The available interfaces depend on your device.
- Add Rule
  Select the ACL rule to be assigned to the interface.
- Add
  To permanently assign the ACL rule to the interface, click the "Add" button. The configuration is shown in the table.
- Remove Rule
  Select the ACL rule to be deleted.
- Remove
  To remove the ACL rule from the interface, click the "Remove" button.

The table has the following columns:

- Rule Order
  Shows the order of the ACL rules.
- Rule Number
  Shows the number of the ACL rule.

- Source MAC
  Shows the unicast MAC address of the source.
- Destination MAC Address
  Shows the unicast MAC address of the destination.
- Action
  Shows the action
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.

IP ACL

Rule configuration

Introduction

On this page, you specify the rules for the IP-based Access Control List. Using the IP-based ACL, you can specify whether frames of certain IPv4 addresses are forwarded or discarded.

Description of the displayed boxes

The table has the following columns:

- Rule Number
  Shows the number of the ACL rule. If you create a new entry, a new row with a unique number is created.
- Source IP
  Enter the IPv4 address of the source.
- Source Subnet Mask
  Enter the subnet mask in which the source is located.
- Dest. IP
  Enter the IP address of the destination.
- Dest. Subnet Mask
  Enter the subnet mask in which the destination is located.
- Action
  Select whether the frame is forwarded or rejected when it corresponds to the ACL rule.
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.

- Ingress Interfaces
  Shows a list of all interfaces to which this rule applies.
- Egress Interfaces
  Shows a list of all interfaces to which this rule applies.

Note

Subnet mask for individual hosts

If you create the rule for a single system (one IPv4 address), specify the subnet mask "255.255.255.255".

Protocol configuration

On this page, you specify the rules for protocols.

Description of the displayed boxes

The table has the following columns:

- Rule Number
  Shows the number of the protocol rule. When you create a rule, a new row with a unique number is created.
- Protocol
  Select the protocol for which this rule is valid.
- Protocol Number
  Enter a protocol number to define further protocols. This box can only be edited if you have set "Other Protocol" for the protocol.
- Source Port Min.
  Enter the lowest possible port number of the source port. This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Source Port Max.
  Enter the highest possible port number of the source port. This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Dest. Port Min.
  Enter the lowest possible port number of the destination port. This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Dest. Port Max.
  Enter the highest possible port number of the destination port. This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Message Type
  Enter a message type to decide the format of the message.
  This box can only be edited if you have set "ICMP" for the protocol.

- Message Code
  Enter a message code to specify the function of the message. This box can only be edited if you have set "ICMP" for the protocol.
- DSCP
  Enter a value for classifying the priority. This box cannot be edited if you have set "ICMP" for the protocol.

Ingress Rules

Introduction

On this page, you specify the ACL rules according to which incoming frames are handled by interfaces. You specify the ACL rules in the "Rules Configuration" tab.

Description of the displayed boxes

Description of the displayed boxes:

- Interface
  Select the required port: a port or a VLAN interface. The available interfaces depend on your device. To select a VLAN interface, an IP interface must be configured.

  Note

  If you use a VLAN interface, the ACL rule applies to all ports that belong to the VLAN.

- Add Rule
  Select the ACL rule to be assigned to the port.
- Add
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- Remove Rule
  Select the ACL rule to be deleted.
- Remove
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- Rule Order
  Shows the order of the ACL rules.
- Rule Number
  Shows the number of the ACL rule.
- Protocol
  Shows the protocol for which this rule is valid.
- Protocol Number
  Shows the protocol number.
- Source IP
  Shows the IP address of the source.

- Source Subnet Mask
  Shows the subnet mask in which the source is located.
- Dest. IP
  Shows the IP address of the destination.
- Dest. Subnet Mask
  Shows the subnet mask of the destination.
- Action
  Select whether the frame is forwarded or rejected when it corresponds to the ACL rule.
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- Source Port Min.
  Shows the lowest possible port number of the source port.
- Source Port Max.
  Shows the highest possible port number of the source port.
- Dest. Port Min.
  Shows the lowest possible port number of the destination port.
- Dest. Port Max.
  Shows the highest possible port number of the destination port.
- Message Type
  Shows a message type to decide the format of the message.
- Message Code
  Shows a message code to specify the function of the message.
- DSCP
  Shows the value for classifying the priority.

Egress Rules

Introduction

On this page, you specify the ACL rule according to which outgoing frames are handled by the port. You specify the ACL rules in the "Rules Configuration" tab.

Description of the displayed boxes

The page contains the following boxes:

- Interface
  Select the required port: a port or a VLAN interface. The available interfaces depend on your device. To select a VLAN interface, an IP interface must be configured.

  Note

  If you use a VLAN interface, the ACL rule applies to all ports that belong to the VLAN.

- Add Rule
  Select the ACL rule to be assigned to the port.
- Add
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- Remove Rule
  Select the ACL rule to be deleted.
- Remove
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- Rule Order
  Shows the order of the ACL rules.
- Rule Number
  Shows the number of the ACL rule.
- Protocol
  Shows the protocol for which this rule is valid.
- Protocol Number
  Shows the protocol number.
- Source IP
  Shows the IP address of the source.
- Source Subnet Mask
  Shows the subnet mask of the source.
- Dest. IP
  Shows the IP address of the destination.
- Dest. Subnet Mask
  Shows the subnet mask of the source.
- Action
  Select whether the frame is forwarded or rejected when it corresponds to the ACL rule.
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- Source Port Min.
  Shows the lowest possible port number of the source port.

- Source Port Max.
  Shows the highest possible port number of the source port.
- Dest. Port Min.
  Shows the lowest possible port number of the destination port.
- Dest. Port Max.
  Shows the highest possible port number of the destination port.
- Message Type
  Shows a message type to decide the format of the message.
- Message Code
  Shows a message code to specify the function of the message.
- DSCP
  Shows the value for classifying the priority.

Management ACL

On this page, you can increase the security of your device. To specify which station with which IP address is allowed to access your device, configure the IP address or an entire address range.

You can select the protocols and the ports with which the station is allowed to access the device. You define the VLAN in which the station may be located. This ensures that only certain stations within a VLAN have access to the device.

Note

Note that a bad configuration may mean that you can no longer access the device.

Settings

- Management ACL
  Enable or disable the function.
- IP Address
  Enter the IPv4 address or the network address to which the rule will apply. If you use the IP address 0.0.0.0, the settings apply to all IP addresses.
- Subnet Mask / Prefix Length
  Enter the subnet mask or the prefix length.
  The subnet mask 255.255.255.255 is for a specific IPv4 address. If you want to allow a subnet, for example for a C subnet enter 255.255.255.0. The subnet mask 0.0.0.0 applies to all subnets.

The table has the following columns:

- Select
  Select the lines to be deleted.
- Rule Order
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.

- **IP Address**
  Shows the IP address.
- **Subnet Mask / Prefix Length**
  Shows the subnet mask or the prefix length.
- **VLANs Allowed**
  Enter the number of the VLAN in which the device is located. The station can only access the device if it is located in this configured VLAN. If this input box remains empty, there is no restriction.
- **SNMP**
  Specify whether the station or the IP address accesses the device using the SNMP protocol.
- **TELNET**
  Specify whether the station or the IP address accesses the device using the TELNET protocol.
- **HTTP**
  Specify whether the station or the IP address accesses the device using the HTTP protocol.
- **HTTPS**
  Specify whether the station or the IP address accesses the device using the HTTPS protocol.
- **SSH**
  Specify whether the station or the IP address accesses the device using the SSH protocol.
- **Px**
  Specify whether the station or the IP address accesses the device via this port.
- **WLAN1** (In client mode only)
  Specify whether the station or the IP address accesses the device via the WLAN interface.
- **VAPX.Y** (only in access point mode)
  Specify whether the station or the IP address accesses the device via the VAP interface.
- **WDSX.Y** (only in access point mode)
  Specify whether the station or the IP address accesses the device via the WDS interface.

Inter AP blocking

Basic

Note

- This tab is only available.
  - SCALANCE W780 / W770
  - in access point mode
- This WBM page can only be configured with the following KEY-PLUGs:
  - W780 iFeatures (MLFB 6GK5 907-8PA00)
  - W700 Security (MLFB 6GK5907-0PA00)

When should Inter AP blocking be used?

The clients connected to an access point can normally communicate with all devices of the layer 2 network.

With inter AP blocking, the communication of the clients connected to the access point can be restricted. Only the devices whose IP addresses are configured in "Allowed Addresses" on the access point are accessible to the clients. Communication with other nodes in the network is therefore prevented.

Settings

- **Refresh Interval [s]**
  Enter the refresh interval for the ARP table.

The table has the following columns:

- **Radio**
  Specifies the WLAN interface to which the settings relate.
- **Port**
  Specifies the VAP interface to which the settings relate.
- **SSID**
  Specifies the SSID to which the settings relate.
- **Enabled**
  When selected, the access restriction is used. You configure the devices that are accessible for the clients in "Security > Inter AP Blocking > Allow IP Addresses".
- **Block Gratuitous ARP Requests**
  When enabled, gratuitous ARP packets are not forwarded.
- **Block Non-IP Frames**
  When enabled, there is no exchange of non-IP packets, for example layer 2 packets between the client and the devices configured on the access point as permitted communications partners.

Allowed Addresses

On this page you specify which devices are accessible to the clients.

Note

- This tab is only available.
  - SCALANCE W780 / W770
  - in access point mode
- This WBM page can only be configured with the following KEY-PLUGs:
  - W780 iFeatures (MLFB 6GK5 907-8PA00)
  - W700 Security (MLFB 6GK5907-0PA00)

Settings

- **Port**
  Select the required port.
- **IP address**
  Enter the IP address of the devices accessible to the client.

The table has the following columns:

- **Select**
  Select the lines to be deleted.
- **Radio**
  Specifies the WLAN interface to which the settings relate
- **Port**
  Specifies the VAP interface to which the settings relate
- **IP address**
  IP Address of the devices accessible to the client. If necessary, you can change the IP address.
- **Resolver IP Address**
  The IP address with which the allowed IP address is resolved. The entry is necessary if the management IP address is located in a different subnet. If the IP address "0.0.0.0" is configured for "IP address resolution", the management IP address is used for the resolution.

Configuring iFeatures

iPCF

Note

iPCF for clients or access points in client mode

- SCALANCE W740, SCALANCE W734-1 RJ-45 and SCALANCE W722-1 RJ-45
- The setting can only be configured with the following KEY-PLUG:
  - Client: W740 iFeatures (MLFB 6GK5 907-4PA00)

iPCF in access point mode

- SCALANCE W780 and SCALANCE W774-1 RJ-45
- The setting can only be configured with the following KEY-PLUG:
  - Access point: W780 iFeatures (MLFB 6GK5 907-8PA00)

When should iPCF be used?

The use of iPCF is advisable particularly if you have a large number of nodes and want to implement highly deterministic operation. This is necessary, for example with PROFINET or other cyclic protocols.

Note

Use of iPCF with other iFeatures

iPCF and other iFeatures (e.g. iPCF-MC, iPRP) are not compatible with each other and cannot be used at the same time on a device.

Settings

In both modes, the table has the following columns:

- **Radio**
  Specifies the WLAN interface to which the settings relate.
- **Enable iPCF**
  Enable or disable iPCF mode. For PROFINET communication, we recommend that you enable the iPCF mode. By enabling iPCF, the data rates provided by the access point are adapted. We strongly recommend that you retain the default setting for the data rates (802.11 a/b/g = 12 Mbps and 802.11n = MCS 2).

In access point mode, the table has the following additional columns:

- **Protocol Support**
  Specify which protocol is handled with priority by the WLAN interface.
  - PROFINET
    if you set PROFINET, there must be no PROFINET controller downstream from the client.
  - EtherNet/IP
    If you set Ethernet/IP, there must be no scanner downstream from the client.
  - Disabled
    The function is disabled.
- **iPCF Cycle Time [ms]**
  Select the required cycle time from the drop-down list.
  The following points need to be taken into account when setting the cycle time. Otherwise it may not be possible to establish stable communication.
  - There is only one access point in the system; in other words, the clients move only in one wireless cell. In this case, update times >= 16 ms are supported.
  - There are several access points in the system that communicate over different channels. The clients roam between the access points. In this case, select update times >= 32 ms.

  In addition to the guide values shown above, remember that the shortest cycle time to be set is calculated according to the formula "2 ms * max. number of nodes".

- Scanning Mode
  The selected setting affects the scanning of the logged-on clients.
  The following settings are available:
  - All Channels
    The client scans all permitted channels and selects the access point with the best signal strength and connects to it.
  - Next Channel
    The client scans the next channel from its permitted channel list. If an access point is there, it connects to it. If it does not find an access point on this channel, it scans the next channel.
- Signal Quality Threshold
  Can only be configured if "Next Channel" is set for "Scanning Mode".
  The access point specifies a signal quality for the client. When scanning the client must receive the signal coming from the access point with at least the specified signal quality. Only then is a connection established. The signal quality is determined by the client based on the RSSI values (Received Signal Strength Indicator) of received packets. The RSSI value indicates how strong the arriving signal is and is displayed in the signal recorder.
  The following threshold values apply to the signal strength:

Range	Signal quality in %	Signal quality in RSSI
1	40	20
2	50	25
3	60	30
4	70	35
5	80	40

iPCF-HT

Note

iPCF for clients or access points in client mode

- SCALANCE W740, SCALANCE W734-1 RJ-45 and SCALANCE W722-1 RJ-45
- The setting can only be configured with the following KEY-PLUG:
  - Client: W740 iFeatures (MLFB 6GK5 907-4PA00)

iPCF in access point mode

- SCALANCE W780 and SCALANCE W774-1 RJ-45
- The setting can only be configured with the following KEY-PLUG:
  - Access point: W780 iFeatures (MLFB 6GK5 907-8PA00)

When should iPCF-HT (High Throughput) be used?

The use of iPCF is advisable particularly if you have a large number of nodes and want to implement highly deterministic operation. This is necessary, for example with PROFINET or other cyclic protocols.

Note

Use of iPCF-HT

The function iPCF-HT

- and other iFeatures (e.g. iPCF, iPCF-MC, iPRP) are not compatible with each other and cannot be used at the same time on a device.
- can only be used in the frequency band 5 GHz and with WLAN mode "(only) IEEE 802.11n".
- is available only on the WLAN interface 1.

It is advisable only to use an MCS index.

Description

In both modes, the table has the following columns:

- **Radio**
  Specifies the WLAN interface to which the settings relate.
- **Enable iPCF-HT**
  Enable or disable iPCF-HT. When enabled, the data rates provided by the access point are adapted. We strongly recommend that you retain the default setting for the data rates (802.11n = MCS 2).

In access point mode, the table has the following additional columns:

- **Protocol Support**
  Specify which protocol is handled with priority by the WLAN interface.
  - PROFINET
    if you set PROFINET, there must be no PROFINET controller downstream from the client.
  - EtherNet/IP
    If you set Ethernet/IP, there must be no scanner downstream from the client.
  - Disabled
    The function is disabled.
- **iPCF-HT Cycle Time [ms]**
  Specify the cycle time.
  The following points need to be taken into account when setting the cycle time. Otherwise it may not be possible to establish stable communication.
  - The range of values is 16 - 512. The set value should correspond the cycle time of PROFINET or EtherNet/IP.
  - There is only one access point in the system; in other words, the clients move only in one wireless cell. In this case, update times >= 16 ms are supported.
  - There are several access points in the system that communicate over different channels. The clients roam between the access points. In this case, select update times >= 32 ms.

  In addition to the guide values shown above, remember that the shortest cycle time to be set is calculated according to the formula "4 ms * max. number of nodes".

- Scanning Mode
  The selected setting affects the scanning of the logged-on clients.
  The following settings are available:
  - All Channels
    The client scans all permitted channels and selects the access point with the best signal strength and connects to it.
  - Next Channel
    The client scans the next channel from its permitted channel list. If an access point is there, it connects to it. If it does not find an access point on this channel, it scans the next channel.
- Signal Quality Threshold
  Can only be configured if "Next Channel" is set for "Scanning Mode".
  The access point specifies a signal quality for the client. When scanning the client must receive the signal coming from the access point with at least the specified signal quality. Only then is a connection established.
  The signal quality is determined by the client based on the RSSI values (Received Signal Strength Indicator) of received packets. The RSSI value indicates how strong the arriving signal is and is displayed in the signal recorder.
  The following threshold values apply to the signal strength:

Range	Signal quality in %	Signal quality in RSSI
1	40	20
2	50	25
3	60	30
4	70	35
5	80	40

iPCF-MC

Requirements to be able to use iPCF-MC:

- The access point has at least two WLAN interfaces (dual AP).
- Access point mode: Only dual APs with KEY-PLUG W780 iFeatures (MLFB 6GK5 907-8PA00)
- Client mode: Client with KEY-PLUG W740 iFeatures (MLFB 6GK5 907-4PA00)
- The management interface and data interface must be operated in the same frequency band and mode and must match in terms of their wireless coverage. iPCF-MC will not work if both wireless interfaces are equipped with directional antennas that cover different areas.
- The management interfaces of all access points to which a client can change must use the same channel. A client scans only this one channel to find accessible access points.
- Transmission based on IEEE801.11h (DFS) cannot be used for the management interface. 801.11h (DFS) is possible for the data interface.
- The client cannot be operated with "Use Allowed Channels only".
- "Force Roaming on link down" is automatically mirrored on the second interface.

- The following applies to clients: All configured and active SSIDs must be assigned to security context 1. An SSID is active when the corresponding check box "Enabled" is selected on the page "Interfaces > WLAN > Client".
- In Japan, iPCF-MC cannot be enabled if the data or management interface uses a frequency of the 4920 MHz - 5080 MH frequency band.

When should iPCF-MC be used?

iPCF was developed to achieve short handover times when roaming between cells. The iPCF-MC technique allows short handover times even for freely mobile clients and when a lot of cells are involved or a large number of channels is being used.

Note

Use of iPCF with other iFeatures

The function iPCF-MC and other iFeatures (e.g.iPCF, iPCF-HTMC, iPRP) are not compatible with each other and cannot be used at the same time on a device.

Assignment of the interfaces

With 11n devices, remember that the assignment of the WLAN interfaces is fixed for iPCF-MC.

- WLAN1: Data interface
- WLAN2: Management interface

Settings

The following setting exists in both modes

- **Enable iPCF-MC**
  Enable or disable the iPCF-MC mode of the device.
  For PROFINET communication, we recommend enabling the iPCF-MC mode. By enabling iPCF-MC, the data rates provided by the access point are adapted.
  We strongly recommend that you retain the default setting for the data rates (802.11 a/b/g = 12 Mbps and 802.11n = MCS 2).
- **iPCF Cycle Time** (only in access point mode)
  Select the update time configured for the network to which the access point is connected. The lowest value for the update time is 32 ms.
- **Protocol Support** (only in access point mode)
  Specify which protocol is handled with priority.
  - PROFINET
    if you set PROFINET, there must be no PROFINET controller downstream from the client.
  - EtherNet/IP
    If you set Ethernet/IP, there must be no scanner downstream from the client.

- **Management Scan Period** (in client mode only)
  This parameter specifies the time between two client background scans (specified in iPCF cycles). If, for example, you select two, the client runs a background scan only in every second iPCF cycle.
  A lower value for the background scan interval provides the basis for fast roaming, however this means that no high data throughput can be achieved. A higher value should be selected for a high data throughput.
- **Management Scan Period**
  This parameter specifies the time between two client background scans (specified in iPCF cycles). If, for example, you select two, the client runs a background scan only in every second iPCF cycle.
  A lower value for the background scan interval provides the basis for fast roaming, however this means that no high data throughput can be achieved. A higher value should be selected for a high data throughput.
- **Roaming Filter** (in client mode only)
  With this setting you specify the number of RSSI single measurements from which the median is determined. With 5, the last 5 measured RSSI values are considered.
  - Median with an odd number of measurements
    The values are arranged in ascending order. The value exactly in the middle is the median.
  - Median with an even number of measurements
    The values are arranged in ascending order. The median is calculated from the average of the two middle numbers.

  If occasionally there are extreme outliers of the incoming signal, you can filter out the worst fluctuations with this roaming filter. This prevents premature roaming of the client.

iPRP

Note

- SCALANCE W770/W740, SCALANCE W770/W730 and SCALANCE W722-1 RJ45

The following KEY-PLUGs are inserted:

- Access point: W780 iFeatures (MLFB 6GK5 907-8PA00)
- Client: W740 iFeatures (MLFB 6GK5 907-4PA00)

When should iPRP be used?

Note

Use of iPRP with other iFeatures

IPRP and other iFeatures (e.g. iPCF. iPCF-HT, iPCF-MC) are not compatible with each other and cannot be used at the same time on a device.

iPRP with oversize frames (jumbo frames)

To be able to use oversize frames, oversize frames (jumbo frames) must be configured on all devices in the network.

Agent VLAN (management VLAN) with iPRP

The iPRP VLAN can be used as the agent VLAN. This depends where the device is located.

- If the device is located in the PRP network A or PRP network B, as the agent VLAN use the VLAN that PRPA or PRPB is assigned to.
- If the access points are located in both PRP networks, you can use one of the two VLANs as the agent VLAN. As an alternative you can also use other VLANs as agent VLANs. The division into PRP networks A and B must remain. A single management VLAN for all devices in network A and B is not possible without further measures.

With the "industrial Parallel Redundancy Protocol" (iPRP) the PRP technology can be used in a wireless network. With IPRP the PRP frames are transferred parallel via two wireless links. The parallel transfer allows disruptions of the transfer on one wireless link to be compensated on the other.

Requirement

- The Base Bridge mode "802.1Q VLAN Bridge" is set.
- The VLANs have been created.
- Access point mode: The VAP interface is enabled.
- Client mode: For "MAC Mode", Layer 2 Tunnel is set.

Settings

The page contains the following:

- **PRP A**
  Select the VLAN assignment for PRP from the drop-down list.
- **PRP B**
  Select the VLAN assignment for PRP B from the drop-down list.

This table contains the following columns:

- Interface
  Shows the available interfaces.

  Note
  Access point
  On the access point the iPRP function is only supported on a VAP interface.

- Enable iPRP
  Enable or disable iPRP for the required interface.
- PRP Network
  Specify the PRP network in which the interface is a member.
- AP Radio Redundancy (in client mode only)
  - Enabled
    Prevents the two clients of a client pair connecting on the same WLAN interface of the access point.
  - Disabled
    When the best access point for a client is the same access point (same WLAN interface) as that of the partner client, a check is made as to whether there is another access point whose signal strength is < 10 dB worse than that of the best access point. In this case the client connects to this access point, otherwise it connects to the same, best access point as the partner client.

iREF

Note
- The page is available only in access point mode.
- The function is supported by SCALANCE W780 / W770.
- The setting can only be configured with the following KEY-PLUG:
  - Access point: W780 iFeatures (MLFB 6GK5 907-8PA00)

When should iREF be used?

With iREF, the data can be sent with the highest possible transmit power. This applies in particular to use cases in which MIMO cannot be used, is of no advantage or when using sector antennas.

Note
Use of iREF with other iFeatures
iREF and other iFeatures (e.g. iPCF) are not compatible with each other and cannot be used at the same time on a device.

Settings

The table has the following columns:

- **Radio**
  Specifies the WLAN interface to which the settings relate.
- **Enable iREF**
  Enable or disable iREF for the required WLAN interface. The result is shown in "Information > WLAN >iFeatures".

Note

To be able to use iREF, there must be at least two antennas.

The antennas are automatically switched to antenna mode RX/TX as soon as iREF is enabled.

AeroScout

Note

- The page is only available
  - SCALANCE W780 / W770
  - in access point mode
- The setting can only be configured with the following KEY-PLUG:
  - Access point: W780 iFeatures (MLFB 6GK5 907-8PA00)

Note

The AeroScout function cannot be combined with other iFeatures (iPCF,iPCF-MC, iREF).

AeroScout can only be used in the 2.4 GHz band according to IEEE 802.11g, IEEE 802.11n and IEEE 802.11n-only.

For more detailed information, please refer to the AeroScout documentation (www.aeroscout.com).

Settings

The table has the following columns:

- **Radio**
  Specifies the WLAN interface to which the settings relate.
- **Enable AeroScout**
  Enable or disable AeroScout support for the required WLAN interface. The result is shown in "Information > WLAN iFeatures". This page is only available in the WBM.

Configuring SCALANCE M

Supported devices

Supported devices

The functions described in this help section are supported by the following products:

- SCALANCE M874, M876 V5.0
- SCALANCE M812, M816 V5.0
- SCALANCE M826 V5.0

General terminology "device"

In this section of the information system the term "device" is used instead of the product names listed above. You will find the functional differences in the section Hardware equipment and system functions (Page 2551) Depending on the selected device, different parameters are available on the individual configuration pages.

Hardware equipment and system functions

Availability of hardware

The following table shows the hardware of the devices.

We reserve the right to make technical changes.

	SCALANCE M874-2 SCALANCE M874-3	SCALANCE M876-3 SCALANCE M876-4	SCALANCE M812	SCALANCE M816	SCA-LANCE M826	SCALANCE M804PB
Interfaces	1 x SMA antenna connector	2 x SMA antenna connector	1 x DSL	1 x DSL	2 x SHDSL	1 x RS 485 (MPI/DP)
	2 x RJ45	4 x RJ45	1x RJ45 (LAN)	4 x RJ45 (LAN)	4 x RJ45	2 x RJ45
Digital input/output	✓	✓	✓	✓	✓	✓
PLUG slot	✓	✓	-	✓	✓	✓
SET button	✓	✓	✓	✓	✓	✓

Availability of the system functions

The following table shows the availability of the system functions on the devices. Note that all functions are described in this configuration manual and in the online help. Depending on the KEY-PLUG, some functions are not available.

We reserve the right to make technical changes.

		SCA-LANCE M87x	SCA-LANCE M812	SCA-LANCE M816	SCA-LANCE M826	SCA-LANCE M804PB
Basic Wizard	IP settings	✓	✓	✓	✓	✓
	Device settings	✓	✓	✓	✓	✓
	SIM	✓	-	-	-	-
	Mobile wireless provider	✓	-	-	-	-
	DSL	-	✓	✓	-	-
	SHDSL	-	-	-	✓	-
	TIA Portal Cloud Connector	-	-	-	-	✓
	Time settings	✓	✓	✓	✓	✓
	SINEMA RC [1)]	✓	-	✓	✓	✓
	DDNS	✓	✓	✓	✓	✓
Information	ARP Table	✓	✓	✓	✓	✓
	Log Tables	✓	✓	✓	✓	✓
	Mobile wireless	✓	-	-	-	-
	DSL	-	✓	✓	-	-
	SHDSL	✓	✓	✓	✓	-
	Redundancy	✓	✓	✓	✓	✓
	VRRPv3	✓	✓	✓	-	-
	SINEMA RC [1)]	✓	-	✓	✓	✓
System	SMTP client	✓	✓	✓	✓	✓
	SNMP	✓	✓	✓	✓	✓
	Time setting	✓	✓	✓	✓	✓
	Automatic logout	✓	✓	✓	✓	✓
	Syslog client	✓	✓	✓	✓	✓
	Fault Monitoring	✓	✓	✓	✓	✓
	PLUG	✓	-	✓	✓	✓
	SMS	✓	-	-	-	-
	DNS	✓	✓	✓	✓	✓
	DHCP Client	-	-	-	✓	-
	DHCP Server	✓	✓	✓	✓	✓
	cRSP/SRS	✓	✓	✓	✓	✓
	Proxy Server	✓	✓	✓	✓	✓
	TIA Portal Cloud Connector	-	-	-	-	✓
	SINEMA RC[1)]	✓	-	✓	✓	✓

		SCA-LANCE M87x	SCA-LANCE M812	SCA-LANCE M816	SCA-LANCE M826	SCA-LANCE M804PB
Interfaces	Ethernet	✓	✓	✓	✓	✓
	Mobile wireless	✓	-	-	-	-
	DSL	-	✓	✓	-	-
	SHDSL	-	-	-	✓	-
Layer 2	Configuration	✓	✓	✓	✓	✓
	VLAN	✓	✓	✓	✓	✓
	Dynamic MAC aging	✓	✓	✓	✓	✓
	LLDP	✓	✓	✓	✓	✓
	Spanning Tree	✓	✓	✓	✓	✓
Layer 3	Static routes	✓	✓	✓	✓	✓
	Subnets	✓	✓	✓	✓	✓
	Spanning Tree	✓	✓	✓	✓	✓
	NAT	✓	✓	✓	✓	✓
	VRRPv3	✓	✓	✓	-	-
Security	Passwords	✓	✓	✓	✓	✓
	User	✓	✓	✓	✓	✓
	AAA (Authentication, Authorization, Accounting)	✓	✓	✓	✓	✓
	Certificates	✓	✓	✓	✓	✓
	Firewall	✓	✓	✓	✓	✓
	IPsec VPN	✓	✓	✓	✓	✓
	OpenVPN	✓	✓	✓	✓	✓

1) KEY-PLUG SINEMA Remote Connect 6GK5908-0PB00

Special features of user management for security functions

Access to security functions

You can only access security functions after activating project protection and after a user has successfully logged on. The logged on user must also have the required engineering and runtime rights.

You will find basic information on user management in STEP 7 in the "Using user management" section of the information system.

Engineering rights for security devices

The engineering rights allow diagnostics and configuration of security devices:

- Security: Security modules with read rights: Show the global security functions and the local security settings of the security devices.
- Security: Security modules with write rights: Configure the global security functions and the local security settings of the security devices and perform diagnostics of the security devices.

Roles with runtime rights for SCALANCE M-800

By default, some system-defined roles with the prefix "NET" are assigned runtime rights for SCALANCE M-800. Runtime rights provide the rights to execute certain tasks on the devices. Just as with engineering rights you cannot change runtime rights for system-defined roles. For user-defined roles you can change the runtime rights as required. You specify the runtime rights of a user-defined role for each device.

There must be at least one user assigned a role with full runtime rights for a security module, e.g. the system-defined role "NET Administrator".

The system-defined roles with the prefix "NET" are assigned the following engineering rights:

Right	NET Administrator	NET Standard	NET Diagnose	NET Remote Access	NET Administrator Radius	NET Radius
Open project read-only	x	x	x	-	-	-
Open project with write rights	x	x	x	-	-	-
Edit security settings "Settings" and "Users and roles"	x	-	-	-	-	-
Security: Security modules with read rights	x	x	x	-	x	x
Security: Security modules with write rights	x	x	-	-	x	-

Runtime rights for SCALANCE M-800

The following runtime rights are available:

Right	NET Administrator	NET Standard	NET Diagnose	NET Remote Access	NET Administrator Radius	NET Radius
Download the configuration files	x	x	-	-	x	-
TIA Portal: Perform diagnostics of the security module	x	x	x	-	x	x

Certificates for security functions

During configuration only certificates generated by the TIA Portal can be used, for example VPN group certificates. Certificates not generated by the TIA Portal. for example CA certificates for SINEMA Remote Connect, cannot be used. You need to add such certificates later via the WBM and download them to the device. You will find more information in the relevant WBM configuration manual.

Editing properties and parameters

Editing options

You have the following options for editing properties and parameters:

- Hardware and network editor
  Once you have inserted the network component, you can edit the properties and parameters, for example the device name. You will find more detailed information in "Hardware and network editor".
- Web Based Management (WBM)
  The parameters and properties are accessible using the supplied HTML pages (WBM pages). Each WBM page has its own help page that describes the properties and parameters. For further information, refer to the configuration manuals:
  - "SCALANCE M-800 Web Based Management"
    You will find this configuration manual on the Internet pages of Siemens Industry Online Support in (https://support.industry.siemens.com/cs/ww/en/ps/15982/man):
- Command Line Interface
  All the configuration settings for the device can be made using the CLI. The CLI provides the same options as Web Based Management (WBM). For further information, refer to the configuration manuals:
  - "SCALANCE M-800 Command Line Interface"
    You will find this configuration manual on the Internet pages of Siemens Industry Online Support in:

Availability

The availability depends on the SCALANCE M device.

The following configuration modes are available:

- Offline configuration
  You can configure a PC station the first time offline. In this mode only the settings that require no connection to the device are available.
- Online diagnostics mode
  If there is an online connection to the device additional pages are available in "Online & Diagnostics". The documentation of these pages includes the note "This page is only available if there is an online connection to the device". With some settings, the additional information "only available online" is included.

Creating and deleting an entry

As an example, an entry will be created and deleted on the Syslog client. The procedure is valid on all pages.

Creating a new entry

1. Select the device in the network or device view.
2. Open the properties of the device in the Inspector window.
3. In the Inspector window go to System > Syslog client.
4. Enter the IP address for the Syslog server.
5. Click in the table.
6. Select the entry "New entry" in the shortcut menu.
   A new entry is created in the table.

Deleting an entry

1. Select the required entry in the table.
2. Select the "Delete" entry in the shortcut menu.

Buttons you require often

- **Refresh the display with "Refresh"**
  Pages that display current parameters have a "Refresh" button at the lower edge of the page. Click this button to request up-to-date information from the device for the current page.
- **Save settings for all ports with "Transfer to table"**
  Pages on which you can configure several ports have 2 tables. In the first table, you can make the settings for all ports and transfer these to the second table. In the last column of the first table, there is the "Transfer to table" button. Click this button to save the settings you have made for all ports.

Error message when downloading the configuration

Potential problems during downloading

Status messages about the progress of the download process are displayed in the output window while the configuration is downloaded to a SCALANCE device.

If problems occur during download to the device, they are usually caused by one of the following:

- Incorrect SCALANCE device in the project
- Incorrect firmware version
- The SCALANCE device is not connected to the configuration PC.

Error message

The following error message can occur during downloading when no connection can be established:

Possible errors	Possible causes	Possible remedy
Error message "The destination device currently cannot accept a download of data"	The destination device is currently in an active application relation with an IO controller.	Disable the application relation between the IO controller and the SCALANCE device.
	The TIA configuration interface on the destination device is disabled.	Enable the TIA configuration interface on the SCALANCE device via WBM or CLI.

Useful information

IPv4 address, subnet mask and address of the gateway

Range of values for IPv4 address

The IPv4 address consists of four decimal numbers with the range from 0 to 255, each number separated by a period; example: 141.80.0.16

IPv4 address format - notation

An IPv4 address consists of 4 bytes. Each byte is represented in decimal, with a dot separating it from the previous one.

XXX.XXX.XXX.XXX

XXX stands for a number between 0 and 255

The IPv4 address consists of two parts:

- The address of the (sub) network
- The address of the node (generally also called end node, host or network node)

Range of values for subnet mask

The subnet mask consists of four decimal numbers with the range from 0 to 255, each number separated by a period; example: 255.255.0.0

The binary representation of the 4 subnet mask decimal numbers must contain a series of consecutive 1s from the left and a series of consecutive 0s from the right.

The 1s specify the network number within the IPv4 address. The 0s specify the host address within the IPv4 address.

Example:

Correct values:

255.255.0.0 D = 1111 1111.1111 1111.0000 0000.0000 0000 B

255.255.128.0 D = 1111 1111.1111 1111.1000 0000.0000 0000 B

255.254.0.0 D = 1111 1111.1111 1110.0000 0000.0000.0000 B

Incorrect value:

255.255.1.0 D = 1111 1111.1111 1111.0000 0001.0000 0000 B

Relationship between the IPv4 address and subnet mask

The first decimal number of the IPv4 address (from the left) determines the structure of the subnet mask with regard to the number of "1" values (binary) as follows (where "x" is the host address):

First decimal number of the IPv4 address	Subnet mask
0 to 127	255.x.x.x
128 to 191	255.255.x.x
192 to 223	255.255.255.x

Classless Inter-Domain Routing (CIDR)

CIDR is a method that groups several IPv4 addresses into an address range by representing an IPv4 address combined with its subnet mask. To do this, a suffix is appended to the IPv4 address that specifies the number of bits of the network mask set to 1. Using the CIDR notation, routing tables can be reduced in size and the available address ranges put to better use.

Example:

IPv4 address 192.168.0.0 with subnet mask 255.255.255.0

The network part of the address covers 3 x 8 bits in binary representation; in other words 24 bits.

This results in the CIDR notation 192.168.0.0/24.
The host part covers 1 x 8 bits in binary notation. This results in an address range of 2 to the power 8, in other words 256 possible addresses.

Value range for gateway address

The address consists of four decimal numbers taken from the range 0 to 255, each number being separated by a period; example: 141.80.0.1

Relationship between IPv4 address and gateway address

The only positions of the IPv4 address and gateway address that may differ are those in which "0" appears in the subnet mask.

Example:

You have entered the following: 255.255.255.0 for the subnet mask; 141.30.0.5 for the IPv4 address and 141.30.128.0 for the gateway address. The Ipv4 address and gateway address may only be different in the 4th decimal number. In the example, however, the 3rd position is different.

You must, therefore, change one of the following in the example:

The subnet mask to: 255.255.0.0 or

the IPv4 address to: 141.30.128.1 or

the gateway address to: 141.30.0.1

VLAN

VLAN

Network definition regardless of the spatial location of the nodes

VLAN (Virtual Local Area Network) divides a physical network into several logical networks that are shielded from each other. Here, devices are grouped together to form logical groups. Only nodes of the same VLAN can address each other. Since multicast and broadcast frames are only forwarded within the particular VLAN, they are also known as broadcast domains.

The particular advantage of VLANs is the reduced network load for the nodes and network segments of other VLANs.

To identify which packet belongs to which VLAN, the frame is expanded by 4 bytes, refer to VLAN tagging (Page 2560). This expansion includes not only the VLAN ID but also priority information.

Options for the VLAN assignment

There are various options for the assignment to VLANs:

- Port-based VLAN
  Each port of a device is assigned a VLAN ID. You configure port-based VLAN in "Layer 2 > VLAN > Port-based VLAN".
- Protocol-based VLAN
  Each port of a device is assigned a protocol group.
- Subnet-based VLAN
  The IP address of the device is assigned a VLAN ID.

VLAN assignment on the device

SCALANCE M812	P1	vlan1 For access from the local network (LAN) to the device
SCALANCE M874 SCALANCE M804PB	P1 and P2	vlan1 For access from the local network (LAN) to the device

SCALANCE M876 SCALANCE M816	P1 to P4	vlan1 For access from the local network (LAN) to the device
SCALANCE M826	P1 to P4 SHDSL 1 and SHDSL 2	vlan1 For access from the local network (LAN) to the device

The VLANs are in different IP subnets. To allow these to communicate with each other, the corresponding route and firewall rule must be configured on the device.

You can change the assignment in "Layer 2 > VLAN > General (Page 2674)".

VLAN tagging

Expansion of the Ethernet frames by four bytes

For CoS (Class of Service, frame priority) and VLAN (virtual network), the IEEE 802.1Q standard defined the expansion of Ethernet frames by adding the VLAN tag.

Note

The VLAN tag increases the permitted total length of the frame from 1518 to 1522 bytes. The end nodes on the networks must be checked to find out whether they can process this length / this frame type. If this is not the case, only frames of the standard length may be sent to these nodes.

The additional 4 bytes are located in the header of the Ethernet frame between the source address and the Ethernet type / length field:

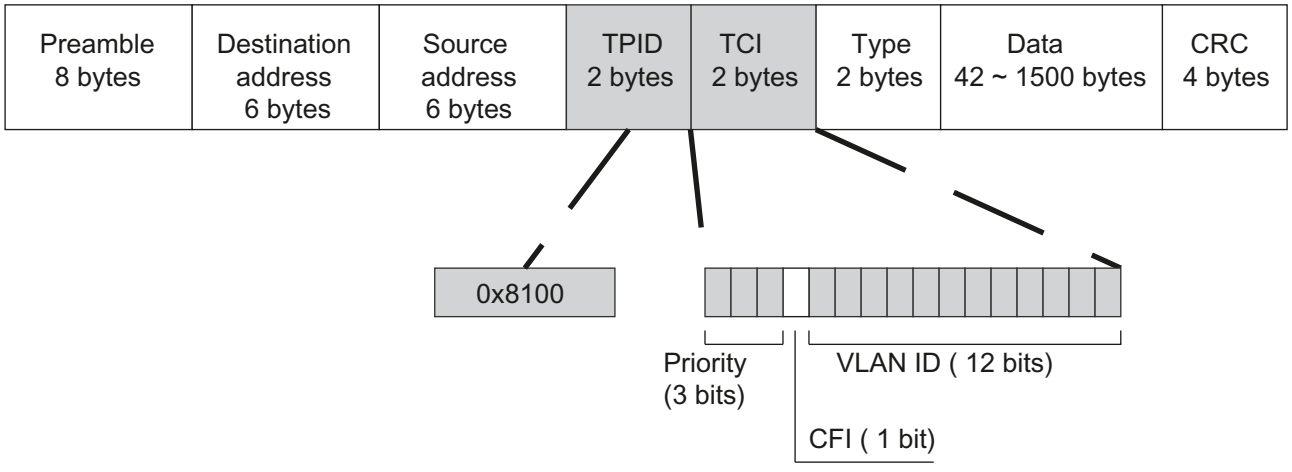

Figure 1-89 Structure of the expanded Ethernet frame

The additional bytes contain the tag protocol identifier (TPID) and the tag control information (TCI).

Tag protocol identifier (TPID)

The first 2 bytes form the Tag Protocol Identifier (TPID) and always have the value 0x8100. This value specifies that the data packet contains VLAN information or priority information.

Tag Control Information (TCI)

The 2 bytes of the Tag Control Information (TCI) contain the following information:

QoS Trust

The tagged frame has 3 bits for the priority that is also known as Class of Service (CoS), see also IEEE 802.1Q.

CoS bits	Priority	Type of the data traffic
000	0 (lowest)	Background
001	1	Best Effort
010	2	Excellent Effort
011	3	Critical Applications
100	4	Video, < 100 ms delay (latency and jitter)
101	5	Voice (language), < 10 ms delay (latency and jitter)
110	6	Internetwork Control
111	7 (highest)	Network Control

The prioritization of the data packets is possible only if there is a queue in the components in which they can buffer data packets with lower priority.

The device has multiple parallel queues in which the frames with different priorities can be processed. As default, first, the frames with the highest priority are processed. This method ensures that the frames with the highest priority are sent even if there is heavy data traffic.

Canonical Format Identifier (CFI)

The CFI is required for compatibility between Ethernet and the token Ring.
The values have the following meaning:

Value	Meaning
0	The format of the MAC address is canonical. In the canonical representation of the MAC address, the least significant bit is transferred first. Standard-setting for Ethernet switches.
1	The format of the MAC address is not canonical.

VLAN ID

In the 12-bit data field, up to 4096 VLAN IDs can be formed. The following conventions apply:

VLAN ID	Meaning
0	The frame contains only priority information (priority tagged frames) and no valid VLAN identifier.
1- 4094	Valid VLAN identifier, the frame is assigned to a VLAN and can also include priority information.
4095	Reserved

SNMP

SNMP

Introduction

With the aid of the Simple Network Management Protocol (SNMP), you monitor and control network components from a central station, for example routers or switches. SNMP controls the communication between the monitored devices and the monitoring station.

Tasks of SNMP:

- Monitoring of network components
- Remote control and remote parameter assignment of network components
- Error detection and error notification

In versions v1 and v2c, SNMP has no security mechanisms. Each user in the network can access data and also change parameter assignments using suitable software.

For the simple control of access rights without security aspects, community strings are used.

The community string is transferred along with the query. If the community string is correct, the SNMP agent responds and sends the requested data. If the community string is not correct, the SNMP agent discards the query. Define different community strings for read and write permissions. The community strings are transferred in plain text.

Standard values of the community strings:

- public
  has only read permissions
- private
  has read and write permissions

Note

Because the SNMP community strings are used for access protection, do not use the standard values "public" or "private". Change these values following the initial commissioning.

Further simple protection mechanisms at the device level:

- Allowed Host
  The IP addresses of the monitoring systems are known to the monitored system.
- Read Only
  If you assign "Read Only" to a monitored device, monitoring stations can only read out data but cannot modify it.

SNMP data packets are not encrypted and can easily be read by others.

The central station is also known as the management station. An SNMP agent is installed on the devices to be monitored with which the management station exchanges data.

The management station sends data packets of the following type:

- GET
  Request for a data record from the SNMP agent
- GETNEXT
  Calls up the next data record.
- GETBULK (available as of SNMPv2c)
  Requests multiple data records at one time, for example several rows of a table.
- SET
  Contains parameter assignment data for the relevant device.

The SNMP agent sends data packets of the following type:

- RESPONSE
  The SNMP agent returns the data requested by the manager.
- TRAP
  If a certain event occurs, the SNMP agent itself sends traps.

SNMPv1/v2c/v3 use UDP (User Datagram Protocol) and use the UDP ports 161 and 162. The data is described in a Management Information Base (MIB).

SNMPv3

Compared with the previous versions SNMPv1 and SNMPv2c, SNMPv3 introduces an extensive security concept.

SNMPv3 supports:

- Fully encrypted user authentication
- Encryption of the entire data traffic
- Access control of the MIB objects at the user/group level

With the introduction of SNMPv3 you can no longer transfer user configurations to other devices without taking special action, e.g. by loading a configuration file or replacing the C-PLUG.

According to the standard, the SNMPv3 protocol uses a unique SNMP engine ID as an internal identifier for an SNMP agent. This ID must be unique in the network. It is used to authenticate access data of SNMPv3 users and to encrypt it.

Depending on whether you have enabled or disabled the “SNMPv3 User Migration” function, the SNMP engine ID is generated differently.

Restriction when using the function

Use the "SNMPv3 User Migration" function only to transfer configured SNMPv3 users to a substitute device when replacing a device.
Do not use the function to transfer configured SNMPv3 users to multiple devices. If you load a configuration with created SNMPv3 users on several devices, these devices use the same SNMP engine ID. If you use these devices in the same network, your configuration contradicts the SNMP standard.

Compatibility with predecessor products

You can only transfer SNMPv3 users to a different device if you have created the users as migratable users. To create a migratable user the "SNMPv3 User Migration" function must be activated when you create the user.

ICMP

The acronym ICMP stands for Internet Control Message Protocol (RFC792) and is used to exchange error and information messages.

- Error message
  Informs the sender of the IP frame that when forwarding the frame an error or a parameter problem occurred.
- Information message
  Can contain information about the time measurement, the address mask, the reachability of the destination or for finding the router.

Structure of the ICMP data packet

0	4	8	12	16	20	24	28	31
ICMP packet type Type of message		Code Further details of the message		Checksum				
Data (optional)								

- **ICMP packet type**
  The most important ICMP packet types are as follows:
  - Redirect
    The router informs the host in one of its subnets that there is a better route to the destination. This ICMP packet type is dealt with in more detail in the following description.
  - Destination Unreachable
    IP frame cannot be delivered.
  - Time Exceeded
    Time limit exceeded
  - Echo-Request
    Echo request, better known as ping.
- **Code**
  The code describes the ICMP packet type in greater detail. The selection depends on the selected ICMP packet type. With "Destination Unreachable,", for example "Code 1" host cannot be reached.

You will find a full list of the ICMP packet types and codes on the website of IANA (http://www.iana.org/assignments/icmp-parameters).

ICMP packet type 5 - Redirect

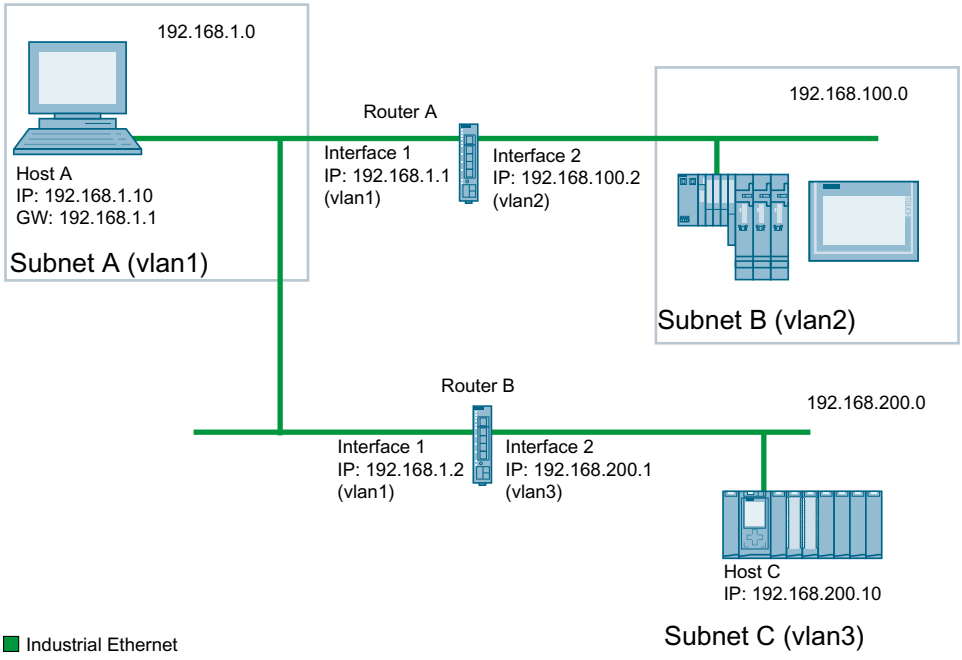

Host A wants to send an IP frame to host C. Host C is not located in the same subnet as host A. For this reason host A sends the IP frame to its default gateway. The default gateway of host A is interface 1 of router A. Router A cannot forward the IP frame because it does not know the destination network. Via its routing table, however, router A knows that subnet C is reachable via router B. Router B connects subnet A with subnet C. Router A sends a redirect message to host A. In this, router A instructs host A in future to send IP frames to host C via router B whose IP address is contained in the redirect message. The initial IP frame is sent by router A directly to router B that forwards it to Host C.

Conditions for sending redirect messages

- The IP frame is received and sent via the same interface of router A.
- The source IP address (host A) is from the same subnet as the next hop address (router B) in the routing table.
- The IP frame is not affected by a source NAT rule (masquerading, source NAT or NETMAP).
- So that router A forwards the initial IP frame to router B, a firewall rule vlanX → vlanX is required.

Security functions

User management (WBM or CLI)

Overview of user management

Access to the device is managed by configurable user settings. Set up users with a password for authentication. Assign a role with suitable rights to the users.

The authentication of users can either be performed locally by the device or by an external RADIUS server. You configure how the authentication is handled on the "Security > AAA > General" page.

Local logon

The local logging on of users by the device runs as follows:

1. The user logs on with user name and password on the device.
2. The device checks whether an entry exists for the user.
   → If an entry exists, the user is logged in with the rights of the associated role.
   → If no corresponding entry exists, the user is denied access.

Login via an external RADIUS server

RADIUS (Remote Authentication Dial-In User Service) is a protocol for authenticating and authorizing users by servers on which user data can be stored centrally.

Depending on the RADIUS authorization mode you have selected on the "Security > AAA > RADIUS Client" page, the device evaluates different information of the RADIUS server.

RADIUS authorization mode "Standard"

If you have set the authorization mode "conventional", the authentication of users via a RADIUS server runs as follows:

1. The user logs on with user name and password on the device.
2. The device sends an authentication request with the login data to the RADIUS server.
3. The RADIUS server runs a check and signals the result back to the device.
   – The RADIUS server reports a successful authentication and returns the value "Administrative User" to the device for the attribute "Service Type".
     → The user is logged in with administrator rights.
   – The RADIUS server reports a successful authentication and returns a different or even no value to the device for the attribute "Service Type".
     → The user is logged in with read rights.
   – The RADIUS server reports a failed authentication to the device:
     → The user is denied access.

RADIUS authorization mode "SiemensVSA"

Requirement

For the RADIUS authorization mode "Siemens VSA" the following needs to be set on the RADIUS server:

- Manufacturer code: 4196
- Attribute number: 1
- Attribute format: Character string (group name)

Procedure

If you have set the authorization mode "SiemensVSA", the authentication of users via a RADIUS server runs as follows:

1. The user logs on with user name and password on the device.
2. The device sends an authentication request with the login data to the RADIUS server.
3. The RADIUS server runs a check and signals the result back to the device.
   Case A: The RADIUS server reports a successful authentication and returns the group assigned to the user to the device.
   - The group is known on the device and the user is not entered in the table "External User Accounts"
     → The user is logged in with the rights of the assigned group.
   - The group is known on the device and the user is entered in the table "External User Accounts"
     → The user is assigned the role with the higher rights and logged in with these rights.
   - The group is not known on the device and the user is entered in the table "External User Accounts"
     → The user is logged in with the rights of the role linked to the user account.
   - The group is not known on the device and the user is not entered in the table "External User Accounts"
     → The user is logged in with the rights of the role "Default".

   Case B: The RADIUS server reports a successful authentication but does not return a group to the device.
   - The user is entered in the table "External User Accounts":
     → The user is logged in with the rights of the linked role "".
   - The user is not entered in the table "External User Accounts":
     → The user is logged in with the rights of the role "Default".

   Case C: The RADIUS server reports a failed authentication to the device:
   - The user is denied access.

Firewall

The security functions of the device include a stateful inspection firewall. This is a method of packet filtering or packet checking.

The IP packets are checked based on firewall rules in which the following is specified:

- The permitted protocols
- IP addresses and ports of the permitted sources
- IP addresses and ports of the permitted destinations

If an IP packet fits the specified parameters, it is allowed to pass through the firewall. The rules also specify what is done with IP packets that are not allowed to pass through the firewall.

Simple packet filter techniques require two firewall rules per connection.

- One rule for the query direction from the source to the destination.
- A second rule for the response direction from the destination to the source

Stateful inspection firewall

You only need to specify one firewall rule for the query direction from the source to the destination. The second rule is added implicitly. The packet filter recognizes when, for example, computer "A" is communicating with computer "B" and only then does it allow replies. A query by computer "B" is therefore not possible without a prior request by computer "A".

You configure the firewall in "Security > Firewall".

Note

IP packets via layer 2 (within the same VLAN)

If the IP packets from the device are sent via a switch port (layer 2), these IP packets are not checked based on firewall rules. The firewall has no effect on packets forwarded at the layer 2 level.

Communication directions

from	to	Meaning
vlan x	vlan x	Access from IP subnet vlan x and the device to IP subnet vlan x. Example: vlan1 (INT) → vlan2 (EXT) Access from the local IP subnet and the device to the external IP subnet.
	ppp0/usb	Access from the IP subnet to the mobile wireless interface of the device.
	Device	Access from the IP subnet to the device.
	SINEMA RC	Access from the IP subnet and the device to the SINEMA RC connection.
	IPsec (all) IPsec <Connection Name> OpenVPN (all) OpenVPN <Connection Name>	Access from the IP subnet to the tunnel partners that can be reached via all VPN connections (all) or via a certain VPN connection (<Connection Name>).
Device	vlan x	Access from the device to the IP subnet.
	ppp0/usb	Access from the device to the mobile wireless interface of the device.
	SINEMA RC	Access from the device to the SINEMA RC connection
	IPsec (all) IPsec <Connection Name> OpenVPN (all) OpenVPN <Connection Name>	Access from the device to the tunnel partners that can be reached via all VPN connections (all) or via a certain VPN connection (<Connection Name>).

from	to	Meaning
SINEMA RC	vlan x	Access from the SINEMA RC server to the IP subnet.
	ppp0/usb	Access from the SINEMA RC server to the mobile wireless interface of the device.
	Device	Access from the SINEMA RC server to the device.
	IPsec (all) IPsec <Connection Name> OpenVPN (all) OpenVPN <Connection Name>	Access from the SINEMA RC Server to the tunnel partners that can be reached via all VPN connections (all) or via a certain VPN connection <Connection Name>.
IPsec (all) IPsec <Connection Name> OpenVPN (all) OpenVPN <Connection Name>	vlan x	Access via tunnel partners to the IP subnet.
	ppp0/usb	Access via tunnel partners to the mobile wireless interface of the device.
	Device	Access via tunnel partners to the device.
	SINEMA RC	Access via tunnel partners to the SINEMA RC connection.
ppp0/usb	vlan x	Access from the mobile wireless interface to the IP subnet.
	Device	Access from the mobile wireless interface to the device.
	SINEMA RC	Access from the mobile wireless interface to the SINEMA RC connection.
	IPsec (all) IPsec <Connection Name> OpenVPN (all) OpenVPN <Connection Name>	Access from the mobile wireless interface to the tunnel partners that can be reached via all VPN connections (all) or via a certain VPN connection (<Connection Name>).

Firewall factory setting

Service	Access	
	from internal (vlan1) to the device	from external (ppp0/usb0) to the device
HTTP	yes	no
HTTPS	yes	no
DNS	yes	no
SNMP	yes	no
Telnet	yes	no
IPsec VPN	no	yes
SSH	yes	no
DHCP	yes	yes (only with M826 for the function DHCP server)
Ping	yes	no
System time	no	no
SMS relay (only with M87x)	yes	no
Cloud Connector (only with M804PB)	Yes	-

NAT

NAT (Network Address Translation) is a method of translating IP addresses in data packets. With this, two different networks (internal and external) can be connected together.

A distinction is made between source NAT in which the source IP address is translated and destination NAT in which the destination IP address is translated.

You will find information on NAT scenarios that are implemented with the device at the following address: (https://support.industry.siemens.com/cs/gb/en/view/109744660)

IP masquerading

IP masquerading is a simplified source NAT. With each outgoing data packet sent via this interface, the source IP address is replaced by the IP address of the interface. The adapted data packet is sent to the destination IP address. For the destination host it appears as if the queries always came from the same sender. The internal nodes cannot be reached directly from the external network. By using NAPT, the services of the internal nodes can be made reachable via the external IP address of the device.

IP masquerading can be used if the internal IP addresses cannot or should not be forwarded externally, for example because the internal network structure should remain hidden.

You configure masquerading in "Layer 3" > "NAT" > "IP Masquerading (Page 2688)".

NAPT

NAPT (Network Address and Port Translation) is a form of destination NAT and is often called port forwarding. This allows the services of the internal nodes to be reached from external that are hidden by IP masquerading or source NAT.

Incoming data packets are translated that come from the external network and are intended for an external IP address of the device (destination IP address). The destination IP address is replaced by the IP address of the internal node. In addition to address translation, port translation is also possible.

The options are available for port translation:

from	to	Response
a single port	the same port	If the ports are the same, the frames will be forwarded without port translation.
a single port	a single port	The frames are translated to the port.
a port range	a single port	The frames from the port range are translated to the same port (n: 1).
a port range	the same port range	If the port ranges are the same, the frames will be forwarded without port translation.

from	to	Response
a port range	another port range	The frames are translated to any free port from the target range. With individual connection, they are normally translated to the first port in the target range. If there are connections at the same time, the round robin method is used to translate to a free port in the target range.
a single port	a port range	The frames are translated to any free port from the target range. With individual connection, they are normally translated to the first port in the target range. If there are connections at the same time, the round robin method is used to translate to a free port in the target range.

Port forwarding can be used to allow external nodes access to certain services of the internal network e.g. FTP, HTTP.

You configure NAPT in "Layer 3" > "NAT" > "NAPT (Page 2689)".

Source NAT

As with masquerading, in source NAT the source address is translated. In addition to this, the outgoing data packets can be restricted. These include limitation to certain IP addresses or IP address ranges and limitation to certain interfaces.

Source NAT can be used if the internal IP addresses cannot or should not be forwarded externally, for example because a private address range such as 192.168.x.x is used.

You configure source NAT in "Layer 3" > "NAT" > "Source NAT (Page 2690)".

NETMAP

With NETMAP it is possible to translate complex subnets to a different subnet. In this translation, the subnet part of the IP address is changed and the host part remains. For translation with NETMAP only one rule is required. NETMAP can translate both the source IP address and the destination IP address. To perform the translation with destination NAT and source NAT, numerous rules would be necessary. NETMAP can also be applied to VPN connections.

You configure NETMAP in "Layer 3" > "NAT" > "NETMAP (Page 2692)".

NAT and firewall

The firewall and NAT router support the "Stateful Inspection" mechanism. If the IP data traffic from internal to external is enabled, internal notes can initiate a communications connection into the external network.

The reply frames from the external network can pass through the NAT router and firewall without it being necessary for their addresses to be included extra in the firewall rule and the NAT address translation. Frames that are not a reply to a query from the internal network are discarded without a matching firewall rule.

NAT translation and firewall rules

Example of NAT translations

	NAT rule						
	Type	Source Interface	Destination Interface	Source IP Subnet	Source IP translated subnet	Destination IP Subnet	Translated destination IP
①	Source	vlan1 (internal)	vlan2 (external)	192.168.1.0/24	10.100.1.0/24	10.10.10.0/24	-
	The rule applies to packets sent from vlan1 (internal) to vlan2 (external). With the packets that arrive at vlan1 there is a check to establish whether the rule applies. If the source IP address in the subnet of the sender (Source IP subnet) and the destination IP address in the subnet of the recipient (Source IP subnet), the source IP address is replaced by the suitable IP address from the "Translated source IP subnet". The subnet part of the source IP address is changed and the host part remains unchanged. A packet, for example with the source IP address 192.168.1.102 is changed to 10.100.1.102. For the devices connected to vlan2 it appears as if the packets were sent from the IP subnet 10.100.1.0/24. This allows for example overlaps of IP subnets to be resolved. The rule is only specified for the send direction. The retranslation is performed implicitly. If the rule does not apply, the packets are forwarded without translation.						
②	Destination	vlan2 (external)	vlan1 (internal)	10.10.10.0/24	-	10.100.1.0/24	192.168.1.0/24
	The rule applies to packets sent from vlan2 (external) to vlan1 (internal). With the packets that arrive at vlan2 there is a check to establish whether the rule applies. If the source IP address in the subnet of the sender (Source IP subnet) and the destination IP address in the subnet of the recipient (Source IP subnet), the source IP address is replaced by the suitable IP address from the "Translated destination IP subnet". A packet, for example with the source IP address 10.10.10.102 is changed to 192.168.1.102. The devices connected to vlan1 can communicate with the devices connected to vlan2. This assumes that the corresponding firewall rule is set. The devices connected to vlan2 must address the devices connected to vlan1 with the virtual IP address from the subnet 10.100.1.0.						

Firewall rules for the NAT rules ① and ②

Example 1:

These IP packet filter rules allow the IP data traffic for all devices for the specified direction.

NAT rule	IP packet filter rules						Description
	Action	From	To	Source (Range)	Destination (Range)	Service	
①	Accept	vlan1 (internal)	vlan2 (external)	192.168.1.0/24 (Source IP subnet)	10.10.10.0/24 (Destination IP subnet)	all	All packets sent from vlan1 (internal) to vlan2 (external) are allowed to pass. This IP packet filter rule applies to the devices connected to vlan1.
②	Accept	vlan2 (external)	vlan1 (internal)	192.168.1.0/24 (Translated destination IP)	10.100.1.0/24 (Destination IP subnet)	all	All packets sent from vlan2 (external) to vlan1 (internal) are allowed to pass.

Example 2:

These IP packet filter rules restrict the IP data traffic to a specific device.

NAT rule	IP packet filter rules						Description
	Action	From	To	Source (Range)	Destination (Range)	Service	
①	Accept	vlan1 (internal)	vlan2 (external)	192.168.1.20/32 (Source IP subnet)	10.10.10.0/24 (Destination IP subnet)	all	Only packets sent to vlan2 (external) from the IP address 192.168.1.20 are allowed to pass.
②	Accept	vlan2 (external)	vlan1 (internal)	192.168.1.20/32 (Translated destination IP subnet)	10.100.1.0/24 (Destination IP subnet)	all	Only packets sent from vlan2 (external) to the IP address 192.168.1.20 are allowed to pass.

Certificates

Certificate types

The device uses different certificates to authenticate the various nodes.

Certificate		Is used in...
CA certificate	The CA certificate is a certificate issued by a Certificate Authority from which the server, device and partner certificates are derived. To allow a certificate to be derived, the CA certificate has a private key signed by the certificate authority. The key exchange between the device and the VPN gateway of the partner takes place automatically when establishing the connection. No manual exchange of key files is necessary.	IPsec VPN
Server certificate	Server certificates are required to establish secure communication (e.g. HTTPS, VPN...) between the device and another network participant. The server certificate is an encrypted SSL certificate. The server certificate is derived from the oldest valid CA, even if this is "out of service". The crucial thing is the validity date of the CA.	SINEMA RC
Device certificate	Certificates with the private key (key file) with which the device identifies itself.	IPsec VPN
Partner certificate	Certificates with which the VPN gateway of the partner identifies itself with the device.	IPsec VPN

File types

File type	Description
*.crt	File that contains the certificate.
*.p12	In the PKCS12 certificate file, the private key is stored with the corresponding certificate and is password protected. The CA creates a certificate file (PKCS12) for both ends of a VPN connection with the file extension ".p12". This certificate file contains the public and private key of the local station, the signed certificate of the CA and the public key of the CA.
*.pem	Certificate and key as Base64-coded ASCII text.

VPN

The device supports the following VPN systems

- IPsec VPN
- OpenVPN

IPsec VPN

You configure the IPsec connections in "Security" > " IPsec VPN".

With IPsec VPN, the frames are transferred in tunnel mode. To allow the device to establish a VPN tunnel, the remote network must have a VPN gateway as the partner.

For the VPN connections, the device distinguishes two modes:

- Roadwarrior mode
  In this mode either the address of the partner is fixed or an IP range is entered from which the connections are taken. The device learns the reachable remote subnets from the partner.
- Standard mode
  In this mode the address of the partner or the remote subnet is entered permanently. The device can either establish the connection actively as a VPN client or wait passively for connection establishment by the partner.

The IPsec method

The device uses the IPsec method in the tunnel mode for the VPN tunnel. Here, the frames to be transferred are completely encrypted and provided with a new header before they are sent to the VPN gateway of the partner. The frames received by the partner are decrypted and forwarded to the recipient.

To provide security, the IPsec protocol suite uses various protocols:

- The IP Authentication Header (AH) handles the authentication and identification of the source.
- The Encapsulation Security Payload (ESP) encrypts the data.

- The Security Association (**SA**) contains the specifications negotiated between the partners, e.g. about the lifetime of the key, the encryption algorithm, the period for new authentication etc.
- Internet Key Exchange (**IKE**) is a key exchange method. The key exchange takes place in two phases:
  - Phase 1
    In this phase, no security services such as encryption, authentication and integrity checks are available yet since the required keys and the IPsec SA still need to be created. Phase 1 serves to establish a secure VPN tunnel for phase 2. To achieve this, the communications partners negotiate an ISAKMP Security Association (ISAKMP SA) that defines the required security services (algorithms, authentication methods used). The subsequent messages and phase 2 are therefore secure.
  - Phase 2
    Phase 2 serves to negotiate the required IPsec SA. Similar to phase 1, exchanging offers achieves agreement about the authentication methods, the algorithms and the encryption method to protect the IP packets with IPsec AH and IPsec ESP.
    The exchange of messages is protected by the ISAKMP SA negotiated in phase 1. Due to the ISAKMP SA negotiated in phase 1, the identity of the nodes is known and the method for the integrity check already exists.

Authentication method

- CA certificate, device and partner certificate (digital signatures)
  The use of certificates is an asymmetrical cryptographic system in which every node (device) has a pair of keys. Each node has a secret, private key and a public key of the partner. The private key allows the device to authenticate itself and to generate digital signatures.
- Pre-shared key
  The use of a pre-shared key is a symmetrical cryptographic system. Each node has only one secret key for decryption and encryption of data packets. The authentication is via a common password.

Local ID and remote ID

The local ID and the remote ID are used by IPsec to uniquely identify the partners (VPN end point) during establishment of a VPN connection.

Encryption methods

The following encryption methods are supported. The selection depends on the phase und the key exchange method (IKE)

	Phase 1		Phase 2	
	IKEv1	IKEv2	IKEv1	IKEv2
3DES	x	x	x	x
AES128 CBC	x	x	x	x
AES192 CBC	x	x	x	x
AES256 CBC	x	x	x	x

	Phase 1		Phase 2	
	IKEv1	IKEv2	IKEv1	IKEv2
AES128 CTR	-	x	x	x
AES192 CTR	-	x	x	x
AES256 CTR	-	x	x	x
AES128 CCM 16	-	x	x	x
AES192 CCM 16	-	x	x	x
AES256 CCM 16	-	x	x	x
AES128 GCM 16	-	x	x	x
AES192 GCM 16	-	x	x	x
AES256 GCM 16	-	x	x	x

x: is supported

-: is not supported

Default Ciphers

During connection establishment a preset list can be transferred to the VPN connection partners. The list contains combinations of the three algorithms (Encryption, Authentication, Key Derivation). To establish a VPN connection, the VPN connection partner must support at least one of these combinations. The combinations depend on the phase und the key exchange method IKE).

Combination			Phase 1		Phase 2	
Encryption	Authentica-tion	Key derivation	IKEv1	IKEv2	IKEv1	IKEv2
AES128	SHA1	DH Group 14	x	x	x	x
AES256	SHA512	DH Group 16	x	x	x	x
AES128 CCM 16	SHA256	DH Group 14	-	x	x	x
AES256 CCM 16	SHA512	DH Group 16	-	x	x	x
AES128	SHA1	none	-	-	x	x
AES256	SHA512	none	-	-	x	x
AES128 CCM 16	SHA256	none	-	-	x	x
AES256 CCM 16	SHA512	none	-	-	x	x

x: Combination is part of the default cipher

-: Combination is not part of the default cipher

none: For phase 2, no separate keys are exchanged. This means that Perfect Forward Secrecy (PFS) is disabled.

Requirements of the VPN partner

The VPN partner must support IPsec with the following configuration to be able to establish an IPsec connection successfully:

- Authentication with partner certificate, CA certificates or pre-shared key
- IKEv1 or IKEv2
- Support of at least one of the following DH groups: Diffie-Hellman group 1, 2, 5 and 14 - 18

- 3DES or AES encryption
- MD5, SHA1, SHA256, SHA384 or SHA512
- Tunnel mode

If the VPN partner is downstream from a NAT router, the partner must support NAT-T. Or, the NAT router must know the IPsec protocol (IPsec/VPN passthrough).

NAT traversal (NAT-T)

There may be a NAT router between the device and the VPN gateway of the remote network. Not all NAT routers allow IPsec frames to pass through. This means that it may be necessary to encapsulate the IPsec frames in UDP packets to be able to pass through the NAT router.

Dead peer detection

This is only possible when the VPN partner supports DPD. DPD checks whether the connection is still operating problem free or whether there has been an interruption on the line. Without DPD and depending on the configuration, it may be necessary to wait until the SA lifetime has expired or the connection must be reinitiated manually. To check whether the IPsec connection is still problem-free, the device itself sends DPD queries to the VPN partner station. If the VPN partner station does not reply after a certain time has elapsed, the connection to the VPN partner station will be declared invalid. You configure the settings for DPD in phase 1.

OpenVPN

With OpenVPN, virtual private networks (VPN) can be established. As an OpenVPN client, the device can establish a VPN connection to a remote network.

You configure the OpenVPN client in "Security" > " OpenVPN Client".

The VPN connection is established via virtual device drivers, the TAP and TUN device. During this, virtual network interfaces are created that act like a physical interface of the device and represent the endpoint of the VPN tunnel.

The device supports the following:

- TUN device: Routing mode
  The LAN Interface and the virtual network interface are located in different IP subnets. The virtual tunnel interface is assigned a virtual IP address from a devised subnet by the OpenVPN server. The IP packets (layer 3) are routed between the virtual tunnel interface and the LAN interface.

Authentication method

- Certificates: CA certificate and device certificate
  The use of certificates is an asymmetrical cryptographic system. Each node (device) has a secret, private key and a public key of the partner. The private key allows the device to authenticate itself and to generate digital signatures.
- User name / password
  Access is restricted by a user name and a password.

Encryption methods

The device also supports the following methods:

- BF CBC
- AES128 CBC
- AES192 CBC
- AES256 CBC
- DES EDE3

VPN connection establishment

The device supports the following options for establishing a VPN connection.

- OpenVPN: Security > OpenVPN > Connections
- IPsec VPN: Security > IPsec VPN > Connections
- SINEMA RC: System > SINEMA RC

Options	Use			Description
	OpenVPN	IP-sec VPN	SINE-MA RC [1]	
start	x	x	-	The device is "active", in other words, it attempts to establish a connection to a partner. The partner is addressed using its configured WAN IP address or the configured FQDN.
wait	-	x	-	The device is "passive", in other words, it waits for the partner to initiate the connection.
on demand	-	x	-	The device attempts to establish a connection to a partner when necessary. The receipt of requests for VPN connection establishment is also possible. For the configured local and remote subnets, an entry is created in the routing table. If a node attempts to send data packets via the VPN tunnel from one of the networks, the VPN connection is established. The settable timeout has the effect that after this time without any further data packets the VPN tunnel is terminated again.
Start on DI	x	x	x	Connection establishment is controlled via the digital input (DI).
Wait on DI	-	x	-	
Digital Input & Wake-up SMS	-	-	x	Connection establishment is controlled via the digital input (DI) or SMS.
Wake-up SMS	-	-	x	Connection establishment is controlled via SMS.
start on SMS	x	x	-	
wait on SMS	-	x	-	

Options	Use			Description
	OpenVPN	IP-sec VPN	SINE-MA RC [1]	
Auto	-	-	x	The device adopts the settings of the SINEMA RC server. You configure the settings on the SINEMA IRC server in "Remote connections > Devices". You will find further information on this topic in the operating instructions "SINEMA RC Server".
Permanent	-	-	x	The device establishes a VPN connection to the SINEMA RC Server. The VPN tunnel is established permanently

[1] KEY-PLUG SINEMA REMOTE CONNECT required

Digital input (DI)

The establishment of the VPN tunnel can also be controlled via the digital input, e.g. using a button. When the button is closed, voltage is applied to the digital input and the LED of the digital input lights up. The lit LED indicates that signal 1 (TRUE / HIGH) is applied. Signal 1 triggers an event on the device with which the establishment of the VPN tunnel is controlled. You will find information on connecting and the maximum current load in the operating instructions of the devices.

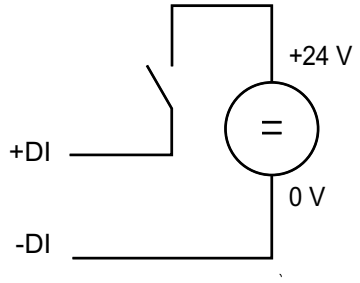

Requirement

- In "System > Events > Configuration" for the "Digital Input" event "VPN Tunnel" is activated. If this setting is not activated, the event is not passed on to the VPN connection.

Options

The device supports the following options for controlling the VPN tunnel via the digital input:

- start on DI
  If the event "Digital Input" occurs, the device becomes "active". The device tries to establish a VPN connection to a remote station (OpenVPN, IPsec, SINEMA RC). SCALANCE M804PB starts the integrated TIA Portal Cloud Connector.
- Wait on DI
  If the event "Digital Input" occurs, the device becomes "passive". The device waits until the connection establishment is initiated by the remote station (OpenVPN, IPsec).
- Digital Input & Wake-up SMS (only with M87x)
  If the event "Digital In" occurs or the device receives a wake-up SMS message, the device becomes "active". The device attempts to establish a VPN connection to the SINEMA RC server.

SMS (only with M87x):

Requirement

- In "System > SMS > SMS Command" you specify who a command SMS of the class "System" will be accepted from.
- In "System > SMS > Event SMS " the function is activated and the telephone number of a recipient is configured.
- If this setting is not activated, the event is not passed on to the VPN connection.

Options

The device supports the following options for controlling the VPN tunnel via an SMS message:

- start on SMS
  When the device receives a command SMS message, the device becomes "active". The command SMS message contains the name of the VPN connection (OpenVPN, IPsec). The device attempts to establish a connection to the partner specified in the VPN connection.
  Format of the SMS message:
  Establishment of the VPN connection:
  - OpenVPN: SYS OVPN UP <name of the OpenVPN connection>, see Command SMS message.
    If you enter a "*" for <name of the OpenVPN connection>, the OpenVPN connections are established that are set in "Operation" "start on SMS".
  - IPsec VPN: SYS IPSEC UP<name of the IPsec VPN connection>, see Command SMS message.
    If you enter a "*" for <name of the IPsec VPN connection>, the IPsec VPN connections are established that are set in "Operation" "start on SMS".

  Termination of the VPN connection:
  - OpenVPN: SYS OVPN DOWN <name of the OpenVPN connection>, see Command SMS message.
    If you enter a "*" for <name of the OpenVPN connection>, the OpenVPN connections are terminated that are set in "Operation" "start on SMS".
  - IPsec VPN: SYS IPSEC DOWN<name of the IPsec VPN connection>, see Command SMS message.
    If you enter a "*" for <name of the IPsec VPN connection>, the IPsec VPN connections are terminated that are set in "Operation" "start on SMS".
- wait on SMS
  When the device receives a command SMS message, the device is "passive". The command SMS message contains the name of the IPsec VPN connection. The device waits for the partner specified in the VPN connection to initiate the connection.
  Format of the SMS message:
  If you enter a "*" for <name of the IPsec VPN connection>, the IPsec VPN connections are established that are set in "Operation" "start on SMS".
  - IPsec VPN: SYS IPSEC UP<name of the IPsec VPN connection>, see Command SMS message.
    If you enter a "*" for <name of the IPsec VPN connection>, the IPsec VPN connections are established that are set in "Operation" "start on SMS".
  - IPsec VPN: SYS IPSEC DOWN<name of the IPsec VPN connection>, see Command SMS message.
    If you enter a "*" for <name of the IPsecVPN connection>, the IPsec VPN connections are terminated that are set in "Operation" "wait on SMS".

- Digital Input & Wake-up SMS
  The settings of the SINEMA RC server are ignored. If the event "Digital In" occurs or the device receives a wake-up SMS message. The device attempts to establish a VPN connection to the SINEMA RC Server.
  Format of the SMS message: SYS SRC UP <address of the SINEMA RC Server>, see Command SMS message.
- Wake-up SMS
  The settings of the SINEMA RC server are ignored. When the device receives a wake-up SMS message, it attempts to establish a VPN connection to the SINEMA RC Server.
  Format of the SMS message:
  – SYS SRC UP <address of the SINEMA RC Server>, see Command SMS message.

Notification options

If the status of the digital input or a VPN tunnel (IPsec, OpenVPN, SINEMA RC) changes, the device provides several options for notification on the "Events" page.

Type of notification	Digital input	VPN tunnel	Behavior if there is a status change
E-mail	x	x	The device sends an e-mail. The e-mail contains the identification of the sending device, a description of the cause of the alarm in plain language, and a time stamp. Requirement: • An SMTP server is set up. • In "System > SMTP Client" the function is activated, a recipient and the IP address of the SMTP server are configured.
Trap	x	x	The device sends an SNMP trap. Requirement: • "SNMPv1 traps" is enabled in "System > Configuration". • In "System > Configuration > Traps" a recipient is configured to which the device sends the SNMP traps.
Log table	x	x	The device writes an entry in the event log table. The content of the event log table is displayed in "Information > Log Table".
Syslog	x	x	The device writes an entry to the Syslog server. Requirement: • A Syslog server has been set up. • In "System > Syslog Client" the function is activated and the IP address of the Syslog server is configured.
Fault LED	x	-	The fault LED lights up on the device.

Type of notification	Digital input	VPN tunnel	Behavior if there is a status change
Digital input	x	x	Controls the digital output or signals the status change with the "DO" LED. A consumer can be connected to the digital output. You will find information on connecting in the operating instructions of the devices. The consumer signals a status change. 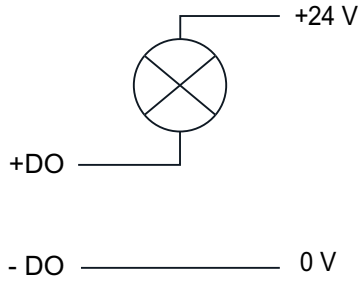 **Note** You can control the digital output directly via CLI or SNMP. In WBM and CLI, you can configure the use of the digital output in "Events". Do not control the digital output directly when you use this in the WBM and CLI.
SMS (only with M87x):	x	x	The device sends an SMS message with a standard text. Requirement: • In "System > SMS > Event SMS " the function is activated and the telephone number of a recipient is configured.
	x	-	The device sends an SMS message with a user-defined text. You configure text in "System > SMS > Event SMS". Requirement: • In "System > SMS > Event SMS " the function is activated, the telephone number of a recipient and the text are configured. • In "System > Events > Configuration", "SMS" is activated for the "Digital Input" event.
Read out the status of the MIB variable	x	-	Using the private MIB variable snMspsDigitalInputLevel, you can read out the status of the digital input. • OID of the private MIB variable snMspsDigitalInputLevel: `iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).siemens(4329).industrialComProducts(20).iComPlatforms(1).simaticNet(1).snMsps(1).snMspsCommon(1).snMspsDigitalIO(39).snMspsDigitalIOObjects(1).snMspsDigitalInputTable(2).snMspsDigitalInputEntry(1).snMspsDigitalInputLevel(6)` • values of the MIB variable – 1: Signal 0 at the digital input (DI) – 2: Signal 1 at the digital input (DI)

Showing information

Versions

This page shows the versions of the hardware and software of the device.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

Table 1 has the following columns:

- **Hardware**
  - Basic Device
    Shows the basic device
  - CELL (M87x only)
    Shows the wireless module being used.
  - xDSL (nur bei M81x)
    Shows the DSL module being used
- **Name**
  Shows the name of the device.
- **Revision**
  Shows the hardware version of the device.
- **Order ID**
  Shows the article number of the device.
- **Software**
  - Firmware
    Shows the current firmware version. If a new firmware file was downloaded and the device has not yet restarted, the firmware version of the downloaded firmware file is displayed here. After the next restart, the loaded firmware is activated and used.
  - Bootloader
    Shows the version of the boot software stored on the device.
  - Firmware_Running
    Shows the firmware version currently being used on the device.
- **Description**
  Shows the short description of the software.
- **Version**
  Shows the version number of the software version.
- **Date**
  Shows the date on which the software version was created.

I&M

Note

The page is only available if there is an online connection to the device.

This page contains information about device-specific vendor and maintenance data such as the article number, serial number, version number etc. You cannot configure anything on this page.

Description of the displayed values

The table has the following rows:

- **Manufacturer ID**
  Shows the manufacturer ID.
- Article number
  Shows the article number.
- **Serial number**
  Shows the serial number.
- **Hardware revision**
  Shows the hardware version.
- **Software version**
  Shows the software version.
- **Revision counter**
  Shows the revision counter: Counter for revisions since the initial commissioning
- **Revision Date**
  Date and time of the last revision
- **Function tag**
  Shows the function tag (plant designation) of the device. The plant designation (HID) is created during configuration of the device with HW Config of STEP 7.
- **Location tag**
  Shows the location tag of the device. The location identifier (LID) is created during configuration of the device with HW Config of STEP 7.
- **Date**
  Shows the date created during configuration of the device with HW Config of STEP 7.
- **Description**
  Shows the description created during configuration of the device with HW Config of STEP 7.

ARP table

Note

The page is only available if there is an online connection to the device.

Assignment of MAC address and IP address

With the Address Resolution Protocol (ARP), there is a unique assignment of MAC address to IPv4 address. This assignment is kept by each network node in its own separate ARP table. The page shows the ARP table of the device.

Description of the displayed values

The table has the following columns:

- Interface
  Shows the interface via which the row entry was learnt.
- MAC Address
  Shows the MAC address of the destination or source device.
- IP address
  Shows the IP address of the target device.
- Media Type
  Shows the type of connection.
  - Dynamic
    The device recognized the address data automatically.
  - Static
    The addresses were entered as static addresses.

Log tables

Event log

Logging events

Note

The page is only available if there is an online connection to the device.

The device allows you to log events that occur, some of which you can specify in " System > Events". This, for example, allows you to record when an authentication attempt failed or when the connection status of a port has changed. The content of the events log table is retained even when the device is turned off.

Description of the displayed values

- Severity Filters

  Note

  For each severity, a maximum of 400 entries in the table are possible. If the maximum number of entries is reached for a severity, the oldest entries of this severity are overwritten in the table. The table remains permanently in the memory

  You can filter the entries in the table according to severity. Select the required entries in the check boxes above the table.

  – Info
    Information

  – When this parameter is enabled, all entries of the category "Info" are displayed.

  – Warning
    Warning
    When this parameter is enabled, all entries of the category "Warning" are displayed.

  – Critical
    Critical
    When this parameter is enabled, all entries of the category "Critical" are displayed.

  The table has the following columns:

  – Restart
    Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.

  – System Up Time
    Shows the time the device has been running since the last restart when the described event occurred.

  – System Time
    Shows the date and time when the described event occurred.
    If the system time is set, the time is also displayed at which the event occurred.

  – Severity
    Shows the severity of the message.

  – Log Message
    Displays a brief description of the event that has occurred. You will find the list of possible messages in Appendix D of the configuration manual.

Security log

Note

The page is only available if there is an online connection to the device.

The page shows the events that occurred during communication via a secure VPN tunnel in the form of a table.

Description of the displayed values

- Severity Filters

  Note

  For each severity, a maximum of 400 entries in the table are possible. If the maximum number of entries is reached for a severity, the oldest entries of this severity are overwritten in the table. The table remains permanently in the memory

  You can filter the entries in the table according to severity. To display all the entries, enable or disable all parameters.

  – 6 - Info
    Informative
    When this parameter is enabled, all entries of the category "Info" are displayed.
  – 4 - Warning
    Warning
    When this parameter is enabled, all entries of the category "Warning" are displayed.
  – 2 - Critical
    Critical
    When this parameter is enabled, all entries of the category "Critical" are displayed.

The table has the following columns:

- Restart
  Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.
- System Up Time
  Shows the time the device has been running since the last restart when the described event occurred.
- System Time
  If the system time is set, the date and time are also displayed at which the event occurred. If no system time is set, the box displays "Date/time not set".
- Severity
  Sorts the entry into the categories above.
- Log Message
  Displays a brief description of the event that has occurred.

Firewall log

Note

The page is only available if there is an online connection to the device.

The firewall log logs the events that occurred on the firewall. When you create firewall rules, you can specify the event severity with which they are logged.

Description of the displayed values

- Severity Filters

  Note

  For each severity, a maximum of 400 entries in the table are possible. If the maximum number of entries is reached for a severity, the oldest entries of this severity are overwritten in the table. The table remains permanently in the memory

  You can filter the entries in the table according to severity. To display all the entries, enable or disable all parameters.

  – 6 - Info
    Informative
    When this parameter is enabled, all entries of the category "Info" are displayed.
  – 4 - Warning
    Warning
    When this parameter is enabled, all entries of the category "Warning" are displayed.
  – 2 - Critical
    Critical
    When this parameter is enabled, all entries of the category "Critical" are displayed.

The table has the following columns:

- Restart
  Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.
- System Up Time
  Shows the time the device has been running since the last restart when the described event occurred.
- System Time
  If the system time is set, the date and time are also displayed at which the event occurred. If no system time is set, the box displays "Date/time not set".
- Severity
  Sorts the entry into the categories above.
- Log Message
  Displays a brief description of the event that has occurred.

Fault

Error status

if an error occurs, it is shown on this page. On the device errors are indicated by the fault LED lighting up.

Internal errors of the device and the errors that you configure on the following pages are indicated:

- "System > Events"
- "System" > Fault Monitoring"

The calculation of the time of an error always begins after the last system start.

If there are no errors present, the fault LED switches off.

Note

The page is only available if there is an online connection to the device.

Displayed values

- **No. of Signaled Faults**
  Indicates how often the fault LED lit up and not how many faults occurred.
- **"Reset counters" button**
  The number is reset with this button. The counter is reset when there is a restart.

The table contains the following columns:

- **Fault Time**
  Shows the time the device has been running since the last restart when the described fault occurred.
- **Fault Description**
  Displays a brief description of the error that has occurred.
- **Clear Fault State**
  If the "Clear Fault State" button is enabled, you can delete the error, e.g. error of the event "Cold/Warm Start".

DHCP Server

This page shows whether IPv4 addresses were assigned to the devices by the DHCP server.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

- **IP Address**
  Shows the IPv4 address assigned to the DHCP client.
- **Pool ID**
  Shows the number of the IPv4 DHCP pool.

- **Identification method**
  Shows the method according to which the DHCP client is identified.
  - Remote ID
    Shows the remote ID of the DHCP client.
  - Circuit ID
    Shows the circuit ID of the DHCP client.
  - DUID
    Shows the DUID of the DHCP client.
- **Identification value**
  Shows the MAC address or he client ID of the DHCP client.
- **Assignment method**
  Shows whether the IPv4 address was assigned statically or dynamically. You configure the static entries in "System > DHCP > Static Leases".
- **Assignment status**
  Shows the status of the assignment.
  - Assigned
    The assignment is used.
  - Not used
    The assignment is not used.
  - Probing
    The assignment is being checked.
  - Unknown
    The status of the assignment is unknown.
- **Expire Time**
  Shows how long the assigned IPv4 address is still valid. When half the period of validity has elapsed, the DHCP client can extend the period of the assigned IPv4 address. When the entire time has elapsed, the DHCP client needs to request a new IPv4 address.

SNMP

Note

The page is only available if there is an online connection to the device.

This page displays the created SNMPv3 groups. You configure the SNMPv3 groups in "System" > SNMP"..

Description of the displayed values

The table has the following columns:

- **Group Name**
  Shows the group name.
- **User Name**
  Shows the user that is assigned to the group.

LLDP

Note

The page is only available if there is an online connection to the device.

Status of the neighborhood table

This page shows the current content of the neighborhood table. This table stores the information that the LLDP agent has received from connected devices. You set the interfaces via which the LLDP agent receives or sends information in the following section: "Layer 2 > LLDP".

Description of the displayed values

The table contains the following columns:

- **System name**
  System name of the connected device.
- **Device ID**
  Device ID of the connected device. The device ID corresponds to the device name assigned via PST (STEP 7). If no device name is assigned, the MAC address of the device is displayed.
- **Local interface**
  Port at which the device received the information
- **Hold Time**
  An entry remains stored on the device for the time specified here. If the device does not receive any new information from the connected device during this time, the entry is deleted.
- **Capability**
  Shows the capabilities of the connected device:
  - Router
  - Bridge
  - Telephone
  - DOCSIS Cable Device
  - WLAN Access Point
  - Repeater
  - Station
  - Other
- **Port ID**
  Port of the device with which the device is connected.

Routing

Introduction

Note

The page is only available if there is an online connection to the device.

This page shows the routes currently being used.

Description of the displayed values

The table has the following columns:

- **Destination Network**
  Shows the destination address of this route.
- **Subnet Mask**
  Shows the subnet mask of this route.
- **Gateway**
  Shows the gateway for this route.
- **Interface**
  Shows the interface for this route.
- **Metric**
  Shows the metric of the route. The higher value, the longer packets require to their destination.
- **Routing Protocol**
  Shows the routing protocol from which the entry in the routing table originates. The following entries are possible:
  - Connected: Connected routes
  - Static: Static routes

Mobile radio

Mobile radio overview

Note

The page is only available if there is an online connection to the device.

The page shows an overview of the current operating status of the device.

Description of the displayed values

The page contains the following:

- **IMEI**
  Shows the IMEI number of the wireless module being used.
  The IMEI (= International Mobile Equipment Identity) is assigned uniquely worldwide.
- **SIM Status**
  Shows the status of the SIM card.
  - present
    SIM card exists
  - missing
    SIM card missing
- **PIN Status**
  Shows the status of the PIN.
  - PIN required
    PIN still needs to be entered.
  - No PIN required
    PIN is not needed.
  - PUK required
    SIM card blocked. With the PUK, the SIM card can be unblocked again.
  - wrong PIN
    The PIN is wrong.
  - PIN valid
    The PIN is valid.
- **IMSI**
  Shows the subscriber ID that is stored on the SIM card being used.
- **Phone Number**
  Shows the phone number of the SIM card.
  If the phone number does not exist or cannot be read out, "no number" is dispayed.
- **Connection Status**
  Shows whether a wireless connection exists, and, if it does, which one:
  - UMTS: IP connection via UMTS or HSPA
  - GPRS: IP connection via EGPRS, GPRS or LTE
- **Packet Switch Status**
  Shows the status of the packet switching.
- **Cell ID**
  Shows the ID of the wireless cell in which the device is logged in.
- **LAC (Location Area Code)**Shows the ID for the current location of the M87x within the mobile wireless network.

- **Signal Strength**
  Shows the signal strength.

dBm	
< -109	No connection
-109 ... -89	Bad signal strength
-89 …-73	Medium signal strength
> -73	Good signal strength

- **Mobile Chip Temperature**
  Shows the temperature of the plugged-in chip.
- **Provider**
  Shows the mobile wireless provider.
- **APN**
  Shows the APN (= Access Point Name) of the wireless link that is being used.
- **External IP address**
  Shows the WAN IP address at which the device can be reached in the mobile wireless network. The WAN IP address is assigned to the device by the service of the mobile wireless provider.
- **DNS Server(s)**
  Shows the DNS server or servers used by the device.

Signal recorder

Note

The page is only available if there is an online connection to the device.

The signal recorder shows the signal strength to the wireless cell in which the device is booked in.

Description of the displayed values

The graphic contains the following elements:

- Scroll bar
  With the scroll bar, you can look through the entire measurement. To do this you can use the "<<" and ">>" buttons or the arrow keys on the keyboard.
- Bar (left)
  In the bar on the left-hand side the signal strength is displayed in real time according to the color scheme.
- Color scheme
  Green = Good, Yellow = Medium, Orange = Weak, Red = Very weak

- x axis
  The x axis shows the course of the measurement in seconds.
- Measurement data
  The measurement data shows the value of the signal strength to according to the color scheme shown.

Displayed Samples

Select how many measured values will be shown in the graphic.

SHDSL

Note

The page is only available if there is an online connection to the device.

The page shows the status of the SHSDL interface.

Description of the displayed values

This table contains the following columns:

- **Interface**
  Shows the SHDSL interfaces of the device.
- **Negotiation**
  The following values are possible:
  - down not ready (only with the CO role)
    There is no electrical connection plugged into the communications partner. The driver is initialized and is ready to negotiate the connection parameters.
  - down ready (only with the CPE role)
    There is no electrical connection plugged into the communications partner. The driver is initialized and is ready to negotiate the connection parameters.
  - initializing
    The connection parameters are negotiated.
  - up data mode
    The connection parameters have been negotiated successfully. The data transfer is now possible.
  - stop down ready
    The negotiation of connection parameters was not successful or the connection was interrupted. Renegotiation of the parameters is pending.
  - unknown
    Driver initialization not yet completed. This status occurs following device startup.
- **Data rate [kbps]**
  Shows the usable data transmission rate.
- **Uptime**
  Shows the time since the last negotiation of the connection parameters or the last connection interruption.

- **SNR [dB]**
  Shows the signal-to-noise ratio: The higher the value, the better the signal.
- **CRC Anomaly**
  Cyclic Redundancy Check
  Shows how many SHDSL network packets were discovered with a bad checksum (CRC).
- **LOSW Defect**
  Loss of Synchronization Word
  Shows how often the synchronization was lost on the SHDSL connection.
- **Negotiation Count**
  Shows how often the connection parameters were renegotiated.

DSL

DSL Overview

Note

The page is only available if there is an online connection to the device.

An ADSL connection consists of two endpoints known as ADSL Transceiver Units (ATU). The ADSL Transceiver Unit Remote (ATU-R) is this device and the ADSL Transceiver Unit Central (ATU-C) is the device in the central office.

The page shows the current operating status of the device and contains information on the ATU-C.

Description of the displayed values

The page contains the following:

- **Modem Status**
  Shows the status of the modem.
  For example when initializing and when establishing the ADSL connection, the following statuses are displayed:
  - showtime-sync
    ADSL connection is established. Data traffic possible.
  - handshake
    ADSL connection is checked and established.
  - full-init
    Modem is initialized.
  - silent
    No ADSL connection

- **Latency Type**
  Shows which method is used.
  - fast
    Fast Path: Short reply time. Less protection against interference.
  - slow
    Protection against interference by interleaving. Interleaving means that the sequence of bits is changed and this extends the reply time.
  - unkown
    Method unknown
- **External IP Address**
  Shows the external IP address at which the device can be reached. The IP address is assigned to the device by the service of the DSL provider.
- **DNS Server(s)**
  Shows the DNS server or servers used by the device

DSL Data Rate

Note

The page is only available if there is an online connection to the device.

The page contains an overview of the data transfer rates for the receive direction (downstream) and the send direction (upstream).

Description of the displayed values

The page contains the following:

- **Downstream Data Rate (kbps)**
  Shows the usable data transfer rate for downstream.
- **Downstream ATTNDR (kbps)**
  ATTNDR (Attainable data rate).
  Shows the maximum possible data transfer rate for downstream.
- **Upstream Data Rate (kbps)**
  Shows the usable data transfer rate for upstream.
- **Upstream ATTNDR (kbps)**
  ATTNDR (Attainable data rate).
  Shows the maximum possible data transfer rate for upstream.

DSL Streams

Note

The page is only available if there is an online connection to the device.

The page shows the parameters that influence the data transfer rate.

Description of the displayed values

For the receive direction (downstream), the following values are displayed:

- Downstream Interleaver Depth
  Shows the depth with which the data is interleaved. The higher the interleaving depth, the longer the reply time.
- Downstream Line Attenuation (dB)
  Shows the line attenuation. The line attenuation depends on the cable diameter and the transmission frequency.
- Downstream Signal Attenuation (dB)
  Shows the signal attenuation. The lower the value, the better the signal.
- Downstream Signal-to-Noise Ratio Margin (dB)
  Shows the signal-to-noise ratio: The higher the value, the better the signal.
- Downstream Actual Aggregate Transmit Power [dB]
  Shows the current transmit power that is achieved in aggregated operation.

For the send direction (upstream), the following values are displayed:

- Upstream Interleaver Depth
  Shows the depth with which the data is interleaved. The higher the interleaving depth, the longer the reply time.
- Upstream Line Attenuation (dB)
  Shows the line attenuation. The line attenuation depends on the cable diameter and the transmission frequency.
- Upstream Signal Attenuation (dB)
  Shows the signal attenuation. The lower the value, the better the signal.
- Upstream Signal-to-Noise Ratio Margin [dB]
  Shows the signal-to-noise ratio: The higher the value, the better the signal.
- Upstream Actual Aggregate Transmit Power [dB]
  Shows the current transmit power that is achieved in aggregated operation.

IPsec VPN

The page shows the status of the activated VPN connections.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

This table contains the following columns:

- Name
  Shows the name of the VPN connection.
- Local Host
  Shows the IP address of the device.

- Local DN
  Shows the Distinguished Name (DN) of the device that was signaled to the remote station during connection establishment. The entry is adopted from the "Local ID" box, the device certificate or the IP address of the device.
- Local Subnet
  Shows the local subnet.
- Remote Host
  Shows the IP address or the host name of the remote device.
- Remote DN
  Shows the Distinguished Name (DN) signaled by the remote device during connection establishment.
- IRemote Subnet
  Shows the remote subnet.
- Rekey Time
  Shows when the validity of the key expires.
- Status
  Shows the status of the VPN connection.

SINEMA RC

Note

The page is only available if there is an online connection to the device.

Shows information on SINEMA RC Server.

Note

For SCALANCE S615 this function can only be used with a KEY-PLUG.

Description of the displayed values

- Status
  Shows the status of the connection to SINEMA RC Server.
- Device Name
  If configured, the name of the device is displayed.
- Device Location
  If configured, the location of the device is displayed.
- GSM Number
  If configured, the phone number of the device is displayed.
- Vendor
  If configured, the entry is displayed.
- Comment
  If configured, the comment is displayed.

- Type of Connection (Server)
  Shows which type of connection is set on the SINEMA RC Server.
- Type of Connection (device)
  Shows which type of connection is set on the device.
- Fingerprint
  Shows the fingerprint of the server certificate. Is only displayed when the fingerprint is used for verification.
- Remote Address
  Shows the IP address of the SINEMA RC Server.
- Connected Local Subnet (s)
  Shows the IP addresses of the local subnets. Is only displayed when the option "Connected local subnets" is enabled on the SINEMA RC Server. You will find further information on this in the Operating Instructions of the SINEMA RC Server.
- Connected Local Host (s)
  Shows the destination IP address of the hosts that can be reached.
- Tunnel Interface Address
  Shows the IP address of the virtual tunnel interface.
- Connected Remote Subnet (s)
  Shows the subnets of the SINEMA RC Server that are reachable for the device. Which subnets are reachable for the device depends on the communications relations on the SINEMA RC Server. You will find further information on this in the Operating Instructions of the SINEMA RC Server.

OpenVPN Client

The page shows the status of the activated OpenVPN connections.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

This table contains the following columns:

- Name
  Shows the name of the OpenVPN connection.
- Remote Server
  Shows the IP address or the hostname of the OpenVPN server.
- Tunnel Interface IP
  Shows the IP address of the virtual tunnel interface.
- Exported Subnets
  Shows the IP address of the local subnets.

- **Routed Subnets**
  Shows the subnets of the OpenVPN server.
- **Status**
  Shows the status of the OpenVPN connection.

Redundancy

Overview

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The page contains the following boxes:

- **Bridge Priority / Root Priority**
  Which device becomes the root bridge is decided by the bridge priority. The bridge with the highest priority (in other words, with the lowest value for this parameter) becomes the root bridge. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames.
- **Bridge Address / Root Address**
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Port**
  Shows the port via which the switch communicates with the root bridge.
- **Root Cost**
  The path costs from this device to the root bridge.
- **Topology Changes / Last Topology Change**
  The entry for the device shows the number of reconfiguration actions due to the spanning tree mechanism since the last startup. For the root bridge, the time since the last reconfiguration is displayed as follows:
  - Seconds: Supplement "sec" after the number
  - Minutes: Supplement "min" after the number
  - Hours: Supplement "hr" after the number
- **Bridge hello time [s] / Root hello time [s]**
  Each bridge sends configuration frames (BPDUs) regularly. The interval between two such frames is the Hello time.

- **Bridge Forward Delay / Root Forward Delay**
  New configuration information is not used immediately by a bridge but only after the forwarding delay specified in the parameter. This ensures that operation is only started with the new topology after all the bridges have the required information. The
- **Bridge Max Age / Root Max Age**
  When the max age timer elapses the received BPDU is discarded to be accepted as valid by the switch. The default value is 20s.
- **Bridge Max Hop Count**
  This parameter specifies how many STP nodes a BPDU may pass through. If an MSTP BPDU is received and has a hop count that exceeds the value configured here, it is discarded.
- **Root Hop Count**
  The number of nodes that need to be run through on the way to the root bridge.

Spanning Tree

Note

The page is only available if there is an online connection to the device.

Introduction

The page shows the current information about the spanning tree and the settings of the root bridge.

Description of the displayed values

The following fields are displayed:

- **Spanning Tree Mode**
  Shows the set mode. You specify the mode in "Layer 2 > Configuration" and in "Layer 2 > Spanning Tree > General".
  The following values are possible:
  - '-'
  - RSTP
- **Bridge Priority / Root Priority**
  Which device becomes the root bridge is decided by the bridge priority. The bridge with the highest priority (in other words, with the lowest value for this parameter) becomes the root bridge. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames. The value for the bridge priority is a whole multiple of 4096 with a range of values from 0 to 32768.

- **Bridge Address / Root Address**
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root switch.
- Root Cost
  Shows the path costs from the device to the root bridge.
- Bridge Status
  Shows the status of the bridge, e.g. whether or not the device is the root bridge.

The table has the following columns:

- **Port**
  Shows the interfaces via which the device communicates.
- **Role**
  Shows the status of the port. The following values are possible:
  - Disabled
    The port was removed manually from the spanning tree and will no longer be taken into account by the spanning tree.
  - Designated
    The ports leading away from the root bridge.
  - Alternate
    The port with an alternative route to a network segment
  - Backup
    If a switch has several ports to the same network segment, the "poorer" Port becomes the backup port.
  - Root
    The port that provides the best route to the root bridge.
  - Master
    This port points to a root bridge located outside the MST region.
- **Status**
  Shows the current status of the interface. The values are only displayed. The parameter depends on the configured protocol.
  - Discarding
    The port receives BPDU frames. Other incoming or outgoing frames are discarded.
  - Listening
    The port receives and sends BPDU frames. The port is involved in the spanning tree algorithm. Other outgoing and incoming frames are discarded.
  - Learning
    The port actively learns the topology; in other words, the node addresses. Other outgoing and incoming frames are discarded.
  - Forwarding
    Following the reconfiguration time, the port is active in the network. The port receives and sends data frames.
- **Oper. Version**
  Shows the compatibility mode of Spanning Tree used by the port.

- **Priority**
  If the path calculated by the spanning tree is possible over several ports of a device, the port with the highest priority (in other words the lowest value for this parameter) is selected. A value between 0 and 240 can be entered for the priority in steps of 16. If you enter a value that cannot be divided by 16, the value is automatically adapted. The default is 128.
- **Path Cost**
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected. If several ports of a device have the same value, the port with the lowest port number is selected.
  If the value in the "Cost Calc" field is "0", the automatically calculated value is displayed. Otherwise, the value of the "Cost Calc" field is displayed.
  The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000
- **Edge Type**
  Shows the type of the connection. The following values are possible:
  - Edge Port
    There is an end device at this port.
  - No Edge Port
    There is a spanning tree or rapid spanning tree device at this port.
- **P.t.P. Type**
  Shows the type of point-to-point link. The following values are possible:
  - P.t.P.
    With half duplex, a point-to-point link is assumed.
  - Shared Media
    With a full duplex connection, a point-to-point link is not assumed.

VRRPv3 statistics

Introduction

This page shows the statistics of the VRRPv3 protocol and all configured virtual routers.

Note

The page is only available if there is an online connection to the device.

Description of the displayed values

The following fields are displayed:

- VRID Errors
  Shows how many VRRPv3 packets containing an unsupported VRID were received.
- Version Errors
  Shows how many VRRPv3 packets containing an invalid version number were received.
- Checksum Errors
  Shows how many VRRPv3 packets containing an invalid checksum were received.

The table has the following columns:

- Interfaces
  Interface to which the settings relate.
- VRID
  Shows the ID of the virtual router. Valid values are 1 ... 255.
- Type
  Shows the version of the IP protocol.
- Become Master
  Shows how often this virtual router changed to the "Master" status.
- Advertisements Received
  Shows how many VRRPv3 packets were received.
- Advertisement Interval Errors
  Shows how many bad VRRPv3 packets were received whose interval does not match the value set locally.
- IP TTL Errors
  Shows how many bad VRRPv3 packets were received whose TTL (Time to live) value in the IP header is incorrect.
- Prio 0 received
  Shows how many VRRPv3 packets with priority 0 were received. VRRPv3 packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- Prio 0 sent
  Shows how many VRRPv3 packets with priority 0 were sent. Packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- Invalid Type
  Shows how many bad VRRPv3 packets were received whose value in the "Type" field of the IP header is invalid.
- Address List Errors
  Shows how many bad VRRPv3 packets were received whose address list does not match the locally configured list.
- Packet Length Errors
  Shows how many bad VRRPv3 packets were received whose length is not correct.

Security

Overview

This page shows the security settings and the local and external user accounts.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the rights of the logged-on user.

Description of the displayed values

Services

The "Services" list shows the security settings.

- **SSH Server**
  You configure the setting in "System > Configuration".
  - Enabled: Encrypted access to the CLI.
  - Disabled: No encrypted access to the CLI.
- **Web Server**
  You configure the setting in "System > Configuration".
  - HTTP/HTTPS: Access to the WBM is possible with HTTP and HTTPS.
  - HTTPS: Access to the WBM is now only possible with HTTPS.
- **SNMP**
  You can configure setting in "System > SNMP > General".
  - "-" (SNMP disabled)
    Access to device parameters via SNMP is not possible.
  - SNMPv1/v2c/v3
    Access to device parameters is possible with SNMP versions 1, 2c or 3.
  - SNMPv3
    Access to device parameters is possible only with SNMP version 3.
- **Management ACL**
  You configure the setting in "Security > Management ACL".
  - Disabled: No access restriction
    The access control is disabled.
  - Enabled: No access restriction
    The access control is enabled but no access rules have been defined.
  - Enabled: Restricted access only
    The access control is enabled and access rules have been defined.

- **Login Authentication**
  You configure the setting in "Security > AAA > General".
  - Local
    The authentication must be made locally on the device.
  - RADIUS
    The authentication must be handled via a RADIUS server.
  - Local and RADIUS
    The authentication is possible both with the users that exist on the device (user name and password) and via a RADIUS server.
    The user is first searched for in the local database. If the user does not exist there, a RADIUS query is sent.
  - RADIUS and fallback local
    The authentication must be handled via a RADIUS server.
    A local authentication is performed only when the RADIUS server cannot be reached in the network.
- **Password Policy**
  Shows which password policy is currently being used.

Local and external user accounts

You configure local user accounts and roles in "Security > User Accounts"

When you create a local user account an external user account is generated automatically. Local user accounts involve users each with a password for logging in on the device.

In the table "External User Accounts" a user is linked to a role, for example the user "Observer" with the role "user" In this case, the user is defined on a RADIUS server. The roll is defined locally on the device. When a RADIUS server authenticates a user, the corresponding group however is unknown or does not exist, the device checks whether or not there is an entry for the user in the table "External User Accounts". If an entry exists, the user is logged in with the rights of the associated role. If the corresponding group is known on the device, both tables are evaluated. The user is assigned the role with the higher rights.

Note

The table "External User Accounts" is only evaluated if you have set "Vendor Specific" in the RADIUS Authorization Mode".

With CLI you can access external user accounts.

The table "Local User Accounts" has the following columns:

- **User Account**
  Shows the name of the local user.
- **Role**
  Shows the role of the user. You can obtain more information on the function rights of the role in "Information > Security > Roles".

Supported Function Rights

The page shows the function rights available locally on the device.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the role of the logged-on user.

Description of the displayed values

- **Function Right**
  Shows the number of the function right. Different rights relating to the device parameters are assigned to the numbers.
- **Description**
  Shows the description of the function right.

Groups

This page shows which group is linked to which role. The group is defined on a RADIUS server. The roll is defined locally on the device.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the role of the logged-on user.

Description of the displayed values

The table has the following columns:

- **Group**
  Shows the name of the group. The name matches the group on the RADIUS server.
- **Role**
  Shows the role. Users who are authenticated with the linked group on the RADIUS server receive the rights of this role locally on the device.
- **Description**
  Shows a a description for the link.

Roles

The page shows the roles valid locally on the device.

Note

The page is only available if there is an online connection to the device.

The values displayed depend on the role of the logged-on user.

Description of the displayed values

This table contains the following columns:

- **Role**
  Shows the role.
- **Function Right**
  Shows the function rights of the role:
  - 1
    Users with this role can read device parameters but cannot change them.
  - 15
    Users with this role can both read and change device parameters.
  - 0
    This is a role that the device assigns internally when the user could not be authenticated. The user is denied access to the device.
- **Description**
  Shows a description of the role.

Configuring system functions

Configuration

System configuration

The page contains the configuration overview of the access options of the device.

Specify the services that access the device. With some services, there are further configuration pages on which more detailed settings can be made.

Description

The page contains the configuration overview of the access options of the device

- **Telnet Server**
  Enable or disable the Telnet server service for unencrypted access to the CLI.
- **SSH Server**
  Enable or disable the SSH server service for encrypted access to the CLI.

- **HTTP Services**
  Specify how the WBM is accessed:
  - HTTPS only
    Access to the WBM is only possible with HTTPS.
  - HTTP/HTTPS
    Access to the WBM is only possible with HTTP and HTTPS.
  - Divert HTTP to HTTPS
    Access via HTTP is automatically diverted to HTTPS.
- **SMTP Client**
  Enable or disable the SMTP client. You can configure other settings in "System > SMTP Client".
- **Syslog Client**
  Enable or disable the Syslog client. You can configure other settings in "System > Syslog Client".
- **DCP Server**
  Specify whether or not the device can be accessed with DCP (Discovery and Configuration Protocol):
  - "-" (disabled)
    DCP is disabled. Device parameters can neither be read nor modified.
  - Read/Write
    With DCP, device parameters can be both read and modified.
  - Read Only
    With DCP, device parameters can be read but cannot be modified.
- **Time**
  Select the required setting. The following settings are possible:
  - Manual
    The system time is set manually. You can configure other settings in "System > System Time > Manual Setting".
  - SIMATIC Time
    The system time is set using a SIMATIC time transmitter. You can configure other settings in "System > System Time > SIMATIC Time Client".
  - SNTP Client
    The system time is set via an SNTP server. You can configure other settings in "System > System Time > SNTP Client".
  - NTP Client
    The system time is set via an NTP server. You can configure other settings in "System > System Time > NTP Client".

- **SNMP**
  Select the required protocol. The following settings are possible:
  - "-" (SNMP disabled)
    Access to device parameters via SNMP is not possible.
  - SNMPv1/v2c/v3
    Access to device parameters is possible with SNMP versions 1, 2c or 3. You can configure other settings in "System > SNMP > General".
  - SNMPv3
    Access to device parameters is possible only with SNMP version 3. You can configure other settings in "System > SNMP > General".
- **SNMPv1/v2 Read Only**
  Enable or disable write access to SNMP variables with SNMPv1/v2c.
- **DHCP Client**
  Enable or disable the DHCP client. You can configure other settings in "System > DHCP".
- **SNMPv1 Traps**
  Enable or disable the sending of SNMPv1 traps (alarm frames). You can configure other settings in "System > SNMP > Traps".
- **Configuration Mode**
  Select the required mode. The following modes are possible:
  - Automatic Save
    Automatic backup mode. Approximately 1 minute after the last parameter change or before you restart the device, the configuration is automatically saved.
    In addition to this, the following message appears in the display area "Changes will be saved automatically in x seconds. Click 'Write Startup Config' to save the changes immediately."

    Note

    Interrupting the save

    Saving starts only after the timer in the message has elapsed. How long saving takes depends on the device.

    During the save, the message "Saving configuration data in progress. Please do not switch off the device" is displayed.

    - Do not switch off the device immediately after the timer has elapsed.
  - Trial
    Trial mode. In Trial mode, although changes are adopted, they are not saved in the configuration file (startup configuration).
    To save changes in the configuration file, use the "Write startup config" button. The display area also shows the message "Trial Mode Active – Press the "Write Startup Config" button to make your settings persistent" as soon as there are unsaved modifications. This message can be seen on every WBM page until the changes made have either been saved or the device has been restarted.

General

Device

This page contains the general device information.

Description

The page contains the following boxes:

- **Current System Time**(Only available online)
  Shows the current system time. The system time is either set by the user or by a time-of-day frame: either SINEC H1 time-of-day frame, NTP or SNTP. (readonly)
- **System Up Time**(only available online)
  Shows the operating time of the device since the last restart. (readonly)
- **Device Type**(only available online)
  Shows the type designation of the device. (readonly)
- **System Name**
  You can enter the name of the device. The entered name is displayed in the selection area. A maximum of 255 characters are possible. The system name is also displayed in the CLI input prompt. The number of characters in the CLI input prompt is limited. The system name is truncated after 16 characters.
- **System Contact**
  You can enter the name of a contact person responsible for managing the device. A maximum of 255 characters are possible.
- **Location**
  You can enter the location of the device. The entered installation location is displayed in the selection area. A maximum of 255 characters are possible.

Note

Permitted characters

The following printable ASCII characters (0x20 to 0x7e) are permitted in the input boxes **"System Name"**, **"System Contact"** and **"System Location"**:

- 0123456789
- A...Z a...z
- !"#$%&'()*+,-./:;<=>?@ [\]_{|}~^`

Coordinates

On this page, you can enter information on the geographic coordinates (latitude, longitude and the height above the ellipsoid according to WGS84).

Getting the coordinates

Use suitable maps for obtaining the geographic coordinates of the device.

The geographic coordinates can also be obtained using a GPS receiver. The geographic coordinates of these devices are normally displayed directly and only need to be entered in the input boxes of this page.

Description

The page contains the following input boxes with a maximum length of 32 characters.

- **"Latitude" input box**
  Enter the north or south latitude for the location of the device.
  E. G. +49° 1´ 31.67" means that the device is located at 49 degrees, 1 arc minute and 31.67 arc seconds north latitude.
  A south latitude is shown by a preceding minus character.
  You can also append the letters N (north latitude) or S (south latitude) to the numeric information (49° 1´ 31.67" N).
- **"Longitude" input box**
  Enter the eastern or western longitude for the location of the device.
  E. G. +8° 20´ 58.73" means that the device is located at 8 degrees, 20 arc minutes and 58.73 arc seconds eastern longitude.
  A western longitude is indicated by a preceding minus sign.
  You can also add the letter E (eastern longitude) or W (western longitude) to the numeric information (8° 20´ 58.73" E).
- **"Height" input box**
  Enter the value of the height above sea level in meters.
  For example, 158 m means that the device is located at a height of 158 m above sea level. Heights below sea level (for example the Dead Sea) are indicated by a preceding minus sign.

Restart

On this page, there is a button with which you can restart the device and various options for resetting to the device defaults.

Note

The page is only available if there is an online connection to the device.

Note

Note the following points about restarting a device:

- You can only restart the device with administrator privileges.
- A device should only be restarted with the buttons of this page or with the appropriate CLI commands and not by a power cycle on the device.
- Any modifications you have made only become active on the device after clicking the "Set Values" button on the relevant page. If the device is in "Trial Mode", configuration modifications must be saved manually before a restart. In "Automatic Save" mode, the last changes are saved automatically before a restart.

Description

The following options are available for restarting:

- **Restart**
  Click this button to restart the system. You must confirm the restart in a dialog box. During a restart, the device is reinitialized, the internal firmware is reloaded, and the device runs a self-test. The learned entries in the address table are deleted. You can leave the browser window open while the device restarts. You then need to log in again.
- **Restore Memory Defaults and Restart**
  Click on this button to restore the factory configuration settings with the exception of the following parameters and to restart:
  - IP addresses
  - Subnet mask
  - IP address of the default gateway
  - DHCP client ID
  - DHCP
  - System name
  - System location
  - System contact
  - Mode of the device
- **Restore Factory Defaults and Restart**
  Click on this button to restore the factory configuration settings. The protected defaults are also reset.
  An automatic restart is triggered.

  Note

  By resetting to the factory configuration settings, the device loses its configured IP address, see section "Requirements for operation".
  - M87x, M81x: The device can once again be reached at the IP address 192.168.1.1 that was set in the factory.
  - M826: An IP address must be assigned to the device. Following this, the device can only be accessed using the Primary Setup Tool or using DHCP.

  With the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

Load & save

Uploading and saving using TFTP

Loading and saving data via a TFTP server

On this page, you can configure the TFTP server and the file names. You can also store device data in an external file on a TFTP server or load such data from an external file from the TFTP server to the devices. This means, for example, that you can also load new firmware from a file located on a TFTP server.

Firmware

The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.

Note

Incompatibility with previous firmware versions with/without PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed.

In this situation, if a PLUG is inserted in the device, following the restart, this has the status "Not Accepted" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data. If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using the WBM page "System > PLUG".

Configuration files

Note

Configuration files and Trial mode /Automatic save

In "Automatic Save" mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In "Trial" mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write startup config" button on the "System > Configuration" WBM page to save changes in the configuration files.

CLI script file

You can download existing CLI configurations (RunningCLI) and upload your own CLI scripts (Script).

Note

The downloadable CLI script is not intended to be uploaded again unchanged.

CLI commands for saving and loading files cannot be executed with the CLI script file (Script).

Exchange of configuration data with STEP 7 Basic/Professional using a file

You use the two file types "RunningSINEMAConfig" and "SINEMAConfig" to exchange configuration data between a device (WBM) and STEP 7 Basic/Professional via a file.

Requirements:

- Same article number
- Same firmware version
- Password
  You assign the password in the WBM under "System > Load&Save > Passwords".

You can use the file types as follows:

- For offline diagnostics
  You can save the faulty configuration of a device as "RunningSINEMAConfig" via the WBM and import it in STEP 7 Basic/Professional. No connection to a real device is required for the diagnostics in STEP 7 Basic/Professional. You can export a corrected configuration and load it as "SINEMAConfig" again using the WBM.
- For configuration
  No connection to a real device is required to configure a device in STEP 7 Basic/Professional. You can export the configuration and load it as "SINEMAConfig" to the real device using the WBM.

Description

- **TFTP Server Address**
  Enter the IP address or the FQDN of the TFTP server with which you exchange data.
- **TFTP Server Port**
  Here, enter the port of the TFTP server via which data exchange will be handled. If necessary, you can change the default value 69 to your own requirements.

The table has the following columns:

- **File type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.

- File name
  A file name is preset here for every file type.

  Note
  Changing the file name

  You can change the file name preset in this column. After loading on the device, the changed file name can also be used with the Command Line Interface.

- Action (only available online)
  Select the required action. The selection depends on the selected file type, for example you can only save the log file.
  The following actions are possible:
  - Save file
    With this selection, you save a file on the TFTP server.
  - Load file
    With this selection, you load a file from the TFTP server, e.g. certificates required to establish a secure connection.

Uploading and saving using SFTP

Loading and saving data via an SFTP server

SFTP (SSH File Transfer Protocol) transfers the files encrypted. On this page, you configure the access data for the SFTP server.

You can also store device data in an external file on your client PC or load such data from an external file from the PC to the devices. This means, for example, that you can also load new firmware from a file located on your Admin PC.

On this page, the certificates required to establish a secure VPN connection can also be loaded.

Firmware

The firmware is signed and encrypted. This ensures that only firmware created by Siemens can be downloaded to the device.

Configuration files

Note
Configuration files and Trial mode /Automatic Save

In "Automatic Save" mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In "Trial" mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write Startup Config" button on the "System > Configuration" WBM page to save changes in the configuration files.

CLI script file

You can download existing CLI configurations (RunningCLI) and upload your own CLI scripts (Script).

Note

The downloadable CLI script is not intended to be uploaded again unchanged.

CLI commands for saving and loading files cannot be executed with the CLI script file (Script)

Description

The page contains the following boxes:

- **SFTP Server Address**
  Enter the IP address or the FQDN of the SFTP server with which you exchange data.
- **SFTP Server Port**
  Enter the port of the SFTP server via which data exchange will be handled. If necessary, you can change the default value 22 to your own requirements.
- **SFTP User**
  Enter the user for access to the SFTP server. This assumes that a user with the corresponding rights has been created on the SFTP server.
- **SFTP Password**
  Enter the password for the user
- **SFTP Password Confirmation**
  Confirm the password.

The table has the following columns:

- **Type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.

- **Filename**
  A file name is preset here for every file type.

  Note

  Changing the file name

  You can change the file name preset in this column. After loading on the device, the changed file name can also be used with the Command Line Interface.

- **Actions**
  Select the required action. The selection depends on the selected file type, for example you can only save the log file.
  The following actions are possible:
  - **Save file**
    With this selection, you save a file on the SFTP server.
  - **Load file**
    With this selection, you load a file from the SFTP server, e.g. certificates required to establish a secure connection.

Passwords

Password for files

There are files to which access is password protected. To load the file on the device, enter the password specified for the file on the page.

Description

- **Type**
  Shows the file type
- **Description**
  Shows the short description of the file type.
- **Enabled**
  When selected, the file is used. Can only be enabled if the password is configured.
- **Password**
  Enter the password for the file.

- **Password Confirmation**
  Confirm the new password.
- **Status** (only available online)
  Shows whether the password matched the file on the device.
  - Valid
    The "Enabled" check box is selected and the password matches the file.
  - Invalid
    The "Enabled" check box is selected but the password does not match the file or no file has been loaded yet.
  - '-'
    The password cannot be evaluated or is not yet being used. The "Enabled" check box is not selected.

Events

Configuration

Selecting system events

On this page, you specify which system events are logged and how.

The following messages are always entered in the event log table and cannot be deselected:

- Changing the admin password
- Starting the device
- Operational status of the device, e.g. whether or not a PLUG is inserted
- Status of errors not yet dealt with

To send these messages to a Syslog server as well, enable the "Syslog" button for the event "System General Logs".

Description

Table 1 has the following columns:

- **All Events**
  Shows that the settings are valid for all events of table 2.
- **E-mail / Trap / Log- Table / Syslog / Fault / SMS / Digital Out / VPN Tunnel**
  Enable or disable the required type of notification for all events. If "No Change" is selected, the entries of the corresponding column in table 2 remain unchanged.
- **Copy To Table**
  If you click the button, the setting is adopted for all events of table 2.

Table 2 has the following columns:

- **Event**
  The "Event" column contains the following:
  - Cold/Warm Start
    The device was turned on or restarted by the user. In the error memory of the device a new entry is generated with the type of restart performed.
  - Link Change
    This event occurs only when the port status is monitored and has changed, see "System > Fault Monitoring > Link Change".
  - Authentication Failure
    This event occurs when access is attempted with an incorrect password.
  - Fault State Change
    The fault status has changed. The fault state can relate to the activated port monitoring, the response of the signaling contact or the power supply monitoring.
  - Security Logs
    An entry is made in the security log if the IPsec method was used for VPN.
  - Firewall Logs
    Each time individual firewall rules are applied, this is recorded in the firewall log. To do this, the LOG function must be enabled for the various firewall functions.
  - DDNS Client Logs
    The event occurs when the DDNS client synchronizes the assigned IP address with the hostname registered at the DDNS provider.
  - System Connection Status
    The connection status has changed.
  - System General Logs
    Connection establishment, change to the configuration.
  - Digital In
    The event occurs when the status of the digital input has changed.
  - VPN Tunnel
    The event occurs when the status of VPN (IPsec, OpenVPN, SINEMA RC) has changed.
  - Secure NTP
    The event occurs when the status of the secure NTP server has changed.
  - Configuration Change
    This event occurs when the configuration of the device has changed.
- **E-mail**
  The device sends an e-mail. This is only possible if the SMTP server is set up and the "SMTP client" function is enabled.
- **Trap**
  The device sends an SNMP trap. This is only possible if "SNMPv1 Traps" is enabled in "System > Configuration".
- **Log Table**
  The device writes an entry in the event log table, see "Information > Log Table"

- Syslog
  The device writes an entry to the system log server. This is only possible if the system log server is set up and the "Syslog client" function is enabled.
- Faults
  The device triggers an error. The fault LED lights up
- SMS (only with M87x)
  The device sends an SMS message. This is only possible if "System > SMS > Event SMS" is enabled and the telephone number of the recipient is configured.
- Digital Out
  Controls the digital output or signals the status change with the "DO" LED.
- VPN Tunnel
  Controls the forwarding of an event to a VPN connection (IPsec, OpenVPN, SINEMA RC). As long as the event is present, the VPN connection is switched to active.

Severity Filters

On this page, you configure the severity for the sending of system event notifications.

Description

The table has the following columns:

- Client Type
  Select the client type for which you want to make settings:
  - E-mail
    Sending system event messages by e-mail.
  - Log Table
    Entry of system events in the log table.
  - Syslog
    Entry of system events in the Syslog file
  - SMS (M87x only)
    Sending system event messages by SMS message.
- Severity
  Select the required level. The following settings are possible:
  - Info
    The messages of all levels are sent or logged.
  - Warning
    The message of this level and the "critical" level are sent or logged.
  - Critical
    Only the messages of this level are sent or logged.

SMTP client

The device provides the option of automatically sending an e-mail if an alarm event occurs (for example to the network administrator). The e-mail contains the identification of the sending device, a description of the cause of the alarm in plain language, and a time stamp. This allows centralized network monitoring to be set up for networks with few nodes based on an E-mail system. When an e-mail event message is received, the WBM can be started by the browser using the identification of the sender to read out further diagnostics information.

On this page, you can configure up to three SMTP servers and the corresponding e-mail addresses.

Description

The page contains the following boxes:

- SMTP Client
  Enable or disable the SMTP client.
- Sender Email Address
  Enter the name of the sender to be included in the e-mail, for example the device name. This setting applies to all configured SMTP servers.
- Send Test Mail
  Send a test e-mail to check your configuration.
- SMTP Port
  Enter the port via which your SMTP server can be reached.
  Factory settings: 25
  This setting applies to all configured SMTP servers.
- SMTP Server Address
  Enter the IP address, the FQDN (Fully Qualified Domain Name) or the host name of the SMTP server.

This table contains the following columns:

- Select
  Select the check box in a row to be deleted.
- SMTP Server Address
  Shows the IP address, the FQDN (Fully Qualified Domain Name) or the host name of the SMTP server.
- Receiver Email Address
  Enter the e-mail address to which the device sends an e-mail if a fault occurs.

SNMP

General

On this page, you make the basic settings for SNMP. Enable the options according to the function you want to use.

Description

- **SNMP**
  Select the SNMP protocol. The following settings are possible:
  - "-" (disabled)
    SNMP is disabled.
  - SNMPv1/v2c/v3
    SNMPv1/v2c/v3 is supported.

    Note

    Note that SNMP in versions 1 and 2c does not have any security mechanisms.

  - SNMPv3
    Only SNMPv3 is supported.
- **SNMPv1/v2 Read-Only**
  If you enable this option, SNMPv1/v2c can only read the SNMP variables.

  Note

  Community String

  For security reasons, do not use the standard values "public" or "private". Change the community strings following the initial installation.

  The recommended minimum length for community strings is 6 characters.

- **SNMPv1/v2c Read Community String**
  Enter the community string for access of the SNMP protocol.
- **SNMPv1/v2c Read/Write Community String**
  Enter the community string for read and write access of the SNMP protocol.
- **SNMPv1 Traps**
  Enable or disable the sending of SNMPv1 traps (alarm frames). In the "Trap" entry, specify the IP addresses of the devices to which SNMPv1 traps will be sent.
- **SNMPv1/v2c Trap Community String - Trap**
  Enter the community string for sending SNMPv1/v2c messages.
- **SNMPv3 User Migration**
  - **Enabled**
    If the function is enabled, an SNMP engine ID is generated that can be migrated. You can transfer configured SNMPv3 users to a different device.
    If you enable this function and load the configuration of the device on another device, configured SNMPv3 users are retained.
  - **Disabled**
    If the function is disabled, a device-specific SNMP engine ID is generated. To generate the ID, the agent MAC address of the device is used. You cannot transfer this SNMP user configuration to other devices.
    If you load the configuration of the device on another device, all configured SNMPv3 users are deleted.
- **SNMP Engine ID** (only available online)
  Shows the SNMP engine ID.

Traps

If an alarm event occurs, a device can send SNMP traps (alarm frames) to up to ten different management stations at the same time. Traps are only sent if the events specified in "Events" occur.

Note

SNMP traps are sent only when the "SNMPv1 Traps" setting was selected in "SNMP > General" or in "System > Configuration".

Description

- **Trap Receiver Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the station to which the device sends SNMP traps. You can specify up to ten different recipients.

The table has the following columns:

- **Trap Receiver Address**
  If necessary, change the IP addresses, the FQDN (Fully Qualified Domain Name) of the stations.
- **Trap**
  Enable or disable the sending of SNMP traps. Stations that are entered but not selected do not receive SNMP traps.

v3 Groups

Security settings and assigning permissions

SNMP version 3 allows permissions to be assigned, authentication, and encryption at protocol level. The security level and read/write permissions are defined according to groups. The settings automatically apply to every member of a group.

Description

- **Group Name**
  Enter the name of the group. The maximum length is 32 characters.
- **Security Level**
  Select the security level (authentication, encryption) valid for the selected group. The available options are as follows:
  - no Auth/no Priv
    No authentication enabled / no encryption enabled.
  - Auth/no Priv
    Authentication enabled / no encryption enabled.
  - Auth/Priv
    Authentication enabled / encryption enabled.

The table has the following columns:

- Group Name
  Shows the defined group names.

  Note

  Once a group name and the security level have been specified, they can no longer be modified after the group is created. If you want to change the group name or the security level , you will need to delete the group and recreate it and reconfigure it with the new name.

- Security Level
  Shows the configured security level.
- Read
  Enable or disable read access for the required group.
- Write
  Enable or disable write access for the required group.

  Note

  For write access to work, you also need to enable read access.

- Persistence
  Shows whether or not the group is assigned to an SNMPv3 user. If the group is not assigned to an SNMPv3 user, no automatic saving is triggered and the configured group is deleted after restarting the device.
  - Yes
    The group is assigned to an SNMPV3 user.
  - No
    The group is not assigned to an SNMPV3 user.

v3 users

User-specific security settings

On the page, you can create new SNMPv3 users and modify or delete existing users. The user-based security model works with the concept of the user name; in other words, a user ID is added to every frame. This user name and the applicable security settings are checked by both the sender and recipient.

Description

- User name
  Enter a freely selectable user name. After you have entered the data, you can no longer modify the name.

The table has the following columns:

- User name
  Shows the created users.
- Group Name
  Select the group to which the user will be assigned.
- Authentication Protocol
  Specify the authentication protocol for which a password will be stored.
  The following settings are available:
  - None
  - MD5
  - SHA
- Encryption Protocol
  Specify whether or not a password should be stored for encryption with the DES algorithm.
  Can only be enabled when an authentication protocol has been selected.
- Authentication Password
  Enter the authentication password. This password must have at least 1 character, the maximum length is 32 characters.

  Note
  Length of the password

  As an important measure to maximize security, we recommend that the password has a minimum length of 6 characters and that it contains special characters, uppercase/ lowercase letters, numbers.

- Confirm authentication password
  Confirm the password by repeating the entry.
- Encryption Password
  Enter your encryption password. This password must have at least 1 character, the maximum length is 32 characters.

  Note
  Length of the password

  As an important measure to maximize security, we recommend that the password has a minimum length of 6 characters and that it contains special characters, uppercase/ lowercase letters and numbers.

- **Confirm Encryption Password**
  Confirm the encryption password by repeating the entry.
- **Persistence**
  Shows whether or not the user is assigned to an SNMPv3 group. If the user is not assigned to an SNMPv3 group, no automatic saving is triggered and the configured user is deleted after restarting the device.
  - Yes
    The user is assigned to an SNMPv3 group.
  - No
    The user is not assigned to an SNMPv3 group.

System time

Manual time setting

Manual setting of the system time

On this page, you set the date and time of the system. For this setting to be used, enable "Manual time setting".

Note

The page is only available if there is an online connection to the device.

Description

- **Time Manually**
  Enable or disable the manual time setting. If you enable the option, the "System Time" input box can be edited.
- **System Time**
  Enter the date and time in the format "MM/DD/YYYY HH:MM:SS".
- **Use PC Time**
  Click the button to use the time setting of the PC.
- **Last Synchronization Time**
  Shows when the last time-of-day synchronization took place.
  If no time-of-day synchronization was possible, the box displays "Date/time not set".

- **Last Synchronization Mechanism**
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame.
  - PTP
    Automatic time-of-day synchronization with PTP
- **Daylight Saving Time**
  Shows whether the daylight saving time changeover is active.
  - active (offset + 1 h)
    The system time was changed to daylight saving time; in other words an hour was added. You can see the current system time at the top right in the selection area of the WBM. The set time continues to be displayed in the "System Time" box.
  - inactive (offset +0 h)
    The current system time is not changed.

SNTP client

There are different methods that can be used to set the system time of the device. Only one method can be active at any one time. If one method is activated, the previously activated method is automatically deactivated.

SNTP (Simple Network Time Protocol) is used for synchronizing the time in the network. The appropriate frames are sent by an SNTP server in the network.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Requirement

To receive the SNTP frames, enable the entry "System Time" under "Security > Firewall > Predefined IPv4 rules".

Description

- **SNTP Client**
  Enable or disable automatic time-of-day synchronization using SNTP.
- **Current System Time** (only available online)
  Shows the current date and current normal time received by the device. If you specify a time zone, the time information is adapted accordingly.
- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place.
- **Last Synchronization Mechanism** (only available online)
  Shows how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time.
  The time in the "Current System Time" box is adapted accordingly.
- **SNTP Mode**
  Select the synchronization mode. The following types of synchronization are possible:
  - Poll
    If you select this protocol type, the input boxes "SNTP Server Address", "SNTP Server Port" and "Poll Interval[s]" are displayed to allow further configuration. With this type of synchronization, the device is active and sends a time query to the SNTP server.
  - Listen
    With this type of synchronization, the device is passive and receives SNTP frames that deliver the time of day.
- **Poll Interval[s]**
  Here, enter the interval between two-time queries. In this box, you enter the query interval in seconds. Possible values are 16 to 16284 seconds.
- **SNTP Server Address**
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the SNTP server.

- **SNTP Server Port**
  Enter the port of the SNTP server.
  The following ports are possible:
  - 123 (standard port)
  - 1025 to 36564
- **Primary**
  The check mark is set for the SNTP server that you create first. If several SNTP servers have been created, the primary server is queried first.

NTP client

If you require time-of-day synchronization using NTP, you can make the relevant settings here.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Requirement

To receive the NTP frames, enable the entry "System Time" under "Security > Firewall > Predefined IPv4 rules".

Description

- **NTP Client**
  When enabled, the device receives the system time from an NTP server.
- **Secure NTP Client Only**
  When enabled, the device receives the system time from a secure NTP server. The setting applies to all server entries.
  To enable the secure NTP client, you configure the parameters for authentication (key ID, hash algorithm, key). These entries must be present on the secure NTP server.
- **Current System Time** (only available online)
  Shows the current date and current normal time received by the IE switch. If you specify a time zone, the time information is adapted accordingly.
- **Last Synchronization Time**(only available online)
  Shows when the last time-of-day synchronization took place.

- **Last Synchronization Mechanism** (only available online)
  Shows how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time.
  The time in the "Current System Time" box is adapted accordingly.
- **NTP Server Index**
  Select the index of the NTP server. The server with the lowest index is queried first.
- **NTP Server Address**
  Enter the IPv4 address or the FQDN (Fully Qualified Domain Name) of the NTP server.
- **NTP Server Port**
  Enter the port of the NTP server.
  The following ports are possible:
  - 123 (standard port)
  - 1025 to 36564
- **Poll Interval**
  Specify the interval between two-time queries. The greater the interval, the less accurate the time of the device.
  Possible values are 64 to 2592000 seconds (30 days).
- **Key ID**
  Enter the ID of the authentication key.
- **Hash Algorithm**
  Specify the format for the authentication key.
- **Key**
  Enter the authentication key. The length depends on the hash algorithm.
  - DES: ASCII 8 characters
  - MD5: ASCII 16 – 128 characters
  - SHA1: ASCII 20 – 128 characters
- **Key Confirmation**
  Enter the authentication key again to confirm it.

SIMATIC time client

Time setting via SIMATIC time client

On this page, you configure time synchronization with the SIMATIC Time Client.

Note

To avoid time jumps, make sure that there is only one time server in the network.

Description

- **SIMATIC Time Client**
  Select this check box to enable the device as a SIMATIC time client.
- **Current System Time** (only available online)
  Shows the current system time.
- **Last Synchronization Time** (only available online)
  Shows when the last time-of-day synchronization took place.
- **Last Synchronization Mechanism** (only available online)
  Shows how the last time-of-day synchronization was performed. The following methods are possible:
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame

NTP server

On this page, you configure the device as an NTP server. The other devices can call up the time made available by the device via this NTP server. This means that the supplied devices are not dependent on a connection to an external time server.

Note

Time synchronization

Also configure the device as NTP client so that it synchronizes the connected devices to a correct time. As NTP client, the device gets the precise time from an external time server and distributes it to its NTP clients as NTP server.

Requirement

To receive the NTP frames, enable the entry "System Time" under "Security > Firewall > Predefined IPv4 rules".

Description

The page contains the following boxes:

- NTP Server
  Enable or disable the NTP server service.

Table 1 has the following columns

- All interfaces
  Shows that the settings are valid for all interfaces of table 2.
- Listen
  In the drop-down list, select the setting for all interfaces. If "No Change" is selected, the entries of the corresponding column in table 2 remain unchanged.
- Copy to Table
  If you click the button, the setting is adopted for all interfaces of table 2.

Table 2 has the following columns

- Interface
  Via this interface the time is transferred using NTP.
- Listen
  When enabled, the other devices can call up the time via this interface.

Auto logout

On this page, set the times after which there is an automatic logout from WBM or the CLI following user in activity.

If you have been logged out automatically, you will need to log in again.

Note

No automatic logout from the CLI

If the connection is not terminated after the set time, check the setting of the "keepalive" mechanism on the Telnet client.

If the interval is shorter than the configured time, the connection is kept alive although no user data is transferred. You have set, for example, 300 seconds for the automatic logout and the "keepalive" function is set to 120 seconds. In this case, a packet is sent every 120 seconds that keeps the connection up.

- Disable the keepalive mechanism (interval time = 0)
  or
- Set the interval high enough so that the underlying connection is terminated when there is inactivity.

Description

- **Web Based Management [s]**
  Enter the time in seconds for the automatic logout from WBM. If you enter the value 0, the automatic logout is disabled.
- **CLI (TELNET, SSH) [s]**
  Enter the time in seconds for the automatic logout from the CLI. If you enter the value 0, the automatic logout is disabled.

Button

Functionality

The SET button is used for:

- Restart
- Loading new firmware,
- Resetting to factory settings.

You will find a detailed description of the functions in the device operating instructions.

On this page, the functionality of the button can be restricted.

Description

The following functionality is possible:

- **Restart / Restore Factory Defaults**
  When disabled, the SET button cannot be used for a restart or to restore factory defaults.

  ⚠ CAUTION

  Button function "Restart / Restore Factory Defaults" active during startup

  If you have disabled this function in your configuration, disabling is only valid during operation. When restarting, for example after power down, the function is active until the configuration is loaded so that the device can inadvertently be reset to the factory settings. This may cause unwanted disruption in network operation since the device then needs to be reconfigured. An inserted PLUG is also deleted and returned to the status as shipped.

  You will find more information on how to restore the device to the factory defaults despite disabled functions in the configuration manual in the section "Upkeep and maintenance".

Syslog client

System event agent

Syslog according to RFC 3164 is used for transferring short, unencrypted text messages over UDP in the IP network. This requires a Syslog server.

Requirements for sending log entries

- The Syslog function is enabled on the device.
- The Syslog function is enabled for the relevant event.
- There is a Syslog server in your network that receives the log entries. Since this is a UDP connection, there is no acknowledgment to the sender.
- The IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server is entered in the device.

Description

The page contains the following boxes:

- Syslog Client
  Enable or disable the Syslog function.
- Syslog Server Address
  Enter the IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server.

The table contains the following columns:

- Syslog Server Address
  Shows the IP address or the FQDN (Fully Qualified Domain Name) of the Syslog server.
- Server Port
  Enter the port of the Syslog server being used.

Fault Monitoring

Link Change

Configuration of fault monitoring of status changes on connections

On this page, you configure whether or not an error message is triggered if there is a status change on a network connection.

If connection monitoring is enabled, an error is signaled

- when there should be a link on a port and this is missing.
- or when there should not be a link on a port and a link is detected.

An error causes the signaling contact to trigger and the fault LED to light up on the device. Depending on the configuration, the error may trigger a trap, an e-mail or an entry in the event log table.

Description of the displayed boxes

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Setting**
  Select the required setting. The available options are as follows:
  - "-" (disabled)
  - Up
  - Down
  - No Change: The setting in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns

- **Port**
  Shows the available ports.
- **Setting**
  Select the setting. You have the following options:
  - Up
    Error handling is triggered when the port changes to the active status.
    (From "Link down" to "Link up")
  - Down
    Error handling is triggered when the port changes to the inactive status.
    (From "Link up" to "Link down")
  - "-" (disabled)
    The error handling is not triggered.

Mobile radio

On this page, you configure whether or not an error message is triggered if there is a status change of the SIM card.

Description

The table has the following columns:

- **SIM card**
  Shows the available SIM cards.
- **Setting**
  Select the setting from the drop-down list. You have the following options:
  - "-" (disabled)
    The error handling is not triggered.
  - SIM missing
    Troubleshooting is triggered if no SIM card is plugged in. An error causes the fault LED to light up on the device.

PLUG

Configuration

Note

The page is only available if there is an online connection to the device.

NOTICE
Do not remove or insert a C-PLUG / KEY-PLUG during operation! A PLUG may only be removed or inserted when the device is turned off. The device checks whether or not a PLUG is inserted at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid KEY-PLUG was inserted in the device, the device changes to a defined error state following the restart. With SCALANCE W, the available wireless interfaces are deactivated in this case. If the device was configured at some time with a PLUG, the device can no longer be used without this PLUG. To be able to use the device again, reset the device to the factory settings.

Information about the configuration of the C-PLUG

This page provides detailed information about the configuration stored on the C-PLUG. It is also possible to reset the PLUG to factory defaults or to load it with new contents.

Note

The action is only executed after you click the "Set Values" button.

The action cannot be undone.

If you decide against executing the function after making your selection, click the "Refresh" button. As a result the data of this page is read from the device again and the selection is canceled.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Description

The table has the following rows:

- **Status**
  Shows the status of the PLUG. The following are possible:
  - ACCEPTED
    There is a PLUG with a valid and suitable configuration in the device.
  - NOT ACCEPTED
    Invalid or incompatible configuration on the inserted PLUG.
  - NOT PRESENT
    There is no C-PLUG or KEY-PLUG inserted in the device.
  - FACTORY
    PLUG is inserted and does not contain a configuration. This status is also displayed when the PLUG was formatted during operation.
  - MISSING
    There is no PLUG inserted. Functions are configured on the device for which a license is required.
- **Device Group**
  Shows the SIMATIC NET product line that used the C-PLUG or KEY-PLUG previously.
- **Device Type**
  Shows the device type within the product line that used the C-PLUG or KEY-PLUG previously.
- **Configuration Revision**
  The version of the configuration structure. This information relates to the configuration options supported by the device and has nothing to do with the concrete hardware configuration. This revision information does not therefore change if you add or remove additional components (modules or extenders), it can, however, change if you update the firmware.
- **File System**
  Displays the type of file system on the PLUG.
- **File System Size [bytes]**
  Shows the maximum storage capacity of the file system on the C-PLUG.

- **File System Usage**
  Displays the storage space in use in the file system of the C-PLUG.
- **Firmware on PLUG**
  When enabled, the firmware will be stored on the PLUG. This means that automatic firmware updates/downgrades can be made with the PLUG.
- **Info String**
  Shows additional information about the device that used the PLUG previously, for example, article number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.
  If a PLUG was configured as a PRESET PLUG this is shown here as additional information in the first row. For more detailed information on creating and using a PRESET PLUG refer to the section "Maintenance (page 363)".
- **Modify PLUG**
  Select the required setting. You have the following options for changing the configuration on the C-PLUG or KEY-PLUG:
  - Write current configuration to PLUG
    This option is available only if the status of the PLUG is "NOT ACCEPTED" or FACTORY. The configuration in the internal flash memory of the device is copied to the PLUG.
  - Erase PLUG to factory default
    Deletes all data from the C-PLUG and triggers low-level formatting.

License

Note

The page is only available if there is an online connection to the device.

NOTICE

Do not remove or insert a C-PLUG / KEY-PLUG during operation!

A PLUG may only be removed or inserted when the device is turned off.
The device checks whether or not a PLUG is present at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid KEY-PLUG was inserted in the device, the device changes to a defined error state following the restart. With SCALANCE W, the available wireless interfaces are deactivated in this case.

If the device was configured at some time with a PLUG, the device can no longer be used without this PLUG. To be able to use the device again, reset the device to the factory settings.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Information about the license of the KEY-PLUG

A C-PLUG can only store the configuration of a device. In addition to the configuration, a KEY-PLUG also contains a license that enables certain functions of your SIMATIC NET device.

Note

Information only available online

You can only view the information on this page if there is an online connection to the device.

Description

- **Status**
  Shows the status of the KEY-PLUG. The following are possible:
  - ACCEPTED
    The KEY-PLUG in the device contains a suitable and valid license.
  - NOT ACCEPTED
    The license of the inserted KEY-PLUG is not valid.
  - NOT PRESENT
    No KEY-PLUG is inserted in the device.
  - MISSING
    There is no KEY-PLUG or a C-PLUG with the status "FACTORY" inserted in the device. Functions are configured on the device for which a license is required.
  - WRONG
    The inserted KEY-PLUG is not suitable for the device.
  - UNKNOWN
    Unknown content of the KEY-PLUG.
  - DEFECTIVE
    The content of the KEY-PLUG contains errors.
- **Article number**
  Shows the article number of the KEY-PLUG. The KEY-PLUG is available for various functional enhancements and for various target systems.

- **Serial Number**
  Shows the serial number of the KEY-PLUG.
- **Info String**
  Shows additional information about the device that used the KEY-PLUG previously, for example, article number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.

Note

When you save the configuration, the information about whether or not a KEY-PLUG was inserted in the device at the time is also saved. This configuration can then only work if a KEY-PLUG with the same article number / license is inserted.

SMS

General

For the device to be able to send an SMS message, you need to enter a specific SMS center (Short Message Server Center, SMS-C) of your mobile wireless provider or your service provider.

- If you use the standard SMS center, you do not need to configure anything on this page. The SIM card already contains the correct information.
- If you do not use the standard SMS center, you can replace the stored call number of the standard SMS center with another number on this page. This depends on your contract.

Description

The page contains the following boxes:

- **Current SMS service center call number** (only available online)
  Shows the stored call number of the SMS center.
- **Override SMS service center call number**
  Enter the call number of the SMS center including the country dialing code. Confirm the entry to store the call number on the SIM card. If the call number is displayed in "Current SMS service center call number", the stored call number of the SMS center has been overwritten and the input box emptied.

Note

When the change takes effect, the mobile wireless connection will be interrupted for a short time. The change can take up to a minute.

Event SMS

If events occur, the device can automatically send an SMS message.

Requirement:

- In "System > Events > Configuration", "SMS" is activated for the relevant event.

Description

The page contains the following:

- Enable Event SMS
  When enabled, the device sends an SMS.

In the area "Customize SMS messages for Digital Input" you specify the SMS text for the digital input.

- Digital input
  Shows the available digital inputs.
- Rising edge
  Enter the SMS text that is sent when the signal at the digital input changes from 0 (LOW) to 1 (HIGH).
- Falling edge
  Enter the SMS text that is sent when the signal at the digital input changes from 1 (HIGH) to 0 (LOW).

Note

Characters permitted for the SMS text

The following characters are permitted in the text:

- 0123456789
- A...Z a...z
- Space
- ! % & / ( ) = * + < > ' , . -

- **Sending Option**
  Specify when the SMS text is sent.
  - None
    The function is disabled.
  - Both
    In both cases, an SMS text is sent.
  - Rising only
    Only the SMS text is sent when the signal at the digital input changes from 0 (LOW) to 1 (HIGH).
  - Falling only
    Only the SMS text is sent when the signal at the digital input changes from 1 (HIGH) to 0 (LOW).
- **Phone Number**
  Enter the full telephone number of the recipient including the country dialing code
- This table contains the following columns:
  - **No.**
    Shows the unique number of the phone number.
  - **Send**
    Specify whether the device sends an SMS message to this recipient.
  - **Phone Number**
    Shows the phone number of the recipient.

Command SMS message

Commands can also be sent with SMS messages. On this page, you specify who can control the device. As identification either the phone number or the sender ID is specified. The sender ID can, for example, be an IP address or a DNS name.

With an SMS command, you can, for example, establish a VPN connection to an application such as the SINEMA RC Server. To do this, the SINEMA RC Server sends a wake-up SMS message containing the required commands. You will find more detailed information in the configuration manual in "Command SMS" or in Getting Started.

Description

The page contains the following:

- **Enable Command SMS**
  Enable or disable receipt of command SMS messages.
- **Phone Number / Sender Identifier**
  Specify the sender ID or the phone number including the country code.
  In phone numbers, you can replace any numbers to the right with the asterisk (*) placeholder. The placeholder can only be positioned at the end of a number sequence.
  Example:
  +49144545* stands for all phone numbers that start with 0144545.

  Note

  Please note that when you use the placeholder "*", command text messages (SMS) from a wide range of phone numbers will be accepted. We therefore recommend that you use a more restrictive input, if possible.

This table contains the following columns:

- **No.**
  Shows the unique number of the entry.
- **Phone Number / Sender Identifier**
  Shows the sender ID or the phone number.
- **System / Relay**
  Specify which class (System / Relay) of commands will be accepted from this sender.

SMS Relay (Outgoing)

With the SMS messaging function, applications connected to the Ethernet interface of the device can send SMS messages. To send an SMS message, the application must establish a TCP/IP connection to the device via the Ethernet interface. Via this TCP/IP connection, the application transfers the text of the SMS to the device that packs the text in an SMS message and sends it.

Format of the SMS text

The text is transferred in a frame via the TCP/IP connection to the device. The frame must have the following format:

Username#Password#CommandCode#Seq-Num;Callnumber;Message:

```
Example: user#password#105#01;0049xxxxxxxxx;my SMS text:
```

- **User name**
  Enter a user name to check the permission for sending an SMS message. Maximum of 10 characters.
- **Password**
  Enter the password belonging to the user name. Maximum of 10 characters.
- **CommandCode**
  Command to send an SMS message from the local network. This value of 105 is fixed and must not be modified.

- **Seq-Num**
  The sequence number is used to assign several requests at the same time. The function is not currently supported.
  The sequence number consists of 2 numeric characters from 01 to 99.
- **Call number**
  Call number of the SMS recipient with a maximum of 40 characters. International numbers (+49) are permitted.
- **Message**
  SMS text with a maximum of 160 characters

Note

Characters permitted for sending SMS messages

The following characters are permitted in the text:

- 0123456789
- A...Z a...z
- Space
- ! " % & / ( ) = ? * + < > ' , . -

Description

The page contains the following:

- **Enable SMS Relay (Outgoing)**
  Enable or disable the sending of SMS messages from the local network.
- **User**
  Enter the user name that must be included before the text is sent as an SMS message.
- **Password**
  Enter the password that must be included before the text is sent as an SMS message.
- **Password (Confirmation)**
  Repeat the password to confirm it.
- **Server Port Number**
  Enter the port at which the server receives the SMS.

SMS Relay (Incoming)

Applications connected to the Ethernet interface of the device can receive SMS messages. To receive an SMS message, the application must establish a TCP/IP connection to the device via the Ethernet interface. The device receives a command SMS message and forwards the text it contains to the application.

Format of the SMS text

The text is sent in a frame via the TCP/IP connection to the application. The frame must have the following format:

Username#Password#CommandCode#Seq-Num;Callnumber;Message:

```
Example: user#password#105#01;0049xxxxxxxxx;my SMS text:
```

- **User name**
  Enter a user name to check the receipt of the SMS message. Maximum of 10 characters.
- **Password**
  Enter the password belonging to the user name. Maximum of 10 characters.
- **CommandCode**
  Command to send an SMS message to the local network. This value of 105 is fixed.
- **Seq-Num**
  The sequence number is used to assign several requests at the same time. The function is not currently supported.
  The sequence number consists of 2 numeric characters from 01 to 99.
- **Call number**
  Phone number of the sender of the command SMS message with a maximum of 40 characters. International numbers (+49) are permitted.
- **Message**
  SMS text with a maximum of 160 characters

Note

Characters permitted for sending SMS messages

The following characters are permitted in the text:

- 0123456789
- A...Z a...z
- Space
- ! " % & / ( ) = ? * + < > ' , . -

Description

The page contains the following:

- **Connection Name**
  Enter a unique name for the relay connection.
- **IP Address**
  Enter the IP address of the recipient. The frame is sent to this IP address.
- **Port Number**
  Enter the port to which the frame will be sent.
- **Username**
  Enter the user name to check receipt of the message. Enter the user name in the frame.
- **Password**
  Enter the password belonging to the user name.
- **Password (Confirmation)**
  Repeat the password to confirm it.

This table contains the following columns:

- Connection Name
  Shows the name of the relay connection.
- IP Address
  Shows the IP address of the recipient.
- Port Number
  Shows the port.
- Username
  Shows the user name contained in the frame.

DNS

DNS client

The DNS server (Domain Name System) assigns a domain name to an IP address so that a device can be uniquely identified.

If this setting is enabled, the device can communicate with a DNS server as a DNS client.

Description

- DNS client
  Enable or disable depending on whether the device should operate as a DNS client.
- DNS server used
  - learned only
    The device uses only the DNS servers assigned by DHCP.
  - manual only
    The device uses only the manually configured DNS servers. The DNS servers must be connected to the Internet.
  - all
    The device uses all available DNS servers.
- DNS Server Address
  Enter the IP address of the DNS server.

The table has the following columns:

- DNS Server Address
  Shows the IP address of the DNS server.
- Origin
  This shows whether the DNS server was configured manually or was assigned by DHCP.

DNS proxy

The device provides a DNS server for the local network. If you enter the IP address of the device in the local application as a DNS server, then the device answers the DNS requests from its cache.

If the device does not know the IP address for a domain address, it forwards the query to an external DNS server. How long the device keeps a domain address in the cache depends on the host being addressed. In addition to the IP address, a DNS request to an external DNS server also supplies the life span of this information.

Description

The page contains the following boxes:

- **Enable DNS Proxy**
  Enable or disable the proxy of the DNS server.
- **Cache Name Errors (NXDOMAIN)**
  Enable or disable the caching of NXDOMAIN replies. If you enable the option, the domain names that were unknown to the DNS server remain in the cache.

DDNS client

The DDNS (Dynamic Domain Name System) is an Internet service that allows a fixed hostname to be set up as a pseudonym for a dynamically changing IP address.

The DDNS client synchronizes the assigned IP address with the hostname registered at the DDNS provider. This means that the device can always be reached using the same hostname.

Requirement:

- User name and password that gives you the right to use the DDNS service.
- Registered hostname, e.g. example.no-ip.com
- UDP port 53 for DNS is enabled and is not used for NAT.

Description

The table has the following columns:

- **Service**
  Shows which providers are supported.
- **Enabled**
  When enabled, the device logs on to the DDNS server.
- **Host**
  Enter the host name that you have agreed with your DDNS provider for the device, e.g. example.no-ip-com.
- **User Name**
  Enter the user name with which the device logs on to the DDNS server.
- **Password**
  Enter the password assigned to the user.
- **Password Confirmation**
  Confirm the password.

DHCP

DHCP client

If the device is configured as a DHCP client, it starts a DHCP request. As the reply to the query the device receives an IPv4 address from the DHCP server. The server manages an address range from which it assigns IPv4 addresses. It is also possible to configure the server so that the client always receives the same IPv4 address in response to its request.

Description

The page contains the following boxes:

- **DHCP Client Configuration Request (Opt. 66, 67)**
  When enabled, the DHCP client uses the options to download the configuration file (option 67) from the TFTP server (option 66). After the restart, the device uses the data from the configuration file.

  Note

  Configuration file and firmware version

  The configuration file is used to store and read in configuration data within a firmware version, e.g. 4.3. Configuration files created with a firmware version <4.2 cannot be read in to a device with a firmware version 4.3.

  Note

  If a configuration file is downloaded, this can trigger a system restart. If the currently running configuration and the configuration in the downloaded configuration file differ, the system restarts.

  Make sure that the option "DHCP Client Configuration Request (Opt. 66, 67)" is no longer set.

- **DHCP Mode**
  Specify the type of identifier with which the DHCP client logs on with its DHCP server.
  - via MAC Address
    Identification is based on the MAC address.
  - via DHCP Client ID
    Identification is based on a freely defined DHCP client ID.
  - via System Name
    Identification is based on the system name. If the system name is 255 characters long, the last character is not used for identification.
  - via Iaid and Duid
    With this the DHCP client can log on with DHCP servers that support parallel operation of IPv4 and IPv6.
    The identification is via the IAID and the DUID and identifies precisely one IP interface of the device.
    IAID (Interface Association Identifier): At least one IAID is generated for each IP interface The IAID remains unchanged when the DHCP client restarts
    DUID (DHCP Unique Identifier): Uniquely identifies server and clients and applies to all IP interfaces of the device. The DUID remains unchanged when there is a restart. Unless the user changes this.

- **DUID-Type**
  Specify which DUID type will be used. The DUID types are defined in RFC 3315.
  - DUID-LLT
    DUID is based on the link layer address of the interface and a time stamp
  - DUID-EN
    DUID is assigned by the vendor (EN = enterprise number)
  - DUID-LL
    DUID is based on the link layer address of the interface
- **Link-layer Address Plus Time (LLT)**
  The value is based on the link layer address of the interface and a time stamp. The value is regenerated each time the factory settings are restored. When necessary the value can be changed.
- **Vendor Enterprise Number (EN)**
  The value is based on the enterprise number specific to the vendor. The value is regenerated each time the factory settings are restored. When necessary the value can be changed.
- **Link-layer address (LL)**
  The link layer address is based on the MAC address. The value is regenerated each time the factory settings are restored. When necessary the value can be changed.

The table has the following columns:

- **Interface**
  Interface to which the setting relates.
- **DHCP**
  Enable or disable the DHCP client for the relevant interface.
- **IAID Value**
  Value with which the interface (DHCP client) identifies itself with the DHCP server.

DHCP server

You can operate the device as a DHCP server. This allows IP addresses to be assigned automatically to the connected devices. The IP addresses are either distributed dynamically from an address band (pool) you have specified or a specific IP address is assigned to a particular device.

Note

Deleting DHCP server bindings

If you deactivate or delete an IPv4 address band or turn the DHCP server off and on again, the DHCP server bindings are deleted see "Information > DHCP Server".

Requirement

The connected devices are configured so that they obtain the IP address from a DHCP server.

Description

The page contains the following boxes:

- **DHCP Server**
  Enable or disable the DHCP server on the device.

  Note

  To avoid conflicts with IPv4 addresses, only one device may be configured as a DHCP server in the network.

  If you want to operate a DHCP server on the devices of a VRRP group, note the information in the section "Layer 3 (IPv4/IPv6) > VRRP/VRRPv3 > Router".

- **Probe address with ICMP Echo before offer**
  When selected, the DHCP server checks whether or not the IP address has already been assigned. To do this the DHCP server sends ICMP echo messages (ping) to the IPv4 address. If no reply is received, the IPv4 address is assigned.

  Note

  This check is not made with static assignments.

  Note

  If there are devices in your network on which the echo service is disabled as default, there may be conflicts with the IPv4 addresses. To avoid this, assign these devices an IPv4 address outside the IPv4 address band.

The table has the following columns:

- **Pool ID**
  Shows the number of the IPv4 address band.
- **Interface**
  Select an IP interface or a router port. The IPv4 addresses are assigned dynamically via this interface.
- **Enable**
  Specify whether or not this IPv4 address band will be used.

  Note

  If you enable the IPv4 address, its settings in this DHCP tab are grayed out and can no longer be edited.

- **Subnet**
  Enter the subnet mask matching the IPv4 address. Use the CIDR notation.
- **Lower IP Address**
  Enter the IPv4 address that specifies the start of the dynamic IPv4 address band. The IPv4 address must be within the network address range you configured for "Subnet".

- **Upper IP address**
  Enter the IPv4 address that specifies the end of the dynamic IPv4 address band. The IPv4 address must be within the network address range you configured for "Subnet".
- **Lease Time (sec)**
  Specify for how many seconds the assigned IPv4 address remains valid. When half the period of validity has elapsed. the DHCP client can extend the period of the assigned IPv4 address. When the entire time has elapsed, the DHCP client needs to request a new IPv4 address.

DHCP options

On this page you specify which DHCP options the DHCP server supports. The various DHCP options are defined in RFC 2132.

Description

The page contains the following boxes:

- **Pool ID**
  Select the required address band.
- **Option Code**
  Enter the number of the required DHCP option.

  Note

  DHCP options supported

  The DHCP options 1, 2, 3. 4. 5, 6, 42, 66, 67 are supported.

  The DHCP options 1, 3, 6, 66 and 67 are created automatically when the IPv4 address band is created. With the exception of option 1, the options can be deleted.

The table has the following columns:

- **Select**
  Select the check box in the row to be deleted
- **Pool ID**
  Shows the number of the address band.
- **Option Code**
  Shows the number of the DHCP option.

- Use Interface IP
  Specify whether or not the internal IP address of the device will be used.
- Value
  Enter the DHCP parameter that is transferred to the DHCP client. The content depends on the DHCP option.

Value	Option name		
1	Subnet Mask	The subnet mask is entered automatically.	Option cannot be deleted.
2	Offset time	Offset time to the coordinated universal time UTC.	Enter the offset time in seconds in hexadecimal format.
3	Router	The IPv4 address for router in the subnet of the DHCP client. If the device itself is the router, the IPv4 address of the interface is used.	You can specify several IPv4 addresses separated by commas.
4	Time server	The IPv4 address of the time server available to the DHCP client.	
5	Name server	The IPv4 address of the name server available to the DHCP client.	
6	DNS Server	The IPv4 address of the DNS server available to the DHCP client. If the device itself is the DNS server, the IPv4 address of the interface is used.	
42	NTP Server	The IPv4 address of the NTP server available to the DHCP client.	
66	TFTP server	The IPv4 address or the hostname of the TFTP server available to the DHCP client.	Enter the address of the TFTP server.
67	Name of the bootfile	The name of the bootfile that the client downloads from the TFTP server.	Enter the name of the boot file in the string format.

Static Leases

On this page you specify that certain devices will be assigned a certain IP address. The address assignment is made based on the MAC address, the client ID or the DUID.

Description

The page contains the following boxes:

- **Pool ID**
  Select the required address band.
- **Client Identification Method**
  Select the method according to which a client is identified.
  - Ethernet MAC
    Identification is based on the MAC address. Enter the MAC address in "Value". A MAC address consists of six byes separated by hyphens in hexadecimal notation, e.g. 00-ab-1d-df-b4-1d.
  - Client ID
    Identification is based on a freely defined DHCP client ID. Enter the required designation in "Value".
  - DUID
    Identification is based on the DUID and IAID. Enter the required designation in "Value" e.g. 00-00-01-C2-00-01-00-01-00-00-00-72-00-1B-1B-B6-32-9D.
- **Value**
  Enter the required value. The entry depends on the selected identification method of the client.

  Note

  A maximum of 20 entries are possible.

The table has the following columns:

- **Select**
  Select the check box in the row to be deleted.
- **Pool ID**
  Shows the number of the address band.
- **Identification Method**
  Shows the method with which the client identifies itself with the DHCP server.
- **Value**
  Shows the MAC address or client ID or DUID of the client.
- **IP Address**
  Specify the IPv4 address that will be assigned to the client. The IPv4 address must be within the address band.

cRSP/SRS

Note

Common Remote Service Platform (cRSP) / Siemens Remote Service (SRS) is a remote maintenance platform via which remote maintenance access is possible.

To use the platform, additional service contracts are necessary and certain constraints must be kept to. If you are interested in cRSP / SRS, call your local Siemens contact or visit Web page (https://support.industry.siemens.com/en/en/sc/2281).

On this page, you configure the access data for the SRS / cRSP acc. to URI syntax. The Uniform Resource Identifier (URI) is defined in RFC 3986.

Description

The page contains the following boxes:

- **Enable DDNS for cRSP / SRS**
  Enable or disable the use of cRSP / SRS.
- **Update Interval**
  Enter the time interval.
- **Validate Server Certificate**
  When enabled, the device checks the validity of the received server certificate.

The table has the following columns:

- **Index**
  The number of the entry.
- **Scheme**
  Identifies the access method and the resource type.
  https: Secure access to a Web page.
- **Authority**
  Contains the address of the destination server
- **Path**
  Contains the target path to the resource. The target path can correspond to a directory name or file name.
- **Query**
  A query can contain parameter values for an application.
- **Frag**.
  Addresses local parts of the resource, e.g. the anchor attribute of a Web page.
- **Status** (only available online)
  Shows the status of the last cRSP / SRS access of the entry.
- **Enabled**
  When enabled, this entry is used.

Proxy Server

On this page, you configure the proxy server that is used by various components, for example SINEMA RC.

Description

- **Proxy Name**
  Enter a name for the proxy server.

The table has the following columns:

- **Name**
  Shows the name of the proxy server.
- **Address**
  Enter the IPv4 address of the proxy server.
- **Type**
  Specify the type of the proxy server.
  - HTTP: Proxy server only for access using HTTP.
  - SOCKS: Universal proxy server
- **Port**
  Enter the port on which the proxy service runs.
- **Auth. Method**
  Specify the authentication method.
  - None
    Without authentication
  - Basic
    Standard authentication. User name and password are sent unencrypted.
  - NTML (NT LAN Manager)
    Authentication according to the NTML standard (Windows user logon)
- **User Name**
  Enter the user name for access to the proxy server.
- **Password**
  Enter the password for access to the proxy server.
- **Password Confirmation**
  Enter the password again to confirm it.

SINEMA RC

On this page, you configure the access to the SINEMA RC Server.

Note

This function can only be used with a KEY PLUG.

Description

The page contains the following:

- Enable SINEMA RC
  - Enabled:
    A connection to the configured SINEMA RC Server is established. These boxes cannot be edited.
  - Disabled:
    The boxes can be edited. Any existing connection is terminated.

"Server settings" area

- SINEMA RC Address
  Enter the IPv4 address or the DNS host name of the SINEMA RC Server.
- SINEMA RC Port
  Enter the port via which the SINEMA RC Server can be reached.

"Server Verification" area

- Verification Type
  - Fingerprint: The identity of the server is verified based on the fingerprint.
  - CA certificate: The identity of the server is verified based on the CA certificate.
- Fingerprint
  Only necessary with the setting "Fingerprint". Enter the fingerprint of the device. The fingerprint is assigned during commissioning of the SINEMA RC Server. Based on the fingerprint, the device checks whether the correct SINEMA RC Server is involved. You will find further information on this in the Operating Instructions of the SINEMA RC Server.
- CA Certificate
  Only necessary with the setting "CA Certificate". Select the CA certificate of the server used to sign the server certificate. Only loaded CA certificates can be selected.

"Device Credentials" area

- Device ID
  Enter the device ID. The device ID is assigned when configuring the device on the SINEMA RC Server. You will find further information on this in the Operating Instructions of the SINEMA RC Server.
- Device Password
  Enter the password with which the device logs on to the SINEMA RC Server. The password is assigned when configuring the device on the SINEMA RC Server. You will find further information on this in the Operating Instructions of the SINEMA RC Server.
- Device Password Confirmation
  Repeat the password to confirm it.

"Optional Settings" area

- Auto Firewall/NAT Rules
  - Enabled
    The firewall and NAT rules are created automatically for the VPN connection. The connections between the configured exported subnets and the subnets that can be reached via the SINEMA RC Server are allowed. The NAT settings are implemented as configured in the SINEMA RC Server.
  - Disabled
    You will need to create the firewall and NAT rules yourself.
- Type of connection
  Specify the type of VPN connection. For more detailed information, refer to the section "VPN connection establishment".
  - Auto
    The device adopts the settings of the SINEMA RC Server. You configure the settings on the SINEMA RC Server in "Remote connections > Devices". You will find further information on this topic in the operating instructions "SINEMA RC Server".
  - Permanent
    The settings of the SINEMA RC Server are ignored. The device establishes a VPN connection to the SINEMA RC Server. The VPN tunnel is established permanently
  - Wake-up SMS (only with M87x)
    The settings of the SINEMA RC Server are ignored. When the device receives a command SMS message (wake-up SMS message), it attempts to establish a connection to the SINEMA RC Server. On condition that in "System > SMS > SMS Command" it is specified who a command SMS of the class "System" will be accepted from.
  - Digital Input
    The settings of the SINEMA RC Server are ignored. If the "Digital In" event occurs, the device attempts to establish a VPN connection to the SINEMA RC Server. This is on condition that the event "Digital Input" is forwarded to the VPN connection. To do this in "System > Events> Configuration" activate "VPN Tunnel" for the "Digital In" event.
  - Digital In & Wake-up SMS (only with M87x)
    The settings of the SINEMA RC Server are ignored. If the "Digital In" event occurs or when the device receives an SMS command, it attempts to establish a VPN connection to the SINEMA RC Server.
- Use Proxy
  Specify whether a connection to the defined SINEMA RC Server is established via a proxy server. Only the proxy servers can be selected that you configured in "System > Proxy Server".
- Autoconfiguration Interval [min]
  Specify the period of time in minutes after which queries are sent to the SINEMA RC Server. With this query, the device checks whether there is a newer firmware file on the SINEMA RC server or whether the connection settings have changed.
  If you enter the value 0, this function is disabled.

Configuring interfaces

Ethernet

Overview

The page shows the configuration for the data transfer for all ports of the device.

Description

- **Port**
  Shows the configurable ports.
- **Port name**
  Shows the name of the port.
- **Port Type** (only with routing)
  Shows the type of the port. The following types are possible:
  - Router port
  - Switch port VLAN hybrid
  - Switch port VLAN trunk
- **Status**
  Shows whether the port is on or off. Data traffic is possible only over an enabled port.
- **Operational State** (only available online)
  Displays the current operational status. The operational status depends on the configured "Status" and the "Link".
  The available options are as follows:
  - up
    You have configured the status "enabled" for the port and the port has a valid connection to the network.
  - down
    You have configured the status "disabled" or "Link down" for the port or the port has no connection.
- **Link** (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - up
    The port has a valid link to the network, a link integrity signal is being received.
  - down
    The link is down, for example because the connected device is turned off.
- **Mode** (only available online)
  Shows the transfer parameters of the port

- **Negotiation**
  Shows whether the automatic configuration is enabled or disabled.
- **MAC Address** (only available online)
  Shows the MAC address of the port.

Configuration

On this page, you configure the Ethernet ports of the device.

Description

- **Port**
  Select the port to be configured.
- **Status**
  Specify whether the port is enabled or disabled.
  - enabled
    The port is enabled. Data traffic is possible only over an enabled port.
  - disabled
    The port is disabled but the connection remains.
  - Link down
    The port is disabled and the connection to the partner device is terminated.
- **Port name**
  Enter a name for the port.
- **MAC Address** (only available online)
  Shows the MAC address of the port.
- **Mode Type**
  Shows the transmission speed and the transmission method of the port. If you set the mode to "Auto negotiation", these parameters are automatically negotiated with the connected end device or network component. This must also be in the "auto negotiation" mode.

  Note

  Before the port and partner port can communicate with each other, the settings must match at both ends.

- **Mode** (only available online)
  Shows the transmission speed and the transmission mode of the port. The display depends on the set "Mode Type".
- **Negotiation**
  Shows whether the automatic configuration of the connection to the partner port is enabled or disabled.

- **Port Type** (only with routing)
  Select the type of the port:
  - Router Port
    The port is an IP interface. It does not support layer 2 functions.
  - Switch-Port VLAN Hybrid
    The port sends tagged and untagged frames. It is not automatically a member of a VLAN.
  - Switch-Port VLAN Trunk
    The port only sends tagged frames and is automatically a member of all VLANs.
- **Operational State** (only available online)
  Displays the current operational status. The operational status depends on the configured "Status" and the "Link".
  The available options are as follows:
  - up
    You have configured the status "enabled" for the port and the port has a valid connection to the network.
  - down
    You have configured the status "disabled" or "Link down" for the port or the port has no connection.
- **Link** (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - Up
    The port has a valid link to the network, a link integrity signal is being received.
  - Down
    The link is down, for example because the connected device is turned off.

Mobile radio

SIM

For access to the mobile wireless network and the mobile wireless services, the following access parameters are necessary. You will receive the access parameters from your mobile wireless provider.

Description

The page contains the following:

- **Enable Mobile Network Interface**
  Enable or disable the mobile wireless interface.
- **PIN**
  Enter the PIN of the SIM card. You obtain the PIN from your mobile wireless provider. The device also works with SIM cards without a PIN, in this case leave the box empty.

  Note

  If you make incorrect entries, the SIM card is blocked

  Make sure that you enter the PIN correctly. If you enter the PIN incorrectly more than three times, the SIM card will be blocked.

  In "Information > Mobile", you can check the status of the SIM card. If "SIM Status" displays "PUK required", the SIM card is blocked.

  To release it, the card needs to be taken out of the device and inserted in a mobile wireless telephone. By entering the PUK, the SIM card can be released again. If necessary, contact your mobile wireless provider.

- **Pin confirmation**
  Repeat the PIN to confirm it.
- **Radio Mode**
  Select the required mobile wireless network. The following options are available:
  - Auto (not with M874-2)
    All services. As first choice, the device attempts to establish a connection to the fastest available mobile wireless system.
  - GSM only
    The EGPRS and GPRS services.
    The device ignores the UMTS and LTE network and establishes a connection to the GSM service that provides the highest bandwidth locally
  - UMTS only (not with M874-2)
    The UMTS and HSPA service. The device ignores the EGPRS, GPRS and LTE services and establishes a connection in the UMTS network.
  - LTE only (available only with the M876-4)
    The LTE service. The device ignores the EGPRS and GPRS and UMTS services and establishes a connection in the LTE network.
  - No data connection
    There is no data connection. The mode is suitable for the situation when the device only wants to send or receive SMS messages.

- **Authentication Method**
  Select the required authentication method.
  - CHAP
    Encrypted transfer of user name and password using the Challenge Handshake Authentication Protocol (CHAP).
  - PAP
    Unencrypted transfer of user name and password using the Password Authentication Protocol (PAP).
  - Auto
    User name and password are transferred automatically with one of the following two methods. CHAP has the higher priority. If the communications partner does not support CHAP, the user name and password are transferred using PAP.
- **Allow Data Roaming**
  When enabled, the device automatically logs in to an available network if the specified network is unreachable.

Mobile wireless provider

The APN (Access Point Name) is the name of the access point from the mobile wireless network to the Internet or to a private company network. Depending on the type of network connected, this is a public or private APN. Information about the APN is provided by the mobile wireless provider.

Description

The page contains the following:

- **Country List** (only available online)
  From this list, select the country in which the device will be deployed.
- **Provider List** (only available online)
  From this list select the appropriate mobile wireless provider. The list depends on the selected country.
  If the mobile wireless provider is not contained in the list of providers, select "-".
- **PLMNID**
  Each mobile wireless provider has an identification number that is unique worldwide known as the Public Land Mobile Network ID (PLMN ID). You will find the Net ID in the documentation of your mobile wireless provider or on their Internet pages.
  - If you know the identification number (Net ID) of the network provider, enter it.
  - If you do not know the PLMNID, enter "Manual".
- **APN**
  Enter the name of the APN. You will find the APN in the documentation of your mobile wireless provider on your provider's Website or ask your mobile wireless provider's hotline.
- **User Name**
  Enter the user name. Some mobile wireless providers do not use access control with user names and/or passwords. In this case, leave the box empty.

- Password
  Enter the password. Some mobile wireless providers do not use access control with user names and/or passwords. In this case, leave the box empty.
- Password Confirmation
  Repeat the password.

This table contains the following columns:

- PLMNID
  Shows the identification number (Net ID) of the mobile wireless provider.
- Operator Name
  Enter the name of your mobile wireless provider.
- APN
  Enter the name of the APN. You will find the APN in the documentation of your mobile wireless provider on your provider's Website or ask your mobile wireless provider's hotline.
- User Name
  Enter the user name for the APN. Some mobile wireless providers do not use access control with user names and/or passwords. In this case, leave the box empty.
- Password
  Enter the password for the APN. Some mobile wireless providers do not use access control with user names and/or passwords. In this case, leave the box empty.
- Password Confirmation
  Repeat the password.
- Enabled
  The entry is used

Connection Check

Use this function to ensure that the connection to the mobile wireless network is retained. On this page you specify which WAN addresses (IPv4 address or FQDN) are used as reference for the connection monitoring.

The device sends echo messages (pings) to the configured WAN addresses at regular intervals (Ping Time Interval).

If, for example, you configure three WAN addresses, the device sends a ping to the first WAN address. If this WAN address is reachable, it sends a reply. When the configured interval has elapsed, the device sends the next ping. If the first WAN address cannot be reached, the device sends a further ping after a waiting time has elapsed (10 s). In total three pings are sent to the WAN address. If the device does not receive a reply after the third ping, the second WAN address is checked. If this WAN address is also unreachable, the next WAN address is checked.

Is none of the three WAN addresses is reachable, the device restarts the mobile wireless interface and logs the restart with the following message "Detected a problem with the provider connection. Trying to reconnect to the provider. Attempt no.: x".

The x stands for the number of attempts. In total the device makes 10 attempts. After restarting the mobile wireless interface, the device waits for some time and then sends a ping to the first WAN address. Following the first restart of the mobile wireless interface the wait time is 2

minutes. With each further restart of the mobile wireless interface the wait time is extended by one minute.

If the 10 attempts are unsuccessful, the device restarts and logs the restart with the following message: "Detected a problem with the provider connection. Reconnect to the provider had no effect. Device will restart now."

Note

Always configure several WAN addresses. If you only configure one WAN address, the sporadic failure of the single WAN address can lead to the WAN connection being constantly re-established.

Description

The page contains the following:

- **Enable Connection Ping**
  Enable or disable the function. Can only be enabled if at least one WAN address is configured.
- **Ping Time Interval [s]**
  Specify the interval at which the pings are sent.
- **Ping Target Address**
  Enter the IPv4 address or FQDN address that will be used for the connection monitoring.

This table contains the following columns:

- **Target Address**
  Shows the WAN address.

DSL

On this page, you configure the DSL access and the parameters for the virtual connection via which the packets will be transferred.

Description

The page contains the following:

- **Enable DSL Interface**
  Enable or disable the DSL interface.
- **Enable PPPoE Passthrough**
  - Enabled
    The device acts as a modem. The settings "Account", "Password" and "Enable Force Disconnect" cannot be edited. Only a connected device can use the DSL connection. This device also handles the authentication (dial-in) with the provider.
  - Disabled
    The device acts as a router and logs in with the user name and password. All connected devices can use the DSL connection.

- User Account
  Enter the user name. You will receive the user name from your DSL provider.
- Password
  Enter the password. You will receive the password name from your DSL provider.
- Password (Confirmation)
  Repeat the password to confirm it.
- VCI (Virtual Channel Identifier)
  Enter the ID for the virtual channel. You will receive setting from your DSL provider.
- VPI (Virtual Path Identifier)
  Enter the ID for the virtual path. You will receive setting from your DSL provider.
- Encapsulation
  The data packets are encapsulated in the required protocol.
  The following options are available:
  - vc-mux (virtual circuit multiplexing)
  - LLC (Logical Link Control)
- Protocol
  Specify the protocol for the Internet connection. You obtain this information from your DSL provider.
  - PPPoE (Point-to-Point over Ethernet)
    The PPP data is encapsulated in an Ethernet frame.
  - PPPoA (Point-to-Point over ATM)
    The PPP data is encapsulated in ATM AAL5 (Adaptation Layer 5). If you select PPPoA, the setting "Enable PPPoE Passthrough" is disabled.
- Noise Margin Delta DS
  Specify the margin to the background noise.
  Range of values -5dB ... +5dB
- Enable Vlan ID
  When enabled, you can configure a VLAN ID.
- VLAN ID
  Enter the VLAN ID. You will receive setting from your DSL provider.
- Forced Disconnect
  After a certain time, the DSL provider terminates the connection. Enable this option if you want to shift the forced disconnect of your provider to a specific time of day, for example at night outside normal office hours.
- Time for Forced Disconnect
  Specify the time of day to which you want to shift the forced disconnect of the DSL provider. This is only possible if the correct system time is set on the device.
  Input format: HH:MM

SHDSL

Overview

This page shows the configuration of the SHDSL interfaces.

Description

The page contains the following:

- **Enable PME Aggregation Function**
  PME = Physical Medium Entities
  When enabled, the SHDSL interfaces or the 2-wire cables are put together to form a single connection with a higher transmission rate.

  Note

  If the "PME Aggregation" function is enabled, the interfaces of a device must each have the same role.

This table contains the following columns:

- **Interface**
  Shows the available SHDSL interfaces.
- **Status**
  Shows whether or not the interface is used.
- **Role**
  Shows the role of the SHDSL interface:
  - Central Office (CO)
    The interface is a central office.
  - Customer Premises Equipment (CPE)
    The interface is an end node device.
- **Target SNR**
  Shows the signal-to-noise ratio:
  - Reliability (10 dB)
    Reliable transfer
  - Normal (6 dB)
    Normal transfer
  - High-speed (0 dB)
    High-speed transfer
- **Port Type**
  Shows the operating mode of the switch port:
  - Switch-Port VLAN Hybrid
    The port accepts tagged and untagged frames.
  - Switch-Port VLAN Trunk
    The port only sends tagged frames and is automatically a member of all VLANs.

Configuration

On this page, you specify the settings for the connection and the role of the device. You will find further information on the settings in ITU-T G.991.2.

Description

The page contains the following:

- **Interface**
  Select the interface.
- **Status**
  Specify whether or not the interface will be used.
- **Port Type**
  Specify the operating mode for the switch port.
  - Switch-Port VLAN Hybrid
    The port accepts tagged and untagged frames.
  - Switch-Port VLAN Trunk
    The port only sends tagged frames and is automatically a member of all VLANs.
- **Role**
  Specify the role:
  - Central Office (CO)
    The interface is a central office.
  - Customer Premises Equipment (CPE)
    The interface is an end node device.
- **Predefined Profile**
  Specify the profile for the transfer. If you use a profile, the following parameters are set automatically.
  - Profiles with a transfer range
    If during calibration of the cable the measured value is within the range, the SHDSL connection is established.

	Link data rate min - max	
Standard	192 - 5696	For medium and longer cable lengths
Reliability	192 - 5696	When operating several SHDSL connections simultaneously.
Extended	64 - 15296	For short and long cable length

  - Profiles with a fixed preset transmission rate
    If during calibration of the cable the measured value precisely matches the fixed preset transmission rate, and SHDSL connection is established.

	Link Data Rate
Fixed (High Rate)	5696
Fixed (Medium Rate)	3072
Fixed (Low Rate)	512

- **Extended Mode**
  Enable or disable the extended mode. Depending on the setting, the selection in the following boxes changes.
- **PAM**
  Pulse Amplitude Modulation
  Specify the modulation method.
- **Line probing**
  When enabled, the cable is calibrated according to ITU-T G.991.2.
- **SNR Model**
  Specify the model for calculating the signal-to-noise ratio.
  - Current condition
    The value for noise is used that was measured when calibrating the cable.
  - Worst Case
    The value for noise is assumed that could occur over several cables of a cable bundle.
- **Target SNR**
  Specify the transmission mode for the signal-to-noise ratio. The current signal-to-noise ratio is displayed under "Information > SHDSL".
  - Reliability (10 dB)
    Reliable transfer
  - Normal (6 dB)
    Normal transfer
  - High-speed (0 dB)
    High-speed transfer
- **Min. Link Data Rate [kbps]**
  Specify the minimum transmission rate for sending and receiving data.
  - Auto (0)
    The device adapts the transmission rate automatically depending on the mode.
  - 64 to 15296
    The transmission rate is specified as a fixed value.
- **Max. Link Data Rate [kbps]**
  Specify the maximum transmission rate for sending and receiving data.
  - Auto (0)
    The device adapts the transmission rate automatically depending on the mode.
  - 64 to 15296
    The transmission rate is specified as a fixed value.

- **Power Regulation**
  Specify whether or not the transmit power is reduced.
  - Disabled
    The function is disabled
  - Normal
    The function is enabled.
  - Force
    For PBO Value, enter the value for the attenuation.
- **PBO Value**
  Power Back Off Values
  Enter the value for the attenuation. An SHDSL connection is only established when the value is reached.
  - Range: 0 - 31 dB, where "0" means no attenuation.

Connection Check

Use this function to ensure that the connection to the SHDSL network is retained. On this page you specify which WAN addresses (IP address or FQDN) are used as reference for the connection monitoring.

The device sends echo messages (pings) to the configured WAN addresses at regular intervals (ping interval).

If, for example, you configure three WAN addresses, the device sends a ping to the WAN addresses. If only one of the three WAN addresses is unreachable the following message is logged: "Could not reach remote device a.b.c.d (Failure count x)".

"a.b.c.d" stands for the IP address of the device and x stands for the number of attempts. In total the device makes 5 attempts.

If the WAN address becomes reachable again, this is logged with the following message: SHDSL connection check: Remote devices are reachable again.

If the 5 attempts are unsuccessful, the device restarts and logs the restart with the following message: "SHDSL connection check: Maximum failure count reached, restarting device."

Description

The page contains the following:

- **Enable Connection Ping**
  Enable or disable the function. Can only be enabled if at least one WAN address is configured.
- **Ping Time Interval [s]**
  Specify the interval at which the pings are sent.
- **Ping Target Address**
  Enter the IPv4 address or FQDN address that will be used for the connection monitoring.

This table contains the following columns:

- **Target Address**
  Shows the WAN address.

Configuring layer 2 functions

Configuration

Configuring layer 2

On this page, you configure passive listening.

Description

- Passive Listening
  When enabled the function ensures that the BPDUs from the RSTP network are forwarded transparently and return again. If this was not the case, loops would form at the connection point between RSTP and the ring.

VLAN

General

On this page you specify whether or not the device forwards frames with VLAN tags transparently (IEEE 802.1D/VLAN-unaware mode) or takes VLAN information into account (IEEE 802.1Q/VLAN-aware mode). If the device is in the "802.1Q VLAN Bridge" mode, you can define VLANs and specify the use of the ports .

Note

Changing the agent VLAN ID

If the configuration PC is connected directly to the device via Ethernet and you change the agent VLAN ID, the device is no longer reachable via Ethernet following the change.

Description

The page contains the following boxes:

- **Base Bridge-Mode**
  Select the required mode. The following modes are possible:

  Note

  Changing Base Bridge Mode

  Note the section "Changing Base Bridge Mode". This section describes how a change affects the existing configuration.

  – 802.1 D Transparent Bridge
    Sets the mode "VLAN-unaware" for the device. In this mode, VLAN tags are not taken into account or changed but are forwarded transparently. In this mode, you cannot create any VLANs.
  – 802.1Q VLAN Bridge
    Sets the mode "VLAN-unaware" for the device. In this mode, VLAN tags are not taken into account or changed but are forwarded transparently. In this mode, you cannot create any VLANs.
    Only a management VLAN is available: VLAN 1.

- **VLAN ID**
  Enter the VLAN-ID (a number between 1 and 4094).
  You can create up to 2 (SCALANCE W760), 8 (SCALANCE W770) or 24 (SCALANCE W780) VLANS.

The table has the following columns:

- **VLAN ID**
  Shows the VLAN ID. The VLAN ID (a number between 1 and 4094) can only be assigned once when creating a new data record and can then no longer be changed. To make a change, the entire data record must be deleted and created again.
- **Name**
  Enter a name for the VLAN. The name only provides information and has no effect on the configuration.

- **Status**
  Shows the status type of the entry in the internal port filter table. Here, static means that the address was entered as a static address by the user. The entry GVRP means that the configuration was registered by a GVRP frame. This is, however, only possible if GVRP was enabled for the device.
- **List of ports**
  Specify the use of the port. The following options are available:
  - "-"
  - The port is not a member of the specified VLAN.
    With a new definition, all ports have the identifier "-".
  - M
    The port is a member of the VLAN. Frames sent in this VLAN are forwarded with the corresponding VLAN tag.
  - R
    The port is a member of the VLAN. A GVRP frame is used for the registration.
  - U (upper case)
    The port is an untagged member of the VLAN. Frames sent in this VLAN are forwarded without the VLAN tag. Frames without a VLAN tag are sent from this port.
  - u (lower case)
    The port is an untagged member of the VLAN, but the VLAN is not configured as a port VLAN. Frames sent in this VLAN are forwarded without the VLAN tag.
  - F
    The port is not a member of the specified VLAN and it is not possible for the VLAN to be registered dynamically at this port using GVRP. You can configure other settings in "Layer 2 > VLAN > Port-based VLAN".
  - T
    This option is only displayed and cannot be selected. This port is a trunk port, making it a member in all VLANs. You configure this function in the CLI (Command Line Interface) using the "`switchport mode trunk`" command or in the WBM under "Interfaces > Ethernet > Configuration".

Changing Base Bridge Mode

VLAN-unaware (802.1D transparent bridge) → VLAN-aware (802.1Q VLAN bridge)

If you change the Base Bridge Mode from VLAN-unaware to VLAN aware, this has the following effects:

- All static and dynamic unicast entries are deleted.

VLAN-aware (802.1Q VLAN bridge) → VLAN-unaware (802.1D transparent bridge)

If you change the Base Bridge Mode from VLAN-aware to VLAN-unaware, this has the following effects:

- All VLAN configurations are deleted.
- A management VLAN is created: VLAN 1.
- All static and dynamic unicast entries are deleted.

802.1Q VLAN Bridge: Important rules for VLANs

Make sure you keep to the following rules when configuring and operating your VLANs:

- Frames with the VLAN ID "0" are handled as untagged frames but retain their priority value.
- As default, all ports on the device send frames without a VLAN tag to ensure that the end node can receive these frames.
- You can find the factory assignment of the ports in the section "VLAN (Page 2559)".
- If an end node is connected to a port, outgoing frames should be sent without a tag (static access port). If, however, there is a further switch at this port, the frame should have a tag added (trunk port).

Port-based VLAN

Processing received frames

On this page, you specify the configuration of the port properties for receiving frames.

Description of the displayed boxes

Table 1 has the following columns:

- **Port**
  Shows that the settings are valid for all ports.
- **Priority / Port VID / Acceptable Frames / Ingress Filtering**
  Select the setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports and link aggregations. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Priority**
  Select the priority assigned to untagged frames.
  The CoS priority (Class of Service) used in the VLAN tag. If a frame is received without a tag, it will be assigned this priority. This priority specifies how the frame is further processed compared with other frames. There are a total of eight priorities with values 0 to 7, where 7 represents the highest priority (IEEE 802.1p Port Priority).
- **Port VID**
  Select the VLAN ID. Only VLAN IDs defined on the "VLAN > General" page can be selected. If a received frame does not have a VLAN tag, it has a tag with the VLAN ID specified here added to it and is sent according to the rules at the port.

- Acceptable Frames
  Specify which types of frames will be accepted. The following alternatives are possible:
  - Tagged Frames Only
    The device discards all untagged frames. Frames tagged with "0" are treated like untagged frames. The device forwards all tagged frames. Otherwise, the forwarding rules apply according to the configuration.
  - Untagged and Priority Tagged Only
    The device discards all tagged frames. The device forwards all untagged frames and frames with a priority (Priority Tagged Frames). Otherwise, the forwarding rules apply according to the configuration. If you have configured the Bridge mode "Provider" this means that the device treats all incoming frames as untagged frames.
  - All
    The device forwards all frames
- Ingress Filtering
  Specify whether the VID of received frames is evaluated. The following options are available:
  - Enabled
    The VLAN ID of received frames decides whether they are forwarded: To forward a VLAN tagged frame, the receiving port must be a member in the same VLAN. Frames from unknown VLANs are discarded at the receiving port.
  - Disabled
    All frames are forwarded.

Dynamic MAC aging

The device automatically learns the source addresses of the connected nodes.

This information is used to forward data frames to the nodes specifically involved. The network load for the other nodes is reduced.
If a device does not receive a frame whose source address matches a learnt address within a certain time, the learnt address is deleted. This mechanism is known as aging. Aging prevents frames being forwarded incorrectly, for example when an end device (for example a programming device) is connected to a different port. If the check box is not enabled, a device does not delete learned addresses automatically.

Description

- Dynamic MAC Aging
  Enable or disable the function for automatic aging of learned MAC addresses.
- Aging Time[s]
  Enter the time in seconds in steps of 15. After this time, a learned address is deleted if the device does not receive any further frames from this sender address.

Note

Rounding of the values, deviation from desired value

When you input the Aging Time, note that it is rounded to correct values. If you enter a value that cannot be divided by 15, the value is automatically rounded down.

Spanning Tree

General

This is the basic page for spanning tree. As default, Rapid Spanning Tree is enabled.

Description

The page contains the following boxes:

- **Spanning Tree**
  Enable or disable spanning tree.
- **Protocol Compatibility**
  The following setting is available:
  – RSTP

ST general

The page consists of the following parts.

- The left-hand side of the page shows the configuration of the device.

Description

The page contains the following boxes:

- **Bridge Priority / Root Priority** (only available online)
  Which device becomes the root bridge is decided by the bridge priority. The bridge with the highest priority (in other words, with the lowest value for this parameter) becomes the root bridge. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames. The value for the bridge priority is a whole multiple of 4096 with a range of values from 0 to 32768.
- **Bridge Address** (only available online)/ **Root Address** (only available online)
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Port** (only available online)
  Shows the port via which the switch communicates with the root bridge.
- **Root Cost** (only available online)
  The path costs from this device to the root bridge.

- **Topology Changes** (only available online) **/ Last Topology Change** (only available online)
  The entry for the device shows the number of reconfigurations due to the spanning tree mechanism since the last startup. For the root bridge, the time since the last reconfiguration is displayed as follows:
  - Seconds: Unit "sec" after the number
  - Minutes: Unit min after the number
  - Hours: Unit hr after the number
- **Bridge hello time [s] / Root hello time [s]** (only available online)
  Each bridge sends configuration frames (BPDUs) regularly. The interval between two configuration frames is the "Hello Time".
- **Bridge Forward Delay[s] / Root Forward Delay[s]** (only available online)
  New configuration data is not used immediately by a bridge but only after the period specified in the "Forward Delay" parameter. This ensures that operation is only started with the new topology after all the bridges have the required information.
- **Bridge Max Age[s] / Root Max Age[s]** (only available online)
  If the BPDU is older than the specified "Max Age" it is discarded.
- **Reset Counter** (only available online)
  Click this button to reset the counters on this page.

ST Port

When the page is called, the table displays the current status of the configuration of the port parameters.

To configure them, click the relevant cells in the port table.

Description

Table 1 has the following columns:

- **All ports**
  Shows that the settings are valid for all ports of table 2.
- **Spanning Tree Status**
  In the drop-down list, select the setting for all ports. If "No Change" is selected, the entries of the corresponding column in table 2 remain unchanged.
- **Copy to Table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Spanning Tree Status**
  Specify whether or not the port is integrated in the spanning tree.

  Note

  If you disable the "Spanning Tree Status" option for a port, this may cause the formation of loops. The topology must be kept in mind.

- **Priority**
  Enter the priority of the port. The priority is only evaluated when the path costs are the same. The value must be divisible by 16. If the value that cannot be divided by 16, the value is automatically adapted.
  Range of values: 0 - 240.
  The default is 128.
- **Cost Calc.**
  Enter the path cost calculation. If you enter the value "0" here, the automatically calculated value is displayed in the "Path costs" box.
- **Path Cost** (only available online)
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected as the path. If several ports of a device have the same value for the path costs, the port with the lowest port number is selected.
  If the value in the Cost Calc." is "0", the automatically calculated value is shown. Otherwise, the value of the "Cost Calc." box is displayed.
  The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000

  The values can, however, also be set individually.

- **Status** (only available online)
  Displays the current status of the port. The values are only displayed and cannot be configured. The "Status" parameter depends on the configured protocol. The following values are possible:
  - Disabled
    The port only receives and is not involved in STP, MSTP and RSTP.
  - Discarding
    In the "Discarding" mode, BPDU frames are received. Other incoming or outgoing frames are discarded.
  - Listening

    In this status, BPDUs are both received and sent. The port is involved in the spanning tree algorithm.
  - Learning
    Stage prior to the "Forwarding" status, the port is actively learning the topology (in other words, the node addresses).
  - Forwarding
    Following the reconfiguration time, the port is active in the network; it receives and forwards data frames.
- **Fwd. Trans** (only available online)
  Specifies the number of changes from the "Discarding" status to the "Forwarding" status.
- **Edge Type**
  Specify the type of "edge port". You have the following options:
  - "-"
    Edge port is disabled. The port is treated as a "no Edge Port".
  - Admin
    Select this option when there is always an end device on this port. Otherwise a reconfiguration of the network will be triggered each time a connection is changed.
  - Auto
    Select this option if you want a connected end device to be detected automatically at this port. When the connection is established the first time, the port is treated as a "no Edge Port".
  - Admin/Auto
    Select these options if you operate a combination of both on this port. When the connection is established the first time, the port is treated as an "Edge Port".

- **Edge** (only available online)
  Shows the status of the port.
  - Enabled

    An end device is connected to this port.
  - Disabled
    There is a Spanning Tree or Rapid Spanning Tree device at this port.

  With an end device, a switch can change over the port faster without taking into account spanning tree frames. If a spanning tree frame is received despite this setting, the port automatically changes to the "Disabled" setting.
- **P.t.P. Type**
  Select the required option from the drop-down list. The selection depends on the port that is set.
  - "-"
    Point to point is calculated automatically. If the port is set to half duplex, a point-to-point link is not assumed.
  - P.t.P.

    Even with half duplex, a point-to-point link is assumed.
  - Shared Media
    Even with a full duplex connection, a point-to-point link is not assumed.

    Note

    Point-to-point connection means a direct connection between two devices. A shared media connection is, for example, a connection to a hub.
- **P.t.P.** (only available online)
  - Enabled
    Shows that a point-to-point link exists.
  - Disabled
    Shows that no point-to-point link exists.

LLDP

Identifying the network topology

LLDP (Link Layer Discovery Protocol) is defined in the IEEE 802.AB standard.

LLDP is a method used to discover the network topology. Network components exchange information with their neighbor devices using LLDP.

Network components that support LLDP have an LLDP agent. The LLDP agent sends information about itself and receives information from connected devices at periodic intervals. The received information is stored in the MIB.

Applications

PROFINET uses the LLDP protocol for topology diagnostics. In the default setting, LLDP is enabled for all ports; in other words, LLDP frames are sent and received on all ports. With this function, you have the option of enabling or disabling sending and/or receiving per port.

Description

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports. The port is made up of the module number and the port number, for example port 0.1 is module 0, port 1.
- **Setting**
  Select whether or not the port will send or receive LLDP frames. The following options are available:
  - Rx
    This port can only receive LLDP frames.
  - Tx
    This port can only send LLDP frames.
  - Rx & Tx
    This port can receive and send LLDP frames.
  - "-" (disabled)
    This port can neither receive nor send LLDP frames.

Configuring layer 3 functions for IPv4

Static routes

Static route

On this page you specify the routes via which a data exchange can take place between the various subnets. Dynamic routing protocols are not supported, for example RIP, OSPF.

Description

The page contains the following boxes:

- Destination Network
  Enter the network address of the destination that can be reached via this route.
- Subnet Mask
  Enter the corresponding subnet mask.
- Gateway
  Enter the IPv4 address of the gateway via which this network address is reachable.
- Interface
  Select the interface of the route.
- Administrative Distance
  Enter the administrative distance for the route. The administrative distance corresponds to the quality of a connection, for example speed, costs. If there are several equal routes, the route with the lowest value is used.
  If you do not enter anything, "not used" is entered automatically. The value for the administrative distance can be changed later.
  Range of values: 1 - 254 or -1 for "not used". Here, 1 is the value for the best possible route.

The table has the following columns:

- Destination Network
  Shows the network address of the destination.
- Subnet Mask
  Shows the corresponding subnet mask.
- Gateway
  Shows the IPv4 address of the next gateway.
- Interface (only available online)
  Shows the Interface of the route.
- Administrative Distance
  Enter the administrative distance for the route.
- State (only available online)
  Shows whether or not the route is active.

Subnets

Overview

The page shows the subnets for the selected interface. A subnet always relates to an interface and is created in the "Configuration" tab.

Description

The page contains the following box:

- Interface
  Select the interface on which you want to configure another subnet.

The table has the following columns:

- Interface
  Shows the interface.
- TIA Interface
  Shows the selected TIA interface. The TIA interface defines the VLAN on which the PROFINET functionalities are available. This mainly affects the device search with or via DCP.
- Interface Name
  Shows the name of the interface.
- MAC Address
  Shows the MAC address.
- IP Address
  Shows the IPv4 address of the subnet.
- Subnet Mask
  Shows the subnet mask.
- Address Type
  Shows the address type. The following values are possible:
  - Primary
    The first IPv4 address that was configured on the interface.
  - Secondary
    All other IPv4 addresses that were configured on the interface.
- IP Assignment Method
  Shows how the IPv4 address is assigned. The following values are possible:
  - Static
    The IPv4 address is static. You enter the settings in "IP Address" and "Subnet Mask".
  - Dynamic (DHCP)
    The device obtains a dynamic IPv4 address from a DHCPv4 server.

- **Address Collision Detection Status** (only available online)
  If new IPv4 addresses become active in the network, the "Address Collision Detection" function checks whether this can result in address collisions. The allows IPv4 addresses that would be assigned twice to be detected.

  Note

  The function does not run a cyclic check.

  This column shows the current status of the function. The following values are possible:

  – Idle
    The interface is not enabled and does not have an IPv4 address.
  – Starting
    This status indicates the start-up phase. In this phase, the device initially sends a query as to whether the planned IPv4 address already exists. If the address is not yet been assigned, the device sends the message that it is using this IP address as of now.
  – Conflict
    The interface is not enabled. The interface is attempting to use an IPv4 address address that has already been assigned.
  – Defending
    The interface uses a unique IPv4 address. Another interface is attempting to use the same IPv4 address.
  – Active
    The interface uses a unique IPv4 address. There are no collisions.
  – Not supported
    The function for detection of address collisions is not supported.
  – Disabled
    The function for detection of address collisions is disabled.
- **MTU**
  Shows the packet size.

Configuration

On this page, you configure the subnet for the interface.

Description

The page contains the following:

- **Interface (Name)**
  Select the desired interface.
- **Interface Name**
  Enter the name of the interface.
- **MAC Address** (only available online)
  Displays the MAC address of the selected interface.
- **DHCP**
  Enable or disable the DHCP client for this IPv4 interface.

- **IP Address**
  Enter the IPv4 address of the interface. The IPv4 addresses must not be used more than once.
- **Subnet Mask**
  Enter the subnet mask of the subnet you are creating. Subnets on different interfaces must not overlap.
- **Broadcast IP Address**
  If a specific IP address is to be used as the broadcast IP address of the subnet, enter this. Otherwise the last IP address of the subnet will be used.
- **Address Type**
  Shows the address type. The following values are possible:
  - Primary
    The first subnet of the interface.
  - Secondary
    All further subnets of the interface.
- **TIA Interface**
  Enable or disable the setting.
  The following conditions apply to the TIA interface:
  - Only interfaces with the address type "Primary" can be enabled as the TIA interface.
  - There must only ever be one TIA interface.
  - There can only be one TIA interface.
  - A TIA interface is always a VLAN interface.
- **MTU**
  MTU (Maximum Transmission Unit) specifies the maximum size of the packet. If packets are longer than the set MTU, they are fragmented.

NAT

Masquerading

On this page, you enable the rules for IP masquerading (Page 2570).

Description

The table has the following columns:

- **Interface**
  Interface to which the setting relates. Only interfaces with a configured subnet are available.
- **Enable Masquerading**
  When enabled, with each outgoing data packet sent via this interface, the source IP address is replaced by the IP address of the interface.

NAPT

On this page, you configure the port forwarding (Page 2570).

Description

The page contains the following boxes:

- **Source Interface**
  Select the interface on which the queries will arrive.
- **Traffic Type**
  Specify the protocol for which the address assignment is valid.
- **Use Interface IP from Source Interface**
  When enabled, the IP address of the selected interface is used for "Dest IP Address".
- **Destination IP Address**
  Enter the destination IP address. The frames are received at this IP address. Can only be edited if "Use Interface IP from Source Interface" is disabled.
- **Destination Port**
  Enter the destination port. Incoming frames with this port as the destination port are forwarded. If the setting is intended to apply to a port range, enter the range with start port "-" end port, for example 30 - 40.
- **Translated Destination IP**
  Enter the IP address of the node to which this frame will be forwarded.
- **Translated Destination Port**
  Enter the number of the port. This is the new destination port to which the incoming frame will be forwarded. If the setting is intended to apply to a port range, enter the range with start port "-" end port, for example 30 - 40.
  If the "Destination Port" and the "Translated Destination Port" are the same, the frames will be forwarded without port translation.

  Note

  If the port is already occupied by a local service, for example Telnet, a warning is displayed. Make sure that you avoid using the following ports:
  - Telnet / SSH: TCP port 23
  - HTTP: TCP port 80
  - HTTPS: TCP port 443
  - SNMP: UDP port 161
  - ISAKMP: UDP port 500
  - IPsec Nat-T: UDP port 4500
  - NTP: UDP port 123

The table has the following columns:

- **Source Interface**
  Shows the interface from which the packets need to come. Only these packets are considered for port forwarding.
- **Traffic Type**
  Shows the protocol for which the address assignment applies.

- **Interface IP**
  Shows whether the IP address of the interface is used.
- **Destination IP**
  Shows the destination IP address. The frames are received at this IP address.
- **Destination Port**
  Shows the destination port. Incoming frames with this port as the destination port are forwarded.
- **Translated Destination IP**
  Shows the IP address of the node to which the packets will be forwarded.
- **Translated Destination Port**
  Shows the destination port to which the packets are translated.

Source NAT

On this page, you configure the rules for source NAT (Page 2570).

Note

Firewall rules with source NAT

If you create a firewall rule for a source NAT rule use the entry from "Source IP Subnet" in "IP Rules" for the "Source (Range)". And for "Destination (Range)" use the entry from "Destination IP Subnet".

Description

- **Source Interface / Destination Interface**
  Specify the direction of the connection establishment. Only connections established in this specified direction are taken into account.
  The virtual interfaces of VPN connections can also be selected:
  - VLANx: VLANs with configured subnet
  - ppp0 or usb0 (only with M876-4): WAN interface
  - SINEMA RC: Connection to SINEMA RC Server
  - IPsec:Either all IPsec VPN connections (all) or a specific IPsec VPN connection
  - OpenVPN: Either all OpenVPN connections (all) or a specific OpenVPN connection

  Note

  When you configure a NAT address translation to or from the direction of the VPN tunnel, only the IP addresses involved in the NAT address translation rules can be reached via the VPN tunnel.

- **Source IP Address(es)**
  Specify the source IP addresses for which this source NAT rule is valid. Only the packets that correspond to the addresses entered are taken into account.
  The following entries are possible:
  - IP address: Applies precisely to the specified IP address.
  - IP address range: Applies to a certain IP address range: Start IP address "-" End IP address, e.g. 192.168.100.10 - 192.168.100.20
  - IP subnet: Applies to several IPv4 addresses grouped together to form an IP address range: IP address/number of bits of the network part (CIDR notation)
- **Use Interface IP from Destination Interface**
  When enabled, the IP address of the selected destination interface is used in "Translated Source IP Address".
- **Translated Source IP Address**
  Enter the IP address with which the IP address of the sender is replaced. Can only be edited if "Use Interface IP from Destination Interface" is disabled.
- **Destination IP Address(es)**
  Specify the destination IP addresses for which this source NAT rule is valid. Only the packets whose destination IP address is in the range of entered addresses are taken into account.
  - IP address: Applies precisely to the specified IP address.
  - IP address range: Applies to a certain IP address range: Start IP address "-" End IP address, e.g. 192.168.100.10 - 192.168.100.20
  - IP subnet: Applies to several IPv4 addresses grouped together to form an IP address range: IP address/number of bits of the network part (CIDR notation)

The table has the following columns:

- Source Interface
  Shows the source interface.
- Destination Interface
  Shows the destination interface.
- Source IP Address(es)
  Shows the IP addresses of the senders for which address translation is required.
- Translated Source IP Address
  Shows the IP address with which the IP address of the sender is replaced.
- Destination IP Address(es)
  Shows the IP addresses of the recipients for which address translation is required.

NETMAP

On this page, you specify the rules for NETMAP (Page 2570). NETMAP is static 1:1 mapping of network addresses in which the host part is retained. For more information, refer to the section "NAT and firewall (Page 2571)".

Note

Firewall rules with source NAT

If you create a firewall rule for a source NAT rule, use the entry from "Source IP Subnet" in "IP Rules" for the "Source (Range)". And for "Destination (Range)" use the entry from "Destination IP Subnet".

Firewall rules with destination NAT

If you create a firewall rule for a destination NAT rule, use the entry from "Translated Destination IP Subnet" in "IP Rules" for the "Source (Range)". And for "Destination (Range)" use the entry from "Destination IP Subnet".

Description

- **Type**
  Specify the type of address translation.
  - Source: Replacement of the source IP address
  - Destination: Replacement of the destination IP address
- **Source Interface**
  Specify the source interface.
  - VLANx: VLANs with configured subnet
  - ppp0 or usb0 (only with M876-4): WAN interface
  - SINEMA RC: Connection to SINEMA RC Server
  - IPsec: Either all IPsec VPN connections (all) or a specific IPsec VPN connection
  - OpenVPN: Either all OpenVPN connections (all) or a specific OpenVPN connection

- **Destination Interface**
  Specify the destination interface.
  - VLANx: VLANs with configured subnet
  - ppp0 or usb0 (only with M876-4): WAN interface
  - SINEMA RC: Connection to SINEMA RC Server
  - IPsec: Either all IPsec VPN connections (all) or a specific IPsec VPN connection
  - OpenVPN: Either all OpenVPN connections (all) or a specific OpenVPN connection
- **Source IP Subnet**
  Enter the subnet of the sender.
  The subnet can also be a single PC or another subset of the subnet. Use the CIDR notation.
- **Translated Source IP Subnet**
  Enter the subnet with which the subnet of the sender will be replaced. Can only be edited in the "SourceNAT" settings.
  The subnet can also be a single PC or another subset of the subnet. Use the CIDR notation.
- **Destination IP Subnet**
  Enter the subnet of the recipient.
  The subnet can also be a single PC or another subset of the subnet. Use the CIDR notation.
- **Translated Destination IP Subnet**
  Enter the subnet with which the subnet of the recipient will be replaced. Can only be edited in the "DestinationNAT" settings.
  The subnet can also be a single PC or another subset of the subnet. Use the CIDR notation.

The table has the following columns:

- Type
  Shows the direction of the address translation.
- Source Interface
  Shows the source interface.
- Destination Interface
  Shows the destination interface.
- Source IP Subnet
  Shows the subnet of the sender.
- Translated Source IP Subnet
  Shows the subnet of the sender with which the subnet of the sender is replaced.
- Destination IP Subnet
  Shows the subnet of the recipient.
- Translated Destination IP Subnet
  Shows the subnet of the recipient with which the subnet of the recipient is replaced.

VRRPv3

Router

Introduction

On this page, you create new virtual routers. You can configure other parameters in "Layer 3 (IPv4) > VRRPv3 > Configuration".

Note

- You can use VRRPv3 on VLAN interfaces. Router ports are not supported.

Requirement

For the incoming VRRP packets to be forwarded to the device, you must configure the following firewall rule:

Security > Firewall > IP Protocol:

- Protocol Name: "VRRP"
- Protokollnummer: 112

Security > Firewall > IP Rules:

- Protocol: IPv4
- Action: Accept
- From: <Interface>
- To: Device
- Source (Range): 0.0.0.0/0
- Destination (Range): 224.0.0.18/32
- Services: VRRP

Description

- **VRRPv3**
  Enable or disable routing using VRRPv3.
- **Reply to pings on virtual interfaces**
  When enabled, the virtual IPv4 addresses also reply to the ping.
- **Track**
  Enable or disable VRID tracking. When enabled, all VRRP instances are monitored.
  If the status of a VRRP instance changes to "Initialize", the priority of all VRRP instances is reduced to the value "1".
  If the status of a VRRP instance changes, the original priority of all VRRP instances is restored.

- Interface
  Select the required VLAN interface operating as virtual router.
- VRID
  Enter the ID of the virtual router. This ID defines the group of routers that form a virtual router (VR). In the group, this is the same. It can no longer be used for other groups.

The table has the following columns:

- Interface
  Shows the IPv4 interface that functions as the virtual router.
- VRID
  Shows the ID of the virtual router.
- Virtual MAC Address (only available online)
  Shows the virtual MAC address of the virtual router.
- Primary IP Address
  Shows the numerically lowest IPv4 address in this VLAN. The entry 0.0.0.0 means that the "Primary" address on this VLAN is used. Otherwise all IPv4 addresses configured on this VLAN in the "Layer 3 > Subnets" menu are valid values.
- Router State (only available online)
  Shows the current status of the virtual router. Possible values are:
  - Master
    The router is the master router and handles the routing functionality for all assigned IPv4 addresses.
  - Backup
    The router is the backup router. If the master router fails, the backup router takes over the tasks of the master router.
  - Initialize
    The virtual router has just been turned on. It will soon change to the "Master" or "Backup" state.
- Master IP Address (only available online)
  Shows the IPv4 address of the master router.
- Priority (only available online)
  Shows the priority of the virtual router.
  If an IPv4 address is assigned to the VRRP router that is also actually configured on the local IPv4 interface, the value 255 is entered automatically. All other priorities can be distributed freely among the VRRP routers. The higher the priority, the earlier the VRRP router becomes "Master".
- Advert. Interval
  Shows the interval at which the master router sends VRRPv3 packets.
- Preempt
  Shows the precedence of a router when changing roles between backup and master.
  - yes
    This router has precedence when changing roles.
  - no
    This router does not have precedence when changing roles.

VRRP and DHCP server

If you want to operate a DHCP server on the devices of a VRRP group, the DHCP server must be configured on the master router. Backup routers do not react to DHCP queries. Make sure that the master router is statically configured and that after a failure, becomes the master of the VRRP group again.

Configuration

Introduction

On this page, you configure the virtual router.

Description

The page contains the following:

- Interface / VRID
  Select the ID of the virtual router to be configured.
- Primary Address
  Select the primary IPv4 address. If the router becomes master router, the router uses this IPv4 address.

  Note

  If you only configure one subnet on this VLAN, no entry is necessary. The entry is then 0.0.0.0.
  If you configure more than one subnet on the VLAN and you want a specific IPv4 address to be used as the source address for VRRP packets, select the IPv4 address. Otherwise, the numerically lowest IPv4 address will be used.

- Master
  If enabled, the numerically lowest IPv4 address is entered for "Associated IP Address". This means that the numerically lowest IPv4 address of the VRRP router is used as the virtual IP address of the virtual master router. The backup routers in this group must disable the option and use the IPv4 address of the router in "Associated IP address".
- Priority
  Enter the priority of this virtual router. Valid values are 1-254.
  If an IPv4 address is assigned to the VRRPv3 router that is also actually configured on the local IPv4 interface, the value 255 is entered automatically. All other priorities can be distributed freely among the VRRPv3 routers. The higher the priority, the earlier the VRRPv3 router becomes "Master".
- Advertisement interval
  Enter the interval in seconds after which a master router sends a VRRPv3 packet again.
- Preempt lower priority Master.
  Allow precedence when changing roles between backup and master based on the selection process.

- VRRP Compatible Mode
  When enabled, the VRRPv3 router also sends and receives VRRPv2 frames in addition to VRRPv3 frames for configured IPv4 addresses. Only necessary when some VRRP routers do not support VRRPv3
- Track ID
  Select a track ID.
- Decrement Priority
  Enter the value by which priority of the VRRP interface will be reduced.
- Current Priority
  Shows the priority of the VRRP interface after the monitored interface has changed to the "down" status.

Addresses Overview

Overview

This page shows which IPv4 addresses the virtual router monitors. Each virtual router can monitor a maximum of 10 IPv4 addresses.

Settings

The table has the following columns:

- Interface
  Shows the Interface that functions as the virtual router.
- VRID
  Shows the ID of this virtual router.
- Number of Addresses
  Shows the number of IPv4 addresses.
- Associated IP Address (1) ...Associated IP Address (10)
  Shows the router IPv4 addresses monitored by this virtual router. If a router takes over the role of master, the routing function is taken over by this router for all these IPv4 addresses.

Addresses Configuration

Creating or changing the monitored IP addresses

On this page, you can create, modify or delete the IPv4 addresses to be monitored.

Settings

The page contains the following:

- Interface / VRID
  Select the ID of the virtual router.
- Associated IP Address
  Enter the IPv4 address that the virtual router will monitor.

The table has the following columns:

- Associated IP Address
  Shows the IPv4 addresses that the virtual router monitors.

Interface Tracking

Introduction

On this page, you configure the monitoring of interfaces.

When the link of a monitored interface changes from "up" to "down", the priority of the assigned VRRP interface is reduced. You configure the value by which the priority is reduced on the page "Layer 3 > VRRP/VRRPv3 > Configuration".

When the link of the interface changes back from "down" to "up", the original priority of the VRRP interface is restored.

Description

The page contains the following boxes:

- Interface
  Select the required interface to be monitored.
- Track ID
  Enter a track ID.
- Track ID
  Select a track ID.
- Track Interface Count
  Enter how many monitored interfaces need to change to the "down" status, before the priority is changed.

The table has the following columns:

- Track ID
  Shows the track ID.
- Interface
  Shows the interface that is being monitored.

Configuring security functions

Users

Local users

Note

The page is only available with some devices if there is an online connection to the device.

Local users

On this page, you create local users for WBM and CLI with the corresponding rights.

When you create or delete a local user this change is also made automatically in the table "External User Accounts". If you want to make change explicitly for the internal or external user table, use the CLI commands.

Note

The values displayed depend on the rights of the logged-in user.

Description

The page contains the following:

- User Account
  Enter the name for the user. The name must meet the following conditions:
  - It must be unique.
  - It must be between 1 and 250 characters long.
  - The following characters must not be included: § ? " ; :

  Note
  User name cannot be changed

  After creating a user, the user name can no longer be modified.

  If a user name needs to be changed, the user must be deleted and a new user created.

  Note
  User names: admin, user, service, debug

  When shipped the following user names are predefined in the factory. user, admin, service, debug.
  - admin: You can configure the device with this user name. If you log on the first time or log on after a "Restore Factory Defaults and Restart", you will be prompted to change the predefined password "admin".
  - user, service, debug: These user names are reserved for service purposes.

- Password Policy
  Shows which password policy is being used.
  - High
    Password length: at least 8 characters, maximum 128 characters
    at least 1 upper case letter
    at least 1 special character
    at least 1 number
  - Low
    Password length: at least 6 characters, maximum 128 characters
- Password Confirmation
  Enter the password again to confirm it.
- Role
  Select a role.
  You can choose between default and self-defined roles, refer to the page "Security > Users > Roles.".

The table contains the following columns:

Note

The preset users as well as logged in users cannot be deleted or changed.

- User Account
  Shows the user name.
- Role
  Shows the role of the user.
- Description
  Displays a description of the user account. The description text can be up to 100 characters long.

Roles

Note

The page is only available if there is an online connection to the device.

Roles

On this page, you create roles that are valid locally on the device.

Note

The values displayed depend on the rights of the logged-in user.

Description of the displayed boxes

The page contains the following:

- Role Name
  Enter the name for the role. The name must meet the following conditions:
  – It must be unique.
  – It must be between 1 and 64 characters long.

  Note
  Role name cannot be changed
  After creating a role, the name of the role can no longer be changed.
  If a name of a role needs to be changed, the role must be deleted and a new role created.

This table contains the following columns:

Note
Deleting roles

Predefined roles and assigned roles cannot be deleted or modified.

- **Role**
  Shows the name of the role.
- **Function Right**
  Select the function rights of the role.
  - 1
    Users with this role can read device parameters but cannot change them. Users with this role can change their own password.
  - 15
    Users with this role can both read and change device parameters.

  Note

  Function right cannot be changed

  If you have assigned a role, you can no longer change the function right of the role.

  If you want to change the function right of a role, follow the steps outlined below:
  1. Delete all assigned users.
  2. Change the function right of the role:
  3. Assign the role again.

- **Description**
  Enter a description for the role. With predefined roles a description is displayed. The description text can be up to 100 characters long.

Groups

Note

The page is only available if there is an online connection to the device.

User Groups

On this page you link a group with a role.

In this example the group "Administrators" is linked to the "admin"role: The group is defined on a RADIUS server. The role is defined locally on the device. When a RADIUS server authenticates a user and assigns the user to the "Administrators" group, this user is given rights of the "admin" role.

Note

The values displayed depend on the rights of the logged-on user.

Description of the displayed boxes

The page contains the following:

- **Group Name**
  Enter the name of the group. The name must match the group on the RADIUS server. The name must meet the following conditions:
  - It must be unique.
  - It must be between 1 and 64 characters long.

The table contains the following columns:

- **Group**
  Shows the name of the group.
- **Role**
  Select a role. Users who are authenticated with the linked group on the RADIUS server receive the rights of this role locally on the device. You can choose between system-defined and self-defined roles, refer to the page "Security > Users > Roles.".
- **Description**
  Enter a description for the link of the group.to a role. The description text can be up to 100 characters long.

AAA

General

Login of network nodes

The designation "AAA" used in the menu stands for "Authentication, Authorization, Accounting". This feature is used to identify and allow network nodes and to make the corresponding services available to them.

On this page, you configure the login.

Description of the displayed boxes

The page contains the following boxes:

Note

To be able to use the login authentication "RADIUS", "Local and RADIUS" or "RADIUS and fallback Local" a RADIUS server must be stored and configured for user authentication.

- Login Authentication
  Specify how the login is made:
  - Local
    The authentication must be made locally on the device.
  - RADIUS
    The authentication must be handled via a RADIUS server.
  - Local and RADIUS
    The authentication is possible both with the users that exist on the device (user name and password) and via a RADIUS server. The user is first searched for in the local database. If the user does not exist there, a RADIUS query is sent.
  - RADIUS and fallback Local
    The authentication must be handled via a RADIUS server. A local authentication is performed only when the RADIUS server cannot be reached in the network.

RADIUS client

Authentication over an external server

The concept of RADIUS is based on an external authentication server.

Each row of the table contains access data for one server. In the search order, the primary server is queried first. If the primary server cannot be reached, secondary servers are queried in the order in which they are entered. If no server responds, there is no authentication.

Description of the displayed boxes

The page contains the following boxes:

- RADIUS Authorization Mode
  For the login authentication, the RADIUS authorization mode specifies how the rights are assigned to the user with a successful authentication.
  - conventional
    In this mode the user is logged in with administrator rights if the server returns the value "Administrative User" to the device for the attribute "Service Type". In all other cases the user is logged in with read rights.
  - Vendor specific
    In this mode the assignment of rights depends on whether and which group the server returns for the user and whether or not there is an entry for the user in the table "External User Accounts".

The table has the following columns:

- **Auth. Server Type**
  Select which authentication method the server will be used for.
  - Login
    The server is used only for the login authentication.
  - 802.1X
    The server is used only for the 802.1X authentication.
  - Login & 802.1X
    The server is used for both authentication procedures.
- **RADIUS Server Address**
  Enter the IPv4 address or the FQDN (Fully Qualified Domain Name) of the RADIUS server.
- **Server Port**
  Here, enter the input port on the RADIUS server. As default, input port 1812 is set. The range of values is 1 ... 65535.
- **Shared Secret**
  Enter your access ID. The range of values is 1 ... 128 characters.
- **Confirm Shared Secret**
  Enter your access ID again as confirmation.
- **Max. Retrans**
  Here, enter the maximum number of retries for an attempted request. The initial connection attempt is repeated the number of times specified here before another configured RADIUS server is queried or the login counts as having failed. As default 3 retries are set, this means 4 connection attempts. The range of values is 1 ... 5.
- **Primary Server**
  Specify whether or not this server is the primary server. You can select one of the options "yes" or "no".
- **Test**
  With this button, you can test whether or not the specified RADIUS server is available. The test is performed once and not repeated cyclically.
- **Test Result**
  Shows whether or not the RADIUS server is available:
  - Failed, no test packet sent
    The IP address is not reachable.
    The IP address is reachable, the RADIUS server is, however, not running.
  - Reachable, key not accepted
    The IP address is reachable, the RADIUS server does not, however accept the shared secret.
  - Reachable, key accepted
    The IP address is reachable, the RADIUS server accepts the specified shared secret.

  The test result is not automatically updated. To delete the test result click the "Refresh" button.

Certificates

Overview

All loaded files (certificates and keys) are shown on this page. You have the following options for loading files on the device:

- System > Load & Save > HTTP
- System > Load & Save > TFTP
- System > Load & Save > SFTP

Description

- **Type**
  Shows the type of the loaded file.
  - CA Cert
    The CA certificate is signed by a CA (Certification Authority).
  - Machine certificate
  - Key File
  - Remote Cert
    Partner certificate
- **Filename**
  Shows the file name.
- **Status**
  Shows whether the certificate is valid or has already expired.
- **Subject DN**
  Shows the name of the applicant.
- **Issuer DN**
  Shows the name of the certificate issuer.
- **Issue Date**
  Shows the start of the period of validity of the certificate
- **Expiry Date**
  Shows the end of the period of validity of the certificate.
- **Used**
  Shows which function uses the certificate.

Certificates

The format of the certificate is based on X.509, a standard of the ITU-T for creating digital certificates. This standard describes the schematic structure of X.509 certificates. You will find further information on this on the Internet at "http://www.itu.int".

On this page, the content of the following structure elements can be displayed. If the structure element does not exist or is not completed in the selected certificate, nothing is shown in the box on the right. Certain entries can only be edited if they are supported.

Description

- **Filename**
  Select the required certificate.
- **Type**
  Shows the type of the loaded file.
  - CA Cert
    The CA certificate is signed by a CA (Certification Authority).
  - Machine certificate
  - Key File
  - Remote Cert
    Partner certificate
- **Subject DN**
  Shows the name of the applicant.
- **Issuer DN**
  Shows the name of the certificate issuer.
- **Subject Alternate Name**
  If it exists, an alternative name of the applicant is displayed.
- **Issue Date**
  Shows the start of the period of validity of the certificate
- **Expiry Date**
  Shows the end of the period of validity of the certificate.
- **Serial Number**
  Shows the serial number of the certificate.
- **Used**
  Shows which function uses the certificate.
- **Crypto Algorithm**
  Shows which cryptographic method is used.
- **Key Usage**
  Shows the purpose that the key belonging to the certificate is used for, e.g. to verify digital signatures.
- **Extended Key Usage**
  Shows whether the purpose is additionally restricted, e.g. only to verify signatures of the CA certificate.
- **Key File**
  Shows the key file.
- **Certificate Revocation List 1st URL**
  Enter the URL with which the revocation list can be called up. Can only be edited if supported by the certificate.

- Certificate Revocation List 2nd URL
  Enter an alternative URL. If the revocation list cannot be called up using the 1st URL, the alternative URL is used. Can only be edited if supported by the certificate.
- Certificate
  Shows the name of the certificate.
- Passphrase
  Enter the password for the certificate. Can only be edited if the encrypted file is password protected.
- Passphrase Confirmation
  Enter the password again. Can only be edited if the encrypted file is password protected.

Firewall

General

On this page, you activate the firewall.

Note

Please remember that if you disable the firewall, your internal network is unprotected.

Description

The page contains the following:

- Activate Firewall
  When enabled, the firewall is active.
- TCP Idle Timeout [s]
  Enter the required time in seconds. If no data exchange takes place, the TCP connection is terminated automatically when this time has elapsed.
  The range of values is 1 to 21474836.
  Default setting: 86400 seconds
- UDP Idle Timeout [s]
  Enter the required time in seconds. If no data exchange takes place, the UDP connection is terminated automatically when this time has elapsed.
  The range of values is 1 to 21474836.
  Default setting: 300 seconds
- ICMP Idle Timeout [s]
  Enter the required time in seconds. If no data exchange takes place, the ICMP connection is terminated automatically when this time has elapsed.
  The range of values is 1 to 21474836.
  Default setting: 300 seconds

Predefined IPv4 rules

The page contains predefined IP packet filter rules. If you create your own IP packet filter rules, these have a higher priority than the predefined IP packet filter rules.

Here, you can set which services of the device should be reachable from which interface/ subnet.

Description

- **Interface**
  Interface to which the setting relates. The list of interfaces/subnets is dynamic and is based on the settings from "Layer 3 > Subnet".
  - pppx or usb0 (only with M876-4): Allows access from the WAN interface to the device.
  - VLANx: Allows access from the IP subnet to the device.
- Access to the following IPv4 services is permitted:
  - All
    All IPv4 services
  - HTTP
    For access to Web Based Management.
  - HTTPS
    For secure access to Web Based Management.

    Note

    HTTP and HTTPS deactivated

    If you disable HTTP and HTTPS, the WBM of the device can no longer be reached.

    HTTPS disabled

    When you disable HTTPS, you can only access the WBM using HTTP. This assumes that "HTTP & HTTPS" is set in "System > Configuration > HTTP Services". If for example "Redirect HTTP to HTTPS" is set, access via HTTP cannot be redirected to HTTPS. This means that the WBM of the device can no longer be reached.
  - DNS
    DNS queries to the device. Necessary only if the "DNS-Relay" function is enabled on the device.
  - SNMP
    Incoming SNMP connections. Required, for example, to access the SNMP information of the device using a MIB browser.
  - Telnet
    For unencrypted access to the CLI.
  - SMS Relay (M87x only)
    For sending SMS messages from the local network.
  - IPSec VPN
    Allows IKE (Internet Key Exchange) data transfer from the external network to the device. Necessary if an IPsec VPN remote station needs to establish a connection to this device.
  - SSH
    For encrypted access to the CLI.
  - DHCP
    Access to the DHCP server or the DHCP client
  - Ping
    Access to the ping function

- System time
  Access to NTP and SNTP

IP Services

On this page, you define the IP services. Using the IP service definitions, you can define firewall rules for specific services. You select a name and assign the service parameters to it. When you configure the IP rules, you simply use this name.

Description

The page contains the following:

- Service Name
  Enter the name of the IP service. The name must be unique.

This table contains the following columns:

- Service Name
  Shows the name of the IP service.
- Transport
  Specify the protocol type.
  - UDP
    The rule applies only to UDP frames.
  - TCP
    The rule applies only to TCP frames.
- Source Port (Range)
  Enter the source port. The rule applies specifically to the specified port.
  - If the rule is intended to apply to a port range, enter the range with start port "-" end port, for example 30 - 40.
  - If the rule is intended to apply to all ports, enter "*".
- Destination Port (Range)
  Enter the destination port. The rule applies specifically to the specified port.
  - If the rule is intended to apply to a port range, enter the range with start port "-" end port, for example 30 - 40.
  - If the rule is intended to apply to all ports, enter "*".

ICMP Services

On this page, you define ICMP services. Using the ICMP service definitions, you can define firewall rules for specific services. You select a name and assign the service parameters to it. When you configure the IP rules, you simply use this name.

Description

The page contains the following:

- **Service Name**
  Enter a name for the ICMP service. The name must be unique.

This table contains the following columns:

- **Service Name**
  Shows the name of the ICMP service.
- **Protocol**
  Shows the version of the ICMP protocol.
- **Type**
  Specify the ICMP packet type. A few examples are shown below:
  - Destination Unreachable
    IP frame cannot be delivered.
  - Time Exeeded
    Time limit exceeded
  - Echo-Request
    Echo request, better known as ping.
- **Code**
  The code describes the ICMP packet type in greater detail. The selection depends on the selected ICMP packet type.
  With "Destination Unreachable", for example "Code 1" host cannot be reached.

IP Protocols

On this page, you can configure user-defined protocols, e.g. IGMP for multicast groups. You select a protocol name and assign the service parameters to it. When you configure the IP rules, you simply use this protocol name.

Description

The page contains the following:

- **Protocol Name**
  Enter a name for the protocol.

The page contains the following check boxes:

- **Protocol Name**
  Shows the protocol name.
- **Protocol Number**
  Enter the protocol number, for example 2. You will find list of the protocol numbers on the Internet pages of iana.org

IP Rules

On this page you specify your own IP packet filter rules for the firewall.

The IP packet filer rules set here have priority:

- over the predefined IP packet filter rules (predefined IPv4) and
- over the IP packet filter rules created automatically due to a connection configuration (SINEMA RC).

Description

This table contains the following columns:

- Select
  Activate the check box in the row to be deleted.
- Protocol
  Shows the version of the IP protocol.
- Action
  Select how incoming IP packets are handled:
  - "Accept" - The data packets can pass through.
  - "Reject" – The data packets are rejected, and the sender receives a corresponding message.
  - "Drop" – The data packets are discarded without any notification to the sender.
- From / To
  Specify the communications direction of the IP rule.
  - VLANx: VLANs with configured subnet
  - Device
  - SINEMA RC: Connection to SINEMA RC Server
  - IPsec: Either all IPsec VPN connections (all) or a specific IPsec VPN connection
  - pppx bzw. usb0: WAN interface
  - OpenVPN: Either all OpenVPN connections (all) or a specific OpenVPN connection
- Source (Range)
  Enter the IP address or an IP range that is allowed to send IP packets.
  - If the rule is intended to apply to an IP range, enter the range with start address "-" end address , for example 192.168.100.10 - 192.168.100.20.
  - If the rule is intended to apply to all IP addresses , enter " 0.0.0.0/0".
- Destination (Range)
  Enter the IP address or an IP range that is allowed to receive IP packets.
  - If the rule is intended to apply to an IP range, enter the range with start address "-" end address , for example 192.168.100.10 - 192.168.100.20.
  - If the rule is intended to apply to all IP addresses , enter " 0.0.0.0/0".
- Service
  Select the service or the protocol name for which this rule is valid.

- **Log**
  Specify whether or not there should be a log entry every time the rule comes into effect and specify the severity of the event.
  The following settings are available:
  - none
    The rule coming into effect is not logged.
  - info / warning / critical
    The rule coming into effect is logged with the selected event severity. The log file is displayed in "Information" > "Log Tables" > "Firewall Log".
- **Precedence**
  Specify the precedence of the rule.

Note

Opening the WBM via a VPN tunnel

To open the WBM via a VPN tunnel, create the following IP rule:

Action: Accept

From: IPsec

To: Device

Service HTTPS (TCP port 443)

IPsec VPN

Overview

To create an IPsec VPN connection, you have the following options

Automatic creation via the VPN groups in STEP 7

You will find further information on creating VPN groups for IPsec VPN connections in the section "IPsec VPN tunnel: Creating and assigning VPN groups"

STEP 7 via VPN groups

As an alternative you can configure IPsec connections via VPN groups, refer to the section "IPsec VPN tunnel: Creating and assigning VPN groups" When you configure via VPN groups the remote addresses and remote subnets are determined by STEP 7 based on the interface names:

- Interface name "INT": STEP 7 handles the interface as an internal interface.
- Interface name "EXT": STEP 7 handles the interface as an external interface.

When adding devices to VPN groups, STEP 7 only creates VPN connections between the external interfaces. Only the local subnets on the internal interfaces are connected together via VPN. You need to configure additional VPN connections manually.

Special features when configuring via VPN groups

- You specify permission to initiate connection establishment with the assignment to VPN groups.
- In the connection-specific VPN settings, the device is not displayed as connection partner so that it can be edited. When the device is selected, the interface can be adapted to other partner devices.
- In terms of the VPN mode, the device always behaves as a routing device.
- The properties of a VPN group are adopted in the local properties of VPN group members. The properties of a VPN group relating to the following parameters can be overwritten locally:
  - Phase 1: Keying Tries, DPD, DPD Period, DPD Timeout
  - Phase 2: Lifebytes, Protocol, Port (Range), Auto Firewall Rules

General

On the page, you configure the basic settings for VPN.

Description

The page contains the following:

- **Activate IPsec VPN**
  Enable or disable the IPsec protocol for VPN.
- **Enforce strict CRL Policy**
  When enabled, the validity of the certificates is checked based on the CRL (Certificate Revocation List). The certificate revocation list lists the certificates issued by the certification authority that have lost their validity before the set expiry date. You configure the certificate revocation list to be used on the page "Certificates (Page 2706)".
- **NAT Keep Alive Time Interval**
  Specify the interval at which sign of life frames (keepalives) are sent. If there is a NAT device between two VPN endpoints, when there is inactivity, the connection is deleted from its dynamic NAT table. To prevent this, keepalives are sent.

Remote End

On this page, you configure the counterpart station (VPN end point).

Description

The page contains the following:

- **Remote End Name**
  Enter the name of the partner station and click "New Entry" to create a new partner station.

This table contains the following columns:

- Name
  Shows the name of the partner.
- Remote Mode
  Specify the role the remote stations will adopt.
  - Roadwarrior
    The reachable remote addresses are entered. The reachable remote subnets are learned from the partner.
  - Standard
    The reachable remote address and the reachable remote subnets are entered permanently.
- Remote Type
  Specify the type of remote station address.
  - Manual
    The address of the partner is known. The device can either establish the VPN connection actively as a VPN client or wait passively for connection establishment by the partner.
  - Any
    Accepts the connection from remote stations with any IP address address. The device can only wait for VPN connections but cannot establish a VPN tunnel as the active partner.
- Remote Address
  Can only be edited with the remote type "Manual".
  - In standard mode, enter the WAN IP address or the DDNS host name of the partner. The network mask is always 32
  - In Roadwarrior mode, you can specify either the address of the partner or enter an IP range from which connections will be accepted.
- Remote Subnet
  - In standard mode, enter the remote subnet of the remote station. Use the CIDR notation.
  - In Roadwarrior mode, the remote address informs the device of its reachable subnets and the device learns them.
- Virtual IP Mode
  Specify whether or not the remote station is offered a virtual IP address.
  The following options are available:
  - User defined IPv4
    The virtual IP address is from the band specified in "Virtual IP".
  - None
    No virtual IP address. The VPN tunnel is established dynamically to the internal IP address of the remote station.
- Virtual IP
  Specify the subnet (CIDR) from which the remote station is offered a virtual IP address.
  Can only be edited if "user defined IPv4" is selected in "Virtual IP Mode".

Connections

On this page, you configure the basic settings for the VPN connection. With these settings, the device (local endpoint) can establish a non-secure VPN tunnel to the partner. You specify the security settings on the "Authentication" page.

Note

Several IPsec VPN connections via the same VPN endpoint

If you have created IPsec VPN connections to different remote subnets via the same VPN endpoint, the VPN connection configured first (lowest index) is the main connection (parent).

All other IPsec VPN connections (children) are created and established via the main connection. If all VPN tunnels are now established and the parent connection is terminated, all child connections are interrupted. After the DPD timeout has elapsed, all IPsec VPN connections are re-established via the main connection.

If only one child connection is terminated, the main connection and the other child connections remain intact.

Note

If you use "NETMAP"

- only auto firewall rules are supported
- For "Operation" the setting "on demand" cannot be selected

Description

The page contains the following boxes:

- **Connection name**
  Enter a name for the VPN connection.

This table contains the following columns:

- Name
  Shows the name of the VPN connection.
- Operation
  Specify who establishes the VPN connection. You will find more detailed information in "Technical basics > VPN connection estalishment (Page 2578)".
  - Disabled
    The VPN connection is disabled.
  - start
    The device attempts to establish a VPN connection to the partner.
  - wait
    The device waits for the partner to initiate the connection.
  - on demand
    The VPN connection is established when necessary.
  - start on DI
    If the event "Digital In" occurs the device attempts to establish a VPN connection to the remote station.
    This is on condition that the event "Digital In" is forwarded to the VPN connection. To do this in "System > Events> Configuration (Page 2621)" activate "VPN Tunnel" for the "Digital Input" event.
  - wait on DI
    If the event "Digital In" occurs, the device waits for the remote station to initiate connection establishment.
    This is on condition that the event "Digital In" is forwarded to the VPN connection. To do this in "System > Events> Configuration (Page 2621)" activate "VPN Tunnel" for the "Digital Input" event.
  - With start on SMS(M87x only)
    If the device receives a command SMS, the device attempts to establish a VPN connection to the remote station. This assumes that the device accepts a command SMS of the class "System" from certain senders. You configure the senders in "System > SMS > SMS Command (Page 2645)".
  - wait on SMS (M87x only)
    When the device receives an SMS command, the device waits until the connection establishment is initiated by the remote station. This assumes that the device accepts a command SMS of the class "System" from certain senders. You configure the senders in "System > SMS > SMS Command (Page 2645)".
- Keying Protocol
  Specify whether IKEv2 or IKEv1 will be used.
- Remote End
  Select the required remote station. Only partners can be configured that have been configured on the "Remote End" page.
- Local Subnet
  Enter the local subnet. Use the CIDR notation. The local network can also be a single PC or another subset of the local network.

- **Request Virtual IP**
  When enabled, a virtual IP address is requested from the remote station during connection establishment.
- **Timeout [sec]**
  Only necessary with the “on demand" setting. Enter the interval after which the VPN connection will be terminated. If no packets are sent during this time, the VPN connection is automatically terminated.

Authentication

On this page, you specify how the VPN connection partners authenticate themselves with each other.

Description

This table contains the following columns:

- **Name**
  Shows the name of the VPN connection to which the settings relate.
- **Authentication**
  Select the authentication method. For the VPN connection, it is essential that the partner uses the same authentication method.
  - Disabled
    No authentication method is selected. Connection establishment is not possible.
  - Remote Cert
    The remote certificate is used for authentication. You specify the certificate in "Remote Certificate"
  - CA Cert
    The certificate of the certification authority is used for authentication. You specify the certificate in "CA Certificate". If you want to establish an IPsec connection to SINEMA RC, you need to add the certificate later via the WBM and download it to the device.
  - PSK
    A key is used for authentication. You configure the key in "PSK".
- **CA Certificate**
  Select the certificate. Only loaded certificates can be selected.
- **Local Certificate**
  Select the machine certificate. Only the device certificates generated by the TIA Portal can be selected.
- **Local ID**
  Enter the local ID from the partner certificate. Only when you use the partner certificate can you leave the box empty. The box is automatically filled with the value from the partner certificate.
- **Remote Certificate**
  Select the remote station certificate. Only the partner certificates generated by the TIA Portal can be selected.

- **Remote ID**
  Enter the "Distinguished Name" or "Alternate Name" from the partner certificate. Only when you use the partner certificate can you leave the box empty. The box is automatically filled with the value from the partner certificate.
- **PSK**
  Enter the key.
- **PSK Confirmation**
  Repeat the key.

Phase 1

Phase 1: Encryption agreement and authentication (IKE = Internet Key Exchange)

On this page, you set the parameters for the protocol of the IPsec key management. The key exchange uses the standardized IKE method for which you can set the following protocol parameters.

Description

This table contains the following columns:

- **Name**
  Shows the name of the VPN connection to which the settings relate.
- **Default Ciphers**
  When enabled, a preset list is transferred to the VPN connection partner during connection establishment. The list contains a combination of the three algorithms (Encryption, Authentication, Key Derivation). To establish a VPN connection, the VPN connection partner must support at least one of the combinations. The selection depends on the key exchange method. Further information can be found in the section "IPsec VPN (Page 2574)"
- **Encryption**
  For phase 1, select the required encryption algorithm. Can only be selected if "Default Ciphers" is disabled.
  The selection depends on the key exchange method. Further information can be found in the section "IPsec VPN (Page 2574)".

  Note

  The AES modes CCM and GCM contain separate mechanisms for authenticating data. If you use a mode AES x CCM for "Encryption", this is also used for authentication. Then only the pseudo random function will be derived from the "Authentication" parameter. So that a VPN connection can be established, all devices need to use the same settings.

- **Authentication**
  Specify the method for calculating the checksum. Can only be selected if "Default Ciphers" is disabled.
  The following methods are supported:
  - MD5
  - SHA1
  - SHA512
  - SHA256
  - SHA384
- **Key derivation**
  Select the required Diffie-Hellmann group (DH) from which a key will be generated. Can only be selected if "Default Ciphers" is disabled.
  The following DH groups are supported:
  - DH group 1
  - DH group 2
  - DH group 5
  - DH group 14
  - DH group 15
  - DH group 16
  - DH group 17
  - DH group 18
- **Keying Tries**
  Enter the number of repetitions for a failed connection establishment. If you enter the value 0, the connection establishment will be attempted endlessly.
- **Lifetime [min]**
  Enter a period in minutes to specify the lifetime of the authentication. When the time has elapsed, the VPN endpoints involved must authenticate themselves with each other again and generate a new key
- **DPD**
  When enabled DPD is used. Using DPD, it is possible to find out whether the VPN connection still exists or whether it has aborted.

  Note

  Sending DPD queries increases the amount of data sent and received. This can lead to increased costs.

- **DPD Period [sec]**
  Enter the period after which DPD requests are sent. These queries test whether or not the remote station is still available

- **DPD Timeout [sec]**
  Enter a period. If there is no response to the DPD queries, the connection to the remote station is declared to be invalid after this time has elapsed.
- **Aggressive Mode**
  - Disabled:
    Main Mode is used.
  - Enabled
    Aggressive Mode is used

  The difference between main and aggressive mode is the "identity protection" used in main mode. The identity is transferred encrypted in main mode but not in aggressive mode.

Phase 2

Phase 2: Data exchange (ESP = Encapsulating Security Payload)

On this page, you set the parameters for the protocol of the IPsec data exchange. The entire communication during this phase is encrypted using the standardized security protocol ESP for which you can set the following protocol parameters.

Description

This table contains the following columns:

- **Name**
  Shows the name of the VPN connection to which the settings relate.
- **Default Ciphers**
  When enabled, a preset list is transferred to the VPN connection partner during connection establishment. The list contains a combination of the three algorithms (Encryption, Authentication, Key Derivation). To establish a VPN connection, the VPN connection partner must support at least one of the combinations. Further information can be found in the section "IPsec VPN (Page 2574)".
- **Encryption**
  For phase 2, select the required encryption algorithm. Can only be selected if "Default Ciphers" is disabled.
  Further information can be found in the section "IPsec VPN (Page 2574)".

  Note

  The AES modes CCM and GCM contain separate mechanisms for authenticating data. If you use a mode AES x CCM or AES x GCM for "Encryption", this will also be used for authentication. Then only the pseudo random function will be derived from the "Authentication" parameter.

- **Authentication**
  Specify the method for calculating the checksum. Can only be selected if "Default Ciphers" is disabled.
  The following methods are supported:
  - MD5
  - SHA1
  - SHA512
  - SHA256
  - SHA384
- **Key Derivation**
  Select the required Diffie-Hellmann group (DH) from which a key will be generated. Can only be selected if "Default Ciphers" is disabled.
  The following DH groups are supported:
  - None: For phase 2, no separate keys are exchanged. This means that Perfect Forward Secrecy (PFS) is disabled.
  - DH group 1
  - DH group 2
  - DH group 5
  - DH group 14
  - DH group 15
  - DH group 16
  - DH group 17
  - DH group 18

  Note

  So that a VPN connection can be established, all devices need to use the same settings or provide compatible key procedures..

- **Lifetime [min]**
  Enter a period in minutes to specify the lifetime of the agreed keys. When the time expires, the key is renegotiated.
- **Lifebytes**
  Enter the data limit in bytes that specifies the lifetime of the agreed key. When the data limit is reached, the key is renegotiated.
- **Protocol**
  Specify the protocol for which the VPN connection is valid e.g. UDP, TCP, ICMP. If the setting is intended to apply to all protocols, enter "*".

- **Port (Range)**
  Specify the port via which the VPN tunnel can communicate. The setting applies specifically to the specified port
  - If the setting is intended to apply to a port range, enter the range with start port "-" end port, for example 30 - 40.
  - If the setting is intended to apply to all ports, enter "*".

  The setting is only effective for port-based protocols.
- **Auto Firewall Rules**
  - enabled
    The firewall rules are created automatically for the VPN connection.
  - disabled
    You will need to create the firewall rules yourself.

OpenVPN Client

General

On this page, you activate the OpenVPN client.

Description

The page contains the following:

- **Activate OpenVPN Client**
  Enable or disable the OpenVPN client.

Connections

On this page, you configure the basic settings for the OpenVPN connection. You specify the security settings on the page "Authentication".

Description

- **Connection name**
  Enter a unique name for the OpenVPN connection.

This table contains the following columns:

- Name
  Shows the name of the OpenVPN connection.
- Operation
  Specify how the VPN connection is estalished. You will find more detailed information in "Technical basics > VPN connection estalishment (Page 2578)".
  - start
    The device attempts to establish a VPN connection to the partner.
  - start on DI
    If the event "Digital In" occurs the device attempts to establish a VPN connection to the remote station.
    This is on condition that the event "Digital In" is forwarded to the VPN connection. To do this in "System > Events> Configuration" activate "VPN Tunnel" for the "Digital In" event.
  - start on SMS (only with M87x)
    If the device receives a command SMS, the device attempts to establish a VPN connection to the partner. This assumes that the device accepts a command SMS of the class "System" from certain senders. You configure the senders in "System > SMS > SMS Command".
  - Disabled
    The VPN connection is disabled.
- Encryption
  Select the required encryption algorithm.
  - AES-128-CBC (default)
  - AES-192-CBC
  - AES-256-CBC
  - DES-EDE3
  - BF-CBC
- Authentication
  Specify the method for calculating the checksum.
  - SHA256 (default)
  - SHA384
  - SHA512
  - SHA224
  - SHA1
  - MD5

- LZO Comp.
  - Deactivated (-)
    The compression is disabled. The server cannot enable compression again.
  - No
    The compression is disabled as default. The server can enable compression
  - Yes
    The compression is enabled as default. The server can disable the compression
  - Adaptive
    As default compression is activated adaptively. Compression is only used when the data is good to compress otherwise compression is deactivated for a certain time.
- Auto Firewall Rules
  - Enabled
    The firewall rules are created automatically for the VPN connection.
  - Disabled
    You will need to create the suitable firewall rules yourself.
- Enable NAT
  With this setting, you enable automatic IP masquerading for this interface. The local devices are not directly reachable from the outside, but only via the IP address of the interface. The local devices can, however, connect to the devices downstream from the OpenVPN server. You will find more information on NAT in "Technical basics > NAT""

Remote

On this page, you configure the counterpart station (OpemVPN end point). Per connection, you can specify several OpenVPN partners. The device tries all configured OpenVPN partners one after the other until a connection is successfully established.

Description

The page contains the following:

- Remote Name
  Enter a name for the OpenVPN partner.

This table contains the following columns:

- Name
  Shows the name of the Open VPN partner.
- Connection
  Select the corresponding connection. Only connections can be configured that you configured on the "Connections" page.
- Remote Address
  Enter the WAN IP address or the DNS host name of the OpenVPN partner.

- **Port**
  Specify the port via which the OpenVPN tunnel can communicate. The setting applies specifically to the specified port.
- **Protocol**
  Specify the protocol for which the OpenVPN connection will be used.

Authentication

On this page, you specify how the VPN connection partners authenticate themselves with each other.

Description

This table contains the following columns:

- **Name**
  Shows the name of the VPN connection to which the settings relate.
- **Method**
  Select the authentication method. For the VPN connection, it is essential that the partner uses the same authentication method.
  - Disabled
    No authentication method is selected. Connection establishment is not possible.
  - Certificates
    Certificates are used for the authentication. The certificates need to be added later via the WBM and downloaded to the device.
  - Username/Password
    The user name/password are used for the authentication. The certificate needs to be added later via the WBM and downloaded to the device.
- **CA Certificate**
  Select the CA certificate via the WBM and load it onto the device.
- **Machine certificate**
  Select the machine certificate via the WBM and download it to the device.
- **User Name**
  Specify the user name.
- **Password**
  Enter the password.
- **Password Confirmation**
  Confirm the password.

1.4.9 Configuring PROFIBUS DP

1.4.9.1 Configuring distributed I/O

In the chapter below you will learn how to configure distributed I/O. This chapter will also provide you with basic information on configuring a DP master system.

Basic information

- Basics of configuring a DP master system (Page 2728)
- DP slaves in the hardware catalog (Page 2730)
- Structure of a configuration with simple DP slaves (mono master system) (Page 2731)
- Installation of a configuration with intelligent slaves (I-slaves) (Page 2732)
- Installation of a configuration with two master systems (Page 2733)

Overview for configuring distributed I/O

Step	Description
1	Create a DP master system (Page 2735).
2	If needed configure the interfaces of the DP master system (Page 2738).
3	Add a DP slave and configure it (Page 2739).
4	If necessary, assign DP slaves to a SYNC/FREEZE group (Page 2743).

See also

Configuring DPV1 devices (Page 2780)

Configuring direct data exchange (Page 2769)

1.4.9.2 The basics of configuring a DP master system

Distributed I/O

DP master systems that consist of a DP master and DP slaves which are connected via a bus and communicate with one another via the PROFIBUS DP protocol are referred to as distributed I/O.

Configuring distributed I/O

The fact that the type of device may vary as far as DP masters/DP slaves are concerned means that these instructions deal only with the basic configuration procedure. However, the process for configuring distributed I/O is practically identical to that of non-distributed configuration.

Creating the DP master system in the network view

After you have used dragged a DP master and a DP slave (for example, a CPU 315--2 and an IM 153--1) from the hardware catalog to the network view, connect them both to a PROFIBUS subnet.

As the network view only allows you to configure devices (not modules), you will need to switch to the device view in order to configure the DP master and slaves.

Configuring a DP slave

Switch to the device view to configure the DP slave. In the device view, populate the DP slave rack with modules and parameterize DP identifiers and I/O addresses. You can check and edit the settings in the device overview and in the inspector window.

Slot numbering for distributed I/O

Depending on the type of DP slave used, you should start the DP slave slot numbers at either "0" or "4".

The numbering of DP slave slots, on an ET 200M for example, is based on the structure for an S7-300:

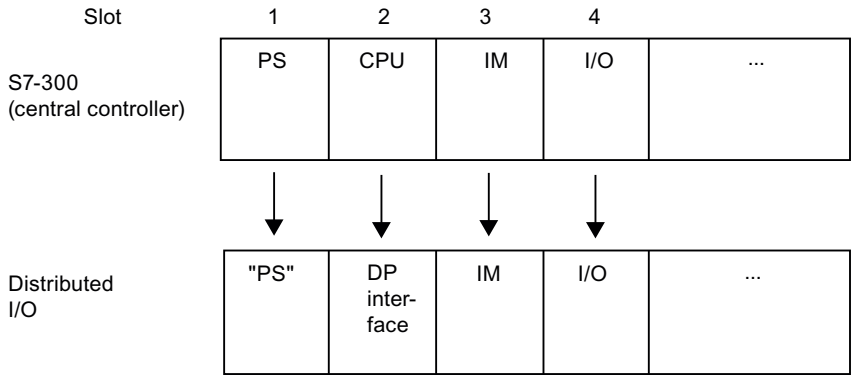

DP slave slot rules in accordance with the S7-300 structure

The following slot rules apply to DP slaves that are structured in accordance with the S7-300 scheme:

- The actual I/O (inputs/outputs) always start at slot 4.
- Slot 1 is always reserved for a power supply module (PS) even if the actual configuration does not happen to include a PS.
- Slot 2 is always reserved for the DP interface.
- Slot 3 is always reserved for an extension interface module (IM) regardless of whether or not the actual I/O is capable of being expanded.

This scheme applies to all modular and compact DP slave types that are structured in accordance with the S7-300 model. The manner in which the slots are assigned is significant from the point of view of evaluating diagnostics messages ("Slot triggering diagnostics").

Additional information

Please note the additional information relating to the scope of available functions, access methods, etc. that can be found in the manuals for the relevant devices or in the online help for specific FCs (for example, DP-SEND and DP-RECEIVE for CP 342-5).

1.4.9.3 DP slaves within the hardware catalog

DP slaves in the hardware catalog

You will find the DP slaves in the "Distributed I/O" folder of the hardware catalog. Compact and modular slaves are located there:

- Compact DP slaves
  Modules with integrated digital/analog inputs and outputs, for example, ET 200L
- Modular DP slaves
  (Interface modules with S7 modules assigned, for example, ET 200M

The available DP master and the desired functionality will determine which DP slaves can be used.

I-slaves within the hardware catalog

You will find devices that you can configure as intelligent DP slaves in the "PLC" folder of the hardware catalog. The following are examples of devices that you can configure as I-slaves:

- CP 342-5 DP
  in the hardware catalog under "PLC > SIMATIC S7-300 > CP > PROFIBUS".
- CPU 315-2 DP, CPU 317-2 DP, CPU 319-3 PN/DP
  in the hardware catalog under "PLC > SIMATIC S7-300 > CPU".
- IM 151-7 CPU
  in the hardware catalog under "PLC > SIMATIC ET200 PLC > ET 200S > CPU".

1.4.9.4 DP/DP coupler in the hardware catalog

Introduction

A DP/DP coupler connects two PROFIBUS DP networks as a gateway so that the DP master from one network can transfer data to the DP master of the other network.

The maximum amount of data that can be transferred is 244 bytes input data and 244 bytes output data.

DP/DP coupler in the hardware catalog

Details of a DP/DP coupler as gateway between two DM master systems are contained in the hardware catalog in the folder "Other field devices > PROFIBUS-DP > Gateways".

Configuring the DP/DP coupler

DP/DP couplers are configured in both PROFIBUS networks, each in their own master systems.

The input and output areas of both networks must thereby be matched to one another. The output data from one end of the DP/DP coupler are accepted as input data at the other respective end and vice versa.

See also

Installing the GSD file (Page 2775)

Creating a DP master system (Page 2735)

1.4.9.5 Configurations involving PROFIBUS DP

Configurations involving "simple" DP slaves

Communication between DP master and DP slave

In the case of a configuration involving simple DP slaves, data is exchanged between the DP master and simple DP slaves, i.e. with I/O modules via the DP master. The DP master polls each of the configured DP slaves in its polling list within the DP master system in succession and sends the output data/receives the input values from the slaves.

Mono-master system

The configuration with only one DP master is also described as mono-master system. A single DP master with its associated DP slaves is connected to a physical PROFIBUS DP subnet.

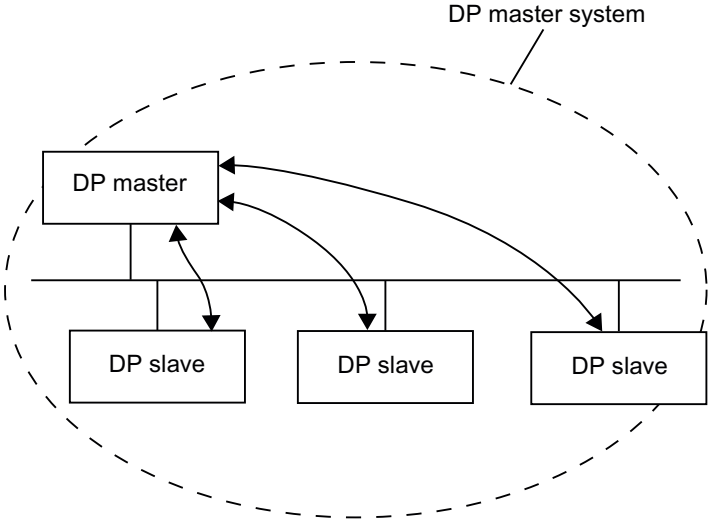

Configurations involving several DP master systems

Configuration methods involving two or more DP master systems

When more than one DP master system is connected to a physical PROFIBUS DP subnet it is sometimes called a multi-master system.

You can use various methods to connect these DP master systems via PROFIBUS DP:

- Without logical connection of the DP master systems to each other
- Direct DP slave > DP master data exchange
- Direct DP slave > I-slave data exchange

Direct DP slave > DP master data exchange

For the configuration of multi-master systems, input data of DP slaves of DP masters can be read on the same physical PROFIBUS DP subnet.

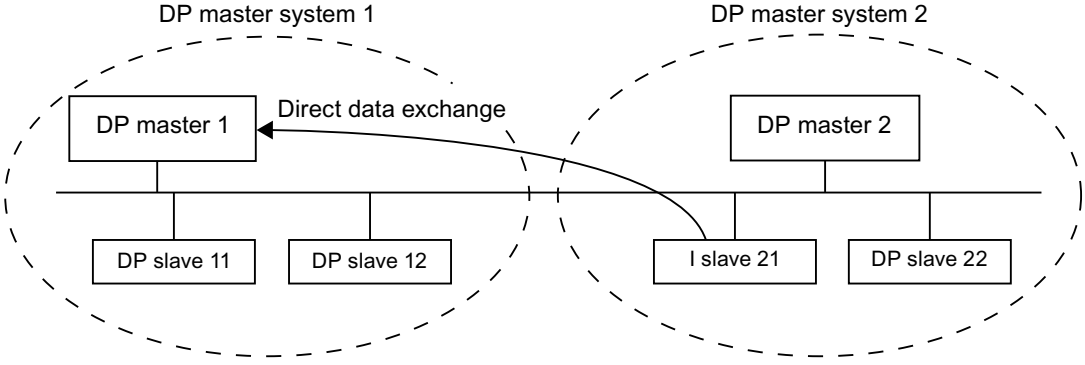

Direct DP slave > I-slave data exchange

This means that an intelligent DP slave (I-slave), such as a CPU 315-2 DP, can support the direct transfer of input data from DP slaves with different DP master systems to its input data area.

In principle, DP slaves of a certain version or higher can make selected input data available for the purpose of direct data exchange. However, only intelligent DP slaves are able to make further use of this input data.

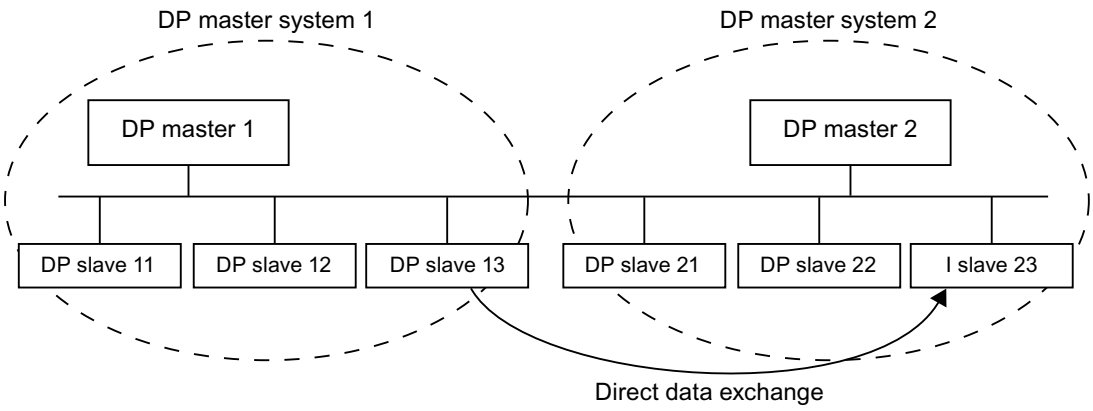

See also

What you need to know about direct data exchange (Page 2767)

Configuring direct data exchange (Page 2769)

Configurations involving intelligent DP slaves

Definition

DP slaves that feature their own preprocessing program are referred to as intelligent DP slaves (I-slaves). Examples of intelligent DP slaves include:

- CPU 315-2 DP
- CPU 317-2 DP
- CPU 319-3 PN/DP

Configurations involving intelligent DP slaves

There are two methods for configuring intelligent DP slaves using PROFIBUS DP:

- I-slave <> DP master data exchange
- Direct DP slave > I-slave data exchange

I-slave <> DP master data exchange

A higher-level automation system processes the automation task, which is broken down into sub-tasks. The necessary control tasks are processed separately and efficiently in a CPU as preprocessing programs. This CPU can be implemented in the form of an intelligent DP slave.

In the case of configurations involving intelligent DP slaves, for example CPU 315-2 DP, the DP master merely accesses the address area of the I-slave CPU, rather than the I/O modules of the intelligent DP slave. The address area must not be assigned to any actual I/O modules on the I-slave. The assignment must be made during I-slave configuration.

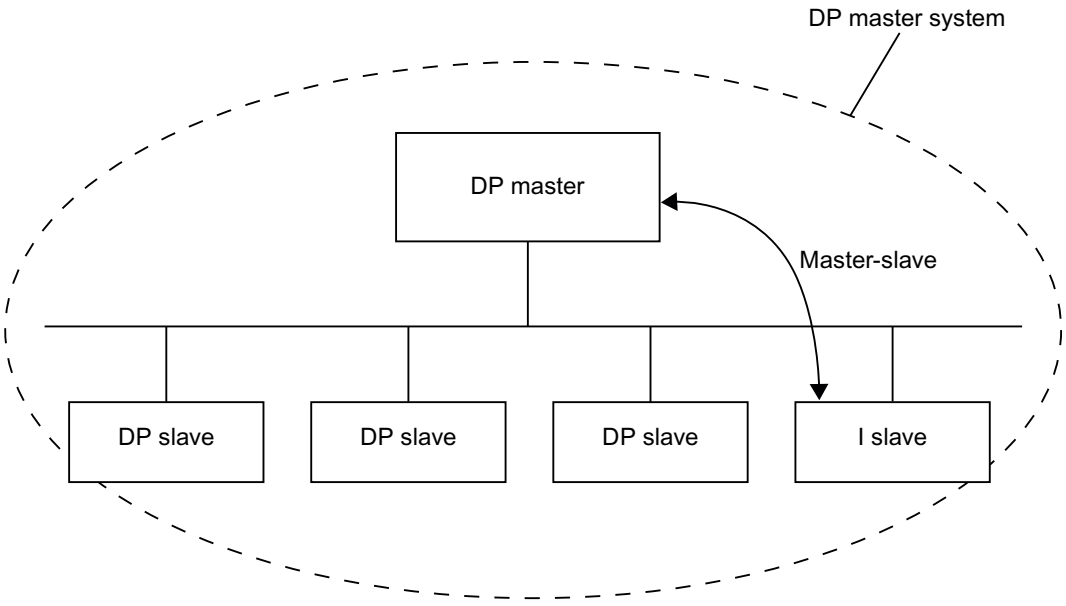

Direct DP slave > I-slave data exchange

With this configuration, input data from simple DP slaves can be transferred to intelligent DP slaves on the PROFIBUS DP subnet at high speed.

In principle, all simple DP slaves (of a certain version or higher) and intelligent DP slaves can make their input data available for the purpose of direct data exchange. Intelligent DP slaves can be used as recipients of this data.

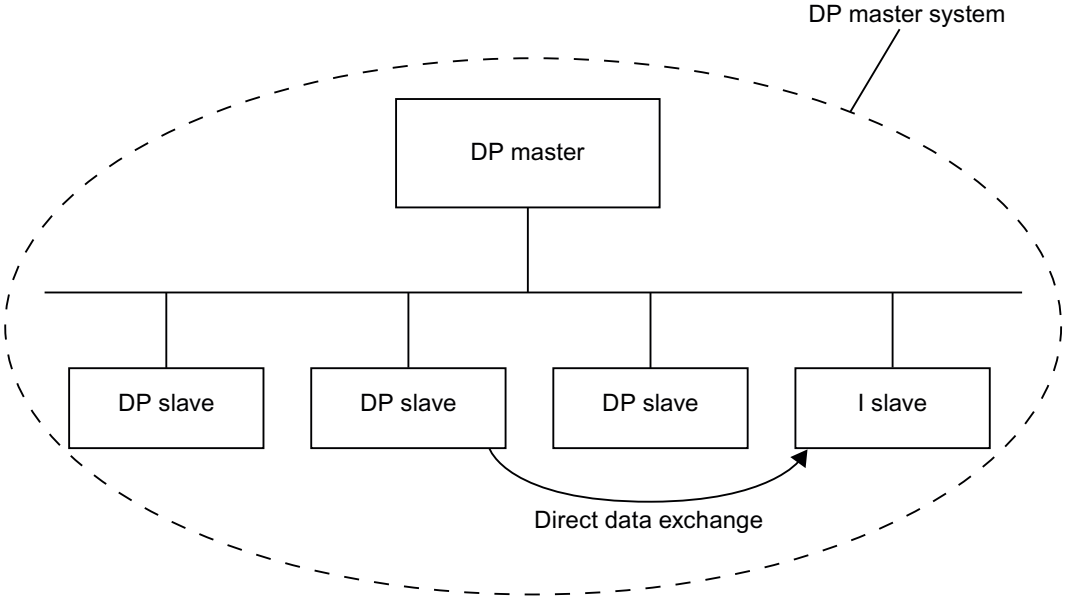

See also

Adding an I-slave to a DP master system (Page 2755)

Example of how to configure an ET 200 CPU as an I-slave (Page 2763)

What you need to know about direct data exchange (Page 2767)

1.4.9.6 Configuring distributed I/O systems

Creating a DP master system

Introduction

To create a PROFIBUS DP master system, you need to have one DP master and at least one DP slave. As soon as you connect a DP master to a DP slave via their PROFIBUS DP interfaces, a master-slave link is established.

DP master and DP slave

As the DP master, you can use the following devices with a DP interface:

- CPU with a permanently integrated or plug-in DP master interface
- CP in conjunction with a CPU
- Interface module assigned to a CPU/FM
- Interface module with a DP master interface

As DP slave, you can use head modules of the distributed I/O with PROFIBUS DP interface or CPUs as intelligent DP slaves.

With some devices, you can switch between the possible modes (such as "DP master" or "DP Slave") under "Operating mode" in the Inspector window of the DP interface. Select the option button with the required mode.

Requirement

- You are in the network view.
- The hardware catalog is open.

Procedure

To create a DP master system for example with a CPU 319-3 PN/DP , follow these steps:

1. Select a CPU 319-3 PN/DP as a potential DP master from the hardware catalog.
2. Drag the CPU onto the free area within the network view.
3. Right click on the DP interface of the CPU.
4. Select "Create master system" from the shortcut menu.

A DP master system with a CPU 319-3 PN/DP as DP master will be created as the only node.

If you connect the DP interface of a DP slave to the DP interface of the DP master, the DP slave will be added automatically to the DP master system. If there is not yet a subnet between the DP master and DP slave, a new subnet is created between the DP master and DP slave.

To include an IM 155-6 DP HF as a DP slave in the DP master system, for example, follow the steps below:

1. Click the DP interface of either the DP master or DP slave.
2. Hold down the mouse button and draw a connecting line between this DP interface and that of the desired communication partner.

or

1. Click on the hyperlink of the DP slave IM 155-6 DP HF.
2. Select the required DP master from the list of possible DP masters displayed.

The DP slave IM 155-6 DP HF is included in the DP master system with the CPU 319-3 PN/DP as DP master.

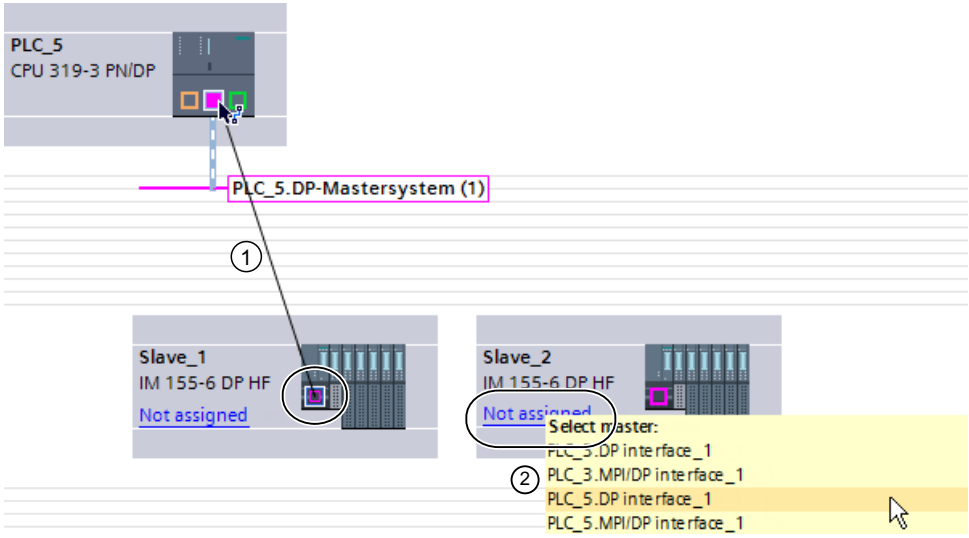

① Drag between the DP interfaces of DP slave to DP master.

② Click on the link of an unassigned DP slave, and a selection of possible DP masters opens.

If necessary, adapt the properties of the PROFIBUS subnet or of the DP master (e.g. PROFIBUS address) in the Inspector window under "Properties".

Note

If you use the mouse to draw a connecting line between the MPI interface of a DP master and the PROFIBUS interface of a DP slave, the MPI interface of the DP master will automatically be configured as a PROFIBUS interface and a DP master system will be created.

DP master display on the DP slave

When a DP slave is connected to a DP master, the name of the DP master is displayed on the DP slave as a hyperlink. Click the hyperlink to select the associated DP master.

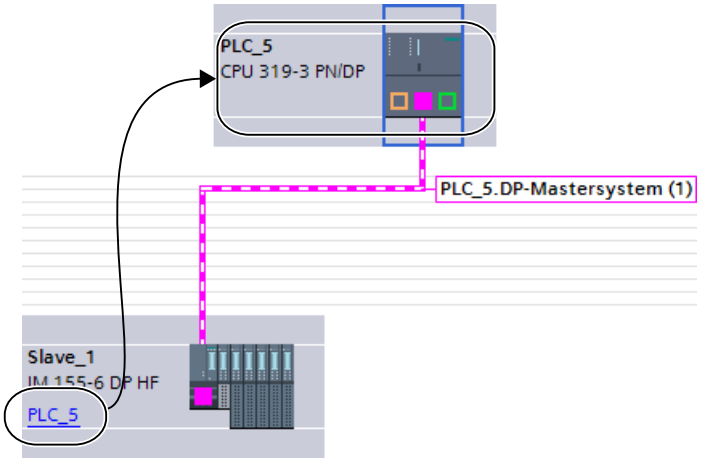

Highlighting applied to the DP master system

When you create a new DP master system, highlighting will be applied to it. This enables you to identify quickly which devices belong to the DP master system. You can also highlight a DP master system yourself by moving the mouse pointer over a subnet. This will result in the names of the available DP master systems being displayed. Click the required DP master system to highlight it.

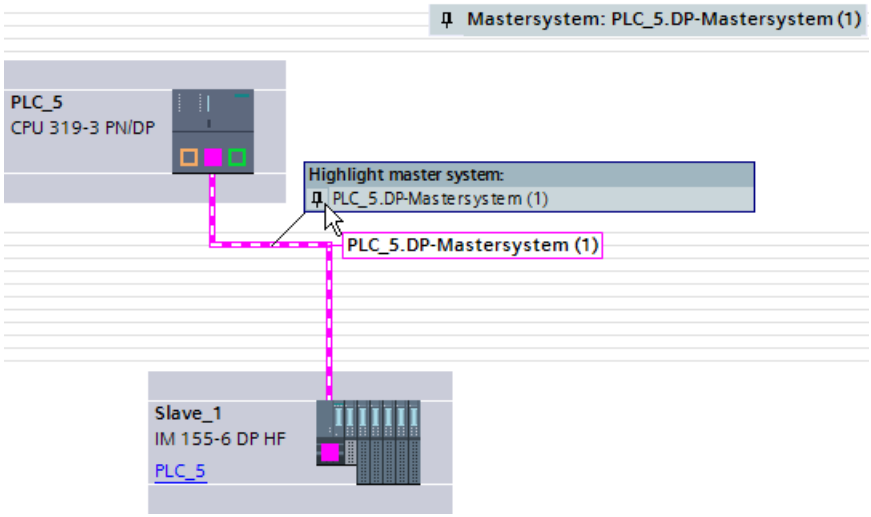

There are various ways of removing the highlighting from a DP master system:

- Highlight a different DP master system.
- Click the drawing pin with the name of the DP master system in the top right-hand corner of the network view.

See also

DP/DP coupler in the hardware catalog (Page 2730)

The basics of configuring a DP master system (Page 2728)

Editing DP master systems and interfaces

Introduction

Once you have created a DP master system, you also have the option of disconnecting the DP master system and its components. This can result in subnets with DP slaves but no DP master.

Generally, there is no need to edit the interfaces of a DP master.

You can change the name and number on the DP master system.

Disconnection from DP master systems

If you have configured either a CPU with an integrated PROFIBUS DP interface or a PROFIBUS CP (that can be configured as an intelligent DP slave) as a DP master with master system, then you will be able to disconnect the DP master from the DP master system. After this, the device will no longer be connected to the DP master system.

Disconnecting the subnet for a DP master effectively eliminates the master system in the sense that it is no longer assigned to a DP master. However, the individual DP slaves remain interconnected via the subnet and any direct data exchange that has been configured remains unaffected.

If you delete the DP slaves or detach them from the master system, the master system is retained on the DP master.

Requirement

- You must be in the network view.
- There has to be a DP master system with one DP master and at least one DP slave.

Disconnecting the DP master from the DP master system

To disconnect the DP master system, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Disconnect from master system" from the shortcut menu.

The selected DP master will be disconnected from the DP master system. A subnet with the DP slaves will be retained.

Adding a DP master to the DP master system

To reassign a DP master to a subnet, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Create master system" from the shortcut menu.
3. Draw connecting lines between the new DP master system and the DP interfaces of the DP slaves.

The DP master combines with the DP slaves to recreate a DP master system.

Ensuring the online capability of PROFIBUS DP interfaces

To enable a CPU's integrated MPI/DP interface to actively participate in PROFIBUS DP communication, proceed as follows:

1. Select the CPU interface symbol in the device or network views.
2. In the inspector window, select the parameter "Interface type: PROFIBUS" under "MPI address"..
3. If necessary change the suggested PROFIBUS address.
4. If required and available, select an existing subnet under "Subnet".

Note

If you use the mouse to draw a connecting line from the PROFIBUS interface of a DP slave to the MPI interface of a DP master, the MPI interface of the DP master will automatically be configured as a PROFIBUS interface and a DP master system will be created.

Editing the properties of a DP master system

To edit the properties of a DP master system, proceed as follows:

1. Move the mouse pointer over a subnet with a DP master system.
2. A message will appear displaying the available DP master systems. Click the one you want to edit. The DP master system will now be color-highlighted.
3. Now click the DP master system that has been color-highlighted.
4. In the inspector window, edit the DP master system attributes under "Properties > General".

Note

If you click a subnet when no DP master system is highlighted, you will be able to edit the properties of the entire subnet under "Properties" in the inspector window.

Adding DP slaves to the master system and configuring them

In the network view, you can add various DP slaves from the hardware catalog directly by using the drag-and-drop function or by double-clicking.

Types of DP slaves

For configuration purposes, DP slaves are broken down into the following categories:

- Compact DP slaves
  (Modules with integrated digital/analog inputs and outputs, for example, ET 200L)
- Modular DP slaves
  (interface modules with S5 or S7 modules assigned, for example ET 200M)
- Intelligent DP slaves (I-slaves)
  (SIMATIC S7-300 with, for example, CP 342-5, CPU 315-2 DP, or ET 200S with IM 151-7 CPU)

Slot rules

- Your DP master system should only contain one DP master, but it may contain one or or more DP slaves.
- Your DP master system must not exceed the number of DP slaves specified on the DP master.

Note

When configuring the DP master system, remember to observe the DP master technical data (max. number of nodes, max. number of slots, max. quantity of user data). Slot or user data restrictions may prevent you from being able to use the maximum number of nodes that is theoretically possible.

Requirements

- You are in the network view.
- A DP master system must have been created.

Adding a DP slave to the DP master system

To add a DP slave from the hardware catalog to the DP master system, follow these steps:

1. Select a modular DP slave (for example ET 200M) from the hardware catalog.
2. Drag-and-drop the DP slave from the hardware catalog into the network view.
3. Draw a connecting line between the DP master's DP interface or a highlighted DP master system and the DP interface of the new DP slave.

A DP master system will be generated and the DP slave will be connected to the DP master automatically.

Note

When a DP master system is highlighted, you can double-click the required DP slave in the hardware catalog. This will result in the DP slave being added to the highlighted DP master system automatically.

Disconnecting a DP slave from the DP master system

To disconnect a DP slave from the DP master system, follow these steps:

1. In the network view, right-click on the DP slave's DP interface.
2. From the shortcut menu, select the method for disconnecting from the DP master system:
   – "Disconnect from subnet": The PROFIBUS connection is broken and the device is no longer connected to the DP master system or a subnet.
   – "Disconnect from master system": The DP slave remains connected to the subnet, but is no longer assigned to the DP master system as a DP slave.

The selected DP slave will be disconnected from the DP master system.

You can also disconnect directly using the shortcut menu of the DP slave:

1. In the network view, right-click on the DP slave.
2. Select "Disconnect from DP master/IO system" from the shortcut menu.

The DP slave remains connected to the subnet, but is no longer assigned to the DP master system as a DP slave.

Assigning a DP slave to a new DP master system

To assign an existing DP slave to a new DP master system, follow these steps:

1. Right-click on the DP slave's DP interface.
2. From the shortcut menu, select "Assign to new master".
   It does not matter whether the DP slave concerned is already assigned to another DP master system.
3. From the selection list, select the DP master with the DP master system to which you want to connect the DP slave.
   The selected DP slave is now assigned to a new DP master system.

The "Assign to new subnet" function works in a similar way in that it allows you to connect a DP slave to a new subnet. However, in this case, the DP slave will not be connected to an existing DP master system.

You can also make the assignment directly using the shortcut menu of the DP slave:

1. Right-click the DP slave.
2. From the shortcut menu, select "Assign new DP master/IO system".
3. Select the DP master from the list of available DP masters.

The selected DP slave has now been assigned to a new DP master system.

Note

If there is only one DP master on a subnet, you only have to connect the interface of the DP slave to the subnet of this DP master with drag-and-drop. The DP slave is then connected to the subnet of the DP master immediately, and the DP slave is assigned to this DP master.

Configuring a DP slave

To configure a DP slave, follow these steps:

1. Switch to the DP slave's device view.
2. Select the desired modules from the hardware catalog. In the case of modular DP slaves, the possible modules can be found within the hardware catalog underneath the relevant DP slave "family". You should, therefore, use the hardware catalog's filter function to facilitate the process of locating them.
3. Drag-and-drop the required modules from the hardware catalog onto the rack of the DP slave.
4. Configuring the DP slave and inserted modules in the inspector window.

See also

Creating a DP master system (Page 2735)

DP slaves within the hardware catalog (Page 2730)

Hint: Quick configuration of master systems

If the DP master system has several DP slaves, use drag-and-drop operation to assign to the master in one step all DP slaves that were placed.

Requirement

DP master and DP slaves are placed in the network view.

Assigning DP slaves to a DP master system

To do this, follow these steps:

1. Select an appropriate zoom factor so that you can see as many DP slaves as possible in the network view.
2. Arrange the DP slaves in a maximum of two rows.

3. Select all DP interfaces with the mouse cursor (not all devices!). This only works if you begin to drag the mouse cursor outside of the first DP slave and release the mouse button at the last DP slave (selection with the lasso).

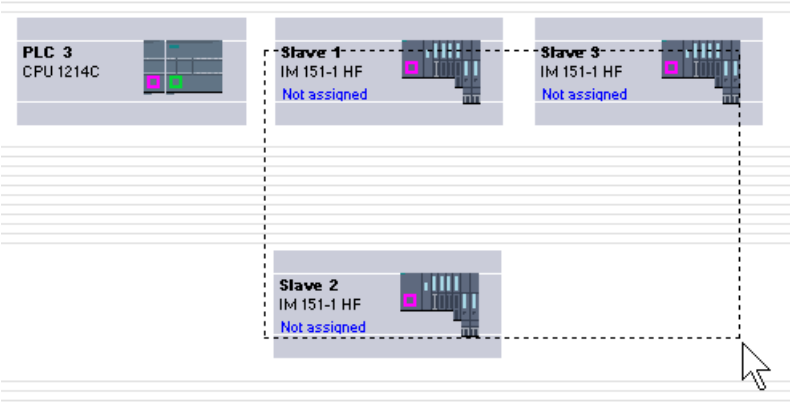

4. Select the shortcut menu "Assign to new master" and select the corresponding DP interface for the DP master in the subsequent dialog.

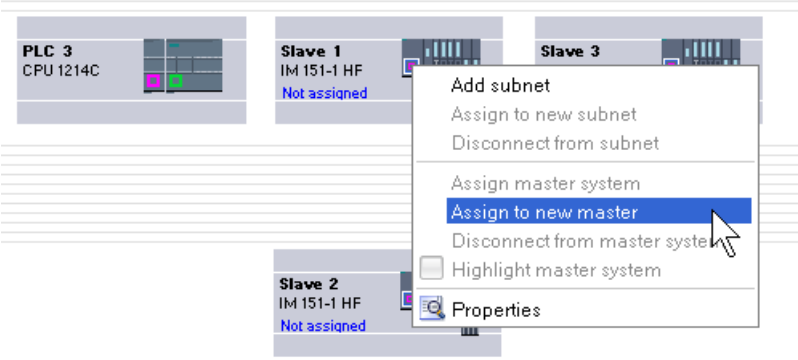

5. The DP slaves are automatically networked with the DP master and combine with it to form a DP master system.

Note

When a DP master system is highlighted, you can double-click on a DP slave in the hardware catalog and thereby quickly add additional DP slaves. This will result in the DP slave being added to the highlighted DP master system automatically.

Assigning a DP slave to a SYNC/FREEZE group

Introduction

Assuming that it supports this function, a DP master can send the SYNC and/or FREEZE control commands to a group of DP slaves simultaneously in order to synchronize them. If you require this, you will need to assign the DP slaves to SYNC/FREEZE groups.

Requirement

A DP master system must have been created.

Procedure

To assign a DP slave to a SYNC/FREEZE group, follow these steps:

1. In the device or network views, select the DP slave interface that you want to assign to a group.
2. In the inspector window under "Sync/Freeze" group, select the check box for the Sync/Freeze group you want.

Note

You can assign a DP slave to a maximum of one SYNC/FREEZE group.

Exception: If you are using a CP 342-5 as the DP master, you can assign up to eight SYNC/FREEZE groups to each assigned DP slave.

Please also note the documentation relating to the CP 342-5.

What you need to know about the SYNC and FREEZE control commands

Using the SYNC and FREEZE control commands on the DP master, you can synchronize the DP slaves on an event-driven basis. The DP master simultaneously sends the control commands to all the DP slaves that make up a group within its master system. Any DP slaves that have failed or currently have diagnostics pending will be ignored.

Before DP slaves can be synchronized using control commands, you must first assign the DP slaves to SYNC/FREEZE groups.

For an S7 CPU, use the instruction DPSYC_FR (SFC 11) to synchronize the DP slaves.

When you select the DP master, a list of the eight SYNC/FREEZE groups is displayed in the Inspector window under "Properties > DP interface > SYNC/FREEZE" or "Properties> MPI/DP interface > SYNC/FREEZE" (when operating as a PROFIBUS interface).

Note

There are DP slaves which are only SYNC-capable or only FREEZE-capable. Refer to the "SYNC/FREEZE" area in the DP interface properties to find out which of these functions a DP slave supports.

In order to be able to assign to such a DP slave of a SYNC or FREEZE group, you need to restrict the affected group to the function that the DP slave supports in the DP master interface properties.

Example: For a slave that is only FREEZE-capable, you need to clear the "SYNC" checkbox in the SYNC/FREEZE group of the DP master.

The SYNC control command

The DP master sends the SYNC command to freeze the output states on a group of DP slaves so that they retain the current value.

When subsequent message frames arrive, the DP slaves will store the output data from the DP master. However, the DP slave output states will remain unchanged.

Then, when a new SYNC control command is sent, each DP slave will align its outputs with the values of the output data from the DP master.

Cyclic updating of the outputs will only resume when the DP master sends the UNSYNC control command.

The FREEZE control command

When they receive the FREEZE control command from the DP master, the DP slaves within the relevant group will freeze the current state of their inputs The DP slaves send this frozen input data to the DP master in the following cyclic frames.

Whenever they receive a new FREEZE control command, the DP slaves will freeze the current state of their inputs again.

Cyclic transfer of the state of the DP slave inputs from the DP slave to the DP master will only resume when the DP master sends the UNFREEZE control command.

See also

Creating a DP master system (Page 2735)

Setting PROFIBUS properties (Page 2801)

1.4.9.7 Example: Configuring ET 200S as DP slave

Configuring an ET 200S

Introduction

DP slaves and I/O devices from the ET 200S family are configured in the same way as other modular DP slaves and I/O devices. The modules for the ET 200S are available in the hardware catalog under "Distributed I/O".

Slot rules for configuring an ET 200S

The following rules apply when configuring an ET 200S:

- Do not leave any gaps when inserting the ET 200S modules.
- Slot 1: only for PM-E or PM-D Power Modules.
- To the left of an Electronics Module (EM): an EM or a Power Module (PM-E or PM-D) only.
- To the left of Motor Starter (MS): an MS, a PM-D, PM-D Fx (1..x..4) Power Module or a PM-X Power Module only.

- To the left of a PM-X: a motor starter or a PM-D only
- Up to 63 modules and one IM Interface Module are permitted

Note

Remember to ensure that the correct PM-E and EM voltage ranges are assigned.

Operation when expected and actual configurations do not match

This parameter controls the response during startup, and when modules are removed and inserted during operation (cyclical data exchange).

Check box is selected (operation is permitted if preset configuration is not the same as actual configuration):

- The module will start up even if the preset configuration is not equal to the actual configuration.
- The module remains in cyclical data exchange following removal and insertion (not a CPU stop).

Check box is cleared (operation is not permitted if preset configuration is not the same as actual configuration):

- The module will not start up (does not switch over to cyclical data exchange) if the preset configuration is different from the actual configuration.
- If modules are removed and inserted during operation, this causes a CPU stop, and the module no longer participates in the cyclical data exchange.

Refer to the manual for the respective module for information on which constraints cause a CPU stop, or prevent the data exchange starting, even if operation is enabled when preset configuration is not the same as the actual configuration.

Interference frequency suppression

The frequency of your alternating current network can interfere with the measured value in the following cases in particular: measurement in small voltage ranges and measurement with thermocouples. Set the frequency of the power for the entire ET 200S to that which predominates in your plant.

The analog electronic modules AI set their integration time or conversion time in relation to the selected interference frequency suppression.

Configuring reference junctions

A reference junction is the connection of a thermocouple to a supply line (generally in the terminal box). The voltage that occurs due to the effects of temperature falsifies the temperature value measured by the module.

On the ET 200S, one channel of the AI RTD module can be programmed as a reference junction. Other AI TC modules can correct their measured values using the temperature at the reference junction as measured by this module.

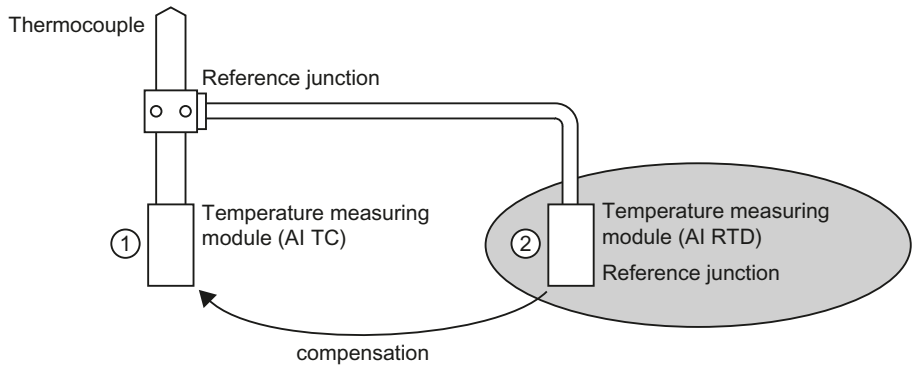

① Configuring the AI TC:
→ Selection of the reference junction used

② Configuring of the AI RTD:
→ Activation of the reference junction
→ Specifying the slot and channel of the AI RTD

Special characteristics to be noted when assigning parameters for reference junctions

The process of assigning parameters for reference junctions will be explained based on the example of a resistance thermometer with a Pt 100 climatic range that is used for measuring the reference junction temperature.

To assign parameters for the reference junction, follow these steps:

1. In the ET 200S device view, insert an analog RTD electronics module, for example a 2AI RTD HF. The module must support the "thermal resistor" measuring type (linear, 4-wire) and the "Pt 100 climatic range" measuring range.
2. Select the module on the rack.
3. Under "Properties > General > Inputs" in the inspector window, set a channel for the reference junctions function.
   Measuring type: Thermal resistor (linear, 4-wire)
   Measuring range: Pt 100 climatic range
4. Select the ET 200S (that is, the interface module).
5. Under "Properties > General > Module parameters >Reference junctions" in the inspector window, select the "Reference junction" check box and specify the slot and the channel number of the relevant RTD module.
6. Insert the analog electronics module for measuring the temperature using a thermocouple (TC module) and select the option "RTD" as a reference junction under "Properties > General > Inputs > Channel x" in the inspector window.

Additional information

For additional information on the various types and uses of ET 200S modules, please refer to the operating instructions and the manual titled "ET 200S Distributed I/O System".

For additional information on analog value processing, please see the documentation for the ET 200S distributed I/O system.

See also

Configuring the ET 200S in DPV1 mode (Page 2753)

Documentation on ET 200S (http://support.automation.siemens.com/WW/view/en/1144348)

Packing addresses

Introduction

DP slaves and I/O devices from the ET 200S family are configured in the same way as other modular DP slaves and I/O devices. As well as supporting all the standard modular DP slave and I/O device functions, the ET 200S also offers the "Pack addresses" function:

When digital electronics modules requiring an address space of 2 or 4 bits are inserted into the device view, they will initially be spread over a total area of 1 byte. However, the address area actually occupied can be compressed after configuration using the "Pack addresses" function.

	Initial state	After "Pack addresses"
Module	I address	I address
2DI (2-bit)	Byte 10	10.0 to 10.1
4DI (4-bit)	Byte 11	10.2 to 10.5

Requirement

- You are in the device view.
- An ET 200S, for example an IM 151-1, must be present.
- A pair of digital electronics modules, for example 2DI AC120V ST, must be inserted into the slots.

Packing addresses

To pack addressed, follow these steps:

1. Select the electronics modules whose addresses are to be packed. The following options are available for selecting multiple electronics modules:
   - Press and hold down <Shift> or <Ctrl> while clicking the relevant electronics modules.
   - Click off the rack and select the required electronics modules by drawing round them with the mouse.
2. Click "Pack addresses" in the shortcut menu for the selected electronics modules.

The address areas for inputs, outputs and motor starters are packed separately. The packed addresses will be displayed in the I address and Q address columns of the device overview.

How packed addresses are generated and structured

If you make use of the "Pack addresses" function, the addresses of the selected electronics modules will be packed in accordance with the following rules:

- The start of the address area is determined by the lowest address of the selected electronics modules: X.0
- If the bit address is not "0", then the next (free) byte address as of which the selected area can be compacted will be selected automatically: (X+n).0
- If no coherent area exists, the addresses will be automatically packed into any available address gaps.

Electronics modules with packed addresses and the same byte address form a packing group.

Unpacking addresses

To unpack addressed, follow these steps:

1. Select one or more electronics modules with packed addresses.
2. Click "Unpack addresses" in the shortcut menu for the selected electronics modules.

The packing groups of the selected electronics modules will be disbanded and the packed addresses for the relevant electronics modules unpacked.

The packing group will also be disbanded and the packed addresses unpacked in the following cases: if you delete electronics modules from a packing group, move electronics modules out of a packing group or insert electronics modules on a free slot within a packing group.

The start addresses of the unpacked electronics modules will be assigned to the next available byte addresses in each case.

Special characteristics of electronics modules with packed addresses

The following special characteristics apply to electronics modules with packed addresses:

- As far as the CPU is concerned, there is no way of assigning a slot for the electronics module. Consequently, the instruction GADR_LGC (SFC 5) outputs error information W#16#8099 "Slot not configured" for the actual slot of the electronic module.
- The instruction LGC_GADR (SFC 49) and SZL-ID W#16#xy91 "module status information" cannot be evaluated for an electronics module.
- The electronics module receives an additional diagnostics address via the DPV1 function, because otherwise the packed addresses would prevent interrupts from being assigned as far as the CPU is concerned.
- The "Insert/remove interrupt" is not possible. This is because the "Pack addresses" and "Insert/remove interrupt" functions are mutually exclusive.

Configuring option handling with standby modules

You can use the option handling function to prepare the ET 200S with PROFIBUS interface for future expansions (options). This section describes option handling with standby modules.

You do this by assembling, wiring, configuring, and programming the maximum configuration envisaged for the ET 200S and by using cost-effective standby modules (138-4AA00 or 138-4AA10) during assembly until it becomes time to replace them with the necessary electronics modules.

Note

The ET 200S can be completely factory-wired with the master cabling, as no connection exists between a standby module and the terminals of the terminal module (and, in turn, to the process).

Requirement

- ET 200S interface module
  - IM 151-1 STANDARD (6ES7 151-1AA03-0AB0 or higher)
  - IM 151-1 FO STANDARD (6ES7 151-1AB02-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the numbered check boxes for the slots that are initially to accommodate the standby modules prior to the future electronics modules.
3. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).

The assembled standby modules can be replaced with the configured modules at a later date without having to modify the configuration.

Note

The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 STANDARD Interface Module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of standby modules, please refer to the documentation for the ET 200S distributed I/O system.

How option handling works during startup

If "Startup when expected/actual config. differ" is not available, the ET 200S will still start up if a standby module is inserted instead of the configured electronics module and option handling has been activated for the slot concerned.

How option handling works during operation

During operation, the option handling mode varies in accordance with the following:

- Option handling enabled for a slot:
  Either the standby module (option) or the configured electronics module can be plugged into this slot. If any other kind of module is present on this slot, diagnostics will be signaled (no module/incorrect module).
- Option handling disabled for a slot:
  Only the configured electronics module can be plugged into this slot. If any other kind of module is present, diagnostics will be signaled (no module/incorrect module).

Standby module substitute values

- Substitute value for digital inputs: 0
- Substitute value for analog inputs: 0x7FFF

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 2 to 63:

- Bit value = 0: Option handling parameterized. Standby modules are permitted.
- Bit value = 1: Option handling canceled. Standby modules are not permitted on this slot.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: This slot has the standby module, an incorrect module or no module plugged into it.
- Bit value = 1: The configured module is plugged into this slot.

See also

Which modules support option handling? (http://support.automation.siemens.com/WW/view/en/22564754)

Configuring option handling without standby modules

You can use the option handling function to prepare the ET 200S for future expansions (options) without installing standby modules. This section describes option handling without standby modules.

Note

ET 200S with PROFINET interface

This description refers to the ET 200S with PROFIBUS interface. In principle, option handling for ET 200S with PROFINET interface functions as described here without standby modules. PN interface modules are to be used instead of the DP interface modules listed here. You can find additional information about option handling for ET 200S with PROFINET interface in the corresponding manuals.

Requirement

- ET 200S interface module
  - IM 151-1 HIGH FEATURE (6ES7151-1BA02 or higher)
  - IM 151-1 STANDARD (6ES7 151-1AA05-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).
3. Configure the slave's maximum configuration. The selection/clearing of options is controlled via the user program.

   Note

   The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 interface module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

   Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

   Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of standby modules, please refer to the documentation for the ET 200S distributed I/O system.

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 1 to 63:

- Bit value = 0: Slot is not available in the actual configuration.
- Bit value = 1: Slot is available in the actual configuration.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: Slot belongs to an option that is not available or module status is faulty.
- Bit value = 1: The configured module is plugged into this slot.

See also

Sample applications for ET 200S, option handling without standby module (http://support.automation.siemens.com/WW/view/en/29430270)

Configuring the ET 200S in DPV1 mode

PROFIBUS DPV1 enables you to access extended PROFIBUS functions.

Requirement

- You must be in the network view.
- An DP master with DPV1 functionality must be available.
- A master-slave connection must be established with PROFIBUS.

Procedure

To switch the DP slave over to DPV1, follow these steps:

1. Select the DP slave.
2. Under "Properties > Module parameters" in the inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list box.

or

1. Select the DP master.
2. In the I/O communications table select the row with the connection between the DP master and the desired DP slave.
3. Under "Properties > Module parameters" in the inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list box.

Special characteristics

The parameters are subject to interdependencies, which are outlined below:

Parameter	DPV0 mode	DPV1 mode
Operation when target configuration does not match actual configuration	Fully available	Fully available
Diagnostics interrupt (OB 82)	Not available, not set	Fully available
Hardware interrupt (OB 40 to 47)	Not available, not set	Fully available
Insert/remove interrupt (OB 83)	Not available, not set	Only available when addresses are not packed. "Startup when target configuration does not match actual configuration" is activated automatically along with an insert/remove interrupt.

Interrupts in the case of modules with packed addresses

If the module is capable of triggering interrupts and the bit address is not equal to 0 because of packed addresses, you will need to assign a diagnostics address in the ET 200S address dialog.

The diagnostics address is required for the purpose of assigning a DPV1 interrupt to the module as an interrupt trigger. The CPU will only be able to assign an interrupt correctly and store information on the interrupt in the interrupt OB start information/in the diagnostics buffer if the module concerned has this "unpacked" address. In this context, the CPU cannot make use of "packed" addresses.

From the point of view of interrupt processing (interrupt OB), this means the module will have the assigned diagnostics address, but from the point of view of processing the input and output data in the user program, the module will have the packed addresses.

Note

When the module addresses are packed, the insert/remove interrupt for the ET 200S is unavailable.

See also

Introduction to PROFIBUS DPV1 (Page 2779)

Configuring DPV1 devices (Page 2780)

Configuring an ET 200S (Page 2745)

Additional configurations

Configuring additional functions

The ET 200S distributed I/O system has numerous additional functions that are available via plug-in modules (e.g. communication module). You can find the descriptions for these in the sections "Using technology functions" and "Configuration for point-to-point connections".

Additional distributed I/O systems

Information on additional distributed I/O systems is available by following further links under "See also".

1.4.9.8 Configuring intelligent DP slaves

Adding an I-slave to a DP master system

Introduction

One of the key characteristics of intelligent DP slaves (I-slaves) is that the DP master is not provided with I/O data directly by a real I/O, but by a preprocessing CPU. Together with the CP, this preprocessing CPU forms the I-slave.

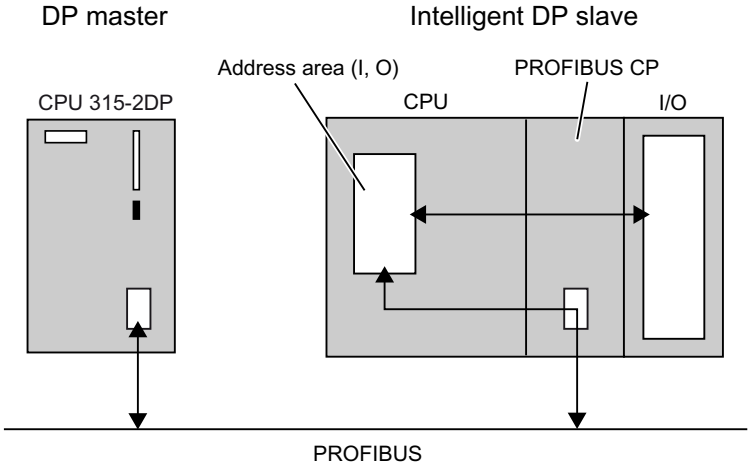

Difference: DP slave v. intelligent DP slave

In the case of a DP slave, the DP master accesses the distributed I/O directly.

In the case of an intelligent DP slave, the DP master actually accesses a transfer area in the I/O address space of the preprocessing CPU instead of accessing the connected I/O of the intelligent DP slave. The user program running on the preprocessing CPU is responsible for ensuring data exchange between the address area and I/O.

Note

The I/O areas configured for data exchange between the DP master and DP slaves must not be used by I/O modules.

Applications

Configurations involving intelligent DP slaves:

- I-slave <> DP master data exchange
- Direct DP slave > I-slave data exchange

Procedure

To add an I-slave to a DP master system, proceed as follows:

1. From the hardware catalog, drag two devices with PROFIBUS DP interface for configuration as DP master and I-slave (e.g. CPU 317-2 DP) into the network view.
2. Switch the device that is to be used as the I-slave over to "Slave" mode under "Properties > DP interface" in the Inspector window. The device will now operate as an intelligent DP slave.
3. Draw a connecting line between the DP interfaces of both devices. This way you connect the I-slave with a DP master in a DP master system.

Result

You have now set up a DP master system with one DP master and one I-slave.

Please note the following I-slave configuration examples:

- Configuring a CP as an I-slave (Page 2761)
- Configuring a CPU as an I-slave (Page 2762)
- Configuring a DP slave as an I-slave (Page 2763)

See also

Interrupts on the I-slave (Page 2759)

Configuring diagnostics addresses (Page 2757)

Configuring diagnostics addresses (S7-300, S7-400)

Introduction

When assigning diagnostics addresses for PROFIBUS DP, you must ensure that DP diagnostics addresses are assigned to both the DP master and the I-slave.

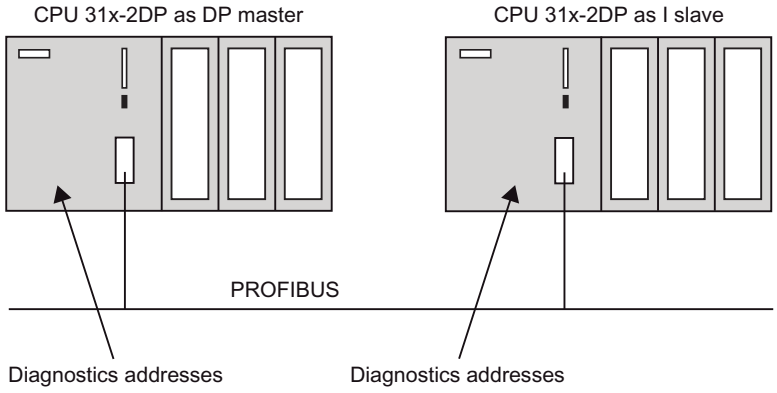

Diagnostics addresses

A distinction is made between diagnostics addresses on the DP master and those on the I-slave:

- Diagnostics addresses on the DP master
  The DP master receives information about the status of the DP slave or a bus interruption via the diagnostics addresses assigned to the DP master.
- Diagnostics addresses on the I-slave
  The I-slave receives information about the status of the DP master or a bus interruption via the diagnostics addresses assigned to the I-slave.

STEP 7 automatically assigns diagnostics addresses for the respective CPUs when the master-I-slave configuration is specified. The following describes how to change the specified diagnostics addresses.

Tip: In order to obtain an overview of all addresses assigned for the CPU, including diagnostics addresses, select the "Address overview" area in the inspector window of the selected CPU.

Diagnostics addresses of the I-slave communication

A table with the diagnostics addresses of the I-slave communication is displayed on the I-slave. The identifier for the diagnostics address is the name of the PLC and the transfer area with the lowest subslot number. You can edit the diagnostics addresses used in the table.

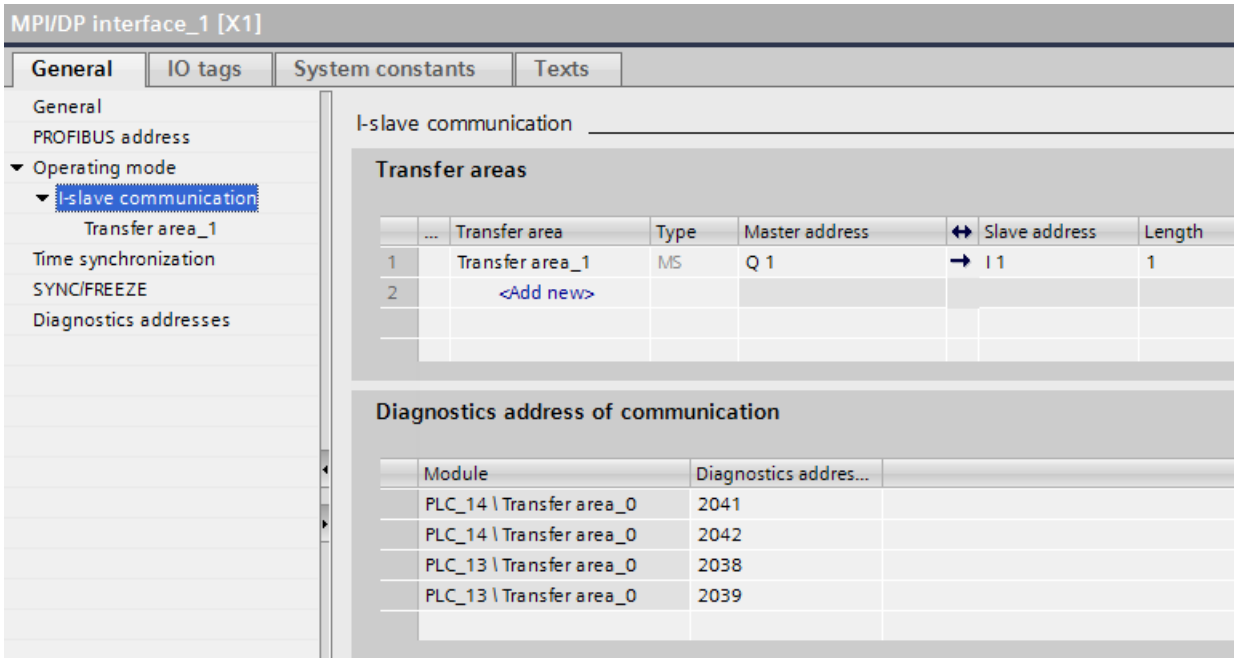

The table is only displayed when at least the DP master or the I-slave is a CPU 300/400:

DP master	I-slave	Display of the diagnostics addresses
CPU 300/400	CPU 300/400	Local and remote addresses of the I-slave
CPU 300/400	CPU 1200/1500	Remote addresses of the I-slave
CPU 1200/1500	CPU 300/400	Local addresses of the I-slave
CPU 1200/1500	CPU 1200/1500	No display

Requirement

- You must be in the network view.
- You must have set up a DP master system with a DP master and an I-slave.

Procedure

To edit the diagnostics addresses for the I-slave communication, follow these steps:

1. Select the DP master or the I-slave.
2. In the I/O communication table, select the line showing "I-slave" mode.
3. Edit the diagnostics addresses under "Properties > I-slave communication > Diagnostics address of communication" in the Inspector window.

Or:

1. Select the I-slave.
2. Edit the diagnostics addresses under "Properties >DP interface > Operating mode > I-slave communication > Diagnostics address of communication" in the Inspector window.

You can select and edit the fields containing the values of the diagnostics addresses both for the slave and the master.

Additional information

For further information on diagnostics addresses, refer to the manuals for the individual S7 CPUs.

Interrupts on the I-slave

Types of interrupt according to DPV1 and DPV0 modes

I-slaves with the advanced instruction "SALRM" (SFB 75) can trigger interrupts on the associated DP master. The following table lists the possible types of interrupt according to the PROFIBUS DPV1 and DPV0 modes:

Type of interrupt	DPV0	DPV1
Diagnostic interrupt (OB 82)	Yes	Yes
Hardware interrupt (OB 40 to 47)	Yes	Yes
Pull/plug interrupt (OB 83)	Yes (if I-slave supports this interrupt)	Yes
Status interrupt (OB 55)	No	Yes
Update interrupt (OB 56)	No	Yes
Manufacturer-specific interrupt (OB 57)	No	Yes

Addresses for triggering interrupts

Use the addresses configured in the I-slave's inspector window under "Properties > DP interface > Mode > I-slave communication" to trigger interrupts with advanced instruction "SALRM". Rather than being assigned to physical modules, these addresses correspond to virtual "slots".

You cannot use the addresses relating to operating mode transitions and device diagnostics for the purpose of triggering interrupts.

How interrupt generation works

The principle of interrupt generation can be explained using the example of a diagnostics interrupt.

- On the I-slave, the output address 0 is assigned to a virtual slot.
- In the example, the output address 0 is used to trigger a diagnostics interrupt (OB 82) on the DP master.

Diagnostics interrupts are accompanied by data relating to individual user programs (AINFO). This data must correspond to the basic structure for supplementary interrupt information.

For example, the following simplified structure is possible:

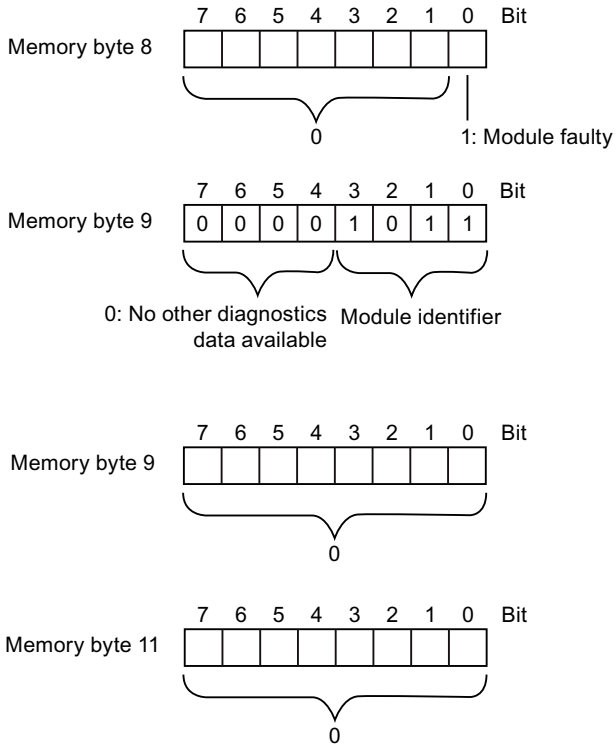

Additional information

Note the additional information on the structure presented above in the reference manual for the S7-300/400 system and standard functions, Volume 2, "Diagnostic data" section, and in the manuals for the system and diagnostic functions of the S7-1500.

Note

The supplementary interrupt information affects the module status data and the group error LED of the I-slave (S7-300/400). The interrupt also affects the module status data and error LEDs of the associated DP master. Therefore, you will need to take account of what the diagnostic data records mean (data record 0 and data record 1) when compiling data for the supplementary interrupt information.

The figure below illustrates the interrupt generation process:

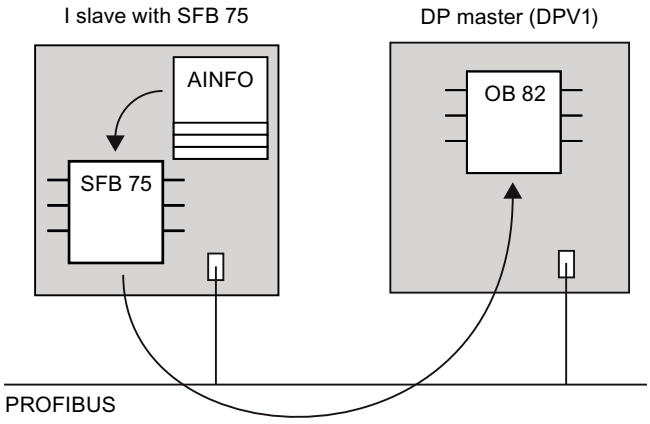

The advanced instruction "SALRM" can be called as follows:

STL

```
CALL   FB    SALRM , DB75
       REQ   :=M0.0
       ID    :=DW#16#8000    //Output address 0
       ATYPE :=1             //Diagnostics interrupt
       ASPEC :=1             //Incoming interrupt, EG faulty
       LEN   :=4             //Interrupt info is 4 bytes in length
       DONE  :=M1.3
       BUSY  :=M1.4
       ERROR :=M1.5
       STATUS:=MD50
       AINFO :=P#M 8.0 BYTE 4 //Area for interrupt info
```

See also

Introduction to PROFIBUS DPV1 (Page 2779)

Configuring DPV1 devices (Page 2780)

Example of how to configure a CP as an I slave

In this example, you are going to configure a station with a CP 342-5 as an I-slave. In order to do this, you will need to turn the CP 342-5 into an intelligent DP slave by means of the "DP slave" mode.

Requirements

- You must be in the network view.
- There must be a CP 342-5 available in the network view.
- The CP 342-5 must have been populated with I/O modules in the device view.
- A DP master (CPU with integrated PROFIBUS DP interface or CP with PROFIBUS DP interface) is available in the network view.

Procedure

To configure a CP as an I-slave, proceed as follows:

1. Select the CP 342-5.
2. In the Inspector window, switch the CP 342-5 to "DP slave" mode under "Properties > Mode".
3. Draw a connecting line between the CP 342-5's DP interface and the DP interface of the DP master.
4. Select the CP 342-5.
5. Click on "I-slave communication" under "Properties > Mode" in the Inspector window.
6. Edit the address assignment between the DP master and I-slave.

Alternatively, after selecting the CP 342-5 in the I/O communications table, you can call the line with the "I-slave" mode in order to edit the address assignment under "Properties" and "I-slave communication" in the Inspector window.

Additional information

For additional information on data exchange between the preprocessing CPU and the DP slave's internal CP 342-5 DP, please refer to the manual that deals with NCM S7 Industrial Ethernet.

Example of how to configure a CPU as an I slave

In this example, you are going to configure a station with a CPU 31X-2 DP featuring an integrated DP interface (a CPU 315-2 DP, for example) as an I-slave. The CPU 31X-2 DP must have an integrated DP interface, and you will need to turn it into an intelligent DP slave by activating "DP slave" mode.

Requirements

- You must be in the network view.
- There must be a CPU 315-2 DP available in the network view.
- The CPU 315-2 DP must have been populated with I/O modules in the device view.
- A DP master (CPU with integrated PROFIBUS DP interface or CP with PROFIBUS DP interface) is available in the network view.

Procedure

To configure a CPU as an I-slave, proceed as follows:

1. Select the CPU 315-2 DP.
2. In the Inspector window, switch the CPU 315-2 DP to "DP slave" mode under "Properties > DP interface > Mode".
3. Draw a connecting line between the DP interface of the CPU 315-2 DP and the DP interface of the DP master.

4. Select the CPU 315-2 DP.
5. Click on "I-slave communication" under "Properties > DP interface > Mode" in the Inspector window.
6. Edit the address assignment between the DP master and I-slave.

Alternatively, after selecting the CPU 315-2 DP in the I/O communications table, you can call the line with the "I-slave" mode in order to edit the address assignment under "Properties > I-slave communication" in the Inspector window.

Additional information

For additional information on configuring CPUs as I-slaves, please see the reference manuals for the relevant CPUs.

Example of how to configure an ET 200 CPU as an I-slave

In this example, you are going to configure an ET 200S with Interface Module IM 151-7 CPU as an I-slave.

Note

You can find the interface module of the ET 200S in the hardware catalog under "PLC > SIMATIC ET 200 PLC > ET 200S > CPU".

Requirements

- You must be in the network view.
- An ET 200S with IM 151-7 CPU is available in the network view.
- The ET 200S has been fitted with I/O expansion modules in the device view.
- A DP master (CPU with integrated PROFIBUS DP interface or CP with PROFIBUS DP interface) is available in the network view.

Procedure

To configure a DP slave as an I-slave, proceed as follows:

1. Select the IM 151-7 CPU.
2. Draw a connecting line between the DP interface of the IM 151-7 CPU and the DP interface of the DP master.

   Note

   Some stations have permanently specified interfaces for master or slave. Under "Properties > MPI/DP interface" and/or "Properties > DP interface" you will find the settings for the respective mode.

3. Select the IM 151-7 CPU.

4. In the I/O communications table, select the line showing "I-slave" mode.
5. Edit the address assignment between the DP master and I-slave under "Properties > I-slave communication" in the Inspector window.

Additional information

For additional information on configuring DP slaves as I-slaves, please refer to the manuals and operating instructions for the ET 200 distributed I/O systems.

Example: Configuring an I-slave (S7-1500, ET 200SP CPU)

Requirements for configuring an I-slave

The I-slave consists, for example, of the following:

- A CPU from S7-1500 and a communications module CM 1542-5/CP 1542-5 (STEP 7 as of V12)
- A CPU from ET 200SP and a communications module CM DP (STEP 7 as of V13 SP1)

Procedure for configuring an I-slave

This section shows how to configure an I-slave based on the example of a CPU 1512SP-1 PN. The procedure for the CPUs from S7-1500 with CM 1542-5/CP 1542-5 and for the CPU 1510SP-1 PN with CM DP is the same.

To configure an I-slave, follow these steps:

1. Drag a CPU 1512SP-1 PN from the hardware catalog to the network view.
2. Open the device view of the CPU.
3. Double-click on the CM DP communications module in the hardware catalog.
   STEP 7 creates the CM DP in the device view.
4. Select the PROFIBUS interface of the CM DP communications module.

5. In the Inspector window in the area navigation, select the "Operating mode" entry and enable the "DP slave" check box.
6. Now you have the option of selecting the DP master in the "Assigned DP Master" drop-down list.
   Once you have selected the DP master, the networking and the DP master system between both devices are displayed in the network view.

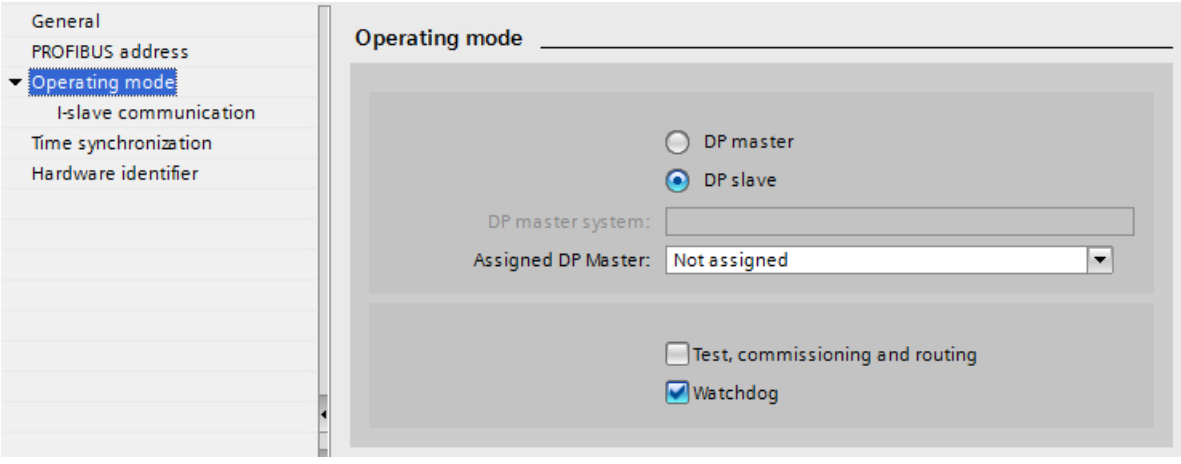

Note

Operation using a GSD file

If you operate an I-slave using a GSD file, you should not enable the "Test, commissioning and routing" check box.

Create a DP subnet on the PROFIBUS interface of the I-slave.

Example: Configuring transfer areas

Requirements for configuring transfer areas

- You have configured an I-slave in STEP 7.
- You are in the device view of the I-device and have selected the PROFIBUS interface of the communications module.

Procedure for configuring transfer areas

To configure transfer areas for an I-slave, follow these steps:

1. In the area navigation, go to the section "Operating mode" > "I-slave communication" > "Transfer areas".
2. Create transfer areas. Set the properties of the created transfer areas.

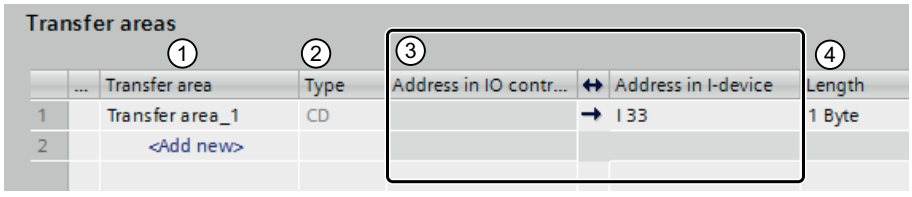

① Click in the first cell of the "Transfer area" column. STEP 7 assigns a default name that you can change.

② Select the type of communication relation: Currently, you can only select MS for "master-slave communications relation".

③ STEP 7 assigns the addresses for the transfer area automatically. Correct the addresses if necessary.

④ Set the length and unit of the transfer area. Specify the length of the transfer area in the following format: [1...64] bytes/word. Examples: "32 Byte", "64 Word"

⑤ Specify whether the transfer area is exchanged in units of bytes or words or consistently over the entire length between the DP master and I-slave.

A separate entry is created in the area navigation for each transfer area. If you select one of these entries, you can adjust the details of the transfer area, or correct them and comment on them.

Diagnostics and interrupt behavior (S7-1500, ET 200SP CPU)

Diagnostics and interrupt behavior

S7 CPUs have numerous diagnostics and interrupt functions that can, for example, report errors/faults or failures of underlying DP systems. These diagnostics alarms reduce downtimes and simplify localization and elimination of problems.

Diagnostics options in the higher-level DP master and in the I-slave

The following diagnostics mechanisms are available to the higher-level DP master and the I-slave:

- OB 82 (diagnostics interrupt)
  When the I-slave changes mode, the DP master calls OB 82 (diagnostics interrupt).
  When the DP master changes mode, the I-slave calls OB 82 (diagnostics interrupt).
- OB 86 (rack failure)
  If the bus connection to the I-slave is interrupted, the DP master calls OB 86 (rack failure).
  If the bus connection to the DP master is interrupted, the I-slave calls OB 86 (rack failure).
- OB 122 (I/O access error)
  If you have not set the attribute "Handle errors within block" for OB 122, the following applies:
  - If the bus connection to the I-slave is interrupted, the DP master calls OB 122 (I/O access error) if there is direct access to the relevant transfer areas.
  - If the bus connection to the DP master is interrupted, the I-slave calls OB 122 (I/O access error) if there is direct access to the relevant transfer areas.

Reaction of the transfer areas to mode changes

DP master	I-slave	Reaction of input transfer areas DP master	Reaction of input transfer areas I-slave
RUN→STOP	RUN	No updating of the process image	The input transfer areas retain their current values. (no access error)
STOP→RUN	RUN	The input transfer areas are updated by the cyclic user program via the process image.	The input transfer areas are updated via the process image.
RUN	RUN→STOP	The I-slave sets the input transfer areas on the DP master to "0".	No updating of the process image
RUN	STOP→RUN	The I-slave sets the input transfer areas on the DP master to "0". After the startup program of the I-slave, the input transfer areas of the DP master are updated via the process image.	Prior to processing of the startup program, the input transfer areas are updated via the process image.

1.4.9.9 Configuring direct data exchange

What you need to know about direct data exchange

Introduction

Direct data exchange (data exchange broadcast) is another form of I/O communication, which is not restricted to the context of a DP master system. With direct data exchange, the receiver has read-only access (in other words it cannot write data).

In a DP master system, a DP master only controls the DP slaves that have been assigned to it. With direct data exchange, a DP master or a DP slave configured as a receiver on the PROFIBUS subnet can "listen in" to find out what input data another DP slave is sending to

its DP master in a sender capacity. This method of transferring data between DP slaves is referred to as direct data exchange.

Principle of operation

In a direct data exchange configuration, local input address areas of an intelligent DP slave or of a DP master are assigned to the input address areas of a PROFIBUS DP partner.

The intelligent DP slave or DP master can then use these input address area assignments to receive data from its PROFIBUS DP partner.

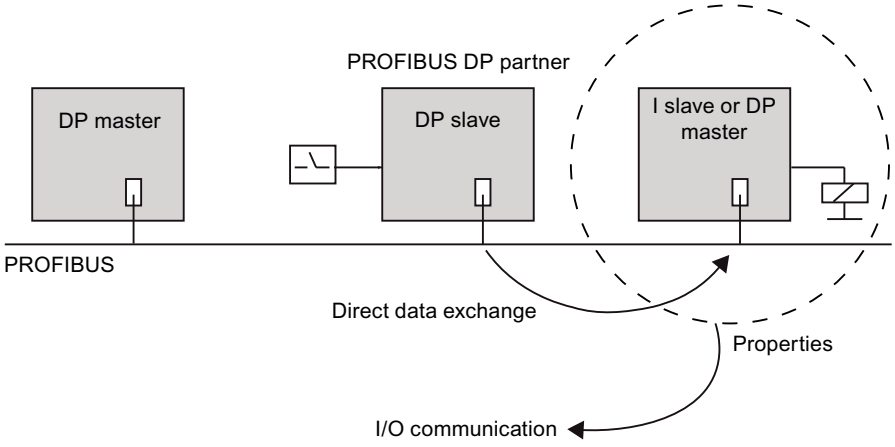

Configuration options

Possible applications of direct data exchange include:

- A configuration involving intelligent DP slaves: DP slave > I slave
- A configuration involving two DP master systems: DP slave > DP master

Number of PROFIBUS DP partners that can be connected

The total number of PROFIBUS DP partners that can be connected directly to a DP interface or that can be addressed via this interface using direct data exchange depends on the specific interface concerned. A maximum of 32 PROFIBUS DP partners can be addressed on the MPI/DP interface.

See also

Configurations involving several DP master systems (Page 2732)

Configurations involving intelligent DP slaves (Page 2733)

Configuring direct data exchange

Requirement

- You must be in the network view.
- A DP master system with input modules is available on the DP slave.
- A DP master or DP slave is linked with the DP master system as a "receiver".

Setting up direct data exchange

To set up direct data exchange, proceed as follows:

1. Click on the required communications partner (sender or receiver) to select it.
2. Press and hold down the left mouse button and then drag the second communications partner into the "Place device here" field, or select the "Partner 2" column of the I/O communications table.

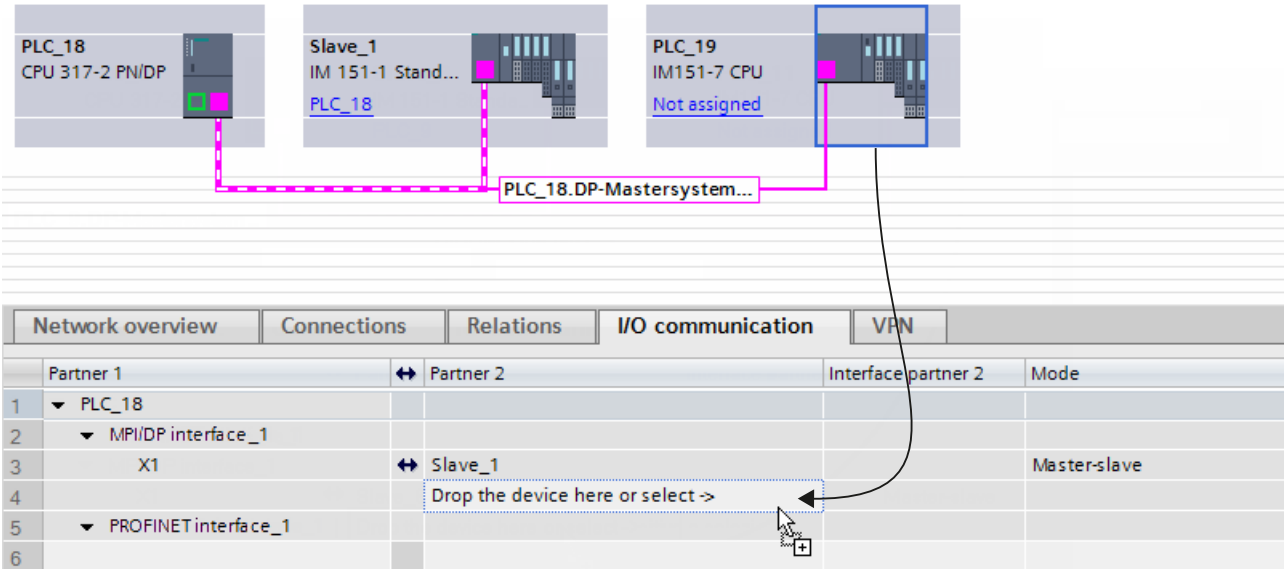

Alternatively, click directly on the "Place device here..." and select the connection partner you want from the drop-down list box.

Using drag-and-drop, you have now managed to set up a relationship in the table between two nodes for the purpose of communication. In a new line of the I/O communications table, "Direct data exchange" will now be displayed.

Note

The sender is always selected first in the case of a direct data exchange between two I-slaves. Subsequently, the receiver is dragged to the "Place device here or select" field or select from the drop-down list box. The communication direction of the direct data exchange determines the order in this case.

If you use the drop-down list in the "Partner 2" column of the I/O communication table to create the direct data exchange, you can select the communication direction of the direct data exchange directly by selecting it in the drop-down list.

Configuring direct data exchange

To configure direct data exchange, proceed as follows:

1. Select a direct data exchange partner.
2. In the I/O communications table, select the line showing "Direct data exchange" mode.
3. Select "Properties > Direct data exchange" in the Inspector window.
4. Create a new transfer area under "Direct data exchange" in the transfer area table:
   In the "Transfer area" column, click on "<Add new>" and select the "DX" communication type for direct data exchange in the "Type" drop-down list box. You can also create the new transfer area by setting the communication type in a new column only under "Type". The transfer area is then created automatically.
5. Under "Direct data exchange", click on the newly created transfer area.
   The detail view of the transfer area is opened.
6. Edit the properties for the sender and receiver and the general parameters of the direct data exchange.

You have now configured direct data exchange.

Deleting direct data exchange

To delete the direct data exchange, proceed as follows:

1. In the I/O communications table, select the line showing "Direct data exchange" mode.
2. Press "Del" or select the "Delete" command from the shortcut menu.

Example of how to configure direct data exchange

Introduction

In this example, you are going to configure three CPUs in a master-slave configuration for direct data exchange. You are then going to assign meaningful names and configure the address areas.

Use the CPUs below, which should be configured as follows:

CPU	Task	Address	Data
317-2 PN/DP	DP master	I 200	8 words from sender
315-2 DP	Sender	Q 100	8 words to master
317-2 DP	Receiver	I 120	First 2 bytes of the 8 words from the sender

Requirement

- You must be in the network view.
- The hardware catalog is open.

Setting up a PROFIBUS DP master system

To set up a PROFIBUS DP master system, proceed as follows:

1. Drag and drop the following CPUs from the hardware catalog into the network view: 315-2 DP, 317-2 DP and 317-2 PN/DP.
2. Give the CPUs meaningful names:
   – Select the CPU 317-2 PN/DP.
   – Assign the name "DP master" to the CPU under "Properties > General" in the Inspector window.

   Now repeat the naming process for the other CPUs, calling them "Sender" and "Receiver" respectively.
3. Select the CPU 317-2 PN/DP and then, if necessary, change the interface type from "MPI" to "PROFIBUS" under "Properties > MPI/DP interface > MPI address" in the Inspector window.
   Note: If the MPI/DP interface is set to "PROFIBUS" type, the entry is no longer "MPI address" but rather "PROFIBUS address".
4. Select CPUs 315-2 DP and 317-2 DP one after the other and in both cases select "Slave" mode under "Properties > DP interface > Mode" in the Inspector window.
   The two CPUs 315-2 DP "Sender" and 317-2 "Receiver" are now being operated as I-slaves.
5. While holding down the mouse button, draw the PROFIBUS connections between the master CPU and the two slave CPUs.

You have now successfully set up a PROFIBUS master system with the CPU 317-2 PN/DP as DP master and the 315-2 DP "Sender" and 317-2 DP "Receiver" as I-slaves. New lines for the communications partners will have been created in the I/O communications table relating to the CPUs. "I-slave" will be displayed in the "Mode" column.

Setting up direct data exchange

To set up direct data exchange within the PROFIBUS master system, proceed as follows:

1. Select the "Sender".
2. Press and hold down the left mouse button and then drag the "Receiver" to the "Place device here" field, which appears in the "Partner 2" column of the I/O communications table. Note: You can also open the drop-down list box in the "Place device here" field of the "Partner 2" column, and select the "Receiver" from this list.

You have now successfully set up a direct data exchange between the two I-slaves. A new line will have been created in the I/O communications table relating to the two I-slaves. "Direct data exchange" will be displayed in the "Mode" column.

Note

The arrow symbols in the I/O communication table between the "Peer 1" and "Peer 2" columns indicate the directions of the communication relations. The arrows point in both directions for master-slave and slave-I-slave relations. The arrows point from the sender to the receiver in the case of communication relations between I-slaves or during direct data exchange.

Configuring master-slave communication

To select master-slave communication, proceed as follows:

1. Select one of the two communication partners of master-slave communication (DP master CPU 317-2 PN/DP or I-slave CPU 315-2 DP "sender")
2. In the I/O communications table, click on the line with the respective second partner and "I-slave" mode.
3. Select "Properties > I-slave communication" in the Inspector window.

Or:

1. Select the CPU 315-2 DP "sender" I-slave.
2. Select "Properties > DP interface > Mode > I-slave communication" in the Inspector window.

To configure master-slave communication, proceed as follows:

1. Create a new transfer area under "I-slave communication" in the transfer area table:
   In the "Transfer area" column, click on "<Add new>" and select the "MS" communication type for master-slave communication in the "Type" drop-down list box. You can also create the new transfer area by setting the communication type in a new column only under "Type". The transfer area is then created automatically.
2. Under "I-slave communication", click on the newly created transfer area.
   The detail view of the transfer area is opened.
3. Make the following entries:
   for "DP master":
   - Address type I
   - Start address 200
   - Length 8
   - Unit WORD
   - Consistency total length

   for "Sender":
   - Address type Q
   - Start address 100

Once the data length has been specified, the slave and master addresses for the required data space are completed. If you click on "I-slave communication" in the Inspector window, you will see an overview of the updated configuration of master-slave communication.

Configuring direct data exchange

To configure direct data exchange, proceed as follows:

1. Select one of the two communication partners of direct data exchange (I-slave CPU 315-2 DP "Sender" or I-slave CPU 317-2 DP "Receiver")
2. In the I/O communications table, click on the line showing "Direct data exchange" mode.
3. Select "Properties > Direct data exchange" in the Inspector window.

To configure direct data exchange, proceed as follows:

1. Create a new transfer area under "Direct data exchange" in the transfer area table:
   In the "Transfer area" column, click on "<Add new>" and select the "DX" communication type for direct data exchange in the "Type" drop-down list box. You can also create the new transfer area by setting the communication type in a new column only under "Type". The transfer area is then created automatically.
2. Under "Direct data exchange", click on the newly created transfer area.
   The detail view of the transfer area is opened.
3. Make the following entries:
   for "Sender":
   - Address 100
   - Data length 1
   - Unit WORD

   for "Receiver":
   - Address 120

Note

If the sender in the direct data exchange is a normal DP slave, no transfer areas will be configured. Instead, an input module has to be added to the sender.

Particularity

In theory, you can also set an start address higher than 200 (202, for example). The length of consistent data will be adapted automatically. Furthermore, you can set a shorter length than that specified by the sender (1 byte, for example).

Note

If a consistent length of 3 bytes or more than 4 bytes is set for the sender and the data is to be transferred with the extended instruction of system function "DPWR_DAT" (SFC 15), the receiver will always need to use extended instruction "DPRD_DAT" (SFC 14) even if only 1 byte is actually being read, for example!

If you are using a loading operation (L IB..) instead of the DPRD_DAT to read 1 byte, a "0" is read in (incorrect value).

DPWR_DAT call on sender (CPU 315-2 DP)

STL

```
CALL  "DPWR_DAT"
       LADDR  :=W#16#64                //Start address Q 100
       RECORD :=P#M 10.0 BYTE 16       //User data source area
       RET_VAL:=MW100                  //Return value
```

DPRD_DAT call on receiver (CPU 317-2 DP)

```
STL
CALL  "DPRD_DAT"
       LADDR  :=W#16#78                //Start address I 120
       RET_VAL:=MW100                  //Return value
       RECORD :=P#M 10.0 BYTE 2        //User data target area
```

1.4.9.10 Using GSD files

GSD revisions

What you need to know about GSD revisions

The properties of DP slaves are made available to configuration tools by means of GSD files. Functional enhancements in the area of the distributed I/O will have an effect on the GSD specification, for example, they will require the definition of new keywords. This results in the versioning of the specification. In the case of GSD files, the version of the specification on which a GSD file is based is called a "GSD revision". From GSD revision 1, the GSD revision must be included as a keyword "GSD_revision" in GSD files. GSD files without this keyword will therefore be interpreted by configuration tools as GSD revision "0".

GSD files can be interpreted up to GSD revision 5. This means that DP slaves that support the following functions, for example, will be supported:

- Diagnostic alarms for interrupt blocks
- Isochronous mode and constant bus cycle time
- SYNC/FREEZE
- Clock synchronization for DP slaves

Installing the GSD file

Introduction

A GSD file (generic station description file) contains all the DP slave properties. If you want to configure a DP slave that does not appear in the hardware catalog, you must install the GSD file provided by the manufacturer. DP slaves installed via GSD files are displayed in the hardware catalog and can then be selected and configured.

Requirement

- The hardware and network editor is closed.
- You have access to the required GSD files in a directory on the hard disk.

Procedure

To install a GSD file, follow these steps:

1. In the "Options" menu, select the "Manage general station description files (GSD)" command.
2. In the "Installed GSDs" tab, select the directory in which the GSD files are stored.
3. Choose one or more files from the list of displayed GSD files.
4. Click on the "Install" button. The selected GSD files are now being installed.
5. To create a log file for the installation, click on the "Save log file" button.
   Any problems during the installation can be tracked down using the log file.
6. Click "Close". You will be notified that the DP slaves from the installed GSD files are entered into the hardware catalog. This process can take a few seconds.

By installing a GSD file you have added the DP slave described in the file to the hardware catalog.

Result

You will find the new DP slaves installed by means of GSD files in the hardware catalog under "Additional field devices > PROFIBUS DP".

GSD files are always saved together with the project, which means all information relevant for display of the device (including symbols) is also available in the saved project.

Deleting GSD file

Introduction

You can delete installed DP slaves using GSD files. These DP slaves are then no longer displayed in the hardware catalog.

Requirement

- The hardware and network editor is closed.
- The hardware catalog contains DP slaves installed using GSD files.

Procedure

To delete a GSD file, follow these steps:

1. In the "Options" menu, select the "Manage general station description files (GSD)" command.
2. In the "Installed GSDs" tab, select the directory in which the GSD file is stored.
3. Select the file that is to be deleted from the list of displayed GSD files.
4. Click the "Delete" button.

The selected GSD file was deleted and the DP slave is no longer located in the hardware catalog.

Finding unused GSD files

Introduction

When you add a GSD-based hardware component to your hardware configuration, different GSD files are added to the project data. If the GSD-based hardware component is once again deleted from the configuration, the GSD files remain in the data inventory of the project. The GSD files are no longer needed in the project if the corresponding GSD-based hardware component is not used in the hardware configuration again. You can find files that are no longer needed in the project with a search function and delete them.

Requirement

- A project must be open.
- The project must not be online.
- The project must not be write-protected.

Procedure

To find and delete unused GSD files, follow these steps:

1. In the "Options" menu, select the "Manage general station description files (GSD)" command.
2. Click on the "Find unused GSDs" button in the "GSDs in the project" tab.

   Note

   Unused GSD files are only available if at least one GSD-based hardware component was added to the configuration and then removed from the configuration. Unused GSD files in the project libraries are included in the search. GSD-based hardware components that are still being used in the configuration are not considered to be unused.

   Installed GSD-based hardware components that are included in the hardware catalog but have not been used in the hardware configuration of the project yet, could not have stored unused GSD data in the project.

3. If GSD files that are no longer needed were found in the project data, you can now delete them. To do so, click the "Delete" button.

GSD files that are no longer needed are deleted from the project data. The GSD-based hardware component is still available in the hardware catalog and, if necessary, can be added

to the configuration once again. The necessary GSD data are once again copied to the project data in this case.

Note

If you delete a GSD-based hardware component with "Options > Manage general station description files (GSD) > Installed GSDs", it is deleted from the hardware catalog. However, unused GSD files of this hardware component can still be a part of the project data and can be deleted with the function for finding the unused GSD files.

NOTICE

Actions that cannot be undone

When you delete GSD files that are not being used as an action, the action stack for actions that can be undone is deleted. This means the action of deleting unused GSD files as well as all previous actions can no longer be undone.

Configuring GSD-based DP slave

DP slaves that you have inserted through installation of a GSD file can be selected as usual via the hardware catalog and inserted in the network view. If you want to insert the modules of the GSD-based DP slaves, you must take into account some particular details.

Requirement

- You have installed a DP slave using a GSD file.
- You have inserted the head module in the network view in the usual manner.
- The device overview opens in the device view.
- The hardware catalog is open.

Procedure

To add the modules of a GSD-based DP slave, proceed as follows:

1. In the hardware catalog, navigate to the modules of the GSD-based DP slave.
   GSD-based DP slaves, also referred to as DP standard slaves, can be found in the "Other field devices" folder of the hardware catalog.
2. Select the desired module.
3. Use drag-and-drop to move the module to a free space in the device overview.
4. Select the module in the device overview to edit parameters.

You have now inserted the module in a free slot of the GSD-based DP slave and can edit its parameters.

Note

You can see only the GSD-based DP slave in the graphic area of the device view. The added modules of GSD-based DP slaves are only found in the device overview.

Preset configuration

For modules with an adjustable preset configuration, you can change this configuration in the inspector window under "Properties > Preset configuration".

1.4.9.11 Using PROFIBUS DPV1

Introduction to PROFIBUS DPV1

General

The DPV1 functionality is an extension of the PROFIBUS standard for distributed I/O. Relative to the functionality in the DPV0 predecessor, the DPV1 functionality has several extensions and simplifications:

- PROFIBUS DPV0 handles the cyclic exchange of data and the diagnostics.
- PROFIBUS DPV1 handles acyclic and cyclic data exchange plus interrupt handling.

Additional functionality for DPV1 devices (DP masters/DP slaves)

DP masters and DP slaves that support DPV1 have the following additional functions compared to devices that only support DPV0:

- The acyclic data exchange between master and slave is supported (read/write data record, e.g. to reassign parameters to a slave in operation). The data records of a module and the structure of these data records is provided in the documentation of the respective module.
- Interrupts can be set by a DPV1 slave that ensure handling of the interrupt triggering event in the master CPU. Even in "STOP" mode the interrupt data will be analyzed in the CPU (update of the diagnostic buffer and module status). However, OB processing does not take place in STOP.
  In addition to the interrupts that are known from SIMATIC, such as diagnostic interrupt for ET 200M, now the interrupt, status interrupt, update interrupt and vendor-specific interrupt are also supported.

See also

S7 integration of DPV1 slaves (http://support.automation.siemens.com/WW/view/en/10259221)

Configuring DPV1 devices

Some devices, such as the DP interface IM 151-1 HF, can toggle between DPV0 and DPV1.

Requirement

- You must be in network view.
- A DP master with DPV1 functionality must be available.
- A master-slave connection must be established with PROFIBUS.

Procedure

To switch the DP slave mode from DPV1 to DPV0 functionality, proceed as follows:

1. Select the DP slave.
2. Under "Properties > General > Module parameters" in the Inspector window, select "DPV0" or "DPV1" mode from the "DP interrupt mode" drop-down list.

or

1. Select the DP master.
2. In the I/O communications table, select the row with the connection between the DP master and the desired DP slave.
3. Under "Properties > General > Module parameters" in the Inspector window, select "DPV0" or "DPV1" mode from the "DP interrupt mode" drop-down list.

Using devices with different PROFIBUS functionality

The following rules apply to the mixed use of devices with DPV0 and DPV1 functionality:

- In principle, DPV0 slaves can also be operated on a DP interface with DPV1 functionality.
- In principle, DPV1 slaves can also be operated on a DP interface with DPV0 functionality. In this case, DPV1 functionality will be switched off. However, vendor-specific configuration rules for certain DP slaves can force DPV1 operation, so that these DP slaves cannot operate with the DP master system.

Assigning the diagnostics address

In the case of DPV1 slaves, the diagnostics address is automatically assigned to the virtual slot 0 as station proxy.

The following assignment basically applies:

- Diagnostics and interrupts that only refer to the entire DP slave are assigned to the virtual slot 0 and its diagnostics address: For example interrupts for modules in non-configured slots, station failure/station recovery (OB86 (Page 856)).
- Diagnostics and interrupts that originate from this module (for example, from a DP connection IM 153-2 in slot 2) are assigned to the remaining slots with their respective start addresses.

See also

Interrupts on the I-slave (Page 2759)

Programming DPV1 devices

For DPV1 functionality, there are OBs and SFBs/SFCs that cannot be used for DPV0.

Interrupt OBs for DPV1 events

DPV1 slaves can trigger interrupts. For diagnostics interrupts, hardware interrupts and remove/insert interrupts, you can use the appropriate OBs made available by the operating system of the S7-CPUs.

The interrupt OBs for DPV1 are presented in the table below. Detailed information is provided in the description of the OBs.

DPV1 interrupt	OB	Explanation
Status interrupt	OB55 (Page 829)	The status interrupt can be triggered if the operating state of a device or a module changes, for example, from RUN to STOP. A precise description of the events for which a status interrupt is triggered is provided in the documentation provided by the manufacturer of the respective DPV1 slave.
Update interrupt	OB56 (Page 831)	An update interrupt can be triggered when new parameters are assigned to a slot. This may have been caused, for example, by local access or partner access to the parameters. A precise description of the events for which an update interrupt is triggered is provided in the documentation provided by the manufacturer of the respective DPV1 slave.
Vendor-specific interrupt	OB57 (Page 834)	The event that triggers the vendor-specific interrupt can be specified by the manufacturer of a DPV1 slave.

Instructions for access to DPV1 slaves

The DPV1 interfaces with their functions are presented in the table below. Detailed information is provided in the description of the extended instructions.

Function	Interface (DPV1)	Comments
Read data record	RDREC	-
Write data record	WRREC	-
Receive interrupt from a DP slave	RALRM	The instruction must be called in the OB that triggers the interrupt.

Note

If a DPV1 slave is used and the DP interface of the DP master is set to "DPV0", no data records may be read from or written to the I/O modules using "WR_REC"/"RD_REC" or "WRREC"/"RDREC" instructions in the user program. In this case the DP master addresses the wrong slot (configured slot+3).

Solution: Change the interface of the DP master to "DPV1".

Check list for changing from DPV0 to DPV1

The following sections of the existing user program must be observed when you have edited the configuration and changed the interface from "DPV0" to "DPV1".

Function	What do I need to observe?
Address conversions	If you have used address conversions in the user program, for example, with extended instructions of system functions "GADR_LGC" (SFC 5), "LGC_GADR" (SFC 49), "RD_LGADR" (SFC 50), you must check the slot <-> logical start address assignment for the DPV1 slaves. In addition, slot 0 has an address. DPV1 slave: Whereas previously the first I/O module of the DP slave was assigned to slot 4 by the extended instructions, now the first I/O module is assigned to slot 1 (as shown in the hardware configuration). DPV1 slave integrated in the hardware catalog (for example, ET 200M): Interface module (slot 2) has its own address.
Reading diagnostics information using the system function "DPNRM_DG"	The originally assigned diagnostics address will also continue to function. Internally this address is assigned to slot 0. The diagnostic data record of the DPV1 slave is structured differently than it is with DPV0 (see description of the DP slave for ET 200M, for example, also under the keyword "Extended diagnostics").

Function	What do I need to observe?
Reading/writing data records	If you transfer data records to a DPV1 slave using extended instruction "WR_REC" (SFC 58) or read data records from a DPV1 slave using extended instruction "RD_REC" (SFC 59) and this slave is working in DPV1 mode, the DP master analyzes the error information received from the slave as follows: If the error information is in the areas W#16#8000 to W#16#80FF or W#16#F000 to W#16#FFFF, the DP master will forward the error information to the SFC. If it is outside of these areas it gives the value W#16#80A2 to the SFC and suspends the slave. For the description of the error information originating from the DPV1 slave, see the description of the "RALRM" (SFB 54) instruction. Additional information is available in the descriptions on jumps in language descriptions, module help and system attributes.
Reading out system status lists	If you use the extended instruction "RDSYSST", for example to read module status information or rack/station status information, you must also take the changed meaning of the slots and the additional slot 0 (see above) into consideration.

Example for analyzing interrupt information

Introduction

A distributed S7 digital input module (start address 288) triggers a hardware interrupt. In OB40 (Page 826), the additional interrupt information of this module can be read using the advanced instruction "RALRM". The system checks whether the first channel has triggered a hardware interrupt.

In the case of S7 modules, additional interrupt information could also be directly read from the start information of the OB40. However, the DPV1 functionality allows up to 59 bytes of additional interrupt information, which is far too much for the start information of the OB40.

Analyzing interrupt information from OB40 with "RALRM"

STL

```
// ...
// ...
//Switch for interrupt triggering address (288)
    L    DW#16#120
    T    "MD10"
    CALL  "RALRM" , "DB54"
      MODE  :=1                  //Function mode: 1=set all output parameters (i.e. F_ID
has no effect)
      F_ID  :="MD10"             //Start address of the slot from which an interrupt is
permitted
      MLEN  :=8                  //Max. length of the additional interrupt info in bytes
(for example, for channel status of the module)
      NEW   :="Alarm_neu"        //Interrupt received? (Yes = 1)
      STATUS:="DP_RALRM_STATUS"  //Return value with function result/error message
      ID    :="Slotaddress_Interrupt" //Start address of the slot from which an interrupt
was received
      LEN   :="Length_Alarminfo"  //Length of the additional interrupt info (4-byte header
info + for example, 4 bytes for S7 I/O modules)
      TINFO :=P#M 100.0 BYTE 28   //Pointer for OB start info +management info: 28 bytes
from MB 100
     AINFO :=P#M 130.0 BYTE 8   //Pointer for target area of the header info + supplemental
interrupt info (max. 59 bytes)
    A    M    124.0              //Is input 1 (bit 0) the interrupt triggerer?
    JC    Alrm
    BEU
Alrm:    S    Q   0.0            // interrupt processing
// ...
```

Example for analyzing diagnostic data

Introduction

The target area for diagnostic data should suffice for standard diagnostics (6 bytes), for identifier-specific diagnostics (3 bytes for 12 slots) and for analyzing the device-specific diagnostics (module status only, which is an additional 7 bytes). Extended instruction "RALRM" in OB82 (Page 843) is used for this.

For more extensive analysis (channel-specific diagnostics), additional bytes would be reserved if the DP slave supports this function.

Analyzing diagnostic data in OB82 with "RALRM" (SFB 54)

STL

```
// ...
// ...
      L     120                          //Specify start address for device/module,
      T     "Slotadress_Diag"           //from which diagnostics must be fetched


      CALL  "RALRM" , "DB54"
       MODE  :="Alle_Params"             // 1 = All output parameters are set
       F_ID  :="Slotadresse_Diag"        //Start address of the slot from which the
diagnostics will be fetched
       MLEN  :=20                      //Max. length of the diagnostic data in bytes
       NEW   :="new"                     //irrelevant
       STATUS:="RET_VAL"                 //Function result, error message
       ID    :="Slotadresse_Alarm"      //Start address of the slot from which an interrupt
was received
      LEN   :="Laenge_Alarminfo"       //Length of additional interrupt info (4-byte header
info +16 bytes diagnostic data)
       TINFO :=P#M 100.0 BYTE 28   //Pointer for OB start info +management info: 28 bytes
from MB 100
       AINFO :=P#M 130.0 BYTE 20        //Pointer on target area in which the diagnostic
data will be stored

// ...
//Structure of the stored diagnostic data:
//   MB 130 to MB 133: Header info (length, identifier, slot)
//   MB 134 to MB 139: Standard diagnostics (6 bytes)
//   MB 140 to MB 142: Identifier-specific diagnostics (3 bytes)
//   MB 143 to MB 149: Module status (7 bytes)
// ...
      A     M    141.0                  //Slot 1 with error?
      JC      stp1
      BE

stp1: L     MB   147                    //Module status slot fetch 1 to 4
      AW    W#16#3                     //Filter out slot 1
      L     W#16#2                     //2-bit status "wrong module" inserted
      ==I
      S     Q      0.1                 //Response to wrong module

      L     MB   147                    //Module status slot fetch 1 to 4
      AW    W#16#3                     //Filter out slot 1
      L     W#16#1                     //2-bit status "invalid data"
      ==I
      S     Q      0.2                 //Response to invalid user data
// ...
```

Slot model of the DPV1 slaves for I slaves

Introduction

This section describes how to make the assignments of I/O addresses and diagnostics addresses to the slots visible for DPV1. In particular, it deals with addresses that do not transport user data and their configuration.

The slot model for DPV1

With DPV1, a DP slave is formed by slots in precisely the same manner as DPV0. The numbers of the slots are 0, 1, ...n. Slot 0 applies only for DPV1 and has prominent significance because it is the proxy of the entire DP slave.

For example, proxy means that interrupts that have been triggered from slot 0 originate from the entire DP slave and not from a specific slot within the DP slave. Diagnostics that come from this slot are assigned to the entire DP slave and not to a specific slot or individual module.

Addresses for DP interfaces

From the CPU perspective, there is a separate logical address for each of its interfaces. The address for the respective DP interface is displayed in various ways:

- When you select a device or a DP interface in the Inspector window under "Properties > PROFIBUS address (Xn)" or "Properties > MPI/DP interface > PROFIBUS address (Xn)".
- With an enabled "Display addresses" in the toolbar in the network view: You will then see the name of any connected subnets for each interface, followed by the assigned PROFIBUS address.

These addresses have nothing to do with the slot model of DP slaves; they serve merely as internal CPU identification, for example, in the event of a failure of the interface. This address is of minor significance for the user program.

Slots and addresses for user data

Every manufacturer of a DP slave can freely determine which data will come from which slot.

As opposed to DPV1 slaves with user data starting with slot 1, DPV0 slaves are subject to a restriction in that the first I/O module must always be located in slot 4. DP slaves that are installed via a GSD file, can have user data starting with slot 1. Because distributed I/O data is usually accessed via its addresses, in the same way as central I/O data, user data of the DPV0 slaves must be addressed from the start address of slot 4. This also applies to intelligent DP slaves.

In the case of intelligent DP slaves, you can allocate I/O memory areas of the I-slaves to I/O memory areas of the DP master by means of the "I-slave communication" transfer areas. During operation (cyclic data exchange), the data that you transfer into the I/O memory areas in the user program of the intelligent DP slave are transferred via the transfer area into the I/O memory area of the DP master. You will find the table for the assignment of the I/O memory areas to the transfer areas by selecting the I-slave connection in the I/O communications table under "Properties > I-slave communication" in the Inspector window. After creating a transfer area, you can carry out the rest of the configuration under "Details of the transfer area".

When configuring the addresses under "Details of the transfer area", slot numbers are also displayed. The slots are not generated by real modules, as is the case with ET 200M, for example, but rather by freely definable lengths of the respective I/O areas. In this case they are also referred to as virtual slots.

The following points are important for an understanding of the address assignments:

- In addition to its real slots, the intelligent DP slave has virtual slots that are located in the I/O memory area.
- Like the real slots, the virtual slots are accessed via logical addresses. In the case of a DP slave, such as ET 200M, the virtual slots are accessed via the start address of a module; in the case of I-slaves via the configured address of the "I-slave communication".
- The addresses of the virtual slots are different from the perspective of the DP master and that of the DP slave. The assignment can be configured. Thus, the same DP slave will usually be accessed by the DP master under a different address from the one with which it is addressed by the DP slave.

Example of an address assignment for user data

The following table shows the assignment of the virtual slots in the transfer area:

Sample address from the perspective of the DP slave	Significance for the DP slave	Slot	Significance for the DP master	Sample address from the perspective of the DP master
		0		
		1		
		2		
		3		
I 2	Via input byte 2, you can read ...	4	... what the master has written in output byte 4.	Q 4
Q 5	what was written to output byte 5 in the slave...	5	... can be read in the master as input byte 6	I 6
I 8	...	6	...	Q 8
		...		
		35		

Note

In the network view, you will find slot assignment for the selected I-slave connection in the I/O communications table under "I-slave communication > Transfer areas" in the Inspector window.

Addresses for system information

Addresses for system information are used to handle diagnostics information, for example, or information about operating state transitions.

Addresses of the DP slave

The system information of a DP slave is assigned to modifiable addresses. In this context, the following addresses are relevant for DPV1:

- Addresses for device diagnostics:
  - The DP master diagnoses a failure or recovery of the I-slave.
  - The I-slave diagnoses a failure or recovery of the DP master.
- Addresses for operating state transitions:
  - The DP master can determine a change in operating state of the DP slave.
  - The intelligent DP slave can determine a change in operating state of the DP master.

The addresses are automatically assigned from top to bottom so that a conflict with user data does not arise. The recommended addresses should be accepted even if they can be changed. If the user program is to run on different CPUs, check whether the addresses are also in the address area of the "smallest" CPU.

The following table shows the assignment of the addresses in I-slave communication:

Sample address from the perspective of the DP slave	Significance for the DP slave	Sample address from the perspective of the DP master	Significance for the DP master
8189	Station failure/station recovery of the DP master	16381	Station failure/station recovery of the DP slave
8188	Operating state transition of the DP master	16380	Operating state transition of the DP slave

Summary

With virtual slots in the transfer area, the I-slave configuration looks like this:

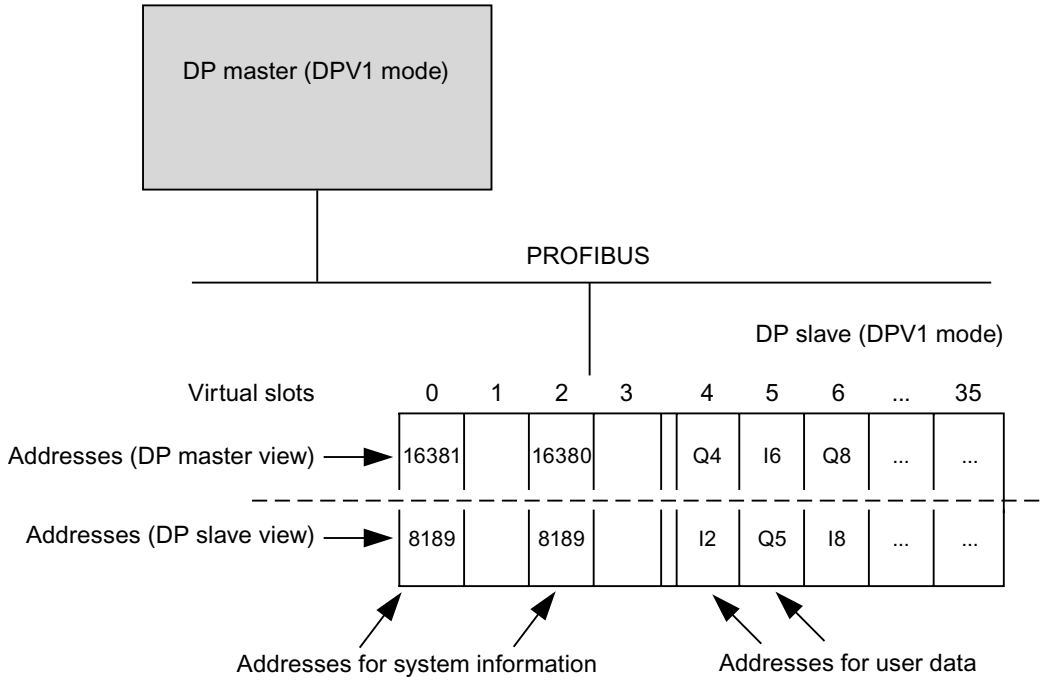

Triggering a hardware interrupt with the DP_PRAL instruction

Using the system instruction in the extended instruction "DP_PRAL", you can trigger a hardware interrupt for each configured address from the user program of the I-slave CPU. This applies both to user data addresses from the I/O area and to the address of virtual slot 2.

In the user program of the I-slave, for example, use the I/O addresses configured in the "Local..." column for instruction "DP_PRAL".

A hardware interrupt will then be triggered in the user program of the master. In the start information of the hardware interrupt OB (for example, OB40 (Page 826)), the address that you have configured in the "PROFIBUS DP partner" column will also be provided as the interrupt triggering address.

1.4.9.12 Constant and isochronous modes

Information about constant bus cycle time and isochronous modes

Same length bus cycles for PROFIBUS subnets

Introduction

The PROFIBUS properties "Isochronous mode" and "Constant bus cycle time" are the basis for synchronized processing cycles.

Constant bus cycle time ensures a precise same length time span for bus cycles. "Same length bus cycles" mean that the PROFIBUS-DP master always starts the DP bus cycle after the same time span. Thus the connected slaves also get their data from the master in time intervals that stay precisely the same. This is also referred to as a "Clocking of the bus cycle".

Constant bus cycle time is the prerequisite for isochronous mode.

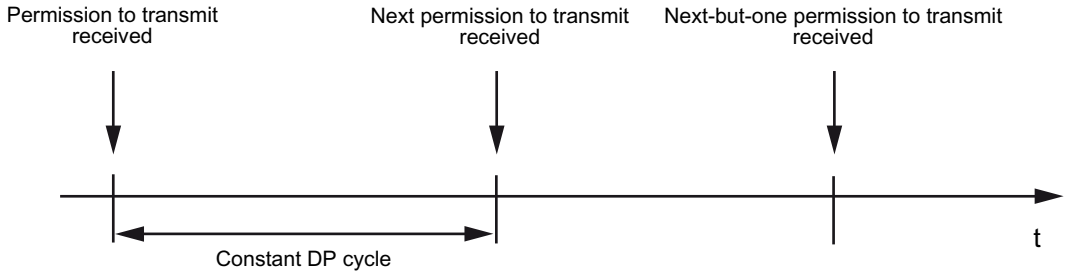

Composition of bus cycle time

The following figure shows how time is composed for a bus cycle.

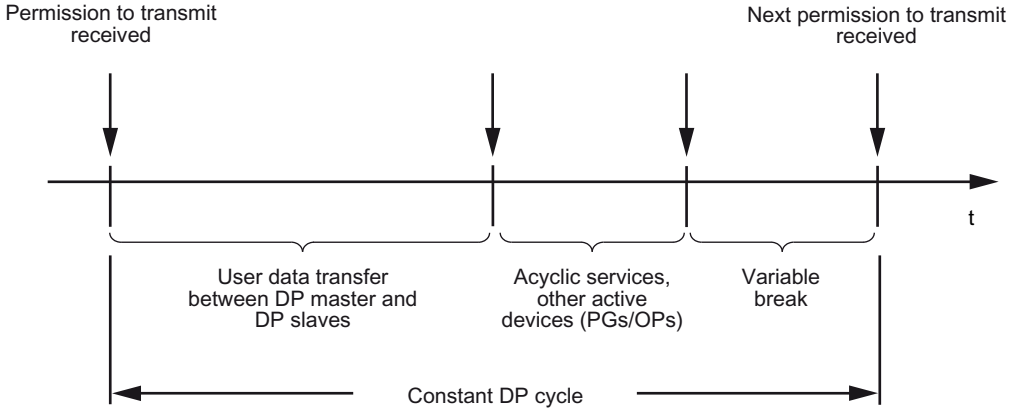

The "variable pause" shown in the figure is always minimal, if communication jobs are still pending, e.g. for additional active stations. The master, or to more precise the constant bus

cycle time master, controls the communication proportions in such a manner that the same duration for a bus cycle is always achieved.

Requirements

Constant bus cycle times have the following requirements:

- The DP master must support the "Constant bus cycle time" function.
- The constant bus cycle time master must be a class 1 DP master. A programming device/PC is not suitable as a constant bus cycle time master.
- The constant bus cycle time master must be the only active station on the PROFIBUS DP. Only one DP master system should be present on the PROFIBUS subnet. Programming devices or PCs may be connected in addition.
- No SYNC/FREEZE groups should be configured.
- Constant bus cycle time is only possible for the "DP" and "user-defined" bus profiles.
- Do not configure CiR (Configuration in Run).
- A H-CPU cannot be connected on the PROFIBUS subnet.

Time for constant bus cycle time DP cycle

The time for the constant bus cycle time for DP cycle is calculated automatically:

- From the PROFIBUS configuration (number of configured stations).
- From additional information for the calculation that is optionally entered: For example programming devices that must be taken into consideration, but that are outside of the project.

The calculated time for the constant bus cycle time for DP cycle takes the user data traffic of the PROFIBUS DP master into account. If errors occur then constant bus cycle time violations can occur.

You can change the calculated time within the displayed interval.

Influence of connected active stations (programming devices/PCs and I slaves)

A programming device/PC must only be taken into consideration if it is directly connected on the PROFIBUS with its PROFIBUS interface. If the programming device is connected via the MPI interface of the CPU, then it does not need to be taken into account as is shown in the figure below.

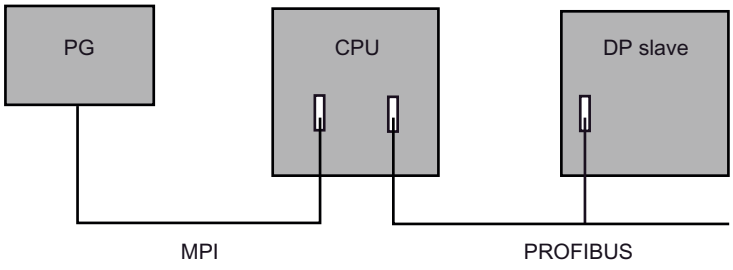

If intelligent DP slaves, e.g. CPU 315-2 DP, are connected then the time for constant bus cycle time for DP cycle must be more generously measured.

Synchronized execution cycles

PROFIBUS supports isochronous mode

Isochronous PROFIBUS is the essential basis for synchronized process cycles. The PROFIBUS system provides a reliable basic clock. The "isochronous mode" system property couples an automation solution with the constant bus cycle time PROFIBUS.

That is:

- Input data is read in synchronism with the DP cycle; all input data is read simultaneously.
- The user program which processes the I/O data is synchronized with the DP cycle by means of isochronous mode interrupt OBs OB 61 to OB 64.
- Data output is synchronized with the DP cycle; all output data take effect simultaneously.
- All input and output data is transferred consistently. That is, all data of the process image are associated logically and based on the same timing.

Synchronization of the single cycles

The following figure shows the bandwidth of possible response times for a processing cycle from a series of isochronous single cycles that occur due to synchronization of the input data read-in.

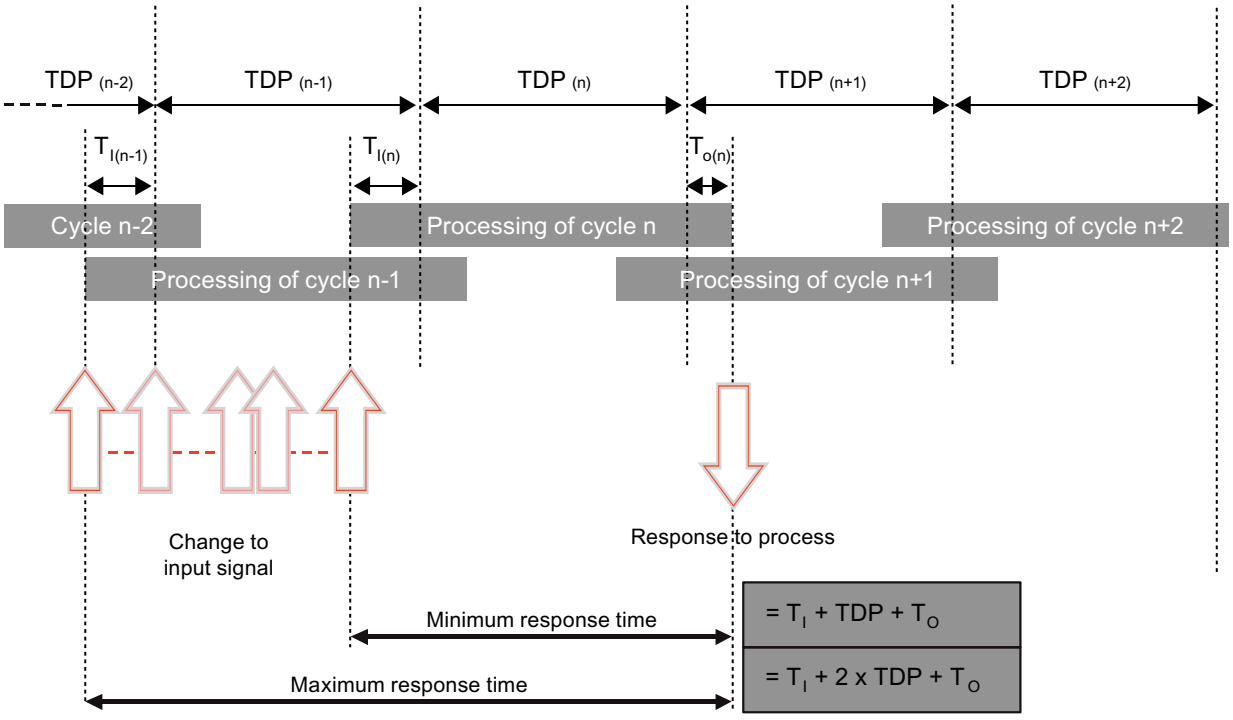

TDP	System cycle
Ti	Time of data input
To	Time of data output

Synchronization of single cycles makes it possible to read input data within cycle "n-1", to process and transfer the data within cycle "n", and to transfer the calculated output data to the "terminals" at the start of cycle "n+1". Resultant process reaction time: from "Ti + TDP + To" to "Ti + (2 x TDP) + To".

The "isochronous mode" system property enables constant cycle times within the SIMATIC system; the SIMATIC system is strictly deterministic on the bus system.

Just in time

The fast and reliable response time of a system operating in isochronous mode is based on the fact that all data is provided just-in-time. The constant bus cycle time (isochronous) DP cycle specifies the cycle for this.

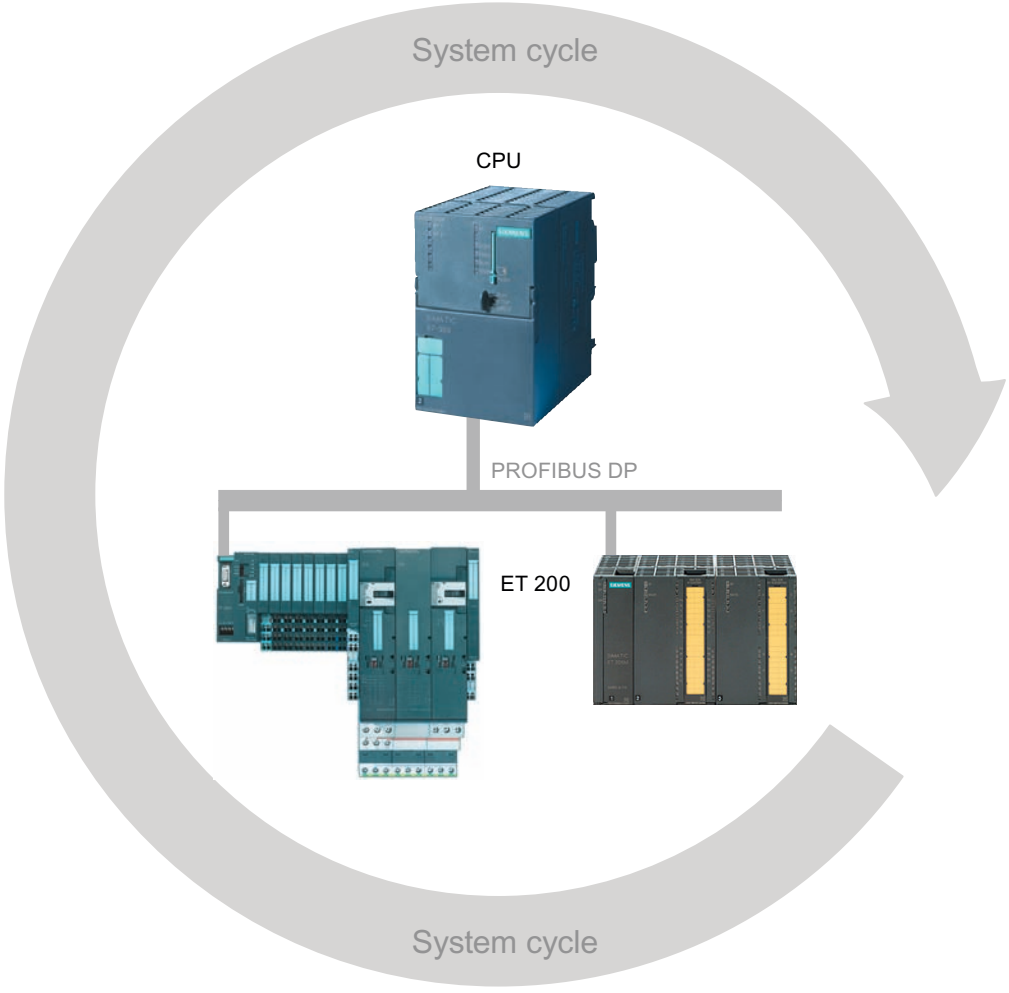

Bias time Ti

The start of the I/O read cycle is advanced by the amount of a bias time Ti in order to make all input data available for transfer on the DP subnet at the start of the next DP cycle. The bias time Ti is configured by the user or it is specified automatically.

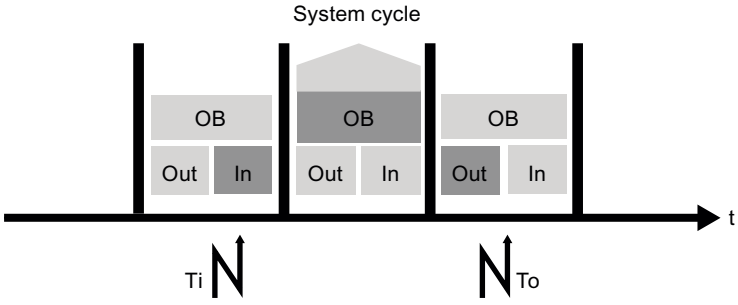

Processing the input data

PROFIBUS transfers the input data to the DP master via the DP subnet. The isochronous mode interrupt OB (OB 61, OB 62, OB 63 or OB 64) is called. The user program in the isochronous mode interrupt OB determines the process reaction and provides the output data in time for the start of the next DP cycle. You can have the length of the DP cycle calculated automatically or you can configure it yourself.

Making the output data available

The following applies for the output data:

- It will be made available promptly at the start of the next DP cycle.
- Is transmitted via the DP subnet to the DP slaves.
- It is isochronously, i.e. simultaneously forwarded to the process at the time To.

The result is a total reproducible reaction time of "Ti + (2 x TDP) + To" for the transfer from the input to the output terminal.

Characteristics of isochronous mode

Isochronous mode is based on three essential characteristics:

- The user program is synchronized with I/O processing, i.e. all operations are coordinated on a time basis. All input data is logged at a defined time. The output data also take effect at a defined time. The I/O data is synchronized with the system cycle up to the terminals. The data of one cycle are always processed within the next cycle, and take effect at the terminals within the subsequent cycle.
- I/O data is processed in isochronous mode, that is, input data is always read at constant intervals, and are always output at constant intervals.
- All I/O data is transferred consistently, that is, all data of a process image is associated logically and based on the same timing.

Example for isochronous capture of multiple measuring points

Introduction

With the "Isochronous mode" system property, measured values and process data can be captured in a fixed system clock. The signal processing until switching on the output terminal occurs in the same system clock. Isochronous mode thus contributes to high quality of the controller and thus to greater manufacturing precision. With isochronous mode the possible fluctuations of process response times are drastically reduced. Time-assured processing can be used for a higher machine clock/cycle.

Basically isochronous mode is an advantage wherever measured values must be captured synchronously, movements coordinated, and process responses must be defined and simultaneously executed as in the following example.

Task

QC requires precise measurement of dimensions within a camshaft production process.

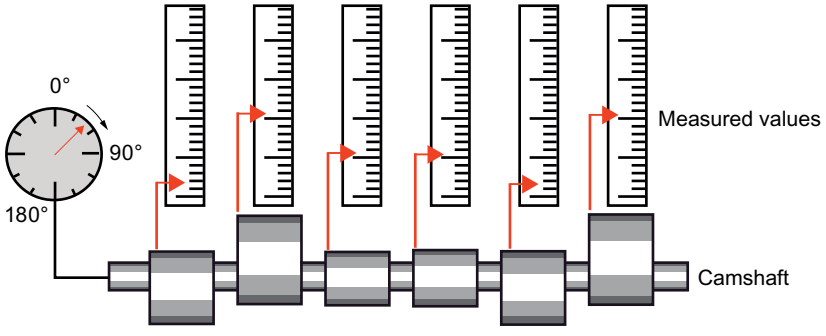

Conventional workflow

1. Machining of the cam shaft on the lathe
2. Stop
3. Measurement of the angular position
4. Measurement of all cam gradients
5. Continue machining, stop, measurement...

Isochronous mode workflow

Implementation of the "isochronous mode" system property and the resultant synchronization of measured value acquisition forms the basis of continuous and accelerated measuring processes. Resultant workflow:

1. Continuous machining of the camshaft
2. Synchronous measurement of the camshaft positions and gradients within a continuous rotation.
3. Machining of the next camshaft

All camshaft positions and the corresponding measured values (red in the figure) are measured synchronously within a single rotation of the camshaft. This increases machine output and maintains or enhances precision of the measurement.

Rules, requirements, and supplemental conditions

What you need to be aware of for configuring isochronous mode is

- Isochronous mode cannot be used on optical PROFIBUS networks.
- Constant bus cycle time and isochronous mode are only possible for the "DP" and "user-defined" bus profiles.
- Isochronous mode is only possible with the DP interfaces integrated in the CPU. Isochronous operation with CPs is not possible for PROFIBUS.
- Only the constant bus cycle time master is allowed as active station on isochronous PROFIBUS-DP. OPs and programming devices (or PCs with programming device functionality) influence the dynamic response of the constant bus cycle time for the DP cycle and thus are not recommended.
- Isochronous mode across subnets is not possible.
- The addresses of the isochronous I/O have to be located in the address area of the process image partitions since the isochronous I/O can only be processed in process image partitions. Without the use of process image partitions isochronous consistent data transfer is not possible. Compliance with configuration limits is monitored because the number of slaves and the number of bytes on the DP master system are limited per process image partition.
- In addition, non-isochronous I/O may be located along with isochronous I/O in the same process image partition. An isochronous module/submodule has to be located in the process image partition.
- A full isochronous mode from "terminal" to "terminal" is only possible if all components participating in the chain support the "isochronous mode" system property. When selecting in the catalog, make sure you check the "isochronous mode" or "processing in isochronous mode" entry in the information field of the module.
- When you configure isochronous mode, you cannot assign a SYNC/FREEZE group to the slave.

See also

Assigning a DP slave to a SYNC/FREEZE group (Page 2743)

Isochronous mode manual (http://support.automation.siemens.com/WW/view/en/15218045)

Configuring and assigning parameters to isochronous mode

Optimizing dynamic response by selecting a suitable topology

Influence of the DP topology

The dynamic response of isochronous mode can be enhanced by selecting a suitable DP topology.

Separation of isochronous and non-isochronous I/O

Isochronous and non-isochronous I/O can be combined in a DP master system.

It is advisable to distribute isochronous and non-isochronous I/O to different DP master systems for good performance.

If TDs, OPs or programming devices are used then you should not operate them on the isochronous DP master system.

Practical distribution of the I/O submodules

Reduce Ti and To times to minimum by setting the focus on a practical distribution of I/O submodules to the interface modules (ET 200M, ET 200S) when you design the DP topology.

Note

Vary the different configurations and calculate the temporal behavior. Select the most favorable configuration.

Additional information

Additional instructions for optimization of the constant bus cycle time are available in the operating instructions of the ET 200M and ET 200S DP slave families.

Assigning constant bus cycle time and isochronous mode parameters

Configuring constant bus cycle time and isochronous mode parameters

Configuration for isochronous mode

A configuration for isochronous mode consists of the following isochronous components:

- CPU with integrated DP interface (for example, CPU 319-3 PN/DP)
- DP interface modules (for example ET 200S interface IM 151-1 High Feature)
- Distributed input/output modules (for example DI 2xDC24V, High Feature [131-4BB00], DO 2xDC24V/2A, High Feature [132-4BB30])

A current list of isochronous components is available in the Service & Support Portal (http://support.automation.siemens.com/WW/view/en/14747677).

Procedure

To configure a constant bus cycle time and isochronous mode, follow these basic steps:

1. Configure a CPU capable of isochronous mode and set the properties for isochronous mode:
   - Isochronous mode interrupt
   - Assigned DP master system
   - Process image partition
   - Delay time
2. Configure isochronous DP slaves with isochronous modules and set the properties for isochronous mode centrally in the PROFIBUS properties or for the specific DP slaves (slave properties). Here, these are essentially the Ti/To values and how they are calculated.

   Note

   You can set all properties for the isochronous I/O centrally in the PROFIBUS properties. Here, you also have an overview of all slaves and their settings on PROFIBUS at the same time.

3. Write the user program with accesses to the isochronous I/O.

You will find a detailed description of the procedure in the next sections.

See also

Setting CPU properties (Page 2799)

Setting PROFIBUS properties (Page 2801)

Setting the isochronous mode of DP slaves (Page 2803)

Creating a user program (Page 2803)

Setting CPU properties

The CPU settings for isochronous mode interrupts link the user program into isochronous mode processing using the relevant organization blocks (for example, isochronous mode interrupt OBs).

Requirement

The CPU must support isochronous mode and the constraints must not exclude isochronous mode.

Procedure

To set the CPU properties, proceed as follows:

1. Select the CPU in the device view or the network view.
2. Select the "Isochronous mode interrupts" CPU properties.
3. Make the following settings for each isochronous mode interrupt OB:
   - DP master system used
   - Required process image partition or partitions
   - Delay time

Additional information about the process image partitions

The assigned process image partitions must be updated in the isochronous mode interrupt OB by calling extended instructions "SYNC_PI" (SFC 126) and "SYNC_PO" (SFC 127). The "SYNC_PI" call updates the process image input partition and the "SYNC_PO" call updates the process image output partition.

If the distributed I/O addresses have been assigned to different process image partitions, enter the numbers of the process image partitions separated by commas (for example, 1,2,3).

Additional information on the delay time

- The delay time is the time between the global control frame and the start of OB 6x. During this time, the DP master performs the cyclic data exchange with the DP slaves.

  Note

  Recommendation: There is a check box "Automatic minimum" next to the input box for the delay time. If you have selected this check box, you do not need to set the delay time again after configuration changes!

- The delay time can then only be set when the "Automatic minimum" check box is deselected.

  Note

  If the delay time is 0, OB 6x starts with the global control frame. In this case, access to isochronous I/O by means of the advanced instruction "SYNC PI" (SFC 26) will very probably fail because the I/O data is not yet available at the start time of the OB.

  If a too large delay time were selected, access to the isochronous I/O would have been possible earlier. A delay time that is too large leads to an unnecessarily long OB runtime.

See also

Configuring constant bus cycle time and isochronous mode parameters (Page 2798)

Rules, requirements, and supplemental conditions (Page 2797)

Setting PROFIBUS properties

Introduction

To edit the properties of isochronous mode on PROFIBUS, you will need to change the settings on the subnet/DP master system.

The constant bus cycle time for isochronous mode in a PROFIBUS subnet with a constant bus cycle time is activated automatically if the following conditions are met:

- Assignment of a DP master system to the isochronous mode interrupt OB in the CPU properties for isochronous mode interrupts.
- Isochronous mode setting for a DP slave on PROFIBUS.

You only need to edit the PROFIBUS properties if you want to change the times manually. Otherwise all times will be calculated automatically.

Requirement

PROFIBUS subnet with constant bus cycle time.

Procedure

To change the properties of isochronous mode in PROFIBUS, follow these steps:

1. In the network view, select the PROFIBUS subnet, and in the Inspector window select "Constant bus cycle time" under "Properties".

   Note

   Make sure that you do not select a highlighted master system. The PROFIBUS properties and the isochronous mode properties can only be changed if the PROFIBUS subnet is selected.

2. Set the properties for the isochronous mode. You will find explanations of the specific settings in the sections below.

Properties for the isochronous mode are not selectable

If the properties for isochronous mode cannot be selected, check the following:

- Attachment of only one DP master on the PROFIBUS subnet. Isochronous mode is not possible if multiple DP masters are attached.
- Support of isochronous mode by the DP master (for example a CPU with an integrated DP interface).
- Configurability of the SYNC/FREEZE groups. SYNC/FREEZE groups and isochronous mode are mutually exclusive.

Setting DP cycle time and Ti/To values manually

If you want to set the DP cycle time manually, first clear the "Automatic minimum DP cycle time" check box. Now you can enter a higher value than the automatically calculated minimum DP cycle time.

If you want to set Ti and To values manually, first clear the "Automatic Ti/To values" check box.

Isochronous mode and Ti/To settings in the detailed overview

The detailed overview allows you to set the following properties for each suitable slave and/or for each suitable module in the slave:

- Isochronous mode
- Ti/To values

The detailed overview is a table. You can use the shortcut menu of the title bar of the table to adapt the tabular display, e.g. to display the values for TDPMin.

It is not possible to manually select or clear the check box "Isochronous mode" for a CPU (PLC). The check box is selected automatically if an isochronous mode interrupt is assigned to the DP master system in the "Interrupts" section of the CPU properties.

Possible settings for calculating the Ti/To values in the detailed overview

You can make the following settings for calculating the Ti/To values:

- From subnet
  If you activate isochronous mode, then the option Ti/TO values "From subnet" is automatically the default. With this option, all slaves that obtain their Ti/To values from the network have identical Ti/To times (those of the PROFIBUS subnet).
- Automatic minimum
  If, for example, you change the DP cycle time manually or change Ti/To of other slaves values manually and this change affects the Ti/To times of the slave, then all necessary adjustments will be made automatically. With this option, the Ti/To values of the slaves can deviate from each other.
- Manual
  With this option you can enter Ti/To values for the slave manually. Errors can occur with this setting:
  - Times selected too short
  - Ti/To of other DP slaves can no longer be calculated automatically due to the settings you made

Display "-" for time Ti (To)

If no value is displayed for Ti or To for the PROFIBUS subnet (display "-") then this is due to the following cause: The common parts of all Ti/To value ranges of all DP slaves is used for automatic calculation of the Ti/To time for the PROFIBUS subnet. The smallest value from these common ranges is selected. If no common range can be formed then "-" is displayed. In this case, a DP slave cannot obtain a Ti/To value from the network!

See also

Configuring constant bus cycle time and isochronous mode parameters (Page 2798)

Ti/To overlap (Page 2804)

Setting the isochronous mode of DP slaves (Page 2803)

Assigning a DP slave to a SYNC/FREEZE group (Page 2743)

Setting the isochronous mode of DP slaves

Settings on the DP slave

Except for the constant bus cycle time for the DP cycle and the Ti/To times for the PROFIBUS subnet, you can make the same settings in the DP slave properties that you make in the PROFIBUS properties.

Procedure

To set the isochronous mode for modules, follow these steps:

1. In network view, select the DP slave whose properties you want to edit.
2. Under "Properties > Isochronous mode", adapt the parameters to suit your requirements.

Setting Ti/To times the same for all slaves

The default behavior after adding DP slaves in network view is: Ti/To values are obtained from the network and are therefore automatically the same.

You can either set the same Ti/To times manually in an edited configuration or set all slaves so that they obtain their Ti/To values from the network.

See also

Configuring constant bus cycle time and isochronous mode parameters (Page 2798)

Setting PROFIBUS properties (Page 2801)

Creating a user program

Procedure

To create the user program for the isochronous mode, proceed as follows:

1. Create the necessary isochronous mode interrupt OBs (for example, OB 61).
2. In the isochronous mode interrupt OB, call the following extended instructions to update the process image partition:
   - "SYNC_PI" (SFC 126) to update the process image input partition
   - "SYNC_PO" (SFC 127) to update the process image output partition

Use the process image partition that is set on the CPU (parameter area "Isochronous mode interrupts").

Note

The following situation can occur, particularly with very short DP cycle times: The execution time of the user program (OB 6x with SYNC_PI/SYNC_PO called) is greater than the shortest cycle (see technical specifications of the CPU, section "Isochronous mode"). In this case, you will need to increase the automatically calculated DP cycle time manually.

See also

Configuring constant bus cycle time and isochronous mode parameters (Page 2798)

Ti/To overlap

Ti/To overlap

Reduction of the system cycle and of the process reaction time

The TDP system cycle and process response time are reduced by overlapping Ti and To, that is, overlap reduces the accumulated time that includes generation and detection of an event, plus the processing time, plus the time required to return a response signal at the outputs.

Requirement of Ti /To overlap

Condition of overlapping Ti/To:

ET 200M: Interface module IM 153-2; article number 6ES7 153-2BAx2-0XB0 or higher

ET 200S: Interface module IM 151-1; article number 6ES7 151-2BAx2-0XB0 or higher

Note

Modules which handle input conversion times (TCI) and output conversion times (TCO) (depending on the operating mode of the I/O submodule) cannot be operated in isochronous mode with Ti/To overlap. Overlap of Ti and To is only possible if these modules **are not** activated for isochronous mode.

See also

Setting PROFIBUS properties (Page 2801)

Operating principle of Ti/To overlapping (Page 2805)

Setting isochronous mode with overlap of Ti and To (Page 2806)

Operating principle of Ti/To overlapping

Overlapping processes

The function principle of Ti/To overlap is based on the fact that the peripheral input module already reads its inputs while the peripheral output module is still busy transferring the process reactions of the user program to its outputs. This functionality reduces the TDP system cycle and process reaction times.

DP bus cycle without Ti/To overlap

The diagram below shows a constant DP bus cycle time in isochronous mode without Ti/To overlap.

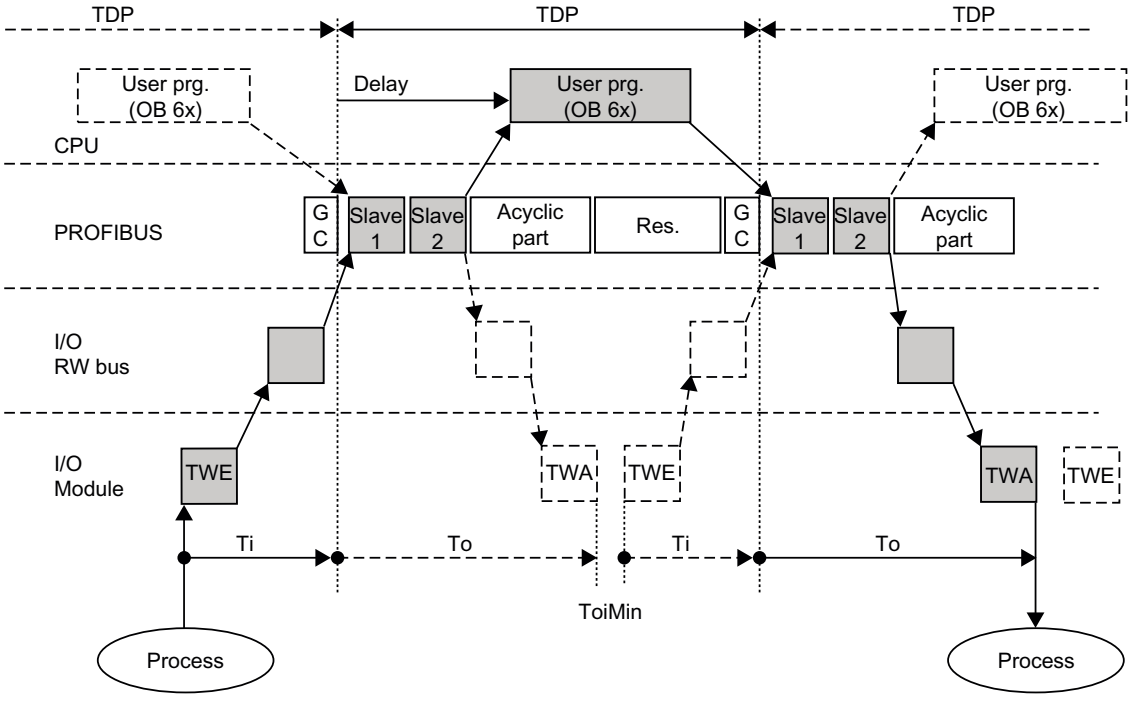

DP bus cycle with Ti/To overlap

The diagram below shows a constant bus cycle time in isochronous mode with Ti/To overlap.

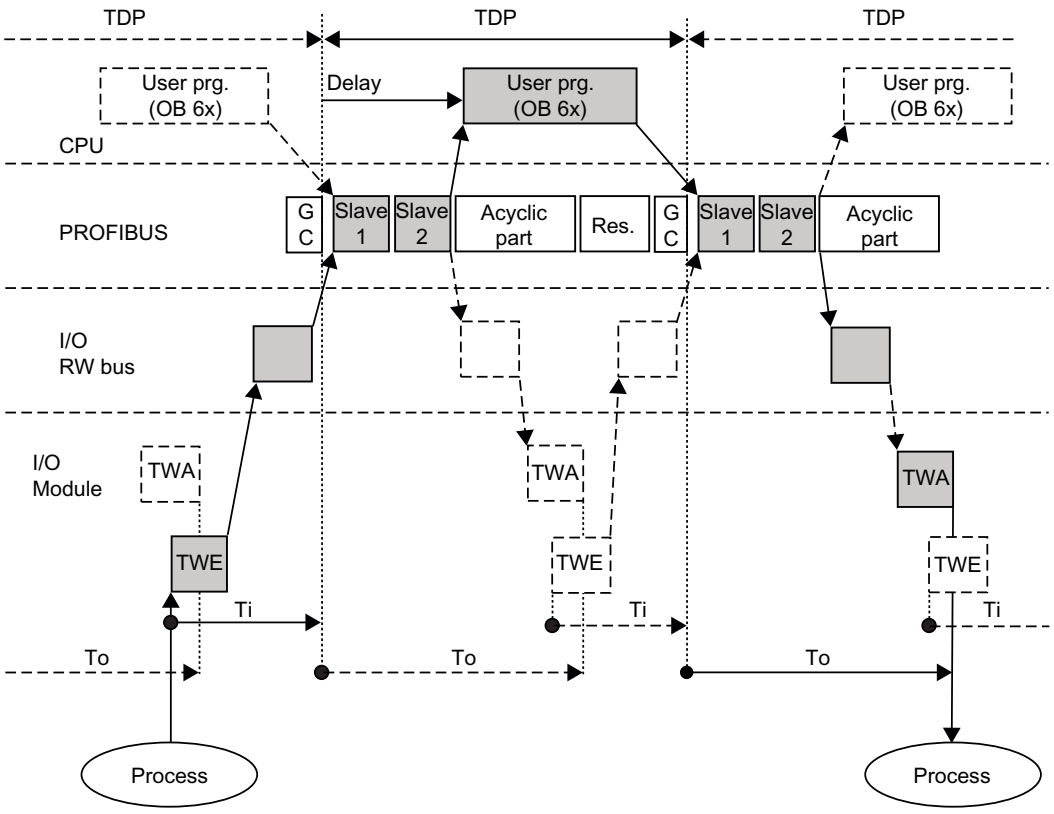

Rules for Ti, To and TDP

Rules for Ti, To and TDP:

- Ti + To ≤ TDP without overlap, that is: TDP ≥ Ti + To
- The conditions Ti + To > TDP or TDP < Ti + To may develop with overlap

See also

Ti/To overlap (Page 2804)

Setting isochronous mode with overlap of Ti and To

Configuring isochronous mode with Ti/To overlap

For isochronous mode with Ti/To overlap you must be aware of the following:

The DP slave for which Ti/To overlap is possible should not get its Ti/To values from the PROFIBUS subnet. For this DP slave select the option "Ti/To values automatic".

The best-possible values for Ti and To are set automatically, depending on the current set constant bus cycle time for DP cycle.

Rules for Ti/To overlap

Rules for operation in isochronous mode with Ti/To overlap:

- Ti ≤ TDP
- To ≤ TDP

Finding system reserves

Visualize the sequences using a calculation table (http://support.automation.siemens.com/WW/view/en/23876584) to determine system reserves and use the result to optimize these sequences in STEP 7.

Note

Disclaimer of liability

Siemens AG shall not be liable for material or immaterial damage caused by the use of the calculation table. Any liability of Siemens AG in particular for direct or consequential injury, damage to assets and financial losses sustained as a result of the use of the calculation table shall be ruled out.

Siemens does not provide support for the calculation table.

Note

Remember the rules governing the optimization of the constant bus cycle times and Ti/To overlap.

Additional information

Additional information for optimizing the constant bus cycle time are provided in the ET 200M operating instructions and in the ET 200S operating instructions.

See also

Ti/To overlap (Page 2804)

1.4.10 Configuring PROFIBUS DP

1.4.10.1 The basics of configuring a DP master system

Distributed I/O

DP master systems that consist of a DP master and DP slaves which are connected via a bus and communicate with one another via the PROFIBUS DP protocol are referred to as distributed I/O.

Firmware version of the S7-1200 CPU

Use of the PROFIBUS functions with the S7-1200, requires CPUs with firmware version 2.0 or higher.

Configuring distributed I/O

Since DP masters and DP slaves are different devices, these instructions only provide a basic configuration procedure. However, the process for configuring distributed I/O is practically identical to that of non-distributed configuration.

Creating the DP master system in the network view

After you have used dragged a DP master and a DP slave (for example, a CM 1243-5 and a CM 1243-5) from the hardware catalog to the network view, connect them both to a PROFIBUS subnet.

Additional information

Observe additional information on the scope of functions in the manuals of the respective device.

1.4.10.2 DP slaves within the hardware catalog

DP slaves within the hardware catalog

You will find the DP slaves in the "Distributed I/O" folder of the hardware catalog. Compact and modular DP slaves are located there:

- Compact DP slaves
  Modules with integrated digital/analog inputs and outputs, for example, ET 200L
- Modular DP slaves
  (Interface modules with S7 modules assigned, for example, ET 200M

The available DP master and the desired functionality will determine which DP slaves can be used.

I slaves within the hardware catalog

The CM 1242-5 is, for example, an DP slave that can be configured as intelligent DP slave. You can find it in the hardware catalog at:

- CM 1242-5
  "PLC > SIMATIC S7 1200 > Communication module > PROFIBUS"

1.4.10.3 DP/DP coupler in the hardware catalog

Introduction

A DP/DP coupler connects two PROFIBUS DP networks as a gateway so that the DP master from one network can transfer data to the DP master of the other network.

The maximum amount of data that can be transferred is 244 bytes input data and 244 bytes output data.

DP/DP coupler in the hardware catalog

Details of a DP/DP coupler as gateway between two DM master systems are contained in the hardware catalog in the folder "Other field devices > PROFIBUS-DP > Gateways".

Configuring the DP/DP coupler

DP/DP couplers are configured in both PROFIBUS networks, each in their own master systems.

The input and output areas of both networks must thereby be matched to one another. The output data from one end of the DP/DP coupler are accepted as input data at the other respective end and vice versa.

1.4.10.4 Configurations involving PROFIBUS DP

Configurations involving basic DP slaves

Communication between DP master and DP slave

In the case of a configuration involving simple DP slaves, data is exchanged between the DP master and simple DP slaves, i.e. with I/O modules via the DP master. The DP master sequentially polls each DP slave configured within the DP master system and contained in its polling list. In doing so, it transfers the output data to the slaves and receives their input data in return.

Mono-master system

The configuration with only one DP master is also described as mono-master system. A single DP master with its associated DP slaves is connected to a physical PROFIBUS DP subnet.

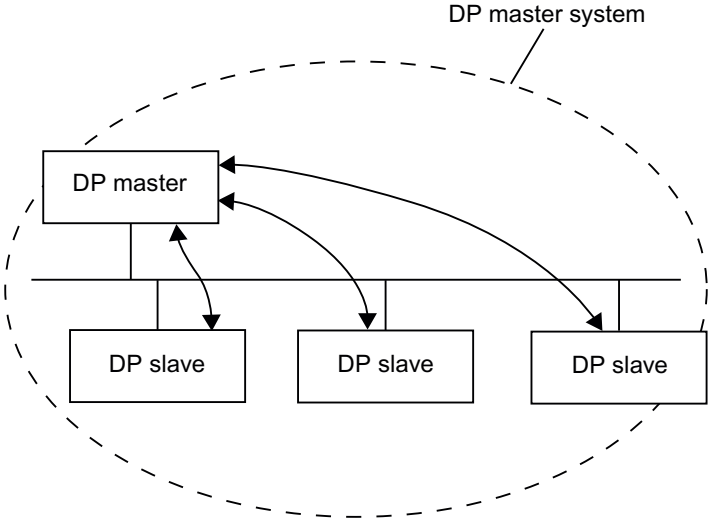

Configurations involving intelligent DP slaves

Definition

DP slaves that feature their own preprocessing program are referred to as intelligent DP slaves (I-slaves).

CM 1242-5 is an intelligent DP slave.

I-slave <> DP master data exchange

A higher-level automation system processes the automation task, which is broken down into sub-tasks. The sub-tasks are processed in the lower-level automation systems. The necessary control tasks are processed separately and efficiently in the CPUs as preprocessing programs.

In the case of configurations involving intelligent DP slaves, the DP master only accesses the operand area of the I-slave's CPU, and not the I/O modules of the intelligent DP slave. The operand area must not be assigned to any actual I/O modules in the I-slave. The assignment must be made during I-slave configuration.

The addresses of the data to be exchanged between the master and slave are configured in the transfer area of the I-slave.

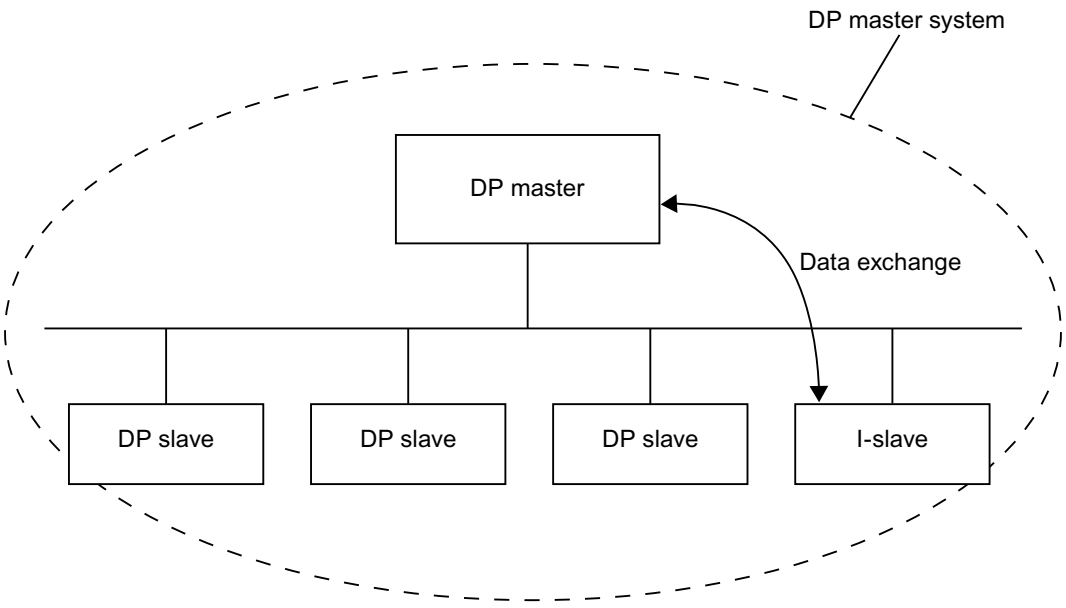

1.4.10.5 Configuring distributed I/O systems

Creating a DP master system

Introduction

To create a PROFIBUS DP master system, you need to have one DP master and at least one DP slave. As soon as you connect a DP master to a DP slave via their PROFIBUS DP interfaces, a master-slave link is established.

DP master and DP slave

The DP master must have a PROFIBUS DP interface. When using an S7-1200 CPU with PROFINET interface, you require the communications module CM 1243-5 with PROFIBUS DP interface for use as the DP master.

As DP slave, you can use head modules of the distributed I/O with PROFIBUS DP interface or CPUs as intelligent DP slaves. When using an S7-1200 CPU with PROFINET interface, you require the communications module CM 1242-5 with PROFIBUS DP interface to use the CPU as an intelligent DP slave.

Requirement

- You are in the network view.
- The hardware catalog is open.

Procedure

To create a DP master system, for example with a CPU 1217C , follow these steps:

1. Select a CPU 1217C from the hardware catalog as a potential DP master.
2. Drag the CPU onto the free area within the network view.
3. Select the communications module CM 1243-5 from the hardware catalog.
4. Drag the communications module to the CPU in the network view.
5. Right-click on the DP interface of the communications module.
6. Select "Create master system" from the shortcut menu.

A DP master system with a CPU 1217C is created as the only node via the CM 1243-5 as DP master.

If you connect the DP interface of a DP slave to the DP interface of the DP master, the DP slave will be added automatically to the DP master system. If there is not yet a subnet between the DP master and DP slave, a new subnet is created between the DP master and DP slave.

To include, for example, a CPU 1217C with communications module CM 1242-5 as an intelligent DP slave in the DP master system, follow the steps below:

1. Click on the DP interface of the DP master (interface of the CM 1243-5) or DP slave (interface of the CM 1242-5).
2. Hold down the mouse button and draw a connecting line between this DP interface and that of the desired communication partner.

or

1. Click on the hyperlink of the DP slave CPU 1217C with CM 1242-5.
2. Select the required DP master from the list of possible DP masters displayed.

The intelligent DP slave CPU 1217C with CM 1242-5 is included in the DP master system as DP master with CPU 1217C and CM 1243-5 .

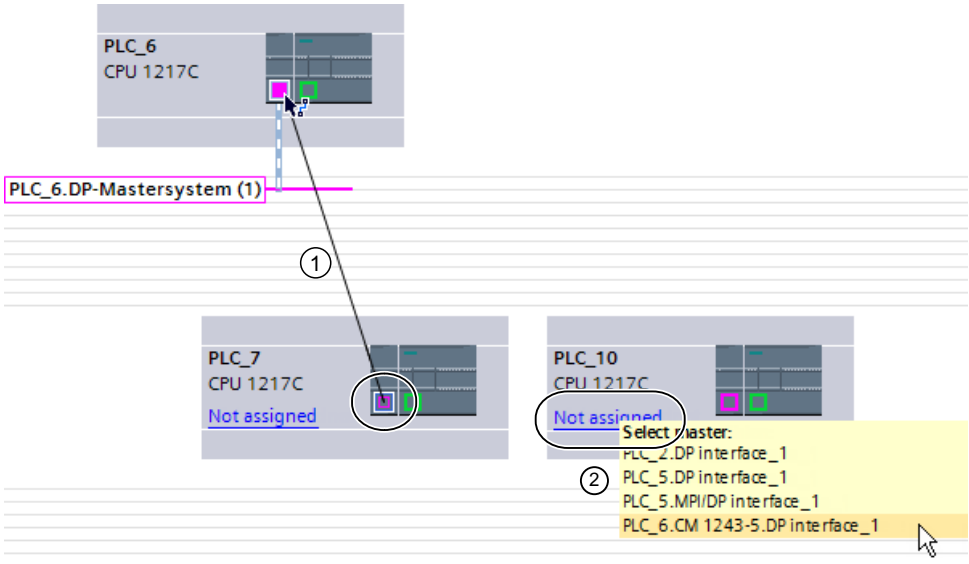

① Drag between the DP interfaces of DP slave to DP master.

② Click on the link of an unassigned DP slave, and a selection of possible DP masters opens.

If necessary, adapt the properties of the PROFIBUS subnet or of the DP master (e.g. PROFIBUS address) in the Inspector window under "Properties".

DP master display on the DP slave

When a DP slave is connected to a DP master, the name of the DP master is displayed on the DP slave as a hyperlink. Click the hyperlink to select the associated DP master.

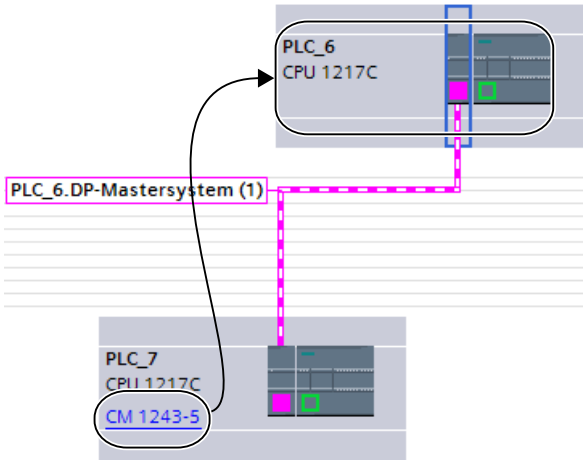

Highlighting applied to the DP master system

When you create a new DP master system, highlighting will be applied to it. This enables you to identify quickly which devices belong to the DP master system. You can also highlight a DP master system yourself by moving the mouse pointer over a subnet. This will result in the names of the available DP master systems being displayed. Click the required DP master system to highlight it.

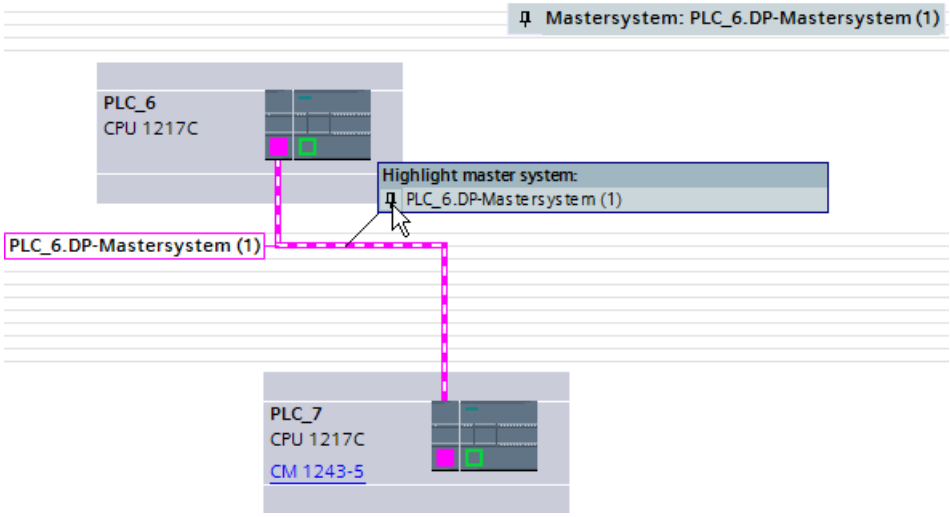

There are various ways of removing the highlighting from a DP master system:

- Highlight a different DP master system.
- Click on the pin with the name of the DP master system in the top right-hand corner of the network view.

Editing DP master systems and interfaces

Introduction

Once you have created a DP master system, you also have the option of disconnecting the DP master system from its components. This can result in subnets with DP slaves but without DP master.

Generally, there is no need to edit the interfaces of a DP master.

You can change the name and number on the DP master system.

Disconnection of master or slaves from the DP master system

If you have configured a PROFIBUS CP as a DP master with master system, you can then disconnect the DP master system from the DP master. After this, the device will no longer be connected to the DP master system.

Disconnecting the subnet from a DP master effectively eliminates the master system in the sense that it is no longer assigned to a DP master. However, the individual DP slaves are still interconnected via the subnet.

If you delete the DP slaves or disconnect them from the master system, the master system is then retained on the DP master.

Requirement

- You must be in the network view.
- There has to be a DP master system with one DP master and at least one DP slave.

Disconnecting the DP master from the DP master system

To disconnect the DP master system, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Disconnect from master system" from the shortcut menu.

The selected DP master will be disconnected from the DP master system. A subnet with the DP slaves will be retained.

Adding a DP master to the DP master system

To reassign a DP master to a subnet, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Create master system" from the shortcut menu.
3. Draw connecting lines between the new DP master system and the DP interfaces of the DP slaves.

The DP master combines with the DP slaves to recreate a DP master system.

Editing the properties of a DP master system

To edit the properties of a DP master system, proceed as follows:

1. Move the mouse pointer over a subnet with a DP master system.
2. A message will appear displaying the available DP master systems. Click the one you want to edit. The DP master system will now be color-highlighted.
3. Click on the highlighted DP master system.
4. In the inspector window, edit the DP master system attributes under "Properties > General".

Note

If you click a subnet when no DP master system is highlighted, you will be able to edit the properties of the entire subnet under "Properties" in the inspector window.

Adding DP slaves to the master system and configuring them

In the network view, you can add various DP slaves from the hardware catalog directly by using the drag-and-drop function or by double-clicking.

Types of DP slaves

For configuration purposes, DP slaves are broken down into the following categories:

- Compact DP slaves
  (Modules with integrated digital/analog inputs and outputs, for example, ET 200L)
- Modular DP slaves
  (interface modules with S5 or S7 modules assigned, for example ET 200M)
- Intelligent DP slaves (I-slaves)
  (SIMATIC S7-300 with, for example, CP 342-5, CPU 315-2 DP, or ET 200S with IM 151-7 CPU)

Slot rules

- Your DP master system should only contain one DP master, but it may contain one or or more DP slaves.
- You may only configure as many DP slaves in a DP master system as are permitted for the specific DP master.

Note

When configuring the DP master system, remember to observe the DP master technical data (max. number of nodes, max. number of slots, max. quantity of user data). User data restrictions may possibly prevent you from being able to use the maximum number of nodes that is theoretically possible.

Requirements

- You are in the network view.
- A DP master system must have been created.

Adding a DP slave to the DP master system

To add a DP slave from the hardware catalog to the DP master system, follow these steps:

1. Select a DP slave from the hardware catalog.
2. Drag-and-drop the DP slave from the hardware catalog into the network view.
3. Draw a connecting line between the DP master's DP interface or a highlighted DP master system and the DP interface of the new DP slave.

A DP master system will be created and the DP slave will be connected to the DP master automatically.

Note

When a DP master system is highlighted, you can double-click the required DP slave in the hardware catalog. This will result in the DP slave being added to the highlighted DP master system automatically.

Disconnecting a DP slave from the DP master system

To disconnect a DP slave from the DP master system, follow these steps:

1. In the network view, right-click on the DP slave's DP interface.
2. From the shortcut menu, select the method for disconnecting from the DP master system:
   – "Disconnect from subnet": The PROFIBUS connection is broken and the device is no longer connected to the DP master system or a subnet.
   – "Disconnect from master system": The DP slave remains connected to the subnet, but is no longer assigned to the DP master system as a DP slave.

The selected DP slave will be disconnected from the DP master system.

You can also disconnect directly using the shortcut menu of the DP slave:

1. In the network view, right-click on the DP slave.
2. Select "Disconnect from DP master/IO system" from the shortcut menu.

The DP slave remains connected to the subnet, but is no longer assigned to the DP master system as a DP slave.

Assigning a DP slave to a new DP master system

To assign an existing DP slave to a new DP master system, follow these steps:

1. Right-click on the DP slave's DP interface.
2. From the shortcut menu, select "Assign to new master".
   It does not matter whether the DP slave concerned is already assigned to another DP master system.
3. From the selection list, select the DP master with the DP master system to which you want to connect the DP slave.
   The selected DP slave is now assigned to a new DP master system.

The "Assign to new subnet" function works in a similar way in that it allows you to connect a DP slave to a new subnet. However, in this case, the DP slave will not be connected to an existing DP master system.

You can also make the assignment directly using the shortcut menu of the DP slave:

1. Right-click the DP slave.
2. From the shortcut menu, select "Assign new DP master/IO system".
3. Select the DP master from the list of available DP masters.

The selected DP slave has now been assigned to a new DP master system.

Note

If there is only one DP master on a subnet, you only have to connect the interface of the DP slave to the subnet of this DP master with drag-and-drop. The DP slave is then connected to the subnet of the DP master immediately, and the DP slave is assigned to this DP master.

Configuring a DP slave

To configure a DP slave, follow these steps:

1. Switch to the DP slave's device view.
2. Select the module you want.
3. Configure the DP slave in the Inspector window.

Hint: Quick configuration of master systems

If the DP master system has several DP slaves, use drag-and-drop operation to assign to the master in one step all DP slaves that were placed.

Requirements

DP master and DP slaves are placed in the network view.

Assigning DP slaves to a DP master system

To do this, follow these steps:

1. Select an appropriate zoom factor so that you can see as many DP slaves as possible in the network view.
2. Arrange the DP slaves in a maximum of two rows.
3. Select all DP interfaces with the mouse cursor (not all devices!). This only works if you begin to drag the mouse cursor outside of the first DP slave and release the mouse button at the last DP slave (selection with the lasso).

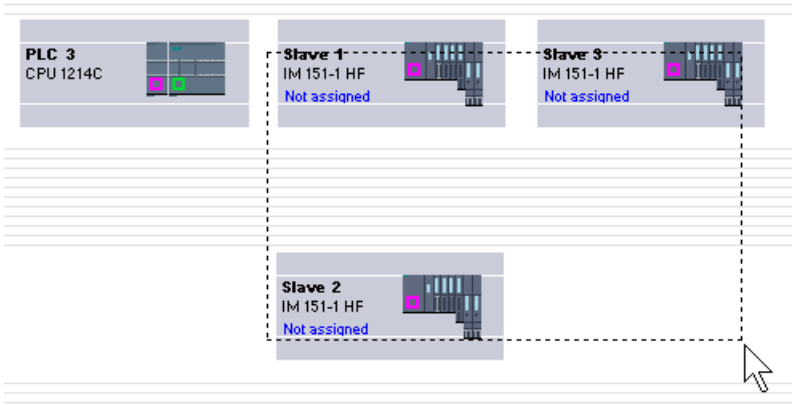

4. Select the shortcut menu "Assign to new master" and select the corresponding DP interface for the DP master in the subsequent dialog.

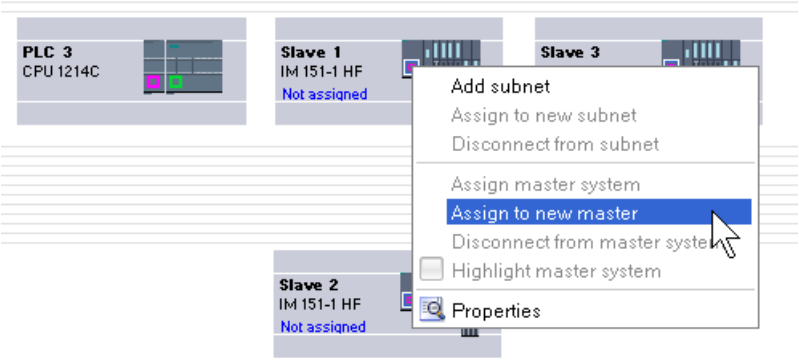

5. The DP slaves are automatically networked with the DP master and combine with it to form a DP master system.

Note

When a DP master system is highlighted, you can double-click on a DP slave in the hardware catalog and thereby quickly add additional DP slaves. This will result in the DP slave being added to the highlighted DP master system automatically.

1.4.10.6 Configuring intelligent DP slaves

Adding an I-slave to a DP master system

Introduction

One of the characteristics of an intelligent DP slave (I-slave) is that the DP master is not provided with I/O data directly by a real I/O, but by a preprocessing CPU. Together with the CP, this preprocessing CPU forms the I-slave.

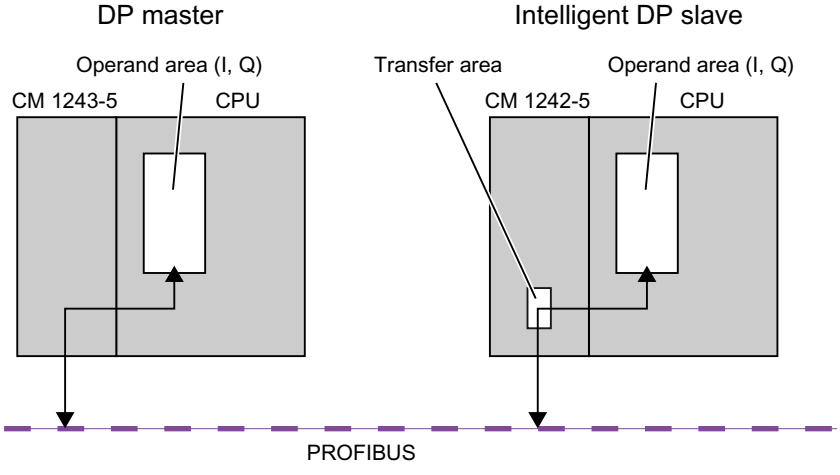

Difference: DP slave v. intelligent DP slave

In the case of a DP slave, the DP master accesses the distributed I/O directly.

In the case of an intelligent DP slave, the DP master actually accesses a transfer area in the I/O address space of the preprocessing CPU instead of accessing the connected I/O of the intelligent DP slave. The user program running on the preprocessing CPU is responsible for ensuring data exchange between the address area and I/O.

Note

The I/O areas configured for data exchange between the DP master and DP slaves must not be used by I/O modules.

Applications

Configurations involving intelligent DP slaves: I-slave <> DP master data exchange

Procedure

To add an I-slave to a DP master system, follow these steps:

1. In the network view, drag from the hardware catalog to a station one CM 1242-5 as an I-slave and one CM 1243-5 as a DP master.
2. Draw a connecting line between the DP interfaces of both devices.
   This way you connect the I-slave with a DP master in a DP master system.
   Result: You have now set up a DP master system with one DP master and one I-slave.

Configuring access to I slave data

Data access

The following applies to the CP 1242-5 in its function as I-slave: The addresses for the data transfer area and the address for the I/O modules in the I-slave differ. This means that the start address occupied by an I/O module can no longer be used for the transfer memory. If the higher-level DP master is to access the data of an I/O module in the I-slave, you must configure this data exchange between the I/O module and transfer area in the I-slave user program.

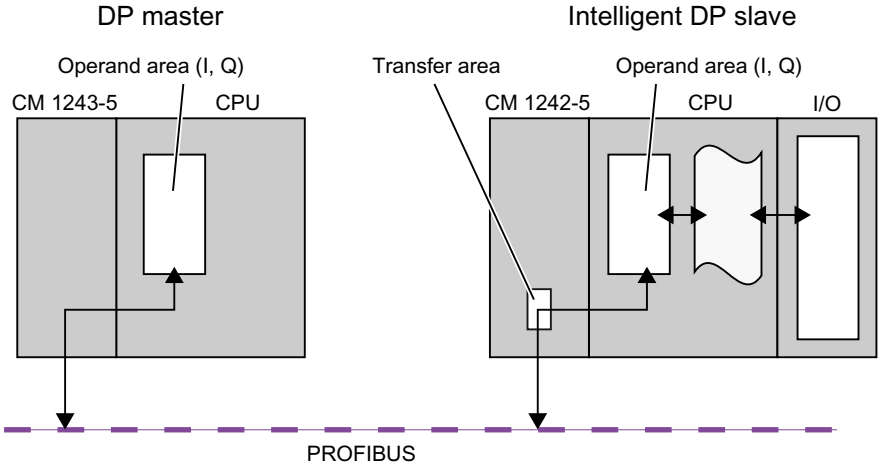

Configuration of the transfer area for the CM 1242-5 (transfer area)

With CM 1242-5, the transfer area for the cyclic PROFIBUS data exchange is configured as transfer area in the parameter group "PROFIBUS interface > Mode > I Slave Communication".

Direct data access from CPU to CPU

Direct data access from CPU to CPU via PROFIBUS is supported by the S7-1200 PROFIBUS CMs only via the PUT/GET services.

1.4.10.7 Configuring DP slaves as distributed I/O devices

Configuring an ET 200S

Slot rules for configuring an ET 200S

The following rules apply when configuring an ET 200S:

- Do not leave any gaps when inserting the ET 200S modules.
- Slot 1: only for PM-E or PM-D Power Modules.
- To the left of an Electronics Module (EM): an EM or a Power Module (PM-E or PM-D) only.

- To the left of Motor Starter (MS): an MS, a PM-D, PM-D Fx (1..x..4) Power Module or a PM-X Power Module only.
- To the left of a PM-X: a motor starter or a PM-D only
- Up to 63 modules and one IM Interface Module are permitted

Note

Remember to ensure that the correct PM-E and EM voltage ranges are assigned.

Configuring reference junctions

A reference junction is the connection of a thermocouple to a supply line (generally in the terminal box). The voltage that occurs due to the effects of temperature falsifies the temperature value measured by the module.

On the ET 200S, one channel of the AI RTD module can be programmed as a reference junction. Other AI TC modules can correct their measured values using the temperature at the reference junction as measured by this module.

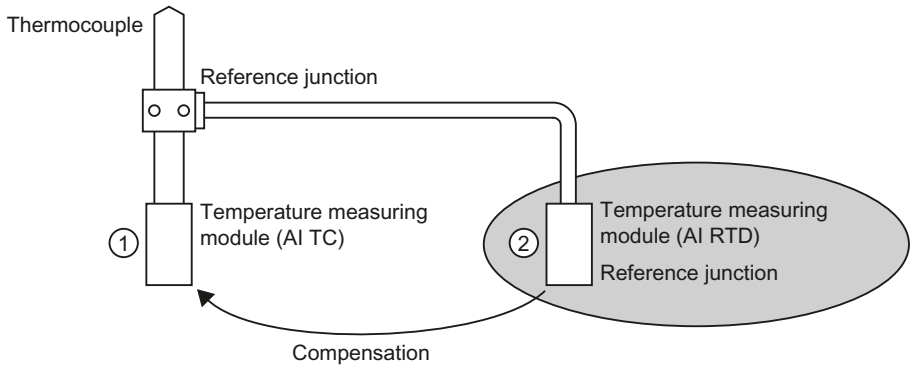

① Configuring the AI TC:
→ Selection of the reference junction used

② Configuring of the AI RTD:
→ Activation of the reference junction
→ Specifying the slot and channel of the AI RTD

Special characteristics to be noted when assigning parameters for reference junctions

The process of assigning parameters for reference junctions will be explained based on the example of a resistance thermometer with a Pt 100 climatic range that is used for measuring the reference junction temperature.

To assign parameters for the reference junction, follow these steps:

1. In the ET 200S device view, insert an analog electronics module, for example a 2AI RTD HF.
2. Select the module on the rack.

3. Under "Properties > Inputs" in the inspector window, set a channel for the reference junctions function to the "RTD-4L Pt 100 climatic range" measuring range.
4. Select the ET 200S.
5. Under "Properties > Module parameters > Reference junctions" in the inspector window, select the "Reference junction" check box and specify the slot and channel number of the relevant RTD module.
6. Insert the analog electronics module for measuring the temperature using a thermocouple (TC module) and parameterize it with the reference junction number of the RTD module.

Additional information

For additional information on the various types and uses of ET 200S modules, please refer to the operating instructions and the manual titled "ET 200S Distributed I/O System".

For additional information on analog value processing, please see the documentation for the ET 200S distributed I/O system.

Packing addresses

Introduction

DP slaves and I/O devices from the ET 200S family are configured in the same way as other modular DP slaves and I/O devices. As well as supporting all the standard modular DP slave and I/O device functions, the ET 200S also offers the "Pack addresses" function:

When digital electronics modules requiring an address space of 2 or 4 bits are inserted into the device view, they will initially be spread over a total area of 1 byte. However, the address area actually occupied can be compressed after configuration using the "Pack addresses" function.

	Initial state	After "Pack addresses"
Module	I address	I address
2DI (2-bit)	Byte 10	10.0...10.1
4DI (4-bit)	Byte 11	10.2...10.5

Requirements

- You are in the device view.
- An ET 200S, for example an IM 151-1, must be present.
- A pair of digital electronics modules, for example 2DI AC120V ST, must be inserted into the slots.

Packing addresses

To pack addressed, follow these steps:

1. Select the electronics modules whose addresses are to be packed. The following options are available for selecting multiple electronics modules:
   – Press and hold down <Shift> or <Ctrl> while clicking the relevant electronics modules.
   – Click off the rack and select the required electronics modules by drawing round them with the mouse.
2. Click "Pack addresses" in the shortcut menu for the selected electronics modules.

The address areas for inputs, outputs and motor starters are packed separately. The packed addresses will be displayed in the I address and Q address columns of the device overview.

How packed addresses are generated and structured

If you make use of the "Pack addresses" function, the addresses of the selected electronics modules will be packed in accordance with the following rules:

- The start of the address area is determined by the lowest address of the selected electronics modules: X.0
- If the bit address is not "0", then the next (free) byte address as of which the selected area can be compacted will be selected automatically: (X+n).0
- If no coherent area exists, the addresses will be automatically packed into any available address gaps.

Electronics modules with packed addresses and the same byte address form a packing group.

Unpacking addresses

To unpack addressed, follow these steps:

1. Select one or more electronics modules with packed addresses.
2. Click "Unpack addresses" in the shortcut menu for the selected electronics modules.

The packing groups of the selected electronics modules will be disbanded and the packed addresses for the relevant electronics modules unpacked.

The packing group will also be disbanded and the packed addresses unpacked in the following cases: if you delete electronics modules from a packing group, move electronics modules out of a packing group or insert electronics modules on a free slot within a packing group.

The start addresses of the unpacked electronics modules will be assigned to the next available byte addresses in each case.

Special characteristics of electronics modules with packed addresses

The following special characteristics apply to electronics modules with packed addresses:

- As far as the CPU is concerned, there is no way of assigning a slot for the electronics module. Consequently, the instruction GADR_LGC (SFC 5) outputs error information W#16#8099 "Slot not configured" for the actual slot of the electronic module.
- The instruction LGC_GADR (SFC 49) and SZL-ID W#16#xy91 "module status information" cannot be evaluated for an electronics module.
- The electronics module receives an additional diagnostics address via the DPV1 function, because otherwise the packed addresses would prevent interrupts from being assigned as far as the CPU is concerned.
- The "Insert/remove interrupt" is not possible. This is because the "Pack addresses" and "Insert/remove interrupt" functions are mutually exclusive.

Configuring option handling with standby modules

You can use the option handling function to prepare the ET 200S with PROFIBUS interface for future expansions (options). This section describes option handling with standby modules.

You do this by assembling, wiring, configuring, and programming the maximum configuration envisaged for the ET 200S and by using cost-effective standby modules (138-4AA00 or 138-4AA10) during assembly until it becomes time to replace them with the necessary electronics modules.

Note

The ET 200S can be completely factory-wired with the master cabling, as no connection exists between a standby module and the terminals of the terminal module (and, in turn, to the process).

Requirement

- ET 200S interface module
  - IM 151-1 STANDARD (6ES7 151-1AA03-0AB0 or higher)
  - IM 151-1 FO STANDARD (6ES7 151-1AB02-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the numbered check boxes for the slots that are initially to accommodate the standby modules prior to the future electronics modules.
3. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).

The assembled standby modules can be replaced with the configured modules at a later date without having to modify the configuration.

Note

The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 STANDARD Interface Module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/ AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of standby modules, please refer to the documentation for the ET 200S distributed I/O system.

How option handling works during startup

If "Startup when expected/actual config. differ" is not available, the ET 200S will still start up if a standby module is inserted instead of the configured electronics module and option handling has been activated for the slot concerned.

How option handling works during operation

During operation, the option handling mode varies in accordance with the following:

- Option handling enabled for a slot:
  Either the standby module (option) or the configured electronics module can be plugged into this slot. If any other kind of module is present on this slot, diagnostics will be signaled (no module/incorrect module).
- Option handling disabled for a slot:
  Only the configured electronics module can be plugged into this slot. If any other kind of module is present, diagnostics will be signaled (no module/incorrect module).

Standby module substitute values

- Substitute value for digital inputs: 0
- Substitute value for analog inputs: 0x7FFF

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 2 to 63:

- Bit value = 0: Option handling parameterized. Standby modules are permitted.
- Bit value = 1: Option handling canceled. Standby modules are not permitted on this slot.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: This slot has the standby module, an incorrect module or no module plugged into it.
- Bit value = 1: The configured module is plugged into this slot.

See also

Which modules support option handling? (http://support.automation.siemens.com/WW/view/en/22564754)

Configuring option handling without standby modules

You can use the option handling function to prepare the ET 200S for future expansions (options) without installing standby modules. This section describes option handling without standby modules.

Note

ET 200S with PROFINET interface

This description refers to the ET 200S with PROFIBUS interface. In principle, option handling for ET 200S with PROFINET interface functions as described here without standby modules. PN interface modules are to be used instead of the DP interface modules listed here. You can find additional information about option handling for ET 200S with PROFINET interface in the corresponding manuals.

Requirement

- ET 200S interface module
  - IM 151-1 HIGH FEATURE (6ES7151-1BA02 or higher)
  - IM 151-1 STANDARD (6ES7 151-1AA05-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).
3. Configure the slave's maximum configuration. The selection/clearing of options is controlled via the user program.

   Note

   The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 interface module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

   Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

   Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of standby modules, please refer to the documentation for the ET 200S distributed I/O system.

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 1 to 63:

- Bit value = 0: Slot is not available in the actual configuration.
- Bit value = 1: Slot is available in the actual configuration.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: Slot belongs to an option that is not available or module status is faulty.
- Bit value = 1: The configured module is plugged into this slot.

See also

Sample applications for ET 200S, option handling without standby module (http://support.automation.siemens.com/WW/view/en/29430270)

Configuring the ET 200S in DPV1 mode

PROFIBUS DPV1 enables you to access extended PROFIBUS functions.

Requirement

- You must be in network view.
- A DP master with DPV1 functionality must be available.
- A master-slave connection must be established with PROFIBUS.

Procedure

To switch the DP slave over to DPV1, follow these steps:

1. Select the DP slave.
2. Under "Properties > Module parameters" in the Inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list.

or

1. Select the DP master.
2. In the I/O communications table, select the row with the connection between the DP master and the desired DP slave.
3. Under "Properties > Module parameters" in the Inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list.

Special characteristics

The parameters are subject to interdependencies, which are outlined below:

Parameter	DPV0 mode	DPV1 mode
Operation when target configuration does not match actual configuration	Fully available	Fully available
Diagnostics interrupt (OB 82)	Not available, not set	Fully available
Hardware interrupt (OB 40 to 47)	Not available, not set	Fully available
Insert/remove interrupt (OB 83)	Not available, not set	Only available when addresses are not packed. "Startup when target configuration does not match actual configuration" is activated automatically along with an insert/remove interrupt.

Interrupts in the case of modules with packed addresses

If the module is capable of triggering interrupts and the bit address is not equal to 0 because of packed addresses, you will need to assign a diagnostics address in the ET 200S address dialog.

The diagnostics address is required for the purpose of assigning a DPV1 interrupt to the module as an interrupt trigger. The CPU will only be able to assign an interrupt correctly and store information on the interrupt in the interrupt OB start information/in the diagnostics buffer if the module concerned has this "unpacked" address. In this context, the CPU cannot make use of "packed" addresses.

From the point of view of interrupt processing (interrupt OB), this means the module will have the assigned diagnostics address, but from the point of view of processing the input and output data in the user program, the module will have the packed addresses.

Note

When the module addresses are packed, the insert/remove interrupt for the ET 200S is unavailable.

1.4.10.8 Using GSD files

GSD revisions

What you need to know about GSD revisions

The properties of DP slaves are made available to configuration tools by means of GSD files. Functional enhancements in the area of the distributed I/O will have an effect on the GSD specification, for example, they will require the definition of new keywords. This results in the versioning of the specification. In the case of GSD files, the version of the specification on which a GSD file is based is called a "GSD revision". From GSD revision 1, the GSD revision must be included as a keyword "GSD_revision" in GSD files. GSD files without this keyword will therefore be interpreted by configuration tools as GSD revision "0".

GSD files can be interpreted up to GSD revision 5. This means that DP slaves that support the following functions, for example, will be supported:

- Diagnostic alarms for interrupt blocks
- Isochronous mode and constant bus cycle time
- SYNC/FREEZE
- Clock synchronization for DP slaves

Installing the GSD file

Introduction

A GSD file (generic station description file) contains all the DP slave properties. If you want to configure a DP slave that does not appear in the hardware catalog, you must install the GSD file provided by the manufacturer. DP slaves installed via GSD files are displayed in the hardware catalog and can then be selected and configured.

Requirement

- The hardware and network editor is closed.
- You have access to the required GSD files in a directory on the hard disk.

Procedure

To install a GSD file, follow these steps:

1. In the "Options" menu, select the "Manage general station description files (GSD)" command.
2. In the "Installed GSDs" tab, select the directory in which the GSD files are stored.
3. Choose one or more files from the list of displayed GSD files.
4. Click on the "Install" button. The selected GSD files are now being installed.

5. To create a log file for the installation, click on the "Save log file" button.
   Any problems during the installation can be tracked down using the log file.
6. Click "Close". You will be notified that the DP slaves from the installed GSD files are entered into the hardware catalog. This process can take a few seconds.

By installing a GSD file you have added the DP slave described in the file to the hardware catalog.

Result

You will find the new DP slaves installed by means of GSD files in the hardware catalog under "Additional field devices > PROFIBUS DP".

GSD files are always saved together with the project, which means all information relevant for display of the device (including symbols) is also available in the saved project.

See also

Overview of hardware and network editor (Page 9)

Deleting GSD file

Introduction

You can delete installed DP slaves using GSD files. These DP slaves are then no longer displayed in the hardware catalog.

Requirement

- The hardware and network editor is closed.
- The hardware catalog contains DP slaves installed using GSD files.

Procedure

To delete a GSD file, follow these steps:

1. In the "Options" menu, select the "Manage general station description files (GSD)" command.
2. In the "Installed GSDs" tab, select the directory in which the GSD file is stored.
3. Select the file that is to be deleted from the list of displayed GSD files.
4. Click the "Delete" button.

The selected GSD file was deleted and the DP slave is no longer located in the hardware catalog.

Finding unused GSD files

Introduction

When you add a GSD-based hardware component to your hardware configuration, different GSD files are added to the project data. If the GSD-based hardware component is once again deleted from the configuration, the GSD files remain in the data inventory of the project. The GSD files are no longer needed in the project if the corresponding GSD-based hardware component is not used in the hardware configuration again. You can find files that are no longer needed in the project with a search function and delete them.

Requirement

- A project must be open.
- The project must not be online.
- The project must not be write-protected.

Procedure

To find and delete unused GSD files, follow these steps:

1. In the "Options" menu, select the "Manage general station description files (GSD)" command.
2. Click on the "Find unused GSDs" button in the "GSDs in the project" tab.

   Note

   Unused GSD files are only available if at least one GSD-based hardware component was added to the configuration and then removed from the configuration. Unused GSD files in the project libraries are included in the search. GSD-based hardware components that are still being used in the configuration are not considered to be unused.

   Installed GSD-based hardware components that are included in the hardware catalog but have not been used in the hardware configuration of the project yet, could not have stored unused GSD data in the project.

3. If GSD files that are no longer needed were found in the project data, you can now delete them. To do so, click the "Delete" button.

GSD files that are no longer needed are deleted from the project data. The GSD-based hardware component is still available in the hardware catalog and, if necessary, can be added to the configuration once again. The necessary GSD data are once again copied to the project data in this case.

Note

If you delete a GSD-based hardware component with "Options > Manage general station description files (GSD) > Installed GSDs", it is deleted from the hardware catalog. However, unused GSD files of this hardware component can still be a part of the project data and can be deleted with the function for finding the unused GSD files.

NOTICE
Actions that cannot be undone When you delete GSD files that are not being used as an action, the action stack for actions that can be undone is deleted. This means the action of deleting unused GSD files as well as all previous actions can no longer be undone.

Configuring GSD-based DP slave

DP slaves that you have inserted through installation of a GSD file can be selected as usual via the hardware catalog and inserted in the network view. If you want to insert the modules of the GSD-based DP slaves, you must take into account some particular details.

Requirement

- You have installed a DP slave using a GSD file.
- You have inserted the head module in the network view in the usual manner.
- The device overview opens in the device view.
- The hardware catalog is open.

Procedure

To add the modules of a GSD-based DP slave, proceed as follows:

1. In the hardware catalog, navigate to the modules of the GSD-based DP slave.
   GSD-based DP slaves, also referred to as DP standard slaves, can be found in the "Other field devices" folder of the hardware catalog.
2. Select the desired module.
3. Use drag-and-drop to move the module to a free space in the device overview.
4. Select the module in the device overview to edit parameters.

You have now inserted the module in a free slot of the GSD-based DP slave and can edit its parameters.

Note

You can see only the GSD-based DP slave in the graphic area of the device view. The added modules of GSD-based DP slaves are only found in the device overview.

Preset configuration

For modules with an adjustable preset configuration, you can change this configuration in the inspector window under "Properties > Preset configuration".

1.4.11 Configuring PROFINET IO

1.4.11.1 What you need to know about PROFINET IO

What is PROFINET IO?

PROFINET IO

PROFINET is an Ethernet-based automation standard of PROFIBUS Nutzerorganisation e.V. (PNO) which defines a manufacturer-neutral communication, automation and engineering model.

Objective

The objective of PROFINET is:

- Integrated communication via field bus and Ethernet
- Open, distributed automation
- Use of open standards

Architecture

The PROFIBUS User Organisation e.V. (PNO) has designated the following aspects for PROFINET architecture:

- Communication between controllers as components within distributed systems.
- Communication between field devices, such as I/O devices and drives.

Implementation by Siemens

The demand for "Communication between controllers as components within distributed systems" is implemented by "Component Based Automation" (CBA). Component Based Automation is used to create a distributed automation solution based on prefabricated components and partial solutions.

The demand for "Communication between field devices" is implemented by Siemens with "PROFINET IO". Just as with PROFIBUS DP, the complete configuration and programming of the components involved is possible using the Totally Integrated Automation Portal.

The following sections deal with the configuration of communication between field devices using PROFINET IO.

Overview of RT classes

RT classes in PROFINET IO

PROFINET IO is a scalable, real-time communication system based on Ethernet technology. The scalable approach is reflected in several real-time classes:

- **RT:** Transmission of data in prioritized, non-isochronous Ethernet frames. The required bandwidth is within the free bandwidth area for TCP/IP communication.
- **IRT:** Isochronous transmission of data with high stability for time-critical applications (for example, motion control). The required bandwidth is from the area of bandwidth reserved for cyclic data.

Depending on the device, not all real-time classes are supported.

Which IO controllers and IO devices support which PROFINET functions?

Additional information and overviews

In the following Siemens Industry Online Support article (http://support.automation.siemens.com/WW/view/en/44383954), you will find an overview of the PROFINET IO controllers and IO devices that support the following PROFINET functions:

- Isochronous real-time communication (IRT)
- Prioritized startup
- Media redundancy (MRP)
- PROFIenergy
- Shared device
- I-device
- Isochronous mode of process data

The functions are explained in the following sections, but without naming the respective hardware that supports these functions.

In the hardware catalog, you can also find an overview of the supported functions in the description below the selected components.

You can also find a description of PROFINET in the respective current STEP 7 version here (http://support.automation.siemens.com/WW/view/en/49948856).

Connecting existing bus systems

Connection of PROFINET and PROFIBUS

PROFINET IO and PROFIBUS DP can be connected with each other as follows:

- via Industrial Ethernet:
  To connect the two network types, Industrial Ethernet (plant control level) and PROFIBUS (cell level/field level), use the IE/PB link, for example.
- via Industrial Wireless LAN:
  PROFIBUS devices, for example, can be connected to PROFINET IO via a wireless LAN/PB link. This allows existing PROFIBUS configurations to be integrated into PROFINET.

AS interface devices can be connected by an IE/AS-i link PN IO to the interface of a PROFINET device. This allows the existing AS-i network to be integrated into PROFINET.

The following figure shows the connection of a PROFIBUS subnet via a PROFINET device with proxy functions.

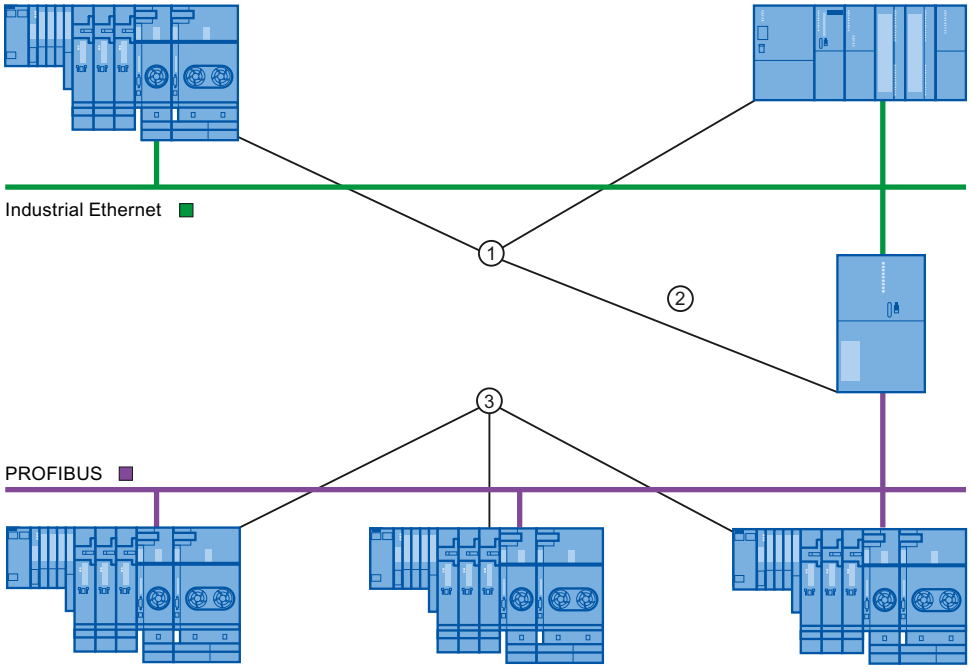

① PROFINET devices
② PROFINET device with proxy functions (for example, IE/PB link)
③ PROFIBUS devices

PROFINET device with proxy functions used as proxy for a PROFIBUS device

The PROFINET device with proxy functions is the proxy for a PROFIBUS device on the Ethernet. The proxy functionality enables a PROFIBUS device to communicate with all devices on the PROFINET and not just with its master.

Existing PROFIBUS systems can easily be integrated into PROFINET communication using the proxy functions.

If, for instance, you connect a PROFIBUS device via an IE/PB link to PROFINET, the IE/PB link acts as a proxy for the PROFIBUS components for communicating via PROFINET.

Configuration using IE/PB link PN IO

Configuration using IE/PB Link PN IO

You can use the IE/PB Link PN IO to connect PROFIBUS DP configurations to PROFINET IO.

From the CPU perspective, the PROFIBUS DP slaves are connected to the same network as the IE/PB Link PN IO. These slaves have the same device name and IP address as the IE/PB Link PN IO, but different device numbers. Furthermore, each also has a specific PROFIBUS address.

In the properties of the IE/PB link, the PROFIBUS addresses of the connected DP slaves are displayed in addition to the PROFINET device numbers, as this device has two addressing schemes.

Handling device numbers and PROFIBUS addresses on the master system

During placement, the same number is assigned for the PROFINET device number and the PROFIBUS address.

In the Inspector window under "General Properties > PROFINET device number", you can find an overview of the device numbers used and the PROFIBUS addresses of an IE/PB link. The device number can also be changed here. You can also set that the device number and the PROFIBUS address should or should not always be identical. If the "PROFINET device number=PROFIBUS address" option is activated, you do not have to update the device number when the PROFIBUS address changes.

The PROFIBUS address can be changed in the properties of the PROFIBUS device.

Restrictions

The following restrictions apply to DP slaves configured as described above on the PROFIBUS subnet of an IE/PB link:

- No pluggable IE/PB link
- No pluggable DP/PA link
- No pluggable Y link
- Not CiR-compliant
- No pluggable redundant slaves
- No isochronous transmission / constant bus cycle time can be configured
- SYNC/FREEZE instructions ("DPSYC_FR") of a CPU on the the Ethernet subnet for DP slaves behind the IE/PB-Link are not supported.

See also

Connect the DP slave via the IE/PB Link to a PROFINET IO system (Page 2878)

Configuration using IWLAN/PN link

Maximum number of devices in a IWLAN segment

If an Ethernet subnet is set up as wireless network (IWLAN = Industrial Wireless LAN), cyclic data exchange between IO controllers and IO devices is possible via a wireless route.

On one side of the wireless route there are fixed installed access points (for example, SCALANCE W 788) and on the other side mobile stations (with, for example IWLAN/PB links with PROFIBUS devices).

If the action radius of the mobile stations is large, it may be necessary to install several access points (SCALANCE W 788). Since each access point forms a segment with its wireless range, the IWLAN is made up of a series of segments.

The mobile devices "on the one side" of the wireless link with their IWLAN/PB links can move along the segments.

Special feature

If several IWLAN/PB links are located within a segment, they have to share the bandwidth that is available for wireless transmission. This leads to a lengthening of the update time for these devices.

Example

In the following example there are two IO devices (IWLAN/PB link) with a segment.

If no more than a maximum of two IWLAN/PB links are present in a IWLAN segment at the same time, enter a "2".

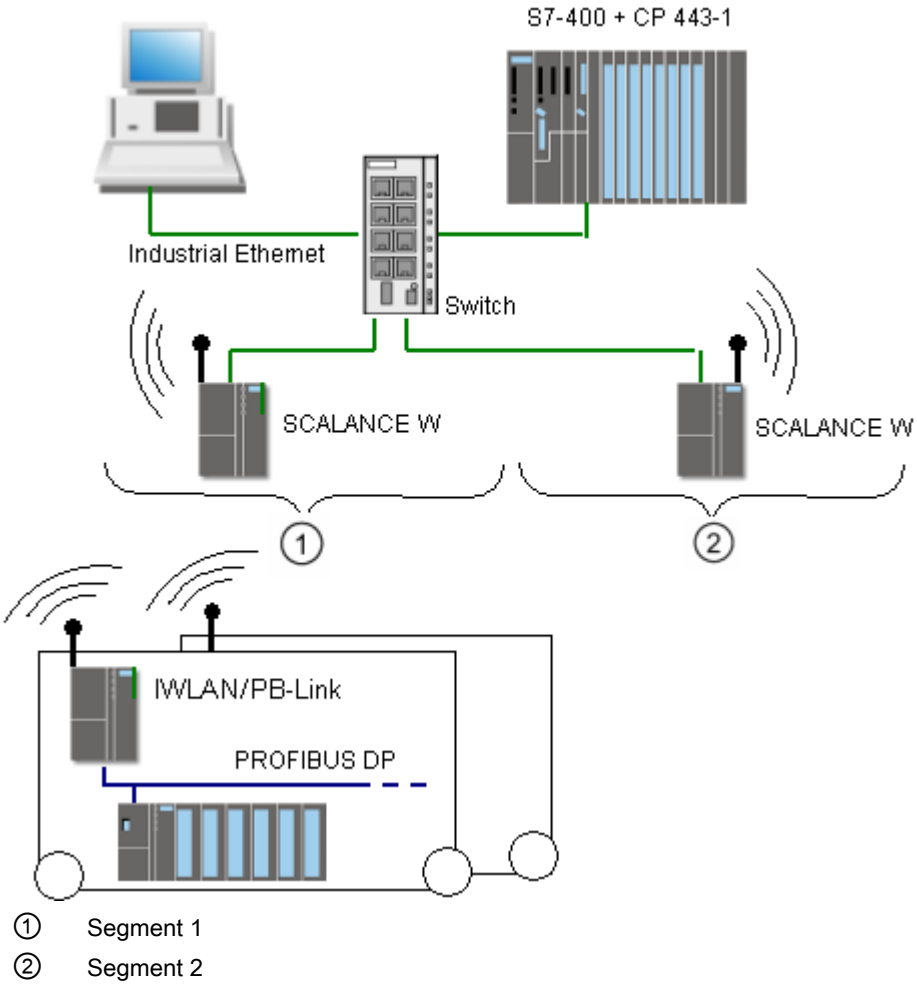

1.4.11.2 Configure PROFINET IO

Addressing PROFINET devices

Assigning addresses and names to PROFINET devices

In this chapter you will learn which address and naming conventions are valid for the PROFINET devices.

IP addresses

All PROFINET devices work with the TCP/IP protocol and therefore require an IP address for Ethernet operation.

You can set the IP addresses in the module properties. If the network is part of an existing company Ethernet network, ask your network administrator for this data.

The IP addresses of the IO devices are assigned automatically, usually at CPU startup. The IP addresses of the IO devices always have the same subnet mask as the IO controller and are assigned in ascending order, starting at the IP address of the IO controller.

Device names

Before an IO device can be addressed by an IO controller, it must have a device name. This procedure was chosen for PROFINET because names are easier to administer than complex IP addresses.

Both the IO controller as well as IO devices have a device name. When the "Generate PROFINET device name automatically" option is activated, the device name is automatically derived from the name configured for the device (CPU, CP or IM):

- The PROFINET device name is made up of the name of the device (for example, the CPU), the name of the interface (only with multiple PROFINET interfaces) and optionally the name of the IO system:
  <CPU name>.<Name of the interface>.<IO system name>
  You cannot change this name directly. You change the PROFINET device name indirectly, by changing the name of the affected CPU, CP or IM in the general properties of the module. This PROFINET device name is also displayed, for example, in the list of accessible devices. If you want to set the PROFINET device name independently of the module name, you have to deactivate the "Generate PROFINET device name automatically" option.
- A "converted name" is generated from the PROFINET device name. This is the device name that is actually loaded into the device.
  The PROFINET device name is only converted if it does not comply to the rules of IEC 61158-6-10. You cannot change this name directly either.

Rules for the converted name

The rules for the converted name are listed in the following section. If the converted name is **not** different from the name of the module, the name of the module must comply with this rule.

- The name consists of one or more labels , which are separated by a dot [.].
- Restricted to a total of 240 characters (lower case letters, numbers, dash, or dot)
- A name component within the device name, which means a character string between two dots, must not exceed 63 characters.
- A name component consists of the characters [a-z, 0-9].
- The device name must not begin or end with the "-" character.
- The device name must not begin with a number.
- The device name form n.n.n.n (n = 0, ... 999) is not permitted.
- The device name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).

Example of device names

```
device-1.machine-1.plant-1.vendor
```

If you assign this name to a CPU, for example, STEP 7 will not convert it since it conforms to the rules described above.

Device number

In addition to the device name, a device number is also automatically assigned when an IO device is plugged in. You can change this number.

Devices in the PROFINET subnet

In a PROFINET subnet the maximum allowable number of devices is monitored during configuration.

See also

Assigning the device name and IP address (Page 2842)

Retentivity of IP address parameters and device names (Page 2852)

Assigning the device name and IP address

Assigning an IP address and subnet mask for an IO controller the first time

There are various options for this.

During parameter assignment of the PROFINET interface you must specify if the IP address is set in the project (which means in the hardware configuration) or if the IP address is to be set on the device.

Assignment of an IP address	Comments
Option "IP address is set in the project": The IO controller receives the IP address by loading the hardware configuration, for example, via one of the PROFINET interfaces, by means of the PROFIBUS interface or via the MPI interface.	When the hardware configuration is loaded to the IO controller (e.g. CPU), the IP address and the device name, if set, are also loaded. Example PROFINET interface: 1. Connect your programming device/PC to the same network as the relevant PROFINET device. The interface of the PG/PC must be set to TCP/IP (Auto) mode. 2. Have a list of accessible devices displayed. 3. Select the target device by using its MAC address and load the hardware configuration including the configured IP address (IP address is then saved retentively). If your PROFINET device has an MPI or PROFIBUS DP interface, connect your programming device/PC directly to the PROFINET device via the MPI or PROFIBUS DP interface. The configured IP address is applied when the hardware configuration is loaded.
Option "IP address is set directly at the device": • Assign online • Assignment by user program (instruction IP_CONFIG for S7-300/400, T_CONFIG for S7-1200/1500) • Assign via CPU display (S7-1500) • Higher-level IO controller makes assignment (only with I-devices)	If you have selected this option in the properties of the PROFINET interface, the IP address can be assigned by the online and diagnostics editor, by the primary setup tool, or by the user program ("IP_CONFIG" instruction). This option is set automatically if the option "Multiple use IO system" has been enabled in the properties of the PROFINET IO system (standard machine project). In case of an S7-1200-CPU, make sure that access to the CPU is not protected by a password. If the CPU is write-protected, no IP address and no device name can be assigned directly on the device.

Commissioning a PROFINET interface

Further details of how to commission a PROFINET interface can also be found in the operating instructions for the PROFINET devices of the SIMATIC family.

Assignment of device name for IO devices when the "Support device replacement without exchangeable medium" option is selected

For IO controllers with the "Support device replacement without exchangeable medium" option selected, you do not have to assign device names to the IO devices locally, for example in the event of device replacement. Another application is automatic commissioning, in which the CPU automatically assigns the device name and IP address parameters to the IO devices during startup.

Requirement: The ports of the devices are interconnected, and the devices involved support LLDP. The devices have been put into the delivery state or – for S7-1500 CPUs version V1.5 and higher – the "Permit overwriting of device names of all assigned IO devices" option is selected for the IO controller ("Advanced options > Interface options" area of the properties of the PROFINET interface).

Assignment of device name and address for an IO device

The following graphic illustrates the process for assigning the device name and address. This process does not apply when the "Support device replacement without exchangeable medium" option is selected.

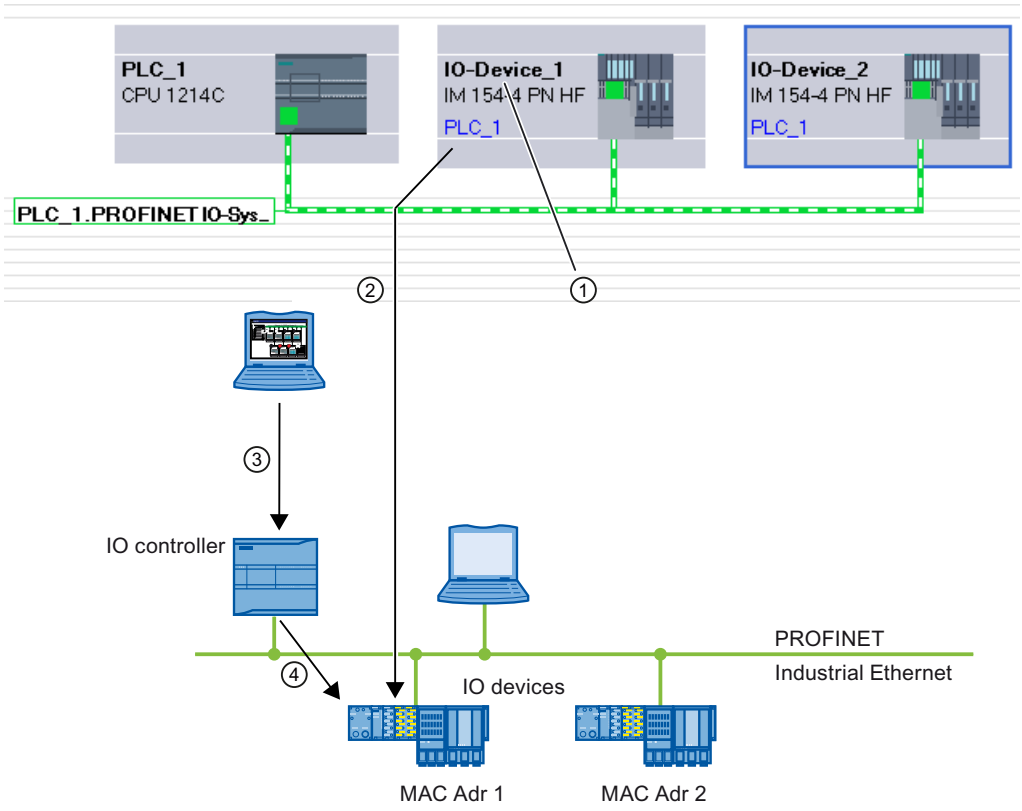

① Each device receives a name; STEP 7 automatically assigns an IP address.

② From the name, STEP 7 generates a PROFINET device name that you assign to an IO device online (MAC address) and that is written to the device.

③ You load the configuration to the IO controller.

④ The IO controller assigns the appropriate IP address to the IO device with the assigned PROFINET device name during startup.

Changing the device name and IP address

You can change the name and IP address manually. The device name must first be changed in the configuration in order to subsequently assign it to the IO device via the memory card or online with programming device/PC.

Offline with memory card:

1. Place the configured data (device name: for example, turbo-3) for the IO device on the MMC in the PG/PC. Use the command "SIMATIC Card Reader > Save Device Name to Memory Card" in the "Project" menu for this.
2. Then insert the MMC into the IO device. The IO device automatically adopts the configured device name.

Online with programming device/PC:

1. Connect the programming device/PC directly to the Ethernet subnetwork via the PROFINET interface.
2. In the network view, select the subnet or I/O device, and click the "Assign device name" command:
   – Either in the shortcut menu of the selected subnet / I/O device or
   – in the menu bar of the graphic view on the corresponding button.
3. In the "Assign PROFINET device name" dialog box, select the suitable PG/PC interface to connect to the Ethernet subnet. All configured PROFINET device names are in the top drop-down list. Select a PROFINET device name from it and select the IO device to receive this device name from the table at the bottom. You can filter the display of devices in the table according to various criteria.
4. You can easily identify the device using the "Flash LED" button.
5. Click "Assign name".

The IO controller recognizes the IO device by its device name and automatically assigns the configured IP address to it.

IP address assignment for special IO devices

Special IO devices, for example, SCALANCE X, S7-300 CPs, support the option of assigning the IP addresses not from the IO controller during startup. In this case, the IP address is assigned in a different way. The option is called "IP address is set directly at the device". For additional information, refer to the manual of the respective PROFINET device of the SIMATIC device family.

Another special case is the option "IP address is set directly at the device" in the "IP protocol" area of the Ethernet address properties of an IO device. This option is set automatically when the option "Multiple use IO system" is selected for a standard machine project in the associated PROFINET IO system. In this case, an adapted IP address is not assigned by the IO controller until the IO controller itself has received a local IP address.

Requirement for additional procedures when assigning IP address and device name

If the IO device, as described above, should not obtain the IP address or device name from the IO controller, proceed as follows:

1. Select device or network view.
2. Open the properties of the respective PROFINET device and select the area "PROFINET interface [X1]" > "Ethernet addresses".
3. Select the option "IP address is set directly at the device" under "IP protocol" or the option "PROFINET device name is set directly at the device" under "PROFINET".

Rules

If the "IP address/device name is set directly at the device" option is used for a PROFINET device, note the following:

- The subnet part of the IP address of the IO device must match the subnet part of the IP address of the IO controller.
- The corresponding PROFINET device cannot be used as a gateway.

See also

Assigning a name in the online and diagnostics view opened via "Accessible devices" (Page 3344)

Enabling device replacement without exchangeable medium (Page 2876)

Example of the assignment of the device name

In this example you assign device names to a PROFINET IO controller and a PROFINET IO device. To make assignment easier, the device names should also contain the names of the PROFINET IO system.

Requirement

- You must be in the network view.
- A CPU 1214C (V2.0 or higher) must be available in the network view.
- An interface module IM 151-3PN exists.
- The PROFINET interfaces of both modules are networked.

Procedure

To assign the names, follow these steps:

1. Select the CPU.
   Make sure that you have selected only the CPU and not the complete device!
2. Assign the name "myController" in the Inspector window, under "General".

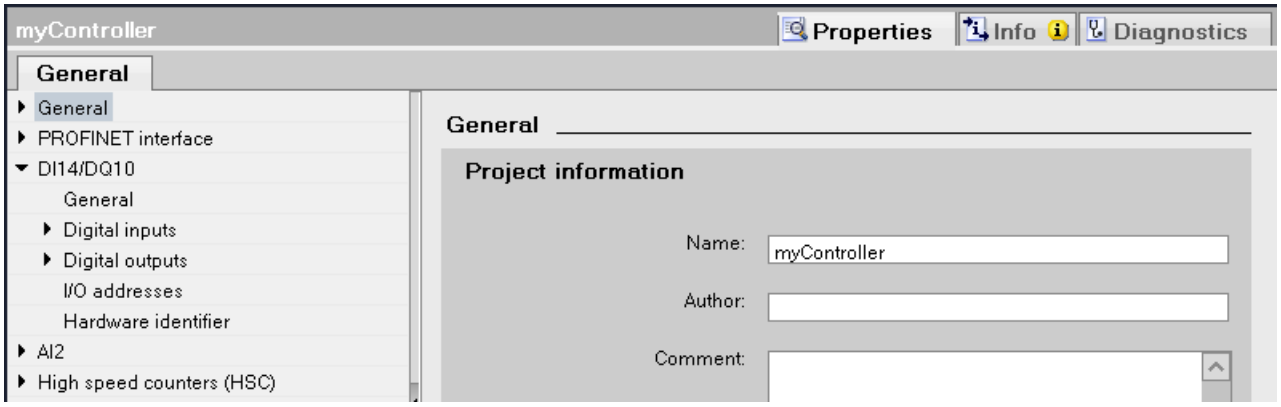

3. Select the interface module.
   Ensure that you have selected only the interface module and not the complete ET 200S device.
4. Assign the name "Device_1" in the Inspector window, under "General".
5. Right-click on the PROFINET IO system and select the "Properties" command.

6. Assign the name "Plant_section1" to the IO system and select the check box "Use name as extension for PROFINET device names".

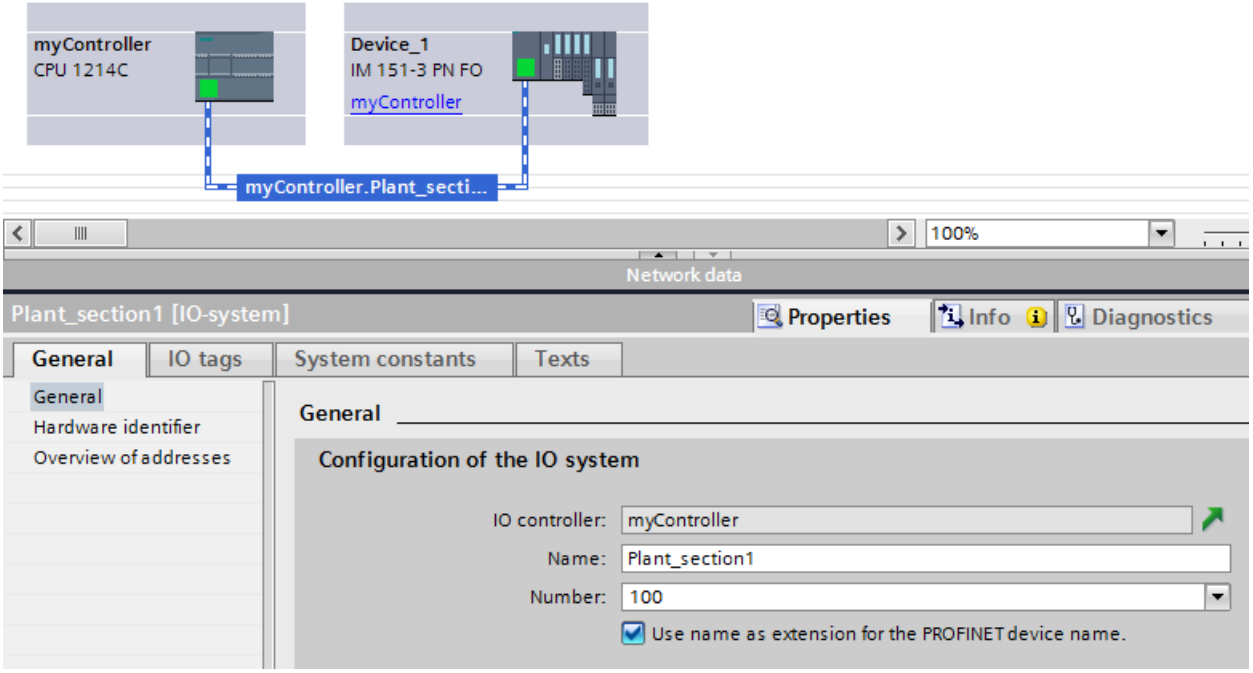

7. You can find the automatically generated PROFINET device names at the selected device in the Inspector window, at "PROFINET interface".

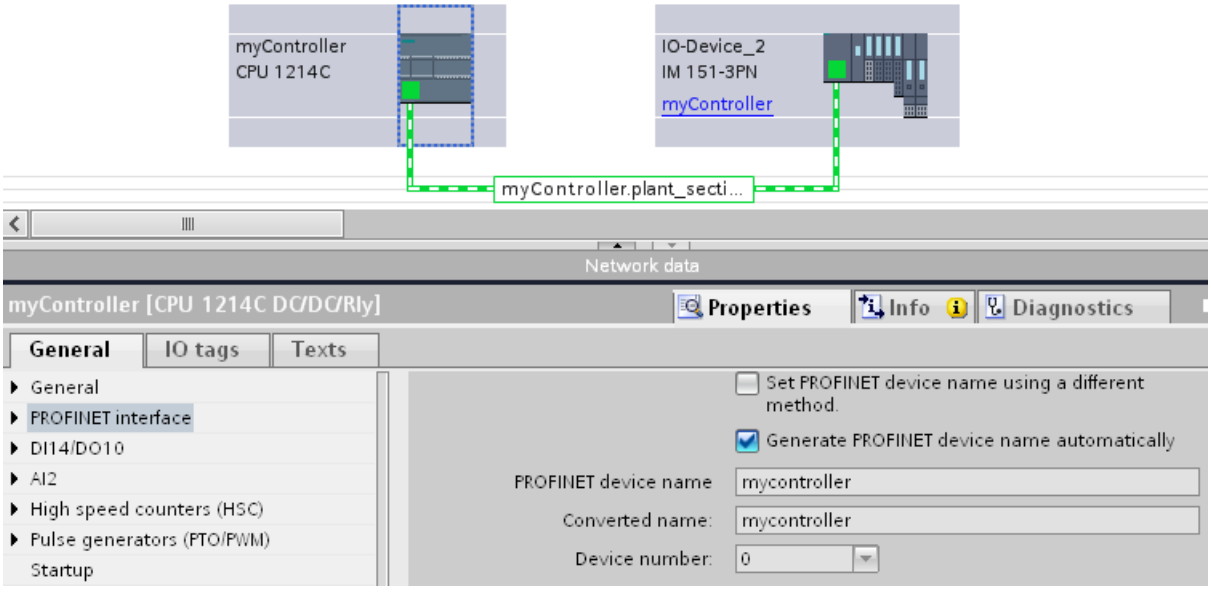

The PROFINET device name corresponds to the name of the module (with the name of the IO system as extension) with the difference that only lower case text is used. Background: No distinction is made between upper and lower case ("case insensitive") for the storing of the name.

If you want to specify the device name independently of the module name, you have to deactivate the "Generate PROFINET device name automatically" option. The PROFINET device name can be edited in this case.
The converted name is displayed below. This is the name that is automatically generated from the PROFINET device name and satisfies the DNS conventions. If you work with STEP 7, you do not require this name. This name is displayed here as a check and corresponds to the name that is stored in the device. If you work with other tools that are able to record the data exchange and read the actual device names, then you find the converted names.

Other special features

For PROFINET devices with multiple PROFINET interfaces, the name of the interface is attached to the name of the module, separated by a dot.

Example:

- Name of the module: myController
- Name of the interface: Interface_1
- PROFINET device name: mycontroller.interface_1

Assign device name via communication table

Introduction

You can assign the device names of PROFINET IO devices configured offline to the devices online. You can do this in the table area of the network view in the table "I/O communication". You can also assign the device names to several devices at the same time.

"Online assignment" tab

In the I/O communication table, you will find the tabs "Offline configuration" and "Online assignment". In the "Online assignment" tab, you can assign the PROFINET device names that were assigned offline to the corresponding IO devices online. To do this, use the buttons "Check devices" and "Assign now".

The objects displayed in the table of the "Online assignment" tab depend on the setting of the filter function. If only selected objects should be displayed, only objects of the corresponding context are displayed depending on the selection in the network view.

- PROFINET subnet: All connected devices and their PROFINET interfaces
- IO system All devices involved and their PROFINET interfaces
- Sync domain: All devices involved and their PROFINET interfaces
- Devices: The device and any existing PROFINET interfaces
- Other subnets or interfaces such as MPI or PROFIBUS are not displayed

If the display is set for all devices using the filter function, all devices are displayed that have a PROFINET interface, regardless of whether they are connected via a PROFINET subnet or are part of an IO system. Devices without a PROFINET interface, for example only with a DP or MPI interface, are not displayed.

General procedure

To assign PROFINET device names, you must first detect the IO devices available online. With this procedure, it matters whether the MAC addresses are known or unknown. This results in a general procedure in two steps:

1. Detecting the IO devices available online
2. Assigning configured PROFINET device names to the IO devices available online

Requirements

- You are in the network view.
- There is an online connection to the devices.

Procedure (step 1)

To detect IO devices available online from the I/O communication table, follow these steps:

1. Optional: Entered known MAC addresses in the "MAC address" column. After every valid entry, the check box under "Assign device" is selected for the relevant row.

   Note

   You can enter, insert or import the MAC address in different formats. The correct format is automatically entered in the cell. The following entries are supported and then converted to the required format:
   - "08:00:06:BA:1F:20"
   - "08 00 06 BA 1F 20"
   - "080006BA1F20"

   The formats used in the example are automatically converted to "08-00-06-BA-1F-20".

2. Click "Check devices" to start the check of the IO devices available online.
3. Set the PG/PC interface in the dialog window and click "Start".

Intermediate result

After the check, the result is displayed for every device in the table. Online data found is automatically entered in the table and the check box "Assign device" is set to "checked" in the rows in which a MAC address was entered or found online. The result of the check is shown as an icon in the "Status" column.

Status	Meaning
	Agreement: Matching device and compatible type. The PROFINET device name and the device type of the device found online matches the device in the project. No action is required.
	Ready for assignment: The device type of the device found online matches the device in the project. With a known MAC address the device is ready for the assignment of the PROFINET device name.
	Different device type: A matching device was found online, but the device type does not match the device type in the project. Please select a device with a matching device type online.

Status	Meaning
	Different device: No device could be found online that uniquely matches the device in the project. Please select a matching device manually.
	Device cannot be accessed: With a known MAC address the device cannot be accessed online. Please check the connection to the device.

Note

The icon "Ready for assignment" appears when a MAC address exists and matching device data was found, but no PROFINET device name was found online.

You can update the data of the detected devices again via their MAC addresses at any time. To do this, you specify the MAC address and the status of the device is displayed immediately without having to re-detect the device.

Procedure (step 2)

All PROFINET device names configured offline will be assigned to the devices available online in a bulk operation.

1. Click the "Assign now" button.

   Note

   The bulk operation cannot be reversed. A message to this effect appears in a dialog window.

2. Click "Start" in the dialog window to start the assignment of the PROFINET device names.

Result

The PROFINET device names configured offline will be assigned to the devices available online. This relates to devices in whose row the check box under "Assign device" is selected, that have a MAC address and have the status "Ready for assignment".

Importing and exporting data

Using the import and export button, you can import or export the data of the I/O communication table for the online assignment:

- When you export, the currently displayed data of the table is exported to a CSV file. Using the filter function of the table, you can select which data will be exported.
- When you import, the data of the CSV file is written to the table. If there are conflicts with values already existing in the table, you can decide whether the data should be overwritten or whether the import needs to be stopped.

Assign device name via memory card

Introduction

You can configure the device names of PROFINET IO devices offline. To do this, store a configured device name on a memory card and then insert the card into the appropriate IO device.

If an IO device has to be completely replaced due to a device defect, the IO controller automatically reconfigures the new device. Using the memory card, a device can be replaced without a programming device.

Requirements

- The programming device has a card reader for memory cards.
- The IO device must support the assignment of the device name via memory card.
- The station and its PROFINET IO system is configured.

Procedure

To store a device name on a memory card, follow these steps:

1. Insert the memory card into the card reader.
2. Select the IO device whose device name is to be assigned by the memory card.
3. Select the "Card reader > Save Device Name to Memory Card" command in the "Project" menu.
   If the memory card is not empty, a message will be issued informing you of this and you will have the option to delete the card.

Retentivity of IP address parameters and device names

The retentivity of IP address parameters (IP address, subnet mask, router setting) and device name depends on how the address is assigned.

The non-retentive, temporary assignment means:

- IP address parameters and device name remain valid for the following time period:
  - Until the next POWER OFF
  - Until the next memory reset
  - Until termination of the online connection (for example, after loading the program)
    After POWER OFF / POWER ON or a memory reset, the CPU can only be accessed via the MAC address.

If the IP address parameters are not retentive, communication based on the IP protocol can no longer take place after the above described events (for example, after POWER OFF/ POWER ON).

The assignment of a temporary IP address also deletes retentively saved IP address parameters.

Assigning IP address parameters and device name non-retentively

IP address parameters and device name are not retentive in the following cases:

- A temporary IP address that is not retentive is implicitly assigned with the "Accessible devices" function, if the device (e.g. CPU) does not yet have an IP address.
- The device is a "normal" IO controller (i.e., not an I-device), and it is specified in the user program ("IP_Conf" instruction) that the IP address parameters/device name are not to be retentive.

Assigning IP address parameters and device name retentively

IP address parameters and device name are retentive in the following cases:

- In the properties of the PROFINET interface, it is specified that the IP address parameters are set in the project ("Set IP address in the project" option).
- In the properties of the PROFINET interface, it is specified that the IP address is to be set on the device.
  - Once the configuration is loaded, the IP address parameters and the device name are assigned via STEP 7 or a setup tool such as PST (STEP 7: online and diagnostic function "Assign IP address"). The assigned IP address parameters are retentive.
  - The device is a "normal" IO controller (i.e., not an I-device), and it is specified in the user program ("IP_Conf" instruction) that the IP address parameters/device name are to be retentive.

Special features of the I-device

In the properties of the PROFINET interface of the I-device, it is specified that the IP address parameters are to be set on the device. The IP address parameters for the I-device are assigned by the higher-level IO controller.

- If prioritized startup is set, the IP address parameters are retentive.
- If **no** prioritized startup is set, the IP address parameters are not retentive.

Recommendation

If possible, use the "Set IP address in project" option and specify an appropriate IP address. In this case, the IP address is assigned retentively.

Resetting of retentive IP address parameters and device names

Retentive IP address parameters and device names are reset via the online and diagnostic function "Reset to factory settings".

Note

Consequences of reassignment of IP address parameters on top of existing IP parameters

- The temporary assignment of IP address parameters/device names results in a reset of any retentively saved IP address parameters/device names.
- With a fixed assignment of IP address parameters/device names, previously retentively-saved parameters are replaced by the newly assigned parameters.

Note

Reuse of devices

Execute the "Reset to factory settings" before you install a device with retentive IP address parameters and device name in another subnet or system or before you place it in storage.

Specifying the router for a PROFINET IO device

Introduction

You always require a router (also referred to as a "Standard Gateway") when the PROFINET device has to communicate with a node whose IP addresses lie outside the own IP subnet. If the PROFINET device sends an IP packet to an IP address outside its own IP subnet, the IP packet first goes to the configured router. The router in turn checks the IP address. If this lies outside its own subnet, the router passes the IP packet on to the next router. The IP packet is routed to the next router until it has reached the target address.

Like all S7-1500 CPUs, S7-1500 CPUs with several PROFINET interfaces provide the possibility to configure the IP address of a router. However, there is the restriction that you can only enter the IP address of a router at a PROFINET interface.

You cannot include an IP address of a router on the other PROFINET interfaces of the CPU. IO devices that are connected to this PROFINET interface adopt this setting. Up to and including STEP 7 V14 SP1, these IO devices did not have any possibility to reach devices in a different IP subnet.

As of STEP 7 V15, you have the possibility to assign the address of a router for an IO device independent of the setting of the IO controller. You can now, for example, set a router address at the IO device in the following cases as well:

- You have not set an IP address of a router for the interface of the associated IO controller.
- You have already set a router address for a different interface in the CPU.

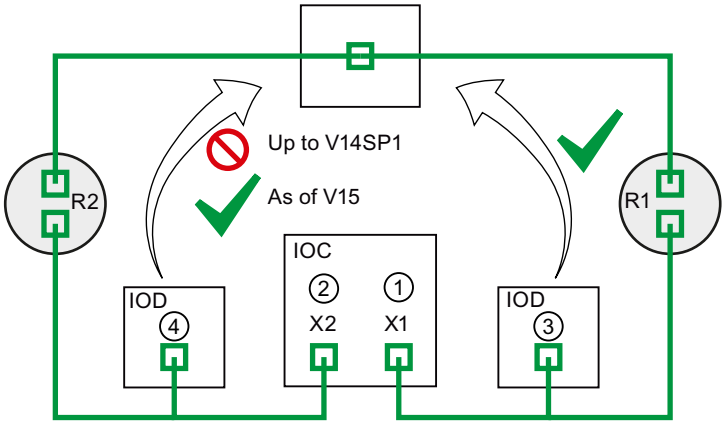

① Router R1 configured at PROFINET X1

② If a router is configured at X1, you cannot configure a router at X2.

③ Because a router is configured at X1, the IO device adopts the IP address of the router R1. The IO device can be reached from a different IP subnet.

④ As of STEP 7 V15, you set the IP address of the router R2 at the IO device irrespective of the setting at the interface X2. The IO device can be reached from a different IP subnet.

Figure 1-90 Specifying the router for an IO device

Further information about the "User router" setting

You have the possibility to configure the use of a router including IP address of the router in the "IP protocol" section of the settings for the PROFINET interface (Ethernet addresses).

Rules

Observe the following rules if you want to configure a router for the PROFINET interface of an IO controller:

- A PROFINET IO device supports exactly one router, irrespective of the number of interfaces.
- You can configure a router for exactly one PROFINET interface. All IO devices that are assigned to the PROFINET interface adopt the configured router from the IO controller.
- You cannot configure a router for the further PROFINET interfaces of the CPU. The further PROFINET interfaces take on the IP address "0.0.0.0" as the router and pass it on to their IO devices.

As of STEP 7 V15, you can configure the use of a router for an IO device. This allows the IO device to communicate with a node outside its own IP subnet, irrespective of the setting of the PROFINET interface of the IO controller.

Configuration example: Configuring a router for an IO device

The following example shows a configuration in which you configure a router at the IO device so that the IO device reaches IP addresses in the higher-level network.

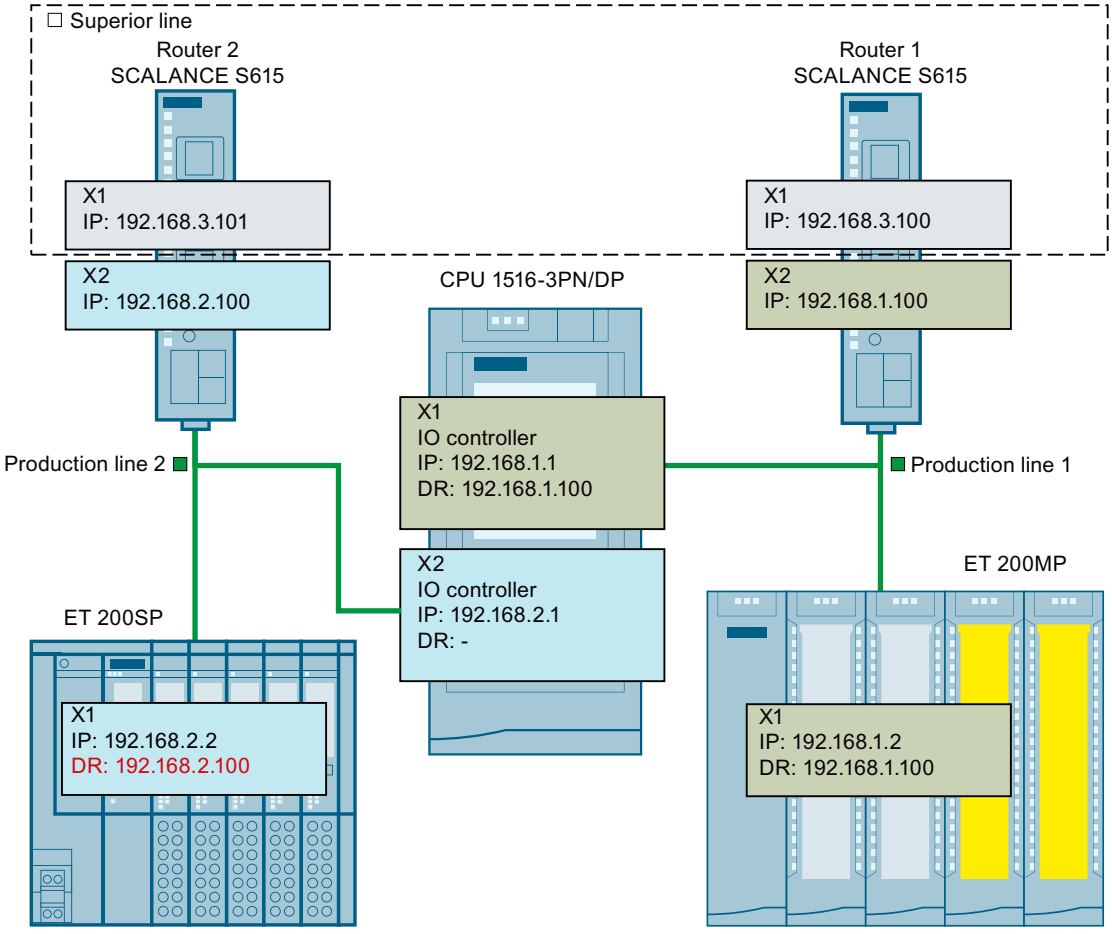

Figure 1-91 Configuration example: Configuring a router for an IO device

You have a CPU 1516-3PN/DP. The two PROFINET interfaces X1 and X2 of the CPU work in the "IO controller" operating mode. The PROFINET interface X1 is connected with the subnet "Production line 1". PROFINET interface X2 is connected with the subnet "Production line 2". The two subnets "Production line 1" and "Production line 2" are each connected via a router with the higher-level network "Superior line".

For PROFINET X1, you configure the router "Router 1" with the IP address 192.168.1.100.

The IO device (ET 200SP) in the "Production line 1" subnet adopts the router from the IO controller.

You cannot configure a router for the PROFINET interface X2 because you have already configured a router for the PROFINET interface X1 of the CPU.

No router is transferred by the PROFINET interface X2 to the IO device in the subnet "Production line 2".

In order for the IO device in the subnet "Production line 2" to reach nodes in the higher-level "Superior line", configure the router "Router 2" with the IP address 192.168.2.100 for the IO device.

Configuring the router for the IO controller

Requirements: You use the "Set IP address in the project" option for the PROFINET interface.

Follow these steps to configure a router for the IO controller in STEP 7:

1. In the network view of STEP 7, select the PROFINET interface of the IO controller.
2. In the Inspector window, navigate to "Properties" > "General" > "Ethernet addresses".
3. Select the "Use router" check box in the "IP protocol" field.
4. Enter the IP address of the router at "Router address".

Configuring a router for an IO device

Requirements:

- STEP 7 as of V15
- CPU 1500 as of firmware version V2.5
- IO device is assigned to the PROFINET interface of an IO controller. The PROFINET interface of the IO controller uses the "Set IP address in the project" option.

Follow these steps to configure a router for the IO device in STEP 7:

1. In the network view of STEP 7, select the PROFINET interface of the IO device.
2. In the Inspector window, navigate to "Properties" > "General" > "Ethernet addresses".
3. Clear the "Synchronize router settings with IO controller" check box.
4. Select the "Use router" check box.
5. Enter the IP address of the router at "Router address".

Creating a PROFINET IO system

Introduction

To create a PROFINET IO system, you need to have a PROFINET IO controller and at least one PROFINET IO device. As soon as you connect an IO controller to an IO device via their PROFINET interfaces, a controller-device link is established.

IO controller and IO device

As IO controller, you can use the following devices with a PROFINET interface:

- CPU with a permanently integrated or plug-in PROFINET interface
- CP in conjunction with a CPU
- Interface module assigned to a CPU/FM
- Interface module with a PROFINET interface

As IO device, you can use head modules of the distributed I/O with PROFINET interface or CPUs as intelligent IO devices.

With some devices, you can switch between modes (such as "IO controller" and "IO device") under "Mode" in the Inspector window of the PROFINET interface. Select the option button with the required mode.

Requirements

- You are in the network view.
- The hardware catalog is open.

Procedure

To create a PROFINET IO system, for example with a CPU 1217C , follow these steps:

1. Select a CPU 1217C as a potential IO controller from the hardware catalog.
2. Drag the CPU onto the free area within the network view.
3. Right-click on the PROFINET interface of the CPU.
4. Select "Create IO system" from the shortcut menu.

A PROFINET IO system with a CPU 1217C as IO controller will be created as the only node.

If you connect the PROFINET interface of an IO device to the PROFINET interface of the IO controller, the IO slave will be added automatically to the IO system. If there is not yet a subnet between the IO controller and IO device, a new subnet is created between the IO controller and IO device.

To include a CPU 1217C as an intelligent IO device in the DP master system, for example, follow the steps below:

1. Click on the PROFINET interface of the IO controller or IO device.
2. Hold down the mouse button and drag a connection from the selected PROFINET interface to the PROFINET interface of the desired communications partner.

or

1. Click on the hyperlink of the IO device CPU 1217C.
2. Select the required IO controller from the displayed list of possible IO controllers.

The intelligent IO device CPU 1217C is included in the PROFINET IO system with the CPU 1217C as IO controller.

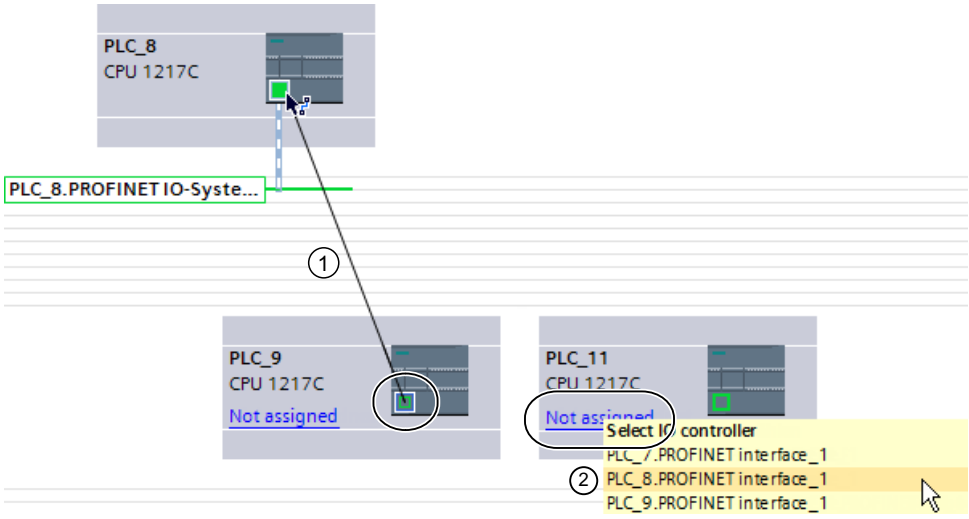

① Drag between the PROFINET interfaces of IO device to IO controller.

② Click on the link of an unassigned IO device, and a selection of possible IO controller opens.

If required, adapt the properties of the Ethernet subnet or the IO controller (for example, IP address) under "Properties" in the inspector window.

Note

If there is only one IO controller on a subnet, you only have to connect the interface of the IO device to the subnet of this IO controller with drag-and-drop. The IO device is then connected to the subnet of the IO controller immediately, and the IO device is assigned to this IO controller.

Display of the IO controller in the IO device

When an IO device is connected to an IO controller, the name of the IO controller is displayed on the IO device as a hyperlink. Click the hyperlink to select the assigned IO controller.

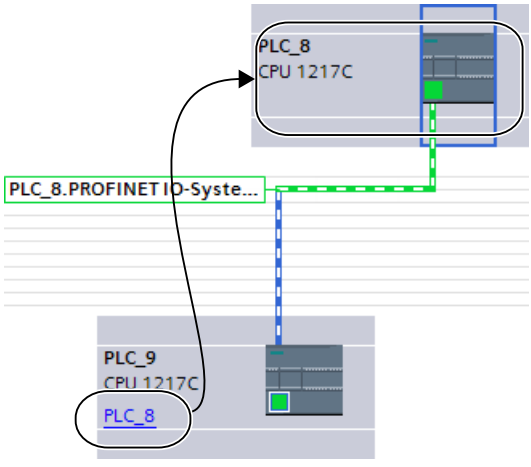

Highlighting the PROFINET IO system

When you have created a new PROFINET IO system, the PROFINET IO system will be highlighted. This enables you to identify quickly which devices belong to the PROFINET IO system. You can also highlight a PROFINET IO system yourself by moving the mouse pointer over a subnet. This will result in the names of the available PROFINET IO systems being displayed. If you click on one of the displayed PROFINET IO systems with the mouse, the corresponding PROFINET IO system is highlighted.

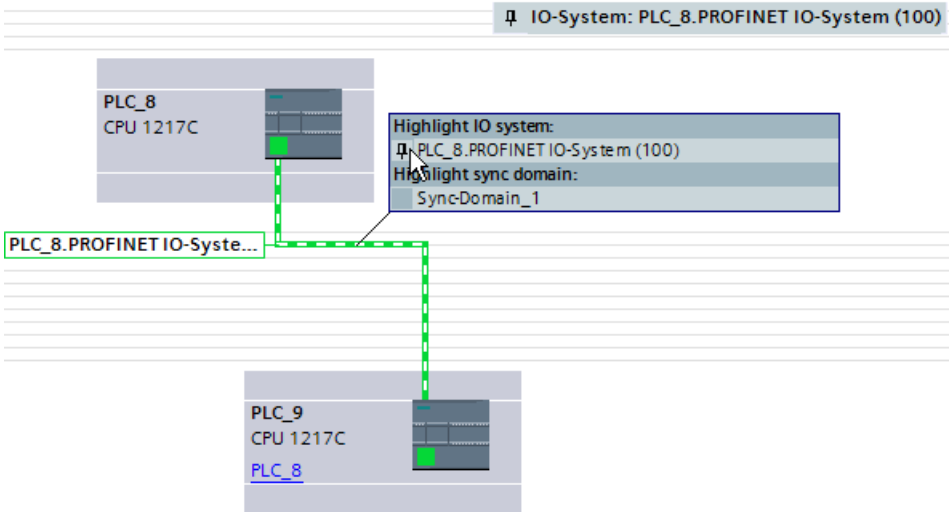

There are various ways of removing the highlighting from a PROFINET IO system:

- Highlight a different PROFINET IO system.
- Click on the pin with the name of the PROFINET IO system in the top right-hand corner of the network view.

Handling PROFINET IO systems

Using shortcut menu commands, you can delete PROFINET IO systems, create new ones or even connect the interface to another subnet from within the network view.

An existing PROFINET configuration can thereby be corrected in the network view.

Create new PROFINET IO system for IO controller

To create a new PROFINET IO for an IO controller, proceed as follows:

1. Make sure that no IO system is already assigned to the IO controller. If an IO system is already assigned to the IO controller, the "Assign IO system" shortcut menu command is disabled.
2. Select the PROFINET interface and then select the "Assign IO system" shortcut menu command.

A new PROFINET IO system is created at the IO controller and you can assign IO devices to this IO system.

Disconnecting PROFINET IO devices from PROFINET IO system

To disconnect an already networked PROFINET IO device from its PROFINET IO system, follow these steps:

1. Click on the PROFINET interface of an IO device.

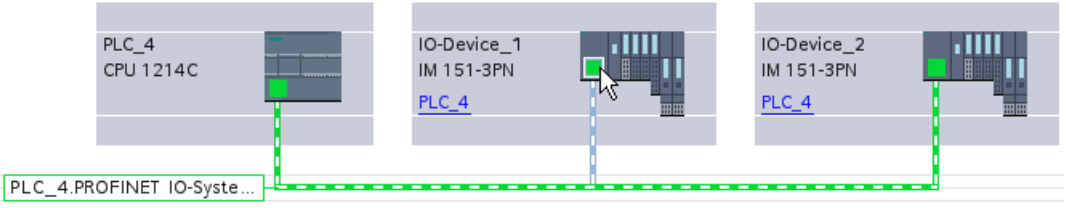

2. Select the "Disconnect from IO system" shortcut menu command.
   The IO device that was assigned to this IO system is then no longer assigned to it.

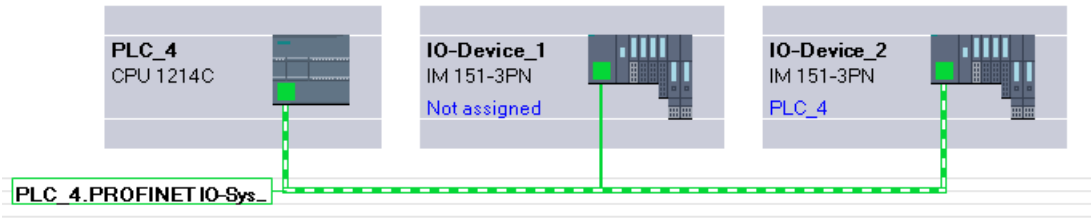

You can create a new IO systems and can assign each of the non-assigned IO devices to an IO controller.

Assign PROFINET IO devices to other IO controllers

Existing PROFINET IO systems can be easily reconfigured in the network view by selecting the IO device or its interface and the required command from the shortcut menu:

When selecting the interface:

1. Select the interface of an IO device and then select the shortcut menu. You have the following options here:
   - Assign a new subnet to the IO device or disconnect it from the existing subnet
   - Assign the IO device to a new IO controller
   - Assign a new IO system to the IO device or disconnect it from the existing subnet
2. To assign another IO controller to the IO device, select the "Assign to new IO controller" shortcut menu command.
   If there is no connection, a subnet is automatically created and the IO device is assigned to the IO system of the new IO controller.

When selecting the entire device:

1. Select the IO device and select the shortcut menu. You have the following options here:
   - Assign the IO device to a new IO controller
   - Disconnect the IO device from the existing subnet
2. To assign the IO device to another IO controller, select the "Assign new DP master/IO controller" shortcut menu command.
   If there is no connection, a subnet is automatically created and the IO device is assigned to the IO system of the new IO controller.

Configuring an IO device through hardware detection

Introduction

As of STEP 7 V15, you have the possibility to detect a real existing IO device and to import it into your project.

You find the IO device in STEP 7 through the "Hardware detection" function. A detected device can be imported into your project. STEP 7 inserts the IO device with all the modules and submodules.

Note that ET 200AL devices are not detected by the hardware detection.

No hardware detection of IE/PB links

Hardware detection only detects devices that have a role as IO devices. Devices and other components without this role, such as IO controllers, I-devices or devices with assignments to multiple IO controllers (multiple assignment), are not taken into account in the hardware detection. IE/PB links are multi-assignment devices and are therefore not displayed during hardware detection.

Device names of PN/PN couplers

The names of the PN/PN couplers must be given special consideration for hardware detection of PN/PN couplers and their subsequent integration in the project configuration. Unlike other IO devices, these PROFINET devices have an additional attribute for automatic generation of device names. When adding a PN/PN coupler to a configuration from hardware detection, the device name reacts the same way, that is, as if the PN/PN coupler had been added to the configuration from the hardware catalog.

The device name of the PN/PN coupler is therefore automatically generated after the hardware detection has been added and may need to be adapted manually afterwards.

Prerequisites

- STEP 7 (TIA Portal) as of V15
- New or existing project must be open.
- It must be possible to technically access IO devices via IP

Procedure

To detect one or more existing IO devices in STEP 7 and add them to the project, follow these steps:

1. In STEP 7, navigate to "Online" > "Hardware detection".
2. Click "PROFINET devices from network...".
   STEP 7 opens the "Hardware detection of PROFINET devices" window.
3. Select the interface of your programming device at "PG/PC interface".
4. Click "Start search".
   STEP 7 begins with the hardware detection. When the hardware detection is completed, STEP 7 displays all the detected IO devices.
5. Select the IO devices that you want to add to the project by clicking the corresponding check box in front of the IO device.
6. Click "Add devices".
   After a brief moment, a window is opens to report about the success or failure of the hardware detection.

Result of the hardware detection

If the hardware detection is successful, STEP 7 inserts the IO device with all the modules and submodules into the project.

An IO device configured via hardware detection responds as follows:

- Modules configured through the "Hardware detection" are configured as if they have been inserted from the catalog.
- MAC address: STEP 7 imports the MAC address of the detected IO device into the project.
- IP settings:
  - If the detected IO device already has an IP address, STEP 7 imports the IP address into the project.
  - If the detected IO device does not have an IP address, STEP 7 automatically assigns an IP address in the project.

  PROFINET device name:
  - If the detected IO device already has a PROFINET device name, STEP 7 imports the PROFINET device name into the project.
  - If the detected IO device does not have a PROFINET device name, STEP 7 automatically assigns a PROFINET device name in the project.
- IO devices configured through "Hardware detection" have neither an IP subnet nor an IO controller assigned.

See also

Adding an unspecified CPU (Page 42)

Tip: Quick configuration of IO systems

If the IO system has a lot of IO devices, assign all IO devices placed by drag-and-drop operation to an IO controller on one step.

Requirements

IO controller and IO devices are placed in the network view.

Assign IO devices to an IO system

To do this, follow these steps:

1. Select an appropriate zoom factor so that you can see as many IO devices as possible in the network view.
2. Arrange the IO devices in not more than of two rows.
3. Select all IO interfaces (not all devices) with the mouse cursor. This only works if you begin to drag the mouse cursor outside of the first IO device and release the mouse button at the last IO device (selection with the lasso).

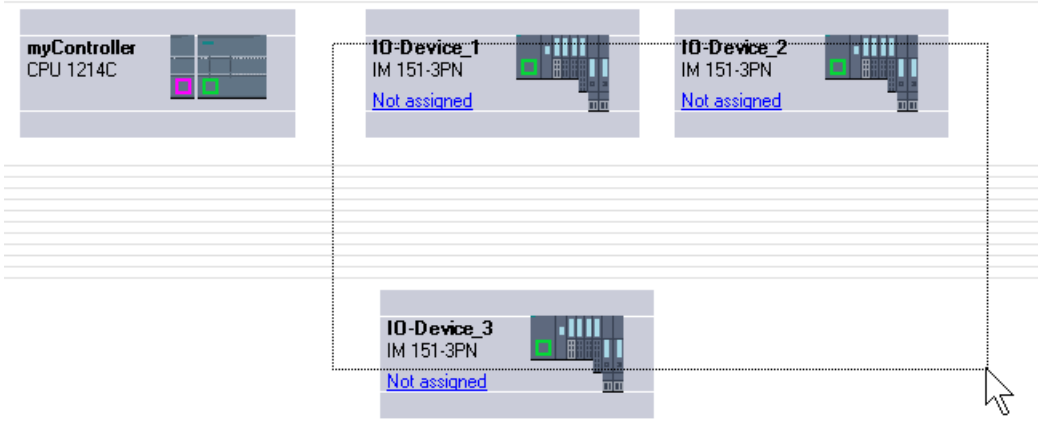

4. Select the shortcut menu "Assign new IO controller" and select the corresponding IO interface of the IO controller in the subsequent dialog.

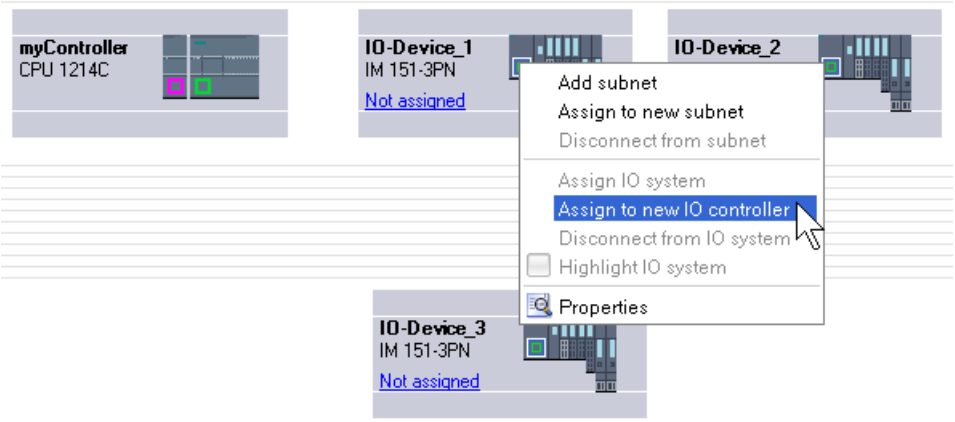

5. The IO devices are automatically networked with the IO controller and combine with it to form an IO system.

Note

When an IO system is highlighted, you can double-click on an IO device in the hardware catalog and thereby quickly add additional IO devices. Result: The IO device is automatically added to the highlighted IO system.

Interconnecting ports

If an IO device is assigned to an IO controller, this does not yet specify how the ports are connected to each other.

Although a port interconnection is not required to make use of the Ethernet/PROFINET functions, it does offer the following advantages:

- A target topology is specified with the port interconnection. Based on an online-offline comparison, it is possible to conduct a target-actual comparison with all devices that support this function.
- Only with IRT communication: If a port interconnection is configured, STEP 7 can determine the required bandwidth more precisely. As a rule, this leads to a higher performance.

Make sure that no invalid ring structures occur through the interconnection of ports.

Port interconnection is only advisable for devices that support the topology configuration.

Interconnecting ports in the Inspector window

To interconnect ports, follow these steps:

1. Select the Ethernet/PROFINET device or the Ethernet/PROFINET interface.
2. Navigate to the port property "Port interconnection".
   When the Ethernet/PROFINET interface is selected, you can find this setting in the Inspector window as follows: Properties > General > Advanced Options > Port [...] > Port Interconnection.
3. In the "Local port" section, you can find the settings at the local port. In the case of fiber-optic cable you can, for example, set the cable names here.
   In the "Partner port" section, click on the black triangle in the "Partner port" box to display and select the available partner ports.
4. If the port interconnection is a port interconnection with copper as medium and the devices support IRT communication, you can also set cable length and signal transit time.

If the Ethernet/PROFINET interface was disconnected, it is automatically networked by this action. In the properties of the subnet you can set whether this subnet should or should not be used for the networking.

Note

Interconnecting an electric with an optical port

If you want to interconnect an electrical and an optical port, you have to distinguish between RT and IRT communication:

- With RT communication, it is not necessary to configure a media converter.
- With IRT communication, you have to make the interconnection via a media converter.

Information on monitoring the partner port

After you have interconnected the two ports, you receive information on the monitoring of the partner port in a text field. The following displays are possible:

- Monitoring of the partner port is not possible.
- Partner port is being monitored.

It is not possible to monitor the partner port if you, for example, select a deactivated port as partner port. In this case it is not possible to monitor the target topology and the signal propagation time. A device replacement is then only possible with a Micro Memory Card.

See also

Overview (Page 295)

Overview of the CPU properties (Page 935)

Setting the send clock

Requirements to change the send clock at the PROFINET device

The PROFINET device must be an IO controller as sync master or IRT (Isochronous Realtime) must not be configured in the IO system.

If the PROFINET device is not an IO controller as sync master, the following requirements must be met for the setting of the send clock:

- No device must be configured at the IO system as a sync slave or sync master.
- All devices at the IO system must be unsynchronized.

When IRT is configured with an IO controller as sync master, the send clock can be set at the device or in the sync domain.

Procedure

To set the send clock on the PROFINET device, follow these steps:

1. Select the PROFINET IO controller in the device or network view.
2. Change the value for the shortest possible update interval in the properties of the PROFINET interface under "PROFINET Interface > Advanced options > Real-time settings > IO communication > Send clock".

The send clock is valid for all PROFINET devices at the IO system. If the synchronization role is set to a value other than "Unsynchronized", you can also set the send clock in the sync domain, in other words, centrally at the PROFINET IO system.

1. Select the IO system using the PROFINET subnet.
2. Change the value for the send clock in the properties of the sync domain under "Domain management > Sync domains > Sync_domain_1 > Send clock".

Setting the update time

Update time

An IO device / IO controller in the PROFINET IO system is supplied with new data from the IO controller / IO device within this time period. The update time can be separately configured for each IO device and determines the time interval in which data is transmitted from the IO controller to the IO device (outputs) as well as data from the IO device to the IO controller (inputs).

STEP 7 calculates the update time automatically in the default setting for each IO device of the PROFINET IO system, taking into account the volume of data to be exchanged as well as the set send clock.

Setting the update time

If you do not want to have the update time calculated automatically, you can change the setting.

To change the update time in the Inspector window, proceed as follows:

1. Select the PROFINET interface of the IO device in the network or device view.
2. Specify the update time in the interface properties under "Advanced options > Real time settings > IO cycle". You can specify the update time in two ways:
   – Automatic calculation of the optimal update time.
   – Update time you can set yourself by selecting different values from a drop-down list.
3. If you want to keep the relationship between the send clock and the update time constant, enable the "Adapt update time when send clock changes" option.
   This option ensures that the update time is not set to less than the send clock.

You can also carry out this setting directly in the I/O communication table via the drop-down list in the "Update time mode" column.

Manual setting of the send clock may result in errors if the available bandwidth is not adequate or when other limits/configuration limits are exceeded (for example, too many nodes are configured).

Note

When you change the update time mode from "Can be set" to "Automatic", a manually entered update time is overwritten by the automatically calculated value.

No update time can be calculated

STEP 7 determines the sequence of the cyclic data exchange based on the configuration information (IO controller properties, IO device properties, number and type of the IO devices, consistency of the cyclic user data...). The cyclic data is packed in frames and sent/received successively at calculated time intervals.

The maximum number/size of the frames and the maximum number of intervals available must be sufficient to "accommodate" all the data. The resulting send/receive interval must also be supported by every PROFINET device.

If the limits relating to the amount of cyclic user data/number of frames or relating to the intervals available are exceeded, STEP 7 cannot calculate an update time.

If there is no common basis for the send/receive interval, calculating the update time is also not possible.

If there is a reason preventing the calculation of the update time, STEP 7 reports the cause when compiling the hardware configuration.

How to eliminate the problem:

- Reduce the number of IO devices
- Reduce the number of modules within the IO devices
- If you are using an IE/PB link: Reduce the number of DP slaves downstream from the IE/PB link
- Use a more powerful IO controller or IE/PB link

- Increase the send clock
  - With RT: in the properties of the IO controller
  - With IRT: in the properties of the sync domain
- Check the device properties of the IO devices ('MinDeviceInterval' and the possible scan rates) due to the common basis for the send/receive interval. Replace unsuitable IO devices. This device properties are stored in the GSD file of the IO device.
- With IRT configuration:
  - Check whether the ports of the sync master and sync slaves are interconnected.
  - Check the order of the IO devices: There can be **no** unsynchronized device connected between the sync master and sync slave
    (Example of bad configuration: Sync master --- unsynchronized device --- sync slave).
  - Check whether you have configured more than one sync master.
  - Check the bandwidth remaining for RT data. The bandwidth for RT data available for the transfer can be restricted by IRT communication on the same Ethernet subnet.
- When using I-devices:
  It may not be possible to use the set send clock together with the existing I-device configuration.
  - Configure the I-device without lower-level IO devices and activate the setting "Parameter assignment of PN interface by higher-level IO controller".
  - Change the send clock of the IO controller to an even value (... 0.250, 0.500, 1.000, ...).

Identification of the IO devices involved:

You can identify the IO devices involved for which no update time can be calculated in the "I/O communication" table of the PROFINET IO system in the network view. No entry is made in the "Update time" column for the IO device involved (entry "-").

Setting the watchdog time

Watchdog time

You can configure the watchdog time for PROFINET IO devices.

If the IO device is not supplied with input or output data (IO data) by the IO controller within the watchdog time, it switches to the safe state.

Do not enter the watchdog time directly, but as "Accepted number of update cycles when IO data is missing". This makes setting easier because the update time can be shorter or longer, depending on the power of the IO device or the setting.

The resulting watchdog time is automatically calculated from the "Accepted number of update cycles when IO data is missing".

Configuring the watchdog time

To specify the watchdog time, follow these steps:

1. Select the PROFINET interface of the IO device in the network or device view.
2. In the properties of the interface, navigate to "Advanced options > Realtime settings > IO cycle".
3. Select the required number of cycles from the drop-down list "Trigger watchdog after # cycles with missing IO data".

The watchdog time is subsequently calculated automatically based on the preset factor. It must not be more than 1.92 seconds.

Note

The default setting should only be changed in exceptional cases, for example, during the commissioning phase.

Calculated bandwidth for cyclic IO data

Calculated bandwidth for cyclic IO data

Adherence to the maximum available bandwidth for cyclic IO data is monitored by the system. The maximum bandwidth depends on the send clock cycle. If the send clock cycle is greater than or equal to 1 ms, the maximum bandwidth is 0.5 ms. If the send clock cycle is shorter, the maximum available bandwidth is also reduced.

The bandwidth actually required for cyclic IO data is determined by the system based on the number of configured IO devices and IO modules. Furthermore, the required bandwidth depends on the update time that is used.

In general, the calculated bandwidth increases in the following cases:

- There is a greater number of IO devices
- There is a greater number of IO modules
- The update times are shorter.

Maximum bandwidth for cyclic IO data depending on the send clock

The following table shows how the maximum available bandwidth for cyclic IO data reacts based on the send clock:

Send clock cycle	Maximum bandwidth for cyclic IO data
250 µs – 468.75 µs	<< 125 µs
500 µs – 968.75 µs	= send clock / 2
1 – 4 ms	= 500 µs

Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

By default, the settings for ports of a PROFINET interface are specified in such a way that problem-free data exchange is ensured in normal situations.

If necessary, you can change the settings. The setting options are described below.

Possible settings for transmission rate/duplex

Depending on the selected device, you can make the following settings for "Transmission rate/duplex":

- Automatic setting
  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the partner port. The "Enable autonegotiation" option is automatically selected by default.
- TP/ITP at x Mbps full duplex (half duplex)
  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:
  - Autonegotiation enabled
    You can use both cross cable and patch cable. The port is monitored with this setting.
  - Autonegotiation disabled
    Make sure that you use the correct cables (patch or cross cables). The port is also monitored with this setting.
- Deactivated
  Depending on the device type, the drop-down list box can contain the "Deactivated" option. With this option, you can prevent access via an unused port for security reasons, for example. With this setting, diagnostic events are not generated.

"Monitor" option

With the "Monitor" option, you enable/disable port diagnostics. You can only enable monitoring if the device supports this function.

Below is a list of what is monitored during operation by the device when monitoring is enabled - regardless of whether you have interconnected the port or not:

- The link status is monitored; in other words, diagnostics is generated when the link is down (e.g. with a wire break or a removed connector). With fiber-optic ports (FO), the link power margin is monitored so that a staged maintenance diagnostics is obtained.
- The media settings are monitored; in other words, a diagnostics message is generated when the local port setting (transmission rate/duplex) during operation does not match the setting of the partner port.
- If the port is interconnected: The cable length/signal delay time is monitored; in other words, a diagnostics message is generated if the locally detected cable length/signal delay time does not match the cable length/signal delay time detected by the partner.

Disabling monitoring

Monitoring does **not** make sense in the following cases:

- The option "Alternative partners" is enabled at the local port, for example, for a docking port of a docking station (tool changer). In this case, a link status change is a normal operational situation.
- The partner port belongs to a non-PROFINET device. Only ports of PROFINET devices transfer their port settings such as the transmission rate, duplex setting etc.
- The local port is connected to a media converter (passive network component with different signal delay times for Ethernet frames).
- At the local port, the option "End of topology discovery" is enabled in the boundaries settings.

Monitoring of the desired topology

With the "Monitoring" option, no monitoring is linked to the adherence to the desired topology. This means that whether the configured interface and port properties such as device name, port name and cable length match the current settings is not influenced by the "Monitoring" option.

Adherence to the desired topology is checked automatically when you interconnect the local port with a partner port; the partner port must not then be disabled. The port properties indicate whether the monitoring of the desired topology is performed or is impossible (port interconnection, partner port refer, to the figure below ①).

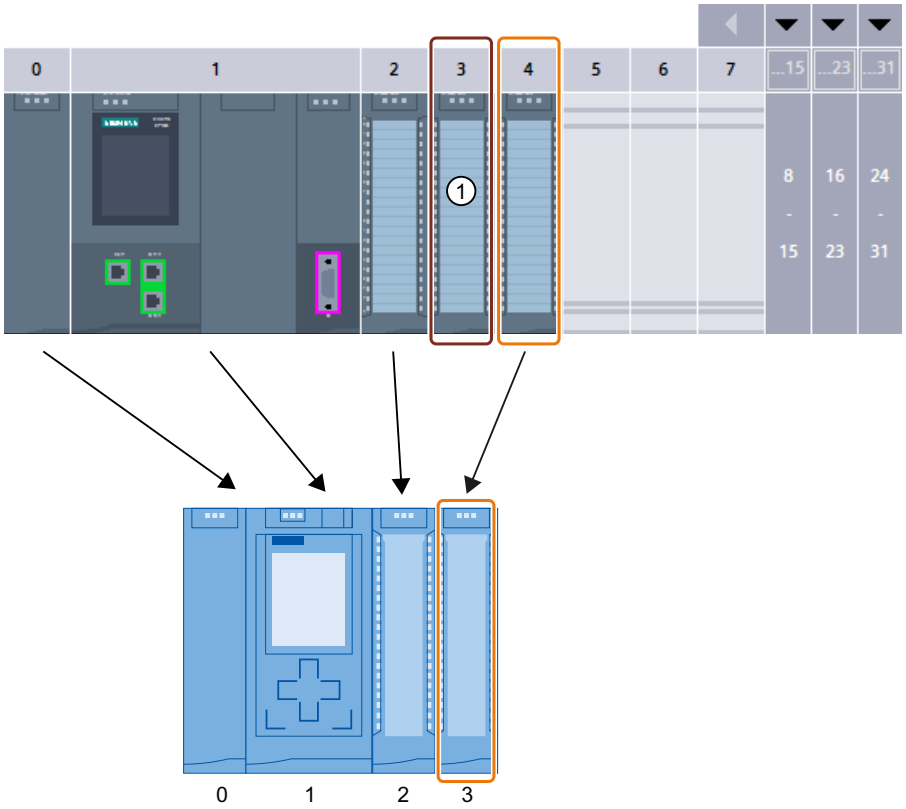

ConfDB

		Name	Data type	Start value	...	...	...	...	Comment
1		Static							
2		ConfigControl	Struct			✓	✓		
3		Block_length	USInt	9		✓	✓		4+number of slots
4		Block_ID	USInt	196		✓	✓		
5		Version	USInt	4		✓	✓		
6		Subversion	USInt	0		✓	✓		
7		slot_0	USInt	0		✓	✓		configured slot_0 / assigned "real" slot 0
8		slot_1	USInt	1		✓	✓		configured slot_1 / assigned "real" slot 1
9		slot_2	USInt	2		✓	✓		configured slot_2 / assigned "real" slot 2
10		slot_3	USInt	16#FF		✓	✓		configured slot_3 / not used ①
11		slot_4	USInt	3		✓	✓		configured slot_4 / assigned "real" slot 3

① Display of whether the device can check the actual properties of the partner port against the configured properties of the partner port. This adherence to the desired topology is checked automatically when you interconnect the local port with a partner port.

② Setting the monitoring to link down, media settings and cable length/signal delay time. At runtime, the local port compares its own settings with the settings of the partner port.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because, with this option, the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can therefore not be optimally set.

Note

When a local port is interconnected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not support the setting, an error message is generated.

Port options for using the GBIT PROFINET interface

The PROFINET interface (X3) of the CPU 1518-4 PN/DP supports a maximum transmission rate of 1000 Mbps (GBIT).

In order to achieve this transmission rate, the following requirements must be met:

- CPU Firmware Version V1.7 or higher.
- Devices on the same PROFINET subnet must also support the 1000 Mbps transmission rate.
- The network infrastructure (network cables and outlets) must be category CAT 5e or better.
- The port options of PROFINET interface X3 must be set as follows:
  - “Transmission rate/Duplex”: Automatic
  - “Autonegotiation”: Activated

See also

Wiring rules for disabled autonegotiation (Page 2874)

Boundaries at the port (Page 2875)

Wiring rules for disabled autonegotiation

Requirement

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission speed
- Autonegotiation incl. autocrossing disabled

This saves you the time required to negotiate the transmission rate during startup.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively to the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in a line using a patch cable (one-to-one wiring of both connectors). To do this, you connect port 2 (P2) of the IO device to port 1 (P1) of the next IO device. The following graphic gives an example with two IO devices.

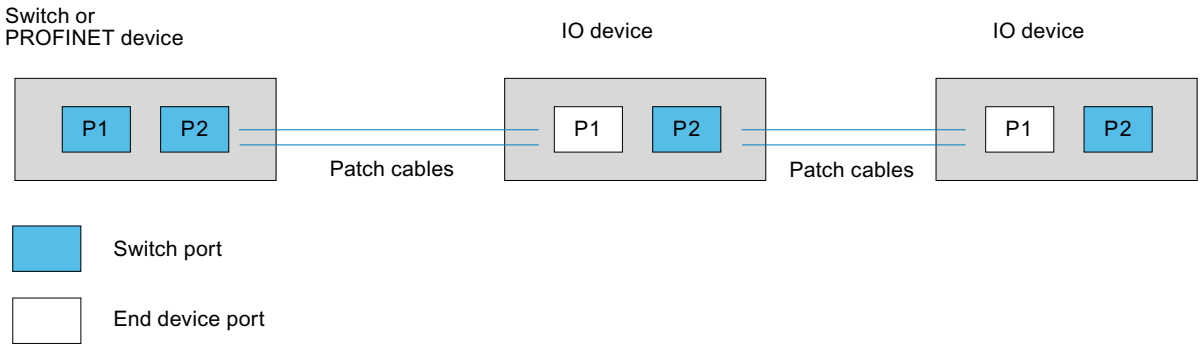

Boundaries at the port

Requirement

The respective device must support boundaries settings in order to work with boundaries. If the device for PROFINET does not support boundary settings, the corresponding parameters are disabled, such as in the CPU 1215C V3, for example.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of detection of accessible devices"
  DCP frames for detection of accessible devices are not forwarded. Devices located behind this port are no longer displayed in the project tree under "Accessible devices". Devices located behind this port can no longer be reached by the CPU.
  The device must have more than one port for this function.
- "End of topology discovery"
  LLDP frames (Link Layer Discovery Protocol) for topology detection are not forwarded.
- "End of sync domain"
  Sync frames transmitted for synchronization of devices within a sync domain are not forwarded.
  For example, if you operate a PROFINET device with more than two ports in a ring, you should prevent the sync frames from being fed into the ring by setting a sync boundary (at the ports not inside the ring).
  The device must have more than one port for this function.
  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device of the other sync domain.

Restrictions

The following restrictions must be observed:

- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been defined for the port, the following check boxes cannot be used:
  - "End of detection of accessible devices"
  - "End of topology discovery"

Enabling device replacement without exchangeable medium

Replacing an IO device without exchangeable medium

It is often necessary to replace IO devices in automation systems. The IO devices are generally assigned a device name by either inserting an exchangeable medium or via the programming device. The IO controller uses this device name to identify the IO device.

Subject to certain conditions, IO devices can also receive their device names without the insertion of an exchangeable medium (e.g., memory card) or without a programming device. For this purpose, the IO controller uses Ethernet mechanisms (LLDP protocol; Link Layer Discovery Protocol) to analyze the relationships between the individual IO devices and the IO controller. From these relationships, the IO controller detects which IO device was replaced and assigns the configured device name to it.

Requirements

- A port interconnection is already configured.
- The affected IO devices in the automation system must support device replacement without exchangeable medium (LLDP protocol).

Note

Use only new IO devices as replacement devices, or restore IO devices already configured to the delivery state prior to commissioning.

For S7-1500 CPUs with firmware version V1.5 or higher, it is not necessary to reset IO devices with an existing parameter assignment to delivery state. Requirement: The "Permit overwriting of device names of all assigned IO devices" option is selected for the IO controller ("Advanced option > Interface options" area of the properties of the PROFINET interface).

Procedure

In order to enable the replacement of an IO device without exchangeable medium, proceed as follows:

1. In the device or network view, select the PROFINET interface of the corresponding IO controller.
2. In the interface properties under "Advanced settings > Interface options", select the "Support device replacement without exchangeable medium" check box.

The option "Support device replacement without exchangeable medium" also permits automatic commissioning, which means you can commission the IO system with the IO devices without assigning their device names in advance.

IO devices without corresponding functionality

When individual IO devices in the automation system do not support device replacement without exchangeable medium, a corresponding alarm is output for the IO device. It is possible to use the functionality nevertheless. However, all of the following conditions have to be fulfilled:

- The function "Device replacement without exchangeable medium" is activated in the higher-level IO controller.
- The IO device supports the LLDP protocol.
- At least one "neighbor" of the IO device supports
  - The function "Device replacement without exchangeable medium" as an IO controller or
  - The LLDP protocol as an IO device.

When you configure the environment of the IO device so that the conditions described above are met for the "Device replacement without exchangeable medium", the usage of the function is possible even if the IO device does not support the functionality directly.

See also

Assigning the device name and IP address (Page 2842)

Components with the the device replacement without exchangeable medium function (http://support.automation.siemens.com/WW/view/en/36752540)

Connect the DP slave via the IE/PB Link to a PROFINET IO system

Requirements

- STEP 7 as of V12
- S7-1500 CPU as of firmware version 1.7
- ET 200SP CPU as of firmware version 1.7
- S7-1500 software controller
- S7-300/400 CPU

Procedure for connecting a DP slave via an IE/PB Link

To connect a DP slave to a PROFINET IO system via an IE/PB Link in STEP 7, follow these steps:

1. Drag a PROFINET CPU, e.g. 1513-1 PN, from the hardware catalog to the network view of STEP 7.
2. Drag an IE/PB Link PN IO from the hardware catalog into the network view of STEP 7. The IE/PB Link PN IO is located under Network components > Gateways > IE/PB Link PN IO.
3. Assign the IE/PB Link PN IO to the CPU.
4. Drag a PROFIBUS interface module, e.g. IM155-6 DP HF, from the hardware catalog to the network view.
5. Assign the interface module to the IE/PB Link.

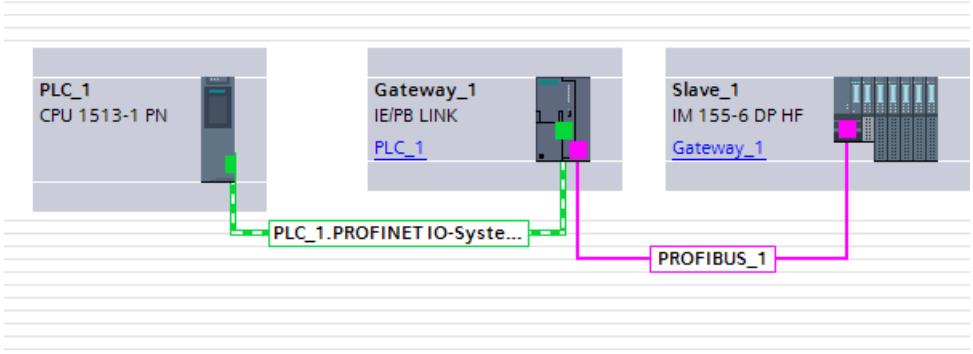

6. Select the IE/PB Link PN IO in the network view of STEP 7.

7. In the Inspector window, go to the "Gateway" area and select the "Network gateway as PROFINET IO proxy" option.

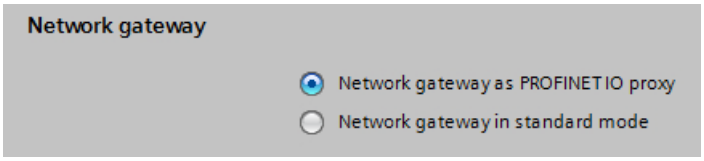

8. In the PROFINET device number area, you can assign a PROFINET device number for the DP slave.
   If you have selected the "Device number = PB address" check box (default), STEP 7 automatically assigns the device number according to the PROFIBUS address of the slave. In addition, you no longer need to update the device number if the PROFIBUS address changes.

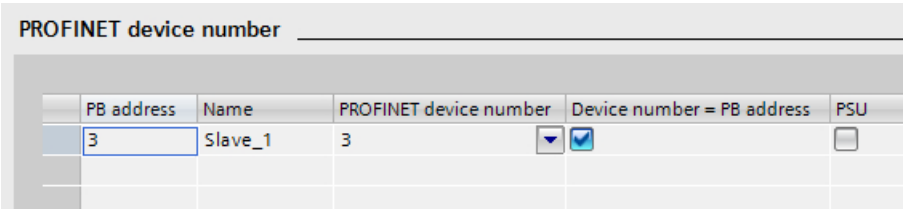

See also

Configuration using IE/PB link PN IO (Page 2838)

Setup recommendations for optimizing PROFINET

Optimizing PROFINET with RT

PROFINET provides you with high-performance communication on all levels.

The figure below shows an example of an optimized PROFINET topology.

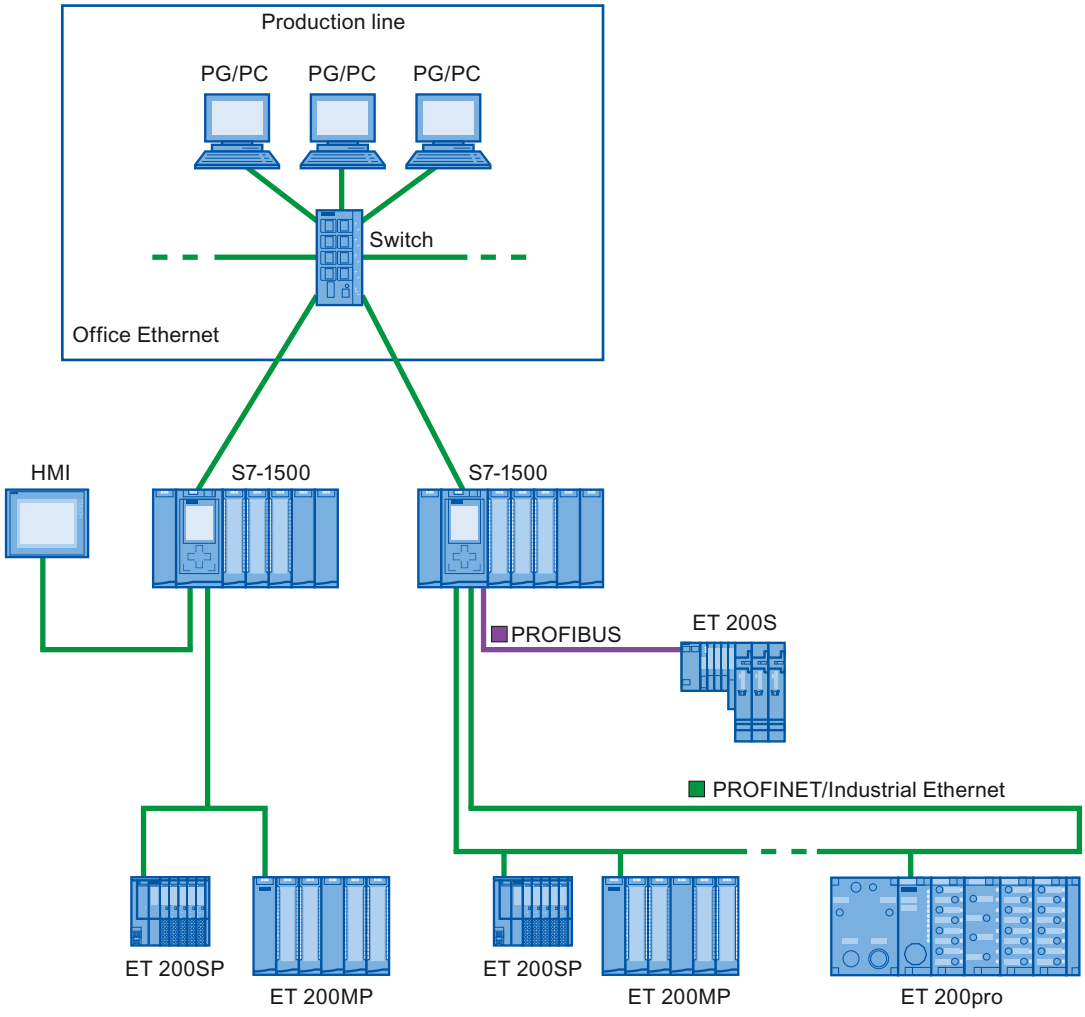

When setting up your PN network topology, take care that the various automation applications are distributed among separate network branches so that a sufficient bandwidth reserve will be available for future expansions.

- When you integrate standard Ethernet devices into the network topology or use standard Ethernet communication, take into account the network load caused by standard Ethernet and adapt the network topology as appropriate (max. bandwidth 100 Mbps).
- For communication with higher-level networks with a high data volume, use the most direct paths possible to the higher-level network infrastructure.

Also observe the Installation Guideline of the PROFIBUS User Organization.

Setting up PROFINET with IRT

Keep in mind the following rules for setting up and operating a PROFINET IO system in IRT mode. These rules will ensure best possible operation of your PROFINET IO system.

- When using IRT, you must configure the topology. This will enable exact calculation of the update time, bandwidth, and other parameters.
- If you would like to use multiple sync domains, configure a sync boundary for the port which is currently connected to the PROFINET device of another sync domain.
- In a sync domain, you can only configure one sync master at a time.
- A PROFINET IO system may only be part of a single sync domain.
- When you configure PROFINET devices in a sync domain and want to synchronize with IRT, the PROFINET devices concerned must support IRT communication.
- If possible, use the same PROFINET device as the PROFINET IO controller and sync master.
- If only some of the PROFINET devices in a PROFINET IO system are synchronized, please keep in mind the following: Assign PROFINET devices that are not participating in the IRT communication to the RT class "RT" and the synchronization role "unsynchronized" in the sync domain.

Applications for CPU with multiple PROFINET IO interfaces

- Connecting machines: Your configuration contains machines located on separate IO lines. You can operate real-time communication between the CPUs over X2 PROFINET IO interfaces. Use the I-device or shared I-device function for this purpose.
  The figure below shows an example configuration for 2 machines connected over the X2 interface with an I-device relationship.

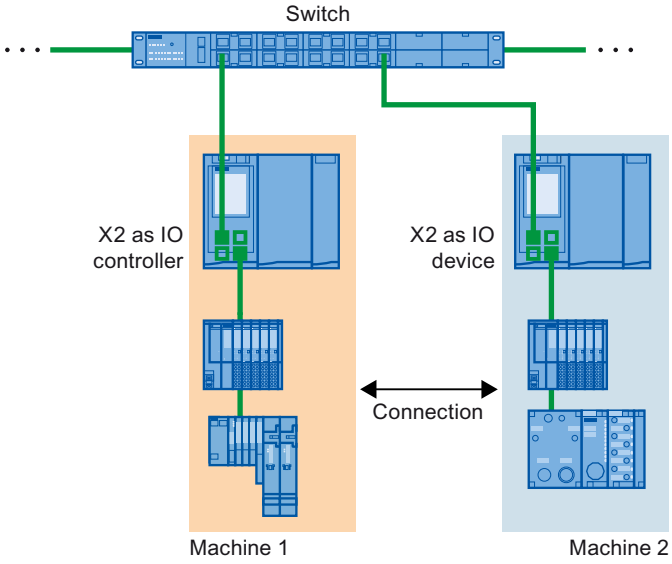

- Distribution by automation tasks:
  - For automation tasks with high performance and deterministic requirements, use PROFINET with IRT over the X1 interface.
  - For other tasks that you can implement with RT, use the X2 interface.

If you use interface X2 as the PROFINET IO interface for one of the following CPUs, this can affect performance:

- CPU 1515(F)-2 PN
- CPU 1515T-2 PN
- CPU 1516(F)-3 PN/DP
- CPU 1516(F)pro-2 PN

You can find additional information in the Cycle and Response Times function manual.

Topological overlap of IO systems in multi-controller applications

In a configuration with multiple IO controllers, shared paths are subject to the combined network loads of all connected PROFINET IO systems.

To avoid high communication loads in multi-controller applications, observe the following recommendations:

- Avoid paths that are shared by multiple IO systems.
  The figure below shows a configuration with two PROFINET IO systems that use the same paths.

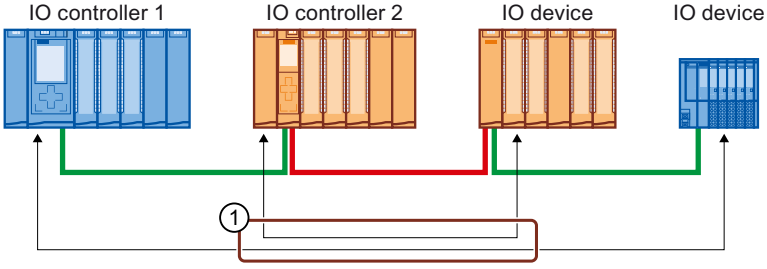

① Shared path

In the figure below, the two PROFINET IO systems do not use shared paths.

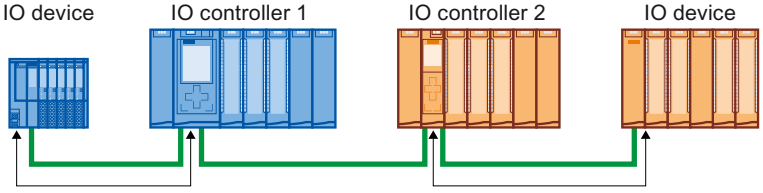

- If separation is not possible: Increase the update time for the affected IO devices.

Installation guidelines of the PROFIBUS user organization

The installation guidelines can be found on the Internet.

Limitation of the data infeed into the network

Limiting the data infeed into the network on PROFINET interfaces

The "Limit data infeed into the network" function limits the network load of standard Ethernet communication which is fed into the network by the interface to a maximum value. This does not apply to cyclic real-time communication (RT/IRT).

In a PROFINET IO system, critical network loads can occur quickly with standard Ethernet communication. All devices in a PROFINET IO system should support the "limitation of the data infeed into the network".

If you use devices that feed a lot of "standard Ethernet communication" into the PROFINET IO system, you should adapt the topology, if necessary.

Depending on the interface, you can activate or deactivate the "Limit data infeed into the network" function. If you use the X1 interface of an S7-1500 CPU as the IO controller or I-device, the "Limit data infeed into the network" function is always enabled. Example: If you are not using the X1 interface of an S7-1500 CPU as IO controller or I-device, you can enable or disable the function.

Uses of the limitation of the data infeed to the network

- Division of the bandwidth for standard Ethernet communication between devices:
  In PROFINET networks cyclic real-time communication and standard Ethernet communication share the same network. This means that only a limited bandwidth remains for standard Ethernet communication. The limitation of the data infeed ensures that the remaining bandwidth for standard Ethernet communication is not used just by one device, but is divided between several devices.
- Smoothing peaks in the data infeed:
  The limitation of the data infeed smooths peak loads of standard Ethernet communication (for example, from Open User Communication, access by the Web server).
- Limiting problems in the data infeed:
  If applications in a device generate too much data, this data is not forwarded to the PROFINET network. Negative effects (for example data loss, communication breakdown) remain limited to the device and its communications partner. Other nodes are not disturbed.

Setting limitation of the data infeed into the network for a CPU

To set the limitation of the data infeed into the network ("infeed load limitation"), follow these steps:

1. In the network view of STEP 7, select the interface of the CPU.
2. In the Inspector window, go to "Properties" > "General" > "Advanced options" > "Interface options".
3. Select or clear the "Limit data infeed into the network" check box.

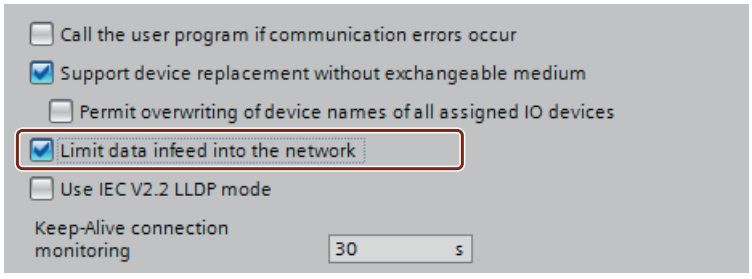

1.4.11.3 Using GSD files

GSD files for IO devices

Basic information on GSD files of IO devices

The properties of PROFINET IO devices are not stored in a keyword-based text file (as for PROFIBUS DP slaves), but in an XML file whose structure and rules are determined by a GSDML schema.

The language used to describe the GSD files is GSDML (Generic Station Description Markup Language). It is defined by the GSDML schema. A GSDML schema contains validation rules

that allow it, for example, to check the syntax of a GSD file. GSDML schemas (as schema files) are acquired by IO device manufacturers from PROFIBUS International.

Functional enhancements in the area of PROFINET IO will have an effect on the GSDML specification and the corresponding schema. A new version of the specification and of the schema is created by the functional enhancement.

Names of GSD files for IO devices

One possible example of a GSD file name for IO devices is:

"GSDML-V1.0-Siemens-ET200S-20030616.xml"

Name component	Explanation
GSDML	String at the start of each GSD file for IO devices
V1.0	Version of the GSDML schema
Siemens	Manufacturer
ET200S	Name of the device
20030616	Version code (date)
.xml	File extension

Versioning of GSD files for IO devices

The version information for GSD files for PROFINET IO devices consists of two parts:

- Version of the GSDML schema: This determines the language scope used by a GSD file.
- Version number in form of a date: The version number of GSD files is incremented, for example, after elimination of an error or introduction of a functional enhancement.

Functional enhancements may result in a new version of the GSDML schema. A new version of a GSDML schema might only be supported with restrictions.

Installing the GSD file

Introduction

A GSD file (general station description file) contains all properties of an IO device. If you want to configure an IO device that is not available in the hardware catalog, you must install the GSD file provided by the manufacturer. IO devices installed via GSD files are displayed in the hardware catalog and can then be selected and configured.

Requirement

- The hardware and network editor is closed.
- You have access to the required GSD files in a directory on the hard disk.

Procedure

To install a GSD file, follow these steps:

1. In the "Options" menu, select the "Manage general station description files (GSD)" command.
2. In the "Installed GSDs" tab, select the directory in which the GSD files are stored.
3. Choose one or more files from the list of displayed GSD files.
4. Click on the "Install" button. The selected GSD files are now being installed.
5. To create a log file for the installation, click on the "Save log file" button.
   Any problems during the installation can be tracked down using the log file.
6. Click "Close". You will be notified that the IO devices from the installed GSD files are entered into the hardware catalog. This process can take a few seconds.

By installing a GSD file you have added the IO device described in the file to the hardware catalog.

Result

You will find the new IO devices installed by means of GSD files in the hardware catalog under "Additional field devices > PROFINET IO".

GSD files are always saved together with the project, which means all information relevant for display of the device (including symbols) is also available in the saved project.

See also

Overview of hardware and network editor (Page 9)

Deleting GSD file

Introduction

You can delete installed DP slaves using GSD files. These DP slaves are then no longer displayed in the hardware catalog.

Requirement

- The hardware and network editor is closed.
- You will find the new IO devices installed by means of GSD files in the hardware catalog under "Additional field devices > PROFINET".

Procedure

To delete a GSD file, follow these steps:

1. In the "Options" menu, select the "Manage general station description files (GSD)" command.
2. In the "Installed GSDs" tab, select the directory in which the GSD file is stored.
3. Select the file that is to be deleted from the list of displayed GSD files.
4. Click the "Delete" button.

The selected GSD file was deleted and the DP slave is no longer located in the hardware catalog.

Finding unused GSD files

Introduction

When you add a GSD-based hardware component to your hardware configuration, different GSD files are added to the project data. If the GSD-based hardware component is once again deleted from the configuration, the GSD files remain in the data inventory of the project. The GSD files are no longer needed in the project if the corresponding GSD-based hardware component is not used in the hardware configuration again. You can find files that are no longer needed in the project with a search function and delete them.

Requirement

- A project must be open.
- The project must not be online.
- The project must not be write-protected.

Procedure

To find and delete unused GSD files, follow these steps:

1. In the "Options" menu, select the "Manage general station description files (GSD)" command.
2. Click on the "Find unused GSDs" button in the "GSDs in the project" tab.

   Note

   Unused GSD files are only available if at least one GSD-based hardware component was added to the configuration and then removed from the configuration. Unused GSD files in the project libraries are included in the search. GSD-based hardware components that are still being used in the configuration are not considered to be unused.

   Installed GSD-based hardware components that are included in the hardware catalog but have not been used in the hardware configuration of the project yet, could not have stored unused GSD data in the project.

3. If GSD files that are no longer needed were found in the project data, you can now delete them. To do so, click the "Delete" button.

GSD files that are no longer needed are deleted from the project data. The GSD-based hardware component is still available in the hardware catalog and, if necessary, can be added to the configuration once again. The necessary GSD data are once again copied to the project data in this case.

Note

If you delete a GSD-based hardware component with "Options > Manage general station description files (GSD) > Installed GSDs", it is deleted from the hardware catalog. However, unused GSD files of this hardware component can still be a part of the project data and can be deleted with the function for finding the unused GSD files.

NOTICE
Actions that cannot be undone When you delete GSD files that are not being used as an action, the action stack for actions that can be undone is deleted. This means the action of deleting unused GSD files as well as all previous actions can no longer be undone.

Changing the revision of a GSD file

Changing the revision of a GSD file

You can change the revision of a GSD file for an IO device:

- Only for the current IO device
- All suitable IO devices within the IO system
- All suitable IO devices within the complete project

First, all existing GSD files for the current IO device are shown. The only difference between the GSD files shown is their revision status. The currently used GSD file is highlighted.

Requirement

- The I/O data is the same for all IO devices whose revision is to be changed.
- The article number has not changed.
- The number of submodules is identical.
- The configuration data has not changed.
- There must be no module or submodule in a slot that is invalid after the new GSD file has been created.

Procedure

To change the revision of one or more IO devices, proceed as follows:

1. Select the IO device whose GSD file revision is to be changed.
2. Click on the "Change revision" button under "General> Catalog information" in the properties of the IO device.
   The "Change revision" dialog box opens.
3. Select the GSD revision you want to use in the "Available revisions" table.
4. Under "Use selected revision for", select the devices whose version are to be changed:
   – Only for the current IO device
   – For all suitable IO devices in the IO system
   – For all suitable IO devices in the project
5. Click the "Apply" button.

1.4.11.4 Special PROFINET configurations

Special features for S7-300/400/1500 and PC stations (S7-300, S7-400, S7-1500, PC)

Address and name assignments for PROFINET devices (S7-300, S7-400, S7-1500, PC)

In this chapter you will learn which address and naming conventions are valid for the PROFINET devices.

IP addresses

All PROFINET devices work with the TCP/IP protocol and therefore require an IP address for Ethernet operation.

You can set the IP addresses in the module properties. If the network is part of an existing company Ethernet network, ask your network administrator for this data.

The IP addresses of the IO devices are assigned automatically, usually at CPU startup. The IP addresses of the IO devices always have the same subnet mask as the IO controller and are assigned in ascending order, starting at the IP address of the IO controller.

Device names

Before an IO device can be addressed by an IO controller, it must have a device name. This procedure was chosen for PROFINET because names are easier to administer than complex IP addresses.

The device name is automatically derived from the name configured for the device (CPU, CP or IM):

- The PROFINET device name is made up of the name of the device (for example, the CPU), the name of the interface (only with multiple PROFINET interfaces) and optionally the name of the IO system:
  <CPU name>.<Name of the interface>.<IO system name>
  You cannot change this name directly. You change the PROFINET device name indirectly, by changing the name of the affected CPU, CP or IM in the general properties of the module. This PROFINET device name is also displayed, for example, in the list of accessible devices. If you want to set the PROFINET device name independently of the module name, you have to deactivate the "Generate PROFINET device name automatically" option.
- A "converted name" is generated from the PROFINET device name. This is the device name that is actually loaded into the device.
  The PROFINET device name is only converted if it does not comply to the rules of IEC 61158-6-10. You cannot change this name directly either.

Device number

In addition to the device name, a device number is also automatically assigned when an IO device is plugged in. You can change this number.

An IO device can be identified in the user program with this device number, for example, with the instruction "LOG_GEO". The device name, unlike the device number, is not visible in the user program.

Devices in the PROFINET subnet

In a PROFINET subnet the maximum allowable number of devices is monitored during configuration.

What you should know about the send clock (S7-300, S7-400, S7-1500, PC)

Send clock - the smallest possible update interval

The send clock shows the time between two successive communication cycles. It is the shortest possible transmission interval for data exchange. The update times are calculated as multiples of the send clock.

Requirements for changing the send clock

For unsynchronized devices or when the device is an IO controller as sync master, the send clock is configured on the device. For synchronized devices (sync master, sync slave), the send clock can be configured in the sync domain.

The send clock cannot be changed on the device if it is not an IO controller as sync master and at least one device is sync slave or sync master in the IO system. In this case, only the send clock which was selected in the properties of the respective synchronization domain is displayed. To change this you must call the properties of the respective sync domain ("Domain Settings" button).

The send clock can be changed on the device if all devices on the IO system are unsynchronized or when the device is an IO controller as sync master. You can see the resulting update times for the individual IO devices in the IO communication table or in the properties of the individual IO devices.

Effect of the send clock on the maximum possible bandwidth for cyclical IO data

The following table shows how the send clock affects the maximum available bandwidth for cyclical IO data:

Send clock	Maximum bandwidth for cyclic IO data
250 µs – 468.75 µs	<< 125 µs
500 µs – 968.75 µs	= send clock / 2
1 – 4 ms	= 500 µs

The send clock influences the update time of IO devices for which the "Automatic" and "Can be set" option is selected (with "Adapt update time when send clock changes" option selected).

When the "Can be set" option is selected for IO devices, this can lead to inconsistencies in the configuration. On these IO devices you must subsequently adjust the update time, if necessary.

Setting the send clock (S7-300, S7-400, S7-1500, PC)

Requirements to change the send clock at the PROFINET device

The PROFINET device must be an IO controller as sync master or IRT (Isochronous Realtime) must not be configured in the IO system.

If the PROFINET device is not an IO controller as sync master, the following requirements must be met for the setting of the send clock:

- No device must be configured at the IO system as a sync slave or sync master.
- All devices at the IO system must be unsynchronized.

When IRT is configured with an IO controller as sync master, the send clock can be set at the device or in the sync domain.

Procedure

To set the send clock on the PROFINET device, follow these steps:

1. Select the PROFINET IO controller in the device or network view.
2. Change the value for the shortest possible update interval in the properties of the PROFINET interface under "PROFINET Interface > Advanced options > Real-time settings > IO communication > Send clock".

The send clock is valid for all PROFINET devices at the IO system. If the synchronization role is set to a value other than "Unsynchronized", you can also set the send clock in the sync domain, in other words, centrally at the PROFINET IO system.

1. Select the IO system using the PROFINET subnet.
2. Change the value for the send clock in the properties of the sync domain under "Domain management > Sync domains > Sync_domain_1 > Send clock".

See also

Configuring PROFINET IO with IRT (Page 2899)

Adjusting odd send clocks (S7-300, S7-400, S7-1500, PC)

S7-1500 CPUs support both even and odd send clocks.

Even send clocks: 0.25 ms, 0.5 ms, 1 ms, 2 ms, etc.

Odd send clocks: 0.375 ms, 0.75 ms, 1.125 ms, etc.

For IO devices that do not support the fast send clock of the CPU (i.e., of the IO controller) or that do not support odd send clocks, the CPU "adapts" the send clock of the IO device. Thus, for example, it is possible that the CPU (IO controller) and a connected IO device both operate with an 875 µs send clock, while another IO device operates with a 2 ms send clock.

Rules

- The send clock adaptation only works for unsynchronized IO devices (RT class "RT").
- For synchronized IO devices (RT class "IRT"), a uniform send clock that is set in the sync domain applies. The update time corresponds to the send clock in this case.
- For IO devices whose update time option is set to "Automatic," an update time of 2 ms or greater is set by STEP 7, i.e., update times of less than 2 ms are not possible with this option.

Example

A PROFINET IO master system consists of the following components:

- IO controller: CPU 1516-3 PN/DP (6ES7 516-3AN00-0AB0)
- IO device 1: ET 200MP (6ES7 155-5AA00-0AB0) with one DI module
- IO device 2: ET 200S (6ES7 151-3BA23-0AB0 V6.0) with one DI module

IO device 2 does not support the odd send clock (here: 0.875 ms).

1. Assign the IO devices to the IO controller.
2. Set the "Sync slave" synchronization role for IO device 1 (inspector window, PROFINET interface area > Advanced options > Realtime settings > Synchronization).
3. For the IO controller, you set the synchronization role "sync master" in the same way.
4. Interconnect the ports of the IO controller and IO device 1 (e.g., in the topology view).
5. IO device 2 remains unsynchronized; the update option is set to "Automatic".
6. Specify the 0.875 ms send clock in the sync domain (subnet properties). All devices in the sync domain then have this send clock or update time.

The unsynchronized IO device 2 automatically receives the update time of 2 ms.

Configuration with external IO controller (S7-300, S7-400, S7-1500, PC)

Configuration with external IO controller

CPs which can be used as an external IO controller have a number of communications options and are therefore not designed for use as an IO controller.

If you use an external IO controller (such as CP 443-1 Advanced), you must integrate an IO system after inserting the IO controller.

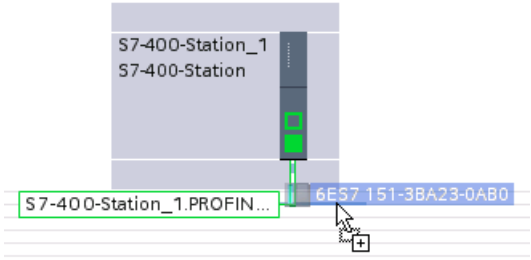

You can then drag the desired IO devices from the hardware catalog to the PROFINET IO system.

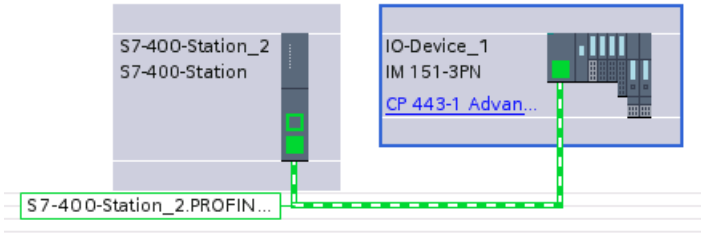

Configuration with SIMATIC PC Station (S7-300, S7-400, S7-1500, PC)

Configuration with SIMATIC PC Station

A "PC Station" is a PC with communications modules and software components within a SIMATIC automation solution.

By using corresponding communications modules and software components you can operate a PC station as a PROFINET IO controller.

Your PC applications on the PC Station have the following access options to the PROFINET IO controller:

- As OPC client via the OPC server PROFINET IO
- Directly via the PROFINET IO user interface (IO base programming interface)

PC applications can only ever use one of these access possibilities at a time (open/close sequence)

Functions	OPC Server	RTE-base programming interface
Reading and writing IO data	Yes	Yes
Reading and writing data records	Yes	Yes
Receiving and acknowledging interrupts	No	Yes

The following figure shows a PC Station with the components described.

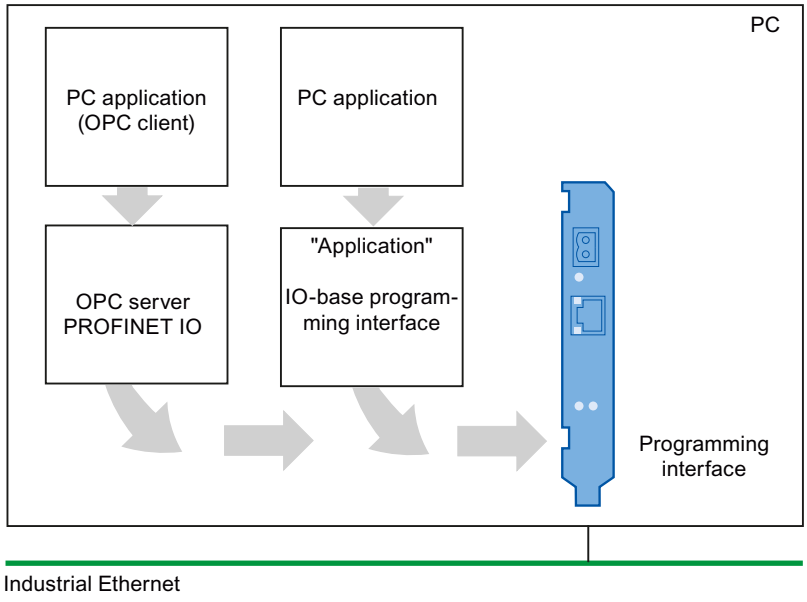

Configuration in the network view

The following figure shows a PC Station with an OPC server and a PC application. The CP 1612 Ethernet interface added converts the PC station into an IO controller with connected IO devices. IE/PB link has also been used as a gateway from PROFINET to PROFIBUS.

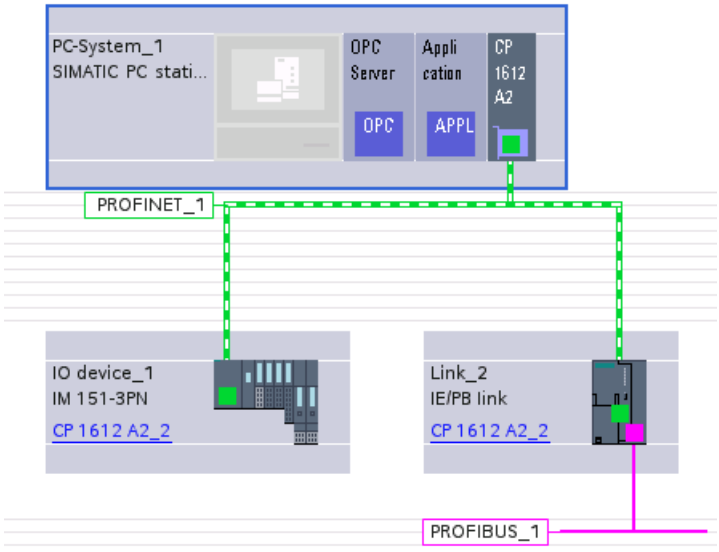

IO routing with CP 1604 / 1616 / 1626 and CP 1616 onboard (S7-300, S7-400, S7-1500, PC)

CP as PROFINET IO router - what this means in practice

You can operate a PC station with the CP 1604 / 1616/ 1626 as PROFINET IO router by employing the relevant CP as PROFINET IO device and PROFINET IO controller simultaneously. This IO routing allows process data to be exchanged between two PROFINET IO systems. If the configuration is set up correctly the CP can, for example, allow certain process level inputs or outputs to be accessed at control level.

In a typical configuration the CP then simultaneously satisfies the following functions:

- PROFINET IO Device for a higher-level PROFINET IO controller of the control level
- PROFINET IO controller with connected IO devices for the subordinate process level

Assign configuration - transfer area "TM" (IO routing area)

In the configuration, select the "IO device" option under "Operating mode" .

The additional "I-device communication" configuration area appears, in which you can make the settings for the IO routing. In the "Transfer area" table there, assign the data areas for data exchange with the higher-level IO controller.

In this case, you have the following options in the "Type" column:

- CD: Controller Device
  Select this option to specify a transfer area between the IO controller and IO device. Refer to additional information on this in the online help under "Configuration for PROFINET IO".
- TM: IO routing (transfer module mapping)
  Select this option for the IO router application. An additional configuration area with the name specified appears in the Inspector window under "Operating mode / I-device communication". In this configuration area, make the assignment for the IO routing in the IO routing table.
- F-PS: PROFIsafe
  Select this option for the configuration of a PROFIsafe transfer area. The PROFIsafe transfer area is bidirectional and occupies two rows in the table of transfer areas. The length of the PROFIsafe transfer areas is limited to 8 bytes of user data each for input and output.

IO routing table

In the higher-level IO system, IO routing areas represent the data routed between the lower-level IO system and the higher-level IO system.

Enter the IO devices from the lower-level IO system in the IO routing table according to the transfer areas.

The table contains the following columns:

Parameter	Meaning
TM offset	Offset in relation to the first bit in the IO routing area. The specification has the form <Byte>.<Bit>. Example: 0.5
Device/module (slot)	Specification of the PROFINET device name of the IO device and its module and slot. The displayed dialog shows the transfer areas of the IO devices configured in the lower-level IO system.
IO type	Select the IO type from the drop-down list. This selection is only possible with an IO routing area of the type "Input", and only if a mixed module is selected as "Device/module (slot)". In this case, you can select whether the inputs are read (Input) or the outputs are read back (Output).
Start	Start address (offset) in the data area of the transfer area in the lower-level IO system. The specification has the form <Byte>.<Bit>. Example: 0.0
End	End address (offset) in the data area of the transfer area in the lower-level IO system. The specification has the form <Byte>.<Bit>. Example: 8.0

PROFINET device diagnostics (S7-300, S7-400, S7-1500, PC)

PROFINET device diagnostics

The diagnostic methods available for PROFIBUS DP are the same as those for PROFINET IO. The procedure is identical. The evaluation of diagnostic information is somewhat different with the extended instructions for the distributed I/O in the S7-300/400 CPU user program.

In PROFINET IO the structure for data records and diagnostic information is manufacturer-independent. Diagnostic information is provided only for defective channels.

By using the system status lists (SSLs) and the extended instructions "RALRM" and "RDREC" you can make the PROFINET IO system status and the S7 user program diagnostic information available:

- In order to get a complete overview of the PROFINET IO system status, read SSL 0x0X91 with "RDSYSST".
- To read diagnostic data records directly from a defective module, use the extended instruction "RDREC" (Read data record). You will receive status-related, detailed error information.
- To read event-related diagnostic data records (triggered by error OB), use the extended instruction "RALRM" (read additional alarm info) in the corresponding error OB.

The extended instruction "RALRM" and "RDREC" can also be used for PROFIBUS DP.

You can find information as to which SSLs and which diagnostic data records are defined for PROFINET IO and how the diagnostic data records are structured here (http://support.automation.siemens.com/WW/view/en/19289930).

Configuring IRT communication

Introduction: Isochronous Realtime Ethernet

Introduction: Isochronous Realtime Ethernet

IRT is a transmission method by which PROFINET devices are synchronized with very high accuracy.

A sync master specifies the clock to which sync slaves are synchronized. An IO controller or an IO device can have the role of sync master.

Sync master and sync slaves are always devices in a sync domain. Bandwidth is reserved within the sync domain for IRT communication. Real-time and non-real-time communication (TCP/IP communication) is possible outside of the reserved bandwidth.

Time ranges of communication cycles

The communication cycle is divided into three time ranges, which are represented in the following chart:

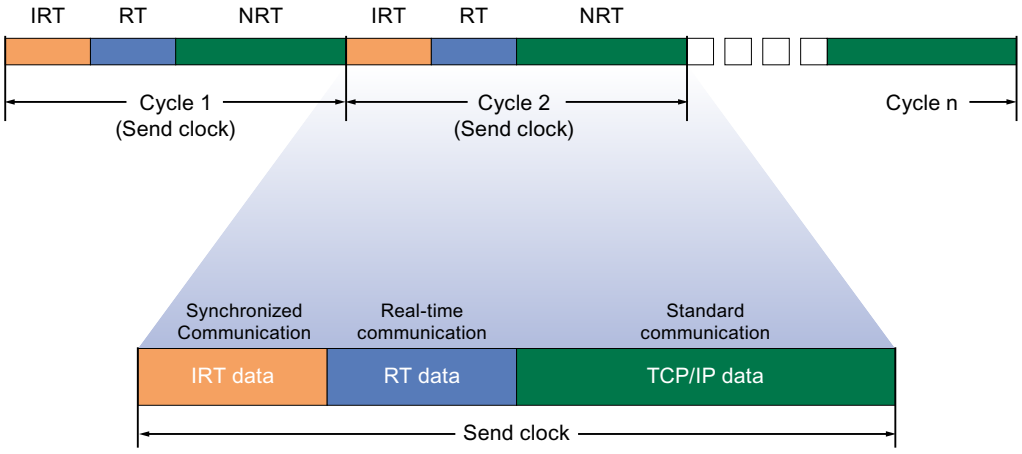

- IRT data (synchronized communication)
  This area can be reserved in specific steps, depending on the send clock. Within this time range the IRT data is transmitted.
- RT data (real-time communication)
  In this time range the cyclic RT date is transmitted. RT data has a higher priority than "normal" TCP/IP data. TCP/IP data or Ethernet frames can have a priority between 1 and 7. RT data has a priority of 6.
- TCP/IP data (standard communication)
  Standard communication (TCP/IP, etc.) is transmitted in the remaining interval of the communication cycle.

Applications of IRT

PROFINET with IRT is especially suited for:

- High performance and deterministics for large quantity structures with regard to user data communications (productive data)
- High performance even with many devices in the line topology with regard to user data communications (productive data).
- Parallel transmission of production and TCP/IP data over wire, and securing the forwarding of production data when data volume is high by reserving transmission bandwidth.

See also

Basic procedures for configuring IRT (Page 2899)

Configuring PROFINET IO with IRT (Page 2899)

Sync domain (Page 2901)

Basic procedures for configuring IRT

Basic procedures for configuring IRT

If you want to upgrade equipment with PROFINET IO to IRT, follow these three steps:

1. Configure the PROFINET IO device. The PROFINET device must support IRT.
2. Set which device acts as the sync master and synchronizes the other devices. For this you must configure a sync domain with a sync master and at least one sync slave.
3. Download the configuration to all involved devices.

Creating PROFINET IO configuration for IRT

As a prerequisite for IRT configuration, there must be an existing PROFINET IO configuration, i.e., one or more stations must be configured with an IO controller and IO devices.

You can read which components are suitable for IRT communication in the task card "Information", once you have selected the corresponding components in the hardware catalog.

The requirements for functioning IRT communication are met simply by configuring a PROFINET IO system with components that support IRT.

Configuring PROFINET IO with IRT

Introduction

If you want to increase the precision of cyclic data exchange via PROFINET IO with IRT, you must first configure the PROFINET IO devices. These PROFINET devices must support IRT. Now set which device is to act as sync master and synchronize the other devices. This is done by configuring a sync domain with a sync master and at least one sync slave.

Requirement

- There is an IO system with an IO controller and at least one IO device.
- All devices involved support IRT.

Procedure

Proceed as follows to configure IRT for an existing IO system:

1. Select the PROFINET interface of the **IO controller**.
2. Navigate to "Advanced options > Realtime settings > Synchronization" in the inspector window.
3. Assign the IO controller the role of sync master under "Synchronization role".
4. Now select the PROFINET interfaces of an associated **IO device**.
5. Navigate to "Advanced options > Realtime settings > Synchronization" in the inspector window.

6. Activate the RT class "IRT". The IO device is then automatically assigned the synchronization role "sync slave".
7. You can check and correct your settings at any time using the "Sync domain" button.
8. Interconnect the ports between the sync slaves and sync master.

Or

1. Highlight the PROFINET IO system in the network view.
2. Click on the PROFINET IO system.
3. Navigate to the node of the required sync domain in the inspector window.
4. Enter all necessary settings in the tables:
   - Select an IO system.
   - Set the synchronization role "sync master" for the IO controller.
   - Set the RT class for the IO devices to "IRT". The IO devices is then automatically assigned the synchronization role "sync slave".
5. Interconnect the ports between the sync slaves and sync master.

You can now load the configuration to the relevant devices with PROFINET IRT.

Note

Automatic assignment of a sync master

You can also simply configure an I-device or IO device as sync slave by switching the RT class from RT to IRT. The IO controller that has been assigned to the I-device or IO device is automatically configured as sync master.

If you have configured the IO controller as a sync slave, this setting is not overwritten. The IO controller is thus not switched automatically from sync slave to sync master.

If you assign more than one IO controller to the IO device as a Shared Device, the IO controller automatically selected as the sync master is the one that is assigned to the interface of the IO device.

Note

Switching between RT and IRT

In hierarchical IO systems, IRT communication can be combined with RT communication but cannot run simultaneously in both IO systems: If the option "Parameter assignment for the PN interface and its ports on the higher-level IO controller" is selected on the I-device, the I-device can use IRT communication on the higher-level IO controller. If the option is disabled, the lower-level IO system can use IRT communication but you cannot switch between RT and IRT on the I-device.

Sync domain

Sync domain

A sync domain is required for synchronization of PROFINET IO devices. The sync domain assures that all devices belonging to it communicate synchronously.

Prerequisite for IRT communication is a synchronization cycle to distribute a common time base for all PROFINET devices in a sync domain. With this base synchronization, a synchronous transmission cycle of PROFINET devices within a sync domain is achieved. The sync master (usually an IO controller) generates the common synchronization clock and specifies the time basis on which all other sync slaves (usually IO devices) are synchronized.

If the sync master fails and no redundant sync master was configured, the communication between the IRT devices reverts to RT quality.

Background processes during configuration of an IO system

If you configure an IO controller and network it with an Ethernet subnet, it will automatically be added to the default sync domain of the Ethernet subnet. The default sync domain is always available. The IO controller operates unsynchronized.

If you assign an additional IO device to the IO system of the IO controller, the IO device automatically assigns the IO controller's sync domain.

See also

Working with sync domains (Page 2901)

Configuring a redundant sync master (Page 2904)

Working with sync domains

Working with sync domains

All components that take part in IRT communication must belong to a sync domain.

A sync domain is a group of PROFINET devices that are synchronized to a common clock. Exactly one device has the role of sync master (clock); all other devices have the role of sync slaves.

Default sync domain

If you have applied an Ethernet subnet, a special sync domain is automatically applied: the default sync domain. All PROFINET devices configured for the Ethernet subnet automatically belong to this sync domain.

The following figures illustrate the interaction:

① Default domain; always present

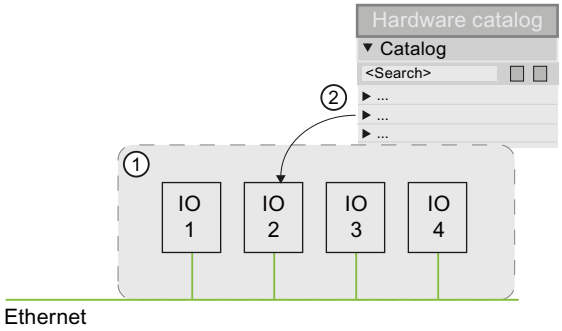

①	Default domain; always present
②	Newly configured PROFINET IO systems are first located in the default domain.

Editing properties of sync domains

Editing properties of sync domains

Details of how to define the name and the send clock in a sync domain are given below. In addition, you will learn how to set a sync domain as default domain.

Requirement

- At least one IO device is created in the project.
- At least one sync domain already exists.

Editing a sync domain

To change the name of a sync domain, adjust the send clock, or set up a default domain, follow these steps:

1. Select the Ethernet subnet in the network view in which the sync domain is set up.
2. Open the Ethernet subnet properties.

3. In the local navigation, choose the entry "Domain management > Sync domains > <Name of sync domain> ".
4. Enter the changes.
   - Enter the desired name of the sync domain in the field "Sync domain name".
   - Choose the send clock from the drop-down menu "Send clock".
   - Select the "Default domain" check box if you wish to set the sync domain as the default domain.

Bandwidths for the various RT classes

Available and calculated bandwidths for the various RT classes within a sync domain are displayed in addition to the name and send clock settings.

Select the "Details" section in the area navigation under the sync domain name.

See also

Bandwidth use for synchronized IO controllers (Page 2903)

Bandwidth use for synchronized IO controllers

Use of the bandwidth

Here you set how much send time there is for the cyclic IO data (real-time data by IRT) and the other data (data by TCP/IP or UDP/IP, RT data).

With this, you specify how the bandwidth (transmission capacity) is used.

Cyclic IO data by IRT

Cyclic IO data is sent using IRT (isochronous real time).

Certain periods are exclusively available for IRT.

Reserved period for IRT

You decide the duration of these reserved periods (time slices) by selecting them in the drop-down list.

All other data is transferred in the time remaining until the beginning of the next send clock.

Select from the following options:

- Maximum 25% cyclic IO data. Focus on non-cyclic data (data using TCP/IP or UDP/IP)
- Maximum 37.5% cyclic IO data. Focus on non-cyclic data (data using TCP/IP or UDP/IP)
- Maximum 50% cyclic IO data. Balanced proportion
- Maximum 90% cyclic IO data. Focus on cyclic data (data using IRT)

See also

Configuration of IRT with high performance (Page 3009)

Defining devices of a sync domain

Defining devices of a sync domain

All components that take part in IRT communication must belong to a sync domain. Instructions on how to define the nodes of a sync domain are given below.

Requirement

PROFINET IO systems are already configured and are in the default domain.

Adding a new device to the sync domain

To assign a sync domain to a device, follow these steps:

1. Select the Ethernet subnet in the network view in which the sync domain is set up.
2. Open the Ethernet subnet properties.
3. In the local navigation, select the entry "Domain management > Sync domains > <Name of sync domain> > Devices".
4. Click "Add devices" in the "Station / IO System" table.
   The "Select IO system" dialog opens.
5. Select all devices you want to assign to the sync domain and click "OK".

The selected IO system will be removed from the original sync domain and assigned to the current sync domain.

Removing devices from the sync domain

To remove a device from a sync domain, follow these steps:

1. Select the Ethernet subnet in the network view in which the sync domain is set up.
2. Open the Ethernet subnet properties.
3. In the local navigation, select the entry "Domain management > Sync domains > <Name of sync domain> > Devices".
4. Select the devices in the table "Station / IO-System" which you would like to remove and press <DEL>.

The devices will be deleted and reassigned to the default domain.

Configuring a redundant sync master

You can configure a redundant sync master for PROFINET IO devices in IRT operation.

Example: An S7-1500 CPU can be configured as a sync master, a Scalance 6GK5 201-3BH00-2BA3 V5.0 can be configured as a redundant sync master.

Principle of operation

A redundant sync master belongs to the same sync domain as the "primary" sync master. It prevents a communication failure in the sync domain in the event the primary sync master fails.

Both the sync master and the redundant sync master send sync frames. A sync slave is synchronized to the sync frame of the sync master. If the sync master fails, a sync slave is synchronized automatically to the sync frames of the redundant sync master.

Rules

- The redundant sync master role is assigned to only one device in the sync domain.
- The sync domain contains only IRT devices with "IRT with high performance" option.
- There are a maximum of 2 nodes between the sync master and the redundant sync master.

Procedure

To configure a redundant sync master, follow these steps:

1. In the device view or network view, select the device that you want to configure as a redundant master.
2. Under "PROFINET interface > Advanced options > Real time settings > Synchronization > Synchronization role", select the "Redundant sync master" option.

Rules and information for IRT configurations

The following rules and information apply to PROFINET configurations. Additional system recommendations for optimizing your PROFINET system can be found here.

Setting up PROFINET with IRT

Keep in mind the following rules for the set up and operation of a PROFINET IO system in IRT operation. They serve to insure an optimal operation of your PROFINET IO system.

- When using IRT, you must configure the topology. This will enable exact calculation of the update time, bandwidth, and additional parameters.
- If you would like to use multiple sync domains, configure a sync domain boundary for the port which is currently connected to the PROFINET device of another sync domain.
- In a sync domain you can only configure one sync master at a time.
- A PROFINET IO system may only be part of a single sync domain.
- If you configure a PROFINET device in a sync domain and want to synchronize with IRT, the PROFINET devices concerned must support IRT communication.
- Wherever possible, use the same PROFINET device as the PROFINET IO controller and sync master.

- If only some of the PROFINET devices in a PROFINET IO system are synchronized, please keep in mind the following: PROFINET devices which are not part of IRT communication are placed outside of the sync domain.
- Higher-level and lower-level IO systems can use RT communication simultaneously. IRT communication can be combined with RT communication but cannot run simultaneously in both IO systems: If the option "Parameter assignment for the PN interface and its ports on the higher-level IO controller" is selected on the I-device, the I-device can use IRT communication on the higher-level IO controller. If the option is disabled, the lower-level IO system can use IRT communication.

See also

Configuration of IRT with high performance (Page 3009)

Configuring isochronous mode

What is isochronous mode?

Objectives of isochronous operation

Assuming public transport were to operate at maximum speed while reducing stop times at the passenger terminals to absolute minimum, the last thing many potential passengers would notice of the departing contraption are its red tail lights. The overall travel time is, however, decided by the train, bus or underground clock, because well adjusted timing is essential to a good service. This also applies to distributed automation engineering. Not only fast cycles but also the adaptation and synchronization of the individual cycles result in optimum throughput.

Just-In-Time

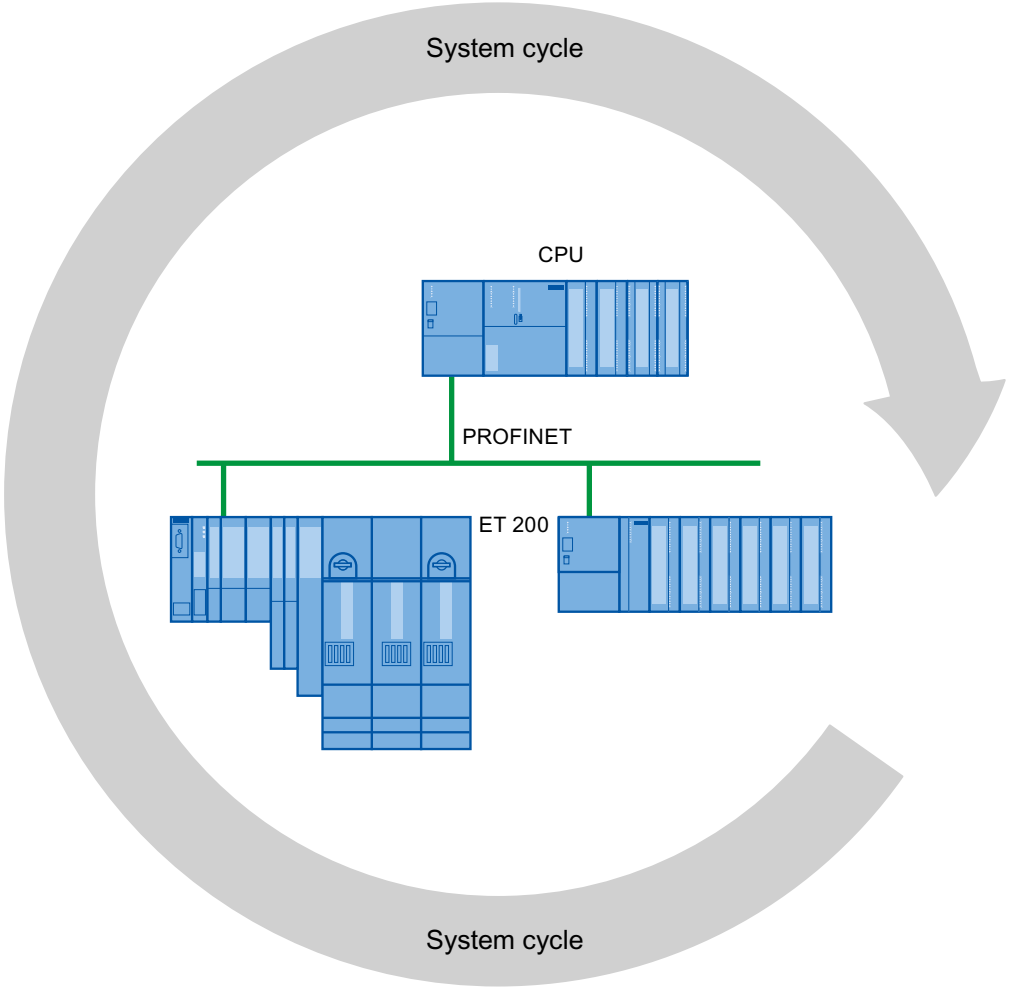

The high speed and reliable reaction time of a system operating in isochronous mode is based on the fact that all data are provided just-in-time. The constant PROFINET IO cycle beats the time here.

Use of isochronous mode

Use of isochronous mode

With the "Isochronous mode" system property, measured values and process data can be acquired in a fixed system clock. The signal processing up to the switching-through to the "output terminal" occurs within the same system clock. Isochronous mode thus contributes to high-quality control and hence to greater manufacturing precision. With isochronous mode, the possible fluctuations of process response times are drastically reduced. The time-assured processing can be utilized to improve machine cycle times.

In principle, isochronous mode is worthwhile whenever measured values must be acquired synchronously, movements must be coordinated, and process responses must be defined and simultaneously executed, as in the following example.

Time sequence of synchronization

From reading-in or input data to outputting of output data

The basic time sequence of all components involved in synchronization is explained in the following:

- Reading-in of input data in isochronous mode
- Transport of input data to the IO controller (CPU) via the PROFINET subnet
- Further processing in the isochronous application of the CPU
- Transport of output data to the outputting IO device via the PROFINET subnet
- Outputting of output data in isochronous mode

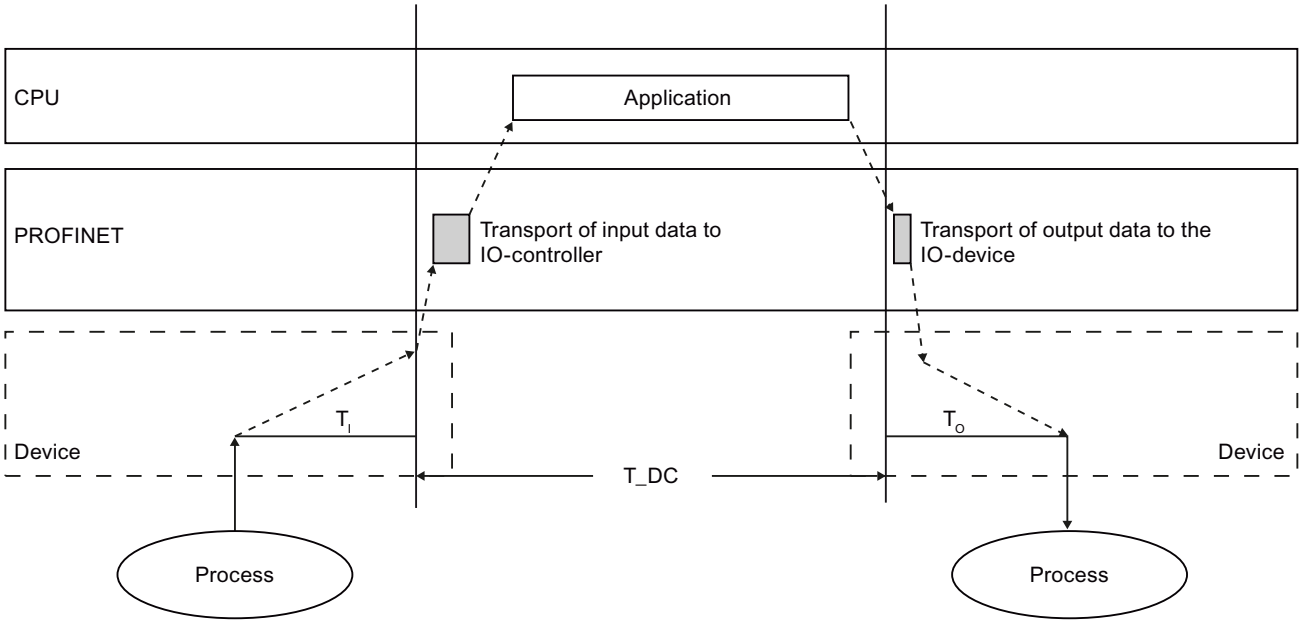

T_DC	Data cycle (Time_DataCycle)
T_I	Time for reading in the input data
T_O	Time for outputting the output data

To ensure that all input data is ready for transportation via the PROFINET IO line when the next PROFINET IO cycle begins, the IO read cycle has a lead time T_I so that it starts earlier. T_I is the "flashbulb" for the inputs; at this instant, all synchronized inputs are read in. T_I is necessary in order to compensate for analog-to-digital conversion, backplane bus times, and the like. The lead time T_I can be set automatically by STEP 7, taken from the settings of the associated isochronous mode interrupt OB, or configured by the user. We recommend allowing STEP 7 to automatically assign the lead time T_I.

The PROFINET IO line transports the input data to the IO controller. The isochronous mode interrupt OB (OB 61, OB 62, OB 63, or OB 64) is called after a parameterizable delay time. The user program in the isochronous mode interrupt OB decides the process reaction and provides the output data in time for the start of the next data cycle. The length of the data cycle is always configured by the user.

T_O is the compensation arising from the backplane bus and the digital-to-analog conversion within the device. T_O is the "flashbulb" for the outputs; at this instant, the synchronized outputs are output. The time T_O can be set automatically by STEP 7, taken from the settings of the associated isochronous mode interrupt OB, or configured by the user. We recommend allowing STEP 7 to automatically assign the time T_O.

Isochronous mode and non isochronous mode distributed I/O

It is possible to combine isochronous mode distributed I/O with non isochronous mode distributed I/O on one IO controller.

The Ti Value

Effect of T_I

The effect of T_I is illustrated by the following figure:

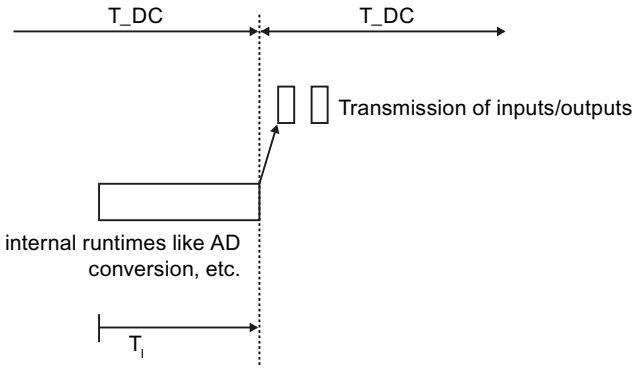

T_DC	Data cycle
T_I	Time for reading in the input data

Sequence

So that a consistent state of the inputs can be transferred to the IO controller at the start of the new system clock cycle, the import must be brought forward by the time T_I. The time T_I for a particular input module includes at least the signal conditioning and conversion time on the electronics modules and the time for transfer to the interface module on the IO device backplane bus.

In the plant, the values are imported at the same time because the T_I of the various input modules is set to the same value and this value is greater than or equal to the highest minimum T_I of all input modules. With the default setting, Step 7 makes sure that the smallest possible mutual T_I is set.

The To Value

Effect of T_O

The effect is illustrated by the following figure:

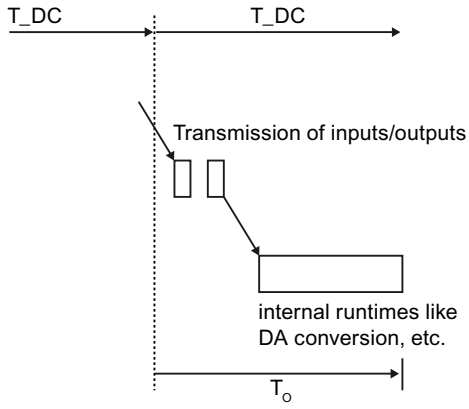

T_DC	Data cycle
T_O	Time of data output

Sequence

To ensure that a consistent status of the outputs can be transferred to the process at the start of the new system clock cycle, the data is output at the time T_o following the last clock beat. The time T_O records the time of the transfer from the IO-controller to the IO device (through PROFINET IO) for a certain output module and in the IO device, the transfer of the outputs from the interface module to the electronic module (backplane bus)(if applicable, plus time for the digital-analog conversion).

The system makes sure that the values are written simultaneously by setting the T_O of all isochronous output modules to the same value. This value must be greater than or equal to the largest minimal T_O of all isochronous output modules. STEP 7 automatically calculates the shared smallest possible T_O.

Isochronous interrupt OB (OB 6x)

You use the isochronous mode interrupt OB (e.g., OB 61) for the isochronous connection of the user program.

Call of isochronous mode interrupt OB

The isochronous interrupt OB is triggered by the system clock, or more precisely by an adjustable delay time after the system clock. The effect is illustrated by the following figure:

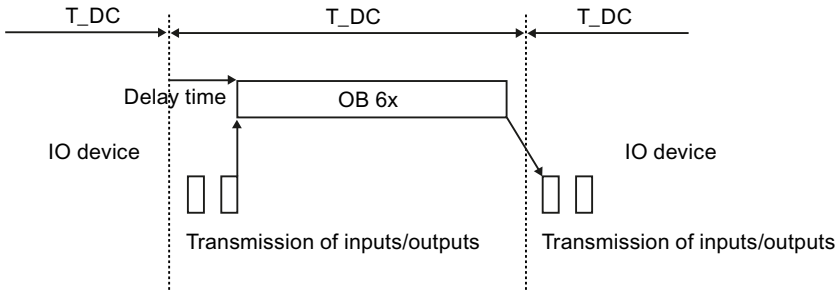

T_DC Data cycle

Delay time

STEP 7 initially automatically calculates a suitable value for the delay time. The delay time compensates the transfer time of the inputs of the isochronous IO devices to the IO controller through the PROFINET IO-network. The execution of OB 6x is linked to the system clock and the specified delay time. The delay time can also be manually corrected, if necessary. For example, you can shorten the delay time in order to process program sections that do not access the isochronous I/O at the start of OB 6x. This gains you runtime for the isochronous user program.

Connection of isochronous I/O to the isochronous mode interrupt OB

You access the isochronous I/O via a process image partition, i.e., the addresses of isochronous modules must be in a process image partition.

Any access to isochronous I/O must be programmed by the SYNC_PI and SYNC_PO instructions within the isochronous mode interrupt OB.

You call SYNC-PI at the start of OB 6x, provided you have selected the automatic setting for the delay time. You call SYNC_PO at the end of OB 6x.

T_DC, application cycle, and send clock

The time T_DC designates the time for one data cycle, i.e., this time is the shortest possible time for reading in, processing, and outputting the data (IPO model). T_DC is always a multiple of the send clock.

If IRT is activated for an IO device, T_DC is then equal to the send clock. The update time corresponds to the send clock.

The application cycle is a multiple of the data cycle. It can be identical to the data cycle if the runtime of the isochronous mode interrupt OB is correspondingly short.

Program processing according to the IPO model with short time

The IPO model with application cycle factor = 1 is characterized by the fact that the I/O data are completely processed within one system clock T_DC. This model is used to achieve the shortest response times.

Program Execution According to the IPO Model with Long Time

Label of the IPO-model with application cycle factor > 1

If the runtime of the isochronous mode OB is longer than one data cycle T_DC, the output must be delayed by one application cycle (OB 6x). For this reason, data output occurs before data input in this model.

The SYNC_PI and SYNC_PO instructions must then be called during the execution window of the first data cycle (only available there when application cycle factor > 1).

The execution window refers to the time period during which a call of the SYNC_PI and SYNC_PO instructions is possible, i.e., after transfer of cyclic data via PROFINET up to the time shortly before the end of data cycle T_DC.

The other user program processing in OB 6x then occurs in the next data cycle or over the following data cycles.

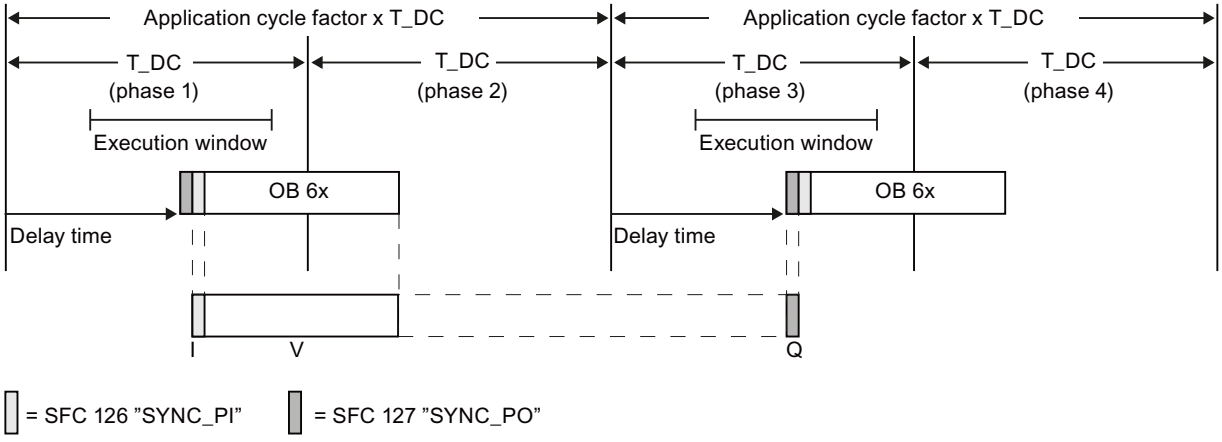

The figure shows the signal sequence with application cycle factor = 2 (application cycle = 2x data cycle) from the time of acquisition (E) through the processing (V) in the IO controller until output (A) of the process values. STEP 7 calculates the delay time. In this time, the read input data is on its way on PROFINET IO.

This model is especially well-suited for large I/O configurations with one extensive user program in OB 6x. With this model, longer calculation times are possible when processing the input data and obtaining the corresponding output data.

The constant processing time from the "input terminal" to the "output terminal" is T_I + (application cycle factor + 1) x T_DC + T_O. T_I + (2 x application cycle factor + 1) x T_DC + T_O can be guaranteed as a process reaction time.

Configuring isochronous mode (S7-1500)

Introduction

The configuring of isochronous mode of a module is described as IO device in the following based on the new ET 200MP distributed I/O series. The procedure described also applies to other distributed I/O systems (e.g., ET 200S or ET 200SP).

The IO controller is an S7-1500 CPU.

Requirement

- The network view is open.
- An S7-1500 CPU is placed (e.g., CPU 1516-3 PN/DP).
- A distributed I/O device is placed (e.g., IM 155-5 PN HF) and networked with the CPU.
- All requirements for an IRT configuration are met, namely:
  - The ports of the CPU and the IM are interconnected.
  - The RT class of the IM is set to "IRT" (area "Advanced options > Real-time settings > Synchronization").

Procedure

To create an isochronous connection between the I/O and user program, follow these steps:

1. Select the distributed I/O device (the IM 155-5 PN HF) in the network view, and switch to the device view.
2. Insert an I/O module with isochronous mode capability (module with name extension "HF", e.g., DI 16x24VDC HF - 6ES7 521-1BH00).
3. Go to "I/O addresses" area in the inspector window of the selected I/O module.
4. Make the following settings in the I/O addresses area:
   - Select the "Isochronous mode" option.
   - Select a process image partition, e.g., process image partition 1.
   - Click on the "Assigned organization block" drop-down list and click the "Add object" button.
     A dialog box for selecting organization blocks opens.
   - Select the "Synchronous Cycle" OB and confirm the selection with "OK".
     In the case of automatic number assignment, OB 61 will be generated and opened. The "Isochronous mode" area is selected in the inspector window, and you can continue directly with the setting of the application cycle and delay time and start the programming of the OB in the instruction section.
5. Add other IO devices as required, network and interconnect these devices, and adapt the configuration and the isochronous mode settings.
6. To retrieve information about calculated bandwidths or to adapt the send clock, select the subnet and navigate to the corresponding area of the sync domain management in the inspector window.

Configuring isochronous mode (S7-300, S7-400, PC)

Introduction

The configuring of isochronous mode of a module is described as IO device in the following based on the new ET 200MP distributed I/O series. The procedure described also applies to other distributed I/O systems (e.g., ET 200S or ET 200SP).

The IO controller is an S7-300/400 CPU.

Requirement

- The network view is open.
- An S7-300/400 CPU is placed (e.g., CPU 319-3 PN/DP).
- A distributed I/O device is placed (e.g., IM 155-5 PN HF) and networked with the CPU.
- All requirements for an IRT configuration are met, namely:
  - The ports of the CPU and the IM are interconnected.
  - The synchronization role of the PROFINET interface is set to "sync master" (properties of the CPU, area "PROFINET interface> Advanced options > Real-time settings > Synchronization").
  - The RT class of the IM is set to "IRT" (area "PROFINET interface > Advanced options > Real-time settings > Synchronization").

Procedure

To create an isochronous connection between the I/O and user program, follow these steps:

1. Select the CPU, and navigate to the "Interrupts > Isochronous mode interrupts" area in the inspector window.
2. Select the distributed I/O device (the IM 155-5 PN HF) in the network view, and switch to the device view.
3. Insert at least one I/O module with isochronous mode capability (module with name extension "HF", e.g., DI 16x24VDC HF - 6ES7 521-1BH00).
4. Assign a process image partition to the I/O module (inspector window, "I/O addresses" area).

5. Select the IM in the device view or network view and navigate in the inspector window to the "PROFINET interface > Advanced options > Isochronous mode" area.
   - Select the "Isochronous mode" option.
   - If you want to change the send clock or the application cycle, click the green link next to the appropriate fields. This takes you to the sync domain where these settings can be centrally changed.
   - For the Ti/To times, select whether their minimum times are to be automatically calculated or whether you want to set them manually. The Ti/To values can only be changed with manual setting.
   - In the detail overview of modules: Select the "Isochronous mode" option for modules that are to be operated in isochronous mode.
6. Add other IO devices as required, network and interconnect these devices, and adapt the configuration and the isochronous mode settings.
7. To retrieve information about calculated bandwidths or to adapt the send clock, select the subnet and navigate to the corresponding area of the sync domain management in the inspector window.

Setting isochronous mode on the subnet

Introduction

The subnet properties include not only the sync domain properties and MRP properties but also provide an overview of the isochronous mode settings.

Requirement

A PROFINET IO system with IO controller and IO devices is configured.

Procedure

To display the isochronous mode properties of the subnet, follow these steps:

1. Select the subnet.
2. Select the "Overview of isochronous mode" area in the properties.

Settings in the overview of isochronous mode

CPU	Selection of a CPU (IO controller) of the subnet
Isochronous application	Selection of an OB of the selected CPU that is assigned to an IO system.
Details of the isochronous application	

Application cycle	Indicates the application cycle (multiple of data cycle). For S7-300/400, you specify the time in the CPU properties, while for S7-1500, you specify the time in the isochronous mode OB properties.
Delay time	Indicates the time between the system clock and start of the isochronous mode interrupt OB (e.g., OB 61).
Send clock	Indicates the time period between two consecutive intervals for IRT or RT communication.
IO system no.	Indicates the no. of the PROFINET IO system assigned to the OB.
Process image partition	Indicates all process image partition numbers that are assigned to the PROFINET IO system.
Ti/To values for OB	
Ti/To values automatic	If this option is selected, STEP 7 then calculates one Ti value and one To value suitable for all modules that will be operated in isochronous mode and are assigned to the OB. You can reference these common Ti/To values in the IO devices/Modules table (Ti/To values = from the OB).
Time Ti	Time for reading-in the isochronous input data.
Time To	Time for outputting the isochronous output data.
IO devices/Modules	
Table with Ti/To values:	Isochronous mode can be activated and deactivated for each IO device/module in the IO device. Ti/To values: Three settings are possible: • Automatic: Ti/To will be calculated automatically. • Manual: Ti/To can be edited. • From the OB: Ti/To values will be taken from the OB. Ti, To: Can only be edited when "manual" setting is selected. TiMin/TiMax: Minimum or maximum time for Ti. ToMin/ToMax: Minimum or maximum time for To. Interval: Specified interval for Ti/To values.

Rules, requirements, and supplemental conditions

What you need to be aware of for configuring isochronous mode is

Take into consideration the following notes on configuring isochronous mode.

Components and PROFINET interfaces

- Isochronous mode is only possible with the interfaces integrated in the CPU. For S7-1500 CPUs with two PROFINET interfaces, only the X1 interface can be used for isochronous mode. Isochronous mode is not possible with S7-1500-CPs/S7-1500-CMs.
- A full isochronous mode from "terminal" to "terminal" is only possible if all components participating in the chain support the "isochronous mode" system property. When selecting in the catalog, make sure you check the "isochronous mode" or "processing in isochronous mode" entry in the information field of the module.
- If you operate an IO device in isochronous mode (i.e., you have assigned it the sync slave role, for example), the IO device must contain at least one module or submodule operated in isochronous mode.

Subnet-overlapping isochronous mode

- Subnet-overlapping isochronous mode is not currently possible.

Process image partitions

- Isochronous I/O can only be processed in process image partitions. Without the use of process image partitions isochronous consistent data transfer is not possible. Compliance with configuration limits is monitored because there is a limit on the number of IO devices and the number of bytes on the DP System per process image partition.
- The addresses of the isochronous I/O have to lie in a process image partition.
- The process image partition for isochronous I/O may also contain non-isochronous I/O.
- S7-300/400: More than one process image partition is possible per isochronous mode interrupt OB or IO system.
- S7-1500: Exactly one process image partition is possible per isochronous mode interrupt OB or IO system. Only modules or submodules of the same IO system or the same isochronous mode interrupt OB may be contained in the same process image partition.

Times

- If isochronous mode is active, the update time in the IO devices calculated by STEP 7 must always be identical to the send clock. It is not possible to manually change the update time.
- The time for the application cycle is limited to 32 ms.
- The delay time for the isochronous mode interrupt OB must be between 0 and (data cycle - 1 µs).

Configuring I-devices

I-Device Functionality

I-Device Functionality

The "I-device" functionality (intelligent IO device) of a CPU allows data to be exchanged with an IO controller and therefore to use the CPU, for example, as an intelligent preprocessing unit of subprocesses. The I-device is linked as an IO device to a "higher-level" IO controller.

The preprocessing is handled by the user program on the CPU. The process values acquired in the central or distributed (PROFINET IO or PROFIBUS DP) I/O are preprocessed by the user program and made available via a PROFINET IO device interface to the CPU or the CP of a higher-level station.

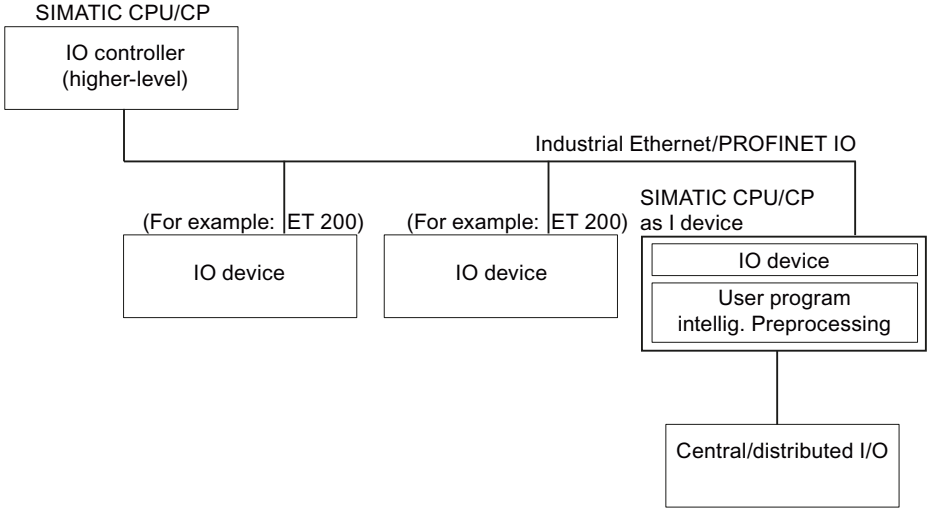

"I-device" naming conventions

In the remainder of this description, a CPU or a CP with the I-device functionality is simply called an "I-device".

Properties and Advantages of the I-Device

Fields of application

Areas of application of the I-device:

- Distributed processing
  A complex automation task can be divided into smaller units/subprocesses. This makes the process easier to handle because the subtasks are simpler.
- Separating subprocesses
  Complicated, widely distributed and extensive processes can be subdivided into several subprocesses with manageable interfaces by using I-devices. These subprocesses can be stored in individual projects if necessary, which can later be merged to form one master project.
- Know-how protection
  Components can only be delivered with a GSD file for the I-device interface description instead of with a project. The know-how of the user program must no longer be published.

Properties

Properties of the I-device:

- Unlinking projects
  Creators and users of an I-device can have completely separated automation projects. The interface between the projects forms the GSD file. This allows a link to standard IO controllers via a standardized interface.
- Real-time communication
  The I device is provided with a deterministic PROFINET IO system via a PROFINET IO interface and therefore supports RT (real-time communication).

Advantages

The I-device offers the following advantages:

- Simple linking of IO controllers without additional software tools
- Real-time communication between SIMATIC CPUs and to standard IO controllers.
- By distributing the computing power among several I devices, the required computing power of the individual CPUs and the IO controller can be reduced.
- Lower communications load by processing process data locally.
- Clarity by processing the subtasks in separate CPUs or projects.

Data Exchange between Higher- and Lower-level IO System

Introduction

The next section shows the data exchange between a higher- and lower-level IO system.

Transfer areas

The data for communication between IO controller and I device is made available in the transfer areas. A transfer area contains an information unit that is exchanged consistently between IO controller and I device.

Application transfer areas are an interface to the user program of the I-device CPU. Inputs are processed in the user program and outputs are the result of the processing in the user program.

The following figure shows the data exchange between a higher- and lower-level IO system. The individual communication relations are explained below based on the numbers.

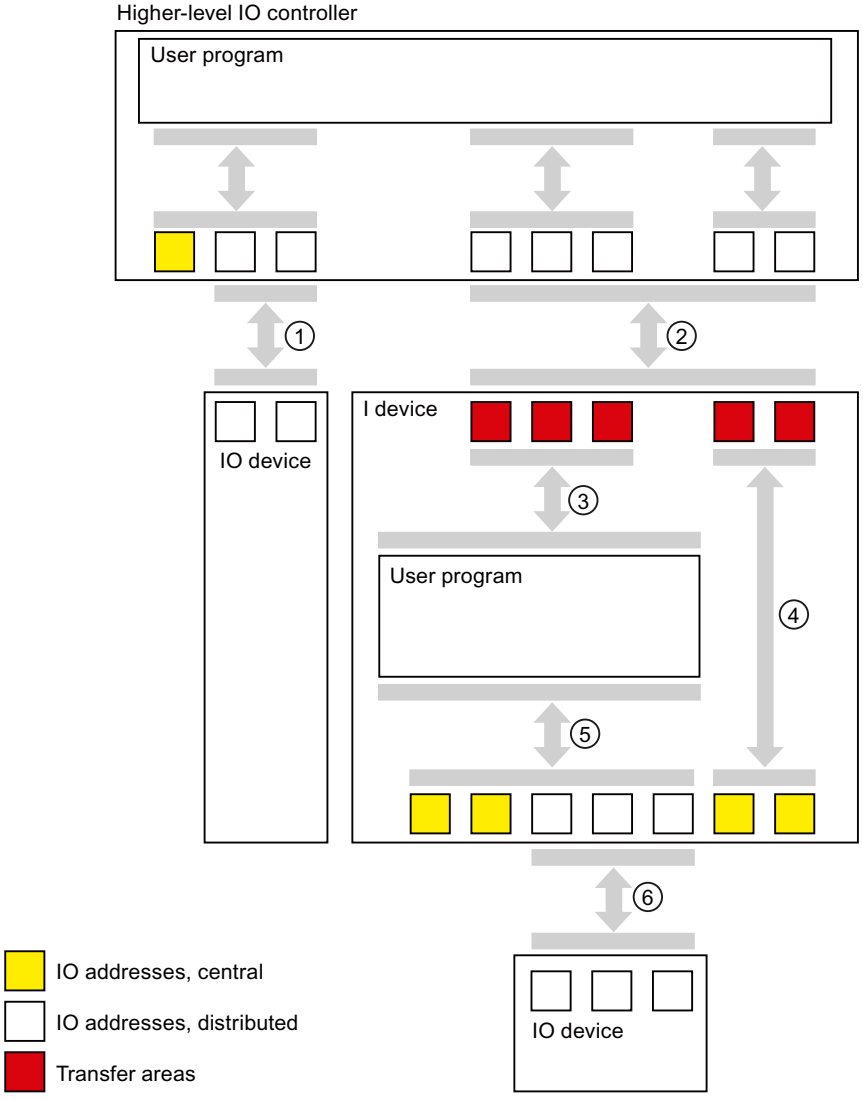

① Data exchange between higher-level IO-controller and normal IO-device

In this way, the IO controller and IO device exchange data through PROFINET.

② Data exchange between the higher-level IO controller and I-device

In this way, the IO controller and the I-device exchange data through PROFINET.

The data exchange between a higher-level IO controller and an I-device is based on the conventional IO controller / IO device relationship.

For the higher-level IO controller, the transfer areas of the I-devices represent submodules of a preconfigured station.

The output data of the IO controller is the input data of the I-device. Analogously, the input data of the IO controller is the output data of the I-device.

③ Transfer relationship between the user program and the transfer area

In this way, the user program and the application transfer area exchange input and output data.

④ Transfer relationship between the transfer area and the I/O of the I-device

This is not currently supported in the configuration.

⑤ Data exchange between the user program and the I/O of the I-device

In this way, the user program and the central / distributed I/O exchange input and output data.

⑥ Data exchange between the I-device and a lower-level IO device

In this way, the I-device and its IO devices exchange data. The data transfer is via PROFINET.

Configuring diagnostics addresses (S7-300, S7-400)

Introduction

When assigning diagnostics addresses for PROFIBUS IO, you must ensure that IO diagnostics addresses are assigned to both the IO controller and the I-device.

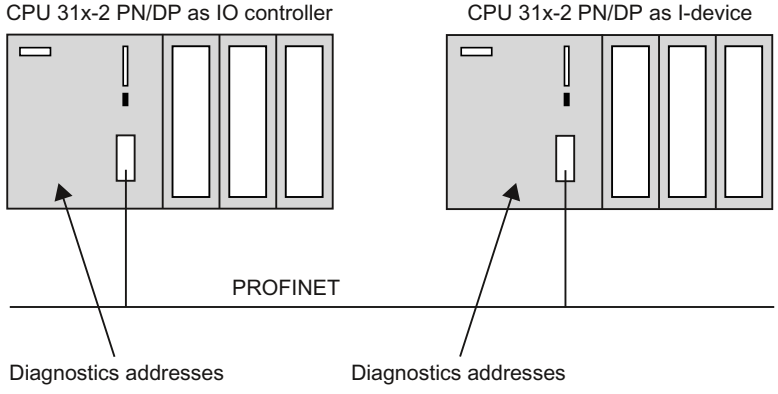

Diagnostics addresses

A distinction is made between diagnostics addresses on the IO controller and those on the I-device:

- Diagnostics addresses on the IO controller
  The IO controller receives information about the status of the IO device or a bus interruption via the diagnostics addresses assigned to the IO controller.
- Diagnostics addresses on the I-device
  The I-device receives information about the status of the IO controller or a bus interruption via the diagnostics addresses assigned to the I-device.

STEP 7 automatically assigns diagnostics addresses for the respective CPUs when the controller-I-device configuration is specified. The following describes how to change the specified diagnostics addresses.

Tip: In order to obtain an overview of all addresses assigned for the CPU, including diagnostics addresses, select the "Address overview" area in the inspector window of the selected CPU.

Diagnostics addresses of the I-device communication

A table with the diagnostics addresses of the I-device communication is displayed on the I-device. The identifier for the diagnostics address is the name of the PLC and the transfer area with the lowest subslot number. When you activate the parameter assignment of the PN interface by the higher-level IO controller for the operating mode of the PROFINET interface on the I-device, the diagnostics addresses for the PROFINET interface and its ports are displayed as well. You can edit the diagnostics addresses used in the table.

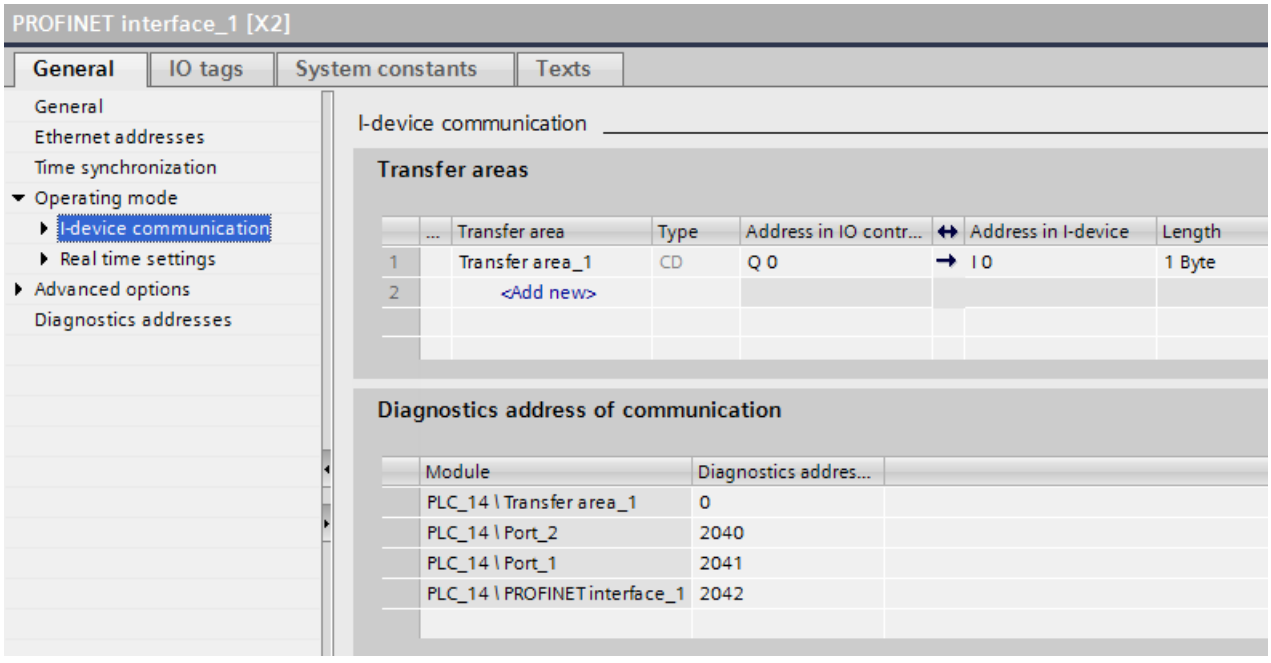

The table is only displayed when the IO controller is a CPU 300/400:

IO controller	Display of the diagnostics addresses
CPU 300/400	Local addresses of the I-device
CPU 1200/1500	No display

Requirement

- You must be in the network view.
- You have created an IO system with IO controller and I-device.

Procedure

To edit the diagnostics addresses for the I-device communication, follow these steps:

1. Select the IO controller or the I-device.
2. In the I/O communication table, select the line showing "I-device" mode.
3. Edit the diagnostics addresses under "Properties > I-device communication > Diagnostics address of communication" in the Inspector window.

Or:

1. Select the I-device.
2. Edit the diagnostics addresses under "Properties > PROFINET interface > Mode > I-device communication > Diagnostics address of communication" in the Inspector window.

You can select and edit the fields containing the values of the diagnostics addresses both for the device and the controller.

Additional information

For further information on diagnostics addresses, refer to the manuals for the individual S7 CPUs.

Configuring the I device

Introduction

There are basically two possibilities for configuration:

- Configuration of an I-device within a project
- Configuration of an I-device that is used in another project or engineering system.

STEP 7 allows to you configure an I-device for another project or for another engineering system by exporting a configured I-device to a GSD file. You import the GSD file in the other project or the other engineering system as with other GSD files. The transfer areas for the data exchange, among other data, is stored in this GSD file.

Configuration of an I-device within a project

1. Drag and drop a PROFINET IO controller from the hardware catalog into the network view.
2. Drag and drop a PROFINET IO controller which can also be configured as IO device from the hardware catalog into the network view. This device is configured as an I-device (for example, CPU 319-3 PN/DP, FW 3.2).
3. Select the PROFINET interface for the I-device.
4. In the area navigation of the Inspector window, select the "Operating mode" entry and select the "IO device" check box.
5. You now have the option of selecting the IO controller in the "Assigned IO controller" drop-down list.
   Once you have chosen the IO controller, the networking and the IO system between both devices are displayed in the network view.
6. With the "Parameter assignment of PN interface by higher-level IO controller" check box, you can specify whether the interface and its ports are set by the I-device itself or a higher-level IO controller.
   If you operate the I-device with a lower-level IO system, then the parameters of the I-device PROFINET interface (for example, port parameters) cannot be set by the higher-level IO controller.
7. Configure the transfer areas. Those are the I/O areas over which the I-device exchanges data with the higher-level IO controller. The transfer areas are found in the area navigation section "I-device communication".
   - Click in the first field of the "Transfer areas" column. STEP 7 assigns a default name which you can change.
   - Select the type of communication relation, for example, CD for "Controller-device communication relation".
   - Addresses are automatically preset; you can correct addresses if necessary and determine the length of the transfer area which is to be consistently transferred.
8. A separate entry is created in the area navigation for each transfer area. If you select one of these entries, you can adjust the details of the transfer area, or correct them and comment on them.
   If the CPU supports it, you can also activate the "Bi-directional address mapping" function. In this case the automatic address areas will be arranged "in the opposite direction", so that CPU input and output areas are occupied symmetrically.

Configuring an I-device with a GSD file

If you use an I-device in another project, or if the I-device is used in another engineering system, then configure the higher-level IO controller and the I-device as described above. The I-device must be exported as GSD file afterward. The generated GSD file represents the configured I-device in another project.

To do this, follow the description for Export of GSD files (Page 2925).

Note

PROFIsafe transfer module of the type "F-PS"

If you have inserted a PROFIsafe transfer module of the type "F-PS", the I-device must not be assigned to an IO controller in this project. A GSD file must be exported. You can use this GSD file to set all PROFIsafe parameters in the other project, and to operate the I-device on an IO controller with PROFIsafe capability.

See also

Configuring an I-device as a shared device (Page 2968)

Multiple assignment of IO controllers (Page 2927)

Exporting an I-device as a GSD file

Introduction

If you want to use an I-device in another project or if the I-device is used in another engineering system, you must export the I-device as GSD file. The GSD file contains the description of the I-device and can subsequently be imported and used in the new project.

Requirement

- An I-device with I-device communication and transfer area was configured.
- The changes to the hardware were compiled.

Procedure

To export an I-device as GSD file, follow these steps:

1. Select the I-device or its PROFINET interface in the hardware and network editor.
2. In the Inspector window, select the "Export" command at the PROFINET interface "Operating mode > I-device communication". The dialog for export of the I-device as GSD file opens.

3. You can now specify the name and a description for the I-device representative and change the storage path of the GSD file.

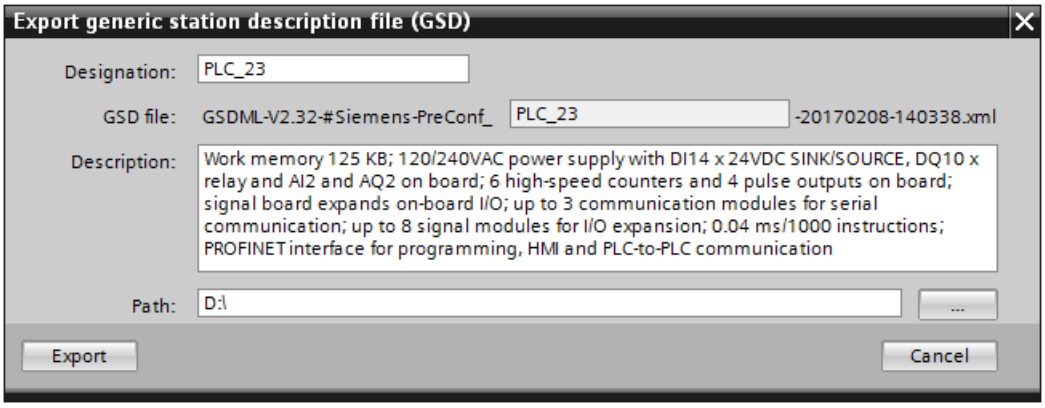

Designation: The designation stands for a text that is used as part of the name of the GSD file. This designation will help you identify your GSD file more easily later. The default setting is the PLC name from the current project.

Note

With a later import of the GSD file into another project, each imported I-device is created with a new designation as new entry in the hardware catalog. If the designation already exists in the hardware catalog, the I-device is merely added as additional version of an already existing I-device.

Description: Here you can edit the description. After import of the GSD file to another project, this text is displayed in the hardware catalog at the I-device in the "Description" box. The default is the description of the PLC from the current project.
Path: Here you can enter where you want to store the GSD file. Select a path that is easy for you to remember.

4. Click the "Export" button. The I-device is now exported as a GSD file.

Now you have exported the I-device as a GSD file and can use it in another project.

Composition of the name of the exported GSD file

You cannot freely select the name of the GSD file because it is subject to a specified scheme. You can only influence the part of the name under which you can find the I-device in the hardware catalog after an import of the GSD file. The structure of the name for the exported GSD file is explained using the following example:

The name of an exported GSD file for I-devices can look as follows:

"GSDML-V2.32-#Siemens-PreConf_PLC_23-20170208-140313.xml"

Name component	Can be edited?	Description
GSDML	No	String at the start of each GSD file for I-devices
V2.32	No	Version of the GSDML schema
Siemens	No	Manufacturer
PreConf	No	Identifier for the GSD file from the export function

Name component	Can be edited?	Description
PLC_23	Yes	Device designation can be changed. The I-device can be found under this name after import in the hardware catalog of the other project.
20170208	No	Identifier of the version as date in the format year-month-day (YYYYMMDD). The system data is used as reference.
140313	No	Identifier of the version as time in the format hour-minute-second (hhmmss). The UTC system time is used as reference.
.xml	No	File extension

Import of the exported GSD file in another project

You import the GSD file with the exported I-device by installing the GSD file as described here. After you have selected and installed the GSD file with the I-slave under "Options > "Manage general station description files (GSD) > Installed GSDs", you can find the I-device under "Additional field devices > PROFINET IO" in the hardware catalog under the name that you have specified as designation in the export dialog. I-devices with the same name are set as versions of the same device in the hardware catalog. You can identify these different versions with the identifier of the version in the file name or by means of a different description. To display these versions, use the "Version" drop-down list after selection of the I-device.

Selected I-devices can be dragged into the configuration just like other devices of the hardware catalog.

GSD files are always saved together with the project, which means all information relevant for display of the device (including symbols) is also available in the saved project.

See also

Configuring the I device (Page 2923)

Multiple assignment of IO controllers

Introduction

If several PROFINET IO interfaces with I-device functionality exist in an S7-1500 PLC, you can also assign several IO controllers to this device.

As of STEP 7 V14, the fact that more than one higher-level IO controller is available for an I-device is also visible: The link "Multiple assignment" is displayed at such devices in the network view.

Examples of PROFINET IO interfaces with I-device functionality:

- S7-1500 CPU, in particular with two PROFINET IO interfaces (for example, CPU 1518-4 PN/DP, CPU 1517-3 PN/DP support PROFINET IO (and I-device functionality) at the interfaces X1 and X2)
- CM 1542-1

The effects of a multiple assignment are described below.

Configuring a device with assignments to several IO controllers

The following section uses an example configuration to show the effect of the assignment of an I-device to several IO controllers.

1. Drag-and-drop several PROFINET IO controllers (for example, CPU 1518-4 PN/DP) from the hardware catalog into the network view.
2. Drag-and-drop a PROFINET IO controller which can also be configured as IO device from the hardware catalog into the network view (for example, CPU 1518-4 PN/DP). You can also insert a CM 1542-1 into this station and also configure the PROFINET interface as an IO device.
3. Select the PROFINET interface of a module.
4. In the area navigation of the Inspector window, select the "Operating mode" entry and select the "IO device" check box.
5. You now have the option of selecting the IO controller in the "Assigned IO controller" drop-down list.
6. Repeat steps 3 and 4 for additional PROFINET interfaces.
   Once you have chosen the IO controller or the PROFINET IO interface of the IO controller, the networking and the IO system between both devices are displayed in the network view.
   As soon as you assign a further PROFINET IO interface to a different IO controller, the "Multiple assignment" link is displayed in the device instead of the name of the higher-level IO controller.
   When you click this link, the selection of assigned IO controllers is displayed and you can place the focus on this device by clicking one of these IO controllers.

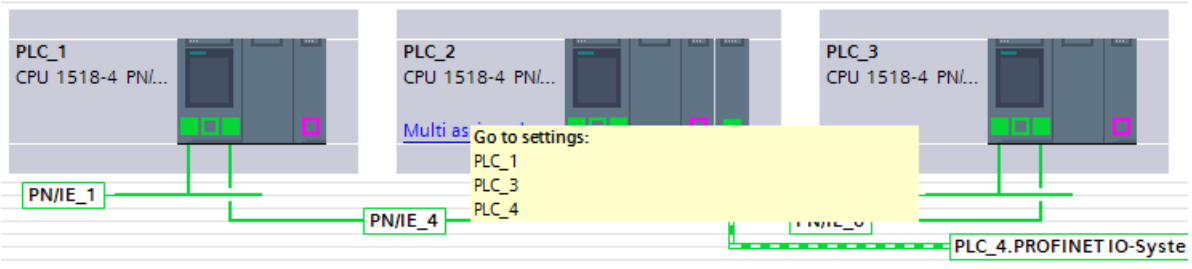

The remaining settings (further parameter assignment of the IO interface, configuration of the transfer areas) are only affected by this function extension in as far as the selection of the devices or of the IO systems is facilitated by a unique ID, also in the case of CPUs with several PROFINET IO interfaces.

If you have assigned the interfaces to exactly one S7 subnet, several IO systems exist on this subnet.

If you click the subnet, the display for selecting the IO systems to be highlighted is shown.

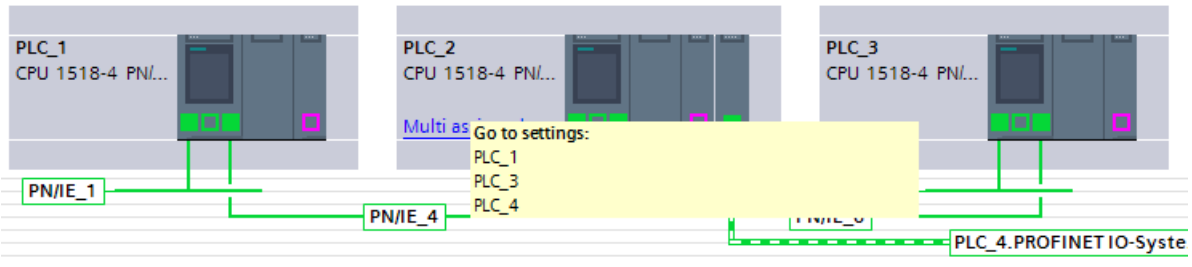

Constraints on the Use of I-Devices

Boundary conditions

When using I-devices, certain constraints must be taken into account in certain situations.

Bandwidth

The number of addresses of the configured transfer areas is reflected in the usable bandwidth of the I-device:

Bandwidth of the transfer areas + bandwidth of the lower-level IO system = total bandwidth used on the I-device.

If the address space of the transfer areas is too large, there is not enough bandwidth for the lower-level IO system to achieve fast update times.

Tip: Keep the address space of the transfer area as small as possible.

Rules for RT and IRT communication

IO systems with I-devices are also suitable for setting up real-time applications with RT communication. In the following, a hierarchical system is required, consisting of a higher-level IO controller to which an I-device is assigned (higher-level IO system). IO devices are, in turn, assigned to the I-device (lower-level IO system). The following rules apply:

Both the higher-level and the lower-level IO system support RT communication. You can use RT communication for both IO systems at the same time.

IRT communication can be combined with RT communication. However, IRT communication cannot take place in both IO systems at the same time.

Example:

- If the option "Parameter assignment for the PN interface and its ports on the higher-level IO controller" is activated on the I-device, the I-device can be operated on the higher-level IO controller with the RT class "IRT". IRT communication is not possible for the lower-level IO system of the I-device in this case.
- If the option "Parameter assignment for the PN interface and its ports on the higher-level IO controller" is deactivated on the I-device, the I-device can be operated on the higher-level IO controller with the RT class "RT". IRT communication is possible for the lower-level IO system.

If you configure the I-device on an IO system with the RT class "RT", you cannot set odd send clocks.

Rules for data access

- The higher-level IO controller can access transfer areas when the I-device is in RUN.
- Only the application transfer areas may be accessed in the I-device CPU:
  - Access to input-application transfer areas possible if the higher-level IO-controller is in RUN.
  - Access to output application transfer areas possible independent of the operating state of the higher-level IO controller.

IP address parameters and device name

Just like every other IO device, an I-device also requires IP address parameters / device name to be able to communicate via PROFINET. IP address parameters consist of three parts, the IP address itself, the subnet mask and the address of the router (gateway).

The IP address parameters / device name for an I-device can be assigned in two ways - via the project or directly on the device.

IP address/device name is set in the project:

- The IP address parameters / device name are permanently assigned during configuration (in the I-device project) in STEP 7. This is the standard method.

Set IP address/device name on the device:

- IP address parameter / device name via DCP: The IP-address parameters / device name are assigned via DCP (Discovery and Configuration Protocol). This can happen in two ways:
  - Through a setup tool such as PST or STEP 7 (via accessible devices).
  - Through the higher-level I/O controller
- IP address parameter / device name via user program: The assignment of the IP address parameters occurs in the user program of the I-device CPU (via the instruction "IP_CONFIG").

Note

Topology illustration

If the device name and the IP address parameters are adapted for an I-device directly on the device, the I-device can be displayed multiple times in the topology editor.

Diagnostics and interrupt characteristics

Diagnostics and interrupt characteristics

S7 CPUs have numerous diagnostics and interrupt functions that can, for example, report errors/faults or failures of underlying IO systems. These diagnostics alarms reduce down times and simplify localization and elimination of problems.

Diagnostics of the I-device on the higher-level IO controller and in the I-device

The following mechanisms are available in the higher-level IO controller and the I-device CPU for diagnosing the state of the respective partner:

- Pull/plug interrupt OB (OB 83) for "Return of submodule" interrupt
- Program execution error OB (OB 85) for, for example, process image transfer errors (only S7-300/400)
- Rack failure OB (OB 86) for the failure of expansion units, IO systems or distributed I/O devices (station failure, station return)
- I/O access error OB (OB 122), optional for S7-1500
- Local troubleshooting (only S7-1500)

Note

The diagnostics alarms of the I/O can be processed in the user program of the I-device CPU and passed on from there to the higher-level IO controller via transfer areas.

Operating state changes and station failures / station returns

In the following tables, you can see the effects which the operating state change or the failure of individual CPUs (I device, IO controller) have on each other.

Please note that an S7-1500 CPU calls the OB 122, if it exists, in the event of errors due to direct I/O access to the transfer areas (for example, because the other CPU goes into STOP). However, in contrast to an S7-300/400 CPU, the CPU does not go into STOP if the OB 122 does not exist. With an S7-1500 CPU, local troubleshooting within the block in which the error occurred is also possible. This situation is indicated in the table with the abbreviated wording "Call of OB 122".

Events

Initial status	Event	I device response	Response of higher-level controller
Both CPUs are in RUN operating state (higher-level IO controller and I-device).	The I device CPU changes to STOP.	/	**S7-300/400/1500:** Error code in RET_VAL at updating of the process image with the instruction UPDAT_PI or UPDAT_PO. OB 122 is called in the case of direct access to the transfer areas to the I-device (optional for S7-1500). **S7-300/400**: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer to the I-device by the system.
I-device CPU is in STOP mode, higher-level IO controller is in RUN mode.	The I device CPU is starting up.	**S7-300/400/1500:** Call of OB 100 (startup). Call of OB 83 for input transfer areas to the higher-level IO controller. Until OB 83 is called, access errors occur in the event of direct access to the input transfer areas to the higher-level IO controller: Call of OB 122. **S7-300/400:** With a configured OB85 call if I/O access error occurs, OB 85 continues to be called at a process image transfer to the higher-level IO controller by the system until OB 83 is called.	**S7-300/400/1500:** Call of OB 83: For all transfer areas to the I-device. Until OB 83 is called, OB 122 continues to be called in the event of direct access to the transfer areas to the I-device. **S7-300/400:** With a configured OB85 call if I/O access error occurs, OB 85 continues to be called at a process image transfer to the I-device by the system until OB 83 is called.
Both CPUs are in RUN mode (higher-level IO controller and I-device).	The higher-level IO controller changes to STOP	**S7-300/400/1500:** Error code in RET_VAL at updating of the process image with the instruction UPDAT_PI or UPDAT_PO. OB 122 is called in the event of direct access to the input transfer areas to the higher-level IO controller. Note: Output transfer areas can still be accessed. **S7-300/400**: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the input transfer areas to the higher-level IO controller by the system.	/

Initial status	Event	I device response	Response of higher-level controller
Higher-level IO controller is in STOP mode, I-device CPU is in RUN mode.	The higher-level IO controller starts up.	**S7-300/400/1500**: Call of OB 83 for input transfer areas to the higher-level IO controller. Until OB 83 is called, OB 122 continues to be called in the event of direct access to the input transfer areas to the higher-level IO controller. **S7-300/400**: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the input transfer areas to the higher-level IO controller by the system.	Call of OB 100 (startup).

Initial status	Event	I device response	Response of higher-level controller
Both CPUs are in RUN (higher-level IO controller and I-device).	Station failure I device, for example, through bus disruption	S7-300/400/1500: If the I device continues to run without a bus connection: Call of OB86 (station failure). Error code in RET_VAL at updating of the process image with the instruction UPDAT_PI or UPDAT_PO. OB 122 is called in the event of direct access to the transfer areas to the higher-level IO controller S7-300/400: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the transfer areas to the higher-level IO controller by the system.	S7-300/400/1500: Call of OB86 (station failure). Error code in RET_VAL at updating of the process image with the instruction UPDAT_PI or UPDAT_PO. OB 122 is called in the case of direct access to the transfer areas to the I-device. S7-300/400: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the transfer areas to the I-device by the system.
Both CPUs are in RUN (higher-level IO controller and I-device). The communication connection between the IO-controller and I-device is disrupted (bus disruption).	The bus connection between the IO-controller and I device is reestablished and the I device is recorded in the usage data traffic again.	S7-300/400/1500: Please note the information under this table about the special considerations when the I-device is powered up. Call of OB 86 (station return). Call of OB 83 for input transfer areas to the higher-level IO controller. Until OB 83 is called, OB 122 continues to be called in the event of direct access to the input transfer areas to the higher-level IO controller. S7-300/400: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the input transfer areas to the higher-level IO controller by the system.	S7-300/400/1500: Call of OB86 (station return) Until OB 86 is called, OB 122 continues to be called in the event of direct access to the transfer areas to the I-device. S7-300/400: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the transfer areas to the I-device by the system.

Note

Special feature when starting up the I device

In contrast to the station return alarm from IO devices in the IO controller, which is covered by calling OB86, the station return alarm of a higher-level IO controller in the I device is divided into 2 parts:

1. Call of OB 86: The initial values for the outputs of the I device are set. The input values, however, are not yet valid; they will first be valid when opening the OB86 in the higher-level IO controller.
2. Call of OB 83 for each input transfer area. The validity of an input transfer area is shown with this call. I-device startup is not complete until OB 83 is called for the input transfer areas. This step can be delayed or not occur at all in the following situations:
   – Higher level IO controller is in STOP: OB 83 is not called until the STOP → RUN transition of the higher-level IO controller.
   – The IRT communication is faulty, for example, because the sync master has failed due to topology errors, etc.: The OB 83 is not started until the IRT communication has been established again.

See also

Insert/remove module interrupt organization block (OB 83) (Page 846)

Rack failure organization block (OB 86) (Page 856)

I/O access error organization block (OB 122) (Page 875)

Priority class error organization block (OB 85) (Page 851)

Online diagnostics for I devices

I device online diagnostics is basically similar to online diagnostics of "normal" IO devices.

Displaying diagnostic status by going online

You will receive a selection of the following information depending on which device you selected when you went online:

- If you have selected a higher-level IO controller:
  The diagnostics status of the I device is shown in the project navigation below the IO controller in the "Distributed I/O" folder. No diagnostics status will be shown in the project navigation for the I device itself.
- You have selected the I device that this IO controller is assigned to.
  The diagnostics status of the I device is shown in project navigation. No diagnostics status of the I device is shown in the project navigation below the IO controller in the "Distributed I/O" folder.

Online and diagnostics view

You will receive a selection of the following information depending on which device you selected in the online and diagnostics view:

- You have selected a higher-level IO controller and are connected online with this device: The online and diagnostic view displays the I device diagnostics data from the higher-level IO controller. The display corresponds to the display of a "normal" IO controller.
- You have selected the I-device that this IO controller is assigned to and are connected online with this device.
  The online and diagnostics view displays I device diagnostics data from the local view, that is, the view of the I device. The display corresponds to the display of a "normal" IO device.

Display in the list of accessible devices

In the list of accessible devices the I device will be displayed as a "normal" CPU.

Changing IO devices during operation

Docking station - changing IO devices during operation (S7-300, S7-400, S7-1500)

Using changing IO devices during operation ("changing partner ports") in a docking station

The following Figure shows an automation cell with a docking system and several docking units.

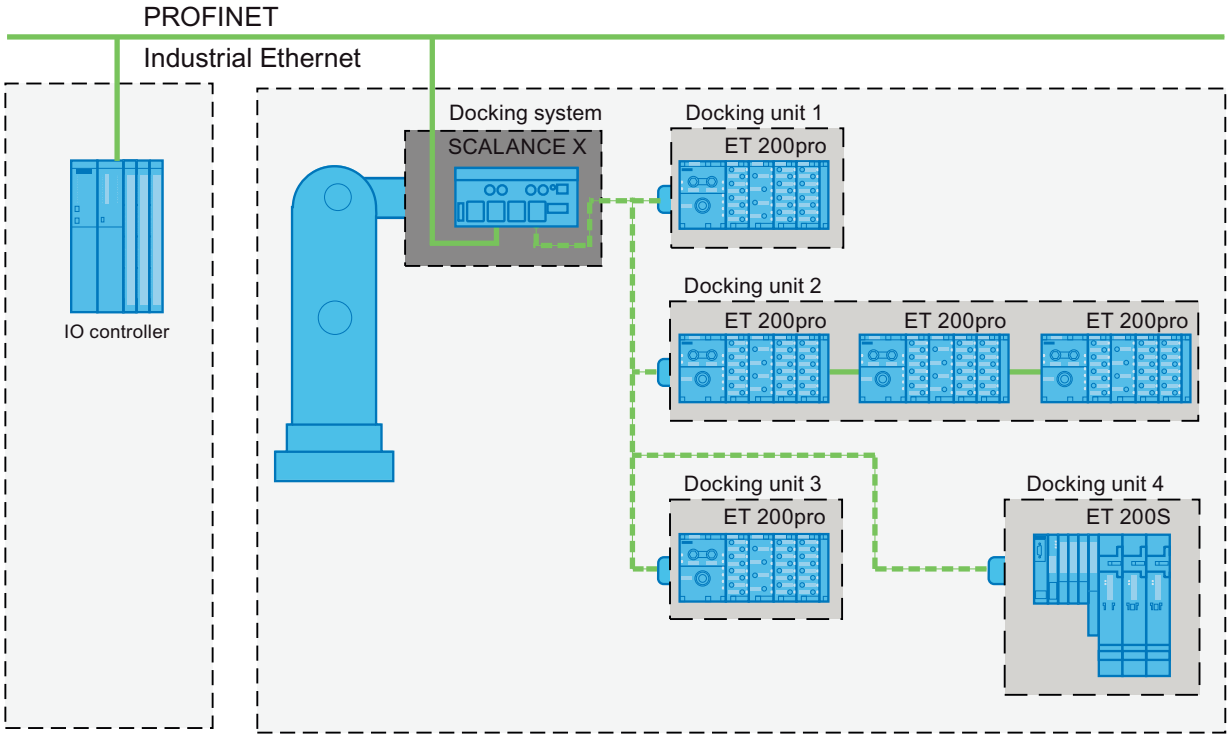

Figure 1-92 Changing IO devices (partner ports) in a docking station

Applicative conditions

The following points should be observed when realizing a docking system with changing IO devices during operation:

- The IO devices of all docking units must be deactivated as default in the configuration.
- Only one docking unit may be active at any given time, in other words only the IO devices of one docking unit can be activated. All IO devices of other docking units must be deactivated or become deactivated before the IO devices of a docking unit can be activated. This is done with the instruction "D_ACT_DP".
- A physical connection to this docking unit and its IO devices must be created in order to activate a docking unit. The IO devices are then switched on (power on). All the IO devices of this docking unit must at the same time be activated in the user program with "D_ACT_DP".
- After the message "IO device activated" you can access the IO device by using the command "Direct I/O access".
- Call the instruction "D_ACT_DP" to activate and deactivate the IO device as close as possible to the start of the OB 1 cycle.

Area of application for changing IO devices during operation

You can use the PROFINET functionality "Changing IO devices during operation" ("changing partner ports"), e.g. for tool change for robots. Typical tools are, for example:

- welding guns and
- holding tools for production parts.

Note

Number of changing IO devices during operation ("changing partner ports") - number of docking units

If you wish to achieve the shortest possible tool-change times, you must observe the following points that are dependent on the CPU or the CP that is being used:

- Only those IO devices that have been configured with the PROFINET function "Prioritized startup" can run in an optimized fashion. The number of IO devices with configuration for this PROFINET function is restricted.
- Only a specific number of IO devices can be enabled at the same time (number depends on the available resources for instruction "D_ACT_DP"). A docking unit should therefore not include more than this number of IO devices. If more IO devices are operated in a docking unit, the IO devices must be enabled one after the other, which takes correspondingly longer.

Example: An S7 CPU 319-3 PN/DP can operate a maximum of 32 IO devices with prioritized startup and can simultaneously activate 8 IO devices with the instruction "D_ACT_DP".

Therefore, for a scheduled optimum use, a docking unit should include no more than 8 IO devices and no more than 32 IO devices should be used in all the changing docking units.

Restriction in the interconnection

The interconnection with a partner port is not possible in the following cases:

- The partner port does not have a suitable type of cable. In this case, a media convertor from the catalog must be inserted.
- The partner port is locked (disabled).
- The two ports that are to be interconnected belong to the same interface (it is possible only to interconnect the ports from different interfaces in a station).
- You are trying to create a ring connection with a module that is not redundant-capable.
- The two ports that are to be interconnected belong to different Ethernet subnets.
- The port of a PROFINET interface of an IO controller cannot be directly configured with the function "Changing IO devices during operation" ("changing partner port").

Configuring docking units in a docking system (S7-300, S7-400, S7-1500)

Configuring docking units in a docking system

To configure more docking units in a docking system, follow these steps:

1. Configure your hardware configuration as usual; first, consider the IO devices which correspond to the changing partners to be normal IO devices.
2. In the inspector window, open the properties of the ports on which you would like to operate the changing IO devices (changing partner ports).
3. Under "General > PROFINET interface > Advanced settings > RJ45 100 Mbit Port [Xn Pm] > Port interconnection", select the check box "Alternative partner".
4. Insert the alternative partner in the table, as described in the "Interconnecting ports" chapter.

See also

Interconnecting ports by changing partner ports (Page 2938)

Optimizing startup time (Page 2940)

Interconnecting ports by changing partner ports

Requirement

- All devices involved support topology configuration
- The IO controller, the changing IO devices (docking unit) and the switch (docking system) to which the changing IO devices are to be operated, must support this feature.
- The docking unit must be connected with a switch that supports the PROFINET function "prioritized startup" (e.g., from the SCALANCE X200IRT family).

Example

- IO controller: CPU 416-3 PN/DP, CPU 1516-3 PN/DP
- Docking system: Scalance X-200IRT
- Docking units (changing IO devices): for example, 2 x ET 200pro (IM 154-4)

Procedure

In the description of the procedure, it is assumed that the configuration matches that of the above-described example. To interconnect ports, follow these steps:

1. Connect all PROFINET interfaces of the devices to each other in the network view.
2. In the network view or device view, select the docking system (Scalance X200 IRT) whose port you want to interconnect.
3. Open the device properties in the Inspector window.
4. Under "Properties > PROFINET interface > Advanced settings > RJ45 100-megabit port [Xn Pm] > Port interconnection" choose the partner port for the CPU (IO controller) from the "Partner port" drop-down list.
   The ports of the CPU and the docking system (Scalance) are interconnected.
5. The docking system is still selected: For any other unconnected port, now select the partner ports for the IO devices:
   - Select the "Alternative partner" check box.
     A table for the partner port is displayed.
   - Within the table, click the line <Add alternative partner...> to open a drop-down list.
     The drop-down list shows all devices with their unconnected ports for selection.
   - Select the port of the first docking unit.
   - Click the next line in the table and select another alternative partner port.
   - Repeat the steps outlined above until all required partner ports are recorded in the table.

Note

Use of a CP 1243-1 (WAN-CP)

When a CPU S7-1200 with CP 1243-1 is used, the procedure is the same as above, except the Ethernet interface of the CP is used instead of the PROFINET interface of the IO controller.

Note

CAx export of a PROFINET configuration for tool changer projects

When you have a PROFINET configuration in which the "Alternative partner" check box selected, you must observe the following when exporting CAx files: The setting "Alternative partner" is not supported during the export. This means that this setting or the topological connection is missing in the subsequent reimport.

Interconnecting several IO devices with changing IO devices during operation

Configure IO devices on a docking unit which are interconnected in line with IO devices changing during operation ("Alternative partner") with the port of the changing IO device as usual; this means the IO device interconnected in line then has no changing partner port.

Deleting the partner port

To remove the interconnection of the partner port that changes during operation, follow these steps:

1. In the table of alternative partners, click the row that contains the alternative partner port that is to be deleted and open the drop-down list for the partner port (black triangle).
2. Select "Any partner" from the list.
   The alternative partner port is removed from the table.

See also

What is prioritized startup? (Page 2940)

Optimizing startup time

What is prioritized startup?

Definition

Prioritized startup, also known as "fast startup", describes the PROFINET functionality for accelerating the startup of IO devices in a PROFINET IO system with RT and IRT communication. It reduces the time that the correspondingly configured IO devices require in order to return to cyclic user data exchange in the following cases:

- After the supply voltage has returned
- After a station has returned
- After IO devices have been activated

Benefits

The PROFINET functionality "prioritized startup" enables PROFINET IO applications in which machine parts or tools and their IO devices have been permanently replaced. Waiting times of several seconds between the scheduled processes of the restart are reduced to a minimum by this optimization. This accelerates the production process with removable IO devices, e.g. in tool-changer applications and enables a greater throughput in production.

The PROFINET functionality "prioritized startup" also offers a considerable increase in performance for applications where a quick startup time of the IO devices after "power on" or after station failure / station return is required or when activating IO devices.

Properties

You can realize the following properties with the PROFINET function "Prioritized startup":

- Communication readiness of the IO devices (distributed I/O) up to minimum 500 ms.
- You can use the prioritized startup for IO devices (distributed I/O) both with RT as well as with IRT communication.

Startup times

The length of the startup time of an IO device (distributed I/O) with the PROFINET function "Prioritized startup" is dependent on the following points:

- IO devices used (distributed I/O)
- IO structure of the IO devices (distributed I/O)
- Used modules of the IO devices (distributed I/O)
- Used IO controller
- Used switch
- Port setting
- Cabling
- Configured RT class of the IO device

Note

Prioritized startup after a startup for the first time

A prioritized startup of the IO device is always available to you after the first configuration of this IO device in the very first startup of the PROFINET IO system. Even in the case of spare parts or a reset to factory settings, the first startup is a standard startup for the respective configured IO devices.

Note

In the following situations, despite prioritized startup, there may be startup times of up to 8 s:

- An IO/device is disconnected and reconnected within 8 seconds.
- On a docking point, you can dock multiple physical IO devices as an IO device with a certain device name and a certain IP configuration (for example, docking point for automatic transport system).

If you wish to achieve a shortest possible startup time of 500 ms, you must perform the following measures:

- Port setting on the IO device
- Wiring dependent on the interconnected PROFINET device
- Measures in the user program

Setting prioritized startup

Requirement

You can enable the PROFINET function "Prioritized startup" for the IO devices (distributed I/O) only in the following cases:

- The IO controller used can prioritize selected IO devices during startup.
- The IO device used supports prioritization.

Note

Prioritized startup

In the case of an accelerated startup (prioritized startup), you must observe special conditions when setting up the PROFINET interface and when wiring if you wish to achieve the shortest possible startup times.

Procedure

1. Select the IO device in the network view or device view for which you wish to accelerate startup.
2. Open the IO device properties in the inspector window.
3. Select "PROFINET interface > Advanced options > Interface options".
4. Select the "Enable prioritized startup" check box.
5. Download the configuration to the IO controller.

Note:

For devices whose DP mode can be selected, the previously described check box can be found under "PROFINET interface > Operating mode". The CP 1616 for PC stations is one example of this type of device.

Note

Number of IO devices (distributed I/O) with prioritized startup

You can only start up a set maximum number of IO devices with PROFINET functionality "Prioritized startup" within one IO system. This maximum number depends on the IO controller used.

See also

Optimizing port settings on the IO device and IO controller (Page 2943)

Optimizing port settings on the IO device and IO controller

Optimizing port settings on the IO device and IO controller

The transfer medium and the duplex option are checked during startup of the IO device in the case of control unit cabling.

These checks require time, but with specific presets of these options you can save time. Make certain that the settings made correspond to the actual conditions (using the correct wiring).

Synchronizing the properties of local port and partner port

To synchronize the properties for the local port and partner port, follow these steps:

1. First, select the relevant local port and then the partner port and perform the following steps for both ports.
2. Synchronize the "Transmission medium/duplex" setting of the respective ports in the properties under "Port options". Use the setting "TP 100 Mbps full duplex".
3. Under "Port options", clear the check box "Enable autonegotiation" for both ports.

Configuring media redundancy

What you should know about media redundancy

Media redundancy options

To increase the network availability of an industrial Ethernet network with optical or electrical line topology, the following options are available:

- Meshing networks
- Parallel switching of transmission modes
- Consolidation of a line topology to a ring topology

Media redundancy in ring topologies

Devices in ring topologies can be the external switches and/or integrated switches of communication modules.

To set up a ring topology with media redundancy you must join both free ends of a line-shaped network topology to one device. Consolidation of a line topology to a ring takes place over 2 ports (ring ports) in a device in the ring. This device is the redundancy manager. All other devices in the ring are the redundancy clients.

The following figure shows the ring-shaped interconnection of the IO devices:

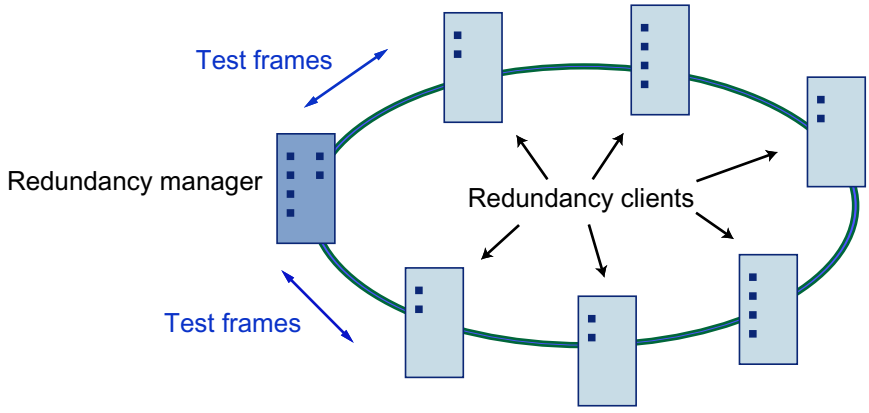

The two ring ports in a device are the ports which create the connection to both of its neighboring devices in a ring topology. The selection and set up of the ring ports takes place during configuration of the respective device.

Download the configuration in the individual devices before physically connecting the ring.

Functions of media redundancy in a ring topology

If a part of the ring is disconnected, then the data paths between the individual devices is automatically reconfigured. After the reconfiguration the devices are once again accessible in the newly created topology.

In the redundancy manager, 2 ring ports will be separated from each other during interruption-free network operation, so that no data frames circulate. The ring topology is a line when viewed from the standpoint of data transfer. The redundancy manager monitors the ring topology. To do so, it sends a test frame not only from ring port 1 but also from ring port 2. The test frames pass through the ring in both directions until they arrive at the other ring port on the redundancy manager.

An interruption in the ring can occur from connection failure between two devices or device failure in the ring.

If the redundancy manager's test telegram does not reach the other ring port because of interruption in the ring, the redundancy manager connects via both ports. With this alternate route a functioning connection is again created among all remaining devices in a line-shaped topology.

As soon as the interruption is eliminated, the original transmission routes are used again, the two ring ports on the redundancy manager are separated from each other, and the redundancy clients are notified of the change. Data traffic now runs again as before the interruption.

Media redundancy processes

In STEP 7 you can use the media redundancy process MRP (Media Redundancy Protocol) for ring topologies; up to 50 devices per ring can be included.

In addition, the real-time capable media redundancy process MRPD (Media Redundancy with Planned Duplication of frames) is also available.

Reconfiguration time

The time between ring interruption and re-establishment of a functional line topology (with resetting up of the address tables) is called the reconfiguration time. It typically amounts to less than 200 ms.

If the watchdog time of an IO device is shorter than the reconfiguration time of the ring, there is a danger that communication with the IO device will be interrupted (station failure).

Recommendation: When configuring the watchdog time of an IO device, select a time longer than 200 ms (in the properties of the IO device: PROFINET interface > Advanced options > Real time settings).

See also

Media Redundancy Protocol (MRP) (Page 2945)

Configuring media redundancy (Page 2948)

Media Redundancy with Planned Duplication of frames (MRPD) (Page 2951)

Managing MRP domains (Page 2952)

Media Redundancy Protocol (MRP)

Media Redundancy Protocol (MRP)

The "MRP" process works in conformity with the Media Redundancy Protocol (MRP) that is specified in the standard IEC 62439-2.

Requirements

- All devices in the ring support MRP.
- You have complied with the rules for topology set out below. STEP 7 monitors compliance with the rules in compilation and outputs corresponding messages.

Topology

The following schematic shows a possible topology for devices in a ring with MRP. The devices inside the shaded oval are in the redundancy domain.

Example of a ring topology with the MRP media redundancy protocol:

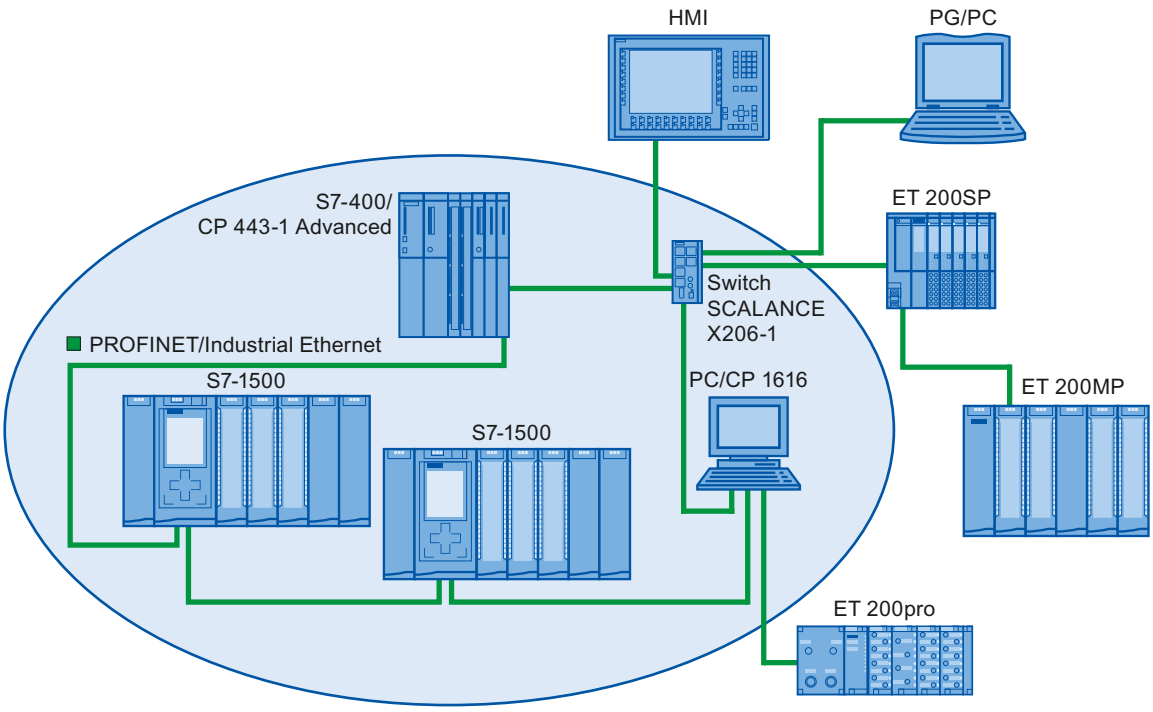

The following rules apply to a ring topology with media redundancy using MRP:

- All devices must be connected to each other via their ring ports.
- All devices in the ring belong to the same redundancy domain.
- A device within the ring is the redundancy manager.
  - One device only has the role of "Manager". No other device may have the role of "Manager". Or
  - Several devices in the ring have the role of "Manager (auto)". The devices with the role of "Manager (auto)" then negotiate between themselves which device is to take on the role of redundancy manager. In this case, no device may have the role of "Manager".
- All other devices in the ring are redundancy clients.
- You can connect up to 50 devices in a ring.

Non MRP-compliant devices can, for example, be connected to the ring via a SCALANCE X switch or via a PC with a CP 1616.

Rules for loading the devices of an MRP domain.

When devices of an MRP domain are loaded, circulating frames and thus failure of the network can occur if there is an invalid MRP configuration.

Example: You change the MRP roles of several devices and consecutively load the configuration into the devices involved. Configurations can arise that contradict the roles named above, for example devices with the "Manager" and "Manager (auto)" role can exist simultaneously in the ring.

In order to ensure that an invalid MRP configuration does not result in the failure of the network, dissolve the ring before loading. Follow these steps:

1. Dissolve the ring.
2. Load the error-free and consistent MRP configuration from your project into all the devices involved and ensure that the devices are in data exchange mode (i.e. the application relations (ARs) are set up).
3. Close the ring.

Boundary conditions

MRP and RT

RT operation is possible with the use of MRP.

Note

The RT communication is disrupted (station failure) if the reconfiguration time of the ring is greater than the selected response monitoring time of the IO device. This is why you should select a response monitoring time of the IO devices that is sufficiently large.

MRP and IRT

IRT mode is not possible together with MRP.

If you want to use media redundancy together with IRT in a ring, only use devices that support MRPD.

MRP and TCP/IP (Open User Communication via, for example, TSEND, Web access via HTTP, etc.)

The TCP/IP communication with MRP is possible, because lost data packages are resent, if applicable.

MRP and prioritized startup

If you configure MRP in a ring, you cannot use the "prioritized startup" function in PROFINET applications on the devices involved.

If you want to use the "Prioritized startup" function for a device, the device may not be a participant in the ring.

See also

Configuring media redundancy (Page 2948)

What you should know about media redundancy (Page 2943)

Configuring multiple rings (Page 2953)

Configuring media redundancy

Configuring MRP

Proceed as follows to create a PROFINET IO configuration with MRP in STEP 7:

1. Generate a ring via the port interconnections in the topology view. First interconnect the devices to a line topology. Connect the unassigned port of the last device in the line with the unassigned port of the first device.
   The following example shows one CPU 1516-3 PN/DP and two interface modules IM 155-6 PN HF that are interconnected in a ring in the topology view of STEP 7.

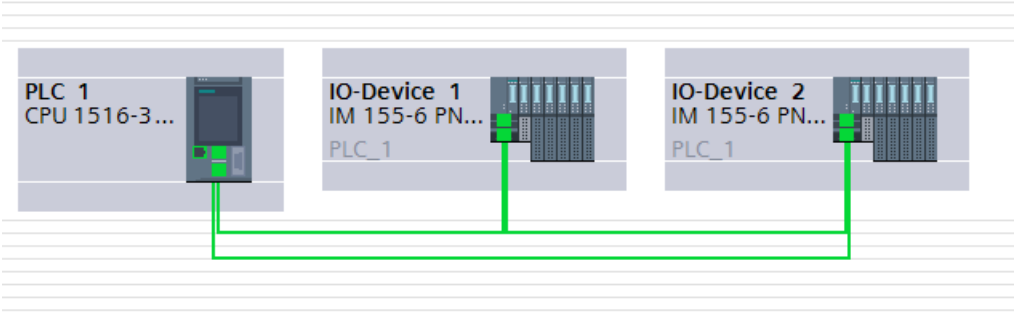

2. Select the PROFINET IO system in the network view.
3. In the Inspector window, navigate to "Properties" > "General" > "PROFINET" > "MRP domains" in the "Ring interconnection" field.
   This field shows you all the topological rings in the IO system with the associated MRP domains.

4. Select the ring generated above in the "Ring interconnection" field.
   The table below it shows all PROFINET devices in the ring.
5. Set the media redundancy role for the PROFINET devices in the MRP role column.
   If you interconnect the devices, the MRP roles are initially set to "Not device in the ring" for all devices. This configuration is not consistent.
   You have the following options: You can manually assign MRP roles according to the rules of MRP configurations (see setting options "Media redundancy"). You can have the MRP roles assigned automatically by STEP 7, see next paragraph.

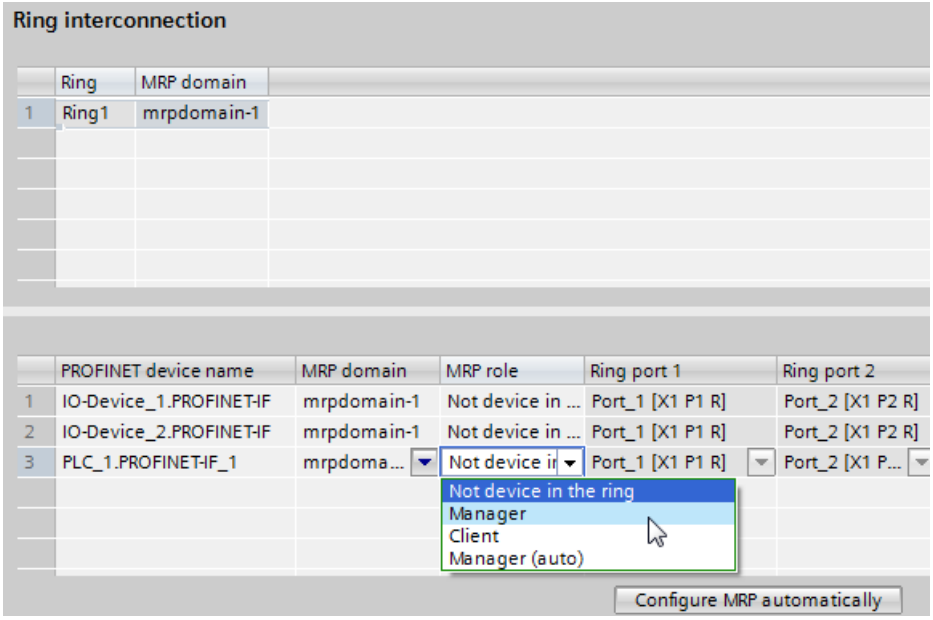

Automatic MRP configuration

You can also have the media redundancy roles assigned automatically for your PROFINET devices in the ring.

To have the media redundancy roles assigned automatically, click the "Configure MRP automatically" button. STEP 7 automatically assigns the media redundancy role for each device in the ring. After the automatic MRP configuration, you can make modifications to the media redundancy roles in the "MRP role" column.

"Media redundancy" setting options

Media redundancy role

Depending on the device used, the roles "Manager", "Manager (Auto)", "Client" and "Not device in the ring" are available.

Rules:

- A ring must have precisely one device with the role "Manager". No additional devices with the role "Manager" or "Manager (Auto)" are permissible. All the other devices may only have the "Client" role.
- If a ring has no device with the "Manager" role, the ring must at least have a device with the role "Manager (Auto)". Any number of devices with the "Client" role may exist.

Ring port 1 / Ring port 2

Select one at a time those ports you want to configure as ring port 1 or ring port 2. The drop-down list box shows the selection of possible ports for each device type. If the ports are set at the factory, then the fields are unavailable.

If you use single-stage commissioning (only Siemens devices with automatically assigned MRP roles in the ring after clicking "Configure MPR automatically"), use the ring ports preset in STEP 7.

Diagnostic interrupts

If diagnostic interrupts to the MRP state are to be output in the local CPU, select the "Diagnostic interrupts" check box. The following diagnostic interrupts can be configured:

- Wiring or port error
  Diagnostic interrupts will be generated for the following errors in the ring ports:
  - A neighbor of the ring port does not support MRP.
  - A ring port is connected to a non-ring port
  - A ring port is connected to the ring port of a different MRP domain
- Interruption / return (redundancy manager only)
  The redundancy manager generates diagnostic interrupts when the ring is interrupted and when the original configuration returns. If both of these interrupts occur within 0.2 seconds, this indicates an interruption of the ring.

You can respond to these events in the user program by programming the appropriate response in the diagnostic error interrupt OB (OB 82).

Tip: Delete the unneeded MRP domains

If you want to delete unneeded MRP domains, for example, because they no longer contain any devices, select the PROFINET IO system and then select the "MRP domains" area:

1. Navigate to the first table. This is the table in which you select the default domain.
2. Select the row with the MRP domain to delete and press the <Del> key.

You can delete any MRP domain except the default domain.

See also

Configuring multiple rings (Page 2953)

Media Redundancy Protocol (MRP) (Page 2945)

What you should know about media redundancy (Page 2943)

Managing MRP domains (Page 2952)

Media Redundancy with Planned Duplication of frames (MRPD)

MRP extension "Media Redundancy with Planned Duplication of frames" (MRPD)

The MRP extension "Media Redundancy with Planned Duplication of frames" (MRPD) provides the advantage that, in the case of a failure of a device or a line in the ring, all other devices continue to be supplied with IO data without interruption and with short update times.

MRPD is based on IRT and MRP. To realize media redundancy with short update times, the PROFINET devices participating in the ring send their data in both directions. The devices receive this data at both ring ports so that there is no reconfiguration time.

Requirements for media redundancy with MRPD

- All the devices of the ring must support MRPD, for example interface module IM 155-6 PN HS as of firmware version 4.0, SCALANCE X300 as of V4.0. (Terminal) devices at the switch that cyclically exchange IRT data with a ring component must also support MRPD.
  Whether or not a device supports MRPD can be seen in the infotext for the device in the STEP 7 hardware catalog (Devices & networks).
- You have configured MRP for all the devices of the ring. You have assigned the MRP role "Not device in the ring" to devices that are not located in the ring.
- You have configured IRT for all the involved components.

Configuring MRPD

You do not have to explicitly activate MRPD in STEP 7. The function is available automatically as soon as all the requirements for MRPD are fulfilled.

Redundancy levels of IO devices with MRPD

The redundancy level of an IO device specifies how strongly the real-time communication is influenced in the case of a power interruption between an IO device and its IO controller.

- Full redundancy: No influence, because the IO controller and IO device are located in the same ring.
- Partial redundancy:
  - If the interruption takes place on a non-redundant part (line) between the IO device and IO controller, real-time communication is influenced.
  - If the interruption takes place on a redundant part (ring), real-time communication is not influenced.
- No redundancy: No redundant path between the IO device and its IO controller, communication is always influenced.

The figure below shows the redundancy levels of the IO devices for an example configuration with MRPD. The three IO devices in the ring and the switch have the redundancy level "Full redundancy". Device 4 has the redundancy level "Partial redundancy", because the connection between the switch and the IO device is not redundant.

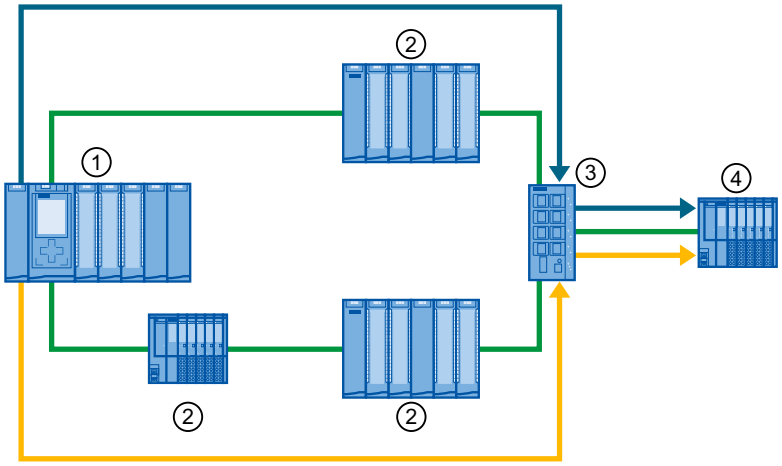

① IO controller in the ring
② IO device in the ring with redundancy level "Full redundancy"
③ Switch in the ring with redundancy level "Full redundancy"
④ IO device with spur line connected to the ring with "Partial redundancy" redundancy level

The redundancy levels are displayed in addition to other properties of the Sync domain in STEP 7:

"Domain management > Sync domains >... > Devices".

Application example: Structure of a ring topology based on "MRPD"

Use the following link to download an application example for the structure of a ring topology based on MRPD:

https://support.industry.siemens.com/cs/ww/en/view/109744035 (http://support.automation.siemens.com/WW/view/en/109744035)

See also

What you should know about media redundancy (Page 2943)

Managing MRP domains

You can monitor and adjust the media redundancy settings centrally.

Requirements

- The components involved must support the media redundancy protocol (MRP).
- The devices are configured and topologically interconnected.

Procedure

To open the MRP domain, follow these steps:

1. Select any PROFINET interface, and select the "Advanced options > Media redundancy" area in the inspector window.
2. Click the "Domain settings" button.

or

1. Select the subnet or the IO system.
2. Navigate to the "PROFINET > Domain management > MRP domains" area in the inspector window.

Here, you can specify centrally which MRP domain is to be the default domain. You can change the names of all domains and check redundancy roles and ring ports.

For each MRP domain, you receive an overview of how many PROFINET interfaces are interconnected in the ring, how many PROFINET interfaces are located outside the ring, and the number of redundancy managers and redundancy clients.

You can have the devices in the ring, including device name, associated MRP domain, media redundancy role, and utilized ring ports, displayed in groups:

- MRP domains
  The list shows you the existing MRP domains of the selected IO system.
- <Name of MRP domain>
  The list shows you the number of PROFINET interfaces and devices with the respective redundancy roles for the selected MRP domain.
- Devices
  The list shows you all configured devices of the selected IO system.

See also

What you should know about media redundancy (Page 2943)

Configuring media redundancy (Page 2948)

Configuring multiple rings

Multiple rings

Use multiple rings to achieve higher availability for PROFINET IO networks with star topology.

In a multiple ring configuration, several PROFINET lines lead from a switch (star topology). The individual PROFINET lines lead from IO device to IO device. PROFINET lines run back to the switch from the last IO device on each individual line, thus forming several rings.

The switch operates as a manager. The manager must have two ring ports for each ring. Several rings are possible. For example, a SCALANCE X414 as of firmware version 3.10 supports up to 4 rings.

The manager monitors all the rings individually: It checks for each particular ring (an MRP domain) whether the transmission path is intact. To do this, it uses an MRP instance in each case. An MRP instance is required for each connected ring (set up automatically by STEP 7).

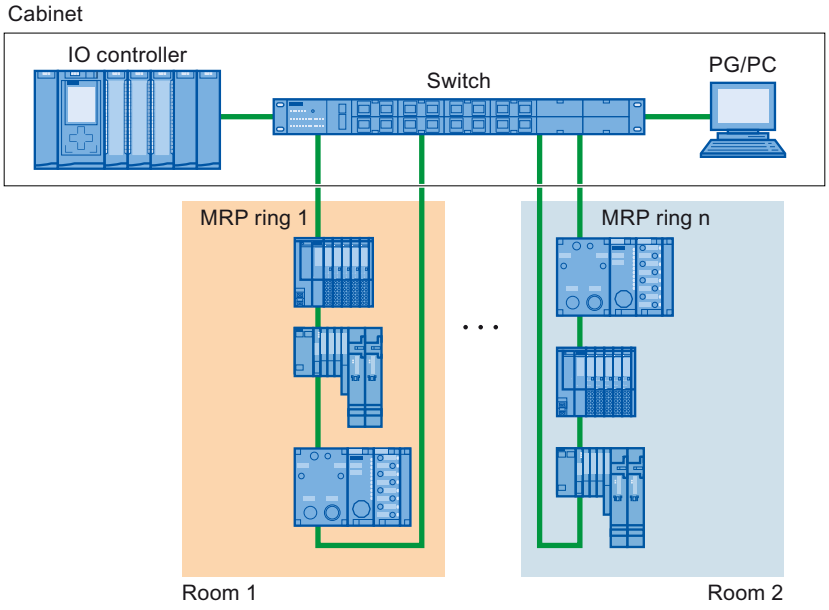

Requirements

The following devices support multiple rings as a manager:

- SCALANCE X300 as of version 4.0
- SCALANCE X414 as of version 3.10

Rules for the configuration of multiple rings

- MRP role at multiple rings:
  - The device that belongs to all the rings must have the MRP role in every instance that is entered in the GSD file in the "SupportedMultipleRole" attribute.
  - The switches from the SCALANCE X300 series V4.0 and higher (only via GSDML) and the X414 switch Version 3.10 and higher support the MRP role "Manager" for multiple rings.
- If the device that belongs to all rings has the "Manager" role in one ring, no devices with the "Manager (Auto)" role may exist in this ring.

Configuring multiple rings

To configure an MRP configuration with multiple rings, follow these steps:

1. In the topology view, interconnect the ring ports of the devices that are intended to belong to an MRP domain to form a ring.
2. Select the PROFINET IO system in the network view.

3. In the Inspector window, navigate to "Properties" > "General" > "PROFINET" > "MRP domains" in the "Ring interconnection" field.
   This field shows you all the topological rings in the IO system with the associated MRP domains.
4. Select one of the rings generated above in the "Ring interconnection" field.
   The table below it shows all PROFINET devices in the selected ring.
5. Set the media redundancy role for the PROFINET devices in the MRP role column.

Adapting preset MRP roles

If you interconnect the devices of the example configuration as described below, the MRP roles are initially set to "Not device in the ring" for all devices.

This configuration is not consistent.

You have the following options:

- You can manually assign MRP roles according to the rules described for MRP configurations. To do this, edit the properties of the PROFINET IO system, in the area "MRP Domains > Ring interconnection".
- You can have the MRP roles assigned automatically by STEP 7.

In both cases, select a configured MRP ring (in the area "MRP domains > Ring interconnection") and click the "Configure MRP automatically" button. Repeat the process for each configured MRP ring.

The rings are then are assigned new MRP domains and the MRP roles and MRP instances are set. The configuration is consistent.

Example of multiple rings

The following figure shows an example configuration for multiple rings with two rings ① and ②.

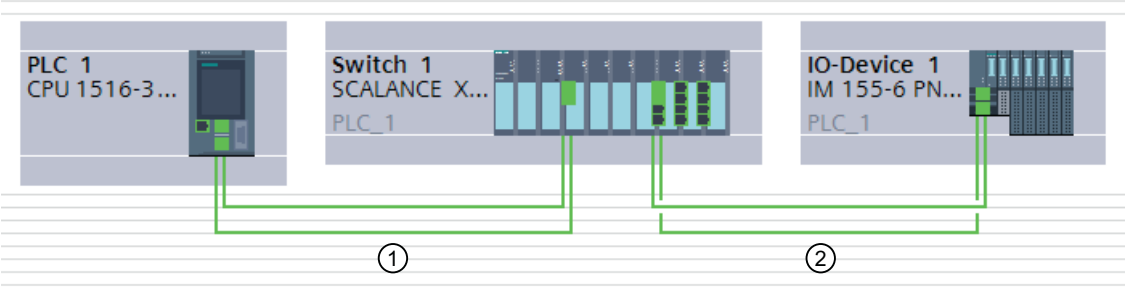

In the example, Switch 1 belongs to two MRP rings. Ring 1 is formed by Switch 1 and PLC 1, Ring 2 by Switch 1 and IO device 1.

The manager is located at the point of intersection of the two rings 1 and 2. The manager monitors the two rings separately. To do this, it uses two MRP instances.

One MRP instance checks whether all devices in ring 1 are reachable, another instance monitors whether all devices in ring 2 are reachable (only one device in each case in the example).

You can configure each MRP instance separately.

The following figure shows the two MRP instances in the manager (PROFINET interface of the switch). Here in the example, MRP instance 1 checks whether the devices in the MRP domain "mrpdomain-1" can be reached. The MRP instance 2 is responsible for monitoring the devices of the MRP domain "mrpdomain-2".

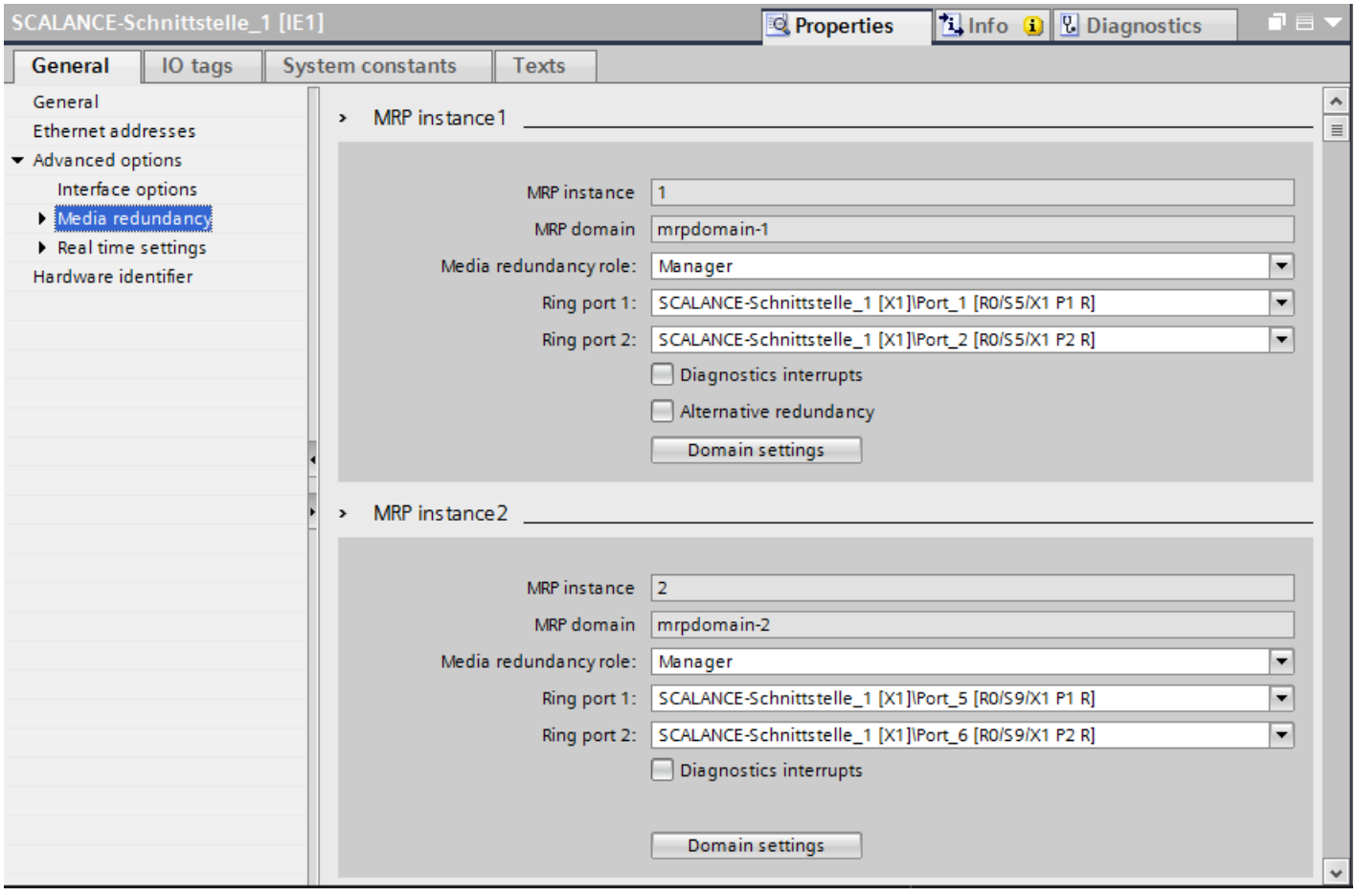

The following figure shows Ring 1 (mrpdomain-1). The participants of the mrpdomain-1 are the PROFINET interface of the CPU as the "Client" and the MRP interface 1 of the PROFINET interface of the switch as the "Manager".

Ring interconnection

	Ring	MRP domain
1	Ring1	mrpdomain-1
2	Ring2	mrpdomain-2

	PROFINET device name	MRP instance	MRP domain	MRP role	Ring port 1	Ring port 2
1	PLC_1.PROFINET-IF_1		mrpdomain-1	Client	Port_1 [X1 P1 R]	Port_2 [X1 P2 R]
2	Switch_1.SCALANCE-IF_1	1	mrpdomain-1	Manager	Port_1 [R0/S5/X1 P1 R]	Port_2 [R0/S5/X1 P2 R]

The following figure shows Ring 2 (mrpdomain-2). The participants of the mrpdomain-2 are the PROFINET interface of the IO device as the "Client" and the MRP instance 2 of the PROFINET interface of the switch as the "Manager".

Ring interconnection

	Ring	MRP domain
1	Ring1	mrpdomain-1
2	Ring2	mrpdomain-2

	PROFINET device name	MRP instance	MRP domain	MRP role	Ring port 1	Ring port 2
1	IO-Device_1.PROFINET-IF		mrpdomain-2	Client	Port_1 [X1 P1 R]	Port_2 [X1 P2 R]
2	Switch_1.SCALANCE-IF_1	2	mrpdomain-2	Manager	Port_5 [R0/S9/X1 P1 R]	Port_6 [R0/S9/X1 P2 R]

Tip: Delete the unneeded MRP domains

If you want to delete unneeded MRP domains, for example, because they no longer contain any devices, select the PROFINET IO system and then select the "MRP Domains" area.

1. Navigate to the first table. This is the table in which you select the default domain.
2. Select the row with the MRP domain to delete and press the <Del> key.

You can delete any MRP domain except the default domain.

See also

Configuring media redundancy (Page 2948)

Media Redundancy Protocol (MRP) (Page 2945)

Configuring PROFIenergy (S7-300, S7-400, S7-1500)

Overview of PROFIenergy (S7-300, S7-400, S7-1500)

Saving energy with PROFIenergy

PROFIenergy is a PROFINET-based data interface for switching off consumers centrally and with full coordination during pause times regardless of the manufacturer or device type. Through this, the process should only be provided with the energy that is absolutely required. The majority of the energy is saved by the process; the PROFINET device itself only contributes a few watts of savings potential.

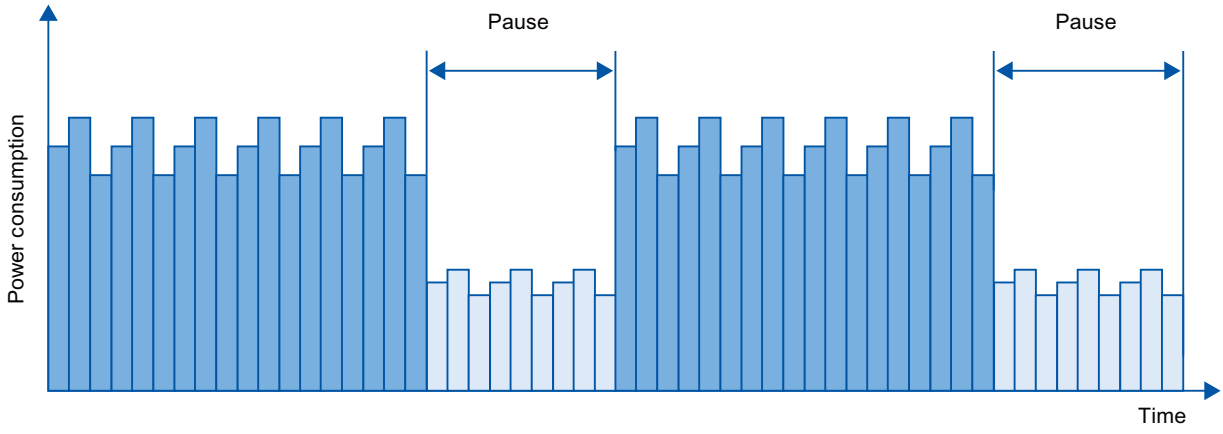

Figure 1-93 Energy savings during pauses with PROFIenergy

Basic information

Turning off the PROFINET devices or the power modules occurs through special commands in the user program of the PROFINET IO Controller. No additional hardware is needed, the PROFIenergy commands are directly interpreted by the PROFINET devices.

Principle of operation

At the beginning and end of pauses, the system manager enables or disables the pause function of the system; then the IO controller sends the PROFIenergy command "Start_Pause" / "End_Pause" to the PROFINET device. The device interprets the contents of the PROFIenergy command and turns on / off.

Through other PROFIenergy functions, device information can be accessed during pauses. The user can use this information in order to punctually transfer the "Start_Pause" / "End_Pause" command.

PROFIenergy instructions for IO controller

Two instructions are needed for controlling and monitoring the PROFIenergy functions.

The PE_START_END instruction allows you to easily activate and deactivate the idle state of PROFINET devices. This occurs by means of an incoming edge or outgoing edge. The PE_START_END instruction provides a simple interface for realizing PROFIenergy commands Start_Pause and End_Pause.

The PE_CMD instruction allows you to transmit all PROFIenergy commands, including Start_Pause and End_Pause. With the other commands, for example, the current status of the PROFINET device or the behavior during the pauses can be inquired. The PE_CMD instruction is a convenient means for handling all PROFIenergy functions.

PROFIenergy instruction for I-devices

The PE_I_DEV instruction allows you to realize PROFIenergy on I-devices, as well. The instruction receives PROFIenergy commands on the I-device and forwards these to the user program for processing. After processing the command, the user program calls the PE_I_DEV instruction again in order to send the acknowledgment to the IO controller. For these replies, each command offers you a helper instruction that supplies the reply data to the instruction.

The instructions can be found in the "Instructions" task card of the STEP 7 program editor.

An application example can be found on the Internet (http://support.automation.siemens.com/WW/view/en/41986454) in the Service and Support portal.

Configuration and programming

The functions can be comfortably integrated into existing systems. Usually, no configuration is required for the use of PROFIenergy (exceptions under "See also"). However, amendments to the user program are required:

- Before the "Start_Pause" command, you must ensure that your system is brought into a condition that is suitable for a pause.
- A sequential control system for the beginning of the pause of the devices and for the punctual restarting of the device on break must be programmed (depending on the required startup times that the respective PROFINET device demands).
- The error messages of the PE_CMD instruction must be evaluated, and the required reaction must be programmed (for example, cancellation or continuation of further commands on lower-level PROFINET devices).

See also

Configuring PROFIenergy for ET 200S (Page 2960)

Application example for PROFIenergy (https://support.industry.siemens.com/cs/us/en/view/109478388)

Saving energy with SIMATIC S7 and SIMATIC HMI (TIA Portal) (https://support.industry.siemens.com/cs/ww/en/view/58235225)

Configuring PROFIenergy for ET 200S (S7-300, S7-400, S7-1500)

The ET 200S distributed I/O system offers a special feature: for certain interface modules you can specify which potential groups or power modules of the IO device are activated for PROFIenergy.

Special features for ET 200S HF V7.0 and ET 200S FO V7.0 interface modules

You can configure the initial behavior of the power modules (e.g., the switchable power module (PM-E DC24V/8A RO)) for the above-named IO devices so that you do not have to bring the device to the energy saving state in the startup program with the PROFIenergy PE_Start_End_Pause instructions.

You activate the energy saving function in the properties of the power module (module parameters area, section "PROFIenergy").

Note that provisions are necessary in the user program for the use of PROFIenergy functions.

Rules

The configuration option described above is not available for ET 200S devices that were incorporated in the hardware catalog using GSDML files.

See also

Overview of PROFIenergy (Page 2958)

Configuring PROFIenergy with I-devices (S7-300, S7-400, S7-1500)

The requirement for program-controlled pauses for saving energy with PROFINET devices is that the PROFINET devices support the PROFIenergy protocol.

Only if a PROFINET device (IO device) supports the PROFIenergy protocol does an IO controller actually send PE commands to this IO device, for example to start or stop pauses.

If an IO device supports the PROFIenergy protocol, this property is saved in its PROFINET GSD file and is available for configuration in an engineering system.

For S7-1500 CPUs as intelligent IO devices (I-devices), you have the option with STEP 7 V13 service pack 1 or later to set PROFIenergy support for each transfer area.

If you have selected the option "Enable PROFIenergy communication" for a transfer area and import the generated PROFINET GSD file into another project, you can handle an I-device as a PE entity there.

Requirement

- STEP 7 as of V13 service pack 1
- S7-15XX-CPU as of firmware version 1.7 configured as I-device with transfer areas
- The user program on the I-slave handles PROFIenergy commands
  Background: You need to program PROFIenergy functions with I-devices in the user program using the "PE_I_DEV" instruction and corresponding auxiliary blocks; this is different compared with IO devices for which this functionality is made available by the firmware. This means that you can only enable PROFIenergy support for transfer areas if the user program on the I-slave is suitably designed.

Enabling PROFIenergy for transfer areas of I-devices

Proceed as follows to configure support of PROFIenergy:

1. Select the PROFINET interface (X1) of the CPU.
2. Select the required transfer area in the area navigation, for example:
   Operating mode > I-device communication > Transfer_area_1.
3. Select the "Enable PROFIenergy communication" check box.

Once the I-device is fully configured, generate the GSD file for the I-device and import this file in the project for the I/O controller. The GSD file generated is compatible with GSD version 2.31 and contains an entry that specifies that the I-device supports the PROFIenergy profile.

To address the I-device, for example for the PE command "PE_START_END", use the hardware identifier of the "PROFIenergy supporting" transfer area in the I-device.

To address the IO controller for the PE command "PE_I_DEV", use the hardware identifier of the transfer area that is supplied with the data for PROFIenergy on the IO controller.

Configuring Shared Devices (S7-300, S7-400, S7-1500)

Useful information on Shared Devices (S7-300, S7-400, S7-1500)

Shared device functionality

Numerous IO controllers are often used in larger or widely distributed systems.

Without the "Shared device" function, each I/O module of an IO device is assigned to the same IO controller. If sensors that are physically close to each other must provide data to different IO controllers, several IO devices are required.

The "Shared device" function allows the modules or submodules of an IO device to be divided up among different IO controllers. Thus allowing flexible automation concepts. You have, for example, the possibility of combining I/O modules lying near other into an IO device.

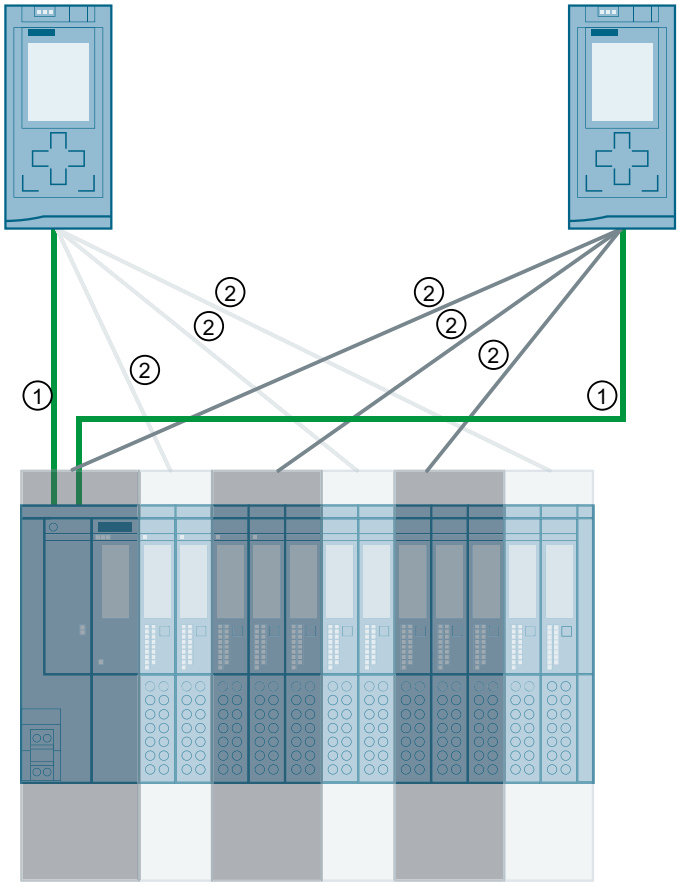

① PROFINET
② logical assignment

Principle

Access to the submodules of the shared device is divided up among the individual IO controllers. Each submodule of the shared device is assigned exclusively to one IO controller.

Requirement

- STEP 7 as of V12 Service Pack 1
- S7-1500 CPU as of firmware 1.1 / ET 200SP CPU as IO controller or S7-300/400 CPU as IO controller
- S7-1200 CPU FW 4.1 or higher.
- S7-300/400 CPU (STEP 7 TIA Portal as of V13 SP1)
- IO device supports the "Shared device" function, e.g., IM 155-5 PN ST interface module
- With GSD configuration of the IO device: GSD file for configuring the IO device is installed

Comments:

- An S7-1500 CPU configured as an I-device can be used as a shared device as of firmware version V1.5. The I-device must be integrated by GSD import into the IO system of the IO controller.
- An S7-300 CPU or S7-400 CPU configured as an I-device can be used as a shared device under the following condition:
  You must export the PROFINET GSD file for the I-device from STEP 7 (V5.5 or higher) or STEP 7 (TIA Portal) and then import it into STEP 7 (TIA Portal).

Configuring the access

The IO device must be present in several projects so that the modules or submodules of an IO device can be assigned to different IO controllers. A separate project is required for each IO controller.

You use the "Shared Device" parameter of the interface module to determine the modules or submodules to which the IO controller has access:

- If the local IO controller has access to the configured module, select the name of the IO controller from the list.
- If the IO controller from a different project and not the local IO controller is to have access to the configured module, select the entry "---".

The configuration is consistent regarding access if each module or submodule in exactly one project is assigned to an IO controller.

Module or submodule is assigned to another IO controller

The paragraph below describes the consequences of the "---" setting of the "Shared Device" parameter from the point of view of the local IO controller.

In this case, the local IO controller does not have access to the module configured in this way. Specifically, this means:

- No data exchange with the module or submodule
- No reception of alarms or diagnostics, which means no display of the diagnostics status in the online view
- No parameter assignment of the module or submodule

Setting of the real-time properties

STEP 7 calculates the communication load and thus the resulting update times. To enable this calculation in the case of shared device configurations, you must enter the number of project-external IO controllers in the project in which the PROFINET interface of the shared device is assigned to the IO controller.

The maximum possible number of IO controllers for the shared device depends on the device. This number is stored in the GSD file of the shared device.

You can set a very short send clock with an S7-1500 CPU as IO controller. The send clock can be shorter than the shortest send clock supported by the shared device. In this case, the

shared device is operated by the IO controller with a send clock that it supports (send clock adaptation).

Example:

An S7-1500 CPU supports send clocks starting from 0.25 ms. A configured IO device also supports send clocks starting at 0.25 ms; another IO device supports send clocks starting at 1 ms. In this case, you have the option of setting the short send clock of 0.25 ms for the CPU. The CPU operates the "slow" IO device with the send clock of 1 ms, for example.

This function is not available if you do not configure one of the IO controllers with access to the shared device with STEP 7. In this case, you must read out the shared send clock in the properties of the shared device in STEP 7 and enter it in the engineering tool for the IO controller.

Rules for the configuration

- IO controllers that use the shared device are created in different projects. In each project, you must ensure that the shared device is configured identically in each station. Only one IO controller may ever have full access to a submodule. Inconsistencies in the configuration result in the failure of the shared device.
- I/O addresses of a module or submodule can only be edited if a module or submodule is assigned to the IO controller in the same project.
- The shared device must have the same IP parameters and the same device name in each project.
- The send clock must be identical for all IO controllers that have access to the shared device.
- The S7 subnet ID of the subnet to which the shared device is connected must be identical in all projects.
- The following functions are only available if the PROFINET interface of the shared device is assigned to the local IO controller:
  - IRT operation
  - Prioritized startup
  - Parameter assignment of the port properties

Boundary conditions

The following boundary conditions result because a shared device configuration is distributed across several projects:

- The addresses of modules or submodules that are not assigned to this IO controller are missing in the address overview of each IO controller that has access to a shared device.
- The modules or submodules that are not assigned are not taken into consideration in the configuration limit calculation for the shared device during the consistency check. This means that you have to check yourself that the maximum number of submodules or the maximum amount of cyclic IO data is not exceeded for the shared device.

  Note

  If only one submodule is assigned to an IO controller via shared device, the name of the submodule is not shown in the CPU display. If more than one submodule was assigned, the number of configured submodules is shown in the CPU display.

- Configuration errors such as the assignment of a module or submodule to several IO controllers are not detected in STEP 7.
- CPUs that are loaded with a shared device configuration do not have any information on whether the IO device is a shared device. Modules or submodules that are assigned to other IO controllers and therefore other CPUs are missing in the loaded configuration. These modules or submodules are therefore displayed neither in the CPU Web server nor in the CPU display.

See also

Displaying IO controllers that access modules of a Shared Device (Page 3318)

Configuring an I-device as a shared device (Page 2968)

Configuring Shared Device (GSD configuration) (S7-300, S7-400, S7-1500)

Below, you will find a description of how to configure a distributed I/O system as a shared device with STEP 7 V12, Service Pack 1 or higher.

A "distributed" configuration with different engineering tools for different IO controller families is always possible. However, the description of the procedure is exclusively based on STEP 7 V12, Service pack 1. The description is limited to two IO controllers of the S7-1500 family that share a shared device.

Two projects are created (Shared-Device-1 and Shared-Device-2), each with one IO controller (PLC1 and PLC2). You must create the shared device in both projects, even though it is physically one and the same IO device.

Requirements

- STEP 7 as of V12 Service Pack 1
- In the context of STEP 7 V12, Service Pack 1: Only CPU 15XX as of FW 1.1 can be configured as IO controller (no CPU 31X, CPU 41X or CPU 12XX).

- In the context of other IO controllers that are to have access to a shared device: Corresponding engineering tool, for example, STEP 7 V5.5, Service Pack 3 for CPU 31x-3 PN/DP.
- IO device supports shared device functionality, e.g., interface module IM 155-5 PN ST V2.0.
- GSD file for configuring the IO device as shared device is installed.

Procedure - Creating project 1

To create the first project with a shared device, follow these steps:

1. Start STEP 7.
2. Create a new project with the name "Shared-Device-1".
3. Insert a CPU 1513-1 PN from the hardware catalog in the network view. Name it "PLC1".
4. Insert an IO device with the "Shared device" function from the hardware catalog (hardware catalog: Other field devices > PROFINET IO > I/O).
5. Assign the IO controller "PLC1" to the IO device.
6. Double-click the IO device and insert all required modules and submodules from the hardware catalog in the device overview table.
7. Assign the module parameters.
8. Save the project.

Procedure - Creating project 2

To create the second project with a shared device, follow these steps:

1. Start STEP 7 once again.
   A new instance of STEP 7 opens.
2. In the new instance, create a new project with the name "Shared-Device-2".
3. Insert a CPU 1513-1 PN in the network view. Name it "PLC2".
4. Copy the IO device from the project "Shared-Device-1" and insert it in the network view of project "Shared-Device-2".
5. Assign the IO controller "PLC2" to the IO device.
6. Save the project.

Both projects now have an identically structured IO device that must be configured in the next step for the different types of IO controller access.

Procedure - Configuring access to the shared device

The modules and submodules you insert in the shared device are automatically assigned to the local CPU. To change the assignment, follow these steps:

1. Select the interface module in the network or device view of project "Shared-Device-1".
2. Select the "Shared device" area in the Inspector window.
   A table shows which CPU has access to the respective module or submodule for all configured modules. The default setting is that the local CPU has access to all modules and submodules.
3. Keep the "PLC1" setting for all modules and submodules that are to remain in the address range of the local CPU
   Select the setting "---" for all modules and submodules that are to be located in the address range of the CPU from the "Shared-Device-2" project (PLC2). This means that an IO controller outside the project is to have access to the module or submodule.
4. Select the interface module in the network or device view of project "Shared-Device-2".
5. Select the "Shared device" area in the Inspector window.
   A table shows which CPU has access to the respective module or submodule for all configured modules.
6. Select the setting "---" for all modules and submodules that are to be located in the address range of the CPU from the "Shared-Device-1" project (PLC1).
7. Finally, check whether the settings for access are "complementary" for each module or submodule in both projects. This means that if the local CPU has access in one project, the option "---" must be set in the other project and vice versa.
   Special note: The option "---" for the PROFINET interface and therefore for the ports has the effect that the associated parameters are deactivated and cannot be changed. Parameters of the PROFINET interface and port parameters can only be edited in the project in which the PROFINET interface is assigned to the local CPU. The ports can be interconnected in both projects regardless of this.
8. Check whether the same IP address parameters and device name are set for the shared device in all projects.
   Check whether the same S7 subnet ID is set in all projects for the subnet to which the shared device is connected (subnet properties, "General" area in the Inspector window).

Note

If you make changes to the shared device: Make the same changes in each project on the shared device. Make sure that only one IO controller has access to a module or submodule.

Procedure - Adjusting the real-time settings

To ensure that all IO controllers and shared devices are operated with the appropriate send clock and that the update times are calculated correctly based on the communication load, you must adjust and check the following settings:

1. Select the project whose IO controllers have access to the PROFINET interface and the ports of the shared device.
2. Select the interface module of the shared device in the network view.

3. In the Inspector window, navigate to the "PROFINET interface > Advanced options > Real time settings > IO cycle" area.
4. In the "Shared device" area, set the number of project-external IO controllers. The maximum number depends on the IO device (default in GSD file).
5. You must set the same send clock for each IO controller that has access to modules and submodules of the shared device:
   - If you configure the IO controller with STEP 7 (TIA Portal):
     Open the corresponding project.
     Select the PROFINET interface of the IO controller.
     Select the "Advanced options > Real time settings > IO communication" area in the Inspector window and set the shared send clock.
   - If you configure the IO controller with a different engineering tool:
     Select the PROFINET interface of the shared device in STEP 7 (TIA Portal) and read out the send clock on the shared device ("Advanced options > Real time settings" area)
     Enter the read send clock in the engineering tool.
   - Special note: If you configure **all** IO controllers that have access to the shared device in STEP 7 (TIA Portal or V5.5), you can set shorter send clocks on the IO controller than supported by the shared device (send clock adaptation).

Compiling and loading

You must compile the configurations for the different IO controllers and load them to the CPUs one after the other.

Due to the distributed configuration with separate projects, STEP 7 does not output consistency errors in the case of incorrect access parameter assignment. Examples for incorrect access parameter assignment:

- Several IO controllers have access to the same module
- IP address parameters or send clocks are not identical

These errors do not have an effect until operation and are signaled as configuration errors, for example.

Configuring an I-device as a shared device (S7-300, S7-400, S7-1500)

Below, you will find a description of how you configure an S7-1500 as an I-device with STEP 7 Version V13 or higher and then use it in two projects as a shared device.

A "distributed" configuration with different engineering tools for different IO controller families is generally also possible here. The procedure described below is based on STEP 7 V13 and is limited to a configuration with two IO controllers of the S7-1500 family that share the transfer areas of an I-device as a shared device. The I-device itself is also an S7-1500 CPU.

First, the general concept is described.

The following sections explain the step-by-step procedure using an example: Three projects are created with one IO controller each (PLC-I-Device, PLC_1, and PLC_2).

PLC-I-Device is used to configure the I-device. The PROFINET GSD variant of PLC-I-Device is used in the PLC_1 and PLC_2 projects in order to assign the transfer areas in the respective higher-level IO controller.

Shared I-device concept

For the introduction of the shared I-device concept, two roles are distinguished:

- The role of manufacturer (e.g., machine manufacturer): The manufacturer configures and programs an I-device that performs a particular automation task. Transfer areas are defined as the I/O interface to the operator of the machine. These transfer areas can be assigned to different IO controllers. For the connection to higher-level IO controllers, the manufacturer provides a PROFINET GSD file and discloses the transfer areas via which the I-device can be accessed.
- The role of the operator: The operator uses the I-device as a PROFINET GSD variant during configuration of the PROFINET IO system and, in this process, specifies the I/O addresses under which the IO controllers access the transfer areas.

Manufacturer view

When assigning parameters, you specify the S7-1500 CPU as an I-device with central and distributed I/O. The following settings are important:

- Required transfer areas
- Number of IO controllers with access to this I-device (with shared device always greater than 1!)

Special note: The I-device is configured without a higher-level IO controller. As a result, only the local I/O addresses of the transfer area are available (= "Address in the I-device") in order to create the user program for editing the addresses from the transfer area.

The I-device that has been completely configured except for the connection to the higher-level IO controller is loaded to the S7-1500 CPU.

You export a PROFINET GSD file from the I-device configuration.

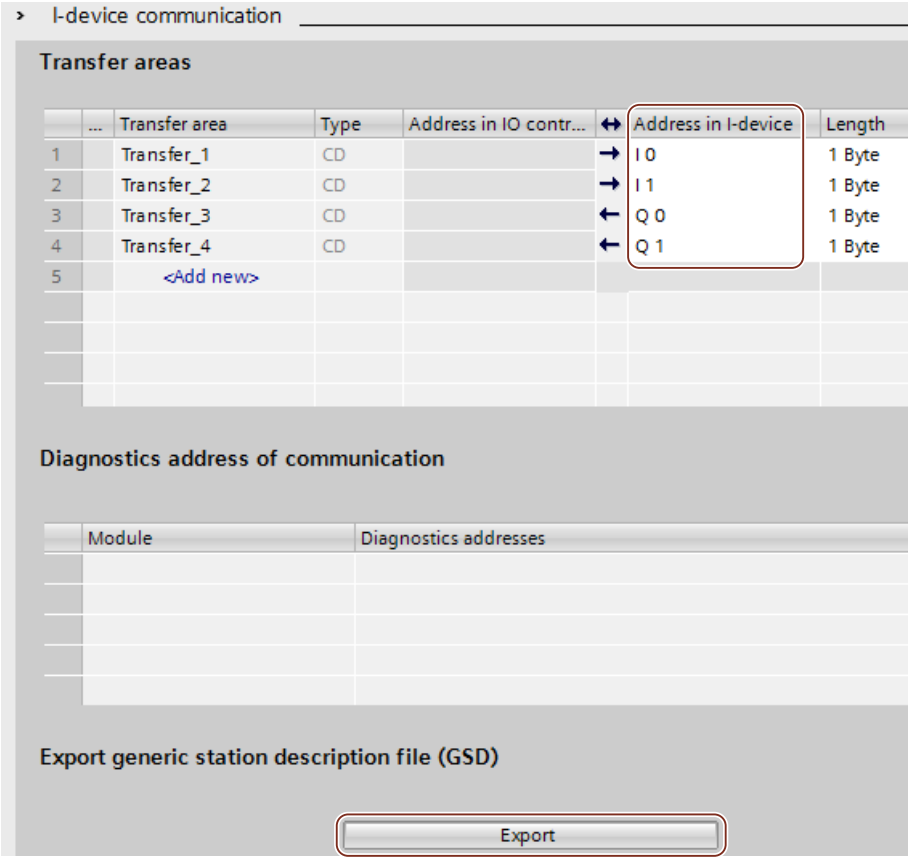

Operator view

You must install the PROFINET GSD file created from the I-device configuration in all engineering systems that are involved in configuring a PROFINET IO system with this shared I-device. If all uses of this I-device will be configured with STEP 7 V13, it is sufficient to install the GSD file in STEP 7.

Each higher-level IO controller with assigned shared I-device transfer area in its own project.

You configure the I-device as a GSD variant on the PROFINET IO system in the projects involved. In STEP 7 V13, this I-device can be found under "Other field devices > PROFINET IO > PLCs & CPs" following installation.

You need a separate project for each assignment between the shared I-device and one of the higher-level IO controllers that share the transfer areas.

The following figure shows the assignment of the shared I-device to the higher-level IO controller PLC_1.

For each higher-level IO-Controller you must create a separate project with the same shared I-device and grant the IO controller access to the desired partial transfer area.

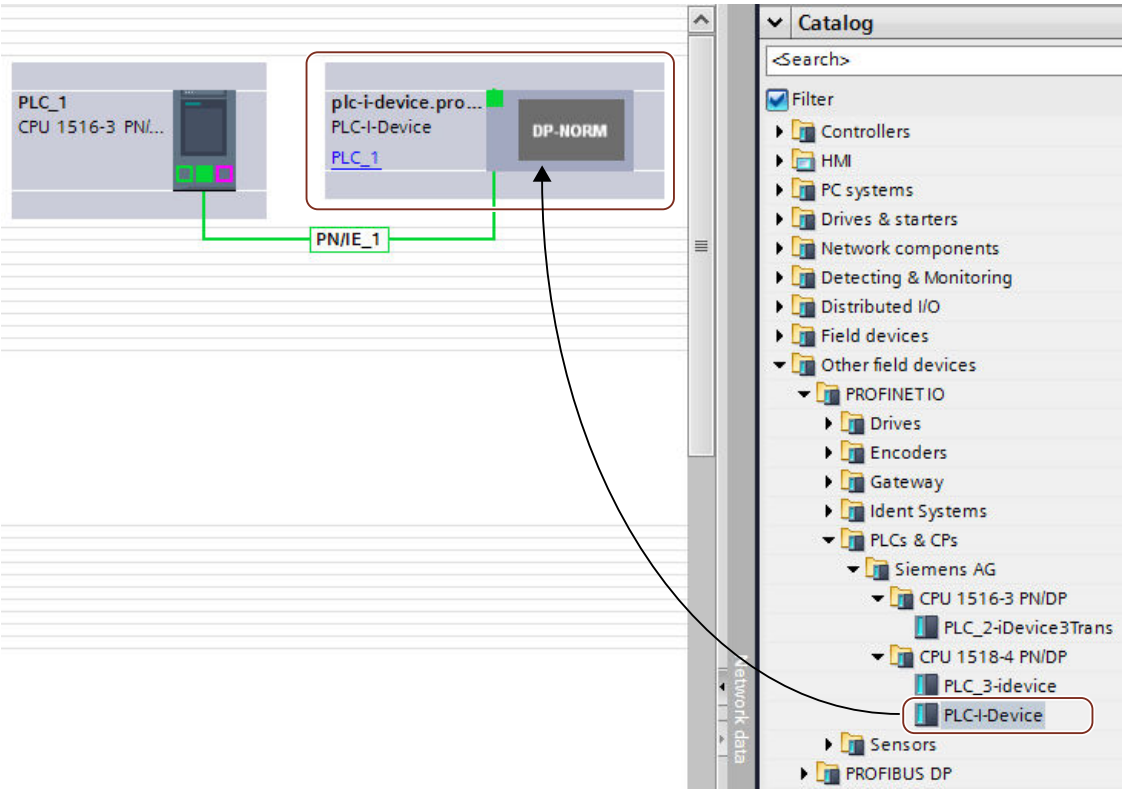

In each of the projects involved, you define which transfer areas are assigned exclusively to the higher-level IO controller (default setting: all). You set the other transfer areas to "---" (not assigned). As a result of this setting, the local IO controller has no access to this transfer area and it can therefore be assigned to another IO controller in another project.

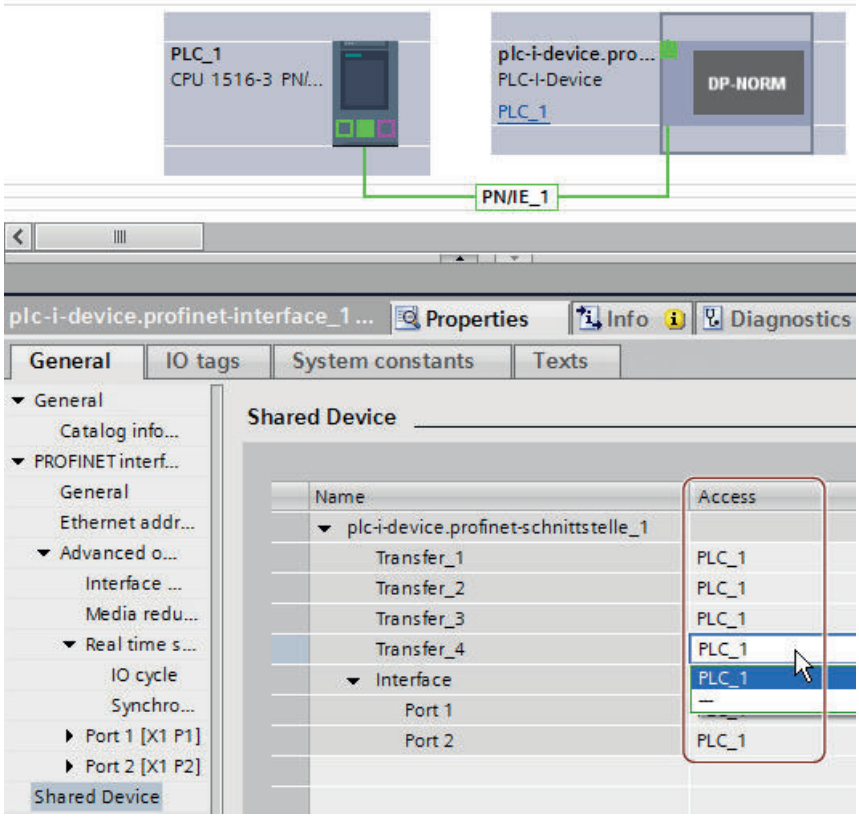

You adapt the addresses from the view of the IO controller in the device overview. To open the device overview, double-click on the I-device.

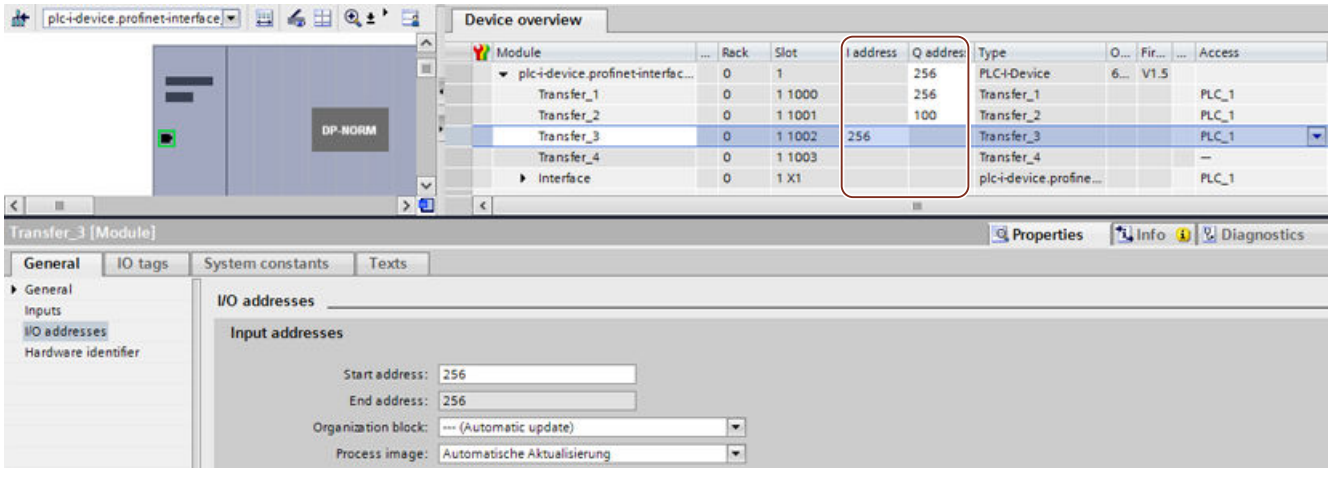

Requirement

- STEP 7 V13 or higher
- In the context of STEP 7 V13 or higher: Only CPU 15XX FW 1.5 or higher can be configured as a shared I-device (no CPU 31X, CPU 41X, or CPU 12XX).
- In the context of other IO controllers that are to have access to a shared I-device: Corresponding engineering tool, for example, STEP 7 V5.5, Service Pack 3 for CPU 31x-3 PN/DP.

Procedure - Creating the PLC-I-device project

To create the project with a shared I-device, follow these steps:

1. Start STEP 7.
2. Create a new project with the name "PLC-I-device".
3. Insert a CPU 1518-4 PN/DP from the hardware catalog in the network view. Assign the name "PLC-I-device".
4. Double-click the IO device and configure all required modules and submodules.

5. Assign the module parameters.
   In particular, the following settings for the CPU are necessary in the area of the PROFINET interface [X1]:
   - Enable the "IO device" option in the "Operating mode" area.
   - Configure the transfer areas in the "Operating mode" > "I-device configuration" area. The "Address in IO controller" column remains empty because no IO controller is assigned.
     Note: To change an input area to an output area, and vice versa, you must navigate to the area of the corresponding transfer area.

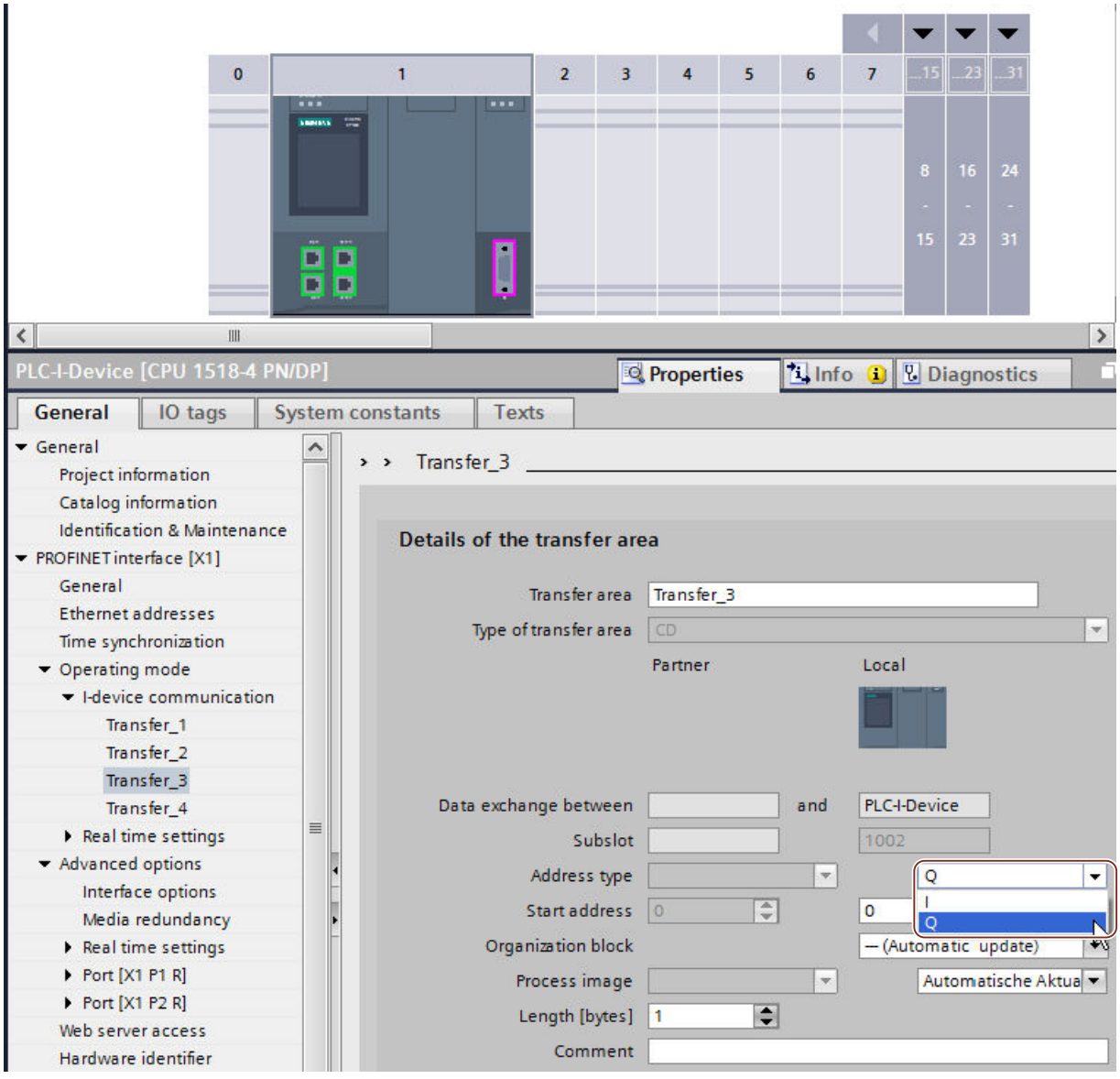

   - Select the number of IO controllers that will access the shared I-device during operation ("Mode" > "Real time settings" area, "Shared Device" section).

6. Save the project.
7. Click the "Export" button ("Mode" > "I-device configuration" area, "Export general station description file (GSD)" section).
   If you do not change the name in the Export dialog, the GSD file has a name in the form "GSDML-V2.31-#Siemens-PreConf_PLC-I-Device-20130925-123456", for example.

Procedure - Creating the PLC_1 project

To create the first project with a shared I-device, follow these steps:

1. Start STEP 7.
2. Install the PROFINET GSD file from the export of the I-device CPU (PLC-I-Device).
3. Create a new project with the name "PLC_1".
4. Insert a CPU 1516-3 PN/DP in the network view. The name of the CPU should be "PLC_1".
5. Insert the I-device from the hardware catalog (Hardware catalog: Other field devices > PROFINET IO > PLCs & CPs).
6. Assign the IO controller "PLC_1" to the I-device.
7. Select the "Shared Device" area in the properties of the I-device.
   In the table, all transfer areas and the PROFINET interface are assigned to the local IO controller (PLC_1).
8. Define the transfer areas to which the PLC_1 CPU should **not** have access. Select the "---" entry for these areas. These transfer areas are provided for PLC_2.
9. Save the project.

Procedure - Creating the PLC_2 project

To create the second project with a shared device, follow these steps:

1. Start STEP 7 once again.
   A new instance of STEP 7 opens.
2. In the new instance, create a new project with the name "PLC_2".
3. Insert a CPU 1516-3 PN/DP in the network view. Assign the name "PLC_2".
4. Insert the I-device from the hardware catalog (Hardware catalog: Other field devices > PROFINET IO > PLCs & CPs).
5. Assign the IO controller "PLC_2" to the I-device.
6. Adapt the access to the transfer areas as in the PLC_1 project. Ensure that no duplicate assignments result.
7. Adapt the parameters of the subnet and PROFINET interface. Because the shared I-device involves the same device in different projects, these data must match.
8. Save the project.

Both projects now have an identically configured shared I-device. The IO controller access and the parameters of the PROFINET interface should still be checked in the different projects during the next step.

Summary - Assigning accesses to the shared device

The transfer areas are automatically assigned to the local IO controller. To change the assignment, follow these steps:

1. Click the "PLC_I-Device" device in the network view of the "PLC_1" project, and select the "Shared Device" area.
2. A table shows which CPU has access to each of the configured transfer areas. The default setting is that the local CPU has access to all modules and submodules.
3. Keep the setting "PLC_1" for all transfer areas that are to remain in the address range of the local CPU
   Select the setting "---" for all transfer areas that are to be located in the address range of the "PLC_2" CPU from the "PLC_2" project. This means that an IO controller outside the project is to have access to the transfer area.
4. Follow the same procedure for the remaining projects.
5. Finally, check whether the settings for access are "complementary" for each module or submodule in both projects. This means that if the local CPU has access in one project, the option "---" must be set in the other project and vice versa.
   Special note: The option "---" for the PROFINET interface and therefore for the ports has the effect that the associated parameters are deactivated and cannot be changed. Parameters of the PROFINET interface and port parameters can only be edited in the project in which the PROFINET interface is assigned to the local CPU. The ports can be interconnected in both projects regardless of this.
6. Check whether the same IP address parameters and device name are set for the shared device in all projects.
   Check whether the same S7 subnet ID is set in all projects for the subnet to which the shared device is connected (subnet properties, "General" area in the Inspector window).

Note

If you make changes to the I-device (e.g., change the number or length of the transfer areas):

Export the I-device as a GSD file again. Re-install the GSD file in each project that uses the I-device as a shared device. Make sure that only one IO controller has access to a transfer area.

Procedure - Adjusting the real-time settings

To ensure that all IO controllers and shared devices are operated with the appropriate send clock and that the update times are calculated correctly based on the communication load, you must adjust and check the following settings:

1. You must set the same send clock for each IO controller that has access to modules and submodules of the shared device:
   - If you configure the IO controller with STEP 7 (TIA Portal):
     Open the corresponding project.
     Select the PROFINET interface of the IO controller.
     Select the "Advanced options > Real time settings > IO communication" area in the Inspector window and set the shared send clock.
   - If you configure the IO controller with a different engineering tool:
     Select the PROFINET interface of the shared device in STEP 7 (TIA Portal) and read out the send clock on the shared device ("Advanced options > Real time settings" area)
     Enter the read send clock in the engineering tool.
   - Special note: If you configure **all** IO controllers that have access to the shared I-device in STEP 7 (TIA Portal or V5.5), you can set shorter send clocks on the IO controller than supported by the shared device (send clock adaptation).

Compiling and loading

You must compile the configurations for the different IO controllers and load them to the CPUs one after the other.

Due to the distributed configuration with separate projects, STEP 7 does not output consistency errors in the case of incorrect access parameter assignment. Examples for incorrect access parameter assignment:

- Several IO controllers have access to the same module
- IP address parameters or send clocks are not identical

These errors do not have an effect until operation and are signaled as configuration errors, for example.

See also

Configuring the I device (Page 2923)

Useful information on Shared Devices (Page 2961)

Module-internal shared input/shared output (MSI/MSO) (S7-300, S7-400, S7-1500)

Introduction

This section describes the module-internal shared input/shared output (MSI/MSO) functionality for the S7-1500 I/O modules that are operated in an ET 200MP on PROFINET.

Module-internal shared input/shared output functionality

The module-internal shared input (MSI) function allows an input module to make its input data available to up to four IO controllers. Each controller has read access to the same channels.

The module-internal shared output (MSO) function allows an output module to make its output data available to up to four IO controllers. One IO controller has write access. The other IO controllers have read access to the same channels.

The following figure shows the MSI/MSO functionality.

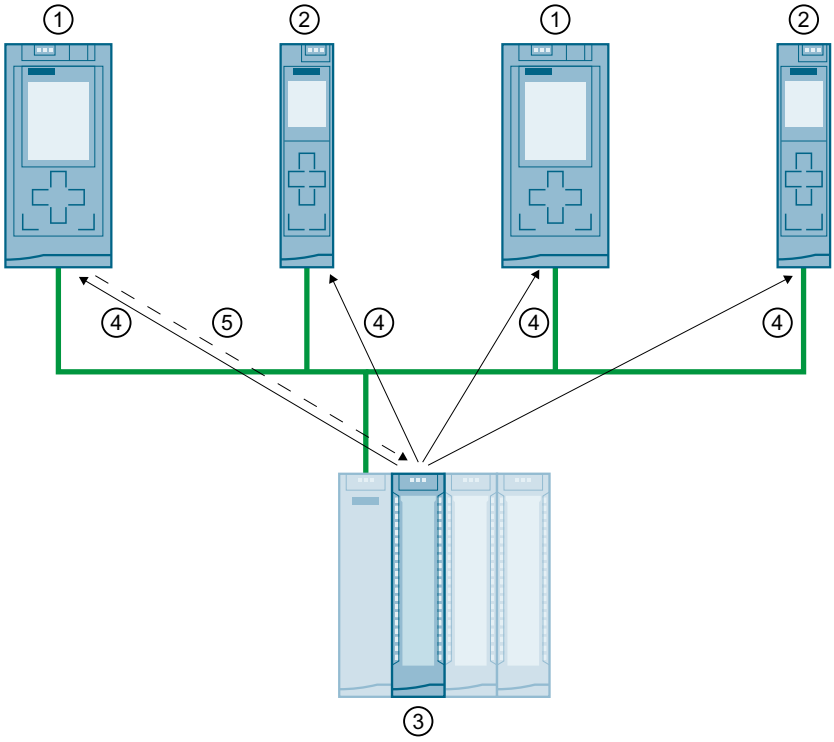

① CPU 1516-3 PN/DP as IO controller
② CPU 1511-1 PN as IO controller
③ I/O module with MSI/MSO
④ Read access to the channels of the I/O module
⑤ Write access to the channels of the I/O module (only with MSO)

Advantages of MSI/MSO

Module-internal shared input/shared output (MSI/MSO) offers the following advantages:

- Real-time acquisition in multiple CPUs
- Lower costs due to saving on additional IO devices and modules
- Lower space requirements due to saving on additional IO devices and modules
- Reduced communication load because no CPU-CPU communication is needed
- No additional programming effort is needed for CPU-CPU communication

Requirements for the use of MSI/MSO

Observe the following requirements:

- MSI/MSO can only be used with PROFINET IO
- Configuration software: STEP 7 (TIA Portal) V12 SP1 or higher with GSD file.
- The IM 155-5 PN ST interface module and the modules support MSI/MSO firmware version V2.0.0 or higher.

Boundary conditions for the use of MSI/MSO

Note the following boundary conditions:

- MSI/MSO cannot be used for modules with channel groups.
- Modules with MSI/MSO cannot be operated in isochronous mode.

MSI submodules

The input values of all channels are copied to a basic submodule and up to three other MSI submodules during MSI configuration of an input module. The channels of the module are then available with identical input values in the basic submodule and the MSI submodules. The submodules can be assigned to up to four IO controllers when the module is used in a shared device. Each IO controller has read access to the same channels.

The following figure shows a digital input module with the basic submodule and three MSI submodules. Each submodule is assigned to an IO controller. Diagnostics and parameter assignment of the digital input module can be performed from the IO controller 1 via the basic submodule.

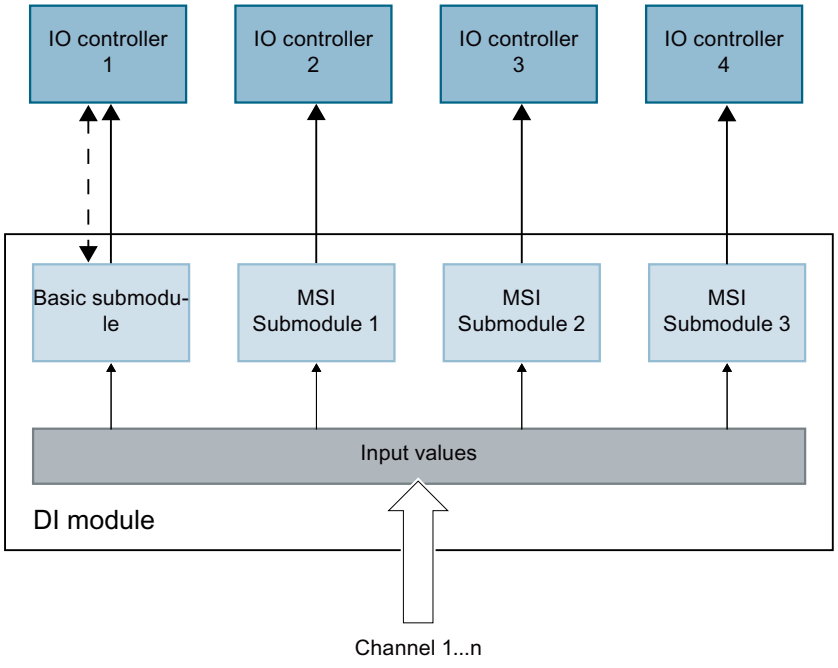

Value status of the MSI submodule (Quality Information, QI)

The meaning of the value status depends on the submodule to which it pertains.

For the basic submodule (first submodule), the value status 0 indicates that the value is incorrect.

For an MSI submodule (submodules 2 to 4), the status value 0 indicates that the value is incorrect or the basic submodule parameters have not yet been assigned (not ready for operation).

MSO submodules

The output values of all channels of the module are copied from a basic submodule to up to three other MSO submodules during MSO configuration of an output module. The channels of the module are then available with identical values in the basic submodule and the MSO submodules. The MSO submodules can be assigned to up to four IO controllers when the module is used in a shared device:

- The IO controller to which the basic submodule is assigned has write access to the outputs of the module. The basic submodule therefore occupies output addresses in the process image of the IO controller.
- The IO controllers to which the MSO submodules are assigned have read access to the outputs of the module. MSO submodules therefore occupy input addresses in the process image of the IO controller.

The following figure shows a digital output module with the basic submodule and three MSO submodules. Each submodule is assigned to an IO controller. Diagnostics and parameter assignment of the digital output module can be performed from IO controller 1 via the basic submodule.

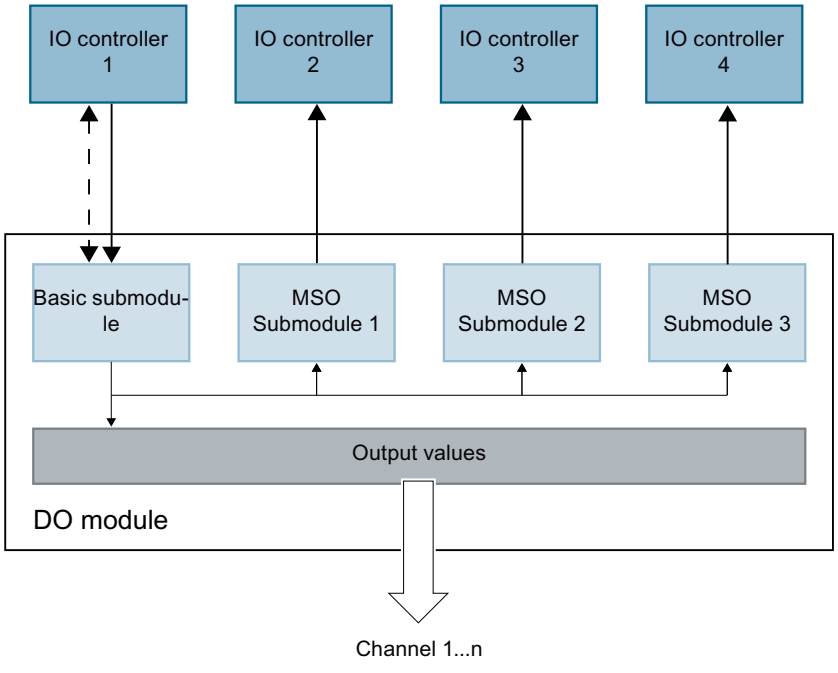

Value status of the MSO submodule (Quality Information, QI)

The meaning of the value status depends on the submodule to which it pertains.

For the basic submodule (first submodule), the value status 0 indicates that the value is incorrect.

For an MSO submodule (submodules 2 to 4), the status value 0 indicates that the value is incorrect or one of the following errors has occurred:

- The basic submodule parameters have not yet been assigned (not ready for operation).
- The connection between the IO controller and the basic submodule has been interrupted.
- The IO controller of the basic submodule is in STOP or POWER OFF state.

Configuring I/O modules with MSI/MSO submodules

Requirement

- Configuration software STEP 7 V12 Service Pack 1 or higher
- GSD file for I/O modules with MSI/MSO.
- IO device supports MSI/MSO (e.g., IM 155-5 PN ST firmware version V2.0.0 or higher)

Procedure

1. In the network view of STEP 7, insert an interface module as of IM 155-5 PN ST V2.0.
2. Double-click the IO device.
   You are now in the device view.
3. Place the modules with MSI/MSO from the hardware catalog in a suitable slot.
4. Configure the modules as described in the product information for the module.

Creating a standard machine project

Overview

Introduction

Standard machine projects are STEP 7 projects that use a set of innovative functions allowing simple configuration and commissioning of flexible automation solutions for standard machines or for machines with a modular structure.

A hardware configuration consisting of an S7-1500 CPU as the IO controller and any connected IO devices represents a "PROFINET IO system master". This master is configured with a maximum configuration based on which various options can be derived for different standard machines, for example with different configuration variants of the IO system.

Greater flexibility at all levels

Standard machine projects have the following central characteristics:

- From one project (IO system master) with an engineered maximum configuration, different variants of a standard machine can be loaded (IO system options). The standard machine project covers all variants (options) of the IO system.
- An IO system option can be integrated in an existing network locally using simple tools.

Flexibility is provided in more ways than one:

- With suitable configuration, adaptation of the IP address parameters of the IO controller is possible locally using simple tools. This allows a standard machine to be integrated in different plants with little effort or to be included in a network several times.
  IO systems with this property are known as **multiple use IO systems**.
- With suitable configuration and programming, different setups of IO system options can be operated locally that differ in terms of the selection of IO devices used or in terms of the arrangement of the IO devices.
  Since the specific configuration of the IO system is controlled by the user program, this is known as **configuration control for IO systems**.
- Independently of the functions described above, with suitable configuration and programming, you can use different station options of central devices or distributed I/O devices in one project. The devices can be different in terms of the selection and arrangement of the modules.
  Since the concrete configuration of the station is controlled by the user program, this is also known as **configuration control**.

Multiple use PROFINET IO systems

What you should know about multiple use IO systems

Multiple use automation solutions

For a flexible reusable automation solution as is the case with series machines, the following use cases are typical:

- The machine (and therefore also the PROFINET IO system) is used more than once by the customers.
- The machine is used in different plants by various customers.

For this to be possible, the automation solution must meet the following requirements:

- A configuration and program can be loaded on various machines of the same type without changes.
- Only a few easy adaptations are needed locally to integrate the machine into an existing network infrastructure.

The following figure shows how an automation solution with a multiple use IO system is loaded on different automation systems and then one automation system is adapted to the existing network infrastructure locally.

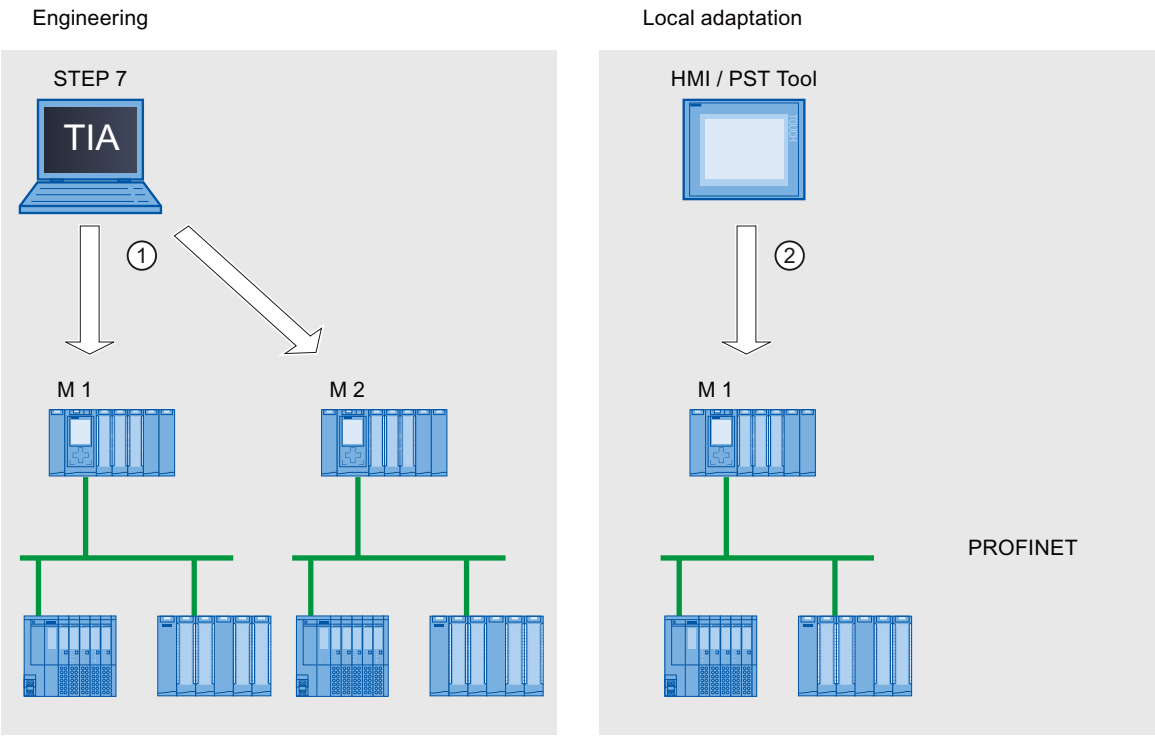

① Load configuration with multiple use IO system

② Set IP address and device name locally on the IO controller

Principle

The automation components for a machine include a PROFINET IO system, consisting of an IO controller (PROFINET interface of a CPU) and the IO devices assigned to it.

With the "Multiple use IO system" setting for the IO system, you turn a STEP 7 project into a "Standard machine project".

The "Multiple use IO system" setting triggers various settings and checks of the configuration by STEP 7. The settings ensure that the IO system is self-contained and there are no dependencies on components outside the IO system.

Requirement

- STEP 7 V13 or higher
- S7-1500 CPU or ET 200SP CPU as of firmware version V1.5 as IO controller

Rules

The following rules apply to a multiple use IO system:

- No IO device can be configured as Shared Device.
- The ports of the devices must be interconnected.
  Devices for which no port interconnection is configured, for example, interface module IM 154-6 IWLAN (ET200pro PN), cannot be operated with STEP 7 V13 as IO devices on a multiple use IO system.
- If an IO device in a multiple use IO system is an I-device (CPU as an "intelligent" IO device): If the I-device has a lower-level IO system, this lower-level IO system can be connected to the same PROFINET interface as the higher-level IO controller. However, only one of the two IO systems supports multiple use according to the following rule:
  "Parameter assignment of PN interface by higher-level controller" (on I-device) = **yes**: Multiple use higher-level IO system.
  "Parameter assignment of PN interface by higher-level controller" (on I-device) = **no**: Multiple use lower-level IO system.

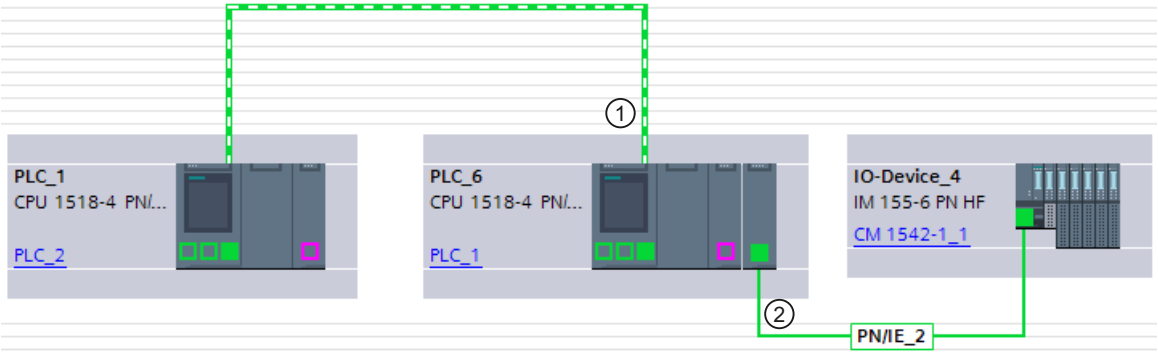

① I-device in multiple use IO system. The PROFINET interface is operated as IO device.

② A lower-level IO system on the I-device is also connected to this PROFINET interface.

- If MRP (Media Redundancy Protocol) is configured:
  - All IO devices on the multiple use IO system must belong to the same MRP domain.
- If IRT (Isochronous Real Time) is configured:
  - All IO devices in multiple use IO systems must belong to the same sync domain.
  - The sync domain must not include any other IO devices.
- IE/PB links cannot be operated as IO device with STEP 7 V13 in a multiple use IO system. IE/PB Links with firmware version V3 or higher can be operated in a multiple use IO system with STEP 7 V13 SP1 or higher.

Configuration

You specify whether or not a configuration can be used multiple times in the properties of the IO system.

All other parameter settings for the configured devices are then set automatically by STEP 7 and checked during compilation.

Boundary conditions

To prevent a standard machine project from having dependencies on other devices outside of the machine, observe the following:

- A standard machine project consists of an IO controller and the corresponding IO devices. You should therefore configure only one CPU as IO controller and the corresponding IO devices in the standard machine project.
- Do not use two-ended connections for the communication. Instead, use only one-ended connections or unspecified connections if necessary.
  Background: To configure the communication in a STEP 7 project, it is always possible to set the IP address parameters in the project. For multiple use IO systems, however, this strategy is not possible since the IP address parameters of the IO controller and the assigned IO devices are assigned locally. At the time of the configuration, the IP address parameters are therefore unknown.
  If you nevertheless want to configure communication with devices on PROFINET, for example with a central coordinator, you can only use communications mechanisms that allow dynamic assignment of the IP address parameters in the user program.
  Example: Open User Communication
  If, for example, the device is configured as an active end point (initiator of the connection), the IP address parameters can be stored, for example, in a data block. You then supply the data block with the currently valid IP address parameters during commissioning. For this dynamic type of IP address parameter assignment, there is no system support; in other words, if you change the configuration of the system, the IP address parameters are not automatically adapted.
  You will find a description of handling instructions for Open User Communication under this keyword in the STEP 7 online help.

See also

Unspecified connections (Page 203)

Specifying an unspecified connection (Page 229)

Creating a new connection - overview (Page 206)

Configuring a multiple use IO system

Requirement

- S7-1500 CPU firmware version V1.5 or higher as IO controller

Procedure

To create a standard machine project, follow these steps:

1. Create a project.
2. Configure an S7-1500 CPU firmware version V1.5 or higher as IO controller.
3. Configure the required IO devices and assign the IO devices to the IO controller.
4. Configure the port interconnection between the devices.

5. Select the IO system so that you can edit the properties in the inspector window.
6. Select the "Multiple use IO system" check box in the "General" area of the inspector window.

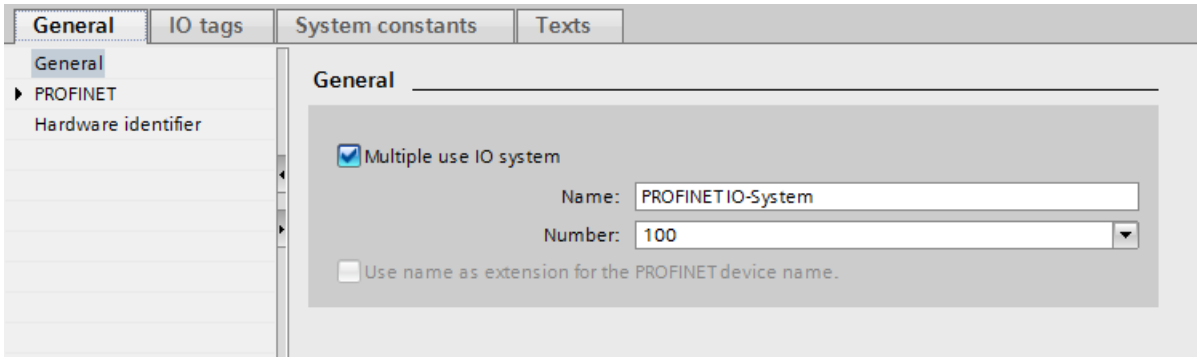

Result: The following settings are made by STEP 7:

- The device name of the IO controller (CPU) in the standard machine project is set to "Permit setting of PROFINET device name directly on the device". The IO controller (CPU) has no PROFINET device name initially.
- The IP protocol of the IO controller (CPU) is permanently set to "IP address is set directly at the device". The CPU has no IP address initially.
- The "Support device replacement without exchangeable medium" option is selected automatically. This option enables automatic commissioning. A commissioning engineer does not have to assign device names to the IO devices locally. The IO controller assigns the device name and IP address to the IO devices based on the preset topology and other settings during startup.
- The device name of the IO devices is set to "Generate PROFINET device name automatically" (from the configured name of the IO device).
- The IP protocol of the IO devices is set to "IP address is set by the IO controller during runtime". The IO devices have no IP address initially. If an IO device is not a typical distributed I/O system (e.g., ET 200 systems), but rather another device such as an HMI device, change the option to "IP address is set directly at the device"; see below.
- The device number for the IO devices is automatically assigned and is used locally for making the IP address unique.

To allow the IO controller to make later changes to the device name for the operator, you must enable the CPU option "Permit overwriting of device names of all assigned IO devices" ("Advanced options > Interface options" in the PROFINET interface properties).

This option is disabled by default.

The following figure shows the above-described settings for the IP address and PROFINET device name.

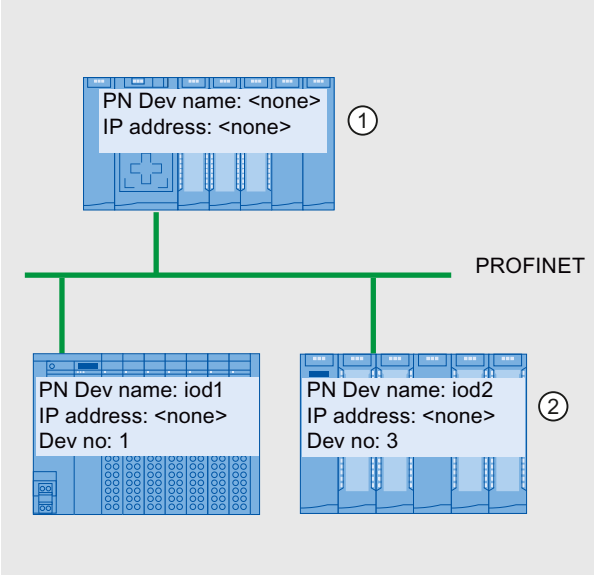

① After the configuration is loaded from the standard machine project, the IO controller has no device name and no IP address.

② Following loading, the IO devices have a device name and a device number but no IP address.

How an IO device obtains an IP address locally

Below, you will find an explanation of the "IP address is set by the IO controller during runtime" and "IP address is set directly at the device" options, which can generally be configured for a multiple use IO system.

If you have set the "Multiple use IO system" option for the IO system, STEP 7 automatically sets the "IP address is set by the IO controller during runtime" option for the IO devices.

In this case, the IO controller assigns the IO device an IP address that results from the locally assigned IP address of the IO controller (see next section). This option is appropriate if the IO device is a field device, e.g., ET 200MP, ET 200SP, or another distributed I/O system.

If the IO device is not a "standard" field device, for example, an HMI device for a Windows operating system, the "IP address is set by the IO controller during runtime" option described above does not work. In this case, choose the "IP address is set directly at the device" option. You must then assign the IP address to the device locally and take steps to ensure that this address is suitable for the IP addresses of the other IO devices and the IP address of the IO controller.

See also

Adapting the IO system locally (Page 2987)

Adapting the IO system locally

A few steps are needed to adapt the machine that was loaded with the standard machine project.

Only the device name and IP address of the IO controller must be adapted locally. The device names and IP addresses of the IO devices result from these adaptations. In this example, the effects of local settings are described for two specific machine modules.

The local settings are possible, for example, with the CPU display and commissioning tools such as Primary Setup Tool (PST) or PRONETA. You do not need a programming device with STEP 7 to make these settings.

Requirement

- The machine was loaded with a standard machine project (see Configuring a multiple use IO system (Page 2985)).
- The display is ready for operation and the desired tool for assigning the IP address and device name is available (e.g., PST Tool, STEP 7).
- The ports of the IO controller and IO devices are interconnected according to the configuration.

Procedure

Observe the boundary conditions and instructions for commissioning an S7-1500. Refer to "See also" for a reference to the S7-1500 system manual in which commissioning is described.

To adapt a standard machine locally, follow these steps:

1. Integrate the machine into the network.
2. Connect the device for assigning the IP address and device name to the CPU, for example a PG/PC with the appropriate software.
3. Assign the desired device name and IP address to the IO controller.

The IO controller (CPU) then assigns the adapted PROFINET device name and a unique IP address to the IO devices.

The following rules apply to the assignment:

- The device names of the IO devices are formed by chaining together the following name components, separated by a period:
  <configured name of the IO device from the standard machine project>.<name of the associated IO controller set on the device>
- The IP addresses of the IO devices result from the locally configured IP address of the associated IO controller and the device number (sum).

Note

Make sure that duplicate IP addresses cannot be created on the subnet during the assignment. The IO controller does not assign a new IP address in this case.

In the following figure, the device name "m1" and the IP address 192.168.1.10 have been assigned to the IO controller of the first machine.

The device name "m2" and the IP address 192.168.1.20 has been assigned for the second machine.

Refer to the figure for the resulting device names and IP addresses.

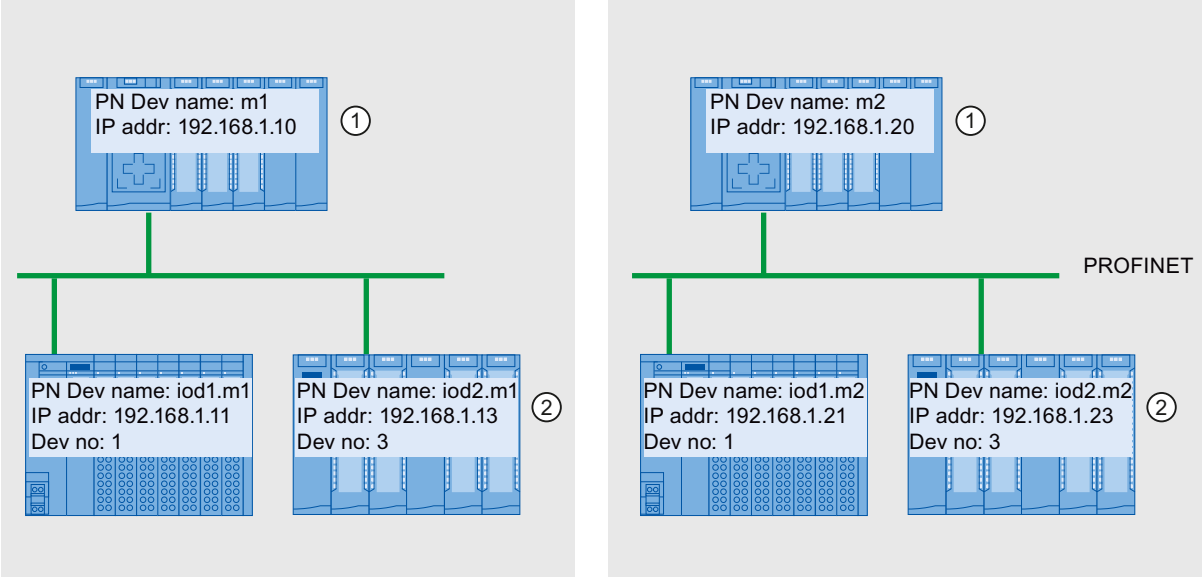

① Set device name and IP address on the IO controller

② After startup, the IO devices have an updated device name (<configured device name>.<device name of IO controller>) and an adapted IP address (= <IP address of IO controller> + <device number>)

See also

Configuring a multiple use IO system (Page 2985)

S7-1500 system manual (http://support.automation.siemens.com/WW/view/en/59191792)

Permit overwriting of PROFINET device name

With S7-1500 CPUs as of firmware version V1.5, you can overwrite PROFINET device names of IO devices during the startup of the CPU. This option reduces the effort during automatic commissioning, for example, when replacing a device.

How the "Permit overwriting of device names of all assigned IO devices" option works

The IO controller (CPU) can overwrite the PROFINET device names of IO devices in the IO system when the "Permit overwriting of device names of all assigned IO devices" option is enabled.

Multiple use IO systems can only be operated when this option is enabled. The IO controller checks prior to overwriting if the type of the IO device matches the configured type.

You must not use the option for partial commissioning (IO devices are not present or bridged in some cases).

CAUTION

Error after partial commissioning

If the device names were incorrectly assigned after a partial commissioning or incorrect wiring, you must correct the wiring and manually delete the device names in order to achieve a correct assignment.

WARNING

Incorrect PROFINET device names from the configuration through incorrectly connected devices

Depending on the connected I/O, there is risk of death, severe injuries or damage caused by malfunctions when incorrect PROFINET device names are entered through incorrectly connected devices.

To rule out any danger, check whether the right replacement device has been connected in the case of a device replacement and whether the port interconnection matches the configured preset topology.

If the "Permit overwriting of device names of all assigned IO devices" option is not selected, the IO controller cannot overwrite the device names of the IO devices. In this case, you must manually assign the PROFINET device name on the IO device if the PROFINET device name changes in the configuration or when a device is replaced, or delete the device names of the IO devices prior to an automatic commissioning.

Typical source of danger

When replacing an IO device ("standard case"), it is almost guaranteed that the replaced device will be connected according to the configured port interconnection.

The figure below shows a scenario whereby the connections of two identical PROFINET cables are swapped at two switch ports. Because the IO controller assigns the device names according to the preset topology, the incorrect connection of the devices has serious effects on the naming.

Due to the control of different actuators, the plant could become hazardous in this case.

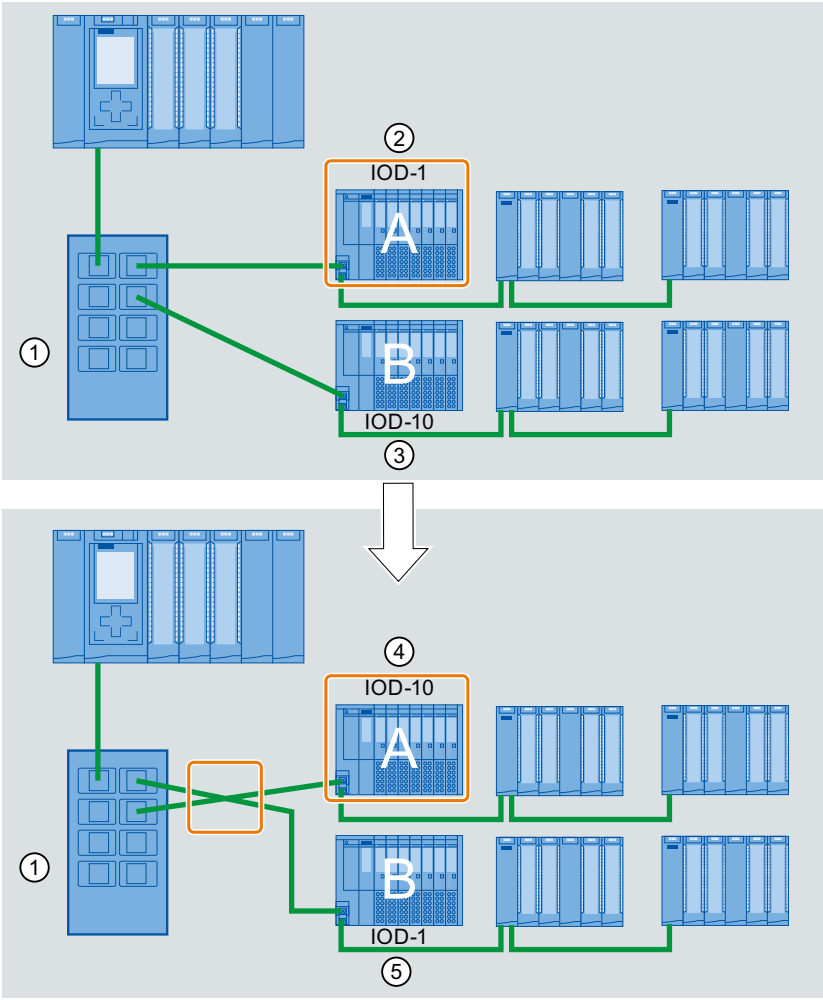

① Switch with connected PROFINET cables
② Device A, device name "IOD-1": controls motor 1
③ Device B, device name "IOD-10": controls motor 10
④ Device A controls motor 10
⑤ Device B controls motor 1

Procedure

Proceed as follows to change the "Permit overwriting of device names of all assigned IO devices" option:

1. Select the PROFINET interface of the CPU for which you want to change the option in the network view or in the device view.
2. Select "Advanced options > Interface options".
3. Change the option.

Configuration control (option handling) for PROFINET IO systems

Information about configuration control of IO systems

Configuration control of IO systems makes it possible to generate several concrete versions of a standard machine from a standard machine project.

You are given the flexibility to vary the configuration of an IO system for a specific application as long as the real configuration can be derived from the set configuration. The configured configuration therefore represents the superset of all real configurations that can be derived from it.

The following figure shows an example of how two PROFINET IO systems with a different number of IO systems arise from one standard machine project.

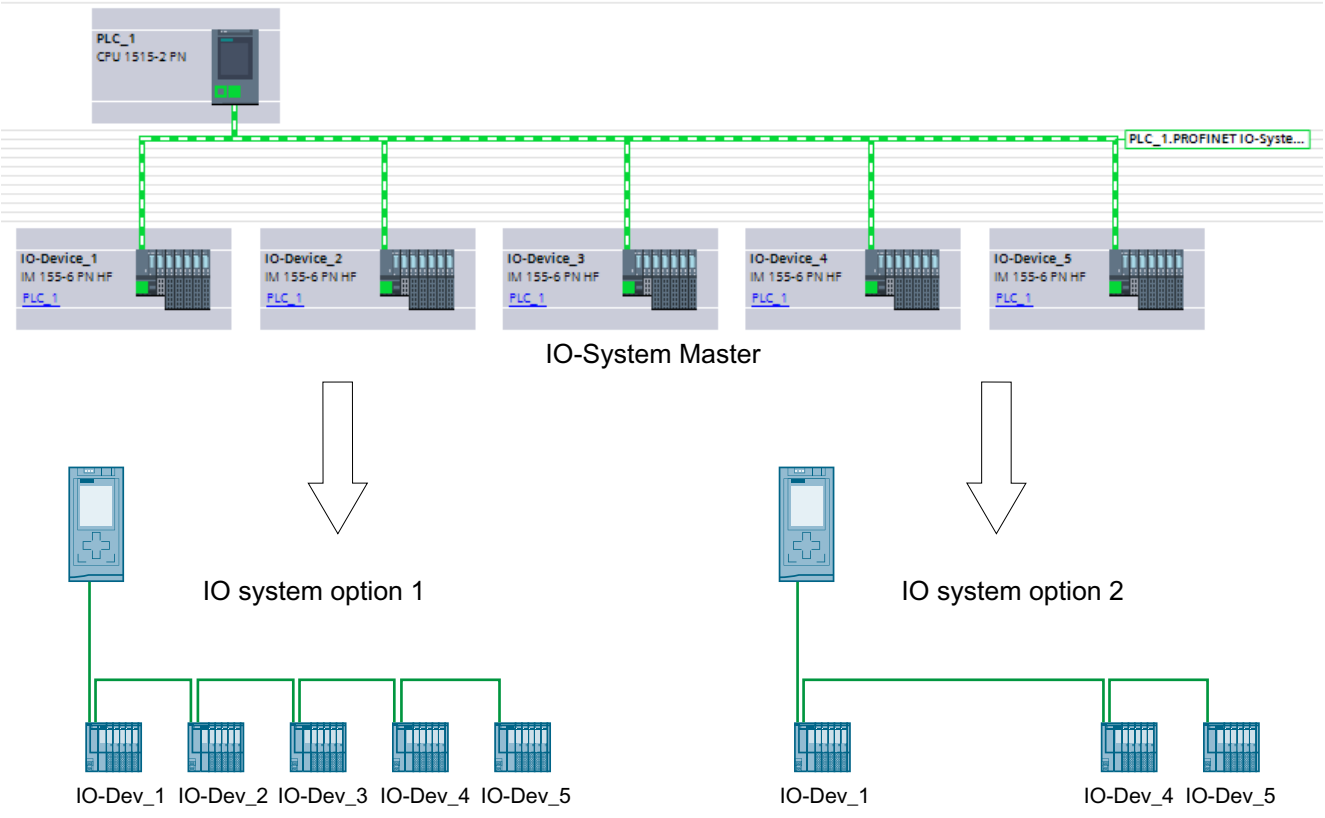

In the following sections, you find a description of how to configure and program a PROFINET IO system to commission, for example, a standard machine on-site without using configuration software.

Concept

The principle of configuration control is already known at the device level for the flexible use of submodules/modules ("option handling"). Different configurations can be derived from one engineering project both for central as well as for distributed I/O.

With S7-1500 CPUs as of firmware version V1.7, this principle can also be applied at the IO system level. You have the option of omitting, adding or changing the order of stations (IO devices) of a PROFINET IO system in a specific plant.

Configuration control for devices and configuration control for IO systems can be combined; the functions are independent of each other.

It is possible to operate variants deviating from a maximum configuration of an IO system. In a standard machine project, you can prepare a kit of IO devices which can be flexibly customized for various configurations using configuration control.

The following variations are available:

- Variation of the number of IO devices involved
  You include optional IO devices for the configuration control in the configuration by transferring a suitable data record with the required configuration in the user program.
- Variation of the order of IO devices involved
  You adapt the port interconnection of the IO devices to the topology being used by transferring a suitable data record with the required topology in the user program.

The following figure shows how serve two different configurations with an IO device marked as optional in the network view of STEP 7.

- Configuration without the optional IO device:
  In this case, you use the instruction "ReconfigIOSystem" to transfer a data record to the PROFINET interface containing the information that no optional IO device is to be included in the configuration.
- Configuration with the optional IO device:
  In this case, you use the instruction "ReconfigIOSystem" to transfer a data record to the PROFINET interface adding the optional IO device to the configuration.

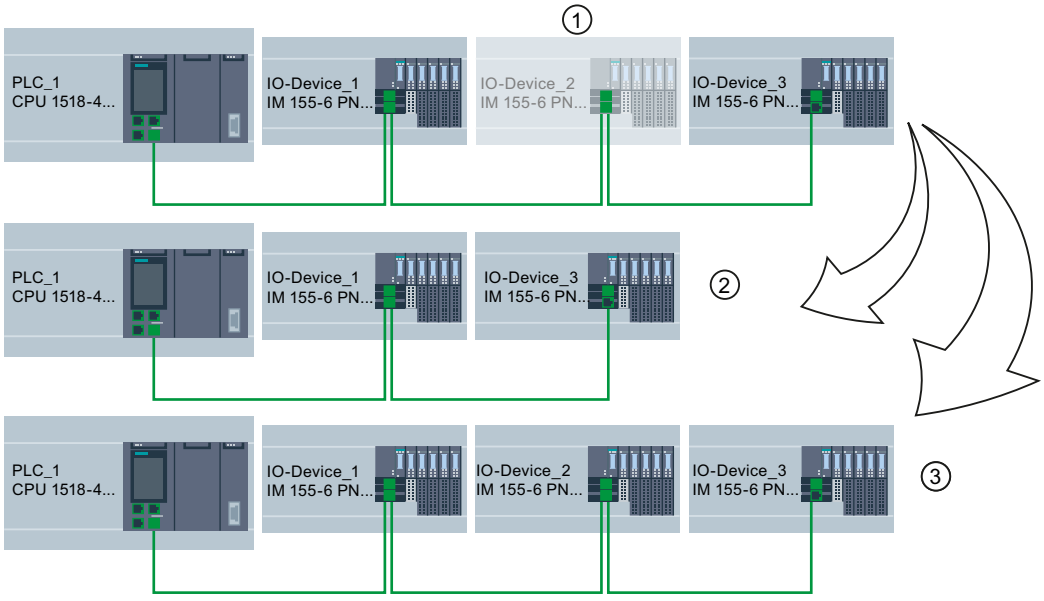

① Determined through parameter assignment: IO Device_2 is optional IO device

② Configuration without the optional IO device

③ Configuration with the optional IO device

Summary: The procedure in principle

The following phases are distinguished when it comes to the implementation of a standard machine concept:

1. Engineering phase: Creating a standard machine project and loading into specific machine or plant:
   - Completely configuring all IO devices (options) ever required in a specific machine or plant
   - Configuring as optional those IO devices that will be omitted in specific machines or plants
   - Preparing user program with the possibility of selecting on-site the actually existing configuration via switch or HMI device
2. Commissioning phase: Preparing specific machine or plant for operation:
   - Integrating machine or plant in the on-site network (see Adapting multiple use IO systems locally)
   - Selecting the currently existing configuration of the IO system via configured option

Configuring IO devices as optional

Requirement

- S7-1500 CPU firmware version V1.7 or higher as IO controller
- STEP 7 V13 SP1 or higher
- The rules (Page 3004) for the establishment and operation of a standard machine project have been considered.

Port interconnection for optional IO devices

As of STEP 7 V15.1, port interconnection is not necessary for optional IO devices.

A port interconnection between the devices of the IO system that you want to customize with the user program is mandatory in the following cases.

- You have configured IRT.
- You have configured MRP.
- You are using STEP 7 <= V15.

Procedure

To configure an IO device as optional IO device, proceed as follows:

1. Create a project.
2. Configure an S7-1500 CPU firmware version V1.7 or higher as IO controller.
3. Configure the required IO devices and assign the IO devices to the IO controller.

4. Select the IO device you want to mark as optional.
5. Select the area "PROFINET interface [X1] > Advanced options".
6. Enable the "Optional IO device" option.

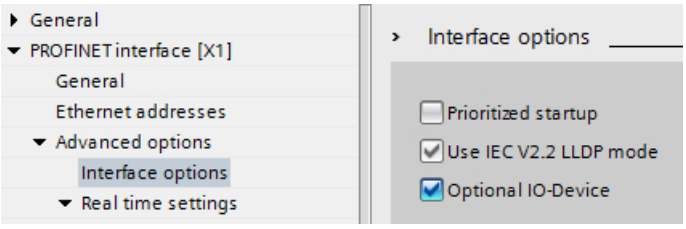

7. Repeat steps 4 to 6 for all IO devices to be configured as optional.
8. Load the configuration onto the CPU.

Result: Once this configuration is loaded, the system behavior is as follows:

- The CPU is prepared for the configuration control of the IO system.
- All IO devices are disabled.
- Irrespective of whether you customize the configuration with the user program (adding optional IO devices) or make no changes to the loaded configuration: You must call the instruction "ReconfigIOSystem" in the user program and notify the current configuration to the system!
  The system will not be operational without calling the instruction "ReconfigIOSystem".
  For further information on the proceeding see Enabling optional IO devices in the program (Page 2995).

Fast parameter assignment in the "I/O communication" table

You can also specify whether or not an IO device is optional in the "I/O communication" tab.

In an additional "Optional IO-Device" column, a selectable check box is available for each IO device that indicates whether or not an IO device is optional. Here, you can adjust the setting centrally.

Enabling optional IO devices in the program

Requirement

- CPU S7-1500 firmware version V1.7 or higher as IO controller.
- STEP 7 V13 SP1.
- At least one IO device was configured as optional IO device.
- The rules (Page 3004) for the establishment and operation of a standard machine project have been considered.

Procedure

Please note the information on and rules for commissioning an S7-1500 system in the documentation for SIMATIC S7-1500!

The following description of the proceeding only includes steps required to understand the program-controlled activation of an optional IO device.

To activate or deactivate optional IO devices, include them in the configuration and activate them in the user program, follow these steps:

1. Create a data record as described here.
2. Call the instruction "ReconfigIOSystem" and select mode 1 to deactivate all IO devices.
   If you set the CPU to STOP or POWER OFF state in order to modify the plant in this status (for example to add an optional IO device), explicit deactivation using "ReconfigIOSystem" with mode 1 is not necessary. In this case, i.e. following a STOP-RUN transition and following a POWER-OFF > POWER-ON transition, all IO devices are deactivated automatically.
3. When you have brought the plant to a safe status that allows restructuring without any danger:
   Put the plant together according to your intended application. Add the required optional IO devices at the points at which you planned this in the configuration (observe the order!) or remove optional IO devices that you no longer require.
4. Cable the IO devices.
5. Startup the S7-1500 system and call again the instruction "ReconfigIOSystem". Select mode 2 to transfer the data record CTRLREC.
6. Following successful transfer of the data record, call again the instruction "ReconfigIOSystem". Select mode 3 to activate all IO devices forming part of the current configuration.

Result

The CPU activates the following IO devices:

- All IO devices that you have not set as optional IO devices.
- All optional IO devices listed in the control data record (CTRLREC).

The following IO devices remain disabled:

- Docking units (IO devices changing during operation).
- Optional IO devices that are not listed in the control data record.

Note

Call the instruction "ReconfigIOSystem" in all modes with the same control data record (CTRLREC)!

If you use different data records in the different modes, this results in an inconsistent customization of the configuration and thus to rather incomprehensible error messages.

Example: Data record structure for the activation of an IO device

The IO device "IO-Device_2" is to be activated as the only IO device in the user program. To do this, you only require the hardware identifier of "IO-Device_2.

Recommendation: Use the system constants of the hardware identifiers instead of the absolute values as shown in this example. With this procedure, the content of the DB is not influenced by changes to the hardware identifiers as the result of changes to the configuration.

The data record is to be stored in a data block and to be transmitted to the PROFINET interface of the IO controller in the user program using the instruction "ReconfigIOSystem".

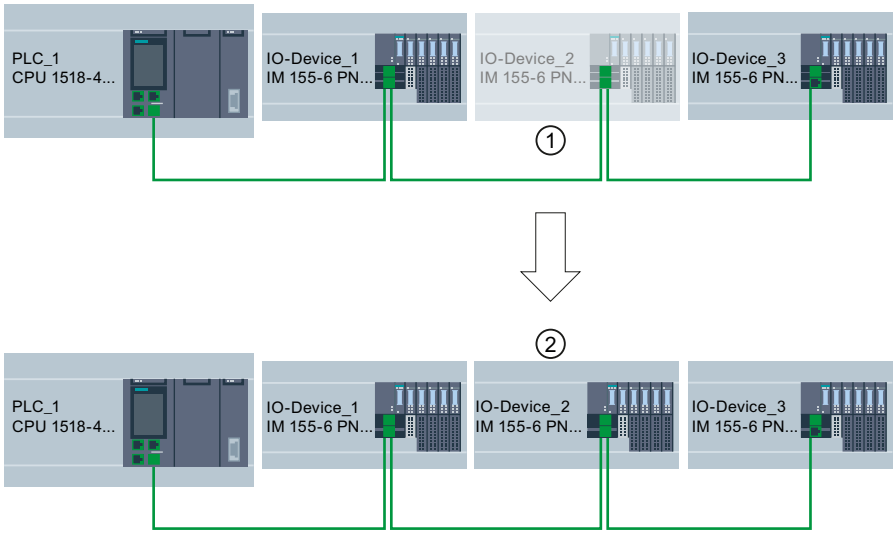

① IO device_2 is configured as optional IO device.

② Once the data record is transmitted and the configuration is activated using the instruction "ReconfigIOSystem" IO device_2 is included in the configuration and participates in the data exchange with the IO controller.

Creating data block

In this example, the control data record is created in a data block. The data block is structured as follows:

Line 2: Array definition: Array of type Word with 4 elements. Array of Word is permitted as the data type.

Line 3: Version of the data record (currently: V1.0).

Line 4: Number of optional IO devices to be activated (here: 1).

Line 5: List of the hardware identifiers of the IO devices, inserted here as a system constant.

Line 6: Number of port interconnections that are set in the user program (here: 0).

Line 7: Additional data records (optional).

myCTRLREC

	Name	Data type	Start value	Comment
1	Static			
2	ArrMachineConfig0	Array[0..3] of Word		Array for one optional IO device
3	ArrMachineConfig0[0]	Word	16#0100	
4	ArrMachineConfig0[1]	Word	16#0001	
5	ArrMachineConfig0[2]	Word	"IO-Device_2~IODevice"	
6	ArrMachineConfig0[3]	Word	16#0000	
7	ArrMachineConfig1	Array[0..8] of Word		

Rules for the call sequence of "ReconfigIOSystem"

- Always supply the instruction "ReconfigIOSystem" with the same control data record (CTRLREC input parameter)!
- Call sequence following POWER OFF -> POWER ON transition:
  - ReconfigIOSystem call with mode 1 (optional).
  - ReconfigIOSystem call with mode 2 (mandatory, even without previous reconfiguration!).
  - ReconfigIOSystem call with mode 3 (mandatory).
- Call sequence following STOP > RUN transition:
  - ReconfigIOSystem call with mode 1 (optional).
  - ReconfigIOSystem call with mode 2 (mandatory, even when configuration was modified in STOP state). Otherwise not required).
  - ReconfigIOSystem call with mode 3 (mandatory).
- Call sequence for reconfiguration in RUN state:
  - ReconfigIOSystem call with mode 1 (mandatory).
  - ReconfigIOSystem call with mode 2 (mandatory).
  - ReconfigIOSystem call with mode 3 (mandatory).

Explanations and recommendations concerning the rules

- If you do not list an IO device to be configured as optional IO device in the control data record or data block, this IO device does not form part of the configuration and does not take part in data exchange with the CPU.
- If you do not activate any optional IO device at all and work with the loaded configuration without reconfiguration, you still have to follow the proceeding described in the above section and transmit the control data record to the CPU.
  The control data record has the simple structure with the following tags:
  - Version (High Byte =1, Low Byte = 0)
  - Number of optional devices to be activated = 0
  - Number of port interconnections that are set in the user program = 0

- Following a STOP > RUN transition and following a POWER-OFF > POWER-ON transition, all IO devices are deactivated automatically. For this reason, no ReconfigIOSystem call with mode 1 is required for configuration control to function properly.
  If you use your project as a universally valid sample for programming the configuration control, we still recommend to perform the ReconfigIOSystem call with mode 1 prior to any reconfiguration. This way, the sample can also be used for reconfigurations in RUN mode.
- Commissioning extensive I/O systems (more than 8 optional IO devices) while using IRT at the same time:
  To keep the startup times short when activating the optional IO devices (ReconfigIOSystem, mode 3), note the following tip: Check the device numbers of the IO devices. The device numbers should follow the topological interconnection starting at the IO controller in ascending order. The further an IO device is from the IO controller topologically, in other words the more IO devices there are between the IO controller and the IO device in question, the higher the device number should be.
  You set the device numbers in the "Ethernet addresses - PROFINET" area in the Inspector window with the PROFINET interface selected.
  Example of the assignment of device numbers with a linear topology:

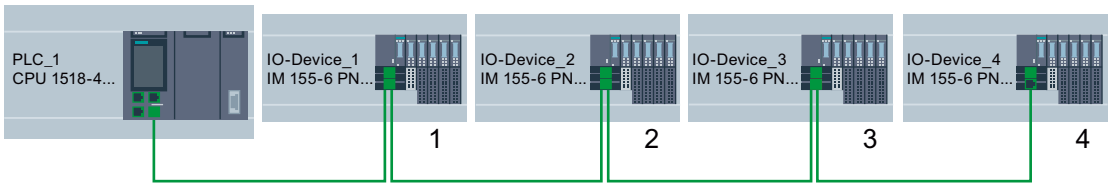

- The CPU processes the "ReconfigIOSystem" instruction for asynchronous transfer of the control data record.
  When you call the instruction in the startup program, you must therefore call "ReconfigIOSystem" repeatedly in a loop repeatedly until the "BUSY" or "DONE" output parameter indicates that the data record is transferred.
  Tip: Use the SCL programming language with the REPEAT ... UNTIL instruction for programming the loop.

```
REPEAT
    "ReconfigIOSystem"(REQ := "start_config_ctrl",
                MODE := 1,
                LADDR := 64,
                CTRLREC := "myCTRLREC".ArrMachineConfig0,
                DONE => "conf_DONE",
                BUSY => "conf_BUSY",
                ERROR => "conf_ERROR",
                STATUS => "conf_STATUS");
UNTIL NOT "conf_BUSY"
END_REPEAT;
```

Additional information

For information on the basic structure of the data record and on using the instruction "ReconfigIOSystem" see here.

See also

Configuring IO devices as optional (Page 2994)

Configuring flexible order of IO devices

The following section shows how you can create the conditions required to change the order of IO devices in a PROFINET IO system.

This function is also supported with optional IO devices. For simplicity, a maximum configuration without optional IO devices is shown below.

Concept

A typical application for a standard machine project consists of composing an entire plant from a set of various plant units which only differ with respect to the different arrangement of the units, e.g. in the case of transport systems. Each plant unit consists of a functional unit of mechanics (rails or conveyor belts) and electrics (power supply, IO device with IO modules, sensors, actuators, motors, PROFINET port for data exchange with central control ...).

The following figure shows how, simply by exchanging two rail segments, a new transport system is created that is adapted with an upstream points to the local conditions.

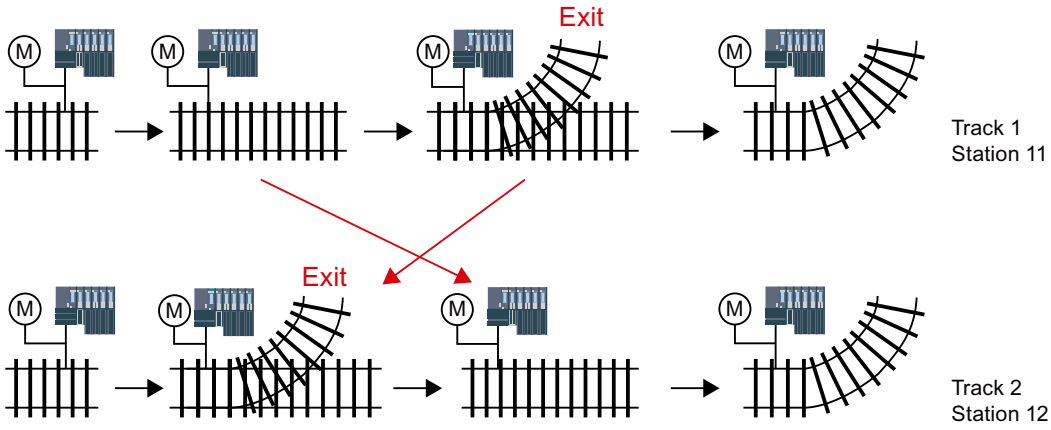

Interesting from the automation viewpoint is the implementation of a flexible order of individual IO devices in a project that is not to be modified any more on-site.

The order of the IO devices is determined by the port interconnection. For each IO device, you define in the port properties the partner port and thus the neighboring device connected at the respective local port. If the partner port is to be defined by the user program, the option "Partner set by user program" is to be selected as partner port.

The figure below shows the initial configuration of the transport system shown above, which is to permit the order of the connected IO devices to be changed via the user program. In the example, the order of IO-Device_2 and IO-Device_3 is to be controlled via the user program.

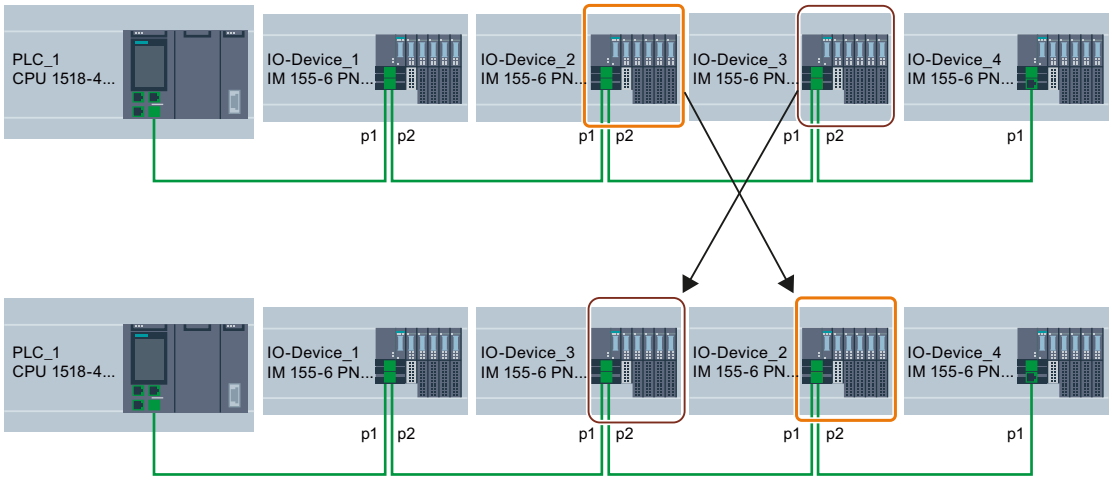

To determine how the partner port settings are to be selected, you must note for each device and each port of a device which partner can be interconnected.

- If the partner is always the same in the different configurations provided, you select the partner port for this partner.
- If the partners vary in the different configurations, you select "Setting partner by user program".

For the example in the figure above, the following port settings result:

Device	Local port	Partner port setting	Explanation
PLC_1	p1	p1 (IO device_1)	Partner of PLC_1 at port 1 is IO device_1 (always)
IO device_1	p1	p1 (PLC_1)	Partner of IO device_1 at port 1 is PLC_1 (always)
IO device_1	p2	Partner is set by user program	Partner of IO device_1 at port 2 is either IO device_2 or IO device_3 => Setting partner by user program
IO device_2	p1	Partner is set by user program	Partner of IO device_2 at port 1 is either IO device_1 or IO device_3 => Setting partner by user program
IO device_2	p2	Partner is set by user program	Partner of IO device_2 at port 2 is either IO device_3 or IO device_4 => Setting partner by user program
IO device_3	p1	Partner is set by user program	Partner of IO device_3 at port 1 is either IO device_2 or IO device_1 => Setting partner by user program
IO device_3	p2	Partner is set by user program	Partner of IO device_3 at port 2 is either IO device_4 or IO device_2 => Setting partner by user program
IO device_4	p1	Partner is set by user program	Partner of IO device_4 at port 1 is either IO device_3 or IO device_2 => Setting partner by user program
IO device_4	p2	any partner	no partner planned at port 2

Requirement

- S7-1500 CPU firmware version V1.7 or higher as IO controller.
- STEP 7 V13 SP1 or higher.
- The rules (Page 3004) for the establishment and operation of a standard machine project have been considered.

Procedure

To set the partner port for a program controlled interconnection, proceed as follows:

1. Select the PROFINET interface of the device (IO controller or IO device) whose port you want to set.
2. In the properties of the PROFINET interface, select the area "Port interconnection" (Extended options > Port [...] > Port interconnection).
3. From the drop-down list, select "Setting partner by user program" as partner port.
4. Repeat steps 1 to 3 for each port to be interconnected via the user program.

See also

Customizing arrangement of IO devices in the program (Page 3002)

Customizing arrangement of IO devices in the program

Requirement

- CPU S7-1500 firmware version V1.7 or higher as IO controller.
- At least one partner port was configured as "Partner set by user program".
- The rules (Page 3004) for the establishment and operation of a standard machine project have been considered.

Procedure

The proceedings corresponds to the proceeding for activating optional IO devices.

Only the structure of the data record must be extended for the program-controlled assignment of the ports. The extension is described in the following sections.

Example: Data record structure for the assignment of partner ports

For the data record structure, you need the HW identifications of the ports.

The data record is to be stored in a data block and to be transmitted to the PROFINET interface of the IO controller in the user program using the instruction "ReconfigIOSystem".

As the input parameter RECORD of the instruction "ReconfigIOSystem" is of the VARIANT data type, you first have to create a data type for the data block.

In the following sections, you find a description of the structure of the PLC data type as well as of the structure of the data block based on this type.

Selecting derived configuration

For the following selected configuration it is shown below what the data record must look like so that the IO devices are interconnected in the planned order by the user program.

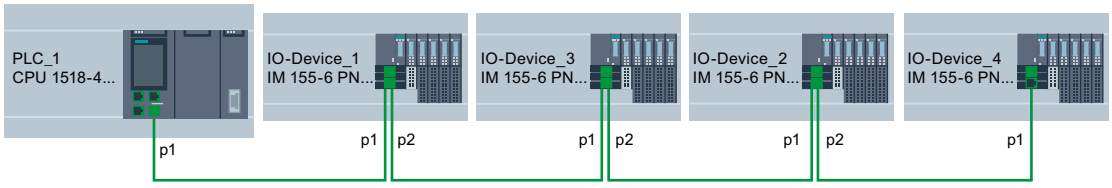

This example is based on the flexible configuration from the previous section (Page 3000) with the settings for the respective partner ports described there.

The partner ports in the specific derived configuration have been defined so that it is possible to name the HW identifications of the ports involved.

The following table only contains those devices whose ports can be defined by the user program. Only these devices are relevant for the data record structure.

Device	Local port	Partner port setting	Partner port of the selected configuration
IO device_1	p2 = Port 2 HW identifier: 251	Partner is set by user program	Port 1 of IO device_3 HW identifier: 261
IO device_2	p1 = Port 1 HW identifier: 281	Partner is set by user program	Port 2 of IO device_3 HW identifier: 291
IO device_2	p2 = Port 2 HW identifier: 311	Partner is set by user program	Port 1 of IO device_4 HW identifier: 321

Creating data block

For the derived configuration, the structure of the data block "DB-IO-SYSTEM-Port-Interconnections" is explained as an example.

This data block is used when calling the instruction "ReconfigIOSystem" at input parameter "CTRLREC".

Instead of the absolute values for the hardware identifiers of the ports, the system constants of the hardware identifiers are used here.

The data block is structured as follows:

Line 2: Declaration of an Array of Word (only this data type is possible).

Line 3: Version of the control data record: V1.0.

Line 4: Number of optional IO devices: 0.

Line 5: Number of specified port interconnections: 3.

Line 6: Port interconnection 1, local port.

Line 7: Port interconnection 1, partner port.

Line 8: Port interconnection 2, local port.

Line 9: Port interconnection 2, partner port

Line 10: Port interconnection 3, local port.

Line 11: Port interconnection 3, partner port.

IO-SYSTEM-Port-Interconnections

	Name	Datentyp	Startwert	Kommentar
1	Static			
2	ArrMachineConfig	Array[0..8] of Word		data record for port interconn settings
3	ArrMachineConfig[0]	Word	16#0100	
4	ArrMachineConfig[1]	Word	16#0000	
5	ArrMachineConfig[2]	Word	16#0003	
6	ArrMachineConfig[3]	Word	"IO_device_1~PROFINET_interface~Port_2"	
7	ArrMachineConfig[4]	Word	"IO_device_3~PROFINET_interface~Port_1"	
8	ArrMachineConfig[5]	Word	"IO_device_2~PROFINET_interface~Port_1"	
9	ArrMachineConfig[6]	Word	"IO_device_3~PROFINET_interface~Port_2"	
10	ArrMachineConfig[7]	Word	"IO_device_2~PROFINET_interface~Port_2"	
11	ArrMachineConfig[8]	Word	"IO_device_4~PROFINET_interface~Port_1"	

Interconnection not listed in data block

If the partner port was configured as "Setting partner by user program" in the port properties and this port is not listed in the data record or data block resp., then the CPU sets this port to the setting "any partner". If no data record is transmitted at all, the CPU sets this "any partner" setting for all program-controlled assignments.

The "any partner" setting is not valid for IRT configurations.

Additional information

For information on the basic structure of the data record and on using the instruction "ReconfigIOSystem" see here.

System behavior and rules

Below, you find a description of how an IO system whose configuration is controlled by the user program behaves in operation.

In addition, rules and restrictions are listed here which must be considered when configuring the maximum structure of the configuration in a standard machine project.

System behavior

- System diagnostics:
  From the system diagnostics viewpoint (online view or Online & Diagnostics) a deactivated IO device is shown as "deactivated".
- Topology view:
  Offline view: As configured. No interconnection is shown for ports with partner ports configured as "Setting partner by user program".
  Online view: Ports and interconnections with deactivated IO devices are shown in a different shade of green as error-free ports and interconnections of activated IO devices.
- Representation in the Web server:
  The names of devices are shown as configured (Properties > General > Project information)
  The assigned PROFINET device name for the CPU is shown on the "Communication" website, at the "Parameter" tab.
  IP address parameters: Currently assigned IP address parameters are shown on the site "Module state".
  Topology: The current topology resulting from any customizations via user program is shown in the Web server. IO devices configured as optional are shown as "deactivated" IO devices in the Web server.

Rules

The rules for standard machine projects as described here (Page 2982) apply.

For configuration-controlled IO systems, the following additional rules apply:

- When configuring MRP (Media Redundancy Protocol):
  The ports configured as ring ports must not be interlinked via user program.
  However, devices with ring ports (devices of an MRP domain) can be optional IO devices.
- When configuring docking stations (= IO devices changing during operation):
  Neither the docking station nor the first IO device of a docking unit may be optional IO devices.
  The ports of the docking units must not be interlinked via user program.
- When configuring IRT:
  The order of synchronized IO devices ("IRT devices") must be defined by the configuration and must not be changed in the different variants of a standard machine. For this reason, the ports of the IRT devices must not be interlinked via user program.
  However, you have the possibility to configure IRT devices as optional IO devices.
  You also have the option to interconnect, by user program, RT devices that are, for example, separated from this line by a switchport (see figure).

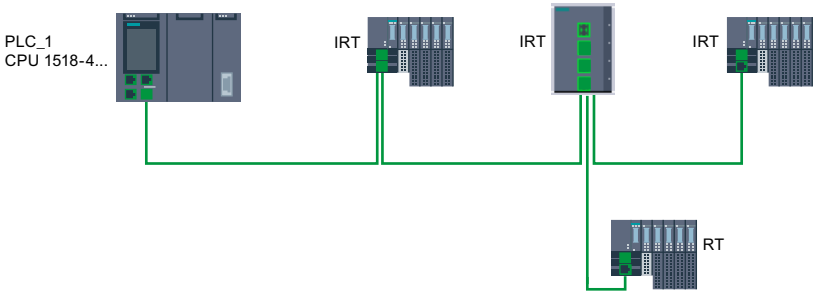

Performance upgrade

Introduction

Performance upgrade

The performance upgrade implements the application class "High Performance" of the PROFINET specification V2.3.

The performance upgrade provides a series of measures that lead to the following improvements for PROFINET with IRT:

- Reduction of runtime delays in the IO devices
- Increase in the bandwidth for cyclic IO data
- Reduction of the bandwidth used for PROFINET frames
- Reduction of the send clocks

The improvements achieved with the performance upgrade mean that you can operate your PROFINET IO system with more devices with the same send clock or the same number of devices with a shorter send clock.

With PROFINET, it was previously possible to reach a send clock of 250 µs.

With the performance upgrade, it is now possible to reach isochronous send clocks of up to 125 µs with the fast forwarding, dynamic frame packing and fragmentation procedures. With short send clocks, standard communication remains possible.

The performance upgrade will help you to implement applications with high speed and send clock requirements.

Dynamic frame packing

Dynamic frame packing

Previously, individual PROFINET IO frames were sent for every PROFINET IO device.

The performance upgrade uses the dynamic frame packing procedure that is based on the summation frame method. With the summation frame method, a frame contains the user data for all nodes on a bus. With dynamic frame packing, every IO device takes its data from the frame and forwards the rest. The frame is shortened from IO device to IO device. Dynamic frame packing improves the use of the bandwidth in a line topology.

The following figure shows how dynamic frame packing works based on the example of a frame containing the user data for 3 IO devices.

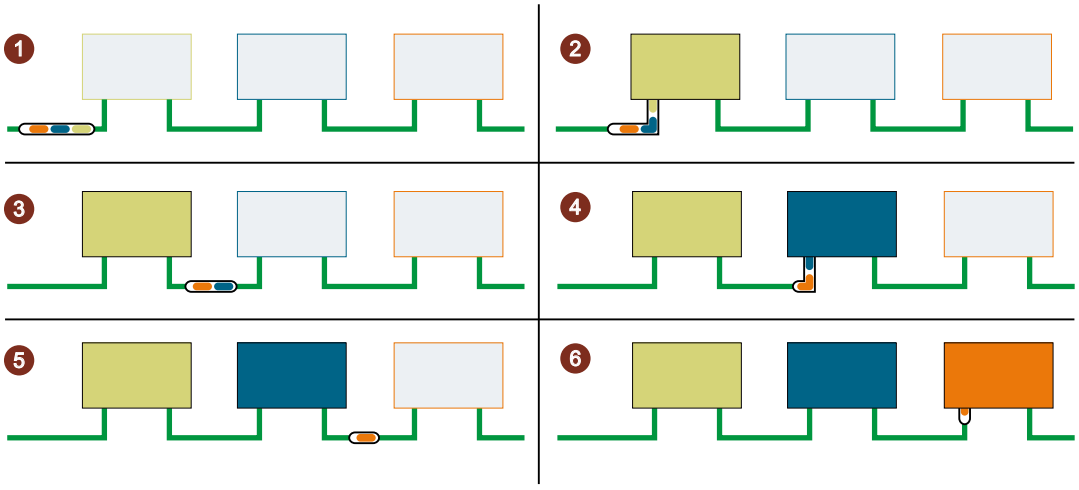

① The PROFINET IO frame contains the user data for all 3 IO devices (green, blue and orange).

② The PROFINET IO frame reaches the first IO device. The IO device takes its user data (green) from the frame and forwards the remaining frame.

③ The PROFINET IO frame contains the user data for two IO devices (blue and orange).

④ The PROFINET IO frame reaches the second IO device. The IO device takes its user data (blue) from the frame and forwards the remaining frame.

⑤ The PROFINET IO frame contains the user data for one IO device (orange).

⑥ The PROFINET IO frame reaches the last IO device. The IO device saves the entire frame including user data (orange).

DFP groups

Dynamic frame packing automatically groups IO devices that support the performance upgrade into DFP groups. To be grouped together in a DFP group. the IO devices must be located one after the other in a line and must have the same update time and watchdog time. As soon as a maximum frame size for the DFP group is exceeded or a maximum number of members for a DFP group is reached, dynamic frame packing automatically opens a new DFP group.

STEP 7 shows the DFP groups in "Domain management" > "Sync domains" > "Name of the sync domain" > "Device" in the "IO devices" box.

IO devices

PROFINET device name	RT class	Synchronization role	Redundancy level	DFP group
plc_3.profinet interface_1	RT,IRT	Sync master		
io-device_2	IRT	Sync slave	No redundancy	2
io-device_3	IRT	Sync slave	No redundancy	2
io-device_4	IRT	Sync slave	No redundancy	3
io-device_5	IRT	Sync slave	No redundancy	3

Note

Dynamic frame packing with high watchdog times

If you set the parameter "Accepted update cycles without IO data" higher than 31, dynamic frame packing does not place this IO device in a DFP group.

Fragmentation

Fragmentation

The transfer of a complete standard Ethernet frame with TCP/IP data takes up to 125 µs. This means that the cycle time for PROFINET IO data cannot be reduced by any desired amount.

The performance upgrade uses the fragmentation procedure, which breaks down TCP/IP frames into sub-frames. These frame segments are transferred to the target device over multiple send clocks; there they are reassembled to the original TCP/IP frame.

Fragmentation will reduce the cycle time to values of less than 250 µs. The bandwidth required for TCP/IP data is less in the shorter send clocks. The bandwidth use for cyclic IO data increases, which means you can transfer more cyclic IO data in the same time.

The following figure shows how fragmentation works.

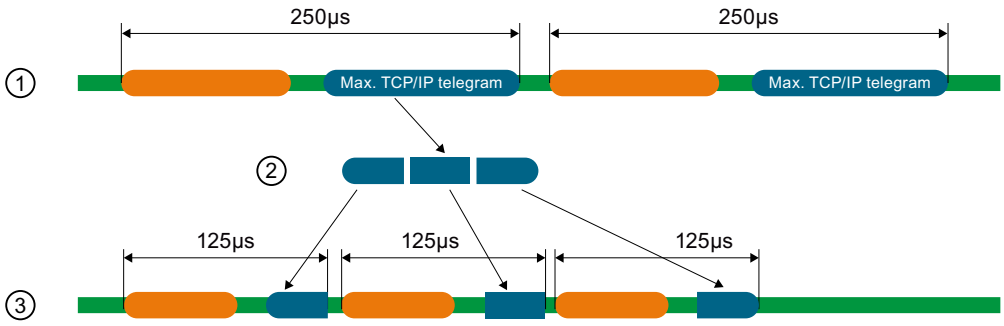

① A standard Ethernet frame with TCP/IP data is at least 125 µs.

② In fragmentation, the standard Ethernet frame is divided into frame segments.

③ The frame segments are divided into multiple short send clocks.

Fast forwarding

Fast forwarding

To be able to decide whether a frame should be forwarded or used, a PROFINET IO device requires the frame ID. It previously took 1440 ns until the frame ID was present in the IO device.

The performance upgrade uses the fast forwarding procedure in which the frame ID is located nearer the front of the frame. The throughput time in the device is reduced to 320 ns.

With fast forwarding, the throughput time of the frame is reduced in your PROFINET IO system. This results in decisive performance advantages, particularly in line topologies, ring topologies and tree topologies.

The figure below compares the throughput of a PROFINET IO frame in an IO device with and without fast forwarding.

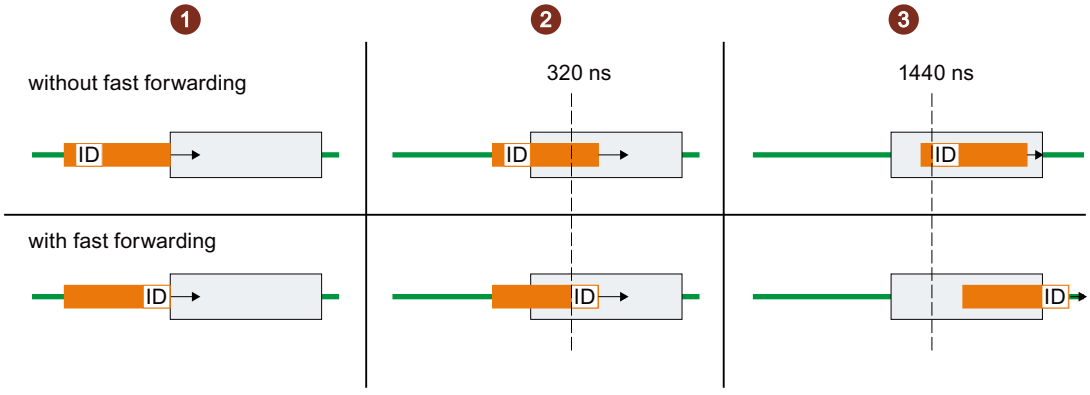

① Start:

- Both PROFINET frames reach the IO device.
  The IO devices start to check the frames for the frame ID

② After 320 ns:

- Without fast forwarding: The IO device is still checking the PROFINET frame for the frame ID.
- With fast forwarding: The IO device receives the frame ID from the PROFINET frame and forwards the frame.

③ After 1440 ns:

- Without fast forwarding: The IO device receives the frame ID from the PROFINET frame and forwards the frame.
- The PROFINET frame with fast forwarding is "ahead" of the PROFINET frame without fast forwarding.

Configuration of IRT with high performance

High-end applications with IO communication require excellent performance in IO processing, for example in the control of wind turbines (converter control).

To use IRT communication with the highest performance in your PROFINET IO system, enable the option "Make 'high performance' possible".

When you enable the "Make 'high performance' possible" option, this has the following effects:

- You can set send clocks of 188 µs and 125 µs (with the CPU 1518).
- You can set more bandwidth use for cyclic IO data.
- You can use the option "Allows the use of 'fast forwarding'".

Requirements

- S7-1500 CPU as of firmware version V2.0

Enable the "Make 'high performance' possible" option.

Follow these steps to activate the "Make 'high performance' possible" option:

1. Select the PROFINET IO system in the network view of STEP 7.
2. In the Inspector window, go to "Properties" > "General" > "PROFINET" > "Domain management" > "Sync domains" > "Name of the sync domain".
3. Enable the "Make 'high performance' possible" option.

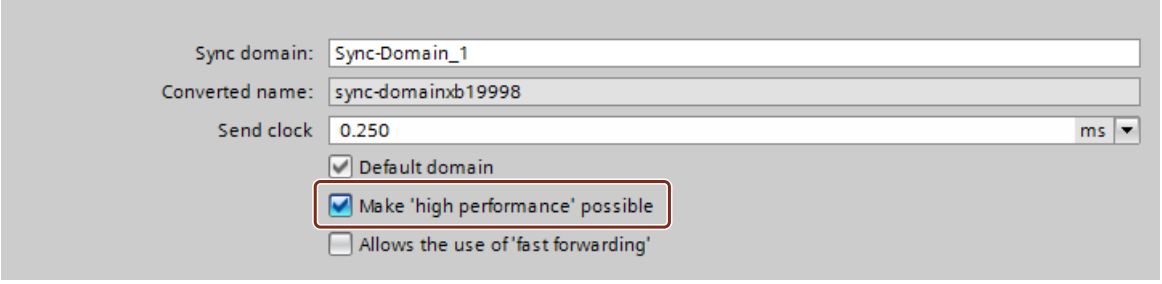

Using more bandwidth for cyclic IO data

Requirement: The "Make 'high performance' possible" option is enabled.

To set more bandwidth for cyclic IO data for your PROFINET IO system, follow these steps:

1. Select your IO system in the network view of STEP 7.
2. In the Inspector window, go to "Properties" > "General" > "PROFINET" > "Domain management" > "Sync domains" > "Name of the sync domain" > "Details".
3. In the drop-down list, select "Maximum 90% cyclic IO data. Focus on cyclic IO data".

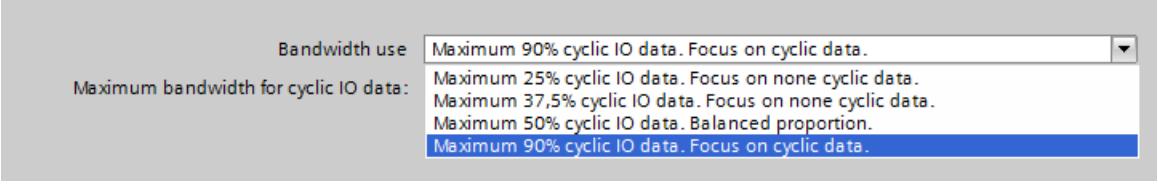

Note

Bandwidth usage in isochronous mode

If you operate your PROFINET IO system in isochronous mode, avoid using the setting for the bandwidth usage "Maximum 90% cyclic IO data. Focus on cyclic IO data."

Setting low send clocks

Requirement: The "Make 'high performance' possible" option is enabled.

1. Select the PROFINET IO system in the network view of STEP 7.
2. In the Inspector window, go to "Properties" > "General" > "PROFINET" > "Domain management" > "Sync domains" > "Name of the sync domain".
3. Select the send clock for "Send clock" in the drop-down list.

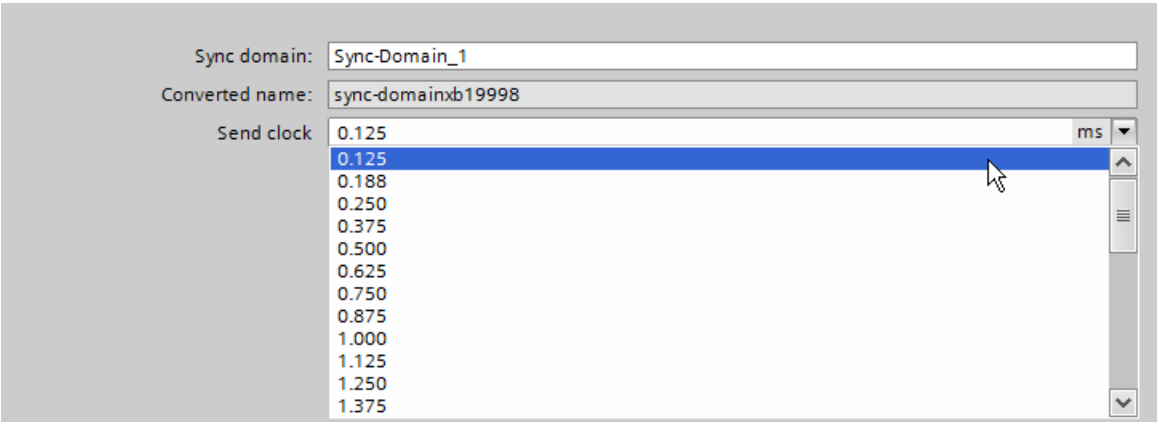

Requirements for the fragmentation process (CPU 1518-4 PN/DP)

If you use the following combinations for send clock and bandwidth settings, the devices in the IO system use the fragmentation process:

- Send clock cycle 125 µs: Always fragmentation irrespective of the bandwidth setting.
- Send clock cycle 188 µs: Fragmentation with the bandwidth settings "Maximum 50% cyclic IO data. Balanced proportion." and "Maximum 90% cyclic IO data. Focus on cyclic IO data."

The PROFINET IO interface of the controller supports fragmentation if all the ports except one are deactivated.

Allow fast forwarding

Requirements:

- The PROFINET IO device has to support the fast forwarding procedure.
- The "Make 'high performance' possible" option is enabled.
- The PROFINET IO interface supports fast forwarding if all the ports except one are deactivated.

Follow these steps to allow fast forwarding:

1. Select the PROFINET IO system in the network view of STEP 7.
2. In the Inspector window, go to "Properties" > "General" > "PROFINET" > "Domain management" > "Sync domains" > "Name of the sync domain".
3. Enable the "Allows the use of ‘fast forwarding'" option.

Note

Fast forwarding and IPv6

The operation of fast forwarding in combination with IPv6 is not supported.

As soon as an IO device in the subnet uses an IPv6 address, you must not activate "fast forwarding".

Optimizing port settings for low send clocks

You can further optimize the bandwidth use in your PROFINET IO system by using cables with a length of < 20 m between the devices or by setting a low signal delay (max. 0.12 µs).

To configure lines with a length < 20 m in STEP 7 or to configure the corresponding signal delay, follow these steps:

1. Select the port in the topology view of STEP 7.
2. Navigate in the Inspector window to "Port interconnection" in the "Partner port" box.
3. Enable the "Cable length" option (default setting) or "Signal delay".
4. Select "< 20 m" in the drop-down list or enter the signal delay.

See also

Rules and information for IRT configurations (Page 2905)

Bandwidth use for synchronized IO controllers (Page 2903)

Sample configuration for IRT with high performance

The figure below shows a sample configuration with which you can achieve maximum performance.

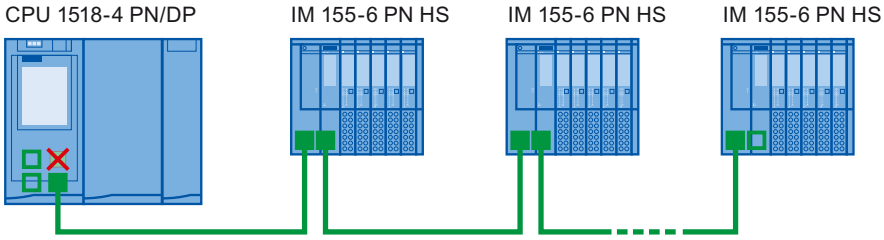

✕ Port 2 of the X1 interface and the ports of the interface modules in the line use the following setting: Medium copper, cable length < 20 m.

■ Port 2 of the X1 interface and the ports of the interface modules on the bus use the following setting: Medium copper, cable length < 20 m.

□ Via the interfaces X2 and X3 of the CPU, a PG connection can be established to the CPU.

Use the following settings for the Sync domain:

- Enable the "Make 'high performance' possible" option.
- Set the send clock to 125 µs.
- Set the bandwidth use to "Maximum 90% cyclic IO data. Focus on cyclic IO data".
- Enable the "Allows the use of ‘fast forwarding'" option.

Standard Ethernet communication for IRT with high performance

Standard Ethernet communication is still possible even in a PROFINET IO system with high performance. Keep in mind that you first arrange the IRT nodes, as seen from the IO controller, and arrange the standard Ethernet nodes at the end of the line.

With a large volume of data through standard Ethernet communication, it makes sense to reduce the load on your network by separating standard Ethernet communication and cyclic real-time communication. Example: Use the X1 interface for the fast PROFINET IO system and the X2 interface for standard Ethernet communication.

Maintenance with asset management data records

Further information about asset management at PROFINET

Operation of machines and plants without a detailed knowledge of devices and assets is difficult to imagine.

Maintenance requires data for this – data that is extensive and up-to-date as far as possible.

The requirement of greater transparency with regard to the data made available by plant components has been fulfilled by PROFIBUS & PROFINET International (PI): The identification and maintenance data familiar since PROFIBUS times have been extended.

The current PROFINET fulfills this requirement through the definition of a special data record: the Asset Management Record (AMR).

The aim of this definition is to enable you to acquire **all** the components to be maintained online - and not only those components that can be addressed and accessed through the PROFINET device model (Device / Module / Submodule). PROFINET now also reaches non-PROFINET components!

Assets general and asset management data records

Assets are components (hardware and software / firmware) of a machine, for example a laser welding machine, or a plant.

A large number of these device components can already be identified through tried-and-tested I&M functions or corresponding data records (I&M0 to I&M5) in the PROFINET context: The device itself as well as its modules and submodules. Meaning all components that can be addressed through the PROFINET device model.

Components that **cannot** be addressed via the PROFINET device model, but whose data it should be possible to acquire online for operation and maintenance, can be identified through **asset management** functions. This asset management data (short: AM data) is stored in a defined structure in a special data record, the **asset management record** (AMR) mentioned above.

The PROFINET Guideline "Identification & Maintenance Functions" differentiates here between I&M functions (I&M data) and asset management functions (AM data): The following sections only deal with the AM data.

The components that can be read additionally online through asset management data records include both hardware components, such as backplane bus modules of a device as well as firmware components such as a drive control unit with own versioning.

Application examples

Importing asset management records enables you to read the following information during installation or operation, for example:

- Are only approved devices being used (whitelist check)?
- A firmware update is due. Obtain a fast overview: Which devices or components are affected and have to be upgraded?

Making asset management data available

The concept for the asset management of PROFINET devices stipulates that the manufacturers of PROFINET devices have to ensure that non-PROFINET automation components are made available through an asset management record. This data record is assigned to the PROFINET device.

In contrast to a "standard" IO device, with an I-device the project engineer has to make the asset management record available. In this case, the central modules of the I-device are assets. From the perspective of PROFINET, these central modules are not visible from the point of view of the higher-level IO controller. The higher-level IO controller only "sees" the transfer areas through which it exchanges IO data with the I-device.

The principle of this provision is explained in the section.

Reading asset management data

The asset management record has the index 0xF880 and is read with standard PROFINET mechanisms by the user of the records, for example a tool or program for evaluating these data.

A user program in the S7-1500 IO controller, for example, can read out the AMR of an IO device with the RDREC instruction (Index 0xF880).

It is not possible to write to this data record.

Further information

Whether and to which extent a PROFINET device supports asset management data, meaning whether it makes an AMR available, is specified in the documentation of the respective device.

Content and structure of an asset management record

Basic structure of the asset management record

You are first provided with an overview of the general structure of the record. The following table describes the framework within which the asset management data blocks are embedded. Each data block represents an asset, a terminal block for example.

Element of the data structure	Designation according to IEC 61158-6-10	Code	Data type / length in bytes
Header AssetManagement-Data	BlockType	0x0035	UINT / 2
	BlockLength	Number of bytes without counting the bytes for Block-Type and BlockLength	UINT / 2
	BlockVersion	0x0100	UINT / 2
AssetManagementInfo AssetManagementBlocks (n)	NumbersOfEntries	Number of AssetManage-mentBlocks	UINT / 2
	AssetManagementBlock 1	See the table below	
	AssetManagementBlock 2		
	...		
	AssetManagementBlock n		

Structure of asset management blocks

Each AssetManagementBlock contains identification data and localization information for an asset. An AssetManagementBlock has a substructure with basic characteristics described below.

The header of an AssetManagementBlocks contains the coded information about which of the three possible AM data compilations the record contains. Devices make a suitable BlockType available in accordance with the various device types:

- Complex devices with information about the hardware and firmware (BlockType "AM_Fullinformation")
- Complex devices with information about the hardware (BlockType "AM_HardwareOnlyInformation")
- Devices with information about the firmware (BlockType "AM_FirmwareOnlyInformation")

The differentiation provides an efficient data structure below the header. Nevertheless, the data record can have a considerable size (maximum of 64 KB, depending on the number of assets that the IO device supplies).

Table 1-153 Structure of AssetManagementBlock

Element of the data structure	Designation according to IEC 61158-6-10	Code	Data type / length in bytes
Header AssetManagementBlock	BlockType	0x0036 (AM_FullInformation) 0x0037 (AM_FirmwareOnlyInformation) 0x0038 (AM_HardwareOnlyInformation)	UINT / 2
	BlockLength	Number of bytes without counting the bytes for BlockType and BlockLength	UINT / 2
	BlockVersion	0x0100	UINT / 2
	Padding	0x0000 (padding byte)	USINT / 1
	Padding	0x0000 (padding byte)	USINT / 1

Element of the data structure	Designation according to IEC 61158-6-10	Code	Data type / length in bytes
AssetManagementBlock (Structure depends on the BlockType. Here it is shown using AM_Fullinformation as an example)	IM_UniqueIdentifier	Manufacturer-generated Universal Unique Identifier (UUID) conforming to ISO/IEC 9834-8. Used as a reference key to uniquely identify this asset. Example: 550c5300-d34a-22b4-11d3-5533991111	Array of Byte / 16
	AM_Location	Description of the location of the asset: Either slot-oriented ("Slot and Subslot-Number format") or hierarchical ("Twelve level tree format"). See following description	Array of Byte / 16
	IM_Annotation	Manufacturer-specific notation Example: "Terminal block, Type xyz123 ". 64 bytes are always used. Spaces are used for padding if the string is shorter.	Array of Char / 64
	IM_OrderID	Manufacturer-specific article number Example: "6ES7 131-6BF00-0BA0 ". 64 bytes are always used. Spaces are used for padding if the string is shorter.	Array of Char / 64
	AM_SoftwareRevision (not at AM_HardwareOnlyInformation)	Manufacturer-specific SW version Example: "V6.3.8 ". 64 bytes are always used. Spaces are used for padding if the string is shorter. If the asset supports IM_Software_Revision, the AM_SoftwareRevision is padded with spaces.	Array of Char / 64
	AM_HardwareRevision (not at AM_FirmwareOnlyInformation)	Manufacturer-specific hardware version Example: "A4 ". 64 bytes are always used. Spaces are used for padding if the string is shorter.	Array of Char / 64
	IM_Serial_Number	Manufacturer-specific unique production-related number. The characters come from the visible range (0x20 ... 0x7E), no control characters. Example: "A78C-1C82 ". 16 bytes are always used. Spaces are used for padding if the string is shorter.	Array of Char / 16
	IM_Software_Revision (not at AM_HardwareOnlyInformation)	Software version, follows a strict structure (SW version prefix, for example "V", digits for functional extension, digits for BugFix, digits for internal change). Example: 'V' 0x01 0x2 0x3	Array of Byte / 4 Prefix (character "V", "R", "P", "U", or "T"), then 3 digits "0" to "9"

Element of the data structure	Designation according to IEC 61158-6-10	Code	Data type / length in bytes
		If AM_SoftwareRevision is padded with spaces, you should evaluate IM_Software_Revision. If the asset does not support any hardware, the coding 'V' 0x00 0x00 0x00.	
	AM_DeviceIdentification	Identification of the device. The structure is as follows: AM_DeviceIdentification.DeviceSubID (for SIEMENS e.g. 0x0000) AM_DeviceIdentification.DeviceID (Device ID from manufacturer, 0x0000 to 0xFFFF) AM_DeviceIdentification.VendorID (Example for Siemens assets: 0x002A) AM_DeviceIdentification.Organization: Example for Siemens assets: 0x0000 (PROFINET)	Array of Byte / 8
	AM_TypeIdentification	Manufacturer-allocated type identification: 0x0000: Unspecified 0x0001: Controller (PLC) 0x0002: PC-based 0x0003: IO module, IO submodule 0x0004: Communications module / submodule 0x0005: Interface module / submodule 0x0006: Active network component 0c0007: Media attached unit (bus adapter) 0x0100 to 0x7FF: Manufacturer-specific	UINT / 2
	IM_Hardware_Revision (not at AM_FirmwareOnlyInformation)	Version of the hardware (0x0000 to 0xFFFF) Example: 0x0003 If AM_HardwareRevision is padded with spaces, you should evaluate IM_Hardware_Revision.	UINT / 2

AM_Location

Asset management at PROFINET supports two formats for coding the location of an asset:

- Slot-oriented format ("Slot and SubslotNumber format")
- Hierarchical format ("Twelve **L**evel **T**ree format" abbreviated **LT** format)

Assets that are part of the PROFINET device use the slot-oriented format. These assets are bound completely to the PROFINET modules and submodules.

Assets that are located outside the PROFINET device use the hierarchical format (LT format) for coding the location of an asset.

These assets are localized by their tree level. The tree level begins with Level 0. The value of Level 0 provides information about the proximity to the PROFINET device:

- If the asset is connected to a module that can be addressed through the PROFINET device model, Level 0 has the value 0. The subsequent levels (Level 1 to Level 3) then have the meaning of slot address, subslot address and channel number. If further assets are connected to this asset, the next Level 4 is used. The limit is reached at Level 11.
- If the asset belongs to a PROFINET device but is not connected to a module that can be addressed through the PROFINET device model, Level 0 has a value between 1 and 0x1FF. An example of such an asset is a power supply unit in the PROFINET device. If a further asset is connected to this power supply unit, for example a sensor, the next tree level is used to localize this sensor (Level 1).
- If the asset is located outside the PROFINET device, but, for example, belongs to a machine into which the PROFINET device is installed, Level 0 has a value between 0x200 and 0x3FE.

The value 0x3FF for a tree level shows that this tree level is not used. This means that no further asset is connected. In this case, all the lower tree levels down to Level 11 must also have this value.

Example AM_Location slot-oriented

A rack and the terminal blocks located on it each supply AM data. The slot assignments are shown in the figure.

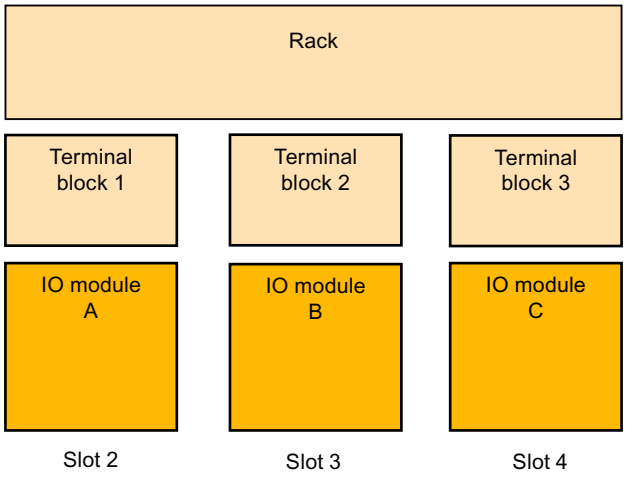

Figure 1-94 Example of assets with slot-oriented AM_Location coding.

Code the AM_Location as an asset for each module as follows:

Bit 0 – 7: AM_Location.Structure = 0x02 (coding "Slot and SubslotNumber format")

Bit 8 – 15: AM_Location.Reserved1 = 0x00 (padding byte)

Bit 16 – 31: AM_Location.BeginSlotNumber = 2 (the "Rack" asset begins from Slot 2 on)

Bit 32 – 47: AM_Location.BeginSubslotNumber = 0xFFFF (the asset encompasses all the subslots of Slot 2. Otherwise you specify the no. of the subslot at which the asset begins)

Bit 48 – 63: AM_Location.EndSlotNumber = 4 (the asset ends at Slot 4)

Bit 64 – 79: AM_Location.EndSubslotNumber = 0xFFFF (the asset encompasses all the subslots of Slot 4. Otherwise you specify the no. of the subslot at which the asset ends)

Bit 80 – 95: AM_Location.Reserved2 = 0x0000 (padding byte)

Bit 96 – 111: AM_Location.Reserved3 = 0x0000

Bit 112 – 127: AM_Location.Reserved4 = 0x0000

Example AM_Location level-oriented

A complex sensor is connected to an IO module (Slot 5, Subslot 1, Channel 1). Two simple sensors, in turn, are connected to the complex sensor. The module can be addressed within the PROFINET device model. Level 0 therefore has the value 0x0000. The next level (Level 1) is specified by the assigned slot. This is followed by the further levels for the subslot and channel and, if appropriate, further subordinate layers.

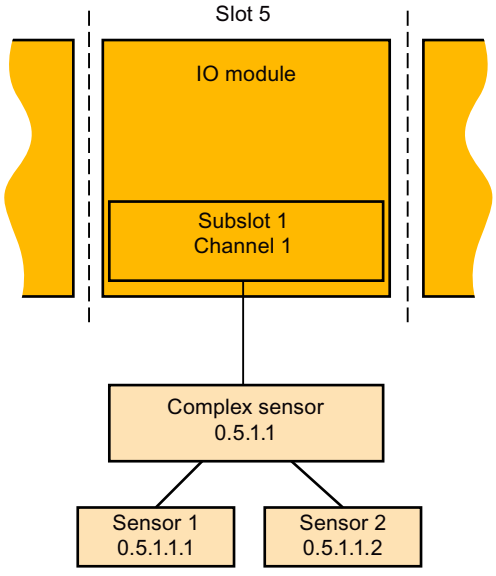

Figure 1-95 Example of assets with hierarchical AM_Location coding.

Detailed coding for the example:

Bit 0 – 7: AM_Location.Structure = 0x01 (LT format)

Bit 8 – 17: AM_Location.Level0 = 0x000 (assets that are assigned to modules always have the Level 0 value 0x000)

Bit 18 – 27: AM_Location.Level1 = 0x005 (Slot 5)

Bit 28 – 37: AM_Location.Level2 = 0x001 (Subslot 1)

Bit 38 – 47: AM_Location.Level3 = 0x001 (Channel 1)

Bit 48 – 57: AM_Location.Level4 = 0x3FF (coding for "Level not used")

Bit 58 – 67: AM_Location.Level5 = 0x3FF (coding for "Level not used")

...

Bit 118 – 127: AM_Location.Level11 = 0x3FF (coding for "Level not used")

Notation used in the screen for the LT coding of complex sensors: 0.5.1.1

The following correspondingly applies for the remaining sensors:

LT coding for simple Sensor 1 at complex sensor: 0.5.1.1.1

LT coding for second simple Sensor 2 at complex sensor: 0.5.1.1.2

Asset management data record for I-devices

With STEP 7 (TIA Portal) as of V15 and with S7-1500 CPUs as of Firmware V2.5, you have the possibility to compile an asset management record via a user program. Configured as an I-device, these CPUs then supply the data from centrally plugged modules to a requesting IO controller as assets.

"S7-1500 CPUs" also refers to the CPU variants S7-1500F, S7-1500T, S7-1500C, S7-1500 SW Controller, S7-1500pro CPUs and ET 200SP CPUs.

Asset management records for I-devices

I-devices often represent machines. The PROFINET IO controller to which the I-device is assigned only sees the PROFINET interface (also configured as an IO device) and the transfer areas of the I-device configured by the machine manufacturer. The local modules of the I-device are not visible or cannot be accessed.

The assigned IO controller can read the central modules as assets of the I-device by means of an asset management record that the user program of the I-device compiles.

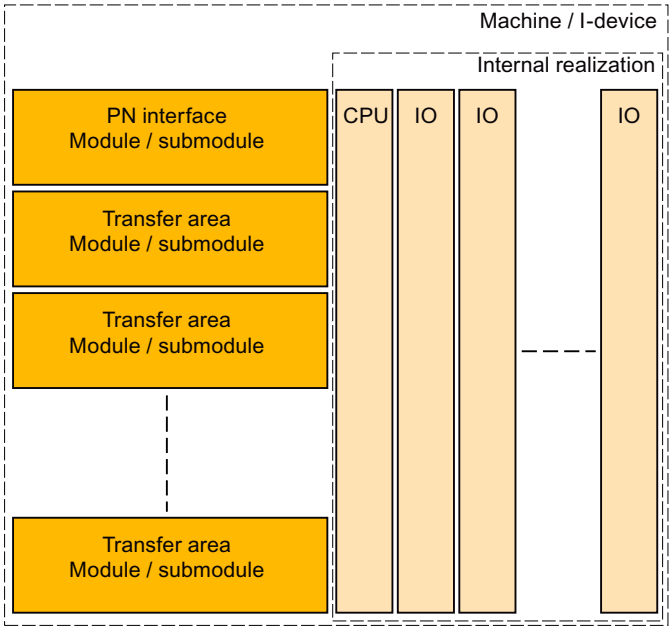

Figure 1-96 Assets of an I-device

Requirement

- S7-1500-CPU as of firmware V2.5, configured as an I-device
- STEP 7 (TIA Portal) as of V15
- If an IO controller is to read the asset management record:
  The PROFINET IO controller is programmed correspondingly to read an asset management record.
  For a SIMATIC IO controller, for example, you call a read instruction (RDREC) with record index 0xF880. The instruction addresses any submodule of the I-device, for example the first configured transfer area submodule.

Basic procedure

The following steps are fundamentally required to create the requirements so that an I-device can make its local modules available as an asset management record to a requesting IO controller:

1. Make the settings in the properties of the PROFINET interface of the CPU.
   - Activate "IO device" operating mode
   - Activate the "Activate asset management using user program" option
     PROFINET interface forwards a request of an IO controller to the user program of the I-device for reading the asset management record only if the option is selected.

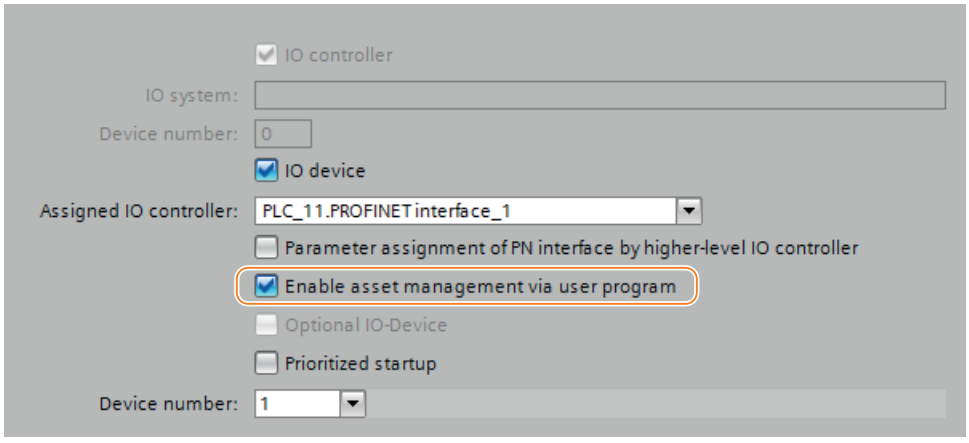

Figure 1-97 Activating asset management using a data record

2. Configure the program routine for compiling the asset management record. The program part collects the required I&M0 data of the plugged central modules and stores them in the corresponding fields of the data record structure of the asset management record.
3. Configure the program part for coordinating the data record provision:
   For this, call the instruction PRVREC (Provide Record) in accordance with the following templates in the corresponding modes:
   - Cyclic calling (for example in the cycle OB) of the PRVREC instruction with Mode 0, in order to recognize the AMR request.
   - When the AM record request is recognized, the PRVREC program has to acknowledge within one second that the request has been recognized. This means that PRVREC must be called with the Mode 2, and with the required AM record. If the I-device does not adhere to the time frame, the I-device acknowledges the record request of the IO controller as negative!
     Particular aspect for configuration of the PRVREC call: PRVREC has to be called with **F_ID = 0**. This codes that this is an IO-device-specific data record. The SLOT and SUBSLOT output parameters therefore also return the value 0.
   - Within 10 seconds the AM record now has to be completed and PRVREC be called with Mode 3 (positive response to the IO controller with provision of the AM record). If the I-device does not adhere to the time frame, the I-device acknowledges the record request of the IO controller as negative!

   A detailed description of the PRVREC instruction and possible error codes for evaluating the function can be found in the online help of STEP 7 (TIA Portal).

Compilation of the asset management record

You have various possibilities for compiling the asset management record for an I-device:

- Recommendation: Recommendation: The Siemens Industry Online Support makes an application available to you that helps you to compile the asset management record.
  The data area of the asset management record is divided in two. The first part consists of an automatically determined area that packages the IM0 data of the slots of the I-device into an asset management block. The second part consists of the user-specific asset management blocks. You configure the user-specific asset management blocks based on pre-configured asset management record structures, fill them with information and make them available to the application.
  The application performs the following:
  - The application determines the required size of all the asset management blocks and generates a new data block in the load memory of the CPU.
  - The application assigns parameters to the data block in accordance with the specifications of an asset management record and fills it with the automatically determined asset management blocks and those transferred by you.
  - The application makes this asset management record available to the higher-level IO controller.

  The application is described in this application example
- You create the asset management record yourself.
  The following section describes how you can compile an asset management record for an I-device yourself.
  The concept assumes you yourself determine the I&M data for each centrally plugged module and fill the asset management record with this information. The I&M0 data of a module contains basic information about the module such as the manufacturer's code, article number, serial number, hardware and firmware version. These are the data that are also required in the AM record for an asset.

Determine the I&M data of centrally plugged modules

The central structure consists of an optionally plugged power supply unit (Slot 0), followed by the I-device CPU (Slot 1), and then followed by the further modules, such as digital modules, analog modules, etc. (as of Slot 2).

You determine the I&M data with the "Get_IM_Data" instruction for the plugged modules with exception of the CPU:

To assign parameters for the "Get_IM_Data" instruction, you require the hardware identifier (LADDR input parameter). You determine the hardware identifier for each occupied slot with the "GEO2LOG" instruction (Determine hardware identifier from slot).

Summary of the theoretical steps:

1. In a loop, determine the hardware identifiers of the plugged modules with the "GEO2LOG" instruction.
2. For each hardware identifier found, determine the I&M data by using the "Get_IM_Data" instruction and store these data in a data block that you address with the input parameter DATA. Use ARRAY of BYTE for the data storage. This corresponds to the description of the AM record contents in the preceding section.

Forming an AM record with the determined I&M data

The following sections are based on the description of the fundamental structure of the AM record, see the preceding section.

Since each module of an S7-1500 contains hardware and firmware information, select the coding for "AM_Fullinformation" for the assigned BlockType.

For the data types used:

- IM_Annotation, IM_OrderID, AM_SoftwareRevision and AM_HardwareRevision: Characters (UTF-8)
- IM_SeriaNumber: Characters ("ASCII characters") with the exception of the characters for DEL (0x7F)
- Do not use String data types. They require additional bytes and therefore do not conform to the conventions of the PROFINET standard 61158-6-10 "Application layer protocol specification"

Form the AM_FullInformationBlock for each module as follows:

Table 1-154 AM_FullInformationBlock for modules

Data record element	Description
IM_UniqueIdentifier	Generate a (pseudo) random UUID (hash value) in accordance with ISO 9834-8 as follows: • Generate an 8-byte hash value across the I&M0 data of the module (as of Slot 2). Use the algorithm Fowler-Noll-Vo (in short: FNV); an algorithm for generating variance coefficients (hash values) across data field, see corresponding example code in the Internet or online support. • Generate an 8-byte hash value across the I&M0 data of the CPU. (Use the algorithm Fowler-Noll-Vo (in short: FNV) as described above) • IM_UniqueIdentifier Byte 0 to 7: Hash value of module I&M0 data Bytes 8 to 15: Hash value for CPU-I&M0 data Required customizations to ISO 9834-8: Byte 8, Bit 7 has to be set to 1, and Byte 8, Bit 6 to 0 (result of the AND operator with **00**11 1111, subsequent OR operator with **10**00 0000) Byte 6, Bit 4 to 7 have to be set to 0100 (result of the AND operation with 0000 1111, then OR operation with 0001 0000) Since this algorithm is based on the I&M0 data of the CPU as well as of the modules, it generates a constant IM_UniqueIdentifier for an individual module. When the configuration changes the IM_UniqueIdentifier also changes.
AM_Location	Byte 0 = 0x02 (slot-oriented coding), see description in the preceding section.

Data record element	Description
IM_Annotation	Example: "S7-1500 module" and pad the remaining bytes of IM_Annotation with spaces (0x20).
IM_OrderID	Copy 20 bytes of the I&M0 data of the module (beginning with offset 2 of the I&M0 data). Pad the remaining 44 bytes with spaces (0x20)
AM_SoftwareRevision	Pad the field with 64 spaces (0x20)
AM_HardwareRevision	Pad the field with 64 spaces (0x20)
IM_Serial_Number	Copy 16 bytes of the I&M0 data of the module (beginning with offset 22 of the I&M0 data)
IM_SoftwareRevision	Copy 4 bytes of the I&M0 data of the module (beginning with offset 40 of the I&M0 data)
AM_DeviceIdentification	Byte 0, 1, 2, 6, 7 = 0x00 Byte 3 = 0x2A (Vendor = Siemens) Byte 4 = 01, Byte 5 = DeviceID (e.g. CPU 15xx = 0x0E)
AM_TypeIdentification	Copy 2 bytes of the I&M0 data of the module (beginning with offset 48 of the I&M0 data)
IM_HardwareRevision	Copy 2 bytes of the I&M0 data of the module (beginning with offset 38 of the I&M0 data)

1.4.12 Bus coupling with PN/PN coupler

1.4.12.1 Application and function

Application

The PN/PN coupler is used to link two Ethernet subnets with one another and to exchange data. That way use data about input or output address areas or datasets can be used. The maximum size of the transferable input and output data is 1024 bytes. The division into input and output data is preferable, so that e.g. 800 byte input data and 200 byte output data can be configured.

As a device, the PN/PN coupler has two PROFINET interfaces, each of which is linked to one subnet.

In the configuration, two IO Devices are produced from this one PN/PN coupler which means that there is one IO Device for each station with its own subnet. The other part of PN/PN coupler in each case is known as the bus node. Once configuring is complete, the two parts are joined.

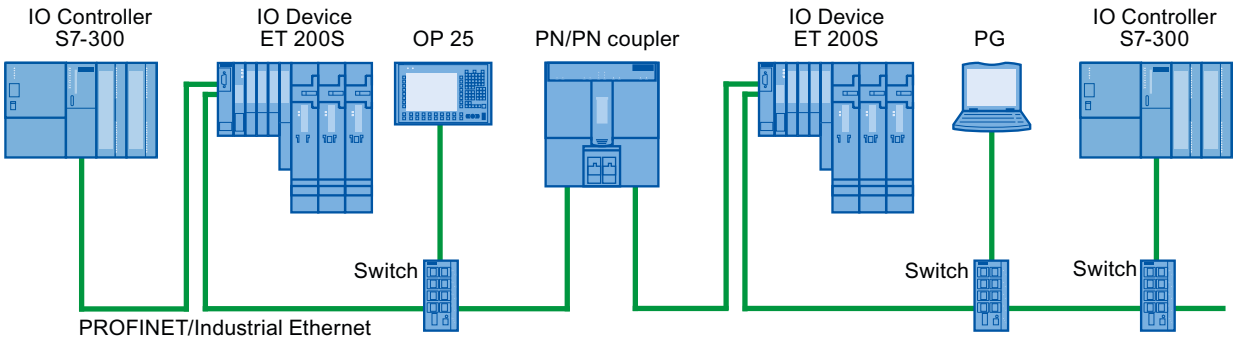

Figure 1-98 Coupling two PROFINET IO subnets with one PN/PN coupler

Additional information

For additional information on "PN/PN couplers", refer to Service & Support on the Internet (http://support.automation.siemens.com/WW/view/en/44319532).

1.4.12.2 Linking Ethernet subnets

Linking Ethernet subnets with a PN/PN coupler

You can link Ethernet subnets with the standard device PN/PN coupler.

To link Ethernet subnets, follow these steps:

1. Create your Ethernet subnets.
2. Select the standard field devices in the hardware catalog. Find the PN/PN coupler as head module in the "PROFINET IO" folder.
3. In the network view, drag the two components X1 and X2 to the required version of the PN/PN coupler per drag-and-drop operation. The components form a device, but are shown separately to make handling easier.
4. Connect the Ethernet interface of the PN/PN coupler X1 to the first Ethernet subnet.
5. Connect the Ethernet interface of the PN/PN coupler X2 to the second Ethernet subnet. The Ethernet subnets are now linked through the two components of the PN/PN coupler.

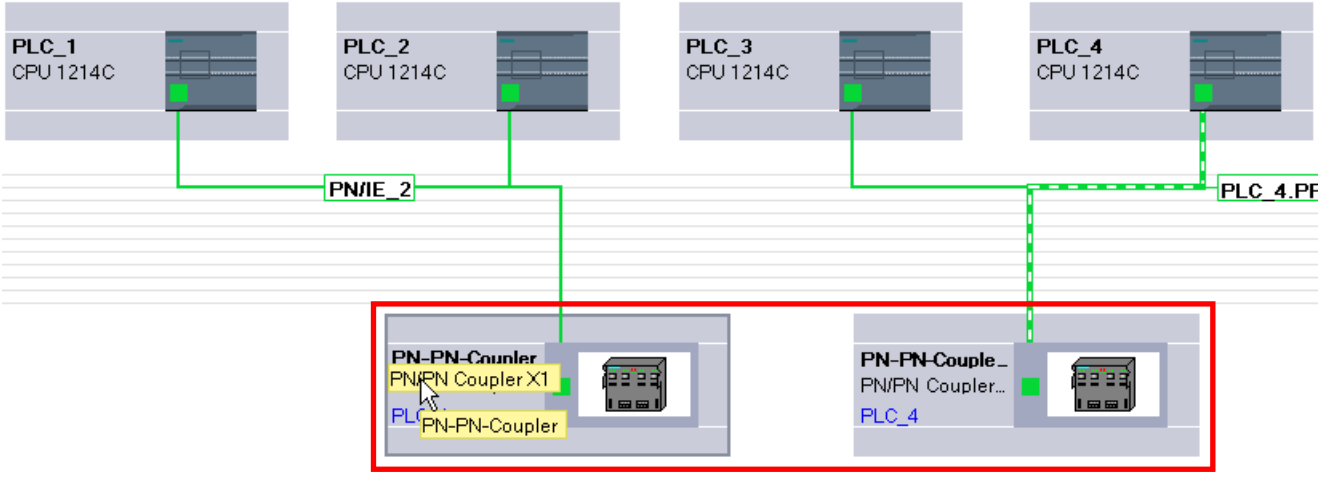

1.4.13 Integrating external tools

1.4.13.1 Integrating S7-external tools

Introduction

Tools external to STEP 7 ("Device Tools") with a special call interface (Tool Calling Interface) can be used to configure distributed devices. Such devices are also referred to as "TCI capable".

The performance range of these tools exceeds the possibilities provided within GSD configuration, for example, they can provide expanded graphical input options.

Distributed devices can be as follows:

- PROFIBUS DP slaves
- Modules within a DP slave
- PROFINET IO devices
- Modules within an IO device

Note

Warranty and liability

Siemens accepts no liability for third-party software (device tools) called with the TCI (Tool Calling Interface) or for proper interaction with the associated devices.

Requirement

The call interface of the tool complies with the TCI specification. Parameters and commands are forwarded to the distributed device via this call interface.

Such tools have to be installed using a setup provided by the manufacturer. The "S7-PCT" (Port Configuration Tool) device tool for IO-Link master modules and IO-Link devices is an exception; this is supplied with STEP 7. Special note: After the installation, the tool is not shown in the list of installed software or in the list of software products in the project.

The GSD file of the distributed device that is to be configured with the Device Tool must be installed.

Starting the device tool

The command for starting the device tool is available in the shortcut menu of the TCI-capable device in the shortcut menu of the graphical and tabular device view: "Start device tool".

See also

Starting the SIMATIC S7-PCT (Page 3029)

Using GSD files (Page 2775)

1.4.13.2 Starting the SIMATIC S7-PCT

Introduction

The "S7-PCT" (Port Configuration Tool) device tool is installed with STEP 7.

The tool is used to assign parameters to the ports of IO-Link modules such as 4SI IO-Link (S7-1200, ET 200S) or 4IOL+8DI+4DO (ET 200eco PN).

Requirement

You have configured the corresponding CPU, the DP slave or the IO device with an IO-Link module.

Procedure

To start via the graphical device view, follow these steps:

1. Select the IO-Link module in the device view.
2. Select "Start device tool" from the shortcut menu.

OR, to start with the tabular device view, follow these steps:

1. Select the IO-Link module in the device view.
2. Arrange the areas in the work area in such a way that the tabular device overview is visible (it is located between the device view and the inspector window).
3. Select the row with the IO-Link module in the device overview.
4. Select "Start device tool" from the shortcut menu.

Result

The tool starts and you can configure the ports.

See also

Integrating S7-external tools (Page 3028)

1.4.14 Loading a configuration

1.4.14.1 Introduction to loading a configuration

In order to commission a device, identical configurations must be stored on the programming device/PC and on the connected devices. To synchronize the configurations on the programming device/PC and the connected devices, you load a configuration. Configuration data can be loaded in two directions:

- From the programming device/PC to a device
- From the device to a PG/PC

See also

Downloading a configuration to a device (Page 3030)

General information on loading to PG/PC (Page 3031)

1.4.14.2 Downloading a configuration to a device

Loading the hardware configuration

After you have inserted a new device in the project and configured it or if you have modified an existing hardware configuration, the next step is to load the current configuration to the device. This ensures that the same configuration is set on the programming device/PC and on the module that is physically present. Use the "Online > Download to device" menu command for this.

In the first download, the complete hardware configuration is downloaded. In subsequent downloads, only changes to the hardware configuration are downloaded.

You have the following options for loading the hardware configuration:

- Loading in the device or network view
- Loading in the project tree
- Loading to an accessible device

WARNING

Perform load operation only in STOP mode

Following loading, the machine or process may behave unexpectedly if the parameter assignment is incorrect. A CPU must be set to STOP mode for the load operation to rule out possible damage to equipment or personal injury.

Special considerations for loading isochronous applications

Isochronous applications consist of a hardware configuration part and a software part.

Example: If you change the number of an IO system, the delay time, or the assignment of a process image partition of the isochronous I/O in the hardware configuration, this affects the parameters of the isochronous mode interrupt OB and thus also the software part.

With isochronous applications, you should always load the complete project (hardware and software). With partial loading (loading hardware and software separately at different times), inconsistencies can arise that, for example, can prevent CPU startup or isochronous operation of the application.

1.4.14.3 Loading a configuration to PG/PC

General information on loading to PG/PC

Introduction

If you bring your PG/PC to a plant and the STEP 7 project used to create the configuration of this plant is not available, load the configuration to a new project on your PG/PC, for example. Use the "Online > Loading the device as new station (hardware and software)" menu command for this.

The list of accessible devices in the project tree is always used when loading a device to your programming device. You can select multiple devices and load them to the project at the same time.

Requirements

- The hardware configuration in the device must be created in TIA Portal, V12 or higher. A hardware configuration present in the device that was created with an older version cannot be loaded and must be upgraded.
- Modules present in the device from GSD (ML), HSPs, or service packs must be installed in TIA Portal on the PG/PC.
- A project must be open. This project can be a new (empty) project or an existing project.
- The opened project is in offline mode.

Scope of load operation

The following list shows an overview of the loadable components of a configuration:

- The device (e.g., a CPU) with all I/O modules and all parameter settings
- PROFIBUS master systems and all PROFIBUS-relevant settings
- PROFINET IO systems and all PROFINET-relevant settings
- I devices and I slaves
- Settings for direct data exchange

After loading a CPU, all other modules within the address area of the CPU are also loaded.

The following connections are also loaded when the configuration is loaded:

- S7 connections (including routed connections) in combined PB/IE networks, including via IE-CP or PB-CM interfaces. S7 connections are automatically accepted as configured at one end when a device configuration is loaded, even if the S7 connection was configured at both ends in the original project. When both connection partners are loaded, the connection is joined together again during the next compilation.
- TCP connections via CPU-internal Ethernet interface, UDP/ISOonTCP connections, and TCP, UDP, ISO, and ISOonTCP connections via IE-CP interface
- Connections via the OUC connection parameter assignment for projects from STEP 7 V13 and higher

Note

The hardware configuration loaded to PG/PC is not completely identical to the configuration originally loaded to the device. Note the additional information on loading, in particular with regard to partially loaded configuration data in the case of cross-device communication.

Loading PC systems, such as WinAC or PC-based automation, is not possible.

Loading specific device configurations

Information on loading

When device configurations are loaded to the PG/PC, all assigned parameters are transferred from the device to the project. If the CPU is connected to a subnet, all parameters of the device are loaded and the CPU is displayed in the network view as networked.

Note

CPUs reset to their factory settings do not have any hardware configuration. Therefore, nothing is loaded in this case after "Load to PG/PC" is selected in the "Online" menu.

Loading S7-300/400 configurations

To avoid conflicts when loading a device to an existing project, the following rules must be followed:

- Device names for CPUs, PROFIBUS slaves (DP slaves, I-slaves) and PROFINET devices (IO devices, I devices) must be unique
- The combination of network name, subnet ID, and IP/DP address for modules must be unique

If conflicts occur, the load operation is canceled and an alarm provides information about the problems that occurred. You can then adapt the project correspondingly or install missing components and then repeat the load operation.

Alarm configurations are not loaded to PG/PC.

Loading S7-1200/1500 configurations

Take the following into consideration when loading:

- CPUs of the S7-1200 series with firmware version V1.0 are not supported when loading.
- Device-specific diagnostic interrupts of the S7-1200 are not supported when loading. For device-specific diagnostic interrupts of the S7-1200 to be generated on the PG/PC once again, the hardware configuration must be compiled once again.
- Module comments of the S7-1200/1500 are loaded from the device to the PG/PC if the same project language is set that was used during loading to the device. You can deselect the loading of comments, if desired.

Note

All types of PC systems, such as WinAC, Embedded Controller, CP 1616 or PC CPs, do not support loading to PG/PC.

Loading distributed I/O

The following functionalities and settings of the distributed I/O are loaded:

- DP master systems/IO systems with associated DP masters/IO controllers (CPUs and CPs), DP slaves/IO devices, the utilized modules and their parameter assignment and properties, such as options handling, status bytes, or SYNC/FREEZE
- Connection of process image partitions (PIP) to organization blocks (OB). Applies to the module and OB properties
- Configured hardware interrupts with the associated properties
- DP master systems with I-slave
- CPs as PROFIBUS I-slave or PROFINET I-device
- Direct data exchange

Master-slave relationships between the I-slave/I-device and assigned DP master/IO controller are only established in the project if both the master as well as the I-slave are loaded to the PG. It does not matter whether you load the DP master/IO controller or the I-slave/I-device first. As soon as both devices are loaded, the master-slave/controller-device relationships will also be re-established.

Loading subnets and devices with MPI, PROFIBUS, Ethernet, and PtP

The following special considerations apply to loading subnets and end points of connections for MPI, PROFIBUS, Ethernet, and PtP with their respective connection properties:

- If a device with PROFIBUS interface is loaded, the bus parameters of the device will initially be different from the settings in the original project. The bus parameters will only match those of the original project after all devices involved are loaded and no additional devices are on the same bus.
- Passive communication devices that are not connected as DP slaves or IO devices to a corresponding master system or IO system do not participate in the data exchange. They are therefore not loaded.
- All devices involved should therefore be loaded for cross-device configurations. A warning is output during compiling of the project for missing network stations. Missing routing information due to communication stations that are not loaded is displayed as warning during compilation. If the configuration is loaded from the PG/PC back to the device, different routing information results.

When you compile the project after loading devices to the PG/PC, STEP 7 checks if all devices to which communication relationships have been configured are available. If devices are missing, you receive an alarm with the number of missing communication stations.

NOTICE
Cross-device communication When you load a configuration with cross-device communication to the PG/PC, you must also load all corresponding network stations to the PG/PC. If required network stations are missing and the configuration is loaded back to the device, there is no guarantee that cross-device communication is working again.

See also

Configuring distributed I/O systems (Page 2735)

Loading configurations with web server

Information on loading

The hardware configuration of a CPU also contains the Web server settings. A number of restrictions apply to loading a Web server configuration to the PG/PC:

- The assignment of the Web server language and project language is not loaded for S7-300/400. The project texts are not loaded and an alarm is output that no project languages are assigned. The languages assigned in STEP 7 are loaded without restriction for S7-1200/1500 CPU.
- User administration data of the S7-1200/1500 can be loaded but not edited. You can use a check box to select whether you want to use the existing data as read-only data or discard these data and enter new data.
- Watch tables of the Web server are not loaded.

The source files of the user-defined web pages (HTML pages, Java scripts, etc.) are not loaded. The program blocks generated during loading can only be edited if you enter the properties and the HTML page yourself.

See also

Information about the web server (Page 1382)

Loading configurations with PROFIBUS

Information on loading

A DP master is loaded to PG/PC. The DP master system and all connected DP slaves are inserted into the project. The respective settings remain unchanged. If a suitable PROFIBUS subnet has already been created, the loaded devices with PROFIBUS interface are connected to the existing subnet.

As a prerequisite for loading DP master systems with standard slaves, the corresponding GSD files must be installed in the TIA Portal and available in the hardware catalog. If a required GSD file is not available in the same version as in the device, differences are identified during the consistency check.

Note

Direct data exchange in a configuration is only loaded if all communication partners involved in the direct data exchange are loaded to the PG/PC.

Isochronous mode

Note the following when loading DP master systems with activated PROFIBUS functionality "Isochronous mode":

- Bus parameters and the settings for isochronous mode are only identical after the device is loaded to the PG/PC if all devices relevant for calculation of isochronous mode are loaded.
- Only mono-master systems with isochronous mode are supported. Therefore, only configurations with just one DP master on the PROFIBUS subnet are loaded.

I-slave

Master-slave relations between DP master and I slave are only established in the project if both the DP master and the I slave are downloaded to the PG. It does not matter whether you load the DP master or the I-slave first. If you load the DP master from a DP master system with connected I-slave, the DP master and its DP slaves are loaded. An I-slave proxy is loaded as dummy module for an involved I-slave.

For I-slave proxy devices:

- DP master can be complied and loaded
- Properties are displayed but cannot be changed
- Diagnostics in the project tree is not carried out

In the network view, proxy devices are represented with a question mark:

It is possible to load a DP master with connected I-slave proxy from the PG/PC to the device.

Replacing the I-slave proxy

In order to edit the I-slave in the project, you must load the I-device from the device to the PG/PC. The I-slave proxy is hereby replaced by the full I-slave with its subordinate DP slaves.

You have the following options for replacing the I-slave proxy with the corresponding I-slave:

- In the menu bar with the command "Online >Upload device as new station (hardware and software)
- With selected I-slave proxy using the shortcut menu command "Upload from device > Hardware and software".
- In the toolbar with the button "Upload from device".

Note

The replacement of I-slave proxy devices is only possible if the required I-device is available in the hardware catalog.

I-slaves protected by a password are not loaded.

See also

Configurations involving PROFIBUS DP (Page 2731)

Using GSD files (Page 2775)

Loading configurations with PROFINET

Information on loading

If you have selected a CPU in the list of accessible devices and perform an load operation to PG/PC, all IO controllers and IO devices associated with this device along with their IO systems will be loaded. Settings for the topology are also transferred. If there is already a suitable Ethernet network in the project, the loaded devices are integrated into the existing network.

Relationships between the IO controllers and IO devices are only established within the project if both the IO controller as well as the I-device are loaded to the PG. It does not matter whether you load the IO controller or the I-devices first.

Supported functions

The following functionalities and settings are loaded:

- PROFINET configurations (RT and IRT) in IO systems with the associated IO controllers (CPUs and CPs), IO devices, and the modules used
- Logical addresses and interface properties
- Port interconnections
- Isochronous mode
- Sync domains/MRP domains
- Redundancy role "Client" or "Manager" for MRP configurations

Note

Empty Sync domains and MRP domains are not included in the loading.

GSD-based IO devices

As a prerequisite for loading GSD-based IO devices, the corresponding GSD files must be installed in the TIA Portal and available in the hardware catalog. If a required GSD file is not available in the same version as in the device, differences are identified during the consistency check.

I-device

If you load the IO controller from an IO system with connected I-device, the IO controller and its IO devices will be loaded. An I-device proxy is loaded as dummy module for an involved I-device. The CPU parameter assignment, including the parameter assignment of the CPU's "own" (subordinate) IO system, is missing in the I-device proxy. Only the interface to the higher-level IO controller is loaded.

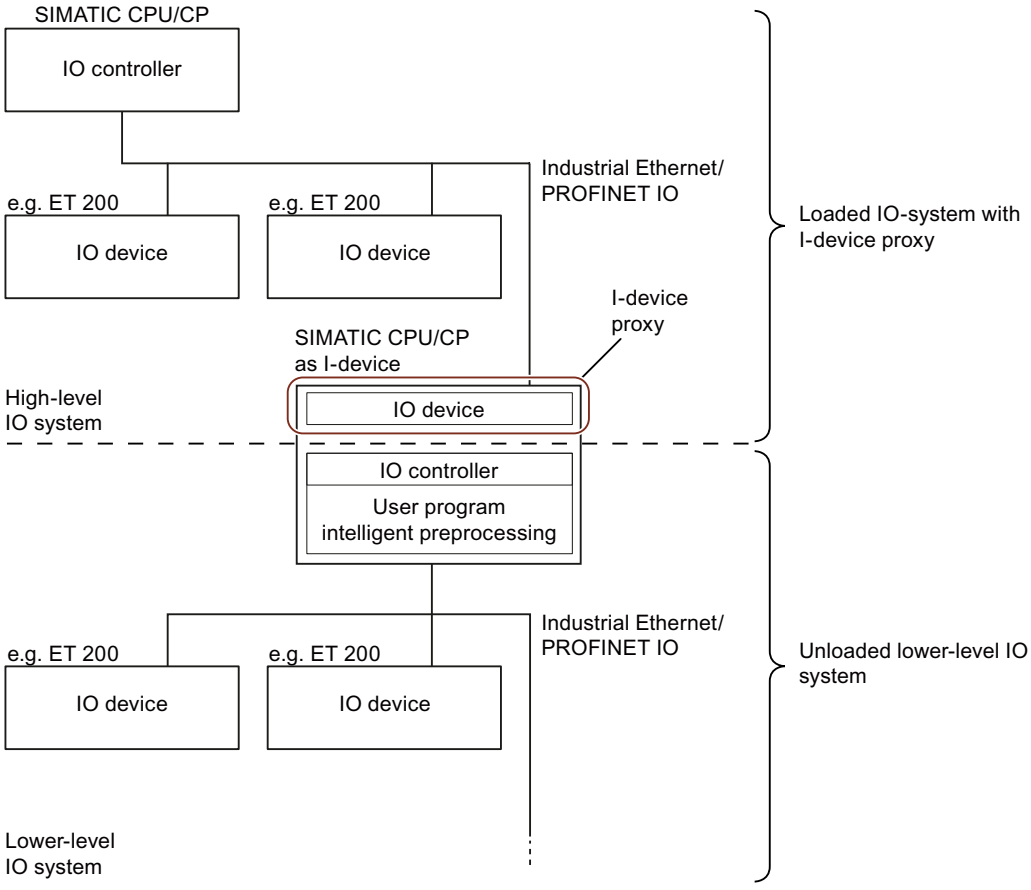

For I-device proxy devices:

- IO controller can be compiled and loaded
- Properties are displayed but cannot be changed
- Diagnostics in the project tree is not carried out

In the network view, proxy devices are represented with a question mark:

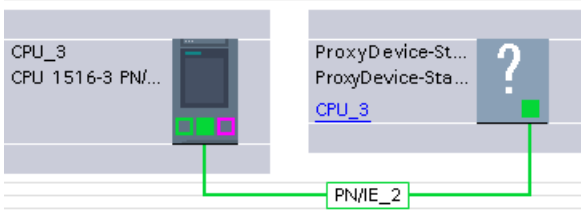

It is possible to load an IO controller with connected I-device proxy from the PG/PC to the device.

Replacing the I-device proxy

In order to edit the I-device in the project, you must load the I-device from the device to the PG/PC. This replaces the I-device proxy with the complete I-device along with its subordinate IO devices.

You have the following options for replacing the I-device proxy with the corresponding I-device:

- In the menu bar with the command "Online >Upload device as new station (hardware and software)
- With selected I-device proxy using the shortcut menu command "Upload from device > Hardware and software".
- In the toolbar with the button "Upload from device".

Note

The replacement of I-device proxy devices is only possible if the required I-device is available in the hardware catalog.

I-devices protected by a password are not loaded.

An I-device proxy representing a SIMOTION I-device cannot be loaded and replaced.

Configurations with IE/PB link

If one of the following configurations with IE/PB Link PN IO as PROFINET IO device is present, the complete configuration with all subordinate PROFIBUS devices is loaded:

- CPU/CP of S7-300/400
- PC station and connected PROFIBUS master system

The complete configuration consists of the following:

- CPU
- CP configuration
- PROFINET IO system with connected IE/PB link
- PROFIBUS master system of the IE/PB link with connected DP slaves

An example configuration consists of an S7-300 CPU with a CP as PROFINET IO controller. An IE/PB link as IO device is connected to the IO controller. As the PROFIBUS DP master, the IE/PB link polls a PROFIBUS DP slave, e.g., ET 200L. If you load the CPU from the device to the PG/PC, the complete configuration is loaded.

Note

If the IE/PB link is not operated as a PROFINET IO proxy but rather as a gateway in standard mode, the IE/PB link functions as a CPU and can be loaded separately.

Loading Shared Devices

The following applies to loading comments: If an input/output module with the Module-internal Shared Input (MSI) or Module-internal Shared Output (MSO) function consists of only one submodule, the submodule does not have its own comment. Instead, it uses the comment of the input/output module. The module must be subdivided into several submodules for the input/output module and all submodules to have their own comment fields.

Loading HMI devices

Information on loading

We distinguish between the following cases when loading HMI devices to PG/PC:

- HMI devices connected to a DP master as DP slave or to an IO controller as IO device are loaded as DP slave or IO device respectively (for example, PP 17-I PROFIsafe).
- HMI devices in a master system as I-slave or in an IO system as I-device are loaded as I-slave proxies or I-device proxies (Page 3036) (for example, SIMATIC Comfort Panels). The settings of the device proxy are read-only.
- HMI devices are not loaded if they are connected to a subnet (PROFIBUS or PROFINET) but not to a master system or IO system (for example, KP600 Basic color DP).

Going online with loaded configurations

You can go online with the loaded project or with loaded portions of the project.

Requirement

The hardware configuration loaded from the device to the PG/PC has been compiled. The statuses of the central and distributed modules are only correctly displayed after compilation.

Note

If you go online before compiling, the diagnostics icon “?” is displayed (diagnostics not possible). A corresponding alarm is displayed under "Info > General" in the Inspector window.

Dependencies

Depending on how completely the hardware configuration was loaded to the PG, restrictions apply to going online and to diagnostics:

- Completely loaded device with all associated central and distributed modules, such as DP slaves or IO devices:
  Going online and diagnostics are possible.
- Loaded device with connected I-devices/I-slaves:
  - I device/I slave is not loaded: Going online for the device and its modules is possible. For the dependent components of the configuration that are not loaded, device proxies are handled with only minimum diagnostics support. The online status is represented as an icon. The standard diagnostics is shown in the online and diagnostics view. I&M data are not loaded.
  - I device/I slave is also loaded: Going online is possible for all devices; diagnostics is fully supported.

1.5 Displaying alarms

1.5.1 Overview of the alarm display

The "Alarm display" function can be used to output asynchronous alarms of diagnostics events and user-defined diagnostics alarms as well as alarms from ALARM instructions.

A note is displayed in case of an overflow of diagnostics events. As soon as the overflow has been eliminated, alarms of the types NOTIFY_AP and ALARM_AP are signaled. Only diagnostics events that are still active after the overflow are signaled. Any diagnostics events that occurred during the overflow and went out again are not signaled (no alarm history).

From the alarm display, you can also start the alarm editor with the "Edit alarm" shortcut menu command and then create user diagnostics alarms.

Icons

The following table shows the icons and their functions:

Icon	Function
Current alarms	Shows the currently active (pending) alarms. Alarms that must be acknowledged are shown in blue lettering.
Alarm archive	Shows the alarms located in the archive.
Clear alarm archive completely	Deletes all alarms in the archive (even with set filter).
Export archive	Exports the current alarm archive to a file in xml format.
Multiple lines	Shows the alarms with multiple lines.
Freeze alarms	Freezes the display of the alarms
Acknowledge alarm(s)	Confirms the selected alarm(s) as read. Alarms requiring acknowledgment are shown in blue lettering.
Filter	Shows the filter for the alarm display. If filter criteria are selected, the icon has a green check mark.

1.5.2 "Current alarms" view

The "Current alarms" view is an image of the alarm acknowledgement memory of the selected CPU(s).

1.5.3 Layout of the alarms in the "Current alarms" view

The "Current alarms" view represents an image of the alarm acknowledgment memory of the selected CPUs. One entry is shown in the table per active alarm. Events of an alarm ("incoming", "outgoing" and "acknowledged") are displayed in one row.

Table structure

All attributes of the alarms can be shown as columns. You can show or hide individual columns as well as modify the width and order of the columns. These settings are saved when the project is closed.

You can sort the columns in ascending or descending order. The sorting of the columns depends on whether the "Freeze alarms" is activated or not.

The alarms can be displayed in one or more rows. In the single row display, only the first row of the multiple-row alarm data is displayed.

The alarms either require acknowledgment or do not require acknowledgment. The alarms requiring acknowledgment that have not yet been acknowledged are highlighted in bold print and can be acknowledged either with the button in the toolbar for the specific context or with the shortcut menu command "Acknowledge alarm(s)".

See also

Sort table in alarm display (Page 3047)

1.5.4 Alarm archive

In the alarm archive, alarms are displayed and archived according to the time they appear. You can set the size of the archive (between 200 and 3000 alarms) with the menu command "Options > Settings > Online & Diagnostics". If the selected archive size is exceeded, the oldest alarm it contains is deleted.

Alarms that must be acknowledged are displayed in blue lettering and can be acknowledged with the shortcut menu command "Acknowledge alarm(s)".

The archive is constantly updated and does not need to be saved explicitly.

See also

Clear alarm archive completely (Page 3044)

Export archive (Page 3044)

1.5.5 Layout of the alarms in the alarm archive

In the alarm archive, all events occurring on the selected CPUs are logged. A new entry is created for each individual event and shown as a further row in the table.

Table structure

All attributes of the alarms can be shown as columns. You can show or hide individual columns as well as modify the width and order of the columns. These settings are saved when the project is closed.

You can sort the columns in ascending or descending order. The sorting of the columns depends on whether the "Freeze alarms" is activated or not.

The alarms can be displayed in one or more rows. In the single row display, only the first row of the multiple-row alarm data is displayed.

The alarms either require acknowledgment or do not require acknowledgment. The alarms requiring acknowledgment that have not yet been acknowledged are highlighted in blue lettering and can be acknowledged either with the button in the toolbar for the particular context or with the shortcut menu command "Acknowledge alarm(s)".

See also

Sort table in alarm display (Page 3047)

1.5.6 Clear alarm archive completely

The archive is organized as a ring buffer, in other words, when it is full, the oldest alarms are deleted from the archive. With the "Clear alarm archive completely" button, you delete the entire archive, regardless of whether or not you have selected filter criteria to display specific alarms.

Procedure

To clear the archive, follow these steps:

1. Click the "Clear alarm archive completely" icon in the toolbar of the alarm display.

1.5.7 Export archive

To archive alarms, you can export the archive. To do this, follow these steps:

1. Change to the alarm archive.
2. Click the "Export archive" icon.
3. In the dialog that opens, select the path to export the archive.

Result

The archive is saved as an xml file at the location you selected.

1.5.8 Receiving alarms

To allow alarms to be displayed, you must first set the receipt of alarms for each CPU.

Procedure

Toe receive alarms, follow these steps:

1. Double-click on the "Online & Diagnostics" folder of the relevant CPU in project navigation.
2. Click the "Online access" group in the area navigation.
3. Select the option "Receive alarms".

Note

If you select this procedure, alarms are only received after you have re-established an online connection to the device.

Or:

1. Select the relevant CPU in the device, network, or topology view.
2. Select the command "Receive alarms" in the "Online" menu or in the shortcut menu.

Or:

1. Select the CPU in project navigation.
2. Select the command "Receive alarms" in the "Online" menu or in the shortcut menu.

Or:

1. In the alarm view, select the desired devices for which you want to receive alarms from the drop-down list "Show alarms".
   - If no project is open, does not contain an online device or a device has not (yet) been selected, the entry "No device selected" is displayed.
   - If you click on the project name, all the subordinate devices are selected and the entry "All devices selected" is displayed
   - If you select individual devices in the project, the entry "Multiple devices selected" is displayed.
2. Confirm your selection by clicking on the green check mark below the drop-down list.

Note

If you select one of the three above-named procedures, you must have first established an online connection to the device.

1.5.9 Freeze alarms

Provides a snapshot of the alarm view in which the screen was frozen. If you click the button again, the updating of the alarms is enabled again.

The button "Freeze alarms" is not activated.

You can only sort the alarms by the columns "Date" and "Time". These settings are saved when the project is closed. Depending on the sorting order, current alarms are displayed at the beginning or the end of the table.

The "Freeze alarms" button is activated.

You can sort every column of the table. If you deactivate the button again, the set sorting is, however, lost again.

1.5.10 Acknowledging alarms

Alarms that must be acknowledged are shown in blue lettering.

Procedure

To acknowledge an alarm, follow these steps:

1. Select the required alarm or alarms from the table.
2. Click on the "Acknowledge alarms" button.

Note

You can select more than one alarm to acknowledge at the same time. To do this, hold down the <Ctrl> key and then select the alarms you want to acknowledge.

Result

The selected alarm was acknowledged and is then shown in normal characters.

Note

In the "Current alarms" view, acknowledged alarms that have already gone are no longer displayed.

1.5.11 Status of the alarms

Depending on whether you are in the "Current alarms" view or the alarm archive, the displayed alarms may have a different status.

Status of the alarms in the "Current alarms" view

- I: Alarm came
- IA: Alarm came and was acknowledged
- IO: Alarm has gone

If more signal changes occur than can be sent (signal overflow), OV is displayed as the status and the status is shown in red.

Status of the alarms in the alarm archive

- No information: only with alarms generated by the PG/PC and displayed in the "Archive" tab, for example logon status, connection abort, mode changes
- I: Alarm came
- A: Alarm came and was acknowledged
- O Alarm has gone
- D: The alarm was deleted.

If more signal changes occur than can be sent (signal overflow), OV is displayed as the status and the status is shown in red.

1.5.12 Sort table in alarm display

Sorting a table in ascending or descending order

To sort the table by a column in ascending or descending order, follow the steps below:

1. Click the table header of a column if you want to sort the column in ascending order.
2. Click again on the same column of the table header to sort the column in descending order.
3. Click a third time on the table header of the same column to cancel the sorting.

The button "Freeze alarms" is not activated.

You can only sort the alarms by the columns "Date" and "Time". These settings are saved when the project is closed.

The "Freeze alarms" button is activated.

You can sort every column of the table. If you deactivate the button again, the set sorting is, however, lost again.

1.5.13 Filter table in alarm display

You can filter all columns except the "Help" column. This is possible in the "Current alarms" and "Alarm archive" views as well as in active mode (alarms are constantly updated) and in the "Freeze alarms" mode.

As soon as a filter criterion is selected, a green check mark appears in the filter icon. If a filter criterion is defined for a column, the corresponding field has an orange background.

Procedure

To filter alarms, follow these steps:

1. Enter a filter criterion in the field.
2. Confirm your entry with the enter key or click any other area to select the filter.
   Up to five filter criteria per column are saved in a drop-down list. You can select the filters from here as needed.
3. If necessary, repeat the procedure for all other required criteria.

Result

Only alarms that correspond to the set criterion (or criteria) are displayed.

If filter criteria are set in the message display when you close the project, these are displayed again when you re-open this (or any other) project. However, these settings are lost when you close the TIA Portal.

1.5.14 Keyboard commands in the alarm display

Alarm display

Function	Shortcut keys
Select all alarms	Ctrl+A
Acknowledge all selected alarms	Ctrl+Q

1.6 Additional information on configurations

1.6.1 Communications modules and network components

1.6.1.1 Industrial Ethernet / PROFINET

Settings

Module access protection / UDP buffering / file system

Reference

Making special settings in the "Properties > General > Settings" parameter group

Depending on the CP type, you will find settings relating to the following areas and these are described below:

- Module protection
- UDP buffering
- File system

Module access protection - "Protection level" option

With this option, you can protect the CP from accidental or unauthorized access. The following options can be selected in the drop-down list:

- Not locked
- Status-dependent
  With this setting, it is only possible to access the CP when the CPU is in STOP mode.

Switch off UDP frame buffering

UDP frames are buffered by the CP depending on the communication load between the CP and the CPU.

You can select between the following using the UDP buffering (switch off UDP frame buffering) option:

- Disabled (default setting)
  Buffering of UDP frames is enabled. UDP frames received by the CP are buffered until they have been transferred to the CPU.
- Enabled
  The CP always transfers the last received, in other words, the current frame to the CPU. As long as no new UDP frame can be transferred between the CP and the CPU due to the current communication load, only the last frame received is buffered in the CP.

Turning off the UDP frame buffering achieves the shortest possible reaction time between the arrival of the UDP frame and its evaluation on the CPU. The characteristics associated with enabling the option can be critical in certain applications with high frame traffic. Buffering of a lot of frames may result in an undesired time offset between the frames accumulated in the CPU and the current frames at the Ethernet interface.

File system - Match case (case sensitive) - option for CPs with IT functions

Enabled: When files are placed in RAM, the CP distinguishes between upper- and lower-case characters in the file names.

Data transfer > 240 bytes

Reference

Making special settings in the "Properties > General > Settings" parameter group

Depending on the CP type, you will find setting "Data transfer > 240 bytes"

Notes on data transfer > 240 bytes

- Data transfer > 240 bytes is supported by the newer versions of the CPs. Refer to the information in the documentation of your Ethernet CP.
- With the S7-300 series, remember that this configuration uses one connection resource (free connection for S7 functions) on the S7-300 CPU. CPU connection resources are also used, for example, by S7-300 CPs in FMS mode or by programming devices or OPs.

Ethernet / PROFINET interface

Advanced options

Keepalive frames for TCP and ISO-on-TCP connections

Reference

Making special settings for the PROFINET interface in the "Properties > General >PROFINET interface > Advanced options > Interface options" parameter group

Keepalive frames (only with TCP and ISO-on-TCP connections)

Here, you can set the interval at which keepalives are sent to the partner of a communications connection.

The Ethernet CP is configured for all connection-oriented services so that keepalive frames are sent. This ensures that connections are terminated if one of the communication partners

fails and the connection resources are released again. The setting made here applies to all TCP and ISOonTCP connections operated via the CP. A connection-oriented setting is not possible.

Range of values

- Default setting: 30 seconds
- Maximum value: see information displayed in the input box
- Function turned off: 0 seconds

Note

Note that the keepalive mechanism can cause subordinate connections (for example, an ISDN telephone connection) to be maintained even though user data is not actually being transferred. If this is unwanted, proceed according to one of the following alternatives:

- Disable the keepalive mechanism (interval time = 0).
- Set the interval so high that the underlying connection is terminated before a keepalive frame is sent when there is no data traffic.

IP configuration

Reference

Setting parameters for the IP protocol in the "Properties > General > PROFINET interface > Ethernet addresses" parameter group

Meaning

You can decide the route and the method with which the IP address of the local interface is obtained and assigned.

With the options available here, it is possible to assign IP addresses "dynamically" outside the configuration.

The selection you make also decides whether communication connections are set up by the project engineering or via the interface in the user program (IP_CONFIG instruction).

The following options are available:

- Set IP address in the project
  This is the default setting for PLCs. You specify the IP address when the device is networked. The IP address CP is therefore fixed.
  With this option, you must configure communications connections.
- Obtain an IP address from a DHCP server
  If you select this option, the IP address is obtained from a DHCP server when the device starts up.
  The DHCP server is informed of the MAC address of the interface or the client ID that can be entered in the configuration.
  The client ID is a string with a maximum of 63 characters.
  Requirements / restrictions: If you select this option, it is initially not possible to create a fully specified connection in the project because the local IP address is not known. You therefore select "unspecified" with passive connection establishment as the connection type.
- Set IP address in the user program
  With this option, you specify that the IP address is set over the user program interface (function block IP_CONFIG). This allows the IP address to be supplied dynamically during operation.
  In this situation, communications connections are set up solely via the interface of the user program, connection configuration is not possible (relates to connections via: TCP, ISO-on-TCP, UDP, ISO transport).
- Set IP address on the device
  With this option, you specify that the IP address is set by other services outside the configuration.
  In this case, connection configuration is no longer possible (relates to connections via: TCP, ISO-on-TCP, UDP, ISO transport).

Note

If you have already configured communications connections via the interface configured here and you select a setting other than "Set IP address in the project", a message will be displayed indicating that the configured connections no longer function.

Time-of-day synchronization

Time-of-day synchronization - mode

Reference

Configuration of the time-of-day synchronization method of the module in the parameter group "Time-of-day synchronization" under the PROFINET interface or in the Security settings.

For time-of-day synchronization of the S7-1200, see section Time-of-day synchronization with S7-1200 (Page 3056).

For configuration of the time-of-day synchronization of the IE/PB Link PN IO (V3) see section Time-of-day synchronization - IE/PB LINK PN IO (V3) (Page 3057).

How the time synchronization modes work

- **SIMATIC mode**
  If the SIMATIC mode is enabled, the module sets its time-of-day according to MMS time-of-day messages (MMS - Manufacturing Message Specifaction).
  The MMS time-of-day messages originate from a SIMATIC time transmitter or a CPU configured as time master. The time-of-day messages can be received by the module (e.g. CP) either from the local CPU or via a LAN. It is possible to set whether the module will forward the received time of day.
  The advantage of the SIMATIC mode is that it is generally more accurate than the NTP mode.
  See also SIMATIC mode - configuration (Page 3053)
- **NTP method**
  (NTP - Network Time Protocol)
  With the NTP method, the module sends time-of-day queries to one or more NTP servers in the subnet (LAN). Based on the replies from the server, the most reliable and most accurate time is determined and the time of day on the querying module is synchronized. In this case, any MMS time messages that are received will be ignored.
  In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred. This corresponds to GMT (Greenwich Mean Time).
  The advantage of NTP mode is that it allows the time to be synchronized across subnets.
  See also NTP method - configuration (Page 3054)

SIMATIC mode - configuration

Time-of-day synchronization in SIMATIC mode:

In "SIMATIC" mode, the CP forwards MMS time-of-day messages when it receives them from a SIMATIC time transmitter. The time-of-day messages can either be received via the backplane bus or via the communications bus (S7-400 CPU) or via LAN.

"Apply time in CP"

The time of day can either be simply received by the CP or received and forwarded.

When the time-of-day is forwarded, you need to specify the direction in which it is forwarded.

"Direction"

Depending on where the time master is located, the direction of the time-of-day forwarding must be taken into account.:

If you have several CPs in your station, take into account the flow of time-of-day messages depending on the time-of-day master since you can forward time-of-day messages from one network to another network using the function described here. In a station there must only be one time master.

If there are several CPs that are connected to the same subnet and if forwarding of the time of day is activated for these, the following applies: The CP with the lowest slot number handles the forwarding of the messages. All other CPs configured for forwarding time-of-day messages, automatically stop forwarding.

You can make the following settings for the direction of time-of-day forwarding:

- Automatic
  The CP receives the timeofday message from the LAN or from the station and forwards it to the station or to the LAN.
  If several CPs are being operated in the station, this automatic setting can lead to collisions. To avoid this, you can specify the direction of forwarding with the following options:
- From station
  The CP receives the time-of-day message from the station and forwards it to the LAN.
  In this case, a local CPU must be configured as the time master or another CP 400 forwards time messages to the K bus.
- From LAN
  The CP receives the time-of-day message from LAN and forwards it to the station.
  In this case, the local CPU must be configured as a time slave.
  Exception: CPU 318

"Use corrected time"

With the this option, you can decide whether a correction factor in the time-of-day frame is taken into account.

The correction factor that can be set in a central clock SICLOCK in units of ½ hours specifies a time offset for example to correct a time zone setting or standard/daylight saving time.

If this option is enabled, the CP calculates the correction factor in the time of day to be forwarded this so that systems to which the time is forwarded do not need to take the factor into account.

Example:

The time 9:21 is transferred in the time-of-day frame. In addition, the time-of-day frame contains the correction factor +2 x 0,5 (= 1h). The time forwarded by the CP is therefore 10:21.

NTP method - configuration

Where possible create a concept for the time-of-day synchronization before the configuration. Make sure that you only use one time source for every station. It makes sense to have only one time source per subnet.

Note the mechanisms of time-of-day synchronization of the S7-1200 in the section Time-of-day synchronization with S7-1200 (Page 3056).

NTP - Network Time Protocol

With the NTP method, the CP or the CPU sends time-of-day queries to one or more NTP servers in the subnet (LAN). Based on the replies from the server, the most reliable and most accurate time is determined and the time of day on the querying module is synchronized. Any MMS time-of-day messages received are then ignored.

The advantage of NTP mode is that it allows the time to be synchronized across subnets.

In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred. This corresponds to GMT (Greenwich Mean Time).

The parameters described below are not available for all modules.

NTP server

You can configure the IP addresses of up to four NTP servers.

Note

S7-1500 CPUs as of version V1.8: Changing the NTP servers fro the user program

For S7-1500 CPUs as of firmware version V1.8 you can change the addresses of 4 NTP servers with the instruction T-CONFIG (SFB 105) at runtime. The requirement for this is that you have configured at least one NTP server in the "Time-of-day synchronization" parameter group.

The configured addresses of the NTP servers are overwritten by the IP addresses of the NTP servers in the instruction T_CONFIG. Even if only 1 NTP server was configured, up to 4 NTP servers can be programmed using the SFB. When necessary, the addresses of the NTP servers can be changed several times with T_CONFIG.

The firmware version V1.8 for S7-1500 CPUs is available with STEP 7 V13.0 SP1 Update 3.

"Time zone"

The time offset from UTC can be set by configuring the local time zone.

"Synchronization interval"

The interval specifies the time cycle of the time queries (in seconds). The value of the interval ranges between 10 seconds and one day.

If the "Time-of-day synchronization on the full minute" option is enabled, the cycle is automatically set to 60 seconds.

"Time-of-day synchronization on the full minute"

If the option is enabled, the synchronized time of day that the CP forwards to the CPU is adopted by the CPU precisely on the full minute.

If the "Time-of-day synchronization on the full minute" option is enabled, the parameter "Update interval" is automatically set to 60 seconds.

"Accept time of day from non-synchronized NTP servers"

If the option is enabled, the CP also accepts the time-of-day from non-synchronized NTP servers with stratum ≥ 15.

If the option is disabled, the CP does not accept the time-of-day from NTP servers with stratum ≥ 15. If the CP receives a time of day frame from an NTP server with stratum ≥ 15, the time of day of the CP is not set according to the frame.

"Forward time of day to station"

With CP 300/400 this option is enabled as default.

Some CPUs provide the option of requesting the time of day automatically from an NTP server. If the time-of-day synchronization is used by the CPU, you should disable the option described here on the CP. This avoids the time of day acquired by the CPU from the NTP server being overwritten again by the time of day received by the CP. Due to forwarding the time of day from the CP to the CPU there could be less accuracy in this case.

Time-of-day synchronization with S7-1200

Module and synchronization method

When using an external time source, the S7-1200 station can obtain the current time of day both via the CPU as well as via a CP.

With the S7-1200 there is no forwarding of the time of day from the station to the subnet.

Time-of-day synchronization using the CPU

As the synchronization method, with the CPU only NTP can be selected.

With the "CPU synchronizes the modules of the device" option, you can cause all intelligent modules of the station (CMs, CPs) to be synchronized with the CPU time in a configurable synchronization cycle of between 10 and 86400 seconds (24 hours).

Note

Time-of-day synchronization when using Secure Open User Communication (Secure OUC)

If the S7-1200 uses Secure OUC blocks and uses the PROFINET interface of the CP for communication, the "CPU synchronizes the modules of the device" option must be enabled.

This data can only be validated against the validity period of a certificate if the date and time are current. If the date and time are outside the validity period, no communication takes place via Secure OUC.

Time-of-day synchronization using a telecontrol CP

The following synchronization methods are available:

- **NTP**
  NTP can only be used when the telecontrol communication is disabled. In this case, the telecontrol CP is used as an extended PROFINET interface of the CPU.
- **Time from partner**
  When telecontrol communication is enabled, the CP automatically takes the time-of-day from the communications partner.

Note

Forwarding the time to the CPU

Depending on the firmware version of the modules involved, the time-of-day of the CP is forwarded to the CPU in different ways:

- Optional forwarding of the CP time to the CPU using a PLC tag
- Obligatory forwarding of the CP time to the CPU via the backplane bus

Forwarding the time from the CP to the CPU

Note

Recommendation: Time-of-day synchronization only by 1 module

Only have the time of day of the station from an external time source synchronized by a single module so that a consistent time of day is maintained within the station.

When the CPU takes the time from the CP, disable time-of-day synchronization of the CPU.

The forwarding of the CP time to the CPU depends on the firmware version of the CP and the CPU. Note the following behaviour.

- **CP firmware ≤ V2.1.5**
  With this firmware version the CP can make the time-of-day available to the CPU as an option via a PLC tag. When this PLC tag is read cyclically by the CPU, the CPU adopts the CP time.
- **CP firmware ≥ V2.1.7 and CPU firmware ≥ V4.2**
  If both modules in the station have one of the named firmware versions, the time of day of the CP is automatically forwarded to the CPU. Since the CPU automatically adopts the CP time, you no longer require the forwarding option using the PLC tag.

If for the CPU the option "CPU synchronizes the modules of the device" is enabled, all smart modules of the station are synchronized with the CPU time.

Time-of-day synchronization - IE/PB LINK PN IO (V3)

Reference

Configuring time-of-day synchronization of the IE/PB LINK PN IO V3

The requirement is that the time of day is made available by a node on the relevant network.

Procedure for time-of-day synchronization

The following methods can be selected with various options:

- **SIMATIC mode**
  If the SIMATIC mode is enabled, the LINK sets its time-of-day according to MMS time-of-day messages (MMS - Manufacturing Message Specifaction).
  The MMS time-of-day messages originate from a SIMATIC time transmitter or a CPU configured as time master. The time-of-day messages can be received by the CP either from the local CPU or via a LAN. You can set whether and in which direction the LINK will forward the time-of-day frames.
  The advantage of the SIMATIC mode is that it is generally more accurate than the NTP mode.
- **NTP method**
  (NTP - Network Time Protocol)
  With the NTP method, the LINK sets its time according to time-of-day messages originating from an NTP server and made available by a node on the Ethernet network. You can set whether the LINK simply adopts the time-of-day frames from the LAN or adopts it and forwards them.
  In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred. This corresponds to GMT (Greenwich Mean Time).
  The advantage of NTP mode is that it allows the time to be synchronized across subnets.

Configuring time-of-day synchronization

Meaning of the options:

- **Use SIMATIC method - forward from Ethernet to PROFIBUS**
  The LINK adopts the time of day from the LAN and forwards it to the PROFIBUS network.
- **Use SIMATIC method - forward from PROFIBUS to Ethernet**
  The LINK adopts the time of day from the PROFUBUS network and forwards it to the LAN.
- **SIMATIC method - take time of day from the Ethernet network**
  The LINK adopts the time of day from the LAN but does not forward it.
- **SIMATIC method - take time of day from the PROFIBUS network**
  The LINK adopts the time of day from the PROFIBUS network but does not forward it.
- **Use NTP method - forward from Ethernet to PROFIBUS**
  The LINK adopts the time of day from the LAN and forwards it to the PROFIBUS network.
- **NTP method - take time of day from the Ethernet network**
  The LINK adopts the time of day from the LAN but does not forward it.

Additional information

Individual network settings

Individual network settings

Here you can make fixed network settings as required. The "Automatic setting" is selected by default. In normal situations, this guarantees problem-free communication.

If problems do occur in communication (such as when connections cannot be established or frequent network faults occur), this may be caused by an unsuitable selected or automatic network setting. In this case, you should select a network setting that is suitable for your network configuration.

Range of values and example of the IP address

Range of values for the IP address

The IP address consists of four decimal numbers from the range 0 to 255 separated by a period.

Notation: xxx.yyy.zzz.www

Example:

141.80.0.16

- The following addresses have special meanings (*stands for any number between numeric 0 and 255):
  - 127.*.*.* Loopback
  - 0.0.0.0 unspecified network
  - 255.255.255.255 broadcast to all devices (permitted only when "All broadcast devices" was selected as the connection peer)
  - Host address 255 broadcast address for all devices of a network:
    For class A: xxx.255.255.255
    For class B: xxx.yyy.255.255
    For class C: xxx.yyy.zzz.255
- The following are not permitted:
  - Host address 0
    For class A: xxx.0.0.0
    For class B: xxx.yyy.0.0
    For class C: xxx.yyy.zzz.0

Range of values and example of the subnet mask

Range of values for subnet mask

To separate the network ID from the host address in the IP address, the subnet mask was introduced.

The subnet mask consists of four decimal numbers from the range 0 to 255 separated by a period, for example 255.255.0.0.

In their binary representation, the four decimal numbers of the subnet mask must include a continuous series of "1" values without any gaps from the left and a series of "0" values without any gaps from the right.

The "1" values decide the network number within the IP address. The "0" values decide the host address within the IP address.

Example:

Correct values: 255.255.0.0 D = 1111 1111.1111 1111.0000 0000.0000 0000 B

255.255.128.0 D = 1111 1111.1111 1111.1000 0000.0000 0000 B

255.254.0.0 D = 1111 1111.1111 1110.0000 0000.0000.0000 B

Incorrect value: 255.255.1.0 D = 1111 1111.1111 1111.0000 0001.0000 0000 B

Relationship between the IP address and subnet mask

The first decimal number of the IP address (from the left) decides the structure of the subnet mask in terms of the number of "1" values (binary) as follows ("x" stands for the host address or "0" in the subnet mask):

- First decimal number of the IP address subnet mask
  Class A: 0 to 126 255.x.x.x
  Class B: 128 to 191 255.255.x.x
  Class C: 192 to 223 255.255.255.x

Note

You can also enter a value between 224 and 255 for the first decimal number of the IP address. This is, however, not advisable since STEP 7 does not check these addresses.

Relationship between the IP address and address of the gateway

Relationship between IP address and gateway address

The only points in the IP address and gateway address that may differ are those in which "0" appears in the subnet mask.

Example:

You have entered the following: 255.255.255.0 for the subnet mask; 141.30.0.5 for the IP address and 141.30.128.1 for the gateway address.

	Usual decimal representation	Binary representation
Subnet mask	255.255.255.0	1111 1111.1111 1111.1111 1111.0000 0000
IP address	141.30.0.5	1000 1101.0001 1110.0000 0000.0000 0101
Gateway	141.30.128.1	1000 1101.0001 1110.1000 0000.0000 0001

Only the fourth decimal number of the IP address and gateway address may have a different value. In the example, however, the 3rd position is different.

You must, therefore, change one of the following in the example:

The subnet mask to: 255.255.**0**.0 or

the IP address to: 141.30.**128**.5 or

the gateway address to: 141.30.**0**.1

Port parameters

Individual network settings for each port

When necessary, you can make fixed network settings for the transmission characteristics of each available interface (port). The "Automatic setting" is selected by default. In normal situations, this guarantees problem-free communication.

If problems do occur in communication (such as when connections cannot be established or frequent network disruptions occur), this may be the result of unsuitable network settings, either selected or automatic. In this case, you should select a network setting that is suitable for your network configuration.

Functional description of the automatic switchover

All ports of the CP provide a 10/110 Mbps full duplex connection with autosensing and autonegotiation of the network settings. These functions run as follows after turning on the CP:

- The CP attempts to detect the transmission speed used by the partner.
- If detection is possible, the CP attempts to negotiate the optimum duplex mode with the partner.
- If no negotiation is possible, the CP uses the previously detected transmission speed and half duplex.

This takes approximately 2 seconds.

The "Automatic setting" also includes an autocrossing mechanism; this means that you can use a crossover or straight-through cable for the connection regardless of the properties of the partner device.

Changing to individual network settings

If you create any configuration manually, automatic switchover is no longer effective. This also applies to the autocrossing mechanism integrated in the CP. If the connected partner device does not support the autocrossing mechanism, you will need to use a crossover cable (for a switch) or a straight-through cable (for an end device).

"Disable" port option:

Depending on the module type, the drop-down list includes the "- Disable -" option. This option, for example, allows you to prevent access to an unused port for security reasons.

See also

Further information

Please read the manual on the relevant CP for further information.

Maximum frame size / volume of data for PROFINET IO devices

Size of the user data with the CP 1616

The size of the user data (process data) is calculated from the maximum frame size (gross volume of data) minus the values for the modules plugged into this IO device. This results in the net volume of data.

Table 1-155 Example of the calculation of the user data (bytes) of the CP 1616 as an IO device with homogeneous modules (for example only DI modules)

Bytes	Explanation
1434	Gross size of the frame
-16	for the maximum number of pluggable modules
= 1418	Net size of the user data

Table 1-156 Example of the calculation of the user data (bytes) of the CP 1616 as an IO device with heterogeneous modules (for example DI/DO)

Bytes	Explanation
1434	Gross size of the frame
-32	for twice the maximum number of pluggable modules
= 1370	Net size of the user data

Size of the user data with the CP 343-1 (Lean/Standard/Advanced)

The size of the user data (process data) is calculated from the maximum frame size (gross volume of data) minus the values for the CP and the interface itself, for the number of ports and the modules plugged into this IO device. This results in the net volume of data.

Table 1-157 Example of calculating the user data (bytes) of the CP 343-1 as an IO device

Bytes	Explanation
548	Gross size of the frame
-1	for the CP itself
-1	for the Interface
-2	for 2 ports
-32	for the maximum number of pluggable modules
= 512	Net size of the user data *

* By using fewer but larger modules, the volume of user data can be increased.

DNS configuration

Reference

Configuration of the DNS server address in the "DNS configuration" parameter group

A DNS server may be required when the module itself, a communications partner, or for example an e-mail server should be reachable via the host name (FQDN).

DNS server for e-mail connections

When configuring an e-mail connection, the address of the e-mail server via which the e-mails are sent must be specified. The address of the mail server can be specified as an IP address or as an FQDN.

If you specify the server address as an FQDN you need to configure a DNS server. In this case the IP address of the mail server is determined via the configured DNS server.

IP access protection

IP access protection

Reference

Making settings for the interface properties in the "Properties > General > IP access protection" parameter group.

Overview

The following options can be assigned for IP access protection according to the CP type:

- "Enable web server" option
  The CP provides you with the functionality of a web server for access by means of a web browser. Certain HTML pages with CP information and diagnostic functions are stored in a memory area of the CP for this.
  Enable this option in order to be granted access to these HTML pages. Port 80 of the CP is thereby enabled.
  Web server access is enabled by default.
- "Enable FTP server" option
  Select this option if you want to allow FTP access to the S7 station via Port 20/21 of the CP.
  If you configure FTP access to file DBs in the CPU in the "FTP" tab, access is only possible if you select the option here.
  FTP server access via Port 20/21 is enabled by default.
- "Enable access protection for IP communication" option
  The Ethernet interface allows you to restrict access to the local device to partners with specified IP addresses. Partners you have not authorized then have no access to data on the local device via this interface.
  Here, you can enable or disable IP access protection and can also enter certain IP addresses in an IP address control list (IP-ACL).
  As default, IP access protection is deactivated.
  Blocked access attempts are registered on the CP and can be viewed with special diagnostics in the "IP access protection" diagnostic object. If the CP has IT functionality, a LOG file is also created in the file system of the CP that you can view with a WEB browser. Please refer to the instructions on handling the LOG file and the notes on loading the configuration data below.

IP access protection for configured communication partners

If you want to restrict access so that only the communication partners you specified in the configuration have access to the local device, you simply need to enable access protection. In this case, you do not need to enter any IP addresses in the list.

These communication partners include:

- Stations to which communication connections are configured;
  With the exception of S7 connections, this also applies to connections on which the connection partner is located in a different subnet.
  You should, however, remember that where you have unspecified connections whose partner addresses are unknown; all IP addresses that are not entered are unauthorized and will be rejected. Even connections in PROFINET CBA will be treated as unspecified connections. You will need to enter the IP addresses of such connections explicitly in the IP access control list.
- PROFINET IO devices when the Ethernet CP is used as a PROFINET IO controller;
  The IP addresses of PROFINET IO devices are entered dynamically in the IP access control list when the CPU is in RUN mode.
  If the CPU changes to STOP mode, the IP addresses of the PROFINET IO devices are deleted from the access list.
- NTP servers, SMTP servers, DNS servers and DHCP servers.
  The IP addresses of NTP servers, SMTP servers, DNS servers and DHCP servers are also entered and removed dynamically when there are requests to these servers.

IP access protection relates to all connection types that use the IP protocol (TCP, ISO-on-TCP, UDP, S7)

Note on dynamically assigned IP addresses:
Since each service manages its dynamic entry in the ACL itself, it is perfectly possible for the same IP address to appear several times in module diagnostics.

IP access protection for partners with specific IP addresses

To allow access only by certain IP addresses, enter these IP addresses in the IP access control list. These can, for example, be the IP addresses of connection partners that remained unspecified in the connection configuration, of individual programming devices or of connection partners on PROFINET CBA.

The IP addresses you specified in the connection configuration always belong to the permitted IP addresses and do not therefore need to be entered explicitly in the IP-ACL.

Viewing the LOG File with a Web browser

On CPs with IT functionality, a LOG file is created in the file system of the CP and this can be viewed with a Web browser. Compared with the recording made in the special diagnostics, the LOG file provides space for up to 512 entries.

The LOG file is available only after activating IP access protection the first time.

You will find the LOG file as an HTML file in the file system of the CP in the following directory:

·ram/security/IPLogFile.htm

Further properties:

The LOG file is created as a ring buffer. When more than 512 entries have been recorded, the oldest entries are then overwritten.

Entries are made chronologically and there are no other criteria for sorting.

Note that this function is supported only by the newer firmware versions of the CPs; you will find more detailed information in the CP device documentation.

With Advanced CPs as of CP 343-1 Advanced (GX30) and CP 443-1 Advanced (GX20) the LOG file is not created. On these CPs, you can view the blocked access attempts directly with Web diagnostics.

Entering access permissions in the IP access control list (IP-ACL)

Reference

Making settings for the interface properties in the "Properties > General > IP access protection" parameter group.

Rules for entering the IP addresses:

Keep to the following rules when entering IP addresses:

- You can enter the IP address singly or as a range.
- You can also enter comments.
- The maximum number of addresses you can enter is: 32
- The maximum number of address ranges you can enter is: 10
- The system does not check whether addresses that have already been entered singly are also included in an address range.
- The system does not check whether a range includes invalid addresses (for example, broadcast addresses can be specified here although they cannot occur as the IP address of a sender).
- Comments begin with the "#" character

Examples:

- Individual IP address with comment: 141.80.0.16 #address of partner x
- Range of IP addresses with comment: 141.80.0.16 - 141.80.0.25 #address range for control level

Rights

The access permissions described here can be selected in the IP access control list with their short designations.

Table 1-158 Access permissions and short designations

Access right	Short designation	Meaning / remark
Access to station	A	**Access** Communications partners with addresses in the specified range have access to the station (CP / CPU) belonging to the CP. This access permission is set implicitly for IP addresses you have specified in the connection configuration.
Modifying the IP access control list	M	**Modify** Communications partners with addresses in the specified range, have the right to access the IP access control list. These communications partners can send entries for the IP access control list to the CP using HTTP. If you select this permission, the permission "A" is also set. Notes on the entries sent using HTTP: • With each list transferred using HTTP, a previous list sent using HTTP becomes invalid. • The access permissions sent using HTTP can be added to entries configured in STEP 7, but the entries cannot be deleted. • A list transferred using HTTP is deleted if there is a power down on the CP (power OFF). You will find information on the syntax of a list transferred using HTTP in the documentation of the CP.
IP routing access to the another IP subnet	R	**Routing** Communications partners with addresses in the specified range have access to other subnets connected to the CP. This access permission is not set automatically for IP addresses you have specified in the connection configuration. Where necessary, this access permission must be set here explicitly.

Note

- **Loading configuration data via TCP/IP**
  When downloading via TCP/IP with access protection enabled, note that you will need to include the IP address of the programming device / PC from which the configuration is downloaded in the IP-ACL. The programming device / PC would otherwise be detected as unauthorized during the loading and the loading routine would be aborted
- **Loading S7 connections later**
  If you want to load individual S7 connections to the S7 station later, these are not included automatically in the IP-ACL. In this case, you will need to load the entire station configuration again.
- **Adopting and editing a configuration by loading to the programming device / PC**
  If you want to edit the connection configuration and have enabled IP access protection, you will require the full connection information on the programming device / PC. This means that you will need to upload the current connection data. When you then load the data to the S7 station again, you will need to load the entire connection data and station data.

Note

Using special diagnostics, you can view the current IP-ACL on the S7 station and, if required, compare it with the configured IP-ACL.

Editing access permissions in the user management

Reference

Setting user permissions in the "Properties > General > User management" parameter group

Meaning of user management

In user management, you specify which users have which permissions when accessing the S7 station via a Web browser or as an FTP client.

In an alphabetical list, under user name you will find the users that have already been entered for which passwords have been stored.

The "All" entry is present as default. This cannot be modified. It is also not possible to assign a password to this entry.

User entries are always linked to a password. Exception: The "All" entry is not password protected.

Note

As default, no rights whatsoever are assigned to the "All" entry. For service purposes, however, it is possible to assign permissions.

You should, however, remember that any permissions assigned to the "All" user are also available to every other user. If you do assign permissions for service purposes, remember to cancel these again afterwards! Otherwise, you allow services to be executed without authorization with every access.

If an access right is set for the "All" user, this is recognizable by the filled check box when you assign rights to other users.

Entering new users

1. Select the first free row in the user list.
   A new user entry is created with the access level "Minimum".
2. Adapt the proposed user name in the "Name" column.
3. Double-click in the "Access level" column of the newly created row.
   The icon for the drop-down list becomes visible.
4. Click on the icon of the drop-down list.
5. In the dialog that then opens, set specific user permissions.
6. Confirm your entries with "OK".
7. Assign a new password in the password dialog that opens.

Changing user permissions or a password:

1. Double-click in the "Access level" column of the row of the required user entry.
   The icon for the drop-down list becomes visible.
2. Click on the icon of the drop-down list.
3. In the next dialog, modify the specific user permissions or click on the "Password" button to assign a new password.
4. Confirm your entries with "OK".
5. Enter the password in the password dialog that opens.

Deleting user entries

1. Select "Delete" in the context menu of the row of the required user entry.

Tag declaration - Edit tag list

Reference

Setting the tags configured on the CPU in the "Properties > General > Tag declaration" parameter group.

Meaning of the tag declaration

To access tags of the CPU using a Web browser and Java applet, the CP must be aware of the names, addresses, and access rights of these tags.

In the symbol list in the tag declaration, you can see the currently known tags with their symbolic names, the corresponding operands and the assigned access rights.

Adding a new tag entry

1. Double click on the first free row in the table.
   The symbol becomes visible for selection from the drop-down list.
2. Select the symbol to open the structure display of the tags on the current device.
3. Select the symbols of the tags or the structure elements to which access will be possible using a Web browser.
   Note: You must first store the symbols in the Symbol Editor!

Deleting a tag entry

1. Select "Delete" in the shortcut menu of the row of the tag to be deleted.
   Caution:

   Note

   There is no prompt for confirmation, the entry is deleted immediately. Multiple selections are not possible.

Setting access permissions - "Writable" option

The default permissions for the tags are read-only.

To assign write/read rights for the tags entered here, follow the steps below:

1. Select one or more tags in the symbol list.
2. Set the "Only reading allowed" option as required for the selected tags.

Tag declaration - Edit tag list (ERPC)

Reference

Configuration of the tags created on the CPU for the ERPC application

You select the tags you want to make accessible to the ERPC application.

Table of the ERPC tags

The table shows the PLC tags configured in the STEP 7 project that are available for the ERPC application. Any ERPC tags that may have been downloaded to the station previously are not displayed.

The following information is displayed in the table of tags configured for ERPC:

- Info
  Additional information from the check of changes made
  ERPC tags that you have changed in the STEP 7 project since the last time you saved have additional information displayed in the "Info" column by the check changes function (see below).
- Tag
  Name of the tag
- Writable
  The permissions configured for ERPC communication (reading from the CPU, writing to the CPU), as default only read permissions are assigned.
- Comment
  Comments on the ERPC tags specifically for ERPC communication.

Below the table:

- Writable
  By enabling the option "Writable", you cancel the write protection of the selected ERPC tags and release them for write jobs of the ERPC application.
- Number of tags:
  The total number of tags used for ERPC

Multiple selection of tags

You can select more than one tag using the <Shift> and <Ctrl> keys.

Check for changes

When you open the tag declaration, a check is made for any changes. The tags selected for the ERPC application are compared with the those in the tag table of the assigned CPU in the STEP 7 project (offline tag table).

Note

There is no comparison of ERPC tags you may have downloaded previously from the STEP 7 project to the station.

The "Info" column displays the relevant information on differences between the ERPC tags and those of the tag table of the CPU.

Display in the "Info" column	Status / information
	Tag unchanged
ACC	The tag cannot currently be edited. Reason: The data block in which the tag is used is currently open.
DEL	Tag deleted in the tag table of the CPU
ADR	Address changed in the tag table of the CPU
TYP	Data type changed in the tag table of the CPU Note: If the address and the data type of a tag are changed in the tag table, the "ADR" information is overwritten by the "TYP" info.
DEF	Tag type not supported
UPL	Tag uploaded from the station (after using the STEP 7 function "Upload to PG")

If entries of the type "DEL", "ADR" and "TYP" exist, you will need to make suitable corrections before you save the STEP 7 project.

FTP configuration for FTP server mode

Ethernet CP as FTP server for the CPU data

Reference

Data acquisition for automatic creation of a file allocation table in the "Properties > General > FTP configuration" parameter group.

With the information in the file allocation table, it is possible to address data blocks in one or more CPUs (up to 4) in an S7 station.

"Activate FTP server" option (only with certain CP types)

Select this option to allow FTP access to the S7 station via port 20/21 of the CP.

This option needs to be enabled to enable FTP access to file DBs on the CPU with the option described below.

FTP server access via Port 20/21 is enabled by default.

"Use FTP server for S7 CPU data" option

If you select this option, the file allocation table configured here is created on the CP when you download the project engineering data and store it in the /config folder of the file system of the CP.

An existing file_db.txt file is overwritten.

Note

With security and the "Allow access only via FTPS" enabled, the following applies:

- The files are transferred encrypted.
- For the user, the rights "FTP: Read files (DBs) from the S7 CPU" or "FTP: Write files (DBs) from the S7 CPU" must be activated.
- If the firewall is activated, the FTP/FTPS protocols must be allowed.

Ethernet CP as FTP server for the S7 CPU data

To transfer data using FTP, you create data blocks on the CPU of your S7 station; due to their special structure, these are known here as file DBs.

When operating as an FTP server, if the Ethernet CP receives an FTP command, it checks a file allocation table (file file_db.txt) to find out how the data blocks used in the S7 station for file transfer are mapped to files.

You can create the file allocation table and transfer it to the CP as follows:

- With an entry in the parameter group "Properties > FTP configuration" described here
  The file allocation table is then created in the FTP configuration based on this information and downloaded automatically to the CP along with the configuration data.
- By creating a file_db.txt file directly
  If you create the file allocation table in this way, you will need to load it to the CP using an FTP command.

The file allocation table file_db.txt is stored in the file system of the Ethernet CP in the /config directory.

See also

Layout and structure of the FTP configuration list (Page 3074)

Structure of the data blocks (file DBs) for FTP services - FTP server mode (Page 3075)

Layout and structure of the FTP configuration list

Layout and structure of the FTP configuration list for generating the file allocation table

The fields of the table in the FTP configuration have the following meaning and syntax:

Titles in the table for FTP configuration	"CPU"	"DB"	File name	Comment
Meaning	Rack/slot assignment of the CPU Can be selected from a drop-down list; cannot be typed in.	Number of the data block Entry can be typed in	File name assigned to the file DB Automatic name proposal; entry can be typed in / modifiable.	Informal comment
Example 1	cpu1 [PLC_1]	20	cpu1_db20.dat	Measured values in the Plant1 area
Example 2	cpu2 [PLC_2]	35	Cpu2_db35.dat	Measured values in the Plant2 area

Notes on the syntax

- Valid delimiters for the entries are "spaces".
- The following applies to the file name of a file DB:
  - The file name begins with "cpuX" (where X=1, 2, 3 or 4);
  - "cpuX" must first be defined in the rack/slot assignment;
  - Length: Maximum of 64 characters (including "cpuX");
- Maximum of 100 entries;
- Permitted characters: Letters "A-Z, a-z"; numbers "0-9", "_", "."

Note

Keep to the notation (lower case for "cpu" and no leading spaces at the start of the row). Otherwise, the files will not be recognized.

The tabulator is **not** permitted as the delimiter.

See also

Ethernet CP as FTP server for the CPU data (Page 3072)

Structure of the data blocks (file DBs) for FTP services - FTP server mode

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length (20 bytes) and structure)
- Section 2: User data (has a variable length and structure) data type "FILE_DB_HEADER

Data type "FILE_DB_HEADER"

Using the "Add new block" function, you assign the type "FILE_DB_HEADER" to a data block directly.

File DB header for FTP server mode

Note: The file DB header described here is largely identical to the file DB header for client mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data ("file does not exist"). • 1: The file DB contains valid user data ("file exists").	The dele FTP command sets EXIST=0; The store FTP command sets EXIST=1;
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The stor and retr FTP commands set LOCKED=1 when they are executed. The following function is also possible when writing from the user program: The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program: 1. Check LOCKED bit; if = 0 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit; if = 0 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program on the S7-CPU must set NEW=0 to allow store to be used again or to be able to delete the file with the dele FTP command.

Parameter	Type	Value / meaning	Supply
WRITE_ACCESS	BOOL	0: The FTP client on the PG/PC has no write access right for the file DBs on the S7 CPU. 1: The FTP client the PG/PC has the write access right for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	This parameter is irrelevant in FTP server mode.	Is set to "0" by the FTP server.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the IT-CP (the initialization value is 1.1.1994 0.0 (midnight)).

See also

Ethernet CP as FTP server for the CPU data (Page 3072)

I/O addresses

Reference

Setting the input/output addresses in the "Properties > General > I/O addresses" parameter group

Meaning

- Inputs, outputs
  Assign a start address to the module. (Outputs only if the option "Address setting for LOCK/UNLOCK with FETCH/WRITE" is selected)
- "Address setting for LOCK/UNLOCK with FETCH/WRITE" option
  Select this option if you want to use the access coordination function with the FCs LOCK/UNLOCK in FETCH/WRITE mode.
  This function uses process output via the backplane/P bus. As a result, the output addresses can be set as soon as this option is selected.

Web

Web server of the module

"Activate Web server" option

The CP provides you with the functionality of a web server for access by means of a web browser. HTML pages with information and diagnostics functions are stored in the memory area of the CP for this.

Enable this option in order to obtain access to these HTML pages in the Web server of the CPU. When you enable the function, port 80 of the CP is released.

Web server access is enabled by default.

Web diagnostic options (availability depends on the CP type)

- "Display topology" option
  The option allows the display of topology information for the networked PROFINET IO interfaces of the CP in the Web server.
  To collect and store the topology information, additional memory is taken up on the CP. With complex project configurations making high demands on the memory resources, it may therefore be advisable to activate the option only temporarily for service purposes.
- "Download firmware via Web" option
  By enabling the option, the function for downloading the firmware of the CP from the download center is enabled in the Web server.
  This option is not restricted by the user administration if security is disabled. It is therefore recommended that the option is enabled only when required.
- "Reload of language files for the diagnostics displays via Web" option
  Diagnostics displays of the CPs are shown in plain language in the Web diagnostics buffer. These displays are language-specific.
  If you enable the option, the function for reloading missing language files from the download center is enabled in the Web server.

Automatic update / "Update interval"

If you select the "Enable" option, the CP updates the displayed Web pages at the selected intervals.

Range of values for the update interval 1 to 999 s

Web diagnostics on the module

Requirement:

Establish a physical connection between the programming device and the SIMATIC S7 station and set the PG/PC interface so that the module is accessible. Further help is available in the "Set PG/PC Interface" dialog box.

"Start of Web diagnostics"

Click the "Web diagnostics" button to display the result of the module diagnostics in the Web browser. The content is supplied by the integrated HTTP server of the module.

Start diagnostics in the Web browser in the "Properties > General > Web diagnostics" parameter group.

- "Access via" option
  From the "Access via" list, select the interface via which the CP is addressed.
  - Available interfaces can be selected in the drop-down list.
  - You can also enter an IP address.

Note

With security functions enabled (if available)

If the option "Allow access only using HTTPS" is enabled, note the following:

- The data is transferred encrypted.
- The "Access to Web diagnostics" right must be activated for the user.
- If the firewall is activated, the HTTP/HTTPS protocols must be allowed.

SNMP

"Activate SNMP" option

If the "Activate SNMP" option is selected, communication using SNMP is enabled allowing read and write access to the module.

You can only select communication according to the SNMPv1 or SNMPv3 specification with modules that support Security functions:

- SNMPv1
  The SNMPv1 agent is enabled on the module. You can change the community strings for read and write access for SNMPv1 for modules that support Security functions.
- SNMPv3
  The SNMPv3 agent is enabled on the module. Further configuration options for encryption, authentication and the assignment of rights for STEP 7 users are available.

CP - OPC UA

Security profiles and certificate validation

Security profiles of the OPC UA server

The following security profiles are supported by the server application of the CP:

- Supported security profiles
  - No security profile
    The CP does not use any Security profile.
  - Basic128Rsa15
    This corresponds to the security profile "Basic128Rsa15" of the OPC UA specification. The CP uses signing and, if configured, 128-bit encryption.
  - Basic256
    This corresponds to the security profile "Basic256" of the OPC UA specification. The CP uses signing and, if configured, 256-bit encryption.
  - Basic256Sha256
    This corresponds to the security profile "Basic256Sha256" of the OPC UA specification. The CP uses signing and, if configured, 256-bit encryption using the hash algorithm SHA-256.

If you enable several options, then depending on the settings on the communications partner (client), the CP selects the profile with the the highest possible security.

- Sign and encrypt
  The options listed above have the following effects:
  - Sign
    The CP only allows communication with signed frames.
  - Sign and encrypt
    The CP only allows communication with signed and encrypted frames.

Certificate validation

In this parameter group you set the options for checking the certificates of the communications partner for the server or client application.

- Checking the certificates
  The CP always checks the certificate of the communications partner.
  If the partner certificate is invalid or is not trustworthy, communication is aborted.
  If the option is enabled, the two following options for restricting the checking routines can be activated.
- No strict certificate validation
  If the option is enabled, the CP allows communication in the following situations:
  - The IP address of the communications partner is not identical to the IP address in its certificate.
  - The use stored in the certificate (OPC UA client/server) differs from the function (OPC UA client/server) of the communications partner.
  - The current time on the CP is outside the period of validity of the partner certificate.

  Regardless of these exceptions, to establish a connection, at least the following requirements must be met:
  - The application URI sent by the requesting client must match the URI of the server application of the CP.
  - If the partner certificate is not trustworthy, the CP must at least have stored a self-signed certificate of the partner.
  - If the partner certificate was issued by several CAs, all CAs must be saved in the certificate store of the CP.
- Do not check period of validity
  If the options is enabled the CP checks the certificate of the communications partner. The CP also allows communication in the following situation:
  - The current time on the CP is outside the period of validity of the partner certificate.

If none of the options is enabled, no certificates are checked.

In particular for the establishment of the communication with third-party applications, note the information in the certificate management on importing certificates.

Special features for the client application of the CP 443-1 OPC UA

If you use the client function of the CP, note the following:

The value of the parameter "CheckServerCertificate" that you programmed in the connection information (UASessionConnectInfo) for the client program block "UA_Connect" is overwritten by the configured settings in the "Certificate validation" parameter group. If the client is to check the certificates of the communications partner (server), you can ignore the parameter in the UDT "UASessionConnectInfo". For the certificate check only the configured values of "Certificate validation" parameter group are relevant.

OPC UA PubSub - TSN

1.6.1.2 Industrial Ethernet (S7-1500)

FTP configuration (S7-1500)

CP as FTP server for the S7 CPU data (S7-1500)

Ethernet CP as FTP server for the CPU data (S7-1500)

Reference

Configuration of the FTP server mode in the parameter group of the Ethernet CP

- With security disabled in "Properties > General > FTP configuration"
- With security enabled in "Properties > Security > General > FTP configuration"

S7-1500 CP as FTP server

The functionality described here allows you to transfer data in the form of files to data blocks or from data blocks of an S7 station using FTP commands. At the same time, the conventional FTP commands for reading, writing and managing files can also be used.

To transfer data with FTP, create data blocks on the CPU of your S7 station. Due to their special structure, these are known as file DBs.

When it receives an FTP command, the CP 1543-1 acting as FTP server queries a file allocation table to find out how the data blocks used for file transfer in the S7 station will be mapped to files.

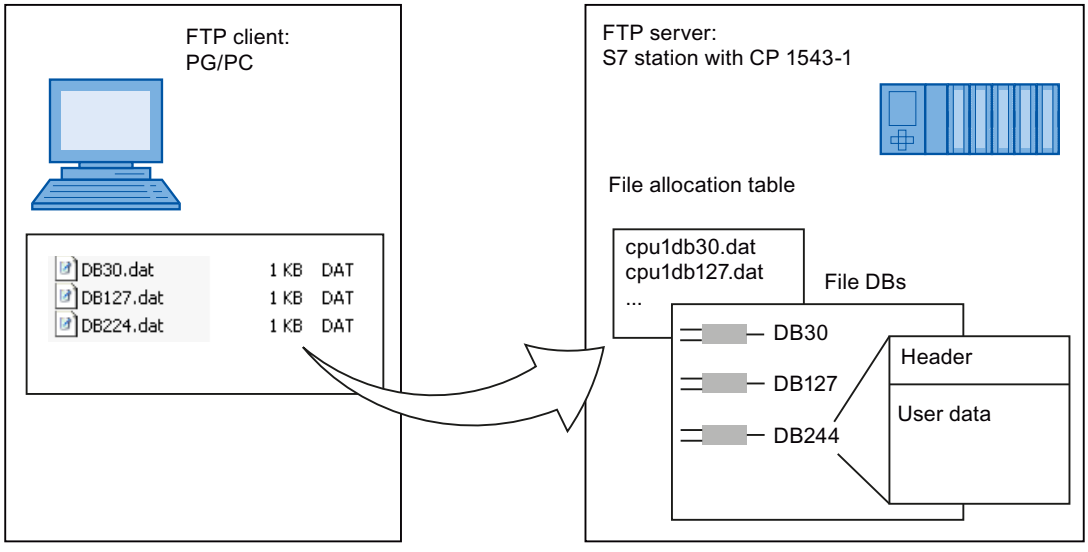

Figure 1-99 S7 CPU with CP 1543-1 as FTP server for the S7 CPU data

File allocation table

The CP 1543-1 as FTP server requires information on how the data blocks used in the S7 station for file transfer are mapped to files.

The file allocation table contains the following entries:

- Assignment of the CPU
- DB assignment
- File name

Requirements in the CPU configuration

As file DBs create data blocks of the type "Array of Byte" or "Array of Char".

For the CP with firmware V1.x use the following settings to allow FTP access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.
- For all data blocks being used as file DBs, disable the "Optimized block access" attribute.

These two settings are no longer necessary for the CP 1543-1 as of firmware version V2.

See also

Layout and structure of the FTP configuration list (Page 3083)

FTP commands on the FTP client (Page 3084)

Structure of the data blocks (file DBs) for FTP services - FTP server mode (Page 3086)

Layout and structure of the FTP configuration list (S7-1500)

Layout and structure of the FTP configuration list for generating the file allocation table

The fields of the table in the FTP configuration have the following meaning and syntax:

Titles in the table for FTP configuration	"CPU"	"DB"	File name	Comment
Meaning	Assignment of the CPU Can be selected from a drop-down list; cannot be typed in.	Number of the data block Entry can be typed in	File name assigned to the file DB Automatic name proposal; entry can be typed in / modifiable.	Informal comment
Example	cpu1 [PLC_1]	20	cpu1_db20.dat	Measured values in the Plant1 area

Notes on the syntax

- The following applies to the file name of a file DB:
  - The file name begins with "cpuX" (where X=1 for S7-1500);
  - Length: Maximum of 64 characters (including "cpuX");

Note

Keep to the notation (lower case for "cpu" and no leading spaces at the start of the row). Otherwise, the files will not be recognized.

See also

Ethernet CP as FTP server for the CPU data (Page 3082)

FTP commands on the FTP client (S7-1500)

Example of access

The following figure shows an example of a typical access sequence in the MS-DOS window.

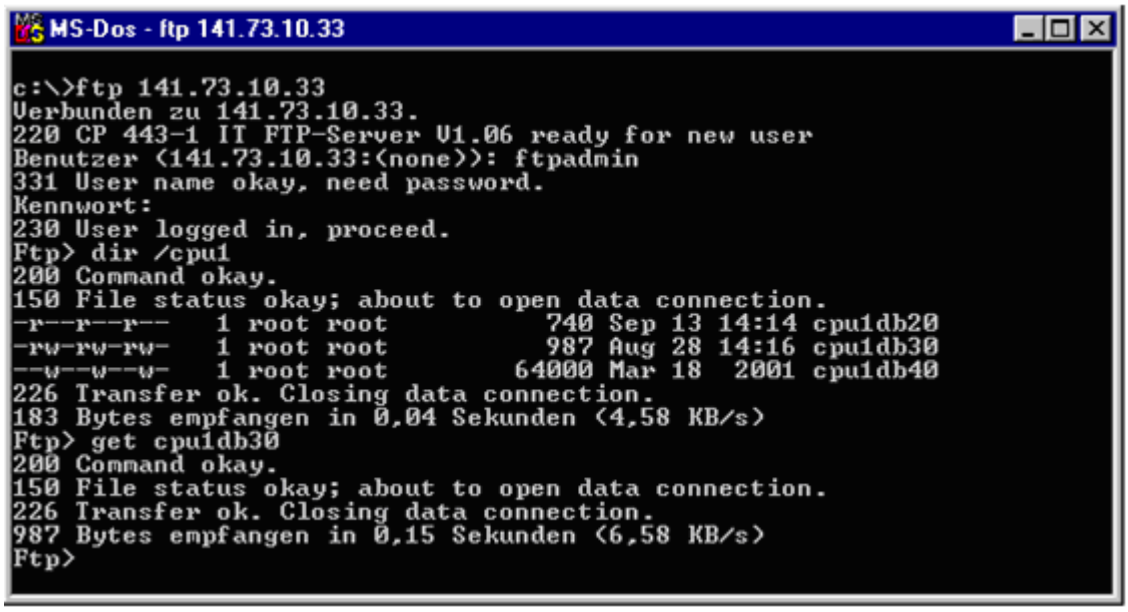

How typical FTP commands use the permitted FTP functions

The following table shows which FTP commands can be executed to access file DBs on the CPU. The table also shows which FTP functions are used for these FTP commands in typical input consoles such as the MS-DOS prompt.

Table 1-159 FTP commands

Typical FTP functions						FTP command	Meaning
open	dir	put	get	close	del		
x						User	Logging on
x						Pass	Authorization by password
	x	x	x			Port	Specifies the port via which the client wants to transfer.
	x					List	Lists the file DBs in the addressed CPU.
					x	Dele	Deletes a file DB by setting the EXIST bit in the file DB header to "0".
			x			Retr	Reads the user data in the specified file DB into the specified file on the FTP client.
		x				Stor	Transfers the specified file from the FTP client to the user data area in the specified file DB.
				x		quit	Terminates the current FTP connection.

Note

You cannot use the FTP commands "rename", "append", "rnfr" and "rnto" with the file DBs.

How FTP commands are executed on the CP

To illustrate how the FTP interface to the file DBs on the CPU works, the execution sequence is explained below based on the example of the stor command.

The FTP server on the CP executes the sequence shown below:

1. Identification of the addressed file DB based on the entry in the file allocation table.
2. Check of the bits in the file DB header.
   The write is executed only when the following is recognized:
   LOCKED bit = 0
   NEW bit = 0
   WRITEACCESS bit = 1

3. Writes the file content to the user data area of the file DB on the CPU. At the beginning of the write function, the LOCKED bit is set and reset when writing is completed.
4. When the write function is completed, the NEW bit is also set in the file DB header and the current date entered in the DATE_TIME field.
5. The FTP server sends a message about the file transfer event to the FTP client.

Transfer mode for file transfer

File transfer only uses the binary mode. To change to this mode, enter the command "binary" in the input console after logging on.

See also

Ethernet CP as FTP server for the CPU data (Page 3082)

Structure of the data blocks (file DBs) for FTP services - FTP server mode (S7-1500)

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length (20 bytes) and structure)
- Section 2: User data (has a variable length and structure)

DB with tag "FILE_DB_HEADER"

In STEP 7 in "Program blocks" > "Add new block" create a data block.

For the attributes of the DB refer to the following section.

As the data type of the first tags in the DB, select the data type "FILE_DB_HEADER". The parameter structure of the SDT is created automatically.

Requirements in the DB attributes

Use the following settings to allow FTP access:

For all data blocks being used as file DBs, disable the "Optimized block access" attribute.

Note

Optimized access as of CP firmware V2.0

If you use the CP 1543-1 as of firmware version V2.0 in STEP 7 as of V14, you can use the option "Optimized block access".

File DB header for FTP server mode

Note: The file DB header described here is largely identical to the file DB header for client mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data ("file does not exist"). • 1: The file DB contains valid user data ("file exists").	The dele FTP command sets EXIST=0; The store FTP command sets EXIST=1;
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The stor and retr FTP commands set LOCKED=1 when they are executed. The following function is also possible when writing from the user program: The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program: 1. Check LOCKED bit; if = 0 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit; if = 0 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0

Parameter	Type	Value / meaning	Supply
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program on the S7-CPU must set NEW=0 to allow store to be used again or to be able to delete the file with the dele FTP command.
WRITE_ACCESS	BOOL	0: The FTP client on the PG/PC has no write access right for the file DBs on the S7 CPU. 1: The FTP client the PG/PC has the write access right for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	This parameter is irrelevant in FTP server mode.	Is set to "0" by the FTP server.
DATE_TIME	DATE_AND_TIME	Date and time of the last change to the file The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the IT-CP (the initialization value is 1.1.1994 0.0 (midnight)).

See also

Ethernet CP as FTP server for the CPU data (Page 3082)

CP as FTP client for the S7 CPU data (S7-1500)

How the FTP client works (S7-1500)

FTP client

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station.

The user program sends FTP jobs that are executed by the CP 1543-1 as an FTP client. In the user program, use the FTP_CMD instruction for connection establishment and data transfer.

Among other things, you specify the following in the job:

- IP address of the FTP server,
- storage location of the file on the FTP server
- File names
- Access information

The following schematics illustrate how the function works when using the FTP_CMD instruction.

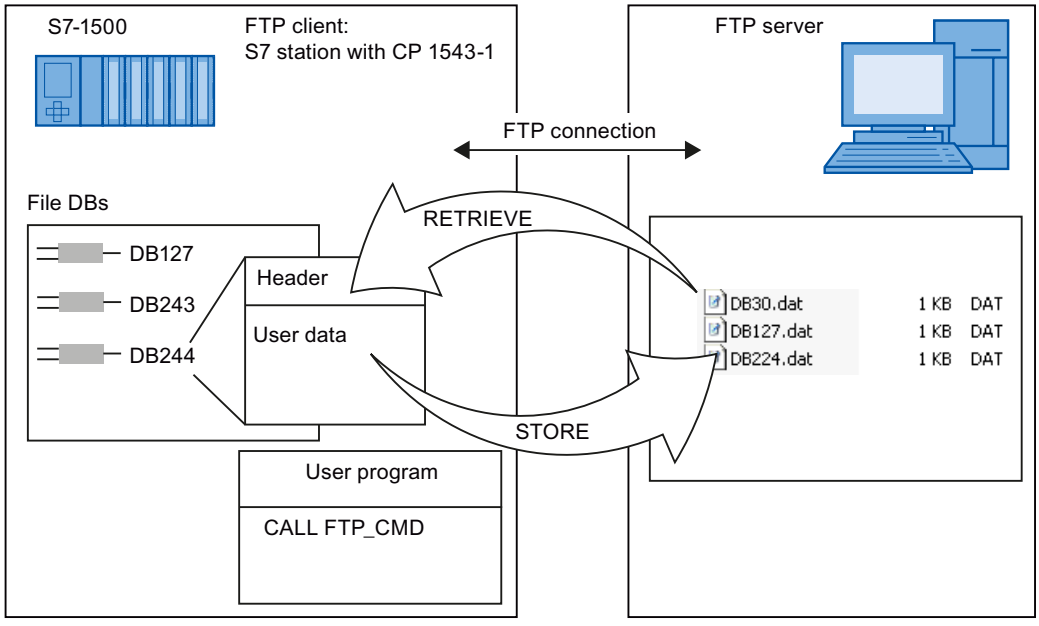

Figure 1-100 CPU with CP 1543-1 as FTP client for the S7 CPU data

As of firmware version V2 of the CP, for the user data the data types "Array of Byte" and "Array of Char" can also be used.

Requirements in the CPU configuration

If you use the CP 1543-1 with firmware version V1.x make the following settings to allow FTP access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.
- For all data blocks being used as file DBs: Disable the "Optimized block access" attribute.

These settings are no longer necessary for the CP 1543-1 as of firmware version V2.

FETCH/WRITE mode (S7-1500)

Use of FETCH/WRITE (S7-1500)

FETCH/WRITE

You can use the FETCH/WRITE services to access system memory areas of the S7-CPU.

- FETCH: Read data directly
- WRITE: Write data directly

Access is possible from the following devices:

- SIMATIC S5
- SIMATIC PC stations
- Non-SIMATIC devices

The S7-1500 CP is the server for FETCH/WRITE services.

The FETCH/WRITE services are supported by the following protocols:

- TCP
- ISO-on-TCP
- ISO transport

A configured connection is not required for using the FETCH/WRITE services.

System memory

You can access the following operand areas in the system memory of the S7-1500 CPU using FETCH or WRITE:

- Data blocks (DB)
  Note the following limit for DB access: the highest DB no. is 255.
  Note the "S7 addressing" parameter for access of S5 or S7 partners when configuring the TSAP table of the CP.
- Bit memory (M)
- Process image input (PII)
- Process image output (PIQ)

- I/O area inputs (PIW, PID, PIB)
- I/O area outputs (PQW, PQD, PQB)
- Counters (C)
- Timers (T)

Requirements in the CPU configuration

Use the following setting in the CPU configuration data to allow FETCH/WRITE access:

- Parameter group "Protection & Security > Connection mechanisms":
  Disable the option "Permit access with PUT/GET communication from remote partner".

For information on the configuration of the CP, refer to the section Configuring FETCH/WRITE access (Page 3091).

Link to non-SIMATIC systems

The FETCH and WRITE mode that is supported for ISO transport connections and ISO-on-TCP or TCP connections can in principle be used to access S7 system memory areas from any non-SIMATIC devices.

To be able to use this type of access also for PC applications, for example, you need to know the PDU structure of the jobs. For more information, refer to section Linking to other systems with FETCH/WRITE (Page 3093).

Configuring FETCH/WRITE access (S7-1500)

For information on the mechanisms of FETCH/WRITE, refer to the section Use of FETCH/WRITE (Page 3090).

FETCH/WRITE

Activate the option to enable FETCH/WRITE access to the S7-1500 CPU via the CP.

For FETCH/WRITE access, create a row for each partner in the table and enter the access identifications in a table. Only FETCH/WRITE access operations whose identification matches an entry in the list are permitted.

The following inputs are required depending on the protocol used for FETCH/WRITE access:

- ISO-on-TCP / ISO-Transport
  TSAP of the partner
- TCP
  Port number of the partner
  For information on usable port numbers, refer to the section Port settings (Page 3092).

TSAP list / TCP port list

- TSAP
  Enter the TSAP of the partner as ASCII string or use the cell of the "TSAP as hex" column instead.
  The corresponding value is automatically added to this column with an entry in the "TSAP as hex" column.
- TSAP as hex
  Enter the TSAP of the partner as hexadecimal value or use the cell of the "TSAP" column instead.
  When entered as hexadecimal value, each character of the TSAP is entered as hexadecimal number separated by periods.
  The corresponding value is automatically added to this column with an entry in the "TSAP" column.
- TCP port
  Enter the port number of the partner.
- S7 addressing
  Use this option to specify the addressing mode when accessing DBs for the FETCH/WRITE call in the SIMATIC S7 station:
  - Option enabled
    S7 addressing: Byte address
    This is the default setting for accessing SIMATIC S7 stations.
  - Option disabled
    S5 addressing: Word address
    This addressing makes it possible for applications to access S5 or S7 stations without adapting the addresses. This is useful for existing S5 applications, for example, that can now be used unchanged when accessing S7 stations.

Port settings (S7-1500)

Port settings

The following table provides information about the values and uses of port addresses that are referenced in the connection descriptions of the OUC program blocks.

Port addresses		Use/note
0		Permanently assigned; must not be used!
1 .. 1023		Ports from this band should not be used (well known ports).
1024 .. 49151	Ports for application-specific protocols	
	2000 .. 5000	Range in which STEP 7 looks for free port addresses and assigns them. You can set the port address in this range individually.
	5001 .. 49151	The port addresses above 5000 are used by the system! Note: If you want to use these port addresses, please contact your system administrator to verify availability.
49152 .. 65535		Dynamically assigned ports The use of these ports is not recommended.

Linking to other systems with FETCH/WRITE (S7-1500)

In principle, the mode for FETCH and WRITE can be used by any third-party devices to access S7 system memory areas.

To be able to use this type of access also for PC applications, for example, you need to know the PDU structure of the jobs. The required S7-specific or S5-specific headers for request and response frames are 16 bytes long as standard. See the following tables for the structure.

Structure of WRITE frame (PDU)

The meaning and values of parameters that are not assigned fixed values can be found in the section "Parameter values".

WRITE request frame			WRITE acknowledgment frame		
0	System ID	="S"	0	System ID	="S"
1		="5"	1		="5"
2	Length header (bytes)	=16 (dec.)	2	Length header	=16 (dec.)
3	ID OP code	=01	3	ID OP code	=01
4	Length OP code	=03	4	Length OP code	=03
5	OP code	=03	5	OP code	=04
6	ORG field code	=03	6	Ack field	=0Fh
7	Length ORG field	=08	7	Length ack field	=03
8	ORG ID		8	Error field	=No.
9	DBNR		9	Empty field code	=FFh
A	Start address	High byte	A	Length empty field	=07
B		Low byte	B		
C	Length	High byte	C		
D		Low byte	D	Not used	
E	Empty field code	=FFh.	E		
F	Length empty field	=02	F		
	Data field with up to 64 KB				

Structure of FETCH frame (PDU)

The meaning and values of parameters that are not assigned fixed values can be found in the section "Parameter values".

FETCH request frame			FETCH response frame		
0	System ID	="S"	0	System ID	="S"
1		="5"	1		="5"
2	Length header	=0x10	2	Length header	=0x10
3	ID OP code	=0x01	3	ID OP code	=0x01
4	Length OP code	=0x03	4	Length OP code	=0x03
5	OP code	=0x05	5	OP code	=0x06
6	ORG field code	=0x03	6	Ack field	=0x0F
7	Length ORG field	=0x08	7	Length ack field	=0x03

8	ORG ID		8	Error field	=No.
9	DBNR		9	Empty field code	=0xFF
A	Start address	High byte	A	Length empty field	=0x07
B		Low byte	B	free	
C	Length	High byte	C		
D		Low byte	D		
I	Empty field	=0xFF	I		
F	Length empty field	=0x02	F		
				Data up to 64 KB, but only if error no. = 0.	

Parameter values

S7 operand area	DB	M	I	A
ORG ID	01_H	02_H	03_H	04_H
	Source/dest. data from/to data block in main memory	Source/dest. data from/to bit memory area	Source/dest. data from/to process image input (PII)	Source/dest. data from/to process image output (PIQ)
DBNR	DB from which the source data is taken or to which the dest. data is transferred.	irrelevant	irrelevant	irrelevant
Permitted range	1...255			
Start address	DW number from which the data is taken or written to.	Memory byte no. from which the data is taken or written to	Input byte no. from which the data is taken or written to.	Output byte no. from which the data is taken or written to.
Permitted range	0...2047	All memory bytes made available by a CPU.	0...127	0...127
Length	Length of the source/ dest. data field in words	Length of the source/ dest. data field in bytes	Length of the source/ dest. data field in bytes	Length of the source/ dest. data field in bytes
Permitted range	Up to 8192 bytes	Up to 8192 bytes	1...128	1...128

S7 operand area	PI/PQ	C	T
ORG ID	05_H	06_H	07_H
	Source/dest. data from/to I/O modules With source data input modules With destination data output modules	Source/dest data from/to counter cells	Source/dest data from/to timer cells
DBNR	irrelevant	irrelevant	irrelevant
Start address	I/O byte no. from which the data is taken or written to.	Number of the counter cell from which the data is taken or written to	Number of the timer cell from which the data is taken or written to.

Permitted range	0...127: Digital I/O 128...255: Analog I/O	0...255	0...255
Quantity	Length of the source/dest. data field in bytes	Length of the source/dest. data field in words (counter cell = 1 word)	Length of the source/dest. data field in words (counter cell = 1 word)
Permitted range	1...256	1	1

See also

Use of FETCH/WRITE (Page 3090)

1.6.1.3 CP PROFIBUS (S7-300, S7-400, PC)

Time-of-day synchronization (S7-300, S7-400, PC)

Reference

Configuring the time-of-day synchronization of PROFIBUS CPs.

Time-of-day synchronization

Decide whether or not the CP will forward time-of-day frames.

Depending on where the time master is located, the following two situations must be distinguished:

- Time-of-day frames come from the subnet and are forwarded to the station.
- Time-of-day frames come from the station and are forwarded to the subnet (LAN).

With some modules, specify the direction in which the time-of-day frames are forwarded. With modules for which forwarding can only be globally enabled or disabled, the CP can forward time-of-day frames in any direction.

If you have several CPs in your station, take into account the flow of time-of-day frames depending on the time-of-day master because you can forward time-of-day frames from one network to another network using the function described here making conflict situations possible.

If there is more than one CP in a station connected to the same network, only a single CP is allowed to forward time-of-day synchronization messages.

Field device parameter settings (data record routing) (S7-300, S7-400, PC)

Reference

Setting the properties for PROFIBUS CPs in the "Properties > Settings > Field device parameter settings" parameter group.

Field device parameter settings (data record routing)

if you select this option, you can use the CP as a router for data records intended for field devices (DP slaves). Data records transferred by devices that are not connected directly to PROFIBUS and that therefore have no direct access to the field devices (DP slaves) can then be forwarded by the CP to the field devices.

SIMATIC PDM (Process Device Manager) is a tool that creates data records of this type for assigning parameters to field devices.

As default, this function is enabled. Since the function requires additional memory resources, you can turn off the option if you are putting a high load on the memory resources of the CP as a whole (connections etc.) and do not require the "Data record routing" function.

CPs with DP functionality - Selecting the mode (S7-300, S7-400, PC)

Reference

Setting the CP mode for PROFIBUS DP in the "Properties > General > Operating mode" parameter group.

Relationship between CP mode and CP functionality

Refer to this table for the configurable CP modes and resulting CP functionality:

CP mode	Possible CP functions				
	PG/OCM via PROFIBUS	FDL via PROFIBUS	DP master	DP slave	Configured S7 connections
No DP mode	X	X	-	-	X
DP master mode	X	X	X	-	X
DP slave active (see below)	X	X	-	X	-
DP slave passive (see below)	-	-	-	X	-

Key: "X" = Function supported; "-" = Function not supported

DP master mode / "DP delay time" parameter

Here, you can set an additional delay for the updating of the DP I/O data. This delay is used by the PROFIBUS CP for processing other protocols, for example FDL connections.

The displayed DP reaction times are only updated when you take one of the following actions:

- Click the "Recalculate..." button or
- Select a different parameter group

If a lot of DP slaves are configured, this recalculation can take several seconds.

DP master class 2 (diagnostic functions - can only be selected with certain CP types)

In this mode, the device can read input and output data from any DP slaves not assigned to it via the PROFIBUS CP.

This means, for example, that a process signal can be acquired by several DP masters and can help to save sensors in the field (shared input / shared output). Programming, diagnostic or management devices typically operate as DP class 2 masters.

DP slave mode / "Test, commissioning, routing" option (the name of the function can vary depending on the CP type)

- Option enabled: DP slave active
  Select the check box if, for example, you want to use PG functions regularly over this interface for commissioning and testing. This also selects the CP mode "DP slave active". (In other words, the interface is involved in token passing on PROFIBUS.)
  The following functions are then possible (the name of the function can vary depending on the CP type):
  - Programming (for example loading)
  - Testing (status/modify)
  - S7 routing (I slave as gateway)
  - S5-compatible communication (SEND/RECEIVE interface)
  - S7 communication (only in server mode on connections configured at one end without communication blocks, configuration on partner)

  Note

  Special effects of the "DP slave active" mode
  - S7 connections for data exchange using communication blocks cannot be configured in this mode
  - This can extend the token rotation time. In time-critical applications and when S7 routing and client functionality are not required for communication, this option should therefore not be enabled.

- Option disabled: DP slave passive
  If the "Test, commissioning, routing" box is not checked, the "DP slave passive" CP mode is selected and the interface only operates as a server for communication services and is not involved in the token rotation on PROFIBUS.
  Server functionality means that only the partner (for example PG, OP or automation system) can initiate communication. Note the following features when the option is not enabled:
  - S7 routing is not possible.
  - Programming and testing are possible but slow (no extension of the bus token rotation time).

CP supports several DP slave types (S7-300, S7-400, PC)

Reference

Setting the DP slave type for PROFIBUS DP in the "Properties > General > Operating mode" parameter group.

DP mode

On CPs that support different DP slave types, you can select the following DP modes depending on the CP type:

- DPV1 functionality (default mode)
  This includes DP slaves complying with the PROFIBUS DPV0 and DPV1 standard as well as Siemens DP slaves.
  This mode is only possible with certain CPU types. Refer to the information in the S7-CP manual for your PROFIBUS CP.
- DPV0 compatible
  This includes DP slaves complying with the PROFIBUS DPV0 standard; DP slaves complying with the DPV1 standard can only be used with restricted functionality.
- S7-compatible
  This includes DP slaves complying with the PROFIBUS DPV0 standard and Siemens DP slaves; DP slaves complying with the DPV1 standard can only be used with restricted functionality.

1.6.1.4 CM/CP PROFIBUS (S7-1500)

Configuring time-of-day synchronization (S7-1500)

Time-of-day synchronization roles of the communications module

With the time-of-day synchronization parameters, you specify the role of the communications module in the forwarding of time-of-day frames.

Depending on the mode of the module on PROFIBUS, the following synchronization roles can be set:

- The communications module is passive node on PROFIBUS (for example DP slave):
  - Slave
  - None
- The communications module is an active node on PROFIBUS:
  - Master
  - Slave
  - None

Meaning of the parameters

The parameters of the time-of-day synchronization have the following meaning.

- Type of synchronization
  The parameter specifies the synchronization role:
  - Slave
    The CM clock is synchronized with another clock. The synchronization frames come from PROFIBUS. With this type of synchronization, the CM forwards the time-of-day frames to the station at a fixed interval.
    Within the S7 station, there may be only one time-of-day slave.
  - Master
    The clerk of the communications module as master sends time-of-day frames to the PROFIBUS network.
    If the synchronization type "Master" is set, the interval for outputting time-of-day frames can be selected.
  - None
    There is no synchronization.

  If more than one module with a clock is present within a network, you need to specify which module will act as the time master and which as the time slave for time-of-day synchronization purposes. There can only be one time-of-day master.
- Update cycle
  With the synchronization role "Master", the parameter defines the update interval at which the time-of-day frames are output to the network.

1.6.1.5 CPs settings that do not depend on the protocol or the network

Replacing a module without a PG

Reference

Making special settings in the "Properties > General > Settings" parameter group

Depending on the CP type, you will see the option "Module replacement without PG"

Effect - nonvolatile, long-term storage on the CPU

If you have selected this option, your configuration data is stored long-term in nonvolatile memory on the CPU instead of on the EEPROM of the CP. Remember, however, that even on the CPU such data is only retained long-term in the event of a failure if the CPU has a backup battery or a SIMATIC memory card is used.

Resource requirements on the CPU

If you select this option, you occupy additional resources on your CPU. When you load the user programs and the configuration data, you will be informed if there is not enough storage space available. Resource bottlenecks can be avoided by using a SIMATIC memory card.

Note

If you are faced with a lack of resources and do not want to use a SIMATIC memory card, you can also deselect this option and store the configuration data on the CP. Later, you can then create a SIMATIC memory card with the configuration data so that the "Module replacement without PG" option is enabled. If you then insert the SIMATIC memory card in the CPU, you can replace the CP at any time. The configuration data is then loaded from the CPU or from the SIMATIC memory card automatically when the CP starts up.

Multiplexing OP connections / occupying internal CPU connection resources (S7-300)

Reference

Making special settings in the "Properties > General > Settings" parameter group

Depending on the CP type, you will see the option "Multiplex OP connections"

Multiplex mode

To connect TD/OPs and/or HMI devices, you can optimize the connection resources in the S7-300 CPU by allowing up to 16 such devices to communicate on a single CPU connection resource (multiplex mode).

If you do not use multiplex mode, the number of TD/OPs and/or HMI devices that can be used will depend on the number of available connection resources of the CPU used.

Use of internal CPU connection resources

Configured S7 connections via the CP use the same multiplex channel that you use in multiplex mode for HMI connections. This means that when you configure S7 connections, you already occupy one CPU connection resource.

Note

PG connections are not operated via the multiplexer; one connection resource is always occupied when you operate a PG.

Note

Addressing in multiplex mode

In multiplex mode, the rack/slot assignment of the CP must be specified instead of the rack/slot assignment of the CPU when assigning addresses for TD/OP/HMI connections.

Applications (for example ProAgent) that require block-related alarms (Alarm_S: SFC 17, SFC 18, SFC 19) are not supported in multiplex mode.

1.6.1.6 Background on connections

Window Time and Inactivity Time - relation

Relevance

ISO transport connections configuration in the parameter group "Properties > General > Dynamics".

Avoid connection abort with appropriate update

The partner station replies to keepalives with a frame. These are sent to the partner station at the intervals specified by the Window Time.

To avoid unwanted connection aborts, the Inactivity Time should be at least three times longer than the Window Time.

"Immediate response when interrupted connection is detected" option

Meaning

If you set the "Immediate response when interrupted connection is detected" option, this means that if the connection is not available, the application immediately receives a code indicating that a communication job (FETCH or WRITE) cannot be executed. The code is sent regardless of whether or not a current attempt to establish the connections succeeds.

If you do not select this option, the application is informed of the nonavailability of a connection only when the connection times out.

The option described here is relevant and available only when the two modes listed below are used:

- The connection is configured as a permanent connection.
  Connections can be configured as permanently established connections (option in the "Details of connection establishment" tab). This means that jobs can be forwarded to the partner with the shortest possible reaction time without needing to establish the connection first.
  If there is a disruption on the connection, the system automatically attempts to re-establish the connection. In this situation, there may be delays before a message can be forwarded to the application.
- You are using the FETCH or WRITE job types (can be selected in the "Options" tab).

An immediate reaction is necessary only in certain situations with these job types if there is no connection.

Address information - E-mail connection

Parameters for E-mail delivery

In "Properties > Address details", you enter the parameters required for the delivery of the E-mail. You can set the following parameters:

Parameter	Description	Example
E-mail server (SMTP)	Address of the mail server via which the E-mails are sent. The IP address can be entered as an absolute address or symbolic name. The following applies to symbolic names: A valid name has 1 to 64 characters of which at least one character must be a letter. If one or more "." characters are used, the following constraint also applies: A valid name must have 1 to 63 further characters after the last "."; here only the characters [a-z], [A-Z], [0-9] or [-] are permitted and at least one of these characters must be a letter. A symbolic name can only be used if the Internet CP knows the address of the Domain Name System (DNS). A suitable entry is required in the configuration of the Internet CP in HW Config. For more detailed information, refer to the HW Config online help. A maximum of 64 characters can be entered.	absolute: 140.80.0.4 symbolic name: t-online.de
Default sender name	Specifies an address that is always included in the E-mail as the sender address if the sender information (FROM parameter) is empty in the header of the E-mail. This information is normally irrelevant for delivery of the mail. It is simply information for the mail recipient. Since some mail servers do not forward jobs if there is no sender information, **it is advisable to enter information here!** A maximum of 126 characters can be entered. **Note:** Remember that a default sender name must be specified if you want a test mail to be initiated by diagnostics.	Station2.CPU412@xy.factory2.de

Symbols in offline status

Symbols in the offline status

The offline status of a connection is shown as follows:

Symbol	Meaning / connection status
	Consistent - connection configuration can be loaded
	Inconsistent - connection configuration cannot be loaded

ISO-on-TCP connection: TSAP

TSAP format

ISO-on-TCP connections have a TSAP length of 1 to 16 bytes. When you are entering values, the current length is displayed automatically. You can enter local TSAPs and partner TSAPs as a hexadecimal value or as an ASCII string. If you opt for ASCII, the characters you enter are also displayed in hexadecimal format. If you enter them in hexadecimal, any printable characters are displayed as ASCII values (8 hexadecimal characters are visible). If you enter non-printing characters, the ASCII display is disabled (ASCII entry is no longer possible), and the non-printing characters appear as dots.

Note

TSAP length

Do not use length 1 or 2 TSAPs, as these TSAPs are already used for internal purposes.

Local TSAPs and partner TSAPs

Partner and local TSAPs may be the same, since the connection is unique due to the different IP addresses. If you have to set up more than one connection between two devices, then the TSAPs must also be different.

Default TSAPs

For configuring the local and partner TSAPs, there is an "ISO-on-TCP-1" default value for the first connection between two partners (can be changed). The default value "ISO-on-TCP-2" is then suggested for a new connection between the same partners. ISO-on-TCP-1 is then suggested for a new connection with a new partner.

Connection setup for FETCH/WRITE (S7-300, S7-400, PC)

Relevance

ISO transport connections / ISO-on-TCP connections / TCP connections configuration in the parameter group "Properties > General > Options".

How connection establishment affects FETCH/WRITE

Please note the following points for FETCH/WRITE mode:

- For S7 stations:
  If you want to use FETCH/WRITE mode (see the "Options" parameter group), clear the "Active connection establishment" option.
- For PC stations / OPC servers:
  If you want to use FETCH/WRITE mode (see the "Options" parameter group), enable the "Active connection establishment" option.

S7 connection

OPC connection parameters for time monitoring

Connection timeout

If there is no reply to a request for a connection from a communication partner within the time set here, OPC jobs are cancelled with an error message. Values from 2,000 to 100,000 ms can be set.

Since successful establishment of a connection can take some time depending on the network structure, the connection establishment timeout should be longer than the job timeout.

The connection timeout starts with the start of the request for the connection. This time depends on the configuration of the connection characteristics (see above). Two cases are possible:

- **Keeping the connection established permanently:** In this case, the connection establishment request begins with the start of the OPC server for the S7 protocol. This start is triggered by connecting an OPC client to the OPC server and searching the S7 namespace and/or by adding and actively monitoring an S7 item.
- **Connection when required:** The request for connection establishment starts with the first access to tags of this S7 connection.

If the S7 connection is interrupted in running operation, an attempt is made to reestablish the connection. This resets and restarts the connection timeout.

Job timeout

Network jobs directed to the partner device are monitored by the job timeout. If no response is issued to a network job within the job timeout, the corresponding OPC jobs are cancelled with an error message. Values from 1 to 100,000 ms can be set.

The job timeout starts at the time a network job is sent in the OPC server. In general, there is only an indirect relationship between write/read/monitor jobs at the OPC interface. Several OPC items can be handled for each network job. Additionally, several OPC clients can request the same OPC item for each network job.

Furthermore, the "Optimize write access" and "Optimize read access" parameters play a role because several OPC jobs can be assembled there on the network level.

OPC connection parameters for the connection state

Immediate response when interrupted connection is detected

Can only be enabled if the "Maintain connection permanently" option is selected. The immediate reporting of S7 connections that have not been established with OPC jobs, means that you avoid repeated job timeouts.

Functional description: An interrupted connection is detected and the connection stored. If the connection has not yet been established or it is currently being attempted to re-establish an uninterrupted connection, OPC jobs are replied to immediately with an error message to this effect. The error message is sent without waiting for the job or connection timeout.

Optimizing write access, optimizing read access

Principle

When accessing variables over S7 connections, the OPC Server for SIMATIC NET provides an optimization algorithm to increase the data transmission speed.

Several simultaneous jobs to access individual variables are converted internally to create a single access to an array on the communication partner. This reduces the number of data packets transported over the network. Utilization of single packets is improved and the payload of a packet is increased.

This optimization is used for both read and write access and is enabled as the default setting. For the OPC client, for example an OPC-compliant HMI device, this optimization algorithm is invisible.

Further information

Additional details are contained in the documentation for the OPC server:

- Industrial communication with the PG/PC Volume 1 in contribution ID: 42783968 (http://support.automation.siemens.com/WW/view/en/42783968)
- Industrial communication with the PG/PC Volume 2 in contribution ID: 42783660 (http://support.automation.siemens.com/WW/view/en/42783660)

One-way S7 connection option

Meaning

If the "One-way" check box is selected, only one-way communication blocks can be used on this connection ("PUT" and "GET" instructions) in the user program of an S7 CPU. One-way means that the connection partner functions as a server on this connection and cannot send or receive actively.

Communications instructions used at both ends (with "pairs of blocks") cannot be used on connections configured at one end; for example USEND and URCV, BSEND and BRCV or USTATUS.

Since communication options are generally clearly defined based on the connection end points, the check box is preset and it may not be possible to change it.

Example 1:

The check box is grayed out (cannot be changed) and enabled in the following configuration:

- The local connection end point is an S7-400 CPU or an S7-300 CPU with loadable S7 communication (via CP) and the partner connection end point is an S7-300 CPU without loadable S7 communication.

Example 2:

The check box is grayed out (cannot be changed) and not enabled in the following configuration:

- The local connection end point and the partner connection end point are S7-400 CPUs. In this case, STEP 7 automatically creates a two-way connection. An ID (local ID and partner ID) is entered for both the local end point and the end point of the connection partner.

OPC server - alarms

Block-related alarms and diagnostic alarms

The options act as filters so that you can set the specific type of alarm that can arrive via the S7 connection.

The communication partner generates block-related alarms with the Alarm_8, Alarm_S or Alarm_SQ blocks.

Block-related alarms can be signaled

- As conditional events
- As simple events (simple events for compatibility with older applications that do not generally expect conditional events).

If you select "Block-related alarms", the OPC server then registers for alarm messages on this connection. The received alarms are transferred via the Alarms&Events OPC interface.

Diagnostic alarms are derived from the system status list (SSL) of the S7 device. Diagnostic events that have not yet been reported to the OPC client are reported if it has registered for

diagnostic alarms. Diagnostic events have no status and do not therefore need to be acknowledged.

Functions to set the priority for single alarm messages and diagnostics alarms can be reached via the area navigation.

Further information

Additional details are contained in the documentation for the OPC server:

- Industrial communication with the PG/PC Volume 1 in contribution ID: 42783968 (http://support.automation.siemens.com/WW/view/en/42783968)
- Industrial communication with the PG/PC Volume 2 in contribution ID: 42783660 (http://support.automation.siemens.com/WW/view/en/42783660)

1.6.1.7 IE/PB Link PN IO

PROFINET device number

Reference

Display and setting of the properties for PROFINET IO devices on the DP master system of an IE/PB LINK PN IO in the "Properties > General > PROFINET device number" parameter group.

Parameters

To allow you to recognize the assignment better, STEP 7 attempts to assign the same digits for the device number and the PROFIBUS address when the DP slave is placed in the DP master system. As default, the "Use PROFIBUS address as device number" option is therefore active as long as no device number has yet been assigned that corresponds to the current PROFIBUS address.

Note

Unique device numbers

Remember that the device numbers assigned in a PROFINET IO system must be unique. The IO devices (DP slaves) connected to the DP master system of the IE/PB LINK PN IO must also be included.

- **PROFIBUS address**
  Display of the PROFIBUS addresses of the DP slaves connected to the DP master system.
- **Name**
  Display of the device names of the DP slaves connected to the DP master system. Cannot be changed here.

- **PROFINET device number**
  Display or selection of the PROFINET device numbers of the DP slaves connected to the DP master system. As default, the PROFINET device number is adopted from the PROFIBUS address.
  If the option "Use PB address as device number" is deactivated, the PROFINET device number can be changed. The range of values depends on the IO controller being used.
  If you activate the "Use PROFIBUS address as device number" option again after specifying the device number, the value will be reset accordingly.
- **Use PB address as device number**
  If the option is deactivated, the PROFINET device number can be changed.
- **Prioritized startup**
  Speeds up the startup of the PROFINET IO device (here of the DP slave as PROFINET IO device). The option can only be selected if the controller supports prioritized startup.
  You can select the option for a maximum of eight DP slaves connected to the DP master system.

Modes of the IE/PB LINK PN IO

Configuration and display of the operating mode

Configuration of the mode of the IE/PB LINK PN IO gateway and display of the resulting mode for PROFIBUS DP.

- Setting the mode:
  "Properties > General > Operating mode" parameter group
- Display of the operating mode for PROFIBUS DP:
  "Properties > General > DP/MPI interface > Operating mode" parameter group

Operating modes of the gateway

You can use the IE/PB LINK PN IO gateway in the following modes:

- **Network gateway as PROFINET IO proxy**
  From the perspective of the PROFINET IO controller on Industrial Ethernet, there is no difference in the access to PROFINET IO devices connected to Industrial Ethernet and to PROFIBUS DP slaves connected to PROFIBUS DP.
  The IE/PB LINK PN IO adopts the role of a proxy for the DP slaves connected to PROFIBUS DP (DP mode "DP master" is active).
- **Network gateway in standard mode**
  In this mode, you have the following connection options
  - PG/OP communication
    PG/OP communication is used to download programs and configuration data, to execute test and diagnostics functions and for operator control and monitoring (HMI systems) of a plant.
  - Field device parameter settings (data record routing)
    You can use the IE/PB LINK PN IO as a router for data records intended for field devices (DP slaves). This allows devices that are not directly attached to PROFIBUS and that therefore do not have direct access to the field devices (DP slaves) to transfer data records to the field devices via the IE/PB LINK PN IO.
    One example of a tool that creates data records of this type for assigning parameters to field devices is SIMATIC PDM (Process Device Manager).
  - Gateway to a DP master system with constant bus cycle time
    The IE/PB LINK PN IO serves as a gateway between Ethernet and the field devices of a DP master system. The IE/PB LINK PN IO is operated as an active node along with a DP master on PROFIBUS with the parameter setting for a constant bus cycle time.
  - Cross-subnet S7 connections for HMI operation
    The IE/PB LINK PN IO forwards communication via S7 connections.
    This service is used, for example, by HMI applications (PC stations).

S7 routing via IE/PB LINK PN IO V2.1 (6GK1411-5AB00)

Routing via the IE/PB LINK PN IO is not possible in the following situations:

- S7 routing between two S7-1500 CPUs
- S7 routing of PG connections to S7-1200/1500 CPUs
- S7 routing of HMI connections to S7-1200/1500 CPUs

This behavior affects the IE/PB LINK with firmware version V2.1 (6GK1 411-5AB00).

1.6.2 Identification systems

1.6.2.1 RFID systems

Blocks

SIMATIC Ident

You will find detailed information on the Ident profile in the manual "Ident Profile and Ident Blocks, Standard Function for Ident Systems" on the pages of "Industry Online Support (https://support.industry.siemens.com/cs/ww/en/ps/14971/man)".

Programming Ident blocks

Basic blocks

Read

The "Read" block reads the user data from the transponder and enters this in the "IDENT_DATA" buffer. The physical address and the length of the data are transferred using the "ADDR_TAG" and "LEN_DATA" parameters. With the RF61xR/RF68xR readers, the block reads the data from memory bank 3 (USER area). Specific access to a certain transponder takes place with the optional "EPCID_UID" and "LEN_ID" parameters.

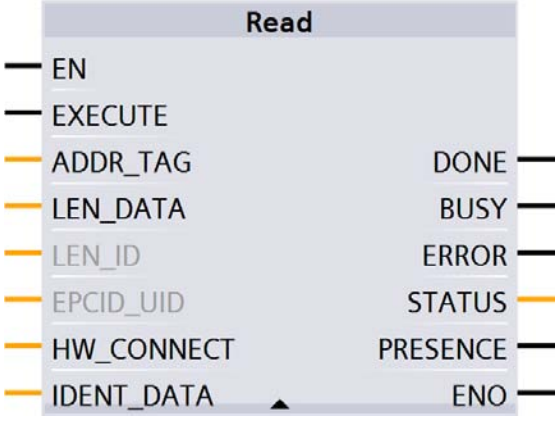

Figure 1-101 "Read" block

Table 1-160 Explanation of the "Read" block

Parameter	Data type	Default values	Description
ADDR_TAG	DWord	DW#16#0	Physical address on the transponder where the read starts. You will find more information on addressing in the section "Transponder addressing of the Ident profile and Ident blocks, standard function for Ident systems manual (https://support.industry.siemens.com/cs/ww/en/ps/14971/man)". With MV: The length of the read code is located starting at address "0" (2 bytes). The read code is located starting at address "2". [1]
LEN_DATA	Word	W#16#0	Length of the data to be read
LEN_ID	Byte	B#16#0	Length of the EPC-ID/UID Default value: 0x00 ≙ unspecified single tag access (RF200, RF300, RF61xR, RF68xR)
EPCID_UID	Array[1...62] of Byte	0	Buffer for up to 62 bytes EPC-ID, 8 bytes UID or 4 bytes handle ID. • 2 - 62-byte EPC-ID is entered at the start of the buffer (length is set by "LEN_ID") • 8-byte UID is entered at the start of the buffer ("LEN_ID = 8") • 4-byte handle ID must be entered in the array element [5]-[8] ("LEN_ID = 8") Default value: 0x00 ≙ unspecified single tag access (RF620R, RF630R)
IDENT_DATA	Any / Variant	0	Data buffer in which the read data is stored. Note: For Variant, currently only an "Array_of_Byte" with a variable length can be created. For Any, other data types/UDTs can also be created.

[1] You can find additional information on working with optical reader systems in the operating instructions for "SIMATIC MV420 / SIMATIC MV440" and "SIMATIC MV500".

Read_MV

The "Read_MV" block reads out the read result of an optical reader. The "Read" block must be used to read the configuration. The length of the data to be read is calculated automatically by the block based on the length of the created receive buffer. The actual length of the read result is output in the "LEN_DATA" output parameter. The data will be saved in the "IDENT_DATA" data buffer. If the buffer is too small, the error message "0xE7FE0400" appears and the expected length is output at "LEN_DATA".

To achieve an optimum speed, we recommend that you adapt the length of the data type "IDENT_DATA" so that this is as close as possible to the maximum expected length of the read result (2 bytes code length + read code).

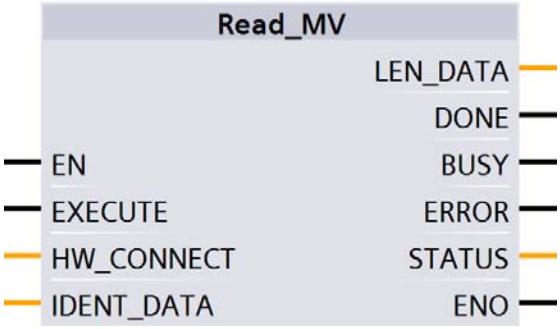

Figure 1-102 "Read_MV" block

Table 1-161 Explanation of the "Read_MV" block

Parameter	Data type	Default values	Description
IDENT_DATA	Any / Variant	0	Read result The length of the read code is located in bytes 0 and 1.
LEN_DATA	Word	W#16#0	Length of the read result ≙ 2 bytes code length + read code Note: For Variant, an "Array_of_Byte" with a variable length can be created. For Any, other data types/UDTs can also be created.

Reset_Reader

Using the "Reset_Reader" block, you can reset all reader types of the Siemens RFID systems. All the readers are reset to the settings stored in the device configuration of the RF120C or that were configured in the RF61xR/RF68xR reader using the WBM. The "Reset_Reader" block does not have any device-specific parameters and is executed using the "EXECUTE" parameter.

You can find descriptions of other Reset blocks for operation with the communications modules RF180C and ASM 456 or optical reader systems in the section "AUTOHOTSPOT".

With the "Reset_Reader" block and the other reset blocks, you can interrupt any active Ident block at any time. These blocks are then ended with ""DONE = true" and "ERROR = false"".

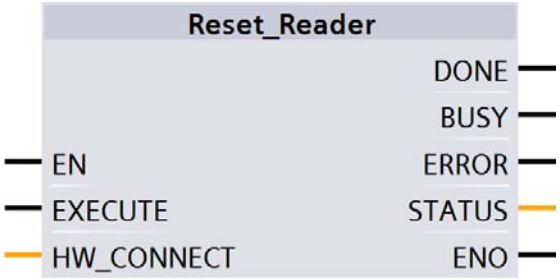

Figure 1-103 "Reset_Reader" block

Set_MV_Program

With the aid of the "Set_MV_Program" block, you can change the program in a camera. The required program number is transferred using the "PROGRAM" parameter.

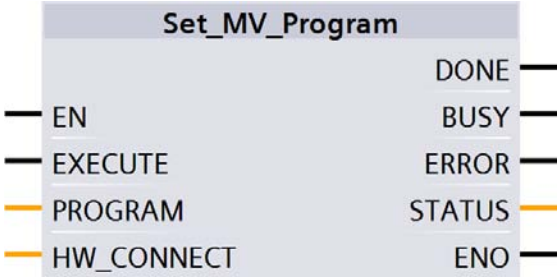

Figure 1-104 "Set_MV_Program" block

Table 1-162 Explanation of the "Set_MV_Program" block

Parameter	Data type	Default values	Description
PROGRAM	Byte	B#16#1	Program number Range of values: 0x01 ... 0x0F

Write

The "Write" block writes the user data from the "IDENT_DATA" buffer to the transponder. The physical address and the length of the data are transferred using the "ADDR_TAG" and "LEN_DATA" parameters. With the RF61xR/RF68xR readers, the block writes the data to memory bank 3 (USER area). Specific access to a certain transponder takes place with the optional "EPCID_UID" and "LEN_ID" parameters.

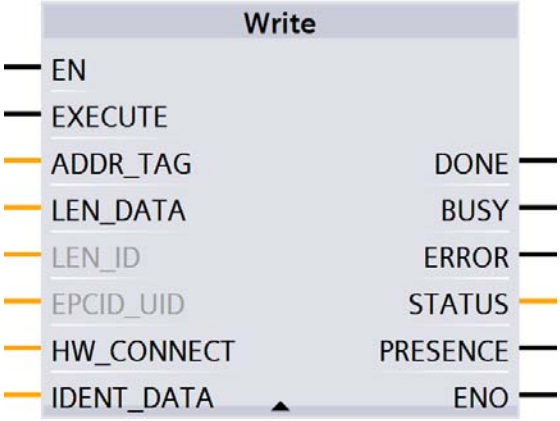

Figure 1-105 "Write" block

Table 1-163 Explanation of the "Write" block

Parameter	Data type	Default values	Description
ADDR_TAG	DWord	DW#16#0	Physical address on the transponder where the write starts. You will find more information on addressing in the section "Transponder addressing of the Ident profile and Ident blocks, standard function for Ident systems manual (https://support.industry.siemens.com/cs/ww/en/ps/14971/man)". With MV: Address is always 0. [1)]
LEN_DATA	Word	W#16#0	Length of the data to be written
LEN_ID	Byte	B#16#0	Length of the EPC-ID/UID Default value: 0x00 ≙ unspecified single tag access (RF200, RF300, RF61xR, RF68xR)

Parameter	Data type	Default values	Description
EPCID_UID	Array[1...62] of Byte	0	Buffer for up to 62 bytes EPC-ID, 8 bytes UID or 4 bytes handle ID. • 2 - 62-byte EPC-ID is entered at the start of the buffer (length is set by "LEN_ID") • 8-byte UID is entered at the start of the buffer ("LEN_ID = 8") • 4-byte handle ID must be entered in the array element [5]-[8] ("LEN_ID = 8") Default value: 0x00 ≙ unspecified single tag access (RF620R, RF630R)
IDENT_DATA	Any / Variant	0	Data buffer with the data to be written. With MV: The first byte encodes the corresponding MV command.[1)] Note: For Variant, currently only an "Array_of_Byte" with a variable length can be created. For Any, other data types/ UDTs can also be created.

[1)] You can find additional information on working with optical reader systems in the operating instructions for "SIMATIC MV420 / SIMATIC MV440" and "SIMATIC MV500".

Extended blocks

Config_Upload/-_Download

Using the "Config_Upload" and "Config_Download" blocks, you can read ("Config_Upload") or write ("Config_Download") the configuration of the RF61xR/RF68xR readers via the control program. This block enables the configurations of the readers connected to this controller to be read and written.

The configuration data is not interpretable data. Save the data on the controller so that it can be written to the reader again if a device is replaced. When multiple readers has the same configuration, you only have to save the configuration data once per reader type. A single backup can be copied to all identically configured readers during the download.

Bytes 6-9 (see table below) contain a configuration ID (CONFIG_ID) with a unique version identifier. With the configuration ID, when performing a "Config_Upload", you can check whether the configuration data read matches the configuration data stored on the controller. You can also use the device status to check the serial number to see if the device has been replaced.

The configuration data has the following structure:

Table 1-164 Structure of the configuration data

Byte	Name
0	Structure identifier (2 bytes)
2	Length information (4 bytes) Length of the version identifier and parameter block
6	Version ID (≙ CONFIG_ID; 4 bytes) Based on the identifier, you can uniquely identify the configuration. This is a time stamp in Linux format. The time stamp indicates how many seconds have passed since January 1, 1979, 00:00 (midnight). The identifier is assigned when a configuration is generated.
10 ... end "DATA"	Parameter block

"Config_Upload/Config_Download" can be executed on every channel of the RF61xR/RF68xR. It is always the same configuration data that is transferred.

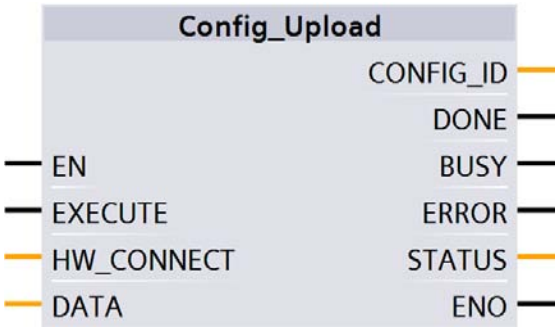

Figure 1-106 "Config_Upload" block

Table 1-165 Explanation of the "Config_Upload" block

Parameter	Data type	Description
DATA	Any / Variant	Data buffer for configuration data. The real length of the data depends on the complexity of the configuration and the firmware version of the reader. With a standard configuration of the RF61xR/RF68xR reader, we recommend a memory size of 4 KB. If you use advanced reader configurations (filtering) or want to change the configuration in the future without needing to adapt the memory size of "DATA", we recommend a memory size of 8-16 KB. Note: For Variant, currently only an "Array_of_Byte" with a variable length can be created. For Any, other data types/UDTs can also be created.
CONFIG_ID	DWord	Version identifier (4 bytes) Based on the identifier, you can uniquely identify the configuration. This is a time stamp in Linux format. The time stamp indicates how many seconds have passed since January 1, 1979, 00:00 (midnight). The identifier is assigned when a configuration is generated.

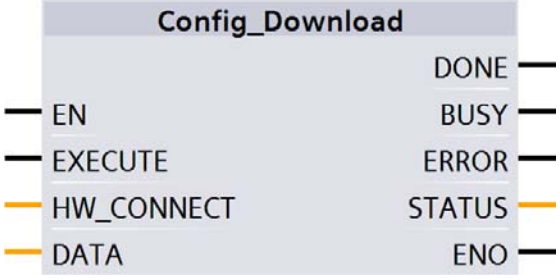

Figure 1-107 "Config_Download" block

Table 1-166 Explanation of the "Config_Download" block

Parameter	Data type	Description
DATA	Any / Variant	Data buffer for configuration data. The real length of the data depends on the complexity of the configuration and the firmware version of the reader. With a standard configuration of the RF61xR/RF68xR reader, we recommend a memory size of 4 KB. If you use advanced reader configurations (filtering) or want to change the configuration in the future without needing to adapt the memory size of "DATA", we recommend a memory size of 8-16 KB. Note: For Variant, currently only an "Array_of_Byte" with a variable length can be created. For Any, other data types/UDTs can also be created.

Inventory

The "Inventory" block activates the taking of inventories. With a single inventory, acquisition cycles of all antennas with all polarizations are queried. For example, if 2 antennas are connected to a reader and each has 3 polarizations, then an inventory will include 6 acquisition cycles.

With the RF620R and RF630R readers, inventories are always taken as soon as the antenna is turned on.

Special feature of the RF610R, RF615R, RF680R, RF685R readers

Note that the length of the data buffer ("IDENT_DATA") must correspond to at least the length of the maximum expected data. If more transponders are identified and data read out than have space in the assigned buffer length of "IDENT_DATA", the data of these transponders is lost. This reaction is indicated by the error "0xE7FE0400" (buffer overflow).

For the RF61xR/RF68xR readers, the parameters "DURATION" and "DUR_UNIT" are also available. Using the parameters, you can specify the duration of the inventories.

With the readers. there are four different modes that you can select with the "ATTRIBUTE" parameter.

- At the start, a certain duration/number (period of time, number of inventories, number of "Observed" events or identified transponders) is specified. A distinction is made between the following options:
  - Duration
    Take inventories for a specified period of time
  - Number of inventories
    Take a specified number of inventories
  - Number of "Observed" events
    Take inventories until a specified number of transponders have been identified at the same time.

  Inventories are then taken by the reader for this time or number of inventories. When the specified time/number is reached, the block is ended and returns all identified transponders in "IDENT_DATA". In other words, other commands can only be executed when all inventories have been taken completely. The unit (time or number) is specified using "DUR_UNIT" and the value (time value or number) using "DURATION". This mode can be executed using the attributes "0x80" and "0x81". Depending on the attribute, more or less data is supplied about the identified transponders.
- With the attributes "0x86" (start "Presence_Mode") and "0x87" (end "Presence_Mode"), inventories can be taken permanently. The presence of a transponder can then always be queried using "PRESENCE" without needing to start the block with "EXECUTE". No information about the identified transponders is returned when the command executes! To obtain information about the identified transponders, use one of the two calls listed above (with time / number of inventories = 0).
  When this mode is active, commands relating to transponders are not executed immediately but only when a transponder is identified. This achieves shorter reaction times since the command is already pending when the transponder enters the antenna field. "Presence_Mode" is practical in the context of the "Repeat command" function.

The "NUMBER_TAGS" output parameter is used to output the number of identified transponders. With the attributes "0x80" and "0x81" on completion of the read operation, the sum of all identified transponders is displayed. With the attribute "0x86" the number of currently identified transponders is shown at the "NUMBER_TAGS" output parameter (max. 15),without needing to start the module with "EXECUTE".

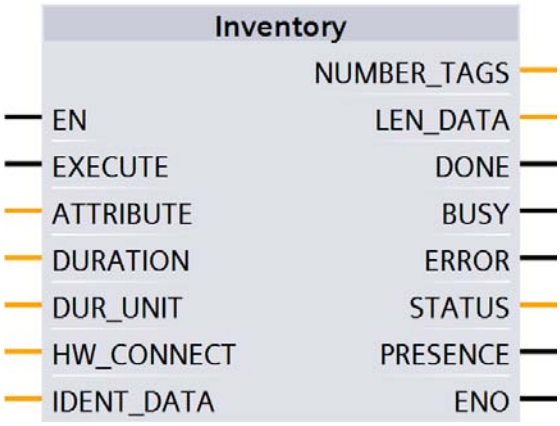

Figure 1-108 "Special" block

Table 1-167 Explanation of the "Inventory" block

Parameter	Data type	Default values	Description
ATTRIBUTE	Byte	B#16#0	Selecting the status mode: • MOBY U: 0x00 • RF620R, RF630R: 0x82 (read out next data record), 0x83, 0x85, 0x90, 0x91, 0x92 • RF610R, RF615R, RF680R, RF685R: 0x80, 0x81, 0x86, 0x87
DURATION	Word	W#16#0	RF610R, RF615R, RF680R, RF685R: Duration dependent on "DUR_UNIT" Period of time or number of inventories or number of "Observed" events Example: • 0x00 ≙ no inventory • 0x01 ≙ one inventory
DUR_UNIT	Word	W#16#0	RF610R, RF615R, RF680R, RF685R: Unit for "DURATION" • 0x00 ≙ time [ms] • 0x01 ≙ inventories • 0x02 ≙ number of transponders that achieve the "Observed" state
IDENT_DATA	Any / Variant	0	Data buffer for inventory data Note: For Variant, an "Array_of_Byte" with a variable length can be created. For Any, other data types/UDTs can also be created.
NUMBER_TAGS	Int	0	Number of transponders in the antenna field
LEN_DATA	Word	W#16#0	Length of the valid data During command processing = 0

Event display

The following data types are not displayed in the TIA Portal for a newly created project in the selection list of the data types. To use these data types, you must manually enter the name of the data type (e.g. "IID_IN_I_81"). After the first use, the data type is automatically listed in the project and also in the selection list.

Results for MOBY U

Table 1-168 ATTRIBUTE "0x00" (data type "IID_INVENT_00_MOBY_U")

Name	Type	Comment
number_MDS	WORD	Number of MDS
UID_length	WORD	length of UID
UID	ARRAY[1..12] of IID_IN_I_8BYTE	
UID[1]	IID_IN_I_8BYTE	
UID	ARRAY[1..8] of BYTE	
UID[1]	BYTE	
UID[2]	BYTE	
UID[3]	BYTE	
UID[4]	BYTE	
UID[5]	BYTE	
UID[6]	BYTE	
UID[7]	BYTE	
UID[8]	BYTE	
UID[2]	"IID_IN_I_8BYTE"	
UID[3]	"IID_IN_I_8BYTE"	
UID[4]	"IID_IN_I_8BYTE"	
UID[5]	"IID_IN_I_8BYTE"	
UID[6]	"IID_IN_I_8BYTE"	
UID[7]	"IID_IN_I_8BYTE"	
UID[8]	"IID_IN_I_8BYTE"	
UID[9]	"IID_IN_I_8BYTE"	
UID[10]	"IID_IN_I_8BYTE"	
UID[11]	"IID_IN_I_8BYTE"	
UID[12]	"IID_IN_I_8BYTE"	

Results for RF620R, RF630R

Table 1-169 ATTRIBUTE "0x83" (data type "IID_INVENT_82_83_RF600") for RF620R, RF630R with EPC-ID/UID

Name	Type	Comment
reserved0	BYTE	
number_MDS	BYTE	Number of MDS
EPC	ARRAY[1..19] of "IID_IN_I_12BYTE"	
EPC[1]	"IID_IN_I_12BYTE"	
ID	ARRAY[1..12] of BYTE	
ID[1]	BYTE	

Name	Type	Comment
ID[2]	BYTE	
ID[3]	BYTE	
ID[4]	BYTE	
ID[5]	BYTE	
ID[6]	BYTE	
ID[7]	BYTE	
ID[8]	BYTE	
ID[9]	BYTE	
ID[10]	BYT	
ID[11]	BYTE	
ID[12]	BYTE	
EPC[2]	"IID_IN_I_12BYTE"	
EPC[3]	"IID_IN_I_12BYTE"	
EPC[4]	"IID_IN_I_12BYTE"	
EPC[5]	"IID_IN_I_12BYTE"	
EPC[6]	"IID_IN_I_12BYTE"	
EPC[7]	"IID_IN_I_12BYTE"	
EPC[8]	"IID_IN_I_12BYTE"	
EPC[9]	"IID_IN_I_12BYTE"	
EPC[10]	"IID_IN_I_12BYTE"	
EPC[11]	"IID_IN_I_12BYTE"	
EPC[12]	"IID_IN_I_12BYTE"	
EPC[13]	"IID_IN_I_12BYTE"	
EPC[14]	"IID_IN_I_12BYTE"	
EPC[15]	"IID_IN_I_12BYTE"	
EPC[16]	"IID_IN_I_12BYTE"	
EPC[17]	"IID_IN_I_12BYTE"	
EPC[18]	"IID_IN_I_12BYTE"	
EPC[19]	"IID_IN_I_12BYTE"	

Note

Number of EPC-IDs

"number_MDS" specifies the number of EPC IDs (1 to 19) transferred with the "INVENTORY" block. To receive the handle IDs of all transponders located in the antenna field, it may be necessary to run the "INVENTORY" block again with ATTRIBUTE "0x82".

Table 1-170 ATTRIBUTE "0x83", "0x90", "0x91" und "0x92" (data type "IID_INVENT_8x_9x_RF6_D") for RF620R, RF630R with handle ID

Name	Type	Comment
reserved	BYTE	
number_MDS	BYTE	Number of MDS

Name	Type	Comment
UID	ARRAY[1..29] of "IID_IN_I_8BYTE"	
UID[1]	"IID_IN_I_8BYTE"	
UID	ARRAY[1..8] of BYTE	
UID[1]	BYTE	
UID[2]	BYTE	
UID[3]	BYTE	
UID[4]	BYTE	
UID[5]	BYTE	
UID[6]	BYTE	
UID[7]	BYTE	
UID[8]	BYTE	
UID[2]	"IID_IN_I_8BYTE"	
UID[3]	"IID_IN_I_8BYTE"	
UID[4]	"IID_IN_I_8BYTE"	
UID[5]	"IID_IN_I_8BYTE"	
UID[6]	"IID_IN_I_8BYTE"	
UID[7]	"IID_IN_I_8BYTE"	
UID[8]	"IID_IN_I_8BYTE"	
UID[9]	"IID_IN_I_8BYTE"	
UID[10]	"IID_IN_I_8BYTE"	
UID[11]	"IID_IN_I_8BYTE"	
UID[12]	"IID_IN_I_8BYTE"	
UID[13]	"IID_IN_I_8BYTE"	
UID[14]	"IID_IN_I_8BYTE"	
UID[15]	"IID_IN_I_8BYTE"	
UID[16]	"IID_IN_I_8BYTE"	
UID[17]	"IID_IN_I_8BYTE"	
UID[18]	"IID_IN_I_8BYTE"	
UID[19]	"IID_IN_I_8BYTE"	
UID[20]	"IID_IN_I_8BYTE"	
UID[21]	"IID_IN_I_8BYTE"	
UID[22]	"IID_IN_I_8BYTE"	
UID[23]	"IID_IN_I_8BYTE"	
UID[24]	"IID_IN_I_8BYTE"	
UID[25]	"IID_IN_I_8BYTE"	
UID[26]	"IID_IN_I_8BYTE"	
UID[27]	"IID_IN_I_8BYTE"	
UID[28]	"IID_IN_I_8BYTE"	
UID[29]	"IID_IN_I_8BYTE"	

Name	Type	Comment
reserved1	DWORD	
Data	ARRAY[1..222] of BYTE	

Note

Number of handle IDs

"number_MDS" specifies the number of handle IDs (1 to 29) transferred with the "INVENTORY" block. To receive the handle IDs of all transponders located in the antenna field, it may be necessary to run the "INVENTORY" block again with ATTRIBUTE "0x82".

Table 1-171 ATTRIBUTE "0x85" (data type "IID_INVENT_85_RF600")

Name	Type	Comment
reserved	BYTE	
number_MDS	STRUCT	Number of MDS
ID	BYTE	
ID[1]	BYTE	
Handle	ARRAY[1..8] of BYTE	
Handle[1]	BYTE	
Handle[2]	BYTE	
Handle[3]	BYTE	
Handle[4]	BYTE	
Handle[5]	BYTE	
Handle[6]	BYTE	
Handle[7]	BYTE	
Handle[8]	BYTE	
EPC	ARRAY[1..12] of BYTE	
EPC[1]	BYTE	
EPC[2]	BYTE	
EPC[3]	BYTE	
EPC[4]	BYTE	
EPC[5]	BYTE	
EPC[6]	BYTE	
EPC[7]	BYTE	
EPC[8]	BYTE	
EPC[9]	BYTE	
EPC[10]	BYTE	
EPC[11]	BYTE	
EPC[12]	BYTE	
ID[2]	"IID_IN_I_20Byte"	

Name	Type	Comment
ID[3]	"IID_IN_I_20Byte"	
ID[4]	"IID_IN_I_20Byte"	
ID[5]	"IID_IN_I_20Byte"	
ID[6]	"IID_IN_I_20Byte"	
ID[7]	"IID_IN_I_20Byte"	
ID[8]	"IID_IN_I_20Byte"	
ID[9]	"IID_IN_I_20Byte"	
ID[10]	"IID_IN_I_20Byte"	
ID[11]	"IID_IN_I_20Byte"	

Note

Number of IDs transferred

"number_MDS" specifies the number of IDs (1 to 11; handle IDs and EPC IDs) transferred with the "INVENTORY" block. To receive the IDs of all transponders located in the antenna field, it may be necessary to run the "INVENTORY" block again with ATTRIBUTE "0x82".

You will find more detailed information on the individual status modes in the manuals matching the modes "FB 45", "FB55" and "SIMATIC RF620R/RF630R".

The identifiers of the status modes correspond to the following identifiers in the other manuals:

0x82 ≙ 0x02
0x83 ≙ 0x03
0x85 ≙ 0x05
0x90 ≙ 0x10
0x91 ≙ 0x11
0x92 ≙ 0x12

Programming ATTRIBUTE "0x82"

If the number of transponders in the antenna field is unknown, repeat the "INVENTORY" block with the ATTRIBUTE = "0x82".

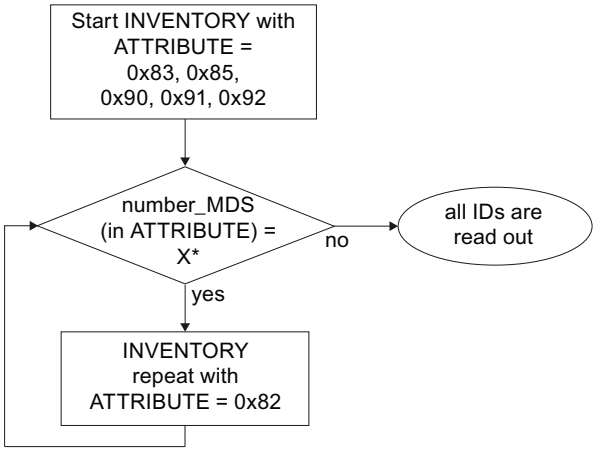

* The number of returned IDs "X" depends on the "ATTRIBUTE" used.

Figure 1-109 Program sequence of ATTRIBUTE "0x82" with unknown transponder populations

Results for RF610R, RF615R, RF680R, RF685R

The number of "TAG_DATA[x]" elements of the data types of the ATTRIBUTE "0x80" and "0x81" depends on the number of transponders to be expected. For this reason, you need to assemble the receive buffer yourself. Not the following structure when creating the receive buffer "IDENT_DATA"/data type:

- The first element "NUM_MDS" is always of the type "WORD".
- The next element "TAG_DATA" is always of the type "ARRAY". The number of transponders to be expected ("n") must be entered in the "ARRAY".

The following tables show an example of the structure of the receive buffer "IDENT_DATA"/ data type for the ATTRIBUTE "0x80" and "0x81".

Table 1-172 ATTRIBUTE "0x80"

Name	Type	Comment
NUM_MDS	WORD	Number of MDS
TAG_DATA	ARRAY[1..n]of IID_IN_I_80	Length of EPC ID
TAG_DATA[1]	IID_IN_I_80	
Reserved	BYTE	
ID_Len	BYTE	Length of EPC ID
EPC_ID	ARRAY[1..62] of BYTE	EPC-ID
tagPC	WORD	
TAG_DATA[2]	IID_IN_I_80	
...	...	
TAG_DATA[n]	IID_IN_I_80	

Table 1-173 ATTRIBUTE "0x81"

Name	Type	Comment
NUM_MDS	WORD	Number of MDS
TAG_DATA	ARRAY[1..n] of IID_IN_I_81	
TAG_DATA[1]	IID_IN_1_81	
reserved	BYTE	
ID_LEN	BYTE	EPC length
EPC_ID	ARRAY[1..62] of BYTE	EPC-ID
tagPC	WORD	
RSSI	BYTE	RSSI value
MaxRSSI	BYTE	highest RSSI value
MinRSSI	BYTE	lowest RSSI value
channel	BYTE	channel; 1..15_ESTI; 1..53:FCC
antenna	BYTE	antenna; bit coded; Bit 0 = antenna 1; Bit 1 = antenna 2; Bit 2 = antenna 3; Bit 3 = antenna 4
polarization	BYTE	polarizatuin of antenna; 0=undefined; 1=circular; 2=vertical linear; 3=horizontal
time	Time_OF_Day	S7 timestamp
power	BYTE	power in dBm
filterDataAvailable	BYTE	0=false; 1=true
Inventoried	WORD	1)
TAG_DATA[2]	IID_IN_1_81	
...	...	
TAG_DATA[n]	IID_IN_1_81	

1) Indicates how often the transponder was identified via the air interface before it changed to the "Observed" status.

Read_EPC_Mem

The "Read_EPC_Mem" block reads data starting at address 4 from the EPC memory of the RF600 transponder. Access is to bank 1 as of the start address 4. The length of the EPC memory to be read out is specified by the "LEN_DATA" parameter.

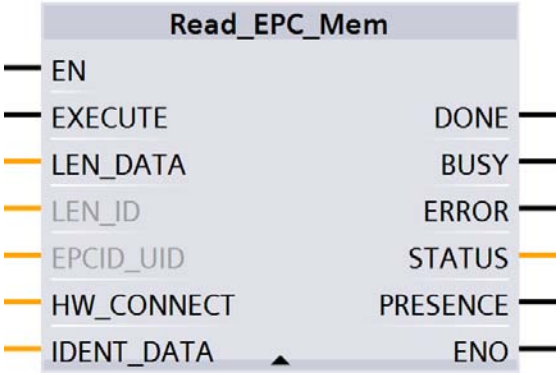

Figure 1-110 "Read_EPC_Mem" block

Table 1-174 Explanation of the "Read_EPC_Mem" block

Parameter	Data type	Default values	Description
LEN_DATA	Word	W#16#0	Length of the EPC memory to be read out (1 ... 62 bytes)
LEN_ID	Byte	B#16#0	Length of the EPC-ID/UID Default value: 0x00 ≙ unspecified single tag access (RF61xR, RF68xR)
EPCID_UID	Array[1...62] of Byte	0	Buffer for up to 62 bytes EPC-ID, 8 bytes UID or 4 bytes handle ID. • 2 - 62-byte EPC-ID is entered at the start of the buffer (length is set by "LEN_ID") • 8-byte UID is entered at the start of the buffer ("LEN_ID = 8") • 4-byte handle ID must be entered in the array element [5]-[8] ("LEN_ID = 8") Default value: 0x00 ≙ unspecified single tag access (RF620R, RF630R)
IDENT_DATA	Any / Variant	0	Data buffer in which the read EPC memory data is stored. Note: For Variant, currently only an "Array_of_Byte" with a variable length can be created. For Any, other data types/ UDTs can also be created.

Read_TID

The "Read_TID" block reads data from the TID memory area (Tag Identification Memory Bank) of the RF600 transponder. The length of the TID to be read is specified by the "LEN_DATA" parameter. The length of the TID varies depending on the transponder and can be found in the transponder data sheet.

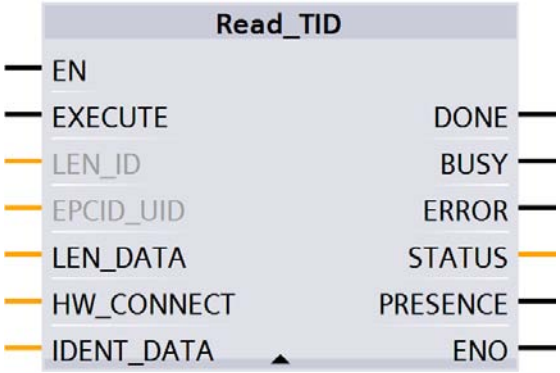

Figure 1-111 "Read_TID" block

Table 1-175 Explanation of the "Read_TID" block

Parameter	Data type	Default values	Description
LEN_ID	Byte	B#16#0	Length of the EPC-ID/UID Default value: 0x00 ≙ unspecified single tag access (RF61xR, RF68xR)
EPCID_UID	Array[1...62] of Byte	0	Buffer for up to 62 bytes EPC-ID, 8 bytes UID or 4 bytes handle ID. • 2 - 62-byte EPC-ID is entered at the start of the buffer (length is set by "LEN_ID") • 8-byte UID is entered at the start of the buffer ("LEN_ID = 8") • 4-byte handle ID must be entered in the array element [5]-[8] ("LEN_ID = 8") Default value: 0x00 ≙ unspecified single tag access (RF620R, RF630R)
LEN_DATA	Word	W#16#4	Length of the EPC memory to be read out (1 ... 62 bytes)
IDENT_DATA	Any / Variant	0	Read TID Note: For Variant, currently only an "Array_of_Byte" with a variable length can be created. For Any, other data types/ UDTs can also be created.

Read_UID

The "Read_UID" block reads the UID of an HF transponder. The UID always has a fixed length of 8 bytes.

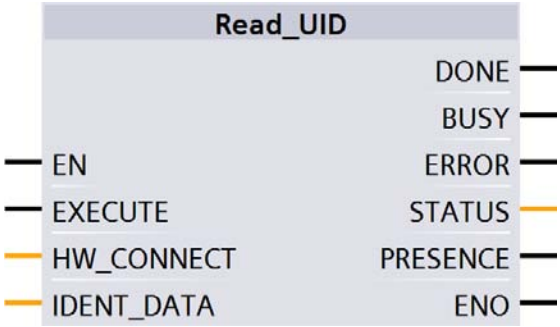

Figure 1-112 "Read_UID" block

Table 1-176 Explanation of the "Read_UID" block

Parameter	Data type	Description
IDENT_DATA	Any / Variant	UID Note: For Variant, currently only an "Array_of_Byte" with a variable length can be created. For Any, other data types/UDTs can also be created.

Set_Ant

With the aid of the "Set_Ant" block, you can turn antennas on or off. There are different blocks for RF300 and RF600. The "Set_Ant_RF300" block can also be used for RF200, MOBY D and MOBY U. The "Set_Ant_RF600" block relates only to the RF620R and RF630R readers.

Set_Ant_RF300

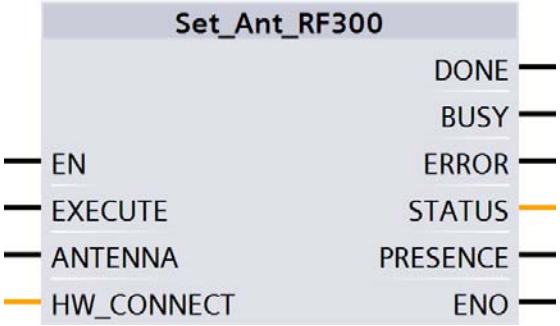

Figure 1-113 "Set_Ant_RF300" block

Table 1-177 Explanation of the "Set_Ant_RF300" block

Parameter	Data type	Description
ANTENNA	Bool	0 = turn antenna off 1 = turn antenna on

Set_Ant_RF600

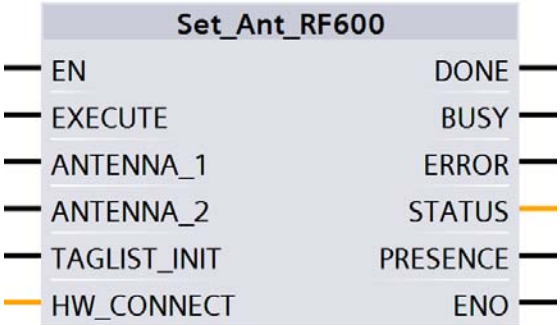

Figure 1-114 "Set_Ant_RF600" block

Table 1-178 Explanation of the "Set_Ant_RF600" block (RF620R/RF630R)

Parameter	Data type	Description
ANTENNA_1	Bool	0 = turn antenna 1 off 1 = turn antenna 1 on
ANTENNA_2	Bool	0 = turn antenna 2 off 1 = turn antenna 2 on
TAGLIST_INIT	Bool	0 = TagList is reset 1 = the existing TagList continues to be used

Set_Param

With the "Set_Param" block, you can change UHF parameters on an RF61xR/RF68xR during runtime (e.g. the antenna power).

Note

Settings saved only temporarily

Note that the parameters in the "Set_Param" block are only stored temporarily. If the power for the reader is interrupted, the stored values are lost and must be set again.

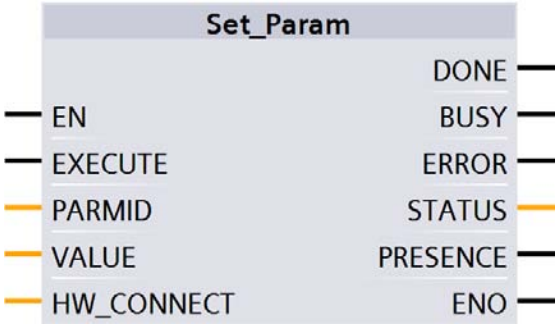

Figure 1-115 "Set_Param" block

Table 1-179 Explanation of the "Set_Param" block

Parameter	Data type	Default values	Description
PARMID	DWORD	0x00	Parameter identifier
VALUE	DWORD	0x00	Parameter value

Table 1-180 Parameter values

PARMID (hex)	PARMID (ASCII)	Parameter	VALUE
0x41315057	A1PW	Antenna 01: Radiated power	Range of values: 0.5 ... 33 Increment: 0.25 Radiated power of the antenna in [dBm]. Bytes 1 and 2 are not used, byte 3 represents the integer and byte 4 the decimal place. Example: A radiated power of 10.25 dBm represents a "VALUE" of "0x0A19".
0x41325057	A2PW	Antenna 02: Radiated power	
0x41335057	A3PW	Antenna 03: Radiated power	
0x41345057	A4PW	Antenna 04: Radiated power	
0x41315452	A1TR	Antenna 01: RSSI threshold	Range of values: 0 ... 255 Threshold value for RSSI. Transponders with lower values are discarded. Value without unit without a direct relationship with the radiated power.
0x41325452	A2TR	Antenna 02: RSSI threshold	
0x41335452	A3TR	Antenna 03: RSSI threshold	
0x41345452	A4TR	Antenna 04: RSSI threshold	
0x5331444C	S1DL	Read point 01: RSSI delta	Range of values: 0 ... 255 Difference for RSSI values. Transponders with lower values relative to the transponder with the highest RSSI value are discarded. Value without unit without a direct relationship with the radiated power.
0x5332444C	S2DL	Read point 02: RSSI delta	
0x5333444C	S3DL	Read point 03: RSSI delta	
0x5334444C	S4DL	Read point 04: RSSI delta	

PARMID (hex)	PARMID (ASCII)	Parameter	VALUE
0x4131504F	A1PO	Antenna 01: Polarization	Range of values: 0, 1, 2, 4 Polarization of the antenna (for intelligent antennas e.g. internal antenna RF685R) • 0: default, undefined • 1: circular • 2: vertical linear • 4: horizontal linear Input is bit coded. Combinations are possible (adding values).
0x4132504F	A2PO	Antenna 02: Polarization	
0x4133504F	A3PO	Antenna 03: Polarization	
0x4134504F	A4PO	Antenna 04: Polarization	
0x52364353	R6CS	Modulation scheme	Range of values: 32, 33, 34, 35, 37, 65 Modulation scheme of the read point Specification of which transponder types are identified (ISO 18000-63/-62). • 32: Tx: 40 Kbps / Rx: 40 Kbps / FM0 • 33: Tx: 40 Kbps / Rx: 62:5 Kbps / Miller4 • 34: Tx: 40 Kbps / Rx: 75 Kbps / Miller4 • 35: Tx: 80 Kbps / Rx: 62:5 Kbps / Miller4 • 37: Tx: 80 Kbps / Rx: 400 Kbps / FM0 • 65: Tx: 40 Kbps / Rx: 40 Kbps / ISO 18000-62
0x57544348	WTCH	Date and time	Range of values: 01.01.2000 00:00 a.m. ... 19.01.2038 3:14 a.m. 01.01.2000 01:00 a.m. ≙ 946684800 Date and time (UTC) Time in seconds since 01.01.1970; Setting of the internal reader clock, the date and time are set.

PARMID (hex)	PARMID (ASCII)	Parameter	VALUE
0x57544F44	WTOD	Time	Range of values: 0:00 – 23:59 p.m. S7 time (TOD, UTC) Milliseconds since midnight; Setting of the internal reader clock, only the time is changed but not the date.
0x57444154	WDAT	Date	Range of values: 01.01.2000 ... 18.01.2038 S7 date Days since 01/01/1990; Setting of the internal reader clock, whereby only the date is changed and not the time.

Write_EPC_ID

The "Write_EPC_ID" block overwrites the EPC-ID of the RF600 transponder and adapts the length of the EPC-ID in the memory of the transponder. The new EPC-ID length to be written is specified with the "LEN_ID_NEW" parameter and the previous EPC-ID is specified using the "LEN_ID" and "EPCID_UID" parameters.

Make sure that when you execute the block only one transponder is located in the antenna field. This ensures that the identification when writing the ID is unique. If there is more than one transponder in the antenna field, a negative response is returned.

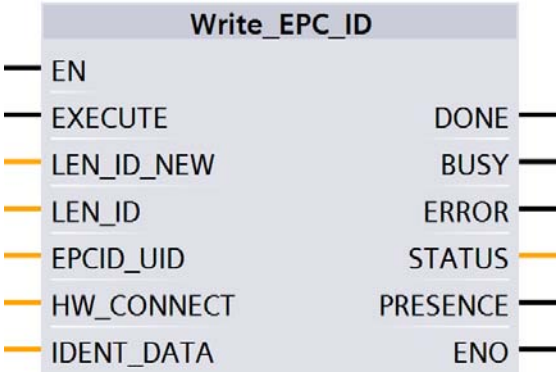

Figure 1-116 "Write_EPC_ID" block

Table 1-181 Explanation of the "Write_EPC_ID" block

Parameter	Data type	Default values	Description
LEN_ID_NEW	Byte	W#16#0C	Length of the current EPC-ID
LEN_ID	Byte	B#16#0	Length of the previous EPC-ID

Parameter	Data type	Default values	Description
EPCID_UID	Array[1...62] of Byte	0	Previous EPC ID
IDENT_DATA	Any / Variant	0	Current EPC ID Note: For Variant, currently only an "Array_of_Byte" with a variable length can be created. For Any, other data types/ UDTs can also be created.

Write_EPC_Mem

The "Write_EPC_Mem" block overwrites the EPC memory of the RF600 transponder starting at address 4. The length of the EPC memory to be overwritten is specified by the "LEN_DATA" parameter.

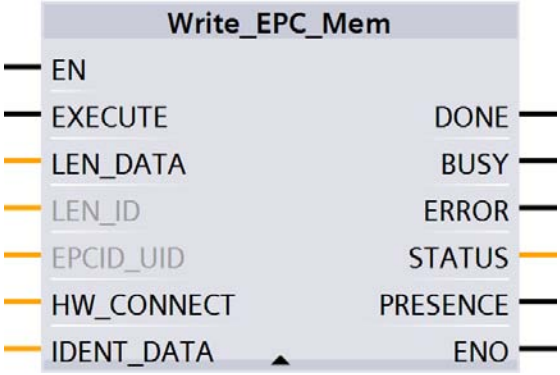

Figure 1-117 "Write_EPC_Mem" block

Table 1-182 Explanation of the "Write_EPC_Mem" block

Parameter	Data type	Default values	Description
LEN_DATA	Word	W#16#0	Length of the EPC memory to be overwritten (1 ... 62 bytes)
LEN_ID	Byte	B#16#0	Length of the EPC-ID/UID Default value: 0x00 ≙ unspecified single tag access (RF61xR, RF68xR)

Parameter	Data type	Default values	Description
EPCID_UID	Array[1...62] of Byte	0	Buffer for up to 62 bytes EPC-ID, 8 bytes UID or 4 bytes handle ID. • 2 - 62-byte EPC-ID is entered at the start of the buffer (length is set by "LEN_ID") • 8-byte UID is entered at the start of the buffer ("LEN_ID = 8") • 4-byte handle ID must be entered in the array element [5]-[8] ("LEN_ID = 8") Default value: 0x00 ≙ unspecified single tag access (RF620R, RF630R)
IDENT_DATA	Any / Variant	0	Data buffer with the EPC memory data to be overwritten. Note: For Variant, currently only an "Array_of_Byte" with a variable length can be created. For Any, other data types/ UDTs can also be created.

AdvancedCMD

With the "AdvancedCmd" block, every command can be executed including commands not represented by other blocks. This general structure can be used for all commands and is intended only for trained users.

This gives you the option of sending the command as a chained command. To allow this, the block provides a CMD buffer for 100 commands. All chained commands must be entered starting at the first position in the buffer. For every chained command, the "chained bit" must also be set in the CMD structure. The "chained bit" is not set in the last command in the chain. You will find further information on the "chained bit" in the section "Link of the manual Ident Profile and Ident Blocks, Standard Function for Ident Systems manual (https://support.industry.siemens.com/cs/ww/en/ps/14971/man)".

The entire command structure must be specified in the "CMD" input parameter. You create the structure for the "CMD" parameter in a data block.

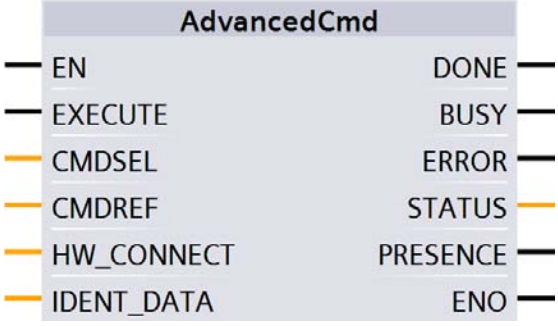

Figure 1-118 "AdvancedCmd" block

Table 1-183 Explanation of the "AdvancedCmd" block

Parameter	Data type	Default values	Description
CMDSEL	Int	1	Selection of the command to be executed "CMDREF"; 1 ⇒ 1. Command, ... The value of the "CMDSEL" parameter can never be > 100.
CMDREF	Any / Variant	--	You will find a detailed description of the parameter in the sections: • "Command overview of the manual Ident Profile and Ident Blocks, Standard Function for Ident Systems manual (https://support.industry.siemens.com/cs/ww/en/ps/14971/man)" • "Command structure of the manual Ident Profile and Ident Blocks, Standard Function for Ident Systems manual (https://support.industry.siemens.com/cs/ww/en/ps/14971/man)"
IDENT_DATA	Any / Variant	0	Buffer for data to be written or read. Note: For Variant, currently only an "Array_of_Byte" with a variable length can be created. For Any, other data types/UDTs can also be created.

Reset blocks

The reset blocks described in this section are required when you want to operate the optical reader systems MV420, MV440 or the communications modules RF180C, ASM 456 with a SIMATIC S7-1200/S7-1500 controller. As an alternative you can also use these blocks for the RF120C if you have selected the appropriate setting in the device configuration.

In the system, these reset blocks have the same function as the "Reset_Reader" block described earlier. However, with the blocks described here, you need to set reader-dependent parameters.

Remember that the default value will be used automatically if you do not select a value manually.

Reset_MOBY_D

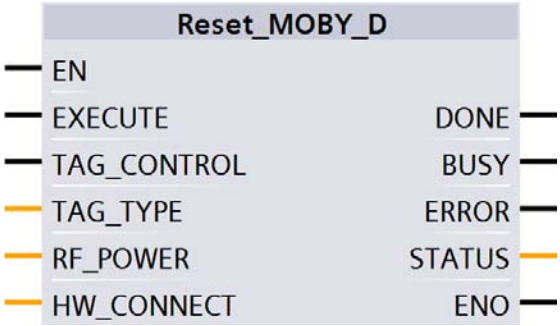

Figure 1-119 "Reset_MOBY_D" block

Table 1-184 Explanation of the "Reset_MOBY_D" block

Parameter	Data type	Default values	Description
TAG_CONTROL	Bool	True	Presence check
TAG_TYPE	Byte	1	Transponder type: • 1 = every ISO transponder
RF_POWER	Byte	0	Output power RF power from 0.5 W to 10 W in increments of 0.25 W (range of values: 0x02 ... 0x28)

Reset_MOBY_U

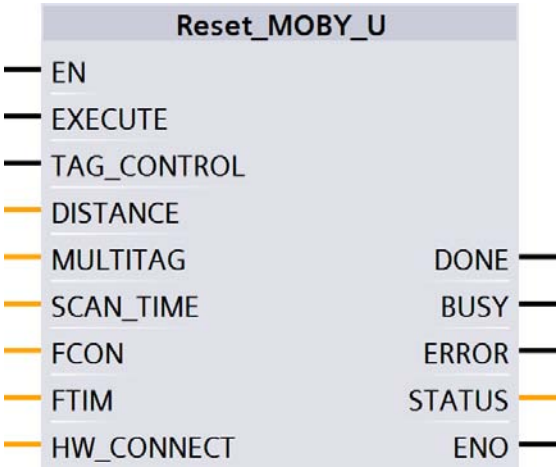

Figure 1-120 "Reset_MOBY_U" block

Table 1-185 Explanation of the "Reset_MOBY_U" block

Parameter	Data type	Default values	Description
TAG_CONTROL	Bool	True	Presence check
DISTANCE	Byte	23h	Range limitation (range of values: 0x02 ... 0x23 or 0x82 ... 0xA3 for reduced transmit power)
MULTITAG	Byte	1	Maximum number of transponders that can be processed at the same time in the antenna field. (Range of values: 0x01 ... 0x12)
SCAN_TIME	Byte	0	Scanning time: Standby time of the transponder (range of values: 0x00 ... 0xC8)
FCON	Byte	0	field_ON_control: BERO mode (range of values: 0x00 ... 0x03)
FTIM	Byte	0	field_ON_time: Time for BERO mode (range of values: 0x00 ... 0xFF)

Reset_MV

To reset cameras of the optical reader systems, call the block and activate the "EXECUTE" parameter.

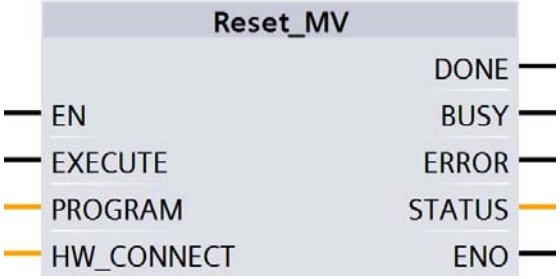

Figure 1-121 "Reset_MV" block

Table 1-186 Explanation of the "Reset_MV" block

Parameter	Data type	Description
PROGRAM	Byte	Program selection • B#16#0: Reset without program selection or in the case of diagnostics, the error code for "IN_OP = 0" is shown at the "STATUS" output parameter. • B#16#1 ... B#16#0F: Number of the program to be started ⇒ Reset with program selection (as of firmware V5.1 of the MV4x0)

Reset_RF200

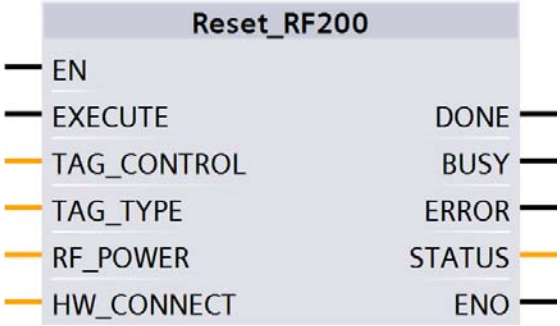

Figure 1-122 "Reset_RF200" block

Table 1-187 Explanation of the "Reset_RF200" block

Parameter	Data type	Default values	Description
TAG_CONTROL	Byte	1	Presence check
TAG_TYPE	Byte	1	Transponder type: • 1 = every ISO transponder • 3 = MDS D3xx - optimization
RF_POWER	Byte	4	Output power; only relevant for RF290R RF power from 0.5 W to 5 W in increments of 0.25 W (range of values: 0x02 ... 0x14). Default value 0x04 ≙ 1 W.

Reset_RF300

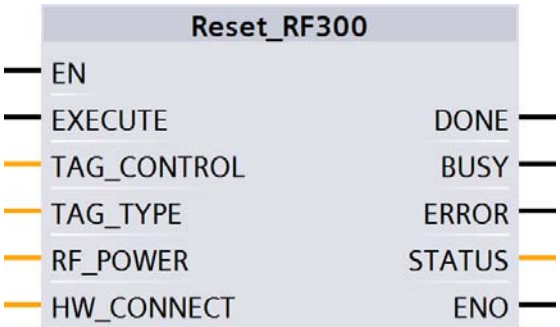

Figure 1-123 "Reset_RF300" block

Table 1-188 Explanation of the "Reset_RF300" block

Parameter	Data type	Default values	Description
TAG_CONTROL	Byte	1	Presence check • 0 = Off • 1 = on • 4 = presence (antenna is off. The antenna is turned on only when a Read or Write command is sent.)
TAG_TYPE	Byte	0	Transponder type: • 1 = every ISO transponder • 0 = RF300 transponder
RF_POWER	Byte	0	Output power; only relevant for RF380R RF power from 0.5 W to 2 W in increments of 0.25 W (range of values: 0x02 ... 0x08). Default value 0x00 ≙ 1.25 W.

Reset_RF600

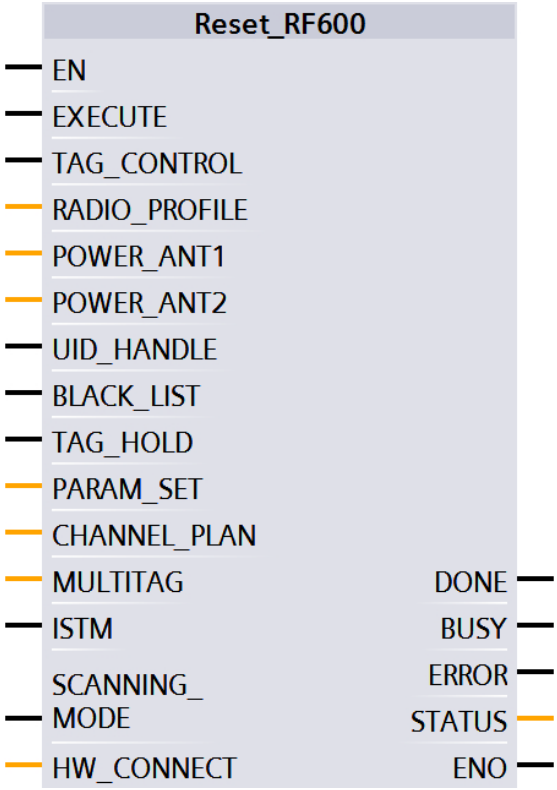

Figure 1-124 "Reset_RF600" block

Table 1-189 Explanation of the "Reset_RF600" block (RF620R/RF630R)

Parameter	Data type	Default values	Description
TAG_CONTROL	Bool	True	Presence check
RADIO_PROFILE	Byte	1	Scanning time: Wireless profile according to EPC Global (range of values: 0x01 ... 0x09 depending on the reader variant)
POWER_ANT1	Byte	0	Transmit power for antenna 1 or internal antenna (range of values: 0x00 ... 0x0F)
POWER_ANT2	Byte	0	Transmit power for antenna 2 or external antenna (range of values: 0x00 ... 0x0F)
UID_HANDLE	Bool	False	Meaning of the UID in the command: True = Handle ID, only the least significant 4 bytes of the UID are evaluated; False = UID/EPC-ID with a length of 8 bytes
BLACK_LIST	Bool	False	True = activate black list
TAG_HOLD	Bool	False	True = activate Tag Hold
PARAM_SET	Byte	0	Field_ON_Control (0 = fast; range of values: 0x00, 0x02)
CHANNEL_PLAN	Byte	0F	Field_ON_Time (range of values: 0x00 ... 0x0F; ETSI only)
MULTITAG	Byte	1	Maximum number of transponders that can be processed at the same time in the antenna field. (Range of values: 0x01 ... 0x50)
ISTM	Bool	False	True = activate intelligent single tag mode
SCANNING_MODE	Bool	False	True = activate scanning mode [1]

[1] Is not currently possible with the Ident blocks.

Reset_Univ

The "Reset_Univ" block is a universal reset block with which all identification systems can be reset. Use this block only after consulting Support.

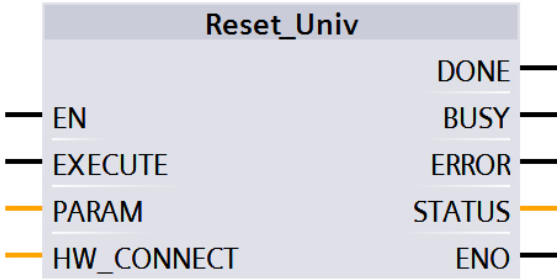

Figure 1-125 "Reset_Univ" block

Table 1-190 Explanation of the "Reset_Univ" block

Parameter	Data type	Description
PARAM	Array [1...16] of Byte	Data for reset frame The data to be set here can be made available by Support when necessary for special settings.

Table 1-191 Structure of the "PARAM" parameter

Byte	1	2...5	6	7...8	9	10	11	12	13...14	15	16
Value	04h	0	0Ah	0	scan-ning_ time	param	op-tion_ 1	dis-tance_ limiting	Num-ber of trans-pond-ers	field_ on_ control	field_ on_ time

Status blocks

Reader_Status

The "Reader_Status" block reads status information from the reader. For the various reader families, there are different status modes that you can select using the "ATTRIBUTE" parameter.

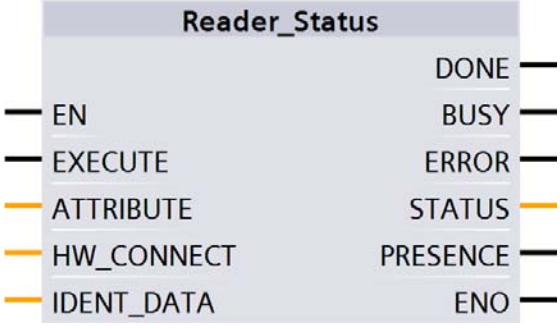

Figure 1-126 "Reader_Status" block

Table 1-192 Explanation of the "Reader_Status" block

Parameter	Data type	Default values	Description
ATTRIBUTE	Byte	B#16#81	Identifier of the status modes / possible entries: • RF200: 0x81 • RF300: 0x81, 0x86 • RF620R, RF630R: 0x87, 0x88, 0xA0, 0xA1 • RF61xR, RF68xR: 0x89 • RF185C, RF186C, RF188C: 0xA2 • MOBY U: 0x81, 0x84, 0x85 • MOBY D: 0x81
IDENT_DATA	Any / Variant	0	Event values depending on attributes Note: You create either an "Array_of_Byte" or a variable with one of the following data types. Other data types / UDTs can also be created for "Any".

Results

Apply the correct data type that is assigned to the ATTRIBUTE value at the "IDENT_DATA" input of the block so that the data can be correctly interpreted.

Note that the UDTs can only be used when the blocks "Reader_Status" or "Tag Status" used.

Table 1-193 ATTRIBUTE "0x81" (data type "IID_READSTAT_81_RF2_3_U")

Name	Type	Comment
status_info	BYTE	SLG status mode
hardware	CHAR	Type of hardware
hardware_version	WORD	Version of hardware
loader_version	WORD	Version of loader
firmware	CHAR	Type of firmware
firmware_version_HB	BYTE	Version of firmware
firmware_version_LB	BYTE	
driver	CHAR	Type of driver
driver_version	WORD	Version of driver
interface	BYTE	Type of interface (RS 232/RS 422)
baud	BYTE	Baudrate
reserved1	BYTE	Reserved
reserved2	BYTE	Reserved
reserved3	BYTE	Reserved
distance_limiting_SLG	BYTE	Distance limiting of SLG
multitag_SLG	BYTE	Multitag SLG

Name	Type	Comment
field_ON_control_SLG	BYTE	Field ON control
field_ON_time_SLG	BYTE	Field On time
sync_SLG	BYTE	Synchronization with SLG
status_ant	BYTE	Status of antenne
stand_by	BYTE	Time of standby after command
MDS_control	BYTE	Presence mode

Table 1-194 ATTRIBUTE "0x84" (data type "IID_READSTAT_84_MOBY_U")

Name	Type	Comment
status_info	BYTE	SLG status mode
number_MDS	BYTE	Range 1..24
UID	ARRAY [1..24] of DWord	

Table 1-195 ATTRIBUTE "0x86" (data type "IID_READSTAT_86_RF300")

Name	Type	Comment
status_info	BYTE	SLG status mode
FZP	BYTE	Error counter passive: distortion without communication
ABZ	BYTE	Dropout counter
CFZ	BYTE	Code error counter
SFZ	BYTE	Signature error counter
CRCFZ	BYTE	CRC-error counter
BSTAT	BYTE	Status of last command
ASMFZ	BYTE	Error counter for host interface (ASM)
reserved0	ARRAY [1..17]	

Table 1-196 ATTRIBUTE "0x87" (data type "IID_READSTAT_87_RF600")

Name	Type	Comment
status_info	BYTE	SLG status mode
hardware	CHAR	Type of hardware
hardware_version	WORD	Version of hardware
reserved0	WORD	
firmware	CHAR	Type of firmware
firmware_version_HB	BYTE	Version of firmware highbyte
firmware_version_LB	BYTE	Version of firmware lowbyte
driver	CHAR	Type of driver

Name	Type	Comment
current_time_hour	BYTE	Hours [1]
current_time_min	BYTE	Minutes
current_time_sec	BYTE	Seconds
reserved1	BYTE	
SLG_version	BYTE	SLG version
baud	BYTE	Baudrate
reserved2	BYTE	
distance_limiting_SLG	BYTE	Selected transmit power
multitag_SLG	BYTE	Multitag SLG
field_ON_control_SLG	BYTE	Selected comunication typ
field_ON_time_SLG	BYTE	Selected channel
expert_mode	BYTE	Expert mode
status_ant	BYTE	Status of antenna [2]
scanning_time_SLG	BYTE	Radio communication profile (country specific radio standart)
MDS_control	BYTE	Presence mode

[1] The internal time stamp of the reader that relates to this event is output. The internal reader time stamp is not synchronized with UTC.

[2] The antenna status relates to the "ATTRIBUTE" (bits 0 and 1) of the last executed "SET-ANT" or to the default value set by "INIT"/"WRITE-CONFIG". With "INIT"/"WRITE-CONFIG" of the RF620R, the default value is "1" (int. antenna on), with the RF630R, it is "3" (antennas 1 and 2 on).

Table 1-197 ATTRIBUTE "0x88" (data type "IID_READSTAT_88_RF600")

Name	Type	Comment
status_info	BYTE	SLG status mode(Subcommand)
hardware	CHAR	Type of hardware
hardware_version	WORD	Version of hardware
reserved_word1	WORD	Reserved
firmware	CHAR	Type of firmware
firmware_version_HB	BYTE	Version of firmware (High-Byte)
firmware_version_LB	BYTE	Version of firmware (Low-Byte)
driver	CHAR	Type of driver
current_time_hour	BYTE	Hours [1]
current_time_minute	BYTE	Minutes [1]
current_time_sec	BYTE	Seconds [1]
current_time_reservByte	BYTE	
SLG_version	BYTE	SLG-Version
baud	BYTE	Baudrate
reserved_byte1	BYTE	Reserved
distance_limiting_SLG	BYTE	Selected transmit power
multitag_SLG	BYTE	Multitag SLG
field_ON_control_SLG	BYTE	Selected communication type

Name	Type	Comment
field_ON_time_SLG	BYTE	Selected channel
expert_mode	BYTE	Expert mode
status_ant	BYTE	Status of antenna [2)]
scanning_time_SLG	BYTE	Radio communication profile (country specific radio standart)
MDS_control	BYTE	Presence mode
blink_pattern	BYTE	Blink Pattern
act_algor_Single_Tag	Bool	Single_Tag [1]
act_algor_ITF_Phase2	Bool	ITF_Phase2 [2]
act_algor_ITF_Phase1	Bool	ITF_Phase1 [3]
act_algor_Smoothing	Bool	Smoothing [4]
act_algor_Blacklist	Bool	Blacklist [5]
act_algor_RSSI_Threshold	Bool	RSSI_Threshold [6]
act_algor_Power_Ramp	Bool	Power_Ramp [7]
act_algor_Power_Gap	Bool	Power_Gap [8]
Reserved1	Bool	Reserved1 [1]
Reserved2	Bool	Reserved2 [2]
Reserved3	Bool	Reserved3 [3]
Reserved4	Bool	Reserved4 [4]
act_algor_EPC_MemBankFilter	Bool	EPC_MemBankFilteres [5]
act_algor_Tag_Holg	Bool	Tag_Hold [6]
act_algor_Multi_Tag	Bool	Multi_Tag [7]
act_algor_ISTM	Bool	ISTM [8]
reserved_word2	WORD	Reserved
reserved_word3	WORD	Reserved
reserved_word4	WORD	Reserved
filtered_max_rssi	BYTE	Maximum RSSI value of a tag, of all filtered tags
reserved_byte2	BYTE	Reserved
filtered_tags_rssi	BYTE	Number of tags, filtered out by the RSSI threshold
reserved_byte3	BYTE	Reserved
filtered_tags_black_list	WORD	Number of tags, filtered out via Black-List
filtered_tags_epc_data	WORD	Number of tags, filtered out via EPC Data Filter
filtered_tags_smoothing	WORD	Number of tags in Tag List of status Not-Observed
itf_ph1_max_detect	WORD	Number of reads of a Tag, filtered out via ITF-phase 1
itf_ph1_tags_detect	WORD	Number of tags, filtered out via ITF-phase 1
itf_ph2_max_detect	WORD	Number of reads of a Tag, filtered out via ITF-phase 2
itf_ph2_tags_detect	WORD	Number of tags, filtered out via ITF-phase 2
filtered_istm_min_dist	WORD	Minimum distance of tags according to sorting criterion of ISTM
filtered_istm_tags	WORD	Number of tags, filtered out via ISTM algorithm

Name	Type	Comment
last_error	BYTE	error code of the last occuring error (last_command)
reserved_byte4	BYTE	Reserved
error_command1	WORD	Last command (has lead to error code) "last_error"
error_command2	WORD	Last command (has lead to error code) "last_error"
error_command3	WORD	Last command (has lead to error code) "last_error"
reserved_word5	WORD	Reserved
reserved_array_byte	ARRAY[1..30] of Byte	

Table 1-198 ATTRIBUTE "0x89" (data type "IID_READSTAT_89_RF68xR")

Name	Type	Comment
status_info	BYTE	SLG status mode(Subcommand)
hardware_version	BYTE	Version of hardware
firmware_version	ARRAY[1..4] of CHAR	Version of firmware
config ID	DWORD	Unix timestamp
inventory_status	WORD	0=inventory not active; 1=inventory active; 2=presence mode active
sum_of_filtered_tags	WORD	All filtered Tags
filtered_smoothing	WORD	Filtered Tags trough Smoothing
filtered_blacklist	WORD	Filtered Tags trough Blacklist
filtered_data-filter	WORD	Filtered Tags trough Data-Filter
filtered_RSSI_threshold	WORD	Filtered Tags trough RSSI Threshold
filtered_RSSI_delta	WORD	Filtered Tags trough RSSI Delta

Table 1-199 ATTRIBUTE "0xA0" and "0xA1" (data type "IID_READSTAT_A0_A1_RF600")

Name	Type	Comment
reserved	BYTE	
Status_info	BYTE	Status-Info, SLG-Status SubCommand 20/21
number_tags_frame	BYTE	Number of Tags in this frame
number_tags_next_frames	BYTE	Number of Tags in the next frames
reserved_byte1	BYTE	Reserved
reserved_byte2	BYTE	Reserved
reserved_byte3	BYTE	Reserved
reserved_byte4	BYTE	Reserved
reserved_byte5	BYTE	Reserved
reserved_byte6	BYTE	Reserved
Black_List_ID	ARRAY[1..13] of "IID_IN_Blacklist"	EPC-ID Length

Name	Type	Comment
Black_List_ID[1]	"IID_IN_Blacklist"	
EPC_Length	BYTE	EPC-ID Length
Antenna	BYTE	Antenna = Default 3
Filtered_Tag	WORD	Number of times - EPC-ID filtered out via BlackList
EPC	ARRAY[1..12] of Byte	EPC-ID
Black_List_ID[2]	"IID_IN_Blacklist"	
Black_List_ID[3]	"IID_IN_Blacklist"	
Black_List_ID[4]	"IID_IN_Blacklist"	
Black_List_ID[5]	"IID_IN_Blacklist"	
Black_List_ID[6]	"IID_IN_Blacklist"	
Black_List_ID[7]	"IID_IN_Blacklist"	
Black_List_ID[8]	"IID_IN_Blacklist"	
Black_List_ID[9]	"IID_IN_Blacklist"	
Black_List_ID[10]	"IID_IN_Blacklist"	
Black_List_ID[11]	"IID_IN_Blacklist"	
Black_List_ID[12]	"IID_IN_Blacklist"	
Black_List_ID[13]	"IID_IN_Blacklist"	

You will find more detailed information on the individual status modes in the manuals matching the modes "FB 45", "FB55" and "SIMATIC RF620R/RF630R".

The identifiers of the status modes correspond to the following identifiers in the other manuals:

0x81 ≙ 0x01
0x82 ≙ 0x02
0x83 ≙ 0x03
0x85 ≙ 0x05
0x87 ≙ 0x07
0x88 ≙ 0x08
0x90 ≙ 0x10
0x91 ≙ 0x11
0x92 ≙ 0x12

0xA0 ≙ 0x20
0xA1 ≙ 0x21

Tag_Status

The "Tag_Status" block reads the status information of the transponder. For the various transponder types and reader families, there are different status modes that you can select using the "ATTRIBUTE" parameter.

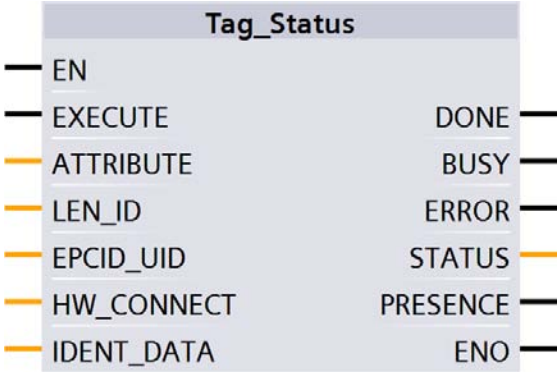

Figure 1-127 "Tag_Status" block

Table 1-200 Explanation of the "Tag_Status" block

Parameter	Data type	Default values	Description
ATTRIBUTE	Byte	B#16#0	Identifier of the status modes / possible entries: • RF200: 0x83 • RF300: 0x04, 0x82, 0x83 [1)] • RF620R, RF630R: 0x84, 0x85 • MOBY D: 0x83 [2)] • MOBY U: 0x80
LEN_ID	Byte	B#16#0	Length of the EPC-ID/UID Default value: 0x00 ≙ unspecified single tag access (RF61xR, RF68xR)

Parameter	Data type	Default values	Description
EPCID_UID	Array[1...62] of Byte	0	Buffer for up to 62 bytes EPC-ID, 8 bytes UID or 4 bytes handle ID. • 2 - 62-byte EPC-ID is entered at the start of the buffer (length is set by "LEN_ID") • 8-byte UID is entered at the start of the buffer ("LEN_ID = 8") • 4-byte handle ID must be entered in the array element [5]-[8] ("LEN_ID = 8") Default value: 0x00 ≙ unspecified single tag access (RF620R, RF630R)
IDENT_DATA	Any / Variant	0	Event values depending on attributes Note: You create either an "Array_of_Byte" or a variable with one of the following data types. Other data types / UDTs can also be created for "Any".

1) With RF300 first generation with the attribute "0x83" the values can only be queried with ISO transponders. With RF300 second generation with this attribute the values can be queried with all transponder types. In "General Mode" only this attribute functions.

2) SLG D10S only

Results

Note that the UDTs can only be displayed when the blocks "Reader_Status" or "Tag_Status" are used.

Table 1-201 ATTRIBUTE "0x04" ("IID_TAG_STATUS_04_RF300" data type)

Name	Type	Comment
reserved	BYTE	
status_info	BYTE	MDS status mode
UID	ARRAY [1..8] of BYTE	
MDS_type	BYTE	Type of MDS
Lock_state	BYTE	Write Protection Status EEPROM
Reserved1	ARRAY[1..6] of BYTE	

Table 1-202 ATTRIBUTE "0x80" ("IID_TAG_STATUS_80_MOBY_U" data type)

Name	Type	Comment
UID	ARRAY [1..4] of BYTE	Unique indentifier (MDS-Number)
MDS_type	BYTE	Type of MDS
sum_subframe_access_1	BYTE	Sum of subframe access Byte 1
sum_subframe_access_2	BYTE	Sum of subframe access Byte 2
sum_subframe_access_3	BYTE	Sum of subframe access Byte 3
sum_subframe_access_4	BYTE	Sum of subframe access Byte 4
sum_searchmode_access_1	BYTE	Sum of search mode access Byte 1
sum_searchmode_access_2	BYTE	Sum of search mode access Byte 2
ST_date_Week	BYTE	Date of last sleep-time change (week of year)
ST_date_Year	BYTE	Date of last sleep-time change (year)
battery_left_1	BYTE	Battery power left (percent) Byte 1
battery_left_2	BYTE	Battery power left (percent) Byte 2
ST	BYTE	Actual sleep-time on MDS

Table 1-203 ATTRIBUTE "0x82" ("IID_TAG_STATUS_82_RF300" data type)

Name	Type	Comment
reserved	BYTE	
status_info	BYTE	MDS status mode
UID	ARRAY [1..8] of BYTE	
LFD	BYTE	Magnetic flux density: correlation between limit-value
FZP	BYTE	Error counter passive: distortion without communication
FZA	BYTE	Error counter active: distortion during communication
ANWZ	BYTE	Presence counter: measure value for presence time
reserved1	ARRAY [1..3] of BYTE	

Table 1-204 ATTRIBUTE "0x83" ("IID_TAG_STATUS_83_ISO" data type)

Name	Type	Comment
reserved	BYTE	
status_info	BYTE	MDS status mode
UID	ARRAY [1..8] of BYTE	

Name	Type	Comment
MDS_Type	BYTE	Type of MDS
IC_version	BYTE	Chip version
size_HB	BYTE	Size of Memory (high Byte)
size_LB	BYTE	Size of memory (low Byte)
lock_state	BYTE	Write protection status EEPROM
block_size	BYTE	Size of a block in addressable memory
number_of_block	BYTE	Number of blocks in addressable memory

Table 1-205 ATTRIBUTE "0x84" ("IID_TAG_STATUS_84_RF600" data type)

Name	Type	Comment
reserved	BYTE	
status_info	BYTE	MDS status mode
UID	ARRAY [1..8] of BYTE	
antenna	BYTE	Antenna which has observed the MDS
RSSI	BYTE	RSSI value
last_observed_hour	BYTE	Last observed time hour
last_observed_min	BYTE	Last observed time minute
last_observed_sec	BYTE	Last observed time seconds
last_observed_channel	BYTE	Last observed time channel
EPC_length	BYTE	EPC-Length
reserved1	BYTE	

1) The internal time stamp of the internal reader clock that relates to this event is output. The internal reader clock is not synchronized with UTC.

Table 1-206 ATTRIBUTE "0x85" ("IID_TAG_STATUS_85_RF600" data type)

Name	Type	Comment
status_info	BYTE	MDS status mode
antenna	BYTE	Antenna which has observed the MDS
channel	BYTE	Channel
UID	ARRAY [1..8] of BYTE	
DT_glimpsed_1	BYTE	Time elasped between acknowledgement and first read in [ms]1 Highbyte
DT_glimpsed_2	BYTE	Time elasped between acknowledgement and first read in [ms]2
DT_glimpsed_3	BYTE	Time elasped between acknowledgement and first read in [ms]3
DT_glimpsed_4	BYTE	Time elasped between acknowledgement and first read in [ms]4 Low-Byte

Name	Type	Comment
reserved1	BYTE	
reserved2	BYTE	
reserved3	BYTE	
reserved4	BYTE	
last_observed_hour	BYTE	Last observed time hour
last_observed_min	BYTE	Last observed time minutes [1]
last_observed_sec	BYTE	Last observed time seconds [1]
last_observed_EPC_length	BYTE	Last observed time EPC length
EPC_ID_Byte	ARRAY [1..62] of BYTE	EPC-ID
reads_HB	BYTE	Number of Reads of MDS in Inventory (1 - 65535)
reads_LB	BYTE	Number of Reads of MDS in Inventory (1 - 65535)
RSSI	BYTE	Current RSSI value of MDS [2]
mean_RSSI	BYTE	Mean RSSI value of MDS
max_RSSI	BYTE	Max RSSI value of MDS
min_RSSI	BYTE	Min RSSI value of MDS
min_POWER	BYTE	Min Power value of MDS
current_POWER	BYTE	Current Power value of MDS [3]
reserved5	ARRAY[1..137] of BYTE	

[1] The internal time stamp of the reader that relates to this event is output. The internal reader time stamp is not synchronized with UTC.

[2] The value "Reads" indicates the total transponder recognitions (inventories) regardless of the set smoothing parameters. In this way, in extreme situations, the "Reads" counter can reach extremely high values without the transponder ever reaching the "Observed" status.

[3] The "current_Power" value is specified as transmit power in 0.25 dBm steps (ERP/EIRP). A "current_Power" value of "72" (0x48) therefore corresponds to 18 dBm (ERP/EIRP).

You will find more detailed information on the individual status modes in the manuals matching the modes "FB 45", "FB55" and "SIMATIC RF620R/RF630R".

The identifiers of the status modes correspond to the following identifiers in the other manuals:

0x04 ≙ 0x01
0x82 ≙ 0x02
0x83 ≙ 0x03
0x84 ≙ 0x04
0x85 ≙ 0x05

Programming the Ident profile

Structure of the Ident profile

Note

Parallel operation using Ident blocks and Ident profile is not possible

Note that the CM or reader cannot be operated at the same time using the Ident blocks and the Ident profile.

The blocks described in the section"AUTOHOTSPOT" represent a simplified interface of the Ident profile. If the functionality available with the blocks is not adequate for your application, you can use the Ident profile as an alternative. Using the Ident profile, you can set complex command structures and work with command repetition. The following graphic shows the Ident profile including the commands that can be implemented with it.

Note

Ident profile for trained users

The Ident profile is a complex block containing all the functionality of the Ident blocks. The Ident profile was developed specially for trained block users who want to configure complex functions with their own blocks. For untrained users, we recommend using the Ident blocks.

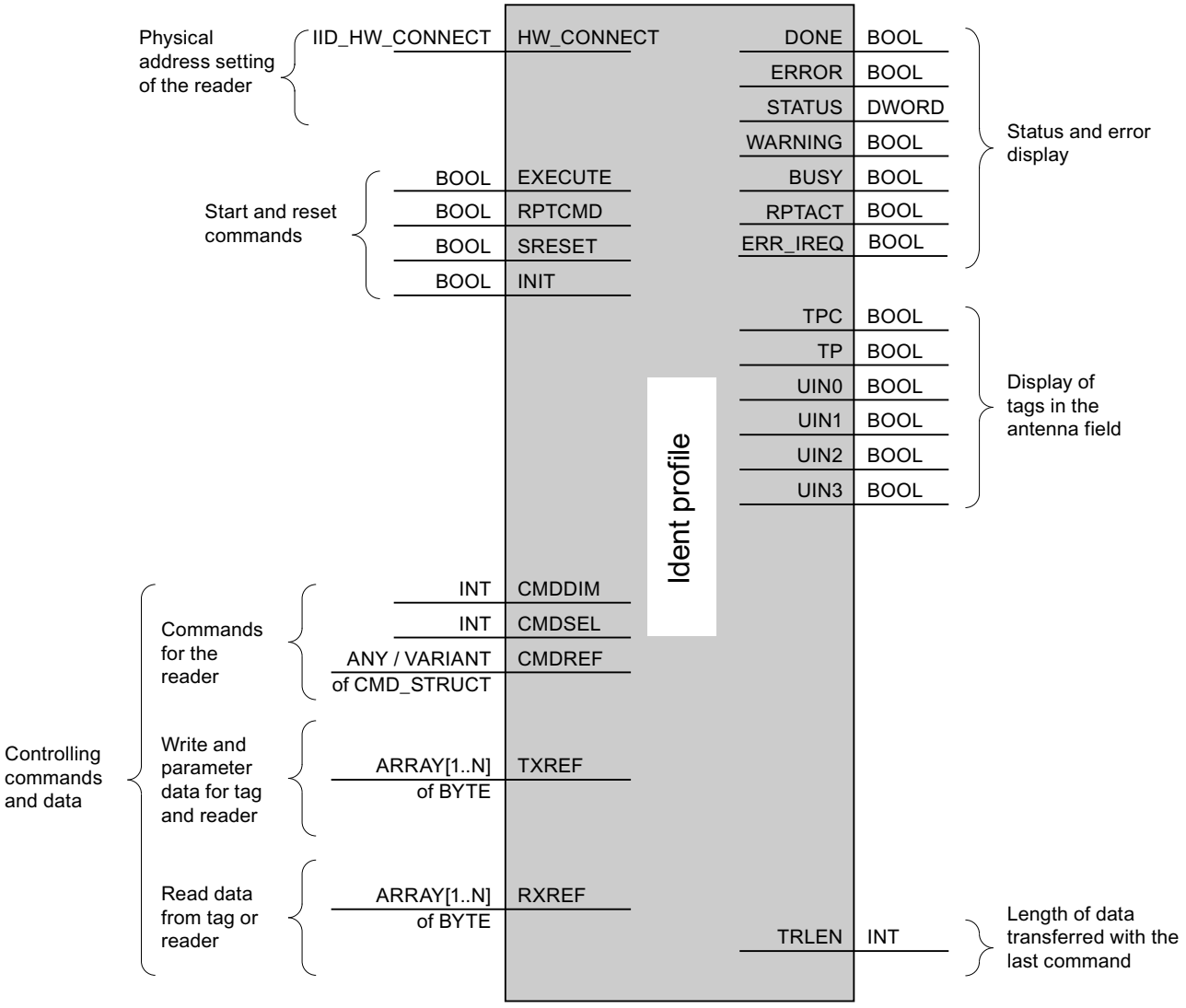

Figure 1-128 The input parameters of the Ident profile

Note

Working with multiple channels

If you work with several channels, you must ensure that for each channel, the block is called with a separate instance DB.

Interface description

Table 1-207 Input parameter

Input parameter	Data type	Default value	Description
HW_CONNECT	HW_CONNECT	--	Own data type for physical addressing of communications modules and readers and for synchronizing the blocks used for each reader. The addressing is as described in the section "Configure 'IID_HW_CONNECT' data type of the Ident Profile and Ident Blocks, Standard Function for Ident Systems manual (https://support.industry.siemens.com/cs/ww/en/ps/14971/man)".
EXECUTE	BOOL	FALSE	TRUE = triggers a new command Before starting you need to set the command and the corresponding parameters in the memory linked to "CMDREF".
RPTCMD	BOOL	FALSE	TRUE = Repeating the command currently being executed or the next command to be executed by communications module
SRESET	BOOL	FALSE	TRUE = Cancellation of the command currently processed on the communications module
INIT	BOOL	FALSE	TRUE = Communications module executes a restart and is re-assigned parameters
CMDDIM	INT	10	Number of commands in the parameter "CMDREF"
CMDSEL	INT	0	Selection of the command to be executed "CMDREF"; 1 ⇒ 1. command, ... The value of the "CMDSEL" parameter can never be higher than the value of the "CMDDIM" parameter.
CMDREF	ARRAY[1...n] of CMD_STRUCT	--	Command field The field can hold up to 100 commands. The commands are complex variables of the type "CMD_STRUCT". You can find more information on "CMDREF" in the section "Command overview of the manual Ident Profile and Ident Blocks, Standard Function for Ident Systems manual (https://support.industry.siemens.com/cs/ww/en/ps/14971/man)".
TXREF	ARRAY[1...n] of BYTE	--	Reference to global memory area for send data. The memory area can be shared with other block instances. The value "n" of the individual blocks is variable and can be up to 32 KB in size.
RXREF	ARRAY[1...n] of BYTE	--	Reference to global memory area for receive data. The memory area can be shared with other block instances. The value "n" of the individual blocks is variable and can be up to 32 KB in size.

Table 1-208 Output parameter

Output parameter	Data type	Default value	Description
DONE	BOOL	FALSE	TRUE = Command was executed free of errors.
ERROR	BOOL	FALSE	TRUE = Error was detected. The error is output in the "STATUS" parameter. The bit is reset automatically when a new command is started.
STATUS	DWORD	FALSE	Warning and error If "ERROR = TRUE" or "WARNING = TRUE", the error or warning information is contained in the "STATUS" parameter. For more information, refer to the section "Error messages of the manual Ident Profile and Ident Blocks, Standard Function for Ident Systems manual (https://support.industry.siemens.com/cs/ww/en/ps/14971/man)".
WARNING	BOOL	FALSE	TRUE = Warning was detected. The warning is output in the "STATUS" parameter. If the "ERROR" parameter is not set at the same time, the data was correctly processed. The bit is reset automatically when a new command is started.
BUSY	BOOL	FALSE	TRUE = the block is executing a command. Other commands except for "INIT" and "SRESET" cannot be started.
RPTACT	BOOL	FALSE	TRUE = "RPTCMD" is active The acknowledgment bit shows that the "Repeat mode" of the CM/reader is active.
ERR_IREQ	BOOL	FALSE	TRUE = An error has occurred on the communications module or reader (e.g. on start-up of the reader, connection termination or with RF61xR, RF68xR on antenna errors or configuration changes)
TPC	BOOL	FALSE	Transponder Presence Changed (only when the Presence_Mode is active) TRUE = A new transponder enters the antenna field of the reader or a transponder has left the antenna field. The parameter is set to "FALSE" after the successful execution of the next "INVENTORY" (0x80, 0x81, 0x87) or "INIT" command.
TP	BOOL	FALSE	Transponder Presence TRUE = There is a transponder in the antenna field of the reader.
UIN0	BOOL	FALSE	With RFID readers, the number of transponders in the antenna field is indicated. With optical reader devices, the various states of the reader device are displayed. UIN0: Corresponds to "IN_OP" bit of the reader UIN1: Corresponds to "RDY" bit of the reader UIN2 + UIN3: These two bits are interpreted as an unsigned value (bit 2 is the less significant bit) that represents the number of available decoded codes. If the value is = 3, three or more decoded codes are available.
UIN1	BOOL	FALSE	
UIN2	BOOL	FALSE	
UIN3	BOOL	FALSE	
TRLEN	INT	0	Number of data elements received after successful execution of the command.

Data structure of the Ident profile

Each time the Ident profile is called, you need to supply the parameters ("HW_CONNECT", "CMDREF", "TXREF" and "RXREF") with values as described in section "AUTOHOTSPOT".

The call for the Ident profile is always via the input parameter "HW_CONNECT" and the "IN/OUT" parameters "CMDREF", "TXREF" and "RXREF". All three parameters need to be created in a data block. The relationship between the three "IN/OUT" parameters is described in greater detail below:

- CMDREF (command buffer):
  - Array[1...n] of CMD_STRUCT
- TXREF (send buffer):
  - Array[1...n] of Byte
  - For Any, other data types/UDTs can also be created.
- RXREF (receive buffer):
  - Array[1...n] of Byte
  - For Any, other data types/UDTs can also be created.

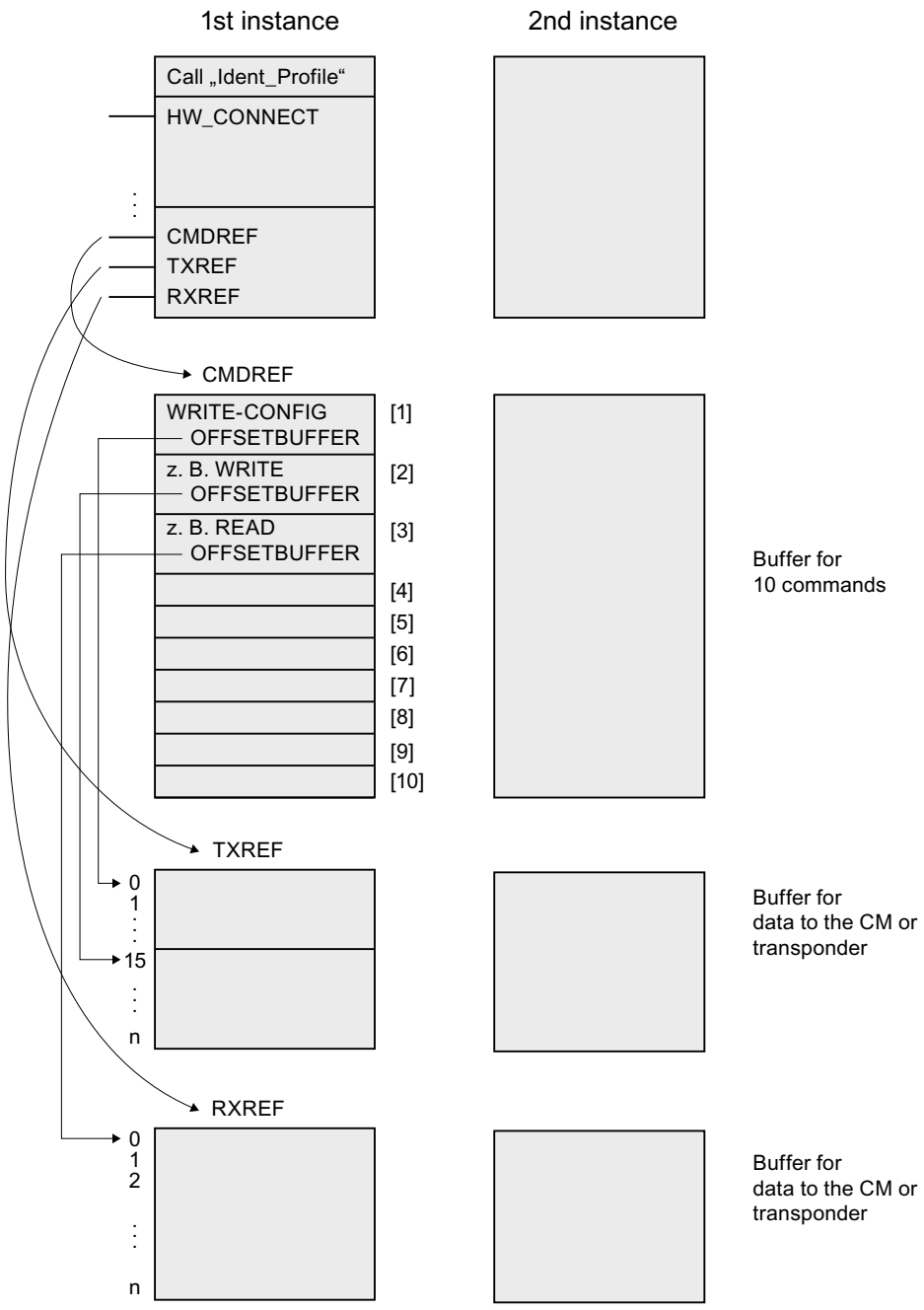

Figure 1-129 Data structure example of the Ident profile

Explanation of the data structure example

- CMDREF[1]:
  Command "WRITE-CONFIG", OFFSETBUFFER = 0
  At the "CMDREF[1]" point you need to set the "WRITE-CONFIG" command so that the "INIT/Reset" is correctly executed.
- CMDREF[2]:
  Command "WRITE", OFFSETBUFFER = 15
- CMDREF[3]:
  Command "READ", OFFSETBUFFER = 0

If the "CMDREF[2]" command is selected, a write command is started and the data to be written is fetched starting at byte 15 of the "TXREF" parameter. If the "CMDREF[3]" command is selected, the read data is stored starting at byte 0 in the "RXREF" parameter.

Communications modules

Communications module RF120C

"Reader" parameter group

In this parameter group, you can configure all reader-specific parameters of the RF120C.

Table 1-209 Parameters of the parameter group "Readers"

Parameter	Parameter value	Default value	Description
Diagnostics messages	None Hard errors	Hard errors	With this parameter, you set whether hardware diagnostics messages will be reported. • None: Apart from standard diagnostics, no other alarms are generated. • Hard errors: Critical hardware errors are reported by the S7 diagnostics.
User mode	Ident profile	Ident profile	Selection depends on the communications module and Ident system being used. With this parameter, you select the block: • Ident profile The program block for the Ident profile is used on the controller.
Ident device / system	RF200 general RF290R RF300 general RF380R RF600 SLG D10S SLG D11S/D12S MOBY U General Reader Parameters via FB / optical readers	RF300 general	Selection of the connected Ident device / system. Depending on the selection you make, the "Ident system" parameter group is adapted.

Parameter group "Ident device / system"

Depending on the set basic parameters you can configure the following reader parameters in this parameter group.

Table 1-210 Overview of the "Reader parameters"

Reader parameter assignment in "Basic parameters"	Transfer speed	MOBY mode	Stand by time	Presence check	Reset ERR LED	HF power	Range limitation	Max. number of transponders	Transponder type	Reader type	Bero mode	BERO time	Byte sequence of the reset parameter [hex]
RF200 general	✓	-	-	✓	✓	-	-	✓	✓	-	-	-	-
RF290 R	✓	-	-	✓	✓	✓	-	✓	✓	-	-	-	-
RF300 general	✓	-	-	✓	✓	-	-	✓	✓	-	-	-	-
RF380 R	✓	-	-	✓	✓	✓	-	✓	✓	-	-	-	-
RF600	✓	-	-	✓	✓	-	-	✓	-	✓	-	-	-
SLG D10S	✓	-	-	✓	✓	✓	-	✓	✓	-	-	-	-
SLG D11S/ D12S	✓	-	-	✓	✓	✓	-	✓	✓	-	-	-	-
MOBY U	✓	-	✓	✓	✓	-	✓	-	-	-	✓	✓	-
General Reader	✓	-	-	-	-	-	-	-	-	-	-	-	✓
Parameters via FB / optical readers	✓	✓	-	-	-	-	-	-	-	-	-	-	-

Table 1-211 Parameters of the "Ident device/system" parameter group

Parameter	Parameter value	Default value	Description
Transmission speed	19.2 kBd 57.6 kBd 115.2 kBd	115.2 kBd	Selection depends on the communications module and Ident system being used. Note that the value specified here is adopted automatically from the device configuration of the connected devices. With this parameter, you set the transmission speed between the communications module and reader. When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power). When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.
MOBY mode	RF200/RF300/RF600; MV400/MV500; MOBY U/D	RF200/RF300/ RF600; MV400/ MV500; MOBY U/D	With this parameter, you set the mode of the communications module.
Standby time	0 - 1400 ms	0 ms	Standby time (scanning_time) for the transponders. If the transponder receives a further command before standby time has expired, this command can be executed immediately. If the transponder receives a command after the standby time has expired, command execution is delayed by the "sleep_time" of the transponder.
Presence check	On Off (RF field on) Off (RF field off)	On	On = As soon as there is a transponder in the antenna field of the reader, its presence is reported. Off (RF field on) = the presence check in the FB is suppressed. The antenna on the reader is nevertheless turned on as long as it has not been turned off by a command. Off (RF field off) = the antenna is turned on only when a command is sent and it then turns itself off again.
Reset ERR LED	On Off	On	On = the flashing of the error LED on the communications module is reset by each FB reset. Off = the error LED always indicates the last error. The display can only be reset by turning off the communications module.
HF power [1)]	0.50 ... 5.00	1.00 1.25	Setting for the output power of the reader. The selectable values depend on the connected device.

Parameter	Parameter value	Default value	Description
Range limitation	0.2 m 0.5 m 1.0 m 1.5 m 2.0 m 2.5 m 3.0 m 3.5 m	1.5 m	Limitation of the radiation range
Max. number of transponders	1 ... 40	1	Number of transponders expected in the antenna field. The selection depends on the connected device.
Transponder type	1)	1)	Selection of the transponder types used. The selection depends on the connected device.
Reader type 1)	RF620R ETSI RF620R FCC RF620R CMIIT RF630R ETSI RF630R FCC RF630R CMIIT	RF620R ETSI	Selection of the reader used. Selecting a reader opens the "Reader type" parameter group. The parameters are described in the table below:
Bero mode	Without BEROs 1 or 2 BEROs 1st BERO on, 2nd BERO off Synchronization by cable connection	Without BEROs	• Without BEROs No reader synchronization • 1 or 2 BEROs The BEROs are ORed. The antenna field is turned on during the actuation of a BERO. • 1st BERO on, 2nd BERO off The 1st BERO turns on the antenna field and the 2nd BERO turns the antenna field off. If there are two BEROs present and "BERO time in s" is set, the antenna field is turned off automatically if the 2nd BERO does not switch within this BERO time. If no "BERO time in s" is set, the antenna field remains turned on until the 2nd BERO is activated. • Synchronization by cable connection Activate reader synchronization via cable connection (see manual for configuring, mounting and service for MOBY U)

Parameter	Parameter value	Default value	Description
BERO time	0 ... 255 s	0 s	Can only be set if the BERO mode is set to "1st BERO on, 2nd BERO off". • 0 Timeout monitoring is deactivated. The 2nd BERO is needed in order to switch the field off. • 1 ... 255 s Activation time for the reader field
Byte sequence of the reset parameter [hex]	1)	1)	Hexadecimal representation of the byte sequence of the reset parameter of the reader.

1) You can find a detailed description of this parameter in the following paragraphs:

The "HF power" parameter

The selectable values depend on the value specified in the parameter "Ident device/system".

- RF290R
  - Value range: 0.50 ... 5.00 W
  - Default value: 1.00 W
- RF380R
  - Value range: 0.50 ... 2.00 W
  - Default value: 1.25 W

The "Transponder type" parameter

Selection of the transponders used. The selection depends on the value specified in the parameter "Ident device/system". The following transponder types can be selected:

Table 1-212 "Transponder type" selection option

"Ident device / system" parameter	Possible Values
RF200 general RF290R	ISO 15693
RF300 general RF380R	RF300 ISO 15693
SLG D10S SLG D11S/D12S	I code ISO 15693 ISO-FRAM

The parameter "Byte sequence of the reset parameter [hex]"

With this parameter, you can specify the Reset parameter in hexadecimal format. This setting is intended only for trained users.

The following Reset parameters are available:

Table 1-213 Reset parameters

Byte	1	2...5 [1]	6	7...8	9	10	11	12	13...14	15	16
Value	0x04	0x00	0x0A	0x00	scan-ning_ time	param	op-tion_1	distance_ limiting	multitag	field_on_ control	field_on_ time

[1] In the communications module RF180C as of V2.2 in conjunction with MOBY U byte 4 is preset with the calendar week and 5 with the year.

Parameter group "Reader type" (only with RF600)

Additional specific parameters are set in the "Reader type" parameter group for the "Reader type" selected in the "Ident system" parameter group.

Table 1-214 Parameters of the "Reader type" parameter group

Parameter	Parameter value	Default value	Description
Wireless profile	-	-	Selection of the relevant wireless profile for ETSI, FCC or CMIIT.
Multitag mode	UID = EPC-ID (8 bytes) UID = Handle ID (4 bytes) UID = 0 (single tag)	UID = EPC-ID (8 bytes)	• UID = EPC-ID (8 bytes) 8 bytes UID bytes 5-12 of 12-byte long EPC-ID • UID = Handle ID (4 bytes) 4 bytes UID as handle ID for accessing the transponder with EPC-ID of any length
Intelligent Single Tag Mode (ISTM)	On Off	Off	Enable/disable the "Intelligent Single Tag Mode ISTM" algorithm [1]
Black list	On Off	Off	Enable/disable the "Black list" [1]
Radiated power internal antenna (RF620R)	0 ... B	4	Setting the radiated power for the internal antenna [1] [2]
Internal antenna (RF620R)	-	-	Enable/disable of internal antenna With the RF620R, either only the internal or only the external antenna can be set.
Transmitting power of external antenna connector (RF620R)	0 ... 9	4	Setting the transmit power for the external antenna [1] [2]
External antenna (SetAnt required) (RF620R)	-	-	Activation/deactivation the external antenna. With the RF620R, either only the internal or only the external antenna can be set.
Transmitting power ANT 1 (RF630R)	0 ... 9	4	Setting the transmitting power for antenna 1[1] [2]
Transmitting power ANT 2 (RF630R)	0 ... 9	4	Setting the transmitting power for antenna 2[1] [2]
Communication speed	Reliable detection Fast detection	Reliable detection	[1]

Parameter	Parameter value	Default value	Description
Tag hold	On Off	Off	Enable/disable "Tag hold" [1]
Scanning mode	On Off	Off	Enable/disable the "Scanning mode" [1]
Channel assignment (only with wireless profile ETSI)	-	-	Selection of the wireless channels to be used [1]

[1] You can find further information in the "RF620R/RF630R Configuration Manual". *

[2] The values for the transmit/radiated power of the antennas can be found in the following table.

* You can find the RF620R/RF630R Configuration Manual here.

Table 1-215 Transmit / radiated power of the antennas

Hex value	RF630R transmit power	RF620R radiated power (internal antenna)			RF620R transmit power
		ETSI	FCC	CMIIT	
	dBm / (mW)	dBm / (mW) ERP	dBm / (mW) EIRP	dBm / (mW) ERP	dBm / (mW)
0	18 / (63)	18 / (63)	20 / (100)	18 / (65)	18 / (63)
1	19 / (79)	19 / (79)	21 / (126)	19 / (79)	19 / (79)
...	...	...	...	...	...
4	22 / (158)	22 / (158)	24 / (251)	22 / (158)	22 / (158)
...	...	...	...	...	...
9	27 / (501)	27 / (501)	29 / (794)	27 / (501)	27 / (501)
O	27 / (501)	28 / (631)	30 / (1000)	28 / (631)	27 / (501)
B (...F)	27 / (501)	29 / (794)	31 / (1259)	29 / (794)	27 / (501)

Communications module RF170C

Communications module for Ident systems - RF170C, RS422/RS232

The module RF170C RS422/RS232 is a communications module for Ident systems for the I/O system ET 200pro. Several RF170C RS422/RS232 communications modules can be mounted and operated on an IM 154-8 CPU and on an interface module IM 154-1 DP, IM 154-2 DP High Feature or IM 154-4 PN High Feature of the ET 200pro.

A maximum of two readers can be connected to the RF170C RS422/RS232. The connected readers are processed at the same time

The function blocks FB 45, FC 45 and the program block for the Ident profile/RFID standard profile allow simple programming using the SIMATIC S7 tools. The function blocks FB 45, FC 45 and the program block for the Ident profile/RFID standard profile can be operated both on the S7-300 and the S7-400. For the S7-1200 and S7-1500 controllers, the Ident profile must be used.

The transponder data is accessed by means of normal addressing or file handler addressing.

Physical addressing of the transponder

Physical addressing of a transponder is also referred to as normal addressing. When using normal addressing you can set up the data structure of a transponder yourself You decide which physical address which data is written to. The addressing of the transponder memory generally begins at the address 0x0000 hex and ends at an end address that corresponds to the size of the transponder memory.

File handler addressing of the transponder

With file handler addressing you specify a file name for access to the data. The file name consists of eight ASCII characters. The file handler manages the user data on the transponder independently. You do not need to configure any data structures on the transponder.

"Module parameters" parameter group

In this parameter group, you can configure all module-specific parameters of the RF170C RS422/RS232.

Table 1-216 Parameters in the "Module parameters" parameter group

Parameter	Parameter value	Default value	Description
User Mode	Ident profile/RFID standard profile FB 45 / FC 45 FB 55 / FC 55	Ident profile/RFID standard profile	Selection depends on the communications module and and Ident system being used. With this parameter, you select the block: • Ident profile/RFID standard profile: The program block for the Ident profile/RFID standard profile is used on the controller. • FB 45 / FC 45: Single tag mode FB 45 (PROFIBUS/ PROFINET) or FC 45 (PROFIBUS) is used on the controller. • FB 55 / FC 55: Multitag mode. FB 55 (PROFIBUS/ PROFINET) or FC 55 (PROFIBUS) is used on the controller.
MOBY Mode	RF200/RF300/RF600; MV400/MV500; MOBY U/D MOBY I/E MV300 Freeport protocol RF300 filehandler	RF200/RF300/ RF600; MV400/ MV500; MOBY U/D	Selection depends on the communications module and and Ident system being used. With this parameter, you set the mode of the communications module. • RF200/RF300/RF600; MV400/MV500; MOBY U/D • MOBY I/E • MV300 • Freeport protocol • RF300 filehandler Normal addressing: The transponder is addressed with physical addresses. Filehandler: Prior to use, the transponder needs to be formatted.
Transmission speed	19.2 kBd 57.6 kBd 115.2 kBd	115.2 kBd	Selection depends on the communications module and and Ident system being used. With this parameter, you set the transmission speed between the communications module and reader. When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again. With a connected optical reader: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.

Parameter	Parameter value	Default value	Description
Diagnostics messages	None Hard errors	None	Selection depends on the communications module and and Ident system being used. With this parameter, you set whether hardware diagnostics messages will be reported. • None: Apart from standard diagnostics, no other alarms are generated. • Hard errors: Critical hardware errors are reported by the S7 diagnostics.
Suppression of Error LED	None Channel 1 Channel 2	None	Disabling the Error LED (ERR) of a channel. The communications module has two channels to which the readers / optical readers can be connected. The Error LED of the other channel flashes permanently when only one of the channels is being used. With the help of the suppression, you can disable the Error LED of the unused channel.
Interface	RS-232 RS-422	--	Selection of the interface type that the connected hardware (reader / optical reader) uses.

"Message Frame" parameter group

The "Frame" parameter group is only displayed when you have selected the parameter value "Freeport protocol" in the "MOBY Mode" parameter of the module parameters. In this parameter group, you can configure all parameters specific to the "Freeport protocol".

Table 1-217 Parameters in the "Frame" parameter group

Parameter	Parameter value	Default value	Description
Data bits	7 8	8	Selection of the number of bits on which a character is represented.
Parity	None Odd Even Fixed value 1 Fixed value 0	None	Parity selection A sequence of data bits can be expanded by a parity bit. With its value "0" or "1", the parity bit is added to the sum of all bits (data bits and parity bits) to form a defined status. This increases data reliability. • None: Data is sent without a parity bit. • Odd: The parity bit is set so that the sum of the data bits (including the parity bit) is odd when the signal state is "1". • Even: The parity bit is set so that the sum of the data bits (including the parity bit) is even when the signal state is "1". • Fixed value 1: The parity bit is set permanently to the value "1". • Fixed value 0: The parity bit is set permanently to the value "0".
Stop bits	1 2	1	Selection of the number of stop bits that indicate the end of a character. The stop bits are appended to every transferred character during transmission.

Parameter	Parameter value	Default value	Description
Specifying end detection	After character delay time elapses On receipt of fixed number of characters On receipt of the end delimiter(s):	After character delay time elapses	Specifies the end detection of a received frame. • After character delay time elapses: The frame has neither a fixed length nor defined end delimiters. The end of a frame is indicated by a gap in the character sequence. The size of this gap is specified by the character delay time. • On receipt of fixed number of characters: The length of the received frame is always the same. When data is received, the end of the frame is recognized when the set number of characters has been received. • On receipt of the end delimiter(s): At the end of the frame there are one or two defined end delimiters. When data is received, the end of the frame is recognized when the set end delimiter is received.
No. of end delimiters	1 2	1	Selection of the number of end delimiters. A maximum of 2 end delimiters can be configured. When data is received, the end of the frame is recognized when the selected end delimiter combination is received.
1st end delimiter	0...7F / 0...FF	3	Input of the 1st end delimiter of maximum two end delimiters for end criteria “On receipt of the end delimiter(s)”. The selected end delimiter or the selected end delimiter combination limits the length of the frame. Parameter value depending on the "Data bits" parameter.
2nd end delimiter	0...7F / 0...FF	0	Input of the 2nd end delimiter of maximum two end delimiters for end criteria “On receipt of the end delimiter(s)”. The selected end delimiter or the selected end delimiter combination limits the length of the frame. Parameter value depending on the "Data bits" parameter.
Frame length	1...233 / 1...229	233	Entry of the frame length in bytes for the end criterion "On receipt of fixed number of characters".
Character delay time	0...65535	15	Entry of the time [ms] that may elapse until a frame end is recognized. Select the character delay time dependent on the send behavior of your communications partner. Depending on the data transmission speed the character delay time is limited to a minimum value. Note that the ASCII driver pauses between two message frames when sending.

"I/O addresses" parameter group

Inputs

1. Assign a start address to the module.
2. If you want to assign the address area to a process image partition, select the desired process image partition in the "Process image" drop-down list.

Outputs

1. Assign a start address to the module.
2. If you want to assign the address area to a process image partition, select the desired process image partition in the "Process image" drop-down list.

Note

Input and output address must be identical.

RF180C communications module

Communications module for Ident systems - RF180C

The module RF180C is a communications module for Ident systems for PROFINET IO.

A maximum of two readers can be connected to the RF180C. The connected readers are processed at the same time

The function blocks FB 45, FB 55, and the program block for the RFID standard profile allow simple programming using the SIMATIC S7 tools. The function blocks FB 45, FB 55,and the program block for the RFID standard profile can be operated both on the S7-300 and the S7-400.

The transponder data is accessed by means of normal addressing or file handler addressing.

Physical addressing of the transponder

Physical addressing of a transponder is also referred to as normal addressing. When using normal addressing you can set up the data structure of a transponder yourself You decide which physical address which data is written to. The addressing of the transponder memory generally begins at the address 0x0000 hex and ends at an end address that corresponds to the size of the transponder memory.

File handler addressing of the transponder

With file handler addressing you specify a file name for access to the data. The file name consists of eight ASCII characters. The file handler manages the user data on the transponder independently. You do not need to configure any data structures on the transponder.

"Module parameters" parameter group

In this parameter group, you can configure all module-specific parameters of the RF180C.

Table 1-218 Parameters in the "Module parameters" parameter group

Parameter	Parameter value	Default value	Description
User Mode	FB 45 FB 55 FB 56 RFID standard profile	FB 45	Selection depends on the communications module and and Ident system being used. With this parameter, you select the block: • FB 45: Single tag mode FB 45 (PROFIBUS/ PROFINET) is used on the controller. • FB 55: Multitag mode. FB 55 (PROFIBUS/ PROFINET) is used on the controller. • FB 56: Multitag mode. FB 56 (PROFIBUS/ PROFINET) is used on the controller. • RFID standard profile The program block for the RFID standard profile is used on the controller.
MOBY Mode	MOBY I/E normal addressing MOBY I file handler RF200/RF300/RF600; MOBY U/D normal adr. MOBY U file handler RF300 filehandler	RF200/RF300/ RF600; MOBY U/D normal addr.	Selection depends on the communications module and and Ident system being used. With this parameter, you set the mode of the communications module. • MOBY I/E normal addressing • MOBY I file handler • RF200/RF300/RF600; MOBY U/D normal adr. • MOBY U file handler • RF300 filehandler Normal addressing: The transponder is addressed with physical addresses. Filehandler: Prior to use, the transponder needs to be formatted.
Transmission speed	19.2 kBd 57.6 kBd 115.2 kBd	115.2 kBd	Selection depends on the communications module and and Ident system being used. With this parameter, you set the transmission speed between the communications module and reader. When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again. With a connected optical reader: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.

Parameter	Parameter value	Default value	Description
Diagnostics messages	None Hard errors Hard/Soft Errors	None	With this parameter, you set whether hardware diagnostics messages will be reported. • None: Apart from standard diagnostics, no other alarms are generated. • Hard errors: Critical hardware errors are reported by the S7 diagnostics. • Hard/Soft Errors: Critical hardware faults and errors occurring when processing commands are reported by the S7 diagnostics.
Suppression of Error LED	None Channel 1 Channel 2	None	Disabling the Error LED (ERR) of a channel. The communications module has two channels to which the readers / optical readers can be connected. The Error LED of the other channel flashes permanently when only one of the channels is being used. With the help of the suppression, you can disable the Error LED of the unused channel.

ASM 456 communications module

Communications module for Ident systems - ASM 456

The ASM 456 module is a communications module for Ident systems for PROFIBUS DP V1.

A maximum of two readers can be connected to the ASM 456. The connected readers are processed at the same time

The function blocks FB 45, FC 45, FB 55, FC 55, FC 56 and the program block for the Ident profile/RFID standard profile allow simple programming using the SIMATIC S7 tools. The function blocks FB 45, FC 45, FB 55, FC 55, FC 56 and the program block for the Ident profile/RFID standard profile can be operated both on the S7-300 and the S7-400. For the S7-1200 and S7-1500 the Ident profile must be used.

The transponder data is accessed by means of normal addressing or file handler addressing.

Physical addressing of the transponder

Physical addressing of a transponder is also referred to as normal addressing. When using normal addressing you can set up the data structure of a transponder yourself You decide which physical address which data is written to. The addressing of the transponder memory generally begins at the address 0x0000 hex and ends at an end address that corresponds to the size of the transponder memory.

File handler addressing of the transponder

With file handler addressing you specify a file name for access to the data. The file name consists of eight ASCII characters. The file handler manages the user data on the transponder independently. You do not need to configure any data structures on the transponder.

"Device-specific parameters" parameter group.

In this parameter group, you can configure all module-specific parameters of the ASM 456.

Table 1-219 Parameters in the "Module parameters" parameter group

Parameter	Parameter value	Default value	Description
User Mode	Ident profile/RFID standard profile FB 45 / FC 45 FB 55 / FC 55 FC 56	Ident profile/RFID standard profile	Selection depends on the communications module and and Ident system being used. With this parameter, you select the block: • Ident profile/RFID standard profile: The program block for the Ident profile/RFID standard profile is used on the controller. • FB 45 / FC 45: Single tag mode FB 45 (PROFIBUS/ PROFINET) or FC 45 (PROFIBUS) is used on the controller. • FB 55 / FC 55: Multitag mode. FB 55 (PROFIBUS/ PROFINET) or FC 55 (PROFIBUS) is used on the controller. • FC 56 File handler for S7-300 and S7-400
MOBY Mode	RF200/RF300/RF600; MV400/MV500; MOBY U/D RF680R/RF685R MOBY I/E normal addressing RF300 filehandler MOBY U file handler MOBY I File handler	RF200/RF300/ RF600; MV400/ MV500; MOBY U/D	Selection depends on the communications module and and Ident system being used. With this parameter, you set the mode of the communications module. • RF200/RF300/RF600; MV400/MV500; MOBY U/D • RF680R/RF685R • MOBY I/E normal addressing • RF300 filehandler • MOBY U file handler • MOBY I file handler Normal addressing: The transponder is addressed with physical addresses. Filehandler: Prior to use, the transponder needs to be formatted.

Parameter	Parameter value	Default value	Description
Transmission speed	19.2 kBd 57.6 kBd 115.2 kBd	115.2 kBd	Selection depends on the communications module and and Ident system being used. With this parameter, you set the transmission speed between the communications module and reader. When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again. With a connected optical reader: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.
Diagnostics messages	None Hard errors Hard/soft errors low priority Hard/soft errors high priority	None	Selection depends on the communications module and and Ident system being used. With this parameter, you set whether hardware diagnostics messages will be reported. • None: Apart from standard diagnostics, no other alarms are generated. • Hard errors: Critical hardware errors are reported by the S7 diagnostics. • Hard/soft errors low priority Critical hardware errors and errors that occur during command processing are reported by the S7 diagnostics. The "Ext_Diag" bit is not set. • Hard/soft errors high priority Critical hardware errors and errors that occur during command processing are reported by the S7 diagnostics. The "Ext_Diag" bit is set.
Suppression of Error LED	None Channel 1 Channel 2	None	Disabling the Error LED (ERR) of a channel. The communications module has two channels to which the readers / optical readers can be connected. The Error LED of the other channel flashes permanently when only one of the channels is being used. With the help of the suppression, you can disable the Error LED of the unused channel.

ASM 475 communications module

What you should know about the ASM 475

Introduction

Up to eight ASM 475 interface modules can be plugged in and operated in a rack of the SIMATIC S7-300. If you have a configuration with several racks (max. four racks), the ASM 475 can be plugged in and operated in each rack. This means that with the maximum configuration of a SIMATIC S7-300, 32 ASM 475 modules can be operated.

A maximum of two SLGs (write/read devices) can be connected to the ASM 475. The processing of the connected SLGs is parallel. FC45 allows simple programming using the SIMATIC S7 tools.

FC45 can be used both in the S7-300 and S7-400.

With S7-400, the ASM 475 is connected via an ET 200M.

Note that the

- IM 153-1 must have at least order number 6ES7 153-1AA03-0XB0 or 6ES7 153-1AA83-0XB0 and
- IM 153-2 at least order number 6ES7 153-2AA02-0XB0 or 6ES7 153-2AB01-0XB0.

Access to the MDS data is with normal addressing.

Physical addressing of the MDS (mobile data storage)

The physical addressing of an MDS is also known as **normal addressing**. Users set up the structure of the MDS themselves. They know the physical MDS address to which data is written. The addressing of the MDS storage generally begins at 0000 hex and ends at an end address corresponding to the size of the MDS storage.

File handler addressing of the MDS

For file handler addressing, the user specifies a file name for access to the data. The file name consists of eight ASCII characters. The file handler manages the user data independently on the MDS. The user does not need to configure data structures on the MDS.

Prior to use, the MDS needs to be formatted.

Communication module RF18xC

Communication module for Ident systems - RF18xC

The SIMATIC RF185C, RF186C and RF188C communications modules are designed for use in all areas of automation. The RF18xC module is a communications module for Ident systems for PROFINET IO and OPC UA. Depending in the device type, 1, 2 or 4 readers can be connected to the communication module. The connected readers are processed at the same time

The function blocks FB 45, FB 55 and the program block for the Ident profile/RFID standard profile allow simple programming using the SIMATIC S7 tools. The function blocks FB 45, FB 55 and the program block for the Ident profile/RFID standard profile can be operated both on the S7-300 and the S7-400. For the S7-1200 and S7-1500 the Ident profile must be used.

The transponder data is accessed by means of normal addressing or file handler addressing.

Physical addressing of the transponder

Physical addressing of a transponder is also referred to as normal addressing. When using normal addressing you can set up the data structure of a transponder yourself You decide which physical address which data is written to. The addressing of the transponder memory generally begins at the address "0x0000" hex and ends at an end address that corresponds to the size of the transponder memory.

"Web Based Management" parameter group

In this parameter group you can start the Web Based Management.

Table 1-220 Parameters in the "Web-based management" parameter group

Parameter	Description
Web Based Management	Start Web Based Management of the communications module. Web Based Management (WBM) offers extensive functions for configuring the communications module. Note: WBM can only be started when either the PROFINET connection between CPU and communications module has been established or the IP address stored in the project has been assigned to the communications module. This means that the device name has been assigned and the TIA configuration must be loaded on the SIMATIC controller.

"Configuration management" parameter group

In this parameter group you can load or save the configuration data.

Table 1-221 Parameters in the "Configuration management" parameter group

Parameter	Description
User name	User name of a user created on the communications module Note that the user must have the required rights.
Password	Entry of the password for the user

Parameter	Description
Load configuration to device	Load configuration data from the STEP 7 project into the communications module.
Check the configuration in the project.	Save configuration data of the communications module in the current STEP 7 project.

Requirement

The following requirements must be met to allow configuration data to be loaded or saved:

- The "PROFINET interface [X1]" entry contains the correct IP address of the communications module.
- The user name and corresponding password have been created in the WBM.
- The entered user has the required rights to run the download/upload.

Note

User name and password only necessary if user management is enabled

The "User name" and "Password" text boxes only need to be completed if the user management of the WBM is enabled.

"Module parameters" parameter group

In this parameter group, you can configure all module-specific parameters of the communications module.

Table 1-222 Parameters in the "Module parameters" parameter group

Parameter	Parameter value	Default value	Description
Diagnostic interrupt	Off On	Off	Switching diagnostic messages of the communication modules on and off.

Parameter group "Module parameters" of the connected reader

In this parameter group, you can configure all module-specific parameters of the connected reader.

Table 1-223 Parameters of the parameter group "Module parameters" of the connected reader

Parameter	Parameter value	Default value	Description
User mode	RFID standard profile FB 45	RFID standard profile	With this parameter, you select the block: • RFID standard profile The program block for the Ident profile is used on the controller. • FB 45: Single tag mode FB 45 (PROFIBUS/ PROFINET) is used on the controller.
MOBY mode	RF200/RF300/RF600; MOBY U/D normal addr.	RF200/RF300/ RF600; MOBY U/D normal addr.	With this parameter, you set the mode of the communications module. • RF200/RF300/RF600; MOBY U/D normal addr. Normal addressing: The transponder is addressed with physical addresses.
Transmission speed	19.2 kBd 57.6 kBd 115.2 kBd	115.2 kBd	Selection depends on the Ident system being used. With this parameter, you set the transmission speed between the communications module and reader. When an optical reader is connected: The transmission speed selected here must match the transmission speed selected for the reader.
Diagnostics messages	None Hard errors Hard/soft errors	None	With this parameter, you determine the scope with which the reader-based diagnostic alarm messages are to be reported. • None: No further alarms are generated. • Hard errors: Critical hardware errors are reported by the S7 diagnostics. • Hard/Soft Errors: Critical hardware faults and errors occurring when processing commands are reported by the S7 diagnostics.

Reader

RF680R/RF685R reader

Readers RF680R/RF685R for the UHF range

The UHF readers RF680R and RF685R are intended for use in logistics and automation, for example on a production line but are equally suitable for applications in logistics. To meet these requirements, the readers were equipped with a high transmit power and degree of protection (IP65). The readers are equipped with extensive diagnostics options and can process ISO 18000-6C and ISO 18000-6B transponders.

The RF685R has one special feature with its internal, adaptive antenna. This significantly increases the reliability of read and write actions even under difficult radio conditions.

The RF680R and RF685R readers are either integrated without problems in SIMATIC S7 automation systems via an integrated PROFINET connector or via the RS-422 interface and the ASM 456 communications module via PROFIBUS. Suitable programming blocks are available. A second Ethernet interface (both M12) can be used for diagnostics during operation so that the connection to the higher-level system does not need to be interrupted.

The WBM (Web Based Management) allows commissioning, configuration and diagnostics of the devices using an Internet browser. This makes additional updates and installation of configuration and diagnostics software unnecessary.

"Web Based Management" parameter group

In this parameter group you can start the Web Based Management.

Table 1-224 Parameters in the "Web-based management" parameter group

Parameter	Description
Web Based Management (WBM)	Starting the Web Based Management of the reader. The Web Based Management provides a wide range of functions for configuring the readers. Note: The WBM can only be started if either the PROFINET connection between the CPU and reader is established or the reader was assigned the IP address stored in the project. This means that the device name has been assigned and the TIA configuration must be loaded on the SIMATIC controller.

"Module parameters" parameter group

In this parameter group, you can configure all module-specific parameters of the RF680R/RF685R readers.

Table 1-225 Parameters in the "Module parameters" parameter group

Parameter	Parameter value	Default value	Description
RFID read point alarm	Off On	Off	Enabling/disabling read-point related diagnostics messages.

"Configuration management" parameter group

In this parameter group you can load or save the configuration data.

Table 1-226 Parameters in the "Configuration management" parameter group

Parameter	Description
User name	User name of a user created on the reader Note that the user must have the required rights.
Password	Entry of the password for the user
Load configuration to device	Loading configuration data on the reader from the STEP 7 project.
Check the configuration in the project.	Saving configuration data of the reader in the current STEP 7 project.

Requirement

The following requirements must be met to allow configuration data to be loaded or saved:

- The correct IP address of the reader is entered in the "PROFINET interface [X1]".
- The user name and corresponding password have been created in the WBM.
- The entered user has the required rights to run the download/upload.

Note

User name and password only necessary if user management is enabled

The "User name" and "Password" text boxes only need to be completed if the user management of the WBM is enabled.

"RFID communication" parameter group

In this parameter group you can configure the address range of the RFID communication.

Table 1-227 Parameters in the "RFID communication" parameter group

Parameter	Description
General	General settings Input boxes for the name of the communication and for a comment.
I/O addresses	Specify the I/O address of the reader ("LADDR") This parameter is used in the "IID_HW_CONNECT" variable.
Hardware identifier	Specify the hardware identifier of the reader ("HW-ID") This parameter is used in the "IID_HW_CONNECT" variable.

"Digital inputs/outputs" parameter group

In this parameter group you can configure the address range of the digital inputs / outputs.

Table 1-228 Parameters in the "Digital inputs / outputs" parameter group

Parameter	Description
General	General settings Input boxes for the name of the of the digital inputs / outputs and for a comment.
I/O addresses	Specify the I/O address of the digital inputs/outputs Using the set address range (I/O address), the digital inputs/outputs configured in the WBM of the reader can be accessed.
Hardware identifier	Specify the hardware identifier of the digital inputs/outputs.

RF615R reader

Readers RF615R for an UHF range

The UHF reader SIMATIC RF615R is intended for use in automation, for example in a production line, but it also suited for applications in the logistics sector. To meet these requirements, the readers were equipped with a high transmit power and degree of protection (IP65). The readers are equipped with extensive diagnostics options and can process ISO 18000-6C and ISO 18000-6B transponders.

The RF615R reader is either integrated without problems in SIMATIC S7 automation systems via an integrated PROFINET connector or via the RS-422 interface and the ASM 456 communications module via PROFIBUS. Suitable programming blocks are available.

The WBM (Web Based Management) allows commissioning, configuration and diagnostics of the devices using an Internet browser. This makes additional updates and installation of configuration and diagnostics software unnecessary.

"Module parameters" parameter group

In this parameter group, you can configure all module-specific parameters of the RF615R reader.

Table 1-229 Parameters in the "Module parameters" parameter group

Parameter	Parameter value	Default value	Description
RFID read point alarm	Off On	Off	Enabling/disabling read-point related diagnostics messages.

1.6.2.2 Optical reading systems

MV400 optical reader

MV400 optical reader

The MV400 optical readers are specifically designed for the recognition and processing of numerous machine-readable codes, plain text as well as objects in industrial production.

The readers can be operated both in the controllers S7-300, S7-400, S7-1200 and S7-1500 as well as with various third-party controllers. The connection to the controller is via PROFINET IO (FB79/Ident profile), TCP, RS-232 or via communications modules.

The function blocks FB79 and the Ident profile allow simple programming using the SIMATIC S7 tools.

"Operating mode" parameter group

In this parameter group, you can configure the mode of the MV400 optical readers.

Table 1-230 Parameters in the "Operating mode" parameter group

Parameter	Parameter value	Default value	Description
Function block	FB79 Ident profile	FB79	With this parameter, you select the block: • FB79 Configuration of the optical reader with help of the "FB79" block. Compatible with the "VS130-2" reader; Compatible controllers: S7-300 and S7-400 • Ident profile Configuration of the optical reader with help of the Ident blocks. Compatible with all Ident devices; Compatible controllers: S7-300, S7-400, S7-1200 and S7-1500

MV500 optical reader

MV500 optical reader

The MV500 optical readers are specifically designed for the recognition and processing of numerous machine-readable codes, plain text as well as objects in industrial production.

The readers can be operated both in the controllers S7-300, S7-400, S7-1200 and S7-1500 as well as with various third-party controllers. The connection to the controller is via PROFINET IO (FB79/Ident profile), TCP, RS-232 or via communications modules.

The function blocks FB79 and the Ident profile allow simple programming using the SIMATIC S7 tools.

"Operating mode" parameter group

In this parameter group, you can configure the mode of the MV500 optical readers.

Table 1-231 Parameters in the "Operating mode" parameter group

Parameter	Parameter value	Default value	Description
Function block	FB79 Ident profile	FB79	With this parameter, you select the block: • FB79 Configuration of the optical reader with help of the "FB79" block. Compatible with "VS130-2" and "MV400" readers; Compatible controllers: S7-300 and S7-400 • Ident profile Configuration of the optical reader with help of the Ident blocks. Compatible with all Ident devices; Compatible controllers: S7-300, S7-400, S7-1200 and S7-1500

"Web Based Management" parameter group

In this parameter group you can start the Web Based Management.

Table 1-232 Parameters in the "Web-based management" parameter group

Parameter	Description
Web Based Management	Start Web Based Management of the reader. Web Based Management (WBM) offers extensive functions for configuring the reader. Note: WBM can only be started when either the PROFINET connection between CPU and reader has been established or the IP address stored in the project has been assigned to the reader. This means that the device name has been assigned and the TIA configuration must be loaded on the SIMATIC controller.

1.6.3 Distributed I/O

1.6.3.1 Distributed I/O systems

SIMATIC ET 200 - The right solution for all applications

SIMATIC ET 200 provides the most varied range of distributed I/O systems.

- Solutions for use in the control cabinet
- Solutions without control cabinet directly at the machine

Additionally, there are also components that can be used in explosive areas. SIMATIC ET 200 systems for construction without a control cabinet are contained in robust, glass-fibre reinforced plastic casing and are therefore shock-resistant, resistant to dirt and watertight.

Their modular design allows the ET 200 systems to be easily scaled and expanded in small steps. Fully-integrated auxiliary modules lower costs and also provide a wide range of possible applications. There are several combination possibilities available:

- Digital and analog I/OS
- Intelligent modules with CPU functions,
- Safety technology,
- Pneumatics,
- Frequency converters
- Various technology modules.

Communication via PROFIBUS and PROFINET, uniform engineering, clear diagnostic possibilities as well as optimal connection to SIMATIC controller and HMI devices vouch for the unique consistency provided by Totally Integrated Automation.

The following table provides an overview of I/O devices for use in the control cabinet:

I/O device	Properties
ET 200L	• Cost-effective digital block I/O • Digital electronic blocks up to 32 channels
ET 200M	• Modular design with standard modules from SIMATIC S7-300 • Failsafe I/O modules • Use in hazardous areas up to Zone 2, sensors and actuators up to Zone 1 • High level of plant availability, for example by plugging and unplugging during operation
ET 200MP	• Modular design with standard SIMATIC S7-1500 modules • Isochronous PROFINET IO with send clocks up to 250 µs • IO device with PROFINET IO interface • DP slave with PROFIBUS DP interface • Technology modules for counting, positioning, time-based IO
ET 200S	• Highly modular design with multiple conductor connections • Multifunctional due to a wide range of modules • I-device • Use in explosive areas (Zone 2)

I/O device	Properties
ET 200S COMPACT	• Highly modular design with multiple conductor connections • Multifunctional due to a wide range of modules • Use in explosive areas (Zone 2) • Integrated DE/DA
ET 200SP	• Compact modules • Permanent wiring with one or more wire connection • PROFINET interface with 3 ports • Isochronous PROFINET IO with the profiles PROFIsafe and PROFIenergy • I-device • Modules for counting, positioning, weighing and measuring • Fail-safe CPUs and modules
ET 200iSP	• Modular design, also possible with redundancy • Robust and intrinsically safe design • Use in explosive areas up to Zone 1/21; sensors and actuators even up to Zone 0/20 • High level of plant availability, for example by plugging and unplugging during operation

The following table provides an overview of I/O devices for use without a control cabinet:

I/O device	Properties
ET 200AL	• Simple installation in all locations, in the smallest spaces • Connection to PROFINET IO, PROFIBUS DP or ET 200SP • High degree of protection IP65/IP67 For temperatures from -25 °C to +55 °C, acceleration up to 5 g • Connection of sensors and actuators using M8 and M12 connector system
ET 200eco PN	• Cost-efficient, space-saving block I/OS • Digital modules up to 16 channels (also configurable) • Analog modules, IO-link master and load voltage distributor • PROFINET connection with 2-port switch in each module • Can be flexibly distributed via PROFINET in line or star shape directly within the plant

I/O device	Properties
ET 200eco	• Cost-effective digital block I/O • Flexible connection possibilities • Failsafe modules • High level of plant availability
ET 200pro	• Modular design with compact housing • Easy assembly • Multifunctional due to a wide range of modules • I-device • High level of plant availability due to plugging and unplugging during operation and permanent wiring • Comprehensive diagnostics
ET 200R	• Specially for use on robots • Assembled directly on the chassis • Resistant to weld spatter due to robust metal housing

See also

Documentation on ET 200L (http://support.automation.siemens.com/WW/view/en/1142908/0/en)

Documentation on ET 200S (http://support.automation.siemens.com/WW/view/en/1144348/0/en)

Documentation on ET 200M (http://support.automation.siemens.com/WW/view/en/1142798/0/en)

Documentation on ET 200pro (http://support.automation.siemens.com/WW/view/en/21210852/0/en)

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/en/28930789/0/en)

Documentation on ET 200R (http://support.automation.siemens.com/WW/view/en/11966255/0/en)

Documentation on ET 200eco PN (http://support.automation.siemens.com/WW/view/en/29999018/0/en)

Documentation on ET 200eco (http://support.automation.siemens.com/WW/view/en/12403834/0/en)

Documentation on ET 200SP (http://support.automation.siemens.com/WW/view/en/84133942/0/en)

Documentation on ET 200MP (http://support.automation.siemens.com/WW/view/en/86140384/en)

Documentation on ET 200AL (http://support.automation.siemens.com/WW/view/en/95242965/0/en)

1.6.3.2 Configuring HART variables

Introduction

Numerous HART field devices make available additional measured quantities (e.g. sensor temperature). These can be read if they are set accordingly in the field device configuration. Using the HART variables. it is possible to apply the set measured values directly from the field device into the I/O area of your automation system.

Regardless of the number of configured channels, a maximum of 8 HART variables can be assigned for HART modules and no more than 4 HART variables per channel. You assign the HART variables to a channel in the properties for the module ("HART variable settings" area). To do this, check the manual of the corresponding module.

Address assignment

By default, the HART modules occupy 16 input/output bytes (user data). If you configure HART variables, the modules occupy an additional 5 bytes for each HART variable.

If you use all 8 HART variables, the HART input module occupies a total of 56 input/output bytes (16 bytes + 8 x 5 bytes = 56 bytes).

The "None" configuration occupies no additional input bytes.

Configuration of HART variables

You can configure up to 4 HART variables for a channel

- PV (Primary Variable)
- SV (Secondary Variable)
- TV (Tertiary Variable)
- QV (Quaternary Variable)

CiR is a placeholder that reserves the address space for a HART variable. You must configure the HART variables you are not using with the "None" parameter.

Configuration of HART variables

Each HART variable occupies 5 bytes of input data and is structured as follows:

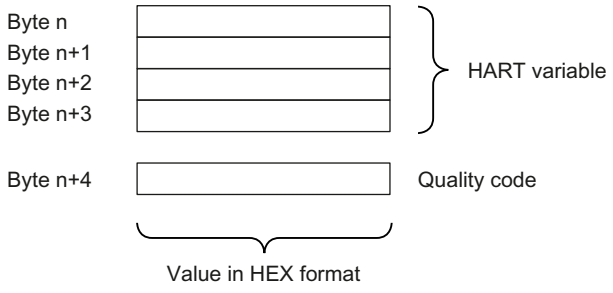

The quality code can accept different values depending on the module. To do this, check the manual of the corresponding module.

See also

Documentation for HART analog modules (http://support.automation.siemens.com/WW/view/en/22063748)

1.6.3.3 ET 200iSP

ET 200iSP Distributed I/O Station

Definition

The ET 200iSP distributed I/O station is a highly modular and intrinsically safe DP slave with degree of protection IP 30.

Area of application

The ET 200iSP distributed I/O station can be operated in potentially explosive atmospheres characterized by gas and dust:

Approval	ET 200iSP Station*	Inputs and outputs
ATEX	Zone 1, Zone 21	up to Zone 0, Zone 20 **
IECEx	Zone 2, Zone 22	up to Zone 0, Zone 20 **
* In combination with an appropriate enclosure ** for electronic module 2 DO Relay UC60V/2A: up to Zone 1, Zone 21		

The ET 200iSP distributed I/O station can, of course, also be used in the safety area.

You can insert almost any combination of ET 200iSP I/O modules directly next to the interface module that transfers the data to the DP master. This means you can adapt the configuration to suit your on-site requirements.

Every ET 200iSP consists of a power supply module, an interface module and a maximum of 32 electronic modules (for example digital electronics modules). Remember not to exceed the maximum current consumption.

Terminal modules and electronic modules

In principle, the ET 200iSP distributed I/O station consists of various passive terminal modules onto which you plug the power supply and the electronic modules.

The ET 200iSP is connected to PROFIBUS RS 485-IS by means of a connector on terminal module TM-IM/EM. Every ET 200iSP is a DP slave on the PROFIBUS RS 485-IS.

DP master

All ET 200iSP modules support communication with DP masters that are compliant with *IEC 61784-1:2002 Ed1 CP 3/1* and operate with "DP" transmission protocol (DP stands for distributed peripherals or distributed I/O).

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Assigning the channel and IEEE tag

Properties

Analog electronic modules 4 AI I 2WIRE/HART, 4 AI I 4WIRE/ HART and 4 AO I HART support up to four IEEE tags.

The process input image (PII) provides up to 20 bytes per module for the IEEE tags. Thus, four blocks of 5 bytes each are available for the four IEEE tags within the PII.

Requirements

The HART field device must support the assigned number of IEEE tags.

Assigning IEEE tags

You assign the IEEE tags of the field devices to any one of the four blocks in the PII.

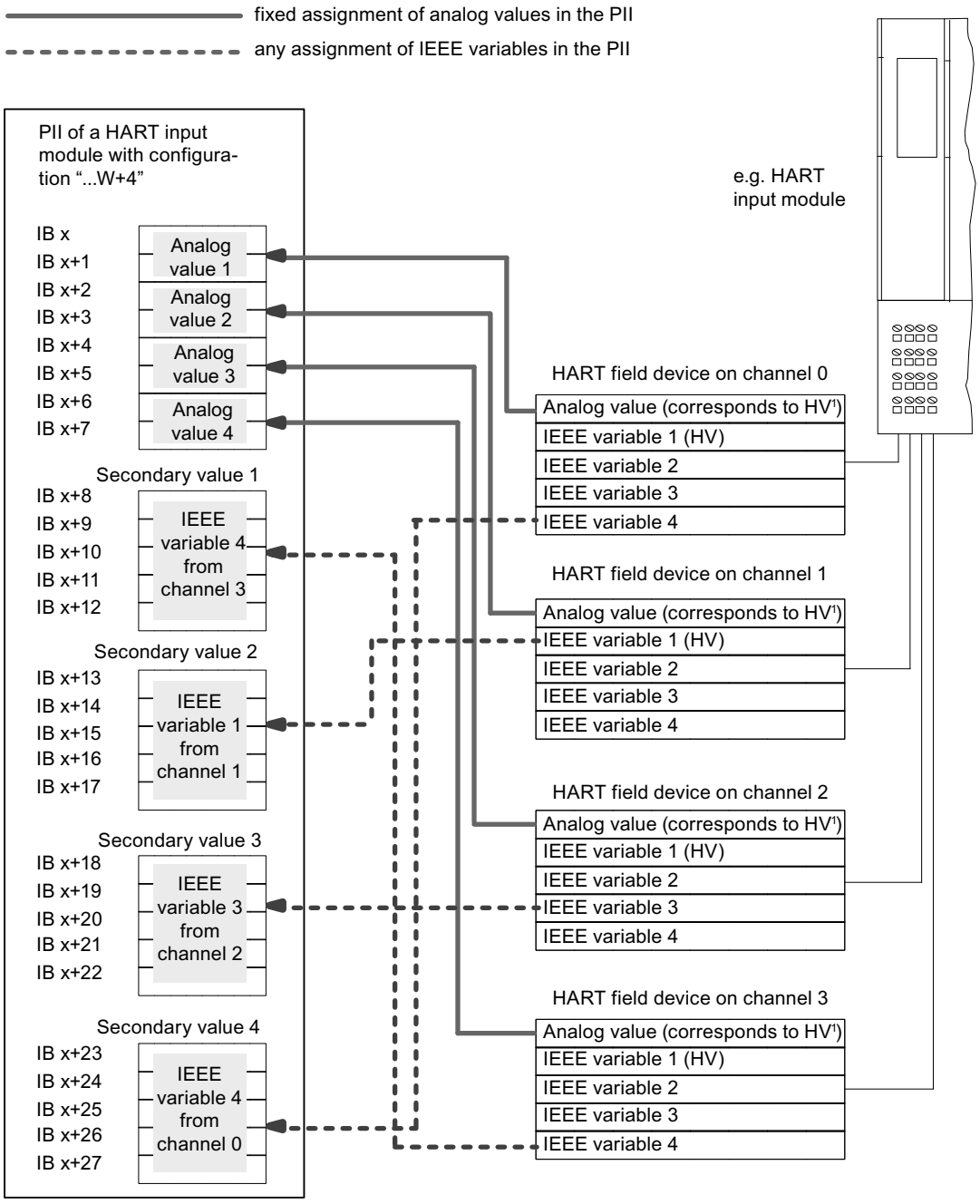

[1] HV = main variable

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Assigning parameters to reference junctions for thermocouples

Compensation of the reference junction temperature

There are various ways of obtaining the reference junction temperature in order to get an absolute temperature value from the temperature difference between the reference junction and the measuring point.

Table 1-233 Compensation of the reference junction temperature

Option	Explanation	Reference junction parameters
No compensation	You record not only the temperature of the measurement point. The temperature of the reference junction (transition from Cu line to thermocouple line) also affects the thermo-electromotive force. The measured value then includes an error.	None
Use of a Pt100 Climatic Range resistance thermometer to record the reference junction temperature (best method)	You can record the reference junction temperature using a resistance thermometer (Pt100 Climatic Range). If parameterized accordingly, this temperature value is distributed to the 4 AI TC modules in the ET 200iSP where it is offset against the temperature value obtained at the measuring location. Number of reference junctions: 2	The parameter assignment of the IM 152 and the 4 AI TC must be coordinated: • 4 AI RTD assigned parameters for Pt100 climatic range in correct slot; • 4 AI TC: Reference junction : "yes"; select reference junction number "1" or "2" • IM 152-1:Assignment of the reference junction to a slot with 4 AI RTD; channel selection;
Internal compensation 4 AI TC	The TC sensor module (temperature sensor) is mounted onto the terminals of terminal module EM 4 AI TC. The temperature sensor reports the temperature of the terminals to the 4 AI TC. This value is then calculated together with the measured value from the channel of the electronic module.	• 4 AI TC: Reference junction number "internal"

Compensation by means of a resistance thermometer at the 4 AI RTD

If thermocouples that are connected to the inputs of the 4 AI RTD have the same reference junction, compensate by means of a 4 AI RTD.

For both channels of the 4 AI TC module, you can select "1", "2" or "internal" as the reference junction number. If you select "1" or "2", the same reference junction (RTD channel) is always used for all four channels.

Setting parameters for the reference junction

You set the reference junctions for the 4 AI TC electronic modules by means of the following parameters:

Table 1-234 Reference junction parameters

Parameter	Module	Range of values	Explanation
Slot reference junction 1 to slot 2	IM 152	none, 4 to 35	With this parameter, you can assign up to 2 slots (none, 4 to 35), on which the channels for reference temperature measurement (calculating the compensation value) are located.
Input reference junction 1 to 4 input reference junction	IM 152	RTD on channel 0 RTD on channel 1 RTD on channel 2 RTD on channel 3	This parameter allows you to set the channel (0/1/2/3) for measuring the reference temperature (calculation of the compensation value) for the assigned slot.
Reference junction E0 to reference junction E3	4 AI TC	None yes	This parameter allows you to enable the use of the reference junction.
Reference junction number	4 AI TC	1 2 Internal	This parameter allows you to assign the reference junction (1, 2) that contains the reference temperature (compensation value).

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Fundamentals of Time Stamping

Properties

Time stamping is possible with the IM 152 in customer applications using FB 62 (FB TIMESTMP).

Principle of operation

A modified input signal is assigned a time stamp and stored in a buffer (data record). If time stamped signals exists or a data record is full, a hardware interrupt is generated to the DP master. The buffer is evaluated with "Read data record". Special messages are generated for events that influence the time stamping (communication with the DP master interrupted, frame failure of time master, ...).

Parameter Assignment

With the parameter assignment you define which IM 152 user data will be monitored. For the time stamping these are digital inputs that are monitoring for signal changes.

Parameter	Setting	Description
Time stamping	• disabled • enabled	Activate the time staming for the channels of the electronics module 8 DI NAMUR.
Edge evaluation incoming event	• rising edge • falling edge	Determine the type of signal change that will be time-stamped.

Counting

Count properties

Counting functions

The 8 DI NAMUR electronics module has configurable counting functions:

- 2 x 16-bit up counters (standard counting function) or
- 2 x 16-bit down counters (standard counting function) or
- 1 x 32-bit down counter (cascading counter function)
- Setting a setpoint with the PIQ
- GATE function
- You can configure the control signals of the counters:
  - Configuration channel 0..1: "Counter", channel 2..7: "DI": Two counters are configured. The control signals of the counters are stored in the PIQ (process image output).
  - Configuration channel 0..1: "Counter", channel 2..7: "Control": Two counters are configured. The control signals of the counters are stored in the PIQ (process image output). They are also controlled by the digital inputs of the 8 DI NAMUR.

See also

Principle of operation (Page 3198)

Configuring counters (Page 3201)

Assigning parameters to counters (Page 3203)

Principle of operation

16-bit up counters (standard counting function)

The counting range is 0 to 65,535.

With each count pulse at the digital input, the count is incremented by 1. Once the count limit is reached, the counter is reset to 0 and it counts up again from this value.

If there is counter overflow, the corresponding output is set in the PII.

A positive edge of the *Reset output* control signal resets the output in the PII. This does not affect the current count value.

In 16-bit up counting operations, the system does not set any outputs in the PIQ. These are always reset.

The positive edge of the *Reset counter* control signal sets the counter to 0 and resets the set counter output.

The *GATE* control signal pauses the counting on a positive edge. Count pulses are processed at the digital input again, but only at the negative edge. The *Reset counter* control signal is also effective when *GATE* is active.

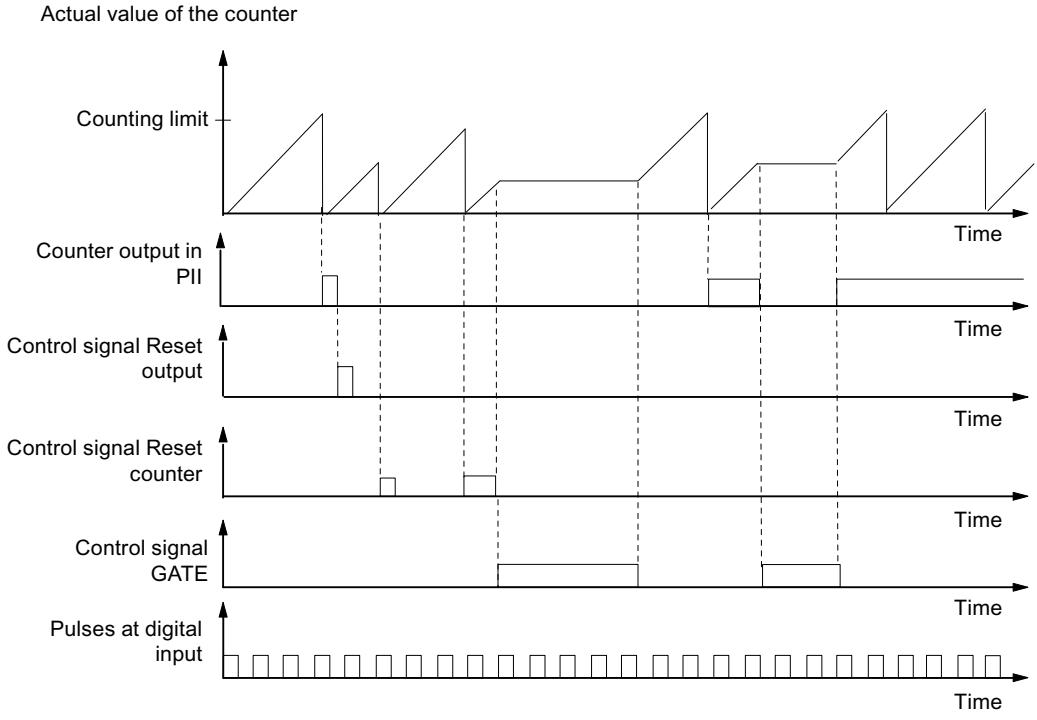

16-bit down counters (periodic counting function)

The maximum counting range is always 65,535 to 0.

When the counter is started, the actual value is set to the selected setpoint. Each counted pulse reduced the actual value by 1. Once the actual value reaches 0, the corresponding output in the PII is turned on and the actual value is set to the selected setpoint. The counter then counts down from this value.

The positive edge of the *Reset counter* control signal resets the selected setpoint and the corresponding output in the PII.

A positive edge of the *Reset output* control signal resets the output in the PII. This does not affect the current count value.

The *GATE* control signal pauses the counting on a positive edge. At the same time, the assigned output in the PII is reset. Count pulses are processed at the digital input again, but only at the negative edge. The *Reset output* and *Reset counter* control signals are also effective when *GATE* is active.

The setpoint of the counter is set and changed using the PIQ. The setpoint is adopted on a positive edge of the *Reset counter* control signal or when the counter has reached zero.

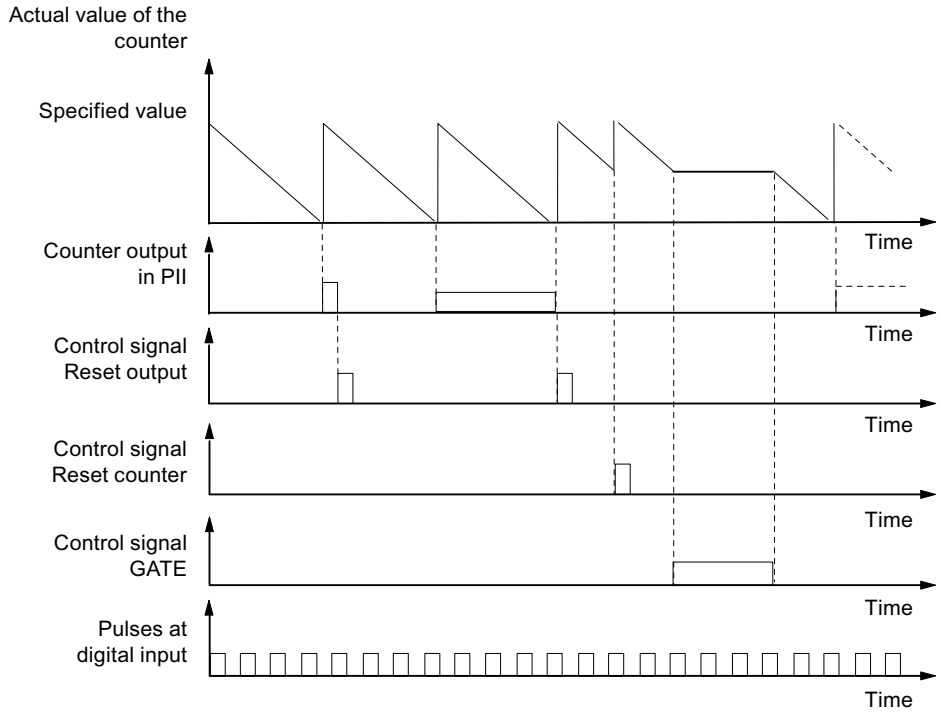

32-bit down counter (cascading counter function)

The maximum counting range is always 4294967295 to 0.

The principle of operation is identical to that of the 16-bit down counter. Channel 1 has no function.

See also

Count properties (Page 3198)

Configuring counters

Procedure

1. Using the mouse, pull module 8 DI Namur from the hardware catalog into distributed I/O station ET 200iSP.
2. Select the required configuration (channel 0..1: "Counter", channel 2..7: "DI" or "Control"). In the module properties (inspector window), you can find this setting under "Parameters > Inputs > Configuration".

Configuration channel 0..1: "Counter", channel 2..7: "DI"

- Assignment of the digital inputs on the electronic module 8 DI NAMUR

Table 1-235 Assignment of digital inputs for channel 0..1: "Counter", channel 2..7: "DI":

Digital input	Terminal	Assignment
Channel 0	1, 2	Counter 1
Channel 1	5, 6	Counter 2 (does not apply to 32-bit down counters)
Channel 2	9, 10	Digital input 2
Channel 3	13, 14	Digital input 3
Channel 4	3, 4	Digital input 4
Channel 5	7, 8	Digital input 5
Channel 6	11, 12	Digital input 6
Channel 7	15, 16	Digital input 7

- Assignment of the process image input (PII)

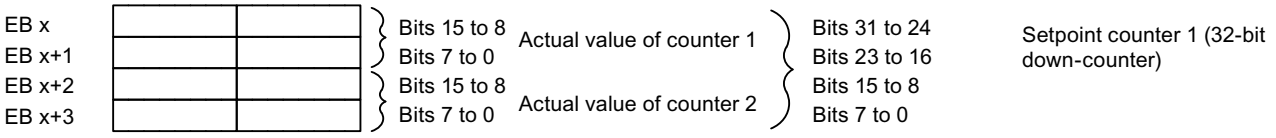

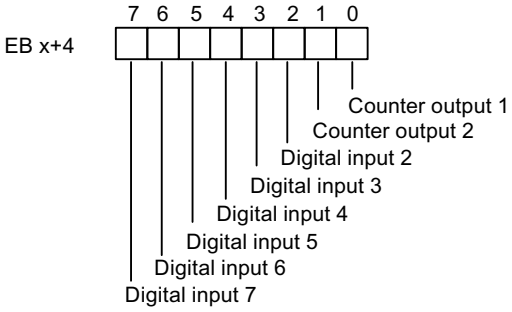

S7 format

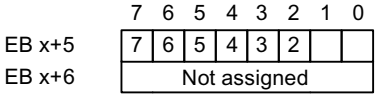

Value status for channels 2 to 7:
1_B: Input signal is valid
0_B: Input signal is invalid

- Assignment of the process image output (PIQ)

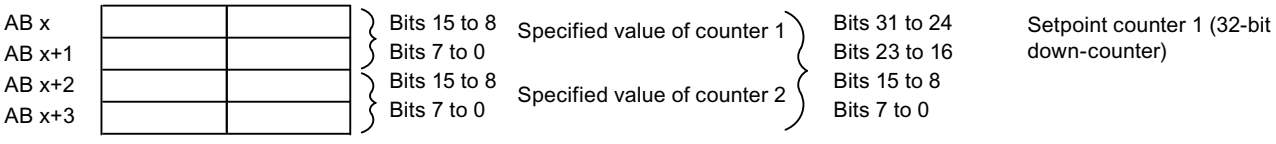

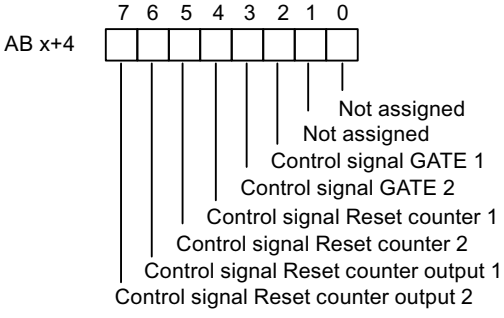

Configuration channel 0..1: "Counter", channel 2..7: "CONTROL"

With this configuration, you can also control the counters over the digital inputs.

- Assignment of the digital inputs on electronic module 8 DI NAMUR
  For further information on input assignments, refer to the technical data for electronic module 8 DI NAMUR.

Table 1-236 Assignment of the digital inputs for 2 Count/ 6 Control

Digital input	Terminal	Assignment
Channel 0	1, 2	Counter 1
Channel 1	5, 6	Counter 2 (does not apply to 32-bit down counters)
Channel 2	9, 10	control signal *GATE 1*
Channel 3	13, 14	control signal *GATE 2*
Channel 4	3, 4	control signal *Reset counter 1*
Channel 5	7, 8	control signal *Reset counter 2*
Channel 6	11, 12	control signal *Reset counter output 1*
Channel 7	15, 16	control signal *Reset counter output 2*

- Assignment of the process image input (PII)
  Assignment is identical to configuration 0..1: "Counter", channel 2..7: "DI".
- Assignment of the process image output (PIQ)
  Assignment is identical to configuration 0..1: "Counter", channel 2..7: "DI".

See also

Count properties (Page 3198)

Assigning parameters to counters

Parameters for the counting function

Only those parameters that are relevant for the counters are explained below. These belong to the parameters of electronic module 8 DI NAMUR and depend on the selected configuration:

Table 1-237 Parameters for the counters

Parameter	Setting	Description
Sensor type counter inputs	• Channel disabled • NAMUR sensor • Single contact, no load resistance	Select the sensor for the respective counter of channels 0 or 1.
Mode for counter 1	• Standard counting function • Periodic counting function • Cascaded counting function	Select the mode for counter 1.
Mode for counter 2	• Standard counting function • Periodic counting function • Cascaded counting function	Select the mode for counter 2. This parameter is not relevant if you have set the "Mode for counter 1" parameter to "Cascaded counter function".

See also

Count properties (Page 3198)

Frequency measurement

Frequency measurement properties

Properties

The electronic module 8 DI NAMUR allows the frequencies to be measured on channel 0 and 1:

- 2 frequency meters from 1 Hz to 5 kHz
- Configurable metering window (GATE)
- The signals of the frequency meter are read in by means of the digital inputs of the electronic module.

See also

Principle of operation (Page 3204)

Configuring frequency meters (Page 3205)

Assigning parameters for the frequency meters (Page 3207)

Principle of operation

Frequency measurement

The signal frequencies are identified from the input signals of channel 0 or 1 of the electronic module. To calculate the frequency the signals are measured within a configurable gate.

The frequency is displayed as 16-bit value in fixed-point format and transferred to the PII.

The frequency meter calculates the frequency according to the follow formula:

$$\text{Frequency [Hz]} = \frac{\text{Number of rising edges at digital input}}{\text{Measuring window [s]}}$$

Exceeding the input frequency

If the input frequency exceeds 5kHz, $7FFF_H$ is reported as actual value. If the input frequency is above approx. 8 kHz it is no longer possible to display correct actual values.

See also

Frequency measurement properties (Page 3204)

Configuring frequency meters

Procedure

1. Using the mouse, pull module 8 DI Namur from the hardware catalog into distributed I/O station ET 200iSP.
2. Select the required configuration (channel 0..1: "Trace", channel 2..7: "DI"). In the module properties (inspector window), you can find this setting under "Parameters > Inputs > Configuration".

Configuration 0..1: "Trace", channel 2..7: "DI"

Assignment of the digital inputs on electronic module 8 DI NAMUR

Digital input	Terminal	Assignment
Channel 0	1, 2	Frequency counter 1
Channel 1	5, 6	Frequency counter 2
Channel 2	9, 10	Digital input 2
Channel 3	13, 14	Digital input 3
Channel 4	3, 4	Digital input 4
Channel 5	7, 8	Digital input 5
Channel 6	11, 12	Digital input 6
Channel 7	15, 16	Digital input 7

Assignment of process image input (PII) for configuration of channel 0..1: "Trace", channel 2..7: "DI"

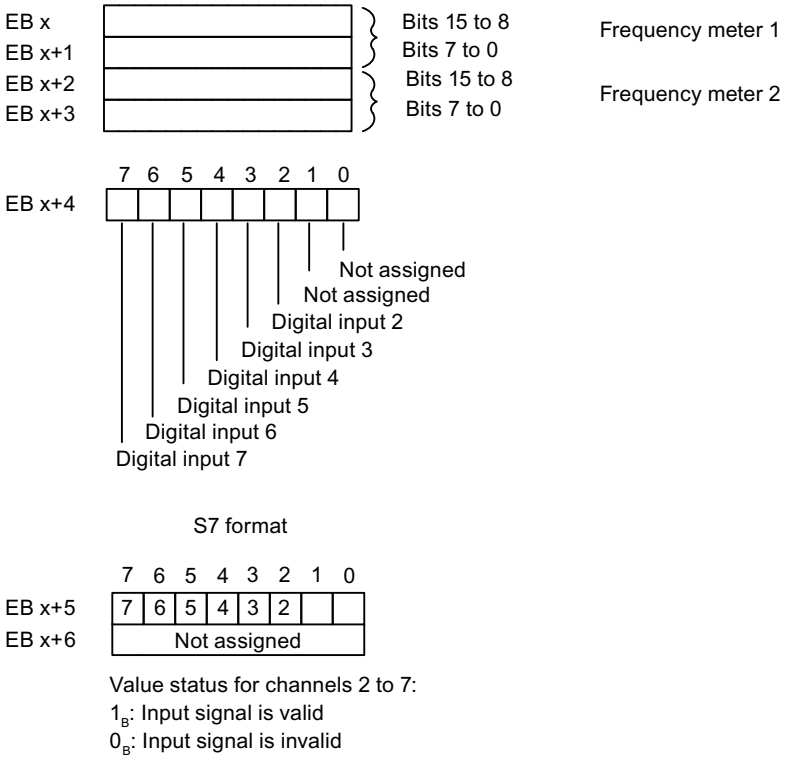

Assignment of the process image output (PIQ): The PIQ is not assigned.

See also

Frequency measurement properties (Page 3204)

Assigning parameters for the frequency meters

Parameters for frequency meter

Only those parameters that are relevant for the frequency meters are explained below. These are part of the parameters of electronic module 8 DI NAMUR.

Table 1-238 Parameters for the frequency meters

Parameter	Setting	Description
Sensor type frequency inputs	• Channel disabled • NAMUR sensor • Single contact, no load resistance	Select the sensor for the relevant frequency meter for channel 0 or 1.
Measuring window (GATE)	• 50 ms • 200 ms • 1 s	Select the required measuring window for channel 0 or 1. To achieve the highest possible accuracy when metering frequencies, remember the following rules: • High frequencies (> 4 kHz): Set a low measuring window (50 ms) • Variable/medium frequencies: set medium measuring window (200 ms) • Low frequencies (< 1 kHz): Set a high measuring window (1 s)

See also

Frequency measurement properties (Page 3204)

1.6.3.4 ET 200eco PN

ET 200eco PN Distributed I/O Device

Definition

The ET 200eco PN distributed I/O device is a compact PROFINET IO device in degree of protection IP 65/66 or IP 67 and UL Enclosure Type 4x, Indoor use only.

Field of application

The fields of application of the ET 200eco PN are derived from its special properties.

- A robust design and degree of protection IP 65/66 or IP 67 make the ET 200eco PN distributed I/O device suitable in particular for use in rugged industrial environments.
- The compact design of the ET 200eco PN is particularly favorable for applications in confined areas.
- The easy handling of ET 200eco PN facilitates efficient commissioning and maintenance.

Properties

The ET 200eco PN has the following properties:

- Integrated switch with 2 ports
- Supported Ethernet services:
  - ping
  - arp
  - Network diagnostics (SNMP)
  - LLDP
- Interrupts
  - Diagnostics interrupts
  - Maintenance interrupts
- Port diagnostics
- Isochronous real-time communication
- Prioritized startup
- Device replacement without programming device
- Media redundancy
- Connection to intelligent sensors/actuators via IO link master interface module.

IO Controller

The ET 200eco PN can communicate with all IO Controllers that conform to IEC 61158.

ET 200eco PN can be configured on a CPU with advanced diagnostics.

See also

Documentation on ET 200eco PN (http://support.automation.siemens.com/WW/view/en/29999018)

Parameter description analog input

Group diagnostics

You can generally enable and disable the diagnostics function of the device with this parameter.

The "Fault" and "Parameter assignment error" diagnostics functions are always independent of the group diagnostics.

Diagnostics, missing 1L+

If you enable this parameter, the check for missing supply voltage is enabled.

Diagnostics, sensor supply short circuit

If you enable this parameter, a diagnostics event is generated if a short-circuit of the sensor supply to ground is detected and the channel is enabled. The sensor supply is monitored for connectors X1, X3, X5 and X7. It is not possible to differentiate which connector has experienced the sensor short circuit.

Interference frequency suppression

With this parameter, you set the integration time of the device, based on the selected interference frequency. Select the frequency of the supply voltage used. Interference frequency suppression **Off** means 500 Hz, which corresponds to an integration time of 2 ms for a measurement channel.

Temperature unit

Specify the unit of the temperature measurement here.

Measurement type (channel-wise)

With this parameter, you set the measurement type, for example, voltage. For any unused channels, you must select the **disabled** setting. For a disabled channel, the conversion time and integration time of the channel = 0 s and the cycle time is optimized.

Measuring range

With this parameter, you set the measuring range of the selected measurement type.

Temperature coefficient (for RTD, thermoresistor)

The correction factor for the temperature coefficients (α-value) indicates by what extent the resistance of specific material changes relatively if the temperature increases by 1 °C.

The α-values conform to EN 60751, GOST 6651, JIS C 1604 and ASTM E-1137.

The temperature coefficient depends on the chemical composition of the material.

Smoothing

Smoothing of the analog values produces a stable analog signal for further processing. The smoothing of analog values is useful when handling wanted signals (measured values) with a slow rate of change, for example, temperature measurements.

The measured values are smoothed by digital filtering. To achieve smoothing, the device generates a mean value from a specified number of converted (digitized) analog values.

You assign a maximum of four levels for the smoothing (none, weak, medium, strong). The level determines the number of module cycles, from which the mean value is generated.

The stronger the smoothing, the more stable the smoothed analog value and the longer it takes until the smoothed analog value is applied following a signal change (see the example below).

The figure below shows the number of cycles a module requires to apply the smoothed analog value at almost 100% after a step response, based on the smoothing function settings. The figure applies to all signal changes at the analog input. The smoothing value defines the number of cycles a module requires to reach 63% of the end value of the changed signal.

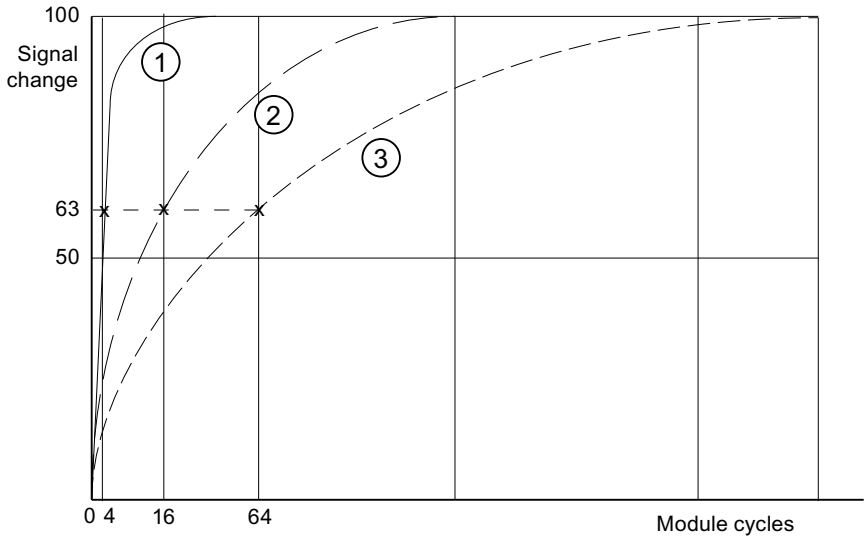

① Smoothing, weak
② Smoothing, medium
③ Smoothing, strong

Diagnostics, wire break

When this parameter is enabled, the **Wire break** diagnostics event is generated when a wire break is detected.

Observe the rules outlined below to handle a wire break in the 1 to 5 V and 4 to 20 mA measuring ranges:

Parameter	Event	Measured value	Explanation
Enable wire break[1]	Wire break	$7FFF_H$	Diagnostics, **wire break**
Wire break disabled[1] Underflow enabled	Wire break	8000_H	Measured value after leaving the underrange Diagnostic message **Lower limit value undershot**
Wire break disabled[1] Underflow disabled	Wire break	8000_H	Measured value after leaving the underrange
[1] Measuring range limits for wire break detection and measuring range undershoot detection: • 1 to 5 V: At 0.296 V • 4 to 20 mA: At 1.185 mA			

Diagnostics, underflow

If you enable this parameter, the **Underflow** diagnostics event is generated when the measured value reaches the underflow range.

Diagnostics, overflow

If you enable this parameter, the **Overflow** diagnostics event is generated when the measured value reaches the overflow range.

Reference junction for thermoresistor (TC)

A difference in temperature between the measuring point and the free ends of the thermocouple (terminal point) generates a voltage between the free ends, namely the thermoelectric voltage. The value of this thermoelectric voltage is determined by the temperature difference between the measuring point and the free ends and by the type of material combination of the thermocouple. Since a thermocouple always measures a temperature difference, the free ends at the reference junction must be maintained at a known temperature in order to determine the temperature of the measuring point.

If you specify **Internal compensation**, the temperature of the measuring point in the housing of the I/O device is measured. With the **External compensation** setting, you can connect a compensation box in series in order to increase the accuracy of the temperature measurement.

Parameter description analog output

Group diagnostics

You can generally enable and disable the diagnostics function of the device with this parameter.

The "Fault" and "Parameter assignment error" diagnostics functions are always independent of the group diagnostics.

Diagnostics, missing 1L+

If you enable this parameter, the check for missing supply voltage is enabled.

Diagnostics, sensor supply short circuit

When this parameter is enabled, the system generates a diagnostics event if it detects a short-circuit of the sensor supply to ground. This diagnostics function is activated when the group diagnostics function is enabled.

Response to CPU/Master STOP

Select how the module's outputs will respond to a CPU STOP:

- Shut down
  The I/O device goes to the safe state. The process image output is deleted (=0).
- Keep last value
  The I/O device retains the last value to be output before STOP.
- Substitute value
  The I/O device outputs the value for the channel set beforehand.

Note

Make sure that the plant is always in a safe state if "Keep last value" is selected.

Type of output

With this parameter, you set the output type, for example, voltage. For any unused channels, select the **disabled** setting. For a disabled channel, the conversion time and integration time of the channel = 0 s, and the cycle time is optimized.

Output range

With this parameter, you set the output range of the selected output type.

Diagnostics, wire break (in current mode)

When this parameter is enabled, the **Wire break** diagnostics event is generated when a wire break is detected. This diagnostics event cannot be detected in the zero range.

Diagnostics, short circuit (in voltage mode)

If you enable this parameter, a diagnostics event is generated in the event of a short circuit in the output line. This diagnostics event cannot be detected in the zero range.

Diagnostics, overload

If you enable this parameter, the diagnostics event is generated in the event of an overload.

Substitute values

With this parameter, you enter a substitute value that the module is to output in CPU-STOP mode. The substitute value must be in the nominal range, overrange, or underrange.

1.6.3.5 ET 200SP

ET 200SP distributed I/O system

Definition

The ET 200SP distributed I/O system is a scalable, highly flexible distributed I/O system for connection of process signals to a central controller via a field bus.

Application area

The ET 200SP is a multi-functional distributed I/O system for various fields of application. The scalable design allows you to configure the system exactly to the specific requirements on location.

The ET 200SP is approved for degree of protection IP 20 and for installation in a control cabinet.

Structure

The ET 200SP is mounted on a mounting rail and comprises:

- An interface module which can communicate with all IO controllers that conform to the PROFINET standard IEC 61158
- Up to 32 I/O modules which can be inserted on passive BaseUnits in any combination
- A server module that completes the design of the ET 200SP.

ET 200SP with own CPU module

ET 200SP with own CPU

The ET 200SP can be equipped with its own CPU, for example with the CPU 1510SP-1 PN or CPU 1512SP-1 PN.

You can connect up to 65 modules of the S7-1500 automation system to this CPU.

Note the configuration rules of the ET 200SP (http://support.automation.siemens.com/WW/view/en/58649293).

Interface module parameters

Status bytes

Status bytes

If you enable the "Status bytes" option, 4 bytes of input data are reserved for the status of the supply voltage of each I/O module.

	7	6	5	4	3	2	1	0
Byte 0	8	7	6	5	4	3	2	1
Byte 1	16	15	14	13	12	11	10	9
Byte 2	24	23	22	21	20	19	18	17
Byte 3	32	31	30	29	28	27	26	25

I/O module slots

Bit = 0: Load voltage missing or I/O module not present

Bit = 1: Load voltage and I/O module present

Note

An inserted or missing server module always reports for the slot bit = 0.

Group diagnostics, missing supply voltage L+

Group diagnostics, missing supply voltage L+

This diagnostics is a group diagnostics that covers the supply voltage status of all I/O modules of a potential group which are defined by BaseUnits with incoming power supply (light-colored BaseUnit BU...D).

The group diagnostics is formed from the states of the supply voltage of the inserted I/O modules within the potential group.

The group diagnostics does not depend on the "Missing supply voltage L+" parameter of the I/O modules being enabled.

The server module does not influence the missing supply voltage L+ group diagnostics.

Requirements for the correct operation of the group diagnostics for missing supply voltage L+:

- I/O modules or BU covers must be inserted on the light-colored and dark-colored BaseUnits. If no I/O module is inserted on a light-colored BaseUnit, the start of this potential group will not be detected by the interface module; the I/O modules of this potential group will thus belong to the previous potential group. A supply voltage L+ group error will then be assigned to the wrong potential group.
  When an I/O module is inserted on the light-colored BaseUnit, the interface module detects the new potential group, re-evaluates the status, and reports a new group diagnostics in the case of an error.
- The server module must be inserted.
  The server module itself does not influence the missing supply voltage L+ group diagnostics.

Configuration control with ET 200SP

Operating principle

Configuration control allows you to operate various real configurations (options) with a single configuration of the distributed I/O device ET 200SP.

Configuration control provides you with the option of configuring the ET 200SP distributed I/O device with its maximum configuration and nevertheless operating it with missing modules. If missing modules are retrofitted later, no new configuration is required and the hardware configuration does not have to be reloaded either.

Using control data record 196, which is transferred to the interface module in the user program, you define a current preset configuration.

- The configured module is not present on a slot.
  - A BU cover can be inserted in this slot instead of the configured I/O module. As there is no configured module on the slot, the term "Configuration control with empty slots" is also used.
  - The module that is configured to the right of the missing module can be inserted on this slot instead of the configured module. The missing module makes the actual configuration appear pushed together. As the configured module is missing but no gap arises in the configuration, this is also referred to as a "Configuration without empty slots".
- The configuration is extended by an already configured module.
  - In the case of configuration control with empty spaces, you extend the configuration by inserting the configured module on the corresponding empty slot.
  - In the case of configuration control without empty spaces, insert the configured module on the right-hand side next to the last module of the ET 200SP.

Requirement

The CPU startup parameter "Compare preset to actual configuration" is set to Startup even if mismatch (default setting). This setting is also selected for the startup parameters of the individual modules of the ET 200SP.

Enabling configuration control

In the properties of the interface module under Module parameters > General > Configuration control, select "Enable reconfiguration of device via user program". This activates configuration control.

Control data record 196 for ET 200SP

The figure below shows the data block 196 for the configuration control of an ET 200SP with four modules.

The value "12" is in the "block_length" element.

If you configure an ET 200SP in STEP 7 with more modules, the data block will be longer. The data record for the maximum configuration with 65 modules is 134 bytes long (configuration with PN interface module).

There are two bytes in the data record for each module.

The positions of these two bytes in the data record each code a module in the original configuration with STEP 7:

- "slot_1" and "info_slot_1" (bytes 4 and 5 in the data record, see figure below) correspond to the module in slot 1 in configuration with STEP 7.
- "slot_2" and "info_slot_2" (bytes 6 and 7) correspond to the module in slot 2 in configuration with STEP 7.
- "slot_3" and "info_slot_3" (bytes 8 and 9) correspond to the module in slot 3 in configuration with STEP 7.
- etc.

"slot_x" byte

The current slot is coded by the figure that is assigned to "slot_x" (by its value). Examples:

- The value "1" in byte 4 means you are assigning the module originally inserted in slot 1 to slot 1 in the current configuration (slot_1 = 1).
- The value "2" in byte 4 means you are assigning the module originally inserted in slot 1 to slot 2 in the current configuration (slot_1 = 2).
- The value "3" in byte 4 means you are assigning the module originally inserted in slot 1 to slot 3 in the current configuration (slot_1 = 3).
- etc.
- The value "1" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 1 in the current configuration (slot_2 = 1).
- The value "2" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 2 in the current configuration (slot_2 = 2).
- The value "3" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 3 in the current configuration (slot_2 = 3).
- etc.

If a BU cover can be inserted instead of a module, code this by adding 128 to the slot (bit 7 of the "slot_x" byte is set). Examples:

- The value "129" in slot_1 means you are also assigning the module originally inserted in slot 1 to slot 1 in the current configuration. A BU cover can also be used instead of this module. In the real plant configuration, either the module or a BU cover is inserted.
- The value "130" in slot_1 means you are assigning the module originally inserted in slot 1 to slot 2 in the current configuration. A BU cover can also be used instead of this module. In the real plant configuration, either the module or a BU cover is inserted.
- The value "131" in slot_1 means you are assigning the module originally inserted in slot 1 to slot 3 in the current configuration. A BU cover can also be used instead of this module. In the real plant configuration, either the module or a BU cover is inserted.

"info_slot_x" byte

If a new potential group is opened with the module, assign the "info_slot_x" byte the value 1 (bit 0 of the byte is set). Examples:

- The value "1" in the "info_slot_2" byte means that a new potential group is opened with module 2.
- The value "1" in the "info_slot_3" byte means that a new potential group is opened with module 3.
- The value "1" in the "info_slot_4" byte means that a new potential group is opened with module 4.

Exception: A new potential group is automatically assigned to the first module in the original configuration in STEP 7. This is not coded in the data record. You can enter any value in "info_slot_1".

You can choose any name for the components of the control data record (for example "slot_1").

Example of control data record 196 for ET 200SP

The figure below shows control data record 196 for an ET 200SP with four modules.

The module inserted in slot 2 in the configuration in STEP 7 can also be inserted in slot 2 in this configuration. It can also be inserted in slot 2 of a BU cover. Otherwise, nothing has changed compared to the original configuration.

Record_196_ET_200SP

	Name	Data type	Default value
1	block_length	USInt	12
2	block_ID	USInt	196
3	version	USInt	2
4	subversion	USInt	0
5	slot_1	USInt	1
6	info_slot_1	USInt	0
7	slot_2	USInt	130
8	info_slot_2	USInt	0
9	slot_3	USInt	3
10	info_slot_3	USInt	0
11	slot_4	USInt	4
12	info_slot_4	USInt	0

Addressing the interface module using the HW identifier

To transfer data record 196 with the instruction WRREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Readback data record 197 for ET 200SP

Readback data record 197 is used to read the actual configuration of a station (in this case, of an ET 200SP).

This readback data record allows you to check the real configuration of the ET 200SP (actual configuration). The readback data record for each configured module specifies whether or not it is actually available.

- The value "1" means that the correct module is inserted in the correct slot.
- The value "0" codes all other possibilities (wrong module, empty slot, BU cover).

Configuration details:

The configuration of the data block corresponds to the original configuration of the ET 200SP with STEP 7.

There are two bytes in the data record for each module. The position of these two bytes in the data record corresponds to the position of a module in the original configuration with STEP 7.

Sequence of bytes:

- "status_slot_1" and "reserve_1" (bytes 4 and 5 in the data record) correspond to the module in slot 1 in the configuration,
- "status_slot_2" and "reserve_2" (bytes 6 and 7) correspond to the module in slot 2 in the configuration
- "status_slot_3" and "reserve_3" (bytes 8 and 9) correspond to the module in slot 3 in the configuration,
- etc.

Example

The original configuration in STEP 7 has been changed by a control data record 196 (see example above): In the modified configuration, module 2 can either be inserted in slot 2 or be replaced by a BU cover.

The figure below shows readback data record 197 which ET 200SP outputs to indicate that there is a module in slot 2: The "status_slot_2" byte has the value "1".

The other modules are also available and are inserted in the correct slots.

Record_197_ET_200SP

	Name	Data type	Default value
1	block_length	USInt	12
2	block_ID	USInt	197
3	version	USInt	2
4	subversion	USInt	0
5	status_slot_1	USInt	1
6	reserve_1	USInt	0
7	status_slot_2	USInt	1
8	reserve_2	USInt	0
9	status_slot_3	USInt	1
10	reserve_3	USInt	0
11	status_slot_4	USInt	1
12	reserve_4	USInt	0

The figure below displays the readback data record 197 which ET 200SP outputs to indicate that a BU cover is being used in slot 2: The "status_slot_2" byte has the value "0".

The other modules are available and are inserted in the correct slots.

Record_197_ET_200SP			
	Name	Data type	Default value
1	block_length	USInt	12
2	block_ID	USInt	197
3	version	USInt	2
4	subversion	USInt	0
5	status_slot_1	USInt	1
6	reserve_1	USInt	0
7	status_slot_2	USInt	0
8	reserve_2	USInt	0
9	status_slot_3	USInt	1
10	reserve_3	USInt	0
11	status_slot_4	USInt	1
12	reserve_4	USInt	0

Reading readback data record 197

You can read readback data record 197 from the ET 200SP with the instruction RDREC. RDREC operates asynchronously. If you call RDREC in the startup OB, you must call the instruction multiple times using a loop until the "BUSY" or "DONE" output parameter indicates that the data record has been read.

To read data record 197 with the instruction RDREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Additional information and examples

Specific examples of configuration control can be found in this application description (http://support.automation.siemens.com/WW/view/en/29430270).

Further information on ET 200SP can be found in the manuals IM 155-6 PN (http://support.automation.siemens.com/WW/view/en/73184046) and IM 155-6 DP (http://support.automation.siemens.com/WW/view/en/73098660)

You will find a library of the data records and other application examples here:

Templates for the data records (http://support.automation.siemens.com/WW/view/en/29430270)

See also

Configuration control with ET 200AL (Page 3262)

Configuration control for ET 200SP with integrated ET 200AL modules (Page 3220)

Configuration control for ET 200SP with integrated ET 200AL modules

ET 200SP with integrated ET 200AL modules

Configuration control for ET 200SP and ET 200AL is described in separate help topics. See the links under "See also".

The steps outlined also apply to configuration control of an ET 200SP with integrated AL modules. The procedure is different for control data record 196 and readback data record 197.

The following help text describes control data record 196 and the readback data record 197 for an ET 200SP expanded with ET 200AL modules.

Control data record 196

The two figures below show parts of control data record 196 for a configuration of an ET 200SP expanded with ET 200AL modules.

This configuration is given as an example:

- The module "BA Send 1xFC" is in slot 1 "slot_1". This module allows you to integrate ET 200AL modules into an ET 200SP. In our configuration example, 16 AL modules are connected to the BA-Send (maximum configuration). If a BA-Send is being used, this module must be inserted in slot 1.
- All slots from 2 to 64 are assigned ET 200SP modules.
- A server module is inserted in slot 65.
- There are 16 AL modules in slots 66 to 81.

The ET 200SP with integrated AL modules, originally configured with STEP 7, is now to be reconfigured from the user program.

The new configuration has the following properties:

- The "BA Send 1xFC" module is inserted in "slot_1_BA-Send" (fixed setting).
- Module 2 "slot_2" is not available in the modified configuration (value "0").
- Module 3 "slot_3" is in slot 2 in the modified configuration (value "2").
- Module 4 "slot_4" is in slot 3 in the modified configuration (value "3").
- All modules from slot 5 to slot 81 are operated in the original configuration with STEP 7.

Configuration_ET200SP_with_AL-Modules

	Name	Data type	Start value
1	Static		
2	block_length	USInt	166
3	block_ID	USInt	196
4	version	USInt	2
5	subversion	USInt	0
6	slot_1_BA-Send	USInt	1
7	info_slot_1_BA-Send	USInt	0
8	slot_2	USInt	0
9	info_slot_2	USInt	0
10	slot_3	USInt	2
11	info_slot_3	USInt	0
12	slot_4	USInt	3
13	info_slot_4	USInt	0
14	slot_5	USInt	5
15	info_slot_5	USInt	0
16	slot_6	USInt	6
17	info_slot_6	USInt	0
18	slot_7	USInt	7
19	info_slot_7	USInt	0

The components of control data record 196 (figure above):

- block_length: Note the length of the control data record here; in the example: 166 (bytes). The length of the control data block is calculated using the formula "2 x number of slots + 4".
- block_ID: Enter the figure 196 here.
- version: The ET 200SP uses version 2 of control data record 196.
- subversion: The ET 200SP uses subversion 0 of control data record 196.

127	slot_62_SP	USInt	62
128	info_slot_62_SP	USInt	0
129	slot_63_SP	USInt	63
130	info_slot_63_SP	USInt	0
131	slot_64_SP	USInt	64
132	info_slot_64_SP	USInt	0
133	slot_65_SP	USInt	65
134	info_slot_65_SP	USInt	0
135	slot_66_AL	USInt	66
136	info_slot_66_AL	USInt	0
137	slot_67_AL	USInt	67
138	info_slot_67_AL	USInt	0
139	slot_68_AL	USInt	68
140	info_slot_68_AL	USInt	0
141	slot_69_AL	USInt	69
142	info_slot_69_AL	USInt	0

The components of control data record 196 (figure above):

- slot_65_SP: This byte relates to the server module in the ET 200SP rack. It ends the backplane bus of the ET 200SP.
- From "slot_66_AL" come the 16 configured ET 200AL modules: Our configuration example does not change the configuration with STEP 7. The byte "slot_66_AL" has the value "66", the byte "slot_67_AL" has the value "67", the byte "slot_68_AL" has the value "68", etc.

Definition of control data record 196

A control data record 196 containing a slot assignment is defined for configuration control.

Byte	Component	Value	Description	
0	block_length	e.g. 166 for maximum configuration with 65 ET-200SP modules and 16 ET 200AL modules (with DP interface module, maximum of 33 ET 200SP modules and 16 ET 200AL modules)	The length of the data record is calculated using the formula: 4 + "number of modules" x 2	Header
1	Block ID	196	ID for control data record 196	
2	version	2	Version 2 of control data record 196	
3	subversion	0	Subversion 0 of control data record 196	
4	slot_1_BA-Send	Real slot for SP module 1 Possible value: 1	When AL modules are integrated in ET 200SP, the module "BA Send 1xFC" must always be inserted in slot 1.	1st slot for SP modules Assignment for the configured SP module 1 to a real slot
5	info_slot_1_BA-Send	0 or 1	The value "1" means that a new potential group is opened with this module. (Not evaluated in this byte)	

6	slot_2	Real slot for SP module 2 Possible values: 2 to 65 (not 66 to 81, reserved for AL modules) 0 (if the configured module 2 is not available)	The configured SP module 2 can be inserted in any real slot from 2 to slot 65 (2 to 33 for a DP interface module). Slots 66 to 81 are for AL modules (34 to 49 with DP interface module).	2nd slot for SP modules Assignment for the configured SP module 2 to a real slot
7	info_slot_2	1	The value "1" means that a new potential group is opened with this module. A new potential group must always be opened in this byte as BA-Send cannot open a new potential group.	
8	slot_3	Real slot for SP module 3 Possible values: 2 to 65 (not 66 to 81, reserved for AL modules) 0 (if the configured module 3 is not available)	The configured SP module 3 can be inserted in any real slot from 2 to slot 65 (2 to 33 for a DP interface module). Slots 66 to 81 are for AL modules (34 to 49 with DP interface module).	3rd slot for SP modules Assignment for the configured SP module 3 to a real slot
9	info_slot_3	1	The value "1" means that a new potential group is opened with this module. .	
:	:	:	:	:
132	slot_65	Real slot for SP module 65 Possible values: 2 to 65 (not 66 to 81, reserved for AL modules) 0 (if the configured module 65 is not available)	The configured SP module 65 can be inserted in any real slot from 2 to slot 65 (2 to 33 for a DP interface module). Slots 66 to 81 are for AL modules (34 to 49 with DP interface module)	65th Slot for SP modules Assignment for the configured SP module 65 to a real slot
133	info_slot_65	0 or 1	The value "1" means that a new potential group is opened with this module (the value is not evaluated in this slot).	

134	slot_66	Real slot for AL module 1 Possible values: 66 to 81 (not 1 to 65, reserved for SP modules) 0 (if the configured AL module 1 is not available)	The configured AL module 1 can be inserted in any slot from 66 to slot 81 (34 to 49 for PROFIBUS).	1st Slot for AL modules Assignment for the configured AL module 1 to a real slot
135	info_slot_66	-	Reserve	
:	:	:	:	:
164	slot_81	Real slot for AL module 16 Possible values: 66 to 81 (not 1 to 65, reserved for SP modules) 0 (if the configured AL module 16 is not available)	The configured AL module 16 can be inserted in any slot from 66 to slot 81 (34 to 49 for DP interface module).	16th Slot for AL modules Assignment for the configured AL module 16 to a real slot
165	info_slot_81	-	Reserve	

Rules

- If the "BA Send 1xFC" module is being used, it must be inserted in slot 1.
- ET 200SP modules are inserted in slots 2 to 65 (slots 2 to 33 for DP interface module).
- AL modules are inserted in slots 66 to 81 (slots 34 to 49 for DP interface module).
- If you expand an ET 200SP with ET 200 AL modules, the 1st AL module is always coded in bytes 134 and 135 of the control data record, the 2nd AL module in bytes 136 and 137, etc., even if not all SP slots are to be occupied by SP modules. Non-assigned SP slots are coded with the value "0".

Error messages

The following error messages are returned if an error occurs when writing control data record 196:

Table 1-239 Error messages

Error code	Meaning
16#80A2	DP protocol error on layer 2 Indicates that a data record has not been acknowledged due to the system.
16#80B1	Invalid length; the length information in data record 196 is not correct.
16#80B5	Configuration control parameters not assigned.

Error code	Meaning
16#80B2	Invalid slot: The configured slot is not assigned.
16#80B8	Parameter error; module signals invalid parameters.
16#80C5	DP slave or module not available. Indicates that a data record has not been acknowledged due to the system.

Readback data record 197 for ET 200SP with AL modules

The actual configuration of an ET 200SP with AL modules can be checked with readback data record 197.

Data record 197 largely corresponds to readback data record 197 for ET 200SP without AL modules; however, it is longer as the additional AL modules also need to be coded.

There are two bytes in the data record for each module. The position of these two bytes in the data record codes a module in the original configuration with STEP 7.

In the figure below:

- The components "status_slot_1" and "reserve_1" (bytes 4 and 5 in the data record) correspond to the module in slot 1 in the configuration with STEP 7,
- "status_slot_2" and "reserve_slot_2" (bytes 6 and 7) correspond to the module in slot 2,
- "status_slot_3" and "reserve_slot_3" (bytes 8 and 9) correspond to the module in slot 3,
- etc.

The following data record is structured for configuration with 65 SP modules and 16 AL modules. The value "166" therefore appears in the "block_length" element of the data record.

If you configure an ET 200SP in STEP 7 with fewer AL modules, the data block will be shorter.

If you use fewer SP modules in a configuration, this has no effect on the length of data record 197 (with an expansion of ET 200SP with ET 200AL modules).

The "reserve_x" component of readback data record 197 is reserved for future applications.

You can choose any name for the components of the readback data record (for example "status_slot_1").

The figure below shows the start of readback data record 197 for reading the actual configuration of an ET 200SP with AL modules.

DB_read_back_ET_200SP_with_AL

	Name	Data type	Start value
1	Static		
2	block_length	USInt	166
3	block_ID	USInt	197
4	version	USInt	2
5	subversion	USInt	0
6	status_slot_1	USInt	1
7	reserve_slot_1	USInt	0
8	status_slot_2	USInt	1
9	reserve_slot_2	USInt	0
10	status_slot_3	USInt	1
11	reserve_slot_3	USInt	0
12	status_slot_4	USInt	1
13	reserve_slot_4	USInt	0
14	status_slot_5	USInt	1
15	reserve_slot_5	USInt	0
16	status_slot_6	USInt	1
17	reserve_slot_6	USInt	0

It does not show the components "status_slot_7" to "reserve_81" (maximum configuration of an ET 200SP with PN interface module) or "status_slot_7" to "reserve_slot_49" (maximum configuration of an ET 200SP with DP interface module).

Meaning of "status_slot_x":

- The value "1" in status_slot_x means that module x is inserted in the correct slot
- The value "0" codes all other possibilities (wrong module, empty slot, BU cover).

The figure below shows part of feedback data record 197 for reading the actual configuration of an ET 200SP with AL modules (and PN interface module). ET 200SP modules are inserted up to slot 65, and then the AL modules. For example, a value of "1" in the byte "status_slot_66_AL" means that the 1st AL module actually exists in the plant and is inserted in the correct slot.

128	status_slot_62_SP	USInt	1
129	reserve_slot_62_SP	USInt	0
130	status_slot_63_SP	USInt	1
131	reserve_slot_63_SP	USInt	0
132	status_slot_64_SP	USInt	1
133	reserve_slot_64_SP	USInt	0
134	status_slot_65_SP	USInt	1
135	reserve_slot_65_SP	USInt	0
136	status_slot_66_AL	USInt	1
137	reserve_slot_66_AL	USInt	0
138	status_slot_67_AL	USInt	1
139	reserve_slot_67_AL	USInt	0
140	status_slot_68_AL	USInt	1
141	reserve_slot_68_AL	USInt	0
142	status_slot_69_AL	USInt	1
143	reserve_slot_69_AL	USInt	0

Reading feedback data record 197

You can read the feedback data record 197 from the ET 200SP with the RDREC instruction. RDREC operates asynchronously. If you call RDREC in the startup OB, you must call the instruction multiple times using a loop until the "BUSY" or "DONE" output parameter indicates that the data record has been read.

Further information and examples

Specific examples of configuration control can be found in this application description (http://support.automation.siemens.com/WW/view/en/29430270).

Further information on ET 200SP can be found in the manuals for IM 155-6 PN (http://support.automation.siemens.com/WW/view/en/73184046) and IM 155-6 DP (http://support.automation.siemens.com/WW/view/en/73098660)

Further information on ET 200AL is available here (http://support.automation.siemens.com/WW/view/en/89254863).

See also

Configuration control with ET 200SP (Page 3215)

Configuration control with ET 200AL (Page 3262)

Expanding ET 200SP with ET 200AL modules

Introduction

The ET 200SP is a distributed I/O system for installation in a control cabinet.

The system can be expanded with modules of the ET 200AL series with IP65/67 degree of protection. ET 200AL modules can be mounted on site, for example on a machine.

Slot rules

The following rules apply to the use of ET 200AL modules in an ET 200SP with interface module:

- The "BA-Send 1xFC" module must be inserted in slot 1 of the ET 200SP if the ET 200SP is to be expanded with ET 200AL modules.
- The ET 200AL modules must be configured with no gaps.

The procedure detailed further on covers the use of ET 200AL modules in an ET 200SP with interface module.

The following, different, rules apply to the use of ET 200AL modules in an ET 200SP with CPU:

- The "BA-Send 1xFC" module can be located in slots 2 to 4 of the ET 200SP CPU.
- A CM DP communication module can be inserted as CPU submodule, but then occupies slot 2.

Procedure

Follow the steps below to configure an ET 200SP with interface module with ET 200AL modules:

1. Drag an interface module (PROFINET or PROFIBUS) from the ET 200SP series to the network view.
2. Go to the device view. You do this by clicking twice on the icon of the module you have just inserted.
3. Insert the module "BA Send 1xFC" into slot 1 of the ET 200SP.
   STEP 7 now generates an ET-Connection rack with 16 slots for ET 200AL modules (figure below).
   An ET-Connection rack is a virtual rack that sets the order of the connected ET 200AL modules.
   Question marks are displayed above the slots as an ET 200AL module has yet to be connected to BA-Send.

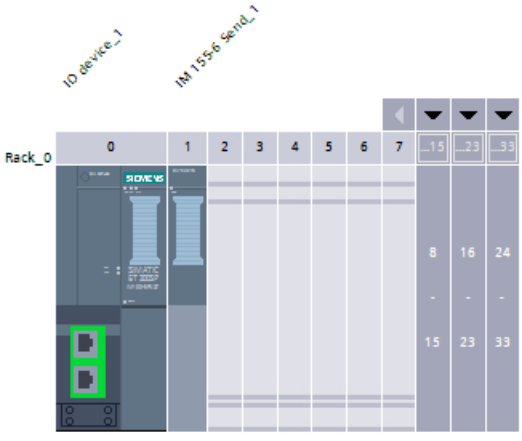

4. From the hardware catalog (subfolder ET 200AL in the ET 200SP folder), select the first ET 200AL module to be connected to the ET 200SP: Drag and drop this module to the 1st slot in the ET connection rack.
   From this module, STEP 7 generates a line to the "BA-Send 1xFC" module and allocates the slot numbers 66 to 81 (figure below).
   If you configure the ET 200SP with a DP interface module, STEP 7 assigns the slot numbers 34 to 49 for the ET 200AL modules.

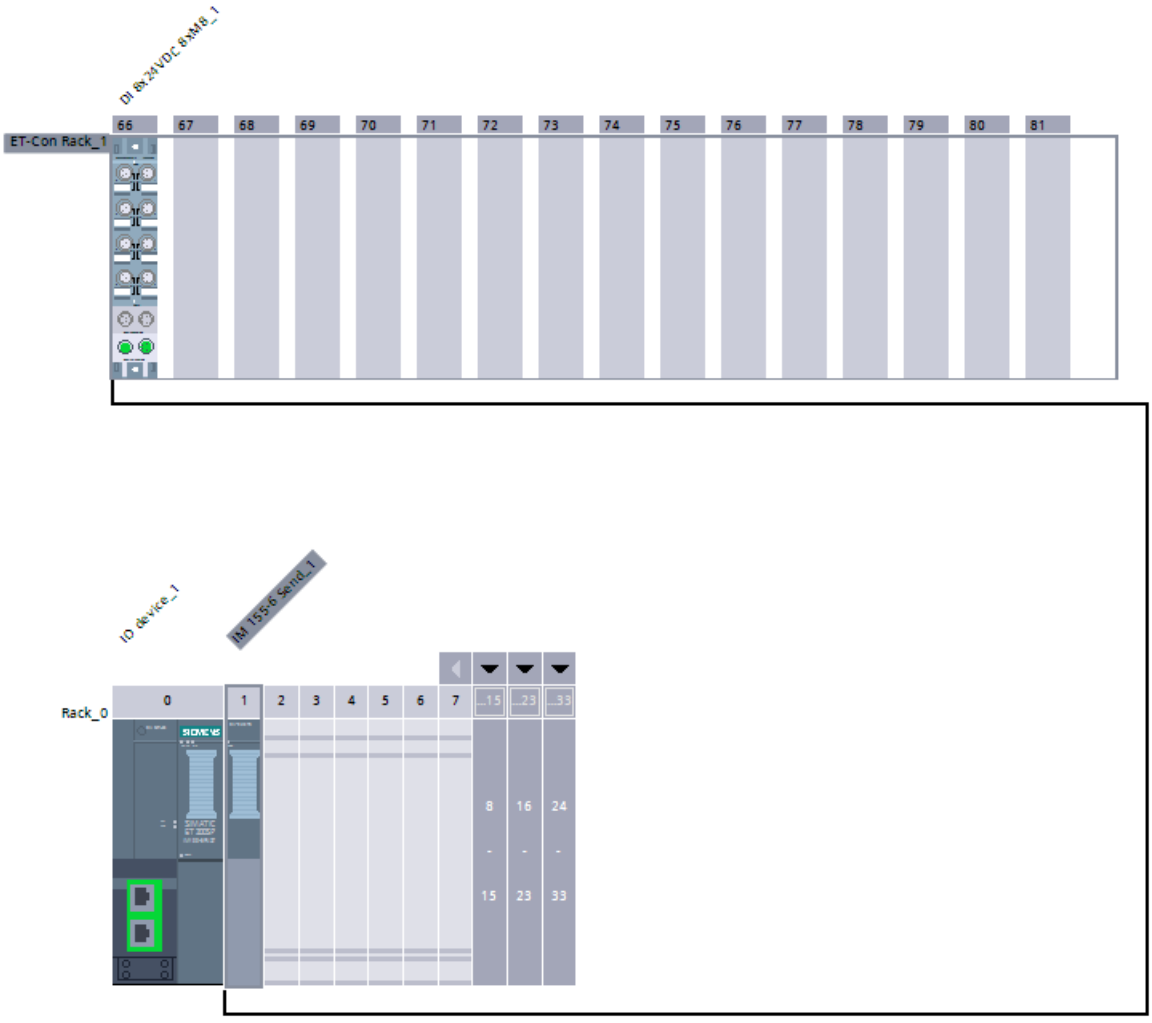

5. Now drag all other ET 200AL modules into the free slots in the ET-Connection rack. STEP 7 automatically inserts the ET-Connection between the individual ET 200AL modules (green loops).
   In the configuration below, five ET 200AL modules are connected in series.

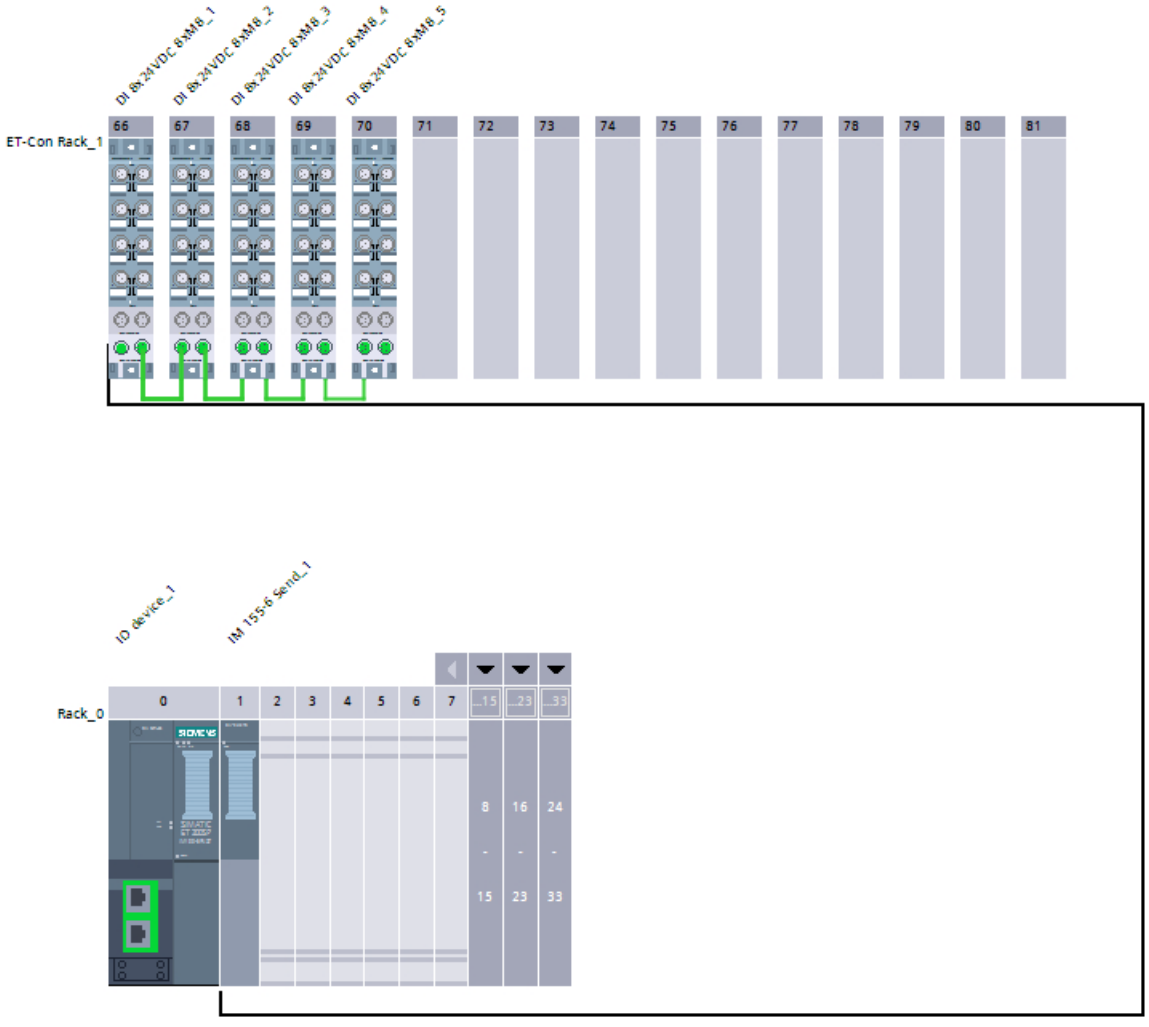

6. Complete the configuration of the ET 200SP: Drag all ET 200SP modules from the hardware catalog to the slots in the ET 200SP.
   Five ET 200SP modules are plugged in the configuration example below: The PN interface module in slot 0, the module "BA-Send 1xFC" in slot 1, one input module each in slots 2 and 3, and a server module in slot 4:

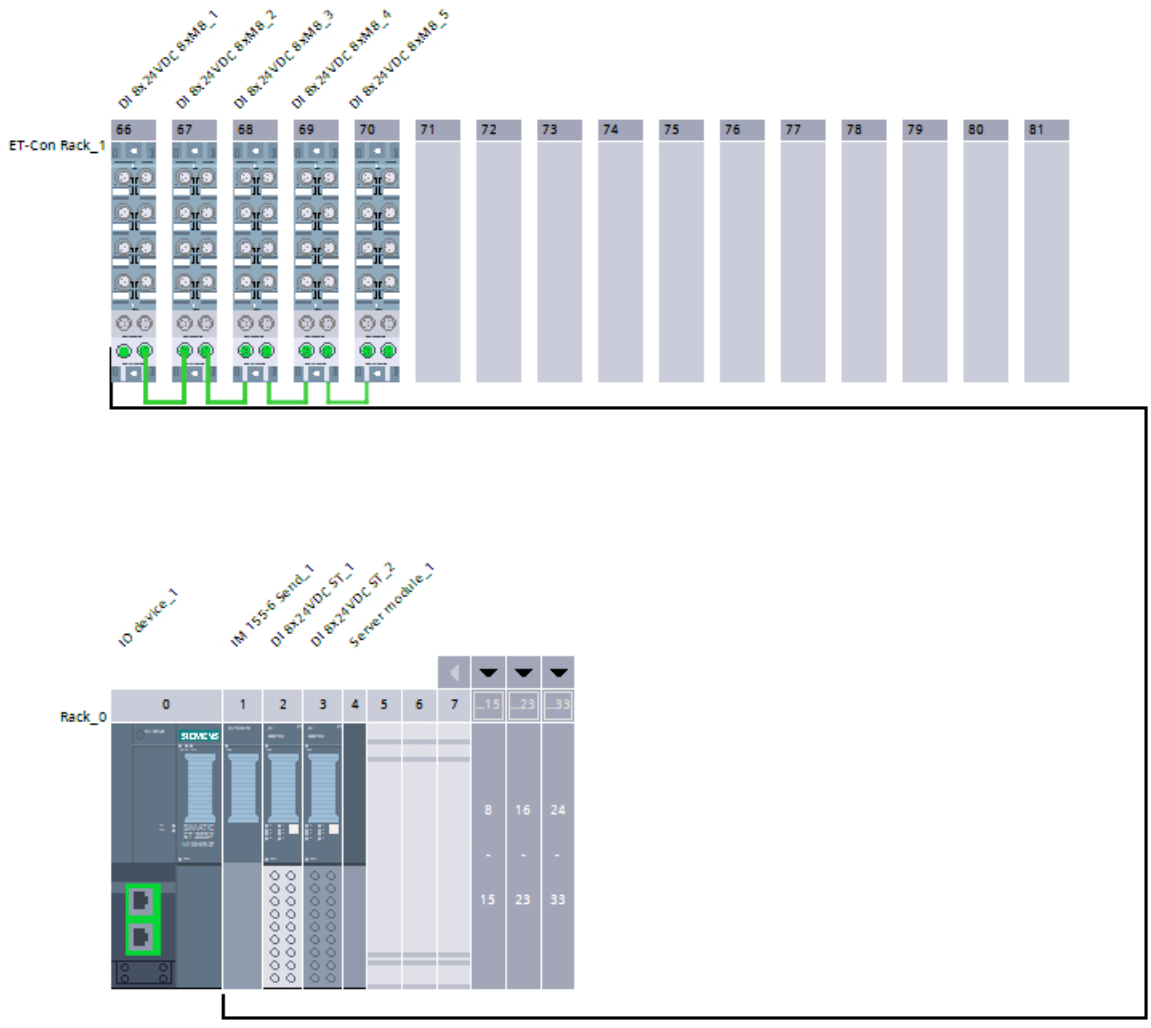

See also

Configuration control with ET 200AL (Page 3262)

ET 200AL distributed I/O system (Page 3255)

Output module parameters

Substitute value reaction

Substitute value reaction

In the ET 200SP, the substitute value reaction is executed by the IO controller per slot.

The respective output reacts according to its set substitute value reaction:

- "Turn off"
- "Output substitute value"
- "Keep last value"

This substitute value reaction is triggered in the following cases:

- IO controller in STOP
- Controller failure (connection interruption)
- Firmware update
- Reset to factory settings
- More than one I/O module withdrawn simultaneously
- Disable the IO device
- Station stop
  - Missing server module
  - More than one I/O module withdrawn simultaneously
  - At least one I/O module is inserted on the wrong BaseUnit

Note

Reducing a configuration

If you reduce the configuration of the ET 200SP and download the configuration to the CPU, the modules which are no longer configured but still present retain their original substitute value reaction. This applies until the supply voltage on the BaseUnit BU...D or on the interface module is turned off.

Input module parameters

Parameters of the digital input modules

Diagnostics missing supply voltage L+

Enabling of the diagnostics for missing or insufficient supply voltage L+.

Diagnostics short-circuit to ground

Enabling of the diagnostics if a short-circuit of the actuator supply to ground occurs.

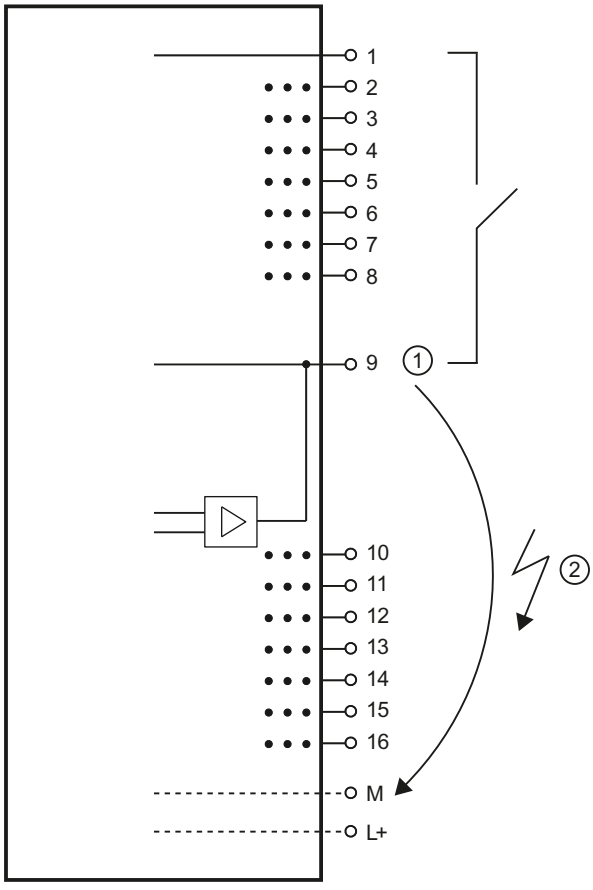

① Encoder supply
② Short-circuit

Diagnostics short-circuit to L+

Enabling of the diagnostics if a short-circuit of the encoder supply to L+ occurs.

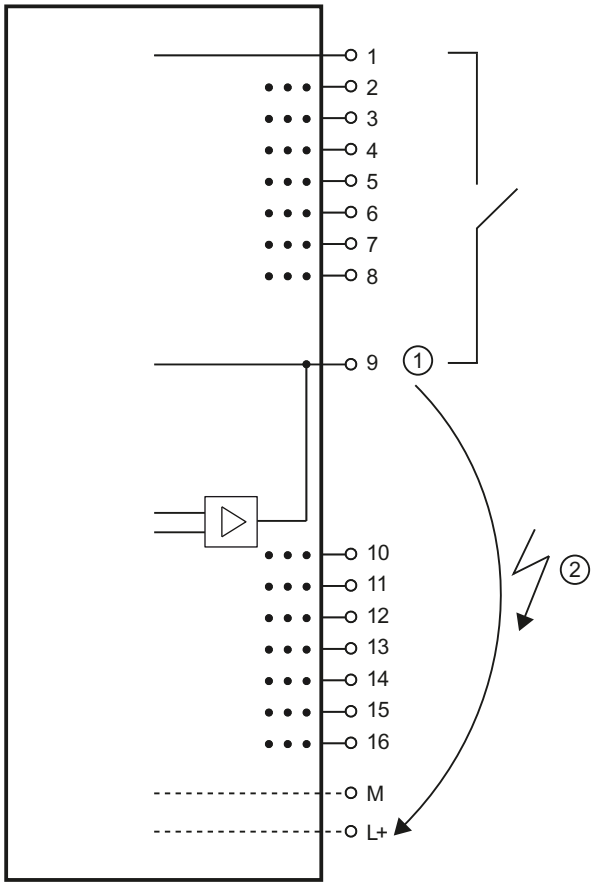

① Encoder supply
② Short-circuit

Diagnostics wire break

Enabling diagnostics if the line to the encoder is interrupted.

Operating mode

Determines whether a channel is enabled or disabled.

Pulse extension (only High Feature modules)

Pulse extension is a function for changing a digital input signal. A pulse at a digital input is extended to at least the configured length. If the input pulse is already longer than the configured length, the pulse is not changed.

Pulse extension is started whenever the state of the input signal changes and no pulse extension is active for this channel.

Potential group of the left module / new potential group

Specifies whether the I/O module is located on a base unit with supply voltage feed (new potential group) or whether it is it located on a base unit without supply voltage feed (in which case it belongs to the potential group of the left module).

Parameters of the analog input modules

Missing supply voltage L+

Enabling of the diagnostics, with missing or too little supply voltage L+.

Reference junction (AI 4xRTD/TC 2-/3-/4-wire HF)

A BaseUnit with internal temperature sensor (BU..T) or the channel 0 of the I/O module can be used as reference junction, provided this has been configured as "Thermal resistor Pt100 climatic range Celsius".

A possible parameter assignment is shown below (see also Information about reference channel mode (Page 3238)):

Table 1-240 RTD channel

Setting	Description
No reference channel operation	Temperature value at channel 0 can be used as module-wide reference value if the parameters of the other channels are assigned accordingly.
Reference channel of Group x	The channel acts as sender for the reference junction temperature of Group x. Distribution is performed via the interface module.

Table 1-241 TC channel

Setting	Description
Reference channel of the module	The corresponding TC channel uses the channel 0 of the same module as reference junction temperature. This must be set as "Thermal resistor Pt 100 climatic range Celsius" and "No reference channel operation"; otherwise, reference junction diagnostics is triggered.
Internal reference junction	The reference junction temperature is read by an internal temperature sensor on the BaseUnit. Reference junction diagnostics is triggered if there is a wrong BaseUnit type.
Reference channel of Group x	With the setting "TC" (thermocouple...), the channel acts as receiver for the reference junction temperature of Group x.
Fixed reference temperature	No temperature compensation occurs. The linearization is executed with an assumed reference junction temperature of 0 °C.

Overflow

Enabling of the diagnostics if the measured value exceeds the overflow range.

Underflow

Enabling of the diagnostics if the measured value falls below the underflow range.

Wire break

Enabling of the diagnostics if the module has no current flow or too low a current flow for the measurement on the corresponding configured input.

Smoothing

The individual measurements are smoothed using digital filtering. The smoothing can be set in 4 stages, whereby the smoothing factor k multiplied by the cycle time of the I/O module corresponds to the time constant of the smoothing filter. The larger the smoothing, the larger the time constant of the filter.

The following figure shows the step response for the various smoothing factors depending on the number of module cycles.

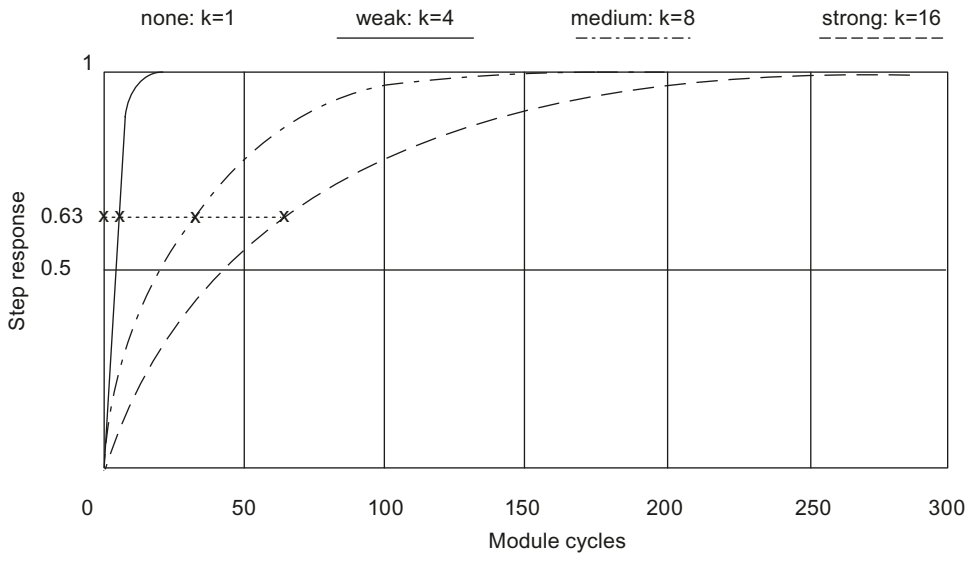

Figure 1-130 Smoothing with AI 4×RTD/TC 2-/3-/4-wire HF

Interference frequency suppression

With analog input modules, suppresses the disturbance caused by the frequency of the AC network used.

The frequency of the AC network may interfere with measured values, particularly for measurements within low voltage ranges and when thermocouples are being used. This parameter is used to define the predominant power frequency of the system.

Hardware interrupt limits

If the high limit 1/2 or the low limit 1/2 is violated, the module triggers a hardware interrupt.

Below are some examples for the selection of the limits 1 and 2.

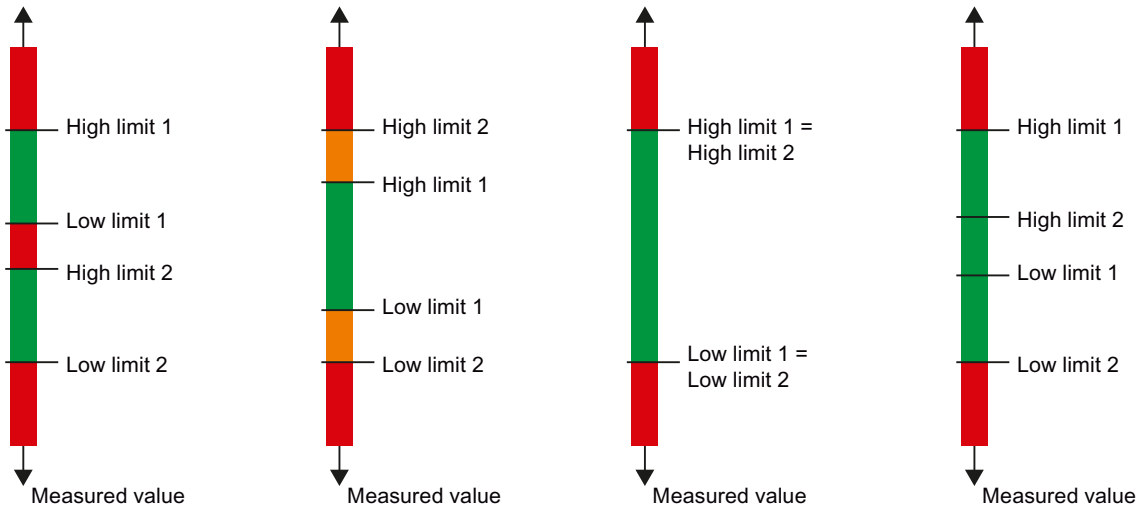

Low limit 1/2

Specify a threshold whose undershoot triggers a hardware interrupt.

High limit 1/2

Specify a threshold whose overrange triggers a hardware interrupt.

Potential group of the left module/New potential group

Specifies whether the I/O module is located on a BaseUnit with supply voltage infeed (new potential group) or on a BaseUnit without supply voltage feed (in which case, it belongs to the potential group of the left module).

Temperature coefficient (measurement type thermoresistor)

The correction factor for the temperature coefficient (α value) defines the relative rate of change of the resistance of a specific material at a temperature rise of 1 °C.

The temperature coefficient depends on the chemical composition of the material. In Europe, only one value is used per sensor type (default value).

The further values facilitate a sensor-specific setting of the temperature coefficient and enhance accuracy.

See also

Special features of AI 4xRTD/TC 2-/3-/4-wire HF (Page 3246)

Information about reference channel mode

An RTD/TC module of the ET 200SP works in reference channel mode when a channel sends the reference temperature to other channels of the station. The receiving channels use the reference temperature for temperature compensation when measuring with thermocouples.

Structure and use of thermocouples

The thermocouple consists of two wires with different metals or alloys that are welded together at the end. The weld is called measuring point.

The other end of the two wires is open. This end is called reference junction.

A thermal voltage, which depends on the temperature at the measuring point, occurs between the two metals/alloys at the measuring point. Other thermal voltages also occur at the reference junction - with transition from thermocouple to copper lines, for example - that falsify the actual measured value and need to be compensated. No compensation is required for a reference junction temperature of 0 °C.

Different methods are used to compensate the reference junction temperature:

- Fixed reference temperature: The reference junction is permanently set to a specific temperature, for example, to 0 °C by an icewater bath (no compensation required).
- Internal reference junction: The reference junction is the terminal of the BaseUnit on which the analog module is plugged. If you select the "Internal reference junction" compensation type, you have to use BaseUnits with integrated temperature measurement to compensate the reference junction temperature. These BaseUnits have the designation "BU..T". The module records the temperature at the reference junction and uses this to determine the actual temperature at the measuring point.
- Reference channel of Group x: An external thermal resistor records the temperature at the reference junction for the Group x (group of channels within a station). The actual temperature at the respective measuring point can be determined as a result. One external thermal resistor is required for each group. The thermal resistors are each connected to one channel of an analog module. These channels are called senders (for the temperature at the reference junction).
  The mode of operation and settings are described in the section "Station-wide distribution of reference temperature".
- Reference channel of the module: The mode of operation is comparable with "Reference channel of Group x". You connect an external thermal resistor to channel 0 of the module to measure the temperature at the reference junction. Other channels of the same module use this reference temperature for temperature compensation.
  The mode of operation and settings are described in the section "Module-wide distribution of reference temperature".

You can find information about the structure and mode of operation of a thermocouple in the Analog value processing (http://support.automation.siemens.com/WW/view/en/67989094) manual.

Station-wide distribution of the reference temperature

You can record the temperature at the reference junction x using a thermal resistor on a channel (sender of the reference temperature) and send it to other channels within a station (receivers). All channels that receive the temperature at a reference junction x form the Group x.

For each group, you assign parameters to exactly one channel as sender of the reference temperature.

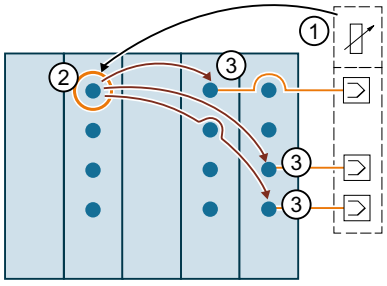

① Thermal resistor at the reference junction

② Channel records temperature at the reference junction and sends it to the other channels within a station (sender of Group x). The temperature value is used for the compensation of the reference junction temperature.

③ Channels of the Group x receive the temperature at the reference junction (receivers)

Parameter assignment of a channel as reference channel (sender for Group 1)

Parameter assignment is described below using Group 1 as example:

1. Open the project in STEP 7
2. Select the required analog module (RTD/TC) in the device view.
3. Then select a channel that is to work as sender of the reference junction temperature.
   The following settings are required:
   "Measurement type": "Thermal resistor", for example, "Thermal resistor (4-wire connection)"
   "Measuring range": "Pt 100 climatic range"
   "Temperature unit": "Degrees Celsius"
   "Reference junction": "Reference channel of Group 1"

The following figure shows the parameter assignment.

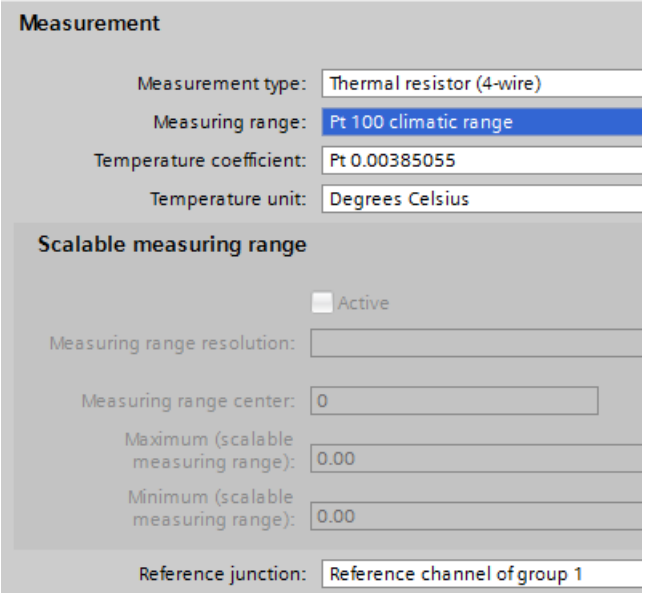

The channel configured in this way (thermal resistor measurement type) works as reference channel of Group 1 and sends the measured temperature to all channels (thermocouple measurement type) that are configured as receivers of Group 1.

In the next section, you will learn how to assign parameters for channels that are receivers of Group 1.

Parameter assignment of a channel as receiver of Group 1

The figure below shows the parameter assignment of a channel that receives the temperature at the reference junction of Group 1.

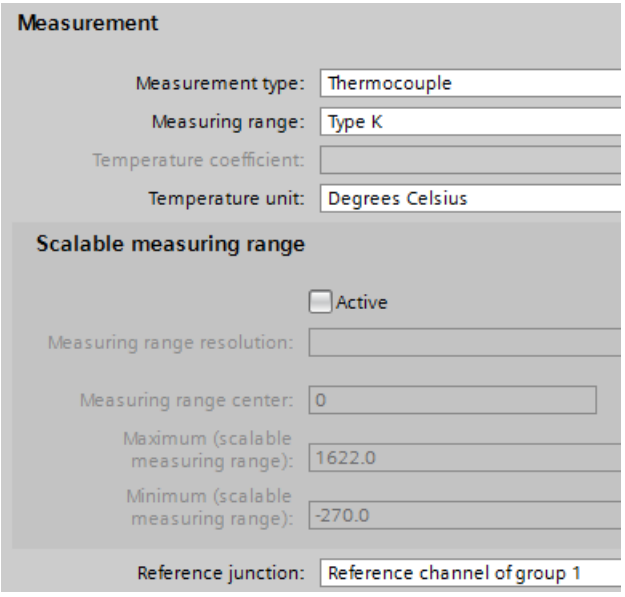

The following settings are required:

- "Measurement type": "Thermocouple"
- "Reference junction": "Reference channel of Group 1"

Module-wide distribution of the reference temperature

You can record the temperature on a reference junction using the channel 0 of the module and use the temperature value for the channels 1, 2, 3... of this module. The recorded temperature value is not sent to the channels of the other modules within the station (no reference channel mode).

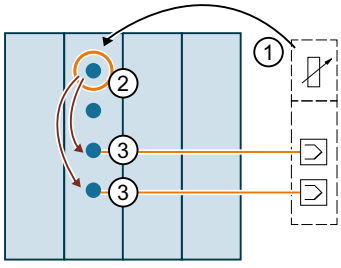

① Thermal resistor at the reference junction

② Channel records temperature at the reference junction, "Reference junction" parameter: "No reference channel mode"

③ Channels of the same module use the temperature value for the compensation of the reference junction temperature for the measurement with thermocouples, "Reference junction" parameter: "Reference channel of the module"

Parameter assignment of channel 0 as reference channel of the module

The figure below shows the parameter assignment of channel 0 of a module that is to be used for recording the temperature at the reference junction:

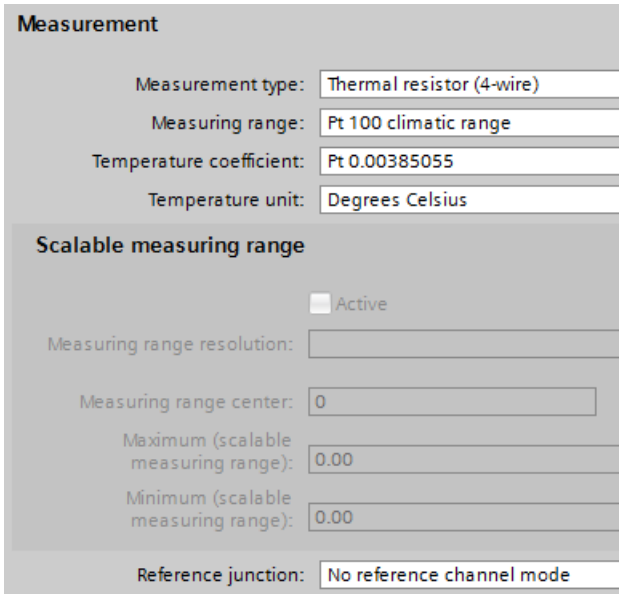

The following settings are required:

- "Measurement type": "Thermal resistor", for example, "Thermal resistor (4-wire connection)"
- "Measuring range": "Pt 100 climatic range"
- "Temperature unit": "Degrees Celsius"
- "Reference junction": "No reference channel mode"

Parameter assignment of a channel that uses channel 0 as reference channel

The figure below shows how parameters have to be assigned for a channel that uses the channel 0 of this module as reference channel for temperature compensation.

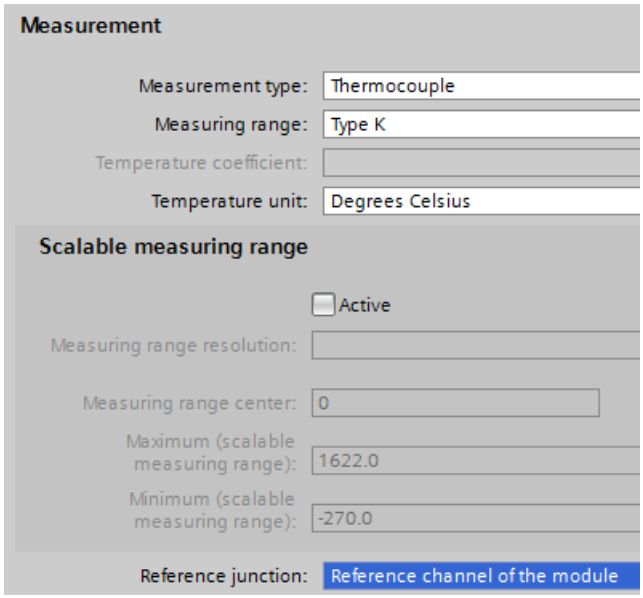

The settings below are required for the channels of the module that compensate the temperature of the reference junction using channel 0:

- "Measurement type": "Thermocouple"
- "Reference junction": "Reference channel of the module"

See also

Parameters of the analog input modules (Page 3235)

Information about the Oversampling function

High-speed analog modules (HS) are available to meet high performance and speed requirements. The main characteristics of these HS analog modules compared to Standard analog modules (ST) is their shorter cycle times. To achieve this goal, the input and output modules are equipped with components with extremely short throughput and conversion times. In addition, the entire architecture of the modules is designed for faster signal processing.

HS analog modules convert the output of measured values and output values at the same time. Each channel within the module has its own A/D or D/A converter. This means the cycle time is basically the conversion time and independent of the number of activated channels. This is true for analog input modules as well as analog output modules. This means HS modules can be used in fast isochronous mode.

Apart from isochronous mode, the HS analog modules also provide benefits in non-isochronous (free-running) mode. Due to the fast processing of the process signals, HS analog modules are able to detect changes in the process values more quickly and to respond to

these events with the appropriate program blocks (for example, hardware interrupt or cyclic interrupt organization blocks).

Isochronous mode

Isochronous mode refers to synchronous coupling

- Of signal acquisition and output via the distributed I/O
- Of signal transmission via PROFIBUS or PROFINET
- Of program processing with the constant bus cycle time of the PROFIBUS or PROFINET.

The result is a system that acquires its input signals in constant time intervals, processes them and outputs the output signals. Isochronous mode guarantees reproducible and defined process reaction times as well as equidistant and synchronous signal processing with distributed I/O.

The bus system and the I/O modules work synchronously with configured isochronous mode. The transmitted input and output data are linked to an "isochronous task" in the CPU. As a result, the data of a cycle are always consistent. All data of a process image belong together logically and in time. Jitter in the user program caused by the acquisition of outdated values is therefore almost impossible.

Even fast processes can be perfectly controlled by the exact timely reproducibility of all processes. Isochronous mode thus contributes to high control quality and hence to greater manufacturing precision. While possible fluctuations of the process reaction times are drastically reduced. The time-assured processing can be utilized to improve machine cycle times. Shorter cycle times increase the processing speed and help to lower production costs.

Oversampling

The use of the oversampling function in analog input or analog output modules requires an isochronous configuration.

With analog input modules, the set send clock is divided into time-equidistant sub-cycles. The send clocks can be subdivided into 2 to 16 sub-cycles. Each sub-cycle reads in a measured value. The read-in measured values of a data cycle are copied to the interface module (IM) in the next send clock and are then available to the processing CPU one clock later.

With analog output modules, the set send clock is also divided into time-equidistant sub-cycles. The send clocks can be subdivided into 2 to 16 sub-cycles. Each sub-cycle returns an output value. The output values are copied to the interface module by the CPU within the same send clock and are written to the process one send clock later.

The read-in and output values are transmitted in the user data of the analog module. In this way, the address space of the module is extended from 2 bytes of user data per channel to 16 x 2 bytes of user data per channel (with 16 sub-cycles). If you subdivide the send clock into fewer than 16 sub-cycles, the unused addresses are assigned the error value 0x7FFF during input. For output, the values of the unused addresses are ignored.

Because the sub-cycles have to be within a send clock, oversampling needs an additional clock to copy the data to the interface module, unlike the 3-cycle model of isochronous mode. The result is a 5-cycle model.

Isochronous mode

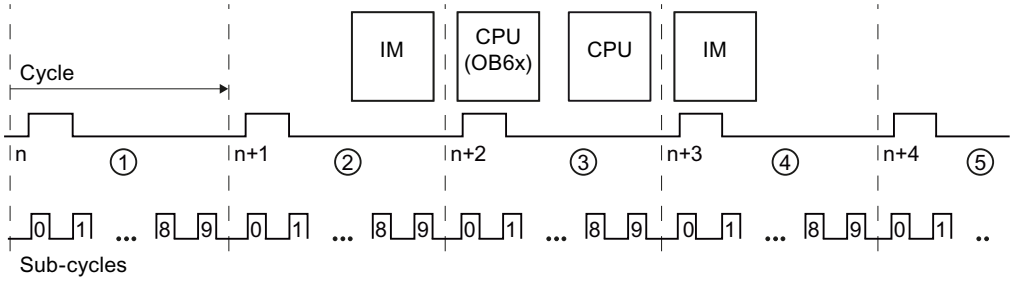

① Send clock is divided into sub-cycles which record the measured value (in this case: 10 sub-cycles)

② Measured values are copied to the IM

③ Measured values are processed and output values determined

④ Output values are copied by the CPU to the IM

⑤ Output values are written to the process

Higher sampling rates through oversampling

Due to the configured modules, the IO device has a minimum possible update time. Within this update period, the IO device/IO controller of the PROFINET IO system is supplied once with new data.

The following applies with respect to the channels of a single standard module in the I/O device: The shortest update time ("sampling rate") is exactly one send clock.

If you want to shorten the update time for the channels of a standard module, you have to shorten the send clock. Because of the properties of the involved components and the structure of the I/O system, this is only possible to a certain extent (e.g., down to 0.25 ms).

Modules with oversampling, however, offer the option of further reducing the update time ("sampling rate") for their channels without having to shorten the send clock for the entire IO device at the same time.

The subdivision of the send clock into time-equidistant sub-cycles enables the processing of faster processes through higher sampling rates.

Example

In practice, the use of oversampling makes sense when the isochronous system works with only one specific send clock (for example, 1 ms) due to the modules used and when faster sampling of the process values is required. By using oversampling and subdividing the send clock into 4 sub-cycles, for example, you can sample the process values in intervals of 250 µs.

Configuring oversampling

Enable the option "Isochronous mode" in the IO device used, and set the corresponding parameters ("Send clock", etc.).

With the distributed analog input modules (e.g. AI 2xU/I 2,4-wire HS), you specify the number of sub-cycles using the "Sampling rate" parameter.

With the distributed analog output modules (e.g. AQ 2xU/I HS), you specify the number of sub-cycles using the "Sampling rate" parameter.

If, for example, you configure a "Sampling rate" of 4 "Values/cycle" for a send clock of 1 ms, the send clock is subdivided into 4 sub-cycles and the process values are sampled at intervals of 250 µs.

Reference

You can find additional information in the manuals on the high-speed analog modules and in the Analog value processing (http://support.automation.siemens.com/WW/view/en/67989094) function manual.

Special features of AI 4xRTD/TC 2-/3-/4-wire HF

Use of Cu10 sensors

- Select "3-wire thermal resistor" and "Cu10" in the parameter assignment.
- Wire the Cu10 sensor using 3-wire connection technology.
- An automatic, internal compensation of the line resistance of the missing measuring line takes place during operation.

Note

To ensure optimum line compensation for Cu10, please note the following:

- The sum of cable resistance and measuring resistance must not exceed 31 Ω.
- Cable resistance should not exceed 8 Ω if you want to use the temperature range up to over 312 °C.
  Example: A 200 m long copper cable with 0.5 mm^2 core cross-section has approx. 7 Ω. A lower cross-section reduces the permissible cable length accordingly.

Use of PTC resistors

PTCs are suitable for temperature monitoring of or as thermal protective equipment for complex drives or transformer windings.

- Select "2-wire resistor" and "PTC" in the parameter assignment.
- Connect the PTC using 2-wire technology.
- Use type A PTC resistors (PTC thermistors) in accordance with DIN/VDE 0660, Part 302.
- If the "Over-/underflow" diagnostics is enabled, a "low limit violation" diagnostics which shows a short circuit is generated for resistance values < 18 Ω.
- Sensor data on PTC resistance:

Table 1-242 Use of PTC resistors

Property	Technical specifications	Note
Switching points	**Reaction to rising temperature**	
	< 550 Ω	**Normal range:** • SIMATIC S7: bit 0 = "0", bit 2 = "0" (in the PII)
	550 Ω to 1650 Ω	**Prewarning range:** • SIMATIC S7: bit 0 = "0", bit 2 = "1" (in the PII)
	< 1650 Ω	**Response range:** • SIMATIC S7: bit 0 = "1", bit 2 = "0" (in the PII)
	Reaction to falling temperature	
	< 750 Ω	**Response range:** • SIMATIC S7: bit 0 = "1", bit 2 = "0" (in the PII)
	750 Ω to 540 Ω	**Prewarning range:** • SIMATIC S7: bit 0 = "0", bit 2 = "1" (in the PII)
	< 540 Ω	**Normal range:** • SIMATIC S7: bit 0 = "0", bit 2 = "0" (in the PII)
(TNF-5) °C (TNF+5) °C (TNF+15) °C Measuring voltage Voltage on the PTC	max. 550 Ω min. 1330 Ω min. 4000 Ω max. 7.5 V	RRT= rated response temperature

- Assignment in the process image inputs (PII) with SIMATIC S7

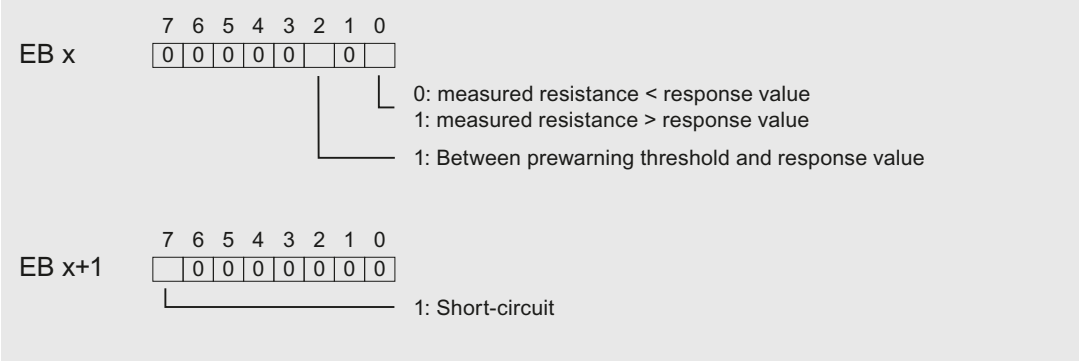

Figure 1-131 Assignment in the process image inputs (PII)

- Notes on programming

Note

Only the bits 0+2 are relevant for the evaluation in the process image inputs. You can use the bits 0+2 to monitor the temperature, for example, of a motor.

The bits 0+2 in the process image inputs have no latching function. When you are assigning parameters, take into consideration that a motor, for example, starts up in a controlled manner (via an acknowledgment).

Bits 0+2 can never be set simultaneously, but are instead set consecutively.

For safety reasons, always evaluate the diagnostic entries of the AI 4×RTD/TC 2-/3-/4-wire HF, as no measurement is possible if I/O modules are unplugged, if the supply voltage of the I/O module has failed, or if there is a wire break or short circuit of the measuring lines.

Example

The following diagram shows the temperature variation and the associated switching points.

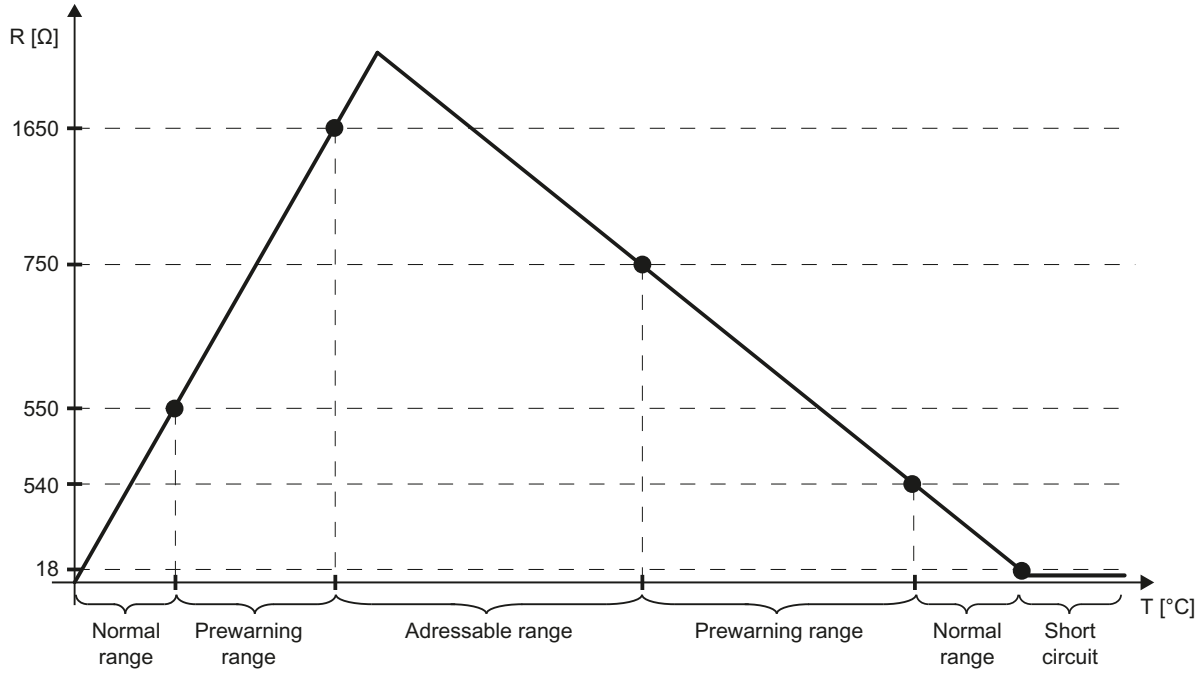

See also

Parameters of the analog input modules (Page 3235)

What you should know about the scalable measuring range (Page 3249)

What you should know about the scalable measuring range

The scalable measuring range

The scalable measuring range is a section from the temperature measuring range of an analog input module (for example, the ET 200SP module "AI 8xRTD/TC 2-wire HF").

In this section, a higher resolution of the measured values is possible, comparable with a magnifier that allows a subsection to be viewed in greater detail.

The scalable measuring range is supported for the following measurement types:

- Thermal resistor (RTD) standard
- Thermocouple

The scalable measuring range is not available with the following measurement types:

- Voltage
- Resistance
- Thermal resistor climatic

Position and resolution of the scalable measuring range

The position and resolution of the scalable measuring range can be set (scaled):

- **Position:** The scalable measuring range can be moved over the entire standard measuring range. This allows you to select the temperature range for which your application requires a higher resolution.
  Exception: The scalable measuring range cannot be moved to the extent that it enters the overflow or underflow of the standard measuring range (Clipping).
  You select the position of the scalable measuring range with the "Measuring range center" parameter (see figure below).
- **Resolution:** The following values can be set:
  - 2 decimal places (0.01 °C)
  - 3 decimal places (0.001 °C)

  You set the resolution with the "Measuring range resolution" parameter (figure below).

Example of a parameter assignment

The following figure shows a parameter assignment for the ET 200SP module "AI 8xRTD/TC 2-,3-,4-wire HF".

In STEP 7, you can find parameters in the properties box via General > AI 4 > Inputs > Channel 0 to channel 3.

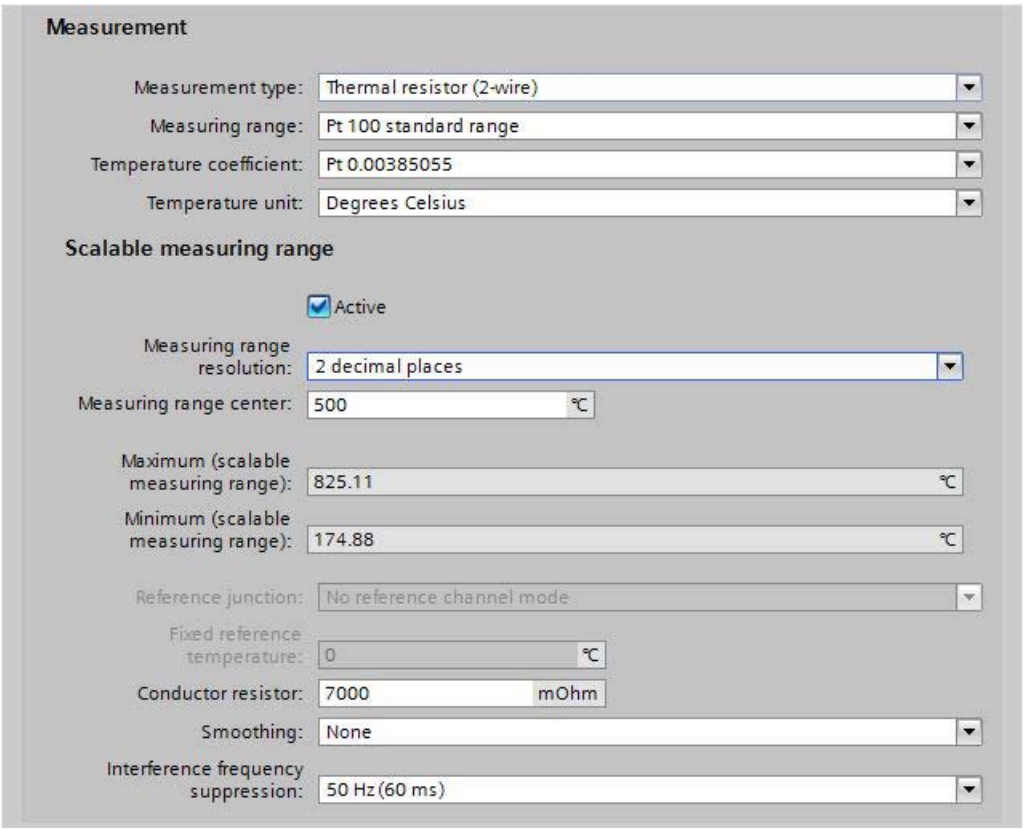

Conductor resistor:

The "Conductor resistor" parameter in the parameter assignment above is enabled only if the "Thermal resistor (2-wire terminal)" measurement type was selected.

Here, enter the value for the resistance of the connecting cable of the thermal resistor: A 200 m long copper cable with 0.5 mm^2 wire cross-section, for example, has a resistance value of seven ohms.

Measuring range resolution:

In the parameter assignment above, a resolution of 0.01 °C was selected (measuring range resolution "2 decimal places").

Measuring range center:

The measuring range center is set to 500 °C.

With a resolution of 0.01 °C, this results in a scalable measuring range from 174.88 °C to 825.11 °C.

At a resolution of 0.01 °C, the scalable measuring range covers 650.23 °C.

Maximum (scalable measuring range):

This value represents the high limit of the scalable measuring range. In the example above, 825.11 °C.

The value is calculated by STEP 7 (at a resolution of 0.001 °C, the high limit is at 532.511 °C, see figure below).

Minimum (scalable measuring range):

This value represents the low limit of the scalable measuring range. In the example above, 174.88 °C.

The value is calculated by STEP 7 (at a resolution of 0.001 °C, the low limit is at 467.488 °C; see figure below).

Higher resolution:

The following figure shows a parameter assignment with a resolution of 0.001 °C (otherwise the same example as in the figure above):

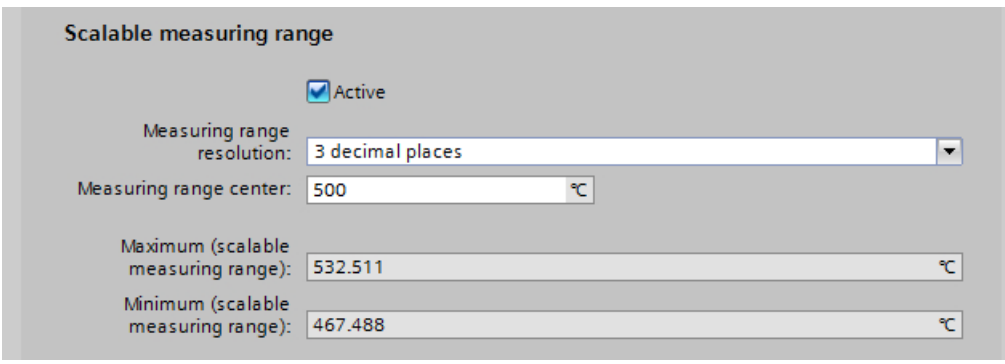

At a resolution of 0.001 °C, the scalable measuring range is between 467.488 and 532.511 °C and covers 65.023 °C (a tenth of the measuring range at a resolution of 0.01 °C).

Standard measuring range with 0.1 °C resolution

The following table shows the standard measuring range for thermal resistors of the type "Pt 100", values in degrees Celsius.

Pt 100 standard in °C (1 digit = 0.1 °C)	Decimal values	Hexadecimal values	Ranges
> 1000.0	32767	7FFF	Overflow
1000.0 : 850.1	10000 : 8501	2710 : 2135	Over range
850.0 : -200.0	8500 : -2000	2134 : F830	Nominal range
-200.1 : -243.0	-2001 : -2430	F82F : F682	Under range
< -243.0	-32768	8000	Underflow

The standard measuring range is the basis for the scalable measuring range.

You can set the measuring range center within the nominal range (-200 °C to 850 °C, see table above).

For temperatures below and above the set measuring range center, you then obtain measured values with a higher resolution.

The width of this range around the measured value center depends on the selected resolution.

Scalable measuring range 0.01 °C and 0.001 °C resolution

The scalable measuring range is identified by the following value ranges:

Scalable measuring range	Measuring range resolution (values in °C)		Hexadecimal values
	2 decimal places	3 decimal places	
Overflow	> 325.11	> 32.511	7FFF
High limit	325.11	32.511	7EFF
Measuring range center	0	0	0
Low limit	325.11	-32.511	8100
Underflow	< -325.11	< -32.511	8000

The maximum and minimum of the scalable measuring range depend on the selected resolution:

- 2 decimal places, resolution of 0.01 ℃:
  The high limit is 325.11 °C above the measuring range center you have set.
  The low limit is 325.11 °C below the measuring range center you have set.
  This means that the scalable measuring range is 650.22 °C around the measuring range center.
- 3 decimal places, resolution of 0.001 ℃:
  The high limit is 32.511 ℃ above the measuring range center you have set.
  The low limit is 32.511 ℃ below the measuring range center you have set.
  This means that the scalable measuring range is 65.022 ℃ around the measuring range center.

Calculation of the temperature

You calculate the temperature value by adding the value you receive from the module to the measuring range center.

Example:

- You have set the measuring range center 500 °C (see example in the section "Example of a parameter assignment"). For the resolution, you selected "2 decimal places".
- From the module, you receive the hexadecimal value "0100" in S7 format:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

- The hexadecimal value "0100" corresponds to the decimal value 256.
- Since you have selected a resolution of 0.**01** °C, the number 256 corresponds to the temperature value "2.**56** °C".
- You now add 500 °C and 2.56 °C and obtain the measured value 502.56 °C.

Scalable measuring range in the standard measuring range

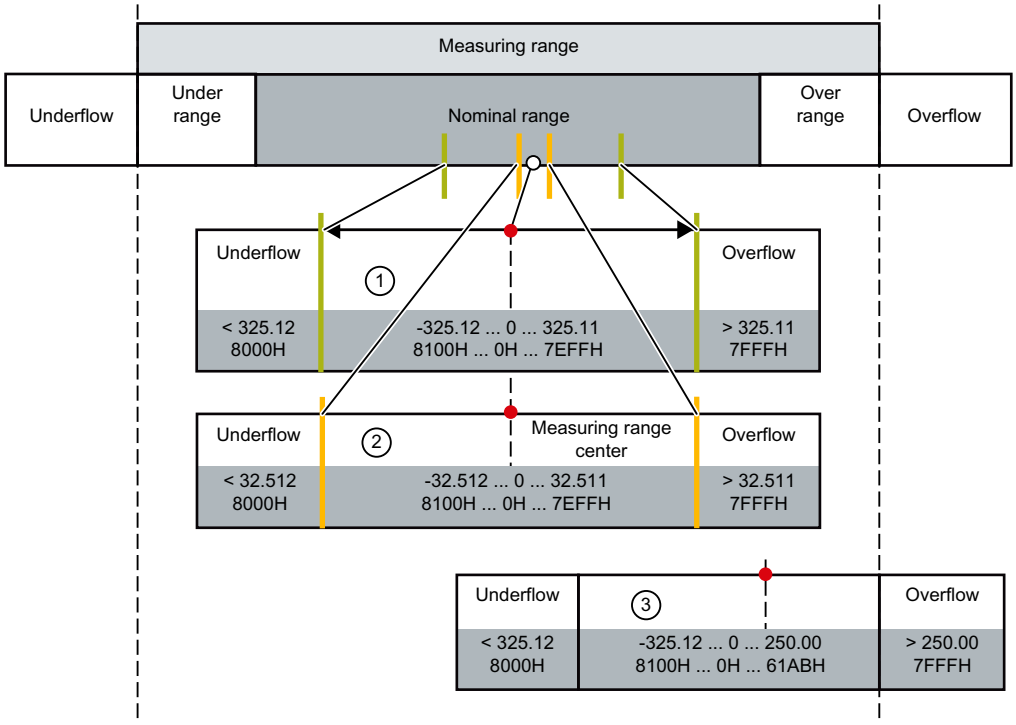

① Scalable measuring range with 2 decimal places, temperature values in S7 format.

② Scalable measuring range with 3 decimal places, temperature values in S7 format.

③ Scalable measuring range cut off at the overflow of the standard measuring range ("Clipping"). The sum of the measuring range center (for example 750 °C) and the measured value returned by the module must not extend into the overflow of the standard measuring range. For this reason, in the example above, the maximum value that the module can return is limited to 250 °C

Clipping

STEP 7 limits the maximum of the scalable measuring range so that the sum of the measured value center and the maximum measured value that the module can return is not located in the overflow of the standard measuring range. In the same way, STEP 7 restricts the minimum of the scalable measuring range.

See also

Special features of AI 4xRTD/TC 2-/3-/4-wire HF (Page 3246)

1.6.3.6 ET 200AL

ET 200AL distributed I/O system

SIMATIC ET 200AL

The SIMATIC ET 200AL distributed I/O system is a scalable and highly flexible, distributed I/O system for connecting process signals to a superordinate control with a field bus.

Properties

- Connection to PROFINET or PROFIBUS or integration into ET 200SP / ET 200SP CPU
- Up to 32 modules on an ET 200AL
- Integration into ET 200SP / ET 200SP CPU: Up to 16 AT modules can be connected to an ET 200AL
- Connection of modules via ET-Connection
- Spatially separated mounting possible
- Module widths of 30 and 45 millimeters
- Degree of protection IP65/IP67
- Suitable for temperatures from -25 to +55 °C and accelerations up to 5 g.
- Installation in all positions
- Color coding of the cables and connections
- CA-compliant labeling of the interfaces
- PROFIenergy integrated
- Configuration control
- Connection of sensors and actuators using M8 and M12 connection system

Area of application

The SIMATIC ET 200AL distributed I/O system is especially well suited for use in tight spaces, moving applications and for assembly and handling technique. Thanks to its scalable construction, you have the option precisely customize its configuration to your on site needs.

The SIMATIC ET 200AL distributed I/O system features protection type IP65/IP67 and is suited for distributed use on a machine or assembly line.

Structure

The SIMATIC ET 200AL distributed I/O system is made up of the following components:

- Interface modules (PROFINET/PROFIBUS)
- Digital and analog I/O modules
- Communications module

After an interface module you can configure 2 lines (ET-Connection), each with 16 modules.

Alternatively, you can configure a line with 16 I/O modules on the SIMATIC ET 200SP distributed I/O system with BusAdapter BA-Send 1xFC.

The ET-Connection backplane bus is designed as a cable. This allows you to create spatial distances of up to 10 m between the modules.

Configuration example

The figure below shows a configuration example of the SIMATIC ET 200AL distributed I/O system with a PROFINET interface module.

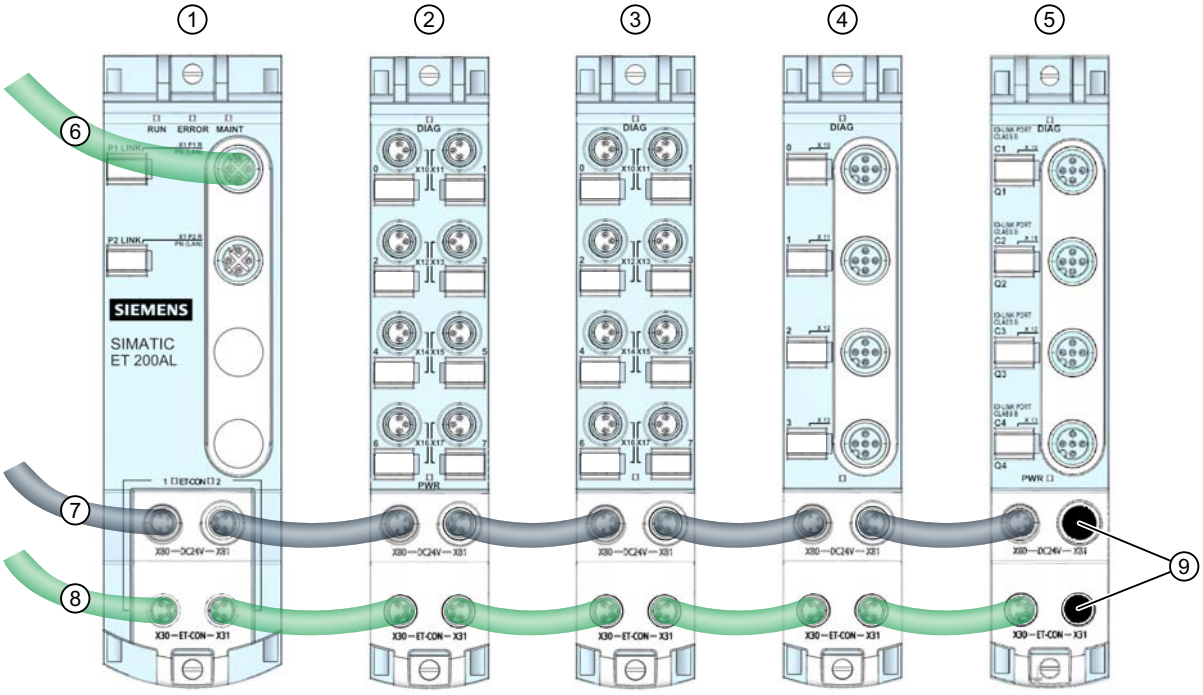

① Interface module (PROFINET)
② Digital input/Digital output module
③ Digital input module
④ Analog input module
⑤ Communications module
⑥ PROFINET cable
⑦ Power supply cable
⑧ ET-Connection line
⑨ Sealing caps

Figure 1-132 Example configuration of the ET 200AL

See also

Expanding ET 200SP with ET 200AL modules (Page 3227)

Configuring ET 200AL

Introduction

ET 200AL is a distributed I/O system with an IP65/67 degree of protection. It is therefore designed for use on site, for example right beside a machine (no control cabinet required).

The system includes interface and communications modules as well as input and output modules.

There are two use cases for the ET 200AL:

1. As I/O device or DP slave: The system interface module is connected to a field bus (PROFINET or PROFIBUS) and connected to the PN or DP interface of a CPU.
2. As an ET 200SP expansion: The ET 200AL modules are connected to the ET 200SP over the "BA Send 1xFC" module ("mixed mode").

The following sections describe how to configure an ET 200AL as I/O device or DP slave (use case 1).

For use case 2, see the link under "See also".

Procedure

Follow the steps below to configure an ET 200AL in STEP 7:

1. Copy an interface module (PROFINET or PROFIBUS) from the ET 200SP series to the network view (drag-and-drop operation from the hardware catalog).
2. Go to the device view. Double-click on the interface module you have just inserted.
   The interface module and two ET-Connection racks are displayed in the device view (figure below).
   There are no slot numbers assigned yet, which is why question marks are displayed above the slots.

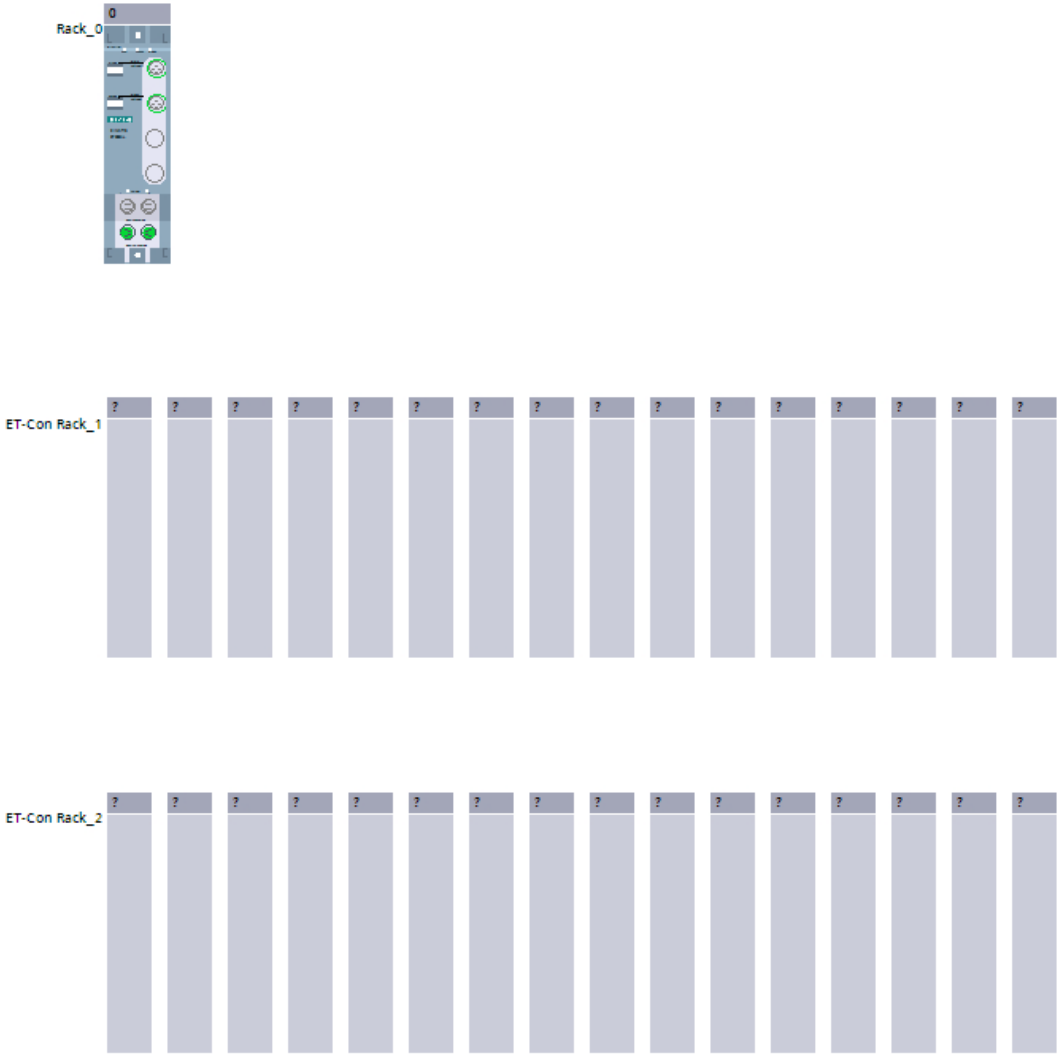

3. Now select the modules (input, output and communication modules) from the hardware catalog (ET 200AL folder) and drag them to the free slots (outlined in blue; not shown). You can place up to 16 ET 200AL modules in each ET-Connection rack (figure below). An ET-Connection rack is a virtual rack that sets the order of the connected ET 200AL modules.

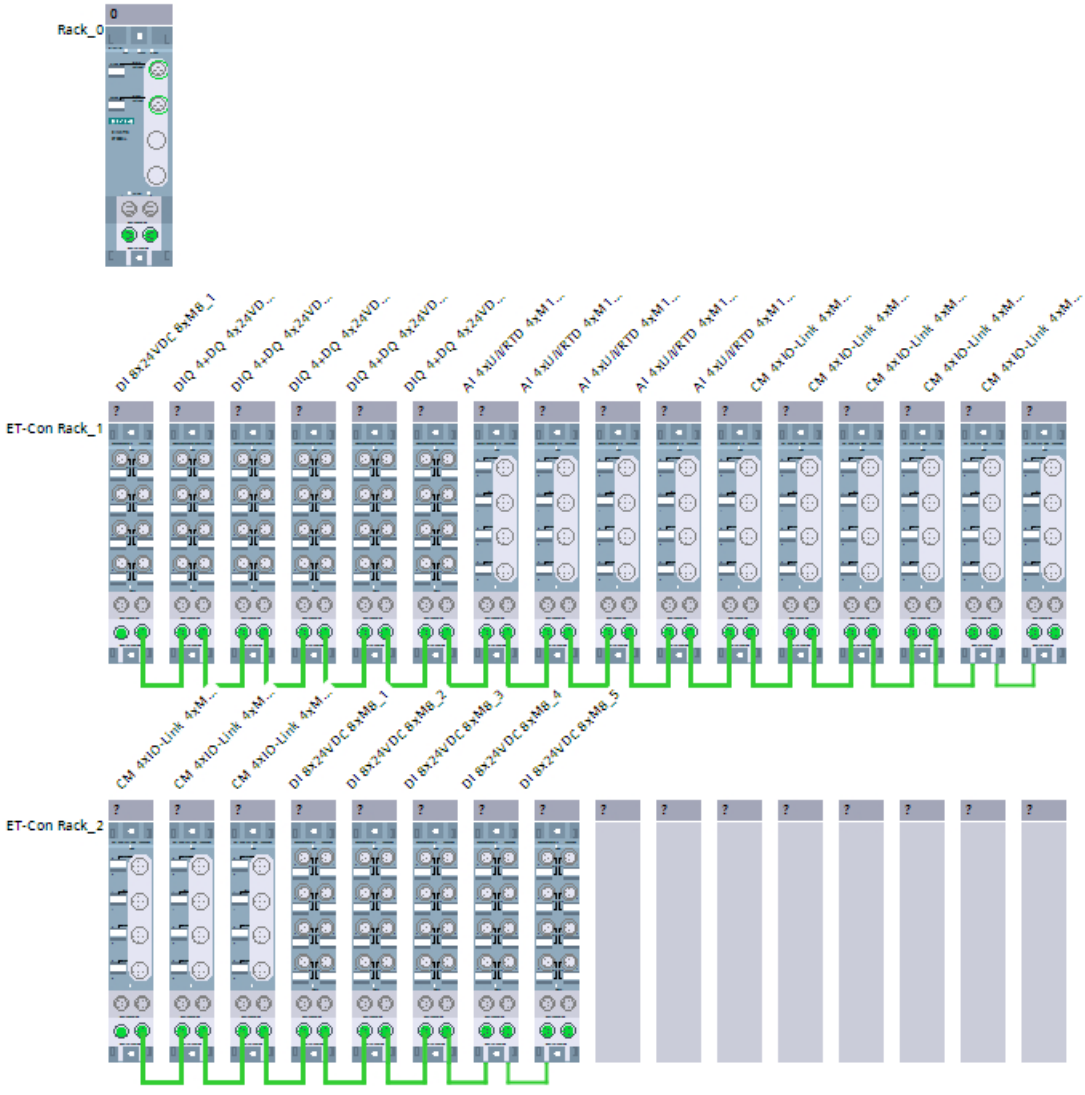

4. Now connect the interface module to the two ET-Connection racks.
   First click on an ET-Connection interface of the interface module, then hold and drag a line to the left-hand ET connection of the first module in one of the two ET-Connection racks. Repeat this step for the second ET-Connection interface of the interface module and the second ET-Connection rack (if used).

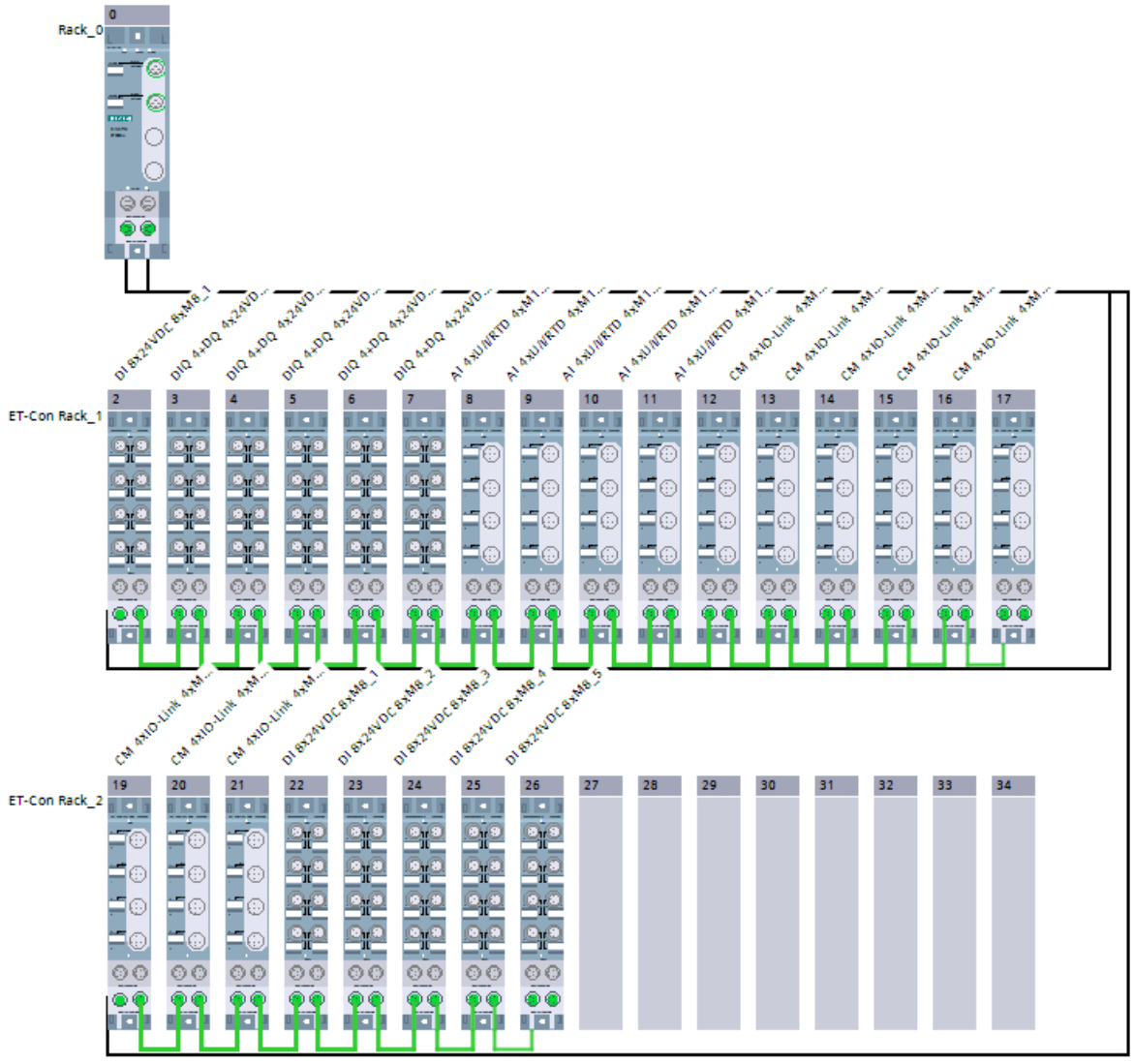

5. Double-click on a module to access the module properties and set the module parameters.

Rules

- The ET 200AL modules must be configured with no gaps.
- The first module in an ET-Connection rack must be connected to the interface module.

See also

ET 200AL distributed I/O system (Page 3255)

Configuration control with ET 200AL

Operating principle

With configuration control, you can change the original configuration of an ET 200AL (created by configuring with STEP 7) with a user program and operate the ET 200AL in this modified configuration. STEP 7 is no longer required for this configuration: You use your user program to signal to the ET 200AL the slot in which a configured module is actually inserted.

In this case, use control data record 196. In this data record, you code which modules are missing or located in different slots in the real configuration compared to the configuration with STEP 7. The configuration control has no effect on the parameter assignment of the modules (for example, the enabling of diagnostic alarms).

You then call the "WRREC" instruction and use it to write the data record to the interface module of the ET 200AL.

Configuration control gives you the flexibility to vary the configuration of an ET 200AL as long as the real configuration can be derived from a preset maximum configuration (originally created with STEP 7).

The following sections describe how to enable configuration control and how to structure the required data record 196 for the ET 200AL.

Requirement

The CPU startup parameter "Compare preset to actual configuration" is set to Startup even if mismatch (default setting). This setting is also selected for the startup parameters of the

Enabling configur

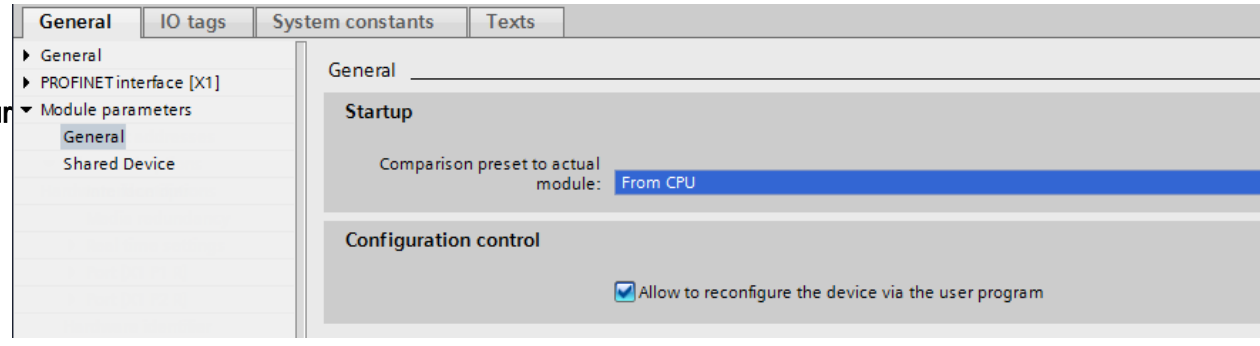

Structure of control data record 196

The configuration of the data block corresponds to the original configuration of the ET 200AL with STEP 7.

There are two bytes in the data record for each module. The position of these two bytes in the data record codes a module in the original configuration with STEP 7.

- Bytes 4 and 5 in the data record correspond to the module in slot 1 in the original configuration.
- Bytes 6 and 7 in the data record correspond to the module in slot 2 in the original configuration.
- Bytes 8 and 9 in the data record correspond to the module in slot 3 in the original configuration.
- etc.

The current (real) slot is coded by the number that is assigned to the "Slot_x" byte (by its value): Examples:

- The value "2" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 2 in the current configuration.
- The value "3" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 3 in the current configuration.
- The value "4" in byte 6 means you are assigning the module originally inserted in slot 2 to slot 4 in the current configuration.
- etc.

Creating control data record 196

The figure below shows part of control data record 196 for configuration of an ET 200AL.

This configuration is given as an example:

- ET-Con1 is inserted in "slot_1" (fixed setting). The two ET-Connection submodules "ET-Con1" and "ET-Con2" are submodules of the interface module of the ET 200AL. They are integrated as fixed modules in the IM module. They cannot be inserted individually.
- ET-Con2 is inserted in "slot_18" (fixed setting).
- In this configuration, 16 AL modules are connected to ET-Con1 ("slot_2" to "slot_17" in the following data record). This is the maximum configuration.
- In this configuration, one AL module is connected to ET-Con2 ("slot_19"). However, a total of 16 AL electronic modules could be connected to ET-Con2 (as to ET-Con1).

The ET 200AL originally configured with STEP 7 is now to be reconfigured from the user program.

The new configuration has the following properties:

- ET-Con1 is inserted in "slot_1" (fixed setting).
- Module 2 is also operated in slot 2 in the modified configuration.
- Module 3 is not used.
- Module 4 is now inserted in slot 3.
- Module 5 is now inserted in slot 4.
- No other modules at ET-Con1 are used.

- ET-Con2 is inserted in "slot_18" (fixed setting)
- The module at ET-Con2 is used.

Record_196_ET_200AL

	Name	Data type	Default value
1	block_length	USInt	42
2	block_ID	USInt	196
3	version	USInt	2
4	subversion	USInt	1
5	slot_1_ET_Con1	USInt	1
6	reserve_1_ET_Con1	USInt	0
7	slot_2	USInt	2
8	reserve_2	USInt	0
9	slot_3	USInt	0
10	reserve_3	USInt	0
11	slot_4	USInt	3
12	reserve_4	USInt	0
13	slot_5	USInt	4
14	reserve_5	USInt	0
15	slot_6	USInt	0
16	reserve_6	USInt	0
17	slot_7	USInt	0
18	reserve_7	USInt	0
19	slot_8	USInt	0
20	reserve_8	USInt	0
21	slot_9	USInt	0
22	reserve_9	USInt	0
23	slot_10	USInt	0
24	reserve_10	USInt	0
25	slot_11	USInt	0
26	reserve_11	USInt	0
27	slot_12	USInt	0
28	reserve_12	USInt	0
29	slot_13	USInt	0
30	reserve_13	USInt	0
31	slot_14	USInt	0
32	reserve_14	USInt	0
33	slot_15	USInt	0
34	reserve_15	USInt	0
35	slot_16	USInt	0
36	reserve_16	USInt	0
37	slot_17	USInt	0
38	reserve_17	USInt	0
39	slot_18 ET_Con2	USInt	1
40	reserve_18_ET_Con2	USInt	0
41	slot_19	USInt	1
42	reserve_19	USInt	0

The components of control data record 196 (definition in section "Control data record 196" below):

- block_length: Note the length of the control data record here; in the example: 42 (bytes). The length of the control data block is calculated using the following formula: 2 x "number of modules" + 4.
- block_ID: Enter the figure 196 here. This number identifies the data record as the data record for configuration control.
- version: The ET 200AL uses version 2 of control data record 196.
- subversion: The ET 200AL uses subversion 1 of control data record 196.
- slot_1: The ET-Connection 1 submodule is always inserted in slot 1 of the ET 200AL.
- reserve_1: This byte is not used (value "0").
- slot_2: The configured module 2 is inserted in slot 2 (value "2").
- reserve_2: This byte is not used (value "0").
- slot_3: The configured module 3 is not present in the current configuration (value "0").
- reserve_3: This byte is not used (value "0").
- slot_4: The configured module 4 is inserted in slot 3 in the current configuration (value "3").
- reserve_4: This byte is not used (value "0").
- slot_5: The configured module 5 is inserted in slot 4 in the current configuration (value "4").
- reserve_5: This byte is not used (value "0").
- slot_6: The configured module 6 is not present in the current configuration (value "0").
- reserve_6: This byte is not used (value "0").
- slot_7: The configured module 7 is not present in the current configuration (value "0").
- reserve_7: This byte is not used (value "0").
- etc.
- slot_18: The ET-Connection 2 submodule is always inserted in slot 18 of the ET 200AL (value "18").
- reserve_18: This byte is not used (value "0").
- slot_19: The configured module 19 is inserted in slot 19 in the current configuration (value "19").
- reserve_19: This byte is not used. (value "0")

Control data record 196 for ET 200AL

A control data record 196 containing a slot assignment is defined for configuration control.

Byte	Element	Value	Description	
0	Block length in bytes	e.g. 14 for ET 200AL with 5 modules	The length of the data record is calculated using the formula: 4 + (number of modules x 2) bytes	Header
1	Block ID	196	ID for control data record 196	
2	Version	2	Version 2 of control data record 196	
3	Subversion	1	Subversion 1 of control data record 196	
4	Configured module 1 (ET-Connection 1)	1	ET-Connection 1 is always assigned to slot 1 (fixed). The value "1" must therefore always be entered in byte 4.	Assignment for configured module 1 (ET-Connection 1) to real slot 1
5	Reserve for configured module 1	0	Not used	
6	Configured module 2	Real slot of module 2 Possible values: 2 up to the number of modules (except 18) 0 (if the configured module 2 is not present)	The configured module 2 can be inserted in any slot from 2 to slot 34. Slot 18 is reserved for ET-Con 2. If the configured module is not used, this byte contains the value "0".	Assignment for the configured module 2 to a real slot
7	Reserve for configured module 2	0	Not used	
8	Configured module 3	Real slot of module 3 Possible values: 2 up to the number of modules (except 18) 0 (if the configured module 2 is not present)	The configured module 3 can be inserted in any slot from 2 to slot 34. Slot 18 is reserved for ET-Connection 2. If the configured module is not used, this byte contains the value "0".	Assignment for the configured module 3 to a real slot
9	Reserve for configured module 3	0	Not used	
:	:	:	:	:
39	Configured module 18 (ET-Connection 2)	18	ET-Connection 2 is always assigned to slot 18 if AT modules are connected at this submodule.	Assignment for configured module 18 (ET-Connection 2) to real slot 18
40	Reserve for configured module 18	0	Not used	
:	:	:	:	:

(Bytes 4 to 70, not byte 39)	Configured module x	Real slot of module x Possible values: 2 up to the number of modules (except 18) 0 (if there is no configured module x)	The configured module x can be inserted in any real slot from 2 to slot 34. Slot 18 is reserved for ET-Con 2 (bytes 39 and 40 in the control data record)	Assignment for configured module x to a real slot y
(Bytes 5 to 71, not byte 40)	Reserve for configured module x	0	Not used	

Rules

- The ET-Connection 1 and ET-Connection 2 submodules must be treated like real modules in configuration control. Restriction: ET-Connection 1 is always placed in slot 1 and ET-Connection 2 is always placed in slot 18 (fixed assignment).
- There are no reserve modules for the ET 200AL (unlike for the ET 200S or with BU cover modules for the ET 200SP). For this reason, bit 7 of "slot_x" must not be set (i.e. only the values 0 to 127 may be used).
- The value "0" of "slot_x" indicates that this module is not inserted in the current configuration.
- Gaps must not be left between the AL modules when configuring with STEP 7.
- If no modules are connected to ET-Con2 when STEP 7 is configured, ET-Con2 is not configured: This shortens data record 196.
- If fewer than 16 modules are connected to ET-Con1 during configuration with STEP 7 and there are also modules connected to ET-Con2, control data record 196 must contain all unassigned slots for ET-Con 1. They are assigned zero as the value for the real slot.

Writing a data record

Transfer the control data record to the ET 200AL module.

To do so, call the extended WRREC (Write data record) instruction, and transfer the control data record created.

If you do not transfer a control data record, the interface module uses the original configuration with STEP 7. Here, the following applies: Configured module x is inserted in real slot x.

Addressing the interface module using the HW identifier

To transfer data record 196 with the instruction WRREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Error messages

The following error messages are returned if an error occurs when writing control data record 196:

Table 1-243 Error messages

Error code	Meaning
16#80A2	DP protocol error on layer 2 Indicates that a data record has not been acknowledged due to the system.
16#80B1	Invalid length; the length information in data record 196 is not correct.
16#80B5	Configuration control parameters not assigned.
16#80B2	Invalid slot: The configured slot is not assigned.
16#80B8	Parameter error; module signals invalid parameters.
16#80C5	DP slave or module not available. Indicates that a data record has not been acknowledged due to the system.

Feedback data record 197 for ET 200AL

The feedback data record 197 is used to read the actual configuration of a station (in this case of an ET 200AL).

This data record allows you to check the real configuration of the ET 200AL (actual configuration). The feedback data record for each configured module specifies whether or not it is actually present.

- The value "1" means that the correct module is inserted in the correct slot.
- The value "0" codes all other possibilities (wrong module, empty slot, BU cover).

Example:

A module has been configured with STEP 7 for slot 4.

This module has then been moved to slot 3 in the current configuration using data record 196.

If this module is also really in slot 3, this is coded by the value "1" (status_slot_4 = 1).

Configuration details:

The configuration of the data block corresponds to the original configuration of the ET 200AL with STEP 7.

There are two bytes in the data record for each module. The position of these two bytes in the data record corresponds to the position of a module in the original configuration with STEP 7.

Sequence of bytes:

- "status_slot_1_ET_Con1" and "reserve_slot_1_ET-Con1" (bytes 4 and 5 in the data record) correspond to the module in slot 1 in the configuration,
- "status_slot_2" and "reserve_slot_2" (bytes 6 and 7) correspond to the module in slot 2 in the configuration
- "status_slot_3" and "reserve_slot_3" (bytes 8 and 9) correspond to the module in slot 3 in the configuration,
- etc.

Example

The following feedback data record 197 is returned by the ET 200AL that was reconfigured with control data record 196 in the example above (section "Creating control data record 196").

Record_197_ET_200AL

	Name	Data type	Default value
1	block_length	USInt	42
2	block_ID	USInt	197
3	version	USInt	2
4	subversion	USInt	1
5	status_slot_1_ET_Con1	USInt	1
6	reserve_1_ET_Con1	USInt	0
7	status_slot_2	USInt	1
8	reserve_2	USInt	0
9	status_slot_3	USInt	0
10	reserve_3	USInt	0
11	status_slot_4	USInt	1
12	reserve_4	USInt	0
13	status_slot_5	USInt	1
14	reserve_5	USInt	0
15	status_slot_6	USInt	0
16	reserve_6	USInt	0
17	status_slot_7	USInt	0
18	reserve_7	USInt	0
19	status_slot_8	USInt	0
20	reserve_8	USInt	0
21	status_slot_9	USInt	0
22	reserve_9	USInt	0
23	status_slot_10	USInt	0
24	reserve_10	USInt	0
25	status_slot_11	USInt	0
26	reserve_11	USInt	0
27	status_slot_12	USInt	0
28	reserve_12	USInt	0
29	status_slot_13	USInt	0
30	reserve_13	USInt	0
31	status_slot_14	USInt	0
32	reserve_14	USInt	0
33	status_slot_15	USInt	0
34	reserve_15	USInt	0
35	status_slot_16	USInt	0
36	reserve_16	USInt	0
37	status_slot_17	USInt	0
38	reserve_17	USInt	0
39	status_slot_18 ET_Con2	USInt	1
40	reserve_18_ET_Con2	USInt	0
41	status_slot_19	USInt	1
42	reserve_19	USInt	0

Modules 2, 4 and 5 are actually connected to ET-Con1.

None of the other modules that were connected to ET-Con1 in the configuration with STEP 7 are present in the current configuration (in line with the control data record 196 settings from the example above).

A module is really connected to ET-Con2 as in the original configuration with STEP 7.

Reading feedback data record 197

You can read the feedback data record 197 from the ET 200AL with the RDREC instruction. RDREC operates asynchronously. If you call RDREC in the startup OB, you must call the instruction multiple times using a loop until the "BUSY" or "DONE" output parameter indicates that the data record has been read.

To read data record 197 with the instruction RDREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Additional information and examples

Further information on ET 200AL is available here (http://support.automation.siemens.com/WW/view/en/89254863).

Specific examples of configuration control can be found here in this application description (http://support.automation.siemens.com/WW/view/en/29430270).

You will find a library of the data records and other application examples here: Templates for the data records (http://support.automation.siemens.com/WW/view/en/29430270)

See also

ET 200AL distributed I/O system (Page 3255)

Expanding ET 200SP with ET 200AL modules (Page 3227)

Configuration control with ET 200SP (Page 3215)

1.6.3.7 ET 200MP

ET 200MP distributed I/O system

Definition

The ET 200MP distributed I/O system is a scalable and flexible distributed I/O system for connection of process signals to a central controller via a field bus.

Application area

The ET 200MP is a multi-functional distributed I/O system for various fields of application. The scalable design allows you to configure the system exactly to the specific requirements on location.

The ET 200MP complies with IP 20 degree of protection and is intended for installation in a control cabinet.

Structure

The ET 200MP is installed on a mounting rail and comprises:

- An interface module that communicates with all IO controllers conforming to the PROFINET standard IEC 61158
- Up to 30 modules (power supply modules and I/O modules from the S7-1500 I/O range) can be inserted to the right of the interface module.
- If you insert a power supply module to the left of the interface module, this yields a possible maximum configuration of 32 modules in total.
- The number of insertable I/O modules is limited by their power requirements.

Slot rules

- Slot 0: Power supply module (optional)
- Slot 1: Interface module
- Slot 2 to 31: I/O modules or power supply modules

Interface module parameters

Supply voltage L+ connected

Parameter "Supply voltage L+ connected"

This parameter influences the diagnostics and the checking of the power budget.

- Diagnostics of the ET 200MP:
  If the actual configuration does not match the preset configuration with regard to the supply voltage of the interface module, the interface module generates a diagnostic alarm.
  Example: You have deactivated the "Supply voltage L+ connected" option, but you have connected the supply voltage in the actual configuration.
- Power budget check during configuration:
  The power budget changes in accordance with the parameter setting: Either the interface module feeds power into the backplane bus or it draws power from the backplane bus.

The default ("Supply voltage L+ connected" option is **activated**) means that the front of the interface module is supplied with 24 V DC and the power is stored in the backplane bus.

If the "Supply voltage L+ connected" option is **deactivated**, the interface module may not be supplied with 24 V DC on the front.

In this case, insert a power supply unit (PS) on the left next to the interface module that supplies the interface module and the modules to the right of the interface module.

Note

We recommend that you always supply the interface module on the front side with 24 V DC. If a system power supply unit (PS) is inserted and connected additionally **before** or on the left next to the interface module, both the power from the system power supply unit (PS) as well as the power from the integrated power supply of the interface module are then available to the configuration.

In this case, you do not have to change the default setting of the parameter.

Configuration control (handling options) with ET 200MP

Operating principle

Configuration control allows you to operate various real configurations (options) with a single configuration of the distributed I/O device ET 200MP.

Configuration control provides you with the option of configuring the ET 200MP distributed I/O device with its maximum configuration and still operating it with modules missing. If missing modules are retrofitted later, no new configuration is required and the hardware configuration does not have to be reloaded either.

Using control data record 196, which is transferred to the interface module in the user program, you define a current configuration. You transfer the control data record with the instruction WRREC.

Readback data record 197 is used to read the actual configuration of an ET 200MP.

Requirements

The CPU startup parameter "Compare preset to actual configuration" is set to Startup even if mismatch (default setting). This setting is also selected for the startup parameters of the individual modules of the ET 200MP.

Enabling configuration control

In the properties of the interface module under Module parameters > General > Configuration control, select "Enable reconfiguration of device via user program". This activates configuration control.

Control data record 196 for ET 200MP

The figure below shows the start of control data record 196 for the configuration control of an ET 200MP.

The data block is 36 bytes long (maximum configuration with 32 modules). The value "36" therefore appears in the "block_length" element of the data record.

If you configure an ET 200MP in STEP 7 with fewer modules, the data block will be shorter: If there are only five modules, for example, the data record is reduced to 9 bytes (4 bytes for the header plus one byte for each module).

There is one byte in the data record for each module. The position of this byte in the data record codes a module in the original configuration with STEP 7:

- "slot_0 power supply" (byte 4 in the data record below) corresponds to the power supply module in slot 0 in the configuration with STEP 7.
- "slot_2" (byte 5 in the data record) corresponds to the module in slot 2 in the configuration.
- "slot_3" (byte 6 in the data record) corresponds to the module in slot 3 in the configuration.
- "slot_4" (byte 7) corresponds to the module in slot 4 in the configuration.
- etc.

Note

Interface module in slot 1

Configuration control is controlled by the interface module (slot 1/submodule 1). The interface module in slot 1 is therefore not an element of the configuration control and is not included in the data record.

Value in slot_x

The current slot is coded by the figure that is assigned to "slot_x" (by its value). Examples:

- The value "2" in slot_2 means you are assigning the module originally inserted in slot 2 to slot 2 in the current configuration (slot_2 = 2).
- The value "3" in slot_2 means you are assigning the module originally inserted in slot 2 to slot 3 in the current configuration (slot_2 = 3).
- The value "4" in slot_2 means you are assigning the module originally inserted in slot 2 to slot 4 in the current configuration (slot_2 = 4).
- etc.

Example for data record 196

The following data record was created for a configuration that changes the original configuration with STEP 7.

The modified configuration has the following properties:

- The module inserted in slot 0 in the configuration (power supply module) is also inserted in slot 0 in the current configuration (specification).
- The module inserted in slot 2 in the configuration (module 2) is also inserted in slot 2 in the current configuration.
- The module inserted in slot 3 in the configuration (module 3) does not exist in the current configuration.
- The module inserted in slot 4 in the configuration (module 4) is inserted in slot 3 in the current configuration.

- The module inserted in slot 5 in the configuration (module 5) is inserted in slot 4 in the current configuration.
- etc.

The bytes "slot_6" to "slot_31" are not shown in the figure below.

Record_196_ET_200MP

	Name	Data type	Default value
1	Block_length	USInt	36
2	Block_ID	USInt	196
3	version	USInt	3
4	subversion	USInt	0
5	slot_0 power supply	USInt	0
6	slot_2	USInt	2
7	slot_3	USInt	127
8	slot_4	USInt	3
9	slot_5	USInt	4

Rules

- If a module dos not exist in the current configuration, this is indicated by the value 127: "slot_x" = 127.
- The power supply module is always in slot 0 ("slot_0 power supply" = 0).
- The interface module in slot 1 is not included in the control data record.

Addressing the interface module using the HW identifier

To transfer data record 196 with the instruction WRREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Readback data record 197 for ET 200MP

Readback data record 197 is used to read the actual configuration of a station (in this case, of an ET 200MP).

This data record allows you to check the real configuration of the ET 200MP (actual configuration). The readback data record for each configured module specifies whether or not it is actually available.

- The value "1" means that the correct module is inserted in the correct slot.
- The value "0" codes all other options (wrong module, empty slot, reserve module).

Example:

A module has been configured with STEP 7 for slot 4.

This module has then been moved to slot 3 in the current configuration using data record 196.

If this module is also really in slot 3, this is coded by the value "1" (status_slot_4 = 1).

Configuration details:

The configuration of the data block corresponds to the original configuration of the ET 200MP with STEP 7.

There is a byte in the data record for each module. The position of this bytes in the data record corresponds to the position of a module in the original configuration with STEP 7.

Sequence of bytes:

- "status_slot_0 power supply" (byte 4 in the data record below) corresponds to the power supply module in slot 0 in the configuration with STEP 7.
- "status_slot_2" (byte 5) corresponds to the module in slot 2 in the configuration.
- "status_slot_3" (byte 6) corresponds to the module in slot 3 in the configuration,
- etc.

You can choose any name for the components (for example "status_slot_2").

Meaning of "status_slot_x":

- The value "1" in status_slot_x means that module x is in the correct slot.
- The value "0" in status_slot_x codes all other possibilities (wrong module, module does not exist).

Example:

The figure below shows readback data record 197 for the configuration of an ET 200MP in which there is no module 3 (the module in slot 3 in the configuration).

All other modules are available and correctly plugged.

The bytes "status_slot_6" to "status_slot_31" are not shown in the figure below.

Record_197_ET_200MP

	Name	Data type	Default value
1	Block_length	USInt	36
2	Block_ID	USInt	197
3	version	USInt	3
4	subversion	USInt	0
5	status_slot_0 power supply	USInt	1
6	status_slot_2	USInt	1
7	status_slot_3	USInt	0
8	status_slot_4	USInt	1
9	status_slot_5	USInt	1

Reading readback data record 197

You can read readback data record 197 from the ET 200MP with the instruction RDREC. RDREC operates asynchronously. If you call RDREC in the startup OB, you must call the instruction multiple times using a loop until the "BUSY" or "DONE" output parameter indicates that the data record has been read.

To read data record 197 with the instruction RDREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Additional information and examples

Further information on ET 200MP can be found in the manual for IM 155-5 PN (http://support.automation.siemens.com/WW/view/en/89261636).

Specific examples of configuration control can be found in this application description (http://support.automation.siemens.com/WW/view/en/29430270).

You will find a library of the data records and other application examples here: Templates for the data records (http://support.automation.siemens.com/WW/view/en/29430270)

See also

Documentation on configuration control (http://support.automation.siemens.com/WW/view/en/67295970)

Input module parameters

Parameters of the analog input modules

Missing supply voltage L+

Enabling of the diagnostics for missing or insufficient supply voltage L+.

Wire break

Enabling of the diagnostics if the module has no current flow or the current is too weak for the measurement at the corresponding configured input or the applied voltage is too low.

Current limit for wire break diagnostics

Threshold at which a wire break is reported. The value can be set to 1.185 mA or 3.6 mA, depending on the sensor used.

Overflow

Enabling of the diagnostics if the measured value exceeds the overrange.

Underflow

Enabling of the diagnostics if the measured value undershoots the underrange.

Common mode error

Enable diagnostics if the valid common mode voltage is exceeded.

Reference channel error (only for AI 8xU/I/RTD/TC ST)

- Enable diagnostics on error at the temperature compensation channel, e.g. wire break.
- Dynamic reference temperature compensation type is configured and no reference temperature has been transferred to the module yet.

Temperature coefficient

The temperature coefficient depends on the chemical composition of the material. In Europe, only one value is used per sensor type (default value).

The correction factor for the temperature coefficient (α value) specifies how much the resistance of a certain material changes when the temperature is raised by 1 °C.

The further values facilitate a sensor-specific setting of the temperature coefficient and enhance accuracy.

Interference frequency suppression

At analog input modules, this suppresses interference caused by the frequency of AC mains.

The frequency of the AC network may interfere with measured values, particularly for measurements within low voltage ranges and when thermocouples are being used. With this parameter, you define the mains frequency in your system.

Smoothing

The individual measured values are smoothed using filtering. Smoothing can be set in 4 stages for the analog input modules AI 8xU/I/RTD/TC ST and AI 8xU/I HS.

Smoothing time = number of module cycles (k) x cycle time of the module.

The figure below shows the number of module cycles after which the smoothed analog value is almost 100%, depending on the set smoothing. Is valid for each signal change at the analog input.

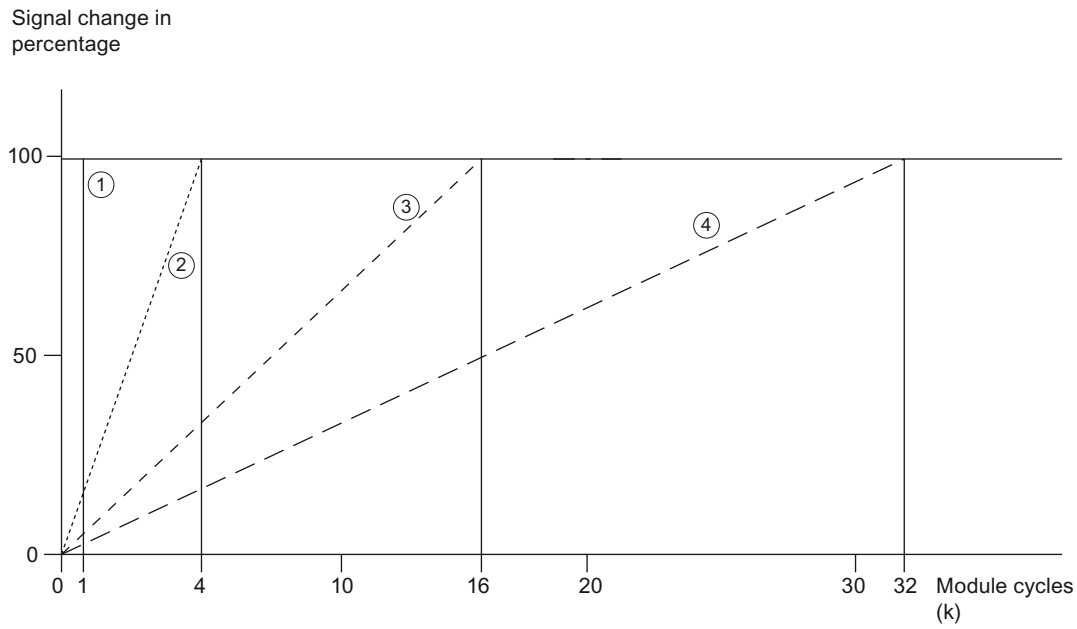

① None (k = 1)
② Weak (k = 4)
③ Medium (k = 16)
④ Strong (k = 32)

Reference junction (only for AI 8xU/I/RTD/TC ST)

The following settings can be configured for the reference junction parameter:

Table 1-244 Possible parameter assignments for the reference junction parameter

Setting	Description
Fixed reference temperature	The reference junction temperature is configured and stored in the module as a fixed value.
Dynamic reference temperature	The reference junction temperature is transferred in the user program from the CPU to the module by data records 192 to 199 using the WRREC (SFB 53) instruction.
Internal reference junction	The reference junction temperature is determined using an integrated sensor of the module.
Reference channel of the module	The reference junction temperature is determined using an external resistance thermometer (RTD) at the reference channel (COMP) of the module.

Note

Fixed reference temperature

During parameter assignment of a thermocouple Type B, only the setting "Fixed reference temperature" with a temperature of 0 °C is possible.

Enable hardware interrupt 1 or 2

Enable a hardware interrupt if high limit 1 or 2 is exceeded, or low limit 1 or 2 is violated.

Low limit 1 or 2

Specifies the low limit threshold that triggers hardware interrupt 1 or 2.

High limit 1 or 2

Specifies the high limit threshold that triggers hardware interrupt 1 or 2.

Temperature compensation for thermocouples

Introduction

You have several options of measuring the reference junction temperature in order to obtain an absolute temperature value as a function of the temperature difference between the reference junction and the measuring point.

You can use various compensation options depending on the required location of the reference junction.

Note

During parameter assignment of a thermocouple Type B, only the setting "Fixed reference temperature" with a temperature of 0 °C is possible.

Options of compensating for the reference junction temperature

Compensation options	Explanation	Application case
Internal reference junction	With this compensation, the reference junction temperature is determined using an integrated sensor of the module. **Procedure** Connect the thermocouple to the I/O module directly or with compensating lines.	• For the connection, you use compensating lines matching the thermocouple material. • If the reference junction temperature and the module temperature are identical in your system, you may also use lines made from a different material.
Reference channel of the module	The reference junction temperature is determined using an external resistance thermometer (RTD). **Procedure** Connect the thermocouple to the supply lines at the reference junction, either directly or with compensating lines. You connect the supply lines to the appropriate terminals of the module. Connect the resistance thermometer (RTD) to the reference channel of the module. The resistance thermometer (RTD) must be placed in the area of the reference junction.	• You want to measure the temperature directly at the reference junction. • The measured temperatures of all channels that you have configured for this compensation type is corrected automatically by the temperature value of the reference junction. • You can use inexpensive lines, e.g., copper lines, from the reference junction to the module.
Dynamic reference temperature	The temperature of the reference junction is determined via a module. This temperature value is transferred to other modules via a data record in the user program. **Procedure** Connect the resistance thermometer (RTD) for the reference junction to any channel. The reference junction temperature is communicated from the CPU to the module by data records 192 to 199 using the WRREC instruction.	• You use multiple modules at the reference junction and can therefore compensate all channels using a common temperature value. • You require only one resistance thermometer (RTD) to acquire the temperature value. • You can use inexpensive lines, e.g., copper lines, from the reference junction to the module.
Fixed reference temperature	The reference junction temperature is stored in the module as a fixed value. **Procedure** Connect the thermocouple to the supply lines at the reference junction, either directly or with compensating lines. You connect the supply lines to the appropriate terminals of the module. When configuring the module, specify a fixed temperature value for the reference junction (e.g. 20 °C).	• You keep the reference junction temperature constant and know the temperature value. • You can use inexpensive lines, e.g., copper lines, from the reference junction to the module.

Output module parameters

Parameters of the analog output modules

Missing supply voltage L+

Enabling of the diagnostics, with missing or too little supply voltage L+.

Short-circuit to ground

Enabling of the diagnostics if a short-circuit of the actuator supply to ground occurs.

Wire break

Enabling diagnostics if the line to the encoder is interrupted.

Overflow

Enabling of the diagnostics if the measured value exceeds the overflow range.

Underflow

Enabling of the diagnostics if the measured value falls below the underflow range.

Reaction to CPU STOP

Determines the reaction of the output to the CPU going into STOP state.

Substitute value

The substitute values are values that the outputs (the output) issue in the event of a CPU STOP.

Pulse-width modulation (PWM)

The channels of some modules support the pulse width modulation (PWM) function. You can use the pulse width modulation function to easily generate periodic pulses with a constant rated voltage and a variable pulse duration for the above-mentioned channels.

Configuration

You configure the pulse width modulation with the following parameters:

- Enable PWM function on module
- Set PWM operating mode for the corresponding channel
- Set period

How it works

In the pulse width modulation mode, the correspondingly configured outputs provide a pulse width modulated output signal.

Pulse width modulation is characterized by its time period (frequency) and its duty factor. The duty factor describes the relation between pulse duration and time period.

The pulse duration is derived from the time period and the duty factor: Pulse duration = duty factor x time period.

Example for duty factor of 50% and time period of 10 ms:

Pulse duration 0.5 x 10 ms = 5 ms

You define the duty factor of the channels in the user program using the output value (0 ... 1000) in the process image output.

The output signal is a square wave signal (pulse sequence of on and off pulses).

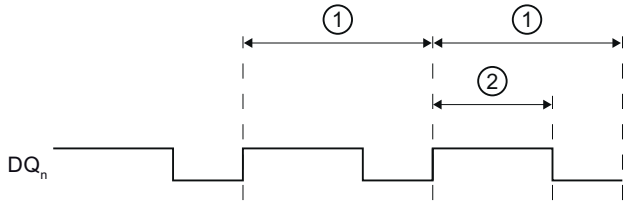

① Time period T (2 to 100 ms); Frequency of the pulse width modulation: f = 1/T (10 to 500 Hz)

② Pulse duration (duty factor x time period)

Figure 1-133 How pulse width modulation works

Minimum pulse duration

The minimum pulse duration is 300 µs due to the hardware. The duty factor can be adjusted from 0.0 to 100.0%. The time period can be adjusted from 2 to 100 ms.

Example: If you configure a time period of 2 ms and set a duty factor of 0.1% for the output, this would result in a pulse duration of 200 µs. In fact, the output works with a minimum pulse duration of 300 µs.

Pulse waveform

The pulse duration of the actual signal profile is slightly longer than the specified, ideal pulse duration.

The figure below shows the reaction of the output to control by PWM. The blue line shows the specified, ideal signal profile (square wave signal), with which the output is controlled. The red dashed line show the actual signal profile on the output graph, caused by the externally connected load.

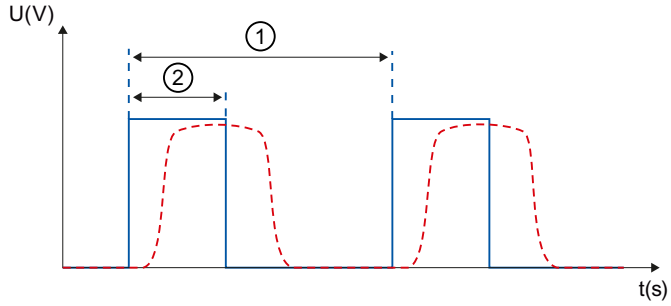

① Time period

② Pulse duration (duty factor x time period)

Figure 1-134 Pulse waveform at output terminal

1.6.3.8 ET 200M

Configuring an ET 200M

Introduction

For the ET 200M series, you can find a wide range of modules in the hardware catalog under "Distributed I/O".

Configuration and parameter assignment

Information on configuration and parameter assignment can be found in the following sections.

ET 200M configuration

Definition

The distributed IO device, ET 200M, is a modular DP slave with an IP 20 degree of protection.

The ET 200M has the configuration technology of the S7-300 automation system and consists of an IM 153-x and I/O modules of the S7-300.

ET 200M supports communication with:

- all DP masters compliant with IEC 61784-1:2002 Ed1 CP 3/1
- all IO controllers compliant with IEC 61158

Configuration of the ET 200M (example)

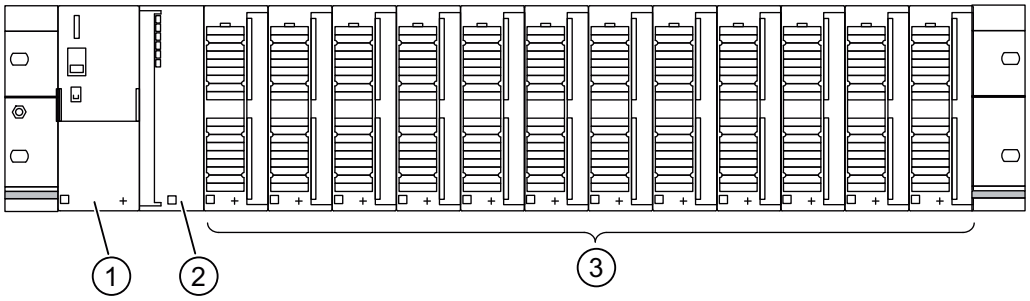

① Power supply module PS 307
② Interface module IM 153-x
③ up to 12 I/O modules (SM / FM / CP)

Configuration of the 'Module replacement during operation' function

Introduction

The ET 200M supports the "Replace modules during operation" function and the associated pull/plug interrupt.

The "Replace modules during operation" function makes it possible for you to pull modules from or plug modules into the ET 200M rack during operation.

Requirement

You have configured an interface module that supports replacing modules during operation. (as of IM 153-1, article no. 153-1AA02-0XB0).

In addition, the configured CPU must also support the function, e.g. for PROFIBUS an S7-400 with DP interface.

You must use the active backplane bus (bus rail with slots) for the hardware configuration. The conventional profile rail with bus connectors between the modules does not support this function.

Configuring

If the configuration requirements have been met, the "Replace modules during operation" parameter is available for selection in the inspector window's "Module parameters" area. Below this parameter, a table for the configured modules is displayed, which shows the required active bus modules for the hardware configuration.

For a PROFIBUS configuration, the "Startup if preset configuration does not match actual configuration" option is displayed. This option is automatically enabled if "Replace modules during operation" is enabled.

Signal modules for process automation

Fundamentals

Introduction

Signal modules for the process automation are S7-300 models, such as SM 321; DI 16xNAMUR or SM 322; DO 16x24VDC/0.5A.

They are being operated in a DP slave (IM 153-2).

Unlike standard modules, they offer the following additional technical functions, such as pulse extension and chatter monitoring.

See also

Changeover contact (Page 3285)

Technological parameters (Page 3286)

Changeover contact

"Changeover contact" sensor type

If the digital inputs of a channel group are configured as "changeover contacts", the module runs diagnostics for the changeover contact sensor type for this channel group.

Changeover contact

A changeover contact is an auxiliary switch with only one moving switch element with one close setting each for closed and open switching device.

Remember the following rule:

- Always connect a normally open contact to the "even" channel
- Always connect a normally closed contact to the "odd" channel.

The tolerated switchover time between the two channels is fixed at 300 ms.

If the result of the check is negative, then

- the module identifies the value status of the normally open channel as "invalid"
- the module generates a diagnostic entry for the normally open channel
- triggers a diagnostic interrupt (if diagnostic interrupts have been enabled)

The digital input signal and the value status are updated only for the normally open channel. For the normally closed channel, the digital input signal is set permanently to "zero" and the value status to "invalid" since this channel is used only to check the sensor.

Diagnostics depends on the "Selection" parameter (of the sensor). You should also note the special features of diagnostics with the changeover contact sensor type in the "Signal Modules for Process Automation" manual.

See also

Documentation on modules for process automation (http://support.automation.siemens.com/WW/view/de/7215812/0/en)

Technological parameters

Pulse extension and flutter monitoring

Pulse extension is a function for changing a digital input signal. A pulse at a digital input is extended to at least the length set in the parameters. If the input pulse is already longer than the specified length, it is not changed.

If you want the pulse to be extended, click in the box to select the time. If you do not want the pulses to be extended, select the "---" entry.

Flutter monitoring is a process control function for digital input signals. It detects and reports signal changes that are unexpected in process control, for example when an input signal fluctuates too often between "0" and "1".

Flutter monitoring is possible only when group diagnostics has also been enabled for this input.

Monitoring window and number of signal changes

Flutter monitoring works with aid of the two parameters Monitoring window and Number of signal changes.

The first time the signal changes, the time set as the monitoring window is started. If the signal changes more often during this time than allowed by the number of signal changes parameter, this is signaled as a flutter error. If no flutter error is detected during the monitoring window time, the monitoring window can be restarted at the next signal change.

Note

If you set pulse extension for an input channel, this also affects the flutter monitoring enabled for this channel. The "extended pulse" signal is the input signal for the flutter monitoring. You should therefore make sure that the values set for pulse extension and flutter monitoring are compatible with each other.

See also

Documentation on modules for process automation (http://support.automation.siemens.com/WW/view/de/7215812/0/en)

IQ Sense module

Properties of 8 IQ-SENSE

Properties

The 8 IQ-SENSE module has the following properties:

- Connection of sensors with IQ-SENSE®, photoelectric proximity switches: for example, reflex sensors, diffuse sensors, and laser sensors.
- It can be used centrally in an S7-300 or distributed in an ET 200M.
- You can connect up to 8 sensors to every module. Each sensor requires a two-wire cable.
- Function reserve that can be assigned parameters.
- Time functions, switching hysteresis, synchronous mode that can be assigned parameters
- Sensitivity and distance values can be specified (*IntelliTeach* using the "IQ-SENSE Opto" FB)
- Teach-in
- Sensors can be removed and inserted during operation (automatic reassignment of parameters)

Anti-interference group

Only for optical IQ Sense devices (IQ profile ID 1).

For IQ Sense devices with IQ profile ID 128 (ultrasound), see "Multiplex/synchronous mode" under the channel-specific parameters.

Prevention of interference (e.g., scattered light) by assigning an anti-interference group. This means:

- Anti-interference group: None (= default)
  Optical sensors on one or more modules can mutually influence each other when unfavorably arranged.
- Anti-interference group: 3 or 4
  Optical sensors on the same module with anti-interference group 3 or 4 cannot mutually influence each other. Similarly, optical sensors on different modules with anti-interference group 3 or 4 cannot mutually influence each other. You need not maintain minimum clearance between the IQ Sense devices and can, for example, align two retroreflective sensors on a single reflector.

Operating principle

The diagram below explains the functioning of the anti-interference group parameter:

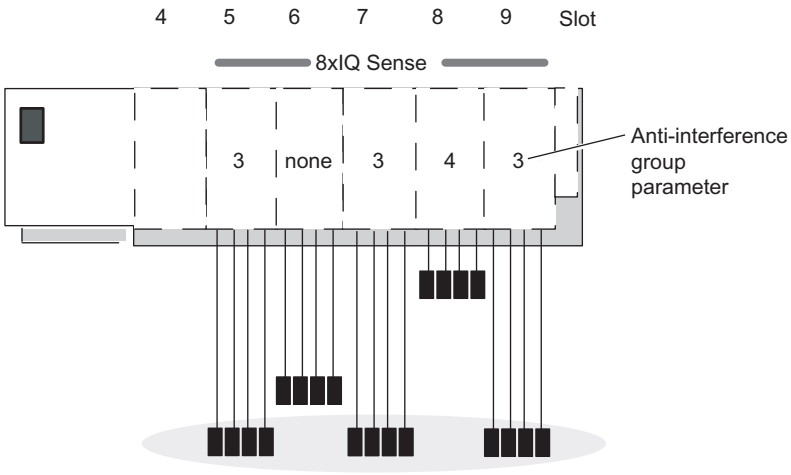

Mutual interference is only possible between the optical sensors of the modules in slot 5, 6, 7 and 9 because they are in the same anti-interference group 3 or "None" is set.

Note

Sensors in the same anti-interference group must be installed to maintain the minimum clearance (see sensor package insert) and to prevent mutual interference.

Encoder type

This parameter is used to set the sensor type per channel:

- Reflex sensor or
- Diffuse sensor or
- Disabled

Diffuse sensor

Table 1-245 Diffuse sensor

Diffuse sensor	Object	
Transmitter Receiver		Circuit state 0: No object detected, which means the object is not in the beam. The receiver does not see any light.
Transmitter Receiver	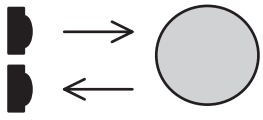	Circuit state 1: Object detected, which means the object is in the beam. The receiver does not see any light.

Reflex sensor

Table 1-246 Reflex sensor

Reflex sensor	Object	
Transmitter Receiver	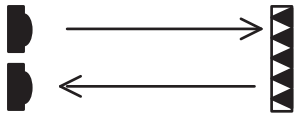	Circuit state 0: No object detected, which means the object is not in the beam. The receiver sees light.
Transmitter Receiver	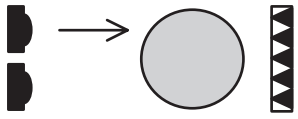	Circuit state 1: Object detected, which means the object is in the beam. The receiver does not see any light.

Switching hysteresis

Faults with the diffuse sensor or in the production process can result in signal wobbles. The measured value then changes the switching threshold by 100 % (object detected - object not detected). You can prevent this switching threshold wobble using the switching hysteresis parameter. This will ensure a stable output signal on the sensor.

You can assigned parameters to 5 %/10 %/20 %/50 % for switching hysteresis.

Requirements

You can only set the switching hysteresis parameter for diffuse sensors with background fadeout.

Operating principle

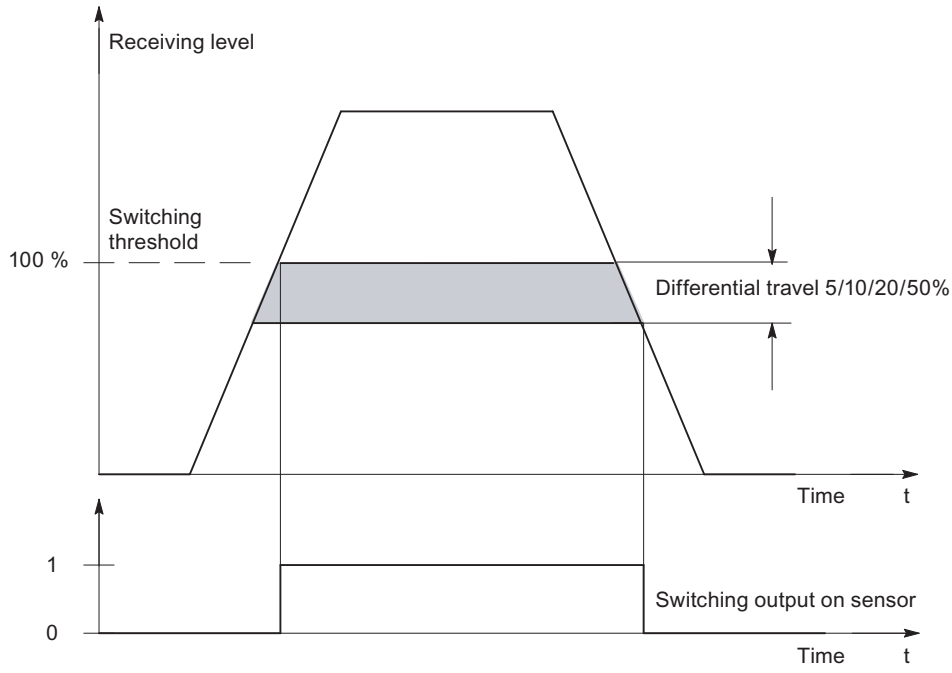

Figure 1-135 Switching hysteresis parameter

Time function,time value

These parameters can be used to set the electronic module for its specific application.

Operating principle

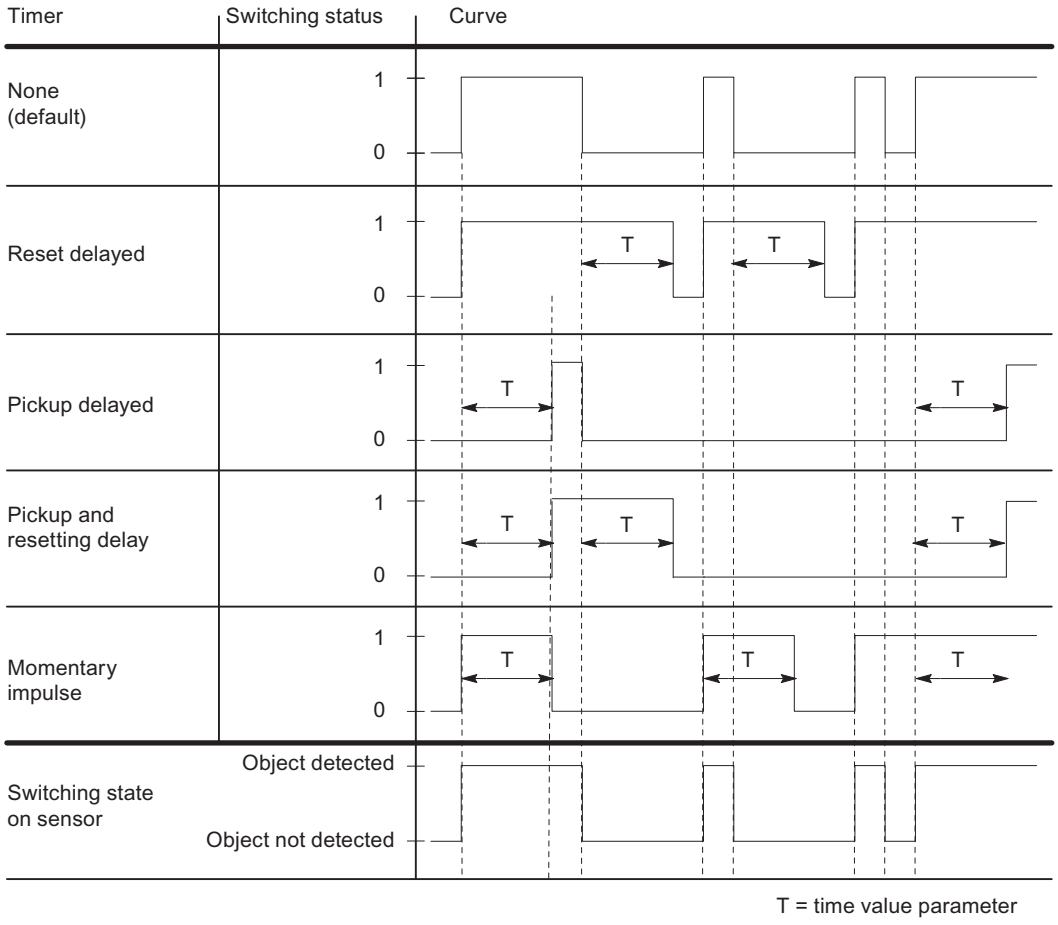

Figure 1-136 Time functions, time values parameters

Multiplex/synchronous mode

For the prevention of mutual influence between IQ Sense ultrasound devices in spatial proximity (devices with IQ profile ID 128), use the "Multiplex/synchronous mode" parameter.

Settings for the multiplex/synchronous mode parameter

Disabled: Mutual influence between IQ Sense ultrasound sensors in spatial proximity is possible (default). The cycle time is determined by the IQ Sense ultrasound sensor.

Multiplex: The IQ Sense ultrasound sensors determine the process value (distance) one after another, preventing them from affecting one another. The cycle time here is the sum of the configured synchronous cycle times of the IQ Sense ultrasound sensors that are to be multiplexed.

Synchronization: The IQ Sense ultrasound sensors determine the process value (distance) at exactly the same time, preventing them from affecting one another. The cycle time here corresponds to the greatest configured synchronous cycle time from among the IQ Sense ultrasound sensors that are to be synchronized.

You can, for example, use synchronous operation for a curtain function in which several IQ Sense ultrasound sensors aligned in parallel share a single extended detection area. The sensors simultaneously emit an ultrasound impulse. When an object enters the detection area, the sensor nearest to the object receives the echo most quickly. The object can therefore not only be detected, it can be located as well.

AFI value

Using the AFI value (application series identifier, as defined in the ISO 15693-3 international standard), transponders can be selected for different applications. Only transponders whose AFI value coincides with the value set on the sensor are processed. If a transponder has the AFI value "0", it can be identified and processed regardless of the AFI value of the sensor.

This parameter is only important if it is supported by the ident system, otherwise any value (normally "0") may be assigned.

Transponder type

Depending on the type of the transponder, you must configure whether it is an ISO transponder or a vendor-specific type.

For transponders in accordance with international standard ISO 15693, the value "1" should be selected; for all other types "0" is set. Based on this setting, one of the two possible air interface drivers is selected in the sensor.

This parameter is only important if it is supported by the ID system, otherwise any value (normally "0") may be assigned.

1.6.3.9 ET 200S

Configuring an ET 200S

Introduction

For the ET 200S series, you can find a wide range of modules in the hardware catalog under "Distributed I/O".

Assigning parameters

For information on configuration and parameter assignment, refer to "See also".

Frequency converters

Use of the frequency converter

Frequency converters

The frequency converters ICU24 and ICU24F (as fail-safe version) are modular frequency converters that are completely embedded in the distributed I/O system ET 200S. For parameterization of both modules, please see the following.

Message frame

The message frame number and the operating mode of the module are only displayed and cannot be modified.

Application ID

You indicate the saved parameters in the frequency converter as a whole with the application ID. Enter an application ID from the value range 0 to 65535. During startup (or pull/plug), this ID is compared with the application ID stored on the converter.

Converters that work with identical applications are usually also identically parameterized and should be identified with the same application ID. Converters with the same application ID may be exchanged between each other. Copying of the complete parameterization of a converter to another converter, for example, via an MMC, is only accepted, if both have the same application ID.

Converters that work with different applications and are parameterized differently must be identified by different application IDs. This prevents a converter with unsuitable parameterization from starting on an incorrect slot, i.e. on the wrong application. This also prevents the parameterization that is saved in the converter from being accidently overwritten with any parameteriation that is stored on an MMC.

Enable diagnostic interrupt

You can enable the diagnostic interrupt for the frequency converter. If diagnostic interrupt is enabled, an OB 82 must be available in a CPU to process the diagnostic events.

See also

Documentation for the frequency converter (http://support.automation.siemens.com/WW/view/en/26291825/0/en)

1.6.3.10 ET 200pro

ET 200pro with own CPU module

ET 200pro with own CPU

The ET 200pro can be equipped with its own CPU, for example with the CPU 1516pro.

In this regard, note the following rules.

Expansion rules with the CPU 1516pro:

- You can expand the CPU 1516pro with up to 16 modules.
- The total width can be a maximum of one meter.

Expansion with wide modules

If the maximum width of one meter is exceeded in a configuration with large modules (for example frequency converters or motor starters), the configuration cannot be compiled. In this case, reduce the configuration.

Configuration control with ET 200pro

Operating principle

Through the configuration control, it is possible to change the original configuration of an ET 200pro (created by configuring with STEP 7) and to operate the ET 200pro in this modified configuration. STEP 7 is no longer required for this configuration: You communicate to the ET 200pro the slot in which a configured module is actually inserted by means of your user program.

In this case, use control data record 196. In this data record, you code which modules are missing or located in different slots in the real configuration compared to the configuration. The configuration control has no effect on the parameter assignment of the modules (for example, the enabling of diagnostic alarms).

You then call the "WRREC" instruction and use it to write the data record to the interface module of the ET 200pro.

Configuration control gives you the flexibility to vary the configuration of an ET 200pro as long as the real configuration can be derived from a preset maximum configuration (originally created with STEP 7).

The following sections describe how to enable configuration control. They also outline how control data record 196 and readback data record 197 are structured.

Requirements

The CPU startup parameter "Compare preset to actual configuration" is set to Startup even if mismatch (default setting). This setting is also selected for the startup parameters of the individual modules of the ET 200pro.

Enabling configur

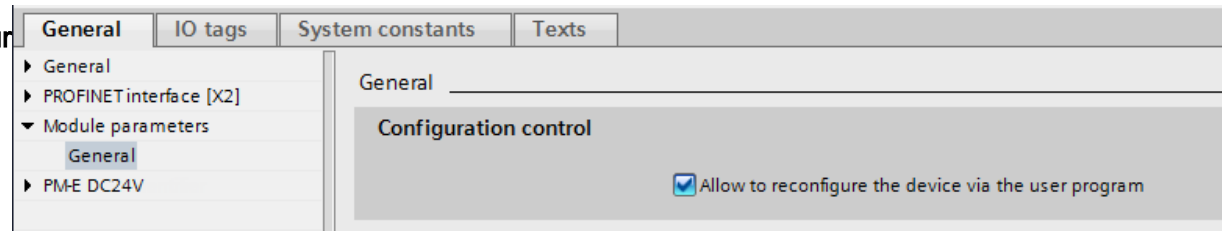

Configuration of control data record 196 for the ET 200pro

There is a byte in control data record 196 for each module.

The position of this byte in the data record codes one module in the original configuration with STEP 7:

- "slot_IM" (byte 4 in the data record, figure below) corresponds to the module in slot 1 in the configuration
- "slot_2" (byte 5) corresponds to the module in slot 2 in the configuration.
- "slot_3" (byte 6) corresponds to the module in slot 3 in the configuration.
- etc.

"slot_x" byte

The current slot is coded by the figure that is assigned to "slot_x" (by its value). Examples:

- The value "2" in byte 5 means you are assigning the module originally inserted in slot 2 to slot 2 in the current configuration (slot_2 = 2).
- The value "3" in byte 5 means you are assigning the module originally inserted in slot 2 to slot 3 in the current configuration (slot_2 = 3).
- The value "4" in byte 5 means you are assigning the module originally inserted in slot 2 to slot 4 in the current configuration (slot_2 = 4).
- etc.

There are no reserve modules for the ET 200pro (unlike for the ET 200S or with BU cover modules for the ET 200SP). For this reason, bit 7 of "slot_x" must not be set.

The value "0" in "slot_x" indicates that this module is not inserted in the current configuration.

Example of control data record 196

The figure below shows control data record 196 for a configuration of an ET 200pro with four modules.

This configuration is given as an example:

- The module originally configured with STEP 7 in slot 1 is also inserted in slot 1 in the current configuration.
- The module in slot 2 is in slot 2 in the current configuration.
- The module in slot 3 does not exist in the current configuration.
- The module in slot 4 is actually inserted in slot 3 in the current configuration.

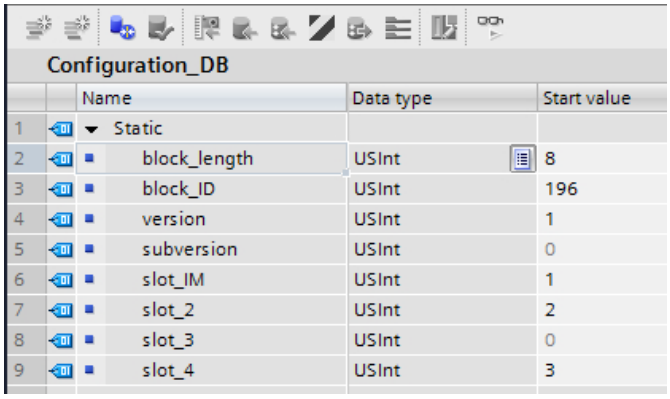

Configuration_DB

	Name	Data type	Start value
1	Static		
2	block_length	USInt	8
3	block_ID	USInt	196
4	version	USInt	1
5	subversion	USInt	0
6	slot_IM	USInt	1
7	slot_2	USInt	2
8	slot_3	USInt	0
9	slot_4	USInt	3

The components of control data record 196 (figure above):

- block_length: Note here the length of the control data record; in the example: 8 (bytes). The length is calculated using the following formula: "Number of assigned slots" + 4.
- block_ID: Enter the figure 196 here.
- version: The ET 200pro uses Version 1 of control data record 196.
- subversion: The ET 200pro uses Subversion 0 of control data record 196.
- slot_IM: The IM module is always inserted in slot 0 of the ET 200pro. Slot 1 always contains the virtual power module that is integrated as a fixed module in the IM module. The "slot_IM" (name can be changed) can contain any values. This byte is not interpreted in the configuration control of the ET 200pro.
- slot_2: The configured module 2 is inserted in slot 2 (value "2").
- slot_3: The configured module 3 is not present in the current configuration (value "0").
- slot_4: The configured module 4 is inserted in slot 3 in the current configuration (value "3").

Definition of control data record 196

A control data record 196 containing a slot assignment is defined for configuration control.

Byte	Element	Value	Description
0	Block length	e.g., 8 for ET 200pro with four modules	The length of the data record is calculated using the formula: 4 + Number of modules in bytes
1	Block ID	196	ID for control data record 196
2	Version	1	Version 1 of control data record 196
3	Subversion	0	Subversion 0 of control data record 196
4	Slot_1	Any values are possible: For example "1"	This byte is not interpreted for the ET 200pro because the power module that is integrated as a fixed module in the IM module of the ET 200pro is always inserted in slot 1.

5	Slot_2	Coding of the actual slot: 2 = Slot 2 3 = Slot 3 4 = Slot 4 etc. Coding for missing module: 0 = No slot, module not present	Byte 5 indicates where the module originally configured with STEP 7 in slot 2 is actually inserted in the current configuration. Example: 2 The module originally configured with STEP 7 in slot 2 is also actually located in slot 2 in the current configuration (value "2").
6	Slot_3	Coding of the actual slot: 2 = Slot 2 3 = Slot 3 4 = Slot 4 etc. Coding for missing module: 0 = No slot, module not present	Byte 6 indicates where the module originally configured with STEP 7 in slot 3 is actually inserted in the current configuration. Example: 0 The module originally configured with STEP 7 in slot 3 is not present in the current configuration (value "0").
7	Slot_4	Coding of the actual slot: 2 = Slot 2 3 = Slot 3 4 = Slot 4 etc. Coding for missing module: 0 = No slot, module not present	Byte 7 indicates where the module originally configured with STEP 7 in slot 4 is actually inserted in the current configuration. Example: 3 The module originally configured with STEP 7 in slot 4 is actually located in slot 3 in the current configuration (value "3").
:	:	:	:

Writing a data record

Transfer the control data record to the ET 200pro module.

To do so, call the extended WRREC (Write data record) instruction, and transfer the control data record created.

If you do not transfer a control data record, the interface module uses the original configuration with STEP 7. Here, the following applies: Configured module x is inserted in real slot x.

Addressing the interface module using the HW identifier

To transfer data record 196 with the instruction WRREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Error messages

The following error messages are returned if an error occurs when writing control data record 196:

Table 1-247 Error messages

Error code	Meaning
16#80A2	DP protocol error on layer 2 Indicates that a data record has not been acknowledged due to the system.
16#80B1	Invalid length; the length information in data record 196 is not correct.
16#80B5	Configuration control parameters not assigned.
16#80B2	Invalid slot: The configured slot is not assigned.
16#80B8	Parameter error; module signals invalid parameters.
16#80C5	DP slave or module not available. Indicates that a data record has not been acknowledged due to the system.

Readback data record 197 for ET 200pro

Readback data record 197 is used to read the actual configuration of a station (in this case of an ET 200pro).

This data record allows you to check the real configuration of the ET 200pro (actual configuration). The feedback data record for each configured module specifies whether or not it is actually present.

- The value "1" means that the correct module is inserted in the correct slot.
- The value "0" codes all other options (wrong module, empty slot).

Example:

A module has been configured with STEP 7 for slot 4.

This module has then been moved to slot 3 in the current configuration using data record 196.

If this module is also really in slot 3, this is coded by the value "1" (status_slot_4 = 1).

Configuration details:

The configuration of the data block corresponds to the original configuration of the ET 200pro with STEP 7.

There is a byte in the data record for each module. The position of this bytes in the data record corresponds to the position of a module in the original configuration with STEP 7.

Sequence of bytes:

- "status_slot_IM" (byte 4 in the data record) corresponds to the module in slot 1 in the configuration,
- "status_slot_2" (byte 5) corresponds to the module in slot 2 in the configuration
- "status_slot_3" (byte 6) corresponds to the module in slot 3 in the configuration,
- etc.

The following example is for a configuration with 4 modules. The value "8" therefore appears in the "block_length" element of the data record.

If you configure an ET 200pro in STEP 7 with fewer modules, the data block will be shorter.

You can choose any name for the components of the control data record (for example "status_slot_2").

Meaning of "status_slot_x":

- The value "1" in status_slot_x means that module x is inserted in the correct slot
- The value "0" in status_slot_x codes all other options (wrong module, module does not exist).

Example:

The figure below shows readback data record 197 for an ET 200pro with four modules.

- There is no module 3 (this was assigned in control data record 196, see section "Example of control data record 196" above)
- The three other modules are actually inserted in the ET 200pro.

Record_197_ET_200pro

	Name	Datentyp	Defaultwert
1	block_length	USInt	8
2	block_ID	USInt	197
3	version	USInt	1
4	subversion	USInt	0
5	status_slot_IM	USInt	1
6	status_slot_2	USInt	1
7	status_slot_3	USInt	0
8	status_slot_4	USInt	1

Reading readback data record 197

You can read readback data record 197 from the ET 200pro with the instruction RDREC. RDREC operates asynchronously. If you call RDREC in the startup OB, you must call the instruction multiple times using a loop until the "BUSY" or "DONE" output parameter indicates that the data record has been read.

To read data record 197 with the instruction RDREC, you must enter the HW identifier of the IM submodule with the extension "~Head" as the input parameter for the instruction. The system constant of this HW identifier is, for example, "IO-Device_2~Head". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the corresponding value for addressing.

Additional information and examples

Further information on the PN interface module of the ET 200pro is available here (http://support.automation.siemens.com/WW/view/en/98099372).

Specific examples of configuration control can be found here in this application description (http://support.automation.siemens.com/WW/view/en/29430270).

You will find a library of the data records and other application examples here:

Templates for the data records (http://support.automation.siemens.com/WW/view/en/29430270)

See also

Configuration control with ET 200SP (Page 3215)

Use of the frequency converter

Frequency converters

The frequency converters ET 200pro FC and ET 200pro F-FC (as fail-safe version) are modularly design frequency converters that are completely embedded in the distributed I/O system ET 200pro. The following section describes how to configure the two modules.

Message frame

The message frame number and the operating mode of the module are only displayed and cannot be modified.

Application ID

You indicate the saved parameters in the frequency converter as a whole with the application ID. Enter an application ID from the value range 0 to 65535. During startup (or pull/plug), this ID is compared with the application ID stored on the converter.

Converters that work with identical applications are usually also identically configured and should be identified with the same application ID. Converters with the same application ID may be exchanged between each other. Copying of the complete configuration of a converter to another converter, for example, via an MMC, is only applied, if both have the same application ID.

Converters that work with different applications and are configured differently must be identified by different application IDs. This prevents a converter with unsuitable configuration from starting on an incorrect slot, in other words on the wrong application. This also prevents the configuration that is saved in the converter from being accidently overwritten with any configuration that is stored on an MMC.

Enable diagnostic interrupt

You can enable the diagnostic interrupt for the frequency converter. If diagnostic interrupt is enabled, an OB 82 must be available in a CPU to process the diagnostic events.

1.6.4 IPv4 routing with CP 1543-1 / CM 1542-1

IPv4 routing with CP 1543-1 / CM 1542-1

The following restrictions apply to the "IPv4 routing over the backplane bus" function of the CM/CP.

- IPv4 routing over the CM / CP to the CPU of the station is not possible.
- IP configuration with CM 1542-1 as PROFINET IO controller
  If the CM with enabled IPv4 routing over the backplane bus is used as PROFINET IO controller, please note the following points for IP configuration of the subordinate IO devices:
  If IP address configuration for the subordinate IO devices is fixed ("Set IP address in the project" option), IPv4 routing over the CM to the IO devices is not possible.
  The following settings are required in the IP configuration of the subordinate IO devices to allow IPv4 routing over the CM to those IO devices:
  - First select the "Use router" option for the IO device.
    Copy the IP address of the IO controller (CM) to the router address field.
  - Activate the "IP address is set directly at the device" option for the IO device.

1.6.5 IPv6 configuration

1.6.5.1 IPv6 protocol

The Internet protocol version 6 (IPv6) extends the address range of the Internet protocol version 4 (IPv4).

Address format IPv6: Notation

IPv6 addresses consist of 8 fields each with four-character hexadecimal numbers (128 bits in total). The fields are separated by a colon.

Example: fd00:0000:0000:0000:0db8:002f:0000:0000

Rules / simplifications:

- Leading zeros within a field can be omitted.
  Example: fd00:0000:0000:0000:0db8:002f:0000:0000, abbreviated fd00:0:0:0:db8:2f:0:0
- If one or more blocks have the value 0, a shortened notation is possible with two successive colons. To ensure uniqueness, this shortened form can only be used once within the entire address. With two or more such abbreviations, it may not be possible to determine how many blocks were shortened.
  Valid example: fd00::db8:2f:0:0 or fd00:0:0:0:db8:2f:: (both clearly recognizable as fd00:0:0:0:db8:2f:0:0)
  Invalid example: fd00::db8:2f:: (not clear, can be interpreted as fd00:0:0:0:db8:2f:0:0 or fd00:0:0:db8:2f:0:0:0 etc.)
- The last 2 fields or 4 bytes can be written in the normal decimal notation with periods.
  Example: The IPv6 address fd00::db8:fd01:2f is equivalent to fd00::db8.253.1.0.47

The entry of IPv6 addresses is possible in the notations described above. IPv6 addresses are always displayed in the notation of the entry.

Different IPv6 addresses of an interface

If you configure multiple IPv6 addresses for the interface of a communication module, please note that the two IPv6 addresses cover different subnet ranges.

See also

IPv6 with the CP 1543-1 (Page 3302)

1.6.5.2 IPv6 with the CP 1543-1

Use of IPv6 with the CP 1543-1

For all IP services, the CP supports the Internet protocol version 4 (IPv4)

The additional specification of addresses in IPv6 format can be used with the CP for the following services and applications:

- FETCH/WRITE: Direct write/read access by PC stations, SIMATIC S5 or third-party devices
- FTP client: FTP access from the S7-1500 CPU to an FTP server with program block FTP_CMD
- FTP server: FTP access from an FTP client to data areas of the S7-1500 CPU
- SNMP: Data query using MIB objects according to SNMP
- E-mail: Data transfer from the S7-1500 CPU using the program block T_Mail

Device and network diagnostics 2

2.1 Hardware diagnostics

2.1.1 Overview of hardware diagnostics

2.1.1.1 Principal methods of hardware diagnostics

Principal methods of hardware diagnostics

Hardware diagnostics can be performed as follows:

- Using the Online and Diagnostics view
- Using the "Online Tools" task card
- Using the "Diagnostics > Device Info" area of the Inspector window
- Using diagnostics icons, for example, in the device view and the project tree

Structure of the Online and Diagnostics view

The Online and Diagnostics view consists of two windows alongside each other:

- The left window shows a tree structure with folders and - when you open the folder - groups.
- The right window contains detailed information on the selected folder or selected group.

The "Online access" group and the "Diagnostics" and "Functions" folders are located here:

- "Online access" group: Displays whether or not there is currently an online connection with the associated target. In addition, you can establish or disconnect the online connection.
- "Diagnostics": Contains several diagnostics groups for the selected module.
- "Functions": Contains several groups, in which you can make settings for the selected module or issue commands to the module.

Function and structure of the "Online Tools" task card

For modules with their own operating mode (such as CPUs), the "Online tools" task card allows you to read current diagnostics information and commands to the module.

If you selected a module without its own operating mode or if you selected several modules before activation of the "Online Tools" task card, the task card relates to the relevant CPU.

The "Online Tools" task card consists of the following panes:

- CPU control panel
- Cycle time
- Memory

Note

A pane is filled with content only if the module controls the associated functions and an online connection exists.

If there is no online connection to the respective module, the display "No online connection" appears in blue in each pane. If an existing online connection was disconnected, then "This target is not available" will be displayed.

Structure or the "Diagnostics" tab of the Inspector window

The "Diagnostics" tab of the Inspector window itself consists of several tabs. Of these tabs, the following is relevant for the hardware diagnostics.

- Device information
  This tab relates to all online devices (e.g. CPUs) to which an online connection has been established and to the devices that are assigned to these online devices (such as PROFINET devices and PROFIBUS slaves). Alarms related to the faulty online devices and the faulty devices are output here.

Note

What is displayed when a module is faulty?

If a module within a device is faulty, only the associated device or its proxy (e.g. head module) are shown, but not the module itself.

The faulty devices are displayed "at the top level" and not in a hierarchical view below their online device (as is the case in the project tree).

2.1.1.2 Determination of which of the devices that are connected online are defective

Overview of the defective devices

In the "Diagnostics > Device Info" area of the Inspector window you will obtain an overview of the defective devices that are or were connected online.

The "Diagnostics> Device Info" area of the Inspector window consists of the following elements:

- Header line with the number of defective devices
- Table with detailed information on each defective device

If you originate the establishment of an online connection to a device which is not reachable or reports one or more faults or is not in RUN mode, it will rank as defective.

Structure of the table with detailed information on the defective devices

The table consists of the following columns:

- Online status: Contains the online status as a diagnostic symbol and in words
- Operating mode: Contains the operating mode as a symbol and in words
- Device / module: Name of the affected device or the affected module
- Message: explains the entry of the previous column
- Details: The link opens the online and diagnostics view for the device, and places it in the foreground. If an online connection does not exist any longer, the link will open the connection establishment dialog.
- Help: The link supplies further information on the defect that has occurred.

See also

Displaying diagnostics status and comparison status using icons (Page 3305)

2.1.1.3 Displaying diagnostics status and comparison status using icons

Determining diagnostics status online and displaying using icons

When the online connection to a device is established, the diagnostics status of the device and, if applicable, its lower-level components is determined. The operating state of the device is also determined, where applicable.

The following is a description of which icons are displayed in specific views.

- Device view
  - The associated diagnostic icon is displayed for every hardware component (except the signal board on the CPU).
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - For hardware components with their own operating state, the operating state icon is also displayed to the left of or above the diagnostic icon.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: No diagnostic icons are displayed (due to the configuration as GSDML device).

Note

Diagnostics status of SCALANCE-XM-400 and SCALANCE-XR-500-managed modules

Incorrect or no diagnostic icons are displayed in case of an existing online connection to the SCALANCE-XM-400 and SCALANCE-XR-500-managed modules.

The correct diagnostic icons will be displayed when you establish the online connection to an associated CPU.

- Device overview
  - The associated diagnostic icon is displayed for every hardware component.
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: For modules assigned to the CPU, the associated diagnostic icon is displayed (modules that are not assigned do not receive a diagnostic icon). The associated diagnostic icon is displayed for plug-in submodules of an assigned module (submodules that are not pluggable are not visible and therefore do not receive a diagnostic icon).
- Network view
  - The associated diagnostic icon is displayed for every device.
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.

- Network overview
  - The associated diagnostic icon is displayed for every hardware component.
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.
- Topology view
  - The associated diagnostic icon is displayed for every device.
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The associated diagnostic icon is displayed for every port. The meaning of the individual colors is described further below.
  - Each cable between two online ports is assigned the color associated with its diagnostics status.
    The color of the cable between two ports depends on the status of the individual ports:

Color of the first port	Color of the second port	Color of the connecting cable
light green	light green	light green
light green	dark green	dark green
green	gray	gray
green	red	red
gray	red	red

  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.
- Topology overview
  - The associated diagnostic icon is displayed for every hardware component.
    To start the Online and Diagnostics view (if available), double-click the diagnostic icon.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.

- Project tree
  - The associated diagnostic icon is displayed behind every hardware component.
  - For a hardware component with lower-level components (e.g., distributed I/O, Slave_1), if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - For hardware components with their own operating state, the operating state icon is also displayed in the top right corner of the diagnostic icon.
  - If forcing is active on a CPU, a red F is displayed at the left margin of the diagnostic icon.
  - The diagnostic icon "Hardware error in lower-level component" is displayed behind the "Local modules" folder when there is a hardware error in at least one of the associated modules.
  - The diagnostic icon "Hardware error in lower-level component" is displayed behind the "Distributed I/O" folder when there is a hardware error in at least one of the associated modules.
  - The diagnostic icon "Hardware error in lower-level component" is displayed behind the project folder when the "Hardware error in lower-level component" diagnostic icon is displayed behind at least one of the "Local modules" or "Distributed I/O" folders.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: The associated diagnostic icon is displayed for modules assigned to the CPU (modules that are not assigned are grayed out and do not receive a diagnostic icon). The associated diagnostic icon is displayed for plug-in submodules of an assigned module (submodules that are not pluggable are not visible and therefore do not receive a diagnostic icon).

Note

If the diagnostic for a hardware component is "not reachable from the CPU", the diagnostic icon "Hardware error in lower-level component" is not additionally shown.

Diagnostic icons for modules and devices

The following table shows the available icons and their meaning.

Icon	Meaning
	The connection with a CPU is currently being established.
	The CPU is not reachable at the set address.
	The configured CPU and the CPU actually present are of incompatible types.
	On establishment of the online connection to a protected CPU, the password dialog was terminated without specification of the correct password.
	No fault

Icon	Meaning
	Maintenance required
	Maintenance demanded
	Error
	The module or device is deactivated.
	The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
	The functionality of the module or submodule is no longer available (for example. input and output data). Possible causes: • Difference between expected and actual configuration. • Access error during updating the process images.
	Diagnostics not or only partially possible. Possible causes: • Difference between actual online and offline configuration data. • You have not executed the command "Compile" for the hardware. • You have not executed the command "Download to device" for the hardware or the hardware configuration. • The object does not support diagnostics.
	The connection is established, but the module status has not yet been determined or is unknown.
	The configured module does not support display of the diagnostics status.
	Hardware error in lower-level component: A hardware fault has occurred in at least one lower-level hardware component (occurs as a separate icon only in the project tree)

Note

Some modules, for example, the FM 450-1, are only indicated as having a problem in the case of an error if you have enabled the diagnostic interrupt when assigning the module property parameters.

Icons for the comparison status

The diagnostic icons can be combined at the bottom right with additional smaller icons that indicate the result of the online/offline comparison. The following table shows the available comparison icons and their meaning.

Icon	Meaning
	Hardware error in lower-level component: The online and offline versions differ (only in the project tree) in at least one lower-level hardware component.
	Software error in lower-level component: The online and offline versions differ (only in the project tree) in at least one lower-level software component.
	Online and offline versions of the object are different

Icon	Meaning
	Object only exists online
	Object only exists offline
	Online and offline versions of the object are the same

Note

If both a comparison icon and the "Error in lower-level component" diagnostic icon are to be displayed at the bottom right in the device view, the following rule applies: The diagnostic icon for the lower-level hardware component has a higher priority than the comparison icon. This means that a comparison icon is only displayed if the lower-level hardware components have no errors.

Display of software errors in the project tree

- The associated comparison icon is shown behind each block.
- Behind each folder, under which exclusively blocks are contained, the diagnostic icon "Software error in lower-level component" is displayed when there is a software error in at least one of the associated blocks.
- For a hardware component with lower-level software components, if there is no hardware error and there is an error in at least one lower-level software component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Software error in lower-level component" is also shown in the lower right corner.

Combined diagnostics and comparison icons

The following table shows examples of icons that are displayed in the diagnostics icon.

Icon	Meaning
	Folder contains objects whose online and offline versions differ (only in the project tree)
	Object only exists online

Operating state icons for CPUs and CPs

The following table shows the available icons and their respective operating states.

Icon	Operating state
	RUN
	STOP
	STARTUP

Icon	Operating state
	HOLD
	DEFECTIVE
	Unknown operating state
	The configured module does not support display of the operating state.

Note

If forcing is active on a CPU, a red F is displayed on a pink background at the bottom right of the operating state icon.

Color marking of ports and Ethernet cables

The following table shows the available colors and their respective meaning.

Color	Meaning
	No fault or maintenance required
	Offline
	Maintenance demanded
	Communication error or topological error
	no diagnostic capability

2.1.1.4 Start online and diagnostics view

Overview of possible ways of starting the Online and Diagnostics view

You can start the Online and Diagnostics view of a module to be diagnosed at the following locations:

- Overview
- Project tree
- Device view
- Device overview
- Network view
- Network overview
- Topology view
- Topological overview

In the following, examples are used to show how to proceed.

Requirement

The project with the module to be diagnosed is open.

Note

This requirement does not apply if you call the online and diagnostics view from the project tree after you have identified the accessible devices.

Procedure

To start the online and diagnostics view of a module, follow these steps:

1. In the project tree, open the respective device folder.
2. Double click on "Online & Diagnostics".

Or:

1. In the project tree, select the respective device folder.
2. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. In the project tree, open the "Online access" folder.
2. Open the folder for the interface with which you want to establish the online connection.
3. Double click on "Show/Update accessible devices".
4. Select the module to be diagnosed.
5. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. In the project tree, open the "Local modules" folder.
2. Select the respective device or the module that is to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the main menu.

Or:

1. Open the device view in the device configuration.
2. Select the module to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. Open the device view in the device configuration.
2. Establish an online connection to the module to be diagnosed.
3. Double-click on the diagnostics icon above the module.

Or:

1. Open the network view in the device configuration.
2. Select the station with the module to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. Open the topology view in the device configuration.
2. Establish an online connection to the module to be diagnosed.
3. In the topology view, double-click the diagnostic icon associated with the module.

Result

The online and diagnostics view of the module to be diagnosed will be started. If an online connection to the associated CPU had previously been created, the header bar of the Online and Diagnostics view will now have an orange background.

Note

If no online connection exists when the online and diagnostics view is started, no online information is displayed and the display fields remain empty.

2.1.1.5 Activation of the "Online Tools" task card

Activation of the "Online Tools" task card

You can activate this task card as follows:

1. Start the online and diagnostics view.
2. Click on the "Online Tools" task card.

Or:

1. Start the device view.
2. Click on the "Online Tools" task card.

Or:

1. Start the network view.
2. Click on the "Online Tools" task card.

2.1.2 Showing non-editable and current values of configurable module properties

2.1.2.1 Showing general properties and system-relevant information for a module

Where do I find the information I need?

The general properties and system-relevant information for a module can be found in the "General" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed.

Structure of the "General" group

The "General" group consists of the following areas:

- Module
- Module information
- Vendor information

"Module" area

This area shows the following data of the module:

- Short designation, for example, CPU 1214C DC/DC/DC
- Article no.
- Hardware
- Firmware
- TIA Portal version

  Note
  TIA Portal version

  The displayed TIA Portal version is the minimum version that is required to open the blocks loaded to the device or to load the blocks to the ES. This means it does not necessarily have to be the version with which you have loaded the blocks to the device.

  Fictitious example: You have installed the TIA Portal in version V14.0 SP1 on your ES and are creating a new project. This project is assigned project version V14.0. You download this project (along with its project version) to the CPU. The project version, which means V14.0, is displayed.

- Racks
- Slot

"Module information" area

This area shows the following data of the module that you configured during hardware configuration:

- Module name
- Installation date (not displayed for all modules)
- Additional information (not displayed for all modules)

"Manufacturer information" area

This area shows the following data of the module:

- Manufacturer
- Serial number
- Profile: Profile ID as hexadecimal number

  Note

  You will find the corresponding profile name in the profile ID table for PROFIBUS International (see "www.profibus.com").

- Profile details: Profile-specific type as hexadecimal number

  Note

  You will find the corresponding profile-specific type name in the profile-specific type table for PROFIBUS International (see "www.profibus.com").

2.1.2.2 Display configured cycle times

Where do I find the information I need?

The required information can be found in the following places:

- In the "Cycle time" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.
- In the "Cycle time" pane of the "Online Tools" task card

Structure of the "Cycle time" group in the "Diagnostics" folder of the Online and Diagnostics view

The "Cycle time" group consists of the following areas:

- Cycle time diagram (graphical display of the assigned and measured cycle times)
- Cycle time configured (display of the assigned cycle times as absolute values)
- Cycle times measured (display of the measured cycle times as absolute values)

Structure of the "Cycle time" pane of the "Online Tools" task card

The "Cycle time" pane displays the cycle time diagram and below it the measured cycle times as absolute values.

Assigned cycle times

The following assigned cycle times are displayed in the cycle time diagram and in the "Cycle time configured" area.

- Minimum cycle time
- Maximum cycle time

In the cycle time diagram, the minimum cycle time and the maximum cycle time correspond to the two markings on the time axis.

In the "Cycle time configured" area, the assigned cycle times are displayed as absolute values.

2.1.2.3 Show interfaces and interface properties of a module

Where do I find the information I need?

The interfaces and interface properties of a module can be found in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed in the following group:

- PROFINET interface

"PROFINET interface" group

This group is divided into the following areas:

- "Ethernet address" with the "Network connection" and "IP Parameters" subareas
- "Ports"

"Network connection" subarea of the "Ethernet address" area

This subarea shows the following data for the module:

- MAC Address:
  MAC address of the interface.
  The MAC address consists of two parts. The first part ("Basic MAC address") identifies the manufacturer (Siemens, 3COM, ...). The second part of the MAC address differentiates between the various Ethernet devices. Each Ethernet module is assigned a unique MAC address.

"IP Parameters" subarea of the "Ethernet address" area

This subarea shows the following data for the module:

- IP address:
  Internet protocol address of the device on the bus (TCP/IP)
- Subnet mask:
  The subnet mask shows which part of the IP address determines the membership of a particular sub-network.
- Default router:
  If the subnet is connected via a router to other subnets, the IP address of the default router must be known. This is the only way a datagram can be forwarded with a non-matching subnet address.
- IP settings:
  Identifier for the path by which the device has obtained its IP settings (IP address, subnet mask, default router).

Identifier	Meaning
0	IP address is not initialized
1	By configuration (i.e., by the configuration loaded to the device from the device or network view)
2	Via the "Assign IP address" group of the online and diagnostics view
3	Via the DHCP server (i.e., the IP parameters are obtained by a special service from a DHCP server (Dynamic Host Configuration Protocol) and assigned for a limited time)
4	IP address is set by a user program
5	Source of IP address unknown

- IP setting time:
  Time stamp of the last change to the IP address directly through the Ethernet connection of the module

"Ports" area

This area shows the following data for the module:

- Ethernet ports
  Physical properties of the PROFINET interface

Properties of the PROFINET interface	Meaning
Port no.	Port number The short description of interface (X + interface no.) and port (P + port no.) is specified in parentheses. An "R" in the short description of a port means that it is a ring port.
Status	Displays the status of the port LINK LED. • Status "OK" means another device (such as a switch) is connected to the port and the physical connection is available. • Status "disconnected" means no other device is connected to the port. • Status "deactivated" means that access to the port is blocked.
Settings	• "Automatic" for automatic network settings of the device • Network settings for speed and transmission method for manual network settings of the device
Operating mode	Network settings for speed and transmission method

If you select a line in the port table, additional help information will be provided for the corresponding port.

2.1.2.4 Displaying IO controllers that access modules of a Shared Device (S7-1500)

Where do I find the information I need?

The display of those IO controllers that access the modules of a Shared Device can be found in the Online and Diagnostics view of the interface module of the Shared Device in the "Diagnostics" folder in the following area of the "PROFINET interface" group:

- IO controller

See also

Useful information on Shared Devices (Page 2961)

2.1.2.5 Displaying sync domain properties of a PROFINET device

Where do I find the information I need?

The sync domain properties of a PROFINET device can be found in the following area of the "PROFINET interface" group in the "Diagnostics" folder of the Online and Diagnostics view of the device to be diagnosed:

- Domain

"Domain" area

This area is divided into the following subareas:

- Sync domain
- MRP domain

What is a sync domain?

A sync domain is a group of PROFINET devices that are synchronized to a common clock. Exactly one device has the role of the sync master (clock generator); all other devices assume the role of a sync slave. The sync master is usually an IO controller or a switch.

Non-synchronized PROFINET devices are not part of a sync domain.

"Sync domain" subarea of the "Domain" area

This subarea shows the following properties of the sync domain:

- Name:
  Name of sync domain
- Role:
  Role of the PROFINET device in the sync domain. The following roles are possible:
  - Sync master
  - Sync slave
- Synchronization interval:
  Interval at which the synchronization is performed
- Send clock
  Smallest possible send interval for the data exchange
- Jitter accuracy of the send clock
- Reserved bandwidth for cyclic communication

2.1.2.6 Displaying MRP domain properties of a PROFINET device

Where do I find the information I need?

The MRP domain properties of a PROFINET device can be found in the following area of the "PROFINET interface" group in the "Diagnostics" folder of the Online and Diagnostics view of the device to be diagnosed:

- Domain

"Domain" area

This area is divided into the following subareas:

- Sync domain
- MRP domain

What is an MRP domain?

The Media Redundancy Protocol (MRP) enables redundant networks to be structured. Redundant transmission paths (ring topology) ensure that, if one transmission path fails, an alternative communication path is available. The PROFINET devices that are part of this redundant network form an MRP domain.

"MRP domain" subarea of the "Domain" area

This subarea shows the following properties of the MRP domain:

- Name:
  Name of MRP domain
- Role:
  Role of the PROFINET device in the MRP domain. The following roles are possible:
  - Manager
  - Manager (Auto)
  - Client
  - Not a device of the ring
- Ring port 1:
  The port of the PROFINET device that has the "Ring port 1" property
- Ring port 2:
  The port of the PROFINET device that has the "Ring port 2" property
- Status of the MRP ring:
  Indicates whether the ring is interrupted ("open" status) or not ("closed" status).

2.1.2.7 Displaying current firmware of a module (S7-300, S7-400, S7-1500)

Displaying firmware

You can display the currently installed firmware of a module.

Requirements

- The module supports a firmware update.
- The module is connected online.

Procedure

To display the current firmware, follow these steps:

1. Open the module in the Online and Diagnostics view.
2. Select the "Firmware update" group in the "Functions" folder.
3. You read off the current firmware in the "Online data" area under "Firmware".

2.1.3 Showing the current values of dynamic modules properties

2.1.3.1 Display measured cycle times

Where do I find the information I need?

The measured cycle times can be found at each of the following places:

- In the "Cycle time" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.
- In the "Cycle time" pane of the "Online Tools" task card

Structure of the "Cycle time" group in the "Diagnostics" folder of the Online and Diagnostics view

The "Cycle time" group consists of the following areas:

- Cycle time diagram (graphical display of the assigned and measured cycle times)
- Cycle time configured (display of the assigned cycle times as absolute values)
- Cycle times measured (display of the measured cycle times as absolute values)

Structure of the "Cycle time" pane of the "Online Tools" task card

The "Cycle time" pane displays the cycle time diagram and below it the measured cycle times as absolute values.

Graphical display of the measured cycle times

The following measured cycle times are displayed in the cycle time diagram:

- Shortest cycle time: Duration of the shortest cycle since the last transition from STOP to RUN
  This corresponds to the dashed gray arrow on the left in the diagram.
- Current / last cycle time: Duration of the last cycle
  This corresponds to the green arrow in the diagram. If the current / last cycle time exceeds the maximum cycle time, the arrow will turn red.

  Note

  If the duration of the last cycle comes close to the maximum cycle time, it may be possible that it will be exceeded. Depending on the CPU type, parameter assignment and your user program, the CPU can switch to STOP mode. If for instance you are monitoring the tags in your program, this will increase the cycle time.

  If the cycle lasts longer than double the maximum cycle time, and you do not restart the maximum cycle time in the user program (by calling the extended RE_TRIGR) instruction, the CPU will switch to STOP mode.

- Longest cycle time: Duration of the longest cycle since the last transition from STOP to RUN.
  This corresponds to the dashed blue arrow on the right in the diagram.

A blue band extends between the two dashed lines; this band corresponds to the entire range of the measured cycle times. If a measured cycle time is greater than the maximum cycle time, the portion of the band that lies outside the assigned limits will be colored red.

Display of the measured cycle times as absolute values

The following measured times are displayed in the "Cycle times measured" area and in the "Cycle time" pane.

- Shortest cycle time since the last transition from STOP to RUN.
- Current/last cycle time:
- Longest cycle time since the last transition from STOP to RUN.

2.1.3.2 Showing the current status of the LEDs of a CPU

Where do I find the information I need?

The current status of the LEDs of a CPU can be found in the display area of the "CPU control panel" pane of the "Online tools" task card.

Display area of the "CPU control panel" pane of the "Online Tools" task card

This area contains the following displays:

- Station name and CPU type (short designation)
- RUN / STOP (corresponds to the "RUN / STOP" LED of the CPU)
- ERROR (corresponds to the "ERROR" LED on the CPU)
- MAINT (corresponds to the "MAINT" LED on the CPU)

2.1.3.3 Showing fill levels of all types of memory on a CPU

Where do I find the information I need?

The fill levels of all types of memory on a CPU can be found on the following two pages:

- In the display area of the "Memory" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed
- In the display area of the "Memory" pane on the "Online Tools" task card

Display area of the "Memory" group in the "Diagnostics" folder of the online and diagnostics view

This area contains the current memory utilization of the associated module and details of the individual memory areas.

The memory utilization is shown both as a bar diagram and as a numerical value (percentage).

The following memory utilizations are shown:

- Load memory
  If no memory card is inserted, the internal load memory is displayed.
  If a memory card is inserted, the operating system only uses the inserted load memory as the load memory. This is displayed here.
- Work memory
- Retentive memory

Display area of the "Memory" pane of the "Online Tools" task card

This area contains the current memory utilization of the associated module. The available memory is shown both as a bar diagram and as a numerical value (percentage). The numerical value is rounded to an integer value.

Note

If less than 1% of a memory area is utilized, the available portion of this memory area is shown as "99%".

The following memory utilizations are shown:

- Load memory
  If no memory card is inserted, the internal load memory is displayed.
  If a memory card is inserted, the operating system only uses the inserted load memory as the load memory. This is displayed here.
- Work memory
- Retentive memory

See also

Load memory (Page 1298)

Work memory (Page 1299)

Retentive memory areas (Page 1300)

2.1.3.4 Displaying fill level of all types of memory of an S7-1500 CPU (S7-1500)

Where do I find the information I need?

The fill levels of all types of memory of an S7-1500 CPU can be found at the following two places:

- In the display area of the "Memory" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed
- In the display area of the "Memory" pane on the "Online Tools" task card

Display area of the "Memory" group in the "Diagnostics" folder of the online and diagnostics view

This area contains the current memory utilization of the associated module and details of the individual memory areas.

The memory utilization is shown both as a bar diagram and as a numerical value (percentage).

The following memory utilizations are shown:

- Load memory

  Note

  The load memory is located on the SIMATIC memory card.

- Code work memory: work memory for program code
- Data work memory: work memory for data blocks
- Retentive memory

Display area of the "Memory" pane of the "Online Tools" task card

This area contains the current memory utilization of the associated module. The available memory is shown both as a bar diagram and as a numerical value (percentage). The numerical value is rounded to an integer value.

Note

If less than 1% of a memory area is utilized, the available portion of this memory area is shown as "99%".

The following memory utilizations are shown:

- Load memory

  Note

  The load memory is located on the SIMATIC memory card.

- Code work memory: work memory for program code
- Data work memory: work memory for data blocks
- Retentive memory

2.1.4 Checking a module for defects

2.1.4.1 Determining the diagnostic status of a module

Where is the diagnostics status of a module displayed?

The diagnostic status of a module is displayed in the "Diagnostic status" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed.

The "Diagnostics status" group consists of the following areas:

- Status
- Standard diagnostics (for S7-300 and S7-400 only for non-CPU modules)

"Status" area

The following status information is displayed in this area:

- Status of the module as viewed by the CPU, for example:
  - Module available and OK.
  - Module defective.
    If the module experiences a fault and you have enabled the diagnostic error interrupt during configuration, the "Module defective" status is displayed.
  - Module configured, but not available.
    Example: Diagnostics data is not available because the current online configuration differs from the offline configuration.
- Detected differences between the configured and the inserted module. Provided it can be ascertained, the article number will be displayed for the set and actual type.

The scope of the displayed information depends on the selected module.

"Standard diagnostics" area

The following diagnostics information for non-CPU modules is displayed in this area:

- Internal and external faults that relate to the overall module
- Associated diagnostics events

Examples of such diagnostics information are:

- Entire backup failed
- Module defective

Note

Diagnostic interrupts

A diagnostic interrupt can be reported to the CPU only if the module has diagnostic interrupt capability and the diagnostic interrupt has been enabled.

The display of the diagnostic interrupt is a snapshot. Sporadic module defects can be identified in the diagnostics buffer of the respective CPU.

2.1.4.2 Performing communication diagnostics of PROFINET ports

Where are port-specific errors of PROFINET ports displayed?

The port-specific errors of PROFINET ports are displayed in the "Communication diagnostics" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.

Structure of the "Communication diagnostics" group

The "Communication diagnostics" group consists of the following areas:

- Table containing the port-specific errors
- "Details of the event" text field
- "Help on event" text field

Structure of the table containing the port-specific errors

Each line of the error table corresponds to one error.

The table consists of the following columns:

- Name: port no. and port designation, as well as diagnostics icon
- Error: description of the error that occurred on this port

"Details" text field

If you select a line in the error table, the "Detail" text field will contain detailed information (if available) on the corresponding error.

"Help on selected diagnostic line" text field

If you select a line in the error table, the "Help on selected diagnostic line" text field will contain help information (if available) on the corresponding error.

2.1.4.3 PROFINET IO component diagnostics

Where do I perform PROFINET IO component diagnostics?

You perform the PROFINET IO component diagnostics in the "PROFINET IO diagnostics" group under "PROFINET interface" in the "Diagnostics" folder of the Online and Diagnostics view.

"PROFINET IO diagnostics" group

Manufacturer-related diagnostics texts for the IO device or the selected module of the IO device are displayed in this area.

If the diagnostics can be read by the IO controller, then communication errors and configuration errors (online/offline discrepancies) are displayed here.

2.1.4.4 Reading out the diagnostics buffer of a CPU

Where do you read out the diagnostics buffer of a CPU?

You read out the diagnostics buffer of a CPU in the "Diagnostics buffer" group in the "Diagnostics" folder in the Online and Diagnostics view.

Structure of the "Diagnostics buffer" group

The "Diagnostics buffer" group consists of the following areas:

- "Events"
- "Settings"

Diagnostics buffer

The diagnostics buffer is used as a log file for the diagnostics events that occurred on the CPU and the modules assigned to it. These are entered in the order of their occurrence, with the latest event shown at the top.

"Events" area

The "Events" area consists of the following elements:

- Check box "CPU time stamp takes into account local PG/PC time"
- Event table
- "Freeze display" or "Cancel freeze" button

- Details of the event: Event no., event ID, module and possibly station or device name, rack/slot, description, help on the event, plant designation, location identifier, coming/going information, event type
- "Open in editor", "Save as" buttons

Check box "CPU time stamp takes into account local PG/PC time"

If you have not activated the check box, the diagnostics buffer entries are shown with the module time.

If you have activated the check box, the diagnostics buffer entries are shown with the time given by the following formula:

Displayed time = module time + time zone offset on your programming device / PC

This requires the module time to be identical to UTC.

You should use this setting if you wish to see the times of the diagnostics buffer entries for the module expressed in the local time of your programming device / PC.

Selecting or clearing the check box immediately changes the times displayed for the diagnostics buffer entries.

Note

If you use the "WR_SYS_T" instruction in your program or if you set the real-time clock of the CPU using an HMI device instead of using UTC, we recommend that you clear the "CPU time stamp takes into account local PG/PC time" check box. In this case, the module time is the sole time of concern.

Event table

The following information is displayed in the table for each diagnostics event:

- Sequential number of the entry
  The first entry contains the latest event.
- Date and time of the diagnostics event
  If no date and time are shown, the module has no integrated clock.
- Short name of the event and, if applicable, the reaction of the CPU

  Note

  If an individual parameter of a text cannot be determined, the character string "###" is shown in its place.

  If no display text is yet available for new modules or new events, the numbers of the events and the individual parameters are stated as hexadecimal values.

- Only for S7-1200 and S7-1500 CPUs: Icon for the event type
  The following table shows the available icons and their meaning.

Icon	Meaning
	OK (no maintenance and/or no fault)
	Maintenance required
	Maintenance demanded
	Error

- Icon for information related to incoming/outgoing status
  The following table shows the available icons and their meaning.

Icon	Meaning
	Incoming event
	Outgoing event
	Incoming event to which there is no independent outgoing event
	User-defined diagnostics event

You can change the order of the columns, adjust the column widths and remove and add individual columns in the event table. In addition, you can sort as follows: by sequential number, by "Date and time" and by "Event".

"Freeze display" or "Cancel freeze" button

The "Freeze display" or "Cancel freeze" button is only enabled when there is an online connection to the CPU.

The default setting is "Freeze display".

The following happens when you click the "Freeze display" button:

- The current display of the diagnostics buffer entries is frozen.
- The labeling of the button changes to "Cancel freeze".

If an error has occurred in your system, diagnostics events can occur very quickly in succession. This produces a high update rate on the display. Freezing the display allows you to calmly examine the situation in more detail.

If the display is frozen and you click the "Cancel freeze" button, the following happens:

- The display of the diagnostics buffer entries is updated again.
- The labeling of the button changes to "Freeze display".

Note

If you freeze the diagnostics buffer display, the CPU continues to enter events in the diagnostics buffer.

Details of the event

If you select a line in the list of events, you obtain detailed information on the respective event:

- Sequential number of the event in the diagnostics buffer
- Event ID
- Module and, if applicable, station or device name
- Rack/slot
- Description of the event with event-specific additional information. Examples of this additional information:
  - Command that caused the event
  - Operating mode switch caused by the diagnostics event
- Help on the event: The selected event is explained in more detail and possible solutions may be specified. With outgoing events, the text "Outgoing event: "No user action required" is displayed.
- Only for S7-1200 and S7-1500 CPUs: Plant designation, location identifier
- Information on whether the event is an incoming or outgoing event
- Type of event. The following types of event are possible:
  - OK (no maintenance and/or no fault)
  - Maintenance required
  - Maintenance demanded
  - Error

"Open in editor" button

The following table shows if the "Open block" button is active and which function it conceals.

When is the "Open in editor" button enabled?	What happens when you click this button?
If the diagnostics event references the relative address of a block. This is the address of the command that caused the event.	The "Open in editor" function opens the referenced block in the offline view at the programming instruction that causes the error. This allows you to check and, if necessary, change the source code of the block at the specified place and then download it again to the CPU.
If the diagnostics event was triggered by a module.	The "Open in editor" function opens the Device view of the module involved.

"Save as ..." button

If you click this button, the content of the diagnostics buffer is saved in a text file. "Diagnostics", depending on the language, with the extension ".txt" is suggested as the file name. You can however change this name.

"Settings" area

The "Settings" area consists of the following elements:

- "Display events" list
- "Apply settings as default" button
- "Output event information in hexadecimal format" check box

List "Display events:"

There is an check box in this list for every event class (default setting: all check boxes are selected). If you clear a check box, the events of that event class is no longer displayed in the "Events" area. Reselecting the check box displays the associated events once again.

"Apply settings as default" button

If you click this button, the settings are also applied to future occasions when the "Events" tab is opened.

"Output event information in hexadecimal format" check box

If you select the check box, the event IDs in the Events list of the "Events" area is displayed in hexadecimal format. If you clear the check box, the event information is given in text form.

See also

Basic information on the diagnostics buffer (Page 3351)

2.1.4.5 Saving service data (S7-1500)

Purpose

In the event of servicing it may be possible that the SIEMENS Customer Support requires very special information about the state of a module of your system for diagnostic purposes.

If such a case occurs in your system, you will be asked by Customer Support to save the service data of the module and send the resulting file to them.

Where do you carry out the saving of service data of a module?

You carry out the saving of service data of a module in its online and diagnostics view at the following points: In the "Functions" folder in the "Save service data" group"

The "Save service data" group consists of the following areas:

- Online data
- Saving service data

"Online data" area

This area shows the following data of the module:

- Article number
- Firmware version
- Module name (you configured this while configuring the hardware.)
- Rack
- Slot

Area "Save service data"

Proceed as follows to create and save a file with special service data:

1. Select the point in the file system at which you want to save the file:
   - You use the path preset in the "Path" field.
   - Click the three-dot (browse) button. In the dialog that opens specify the desired path and enter the file name.
2. Click the "Save data" button.

2.1.5 Changing the properties of a module or the programming device/PC

2.1.5.1 Changing the mode of a CPU

Requirement

There is an online connection to the CPU whose mode you want to change.

Procedure

To change the mode of the CPU, follow these steps:

1. Enable the "Online tools" task card of the CPU.
2. Click the "RUN" button in the "CPU control panel" pane if you want to change the CPU to RUN mode or the "STOP" button if you want to change the CPU to STOP mode.

   Note

   The only button active is the one that can be selected in the current operating mode of the CPU.

3. Acknowledge the confirmation prompt with "OK".

Or:

1. Open the "Online" menu.
2. Choose the "Start CPU" menu command if you want to set the CPU to RUN mode and "Stop CPU" if you want to set the CPU to STOP mode.

   Note

   The only button that is active is the one that can be chosen in the current operating mode of the CPU.

3. Acknowledge the confirmation prompt with "OK".

Or:

1. Click the "Start CPU" button in the toolbar if you want to set the CPU to RUN mode and the "Stop CPU" button if you want to set the CPU to STOP mode.

   Note

   The only button that is active is the one that can be chosen in the current operating mode of the CPU.

2. Acknowledge the confirmation prompt with "OK".

Result

The CPU will be switched to the required operating mode.

2.1.5.2 Performing a memory reset

Requirement

- There is an online connection to the CPU on which the memory reset is to be performed.
- The CPU is in STOP mode.

  Note

  If the CPU is still in RUN mode and you start the memory reset, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To perform a memory reset on a CPU, follow these steps:

1. Enable the "Online Tools" task card of the CPU.
2. Click the "MRES" button in the "CPU control panel" pane.
3. Acknowledge the confirmation prompt with "OK".

Result

The CPU is switched to STOP mode, if necessary, and the memory reset is performed on the CPU.

See also

Basics of a memory reset (Page 1296)

Basics of a memory reset (Page 719)

2.1.5.3 Determining and setting the time of day on a CPU

Where do I find the functions I need?

You determine and change the time of day on a CPU in the "Set time of day" group in the "Functions" folder of the Online and Diagnostics view. This requires an online connection.

Structure of the "Set time of day" group

The "Set time of day" group consists of the following areas:

- Area for reading out and setting the time of day
- Time system (This area does not exist for S7-1200 and will not be examined here.)

Structure of the area for reading out and setting the time of day

This area consists of the following parts:

- Programming device / PC time
  Here the time zone setting, the current date and the current time setting of your programming device / PC are displayed.
- Module time
  Here the date and time values currently read from the module (for example the CPU), are converted to local time and date and displayed.
  If the "Take from PG/PC" check box is selected, when you click the "Apply" button, the date and the PG/PC time converted to UTC are transferred to the module.
  If the "Take from PG/PC" check box is not selected, you can assign the date and time for the integrated clock of the module. After clicking the "Apply" button, the date and the time recalculated to UTC time are transferred to the module.

2.1.5.4 Resetting an S7-1200 CPU to factory settings (S7-1200)

Requirement

- There is no memory card inserted in the CPU.
- There is an online connection to the CPU that you want to reset to the factory settings.
- The CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the reset operation, you can place it in STOP mode by answering the security prompt with yes.

Procedure

To reset an S7-1200 CPU to factory settings, follow these steps:

1. Open the Online and Diagnostics view of the CPU.
2. Select the "Reset to factory settings" group in the "Functions" folder.
3. Select the "Keep IP address" check box if you want to keep the IP address or the "Delete IP address" check box if you want to delete the IP address.

   Note

   The two check boxes mentioned are only available if the module to be reset is able to choose whether to retain or delete the IP address.

4. Click the "Reset" button.
5. Acknowledge the confirmation prompt with "OK".

Result

The module is switched to STOP mode if necessary and the settings are then reset to factory settings. This means:

- The work memory and the internal load memory and all operand areas are cleared.
- All parameters are reset to their defaults.
- The diagnostic buffer is cleared.
- The time is reset.
- The IP address is kept or deleted depending on which setting you made.

2.1.5.5 Resetting an S7-1500 CPU to factory settings (S7-1500)

Requirement

- If you start a reset to factory settings from the project context, an online connection to the relevant CPU must exist.
- The relevant CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the reset operation, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To reset an S7-1500 CPU to factory settings, follow these steps:

1. Open the Online and Diagnostics view of the CPU (either from the project context or via "Accessible devices").
2. Select the "Reset to factory settings" group in the "Functions" folder.
3. Select the "Keep IP address" check box if you want to keep the IP address or the "Delete IP address" check box if you want to delete the IP address.

   Note

   With "Delete IP address", all IP addresses are deleted. This applies regardless of how you created the online connection.

   If a memory card is inserted, selecting the "Delete IP address" option causes the following: The IP addresses are deleted and the CPU is reset to factory settings. Then, the configuration (including IP addresses) that is stored on the memory card is transferred into the CPU (see below). If the memory card was formatted before resetting to factory settings or if it is empty, no IP address is transferred into the CPU.

4. Click the "Reset" button.
5. Acknowledge the confirmation prompt with "OK".

Result

The module is switched to STOP mode if necessary and the settings are then reset to factory settings. This means:

- The work memory and the internal retentive system memory and all operand areas are cleared.
- All parameters are reset to their defaults.
- The diagnostic buffer is cleared.
- The time of day is reset.
- The I&M data are deleted except for I&M0 data.

- The runtime meters are reset.
- The IP address is kept or deleted depending on which setting you made.
- If a memory card was inserted prior to the reset to factory settings, the configuration contained on the memory card (hardware and software) is downloaded to the CPU.

2.1.5.6 Resetting a PROFINET IO device to factory settings (S7-1500)

Requirements

- There must be an online connection to the PROFINET IO device that is to be reset.

Procedure

To reset a PROFINET IO device to factory settings, follow these steps:

1. Open the Online and Diagnostics view of the device (either from the project context or via "Accessible devices").
2. In the "Functions" folder, select the "Reset to factory settings" group.
3. Select the "Retain identification and maintenance data" option button or the "Delete identification and maintenance data" option button, depending on whether you want to retain or delete the identification and maintenance data IM0 to IM3 on the device.
4. Click the "Reset" button.
5. Click "OK" in response to the confirmation prompt.
   If the PROFINET IO device to be reset supports the Reset-to-Factory command according to PROFINET V2.3, its PROFINET device name, its IP address and the SNMP parameters (SNMP: Simple Network Management Protocol) are reset. Your selection in step 3 determines whether the identification and maintenance data is retained or deleted.
   The reset process is now complete.
6. If the PROFINET IO device to be reset does not support the Reset-to-Factory command according to PROFINET V2.3, however, you will see a message informing you that all data on the module will be lost during a reset to factory settings.
   Confirm the security prompt. The reset process will be conducted.
   If you have selected "Delete identification and maintenance data" in step 3, the identification and maintenance data IM0 to IM3 is deleted during the reset.
   However, if you have selected "Retain identification and maintenance data" in step 3, it can still happen that the identification and maintenance data IM0 to IM3 is deleted during the reset: The reset behavior depends on the module in this case.

Result

The PROFINET IO device is reset. The data deleted in the process depends on the factors described above.

2.1.5.7 Formatting an S7-1500 memory card (S7-1500)

Requirement

- If you start the formatting of the memory card from the project context, an online connection to the relevant CPU must exist.
- The relevant CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start a formatting operation, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To format an S7-1500 memory card, follow these steps:

1. Open the Online and Diagnostics view of the CPU (either from the project context or via "Accessible devices").
2. Select the "Format memory card" group in the "Functions" folder.
3. Click the "Format" button.
4. Answer the safety prompt with "Yes".

Result

- The memory card is formatted.
- The CPU is temporarily unavailable.
- The project data on the CPU are deleted with the exception of the IP address.
- If you start the formatting of the memory card from the project context, the Online and Diagnostics view remains open. If formatting is started via "Accessible devices", the Online and Diagnostics view will close.

2.1.5.8 Updating the firmware of a module

Basic information on the firmware update of a module

Performing a firmware update

Using firmware files, you can update the firmware of a module.

Requirements

- The module is connected online.
- The module supports a firmware update.
- For those modules that require supply voltage to perform the firmware update correctly: The supply voltage of the module is ensured. For details, see the documentation of the module.

Procedure

You can call up the firmware update of a module from the project tree or from the project context.

WARNING

Impermissible system states possible

An S7-1500 CPU immediately goes to STOP mode when you start the firmware update, which can have an effect on the operation of an online process or a machine.
Unexpected operation of a process or a machine can result in severe or fatal injuries and/or damage to property.

Note

After you have run a firmware update, you will need to replace the affected module with the same module with the current firmware version in the hardware configuration of your project. The engineering configuration then matches the actual physical configuration again.

"Run firmware after update" check box

If you have not selected the "Run firmware after update" check box, the previous firmware remains active until the module is reset (for example by power cycling). The new firmware only becomes active after the module has been reset.

If you have selected the check box, the module is automatically reset after the firmware has been downloaded and it then runs with the new firmware.

Activating the firmware following the update has the following consequences:

- A station executes a restart. All modules in the station become unavailable as a result.
- If the corresponding CPU is in RUN, activating the firmware can lead to access errors or other problems in the user program and might even mean that the CPU remains permanently in STOP.

  Note

  For some CPUs, the "Run firmware after update" check box is grayed out and cleared. In this case, you must restart the CPU manually.

  For S7-1500 CPUs, the "Run firmware after update" check box is grayed out and selected. In this case, the new firmware is activated immediately after the download operation.

Calling the firmware update from the project tree

Requirement

- The module is located in the first hierarchical level behind the head module (IM or CPU of the module family S7-1500 or ET200SP).
- The head module is connected to the engineering system via PROFINET.

Procedure

Follow these steps:

1. Select the relevant head module in the project tree.
2. Open the online and diagnostics view of the head module using the shortcut menu.
3. Select the "Firmware update" group in the "Functions" folder.
4. Select "Local modules".
   In the "Online data" area, a table lists modules of the first hierarchy level behind the head module. With Siemens modules, these are the modules that support a firmware update. With GSDML-based devices, this is all modules.
5. Select the module whose firmware you want to update.
6. Click the "Browse" button in the "Firmware loader" area in order to select the path to the firmware update files.
7. Select one of these files. The table then lists all modules for which an update is possible with the selected firmware file.
8. Optional: Select the "Run firmware after update" check box to reset the module after the load operation and to start the new firmware.
9. Click the "Start update" button. If the selected file can be interpreted by the module, it is downloaded to the module. If the operating mode of the CPU needs to be changed for this purpose, you will be prompted to do this by means of dialogs.

Starting the firmware update from the project context

Procedure

To perform a firmware update, follow these steps:

1. Open the module in the Online and Diagnostics view.
2. Select the "Firmware update" group in the "Functions" folder.

   Note

   For S7-1500-CPUs, this group is subdivided into "PLC" and "Display".

3. Click the "Browse" button in the "Firmware loader" area in order to select the path to the firmware update files.

4. Select one of these files. The table then lists all modules for which an update is possible with the selected firmware file.
5. Optional: Select the "Run firmware after update" check box to reset the module after the load operation and to start the new firmware.
6. Click the "Start update" button. If the selected file can be interpreted by the module, it is downloaded to the module. If the operating mode of the CPU needs to be changed for this purpose, you will be prompted to do this by means of dialogs.

2.1.5.9 Assigning an IP address to a PROFINET IO device

Basic information on assigning an IP address to a PROFINET IO device

Overview

All PROFINET IO devices work with the TCP/IP protocol and therefore require an IP address for operation on Industrial Ethernet. Once an IP address has been assigned to an IO device, it can be accessed via this address. You can then download configuration data or perform diagnostics, for example.

Requirement

- The Ethernet LAN connection must already be established.
- The Ethernet interface of your programming device or PC must be accessible.
- The IO device that is to be assigned an IP address must be in the same IP band as the programming device or PC.

Starting the address assignment via "Accessible devices"

Requirement

- The devices accessible via the associated interface of the PG/PC are displayed in the project tree (to display these, either double-click "Update accessible devices" in the project tree or select the "Accessible devices..." command in the "Online" menu.).
- You have double-clicked "Online access" -> <Selected interface> -> <PROFINET IO device> -> "Online & Diagnostics" in the project tree to open the Online and Diagnostics view.

Procedure

1. Open the "Functions" folder and the "Assign IP address" group inside this folder. The "MAC address" field displays the MAC address of the PROFINET IO device. The "Accessible devices" button is grayed out.
2. Enter the desired IP address.
3. Enter the subnet mask.

4. If a router is to be used, select the "Use router" check box and enter its IP address.
5. Click the "Assign IP address" button.

Result

The IP address is permanently assigned to the IO device or to the relevant PROFINET interface of the IO device. It is retained even through a startup or a power failure.

Note

For an S7-1500 CPU, you can also use the above-described method to change the IP address of a PROFINET interface, if a project has already been downloaded to the CPU via this interface. This overwrites the IP address downloaded via the project.

Starting the address assignment from the project context

Requirement

- An online connection to the PROFINET IO device exists.
- You have opened the Online and Diagnostics view of the PROFINET IO device from the project context.
- The PROFINET IO device is not assigned to any IO controller.

Procedure

1. Open the "Functions" folder and the "Assign IP address" group inside this folder.
2. Click the "Accessible devices" button in order to identify the devices that can be accessed. Note: For an S7-1500 CPU, there are two entries here because it has two PROFINET interfaces.
3. Select the IO device. The "IP address" field, "Subnet mask" field, "Use router" check box and "Router address" field are grayed out and contain the node properties you used to establish the current online access.
4. Click the "Assign IP address" button.

Result

The IP address is permanently assigned to the IO device or to the relevant PROFINET interface of the IO device. It is retained even through a startup or a power failure.

2.1.5.10 Assigning a PROFINET device name

Basic information on assigning a name to a PROFINET IO device

Device name

Before an IO device can be addressed by an IO controller, it must have a device name. This procedure was chosen for PROFINET because names are easier to handle than complex IP addresses.

Assigning a device name to a PROFINET IO device is comparable to setting the PROFIBUS address for a DP slave.

An IO device has no device name in its delivery state. For an IO controller to address an IO device, it must first be assigned a device name using the programming device or PC. It is now ready to transfer the configuration information including the IP address during startup or exchange user data in cyclic operation.

Rules for the device name

The device name is subject to the following limitations:

- Restricted to a total of 240 characters (lower case letters, numbers, dash, or dot)
- A name component within the device name, which is a character string between two dots, must not exceed 63 characters.
- No special characters such as umlauts, brackets, underscore, slash, blank space, etc. The only special character permitted is the dash.
- The device name must not begin or end with the "-" character.
- The device name must not begin with a number.
- The device name form n.n.n.n (n = 0, ... 999) is not permitted.
- The device name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).

Where do I find the function I am seeking?

You can assign the name of a PROFINET IO device at the following places:

- In the online and diagnostics view of the device in the "Assign name" group of the "Functions" folder. The user interface for this group differs depending on how you open the online and diagnostics view:
  - Open via "Accessible devices"
  - Open from the project context
- In the "Assign PROFINET device name" dialog

See also

Assigning a name in the online and diagnostics view opened via "Accessible devices" (Page 3344)

Assigning a name in the online and diagnostics view opened from the project context (Page 3344)

Assigning a name in the "Assign PROFINET device name" dialog (Page 3345)

Assigning a name in the online and diagnostics view opened via "Accessible devices"

Requirement

- You have opened the online and diagnostics view of the PROFINET IO device using "Update accessible devices" (in the project tree) or "Accessible devices..." ("Online" menu).

Procedure

1. Open the "Functions" folder and the "Assign name" group inside this folder. The "Type" field displays the module type of the PROFINET IO device.
2. Enter the required device name in the "PROFINET device name" input box.
3. Optional: Select the "Flash LED" check box in order to run an LED flash test on the PROFINET IO device. In this way, you verify that you are naming the desired IO device.

   Note

   The LED flash test is not supported by all PROFINET IO devices.

   The LED flash test runs until you cancel it. This is done, for example, by clearing the "Flash LED" check box or by closing the online and diagnostics view.
4. Click "Assign name".

Result

The entered name is assigned to the PROFINET IO device.

Assigning a name in the online and diagnostics view opened from the project context

Requirement

- You have opened the online and diagnostics view of the PROFINET IO device from the project context.
- The PROFINET IO device can be accessed using at least one PG/PC interface.

Procedure

1. Open the "Functions" folder and the "Assign name" group inside this folder. The "PROFINET device name" drop-down list displays the current name in the offline project, and the "Type" box shows the module type of the PROFINET IO device.

   Note

   For CPUs with several PROFINET interfaces, the names of all existing PROFINET interfaces are displayed in the offline project.

2. Choose a different name from the drop-down list, if necessary.

   Note

   In steps 3 to 5, you determine the IO devices that are present in the PROFINET subnet.

3. In the "PG/PC interface for assignment" drop-down list, select the PG/PC interface you want to use to establish the online connection.
4. Optional: Use the three check boxes to make a selection from all IO devices available online.
5. Click the icon for determining the IO devices present in the PROFINET subnet. The table is then updated.
6. Select the desired IO device in the table.
7. Optional: Select the "Flash LED" check box in order to run an LED flash test on the PROFINET IO device. In this way, you verify that you are naming the desired IO device.

   Note

   The LED flash test is not supported by all PROFINET IO devices.

   The LED flash test runs until you cancel it. This is done, for example, by clearing the "Flash LED" check box, by selecting another IO device in the table, or by closing the online and diagnostics view.

8. Click "Assign name".

Result

The selected name is assigned to the PROFINET IO device or one of its PROFINET interfaces.

Assigning a name in the "Assign PROFINET device name" dialog

Requirement

- You have opened the "Assign PROFINET device name" dialog from the network view (from the shortcut menu of a PN/IE connection).
- The PROFINET IO device can be accessed using at least one PG/PC interface.

Procedure

1. The following is displayed in the "PROFINET device name" drop-down list:
   - the name of the interface that was used to open the project in the current offline project
   - the names of those IO devices that are connected by means of this interface

   The "Type" field displays the module type of the PROFINET IO device.
   Choose a different name from the drop-down list, if necessary.

   Note

   In steps 2 to 4, you determine the IO devices that are present in the PROFINET subnet.

2. In the "PG/PC interface for assignment" drop-down list, select the PG/PC interface you want to use to establish the online connection.
3. Optional: Use the three check boxes to make a selection from all IO devices available online.
4. Click the icon for determining the IO devices present in the PROFINET subnet. The table is then updated.
5. Select the desired IO device in the table.
6. Optional: Select the "Flash LED" check box in order to run an LED flash test on the PROFINET IO device. In this way, you verify that you are naming the desired IO device.

   Note

   The LED flash test is not supported by all PROFINET IO devices.

   The LED flash test runs until you cancel it. This is done, for example, by clearing the "Flash LED" check box or by selecting another IO device in the table.
7. Click "Assign name".

Result

The selected name is assigned to the PROFINET IO device or the interface that was used to open the dialog.

2.1.5.11 Calibrating an S7-1500 analog module (S7-1500)

Calibrating an S7-1500 analog module - Overview (S7-1500)

Where do you calibrate an S7-1500 analog module?

You calibrate an S7-1500 analog module in its Online and Diagnostics view in the "Calibrate" group of the "Functions" folder.

Overview of the function scope of the calibrating function

You can perform the following functions for an S7-1500 analog module in the "Calibrate" group:

- Specifying the current calibration of all channels
- Calibrating a channel
- Canceling a running calibration
- Resetting the calibration of a channel to the factory settings

Requirement for the calibrating function described below

The following is required for the calibrating function described below:

- You have opened the Online and Diagnostics view from the project context (thus not from the project tree or via the "Online" menu).
- There is an online connection to the analog module that is to be calibrated.
- Offline and online configuration are identical.

Calibrating an S7-1500 analog module (S7-1500)

Overview of the calibration of a channel of an S7-1500 analog module

The calibration of a channel of an S7-1500 analog module consists of the following steps:

1. Start calibration
2. Perform the second step up to the next to last step of the calibration
3. Complete calibration

These steps are described in more detail in the following section.

Requirement

- You have opened the Online and Diagnostic view of the S7-1500 analog module from the project context and are in the "Calibrate" group of the "Functions" folder.
- The associated CPU is online.
- No calibration is currently running on the analog module (if you want to start the calibration) or the last step initiated has been performed successfully (if you want to resume or complete the calibration).

Procedure for starting the calibration

To start the calibration, follow these steps:

1. In the overview table, select the line that belongs to the channel to be calibrated.
2. Click the "Start manual calibration" button.

The user interface then changes as follows:

- The overview table and the "Start manual calibration" button and "Set to factory settings" buttons become inactive.
- The step display is activated and displays the numbers of the current and last steps.
- The "Command" field becomes active and indicates what the user must do in the next calibration step.
- The "Status" field becomes active and shows the current status of the calibration, e.g., "Calibration successfully started".
- The "Measured value" field becomes active. For an input module a value is displayed here; you must enter a value here for an output module.
- The "Cancel" button becomes active.
- With certain analog modules (AI Energy Meter) the "Skip" button becomes active. This button can be used to skip to the next step of the calibration.
- The "Next" button becomes active. This button can be used to advance to the next step of the calibration.

Procedure for the second to the next to last step of the calibration

Follow these steps:

1. Click the "Next" button.

The fields of the user interface described above are then updated.

With certain analog modules (AI Energy Meter), you can skip the next calibration step as long as the next step is not the last step of the calibration procedure. To do this, follow these steps:

1. Click the "Skip" button.

The fields of the user interface described above are then updated.

Procedure for the last step of the calibration

Follow these steps:

1. Click the "Next" button.

The user interface then changes as follows:

- The overview table becomes active.
- The calibration display of the calibrated channel is updated.
- The "Start manual calibration" button and "Set to factory settings" buttons become active.
- The step display is deactivated and the numbers of the current step and last steps are empty.
- The "Command" field becomes inactive and is empty.
- The "Status" field becomes inactive and shows the last status of the calibration, e.g., "Calibration successfully finished".
- The "Measured value" field becomes inactive and is empty.

- The "Cancel" button becomes inactive.
- The "Skip" button becomes inactive.
- The "Next" button becomes inactive.

Error occurrence

If an error occurs during the calibration, the module cancels the calibration. Afterwards, the channel that was to be calibrated has the same settings as before the start of the calibration.

Except for the "Status" field, the user interface appears the same after the occurrence of an error as before the calibration. The "Status" field displays the error that the module detected during the calibration.

Canceling a running calibration of an S7-1500 analog module (S7-1500)

Requirement

- You have opened the Online and Diagnostic view of the S7-1500 analog module from the project context and are in the "Calibrate" group of the "Functions" folder.
- The associated CPU is online.
- A calibration is currently running on the analog module.

Procedure

To cancel a running calibration, follow these steps:

1. Click the "Cancel" button.

Result

The running calibration is canceled, and afterwards the channel to be calibrated has the same settings as before the calibration.

All operator controls in the user interface are deactivated until the cancelation is complete. Except for the "Status" field, the user interface appears the same afterwards as before the calibration. The "Status" field displays the result of the cancelation.

Resetting an S7-1500 analog module to factory settings (S7-1500)

Requirement

- You have opened the Online and Diagnostic view of the S7-1500 analog module from the project context and are in the "Calibrate" group of the "Functions" folder.
- The associated CPU is online.

Procedure

To reset a channel of an S7-1500 analog module to factory settings, follow these steps:

1. Select the line associated with the channel to be reset in the overview table.
2. Click the "Set to factory settings" button.

Result

All operator controls in the user interface are deactivated until the reset operation is complete. Except for the "Status" field, the user interface appears the same afterwards as before the reset operation. The "Status" field displays the result of the reset operation.

2.1.5.12 Loading I&M data to PROFINET IO devices and your modules

Which I&M data can be loaded to PROFINET IO devices and your modules?

You can load I&M 1 data (plant designation and location identifier) and/or I&M 2 data (installation date) and/or I&M 3 data (additional information) to the actual hardware.

Requirements

- In the project settings (Options > Settings, Hardware configuration > Compiling and downloading), the option "Download I&M data" must be enabled.
- There is an online connection to the PROFINET IO devices and the modules to which you want to load I&M data.
- You have entered the I&M data you want to download in the properties of the respective PROFINET IO devices and your modules (Inspector window: "Properties" tab > "General" tab, Settings > Identification & Maintenance).

Where do I specify which I&M data is downloaded to which PROFINET IO devices?

You specify which I&M data you want to download to which PROFINET IO devices in the "Load preview" dialog. You will find the following alternatives in the drop-down list of the "Identification and maintenance data (I&M)" row:

- Load nothing
  The check boxes for all PROFINET IO devices as well as the check boxes for the loadable I&M data are cleared.
  No I&M data is transferred to the actual hardware during loading with this setting.
- Load data
  The check boxes for all PROFINET IO devices as well as the check boxes for the loadable I&M data are selected.
  The respective I&M 1, I&M 2 and I&M 3 data is transferred to all PROFINET IO devices during loading with this setting.
- Load selected
  You select the check boxes of those PROFINET IO devices to which you want to load I&M data. You also select the check boxes of the identification data you want to load.
  With this setting, you transfer the selected I&M data to the selected PROFINET IO devices during loading.

Note

Language dependency of the I&M data to be loaded

The I&M data are loaded to the real hardware in the form that you specified in the properties of the relevant PROFINET IO devices and your modules. There is no language dependency.

2.1.6 Diagnostics in STOP mode

2.1.6.1 Basic information on the diagnostics buffer

Function

The operating system of the CPU enters the errors detected by the CPU and the diagnostics-capable modules into the diagnostics buffer in the order in which they occurred. This includes the following events:

- Every mode change of the CPU (POWER UP, change to STOP mode, change to RUN mode)
- Every hardware and diagnostic error interrupt

The top entry contains the most recent event. The entries in the diagnostics buffer are stored permanently. They are retained even if the power supply fails and can only be deleted by resetting the CPU to factory settings.

A diagnostics buffer entry contains the following elements:

- Time stamp
- Error ID
- Additional information specific to the error ID

Advantages of the diagnostics buffer

The diagnostics buffer offers the following advantages:

- After the CPU has changed to STOP mode, you can evaluate the last events prior to the STOP so that you can locate and identify the cause of the STOP.
- You can detect and eliminate the causes of errors more quickly and thus increase the availability of the system.
- You can evaluate and optimize the dynamic system response.

Organization of the diagnostics buffer

The diagnostics buffer is a ring buffer. With the S7-1200 CPUs, the maximum number of entries is 50. When the diagnostics buffer is full and a further entry needs to be made, all existing entries are shifted one position (which means that the oldest entry is deleted) and the new entry is made at the first position that is now free (FIFO principle: first in, first out).

Evaluation of the diagnostics buffer

The contents of the diagnostics buffer can be accessed as follows:

- Using the Online and Diagnostics view

The evaluation of events occurring prior to the error event (e.g., transition to STOP mode) allows you to obtain a picture of the possible causes or to zero in more closely or specify in more detail the possible causes (depending on the error type).

Read the detailed information about the events carefully and use the "Help on event" text box to obtain additional information and possible causes of individual entries.

Note

To make the best use of the time stamp information on the diagnostics buffer entries in time-critical systems, it is advisable to check and correct the date and time of day on the CPU occasionally.

Alternatively, it is possible to perform a time-of-day synchronization using an NTP time server.

See also

Resetting an S7-1200 CPU to factory settings (Page 3335)

Determining the cause of a STOP of a CPU (Page 3353)

Determining and setting the time of day on a CPU (Page 3334)

Assigning the clock parameters (Page 1323)

2.1.6.2 Determining the cause of a STOP of a CPU

Requirement

The CPU you want to analyze is in STOP mode.

Procedure

To find out the reason why a CPU changed to STOP, follow these steps:

1. Open the online and diagnostics view of the CPU.
2. Select the "Diagnostics buffer" group from the "Diagnostics" folder.
3. Evaluate the events occurring prior to the transition to STOP mode. Use this to obtain a picture of the possible causes or to zero in on or specify in more detail the possible causes (depending on the error type).
   Read the detailed information about the events carefully and use the "Help on event" button to obtain additional information and possible causes of individual entries.

Result

You were able to zero in on or determine in more detail the cause of the CPU STOP.

Note

If the analysis does not enable you to overcome the problem, contact Customer Support. In this case, use the "Save as" button to back up the content of the diagnostics data to a text file and submit it to Customer Support.

See also

Reading out the diagnostics buffer of a CPU (Page 3327)

2.1.7 Online accesses in the Online and Diagnostics view

2.1.7.1 Displaying status of the online connection

Requirement

- The associated device can be accessed using at least one PG/PC interface.

Procedure

1. Open the online and diagnostics view for the device whose online connection status you want to display.
2. Select the "Online access" group.

   Note

   The "Online access" group exists only for CPUs and some CPs. However, if you have opened the online and diagnostics view using the "Show/update accessible devices" function, it is not displayed.

Result

The status of the online connection is displayed in the "Status" area both graphically and in text form.

2.1.7.2 Specifying a PG/PC interface, going online

Requirement

- The associated device can be accessed using at least one PG/PC interface.
- There is currently no online connection to the relevant device.

Procedure

1. Open the Online and Diagnostics view of the device to which you want to establish an online connection.
2. Choose the "Online access" group and the "Online access" area within this group.

   Note

   The "Online access" group exists only for CPUs and some CPs. If you have opened the Online and Diagnostics view using the "Show/update accessible devices" function, it is not displayed.

3. If an online connection was established previously for the device, the associated data for this online connection is preset in the drop-down lists. In this case, you can immediately continue with the last step of this operating instruction, provided you have not changed the IP address in the meantime using the Online and Diagnostics view.
4. Choose the interface type in the "Type of PG/PC interface" drop-down list.
   The "PG/PC interface for online access" drop-down list then shows only the interfaces of the programming device or PC that match the selected interface type.
5. In the "PG/PC interface for online access" drop-down list, select the programming device or PC interface via which you want to establish the online connection.
6. Optional: Click the "Properties" button to change the properties of the associated CP.

7. In the "Connection to subnet" drop-down list, select the subnet via which the device is connected to the PG/PC interface.

   Note

   The PG/PC interface is connected to an interface of a device.

   If you only want to access this device, select the setting "Directly at slot <interface name>" in the drop-down list.

   If, however, you want to access another device by means of routing, create a subnet at this interface in the device configuration and then select this subnet in the drop-down list.

8. If the device is accessible via a gateway, select the gateway that connects the two subnets involved in the "1st gateway" drop-down list.
9. In the "Device address" entry field, enter the IP address of the device to which you want to establish an online connection, if necessary.

   Note

   For CPUs with multiple IP addresses, select the IP address of the PROFINET interface you want to use to establish an online connection from the "Device address" drop-down list.

10. Alternatively: Click the "Show accessible devices" button and choose the device from the list of accessible devices to which you want to establish an online connection.
11. Click the "Go online" button.

Result

The online connection to the desired device is established.

2.1.7.3 Going offline

Requirement

- There is currently an online connection to the relevant device.

Procedure

1. Open the online and diagnostics view of the device for which you want to disconnect the online connection.
2. Choose the "Online access" group and the "Online access" area within this group.

   Note

   The "Online access" group exists only for CPUs and some CPs. However, if you have opened the online and diagnostics view using the "Show/update accessible devices" function, it is not displayed.

3. Click the "Go offline" button.

Result

The online connection to the desired device will be disconnected.

2.1.7.4 Performing the flash test for a device with an online connection

Requirement

- There is currently an online connection to the relevant device.
- The FORCE function is not active.

Procedure

1. Open the online and diagnostics view of the device for which you want to perform a flash test.
2. Choose the "Online access" group and the "Status" area within this group.

   Note

   The "Online access" group exists only for CPUs and some CPs. However, if you have opened the online and diagnostics view using the "Show/update accessible devices" function, it is not displayed.

3. Select the "LED flash test" check box.

Result

- On an S7-1200 CPU, the RUN/STOP, ERROR and MAINT LEDs flash.
- On an S7-1500 CPU, the RUN/STOP, ERROR, and MAINT LEDs flash.
- On an S7-300 or S7-400 CPU, the FRCE LED flashes.

The LEDs flash until you cancel the flash test. This is done, for example, by clearing the "LED flash test" check box, by changing to another group of the online and diagnostics view, or by changing settings in the "Online access" area.

2.1.8 Checking PROFIBUS DP subnets for faults

2.1.8.1 Basic information on the diagnostic repeater

What is the diagnostic repeater?

The diagnostic repeater is a repeater that can monitor a segment of an RS 485 PROFIBUS subnet (copper cable) during operation and signal line errors to the DP master via a diagnostics message frame.

The diagnostic repeater detects, localizes and visualizes line errors during operation at an early stage. As a result, problems in the system are identified early and production downtimes will be minimized.

Function of the diagnostic repeater

The diagnostic repeater can perform line diagnostics on the DP2 and DP3 segments because it has a measuring circuit for these segments.

The line diagnostics run in two steps:

- Step 1: Topology determination
  You start the topology determination by calling the "DP_TOPOL" instruction in your program. The diagnostic repeater then determines the PROFIBUS addresses and the distance of the devices and creates a topology table.
- Step 2: Error localization
  The diagnostic repeater checks the lines during operation. It determines the distance to the point of the error and the reason for the error; it then issues a diagnostics alarm with relative information on the error location.

Display of detailed information on the determined error location

You receive detailed information on the determined error location in the Online and Diagnostics view of the diagnostic repeater.

- By means of icons
- By means of a display with graphics and text

See also

Displaying the status of the segment diagnostics using icons (Page 3357)

Displaying the status of the segment diagnostics using graphics and text (Page 3358)

2.1.8.2 Displaying the status of the segment diagnostics using icons

Where do I find the information I need?

The icons for the status of the segment diagnostics are available:

- In the expanded "Segment diagnostics" folder in the navigation pane of the Online and Diagnostics view of the relevant diagnostic repeater.

The diagnostics icon associated with the segment will be displayed behind the segment designation. It must be noted here that line errors will be displayed for the DP2 and DP3 segments only. The DP1 and programming device segments do not display errors in the form of a diagnostics icon; rather, they signal only a few bus errors.

Diagnostics icons

The following table shows the available icons and their meaning.

Icon	Meaning
	Segment is error-free
	Segment contains errors
	Segment is deactivated

2.1.8.3 Displaying the status of the segment diagnostics using graphics and text

Where is the status of the segment diagnostics displayed with graphics and text?

The status of the segment diagnostics will be displayed using graphics and text in the "DP1", "DP2", "DP3", and "PG" groups of the "Segment diagnostics" folder in the Online and Diagnostics view of the relevant diagnostic repeater.

Structure of the "DP1", "DP2" "DP3", and "PG" groups

The "DP1", "DP2", "DP3", and "PG" groups consist of the following elements:

- "Error location" field
- "Error" field
- "Resolution" field
- "Help on event" button
- "Freeze display" or "Cancel freeze" button

"Error location" field

This field displays the error location graphically, provided the diagnostic repeater can determine the location.

The following picture shows an example for a line error occurring in the DP2 segment.

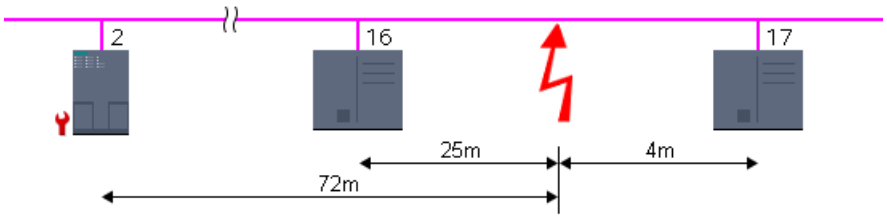

In this example, the diagnostic repeater has the PROFIBUS address 2, and a line error has occurred between the devices with PROFIBUS addresses 16 and 17. This line error is located 25 m from device 16, 4 m from device 17, and 72 m from the diagnostic repeater.

"Error" field

The error is explained in plain text in this field.

"Resolution" field

Here, you will find actions for resolving the error.

"Help on event" button

Click this button to obtain a more detailed explanation of the error and additional details on resolving the error, if applicable.

"Freeze display" or "Cancel freeze" button

The "Freeze display" or "Cancel freeze" button is only enabled when there is an online connection to the diagnostic repeater.

The default setting is "Freeze display".

The following happens when you click the "Freeze display" button:

- The current display of the segment diagnostics is frozen.
- The labeling of the button changes to "Cancel freeze".

If the display is frozen and you click the "Cancel freeze" button, the following happens:

- The display of the segment diagnostics is updated again.
- The labeling of the button changes to "Freeze display".

2.2 Connection diagnostics

2.2.1 Overview of connection diagnostics

Basics

Connection diagnostics, as described below, refers to the diagnostics of communication connections.

The connection diagnostics is started each time an online connection is established to a module (CPU or CP) that participates in one or more communication services. The connection status is updated automatically in the background.

In the case of one-way connections, an online connection must exist to the communication partner that has established the communication connection.

On connections configured at both ends, a distinction between the following two situations must be made:

- If there is an online connection to only one connection endpoint, only the part of the connection belonging to this connection endpoint can be diagnosed.
- If there is an online connection to both connection endpoints, both parts of the connection (and therefore the entire connection) can be diagnosed.

Basic connections diagnostics options

Connection diagnostics can be performed as follows:

- Using icons on the connection status display
  This display is generated in the connection table.
- Through detailed connection diagnostics
  This step is available in the "Diagnostics > Connection information" area of the Inspector window.

Requirement for the connection diagnostics described below

The connection diagnostics is only possible if the devices participating in the connection make the corresponding diagnostics data available.

You can display the details of either all the communication connections created in the project (default) or selected communication connections in the connection table.

The connection diagnostics described in the following assume that you display the details of selected communication connections. To do this, clear the "Show all connections" option in the shortcut menu.

See also

Working with the connection table (Page 222)

2.2.2 Displaying the connection status using icons

Content of connection table without an online connection

- For a CPU or CP, the connection table lists the communication connections (including properties) configured offline, if an online connection is not established.

Content of connection table with an online connection

After the online connection has been established, the properties of the communication connections listed offline will be expanded to include diagnostics icons for the connection status ("Online status" column).

In addition, entries for all communication connections that exist online only (e.g., connections for the instructions for Open User Communication, programming device and OP connections, connections for web server access) will now be added to the connection table.

For connections that exist online or offline only, the diagnostics icon at the bottom right is combined with a smaller additional comparison status icon.

Diagnostics icons for communication connections

The following table shows the diagnostics icons for communication connections.

Icon	Meaning
	Connection setup
	Connection not setup / is being setup
	Connection not available

Diagnostics icons for the comparison status

The diagnostic icons for communication connections can be combined at the bottom right with additional smaller icons that indicate the result of the online/offline comparison. The following table shows the available comparison icons and their meaning.

Icon	Meaning
	Connection exists online only
	Connection exists offline only

2.2.3 Detailed connection diagnostics

2.2.3.1 Detailed connection diagnostics - overview

Where do I perform detailed connection diagnostics?

To perform detailed connection diagnostics, go to the "Diagnostics > Connection" information of the Inspector window.

How do I open the "Diagnostics > Connection information" area of the Inspector window?

The following options are available for opening the "Connection information" tab of the Inspector window.

- Select the line of the relevant connection in the connection table. Click the "Diagnostics" and "Connection information" tabs one after the other in the Inspector window.
- Double-click the diagnostics icon of the relevant connection in the connection table.
- This step takes you to the programming editor for a S7 communication instruction or open user communication instruction. Double-click the diagnostic icon of the instruction (stethoscope).

Structure or the "Diagnostics > Connection information" area of the Inspector window

Requirements: the content of the "Connection information" tab has been filled, and an online connection to at least one end point of the relevant connection has been established.

If a module has been selected (network view), the tab will contain the following group:

- Connection resources (for S7-1200 and S7-1500)

If a connection has been selected (connection table), it will contain the following groups:

- Connection details
- Address details of the connection (for S7-1200 and S7-1500)

2.2.3.2 Determining online connection resources for S7-1200 (S7-1200)

Where do you determine the online connection resources?

The online connection resources are obtained in the "Connection resources" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window. It is displayed only if you have selected a module in the network view to which an online connection exists.

Number of connection resources

- Maximum number: Specifies the maximum number of available connection resources of the module.
- Not assigned: Indicates how many connection resources are not yet assigned. If connection resources are already reserved for certain types of communication, then the unreserved connection resources cannot always be used for the various connection types.

Reserved and currently assigned connection resources

For the communication types indicated below, the connection resources that are reserved and currently assigned by the module will be displayed.

Communication type	Meaning
PG communication	Resources for connections between the module and programming devices (for example, for the establishment of a connection from the project tree, for online diagnostics, etc.)
HMI communication	Resources for connections between the module and HMI devices
Open User Communication	Resources for connection of open user communication instructions
S7 communication	Resources for configured S7 connections, through which data can be exchanged by calling instructions in the user program.
Other communication	Specifies other assigned connection resources for which connection resources are not reserved.

2.2.3.3 Determining online connection resources for S7-1500 (S7-1500)

Where do you determine the online connection resources?

The online connection resources are obtained in the "Connection resources" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window. It is displayed only if you have selected a module in the network view to which an online connection exists.

Description of the detailed display of the connection resources

The detailed display of the connection resources includes:

- Number of available connection resources
- Number of configured connection resources
- Number of connection resources still available

For a description of these, go to here .

2.2.3.4 Determining connection details

Where do I determine the connection details?

The connection details are obtained in the "Connection details" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window.

When is the "Connection details" group filled in?

The following requirements must be met to fill in the "Connection details" group on the "Connection information" tab:

- An online connection to the end point of the relevant connection must exist.
- You have selected a line in the connection table.

Structure of the "Connection details" group

The "Connection details" group consists of the following elements:

- Local ID (hex)
- Connection type (for S7-1200 and S7-1500)
- Protocol
- Connection status: icon and description
- Details
- Last status change (for S7-300 and S7-400 only)

2.2.3.5 Determining the address details of a connection (S7-1200, S7-1500)

Where do I determine the address details of a connection?

The address details of a connection are obtained in the "Connection address details" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window.

For which CPUs is the "Connection address details" group available?

The "Connection address details" group of the "Connection information" tab is available for S7-1200 and S7-1500 CPUs.

When is the "Connection address details" group filled in?

The following requirements must be met to fill in the "Connection address details" group on the "Connection information" tab:

- An online connection to the end points of the relevant connection must exist.
- You have selected a line in the connection table.

Structure of the "Connection address details" group

The address details relevant to the connection type are specified for the two communication partners.

2.3 System diagnostics with 'Report System Errors'

2.3.1 Introduction to system diagnostics with 'Report System Errors'

Introduction

When a system error occurs, hardware components and devices of other manufacturers (slaves whose properties are set by the GSD files) can trigger organization block calls.

Example: If a wire breaks, a module with diagnostics capability can call OB 82.

The hardware components provide information on the system error that has occurred. The start event information, in other words, the local data of the assigned OB (including data record 0), provides general information on the location (for example, logical address of the module) and type (for example channel error or module fault) of the error.

The problem can also be specified in greater detail using additional diagnostics information (reading data record 1 with the "RDSYSST" instruction or reading the diagnostics frame from standard DP slaves with the "DPNRM_DG" instruction): For example, channel 0 or 1, broken wire or measuring range exceeded.

System diagnostics with "Report System Errors" (RSE) provides you with a convenient way of evaluating this diagnostics information for S7-300/400 PLCs, ET200S, ET200Pro and software PLCs and to display it in the form of messages. The required blocks and alarm texts are created in the properties of the particular PLC. You only need to download the generated blocks to the CPU and, if required, transfer the texts to connected HMI devices.

To display diagnostics events graphically, for example on an HMI device or via a Web server, you can generate one or more status DBs. These status DBs are updated by the system diagnostics blocks and then contain information on the current state of the system.

2.3.2 Basics of system diagnostics

Using system diagnostics with "Report System Errors", you can generate blocks that analyze errors in the system and generate alarms with a textual error description and the error location.

These alarms are defined per component with alarm capabilities (for example channel errors or rack errors) and are limited to 255 alarms component with alarm capability. You will receive a message if this limit is exceeded.

General settings that you make or change in system diagnostics are stored along with the project. The influence on the system diagnostics only takes effect after generating the blocks, compiling the hardware configuration and downloading the configuration to the components involved.

Recommended procedure

Make the settings for reporting system errors and the structure of the alarms. Specify which diagnostics blocks will be created and configure the status DBs. Generate the blocks and download the configuration to apply the changes.

You will find detailed information in the sections below.

Note

If you use system diagnostics, the system response of the plant may change if an error occurs. For example, the CPU may not change to "STOP" mode as it would without system diagnostics.

Make sure that all protective mechanisms of the plant are working properly.

2.3.3 Modules with limited diagnostic possibilities

Introduction

The components of the S7-300 stations, S7-400 stations, PROFINET IO devices, DP slaves, and WinAC are supported by "Report System Errors", provided that they support functions such as diagnostic interrupts, pull/plug interrupts, and channel-specific diagnostics.

Components not supported

The following components are not supported by "Report System Errors":

- M7 configurations
- PROFIBUS DP configurations using a DP master interface module (CP 342-5 DP) in S7-300 stations
- PROFINET IO devices using an external controller (CP 343-1 Advanced) in S7-300 stations

Please note that a hot restart can also result in missing interrupt alarms. This behavior is due to the fact that the alarm acknowledgment memory of the CPU is not cleared during a hot restart, but "Report System Errors" resets the internal data. Not all of the module or channel faults that occur before startup or during a failure will be reported.

A maximum of 8 channel faults are reported per module.

Note

If you are using a CP 443-5 and it is in STOP mode, a master system failure is not reported at startup.

PROFIBUS DP

For all PROFIBUS DP slaves, station faults (failure/return) are indicated with a plain text alarm.

For all PROFIBUS DP slaves, the manufacturer-specific diagnostics are supported with the following limitations:

- Only V1 slaves are supported. These are slaves in whose GSD file the entry "DPV1_Slave=1" is set.
- The DP interrupt mode for this slave must be set to "DPV0".

Diagnostics repeater

The alarms of the diagnostics repeater are output as plain text in DPV0 mode. The texts are read out from the GSD file.

PROFINET IO

Below you will find the diagnostics of the various PROFINET IO devices that are supported by "Report System Errors".

Errors that occur during operation (CPU in RUN):

- Device faults (failure, return) are supported
- Module faults and submodule faults (Module/submodule removed, Wrong module/submodule, Compatible module/submodule) are supported
- Channel faults are supported and extended channel fault information is evaluated

AS-Interface

For AS-Interface slaves, an alarm is sent if the actual configuration does not match the expected configuration.

Shared devices

"Report System Errors" evaluates the information as to whether a (sub)module has been configured as a shared device. Because "Report System Errors" is always looking at a CPU, only the (sub)modules that have the full access setting for this CPU are taken into consideration for the diagnostics. The (sub)modules configured as "not assigned" are not evaluated.

2.3.4 Diagnostics blocks for reporting system errors

Once you have made the settings for reporting system errors, the required blocks (FB with assigned instance DB and one or more global DBs, an FC and, depending on the setting, also OBs that do not yet exist) and status DBs are created the next time you compile the hardware configuration.

The following blocks are created:

- Diagnostics FB (default: FB 49),
- Instance DB for the diagnostics FB (default: FB 49),
- Global DB (default: FB 50),
- Diagnostics FC (default: FC 49),
- Error OBs (are automatically created if the CPU supports this function),
- Cyclic OBs (if you have selected this option in the "OB configuration" group box),
- Optional status DBs

The created blocks (except OBs) are stored in the project tree under "Program blocks > System blocks > Report System Errors".

Note

The block numbers set for the created system diagnostics blocks must not overlap existing block numbers. Moreover, the numbers must not overlap the numbers of blocks located in libraries.

If problems occur in the generation of blocks, check whether the preset numbers (FB 49, DB 49, FC 49, DB 50) have been changed. If this is the case, try to repeat the generation with the preset block numbers or use numbers which you are certain are not in use elsewhere.

See also

Properties of the blocks (Page 3369)

2.3.5 Properties of the blocks

Diagnostics blocks

The created diagnostics blocks (FB with assigned instance DB and one or more global DBs and an FC) evaluate the local data of the error OB and read any additional diagnostics information from the hardware component that triggered the error.

They have the following properties:

- Created in RSE language (report system errors) (also applies to the blocks listed above)
- Know-how protected (also applies to the blocks listed above)
- Delay incoming interrupts during runtime

Status DBs

The status DBs act as the interface for the diagnostics blocks and allow diagnostics events to be displayed graphically on an HMI device or a Web server.

See also

Settings for system diagnostics blocks (Page 3376)

2.3.6 Supported error OBs

The following error OBs are created automatically if they are supported by the configured CPU:

OB	Diagnostics FB call	Description
OB 70 (IO redundancy error)	possible	This OB exists only with H-CPUs.
OB 72 (CPU redundancy error)	possible	This OB exists only with H-CPUs.
OB 73 (communications redundancy error)	possible	This OB exists only with a few H-CPUs.
OB 81 (power supply error)	possible	
OB 82 (diagnostics interrupt OB)	possible	
OB 83 (pull/plug interrupt)	possible	
OB 85 (program execution error)	not possible	If 'RSE' creates this OB when it is generating the diagnostics blocks, additional networks are inserted to implement the following program sequence: • If errors occur when updating the process image (for example removing the module), the CPU is prevented from changing to STOP so that the diagnostic FB can be run in OB 83. If there is a setting for "CPU STOP" after an alarm from "RSE", this takes effect in OB 83. • With all other error events of OB 85, the CPU changes to STOP.
OB 86 (failure of an expansion rack, a DP master system or a distributed I/O device)	possible	

If the OBs already exist...

Existing error OBs on not overwritten. If appropriate, the call for the diagnostics FB is appended in the existing OB.

If the configuration includes distributed I/O devices...

To evaluate errors in the distributed I/O, the created FB automatically calls the "DPNRM_DG" instruction (read diagnostics data of the DP slaves). For this function to operate correctly, the created FB must be called either only in OB 1 or in a cyclic interrupt OB with a short call interval and in the startup OBs.

Note

Please note the following:

- Due to the OB 85 created by the system diagnostics, the CPU does not change to STOP if the "error updating the process image" error event occurs.
- OB 85 is also called by the CPU in the following error situations.
  - "Error event for an OB that is not loaded"
  - "Error calling or accessing a block that is not loaded"
    When these errors occur, the OB 85 generated by the system diagnostics still changes the CPU to STOP, just like before the use of system diagnostics.

2.3.7 Diagnostics status DB

Within system diagnostics, you have the option of creating a status DB. This status DB is updated by the system diagnostics blocks and then contains information on the current system status.

Interface for the diagnostics status DB

The created data block (default: DB 127) allows you to query the system state of a configured component and, if required, all its lower-level components.

This data block is required to support system diagnostics using the Web server or diagnostics viewer. As default, it is enabled.

Note

Please note that system diagnostics must be activated both on the Web server or diagnostics viewer and in the Inspector window of the PLC to be able to display diagnostics data of the status DBs.

After restarting a CPU with Web server, the module state is displayed following a delay. To shorten the waiting time, you can call the RSE diagnostics block in a cyclic interrupt OB.

Structure of the diagnostics status DB

Address	Name	Data type	Description
+0	Directory		
0	D_Version	WORD	Version supported by system diagnostics
2	D_pGlobalState	WORD	Byte offset to the start of the "GlobalState" section
4	D_pQuery	WORD	Byte offset to the start of the "Query" section
6	D_pComponent	WORD	Byte offset to the start of the "Component" section
8	D_pError	WORD	Byte offset to the start of the "Error" section
10	D_pState	WORD	Byte offset to the start of the "State" section
12	D_pAlarm	WORD	Byte offset to the start of the "Alarm" section
14	D_pSubComponent	WORD	Byte offset to the start of the "Subcomponent" section
+16	GlobalState		
0	G_EventCount	WORD	ID of the last event (counter)
2.0	G_StartReporting	BOOL	Startup evaluation active
+20	Query		
0	Q_ClientID_User	DWORD	ID of the client; here, please use a value between 1 and 255. Make sure that different clients use different IDs.
4	Q_ClientID_Intern	DWORD	ID of the client (internal)
8.0	Q_WithSubComponent	BOOL	With/without status of the lower-level components (slower)
8.1	Q_SubComponentAlarm	BOOL	AS-i master returns AS-i slave interrupts
8.2	Q_Reserved2	BOOL	Reserved
8.3	Q_Reserved3	BOOL	Reserved
8.4	Q_Reserved4	BOOL	Reserved
8.5	Q_Reserved5	BOOL	Reserved
8.6	Q_Reserved6	BOOL	Reserved
8.7	Q_Reserved7	BOOL	Reserved
9.0	Q_Start	BOOL	Start query
10	Q_Error	BYTE	Internal error in query
11	Q_Reserved8	BYTE	Reserved
+32	Component		
0	C_AddressMode	BYTE	Addressing mode of the module
1	C_Reserved1	BYTE	Reserved
2	C_ComponentID	WORD	Hardware ID of the component (internal)
+36	Error		
0	E_ErrorNo	WORD	Index of the requested/actual error
2.0	E_LastError	BOOL	Is set when E_ErrorNo does not equal 0. Value TRUE if E_ErrorNo is the index of the last error otherwise FALSE
+40	State		

Address	Name	Data type	Description
0	S_Hierarchy	BYTE	Reserved
1	S_Periphery	BYTE	Reserved
2.0	S_SupFault	BOOL	The component cannot be reached
2.1	S_NotAvailable	BOOL	The component does not exist
2.2	S_Faulty	BOOL	The component is faulty, the "Alarm" section is not empty
2.3	S_MoreErrors	BOOL	There are more errors than system diagnostics can store
2.4	S_Maintenance1	BOOL	A maintenance request is pending
2.5	S_Maintenance2	BOOL	A maintenance demand is pending
2.6	S_Deactivated	BOOL	The component was deactivated *)
2.7	S_Reserved2	BOOL	Reserved
3.0	S_SubFault	BOOL	A subcomponent is faulty
3.1	S_SubMaintenance1	BOOL	A maintenance request is pending for a subcomponent
3.2	S_SubMaintenance2	BOOL	A maintenance demand is pending for a subcomponent
3.3	S_SubDeactivated	BOOL	A subcomponent is deactivated
3.4	S_Reserved4	BOOL	Reserved
3.5	S_Reserved5	BOOL	Reserved
3.6	S_Reserved6	BOOL	Reserved
3.7	S_Reserved7	BOOL	Reserved
4	S_TIAMS	DWORD	Maintenance state of the component
8	S_TIAMSChannelExist	DWORD	Maintenance state: Configured channels
12	S_TIAMSChannelOK	DWORD	Maintenance state: Faulty channels
16	S_ChannelCount	WORD	Number of channels; valid only when Q_WithSub-Component is set
18.0	S_ChannelVector	ARRAY [0 to 255] BOOL	Number of channels affected; valid only when Q_WithSubComponent is set
+90	Alarm		
0	A_ComponentID	WORD	Hardware ID of the component (internal)
2	A_TextID1	WORD	ID of the first error text
4	A_TextLexikonID1	WORD	ID of the first error text lexicon
6	A_HelpTextLexikonID1	WORD	ID of the first help text lexicon
8	A_MapTextID	WORD	ID of the first error text (HMI)
10	A_MapHelpTextID	WORD	ID of the first help text (HMI)
12	A_TextID2	WORD	ID of the second error text
14	A_TextLexikonID2	WORD	ID of the second error text lexicon
16	A_HelpTextLexikonID2	WORD	ID of the second help text lexicon
18	A_MapTextID2	WORD	ID of the second error text (HMI)
20	A_MapHelpTextID2	WORD	ID of the second help text (HMI)
22	A_AlarmID	DWORD	Alarm number

Address	Name	Data type	Description
26	A_ValueCount	WORD	Number of further bytes occupied (12)
28	A_AssociatedValue	ARRAY [1 to n] WORD	Associated values of the alarm n = A_ValueCount / 2 (= 6)
+130	SubComponent		
0	U_SubComponentCount	WORD	Number of subcomponents
2	U_SubComponentFault	ARRAY [1 to n] BYTE	List of subcomponents "n" depends on the configuration **)

*) If the component was deactivated, the index of the requested/actual error is not changed and "E_LastError" is set to "true". The tag area of the alarm is also not filled in.

**) The list of subcomponents is valid only when Q_WithSubComponent is set. The ARRAY has a status byte for each configured component. For a master, the ARRAY contains the status of the configured stations sorted in ascending order according to the station ID. For a station, the ARRAY contains the status of the configured slots sorted in ascending order according to slot number. This field can contain a maximum of 4096 entries (for an IO system): only the actual maximum size is displayed.
The status byte for each lower-level component is defined as follows:
Bit 0 = SubFault: the component cannot be reached
Bit 1 = Fault: the component is not available or is faulty
Bit 2 = Maintenance1: the component has reported a need for maintenance
Bit 3 = Maintenance2: the component has reported a need for maintenance
Bit 4 = Deactivated: the component was deactivated
Bit 5 = SubFault: a subcomponent is faulty
Bit 6 = SubMaintenance1: a subcomponent has reported a need for maintenance
Bit 7 = SubMaintenance2: a subcomponent has reported a need for maintenance

2.3.8 Displaying settings of system diagnostics

Requirement

You are in the "Common data" folder in the project tree.

Procedure

To display and edit the settings for system diagnostics, follow these steps:

1. Double-click the "System diagnostics settings" entry.
2. You can view and edit the settings in the system diagnostics.

2.3.9 Settings

2.3.9.1 General settings

Activate system diagnostics for this PLC

This option is deactivated by default.

Note

When this option is cleared, diagnostics data is also not read out in the Web server and diagnostics viewer, even if the option is selected there.

If you clear this option on a PLC on which it was previously selected, all the diagnostics data is deleted the next time you compile the hardware configuration.

2.3.9.2 Alarm settings

For the alarm categories listed below, you can define whether an alarm is output and whether or not this alarm requires acknowledgment:

- Error
- Maintenance demanded
- Maintenance required
- Info

To do this, select the relevant options in the "Alarm" and "Acknowledgment" columns.

The set alarm class is displayed in the "Alarm class" column.

2.3.9.3 Settings for diagnostics support

With the help of diagnostics support, diagnostics data is made available in special status DBs by the diagnostics FB. This can then be displayed graphically, for example on an HMI device.

Status DB

Enable the status DB in order to read out the current system status of configured components using this data block.

The following status DB is preset by default:

Block	Name	Number
Diagnostics status DB	RSE_DIAGNOSTIC_STA-TUS_DB	127

You can, however, assign a different name and/or number for this block, as long as it is not already being used.

If the "Enable Web server on this module" functionality is enabled in the properties of the CPU, the diagnostics status DB must be enabled. As default, DB 127 is generated if the CPU supports the Web server and if this is enabled in the properties of the CPU.

Extended diagnostics settings

This query is only possible if the PLC includes the "D_ACT_DP" instruction.

Enable the option "Query for status "activated/deactivated" after PLC startup" if you want the status of slaves to be queried.

Enable the "Send alarm if status changes from/to activated or deactivated" if you want an alarm to be output when there is a status change. To use this option, execute the "D_ACT_DP" instruction in mode 3/4 and select the diagnostics DB call in OB 86 in the "OB configuration" group.

2.3.9.4 Settings for system diagnostics blocks

As default, the following values are set for the system diagnostics blocks:

Block	Name	Number
Diagnostics FB	RSE_FB	49
Diagnostics DB	RSE_DB	49
first global DB	RSE_GLOBAL_DB	50
Diagnostics FC	RSE_FC	49

You can, however, assign different numbers and/or names for these blocks as long as they are not already being used.

2.3.10 Exporting and importing system diagnostic settings

2.4 System diagnostics for S7-1500 PLCs (S7-1500)

2.4.1 Introduction (S7-1500)

Introduction

When a system error occurs, hardware components and devices of other manufacturers (slaves whose properties are set by their GSD file) can trigger a system reaction. The hardware components provide information on the system error that has occurred.

For S7-1500 PLCs, the system diagnostics provides a convenient way to evaluate this diagnostic information and display it in the form of alarms.

In addition, additional diagnostic information allows the error to be specified in more detail, e.g., channel "switch_top", wire break, or measuring range exceeded.

2.4.2 Basics of system diagnostics (S7-1500)

You can use the system diagnostics to analyze errors in the system and generate alarms with a written error description and an indication of the error location.

These alarms are defined for each component with alarm capability (e.g., channel fault, rack error).

Recommended procedure

Make the settings for the system diagnostics and the structure of the alarms.

You will find detailed information in the sections below.

2.4.3 Displaying settings of system diagnostics (S7-1500)

Requirement

You are in the "Common data" folder in the project tree.

Procedure

To display and edit the settings for system diagnostics, follow these steps:

1. Double-click the "System diagnostics settings" entry.
2. You can view and edit the settings in the system diagnostics.

2.4.4 Settings (S7-1500)

2.4.4.1 General settings (S7-1500)

Activate system diagnostics for this PLC

This option is selected by default and cannot be cleared.

Note

When this option is cleared, diagnostics data is also not read out in the Web server and diagnostics viewer, even if the option is selected there.

If you clear this option on a PLC on which it was previously selected, all the diagnostics data is deleted the next time you compile the hardware configuration.

Signaling network errors not as errors but as maintenance demanded

When you select this option, network errors are signaled as 'Maintenance demanded'.

2.4.4.2 Alarm settings (S7-1500)

For the alarm categories listed below, you can define whether an alarm is output and whether or not this alarm requires acknowledgment:

- Error
- Maintenance demanded
- Maintenance required
- Info

To do this, select the relevant options in the "Alarm" and "Acknowledgment" columns.

The set alarm class is displayed in the "Alarm class" column.

2.4.5 Exporting and importing system diagnostic settings (S7-1500)

2.4.5.1 Basics for exporting and importing system diagnostic settings (S7-1500)

Description

You can export the system diagnostic settings of a project and import them into another project. The advantage of importing these settings is that they do not have to be created separately for each project.

The system diagnostic settings are saved in a data file (*.dat).

2.4.5.2 Exporting system diagnostic settings (S7-1500)

Requirement

A project is open.

Procedure

To export the system diagnostic settings, follow these steps:

1. In the project tree, click on the "Shared data" folder and double-click "System diagnostic settings".
2. In the system diagnostics, click on the "Exporting system diagnostic settings" icon in the toolbar. The "Save as" dialog opens.
3. Select the path to which you want to save the export file.
4. Click the "Save" button.

Result

A .dat file is created at the selected memory location.

2.4.5.3 Importing system diagnostic settings (S7-1500)

Requirement

The project into which you want to import the system diagnostic settings is open.

Procedure

To import the system diagnostic settings, follow these steps:

1. In the project tree, click on the "Shared data" folder and double-click "System diagnostic settings".
2. In the system diagnostics, click on the "Importing system diagnostic settings" icon in the toolbar. The "Open" dialog opens.
3. Select the path where the file to be imported is stored.
4. Select the required .dat file.
5. Click the "Open" button.

Note

The "Import" icon is grayed out when the open project is read-only (e.g. reference object).

If an error occurs during the import, an alarm is displayed and the process is aborted.

Result

The system diagnostic settings are displayed.

2.5 Using additional diagnostics options (S7-300, S7-400)

2.5.1 Hardware diagnostics (S7-300, S7-400)

2.5.1.1 Displaying non-editable and current values of assignable module properties (S7-300, S7-400)

Showing general properties and system-relevant information for a module (S7-300, S7-400)

Where do I find the information I need?

The general properties and system-relevant information for a module can be found in the "General" group in the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.

Structure of the "General" group

The "General" group consists of the following areas:

- Module
- Module information
- Manufacturer information

"Module" area

This area displays the following additional module information for S7-300 and S7-400:

- Version of bootloader

"Module information" area

This area shows the following additional module information for S7-300 and S7-400 that you have specified during the hardware configuration:

- Device name (you specified this in the network view).
- Plant designation
  If a CPU, a slave, an I/O device, or a module in the device supports the assignment of a plant-wide unique plant designation, then this is shown here. (The plant designation clearly identifies parts of the plant according to functional aspects. It is hierarchically structured in accordance with IEC 1346-1). This allows, for instance, the CPU to be identified from the user program. You can read out the plant designation with the "RDSYSST" instruction.
- Location designation
  If a CPU, a slave, an IO device, or a module in the device supports the assignment of a location designation, then this is shown here (the location designation is part of the equipment identifier that identifies the precise location, e.g., a process tag within a process manufacturing plant, similar to a road in a road map).

"Manufacturer information" area

This area displays the following additional module information for S7-300 and S7-400:

- Copyright law entry

Showing the communications properties of a module (S7-300, S7-400)

Where do I find the information I need?

The communications properties of a module can be found in the "Communication" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.

This shows the following properties:

- Cycle load due to communication, configured:
  Set value of the "Cycle loading by communications" parameter, i.e., the proportion of the CPU processing power available for communication in each cycle.
- Connection resources, maximum number:
  Specifies the maximum possible number of available connection resources of the module. A certain number of the available connection resources are reserved for programming device and OP communications and on the S7-300 (with the exception of the 318-2 DP CPU) for S7 basic communications. You can however establish other programming device and OP connections in addition; the number of unassigned connection resources is then reduced accordingly.

Show interfaces and interface properties of a module (S7-300, S7-400)

Where do I find the information I need?

The interfaces and interface properties of a module can be found in the following groups of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed:

- MPI interface
- MPI/DP interface
- DP interface
- PROFINET interface

"MPI interface" group

This group displays the following data:

- Transmission speed:
  Current transmission rate setting at the MPI interface for the automation system

"MPI/DP interface" group

This group displays the following data:

- Transmission speed:
  Current transmission rate setting for the MPI/DP interface of the automation system
- DP mode:
  Compatibility mode of the MPI/DP interface whose interface type is set to "PROFIBUS". In the case of modules that support the "DPV1" mode, these can assume the values "S7 compatible" and "DPV1".

  Note

  If you have set the interface type of the MPI/DP interface to "MPI", then "DP mode" does not contain a valid value.

"DP interface" group

This group displays the following data:

- DP mode:
  Compatibility mode of the DP interface. In the case of modules that support the "DPV1" mode, these can assume the values "S7 compatible" and "DPV1".

"PROFINET interface" group

This group is divided into the following areas:

- "Ethernet address" with the "Network connection" and "IP Parameters" subareas
- "Ports"
- "Statistics" (this area is not considered here, because it does not deal with the current values of configurable module properties).

Note

The "Ethernet address" and "Ports" areas have already been described outside the chapter "Using additional diagnostics options" as they apply for S7-1200 and S7-300/S7-400.

Showing the operand areas, organization blocks and system blocks of a CPU (S7-300, S7-400)

Where do I find the information I need?

The operand areas, organization blocks, and system blocks of a CPU can be found in the "Performance data" group in the "Diagnostics" folder of the Online and Diagnostics view of the module to be diagnosed.

"Performance data" group

The following information for the respective CPU is shown in the "Performance data" group:

- Operand areas:
  The table gives information regarding
  - The operand type, for example process image input, counters
  - The number of operands or size of their areas
  - The available address range of the operands, for example I0.0 to I127.7
- Organization blocks:
  All organization blocks (OBs) including their function
  You can sort the table rows as follows: Click a column heading to arrange the table rows in such a way that the contents of this column are ordered alphabetically starting from the top.
- System blocks:
  All system functions (SFCs) and system function blocks (SFBs)
  You can sort the table rows as follows: Click a column heading to arrange the table rows in such a way that the contents of this column are ordered alphabetically starting from the top.

Displaying properties of the time system of a module (S7-300, S7-400)

Where do I find the functions I need?

You can find the properties of the time system of a module in the "Set time of day" group in the "Functions" folder of the Online and Diagnostics view.

Structure of the "Set time of day" group

The "Set time of day" group consists of the following areas:

- Area for reading out and setting the time of day (this area is not examined here)
- Time system

Structure of the "Time system" area

The "Time system" area consists of the following subareas:

- "Clock"
- "Synchronization"

"Clock" subarea

The "Clock" subarea consists of the following elements:

- Resolution:
  States the precision with which the calculation and output of the time is performed
- Real-time clock:
  Displays whether the CPU has a real-time clock: "present" if yes; "not present" if no
- Correction factor:
  Assigned correction factor for the realtime clock of the module. The daily deviation of the clock is compensated with the correction factor.

"Synchronization" subarea

Synchronization mechanisms allow creation of synchronicity between the clocks in different sub-systems. The settings for time-of-day synchronization are made when assigning the module parameters.

The integral real-time clock - where present - can thus synchronize the clocks of other modules (master), or it can itself be synchronized by the clocks of other devices (slave), or it can take no part in the synchronization. Synchronization can be performed in the AS or on the MPI.

2.5.1.2 Displaying current values of dynamic module properties (S7-300, S7-400)

Displaying current status of the LEDs and the mode selector of a CPU (S7-300, S7-400)

Where do I find the information I need?

The current status of the LEDs of the mode selector of a CPU can be found in the display area of the "CPU control panel" pane of the "Online tools" task card.

Display area of the "CPU control panel" pane of the "Online Tools" task card

This area contains the following displays:

- Device name and CPU type
- Error (corresponds to the "SF" LED on S7-300 CPUs)
- RUN (corresponds to the "RUN" CPU operating mode)
- STOP (corresponds to the "STOP" CPU operating mode)
- FORCE (corresponds to the "FRCE" LED on the CPU)
- Current position of the mode selector

Note

"RUN-P" may be displayed erroneously for S7-300 CPUs with a mode selector. In this case, the mode selector is set to "RUN".

Showing connection resources and occupied and reserved connections for a CPU (S7-300, S7-400)

Where do I find the information I need?

The connection resources not yet assigned and the assigned and reserved connections for a CPU can be found in the "Communications" group in the "Diagnostics" folder in the Online and Diagnostics view.

The "Connection resources" area contains the current values of the following dynamic module properties.

- Number of connection resources not yet assigned.
  These are shown under "Not assigned".

  Note

  If connection resources are already reserved for certain types of communication, the unassigned resources cannot always be used for the various connection types.

- PG communication "reserved" and "assigned":
  Number of connection resources reserved/assigned for connections between the module and programming devices.

  Note

  The number of connection resources assigned can be larger than the number of connection resources reserved, since - within the overall maximum number - other free connection resources can be used for programming device communication. On S7-300-CPUs with assignable connection resource parameters, you can increase the number of connection resources for programming device communication at the expense of the S7 basic communication.

- OP communication "reserved" and "assigned":
  Number of connection resources reserved/assigned for connections between the module and HMI devices (TDs, OPs).

  Note

  The number of connection resources assigned can be larger than the number of connection resources reserved, since - within the overall maximum number - other free connection resources can be used for OP communications. On S7-300-CPUs with assignable connection resource parameters, you can increase the number of connection resources for OP communication at the expense of the S7 basic communication.

- S7 communication "reserved" and "assigned":
  This is displayed on S7-400 CPUs, on 318-2 CPUs and on 317-2 PN/DP CPUs. Reserved/assigned connection resources for configured S7 connections intended for exchanging data by calling communications function blocks (SFBs) in the user program. These can be regarded as hard wired connections (for instance CPU-CPU connections). They can be automatically established when starting communications or they can be established and broken off under program control. S7 communication can be performed via MPI, PROFIBUS, and Industrial Ethernet.

- S7 basic communication "reserved" and "assigned":
  This is displayed on S7-300 CPUs (but not on 318-2 CPUs and not on 317-2 PN/DP CPUs). Reserved/assigned connection resources for non-configured connections intended for exchanging data by calling communications functions (SFCs) in the user program. S7 basic communication can only be performed via MPI or within the automation system.
- Other communication "assigned":
  Additional connection resources for which no connection resources are reserved. Examples:
  - Communication by a S7-300 CPU using a CP (can be performed only on CPUs with assignable connection resource parameters)
  - Communication between networks (S7 routing) on S7-400 CPUs

Note

By means of a server function, in the case of assigned connection resources for S7 communication (i.e., if an S7-400 CPU is communicating with an S7-300 CPU via the SFCs "PUT" or "GET") the following resources are displayed for S7-300 CPUs:

- If the CPU has no assignable connection resource parameters: Under "S7 basic communication"
- If the CPU has assignable connection resource parameters: In the case of CPUs 317-2 PN / DP and 318-2 under "S7 communication", for all others under "Other communication"

Displaying statistical data of data transmissions made using the Ethernet interface (S7-300, S7-400)

Where do I find the information I need?

Statistical data for the data transmissions that took place via the Ethernet interface can be found in the "Statistics" area under the "PROFINET interface" group in the "Diagnostics" folder of the Online and Diagnostics view.

"Statistics" area

- Display
  Displays statistical data for sent and received data packets. This data allows the data transmission quality to be evaluated.
  The start of the recordings of the statistical data is shown in the first line ("Data packets since:"). This time is reset every time the statistical data is reset ("Reset" button).
  The statistical data display is updated continuously.
- "Reset" button
  The "Data packets since" time is reset and the statistical data are reset to "0".

Displaying information about the runtime meter of a CPU (S7-300, S7-400)

Where do I find the information I need?

Information about the runtime meter can be found in the "Runtime meter" group in the "Diagnostics" folder of the Online and Diagnostics view.

"Runtime meter" group

The "Runtime meter" group displays the following information for the active CPU runtime meter:

- No. of the runtime meter including information about whether it is a 16-bit or 32-bit runtime meter.
- Elapsed operating hours
- Status
- Information on whether or not an overflow has occurred

If the CPU has one or more 32-bit runtime meters, the respective symbol is given a red border when the metered value exceeds 32767 (maximum value for a 16-bit runtime meter). The "Overrun" column shows "no" if the runtime meter has not yet exceeded the maximum value for a 32-bit runtime meter.

The 32-bit runtime meter is set, started, stopped, and read out by the "RTM" instruction.

2.5.1.3 Checking a module for defects (S7-300, S7-400)

Performing channel-specific diagnostics of non-CPU modules (S7-300, S7-400)

Where are channel-specific diagnostic events of non-CPU modules displayed?

The channel-specific diagnostic events of non-CPU modules are displayed in the "Channel diagnostics" group in the Online and Diagnostics view.

"Channel diagnostics" group

The display consists of the following elements for each event:

- Slot of faulty module
- Number of faulty channel
  Note that the sequence of channel numbers in this display starts at 0.
- Fault type

Note

Diagnostic interrupts

A diagnostic interrupt can be reported to the CPU only if the module has diagnostic interrupt capability and the diagnostic interrupt has been enabled.

The display of the diagnostic interrupt is a snapshot. Sporadic module defects can be identified in the diagnostics buffer of the respective CPU.

If you select a line in the table containing the channel errors, additional information will be provided for this error (if present).

Performing communication diagnostics of PROFINET ports (S7-300, S7-400)

Where are port-specific errors of PROFINET ports displayed?

The port-specific errors of PROFINET ports are displayed in the "Communication diagnostics" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.

Structure of the "Communication diagnostics" group

The "Communication diagnostics" group consists of the following areas:

- Table containing the port-specific errors
- "Details of the event" text field
- "Help on event" text field

Structure of the table containing the port-specific errors

Each line of the error table corresponds to one error.

The table consists of the following columns:

- Name: port no. and port designation, as well as diagnostics icon
- Error: description of the error that occurred on this port

"Details" text field

If you select a line in the error table, the "Detail" text field will contain detailed information (if available) on the corresponding error.

"Help on selected diagnostic line" text field

If you select a line in the error table, the "Help on selected diagnostic line" text field will contain help information (if available) on the corresponding error.

PROFINET IO component diagnostics (S7-300, S7-400)

Where do I perform PROFINET IO component diagnostics?

You perform the PROFINET IO component diagnostics in the "PROFINET IO diagnostics" group under "PROFINET interface" in the "Diagnostics" folder of the Online and Diagnostics view.

"PROFINET IO diagnostics" group

Manufacturer-related diagnostics texts for the IO device or the selected module of the IO device are displayed in this area.

If the diagnostics can be read by the IO controller, then communication errors and configuration errors (online/offline discrepancies) are displayed here.

Starting NCM S7 diagnostics (S7-300, S7-400)

Requirements

- An online connection to the module is not necessary.

Procedure

To start the NCM S7 diagnostics function, follow these steps:

1. Open the module in the Online and Diagnostics view.
2. Select the "Special diagnostics" group in the "Functions" folder.
3. Click the "Start special diagnostics" button in the "Special diagnostics" area.

Result

The NCM S7 diagnostics will be started.

Online connections for special diagnostics and firmware loader (S7-300, S7-400)

Special diagnostics: Establishing an online connection

Some network components (S7-300/400 CPs, PC CPs, gateways) provide expanded diagnostics data in special diagnostics.

Requirement:

A PG/PC interface is physically connected to the destination device, e.g. via an Ethernet cable.

Procedure:

Follow the steps outlined below to start special diagnostics:

1. Open the online and diagnostics view of the module.
2. In the "Functions" folder, select the group "Special diagnostics",
3. In the "Special diagnostics" area, click the "Start special diagnostics" button.

As of STEP 7 Professional V14 for several CPs only one online connection can be established from the engineering station via the CP. This connection can be established either with STEP 7 or special diagnostics.

If special diagnostics is not started by the steps listed above and an error message is output follow the steps below:

1. Disconnect the existing connection to the station.
2. Open the online and diagnostics view of the module again as described above and start special diagnostics.

32-bit applications; NCM S7 diagnostics, firmware loader

If you have established a connection to a PC CP from a 32-bit application and want to establish another online connection via the same interface of the PC CP, you must first terminate the connection of the 32-bit application.

Special diagnostics and firmware loader Chinese GUI

When you open the special diagnostics or the firmware loader with a Chinese GUI, you need to set the language for non-Unicode applications in the operating system to Chinese.

2.5.1.4 Changing the properties of a module or the programming device/PC (S7-300, S7-400)

Special feature for S7-400 CPU operating mode switchover (S7-300, S7-400)

Specifying the startup type of an S7-400 CPU for switchover to "RUN" mode

If you want to switch an S7-400 CPU to "RUN" mode, choose one of the following startup types:

- Warm restart
- Cold restart
- Hot restart

Loging onto the CPU for alarms and system diagnostics events (S7-300, S7-400)

Requirement

- The PG/PC interface allows access to one or more devices in a subnet.

Procedure

1. Open the Online and Diagnostics view of the respective CPU.
2. Select the "Online access" group.
3. Select the "Receive alarms" check box in the "Alarms" area.

Result

During the next online connection, your programming device or PC will be logged on for receiving alarms and system diagnostics events from the relevant CPU.

Note

The Alarm view requires a continuous online connection for every CPU from which alarms are to be received.

Logging on when an online connection already exists

If the online connection to a CPU has already been established, you can still log on or off as follows for receipt of alarms:

- In the "Online" menu via the "Receive alarms" command
- In project tree using the CPU shortcut menu
- In the device view of the hardware configuration using the CPU shortcut menu
- In the network view of the hardware configuration using the CPU shortcut menu

See also

Receiving alarms (Page 3045)

Compressing the memory and copying from RAM to ROM (S7-300, S7-400)

Where do I find the information I need?

The reorganization of the memory and copying from RAM to ROM can be found in the command area of the "Memory" pane in the "Online Tools" task card.

The reorganization of the memory can also be found in the command area of the "Memory" group in the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.

Command area of the "Memory" pane on the "Online Tools" task card

This area can contain the following buttons:

- Compress
  Pressing this button reorganizes the memory, to increase the size of the largest continuous free memory area.
- Copy RAM to ROM
  Pressing this button copies the content of the RAM load memory (user program) to a FEPROM:
  - To the memory card inserted into the CPU.
  - To the integral FEPROM.

Note

A button is specifically displayed when the module commands the respective function.

Command area of the "Memory" group in the "Diagnostics" folder of the Online and Diagnostics view

This area contains the following buttons:

- Compress
  Pressing this button reorganizes the memory, to increase the size of the largest continuous free memory area.

Resetting the PROFINET interface parameters of an S7-300 or S7-400 CPU (S7-300, S7-400)

Requirement

- You are using an S7-300 CPU with any firmware version or an S7-400 CPU with firmware version >= V6.0.
- There is no memory card inserted in the CPU.
- There is an online connection to the CPU whose PROFINET interface parameters you want to reset.
- The CPU is in STOP mode.

  Note

  If the CPU is still in RUN mode and you start the reset operation, you can place it in STOP mode by answering the security prompt with yes.

Procedure

To reset the PROFINET interface parameters of an S7-300 or S7-400 CPU, follow these steps:

1. Open the Online and Diagnostics view of the CPU.
2. Select the "Reset of PROFINET interface parameters" group in the "Functions" folder.

   Note

   The user interface of the "Reset of PROFINET interface parameters" group of the Online and Diagnostics view depends on how you have opened the Online and Diagnostics view:
   - If you opened it from the project tree with "Update accessible devices", the MAC address of the associated module is known and grayed out.
   - If you opened it from within the project context, you must enter the relevant MAC address. Otherwise, the "Reset" button remains grayed out.

3. Click the "Reset" button.
4. Acknowledge the confirmation prompt with "OK".

Result

The module is switched to STOP mode, if necessary, and the PROFINET interface parameters are reset. This means:

- The Ethernet interface settings are reset.
- The IP address is deleted.

Starting the firmware loader for an S7-300 or S7-400 CP (S7-300, S7-400)

Requirements

- The CP to be updated supports a firmware update.
- An online connection to the CP is not necessary.

Procedure

To start the firmware loader for an S7-300 or S7-400 CP, follow these steps:

1. Open the CP in the Online and Diagnostics view.
2. Select the "Firmware update" group in the "Functions" folder.
3. Click the "Start firmware loader" button in the "Firmware update" area.

Result

The firmware loader will be started.

Note

Updating the firmware with S7-300 and S7-400 CPs using the firmware loader differs from the standard mechanisms for updating firmware.

Updating the firmware of a TIM 3V / TIM 4R (S7-300, S7-400)

Validity

The information below applies to the following TIM modules for SIMATIC S7-300 / 400:

- ST7-TIM
  - TIM 3V-IE
  - TIM 3V-IE Advanced
  - TIM 4R-IE
  - TIM 4R-IE Standalone
- DNP3-TIM
  - TIM 3V-IE DNP3
  - TIM 4R-IE DNP3

Sequence of the update

Different files are provided for the firmware update of these TIM modules:

- Firmware files
- Driver package

Note

Sequence of the update

Adhere to the correct update sequence for a firmware update:

1. **First update the driver package.(*.Z).**
2. **Then update the firmware files.(*.FWL).**

If you update the firmware files first and inadvertently select the wrong file, the TIM module may no longer be accessible in certain circumstances.

Files for the firmware update

The files for the firmware update are provided as compressed file archives. The archives contain files for the driver package and the firmware as well as readme files, which include information about the firmware version.

One archive is made available for each of the following TIM modules:

- Archive for ST7-TIM 3V..
  - TIM 3V-IE
  - TIM 3V-IE Advanced
- Archive for ST7-TIM 4R..
  - TIM 4R-IE
  - TIM 4R-IE Standalone
- Archive for TIM 3V-IE DNP3
- Archive for TIM 4R-IE DNP3

To save a file archive, follow these steps:

1. Copy the archive to the file system of your engineering station.
   Specify the directory in the online dialog for the update process.
   Recommendation:
   Copy the archive for the firmware versions of various module types in a separate directory.
2. Unzip the archive after you have copied it into the file system of your engineering station.

See also

Online connections for special diagnostics and firmware loader (Page 3392)

Assigning an IP address to a PROFINET IO device (S7-300, S7-400)

Basic information on assigning an IP address to a PROFINET IO device (S7-300, S7-400)

Overview

You have two basic options for choosing an IP address:

- You assign the IP address yourself.
- You obtain the IP address from a DHCP server. (This option is not available for all IO devices.) In this case, the client ID, the MAC address, or the device name of the IO device is communicated to the DHCP server, depending on the option selected.

Requirement

The following assumes that the IP address of the IO device can also be obtained from a DHCP server. This is the case, for example, for some S7-300 CPs.

Starting the address assignment via "Accessible devices" (S7-300, S7-400)

Requirement

- The devices accessible via the associated interface of the PG/PC are displayed in the project tree (to display these, either double-click "Update accessible devices" in the project tree or select the "Accessible devices..." command in the "Online" menu.).
- You have double-clicked "Online access" -> <Selected interface> -> <PROFINET IO device> -> "Online & Diagnostics" in the project tree to open the Online and Diagnostics view.

Procedure for specifying the IP address yourself

To assign the IP address you have specified yourself to the IO device, follow these steps:

1. Open the Online and Diagnostics view of the IO device.
2. Select the "Assign IP address" group in the "Functions" folder.
3. Select the "Use IP parameters" option.
4. Enter the desired IP address.
5. Enter the subnet mask.
6. If a router is to be used, select the "Use router" check box and enter its IP address.
7. Click the "Assign IP address" button.

Procedure for obtaining the IP address from a DHCP server

To assign an IP address specified by a DHCP server to the IO device, follow these steps:

1. Open the Online and Diagnostics view of the IO device.
2. Select the "Assign IP address" group in the "Functions" folder.
3. Select the "Obtain IP address from a DHCP server" option.
4. If the DHCP server is to identify the target device using its client ID, choose the "Client ID" option. Enter the client ID in the entry field of the same name.

   Note

   The client ID is a string with a maximum of 63 characters. Only the following characters may be used: a-z, A-Z, 0-9 and - (hyphen)

5. Alternatively: If the DHCP server is to identify the target device using its MAC address, choose the "MAC address" option

6. Alternatively: If the DHCP server is to identify the target device using its device name, choose the "Device name" option.

   Note

   If you specify that the DHCP server is to identify the target device using its device name, you must have assigned a device name to the target device beforehand.

7. Click the "Assign IP address" button.

Result

The IP address is permanently assigned to the IO device. It is retained even through a startup or a power failure.

Starting the address assignment from the project context (S7-300, S7-400)

Requirement

- An online connection to the PROFINET IO device exists.
- You have opened the Online and Diagnostics view of the PROFINET IO device from the project context.

Procedure

1. Open the "Functions" folder and the "Assign IP address" group inside this folder. The "Use IP parameters" option is selected.
2. Click the "Accessible devices" button in order to identify the devices that can be accessed via MAC addresses. After the search is complete, select the IO device with the MAC address known to you.
3. Click the "Assign IP address" button.

Result

The IP address is permanently assigned to the IO device. It is retained even through a startup or a power failure.

2.5.2 Connection diagnostics (S7-300, S7-400)

2.5.2.1 Detailed connection diagnostics (S7-300, S7-400)

Determining the connection details of an NCM connection (S7-300, S7-400)

Where do I determine the connection details of an NCM connection?

The connection details of an NCM connection can be found in the "Connection details" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window.

When is the "Connection details" group filled in?

The following requirements must be met to fill in the "Connection details" group on the "Connection information" tab:

- There is an online connection to the relevant module.
- You have selected a line in the connection table.

Structure of the "Connection details" group for an NCM connection

In case of an NCM connection, the "Connection details" group consists of the following elements:

- Local connection ID (hex)
- Connection name
- Connection status
- Send status
- Receive status
- Number of bytes sent
- Number of bytes received
- "Special diagnostics" button
  Click this button to start the NCM connection diagnostics.

See also

Starting NCM S7 diagnostics (Page 3391)

Index

"

"IP access protection" tab, 3064
"IP configuration" tab, 3051
"OPC properties" tab, 268
"Port parameters" tab, 3061
"Target computer configuration" dialog box, 1495
"WWW" instruction, 1279, 1404

*

*.cer, 471
*.dat, 471
*.p12, 471

:

:P, 918, 1304

3

32-bit down counter, 3200
3964(R)
    Receiving data, 1559
    Sending data, 1558
3964(R) procedure, 1557, 1600
    Block check character, 1602
    Control characters, 1557, 1601
    Priority, 1557, 1601
    receiving data, 1606
    Sending Data, 1603
3964(R) protocol (CPU 31xC), 802
3964R procedure
    Block check character, 1558

A

Ability to read back connection parameters, 122
Accelerating startup (PROFINET), 2940
Access
    direct ~ to the I/O, 813
Access control, 2209
    Automatic learning, 2209
Access level, 970, 975
Access point
    Overlapping channels, 2387
    Overview, 2382
Access protection, 981
    Parameter assignment, 1466
Accessible devices, 2876
Accompanying signals, 1529
Accumulators, 812
ACL, 2209
ACT_TINT, 750
Activating and deactivating IO devices, 2937
Active bus module (ET 200M), 3284
Active nodes, 451
Additional field devices (PROFIBUS and PROFINET), 2778, 2834, 2886, 2887, 2925
Address, (see Module start address)
Address assignment, 1096
    User data with DPV1, 2787
Address assignment (value status), 920
Address assignment rules, 1096
Address of the gateway, 554, 2558
Address overview, 62, 1105
Address packing, 2748, 2824
Address parameter
    Broadcast, 259
Address range, 402
    Changing, 1098, 1378
Addresses, 3103
    Assigning, 1110, 1380
    DP interfaces, 2786
    DP slave, 2788
    Interrupt with packed, 2754, 2830
    Interrupts on the I-slave, 2759
    Pack, 2748, 2824
    System information, 2787
    Unpack, 2749, 2824
    User data with DPV1, 2786
Addressing, 918, 1304
    Changing, 1098, 1378
    General, 1096, 1097, 1377
Advanced Encryption Algorithm, 306
Advanced mode
    Global firewall rules, 372
Advanced offline/online comparison
    Automatic device assignment, 302
AES, 306
AES-128, 433
Aging time, 2213, 2216

Alarm display, 3042
"Active alarms" view, 3043
Acknowledging an alarm, 3046
Archive view, 3043
Clear alarm archive completely, 3044
Export archive, 3044
Layout of the alarms in the "Active alarms" view, 3043
Layout of the alarms in the archive view, 3044
Receiving alarms, 3045
Using the keyboard, 3048
Alarm events, 610, 2624
Alarm settings, 935
Alarms
Logging on for receiving from CPU, 3393
Alternative partners (IO devices changing during operation), 2938
Analog module
Resetting to factory settings, 3350
Analyzing
Diagnostic information with DPV1, 2784
Interrupt information with DPV1, 2783
Applicant, 308
Application, 3026
Application cycle, 2912
Application options, 1626
Application options of the communication module, 1628
Area of unplugged modules, 19
ARP, 521
Diagnostics, 521
Table, 521
Array
Web server, 1273, 1275
Article number, 29, 580, 2025, 2362, 2585
AS interface, 2837
ASCII code table, 125
ASCII driver, 1590
Receive buffer, 1596
receiving data, 1592, 1625
RS232C accompanying signals, 1597, 1622
Sending Data, 1590
ASCII protocol, 1549
ASCII protocol (CPU 31xC), 795
ASCII TSAP, 125
Asset management, 3014
Assigning access rights to specific OPC items, 1502, 1510
Assigning an IP address
Basics, 3341, 3397
from the project context, 3342, 3399
Using "Accessible devices", 3341, 3398
Assigning connection path, 206
Assigning interfaces, 1472, 1474
Reference lines, 1472
Assigning symbol, 1110, 1380
Assigning tag, 1110, 1380
Assignment list, 911, 912
Asymmetric encryption, 306
Asynchronous data transmission, 1532
Asynchronous error
Event ID, 879
Settings, 749
ATTACH, 1311
Attacks, 979, 1345
Audit log, 467
Authentication, 615, 2112, 2300, 2437, 2628
Authentication methods, 437, 439
Automatic commissioning, 2876
Automatic mode of the PROFINET IO controller, 1482
Automatic operation of accompanying signals, 1546
Automatic update, 1107
Automatically determining connection path, 212
Autonegotiation, 2875
Available system functions, 551, 1961, 2315, 2320, 2323, 2552
AWP command, 1261, 1264, 1391, 1393
AWP_In_Variable, 1395
AWP_Out_Variable, 1265, 1394

B

Background OB (OB 90) (S7-300, S7-400), 866
Backup, 728
Bandwidth (IRT communication), 2902
BA-Send, 3228
BA-Send 1xFC, 3228
Basic submodule, 920
BCC, 1557
Bias time Ti, 2795
Bidirectional data traffic, 1532
Bidirectional multicast, 2011
Bit memory, 913
Block check character, 1558, 1602
Block parameters
Basics, 1371
Block stack, 727, 734
Bootstrap router, 2011
Boundaries, 2876
Bridge, 591, 659, 2602, 2679
Bridge priority, 591, 659, 2602, 2679
Root bridge, 591, 659, 2602, 2679
Bridge Max Age, 591, 2603

Bridge Max Hop Count, 592, 2603
Bridge priority, 2354
Broadcast, 228, 270, 403, 411, 1563, 1637, 2217
    With UDP, 258
Brute force, 979, 1345
Brute-force attack, 979, 1345
Brute-force attacks, 979, 1345
BSR border, 2011
Bus cycles, 2790
BUSY signal, 1625
button, 623, 2636

C

CA certificate, 354, 359, 360, 568, 2573
CA group certificate, 360
Cabling rules (PROFINET), 2875
Calibrating, 3347
    Overview, 3346
Call structure, 911, 912
Calling FAQs for a module, 34
Calling manuals for a module, 34
Canceling a calibration, 3349
Card type, (See Memory card)
Cascaded counting function, 3200
Catalog, (see Hardware catalog)
Central racks
    Selecting from the hardware catalog, 1088
CER, 327
Certificate, 355, 438
    Assign, 360
    Exporting, 355
    Importing, 355
    Renewing, 358
    Replace, 359
    Replacing, 359
    self-signed, 358
    signed by certificate authority, 358
Certificate authorities, 308
Certificate authority, 354, 356
Certificate holder, 308
Certificate manager, 355, 982
Certificate manager (global), 324
Certificate manager (local), 983
Certificates, 687, 2707
    Assigning, 987
    Creating, 985
    Deleting (global), 330
    Deleting (local), 988
    Exporting (global), 326
    Exporting (local), 987
    for OPC UA server, 990
    for Web server, 989
    Importing, 326
    Renewing, 329
    Replacing, 329
    Self-signed, 985
    Show, 330
Changeover contact, 3285
Changeover contact sensor type, 3285
Changing a port interconnection
    Graphic view, 296
Changing connection partners, 223
Changing connections interactively, 224
Changing from DPV0 to DPV1, 2782
Changing partner port, 2936
Changing partner ports during operation, 2936
Changing the connection path in the Inspector window, 225
Changing the path of a configured connection, 224
Channel address, (see Module start address)
Channel diagnostics, 3390
Channel-specific diagnostics, 3389
Character delay time, 1552
Character delay time CDT, 1563, 1638
Character set
    in message texts, 1622
Class of Service, 2157
Client
    Overview, 2388
Client Supplicant, 2526
Clipping, 3249
Clock function, 756, 968, 969
    Clock parameters, 755
    Reading the time, 967
    Setting the time, 967
Clock memory, 756, 935, 970, 1314, 1338
Clock parameters, 756, 968, 969
Code block, 911
Code transparency, 1554
Cold restart, 710
Cold restart (S7-300, S7-400), 867
Color
    Diagnostics of Ethernet cables, 3311
    Diagnostics of ports, 3311
Command message frame, 1610
Communication, 978, 1344, 1474
    Cycle load, 945
    Interfaces, 1474
    Number of connections (PROFINET/PROFIBUS), 167
    Open User Communication, 205
Communication connection, 186
Communication diagnostics, 3326, 3390

Communication events, 887
Communication load, 935, 945, 1314
Communication properties of a module, 3382
Communication via PUT/GET instructions
Basic information on PUT/GET instructions, 150
Creating and assigning parameters to a connection, 155
Deleting an interconnection, 156
Overview of connection configuration, 151
Requirements, 150
Starting connection parameter assignment, 154
Communications
Connection resources, 759
Cycle load, 1321
Communications error OB (OB 87) (S7-300, S7-400), 862
Communications load, 1321
Communications modules, 1330
Properties, 1330
Communications options, 2525
Communications port
Configuring, 1330
Communications processor
SIMATIC NET PC software, 1479
STEP 7 V5 + STEP 7 Professional, 1489
Communications protocol
Defining, 1333
Overview, 1333
Compare editor
Device comparison, 67
Compare offline/online
Automatic device assignment, 300
Compatibility
S7-1200 CPU firmware versions, 1375
Compressing
Memory in the Online and Diagnostics view, 3393
Computer connection RK512, 1610
Command message frame, 1610, 1611
Fetching data, 1616
Response message frame, 1610
Sending Data, 1613
Configuration
Distributed I/O, 2728
DP slave, 2729, 2742, 2818
DP slave, simple, 2731, 2809
Hardware, 40
I-slave, 2733
Loading to device, 3030
Loading to PG/PC, 3031
Modules, 1077
PC communication interface, 1467
PPP, 653
Software Controller, 1467
Configuration control, 1102, 1112, 1120, 1416, 1425, 2982, 3215, 3220, 3262, 3272, 3294
Configuration control for IO systems, 1124, 2982, 2992, 3005
Configuration control with ET 200pro, 3294
Configuration control with ET 200AL, 3262
Configuration control with ET 200MP, 3272
Configuration control with ET 200SP, 3215
Configuration file, 1442, 1443
Configuration mode, 601, 2076, 2404, 2612
Configuration Package Backup, 2151
Configure a PROFIBUS subnet
Matching parameters to one another, 80
Setting bus parameters, 80
Configuring
CP as I-slave, 2762
CPU as I-slave, 2762
Diagnostics addresses on the I-device, 2923
Diagnostics addresses on the I-slave, 2758
Direct data exchange, 2770
DP slave as I-slave, 2763
HART variable, 3192
I-slave, 2756, 2820
PC station, offline, 1443
Configuring a CPU
Loading a project, 1439
Configuring a PROFIBUS subnet
Meaning of the bus parameters, 81
Configuring connections, 187
Dialog guided, 206, 215
Graphic, 206
Configuring internal subnets manually, 512, 548
Configuring IP network nodes manually, 512
Configuring MAC network nodes manually, 512
Configuring the network with Ethernet, 85
Creating private subnets, 87
Linking networks, 87
Relationship between IP address and subnet mask, 86
Setting the IP address, 86
Setting the subnet mask, 86
Connecting cables, 1536, 1538, 1542
Connection, 127, 137, 207
Address details, 3364
Configuring, 187
Configuring a connection when there is no or no clear network assignment, 142, 214
Configuring without networking, 218
Creating, 100, 206
Deleting, 103

Displaying assigned and reserved, 3386
E-mail, 187, 191, 193, 195
FDL, 187, 191, 193, 195
Influence of the networking - response when networking is missing, 208
ISO transport, 187, 191, 193, 195
ISO-on-TCP, 187, 191, 193, 195
Programmed, 204, 205
TCP, 187, 191, 193, 195
to a specific station in any project, 203
UDP, 187, 191, 193, 195
Unspecified, 207
without specifying a port, 203
Connection configuration
Connection parameters, 96
General, 92
Overview, 94
Starting, 99
Connection configuration with an OPC server, 202
Connection description
Changing parameter values, 122
Data block, 106, 109, 110, 112, 113, 114
Structure, 106, 109, 111, 112, 113, 115
Connection details, 3364
Connection diagnostics
Detailed, 3362
Overview, 3360
Connection display in the network view, 210
Connection information, 3362
Connection interrupted, 1806
Connection mechanisms, 978, 1344
Connection mode, 212
Connection parameter assignment of PUT/GET instructions, 154
Connection parameters of PUT/GET instructions, 153
Connection path
Highlighting in the network view, 219
Connection resource, 128, 138
Connection resources, 162, 3386
Data record routing, 175
Display in STEP 7, 177
Display in the Web server, 180
HMI communication, 175
Module-specific, 168, 178
occupying, 176
Online, 3362, 3363
Overview, 171
S7 routing, 175
Station-specific, 168, 178
Connection rules, 533, 546
Connection status
Displaying using icons, 3361
Connection table, 210, 222
Connections
Configuring graphically, 211
Number of connections (PROFINET/ PROFIBUS), 167
Consistency
Slot rules, 48
Consistency check, 508
local, 353
project-wide, 353
Consistency of data, 184
Constant bus cycle time, 2792
Activation in the PROFIBUS network, 2801
Composition of bus cycles, 2790
Definition, 2790
Requirements, 2791
Time for constant bus cycle time DP cycle, 2791
Context filter, 31
Continuation FETCH frame, 1618
Continuation message frame, 1610
Continuation SEND message frame, 1614
Control characters, 1601
in message texts, 1622
Control data record, 1112, 1416, 3220, 3262, 3294
S7-1500, 1122
Control data record 196, 3215, 3220, 3272, 3294
Converted name (PROFINET), 2847
Cookie, 1269, 1396
Copy
Hardware component, 54
CoS, 2157
Queue, 2157
CoS (Class of Service), 558, 1979, 2336, 2561
Count (8 DI NAMUR), 3198
Count (CPU 31xC), 763
Counter input, 1325
Counter mode, 1325
Counter, high-speed, 1324
CP
As I-slave, 2762
Point-to-point connection, 1577
With separate MPI address, 1096
C-PLUG, 633, 2453, 2639
Formatting, 634, 2141, 2455, 2641
Saving the configuration, 634, 2455, 2641
CPU
As I-slave, 2762
Assigned and reserved connections, 3386
Displaying the current LED status, 3322
Fill level of all types of memory, 3322, 3323

Inserting a signal board, 1382
Logging on for alarms, 3393
Not assigned connection resources, 3386
Number of communication connections, 167
Parameter, 741
Properties, 1313
Reading out a diagnostics buffer, 3327
Selecting from the hardware catalog, 44
Switching operating mode, 3332
CPU 1516-3 PN/DP, 955
CPU 1518-4 PN/DP, 955
CPU alarms - OPC server, 1521
CPU control panel
Display area, 3322, 3386
CPU display, (See display)
CPU firmware, 1375
CPU hardware fault OB (OB 84) (S7-300, S7-400), 850
CPU memory reset, 719
CPU properties, 935, 1314
CPU registers, 811
CPU RUN, 1576, 1701
CPU STOP, 1576, 1701
CPU warm restart, 1702
CRC, 1563, 1638
Creating a route, 503
CRL, 327
CRT, 327
CTS, 1536
Cu10 sensor, 3246
Cycle load, 945, 1321
Cycle load due to communication, 745
Cycle OB (S7-1500), 1000
Cycle time, 935, 944, 1314, 1320
display configured, 3315
display measured, 3321
Cyclic behavior, 744
Cyclic interrupt, 1361
OB 30 to OB 38 (S7-300, S7-400), 825
Settings, 751
Cyclic interrupt (S7-1500), 1008
Cyclic interrupt OB
Assigning parameters, 1372
Description, 1361
Parameter assignment, 1031
Cyclic interrupt OBs (S7-1500), 1008
Cyclic program (OB 1) (S7-300, S7-400), 819
Cyclic program (S7-1500), 1000
Cyclic program execution
Options for interrupting, 1354
Programming, 1354

D

D_ACT_DP, 1102
D_ACT_DP (activating and deactivating IO devices), 2937
Data bit (DBX), 1299
Data block, 911, 916
Data block 196, 3215, 3272, 3294
Data block 197, 3215, 3272, 3294
Data byte (DBB), 1299
Data consistency, 184
Data cycle, 997
Data double word (DBD), 1299
Data exchange
DP slave, 2731, 2809
I-slave - DP master, 2733, 2810
Data exchange broadcast, (See Direct Data Exchange)
Data flow control, 1332, 1529, 1535, 1543
Hardware, 1544
Software, 1544
Data log, 1387
Data transmission rates, 1528
Data transmission with the ET 200S Modbus master, 1648
Data type
Pip, 1109
Data word (DBW), 1299
DCD, 1536
DCP server, 600, 2076, 2404, 2611
Dead peer detection, 572, 2577
Dead peer detection (DPD), 455
Declaring ready for communication
- passive connection establishment, 203
Default domain, 2901
Default routes
IPv6 routes, 2410
Default settings, 1633
Defective devices, 3304
Delay time
Isochronous mode, 2800
Deleting
Hardware component, 59
Dependencies of rights, 365
DER, 327
DES, 433
Designated forwarder, 2011
Designated router, 2011
Details of connection establishment, 238, 244, 249
Determine diagnostics status, 3325
Determining time of day, 3334

Device
Adding to a hardware configuration, 41
Copying, 54
Defective devices, 3304
Deleting, 59
Moving, 56
Multiple selection, 51
Rename, 57
Renaming, 299
Device address, 1096, 1097, 1377
Device certificate, 568, 2573
Device certificates, 984
Device comparison, 67
Device information, 3304
Device name, 2840, 2847, 2991
Device name, automatic commissioning (PROFINET), 2843
Device names
PROFINET, 2890
Device number, 2842
PROFINET, 2890
Device overview
Address range, 1096, 1097, 1377
Device replacement, 57
Device Tool, 3028, 3029
Device view
Area of unplugged modules, 19
Edit parameters, 61
Edit properties, 61
Hardware and network editor, 14
Inserting a signal board, 1381
Inserting module, 48
Racks, 46, 1092
DHCP
Client, 640, 2098, 2429, 2652
Server configuration, 506
DHCP server, 507
DHCPv6
Client, 2107
PD Sub Client, 2108
Prefix delegation, 2107, 2108
DHCPv6 Relay Agent
General settings, 2274
Diagnose repeater, 3356
Diagnostic address, 1096, 1098
Diagnostic error interrupt, 1365
Diagnostic error interrupt (S7-1500), 1015
Diagnostic functions, 1577, 1702
Diagnostic information
Analyzing with DPV1, 2784
Diagnostic interrupt OB, 1365
Diagnostic interrupt OB (OB 82) (S7-300, S7-400), 843
Diagnostic interrupt OB (S7-1500), 1015
Diagnostics, 459, 1577, 1702, 3209
Channel-specific, 3389
I-slave, 2766
Online & diagnostics in TIA Portal, 1438
PC System V2.0, 1438
PROFINET, 2897
PROFINET IO components, 3327, 3391
Diagnostics address
Assigning, 2780
Configuring, 2758, 2923
I-device, 2921
I-slave, 2757
Diagnostics buffer, 979, 1244, 1344
Basics, 735, 1303, 3351
Organization, 735, 3351
Diagnostics data, 737
Diagnostics event
for modules, 888
Diagnostics status
Determining and displaying online, 3305
Dialog ", 1496
Digital certificates, 308
Direct access to the I/O, 813, 918, 1304
Direct communication, 1807
Direct data exchange
Configuring, 2770
DP slave - DP master, 2732
DP slave - I-slave, 2732, 2734
Principle of operation, 2768
PROFIBUS, 2768
Setting up, 2769
DIRECT_OPERATE, 1808
Display, 944
Display (CPU), 981
Display class, 758
Display classes for messages, 1257
Displaying alarms, 3042
Displaying memory types of a CPU, 3322, 3323
Displaying support for a module, 34
Distributed I/O, 2728, 2808, 3207, 3228, 3259
Configuring, 2728
Slot numbers, 2729
DLE, 1557
DM 370 Dummy, 1078
DNS client, 2081
DNS domain, 2082
Docking station, 2938
Docking system, 2938
Docking unit, 2936, 2938

Download, (See Loading to device)
Download firmware via Web, 3078
DP cycle
    Time for constant bus cycle time DP cycle, 2791
DP interface, 2735, 2812, 3383
    Addresses, 2786
    Changeover, 2739
DP master, 3194
    Add DP master system, 2739, 2815
    Devices and modules, 2735, 2811
    Disconnection from DP master system, 2738, 2815
    Display on DP slave, 2737, 2813
    DP interface, 2735, 2812
    DPV1 functionality, 2780
DP master system
    Add DP slave, 2740, 2816
    Creating, 2729, 2735, 2742, 2808, 2811, 2818
    Disconnect DP slave, 2741, 2817
    Disconnection from subnet, 2738, 2814
    Edit properties, 2739, 2815
    Highlight, 2737, 2814
    More than one, 2732
    Node disconnection, 2738, 2814
    Setting up, 2771
DP slave
    Add to DP master system, 2740, 2816
    Addresses, 2788
    As I-slave, 2763
    Assign DP master system, 2741, 2817
    Configuring, 2729, 2731, 2742, 2809
    Data exchange, 2731, 2809
    Disconnect from DP master system, 2741, 2817
    DP master display, 2737, 2813
    Hardware catalog, 2730, 2808
    Intelligent, (See I-slave)
    Networking, 2742, 2818
    Setting the isochronous mode, 2803
    Slot rules, 2729, 2740, 2816
    SYNC/FREEZE groups, 2744
    Types, 2740, 2816
    With preprocessing, (See I-slave)
DP standard slave,
DP/DP coupler, 2730, 2809
DPRD_DAT, 1102
DPSYC_FR, 1102
DPV0/DPV1, 2779
DPV1, 752
    Analyzing diagnostic information, 2784
    Analyzing interrupt information, 2783
    Configuring an ET 200S, 2753, 2829
    OBs for interrupts, 2781
    Slot model, 2786
DPWR_DAT, 1102
DSCP, 2158
DSL
    Overview, 2597
DSR, 1536
DST
    Daylight saving time, 2114, 2115, 2440, 2441
DTR, 1536
Dummy module, 1078
DxB, (See Direct Data Exchange)
Dynamic reference temperature, 3279

E

EC31-HMI/RTX
    Visualization, 1477
Edit networking
    Disconnecting from a network, 77
Editing networking
    Copying a subnet, 77
    Copying subnets and devices, 78
Effect of Ti
    Isochronous mode, 2909
Effect of To
    Isochronous mode, 2910
E-mail
    Sending, 277
E-mail connection
    Configuring, 275
E-mail connectioon - DNS configuration, 3063
E-mail DB, 278
E-Mail function, 610, 2624
    Alarm events, 610, 2624
    Line monitoring, 610, 2624
E-mail server, 277
Embedded controller, 1478
    Load:Download, 1478
Enable autonegation, 2874
Enabling the firewall
    SCALANCE S, 494, 496
Enabling tunneled communication
    CP 343-1 Adv. / 443-1 Adv., 526
    SCALANCE S, 494, 496
Encryption, 348
End criterion, 1592
    After character delay time elapses, 1592
    End delimiter, 1593
    Fixed message frame length, 1595
End delimiter, 1626

End detection indicator of a receive message frame, 1626
End of detection of accessible devices, 2876
End of sync domain, 2876
End of topology discovery, 2876
End sequence, 1552
End-entity certificate, 311
Enhanced Write Filter (EWF), 1478
Enumeration types, 1271, 1398
Enumerations
    Web server, 1272, 1400
ERPC tag, 3070
Error status, 585, 2028, 2367, 2589
ET 200AL, 3190, 3220
ET 200eco, 3190
ET 200eco PN, 3190, 3207
ET 200iSP, 3189
ET 200iSP distributed I/O station
    Definition, 3193
ET 200L, 3189
ET 200M, 2804, 3189, 3284
    Definition, 3283
    Slot rules, 2729
ET 200MP, 3189, 3270
    Pulse width modulation, 3281
ET 200pro, 3190
ET 200R, 3190
ET 200S, 2804, 3189
    DPV1 mode, 2753, 2829
    Option handling, 2750, 2752, 2826, 2828
    Reference junctions, 2746, 2822
    Slot rules, 2745, 2821
ET 200S 1SI, 1577
ET 200S COMPACT, 3189
ET 200SP, 3189, 3213, 3220
    Application area, 3213
ET 200AL, 3228, 3259
    Area of application, 3256
    Configuration example, 3257
ET 200M distributed IO device, 3283
ET 200SP, 3228
ET 200SP with ET 200AL mixed mode, 3228, 3259
ET-Connection, 3228
ET-Connection rack, 3228
Ethernet, 85, 87
    Number of communication connections, 167
Ethernet interfaces
    EC31-HMI/RTX, 1476
    EC31-RTX, 1476
    EC31-RTX F, 1476
Ethernet non IP frames, 370
Ethernet Statistics
    Frame length, 2041, 2372
    Frame type, 2042, 2373
    Interface statistics, 2040, 2371
    Packet Errors, 2043, 2044, 2373
ETX, 1557
Evaluating interrupt information with DPV1, 2783
Event, 876
Event class, 876
Event class 1 - standard OB events, 878
Event class 2 - synchronous errors, 878
Event class 3 - synchronous errors, 879
Event class 4 - stop events and other mode changes, 882
Event class 5 - status runtime events, 885
Event class 6 - communication events, 887
Event class 8 - diagnostics events for modules, 888
Event class 9 - standard user events, 889
Event classes A and B - free user events, 891
Event ID, 876
Event name, 1373
Events
    Configuration, 2093
    Log table, 582, 2027, 2364, 2586
    Severity filter, 2096, 2427
Events and event ID, 876
Example
    Configuration ET 200AL, 3257
    Specification of a substitute value, 876
Exchangeable medium, 2876
Execution cycle, 2792
Expand ET 200SP, 3228
Expandable racks, 1093
Expanding ET 200SP with ET 200AL modules, 3228
Expansion rack
    ER1 (S7-400), 1093
    IM 46x, 1093
    Rules, 1094
Export file for OPC UA, 1159
Exporting an NTP server, 431
ExtendedKeyUsage, 985

F

Facility, 408
Factory settings
    Resetting to, 3335, 3336, 3337
Fail-safe modules
    Selecting from the hardware catalog, 45, 1089
FAQs
    Download certificate, 1256
    User page as start page, 1263

Fast startup (PROFINET IO), (See Prioritized startup)
Fault monitoring
    Connection status change, 631, 2135, 2449, 2637
    Power supply, 2134, 2448
    Redundancy, 2136
    SIM card, 2638
F-CPU, 45
FDL connection
    Free layer 2 access, 270
    Unspecified, 270
    With broadcast, 270
    With multicast, 272
Feedback data record, 3220, 3262
FETCH frame, 1610
FETCH/WRITE, 978, 1344
FETCH/WRITE services
    Configuration for ISO transport, 263
FETCH_RK, 1580
Fetching data
    RK512, 1616
Field device, 2778, 2834, 2886, 2887, 2925
File allocation table, 3072, 3083
File browser, 1244
File format
    CER, 327
    CRL, 327
    CRT, 327
    DER, 327
    P12, 327
    PEM, 327
Filter
    Hardware catalog, 31
Firewall, 1248
    Creating service groups, 389, 392
    Defining ICMP services, 388
    Firewall rules, 371
    Managing service groups, 389, 392
Firmware, 1375
Firmware card, 1296
Firmware download for S7-300/400 PROFIBUS CPs via CP 5612 / CP 5622, 1490
Firmware loader - Chinese GUI, 3392
Firmware update, 935
    Basics, 3338
    from the project context, 3340
    from the project tree, 3340
Firmware update memory card, 907, 1297
Fixed frame length, 1552
Fixed reference temperature, 3279
Flash test, 3356
Flexible memory capacity, 728
Flutter monitoring, 3286
FM
    With separate MPI address, 1096
Force job, 907
Forward Delay, 591, 2603
Fragment, 1276, 1281, 1401, 1407
Frame size IO device, 3062
frame structure, 1562
Free layer 2 access, 270
Free user events, 891
Freeport, 1571
    Code transparency, 1554
    End criteria, 1552
    Message end, 1550
    Message start, 1550
    Receive buffer, 1555
    Start criteria, 1551
Freeport protocol, 1549
FREEZE, 2745
Frequency measurement (CPU 31xC), 769
Frequency meter, 3204, 3207
FTP, 361, 363, 978, 1344
FTP client
    S7-1500, 3089
FTP configuration, 3072
    File allocation table, 3072
FTP server
    S7-1500, 3082
FTPS certificates, 355
Full-duplex mode, 1583
Full-duplex operation, 1532
Functionality
    PROFIBUS DPV0/DPV1, 2779

G

Gateway, 85
GBIT interface (CPU 1518-4 PN/DP), 2874
Generation of packed addresses, 2749, 2824
Geographic coordinates, 603, 2079, 2407, 2614
Get_Features, 1531, 1571
Gigabit PROFINET interface, 2874
Global firewall rules, 372
    Assigning, 373
Global packet filter rules, 373
GMRP, 2215
Go online, 955, 3354
Going offline, 3355
Gross volume of data, 3062
Group alarm, 979, 1345
Group name, 387, 390
Group properties, 439

Groups, 682, 2295, 2513, 2702
GSD files
Configuring devices (PROFIBUS), 2778, 2834
Creating for I-devices, 2969
Delete, 2832
Delete unused GSD data, 2777, 2833, 2888
Deleting, 2776
GSD revisions (PROFIBUS), 2775, 2831
Installation, 2776, 2831
GSD files (PROFINET), 2884
Changing revision, 2888
Delete, 2887
Export, 2925
Installation, 2886
GSDML, (See GSD files (PROFINET))

H

Half-duplex mode, 1583
Half-duplex operation, 1532
Handshake Protocol, 307
Handshaking, 1332, 1543
Hardware
Configuring and assigning parameters, 40
Edit parameters, 61
Edit properties, 61
Hardware and network editor
Device view, 14
Network view, 12
Topology view, 21
Hardware catalog
Adding device, 41
Browsing, 31
DP slave, 2730, 2808
DP/DP coupler, 2730, 2809
I slave, 2809
I-slave, 2730
Selecting CPUs, 1088
Selecting fail-safe modules, 45, 1089
Selecting the hardware component, 44
Selection of the hardware components, 1435
Task card, 29
Hardware compatibility, 935
Hardware configuration, 911
Adding device, 41
Adding module, 48
Overview of the settings, 11
Hardware data flow control, 1544
Hardware detection, 43
Hardware diagnostics, 3303
Hardware editor
Components, 10
Function, 9
Hardware catalog, 29
Inspector window, 23
Hardware identifier, 1097, 1099, 1103, 1377, 3217, 3267, 3274, 3298, (See Hardware identifier)
Hardware interrupt, 1362
Example, 930
How it works, 1311
I-slave, 2789
OB 40 to OB 47 (S7-300, S7-400), 826
Settings, 748
Hardware interrupt (S7-1500), 1010
Hardware interrupt OB
Description, 1362
Parameter assignment, 1032, 1373
Hardware interrupt OBs (S7-1500), 1010
Hardware RTS always ON, 1545
Hardware RTS always ON, ignore DTR/DSR, 1545
Hardware RTS always switched, 1546
Hardware version, 580, 2025, 2362, 2585
Hardware-controlled data flow control, 1332
HART, 3195
HART variables
Configuring, 3192
Structure, 3192
Hello time, 591, 2602
High-speed analog modules
Isochronous mode, 3244
High-speed counter
Configuring, 1327
General, 1324
How it works, 1324
HMI connection, 127, 978, 1344
HOLD, 718
Hot restart, 711, 743
Hot restart (S7-300, S7-400), 867
Hot restart of the CPU, 1702
How the frequency meter works, 3204
HSC, 1324
HTML5, 1288
HTTP, 388
Load/save, 2086, 2417
HTTPS, 495, 757, 1255, 1386
HTTPS server only, 2075
HW ID, (See Hardware identifier)
See Hardware identifier, 1377

I

I address, 1097, 1377
I device, 2918
I device diagnostics, 2930

I slave
Hardware catalog, 2809
I&M data
downloading to PROFINET IO devices, 3350
I/O
Direct access to, 918, 1304
Direct access to ~, 813
I/O access error, 731, 1302
I/O access error (S7-1500), 1021
I/O access error OB (S7-1500), 1021
I/O access error OB (S7-300, S7-400), 875
I/O address, 918, 1096, 1097, 1304, 1377
I/O addressing, 739
I/O communication, 2849
I/O data area
Access, 736
Data records of the module, 738
Diagnostics data, 737
Parameter data, 737
User data, 737
I/O input, 1299
I/O output, 1299
ICMP, 394, 555, 2565
Icon
For comparison, 3309
For comparison status, 3361
for connection status, 3361
For hardware diagnostics, 3308
for operating state, 3310
For software diagnostics, 3309
Overlay icon, 3310
Identifiers of elements and blocks, 815
I-device, 2836, 2852, 2961
Diagnostics addresses, 2921
Exporting as a GSD file, 2969
Proxy device, 3038
Idle line, 1551
IE/AS-i link PN IO, 2837
IE/PB Link, 2878
IEC V2.2, 959
IEC V2.3, 959
IEEE 802.11n, 2339, 2471
Channel bonding, 2341
Frame aggregation, 2341
Guard interval, 2341
Maximum ratio combining, 2340
MIMO, 2340
Spatial multiplexing, 2340
IEEE 802.3, 370
IEEE tag, 3194
IEEE-compliant LLDP, 959
iFeatures
iREF, 2348
IGMP, 2213
IGMP Join, 2011
IGMP querier, 2213
IIN1.6, 1812
IKE settings, 439
Importing NTP servers, 431
Index, 1468
Industrial PC, (see IPC)
Industrial Security, 1288
Information
ARP table, 581, 2026, 2363, 2586
DSL, 2597
Ethernet Statistics, 2041, 2372
Event log, 582, 2027, 2364, 2586
Frame type, 2042, 2373
Groups, 599, 2075, 2380, 2609
Hardware, 579, 2584
Inter AP blocking, 2382
IPsec VPN, 588, 2599
IPv6 Neighbor Table, 2027, 2362
LLDP, 586, 2049, 2592
Log table, 584, 2588
Mobile, 2593
OpenVPN client, 590, 2601
Packet Errors, 2043, 2044, 2373
Role, 598, 2074, 2381, 2610
Security, 596, 598, 2072, 2074, 2378, 2380, 2607, 2609
Security log, 583, 2587
SHDSL, 2596
SINEMA RC, 589, 2600
SNMP, 586, 2071, 2377, 2591
Software, 579, 2584
Spanning Tree, 2030, 2368
Versions, 579, 2024, 2361, 2584
WLAN authentication log, 2365
Initial state of receive line, 1631
Initialization, 1701
Initialization (user-defined web pages), 1259, 1389
Input (I), 1299
Input byte (IB), 1299
Input word (IW), 1299
Inserting/removing the USB network adapter, 1481
Inspector window, 61
Diagnostics tab, 3304
Hardware and network editor, 23
Installation requirement
PC Station, 1436
Intelligent DP slave, 2764, 2765, (See I-slave)
Intelligent IO device, 2918

Inter AP blocking
Allowed addresses, 2540
Basic, 2539
Configuration, 2539
Information, 2382
Interconnecting, 87
Interconnecting of ports, 2866
Interconnecting ports
Graphic view, 296, 298
Table view, 297, 299
Interface
Displaying, 3316, 3383
Renaming, 299
X27 (RS 485), 1542
X27 (RS422), 1538
Interface module, 1092
Interface properties, 3316, 3383
Interfaces, 1528
Interfaces (PROFINET), 955
Interference frequency suppression, 2746, 3277
Internal network nodes
Configuring, 451
Diagnostics, 465
Internal reference junction, 3279
Interpolator OB, 1026, 1370
Interrupt
Sending in the I-slave, 2759
Interrupt stack, 733
Interrupts
Cyclic interrupt (S7-300, S7-400), 825
Diagnostic interrupt (S7-300, S7-400), 843
Hardware interrupt (S7-300, S7-400), 826
OBs for DPV1, 2781
Pull/plug interrupt (S7-300, S7-400), 846
Time-delay (S7-300, S7-400), 823
Time-of-day interrupt (S7-300, S7-400), 820
With packed addresses, 2754, 2830
IO controller
Devices and modules, 2857
Display in the IO device, 2859
PN interface, 2858
IO device
Display of the IO controller, 2859
Networking, 2864
Update time, 2867
Watchdog time, 2869
IO routing, 2895
IO system, 2861, 2982, 2985, 2988, 2992, 2994, 2996, 3002
Creating, 2864
IO-Link, 3029
IP access control list, 362
IP access protection, 205, 3063, 3066
IP access control list, 3065
IP address, 85, 87, 2840, (See display)
Configuration, 667, 2687
IP address parameters, 2842, 2852
IP addresses
PROFINET, 2889
IP addressing using DHCP, 203
IP mapping, 2392
IP packet filter rules, 394
IP parameters, 3317
IP rule sets, 372
IP services, 387
IP subnet, 87
IP subnet address, 955
IP-ACL (IP access control list), 3064
iPad, 1288
IPC, 1433
iPCF
Configuration, 2399, 2400, 2541
How it works, 2344
PROFINET communication, 2344
Restriction, 2346
iPCF-HT
Configuration, 2543
How it works, 2344
iPCF-MC
PROFINET communication, 2344
iPRP
Configuration, 2548
Information, 2402
IPsec method, 569, 2574
IPsec settings, 439
IPsec VPN
NETMAP, 567, 2571
Source NAT, 567, 2571
IPv4
Notation, 552, 2557
OSPF, 2005
VRRP, 2003
VRRPv3, 2004
IPv4 address, 552, 2557
IPv4 addresses, 402
IPv4 routing
MSDP Cache, 2062
NAT Translations, 2058
PIM BSRs, 2061
PIM interfaces, 2059
PIM Neighbors, 2059
PIM Routes, 2059
PIM RPs, 2060
Routing table, 587, 2593

IPv4 routing over backplane bus (CM/
CP 154x-1), 3301
IPv6
E-mail, 3302
FETCH/WRITE, 3302
FTP client, 3302
FTP server, 3302
Notation, 1974, 2334, 3301
OSPFv3, 2008
SNMP, 3302
Use with the CP 1543-1, 3302
VRRPv3, 2004
IPv6 addresses, 403
IPv6 routing
Default routes, 2410
OSPFv3 AS-Scope LSDB, 2067, 2068
OSPFv3 Interfaces, 2063
OSPFv3 Link-Scope LSDB, 2069
OSPFv3 Neighbors, 2064
OSPFv3 virtual neighbors, 2066
Routing table, 2062, 2069, 2376
Static routes, 2269
IQ sense, 3287
iREF, 2348
Configuration, 2549
IRT, 2902
Setup recommendations, 2881
IRT (Isochronous Realtime Ethernet), 2897, 2901, 2905
ISAKMP, 456
I-slave, 2733, 2755, 2764, 2765, 2810, 2819
Configuring, 2733, 2756, 2820
Configuring a DP slave, 2763
Configuring CP, 2762
Configuring CPU, 2762
Data access, 2821
Data exchange, 2733, 2810
Devices, 2730
Diagnostics addresses, 2757
Direct data exchange, 2734
DP master, 2733
Hardware catalog, 2730
Proxy device, 3035
Sending interrupt with SALRM, 2759
Slot model of the DP slaves, 2786
Triggering hardware interrupt, 2789
I-slave (intelligent DP slave)
Diagnostics, 2766
Interrupt behavior, 2766
ISO protocol, 510
Isochronous mode, 2792, 2836, 2912, 3030, 3035
Combination of DP with and without isochronous mode, 2798
Configuring, 2798, 2913, 2914
CPU properties, 2800
Creating a user program, 2803
Delay time, 2800
DP cycle time and Ti/To values, 2802
Effect of Ti, 2909
Effect of To, 2910
Example: Capture of multiple process tags, 2796
Not selectable, 2801
Oversampling, 3244
Process image partitions, 2800
Properties on PROFIBUS, 2801
Rules, requirements, and supplemental conditions, 2797, 2916
Same Ti/To times for all slaves, 2803
Sequence, 2908
Setting on the subnet, 2915
Settings on the DP slave, 2803
System reserves, 2807
Ti/To overlap, 2806
Time synchronization, 997
Why?, 2906
Isochronous mode interrupt OB, 997, 2795
Assigning parameters, 1033
OB 61 to OB 64 (S7-300, S7-400), 836
Isochronous mode interrupt OBs (S7-1500), 1012
Isochronous PROFIBUS, (See Isochronous mode)
Isochronous real-time communication (IRT), 2836
ISO-on-TCP
Characteristics, 105
TSAP, 123

J

Job buffer, 270, 272
Job header, 270, 272
jQuery, 1288

K

Keepalive frame, 3050
Keepalive frames, 3050
Keyboard operation, 37
KEY-PLUG, 2139, 2142
Formatting, 634, 2141, 2455, 2641
iFeatures, 635, 2456, 2642
KeyUsage, 985

L

LACP, 2201
Language (Web server and CPU display), 948
Language (web server), 1253
Language abbreviation, 1411
Language files
    For diagnostics display, 3078
Language switching, 1285, 1286, 1412
Layer 2, 370, 654, 2674
    MLD (IPv6), 2217
Layer 3, 370, 2142
Layer 3 (IPv6), 2266
    Configuration, 2266
LDB configuration, 1493
Lead time, 997
Learning functionality, 452
Learning mode, 510
LED status, 3386
LEDs, 1470
LEDs of the hardware platform, 1470
Life of certificates, 436
Limit value, 3235
Line break, 1551, 3210
Line monitoring, 610, 2624
Linear axis (CPU 31xC), 785, 792
Link Check, 2038
Link Check Status, 2038
Link Layer Discovery Protocol (LLDP), 959
Linking racks, 1092, 1093
LLDP, 362, 363, 586, 2049, 2592, 2876
LLDP (Link Layer Discovery Protocol), 2876
LLDP (see Link Layer Discovery Protocol), 959
Load memory, 724, 728, 911, 1298
Loading
    Configurations with PROFIBUS, 3035
    Configurations with PROFINET, 3036
    Configurations with Web server, 3034
    Device configurations, 3032
    Distributed I/O, 3033
    Going online, 3041
    GSDML, 3037
    HMI devices, 3040
    I-device, 3038
    IE/PB-Link, 3039
    I-slave, 3035
    Isochronous applications, 3030, 3035
    Proxy device, 3035, 3038
    Shared Devices, 3040
    Subnets, 3034
    to PG/PC, 3031
    to the device, 3030
Local access protection, 981
Local data bit (L), 1299
Local data byte (LB), 1299
Local data double word (LD), 1299
Local data stack, 732
Local data word (LW), 1299
Local logging, 459
Local time, 935, 1314
Local users, 679, 2291, 2510, 2699
Location identifier, 741
Log table
    Event log, 582, 2027, 2364, 2586
    Firewall log, 584, 2588
    Security log, 583, 2587
    WLAN authentication log, 2365
Logging, 459
    CP 343-1 Adv. / 443-1 Adv., 526
    SCALANCE S, 494, 496
Logon, 1250
Logon for web server, 1384
Logon web server, 1384
Logout
    Automatic, 622, 2447, 2636
LSAP configuration, 1490

M

MAC packet filter rules, 400
MAC rule sets, 372
MAC services, 390
Machine tailoring, (see Configuration control for IO systems)
    see Configuration control for IO systems, 2992
Main, 1354
Main (S7-1500), 1000
Main entry, 1571
Man-in-the-middle attack, 308
Manual fragment, 1281, 1407
Manual hot restart, 743
Master, 1632
Maximum configuration of an S7-1500, 1082
Maximum cycle time, 905, 944, 945, 1295, 1320
Maximum number of characters, 1552
MC-Interpolator OB, 1026, 1370
MC-PostServo OB, 1027, 1369
MC-PreServo OB, 1026, 1369
MC-Servo
    Assigning parameters, 1024, 1375
MC-Servo OB, 1023, 1368
MD5, 433

Measuring range, 3249
Measuring window (GATE), 3207
Media converter, 2866
Media redundancy
  Configuring, 2948
Media redundancy (MRP), 2836, 2943, 2952
Media Redundancy Protocol (MRP), 2945
Memory, 911, 915
  Command area, 3393
  Compressing, 3393
  Copying from RAM to ROM, 3393
Memory area
  Distribution, 723
  Load memory, 1298
  of the inputs and outputs, 812
  PI and PQ, 813
  Retentive memory areas, 1300
  Work memory, 1299
Memory areas
  Basis for the calculation, 727
  Load memory, 724
  System memory, 724
  Work memory, 724
Memory bit (M), 1299
Memory byte (MB), 1299
Memory card, 722, 907, 911, 1296, (See Memory Card)
  Formatting, 3338
Memory concept of the S7-400, 726
Memory double word (MD), 1299
Memory reserve, 917
Memory reset, 719, 906, 915, 1296
  Making, 3333
Memory size, 728
Memory test, 1465
Memory word (MW), 1299
Message
  Sending, 1335
  Specifying the end, 1337
  Specifying the start, 1336
Message frame header
  Format of a RK512 command message frame, 1611
Message frame structure, 1637
Message length from message, 1552
Message texts, 1622
  Character conversion table, 1622
  Control characters, 1622
  Control characters table, 1622
  Page layout, 1622
Message timeout, 1552
MIB, 362
Micro memory card, 720
Minimum cycle time, 905, 944, 1295, 1320
Mirroring, 2175, 2178
Missing supply voltage, 3276
Mixed mode, 450
MLD, 2216
Mobile
  Information, 2593
MODB_341, 1581
MODB_441, 1581
Modbus
  End of frame, 1563
  Exception code, 1563
  RS232 signals, 1546
Modbus communication, 1561
Modbus instructions, 1573
Modbus_Comm_Load, 1531, 1573
Modbus_Master, 1531, 1573
Modbus_Slave, 1531, 1573
Mode selector switch, 3386
Module
  Addressing, 1096, 1097, 1377
  Communication properties, 3382
  Communication Status, 464
  Copying, 54
  Deleting, 59
  Determine diagnostics status, 3325
  Inserting, 49, 50
  Moving, 56
  Multiple selection, 51
  Rename, 57
  Replacing, 57
  Select, 47
  Time of day on a module, 3334
Module addressing, 738, 1303
Module arrangement, 48
Module start address, 918, (see Module start address)
Module titles, 46
Module version
  Updating, 60
Module-internal shared input (MSI), 2977
Module-internal shared output (MSO), 2977
Modules
  Configuration overview, 1077
  Parameters in the user program, 1091
  Properties of the CPUs, 1091
Monitor, 2871
Monitoring of the partner port..., 2866
Monitoring window, 3286
Mono-master system, 2731, 2810

Move view
Keyboard operation, 38
Overview navigation, 13, 15, 22
Moving
Hardware components, 56
MPI
Address assignment, 1096
MPI interface, 3383
MPI/DP interface, 3383
MRP, 2945
MRP (Media Redundancy Protocol), 2943
MRP domain, 2952, 3320
MRP domain properties, 3319
MRP Interconnection, 2039
Operating principle, 2001
Topology, 2001
MSDP, 2012, 2264
Peer, 2265
MSDP Cache, 2062
MSI, 919
MSI (module-internal shared input), 920
MSI submodule, 920
MSI/MSO, 2977
MSO, 919
MSTP, 2196, 2502
Multicast, 228, 272, 411, 2211
With UDP, 258, 259
Multilingual, 948
Multi-master systems, 2732
Multiple selection, 47
Hardware component, 51
Multiple Spanning Tree, 2196, 2502
Multiple use IO system, 2982, 2985, 2988, 2991
Multiple use IO systems, 2982
Multiplex/synchronous mode, 3292
Multipoint, 1632

N

NAK, 1557
NAPT, 2343
Configuration, 2507
Configuring, 669, 2230, 2689
NAT, 2343
1-to-1 NAT, 673, 2692
Configuration, 2506
Configuring, 669, 2227, 2229, 2688
Masquerading, 566, 2570
NAPT, 566, 2570
NAT traversal, 572, 2577
NETMAP, 567, 2571
Source NAT, 567, 2571
NAT Translations, 2058
NAT traversal, 572, 2577
NAT/NAPT
Routing, 410
NCM connection
Connection details, 3400
Negotiation, 650, 2127, 2459, 2663
Net volume of data, 3062
Network editor
Components, 10
Function, 9
Hardware catalog, 29
Inspector window, 23
Network ID, 503
Network Load Balancing Manager (NLB), 1519
Network overview
Basic functions, 73
Basic functions for editing the network overview table, 74
Network Syslog, 459
Network view
Adding device, 41
Hardware and network editor, 12
I/O communication, 2849
Meaning, 219
Networking devices
Basics for configuring S7 networks, 69
Editing interface addresses, 76
Editing interface parameters, 75
Editing network parameters, 75
Networking several interfaces at the same time, 71
Networking with an existing subnet, 72
Networking without an existing subnet, 71
Networks within a project, 70
Requirements, 75
Types of communication, 69
Networking in the device view
Networking options, 74
Procedure, 74
Networking mode, 72
No supply voltage, 3235
NTP, 2211
Server, 621, 2123, 2635
NTP (secure), 427
NTP server, 427

O

OB
Event source, 928

Events and OBs, 927, 1305
Overview, 927, 1305
OB 1, 819, 1354
OB 1 PI (OB 1 process image), 812
OB 10 to OB 17 (S7-300, S7-400), 820
OB 100
OB 101 and OB 102 (S7-300, S7-400), 867
OB 121 (S7-300, S7-400), 872
OB 122 (S7-300, S7-400), 875
OB 20 to OB 23 (S7-300, S7-400), 823
OB 30 to OB 38 (S7-300, S7-400), 825
OB 40 to OB 47 (S7-300, S7-400), 826
OB 61 (S7-300, S7-400), 836
OB 62 (S7-300, S7-400), 836
OB 63 (S7-300, S7-400), 836
OB 64 (S7-300, S7-400), 836
OB 80, 1364
OB 80 (S7-300, S7-400), 837
OB 81 (S7-300, S7-400), 840
OB 82, 1365
OB 82 (S7-300, S7-400), 843
OB 83 (S7-300, S7-400), 846
OB 84 (S7-300, S7-400), 850
OB 85, 731
OB 85 (S7-300, S7-400), 851
OB 86 (S7-300, S7-400), 856
OB 87 (S7-300, S7-400), 862
OB 88 (S7-300, S7-400), 864
OB for manufacturer-specific interrupt, 1359
OB for manufacturer-specific interrupt (S7-1500), 1006
OB for manufacturer-specific interrupts (OB 57) (S7-300, S7-400), 834
OB for profile-specific interrupt, 1359
OB for profile-specific interrupt (S7-1500), 1006
OB priority, 928
OB start event, 927
OB start information, 999
OB 61 to OB 64, 2795
OB 83, 1366
Occupation of connection resources, 176
Odd send clock, 2892
Offline, 347
Online, 347
Hardware detection, 43
Online and Diagnostics view, 3303
Online assignment (PROFINET device names), 2849
Online device, 3304
Online diagnostics, 461
Online displays
Orange color, 3312
Online Tools, 3303, 3313
OPC configuration, 1491
OPC properties, 241, 247, 253
OPC server
Central communication service provider, 1431
Slot rules, 1480
Using as SMTP server, 276
OPC server - PLC alarms, 1521
OPC server functions
Data record functions, 1492
For DP master class 2, 1492
OPC UA
Certificate generator, 1142
DB tags, 1156
End points, 1151
Identifier, 1129
Introduction, 1126, 1128
Layer model, 1147
Namespace, 1128
OpenSSL, 1143
PLC tags, 1156
Secure channel, 1146
Secure connection, 1146
Security mechanisms, 1137
Security settings, 1151
Signing and encryption, 1139
X.509 certificates, 1142
OPC UA client, 1518
Authentication, 1237
Basics, 1134
Certificate, 1172, 1235
OPC UA redundancy, 1518
OPC UA server, 1518
Address space, 1129
Addressing, 1162
Application name, 1161
Authentication, 1179
Basics, 1149
Commissioning, 1160
Customizing the server certificate, 1174
Generating a server certificate, 1167
Performance, 1158
Performance increase, 1158
Publishing interval, 1165
Runtime licenses, 1183
Sampling interval, 1166
Security settings, 1170, 1177
Subscription, 1164
TCP port, 1164, 1165
Write and read rights, 1156
XML export file, 1159

Open log file
Packet filter events, 468
Security events, 467
System events, 466
Open User Communication, 205
Ability to read back, 122
Changing parameter values, 122
Connection configuration, 92, 94
Connection description, 106, 109, 114
Connection establishment, 92
Connection parameters, 96
Creating a connection, 100
Deleting an interconnection, 103
General, 91
Instructions, 92
Port numbers,
Protocols used, 104
Starting connection parameter assignment, 99
TCON_FDL, 117
TCON_IP_RFC, 115
TCON_IP_v4, 109
TCON_IP_V4_SEC, 111
TCON_Param, 106
TCON_QDN,
TCON_QDN_SEC,
TSAP, 123
via Industrial Ethernet, 205
Opening Siemens Industry Online Support, 34
OpenSSL, 1143
Operand area, 728, 1299
Displaying, 3384
Operating behavior
Adjusting and loading parameters, 740
Default settings, 740
Enabling system memory, 969, 1338
Module parameters, 740
Using clock memory, 756, 970, 1338
Using time-of-day functions, 753, 967, 1322
Using timers and counters, 736
Operating mode, 525
HOLD, 718
Introduction, 899, 1290
Overview, 707
Priority, 709
RUN, 718, 905, 1295
Runtime events, 885
STARTUP, 901, 1292
STOP, 709, 905, 906, 1295
Transitions, 708, 882, 900, 1291
Operating mode transitions, 1576, 1701
Operation when expected and actual configurations do not match, 2746
Operator
Interfaces, 2666
Optimized block access (FTP - S7-1500), 3086
Option handling, 2750, 2752, 2826, 2828, (See Configuration control (ET 200MP)), (See Configuration control (ET 200SP))
Option handling (see Configuration control), 1112, 1416, 3220, 3262, 3294
Option handling for networks, (see Configuration control for IO systems)
Optional IO device, 2994
Order number, (See Article number)
Order numbers, 1527
Organization block, 1107
Organization block (OB)
Background OB (OB 90) (S7-300, S7-400), 866
Cold restart OB (OB 102) (S7-300, S7-400), 867
Communications error OB (OB 87) (S7-300, S7-400), 862
Complete overview of OBs, 816
CPU hardware fault OB (OB 84) (S7-300, S7-400), 850
Cyclic interrupt OBs (OB 30 to OB 38) (S7-300, S7-400), 825
Cyclic program (OB 1) (S7-300, S7-400), 819
Diagnostic interrupt OB (OB 82) (S7-300, S7-400), 843
Displaying, 3384
Hardware interrupt OBs (OB 40 to OB 47) (S7-300, S7-400), 826
Hot restart OB (OB 101) (S7-300, S7-400), 867
I/O access error OB (OB 122) (S7-300, S7-400), 875
Isochronous mode interrupt OB (OB 61 to OB 64) (S7-300, S7-400), 836
OB for manufacturer-specific interrupts (OB 57) (S7-300, S7-400), 834
Power supply error OB (OB 81) (S7-300, S7-400), 840
Processing abort OB (OB 88) (S7-300, S7-400), 864
Program execution error OB (OB 85) (S7-300, S7-400), 851
Programming error OB (OB 121) (S7-300, S7-400), 872
Pull/plug interrupt OB (OB 83) (S7-300, S7-400), 846
Rack failure OB (OB 86) (S7-300, S7-400), 856
Status interrupt OB (OB 55) (S7-300, S7-400), 829
Time error OB (OB 80) (S7-300, S7-400), 837

Time-delay interrupt OBs (OB 20 to OB 23) (S7-300, S7-400), 823
Time-of-day interrupt OBs (OB 10 to OB 17) (S7-300, S7-400), 820
Update interrupt OB (OB 56) (S7-300, S7-400), 831
Warm restart OB (OB 100) (S7-300, S7-400), 867
OSPF
Area range, 2251
Areas, 2005, 2250
Interfaces, 2252
Link State Advertisement, 2005
OSPFv2 interfaces, 2052
OSPFv2 LSDB (information), 2056
OSPFv2 neighbors, 2054
OSPFv2 virtual neighbors, 2055
Router, 2005
Router status, 2005
Virtual Links, 2255
OSPF (IPv6)
Area Range, 2283
Areas, 2282
OSPFv3 AS-Scope LSDB, 2067, 2068
OSPFv3 interfaces, 2284
OSPFv3 Link-Scope LSDB, 2069
OSPFv3 Neighbors, 2064
OSPFv3 virtual neighbors, 2066
Virtual Links, 2286
OSPFv2
Configuration, 2249
OSPFv3
Configuration, 2281
OUC, (See Open User Communication)
Output (O), 1299
Output byte (QB), 1299
Output double word (QD), 1299
Output rate (high-speed analog modules), 3245
Output word (QW), 1299
Overflow, 3211, 3235
Overlay icon, 3310
Overload behavior, 929
Oversampling, 3244
Overview
Access point, 2382
Clients, 2388
Overlap APs, 2387
Overlapping channels, 2387
Overview navigation, 13, 15, 22
Overwriting the PROFINET device name, 2991

P

P_PRINT, 1579
P_PRINT341, 1579
P_RCV, 1579
P_RCV_RK, 1579
P_RESET, 1579
P_SEND, 1579
P_SND_RK, 1579
P12, 327
P3964_Config, 1531
Packet filter log, 468
Page layout
in message texts, 1622
Panning, (See move view)
Parameter
For CPU, 1314
Parameter (CPU), 935
Parameter assignment
Hardware, 23
Parameter assignment parameters
Hardware, 40
Parameter data, 737
Partner port
Information on monitoring, 2866
Password, 981
For CPU access protection, 935, 1314
Options, 2297, 2510
Recommendations for password assignment, 1254
Password protection, 761, 935, 970, 975, 1314, 1339, 1340, 1384
PC CP, 346
PC Station
Common features of PC Station V2.0 and V1.0, 1433
Configuring offline,
Connections,
Improvements from PC Station V2.0 to V2.1, 1433
Improvements from PC Station V2.1 to V2.2, 1433
Improvements of PC Station V2.0 compared to V1.0, 1433
Installation requirement, 1436
OPC configuration,
Plugging component,
Plugging submodules,
Selecting the version, 1427
Slot rules,
XDB configuration,

PC system
Configuring, 1435
Selection from hardware catalog, 1435
PEM, 327
Performance data, 3384
Period (CPU 31xC), 774
Permit access with PUT/GET communication from remote partners, 978, 1344
Permit overwriting of device name, 2876
Phase shift (S7-1500), 1008
PIM
Interfaces, 2261
Protocol, 2261
RP Candidate, 2263
RP Static, 2262
PIM BSRs, 2061
PIM interfaces, 2059
PIM Neighbors, 2059
Ping, 2143, 2457
PIP, (See process image partition)
Pip (data type), 1109
Plant designation, 741
PLC tags, 1264, 1265, 1268, 1393, 1394, 1395
PLUG, 2142
C-PLUG, 633, 2453, 2639, (C-PLUG)
KEY-PLUG, 635, 2456, 2642, (KEY-PLUG)
Plugging component
PC station, 1480
Plugging submodules
PC station, 1480
PN/PN couplers
Grouping, 3027
Linking Ethernet subnets, 3027
point-to-point, 1986, 2355
Point-to-point communication (PtP), 1329
Freeport protocol, 1329
Point-to-point connection, 1529
Point-to-point connection ((3964(R), CPU 31xC)), 802
Point-to-point connection ((ASCII), CPU 31xC), 795
Point-to-point connection ((RK 512), CPU 31xC)), 806
Point-to-point connection (CPs, ET 200S 1SI), 1577
Port, 87
102 (S7 protocol - TCP), 388
123 (NTP), 411, 426
161 (SNMP), 411
20/21 (FTP), 388
443 (HTTPS), 411
4500 (IPsec), 411
500 (IPsec), 411
500 (ISAKMP - UDP), 456
514 (Syslog), 411
80 (HTTP), 388
8448 (security diagnostics), 538, 541
Configuration, 651, 2129, 2459, 2663
Interconnecting, 2866
Link Check, 2038
Port configuration, 625, 630, 649, 2126, 2134, 2458, 2461, 2662
Renaming, 299
Port configuration, 630, 2134, 2461
Port diagnostics
SFP diagnostics, 636, 2150
Port numbers, 121
Port options, 2875
Enable autonegation, 2874
Monitor, 2871
Transmission rate/duplex, 2871
Port_Config, 1531, 1571
Position feedback (CPU 31xC), 780, 788
Positioning (CPU 31xC), 779
PostServo OB, 1027, 1369
Power budget, 3271
Power segment, 1081
Power supply
Monitoring, 630
Power supply error OB (OB 81) (S7-300, S7-400), 840
Power supply modules, redundant, 1079
PPP
Configuration, 653
Overview, 652
Predefined firewall rules
CP 1543-1, 538, 541
CP 1628, 529
CP 343-1 Adv. / 443-1 Adv., 527
SCALANCE S, 494, 496
PreServo OB, 1026, 1369
Pre-shared keys, 438
Principle of operation
Direct data exchange, 2768
Interrupt generation on the I-slave, 2759
Printer driver
BUSY signal, 1625
Examples, 1623
Format string, 1623
Handshake procedure, 1624
Message text output, 1624
Message texts, 1622
Software handshake, 1624
Variables, 1622

Printing
Printing device view, 34
Printing network view, 34
Prioritization, 2159
Prioritized startup, 2836, 2937
Prioritized startup (PROFINET), 2940
Priority, 2159
Priority class, 822, 882
Private Key, 304
Process data area
I/O addressing, 739
Module addressing, 738, 1303
Module start address, 1303
Process image, 918, 1106, 1107, 1109, 1304
Basics, 730, 1301
I/O access error, 731, 1302
of inputs and outputs, 812
Updating, 731, 1302
Process image partition, 1107, 1109, 1110, 2800
Process mode, 762
Processing abort OB (OB 88) (S7-300, S7-400), 864
ProDiag OB, 1023
ProDiag organization block, 1023
PROFIBUS, 82, 2792
Connection with PROFINET, 2837
Different functionality, 2780
Direct data exchange, 2768
DPV0/DPV1, 2779
ET 200S in DPV1 mode, 2753, 2829
Number of communication connections, 167
Setting up a DP master system, 2771
Switch over DP interface, 2739
PROFIBUS cable configuration
Optical ring, 84
PROFIBUS DP, 2742, 2818
PROFIBUS DPV1, 752
PROFIBUS profile, 82
Different profiles on the same subnet, 82
Effect on the transmission rate, 82
Meaning of profiles, 83
PROFIBUS DP, 2735, 2812
PROFIdrive mode, 1492
PROFIenergy, 2836, 2960, 2961
Profile, (See bus profile)
Profiles, 1359
Profiles (S7-1500), 1006
PROFINET, 87, 510, 959, 1976, 2342, 2451, 2892, 2897, 2901, 2930, 2952
Assigning a device name, 2849
Connection with PROFIBUS, 2837
Device names, 2890
Device number, 2890
Device replacement without exchangeable medium, 2876
Diagnostics, 2897
IP addresses, 2889
Media redundancy, 2943
Number of communication connections, 167
Optimizing with IRT, 2881
Optimizing with RT, 2879
PROFINET device name, 2847, 2991, (See display)
PROFINET interface, 935, 955, 1314, 2852, 3316, 3384
PROFINET interface parameters
Reset, 3395
PROFINET IO, 1976, 2342, 2835, 2836, 2840, 2842, 2858, 2864, 2918
Port options, 2871
PROFINET IO device
Assigning a name, 3343
Name assignment in the dialog, 3345
Name assignment in the online and diagnostics view, 3344
PROFINET IO system, 2861
Creating, 2857
Highlighting, 2860
PROFINET IRT, 2905
PROFINET RT class, 2836
Program
Cyclic (S7-300, S7-400), 819
Program card, 1296
Program cycle OB
Description, 1354
Program execution, 1052
Cyclic, 1354
Program execution error OB (OB 85) (S7-300, S7-400), 851
Program information, 912
Programming
Modbus, 1572
PtP, 1569
USS, 1574
Programming error (S7-1500), 1019
Programming error OB (S7-1500), 1019
Programming error OB (S7-300, S7-400), 872
Project
Load, 1439
Project tree
Adding device, 41
Properties (CPU), 935, 1314
Properties of the VPN group, 439
Properties window, (See Inspector window)
Protection concept, 971, 978, 981, 1341, 1344
Protection level, 761, 970, 975, 1339, 1340

Protocol, 387
Protocol parameters, 1626
Protocols of the communication modules, 1529
Proxy ARP, 521
Proxy device, 3035, 3038
PTC resistor, 3246
PtP instructions, 1571
Public Key, 304
Pull or plug of modules (S7-1500), 1016
Pull/plug interrupt (ET 200M), 3284
Pull/plug interrupt OB, 1366
Pull/plug interrupt OB (S7-1500), 1016
Pull/plug interrupt OB (S7-300, S7-400), 846
Pulling and plugging Ethernet PC CPs, 1481
Pulse extension, 3234
Pulse stretching, 3286
Pulse width modulation
    ET 200MP, 3281
Pulse-width modulation (CPU 31xC), 773
PUT / GET, 978, 1344
PUT frame, 1610
PWM
    ET 200MP, 3281

Q

Q address, 1097, 1377
QI (quality information), (See value status)
QoS, 2159
QoS Trust, 558, 1979, 2336, 2561

R

R, 82
Rack error OB, 1367
Rack failure OB (OB 86) (S7-300, S7-400), 856
Rack failure OB (S7-1500), 1017
Rack or station failure, 1367
Rack or station failure (S7-1500), 1017
Racks, 46, 1092
    Inserting a module, 49
    Maximum number, 1078, 1079
    S7-300 rules for arranging components, 1078
    S7-400 rules for arranging components, 1079
RADIUS, 684, 2298, 2517, 2704
Range of values for IPv4 address, 552, 2557
RCV_PTP, 1580
RDREC, 1102
Readback data record, 3294, (See feedback data record)
Readback data record 197, 3215, 3272, 3294
Reading out a diagnostics buffer, 3327
Reading out service data, 1249
Reading PLC tags
    String or character tags in expressions, 1267
    Tags of the String and Character type, 1266
Reading tags, 1265, 1394
Read-only, 1339
Real-time clock, 3385
Receive buffer, 1555, 1596, 1635
Receive buffer size, 1528
Receive line initial state, 1538
Receive_Config, 1531, 1571
Receive_P2P, 1530, 1571
Receive_Reset, 1531, 1571
Receiver module, 1092
Receiving data, 1571
    3964(R) procedure,
    ASCII driver,
Recipe, 1387
Reconfiguration via user program, 1112, 1416, 3215, 3220, 3262, 3272, 3294
Record Protocol, 307
RECV_440, 1580
Reduction of process reaction time, 2804
Reduction of the system cycle, 2804
Redundancy
    Redundancy domains, 2945
Redundancy group, 1518
Redundancy methods
    HRP, 1987
Redundancy port, 1518
Redundant networks, 659, 2679
Redundant power supply modules, 1079
Redundant sync master, 2904
Reference channel, 3235
Reference channel error, 3277
Reference channel of the module, 3279
Reference junction, 2747, 2822, 3196, 3211, 3235, 3238, 3278, 3279
Reference temperature, 3235, 3238, 3278, 3279
Regional language for the web server, 948, 1253
Rename, 57
Rendezvous point, 2011
Renewing the CA group certificate, 451
Replace modules during operation, 3284
Replacing
    Module, 57
Replacing devices, 57
Report System Errors, 1251, (See Report System Errors), (See System diagnostics)
    Created error OBs, 3370
Reporting system errors, (See System diagnostics)

RES_RCVB, 1580
RES_RECV, 1580
Reserved event classes, 891
Reset, 2083, 2412, 2615
of PROFINET interface parameters, 3395
To factory settings, 3335, 3336
Reset to factory settings, 915
Resetting to factory settings, 3337
Resistance thermometers, (see Thermal resistor)
Resources, 912
Response message frame, 1610
Structure and contents, 1612
Response timeout, 1552
Restart, 743, 2083, 2412, 2615
Retentive address area
Load memory, work memory, system memory, 725
Retentive behavior, 746
Retentive behavior after power failure, 712
Retentive memory, 725, 911
Retentivity, 915, 1300
Retentivity of IP address parameters, 2852
RFC 5280, 304
RI, 1536
Ring port, 2952
Ring Redundancy, 2184
HRP,
MRP,
RIP
RIPv2 statistics, 2058
RIPng
Configuration, 2287
Interfaces, 2288
RIPng Statistics, 2069
RIPv2
Configuration, 2257
Interfaces, 2258
RK 512 protocol (CPU 31xC), 806
RMON
History, 2221
Statistics, 2220
Roles, 681, 2293, 2512, 2701
Root bridge, 2354
Root certificate, 311
Root certification authorities, 358
Root Max Age, 591, 2603
Rotary axis (CPU 31xC), 785, 793
Router, 85, 2855
Router IP address, 503
Routing, 2003
ICMP, 555, 2565
IPv4 routing table, 587, 2593
IPv6 routing table, 2062, 2376
Routing table, 2052
Static routes, 2003
VRRP, 2003
Routing mode, 409
RS232 accompanying signals
Automatic use, 1546
RS232 mode, 1534
RS232 signals, 1535
RS-232/PIP multi-master cable, 1334
RS232C accompanying signals, 1597
Automatic use, 1598
RS422 mode, 1538
RS422 signals, 1538, 1541
RS485 mode, 1541
RSTP, 659, 2679
RT class, 2836
RTS, 1536
Rules for configuring MPI networks
Rules for assigning the MPI address, 78
Rules for network configuration for PROFIBUS networks
Assigning device addresses, 78, 79
RUN, 716, 718, 905, 1295
RUN operating mode, 1049
Runtime meter
Display information, 3389
Run-time meter, 756

S

S_MODB, 1581
S_RCV, 1581
S_RTS, 1581
S_SEND, 1581, 1649
S_USSI, 1581
S_USSR, 1581
S_USST, 1581
S_V24, 1581
S_VSET, 1581
S_VSTAT, 1581
S_XON, 1581
S7 communication, 175
S7 connection, 137
TSAP, 148, 235
S7 CPU
Load memory, 1298
Memory areas, 723
Operand area, 728, 1299
Work memory, 1299
S7 routing
Connection resources, 175

S7-1500
Application area, 1080
Definition, 1080
Power segment, 1081
Properties, 1080
S7-300
Slot rules, 1078
S7-400
Slot rules, 1079
S7-400 CPUs
Memory types, 726
S7-CP, 346
S7-modular Embedded Controller, 1476
Configuring, 1477
S7-PCT, 3029
SALRM, 2759
Sample program
OB 122, 876
OB 81 (S7-300, S7-400), 842
Sampling rate (high-speed analog modules), 3245
Save message text, 1622
Scalable measuring range, 3249
SCALANCE S, 346
SD card, (See Memory card)
Search
Hardware catalog, 31
Secondary sync master, (see Redundant sync master)
Secure communication, 304
Secure Socket Layer, 307
Security (Web server), 1248
Security Configuration Tool
Configuration views, 347
Security event, 979, 1344, 1345
Security events, 467
Security module, 346
Security settings, 613, 982, 2110, 2436, 2626
Roles and rights, 325
Segment diagnostics
Graphic display, 3358
Icons, 3357
Text display, 3358
select before operate, 1808
Selecting specific connection path, 213
Self-signed certificates, 309, 985
Send clock, 997, 2892, 2902
Send clock adaptation, 2892
Send clock cycle, 2867
SEND frame, 1610, 1614
SEND/PUT message frame, 1610, 1611, 1614
SEND_440, 1580
Send_Config, 1531, 1571
Send_P2P, 1530, 1571
SEND_PTP, 1580
SEND_RK, 1580
Sender module, 1092
Sending Data,
3964(R) procedure, 1603
ASCII driver, 1590, 1624
RK512, 1613
Serial data transmission, 1532
Serial number, 580, 2025, 2362, 2585
SERVE_RK, 1580
Server certificate, 568, 1174, 2573
Service data
Saving, 3331
Service groups, 389, 392
Servo OB, 1023, 1026, 1027, 1368, 1369
SET button, 623, 2636
Set_Features, 1531, 1571
Setting the communication load, 85
Setup
Direct data exchange, 2769
Severity, 408
SFP diagnostics, 636, 2150
SFTP
Load/save, 605, 2090, 2422, 2618
SHA algorithm, 613, 2111, 2436, 2626
SHA1, 433
Shared device, 920, 2836, 2961, 2977
Associated IO controllers, 3318
Shared I-device, 2969
SHDSL
Interface, 2671
Overview, 2596, 2670
Short-circuit to ground, 3233
Short-circuit to L+, 3234
SiClock, 392
Signal board, 1381
Inserting, 1382
Signal recorder, 2482, 2595
Signal_Get, 1531, 1571
Signal_Set, 1531, 1571
Signature, 310
SIM
Interfaces, 2664
SIMATIC ET 200AL, 3228, 3255, 3259
SIMATIC memory card, 911
Formatting, 3338
SIMATIC NET PC software for communications processors, 1479
SIMATIC NET version
Onboard interface, 1475
PC Station, 1475

SIMATIC-ACC, 148, 235
Size of the user data IO device, 3062
Slave, 1632
Slot
  Racks, 46
  Select, 47
  Slot model, 2786
  Slot model with DPV1, 2786
  User data with DPV1, 2786
Slot model with DPV1, 2786
Slot rules, 47
  DP slave, 2729, 2740, 2816
  DPV1, 2786
  ET 200M, 2729
  ET 200S, 2745, 2821
  OPC server, 1480
  PC station, 1480
  S7-1500, 1081
  S7-300, 1078
  S7-400, 1079
Smoothing, 3210, 3277
SMS
  receiving, 2647
  sending, 2646
SMS text (TCON_Phone), 116
SMTP
  Client, 600, 2611
SMTP Client, 2096, 2427
SNAT
  Configuring, 671, 2691
SNMP, 362, 559, 601, 613, 1983, 2077, 2110, 2352, 2404, 2436, 2562, 2612, 2626, 3079
  Groups, 613, 2110, 2436, 2626
  Overview, 586, 2071, 2377, 2591
  SNMP trap, 613, 2110, 2435, 2626
  SNMPv1, 559, 1983, 2352, 2562
  SNMPv2c, 559, 1983, 2352, 2562
  SNMPv3, 559, 1983, 2352, 2562
  Users, 614, 2112, 2437, 2627
SNMPv1, 433
SNMPv3, 433
SOFTNET Security Client
  Configuring in the project, 470
  Creating a configuration file, 470
  Database, 471
  Using, 469
SOFTNET Security Client, 346
Software change, 916
Software data flow control, 1544
Software handshake, 1624
Software version, 580, 2025, 2362, 2585
Software-controlled data flow control, 1332
Source NAT
  Masquerading, 566, 2570
Source specific multicast, 2012
Spanning Tree,
  Information, 2030, 2368
  Rapid Spanning Tree, 1986, 2355
Sparse mode, 2011
Special characters
  in tags for Web server, 1264, 1393
Special diagnostics, 3392
Special diagnostics - Chinese GUI, 3392
Special tag (web server), 1269, 1396
Specific blocks, 1052
Spectrum analyzer, 2487
Speedy Splitter, 12, 14, 21, 37
SSH
  Server, 599, 2610
SSH server, 2075
SSL, 307
Standard application, 1430
Standard device, 2884
Standard machine project, 1124, 2981, 2982, 2985, 2988, 2991, 2994, 2996, 3002
Standard mode, 569, 2574
Standard OB events, 878
Standard router, 503
Standard slave, 2778, 2834
Standard user events, 889
Standby module
  Substitute value, 2751, 2827
Standby redundancy, 1989
Start address, 918, 1096, 1097, 1303, 1377
Start character, 1551
Start information, 999
  As for S7-300 and S7-400, 999, 1000, 1001, 1004, 1005, 1006, 1008, 1009, 1011, 1013, 1014, 1015, 1016, 1017, 1019, 1022
  optimized, 999, 1000, 1001, 1004, 1005, 1006, 1008, 1009, 1011, 1013, 1014, 1015, 1016, 1017, 1019, 1022
Start sequences, 1551
Start the online and diagnostics view, 3312
Start value, 911
Starting firmware loader, 3395
Starting NCM S7 diagnostics, 3391
Starting the topology view, 287
STARTUP, 710
  Automatic restart (warm restart), 711
  Automatic restart (warm restart) without battery backup, 711
  Cancellation, 715
  Complete overview,

Function, 901, 1292
Manual restart (warm restart), 711
Organization blocks,
Retentive behavior, 712
STARTUP activities, 714, 903, 1293
Warm restart, 711, 902, 1293
Startup (S7-1500), 999
Startup characteristics, 1701
STARTUP mode, 1047
Startup OB
Description, 1353
Startup OBs (S7-1500), 999
Startup OBs OB 100, OB 101, OB 102 (S7-300, S7-400), 867
Startup of IO devices, 2937
Startup parameters, 742
Start-up parameters, 904, 1294
Startup program (S7-1500), 999
Startup routine, 1353
Startup type, 935, 1314
Stateful inspection firewall, 2568
Stateful packet inspection, 370
Static routes
IPv6 routes, 2269
Station
Copying, 54
Deleting, 59
Moving, 56
Multiple selection, 51
Rename, 57
Renaming, 299
Station failure, (Rack error)
Station name, (See device name)
Station_Info, 1102
Statistical data, 3388
Status, 1358
Module defective, 3325
Status interrupt, 1358
Status interrupt (S7-1500), 1005
Status interrupt OB, 1358
Status interrupt OB (OB 55) (S7-300, S7-400), 829
Status interrupt OB (S7-1500), 1005
Status of the online connection, 3353
Status symbols, 1497
STOP, 709, 716, 905, 906, 1295
Stop events, 882
STOP mode, 1048
Storing data
Storage location for retentive data, 1469
Structure, 3213, 3256, 3271
Structures
Web server, 1275
STX, 1557
Sub-cycle, 3244
Submodule, 2977
Submodule configuration, 920
Subnet, 87
Configuration, 667, 2687
Overview, 665, 2685
Subnet mask, 85, 553, 1972, 2332, 2557, 2842
Subnets
Configuration (IPv6), 2267
Connected Subnets (IPv6), 2267
Default gateway, 2227
Substitute value
Standby module, 2751, 2827
Summary of bit, byte, word, and double word identifiers, 814
Supply voltage, 3235
Supported hardware compatibility, 935
Switching operating mode, 3332
Switchover time, 1335
Switchport, 2875
Symmetric encryption, 306
SYNC, 2745
Sync domain, 2876, 2901, 2902, 3319
Sync domain properties, 3318
Sync master, 2904
SYNC/FREEZE control command, 2744
SYNC_PI, 752
SYNC_PO, 752
Synchronization (user-defined web pages), 1259, 1389
Synchronized execution cycles, 2792
Synchronizing the time, 755, 968
Synchronous cycle (S7-1500), 1012
Synchronous cycle interrupt, 752
Synchronous error
OB 121 (S7-300, S7-400), 872
OB 122 (S7-300, S7-400), 875
Synchronous errors, 878
Syntax for AWP commands, 1264, 1393
Syntax for constants, 814
Syslog
Client, 600, 2611
Syslog Client, 624, 2126, 2448, 2637
Syslog server, 459
System
Configuration, 599, 2403, 2610
General information, 602, 2078, 2405, 2613
System configuration, 2075
System block
Displaying, 3384
System cycle, 2907

System diagnostics
Created blocks, 3369
Diagnostics status DB, 3371
Introduction, 3366, 3377
System events, 466
Configuration, 607, 2424, 2621
Severity filter, 610, 2623
System functions
Parameters in the user program, 1091
System information
Addresses, 2787
System log, 466
System memory, 724, 935, 969, 1314, 1338
Access via FETCH/WRITE, 264
Block stack, 734
Diagnostics buffer, 735, 1303, 3351
Interrupt stack, 733
Local data stack, 732
Operand areas, 728, 1299
process image input/output, 730, 1301
System power supply, 3271
System Time
NTP Client, 619, 2119, 2444, 2632
PTP Client, 2122
SNTP Client, 617, 2117, 2442, 2631
System-relevant information
Manufacturer´s specifications, 3381
Module information, 3314
Module specifications, 3381
Vendor information, 3314

T

T_DC, 997
Tablet, 1288
Tag name
Web server (PLC), 1264, 1393
Tag table
Web server, 1249
Tags, 3070
Task Card
Hardware catalog,
Online Tools, 3313
TCI (Tool Calling Interface), 3028
TCON_FDL, 117
TCON_IP_RFC, 115
TCON_IP_v4, 109
TCON_IP_V4_SEC, 111
TCON_Param, 106
TCON_PHONE for telecontrol connections, 116
TCON_QDN, 112
TCON_QDN_SEC, 113
TCP, 387
Characteristics, 104
Port numbers, 121
TDP (System cycle), 2793
Telnet
Server, 599, 2610
Temperature coefficient, 3209, 3277
Temperature compensation, 3196, 3238
Temporary tags (TEMP)
in cyclic program (S7-300, S7-400), 819
required for OBs, 875
Terminal modules and electronic modules, 3193
Test mode, 762
TFTP
Load/save, 603, 2088, 2419, 2616
Thermal resistor, 3238, 3249
Thermocouple, 3238, 3249
Ti, 997
Ti (input data time read-in), 2804
Ti (read in time of input data), 2793, 2915
Ti/To overlap
Operating principle, 2805
Requirement, 2804
Rules, 2807
Setting, 2806
Time, 935, 1314, 2076, 2404
Time delay interrupt, 1360
Time error interrupt, 1364
Time error interrupt (S7-1500), 1014
Time error OB, 1364
Threshold mechanism, 930
Time error OB (OB 80) (S7-300, S7-400), 837
Time error OB (S7-1500), 1014
Time of day
Precision Time Protocol, 2122
Setting the time of day in the Online and Diagnostics view, 3334
SNTP (Simple Network Time Protocol), 617, 2117, 2442, 2630
Synchronization, 3385
System Time, 616, 2113, 2439, 2629
Time-of-day synchronization, 617, 2117, 2442, 2630
Time of day (S7-1500), 1001
Time setting, 600, 2611
Time stamping, 3197
Time synchronization, 427, 935, 1314
Time-delay interrupt
OB 20 to OB 23 (S7-300, S7-400), 823
Settings, 748
Time-delay interrupt (S7-1500), 1007
Time-delay interrupt OB, 1360

Time-delay interrupt OBs (S7-1500), 1007
Time-of-day function
Basics, 753, 967, 1322
Clock parameters, 1323
Reading the time of day, 754
Reading the time-of-day, 1323
Run-time meter, 756
Setting the time of day, 754
Setting the time-of-day, 1323
Time-of-day format, 753, 967, 1322
Time-of-day interrupt, 749
Cancel, 1356
Function, 1356
OB 10 to OB 17 (S7-300, S7-400), 820
Rules, 1356
Set and activate, 1356
Status query, 1356
Time-of-day interrupt (S7-1500), 1001
Time-of-day interrupt OB
Parameter assignment, 1030, 1371
Time-of-day interrupt OB (S7-1500), 1001
Time-of-day synchronization, 3052, 3098
PROFIBUS CP, 3095
TLS, 307
TM, 2895
To, 997
To (output data time read-in), 2793, 2804
To (output time of output data), 2915
Tool Calling Interface (TCI), 3028
Tool changeover, (See Docking Station)
Tools, (See Docking unit)
Topologies, 1632
Topology
Ring, 2945
Topology discovery, 2876
Topology display, 3078
Topology editor, 87
Topology overview, 21
Topology view
Adding device, 41
Adopt devices identified online, 302
Adopt port interconnections identified online, 301
Advanced offline/online comparison, 294
Compare offline/online, 292
Configured topology, 288, 289
Diagnostics status in the graphic view, 290
Diagnostics status in the table view, 291
Differences compared with the network view, 286
Functions, 285
Hardware and network editor, 21
Interconnecting ports, 295
Transfer area, 2765, 2766, 2786, 2961
Transfer card, 1296, (See Memory Card)
Transfer card; see Transfer card, 1296
Transfer module mapping, 2895
Transmission integrity, 1587
with ASCII driver, 1588
With RK512, 1589
Transmission medium/duplex, 2871
Transmission security, 1532
for Modbus and USS, 1534
with 3964(R), 1534
with Freeport, 1533
Transport Layer Security, 307
Trust Mode, 2159
TSAP
ASCII code table, 125
Structure, 123, 148, 235
TSAP assignment
Examples, 125
Type of CPU startup, 1465
Types of DP slaves, 2740, 2816

U

UA redundancy server, 1516
UART data transmission, 1329
UDP, 387, 394, 426
Characteristics, 105
Port numbers, 121
UDP connection / FDL connection
Unspecified, 228
Underflow, 3211, 3235
Unidirectional/bidirectional data traffic, 1535, 1583, 1598
Unplugged module, 19
Unspecified connection
Characteristics, 229
Configuring graphically, 226
Configuring interactively, 227
Unspecified connections
In PROFIBUS, 203
On Industrial Ethernet, 203
Unspecified CPU, 42
Update, 1359
Update interrupt, 1359
Update interrupt (S7-1500), 1005
Update interrupt OB, 1359
Update interrupt OB (OB 56) (S7-300, S7-400), 831
Update interrupt OB (S7-1500), 1005
Update time, 2867
Updating automatically, 1107
Upload, 3031
User administration, 758

User constant, 1109
User data, 737
Addresses, 2786
Area, 739, 918, 1304
Slots, 2786
User Groups, 682, 2295, 2513, 2702
User interface language for the web server, 948
User Management,
Web server, 1253
User pages, (See User-defined web pages)
User page as start page, 1262
User permissions, 3068
User program, 1051
User-defined web pages, 1257, 1261, 1272, 1273, 1275, 1276, 1278, 1279, 1288, 1388, 1391, 1400, 1401, 1402, 1403, 1404
USS communication, 1567
USS instructions, 1575
USS master
Overview of functions, 1569, 1699
Sequence of function calls, 1700
USS protocol, 1567, 1697
USS protocol: Data field, 1568
USS protocol: Network data block, 1698
USS protocol:Data encryption, 1568, 1698
USS protocol:Data transmission procedure, 1568, 1698
USS protocol:frame structure, 1567
USS protocol:Message frame structure, 1698
USS protocol
General structure of the data block:Parameter area (PKW), 1568
General structure of the data block:Process data area (PZD), 1568
General structure of the network data block:Parameter area (PKW), 1699
General structure of the network data block:Process data area (PZD), 1699
USS_Drive_Control, 1531, 1575
USS_Port_Scan, 1531, 1575
USS_Read_Param, 1531, 1575
USS_Write_Param, 1531, 1575

V

V24_SET, 1579
V24_SET_340, 1579
V24_SET_441, 1580
V24_STAT, 1579
V24_STAT_340, 1579
V24_STAT_441, 1580
Value ranges for the IP address, subnet mask and address of the gateway, 3059
Value status, 919, 920, 2977
Vendor ID, 580, 2025, 2362, 2585
Version overview, 1375
Visualization, 1477
VLAN, 556, 1977, 2337, 2559
Day, 657, 2169, 2494, 2677
Port VID, 657, 2169, 2494, 2677
Priority, 657, 2169, 2494, 2677
VLAN ID, 559, 1979, 2337, 2561
VLAN tag, 558, 1978, 2336, 2560
VLAN operation, 438
VLAN tagging, 438
VPN connection
Status, 588, 2599
Status OpenVPN client, 590, 2601
VPN group, 434
VRRP
Address overview, 2242
Backup router, 2003
Configuration, 2240
Interface Tracking, 2243, 2248, 2280
Master router, 2003
Router, 2239
Virtual router, 2003
VRRP addresses overview (IPv4), 677, 2697
VRRP router, 2003
VRRP routers (IPv4), 674, 2694
VRRP Statistics, 2032
VRRPv3 address configuration (IPv6), 2279
VRRPv3 Addresses Configuration (IPv4), 678, 2247, 2697
VRRPv3 addresses overview (IPv4), 2247
VRRPv3 addresses overview (IPv6), 2279
VRRPv3 Configuration (IPv4), 676, 2246, 2696
VRRPv3 Configuration (IPv6), 2278
VRRPv3 routers (IPv4), 2244, 2276
VRRPv3
Interface Tracking, 678, 2698
VRRPv3 Statistics, 594, 2034, 2605

W

WAN IP address
Specifying, 456
Warm restart, 711, 902, 1293
Warm restart (S7-300, S7-400), 867
Watch table
Web server, 1249
Watchdog time, 2869
WDS, 2473

Web access, 1250
Web application, 1257, 1388
Web Control DB, 1280, 1405
Web data blocks, 1278, 1402, 1403
Web diagnostics, 3078, 3079
Web pages
    Update interval, 3078
Web pages in browser, 1285, 1410
Web server, 757, 948, 1244, 1253, 1257, 1261, 1278, 1285, 1288, 1382, 1384, 1388, 1391, 1402, 1403, (See Web server)
    Array, 1273, 1275
    Certificate, 1255
    Display classes for messages, 1257
    Enable, 1251, 1385
    Enumerations, 1272, 1400
    Fragment, 1276, 1401
    HTTPS, 1255, 1386
    Managing access, 1251
    Rules PLC tag names, 1264, 1393
    Structures, 1275
    User Management, 1253
    User-defined web pages, 1257, 1388
WinCC RT), 1429
Wire break, 3210, 3276
Wireless LAN/PB link, 2837
WLAN
    WLAN authentication log, 2365
WLAN statistics
    Fault, 2394
    Received data frames, 2398
    Sent data frames, 2398
Work memory, 724, 728, 911, 1299
Writing tags, 1395
WRREC, 1102
WWW (instruction), 1279, 1404

X

X.509, 304
X27 (RS 485) interface, 1542
X27 (RS422) interface, 1538
X27 (RS422/485) interface, 1631
XDB configuration, 1442, 1443, 1498
XON/XOFF, 1544

Z

Zoom
    Adjusting the zoom setting, 12, 15, 21
    Keyboard operation, 38
Zoom in/out, (See Zoom)

SIEMENS

Programming the PLC

Programming and Operating Manual

Programming basics 1

Programming recommendations (S7-1200, S7-1500) 2

Data types 3

Instructions 4

Program editor 5

Use software units (S7-1500) 6

Creating and managing blocks 7

Protecting blocks 8

Declaring the block interface 9

Programming data blocks 10

Declaring PLC tags 11

Declaring PLC data types (UDT) 12

Creating LAD programs 13

Creating FBD programs 14

Creating STL programs (S7-300, S7-400, S7-1500) 15

Creating SCL programs 16

10/2018

Continued on next page

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Ⓟ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Programming the PLC

Programming and Operating Manual

Continued

Creating GRAPH programs (S7-300, S7-400, S7-1500) 17

Configuring alarms 18

Compiling and downloading PLC programs 19

Comparing PLC programs 20

Displaying program information 21

Displaying cross-references 22

Testing the user program 23

Supervision of machinery and plants with ProDiag (S7-1500) 24

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Programming basics....35

1.1 Operating system and user program....35

1.2 Blocks in the user program....37
1.2.1 Linear and structured programming....37
1.2.2 Overview of the block types....39
1.2.3 Organization blocks (OB)....39
1.2.4 Functions (FCs)....40
1.2.5 Function blocks (FB)....41
1.2.6 Global data blocks (DB)....42
1.2.7 Global ARRAY data blocks (DB)....43
1.2.8 Example of the use of ARRAY data blocks....44
1.2.9 Instance data blocks....53
1.2.10 CPU data blocks....53
1.2.11 Blocks with optimized access....55
1.2.11.1 Basics of block access....55
1.2.11.2 Setting up block access....57

1.3 Block calls....60
1.3.1 Principles of block calls....60
1.3.2 Call hierarchy....61
1.3.3 Instances....62
1.3.3.1 Basics of instances....62
1.3.3.2 Single instances....63
1.3.3.3 Multi-instances....65
1.3.3.4 Parameter instances....68
1.3.3.5 Example of the use of parameter instances....71
1.3.4 Parameter transfer at block call....76
1.3.4.1 Rules for supplying block parameters....76
1.3.4.2 Parameter assignment to functions....78
1.3.4.3 Parameter assignment to function blocks....80
1.3.4.4 Transfer parameter as copy or as pointer....80
1.3.4.5 Forwarding of block parameters....83

1.4 Using and addressing operands....89
1.4.1 Basic information about operands....89
1.4.2 Keywords....89
1.4.3 Using tags within the program....95
1.4.4 Constants....97
1.4.4.1 Basics of constants....97
1.4.4.2 Declaration of symbolic names for constants....98
1.4.4.3 Data types of constants....100
1.4.4.4 Examples of using constants....101
1.4.5 Addressing operands....103
1.4.5.1 Addressing global variables....103
1.4.5.2 Addressing variables in data blocks....106
1.4.5.3 Addressing operands indirectly....114

1.5 Handling program execution errors....136
1.5.1 Causes of error....136
1.5.2 Overview of mechanisms for error handling....138
1.5.3 EN/ENO mechanism....141
1.5.3.1 Basics of the EN/ENO mechanism....141
1.5.3.2 EN/ENO mechanism is LAD....143
1.5.3.3 EN-/ENO mechanism in LAD....148
1.5.3.4 EN/ENO mechanism in STL (S7-1500)....152
1.5.3.5 EN/ENO mechanism in SCL....155
1.5.3.6 EN/ENO mechanism in GRAPH (S7-1500)....159
1.5.3.7 EN/ENO mechanism in blocks with different network languages (S7-1200, S7-1500)....161
1.5.4 Evaluating errors with output parameter RET_VAL....162
1.5.5 Use of the instructions GET_ERROR and GET_ERR_ID....166
1.5.6 Example of how to handle program execution errors....167

1.6 Program flow control....173
1.6.1 Status word for S7-300, S7-400 (S7-300, S7-400)....173
1.6.1.1 Basic information on the status word (S7-300, S7-400)....173
1.6.1.2 Setting status bits (S7-300, S7-400)....175
1.6.1.3 Querying status bits in STL (S7-300, S7-400)....183
1.6.2 Status word for S7-1500 (S7-1500)....186
1.6.2.1 Basic information on the status word (S7-1500)....186
1.6.2.2 Querying and setting status bits in STL (S7-1500)....187
1.6.3 Master Control Relay (S7-300, S7-400)....191
1.6.3.1 Master control relay (S7-300, S7-400)....191
1.6.3.2 Important notes on using MCR functions (S7-300, S7-400)....192

2 Programming recommendations (S7-1200, S7-1500)....193

2.1 The new S7-1200/1500 CPU functions and programming recommendations at a glance (S7-1200, S7-1500)....193

2.2 Symbolic addressing (S7-1200, S7-1500)....198

2.3 Indirect addressing of ARRAY components (S7-1200, S7-1500)....200

2.4 Using the DB_ANY data type (S7-1200, S7-1500)....204

2.5 Using PLC data types (UDT) (S7-1200, S7-1500)....220

2.6 Using MOVE instructions in STL (S7-1500)....224

2.7 Using IEC timers and counters (S7-1200, S7-1500)....227

3 Data types....231

3.1 Overview of the valid data types....231

3.2 Binary numbers....236
3.2.1 BOOL (bit)....236
3.2.2 Bit strings....236
3.2.2.1 BYTE....236
3.2.2.2 WORD....237
3.2.2.3 DWORD....238
3.2.2.4 LWORD (S7-1500)....240

3.3 Integers....241
3.3.1 SINT (8-bit integers) (S7-1200, S7-1500)....241

3.3.2 USINT (8-bit integers) (S7-1200, S7-1500)....242
3.3.3 INT (16-bit integers)....243
3.3.4 UINT (16-bit integers) (S7-1200, S7-1500)....244
3.3.5 DINT (32-bit integers)....245
3.3.6 UDINT (32-bit integers) (S7-1200, S7-1500)....246
3.3.7 LINT (64-bit integers) (S7-1500)....247
3.3.8 ULINT (64-bit integers) (S7-1500)....248

3.4 Floating-point numbers....251
3.4.1 REAL....251
3.4.2 LREAL (S7-1200, S7-1500)....252
3.4.3 Invalid floating-point numbers....253

3.5 Timers....255
3.5.1 S5TIME (duration) (S7-300, S7-400)....255
3.5.2 TIME (IEC time)....256
3.5.3 LTIME (IEC time) (S7-1500)....257

3.6 Date and time....258
3.6.1 DATE....258
3.6.2 TIME_OF_DAY (TOD)....258
3.6.3 LTOD (LTIME_OF_DAY) (S7-1500)....259
3.6.4 DATE_AND_TIME (date and time of day)....259
3.6.5 LDT (DATE_AND_LTIME) (S7-1500)....261
3.6.6 DTL (S7-1200, S7-1500)....261

3.7 Character strings....263
3.7.1 Character....263
3.7.1.1 CHAR....263
3.7.1.2 WCHAR (S7-1200, S7-1500)....263
3.7.2 Character strings....264
3.7.2.1 STRING....264
3.7.2.2 Structure of a STRING tag....266
3.7.2.3 WSTRING (S7-1200, S7-1500)....267
3.7.2.4 Addressing individual characters of a STRING or WSTRING (S7-1200, S7-1500)....270
3.7.2.5 Transferring a tag of the STRING or WSTRING data type....271

3.8 PLC data types (UDT)....273
3.8.1 Basics of PLC data types (UDT)....273
3.8.2 Transferring a tag of the PLC data type (UDT)....276
3.8.3 Addressing PLC data types (UDT)....277

3.9 STRUCT data structure (anonymous structures)....278
3.9.1 Basic information on STRUCT....278
3.9.2 Structure of a STRUCT tag....280
3.9.3 Addressing STRUCT components....281
3.9.4 Transferring tags of the STRUCT data type....282

3.10 ARRAY....283
3.10.1 Basic information on ARRAY....283
3.10.2 Structure of an ARRAY tag....287
3.10.3 Addressing ARRAY components....289
3.10.4 Transferring tags of the ARRAY / ARRAY[*] data type....291
3.10.5 Examples of how to use ARRAYs....295
3.10.5.1 Example of a multi-dimensional ARRAY....295
3.10.5.2 Example for calculating the scalar product of two vectors with ARRAY[*]....295

3.11 Pointer....300
3.11.1 References (S7-1500)....300
3.11.1.1 Basic information about references (S7-1500)....300
3.11.1.2 Declaring references (S7-1500)....303
3.11.1.3 Referencing (S7-1500)....305
3.11.1.4 Dereferencing (S7-1500)....309
3.11.1.5 Standard instructions with references (S7-1500)....311
3.11.1.6 Assignment attempt from VARIANT to a reference (S7-1500)....314
3.11.1.7 Transferring references as block parameters (S7-1500)....316
3.11.1.8 Example: Transferring tags of different data types using references (S7-1500)....319
3.11.1.9 Example: Return data to the calling block via reference (S7-1500)....321
3.11.2 VARIANT (S7-1200, S7-1500)....324
3.11.2.1 Basic information on VARIANT (S7-1200, S7-1500)....324
3.11.2.2 Applications for pointers in the comparison (S7-1200, S7-1500)....326
3.11.2.3 VARIANT instructions (S7-1200, S7-1500)....327
3.11.2.4 Application options for VARIANT instructions (S7-1200, S7-1500)....328
3.11.2.5 Initializing VARIANT (S7-1200, S7-1500)....331
3.11.2.6 Transferring and reading out various data types with VARIANT (S7-1200, S7-1500)....332
3.11.2.7 Examples of using the VARIANT data type (S7-1200, S7-1500)....333
3.11.3 POINTER (S7-300, S7-400, S7-1500)....341
3.11.4 ANY (S7-300, S7-400, S7-1500)....344

3.12 Parameter types....348
3.12.1 Parameter types....348

3.13 System data types....349
3.13.1 System data types....349

3.14 Hardware data types....351
3.14.1 Hardware data types....351

3.15 Data type conversion for S7-1500: (S7-1500)....354
3.15.1 Overview of data type conversion (S7-1500)....354
3.15.2 Implicit conversions (S7-1500)....356
3.15.2.1 Activate or deactivate IEC check (S7-1500)....356
3.15.2.2 Binary numbers (S7-1500)....357
3.15.2.3 Integers (S7-1500)....364
3.15.2.4 Floating-point numbers (S7-1500)....378
3.15.2.5 Timers (S7-1500)....381
3.15.2.6 Date and time-of-day (S7-1500)....387
3.15.2.7 String (S7-1500)....399
3.15.3 Explicit conversions (S7-1500)....405
3.15.3.1 Binary numbers (S7-1500)....405
3.15.3.2 Integers (S7-1500)....421
3.15.3.3 Floating-point numbers (S7-1500)....449
3.15.3.4 Timers (S7-1500)....455
3.15.3.5 Date and time-of-day (S7-1500)....463
3.15.3.6 String (S7-1500)....479

3.16 Data type conversion for S7-1200 (S7-1200)....489
3.16.1 Overview of data type conversion (S7-1200)....489
3.16.2 Implicit conversions (S7-1200)....491
3.16.2.1 Activate or deactivate IEC check (S7-1200)....491
3.16.2.2 Binary numbers (S7-1200)....492

3.16.2.3 Integers (S7-1200)....496
3.16.2.4 Floating-point numbers (S7-1200)....502
3.16.2.5 Timers (S7-1200)....504
3.16.2.6 Date and time (S7-1200)....505
3.16.2.7 Character strings (S7-1200)....507
3.16.3 Explicit conversions (S7-1200)....511
3.16.3.1 Binary numbers (S7-1200)....511
3.16.3.2 Integers (S7-1200)....521
3.16.3.3 Floating-point numbers (S7-1200)....539
3.16.3.4 Timers (S7-1200)....545
3.16.3.5 Clock and calendar (S7-1200)....547
3.16.3.6 Character strings (S7-1200)....550

3.17 Data type conversion for S7-300, S7-400 (S7-300, S7-400)....555
3.17.1 Overview of data type conversions (S7-300, S7-400)....555
3.17.2 Implicit conversions (S7-300, S7-400)....557
3.17.2.1 Activate or deactivate IEC check (S7-300, S7-400)....557
3.17.2.2 Binary numbers (S7-300, S7-400)....558
3.17.2.3 Integers (S7-300, S7-400)....562
3.17.2.4 Floating-point numbers (S7-300, S7-400)....563
3.17.2.5 Timers (S7-300, S7-400)....564
3.17.2.6 Date and time (S7-300, S7-400)....566
3.17.2.7 Character strings (S7-300, S7-400)....568
3.17.3 Explicit conversions (S7-300, S7-400)....569
3.17.3.1 Binary numbers (S7-300, S7-400)....569
3.17.3.2 Integers (S7-300, S7-400)....577
3.17.3.3 Floating-point numbers (S7-300, S7-400)....583
3.17.3.4 Timer operations (S7-300, S7-400)....584
3.17.3.5 Date and time (S7-300, S7-400)....586
3.17.3.6 String (S7-300, S7-400)....589
3.17.3.7 Additional conversion functions (S7-300, S7-400)....591

4 Instructions....593

4.1 Instructions....593
4.1.1 Example libraries....593
4.1.1.1 Sample Library for Instructions....593
4.1.1.2 Library of General Functions....595
4.1.2 Asynchronous instructions....596
4.1.2.1 Difference between synchronous and asynchronous instructions....596
4.1.3 Basic instructions....599
4.1.3.1 LAD....599
4.1.3.2 FBD....942
4.1.3.3 STL (S7-1500)....1288
4.1.3.4 SCL....1659
4.1.3.5 GRAPH (S7-1500)....1905
4.1.4 Extended instructions....2089
4.1.4.1 Date and time-of-day....2089
4.1.4.2 String + Char....2124
4.1.4.3 Process image (S7-1500)....2197
4.1.4.4 Distributed I/O....2206
4.1.4.5 PROFIenergy (S7-1200, S7-1500)....2366
4.1.4.6 Module parameter assignment (S7-1500)....2420
4.1.4.7 Interrupts....2433

4.1.4.8 Alarms....2522
4.1.4.9 Diagnostics....2571
4.1.4.10 Pulse (S7-1200)....2643
4.1.4.11 Recipes and data logging....2647
4.1.4.12 Data block functions....2710
4.1.4.13 Addressing....2738
4.1.4.14 File handling....2759
4.1.4.15 R/H system....2763
4.1.5 Technology....2765
4.1.5.1 High-speed counters (S7-1200)....2765
4.1.5.2 Counting and measurement (S7-1500) (S7-1500)....2770
4.1.5.3 PID Control (S7-1200, S7-1500)....2787
4.1.5.4 Motion Control (S7-1200, S7-1500, S7-1500T)....3004
4.1.5.5 Time-based IO (S7-1500)....3422
4.1.6 Communication....3442
4.1.6.1 S7 communication....3442
4.1.6.2 Open User Communication....3477
4.1.6.3 OPC UA....3640
4.1.6.4 Web server....3838
4.1.6.5 Communications processor....3849
4.1.6.6 TeleService (S7-1200)....4178

4.2 Instructions....4186
4.2.1 Asynchronous instructions....4186
4.2.1.1 Difference between synchronous and asynchronous instructions....4186
4.2.2 Basic instructions....4189
4.2.2.1 LAD....4189
4.2.2.2 FBD....4397
4.2.2.3 STL....4611
4.2.2.4 SCL....4899
4.2.2.5 GRAPH....5045
4.2.3 Extended instructions....5173
4.2.3.1 Date and time....5173
4.2.3.2 String + Char....5205
4.2.3.3 Process image....5218
4.2.3.4 Distributed I/O....5224
4.2.3.5 PROFIenergy (S7-300, S7-400)....5303
4.2.3.6 Module parameter assignment....5355
4.2.3.7 Interrupts....5366
4.2.3.8 Alarms....5394
4.2.3.9 Diagnostics....5442
4.2.3.10 Data block functions....5514
4.2.3.11 Table functions....5527
4.2.3.12 Addressing....5550
4.2.3.13 Additional functions (S7-300)....5559
4.2.4 Technology....5560
4.2.4.1 PID Control (S7-300, S7-400)....5560
4.2.4.2 Motion Control (S7-300, S7-400)....5808
4.2.4.3 Function modules (S7-300, S7-400)....5870
4.2.4.4 300C functions (S7-300)....6097
4.2.5 Communication....6126
4.2.5.1 Common parameter and error information....6126
4.2.5.2 S7 communication....6131

4.2.5.3 Open User Communication....6186
4.2.5.4 Web server....6242
4.2.5.5 Additional communication....6244
4.2.5.6 Communications processor....6244
4.2.5.7 300C functions....6765
4.2.5.8 Communication with iSlave / iDevice....6784
4.2.5.9 PROFINET CBA....6789
4.2.5.10 MPI communication....6794
4.2.5.11 TeleService....6804

5 Program editor....6831

5.1 Overview of the program editor....6831

5.2 Using the keyboard in the program editor....6836

5.3 Enlarging the programming window....6841

5.4 Setting the mnemonics....6843

5.5 Displaying symbolic and absolute addresses....6844

5.6 Find instructions....6846

5.7 Editing multilingual project texts in blocks....6847

5.8 Use instruction versions....6850
5.8.1 Basic information on instruction versions....6850
5.8.2 Upgrade project and instructions....6851
5.8.3 Update program in the current PLC....6852

5.9 Using instruction profiles....6853
5.9.1 Basics of instruction profiles....6853
5.9.2 Creating new instruction profiles....6853
5.9.3 Opening and editing instruction profiles....6855
5.9.4 Activating and deactivating instruction profiles....6856
5.9.5 Deleting instruction profiles....6857

5.10 Using autocompletion....6859
5.10.1 Basics of autocompletion....6859
5.10.2 Using autocompletion in graphic programming languages....6859
5.10.3 Using autocompletion in textual programming languages....6860

5.11 General settings for the PLC programming....6861
5.11.1 Overview of the general settings....6861
5.11.2 Changing the settings....6864

5.12 Eliminating syntax errors in the program....6865
5.12.1 Basic information on syntax errors....6865
5.12.2 Finding syntax errors in the program....6865

5.13 Changing the programming language....6867
5.13.1 Rules for changing the programming language....6867
5.13.2 Change the programming language....6867

6 Use software units (S7-1500)....6869

6.1 Basics on software units (S7-1500)....6869
6.1.1 Introduction to software units (S7-1500)....6869
6.1.2 Creating a PLC program in software units (S7-1500)....6873

6.1.3 Possible program structures with software units (S7-1500)....6875

6.2 Creating and managing software units (S7-1500)....6878
6.2.1 Creating software units (S7-1500)....6878
6.2.2 Renaming software units (S7-1500)....6878
6.2.3 Copying and pasting software units (S7-1500)....6879
6.2.4 Deleting software units (S7-1500)....6880
6.2.5 Displaying and modifying properties of a software unit (S7-1500)....6881
6.2.6 Managing software unit elements in libraries (S7-1500)....6881

6.3 Inserting program elements (S7-1500)....6883
6.3.1 Creating blocks (S7-1500)....6883
6.3.2 Declaring PLC tags (S7-1500)....6884
6.3.3 Creating PLC data types (S7-1500)....6886
6.3.4 Inserting block calls (S7-1500)....6887
6.3.5 Copying and moving program elements (S7-1500)....6888

6.4 Creating relations (S7-1500)....6892
6.4.1 Basics of relations (S7-1500)....6892
6.4.2 Publish program elements (S7-1500)....6893
6.4.3 Setting up relations (S7-1500)....6895

6.5 Compiling and downloading software units (S7-1500)....6897
6.5.1 Basics for compiling and downloading software units (S7-1500)....6897
6.5.2 Compiling software units (S7-1500)....6899
6.5.3 Downloading software units (S7-1500)....6900
6.5.3.1 Loading software units to a device (S7-1500)....6900
6.5.3.2 Loading software units from a device (S7-1500)....6902
6.5.3.3 Loading software units to a memory card (S7-1500)....6903
6.5.3.4 Loading software units from a memory card (S7-1500)....6904

6.6 Comparing software units (S7-1500)....6906
6.6.1 Basics for software unit comparison (S7-1500)....6906

7 Creating and managing blocks....6907

7.1 Creating blocks....6907
7.1.1 Block folder....6907
7.1.2 Creating organization blocks....6908
7.1.3 Creating functions and function blocks....6909
7.1.4 Creating data blocks....6910
7.1.5 Using blocks from libraries....6911
7.1.6 Copying and pasting blocks....6913
7.1.6.1 Basics of copying and pasting blocks....6913
7.1.6.2 Copying blocks....6916
7.1.6.3 Pasting blocks....6917
7.1.7 Entering a block title....6917
7.1.8 Entering a block comment....6918

7.2 Specifying block properties....6920
7.2.1 Basics of block properties....6920
7.2.2 Overview of block properties....6921
7.2.3 Block time stamps....6925
7.2.4 Displaying and editing block properties....6928

7.3 Managing blocks....6929
7.3.1 Opening blocks....6929

7.3.2 Saving blocks....6930
7.3.3 Closing blocks....6931
7.3.4 Renaming blocks....6931
7.3.5 Deleting blocks offline....6932
7.3.6 Deleting blocks online (S7-300, S7-400)....6933
7.3.7 Deleting CPU data blocks....6933

7.4 Using external source files....6935
7.4.1 Basics of using external source files....6935
7.4.2 Rules for programming external source files....6936
7.4.3 Saving blocks as external source files....6938
7.4.4 Inserting external source files....6939
7.4.5 Opening and editing external source files....6940
7.4.6 Generating blocks from external source files....6941

8 Protecting blocks....6943

8.1 Protecting blocks....6943

8.2 Connecting the password provider....6946

8.3 Show available password provider....6949

8.4 Recording activities of the password providers....6950

8.5 Define the type of password assignment....6952

8.6 Setting up and removing block copy protection (S7-1200, S7-1500)....6954

8.7 Setting up and removing block write protection....6956

8.8 Setting up block know-how protection....6959

8.9 Opening know-how protected blocks....6960

8.10 Printing know-how protected blocks....6962

8.11 Changing a password....6963

8.12 Removing block know-how protection....6964

9 Declaring the block interface....6965

9.1 Overview of the block interface....6965

9.2 Rules for declaring the block interface....6967
9.2.1 General rules for declaring the block interface....6967
9.2.2 Valid data types in the block interface (S7-300, S7-400)....6968
9.2.3 Valid data types in the block interface (S7-1200, S7-1500)....6969
9.2.4 Setting the retentivity of local tags....6971

9.3 Selecting tabular or textual representation of the block interface....6973

9.4 Declaring the block interface....6974
9.4.1 Layout of the block interface....6974
9.4.2 Declaring local tags and constants in the block interface....6975
9.4.3 Declaring a local tag in the program editor....6977
9.4.4 Declaring tags of the ARRAY data type....6978
9.4.5 Declaring tags of STRUCT data type....6980
9.4.6 Declare variables of the STRING and WSTRING data types....6981
9.4.7 Declaring tags based on a PLC data type....6982
9.4.8 Declaring higher-level tags....6983

9.4.9 Declaring multi-instances....6983
9.4.10 Declaring parameter instances....6985
9.4.11 Hiding parameters during the block call....6986
9.4.12 Declaring predefined actual parameters....6987
9.4.13 Edit table of the block interface....6988
9.4.13.1 Inserting table rows....6988
9.4.13.2 Inserting table rows....6988
9.4.13.3 Deleting tags....6988
9.4.13.4 Automatically filling in successive cells....6989
9.4.13.5 Show and hide table columns....6989
9.4.13.6 Editing tags with external editors....6990

9.5 Declare block interface in the textual view....6991
9.5.1 Structure of the textual block interface....6991
9.5.2 Block declaration and return value....6992
9.5.3 Declaration sections....6993
9.5.4 Variable declaration and initialization....6994
9.5.5 Declaration of STRUCT....6995
9.5.6 Declaration of ARRAY....6998
9.5.7 Declaration of STRING and WSTRING....7000
9.5.8 Declaration of instances....7001
9.5.9 Attributes for variables....7002
9.5.10 Comments....7003
9.5.11 Edit textual block interface....7004
9.5.12 Restore default interface of an organization block....7006
9.5.13 Example of a textual block interface....7007

9.6 Editing the properties of local tags and constants....7009
9.6.1 Properties of local tags and constants....7009
9.6.2 Changing properties of local tags and constants....7011
9.6.3 Editing multilingual comments in the block interface (S7-1500)....7013

9.7 Updating the block interface....7015

9.8 Extending the block interface (S7-1200, S7-1500)....7016

10 Programming data blocks....7019

10.1 Basic principles for programming of data blocks....7019

10.2 Structure of the declaration table for data blocks....7021

10.3 Creating data blocks....7023

10.4 Updating data blocks....7025

10.5 Extending data blocks (S7-1200, S7-1500)....7027

10.6 Creating a data structure for global data blocks....7029
10.6.1 Declaring tags of elementary data type....7029
10.6.2 Declaring tags of the ARRAY data type....7030
10.6.3 Declaring tags of STRUCT data type....7031
10.6.4 Declaring tags of the STRING data type....7033
10.6.5 Declaring tags based on a PLC data type....7033

10.7 Defining start values....7035
10.7.1 Basic information on start values....7035
10.7.2 Defining start values....7036

10.8 Setting retentivity....7037
10.8.1 Retentivity of tags in data blocks....7037
10.8.2 Setting retentivity in an instance data block....7037
10.8.3 Setting retentivity in a global data block....7038

10.9 Editing the properties of tags in data blocks....7040
10.9.1 Properties of the tags in data blocks....7040
10.9.2 Changing the properties of tags in instance data blocks....7042
10.9.3 Changing the properties of tags in global data blocks....7044
10.9.4 Editing multilingual comments in the data block (S7-1500)....7045

10.10 Editing the declaration table for data blocks....7047
10.10.1 Inserting table rows....7047
10.10.2 Inserting table rows....7047
10.10.3 Deleting tags....7047
10.10.4 Automatically filling in successive cells....7048
10.10.5 Show and hide table columns....7049
10.10.6 Editing tags with external editors....7049

10.11 Online and diagnostic functions for data blocks....7051
10.11.1 Editing data blocks in online mode....7051
10.11.2 Overview of the online and diagnostic functions in data blocks....7053
10.11.3 Monitoring tags....7054
10.11.4 Modifying tags....7055
10.11.5 Loading start values as actual values....7056
10.11.6 Creating a snapshot of the actual values....7058
10.11.7 Loading snapshots as actual values....7059
10.11.8 Copying the snapshot to the start values....7061
10.11.9 Using setpoints during commissioning....7063
10.11.9.1 Basic information on adjusting setpoints during commissioning....7063
10.11.9.2 Marking data as values that can be set....7063
10.11.9.3 Initializing setpoint values in the online program....7065
10.11.9.4 Applying values from the online program as start values....7066

11 Declaring PLC tags....7069

11.1 Overview of PLC tag tables....7069

11.2 Structure of the PLC tag tables....7070

11.3 Rules for PLC tags....7072
11.3.1 Valid names of PLC tags....7072
11.3.2 Permissible addresses and data types of PLC tags....7073

11.4 Creating and managing PLC tag tables....7076
11.4.1 Creating a PLC tag table....7076
11.4.2 Grouping PLC tag tables....7076
11.4.3 Opening the PLC tag table....7077

11.5 Declaring PLC tags....7079
11.5.1 Entering a PLC tag declaration....7079
11.5.1.1 Declaring tags in the PLC tag table....7079
11.5.1.2 Declaring PLC tags in the program editor....7080
11.5.1.3 Navigate to the device view....7081
11.5.2 Setting the retentivity of PLC tags....7081
11.5.2.1 Retentive behavior of PLC tags....7081

11.5.2.2 Setting the retentive behavior of PLC tags (S7-1200, S7-1500)....7082
11.5.2.3 Setting the retentive behavior of PLC tags (S7-300, S7-400)....7082

11.6 Grouping PLC tags for inputs and outputs in structures....7084
11.6.1 Other useful information regarding structured PLC tags....7084
11.6.2 Creating structured PLC tags....7086

11.7 Declaring global constants....7088
11.7.1 Rules for global user constants....7088
11.7.2 Rules for global system constants....7089
11.7.3 Declaring global constants....7089

11.8 Editing properties....7091
11.8.1 Editing the properties of PLC tags....7091
11.8.1.1 Properties of PLC tags....7091
11.8.1.2 Editing the properties of PLC tags....7091
11.8.2 Editing properties of global constants....7092
11.8.2.1 Properties of global constants....7092
11.8.2.2 Editing properties of global constants....7093
11.8.3 Editing multilingual comments in the PLC tag table (S7-1500)....7093

11.9 Monitoring of PLC tags....7095
11.9.1 Monitoring of PLC tags....7095

11.10 Editing PLC tag tables....7096
11.10.1 Inserting a table row in the PLC tag table....7096
11.10.2 Copying entries in the PLC tag table....7096
11.10.3 Deleting entries in the PLC tag table....7097
11.10.4 Sorting rows in the PLC tag tables....7097
11.10.5 Automatically filling in cells in the PLC tag table....7097
11.10.6 Show and hide table columns....7098
11.10.7 Exporting and importing PLC tags....7098
11.10.7.1 Basics for exporting and importing PLC tags....7098
11.10.7.2 Format of the export file (*.xlsx)....7099
11.10.7.3 Format of the export file (*.xml)....7100
11.10.7.4 Format of the export file (*.sdf)....7100
11.10.7.5 Exporting PLC tags....7101
11.10.7.6 Importing PLC tags....7101
11.10.7.7 Editing individual PLC tags with external editors....7102

12 Declaring PLC data types (UDT) 7105

12.1 Structure of the declaration table for PLC data types....7105

12.2 PLC data types in the project tree....7107

12.3 Creating PLC data types....7108

12.4 Delete PLC data types....7109

12.5 Renumbering PLC data types....7110

12.6 Programming the structure of PLC data types....7111
12.6.1 Declaring tags of elementary data type....7111
12.6.2 Declaring tags of the ARRAY data type....7111
12.6.3 Declaring tags of STRUCT data type 7113
12.6.4 Declaring tags based on a different PLC data type....7114

12.7 Editing tag properties in PLC data types....7115

12.7.1 Properties of tags in PLC data types....7115
12.7.2 Changing the properties of tags in PLC data types....7116
12.7.3 Editing multilingual comments in the PLC data type (S7-1500)....7116

12.8 Editing the declaration table for PLC data types....7118
12.8.1 Inserting table rows....7118
12.8.2 Inserting table rows....7118
12.8.3 Deleting tags....7118
12.8.4 Automatically filling in successive cells....7119
12.8.5 Show and hide table columns....7119

13 Creating LAD programs....7121

13.1 Basic information on LAD....7121
13.1.1 LAD programming language....7121
13.1.2 Overview of the LAD elements....7121

13.2 Settings for LAD....7124
13.2.1 Overview of the settings for LAD....7124
13.2.2 Changing the settings....7124

13.3 Working with networks....7126
13.3.1 Using networks....7126
13.3.2 Inserting networks....7127
13.3.3 Selecting networks....7128
13.3.4 Copying and pasting networks....7129
13.3.5 Deleting networks....7129
13.3.6 Expanding and collapsing networks....7130
13.3.7 Inserting network title....7131
13.3.8 Entering a network comment....7132
13.3.9 Navigating networks....7133

13.4 Inserting LAD elements....7134
13.4.1 Rules for the use of LAD elements....7134
13.4.2 Prohibited interconnections in LAD....7136
13.4.3 Inserting LAD elements using the "Instructions" task card....7136
13.4.4 Inserting LAD elements using an empty box....7137
13.4.5 Selecting the data type of a LAD element....7138
13.4.5.1 Selecting a data type....7138
13.4.5.2 Defining the data type of an instruction....7139
13.4.6 Using favorites in LAD....7141
13.4.6.1 Adding LAD elements to Favorites....7141
13.4.6.2 Inserting LAD elements using favorites....7141
13.4.6.3 Removing LAD elements from Favorites....7142
13.4.7 Insert block calls in LAD....7143
13.4.7.1 Inserting block calls using a drag-and-drop operation....7143
13.4.7.2 Updating block calls in LAD....7144
13.4.7.3 Changing the block call....7146
13.4.7.4 Changing the instance type....7146
13.4.8 Inserting complex LAD instructions....7147
13.4.8.1 Using the "Calculate" instruction (S7-1200, S7-1500)....7147
13.4.9 Using free-form comments....7148
13.4.9.1 Basic information on using free-form comments in LAD....7148
13.4.9.2 Inserting free-form comments....7149
13.4.9.3 Editing free-form comments....7149

13.4.9.4 Deleting free-form comments....7151

13.5 Editing LAD elements....7152
13.5.1 Selecting LAD elements....7152
13.5.2 Copying LAD elements....7153
13.5.3 Cutting LAD elements....7153
13.5.4 Pasting an LAD element from the clipboard....7154
13.5.5 Replacing LAD elements....7155
13.5.6 Inserting additional inputs and outputs in LAD elements (S7-1200, S7-1500)....7155
13.5.7 Removing inputs and outputs (S7-1200, S7-1500)....7156
13.5.8 Deleting LAD elements....7157

13.6 Inserting operands into LAD instructions....7159
13.6.1 Inserting operands....7159
13.6.2 Wiring hidden parameters....7160
13.6.3 Displaying or hiding tag information....7161

13.7 Branches in LAD....7163
13.7.1 Basic information on branches in LAD....7163
13.7.2 Rules for branches in LAD....7163
13.7.3 Inserting branches into the LAD network....7164
13.7.4 Closing branches in the LAD network....7165
13.7.5 Deleting branches in LAD networks....7165

13.8 Crossings in LAD....7166
13.8.1 Basic information on crossings in LAD....7166
13.8.2 Inserting crossings....7166
13.8.3 Rearranging crossings....7167
13.8.4 Deleting crossings....7167

13.9 Rungs in LAD....7169
13.9.1 Basic information on rungs in LAD....7169
13.9.2 Insert rung....7170
13.9.3 Deleting a rung....7170

13.10 LAD programming examples....7172
13.10.1 Example of controlling a conveyor belt....7172
13.10.2 Example of detecting the direction of a conveyor belt....7174
13.10.3 Example of detecting the fill level of a storage area....7176
13.10.4 Example of calculating an equation....7179
13.10.5 Example of controlling room temperature....7181

14 Creating FBD programs....7185

14.1 Basic information on FBD....7185
14.1.1 FBD programming language....7185
14.1.2 Overview of the FBD elements....7185

14.2 Settings for FBD....7188
14.2.1 Overview of the settings for FBD....7188
14.2.2 Changing the settings....7188

14.3 Working with networks....7190
14.3.1 Using networks....7190
14.3.2 Inserting networks....7191
14.3.3 Selecting networks....7192
14.3.4 Copying and pasting networks....7192

14.3.5 Deleting networks....7193
14.3.6 Expanding and collapsing networks....7194
14.3.7 Inserting network title....7194
14.3.8 Entering a network comment....7196
14.3.9 Navigating networks....7197

14.4 Inserting FBD elements....7198
14.4.1 Rules for the use of FBD elements....7198
14.4.2 Inserting FBD elements using the "Instructions" task card....7199
14.4.3 Inserting FBD elements using an empty box....7200
14.4.4 Selecting the data type of an FBD element....7201
14.4.4.1 Selecting a data type....7201
14.4.4.2 Defining the data type of an instruction....7202
14.4.5 Using favorites in FBD....7204
14.4.5.1 Adding FBD elements to Favorites....7204
14.4.5.2 Inserting FBD elements using favorites....7204
14.4.5.3 Removing FBD elements from Favorites....7205
14.4.6 Inserting block calls in FBD....7206
14.4.6.1 Inserting block calls using a drag-and-drop operation....7206
14.4.6.2 Updating block calls in FBD....7207
14.4.6.3 Changing the block call....7209
14.4.6.4 Changing the instance type....7209
14.4.7 Inserting complex FBD instructions....7210
14.4.7.1 Using the "Calculate" instruction (S7-1200, S7-1500)....7210
14.4.8 Using free-form comments....7211
14.4.8.1 Basic information on using free comments in FBD....7211
14.4.8.2 Inserting free-form comments....7212
14.4.8.3 Editing free-form comments....7212
14.4.8.4 Deleting free-form comments....7214

14.5 Editing FBD elements....7215
14.5.1 Selecting FBD elements....7215
14.5.2 Copying FBD elements....7215
14.5.3 Cutting FBD elements....7216
14.5.4 Pasting an FBD element from the clipboard....7217
14.5.5 Replacing FBD elements....7217
14.5.6 Adding additional inputs and outputs to FBD elements (S7-1200, S7-1500)....7218
14.5.7 Removing instruction inputs and outputs (S7-1200, S7-1500)....7219
14.5.8 Deleting FBD elements....7220

14.6 Inserting operands in FBD instructions....7222
14.6.1 Inserting operands....7222
14.6.2 Wiring hidden parameters....7223
14.6.3 Displaying or hiding tag information....7224

14.7 Branches in FBD....7226
14.7.1 Basic information on branches in FBD....7226
14.7.2 Rules for branches in FBD....7226
14.7.3 Inserting branches in FBD networks....7227
14.7.4 Deleting branches in FBD networks....7227

14.8 Logic paths in FBD....7229
14.8.1 Basic information on logic paths in FBD....7229
14.8.2 Inserting a logic path....7230
14.8.3 Deleting a logic operation path....7230

14.9 FBD programming examples....7232
14.9.1 Example of controlling a conveyor belt....7232
14.9.2 Example of detecting the direction of a conveyor belt....7234
14.9.3 Example of detecting the fill level of a storage area....7236
14.9.4 Example of calculating an equation....7240
14.9.5 Example of controlling room temperature....7242

15 Creating STL programs (S7-300, S7-400, S7-1500)....7245

15.1 STL Basics (S7-300, S7-400, S7-1500)....7245
15.1.1 STL programming language (S7-300, S7-400, S7-1500)....7245
15.1.2 Data exchange in STL (S7-300, S7-400, S7-1500)....7245
15.1.3 Parameter passing via registers (S7-300, S7-400, S7-1500)....7247
15.1.3.1 Parameter assignment via registers during block call in STL (S7-300, S7-400, S7-1500)..7247
15.1.3.2 Activating parameter passing via registers (S7-1500)....7249

15.2 Settings for STL (S7-300, S7-400, S7-1500)....7250
15.2.1 Overview of the settings for STL (S7-300, S7-400, S7-1500)....7250
15.2.2 Changing the settings (S7-300, S7-400, S7-1500)....7250

15.3 Working with networks (S7-300, S7-400, S7-1500)....7251
15.3.1 Using networks (S7-300, S7-400, S7-1500)....7251
15.3.2 Inserting networks (S7-300, S7-400, S7-1500)....7251
15.3.3 Selecting networks (S7-300, S7-400, S7-1500)....7252
15.3.4 Copying and pasting networks (S7-300, S7-400, S7-1500)....7253
15.3.5 Deleting networks (S7-300, S7-400, S7-1500)....7253
15.3.6 Expanding and collapsing networks (S7-300, S7-400, S7-1500)....7254
15.3.7 Inserting network title (S7-300, S7-400, S7-1500)....7255
15.3.8 Entering a network comment (S7-300, S7-400, S7-1500)....7256
15.3.9 Navigating networks (S7-300, S7-400, S7-1500)....7257

15.4 Inserting STL instructions (S7-300, S7-400, S7-1500)....7259
15.4.1 Rules for STL (S7-300, S7-400, S7-1500)....7259
15.4.2 Instructions from the global library "Long Functions" (S7-300, S7-400, S7-1500)....7260
15.4.3 Entering STL instructions manually (S7-300, S7-400, S7-1500)....7262
15.4.4 Inserting STL instructions using the "Instructions" task card (S7-300, S7-400, S7-1500)..7263
15.4.5 Selecting the data type of an STL instruction (S7-300, S7-400, S7-1500)....7265
15.4.5.1 Selecting a data type (S7-300, S7-400, S7-1500)....7265
15.4.5.2 Defining the data type of an instruction (S7-300, S7-400, S7-1500)....7265
15.4.6 Inserting operands (S7-300, S7-400, S7-1500)....7265
15.4.7 Displaying or hiding tag information (S7-300, S7-400, S7-1500)....7267
15.4.8 Using favorites in STL (S7-300, S7-400, S7-1500)....7268
15.4.8.1 Adding STL elements to Favorites (S7-300, S7-400, S7-1500)....7268
15.4.8.2 Inserting STL instructions using favorites (S7-300, S7-400, S7-1500)....7268
15.4.8.3 Removing STL instructions from Favorites (S7-300, S7-400, S7-1500)....7269
15.4.9 Inserting block calls in STL (S7-300, S7-400, S7-1500)....7270
15.4.9.1 Manually inserting block calls (S7-300, S7-400, S7-1500)....7270
15.4.9.2 Inserting block calls using a drag-and-drop operation (S7-300, S7-400, S7-1500)....7271
15.4.9.3 Updating block calls in STL (S7-300, S7-400, S7-1500)....7272
15.4.9.4 Changing the instance type (S7-300, S7-400, S7-1500)....7274
15.4.10 Inserting comments (S7-300, S7-400, S7-1500)....7275

15.5 Editing STL instructions (S7-300, S7-400, S7-1500)....7277
15.5.1 Selecting STL instructions (S7-300, S7-400, S7-1500)....7277
15.5.2 Copying STL instructions (S7-300, S7-400, S7-1500)....7277

15.5.3 Cutting STL instructions (S7-300, S7-400, S7-1500)....7278
15.5.4 Pasting STL instructions from the Clipboard (S7-300, S7-400, S7-1500)....7279
15.5.5 Deleting STL instructions (S7-300, S7-400, S7-1500)....7279

15.6 STL programming examples (S7-300, S7-400, S7-1500)....7280
15.6.1 Example of controlling a conveyor belt (S7-300, S7-400, S7-1500)....7280
15.6.2 Example of detecting the direction of a conveyor belt (S7-300, S7-400, S7-1500)....7282
15.6.3 Example of detecting the fill level of a storage area (S7-300, S7-400, S7-1500)....7284
15.6.4 Example of calculating an equation (S7-300, S7-400, S7-1500)....7286
15.6.5 Example for heating an oven (S7-300, S7-400, S7-1500)....7288
15.6.6 Example of a step sequence (S7-300, S7-400, S7-1500)....7290

16 Creating SCL programs....7293

16.1 Basics of SCL....7293
16.1.1 Programming language SCL....7293
16.1.2 Expressions....7294
16.1.3 Arithmetic expressions....7295
16.1.4 Relational expressions....7297
16.1.5 Logical expressions....7303
16.1.6 Operators and operator precedence....7304
16.1.7 Value assignments....7305
16.1.8 Calculating with floating-point numbers (REAL and LREAL) in SCL....7308
16.1.9 Calculating with constants in SCL....7313

16.2 Settings for SCL....7317
16.2.1 Overview of the settings for SCL....7317
16.2.2 Changing the settings....7318

16.3 The programming window of SCL....7319
16.3.1 Overview of the programming window....7319
16.3.2 Customizing the programming window....7320
16.3.3 Formatting SCL code....7321
16.3.4 Expanding and collapsing sections of code....7322
16.3.5 Navigate to definitions....7323
16.3.6 Using bookmarks....7324
16.3.6.1 Basics of bookmarks....7324
16.3.6.2 Setting bookmarks....7325
16.3.6.3 Navigating between bookmarks....7325
16.3.6.4 Deleting bookmarks....7326

16.4 Entering SCL instructions....7328
16.4.1 Rules for SCL instructions....7328
16.4.2 Entering SCL instructions manually....7329
16.4.3 Inserting SCL instructions using the "Instructions" task card....7330
16.4.4 Surround program code with structure elements....7331
16.4.5 Working with regions....7331
16.4.5.1 Using regions....7331
16.4.5.2 Inserting regions....7334
16.4.5.3 Copying and inserting regions....7335
16.4.5.4 Navigating in regions....7336
16.4.5.5 Deleting regions....7338
16.4.6 Defining the data type of an SCL instruction....7339
16.4.6.1 Basic information on the data types of SCL instructions....7339
16.4.6.2 Changing the data type of an SCL instruction....7340

16.4.6.3 Modifying the data types of IEC timers and IEC counters....7340
16.4.6.4 Example for changing the data type of an SCL instruction....7340
16.4.7 Displaying or hiding tag information....7341
16.4.8 Using Favorites in SCL....7342
16.4.8.1 Adding SCL instructions to the Favorites....7342
16.4.8.2 Inserting SCL instructions using Favorites....7343
16.4.8.3 Removing SCL instructions from the Favorites....7343
16.4.9 Insert block calls in SCL....7344
16.4.9.1 Basic information on the block call in SCL....7344
16.4.9.2 Manually inserting block calls....7349
16.4.9.3 Inserting block calls with drag-and-drop....7351
16.4.9.4 Updating block calls....7352
16.4.10 Expanding and reducing the parameter list....7354
16.4.11 Insert pragmas (S7-1200, S7-1500)....7355
16.4.12 Inserting comments....7356

16.5 Editing SCL instructions....7358
16.5.1 Selecting instructions....7358
16.5.2 Copying, cutting and pasting instructions....7358
16.5.3 Deleting instructions....7359

16.6 SCL programming examples....7360
16.6.1 Example of controlling a conveyor belt....7360
16.6.2 Example of detecting the direction of a conveyor belt....7362
16.6.3 Example of detecting the fill level of a storage area 7364

17 Creating GRAPH programs (S7-300, S7-400, S7-1500)....7369

17.1 Basic information on GRAPH (S7-300, S7-400, S7-1500)....7369
17.1.1 GRAPH programming language (S7-300, S7-400, S7-1500)....7369
17.1.2 Sequencers (S7-300, S7-400, S7-1500)....7371
17.1.2.1 Basics on sequencers (S7-300, S7-400, S7-1500)....7371
17.1.2.2 Elements of a sequencer (S7-300, S7-400, S7-1500)....7372
17.1.3 Steps and transitions (S7-300, S7-400, S7-1500)....7375
17.1.3.1 Basics on steps (S7-300, S7-400, S7-1500)....7375
17.1.3.2 Step elements (S7-300, S7-400, S7-1500)....7376
17.1.3.3 Basics on transitions (S7-300, S7-400, S7-1500)....7377
17.1.4 Actions (S7-300, S7-400, S7-1500)....7378
17.1.4.1 Basics on actions (S7-300, S7-400, S7-1500)....7378
17.1.4.2 Standard actions (S7-300, S7-400, S7-1500)....7380
17.1.4.3 Event-dependent actions (S7-300, S7-400, S7-1500)....7381
17.1.4.4 Block calls (S7-300, S7-400, S7-1500)....7384
17.1.4.5 Timers (S7-300, S7-400, S7-1500)....7385
17.1.4.6 Counter operations (S7-300, S7-400, S7-1500)....7387
17.1.5 Permanent instructions (S7-300, S7-400, S7-1500)....7388
17.1.6 Conditions (S7-300, S7-400, S7-1500)....7389
17.1.7 Interlock and supervision alarms (S7-300, S7-400, S7-1500)....7389
17.1.8 Block interface of a GRAPH function block (S7-300, S7-400, S7-1500)....7390
17.1.8.1 Basics on the block interface of a GRAPH function block (S7-300, S7-400, S7-1500)....7390
17.1.8.2 Input parameters (S7-300, S7-400, S7-1500)....7393
17.1.8.3 Output parameters (S7-300, S7-400, S7-1500)....7397
17.1.8.4 Static parameters (S7-300, S7-400, S7-1500)....7400
17.1.8.5 Extending the standard behavior of GRAPH (S7-1500)....7409

17.2 Settings for GRAPH (S7-300, S7-400, S7-1500)....7412

17.2.1 Overview of the settings for GRAPH (S7-300, S7-400, S7-1500)....7412
17.2.2 Changing the settings (S7-300, S7-400, S7-1500)....7414
17.2.3 GRAPH versions (S7-300, S7-400, S7-1500)....7415
17.2.3.1 Overview of GRAPH versions (S7-300, S7-400, S7-1500)....7415
17.2.3.2 Defining the version of the GRAPH function block (S7-300, S7-400, S7-1500)....7415
17.2.4 Specifying the programming language in networks (S7-300, S7-400, S7-1500)....7416
17.2.5 Specifying the interface parameter set (S7-300, S7-400, S7-1500)....7417
17.2.6 Specifying the memory space model of the data block (S7-300, S7-400)....7418
17.2.7 Modifying settings for internal parameters (S7-300, S7-400, S7-1500)....7419

17.3 Programming window of GRAPH (S7-300, S7-400, S7-1500)....7421
17.3.1 Overview of the programming window (S7-300, S7-400, S7-1500)....7421
17.3.2 "Permanent pre-instructions" view (S7-300, S7-400, S7-1500)....7423
17.3.3 Sequence view (S7-300, S7-400, S7-1500)....7424
17.3.4 Single step view (S7-300, S7-400, S7-1500)....7426
17.3.5 "Permanent post-instructions" view (S7-300, S7-400, S7-1500)....7428
17.3.6 Alarm view (S7-300, S7-400, S7-1500)....7429

17.4 Using Favorites in GRAPH (S7-300, S7-400, S7-1500)....7431
17.4.1 Adding GRAPH elements to Favorites (S7-300, S7-400, S7-1500)....7431
17.4.2 Inserting GRAPH elements using Favorites (S7-300, S7-400, S7-1500)....7432
17.4.3 Removing GRAPH elements from Favorites (S7-300, S7-400, S7-1500)....7432

17.5 Programming the structure of a sequencer (S7-300, S7-400, S7-1500)....7434
17.5.1 Inserting the sequencers (S7-300, S7-400, S7-1500)....7434
17.5.2 Renaming sequencers (S7-300, S7-400, S7-1500)....7434
17.5.3 Adding comments for sequencers (S7-300, S7-400, S7-1500)....7435
17.5.4 Inserting elements in a sequencer (S7-300, S7-400, S7-1500)....7436
17.5.5 Ending sequencers (S7-300, S7-400, S7-1500)....7437
17.5.6 Deleting sequencers (S7-300, S7-400, S7-1500)....7437
17.5.7 Using steps and transitions (S7-300, S7-400, S7-1500)....7438
17.5.7.1 Insert steps and transitions (S7-300, S7-400, S7-1500)....7438
17.5.7.2 Specifying and canceling the initial step (S7-300, S7-400, S7-1500)....7440
17.5.7.3 Renumbering steps and transitions (S7-300, S7-400, S7-1500)....7441
17.5.7.4 Renaming steps and transitions (S7-300, S7-400, S7-1500)....7443
17.5.7.5 Adding comments for steps (S7-300, S7-400, S7-1500)....7444
17.5.7.6 Copying and inserting steps and transitions (S7-300, S7-400, S7-1500)....7444
17.5.7.7 Skip steps (S7-300, S7-400, S7-1500)....7445
17.5.7.8 Deleting steps and transitions (S7-300, S7-400, S7-1500)....7446
17.5.7.9 Displaying and editing translations of step and transition names (S7-1500)....7447
17.5.8 Using branches (S7-300, S7-400, S7-1500)....7448
17.5.8.1 Creating alternative branches (S7-300, S7-400, S7-1500)....7448
17.5.8.2 Creating a simultaneous branch (S7-300, S7-400, S7-1500)....7448
17.5.8.3 Closing branches (S7-300, S7-400, S7-1500)....7449
17.5.8.4 Deleting branches (S7-300, S7-400, S7-1500)....7450
17.5.9 Using jumps (S7-300, S7-400, S7-1500)....7451
17.5.9.1 Inserting jumps (S7-300, S7-400, S7-1500)....7451
17.5.9.2 Changing the jump destination (S7-300, S7-400, S7-1500)....7451
17.5.9.3 Navigating to the jump destination (S7-300, S7-400, S7-1500)....7452
17.5.9.4 Deleting jumps (S7-300, S7-400, S7-1500)....7453

17.6 Programming actions (S7-300, S7-400, S7-1500)....7454
17.6.1 Inserting actions (S7-300, S7-400, S7-1500)....7454
17.6.2 Defining the data type of a GRAPH instruction (S7-300, S7-400, S7-1500)....7457

17.6.2.1 Basic information on the data types of GRAPH instructions (S7-300, S7-400, S7-1500)..7457
17.6.2.2 Changing the data type of a GRAPH instruction (S7-300, S7-400, S7-1500).....7458
17.6.3 Changing actions (S7-300, S7-400, S7-1500).....7459
17.6.4 Copying, cutting, and inserting actions (S7-300, S7-400, S7-1500).....7460
17.6.5 Moving actions (S7-300, S7-400, S7-1500).....7461
17.6.6 Deleting actions (S7-300, S7-400, S7-1500).....7462

17.7 Programming conditions (S7-300, S7-400, S7-1500).....7463
17.7.1 Programming interlocks (S7-300, S7-400, S7-1500).....7463
17.7.2 Programming supervisions (S7-300, S7-400, S7-1500).....7464
17.7.2.1 Inserting supervision conditions (S7-300, S7-400, S7-1500).....7464
17.7.2.2 Using step time monitoring (S7-1500).....7465
17.7.3 Programming transitions (S7-300, S7-400, S7-1500).....7469
17.7.4 Programming permanent instructions (S7-300, S7-400, S7-1500).....7471
17.7.4.1 Working with networks (S7-300, S7-400, S7-1500).....7471
17.7.4.2 Inserting permanent instructions (S7-300, S7-400, S7-1500).....7477
17.7.5 Displaying or hiding tag information (S7-300, S7-400, S7-1500).....7478

17.8 Inserting block calls in GRAPH (S7-300, S7-400, S7-1500).....7480
17.8.1 Inserting block calls manually (S7-300, S7-400, S7-1500).....7480
17.8.2 Inserting block calls using drag-and-drop operation (S7-300, S7-400, S7-1500).....7481
17.8.3 Updating block calls in GRAPH (S7-300, S7-400, S7-1500).....7483
17.8.4 Changing the block call (S7-300, S7-400, S7-1500).....7484

17.9 Specifying alarms (S7-300, S7-400, S7-1500).....7485
17.9.1 Enabling and disabling the alarm view (S7-300, S7-400, S7-1500).....7485
17.9.2 Editing alarms (S7-300, S7-400, S7-1500).....7485

17.10 Acquiring initial values (S7-1500).....7488
17.10.1 Overview of initial value acquisition (S7-1500).....7488
17.10.2 Activating initial value acquisition (S7-1500).....7489
17.10.3 Resetting initial values (S7-1500).....7490
17.10.4 Example of order of the list (S7-1500).....7491

18 Configuring alarms.....7493

18.1 Introduction to alarm configuration (S7-300, S7-400, S7-1500).....7493

18.2 Assigning alarm numbers (S7-300, S7-400, S7-1500).....7494

18.3 Components of an alarm (S7-300, S7-400, S7-1500).....7495

18.4 Available alarm blocks (S7-300, S7-400, S7-1500).....7496

18.5 Alarm type and alarms (S7-300, S7-400, S7-1500).....7498

18.6 Formal parameters, alarm data types and alarm blocks (S7-300, S7-400, S7-1500).....7499

18.7 Layout of the alarm editor (S7-300, S7-400, S7-1500).....7501

18.8 Working with the alarm editor (S7-300, S7-400, S7-1500).....7502

18.9 Creating and editing alarms (S7-300, S7-400).....7503
18.9.1 Creating program alarms (S7-300, S7-400).....7503
18.9.2 Editing program alarms in the alarm editor (S7-300, S7-400).....7503
18.9.3 Editing program alarms in the program editor (S7-300, S7-400).....7504
18.9.4 Deleting program alarms (S7-300, S7-400).....7504
18.9.5 Editing the alarm type (S7-300, S7-400).....7505
18.9.6 Creating an instance DB and editing alarm instances (S7-300, S7-400).....7505

18.9.7 Creating user diagnostic alarms (S7-300, S7-400)....7506
18.9.8 Editing user diagnostic alarms (S7-300, S7-400)....7506
18.9.9 Deleting user diagnostic alarms (S7-300, S7-400)....7507
18.9.10 Insert associated values in alarms (S7-300, S7-400)....7507
18.9.11 Structure of associated values (S7-300, S7-400)....7508
18.9.12 Examples of associated values (S7-300, S7-400)....7509
18.9.13 Deleting associated values (S7-300, S7-400)....7510

18.10 Creating and editing alarms (S7-1500)....7511
18.10.1 Creating program alarms (S7-1500)....7511
18.10.2 Editing program alarms in the alarm editor (S7-1500)....7512
18.10.3 Editing program alarms in the program editor (S7-1500)....7512
18.10.4 Deleting program alarms (S7-1500)....7513
18.10.5 Editing the alarm type (S7-1500)....7513
18.10.6 Creating an instance DB and editing alarm instances (S7-1500)....7513
18.10.7 Insert associated values in alarms (S7-1500)....7514
18.10.8 Settings when entering associated values (S7-1500)....7516
18.10.9 Special features of associated values (S7-1500)....7517
18.10.10 Examples of associated values (S7-1500)....7517
18.10.11 Deleting associated values (S7-1500)....7518

18.11 Texts and attributes (S7-300, S7-400, S7-1500)....7519
18.11.1 Keywords for alarm texts (S7-300, S7-400, S7-1500)....7519
18.11.2 Entering texts (S7-300, S7-400, S7-1500)....7520
18.11.3 Basics for locking/unlocking texts and attributes (S7-300, S7-400, S7-1500)....7520
18.11.4 Locking new alarms and show/hide interlock symbols (S7-300, S7-400, S7-1500)....7521
18.11.5 Locking texts (S7-300, S7-400, S7-1500)....7522
18.11.6 Locking attributes (S7-300, S7-400, S7-1500)....7523

18.12 Text lists for alarms (S7-300, S7-400, S7-1500)....7525
18.12.1 Basics of alarm text lists (S7-300, S7-400, S7-1500)....7525
18.12.2 Editing text lists for alarms (S7-300, S7-400, S7-1500)....7525
18.12.3 Integrating texts from text lists in alarms (S7-300, S7-400, S7-1500)....7526
18.12.4 Example of integrating texts from text lists in alarms (S7-300, S7-400, S7-1500)....7527

18.13 Alarm classes (S7-300, S7-400, S7-1500)....7528
18.13.1 Creating alarm classes (S7-300, S7-400, S7-1500)....7528
18.13.2 Editing alarm classes (S7-300, S7-400, S7-1500)....7529
18.13.3 Deleting alarm classes (S7-300, S7-400, S7-1500)....7529
18.13.4 Priority of the alarm class (S7-300, S7-400, S7-1500)....7530
18.13.5 Exporting and importing alarm classes (S7-300, S7-400, S7-1500)....7531
18.13.5.1 Basics for exporting and importing alarm classes (S7-300, S7-400, S7-1500)....7531
18.13.5.2 Exporting alarm classes (S7-300, S7-400, S7-1500)....7531
18.13.5.3 Importing alarm classes (S7-300, S7-400, S7-1500)....7532

18.14 Exporting and importing alarm texts (S7-300, S7-400, S7-1500)....7533
18.14.1 Basics for exporting and importing alarm texts (S7-300, S7-400, S7-1500)....7533
18.14.2 Exporting alarm texts (S7-300, S7-400, S7-1500)....7534
18.14.3 Importing alarm texts (S7-300, S7-400, S7-1500)....7535

19 Compiling and downloading PLC programs....7537

19.1 Basics for compiling and downloading PLC programs....7537

19.2 Compiling blocks....7539
19.2.1 Basic information on compiling blocks....7539

19.2.2 Compiling blocks in the project tree....7541
19.2.3 Compiling blocks in the program editor....7542
19.2.4 Correcting compilation errors....7543

19.3 Downloading blocks for S7-1200/1500 (S7-1200, S7-1500)....7544
19.3.1 Introduction to downloading blocks (S7-1200, S7-1500)....7544
19.3.2 Downloading blocks in the "RUN" operating mode to the device (S7-1200, S7-1500)....7548
19.3.3 Downloading blocks from program editor to device (S7-1200, S7-1500)....7552
19.3.4 Downloading blocks from the project tree to the device (S7-1200, S7-1500)....7554
19.3.5 Uploading blocks from device (S7-1200, S7-1500)....7556
19.3.6 Downloading blocks to a memory card (S7-1200, S7-1500)....7557
19.3.7 Uploading blocks from a memory card (S7-1200, S7-1500)....7559
19.3.8 Switching off the sequencer prior to loading a GRAPH DB (S7-1500)....7559
19.3.9 Loading block extensions without reinitialization (S7-1200, S7-1500)....7560
19.3.9.1 Basic information on loading block extensions without reinitialization (S7-1200, S7-1500)....7560
19.3.9.2 Setting the reserved memory for block extensions (S7-1200, S7-1500)....7562
19.3.9.3 Activating memory reserve (S7-1200, S7-1500)....7564
19.3.9.4 Deactivating memory reserve (S7-1200, S7-1500)....7565
19.3.9.5 Extending the block interface or data block (S7-1200, S7-1500)....7565
19.3.9.6 Resetting the reserved memory (S7-1200, S7-1500)....7566

19.4 Downloading blocks for S7-300/400 (S7-300, S7-400)....7568
19.4.1 Introduction to loading of blocks (S7-300, S7-400)....7568
19.4.2 Downloading blocks in the "RUN" operating state to the device (S7-300, S7-400)....7571
19.4.3 Downloading blocks in the program editor to device (S7-300, S7-400)....7575
19.4.4 Downloading blocks from the project tree to the device (S7-300, S7-400)....7576
19.4.5 Uploading blocks from device (S7-300, S7-400)....7578
19.4.6 Downloading blocks to a memory card (S7-300, S7-400)....7579
19.4.7 Uploading blocks from a memory card (S7-300, S7-400)....7580
19.4.8 Switching off the sequencer prior to loading a GRAPH DB (S7-300, S7-400)....7581

19.5 Downloading PLC tags (S7-1200, S7-1500)....7582
19.5.1 Introduction to downloading PLC tags (S7-1200, S7-1500)....7582
19.5.2 Online/offline visibility of PLC tags (S7-1200, S7-1500)....7584

20 Comparing PLC programs....7585

20.1 Basic information on comparing PLC programs....7585
20.1.1 Introduction to comparing PLC programs....7585
20.1.2 Comparing blocks and PLC data types....7587
20.1.3 Comparing PLC tags....7588
20.1.4 Comparison of PLC programs based on checksums....7588

20.2 Comparing blocks....7590
20.2.1 Comparing blocks in the compare editor....7590
20.2.2 Performing detailed block comparisons....7591
20.2.2.1 Start a detailed comparison for LAD/FBD/STL/SCL blocks....7591
20.2.2.2 Start detailed comparison for GRAPH blocks (S7-300, S7-400, S7-1500)....7593
20.2.2.3 Visualization of the comparison result....7596
20.2.2.4 Navigating in the detailed comparison....7612
20.2.2.5 Changing blocks during the detailed comparison....7613
20.2.2.6 Updating comparison results....7615

20.3 Comparing PLC tags....7616

20.4 Comparing PLC data types....7619

21 Displaying program information....7623

21.1 Overview of available program information....7623

21.2 Displaying an assignment list....7624
21.2.1 Introduction to the assignment list....7624
21.2.2 Layout of the assignment list....7625
21.2.3 Symbols in the assignment list....7626
21.2.4 Displaying an assignment list....7627
21.2.5 Setting the view options for the assignment list....7627
21.2.6 Filter options in the assignment list....7628
21.2.7 Defining filters for assignment list....7629
21.2.8 Filtering an assignment list....7630
21.2.9 Defining retentive memory areas for bit memories....7631
21.2.10 Enabling the display of retentive bit memories....7632

21.3 Displaying the call structure....7633
21.3.1 Introduction to the call structure....7633
21.3.2 Symbols in the call structure....7634
21.3.3 Layout of the call structure....7635
21.3.4 Displaying the call structure....7636
21.3.5 Setting the view options for the call structure....7637
21.3.6 Introducing the consistency check in the call structure....7638
21.3.7 Checking block consistency in the call structure....7638

21.4 Displaying the dependency structure....7640
21.4.1 Introduction to the dependency structure....7640
21.4.2 Layout of the dependency structure....7641
21.4.3 Symbols in the dependency structure....7641
21.4.4 Displaying the dependency structure....7642
21.4.5 Setting the view options for the dependency structure....7643
21.4.6 Introducing the consistency check in the dependency structure....7643
21.4.7 Checking block consistency in the dependency structure....7644

21.5 Displaying CPU resources....7646
21.5.1 Introducing resources....7646
21.5.2 Layout of the "Resources" tab....7648
21.5.3 Displaying resources....7649
21.5.4 Selecting the maximum load memory available....7650

22 Displaying cross-references....7651

22.1 General information about cross references....7651

22.2 Structure of the cross-reference list....7653

22.3 Settings in the cross-reference list....7655

22.4 Displaying the cross-reference list....7658

22.5 Sorting the cross-reference list....7660

22.6 Filter options for cross-references....7662
22.6.1 Introduction to filtering of cross-references....7662
22.6.2 Structure of the filter dialog....7664
22.6.3 Attributes, operators and values in the filter dialog....7665
22.6.4 Creating user-defined filters for cross-references....7667
22.6.5 Editing user-defined filters....7668

22.6.6 Deleting user-defined filters....7670
22.6.7 Exporting and importing user-defined filters....7671
22.6.8 Setting a filter as the default setting....7673
22.6.9 Duplicating a filter....7674
22.6.10 Filtering the cross-reference list....7675

22.7 Printing the cross-reference list....7676

22.8 Adding a new object to the cross-reference list....7677

22.9 Show overlapping accesses in the cross-reference list....7678

22.10 Restoring cross-references after project upgrade....7681

22.11 Displaying and editing cross-references in the Inspector window....7682

23 Testing the user program....7685

23.1 Basics of testing the user program....7685

23.2 Testing with program status....7687
23.2.1 Introduction to testing with program status....7687
23.2.2 Setting the call environment....7689
23.2.2.1 Basics of the call environment....7689
23.2.2.2 Setting the call environment for block monitoring....7690
23.2.3 Switching test with program status on/off....7691
23.2.4 Monitoring loops....7693
23.2.5 Monitoring structures....7695
23.2.6 Editing blocks during the program test....7697
23.2.7 Modifying tags during monitoring....7697
23.2.8 Switching display formats in the program status....7698
23.2.9 Special considerations when testing GRAPH programs (S7-300, S7-400, S7-1500)....7699
23.2.9.1 Basics on testing GRAPH programs (S7-300, S7-400, S7-1500)....7699
23.2.9.2 Specifying the test settings (S7-300, S7-400, S7-1500)....7702
23.2.9.3 Controlling a sequencer during testing (S7-300, S7-400, S7-1500)....7703
23.2.10 Examples of program status display....7708
23.2.10.1 Program status display for LAD programs....7708
23.2.10.2 Program status display for FBD programs....7709
23.2.10.3 Program status display for STL programs (S7-300, S7-400, S7-1500)....7710
23.2.10.4 Program status display for SCL programs....7711
23.2.10.5 Program status display for GRAPH programs (S7-300, S7-400, S7-1500)....7712

23.3 Testing with breakpoints (S7-300, S7-400, S7-1500)....7716
23.3.1 Introduction to testing with breakpoints (S7-300, S7-400, S7-1500)....7716
23.3.2 Security information for testing with breakpoints (S7-300, S7-400, S7-1500)....7717
23.3.3 Safety information for testing with breakpoints (S7-300, S7-400, S7-1500)....7718
23.3.4 Rules for testing with breakpoints (S7-300, S7-400, S7-1500)....7720
23.3.5 Status display for breakpoints (S7-300, S7-400, S7-1500)....7721
23.3.6 Setting the call environment for breakpoints (S7-300, S7-400, S7-1500)....7722
23.3.7 Setting breakpoints (S7-300, S7-400, S7-1500)....7723
23.3.8 Enabling breakpoints (S7-300, S7-400, S7-1500)....7725
23.3.9 Disabling breakpoints (S7-300, S7-400, S7-1500)....7728
23.3.10 Navigating to the breakpoints (S7-300, S7-400, S7-1500)....7729
23.3.11 Deleting breakpoints (S7-300, S7-400, S7-1500)....7730
23.3.12 Testing with breakpoints in single step mode (S7-300, S7-400, S7-1500)....7731
23.3.13 "Run to cursor" function (S7-300, S7-400, S7-1500)....7733
23.3.14 Monitoring tags when the breakpoint is reached (S7-1500)....7734

23.3.15 Modifying tags when the breakpoint reached (S7-300, S7-400, S7-1500).....7736
23.3.16 Resuming program execution (S7-300, S7-400, S7-1500).....7738

23.4 Testing with the watch table.....7739
23.4.1 Introduction to testing with the watch table.....7739
23.4.2 Layout of the watch table.....7740
23.4.3 Basic mode and expanded mode in the watch table.....7741
23.4.4 Icons in the watch table.....7742
23.4.5 Creating and editing watch tables.....7743
23.4.5.1 Creating a watch table.....7743
23.4.5.2 Opening a watch table.....7744
23.4.5.3 Copying and pasting a watch table.....7744
23.4.5.4 Saving a watch table.....7745
23.4.6 Entering tags in the watch table.....7745
23.4.6.1 Basic information on entering tags in the watch table.....7745
23.4.6.2 Permitted operands for the watch table.....7746
23.4.6.3 Permissible modify values for the watch table.....7748
23.4.6.4 Overview of the display formats.....7750
23.4.6.5 Selecting the display format for tags.....7752
23.4.6.6 Creating and editing comment lines.....7752
23.4.7 Monitoring tags in the watch table.....7754
23.4.7.1 Introduction to monitoring tags in the watch table.....7754
23.4.7.2 Setting the monitoring and modify mode.....7755
23.4.7.3 "Monitor all" command for tags.....7757
23.4.7.4 "Monitor now" command for tags.....7758
23.4.8 Modifying tags in the watch table.....7759
23.4.8.1 Introduction to modifying tags.....7759
23.4.8.2 Modify tags to "0".....7760
23.4.8.3 Modify tags to "1".....7761
23.4.8.4 "Modify now" command for tags.....7762
23.4.8.5 "Modify with trigger" command for tags.....7764
23.4.8.6 Enable peripheral outputs.....7765

23.5 Testing with the force table.....7768
23.5.1 Introduction for testing with the force table.....7768
23.5.2 Safety precautions when forcing tags.....7769
23.5.3 Layout of the force table.....7770
23.5.4 Basic mode and expanded mode in the force table.....7771
23.5.5 Icons in the force table.....7771
23.5.6 Open and edit force table.....7772
23.5.6.1 Display force table.....7772
23.5.6.2 Open force table.....7773
23.5.6.3 Save force table.....7773
23.5.7 Entering tags in the force table.....7774
23.5.7.1 Basic principles for entering tags in the force table.....7774
23.5.7.2 Permitted operands for the force table.....7775
23.5.7.3 Permitted force values for the force table.....7776
23.5.7.4 Overview of the display formats.....7777
23.5.7.5 Selecting the display format for tags.....7779
23.5.7.6 Creating and editing comment lines.....7780
23.5.8 Monitoring tags in the force table.....7781
23.5.8.1 Introduction to monitoring tags in the force table.....7781
23.5.8.2 Setting the monitoring mode in the force table.....7782

23.5.8.3 "Monitor all" command for tags....7783
23.5.8.4 "Monitor now" command for tags....7784
23.5.9 Forcing tags in the force table....7784
23.5.9.1 Introduction to forcing tags....7784
23.5.9.2 Safety precautions when forcing tags....7786
23.5.9.3 Updating forced operands....7787
23.5.9.4 Force tags to "0"....7788
23.5.9.5 Force tags to "1"....7789
23.5.9.6 "Force all" command for tags....7791
23.5.10 Stop forcing tags....7793
23.5.10.1 Stop forcing all tags....7793
23.5.10.2 Stop forcing individual tags....7794

24 Supervision of machinery and plants with ProDiag (S7-1500)....7797

24.1 System requirements for ProDiag (S7-1500)....7797
24.1.1 Notes on licenses (S7-1500)....7797
24.1.2 Setting ProDiag licenses (S7-1500)....7797
24.1.3 Software and hardware requirements (S7-1500)....7798

24.2 Technical specifications for ProDiag (S7-1500)....7799
24.2.1 Technical specifications for ProDiag (S7-1500)....7799

24.3 Basics of supervision with ProDiag (S7-1500)....7800
24.3.1 Introduction to ProDiag (S7-1500)....7800
24.3.2 Example for creating a supervision based on an operand supervision (S7-1500)....7803
24.3.3 Types of supervision (S7-1500)....7806
24.3.3.1 Overview of the types of supervision (S7-1500)....7806
24.3.3.2 Operand supervision (S7-1500)....7807
24.3.3.3 Interlock supervision (S7-1500)....7809
24.3.3.4 Action supervision (S7-1500)....7811
24.3.3.5 Reaction supervision (S7-1500)....7812
24.3.3.6 Position supervision (S7-1500)....7814
24.3.3.7 Text message (S7-1500)....7816
24.3.3.8 Error message (S7-1500)....7817
24.3.4 Examples for individual types of supervision (S7-1500)....7819
24.3.4.1 Introduction (S7-1500)....7819
24.3.4.2 Example of an operand supervision (S7-1500)....7820
24.3.4.3 Example of an interlock supervision (S7-1500)....7822
24.3.4.4 Example of an action supervision (S7-1500)....7825
24.3.4.5 Example of a reaction supervision (S7-1500)....7828
24.3.4.6 Example of a position supervision (S7-1500)....7831
24.3.4.7 Example for a text message (S7-1500)....7834
24.3.4.8 Example of an error message (S7-1500)....7835
24.3.5 Application examples for supervisions with ProDiag (S7-1500)....7837

24.4 Creating ProDiag function blocks (S7-1500)....7838
24.4.1 Overview of the ProDiag function blocks (S7-1500)....7838
24.4.2 Structure of a ProDiag function block (S7-1500)....7839
24.4.3 Saving the layout of the columns in the ProDiag FB (S7-1500)....7839
24.4.4 Creating ProDiag function blocks (S7-1500)....7839
24.4.4.1 Creating ProDiag function blocks when creating global supervisions (S7-1500)....7839
24.4.4.2 Creating ProDiag function blocks during the block call (S7-1500)....7840
24.4.4.3 Creating ProDiag function blocks at the instance data block of a function block (S7-1500)..7841
24.4.4.4 Creating ProDiag function blocks by using the "Add new block" dialog. (S7-1500)....7842

24.4.4.5 Creating ProDiag function blocks in the ProDiag overview tables (S7-1500).....7843

24.5 Using a central time stamp (S7-1500).....7845
24.5.1 Using a central time stamp (S7-1500).....7845
24.5.2 Multiple supervision alarms of a ProDiag FB within one cycle time (S7-1500).....7846

24.6 Using ProDiag instance data bases (S7-1500).....7848
24.6.1 Structure of a ProDiag instance data block of version 1.0 (S7-1500).....7848
24.6.2 Content of a ProDiag instance data block of version 1.0 (S7-1500).....7849
24.6.3 Structure of a ProDiag instance data block of version 2.0 (S7-1500).....7855
24.6.4 Content of a ProDiag instance data block of version 2.0 (S7-1500).....7856

24.7 Defining properties for a ProDiag function block (S7-1500).....7862
24.7.1 Structure of supervision settings at a ProDiag FB (S7-1500).....7862
24.7.2 Creating the enablers for supervision alarms (S7-1500).....7863
24.7.2.1 Overview of the enablers (S7-1500).....7863
24.7.2.2 Using the global enabler (S7-1500).....7864
24.7.2.3 Using the category enabler (S7-1500).....7865
24.7.3 Defining an acknowledge tag for individual categories (S7-1500).....7866
24.7.3.1 Overview of the acknowledge tags (S7-1500).....7866
24.7.3.2 Use acknowledge tag (S7-1500).....7866
24.7.4 Setting display classes (S7-1500).....7867
24.7.4.1 Overview of display classes (S7-1500).....7867
24.7.4.2 Setting the display class (S7-1500).....7867
24.7.5 Displaying the number of supervisions for a ProDiag FB (S7-1500).....7868

24.8 Defining the ProDiag supervision settings (S7-1500).....7869
24.8.1 Overview of the ProDiag supervision settings (S7-1500).....7869
24.8.2 Defining general ProDiag supervision settings (S7-1500).....7869
24.8.2.1 Structure of the ProDiag supervision settings (S7-1500).....7869
24.8.2.2 Categories (S7-1500).....7870
24.8.2.3 Subcategories (S7-1500).....7873
24.8.2.4 Types of supervision (S7-1500).....7876
24.8.3 Specifying central ProDiag alarm texts (S7-1500).....7879
24.8.3.1 Structure of the alarm text editor (S7-1500).....7879
24.8.3.2 Structure of a GRAPH supervision alarm (S7-1500).....7881
24.8.3.3 Structure of a global supervision alarm (tag) (S7-1500).....7882
24.8.3.4 Structure of a local supervision alarm (parameter) (S7-1500).....7884
24.8.3.5 Creating alarm texts (S7-1500).....7885
24.8.3.6 Exporting ProDiag and GRAPH texts (S7-1500).....7886
24.8.4 Exporting ProDiag supervision settings (S7-1500).....7887
24.8.5 Importing ProDiag supervision settings (S7-1500).....7887

24.9 Creating ProDiag supervisions (S7-1500).....7889
24.9.1 Overview of the supervisions (S7-1500).....7889
24.9.2 Possible uses of supervisions (S7-1500).....7892
24.9.3 Creating global supervision (S7-1500).....7893
24.9.4 Creating a local supervision (S7-1500).....7895
24.9.5 Creating a specific text field (S7-1500).....7898
24.9.5.1 Use of the associated values.....7898
24.9.5.2 Creating a specific text field (S7-1500).....7900
24.9.5.3 Displaying and editing translations of the specific text field (S7-1500).....7901
24.9.6 Deleting a supervision (S7-1500).....7902

24.10 Copying ProDiag function blocks (S7-1500).....7904

24.10.1 Overview of copying ProDiag function blocks (S7-1500)....7904
24.10.2 Copying ProDiag function block (S7-1500)....7904
24.10.3 Pasting a ProDiag function block (S7-1500)....7905

24.11 Assigning ProDiag function blocks (S7-1500)....7906
24.11.1 Overview of the assignment to a ProDiag FB (S7-1500)....7906
24.11.2 Assigning global supervisions to a ProDiag FB (S7-1500)....7906
24.11.3 Assigning local supervisions to a ProDiag FB (S7-1500)....7907
24.11.4 Assigning an instance data block to a ProDiag FB (local supervisions) (S7-1500)....7909
24.11.5 Changing the existing assignment of a ProDiag FB (S7-1500)....7910
24.11.6 Removing the existing assignment of a ProDiag FB (S7-1500)....7911

24.12 Editing ProDiag function blocks (S7-1500)....7913
24.12.1 Inserting new supervisions in the ProDiag FB (S7-1500)....7913
24.12.2 Copying supervisions in the ProDiag FB (S7-1500)....7913
24.12.3 Deleting supervisions in the ProDiag FB (S7-1500)....7914
24.12.4 Sorting supervisions in the ProDiag FB (S7-1500)....7915

24.13 Calling the ProDiag function blocks (S7-1500)....7916
24.13.1 Overview of calling the ProDiag function blocks (S7-1500)....7916

24.14 Load ProDiag blocks (S7-1500)....7917
24.14.1 Load ProDiag blocks (S7-1500)....7917

24.15 Displaying ProDiag supervisions (S7-1500)....7918
24.15.1 Overview of the display options (S7-1500)....7918
24.15.2 Displaying supervisions in the ProDiag overview editor (S7-1500)....7919
24.15.2.1 Structure of the ProDiag overview editor (S7-1500)....7919
24.15.2.2 Display supervisions of all global tags (S7-1500)....7919
24.15.2.3 Displaying supervisions of all function blocks (S7-1500)....7920
24.15.2.4 Displaying supervisions of all called function blocks (S7-1500)....7921
24.15.3 Working with the ProDiag overview editor (S7-1500)....7921
24.15.3.1 Inserting new tag supervisions in the ProDiag overview editor (S7-1500)....7921
24.15.3.2 Copying tag supervisions in the ProDiag overview editor (S7-1500)....7922
24.15.3.3 Deleting tag supervisions in the ProDiag overview editor (S7-1500)....7923
24.15.3.4 Sorting tag supervisions in the ProDiag overview editor (S7-1500)....7923
24.15.3.5 Editing supervising definitions on the function block in the ProDiag overview editor (S7-1500)....7924
24.15.3.6 Deleting supervision definitions on the function block in the ProDiag overview editor (S7-1500)....7925
24.15.4 Displaying supervisions of a function block (S7-1500)....7925
24.15.5 Displaying supervisions of a ProDiag function block (S7-1500)....7926

24.16 Displaying supervised user-defined blocks (S7-1500)....7927
24.16.1 Displaying supervised user-defined blocks (S7-1500)....7927

24.17 Exporting and importing ProDiag supervisions and properties of a ProDiag FB (S7-1500)..7928
24.17.1 Overview of exporting and importing supervisions and properties of a ProDiag FB (S7-1500)....7928
24.17.2 Exporting supervisions and properties of a ProDiag FB (S7-1500)....7930
24.17.3 Importing supervisions and properties of a ProDiag FB (S7-1500)....7932

24.18 Exporting and importing assignments of ProDiag FBs to Instance DBs (S7-1500)....7935
24.18.1 Overview of exporting and importing assignments (S7-1500)....7935
24.18.2 Exporting assignments (S7-1500)....7935
24.18.3 Importing assignments (S7-1500)....7936

24.19 Displaying ProDiag cross-references (S7-1500)....7938
24.19.1 Overview of cross-references (S7-1500)....7938
24.19.2 Displaying cross-references (S7-1500)....7938

24.20 Acquiring initial values (S7-1500)....7940
24.20.1 Overview of initial value acquisition (S7-1500)....7940
24.20.2 Activating initial value acquisition (S7-1500)....7943
24.20.3 Example of order of the list (S7-1500)....7943

Index....7947

Programming basics

1

1.1 Operating system and user program

The following figure shows the interaction of the operating system and the user program:

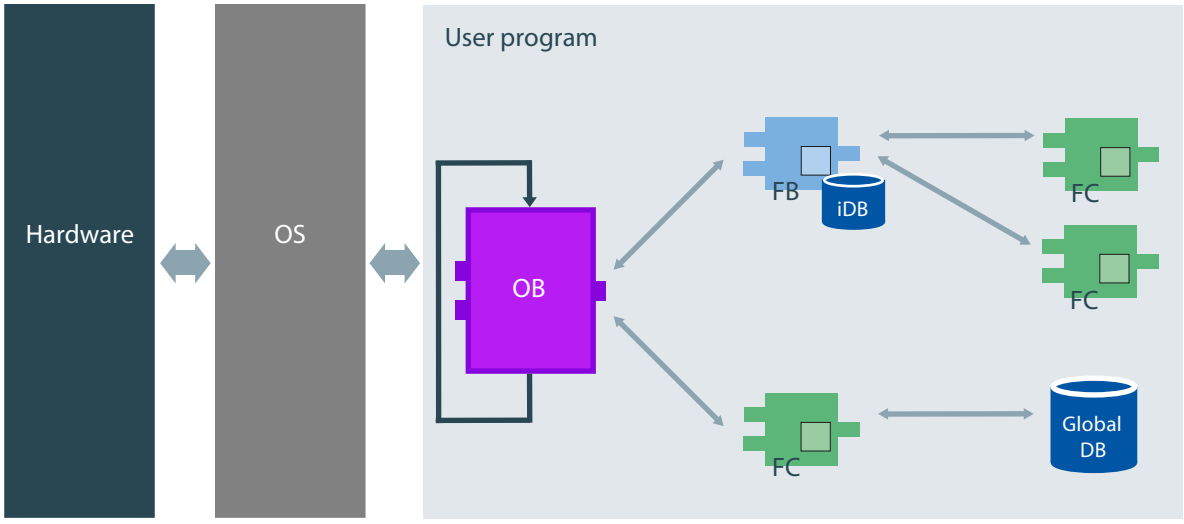

Operating system (OS)

The operating system is contained in every CPU and organizes all CPU functions and sequences that are not associated with a specific control task.

The tasks of the operating system, for example, include the following:

- Processing a warm restart
- Updating the process image of the inputs and outputs
- Calling the user program
- Detecting interrupts and calling interrupt OBs
- Detecting and handling errors
- Managing memory areas

The operating system is a component of the CPU and is already installed there upon delivery.

User program

The user program contains all functions that are necessary for processing your specific automation task.

The tasks of the user program include:

- Checking the requirements for a (warm) restart using startup OBs, for example, limit switch in correct position or safety relay active.
- Processing process data, e.g. linking binary signals, reading in and evaluating analog values, defining binary signals for output, and outputting analog values
- Reaction to interrupts, for example, diagnostic error interrupt if the limit value of an analog expansion module is overshot
- Error handling in normal program execution

You write the user program and load it into the CPU.

1.2 Blocks in the user program

1.2.1 Linear and structured programming

Linear programming

Solutions for small automation tasks can be programmed linearly in a program cycle OB. This is only recommended for simple programs.

The following figure shows a linear program schematically: The "Main" cycle OB contains the complete user program.

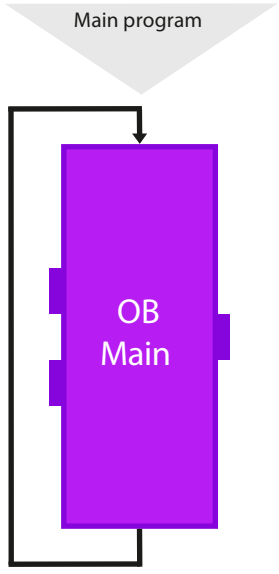

Structured programming

Complex automation tasks can be more easily handled and managed by dividing them into smaller sub-tasks that correspond to the technological functions of the process or that can be reused. These sub-tasks are represented in the user program by blocks. Each block is then an independent section of the user program.

Structuring the program offers the following advantages:

- Extensive programs are easier to program through the structure.
- Individual program sections can be standardized and used repeatedly with changing parameters.
- Program organization is simplified.
- Changes to the program can be made more easily.
- Debugging is simplified since separate sections can be tested.
- Commissioning is simplified.

The following figure shows a structured program schematically: The "Main" cycle OB calls subprograms in succession to execute defined subtasks.

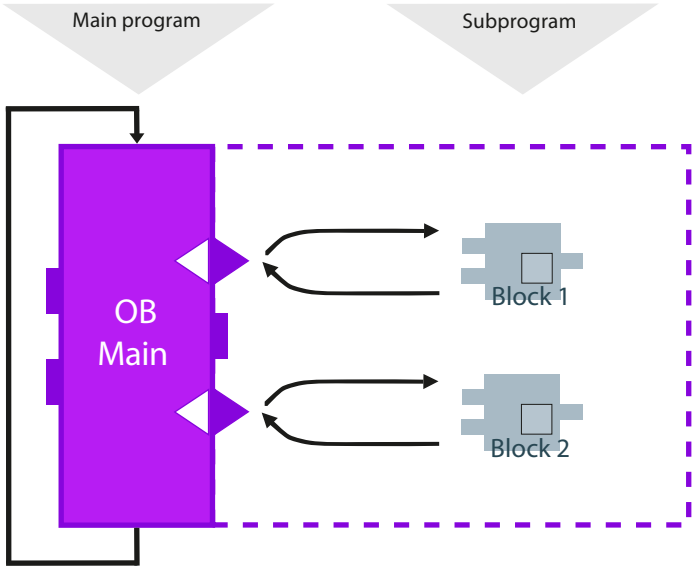

Nesting depth for blocks

The permissible nesting depth for blocks depends on the CPU that is used.

The following table shows the guide values for the maximum nesting depth. For detailed information on the CPU you are using, refer to the technical specifications in the hardware documentation. To call the hardware documentation on the Internet, click the links in the table.

CPU family	Nesting depth (guide value)	Link to hardware documentation
S7-1500	24 blocks per priority class	SIMATIC S7-1500 / ET 200MP Manual Collection (https://support.industry.siemens.com/cs/ww/en/view/86140384)
S7-1200	16 blocks from cycle or startup OB, 6 additional blocks within any interrupt event OB	SIMATIC S7-1200 Automation System (https://support.industry.siemens.com/cs/ww/en/view/91696622)
S7-400	24 blocks per priority class, 1-2 additional blocks within an error OB	SIMATIC S7-400 automation system S7-400 CPU data (https://support.industry.siemens.com/cs/de/en/view/53385241)
S7-300	16 blocks per priority class, 4 additional blocks within an error OB	SIMATIC S7-300 CPU 31xC and CPU 31x: Technical specifications (https://support.industry.siemens.com/cs/de/de/view/12996906/en)
ET 200SP	24 blocks	SIMATIC ET 200SP Manual Collection (https://support.industry.siemens.com/cs/ww/en/view/84133942)

Nesting depth for structures

Structures (STRUCT) and PLC data types (UDT) can be nested to a depth of 8. This nesting depth is independent of the CPU used.

1.2.2 Overview of the block types

Block types

Different BLOCK types are available to perform tasks within an automation system. The following table shows the available block types:

Block type	Brief description
Organization blocks (Page 39) (OB)	Organization blocks define the structure of the user program.
Functions (Page 40) (FC)	Functions contain program routines for recurring tasks. They have no "memory".
Function blocks (Page 41) (FB)	Function blocks are code blocks that store their values permanently in instance data blocks, so that they remain available even after the block has been executed.
Instance data blocks (Page 53)	Instance data blocks are assigned to a function block when it is is called for the purpose of storing program data.
Global data blocks (Page 42)	Global data blocks are data areas for storing data that can be used by any blocks.

1.2.3 Organization blocks (OB)

Definition

Organization blocks (OBs) form the interface between the operating system and the user program. They are called by the operating system and control, for example, the following operations:

- Startup characteristics of the automation system
- Cyclic program processing
- Interrupt-driven program execution
- Error handling

You can program the organization blocks and at the same time determine the behavior of the CPU. Various organization blocks are available to you depending on the CPU used.

For more information on organization blocks, refer to the descriptions of the modes of operation of CPUs in the "Additional information on configurations" chapter in "Configuring Hardware and Networks".

Start information of organization blocks

When certain organization blocks are started, the operating system provides information that can be evaluated in the user program. Refer to the descriptions of the organization blocks to find out which information is provided, if any.

See also

Creating organization blocks (Page 6908)

1.2.4 Functions (FCs)

Definition

Functions (FCs) are code blocks without memory. You have no data memory in which values of block parameters can be stored. Therefore, when a function is called, all formal parameters must be assigned actual parameters.
Functions can use global data blocks to store data permanently.

Application

A function contains a program that is executed when the function is called by another code block. Functions can be used, for example, for the following purposes:

- To return function values to the calling block, e.g. for mathematical functions
- To execute technological functions, e.g. individual controls using bit logic operations

A function can also be called several times at different points in a program. As a result, they simplify programming of frequently recurring functions.

Note

Parameter transfer when calling functions

To avoid errors when working with functions, observe the information in chapter "Parameter transfer at block call (Page 76)".

See also

Creating functions and function blocks (Page 6909)

1.2.5 Function blocks (FB)

Definition

Function blocks are code blocks that store their input, output and in-out parameters permanently in instance data blocks, so that they remain available even after the block has been executed. Therefore they are also referred to as blocks "with memory".

Function blocks can also operate with temporary tags. Temporary tags are will not be stored in the instance DB, but are available for one cycle only.

Application

Function blocks contain subroutines that are always executed when a function block is called by another code block. A function block can also be called several times at different points in a program. As a result, they simplify programming of frequently recurring functions.

Instances of function blocks

A call of a function block is referred to as an instance. An instance data block is required for each instance of a function block; it contains instance-specific values for the formal parameters declared in the function block.

The function block can store its instance-specific data in its own instance data block or in the instance data block of the calling block.

Access modes

S7-1200 and S7-1500 offer two different access options for the instance data blocks, which can be assigned to a function block when this is called:

- Data blocks with optimized access
  Data blocks with optimized access have no firmly defined memory structure. The data elements contain only a symbolic name in the declaration, no fixed address within the block.
- Data blocks with standard access (compatible with S7-300/400)
  Data blocks with standard access have a fixed memory structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block.

Note

To avoid errors when working with function blocks, refer to the section "Parameter transfer at block call (Page 76)".

See also

Creating functions and function blocks (Page 6909)

Multi-instances (Page 65)

Instance data blocks (Page 53)

Basics of block access (Page 55)

1.2.6 Global data blocks (DB)

Definition

Data blocks are used to store program data. Data blocks thus contain variable data that is used by the user program. Global data blocks store data that can be used by all other blocks.

The maximum size of data blocks varies depending on the CPU. You can define the structure of global data blocks anyway you please.

You also have the option of using PLC data types (UDT) as a template for creating global data blocks.

Global data blocks in the user program

Every function block, function, or organization block can read the data from a global data block or can itself write data to a global data block. This data remains in the data block even after the data block is exited. A global data block and an instance data block can be open at the same time.

The following figure shows the different accesses to data blocks:

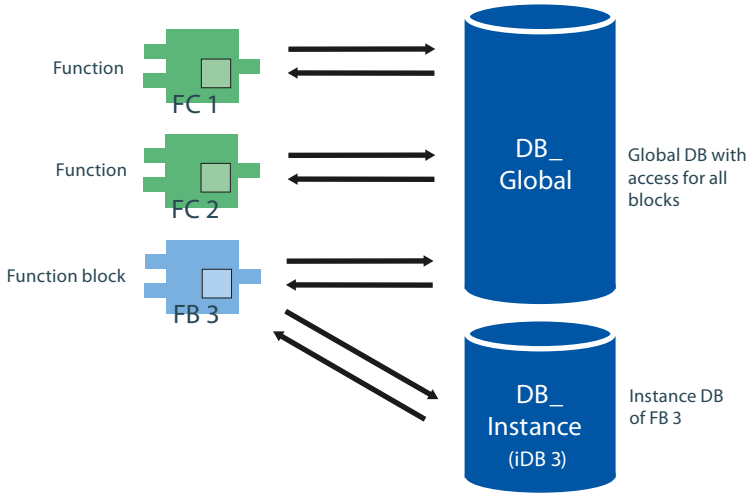

Access modes

S7-1200 and S7-1500 offer two different access options for global data blocks:

- Data blocks with optimized access
  Data blocks with optimized access have no fixed defined structure. In the declaration, the data elements are assigned only a symbolic name and no fixed address within the block.
- Data blocks with standard access (compatible with S7-300/400)
  Data blocks with standard access have a fixed structure. In the declaration, the data elements are assigned both a symbolic name and a fixed address within the block.

See also

Creating data blocks (Page 7023)

Basics of block access (Page 55)

1.2.7 Global ARRAY data blocks (DB)

ARRAY data blocks (S7-1500)

ARRAY data blocks are a particular type of global data block. These consist of an ARRAY of any data type. For example, an ARRAY of a PLC data type (UDT) is possible. The DB contains no other elements besides the ARRAY. Because of their flat structure, ARRAY data blocks facilitate access to the ARRAY elements and their transfer to called blocks.

The "Optimized block access" attribute is always enabled for ARRAY data blocks. ARRAY data blocks with standard access are not possible.

The "Instructions > Basic instructions" task card in the "Move > ARRAY DB" section offers extended options for addressing ARRAY DBs. These instructions give you the option, for example, to address the DB name indirectly:

- ReadFromArrayDB: Read from array data block
- WriteToArrayDB: Write to ARRAY data block
- ReadFromArrayDBL: Read from ARRAY data block in load memory
- WriteToArrayDBL: Write to array data block in load memory

See also

Example of the use of ARRAY data blocks (Page 44)

Addressing ARRAY DB (Page 109)

Creating data blocks (Page 7023)

1.2.8 Example of the use of ARRAY data blocks

Using an ARRAY data block (S7-1500)

ARRAY data blocks are global data blocks that consist exclusively of an ARRAY. A data block with a tag of the ARRAY data type is sufficient for most applications because access can be programmed intuitively with a tag of the ARRAY data type (for example, #myArray[#index]) and offer a better runtime performance than ARRAY data blocks. In certain scenarios, however, it is necessary to process ARRAYs with different lengths. The ARRAY data block is particularly suitable for these scenarios.

The following example shows how you can use an ARRAY data block.

Programming example

Individual pieces of material are transported on a conveyor belt. These pieces of material pass a scanner which can read out the information that a piece of material carries with it. The information is read out and transmitted to a panel. Because the scanner and the panel have different clock cycles/speeds, the material information must be cached in each case.

In the following programming example, we show you how to program the program code for passing on material information. For this purpose, you use an ARRAY data block.

The example is set up to be flexible, so that you do not have to know at the time the program code is created which ARRAY data block will be read or written or how large it is. You can thus use ARRAYs of different length. Use of data type DB_ANY gives you this flexibility.

The data type VARIANT is used so that you also have flexibility when specifying the value that is to be read or written.

Not until runtime is the ARRAY data block transferred in order to then access the values in the program block. The data type of the ARRAY elements and the data type of the value that is to be read or written are determined.

You can determine the number of objects and the fill level of the ARRAY data block using special instructions.

Procedure

Create the PLC data type "UDT_Queue". This PLC data type is used by both functions ("FC_Enqueue" and "FC_Dequeue"). This is important, for example, for accessing the tag #Queue.Used, as the function "FC_Enqueue" increments the tag by one and the function "FC_Dequeue" decrements the tag by one.

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare the following lines within the PLC data type:
   DB > Data type: DB_ANY
   Size > Data type: DINT
   Used > Data type: DINT
   ReadPos > Data type: DINT
   WritePos > Data type: DINT

UDT_Queue

		Name	Data type	Default value	Comment
1		DB	DB_ANY	0	
2		Size	DInt	0	
3		Used	DInt	0	
4		ReadPos	DInt	0	Pointer Reading Position
5		WritePos	DInt	0	Pointer Writing Position

Program the "FC_Enqueue" function, which writes the values of the material information to an ARRAY data block. The specific ARRAY data block and the data type of the value do not have

to known at this time, because the interfaces are being programmed with the VARIANT and DB_ANY data types.

1. Create an SCL function and name it "FC_Enqueue".
2. Declare the block interface as follows:

FC_Enqueue

	Name	Data type	Comment
1	Input		
2	Value	Variant	
3	Output		
4	InOut		
5	Queue	"UDT_Queue"	
6	DB	DB_ANY	
7	Size	DInt	
8	Used	DInt	
9	ReadPos	DInt	Pointer Reading Position
10	WritePos	DInt	Pointer Writing Position
11	Temp		
12	Error	Int	
13	<Add new>		
14	Constant		
15	Return		
16	FC_Enqueue	Int	

3. Write the following program code:

```
1  IF #Queue.Used < #Queue.Size THEN // As long as there is enough space in the DB
2                                    // the function keeps on writing.
3
4      #Error := WriteToArrayDB(db := #Queue.DB, index := #Queue.WritePos, value := #Value);
5
6      IF #Error = 0 THEN
7
8          #Queue.Used := #Queue.Used + 1;
9          #Queue.WritePos := #Queue.WritePos + 1;
10
11         IF #Queue.WritePos >= #Queue.Size THEN // If this condition is fullfilled
12                                                // the end of the DB is reached.
13                                                // End of writing pointer.
14
15             #Queue.WritePos := 0;
16
17         END_IF;
18
19     END_IF;
20
21 ELSE
22
23     #Error := 4711; // The queue is full.
24
25 END_IF;
```

With this function, you check whether there is still space free in the data block. If so, write the value that is specified at the parameter value into the data block at the parameter db. With each new item of material information that is written, the tag #Queue.Used and the pointer tag #Queue.WritePos are incremented by one. As soon as the cursor reaches the end of the data block, it is reset to 0. If the data block is full, the error code #4711 is returned.

Program the "FC_Dequeue" function, which reads the material information from an ARRAY data block and writes it to the panel. The specific ARRAY data block and the data type of the value do not have to known at this time, because the interfaces are being programmed with

the VARIANT and DB_ANY data types. The material information can then be displayed on a panel, for example:

1. Create an SCL function and name it "FC_Dequeue".
2. Declare the block interface as follows:

FC_Dequeue

	Name	Data type	Comment
1	Input		
2	Output		
3	InOut		
4	Value	Variant	
5	Queue	"UDT_Queue"	
6	DB	DB_ANY	
7	Size	DInt	
8	Used	DInt	
9	ReadPos	DInt	Pointer Reading Position
10	WritePos	DInt	Pointer Writing Position
11	Temp		
12	Error	Int	
13	Constant		
14	Return		
15	FC_Dequeue	Int	

3. Write the following program code:

```
1  IF #Queue.Used > 0 THEN // As long as there is some information in the DB
2                          // the function keeps on reading.
3
4      #Error := ReadFromArrayDB(db := #Queue.DB, index := #Queue.ReadPos, value => #Value);
5
6      IF #Error = 0 THEN
7
8          #Queue.Used := #Queue.Used - 1;
9          #Queue.ReadPos := #Queue.ReadPos + 1;
10
11         IF #Queue.ReadPos >= #Queue.Size THEN // If this condition is fullfilled
12                                               // the end of the DB is reached.
13                                               // End of reading pointer.
14
15             #Queue.ReadPos := 0;
16
17         END_IF;
18
19     END_IF;
20
21 ELSE
22
23     #Error := 4712; // The queue is empty.
24
25 END_IF;
```

With this function, you check whether material information is available in the data block. If so, read the value to which the pointer #Queue.ReadPos points, and write it to the tag #Value. With each item of material information that is read, the tag #Queue.Used is decremented by one and the pointer tag #Queue.ReadPos is incremented by one. As soon as the cursor reaches the end of the data block, it is reset to 0. If the data block is empty, the error code #4712 is returned.

To save the material data, create an ARRAY data block. Use the PLC data type "UDT_Material" as the data type of the ARRAY data block.

First, create the PLC data type "UDT_Material". This PLC data type contains a structure for the material information supplied by the scanner.

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare the following lines within the PLC data type:
   ArticleNumber > Data type: DINT
   ArticleName > Data type: STRING
   Amount > Data type: REAL
   Unit > Data type: STRING

UDT_Material

	Name	Data type
1	ArticleNumber	DInt
2	ArticleName	String
3	Amount	Real
4	Unit	String

Create the ARRAY data block "DB_MaterialBuffer". The ARRAY data block is to contain data records with material information of data type "UDT_Material". The material information is written to the ARRAY data block using the "FC_Enqueue" function.

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_MaterialBuffer".
4. Select the type of the data block "ARRAY DB".
5. Select the PLC data type "UDT_Material" as ARRAY data type.
6. Specify "1000" as high ARRAY limit.
7. Click "OK".

DB_MaterialBuffer

	Name	Data type
1	DB_MaterialBuffer	Array[0..1000] of "UDT_Material"
2	DB_MaterialBuffer[0]	"UDT_Material"
3	ArticleNumber	DInt
4	ArticleName	String
5	Amount	Real
6	Unit	String
7	DB_MaterialBuffer[1]	"UDT_Material"
8	DB_MaterialBuffer[2]	"UDT_Material"
9	DB_MaterialBuffer[3]	"UDT_Material"
10	DB_MaterialBuffer[4]	"UDT_Material"
11	DB_MaterialBuffer[5]	"UDT_Material"

Create the startup organization block (OB) "OB_MaterialQueue". In this organization block, you initialize the tags DB and Size.

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Organization block (OB)" button.
3. Specify the name "OB_MaterialQueue".
4. Select the type "Startup".
5. Select SCL as the language of the organization block.
6. Click "OK".
7. Write the following program code:

```
1  "DB_MaterialQueue".DB := "DB_MaterialBuffer";
2  "DB_MaterialQueue".Size := 1000;
3  "DB_MaterialQueue".Used := 0;
4  "DB_MaterialQueue".ReadPos := 0;
5  "DB_MaterialQueue".WritePos := 0;
```

By assigning the data block, you connect the ARRAY data block "DB_MaterialBuffer" with the SCL functions "FC_Enqueue" and "FC_Dequeue". Enter the size of the ARRAY data block at the Size parameter. The start value of the Used parameter is "0". The first item of material information is thus written to the ARRAY element "0".

1. Declare the following tags in the "Default tag table":

Name	Data type	Address
MaterialReceived	Bool	%I0.0
MaterialReceivedEdge	Bool	%M0.0
MaterialDone	Bool	%I1.0
MaterialDoneEdge	Bool	%M0.1

2. Call the SCL function "FC_Enqueue" within the function block in which the scanner reads out the material information.
3. Declare the tag "ConnectionToUDT" in the section "Temp" in the block interface and link this tag with the PLC data type "UDT_Material":

Temp	
ConnectionToUDT	"UDT_Material"
ArticleNumber	DInt
ArticleName	String
Amount	Real
Unit	String

4. Link the function call with the following tags and, at the enable input EN, create the signal edge "P: Scan operand for positive signal edge". Link the signal edge with the global tags from the standard tag table:

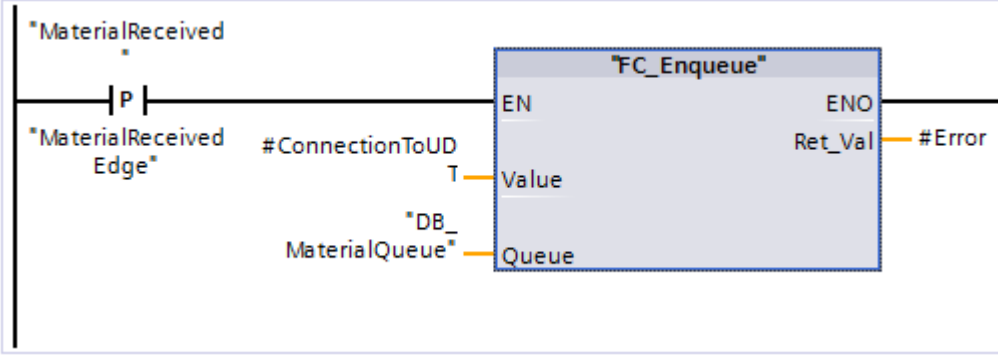

5. Call the SCL function "FC_Dequeue"
6. Link the function call with the following tags and, at the enable input EN, create the signal edge "P: Scan operand for positive signal edge". Link the signal edge with the global tags from the standard tag table:

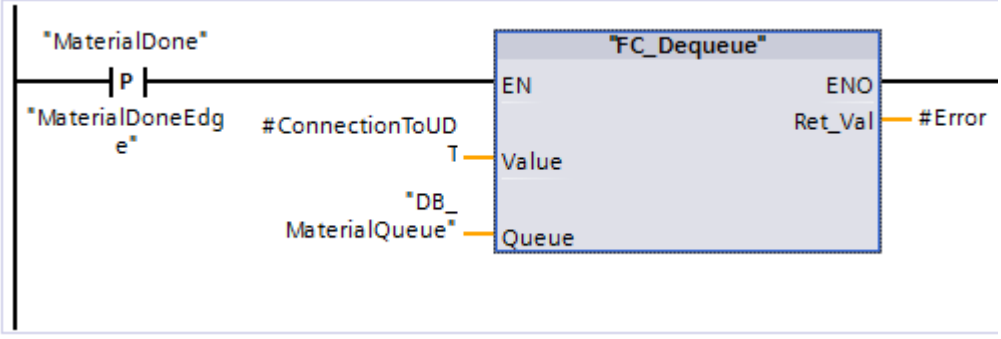

Result

As soon as a positive signal edge occurs, material information is written to an ARRAY data block with the help of the instruction "WriteToArrayDB" and sent to the panel with the help of the instruction "ReadFromArrayDB".

See also

Global ARRAY data blocks (DB) (Page 43)

Addressing ARRAY DB (Page 109)

1.2.9 Instance data blocks

Definition

The call of a function block is referred to as an instance. The data with which the instance works is stored in an instance data block.

The maximum size of instance data blocks varies depending on the CPU. The tags declared in the function block determine the structure of the instance data block.

Access modes

S7-1200 and S7-1500 offer two different access options for the instance data blocks, which can be assigned to a function block when this is called:

- Data blocks with optimized access
  Data blocks with optimized access have no firmly defined structure. The declaration elements contain only one symbolic name in the declaration, and no fixed address within the block.
- Data blocks with standard access (compatible with S7-300/400)
  Data blocks with standard access have a fixed structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block.

See also

Creating data blocks (Page 6910)

Basics of block access (Page 55)

Instances (Page 62)

Insert block calls in LAD (Page 7143)

Inserting block calls in FBD (Page 7206)

1.2.10 CPU data blocks

Definition

CPU data blocks are generated by the CPU at runtime. To this purpose, insert the "CREATE_DB" instruction into your user program. You can use the data block that is generated at runtime to save your data.

CPU data blocks are indicated by means of a small CPU icon in the "Program blocks" folder of an available node. You can monitor the values of the variables of a CPU data block in online mode, similar to those of a different data block type.

You cannot create CPU data blocks in your offline project.

Loading CPU data blocks

The CPU data block that the user program has generated by means of the "CREATE_DB" instruction is initially only available on the device in online mode. All CPU data blocks will be included with the other blocks the next time you perform a complete download from the device to the project. The CPU data blocks are marked with a small CPU icon in the process. However, you cannot upload these CPU data blocks to your device again.

Restrictions on CPU data blocks in the project

Once the CPU data blocks have been loaded into your offline project, you can open and view their content. However, note that the CPU data blocks in the project are write-protected. The CPU data blocks in the project are therefore subject to the following restrictions:

- You cannot edit CPU data blocks, or convert these into a different data block type.
- CPU data blocks cannot be assigned a know-how protection.
- You cannot change the programming language of a CPU data block.
- CPU data blocks cannot be compiled or downloaded to a device.

Comparing CPU data blocks

Once the CPU data blocks have been loaded into your offline project, you can run an offline/online comparison for the CPU DBs loaded. The comparison editor provides you with a corresponding overview of the differences. It is possible to synchronize the online and off-line version of CPU data blocks if differences are found, but not by downloading the offline version to the device.

Deleting CPU data blocks

You can delete CPU data blocks both from the project and from the CPU.

See also

Deleting CPU data blocks (Page 6933)

1.2.11 Blocks with optimized access

1.2.11.1 Basics of block access

Introduction

STEP 7 offers data blocks with different access options:

- Data blocks with optimized access (S7-1200/S7-1500)
- Data blocks with standard access (S7-300 / S7-400 / S7-1200 / S7-1500)

Within one program you can combine the two types of blocks.

Data blocks with optimized access

Data blocks with optimized access have no fixed defined structure. In the declaration, the data elements are assigned only a symbolic name and no fixed address within the block. The elements are saved automatically in the available memory area of the block so that there are no gaps in the memory. This makes for optimal use of the memory capacity.

Tags are identified by their symbolic names in these data blocks. To address the tag, enter its symbolic name. For example, you access the "Fill Level" tag in the "Data" DB as follows: `"Data".Fill Level`

Blocks with optimized access offers the following advantages:

- You can create data blocks with any structure without paying attention to the physical arrangement of the individual data elements.
- Quick access to the optimized data is always available because the data storage is optimized and managed by the system.
- Access errors, as with indirect addressing or from the HMI, for example, are not possible.
- You can define specific individual tags as retentive.
- Optimized blocks are equipped with a memory reserve by default which lets you expand the interfaces of function blocks or data blocks during operation. You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

Note

The "Optimized block access" attribute is always enabled for the following blocks and cannot be deselected.

- GRAPH blocks
- ARRAY data blocks

Data blocks with standard access

Data blocks with standard access have a fixed structure. In the declaration, the data elements are assigned both a symbolic name and a fixed address within the block. The address is shown in the "Offset" column.

Tags in these data blocks can be addressed in both symbolic and absolute form.

```
"Data".Fill Level
DB1.DBW2
```

Setting Retentivity for Optimized Access or Standard Access

If you define data as retentive, its values are retained even after a power failure or a network off. A retentive tag is not initialized after the hot restart but retains the value it had prior to the power failure. If a DB tag is defined as retentive, it is stored in the retentive memory area of the data block.

The options for setting the retentivity depend on the access type of the block.

- In data blocks with standard access, you cannot set the retentive behavior of individual tags. The retentivity setting is valid for all tags of the data block.
- In data blocks with optimized access you can define the retentive behavior of individual tags.
  For structured data type tags, the retentivity setting always applies to the entire structure. You cannot make any individual retentivity setting for separate elements within the data type.

Setting Addressing Options for Optimized Access or Standard Access

Blocks with optimized access permit only "type-safe" access. Type-safe access addresses tags by their symbolic name only. This means even changes to the block or the block interface will not result in inconsistencies in the program or access errors.

The following table shows the permitted addressing options for optimized data:

Addressing	Block with standard access	Block with optimized access
Symbolic addressing	x	x
Indexed addressing of ARRAYs		x
Slice access	x	x
Overlapping with AT	x	-
Absolute addressing	x	-
Indirect addressing via ANY	x	-
Indirect addressing via POINTER and VARIANT	x	with symbolic notation only

See also

Setting up block access (Page 57)

Addressing variables in data blocks (Page 106)

Creating blocks (Page 6907)

1.2.11.2 Setting up block access

Introduction

Block access is set up automatically when you create a block:

- Blocks created on CPUs of the S7-1200/1500 product range provide optimized access by means of a default setting.
- New blocks created on CPUs of the S7-300/S7-400 product range provide standard access by means of a default setting.

Access to a block that you copy or migrate to a CPU of a different product range is not converted automatically. However, in certain situations it may be useful to change block access in manual mode, e.g., in order to utilize the full functional scope of the CPU.

In most cases, you will have to recompile and load the program after block access has been converted.

NOTICE
Optimized block access for GRAPH blocks The "Optimized block access" attribute is always enabled for GRAPH blocks in S7-1500 and cannot be deselected.

Procedure

To set the block access, proceed as follows:

1. Open the "Program blocks" folder in the project tree.
2. Right-click on the block whose block access you want to change.
3. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the block opens.
4. Click "Attributes" in the area navigation.
5. Enable or disable the "Optimized block access" option.
6. Confirm your entries with "OK".

Restrictions and special features

As a matter of principle, you can only convert block access on CPUs of the S7-1200/1500 product range, as only these support the "optimized" access mode.

The following restrictions or special features apply in this context:

- Instance data blocks
  The block access of instance data blocks is always determined by the assigned function block and cannot be changed in manual mode. If you change the access mode on a function block, you also need to update the assigned instance data blocks. This update procedure adapts the access mode of the instance data block.
- System blocks and know-how protected blocks
  You cannot manually edit the block access for system blocks and know-how protected blocks.
- Organization blocks
  The start information of an OB with standard access is always stored in the first 20 bytes of the "Temp" section in the block interface. By contrast, the start information of an OB with optimized access is always written to the "Input" section. For this reason, the block interface of OBs will also change whenever you convert block access. Additional information is provided in the following sections.

Converting block access from "standard" to "optimized".

A block copied from the CPU of the S7-300/400 product range to a CPU of the S7-1200/1500 product range will initially retain the "standard" access mode. However, you can significantly increase the performance of program execution by using blocks with optimized access, which is why it may be useful to modify the access mode manually.

The blocks are adapted as follows in the course of conversion:

- Function blocks
  All interface parameters are assigned the "Non-retain" retentivity setting.
- Global data blocks
  The retentivity setting that was assigned centrally to the entire data block is transferred to the individual interface parameters. It is now possible to manipulate the retentivity setting of the various parameters.
  However, the following rule will still apply: For structured data type tags, the retentivity setting always applies to the entire structure. You cannot assign separate retentivity settings to the various elements within a structured data type. It therefore follows that you cannot assign individual retentivity settings to the tags of data blocks that are based on PLC data types.
- Organization blocks
  All interface parameters that are stored in the first 20 bytes of the "Temp" section will be deleted. New CPU-specific start information is entered in the "Input" section. Naming conflicts with user-defined interface parameters occurring in the process are resolved by renaming the user-defined interface parameters.

CAUTION

The conversion of the block access has the following consequences:

- Absolute addressing of the interface parameters of the block is no longer possible after conversion of block access to the "optimized mode.
  Example: `#L0.1` is no longer valid.
- Since conversion to the "optimized" block access mode of organization blocks also modifies the OB interface,

you may possibly have to adapt, recompile and load the program again due to these changes.

Converting block access from "optimized" to "standard".

If you want to copy or move a block from the CPU of the S7-300/400 product range to a CPU of the S7-1200/1500 product range, you first need to set the "standard" access mode.

The blocks are adapted as follows in the course of conversion:

- Function blocks and global data blocks.
  You can no longer set a retentivity in the function block. The corresponding setting is made in the instance data block.
  All interface parameters in the instance DB or global DB are assigned the same retentivity setting. The conversion is subject to the following rule:
  - If all interface parameters in the original block were retentive, the entire block will be retentive after conversion.
  - If all interface parameters in the original block were non-retentive, the entire block will be non-retentive after conversion.
  - If the interface parameters in the original block had different retentivity settings, the entire block will be non-retentive after conversion.
- Organization blocks
  All interface parameters stored in the "Input" section will be deleted. New CPU-specific start information is entered in the "Temp" section. This data is written to the first 20 bytes. Naming conflicts with user-defined interface parameters occurring in the process are resolved by renaming the user-defined interface parameters.

CAUTION

The conversion of the block access has the following consequences:

Since a conversion to "standard" block access mode might change the retentivity settings of the interface parameters, you may possibly have to adapt, recompile and load the program again due to these changes.

See also

Basics of block access (Page 55)

Specifying block properties (Page 6920)

1.3 Block calls

1.3.1 Principles of block calls

Function of block calls

For your blocks to be executed in the user program, they need to be called from another block.

When one block calls another block, the instructions of the called block are executed. Only when execution of the called block has been completed does execution of the calling block resume. The execution is continued with the instruction that follows on the block call.

The following figure shows the sequence of a block call within a user program:

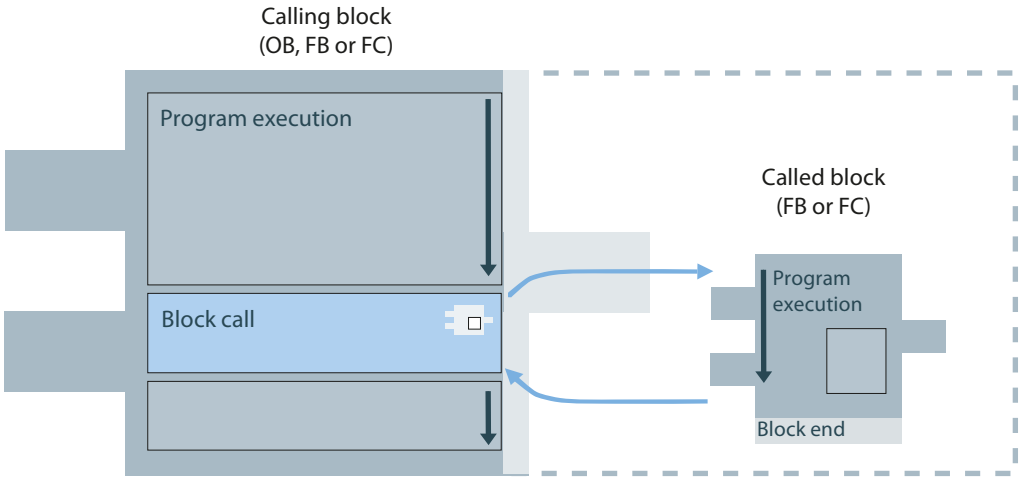

Parameter transfer

When a block is called, you must assign values to the parameters in the block interface. By providing input parameters you specify the data with which the block is executed. By providing the output parameters you specify where the execution results are saved.

See also

Call hierarchy (Page 61)

Basics of instances (Page 62)

Parameter transfer at block call (Page 76)

1.3.2 Call hierarchy

Definition

The order and nesting of block calls is referred to as the call hierarchy.

The following figure shows an example of the order and nesting of block calls within an execution cycle:

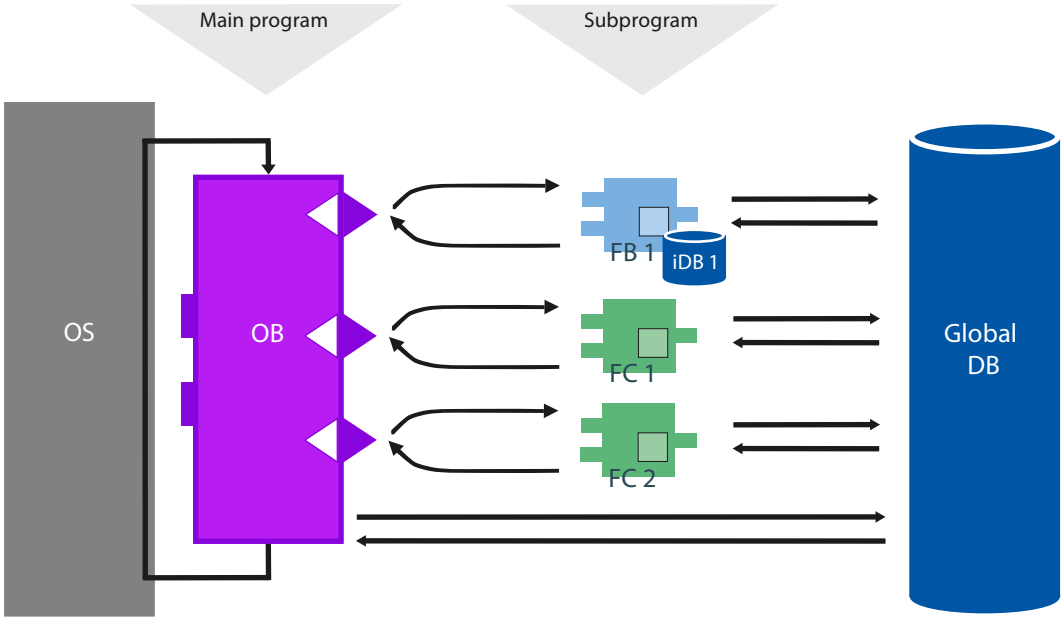

Nesting depth for blocks

The permissible nesting depth for blocks depends on the CPU that is used.

The following table shows the guide values for the maximum nesting depth. For detailed information on the CPU you are using, refer to the technical specifications in the hardware documentation. To call the hardware documentation on the Internet, click the links in the table.

CPU family	Nesting depth (guide value)	Link to hardware documentation
S7-1500	24 blocks per priority class	SIMATIC S7-1500 / ET 200MP Manual Collection (https://support.industry.siemens.com/cs/ww/en/view/86140384)
S7-1200	16 blocks from cycle or startup OB, 6 additional blocks within any interrupt event OB	SIMATIC S7-1200 Automation System (https://support.industry.siemens.com/cs/ww/en/view/91696622)
S7-400	24 blocks per priority class, 1-2 additional blocks within an error OB	SIMATIC S7-400 automation system S7-400 CPU data (https://support.industry.siemens.com/cs/de/en/view/53385241)

CPU family	Nesting depth (guide value)	Link to hardware documentation
S7-300	16 blocks per priority class, 4 additional blocks within an error OB	SIMATIC S7-300 CPU 31xC and CPU 31x: Technical specifications (https://support.industry.siemens.com/cs/de/de/view/12996906/en)
ET 200SP	24 blocks	SIMATIC ET 200SP Manual Collection (https://support.industry.siemens.com/cs/ww/en/view/84133942)

Nesting depth for structures

Structures (STRUCT) and PLC data types (UDT) can be nested to a depth of 8. This nesting depth is independent of the CPU used.

See also

Basics of instances (Page 62)

Principles of block calls (Page 60)

1.3.3 Instances

1.3.3.1 Basics of instances

Definition

After a function block is called, it needs memory for its working data. This data is referred to as an instance.

Instances have the following properties:

- Instances are always assigned to an FB.
- The structure of an instance is derived from the interface of the associated FB and can only be changed there.
- Instances are created automatically when a function block is called.

Storage of instances

You have the following options for storing instances:

- Single instance:
  The called function block saves its data in an instance data block of its own.
  See also: Single instances (Page 63)
- Multi-instance:
  The called function block does not store its data in an instance data block of its own, but rather in the instance of another function block.
  See also: Multi-instances (Page 65)
- Parameter instance:
  You transfer the instance of a function block to another block as an in-out parameter (InOut). This block can access the data in the transferred instance or call the associated FB.
  See also: Parameter instances (Page 68)

See also

Overview of the block interface (Page 6965)

Addressing instance data (Page 108)

Addressing variables in data blocks (Page 106)

Principles of block calls (Page 60)

Call hierarchy (Page 61)

1.3.3.2 Single instances

Definition

An instance of a function block (FB) that is located in its own instance DB is referred to as a single instance. The instance DB thus contains the working data for an individual block call.

Advantages

The use of single instances provides the following advantages:

- Reusability of the function blocks
- Good structuring options for simple programs

How single instances work

The following figure shows an FB that uses another FB ("FB_Valve"). "FB_Valve" is called as a single instance, i.e. it stores its data in its own instance DB.

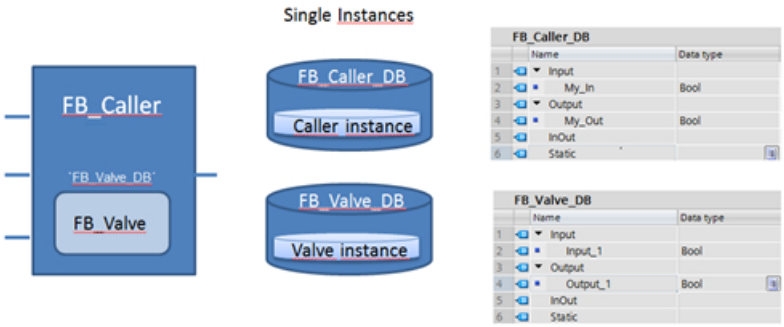

The structure of the instance data block is defined by the interface of the corresponding FB and can only be changed there. The instance DB contains the following data:

- Block parameters
  The block parameters in the "Input", "Output" and "InOut" sections form the interface of the block for the call in the program.
- Static local data
  Static local data in the "Static" section is used for permanently storing intermediate results beyond the current program cycle, for example for storing the signal state for an edge evaluation.

Creating single instances

You have the following options for creating a single instance:

- When you call an FB in the program, the "Call options" dialog appears. Here, you can specify whether you want to call the FB as a single instance, multi-instance or parameter instance.
- When you create a new data block, the "Add new block" dialog opens. Here, you can select a function block that you want to create an instance data block for under "Type".

Calling single instances

When a single instance is called, the assigned instance data block is indicated.

The following figure shows the call of the "Called_Block" block as a single instance in LAD. The instance data block "DB_Called_Block" is above the call.

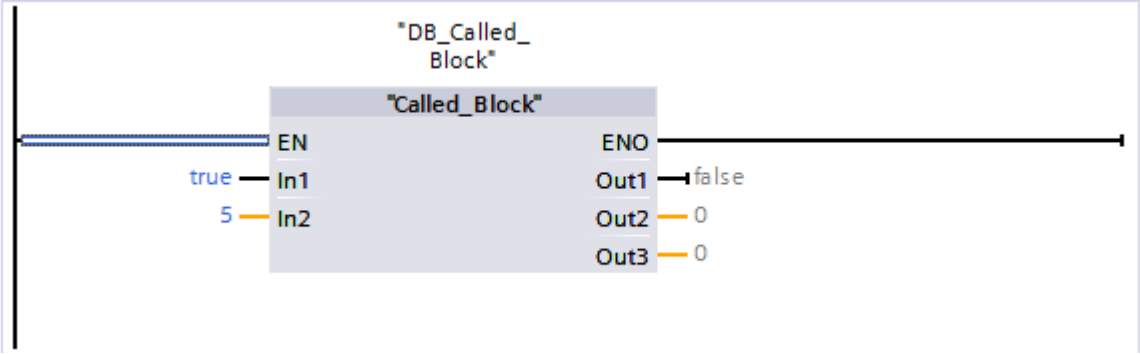

See also

Basics of instances (Page 62)

Multi-instances (Page 65)

1.3.3.3 Multi-instances

Definition

When a function block (FB) calls another FB, its instance data can also be stored in the instance DB of the calling FB. This type of block call is referred to as a multi-instance.

Advantages

The use of multi-instances provides the following advantages:

- Good structuring options for complex blocks
- Lower number of instance DBs
- Easy programming of local subprograms, for example for local timers or edge evaluations.

How multi-instances work

The following figure shows an FB that uses another FB ("FB_Valve"). "FB_Valve" is called as a multi-instance, i.e. it stores its data in the instance DB of the calling FB. Multi-instance data is located in the "Static" section of the calling block. In CPUs of the S7-1200/S7-1500 series, the instance can also be located in the instance DB of another function block.

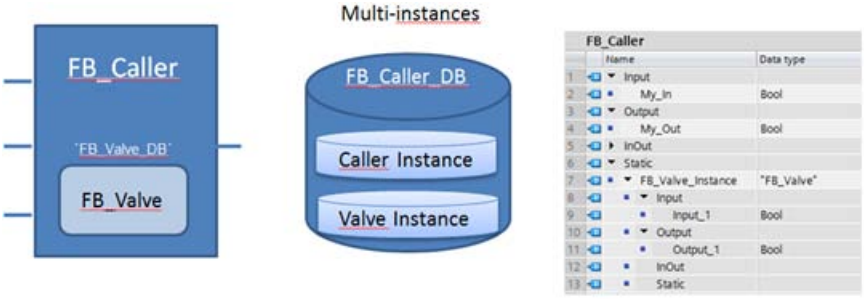

Creating multi-instances

You have the following options for creating a multi-instance:

- When you call an FB in the program, the "Call options" dialog appears. Here, you can specify whether you want to call the FB as a single instance, multi-instance or parameter instance.
- You declare the multi-instance directly in the interface of the calling block.
- You declare the multi-instance in the interface of another function block (S7-1200/S7-1500).

See also: Declaring multi-instances

ARRAYs of multi-instances

Multi-instances can also be created as an ARRAY. You can address the individual ARRAY elements using a variable index, for example when processing program loops.

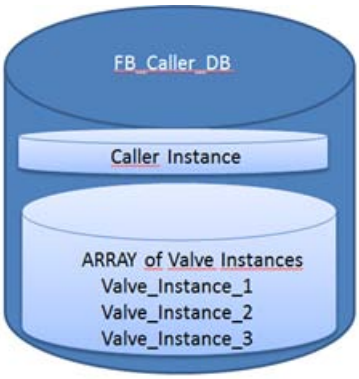

You declare ARRAYs of multi-instances directly in the interface of a function block.

Calling multi-instances

When a multi-instance is called, the assigned instance is indicated. The following figures show the call of the "Called_Block" block as a multi-instance in LAD.

In the following example, the instance is located locally in the instance DB of the calling block:

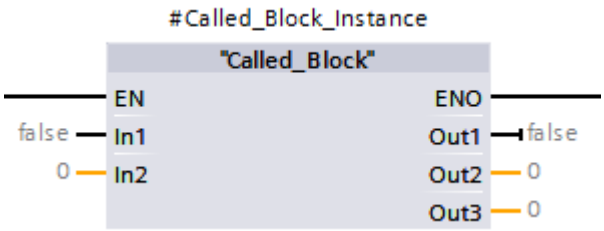

In the following example, the instance is located in the instance DB "MyOtherFB" (S7-1200/S7-1500):

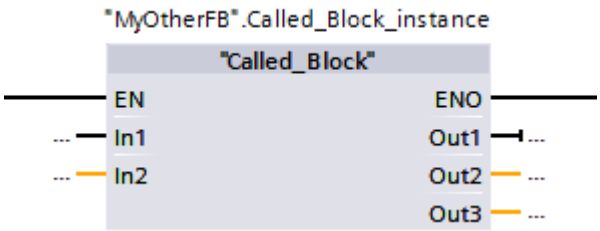

In the following example, the instance is located in an ARRAY of multi-instances in instance DB "MyOtherFB" (S7-1200/S7-1500):

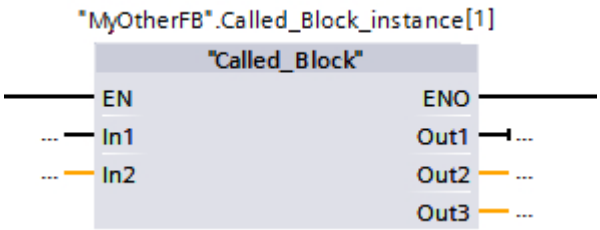

See also

Basics of instances (Page 62)

Single instances (Page 63)

Example of the use of parameter instances (Page 71)

1.3.3.4 Parameter instances

Definition

The parameter instance provides a special option for instantiation:

Here, you transfer the block instance to be used to the calling block as an in-out parameter (InOut) during runtime.

Advantages

The use of parameter instances provides the following advantages:

- At runtime, you can define which instance is currently being used.
- You can process different instances iteratively in program loops.

Note

ARRAYs of multi-instances

To simplify the iterative processing of instances in program loops, define an ARRAY of multi-instances.

See also: Multi-instances (Page 65)

How parameter instances work

The following figure shows the "FB_Caller", which uses another FB ("Valve"). An instance of "Valve" is transferred as a parameter instance. To do this, define an in-out parameter ("#Valve_Instance") that will be used to transfer the specific instance at runtime.

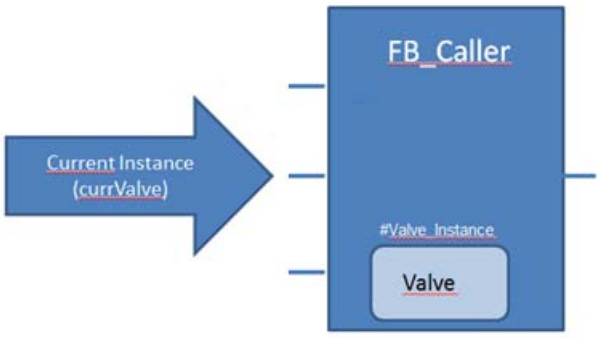

Creating parameter instances

Parameter instances are defined when a function block is called: When you call an FB, a dialog appears in which you can specify whether the FB will be called as a single instance, multi-instance or parameter instance. As an alternative, you also have the option of entering parameter instances manually directly in the block interface.

See also: Declaring parameter instances (Page 6985)

The following figure shows the call of a parameter instance for the FB "Valve":

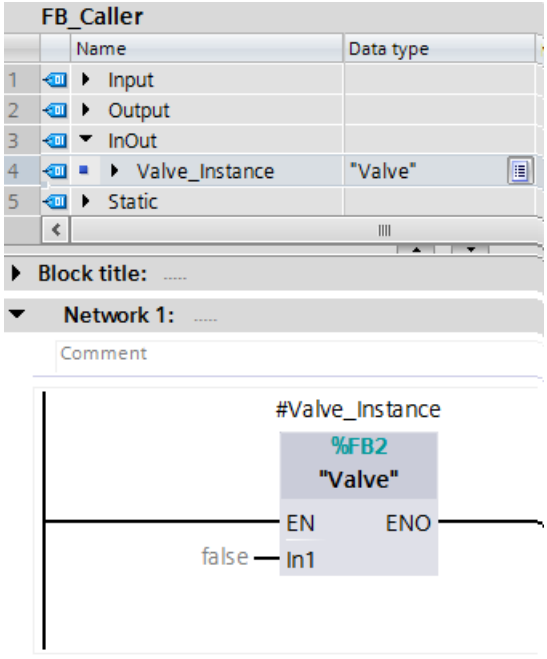

Transferring the instance as a parameter

Each time the higher-level FB is called ("FB_Caller" in the example) transfer an instance for the called FB (in the example "Valve"). You can transfer the following types of instances:

- Single instance
  Transfer an existing instance DB of the called FB.
- Multi-instance
  Transfer an existing multi-instance of the called FB.
- Individual element of an ARRAY of multi-instances
  Transfer an element of an existing ARRAY of multi-instances.

The following figure shows the call of "FB _Caller". With the "Valve_Instance" parameter, the instance "#currValve" is transferred. This instance, which is pending at the #currValve parameter during runtime, is used to call "Valve".

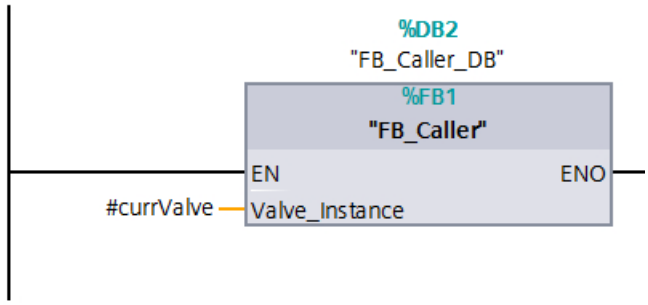

In CPUs of the S7-1200/S7-1500 series, you can also transfer instances as parameters that are located in the instance DB of another block.

In the following example, the transferred instance is located in instance DB "MyOtherIDB" (S7-1200/S7-1500):

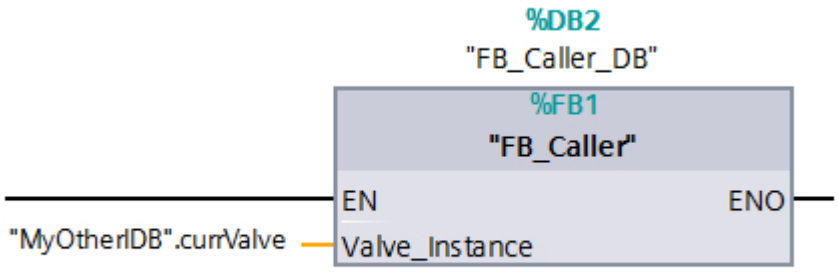

In the following example, the transferred instance is located in an ARRAY of multi-instances in instance DB "MyOtherIDB" (S7-1200/S7-1500):

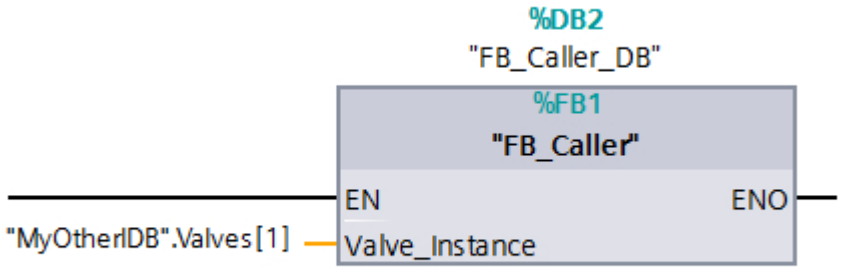

See also

Example of the use of parameter instances (Page 71)

1.3.3.5 Example of the use of parameter instances

Description

The use of parameter instances allows you to transfer the instance of a function block to another block (FB or FC) for further processing, for example, a data query or an error analysis, or even have the function block executed with the passed instance.

The ARRAY multi-instances can be used to combine objects of the same type and process their instances indexed in a program loop. Indexed ARRAY elements can also be passed to another block as a parameter instance.

To illustrate this, the object "Valve" (FB_Valve) was selected in the following example. On this basis, you are provided with all required information on valve processing in a program block, "FB_ValveControl" in this case.

The figure below illustrates the features needed and how they can be used:

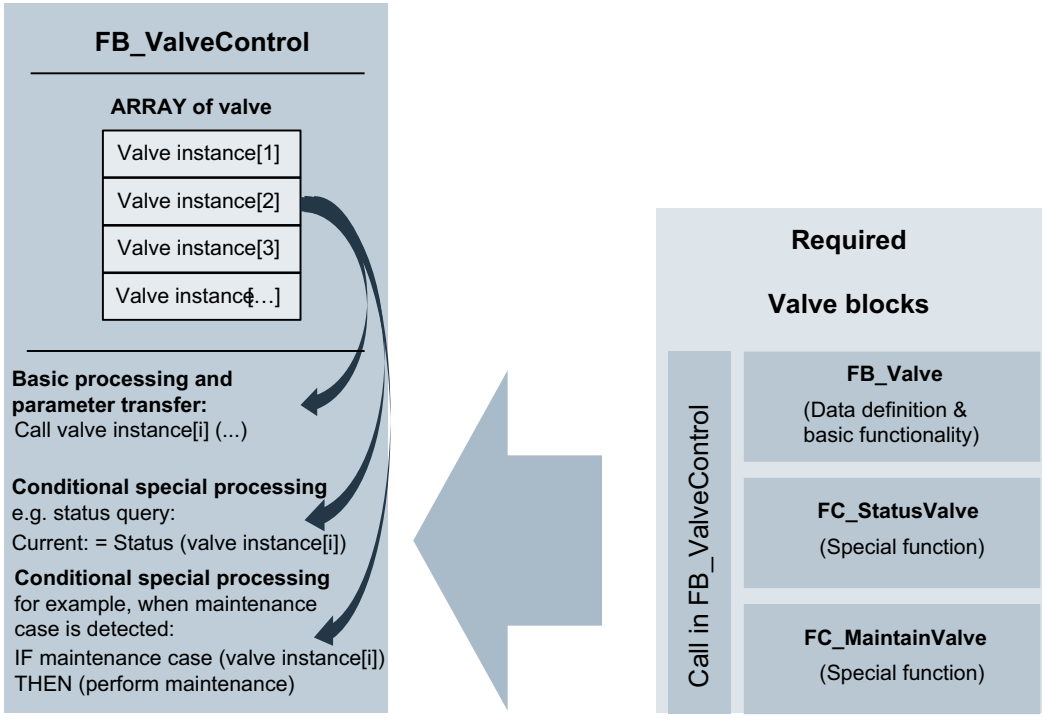

- The two functions "FC_StatusValve" and "FC_MaintainValve" process a parameter instance of the "FB_Valve" program block which they received in the call.
- In the "FB_ValveControl" program block, you use the ARRAY multi-instances on the one hand to declare the number of existing valves and, on the other hand, to manage all of the valves in a loop by means of various functions.

Note

Completeness

This solution is only an example to illustrate how this task can be approached. Keep in mind that you need to adjust the program code to your actual task.

The following program blocks are required to implement this example:

Block	Description	Programming language
FB_ValveCon-trol	This function block is used as a control block, where you can process all the valves using an ARRAY of multi-instances.	SCL
FB_Valve	This function block contains the definition of the valve data and the program code for processing a valve.	SCL (The three program blocks are used by the "FB_Valve-Control" program block.)
FC_StatusValve	The function returns the status of the currently pro-cessed valve.	
FC_Maintain-Valve	This function checks if maintenance is required on the valve. If required, the maintenance is performed and the function value TRUE is returned when everything is completed.	

Procedure: Create "FB_Valve"

To create the SCL function block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog opens.
2. Click on the "Function block (FB)" button.
3. Enter the name "FB_Valve".
4. Select "SCL" as the language.
5. Click "OK".

6. The declaration of the block interface is based on a valve which can open and close, and could look like this, for example:

FB_Valve			
	Name	Data type	Comment
1	Input		
2	R_max_A_Valve	Real	Maximum valve performance in mA
3	period_duration_factor	Int	Duration = "period_duration_factor" x 0.1 x configured duration
4	MANUAL_DO	Bool	Check output manually
5	SET_DO	Bool	Set output
6	Output		
7	I_actual_A	Real	Actual valve performance in mA
8	Current_Limit	Bool	Maximum valve performance exceeded (< 4A)
9	InOut		
10	SW_ENABLE	Bool	When this tag has the signal state TRUE, maintenance is required.
11	Static		
12	I_actual_mil	Int	The actual value of the program block "ValveControl", that is queried every millisecond.
13	control_byte	Byte	Control Byte: Bit0=SW_ENABLE; Bit1=MANUAL_DO; Bit2=SET_DO

7. The program code for controlling the valves could look like this, for example:

```
1  #I_actual_mil := #period_duration_factor;
2  #SW_ENABLE := #control_byte.%X4;
```

Procedure: Create "FC_StatusValve"

To create the SCL function "FC_StatusValve", follow these steps:

1. Double-click on the "Add new block" command.
   The "Add new block" dialog opens.
2. Click on the "Function (FC)" button.
3. Enter the name "FC_StatusValve".
4. Select "SCL" as the language.
5. Click "OK".
6. The declaration of the block interface with integration of the parameter instance "FB_Valve" could look like this, for example:

FC_StatusValve			
	Name	Data type	Comment
1	Input		
2	Output		
3	InOut		
4	CurrentValve	"FB_Valve"	The function block "FB_Valve" is used as a parameter instance to exchange its data with the "FB_Valve".
5	<Add new>		
6	Temp		
7	Constant		
8	Return		
9	FC_StatusValve	Byte	Return value of the function

7. Write the following program code, for example:

```
1  #FC_StatusValve := #CurrentValve.control_byte;
```

Procedure: Create "FC_MaintainValve"

To create the SCL function "FC_MaintainValve", follow these steps:

1. Double-click on the "Add new block" command.
   The "Add new block" dialog opens.
2. Click on the "Function (FC)" button.
3. Enter the name "FC_MaintainValve".
4. Select "SCL" as the language.
5. Click "OK".
6. The declaration of the block interface with integration of the parameter instance "FB_Valve" could look like this, for example:

FC_MaintainValve

	Name	Data type	Comment
1	Input		
2	NewValue	Int	Specifies a new value for the duration of the current valve.
3	Output		
4	InOut		
5	CurrentValve	"FB_Valve"	The function block "FB_Valve" is used as a parameter instance to exchange its data with the "FB_Valve".
6	Temp		
7	Constant		
8	Return		
9	FC_MaintainValve	Bool	Return value of the function

7. Write the following program code, for example:

```
IF #CurrentValve.SW_ENABLE THEN // Valve control enabled
    // Do some tuning and maintenance
    #CurrentValve.period_duration_factor := #NewValue;
    #CurrentValve.control_byte.%X0 := true;
    #FC_MaintainValve := true;
ELSE
    // Tuning currently not allowed
    #FC_MaintainValve := false;
    ;
END_IF;
```

Procedure: Create "FB_ValveControl"

To create the SCL function block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog opens.
2. Click on the "Function block (FB)" button.
3. Enter the name "FB_ValveControl".
4. Select "SCL" as the language.
5. Click "OK".

6. Define the user constant:

	Name	Data type	Value	Comment
1	vmax	Int	99	Number of the valves

The user constant "vmax" is used in this example in order to easily adapt the program to a different number of valves.

7. The declaration of the block interface with integration of the valve instances from "FB_Valve" could look like this, for example:

FB_ValveControl			
	Name	Data type	Comment
1	Input		
2	ValueMaintain	Int	Maintenance
3	Duration	Int	Duration of the valve control
4	Output		
5	Bypass	Bool	True or false
6	InOut		
7	Index	Int	Number index
8	Static		
9	ValveInstances	Array[0.."vmax"] of "FB_Valve"	Multiple instances of the valve
10	ValveStatus	Array[0.."vmax"] of Byte	Status of the valve
11	CurrentAmpere	Real	Current number of amperes

8. Write the following program code, for example:

```
1  // Set #Index, given from outside, to a value that adresses an existing valve.
2  #Index := (#Index + 1) MOD ("vmax" + 1);
3
4  // Indirect call of a valve that is addressed via an ARRAY index.
5  #ValveInstances[#Index](R_max_A_Valve := 1.0,
6                          period_duration_factor := #Duration,
7                          MANUAL_DO := 0,
8                          SET_DO := 0,
9                          I_actual_A => #CurrentAmpere,
10                         Current_Limit => #Bypass);
11
12 // Do some checks and tuning on the valve.
13 // FC_MaintainValve sets some values of the passed valve.
14 // FC_StatusValve returns the current status that is stored as internal value in the Instance.
15 IF "FC_MaintainValve"(CurrentValve := #ValveInstances[#Index],
16                       NewValue := #ValueMaintain) THEN
17     #ValveStatus[#Index] := #ValveStatus[#Index] XOR "FC_StatusValve"(CurrentValve := #ValveInstances[#Index]);
18 END_IF;
```

Procedure: Call of "FB_ValveControl" in OB 1

To call the "FB_ValveControl" function block in OB 1, follow these steps:

1. Open the "Main [OB1]" block with a double-click.
2. Use a drag-and-drop operation to add the "FB_ValveControl" function block to OB1.

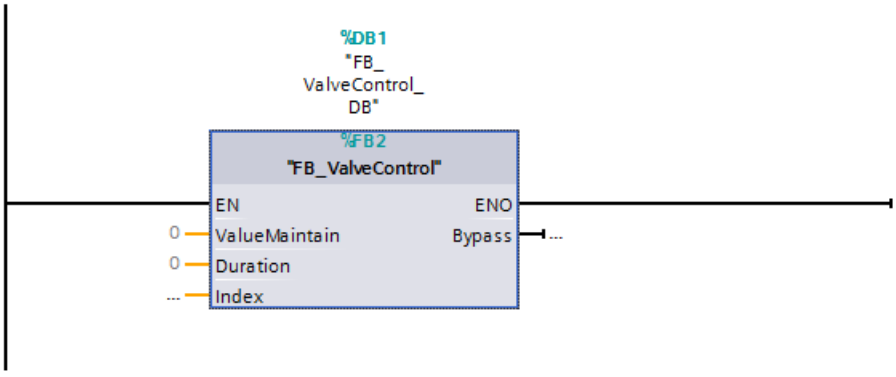

Result

One valve is processed in each program cycle. The respective valve selected by index is processed, its status queried and maintenance is performed, if needed.

In this example, one valve per cycle has been processed. Of course, it is also possible to process all valves in one program loop, but then the cycle time increases accordingly. If you need a short program cycle, we recommend that you avoid working with a program loop.

See also

Parameter instances (Page 68)

Multi-instances (Page 65)

1.3.4 Parameter transfer at block call

1.3.4.1 Rules for supplying block parameters

Block parameters

The calling block gives the called block the values with which it is to work. These values are referred to as block parameters. The input parameters provide the called block with the values that it has to process. The block returns the results via the output parameters.

Block parameters therefore form the interface between the calling and the called block.

You use input parameters when you want to only query or read values, and output parameters when you want to set or write these values. If block parameters are read and written you have to create these as in-out parameters.

The following rules apply to the use of block parameters within the block:

- Input parameters may only be read.
- Output parameters may only be written.
- In/out parameters may be read and written.

Formal and actual parameters

The block parameters are defined in the interface of the called block. These parameters are referred to as formal parameters. They are placeholders for the parameters that are transferred to the block when it is called. The parameters transferred to the block when it is called are referred to as actual parameters.

The data types of actual and formal parameters must be identical or convertible according to the rules of data type conversion.

Note

S7-1200/1500: Transferring peripheral inputs or outputs as block parameters

If you supply an input parameter with a peripheral input or output, it is possible that I/O access errors will occur during runtime when the block is called, e.g. a read error when an input module is accessed directly.

The system response of S7-1500 series CPUs with firmware version V2.1 or higher is as follows:

The block is called and processed with the substitute value of the signal.

The system response of S7-1200 and S7-1500 series CPUs with firmware versions lower than V2.1 is as follows:

The block is not called as a result of the I/O access error. Program execution is continued after the block call. If OB 122 exists or local error handling is enabled, these are executed.

To prevent the situation that the block is not called as a result of an I/O access error, first copy the peripheral input or output to a local tag (Temp) and transfer it as a block parameter to the called block.

The FAQ below provide additional information on this topic:

FAQ 89377245: Why are blocks in the S7-1200/S7-1500 not processed when they are addressed with unavailable PROFINET components? (https://support.industry.siemens.com/cs/ww/en/view/89377245)

See also

Using tags within the program (Page 95)

Parameter assignment to function blocks (Page 80)

Parameter assignment to functions (Page 78)

Rules for declaring the block interface (Page 6967)

PLC data types (UDT) (Page 273)

STL Basics (Page 7245)

1.3.4.2 Parameter assignment to functions

Parameters of functions (FC)

Functions have no data memory in which values of block parameters can be stored. Therefore, when a function is called, all formal parameters must be assigned actual parameters.

Input parameters (Input)

Input parameters are only read once before each block call. Therefore, the rule is that writing an input parameter within the block does not affect the actual parameter. Only the formal parameter is written.

Output parameters (Output)

Output parameters are only read once after each block call. Therefore, the rule is that output parameters should not be read within the block. If you nevertheless read an output parameter, please note that only the value of the formal parameter is read. The value of the actual parameter cannot be read within the block.

If an output parameter of a function is not written in this function, the value that is predefined for the specified data type is used. For example, the value "false" is predefined for BOOL. However, structured output parameters are not pre-assigned with a value.

To prevent unintentional further processing of the predefined value or an undefined value, note the following when programming the block:

- Make sure that the output parameters are written with values for all possible program paths within the block. In doing so, note that jump commands may skip instruction sequences in which outputs are set, for example.
- Note that the set and reset commands are dependent on the result of the logic operation. If the value of an output parameter is determined with these commands and RLO = 0, a value will not be generated.
- If possible, assign a default value for the output parameters of functions.

In/out parameters (InOut)

In/out parameters are read before the block call and written after the block call. If you read or write the parameter within the block, you only access its formal parameter.

An exception is in/out parameters with a structured data type. Structured data types consist of several data elements, for example ARRAY or STRUCT. These are transferred to the called block by a pointer. You therefore always access the actual parameter when you read or write a structured in/out parameter within a block.

When an in/out parameter of a function is not written to this function, the old output value or the input value is used as a value. Nevertheless, you should observe the information provided above for output parameters so that old values are not inadvertently processed further.

Temporary local data (Temp)

Temporary local data is only available during block processing. It is treated differently depending on the optimization setting of the block:

- Standard access
  The following rule applies to code blocks with standard access as well as to all tags with retentivity setting "Set in IDB":
  If you are using temporary local data, you must ensure that the values are initialized prior to use. Otherwise, the values will be random. Temporary data of the STRING of WSTRING data type is an exception: It is automatically pre-assigned the actual length 0.
- Optimized access
  The following rules apply to code blocks with optimized access:
  - If a temporary tag is not written within a function, the value that is predefined for the specified data type is used.
    The following table shows some examples of pre-defined values:

Data type	Pre-defined value
Bool	False
Int	0
REAL	0.0
Char	' '
Wchar	WCHAR#' '
DTL	DTL#1970-01-01-00:00:00
DATE_AND_TIME	DT#1990-01-01-00:00:00
Date	D#1990-01-01

  - Elements of the PLC data types are pre-assigned with the default value that is specified in the declaration of the PLC data type (UDT).
  - STRINGs and WSTRINGs are pre-assigned with the actual value "0", even if they are used within a PLC data type.
  - Elements of the data type ARRAY are pre-assigned with the value "0", even if they are used within a PLC data type.

Function value (Return)

Functions normally calculate a function value. This function value can be returned to the calling block via the RET_VAL output parameter. For this, the RET_VAL output parameter must be declared in the interface of the function. RET_VAL is always the first output parameter of a function. All data types are permitted for the RET_VAL parameter except ARRAY and STRUCT, as well as parameter types TIMER and COUNTER.

In the SCL programming language functions can be call directly in an expression. The result of the expression is then formed with the calculated function value. Therefore, the data type ANY is not permitted in SCL for the function value.

See also

Parameter assignment to function blocks (Page 80)

Rules for supplying block parameters (Page 76)

Calling functions (Page 7345)

Examples for calling functions in SCL (Page 7348)

1.3.4.3 Parameter assignment to function blocks

Supplying parameters of function blocks (FB)

In the case of function blocks the parameter values will be stored in the instance data.

If the input, output, or in-out parameters of a function block were not assigned with current values, the stored values are used when the parameter has already been initialized once in a previous cycle. If this is not the case, the CPU goes to STOP with a runtime error.

In some cases, it is mandatory to specify an actual parameter.

The following table shows which parameters of a function block must be assigned actual parameters:

Parameter	Elementary data type	Structured data type	Parameter type
Input (Input)	Optional	Optional	Optional
Output (Output)	Optional	Optional	Not permitted
In-out (InOut)	Optional	• Call of non-optimized block: not absolutely necessary, but the value must be initialized once by a call, otherwise a runtime error is output. • Call of optimized blocks: required	Not permitted

See also

Rules for supplying block parameters (Page 76)

Parameter assignment to functions (Page 78)

Parameter types (Page 348)

1.3.4.4 Transfer parameter as copy or as pointer

Introduction

When a block is called, you transfer data to the parameters in the block interface. At the input parameters, you transfer the data with which the block is to work. At the output parameters, you specify where the results of the processing are saved. In/out parameters are used to transfer data to the called block as well as to return results.

Internally, STEP 7 recognizes two different methods of parameter transfer: The data is transferred either as pointer or as copy, depending on the transfer range and data type of the parameter.

Transfer as copy (Call by value)

During a block call, the value of the operand is copied to the input parameter of the called block. With function blocks, the copy is stored in the instance DB; with functions, it is stored in the block stack. Additional storage space is required for the copy.

This means that the called block always works with the value that the specified operand had at the time of the block call. It cannot access the operand directly. Write access modifies only the copy, but not the actual value of the specified operand. Read access reads only the copy that was created at the time of the block call.

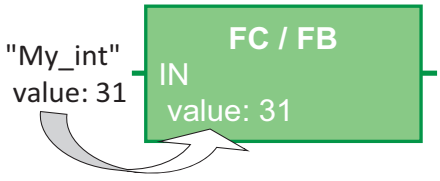

Transfer as pointer (Call by reference)

The parameters are referenced via a pointer during the block call.

This means that the called block can directly access the memory address of the operand that is specified as parameter: Write access directly results in the changing of the specified operand. Read access reads the value of the operand directly at the time of access. As no copy is created, no additional memory is required.

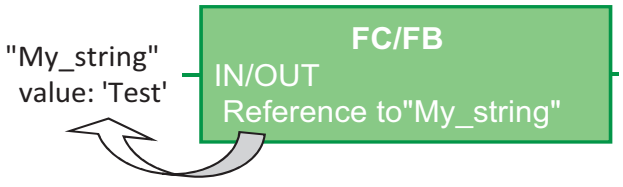

Note

Declare structured data types in the "InOut" area

If possible, use the "InOut" area in the block interface for structured tags (e.g., of data type ARRAY, STRUCT, STRING, …). Because structured in/out parameters by default are transferred as pointers, the required data memory is not increased unnecessarily.

Parameter transfer with S7-1200/1500

The following table shows how block parameters with elementary or structured data type are transferred in S7-1200/1500. Elementary data types are, for example, BOOL, INT or BYTE. Structured data types are, for example, ARRAY, STRUCT or STRING.

		Elementary data types	Structured data types
FC	Input	Copy	Pointer
	Output	Copy	Pointer
	InOut	Copy	Pointer
FB	Input	Copy	Copy
	Output	Copy	Copy
	InOut	Copy	Pointer

Note

Parameter transfer between blocks with optimized access and blocks with standard access

If structures are transferred to the called block as in/out parameters during a block call, they are transferred by default as pointers (Call by reference).

However, this does not apply if the two blocks have different optimization settings: If one of the blocks has the property "Optimized access" and the other block has the property "Standard access", all parameters are always transferred as copies (Call by value).

If the block contains numerous structured parameters, this can quickly lead to the temporary memory area (local data stack) overflowing.

Problems can also occur if the original operands are changed by asynchronous processes, such as by HMI access or interrupt OBs. If the copies are copied back to the original operands after the block processing, the changes carried out asynchronously since the creation of the copy during the block call are hereby overwritten at the original operands.

You can avoid this by setting the same access type for both blocks or by having the asynchronous access written first to a separate memory area and then copying this area synchronously at a suitable time.

See also:

Blocks with optimized access (Page 55)

FAQ 109478253: How can it happen that data of the HMI system or of the Web server is overwritten in the S7-1500? (https://support.industry.siemens.com/cs/de/en/view/109478253)

Parameter transfer with S7-300/400

The following table shows how block parameters with elementary or structured data type are transferred in S7-300/400.

		Elementary data types	Structured data types
FC	Input	Copy*	Pointer
	Output	Copy*	Pointer
	InOut	Copy*	Pointer
FB	Input	Copy	Copy
	Output	Copy	Copy
	InOut	Copy	Pointer

* Exception: Operands from the memory areas I, Q, M, P, L and partly qualified DB addresses (for example, "DW 2") are transferred as pointer.

Note

Special aspects of transfer as pointer in S7-300/400

In cases in which the parameters are transferred via a pointer, it is not possible to forward output parameters or in/out parameters from the calling block to the input parameters of the called block.

1.3.4.5 Forwarding of block parameters

Basic information on forwarding block parameters

Definition

The "Forwarding" of block parameters is a special type of parameter use. In this case the block parameters of the calling block are forwarded to the parameters of the called block. The called block uses the values that are currently present at the block parameters of the calling block as the actual parameters.

The following figure shows how the parameters of the function FC 10 are forwarded to the function FC 12:

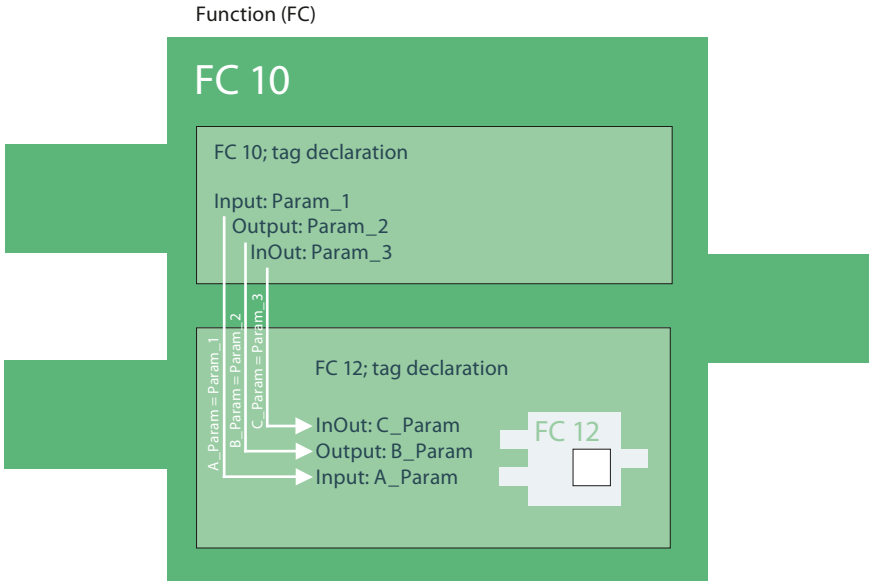

Rules for LAD/FBD

The following general rules apply in LAD and FBD:

- Input parameters can only be forwarded to input parameters.
- Output parameters can only be forwarded to output parameters.
- In/out parameters can be forwarded to all parameter types.
- In S7-300/400, the two block parameters must have the same data type.
- In S7-1200/1500, the parameters can also be converted according to the rules of implicit conversion.

Rules for STL

The following general rules apply in STL:

- Input parameters can only be forwarded to input parameters.
- Output parameters can only be forwarded to output parameters.
- In/out parameters can be forwarded to all parameter types.
- Both block parameters must have the same data type. In STL, this rule applies to all CPU families.

Rules for SCL

The rules for SCL are less stringent. So that programs from previous SCL versions can be taken over more easily, additional parameter transfer options are permissible. You can, for example, forward a structured in/out parameter to an input parameter. However, in this case you must ensure that the actual parameter is not located in the temporary local data or in a global data block.

Additional rules are described in detail in the following chapters.

See also

Calling a function by another function (Page 85)
Call of a function by a function block (Page 86)
Call of a function block by a function (Page 86)
Call of a function block by another function block (Page 87)

Calling a function by another function

Permissible data types for the call of a function by another function

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FC calls FC		Data types						
Actual parameter (calling block)	Formal parameter (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any	REF_TO
Input	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 as of V2 S7-1500	S7-1500
Output	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-	-
InOut	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-	S7-1500
InOut	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-	-
InOut	InOut	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-	-

See also

Basic information on forwarding block parameters (Page 83)

Call of a function by a function block

Permissible data types for the call of a function by a function block

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FB calls FC		Data types						
Actual parameter (calling block)	Formal parameter (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any	REF_TO
Input	Input	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 as of V2 S7-1500	S7-1500
Output	Output	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	-	S7-1200 S7-1500	-	-	-
InOut	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-	S7-1500
InOut	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-	-
InOut	InOut	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-	-

See also

Basic information on forwarding block parameters (Page 83)

Call of a function block by a function

Permissible data types for the call of a function block by a function

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FC calls FB		Data types					
Actual parameter (calling block)	Formal parameters (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any

Input	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-300/400 S7-1500	S7-1200 as of V2 S7-1500
Output	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-
InOut	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	InOut	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

See also

Basic information on forwarding block parameters (Page 83)

Call of a function block by another function block

Permissible data types for the call of a function block by another function block

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FB calls FB		Data types					
Actual parameter (calling block)	Formal parameters (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any
Input	Input	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-300/400 S7-1500	S7-1200 as of V2 S7-1500
Output	Output	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

InOut	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	InOut	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

See also

Basic information on forwarding block parameters (Page 83)

1.4 Using and addressing operands

1.4.1 Basic information about operands

Introduction

When you program instructions you must specify which data values the instruction should process. These values are referred to as operands. You can, for example, use the following elements as operands:

- PLC tags
- Constants
- Tags in instance data blocks
- Tags in global data blocks

Absolute address and symbolic name

Operands are identified by means of an absolute address and a symbolic name. You define the names and addresses in the PLC tag table or in the tag declaration of the blocks.

Data blocks with optimized access (S7-1200, S7-1500)

Data elements in data blocks with optimized access only receive a symbolic name and no absolute address in the declaration.

See also

Displaying symbolic and absolute addresses (Page 6844)

Basics of block access (Page 55)

1.4.2 Keywords

SIMATIC recognizes a range of key words whose definitions are fixed and which have a certain meaning in the program. You should not use these keywords as names for tags or constants.

Table of keywords

The following table shows all keywords.

Keywords German mnemonics	Keywords English mnemonics	Description
&	&	And logical operation of logical expressions
^	^	Operator for reading or writing the value of a reference
A	Q	Output, bit
A1	CC1	Condition code bit
A0	CC0	Condition code bit
AB	QB	Output, byte
AD	QD	Output, double word
AND	AND	And logical operation of logical expressions
ANY	ANY	Data type, pointer
AR1	AR1	Address Register 1
AR2	AR2	Address Register 2
ARRAY	ARRAY	Introduces the specification of an array and is followed by the index list between "[" and "]"
AT	AT	Overlaying tag declaration
AUTHOR	AUTHOR	Name of the author, company name, department name, or other name (max. 8 characters, no spaces)
AW	QW	Output, word
B	B	Byte
BEGIN	BEGIN	Introduces the instruction part for code blocks or initialization part for a data block
BIE	BR	Binary result
BLOCK_FB	BLOCK_FB	Parameter type for specification of an FB
BLOCK_FC	BLOCK_FC	Parameter type for specification of an FC
BLOCK_SDB	BLOCK_SDB	Parameter type for specification of an SDB
BOOL	BOOL	Data type
BY	BY	Increment of the FOR loop
BYTE	BYTE	Data type
CALL	CALL	Call
CASE	CASE	Introduction to the CASE statement
CHAR	CHAR	Elementary data type

Keywords German mnemonics	Keywords English mnemonics	Description
CODE_VERSION1	CODE_VERSION1	Label, whether an FB is multiple instance capable or not. If you want to declare multiple instances, the FB must not have this characteristic.
CONST	CONST	Start of the constant declaration
CONTINUE	CONTINUE	Instruction to exit a loop in SCL
COUNTER	COUNTER	Parameter type for specification of a counter
DATA_BLOCK	DATA_BLOCK	Introduces the data block
DATE	DATE	Data type
DATE_AND_TIME	DATE_AND_TIME	Data type
DB	DB	Data block
DB_ANY	DB_ANY	Data type
DBB	DBB	Data block, data byte
DBD	DBD	Data block, data double word
DBLG	DBLG	Data block length
DBNO	DBNO	Data block number
DBW	DBW	Data block, data word
DBX	DBX	Data block, data bit
DI	DI	Instance data block
DIB	DIB	Instance data block, data byte
DID	DID	Instance data block, data double word
DILG	DILG	Instance data block length
DINO	DINO	Instance data block number
DINT	DINT	Data type
DIW	DIW	Instance data block, data word
DIX	DIX	Instance data block, data bit
DO	DO	Introduction of the instruction part in FOR and WHILE instruction
DT	DT	Data type
DTL	DTL	Data type
DWORD	DWORD	Data type
E	I	Input (via process image), bit
EB	IB	Input (via process image), byte
ED	ID	Input (via process image), double word
ELSE	ELSE	Alternative branch in IF and CASE statement
ELSIF	ELSIF	Alternative condition of the IF instruction
EN	EN	System operand of the EN/ENO mechanism

Keywords German mnemonics	Keywords English mnemonics	Description
ENO	ENO	System operand of the EN/ENO mechanism
END_CASE	END_CASE	End of the CASE statement
END_DATA_BLOCK	END_DATA_BLOCK	Ends the data block
END_FOR	END_FOR	End of the FOR statement
END_FUNCTION	END_FUNCTION	Ends the function
END_FUNCTION_BLOCK	END_FUNCTION_BLOCK	Ends the function block
END_IF	END_IF	End of the IF instruction
END_ORGANIZATION_BLOCK	END_ORGANIZATION_BLOCK	Ends the organization block
END_REGION	END_REGION	End of a block
END_REPEAT	END_REPEAT	End of the REPEAT statement
END_STRUCT	END_STRUCT	Ends the specification of a structure
END_SYSTEM_FUNCTION	END_SYSTEM_FUNCTION	Ends the system function
END_SYSTEM_FUNC-TION_BLOCK	END_SYSTEM_FUNC-TION_BLOCK	Ends the system function block
END_TYPE	END_TYPE	Ends the PLC data type
END_VAR	END_VAR	Ends a declaration block
END_WHILE	END_WHILE	End of the WHILE instruction
EW	IW	Input (via process image), word
EXIT	EXIT	Instruction to exit a loop in SCL
FALSE	FALSE	Predefined Boolean constant: Logical condition false, value equal to 0
FAMILY	FAMILY	Block family name: e.g. controller
FB	FB	Function block
FC	FC	Function
FOR	FOR	Introduction of the FOR statement
FUNCTION	FUNCTION	Introduces the function
FUNCTION_BLOCK	FUNCTION_BLOCK	Introduces the function block
GOTO	GOTO	Introduction of the GOTO statement
IF	IF	Introduction of the IF instruction
INSTANCE	INSTANCE	Data type
INT	INT	Data type
KNOW_HOW_PROTECT	KNOW_HOW_PROTECT	Block protection
L	L	Local data bit
LB	LB	Local data byte
LD	LD	Local data double word
LDT	LDT	Data type
LINT	LINT	Data type
LTIME	LTIME	Data type

Keywords German mnemonics	Keywords English mnemonics	Description
LTOD	LTOD	Data type
LW	LW	Local data word
LWORD	LWORD	Data type
M	M	Memory bit
MB	MB	Memory byte
MD	MD	Memory double word
MOD	MOD	Modulo operator
MW	MW	Memory word
NAME	NAME	Block name
NETWORK	NETWORK	Network
NOT	NOT	Logic inversion
NULL	NULL	Zero pointer
OB	OB	Organization block
OF	OF	Introduction of the data type specification / Introduction of the instruction part of the CASE statement
OR	OR	Or logical operation of logical expressions
ORGANIZATION_BLOCK	ORGANIZATION_BLOCK	Introduces the organization block
OS	OS	Save overflow
OV	OV	Overflow
PA	PQ	Output (direct peripherals), bit
PAB	PQB	Output (direct peripherals), byte
PAD	PQD	Output (direct peripherals), double word
PAW	PQW	Output (direct peripherals), word
PE	PI	Input (direct peripherals), bit
PEB	PIB	Input (direct peripherals), byte
PED	PID	Input (direct peripherals), double word
PEW	PIW	Input (direct peripherals), word
POINTER	POINTER	Data type
PRAGMA_BEGIN	PRAGMA_BEGIN	Introduction of a Pragma
PRAGMA_END	PRAGMA_END	End of a Pragma
READ_ONLY	READ_ONLY	Write protection for data blocks
REAL	REAL	Data type
REF	REF	Instruction for generating a reference
REF_TO	REF_TO	Introduction for declaring a reference
REGION	REGION	Introduction of a block
REPEAT	REPEAT	Introduction of the REPEAT statement

Keywords German mnemonics	Keywords English mnemonics	Description
RET_VAL	RET_VAL	Return value
RETURN	RETURN	RETURN statement in SCL
S5T	S5T	Syntax for data type S5TIME
S5TIME	S5TIME	Data type
S7_	S7_	Keywords for system attributes
SDB	SDB	System data block
SFB	SFB	System function block
SFC	SFC	System function
SINT	SINT	Data type
STRING	STRING	Data type
STRUCT	STRUCT	Introduces the specification of a structure and is followed by a list of components
STW	STW	Status word
SYSTEM_FUNCTION	SYSTEM_FUNCTION	System function
SYSTEM_FUNCTION_BLOCK	SYSTEM_FUNCTION_BLOCK	System function block
T	T	Time element (timer)
THEN	THEN	Introduction of the instruction part of an IF instruction
THIS	THIS	Syntax for access to an ARRAY data block
TIME	TIME	Elementary data type for time information
TIME_OF_DAY	TIME_OF_DAY	Data type
TIMER	TIMER	Parameter type for specification of a timer
TITLE	TITLE	Optional block title or network title
TO	TO	Definition of the full-scale value of a FOR statement
TOD	TOD	Data type
TRUE	TRUE	Predefined Boolean constant: Logical condition true, value not equal to 0
TYPE	TYPE	Introduction of the PLC data type
UDT	UDT	Global or PLC data type
UDINT	UDINT	Data type
UINT	UINT	Data type
ULINT	ULINT	Data type
UNLINKED	UNLINKED	Marking 'non runtime-related'
UNTIL	UNTIL	End of the instruction part of a REPEAT statement
USINT	USINT	Data type
UO	UO	Query after (Q1=1) AND (Q0=1)

Keywords German mnemonics	Keywords English mnemonics	Description
VAR	VAR	Introduces a declaration block
VAR_IN_OUT	VAR_IN_OUT	Introduces a declaration block
VAR_INPUT	VAR_INPUT	Introduces a declaration block
VAR_OUTPUT	VAR_OUTPUT	Introduces a declaration block
VAR_TEMP	VAR_TEMP	Introduces a declaration block
VARIANT	VARIANT	Data type
VERSION	VERSION	Version number of the block
VOID	VOID	Function has no return value
WCHAR	WCHAR	Data type
WSTRING	WSTRING	Data type
WHILE	WHILE	Introduction of a WHILE instruction
WORD	WORD	Data type
XOR	XOR	Logic operation
Z	C	Counter

1.4.3 Using tags within the program

Definition

A variable is a placeholder for a data value that can be changed in the program. The format of the data value is defined. The use of variables makes your program more flexible. For example, you can assign different values to variables that you have declared in the block interface for each block call. As a result, you can reuse a block you have already programmed for various purposes.

A variable consists of the following elements:

- Name
- Data type
- Absolute address
  - PLC tags and DB variables in blocks with standard access have an absolute address.
  - DB variables in blocks with optimized access have no absolute address.
- Value (optional)

Declaring Variables

You can define variables with different scopes for your program:

- PLC tags that apply in all areas of the CPU
- DB variables in global data block that can be used by all blocks throughout the CPU.
- DB variables in instance data blocks that are predominantly used within the block in which they are declared.

The following table shows the difference between the variable types:

	PLC tags	Variables in instance DBs	Variables in global DBs
Range of application	• Are valid throughout the entire CPU. • Can be used by all blocks on the CPU. • The name is unique within the CPU.	• Are predominantly used in the block in which they are defined. • The name is unique within the instance DB.	• Can be used by all blocks on the CPU. • The name is unique within the global DB.
Permissible characters	• Letters, numbers, special characters • Quotation marks are not permitted. • Reserved keywords are not permitted.	• Letters, numbers, special characters • Reserved keywords are not permitted.	• Letters, numbers, special characters • Reserved keywords are not permitted.
Use	• I/O signals (I, IB, IW, ID, Q, QB, QW, QD) • Bit memory (M, MB, MW, MD)	• Block parameters (input, output and in-out parameters), • Static data of a block	• Static data
Location of definition	PLC tag table	Block interface	Declaration table of the global DB

Note

You can find more information on permissible characters in tag names in the Siemens Industry Online Support in the following entries:

When should identifiers and operands be used in "quotation marks" in STEP 7 (TIA Portal)?

https://support.industry.siemens.com/cs/ww/en/view/109477857 (https://support.industry.siemens.com/cs/ww/en/view/109477857)

You can find recommendations for uniform, project-wide tag naming in the programming style guide:

https://support.industry.siemens.com/cs/document/109478084 (https://support.industry.siemens.com/cs/ww/en/view/81318674)

See also

Keywords (Page 89)

Basic information about operands (Page 89)

Displaying symbolic and absolute addresses (Page 6844)

Valid names of PLC tags (Page 7072)

Permissible addresses and data types of PLC tags (Page 7073)

1.4.4 Constants

1.4.4.1 Basics of constants

Definition

Constants are data with a fixed value that you cannot change during program runtime. Constants can be read by various program elements during the execution of the program but cannot be overwritten. There are defined notations for the value of a constant, depending on the data type and data format. A distinction is made between the typed and non-typed notation.

Non-typed constants

In the non-typed notation, you only enter the value of the constant without a data type. Non-typed constants do not receive their data type until the first arithmetic or logical operation in which they are used.

The example below shows the non-typed notation:

SCL

```
#My_Int1 := #My_Int2 + 12345        (*The data type of the constant "12345"
                                     results from the addition with My_Int. 2. "12345"
                                     receives the data type INT.*)
#My_Real1 := #My_Real2 + 12345      (*The data type of the constant "12345" results
                                     from the addition with My_Real2. "12345" re-
                                     ceives the data type REAL.*)
```

Typed constants

In the typed notation, you specify a data type in addition to the value of the constant.

The example below shows the typed notation:

SCL

```
#My_Int1 :=  INT#12345                    (*The data type of the constant is always
                                          INT.*)
```

Note

Constants of BOOL type in LAD/FBD

The use of constants of type BOOL for inputs of instructions in LAD/FBD is only possible in CPUs of the S7-1200 series FW V4.0 or higher and S7-1500 series FW V1.8 or higher.

In older firmware versions, BOOL-type constants may only be used for instructions that are an internal function block (FB) of the system. These instructions are identified by the fact that the "Call options" dialog opens when you insert the instruction into a network. For all other instructions, Boolean constants must not be used as inputs.

Use of Boolean constants in S7-300/400 is not permitted.

See also

Declaring the block interface (Page 6974)

Declaring global constants (Page 7088)

Calculating with constants in SCL (Page 7313)

1.4.4.2 Declaration of symbolic names for constants

Symbolic constants

You have the option of declaring symbolic names for constants and thus making constant values available under a name in the program. This makes a program more readable and easier to maintain when changing constant values.

A symbolic constant consists of the following elements:

- Name
- Data type
  Symbolic constants always have a data type; non-typed notation is not possible for symbolic constants.
- Constant value
  You can select any value from the value range of the specified data type as constant value. For information on the value ranges, refer to the "Data types" chapter.

Declaration of constants

You can define constants with different scopes of validity:

- Global constants that apply to all areas of the CPU
- Local constants that only apply within a block

The table below shows the difference between the constant types:

	Global constants	Local constants
Scope of validity	• Are valid throughout the entire CPU • The name is unique within the CPU.	• Are valid only in the block in which they were declared. • The name is unique within the block.
Permitted characters	• The permitted characters in constant names are letters, digits and special characters.	• The permitted characters in constant names are letters, digits and special characters.
Location of definition	"Constants" tab from the PLC tag table	Block interface
Representation	In quotation marks, for example.: `"Glob_Const"`	Prefixed with a number sign, for example: `#Loc_Const`

Name conflicts can occur if you have declared a local and a global constant with the same symbolic name and have used this doubly specified name as the default value of a tag. In this case, the local constant is automatically used.

Note

Downloading constant declarations (S7-300/400)

Local and global constant declarations are not downloaded into the CPU. If you download a program from a device, the constant declarations may no longer be available.

Note

You can find more information on permissible characters in constant names in the Siemens Industry Online Support in the following entries:

When should identifiers and operands be used in "quotation marks" in STEP 7 (TIA Portal)?

https://support.industry.siemens.com/cs/ww/en/view/109477857 (https://support.industry.siemens.com/cs/ww/en/view/109477857)

See also

Overview of the valid data types (Page 231)

Calculating with constants in SCL (Page 7313)

Declaring global constants (Page 7088)

Declaring the block interface (Page 6974)

1.4.4.3 Data types of constants

Permitted data types

For constants, all basic data types as well as all derived data types are permitted:

- Binary numbers
- Bit strings
- Integers
- Floating-point numbers
- Timers
- Date and time
- Character strings

All general rules for explicit and implicit type conversion apply.

Data types of non-typed constants

Non-typed constants do not contain an explicit type specification. They do not receive their data type until the first arithmetic or logical operation in which they are used.

The example below shows how non-typed constants are used:

SCL

```
#My_Int1 := #My_Int2 + 12345        (*The data type of the constant "12345"
                                     results from the addition with My_Int. 2. "12345"
                                     receives the data type INT.*)
#My_Real1 := #My_Real2 + 12345      (*The data type of the constant "12345" results
                                     from the addition with My_Real2. "12345" re-
                                     ceives the data type REAL.*)
```

Note

STEP 7 always uses the data type with the highest possible precision

Unless the data type of a constant can be clearly defined in an expression, the data type with the highest precision available on the current CPU is used.

Example:

```
#My_Real := #My_Int / 3.5
```

In this expression an integer tag is combined with a non-typed floating point constant. In S7-300/400 the right part of the assignment is calculated in the REAL format. In S7-1200/1500, calculation is performed using the highest possible precision, which in this case means LREAL. As a result, the assignment to a REAL tag is invalid or generates a warning.

To precisely define the data type of a constant, use the typed notation.

Example:

```
#My_Real := #My_Int / REAL#3.5
```

See also

Overview of the valid data types (Page 231)

Calculating with constants in SCL (Page 7313)

Declaring global constants (Page 7088)

Declaring the block interface (Page 6974)

1.4.4.4 Examples of using constants

Use in instructions, assignments and expressions

Constants can be used in place of tags in instructions or assignments. You can also use constants in expressions in SCL. But because constants cannot be written, they may only be used as inputs.

The example below shows possible uses of constants:

SCL

```
#My_Int :=   3;
#My_Real1 := #My_Real2 * 3;
#My_Real1 := #My_Real2 * #My_local_const;
#My_Real1 := #My_Real2 * "My_global_const";
```

Use as a default value

You can use constants as the default value of a tag. To do so, enter either the value or the symbolic name of the constant in the "Default value" column of the block interface. The data type of the constant must match the data type of the tag or be convertible with it according to the implicit conversion with IEC check.

Use as maximum STRING length

You can use local or global constants of data type UINT, UDINT, ULINT, SINT, INT, DINT, LINT as maximum STRING length.

The example below shows the use of constants as maximum STRING length:

SCL

```
STRING[#My_local_const1]
STRING["My_global_const1"]
```

Use as an ARRAY limit

You can use local or global constants of data type UINT, UDINT, ULINT, SINT, INT, DINT, LINT as ARRAY limits.

The example below shows the use of constants as ARRAY limits:

SCL

```
Array[#My_local_const1..#My_local_const2] of REAL
Array["My_global_const1".."My_global_const1"] of REAL
```

Note

Constants as ARRAY limits or as maximum STRING length

- Constants which are used as ARRAY limits or maximum STRING length cannot be changed if the memory reserve of the block is activated. This applies to both local and global constants. To change these constants, you must first disable the memory reserve.
- Changes to global constants result in inconsistencies in the blocks which use them. The inconsistencies are marked in red in the block used. To remedy these inconsistencies, the data blocks have to be updated.
  See also: Updating data blocks (Page 7025)

See also

Overview of the valid data types (Page 231)

Calculating with constants in SCL (Page 7313)

Basic information on ARRAY (Page 283)

Declaring global constants (Page 7088)

Declaring the block interface (Page 6974)

1.4.5 Addressing operands

1.4.5.1 Addressing global variables

Addressing global variables

To address a global PLC variable, you can use the absolute address or the symbolic name.

Note

The LWORD, LINT, ULINT, LREAL, LTIME, LTOD and LDT data types can only be addressed by means of their symbolic name.

Addressing global variables in symbolic form

When you use addressing in symbolic form, you enter the variable name from the PLC tag table. The symbolic name of global variables are automatically enclosed in quotation marks.

You address structured tags that are based on a PLC data type with the symbolic name of the PLC tag. You can also indicate the names of the individual components separated by a dot.

Addressing global variables in absolute form

When you use addressing in absolute form, you enter the address of the variables from the PLC tag table. The absolute address uses numerical addresses starting with zero for each operand range. The address identifier % is set automatically as prefix for the absolute address of global tags.

Examples

The following examples show applications of symbolic and absolute addressing:

Addressing	Explanation
%Q1.0	Absolute address: Output 1.0
%I16.4	Absolute address: Input 16.4
%IW4	Absolute address: Input word 4
"Motor"	Symbolic address "Motor"
"Value"	Symbolic address "Value"
"Structured_Tag"	Symbolic address of a tag that is based on a PLC data type
"Structured_Tag".Component	Symbolic address of the component of a structured tag.

See also

Displaying symbolic and absolute addresses (Page 6844)

Permissible addresses and data types of PLC tags (Page 7073)

Accessing I/O devices (Page 104)

Accessing I/O devices

Description

The process image of the CPU is updated once in a cycle. In time-critical applications, however, it can be that the current state of a digital input or output has to be read or transferred more often than once per cycle. For this purpose you can use a suffix for I/O access identifiers on the operand to directly access the I/O.

If you want to read the input directly from the peripherals, use the peripheral inputs memory area (PI) instead of the process image input (I). The peripherals memory area can be read as a bit, byte, word, or double word.

If you want to write directly to the output, use the peripheral output (PQ) memory area instead of the process image output (Q). The peripheral output memory area can be written as a bit, byte, word, or double word.

To read or write a signal directly from a peripheral input, you can add the suffix for I/O access ":P", to the operand.

Components of structured PLC tags can also be addressed with ":P". However, access to the higher-level tag with ":P" is not possible.

⚠ WARNING
Direct writing of the I/O Immediate writing to the I/O can lead to hazardous states, for example when writing multiple times to an output in one program cycle.

Note

S7-1200/1500: Transferring peripheral inputs or outputs as block parameters

If you supply an input parameter with a peripheral input or output, it is possible that I/O access errors will occur during runtime when the block is called, e.g. a read error when an input module is accessed directly.

The system response of S7-1500 series CPUs with firmware version V2.1 or higher is as follows:

The block is called and processed with the substitute value of the signal.

The system response of S7-1200 and S7-1500 series CPUs with firmware versions lower than V2.1 is as follows:

The block is not called as a result of the I/O access error. Program execution is continued after the block call. If OB 122 exists or local error handling is enabled, these are executed.

To prevent the situation that the block is not called as a result of an I/O access error, first copy the peripheral input or output to a local tag (Temp) and transfer it as a block parameter to the called block.

The FAQ below provide additional information on this topic:

FAQ 89377245: Why are blocks in the S7-1200/S7-1500 not processed when they are addressed with unavailable PROFINET components? (https://support.industry.siemens.com/cs/ww/en/view/89377245)

Syntax

```
<Operand>:P
```

Example

The following example shows applications of I/O access identifiers:

Addressing	Description
"Motor"	Addresses the "Motor" tag in the process image.
"Motor":P	Addresses the "Motor" tag in the I/O memory area (PI or PQ).
"Structured_Tag".Component	Addresses the component of a structured PLC tag in the process image.
"Structured_Tag".Component:P	Addresses the component of a structured PLC tag in the I/O memory area (PI or PQ).

See also

Addressing global variables (Page 103)

1.4.5.2 Addressing variables in data blocks

Addressing variables in global data blocks

Description

Tags in global data blocks can be addressed in symbolic or absolute form. For symbolic addressing, you use the name of the data block and the name of the tag, separated by a dot. The name of the data block is enclosed in quotation marks.

For absolute addressing, you use the number of the data block and the absolute address of the tags in the data block, separated by a dot. The address identifier % is set automatically as prefix for the absolute address.

The S7-1200/1500 provides you with an option of accessing a data block that is not yet known during programming. For this purpose, create a block parameter of data type DB_ANY in the block interface of the accessing block. The data block name or data block number is transferred to this parameter during runtime. In order to access the internal tags of the data block, use the name of the block parameter of data type DB_ANY and the absolute address of the tag, separated by a dot.

Note

Transfer DB with memory reserve to the parameter DB_ANY.

It is not possible to transfer a DB with memory reserve to a block parameter of data type "DB_ANY".

Note

Addressing DB tags in absolute form

Absolute addressing is not possible for the following tags:

- Tags in blocks with optimized access.
- Tags of data type LWORD, LINT, ULINT, LREAL, LTIME, LTOD and LDT.

Best practice is to use the more convenient symbolic addressing for these tags.

ARRAY data blocks

ARRAY data blocks are a particular type of global data block. These consist of an ARRAY of any data type. For example, an ARRAY of a PLC data type (UDT) is possible.

You address elements in ARRAY data blocks with the help of the keyword "THIS". The index is then specified in square brackets. The index can be a constant as well as a tag. Integers with a width of up to 32 bits are permitted as tags for the index.
Extended options for addressing ARRAY DBs are available in the "Move" section of the "Instructions" task card. These instructions give you the option, for example, to also address the DB name indirectly.

Syntax

```
"<DBname>".TagName
%<DBnumber>.absoluteAddress
#<DBAny_name>.%absoluteAddress
"<ArrayDBname".THIS[#i].<Component>.<ComponentElement>
```

SCL:
```
"<ArrayDBname"."THIS"[#i].<Component>.<ComponentElement>
```

The following table show the possible absolute addresses of tags in data blocks:

Data type	Absolute address	Example	Description
BOOL	%DBn.DBXx.y	%DB1.DBX1.0	Data bit 1.0 in DB1
BYTE, CHAR, SINT, USINT	%DBn.DBBy	%DB1.DBB1	Data bit 1 in DB1
WORD, INT, UINT	%DBn. DBWy	%DB1.DBW1	Data word 1 in DB1
DWORD, DINT, UDINT, REAL, TIME	%DBn.DBDy	%DB1.DBD1	Data double word 1 in DB1

Example

The following examples show the addressing of tags in global data blocks:

Addressing	Description
"Motor".Value	Symbolic addressing of the "Value" tag in the "Motor" global data block.
%DB1.DBX1.0	Absolute addressing of the "DBX1.0" tags in the "DB1" global data block.
#MyDBAny.%DBX30.0	Absolute addressing of the "DBX30.0" tag in the global data block that is transferred at runtime at the "MyDBany" parameter.
"MyARRAY_DB".THIS[#MyIndex].MyComponent.MyComponentElement	Addressing an ARRAY data block. The ARRAY index is specified with the "MyIndex" tag. The ARRAY element has two additional substructures: "MyComponent" and "MyComponentElement".

See also

Using the DB_ANY data type (Page 204)

Addressing individual characters of a STRING or WSTRING (Page 270)

Addressing PLC data types (UDT) (Page 277)

Addressing STRUCT components (Page 281)

Addressing ARRAY components (Page 289)

Addressing areas of a tag with slice access (Page 110)

Basics of indirect addressing (Page 114)

Addressing instance data (Page 108)

Addressing instance data

Description

You can address data elements from the interface of the current block. These tags are stored in the instance data block.

Note

Tags in blocks with optimized access can only be addressed in symbolic form.

To address a tag from the interface of the current block, enter the character # followed by the symbolic tag name.

You can also access the tags of a multi-instance block. Within the multi-instance block, also use the character # followed by the tag name to address the data. You access the data of the multi-instance block from the calling block using #<Multi-instanceName.TagName>.

Syntax

Use the following syntax for addressing tags in instance data blocks:

```
#<tag name>
#<multi-instance name>.<tag name>
```

Examples

The following examples show the addressing of tags in instance data blocks:

Addressing	Description
#Value	Addressing the "Value" tag in the instance data block.
#On	Addressing the "On" tag within the multi-instance block
#Multi.On	Addressing the "On" tag of the multi-instance block from the calling block

See also

Addressing variables in global data blocks (Page 106)

Instances (Page 62)

Addressing ARRAY components (Page 289)

Addressing areas of a tag with slice access (Page 110)

Basics of indirect addressing (Page 114)

Addressing ARRAY DB (S7-1500)

Addressing ARRAY data blocks

ARRAY data blocks are a particular form of the ARRAY. ARRAY data blocks are global data blocks that consist of exactly one ARRAY. You address elements in ARRAY data blocks using the following syntax:

```
"<GlobArrayDBname>".THIS[#i].<elementname>
```

SCL:

```
"<GlobArrayDBname>"."THIS"[#i].<elementname>
```

Examples

The following examples show the addressing of elements in an ARRAY DB:

Addressing in SCL	Explanation
"My_ARRAY_DB"."THIS"[0]	"My_ARRAY_DB" is an ARRAY DB, which contains components of the REAL data type. The first component within "My_ARRAY_DB" is addressed here.
"MY_UDT_ARRAY_DB"."THIS"[3].MyREAL	"MY_UDT_ARRAY_DB" is an ARRAY DB, which contains components of a PLC data type (UDT). The PLC data type contains elements of the BYTE data type. In the "MY_UDT_ARRAY_DB", the fourth ARRAY component and the "MyREAL" element in it are addressed here.

Instructions for addressing ARRAY DBs

The "Instructions > Basic instructions" task card in the "Move > ARRAY DB" section offers extended options for addressing ARRAY DBs. These instructions give you the option, for example, to address the DB name indirectly:

- ReadFromArrayDB: Read from array data block
- WriteToArrayDB: Write to array data block
- ReadFromArrayDBL: Read from array data block in load memory
- WriteToArrayDBL: Write to array data block in load memory

See also

Basics of indirect addressing (Page 114)
Indirect indexing of ARRAY components (Page 121)
Example of the use of ARRAY data blocks (Page 44)
Global ARRAY data blocks (DB) (Page 43)
ARRAY (Page 283)

Addressing areas of a tag with slice access (S7-1200, S7-1500)

Description

You have the option to specifically address areas within declared tags. You can access areas of the 1-bit, 8-bit, 16-bit, or 32-bit width. The division of a memory area (e.g. BYTE or WORD) into a smaller memory area (e.g. BOOL) is also referred to as a "slice".

Structures, constants and tags overlaying AT cannot be addressed with slice access.

Syntax

The following syntax is used for addressing:

```
<Tag>.X<Bit number>
<Tag>.B<BYTE number>
<Tag>.W<WORD number>
<Tag>.D<DWORD number>
```

The syntax has the following components:

Part	Description
<Tag>	Tag that you access. The tag must be of the "Bit string" or "Integer" data type. With SCL, you can only program slice access to tags of the "Integer" data type when the IEC check is disabled.
X B W D	ID for the access width "Bit (1Bit)" ID for the access width "Byte (8 Bit)" ID for the access width "Word (16 Bit)" ID for access width "DWord (32-bit)"
<BIT number>	Bit number within <tag> that is accessed. Number 0 accesses the least significant BIT.
<BYTE number>	Byte number within <tag> that is accessed. The number 0 accesses the least significant BYTE.
<WORD number>	Word number within <tag> that is accessed. The number 0 accesses the least significant WORD.
<DWORD number>	DWord number within <tag> that is accessed. The number 0 accesses the least significant DWORD.

Examples of slice access

The following examples show the addressing of slices in bit, byte, word and double word modes:

Addressing	Description
"Engine".Motor.X0 "Engine".Motor.X7	"Motor" is a tag of the BYTE, WORD, DWORD or LWORD data type in the global data block "Engine". X0 addresses the bit address 0, X7 the bit address 7 within "Motor".
"Engine".Speed.B0 "Engine".Speed.B1	"Speed" is a tag of the WORD, DWORD or LWORD data type in the global data block "Engine". B0 addresses the byte address 0, B1 the byte address 1 within "Speed".
"Engine".Fuel.W0 "Engine".Fuel.W1	"FUEL" is a tag of the DWORD or LWORD data type in the global data block "Engine". W0 addresses the word address 0, W1 the word address 1 within "Fuel".
"Engine".Data.D0 "Engine".Data.D1	"Data" is a tag of the LWORD data type in the global data block "Engine". D0 addresses the double word address 0, D1 the double word address 1 within "Data".

Programming example

You can find a detailed example in the Siemens Industry Online Support:

https://support.industry.siemens.com/cs/ww/de/view/57374718 (https://support.industry.siemens.com/cs/ww/en/view/57374718)

Overlaying tags with AT

Description

To access data areas within a declared tag, you can overlay the declared tags with an additional declaration. This provides you with the option of addressing an already declared tag with a different data type. You can, for example, address the individual bits of a tag of WORD data type with an ARRAY of BOOL.

Alternatively, you can also use the instructions "SCATTER" and "GATHER" to break down bit sequences into an ARRAY of BOOL or group together individual bits to form a bit sequence. You can find these instructions in the "Basic instructions" task card in the "Move" area.

Rules

The following general rules are valid for tag overlaying:

- Overlaying is possible in S7-1200 and S7-1500 in STL, LAD, FBD and GRAPH.
- SCL supports overlaying in all CPU families.

- Overlaying of tags is possible in the following blocks:
  - In code blocks with standard access
  - In code blocks with optimized access for tags with the retain setting "Set in IDB"
- The data width of the overlaying tag must be equal to or less than that of the overlaid tag.
- It is not possible to overlay tags of the VARIANT and INSTANCE data types.
- If the overlaid tag is of the data type STRING, ARRAY of BYTE, ARRAY of CHAR, etc., the length should correspond to an even number of bytes.
- Blocks from libraries which are declared as parameters in the interface cannot be overlaid.
- Structured PLC tags that are declared as parameters in the interface cannot be overlaid.
- You cannot address overlaying tags with slice access.

Note

S7-1200/1500: Using AT in FCs

The data widths of the overlaying tag and the overlaid tag must be identical for FCs in S7-1200/1500. If this is not possible in your program, check to see if you can use slice access instead of the AT construct, or use the instructions "SCATTER" or "GATHER".

See also: Addressing areas of a tag with slice access (Page 110)

The following combination rules are also valid:

		Overlaying tag	Overlaid tag			
			Elementary	Structured *	Any/Pointer	DB_ANY
FB	Input	Elementary	x	x		x
		Structured *	x	x	x	x
		Any/Pointer		x		
	Temp	Elementary	x	x		
		Structured	x	x	x	
		Any/Pointer		x		
	Static, Output	Elementary	x	x		x
		Structured	x	x		x
		Any/Pointer				
	InOut	Elementary	x			x
		Structured		x		
		Any/Pointer				

		Overlaying tag	Overlaid tag			
FC	Temp	Elementary	x	x		
		Structured	x	x	x	
		Any/Pointer		x		
	Input, Output, InOut	Elementary (both tags must have the same bit width)	x			x
		Structured		x		
		Any/Pointer				
OB	Temp	Elementary	x	x		
		Structured	x	x	x	
		Any/Pointer		x		

* Structured data types consist of several data elements, e.g. ARRAY or STRUCT.

Declaration

To overlay a tag, declare an additional tag directly after the tag that is to be overlaid and identify it with the keyword "AT".

Example

The following figure shows the declaration of an overlaid tag in the interface of a FB:

Input		
MyByte		Byte
AT	AT "MyByte"	Array [0..7] of Bool
AT[0]		Bool
AT[1]		Bool
AT[2]		Bool
AT[3]		Bool
AT[4]		Bool
AT[5]		Bool
AT[6]		Bool
AT[7]		Bool

When a block is called with the shown tag declaration, the "MyByte" tag is assigned. Within the block there are now two options for interpreting the data:

- as a byte
- As one-dimensional ARRAY of BOOL

See also

Setting retentivity (Page 7037)

1.4.5.3 Addressing operands indirectly

Indirect addressing (S7-1200, S7-1500)

Basics of indirect addressing (S7-1200, S7-1500)

Introduction

Indirect addressing offers you the option of addressing operands whose address is not calculated until runtime. With indirect addressing, you can also have program sections run several times and use a different operand during each run. For example, you can use a different index in each run in program loops.

This gives you the following advantages:

- You have more flexibility within your program.
- Indirect addressing is available to you in all STEP 7 programming languages.
- The program code is more straightforward.
- The existing names of data blocks and tags are used (symbolic addressing). This improves readability of the program code.

WARNING

Risk of access errors

Since operands are only calculated at runtime with indirect addressing, there is a risk that access errors may occur and that the program will operate with incorrect values. In addition, memory areas may inadvertently be overwritten with incorrect values. The automation system can then react in an unexpected manner.

Therefore, indirect addressing should be used with caution.

General indirect addressing options in S7-1200 and S7-1500

The following indirect addressing options are available in all programming languages:

- Indirect addressing via pointer
- Indirect indexing of ARRAY components
- Indirect addressing of a data block via DB_ANY data type.

Language-specific options of indirect addressing

The following specific addressing options are also available in the various programming languages:

- In STL, you can address operands indirectly via the address register.
- In SCL, you can read or write a variable memory area with the following instructions:
  - POKE - Write memory address
  - POKE_BOOL - Write memory bit
  - PEEK - Read memory address
  - PEEK_BOOL - Read memory bit
  - POKE_BLK - Write memory area

See also

Addressing variables in global data blocks (Page 106)

POKE: Write memory address (Page 1781)

POKE_BOOL: Write memory bit (Page 1783)

PEEK: Read memory address (Page 1778)

PEEK_BOOL: Read memory bit (Page 1780)

POKE_BLK: Write memory area (Page 1785)

Indirect addressing using a pointer (Page 120)

Indirect indexing of ARRAY components (Page 121)

Indirect addressing in STL (Page 127)

Examples of indirect addressing (S7-1200, S7-1500)

1. Program example

In the example below, you use the index to access three tags from different memory areas.

Overview of the three tags that are assigned to one index each:

Index	Accessing tag	Memory area
1	Input_WORD_0	IW 0
2	"Processdata".Temperature	DB 1
3	Output_WORD_4	QW 4

Declare the following two tags in the "Default tag table":

Default tag table

	Name	Data type	Address
1	Input_WORD_0	Word	%IW0
2	Output_WORD_4	Word	%QW4

Create a global data block:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_Processdata".
4. Select "Global DB" as the type of the data block.
5. Click "OK".
6. Declare the data block element "Temperature":

DB_Processdata

	Name	Data type	Start value
1	Static		
2	Temperature	DInt	50

Declare indirect access using an index within a function.

1. Create an SCL function and name it "FB_AccessGroupInt".
2. Declare the block interface as follows:

FC_AccessGroupInt

	Name	Data type
1	Input	
2	Index	Int
3	Output	
4	InOut	
5	Temp	
6	Constant	
7	Return	
8	FC_AccessGroupInt	DInt

3. Write the following program code:

```
1
2  CASE #Index OF
3
4      1: // First case: "Input_WORD_0"
5          #FC_AccessGroupInt := "Input_WORD_0";
6      2: // Second case: "DB_Processdata".Temperature
7          #FC_AccessGroupInt := "DB_Processdata".Temperature;
8      3: // Third case: "Output_WORD_4"
9          #FC_AccessGroupInt := "Output_WORD_4";
10
11 END_CASE;
```

4. Call the "FC_AccessGroupInt" function in OB1:

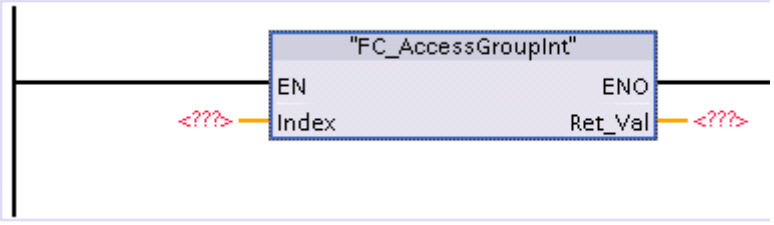

Depending on which number (1, 2 or 3) you specify at the Index parameter, the first, second or third case of the "FC_AccessGroupInt" instruction is executed.

2. Program example

In the example below, you use the index to access three different optimized data blocks.

Since all data blocks should contain the same tags, you can use a PLC data type (UDT) in this case.

1. To create a PLC data type, double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Rename the PLC data type to "UDT_SiloContents".
3. Declare the following lines within the PLC data type:
   MyBool > Data type: BOOL
   MyInt > Data type: INT
   MyWord > Data type: WORD

UDT_SiloContents			
		Name	Data type
1	...	MyBool	Bool
2	...	MyInt	Int
3	...	MyWord	Word

Create three global data blocks.

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the names "DB_SiloWater", "DB_SiloSugar" and "DB_SiloMilk".
4. Select the data blocks "UDT_SiloContents" as the type of the data blocks.
5. Click "OK".

DB_SiloWater

	Name	Data type
1	Static	
2	MyBool	Bool
3	MyInt	Int
4	MyWord	Word

DB_SiloSugar

	Name	Data type
1	Static	
2	MyBool	Bool
3	MyInt	Int
4	MyWord	Word

DB_SiloMilk

	Name	Data type
1	Static	
2	MyBool	Bool
3	MyInt	Int
4	MyWord	Word

Create a function to read the values of the data block tags and to write these to a PLC data type.

1. Create an SCL function and name it "FC_AccessGroupSiloRead".
2. Declare the block interface as follows:

FC_AccessGroupSiloRead		
	Name	Data type
1	Input	
2	Index	DInt
3	Output	
4	InOut	
5	Temp	
6	Constant	
7	Return	
8	FC_AccessGroupSiloRead	"UDT_SiloContents"

3. Write the following program code:

```
1
2  CASE #Index OF
3
4      1: // First case: "DB_SiloWater"
5          #FC_AccessGroupSiloRead := "DB_SiloWater";
6      2: // Second case: "DB_SiloSugar"
7          #FC_AccessGroupSiloRead := "DB_SiloSugar";
8      3: // Third case: "DB_SiloMilk"
9          #FC_AccessGroupSiloRead := "DB_SiloMilk";
10
11     ELSE // Optional case
12         #FC_AccessGroupSiloRead := "DB_SiloWater";
13
14 END_CASE;
```

4. Call the "FC_AccessGroupSiloRead" function in OB1:

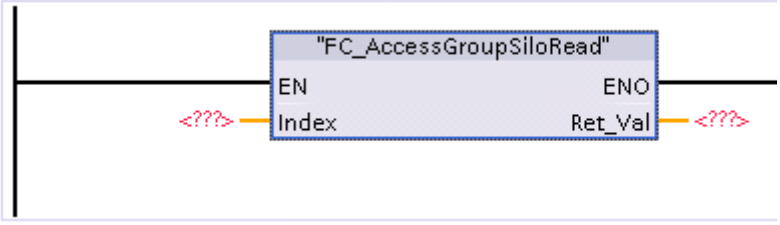

Depending on which number (1, 2 or 3) you specify at the Index parameter, the first, second or third case of the "FC_AccessGroupSiloRead" instruction is executed.

See also

Indirect addressing in STL (Page 133)
Indirect addressing in SCL (Page 130)
Addressing operands indirectly (Page 114)

Indirect addressing using a pointer (S7-1200, S7-1500)

Description

For indirect addressing, a special data format is required that contains the address and possibly also the range and the data type of an operand. This data format is referred to as pointer. The following types of pointers are available to you:

- References (S7-1500)
- VARIANT (S7-1200/1500)
- POINTER (S7-1500)
- ANY (S7-1500, only for blocks with standard access)

For more information on the pointer data types, refer to "See also".

Note

POINTER in SCL

In SCL the use of the POINTER is restricted. The only option available is to forward it to the called block.

Example

The following example shows an indirect addressing with an area-internal pointer:

Addressing in STL	Explanation
`L P#10.0`	`// Load pointer (P#10.0) in accumulator 1`
`T MD20`	`// Transfer pointer to the operand MD20`
`L MW [MD20]`	`// Load MW10 in Accumulator 1`
`....`	`// Any program`
`L MD [MD20]`	`// Load MD10 in Accumulator 1`
`....`	`// Any program`
`= M [MD20]`	`// If RLO=1, set the memory bit M10.0`

The pointer P#10.0 is transferred to the operand MD20. If the operand MD20 in square brackets is programmed, this will be replaced in runtime by the address that is contained in the pointer.

See also

Basics of indirect addressing (Page 114)

Pointer (Page 300)

Indirect indexing of ARRAY components (S7-1200, S7-1500)

ARRAY access with variable index

For addressing ARRAY elements, you can specify either constants or tags of the integer data type as index. Integers with a length of up to 32 bits are allowed here.

With indirect addressing using a tag, the index is only calculated during program runtime. You can, for example, use a different index for each cycle in program loops.

Syntax

The following syntax is used for the indirect indexing of a ARRAY:

```
"MyDB".MyArray[#i]   // One-dimensional ARRAY
"MyDB".MyArray[#i].a // One-dimensional ARRAY of STRUCT
"MyDB".MyArray[#i,#j]// Multi-dimensional ARRAY
"MyDB".MyArray[#i].a // Multi-dimensional ARRAY of STRUCT
```

The syntax has the following components:

Part	Description
MyDB	Name of the data block in which the ARRAY is located
MyArray	Tag of the ARRAY data type
i, j	PLC tags of the integer data type that are used as pointers
a	Additional partial tag of the structure

Note

When you call a block and transfer an indirectly indexed ARRAY component ("MyDB".MyArray[#i]) to it as in/out parameter (InOut), you cannot change the value of the index tag [i] while the block is being executed. The value is therefore always written back to the same ARRAY component from which it was read.

Indexing ARRAY components using the "FieldRead" and "FieldWrite" instructions

You may also use the following instructions for indirect indexing of ARRAY components in LAD and FBD:

- FieldWrite - Write field
- FieldRead - Read field

See also

Examples of indirect indexing of ARRAYs (Page 122)
Indirect addressing of ARRAY components (Page 200)
Basics of indirect addressing (Page 114)
Addressing ARRAY components (Page 289)
ARRAY (Page 283)

Examples of indirect indexing of ARRAYs (S7-1200, S7-1500)

Example: ARRAY access with variable index in LAD

The following example shows the indirect indexing of an ARRAY component using LAD as example. "MyArray" is a three-dimensional ARRAY. #Tag_1, #Tag_2 and #Tag_3 are input parameters of the data type "Integer". Depending on their values, one of the MyArray components is copied to the "MyTarget" tag.

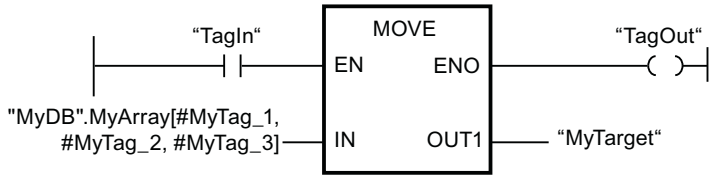

Example: ARRAY access with variable index in SCL

The following example shows you how to address multiple data blocks using a variable index. Five speed-controlled axes are used in the example. These axes are to be processed with SCL iteratively in a FOR loop.

1. First you create five speed-controlled axes. All five axes must be of the same type. A data block is created for each axis in the project navigator:

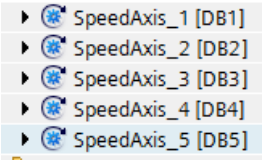

2. Next, generate a global data block and declare an ARRAY of the data type ANY with five components.

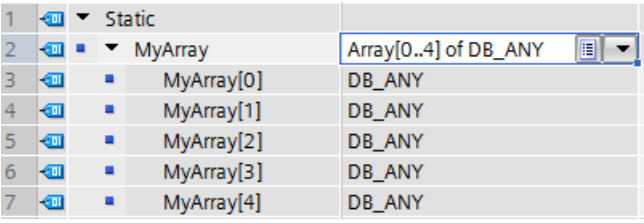

3. For initialization, you assign the symbolic names of the five axes to the individual ARRAY elements in the startup OB.

```
1  "MyGlobalDB".MyArray[0] := "SpeedAxis_1";
2  "MyGlobalDB".MyArray[1] := "SpeedAxis_2";
3  "MyGlobalDB".MyArray[2] := "SpeedAxis_3";
4  "MyGlobalDB".MyArray[3] := "SpeedAxis_4";
5  "MyGlobalDB".MyArray[4] := "SpeedAxis_5";
6
```

4. In SCL, access to the individual axes takes place iteratively with the help of a FOR loop. The axes are transferred to the function "MaximumVelocity" for execution one after the other.

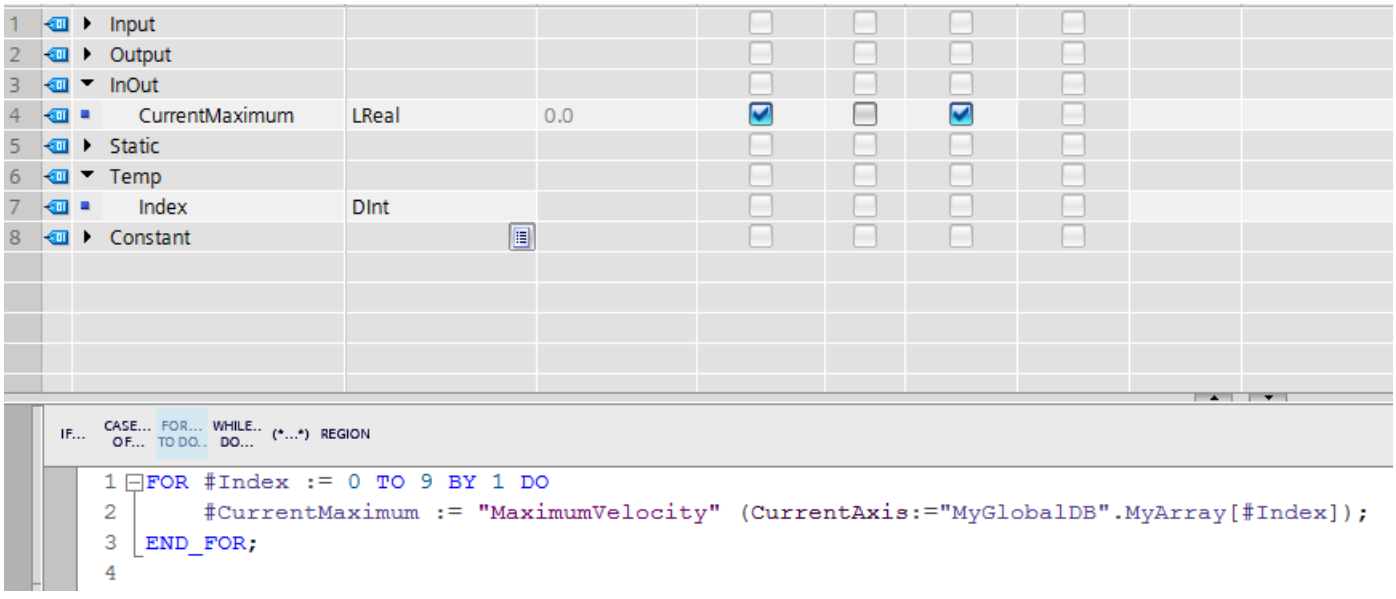

Example: ARRAY access with variable index in STL

An application example for ARRAY access with variable index in STL is available under "Addressing ARRAY elements indirectly". (Page 200)

See also

Indirect indexing of ARRAY components (Page 121)

Basics of indirect addressing (Page 114)

Addressing ARRAY components (Page 289)

ARRAY (Page 283)

Indirect addressing of individual characters of a STRING (S7-1200, S7-1500)

Description

For addressing the individual characters of a STRING or WSTRING, you can specify constants and also tags as the index. The tags must be of the Integer data type. When tags are used, the index is calculated during runtime. You can, for example, use a different index for each cycle in program loops.

If you transfer a STRING or WSTRING with variable index to an in/out parameter during a block call, please note that: The index tag [i] is read once at the start of the block call and cannot be changed by the called block while it is being executed.

Note

Monitoring STRING access in runtime

When a STRING or WSTRING that exceeds the defined length is written in runtime, unwanted reactions may occur in the program. Violation of the STRING or WSTRING length is monitored in S7-1200/1500. On read access to the STRING, you receive the character '$00' or '$0000'; write access to the STRING is not executed. If the instruction has the enable output ENO, ENO is set to the signal state FALSE. The CPU does not change to STOP.

Syntax

The following syntax is used for the indirect indexing of a STRING or WSTRING:

```
"<Data block>".<STRING>["i"]
"<Data block>".<WSTRING>["i"]
```

Example

The example below shows indirect indexing of a STRING using SCL as an example. "STRING", "WSTRING", "CHAR" and "WCHAR" are tags. "Tag_1" is a PLC tag of the "Integer" data type.

Addressing in SCL	Description
`STRING["Tag_1"] := CHAR;`	`(*Indirect addressing: Assignment of "CHAR" to the character of the STRING*) specified by "Tag_1"`
`WSTRING["Tag_1"] := WCHAR;`	`(*Indirect addressing: Assignment of "WCHAR" to the character of the WSTRING*) specified by "Tag_1"`
`WCHAR := WSTRING["Tag_1"];`	`(*Indirect addressing: Assignment of the WSTRING character specified by "Tag_1" to WCHAR*)`

See also

STRING (Page 264)

WSTRING (Page 267)

Indirect addressing in STL (S7-1500)

Basic information about address registers (S7-1500)

Introduction

Two address registers are available for the indirect addressing of operands: address register 1 (AR1), and address register 2 (AR2). The address registers are equal and are 32 bits in length. You can store area-internal and cross-area pointers in the address registers. To define the address of an operand, you can call the stored data in the program.

Data is exchanged between the registers and the other available memory areas with the assistance of load and transfer instructions.

Note

In S7-1500, special rules apply to data exchange via address register and data block register:

- The values in the registers do not remain in existence beyond the block limits.
- The registers are reset when the language is changed within a block.
- You can only reference data in blocks with optimized access if these have the retain setting "Set in IDB".
- It is not possible to reference local data in blocks with optimized access with the help of the address registers (across areas).

See also

LAR1: Load AR1 with contents of accumulator 1 (Page 4791)

LAR1 <D>: Load AR1 with double word or area pointer (Page 4792)

LAR1 AR2: Load AR1 with contents of AR2 (Page 4793)

LAR2: Load AR2 with contents of accumulator 1 (Page 4794)

CAR: Switch AR1 and AR2 (Page 4799)

TAR1: Transfer AR1 to accumulator 1 (Page 4800)

TAR1 <D>: Transfer AR1 to double word (Page 4801)

TAR1 AR2: Transfer AR1 to AR2 (Page 4802)

TAR2: Transfer AR2 to accumulator 1 (Page 4803)

+AR1: Add accumulator 1 to AR1 (Page 4889)

+AR2: Add accumulator 1 to AR2 (Page 4890)

Indirect addressing in STL (Page 127)

Data exchange in STL (Page 7245)

Addressing areas of a tag with slice access (Page 110)

Load and transfer (Page 1556)

Setting retentivity (Page 7037)

Indirect addressing in STL (S7-1500)

In STL, the following options are available for indirect addressing:

- Memory-indirect addressing
- Register-indirect area-internal addressing
- Register-indirect cross-area addressing

Memory-indirect addressing

In the case of memory-indirect addressing, you store the address in a tag. The tag can be of WORD or DWORD data type. The tag can be located in the memory areas "Data" (DB or DI), "Bit memory" (M) or "Temporary local data" (L). In S7-1500, FB parameters can also be used to store the address. If the tag is located in a data block, it must be a data black with standard access.

The following example shows applications of memory-indirect addressing:

Addressing in STL	Explanation
`U E [MD 2]`	`// Execute an AND logic operation with a variable input bit. The address of the input bit is located in the memory double word MD2.`
`= DIX [DBD 2]`	`// Assign the RLO to a variable data bit. The address of the data bit is located in the data double word DBD2.`
`L EB [DID 4]`	`// Load a variable input byte to ACCU 1. The address of the input byte is located in the instance double word DID4.`
`AUF DB [LW 2]`	`// Open a variable data block. The number of the data block is located in the local data word LW2.`

Register-indirect area-internal addressing

Register-indirect addressing uses one of the address registers (AR1 or AR2) to pick up the address of the operand.

In the case of register-indirect, area-internal addressing, you index only the bit address and the byte address via the address register (e.g. `P#10.0`). You do not enter the memory area for which the address in the address register is to apply until during programming of the instruction. The address in the address register then moves to the memory area specified in the instruction.

Possible memory areas are "Inputs" (I), "Outputs" (Q), "I/O" (PI or PQ), "Bit memory" (M), "Temporary local data" (L) and "Data" (DB or DI). If the operand is located in a data block, it must be a data black with standard access.

When you enter register-indirect, area-internal addressing, specify an offset after the specification of the address register. This offset is added to the contents of the address register

without changing the address register. This offset also has the format of a pointer. The specification of a pointer is mandatory and must be entered as constant (e.g. `P#0.0` or `P#2.0`).

The following example shows an application of register-indirect area-internal addressing:

STL	Explanation
`LAR1 P#10.0`	`// Load pointer (P#10.0) to address register 1`
`L IW [AR1, P#2.0]`	`// Increase contents of address register 1 (P#10.0) by offset P#2.0.`
	`// Load contents of input word IW12 into accumulator 1`
`L IW [AR1, P#0.0]`	`// Increase contents of address register 1 (P#10.0) by offset P#0.0.`
	`// Load contents of input word IW10 into accumulator 1`

Register-indirect cross-area addressing

In the case of register-indirect, cross-area addressing, use the address register to index the entire address of the operand, in other words, the bit address and byte address, as well as the memory area. Possible memory areas are "Inputs" (I), "Outputs" (Q), "I/O" (P), "Bit memory" (M), "Temporary local data" (L) and "Data" (DB or DI). If the operand is located in a data block, it must be a data black with standard access or the operand must have the retain setting "Set in IDB".

In the instruction, program only the operand width. Possible operand widths are bit, byte, word, and double word.

The following example shows an application of register-indirect cross-area addressing:

`LAR1 P#M10.0`	`// Load cross-area pointer (P#M10.0) to address register 1`
`L W [AR1, P#2.0]`	`// Increase contents of address register 1 (P#M10.0) by offset P#2.0.`
	`// Load contents of memory word "MW12" into accumulator 1`
`LAR1 P#A10.0`	`// Load cross-area pointer (P#A10.0) to address register 1`
`L W [AR1, P#2.0]`	`// Add contents of address register 1 (P#A10.0) by offset P#2.0`
	`// Load contents of output word QW12.0 into accumulator 1`

Note

Special features in S7-1500

In S7-1500, special rules apply to data exchange via address register and data block register:

- The values in the registers do not remain in existence beyond the block limits. The registers are also reset when the language is changed within a block.
- If you access an operand of the BYTE, WORD or DWORD type using register-indirect addressing, the address must begin at a byte limit.
  Examples:
  LAR1 P#0.0
  L MW [AR1, P#0.0] // P#0.0 + P#0.0 = P#0.0 - The addressing is allowed, because P#0.0 points to a byte limit.
  L MW [AR1, P#2.1] // P#0.0 + P#2.1 = P#2.1 - The addressing is not allowed, because P#2.1 does not point to a byte limit.

See also

Basics of indirect addressing (Page 114)

Indirect addressing of data blocks (Page 130)

Addressing ARRAY components (Page 289)

Basic information about address registers (Page 126)

Data exchange in STL (Page 7245)

Indirect addressing (S7-300, S7-400)

Indirect addressing via pointer (S7-300, S7-400)

Description

For indirect addressing, a special data format is required that contains the address and possibly also the range and the data type of an operand. This data format is referred to as pointer. The following types of pointer are available to you:

- POINTER (S7-300/400)
- ANY (S7-300/400)
- VARIANT (S7-1200/1500)

For more information on the pointer data type, refer to "See also".

Note

In SCL the use of the pointer data type is restricted. The only option available is to forward it to the called block.

Example

The following example shows an indirect addressing with an area-internal pointer:

Addressing in STL	Description
L P#10.0	// Load pointer (P#10.0) in accumulator 1
T MD20	// Transfer pointer to the operand MD20
L MW [MD20]	// Load MW10 in accumulator 1
....	// Any program
L MD [MD20]	// Load MD10 in accumulator 1
....	// Any program
= M [MD20]	// If RLO=1, set the memory bit M10.0

The pointer P#10.0 is transferred to the operand MD20. If the operand MD20 in square brackets is programmed, this will be replaced in runtime by the address that is contained in the pointer.

See also

Basics of indirect addressing (Page 114)

Pointer (Page 300)

Indirect addressing in SCL (S7-300, S7-400)

Indirect addressing of data blocks (S7-300, S7-400)

Indirect addressing of data blocks in SCL

You can use the conversion function WORD_TO_BLOCK_DB to address data blocks indirectly. The DB number is therefore specified as the tag or expression with WORD data type.

Syntax

The following syntax is used for the indirect addressing of a data block:
`WORD_TO_BLOCK_DB(Index).Operand ID (Address)`

Examples

Example 1: The global tag "Address index" of WORD data type is used as the number of the DB.

Addressing in SCL
%M0.0:=WORD_TO_BLOCK_DB("Addressindex").DX(0,0);
%MW0:=WORD_TO_BLOCK_DB("Addressindex").DW(4);

Example 2: The global tag "Address index" of WORD data type is used as the number of the DB. The data element within the DB is also specified by means of an index:

Addressing in SCL

```
%M0.0:=WORD_TO_BLOCK_DB("Addressindex").DX(#i,#y);
%MW0:=WORD_TO_BLOCK_DB("Addressindex").DW(#y);
```

See also

Basics of indirect addressing (Page 114)

Indirect addressing in STL (Page 127)

Addressing ARRAY components (Page 289)

Indirect addressing of tags (S7-300, S7-400)

Indirect addressing of tags in SCL

Indirect addressing is performed similarly to absolute addressing. An offset in round brackets is specified instead of the address. The offset consists of one byte tag or, in the case of Boolean operands, one byte tag and one bit tag. Byte and bit tags must be of data type INT.

Timers and counters from the PLC tag table cannot be addressed indirectly in this way.

Syntax

The following syntax is used for the indirect addressing of a global tag:

```
Operand ID (Byte tag)
Operand ID (Byte tag.Bit tag)
```

The following syntax is used for the indirect addressing of a DB tag:

```
MyDB.OperandID(ByteTag)
MyDB.OperandID(ByteTag.BitTag)
```

Examples

Example 1: One of the input words 2 to 8 is set to 0, depending on the value of run tag i.

Addressing in SCL

```
#i:=2;
FOR #i := 2 TO 8 DO
    %IW(#i) := 0 ;
END_FOR;
```

Example 2: One of the words 2 to 8 in DB10 is set to 0, depending on the value of run tag i.

Addressing in SCL

```
#i:=2;
FOR #i := 2 TO 8 DO
    %DB10.DW(#i) := 0 ;
END_FOR;
```

Indirect addressing of ARRAY components (S7-300, S7-400)

Description

For addressing the components of an ARRAY, you can enter tags of the integer data type as well as constants as the index. When using tags, the index will be calculated during runtime. You can, for example, use a different index for each cycle in program loops.

Note

The index tag [i] is read once at the start of the block call and cannot be changed by the called block while it is being processed.

When you call a block and transfer an indirectly indexed ARRAY component ("<DataBlock>".<ARRAY>["i"]) to it as in/out parameter (InOut), you cannot change the value of the index tag while the block is being processed. The value is therefore always written back to the same ARRAY component from which it was read.

Syntax

The following syntax is used for the indirect indexing of a ARRAY:

```
"<Data block>".<ARRAY>["i"] // one-dimensional ARRAY
"<Data block>".<ARRAY>["i"] // one-dimensional ARRAY of STRUCT
"<Data block>".<ARRAY>["i"] // multidimensional ARRAY
"<Data block>".<ARRAY>["i"] // multidimensional ARRAY of STRUCT
```

The syntax has the following components:

Part	Description
Data block	Name of the data block in which the ARRAY is located
ARRAY	Tag of the ARRAY data type
i, j	PLC tags of the integer data type that are used as pointers
a	Additional partial tag of the structure

Examples

The following examples are based on SCL and demonstrate indirect indexing of an ARRAY component. MOTOR is a one-dimensional ARRAY_of_INT with three rows. VALUES is a PLC tag of data type "Integer".

Addressing in SCL	Explanation
`MOTOR[2] := VALUES;`	`(*Direct addressing: Assignment of VALUES to the second row of the ARRAY MOTOR*)`
`MOTOR["Tag_1"] := VALUES;`	`(*Indirect addressing: Assignment of VALUES to the rows of ARRAY MOTOR*) specified by "Tag_1"`
`#MOTOR["Tag_2"+"Tag_3"] := #Values;`	`(*Indirect addressing: Assignment of VALUES to the row of the MOTOR*) ARRAY specified by the expression "Tag_2"+"Tag_3"`

See also

Addressing ARRAY components (Page 289)

Indirect addressing in STL (S7-300, S7-400)

Basic information about address registers (S7-300, S7-400)

Introduction

Two address registers are available for the indirect addressing of operands: address register 1 (AR1), and address register 2 (AR2). The address registers are equal and are 32 bits in length. You can store area-internal and cross-area pointers in the address registers. To define the address of an operand, you can call the stored data in the program.

Data is exchanged between the registers and the other available memory areas with the assistance of load and transfer instructions.

See also

LAR1: Load AR1 with contents of accumulator 1 (Page 4791)

LAR1 <D>: Load AR1 with double word or area pointer (Page 4792)

LAR1 AR2: Load AR1 with contents of AR2 (Page 4793)

LAR2: Load AR2 with contents of accumulator 1 (Page 4794)

CAR: Switch AR1 and AR2 (Page 4799)

TAR1: Transfer AR1 to accumulator 1 (Page 4800)

TAR1 <D>: Transfer AR1 to double word (Page 4801)

TAR1 AR2: Transfer AR1 to AR2 (Page 4802)

TAR2: Transfer AR2 to accumulator 1 (Page 4803)

+AR1: Add accumulator 1 to AR1 (Page 4889)

+AR2: Add accumulator 1 to AR2 (Page 4890)

Indirect addressing in STL (Page 134)

Data exchange in STL (Page 7245)

Load and transfer (Page 1556)

Indirect addressing in STL (S7-300, S7-400)

In STL, the following options are available for indirect addressing:

- Memory-indirect addressing
- Register-indirect area-internal addressing
- Register-indirect cross-area addressing

Memory-indirect addressing

In the case of memory-indirect addressing, you store the address in a tag. The tag can be of WORD or DWORD data type. The tag can be located in the memory areas "Data" (DB or DI), "Bit memory" (M) or "Temporary local data" (L).

The following example shows applications of memory-indirect addressing:

Addressing in STL	Description
`U E [MD 2]`	`// Execute an AND logic operation with a variable input bit. The address of the input bit is located in the memory double word MD2.`
`= DIX [DBD 2]`	`// Assign the RLO to a variable data bit. The address of the data bit is located in the data double word DBD2.`
`L EB [DID 4]`	`// Load a variable input byte to ACCU 1. The address of the input byte is located in the instance double word DID4.`
`AUF DB [LW 2]`	`// Open a variable data block. The number of the data block is located in the local data word LW2.`

Register-indirect area-internal addressing

Register-indirect addressing uses one of the address registers (AR1 or AR2) to pick up the address of the operand.

In the case of register-indirect, area-internal addressing, you index only the bit address and the byte address via the address register (e.g. `P#10.0`). You do not enter the memory area for which the address in the address register is to apply until during programming of the instruction. The address in the address register then moves to the memory area specified in the instruction.

Possible memory areas are "Inputs" (I), "Outputs" (Q), "I/O" (PI or PQ), "Bit memory" (M), "Temporary local data" (L) and "Data" (DB or DI).

When you enter register-indirect, area-internal addressing, specify an offset after the specification of the address register. This offset is added to the contents of the address register without changing the address register. This offset also has the format of a pointer. The specification of a pointer is mandatory and must be entered as constant (e.g. `P#0.0` or `P#2.0`).

The following example shows an application of register-indirect area-internal addressing:

STL	Description
LAR1 P#10.0	// Load pointer (P#10.0) to address register 1
L IW [AR1, P#2.0]	// Increase content of address register 1 (P#10.0) by offset P#2.0.
	// Load content of input word IW12 into accumulator 1
L IW [AR1, P#0.0]	// Increase content of address register 1 (P#10.0) by offset P#0.0.
	// Load content of input word IW10 into accumulator 1

Register-indirect cross-area addressing

In the case of register-indirect, cross-area addressing, use the address register to index the entire address of the operand, i.e., the bit address and byte address, as well as the memory area. Possible memory areas are "Inputs" (I), "Outputs" (Q), "I/O" (P), "Bit memory" (M), "Temporary local data" (L) and "Data" (DB or DI). In the instruction, program only the operand width. Possible operand widths are bit, byte, word, and double word.

The following example shows an application of register-indirect cross-area addressing:

LAR1 P#M10.0	// Load cross-area pointer (P#M10.0) to address register 1
L W [AR1, P#2.0]	// Increase content of address register 1 (P#M10.0) by offset P#2.0.
	// Load content of memory word "MW12" into accumulator 1
LAR1 P#A10.0	// Load cross-area pointer (P#A10.0) to address register 1
L W [AR1, P#2.0]	// Add content of address register 1 (P#A10.0) by offset P#2.0
	// Load content of output word QW12.0 into accumulator 1

See also

Basics of indirect addressing (Page 114)

Indirect addressing of data blocks (Page 130)

Addressing ARRAY components (Page 289)

Basic information about address registers (Page 133)

Data exchange in STL (Page 7245)

1.5 Handling program execution errors

1.5.1 Causes of error

Introduction

There are different causes for various error qualities in TIA Portal to which you can respond with different mechanisms. These mechanisms with which you can react to an error are independent of the programming languages. Only the display of the individual mechanisms depends on the respective programming language.

The situation determines what counts as an error. The overflow in the case of an addition, for example, can be an error because it does not return the desired result. In some cases, however, the addition overflow is acceptable and defined behavior and therefore not an error.

When creating a program code, you must be aware that certain situations may arise. When you program communication, for example, you must be aware that this connection can be canceled at any time. To be prepared for this situation, you must install a corresponding fault reaction because any connection abort prevents the T_SEND program block from transmitting a message. This is why it is important that the connection abort is signaled to inform the operator that the message can no longer be transmitted. Because the instruction T_SEND cannot prevent the connection abort, the error output is the correct behavior for T_SEND. Ignoring this output can be interpreted as an error made by the programmer.

Below, we will always refer to an error even if it is defined behavior.

Different causes for an error

We can distinguish between the following causes for an error:

	Faulty parameter values	Faulty programming	Failure of resources
Description	Errors that are dealt with directly in the instruction.	Programming or access errors that result in the execution of an instruction being aborted.	Errors that are handled by the operating system and for which you can program a reaction in the program code.
Error quality	Handled error	Synchronous error	Asynchronous error
Example	Overflow of an arithmetic instruction	Faulty programming: • Querying a non-existent peripheral input • When accessing an ARRAY by means of a variable index, the value of the index is located outside the valid ARRAY limits	Occurrence of a specific event outside the program code.

	Faulty parameter values	Faulty programming	Failure of resources
Reaction of the program or operating system	Instructions that are linked with the enable output ENO are not executed.	If you have not programmed an error OB, the operating system response depends on the CPU.	1. If an organization block (OB) is not assigned to the event, the operating system executes the default system response when the event occurs. 2. If an organization block (OB) is assigned to the event, it is called.
Mechanisms for error handling in the program	Depending on the instruction, you have different local error handling options [2]: • EN/ENO mechanism • The output parameters: – RET_VAL – STATUS – ERROR	Global error handling [1]: • Program execution error OB • Programming error OB • I/O access error OB Local error handling [2] with the instructions: • GET_ERROR • GET_ERR_ID	Error organization blocks (OB): • Time error (OB 80) • Diagnostic interrupt (OB 82) • Insert/remove module interrupt (OB 83) • Rack error (OB 86) Possible system responses without assigned error OB: • The event is ignored by the operating system. • The CPU changes to STOP mode. • If possible, the error handling takes place locally. If an error OB has been assigned, it will be called in case of a corresponding event. An application example for diagnostics in the user program With evaluation of the errors in the error OBs (https://support.industry.siemens.com/cs/document/98210758?lc=en-WW)

[1] You implement global error handling with the help of organization blocks.
[2] You program local error handling within your program code.

Note

Asynchronous error handling

For a CPU of the S7-1500 series, the error OBs are called asynchronously. This means I/O access errors or programming error OBs may not be processed immediately when the error occurs, but delayed depending on the set priority. If additional errors occur before the I/O access error or the programming error OB was processed completely, no additional I/O access error or programming error OB is called. If you want to prevent I/O access or programming error OBs from being discarded, set the priority correspondingly high.

See also

Example of how to handle program execution errors (Page 167)

1.5.2 Overview of mechanisms for error handling

Overview

There are different mechanisms by which you can track parameter, programming or access errors:

Mechanism	Task	Error handling
Enable input EN or IF instruction	Prevent execution of a program code	Local
Enable output ENO or binary result bit	Indicate an error	
Parameter outputs RET_VAL, STATUS and ERROR		
GET_ERROR and GET_ERR_ID instructions	Respond to an error	
Organization blocks		Global

Local error handling in the case of faulty parameter values

Not only does local error handling let you react to an error immediately after it has occurred, you can also get a specific reaction within your program code. You program the local error handling directly in a program block (OB, FB or FC). It only handles errors that occur within this individual program block.

Advantages of local error handling:

- You can use the error information to program a reaction to the error that occurred in the program block.
- Programmed error evaluation and error reactions do not interrupt the program cycle.
- System performance is not unnecessarily affected by local error handling. If no errors occur, programmed error analyses and reactions are not executed.

The following local error handling options are available:

Error handling method	Validity	Explanation
EN/ENO mechanism [1]	S7-300 / S7-400 / S7-1200 / S7-1500	You can detect and handle specific runtime errors with the help of the enable output ENO. Execution of subsequent instructions depends on the signal state of the enable output. You can avoid program crashes by using the EN/ENO mechanism. The block status is passed on in the form of a Boolean tag. For additional information on the EN/ENO mechanism, refer to: Basics of the EN/ENO mechanism (Page 141)
Output parameters STATUS and ERROR	S7-300 / S7-400 / S7-1200 / S7-1500	Using the STATUS and ERROR parameters as return values of system function blocks (SFBs), you can query block-specific error information. The error information is output in a pre-defined structure. For further information on the output parameters, refer to the descriptions of the individual instructions in the information system.
Output parameter RET_VAL	S7-300 / S7-400 / S7-1200 / S7-1500	You can display general error codes or specific error codes with the help of the output parameter RET_VAL as return value of sequential functions (SFCs). The general error codes refer to any instruction and the specific error codes refer only to a specific instruction. A maximum of one tag of the data type INT or WORD can be output. Additional information on the output parameter RET_VAL is available here: Evaluating errors with output parameter RET_VAL (Page 162)
[1] If the parameters of an instruction did not cause any memory access errors, the associated enable output ENO returns the signal state "1" and the outputs return valid values which you can query.		

Global and local error handling in the case of faulty programming

With the global and local error handling, you can react immediately to errors that have occurred without the CPU changing to "STOP" mode. The following options are available for handling programming and access errors:

Type of global error handling	Validity	Explanation
Program execution error OB (OB 85)	S7-300 / S7-400	If you are not using OB 85, the CPU changes from RUN mode to STOP mode in case of a program execution error and generates an entry in the diagnostics buffer. Additional information on OB 85 can be found here: AUTOHOTSPOT
CPU internal error handling in case of programming and access errors	S7-1200	In the event of an error, the CPU generates an entry in the diagnostics buffer and remains in RUN mode without additional programming effort.
Programming error OB (OB 121)	S7-300/ S7-400 / S7-1500	If you are not using OB 121, the CPU changes from RUN mode to STOP mode in the case of a programming error and generates an entry in the diagnostics buffer. Additional information on OB 121 can be found here: S7-300 / S7-400: AUTOHOTSPOT S7-1500: AUTOHOTSPOT

Type of global error handling	Validity	Explanation
I/O access error OB (OB 122)	S7-300/ S7-400 / S7-1500	S7-300 / S7-400: If you are not using OB 122, the CPU changes from RUN mode to STOP mode in the case of an access error. AUTOHOTSPOT S7-1500: In the case of an I/O access error, the CPU always remains in RUN mode and generates an entry in the diagnostics buffer, even if you have not used OB 122. Additional information on OB 122 can be found here: AUTOHOTSPOT
Additional information on reading out the diagnostics buffer can be found here:AUTOHOTSPOT		

You can integrate local error handling directly in your program code with the help of the GET_ERROR and GET_ERR_ID instructions. You also have the option of receiving detailed information on the error and evaluating it in your program close to the error location. You program the local error handling directly in a program block (OB, FB or FC). It only handles errors that occur within this individual program block.

Type of local error handling	Validity	Explanation
GET_ERROR and GET_ERR_ID instructions	S7-1200/ S7-1500	The instructions give you the opportunity to obtain an error ID or detailed error information and to program a direct response in the program code. When information about the first error is queried, the memory space of the error in the system memory is enabled again. Then, when additional errors occur, information about the next error is output. Additional information on the GET_ERROR and GET_ERR_ID instructions and an overview for prioritizing the errors that occur is available here: Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

When using local error handling, query via the GET_ERROR instruction, the following default reactions exist:

- In the case of a write error: The error is ignored and processing of the program continues.
- In the case of a read error: Processing of the program continues with the substitute value "0" for arithmetic instructions.
- In the case of an execution error: Processing of the instruction is canceled and processing of the program continues with the next instruction.

Advantages of local error handling:

- Error information that you can query and evaluate is stored in the system memory (e. g. with the help of the GET_ERROR and GET_ERR_ID instructions).
- You can use the error information to program a reaction to the error that occurred in the program block.
- Programmed error evaluation and error reactions do not interrupt the program cycle.

- Local error handling has a lower effect than global error handling on the system performance. If no errors occur, programmed error analyses and reactions are not executed.
- If local error handling has been set for a program block, no global error handling is conducted in the case of errors.

Note

To prevent the CPU from switching to STOP mode in the case of an error, all programming and I/O access errors must be captured either by global or local error handling.

Example

A detailed example of local error handling that contains several of the above described options is available here: Example of how to handle program execution errors (Page 167)

See also

EN/ENO mechanism (Page 141)

1.5.3 EN/ENO mechanism

1.5.3.1 Basics of the EN/ENO mechanism

Introduction

You can detect and handle certain runtime errors using the enable output ENO. Execution of subsequent instructions depends on the signal state of the enable output. You can avoid program crashes by using the EN/ENO mechanism. The block status is passed on in the form of a Boolean tag.

The EN/ENO mechanism can be used at two levels:

- For individual instructions (instruction ENO)

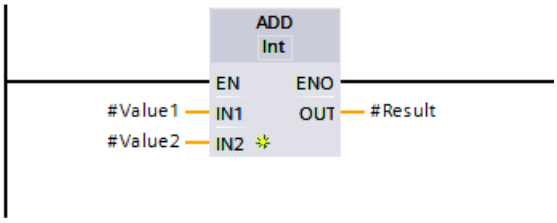

- For program block calls (block ENO)

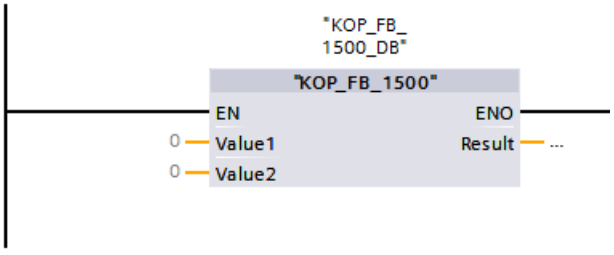

The EN/ENO mechanism is already available in LAD and FBD for calling the simple and advanced instructions in the program code.

With the EN/ENO mechanism you can influence the calling of the following instructions and customize the jumping out from a program block using the instruction "RET: Return". At the same time you can also assign an individual value (0 or 1) to the enable output ENO of the program block. This behavior is generally used for LAD and FBD program blocks. You can also use this procedure for SCL program blocks. You do not necessarily have to use the instruction "RET" to influence the enable output ENO of the SCL program block.

At the network borders the enable output is repeatedly set to the signal state "1". In a LAD program block you can recognize this, for example, by the fact that the left busbar always supplies current, even when the enable output ENO of the last instruction of the previous network returns the signal state "0".

Function of the instruction "RET: Return" (LAD/FBD)

When the result of logic operation = 0 the instruction is not executed and the next network is processed.

When the result of logic operation = 1 the instruction is executed and a return to the calling program block takes place.

The signal state of the block ENO can be determined with four options:

- RLO: RLO = 1 is applied, i.e. the block ENO is set to TRUE.
- TRUE: The block ENO is set to TRUE.
- FALSE: The block ENO is set to FALSE.
- Operand: The signal state of the specified operand determines the block ENO.

Programming and I/O access errors

You cannot evaluate programming and I/O access errors with the EN/ENO mechanism. To do this, use either the global error handling via OBs or the local error handling using the "GET_ERROR" or "GET_ERR_ID" instructions (S7-1200/1500 only). If none of the errors occurred for an instruction, you can evaluate the associated enable output ENO.

Program block calls in all programming languages (S7-300/400)

The following applies to a CPU of the S7-300/400 series: When you call program blocks that do not include any instructions, neither the BR bit nor the ENO enable output are influenced. The signal state of the BR bit remains constant as a result. No statement can be made about the success of the program block call.

1.5.3.2 EN/ENO mechanism is LAD

Overview of the EN/ENO mechanism in LAD

To increase the performance, the EN/ENO mechanism is disabled by default for the instructions. It can be enabled at any time also for individual instructions. For additional information on the enabling and disabling of the EN/ENO mechanism, refer to: Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 144)

For LAD/FBD program block the EN/ENO mechanism only works with an activated enable output ENO at the respective instructions.

EN/ENO mechanism within an instruction.

You can use the enable input EN to make the execution of the instruction dependent on conditions. The instruction is only executed if the signal state at the enable input EN is "1".

You can use the enable output ENO to query runtime errors in an instruction and react to these:

- The enable output ENO has the signal state "1" if no error occurred during the execution.
- Enable output ENO has the signal state "0" if one of the following conditions applies:
  - Enable input EN has the signal state "0".
  - An error occurred during processing.

EN/ENO mechanism with program block calls

All program block calls are provided with an enable input EN and an enable output ENO. This applies to all calling program blocks, regardless of the programming language in which they were created. This means that when you call an STL or SCL program block in which there is no default and pre-configured EN/ENO mechanism, the LAD or FBD program block call still has an EN/ENO mechanism.

You can use the enable input EN to call a program block depending on conditions. The program block is only executed if the signal state at the enable input EN is "1".

You can query the error status of the program block using the enable output ENO:

- The enable output ENO returns the signal state "1" when the program block is processed.
- If, in the case of error, you do not explicitly set the enable output ENO within the called program block to the signal state "0", it retains the signal state "1". Using the instruction "RET: Return" you can set the block ENO to the signal state "0".
  Additional information on using the RET instruction is available here: Influencing the block ENO of an LAD/FBD program block (Page 145)

Enabling and disabling the EN/ENO mechanism in LAD and FBD

In LAD and FBD, certain instructions have an enable output ENO and thus use the EN/ENO mechanism. This allows you to query runtime errors in instructions and react to them. In order to increase the performance of the CPU, the EN/ENO mechanism is disabled in the default setting. This means that you are not initially able to react to runtime errors of the instruction using the ENO value. However, you can enable the EN/ENO mechanism again at any time, if required.

You can enable the EN/ENO mechanism individually for each instruction in order to generate the ENO. If you enable the EN/ENO mechanism for an instruction, other instructions that you subsequently add to your program are also inserted with the EN/ENO mechanism enabled. You can disable the EN/ENO mechanism again at any time if you do not want to use the evaluation of ENO for an instruction. Further instructions that you subsequently add to your program will then be inserted without the EN/ENO mechanism.

Runtime errors do not cause the CPU to go to STOP when the enable output ENO is enabled.

Activating the EN/ENO mechanism

Proceed as follows to activate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to activate the EN/ENO mechanism.
2. Select the "Generate ENO" command from the shortcut menu.
   The ENO value is again generated for the instruction. Additional instructions are then also inserted with the activated enable output.

Deactivating the EN/ENO mechanism

Proceed as follows to deactivate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to deactivate the EN/ENO mechanism.
2. Select the "Do not generate ENO" command from the shortcut menu.
   The ENO value is no longer generated for the instruction. Additional instructions are then also inserted without the activated enable output.

See also

Basics of the EN/ENO mechanism (Page 141)

Influencing the block ENO of an LAD/FBD program block

Description

Using the instruction "RET: Return" you jump out from a program block and hereby influence the signal state of the ENO block.

Procedure

To influence the signal state of the block ENO, follow these steps:

1. Enable the EN/ENO mechanism of an instruction (for example, the addition "ADD").
2. Then use a negation to program the instruction "RET: Return" with signal state FALSE at the enable output ENO.

Result

In case of error (for example, an overflow in the result), the signal state "0" is first returned at the enable output ENO. This signal state "0" becomes signal state "1" using the negation and thus forms the RLO = 1 for executing the instruction "RET", which returns the value FALSE. The block ENO of the program block is hereby FALSE and a return takes place from the program block to the next instruction after the previous program block call. This procedure can be programmed in each network (for example, with multiple mathematical functions etc.).

However, it is not compulsory to program a jump out of the program block. Within a network you can also use the signal state "0" of the enable output ENO of a single instruction to determine that the subsequent instruction is not executed. The block ENO is not hereby influenced.

Note

Influencing of the block ENO

You can only influence the signal state of the block ENO by jumping out of the program block.

Even if the enable output ENO of the last instruction in the last network within a program block has the signal state "0", the block ENO is not influenced.

Example of the use of the EN/ENO mechanism in LAD

Example of an instruction with EN/ENO mechanism

The following example shows the "Add" instruction with EN/ENO circuit and a RET coil (Ret False):

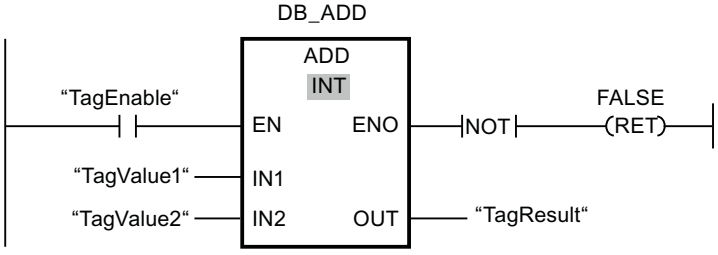

After the NO contact "TagEnable", the enable input EN contains the result of the preceding logic operation:

- If the operand "TagEnable" has the signal state "0", the "Add" instruction is not executed. The enable output ENO is set to signal state "0" and the called program block is exited. The enable output ENO of the calling program block then also has the signal state "0".
- If the operand "TagEnable" has the signal state "1", the enable input "EN has the signal state "1" and the instruction "Add" is executed. The instruction adds two values of the data type INT. If the expected result is outside the value range of INT (16 bits: -32768 to +32767), the instruction still returns a result but it is within the value range of INT. The reason for this is that the 16th bit for INT is the sign bit. This result does not indicate that there was an overflow. Which is why the following applies in case of an addition => ENO:= NOT(OV). If an error occurs during processing of the program block, the enable output ENO is set to signal state "0" and the called program block is exited. The enable output ENO of the calling program block then also has the signal state "0".
- If the operand "TagEnable" has the signal state "1", the enable input "EN has the signal state "1" and the instruction "Add" is executed. If the instruction is executed without errors, the enable output ENO also has the signal state "1" and the output "TagResult" the result.

A detailed description of the instruction "RET: Return" is available here: --(RET): Return (Page 869)

Example of the effect of the enable output ENO

The following example shows you how to use instructions with the ENO enable output enabled and disabled:

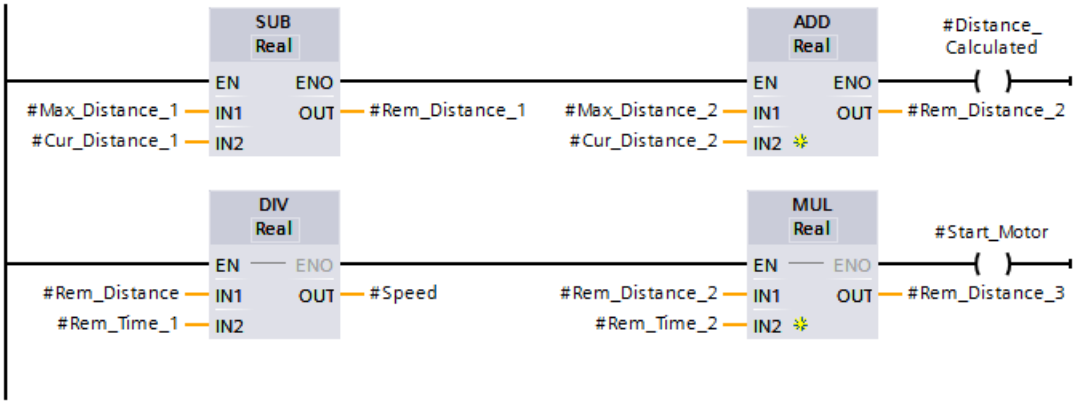

If you have activated the ENO enable output, for example with the SUB instruction, all subsequent instructions are also applied with an activated ENO enable output. If an arithmetic error occurs during the execution of the SUB instruction, the ADD instruction is not executed.

The ENO enable output is disabled for the DIV instruction in the second branch. If a runtime error occurs during execution, the MUL instruction is executed anyway.

Example of a program block call with EN/ENO mechanism

The following example shows the call of the program block with EN/ENO circuit:

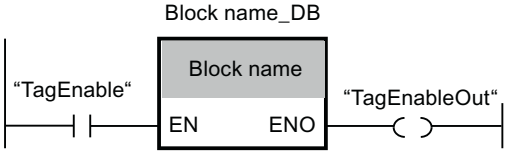

The program block is executed only if the operand "TagEnable" has the signal state "1":

- The enable output ENO has a signal state that depends on what was programmed within the program block.
- If the operand "TagEnable" has the signal state "0", the program block call is not executed. The enable input EN and the enable output ENO both have the signal state "0".

See also

Basics of the EN/ENO mechanism (Page 141)

1.5.3.3 EN-/ENO mechanism in LAD

Overview of the EN/ENO mechanism in LAD

To increase the performance, the EN/ENO mechanism is disabled by default for the instructions. It can be enabled at any time also for individual instructions. For additional information on the enabling and disabling of the EN/ENO mechanism, refer to: Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 149)

For LAD/FBD program block the EN/ENO mechanism only works with an activated enable output ENO at the respective instructions.

EN/ENO mechanism within an instruction.

You can use the enable input EN to make the execution of the instruction dependent on conditions. The instruction is only executed if the signal state at the enable input EN is "1".

You can use the enable output ENO to query runtime errors in an instruction and react to these:

- The enable output ENO has the signal state "1" if no error occurred during the execution.
- Enable output ENO has the signal state "0" if one of the following conditions applies:
  - Enable input EN has the signal state "0".
  - An error occurred during processing.

EN/ENO mechanism with program block calls

All program block calls are provided with an enable input EN and an enable output ENO. This applies to all calling program blocks, regardless of the programming language in which they were created. This means that when you call an STL or SCL program block in which there is no default and pre-configured EN/ENO mechanism, the LAD or FBD program block call still has an EN/ENO mechanism.

You can use the enable input EN to call a program block depending on conditions. The program block is only executed if the signal state at the enable input EN is "1".

You can query the error status of the program block using the enable output ENO:

- The enable output ENO returns the signal state "1" when the program block is processed.
- If, in the case of error, you do not explicitly set the enable output ENO within the called program block to the signal state "0", it retains the signal state "1". Using the instruction "RET: Return" you can set the block ENO to the signal state "0".
  Additional information on using the RET instruction is available here: Influencing the block ENO of an LAD/FBD program block (Page 149)

Enabling and disabling the EN/ENO mechanism in LAD and FBD

In LAD and FBD, certain instructions have an enable output ENO and thus use the EN/ENO mechanism. This allows you to query runtime errors in instructions and react to them. In order to increase the performance of the CPU, the EN/ENO mechanism is disabled in the default setting. This means that you are not initially able to react to runtime errors of the instruction using the ENO value. However, you can enable the EN/ENO mechanism again at any time, if required.

You can enable the EN/ENO mechanism individually for each instruction in order to generate the ENO. If you enable the EN/ENO mechanism for an instruction, other instructions that you subsequently add to your program are also inserted with the EN/ENO mechanism enabled. You can disable the EN/ENO mechanism again at any time if you do not want to use the evaluation of ENO for an instruction. Further instructions that you subsequently add to your program will then be inserted without the EN/ENO mechanism.

Runtime errors do not cause the CPU to go to STOP when the enable output ENO is enabled.

Activating the EN/ENO mechanism

Proceed as follows to activate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to activate the EN/ENO mechanism.
2. Select the "Generate ENO" command from the shortcut menu.
   The ENO value is again generated for the instruction. Additional instructions are then also inserted with the activated enable output.

Deactivating the EN/ENO mechanism

Proceed as follows to deactivate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to deactivate the EN/ENO mechanism.
2. Select the "Do not generate ENO" command from the shortcut menu.
   The ENO value is no longer generated for the instruction. Additional instructions are then also inserted without the activated enable output.

See also

Basics of the EN/ENO mechanism (Page 141)

Influencing the block ENO of an LAD/FBD program block

Description

Using the instruction "RET: Return" you jump out from a program block and hereby influence the signal state of the ENO block.

Procedure

To influence the signal state of the block ENO, follow these steps:

1. Enable the EN/ENO mechanism of an instruction (for example, the addition "ADD").
2. Then use a negation to program the instruction "RET: Return" with signal state FALSE at the enable output ENO.

Result

In case of error (for example, an overflow in the result), the signal state "0" is first returned at the enable output ENO. This signal state "0" becomes signal state "1" using the negation and thus forms the RLO = 1 for executing the instruction "RET", which returns the value FALSE. The block ENO of the program block is hereby FALSE and a return takes place from the program block to the next instruction after the previous program block call. This procedure can be programmed in each network (for example, with multiple mathematical functions etc.).

However, it is not compulsory to program a jump out of the program block. Within a network you can also use the signal state "0" of the enable output ENO of a single instruction to determine that the subsequent instruction is not executed. The block ENO is not hereby influenced.

Note

Influencing of the block ENO

You can only influence the signal state of the block ENO by jumping out of the program block.

Even if the enable output ENO of the last instruction in the last network within a program block has the signal state "0", the block ENO is not influenced.

Example of the use of the EN/ENO mechanism in FBD

Example of an instruction with EN/ENO mechanism

The following example shows the "Add" instruction with EN/ENO circuit and a RET coil (Ret False):

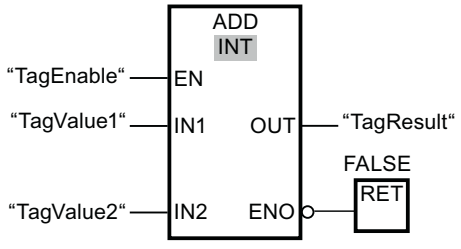

After the NO contact "TagEnable", the enable input EN contains the result of the preceding logic operation:

- If the operand "TagEnable" has the signal state "0", the "Add" instruction is not executed. The enable output ENO is set to signal state "0" and the called program block is exited. The enable output ENO of the calling program block then also has the signal state "0".
- If the operand "TagEnable" has the signal state "1", the enable input "EN has the signal state "1" and the instruction "Add" is executed. The instruction adds two values of the data type INT. If the expected result is outside the value range of INT (16 bits: -32768 to +32767), the instruction still returns a result but it is within the value range of INT. The reason for this is that the 16th bit for INT is the sign bit. This result does not indicate that there was an overflow. Which is why the following applies in case of an addition => ENO:= NOT(OV). If an error occurs during processing of the program block, the enable output ENO is set to signal state "0" and the called program block is exited. The enable output ENO of the calling program block then also has the signal state "0".
- If the operand "TagEnable" has the signal state "1", the enable input "EN has the signal state "1" and the instruction "Add" is executed. If the instruction is executed without errors, the enable output ENO also has the signal state "1" and the output "TagResult" the result.

A detailed description of the instruction "RET: Return" is available here: RET: Return (Page 1217)

Example of the effect of the enable output ENO

The following example shows you how to use instructions with the ENO enable output enabled and disabled:

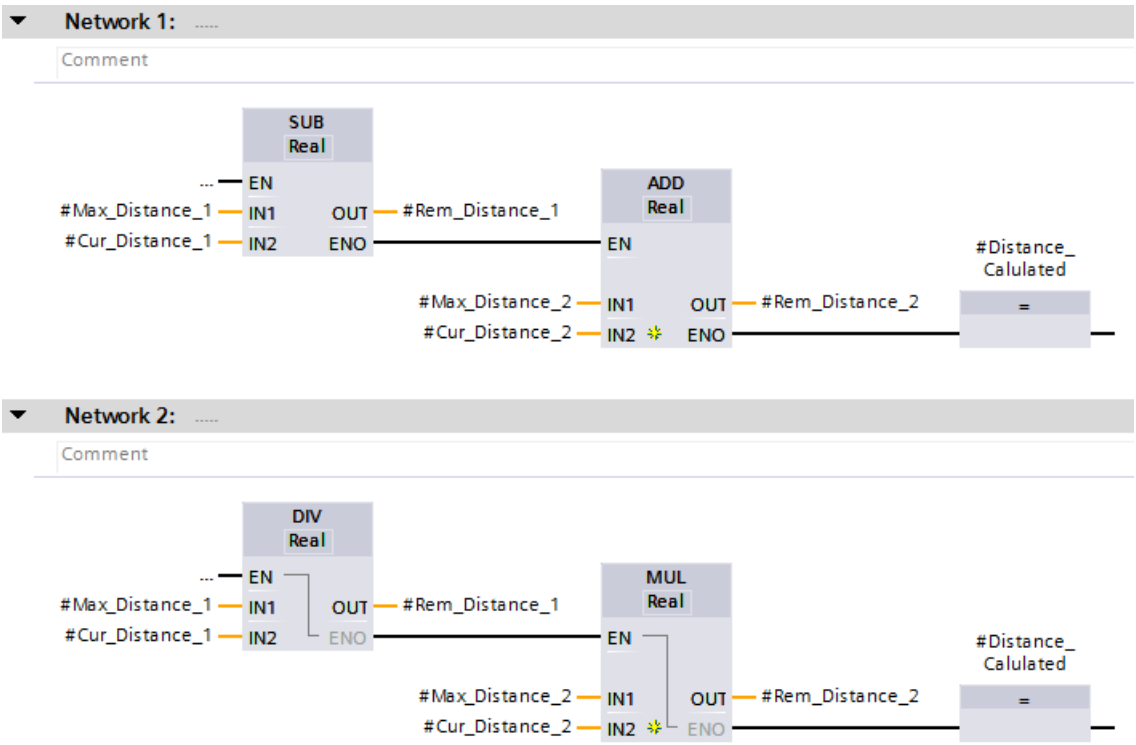

If you have activated the ENO enable output, for example with the SUB instruction, all subsequent instructions are also applied with an activated ENO enable output. If an arithmetic error occurs during the execution of the SUB instruction, the ADD instruction is not executed.

The enable output ENO is disabled for the instruction DIV. If a runtime error occurs during execution, the MUL instruction is executed anyway.

Example of a program block call with EN/ENO mechanism

The following example shows the call of the program block with EN/ENO circuit:

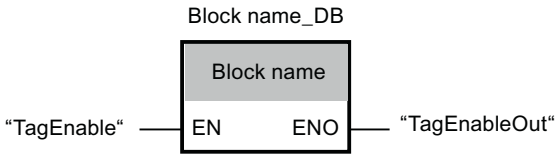

The program block is executed only if the operand "TagEnable" has the signal state "1":

- The enable output ENO has a signal state that depends on what was programmed within the program block.
- If the operand "TagEnable" has the signal state "0", the program block call is not executed. The enable input EN and the enable output ENO both have the signal state "0".

See also

Basics of the EN/ENO mechanism (Page 141)

1.5.3.4 EN/ENO mechanism in STL (S7-1500)

Overview of the EN/ENO mechanism in STL (S7-1500)

EN/ENO mechanism within an instruction.

The EN/ENO mechanism is not available for individual instructions. It is mapped by language-specific instruction sequences and use of the status word (BR bit).

For additional information on the status word, refer to: Status word for S7-1500 (Page 186)

Additional information is available here: Example of the simulation of the EN/ENO mechanism in STL (Page 153)

EN/ENO mechanism with program block calls

A program block that you call from an STL program block is not provided with the EN and ENO parameters. Regardless of the programming language in which the program block was created, you can get an error statement to the calling STL program block by using the BR bit of the status word.

You can evaluate the error status of the called program block by linking the BR bit of the status word with the result of logic operation (RLO). The RLO has the signal state "1" as soon as the

program block execution of the called program block starts. If you do not explicitly reset the RLO to "0" after the execution of the actions, it retains the signal state "1". To signal an error statement back to the calling program block, you have to explicitly reset the RLO to the signal state "0". The error statement is set with the instructions "SAVE: Save RLO in BR bit" and "JNB: Jump if RLO = 0 and save RLO".

An error analysis of the BR bit is available in STL. The BR bit is the basis for generating the enable output ENO in the programming languages LAD, FBD and SCL.

The value "0" in the BR bit of the status word indicates that an error occurred during an STL program block call (calling program block is an STL program block).

CPU execution of the instruction	BR bit	Return value	Sign of the integer
With error(s)	0	Less than "0"	Negative (sign bit is "1")
Without error	1	Greater than or equal to "0"	Positive (sign bit is "0")

For additional information on the BR bit, refer to: Evaluating errors with output parameter RET_VAL (Page 162)

Example of the simulation of the EN/ENO mechanism in STL (S7-1500)

Example of a program sequence with EN/ENO mechanism

The following example shows a program section for adding values with EN/ENO mechanism:

STL	Description
A "TagEnable"	// Query the signal state of the "TagEnable" operand for "1" and AND with the current RLO.
JNB MyLABEL	// Evaluation of the enable input EN
	// If RLO = "0", jump to jump label "MyLABEL" and save the current RLO in the BR bit. The following actions are not executed.
	// If the RLO = "1", execute the following actions.
L "Tag_Input_1"	// Load first value for addition.
L "Tag_Input_2"	// Load second value for addition.
+I	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".
AN OV	// Query whether an overflow has occurred.
SAVE	// Transfer signal state of the RLO to the BR bit.
CLR	/// Reset RLO to "0" and end logic sequence.
MyLABEL: A BR	// Jump label "MyLABEL"
	// Query BR bit.

STL	Description
= "Tag_Output"	// Assign signal state of the RLO to the "Tag_Output" operand.

The query of the operand "TagEnable" provides the result of the preceding logic operation (RLO). The instruction "JNB: Jump if RLO = 0 and save RLO" saves the RLO in the BR bit. Additionally, the instruction evaluates the RLO and executes the following actions depending on the signal state:

- If the RLO is "0", the processing of the program is continued at the jump label "MyLABEL" with the query of the BR bit. The addition is not executed. The current RLO is assigned with the operand "Tag_Output".
- If the RLO is "1", the addition is executed. A query of the overflow bit (OV) shows whether an error occurred during the addition. The query result is saved in the BR. The instruction "CLR: Reset RLO to 0" resets the RLO to "0" and ends the logic sequence. The BR bit is then queried and assigned to the operand "Tag_Output". The signal state of the BR bit, and the operand "Tag_Output" shows whether the addition was executed without any error.

Example of a program block call with EN/ENO mechanism

The following example shows the call of the program block with EN/ENO circuit:

STL	Description
A "TagEnable"	// Query the signal state of the "TagEnable" operand for "1" and AND with the current RLO.
JNB MyLABEL	// Evaluation of the enable input EN
	// If RLO = "0", jump to jump label "MyLABEL" and save the current RLO in the BR bit. The following actions are not executed.
	// If the RLO = "1", execute the following actions.
CALL "Block name", "Block name_DB"	// The program block is called.
MyLABEL: A BR	// Jump label "MyLABEL"
	// Query BR bit and link with the RLO after AND.
= "Tag_Output"	// Assign signal state of the RLO to the "Tag_Output" operand.

See also

Basics of the EN/ENO mechanism (Page 141)

1.5.3.5 EN/ENO mechanism in SCL

Overview of the EN/ENO mechanism in SCL

To increase performance, the EN/ENO mechanism is disabled by default. However, it can be enabled at any time using the block attribute "Set ENO automatically". For additional information on the enabling and disabling of the EN/ENO mechanism, refer to: Enabling and disabling the ENO mechanism in SCL (Page 155)

In addition to enabling the block attribute "Set ENO automatically", you can also influence the enable output EN using an IF instruction. Additional information is available here: Influencing enable output ENO using an IF instruction (Page 157)

EN/ENO mechanism within an instruction.

The EN/ENO mechanism is optional for instructions.

Additional information is available here: Using enable output ENO with SCL instructions (Page 156)

EN/ENO mechanism with program block calls

The call of an SCL program block is also executed unconditionally, regardless of whether the EN/ENO mechanism is enabled or not. However, you can link the call to an IF instruction and then make the call dependent on various conditions. One possible condition can, for example, be the block ENO of the SCL program block. However, the EN/ENO mechanism then has to be enabled for this.

If the block ENO returns the signal state "0" because of a runtime error in a previous instruction, the block call is not executed.

Additional information is available here: Influencing enable output ENO of an SCL program block (Page 157)

Enabling and disabling the ENO mechanism in SCL

Description

The EN/ENO mechanism checks during runtime whether errors occur in the processing of certain instructions. If a runtime error occurs the block ENO is set to "0".

Enabling EN/ENO mechanism in the block properties

Proceed as follows to activate the EN/ENO mechanism in the block properties:

1. Open the "Program blocks" folder in the project navigator.
2. Right-click the SCL block whose properties you want to display.
3. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the block opens.
4. Click the "Attributes" group in the area navigation.

5. Activate the property "Set ENO automatically".
6. Confirm your entries with "OK".

Result: The EN/ENO mechanism is activated only for the selected SCL program block.

Activate EN/ENO mechanism in the program properties

Proceed as follows to activate the EN/ENO mechanism in the program properties:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the working area.
2. In the area navigation, select the "PLC programming" group.
3. Select the group "SCL (Structured Control Language)".
4. Activate the property "Set ENO automatically".

Result: The EN/ENO mechanism is activated by default for all new SCL program blocks.

Using enable output ENO with SCL instructions

Description

With each SCL instruction, such as a mathematical function, you must query the enable output ENO in order to be able to work with it.

Procedure

To query the enable output ENO, proceed as follows:

1. Activate the EN/ENO mechanism.
2. After the SCL instruction, query the enable output ENO (for example, #MyOutputBool := ENO;)
   If the error case occurs, for example an overflow in the result, returns the enable output ENO returns the signal state "0". With this signal state you can continue to work and, for example, program the instruction "RETURN", a fault display or a substitute value.
3. Reset the signal state of the enable output ENO to "1" (for example, ENO :=1;) before processing the next instruction.
   The signal state for the next error case can hereby be reset to "0". If you do not do this, the signal state of the enable output ENO remains "0".

Note

Jump to an SCL block.

With a jump to an SCL block, the signal state of the enable output ENO is automatically set to "1".

Influencing enable output ENO of an SCL program block

Description

The last instruction ENO within an SCL program block always influences the block ENO.

In addition, you can jump out of the program block and at the same time assign the value "1" or "0" to the block ENO using the instruction "RETURN".

Procedure

To influence the signal state of the block ENO, follow these steps:

1. Activate the EN/ENO mechanism.
2. Then program:
```
IF ENO := 0 THEN
RETURN;
END_IF;
```

Result

In case of error (for example, an overflow in the result), the signal state "0" is initially returned at the enable output ENO, such as an addition instruction. The block ENO is hereby FALSE and a jump takes place from the program block to the next instruction after the previous program block call. This procedure can be programmed multiple times in an SCL program block (for example, with multiple mathematical functions, etc.).

However, it is not compulsory to program a jump out of the program block. Within the SCL program block you can also use the signal state "0" of the enable output ENO of a single instruction to determine that the subsequent instruction is not executed. For this purpose, the instruction ENO is linked with the following instruction in a subsequent IF instruction.

Note

Influencing of the block ENO

With an SCL program block the signal state of the last instruction ENO is forwarded at the end of the block to the block ENO

The instruction ENO thus directly influence the block ENO.

Influencing enable output ENO using an IF instruction

Description

If you influence the enable output EN using an IF instruction, you do not need to activate the block attribute "Set ENO automatically".

Procedure

To influence the ENO enable output using an IF instruction, proceed as follows:

1. Program an IF instruction.
2. Use the enable output ENO either within the IF instruction or with ELSE.

Result

Depending on a Boolean value or a value comparison, the enable output ENO within the IF instruction is set to signal state "1" (ENO := 1;) or with ELSE to the signal state "0" (ENO := 0;).

Example of the use of the EN/ENO mechanism in SCL

Example of an instruction with EN/ENO mechanism

To use the EN/ENO mechanism for instructions, you have to activate the block property "Set ENO automatically". The following example shows the use of the enable output ENO for the operation "a + b":

```
"MyOutputREAL":= #a + #b;
"MyOutputBOOL":= ENO;
```

If the operation "a/b" is executed error-free, the operand "MyOutputBool" has the signal state "1". In case of error, the enable output ENO has the signal state "0".

Example of a program block call with EN/ENO mechanism

The following example shows the call of the program block. For this purpose the EN/ENO mechanism must be activated and an instruction that influences the block ENO must be programmed beforehand:

```
IF ENO := 1 THEN
"Block name_DB"();
END_IF;
```

The program block is only called and executed if the enable output ENO has the signal state "1". The called program block also returns a block ENO if its EN/ENO mechanism is activated. This block Eno is then further used in the called program block.

See also

Basics of the EN/ENO mechanism (Page 141)

1.5.3.6 EN/ENO mechanism in GRAPH (S7-1500)

Overview of the EN/ENO mechanism in GRAPH (S7-1500)

EN/ENO mechanism within an instruction.

You have no access to the enable output ENO of the instructions, which means you cannot influence the status of the enable output ENO at the GRAPH function block. However, with conversions, mathematical functions or LAD/FBD instructions, for example, you can have the enable output ENO displayed in the program status.

EN/ENO mechanism with program block calls

You can use the enable input EN to call a program block depending on conditions. The program block is only executed if the signal state at the enable input EN is "1".

You can query the error status of the program block using the enable output ENO:

- The enable output ENO has the signal state "1", as soon as the called program block has been executed error-free.
- The enable output ENO has the signal state "0" if an error occurred during the execution of the called program block.
- The enable output ENO cannot be explicitly set or reset.
- The enable output ENO is not influenced by the ENO of the instructions.

Activating and deactivating EN/ENO mechanism in GRAPH (S7-1500)

Description

The status of the enable output ENO is displayed during testing with program status. This has the value TRUE if the action was successful and FALSE if the action failed.

The display option of the status of the enable output ENO is available at the following positions in the GRAPH program block:

- Permanent pre-instructions
- Sequence view > Actions
- Permanent post-instructions

Enabling EN/ENO mechanism in the block properties

Proceed as follows to activate the EN/ENO mechanism in the block properties:

1. Open the "Program blocks" folder in the project navigator.
2. Right-click the GRAPH block whose properties you want to display.
3. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the block opens.

4. Click the "Attributes" group in the area navigation.
5. Activate the property "Set ENO automatically".
6. Confirm your entries with "OK".

Result: The EN/ENO mechanism is activated only for the selected GRAPH block.

Example of the EN/ENO mechanism in GRAPH (S7-1500)

Example of a program status with enable output ENO

The following example shows the program status of the enable output ENO in the sequence view under actions:

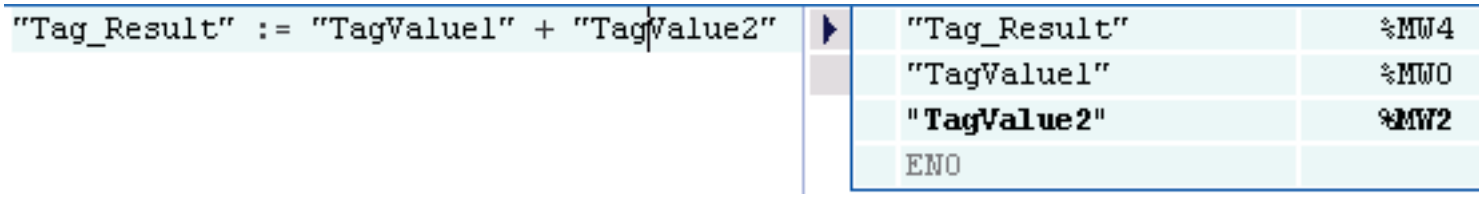

If an error occurs during the execution, the enable output ENO has the signal state FALSE.

Example of a program block call

The following example shows the call of the GRAPH program block in a LAD program block with EN/ENO connected:

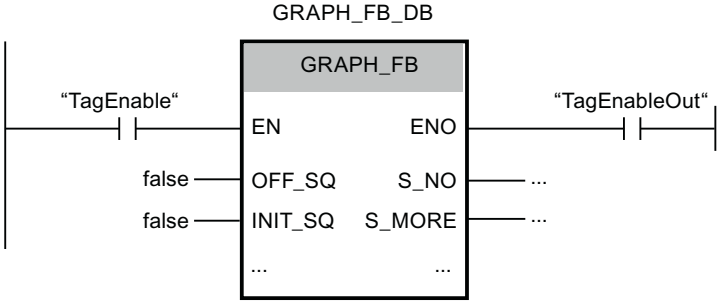

The program block is executed only if the operand "TagEnable" has the signal state "1":

- If the operand "TagEnable" has the signal state "0", the called program block call is not executed. Both the enable input EN and the enable output ENO have the signal state "0".
- If the operand "TagEnable" has the signal state "1", the enable input "EN has the signal state "1" and the called program block is executed. The enable output ENO has a signal state that depends on what was programmed within the program block.

See also

Basics of the EN/ENO mechanism (Page 141)

1.5.3.7 EN/ENO mechanism in blocks with different network languages (S7-1200, S7-1500)

Description

You can also use the EN/ENO mechanism in blocks with different network languages. Each programming language displays the error status differently:

- SCL has an ENO tag that stores the error status and can be queried. Direct access to this tag is only possible with SCL.
- LAD/FBD/STL have no special tag for ENO. However, you can read the error status for STL from the BR bit, and query it via the RET coil for LAD/FBD.

The following rules apply to reading out the error status for the entire block:

- The last network in the block is a LAD/FBD network:
  If you do not use RET coils, the error status is TRUE by default.
- The last network in the block is an STL network:
  The BR bit determines the error status. The BR bit can be edited in STL networks using the BR tab.
- The last network in the block is an SCL network:
  The ENO tag determines the error status of the block.

Influencing of the block ENO

For LAD/FBD program blocks with multiple networks that could consist of different programming languages (such as LAD, FBD, STL or SCL), the last processed network determines the signal state of the block ENO.

If the last processed network of such a program block is a LAD or FBD network, the signal state of the block ENO is not necessarily influenced. With an instruction in which the EN/ENO mechanism is activated, it is only influenced if the enable output ENO has the signal state "0" and it is possible to return for the program block using the instruction "RET: Return". If this instruction was not programmed, the block ENO always has the signal state "1". The signal state "0", if any, of an instruction ENO has no influence on the block ENO, as the signal state "1" is always set at the start of a network.

This applies also to a LAD or FBD network that is before another network (LAD, FBD or SCL). The signal state of the block ENO is influenced only using the instruction "RET: Return". Otherwise, the signal state of the block ENO is reset to "1" at the transition to the network.

If the last network of a mixed program block is an SCL network, the block ENO is always influenced by this.

See also

Basics of the EN/ENO mechanism (Page 141)

1.5.4 Evaluating errors with output parameter RET_VAL

Basics on error analysis of library blocks (SFB and SFC)

Two additional error evaluation options are available in addition to the output parameter RET_VAL:

- With the EN/ENO mechanism (LAD, FBD, and SCL)
  For additional information on the EN/ENO mechanism, refer to: Basics of the EN/ENO mechanism (Page 141)
- With the BR bit (binary result bit) of the status word (STL)
- With the output parameter RET_VAL (return value)

The enable output ENO only informs you that an error has occurred. If you want to know which error occurred exactly, the output parameter RET_VAL will provide more information. You can use it to determine whether or not the CPU was able to execute the instruction successfully. In the case of an error, you will also learn why the instruction could not be executed successfully.

The section below includes detailed information on the two error evaluation options RET_VAL and BR bit.

Recommendation for sequence of error analyses

Before evaluating the instruction-specific output parameters (for example, OUT), you should always follow the steps below:

1. First evaluate the enable output ENO or the BR bit of the status word in STL.
2. Then check the output parameter RET_VAL.

If the enable output ENO or the BR bit indicates that an error has occurred during the execution of the instruction or if output parameter RET_VAL contains a general error code, the instruction-specific output parameters return an invalid value.

If a general error occurs with the output parameter RET_VAL, this is only indicated by the value "0" in the BR bit of the status word. The return value is of the data type integer (INT). The relationship of the return value to the value "0" indicates whether or not an error occurred during execution of the instruction.

General and specific error codes (RET_VAL)

The output parameter RET_VAL has two types of error codes:

- A general error code that all instructions can output and
- A specific error code that an instruction can output depending on its specific functions.

The data type of the output parameter RET_VAL is an integer (INT). The error codes of the instruction are grouped according to hexadecimal values. If you want to examine a return value and compare the value with the error codes listed in this documentation, then display the error code in hexadecimal format.

You can write your program so that it reacts to the errors that occur during execution of an instruction. This way you prevent further errors occurring as a result of the first error.

Note

Error during supply of the input parameters

When the instruction containing the RET_VAL parameter is executed and errors during supply of the input parameters occur, the RET_VAL parameter outputs an invalid error code and the output parameters of the instruction must not be evaluated.

The figure below shows the structure of a system function error codes in hexadecimal format:

Error code, e.g. W#16#8081

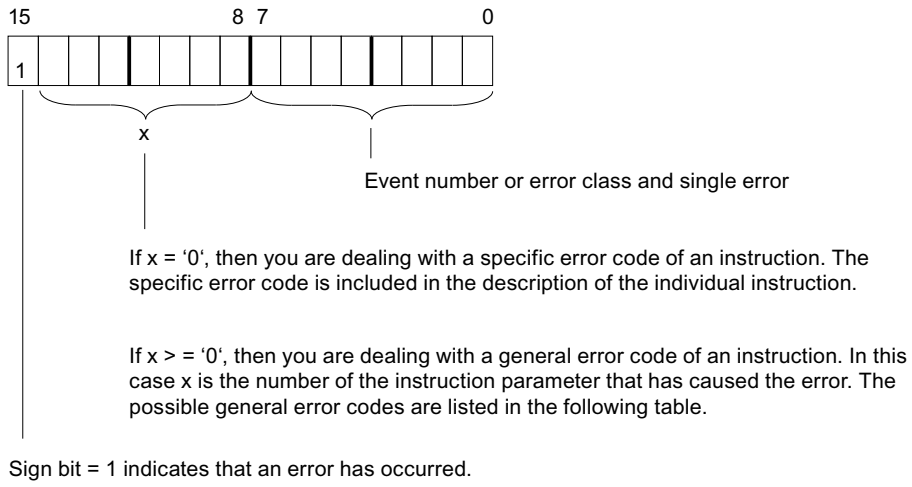

It is easy for you to determine whether an error has occurred because the output parameter RET_VAL is of the type INT:

- If the value is < 0, there is an error.
- If the value is = 0, no error has occurred.
- If the value is > 0, no error has occurred but the instruction was not executed successfully. This is used in asynchronous instructions, for example, to indicate that processing of the instruction was started but is not yet complete. See the return values of T_SEND or WRIT_DBL.

General error codes

The general error code indicates errors that can occur in all instructions. A general error code consists of the following two numbers:

- A parameter number from 1 to 111, where 1 indicates the first parameter of the called instruction, 2 the second parameter, and so forth.
- An event number between 0 and 127. The event number indicates an error.

The figure below shows the structure of a general error code:

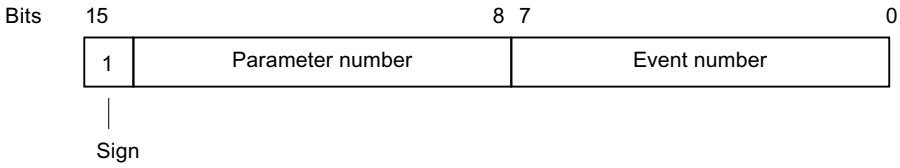

Note

Reaction to a general error code

If a general error code was entered in the output parameter RET_VAL, the following situations are possible:

- The action associated with the instruction was started or has already been completed.
- A specific instruction error occurred when the action was performed. As a result of a general error that occurred later, the specific error could, however, no longer be indicated.

The following table explains the general error codes of a return value. The error code is shown in hexadecimal format. The letter x in each code number is simply a placeholder and represents the number of the system function parameter that caused the error:

Error code (W#16#...)	Description
8x01	Illegal syntax ID at an VARIANT parameter
8x22	Range length error when reading a parameter.
8x23	Range length error when writing a parameter. This error code indicates that the parameter x is located either entirely or partly outside the range of an address, or that the length of a bit range is not a multiple of 8 with an VARIANT parameter.
8x24	Range error when reading a parameter.
8x25	Range error when writing a parameter. This error code indicates that the parameter x is located in a range that is illegal for the system function. Refer to the descriptions of the individual functions for information about the illegal ranges.
8x26	The parameter contains a timer cell number that is too high. This error code indicates that the timer cell specified in parameter x does not exist.
8x27	The parameter contains a counter cell number that is too high (counter number error). This error code indicates that the counter cell specified in parameter x does not exist.
8x28	Alignment error when reading a parameter.
8x29	Alignment error when writing a parameter. This error code indicates that the reference to parameter x is an operand with bit address that is not equal to 0.
8x30	The parameter is located in a read-only global DB.
8x31	The parameter is located in a read-only instance DB. This error code indicates that parameter x is located in a read-only data block. If the data block was opened by the system function itself, the system function always returns the value W#16#8x30.
8x32	The parameter contains a DB number that is too high (DB number error).

Error code (W#16#...)	Description
8x34	The parameter contains an FC number that is too high (FC number error).
8x35	The parameter contains an FB number that is too high (FB number error). This error code indicates that parameter x contains a block number higher than the highest permitted number.
8x3A	The parameter contains the number of a DB that is not loaded.
8x3C	The parameter contains the number of an FC that is not loaded.
8x3E	The parameter contains the number of an FB that is not loaded.
8x42	An access error occurred while the system was attempting to read a parameter from the peripheral input area.
8x43	An access error occurred while the system was attempting to write a parameter to the peripheral output area.
8x44	Error in the nth (n > 1) read access after an error occurred.
8x45	Error in the nth (n > 1) write access after an error occurred. This error code indicates that access to the required parameter is denied.
8x7F	Internal error This error code indicates an internal error at parameter x.

Specific error codes

Some instructions have a return value that provides an error code specific for the instruction. A specific error code indicates errors that can occur only in specific instructions.

A specific error code consists of the following two numbers:

- An error class from 0 to 7.
- An error number from 0 to 15.

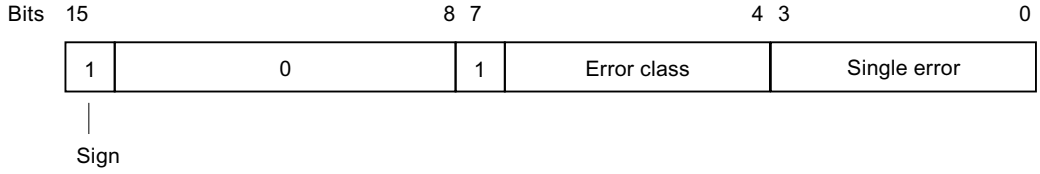

For further information on specific error codes, refer to the descriptions of the individual instructions in the information system.

See also

Difference between synchronous and asynchronous instructions (Page 596)

1.5.5 Use of the instructions GET_ERROR and GET_ERR_ID

Introduction

Local error handling makes it possible to query the occurrence of errors within a program block and evaluate the associated error information. You can set local error handling for organization blocks (OBs), function blocks (FBs), and functions (FCs). If local error handling is enabled, the system reaction is ignored.

You can read out the error number from the error information of the instruction GET_ERR_ID. You can find out, for example, which parameter caused an access error from the error information of the instruction GET_ERROR. In order for the instructions to output the required error information, they must be programmed in the user program for each individual program block from which potential errors are to be evaluated. If you work with the instructions, no error OB is called and there is no entry in the diagnostics buffer. With this method of error handling, you actively intervene in the program sequence by programming a reaction to the errors that occurred. Because errors can occur at any location within the program block, we recommend that you integrate the instruction at the end of the program block.

The instructions GET_ERROR and GET_ERR_ID differ in the amount of error information that is output with each one.

As soon as you have integrated one of the two instructions in your program code, the "Handle errors within block" check box is selected in the inspector window under "Properties > Attributes". This setting cannot be edited in the Inspector window. Local error handling can be deactivated by deleting the inserted instructions for local error handling.

Note

Block property "Handle errors within block"

This setting is not applied by a calling block, nor is it transferred to called program blocks. The system settings still apply to higher-level and lower-level program blocks, provided dedicated local error handling has not been programmed for these blocks.

Error output priorities

In local error handling, information about the first error that occurred is displayed using the instructions GET_ERROR or GET_ERR_ID. If multiple errors occur at the same time while an instruction is being executed, these errors are displayed according to their priority. The following table shows the priority of different types of errors:

Priority	Error type
1	Error in the program code
2	Missing reference
3	Invalid range
4	DB does not exist
5	Operands are not compatible
6	Width of specified area is not sufficient
7	Timers or counters do not exist
8	No write access to a DB

Priority	Error type
9	I/O error
10	Instruction does not exist
11	Block does not exist
12	Invalid nesting depth

The highest priority is 1 and the lowest priority is 12.

Additional information

For additional information on the instructions, refer to the information system under PLC programming > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions. The instructions are available in the programming languages LAD/FBD/STL/SCL and GRAPH.

See also

GET_ERROR: Get error locally (Page 1226)

GET_ERR_ID: Get error ID locally (Page 883)

GET_ERR_ID: Get error ID locally (Page 1230)

GET_ERROR: Get error locally (Page 879)

1.5.6 Example of how to handle program execution errors

Introduction

The local error handling options can be programmed individually or in combination with one another. To ensure that every error scenario that may occur within your program is recognized, we recommend a combination of local error handling options, as shown in the example below.

For a more precise error analysis, in addition to the RET_VAL output parameter you can also use the "GET_ERROR" or "GET_ERR_ID" instruction. These options provide you with error codes whose detailed explanations are available in the descriptions of the respective instructions.

There are also error scenarios in which the RET_VAL output parameter does not output a valid error code. If an access error occurs while reading an input parameter, for example, the outputs of the instruction are no longer written, because the execution of the instruction is interrupted. In this case, we recommend that you integrate the two instructions "GET_ERROR" and "GET_ERR_ID" in your program because they provide reliable error information even when this type of error occurs.

WARNING

Access error during reading of an input parameter

The RET_VAL parameter does not return a valid error code and no detailed error information is output in the diagnostics buffer.

A first indicator for an error can be either the BR bit of the status word or the ENO enable output. If these return signal state "0", there is an error in the execution of the instruction. Signal state "1" indicates that there is no error and no further error analysis is necessary, except when a memory access error is involved. In this case, signal state "1" can also indicate an error.

Procedure

The following example shows you how to recognize an access error during reading of an input parameter:

1. Declare the block interface of your program block as follows:

Interface			
	Name	Data type	Default value
1	Input		
2	SrcField	Array [1..100] of Real	
3	index	DInt	0
4	<Add new>		
5	Output		
6	TagOutput	Real	0.0
7	Test2	Bool	false
8	TagRet_Val	Int	0
9	DestField	Array [0..20] of Real	
10	Test3	Bool	false
11	<Add new>		
12	InOut		
13	Static		
14	Temp		
15	TagError_ID	Word	

2. Write the following program code:

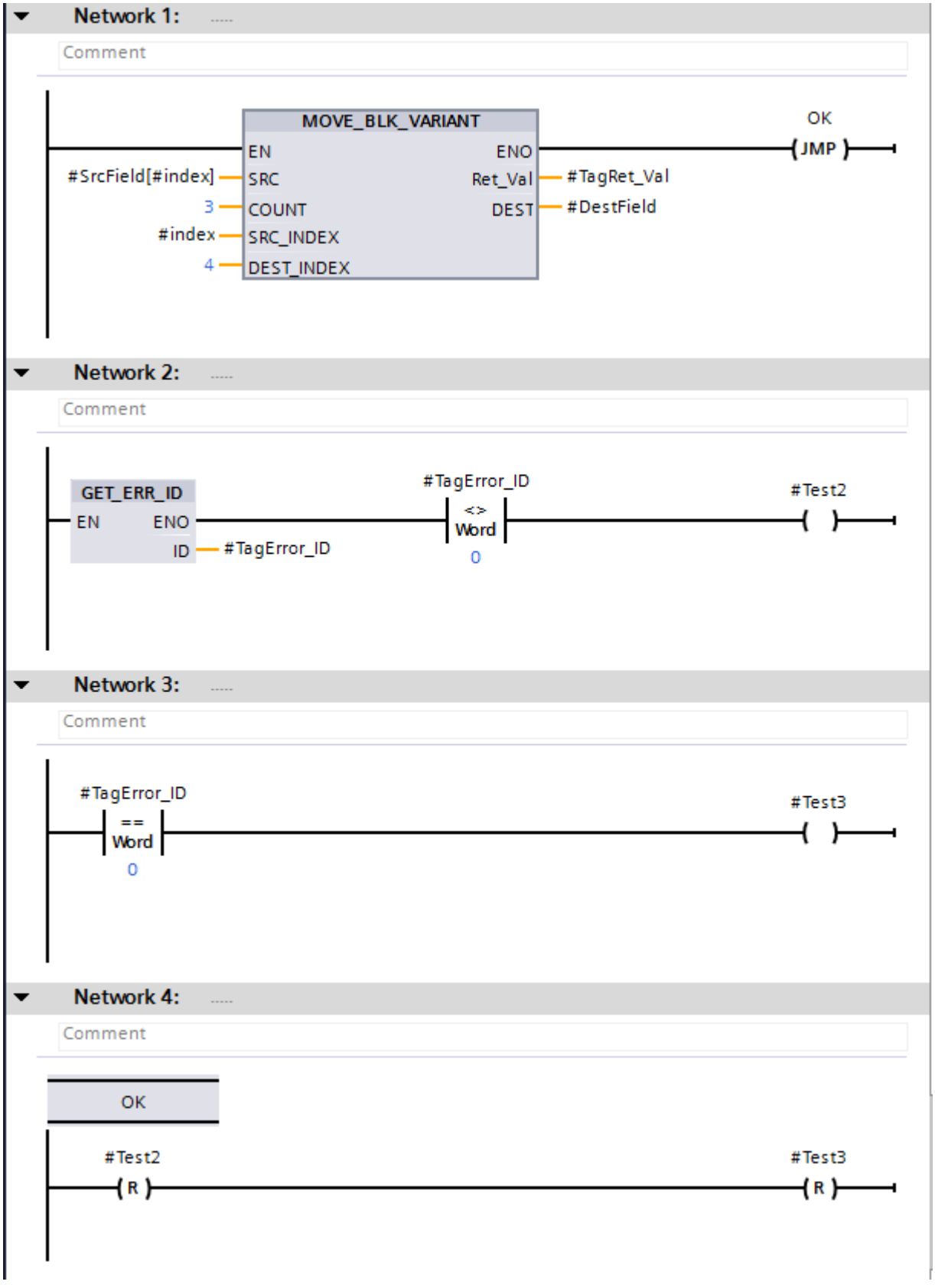
Network 1:
Comment
MOVE_BLK_VARIANT
EN
ENO
#SrcField[#index]
SRC
Ret_Val
#TagRet_Val
3
COUNT
DEST
#DestField
#index
SRC_INDEX
4
DEST_INDEX
OK
JMP
Network 2:
Comment
GET_ERR_ID
EN
ENO
ID
#TagError_ID
#TagError_ID
<>
Word
0
#Test2
Network 3:
Comment
#TagError_ID
==
Word
0
#Test3
Network 4:
Comment
OK
#Test2
R
#Test3
R

In network 1, the instruction "MOVE_BLK_VARIANT": Move block" is called. The "SrcField" source area is accessed using a variable index in the SRC parameter. If no errors occur during execution of the instruction, the ENO enable output returns signal state "1", and program execution jumps to network 4 and continues there.

If an access error occurs during execution of the instruction, for example, due to the variable index, the instruction "GET_ERR_ID: Get error ID locally" in network 2 returns an error ID. The comparison of the error ID for "UNEQUAL" with the value "0" in network 2 returns the result #Test2 = TRUE. The comparison of the error ID for "EQUAL" with the value "0" in network 3 returns the result #Test3 = TRUE.

The #TagRet_Val operand at the RET_VAL output parameter returns no valid error code in this case.

Exceptions

There are some instructions, however, for which you cannot program the error handling as described in the example above. These includes the following instructions:

- Instructions that generally have no EN/ENO mechanism
- Instructions in which the ENO was disabled
- S_COMP
- PEEK, PEEK_BOOL, POKE, POKE_BOOL and POKE_BLK

The BR bit or the ENO enable output is set to TRUE in these instructions even if it may result in an access error.

The following example shows how you can program reliable error handling in the STL programming language:

	Name	Data type
1	Input	
2	StringArray	Array[0..10] of String
3	index	DInt
4	Output	
5	TagResult	Bool
6	InOut	
7	Static	
8	Temp	
9	TagError_ID	Word
10	Constant	

STL	Explanation
SET L 0 T #Tag_ErrorID	// The #Tag_ErrorID operand is initialized with "0".
CALL S_COMP	// The instruction is called.

STL	Explanation
`src_type := String`	`// Data type of parameters IN1 and IN2`
`relation := EQ`	`// The comparison type of the instruction`
`IN1 := #StringArray.THIS[#index]`	`// Variable access to the ARRAY component.`
`IN2 := 'STRING'`	`// The two values are compared with each other.`
`OUT := #TagResult`	`// If both values are identical, the #TagResult operand receives the signal state "1".`
`A BR`	`// The BR bit is queried.`
`CALL GET_ERR_ID`	`// The instruction is called.`
`RET_VAL := #Tag_ErrorID`	`// The instruction outputs an error code in the case of an access error.`

Even if the BR bit has signal state "1", the access error is detected. You can query the error code by evaluating the #Tag_ErrorID operand of instruction "GET_ERR_ID: Get error ID locally".

1.6 Program flow control

1.6.1 Status word for S7-300, S7-400 (S7-300, S7-400)

1.6.1.1 Basic information on the status word (S7-300, S7-400)

Description

The status word combines status bits that the CPU uses to control the binary logic operations and sets for digital processing. You can query the status bits and influence them individually.

The following table shows the arrangement of the status bits in the status word:

	Status word									
Bit number	15-9	8	7	6	5	4	3	2	1	0
Content	0	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC

The status bits /FC, RLO, STA, OR and BR are binary codes which the CPU uses for binary instructions. The status bits OS, OV, CC 0 and CC 1 are digital codes which generally express math function results.

/FC (first check)

The /FC bit signal state controls a logic operation string.

A logic operation string begins with an /FC bit signal state of "0" and a binary scan instruction (first scan). The first scan is the first binary bit logic operation or scan in a network. This sets the /FC bit to "1". The logic operation string ends with a binary value assignment (e.g. the "Set" instruction), a conditional jump or a change of block. This sets the /FC bit to "0".

RLO (result of logic operation)

The status bit RLO is the buffer in binary logic operations.

The CPU transfers the scan results of the first scan to the RLO. The scan results for each subsequent scan are combined with the RLO saved and the result obtained is then saved in the RLO.

You can set, reset or negate the RLO or save it in the binary result (BR) using the corresponding instructions.

The RLO is used to control the memory, timer and counter instructions and execute certain jump instructions.

STA (status)

The STA status bit is equal to the signal state of the binary operand queried.

The status of a logic operation that reads memory is always the same as the value of the addressed bit. The status of a logic operation that writes to memory (e.g. the "Set" or "Reset"

instructions) always has the same value as the bit written by the instruction. If the instruction does not write to memory, the status has the same value as the addressed bit. The status bit has no significance for logic operation instructions that do not access the memory. These instructions set the status bit to "1". The status bit is not queried by instructions. It is only evaluated when displaying the online status of program variables.

OR (OR bit)

The OR status bit is used when an AND logic operation is executed before an OR logic operation.

The OR bit is set if the RLO of the AND logic operation is "1". This anticipates the result of the OR logic operation. Every other binary instruction resets the OR bit.

OS (Overflow, stored)

The OS status bit saves the set OV status bit.

When the CPU sets the OV status bit, it also sets the OS status bit. However, while the next properly executed instruction resets the OV bit, the OS status bit remains set. You are therefore able to query a number range overflow or, for example, the use of invalid floating-point numbers at a later stage in the current CPU cycle.

OV (Overflow)

The OV status bit shows a number range overflow or the use of invalid floating-point numbers.

The OV status bit can be influenced by math functions, conversion instructions, and floating-point number comparisons.

CC 0 and CC 1 (condition code bits)

The CC 0 and CC 1 status bits provide information on the results of the following instructions:

- Comparison instructions
- Math functions
- Word logic operations
- Shift and rotate instructions

BR (binary result)

The BR status bit is used to execute the EN/ENO mechanism for boxes or as a condition in certain jump instructions (STL). You can influence the BR status bit using certain instructions (e.g. SAVE).

1.6.1.2 Setting status bits (S7-300, S7-400)

Setting the status bit for bit logic operations (S7-300, S7-400)

Description

Binary instructions affect the OR, STA, RLO and /FC status bits.

Setting the status bits in LAD

The following table shows how the status bits are set in the LAD programming language:

Instruction		Status bit			
Mnemonics	Title	OR	STA	RLO	/FC
-| |-	Normally open contact	x	x	x	1
-| / |-	Normally closed contact	x	x	x	1
-|NOT|-	Invert RLO	-	1	x	-
-( )-	Assignment	0	x	-	0
-(R)-	Reset output	0	x	-	0
-(S)-	Set output	0	x	-	0
SR	Set/reset flip-flop	x	x	x	1
RS	Reset/set flip-flop	x	x	x	1
-|P|-	Scan operand for positive signal edge	0	1	x	1
-|N|-	Scan operand for negative signal edge	0	1	x	1
P_TRIG	Scan RLO for positive signal edge	0	x	x	1
N_TRIG	Scan RLO for negative signal edge	0	x	x	1
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.					

Setting the status bits in FBD

The following table shows how the status bits are set in the FBD programming language:

Instruction		Status bit			
Mnemonics	Title	OR	STA	RLO	/FC
&	AND logic operation	x	x	x	1
<=1	OR logic operation	x	x	x	1
X	EXCLUSIVE OR logic operation	x	x	x	1
-|	Insert binary input	-	1	x	-
-o|	Invert RLO	-	1	x	-
-( )-	Assignment	0	x	-	0
R	Reset output	0	x	-	0

Instruction		Status bit			
Mnemonics	Title	OR	STA	RLO	/FC
S	Set output	0	x	-	0
SR	Set/reset flip-flop	x	x	x	1
RS	Reset/set flip-flop	x	x	x	1
P	Scan operand for positive signal edge	0	1	x	1
N	Scan operand for negative signal edge	0	1	x	1
P_TRIG	Scan RLO for positive signal edge	0	x	x	1
N_TRIG	Scan RLO for negative signal edge	0	x	x	1
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.					

Setting the status bits in STL

The following table shows how the status bits are set in the STL programming language:

Instruction		Status bit			
Mnemonics	Title	OR	STA	RLO	/FC
A	AND logic operation	0	x	x	1
AN	Negated AND logic operation	0	x	x	1
O	OR logic operation	0	x	x	1
ON	Negated OR logic operation	0	x	x	1
X	EXCLUSIVE OR logic operation	0	x	x	1
XN	Negated EXCLUSIVE OR logic operation	0	x	x	1
O	OR logic operation from AND functions	0	1	-	0
A(	AND logic operation with branch	0	1	-	0
AN(	Negated AND logic operation with branch	0	1	-	0
O(	OR logic operation with branch	0	1	-	0
ON(	Negated OR logic operation with branch	0	1	-	0
X(	EXCLUSIVE OR logic operation with branch	0	1	-	0
XN(	Negated EXCLUSIVE OR logic operation with branch	0	1	-	0
=	Assignment	0	x	-	0
R	Reset output	0	x	-	0
S	Set output	0	x	-	0
NOT	Invert RLO	-	1	x	-
SET	Set RLO to 1	0	1	1	0
CLR	Reset RLO to 0	0	0	0	0
SAVE	Save RLO in BR bit	-	-	-	-
FN	Scan RLO for negative signal edge	0	x	x	1

Instruction		Status bit			
Mnemonics	Title	OR	STA	RLO	/FC
FP	Scan RLO for positive signal edge	0	x	x	1
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.					

Setting status bits in timer and counter instructions (S7-300, S7-400)

Description

The timer and counter instructions affect the OR, STA, RLO and /ER status bits.

Setting the status bits in LAD and FBD

The following table shows how the status bits are set in the LAD and FDB programming languages:

Instruction		Status bit				
Mnemonics	Title	OS	OR	STA	RLO	/FC
S_PULSE	Assign pulse timer parameters and start	-	x	x	x	1
S_PEXT	Assign extended pulse timer parameters and start	-	x	x	x	1
S_ODT	Assign on-delay timer parameters and start	-	x	x	x	1
S_ODTS	Assign retentive on-delay timer parameters and start	-	x	x	x	1
S_OFFDT	Assign off-delay timer parameters and start	-	x	x	x	1
SP	Start pulse timer	-	0	-	-	0
SE	Start extended pulse timer	-	0	-	-	0
SD	Start on-delay timer	-	0	-	-	0
SS	Start retentive on-delay timer	-	0	-	-	0
SF	Start off-delay timer	-	0	-	-	0
S_CUD	Assign parameters and count up / down	-	x	x	x	1
S_CU	Assign parameters and count up	-	x	x	x	1
S_CD	Assign parameters and count down	-	x	x	x	1
SC	Set counter value	-	0	x	-	0
CU	Count up	-	0	x	-	0
CD	Count down	-	0	-	-	0
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.						

Setting the status bits in STL

The following table shows how the status bits are set in the STL programming language:

Instruction		Status bit				
Mnemonics	Title	OS	OR	STA	RLO	/FC
FR	Enable timer	-	0	-	-	0
R	Reset timer	-	0	-	-	0
SP	Start pulse timer	-	0	-	-	0
SE	Start extended pulse timer	-	0	-	-	0
SD	Start on-delay timer	-	0	-	-	0
SS	Start retentive on-delay timer	-	0	-	-	0
SF	Start off-delay timer	-	0	-	-	0
FR	Enable counters	-	0	-	-	0
R	Reset counter	-	0	-	-	0
S	Set counter	-	0	-	-	0
CU	Count up	-	0	-	-	0
CD	Count down	-	0	-	-	0
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.						

Setting the status bits in instructions with integers (S7-300, S7-400)

Description

Instructions with integers affect status bits CC 1, CC 0, OV and OS in the status word.

With a negative result, status bits CC 1 and CC 0 are set to "0". A positive result sets status bit CC 1 to "1" and status bit CC 0 to "0". Status bit CC 1 is set to "0" and status bit CC 0 is set to "1" if the result is negative.

In the case of a number range overflow, status bits OV and OS are set to "1". A divide by zero is indicated by status bits CC 1, CC 0, OV and OS having signal status "0".

The following tables show how the status bits are set when using integers:

Valid range	CC 1	CC 0	OV	OS
0 (zero)	0	0	0	*
16 bits: -32768 <=result < 0 (negative number) 32 bits: -2147483648 <=result < 0 (negative number)	0	1	0	*
16 bits: 32767 >= result > 0 (positive number) 32 bits: 2147483647 >= result > 0 (positive number)	1	0	0	*
* The OS bit is not influenced by the result of the operation.				

Invalid range	CC 1	CC 0	OV	OS
Underflow at addition 16 bits: Result = -65536 32 bits: Result = -4294967296	0	0	1	1
Underflow at multiplication 16 bits: Result < -32768 (negative number) 32 bits: Result < -2147483648 (negative number)	0	1	1	1
Overflow at addition, subtraction 16 bits: Result > 32767 (positive number) 32 bits: Result > 2147483647 (positive number)	0	1	1	1
Overflow at multiplication, division 16 bits: Result > 32767 (positive number) 32 bits: Result > 2147483647 (positive number)	1	0	1	1
Underflow at addition, subtraction 16 bits: Result < -32768 (negative number) 32 bits: Result < -2147483648 (negative number)	1	0	1	1
Division by 0	1	1	1	1

Setting the status bits in instructions with floating-point numbers (S7-300, S7-400)

Description

Instructions with floating point numbers affect status bits CC 1, CC 0, OV and OS in the status word.

With a negative result, status bits CC 1 and CC 0 are set to "0". A positive result sets status bit CC 1 to "1" and status bit CC 0 to "0". Status bit CC 1 is set to "0" and status bit CC 0 is set to "1" if the result is negative.

In the case of a number range overflow, status bits OV and OS are set to "1". An invalid floating point number is indicated by status bits CC 1, CC 0, OV and OS having signal status "0".

The following tables show how the status bits are set when using floating point numbers:

Valid range	CC 1	CC 0	OV	OS
+0, -0 (zero)	0	0	0	*
-3.402823E+38 < result < -1.175494E-38 (negative number)	0	1	0	*
+1.175494E-38 < result < 3.402824E+38 (positive number)	1	0	0	*
* The OS bit is not influenced by the result of the operation.				

Invalid range	CC 1	CC 0	OV	OS
Underflow -1.175494E-38 < result < -1.401298E-45 (negative number)	0	0	1	1
Underflow +1.401298E-45 < result < +1.175494E-38 (positive number)	0	0	1	1
Overflow Result < -3.402823E+38 (negative number)	0	1	1	1
Overflow Result > 3.402823E+38 (positive number)	1	0	1	1
Not a valid floating-point number or invalid operation (input value outside of the valid value range)	1	1	1	1

Setting the status bit for comparison instructions (S7-300, S7-400)

Description

Comparison statements affect status bits CC 1, CC 0, OV, OS in the status word.

Comparison statements set status bits CC 1 and CC 0 regardless of the comparison relationship.

When invalid floating point numbers are used, status bits CC 1, CC 0, OV and OS are set to "1".

The following table shows how the status bits are set with comparison statements:

Result	CC 1	CC 0	OV	OS
Equal to (ACCU2 = ACCU1)	0	0	0	*
Greater than (ACCU2 > ACCU1)	1	0	0	*
Smaller than (ACCU2 < ACCU1)	0	1	0	*
Invalid floating-point number	1	1	1	1
* The OS bit is not influenced by the result of the operation.				

Setting status bits in program control operation instructions (S7-300, S7-400)

Description

The instructions for program control affect the OS, OR, STA, RLO and /FC status bits.

Setting the status bits in LAD and FBD

The following table shows how the status bits are set in the LAD and FDB programming languages:

Instruction		Status bit				
Mnemonics	Title	OS	OR	STA	RLO	/FC
JMP	Jump if RLO = 1	-	0	1	1	0
JMPN	Jump if RLO = 0	-	0	1	1	0
RET	Return	0	0	1	1	0
CALL	Call block	0	0	1	-	0
OPN	Open global data block	-	-	-	-	-
OPNI	Open instance data block	-	-	-	-	-
MCR<	Open MCR ranges	-	0	1	-	0
MCR>	Close MCR ranges	-	0	1	-	0
MCRA	Enable MCR range	-	-	-	-	-
MCRD	Disable MCR range	-	-	-	-	-
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.						

Setting the status bits in STL

The following table shows how the status bits are set in the STL programming language:

Instruction		Status bit				
Mnemonics	Title	OS	OR	STA	RLO	/FC
JU	Unconditional jump	-	-	-	-	-
JC	Jump if RLO = 1	-	0	1	1	0
JCN	Jump if RLO = 0	-	0	1	1	0
JCB	Jump if RLO = 1 and save RLO	-	0	1	1	0
JNB	Jump if RLO = 0 and save RLO	-	0	1	1	0
JBI	Jump if BR = 1	-	0	1	-	0
JNBI	Jump if BR = 1	-	0	1	-	0
JO	Jump if OV = 1	-	-	-	-	-
JOS	Jump if OS = 1	0	-	-	-	-
JZ	Jump if the result is zero	-	-	-	-	-
JN	Jump if the result is not zero	-	-	-	-	-
JP	Jump if the result is greater than zero	-	-	-	-	-
JM	Jump if the result is less than zero	-	-	-	-	-
JPZ	Jump if the result is greater than or equal to zero	-	-	-	-	-
JMZ	Jump if the result is less than or equal to zero	-	-	-	-	-
JUO	Jump if the result is invalid	-	-	-	-	-
JL	Defining the jump list	-	-	-	-	-

Instruction		Status bit				
Mnemonics	Title	OS	OR	STA	RLO	/FC
LOOP	Loop	-	-	-	-	-
OPN	Open global data block	-	-	-	-	-
OPN_DI	Open instance data block	-	-	-	-	-
CALL	Call block	0	0	1	-	0
CC	Conditional block call	0	0	1	1	0
UC	Unconditional block call	0	0	1	-	0
BE	Block end	0	0	1	-	0
BEC	Conditional block end	x	0	1	1	0
BEU	Conditional block end	0	0	1	-	0
MCR(	Open MCR ranges	-	0	1	-	0
)MCR	Close MCR ranges	-	0	1	-	0
MCRA	Enable MCR range	-	-	-	-	-
MCRD	Disable MCR range	-	-	-	-	-
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.						

Setting the status bit for word logic operations (S7-300, S7-400)

Description

Word logic instructions affect status bits CC 1, CC 0, and OV in the status word.

If all the bits in the result of a word logic instruction are "0", status bit CC 1 is reset to "0". If at least one bit in the result is "1", status bit CC 1 is set to "1".

The following table shows how the status bits are set with word logic instructions:

Result	CC 1	CC 0	OV
Zero	0	0	0
Not zero	1	0	0

Setting the status bits for shift and rotate instructions (S7-300, S7-400)

Description

Shift and rotate instructions affect status bits CC 1, CC 0, and OV in the status word.

In the case of shift and rotate instructions, the signal status of the last shifted or rotated bit is transferred to status bit CC 1. Status bits CC 0 and OV are reset.

The following table shows the setting of the status bit for shift and rotate instructions:

Displaced/rotated bit	CC 1	CC 0	OV
"0"	0	0	0
"1"	1	0	0
For shift or rotate number 0	*	*	*
* The bit is not influenced by the result of the operation.			

1.6.1.3 Querying status bits in STL (S7-300, S7-400)

Querying status bits with bit logic operation instructions (S7-300, S7-400)

Description

You can use the following bit logic instructions to query the signal state of bits in the status word:

- A: AND logic operation
- AN: Negated AND logic operation
- O: OR logic operation
- ON: Negated OR logic operation
- X: EXCLUSIVE OR logic operation
- XN: Negated EXCLUSIVE OR logic operation

The following table shows the operands which can be used to scan the signal state of the bits in the status word:

Operand	Querying the status word
== 0	Scan for ((CC 0 = 0) AND (CC 1 = 0))
<> 0	Scan for CC 0 <>CC 1
> 0	Scan for ((CC 0 = 0) AND (CC 1 = 1))
< 0	Scan for ((CC 0 = 1) AND (CC 1 = 0))
>= 0	Scan for CC 0 = 0
<= 0	Scan for CC 1 = 0
UO	Scan for ((CC 1 = 1) AND (CC 0 = 1))
OV	Scan for OV = 1
OS	Scan for OS = 1
BR	Scan for BR = 1

See also

A: AND logic operation" (Page 4708)

AN: Negated AND logic operation (Page 4709)

O: OR logic operation (Page 4710)

ON: Negated OR logic operation (Page 4712)

X: EXCLUSIVE OR logic operation (Page 4713)

XN: Negated EXCLUSIVE OR logic operation (Page 4714)

O: OR logic operation of AND functions (Page 4715)

A(: AND logic operation with branch (Page 4716)

AN(: Negated AND logic operation with branch (Page 4717)

O(: OR logic operation with branch (Page 4718)

ON(: Negated OR logic operation with branch (Page 4719)

X(: EXCLUSIVE OR logic operation with branch (Page 4720)

XN(: Negated EXCLUSIVE OR logic operation with branch (Page 4721)

): Close branch (Page 4722)

=: Assignment (Page 4723)

R: Reset (Page 4724)

S: Set (Page 4725)

NOT: Invert RLO (Page 4726)

SET: Set RLO to 1 (Page 4727)

CLR: Reset RLO to 0 (Page 4728)

SAVE: Save RLO in BR bit (Page 4729)

FN: Scan RLO for negative signal edge (Page 4730)

FP: Scan RLO for positive signal edge (Page 4731)

Querying status bits with jump instructions (S7-300, S7-400)

Description

The following table shows the jump instruction evaluation of the status bits:

RLO	BR	CC 1	CC 0	OV	OS	Executed instruction mnemonics
1	-	-	-	-	-	JC, JCB
0	-	-	-	-	-	JCN, JNB
-	1	-	-	-	-	JBI
-	0	-	-	-	-	JNBI
-	-	0	0	-	-	JZ, JMZ, JPZ
-	-	1	0	-	-	JN, JP, JPZ
-	-	0	1	-	-	JN, JM, JMZ
-	-	1	1	-	-	JUO
-	-	-	-	1	-	JO
-	-	-	-	-	1	JOS

The following table shows the dependency between status bits CC1 and CC0 and the conditional jump instructions:

CC 1	CC 0	Result	Executed instruction mnemonics
0	0	= 0	JZ
1 or 0	0 or 1	<> 0	JN
1	0	> 0	JP
0	1	< 0	JM
0 or 1	0	>= 0	JPZ
0	0 or 1	<= 0	JMZ
1	1	UO (invalid)	JUO

See also

JU: Unconditional jump (Page 4822)

JC: Jump if RLO = 1 (Page 4823)

JCN: Jump if RLO = 0 (Page 4824)

JCB: Jump if RLO = 1 and save RLO (Page 4825)

JNB:: Jump if RLO = 0 and save RLO (Page 4827)

JBI: Jump if BR = 1 (Page 4828)

JNBI: Jump if BR = 0 (Page 4829)

JO: Jump if OV = 1 (Page 4830)

JOS: Jump if OS = 1 (Page 4831)

JZ: Jump if the result is zero (Page 4833)

JN: Jump if the result is not zero (Page 4834)

JP: Jump if the result is greater than zero (Page 4836)

JM: Jump if the result is less than zero (Page 4837)

JPZ: Jump if the result is greater than or equal to zero (Page 4839)

JMZ: Jump if the result is less than or equal to zero (Page 4841)

JUO: Jump if the result is invalid (Page 4842)

JL: Define jump list (Page 4844)

LOOP: Loop (Page 4845)

1.6.2 Status word for S7-1500 (S7-1500)

1.6.2.1 Basic information on the status word (S7-1500)

Description

The status word combines status bits that the CPU uses to control the binary logic operations and sets for digital processing. You can query the status bits and influence them individually.

The following table shows the arrangement of the status bits in the status word:

	Status word									
Bit number	15-9	8	7	6	5	4	3	2	1	0
Content	0	BR	CC 1	CC 0	OV	OS	0	0	0	0

The status bits OS, OV, CC 0 and CC 1 are digital codes which generally express math function results.

OS (Overflow, stored)

The OS status bit saves the set OV status bit.

When the CPU sets the OV status bit, it also sets the OS status bit. However, while the next properly executed instruction resets the OV bit, the OS status bit remains set. You are therefore able to query a number range overflow or, for example, the use of invalid floating-point numbers at a later stage in the current CPU block.

OV (Overflow)

The OV status bit shows a number range overflow or the use of invalid floating-point numbers.

The OV status bit can be influenced by math functions, conversion instructions, and floating-point number comparisons.

CC 0 and CC 1 (condition code bits)

The CC 0 and CC 1 status bits provide information on the results of the following instructions:

- Comparison instructions
- Math functions
- Word logic operations
- Shift and rotate instructions

BR (binary result)

The BR status bit is used to execute the EN/ENO mechanism for boxes or as a condition in certain jump instructions (STL). You can influence the BR status bit using certain instructions (e.g. SAVE).

RLO (result of logic operation)

Status bit RLO is the buffer for binary logic operations and is not a component of the status word.

The CPU transfers the scan results of the first scan to the RLO. The scan results for each subsequent scan are combined with the RLO saved and the result obtained is then saved in the RLO.

You can set or reset the RLO using the corresponding instructions.

The RLO is used to control the memory, timer and counter instructions and execute certain jump instructions.

Note

Since the status bits are only partly needed for internal processing, they are only displayed if they are available for further program execution.

1.6.2.2 Querying and setting status bits in STL (S7-1500)

Querying status bits with bit logic operation instructions (S7-1500)

Description

You can use the following bit logic instructions to query the signal state of bits in the status word:

- A: AND logic operation
- AN: Negated AND logic operation
- O: OR logic operation
- ON: Negated OR logic operation
- X: EXCLUSIVE OR logic operation
- XN: Negated EXCLUSIVE OR logic operation

The following table shows the operands which can be used to scan the signal state of the bits in the status word:

Operand	Querying the status word
== 0	Scan for ((CC 0 = 0) AND (CC 1 = 0))
<> 0	Scan for CC 0 <>CC 1
> 0	Scan for ((CC 0 = 0) AND (CC 1 = 1))
< 0	Scan for ((CC 0 = 1) AND (CC 1 = 0))
>= 0	Scan for CC 0 = 0
<= 0	Scan for CC 1 = 0
UO	Scan for ((CC 1 = 1) AND (CC 0 = 1))

Operand	Querying the status word
OV	Scan for OV = 1
OS	Scan for OS = 1
BR	Scan for BR = 1

See also

A: AND logic operation (Page 1482)
AN: Negated AND logic operation (Page 1483)
O: OR logic operation (Page 1484)
ON: Negated OR logic operation (Page 1485)
X: EXCLUSIVE OR logic operation (Page 1487)
XN: Negated EXCLUSIVE OR logic operation (Page 1488)
O: OR logic operation of AND functions (Page 1489)
A(: AND logic operation with branch (Page 1490)
AN(: Negated AND logic operation with branch (Page 1491)
O(: OR logic operation with branch (Page 1492)
ON(: Negated OR logic operation with branch (Page 1493)
X(: EXCLUSIVE OR logic operation with branch (Page 1494)
XN(: Negated EXCLUSIVE OR logic operation with branch (Page 1495)
): Close branch (Page 1495)
=: Assignment (Page 1496)
R: Reset (Page 1497)
S: Set (Page 1498)
NOT: Invert RLO (Page 1499)
SET: Set RLO to 1 (Page 1500)
CLR: Reset RLO to 0 (Page 1500)
SAVE: Save RLO in BR bit (Page 1501)
FN: Scan RLO for negative signal edge (Page 1502)
FP: Scan RLO for positive signal edge (Page 1503)

Querying status bits with jump instructions (S7-1500)

Description

The following table shows the jump instruction evaluation of the status bits:

RLO	BR	CC 1	CC 0	OV	OS	Executed instruction mnemonics
1	-	-	-	-	-	JC, JCB
0	-	-	-	-	-	JCN, JNB
-	1	-	-	-	-	JBI
-	0	-	-	-	-	JNBI
-	-	0	0	-	-	JZ, JMZ, JPZ
-	-	1	0	-	-	JN, JP, JPZ
-	-	0	1	-	-	JN, JM, JMZ
-	-	1	1	-	-	JUO
-	-	-	-	1	-	JO
-	-	-	-	-	1	JOS

The following table shows the dependency between status bits CC 0 and CC 1 and the conditional jump instructions:

CC 1	CC 0	Result	Executed instruction mnemonics
0	0	= 0	JZ
1 or 0	0 or 1	<> 0	JN
1	0	> 0	JP
0	1	< 0	JM
0 or 1	0	>= 0	JPZ
0	0 or 1	<= 0	JMZ
1	1	UO (invalid)	JUO

See also

JU: Unconditional jump (Page 4822)

JC: Jump if RLO = 1 (Page 4823)

JCN: Jump if RLO = 0 (Page 4824)

JCB: Jump if RLO = 1 and save RLO (Page 4825)

JNB:: Jump if RLO = 0 and save RLO (Page 4827)

JBI: Jump if BR = 1 (Page 4828)

JNBI: Jump if BR = 0 (Page 4829)

JO: Jump if OV = 1 (Page 4830)

JOS: Jump if OS = 1 (Page 4831)

JZ: Jump if the result is zero (Page 4833)

JN: Jump if the result is not zero (Page 4834)

JP: Jump if the result is greater than zero (Page 4836)

JM: Jump if the result is less than zero (Page 4837)

JPZ: Jump if the result is greater than or equal to zero (Page 4839)

JMZ: Jump if the result is less than or equal to zero (Page 4841)

JUO: Jump if the result is invalid (Page 4842)

JL: Define jump list (Page 4844)

LOOP: Loop (Page 4845)

Jump label (Page 1588)

JU: Unconditional jump (Page 1589)

JC: Jump if RLO = 1 (Page 1590)

JCN: Jump if RLO = 0 (Page 1591)

JCB: Jump if RLO = 1 and save RLO (Page 1592)

JNB: Jump if RLO = 0 and save RLO (Page 1594)

JBI: Jump if BR = 1 (Page 1595)

JNBI: Jump if BR = 0 (Page 1596)

JO: Jump if OV = 1 (Page 1597)

JOS: Jump if OS = 1 (Page 1598)

JZ: Jump if the result is zero (Page 1600)

JN: Jump if the result is not zero (Page 1601)

JP: Jump if the result is greater than zero (Page 1603)

JM: Jump if the result is less than zero (Page 1604)

JPZ: Jump if the result is greater than or equal to zero (Page 1606)

JMZ: Jump if the result is less than or equal to zero (Page 1608)

JUO: Jump if the result is invalid (Page 1609)

JL: Define jump list (Page 1611)

LOOP: Loop (Page 1612)

Setting the status bit for word logic operations (S7-1500)

Description

Word logic instructions affect status bits CC 0 , CC 1, and OV in the status word. The status bits are only set if they are used in the program sequence.

If all the bits in the result of a word logic instruction are "0", status bit CC 1 is reset to "0". If at least one bit in the result is "1", status bit CC 1 is set to "1".

The following table shows how the status bits are set with word logic instructions:

Result	CC 1	CC 0	OV
Zero	0	0	0
Not zero	1	0	0

See also

AW: AND logic operation word by word (Page 1626)

OW: OR logic operation word by word (Page 1628)

XOW: EXCLUSIVE OR logic operation word by word (Page 1630)

AD: AND logic operation double word by double word (Page 1632)

OD: OR logic operation double word by double word (Page 1633)

XOD: EXCLUSIVE OR logic operation double word by double word (Page 1635)

1.6.3 Master Control Relay (S7-300, S7-400)

1.6.3.1 Master control relay (S7-300, S7-400)

Description of the MCR functionality

With conventional programmable controllers, the Master Control Relay (MCR) enables or disables the signal flow in one part of the controller.

Note

Refer to the information in the "Important notes on using MCR functions (Page 192)" section

WARNING

To prevent personal injury or property damage, never use the MCR to replace a hard-wired mechanical master control relay that is being used as an emergency stop or safety switching device.

The MCR affects all instructions that write a value back to the memory. When MCR dependency is enabled these instructions respond regardless of any previous binary or digital logic operation.

The following instructions are available for the implementation of the MCR:

- MCRA: Enable MCR area
- MCR: Open MCR ranges

- MCR: Close MCR ranges
- MCRD: Disable MCR area

The MCR dependency for this area is enabled when the result of logic operation is "1" directly before the opening of an MCR area. If the RLO of the area is "0" when it is opened, the processing in this area is without MCR dependency.

1.6.3.2 Important notes on using MCR functions (S7-300, S7-400)

CAUTION

Note the following for blocks in which the master control relay was enabled with the instruction "Enable MCR range" (MCRA):

- When the MCR is switched off, the value "0" is written by all assignments in program sections between the instructions "Open MCR range" (MCR( ) and "Close MCR range".
- The MCR is then switched off exactly at the point when the result of logic operation (RLO) was "0" before an "Open MCR range" (MCR( ) instruction.

DANGER

CPU in STOP or undefined runtime characteristics!

For address calculations and also writes to local data, the compiler accesses the temporary variables defined after VAR_TEMP. The following command sequences therefore change the AS to STOP or lead to undefined runtime characteristics (refer to following explanations).

Formal parameter access

- Access to components of complex FC parameters of the type STRUCT, PLC data type (UDT), ARRAY, STRING
- Access to components of complex FB parameters of the type STRUCT, PLC data type (UDT), ARRAY, STRING from the InOut area in a block with multiple instance capability.
- Access to parameters of a function block with multiple instance capability if its address is greater than 8180.0.
- Access in a function block with multiple instance capability to a parameter of the type BLOCK_DB opens DB0. Any subsequent data access changes the CPU to STOP. T 0, C 0, FC0, or FB0 are also always used for TIMER, COUNTER, BLOCK_FC, and BLOCK_FB.

Parameter transfer

In block calls in which parameters are transferred.

Remedy

Free the above commands from their dependence on the MCR:

1. Use the instruction "Disable MCR range" (MCRD) to disable the master control relay before the assignment or network in question.
2. Use the instruction "Enable MCR range" (MCRA) to enable the master control relay before the assignment or network in question.

Programming recommendations (S7-1200, S7-1500) 2

2.1 The new S7-1200/1500 CPU functions and programming recommendations at a glance (S7-1200, S7-1500)

Higher performance

The S7-1200/1500 CPUs offer much higher performance than the S7-300/400 CPUs. When programming with STEP 7 V5.x, you were probably used to methods such as absolute addressing to achieve higher performance from the CPU and leaner program code. Due to the high performance provided by CPUs of the S7-1200/1500 series, this is now no longer necessary. If you wish to reuse the programming code created on an S7-300/400CPU on an S7-1200/1500 CPU, it will probably be necessary to modify the program code here and there in order to fully exploit the high performance.

In the paragraphs below, we would like to introduce some new programming options of the S7-1200/1500 CPUs.

You can find more tips on improving the performance here: Increasing performance in STEP 7 (TIA Portal) (https://support.industry.siemens.com/cs/ww/en/view/37571372)

Universal symbolism

The S7-1500 allows you to use the symbolism throughout the entire project. Using the auto-complete function, you are given context-dependent support for programming with symbols within the programming editors. The data elements, for example those in a data block, are assigned only a symbolic name in the declaration but no fixed address within the data block. This allows you to fully exploit the high performance of the S7-1500 when accessing these data elements. The absolute addresses of operands need not be known and access errors are avoided.

Your program code will be clearer due to the symbolism and you have to comment less. All points of use are automatically updated when a correction is made to the symbolism.

Note

You can find additional information on the use of universal symbols at:

How can I address symbolically?

Symbolic addressing (Page 198)

Optimized block access

With optimized block access, the declared data elements are arranged automatically in the available memory area of the block in a way that makes optimal use of its capacity. The data is structured and saved in way that is optimal for the CPU used. The storage is left to the system. The data elements are assigned only a symbolic name in the declaration by which the tag within the block can be addressed. This allows you to increase the performance of the CPU. Access errors, from the HMI, for example, are not possible.

Comparison of block accesses
Standard < > Optimized

SIEMENS

	Standard block access (S7-1200/1500 compatible with S7-300/400)	Optimized block access (S7-1200/1500 only)
Data management	You can address tags using both symbolic (memory-optimized) and absolute (user-defined) addresses.	The system manages and optimizes the data storage. This results in optimal use of the memory capacity.
Performance	The access to a CPU of the S7-1200/1500 series does not always occur as fast as possible because the data storage can be inefficient due to absolute addressing.	The access always occurs as fast as possible because the data storage is optimized by the system and no fixed addresses are assigned.
Susceptibility to errors	Absolute addressing (e.g., from HMI or when indirect addressing is used) can lead to inconsistencies after a change to the fixed address.	Access errors, e.g., from HMI or when indirect addressing is used, are not possible because the access is symbolic.
Retentivity	Valid for all tags of a data block	Valid for individual tags

Recommendation:
To achieve the best-possible performance, the combining of different block access types within your program is not recommended.

Note

You can find additional information on blocks with optimized access here:

Blocks with optimized access (Page 55)

New data types

The new data types LWORD, LINT, ULINT, LTIME, LTOD, LDT and ARRAY (32-bit limit) offer much higher calculation accuracy when using mathematical functions. In the area of implicit and explicit data type conversion, you have more options in comparison to the CPUs of the S7-300/400 series.

Note

You can find additional information on the new data types here:

Overview of the valid data types (Page 231)

PLC data types

PLC data types (UDT) are data structures that you define and that can be used multiple times within the program. The structure of a PLC data type is made up of several components, each of which can contain different data types. You define the type of components in the declaration of the PLC data type.

You can use the PLC data type as a base data type for defining tags and as a template for creating global data blocks. If you later make changes to the PLC data type, the changes are automatically updated at all points of use.

You can also symbolically access individual elements of an ARRAY within a PLC data type.

Note

You can find additional information on the PLC data types here:

What does a PLC data type look like and how can I declare it in my program?

- PLC data types (UDT) (Page 273)

You can find programming recommendations for the PLC data type here:

- Using PLC data types (UDT) (Page 220)
- Using the DB_ANY data type (Page 204)

Uniform instructions in all programming languages

You are provided with a uniform set of instructions in all programming languages (LAD, FBD, STL, SCL, and GRAPH).

Slice access

Slice access gives you the option of specifically addressing areas within declared tags. You can implement symbolic access to a single bit up to the level of the tag. The single bit is then addressed absolutely.

Note

You can find additional information on slice access here:

How can I address with slice access?

- Addressing areas of a tag with slice access (Page 110)
- Basics of block access (Page 55)

Indirect addressing

Indirect addressing offers you the option of addressing operands whose address is not calculated until runtime. All programming languages provide general methods for indirect addressing, for example, via POINTER. In the STL and SCL programming languages, you can also use PEEK and POKE instructions.

Note

You can find additional information on indirect addressing here:

Indirect addressing (Page 114)

"Sample Library for Instructions"

You can use the "Sample Library for Instructions" to assist in programming. You will find practical and easy-to-understand programming examples for SIMATIC S7-1500 and S7-1200 in the LAD language here. You can simply insert the examples into your program and reuse them there.

Note

You can find additional information on the program examples here:

Sample Library for Instructions (Page 593)

Additional programming recommendations

You can find additional information about programming recommendations and a programming guide in the Siemens Industry online support under:

- FAQs: Programming recommendations (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&Datakey=47071380&extranet=standard&groupid=4000002&viewreg=WW&nodeid0=29156492&objaction=csopen)
- Background and system descriptions in the Programming Guideline and Programming Styleguide (https://support.industry.siemens.com/cs/document/81318674/programming-guideline-and-programming-styleguide-for-s7-1200-and-s7-1500?dti=0&lc=en-WW)

2.2 Symbolic addressing (S7-1200, S7-1500)

Advantages of symbolic addressing

The use of universally applied and meaningful symbols in the overall project makes the program code easier to read and understand.

This gives you the following advantages:

- You do not have to write detailed comments.
- Data access is faster.
- No errors occur when accessing data.
- You no longer have to work with absolute addresses.
- The assignment of the symbol to the memory address is monitored by STEP 7, which means all points of use are automatically updated when the name or the address of a tag changes.

Programming in STEP 7 V5.x

STEP 7 V5.x already gave you the option to write a program that is easier to read by using descriptive names for operands and blocks. You did this by assigning the symbolic operands to memory addresses and blocks in the symbol table. In order for a change in the symbolism to also have an affect on the program code in the program editor, you had to use the "Operand priority" property to specify whether the symbol or the absolute value had priority.

The use of symbolic addressing allowed you to create a program with greater clarity. However, in some cases, such as when programming with user-defined data types (UDT), this could impair performance.

You could increase the performance by ignoring the symbolism in the UDT and using absolute addressing. This made it necessary, however, to know the data storage. Changes to the UDT were not automatically updated. Using absolute addressing, you could also access parts of a tag and edit these. The disadvantage of exclusively absolute addressing, however, was that the program code became cluttered after a certain level and you had to insert additional comments for better orientation.

Procedure in STEP 7 TIA Portal

The S7-1500 CPU offers much higher performance than the S7-300/400 CPUs. To make full use of this high performance, we recommend that you enable optimized block access for all blocks and use symbolic addressing in the program code.

The program editor helps you work with symbols through context-sensitive input help, for example, auto-completion. Using it, you can easily access existing tags or instructions during programming.

Programming example

The following example shows you how to symbolically access individual elements:

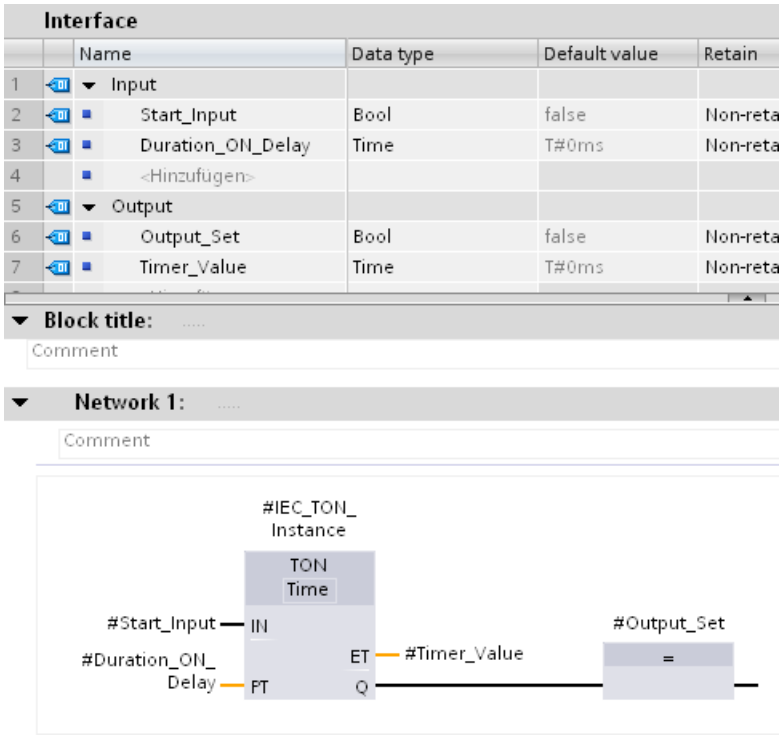

You can use the tag names that you have defined in the block interface directly at the parameters of the TON instruction without knowing the absolute address of the tags.

See also

Setting up block access (Page 57)

Basics of block access (Page 55)

Addressing areas of a tag with slice access (Page 110)

Addressing global variables (Page 103)

Addressing variables in global data blocks (Page 106)

Using autocompletion (Page 6859)

2.3 Indirect addressing of ARRAY components (S7-1200, S7-1500)

Implementing ARRAY access in TIA Portal with a variable index

It is advisable to use an ARRAY when you want to process assembled data of the same data type. For addressing ARRAY elements, you can specify either constants or tags of the integer data type as index. Integers with lengths of up to 32 bits are allowed here.

With indirect addressing using a tag, the index is only calculated during program runtime. You can, for example, use a different index for each cycle in program loops. You can also access an ARRAY within a PLC data type (UDT).

This gives you the following advantages:

- No addressing is necessary with address registers or with self-assembled pointers, for example, an ANY pointer.
- More flexibility within your program.
- The variable index is available in all STEP 7 programming languages.
- It uses the existing names of data blocks and ARRAY tags (symbolic addressing). This improves readability of the program code.
- The start address of the ARRAY does not have to be known.
- The program code is easier to create and the compiler generates optimized program code.

Procedure in STEP 7 V5.x

In STEP 7 V5.x, you had to use an address register with the help of a self-configured POINTER for indirect addressing of ARRAY elements. The following aspects had to be taken into consideration for this:

- The name of the ARRAY was not used. This reduced the readability of the program code and make additional comments necessary.
- The start address of the ARRAY had to be known to perform the addressing.

The SCL programming language already supported indirect addressing with variable index.

Programming example in STEP 7 V5.x

The "Data_classic" data block is required for the following STL example. To address an element of the "Quantities" ARRAY, the following commands must be used:

STL	Explanation
OPN "Data_classic"	// The "Data_classic" data block is called.
L #index	// The value of the local tag #index is loaded into accumulator 1.
SLD 3	// Move bits 0 to 31 of accumulator 1 by 3 positions to the left.
	// Fill the now empty bit places with zeros.
LAR1	// Load address register 1 with contents of accumulator 1.

STL	Explanation
L DBW [AR1, P#10.0]	// Load the ARRAY element addressed with #index into accumulator 1. // P#10.0 = Start address of the array

Programming example in STEP 7 TIA Portal

In the example below, you see the indirect addressing of an ARRAY element in STL in the TIA Portal.

Create a global data block for this purpose:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_Quantities".
4. Select the type of the data block "ARRAY DB".
5. Select the "DINT" data type as ARRAY data type.
6. Specify "10" as high ARRAY limit.
7. Click "OK".

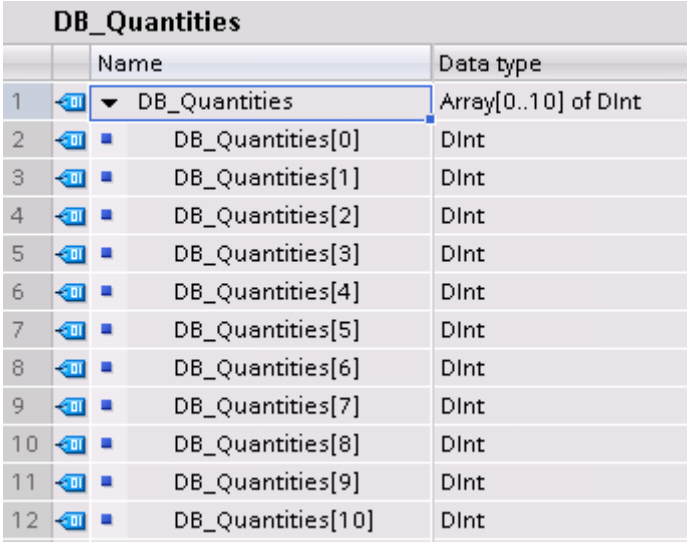

DB_Quantities

	Name	Data type
1	DB_Quantities	Array[0..10] of DInt
2	DB_Quantities[0]	DInt
3	DB_Quantities[1]	DInt
4	DB_Quantities[2]	DInt
5	DB_Quantities[3]	DInt
6	DB_Quantities[4]	DInt
7	DB_Quantities[5]	DInt
8	DB_Quantities[6]	DInt
9	DB_Quantities[7]	DInt
10	DB_Quantities[8]	DInt
11	DB_Quantities[9]	DInt
12	DB_Quantities[10]	DInt

1. Create a function block and name it "FB_Quantities".
2. Declare the block interface as follows:

FB_Quantities		
	Name	Data type
1	Input	
2	Output	
3	InOut	
4	Static	
5	Temp	
6	Index	Int
7	Constant	

3. Write the following program code:

Network 1:

Comment

```
1
2        L     "DB_Quantities".THIS[#index]
3
```

You need just one more program line for addressing an ARRAY element in the TIA Portal. The value of the ARRAY element #index is loaded directly from the data block into accumulator 1.

4. Call the "FB_Quantities" function block in OB1 and assign it an index between 0 - 10:

Network 1:

Comment

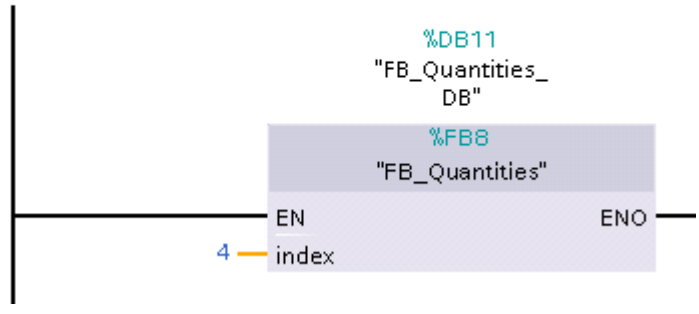

Note the following to get the best performance:

- Declare tags that are used as an ARRAY index as an integer of less than or equal to 32 bits.
- Store intermediate results and ARRAY indexes in the temporary local data area.

See also

Indirect addressing of ARRAY components (Page 132)
Basic information on ARRAY (Page 283)
ARRAY (Page 283)
Indirect addressing in STL (Page 133)
Indirect addressing in SCL (Page 130)

2.4 Using the DB_ANY data type (S7-1200, S7-1500)

Using the DB_ANY data type (S7-1200/1500)

The DB_ANY data type is used to identify any data block. For CPUs of the S7-1200/1500 series, you have the option of accessing a data block that is not yet available during programming. For this purpose, create a block parameter of the DB_ANY data type in the block interface of the accessing block. The data block name or a tag of data type DB_ANY which was previously assigned to the data block name is transferred to this parameter during runtime. You can symbolically process the content of a data block by means of the following instructions:

- VARIANT_TO_DB_ANY: Convert VARIANT to DB_ANY
- DB_ANY_TO_VARIANT: Convert DB_ANY to VARIANT

For more information about the instructions, refer to "Basic instructions > STL/SCL > Conversion operations > VARIANT".

The advantage of this procedure: You can create the program code before you know which data block will be processed.

The following objects are required for this example:

Call structure of PLC S7-1500

	Call structure	!	Address	Details
1	Main		OB1	
2	FB_PrepareNextOrder, DB_FB_PrepareNextOrder		FB1, DB4	@Main ▶ NW1
3	DB_OrderList		DB2	@FB_PrepareNextOrder ▶ Program code
4	FC_PrepareOrder		FC3	@FB_PrepareNextOrder ▶ Program code
5	FC_PrepareHole		FC1	@FC_PrepareOrder ▶ Program code
6	DB_PunchList		DB3	@FC_PrepareHole ▶ Program code
7	DB_PunchList		DB3	@FC_PrepareHole ▶ Program code
8	DB_PunchList		DB3	@FC_PrepareHole ▶ Program code
9	DB_PunchList		DB3	@FC_PrepareHole ▶ Program code
10	FC_PrepareWindowFlexible		FC5	@FC_PrepareOrder ▶ Program code
11	DB_PunchList		DB3	@FC_PrepareWindowFlexible ▶ Program code
12	DB_PunchList		DB3	@FC_PrepareWindowFlexible ▶ Program code
13	DB_PunchList		DB3	@FC_PrepareWindowFlexible ▶ Program code
14	DB_PunchList		DB3	@FC_PrepareWindowFlexible ▶ Program code
15	FC_PrepareWindowStatic		FC2	@FC_PrepareOrder ▶ Program code
16	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
17	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
18	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
19	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
20	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
21	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
22	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
23	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
24	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
25	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
26	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
27	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
28	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
29	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
30	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
31	DB_PunchList		DB3	@FC_PrepareWindowStatic ▶ Program code
32	UDT_Hole		UDT1	@FC_PrepareOrder ▶ Program code
33	UDT_RectangleWindowFlexible		UDT3	@FC_PrepareOrder ▶ Program code
34	UDT_RectangleWindowStatic		UDT2	@FC_PrepareOrder ▶ Program code
35	FC_SwapOrderList		FC4	@Main ▶ NW2
36	DB_OrderList		DB2	@FC_SwapOrderList ▶ Program code
37	FB_PrepareNextOrder, #OrderList		FB1	Block interface
38	DB_OrderList		DB2	@FB_PrepareNextOrder ▶ Program code
39	FC_PrepareOrder		FC3	@FB_PrepareNextOrder ▶ Program code
40	MOVE_BLK_VARIANT		FC900	@FC_SwapOrderList ▶ Program code
41	OrderRepository		DB6	@Main ▶ NW2
42	OrderRepository		DB6	@Main ▶ NW2
43	DB_OrderHole (Data block derived from UDT_Hole)		DB1	

Programming example

The following example shows you how to use the DB_ANY data type:

A punching machine can punch out a variety of geometric shapes. The punching jobs are transferred to the machine and there are specific job data for each individual job. The job data differ in their job type as well as in their values.

Procedure - Creating the PLC data types

In the first job, a round hole is to be punched out of a piece of sheet metal. For the punching machine to be able to execute this job, it needs the center point coordinates and the radius of the hole. You can transfer these job data to the punching machine collectively in a PLC data type (UDT).

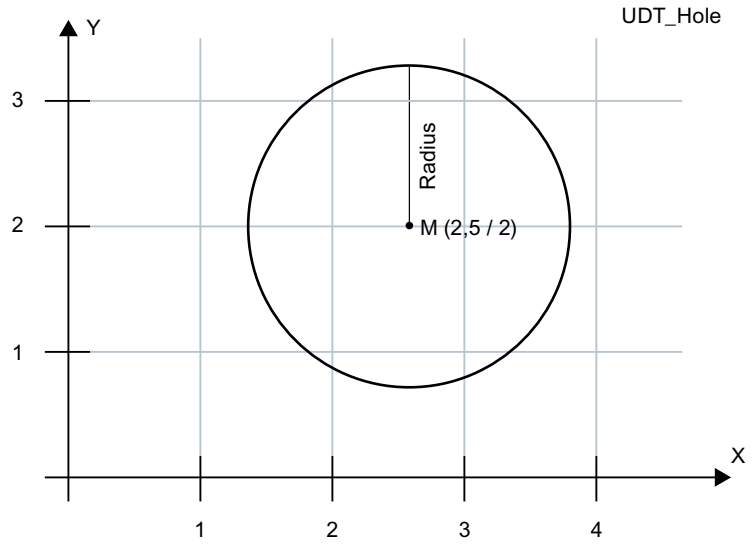

Create the PLC data type "UDT_Hole" to transfer the job data:

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare the following lines within the PLC data type:
   X-coordinate > REAL
   Y-coordinate > REAL
   Diameter > REAL

UDT_Hole

	Name	Data type
1	X-coordinate	Real
2	Y-coordinate	Real
3	Diameter	Real

A rectangle is to be punched out of a piece of sheet metal in the second job. For this job, you need two coordinates which define the upper left-hand point and the bottom right-hand point of the rectangle. You can transfer these job data to the punching machine collectively in the PLC data type "UDT_RectangleWindowStatic".

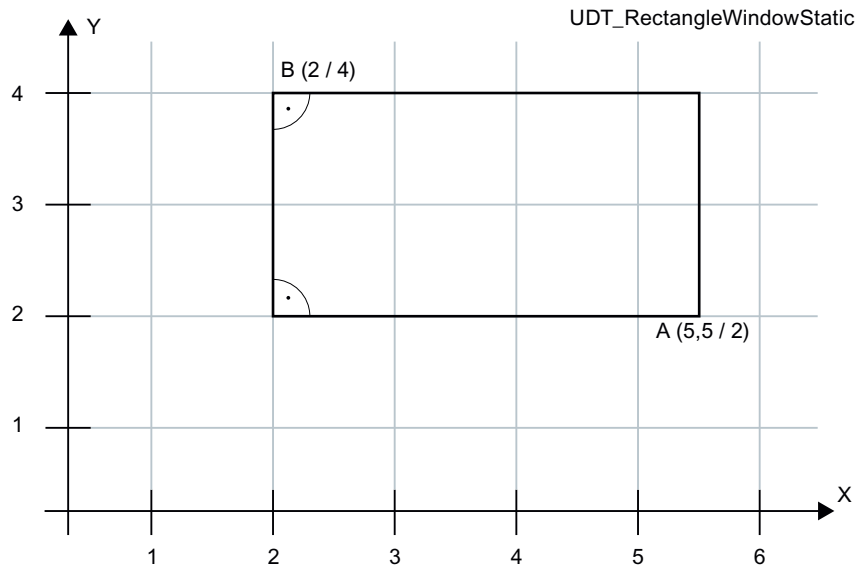

Create the PLC data type "UDT_RectangleWindowStatic":

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare the following lines within the PLC data type:
   X1-coordinate > REAL
   Y1-coordinate > REAL
   X2-coordinate > REAL
   Y2-coordinate > REAL

UDT_RectangleWindowStatic		
	Name	Data type
1	X1-coordinate	Real
2	Y1-coordinate	Real
3	X2-coordinate	Real
4	Y2-coordinate	Real

The job data of the "UDT_RectangleWindowStatic" can only be used to punch out rectangles whose edges are aligned parallel to the x and y axis.

If you want to punch out a rectangle with different alignment, i.e., not parallel to the x and y axis, you need an additional PLC data type. In it, you can specify the height and width, for example, as well as the alignment of the rectangle to the x axis by means of an angle.

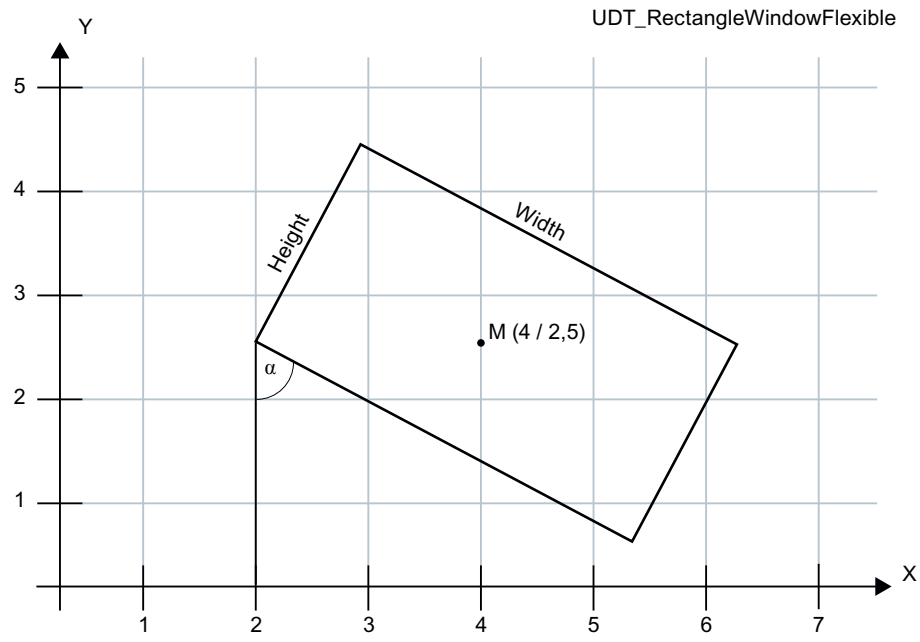

Create the PLC data type "UDT_RectangleWindowFlexible":

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare the following lines within the PLC data type:
   X-coordinate > REAL
   Y-coordinate > REAL
   Height > REAL
   Width > REAL
   Angle > REAL

UDT_RectangleWindowFlexible		
	Name	Data type
1	X-coordinate	Real
2	Y-coordinate	Real
3	Height	Real
4	Width	Real
5	Angle	Real

The x and y coordinates specify the center of the rectangle.

Procedure - Creating the data blocks

In the next section, you will learn how to transfer simple geometric shapes whose job data you have defined in the PLC data types to the punching machine. The punch jobs are broken down into individual punches in the program code; these punches are then executed consecutively by the punching machine. The punching machine has a cross table on which a sheet metal has been firmly clamped. You can move a cross table along the x and/or y axis just as in a coordinate system. The cross table is moved by two motors. The tool has different dies to punch out shapes from the sheet metal, such as circles and rectangles of different sizes. The tool can also be rotated by up to 90 degrees to cut out rectangles at different alignments.

You now use the PLC data types to create several instance data blocks. The instance data block then includes the specific values, for example, for a hole.

Create the instance data block "DB_OrderHole":

1. Double-click on the "Add new block" command under the "Program blocks" folder.
   The "Add new block" dialog box opens.
1. Click the "Data block (DB)" button.
2. Specify the name "DB_OrderHole".
3. Select the data block "UDT_Hole" as the type.
4. Click "OK".

Enter the appropriate start values:

DB_OrderHole

	Name	Data type	Start value
1	Static		
2	X-coordinate	Real	10.0
3	Y-coordinate	Real	900.0
4	Diameter	Real	4.0

To manufacture a specific sheet metal part such as the side panel of a control cabinet, for example, the necessary geometric shapes are loaded to the punching machine. For this purpose, you need to create another data block which includes a list of data blocks.

Create the data block "DB_OrderList":

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_OrderList".
4. Select "Global DB" as the type of the data block.
5. Click "OK".

Create the following job list in the data block:

DB_OrderList				
	Name	Data type	Start value	Comment
1	Static			
2	Order	Array[1..100] of DB_ANY		
3	Order[1]	DB_ANY	0	Supply
4	Order[2]	DB_ANY	0	Display
5	Order[3]	DB_ANY	0	Anchoring1
6	Order[4]	DB_ANY	0	Anchoring2

Procedure - Creating the program code

A separate function is created for each job type. The punch jobs are broken down into individual punches here and collected in an ARRAY.

1. Create the PLC data type "UDT_Punch".
2. Double-click the "Add new data type" command under "PLC data types".
   A new PLC data type with the name "UserDataType_x" is created.
3. Rename the PLC data type "UDT_Punch".
4. Declare the following lines within the PLC data type:
   Tool > DINT
   x > REAL
   y > REAL
   w > REAL

UDT_Punch		
	Name	Data type
1	Tool	DInt
2	x	Real
3	y	Real
4	w	Real

5. Create the ARRAY data block "DB_PunchList".

DB_PunchList		
	Name	Data type
1	DB_PunchList	Array[0..100] of "UDT_Punch"
2	DB_PunchList[0]	"UDT_Punch"
3	DB_PunchList[1]	"UDT_Punch"
4	DB_PunchList[2]	"UDT_Punch"
5	DB_PunchList[3]	"UDT_Punch"
6	DB_PunchList[4]	"UDT_Punch"
7	DB_PunchList[5]	"UDT_Punch"
8	DB_PunchList[6]	"UDT_Punch"
9	DB_PunchList[7]	"UDT_Punch"
10	DB_PunchList[8]	"UDT_Punch"

To prepare the punch job for a hole and break it down into individual punches, create an SCL function and name it "FC_PrepareHole".

1. Declare the block interface as follows:

FC_PrepareHole		
	Name	Data type
1	Input	
2	a	"UDT_Hole"
3	Output	
4	InOut	
5	Temp	
6	Constant	
7	Return	
8	FC_PrepareHole	Void

2. Write the following program code:

```
IF #a.Diameter >= 4.0 THEN
    "DB_PunchList"."THIS"[0].Tool := 1;
END_IF;

IF #a.Diameter >= 6.0 THEN
    "DB_PunchList"."THIS"[0].Tool := 2;
END_IF;

 "DB_PunchList"."THIS"[0].x := #a."X-coordinate";
 "DB_PunchList"."THIS"[0].y := #a."Y-coordinate";
```

To prepare the punch job for a window, you need a function which combines four punch sequences into the punch job. Create an SCL function and name it "FC_PrepareWindowStatic".

1. Declare the block interface as follows:

FC_PrepareWindowStatic		
	Name	Data type
1	Input	
2	a	"UDT_RectangleWindowStatic"
3	Output	
4	InOut	
5	Temp	
6	delta	Real
7	x	Real
8	y	Real
9	i	DInt
10	start	DInt
11	punches_x	DInt
12	punches_y	DInt
13	Constant	
14	Return	
15	FC_PrepareWindowStatic	Void

2. Write the following program code:

```
1  #delta := 6.0; // defined by tool 7
2  #i := 0;
3
4  #punches_x := REAL_TO_DINT(( #a.X2coordinate - #a.X1coordinate ) / #delta );
5  #punches_y := REAL_TO_DINT(( #a.Y2coordinate - #a.Y1coordinate ) / #delta );
6  #x := #a.X1coordinate + #delta / 2.0;
7  #y := #a.Y1coordinate + #delta / 2.0;
8
9  #start := 0;
10 FOR #i := 0 TO #punches_x-1 DO
11     "DB_PunchList"."THIS"[ #i + #start ].Tool := 7;
12     "DB_PunchList"."THIS"[ #i + #start ].x := #x + DINT_TO_REAL( #i )* #delta;
13     "DB_PunchList"."THIS"[ #i + #start ].y := #y;
14     "DB_PunchList"."THIS"[ #i + #start ].a := 0; // don't rotate
15 END_FOR;
16
17 #start += #punches_x;
18 FOR #i := 0 TO #punches_y-1 DO
19     "DB_PunchList"."THIS"[ #i + #start ].Tool := 7;
20     "DB_PunchList"."THIS"[ #i + #start ].x := #x + DINT_TO_REAL( #punches_x )* #delta;
21     "DB_PunchList"."THIS"[ #i + #start ].y := #y + DINT_TO_REAL( #i ) * #delta;
22     "DB_PunchList"."THIS"[ #i + #start ].a := 0; // don't rotate
23 END_FOR;
24
25 #start += #punches_y;
26 FOR #i := 0 TO #punches_x-1 DO
27     "DB_PunchList"."THIS"[ #i + #start ].Tool := 7;
28     "DB_PunchList"."THIS"[ #i + #start ].x := #x + DINT_TO_REAL( #punches_x - #i ) * #delta;
29     "DB_PunchList"."THIS"[ #i + #start ].y := #y + DINT_TO_REAL( #punches_y ) * #delta;
30     "DB_PunchList"."THIS"[ #i + #start ].a := 0; // don't rotate
31 END_FOR;
32
33 #start += #punches_x;
34 FOR #i := 0 TO #punches_y-1 DO
35     "DB_PunchList"."THIS"[ #i + #start ].Tool := 7;
36     "DB_PunchList"."THIS"[ #i + #start ].x := #x;
37     "DB_PunchList"."THIS"[ #i + #start ].y := #y + DINT_TO_REAL( #punches_y - #i) * #delta;
38     "DB_PunchList"."THIS"[ #i + #start ].a := 0; // don't rotate
39 END_FOR;
```

To prepare the punch job for a variable rectangle, you need a function which combines four punch sequences into the punch job. Create an SCL function and name it "FC_PrepareWindowFlexible".

1. Declare the block interface as follows:

FC_PrepareWindowFlexible		
	Name	Data type
1	Input	
2	a	"UDT_RectangleWindowFlexible"
3	Output	
4	InOut	
5	Temp	
6	delta	Real
7	delta_x	Real
8	delta_y	Real
9	x	Real
10	y	Real
11	i	DInt
12	start	DInt
13	punches_x	DInt
14	punches_y	DInt
15	Constant	
16	Return	
17	FC_PrepareWindowFlexible	Void

2. Write the following program code:

```
#delta := 6.0; // defined by tool 7
#i := 0;

#punches_x := REAL_TO_DINT(( #a.Width ) / #delta );
#punches_y := REAL_TO_DINT(( #a.Height ) / #delta );

// delta_x := ...
// delta_y := ...
// #x :=
// #y :=

#start := 0;
FOR #i := 0 TO #punches_x-1 DO
    "DB_PunchList"."THIS"[ #i + #start ].Tool := 7;
    "DB_PunchList"."THIS"[ #i + #start ].x := #x + DINT_TO_REAL( #i )* #delta_x;
    "DB_PunchList"."THIS"[ #i + #start ].y := #y + DINT_TO_REAL( #i )* #delta_y;
    "DB_PunchList"."THIS"[#i + #start].a := #a.Angle;
END_FOR;

// ....
```

The punching machine should now start processing the jobs. If it is already processing the jobs, it should get the next job from the job list and prepare it.

1. Create an SCL function block.
2. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.

3. Click the "Function block (FB)" button.
4. Specify the name "FB_PrepareNextOrder".
5. Declare the block interface as follows:

FB_PrepareNextOrder		
	Name	Data type
1	Input	
2	Output	
3	InOut	
4	Static	
5	OrderReady	Bool
6	OrderNo	UDInt
7	Order	DB_ANY
8	OrderCount	UDInt
9	Temp	
10	Error	Int

6. Declare the global tag "AllOrdersDone":

AllOrdersDone	Bool	%M0.0

7. Write the following program code:

```
IF #OrderReady AND NOT "AllOrdersDone" THEN
    IF #OrderNo < #OrderCount THEN
        #Order := "DB_OrderList".Order[#OrderNo];
        #OrderNo := #OrderNo + 1;
        #OrderReady := FALSE;
        "FC_PrepareOrder"( Order := #Order , Error => #Error );
    ELSE;
        "AllOrdersDone" := TRUE;
    END_IF;
END_IF;
```

The next job on the list is prepared by breaking down the current punch job into individual punches. The punching machine must be able to recognize which punch job it is processing.

1. Create an SCL function.
2. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
3. Click the "Function (FC)" button.
4. Specify the name "FC_PrepareOrder".

5. Declare the block interface as follows:

FC_PrepareOrder		
	Name	Data type
1	Input	
2	Order	DB_ANY
3	Output	
4	Error	Int
5	InOut	
6	Temp	
7	WindowFlexible	"UDT_RectangleWindowFlexible"
8	WindowStatic	"UDT_RectangleWindowStatic"
9	Hole	"UDT_Hole"
10	a	Variant
11	Constant	
12	Return	
13	FC_PrepareOrder	Void

6. Write the following program code:

```
#a := DB_ANY_TO_VARIANT(in := #Order, err => #Error);

IF TypeOf(#a) = UDT_Hole THEN
    "FC_PrepareHole"(#Hole);
END_IF;

IF TypeOf(#a) = UDT_RectangleWindowStatic THEN
    "FC_PrepareWindowStatic"(#WindowStatic);
END_IF;

IF TypeOf(#a) = UDT_RectangleWindowFlexible THEN
    "FC_PrepareWindowFlexible"(#WindowFlexible);
END_IF;
```

Call the SCL function "FC_PrepareOrder" in the SCL function block "FB_PrepareNextOrder"

```
IF #OrderReady AND NOT "AllOrdersDone" THEN
    IF #OrderNo < #OrderCount THEN
        #Order := "DB_OrderList".Order[#OrderNo];
        #OrderNo := #OrderNo + 1;
        #OrderReady := FALSE;
        "FC_PrepareOrder"( Order := #Order , Error => #Error );
    ELSE;
        "AllOrdersDone" := TRUE;
    END_IF;
END_IF;
```

Then, call the "FB_PrepareNextOrder" in OB1:

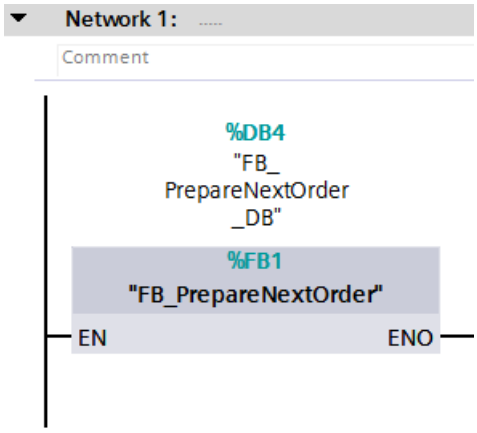

In addition to the option as described above, of loading the jobs with the help of the data block "DB_OrderList" to the punching machine in a pre-defined job, the job list could also be dynamically generated, for example. Or there is a selection of different job lists, for example. Then you have the option of loading a new job list to the punching machine after an job list has been processed. You need the following additional code for this.

Create an instance data block of the "FB_PrepareNextOrder" function block.

1. Double-click on the "Add new block" command under the "Program blocks" folder. The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_FB_PrepareNextOrder".
4. Select the data block "FB_PrepareNextOrder" as the type.
5. Click "OK".

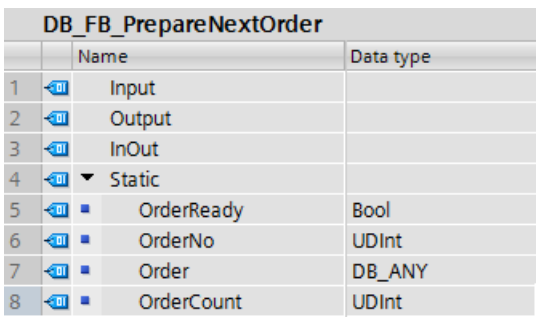

DB_FB_PrepareNextOrder

		Name	Data type
1		Input	
2		Output	
3		InOut	
4		Static	
5		OrderReady	Bool
6		OrderNo	UDInt
7		Order	DB_ANY
8		OrderCount	UDInt

In order to be able to swap the job list, create an additional SCL function:

1. Create the STL function "FC_SwapOrderList".
2. Declare the block interface as follows:

FC_SwapOrderList		
	Name	Data type
1	Input	
2	List	Array[*] of DB_ANY
3	ListLength	UDInt
4	Output	
5	InOut	
6	OrderList	"FB_PrepareNextOrder"
7	Temp	
8	Error	Int
9	Constant	
10	Return	
11	FC_SwapOrderList	Void

3. Declare the global tag "NewModelNr":

NewModelNr	Int	%MW2

4. Write the following program code:

```
IF "AllOrdersDone" = TRUE THEN
    #Error := MOVE_BLK_VARIANT(
        SRC := #List,
        COUNT := #ListLength,
        SRC_INDEX := 0,
        DEST_INDEX := 0,
        DEST => "DB_OrderList".Order);
    IF #Error = 0 THEN
        #OrderList.OrderNo := 0;
        #OrderList.OrderReady := FALSE;
        #OrderList.OrderCount := #ListLength;
        "AllOrdersDone" := FALSE;
        "NewModelNr" := 0;
    END_IF;
END_IF;
```

Create the data block "DB_OrderRepository":

1. Double-click on the "Add new block" command under the "Program blocks" folder. The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_OrderRepository".
4. Select "Global DB" as the type of the data block.
5. Click "OK".

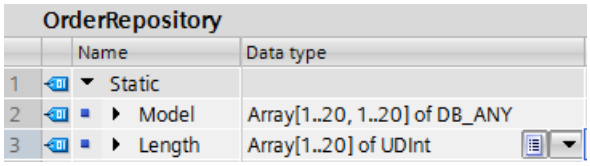

Then, call the "FC_SwapOrderList" in OB1:

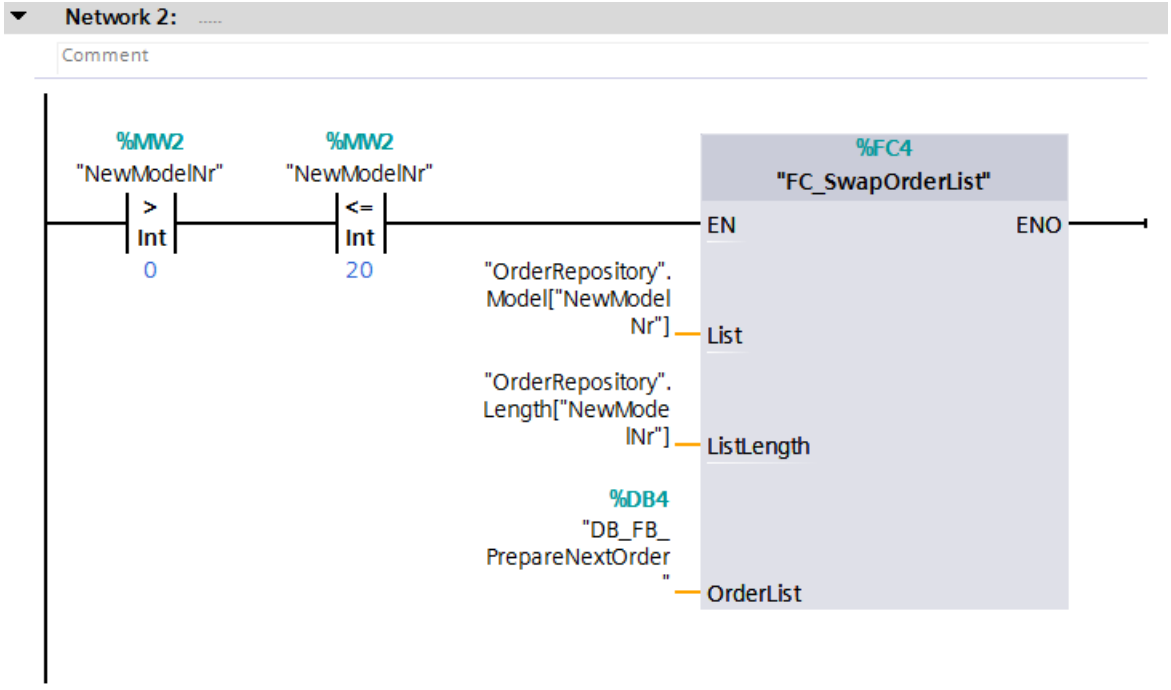

Result

This example shows you how to use the instruction "DB_ANY_TO_VARIANT: Convert DB_ANY to VARIANT" to determine the PLC data type of a data block and how to select and execute a suitable function based on this.

See also

Addressing variables in global data blocks (Page 106)

2.5 Using PLC data types (UDT) (S7-1200, S7-1500)

Using a PLC data type

PLC data types (UDT) are data structures that you define yourself and that can be used multiple times in the program. The structure can be composed of multiple elements of different data types. You define the data types of the individual elements during the declaration of a PLC data type.

PLC data types are frequently used if an assembled data record with various data types is required and these are to be processed from different points in the program. For example, these could be:

- Data records for material tracking
- Parameter sets for a motor setting
- Recipes

The use of PLC data types provides you with the following benefits:

- The elements of a PLC data type can also be addressed indirectly, which means the address is variable and is not calculated until during runtime.
- Tags based on a PLC data type inherit all properties of the PLC data type. When there is a change to the PLC data type, all tags based on this PLC data type are therefore adapted automatically.
- Using universal symbolism makes the program easier to read, because the names of the individual elements of a PLC data type are displayed in the program.
- Optimum utilization of the high performance of an S7-1500 CPU.
- The PLC data type can be passed as a complete structure for a block call.
- A simplified call interface due to a lower number of parameters to be supplied.

Procedure in STEP 7 V5.x

STEP 7 V5.x already gave you the option to create a data record as a structured tag using the STRUCT data type or a PLC data type (UDT). However, the performance was adversely affected by the use of symbolic addressing.

The declaration in the data blocks was mostly implemented as an anonymous structure. The blocks themselves were then programmed to pass the values of the structure as actual parameters and the calculated values were copied back into the structure. This enabled you to also pass the data block number and use absolute addressing in the block. The number of parameters that you had to supply was often very large. The actual data was stored in the data blocks and the calculated values were passed to other blocks. However, no symbolism was available when passing the data block tags.

Programming example in STEP 7 TIA Portal

You can assign both a formula parameter and an actual parameter to a PLC data type. This means that you no longer have to declare individual parameters. If a block has an input parameter that is based on a PLC data type, you must transfer a tag of the same PLC data type as actual parameter.

The following example shows the call and the parameter assignment of a function block (FB) with two formal parameters:

1. To create a PLC data type, double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Rename the PLC data type to "UDT_Material".
3. Declare the following lines within the PLC data type:
   ArticleNumber > Data type: DINT
   ArticleName > Data type: STRING
   Amount > Data type: REAL
   Unit > Data type: STRING

UDT_Material

		Name	Data type
1		ArticleNumber	DInt
2		ArticleName	String
3		Amount	Real
4		Unit	String

Use the PLC data type within a global data block. You can specify the PLC data type either directly as data type of the data block or within the data block as data type of a tag.

Create a global data block for this purpose:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_MaterialBuffer".
4. Select the type of the data block "ARRAY DB".
5. Select the PLC data type "UDT_Material" as ARRAY data type.

6. Specify "1000" as high ARRAY limit.
7. Click "OK".

DB_MaterialBuffer

	Name	Data type
1	DB_MaterialBuffer	Array[0..1000] of "UDT_Material"
2	DB_MaterialBuffer[0]	"UDT_Material"
3	ArticleNumber	DInt
4	ArticleName	String
5	Amount	Real
6	Unit	String
7	DB_MaterialBuffer[1]	"UDT_Material"
8	DB_MaterialBuffer[2]	"UDT_Material"
9	DB_MaterialBuffer[3]	"UDT_Material"
10	DB_MaterialBuffer[4]	"UDT_Material"
11	DB_MaterialBuffer[5]	"UDT_Material"

At the function block call, interconnect the formal parameters with tags from the global data block "DB_MaterialBuffer".

1. Create an SCL function block and name it "FB_Material".
2. Declare the block interface as follows:

FB_Material

	Name	Data type
1	Input	
2	Source	Variant
3	SourceIndex	DInt
4	DestinationIndex	DInt
5	Output	
6	InOut	
7	Destination	Variant
8	Static	
9	Temp	
10	Result	Int
11	Constant	

3. Write the following program code:

```
1
2 #Result := MOVE_BLK_VARIANT(SRC:=#Source,
3                             COUNT:=3,
4                             SRC_INDEX:=#SourceIndex,
5                             DEST_INDEX:=#DestinationIndex,
6                             DEST=>#Destination);
```

4. Call the "FB_Material" function block in OB1 and interconnect the formal parameters with tags of the global data block "DB_MaterialBuffer":

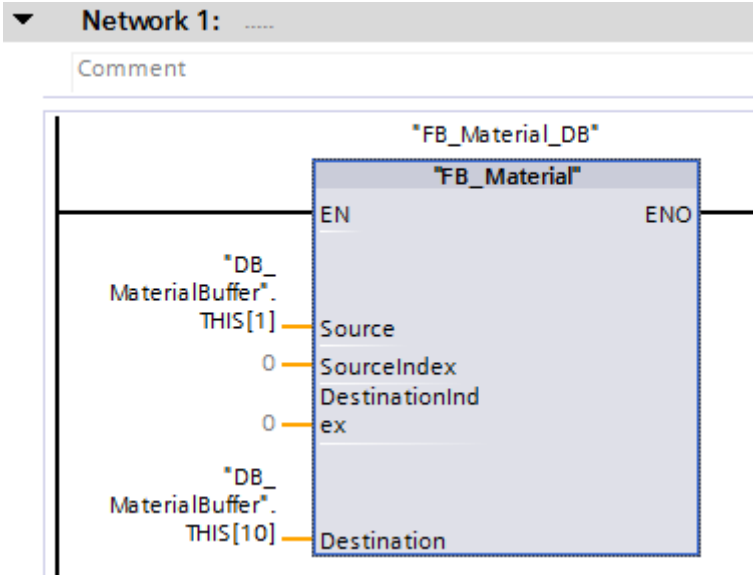

The material data are moved within the global data block "DB_MaterialBuffer".

See also

Other useful information regarding structured PLC tags (Page 7084)

Creating structured PLC tags (Page 7086)

Rules for supplying block parameters (Page 76)

Basics of PLC data types (UDT) (Page 273)

PLC data types (UDT) (Page 273)

2.6 Using MOVE instructions in STL (S7-1500)

Possible applications

You can now program with MOVE instructions in STL on an S7-1500 CPU.

This gives you the following advantages:

- The program structure is easier to create.
- The performance of the CPU increases.

Programming in STEP 7 V5.x

In STEP 7 V5.x, you used the "BLKMOV: Move block" and "UBLKMOV: Move block uninterruptible" system functions to implement the MOVE functionality.

Procedure in STEP 7 TIA Portal

The following new MOVE instructions are available in STEP 7 TIA Portal:

- MOVE: Move value
- MOVE_BLK: Move block
- MOVE_BLK_VARIANT: Move block
- UMOVE_BLK: Move block uninterruptible

Programming example

The following example shows you how the "MOVE_BLK instruction works: Move block". An ARRAY block is copied into another ARRAY block:

Data_DB

	Name	Data type
1	Static	
2	Array_1	Array [0..10] of Int
3	Array_1[0]	Int
4	Array_1[1]	Int
5	Array_1[2]	Int
6	Array_1[3]	Int
7	Array_1[4]	Int
8	Array_1[5]	Int
9	Array_1[6]	Int
10	Array_1[7]	Int
11	Array_1[8]	Int
12	Array_1[9]	Int
13	Array_1[10]	Int
14	Array_2	Array [0..10] of Int
15	Array_2[0]	Int
16	Array_2[1]	Int
17	Array_2[2]	Int
18	Array_2[3]	Int
19	Array_2[4]	Int
20	Array_2[5]	Int
21	Array_2[6]	Int
22	Array_2[7]	Int
23	Array_2[8]	Int
24	Array_2[9]	Int
25	Array_2[10]	Int

Network 1:

Comment

```
1    CALL  MOVE_BLK
2        Int  UInt
3        IN    :="Data_DB".Array_1[0]
4        COUNT :=10
5        OUT   :="Data_DB".Array_2[0]
6
7
```

Using the MOVE_BLK instruction, ten elements from "Array_1" of the "Data_DB" data block are copied into "Array_2" of the same data block.

See also

MOVE: Move value (Page 1333)

MOVE_BLK: Move block (Page 1345)

MOVE_BLK_VARIANT: Move block (Page 1347)
UMOVE_BLK: Move block uninterruptible (Page 1351)

2.7 Using IEC timers and counters (S7-1200, S7-1500)

Advantages of IEC timers and counters

The universal use of IEC timers and counters makes your program code more efficient.

This gives you the following advantages:

- The blocks can be called multiple times with newly generated instance data blocks.
- The IEC counters have a large counting range.
- Compared to S5 timers, IEC timers have better performance and greater time accuracy.

Programming in STEP 7 V5.x

The S5 timers and counters in STEP 7 V5.x were addressed absolutely via a number. Due to this dependency on numbers, program blocks were not reusable with S5 timers and counters.

The value range of a timer was limited to 9990 seconds and that of a counter limited to a maximum of 999.

Procedure in STEP 7 TIA Portal

You declare IEC timers and counters in the program block where they are called or needed. The IEC timer is a structure of data type IEC_TIMER, IEC_LTIMER, or TON_TIME and TON_LTIME, for example, which you can also declare as a local tag in a block. The IEC counter is a structure of data type IEC_SCOUNTER, IEC_USCOUNTER, etc.

Programming examples in the TIA Portal

The following example shows you how to declare an IEC timer and IEC counter as a local tag:

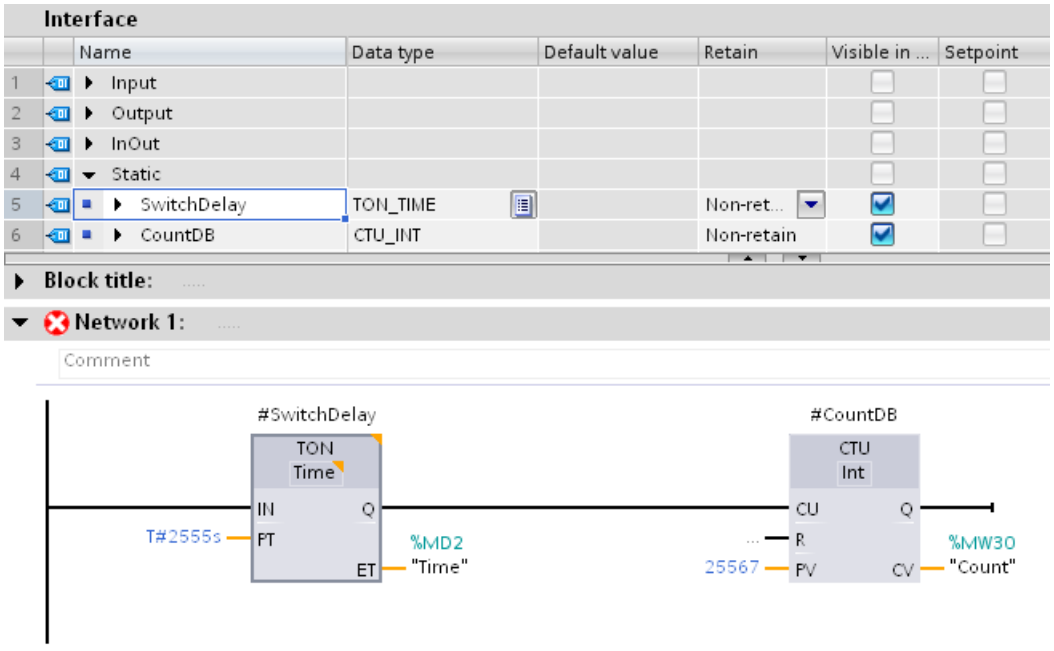

The data of the TON IEC timer and the CTU IEC counter is stored as a local tag (multi-instance) in the block interface.

You also have the option to create IEC timers and IEC counters within structures as multiple instances and to use them in your program code.

1. To do this, for example, create a global data block with an ARRAY of TON. The data type TON does not appear in the drop-down list, but can be entered manually:

Global ARRAY DB

		Name	Data type
1		Static	
2		Timer	Array[0..10] of TON
3		Timer[0]	TON
4		Timer[1]	TON
5		Timer[2]	TON
6		Timer[3]	TON
7		Timer[4]	TON
8		Timer[5]	TON
9		Timer[6]	TON
10		Timer[7]	TON
11		Timer[8]	TON
12		Timer[9]	TON
13		Timer[10]	TON

2. Create a function block and drag the instruction "TON: Generate on-delay" to a network. Call the instance of IEC timer TON as follows:

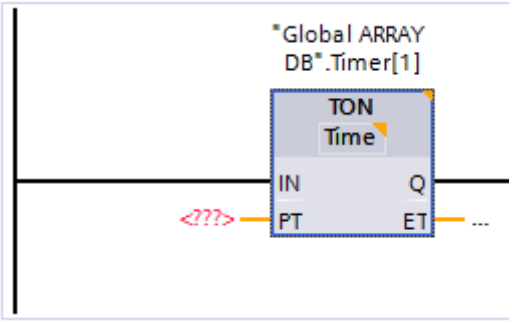

Call of a timer as multi-instance

If you want to use the IN parameter to start a timer as multi-instance, you must not initialize it beforehand in your program code. In this case, the timer called at the IN parameter no longer recognizes a positive signal edge and the timer is not started:

1. Create the timer "Time_1" as a multi-instance of the data type "TP_TIME" in the "Static" section of the block interface.

2. Write the following program code:

STL	Explanation
`A "Tag_Output"`	`// When the "Tag_Output" output receives the signal state 1,`

STL	Explanation
= #Time_1.IN	// the IN parameter of the multi-instance timer #Timer_1 is initialized with a positive signal edge.

CALL #Time_1
time_type := Time
IN := "Tag_Output"
PT := T#30s
Q := "Tag_4"
ET := "Tag_ElapsedTime"
	// When the multi-instance timer is now called and the IN parameter queried again, the timer is not started because there is no new positive signal edge.
	
	// Enter TIME as data type of the instruction.

This is why you must program the initialization of the multi-instance timer within the call.

STL	Explanation

CALL #Time_1
time_type := Time
IN := "Tag_Output"
PT := T#30s
Q := "Tag_4"
ET := "Tag_ElapsedTime"
	// The timer is called and started.
	
	// Enter TIME as data type of the instruction.

See also

Calling IEC timers (Page 1661)

Calling IEC counters (Page 1691)

Timers (Page 255)

Data types 3

3.1 Overview of the valid data types

Validity of data type groups

The data type groups define the properties of the data, for example, the representation of the contents and the valid memory areas.

In the user program, you have the option of using pre-defined data types, which you add to the data types you have defined yourself. The following type categories are available for this:

- Elementary data types (binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD CHAR, WCHAR)
- Complex data types (DT, LDT, DTL, STRING, WSTRING, ARRAY, STRUCT)
- User-defined data types (PLC data type (UDT))
- Pointer
- Parameter types
- System data types
- Hardware data types

The following tables show the availability of data types in the various S7-CPUs:

Table 3-1 Binary numbers

Binary numbers	S7-300/400	S7-1200	S7-1500
BOOL (Page 236)	X	X	X
Bit strings			
BYTE (Page 236)	X	X	X
WORD (Page 237)	X	X	X
DWORD (Page 238)	X	X	X
LWORD (Page 240)	-	-	X

Table 3-2 Integers

Integers	S7-300/400	S7-1200	S7-1500
SINT (Page 241)	-	X	X
INT (Page 243)	X	X	X
DINT (Page 245)	X	X	X
USINT (Page 242)	-	X	X
UINT (Page 244)	-	X	X
UDINT (Page 246)	-	X	X

Integers	S7-300/400	S7-1200	S7-1500
LINT (Page 247)	-	-	X
ULINT (Page 248)	-	-	X

Table 3-3 Floating-point numbers

Floating-point numbers	S7-300/400	S7-1200	S7-1500
REAL (Page 251)	X	X	X
LREAL (Page 252)	-	X	X

Table 3-4 Timers

Timers	S7-300/400	S7-1200	S7-1500
S5TIME (Page 255)	X	-	X
TIME (Page 256)	X	X	X
LTIME (Page 257)	-	-	X

Table 3-5 Date and time

Date and time	S7-300/400	S7-1200	S7-1500
DATE (Page 258)	X	X	X
TIME_OF_DAY (TOD) (Page 258)	X	X	X
LTOD (LTIME_OF_DAY) (Page 259)	-	-	X
DT (DATE_AND_TIME) (Page 259)	X	-	X
LDT (Page 261)	-	-	X
DTL (Page 261)	-	X	X

Table 3-6 Character strings

Character strings	S7-300/400	S7-1200	S7-1500
CHAR (Page 263)	X	X	X
WCHAR (Page 263)	-	X	X
STRING (Page 264)	X	X	X
WSTRING (Page 267)	-	X	X

Table 3-7 PLC data types (UDT)

PLC data types (UDT)	S7-300/400	S7-1200	S7-1500
PLC data type (Page 273) (UDT)	X	X	X

Table 3-8 Anonymous structures

Anonymous structures	S7-300/400	S7-1200	S7-1500
STRUCT (Page 278)	X	X	X

Table 3-9 ARRAY

ARRAY	S7-300/400	S7-1200	S7-1500
ARRAY [....] of <data type> (Page 283)	X	X	X

Table 3-10 Pointer

Pointer	S7-300/400	S7-1200	S7-1500
References (Page 300)	-	-	X
VARIANT (Page 324)	-	X	X
POINTER (Page 341)	X	-	X
ANY (Page 344)	X	-	X

Table 3-11 Parameter types

Parameter types	S7-300/400	S7-1200	S7-1500
TIMER (Page 348)	X	-	X
COUNTER (Page 348)	X	-	X
BLOCK_FC (Page 348)	X	-	X
BLOCK_FB (Page 348)	X	-	X
BLOCK_DB (Page 348)	X	-	-
BLOCK_SDB (Page 348)	X	-	-
VOID (Page 348)	X	X	X
PARAMETER (Page 348)	-	X	X

Table 3-12 System data types

System data types	S7-300/400	S7-1200	S7-1500
IEC_TIMER (Page 349)	X[1)]	X	X
IEC_LTIMER (Page 349)	-	-	X
IEC_SCOUNTER (Page 349)	-	X	X
IEC_USCOUNTER (Page 349)	-	X	X
IEC_COUNTER (Page 349)	X[2)]	X	X
IEC_UCOUNTER (Page 349)	-	X	X
IEC_DCOUNTER (Page 349)	-	X	X
IEC_UDCOUNTER (Page 349)	-	X	X
IEC_LCOUNTER (Page 349)	-	-	X
IEC_ULCOUNTER (Page 349)	-	-	X

System data types	S7-300/400	S7-1200	S7-1500
ERROR_STRUCT (Page 349)	-	X	X
NREF (Page 349)	-	X	X
CREF (Page 349)	-	X	X
VREF (Page 349)	-	X	X
SSL_HEADER (Page 349)	X	-	-
CONDITIONS (Page 349)	-	X	-
TADDR_Param (Page 349)	-	X	X
TCON_Param (Page 349)	-	X	X
HSC_Period (Page 349)	-	X	-
1) For S7-300/400 CPUs, the data type is represented by TP, TON and TOF. 2) For S7-300/400 CPUs, the data type is represented by CTU, CTD and CTUD.			

Table 3-13 Hardware data types

Hardware data types	S7-300/400	S7-1200	S7-1500
REMOTE (Page 351)	-	X	X
HW_ANY (Page 351)	-	X	X
HW_DEVICE (Page 351)	-	X	X
HW_DPMASTER (Page 351)	-	-	X
HW_DPSLAVE (Page 351)	-	X	X
HW_IO (Page 351)	-	X	X
HW_IOSYSTEM (Page 351)	-	X	X
HW_SUBMODULE (Page 351)	-	X	X
HW_MODULE (Page 351)	-	-	X
HW_INTERFACE (Page 351)	-	X	X
HW_IEPORT (Page 351)	-	X	X
HW_HSC (Page 351)	-	X	X
HW_PWM (Page 351)	-	X	X
HW_PTO (Page 351)	-	X	X
EVENT_ANY (Page 351)	-	X	X
EVENT_ATT (Page 351)	-	X	X
EVENT_HWINT (Page 351)	-	X	X
OB_ANY (Page 351)	-	X	X
OB_DELAY (Page 351)	-	X	X
OB_TOD (Page 351)	-	X	X
OB_CYCLIC (Page 351)	-	X	X
OB_ATT (Page 351)	-	X	X
OB_PCYCLE (Page 351)	-	X	X
OB_HWINT (Page 351)	-	X	X
OB_DIAG (Page 351)	-	X	X
OB_TIMEERROR (Page 351)	-	X	X
OB_STARTUP (Page 351)	-	X	X

Hardware data types	S7-300/400	S7-1200	S7-1500
PORT (Page 351)	-	X	X
RTM (Page 351)	-	X	X
PIP (Page 351)	-	-	X
CONN_ANY (Page 351)	-	X	X
CONN_PRG (Page 351)	-	X	X
CONN_OUC (Page 351)	-	X	X
CONN_R_ID (Page 351)	-	-	X
DB_ANY (Page 351)	-	X	X
DB_WWW (Page 351)	-	X	X
DB_DYN (Page 351)	-	X	X

Note

Depending on the CPU version, the actually valid data types can deviate slightly from the table.

3.2 Binary numbers

3.2.1 BOOL (bit)

Description

An operand of data type BOOL represents a bit value and contains one of the following values:

- TRUE
- FALSE

The following table shows the properties of data type BOOL:

Length (bits)	Format	Value range	Examples of value input
1	Boolean	FALSE or TRUE BOOL#0 or BOOL#1 BOOL#FALSE or BOOL#TRUE	TRUE BOOL#1 BOOL#TRUE
	Unsigned integers (decimal system)	0 or 1	1
	Binary numbers	2#0 or 2#1	2#0
	Octal numbers	8#0 or 8#1	8#1
	Hexadecimal numbers	16#0 or 16#1	16#1

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", the bit has a length of 1 byte.

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.2.2 Bit strings

3.2.2.1 BYTE

Description

An operand of data type BYTE is a bit string of 8 bits.

The following table shows the properties of data type BYTE:

Length (bits)	Format	Value range	Examples of value input	
			Constants	Absolute and symbolic addresses
8	Integers[1] (decimal system)	Signed integers: -128 to +127 Unsigned integers: 0 to 255	• 15 • BYTE#15 • BYTE#10#15 • B#15	• IB2 • MB10 • DB1.DBB4 • Tag_Name
	Binary numbers	2#0 to 2#1111_1111	• 2#0000_1111 • BYTE#2#0000_1111 • B#2#0000_1111	
	Octal numbers	8#0 to 8#377	• 8#17 • BYTE#8#17 • B#8#17	
	Hexadecimal numbers	16#0 to 16#FF	• 16#0F • BYTE#16#0F • B#16#0F	
[1] The value range depends on the relevant interpretation or conversion.				

Note

The BYTE data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the SINT and USINT data types.

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.2.2.2 WORD

Description

An operand of data type WORD is a bit string of 16 bits.

The following table shows the properties of data type WORD:

Length (bits)	Format	Value range	Examples of value input	
			Constants	Absolute and symbolic addresses
16	Integers (decimal system)	Signed integers: -32_768 to +32_767 Unsigned integers: 0 to 65_535	• 61_680 • WORD#61_680 • WORD#10#61_680 • W#61_680	• MW10 • DB1.DBW2 • Tag_Name
	Binary numbers	2#0 to 2#1111_1111_1111_1111	• 2#1111_0000_1111_0000 • WORD#2#1111_0000_1111_0000 • W#2#1111_0000_1111_0000	
	Octal numbers	8#0 to 8#177_777	• 8#170_360 • WORD#8#170_360 • W#8#170_360	
	Hexadecimal numbers	16#0 to 16#FFFF	• 16#F0F0 • WORD#16#F0F0 • W#16#F0F0	
	BCD	C#0 to C#999	C#55	
	Decimal sequence	B#(0, 0) to B#(255, 255)	B#(127, 200)	

Note

The WORD data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the INT and UINT data types.

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.2.2.3 DWORD

Description

An operand of data type DWORD is a bit string of 32 bits.

The following table shows the properties of data type DWORD:

Length (bits)	Format	Value range	Examples of value input	
			Constants	Absolute and symbolic addresses
32	Integers (decimal system)	Signed integers: -2_147_483_647 to +2_147_483_647 Unsigned integers: 0 to 4_294_967_295	• +15_793_935 • DWORD#+15_793_935 • DWORD#10# +15_793_935 • DW#+15_793_935	• MD10 • DB1.DBD8 • Tag_Name
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111	• 2#0000_0000_1111_0000_1111_1111_0000_1111 • DWORD#2#0000_0000_1111_0000_1111_1111_0000_1111 • DW#2#0000_0000_1111_0000_1111_1111_0000_1111	
	Octal numbers	8#0 to 8#37_777_777_777	• 8#74_177_417 • DWORD#8#74_177_417 • DW#8#74_177_417	
	Hexadecimal numbers	16#0000_0000 to 16#FFFF_FFFF	• 16#00F0_FF0F • DWORD#16#00F0_FF0F • DW#16#00F0_FF0F	
	Decimal sequence	B#(0, 0, 0, 0) to B#(255, 255, 255, 255)	B#(127, 200, 127, 200)	

Note

The DWORD data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the DINT and UDINT data types.

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.2.2.4 LWORD (S7-1500)

Description

An operand of data type LWORD is a bit string of 64 bits.

The following table shows the properties of data type LWORD:

Length (bits)	Format	Value range	Examples of value input
64	Integers (decimal system)	Signed integers: -9_223_372_036_854_775_808 to +9_223_372_036_854_775_807 Unsigned integers: 0 to 18_446_744_073_709_551_615	• +26_123_590_360_715 • LWORD#+26_123_590_360_715 • LWORD#10#+26_123_590_360_715 • LW#+26_123_590_360_715
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111	• 2#0000_0000_0000_0000_0000_1011_1110_0001_0010_1111_0101_0010_1101_1110_1000_1011 • LWORD#2#0000_0000_0000_0000_0000_1011_1110_0001_0010_1111_0101_0010_1101_1110_1000_1011 • LW#2#0000_0000_0000_0000_0000_1011_1110_0001_0010_1111_0101_0010_1101_1110_1000_1011
	Octal numbers	8#0 to 8#1_777_777_777_777_777_777_777	• 8#13_724_557_213 • LWORD#8#13_724_557_213 • LW#8#13_724_557_213
	Hexadecimal numbers	16#0000_0000 to 16#FFFF_FFFF_FFFF_FFFF	• 16#0000_0000_5F52_DE8B • LWORD#16#0000_0000_5F52_DE8B • LW#16#0000_0000_5F52_DE8B
	Decimal sequence	B#(0, 0, 0, 0, 0, 0, 0, 0) to B#(255, 255, 255, 255, 255, 255, 255, 255)	B#(127, 200, 127, 200, 127, 200, 127, 200)

Note

The LWORD data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the LINT and ULINT data types.

See also

Overview of the valid data types (Page 231)

Overview of data type conversion (Page 354)

Basics of constants (Page 97)

Implicit conversions (Page 356)

Explicit conversions (Page 405)

Data type conversion for S7-1200 (Page 489)

3.3 Integers

3.3.1 SINT (8-bit integers) (S7-1200, S7-1500)

Description

An operand of data type SINT (Short INT) has a length of 8 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 6 represent the number value. The signal state of bit 7 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

An operand of data type SINT occupies one BYTE in the memory.

The following table shows the properties of data type SINT:

Length (bits)	Format	Value range	Examples of value input
8	Signed integers (decimal system)	-128 to +127	• +44 • SINT#+44 • SINT#10#+44 The value range extends to a maximum of SINT#255 when using the type SINT#. This value is interpreted as an integer with -1.
	Binary numbers (only positive)	2#0 to 2#0111_1111	• 2#0010_1100 • SINT#2#0010_1100 • SINT#2#10
	Octal numbers (only positive)	8#0 to 8#177	• 8#54 • SINT#8#54
	Hexadecimal numbers (only positive)	16#0 to 16#7F	• 16#2C • SINT#16#2C The value range extends to a maximum of SINT#16#FF when using the type SINT#. This value is interpreted as an integer with -1.

Example

The following figure shows the integer +44 as a binary number:

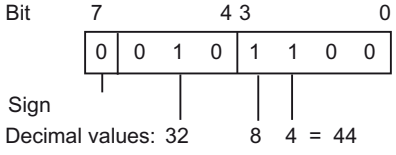

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.3.2 USINT (8-bit integers) (S7-1200, S7-1500)

Description

An operand of data type USINT (Unsigned Short INT) has a length of 8 bits and contains unsigned numerical values:

An operand of data type USINT occupies one BYTE in the memory.

The following table shows the properties of data type USINT:

Length (bits)	Format	Value range	Examples of value input
8	Unsigned integers (decimal system)	0 to 255	• 78 • USINT#78 • USINT#10#78
	Binary numbers	2#0 to 2#1111_1111	• 2#0100_1110 • USINT#2#0100_1110 • USINT#2#10
	Octal numbers	8#0 to 8#377	• 8#116 • USINT#8#116
	Hexadecimal numbers	16#0 to 16#FF	• 16#4E • USINT#16#4E

Example

The following figure shows the integer 78 as a binary number:

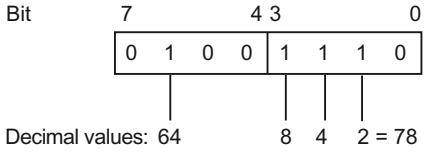

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.3.3 INT (16-bit integers)

Description

An operand of data type INT has a length of 16 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 14 represent the number value. The signal state of bit 15 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

An operand of data type INT occupies two BYTE in the memory.

The following table shows the properties of data type INT:

Length (bits)	Format	Value range	Examples of value input
16	Signed integers (decimal system)	-32_768 to +32_767	• +3_785 • INT#+3_785 • INT#10#+3_785
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_1111	• 2#0000_1110_1100_1001 • INT#2#0000_1110_1100_1001 • INT#2#10
	Octal numbers (only positive)	8#0 to 8#7_7777	• 8#7311 • INT#8#7311
	Hexadecimal numbers (only positive)	16#0 to 16#7FFF	• 16#0EC9 • INT#16#0EC9

Example

The following figure shows the integer +3785 as a binary number:

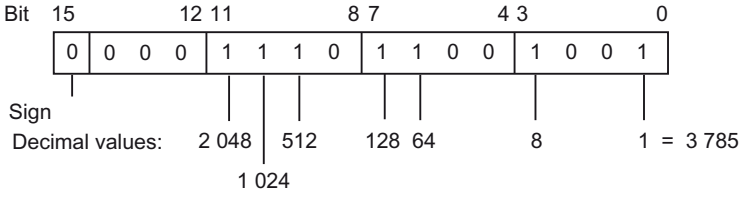

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.3.4 UINT (16-bit integers) (S7-1200, S7-1500)

Description

An operand of data type UINT (Unsigned INT) has a length of 16 bits and contains unsigned numerical values.

An operand of data type UINT occupies two BYTE in the memory.

The following table shows the properties of data type UINT:

Length (bits)	Format	Value range	Examples of value input
16	Unsigned integers (decimal system)	0 to 65_535	• 65_295 • UINT#65_295 • UINT#10#65_295
	Binary numbers	2#0 to 2#1111_1111_1111_1111	• 2#1111_1111_0000_1111 • UINT#2#1111_1111_0000_1111 • UINT#2#10
	Octal numbers	8#0 to 8#17_7777	• 8#17_7417 • UINT#8#17_7417
	Hexadecimal numbers	16#0 to 16#FFFF	• 16#FF0F • UINT#16#FF0F

Example

The following figure shows the integer 65295 as a binary number:

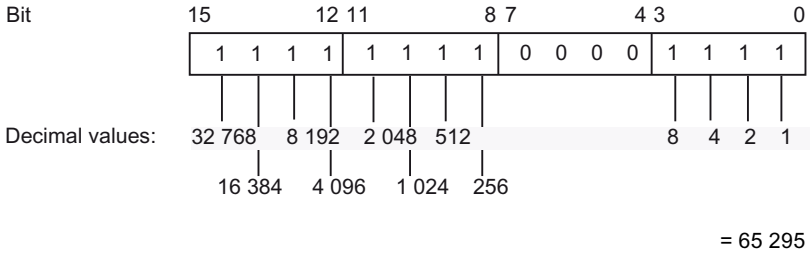

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.3.5 DINT (32-bit integers)

Description

An operand of data type DINT (Double INT) has a length of 32 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 30 represent the number value. The signal state of bit 31 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

An operand of data type DINT occupies four BYTE in the memory.

The following table shows the properties of data type DINT:

Length (bits)	Format	Value range	Examples of value input
32	Signed integers (decimal system)	-2_147_483_648 to +2_147_483_647	• +125_790 • DINT#+125_790 • DINT#10#+125_790 • L#275
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_1111_1111_1111_1111_1111	• 2#0000_0000_0000_0001_1110_1011_0101_1110 • DINT#2#0000_0000_0000_0001_1110_1011_0101_1110 • DINT#2#10
	Octal numbers (only positive)	8#0 to 8#177_7777_7777	• 8#36_5536 • DINT#8#36_5536
	Hexadecimal numbers	16#0 to 16#7FFF_FFFF	• 16#0001_EB5E • DINT#16#0001_EB5E

Example

The following figure shows the integer +125790 as a binary number:

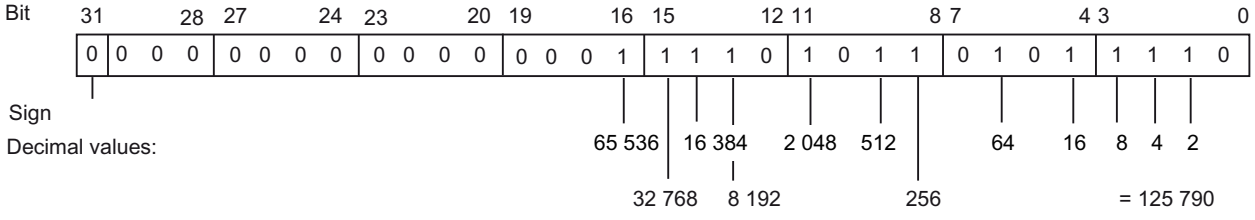

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.3.6 UDINT (32-bit integers) (S7-1200, S7-1500)

Description

An operand of data type UDINT (Unsigned Double INT) has a length of 32 bits and contains unsigned numerical values.

An operand of data type UDINT occupies four BYTE in the memory.

The following table shows the properties of data type UDINT:

Length (bits)	Format	Value range	Examples of value input
32	Unsigned integers (decimal system)	0 to 4_294_967_295	• 4_042_322_160 • UDINT#4_042_322_160 • UDINT#10#4_042_322_160
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111	• 2#1111_0000_1111_0000_1111_0000_1111_0000 • UDINT#2#1111_0000_1111_0000_1111_0000_1111_0000 • UDINT#2#10
	Octal numbers	8#0 to 8#377_7777_7777	• 8#360_7417_0360 • UDINT#8#360_7417_0360
	Hexadecimal numbers	16#0 to 16#FFFF_FFFF	• 16#F0F0_F0F0 • UDINT#16#F0F0_F0F0

Example

The following figure shows the integer 4042322160 as a binary number:

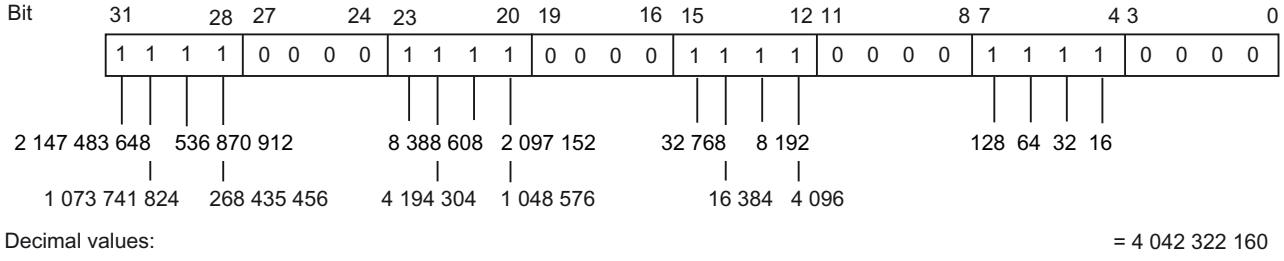

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.3.7 LINT (64-bit integers) (S7-1500)

Description

An operand of data type LINT (Long INT) has a length of 64 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 62 represent the number value. The signal state of bit 63 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

An operand of data type LINT occupies eight BYTE in the memory.

The following table shows the properties of data type LINT:

Length (bits)	Format	Value range	Examples of value input
64	Signed integers (decimal system)	-9_223_372_036_854_775_808 to +9_223_372_036_854_775_807	• +154_325_790_816_159 • LINT#+154_325_790_816_159 • LINT#10#+154_325_790_816_159
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111	• 2#0000_0000_0000_0000_1000_1100_0101_1011_1100_0101_1111_0000_1111_0111_1001_1111 • LINT#2#0000_0000_0000_0000_1000_1100_0101_1011_1100_0101_1111_0000_1111_0111_1001_1111 • LINT#2#10
	Octal numbers (only positive)	8#0 to 8#7_7777_7777_7777_7777_7777	• 8#4305_5705_7417_3637 • LINT#8#4305_5705_7417_3637
	Hexadecimal numbers (only positive)	16#0 to 16#7FFF_FFFF_FFFF_FFFF	• 16#0000_8C5B_C5F0_F79F • LINT#16#0000_8C5B_C5F0_F79F

Example

The following figure shows the integer +154325790816159 as a binary number:

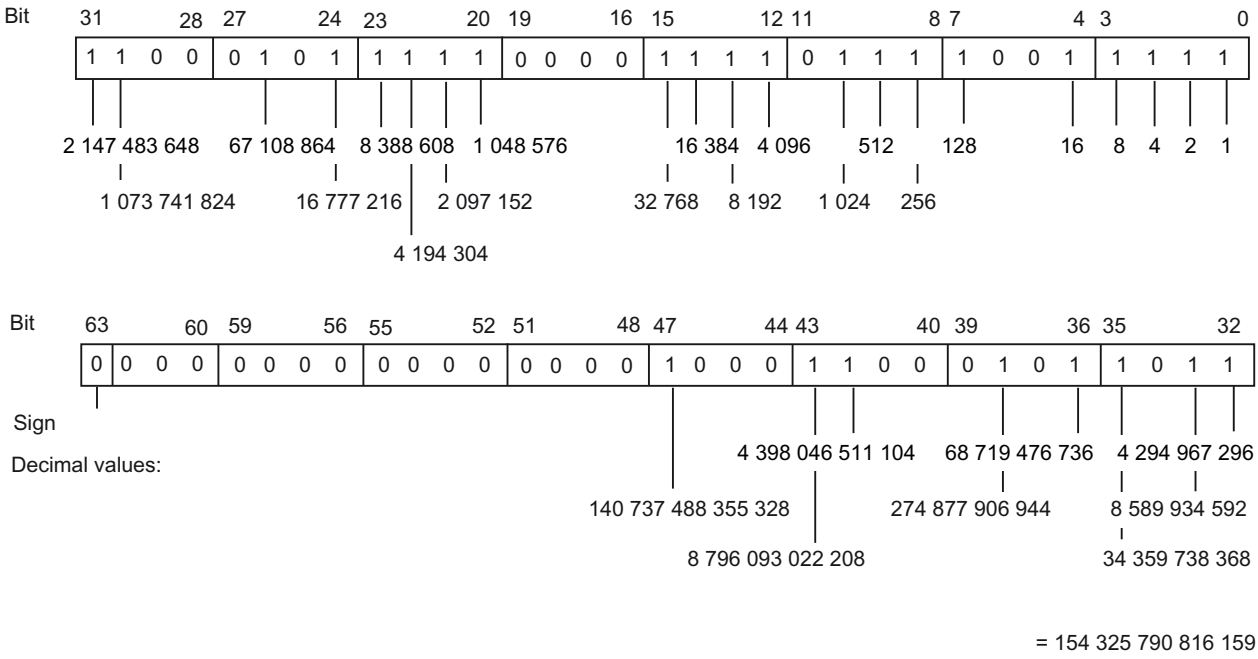

See also

Overview of the valid data types (Page 231)

Overview of data type conversion (Page 354)

Basics of constants (Page 97)

Implicit conversions (Page 356)

Explicit conversions (Page 405)

Data type conversion for S7-1200 (Page 489)

3.3.8 ULINT (64-bit integers) (S7-1500)

Description

An operand of data type ULINT (Unsigned Long INT) has a length of 64 bits and contains unsigned numerical values.

An operand of data type ULINT occupies eight BYTE in the memory.

The following table shows the properties of data type ULINT:

Length (bits)	Format	Value range	Examples of value input
64	Unsigned integers (decimal system)	0 to 18_446_744_073_709_551_615	• 154_325_790_816_159 • ULINT#154_325_790_816_159 • ULINT#10#154_325_790_816_159
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111	• 2#0000_0000_0000_0000_1000_1100_0101_1011_1100_0101_1111_0000_1111_0111_1001_1111 • ULINT#2#0000_0000_0000_0000_1000_1100_0101_1011_1100_0101_1111_0000_1111_0111_1001_1111 • ULINT#2#10
	Octal numbers	8#0 to 8#17_7777_7777_7777_7777_7777	• 8#4305_5705_7417_3637 • ULINT#8#4305_5705_7417_3637
	Hexadecimal numbers	16#0 to 16#FFFF_FFFF_FFFF_FFFF	• 16#0000_8C5B_C5F0_F79F • ULINT#16#0000_8C5B_C5F0_F79F

Example

The following figure shows the integer 154325790816159 as a binary number:

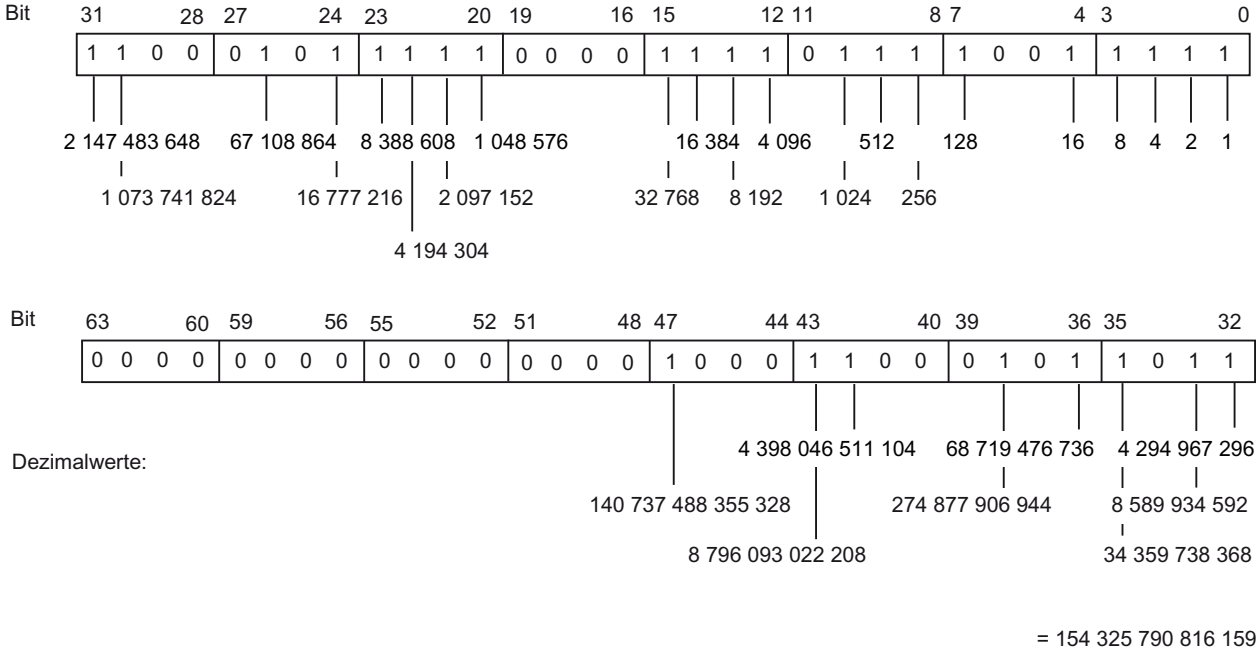

See also

Overview of the valid data types (Page 231)
Overview of data type conversion (Page 354)
Basics of constants (Page 97)
Implicit conversions (Page 356)
Explicit conversions (Page 405)
Data type conversion for S7-1200 (Page 489)

3.4 Floating-point numbers

3.4.1 REAL

Description

Operands of the data type REAL have a length of 32 bits and are used to represent floating-point numbers. An operand of the REAL data type consists of the following three components:

- Sign: The sign is determined by the signal state of bit 31. The bit 31 assume the value "0" (positive) or "1" (negative).
- 8-bit exponents to basis 2: The exponent is increased by a constant (base, +127), so that it has a value range of 0 to 255.
- 23-bit mantissa: Only the fraction part of the mantissa is shown. The integer part of the mantissa is always 1 with normalized floating-point numbers and is not stored.

The REAL data type is processed with a precision of 6 digits.

The following figure shows the structure of the REAL data type:

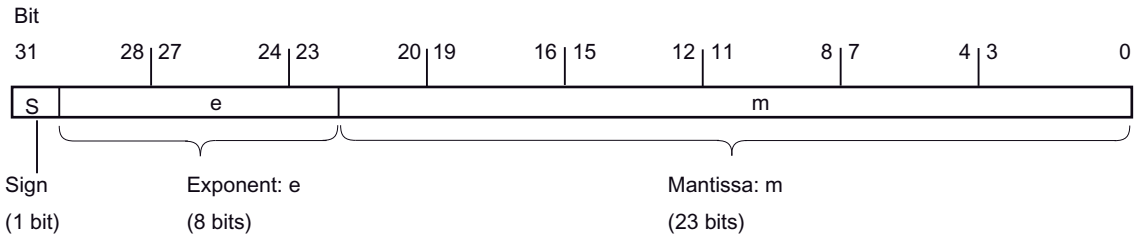

Note

With floating-point numbers, only the precision defined by the IEEE754 standard is stored. Additionally specified decimals are rounded off according to IEEE754.

The number of decimal places may decrease for frequently nested arithmetic calculations.

If more decimal places are specified than can be stored by the data type, the number is rounded to the value corresponding to the precision allowed by this value range.

The following table shows the properties of data type REAL:

Length (bits)	Format	Value range	Examples of value input
32	Floating-point numbers according to IEEE754	-3.402823e+38 to -1.175495e-38 ±0.0 +1.175495e-38 to +3.402823e+38	1.0e-5; REAL#1.0e-5
	Floating-point numbers		1.0; REAL#1.0

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Calculating with floating-point numbers (REAL and LREAL) in SCL (Page 7308)

Data type conversion for S7-1200 (Page 489)

3.4.2 LREAL (S7-1200, S7-1500)

Description

Operands of the data type LREAL have a length of 64 bits and are used to represent floating-point numbers. An operand of the LREAL data type consists of the following three components:

- Sign: The sign is determined by the signal state of bit 63. The bit 63 assumes the value "0" (positive) or "1" (negative).
- 11-bit exponents to base 2: The exponent is increased by a constant (base, +1023), so that it has a value range of 0 to 2047.
- 52-bit mantissa: Only the fraction part of the mantissa is shown. The integer part of the mantissa is always 1 with normalized floating-point numbers and is not stored.

The LREAL data type is processed with a precision of 15 digits.

The following figure shows the structure of the LREAL data type:

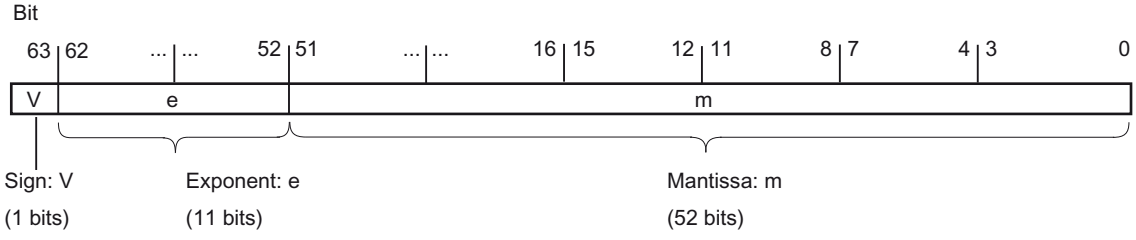

The following table shows the properties of data type LREAL:

Length (bits)	Format	Value range	Examples of value input
64	Floating-point numbers according to IEEE754	-1.7976931348623157e+308 to -2.2250738585072014e-308 ±0.0 +2.2250738585072014e-308 to +1.7976931348623157e+308	1.0e-5; LREAL#1.0e-5
	Floating-point numbers		1.0; LREAL#1.0

Note

With floating-point numbers, only the precision defined by the IEEE754 standard is stored. Additionally specified decimals are rounded off according to IEEE754.

The number of decimal places may decrease for frequently nested arithmetic calculations.

If more decimal places are specified than can be stored by the data type, the number is rounded to the corresponding value of the precision allowed by this value range .

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Calculating with floating-point numbers (REAL and LREAL) in SCL (Page 7308)

Data type conversion for S7-1200 (Page 489)

3.4.3 Invalid floating-point numbers

Description

We distinguish between four number ranges for data types REAL and LREAL:

- normalized numbers stored with full accuracy
- denormalized numbers not stored with full accuracy
- Infinite numbers: +Inf/-Inf (infinity)
- Invalid numbers: NaN (Not a Number)

Note

Floating-point numbers are stored as specified by the IEEE754 standard. Results of conversion or arithmetic functions with a denormalized, infinite or NaN (Not a Number) floating point depend on the CPU.

If you are not working with normalized floating-point numbers in mathematical functions, the result will show significant differences depending on the series of the CPU which you are using.

A CPU cannot calculate with denormalized floating-point numbers, with the exception of older CPU versions of the S7-300 and S7-400 series. The bit pattern of a denormalized number is interpreted as a zero. If the result of calculation falls into this range, it is continued with zero; the status bits OV and OS are set (number range undershoot).

Even though the values of invalid floating-point numbers can only be displayed with limited accuracy for mathematical functions, numbers with an exponent of -39 (e.g. 2.4408e-039) can

be monitored in the TIA Portal and therefore do not necessarily represent a faulty result. This means that floating-point values may be located outside the range of valid numerical values.

Note

The following applies to CPUs of the series S7-1200 V1, V2 and V3:

The comparison operation "Equal" uses the bit pattern of the invalid floating-point number. If two "NaN numbers" with the same bit pattern are compared, the output of the "Equal" comparison operation returns the result TRUE.

Note

The following applies to CPUs of the S7-1200 V4 and S7-1500 series:

If two invalid numbers (NaN) are compared with each other, the result is always FALSE, regardless of the bit pattern of the invalid number or the relation (>, >, ...).

Note

Comparison of denormalized floating-point numbers

For the comparison operation "Equal" with two denormalized floating-point numbers, the output for CPUs of the S7-300/400 series is set to the signal state "0" and for CPUs of the S7-1200/1500 series to the signal state "1".

If the input variables of a mathematical function represent an invalid floating-point number, an invalid floating-point number will also be output as result.

You have the following options to evaluate possible errors caused by invalid floating-point numbers:

- In LAD/FBD and SCL, you can query the enable output ENO for FALSE
- In STL, you can evaluate the status bit OV

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Calculating with floating-point numbers (REAL and LREAL) in SCL (Page 7308)

Data type conversion for S7-1200 (Page 489)

3.5 Timers

3.5.1 S5TIME (duration) (S7-300, S7-400)

Format

Data type S5TIME stores the duration in BCD format. The duration is the product from a time in the range 0 to 999 and a time basis. The time basis indicates the interval at which a timer decrements the time value by one unit until it reaches "0". The resolution of the times can be controlled via the time basis.

The following table shows the range of values for data type S5TIME:

Length (bits)	Format	Range of values	Examples of value input
16	S7 time in increments of 10 ms (default value)	S5T#0MS to S5T#2H_46M_30S_0MS	S5T#10s, S5TIME#10s

The following table shows the time base coding for S5TIME:

Time basis	Binary code for time basis
10 ms	00
100 ms	01
1 s	10
10 s	11

Always observe range limits and the resolution of time values when using data type S5TIME with timers. The table below specifies the range associated with each resolution:

Resolution	Range
0.01 s	10 ms to 9 s 990 ms
0.1 s	100 ms to 1 min 39 s 900 ms
1 s	1 s to 16 min 39 s
10 s	10 s to 2 h 46 min 30 s

Values that exceed 2h46m30s are not accepted.

Example

The following figure shows the content of the time operand for a time value of 127 and a time base of 1 s:

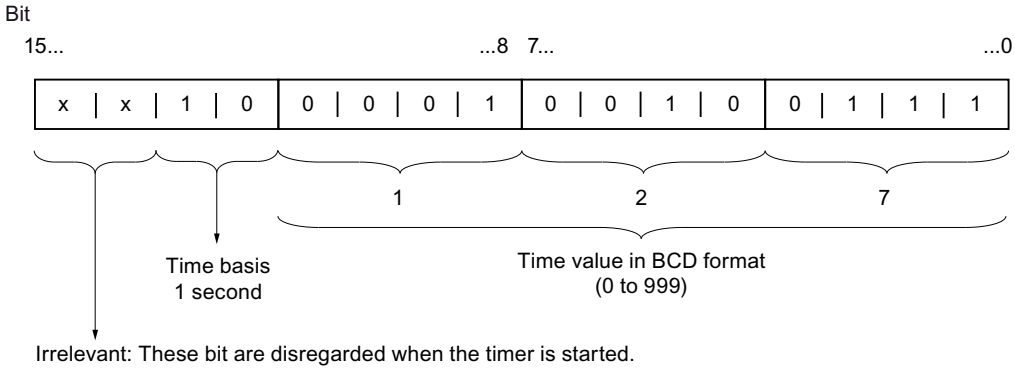

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.5.2 TIME (IEC time)

Description

The contents of an operand of the data type TIME is interpreted as milliseconds. The representation contains information for days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms).

The following table shows the properties of data type TIME:

Length (bits)	Format	Value range	Examples of value input
32	Signed duration	T#-24d_20h_31m_23s_648ms to T# +24d_20h_31m_23s_647ms	T#10d_20h_30m_20s_630ms, TIME#10d_20h_30m_20s_630ms

It is not necessary to specify all time units. T#5h10s is a valid entry, for example. If only one unit is specified, the absolute value of days, hours, and minutes must not exceed the high or low limits. When more than one time unit is specified, the value must not exceed 24 days, 23 hours, 59 minutes, 59 seconds or 999 milliseconds.

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.5.3 LTIME (IEC time) (S7-1500)

Description

The contents of an operand of data type LTIME is interpreted as nanoseconds. The representation contains information for days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms), microseconds (us) and nanoseconds (ns).

The following table shows the properties of data type LTIME:

Length (bits)	Format	Value range	Examples of value input
64	Signed duration	LT#-106751d_23h_47m_16s_854ms_775us_808ns to LT#+106751d_23h_47m_16s_854ms_775us_807ns	LT#11350d_20h_25m_14s_830ms_652us_315ns, LTIME#11350d_20h_25m_14s_830ms_652us_315ns

It is not necessary to specify all time units. LT#5h10s is therefore a valid entry, for example. If only one unit is specified, the absolute value of days, hours, and minutes must not exceed the high or low limits. When more than one time unit is specified, the value must not exceed 106751 days, 23 hours, 59 minutes, 59 seconds, 999 milliseconds, 999 microseconds or 999 nanoseconds.

See also

Overview of the valid data types (Page 231)

Overview of data type conversion (Page 354)

Basics of constants (Page 97)

Implicit conversions (Page 356)

Explicit conversions (Page 405)

Data type conversion for S7-1200 (Page 489)

3.6 Date and time

3.6.1 DATE

Format

The DATE data type saves the date as an unsigned integer. The representation contains the year, the month, and the day.

The contents of an operand of DATE data type correspond in hexadecimal format to the number of days since 01-01-1990 (16#0000).

The following table shows the properties of data type DATE:

Length (bytes)	Format	Value range	Examples of value input
2	IEC date (Year-Month-Day)	D#1990-01-01 to D#2169-06-06	D#2009-12-31, DATE#2009-12-31

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.6.2 TIME_OF_DAY (TOD)

Format

Data type TOD (TIME_OF_DAY) occupies a double word and stores the number of milliseconds since the beginning of the day (0:00 h) as unsigned integer.

The following table shows the properties of data type TOD:

Length (bytes)	Format	Value range	Examples of value input
4	Time-of-day (hours:minutes:seconds.milliseconds)	TOD#00:00:00.000 to TOD#23:59:59.999	TOD#10:20:30.400, TIME_OF_DAY#10:20:30.400

You always need to specify the hours, minutes and seconds. The specification of milliseconds is optional.

See also

Overview of the valid data types (Page 231)
Basics of constants (Page 97)
Data type conversion for S7-1200 (Page 489)

3.6.3 LTOD (LTIME_OF_DAY) (S7-1500)

Format

Data type LTOD (LTIME_OF_DAY) occupies two double words and stores the number of nanoseconds since the beginning of the day (0:00 h) as unsigned integer.

The following table shows the properties of data type LTOD:

Length (bytes)	Format	Value range	Examples of value input
8	Time-of-day (hours:minutes:seconds.nanoseconds)	LTOD#00:00:00.000000000 to LTOD#23:59:59.999999999	LTOD#10:20:30.400_365_215, LTIME_OF_DAY#10:20:30.400_365_215

You always need to specify the hours, minutes and seconds. The specification of nanoseconds is optional.

See also

Overview of the valid data types (Page 231)
Overview of data type conversion (Page 354)
Basics of constants (Page 97)
Implicit conversions (Page 356)
Explicit conversions (Page 405)
Data type conversion for S7-1200 (Page 489)

3.6.4 DATE_AND_TIME (date and time of day)

Format

The DT (DATE_AND_TIME) data type saves the information on date and time of day in BCD format.

The following table shows the properties of data type DT:

Length (bytes)	Format	Value range	Example of value input
8	Date and time (year-month-day-hour:minute:second:millisecond [3])	Min.: DT#1990-01-01-00:00:00.000 Max.: DT#2089-12-31-23:59:59.999	DT#2008-10-25-08:12:34.567, DATE_AND_TIME#2008-10-25-08:12:34.567

The following table shows the structure of the DT data type:

Byte	Contents	Value range
0	Year	0 to 99 (Years 1990 to 2089) BCD#90 = 1990 ... BCD#0 = 2000 ... BCD#89 = 2089
1	Month	BCD#1 to BCD#12
2	Day	BCD#1 to BCD# 31
3	Hour	BCD#0 to BCD#23
4	Minute	BCD#0 to BCD#59
5	Second	BCD#0 to BCD#59
6	The two most significant digits of MSEC	BCD#0 to BCD#999
7 (4MSB) [1]	The least significant digit of MSEC	BCD#0 to BCD#9
7 (4LSB) [2]	Weekday	BCD#1 to BCD#7 BCD#1 = Sunday ... BCD#7 = Saturday

[1] MSB: Most significant bit
[2] LSB: Least significant bit
[3] Fixed point number

See also

Overview of the valid data types (Page 231)
Basics of constants (Page 97)
Data type conversion for S7-1200 (Page 489)

3.6.5 LDT (DATE_AND_LTIME) (S7-1500)

Format

Data type LDT (DATE_AND_LTIME) stores the date and time-of-day information in nanoseconds since 01/01/1970 0:0.

The following table shows the properties of data type LDT:

Length (bytes)	Format	Value range	Example of value input
8	Date and time (Year-Month-Day-Hour:Minute:Second.Nanoseconds)	Min.: LDT#1970-01-01-00:00:00.000000000 Max.: LDT#2262-04-11-23:47:16.854775807	LDT#2008-10-25-08:12:34.567

See also

Overview of the valid data types (Page 231)

Overview of data type conversion (Page 354)

Basics of constants (Page 97)

Implicit conversions (Page 356)

Explicit conversions (Page 405)

Data type conversion for S7-1200 (Page 489)

3.6.6 DTL (S7-1200, S7-1500)

Description

An operand of data type DTL has a length of 12 bytes and stores date and time information in a predefined structure.

The following table shows the properties of data type DTL:

Length (bytes)	Format	Value range	Example of value input
12	Date and time (Year-Month-Day-Hour:Minute:Second.Nanoseconds)	Min.: DTL#1970-01-01-00:00:00.0 Max.: DTL#2262-04-11-23:47:16.854775807	DTL#2008-12-16-20:30:20.250

The structure of data type DTL consists of several components each of which can contain a different data type and range of values. The data type of a specified value must match the data type of the corresponding components.

Note

Invalid monitor value of DTL tags in hexadecimal format

If the monitor value of the DTL tags is represented in hexadecimal format, this can be because one of the values (YEAR, MONTH, DAY, etc.) is invalid. For example, this is the case if a value > 24 was specified at the HOUR tag.

The following table shows the structure components of data type DTL and their properties:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2262
1			
2	Month	USINT	1 to 12
3	Day	USINT	1 to 31
4	Weekday	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanosecond	UDINT	0 to 999999999
9			
10			
11			

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.7 Character strings

3.7.1 Character

3.7.1.1 CHAR

Description

A tag of the CHAR (Character) data type has a length of 8 bits and occupies one BYTE of memory.

The CHAR data type stores a single character in ASCII coding. You can find information on the encoding of special characters under "See also > STRING".

The following table shows the value range of the CHAR data type:

Length (bits)	Format	Value range	Example of value inputs
8	ASCII characters	ASCII character set	'A', CHAR#'A'

See also

Overview of the valid data types (Page 231)

STRING (Page 264)

3.7.1.2 WCHAR (S7-1200, S7-1500)

Description

A tag of the data type WCHAR (Wide Characters) has a length of 16 bits and occupies two BYTE of memory.

The WCHAR data type saves a single character of an expanded character set in UFT-16 coding. However, only a subset of the entire Unicode range is covered. Characters that cannot be displayed are made displayable using an escape character.

The following table shows the value range of the WCHAR data type:

Length (bits)	Format	Value range	Example of value input
16	Unicode	$0000 - $D7FF	WCHAR#'a'

See also

Overview of the valid data types (Page 231)

3.7.2 Character strings

3.7.2.1 STRING

Description

An operand of the STRING data type saves several characters in a character string that can consist of up to 254 characters. In a character string, all characters of the codepage created on the system are permitted. The characters are specified in single quotation marks.

A character string can also contain special characters. The escape character $ is used to identify control characters, dollar signs and single quotation marks.

Note

Different code pages

Please note that the special characters are coded using the code page currently set in Windows. This means that a string that contains special characters can be displayed differently on a different operating system with a different code page.

The dependency of the codepage on the created system makes an international use of the user program more difficult. Only the characters from the 7-bit ASCII coding are internationally valid.

The following table shows the properties of a STRING tag:

Length (bytes)	Format	Value range	Example of value input
n + 2 [1]	ASCII character string incl. special characters	0 to 254 characters	• 'Name' • STRING#'NAME' • STRING#'Na... (The actual length of the string is longer than the space available on the screen.) • STRING#'' (The string is empty.)
[1] An operand of the STRING data type occupies two bytes more than the specified maximum length in the memory.			

The table below shows examples for the notation of special characters:

Character	Hex	Meaning	Example
$L or $l	0A	Line feed	'$LText', '$0AText'
$N	0A and 0D	Line break The line break occupies 2 characters in the character string and is converted to RL in the display in the editor.	'$NText', '$0A$0DText'
$P or $p	0C	Page feed	'$PText', '$0CText'
$R or $r	0D	Carriage return (CR)	'$RText','$0DText'

Character	Hex	Meaning	Example
$T or $t	09	Tab	'$TText', '$09Text'
$$	24	Dollar sign	'100$$', '100$24'
$'	27	Single quotation mark	'$'Text$','$27Text$27'

If the escape character $ is followed by a letter from the table, the character indicated in the table is entered in the string. If the escape character $ is followed by a letter that is not in the table, this letter is entered in the string. If the escape character $ is followed by two hexadecimal digits or only one digit, this code is entered in the string.

Use in the watch table

The following applies to a CPU of the S7-300/400 series: If a tag of the STRING data type is being monitored, only the first 30 characters will be displayed. If the actual length is greater than 30 characters, an ellipsis (…) is displayed instead of the final apostrophe ('). STRING values with more than 30 characters cannot be used for modifying.

Maximum length of a character string

The maximum length of the character string can be specified during the declaration of an operand using square brackets after the keyword STRING (for example, STRING[4]). You can also use local or global constants to declare the maximum length (for example, STRING[#loc_const] or STRING["glob_const"]). If the specification of the maximum length is omitted, the standard length of 254 characters is set for the respective operand.

You can find additional information about using local or global constants to declare the maximum length here:

- Declaring the block interface: Declare variables of the STRING and WSTRING data types (Page 6981)
- Programming data blocks: Declaring tags of the STRING data type (Page 7033)
- Examples of using constants (Page 101)

If the actual length of a specified character string is shorter than the declared maximum length, the characters are written to the character string left-justified and the remaining character spaces remain undefined. Only occupied character spaces that define the actual length of the string are considered for the value processing and all displays.

Note

For S7-300/400 CPUs, please note: If a temporary tag of the STRING data type was defined, you must describe the BYTE "Max. length of string" with the defined length before you use the tags in the user program.

Transferring a STRING for parameter supply

You can transfer the STRING data type as a parameter. You can find additional information on parameter supply with STRING here:

- Transferring a tag of the STRING or WSTRING data type (Page 271)
- For S7-1200/1500: Valid data types in the block interface (Page 6969)
- For S7-300/400: Valid data types in the block interface (Page 6968)

Example

The example below shows the byte sequence if the STRING[4] data type is specified with output value 'QB':

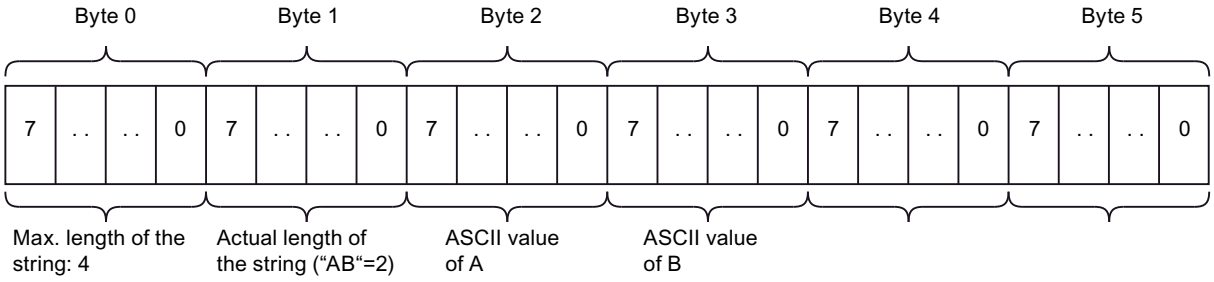

See also

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

Data type conversion for S7-1200 (Page 489)

3.7.2.2 Structure of a STRING tag

Introduction

A tag with STRING data type (character string) is maximum 256 characters long with 254 bytes user data. It starts in a non-optimized block at a word limit (at a byte with even address). In an optimized block it can start at any byte limit.

When tags are created their maximum length is specified. In the case of the preassignment or editing of the character string, the current length (the actually used length of the character string = number of valid characters) is entered. The maximum length is located in the first byte of the character string. The current length is in the second byte. This is followed by the characters, coded according to the codepage set in Windows.

Structure of a STRING tag

		Data type	Range	
Byte n1)	max. length	USINT	0 to 254	(k)
Byte n+1	Current length	USINT	0 to 254	(m, m <= k)
Byte n+2	1st character	CHAR	Current length (m)	maximum Length (k)
Byte n+3	2nd character	CHAR		
Byte ...	...	CHAR		
Byte n+m+1	with sign	CHAR		
Byte ...	...	CHAR		
Byte n+k+1	...	CHAR		

1) n = straight

3.7.2.3 WSTRING (S7-1200, S7-1500)

Description

An operand of data type WSTRING (Wide Character String) stores several Unicode characters of data type WCHAR in one character string. If you do not specify a length, the character string has a preset length of 254 characters. In a character string, all characters that are supported by the operating system are permitted. This means you can also use Chinese characters in a character string. Windows supports only some (but sufficient number) of the characters defined in Unicode.

Note

Coding

STEP 7 prohibits all coding of $D000 to $FFFF.

When declaring an operand of data type WSTRING you can define its length using square brackets (for example WSTRING[10]). If you do not specify a length, the length of the WSTRING is set to 254 characters by default. You can declare a length of up to 16382 characters (WSTRING[16382]).

The specification of the characters occurs in single quotes and always with the qualifier WSTRING#.

The following table shows the properties of a WSTRING tag:

Length (WORD)	Format	Value range	Example of value input
n + 2 [1]	Unicode character string; n specifies the length of the character string.	Preset value: 0 to 254 characters Max. possible value: 0 to 16382	• WSTRING#'Hello World' • WSTRING#'Hello Wo... (The actual length of the string is longer than the space available on the screen.) • WSTRING#'' (The string is empty.)
[1] An operand of the WSTRING data type occupies two WORDs more in the memory than the specified maximum length.			

A character string can also contain special characters. The escape character $ is used to identify control characters, dollar signs and single quotation marks.

The table below shows examples for the notation of special characters:

Character	Hex	Meaning	Example
$L or $l	000A	Line feed	'$LText', '$000AText'
$N	000A and 000D	Line break The line break occupies 2 characters in the character string and is converted to RL in the display in the editor.	'$NText', '$000A $000DText'
$P or $p	000C	Page feed	'$PText', '$000CText'
$R or $r	000D	Carriage return (CR)	'$RText','$000DText'
$T or $t	0009	Tab	'$TText', '$0009Text'
$$	0024	Dollar sign	'100$$', '100$0024'
$'	0027	Single quotation mark	'$'Text$'','$0027Text $0027'

If the escape character $ is followed by a letter from the table, the character indicated in the table is entered in the string. If the escape character $ is followed by a letter that is not in the table, this letter is entered in the string. If the escape character $ is followed by four hexadecimal digits, this code is entered in the string.

Note

Conversion of WSTRING tags

Implicit conversion of the WSTRING data type is not possible. Explicit conversion of the WSTRING data type to STRING is generally possible. However, as standard, it is only possible to convert characters in the code range from 0 - 127 in all Windows code pages. For all characters outside this range, the code page character and the lower byte of the Unicode character must be in exactly the same position for the conversion to work without errors.

Use in the watch table

If a tag of the WSTRING data type is being monitored, only the first 254 characters are displayed. If the actual length is greater than 254 characters, an ellipsis (…) is displayed instead of the closing apostrophe ('). WSTRING values with more than 254 characters cannot be used for modifying.

Maximum length of a character string

The maximum length of the character string can be specified during the declaration of an operand using square brackets after the keyword WSTRING (for example, WSTRING[4]). You can also use local or global constants to declare the maximum length (for example, WSTRING[#loc_const] or WSTRING["glob_const"]). If the specification of the maximum length is omitted, the standard length of 254 characters is set for the respective operand.

You can find additional information about using local or global constants to declare the maximum length here:

- Declaring the block interface: Declare variables of the STRING and WSTRING data types (Page 6981)
- Programming data blocks: Declaring tags of the STRING data type (Page 7033)
- Examples of using constants (Page 101)

If the actual length of a specified character string is shorter than the declared maximum length, the characters are written to the character string left-justified and the remaining character spaces remain undefined. Only occupied character spaces are considered in the value processing.

Transferring a WSTRING for parameter supply

Operands of the data type WSTRING can be transferred as parameters up to the maximum length for blocks with "optimized" access.

For function blocks (FB) with "standard" access, operands of the data type WSTRING can be declared as parameters in all sections of the block interface except in the section "InOut". For a function (FC) with "standard" access, only operands of the WSTRING data type can be transferred as parameters.

The function value of an FC in the "Return" section and expressions in the SCL programming language are another exception to this rule. In these cases, the WSTRING tag must not be longer than 1022 characters.

The declared lengths of formal and actual parameters may be different. You can find additional information on parameter supply with WSTRING here:

- Transferring a tag of the STRING or WSTRING data type (Page 271)
- Valid data types in the block interface (Page 6969)

Example

The example below shows the byte sequence if the WSTRING[4] data type is specified with output value 'QB':

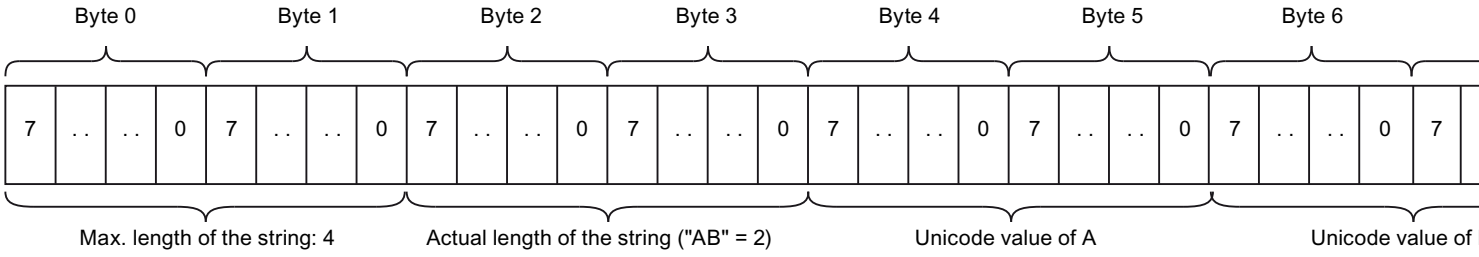

See also

Overview of the valid data types (Page 231)

Constants (Page 97)

3.7.2.4 Addressing individual characters of a STRING or WSTRING (S7-1200, S7-1500)

Addressing individual characters of a STRING or WSTRING

Use the syntax `StringName[i]` to access an individual character of a STRING or WSTRING tag. The counting index "i" begins with "1". Thus, you access the first character of the string with StringName[1].

You cannot access individual characters of a STRING or WSTRING constant.

Examples

The following examples show the addressing of (W)STRINGs:

Addressing	Explanation
"myDB".mySTRING[3]	Addresses the third character of the STRING in the data block.
#myWSTRING[3]	Addresses the third character of the WSTRING.

Troubleshooting W(STRING) access

Access errors result during runtime when you access a character that is located outside the STRING length. On read access to the character string, you receive the character '$00' or '$0000'; write access to the character string is not executed. If the instruction has the enable output ENO, ENO is set to the signal state FALSE. The CPU does not change to STOP.

The only exception is when the character is written directly after the actual length of the character string.

The following example shows the string 'Hello' with the actual length of 5. The 27th character of the STRING lies outside the actual length and can therefore not be written. The STRING remains unchanged; the result of the assignment is 'hello'.

SCL

```
MyDB.mystring := 'hello';
MyDB.mystring[27] := CHAR_TO_BYTE('!');
```

The following example shows the specified exception: The character is written directly after the STRING in the 6th character. The result of the assignment is 'hello!'.

SCL

```
MyDB.mystring := 'hello';
MyDB.mystring[6] := CHAR_TO_BYTE('!');
```

Where possible, use instructions from the "Extended instructions > String + Char" pane to handle STRINGs.

```
CONCAT(IN1 := 'hello', IN2 := '!');
```

3.7.2.5 Transferring a tag of the STRING or WSTRING data type

Description

You can transfer tags of the STRING or WSTRING data type as parameters. The following table shows the rules that apply to the (W)STRING transfer in the various CPU families:

CPU family	Data type	Rules for transfer at the block call
S7-300/400	STRING	The declared lengths of formal and actual parameters must be identical.
S7-1200/1500	STRING WSTRING	The declared lengths of formal and actual parameters may be different. If the declared length of the target parameter during runtime is insufficient to accept the (W)STRING, the (W)STRING is truncated and the enable output ENO is set to FALSE. In the programming editor, a gray rectangle at the parameter indicates that (W)STRING might be truncated during runtime. Exception: When calling STL blocks, the declared lengths of formal and actual parameters must always be identical.

The following figure shows a block call in which the lengths of the declared formal and actual parameters differ. Due to the different declared lengths, both "Input_String_20" and "Output_String_10" could be truncated during runtime.

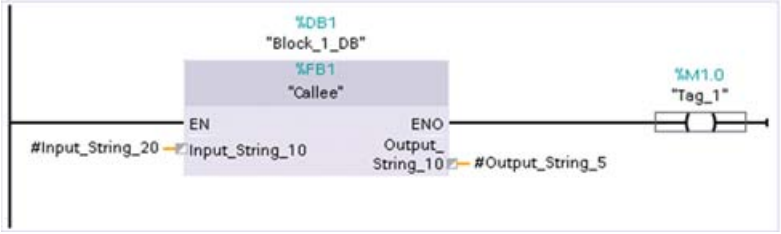

See also

Rules for supplying block parameters (Page 76)

3.8 PLC data types (UDT)

3.8.1 Basics of PLC data types (UDT)

Description

A PLC data type (UDT) is a complex, user-defined data type that is used for the declaration of a tag. It represents a data structure made up of several components of different data types. The components can also be derived from another PLC data type, from an ARRAY or directly using the STRUCT keyword as a structure. The nesting depth is hereby limited to 8 levels.

You can change a PLC data type (UDT) centrally and use it repeatedly in the program code. All locations of use are updated automatically.

Advantages of the PLC data types:

- Simple data exchange via block interfaces among several blocks
- Group data according to the process control
- Transferring a parameter as a data unit

Using a PLC data type

You can declare PLC data types as a type when creating data blocks. Based on this type, you can create a number of data blocks, all of which have the same data structure. These data blocks can be adapted by entering different actual values for the corresponding task.

For instance, create a PLC data type for a recipe for blending paints. You can then assign this data type to several data blocks, each of which contains different quantity information.

The following figure shows this application:

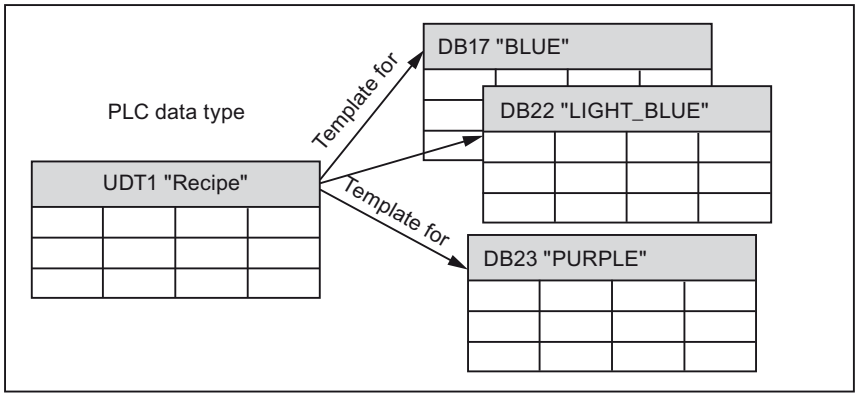

PLC data types can be used for the following applications:

- PLC data types can be used as data types for tags in the tag declaration of logic blocks or in data blocks.
- PLC data types can be used as templates for the creation of global data blocks with identical data structures.
- PLC data types can be used in S7-1200 and S7-1500 as a template for the creation of structured PLC tags.

Nesting depth and potential number of PLC data types

A maximum of 64K objects of FCs, FBs, DBs and PLC data types are defined (the total number is restricted accordingly for CPUs, however).

A maximum of 16K components can be defined per hierarchy level for each structure description (for example, a DB or a PLC data type (UDT)). These components can also be an ARRAY or a STRUCT data type. When a structured component is hereby derived from an explicitly declared PLC data type, there is no restriction. In other words, 16K components of PLC data type (UDT)/SDT are supported for each hierarchy level. However, a maximum of 252 of the 16K components can have "anonymous structures".

PLC data types can be nested to a depth of 8 hierarchy levels. This nesting depth is independent of the CPU used.

Disadvantages of anonymous structures

The components of the structured tag can be identically addressed. This is regardless of whether the declaration has been made using a PLC data type or an anonymous structure.

The following disadvantages apply when using anonymous structures:

- Reusability of the same structure by copying. This makes it difficult to change the structure
- Anonymous structures are not compatible with PLC data types (UDT) of the same structure
- Performance, since the matching types for structural components are checked during runtime
- The high limit is more easily reached when using anonymous structures, since all components must be evaluated individually.

Example

The following example shows the data type definition of the "MyUDT" PLC data type:

myUDT

		Name	Datentyp
1		FirstEntry	Struct
2		Value1	Int
3		Value2	Real
4		Value3	"myOtherUDT"
5		Station1	Int
6		Station2	UInt
7		Station3	DInt

PLC data types generated by the system

Some instructions generate their own PLC data types during instancing which are saved in the "PLC data types" project folder. Do not use these PLC data types created by the system in a library, because this may result in undesirable system behavior. If necessary, these PLC data types are created by the system, which means they must not be stored in a library.

Generating external sources from blocks

When generating external sources from blocks, the changes made directly in the block interface to the default values of PLC data types are not exported to the sources. This means that these values are not available when the sources are imported once again. The default values are applied instead. To prevent this loss of data for the modified default values, the changes must be made directly in the PLC data type and not in the block interface. In this case, the changes are also exported when external sources are generated.

Additional information on the PLC data types (UDTs)

Note

Declaring PLC data types (UDTs):

Structure of the declaration table for PLC data types (Page 7105)

Programming recommendations for PLC data types:

Using PLC data types (UDT) (Page 220)

Using the DB_ANY data type (Page 204)

Comparison of PLC data types (UDTs) in the program:

Comparing PLC data types (Page 7619)

Note

You can find more information on the PLC data types in the Siemens Industry Online Support in the following contributions:

How can you initialize structures in optimized memory areas at the S7-1500 in STEP 7 (TIA Portal)?

https://support.industry.siemens.com/cs/ww/de/view/78678760 (https://support.industry.siemens.com/cs/ww/en/view/78678760)

How do you create a PLC data type (UDT) at an S7-1500 control system?

https://support.industry.siemens.com/cs/ww/de/view/67599090 (https://support.industry.siemens.com/cs/ww/en/view/67599090)

How is the specific application of own PLC data types (UDT) effected in STEP 7 (TIA Portal)?

https://support.industry.siemens.com/cs/de/de/view/67582844 (https://support.industry.siemens.com/cs/de/en/view/67582844)

Why should complete structures be transferred at a block call for S7-1500 instead of many individual components?

https://support.industry.siemens.com/cs/ww/de/view/67585079 (https://support.industry.siemens.com/cs/ww/en/view/67585079)

See also

Addressing PLC data types (UDT) (Page 277)

Declaring tags based on a PLC data type (Page 6982)

Declaring tags based on a PLC data type (Page 7033)

Comparing PLC tags (Page 7588)

Creating structured PLC tags (Page 7086)

Permissible addresses and data types of PLC tags (Page 7073)

CMP ==: Equal (Page 687)

Transferring a tag of the PLC data type (UDT) (Page 276)

3.8.2 Transferring a tag of the PLC data type (UDT)

Description

You can also transfer tags that are declared as PLC data type (UDT) as actual parameters. If the formal parameter is declared as a PLC data type (UDT) in the tag declaration, you must transfer a tag that has the same PLC data type (UDT) as an actual parameter.

An element of a tag declared using the PLC data type (UDT) can also be transferred as an actual parameter for the block call, provided the data type of the element of the tag matches the data type of the formal parameter.

Structures and user-defined PLC data types can be assigned to one another if they have identical structures. This means that the data types and the order of all structure components must be identical. In nested structures, the data types and the order of the lower-level structures and their components must also match.

Two different PLC data types can also be assigned to one another if the data types and the order of all structure components including the lower-level structures are identical. The names of the PLC data types do not have to match.

User-defined PLC data types and structures cannot be assigned to system data types.

See also

Rules for supplying block parameters (Page 76)

3.8.3 Addressing PLC data types (UDT)

Addressing data elements of an PLC data type

The following syntax is used to access elements of a PLC data type:

```
<PLCDataTypeName>.<ElementName>
```

Examples

The following examples show the addressing of structured data type tags:

Addressing	Description
Values.Temperature	Addressing of the "Temperature" element in the "Values" tag, which is based on a PLC data type.

See also

Addressing variables in global data blocks (Page 106)

3.9 STRUCT data structure (anonymous structures)

3.9.1 Basic information on STRUCT

Description

The STRUCT data type represents a data structure that consists of a fixed number of components of different data types. Components of STRUCT or ARRAY data type can also be nested in a structure. Structures can be used to group data according to the process control system and to transfer parameters as one data unit.

Structure declarations that are used directly at the tag are called anonymous structures. An anonymous structure can have the following form:

Global_DB

	Name	Data type
1	Static	
2	Porsche	Struct
3	Motor	Struct
4	Valves	Array[0..5] of Bool
5	Speed_Min	Byte
6	Speed_Max	Byte
7	Temperature	Word
8	Axes	Array[0..4] of Bool
9	Oil	Byte
10	Carriage	Struct
11	Tuning	Struct
12	BMW	Struct
13	Audi	Struct

All of the subsequent explanations refer to this structure image.

Nesting depth and potential number of structures

A nested structure contains at least on additional structure as component. In each case 65535 FCs, FBs, DBs and PLC data types can be defined. For each CPU there is a special high limit relating to the block types and the total of all blocks.

A maximum of 16K components can be defined per hierarchy level for each structure description (for example, a DB or a PLC data type (UDT)). These components can also be an ARRAY or a STRUCT data type. When a structured component is hereby derived from an explicitly declared PLC data type, there is no restriction. For each hierarchy level, 16K components of PLC data type (UDT)/SDT are supported. However, a maximum of 252 "anonymous structures" may exist within a block.

Structures (STRUCT) can be nested to a depth of 8 hierarchy levels. This nesting depth is independent of the CPU used.

Transferring a STRUCT for parameter supply

You can transfer the STRUCT data type as a parameter. You can find additional information on parameter supply with STRUCT here:

Transferring tags of the STRUCT data type (Page 282)

Disadvantages of anonymous structures

The components of the structured tag can be identically addressed. This is regardless of whether the declaration has been made using a PLC data type or an anonymous structure.

The following disadvantages apply when using anonymous structures:

- Increased maintenance costs: If an anonymous structure was copied multiple times, then it must also have been changed multiple times during a change.
- Anonymous structures are not compatible with PLC data types (UDT) of the same structure
- Performance, since the matching types of all structural components are checked
- Increased memory requirement: Each anonymous structure is a separate object, whose description is loaded to the AS.

Example

If you declare the tag of the data type STRUCT in a PLC data type (UDT), more usage options are available to you (see figure on left). However, you can also declare the tag directly with the data type STRUCT (see figure on right).

Declaration of the structured tag "Motor" with or without PLC data type (UDT):

Structured tag with PLC data type (UDT)	Structured tag without PLC data type (UDT)

Global_DB

	Name	Data type
1	Static	
2	Porsche	"Vehicle"
3	Motor	Struct
4	Valves	Array[0..5] of Bool
5	Speed_Min	Byte
6	Speed_Max	Byte
7	Temperature	Word
8	Axes	Array[0..4] of Bool
9	Oil	Byte
10	Carriage	Struct
11	Tuning	Struct
12	BMW	"Vehicle"
13	Audi	"Vehicle"

Global_DB

	Name	Data type
1	Static	
2	Porsche	Struct
3	Motor	Struct
4	Valves	Array[0..5] of Bool
5	Speed_Min	Byte
6	Speed_Max	Byte
7	Temperature	Word
8	Axes	Array[0..4] of Bool
9	Oil	Byte
10	Carriage	Struct
11	Tuning	Struct
12	BMW	Struct
13	Audi	Struct

Processing of complex structures

A series of instructions (e.g. "Serialize: Serialization", "Deserialize: Deserialization", "CMP" (comparator) and "MOVE: Move value") can process very large, complex structured tags. In doing so, the CPU analyzes the form of the tag structure and executes the corresponding instruction for each substructure contained in the total structure or for all contained elementary components.

With a very complex structure, this structure analysis may lead to an unexpected increase in the run time of the corresponding instruction. In addition to the complexity of structured tags specified in the operation, the total number of anonymous structures declared in the program also has an effect on the run time. A very large number of different anonymous structure definitions can also increase the run time.

Solution:

- Avoid anonymous structures Use PLC data types instead.
- Avoid multiple declaration of data structures that are structured very similarly. Try to assemble these into a structure declaration.
- Avoid the declaration of numerous individual tags in structures and data blocks, if they have the same data type. Use the ARRAY data type in this case, if possible.

See also

Overview of the valid data types (Page 231)

3.9.2 Structure of a STRUCT tag

Introduction

A STRUCT tag always starts in a non-optimized block at a word limit, i.e. at a byte with even address. The individual components are then located in the order of their declaration in the memory. STRUCT tags occupy the memory up to the next word limit.

Components with BOOL data type start in the least significant bit. Components with BYTE and CHAR data type start in the next byte. Components with different data type start at a word limit. In optimized blocks the structures are stored so that all components are aligned according to their size. The order in the memory does not correspond to the display in the editor.

A nested structure is a structure that is a component of another structure. A nesting depth of maximum 8 structures is possible. A nesting depth of maximum 9 structures is possible in the "InOut" section. The nesting depth is calculated including the highest structure element (example: "Anna.Berta.Carla" corresponds to a nesting depth of 3).

All components can be separately addressed. The individual names of components are separated by a dot.

Structure of a STRUCT tag with elementary and complex data types in the non-optimized memory area:

- Declaration of the "Motor" STRUCT tag

Global_DB

	Name	Data type
1	Static	
2	Porsche	Struct
3	Motor	Struct
4	Valves	Array[0..5] of Bool
5	Speed_Min	Byte
6	Speed_Max	Byte
7	Temperature	Word
8	Axes	Array[0..4] of Bool
9	Oil	Byte
10	Carriage	Struct
11	Tuning	Struct
12	BMW	Struct
13	Audi	Struct

- Required memory of the "Motor" STRUCT tag

	7	6	5	4	3	2	1	0	
Byte n[1]	...	...	5	4	3	2	1	0	Valves (bit components and padding bits)
Byte n+1	Byte 1								Speed_Min (byte component)
Byte n+2	Byte 2								Speed_Max (byte component)
Byte n+3	(Padding byte)								
Byte n+4	Word 1								Temperature (word component)
Byte n+5									
Byte n+6	...	...	...	4	3	2	1	0	Axes (bit components and padding bits)
Byte n+7	Byte 3								Oil (byte component)

[1] n = even

3.9.3 Addressing STRUCT components

Addressing components in structures

You access individual components of a structure using the following syntax:

```
"DBname".name1.name2.name3
```

Example

The following example shows the addressing of structured data type tags:

Addressing	Explanation
"Global_DB".Motor.Valves[2]	Addressing of the 3rd valve of the "Valves" element in the "Motor" structure of the "Global_DB" global data block.

3.9.4 Transferring tags of the STRUCT data type

Description

You can also transfer individual components of a STRUCT tag as actual parameters if the components correspond to the data type of the formal parameter.

You can also transfer complete structures as parameters. If a block has an input parameter of the STRUCT data type, you must transfer a STRUCT with identical structure as the actual parameter. This means that the names and data types of all structure components have to be identical.

Structures and user-defined PLC data types can be assigned to one another if they have identical structures. This means that the data types and the order of all structure components must be identical. In nested structures, the data types and the order of the lower-level structures and their components must also match.

Two different PLC data types can also be assigned to one another if the data types and the order of all structure components including the lower-level structures are identical. The names of the PLC data types do not have to match.

User-defined PLC data types and structures cannot be assigned to system data types.

Note

We recommend that you program structures as PLC data types (UDT). PLC data types (UDT) make programming easier, since they can be used repeatedly and changed centrally.

See also

Rules for supplying block parameters (Page 76)

3.10 ARRAY

3.10.1 Basic information on ARRAY

Description

A tag of the ARRAY data type represents a data structure that consists of a fixed number of components of the same data type. All data types except ARRAY are permitted for the components.

When you create an ARRAY tag the index limits are defined in square brackets and the data type is defined after the keyword "of". The ARRAY limits can be either defined fixed using integers or global and local constants or as formal parameters of a block, or defined variably using ARRAY[*]. The low limit must be smaller than or equal to the high limit. An ARRAY may contain up to six dimensions, the limits of which can be specified separated by a comma.

The structure of a one-dimensional tag of the ARRAY data type has, for example, the following form:

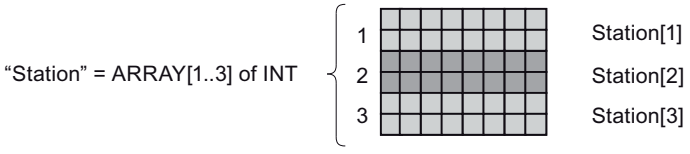

The maximum ARRAY limits depend on the following factors:

- Data type of the ARRAY component
- Memory reserve (only in blocks with optimized access)
  You can find additional information under: Basic information on loading block extensions without reinitialization (Page 7560).
- Maximum size of the data block
- Maximum memory capacity of the CPU (you can find more information in the manual for the device)

The following table shows the properties of the ARRAY data type:

Block property	Format	ARRAY limits	Data type
Standard	ARRAY[low limit..high limit] of <DataType>	[-32 768..32 767] of <DataType>	All data types except ARRAY
Optimized		[-2 147 483 648..2 147 483 647] of <DataType>	Multi-instances

Various options are available for defining the ARRAY limits depending on the ARRAY that you declared.

ARRAY limits	Global data block / tag table	Block interface
Using integers as fixed ARRAY limits	x	x
Using global constants as fixed ARRAY limits	x	x
Using local constants as fixed ARRAY limits	-	x
Using variable ARRAY limits ARRAY[*]	-	x
ARRAY of multi-instances	-	x

For more information on the section of the block interface in which you can define an ARRAY or an ARRAY[*], refer to: Valid data types in the block interface (Page 6969)

Note

Applies to CPUs of the S7-1500 series

For a block with the "Optimized block access" block property, an element of the BOOL data type requires 1 byte of memory. This is also true when you use an ARRAY of <data type>. An ARRAY[0..1] of BOOL, for example, thus requires 2 bytes in an optimized program block.

Using integers as fixed ARRAY limits

You can use integers as fixed ARRAY limits:

- Example of a one-dimensional ARRAY

	Static	
	ValueArray	Array[0..2] of Int
	ValueArray[0]	Int
	ValueArray[1]	Int
	ValueArray[2]	Int

- Example of a three-dimensional ARRAY:

	Static	
	ValueArray	Array[0..2, 0..2] of Int
	ValueArray[0,0]	Int
	ValueArray[0,1]	Int
	ValueArray[0,2]	Int
	ValueArray[1,0]	Int
	ValueArray[1,1]	Int
	ValueArray[1,2]	Int
	ValueArray[2,0]	Int
	ValueArray[2,1]	Int
	ValueArray[2,2]	Int

Using constants as fixed ARRAY limits

You can use local or global constants as fixed ARRAY limits:

- Fixed ARRAY limits for a one-dimensional ARRAY, consisting of two global user constants

Static	
ValueArray	Array["a"..."n"] of Int
ValueArray[0]	Int
ValueArray[1]	Int
ValueArray[2]	Int

- Fixed ARRAY limits for a one-dimensional ARRAY, consisting of two global user constants

Input		
ValueArray	Array[#b..#d] of Int	
ValueArray[1]	Int	0
ValueArray[2]	Int	0
ValueArray[3]	Int	0
ValueArray[4]	Int	0
ValueArray[5]	Int	0
Output		
InOut		
Static		
Temp		
Constant		
b	Int	1
d	Int	5

- Fixed ARRAY limits for a two-dimensional ARRAY, consisting of global and local user constants

Input		
ValueArray	Array["a"..#b, "n"..#d] of Int	
Output		
InOut		
Static		
Temp		
Constant		
b	Int	1
d	Int	5

For more information on creating and using constants as ARRAY limits, refer to:

- Declaring local tags and constants in the block interface (Page 6975)
- Examples of using constants (Page 101)

Using integers and global and local constants as fixed ARRAY limits

You can also combine all three above-named options with each other and also used integers and global and local constants mixed within an ARRAY as ARRAY limits.

Input		
▪ ▸ ValueArray	Array["a"..#b, 2..#d, 1.."n"] of Int	
▸ Output		
▸ InOut		
▸ Static		
▸ Temp		
▾ Constant		
▪ b	Int	1
▪ d	Int	5

Using variable ARRAY limits ARRAY[*] (only in blocks with optimized access)

You can use an ARRAY[*] to program blocks that can process an ARRAY with flexible length. To do this, you declare variable ARRAY limits for the parameters of a function or function block. ARRAY[*] is available in all programming languages.

Using the "LOWER_BOUND" and "UPPER_BOUND" instructions, you can read the low and high limits of an ARRAY with variable limits during runtime.

Note

Availability of ARRAY[*]

ARRAY[*] is available in optimized blocks for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. In functions (FC) you can use ARRAY[*] in all declaration subsections. In function blocks (FB) you can declare ARRAY[*] only as an in-out parameter in the "InOut" section.

- Variable ARRAY limits for a one-dimensional ARRAY:

▸ Input	
▸ Output	
▾ InOut	
▪ ▾ ValueArrayVariable	Array[*] of Int
▪ ValueArrayVariable[*]	Int
▸ Static	
▸ Temp	
▸ Constant	

- Variable ARRAY limits for a one-dimensional ARRAY:

▸ Input	
▸ Output	
▾ InOut	
▪ ▾ ValueArrayVariable	Array[*, *, *] of Int
▪ ValueArrayVariable[*, *, *]	Int
▸ Static	
▸ Temp	
▸ Constant	

You can find additional information on the use of ARRAY[*] here:

- Transferring tags of the ARRAY / ARRAY[*] data type (Page 291)
- Declaring tags of the ARRAY data type (Page 6978) in the block interface

Using multiple instances (only in blocks with optimized access)

You can also create multi-instances as ARRAY, in order to be able, for example, to use a variable index to address the individual multi-instances during the processing of program loops:

Static	
MultiinstanzArray	Array[0..1] of "SCL_FB_1500"
MultiinstanzArray[0]	"SCL_FB_1500"
Input	
Tag_In	Bool
PresetTime	Time
Output	
InOut	
Static	
MultiinstanzArray[1]	"SCL_FB_1500"

For more information on creating and using multi-instances, refer to:

- Multi-instances (Page 65)
- Declaring multi-instances (Page 6983)

See also

Overview of the valid data types (Page 231)

Indirect addressing of ARRAY components (Page 200)

3.10.2 Structure of an ARRAY tag

Introduction

In non-optimized blocks an ARRAY tag never starts at a word limit, i.e. at a byte with even address. ARRAY tags occupy the memory up to the next word limit.

In optimized block the ARRAY requires as much space as the widest element of the structure. This means, for example, that an ARRAY of BYTE is located at a Byte limit and an ARRAY of LREAL is located at an 8-byte limit.

Structure of an ARRAY tag in the one-dimensional array:

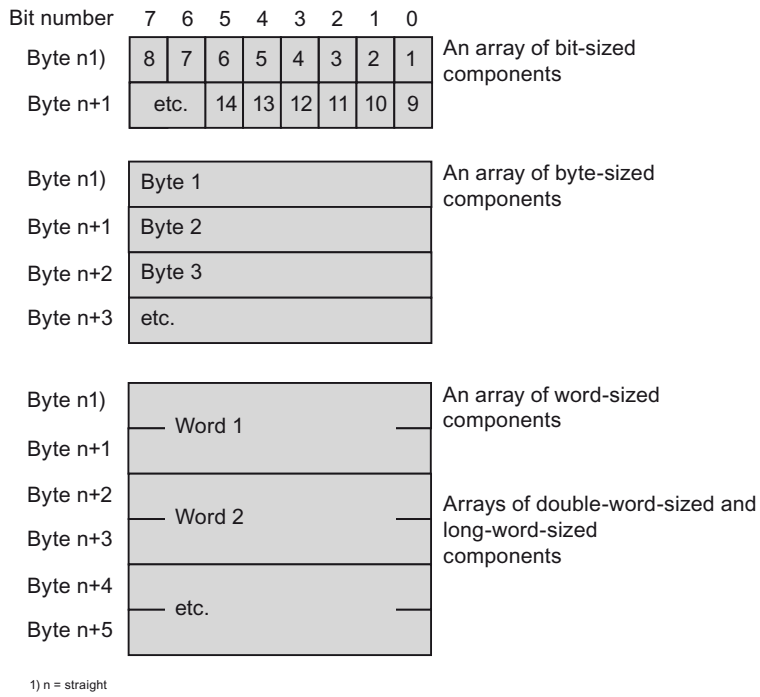

Components with BOOL data type start in the least significant bit. Components with BYTE and CHAR data type start in the right byte. The individual components are listed in order.

Structure of an ARRAY tag in the multidimensional array:

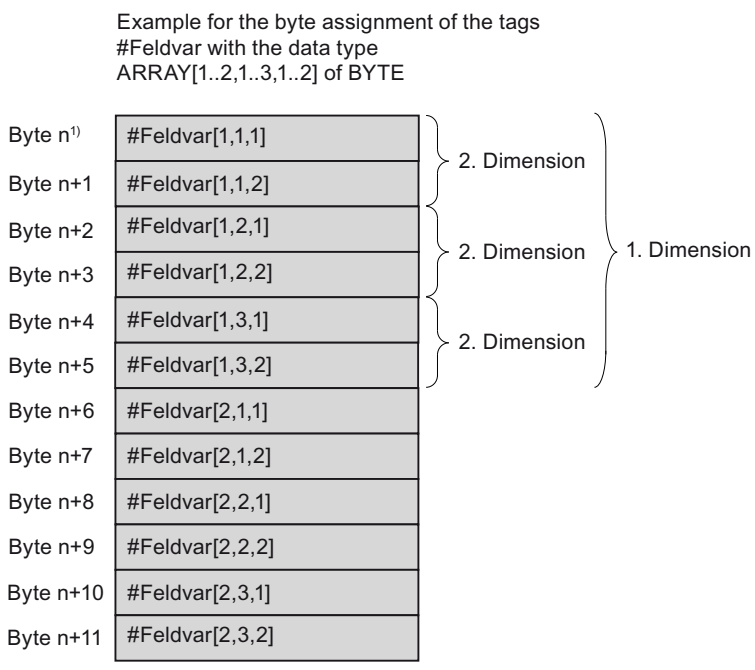

In multidimensional arrays, the components are stored line-by-line (dimension-by-dimension), starting with the first dimension. In bit and byte components a new dimension always starts at

the next byte. In components of other data types a new dimension always starts in the next word (in the next even byte) in non-optimized blocks.

In optimized blocks each dimension requires as much space as the first dimension.

3.10.3 Addressing ARRAY components

Introduction

ARRAY components are addressed using fixed of variable index. The components of an ARRAY tag can be treated like tags of the same data type.

The index of the component is specified in square brackets. The index includes an integer value (fixed index) or a tag (variable index) for each ARRAY dimension.

Addressing of ARRAY components using fixed index

The addressing of an ARRAY component using fixed index has the following form:

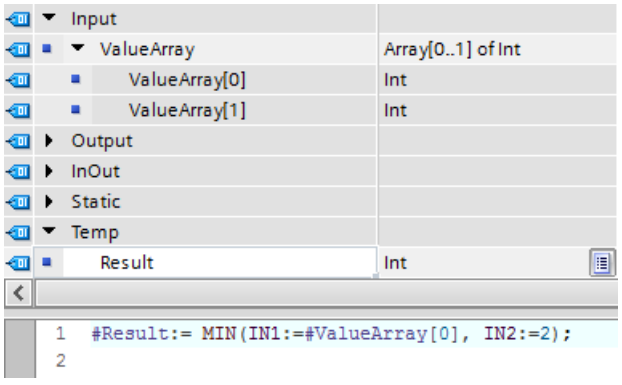

Addressing ARRAY components using variable index

An ARRAY component can also be addressed with a tag the value of which is only calculated during runtime. The tag can be an absolute or symbolic addressed global or local tag of integer data type. This addressing is also possible with multidimensional ARRAYs and with the addressing of sub-arrays. (<ArrayName>[i,j,k...])

A change of the tags in the called block has no effect on an ARRAY component that is created as actual parameter at an in/out parameter and addressed using a variable index. The value is written back to the same ARRAY component, transferred during the call, from which it was read.

The addressing of an ARRAY component using an index tag has the following form:

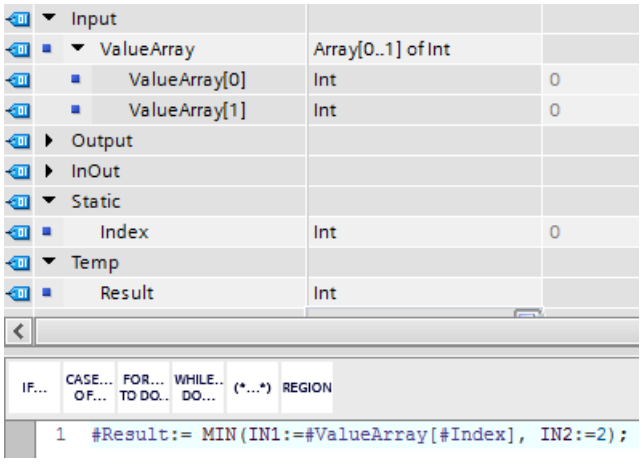

For more information on the indirect addressing of ARRAY components, refer to:

- Indirect indexing of ARRAY components (Page 121)
- Programming recommendations:Indirect addressing of ARRAY components (Page 200)

Error handling for ARRAY access

Access errors result when you access an element during runtime which is located outside the declared ARRAY limits. The various CPU families react differently to violations of the ARRAY limits:

- S7-300/400
  - The CPU changes to "STOP" mode.
  - You can program the program execution error OB (OB 85) to prevent this.
  - In SCL, you also have the option of enabling the attribute "Check ARRAY limits" in the block properties. This causes the enable output ENO to be set to FALSE in the case of ARRAY access errors.
- S7-1200
  - The CPU generates a diagnostic buffer entry and remains in "RUN" mode.
- S7-1500
  - The CPU changes to "STOP" mode.
  - You can program the programming error OB (OB 121) to prevent this.
  - You also have the option of programming the local error handling with the instructions "GET_ERROR: Get error locally" or "GET_ERROR_ID: Get error ID locally".

Note

Monitoring ARRAY access errors with ENO

The enable output ENO is not set to the signal state FALSE if the ARRAY limits are violated during execution of an instruction. The only exception is SCL blocks on CPUs of the S7-300/400 series for which the block property "Check ARRAY limits" is set.

3.10.4 Transferring tags of the ARRAY / ARRAY[*] data type

Transferring tags of the ARRAY data type

You can also transfer individual elements of an ARRAY as actual parameter if the element corresponds to the data type of the formal parameter.

You can transfer tags of the ARRAY data type as parameters. If a block has an input parameter of ARRAY data type, you must pass an ARRAY with identical structure as the actual parameter. This means that the data type, the number of dimensions and the number of field components must be identical.

ARRAYs can be assigned to one another if they have an identical structure. This means that the data type, the number of dimensions and the number of array components must be identical. The names of the ARRAYs do not have to match.

Transferring tags of the ARRAY data type[*]

You can use ARRAY[*] to declare ARRAYs with variable limits at the parameters of a function or function block. When you create the block you can define the limits of the ARRAY that will be transferred later during runtime after the call during runtime.

The following figure shows two calls of a block with an input parameter of the ARRAY[*] data type. ARRAYs of different length are transferred with each call.

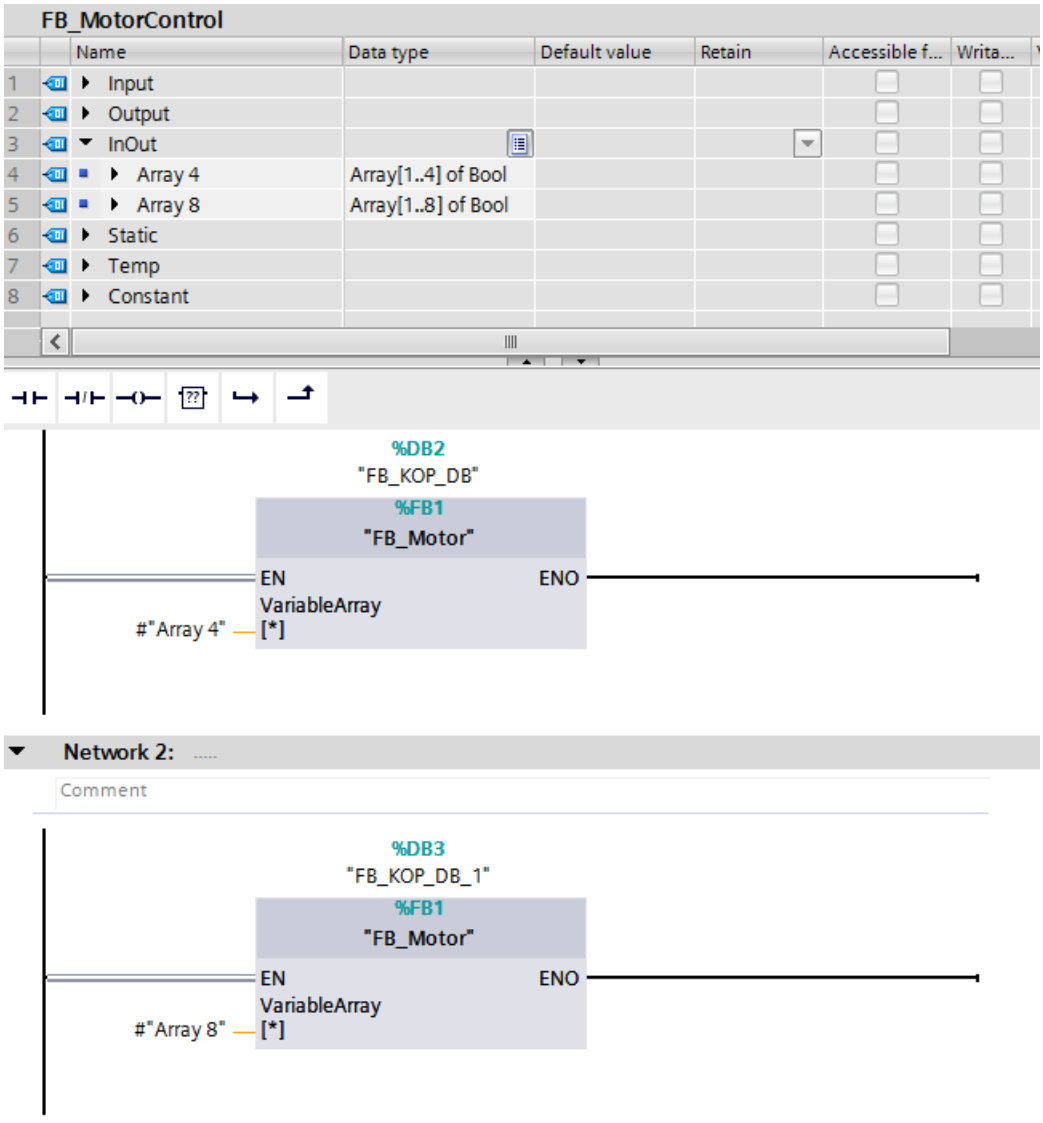

Applicable rules when transferring ARRAYs with flexible limits

Create the function "BlockWithArrayStarIn_FC" for the example so that it can be called later:

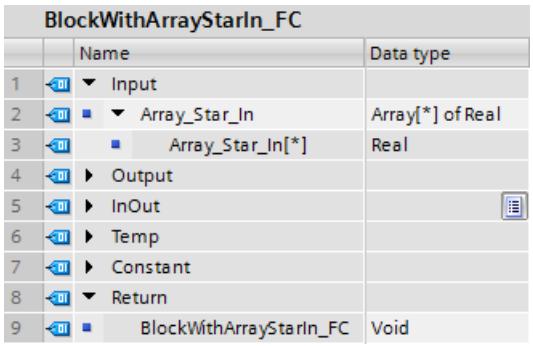

The assignment of ARRAY[*] to ARRAY[*] is permitted if the number of dimensions and the data type match. However, you cannot transfer individual ARRAY components in this process:

1. Declare the function block "BlockCaller_FB" and call the function "BlockWithArrayStarIn_FC":

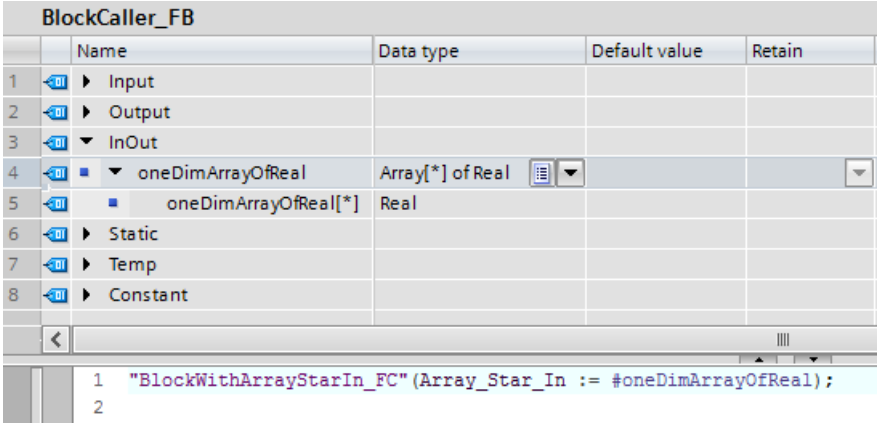

The assignment of an ARRAY with known limits to an ARRAY[*] is permitted if the number of dimensions and the data type match. Here, you can also assign individual ARRAY components.

1. Declare the function block "BlockCallerFixLimits_FB" and call the function "BlockWithArrayStarIn_FC" twice:

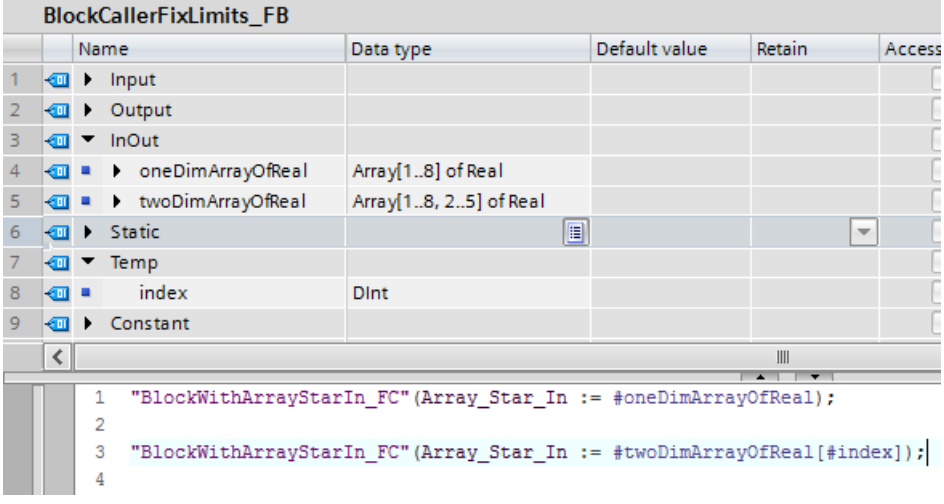

The assignment of ARRAY[*] to VARIANT is permitted. Here, you can also assign individual ARRAY components.

1. Create the function "BlockWithVariantIn_FC" so that it can be called:

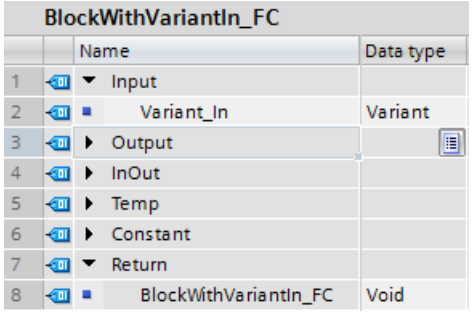

BlockWithVariantIn_FC

	Name	Data type
1	Input	
2	Variant_In	Variant
3	Output	
4	InOut	
5	Temp	
6	Constant	
7	Return	
8	BlockWithVariantIn_FC	Void

2. Declare the function "BlockWithArrayStarInVariant_FC" and call the function "BlockWithVariantIn_FC":

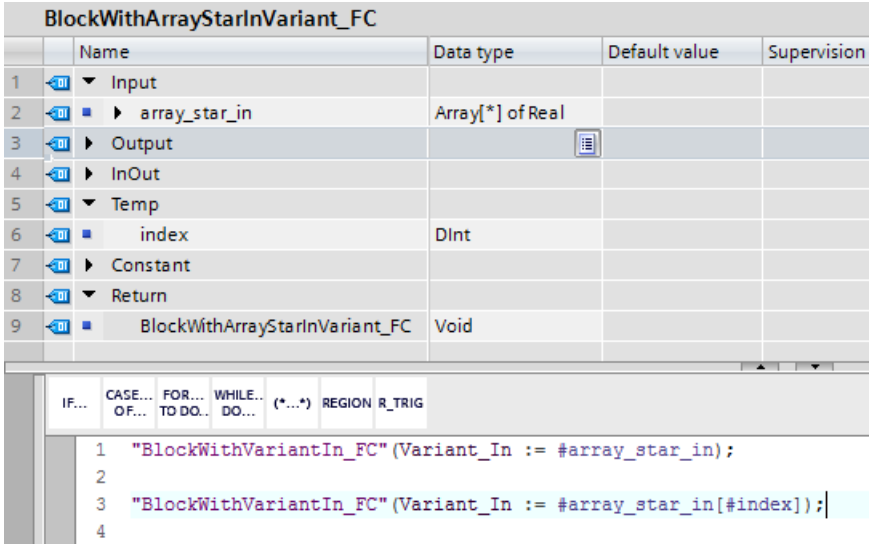

BlockWithArrayStarInVariant_FC

	Name	Data type	Default value	Supervision
1	Input			
2	array_star_in	Array[*] of Real		
3	Output			
4	InOut			
5	Temp			
6	index	DInt		
7	Constant			
8	Return			
9	BlockWithArrayStarInVariant_FC	Void		

IF... CASE... OF... FOR... TO DO... WHILE... DO... (*...*) REGION R_TRIG

```
1  "BlockWithVariantIn_FC"(Variant_In := #array_star_in);
2
3  "BlockWithVariantIn_FC"(Variant_In := #array_star_in[#index]);
4
```

The assignment of ARRAY[*] to ARRAYs with fixed limits is not permitted.

Note

Block parameters of the ARRAY[*] type must be supplied with an actual parameter.

See also

Rules for supplying block parameters (Page 76)

3.10.5 Examples of how to use ARRAYs

3.10.5.1 Example of a multi-dimensional ARRAY

Description

Tags of the ARRAY data type can contain up to 6 dimensions. The same rules apply as for one-dimensional ARRAYs. The dimension areas are written in the declaration in square brackets, each separated by a comma. In multidimensional arrays, the components are stored starting with the first dimension.

The following table shows the declaration of a two-dimensional tag of the ARRAY data type:

Name	Data type	Value	Comment
Betr_Temp	ARRAY[1..2, 1..3] of INT	1,1,4(0)	Two-dimensional tag of the ARRAY data type with 6 components The first two components are assigned the value "1". The remaining four components are assigned the value "0".

The following figure shows the structure of the declared tag of the ARRAY data type:

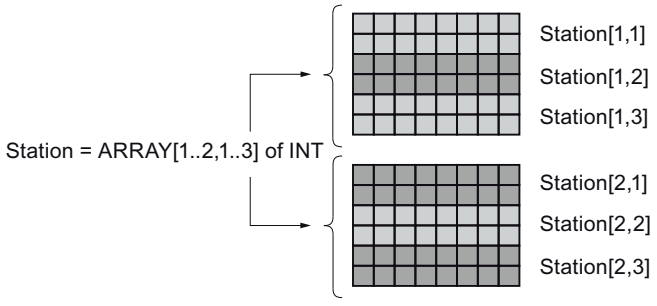

Access to the components

The values of the individual components are accessed via an index. You can use a constant or a tag for the index. The index of the first component is, for example, [1,1] and the index of the fourth array component is [2,1]. For example, you need to declare "Station[2,1]" in the program to enable access to the value of the fourth component.

3.10.5.2 Example for calculating the scalar product of two vectors with ARRAY[*]

Description

The scalar product is a mathematical link that assigns a number to two vectors (scalar).

The scalar product of two vectors results in a scalar variable and is defined by:

$$\vec{a} \cdot \vec{b} = a \cdot b \cdot \cos\alpha$$
$$= a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$$

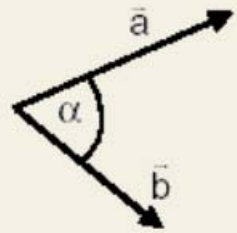

Here ∝ is the angle between the two vectors -$\vec{a}$ and $\vec{b}$.

An example for calculating the scalar product is:

$$\vec{a} = \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix}, \vec{b} = \begin{pmatrix} 2 \\ 0 \\ 4 \end{pmatrix}, \vec{a} * \vec{b} = (1*2+3*0+5*4) = 22$$

In this case, the result is the number 22.

In the following programming example, you see how you can calculate the scalar product of two vectors with flexible ARRAY bounds using ARRAY[*]. Here, the function "ScalarProduct_FC" serves as a template for calculating the individual scalar products.

You need the following objects for the programming example:

- Two data blocks and a PLC data type (UDT) for managing the vector data.
- Function that contains the program code for calculating the scalar product.
- The two instructions "LOWER_BOUND" and "UPPER_BOUND" to read out the ARRAY bounds.
- Organization block for calculating the scalar products

Procedure

In this example vectors, that is to say ARRAYs with 1 dimension, are used. Other calculations, such as matrix multiplications, can of course work with multi-dimensional ARRAYs.

To calculate the scalar products, follow these steps:

1. Create the PLC data type (UDT) "VectorArrays_UDT":

VectorArrays_UDT		
	Name	Data type
1	VectorD5Coordinates	Array[1..5] of Real
2	VectorD5Coordinates[1]	Real
3	VectorD5Coordinates[2]	Real
4	VectorD5Coordinates[3]	Real
5	VectorD5Coordinates[4]	Real
6	VectorD5Coordinates[5]	Real
7	VectorE13Coordinates	Array[1..13] of Real
8	VectorE13Coordinates[1]	Real
9	VectorE13Coordinates[2]	Real
10	VectorE13Coordinates[3]	Real
11	VectorE13Coordinates[4]	Real
12	VectorE13Coordinates[5]	Real
13	VectorE13Coordinates[6]	Real
14	VectorE13Coordinates[7]	Real
15	VectorE13Coordinates[8]	Real
16	VectorE13Coordinates[9]	Real
17	VectorE13Coordinates[10]	Real
18	VectorE13Coordinates[11]	Real
19	VectorE13Coordinates[12]	Real
20	VectorE13Coordinates[13]	Real

The two ARRAYs "VectorD5Coordinates" and "VectorE13Coordinates" provide a data basis with which you can then calculate scalar products.

2. Create the "VectorArrays1_DB" data block on the basis of the "VectorArrays_UDT" PLC data type:

VectorArrays1_DB		
	Name	Data type
1	Static	
2	VectorD5Coordinates	Array[1..5] of Real
3	VectorE13Coordinates	Array[1..13] of Real

3. Create the second "VectorArrays2_DB" data block. Besides the vectors from the "VectorArrays_UDT" PLC data type, it also contains two other vectors:

VectorArrays2_DB		
	Name	Data type
1	Static	
2	VectorA10Coordinates	Array[1..10] of Real
3	VectorB13Coordinates	Array[1..13] of Real
4	VectorsC	"VectorArrays_UDT"
5	VectorD5Coordinates	Array[1..5] of Real
6	VectorE13Coordinates	Array[1..13] of Real

You have now created the data basis for calculating the scalar products.

4. Create the "ScalarProduct_FC" function as a template for creating the calculation procedure for calculating the scalar product:
   The block interface:

ScalarProduct_FC			
	Name	Data type	Comment
1	Input		
2	Vector1	Array[*] of Real	Interface for data input for Vector 1
3	Vector1[*]	Real	Interface for data input for Vector 1
4	Vector2	Array[*] of Real	Interface for data input for Vector 1
5	Vector2[*]	Real	Interface for data input for Vector 1
6	Output		
7	InOut		
8	Temp		
9	Sum	Real	
10	i	Int	
11	LowerBound1	DInt	
12	LowerBound2	DInt	
13	Upperbound1	DInt	
14	UpperBound2	DInt	
15	Constant		
16	Return		
17	ScalarProduct_FC	Real	

   Program code:

```
1   #LowerBound1 := LOWER_BOUND(ARR := #Vector1, DIM := 1);
2   #LowerBound2 := LOWER_BOUND(ARR := #Vector2, DIM := 1);
3   #Upperbound1 := UPPER_BOUND(ARR := #Vector1, DIM := 1);
4   #Upperbound1 := UPPER_BOUND(ARR := #Vector1, DIM := 1);
5
6  IF #LowerBound1 <> #LowerBound2 OR #Upperbound1 <> #UpperBound2 THEN
7      #ScalarProduct_FC := -1;
8      RETURN;
9  END_IF;
10  #Sum := 0;
11 FOR #i := #LowerBound1 TO #Upperbound1 DO
12     #Sum += #Vector1[#i] * #Vector2[#i];
13 END_FOR;
14
15  #ScalarProduct_FC := #Sum;
16
```

   The upper and lower ARRAY bounds of vector 1 and vector 2 are queried in rows 1–4. It is then known how many coordinates the two ARRAYs of the vectors have. As the scalar product cannot be established until the number of coordinates of the two vectors that you wish to multiply is equal, rows 6 to 9 are required
   If the two lower or upper bounds of the ARRAYs are not the same, the "ScalarProduct_FC" function produces the function value "-1" and the program block is exited (RETURN).
   If the two lower or upper bounds of the ARRAY are the same, the #Sum tag in initialized (row 10) with the value "0" and the calculation of the scalar product is executed (rows 11 - 13).

5. Create the "Main_OB" organization block with the event class "Program cycle". The "ScalarProduct_FC" function created in step 4 now serves as a template for calculating scalar products in "Main_OB":
   The block interface:

Main_OB

	Name	Data type
1	Input	
2	Initial_Call	Bool
3	Remanence	Bool
4	Temp	
5	Result1	Real
6	Result2	Real
7	Result3	Real
8	Result4	Real
9	Result5	Real
10	Result6	Real
11	Result7	Real
12	Constant	

Program code:

```
#Result1 := "ScalarProduct_FC"(Vector1 := "VectorArrays2_DB".VectorA10Coordinates,
                               Vector2 := "VectorArrays1_DB".VectorE13Coordinates);
#Result2 := "ScalarProduct_FC"(Vector1 := "VectorArrays2_DB".VectorsC.VectorD5Coordinates,
                               Vector2 := "VectorArrays1_DB".VectorD5Coordinates);
#Result3 := "ScalarProduct_FC"(Vector1 := "VectorArrays2_DB".VectorB13Coordinates,
                               Vector2 := "VectorArrays1_DB".VectorE13Coordinates);
#Result4 := "ScalarProduct_FC"(Vector1 := "VectorArrays2_DB".VectorA10Coordinates,
                               Vector2 := "VectorArrays2_DB".VectorA10Coordinates);
#Result5 := "ScalarProduct_FC"(Vector1 := "VectorArrays2_DB".VectorA10Coordinates,
                               Vector2 := "VectorArrays1_DB".VectorE13Coordinates);
```

Result

The results of the individual scalar product calculations are written in the tags #Result1 -5.

3.11 Pointer

3.11.1 References (S7-1500)

3.11.1.1 Basic information about references (S7-1500)

Description

A reference is a tag that itself contains no value but rather points to the memory location of another tag.

References enable the forwarding of tags beyond block boundaries and thus the direct manipulation of tag values without having to create a copy of the tags.

When you declare the reference you specify from which data type the referenced tag must be. References are type-safe. This is a key aspect in particular for control system on which runtime errors need to be avoided. The restriction, made in IEC, that references must point to temporary data elements further increases the reliability. In this way runtime errors are avoidable.

It is ensured that references either refer to a valid memory space of the correct data type or are assigned the value NULL.

Using references in the program

Requirement for using references is a CPU of the S7-1500 series with the firmware version V2.5 or higher.

The following graphic provides an overview of options for using references.

Instruction	Application	Additional information
Declaring references REF_TO <data type> Declaration of the reference in the block interface	#myRefInt : REF_TO INT; #myRefType : REF_TO myType; #myRefArray : REF_TO Array[0..100] of myType;	Declaring references
Referencing REF (<tag>) Generation of a reference to a specific tag	#myRefInt := REF (#myInt); #myRefType := REF ("myDB".myType); #myRefArray := REF (#myArray); //myInt and myArray defined in Static	Referencing
Dereferencing <Reference>^ Access to the value of the referenced tag	#myInt := #myRefInt^; #myInt := #myRefType^.elementOfInt; #myInt := #myRefArray^[#index].elementOfInt;	Dereferencing
Standard instructions with references Assignment Comparison	#myRefInt := #yourRefInt; #myVARIANT := #myReference; //myVARIANT defined in Temp IF #myRefInt = NULL THEN...; IF #myRefInt <> NULL THEN...;	Standard instructions with references
Assignment attempt <Reference> ?= <VARIANT> Assignment attempt of a tag of data type VARIANT to a reference tag	#myReference ?= #myVariant;	Assignment attempt
Assignment during block call Transfer of references when calling functions or function blocks	"Callee"(MyRefIn1:= REF("MyDataBlock".MyInt), MyRefIn2:= NULL, MyRefOut=> #MyRefWord); //MyRefWord defined in Temp	Transfer references as block parameters

Differences between references and VARIANT

A tag of VARIANT data type is like a pointer that can point to other tags of any data type. For this reason, the data type that the VARIANT tag will point to does not have to be defined yet when the program is created. The data type is not defined until runtime. A VARIANT tag can even have different data types in different program cycles. The data type VARIANT is suitable for creating generic programs and for indirect addressing. If you want to further process a VARIANT tag in the program code, however, you need to use special instructions to determine which data type is currently present. You also cannot directly read and write VARIANT tags. Rather, special instructions, e.g. VariantGet and VariantPut must likewise be used for this.

If you use references, you define the data type when creating the program. Because the data type does not have to be determined at runtime, the program will perform better and be more clearly structured. Through dereferencing you can have direct write or read access to the referenced tag without having to integrate additional instructions in your program.

Unlike VARIANT, however, references may only point to data that are located in an optimized memory area.

Examples

The example below shows various possible applications for references.

The block interface contains the "myRefInt" tag that was declared as a reference:

Name	Data type	Default value
Input		
Output		
InOut		
myVARIANT	Variant	
Static		
a	Int	5
b	Int	10
result	Int	0
Temp		
myRefInt	REF_TO Int	

The figure below shows how this tag is used in SCL:

```
#result := #a + #b;          //#result has value 15
#myRefInt := REF(#a);        //#myRefInt points to #a
#myRefInt^:= 3;              //#a contains value 3
#result := #a + #b;          //#result has value 13



#myRefInt ?= #myVARIANT;     //Ok, when tag of data type INT is passed
                             // to #myVARIANT via block call
IF #myRefInt <> NULL THEN    //Check if Assignment Attempt was successful
    #result := #myRefInt^ + #a;
END_IF;
```

3.11.1.2 Declaring references (S7-1500)

Declaring references	
REF_TO <data type> Declaration of the reference in the block interface	#myRefInt : REF_TO INT; #myRefType : REF_TO myType; #myRefArray : REF_TO Array[0..100] of myType;

Description

References can be declared in the block interface of functions and function blocks. The following declaration areas are permitted for this purpose:

- FC: Input, Output, Temp, Return
- FB: Temp
- OB: Temp

To declare references, use keyword "REF_TO" and specify the required data type of the referenced tag. However, do not yet specify which specific tag the reference points to:

1	Input		
2	Output		
3	InOut		
4	Static		
5	Temp		
6	myRefInt	REF_TO Int	
7	myRefByte	REF_TO Byte	
8	myRefWord	REF_TO Word	
9	myRefDWord	REF_TO DWord	
10	myRefReal	REF_TO Real	
11	myRefLReal	REF_TO LReal	
12	myRefType	REF_TO "myUDT"	
13	myRefARRAY	REF_TO Array[0..3] of Int	
14	myRefARRAY[0]	Int	
15	myRefARRAY[1]	Int	
16	myRefARRAY[2]	Int	
17	myRefARRAY[3]	Int	

References can point to the following elements:

- Bit strings
  - References to BOOL are not possible.
- Integers
- Floating-point numbers
- Character strings
  - Length declarations for character strings are not possible.
- IEC timers
  - References to IEC_TIMER and IEC_LTIMER are possible.
  - References to derived data types, such as TON, are not possible.
- IEC counters
  - References to IEC_COUNTER / IEC_UCOUNTER, IEC_SCOUNTER / IEC_USCOUNTER, IEC_DCOUNTER / IEC_UDCOUNTER are possible.
  - References to derived data types, such as CTU, are not possible.
- PLC data types (UDT)
- System data types (SDT)
- ARRAYs of the named data types
  - References to ARRAY[*] are not possible.
  - In the current version of the TIA Portal, it is not possible to declare an ARRAY of references. The following declaration would therefore not be permitted:
    `ARRAY of REF_TO<Data type>`

You can only point to a whole data block if it was originally derived from a PLC data type (UDT) or system data type (SDT).

Initialization

When a reference tag is created, the system initializes it with the value NULL. This means that although the reference itself exists, it still does not refer to any valid memory. If a ZERO reference is accessed during runtime, a programming error is output. You cannot yourself enter an initialization in the block interface.

Use the instruction "REF ()" to initialize a reference.

See also: Referencing (Page 305)

Retentivity

References cannot be retentive. However, they can point to retentive data.

Note

Maximum number of reference parameters per block

The permissible maximum amount of parameters of data type "REF_TO" in a block depends on various factors, such as the block type and the number of other declared parameters with structured data type (ARRAYs, PLC data types, etc.) as well as the number of declared instances.

If the maximum number is exceeded, you receive a message during the compilation process. In this case, you can combine several parameters in a PLC data type (UDT) or in a global data block (DB) and transfer it as a block parameter.

See also

Basic information about references (Page 300)

Valid data types in the block interface (Page 6969)

3.11.1.3 Referencing (S7-1500)

Referencing REF (<tag>) Generation of a reference to a specific tag	#myRefInt := REF (#myInt); #myRefType := REF ("myDB".myType); #myRefArray := REF (#myArray); //myInt and myArray defined in Static

Description

The keyword "REF()" is used to specify the tag to which a previously declared reference is to point. As parameter, specify the tag to be referenced. The data type of the tag must correspond exactly to the data type of the declared reference. This means that a reference with data type "REF_TO Int" can only point to a tag of data type "Int". No data type conversion is made.

Note

Transferring "REF()" as an actual parameter during the block call

You can also transfer "REF()" as an actual parameter to a called block in whose interface references are declared.

See also: Transferring references as block parameters (Page 316)

Rules

The following rules apply to referencing:

- The data to which a reference points must be located in an optimized memory space.
- The reference must only point to data in a global DB or static tags.
- The reference must not refer to the following data:
  - Temporary local data
  - Global tags from the PLC tag table
  - Block parameters
  - Constants
  - Write-protected tags
- The following applies to references to arrays:
  - The array limits and dimensions of the reference and the referenced tag must be identical.
  - References to arrays with dynamic limits (Array[*]) cannot be created.
  - References to ARRAY DBs that are based on a PLC data type must be created as follows:
    ```
    REF("my_ArrayDB_UDT"."THIS")
    REF("my_ArrayDB_UDT"."THIS"[i])
    ```

Examples

The example below shows the interface of a block: The interface contains several parameters that were declared with data type "REF_TO". When declaring the reference, you define only the data type of the referenced tag. You do not yet specify the tag to which the reference points.

Name	Data type
Input	
Output	
InOut	
Static	
a	Int
Temp	
myRefInt	REF_TO Int
myRefType	REF_TO "myUDT"
myRefArrayInt	REF_TO Array[0..1] of Int
Constant	

In the program code you specify to which specific tag the declared reference parameters should point.

Examples in SCL:

```
#myRefInt := REF(#a);

#myRefType := REF("myDB".myUDT);

#myRefType := REF("myArrayDB_UDT"."THIS"[1]);

#myRefArrayInt := REF("myDB".myArray);
```

Examples in LAD:

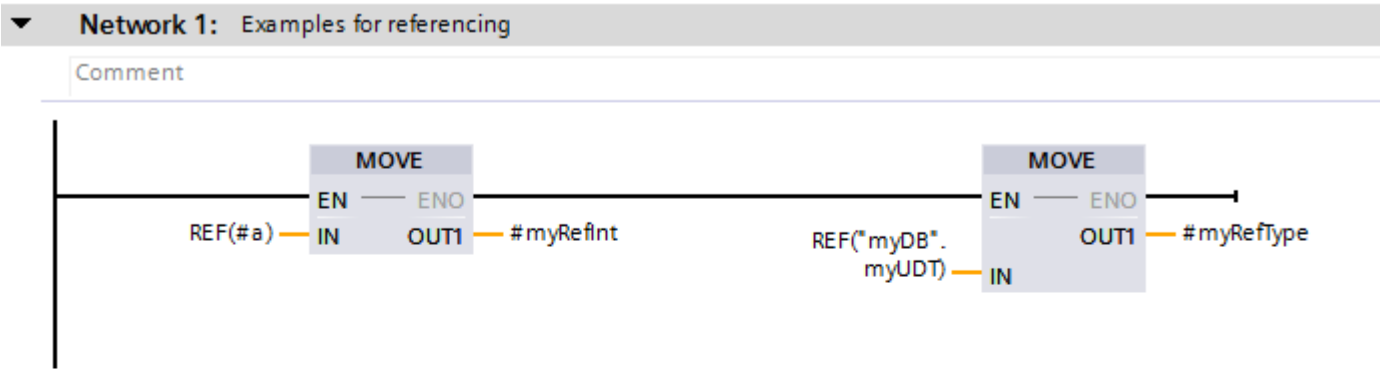

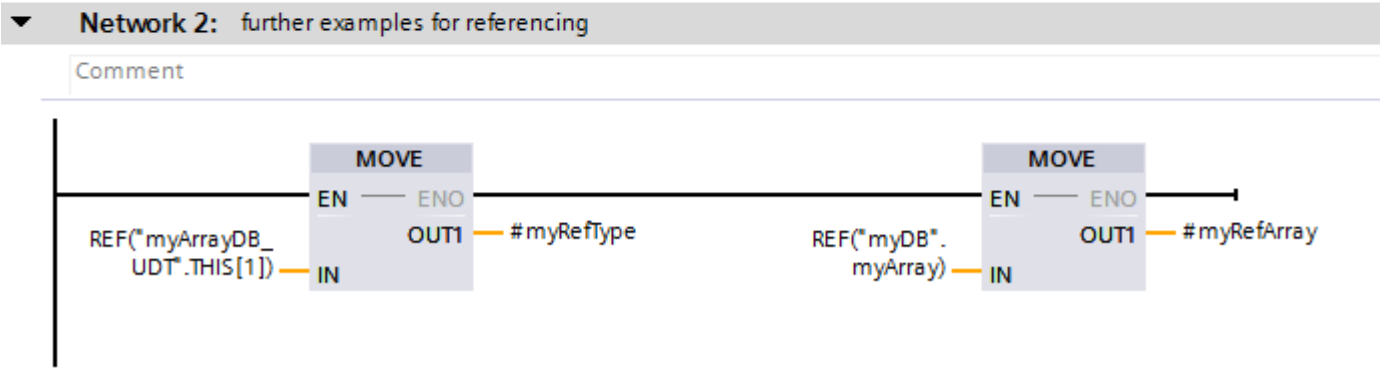

Examples in STL:

Network 1: Examples for referencing

Comment

```
1        CALL  MOVE
2           Variant
3           IN  :=REF(#a)
4           OUT :=#myRefInt
5
6
7        CALL  MOVE
8           Variant
9           IN  :=REF("myDB".myUDT)
10          OUT :=#myRefType
11
12
13       CALL  MOVE
14          Variant
15          IN  :=REF("myArrayDB_UDT".THIS[1])
16          OUT :=#myRefType
17
18       CALL  MOVE
19          Variant
20          IN  :=REF("myDB".myArray)
21          OUT :=#myRefArray
```

See also

Basic information about references (Page 300)

Declaring references (Page 303)

3.11.1.4 Dereferencing (S7-1500)

Dereferencing <Reference>^ Access to the value of the referenced tag	#myInt := #myRefInt^; #myInt := #myRefType^.elementOfInt; #myInt := #myRefArray^[#index].elementOfInt;

Description

To have read or write access to the value of a referenced tag, you use the caret character "^". This type of access is called "dereferencing".

Note

Before you can access the value of a reference, you must specify the tag to which the reference points with the "REF()" instruction.

Examples

The example below shows the interface of a block: The interface contains the "myRefInt" parameter that was declared as a reference and a few static parameters that have already been initialized with a value:

	Name	Data type	Default value
1	Input		
2	Output		
3	InOut		
4	Static		
5	a	Int	10
6	b	Int	20
7	result	Int	0
8	Temp		
9	myRefInt	REF_TO Int	
10	Constant		

The figure below shows how the "REF()" instruction is used to specify that "myRefInt" points to "#a" and how "#a" is used in calculations in SCL.

```
1  #result := #a + #b;              // #result has the value 30
2
3  #myRefInt := REF (#a);           // #myRef1 points to #a
4  #myRefInt^ := 40;                  // #a has the value 40
5  #result := #a + #b;              // result has the value 60
6  #result := #myRefInt^ + #b;        // result also has the value 60
7
```

See also

Basic information about references (Page 300)

3.11.1.5 Standard instructions with references (S7-1500)

Standard instructions with references	
Assignment	#myRefInt := #yourRefInt; #myVARIANT := #myReference; //myVARIANT defined in Temp
Comparison	IF #myRefInt = NULL THEN...; IF #myRefInt <> NULL THEN...;

You can use references as input or output parameters of assignments or comparison instructions.

You can use dereferenced values as parameters of any instruction, provided they have the correct data type.

Assignment

References can be assigned to each other like any other tag. In this case, the address of the reference, not its value, is assigned to the second reference. References can only be assigned to each other if they refer to the same data type. No implicit data type conversion is made.

References to PLC data types, too, must be of the same data type. It is not enough for both PLC data types to have the same structure.

You can also assign a reference to a VARIANT. In this case the VARIANT must be declared as temporary tag (Temp).

Comparison

References can be used in compare instructions. You can use a comparison with NULL to determine whether a tag has already been assigned to a reference. This is always advisable when the program flow does not explicitly ensure that a reference has been initialized.

Other comparisons are not possible.

Examples

The following example shows the use of references in assignments and comparison instructions in SCL:

```
1  // Assignment of one reference to another reference
2  #myRefInt1 := #myRefInt2;
3
4  // Assignment of a reference to a variant
5  #myVariant := #myReference;
6
7  // Assignment of a value to the element "SetLimit"
8  // in the referenced PLC data type "Speed"
9  #myRefType^.SetLimit := 8;
10
11 // Assignment of a value to an element in a referenced ARRAY
12 #RefARRAY^[0] := 8;
13
14 // A reference is compared to NULL to determine if
15 // a valid reference for myRefInt has already been created.
16 IF #myRefInt <> NULL THEN
17     #myRefInt^ := 10;
18 END_IF;
19
20
```

The following example shows the use in LAD:

Network 1:

Assignment of one reference to another reference

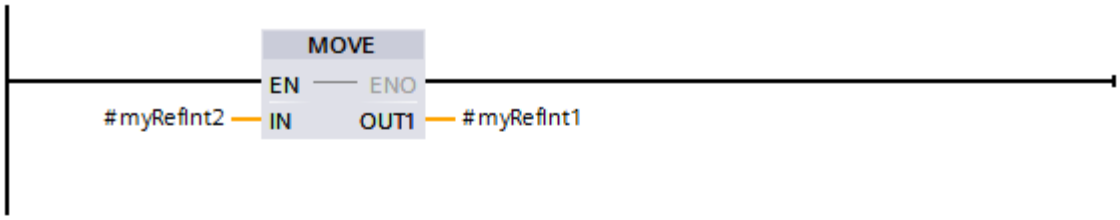

Network 2:

Assignment of a reference to a variant

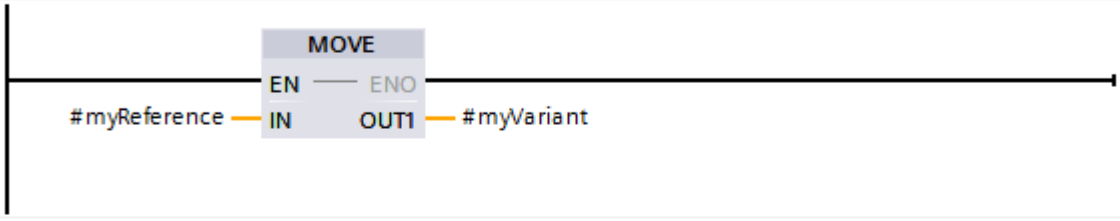

Network 3:

Assignment of a value to the element "SetLimit" in the referenced PLC data type "Speed"

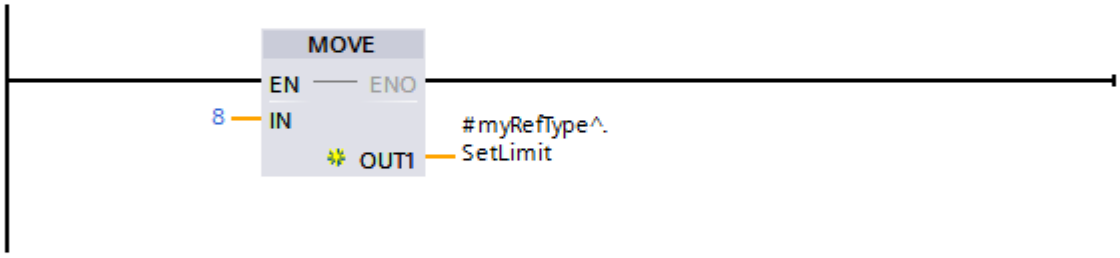

Network 4:

Assignment of a value to an element in a referenced ARRAY

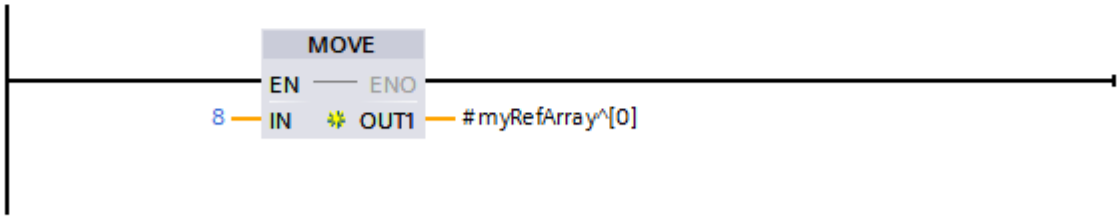

Network 5:

A reference is compared to NULL to determine if a valid reference for "myRefInt" has already been created

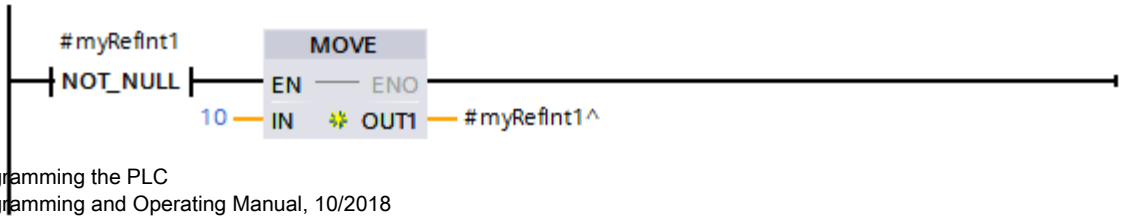

See also

Basic information about references (Page 300)

3.11.1.6 Assignment attempt from VARIANT to a reference (S7-1500)

Assignment attempt	
<Reference> ?= <VARIANT> Assignment attempt of a tag of data type VARIANT to a reference tag	#myReference ?= #myVariant;

Description

You can use the assignment attempt with "?=" to assign a tag of data type VARIANT to a reference.

If the tag addressed by VARIANT at runtime has the correct data type and is located in an optimized memory area, there is a valid reference in the destination tag after execution of the assignment, otherwise NULL.

After the assignment attempt, you can check if the attempt was successful and, depending on the result, continue processing the program. In LAD and FBD, you can use the enable output ENO for the check. ENO returns the signal state "1" if the assignment attempt was successful. Only then are the subsequent instructions executed in the network.

In STL and SCL you can use the instructions "IS_NULL" or "NOT_NULL", for example, to check the success of an assignment attempt. See the examples below.

Rules

The following rules apply to the assignment attempt:

- The assigned VARIANT must not point to a temporary tag.
- The assigned VARIANT must point to a tag in the optimized memory area.
- If you want to assign a tag of data type VARIANT to a reference to an ARRAY, VARIANT must point to an ARRAY whose limits exactly match the limits of the declared reference. A VARIANT which points to an ARRAY [0..9], does not match a tag REF_TO ARRAY[1..10].
- In SCL, assignment attempts cannot be used in multiple assignments (a := b := c;).

Example

The following example shows how you can use references to symbolically read or write the value of a VARIANT tag directly. It is not necessary to copy the value with the "VariantGet" and "VariantPut" instructions.

Name	Data type
Input	
<Hinzufügen>	
Output	
<Hinzufügen>	
InOut	
variantTelegramData	Variant
Static	
<Hinzufügen>	
Temp	
refTelegram1	REF_TO "Telegram1"
refTelegram2	REF_TO "Telegram2"

The VARIANT tag "variantTelegramData" can transfer data of type "Telegram1" and data of type "Telegram2".

An assignment attempt is used to test whether the data is of the type "Telegram1". If this is the case, the values "T" and "W" are assigned to the parameters "Info1" and "Info2".

Example in SCL:

```
1  #refTelegram1 ?=  #variantTelegramData: // AssignmentAttempt
2 IF #refTelegram1 <> NULL THEN           //Check if Assignment Attempt was successful
3          #refTelegram1^.Info1 := 'T' ; // Dereference by using a
4          #refTelegram1^.Info2 := 'W' ; // succeeding caret ^
5 END_IF;
6
```

Example in LAD:

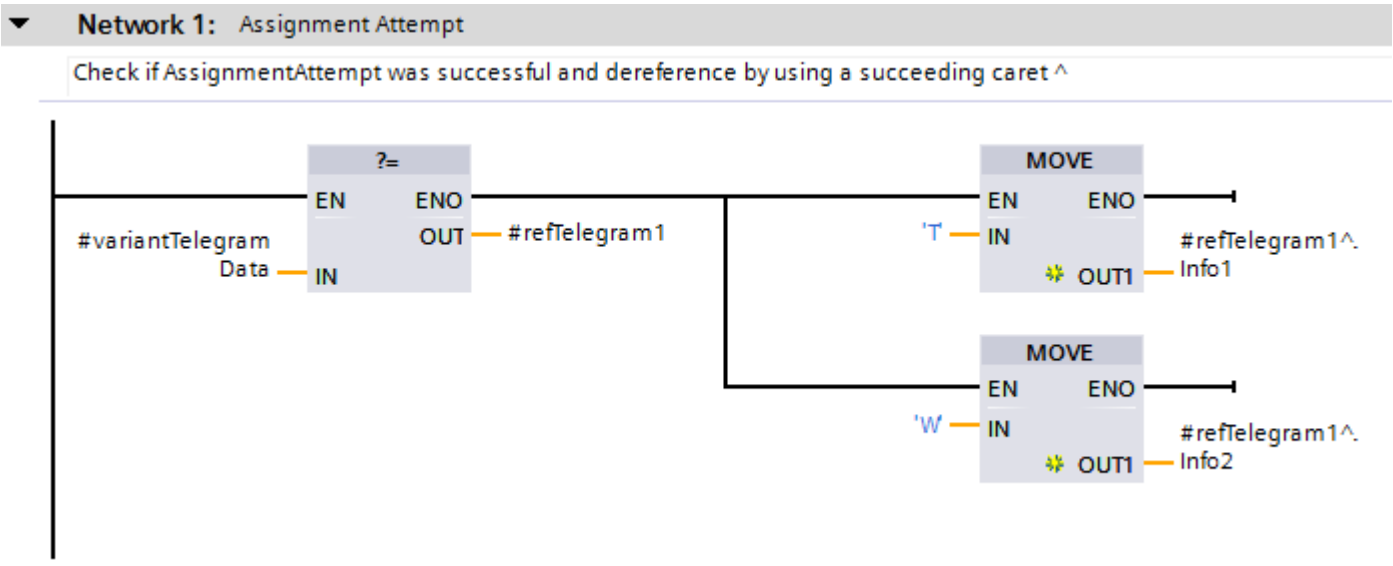

See also

Basic information about references (Page 300)

3.11.1.7 Transferring references as block parameters (S7-1500)

Description

References can also be transferred as block parameters when calling functions or function blocks.

When you call a block in whose interface a reference is declared as a formal parameter, you can transfer a reference in the form of "REF()" or "#MyRef" as an actual parameter.

Rules

The following rules apply to the transfer of references as block parameters:

- "REF()" can only be transferred as an actual parameter for an input parameter (Input) of a function.
- "REF()" cannot be transferred as an actual parameter to a block parameter of data type "VARIANT".
- The data type of the transferred tag must correspond exactly to the data type of the declared reference. No data type conversion is made.

Example 1

The example below shows the interface of the "Callee" function. The interface contains several parameters that were declared as reference:

Callee		
	Name	Data type
1	Input	
2	MyRefIn1	REF_TO Int
3	MyRefIn2	REF_TO Int
4	Output	
5	MyRefOut	REF_TO Word
6	InOut	
7	Temp	
8	Constant	
9	Return	
10	Callee	REF_TO Word

The following figure shows the call of this block in SCL and the parameter assignment for the declared references.

```
1
2 #ref := "Callee"(MyRefIn1 := REF("MyDataBlock".MyInt),
3                  MyRefIn2 := NULL,
4                  MyRefOut => #MyRefWord);
```

Example 2

The following example shows additional options for transferring references during a block call:

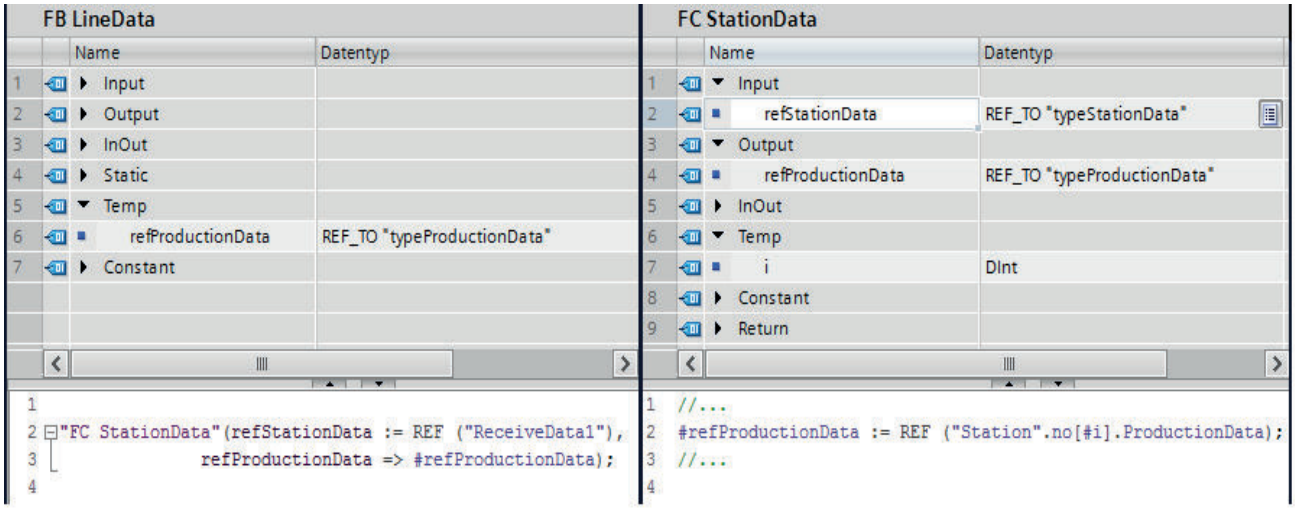

The example shows two possibilities for the transfer:

- Transfer from the FB to the called FC:
  The input parameter "refStationData" is declared in the "FC StationData".
  "FB LineData" transfers `REF("ReceiveData1")` as an actual parameter (line 2 in the program code of the "FB LineData") during the call. "ReceiveData1" is a data block that is based on the PLC data type "typeStationData".
- Return from the FC to the calling FB:
  References with the same data type are declared in the interface of both blocks (in the example: `"refPoductionData"`).
  During the block call, the two references are assigned to each other (line 3 in the program code of the "FB LineData").
  The reference is initialized in the called "FC StationData" (line 2) and is written to the temporary data of the calling FB via the output parameter.

The example shows a section from the programming example "Return data to the calling block via reference". You can find the detailed program code under "See also".

See also

Basic information about references (Page 300)

Dereferencing (Page 309)

Example: Return data to the calling block via reference (Page 321)

3.11.1.8 Example: Transferring tags of different data types using references (S7-1500)

Task

In the following application example, a production plant requests data. Different data is transferred to the plant depending on the type of data structure requested.

The data structures are mapped in two different PLC data types. The PLC data type "typeTelegram1" contains general information of the data type "CHAR", the PLC data type "typeTelegram2" contains the part ID of the data type "STRING". A data block is generated from each of the two PLC data types:

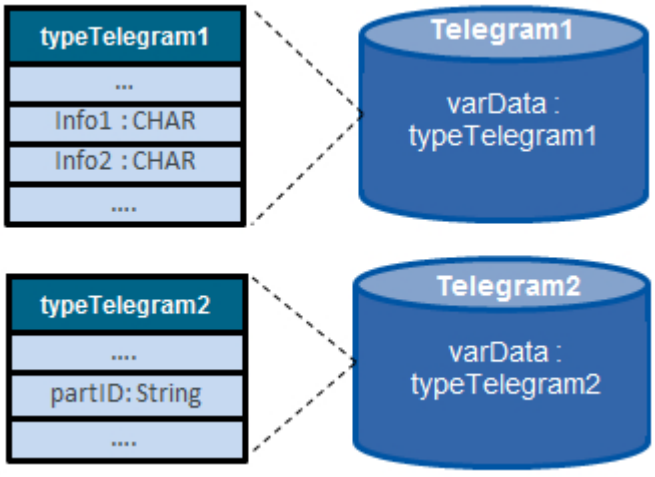

Example programs

The following figure shows the implementation of the task with and without the use of references in SCL:

Example without references

	Name	Data type
1	Input	
2	Output	
3	InOut	
4	telegram	Variant
5	Static	
6	Temp	
7	tempTelegram1	"typeTelegram1"
8	tempTelegram2	"typeTelegram2"
9	Constant	

```
1
2  CASE TypeOf(#telegram) OF
3    typeTelegram1:
4      VariantGet(SRC := #telegram,    // copy data to "Temp"
5                 DST => #tempTelegram1);
6
7      #tempTelegram1.Info1 := 'T';
8      #tempTelegram1.Info2 := 'W';
9
10     VariantPut(SRC := #tempTelegram1,// copy back to "Inout"
11                DST := #telegram);
12
13   typeTelegram2:
14     VariantGet(SRC := #telegram,    // copy data to "Temp"
15                DST => #tempTelegram2);
16
17     #tempTelegram2.partID := 'RE2346';
18
19     VariantPut(SRC := #tempTelegram2,// copy back to "Inout"
20                DST := #telegram);
21 END_CASE;
```

Example with references

	Name	Data type
1	Input	
2	Output	
3	InOut	
4	telegram	Variant
5	Static	
6	Temp	
7	refTelegram1	REF_TO "typeTelegram1"
8	refTelegram2	REF_TO "typeTelegram2"
9	Constant	

```
1
2  CASE TypeOf(#telegram) OF
3    typeTelegram1:
4      #refTelegram1 ?= #telegram;   // AssignmentAttempt
5      #refTelegram1^.Info1 := 'T';  // Dereference by using
6      #refTelegram1^.Info2 := 'W';  // a succeeding caret ^
7
8    typeTelegram2:
9      #refTelegram2 ?= #telegram;  // AssignmentAttempt
10     #refTelegram2^.partID:='RE2346';
11 END_CASE;
12
```

Because both blocks should be able to process different telegram formats, they have an in-out parameter (InOut) of data type "VARIANT" at which the telegrams are transferred as structured tags. Since the data type of a VARIANT parameter is not yet known at the time the program is created, it is not possible to directly access the parameter. Therefore, in both examples, "TypeOf" is used to determine which data type is currently available at the time of the call.

In the left block, the tag pending during runtime is now copied into a temporary tag of the corresponding data type using the "VariantGet" instruction, where it is described with the appropriate values. Then the structure is copied back to the "telegram" parameter using the "VariantPut" instruction. Copying has a detrimental effect on the program runtime and costs memory space.

In the right block, references are used to solve the task: The interface contains one reference parameter each for the two possible data structures. Depending on the currently pending tag, the matching reference is initialized using "AssignmentAttempt" and now points to the memory location of the tag in the data block. The values are now written directly to the data block.

The program code is clearer and easier to maintain. Because the structured tags do not have to be copied, the program runtime and memory requirements are not affected.

3.11.1.9 Example: Return data to the calling block via reference (S7-1500)

Task

In the following application example, we are configuring a production plant that consists of multiple stations. The stations are controlled by two CPUs which send data to a control CPU. This CPU must analyze for which station new data exists and execute the higher-level control tasks, for example, adding up the number of produced pieces.

The following figure shows the structure of the production plant:

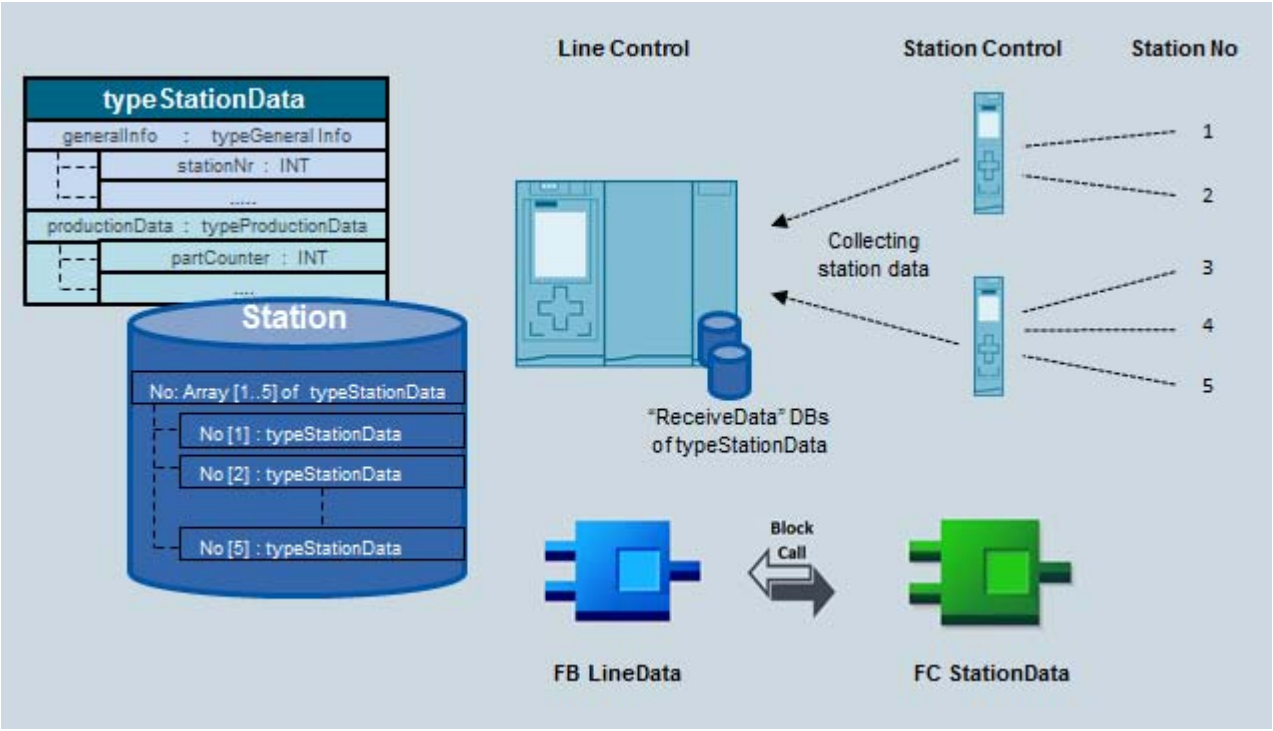

Blocks in the example program

The program on the control CPU is structured as follows:

- PLC data type "typeStationData"
  The data of a station is mapped in the PLC data type "typeStationData". The PLC data type contains two lower-level PLC data types:
  - "typeGeneralInfo"
    The data type contains the station number.
  - "typeProductionData"
    The data type contains the number of pieces of the station.
- DB "ReceiveData"
  For communication between the CPUs, a "ReceiveData" data block of the type "typeStationData" is available for each CPU. The communication data is written to this block.
- "FB LineData"
  The FB runs a complete analysis of the production line. This includes, for example, adding up all unit counters of the individual stations. It also calls the "FC StationData" which, among other things, copies the received data from the "ReceiveData" DB to the global DB "Station".
- "FC StationData"
  The FC checks from which station new data was received and copies it to the corresponding ARRAY element in the global DB "Station". The ARRAY element is then returned as reference to the calling "FB LineData" for further processing.
- Global DB "Station"
  This DB contains the station information of the five stations. It is stored in an ARRAY of the type "typeStationData" from five components.

Example program

The following figure shows the implementation of the task in SCL:

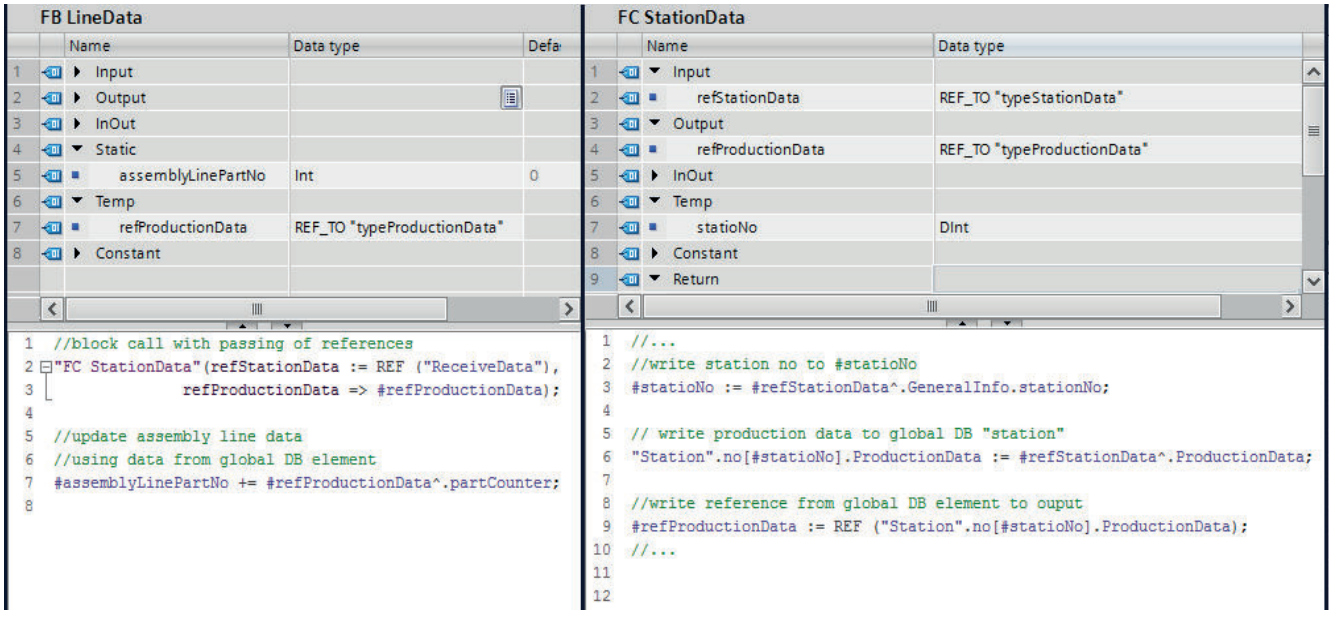

The "FB LineData" calls the "FC StationData" and transfers the reference to the receive data block when being called by `REF("ReceiveData")`.

By dereferencing to the transferred receive data block, the FC reads out the station number and writes it to the temporary tag "stationNo".

The production data from the receive data block is then copied to the corresponding ARRAY element in the global DB "Station".

This ARRAY element is returned as reference to the calling "FB LineData" via the "refProductionData" output.

It can subsequently access the transferred ARRAY element directly by dereferencing and thus update the unit counter for the entire production line.

See also

Transferring references as block parameters (Page 316)

3.11.2 VARIANT (S7-1200, S7-1500)

3.11.2.1 Basic information on VARIANT (S7-1200, S7-1500)

Description

A parameter of VARIANT data type is a pointer or a reference It can point to tags of various data types. The VARIANT pointer cannot point to an instance and therefore cannot point to a multiple instance or ARRAY of multiple instances. The VARIANT pointer can be an object of an elementary data type, such as INT or REAL. It can also be a STRING, DTL, ARRAY of STRUCT, UDT, or ARRAY of UDT. The VARIANT pointer can recognize structures and point to individual structure components. An operand of data type VARIANT occupies no space in the instance data block or work memory. However, it will occupy memory space on the CPU.

A tag of the VARIANT type is not an object but rather a reference to another object. Individual elements of the VARIANT type can only be declared on formal parameters within the block interface of a function in the VAR_IN, VAR_IN_OUT and VAR_TEMP sections. For this reason, it cannot be declared in a data block or in the static section of the block interface of a function block, for example, because its size is unknown. The size of the referenced objects can change.

Using the VARIANT data type, you can in particular create generic, standardized function blocks (FB) or functions (FC) for various data types. Various instructions in all programming languages are available to you for this purpose. During the creation of the program, you can specify which data types the block should be able to process. The VARIANT data type supports you here by allowing the interconnection of any tags. You can then react accordingly to their data types in the block. When a block is called, you can connect the parameters of the block to tags of any data type. When a block is called, the type information of the tag is transferred in addition to a pointer to the tag. The code of the block can then be executed according to its type in line with the tag transferred during runtime.

If, for example, a block parameter of a function has the VARIANT data type, then a tag of the integer data type can be transferred at one point in the program, and a tag of the PLC data type can be transferred at another point in the program. With the help of the VARIANT instructions, the function is then in a position to react to the situation without errors.

Note

You can only point to a complete data block if it was originally derived from a user-defined data type (UDT).

Note

Address I/O

Direct reading or writing of a signal from an I/O input or output is only possible on a CPU of an S7-1500 module. (<Operand>: P)

The following table shows the properties of the VARIANT pointer:

Length (bytes)	Repre-sentation	Format	Example of value input
0	Symbolic	Operand	"TagResult"
		NameDataBlock.NameOper-and.Component	"Data_TIA_Portal".StructVariable.First-Component
	Absolute	Operand	%MW10
		DataBlockNumber.Operand Type Length	P#DB10.DBX10.0 INT 12 [1]
		NULL pointer	NULL
[1] If you use the prefix P#, you can only point to memory areas with "standard" access mode.			

Coding of data types

If you use absolute addressing with P#, the following data types are permitted:

- BOOL
- BYTE
- CHAR
- WORD
- INT
- DWORD
- DINT
- REAL
- TIME
- S5TIME
- DATE
- TOD
- DT

Example

The following example shows how VARIANT works using the "MOVE: Move value" instruction of STL:

STL	Explanation
`CALL MOVE`	`// The instruction is called.`
`value_type := VARIANT`	`// Data type of the parameters IN and OUT`
`IN := "Data_TIA_Portal".StructVaria-ble.FirstComponent`	`// The contents of the "FirstComponent" operand are moved from the "Data_TIA_Por-tal" DB.`
`OUT := "MotorDB".StructResult.TagResult`	`// And transferred to the "TagResult" op-erand from the "MotorDB" DB.`

See also

Basics of indirect addressing (Page 114)
Overview of the valid data types (Page 231)
Basics of constants (Page 97)
Basics of block access (Page 55)
Principles of block calls (Page 60)
VARIANT instructions (Page 327)
Accessing I/O devices (Page 104)
Data type conversion for S7-1200 (Page 489)

3.11.2.2 Applications for pointers in the comparison (S7-1200, S7-1500)

Application cases for pointers with an S7-1200/1500 CPU as compared to S7-300/400

The following table provides you with an overview of the various applications for pointers on a CPU of the S7-300/400 series (ANY pointer) and their solution with a CPU of the S7-1200/1500 series.

In most applications, the use of a pointer is no longer required on a CPU of the S7-1200/1500 series. Instead, much simpler language resources are available.

It is only advisable to use the VARIANT data type for indirect addressing, when the data types are only determined during the program runtime.

What is the purpose of the ANY pointer?	Recommendations in the TIA Portal (S7-1200/ S7-1500)
Moving data of any source and destination data type in the program using the instruction "BLKMOV: Move block".	Definition of tags within a PLC data type. You can use the instructions "Serialize" and "Deserialize" to move the tags.
Initializing an ARRAY structure	Using the instruction "FILL_BLK: Fill block", you initialize or fill an ARRAY structure.
Moving ARRAY elements	Using the instruction "MOVE_BLK: Move block". you move the content of multiple elements of an ARRAY structure to another ARRAY structure.
Memory and performance optimization using structured data	Use the InOut section in the block interface to optimize memory and performance. You can find additional information in the "Programming Guideline for S7-1200/1500" under the following link Programming Guideline for S7-1200/1500 (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=4&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid4=20229695&objaction=csopen)

What is the purpose of the ANY pointer?	Recommendations in the TIA Portal (S7-1200/ S7-1500)
Access to individual bits/bytes of a WORD	Use the "slice access" You can find additional information here: Example of a slice access (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid0=29156492&objaction=csopen)
Determination of the length of structures or data blocks	Use an ARRAY and read its length using the instruction "CountofElements: Get number of ARRAY elements". The instruction only works in conjunction with the data type VARIANT.
Indirect addressing	You can use the VARIANT pointer for the indirect addressing of data types that will only be known during runtime. You can use the data type DB_ANY for indirect access to a data block.

3.11.2.3 VARIANT instructions (S7-1200, S7-1500)

VARIANT instructions

The following instructions for working with VARIANT are available to you in the TIA Portal:

Basic instructions		
Category	Instruction	Description
Comparator operations	EQ_Type	Compare data type for EQUAL with the data type of a tag
	NE_Type	Compare data type for UNEQUAL with the data type of a tag
	EQ_ElemType	Compare data type of an ARRAY element for EQUAL with the data type of a tag
	NE_ElemType	Compare data type of an ARRAY element for UNEQUAL with the data type of a tag
	IS_NULL	Query for EQUALS ZERO pointer
	NOT_NULL	Query for UNEQUALS ZERO pointer
	IS_ARRAY	Check for ARRAY
	TypeOf	Check data type of a VARIANT tag
	TypeOfElements	Check element data type of a VARIANT tag
Move operations	MOVE_BLK_VARIANT	Move block
	VariantGet	Read out VARIANT tag value
	VariantPut	Write VARIANT tag value
	CountOfElements	Get number of ARRAY elements
Conversion operations	VARIANT_TO_DB_ANY	Convert VARIANT to DB_ANY
	DB_ANY_TO_VARIANT	Convert DB_ANY to VARIANT

Note

Differences between MOVE, MOVE_BLK and MOVE_BLK_VARIANT

- You can use the "MOVE" instruction to copy complete structures.
- You can use the "MOVE_BLK" instruction to move parts of ARRAYs with known data type.
- The MOVE_BLK_VARIANT instruction is only required if you want to move parts of ARRAYs with a data type that is only known during program runtime.

You can find additional information on the individual instructions in the information system under "Basic instructions > Respective programming language".

You can also find additional instructions which also work with the VARIANT data type under the "Extended instructions".

See also

Basic information on VARIANT (Page 324)

Indirect addressing with the VARIANT data type (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid0=29156492&objaction=csopen)

3.11.2.4 Application options for VARIANT instructions (S7-1200, S7-1500)

Introduction

In the following chapter, you will learn which application options you have with VARIANT instructions.

Evaluating data types of tags to which a VARIANT points

In the table below, you see which instructions are available to you to evaluate data types of tags to which a VARIANT points:

Function	Instruction	Description
To determine the data type	TypeOf(): Check data type of a VARIANT tag (This instruction is available only in SCL and only in conjunction with an IF instruction.)	You use this instruction to compare the data type to which a VARIANT tag points with the data type of any other tag. You can also make the comparison with a PLC data type.
	TypeOfElements(): Check element data type of a VARIANT tag (This instruction is available only in SCL and only in conjunction with an IF instruction.)	You use this instruction to compare the data type to which a VARIANT tag points with the data type of any other tag. You can also make the comparison with a PLC data type. If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.
	EQ_Type: Compare data type for EQUAL with the data type of a tag NE_Type: Compare data type for UNEQUAL with the data type of a tag	You use this instruction to compare the data type to which a VARIANT tag points with the data type of any other tag. You can also make the comparison with a PLC data type.
	EQ_ElemType: Compare element data type for EQUAL with the data type of a tag NE_ElemType: Compare element data type for UNEQUAL with the data type of a tag	You use this instruction to compare the data type to which a VARIANT tag points with the data type of any other tag. You can also make the comparison with a PLC data type. If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.
To evaluate ARRAY elements	IS_ARRAY: Check for ARRAY	You use this instruction to check whether the data type to which a VARIANT tag points is an ARRAY.
	CountOfElements: Get number of ARRAY elements	You use this instruction to read out how many ARRAY elements the tag has to which the VARIANT tag points.

You can find additional information on the individual instructions in the information system under "Basic instructions > Respective programming language".

Reading data to which a VARIANT points

To be able to use the data you must move this data to a tag as an intermediate step, as it cannot be processed directly.

Instruction	Description	Example		Result
		VARIANT points to	Destination data type	
VariantGet: Read out VARIANT tag value	You use this instruction to move the value of a single tag to another tag. The data types of both tags must match.	UDT_1	UDT_1	The instruction is executed
		REAL	REAL	
		DINT	DWORD	The instruction is not executed.

Assigning data to a VARIANT tag

You cannot use the instruction to initialize VARIANT tags. The VARIANT tags must therefore already be initialized when the data is returned to the tag. Do not use uninitialized temporary VARIANT tags.

Instruction	Description	Example		Result
		Source data type	VARIANT points to:	
VariantPut: Write VARIANT tag value	You use this instruction to move the value of a single tag to another tag. The data types of both tags must match.	UDT_1	UDT_1	The instruction is executed
		REAL	REAL	
		DINT	DWORD	The instruction will not be executed, as the data types are different.

Processing dynamic ARRAY structures

To evaluate ARRAY elements	TypeOfElements(): Check data type of an ARRAY element of a VARIANT tag (This instruction is available only in SCL and only in conjunction with an IF instruction.)	You use this instruction to compare the data type to which a VARIANT tag points with the data type of any other tag. You can also make the comparison with a PLC data type. If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.
	IS_ARRAY: Check for ARRAY	You use this instruction to check whether the data type to which a VARIANT tag points is an ARRAY.
	CountOfElements: Get number of ARRAY elements	You use this instruction to read out how many ARRAY elements the tag has to which the VARIANT tag points.
	MOVE_BLK_VARIANT: Move block	This instruction is used to move dynamic and type-safe (integrated type test) ARRAYs. You can freely select the limit values for the source and destination ARRAYs. The data types of ARRAY elements must match.

Note

Differences between MOVE, MOVE_BLK and MOVE_BLK_VARIANT

- You can use the "MOVE" instruction to copy complete structures.
- You can use the "MOVE_BLK" instruction to move parts of ARRAYs with known data type.
- The MOVE_BLK_VARIANT instruction is only required if you want to move parts of ARRAYs with a data type that is only known during program runtime.

Additional information on the use of the MOVE_BLK_VARIANT instruction can be found in the "Moving data" programming example.

See also

Basic information on VARIANT (Page 324)

Example of moving data (Page 333)

3.11.2.5 Initializing VARIANT (S7-1200, S7-1500)

Description

Initialize the VARIANT data type by assigning a specific tag to the VARIANT block parameter at the block call. This forms a reference to the address of the passed tag. To do this, create a block parameter of the VARIANT data type in the block interface. In the following example, these are the two block parameters SourceArray and DestinationArray in the section InOut.

FC_PartialArrayCopy

	Name	Data type
1	Input	
2	Count	UDInt
3	SourceIndex	DInt
4	DestinationIndex	DInt
5	Output	
6	InOut	
7	SourceArray	Variant
8	DestinationArray	Variant
9	Temp	
10	Error	Int
11	Constant	
12	<Add new>	
13	Return	
14	FC_PartialArrayCopy	Int

The example shows a section from the programming example "Example of moving data". You can find the detailed program code under "See also".

Note

Direct passing of a tag to a VARIANT tag is not possible, such as myVARIANT := #Variable

See also

Example of moving data (Page 333)

3.11.2.6 Transferring and reading out various data types with VARIANT (S7-1200, S7-1500)

The example shows a section from the programming example "Example of moving data". You can find the detailed program code under "See also".

Note

VARIANT as formal parameter

If the VARIANT is declared as a formal parameter, no write-protected data may be transferred as actual parameters.

Passing various data types

In the following example, you see how the VARIANT block parameter can be initialized with different tags when a generic, standardized function is called multiple times:

The "FC_PartialArrayCopy" function is called twice. The VARIANT parameter SourceArray is interconnected with an ARRAY of "my_struct" with the left call. The VARIANT parameter SourceArray is interconnected with an ARRAY of REAL with the right call.

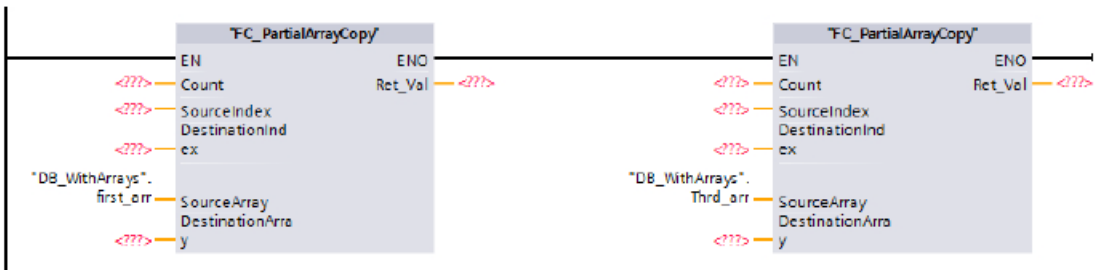

Reading out and checking data types

Various comparison instructions are available to you to read out the data type of a tag or an element and compare it with data types of other tags or elements.

In the figure below, you see the usage of multiple comparison instructions to check whether the elements of the ARRAYs have the same data type:

```
6  IF IS_ARRAY(#SourceArray) AND IS_ARRAY(#DestinationArray)
7      AND TypeOfElements(#SourceArray) = TypeOfElements(#DestinationArray) THEN
8      #Error := MOVE_BLK_VARIANT(COUNT := #Count, SRC := #SourceArray, SRC_INDEX := #SourceIndex,
9                                 DEST => #DestinationArray, DEST_INDEX := #DestinationIndex);
10 END_IF;
```

The MOVE_BLK_VARIANT instruction is only executed if the data types of the ARRAY elements are the same.

See also

Example of moving data (Page 333)

3.11.2.7 Examples of using the VARIANT data type (S7-1200, S7-1500)

Example of moving data (S7-1200, S7-1500)

Programming example

In this programming example, data values, which are collected for example during a production shift, are moved for further processing. The data is collected in an ARRAY. Using the "MOVE_BLK_VARIANT instruction: Move block", you can move either the entire ARRAY or only individual ARRAY elements dynamically and type-safe. You can freely select the ARRAY limits for the respective source and destination ARRAYs and these do not have to match. However, the data type of the data values that are to be moved must match. This instruction is available in all programming languages.

By using the VARIANT data type, you can use the created program code to move data for other production shifts as well, by specifying a different source and destination area at the block call.

Procedure

1. Create a function using the SCL programming language and give it the name "FC_PartialArrayCopy".
2. Declare the block interface as follows:

FC_PartialArrayCopy

	Name	Data type
1	Input	
2	Count	UDInt
3	SourceIndex	DInt
4	DestinationIndex	DInt
5	Output	
6	InOut	
7	SourceArray	Variant
8	DestinationArray	Variant
9	Temp	
10	Error	Int
11	Constant	
12	<Add new>	
13	Return	
14	FC_PartialArrayCopy	Int

3. Create the SCL program code as follows:
   You can find the program code below as template.

```
IF IS_ARRAY(#SourceArray) AND IS_ARRAY(#DestinationArray)
    AND TypeOfElements(#SourceArray) = TypeOfElements(#DestinationArray) THEN
    #Error := MOVE_BLK_VARIANT(COUNT := #Count, SRC := #SourceArray, SRC_INDEX := #SourceIndex,
                               DEST => #DestinationArray, DEST_INDEX := #DestinationIndex);
END_IF;
#FC_PartialArrayCopy := #Error;
```

4. Create the PLC data type "UDT_MyStruct":

UDT_MyStruct

	Name	Data type	Default value
1	a	Bool	1
2	b	Bool	false
3	c	Bool	1
4	d	Bool	false
5	e	Bool	1
6	arr_int	Array[1..100] of Int	
7	f	Int	0
8	g	Real	0.0
9	h	LReal	0.0

5. Create the global data block "DB_WithArrays":

DB_WithArrays

	Name	Data type
1	Static	
2	first_arr	Array[0..100] of "UDT_MyStruct"
3	Sec_arr	Array[1..200] of "UDT_MyStruct"
4	Thrd_arr	Array[1..200] of Real
5	For_arr	Array[1..500] of Real

6. Call the "FC_PartialArrayCopy" function in an organization block, for example OB1, and initialize the parameters with the DB_WithArrays data block. Enter the named constants:

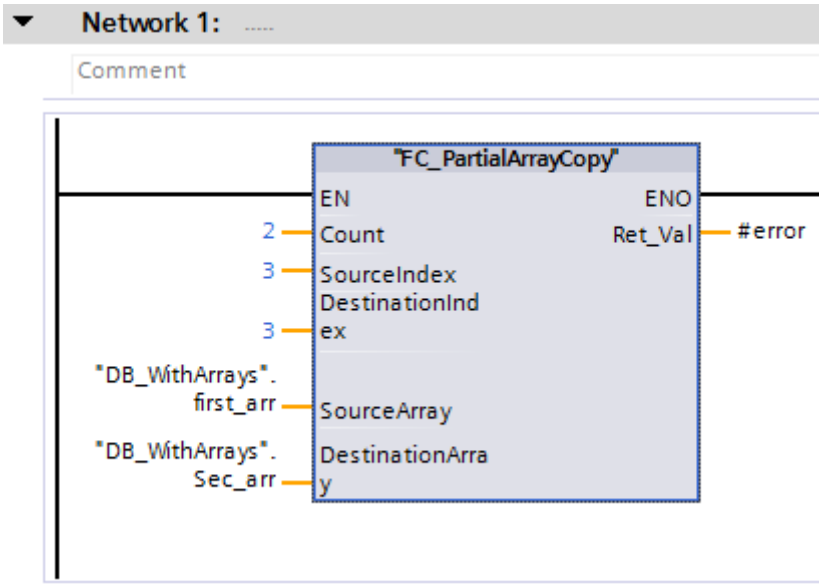

7. Instead of using the first two ARRAYs, which have the data type UDT_MyStruct, you can also use the third and fourth ARRAY, which have the data type REAL:

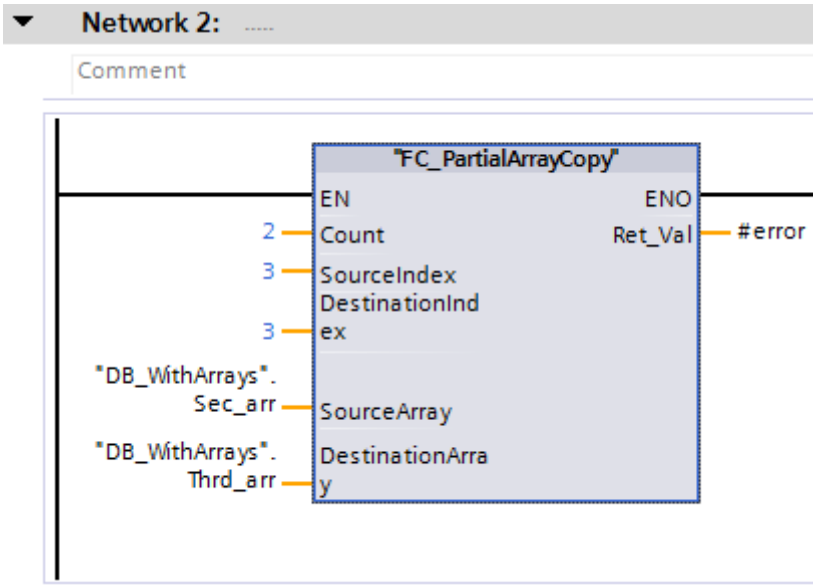

Result

As soon as the "FC_PartialArrayCopy" block is called in the program cycle, two data values, starting with the fourth element, are copied from the first ARRAY of the "DB_WithArrays" global data block to the second ARRAY of the data block. The copied data values are inserted in the second ARRAY, starting from the fourth element.

SCL program code for copying:

SCL

```
IF IS_ARRAY(#SourceArray) AND TypeOfElements(#SourceArray) =
TypeOfElements(#DestinationArray) THEN
#Error := MOVE_BLK_VARIANT(COUNT := #Count, SRC := #SourceArray, SRC_INDEX := #SourceIndex,
                           DEST => #DestinationArray, DEST_INDEX := #DestinationIndex);
END_IF;
#FC_PartialArrayCopy := #Error;
```

See also

Basic information on VARIANT (Page 324)

Example of programming a queue (FIFO) (S7-1200, S7-1500)

Programming example

In the following example, you program a ring buffer which consists of an ARRAY and is written and read according to the FIFO principle. The program code has a read-VARIANT pointer and a write-VARIANT pointer. Using the VARIANT instructions, you can program the program code robustly and ensure reliable copying or deleting.

Using the VARIANT data type, program sections can be influenced during runtime. The VARIANT pointer is a type-safe pointer, i.e. a type test is performed during runtime. For blocks that have been created with the block property "optimized", sub-functions that were previously programmed with an ANY pointer can now be resolved with a VARIANT pointer. The VARIANT data type is used to transfer structures to system function blocks.

Procedure

1. Create an SCL function block and name it "FIFOQueue".
2. Declare the block interface as follows:

Declaration	Parameters	Data type	Comment
Input	request	BOOL	The instruction is executed when a positive signal edge is detected at the "request" parameter.
	mode	BOOL	0 = The first entry in the ring buffer is returned. 1 = An entry is written at the last position in the ring buffer.
	initialValue	VARIANT	Value with which the ARRAY of the ring buffer is initialized.
Output	error	INT	Error information
InOut	item	VARIANT	The entry that is either returned from the ring buffer or written to the ring buffer.
	buffer	VARIANT	An ARRAY that is used as a ring buffer.
Static	edgeupm	BOOL	Edge memory bit in which the RLO of the previous query is saved.
	firstItemIndex	INT	Index of the oldest entry in the ring buffer
	nextEmptyItemIndex	INT	Index of the next free entry in the ring buffer
Temp	edgeup	BOOL	Result of edge evaluation
	internalError	INT	Error information
	newFirstItemIndex	INT	Variable index
	newNextEmptyItemIndex	INT	Variable index
	bufferSize	UDINT	Number of ARRAY elements in the ring buffer

3. Create the following program code in the "FIFOQueue" function block:

```
(* This program code section is only executed once after a positive signal
edge. If there is no change in the signal state of the result of logic
operation, the program processing of the "FIFOQueue" FB is terminated. *)
#edgeup := #request & NOT #edgeupm;
#edgeupm := #request;
IF NOT (#edgeup) THEN
   RETURN;
END_IF;
```

```
// ------Validation of whether all parameter inputs are valid.----
(* This program code section checks whether the ring buffer is an ARRAY. If
so, the number of the ARRAY elements is read out. If it is not an ARRAY,
the program execution is terminated at this point and the error code "-10"
is output. *)
IF NOT (IS_ARRAY(#buffer)) THEN
   #error := -10;
   RETURN;
ELSE
   #bufferSize := CountofElements(#buffer);
END_IF;

(* This program code section checks whether the data type of the ARRAY
elements matches the data type of the entry (tag #item). If the data types
do not match, the program execution is terminated at this point and the
error code "-11" is output. *)
IF NOT (TypeOf(#item) = TypeOfElements(#buffer)) THEN
   #error := -11;
   RETURN;
END_IF;

(* This program code section checks whether the initial value of the ring
buffer matches the entry (tag #item). If the data types do not match, the
program execution is terminated at this point and the error code "-12" is
output. *)
IF NOT (TypeOf(#item) = TypeOf(#initialValue)) THEN
   #error := -12;
   RETURN;
END_IF;

(* This program code section checks whether the variable indices are within
the ARRAY limits. If this is not the case, the program execution is
terminated at this point and either error code "-20" or "-21" is output
depending on the index. *)
IF (#nextEmptyItemIndex >= #bufferSize) THEN
   #error := -20;
   RETURN;
END_IF;
IF (#firstItemIndex >= #bufferSize) THEN
   #error := -21;
   RETURN;
END_IF;
```

```
//-----------Program code execution, depending on the Mode
parameter-------------
// The execution of the instructions depends on the signal state of the Mode
parameter.
IF #mode = 0 THEN

// If the Mode parameter has the signal state "0", the first entry from the
passed ring buffer is returned.
(* This program code section checks whether the ring buffer is empty. If
this is the case, program execution is terminated at this point and the
error code "-40" is output. *)
   IF (#firstItemIndex = -1) THEN
      #error := -40;
      RETURN;
   END_IF;

// This program code section returns the first entry of the ring buffer.
   #internalError := MOVE_BLK_VARIANT(SRC := #buffer,
                                      COUNT := 1,
                                      SRC_INDEX := #firstItemIndex,
                                      DEST_INDEX := 0,
                                      DEST => #item);

   IF (#internalError = 0) THEN
(* This program code section checks whether the ring buffer contains ARRAY
elements. If it does, the first entry is passed on and the index is
incremented by 1. *)
      #internalError := MOVE_BLK_VARIANT(SRC := #initialValue,
                                         COUNT := 1,
                                         SRC_INDEX := 0,
                                         DEST_INDEX := #firstItemIndex,
                                         DEST => #buffer);

// This program code section calculates the new index of the first entry.
      #newFirstItemIndex := #firstItemIndex +1;
      #newFirstItemIndex := #newFirstItemIndex MOD
UDINT_TO_INT(#bufferSize);

// This program section checks whether the ring buffer is empty.
      IF (#nextEmptyItemIndex = #newFirstItemIndex) THEN
// If the ring buffer is empty, the index is set to 0.
         #firstItemIndex := -1;
         #nextEmptyItemIndex := 0;
      ELSE
// The index of the first entry is changed.
         #firstItemIndex := #newFirstItemIndex;
```

```
      END_IF;
   END_IF;
ELSE


// If the Mode parameter has the signal state "1", the entry is written to
the passed ring buffer.
(* This program code section checks whether the ring buffer is full. If this
is the case, program execution is terminated at this point and the error
code "-50" is output. *)
   IF (#nextEmptyItemIndex = #firstItemIndex) THEN
      #error := -50;
      RETURN;
   END_IF;

// This program code section writes the entry to the ring buffer.
   #internalError := MOVE_BLK_VARIANT(SRC := #item,
                                      COUNT := 1,
                                      SRC_INDEX := 0,
                                      DEST_INDEX := #nextEmptyItemIndex,
                                      DEST => #buffer);

   IF (#internalError = 0) THEN
// This program code section increments the index by 1 and calculates the
new empty entry index.
      #newNextEmptyItemIndex := #nextEmptyItemIndex +1;
      #newNextEmptyItemIndex := #newNextEmptyItemIndex MOD #bufferSize;
      #nextEmptyItemIndex := #newNextEmptyItemIndex;

(* This program code section checks which index the "#firstItemIndex" tag
has. If the number = -1, the ring buffer is initialized and the entry is
written to the ring buffer. This is why the tag value "0" must be assigned.
*)
      IF (#firstItemIndex = -1) THEN
         #firstItemIndex := 0;
      END_IF;
   END_IF;
END_IF;

//-------------------------Local error handling----------------------------
(* This program code section checks whether a local error has occurred. If
this is the case, program execution is terminated at this point and the
error code "-100" is output. *)
IF (#internalError > 0) THEN
   #error := -100;
   RETURN;
```

```
END_IF;

// If no error occurred during program execution, the error code "0" is
output.
#error := 0;
```

Result

Call the SCL function block at the position in your program at which the FIFO queue is to run.

3.11.3 POINTER (S7-300, S7-400, S7-1500)

Description

A parameter of the type POINTER is a pointer that can point to a specific tag. It occupies 6 bytes (48 bits) in memory and may contain the following tag information:

- DB number, or 0 if the data is not stored in a DB
- Memory area in the CPU
- Tag address

The following figure shows the structure of parameter type POINTER:

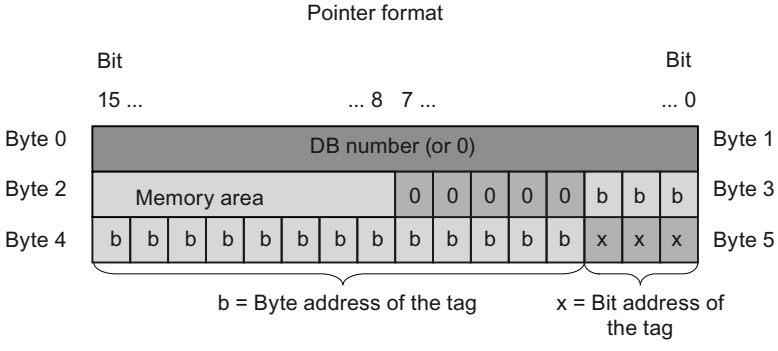

Types of pointer

Depending on the information, you can use parameter type POINTER to declare the following four types of pointer:

- Area-internal pointer:
  An area-internal pointer contains information on the address of a tag.
- Cross-area pointer:
  A cross-area pointer contains information on the memory area and the address of a tag.

- DB pointer:
  You can use a DB pointer to point to a data block tag. A DB pointer also contains a data block number in addition to the memory area and the address of a tag.
- Zero pointer:
  Use the zero pointer to indicate a missing value. A missing value may indicate that no value exists, or that the value is not yet known. A zero value represents the absence of a value, but is also a value.

The following table shows the formats for the declaration of various pointer types:

P#ByteRepresentation	Format	Example of value input	Description
Symbolic	P#Byte.Bit	"MyTag"	Area-internal pointer
	P#OperandAreaByte.Bit	"MyTag"	Cross-area pointer
	P#Data_block.Data_operand	"MyDB"."MyTag"	DB pointer
	P#Zero value	-	Zero pointer
Absolute	P#Byte.Bit	P#20.0	Area-internal pointer
	P#OperandAreaByte.Bit	P#M20.0	Cross-area pointer
	P#Data_block.Data_operand	P#DB10.DBX20.0	DB pointer
	P#Zero value	P#0.0, ZERO	Zero pointer

Note

SCL: Access to data blocks via absolute DB pointers

If you address a data block from the "System blocks" folder in SCL via an absolute DB pointer (e.g. P#DB1.dbx0.0) and this is the only access to the DB, in rare cases the data block will be deleted on the next compilation. The compiler does not recognize the DB pointer as an access and therefore assumes that the DB is not used in the program. Use a read access to the DB (e.g. "MyDB.MyFirstMember") instead of a pointer. This form of access is always performed correctly.

The prefix P#

Enter the actual value without prefix P# in the block call to supply a formal parameter of the POINTER data type. The entry is then automatically converted to the POINTER format.

If you use the prefix P#, you can only point to memory areas with "Standard" access mode. You can find additional information and a direct comparison of the two access options here:

- Basics of block access (Page 55)
- The new S7-1200/1500 CPU functions and programming recommendations at a glance (Page 193)

The following should be noted when using the prefix P# in the STL programming language:

Block type	"Optimized" access mode	"Standard" access mode
Function (FC)	A tag cannot be used in the program code with the prefix P#.	The following tags can be used in the program code with the prefix P#. • Structured tags that are declared in the InOut section of the block interface. Tags with elementary data types cannot be used with the prefix P#.
Function block (FB)	A tag cannot be used in the program code with the prefix P#.	The following tags can be used in the program code with the prefix P#. • Tags that have been declared in the sections Static, Input and Output of the block interface. • Tags with the PLC data type (UDT) that are declared in the InOut section of the block interface.

You can find additional information about structured and elementary data types here: Transfer parameter as copy or as pointer (Page 80)

Memory areas

The following table shows the hexadecimal codes of the memory areas for parameter type POINTER:

Hexadecimal code	Memory area	Description
B#16#80[1)]	P	Peripherals on a CPU S7-300/400
16#1	P	Peripheral inputs on a CPU S7-1500
16#2	P	Peripheral outputs on a CPU S7-1500
B#16#81	I	Memory area of inputs
B#16#82	Q	Memory area of outputs
B#16#83	M	Memory area of bit memory
B#16#84	DBX	Data block
B#16#85	DIX	Instance data block
B#16#86	L	Local data
B#16#87	V	Previous local data
[1)] These data types can only be used for the POINTER pointer on a CPU S7-300/400.		

See also

Basics of indirect addressing (Page 114)

Overview of the valid data types (Page 231)

Basics of constants (Page 97)

3.11.4 ANY (S7-300, S7-400, S7-1500)

Description

An ANY type parameter points to the start of a data area and specifies its length. An ANY pointer occupies 10 bytes of memory and may contain the following information:

- Data type:
  Data type of the elements of the data area
- Repetition factor:
  Number of elements of the data area
- DB number:
  Data block that contains the declaration of data area elements.
- Memory area:
  Memory area of the CPU that stores the data area elements.
- Start address of the data in the format "byte.bit":
  Data area start identified by the ANY pointer.
- Zero pointer:
  Use the zero pointer to indicate a missing value. A missing value may indicate that no value exists, or that the value is not yet known. A zero value represents the absence of a value, but is also a value.

Note

ANY pointer to a STRING

If an ANY pointer is generated in an LAD/FBD/STL or SCL network through the assignment of a string, an ANY pointer of the data type BYTE and the repetition factor maxstringlen is created.

If an ANY pointer is generated in an SCL block through the assignment of a string, an ANY pointer to a STRING is created.

A pointer of this type can, for instance, be created by transferring it from an SCL block to an LAD network or by superimposing it with an AT construct and manually setting the data type to STRING.

Note

Access options

For a CPU of the S7-1500 series, the ANY pointer can also only point to memory areas with "Standard" access mode. You can find additional information and a direct comparison of the two access options here:

- Basics of block access (Page 55)
- The new S7-1200/1500 CPU functions and programming recommendations at a glance (Page 193)

In the programming languages SCL and STL, memory of any kind can be transferred upon a block call if an ANY pointer has been programmed at a block parameter.

The ANY pointer cannot, however, store any information on the structure of the memory. For example, the ANY pointer does not save the information that it points to a tag of the PLC data type. The ANY pointer sees this as an ARRAY of BYTE.

Parameters of the ANY data type can be passed to system function blocks (SFBs) or system functions (SFCs).

The following figure shows the structure of the ANY pointer:

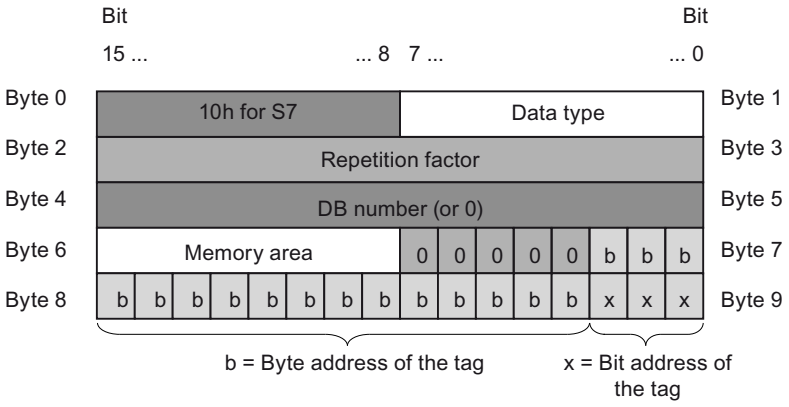

An ANY pointer cannot identify structures. It can only be assigned to local tags.

The following table shows the formats for the declaration of an ANY pointer:

Representation	Format	Example of value input	Description
Symbolic	P#DataBlock.MemoryArea DataAddress Type Number	"MyDB".StructTag.InitialComponents	Area with 10 words in global DB11 starting with DBB20.0
	P#MemoryArea DataAddress Type Number	"MyMarkerTag"	Area with 4 bytes starting with MB 20.0
		"MyTag"	Input I1.0
	P#Zero value	-	Zero value
Absolute	P#DataBlock.MemoryArea DataAddress Type Number	P#DB11.DBX20.0 INT 10	Area with 10 words in global DB11 starting with DBB20.0
	P#MemoryArea DataAddress Type Number	P#M20.0 BYTE 10	Area with 10 bytes starting with MB 20.0
		P#I1.0 BOOL 8	Range with 8 bits from input I1.0 (the range length specified must be divisible by 8.)
	P#Zero value	P#P0.0 VOID 0, NULL [1]	Zero value

[1] In the programming languages LAD and FBD, only NULL is a value entry for the zero value.

Coding of data types

The following table lists the coding of data types for the ANY pointer:

Hexadecimal code	Data type	Description
B#16#00	NIL	Null pointer
B#16#01[1)]	BOOL	Bits
B#16#02	BYTE	bytes, 8 bits
B#16#03	CHAR	8-bit characters
B#16#04	WORD	16-bit words
B#16#05	INT	16-bit integers
B#16#06	DWORD	32-bit words
B#16#07	DINT	32-bit integers
B#16#08	REAL	32-bit floating-point numbers
B#16#0B	TIME	Time duration
B#16#0C	S5TIME	Time duration
B#16#09	DATE	Date
B#16#0A	TOD	Date and time
B#16#0E	DT	Date and time
B#16#13	STRING	Character string
B#16#17[1)]	BLOCK_FB	Function block
B#16#18[1)]	BLOCK_FC	Function
B#16#19[1)]	BLOCK_DB	Data block
B#16#1A[1)]	BLOCK_SDB	System data block
B#16#1C[1)]	COUNTER	Counter
B#16#1D[1)]	TIMER	Timer
[1)] These data types can only be used for the ANY pointer on a CPU S7-300/400.		

Coding of the memory area

The following table lists the coding of the memory areas for the ANY pointer:

Hexadecimal code	Area	Description
B#16#80[1)]	P	I/O
B#16#81	I	Memory area of inputs
B#16#82	Q	Memory area of outputs
B#16#83	M	Memory area of bit memory
B#16#84	DBX	Data block
B#16#85[1)]	DIX	Instance data block
B#16#86	L	Local data
B#16#87	V	Previous local data
[1)] These memory areas can only be used for the ANY pointer on an S7-300/400 CPU.		

See also

Basics of indirect addressing (Page 114)
Overview of the valid data types (Page 231)
Basics of constants (Page 97)
Transfer parameter as copy or as pointer (Page 80)

3.12 Parameter types

3.12.1 Parameter types

Description

The parameter types are data types for formal parameters that are transferred to called blocks. A parameter type can also be a PLC data type.

The following table shows the available parameter data types and their purpose:

Parameter type	Length (bits)	Description
TIMER	16	Is used to specify a timer that is used in the called code block. If you supply a formal parameter of the TIMER parameter type, the associated actual parameter must be a timer. Example: T1
COUNTER	16	Is used to specify a counter that is used in the called code block. If you supply a formal parameter of the COUNTER parameter type, the associated actual parameter must be a counter. Example: C10
BLOCK_FC	16	Is used to specify a block that is used as input in the called code block. The declaration of the parameter determines the block type (for example FB, FC, DB) that is to be used. If you supply a formal parameter of the BLOCK parameter type, specify a block address as the actual parameter. Example: DB3
BLOCK_FB	16	
BLOCK_DB	16	
BLOCK_SDB	16	

Note

Parameter type "Block_DB" to transfer an instance DB

In LAD and FBD, you cannot enter the instance DB of a function block using an input of the data type "BLOCK_DB". Use a parameter instance instead.

See also: Parameter instances (Page 68)

See also

Overview of the valid data types (Page 231)

3.13 System data types

3.13.1 System data types

Description

The system data types (SDT) are made available by the system and have a predefined structure. The structure of a system data type consists of a fixed number of components that can have various data types. It is not possible to change the structure of a system data type.

System data types can only be assigned to one another if they are of the same type and their name matches. This also applies to PLC data types generated by the system, such as IEC_Timer, etc.

The system data types can only be used for specific instructions. The following table shows the available system data types and their purpose:

System data type	Length (bytes)	Description
IEC_TIMER	16	Structure of a timer in which the PT, ET, IN and Q parameters are declared. The time values are of TIME data type. This data type is used for the "TP", "TOF", "TON", "TONR", "RT" and "PT" instructions, for example.
IEC_LTIMER	32	Structure of a timer in which the PT, ET, IN and Q parameters are declared. The time values are of LTIME data type. This data type is used for the "TP", "TOF", "TON", "TONR", "RT" and "PT" instructions, for example.
IEC_SCOUNTER	3	Structure of a counter whose count values are of SINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_USCOUNTER	3	Structure of a counter whose count values are of USINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_COUNTER	6	Structure of a counter whose count values are of INT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_UCOUNTER	6	Structure of a counter whose count values are of UINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_DCOUNTER	12	Structure of a counter whose count values are of DINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.

System data type	Length (bytes)	Description
IEC_UDCOUNTER	12	Structure of a counter whose count values are of UDINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_LCOUNTER	24	Structure of a counter with count values of data type UDINT. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_ULCOUNTER	24	Structure of a counter with count values of data type UINT. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
ERROR_STRUCT	28	Structure of an error information to a programming or I/O access error. This data type is used, for example, for the "GET_ERROR" instruction.
CREF	8	Components of the ERROR_STRUCT data type, in which information about the address of a block is saved.
NREF	8	Components of the ERROR_STRUCT data type, in which information about the address of an operand is saved.
VREF	12	Is used for storage of a VARIANT pointer. This data type is, for example, used for instructions from S7-1200/1500 Motion Control.
SSL_HEADER	4	Specifies the data structure in which information about the data records are saved during the reading of the system status lists. This data type is used, for example, for the "RDSYSST" instruction.
CONDITIONS	52	User-defined data structure defining the conditions for start and end of a data reception. This data type is used, for example, for the "RCV_CFG" instruction.
TADDR_Param	8	Specifies the structure of a data block which stores descriptions of connections for Open User Communication via UDP. This data type is used for the "TUSEND" and "TURSV" instructions, for example.
TCON_Param	64	Specifies the structure of a data block which stores descriptions of connections for Open User Communication via Industrial Ethernet (PROFINET). This data type is used for the "TSEND" and "TRSV" instructions, for example.
HSC_Period	12	Specifies the structure of the data block for the time period measurement with the extended high-speed counter. This data type is used, for example, for the "CTRL_HSC_EXT" instruction.

See also

Overview of the valid data types (Page 231)

3.14 Hardware data types

3.14.1 Hardware data types

Description

The hardware data types are made available by the CPU. The number of available hardware data types depends on the CPU.

Constants of a specific hardware data type are stored based on the modules set in the hardware configuration. When an instruction for controlling or activating a configured module is inserted in the user program, the available constants can be used for the parameters.

The following table shows the available hardware data types and their purpose:

Data type	Basic data type	Description
REMOTE	ANY	Serves to specify the address of a remote CPU. This data type is used, for example, for the "PUT" and "GET" instructions.
HW_ANY	UINT	Identification of any hardware component, e.g. a module.
HW_DEVICE	HW_ANY	Identification of a DP slave/PROFINET IO device
HW_DPMASTER	HW_INTERFACE	Identification of a DP master
HW_DPSLAVE	HW_DEVICE	Identification of a DP slave
HW_IO	HW_ANY	Identification number of the CPU or the interface The number is automatically allocated and is stored in the properties of the CPU or of the interface in the hardware configuration.
HW_IOSYSTEM	HW_ANY	Identification of a PN/IO system or DP master system
HW_SUBMODULE	HW_IO	Identification of a central hardware component
HW_MODULE	HW_IO	Identification of a module
HW_INTERFACE	HW_SUBMODULE	Identification of an interface component
HW_IEPORT	HW_SUBMODULE	Identification of a port (PN/IO)
HW_HSC	HW_SUBMODULE	Identification of a high-speed counter This data type is used, for example, for the "CTRL_HSC" and "CTRL_HSC_EXT" instructions.
HW_PWM	HW_SUBMODULE	Identification of a pulse width modulation This data type is used, for example, for the "CTRL_PWM" instruction.
HW_PTO	HW_SUBMODULE	Identification of a pulse encoder This data type is used for Motion Control.
EVENT_ANY	AOM_IDENT	Used to identify any event
EVENT_ATT	EVENT_ANY	Is used to specify an event that can be assigned dynamically to an OB This data type is used, for example, for the "ATTACH" and "DETACH" instructions.

Data type	Basic data type	Description
EVENT_HWINT	EVENT_ATT	Is used to specify a hardware interrupt event
OB_ANY	INT	Serves to specify any organization block.
OB_DELAY	OB_ANY	Used to specify an organization block that is called when a time-delay interrupt occurs. This data type is used, for example, for the "SRT_DINT" and "CAN_DINT" instructions.
OB_TOD	OB_ANY	Specifies the number of a time-of-day interrupt OB. This data type is used, for example, for the "SET_TINT", "CAN_TINT", "ACT_TINT" and "QRY_TINT" instructions.
OB_CYCLIC	OB_ANY	Is used to specify an organization block that is called when a watchdog interrupt occurs.
OB_ATT	OB_ANY	Is used to specify an organization block that can be assigned dynamically to an event. This data type is used, for example, for the "ATTACH" and "DETACH" instructions.
OB_PCYCLE	OB_ANY	Is used to specify an organization block that can be assigned to an event of the "Cyclic program" event class.
OB_HWINT	OB_ATT	Is used to specify an organization block that is called when a hardware interrupt occurs.
OB_DIAG	OB_ANY	Is used to specify an organization block that is called when a diagnostic interrupt occurs.
OB_TIMEERROR	OB_ANY	Is used to specify an organization block that is called when a time error occurs.
OB_STARTUP	OB_ANY	Is used to specify an organization block that is called when a startup event occurs.
PORT	HW_SUBMODULE	Serves to specify a communication port. This data type is used for point-to-point communication.
RTM	UINT	Serves to specify the number of an operating hours counter. This data type is used, for example, for the "RTM" instruction.
PIP	UINT	Is used to create and connect a "Synchronous Cycle" OB. This data type is used for the SFCs 26, 27, 126 and 127.
CONN_ANY	WORD	Serves to specify any connection.
CONN_PRG	CONN_ANY	Serves to specify a connection for open communication over UDP.
CONN_OUC	CONN_ANY	Used to specify a connection for open communication over Industrial Ethernet (PROFINET).
CONN_R_ID	DWORD	Data type for the R_ID parameter on the S7 communication blocks.
DB_ANY	UINT	Identification (name or number) of a DB The data type "DB_ANY" has the length 0 in the section "Temp".

Data type	Basic data type	Description
DB_WWW	DB_ANY	Number of a DB generated via the Web application (for example, "WWW" instruction) The data type "DB_WWW" has the length 0 in the section "Temp".
DB_DYN	DB_ANY	Number of a DB generated by the user program
C_ALARM_SD	DWORD	Is used for recording the message number. This data type can only be used with the CPU S7-410H.

See also

Overview of the valid data types (Page 231)

3.15 Data type conversion for S7-1500: (S7-1500)

3.15.1 Overview of data type conversion (S7-1500)

Introduction

If you combine several operands in an instruction, you must make sure that the data types are compatible. This also applies when you assign or supply block parameters. If the operands are not of the same data type, a conversion has to be carried out.

There are two options for conversion:

- Implicit conversion
  Implicit conversion is supported by the programming languages LAD, FBD, SCL and GRAPH. Implicit conversion is not possible in the STL programming language.
- Explicit conversion
  You use an explicit conversion instruction before the actual instruction is executed.

Note

Converting bit strings to SCL

All bit strings (BYTE, WORD, DWORD and LWORD) are handled like the corresponding unsigned integers (USINT, UINT, UDINT and ULINT) in expressions. Therefore, implicit conversion from DWORD to REAL is carried out like a conversion from UDINT to REAL, for example.

Note

Converting REF()

A data type conversion of REF() is not possible. The transferred tag must correspond exactly to the target data type.

Implicit conversion

Implicit conversion is executed automatically if the data types of the operands are compatible. This compatibility test can be carried out according to strict or less strict criteria:

- With IEC check (default setting)
  If IEC check is set, the following rules are applied:
  - Implicit conversion of BOOL into other data types is not possible.
  - Only the data types REAL, BYTE, WORD, DWORD, DINT, INT, SINT, UDINT, UINT, USINT, TIME, LDT, DTL, DT, TOD, WCHAR and CHAR can be converted implicitly.
  - The bit length of the source data type may not exceed the bit length of the target data type. A WORD data type operand, for example, cannot be specified at a parameter at which the BYTE data type is expected.
- Without IEC check
  If IEC check is not set, the following rules are applied:
  - Implicit conversion of BOOL into other data types is not possible.
  - Only the data types REAL, LREAL, BYTE, WORD, DWORD, LWORD, SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, TIME, LTIME, S5TIME, LDT, DTL, TOD, LTOD, DATE, STRING, WSTRING, WCHAR and CHAR can be converted implicitly.
  - The bit length of the source data type may not exceed the bit length of the target data type. A DWORD data type operand, for example, cannot be specified at a parameter at which the WORD data type is expected.
  - The bit length of an operand entered at in-out parameters (InOut) must be the same as the programmed bit length for the parameter in question.

  Note

  Implicit conversion without IEC check

  The programming editor uses a gray rectangle to mark operands that are implicitly converted. The dark gray rectangle signals that an implicit conversion is possible without any accuracy loss, for example, if you convert the data type INT to DINT. A light gray rectangle signals that implicit conversion is possible, but errors could occur during runtime. If, for example, you are converting the data type LINT to DINT and an overflow occurs, the enable output ENO is set to "0".

For more information about the setting of the IEC check and the implicit conversion, refer to "See also".

Explicit conversion

If the operands are not compatible and implicit conversion is therefore not possible, you can perform an explicit conversion. You can do this by using the conversion instructions in the "Instructions" Task Card or you can insert the conversion into the program manually. You can find the format for explicit conversion functions under "See also".

Any overflow will be displayed on the ENO enable output. An overflow occurs, for example, if the value of the source data type is greater than the value of the target data type.

Note

Shifting of bit patterns

If the explicit conversion involves shifting a bit pattern, the enable output ENO is not set.

You can find additional information on explicit conversion under "See also".

The following figure shows an example in which explicit data type conversion must be carried out:

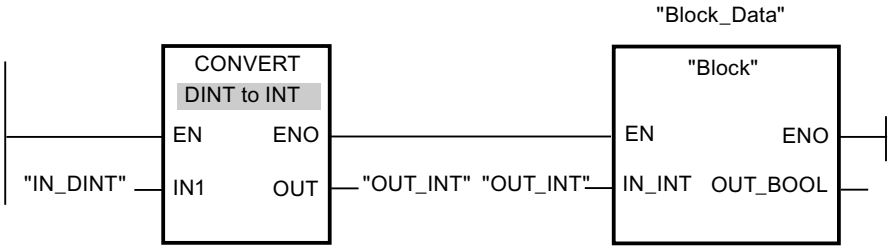

The "Block" function block expects a tag of the INT data type at the "IN_INT" input parameter. The value of the "IN_DINT" tag has therefore to the converted first from DINT to INT. Conversion is performed if the value of the "IN_DINT" tag is within the admissible value range for the INT data type. Otherwise an overflow is reported. However, a conversion takes place even in the event of an overflow, but the values are truncated and the ENO enable output is set to "0".

See also

Implicit conversions (Page 356)

Explicit conversions (Page 405)

3.15.2 Implicit conversions (S7-1500)

3.15.2.1 Activate or deactivate IEC check (S7-1500)

The data types of the operands used are checked for compatibility. This compatibility test can be carried out according to criteria that are more or less strict. If "IEC check for code blocks" is activated, stricter criteria are applied.

You can set the IEC check centrally for all new blocks of the project or for individual blocks.

Setting IEC check for new blocks

To set the IEC check for all new blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. Select or clear the "IEC check for code blocks" check box in the "Default settings for new blocks" group.
   The IEC check is enabled or disabled for all new blocks in the program.

Setting IEC check for a block

To set the IEC check for a block, proceed as follows:

1. Open the block.
2. Open the "Properties" tab in the inspector window.
3. Select the "Attributes" group in the area navigation.
4. Select or clear the "IEC check" check box.
   The IEC check is enabled or disabled for this block. The setting is stored together with the project.

3.15.2.2 Binary numbers (S7-1500)

Implicit conversion of BOOL (S7-1500)

Implicit conversion options

The BOOL data type cannot be implicitly converted.

See also

BOOL (bit) (Page 236)

Overview of data type conversion (Page 354)

Activate or deactivate IEC check (Page 356)

Explicit conversion of BOOL (Page 405)

Bit strings (S7-1500)

Implicit conversion of BYTE (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of BYTE data type:

Source	Destination	With IEC check	Without IEC check	Explanation
BYTE	BOOL	-	-	No implicit conversion
	WORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	DWORD	X	X	
	LWORD	X	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

BYTE (Page 236)

Activate or deactivate IEC check (Page 356)

Overview of data type conversion (Page 354)
Explicit conversion of BYTE (Page 407)

Implicit conversion of WORD (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of WORD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
WORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The low byte is transferred to the destination data type; the high byte is ignored.
	DWORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LWORD	X	X	
	SINT	-	X	The low byte is transferred to the destination data type; the high byte is ignored.
	USINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LDT	-	-	No implicit conversion
	DTL	-	-	
	DT	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	TOD	-	-	No implicit conversion
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

WORD (Page 237)
Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
Explicit conversion of WORD (Page 410)

Implicit conversion of DWORD (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of DWORD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The right byte is transferred to the destination data type; the left byte is ignored.
	WORD	-	X	
	LWORD	X	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	-	-	No implicit conversion
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

DWORD (Page 238)

Activate or deactivate IEC check (Page 356)

Overview of data type conversion (Page 354)
Explicit conversion of DWORD (Page 413)

Implicit conversion of LWORD (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of LWORD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The right byte is transferred to the destination data type; the left byte is ignored.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	-	No implicit conversion
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	S5TIME	-	-	No implicit conversion
	LDT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
LWORD (Page 240)
Explicit conversion of LWORD (Page 417)

3.15.2.3 Integers (S7-1500)

Implicit conversion of SINT (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of SINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
SINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	USINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value transfer from SINT #-1 -> INT #-1, not filled with "0".)
	INT	X	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	LINT	X	X	
	ULINT	-	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)

Overview of data type conversion (Page 354)

SINT (8-bit integers) (Page 241)
Explicit conversion of SINT (Page 421)

Implicit conversion of USINT (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of USINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
USINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from USINT #10 -> DINT #10 or USINT #128 -> SINT #-128)
	INT	X	X	
	UINT	X	X	
	DINT	X	X	
	UDINT	X	X	
	LINT	X	X	
	ULINT	X	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
USINT (8-bit integers) (Page 242)
Explicit conversion of USINT (Page 425)

Implicit conversion of INT (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of INT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
INT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from INT #-1 -> SINT #-1, or INT #-32 767 -> UINT #32 769)
	USINT	-	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	LINT	X	X	
	ULINT	-	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	TOD	-	-	No implicit conversion
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)

Overview of data type conversion (Page 354)

INT (16-bit integers) (Page 243)
Explicit conversion of INT (Page 428)

Implicit conversion of UINT (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of UINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
UINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from UINT #100 -> DINT #100 or UINT #60 000 -> INT #-5536)
	USINT	-	X	
	INT	-	X	
	DINT	X	X	
	UDINT	X	X	
	LINT	X	X	
	ULINT	X	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	TOD	-	-	No implicit conversion
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
UINT (16-bit integers) (Page 244)
Explicit conversion of UINT (Page 432)

Implicit conversion of DINT (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of DINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	UDINT	-	X	
	LINT	X	X	
	ULINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
DINT (32-bit integers) (Page 245)
Explicit conversion of DINT (Page 435)

Implicit conversion of UDINT (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of UDINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
UDINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	LINT	X	X	
	ULINT	X	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
UDINT (32-bit integers) (Page 246)
Explicit conversion of UDINT (Page 439)

Implicit conversion of LINT (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of LINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> SINT #-1 or LINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	ULINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 9 007 199 254 740 992)
	TIME	-	-	No implicit conversion
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (duration in nanoseconds)
	S5TIME	-	-	No implicit conversion
	LDT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
LINT (64-bit integers) (Page 247)
Explicit conversion of LINT (Page 442)

Implicit conversion of ULINT (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of ULINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
ULINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from ULINT #-1 -> SINT #-1 or ULINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 9 007 199 254 740 992)
	TIME	-	-	No implicit conversion
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (duration in nanoseconds)
	S5TIME	-	-	No implicit conversion
	LDT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
ULINT (64-bit integers) (Page 248)
Explicit conversion of ULINT (Page 446)

3.15.2.4 Floating-point numbers (S7-1500)

Implicit conversion of REAL (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of REAL data type:

Source	Destination	With IEC check	Without IEC check	Explanation
REAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	LWORD	-	-	No implicit conversion
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the destination data type. (For example, rounding off and value conversion of REAL #2.5 -> INT #2, or negative numbers REAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined REAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	LREAL	X	X	The value is transferred to the destination data type.
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
REAL (Page 251)
Explicit conversion of REAL (Page 449)

Implicit conversion of LREAL (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of LREAL data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LREAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the destination data type. (For example, rounding off and value conversion of LREAL #2.5 -> INT #2, or negative numbers LREAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined LREAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	X	The value is transferred to the destination data type.
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)

Overview of data type conversion (Page 354)

LREAL (Page 252)
Explicit conversion of LREAL (Page 452)

3.15.2.5 Timers (S7-1500)

Implicit conversion of S5TIME (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of data type S5TIME:

Source	Destination	With IEC check	Without IEC check	Explanation
S5TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	DWORD	-	-	No implicit conversion
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
S5TIME (duration) (Page 255)
Explicit conversion of S5TIME (Page 455)

Implicit conversion of TIME (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of TIME data type:

Source	Destination	With IEC check	Without IEC check	Explanation
TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	LWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	UDINT	-	X	
	LINT	-	-	No implicit conversion
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	S5TIME	-	-	
	LTIME	X	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds. (1 ms = 1 000 000 ns)
	LDT	-	-	No implicit conversion
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	If the source value is between 0 s and 84599.999 s, the bit pattern of the source value is transferred unchanged to the destination data type. (Display in nanoseconds) The target value remains unchanged otherwise. The result of the conversion shows the time that has passed since midnight.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
TIME (IEC time) (Page 256)
Explicit conversion of TIME (Page 457)

Implicit conversion of LTIME (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of LTIME data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LTIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds.
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds.
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	S5TIME	-	-	
	TIME	-	X	If the source value is outside the value range of the destination data type, the target value remains unchanged. (0.123456789 s becomes 0.123 s)
	LDT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds since 1/1/1970.
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds since 1/1/1970.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
LTIME (IEC time) (Page 257)
Explicit conversion of LTIME (Page 460)

3.15.2.6 Date and time-of-day (S7-1500)

Implicit conversion of DT (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of DT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	X	X	The source value is transferred unchanged with the same value to the destination data type. (12/24/2012 14:30 remains 12/24/2012 14:30)
	DTL	X	X	
	DATE	-	-	No implicit conversion
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
DATE_AND_TIME (date and time of day) (Page 259)
Explicit conversion of DT (Page 463)

Implicit conversion of LDT (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of LDT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LDT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	S5TIME	-	-	No implicit conversion
	DT	-	X	The bit pattern of the source value is converted with a loss in accuracy to the destination data type. (12/24/2012 12:34:56.123456789 becomes 12/24/2012 12:34:56.123)
	DTL	X	X	The source value is transferred unchanged with the same value to the destination data type. (12/24/2012 14:30 remains 12/24/2012 14:30)
	DATE	-	-	No implicit conversion
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
Explicit conversion of LDT (Page 466)
LDT (DATE_AND_LTIME) (Page 261)

Implicit conversion of DTL (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of DTL data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DTL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	X	X	The source value is transferred unchanged with the same value to the destination data type. (12/24/2012 14:30 remains 12/24/2012 14:30)
	DT	-	X	The bit pattern of the source value is converted with a loss in accuracy to the destination data type. (12/24/2012 12:34:56.123456789 becomes 12/24/2012 12:34:56.123)
	DATE	-	-	No implicit conversion
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
DTL (Page 261)
Explicit conversion of DTL (Page 469)

Implicit conversion of DATE (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of data type DATE:

Source	Destination	With IEC check	Without IEC check	Explanation
DATE	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion is equal to the number of days since 01/01/1990.
	DWORD	-	-	No implicit conversion
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion is equal to the number of days since 01/01/1990.
	UINT	-	X	
	DINT	-	-	No implicit conversion
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DT	-	-	
	DTL	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)

Overview of data type conversion (Page 354)

DATE (Page 258)
Explicit conversion of DATE (Page 472)

Implicit conversion of TOD (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of TOD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
TOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	LWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	UDINT	-	X	
	LINT	-	-	No implicit conversion
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DT	-	-	
	DTL	-	-	
	DATE	-	-	
	LTOD	X	X	The source value is transferred unchanged with the same value to the destination data type. (12:34:56.123 becomes 12:34:56.123000000)
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
TIME_OF_DAY (TOD) (Page 258)
Explicit conversion of TOD (Page 474)

Implicit conversion of LTOD (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of LTOD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LTOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of nanoseconds since the start of day (0:00 hrs).
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of nanoseconds since the start of day (0:00 hrs).
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of nanoseconds since the start of day (0:00 hrs).
	S5TIME	-	-	No implicit conversion
	LDT	-	-	
	DT	-	-	
	DTL	-	-	
	DATE	-	-	
	TOD	-	X	The source value is transferred unchanged with the same value rounded off to the destination data type. (12:34:56.123456789 becomes 12:34:56.123)
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
Explicit conversion of LTOD (Page 476)
LTOD (LTIME_OF_DAY) (Page 259)

3.15.2.7 String (S7-1500)

Implicit conversion of CHAR (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of CHAR data type:

Source	Destination	With IEC check	Without IEC check	Explanation
CHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	WCHAR	-	-	
	STRING	X	X	The STRING is shortened to length 1 and includes the character.
	WSTRING	-	-	No implicit conversion
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)

Overview of data type conversion (Page 354)

CHAR (Page 263)
Explicit conversion of CHAR (Page 479)

Implicit conversion of WCHAR (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of WCHAR data type:

Source	Destination	With IEC check	Without IEC check	Explanation
WCHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	CHAR	-	-	
	STRING	-	-	
	WSTRING	X	X	The WSTRING is shortened to length 1 and includes the character.
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
WCHAR (Page 263)
Explicit conversion of WCHAR (Page 481)

Implicit conversion of STRING (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of STRING data type:

Source	Destination	With IEC check	Without IEC check	Explanation
STRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	CHAR	-	X	The first character of the STRING is returned if the STRING includes one or more characters. Otherwise, the character is output with coding $00.
	WCHAR	-	-	No implicit conversion
	WSTRING	-	-	
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)

Overview of data type conversion (Page 354)

STRING (Page 264)
Explicit conversion of STRING (Page 483)

Implicit conversion of WSTRING (S7-1500)

Implicit conversion options

The following table shows the options for the implicit conversion of WSTRING data type:

Source	Destination	With IEC check	Without IEC check	Explanation
WSTRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	CHAR	-	-	
	WCHAR	-	X	The first character of the WSTRING is returned if the WSTRING includes one or more characters. Otherwise, the character is output with coding $0000.
	STRING	-	-	No implicit conversion
x: Conversion possible -: Conversion not possible				

See also

Activate or deactivate IEC check (Page 356)
Overview of data type conversion (Page 354)
WSTRING (Page 267)
Explicit conversion of WSTRING (Page 486)

3.15.3 Explicit conversions (S7-1500)

3.15.3.1 Binary numbers (S7-1500)

Explicit conversion of BOOL (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BOOL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
BOOL	BYTE	X	Only the LSB (Least Significant Bit) is set in the destination data type. The enable output ENO is always "1".	BOOL_TO_BYTE
	WORD	X		BOOL_TO_WORD
	DWORD	X		BOOL_TO_DWORD
	LWORD	X		BOOL_TO_LWORD
	SINT	X		BOOL_TO_SINT
	USINT	X		BOOL_TO_USINT
	INT	X		BOOL_TO_INT
	UINT	X		BOOL_TO_UINT
	DINT	X		BOOL_TO_DINT
	UDINT	X		BOOL_TO_UDINT
	LINT	X		BOOL_TO_LINT
	ULINT	X		BOOL_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	LTIME	-		-
	S5TIME	-		-
	LDT	-		-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

Overview of data type conversion (Page 354)
Implicit conversion of BOOL (Page 357)
BOOL (bit) (Page 236)

Bit strings (S7-1500)

Explicit conversion of BYTE (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BYTE data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
BYTE[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	BYTE_TO_BOOL
	WORD[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_WORD
	DWORD[1)]	X		BYTE_TO_DWORD
	LWORD[1)]	X		BYTE_TO_LWORD
	SINT	X		BYTE_TO_SINT
	USINT	X		BYTE_TO_USINT
	INT	X		BYTE_TO_INT
	UINT	X		BYTE_TO_UINT
	DINT	X		BYTE_TO_DINT
	UDINT	X		BYTE_TO_UDINT
	LINT	X		BYTE_TO_LINT
	ULINT	X		BYTE_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_TIME
	LTIME	X		BYTE_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_TOD
	LTOD	X		BYTE_TO_LTOD
	DATE	X		BYTE_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_CHAR
	WCHAR	X		BYTE_TO_WCHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.				

See also

Overview of data type conversion (Page 354)

Implicit conversion of BYTE (Page 358)

BYTE (Page 236)

Explicit conversion of WORD (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WORD[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	WORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_BYTE
	DWORD[1)]	X		WORD_TO_DWORD
	LWORD[1)]	X		WORD_TO_LWORD

	SINT	X	ENO = TRUE
			#sint1 := WORD_TO_SINT(16#FFFF); // -1 to
			#sint1 := WORD_TO_SINT(16#FF80); // -128
			#sint1 := WORD_TO_SINT(16#0); // 0 to
			#sint1 := WORD_TO_SINT(16#007F); // 127
			
			ENO = FALSE
			#sint1 := WORD_TO_SINT(16#FF7F); // -129 to
			#sint1 := WORD_TO_SINT(16#8000); // -32768
			#sint1 := WORD_TO_SINT(16#0080); // 128 to
			#sint1 := WORD_TO_SINT(16#7FFF); // 32767
				WORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_USINT
	INT	X		WORD_TO_INT
	UINT	X		WORD_TO_UINT
	DINT	X		WORD_TO_DINT
	UDINT	X		WORD_TO_UDINT
	LINT	X		WORD_TO_LINT
	ULINT	X		WORD_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_TIME
	LTIME	X		WORD_TO_LTIME
	S5TIME	X		WORD_TO_S5TIME
	LDT	X		WORD_TO_LDT
	DT	-	No explicit conversion	-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_TOD
	LTOD	X		WORD_TO_LTOD
	DATE	X		WORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_CHAR
	WCHAR	X		WORD_TO_WCHAR
WORD_ BCD16	INT	X	The value to be converted has data type WORD and is accepted as a BCD-coded value between -999 and +999. The result is available after conversion as an integer (in binary notation) of the type INT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	WORD_BCD16_TO_INT
BCD16	INT	X		BCD16_TO_INT

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.

See also

Implicit conversion of WORD (Page 359)

Overview of data type conversion (Page 354)

WORD (Page 237)

Explicit conversion of DWORD (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DWORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DWORD[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	DWORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_BYTE
	WORD[1)]	X		DWORD_TO_WORD
	LWORD[1)]	X		DWORD_TO_LWORD

	SINT	X	ENO = TRUE
			#sint1 := DWORD_TO_SINT(16#FFFF_FFFF); // -1 to
			#sint1 := DWORD_TO_SINT(16#FFFF_FF80); // -128
			#sint1 := DWORD_TO_SINT(16#0); // 0 to
			#sint1 := DWORD_TO_SINT(16#0000_007F); // 127
			
			ENO = FALSE
			#sint1 := DWORD_TO_SINT(16#FFFF_FF7F); // -129
			#sint1 := DWORD_TO_SINT(16#8000_0000); // -2147483648
			#sint1 := DWORD_TO_SINT(16#0000_0080); // 128 to
			#sint1 := DWORD_TO_SINT(16#7FFF_FFFF); // 2147483647
				DWORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_USINT

	INT	X	ENO = TRUE
			#int1 := DWORD_TO_INT(16#FFFF_FFFF); // -1 to
			#int1 := DWORD_TO_INT(16#FFFF_8000); // -32768
			#int1 := DWORD_TO_INT(16#0); // 0 to
			#int1 := DWORD_TO_INT(16#0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := DWORD_TO_INT(16#FFFF_7FFF); // -32769
			#int1 := DWORD_TO_INT(16#8000_0000); //
				DWORD_TO_INT

Source	Destination	Conversion	Explanation	Mnemonics of the instruction

			-2147483648
			#int1 := DWORD_TO_INT(16#8000); // 32768 to
			#int1 := DWORD_TO_INT(16#7FFF_FFFF); // 2147483647	

	UINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If no errors occur during the conversion, the signal state of ENO = 1; if an error occurs during processing, the signal state of ENO = 0.	DWORD_TO_UINT
	DINT	X		DWORD_TO_DINT
	UDINT	X		DWORD_TO_UDINT
	LINT	X		DWORD_TO_LINT
	ULINT	X		DWORD_TO_ULINT
	REAL	X		DWORD_TO_REAL
	LREAL	-	No explicit conversion	-
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_TIME
	LTIME	X		DWORD_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_TOD
	LTOD	X		DWORD_TO_LTOD
	DATE	X		DWORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_CHAR
	WCHAR	X		DWORD_TO_WCHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DWORD_BCD32	DINT	X	The value to be converted has data type DWORD and is accepted as a BCD-coded value between -9999999 and +9999999. The result is available after conversion as an integer (in binary notation) of the type DINT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	DWORD_BCD32_TO_DINT
BCD32	DINT	X		BCD32_TO_DINT

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.

See also

Implicit conversion of DWORD (Page 361)

Overview of data type conversion (Page 354)

DWORD (Page 238)

Explicit conversion of LWORD (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LWORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LWORD[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	LWORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_BYTE
	WORD[1)]	X		LWORD_TO_WORD
	DWORD[1)]	X		LWORD_TO_DWORD

	SINT	X	ENO = TRUE
			#sint1 := LWORD_TO_SINT(16#FFFF_FFFF_FFFF_FFFF); // -1 to
			#sint1 := LWORD_TO_SINT(16#FFFF_FFFF_FFFF_FF80); // -128
			#sint1 := LWORD_TO_SINT(16#0); // 0 to
			#sint1 := LWORD_TO_SINT(16#0000_0000_0000_007F); // 127
			
			ENO = FALSE
			#sint1 := LWORD_TO_SINT(16#FFFF_FFFF_FFFF_FF7F); // -129
			#sint1 := LWORD_TO_SINT(16#8000_0000_0000_0000); // -9223372036854775808
			#sint1 := LWORD_TO_SINT(16#0000_0000_0000_0080); // 128
			#sint1 := LWORD_TO_SINT(16#7FFF_FFFF_FFFF_FFFF); // 9223372036854775807
				LWORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_USINT

	INT	X	ENO = TRUE
			#int1 := LWORD_TO_INT(16#FFFF_FFFF_FFFF_FFFF); // -1 to
			#int1 := LWORD_TO_INT(16#FFFF_FFFF_FFFF_8000); // -32768
			#int1 := LWORD_TO_INT(16#0); // 0 to
			#int1 := LWORD_TO_INT(16#0000_0000_0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := LWORD_TO_INT(16#FFFF_FFFF_FFFF_7FFF); // -32769
			#int1 := LWORD_TO_INT(16#8000_0000_0000_0000); // -2147483648
				LWORD_TO_INT

Source	Destination	Conver-sion	Explanation	Mnemonics of the in-struction
			#int1 := LWORD_TO_INT(16#0000_0000_0000_8000); // 32768 to #int1 := LWORD_TO_INT(16#7FFF_FFFF_FFFF_FFFF); // 2147483647	
	UINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_UINT

	DINT	X	ENO = TRUE
			#dint1 := LWORD_TO_DINT(16#FFFF_FFFF_FFFF_FFFF); // -1 to
			#dint1 := LWORD_TO_DINT(16#FFFF_FFFF_8000_0000); // -2147483648
			#dint1 := LWORD_TO_DINT(16#0); // 0 to
			#dint1 := LWORD_TO_DINT(16#0000_0000_7FFF_FFFF); // 2147483647
			
			ENO = FALSE
			#dint1 := LWORD_TO_DINT(16#FFFF_FFFF_7FFF_FFFF); // -2147483649 to
			#dint1 := LWORD_TO_DINT(16#8000_0000_0000_0000); // -9223372036854775808
			#dint1 := LWORD_TO_DINT(16#0000_0000_8000_0000); // 2147483648 to
			#dint1 := LWORD_TO_DINT(16#7FFF_FFFF_FFFF_FFFF); // 9223372036854775807
				LWORD_TO_DINT

	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_UDINT
	LINT	X		LWORD_TO_LINT
	ULINT	X		LWORD_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If no errors occur during the conversion, the signal state of ENO = 1; if an error occurs during processing, the signal state of ENO = 0.	LWORD_TO_LREAL
	TIME	X		LWORD_TO_TIME
	LTIME	X		LWORD_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_TOD
	LTOD	X		LWORD_TO_LTOD
	DATE	X		LWORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_CHAR
	WCHAR	X		LWORD_TO_WCHAR

Source	Destination	Conver-sion	Explanation	Mnemonics of the in-struction
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.				

See also

Implicit conversion of LWORD (Page 362)
Overview of data type conversion (Page 354)
LWORD (Page 240)

3.15.3.2 Integers (S7-1500)

Explicit conversion of SINT (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the SINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
SINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	SINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	SINT_TO_BYTE
	WORD[1)]	X		SINT_TO_WORD
	DWORD[1)]	X		SINT_TO_DWORD
	LWORD[1)]	X		SINT_TO_LWORD
	USINT	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	SINT_TO_USINT
	INT	X		SINT_TO_INT
	UINT	X		SINT_TO_UINT
	DINT	X		SINT_TO_DINT
	UDINT	X		SINT_TO_UDINT
	LINT	X		SINT_TO_LINT
	ULINT	X		SINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	SINT_TO_REAL, NORM_X
	LREAL	X		SINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	SINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	SINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	SINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value	SINT_TO_TOD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			"-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0)	SINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	SINT_TO_DATE
	STRING	X	The value is converted into a character string. • LAD / FBD / STL: The first characters of the string are padded with spaces. The number of spaces depends on the length of the numerical value. Positive numeric values are output without a sign. • SCL: No spaces are inserted. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	SINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		SINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	SINT_TO_CHAR
	WCHAR[1)]	X		SINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of SINT (Page 364)
Overview of data type conversion (Page 354)
SINT (8-bit integers) (Page 241)

Explicit conversion of USINT (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the USINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
USINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	USINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	USINT_TO_BYTE
	WORD[1)]	X		USINT_TO_WORD
	DWORD[1)]	X		USINT_TO_DWORD
	LWORD[1)]	X		USINT_TO_LWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign is changed during the conversion, the enable output ENO is set to "0".	USINT_TO_SINT
	INT	X	The bit pattern of the source value is converted and transferred to the destination data type.	USINT_TO_INT
	UINT	X		USINT_TO_UINT
	DINT	X		USINT_TO_DINT
	UDINT	X		USINT_TO_UDINT
	LINT	X		USINT_TO_LINT
	ULINT	X		USINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	USINT_TO_REAL, NORM_X
	LREAL	X		USINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	USINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	USINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	USINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in milliseconds since 0:0)	USINT_TO_TOD
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in nanoseconds since 0:0)	USINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in days since 1990-1-1)	USINT_TO_DATE
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	USINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		USINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type.	USINT_TO_CHAR
	WCHAR[1)]	X		USINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of USINT (Page 365)

Overview of data type conversion (Page 354)

USINT (8-bit integers) (Page 242)

Explicit conversion of INT (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the INT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
INT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	INT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	INT_TO_BYTE
	WORD[1)]	X		INT_TO_WORD
	DWORD[1)]	X		INT_TO_DWORD
	LWORD[1)]	X		INT_TO_LWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	INT_TO_SINT
	USINT	X		INT_TO_USINT
	UINT	X		INT_TO_UINT
	DINT	X		INT_TO_DINT
	UDINT	X		INT_TO_UDINT
	LINT	X		INT_TO_LINT
	ULINT	X		INT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	INT_TO_REAL, NORM_X
	LREAL	X		INT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	INT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	INT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	INT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO	INT_TO_TOD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			is set to "0". (interpretation in milliseconds since 0:0; check for 24h limit)	
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0; check for 24h limit)	INT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1; check for negative value)	INT_TO_DATE
	STRING	X	The value is converted into a character string. • LAD / FBD / STL: The first characters of the string are padded with spaces. The number of spaces depends on the length of the numerical value. Positive numeric values are output without a sign. • SCL: No spaces are inserted. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	INT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		INT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	INT_TO_CHAR
	WCHAR[1)]	X		INT_TO_WCHAR
	BCD16	X	The value to be converted has type INT and is accepted as an integer with a value between -999 and +999. The result is available after conversion as a BCD-coded number of the type WORD. A real conversion takes place. If the value is outside the target	INT_TO_BCD16
	BCD16_WORD	X		INT_TO_BCD16_WORD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			area, a synchronous error is not triggered, but rather only the status bit OV is set.	
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of INT (Page 367)

Overview of data type conversion (Page 354)

INT (16-bit integers) (Page 243)

Explicit conversion of UINT (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the UINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
UINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	UINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If bits are lost in the process, the enable output ENO is set to "0".	UINT_TO_BYTE
	WORD[1)]	X		UINT_TO_WORD
	DWORD[1)]	X		UINT_TO_DWORD
	LWORD[1)]	X		UINT_TO_LWORD
	SINT	X		UINT_TO_SINT
	USINT	X		UINT_TO_USINT
	INT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UINT_TO_INT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	UINT_TO_DINT
	UDINT	X		UINT_TO_UDINT
	LINT	X		UINT_TO_LINT
	ULINT	X		UINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	UINT_TO_REAL, NORM_X
	LREAL	X		UINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	UINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	UINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	UINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in milliseconds since 0:0; check for 24h limit)	UINT_TO_TOD
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in nanoseconds since 0:0; check for 24h limit)	UINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in days since 1990-1-1; check for negative value)	UINT_TO_DATE, T_CONV
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		UINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the destination data type. The enable output ENO is set to "0" in the event of overflow.	UINT_TO_CHAR
	WCHAR[1)]	X		UINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of UINT (Page 368)

Overview of data type conversion (Page 354)

UINT (16-bit integers) (Page 244)

Explicit conversion of DINT (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	DINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	DINT_TO_BYTE
	WORD[1)]	X		DINT_TO_WORD
	DWORD[1)]	X		DINT_TO_DWORD
	LWORD[1)]	X		DINT_TO_LWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	DINT_TO_SINT
	USINT	X		DINT_TO_USINT
	INT	X		DINT_TO_INT
	UINT	X		DINT_TO_UINT
	UDINT	X		DINT_TO_UDINT
	LINT	X		DINT_TO_LINT
	ULINT	X		DINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	DINT_TO_REAL, NORM_X
	LREAL	X		DINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	DINT_TO_TIME, T_CONV
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	DINT_TO_LTIME, T_CONV
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	DINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1"	DINT_TO_TOD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			(16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0)	DINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	DINT_TO_DATE
	STRING	X	The value is converted into a character string. • LAD / FBD / STL: The first characters of the string are padded with spaces. The number of spaces depends on the length of the numerical value. Positive numeric values are output without a sign. • SCL: No spaces are inserted. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	DINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		DINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	DINT_TO_CHAR
	WCHAR[1)]	X		DINT_TO_WCHAR
	BCD32	X	The value to be converted has type DINT and is accepted as an integer with a value between –999999 and +9999999. The result is available	DINT_TO_BCD32
	BCD32_DWORD	X		DINT_TO_BCD32_DWORD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			after conversion as a BCD-coded number of the type DWORD. The enable output is set to "0" in the event of overflow. A real conversion takes place. If the value is outside the target area, a synchronous error is not triggered, but rather only the status bit OV is set.	
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of DINT (Page 370)
Overview of data type conversion (Page 354)
DINT (32-bit integers) (Page 245)

Explicit conversion of UDINT (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the UDINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
UDINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	UDINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If bits are lost in the process, the enable output ENO is set to "0".	UDINT_TO_BYTE
	WORD[1)]	X		UDINT_TO_WORD
	DWORD[1)]	X		UDINT_TO_DWORD
	LWORD[1)]	X		UDINT_TO_LWORD
	SINT	X		UDINT_TO_SINT
	USINT	X		UDINT_TO_USINT
	INT	X		UDINT_TO_INT
	UINT	X		UDINT_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UDINT_TO_DINT
	LINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	UDINT_TO_LINT
	ULINT	X		UDINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	UDINT_TO_REAL, NORM_X
	LREAL	X		UDINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	UDINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	UDINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	UDINT_TO_LDT
	DT	-	No explicit conversion	-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0; check for 24h limit)	UDINT_TO_TOD, T_CONV
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0; check for 24h limit)	UDNT_TO_LTOD, T_CONV
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1; check for negative value)	UDINT_TO_DATE
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UDINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		UDINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the destination data type. The enable output ENO is set to "0" in the event of overflow.	UDINT_TO_CHAR
	WCHAR[1)]	X		UDINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of UDINT (Page 372)

Overview of data type conversion (Page 354)

UDINT (32-bit integers) (Page 246)

Explicit conversion of LINT (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	LINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	LINT_TO_BYTE
	WORD[1)]	X		LINT_TO_WORD
	DWORD[1)]	X		LINT_TO_DWORD
	LWORD[1)]	X		LINT_TO_LWORD
	SINT	X		LINT_TO_SINT
	USINT	X		LINT_TO_USINT
	INT	X		LINT_TO_INT
	UINT	X		LINT_TO_UINT
	DINT	X		LINT_TO_DINT
	UDINT	X		LINT_TO_UDINT
	ULINT	X		LINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	LINT_TO_REAL, NORM_X
	LREAL	X		LINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	LINT_TO_TIME, T_CONV
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	LINT_TO_LTIME, T_CONV
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	LINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative	LINT_TO_TOD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0)	LINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	LINT_TO_DATE
	STRING	X	The value is converted into a character string.	LINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X	• LAD / FBD / STL: The first characters of the string are padded with spaces. The number of spaces depends on the length of the numerical value. Positive numeric values are output without a sign. • SCL: No spaces are inserted. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	LINT_TO_WSTRING
	CHAR[1]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	LINT_TO_CHAR
	WCHAR[1]	X		LINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of LINT (Page 374)
Overview of data type conversion (Page 354)
LINT (64-bit integers) (Page 247)

Explicit conversion of ULINT (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the ULINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
ULINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	ULINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	ULINT_TO_BYTE
	WORD[1)]	X		ULINT_TO_WORD
	DWORD[1)]	X		ULINT_TO_DWORD
	LWORD[1)]	X		ULINT_TO_LWORD
	SINT	X		ULINT_TO_SINT
	USINT	X		ULINT_TO_USINT
	INT	X		ULINT_TO_INT
	UINT	X		ULINT_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign bit is overwritten during the conversion, the enable output ENO is set to "0".	ULINT_TO_DINT
	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	ULINT_TO_UDINT
	LINT	X		ULINT_TO_LINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	ULINT_TO_REAL, NORM_X
	LREAL	X		ULINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	ULINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	ULINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	ULINT_TO_LDT
	DT	-	No explicit conversion	-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in milliseconds since 0:0)	ULINT_TO_TOD, T_CONV
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in nanoseconds since 0:0)	ULINT_TO_LTOD, T_CONV
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in days since 1990-1-1)	ULINT_TO_DATE
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	ULINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		ULINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type.	ULINT_TO_CHAR
	WCHAR[1)]	X		ULINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of ULINT (Page 376)

Overview of data type conversion (Page 354)

ULINT (64-bit integers) (Page 248)

3.15.3.3 Floating-point numbers (S7-1500)

Explicit conversion of REAL (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the REAL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
REAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	REAL_TO_DWORD
	LWORD	-	No explicit conversion	-
	SINT	X	The value is converted into the destination data type. The result of conversion depends on the instruction employed. The enable output ENO is set to "0" if the admissible range of values of the destination data type is exceeded during conversion or the value to be converted has an invalid floating-point number.	REAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		REAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		REAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		REAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		REAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		REAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LINT	X		REAL_TO_LINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	ULINT	X		REAL_TO_ULINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LREAL	X	The value is converted into the destination data type. The result of the conversion depends on the instruction used, e.g. TRUNC(2.5) = 2.0; CEIL(2.5) = 3.0	REAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	LTIME	-		-
	S5TIME	-		-
	LDT	-		-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	X	The value is converted into a character string. The enable output ENO is set to "0" if the character string length is exceeded or the value to be converted has an invalid floating-point number. The min. length of the character string is 14 characters.	REAL_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		REAL_TO_WSTRING
	CHAR	-	No explicit conversion	-
	WCHAR	-		

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of REAL (Page 378)
Overview of data type conversion (Page 354)
REAL (Page 251)

Explicit conversion of LREAL (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LREAL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LREAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LREAL_TO_LWORD
	SINT	X	The value is converted into the destination data type. The result of conversion depends on the instruction employed. The enable output ENO is set to "0" if the admissible range of values of the destination data type is exceeded during conversion or the value to be converted has an invalid floating-point number.	LREAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		LREAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		LREAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		LREAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		LREAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		LREAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LINT	X		LREAL_TO_LINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	ULINT	X		LREAL_TO_ULINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	REAL	X	The value is converted into the destination data type. If the permitted value range of the destination data type is violated during the conversion or if the value to be converted is an invalid floating-point number, the enable output ENO is set to "0". A loss in accuracy is tolerated.	LREAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	LTIME	-		-
	S5TIME	-		-
	LDT	-		-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	X	The value is converted into a character string. The enable output ENO is set to "0" if the character string length is exceeded or the value to be converted has an invalid floating-point number. The min.	LREAL_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		LREAL_TO_WSTRING

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
			length of the character string is 21 characters.	
	CHAR	-	No explicit conversion	-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of LREAL (Page 380)
Overview of data type conversion (Page 354)
LREAL (Page 252)

3.15.3.4 Timers (S7-1500)

Explicit conversion of S5TIME (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the S5TIME data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
S5TIME	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	S5TIME_TO_WORD
	DWORD	-	No explicit conversion	-
	LWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	LINT	-		-
	ULINT	-		-
	REAL	-		-
	LREAL	-		-
	TIME	X	A conversion of the source value into the destination data type takes place here (for example, s5t#10ms becomes T#10ms)	S5TIME_TO_TIME
	LTIME	X	A conversion of the source value into the destination data type takes place here (for example, s5t#10ms becomes LTIME#10ms)	S5TIME_TO_LTIME
	LDT	-	No explicit conversion	-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of S5TIME (Page 381)
Overview of data type conversion (Page 354)
S5TIME (duration) (Page 255)

Explicit conversion of TIME (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TIME data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
TIME	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TIME_TO_BYTE
	WORD[1)]	X		TIME_TO_WORD
	DWORD[1)]	X		TIME_TO_DWORD
	LWORD[1)]	X		TIME_TO_LWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the destination data type.	TIME_TO_SINT
	USINT	X		TIME_TO_USINT
	INT	X		TIME_TO_INT
	UINT	X		TIME_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The result of conversion shows the duration in milliseconds.	TIME_TO_DINT, T_CONV
	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the destination data type. A change in sign has the result that the enable output ENO is set to "0".	TIME_TO_UDINT
	LINT	X		TIME_TO_LINT
	ULINT	X		TIME_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	X	A conversion of the source value into the destination data type takes place here (for example, T#10ms becomes s5t#10ms)	TIME_TO_TIME
	LTIME	X	A conversion of the source value into the destination data type takes place here (for example, T#10ms becomes LTIME#10ms)	TIME_TO_LTIME
	LDT	-	No explicit conversion	-
	DT	-		-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the source value is outside the value range of TOD, the destination data type remains unchanged.	TIME_TO_TOD
	LTOD	-	No explicit conversion	-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of TIME (Page 383)
Overview of data type conversion (Page 354)
TIME (IEC time) (Page 256)

Explicit conversion of LTIME (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LTIME data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LTIME	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LTIME_TO_BYTE
	WORD[1)]	X		LTIME_TO_WORD
	DWORD[1)]	X		LTIME_TO_DWORD
	LWORD[1)]	X		LTIME_TO_LWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as nanoseconds to the destination data type. A change in sign has the result that the enable output ENO is set to "0".	LTIME_TO_SINT
	USINT	X		LTIME_TO_USINT
	INT	X		LTIME_TO_INT
	UINT	X		LTIME_TO_UINT
	DINT	X		LTIME_TO_DINT
	UDINT	X		LTIME_TO_UDINT
	LINT	X		LTIME_TO_LINT, T_CONV
	ULINT	X		LTIME_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	X	A conversion of the source value into the destination data type takes place here (for example, LTIME#10ms becomes s5t#10ms)	LTIME_TO_S5TIME
	TIME	X	A conversion of the source value into the destination data type takes place here (for example, LTIMET#10ms becomes T#10ms)	LTIME_TO_TIME
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, LTIMET#10ms becomes LDT#1970-1-1-0:0:0.010000000)	LTIME_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	LTOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the source value is outside the value range of LTOD, the destination data type remains unchanged.	LTIME_TO_LTOD
	DATE	-	No explicit conversion	-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of LTIME (Page 385)
Overview of data type conversion (Page 354)
LTIME (IEC time) (Page 257)

3.15.3.5 Date and time-of-day (S7-1500)

Explicit conversion of DT (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DT	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	LINT	-		-
	ULINT	-		-
	REAL	-		-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes LDT#1990-1-2-0:0:1.0)	DT_TO_LDT
	DTL	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes DTL#1990-1-2-0:0:1.0)	DT_TO_DTL
	TOD	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes TOD#1990-1-2-0:0:1.0)	DT_TO_TOD
	LTOD	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes LTOD#1990-1-2-0:0:1.0)	DT_TO_LTOD
	DATE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DT_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of DT (Page 387)
Overview of data type conversion (Page 354)
DATE_AND_TIME (date and time of day) (Page 259)

Explicit conversion of LDT (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LDT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LDT	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LDT_TO_BYTE
	WORD[1)]	X		LDT_TO_WORD
	DWORD[1)]	X		LDT_TO_DWORD
	LWORD[1)]	X		LDT_TO_LWORD
	SINT	X		LDT_TO_SINT
	USINT	X		LDT_TO_USINT
	INT	X		LDT_TO_INT
	UINT	X		LDT_TO_UINT
	DINT	X		LDT_TO_DINT
	UDINT	X		LDT_TO_UDINT
	LINT	X		LDT_TO_LINT
	ULINT	X		LDT_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LDT_TO_LTIME
	DT	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes DT#1990-1-2-0:0:1.0)	LDT_TO_DT
	DTL	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes DTL#1990-1-2-0:0:1.0)	LDT_TO_DTL
	TOD	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes TOD#1990-1-2-0:0:1.0)	LDT_TO_TOD
	LTOD	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes LTOD#1990-1-2-0:0:1.0)	LDT_TO_LTOD
	DATE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LDT_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of LDT (Page 389)

Overview of data type conversion (Page 354)

LDT (DATE_AND_LTIME) (Page 261)

Explicit conversion of DTL (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DTL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DTL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	LINT	-		-
	ULINT	-		-
	REAL	-		-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, DTL#1990-1-2-0:0:1.0 becomes LDT#1990-1-2-0:0:1.0)	DTL_TO_LDT
	DT	X	A conversion of the source value into the destination data type takes place here (for example, DTL#1990-1-2-0:0:1.0 becomes DT#1990-1-2-0:0:1.0)	DTL_TO_DT
	TOD	X	During the conversion, the time of day is extracted from the DTL format and written to the destination data type.	DTL_TO_TOD, T_CONV
	LTOD	X		DTL_TO_LTOD
	DATE	X	During the conversion, the date is extracted from the DTL format and written to the destination data type. The enable output ENO is set to "0" in the event of overflow.	DTL_TO_DATE, T_CONV
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of DTL (Page 391)
Overview of data type conversion (Page 354)
DTL (Page 261)

Explicit conversion of DATE (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DATE data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DATE	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DATE_TO_BYTE
	WORD[1)]	X		DATE_TO_WORD
	DWORD[1)]	X		DATE_TO_DWORD
	LWORD[1)]	X		DATE_TO_LWORD
	SINT	X	The number of days since 1/1/1990 is returned as result.	DATE_TO_SINT
	USINT	X		DATE_TO_USINT
	INT	X		DATE_TO_INT
	UINT	X		DATE_TO_UINT
	DINT	X		DATE_TO_DINT
	UDINT	X		DATE_TO_UDINT
	LINT	X		DATE_TO_LINT
	ULINT	X		DATE_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	DT	X	The conversion sets the date in the destination type.	DATE_TO_DT
	LDT	X		DATE_TO_LDT
	DTL [2)]	X		DATE_TO_DTL
	TOD	-	No explicit conversion	-
	LTOD	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

[2)] The data type DTL is initialized as follows: 1970-1-1-0:0:0. Conversion from DATE_TO_DTL only sets the corresponding part of the DTL. The rest of the DTL remains as it was initialized. This results in the following differences for the instructions "T_COMBINE" and "T_CONV":

- T_COMBINE(D#2015-24-12, LTOD#12:13:14) returns DTL#2015-24-12-12:13:14
- myDTL := T_CONV(D#2015-24-12); myDTL := T_CONV(LTOD#12:13:14) results in myDTL = DTL#1970-1-1-12:13:14

See also

Implicit conversion of DATE (Page 393)
Overview of data type conversion (Page 354)
DATE (Page 258)

Explicit conversion of TOD (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TOD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
TOD	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TOD_TO_BYTE
	WORD[1)]	X		TOD_TO_WORD
	DWORD[1)]	X		TOD_TO_DWORD
	LWORD[1)]	X		TOD_TO_LWORD
	SINT	X	The number of milliseconds since midnight is returned as result.	TOD_TO_SINT
	USINT	X		TOD_TO_USINT
	INT	X		TOD_TO_INT
	UINT	X		TOD_TO_UINT
	DINT	X		TOD_TO_DINT
	UDINT	X	The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).	TOD_TO_UDINT, T_CONV
	LINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TOD_TO_LINT
	ULINT	X		TOD_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	X	The duration since midnight is returned as result.	TOD_TO_TIME
	LTIME	-	No explicit conversion	-
	DT	X	The conversion sets the time in the destination data type.	TOD_TO_DT
	LDT	X		TOD_TO_LDT
	DTL	X		TOD_TO_DTL
	DATE	-	No explicit conversion	-
	LTOD	X	A conversion of the source value into the destination data type takes place here (for example, TOD#1:0:0.0 becomes LTOD#1:0:0.0)	TOD_TO_LTOD
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of TOD (Page 395)
Overview of data type conversion (Page 354)
TIME_OF_DAY (TOD) (Page 258)

Explicit conversion of LTOD (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LTOD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LTOD	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LTOD_TO_BYTE
	WORD[1)]	X		LTOD_TO_WORD
	DWORD[1)]	X		LTOD_TO_DWORD
	LWORD[1)]	X		LTOD_TO_LWORD
	SINT	X	The number of nanoseconds since midnight is returned as result.	LTOD_TO_SINT
	USINT	X		LTOD_TO_USINT
	INT	X		LTOD_TO_INT
	UINT	X		LTOD_TO_UINT
	DINT	X		LTOD_TO_DINT
	UDINT	X		LTOD_TO_UDINT
	LINT	X		LTOD_TO_LINT
	ULINT	X		LTOD_TO_ULINT, T_CONV
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LTOD_TO_LTIME
	DT	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes DT#1:0:0.0)	LTOD_TO_DT
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes LDT#1:0:0.0)	LTOD_TO_LDT
	DTL	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes DTL#1:0:0.0)	LTOD_TO_DTL
	DATE	-	No explicit conversion	-
	TOD	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes TOD#1:0:0.0)	LTOD_TO_TOD
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of LTOD (Page 397)
Overview of data type conversion (Page 354)
LTOD (LTIME_OF_DAY) (Page 259)

3.15.3.6 String (S7-1500)

Explicit conversion of CHAR (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the CHAR data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
CHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	CHAR_TO_BYTE
	WORD[1)]	X		CHAR_TO_WORD
	DWORD[1)]	X		CHAR_TO_DWORD
	LWORD[1)]	X		CHAR_TO_LWORD
	SINT	X		CHAR_TO_SINT
	USINT	X		CHAR_TO_USINT
	INT	X		CHAR_TO_INT
	UINT	X		CHAR_TO_UINT
	DINT	X		CHAR_TO_DINT
	UDINT	X		CHAR_TO_UDINT
	LINT	X		CHAR_TO_LINT
	ULINT	X		CHAR_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	X	The value is converted into the first character in the character string (STRING). The length "1" is set after conversion if the character string length is not defined. If the character string length is defined, it remains unchanged following conversion.	CHAR_TO_STRING, S_CONV
	WSTRING	-	No explicit conversion	-
	WCHAR	X		CHAR_TO_WCHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD, LWORD) and the data type WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of CHAR (Page 399)
Overview of data type conversion (Page 354)
CHAR (Page 263)

Explicit conversion of WCHAR (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WCHAR data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WCHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WCHAR_TO_BYTE
	WORD[1)]	X		WCHAR_TO_WORD
	DWORD[1)]	X		WCHAR_TO_DWORD
	LWORD[1)]	X		WCHAR_TO_LWORD
	SINT	X		WCHAR_TO_SINT
	USINT	X		WCHAR_TO_USINT
	INT	X		WCHAR_TO_INT
	UINT	X		WCHAR_TO_UINT
	DINT	X		WCHAR_TO_DINT
	UDINT	X		WCHAR_TO_UDINT
	LINT	X		WCHAR_TO_LINT
	ULINT	X		WCHAR_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	X	The value is converted into the first character in the character string (WSTRING). The length "1" is set after conversion if the character string length is not defined. If the character string length is defined, it remains unchanged following conversion.	WCHAR_TO_WSTRING
	CHAR	X		WCHAR_TO_CHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data type CHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Overview of data type conversion (Page 354)
WCHAR (Page 263)
Implicit conversion of WCHAR (Page 400)

Explicit conversion of STRING (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the STRING data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
STRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are admissible for conversion: • Digit • Sign • Dot The first character in the character string may be a sign (+, -) or a digit. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator to the left of the decimal point is permitted but is ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	STRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		STRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		STRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		STRING_TO_UINT, S_CONV, STRG_VAL
	DINT	X		STRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		STRING_TO_UDINT, S_CONV, STRG_VAL
	LINT	X		STRING_TO_LINT, S_CONV, STRG_VAL
	ULINT	X		STRING_TO_ULINT, S_CONV, STRG_VAL
	REAL	X		STRING_TO_REAL, S_CONV, STRG_VAL
	LREAL	X		STRING_TO_LREAL, S_CONV, STRG_VAL
	S5TIME	-	No explicit conversion	-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	CHAR	X	The first character in the character string (STRING) is transferred to the destination data type. The value "0" is written to the destination data type if the character string is empty.	STRING_TO_CHAR, S_CONV
	WCHAR	-	No explicit conversion	-
	WSTRING	X		STRING_TO_WSTRING
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of STRING (Page 402)
Overview of data type conversion (Page 354)
STRING (Page 264)

Explicit conversion of WSTRING (S7-1500)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WSTRING data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WSTRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (WSTRING) and stops at the end of the string or at the first inadmissible character. The following characters are admissible for conversion: • Digit • Sign • Dot The first character in the character string may be a sign (+, -) or a digit. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator to the left of the decimal point is permitted but is ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	WSTRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		WSTRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		WSTRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		WSTRING_TO_UINT, S_CONV, STRG_VAL
	DINT	X		WSTRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		WSTRING_TO_UDINT, S_CONV, STRG_VAL
	LINT	X		WSTRING_TO_LINT, S_CONV, STRG_VAL
	ULINT	X		WSTRING_TO_ULINT, S_CONV, STRG_VAL
	REAL	X		WSTRING_TO_REAL, S_CONV, STRG_VAL
	LREAL	X		WSTRING_TO_LREAL, S_CONV, STRG_VAL
	S5TIME	-	No explicit conversion	-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	CHAR	-		-
	WCHAR	X	The first character in the character string (WSTRING) is transferred to the destination data type. The value "0" is written to the destination data type if the character string is empty.	WSTRING_TO_WCHAR
	STRING	X		WSTRING_TO_STRING

x: Conversion possible
- : Conversion not possible

See also

Overview of data type conversion (Page 354)
WSTRING (Page 267)
Implicit conversion of WSTRING (Page 403)

3.16 Data type conversion for S7-1200 (S7-1200)

3.16.1 Overview of data type conversion (S7-1200)

Introduction

If you link several operands in an instruction, you must make sure that the data types are compatible. This applies also for assignments or for supplying block parameters. If the operands are not the same data type, a conversion has to be carried out.

There are two options for the conversion:

- Implicit conversion
  The conversion take place automatically when the instruction is executed.
- Explicit conversion
  You use an explicit conversion instruction before the actual instruction is executed.

Note

The data type conversion options described always refer to the latest version of the CPU (V. 4) Conversions marked as possible may not be available in CPU versions 1 - 3.

Note

Converting bit strings in SCL

All bit strings (BYTE, WORD, and DWORD) are handled like the corresponding unsigned integers (USINT, UINT, and UDINT) in expressions. Therefore, implicit conversion from DWORD to REAL is carried out like a conversion from UDINT to REAL, for example.

Implicit conversion

An implicit conversion is executed automatically if the data types of the operands are compatible. This compatibility test can be carried out according to criteria that are more or less strict:

- With IEC check (default)
  If IEC check is set, the following rules are applied:
  - Implicit conversion of BOOL to other data types is not possible.
  - Only the REAL, BYTE, WORD, DINT, INT, SINT, UDINT, UINT, USINT, TIME, DT, STRING, CHAR and WCHAR data types can be converted implicitly.
  - The bit length of the source data type must not exceed the bit length of the target data type. An operand of data type WORD, for example, cannot be declared at a parameter at which data type BYTE is expected.
- Without IEC check
  If IEC check is not set, the following rules are applied:
  - Implicit conversion of BOOL to other data types is not possible.
  - Only the REAL, LREAL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, TIME, DTL, TOD, DATE, STRING, CHAR and WCHAR data types can be converted implicitly.
  - The bit length of the source data type must not exceed the bit length of the target data type. An operand of data type DWORD, for example, cannot be declared at a parameter at which data type WORD is expected.
  - The bit length of an operand entered in-out parameters InOut) must be the same as the programmed bit length for the parameter in question.

Note

Implicit conversion without IEC check

The programming editor uses a gray rectangle to mark operands that are implicitly converted. The dark gray rectangle signals that an implicit conversion is possible without any accuracy loss, for example, if you convert the data type SINT to INT. A light gray rectangle signals that implicit conversion is possible, but errors could occur during runtime. If, for example, you are converting the data type DINT to INT and an overflow occurs, the enable output ENO is set to "0".

For more information about the setting of the IEC check and the implicit conversion, refer to "See also".

Explicit conversion

If the operands are not compatible and an implicit conversion is therefore not possible, you can use an explicit conversion instruction. You can find the conversion instructions in the "Instructions" task card.

A possible overflow is displayed at the ENO enable output. An overflow is created, for example, if the value of the source data type is greater than the value of the target data type.

Note

Shifting of bit patterns

If the explicit conversion involves shifting a bit pattern, the enable output ENO is not set.

For more information about explicit conversion, refer to "See also".

The following figure shows an example in which an explicit data type conversion must be carried out:

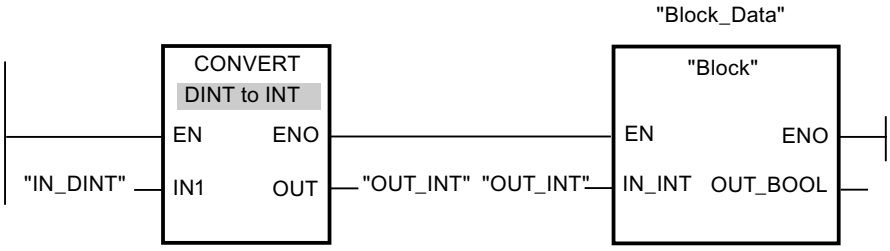

The "Block" function block expects a tag of the INT data type at the "IN_INT" input parameter. Therefore, the value of the "IN_DINT" tag must first be converted from DINT to INT. If the value of the "IN_DINT" tag is within the permitted value range of the INT data type, the conversion takes place. Otherwise, an overflow is signaled. A conversion still takes place even in case of an overflow, but the values are cut off and the enable output ENO is set to "0".

See also

Activate or deactivate IEC check (Page 491)

Implicit conversions (Page 491)

Explicit conversions (Page 511)

3.16.2 Implicit conversions (S7-1200)

3.16.2.1 Activate or deactivate IEC check (S7-1200)

The data types of the operands used are check for compatibility. This compatibility test can be carried out according to criteria that are more or less strict. If "IEC check for code blocks" is activated, stricter criteria are applied.

You can set the IEC check centrally for all new blocks of the project or for individual blocks.

Setting IEC check for new blocks

To set the IEC check for all new blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. Select or clear the "IEC check for code blocks" check box in the "Default settings for new blocks" group.
   The IEC check is enabled or disabled for all new blocks in the program.

Setting IEC check for a block

To set the IEC check for a block, proceed as follows:

1. Open the block.
2. Open the "Properties" tab in the Inspector window.
3. Select the "Attributes" group in the area navigation.
4. Select or clear the "IEC check" check box.
   The IEC check is enabled or disabled for this block. The setting is stored together with the project.

3.16.2.2 Binary numbers (S7-1200)

Implicit conversion of BOOL (S7-1200)

Options for implicit conversion

The implicit conversion of the BOOL data type is not possible.

See also

BOOL (bit) (Page 236)

Bit strings (S7-1200)

Implicit conversion of BYTE (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the BYTE data type:

Source	Target	With IEC check	Without IEC check	Explanation
BYTE	BOOL	-	-	No implicit conversion
	WORD	x	x	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DWORD	x	x	
	SINT	-	x	
	USINT	-	x	
	INT	-	x	
	UINT	-	x	
	DINT	-	x	
	UDINT	-	x	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	x	The bit pattern of the source value is transferred unchanged to the target data type.
	WCHAR	-	x	
x: Conversion possible -: Conversion not possible				

See also

BYTE (Page 236)
Activate or deactivate IEC check (Page 491)
Overview of data type conversion (Page 489)
Explicit conversion of BYTE (Page 512)

Implicit conversion of WORD (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the WORD data type:

Source	Target	With IEC check	Without IEC check	Description
WORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The least significant byte is transferred to the target data type, while the most significant byte is ignored.
	DWORD	X	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	SINT	-	X	The least significant byte is transferred to the target data type, while the most significant byte is ignored.
	USINT	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged to the target data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

WORD (Page 237)

Activate or deactivate IEC check (Page 491)

Overview of data type conversion (Page 489)

Explicit conversion of WORD (Page 514)

Implicit conversion of DWORD (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the DWORD data type:

Source	Target	With IEC check	Without IEC check	Explanation
DWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The right bytes are transferred to the target data type; the left bytes are ignored.
	WORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	UDINT	-	X	
	REAL	-	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	-	-	No implicit conversion
	TIME	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DTL	-	-	No implicit conversion
	TOD	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

DWORD (Page 238)

Activate or deactivate IEC check (Page 491)

Overview of data type conversion (Page 489)

Explicit conversion of DWORD (Page 517)

3.16.2.3 Integers (S7-1200)

Implicit conversion of SINT (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the SINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
SINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	USINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value transfer from SINT #-1 -> INT #-1, not filled with "0".)
	INT	X	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

SINT (8-bit integers) (Page 241)
Activate or deactivate IEC check (Page 491)
Overview of data type conversion (Page 489)
Explicit conversion of SINT (Page 521)

Implicit conversion of USINT (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the USINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
USINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from USINT #10 -> DINT #10 or USINT #128 -> SINT #-128)
	INT	X	X	
	UINT	X	X	
	DINT	X	X	
	UDINT	X	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "1", for example, is converted to the value "1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

USINT (8-bit integers) (Page 242)
Activate or deactivate IEC check (Page 491)
Overview of data type conversion (Page 489)
Explicit conversion of USINT (Page 524)

Implicit conversion of INT (S7-1200)

Options for implicit conversion

The following table shows the options for the implicit conversion of the INT data type:

Source	Target	With IEC check	Without IEC check	Explanation
INT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from INT #-1 -> SINT #-1, or INT #-32 767 -> UINT #32 769)
	USINT	-	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

INT (16-bit integers) (Page 243)

Activate or deactivate IEC check (Page 491)

Overview of data type conversion (Page 489)

Explicit conversion of INT (Page 527)

Implicit conversion of UINT (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the UINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
UINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from UINT #100 -> DINT #100 or UINT #60 000 -> INT #-5536)
	USINT	-	X	
	INT	-	X	
	DINT	X	X	
	UDINT	X	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "1", for example, is converted to the value "1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

UINT (16-bit integers) (Page 244)
Activate or deactivate IEC check (Page 491)
Overview of data type conversion (Page 489)
Explicit conversion of UINT (Page 530)

Implicit conversion of DINT (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the DINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
DINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	UDINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DTL	-	-	No implicit conversion
	TOD	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

DINT (32-bit integers) (Page 245)

Activate or deactivate IEC check (Page 491)

Overview of data type conversion (Page 489)

Explicit conversion of DINT (Page 533)

Implicit conversion of UDINT (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the UDINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
UDINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the format of the target data type. (The value "1", for example, is converted to the value "1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DTL	-	-	No implicit conversion
	TOD	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

UDINT (32-bit integers) (Page 246)

Activate or deactivate IEC check (Page 491)

Overview of data type conversion (Page 489)

Explicit conversion of UDINT (Page 536)

3.16.2.4 Floating-point numbers (S7-1200)

Implicit conversion of REAL (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the REAL data type:

Source	Target	With IEC check	Without IEC check	Explanation
REAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the target data type. (for example, rounding off and value conversion of REAL #2.5 -> INT #2 or negative number REAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined REAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LREAL	X	X	The value is transferred to the target data type.
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

REAL (Page 251)
Activate or deactivate IEC check (Page 491)
Overview of data type conversion (Page 489)
Explicit conversion of REAL (Page 539)

Implicit conversion of LREAL (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the LREAL data type:

Source	Target	With IEC check	Without IEC check	Explanation
LREAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the target data type. (for example, rounding off and value conversion of REAL #2.5 -> INT #2 or negative number REAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined REAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Explicit conversion of LREAL (Page 542)

3.16.2.5 Timers (S7-1200)

Implicit conversion of TIME (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the TIME data type:

Source	Target	With IEC check	Without IEC check	Explanation
TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion shows the duration in milliseconds.
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion shows the duration in milliseconds.
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	DTL	-	-	
	TOD	-	X	The bit pattern of a source value that is less than 24 h (86 400 00 ms) is transferred without changes to the target data type. No further changes are made to the target value. The result of the conversion shows the time that has passed since midnight.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME (IEC time) (Page 256)

Activate or deactivate IEC check (Page 491)

Overview of data type conversion (Page 489)

Explicit conversion of TIME (Page 545)

3.16.2.6 Date and time (S7-1200)

Implicit conversion of DATE (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the DATE data type:

Source	Target	With IEC check	Without IEC check	Explanation
DATE	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	DWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	INT	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	UINT	-	X	
	DINT	-	-	No implicit conversion
	UDINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	DTL	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	TOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DATE (Page 258)
Activate or deactivate IEC check (Page 491)
Overview of data type conversion (Page 489)
Explicit conversion of DATE (Page 547)

Implicit conversion of TOD (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the TOD data type:

Source	Target	With IEC check	Without IEC check	Explanation
TOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	DTL	-	-	No implicit conversion
	DATE	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME_OF_DAY (TOD) (Page 258)

Activate or deactivate IEC check (Page 491)

Overview of data type conversion (Page 489)

Explicit conversion of TOD (Page 548)

Implicit conversion of DTL (S7-1200)

Options for implicit conversion

The DTL data type cannot be implicitly converted.

See also

Explicit conversion of DTL (Page 549)

3.16.2.7 Character strings (S7-1200)

Implicit conversion of CHAR (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the CHAR data type:

Source	Target	With IEC check	Without IEC check	Explanation
CHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	WCHAR	-	-	
	STRING	X	X	The STRING is shortened to length 1 and includes the character.
	WSTRING	-	-	No implicit conversion
x: Conversion possible -: Conversion not possible				

See also

CHAR (Page 263)
Activate or deactivate IEC check (Page 491)
Overview of data type conversion (Page 489)
Explicit conversion of CHAR (Page 550)

Implicit conversion of WCHAR (S7-1200)

Options for implicit conversion

The following table shows the options for the implicit conversion of the WCHAR data type:

Source	Target	With IEC check	Without IEC check	Explanation
WCHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	CHAR	-	-	
	STRING	-	-	
	WSTRING	X	X	The WSTRING is shortened to length 1 and includes the character.
x: Conversion possible -: Conversion not possible				

Implicit conversion of STRING (S7-1200)

Options for implicit conversion

The following table shows the options for implicit conversion of the STRING data type:

Source	Target	With IEC check	Without IEC check	Explanation
STRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	DATE	-	-	
	TOD	-	-	
	CHAR	-	X	The first character of the STRING is returned if the STRING includes one or more characters. Otherwise, the character is output with coding $00.
	WCHAR	-	-	No implicit conversion
	WSTRING	-	-	
x: Conversion possible -: Conversion not possible				

See also

Explicit conversion of STRING (Page 552)

Implicit conversion of WSTRING (S7-1200)

Options for implicit conversion

The following table shows the options for the implicit conversion of the WSTRING data type:

Source	Target	With IEC check	Without IEC check	Explanation
WSTRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	DATE	-	-	
	TOD	-	-	
	CHAR	-	-	
	WCHAR	-	X	The first character of the WSTRING is returned if the WSTRING includes one or more characters. Otherwise, the character is output with coding $0000.
	STRING	-	-	No implicit conversion
x: Conversion possible -: Conversion not possible				

3.16.3 Explicit conversions (S7-1200)

3.16.3.1 Binary numbers (S7-1200)

Explicit conversion of BOOL (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BOOL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
BOOL	BYTE	X	Only the LSB (Least Significant Bit) is set in the target data type. The enable output ENO is always "1".	BOOL_TO_BYTE
	WORD	X		BOOL_TO_WORD
	DWORD	X		BOOL_TO_DWORD
	SINT	X		BOOL_TO_SINT
	USINT	X		BOOL_TO_USINT
	INT	X		BOOL_TO_INT
	UINT	X		BOOL_TO_UINT
	DINT	X		BOOL_TO_DINT
	UDINT	X		BOOL_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

BOOL (bit) (Page 236)

Implicit conversion of BYTE (Page 493)

Overview of data type conversion (Page 489)

Bit strings (S7-1200)

Explicit conversion of BYTE (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BYTE data type:

Source	Target	Conversion	Explanation	Mnemonics of Instruction
BYTE[1)]	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bits are not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	BYTE_TO_BOOL
	WORD[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	BYTE_TO_WORD
	DWORD[1)]	X		BYTE_TO_DWORD
	SINT	X		BYTE_TO_SINT
	USINT	X		BYTE_TO_USINT
	INT	X		BYTE_TO_INT
	UINT	X		BYTE_TO_UINT
	DINT	X		BYTE_TO_DINT
	UDINT	X		BYTE_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	BYTE_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	BYTE_TO_TOD
	DATE	X		BYTE_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	BYTE_TO_CHAR
	WCHAR	X		BYTE_TO_WCHAR

Source	Target	Conversion	Explanation	Mnemonics of Instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD) are interpreted as an unsigned integer with the same bit length.. Data type BYTE is interpreted as USINT, WORD as UINT and DWORD as UDINT.				

See also

BYTE (Page 236)

Implicit conversion of BYTE (Page 493)

Overview of data type conversion (Page 489)

Explicit conversion of WORD (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WORD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
WORD[1)]	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	WORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_BYTE
	DWORD[1)]	X		WORD_TO_DWORD

	SINT	X	ENO = TRUE
			#sint1 := WORD_TO_SINT(16#FFFF); // -1 to
			#sint1 := WORD_TO_SINT(16#FF80); // -128
			#sint1 := WORD_TO_SINT(16#0); // 0 to
			#sint1 := WORD_TO_SINT(16#007F); // 127
			
			ENO = FALSE
			#sint1 := WORD_TO_SINT(16#FF7F); // -129 to
			#sint1 := WORD_TO_SINT(16#8000); // -32768
			#sint1 := WORD_TO_SINT(16#0080); // 128 to
			#sint1 := WORD_TO_SINT(16#7FFF); // 32767
				WORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_USINT
	INT	X		WORD_TO_INT
	UINT	X		WORD_TO_UINT
	DINT	X		WORD_TO_DINT
	UDINT	X		WORD_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_TOD
	DATE	X		WORD_TO_DATE

Source	Target	Conversion	Explanation	Mnemonics of the instruction
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_CHAR
	WCHAR	X		WORD_TO_WCHAR
WORD_BCD16	INT	X	The value to be converted has data type WORD and is accepted as a BCD-coded value between -999 and +999. The result is available after conversion as an integer (in binary notation) of the type INT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	WORD_BCD16_TO_INT
BCD16	INT	X		BCD16_TO_INT

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD) are interpreted as an unsigned integer with the same bit length.. Data type BYTE is interpreted as USINT, WORD as UINT and DWORD as UDINT.

See also

WORD (Page 237)

Implicit conversion of WORD (Page 494)

Overview of data type conversion (Page 489)

Explicit conversion of DWORD (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DWORD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DWORD[1)]	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	DWORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_BYTE
	WORD[1)]	X		DWORD_TO_WORD

	SINT	X	ENO = TRUE
			#sint1 := DWORD_TO_SINT(16#FFFF_FFFF); // -1 bis
			#sint1 := DWORD_TO_SINT(16#FFFF_FF80); // -128
			#sint1 := DWORD_TO_SINT(16#0); // 0 bis
			#sint1 := DWORD_TO_SINT(16#0000_007F); // 127
			
			ENO = FALSE
			#sint1 := DWORD_TO_SINT(16#FFFF_FF7F); // -129
			#sint1 := DWORD_TO_SINT(16#8000_0000); // -2147483648
			#sint1 := DWORD_TO_SINT(16#0000_0080); // 128 bis
			#sint1 := DWORD_TO_SINT(16#7FFF_FFFF); // 2147483647
				DWORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_USINT

	INT	X	ENO = TRUE
			#int1 := DWORD_TO_INT(16#FFFF_FFFF); // -1 bis
			#int1 := DWORD_TO_INT(16#FFFF_8000); //
				DWORD_TO_INT

Source	Target	Conversion	Explanation	Mnemonics of the instruction

			-32768
			#int1 := DWORD_TO_INT(16#0); // 0 bis
			#int1 := DWORD_TO_INT(16#0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := DWORD_TO_INT(16#FFFF_7FFF); // -32769
			#int1 := DWORD_TO_INT(16#8000_0000); // -2147483648
			#int1 := DWORD_TO_INT(16#8000); // 32768 bis
			#int1 := DWORD_TO_INT(16#7FFF_FFFF); // 2147483647	

	UINT	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_UINT
	DINT	X		DWORD_TO_DINT
	UDINT	X		DWORD_TO_UDINT
	REAL	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If no errors occur during the conversion, the signal state of ENO = 1; if an error occurs during processing, the signal state of ENO = 0.	DWORD_TO_REAL
	LREAL	-	No explicit conversion	-
	TIME	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_TOD
	DATE	X		DWORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DWORD_TO_CHAR
	WCHAR	X		DWORD_TO_WCHAR

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DWORD_BCD32	DINT	X	The value to be converted has data type DWORD and is accepted as a BCD-coded value between -9999999 and +9999999. The result is available after conversion as an integer (in binary notation) of the type DINT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	DWORD_BCD32_TO_DINT
BCD32	DINT	X		BCD32_TO_DINT

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD) are interpreted as an unsigned integer with the same bit length.. Data type BYTE is interpreted as USINT, WORD as UINT and DWORD as UDINT.

See also

DWORD (Page 238)

Implicit conversion of DWORD (Page 495)

Overview of data type conversion (Page 489)

3.16.3.2 Integers (S7-1200)

Explicit conversion of SINT (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the SINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
SINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	SINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	SINT_TO_BYTE
	WORD[1)]	X		SINT_TO_WORD
	DWORD[1)]	X		SINT_TO_DWORD
	USINT	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type.	SINT_TO_USINT
	INT	X		SINT_TO_INT
	UINT	X		SINT_TO_UINT
	DINT	X		SINT_TO_DINT
	UDINT	X		SINT_TO_UDINT
	REAL	X	The value is converted into the format of the target data type (the value "-1" will be converted into the value "-1.0" with the "Convert value" (CONVERT) instruction.	SINT_TO_REAL, NORM_X
	LREAL	X		SINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	SINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0)	SINT_TO_TOD
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1)	SINT_TO_DATE
	STRING	X	The value is converted to a character string. • LAD/FBD: The first characters of the string are padded with spaces. The number of	SINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		SINT_TO_WSTRING

Source	Target	Conversion	Explanation	Mnemonics of the instruction
			spaces depends on the length of the numerical value. Positive numeric values are output without a sign. • STL/SCL: No spaces are inserted. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type.	SINT_TO_CHAR
	WCHAR[1)]	X		SINT_TO_WCHAR
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.				

See also

SINT (8-bit integers) (Page 241)

Implicit conversion of SINT (Page 496)

Overview of data type conversion (Page 489)

Explicit conversion of USINT (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the USINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
USINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	USINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	USINT_TO_BYTE
	WORD[1)]	X		USINT_TO_WORD
	DWORD[1)]	X		USINT_TO_DWORD
	SINT	X	The bit pattern of the source value is transferred unchanged to the target data type. If the sign is changed during the conversion, the enable output ENO is set to "0".	USINT_TO_SINT
	INT	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	USINT_TO_INT
	UINT	X		USINT_TO_UINT
	DINT	X		USINT_TO_DINT
	UDINT	X		USINT_TO_UDINT
	REAL	X	The value is converted into the format of the target data type (the value "1" will be converted into the value "1.0" with the "Convert value" (CONVERT) instruction.	USINT_TO_REAL, NORM_X
	LREAL	X		USINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	USINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	USINT_TO_TOD
	DATE	X		USINT_TO_DATE
	STRING	X	The value is converted to a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	USINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		USINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the target data type.	USINT_TO_CHAR
	WCHAR[1)]	X		USINT_TO_WCHAR

Source	Target	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width (the non-existing sign is replaced with zeros) and then the bits are copied. The source type determines the interpretation.				

See also

USINT (8-bit integers) (Page 242)

Implicit conversion of USINT (Page 497)

Overview of data type conversion (Page 489)

Explicit conversion of INT (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the INT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
INT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	INT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	INT_TO_BYTE
	WORD[1)]	X		INT_TO_WORD
	DWORD[1)]	X		INT_TO_DWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	INT_TO_SINT
	USINT	X		INT_TO_USINT
	UINT	X		INT_TO_UINT
	DINT	X		INT_TO_DINT
	UDINT	X		INT_TO_UDINT
	REAL	X	The value is converted to the format of the target data type (the value "-1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "-1.0").	INT_TO_REAL, NORM_X
	LREAL	X		INT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	INT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0; check for 24h limit)	INT_TO_TOD
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0". (interpretation	INT_TO_DATE

Source	Target	Conversion	Explanation	Mnemonics of the instruction
			in days since 1990-1-1; check for negative value)	
	STRING	X	The value is converted to a character string. • LAD/FBD: The first characters of the string are padded with spaces. The number of spaces depends on the length of the numerical value. Positive numeric values are output without a sign. • STL/SCL: No spaces are inserted. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	INT_TO_STRING, S_CONV, VAL_STRG[)]
	WSTRING	X		INT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	INT_TO_CHAR
	WCHAR[1)]	X		INT_TO_WCHAR
	BCD16	X	The value to be converted has type INT and is accepted as an integer with a value between -999 and +999. The result is available after conversion as a BCD-coded number of the type WORD. A real conversion takes place. If the value is outside the target area, a synchronous error is not triggered, but rather only the status bit OV is set.	INT_TO_BCD16
	BCD16_WORD	X		INT_TO_BCD16_WORD

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

INT (16-bit integers) (Page 243)

Implicit conversion of INT (Page 498)

Overview of data type conversion (Page 489)

Explicit conversion of UINT (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the UINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
UINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	UINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If bits are lost in the process, the enable output ENO is set to "0".	UINT_TO_BYTE
	WORD[1)]	X		UINT_TO_WORD
	DWORD[1)]	X		UINT_TO_DWORD
	SINT	X		UINT_TO_SINT
	USINT	X		UINT_TO_USINT
	INT	X	The bit pattern of the source value is transferred unchanged to the target data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UINT_TO_INT
	DINT	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	UINT_TO_DINT
	UDINT	X		UINT_TO_UDINT
	REAL	X	The value is converted to the format of the target data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "1.0").	UINT_TO_REAL, NORM_X
	LREAL	X		UINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	UINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0; check for 24h limit)	UINT_TO_TOD
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output	UINT_TO_DATE, T_CONV

Source	Target	Conversion	Explanation	Mnemonics of the instruction
			ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1; check for negative value)	
	STRING	X	The value is converted to a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		UINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type. The enable output ENO is set to "0" in the event of overflow.	UINT_TO_CHAR
	WCHAR[1)]	X		UINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data type CHAR are first extended to the necessary width (the non-existing sign is replaced with zeros) and then the bits are copied. The source type determines the interpretation.

See also

UINT (16-bit integers) (Page 244)

Implicit conversion of UINT (Page 499)

Overview of data type conversion (Page 489)

Explicit conversion of DINT (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	DINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	DINT_TO_BYTE
	WORD[1)]	X		DINT_TO_WORD
	DWORD[1)]	X		DINT_TO_DWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	DINT_TO_SINT
	USINT	X		DINT_TO_USINT
	INT	X		DINT_TO_INT
	UINT	X		DINT_TO_UINT
	UDINT	X		DINT_TO_UDINT
	REAL	X	The value is converted to the format of the target data type (the value "-1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "-1.0").	DINT_TO_REAL, NORM_X
	LREAL	X		DINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	DINT_TO_TIME, T_CONV
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	DINT_TO_TOD
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target	DINT_TO_DATE

Source	Target	Conversion	Explanation	Mnemonics of the instruction
			data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	
	STRING	X	The value is converted to a character string. • LAD/FBD: The first characters of the string are padded with spaces. The number of spaces depends on the length of the numerical value. Positive numeric values are output without a sign. • STL/SCL: No spaces are inserted. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	DINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		DINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	DINT_TO_CHAR
	WCHAR[1)]	X		DINT_TO_WCHAR
	BCD32	X	The value to be converted has type DINT and is accepted as an integer with a value between –999999 and +9999999. The result is available after conversion as a BCD-coded number of the type DWORD. The enable output is set to "0" in the event of overflow. A real conversion takes place. If the value is outside the target area, a synchronous error is not triggered, but rather only the status bit OV is set.	DINT_TO_BCD32
	BCD32_DWORD	X		DINT_TO_BCD32_DWORD

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

DINT (32-bit integers) (Page 245)

Implicit conversion of DINT (Page 500)

Overview of data type conversion (Page 489)

Explicit conversion of UDINT (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the UDINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
UDINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	UDINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type. If bits are lost in the process, the enable output ENO is set to "0".	UDINT_TO_BYTE
	WORD[1)]	X		UDINT_TO_WORD
	DWORD[1)]	X		UDINT_TO_DWORD
	SINT	X		UDINT_TO_SINT
	USINT	X		UDINT_TO_USINT
	INT	X		UDINT_TO_INT
	UINT	X		UDINT_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged to the target data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UDINT_TO_DINT
	REAL	X	The value is converted to the format of the target data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "1.0").	UDINT_TO_REAL, NORM_X
	LREAL	X		UDINT_TO_LREAL, NORM_X
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the target data type.	UDINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0; check for 24h limit)	UDINT_TO_TOD, T_CONV
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output	UDINT_TO_DATE

Source	Target	Conversion	Explanation	Mnemonics of the instruction
			ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1; check for negative value)	
	STRING	X	The value is converted to a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UDINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		UDINT_TO_WCHAR
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type. The enable output ENO is set to "0" in the event of overflow.	UDINT_TO_CHAR
	WCHAR[1)]	X		UDINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data type CHAR are first extended to the necessary width (the non-existent sign is replaced with zeros) and then the bits are copied. The source type determines the interpretation.

See also

UDINT (32-bit integers) (Page 246)

Implicit conversion of UDINT (Page 501)

Overview of data type conversion (Page 489)

3.16.3.3 Floating-point numbers (S7-1200)

Explicit conversion of REAL (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the REAL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
REAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	X	The bit pattern of the source value is transferred unchanged to the target data type.	REAL_TO_DWORD
	SINT	X	The value is converted to the target data type. The result of the conversion depends on the instruction used. Enable output ENO is set to "0" if the valid range of values of the target data type is exceeded during conversion, or if the value to be converted is an invalid floating-point number.	REAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		REAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		REAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		REAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		REAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		REAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LREAL	X	The value is converted to the target data type. The result of the conversion depends on the instruction used, e.g. TRUNC(2.5) = 2.0; CEIL(2.5) = 3.0	REAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	X	The value is converted to a character string. Enable output ENO is set to "0" if the character string length is exceeded, or if the value to be converted is an invalid floating-point number. The string has a minimum length of 14 characters.	REAL_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		REAL_TO_WSTRING
	CHAR	-	No explicit conversion	-
	WCHAR	-		-

Source	Target	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible -: Conversion not possible				

See also

REAL (Page 251)

Implicit conversion of REAL (Page 502)

Overview of data type conversion (Page 489)

Explicit conversion of LREAL (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the LREAL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
LREAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	SINT	X	The value is converted to the target data type. The result of the conversion depends on the instruction used. Enable output ENO is set to "0" if the valid range of values of the target data type is exceeded during conversion, or if the value to be converted is an invalid floating-point number.	LREAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		LREAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		LREAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		LREAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		LREAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		LREAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	REAL	X	The value is converted to the target data type. Enable output ENO is set to "0" if the valid range of values of the target data type is exceeded during conversion, or if the value to be converted is an invalid floating-point number. A loss in accuracy is tolerated.	LREAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	X	The value is converted to a character string. Enable output ENO is set to "0" if the character string length is exceeded, or if the value to be converted is an invalid floating-point number. The string has a minimum length of 21 characters.	LREAL_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		LREAL_TO_WSTRING
	CHAR	-	No explicit conversion	-
	WCHAR	-		-
x: Conversion possible -: Conversion not possible				

See also

LREAL (Page 252)
Overview of data type conversion (Page 489)

3.16.3.4 Timers (S7-1200)

Explicit conversion of TIME (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the TIME data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
TIME	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type.	TIME_TO_BYTE
	WORD[1)]	X		TIME_TO_WORD
	DWORD[1)]	X		TIME_TO_DWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the target data type.	TIME_TO_SINT
	USINT	X		TIME_TO_USINT
	INT	X		TIME_TO_INT
	UINT	X		TIME_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion shows the duration in milliseconds.	TIME_TO_DINT, T_CONV
	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the target data type. Enable output ENO is set to "0" if the sign changes.	TIME_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If the source value exceeds the range of values of TOD, the target data type remains unchanged.	TIME_TO_TOD
	DATE	-	No explicit conversion	-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

Source	Target	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible -: Conversion not possible 1) Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.				

See also

TIME (IEC time) (Page 256)

Implicit conversion of TIME (Page 504)

Overview of data type conversion (Page 489)

3.16.3.5 Clock and calendar (S7-1200)

Explicit conversion of DATE (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DATE data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DATE	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DATE_TO_BYTE
	WORD[1)]	X		DATE_TO_WORD
	DWORD[1)]	X		DATE_TO_DWORD
	SINT	X	The number of days since 1/1/1990 is returned as result.	DATE_TO_SINT
	USINT	X		DATE_TO_USINT
	INT	X		DATE_TO_INT
	UINT	X		DATE_TO_UINT
	DINT	X		DATE_TO_DINT
	UDINT	X		DATE_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	DTL [2)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	DATE_TO_DTL
	TOD	-	No explicit conversion	-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.

[2)] The data type DTL is initialized as follows: 1970-1-1-0:0:0. Conversion from DATE_TO_DTL only sets the corresponding part of the DTL. The rest of the DTL remains as it was initialized. This results in the following differences for the instructions "T_COMBINE" and "T_CONV":

- T_COMBINE(D#2015-24-12, LTOD#12:13:14) returns DTL#2015-24-12-12:13:14
- myDTL := T_CONV(D#2015-24-12); myDTL := T_CONV(LTOD#12:13:14) results in myDTL = DTL#1970-1-1-12:13:14

See also

DATE (Page 258)
Implicit conversion of DATE (Page 505)
Overview of data type conversion (Page 489)

Explicit conversion of TOD (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the TOD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
TOD	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type.	TOD_TO_BYTE
	WORD[1)]	X		TOD_TO_WORD
	DWORD[1)]	X		TOD_TO_DWORD
	SINT	X	The number of milliseconds since midnight is returned as result.	TOD_TO_SINT
	USINT	X		TOD_TO_USINT
	INT	X		TOD_TO_INT
	UINT	X		TOD_TO_UINT
	DINT	X		TOD_TO_DINT
	UDINT	X	The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).	TOD_TO_UDINT, T_CONV
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The duration since midnight is returned as result.	TOD_TO_TIME
	DTL	X	The date is set to 1.1.1970 as a result.	TOD_TO_DTL
	DATE	-	No explicit conversion	-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

-: Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

TIME_OF_DAY (TOD) (Page 258)
Implicit conversion of TOD (Page 506)
Overview of data type conversion (Page 489)

Explicit conversion of DTL (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DTL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DTL	BYTE	-	No explicit conversion	-
	WORD	-		-
	DWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	REAL	-		-
	LREAL	-		-
	TIME	-		-
	TOD	X	During the conversion, the time-of-day is extracted from the DTL format and written to the target data type.	DTL_TO_TOD, T_CONV
	DATE	X	During the conversion, the date is extracted from the DTL format and written to the target data type. The enable output ENO is set to "0" in the event of overflow.	DTL_TO_DATE, T_CONV
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible -: Conversion not possible				

See also

DTL (Page 261)

Overview of data type conversion (Page 489)

3.16.3.6 Character strings (S7-1200)

Explicit conversion of CHAR (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the CHAR data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
CHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	CHAR_TO_BYTE
	WORD[1)]	X		CHAR_TO_WORD
	DWORD[1)]	X		CHAR_TO_DWORD
	SINT	X		CHAR_TO_SINT
	USINT	X		CHAR_TO_USINT
	INT	X		CHAR_TO_INT
	UINT	X		CHAR_TO_UINT
	DINT	X		CHAR_TO_DINT
	UDINT	X		CHAR_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	X	The value is converted to the first character in the character string (STRING). If the length of the character string is not defined, the length "1" is set after the conversion. If the length of the character string is defined, it remains unchanged after the conversion.	CHAR_TO_STRING
	WSTRING	-	No explicit conversion	-
	WCHAR	X		CHAR_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.

See also

CHAR (Page 263)
Implicit conversion of CHAR (Page 507)
Overview of data type conversion (Page 489)

Explicit conversion of WCHAR (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the WCHAR data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
WCHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WCHAR_TO_BYTE
	WORD[1)]	X		WCHAR_TO_WORD
	DWORD[1)]	X		WCHAR_TO_DWORD
	SINT	X		WCHAR_TO_SINT
	USINT	X		WCHAR_TO_USINT
	INT	X		WCHAR_TO_INT
	UINT	X		WCHAR_TO_UINT
	DINT	X		WCHAR_TO_DINT
	UDINT	X		WCHAR_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	X	The value is converted to the first character in the character string (WSTRING). If the length of the character string is not defined, the length "1" is set after conversion. If the length of the character string is defined, it remains unchanged after conversion.	WCHAR_TO_WSTRING
	CHAR	X		WCHAR_TO_CHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.

Explicit conversion of STRING (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the STRING data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
STRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are permitted for conversion: • Digit • Sign • Dot The first character of the string may be a sign (+, -) or a number. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousands separator is permitted to the left of the decimal point but will be ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	STRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		STRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		STRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		STRING_TO_UINT, S_CONV, STRG_VAL
	DINT	X		STRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		STRING_TO_UDINT, S_CONV, STRG_VAL
	REAL	X		STRING_TO_REAL, S_CONV, STRG_VAL
	LREAL	X		STRING_TO_LREAL, S_CONV, STRG_VAL
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	CHAR	X	The first character in the character string (STRING) is transferred to the destination data type. If the string is empty, then the value "0" will be written in the destination data type.	STRING_TO_CHAR, S_CONV
	WCHAR	-	No explicit conversion	-
	WSTRING	X		STRING_TO_WSTRING
x: Conversion possible - : Conversion not possible				

See also

STRING (Page 264)
Overview of data type conversion (Page 489)

Explicit conversion of WSTRING (S7-1200)

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the WSTRING data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
WSTRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are permitted for conversion: • Digit • Sign • Dot The first character of the string may be a sign (+, -) or a number. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousands separator is permitted to the left of the decimal point but will be ignored. If the layout of the character string is invalid for the conversion, or an overflow occurs, the enable output ENO will be set to "0".	WSTRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		WSTRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		WSTRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		WSTRING_TO_UINT, S_CONV, STRG_VAL
	DINT	X		WSTRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		WSTRING_TO_UDINT, S_CONV, STRG_VAL
	REAL	X		WSTRING_TO_REAL, S_CONV, STRG_VAL
	LREAL	X		WSTRING_TO_LREAL, S_CONV, STRG_VAL
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	CHAR	-		-
	WCHAR	X	The first character in the character string (WSTRING) is transferred to the target data type. If the string is empty, the value "0" will be written in the target data type.	WSTRING_TO_WCHAR
	STRING	X		WSTRING_TO_STRING

x: Conversion possible
- : Conversion not possible

3.17 Data type conversion for S7-300, S7-400 (S7-300, S7-400)

3.17.1 Overview of data type conversions (S7-300, S7-400)

Introduction

If you combine several operands in an instruction, you must make sure that the data types are compatible. This also applies when you assign or supply block parameters. If the operands are not of the same data type, a conversion has to be carried out.

There are two options for conversion:

- Implicit conversion
  Implicit conversion is supported by the programming languages LAD, FBD, SCL and GRAPH. Implicit conversion is not possible in the STL programming language.
- Explicit conversion

Note

Converting bit strings to SCL

All bit strings (BYTE, WORD, and DWORD) are handled like the corresponding unsigned integers (USINT, UINT, and UDINT) in expressions. Therefore, implicit conversion from DWORD to REAL is carried out like a conversion from UDINT to REAL, for example.

Implicit conversion

Implicit conversion is executed automatically if the data types of the operands are compatible. This compatibility test can be carried out according to strict or less strict criteria:

- With IEC check
  The following rules apply in the LAD, FBD and GRAPH programming languages when the IEC check has been set:
  - The only data types for which implicit conversion is possible are BYTE and WORD.
  - The bit length of the source data type may not exceed the bit length of the target data type. A WORD data type operand, for example, cannot be specified at a parameter at which the BYTE data type is expected.

  The following rules apply in the SCL programming language when the IEC check has been set:
  - Implicit conversion of bit strings into other data types is not possible. A WORD data type operand, for example, cannot be specified at a parameter at which the INT data type is expected.
  - The bit length of the source data type may not exceed the bit length of the target data type. A WORD data type operand, for example, cannot be specified at a parameter at which the BYTE data type is expected.
- Without IEC check (default setting)
  The following rules apply in the LAD, FBD and GRAPH programming languages when the IEC check has not been set:
  - Implicit conversion of data types BYTE, WORD, DWORD, INT, DINT, TIME, S5TIME, TOD, DATE and CHAR is possible.
  - The bit length of the source data type may not exceed the bit length of the target data type. A DWORD data type operand, for example, cannot be specified at a parameter at which the WORD data type is expected.

  The following rules apply in the SCL programming language when the IEC check has not been set:
  - Bit strings can be implicitly converted into other data types. A WORD data type operand can, for example, be given at a parameter at which the WORD data type is expected.
  - Bit strings cannot be implicitly converted into floating-point numbers. A WORD data type operand, for example, cannot be specified at a parameter at which the REAL data type is expected.
  - Bit strings can only be implicitly converted into the data types TIME, TOD, DATE and CHAR if these have the same bit length. A DWORD data type operand, for example, cannot be specified at a parameter at which the DATE data type is expected.
  - The bit length of the source data type may not exceed the bit length of the target data type. A DINT data type operand, for example, cannot be specified at a parameter at which the INT data type is expected.
  - The bit length of an operand entered at in-out parameters must be the same as the programmed bit length for the parameter in question.

Explicit conversion

If the operands are not compatible and implicit conversion is therefore not possible, you can use an explicit conversion instruction. You can find the conversion instructions in the "Instructions" task card.

Any overflow will be displayed on the ENO enable output. An overflow occurs, for example, if the value of the source data type is greater than the value of the target data type.

Note

Shifting of bit patterns

If the explicit conversion involves shifting a bit pattern, the enable output ENO is not set.

You can find additional information on explicit conversion under "See also".

The following figure shows an example in which explicit data type conversion must be carried out:

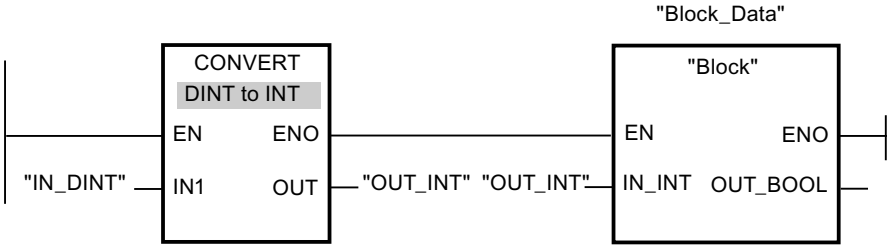

The "Block" function block expects a tag of the INT data type at the "IN_INT" input parameter. The value of the "IN_DINT" tag has therefore to the converted first from DINT to INT. Conversion is performed if the value of the "IN_DINT" tag is within the admissible value range for the INT data type. Otherwise an overflow is reported. However, a conversion takes place even in the event of an overflow, but the values are truncated and the ENO enable output is set to "0".

See also

Activate or deactivate IEC check (Page 557)

Implicit conversions (Page 557)

Explicit conversions (Page 569)

3.17.2 Implicit conversions (S7-300, S7-400)

3.17.2.1 Activate or deactivate IEC check (S7-300, S7-400)

The data types of the tags used are checked for compatibility. This compatibility test can be carried out according to criteria that are more or less strict. If "IEC check for code blocks" is activated, stricter criteria are applied.

You can set the IEC check centrally for all new blocks of the project or for individual blocks.

Setting IEC check for new blocks

To set the IEC check for all new blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. Select or clear the "IEC check for code blocks" check box in the "Default settings for new blocks" group.
   The IEC check is enabled or disabled for all new blocks in the program.

Setting IEC check for a block

To set the IEC check for a block, proceed as follows:

1. Open the block.
2. Open the "Properties" tab in the inspector window.
3. Select the "Attributes" group in the area navigation.
4. Select or clear the "IEC check" check box.
   The IEC check is enabled or disabled for this block. The setting is stored together with the project.

See also

Overview of data type conversions (Page 555)

3.17.2.2 Binary numbers (S7-300, S7-400)

Implicit conversion of BOOL (S7-300, S7-400)

Implicit conversion options

The BOOL data type cannot be implicitly converted.

See also

BOOL (bit) (Page 236)

Bit strings (S7-300, S7-400)

Implicit conversion of BYTE (S7-300, S7-400)

Implicit conversion options

The following table shows the options for the implicit conversion of BYTE data type:

Source	Target	With IEC check	Without IEC check	Explanation
BYTE	BOOL	-	-	No implicit conversion
	WORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	DWORD	X	X	
	INT	-	-	No implicit conversion
	DINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	REAL	-	-	No implicit conversion
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged to the target data type.
x: Conversion possible -: Conversion not possible				

See also

BYTE (Page 236)

Overview of data type conversions (Page 555)

Explicit conversion of BYTE (Page 570)

Activate or deactivate IEC check (Page 557)

Implicit conversion of WORD (S7-300, S7-400)

Implicit conversion options

The following table shows the options for the implicit conversion of WORD data type:

Source	Target	With IEC check	Without IEC check	Explanation
WORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	Only the low byte is transferred to the destination data type.
	DWORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	INT	-	X	
	DINT	-	X	
	REAL	-	-	No implicit conversion
	TIME	-	-	
	S5TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	DT	-	-	No implicit conversion
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	STRING	-	-	No implicit conversion
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

WORD (Page 237)
Overview of data type conversions (Page 555)
Explicit conversion of WORD (Page 572)
Activate or deactivate IEC check (Page 557)

Implicit conversion of DWORD (S7-300, S7-400)

Implicit conversion options

The following table shows the options for the implicit conversion of DWORD data type:

Source	Target	With IEC check	Without IEC check	Explanation
DWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	WORD	-	X	
	INT	-	-	No implicit conversion
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	REAL	-	-	No implicit conversion
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	S5TIME	-	-	No implicit conversion
	DT	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DWORD (Page 238)
Overview of data type conversions (Page 555)
Explicit conversion of DWORD (Page 574)
Activate or deactivate IEC check (Page 557)

3.17.2.3 Integers (S7-300, S7-400)

Implicit conversion of INT (S7-300, S7-400)

Options for implicit conversion

The following table shows the options for the implicit conversion of INT data type:

Source	Target	With IEC check	Without IEC check	Explanation
INT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	DWORD	-	-	No implicit conversion
	DINT [1)]	X	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	REAL	X	X	
	TIME	-	-	No implicit conversion
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	

x: Conversion possible
-: Conversion not possible
[1)]: Only possible in SCL

See also

INT (16-bit integers) (Page 243)
Overview of data type conversions (Page 555)
Explicit conversion of INT (Page 577)
Activate or deactivate IEC check (Page 557)

Implicit conversion of DINT (S7-300, S7-400)

Implicit conversion options

The following table shows the options for the implicit conversion of DINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
DINT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	INT	-	-	No implicit conversion
	REAL	-	-	
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	S5TIME	-	-	No implicit conversion
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DINT (32-bit integers) (Page 245)
Overview of data type conversions (Page 555)
Explicit conversion of STRING (Page 590)
Activate or deactivate IEC check (Page 557)

3.17.2.4 Floating-point numbers (S7-300, S7-400)

Implicit conversion of REAL (S7-300, S7-400)

Implicit conversion options

The REAL data type cannot be implicitly converted.

See also

REAL (Page 251)
Overview of data type conversions (Page 555)
Explicit conversion of CHAR (Page 589)
Activate or deactivate IEC check (Page 557)

3.17.2.5 Timers (S7-300, S7-400)

Implicit conversion of TIME (S7-300, S7-400)

Implicit conversion options

The following table shows the options for the implicit conversion of TIME data type:

Source	Target	With IEC check	Without IEC check	Explanation
TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	INT	-	-	No implicit conversion
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	REAL	-	-	No implicit conversion
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME (IEC time) (Page 256)
Overview of data type conversions (Page 555)
Explicit conversion of TIME (Page 584)
Activate or deactivate IEC check (Page 557)

Implicit conversion of S5TIME (S7-300, S7-400)

Implicit conversion options

The following table shows the options for the implicit conversion of S5TIME data type:

Source	Target	With IEC check	Without IEC check	Explanation
S5TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	DWORD	-	-	No implicit conversion
	INT	-	-	
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

S5TIME (duration) (Page 255)
Overview of data type conversions (Page 555)
Explicit conversion of S5TIME (Page 585)
Activate or deactivate IEC check (Page 557)

3.17.2.6 Date and time (S7-300, S7-400)

Implicit conversion of DATE (S7-300, S7-400)

Implicit conversion options

The following table shows the options for the implicit conversion of DATE data type:

Source	Target	With IEC check	Without IEC check	Explanation
DATE	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of days since 01/01/1990.
	DWORD	-	-	No implicit conversion
	INT	-	-	
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DATE (Page 258)
Overview of data type conversions (Page 555)
Activate or deactivate IEC check (Page 557)

Implicit conversion of TOD (S7-300, S7-400)

Implicit conversion options

The following table shows the options for the implicit conversion of TOD data type:

Source	Target	With IEC check	Without IEC check	Explanation
TOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	INT	-	-	No implicit conversion
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME_OF_DAY (TOD) (Page 258)
Overview of data type conversions (Page 555)
Activate or deactivate IEC check (Page 557)

Implicit conversion of DT (S7-300, S7-400)

Implicit conversion options

The DT data type cannot be implicitly converted.

See also

DATE_AND_TIME (date and time of day) (Page 259)
Overview of data type conversions (Page 555)

Explicit conversion of DT (Page 588)
Activate or deactivate IEC check (Page 557)

3.17.2.7 Character strings (S7-300, S7-400)

Implicit conversion of CHAR (S7-300, S7-400)

Implicit conversion options

The following table shows the options for the implicit conversion of CHAR data type:

Source	Target	With IEC check	Without IEC check	Explanation
CHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	WORD	-	-	No implicit conversion
	DWORD	-	-	
	INT	-	-	
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	X	X	The bit pattern of the source value is transferred unchanged to the destination data type.
x: Conversion possible -: Conversion not possible				

See also

CHAR (Page 263)
Overview of data type conversions (Page 555)
Explicit conversion of CHAR (Page 589)
Activate or deactivate IEC check (Page 557)

Implicit conversion of STRING (S7-300, S7-400)

Implicit conversion options

The STRING data type cannot be implicitly converted.

See also

STRING (Page 264)
Overview of data type conversions (Page 555)
Explicit conversion of STRING (Page 590)
Activate or deactivate IEC check (Page 557)

3.17.3 Explicit conversions (S7-300, S7-400)

3.17.3.1 Binary numbers (S7-300, S7-400)

Explicit conversion of BOOL (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BOOL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
BOOL	BYTE	X	Only the LSB (Least Significant Bit) is set in the destination data type. The enable output ENO is always "1".	CONVERT
	WORD	X		
	DWORD	X		
	INT	X		BOOL_TO_INT
	DINT	X		BOOL_TO_DINT
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

Overview of data type conversions (Page 555)
BOOL (bit) (Page 236)
Implicit conversions (Page 557)

Bit strings (S7-300, S7-400)

Explicit conversion of BYTE (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BYTE data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
BYTE	BOOL	X	The following possibilities can occur: • If the source is "0", the target data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the target data type is set according to LSB and the enable output ENO is "0".	CONVERT
	WORD	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. The remaining bits are set to "0".	CONVERT
	DWORD	X		
	INT	X		
	DINT	X		
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. The remaining bits are set to "0".	CONVERT
x: Conversion possible - : Conversion not possible				

See also

BYTE (Page 236)
Overview of data type conversions (Page 555)
Implicit conversion of BYTE (Page 559)

Explicit conversion of WORD (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WORD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
WORD	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO is "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	CONVERT
	BYTE	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If the permitted value range of the destination data type is violated, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	
	DWORD	X		
	INT	X		
	DINT	X		
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	MOVE
	DT	-	No explicit conversion	-
	TOD	-		
	DATE	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	MOVE
	STRING	-	No explicit conversion	-
	CHAR	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	WORD_TO_CHAR
	BLOCK_DB	X	The bit pattern of WORD is interpreted as a data block number.	WORD_TO_BLOCK_DB

Source	Target	Conversion	Explanation	Mnemonics of the instruction
WORD_BCD[1) 2)]	INT	X	If the permitted value range of the destination data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	WORD_BCD_TO_INT
BCD[1) 2)]	INT	X		BCD_TO_INT

x: Conversion possible

- : Conversion not possible

[1)] The value to be converted has data type WORD and is accepted as a BCD-coded value between -999 and +999. The result is available after conversion as an integer (in binary notation) of the type INT.

[2)] Valid for SCL: If the BCD code contains a pseudo tetrad, the CPU reports a programming error and opens the organization block "OB121". If the organization block OB121 is not present, the CPU changes to STOP mode.

See also

WORD (Page 237)

Overview of data type conversions (Page 555)

Implicit conversion of WORD (Page 560)

Explicit conversion of DWORD (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DWORD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DWORD	BOOL	X	The following possibilities can occur: • If the source is "0", the target data type is also "0" and the enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO is "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	CONVERT
	BYTE	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	
	WORD	X		

	INT	X	ENO = TRUE
			#int1 := DWORD_TO_INT(16#FFFF_FFFF); // -1 bis
			#int1 := DWORD_TO_INT(16#FFFF_8000); // -32768
			#int1 := DWORD_TO_INT(16#0); // 0 bis
			#int1 := DWORD_TO_INT(16#0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := DWORD_TO_INT(16#FFFF_7FFF); // -32769
			#int1 := DWORD_TO_INT(16#8000_0000); // -2147483648
			#int1 := DWORD_TO_INT(16#8000); // 32768 bis
			#int1 := DWORD_TO_INT(16#7FFF_FFFF); // 2147483647	

	DINT	X	The bit pattern of the source value is transferred un-	MOVE
	REAL	X		

Source	Target	Conversion	Explanation	Mnemonics of the instruction
	TIME	X	changed, right-justified, to the target data type.	
	S5TIME	-	No explicit conversion	-
	DT	-		
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	MOVE
	DATE	-	No explicit conversion	-
	STRING	-		
	CHAR	-		
DWORD_BCD[1) 2)]	DINT	X	If the permitted value range of the destination data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	DWORD_BCD_TO_DINT
BCD[1) 2)]	DINT	X		BCD_TO_DINT

x: Conversion possible

- : Conversion not possible

[1)] The value to be converted has data type DWORD and is accepted as a BCD-coded value between -9999999 and +9999999. The result is available after conversion as an integer (in binary notation) of the type DINT.

[2)] Valid for SCL: If the BCD code contains a pseudo tetrad, the CPU reports a programming error and opens the organization block "OB121". If the organization block OB121 is not present, the CPU changes to STOP mode.

See also

DWORD (Page 238)

Overview of data type conversions (Page 555)

Implicit conversion of DWORD (Page 561)

3.17.3.2 Integers (S7-300, S7-400)

Explicit conversion of INT (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the INT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
INT	BOOL	X	The value is first converted to WORD and then to BOOL. The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO is "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	INT_TO_BOOL
	BYTE	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. The enable output ENO is set to "0" if a negative value is converted to an unsigned destination data type or if overflow occurs.	CONVERT
	WORD	X		MOVE
	DWORD	X		
	DINT	X		CONVERT, ITD
	REAL	X	The value is converted into the format of the destination data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) into the value "1.0").	CONVERT, SCALE, ITR
	TIME	-	No explicit conversion	-
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	S_CONV, CONVERT
	CHAR	X	The bit pattern of the source value is transferred unchanged to the destination data type. With the conversion of negative values or with an overflow, the enable output ENO will be set to "0".	CONVERT
	BCD[1) 2)]	X	If the permitted value range of the destination data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	INT_TO_BCD
	BCD_WORD [1) 2)]	X		INT_TO_BCD_WORD

Source	Target	Conversion	Explanation	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible 1) The value to be converted has type INT and is accepted as an integer with a value between -999 and +999. The result is available after conversion as a BCD-coded number of the type WORD. 2) Valid for SCL: If the BCD code contains a pseudo tetrad, the CPU reports a programming error and opens the organization block "OB121". If the organization block OB121 is not present, the CPU changes to STOP mode.				

See also

INT (16-bit integers) (Page 243)

Overview of data type conversions (Page 555)

Implicit conversion of INT (Page 562)

Explicit conversion of DINT (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DINT	BOOL	X	The value is first converted to DWORD and then to BOOL. The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO is "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	DINT_TO_BOOL
	BYTE	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type. The enable output ENO is set to "0" if a negative value is converted to an unsigned destination data type or if overflow occurs.	CONVERT
	WORD	X		
	DWORD	X		MOVE
	INT	X		CONVERT
	REAL	X	The value is converted into the format of the destination data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) into the value "1.0").	CONVERT, DTR, SCALE
	TIME	X	The bit pattern of the source value is transferred unchanged to the destination data type.	T_CONV, MOVE
	S5TIME	-	No explicit conversion	-
	DT	-		
	TOD	X	The bit pattern of the source value is transferred unchanged, right-justified, to the target data type.	CONVERT
	DATE	X		
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	S_CONV, CONVERT
	CHAR	X	The bit pattern of the source value is transferred unchanged to the destination data type. With the conversion of negative values or with an overflow, the enable output ENO will be set to "0".	DINT_TO_CHAR
	BCD[1) 2)]	X	If the permitted value range of the destination data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	DINT_TO_BCD
	BCD_DWORD[1) 2)]	X		DINT_TO_BCD_DWORD

Source	Target	Conversion	Explanation	Mnemonics of the instruction

x: Conversion possible

- : Conversion not possible

[1] The value to be converted has data type DWORD and is accepted as a BCD-coded value between -9999999 and +9999999. The result is available after conversion as an integer (in binary notation) of the type DINT.

[2] Valid for SCL: If the BCD code contains a pseudo tetrad, the CPU reports a programming error and opens the organization block "OB121". If the organization block OB121 is not present, the CPU changes to STOP mode.

See also

DINT (32-bit integers) (Page 245)

Overview of data type conversions (Page 555)

Implicit conversions (Page 557)

3.17.3.3 Floating-point numbers (S7-300, S7-400)

Explicit conversion of REAL (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the REAL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
REAL	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	X	The bit pattern of the source value is transferred unchanged to the destination data type.	CONVERT, MOVE
	INT	X	The value is converted into the destination data type. The result of conversion depends on the instruction employed. The enable output ENO is set to "0" if the admissible range of values of the destination data type is exceeded during conversion or the value to be converted has an invalid floating-point number.	CONVERT, ROUND, RND, CEIL, RND+, FLOOR, RND-, TRUNC, UNSCALE
	DINT	X		
	TIME	-	No explicit conversion	-
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	X	The value is converted into a character string. The enable output ENO is set to "0" if the character string length is exceeded or the value to be converted has an invalid floating-point number.	S_CONV, CONVERT
	CHAR	-	No explicit conversion	-
x: Conversion possible - : Conversion not possible				

See also

REAL (Page 251)

Overview of data type conversions (Page 555)

Implicit conversions (Page 557)

3.17.3.4 Timer operations (S7-300, S7-400)

Explicit conversion of TIME (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TIME data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
TIME	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.	CONVERT
	INT	-	No explicit conversion	-
	DINT	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.	T_CONV, CONVERT
	REAL	-	No explicit conversion	-
	S5TIME	X	The value is converted to the destination data type format. The enable output ENO is set to "0" in the event of overflow.	T_CONV, CONVERT
	DT	-	No explicit conversion	-
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

TIME (IEC time) (Page 256)

Overview of data type conversions (Page 555)

Implicit conversion of TIME (Page 564)

Explicit conversion of S5TIME (S7-300, S7-400)

Options for explicit conversion in LAD, FBD, STL and GRAPH

The following table shows the options and instructions for the explicit conversion of the S5TIME data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
S5TIME	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	X	The value is converted to the destination data type format.	S5TIME_TO_WORD
	DWORD	-	No explicit conversion	-
	INT	-		
	DINT	-		
	REAL	-		
	TIME	X	The value is converted to the destination data type format.	T_CONV, CONVERT
	DT	-	No explicit conversion	-
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

S5TIME (duration) (Page 255)

Overview of data type conversions (Page 555)

Implicit conversion of S5TIME (Page 565)

3.17.3.5 Date and time (S7-300, S7-400)

Explicit conversion of DATE (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DATE data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DATE	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	X	The value is converted to the destination data type format.	DATE_TO_WORD
	DWORD	-	No explicit conversion	-
	INT	X	The value is converted to the destination data type format.	DATE_TO_INT
	DINT	X		DATE_TO_DINT
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

DATE (Page 258)

Overview of data type conversions (Page 555)

Implicit conversion of DATE (Page 566)

Explicit conversion of TOD (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TOD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
TOD	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	X	The value is converted to the destination data type format.	TOD_TO_DWORD
	INT	-	No explicit conversion	-
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TOD_TO_DINT
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

TIME_OF_DAY (TOD) (Page 258)

Overview of data type conversions (Page 555)

Implicit conversion of TOD (Page 567)

Explicit conversion of DT (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DT	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	-		
	INT	-		
	DINT	-		
	REAL	-		
	TIME	-		
	S5TIME	-		
	TOD	X	During conversion, times are extracted from the DTL format and transferred to the destination data type.	T_CONV, CONVERT
	DATE	X	During the conversion, the date information is extracted from the DTL format and transferred to the destination data type.	
	STRING	-	No explicit conversion	-
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

DATE_AND_TIME (date and time of day) (Page 259)

Overview of data type conversions (Page 555)

3.17.3.6 String (S7-300, S7-400)

Explicit conversion of CHAR (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the CHAR data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
CHAR	BOOL	-	No explicit conversion	-
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	CONVERT
	WORD	X		
	DWORD	X		
	INT	X		
	DINT	X		
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	X	The value is converted into the first character in the character string (STRING). The length "1" is set after conversion if the character string length is not defined. If the character string length is defined, it remains unchanged following conversion.	S_CONV, CONVERT
x: Conversion possible - : Conversion not possible				

See also

CHAR (Page 263)

Overview of data type conversions (Page 555)

Implicit conversion of CHAR (Page 568)

Explicit conversion of STRING (S7-300, S7-400)

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the STRING data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
STRING	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	-		
	INT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are admissible for conversion: • Digit • Sign • Dot The first character in the character string may be a sign (+, -) or a digit. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator to the left of the decimal point is permitted but is ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	S_CONV, CONVERT
	DINT	X		
	REAL	X		
	TIME	-	No explicit conversion	-
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	CHAR	X	The first character in the character string (STRING) is transferred to the target data type. The value "0" is written to the target data type if the character string is empty.	S_CONV, CONVERT
x: Conversion possible - : Conversion not possible				

See also

STRING (Page 264)

Overview of data type conversions (Page 555)

Implicit conversions (Page 557)

3.17.3.7 Additional conversion functions (S7-300, S7-400)

Additional explicit conversion functions in SCL (S7-300, S7-400)

Additional options for explicit conversion in SCL

The following table shows the additional options and operations for explicit conversion in SCL:

Source	Target	Explanation	Mnemonics of the instruction
WORD	BLOCK_DB	The bit pattern of WORD is interpreted as a data block number.	WORD_TO_BLOCK_DB
BLOCK_DB	WORD	The data block number is interpreted as a bit pattern of WORD.	BLOCK_DB_TO_WORD

Instructions

4

4.1 Instructions

4.1.1 Example libraries

4.1.1.1 Sample Library for Instructions

The "Sample Library for Instructions" is the global library for the program examples using the instructions in STEP 7 (TIA Portal). This library offers you support during programming with its compact, easy to understand program examples in the programming languages LAD or SCL. The precondition for using the library is at least one CPU from the SIMATIC S7-1200 / S7-1500 series.

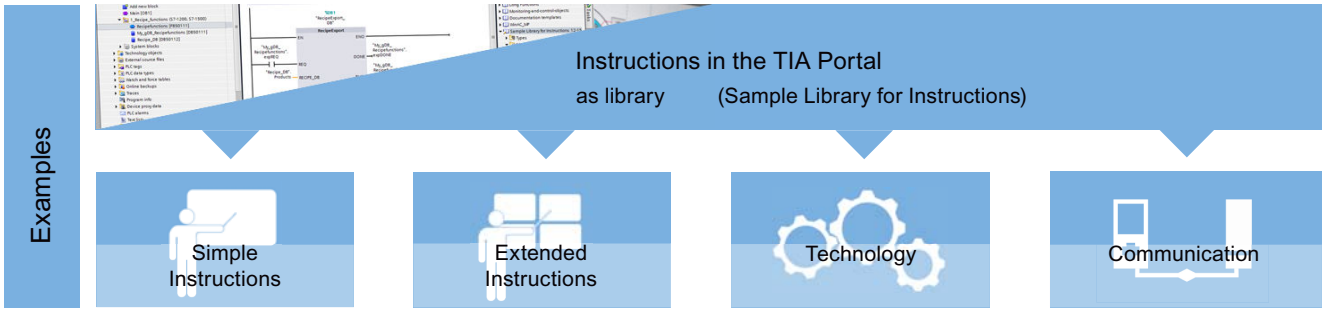

Programming examples in the TIA Portal information system

Programming examples in the "Sample Library for Instructions" match the instructions used in the programming examples in the TIA Portal information system. This means that you can call the respective programming example documentation via the help (<F1>) in the TIA Portal.

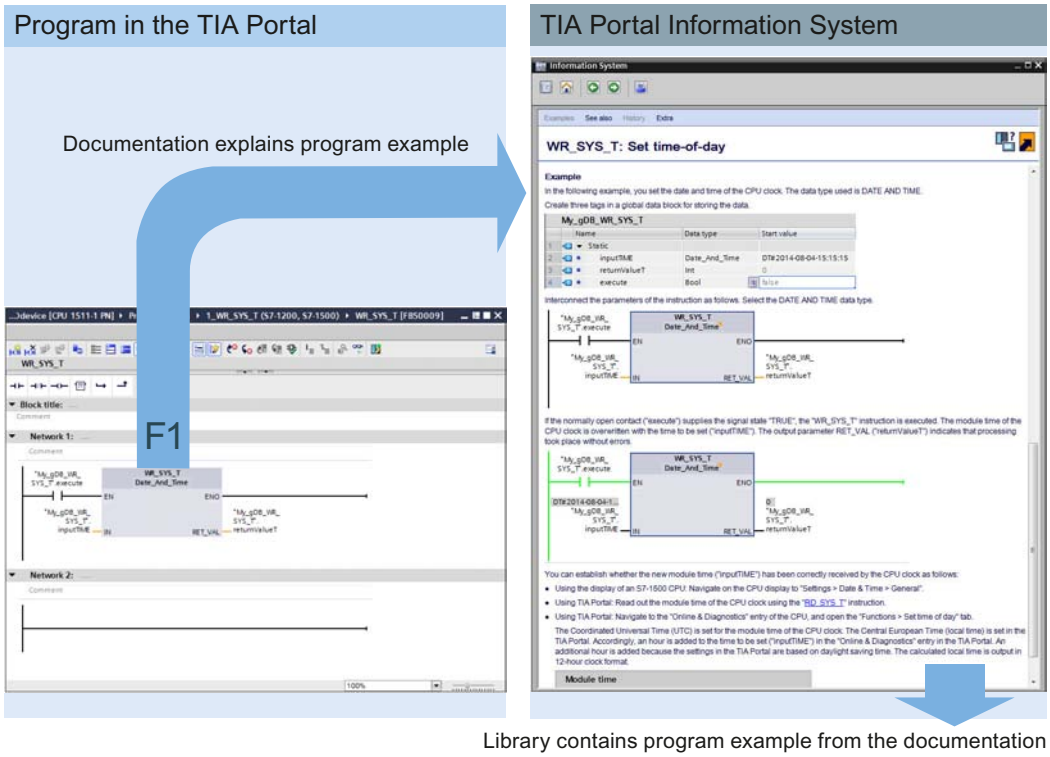

Downloading the library

The "Sample Library for Instructions" is available at the FAQ page (https://support.industry.siemens.com/cs/ww/en/view/109476781).

Note

Adding program examples to a project correctly

Due to the differing preconditions and functions, program examples may contain not only program blocks but also other components (e.g. PLC tags, PLC data types, watch tables or the like). There are also program examples that require special hardware (multiple CPUs, distributed I/O, input modules and the like).

Please observe the following rules:

- Never simply add all program examples or the entire "Sample Library for Instructions" folder to a CPU.
- Make sure that the corresponding program example does not consist of components for multiple CPUs.
- Always add the individual components of the program examples one at a time to the appropriate project folder. Also make sure that you select the appropriate CPU.

4.1.1.2 Library of General Functions

The "Library of General Functions" is the global library for supplementary functions for the instructions in STEP 7 (TIA Portal). To expand the scope of instructions, this library offers blocks with useful basic functions that are often required in automation projects. All contents of the library can be used with a CPU of the series SIMATIC S7-1200(F) / S7-1500(F).

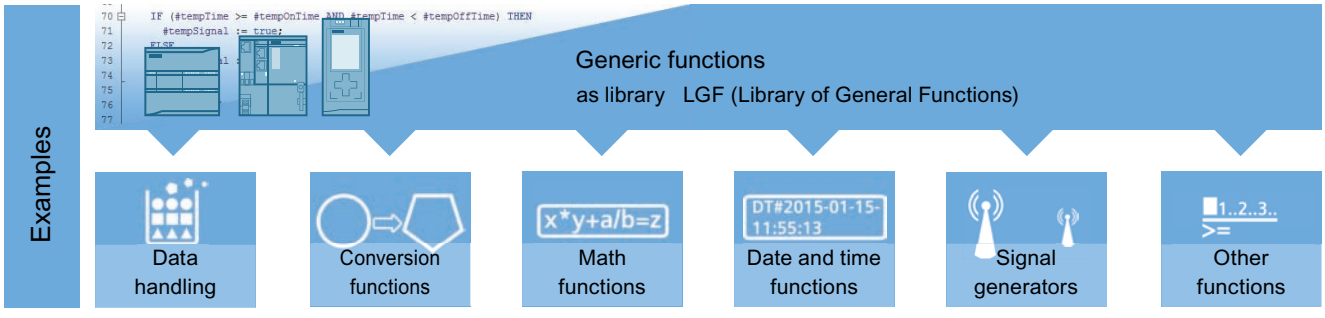

General functions

The general functions of the "Library of General Functions" are provided as blocks in the SCL programming language. The library includes the following functions: FIFO, search function, matrix calculations, astro clock. You can immediately use the functions by means of parameter assignment and implement them universally.

Downloading the library

The "Library of General Functions" is stored on the FAQ page (https://support.industry.siemens.com/cs/ww/en/view/109479728).

4.1.2 Asynchronous instructions

4.1.2.1 Difference between synchronous and asynchronous instructions

Asynchronous instructions

In program processing, a differentiation is made between synchronous and asynchronous instructions.

The "synchronous" and "asynchronous" properties relate to the temporal relationship between the call and execution of the instruction.

- The following applies to synchronous instructions: When the call of a synchronous instruction is ended, the execution is also ended.
- This is different in the case of asynchronous instructions: The execution of an asynchronous instruction can extend over multiple calls. The CPU processes asynchronous instructions in parallel with the cyclic user program. Asynchronous instructions occupy resources in the CPU while they are being processed.

Asynchronous instructions are usually instructions for transferring data (data records for modules, communication data, diagnostics data).

Additional information about asynchronous instructions can be found here: SIMATIC S7-1500 / ET 200MP Manual Collection (https://support.industry.siemens.com/cs/ww/en/view/86140384) in the system manual under Basics of program processing > Asynchronous instructions

Identification of the job

To execute an instruction over multiple calls, the CPU must be able to uniquely assign a subsequent call to a running job of the instruction. To assign a call to a job, the CPU uses one of the following two mechanisms, depending on the type of the instruction:

- The instance of the instruction (with type "SFB" system function block)
- The input parameter of the instruction identifying the job

If you use asynchronous instructions to trigger a process interrupt, output control commands to DP slaves, start a data transfer, or abort a non-configured connection and then call the instruction again before the current job is completed, the reaction of the instruction will depend on whether the second call involves the same job.

Passing parameters to program blocks that are to be executed asynchronously

Code blocks (FBs/FCs) and data blocks (DBs) can be created with different access types ("standard" and "optimized"). In code blocks, you can call any instructions. Some instructions (e.g. "WRIT_DBL" and "READ_DBL") are executed asynchronously. These blocks cannot be supplied with tags from TEMP because the data must not be changed during execution.

Ensure that you do not use these instructions in programs in which code blocks of different access types are called reciprocally. This could cause the following to occur:

- A structure from a standard data block is directly or indirectly passed to an optimized code block, which forwards this structure directly or indirectly to one of the blocks mentioned above.
- The reverse scenario, whereby a structure from an optimized code block is directly or indirectly passed to a standard data block, which forwards this structure directly or indirectly to one of the program blocks mentioned above.

Otherwise, a hidden copy of the passed data is created in TEMP, which leads to the asynchronous program blocks returning a negative acknowledgment.

Parameter REQ

The input parameter REQ (request) is used solely to start the job or prepare the start:

- Some instructions, such as RecipeExport and RecipeImport are activated by a positive edge at REQ. A preparatory call with REQ = 0 is necessary for these instructions before you make the first call with REQ = 1.
- Trigger the job by setting the input parameter REQ to "1" (case 1).
- If a particular job has been started and not yet completed and you call the instruction again to perform the same job (for example, in a cyclic interrupt OB), then REQ is not evaluated by the instruction (case 2).
- If the job is completed, but the input parameter REQ is still set to "1", the job is immediately started again.

Note

REQ bit = "1"

Please note that, when the REQ bit is set, the asynchronous instruction will be started again after the previous call is completed. This may not be desired. To make the project clearer and easier to maintain, it is therefore advisable to reset the REQ bit to "0" as early as possible.

Parameter RET_VAL and BUSY

The output parameters RET_VAL and BUSY indicate the status of the job.

Pay attention to the note in section: Evaluating errors with output parameter RET_VAL (Page 162)

- For a call with REQ = 0 and available system resources, W#16#7000 is entered in RET_VAL and BUSY is set to "0".

  Note

  Call with REQ = 0 when system resources are assigned

  If an asynchronous instruction with REQ = 0 is called for an S7-1500 CPU, all available resources are assigned at this time and this call not an intermediate call of a job that is already running, the error code W#16#80C3 (temporary resource error) is entered in RET_VAL.

  For an S7-300 CPU, the error code W#16#7000 (first call with REQ = 0) is entered in RET_VAL under these boundary conditions.

- In case 1 (first call with REQ = 1), code W#16#7001 is entered in RET_VAL and BUSY is set to "1" if system resources are available and the supply of the input parameters is correct. If the required system resources are currently being used or the input parameters contain an error, the corresponding error code is entered in RET_VAL and BUSY receives the value "0".
- In case 2 (intermediate call), the code W#16#7002 is entered in RET_VAL (this corresponds to the message that the call is currently still being processed) and BUSY is set to "1".
- The following applies to the last call for a job:
  - For instruction "DPNRM_DG (Page 2353)", the number of data in bytes will entered as integer in RET_VAL in case there are no errors in data transmission. BUSY has the value "0" in this case.
    If there is an error, then the error information will be entered in RET_VAL and you should not evaluate BUSY in this case.
  - For all other instructions, "0" will be entered in RET_VAL if the job was executed without errors and BUSY will receive the value "0". If there is an error, the error code will be entered in RET_VAL and BUSY will receive the value "0".

Note

If the first and last call coincide, the reaction for RET_VAL and BUSY is the same as for the last call.

Overview

The following table provides you with an overview of the relationships explained above. In particular, it shows the possible values of the output parameters if the execution of the job is not completed after an instruction call has been completed.

Note

Following every call, you must evaluate the relevant output parameters in your program.

Relationship between call, REQ, RET_VAL and BUSY during a "running" job.

Number of the call	Type of call	REQ	RET_VAL	BUSY	DONE	ERROR	ENO
0	Preparatory call	0	W#16#7000	0	0	0	True
1	First call	1	W#16#7001	1	0	0	True
			Error code	0	0	1	False
2 to (n - 1)	Intermediate call	irrelevant	W#16#7002	1	0	0	True
n	Last call	irrelevant	W#16#0000 (exceptions: RD_REC (Page 2332) and RD_DPARA (Page 2427): The successfully read data length is written to the parameter STATUS.)	0	1	0	True
			Error code if errors occurred	0	0	1	False

4.1.3 Basic instructions

4.1.3.1 LAD

Bit logic operations

---| |---: Normally open contact

Description

The activation of the normally open contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally open contact closes and the signal state at the output is set to the signal state of the input.

When the operand has signal state "0", the normally open contact is not activated and the signal state at the output of the instruction is reset to "0".

Two or more normally open contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally open contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Operand whose signal state is queried.

Example

The following example shows how the instruction works.

Create a global data block with the following content for this purpose:

Name of the block: SLI_gDB_NOContact	
Name	Data type
start1	BOOL
start2	BOOL
start3	BOOL
startOut	BOOL

Write the following program code:

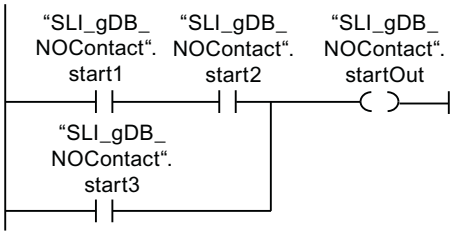

Operand "SLI_gDB_NOContact".startOut is set when one of the following conditions is fulfilled:

- The operands "SLI_gDB_NOContact".start1 and "SLI_gDB_NOContact".start2 have the signal state "1".
- The operand "SLI_gDB_NOContact".start3 has the signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

---| / |---: Normally closed contact

Description

The activation of the normally closed contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally closed contact opens and the signal state at the output of the instruction is reset to "0".

When the operand has signal state "0", the normally closed contact is not enabled and the signal state of the input is transferred to the output.

Two or more normally closed contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally closed contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

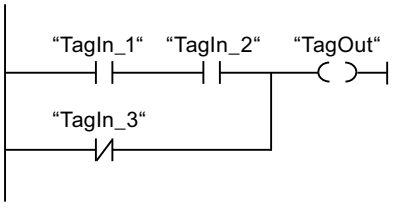

The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

--|NOT|--: Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). If the signal state is "1" at the input of the instruction, the output of the instruction has signal state "0". If the signal state is "0" at the input of the instruction, the output has the signal state "1".

Example

The following example shows how the instruction works:

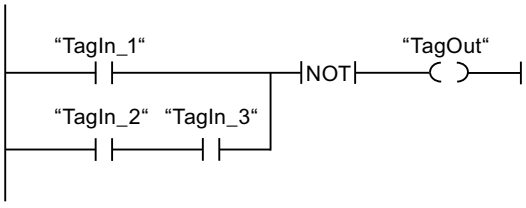

Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The signal state of the operands "TagIn_2" and "TagIn_3" is "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

---( )---: Assignment

Description

You can use the "Assignment" instruction to set the bit of a specified operand. If the result of logic operation (RLO) at the input of the coil has signal state "1", the specified operand is set to signal state "1". If the signal state is "0" at the input of the coil, the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output.

Parameter

The following table shows the parameters of the "Assignment" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:

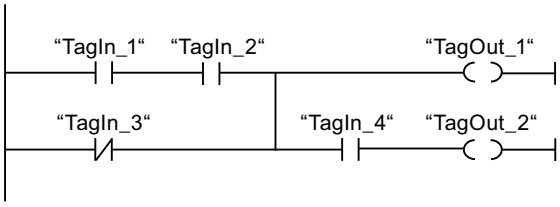

The "TagOut_1" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

The "TagOut_2" operand is set when one of the following conditions is fulfilled:

- Operands "TagIn_1", "TagIn_2", and "TagIn_4" have signal state "1".
- The signal state of the "TagIn_3" operand is "0" and the signal state of the "TagIn_4" operand is "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

--(/)--: Negate assignment

Description

The "Negate assignment" instruction inverts the result of logic operation (RLO) and assigns it to the specified operand. When the RLO at the input of the coil is "1", the operand is reset. When the RLO at the input of the coil is "0", the operand is set to signal state "1".

Parameter

The following table shows the parameters of the "Negate assignment" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:

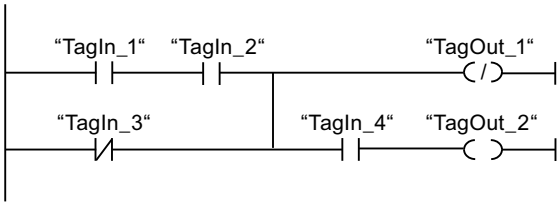

Operand "TagOut_1" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

---(R)---: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is reset to "0". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

Parameter

The following table shows the parameters of the "Reset output" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Operand that is reset when RLO = "1".

Example

The following example shows how the instruction works:

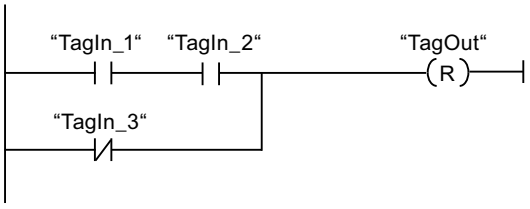

Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

---(S)---: Set output

Description

You can use the "Set output" instruction to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is set to "1". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

Parameter

The following table shows the parameters of the "Set output" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:

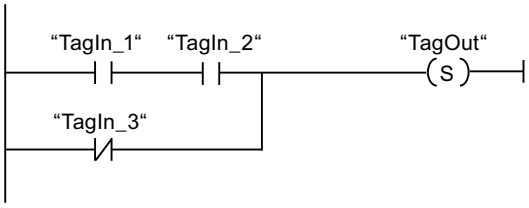

The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

SET_BF: Set bit field

Description

You use the "Set bit field" instruction to set several bits starting from a certain address.

You determine the number of bits to be set using the value of <Operand1>. The address of the first bit to be set is defined by <Operand2>. If the value of <Operand1> may not be greater than the number of bits in a selected byte. If this value is greater, the instruction is not executed and the error message "Range violation for index <Operand1>" is displayed. The bits remain set until they are explicitly reset by another instruction.

Specify <Operand1> at the operand placeholder below the instruction. Specify <Operand2> at the operand placeholder above the instruction.

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If the RLO at the input of the coil is "0", the instruction does not execute.

Bit fields of the type PLC data type, STRUCT or ARRAY

With structures of the type PLC data type, STRUCT or ARRAY, the number of bits contained in the structure represents the maximum number of bits that can be reset:

- If you specify the value "20" at <Operand1>, for example, and the structure only contains 10 bits, then only these 10 bits are set.
- If you specify the value "5" at <Operand1>, for example, and the structure contains 10 bits, then exactly 5 bits are set.

Parameter

The following table shows the parameters of the "Set bit field" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	UINT	Constant	Number of bits to be set
<Operand2>	Output	BOOL	I, Q, M In the case of a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be set.

Example

The following example shows how the instruction works:

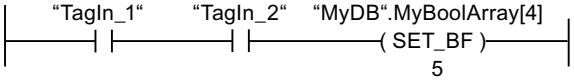

If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are set starting at the address of the operand "MyDB".MyBoolArray[4].

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

RESET_BF: Reset bit field

Description

You use the "Reset bit field" instruction to reset several bits starting from a certain address.

You specify the number of bits to be reset using the value of <Operand1>. The address of the first bit to be reset is defined by <Operand2>. If the value of <Operand 1> may not be greater than the number of bits in a selected byte. If this value is greater, the instruction is not executed

and the error message "Range violation for index <Operand 1>" is displayed. The bits remain set until they are explicitly reset by another instruction.

Specify <Operand1> at the operand placeholder below the instruction. Specify <Operand2> at the operand placeholder above the instruction.

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If the RLO at the input of the coil is "0", the instruction does not execute.

Bit fields of the type PLC data type, STRUCT or ARRAY

With structures of the type PLC data type, STRUCT or ARRAY, the number of bits contained in the structure represents the maximum number of bits that can be set:

- If you specify the value "20" at <Operand1>, for example, and the structure only contains 10 bits, then only these 10 bits are reset.
- If you specify the value "5" at <Operand1>, for example, and the structure contains 10 bits, then exactly 5 bits are reset.

Parameter

The following table shows the parameters of the "Reset bit field" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	UINT	Constant	Number of bits to be reset
<Operand2>	Output	BOOL	I, Q, M With a DB or an instance DB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be reset.

Example

The following example shows how the instruction works:

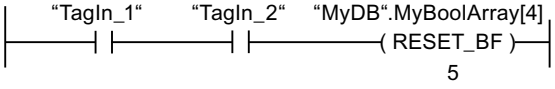

If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are reset starting at the address of the operand "MyDB".MyBoolArray[4]

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

SR: Set/reset flip-flop

Description

Use the instruction "Set/reset flip-flop" to set or reset the bit of the specified operand, depending on the signal state of the inputs S and R1. If the signal state is "1" at input S and "0" at input R1, the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Parameter

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
S	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Enable setting
R1	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable resetting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is set or reset.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:

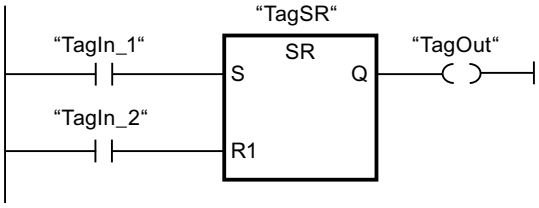

The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

RS: Reset/set flip-flop

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Parameter

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
R	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Enable resetting
S1	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable setting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is reset or set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:

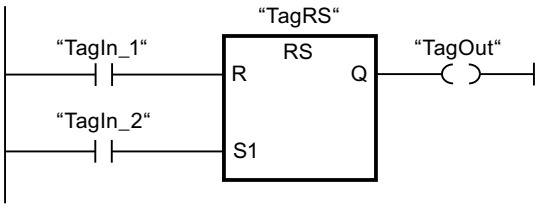

The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

--|P|--: Scan operand for positive signal edge

Description

You can use the "Scan operand for positive signal edge" instruction to determine if there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

The following figure shows the change in the signal state in case of a negative and a positive signal edge:

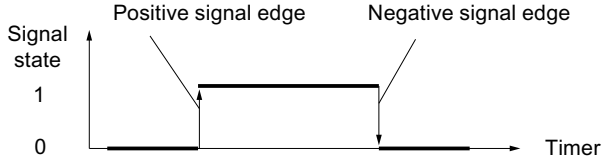

The positive signal edge is queried each time the instruction executes. When a positive signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameter

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:

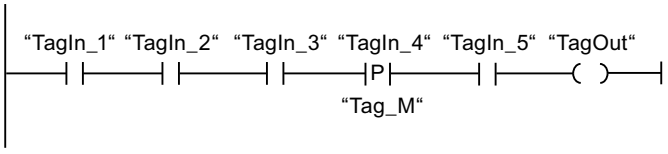

Operand "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a rising edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

--|N|--: Scan operand for negative signal edge

Description

You can use the "Scan operand for negative signal edge" instruction to determine if there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan that is saved in an edge memory bit <Operand2>. If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

The following figure shows the change in the signal state in case of a negative and a positive signal edge:

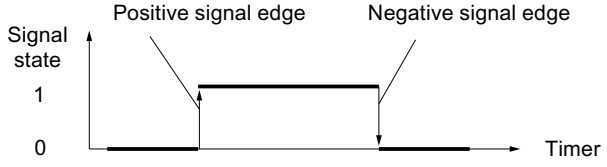

The negative signal edge is queried each time the instruction executes. When a negative signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameter

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:

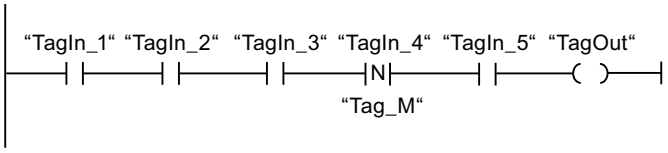

Operand "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a negative signal edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

--(P)--: Set operand on positive signal edge

Description

You can use the "Set operand on positive signal edge" instruction to set a specified operand (<Operand1>) when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand2>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

The positive signal edge is queried each time the instruction executes. When a positive signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

You specify the operand (<Operand1>) to be set in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameter

The following table shows the parameters of the instruction "Set operand on positive signal edge":

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set by a positive edge.
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows how the instruction works:

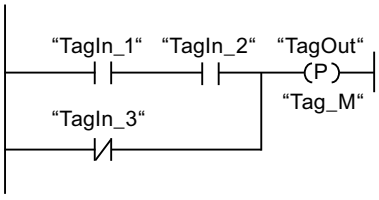

Operand "TagOut" is set for one program cycle, when the signal state at the input of the coil switches from "0" to "1" (positive signal edge). In all other cases, the operand "TagOut" has the signal state "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

--(N)--: Set operand on negative signal edge

Description

You can use the "Set operand on negative signal edge" instruction to set a specified operand (<Operand1>) when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand2>). If the instruction detects a change in the RLO from "1" to "0", there is a negative signal edge.

The negative signal edge is queried each time the instruction executes. When a negative signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

You specify the operand (<Operand1>) to be set in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameter

The following table shows the parameters of the instruction "Set operand on negative signal edge":

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set by a negative edge.
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows how the instruction works:

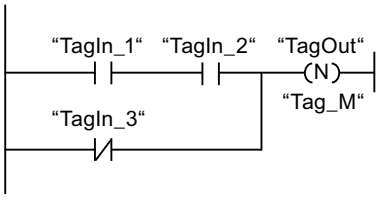

Operand "TagOut" is set for one program cycle, when the signal state at the input of the coil switches from "1" to "0" (negative signal edge). In all other cases, the operand "TagOut" has the signal state "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

P_TRIG: Scan RLO for positive signal edge

Description

Use the "Scan RLO for positive signal edge" instruction to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

The positive signal edge is queried each time the instruction executes. As soon as a positive single edge is detected, the output Q of the instruction returns the signal state "1" for the length of a program cycle. In all other cases, the output returns the signal state "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameter

The following table shows the parameters of the instruction "Scan RLO for positive signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L or constant	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

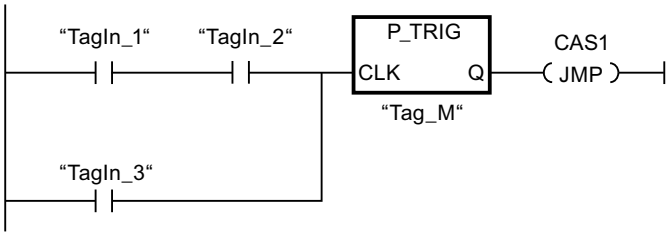

The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

N_TRIG: Scan RLO for negative signal edge

Description

Use the "Scan RLO for negative signal edge" instruction to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative signal edge.

The negative signal edge is queried each time the instruction executes. As soon as a negative single edge is detected, the output Q of the instruction returns the signal state "1" for the length of a program cycle. In all other cases, the signal state at the output of the instruction is "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameter

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L or constant	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

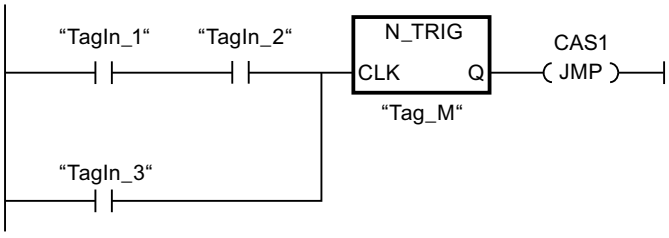

The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

R_TRIG: Detect positive signal edge

Description

With the "Detect positive signal edge" instruction, you can detect a state change from "0" to "1" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "0" to "1", a positive signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

Parameter

The following table shows the parameters of the "Detect positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

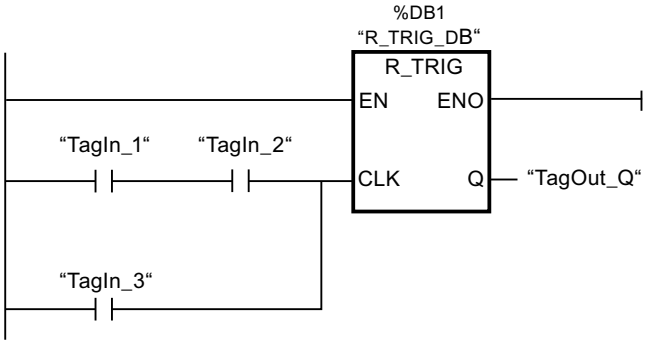

The previous state of the tag at the CLK input is stored in the "R_TRIG_DB" tag. If a change in the signal state from "0" to "1" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

F_TRIG: Detect negative signal edge

Description

With the "Detect negative signal edge" instruction, you can detect a state change from "1" to "0" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "1" to "0", a negative signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

Parameter

The following table shows the parameters of the "Detect negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

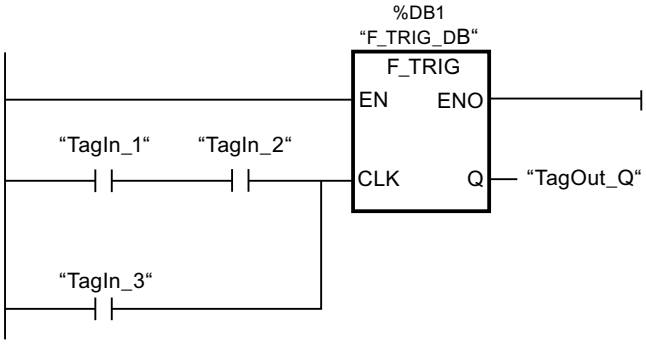

The previous state of the tag at the CLK input is stored in the "F_TRIG_DB" tag. If a change in the signal state from "1" to "0" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

Timer operations

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the Q output for a programmed duration. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. Output Q is set for the duration PT, regardless of the subsequent course of the input signal. Even if a new positive signal edge is detected, the signal state at the output Q is not affected as long as the PT time duration is running.

You can scan the current time value at the ET output. The timer value starts at T#0s and ends when the value of the time duration PT is reached. When the time PT has elapsed and the signal state at input IN is "0", the ET output is reset.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TP_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Generate pulse" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, P, or constant	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the pulse The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Pulse output
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:

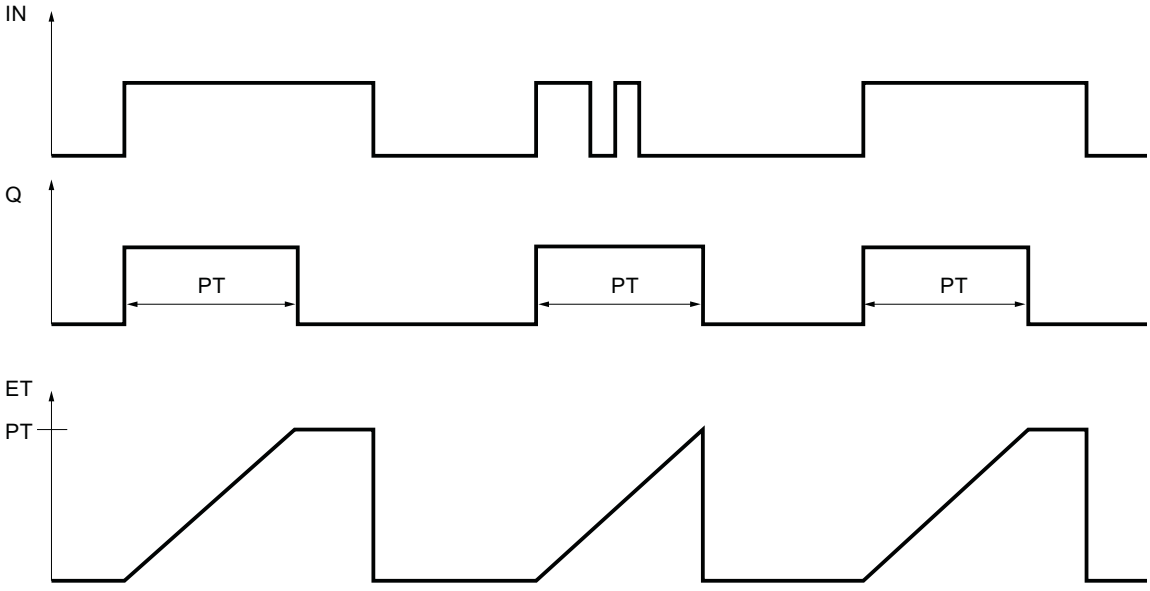

Example

The following example shows how the instruction works:

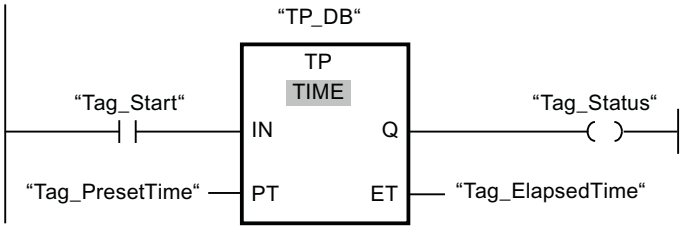

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Start	Signal transition "0" => "1"
PT	Tag_PresetTime	T#10s
Q	Tag_Status	TRUE
ET	Tag_ElapsedTime	from T#0s => T#10s

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTime" operand. When the time expires, the "Tag_Status" operand is reset to signal state "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Example of controlling room temperature (Page 7181)

Basic information on LAD (Page 7121)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the programmed time PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the time PT has elapsed, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

The current time value can be queried at the ET output. The timer value starts at T#0s and ends when the value of the time duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TON_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, P, or constant	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:

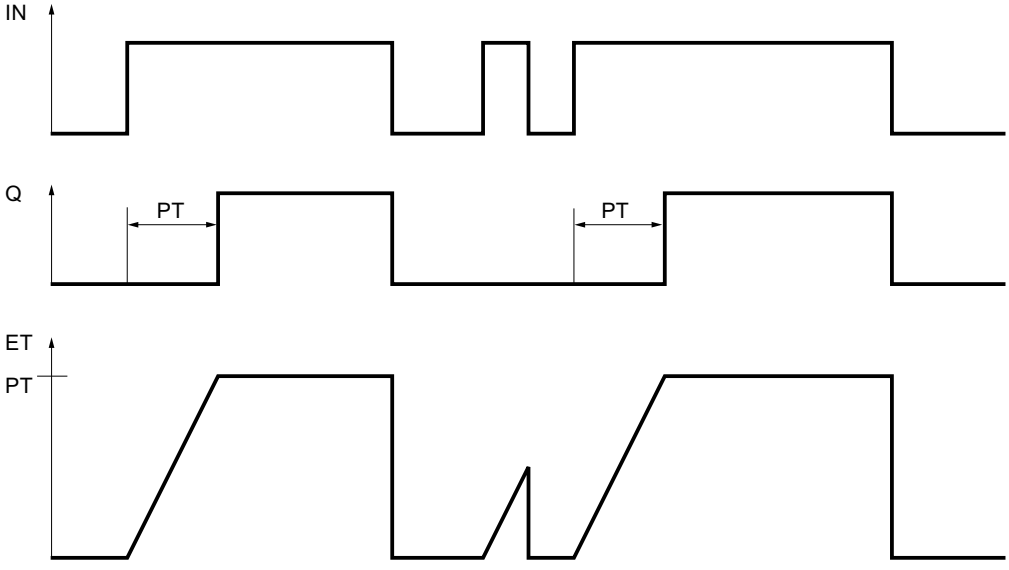

Example

The following example shows how the instruction works:

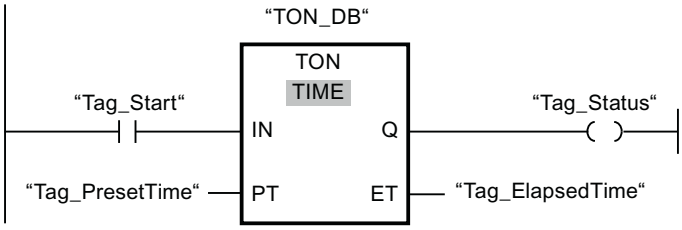

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Start	Signal transition "0" => "1"
PT	Tag_PresetTime	T#10s
Q	Tag_Status	FALSE; after 10 s => TRUE
ET	Tag_ElapsedTime	from T#0s => T#10s

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. When the time duration expires, the "Tag_Status" operand is set to the signal state "1". The Tag_Status operand remains set to "1" as long as the Tag_Start operand has signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand. When the signal state at the operand "Tag_Start" changes from "1" to "0", the "Tag_Status" operand is reset.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed time PT. The Q output is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). When the signal state at input IN changes back to "0", the programmed time PT starts. Output Q remains set as long as the time duration PT is running. When the PT time duration expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q continues to be "1".

The current time value can be queried at the ET output. The timer value starts at T#0s and ends when the value of the time duration PT is reached. When the PT time duration expires, the ET output remains set to the current value until input IN changes back to "1". If input IN switches to "1" before the duration PT has expired, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC Timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TOF_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Generate off-delay" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, P, or constant	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is reset when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate off-delay" instruction:

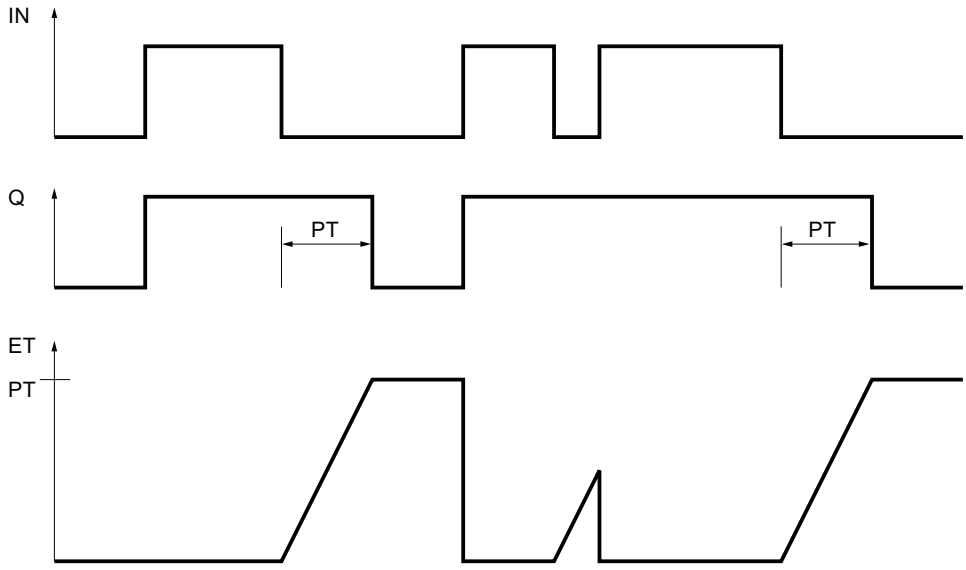

Example

The following example shows how the instruction works:

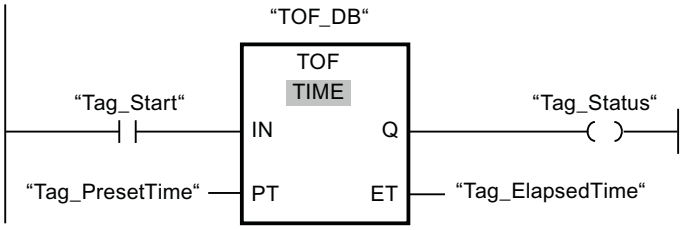

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Start	Signal transition "0" => "1"; signal transition "1" => "0"
PT	Tag_PresetTime	T#10s
Q	Tag_Status	TRUE
ET	Tag_ElapsedTime	from T#10s => T#0s

When the signal state of the operand "Tag_Start" changes from "0" to "1", the "Tag_Status" operand is set to signal state "1". When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set to TRUE. When the time has expired, the "Tag_Status" operand is reset to FALSE. The current time value is stored in the "Tag_ElapsedTime" operand.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

TONR: Time accumulator

Description

The instruction "Time accumulator" is used to accumulate time values within a period set by the parameter PT. When the signal state at the IN input changes from "0" to "1" (positive signal edge), the instruction is executed and time PT starts. While the time PT is running, the time values are accumulated that are recorded when the IN input has signal state "1". The accumulated time is written to the ET output and can be queried there. When the duration PT expires, the output Q has signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes from "1" to "0" (negative signal edge).

The R input resets the ET and Q outputs regardless of the signal state of the start input.

Each call of the "Time accumulator" instruction must be assigned an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TONR_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, P, or constant	Start input
R	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Reset input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Maximum duration of time recording The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Accumulated time

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:

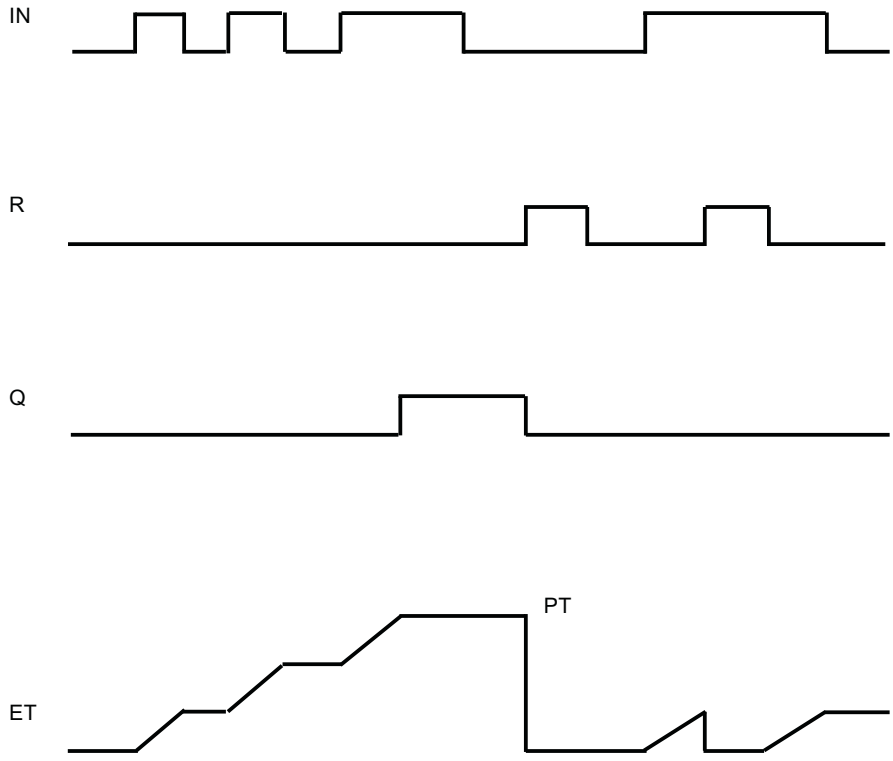

Example

The following example shows how the instruction works:

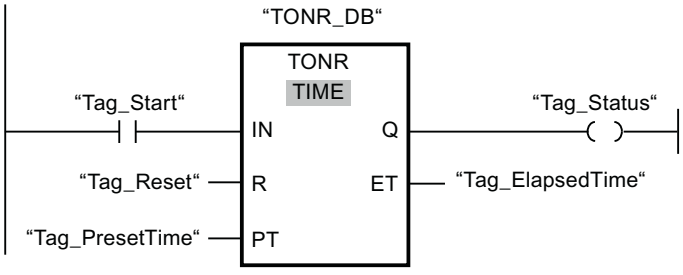

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Start	Signal transition "0" => "1"
PT	Tag_PresetTime	T#10s

Parameters	Operand	Value
Q	Tag_Status	FALSE; after 10 s => TRUE
ET	Tag_ElapsedTime	Signal transition from "0" => "1" The time T#10s expires. After 5 s signal transition from "1" => "0": The time at the "Tag_Elapsed-Time" operand remains at T#5s. After approx. 2 s, renewed signal transition from "0" => "1": The time at the "Tag_Elapsed-Time" operand continues to run at T#5s.

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. The time continues to run as long as the "Tag_Start" operand has the signal state "1". When the signal state of the "Tag_Start" operand changes from "1" to "0", the time stops and the current time value at the Tag_ElapsedTime operand is recorded. When the signal state at the "Tag_Start" operand changes again from "0" to "1", the time continues to run starting at the time value that was recorded at the transition from "1" to "0". When the time value specified for the PT parameter is reached, the "Tag_Status" operand is set to the signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(TP)---: Start pulse timer

Description

Use the "Start pulse timer" instruction to start an IEC timer with a specified duration as pulse. The IEC timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC timer runs for the specified time regardless of any subsequent changes in the RLO. The run of the IEC timer is also not affected by the detection of a new positive signal edge. As long as the IEC timer is running, the querying of the timer status for "1" returns the signal state "1". When the IEC timer has expired, the timer status returns the signal state "0".

Specify the duration of the pulse under the instruction at <Operand1> (duration) and specify the IEC time to be started above the instruction at <Operand2> (IEC time).

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER or TP_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_LTIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- The IEC_Timer structure is updated when the instruction is called. The time value at the ET output is only updated if the ET or Q outputs are scanned (e.g. "MyTimer".Q or "MyTimer".ET).
- When the assigned timer is accessed.

The current timer status is stored in the Q structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0".

The execution of the "Start pulse timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

The following table shows the parameters of the "Start pulse timer" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TP_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME	D, L	IEC timer that is started.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction:

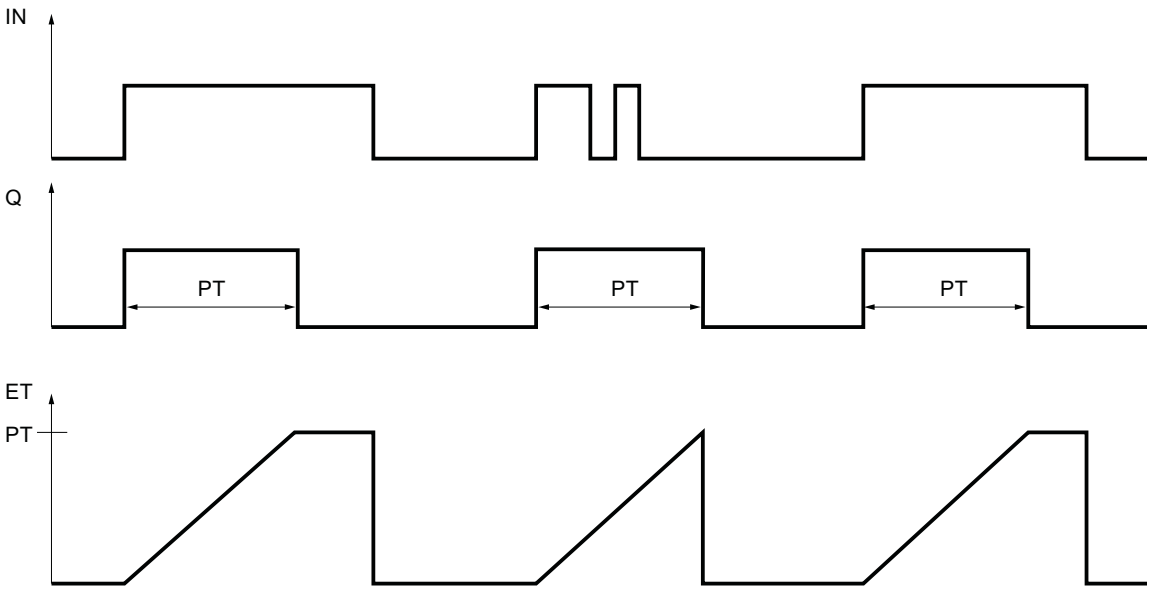

Example

The following example shows how the instruction works:

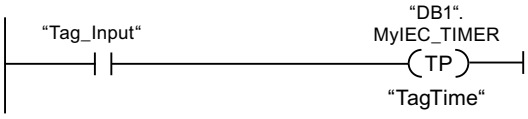

The "Start pulse timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The timer "DB1".MyIEC_TIMER is started for the time stored in the operand "TagTime".

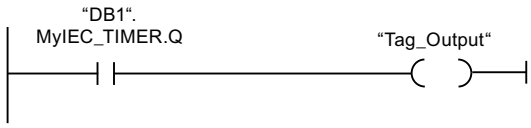

As long as the timer "DB1". MyIEC_TIMER is running, the timer status ("DB1".MyIEC_TIMER.Q) has signal state "1" and the operand "Tag_Output" is set. When the IEC timer has expired, the signal state of the time status changes back to "0" and the "Tag_Output" operand is reset.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(TON)---: Start on-delay timer

Description

Use the "Start on-delay timer" instruction to start an IEC timer with a specified duration as on-delay. The IEC timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC timer runs for the specified time. The output returns the signal state "1" if the RLO at the input of the instruction has the signal state "1". If the RLO changes to "0" before the time expires, the IEC timer is reset. In this case, querying the timer status for "1" returns signal state "0". The IEC timer restarts when the next positive signal edge is detected at the input of the instruction.

Specify the duration of the ON delay under the instruction at <Operand1> (duration) and specify the IEC time to be started above the instruction at <Operand2> (IEC time).

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start on-delay timer" stores its data in a structure of the data type IEC_TIMER or TON_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start on-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- The IEC_Timer structure is updated when the instruction is called. The time value at the ET output is only updated if the ET or Q outputs are scanned (e.g. "MyTimer".Q or "MyTimer".ET).
- When the assigned timer is accessed.

The current timer status is stored in the ET structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0".

The execution of the "Start on-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

The following table shows the parameters of the "Start on-delay timer" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TON_TIME	IEC_TIMER, IEC_LTIMER, TON_TIME, TON_LTIME	D, L	IEC timer that is started.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction:

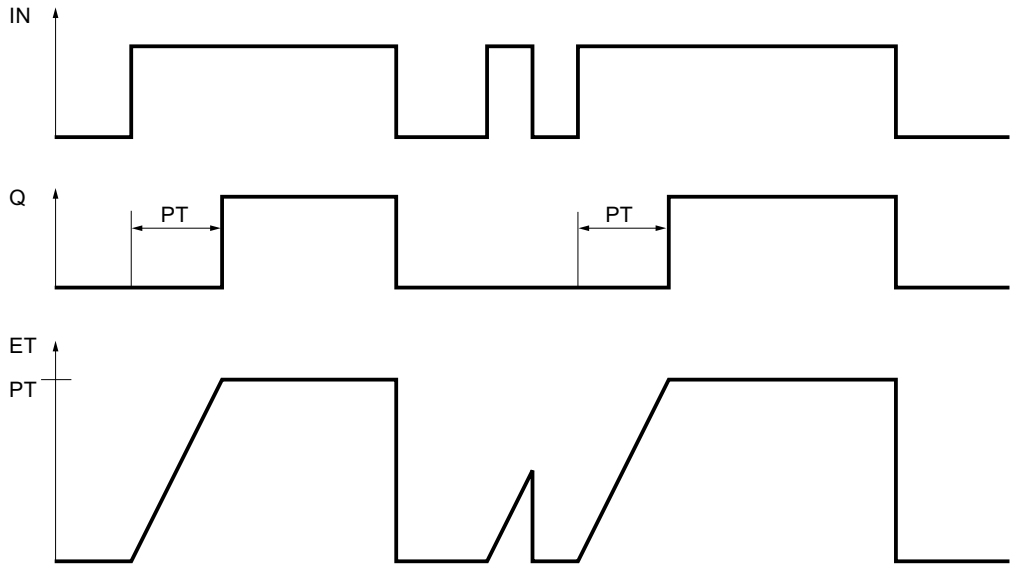

Example

The following example shows how the instruction works:

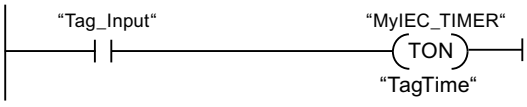

The "Start on-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The "MyIEC_TIMER timer is started for the time stored in the "TagTime" operand.

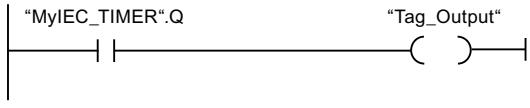

If the timer "MyIEC_TIMER" has expired and the operand "Tag_Input" has the signal state "1", querying the timer status ("MyIEC_TIMER).Q) returns signal state "1" and the "Tag_Output" operand is set. When the signal state of the operand "Tag_Input" changes to "0", the querying of the timer status returns the signal state "0" and the operand "Tag_Output" is reset.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(TOF)---: Start off-delay timer

Description

Use the "Start off-delay timer" instruction to start an IEC timer with a specified duration as on-delay. The query of the timer status for "1" returns the signal state "0" if the result of logic operation (RLO) at the input of the instruction has the signal state "1". When the RLO changes from "1" to "0" (negative signal edge), the IEC timer starts with the specified time. The timer status remains at signal state "1" as long as the IEC timer is running. When the timer has run out and the RLO at the input of the instruction has the signal state "0", the timer status is set to the signal state "0". If the RLO changes to "1" before the time expires, the IEC timer is reset and the timer status remains at signal state "1".

Specify the duration of the OFF delay under the instruction at <Operand1> (duration) and specify the IEC time to be started above the instruction at <Operand2> (IEC time).

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start off-delay timer" stores its data in a structure of the data type IEC_TIMER or TOF_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start off-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- The IEC_Timer structure is updated when the instruction is called. The time value at the ET output is only updated if the ET or Q outputs are scanned (e.g. "MyTimer".Q or "MyTimer".ET).
- When the assigned timer is accessed.

The current timer status is stored in the ET structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0".

The execution of the "Start off-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

The following table shows the parameters of the "Start off-delay timer" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TOF_TIME	IEC_TIMER, IEC_LTIMER, TOF_TIME, TOF_LTIME	D, L	IEC timer that is started.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction:

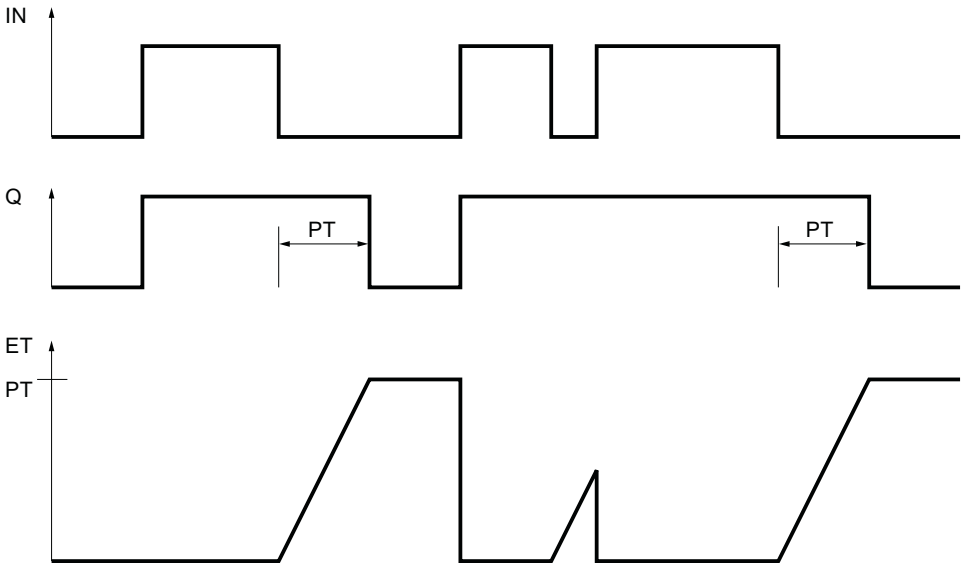

Example

As long as timer #MyIEC_TIMER is running, the query of the time status (#MyIEC_TIMER.Q) returns the signal state "1" and operand "Tag_Output" is set. If the timer has expired and the operand "Tag_Input" has the signal state "0", the query of the timer status returns the signal state "0". If the signal state of the operand "Tag_Input" changes to "1" before timer #MyIEC_TIMER expires, the timer is reset. When the signal state of the operand "Tag_Input" is "1", the query of the timer status returns the signal state "1".

The following example shows how the instruction works:

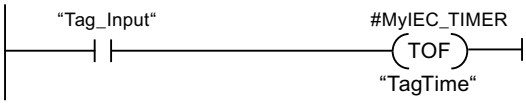

The "Start off-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "1" to "0". The timer #MyIEC_TIMER is started for the time stored in the operand "TagTime".

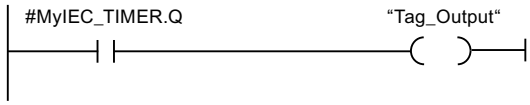

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(TONR)---: Time accumulator

Description

You can use the "Time accumulator" instruction to record how long the signal is at the input of instruction "1". The instruction is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The time is recorded as long at the RLO is "1". If the RLO changes to "0", the instruction is halted. If the RLO changes back to "1", the time recording is continued. The query of the timer status for "1" returns the signal state "1" if the recorded time exceeds the value of the specified duration and the RLO at the input of coil is "1".

The timer status and the currently expired timer can be reset to "0" using the "Reset timer" instruction.

Specify the duration under the instruction at <Operand1> (duration) and specify the IEC time to be started above the instruction at <Operand2> (IEC time).

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed only at the end of the network.

For S7-1200 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER or TONR_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- The IEC_Timer structure is updated when the instruction is called. The time value at the ET output is only updated if the ET or Q outputs are scanned (e.g. "MyTimer".Q or "MyTimer".ET).
- When the assigned timer is accessed.

The current timer status is stored in the ET structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0".

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TONR_TIME	IEC_TIMER, IEC_LTIMER, TONR_TIME, TONR_LTIME	D, L	IEC timer that is started.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction:

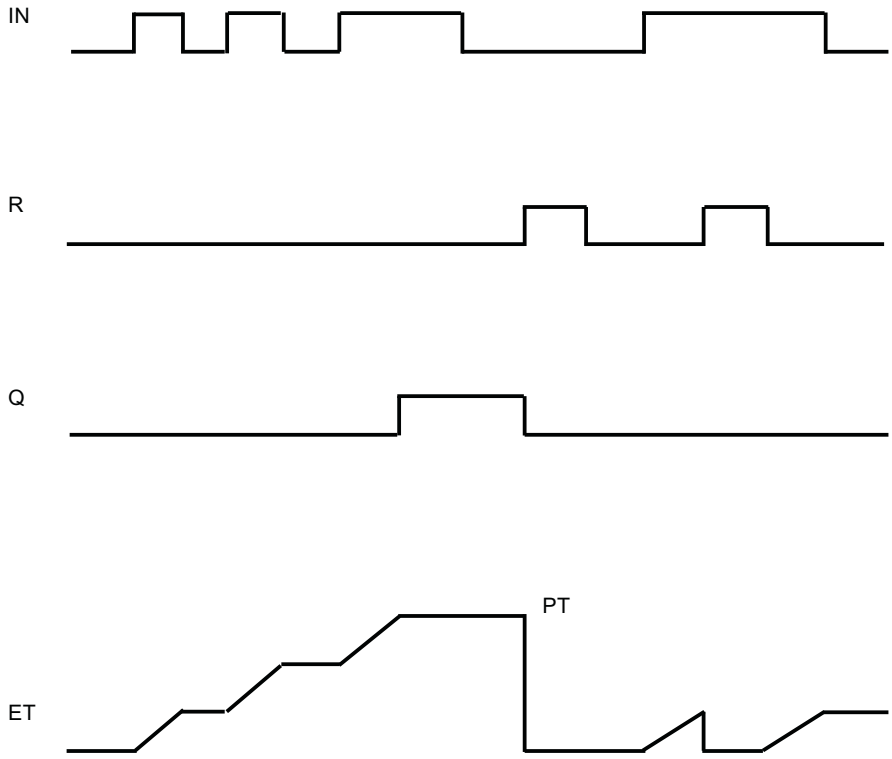

Example

The following example shows how the instruction works:

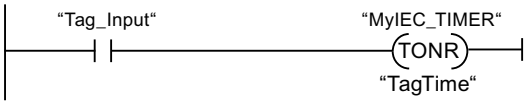

The "Time accumulator" instruction executes on a positive signal edge in the RLO. The time is recorded as long as the operand "Tag_Input" has the signal state "1".

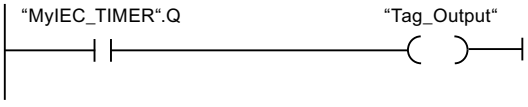

If the recorded time exceeds the value of the operand "TagTime", then the query of the timer status ("MyIEC_TIMER".Q) will return the signal state "1" and the operand "Tag_Output" will be set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(RT)---: Reset timer

Description

You can use the "Reset timer" instruction to reset an IEC timer to "0". The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If current is flowing to the coil (RLO is "1"), the structure components of the timer in the specified data block are reset to "0". If the RLO at the input of the instruction is "0", the timer remains unchanged.

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

You assign the "Reset timer" instruction an IEC timer that has been declared in the program.

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

Parameters

The following table shows the parameters of the "Reset timer" instruction":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer that is reset.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

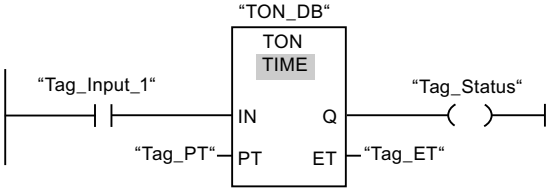

The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The timer stored in the "TON_DB" instance data block starts running for the time duration specified by operand "Tag_PT".

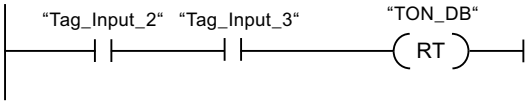

If operands "Tag_Input_2" and "Tag_Input_3" have the signal state "1", the "Reset timer" instruction is executed and the timer stored in the "TON_DB" data block.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(PT)---: Load time duration

Description

You can use the "Load time duration" instruction to set the time for an IEC timer. The instruction is executed in every cycle when the result of logic operation (RLO) at the input of the instruction has the signal state "1". The instruction writes the specified time to the structure of the specified IEC timer.

Specify the duration that is loaded under the instruction at <Operand1> (duration) and specify the IEC time to be started above the instruction at <Operand2> (IEC time).

Note

If the specified IEC timer is running while the instruction executes, the instruction overwrites the current time of the specified IEC timer. This can change the timer status of the IEC timer.

You assign an IEC timer declared in the program to the "Load time duration" instruction.

The instruction data is updated only when the instruction is called and each time the assigned IEC timer is accessed. The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Parameter

The following table shows the parameters of the "Load time duration" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer, the duration of which is set.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

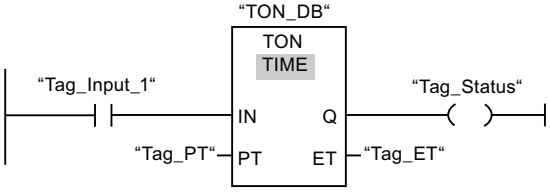

The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".

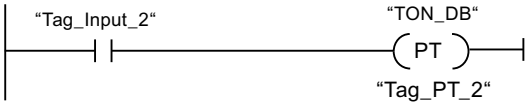

The "Load time duration" instruction is executed when the operand "Tag_Input_2" has the signal state "1". The instruction writes the time duration "Tag_PT_2" in the instance data block "TON_DB" and at the same time overwrites the value of the operand "Tag_PT" within the data block. The signal state of the timer status may therefore change at the next query or when "MyTimer".Q or "MyTimer".ET are accessed.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

Legacy (S7-1500)

S_PULSE: Assign pulse timer parameters and start (S7-1500)

Description

The "Assign pulse timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as soon as the signal state at input S is "1". If the signal state at input S changes to "0" before the programmed duration expires, the timer is stopped. In this case, the signal state at output Q is "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign pulse timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

For an example of how a time cell can be structured, refer to See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction "Assign pulse timer parameters and start":

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L or constant	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction "Assign pulse timer parameters and start":

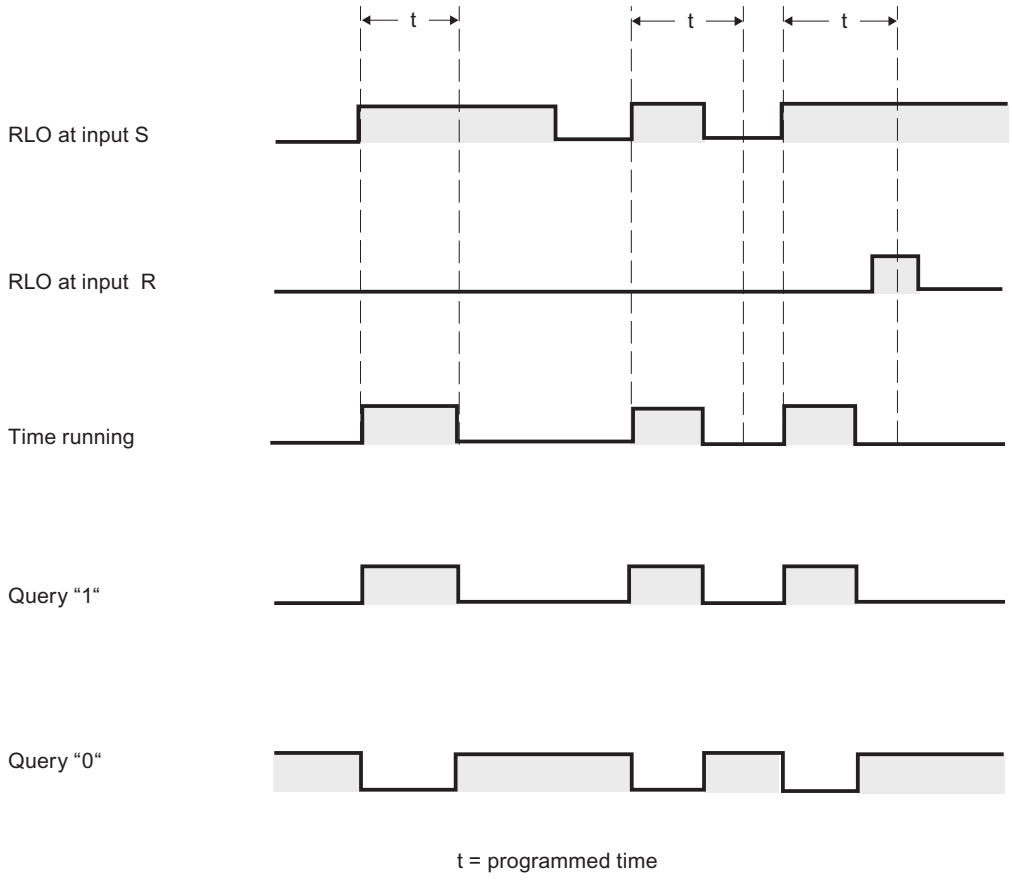

Example

The following example shows how the instruction works:

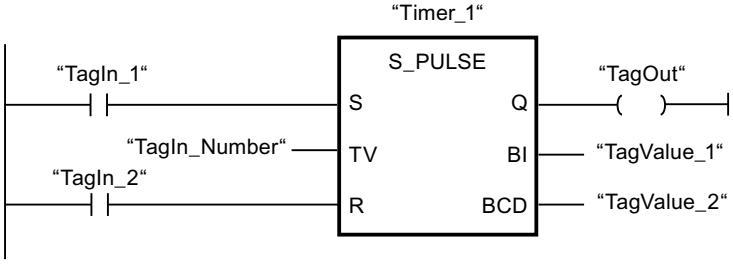

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" as long as the operand "TagIn_1" has the signal state "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer "Timer_1" is stopped. The operand "TagOut" is reset to "0" in this case.

The operand "TagOut" has the signal state "1" as long as the timer is running and the operand "TagIn_1" has the signal state "1". When the timer expires or is reset, the operand "TagOut" is reset to "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on LAD (Page 7121)

S_PEXT: Assign extended pulse timer parameters and start (S7-1500)

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". As long as the timer is running, the output Q has the signal state "1". When the timer expires, the output Q is reset to "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input TV.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign extended pulse timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

For an example of how a time cell can be structured, refer to See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction "Assign extended pulse timer parameters and start":

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L or constant	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:

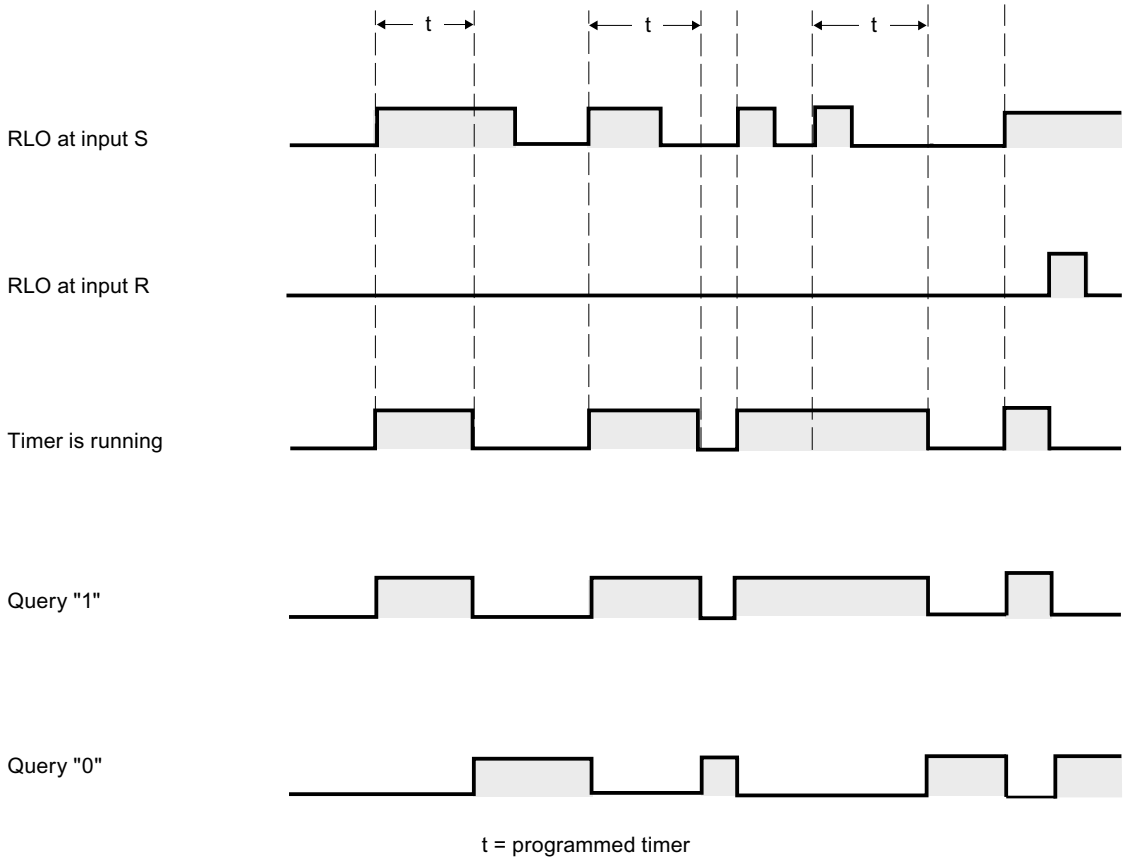

Example

The following example shows how the instruction works:

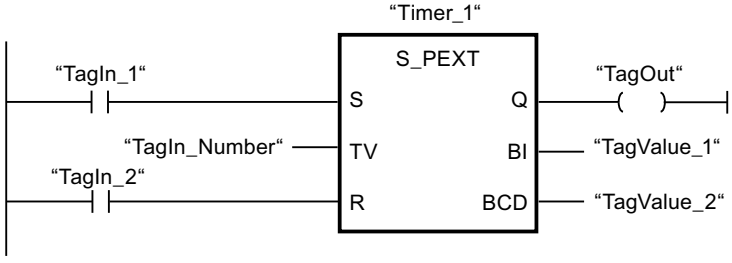

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative edge at input S. If the signal state of the operand "TagIn_1" changes from "0" to "1" before the timer expires, the timer is restarted.

The operand "TagOut" has the signal state "1" as long as the timer is running. When the timer expires or is reset, the operand "TagOut" is reset to "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on LAD (Page 7121)

S_ODT: Assign on-delay timer parameters and start (S7-1500)

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as soon as the signal state at input S is "1". If the timer expires correctly and input S still has signal state "1" then output Q returns signal state "1". If the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the time is running and the signal state at input R changes from "0" to "1" then the current time value and the time base are also set to zero. In this case, the signal state at output Q is "0". The timer is reset if the signal state is "1" at the R input even if the timer is not running and the RLO at input S is "1".

Specify the timer of the instruction in the placeholder above the box. The timer must be declared with the data type TIMER.

The "Assign on-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

For an example of how a time cell can be structured, refer to See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign on-delay timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L or constant	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:

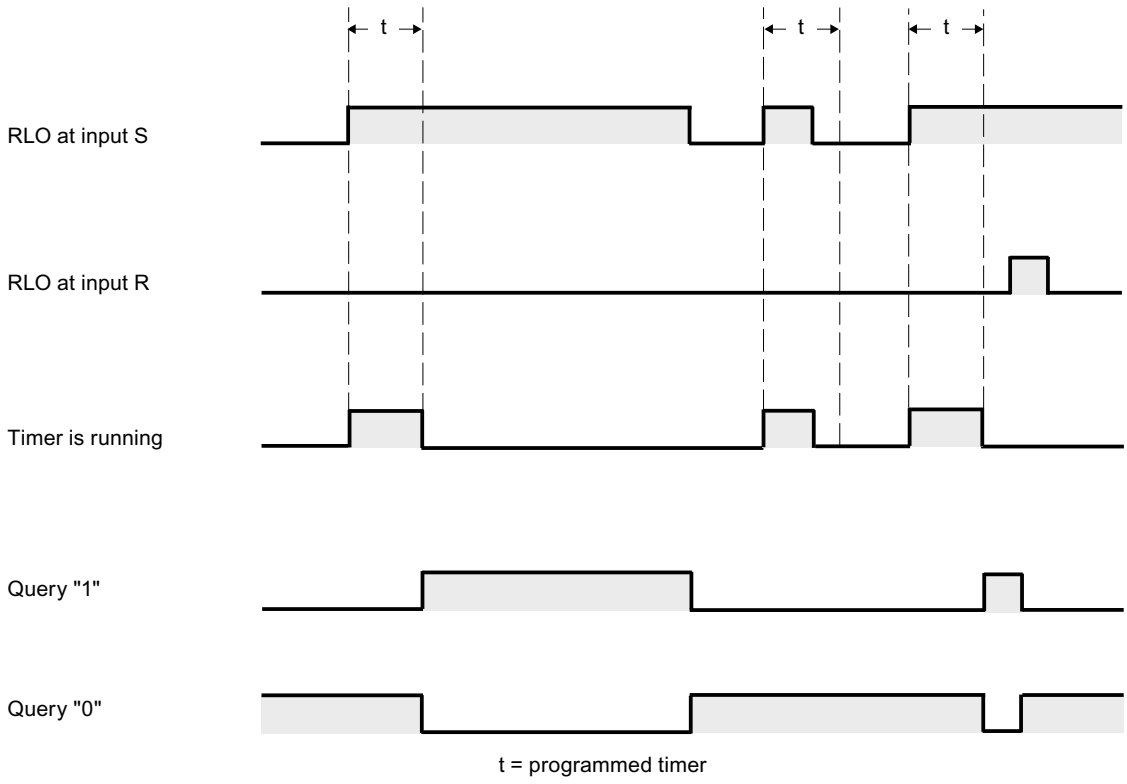

Example

The following example shows how the instruction works:

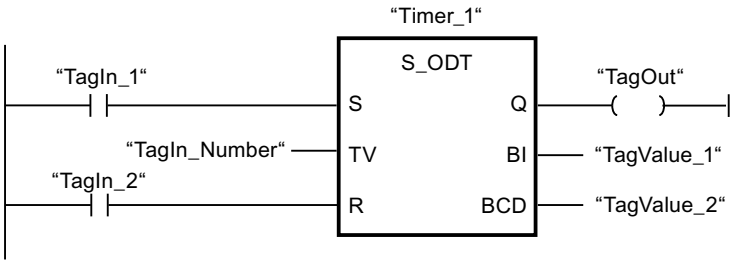

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". If the timer expires and the operand has the signal state "1", the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. The operand "TagOut" has the signal state "0" in this case.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on LAD (Page 7121)

S_ODTS: Assign retentive on-delay timer parameters and start (S7-1500)

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". If the timer expires, the "Q" output returns signal state "1" regardless of the signal state at input "S". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input (TV).

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0" regardless of the signal state at start input S. In this case, the signal state at output Q is "0".

The "Assign retentive on-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

For an example of how a time cell can be structured, refer to See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign retentive on-delay timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L or constant	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:

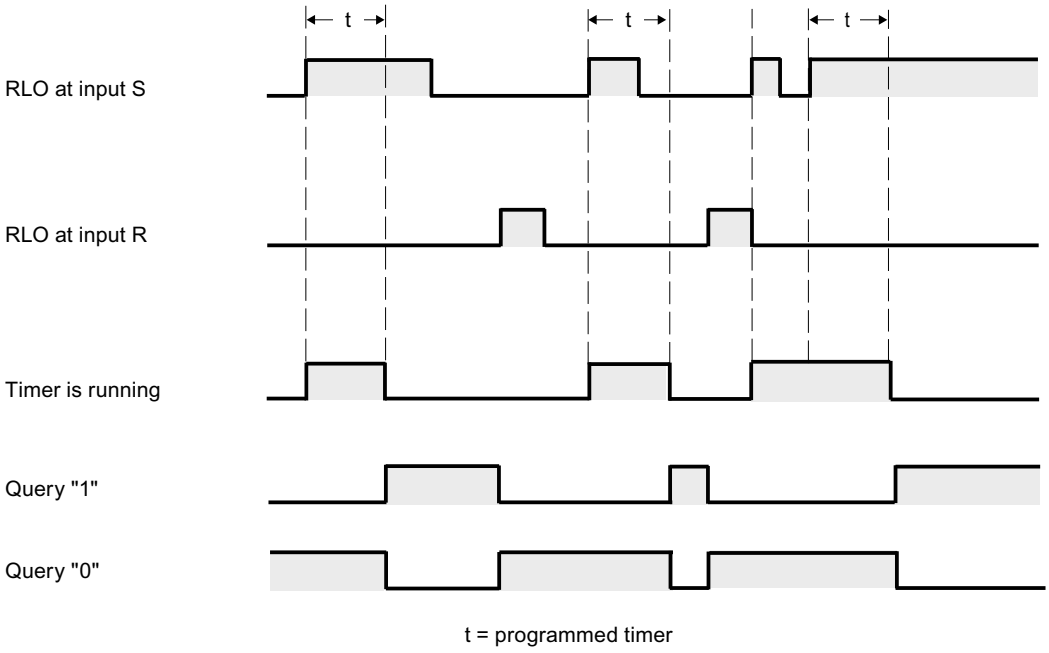

Example

The following example shows how the instruction works:

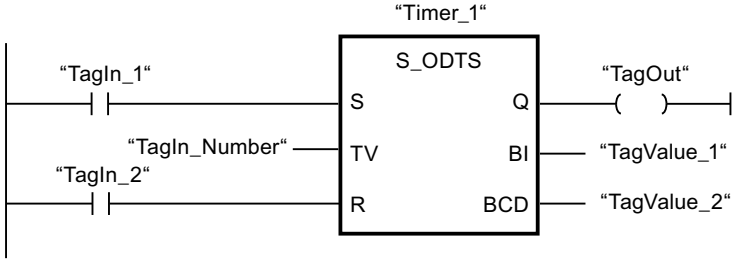

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number", even if the signal state of the operand "TagIn_1" changes to "0". When the timer expires, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is restarted.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on LAD (Page 7121)

S_OFFDT: Assign off-delay timer parameters and start (S7-1500)

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV). As long as the timer is running or input S returns signal state "1", then output Q has signal state "1". When the timer expires and the signal state at input S is "0", output Q is reset to the signal state "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at input S.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0". In this case, the signal state at output Q is "0".

The "Assign off-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

For an example of how a time cell can be structured, refer to See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign off-delay timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L or constant	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:

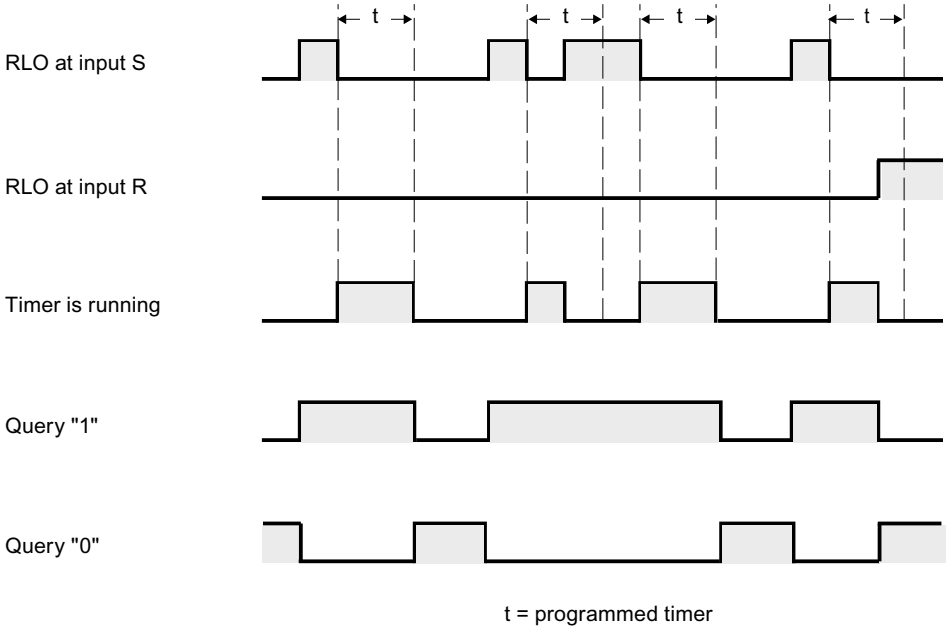

Example

The following example shows how the instruction works:

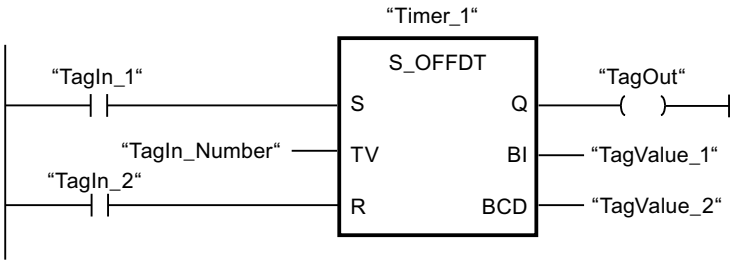

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "1" to "0". The timer expires with the time value of the operand "TagIn_Number". The operand "TagOut" is set to "1" when the timer is running and when the operand "TagIn_1" has the signal state "0". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is reset.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on LAD (Page 7121)

---(SP): Start pulse timer (S7-1500)

Description

The "Start pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer runs with the specified duration as long as the RLO has the signal state "1". As long as the timer is running, the query of timer status "1" returns the signal state "1". If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer stops. In this case, the querying of the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down to zero. The time base indicates the time period with which the time value is changed.

Specify the duration with which the time will run down at the operand placeholder <Operand1> (duration) and specify the time to be started above the instruction in <Operand2>.

The "Start pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

For an example of how a time cell can be structured, refer to See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction "Start pulse timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Operand2>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

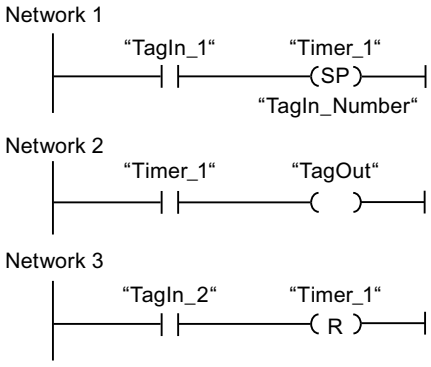

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand as long as the signal state of the "TagIn_1" operand is "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. As long as the timer is running, the "TagOut" operand returns signal state "1". A signal state change of the operand "TagIn_2" from "0" to "1" resets the timer, which stops the timer and sets the current time value to "0".

The figure below shows the timer diagram of the example:

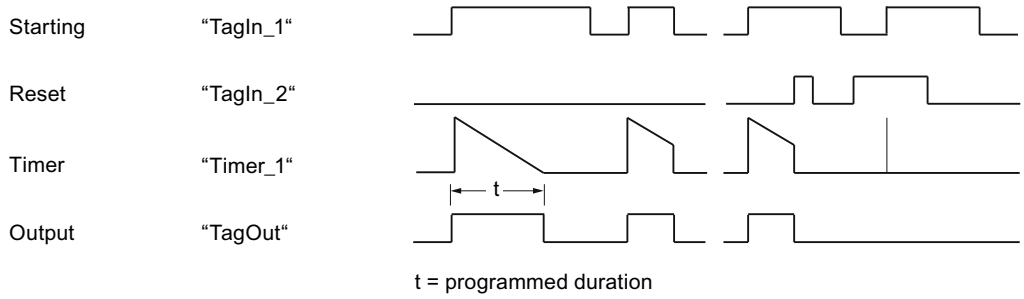

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on LAD (Page 7121)

---(SE): Start extended pulse timer (S7-1500)

Description

The "Start extended pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer runs for the specified time period even when the RLO changes to signal state "0". As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is restarted with the programmed time period. The querying of the timer status for "1" returns the signal state "0" when the timer expires.

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down to zero. The time base indicates the time period with which the time value is changed.

Specify the duration with which the time will run down at the operand placeholder <Operand1> (duration) and specify the time to be started above the instruction in <Operand2>.

The "Start extended pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 1506)

Parameters

The following table shows the parameters of the "Start extended pulse timer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Operand2>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

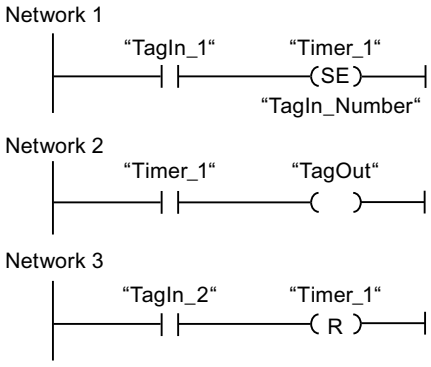

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative signal edge in the RLO. As long as the timer is running, the "TagOut" operand returns signal state "1". If the signal state of the operand "TagIn_1" changes again from "0" to "1" before the timer expires, the timer is restarted.

The figure below shows the timer diagram of the example:

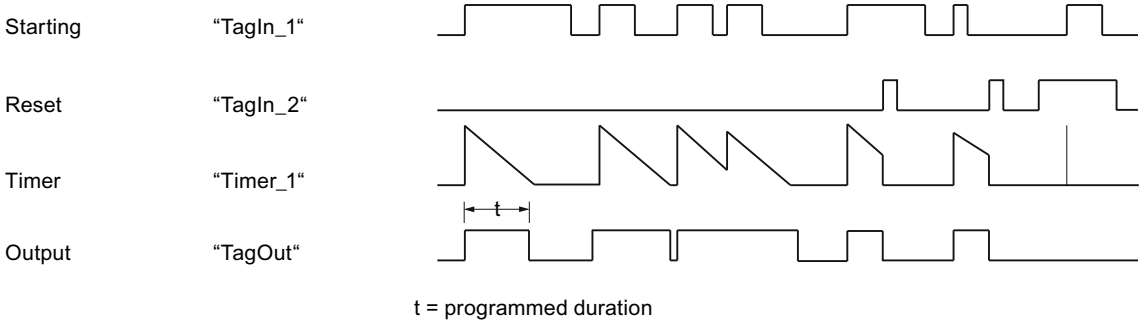

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(SD): Start on-delay timer (S7-1500)

Description

The "Start on-delay timer" instruction starts a programmed timer when the signal state "1" is detected at the start input. As long as this signal state remains "1", the timer expires with the specified duration. When the timer expires and the signal state at the start input is still "1", the query for timer status also returns "1". If the signal at the start input is "0", the timer is reset. In this case, querying the timer status returns the signal state "0". As soon as the signal at the start input changes to "1" again, the timer starts running again.

The signal state of the timer output is identical to the signal state of the start input. The start input is directly interconnected to the output and is not linked to the timer.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero.

Specify the duration with which the time will run down at the operand placeholder <Operand1> (duration) and specify the time to be started above the instruction in <Operand2>.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 1506)

Parameters

The following table shows the parameters of the "Start on-delay timer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Operand2>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

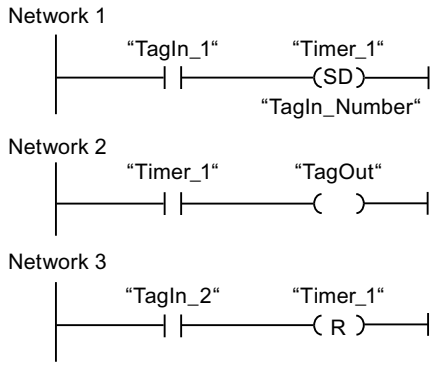

Network 1:

"#Timer_1" starts when the signal state of the "#TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "#TagIn_Number". If the signal state of the "#TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is reset.

Network 2:

When the timer has expired, the #TagIn_1 operand at the start input has signal state "1" and the timer was not reset, the #TagOut operand is "1".

Network 3:

When the signal state of the #TagIn_2 operand has signal state "1", the timer #Timer_1 and the output #TagOut are reset.

If you want to restart #Timer_1, the signal state at the #TagIn_2 operand must be "0" and the signal state at the start input #TagIn_1 must change from "0" to "1".

The figure below shows the timer diagram of the example:

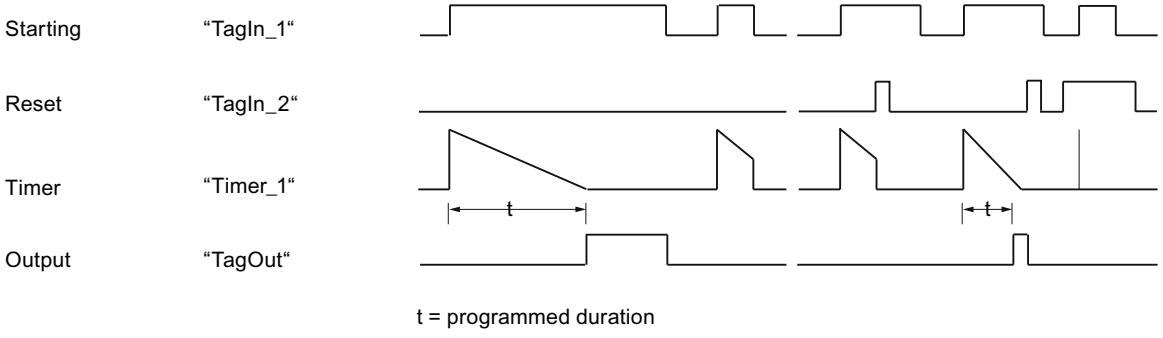

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(SS): Start retentive on-delay timer (S7-1500)

Description

The "Start retentive on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration, even if the RLO changes to the signal state "0". When the timer expires, the querying of the timer status for "1" returns the signal state "1". After expiry of the timer, the timer can only be restarted if it is explicitly reset.

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

Specify the duration with which the time will run down at the operand placeholder <Operand1> (duration) and specify the time to be started above the instruction in <Operand2>.

The "Start retentive on-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 1506)

Parameters

The following table shows the parameters of the "Start retentive on-delay timer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Operand2>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

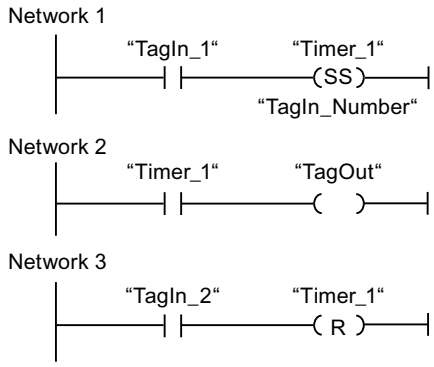

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". When the timer expires, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is restarted. If the signal state of the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

The figure below shows the timer diagram of the example:

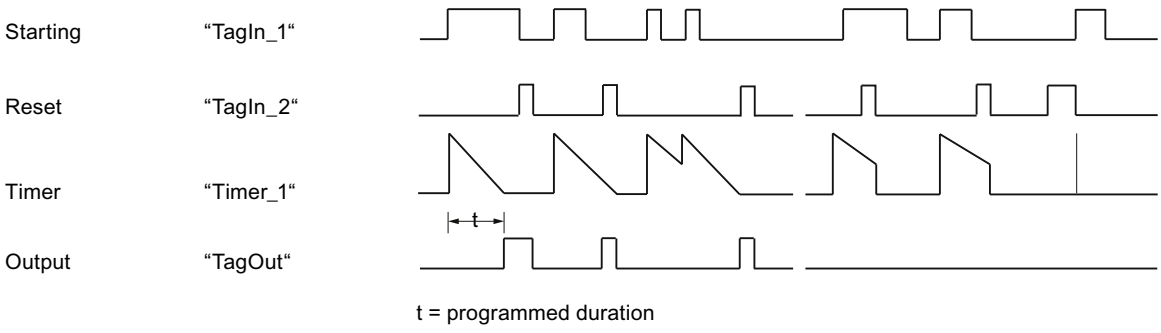

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(SF): Start off-delay timer (S7-1500)

Description

The "Start off-delay timer" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration. As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is reset. The timer is always restarted when the RLO changes from "1" to "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The queries of the timer status for "1" return the query result "1" if the signal state of the result of logic operation is "1" at the execution of the instruction. If the RLO is "0", queries of the timer status for "1" return the query result "0".

Specify the duration with which the time will run down at the operand placeholder <Operand1> (duration) and specify the time to be started above the instruction in <Operand2>.

The "Start off-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

For an example of how a time cell can be structured, refer to See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction "Start off-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Operand2>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

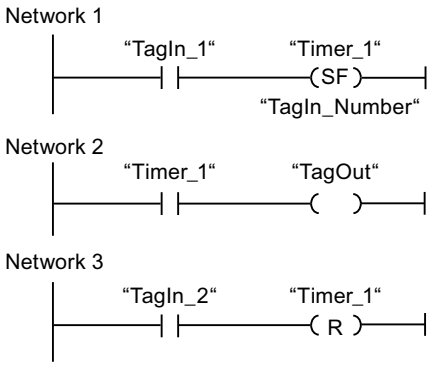

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "1" to "0". The timer expires with the time value of the operand "TagIn_Number". As long as the timer is running, the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" while the timer is running, the timer is restarted. If the signal state of the "TagIn_2" operand is "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

The figure below shows the timer diagram of the example:

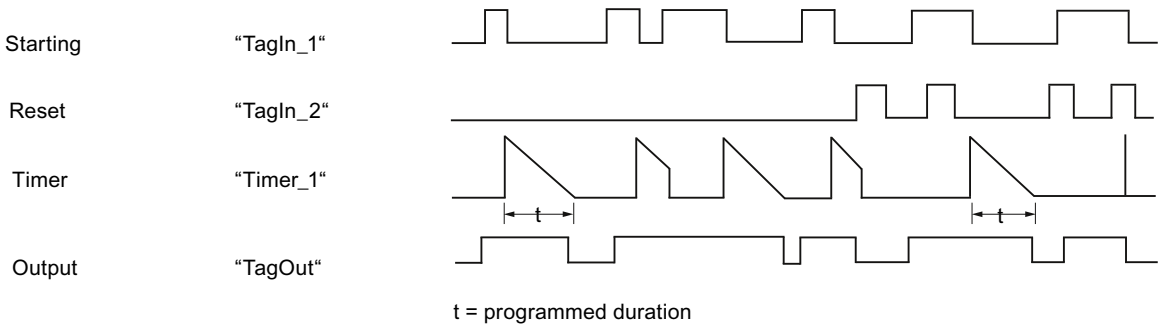

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on LAD (Page 7121)

Counter operations

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the output CV. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can query the counter status in the Q output. The signal state at the Q output is determined by the PV parameter. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value at the CV output is reset to zero when the signal state at input R changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Reset input
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

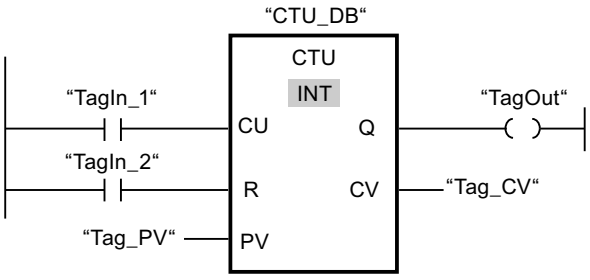

When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CV" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output has signal state "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Setting the retentivity of local tags (Page 6971)

Setting retentivity in an instance data block (Page 7037)

Basic information on LAD (Page 7121)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can query the counter status in the Q output. If the current counter value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value at the CV output is set to the value of the PV parameter when the signal state at the LD input changes to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
LD	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

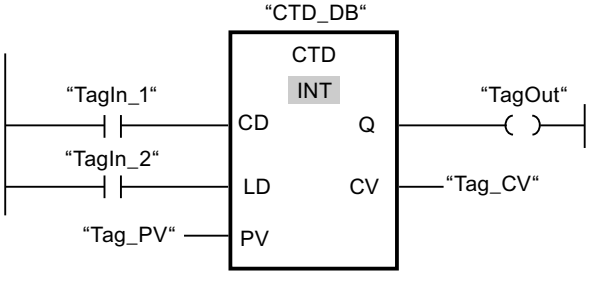

When the signal state of the "TagIn_1" operand changes from "0" to "1", the instruction is executed and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the counter value is decremented until the low limit of the specified data type (INT = -32768) is reached.

The "TagOut" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut" output has signal state "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Setting retentivity in an instance data block (Page 7037)

Setting the retentivity of local tags (Page 6971)

Basic information on LAD (Page 7121)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV output. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter. As long as the LD input has signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0".

You can query the status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value at which the QU output is set / value to which the CV output is set when LD = 1.
QU	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status of the down counter
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

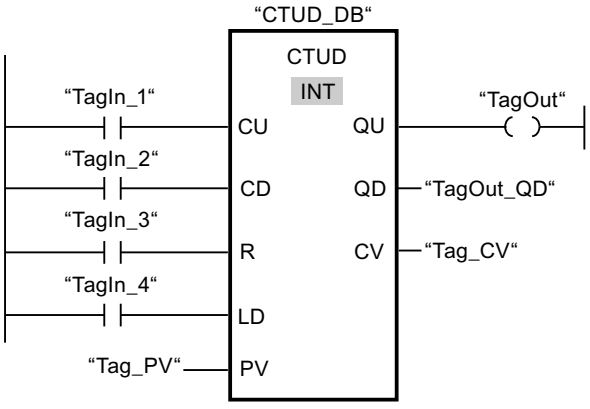

If the signal state at the "TagIn_1" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input, the current counter value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_2" input, the counter value

is decremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the CU input, the counter value is incremented until it reaches the high limit of 32767. If input CD has a positive signal edge, the counter value is decremented until it reaches the low limit of INT = -32768.

The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut" output has signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut_QD" output has signal state "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Setting retentivity in an instance data block (Page 7037)

Setting the retentivity of local tags (Page 6971)

Example of detecting the fill level of a storage area (Page 7176)

Basic information on LAD (Page 7121)

Legacy (S7-1500)

S_CU: Assign parameters and count up (S7-1500)

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CU is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L or constant	Count up input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
BI	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

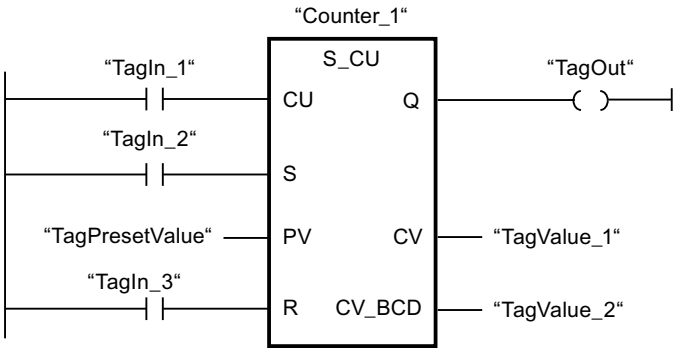

When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. When the signal state at input "TagIn_2" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

S_CD: Assign parameters and count down (S7-1500)

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value is decremented until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CD is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count down" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
BI	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

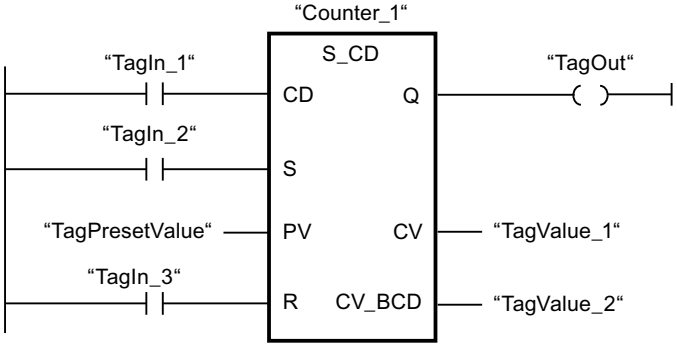

When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decremented by one. When the signal state at input "TagIn_2" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

S_CUD: Assign parameters and count up / down (S7-1500)

Description

You can use the "Assign parameters and count up / down" instruction to increment or decrement the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. If there is a positive signal edge at the CU and CD inputs in one program cycle, the counter value remains unchanged.

The counter value can be incremented until the high limit of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit "0" is reached, the counter value is not decremented any further.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO is "1" at the inputs CU and CD, the counter counts accordingly in the next scan cycle, even if no change in the signal edge was detected.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU, CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up / down" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count up / down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L, T, C or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
BI	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

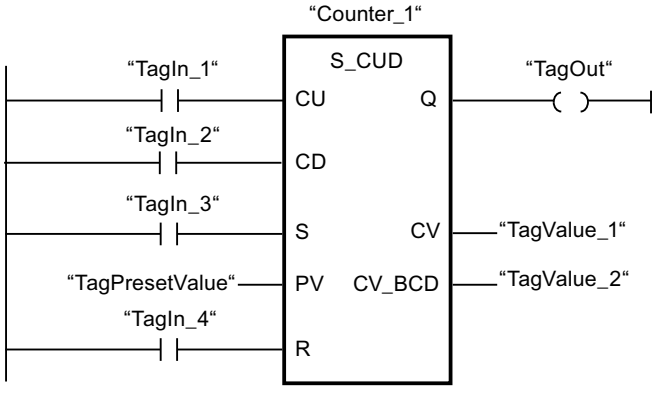

If the signal state at the "TagIn_1"" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Assign parameters and count up / down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input and the current counter value is less than "999", the counter value is incremented by one. When there is a positive signal edge at the "TagIn_2" input and the current counter value is greater than "0", the counter value is decremented by one.

When the signal state at input "TagIn_3" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_4" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(SC): Set counter value (S7-1500)

Description

You can use the "Set counter value" instruction to set the value of a counter. The instruction is executed when the result of logic operation (RLO) at the input changes from "0" to "1". When the instruction is executed, the counter is set to the specified counter value.

At the operand placeholder <Operand1> below the instruction, specify the duration with which counter is preset and specify the counter above the instruction in <Operand2>.

The "Set counter value" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table lists the parameters of the "Set counter value" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	WORD	I, Q, M, D, L or constant	Value with which the counter is preset in the BCD format. (C#0 to C#999)
<Operand2>	InOut/Input	COUNTER	C	Counter that is preset.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

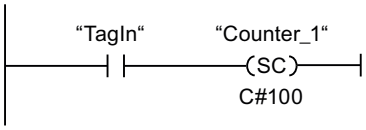

the counter "Counter_1" starts with the value "100" when the signal state of the "TagIn" operand changes from "0" to "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(CU): Count up (S7-1500)

Description

With the "Count up" instruction you can increment the value of the specified counter by one if there is a positive signal edge in the result of logic operation (RLO). The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

The "Count up" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter whose value is incremented.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

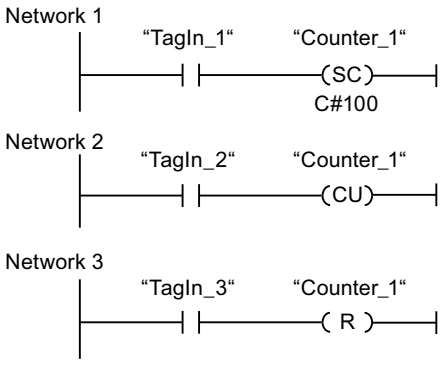

When the signal state of the "TagIn_1" operand changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of the counter "Counter_1" is incremented by one when the signal state of the operand "TagIn_2" changes from "0" to "1".

When the operand "TagIn_3" has the signal state "1", the value of the counter "Counter_1" is reset to "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(CD): Count down (S7-1500)

Description

With the "Count down" instruction you can increment the value of the specified counter by one if there is a positive signal edge in the result of logic operation (RLO). The counter value can be decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer changed on a positive signal edge.

The "Count down" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter whose value is decremented.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

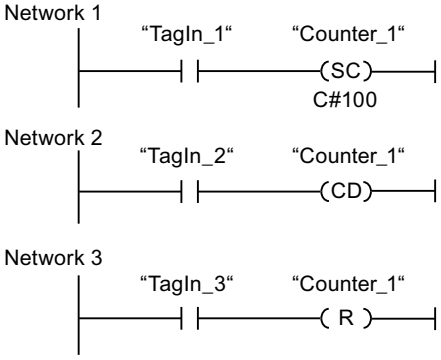

When the signal state of the "TagIn_1" operand changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of the counter "Counter_1" is decremented by one when the signal state of the operand "TagIn_2" changes from "0" to "1".

When the operand "TagIn_3" has the signal state "1", the value of the counter "Counter_1" is reset to "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to determine if a first comparison value (<Operand1>) is equal to a second comparison value (<Operand2>).

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width of the operands must be the same.

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, then instead of the instruction "CMP ==: Equal" we recommend using the instruction "IN_RANGE: Value within range".

When floating-point numbers are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check.

The special bit patterns of invalid floating-point numbers (NaN) that are the outcome of undefined results (e.g. root of -1) are not comparable. This means that if one of the operands has the value NaN, the instruction "CMP ==: Equal" has the result FALSE.

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'aa'	'aaa'	0

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP ==: Equal" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP ==: Equal" will become inactive.

Result:

For the instruction "CMP ==: Equal" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP ==: Equal" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP ==: Equal" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

Bit patterns of invalid timers, date and times, e.g. DT#2015-13-33-25:62:99.999_999_999, cannot be compared. This means that if one of the operands has an invalid value, the instruction "CMP ==: Equal" has the result FALSE.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Comparison of hardware data types

To be able to compare the operands of the PORT data type, you need to select the WORD data type from the drop-down list in the instruction box.

If you want to compare the two hardware data types HW_IO and HW_DEVICE, you must first create a tag of data type HW_ANY in the section "Temp" of the block interface and then copy the LADDR (of data type HW_DEVICE) to the tag. It is then possible to compare HW_ANY and HW_IO.

Comparison of structures

Note

Availability of comparison of structures

The option to compare structures is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

You can compare the values of two structured operands when both tags are of the same structure data type. When structures are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check. An exception is comparisons in which one of the two operands is a VARIANT or an ANY. If the data type is not yet known when the program is created, you can use the VARIANT data type. In this case you can also compare the operand with a structured tag of any data type. You can also compare two tags of the data type VARIANT or ANY with each other.

You can select the data type VARIANT for the comparison of structures from the drop-down list in the instruction box. Tags from the following data types are possible:

- PLC data type (UDT)
- STRUCT (either the structure of the STRUCT data type must be contained in a PLC data type or the two structures to be compared must be two elements of an ARRAY of STRUCT. Instance data blocks and tags of anonymous structures are not permitted.)
- Data blocks derived from a PLC data type.
- Tag to which ANY is pointing
- Tag to which VARIANT is pointing

The following requirements must be met to compare two tags of the data type ARRAY with VARIANT selected:

- The elements must each have the same data type.
- The two ARRAYs must have the same dimension.
- All dimensions must have the same number of elements. The exact ARRAY limits do not have to match.

Note

ARRAY of BOOL

When you compare two operands of the data type ARRAY of BOOL with each other, you have selected the data type VARIANT from the drop-down list, and the number of elements cannot be divided by 8, the fill bits are compared as well. This may affect the comparison result.

If an element within the structures that you want to compare is an invalid STRING/WSTRING, an invalid time or date or an invalid floating-point number, the comparison result in the RLO (result of logic operation) will return the signal state "0".

The table below shows an example of a comparison of structures:

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type A <PLC data type>		Tag value	1
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type B <PLC data type>		Tag value	0
	BOOL	FALSE		BOOL	TRUE	
	INT	2		INT	3	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	VARIANT (supplied with tag of data type A)		Tag value	1
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

Parameters

The following table shows the parameters of the instruction "Equal":

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time, ARRAY of <data type> with fixed and variable ARRAY limits, STRUCT, VARIANT, ANY, PLC data type	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time, ARRAY of <data type> with fixed and variable ARRAY limits, STRUCT, VARIANT, ANY, PLC data type	I, Q, M, D, L, P or constant	Second value to compare
As detailed above, the data types ARRAY, STRUCT (in a PLC data type), VARIANT, ANY and PLC data type (UDT) are only available as of firmware version 2.0 or 4.2.				

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

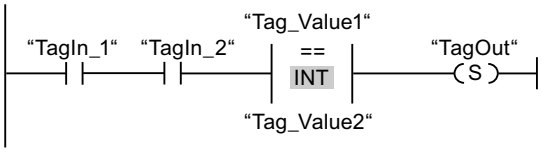

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" = "Tag_Value2".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to determine if a first comparison value (<Operand1>) is not equal to a second comparison value (<Operand2>).

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width of the operands must be the same.

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, then instead of the instruction "CMP <>: Not equal" we recommend using the instruction "OUT_RANGE: Value outside range".

If floating-point numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

The special bit patterns of invalid floating-point numbers (NaN) that are the outcome of undefined results (e.g. root of -1) are not comparable. This means that if one of the operands has the value NaN, the instruction "CMP <>: Not equal" has the result FALSE.

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0
'aa'	'aaa'	1

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP <>: Not equal" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP <>: Not equal" will become inactive.

Result:

For the instruction "CMP <>: Not equal" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP <>: Not equal" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP <>: Not equal" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

Bit patterns of invalid timers, date and times, e.g. DT#2015-13-33-25:62:99.999_999_999, cannot be compared. This means that if one of the operands has an invalid value, the instruction "CMP <>: Not equal" has the result FALSE.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Comparison of hardware data types

To be able to compare the operands of the PORT data type, you need to select the WORD data type from the drop-down list in the instruction box.

If you want to compare the two hardware data types HW_IO and HW_DEVICE, you must first create a tag of data type HW_ANY in the section "Temp" of the block interface and then copy the LADDR (of data type HW_DEVICE) to the tag. It is then possible to compare HW_ANY and HW_IO.

Comparison of structures

Note

Availability of comparison of structures

The option to compare structures is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

You can compare the values of two structured operands when both tags are of the same structure data type. When structures are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check. An exception is comparisons in which one of the two operands is a VARIANT or an ANY. If the data type is not yet known when the program is created, you can use the VARIANT data type. In this case you can also compare the operand with a structured tag of any data type. You can also compare two tags of the data type VARIANT or ANY with each other.

You can select the data type VARIANT for the comparison of structures from the drop-down list in the instruction box. The following data types are possible:

- PLC data type
- STRUCT (the structure of the data type STRUCT must be contained in a PLC data type (UDT) or the two structures to be compared are two elements of an ARRAY of STRUCT. Anonymous structures are not permitted.)
- Tag to which ANY is pointing
- Tag to which VARIANT is pointing

The following requirements must be met to compare two tags of the data type ARRAY with VARIANT selected:

- The elements must each have the same data type.
- The two ARRAYs must have the same dimension.
- All dimensions must have the same number of elements. The exact ARRAY limits do not have to match.

Note

ARRAY of BOOL

When you compare two operands of the data type ARRAY of BOOL with each other, you have selected the data type VARIANT from the drop-down list, and the number of elements cannot be divided by 8, the fill bits are compared as well. This may affect the comparison result.

If an element within the structures that you want to compare is an invalid STRING/WSTRING, an invalid time or date or an invalid floating-point number, the comparison result in the RLO (result of logic operation) will return the signal state "0".

The table below shows an example of a comparison of structures:

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type A <PLC data type>		Tag value	0
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type B <PLC data type>		Tag value	1
	BOOL	FALSE		BOOL	TRUE	
	INT	2		INT	3	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	VARIANT (supplied with tag of data type A)		Tag value	0
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

Parameters

The following table shows the parameters of the "Not equal" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time, ARRAY of <data type> with fixed and variable ARRAY limits, STRUCT, VARIANT, ANY, PLC data type	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time, ARRAY of <data type> with fixed and variable ARRAY limits, STRUCT, VARIANT, ANY, PLC data type	I, Q, M, D, L, P, or constant	Second value to compare
As detailed above, the data types ARRAY, STRUCT (in a PLC data type), VARIANT, ANY and PLC data type (UDT) are only available as of firmware version 2.0 or 4.2.				

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

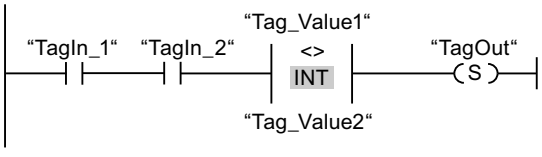

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <> "Tag_Value2".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to determine if a first comparison value (<Operand1>) is greater than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'Hello World'	'Hello World'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'AAA'	'a'	0

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >=: Greater or equal" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >=: Greater or equal" will become inactive.

Result:

For the instruction "CMP >=: Greater or equal" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >=: Greater or equal" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP >=: Greater or equal" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is greater (more recent) than or equal to the point of time at <Operand2>.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Parameters

The following table shows the parameters of the "Greater or equal" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

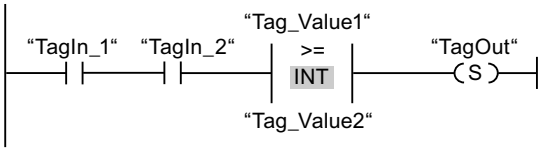

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" >= "Tag_Value2".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Example of detecting the fill level of a storage area (Page 7176)

Basic information on LAD (Page 7121)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to determine if a first comparison value (<Operand1>) is less than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'Hello World'	'Hello World'	1
'HelloWorld'	'Hello World'	0
'BB'	'AA'	0
'AAA'	'AA'	0

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP <=: Less or equal" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP <=: Less or equal" will become inactive.

Result:

For the instruction "CMP <=: Less or equal" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP <=: Less or equal" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP <=: Less or equal" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is less (less recent) than or equal to the point of time at <Operand2>.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Parameters

The following table shows the parameters of the "Less or equal" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

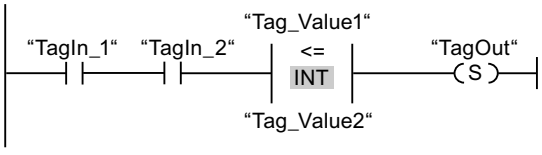

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <= "Tag_Value2".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

CMP >: Greater than

Description

You can use the "Greater than" instruction to determine if a first comparison value (<Operand1>) is greater than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'AA'	'aa'	0
'AAA'	'a'	0

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >: Greater than" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >: Greater than" will become inactive.

Result:

For the instruction "CMP >: Greater than" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >: Greater than" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP >: Greater than" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is greater (more recent) than the point of time at <Operand2>.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Parameters

The following table shows the parameters of the "Greater than" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

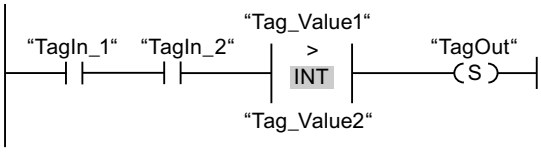

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" > "Tag_Value2".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

CMP <: Less than

Description

You can use the "Less than" instruction to determine if a first comparison value (<Operand1>) is less than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'BB'	'AA'	0
'AAA'	'AA'	0

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP <: Less than" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP <: Less than" will become inactive.

Result:

For the instruction "CMP <: Less than" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP <: Less than" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP <: Less than" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is less (less recent) than the point of time at <Operand2>.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Parameters

The following table shows the parameters of the "Less than" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

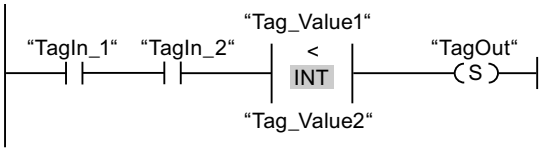

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" < "Tag_Value2".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Example of detecting the fill level of a storage area (Page 7176)

Basic information on LAD (Page 7121)

IN_RANGE: Value within range

Description

You can use the "Value within range" instruction to query whether of the value at input VAL is within a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value within range" instruction compares the value at the VAL input with the values of the MIN and MAX inputs and sends the result to the box output. If the value at input VAL satisfies the comparison MIN <= VAL or VAL <=MAX, the box output has the signal state "1". If the comparison is not fulfilled, the box output has the signal state "0".

If the box input has the signal state "0", the "Value within range" instruction is not executed.

The comparison function can only execute if the values to be compared are of the same data type and the box input is interconnected.

Parameters

The following table shows the parameters of the "Value within range" instruction:

Parameters	Declaration	Data type	Memory area	Description
Box input	Input	BOOL	I, Q, M, D, L or constant	Result of the previous logic operation
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:

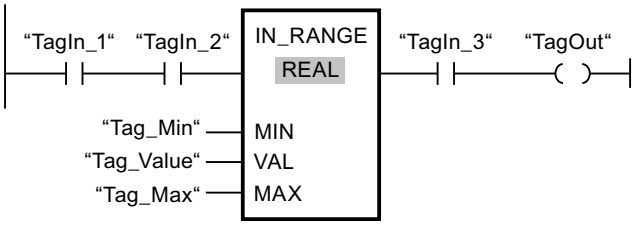

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The value of the operand "Tag_Value" is within the value range that is specified by the current values of the operands "Tag_Min" and "Tag_Max" (MIN <= VAL or VAL <= MAX).
- The operand "TagIn_3" has the signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

OUT_RANGE: Value outside range

Description

You can use the "Value outside range" instruction to query whether of the value at input VAL is outside a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value outside range" instruction compares the value at input VAL with the values of the inputs MIN and MAX and sends the result to the box output. If the value at input VAL satisfies the comparison MIN > VAL or VAL >MAX, the box output has the signal state "1". The box output also has the signal state "1" if a specified operand with the REAL data type shows an invalid value.

The box output returns the signal state "0", if the value at input VAL does not satisfy the MIN > VAL or VAL > MAX condition.

If the box input has the signal state "0", the "Value outside range" instruction is not executed.

The comparison function can only execute if the values to be compared are of the same data type and the box input is interconnected.

Parameters

The following table shows the parameters of the "Value outside range" instruction:

Parameters	Declaration	Data type	Memory area	Description
Box input	Input	BOOL	I, Q, M, D, L or constant	Result of the previous logic operation
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

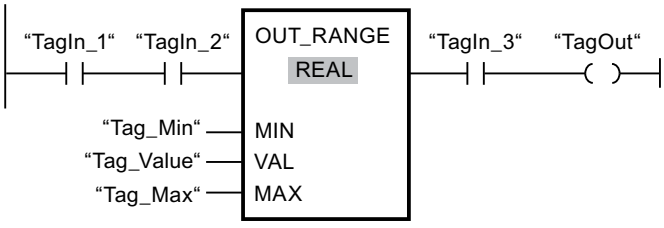

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The value of the operand "Tag_Value" is outside the value range that is specified by the values of the operands "Tag_Min" and "Tag_Max" (MIN > VAL or VAL > MAX).
- The operand "TagIn_3" has the signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

----I OK I----: Check validity

Description

You can use the "Check validity" instruction to check if the value of an operand (<operand>) is a valid floating-point number. The query is started in each program cycle when the signal state at the input of the instruction is "1".

The output of the instruction has signal state "1" when the value of the operand is a valid floating-point number at the time of the query and the input of the instruction has signal state "1". In all other cases, the signal state at the output of the "Check validity" instruction is "0".

You can use the "Check validity" instruction together with the EN mechanism. If you connect the instruction box to an EN enable input, the enable input is set only when the result of the validity query of the value is positive. You can use this function to ensure that an instruction is enabled only when the value of the specified operand is a valid floating-point number.

Parameters

The following table shows the parameters of the "Check validity" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be queried.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

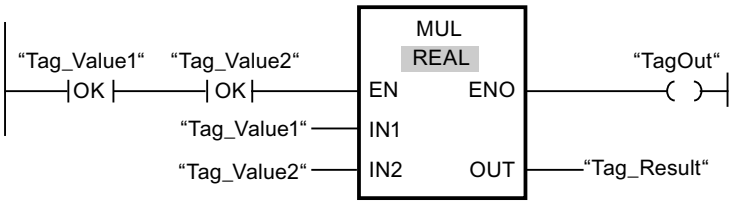

When the values of the operands "Tag_Value1" and "Tag_Value2" show valid floating-point numbers, the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The product of the multiplication is then written in the operand "Tag_Result". If the instruction is executed without errors, the TagOut operand at the "ENO" enable output is set to signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

----I NOT_OK I----: Check invalidity

Description

You can use the "Check invalidity" instruction to check if the value of an operand (<operand>) is an invalid floating-point number. The query is started in each program cycle when the signal state at the input of the instruction is "1".

The output of the instruction has signal state "1" when the value of the operand is an invalid floating-point number at the time of the query and the input of the instruction has signal state "1". In all other cases, the signal state at the output of the "Check invalidity" instruction is "0".

Parameters

The following table shows the parameters of the instruction "Check invalidity":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be queried.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

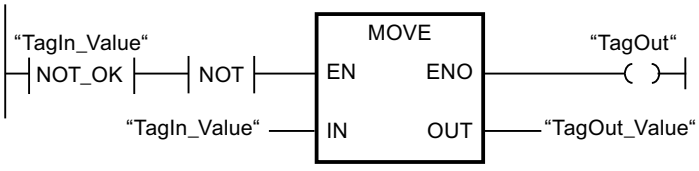

When the value of operand "TagIn_Value" is an invalid floating-point number, the instruction "Move value" is not executed. The "TagOut" operand at the ENO enable output is set to the signal state "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

VARIANT

EQ_Type: Compare data type for EQUAL with the data type of a tag

Description

You can use the "Compare data type for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You are comparing the data type of the tag (<Operand1>), which you declared in the block interface, with the data type of a tag (<Operand2>) for "Equal to".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CMP ==". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CMP ==: Equal (Page 687)

Parameters

The following table shows the parameters of the "Compare data type for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Second operand

Example

The following example shows how the instruction works:

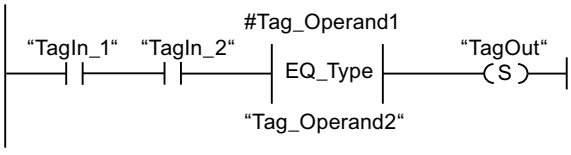

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand1 operand is equal to "Tag_Operand2".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

NE_Type: Compare data type for UNEQUAL with the data type of a tag

Description

You can use the "Compare data type for UNEQUAL with the data type of a tag" instruction to query the data type which a tag does not have to which a VARIANT points. You are comparing the data type of the tag (<Operand1>), which you declared in the block interface, with the data type of a tag (<Operand2>) for "Not equal to".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CMP <>". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CMP <>: Not equal (Page 693)

Parameters

The following table shows the parameters of the "Compare data type for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
<Operand2>	Input	Bit sequences, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Second operand

Example

The following example shows how the instruction works:

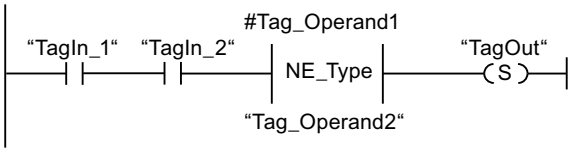

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand1 operand is not equal to "Tag_Operand2".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

EQ_ElemType: Compare data type of an ARRAY element for EQUAL with the data type of a tag

Description

You can use the "Compare data type of an ARRAY element for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You are comparing the data type of the tag (<Operand1>), which you declared in the block interface, with the data type of a tag (<Operand2>) for "Equal to".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag (<Operand1>) is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CMP ==". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CMP ==: Equal (Page 687)

Parameters

The following table shows the parameters of the "Compare data type of an ARRAY element for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Second operand

Example

The following example shows how the instruction works:

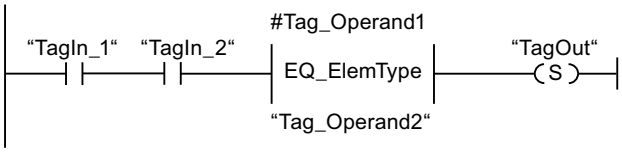

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand1 operand is equal to "Tag_Operand2".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

NE_ElemType: Compare data type of an ARRAY element for UNEQUAL with the data type of a tag

Description

You can use the "Compare data type of an ARRAY element for UNEQUAL with the data type of a tag" instruction to query which data type a tag does not have to which a VARIANT points. You are comparing the data type of the tag (<Operand1>), which you declared in the block interface, with the data type of a tag (<Operand2>) for "Not equal to".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag (<Operand1>) is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CMP <>". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CMP <>: Not equal (Page 693)

Parameters

The following table shows the parameters of the "Compare data type of an ARRAY element for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Second operand

Example

The following example shows how the instruction works:

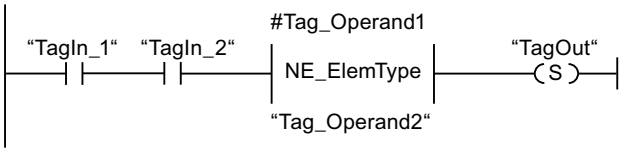

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand1 operand is not equal to "Tag_Operand2".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

IS_NULL: Check for EQUALS NULL pointer

Description

You can use the instruction "Check for EQUALS NULL pointer" to query whether the VARIANT or the reference points to a NULL pointer and therefore does not point to an object.

The <operand> must have the data type VARIANT or REF_TO <data type>.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "0" even if the ANY pointer is NULL.

Parameters

The following table shows the parameters of the "Check for EQUALS NULL pointer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT or REF_TO <data type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand that is compared for EQUALS NULL

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

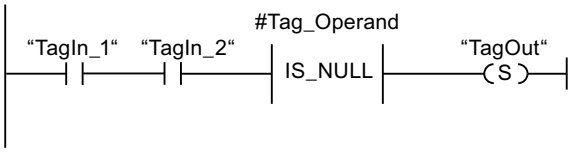

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand operand does not point to an object.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

NOT_NULL: Check for UNEQUALS NULL pointer

Description

You can use the instruction "Check for UNEQUALS NULL pointer" to query whether the VARIANT or the reference does not point to a NULL pointer and therefore points to an object.

The <operand> must have the data type VARIANT or REF_TO <data type>.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "1" even if the ANY pointer is NULL.

Parameters

The following table shows the parameters of the "Check for UNEQUALS NULL pointer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT or REF_TO <data type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand that is compared for UNEQUALS NULL

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

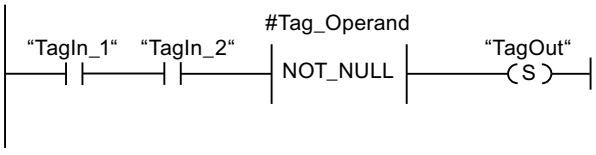

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand operand points to an object.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

IS_ARRAY: Check for ARRAY

Description

You can use the "Check for ARRAY" instruction to query whether the VARIANT points to a tag of the ARRAY data type.

<Operand> must have the VARIANT data type.

Parameters

The following table shows the parameters of the "Check for ARRAY" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand that is queried for ARRAY

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

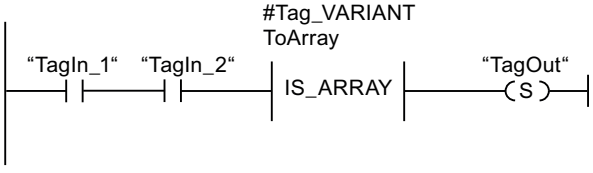

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_VARIANTToArray" operand is of the ARRAY data type.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

EQ_TypeOfDB: Compare data type of an indirectly addressed DB for EQUAL with a data type

Description

The "Compare data type of an indirectly addressed DB for EQUAL with a data type" instruction is used to query which data type of the data block has that the tag of the DB_ANY data type addresses. You compare the data type of the DB addressed by the tag <Operand1> either with the data type of another tag or directly with a data type (<Operand2>) for "Equal".

<Operand1> must have the DB_ANY data type. <Operand2>, for example, can be a PLC data type, a system data type, an axis or an FB.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	DB_ANY	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
<Operand2>	Input	TYPE_ID	I, Q, M, D, L	I, Q, M, D, L, P	Second operand

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

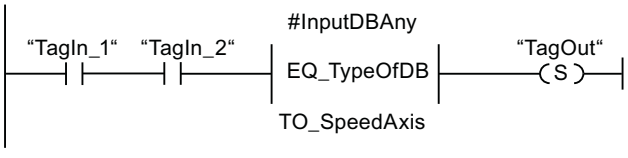

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the data type of the #InputDBAny operand addressed DB is equal with the TO_SpeedAxis data type.

The "TagOut" output is not set when the following conditions are fulfilled:

- The number of the data block is "0".
- The data block does not exist.
- The data block is an ARRAY DB.

See also

Overview of the valid data types (Page 231)

Using the DB_ANY data type (Page 204)

Basic information on LAD (Page 7121)

NE_TypeOfDB: Compare data type of an indirectly addressed DB for UNEQUAL with a data type

Description

The "Compare data type of an indirectly addressed DB for UNEQUAL with a data type" instruction is used to query which data type the data block does not have, that the tag of the DB_ANY data type addresses. You compare the data type of the DB addressed by the tag (<Operand1>) either with the data type of another tag or directly with the data type (<Operand2>) for "Not equal".

<Operand1> must have the DB_ANY data type. <Operand2>, for example, can be a PLC data type, a system data type, an axis or an FB.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	DB_ANY	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
<Operand2>	Input	TYPE_ID	I, Q, M, D, L	I, Q, M, D, L, P	Second operand

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

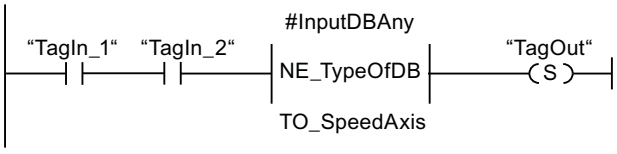

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the data type of the #InputDBAny operand addressed DB is unequal with the TO_SpeedAxis data type.

The "TagOut" output is not set when the following conditions are fulfilled:

- The number of the data block is "0".
- The data block does not exist.
- The data block is an ARRAY DB.

See also

Overview of the valid data types (Page 231)

Using the DB_ANY data type (Page 204)

Basic information on LAD (Page 7121)

Math functions

CALCULATE: Calculate

Description

The "Calculate" instruction is used to define and execute an expression for the calculation of mathematical operations or complex logic operations depending on the selected data type.

You can select the data type of the instruction from the "???" drop-down list of the instruction box. Depending on the data type selected, you can combine the functions of certain instructions to perform a complex calculation. The expression to be calculated is specified via a dialog you can open via the "Calculator" icon at the top of the instruction box. The expression can contain names of input parameters and the syntax of the instructions. Operand names and operand addresses cannot be specified.

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box.

The values of the inputs are used to execute the specified expression. Not all of the defined inputs have to be used in the expression. The result of the instruction is transferred to the output OUT.

Note

If one of the mathematical operations fails in the expression, no result is transferred to the output OUT and the enable output ENO returns the signal state "1".

If you use inputs in the expression that are not available in the box, they are inserted automatically. This requires that there are no gaps in the numbering of the inputs to be newly defined in the expression. For example, you cannot use the IN4 input in the expression unless the IN3 input has been defined.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The result of the "Calculate" instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.
- An error occurred during execution of one of the instructions in the expression.

The following table shows the instructions that can be executed together in the expression of the instruction "Calculate", depending on the selected data type:

Data type	Instruction	Syntax	Example
Bit strings	AND: AND logic operation	AND	IN1 AND IN2 OR IN3
	OR: OR logic operation	OR	
	XOR: EXCLUSIVE OR logic operation	XOR	
	INV: Create ones complement	NOT	
	SWAP: Swap [1)]	SWAP	

Data type	Instruction	Syntax	Example
Integers	ADD: Add	+	(IN1 + IN2) * IN3; (ABS(IN2)) * (ABS(IN1))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	MOD: Return remainder of division	MOD	
	INV: Create ones complement	NOT	
	NEG: Create twos complement	-(in1)	
	ABS: Form absolute value	ABS( )	
Floating-point numbers	ADD: Add	+	((SIN(IN2) * SIN(IN2) + (SIN(IN3) * SIN(IN3)) / IN3)); (SQR(SIN(IN2)) + (SQR(COS(IN3)) / IN2))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	EXPT: Exponentiate	**	
	ABS: Form absolute value	ABS( )	
	SQR: Form square	SQR( )	
	SQRT: Form square root	SQRT( )	
	LN: Form natural logarithm	LN( )	
	EXP: Form exponential value	EXP( )	
	FRAC: Return fraction	FRAC( )	
	SIN: Form sine value	SIN( )	
	COS: Form cosine value	COS( )	
	TAN: Form tangent value	TAN( )	
	ASIN: Form arcsine value	ASIN( )	
	ACOS: Form arccosine value	ACOS( )	
	ATAN: Form arctangent value	ATAN( )	
	NEG: Create twos complement	-(in1)	
	TRUNC: Truncate numerical value	TRUNC( )	
	ROUND: Round numerical value	ROUND( )	
	CEIL: Generate next higher integer from floating-point number	CEIL( )	
	FLOOR: Generate next lower integer from floating-point number	FLOOR( )	
[1] Not possible for data type BYTE.			

Parameters

The following table shows the parameters of the "Calculate" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P, or constant	First available input
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second available input
INn	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P, or constant	Additionally inserted inputs
OUT	Output	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P	Output to which the end result is to be transferred.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

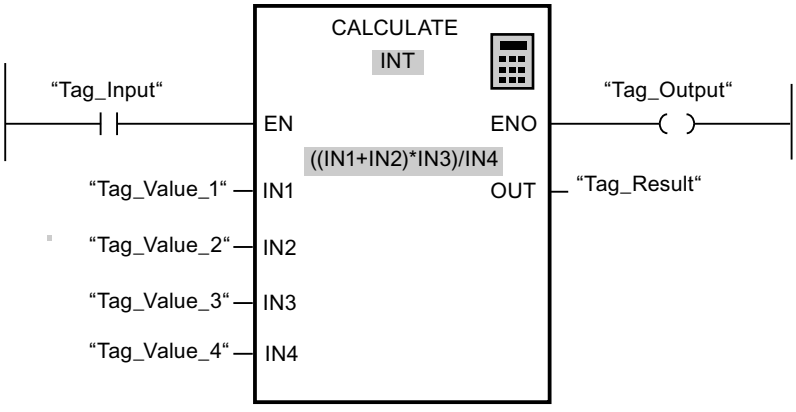

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value_1	4
IN2	Tag_Value_2	4
IN3	Tag_Value_3	3
IN4	Tag_Value_4	2
OUT	Tag_Result	12

If the "Tag_Input" input has the signal state"1", the "Calculate" instruction is executed. The value of operand "Tag_Value_1" is added to the value of operand "Tag_Value_2". The sum is multiplied with the value of operand "Tag_Value_3". The product is divided by the value of operand "Tag_Value_4". The quotient is transferred as end result to the operand "Tag_Result" at the OUT output of the instruction. If the instruction is executed without errors, the enable output ENO and operand "Tag_Output" are set to "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Using the "Calculate" instruction (Page 7147)

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Example of calculating an equation (Page 7179)

Basic information on LAD (Page 7121)

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 and the value at input IN2 and query the sum at output OUT (OUT := IN1+IN2).

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are added. The sum is stored at the OUT output.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Add" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	First number to be added
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second number to be added
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Optional input values that are added.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Sum

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:

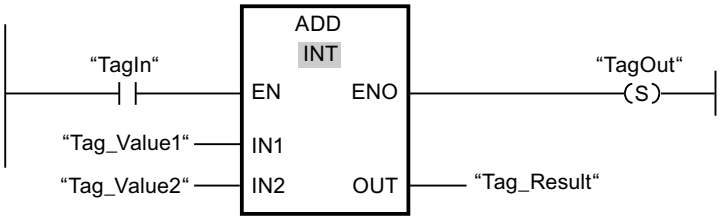

If operand "TagIn" has the signal state "1", the "Add" instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Selecting a data type (Page 7138)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at output OUT (OUT := IN1-IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Subtract" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Subtracting
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Difference

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

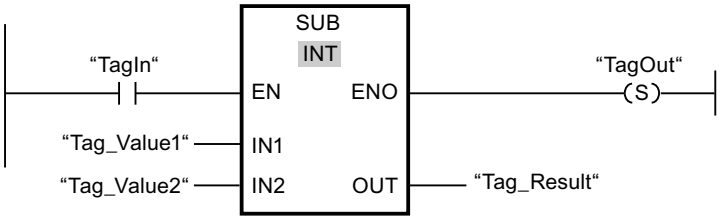

If operand "TagIn" has the signal state "1", the "Subtract" instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at input IN1 with the value at input IN2 and query the product at output OUT (OUT := IN1*IN2).

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are multiplied. The product is stored at the OUT output.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN input has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Multiply" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Multiplier
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Number being multiplied
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Optional input values that can be multiplied.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Product

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

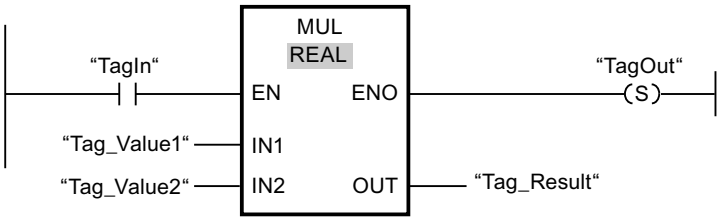

If operand "TagIn" has the signal state "1", the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The result of the multiplication is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT (OUT := IN1/IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Note

Division by the value "0"

The division of a dividend (IN1) by a divisor (IN2) with the value "0" sets the enable output (ENO) to signal state "TRUE".

On division by "0", the quotient value (OUT) is influenced by the data type as follows:

- The data type is INT or LREAL: Quotient value (OUT) is "0".
- The data type is REAL: Quotient value (OUT) is the maximum value (2143289344).

Parameters

The following table shows the parameters of the "Divide" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Quotient value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

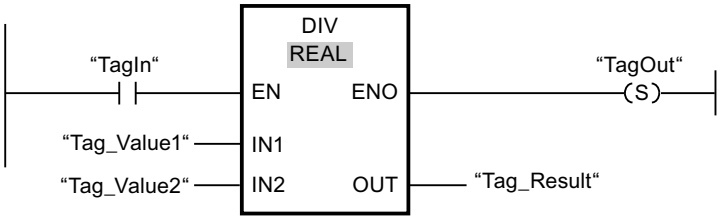

If operand "TagIn" has the signal state "1", the "Divide" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The result of the division is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at input IN1 by the value at input IN2 and query the remainder of division at output OUT.

Parameters

The following table shows the parameters of the "Return remainder of division" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers	I, Q, M, D, L, P, or constant	Dividend
IN2	Input	Integers	I, Q, M, D, L, P, or constant	Divisor
OUT	Output	Integers	I, Q, M, D, L, P	Remainder of division

You can select the data type of the instruction from the "Auto (???)" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

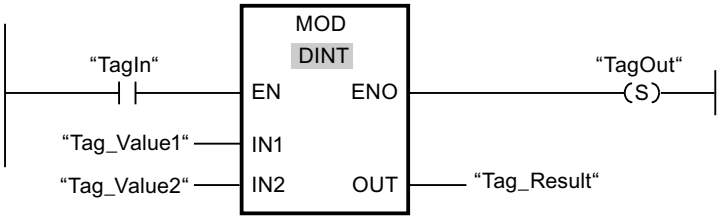

If operand "TagIn" has the signal state "1", the "Return remainder of division" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder of division is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at the IN input and query the result at the OUT output. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

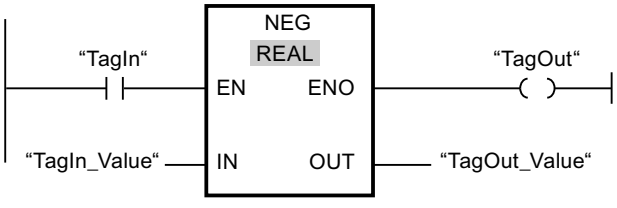

If operand "TagIn" has the signal state "1", the "Create twos complement" instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is provided at output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

INC: Increment

Description

You can use the "Increment" instruction to change the value of the operand at the IN/OUT parameter to the next higher value and query the result. The "Increment" instruction is only started when the signal state at the EN enable input is "1". If no overflow error occurs during the execution, the ENO enable output also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Increment" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	Value to be incremented.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

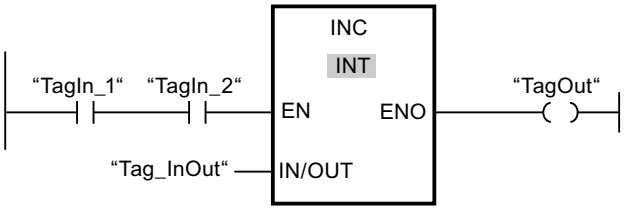

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of operand "Tag_InOut" is incremented by one and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

DEC: Decrement

Description

You can use the "Decrement" instruction to change the value of the operand at the IN/OUT parameter to the next lower value and query the result. The "Decrement" instruction is only started when the signal state at the EN enable input is "1". If the range of values of the selected data type is not exceeded during processing, the ENO output also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Decrement" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	Value to be decremented.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

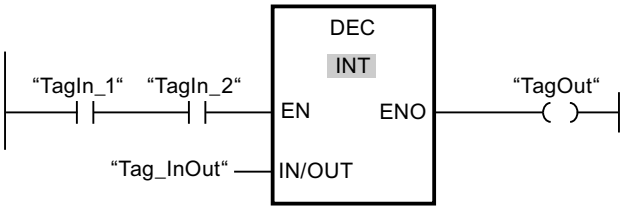

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of operand "Tag_InOut" is decremented by one and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

ABS: Form absolute value

Description

You can use the "Form absolute value" instruction to calculate the absolute value of the value specified at input IN. The result of the instruction is sent to the OUT output and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Form absolute value" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

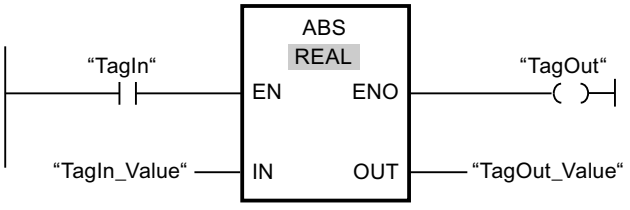

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

If operand "TagIn" has the signal state "1", the "Form absolute value" instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the available inputs and writes the lowest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

A minimum of two and a maximum of 100 inputs must be specified for the execution of the instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P, or constant	First input value
IN2	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P, or constant	Second input value
INn	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P, or constant	Additionally inserted inputs whose values are to be compared
OUT	Output	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P	Result
If the IEC check is not activated, you can also use tags with the data types TIME, LTIME, TOD, LTOD, DATE and LDT by selecting a bit string or integer with the same length as the data type of the instruction. (for example, instead of TIME => DINT, UDINT or DWORD = 32 bits).				

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

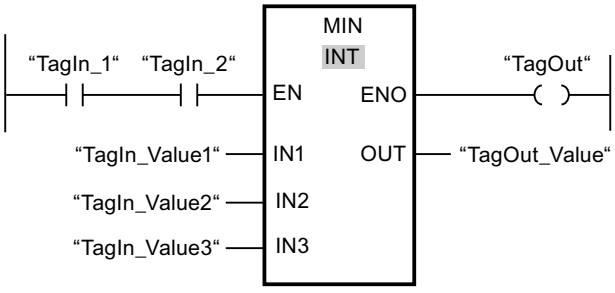

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the available inputs and writes the highest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

A minimum of two and a maximum of 100 inputs must be specified for the execution of the instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P, or constant	First input value
IN2	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P, or constant	Second input value
INn	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P, or constant	Additionally inserted inputs whose values are to be compared
OUT	Output	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P	Result
If the IEC check is not activated, you can also use tags with the data types TIME, LTIME, TOD, LTOD, DATE and LDT by selecting a bit string or integer with the same length as the data type of the instruction. (for example, instead of TIME => DINT, UDINT or DWORD = 32 bits).				

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

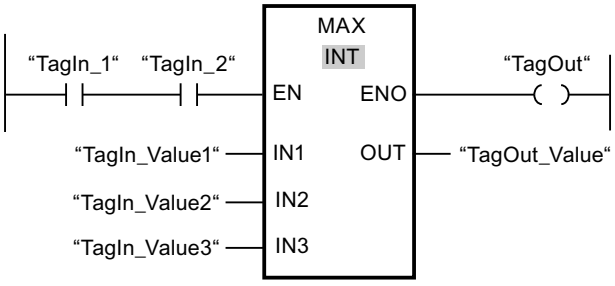

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN, the output OUT is set to the value of the MN input. If the high limit MX is exceeded, output OUT is set to the value of the input MX.

If the value at the MN input is greater than at the MX input, the result is the value specified at the IN parameter and the enable output ENO is "0".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN input is greater than the value of the MX input.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
MN	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL, DT	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DTL, DT	I, Q, M, D, L, P, or constant	Low limit
IN	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL, DT	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DTL, DT	I, Q, M, D, L, P, or constant	Input value
MX	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL, DT	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DTL, DT	I, Q, M, D, L, P, or constant	High limit
OUT	Output	Integers, floating-point numbers, TIME, TOD, DATE, DTL, DT	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DTL, DT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

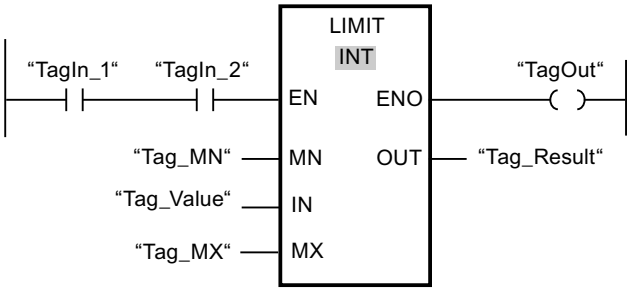

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value of operand "Tag_Value" is less than the low limit, the value of operand "Tag_MN" is copied to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

SQR: Form square

Description

The instruction "Form square" is used to square the value of a floating-point number at input IN and to write the result to output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form square":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

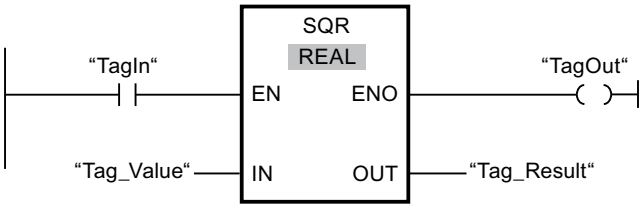

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If the operand "TagIn" has the signal state "1", the "Form square" instruction is executed. The instruction squares the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

Selecting the data type of a LAD element (Page 7138)

SQRT: Form square root

Description

The instruction "Form square root" is used form the square root from the value of the floating-point number at input IN and to write the result to output OUT. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number. If the value at input IN is "0", the result is also "0".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form square root":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square root of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

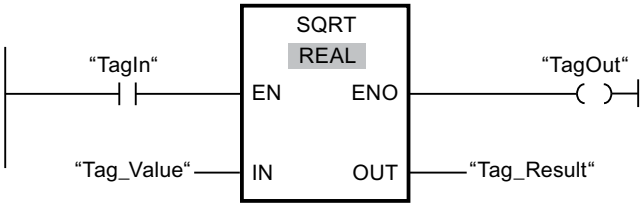

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If the operand "TagIn" has the signal state "1", the "Form square root" instruction is executed. The instruction calculates the square root of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

Selecting the data type of a LAD element (Page 7138)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) of the value at input IN. The result is sent to output OUT and can be queried there. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form natural logarithm":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

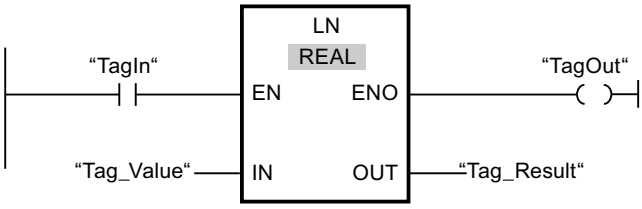

If the operand "TagIn" has the signal state "1", the "Form natural logarithm" instruction is executed. The instruction forms the natural logarithm of the value at input "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

EXP: Form exponential value

Description

You can use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value specified at input IN. The result is provided at output OUT and can be queried there (OUT = e^{IN}).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form exponential value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Exponential value of input value IN

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

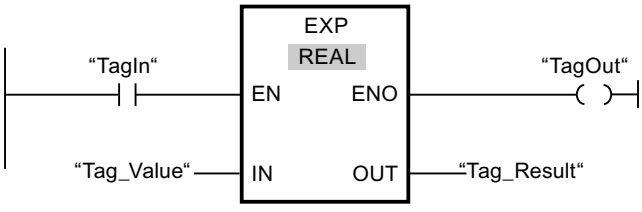

If the operand "TagIn" has the signal state "1", the "Form exponential value" instruction is executed. The instruction forms the exponent from base e and the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form sine value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Sine of the specified angle

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

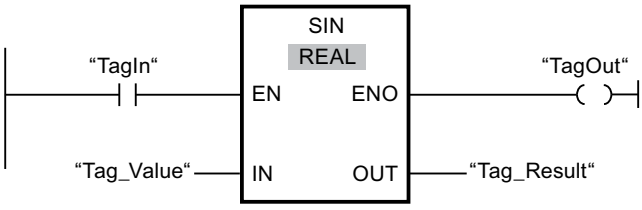

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

If operand "TagIn" has the signal state "1", the "Form sine value" instruction is executed. The instruction calculates the sine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form cosine value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Cosine of the specified angle

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

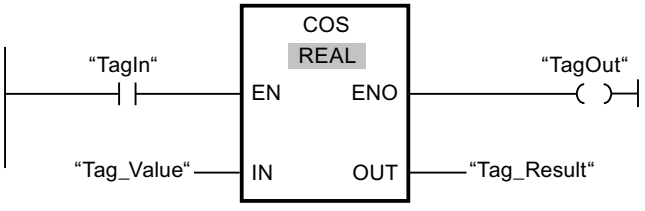

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

If operand "TagIn" has the signal state "1", the "Form cosine value" instruction is executed. The instruction calculates the cosine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the tangent of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form tangent value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Tangent of the specified angle

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

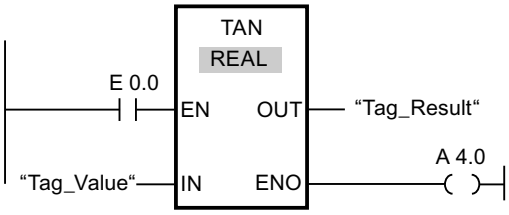

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the tangent of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from the arcsine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the "Form arcsine value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

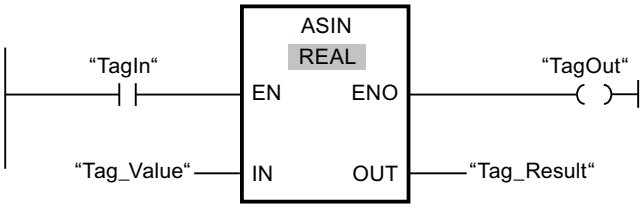

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

If operand "TagIn" has the signal state "1", the "Form arcsine value" instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from the cosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from 0 to +π.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the "Form arccosine value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

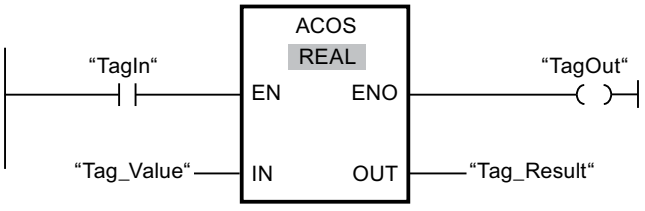

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

If operand "TagIn" has the signal state "1", the "Form arccosine value" instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from the tangent value specified at input IN, which corresponds to this value. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) at input IN. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form arctangent value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Tangent value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

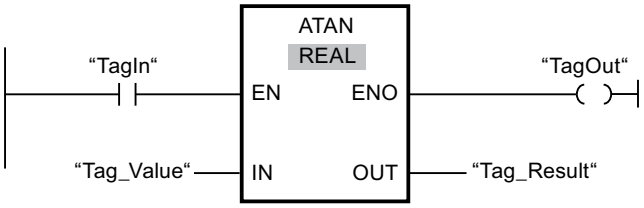

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398 (π/4)

If the operand "TagIn" has the signal state "1", the "Form arctangent value" instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at input "Tag_Value". The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Invalid floating-point numbers (Page 253)

Basic information on LAD (Page 7121)

FRAC: Return fraction

Description

You can use the "Return fraction" instruction to determine the decimal places of the value at the IN input. The result of the query is stored at the OUT output and can be queried there. If, for example, the IN input has the value 123.4567, the OUT output returns the value 0.4567.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Return fraction" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Value, whose decimal places are to be determined.
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Decimal places of the value at the IN input

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

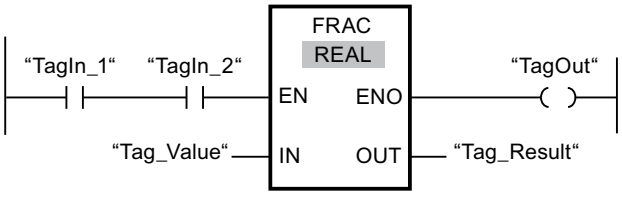

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	2.555
OUT	Tag_Result	0.555

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Return fraction" instruction is started. The decimal places from the value of operand "Tag_Value" are copied to operand "Tag_Result". If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

EXPT: Exponentiate

Description

You can use the "Exponentiate" instruction to raise the value at the IN1 input to a power specified with the value at the IN2 input. The result of the instruction is provided at the OUT output and can be queried there (OUT =$IN1^{IN2}$).

The IN1 input can only be assigned valid floating-point numbers. Integers can also be assigned to the IN2 input.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- Errors occur during execution of the instruction, for example, an overflow occurs.

Parameters

The following table shows the parameters of the "Exponentiate" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Base value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Value with which the base value is exponentiated
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

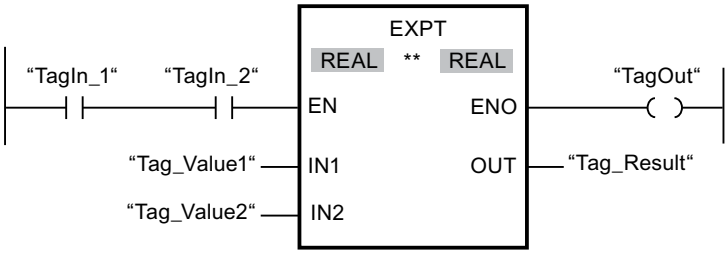

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Exponentiate" instruction is started. The value of operand "Tag_Value1" is raised to a power specified with the value of operand "Tag_Value2". The result is stored in the "Tag_Result" output. If the instruction is

executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of ascending address.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data type at the IN parameter does not correspond to the specified data type at the OUT1 parameter.

The following table lists the possible transfers for the S7-1200 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE ,TOD, CHAR
WORD	WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
USINT	USINT, UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
INT	INT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UINT	UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
UDINT	UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LREAL
TIME	TIME	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME
DATE	DATE	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, DATE
TOD	TOD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, CHAR, Character of a string[1)]
WCHAR	WCHAR	BYTE, WORD, DWORD, CHAR, WCHAR, character of a character string[1)]
Character of a string[1)]	Character of a string	CHAR, WCHAR, character of a string
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER

The following table lists the possible transfers for the S7-1500 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE ,TOD, LTOD, CHAR
WORD	WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, S5TIME, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
DWORD	DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
LWORD	LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LREAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
USINT	USINT, UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
INT	INT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UINT	UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD. LTOD
DINT	DINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UDINT	UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
LINT	LINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
ULINT	ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LWORD, LREAL
S5TIME	S5TIME	WORD, S5TIME
TIME	TIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME
LTIME	LTIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTIME
DATE	DATE	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, DATE
DT	DT	DT
LDT	LDT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LDT
TOD	TOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TOD
LTOD	LTOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, LWORD, CHAR, Character of a string[1)]
WCHAR	WCHAR	BYTE, WORD, DWORD, LWORD, CHAR, WCHAR, Character of a string[1)]

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
Character of a string[1)]	Character of a string	CHAR, WCHAR, character of a string
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
COUNTER	COUNTER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	TIMER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_LTIMER	IEC_LTIMER	IEC_LTIMER
IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNT-ER	IEC_LCOUNTER	IEC_LCOUNTER
IEC_UL-COUNTER	IEC_ULCOUNTER	IEC_ULCOUNTER
REF()	There is no data type conversion. The transferred tag must correspond exactly to the target data type.	

[1)] You can also use the "Move value" instruction to transfer individual characters of a string to operands of the CHAR or WCHAR data type. The number of the character to be transferred is specified in square brackets next to the operand name. "MyString[2]", for example, transfers the second character of the "MyString" string. The transfer of the operand of the data type CHAR or WCHAR to the individual character of a string is also possible. You can also replace a specific character of a string with the character of another string.

[2)] Transferring entire arrays (ARRAY) is possible only when the array components of the operands at input IN and at output OUT1 are of the same data type.

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at input IN is less than the bit length of the data type at output OUT1, the higher-order bits of the destination value will be overwritten with zeros.

In its initial state, the instruction box contains 1 output (OUT1). The number of outputs can be extended. The added outputs are numbered in ascending order on the box. During the execution of the instruction, the content of the operand at the input IN is transferred to all available outputs. The instruction box cannot be extended if structured data types (DTL, STRUCT, ARRAY) or characters of a string are transferred.

You can also use the "Move block" (MOVE_BLK) and "Move block uninterruptible" (UMOVE_BLK) instructions to move operands of the ARRAY data type. Operand of the data type STRING or WSTRING can be copied with the instruction "Move character string" (S_MOVE).

Parameter

The following table shows the parameters of the "Move value" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, timers, date and time, CHAR, WCHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, timers, date and time, CHAR; WCHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data type, PLC data type (UDT)	I, Q, M, D, L or constant	Source value
OUT1	Output	Bit strings, integers, floating-point numbers, timers, date and time, CHAR, WCHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, timers, date and time, CHAR; WCHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data type, PLC data type (UDT)	I, Q, M, D, L	Operands to which in the source value is transferred.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

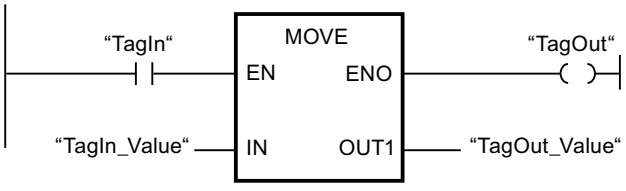

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If the "TagIn" operand returns signal state "1", the instruction is executed. The instruction copies the contents of operand "TagIn_Value" to operand "TagOut_Value" and sets output "TagOut" to signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

Note

You can find more information on the MOVE instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

MOVE_BLK: Move block (Page 776)

UMOVE_BLK: Move block uninterruptible (Page 783)

S_MOVE: Move character string (Page 2124)

Basic information on LAD (Page 7121)

Deserialize: Deserialize

Description

You can use the "Deserialize" instruction to convert back the sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type> and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type> is located must have the ARRAY of BYTE or ARRAY of CHAR data type and be declared with standard access in version 1.0. Optimized memory areas also permitted as of version 2.0. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion. If the memory area is filled using the instruction "Serialize" and it is possible that fill bytes were inserted, their conversion is not taken into account.

It is recommended to define the low limit of the ARRAY with "0", because then the index within ARRAY corresponds to the value of the POS parameter, e.g. ARRAY[0] = POS 0. The description and example below have been formed based on this.

The instruction enables you to convert multiple sequential representations of converted data back to their original state step-by-step.

If you only want to convert back a single sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type>, you can also directly use the instruction "TRCV: Receive data via communication connection".

Size of the memory area

The alignment rules mean that simple structures in the optimized memory area do not contain filling bytes. This results in a structure in the optimized memory area being smaller than in the standard memory area. ARRAYs of structures and structures in which other structures are located contain filling bytes. There is no general rule as to the memory area in which a composed structure requires more space.

Applies to CPUs of the S7-1500 series:

For a block with the block property "Optimized block access", the BOOL has a length of 1 byte. A structure that consists largely of the data type BOOL can therefore be larger in the optimized memory area than in the standard memory area. Composed structures with a low proportion of BOOL data types are smaller in the optimized memory area than in the standard memory area.

Optimized memory area

To deserialize larger structures, you can declare the memory area for sequential representation with optimized access for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. The content of the sequential representation remains unchanged, as for a standard memory area. Only symbolic access to the bytes in the ARRAY is possible.

Parameters

The following table shows the parameters of the "Deserialize" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_ARRAY	Input	VARIANT ARRAY[*] of BYTE [1)]	D, block interface of an FB (the sections Input, Output, Static and Temp are possible.)	Pointer to an ARRAY of BYTE or ARRAY of CHAR in which the data stream is saved.
DEST_VARIABLE	InOut	VARIANT	D, block interface of an FB	Pointer to a tag of the STRUCT, ARRAY or PLC data type (UDT) to be deserialized.

Parameters	Declaration	Data type	Memory area	Description
POS	InOut	DINT	I, Q, M, D, L	The operand at the POS parameter stores the index of the first byte based on the number of bytes occupied by the converted customer data. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information
[1)] Possible for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The tag at the SRC_ARRAY parameter is not in a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains a ZERO pointer.
8151	No valid reference at the SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8154	Invalid data type at the SRC_ARRAY parameter
8250	A NULL pointer was transferred at parameter DEST_ARRAY
8251	No valid reference at the DEST_VARIABLE parameter
8382	The value at the POS parameter is outside the limits of the array.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

For a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0, the following error code has a different meaning:

Error code* (W#16#...)	Explanation
8136	Access to the memory area at the SRC_ARRAY parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface of an FB or FC.
BufferPos	DINT	In the "Temp" section of the block interface of an FB or FC.
Error	INT	
Label	STRING[4]	

The following table shows the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

The following example shows how the instruction works:

Network 1:

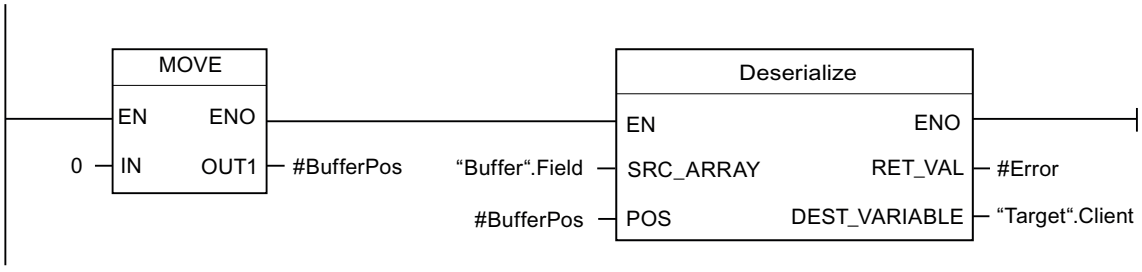

The "Move value" instruction moves the value "0" to the #BufferPos operand. The "Deserialize" instruction deserializes the sequential representation of the customer data from the "Buffer" tag and writes it to the "Target" tag. The #BufferPos operand stores the index of the first byte based on the number of bytes occupied by the converted customer data.

Network 2:

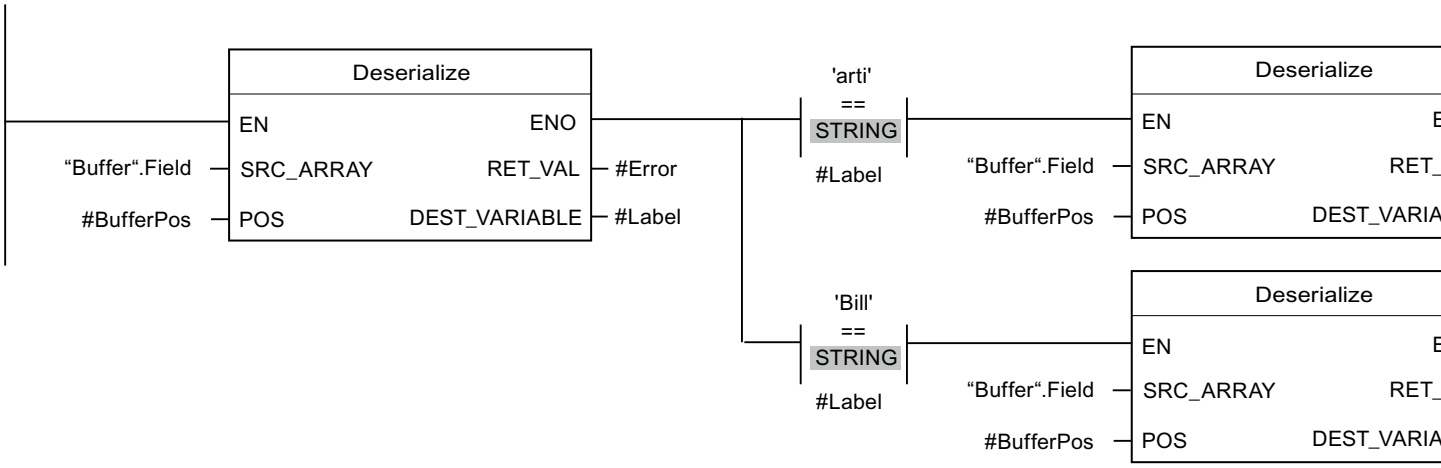

The "Deserialize" instruction deserializes the sequential representation of the separator sheet, which was stored in the sequential representation after the customer data, from the "Buffer" tag and writes the characters to the #Label operand. The characters are compared using comparison instructions "arti" and "Bill". If the comparison for "arti" = TRUE, these are article data that have been deserialized and written to the "Target" tag. If the comparison for "Bill" = TRUE, these are billing data that have been deserialized and written to the "Target" tag.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on VARIANT (Page 324)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of PLC data types (UDT) (Page 273)

Structure of an ARRAY tag (Page 287)

Structure of a STRUCT tag (Page 280)

Structure of a STRING tag (Page 266)

Basic information on LAD (Page 7121)

Serialize: Serialize

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT), STRUCT or ARRAY of <data type> to a sequential representation without losing parts of their structure.

You use the instruction to temporarily save multiple structured data items from your program in a buffer, which should preferably be in a global data block, and send them to another CPU. The memory area in which the converted data is stored must have the ARRAY of BYTE or ARRAY of CHAR data type and be declared with standard access in version 1.0. Optimized data is also permitted as of version 2.0. Fill data of the source data area is undefined in the target array. These can be fill bytes or fill bits of a data area (e.g. ARRAY, STRUCT or PLC data type (UDT)) as well as the characters of a string currently not in use.

The capacity of the standard memory area is 64 KB. Structures that are larger than 64 KB in accordance with the rules for standard memory areas cannot be serialized if the operand at the DEST_ARRAY parameter is located in a standard memory area.

It is recommended to define the low limit of the ARRAY with "0", because then the index within ARRAY corresponds to the value of the POS parameter, e.g. ARRAY[0] = POS 0. The description and example below have been formed based on this.

The operand at the POS parameter contains information about the number of bytes used by the data.

If you want to send a single PLC data type (UDT), STRUCT or ARRAY of <data type>, you can directly call the instruction "TSEND: Send data via communication connection".

Note

Comparison of structures

To compare structures, they have to be serialized first. Use the "CMP" instruction instead.

You can find additional information in the section "Comparator".

Size of the memory area

The alignment rules mean that simple structures in the optimized memory area do not contain filling bytes. This results in a structure in the optimized memory area being smaller than in the standard memory area. ARRAYs of structures and structures in which other structures are located contain filling bytes. There is no general rule as to the memory area in which a composed structure requires more space.

Applies to CPUs of the S7-1500 series:

For a block with the block property "Optimized block access", the length of the data type BOOL depends on the data type that follows. This means if a BYTE follows, for example, the data type BOOL has a length of 1 byte. When a WORD follows, for example, the data type BOOL has a length of 2 bytes. A structure that consists largely of the data type BOOL can therefore be larger in the optimized memory area than in the standard memory area. Composed structures with a low proportion of BOOL data types are smaller in the optimized memory area than in the standard memory area.

We therefore recommend that the source data area for the serialization starts with the large data types and ends with Boolean elements. This greatly reduces the filling with filling bits.

Optimized memory area

To serialize larger structures, you can declare the memory area with optimized access for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. The sequential representation remains unchanged, as for a standard memory area.

Parameters

The following table shows the parameters of the instruction "Serialize":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_VARIABLE	Input	VARIANT	D, block interface of an FB	Pointer to a tag of the STRUCT, ARRAY or PLC data type (UDT) to be serialized.
DEST_ARRAY	InOut	VARIANT	D, block interface of an FB (the sections Input, Output, Static and Temp are possible.)	Pointer to an ARRAY of BYTE or ARRAY of CHAR in which the generated data stream is saved.
POS	InOut	DINT	I, Q, M, D, L	The operand at the POS parameter stores the index of the first byte based on the total number of bytes that the converted customer data has occupied. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains a ZERO pointer.
8151	No valid reference at the SRC_VARIABLE parameter
8236	The tag at the SRC_ARRAY parameter is not in a block with standard access.
8250	A NULL pointer was transferred at parameter DEST_ARRAY.
8251	No valid reference at the DEST_ARRAY parameter
8253	The tag at the parameter DEST_ARRAY does not provide enough memory space for the content of the tag at the parameter SRC_VARIABLE. Due to the input value of the tag at parameter POS the available memory space is reduced. The input value of the parameter POS decides where to begin within the tag at the parameter DEST_ARRAY.

Error code* (W#16#...)	Explanation
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at the POS parameter is outside the limits of the array.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

For a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0, the following error code has a different meaning:

Error code* (W#16#...)	Explanation
8236	Access to the memory area at the DEST_ARRAY parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

Network 1:

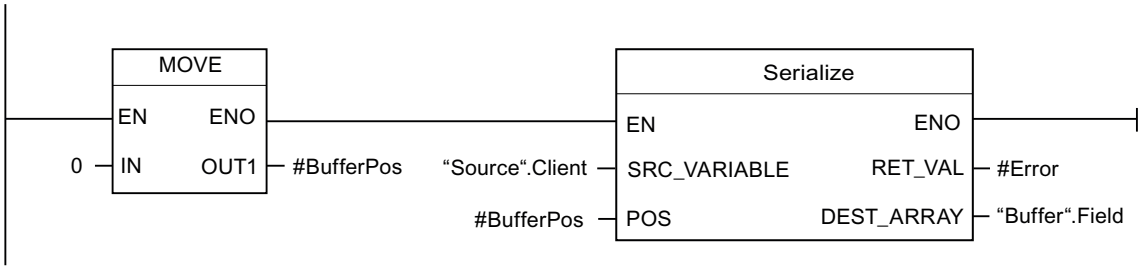

The "Move value" instruction moves the value "0" to the #BufferPos operand. The "Serialize" instruction serializes the customer data from the "Source" tag and writes the sequential representation to the "Buffer" tag. The index of the next unwritten byte of the "Buffer".Field operand is saved in the #BufferPos operand.

Network 2:

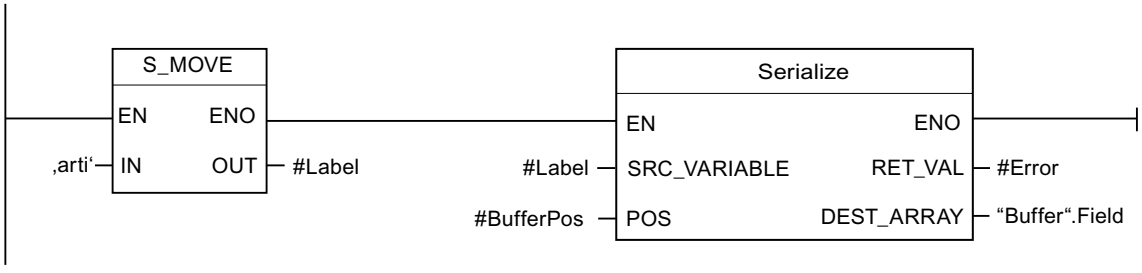

A kind of separator sheet is inserted now to make it easier to deserialize the sequential representation later. The "Move character string" instruction moves the "arti" characters to the #Label operand. The "Serialize" instruction writes these characters after the customer data to the "Buffer" tag. The value of the #BufferPos operand is incremented accordingly.

Network 3:

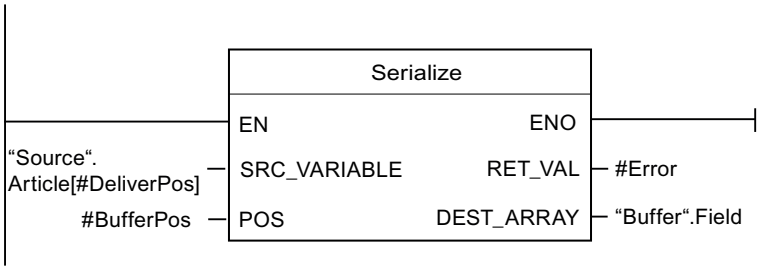

The "Serialize" instruction serializes the data of a specific article, which is calculated in runtime, from the "Source" tag and writes it in sequential representation to the "Buffer" tag after the "arti" characters.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table shows the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on VARIANT (Page 324)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of PLC data types (UDT) (Page 273)

Structure of an ARRAY tag (Page 287)

Structure of a STRUCT tag (Page 280)

Structure of a STRING tag (Page 266)

Basic information on LAD (Page 7121)

MOVE_BLK: Move block

Description

You use the "Move block" instruction to move the content of a memory area (source range) to another memory area (target range). The number of elements to be moved to the target range is specified at input COUNT. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source range and the target range have the same data type.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Note

Use of ARRAYs

The instruction copies the contents from the defined element n elements (n = depending on the value at the parameter COUNT) from the source range to the target range, starting at the specified index.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be moved from the source range to the target range.
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the target range to which the contents of the source range are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.					

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

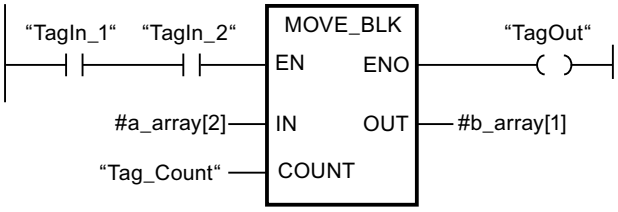

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the "a_array" operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the "b_array" operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

Note

You can find more information on the MOVE_BLK instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

MOVE_BLK_VARIANT: Move block

Description

You use the "Move block" instruction to move the content of a memory area (source range) to another memory area (target range). You can copy a complete ARRAY or elements of an ARRAY to another ARRAY of the same data type. The size (number of elements) of source and destination ARRAY may be different. You can copy multiple or single elements within an ARRAY.

The number of elements that are to be copied must not exceed the selected source range or target range.

If you use the instruction at the time the block is being created, the ARRAY does not yet have to be known, as the source and the target are transferred per VARIANT.

Counting at the parameters SRC_INDEX and DEST_INDEX always begins with the low limit "0", regardless of the later declaration of the ARRAY.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is copied than is made available.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input [2)]	VARIANT (which points to an ARRAY or an individual ARRAY element), ARRAY of <Data_type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L or constant	Number of elements which are copied Assign the value "1" to the parameter COUNT, if no ARRAY is specified at parameter SRC or at parameter DEST.
SRC_INDEX	Input	DINT	I, Q, M, D, L or constant	Defines the first element to be copied: • The SRC_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter SRC, the integer at parameter SRC_INDEX specifies the first element within the source area from which it is to be copied. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter SRC or only one single element of an ARRAY is specified, then assign the value "0" at parameter SRC_INDEX.

Parameter	Declaration	Data type	Memory area	Description
DEST_INDEX	Input	DINT	I, Q, M, D, L or constant	Defines the start of the destination memory area: • The DEST_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter DEST, the integer at parameter DEST_INDEX specifies the first element within the target range that is to be copied into. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter DEST, then assign the value "0" at parameter DEST_INDEX.
DEST	Output [1)]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Destination area into which the contents of the source block are copied.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
1) The DEST parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface. 2) With the SRC parameter, the data types BOOL and ARRAY of BOOL are not permitted.				

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data types do not correspond. Instead of an ARRAY of Struct, use an ARRAY of PLC data type (UDT).
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at the SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value

Error code* (W#16#...)	Explanation
8382	The value at the SRC_INDEX parameter is less than zero.
8383	The value at the SRC_INDEX is outside the high limit of the ARRAY.
8482	The value at the DEST_INDEX parameter is less than zero.
8483	The value at the DEST_INDEX is outside the high limit of the ARRAY.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at the DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

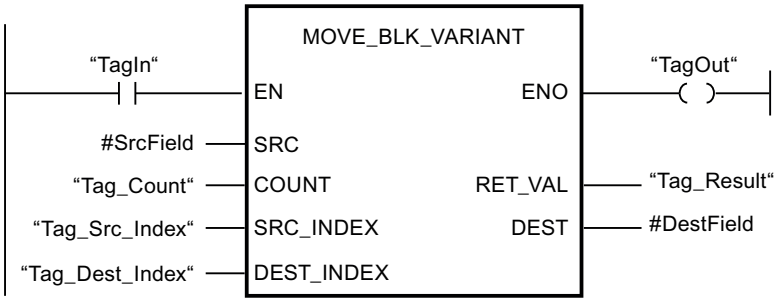

The following table shows how the instruction works using specific operand values:

Parameter	Declaration in the block interface	Operand	Value
SRC	Input	#SrcField	The local operand #SrcField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[0..10] of "MOVE_UDT")
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3

Parameter	Declaration in the block interface	Operand	Value
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	The local operand #DestField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[10..20] of "MOVE_UDT")

If the "TagIn" operand returns signal state "1", the instruction is executed. Two elements are moved from the source range, starting from the fourth element of the ARRAY [0..10] of MOVE_UDT, to the target range. The copies are pasted in the ARRAY [10..20] of MOVE_UDT starting from the fourth element. If the instruction is executed without errors, the enable output ENO has the signal state TRUE and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

Note

You can find more information on the MOVE_BLK_VARIANT instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

VariantGet: Read out VARIANT tag value (Page 823)

Example of moving data (Page 333)

Basic information on VARIANT (Page 324)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

UMOVE_BLK: Move block uninterruptible

Description

You use the "Move block uninterruptible" instruction to move the content of a memory area (source range) to another memory area (target range). The instruction cannot be interrupted. The number of elements to be moved to the target range is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source range and the target range have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Note

Use of ARRAYs

The instruction copies the contents from the defined element n elements (n = depending on the value at the parameter COUNT) from the source range to the target range, starting at the specified index.

You use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be moved from the source range to the target range.
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the target range to which the contents of the source range are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.					

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

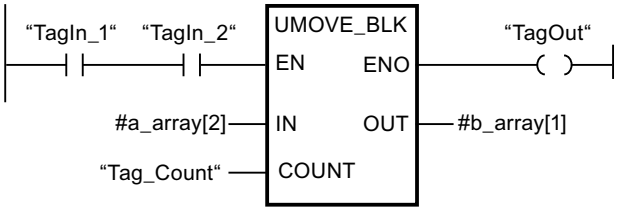

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the "a_array" operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the "b_array" operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The move operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

Note

You can find more information on the UMOVE_BLK instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (target range) with the value of the IN input. The target range is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is moved to the target range as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source range and the target range have the same data type.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The maximum number of elements changed is the number of elements in the ARRAY or structure. If you copy more data than there are elements at the OUT output, you will get an unintended result.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Note

Use of ARRAYs

The instruction reads the content from the selected element from the source range and copies this content n-times (n = depending on the value at the parameter COUNT) to the destination, starting at the specified index.

Filling structures

As well as the elements of an ARRAY, you can also fill multiple elements of a structure (STRUCT, PLC data type) with the same value. The structure whose elements you want to fill must only contain individual elements of the same elementary data type. The structure itself can, however, be embedded in another structure.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P, or constant	Element used to fill the target range
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P, or constant	Number of repeated move operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	Address in target range from which filling starts

For additional information on valid data types, refer to "See also".

Example with an ARRAY

The following example shows how the instruction works when you want to fill an ARRAY:

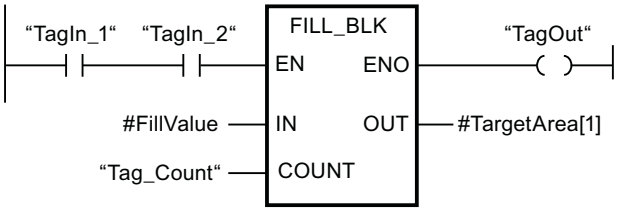

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	FillValue	The data type of the operand is INT.
COUNT	Tag_Count	3
OUT	TargetArea	The data type of the operand TargetArea is ARRAY[1..5] of INT. It consists of five elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction copies the value of the operand #FillValue to the #TargetArea output tag three times, starting with the first element. If the instruction is executed without errors, the TagOut operand at the "ENO" enable output is set to signal state "1".

Examples with a structure

The following examples show how the instruction works when you want to fill a structure:

Create a global data block with the following elements:

Data_block_1		Data type
MyStruct1		STRUCT
	Member_1	INT
	Member_2	INT
	Member_3	INT
	Member_4	INT
MyStruct2		STRUCT

Data_block_1					Data type
	SubArray				ARRAY[1..2] of STRUCT
		SubArray[1]			STRUCT
			NestedStruct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT
		SubArray[2]			STRUCT
			Nested-Struct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT

Generate the following program code to address the MyStruct1 tag:

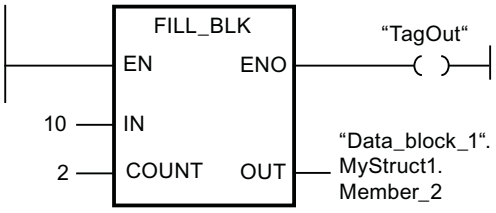

Generate the following program code to address the MyStruct2 tag:

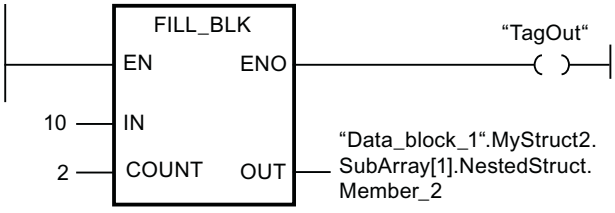

In each of the two examples, the value 10 of the IN parameter is copied twice to the operand at the OUT parameter, starting with the element Member_2. This means that the value 10 is copied to the elements Member_2 and Member_3. The other two elements, Member_1 and Member_4, are not changed. If the instruction is executed without errors, the TagOut operand is set to signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

UFILL_BLK: Fill block uninterruptible

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (target range) with the value of the IN input. The instruction cannot be interrupted. The target range is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is moved to the target range as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source range and the target range have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The maximum number of elements changed is the number of elements in the ARRAY or structure. If you copy more data than there are elements at the OUT output, you will get an unintended result.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Note

Use of ARRAYs

The instruction reads the content from the selected element from the source range and copies this content n-times (n = depending on the value at the parameter COUNT) to the destination, starting at the specified index.

You use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Filling structures

As well as the elements of an ARRAY, you can also fill multiple elements of a structure (STRUCT, PLC data type) with the same value. The structure whose elements you want to fill must only contain individual elements of the same elementary data type. The structure itself can, however, be embedded in another structure.

Parameters

The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P, or constant	Element used to fill the target range.
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P, or constant	Number of repeated move operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	Address in target range from which filling begins.

For additional information on valid data types, refer to "See also".

Example with an ARRAY

The following example shows how the instruction works when you want to fill an ARRAY:

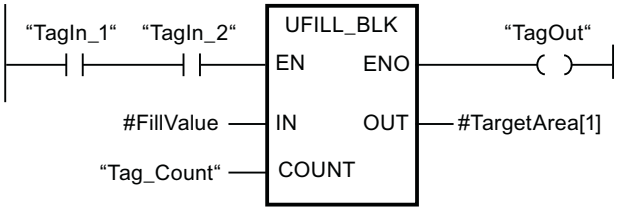

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	FillValue	The data type of the operand is INT.
COUNT	Tag_Count	3
OUT	TargetArea	The data type of the operand TargetArea is ARRAY[1..5] of INT. It consists of five elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction copies the value of the operand #FillValue to the #TargetArea output tag three times, starting with the first element. The move operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the TagOut operand at the "ENO" enable output is set to signal state "1".

Examples with a structure

The following examples show how the instruction works when you want to fill a structure:

Create a global data block with the following elements:

Data_block_1					Data type
MyStruct1					STRUCT
	Member_1				INT
	Member_2				INT
	Member_3				INT
	Member_4				INT
MyStruct2					STRUCT
	SubArray				ARRAY[1..2] of STRUCT
		SubArray[1]			STRUCT
			NestedStruct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT
		SubArray[2]			STRUCT
			Nested-Struct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT

Generate the following program code to address the MyStruct1 tag:

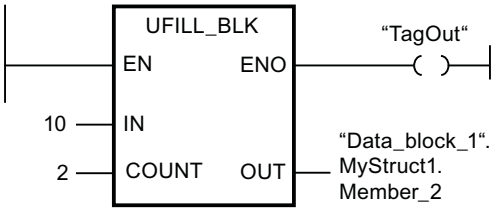

Generate the following program code to address the MyStruct2 tag:

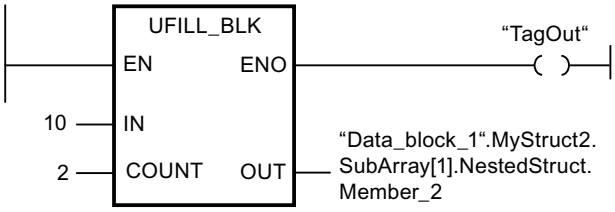

In each of the two examples, the value 10 of the IN parameter is copied twice to the operand at the OUT parameter, starting with the element Member_2. This means that the value 10 is copied to the elements Member_2 and Member_3. The other two elements, Member_1 and Member_4, are not changed. If the instruction is executed without errors, the TagOut operand is set to signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

SCATTER: Parse the bit sequence into individual bits

Description

The "Parse the bit sequence into individual bits" instruction parses a tag of the BYTE, WORD, DWORD or LWORD data type into individual bits and saves them in an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements.

Note

Multi-dimensional ARRAY of BOOL

With the "Parse the bit sequence into individual bits" instruction, the use of a multidimensional ARRAY of BOOL is not permitted.

Note

Length of the ARRAY, STRUCT or PLC data type

The ARRAY, the anonymous STRUCT or the PLC data type must have exactly the number of elements that is specified by the bit sequence.

This means for the data type BYTE, for example, the ARRAY, STRUCT or the PLC data type must have exactly 8 elements (WORD = 16, DWORD = 32 and LWORD = 64).

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

This way you can, for example, parse a status word, and read and change the status of the individual bits using the index. Using GATHER, you can merge the bits once again into a bit sequence.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- If the ARRAY, STRUCT or PLC data type does not provide enough BOOL elements.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD	I, Q, M, D, L	Bit sequence that is parsed. The values must not be located in the I/O area or in the DB of a technology object.
OUT	Output	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	I, Q, M, D, L	ARRAY, STRUCT or PLC data type in which the individual bits are stored

You can find additional information on valid data types under "See also".

Example with an ARRAY

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceWord		WORD

Tag	Section	Data type
EnableOut	Output	BOOL
DestinationArray		ARRAY[0..15] of BOOL

The following example shows how the instruction works:

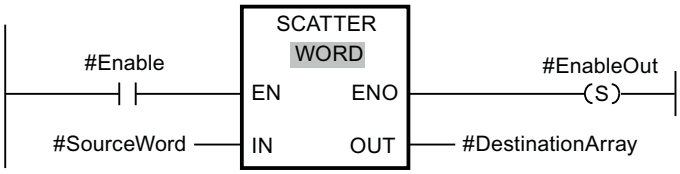

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceWord	WORD (16 bits)
OUT	DestinationArray	The operand "DestinationArray" is of the data type ARRAY[0..15] of BOOL. It consists of 16 elements and is therefore just as large as the WORD that is to be parsed.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The operand #SourceWord of the data type WORD is parsed into its individual bits (16) and assigned to the individual elements of the #DestinationArray operand. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Example with a PLC data type

Create the following PLC data type "myBits":

myBits		
	Name	Data type
1	GeneralError	Bool
2	DeviceError	Bool
3	CommError	Bool
4	myError1	Bool
5	myError2	Bool
6	myError3	Bool
7	myError4	Bool
8	myError5	Bool
9	myError6	Bool
10	myError7	Bool
11	myError8	Bool
12	myError9	Bool
13	myError10	Bool
14	myError11	Bool
15	myError12	Bool
16	myError13	Bool

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceWord		WORD
EnableOut	Output	BOOL
DestinationUDT		"myBits"

The following example shows how the instruction works:

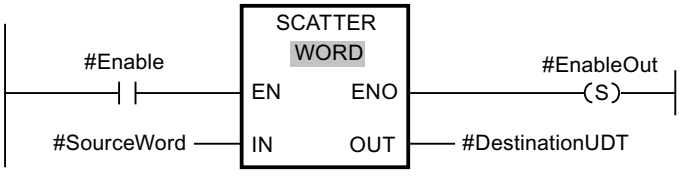

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceWord	WORD (16 bits)
OUT	DestinationUDT	The "DestinationUDT" operand has the PLC data type (UDT). It consists of 16 elements and is therefore just as large as the WORD that is to be parsed.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The #SourceWord operand of the data type WORD is parsed into its individual bits (16) and assigned to the individual elements of the #DestinationArray operand. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

SCATTER_BLK: Parse elements of an ARRAY of bit sequence into individual bits

Description

The "Parse elements of an ARRAY of bit sequence into individual bits" instruction parses one or more elements of an ARRAY of BYTE, WORD, DWORD or LWORD into individual bits and saves them in an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements. At the COUNT_IN parameter you specify how many elements of the source ARRAY are going to be parsed. The source ARRAY at the IN parameter may have more elements than specified at the COUNT_IN parameter. The ARRAY of BOOL, the anonymous STRUCT or the PLC data type must have sufficient elements to save the bits of the parsed bit sequences. However, the destination memory area may also be larger.

Note

Multi-dimensional ARRAY of BOOL

If the ARRAY is a multi-dimensional ARRAY of BOOL, the filling bits of the dimensions contained are counted as well even if they were not explicitly declared.

Example 1: An ARRAY[1..10,0..4,1..2] of BOOL is handled like an ARRAY[1..10,0..4,1..8] of BOOL or like an ARRAY[0..399] of BOOL.

Example 2: At the IN parameter, an ARRAY[0..5] of WORD (sourceArrayWord[2]) is interconnected. The COUNT_IN parameter has the value "3". At the OUT parameter, an ARRAY[0..1,0..5,0..7] of BOOL (destinationArrayBool[0,0,0]) is interconnected. Both the array at the IN parameter and at the OUT parameter has a size of 96 bits. The ARRAY of WORD is parsed into 48 individual bits.

Note

If the ARRAY low limit of the target ARRAY is not "0", note the following:

For performance reasons the index must always start at a BYTE, WORD, DWORD or LWORD limit. This means the index must be calculated starting at the low limit of the ARRAY. The following formula is used as basis for this calculation:

Valid indices = ARRAY low limit + n(number of bit sequences) × number of bits of the desired bit sequence

For an ARRAY[-2..45] of BOOL and the bit sequence WORD the calculation looks as follows:

- Valid index (-2) = -2 + 0 × 16
- Valid index (14) = -2 + 1 × 16
- Valid index (30) = -2 + 2 × 16

You can find an example described below.

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

This way you can, for example, parse status words, and read and change the status of the individual bits using the index. Using GATHER, you can merge the bits once again into a bit sequence.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- The source ARRAY has fewer elements than specified at the COUNT_IN parameter.
- The index of the destination ARRAY does not start at a BYTE, WORD, DWORD or LWORD limit. In this case, no result is written to the ARRAY of BOOL.
- The ARRAY[*] of BOOL, STRUCT or PLC data type does not provide the required number of elements. In this case as many bit sequences as possible are parsed and written to the ARRAY of BOOL, the anonymous STRUCT or the PLC data type. The remaining bit sequences are no longer taken into account.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Element of an ARRAY[*] of <bit sequence>	Element of an ARRAY[*] of <bit sequence>	I, Q, M, D, L	ARRAY of <bit sequence> that is parsed. The values must not be located in the I/O area or in the DB of a technology object.

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
COUNT_IN	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Counter for the number of elements of the source ARRAY that are going to be parsed. The value must not be in the I/O area or in the database of a technology object.
OUT	Output	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	I, Q, M, D, L	ARRAY, STRUCT or PLC data type in which the individual bits are stored

You can select the required bit sequence from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example of a destination ARRAY with the low limit "0"

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceArrayWord		ARRAY[0..5] of WORD
CounterInput		UDINT
EnableOut	Output	BOOL
DestinationArrayBool		ARRAY[0..95] of BOOL

The following example shows how the instruction works:

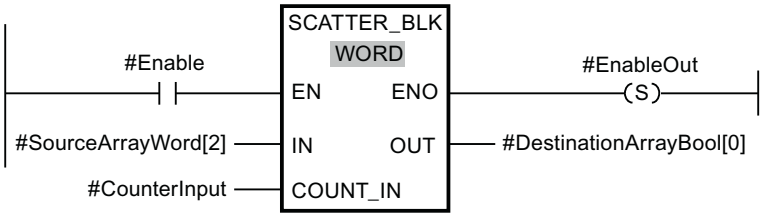

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayWord[2]	ARRAY[0..5] of WORD (96 bits can be parsed.)
COUNT_IN	CounterInput = 3	UDINT3 (3 WORDs or 48 bits are to be parsed. This means at least 48 bits must be available in the destination ARRAY.)
OUT	DestinationArrayBool[0]	The operand "DestinationArray-Bool" is of the data type AR-RAY[0..95] of BOOL. This means it provides 96 BOOL ele-ments.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The 3rd, 4th and 5th WORD of the #SourceArrayWord operand is parsed into its individual bits (48) and as of the 1st element assigned to the individual elements of the #DestinationArrayBool operand. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

Example of a destination ARRAY with the low limit "-2"

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceArrayWord		ARRAY[0..5] of WORD
CounterInput		UDINT
EnableOut	Output	BOOL
DestinationArrayBool		ARRAY[-2..93] of BOOL

The following example shows how the instruction works:

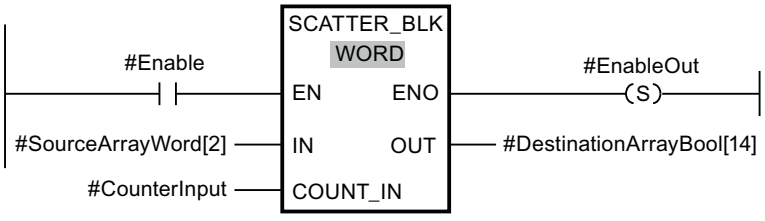

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayWord[2]	ARRAY[0..5] of WORD (96 bits can be parsed.)
COUNT_IN	CounterInput = 3	UDINT3 (3 WORDs or 48 bits are to be parsed. This means at least 48 bits must be available in the destination ARRAY.)
OUT	DestinationArrayBool[14]	The operand "DestinationArray-Bool" is of the data type AR-RAY[-2..93] of BOOL. This means it provides 96 BOOL ele-ments.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The 3rd, 4th and 5th WORD of the #SourceArrayWord operand is parsed into its individual bits (48) and as of the 16th Element assigned to the individual elements of the #DestinationArrayBool operand. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO. The remaining 32 bits are not written.

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

GATHER: Merge individual bits into a bit sequence

Description

The "Merge individual bits into a bit sequence" instruction merges the bits from an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements into a bit sequence. The bit sequence is saved in a tag of the data type BYTE, WORD, DWORD or LWORD.

Note

Multi-dimensional ARRAY of BOOL

With the "Merge individual bits into a bit sequence" instruction, the use of a multidimensional ARRAY of BOOL is not permitted.

Note

Length of the ARRAY, STRUCT or PLC data type

The ARRAY, STRUCT or the PLC data type must have exactly the number of elements that is specified by the bit sequence.

This means for the data type BYTE, for example, the ARRAY, anonymous STRUCT or the PLC data type must have exactly 8 elements (WORD = 16, DWORD = 32 and LWORD = 64).

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- If the ARRAY, the anonymous STRUCT or the PLC data type (UDT) has fewer or more BOOL elements than specified by the bit sequence, they are not transferred and the ENO returns signal state "0". The ENO also returns "0" if fewer than the necessary number of bits is available.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	I, Q, M, D, L	ARRAY, STRUCT or PLC data type, the bits of which are merged into a bit sequence. The values must not be located in the I/O area or in the DB of a technology object.
OUT	Output	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD	I, Q, M, D, L	Merged bit sequence, saved in a tag

You can select the required bit sequence from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example with an ARRAY

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceArray		ARRAY[0..15] of BOOL
EnableOut	Output	BOOL
DestinationWord		WORD

The following example shows how the instruction works:

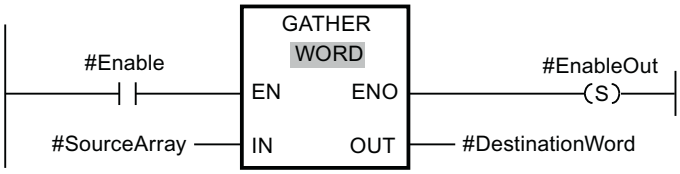

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArray	The operand "SourceArray" is of the data type ARRAY[0..15] of BOOL. It consists of 16 elements and is therefore just as large as the WORD in which the bits are to be merged.
OUT	DestinationWord	WORD (16 bits)

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The bits of the #SourceArray operand are merged into a WORD. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Example with a PLC data type (UDT)

Create the following PLC data type "myBits":

myBits		
	Name	Data type
1	GeneralError	Bool
2	DeviceError	Bool
3	CommError	Bool
4	myError1	Bool
5	myError2	Bool
6	myError3	Bool
7	myError4	Bool
8	myError5	Bool
9	myError6	Bool
10	myError7	Bool
11	myError8	Bool
12	myError9	Bool
13	myError10	Bool
14	myError11	Bool
15	myError12	Bool
16	myError13	Bool

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceUDT		"myBits"
EnableOut	Output	BOOL
DestinationWord		WORD

The following example shows how the instruction works:

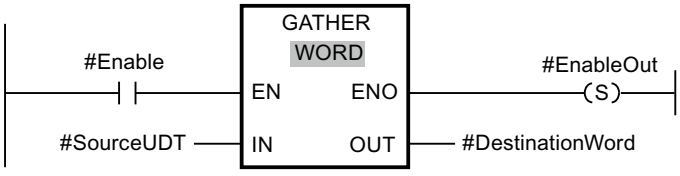

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceUDT	The "SourceUDT" operand has the PLC data type (UDT). It consists of 16 elements and is therefore just as large as the WORD in which the bits are to be merged.
OUT	DestinationWord	WORD (16 bits)

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The bits of the #SourceUDT operand are merged into a WORD. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

GATHER_BLK: Merge individual bits into multiple elements of an ARRAY of bit sequence

Description

The "Merge individual bits into multiple elements of an ARRAY of bit sequence" instruction merges the bits from an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements into one or multiple elements of an ARRAY of <bit sequence>. At the COUNT_OUT parameter you specify how many elements of the destination ARRAY are going to be written. With this step you also implicitly specify how many elements of the ARRAY of BOOL, the anonymous STRUCT or the PLC data type are required. The destination ARRAY at the OUT parameter may have more elements than specified at the COUNT_OUT parameter. The ARRAY of <bit sequence> must have sufficient elements to save the bits that are going to be merged. However, the destination ARRAY may also be larger.

Note

Multi-dimensional ARRAY of BOOL

If the ARRAY is a multi-dimensional ARRAY of BOOL, the filling bits of the dimensions contained are counted as well even if they were not explicitly declared.

Example 1: An ARRAY[1..10,0..4,1..2] of BOOL is handled like an ARRAY[1..10,0..4,1..8] of BOOL or like an ARRAY[0..399] of BOOL.

Example 2: At the OUT parameter, an ARRAY[0..5] of WORD (sourceArrayWord[2]) is interconnected. The COUNT_IN parameter has the value "3". At the IN parameter, an ARRAY[0..1,0..5,0..7] of BOOL (destinationArrayBool[0,0,0]) is interconnected. Both the array at the IN parameter and at the OUT parameter has a size of 96 bits. 48 individual bits are merged from the ARRAY of BOOL.

Note

If the ARRAY low limit of the source ARRAY is not "0", note the following:

For performance reasons the index must always start at a BYTE, WORD, DWORD or LWORD limit. This means the index must be calculated starting at the low limit of the ARRAY. The following formula is used as basis for this calculation:

Valid indices = ARRAY low limit + n(number of bit sequences) × number of bits of the desired bit sequence

For an ARRAY[-2..45] of BOOL and the bit sequence WORD the calculation looks as follows:

- Valid index (-2) = -2 + 0 × 16
- Valid index (14) = -2 + 1 × 16
- Valid index (30) = -2 + 2 × 16

You can find an example described below.

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- The index of the source ARRAY does not start at a BYTE, WORD, DWORD or LWORD limit. In this case, no result is written to the ARRAY of <bit sequence>.
- The ARRAY[*] of <bit sequence> does not provide the required number of elements. In this case as many bit sequences as possible are merged and written to the ARRAY of <bit sequence>. The remaining bits are no longer taken into account.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	I, Q, M, D, L	ARRAY of BOOL, STRUCT or PLC data type whose bits are merged (source ARRAY) The values must not be located in the I/O area or in the DB of a technology object.

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
COUNT_OUT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Counter how many elements of the target ARRAY are to be described. The value must not be in the I/O area or in the database of a technology object.
OUT	Output	Element of an ARRAY[*] of <bit sequence>	Element of an ARRAY[*] of <bit sequence>	I, Q, M, D, L	ARRAY of <bit sequence> to which the bits are saved (destination ARRAY)

You can select the required bit sequence from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example of a source ARRAY with the low limit "0"

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceArrayBool		ARRAY[0..95] of BOOL
CounterOutput		UDINT
EnableOut	Output	BOOL
DestinationArrayWord		ARRAY[0..5] of WORD

The following example shows how the instruction works:

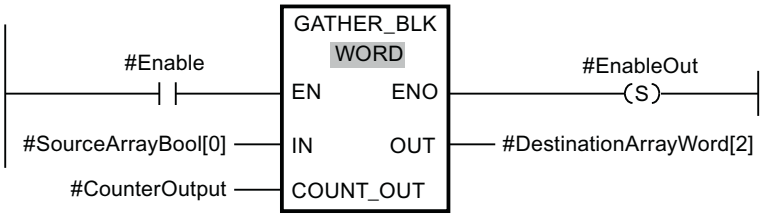

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayBool[0]	The operand "SourceArrayBool" is of the data type ARRAY[0..95] of BOOL. This means it provides 96 BOOL elements that can merged into words once again.
COUNT_OUT	CounterOutput = 3	UDINT3 (3 words are to be written. This means 48 bits must be available in the source ARRAY.)
OUT	DestinationArrayWord[2]	The operand "DestinationArrayWord" is of the data type ARRAY[0..5] of WORD. This means 6 WORD elements are available.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. As of the 1st element of the #SourceArrayBool operand, 48 bits are merged in the #DestinationArrayWord operand. The starting point in the destination ARRAY is the 3rd element. This means that the first 16 bits are written into the 3rd word, the second 16 nits into the 4th word and the third 16 bits into the 5th word of the destination ARRAY. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

Example of a source ARRAY with the low limit "-2"

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceArrayBool		ARRAY[-2..93] of BOOL
CounterOutput		UDINT
EnableOut	Output	BOOL
DestinationArrayWord		ARRAY[0..5] of WORD

The following example shows how the instruction works:

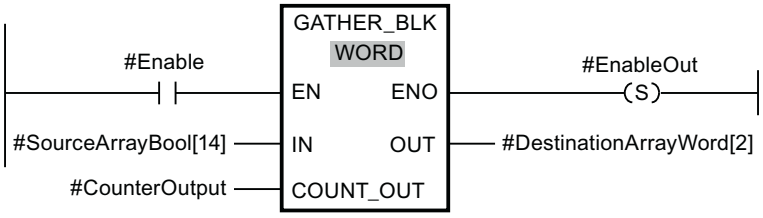

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayBool[14]	The operand "SourceArrayBool" is of the data type ARRAY[-2..93] of BOOL. Because the starting point is the 16th element, only 80 BOOL elements that can be merged into words again are available.
COUNT_OUT	CounterOutput = 3	UDINT3 (3 words are to be written. This means 48 bits must be available in the source ARRAY.)
OUT	DestinationArrayWord[2]	The operand "DestinationArrayWord" is of the data type ARRAY[0..5] of WORD. This means 6 WORD elements are available.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. Starting with the 16th element of the #SourceArrayBool operand, 48 bits are merged into the #DestinationArrayWord operand. The starting point in the destination ARRAY is the 3rd element. This means the first 16 bits of the source ARRAY are ignored. The second 16 Bits are written into the 3rd word, the third 16 bits into the 4th word and the fourth 16 bits into the 5th word of the destination ARRAY. The remaining 64 bits of the source ARRAY are not taken into account either. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

AssignmentAttempt: Attempt assignment of a VARIANT to a reference (S7-1500)

Description

With the "AssignmentAttempt" instruction, you attempt to assign a VARIANT tag to a reference tag. The data type of a reference tag is specified at the time of the declaration, while the data type of a VARIANT tag is determined during runtime. An implicit data type conversion is not permitted with reference tags. To assign the two data types to each other therefore, use the assignment attempt.

With the assignment attempt, a check is made at runtime to find out whether the VARIANT tag is of the correct data type. If this is the case, the instruction is executed. If the instruction is executed successfully, there is a valid reference in the target tag, otherwise a NULL.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL		Enable input
ENO	Output	BOOL		Enable output
SRC	Input	VARIANT	• Block interface of an FC: Input, Output, InOut, Temp • Block interface of an FB: Input, InOut, Temp	Pointer to the source tag whose address will be read out
DST	Output	Reference to: • Bit sequences, except BOOL, • Integers, • Floating-point numbers, • Character strings, • PLC data types (UDT), • System data types (SDT), • ARRAYs of the named data types	• Block interface of an FC: Input, Output, Temp, Return • Block interface of an FB: Temp	Reference to which the address of the source tag will be transferred.

You will find additional information on valid data types under "See also".

Rules

The following rules apply to the assignment attempt: VARIANT tags that do not match these rules return the value "NULL" at runtime.

- The VARIANT must point to an address in the optimized memory area.
- The VARIANT must not point to an address in a temporary memory area.
- If you want to assign a VARIANT to a reference to an ARRAY the following rules apply:
  - The VARIANT tag must point to an ARRAY whose limits exactly match with those of the declared reference. A VARIANT tag which points to an ARRAY [0..9], does not match a tag REF_TO ARRAY[1..10].
  - Moreover you should compile the blocks that form the value of the VARIANT tag once in a CPU of the S7-1500 series, firmware version V2.5.

Example

The following example shows how the instruction works:

In the interface of the block, the VARIANT "myVariant" and the reference tag "myReference" were declared.

	Name	Data type
	Input	
	Output	
	InOut	
	myVariant	Variant
	Static	
	Temp	
	myReference	REF_TO Int
	Constant	

In the program code an attempt is made to assign "myVariant" to the reference tag "myReference". If "myVariant" has the data type "Int" at runtime, "myReference" contains a valid reference to the source tag of the VARIANT, otherwise NULL. When the assignment was successful, the enable output "ENO" has the signal state "1" and the value "10" can be written to the target tag.

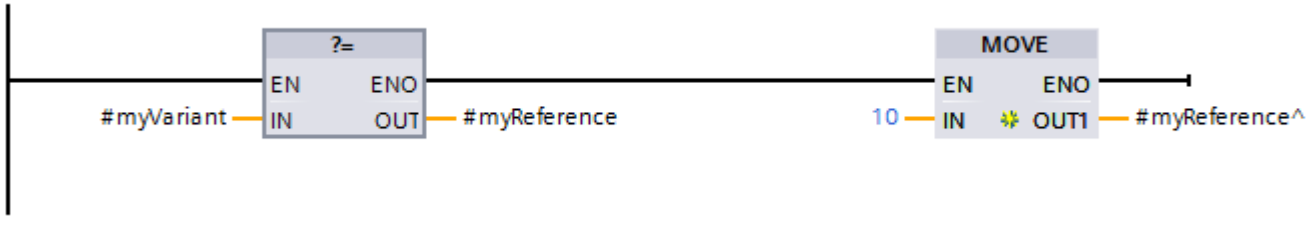

See also

Basic information about references (Page 300)

Assignment attempt from VARIANT to a reference (Page 314)

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Sample Library for Instructions (Page 593)

Basic information on LAD (Page 7121)

SWAP: Swap

Description

You can use the "Swap" instruction to change the order of the bytes at input IN and query the result at output OUT.

The following figure shows how the bytes of an operand of the DWORD data type are swapped using the "Swap" instruction:

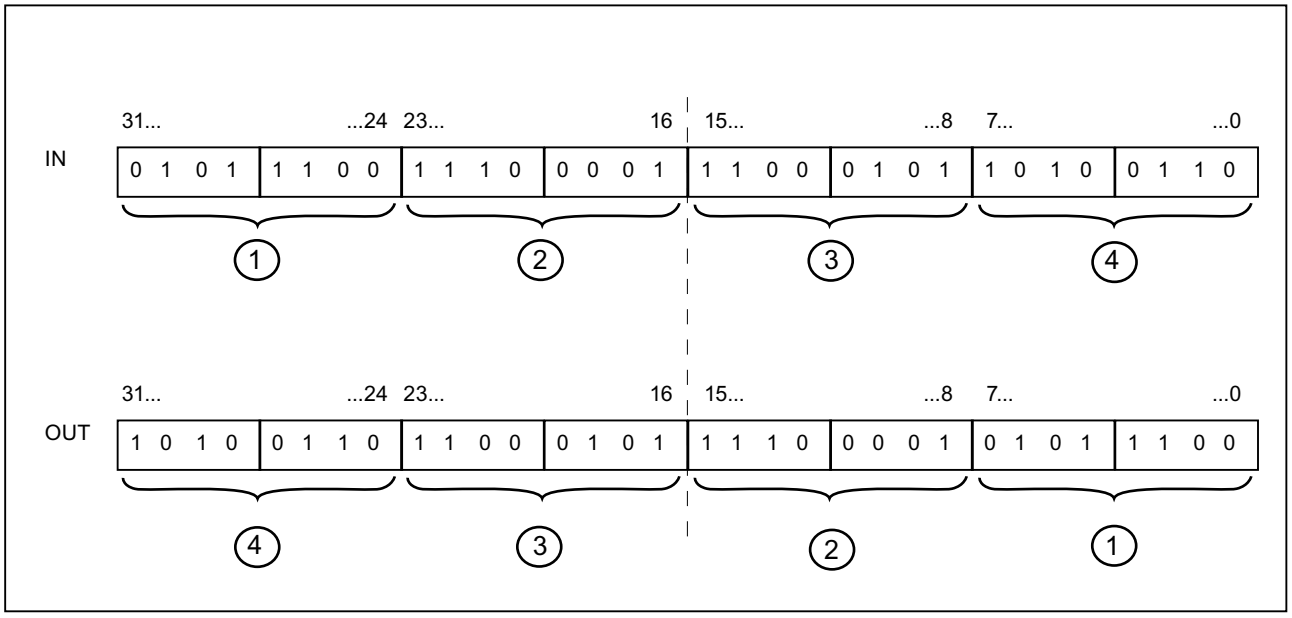

Parameters

The following table lists the parameters of the "Swap" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L or, P constant	Operand whose bytes are swapped.
OUT	Output	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

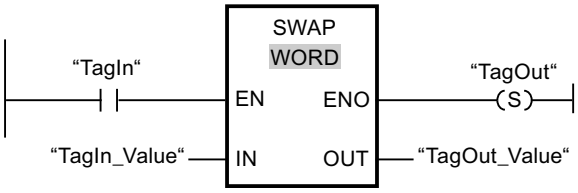

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 0101 0101
OUT	TagOut_Value	0101 0101 0000 1111

If operand "TagIn" has the signal state "1", the "Swap" instruction is executed. The order of the bytes is changed and stored in operand "TagOut_Value".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

ARRAY DB (S7-1500)

ReadFromArrayDB: Read from ARRAYdata block (S7-1500)

Description

The instruction "Read from ARRAY data block" is used to read the element from a data block of the ARRAY DB block type to which the index references, and write the value of the element to the target range.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

Enable output ENO returns the signal state FALSE if one of the following conditions applies:

- Enable input EN returns the signal state FALSE.
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Read from ARRAY data block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB that is read. The specification can be a constant, a global tag or an indexed value.
VALUE	Output [1)]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Value that is read and output
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
1) The VALUE parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, write protected, or is located in load memory.
8135	The ARRAY data block contains invalid values.
8154	The data block has the incorrect data type.
8282	The value at the INDEX parameter is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8452	Code generation error

Error code* (W#16#...)	Explanation
8453	There are two possible causes of the error: • The size of the VALUE parameter does not match the element length in the ARRAY data block. • The two tags are not in a memory area with optimized access. For additional information on the memory area access types, refer to: Basics of block access (Page 55)
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

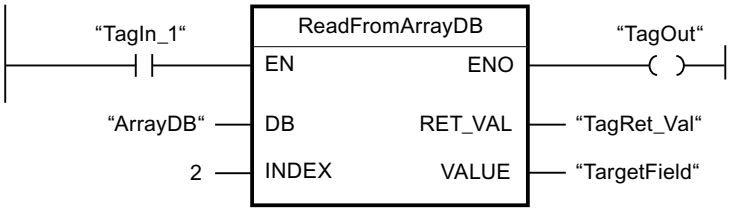

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type Array [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	TargetField	The "TargetField" operand is a global tag of the data type INT.

If the "TagIn1" operand returns signal state "1", the instruction is executed. The 2nd element is read from "ArrayDB" and written to the "TargetField" operand. If the instruction is executed without errors, the enable output ENO has the signal state TRUE and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Example of the use of ARRAY data blocks (Page 44)

Basic information on VARIANT (Page 324)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

WriteToArrayDB: Write to ARRAY data block (S7-1500)

Description

The instruction "Write to ARRAY data block" is used to write the element to which the index references to a data block of the ARRAY DB block type.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

Enable output ENO returns the signal state FALSE if one of the following conditions applies:

- Enable input EN returns the signal state FALSE.
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Write to ARRAY data block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written. The specification can be a constant, a global tag or an indexed value.
VALUE	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Value to be written
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, or is located in load memory.
8134	The data block is write protected.
8135	The data block is not an ARRAY data block.
8154	The data block has the incorrect data type.
8282	The value at the INDEX parameter is outside the limits of the ARRAY.
8350	The data type VARIANT at parameter VALUE provides the value "0".
8352	Code generation error
8353	There are two possible causes of the error: • The size of the VALUE parameter does not match the element length in the ARRAY data block. • The two tags are not in a memory area with optimized access. For additional information on the memory area access types, refer to: Basics of block access (Page 55)
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

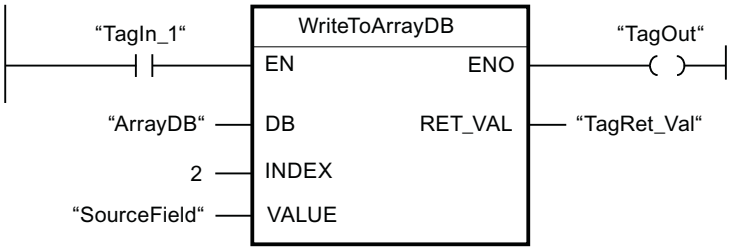

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type Array [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	SourceField	The "SourceField" operand is a global tag of the data type INT.

If the "TagIn1" operand returns signal state "1", the instruction is executed. The value of the "SourceField" operand is written to the 2nd element of the ARRAY DB. If the instruction is executed without errors, the enable output ENO has the signal state TRUE and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Example of the use of ARRAY data blocks (Page 44)

Basic information on VARIANT (Page 324)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

ReadFromArrayDBL: Read from ARRAY data block in load memory (S7-1500)

Description

The instruction "Read from ARRAY data block in load memory" is used to read the element ti which the index references from a data block of the ARRAY DB block type in the load memory and write it to the target range.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

Enable output ENO returns the signal state FALSE if one of the following conditions applies:

- Enable input EN returns the signal state FALSE.
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Read from ARRAY data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Begin with the reading of the ARRAY DB
DB [1]	Input	DB_ANY	I, Q, M, D, L	ARRAY data block that is read.
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB that is read. The specification can be a constant, a global tag or an indexed value.
VALUE [1]	InOut	VARIANT	D (element of a global data block) L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Pointer to the DB in the work memory that is to be read and the value of which is to be written. No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the ERROR parameter.
[1] The data blocks must be created with the "Optimized" block property.				

Parameter ERROR

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.

Error code* (W#16#...)	Explanation
8382	The value at the INDEX parameter is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL" and "WRIT_DBL" in the corresponding descriptions of the instructions.

Example

The following example shows how the instruction works:

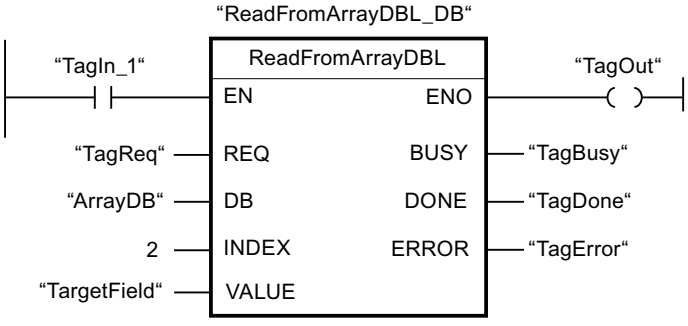

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type ARRAY [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	TargetField	The "TargetField" operand is a global tag of the data type INT.
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq" operand, the instruction is executed. The 2nd element is read from "ArrayDB" and output at the "VALUE" parameter. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. If the instruction is executed without errors, the enable output ENO has the signal state TRUE and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand has the signal state TRUE.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

READ_DBL: Read from data block in the load memory (Page 2714)

WRIT_DBL: Write to data block in the load memory (Page 2718)

Example of the use of ARRAY data blocks (Page 44)

Basic information on VARIANT (Page 324)

Evaluating errors with output parameter RET_VAL (Page 162)

Difference between synchronous and asynchronous instructions (Page 596)

Basic information on LAD (Page 7121)

WriteToArrayDBL: Write to ARRAY data block in load memory (S7-1500)

Description

The instruction "Write to ARRAY data block in load memory" is used to write the element to which the index references to a data block of the ARRAY DB block type in load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Write to ARRAY data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Start writing to the array DB
DB [1]	Input	DB_ANY	I, Q, M, D, L	ARRAY data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written. The specification can be a constant, a global tag or an indexed value.
VALUE [1]	Input	VARIANT	D (element of a global data block) L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Pointer to the DB in the work memory that is to be read and the value of which is to be written. No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the ERROR parameter.
[1] The data blocks must be created with the "Optimized" block property.				

Parameter ERROR

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8234	The data block is write protected.
8235	The data block is not an ARRAY DB.

Error code* (W#16#...)	Explanation
8254	The data block has the incorrect data type.
8382	The value at the INDEX parameter is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL" and "WRIT_DBL" in the corresponding descriptions of the instructions.

Example

The following example shows how the instruction works:

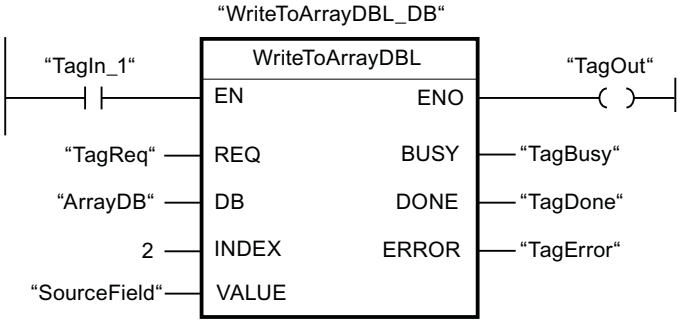

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type ARRAY [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	SourceField	The "SourceField" operand is a global tag of the data type INT.
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq" operand, the instruction is executed. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written to the 2nd element of "ArrayDB". If the instruction is executed without errors, the enable output ENO has the signal state TRUE and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand has the signal state TRUE.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

READ_DBL: Read from data block in the load memory (Page 2714)

WRIT_DBL: Write to data block in the load memory (Page 2718)

Example of the use of ARRAY data blocks (Page 44)

Basic information on VARIANT (Page 324)

Evaluating errors with output parameter RET_VAL (Page 162)

Difference between synchronous and asynchronous instructions (Page 596)

Basic information on LAD (Page 7121)

VARIANT

VariantGet: Read out VARIANT tag value

Description

You can use the "Read out VARIANT tag value" instruction to read the value of the tag to which the VARIANT at the SRC parameter points and write it in the tag at the DST parameter.

The SRC parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the DST parameter.

The data type of the tag at the DST parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction. For additional information, refer to "See also".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameter

The following table shows the parameters of the instruction "Read out VARIANT tag value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
SRC	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		Tag to be read
DST	Output	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

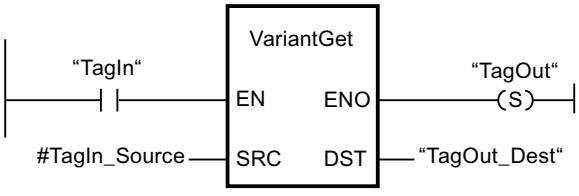

If the "TagIn" operand has signal state "1", the instruction is executed. The value of the tag to which the VARIANT at the "#TagIn_Source" operand points, is read and written to the "TagOut_Dest" operand.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

MOVE_BLK_VARIANT: Move block (Page 778)

Basic information on LAD (Page 7121)

VariantPut: Write VARIANT tag value

Description

You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter to the memory at the DST parameter to which the VARIANT points.

The DST parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT instruction: Move block" instruction. For additional information, refer to "See also".

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameters

The following table shows the parameters of the instruction "Write VARIANT tag value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
SRC	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Tag to be read
DST	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

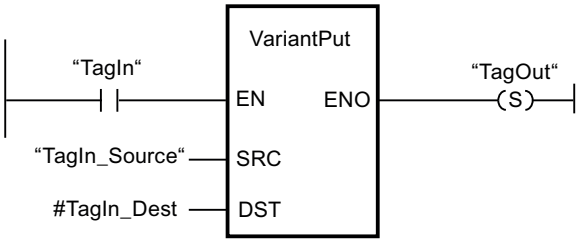

If the "TagIn" operand returns signal state "1", the instruction is executed. The value of the "TagIn_Source" operand is written to the tag to which the VARIANT at the #TagIn_Dest operand points.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

MOVE_BLK_VARIANT: Move block (Page 778)

Basic information on LAD (Page 7121)

CountOfElements: Get number of ARRAY elements

Description

You use the "Get number of ARRAY elements" instruction to query how many ARRAY elements a tag to which VARIANT points has.

If it is a single-dimensional ARRAY, the number of ARRAY elements is output as the result. (The difference between the high and low limit +1). If it is a multi-dimensional ARRAY, the number of all dimensions is output as the result.

If you want to query the elements of an ARRAY DB, you should use the instructions "ReadFromArrayDB" or "WriteFromArrayDB", because here a more precise error evaluation for the number of elements is possible.

Note

Instances

The VARIANT pointer can point to no instance and therefore also to no multi-instance or an ARRAY of multi-instances.

Note

ARRAY within a data block

If you want to query the number of the elements of an ARRAY which is located in a data block, the block attribute "Data block write-protected in the device" must not be activated in this block. Otherwise, the RET_VAL parameter returns the result "0", irrespective of how many elements the ARRAY contains.

The result is also "0" if the VARIANT tag is not an ARRAY.

If the VARIANT points to an ARRAY of BOOL, the fill elements are included in the count. (For example, 8 is returned for an ARRAY[0..1] of BOOL).

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The VARIANT tag is not an ARRAY. (The result is "0".)

Parameters

The following table shows the parameters of the "Get number of ARRAY elements" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Tag to be queried
RET_VAL	Output	UDINT	I, Q, M, D, L	Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

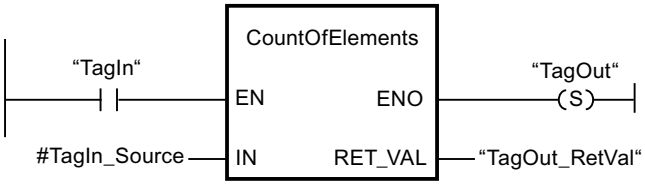

If the "TagIn" operand returns signal state "1", the instruction is executed. The number of ARRAY elements of the tag to which the VARIANT at the #TagIn_Source operand points is read and output at the "TagOut_RetVal" operand.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

ARRAY[*]

LOWER_BOUND: Read out low ARRAY limit

Description

You can declare tags of the data type ARRAY[*] in the block interface of a function block or function. For these local tags, you can read out the limits of the ARRAY. You will need to specify the required dimension at the DIM parameter.

The "Read out ARRAY low limit" instruction is used to read out the variable low limit of the ARRAY.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The dimension specified at input DIM does not exist.

Note

Availability of the instruction

The instruction is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

Parameters

The following table shows the parameters of the instruction "Read out ARRAY low limit":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
ARR	Input	ARRAY[*]	FB => Section InOut FC => Sections Input and InOut	ARRAY of which the variable low limit is to be read.
DIM	Input	UDINT	I, Q, M, D, L, P or constant	Dimension of the ARRAY of which the variable low limit is to be read.
OUT	Output	DINT	I, Q, M, D, L, P	Result

Example

The following example shows how the instruction works:

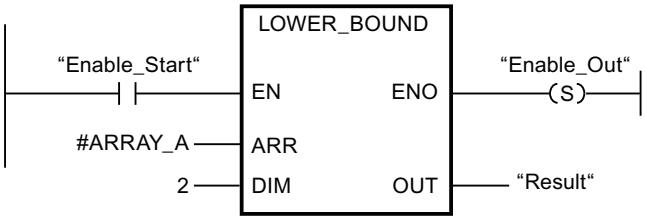

If the "Enable_Start" operand returns signal state "1", the instruction is executed. It reads out the variable low limit of the ARRAY #ARRAY_A from the second dimension. If the instruction is executed without errors, the result is written to operand "Enable_Out" and operand "Result" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on ARRAY (Page 283)

Basic information on LAD (Page 7121)

UPPER_BOUND: Read out high ARRAY limit

Description

You can declare tags of the data type ARRAY[*] in the block interface of a function block or function. For these local tags, you can read out the limits of the ARRAY. You will need to specify the required dimension at the DIM parameter.

The instruction "Read out ARRAY high limit" is used to read out the variable high limit of the ARRAY.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The dimension specified at input DIM does not exist.

Note

Availability of the instruction

The instruction is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

Parameters

The following table shows the parameters of the instruction "Read out ARRAY high limit":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
ARR	Input	ARRAY[*]	FB => Section In-Out FC => Sections Input and InOut	ARRAY of which the variable high limit is to be read.
DIM	Input	UDINT	I, Q, M, D, L, P or constant	Dimension of the ARRAY of which the variable high limit is to be read.
OUT	Output	DINT	I, Q, M, D, L, P	Result

Example

The following example shows how the instruction works:

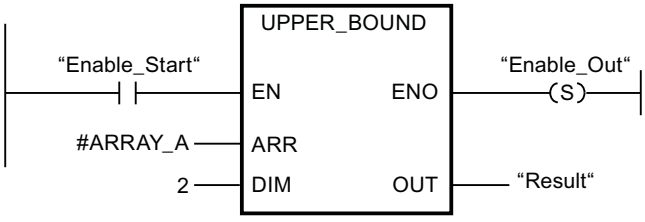

If the "Enable_Start" operand returns signal state "1", the instruction is executed. It reads out the variable high limit of the ARRAY #ARRAY_A from the second dimension. If the instruction is executed without errors, the result is written to operand "Enable_Out" and operand "Result" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on ARRAY (Page 283)

Basic information on LAD (Page 7121)

Legacy (S7-1500)

FieldRead: Read field (S7-1500)

Description

You can use the "Read field" instruction to read out a specific component from the field specified at input MEMBER and transfer its content to the tag at output VALUE. You specify the index of the field component to be read at input INDEX. Specify the first component of the field from which reading is to occur at input MEMBER.

The data types of the field component at parameter MEMBER, the index and the tags at parameter VALUE must correspond to the data type of the instruction "Read field" because implicit conversion is not possible.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The field component specified at input INDEX is not defined in the field specified at output MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Read field":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
INDEX	Input	DINT	DINT	I, Q, M, D, L, P, or constant	Index of field component whose content is read out

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
MEMBER	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR and WCHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD, CHAR and WCHAR as components of an ARRAY tag	D, L	First component of the field from which reading occurs.
VALUE	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD CHAR, WCHAR	I, Q, M, D, L, P	Operand to which the content of the field component is transferred

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

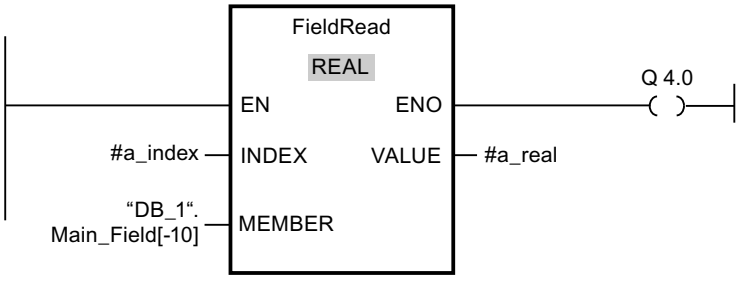

The following table shows how the instruction works using specific operand values:

Parameter	Tag	Value
INDEX	a_index	4
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the data block "DB_1"
VALUE	a_real	Component with index 4 of the "Main_Field[-10..10] of REAL" field

The field component with index 4 is read from the "Main_Field[-10...10] of REAL" field and written to the "a_real" tag. The field component to be read is specified by the value at input INDEX.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

FieldWrite: Write field (S7-1500)

Description

The "Write field" instruction is used to transfer the content of the tag at the VALUE input to a specific component of the field at the MEMBER output. You can use the value at the INDEX input to specify the index of the field component that is described. At the MEMBER output, enter the first component of the field which is to be written to.

The data types of the field component at parameter MEMBER, the index and the tags at parameter VALUE must correspond to the data type of the instruction "Read field" because implicit conversion is not possible.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The field component specified at the input INDEX is not defined in the field specified at the output MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Write field":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
INDEX	Input	DINT	DINT	I, Q, M, D, L, P, or constant	Index of the field component that is being written with the content of VALUE.

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD, CHAR, WCHAR	I, Q, M, D, L, P, or constant	Operand whose content is copied.
MEMBER	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR and WCHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD, CHAR and WCHAR as components of an ARRAY tag	D, L	First component of the field to which the content of VALUE is written.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

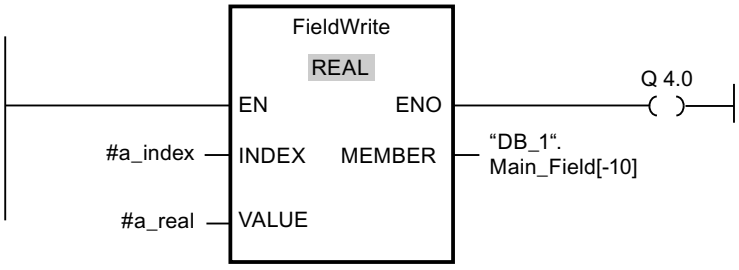

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
INDEX	a_index	4
VALUE	a_real	10.54
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the data block "DB_1"

The value "10.54" of the "a_real" tag is written to the field component with index 4 of the "Main_Field[-10..10] of REAL" field. The index of the field component to which the content of the tag "a_real" is transferred is specified by the value at the input INDEX.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

BLKMOV: Move block (S7-1500)

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can use this tag in memory areas "with optimized block access".

The following figure shows the principle of the move operation:

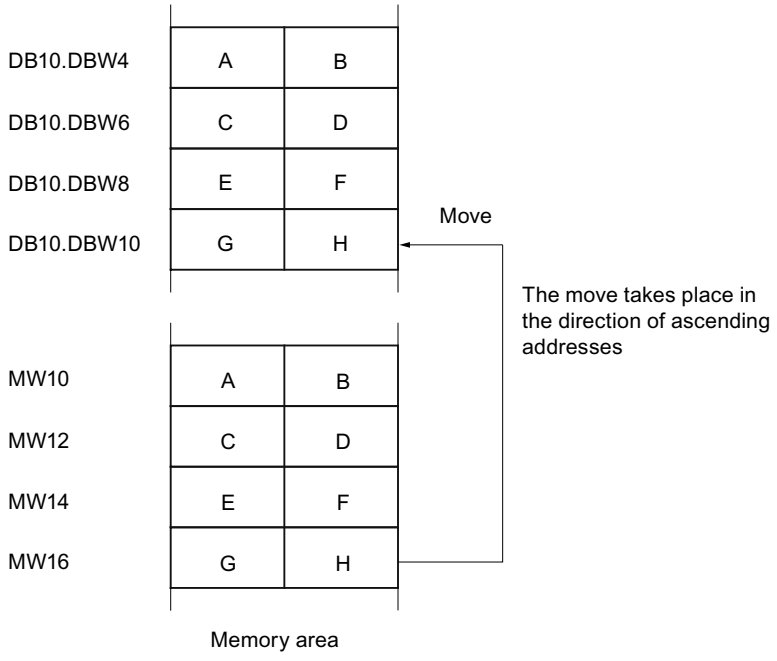

Consistency of source and destination data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1)]	VARIANT	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

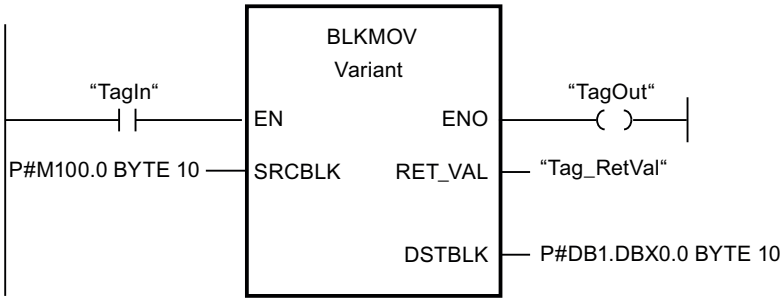

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 883)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

UBLKMOV: Move block uninterruptible (S7-1500)

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can use this tag in memory areas "with optimized block access".

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a destination or source area defined as a formal parameter is less than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1)]	VARIANT	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

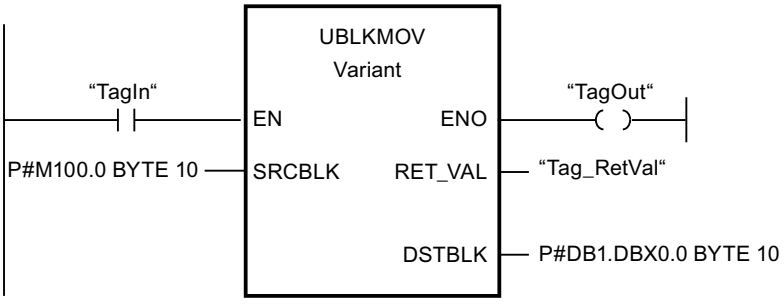

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 883)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

FILL: Fill block (S7-1500)

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can use this tag in memory areas "with optimized block access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK instruction. Fill block" instruction.

The following figure shows the principle of the move operation:

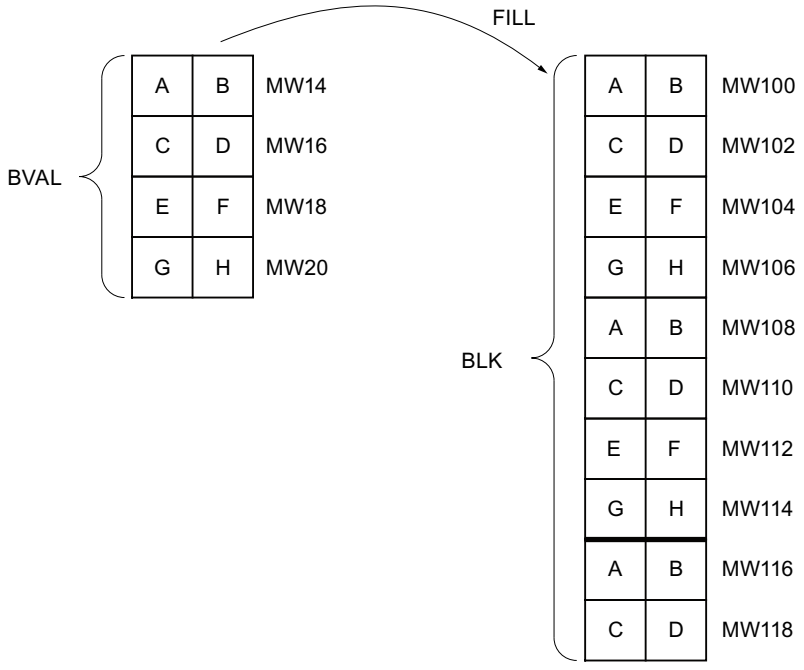

Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory

- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the input parameter BVAL, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
BVAL	Input	VARIANT	I, Q, M, D, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
BLK	Output [1)]	VARIANT	I, Q, M, D, L, P	Specification of the memory area that will be filled with the content of the source area.
1) The BLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the BVAL parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the BLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

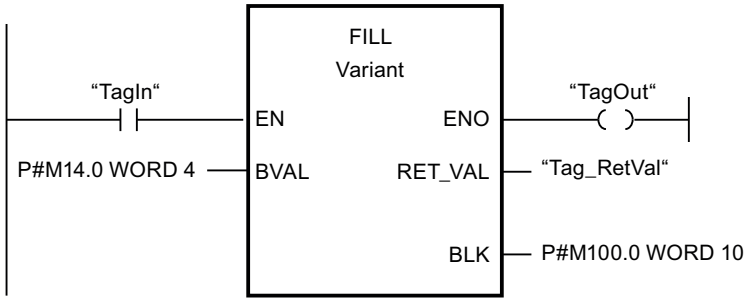

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 883)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types selected in the instruction box. The converted value is output at the OUT output.

For information on possible conversions, refer to the "Explicit conversion" section at "See also".

The enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- Errors, such as an overflow, occur during execution.

Conversion options for bit strings

BYTE and WORD bit strings cannot be selected in the instruction box. However, it is possible to specify an operand of data type DWORD or LWORD at a parameter of the instruction if the lengths of the input and output operands match. The operand is then interpreted by the data type of a bit string according to the data type of the input or output parameter and is implicitly converted. The data type DWORD is for example interpreted as DINT/UDINT and LWORD as LINT/ULINT. These conversion options are also available to you when "IEC check" is enabled.

Note

For CPUs of the S7-1500 series: The data types DWORD and LWORD can only be converted to or from data type REAL or LREAL.

During the conversion, the bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If no errors occur during the conversion, the signal state of enable output ENO = 1; if an error occurs during processing, the signal state of the enable output ENO = 0.

Parameters

The following table shows the parameters of the "Convert value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings, integers, floating-point numbers, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted.
OUT	Output	Bit strings, integers, floating-point numbers, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

Examples

The following example shows the conversion of an integer (16 bit) to another integer (32 bit):

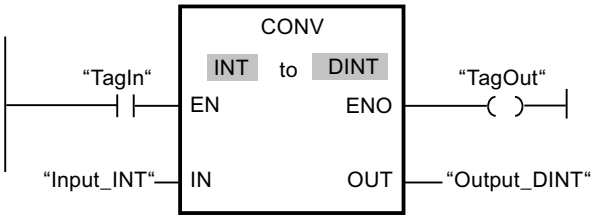

The following example shows the conversion of a byte (8 bit) to the integer SINT (8 bit):

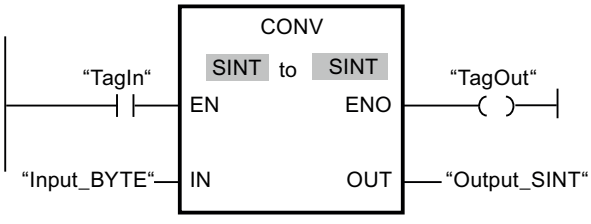

The following example shows the conversion of a byte (8 bit) to the unsigned integer USINT (8 bit):

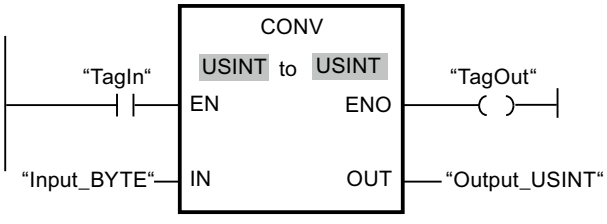

The conversions are possible, because the length of the two operands is the same.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Data type conversion for S7-1200 (Page 489)

Basic information on LAD (Page 7121)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to an integer of data type DINT. If the input value is exactly between an even and odd number, the even number is selected. The result of the instruction is sent to the OUT output and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value to be rounded.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result of rounding

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

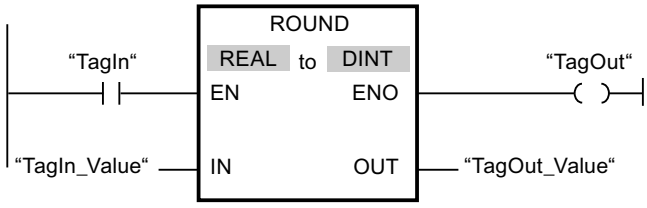

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

CEIL: Generate next higher integer from floating-point number

Description

You can use the "Generate next higher integer from floating-point number" instruction to round the value at input IN to the next higher integer. The instruction interprets the value at the IN input as a floating-point number and converts this number to the next higher integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be greater than or equal to the input value.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Generate next higher integer from floating-point number" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next higher integer

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

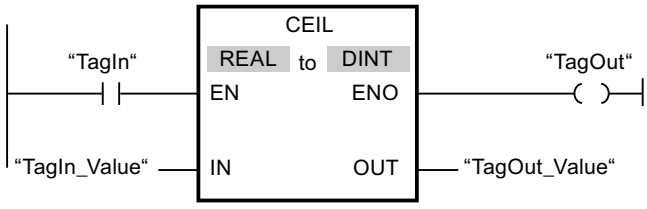

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at the "TagIn_Value" input is rounded to the next higher integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

FLOOR: Generate next lower integer from floating-point number

Description

You can use the "Generate next lower integer from floating-point number" instruction to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be less than or equal to the input value.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Generate next lower integer from floating-point number" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next lower integer

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

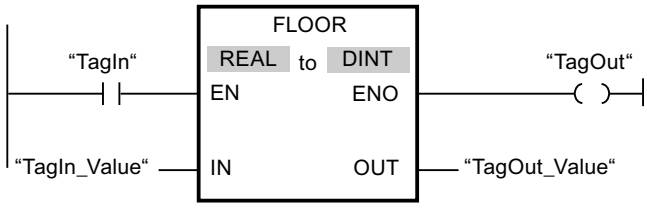

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at the IN input. The value at the IN input is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to the OUT output without decimal places.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Truncate numerical value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Integer part of the input value

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

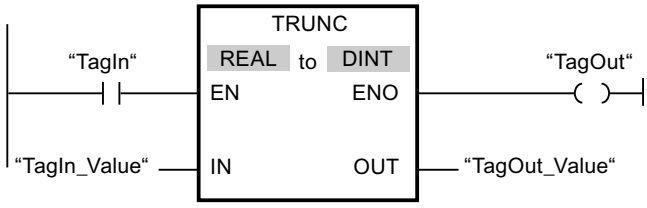

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The integer part of the floating-point number at the "TagIn_Value" input is converted to an integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

SCALE_X: Scale

Description

You can use the "Scale" instruction to scale the value at the VALUE input by mapping it to a specified value range. When the "Scale" instruction is executed, the floating-point value at the VALUE input is scaled to the value range that was defined by the MIN and MAX parameters. The result of the scaling is an integer, which is stored in the OUT output.

The following figure shows an example of how values can be scaled:

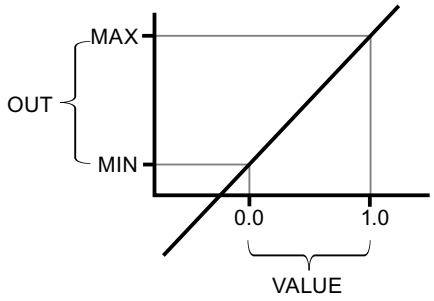

The "Scale" instruction works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the MIN input is greater than or equal to the value at the MAX input.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value at the VALUE input is NaN (Not a Number = result of an invalid arithmetic operation).

Note

For more information on the converting of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range

Parameter	Declaration	Data type	Memory area	Description
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result of scaling

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

You can find additional information on valid data types under "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

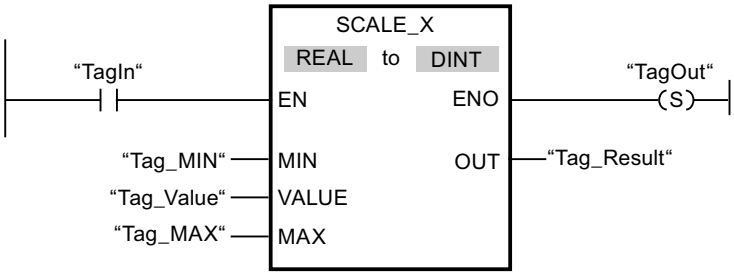

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5
MAX	Tag_MAX	30
OUT	Tag_Result	20

If the "TagIn" operand has signal state "1", the instruction is executed. The value at the "Tag_Value" input is scaled to the range of values defined by the values at the "Tag_MIN" and "Tag_MAX" inputs. The result is stored in the "Tag_Result" output. If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

NORM_X: Normalize

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. The result at the OUT output is calculated and stored as a floating-point number depending on the location of the value to be normalized within this value range. If the value to be normalized is equal to the value at the MIN input, the OUT output has the value "0.0". If the value to be normalized is equal to the value at input MAX, output OUT returns the value "1.0".

The following figure shows an example of how values can be normalized:

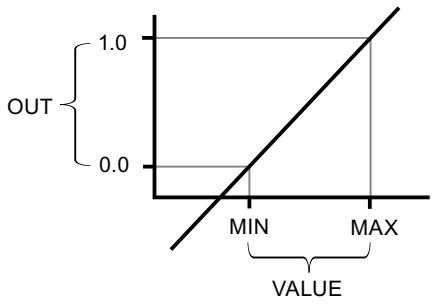

The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the MIN input is greater than or equal to the value at the MAX input.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- The value at the VALUE input is NaN (result of an invalid arithmetic operation).

Note

For more information on the converting of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the "Normalize" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MIN [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Value to be normalized.
MAX [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Floating-point numbers	I, Q, M, D, L	Result of the normalization
[1] If you use constants in these three parameters, you only need to declare one of them.				

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

You can find additional information on valid data types under "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

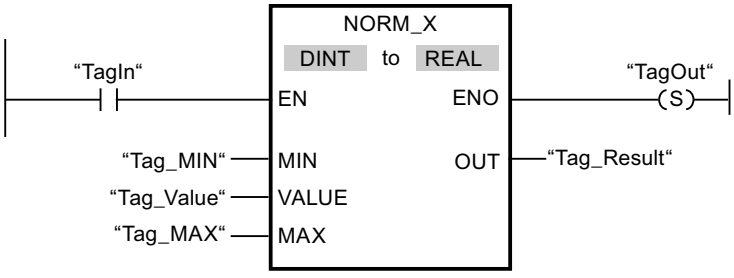

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	20
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

If the "TagIn" operand has signal state "1", the instruction is executed. The value at the "Tag_Value" input is mapped to the range of values that were defined by the values at the "Tag_MIN" and "Tag_MAX" inputs. The tag value at the "Tag_Value" input is normalized to the defined value range. The result is stored as a floating-point number in the "Tag_Result" output. If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

Legacy (S7-1500)

SCALE: Scale (S7-1500)

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number, which can be scaled in physical units between a low limit value and a high limit value. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output on the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0,0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameter

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	INT	I, Q, M, D, L, P, or constant	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P, or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P, or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

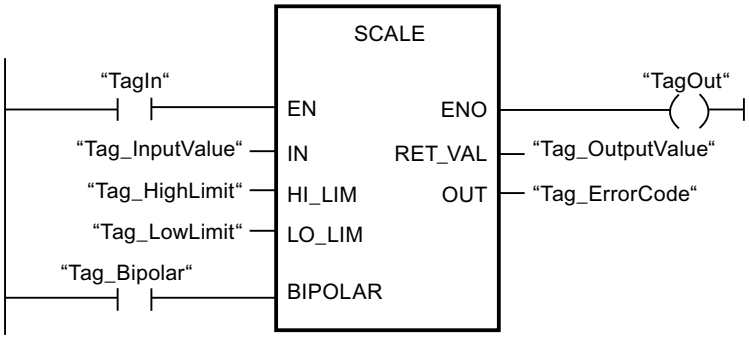

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 883)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

UNSCALE: Unscale (S7-1500)

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at parameter IN is not within the limits defined by HI_LIM and LO_LIM, an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled inversely proportionate to the input value.

Parameter

The following table shows the parameters of the "Unscale" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Input	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally"
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

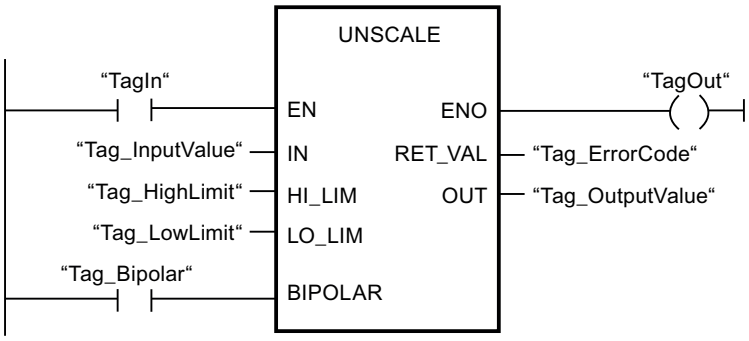

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

GET_ERR_ID: Get error ID locally (Page 883)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

Program control operations

---(JMP): Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil is permitted within a network.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:

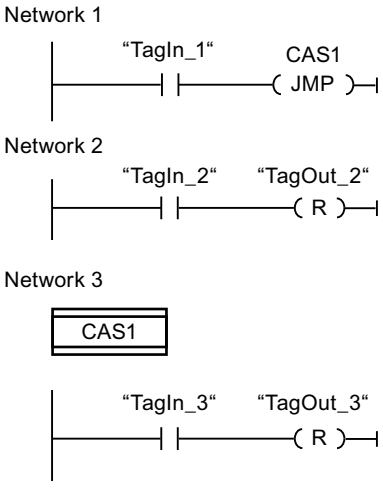

If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has signal state "1", the "TagOut_3" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(JMPN): Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil is permitted within a network.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of logic operation at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:

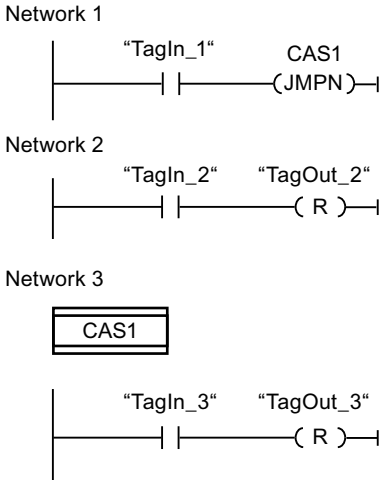

If operand "TagIn_1" has the signal state "0", the "Jump if RLO = 0" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has signal state "1", the "TagOut_3" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

LABEL: Jump label

Description

You can use a jump label to identify a destination network, in which the program execution should resume when a jump is executed.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block. You can declare up to 32 jump labels when you use a CPU S7-1200 and a maximum of 256 jump labels when you use a CPU S7-1500.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

Adhere to the following grammatical rules for jump labels:

- Letters (a to z, A to Z)
- A combination of letters and numbers. Note the order, i.e. first the letters and then the letters (a - z, A - Z, 0 - 9).
- Special characters or a combination of letters and numbers in reverse order, i.e. first the numbers and then the letters (0 - 9, a - z, A - Z) must not be used.

Example

The following example shows how the instruction works:

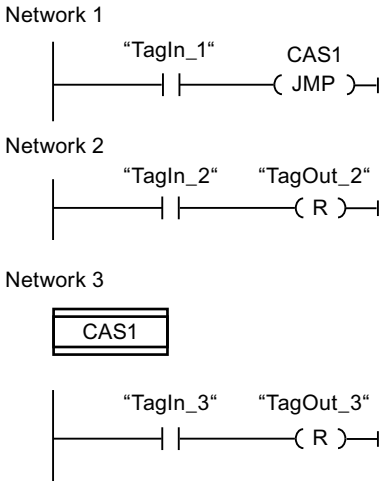

If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has signal state "1", the "TagOut_3" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

JMP_LIST: Define jump list

Description

You can use the "Define jump list" instruction to define several conditional jumps and continue the program execution in a specific network depending on the value of the K parameter.

You define the jumps with jump labels (LABEL), which you specify at the outputs of the instruction box. The number of outputs can be expanded in the instruction box. You can declare up to 32 outputs when you use a CPU S7-1200 and a maximum of 256 outputs when you use a CPU S7-1500.

The numbering of the outputs begins with the value "0" and is continued in ascending order with each new output. Only jump labels can be specified at the outputs of the instruction. Instructions or operands cannot be specified.

The value of the K parameter specifies the number of the output and thus the jump label where the program execution is to be resumed. If the value in the K parameter is greater than the number of available outputs, the program execution is resumed in the next network of the block.

The "Define jump list" instruction is only executed if the signal state is "1" at the EN enable input.

Parameters

The following table shows the parameters of the "Define jump list" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
K	Input	UINT	I, Q, M, D, L or constant	Specifies the number of the output and thus the jump that is to be made.
DEST0	-	-	-	First jump label
DEST1	-	-	-	Second jump label
DESTn	-	-	-	Optional jump labels

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

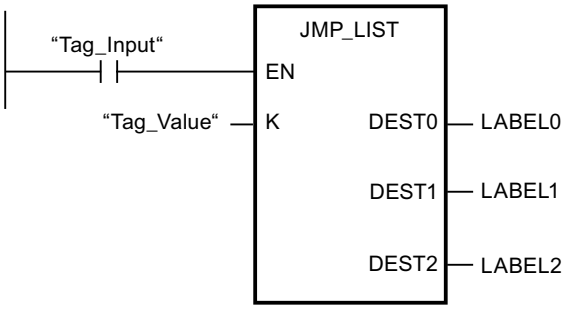

The following table shows how the instruction works using specific operand values:

Parameters	Operand/jump label	Value
K	Tag_Value	1
Dest0	LABEL0	Jump to the network that is identified with the jump label "LABEL0".
Dest1	LABEL1	Jump to the network that is identified with the jump label "LABEL1".
Dest2	LABEL2	Jump to the network that is identified with the jump label "LABEL2".

If operand "Tag_Input" has the signal state "1", the "Define jump list" instruction is executed. The execution of the program is continued according to the value of the operand "Tag_Value" in the network that is identified with the jump label "LABEL1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

SWITCH: Jump distributor

Description

You can use the "Jump distributor" instruction to define multiple program jumps to be executed depending on the result of one or more comparison instructions.

You specify the value to be compared in the K parameter. This value is compared with the values that are provided by the various inputs. You can select the comparison method for each individual input. The availability of the various comparison instructions depends on the data type of the instruction.

The following table shows the comparison instructions that are available depending on the selected data type:

Data type		Instruction	Syntax
S7-1200	S7-1500		
Bit strings	Bit strings	Equal	==
		Not equal	<>
Integers, floating-point numbers, TIME, DATE, TOD	Integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	Equal	==
		Not equal	<>
		Greater or equal	>=
		Less or equal	<=
		Greater than	>
		Less than	<

You can select the data type of the instruction from the "???" drop-down list of the instruction box. If you select a comparison instruction and the data type of the instruction is not yet defined, then the "???" drop-down list will only list those data types that are permitted for the selected comparison instruction.

Execution of the instruction begins with the first comparison and runs until a comparison condition is met. If a comparison condition is met, the subsequent comparison conditions are not considered. If none of the specified comparison conditions are fulfilled, the jump is executed at output ELSE. If no program jump is defined at the ELSE output, execution of the program continues in the next network.

The number of outputs can be expanded in the instruction box. The numbering of the outputs begins with the value "0" and is continued in ascending order with each new output. Specify jump labels (LABEL) at the outputs of the instruction. Instructions or operands cannot be specified at the outputs of the instruction.

An input is automatically inserted to each additional output. The jump programmed at an output is executed if the comparison condition of the corresponding input is fulfilled.

Parameters

The following table shows the parameters of the instruction "Jump distributor":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
K	Input	UINT	UINT	I, Q, M, D, L or constant	Specifies the value to be compared.
<Comparison values>	Input	Bit strings, integers, floating-point numbers, TIME, DATE, TOD	Bit strings, integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	I, Q, M, D, L or constant	Input values with which the value of the parameter K is compared.

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
DEST0	-	-	-	-	First jump label
DEST1	-	-	-	-	Second jump label
DEST(n)	-	-	-	-	Optional jump labels: • S7-1200: n = 2 to 32 • S7-1500: n = 2 to 256
ELSE	-	-	-	-	Program jump which is executed if none of the comparison conditions are fulfilled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

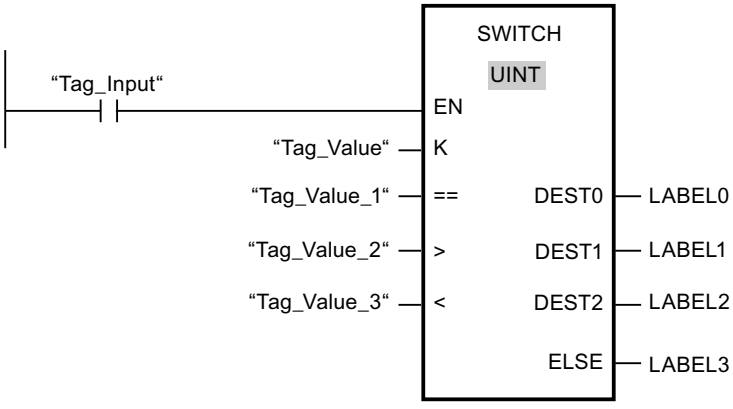

The following table shows how the instruction works using specific operand values:

Parameters	Operand/jump label	Value
K	Tag_Value	23
==	Tag_Value_1	20
>	Tag_Value_2	21
<	Tag_Value_3	19
DEST0	LABEL0	Jump to jump label "LABEL0", if the value of parameter K is equal to 20.
DEST1	LABEL1	Jump to jump label "LABEL1", if the value of parameter K is greater than 21.

Parameters	Operand/jump label	Value
DEST2	LABEL2	Jump to jump label "LABEL2", if the value of parameter K is less than 19.
ELSE	LABEL3	Jump to jump label "LABEL3", if the none of the comparison conditions are fulfilled.

If the operand "Tag_Input" changes to signal state "1", the instruction "Jump distributor" is executed. The execution of the program is continued in the network that is identified with the jump label "LABEL1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

--(RET): Return

Description

You can use the "Return" instruction to stop the execution of a program block conditionally or absolutely. The signal state of the return value (operand) is mapped in the enable output ENO of the calling program block when the program block is exited.

There are three ways that the execution of a program block can be ended:

Termination of the program block	Description
Without a call of the instruction	The program block is exited after the execution of the last network. The enable output ENO of the calling program block is set to signal state "1".
Call of the instruction with preceding logic operation (see example)	If the preceding logic operation is fulfilled, the program execution in the currently called program block is ended and the program block is exited. (Conditional program block end) The program execution is continued in the calling program block (e.g. an OB) after the program block call. The enable output ENO of the calling program block corresponds to the operand.
Call of the instruction without preceding logic or the instruction is directly connected to the left power rail.	The program block is exited unconditionally. (Absolute program block end) The enable output ENO of the calling program block corresponds to the operand.

If an organization block (OB) is ended, another program block is selected by the execution level system and started or further executed.

- If the OB program cycle is ended, it is restarted.
- If an OB is ended that has interrupted another program block (e.g. an interrupt OB), then the interrupted program block (e.g. program cycle OB) is further executed.

Note

RET instruction in relation with JMP and JMPN

If a network already contains the "JMP: Jump if RLO = 1" or "JMPN: Jump if RLO = 0 instruction, the RET: Return" instruction must not be used. Only one jumping coil may be used in a network

Possible settings of the return value (operand)

The return value of the instruction can assume the following values:

- Ret (RLO, corresponds to the result of logic operation RLO. The signal state "1" is supplied to the enable output ENO of the calling program block because the RET instruction is only then executed as a conditional instruction when the condition is TRUE.)
- Ret True or Ret False (the respective value of the constant, TRUE or FALSE, is supplied to the calling program block.)
- Ret Value (the value of the Boolean tag <Operand> is supplied to the calling program block.)

To set the return value of the instruction, click the small yellow triangle of the instruction and select the desired value in the drop-down list.

The following table shows the status of the calling program block when the instruction is programmed in a network within the called program block:

RLO	Return value	ENO of the calling program block
1	RLO	1
	TRUE	1
	FALSE	0
	<Operand> Boolean tag with possible memory areas I, Q, M, D, L, T, and C	<Operand>
0	RLO	The program execution is continued in the next network of the called program block.
	TRUE	
	FALSE	
	<Operand>	

Example

The following example shows how the instruction works:

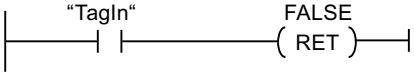

If the "TagIn" operand has signal state "1", the instruction is executed. Program execution is ended in the called program block and continues in the calling program block. The enable output ENO of the calling program block is reset to signal state "0".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Example of the use of the EN/ENO mechanism in LAD (Page 146)

Basic information on LAD (Page 7121)

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether configured passwords may be legitimated or not for the CPU. You can prevent legitimated connections, even when the correct password is known.

When you call the command and the REQ parameter has the signal state "0", only the currently set state is displayed at the output parameters. If changes are made to the input parameters, these changes are not transferred to the output parameters.

When the instruction is called and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD).

- If signal state "0" is present, authorization via password is not allowed.
- If signal state "1" is present, the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited, except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Non-configured passwords must have the TRUE signal state at the input and return the TRUE signal state at the output. The fail-safe password can only be configured for an F-CPU and, therefore, must always be interconnected with theTRUE signal state in a standard CPU. If the instruction return an error, the call remains without effect, i.e. the previous lock remains in force.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the operating mode switch to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Inserting an empty memory card (transfer card or program card) into S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access to the HMI system, if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing accidental lockout out with an S7-1500 CPU

The same setting can be made on the front panel of the S7-1500 CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode selector to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing accidental lockout with an S7-1200 CPU

The protection for POWER OFF - POWER ON is disabled because a S7-1200 CPU has no operating mode switch. As a result it is possible and advisable to use certain program sequences within your program to prevent an accidental lockout.

To do this, program a time control using either a cyclic interrupt OB or a timer in the main OB (OB 1). This provides you with the option to call the "Limit and enable password legitimation" instruction again without delay in the corresponding OB (e.g. OB 1 or OB 35) after a transition from POWER OFF - POWER ON. Call the instruction in the startup OB (OB 100) in order to keep as short as possible the time window in which the application is not active and thus ensure that no restrictions exist in the password legitimation. This procedure offers you the largest possible protection against unauthorized access.

If an unintended lockout occurs you can skip the call in the startup OB (for example, by querying an input parameter) and have the set time (for example 10 seconds to 1 minute), to establish a connection to the CPU before the lock becomes active again.

If you have set no timer in your program code and are locked out, insert an empty transfer card or empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then load the user program again from STEP 7 in the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

WARNING

Inserting empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

After you remove the transfer card the content of the transfer card is in the internal load memory. Make sure at this point that the card contains no program.

WARNING

Inserting empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. The internal load memory is empty after the removal of the formerly empty program card.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200-CPU: the lock is disabled and the instruction must be called again in the program (in the startup OB, for example). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Parameters

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password

Parameters	Declaration	Data type	Memory area	Description
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L or constant	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L or constant	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L or constant	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported
80D0	The password for fail-safe is not configured. With standard CPUs, the signal state must be TRUE.
80D1	The read/write access is not configured

Error code* (W#16#...)	Explanation
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

PC systems

SHUT_DOWN: Shut down target system

Description

You can use the instruction "SHUT_DOWN: Shutdown target system" to shut down the PC-based automation system and restart the S7 software controller CPU 150xS or Windows on the PC-based automation system.

The instruction is available in the "Instructions" task card under Basic instructions > Program control > Runtime control.

A restart is, for example, advisable in the following situations:

- An industrial UPS (uninterruptible power supply) reports a power failure via a digital input.
- Windows stops responding or shows a "blue screen".
- Too many error OBs are called in the user program.

Parameter

The following table shows the parameters of the "Shutdown target system" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	MODE = 1: Shut down CPU 150xS and Windows The CPU goes to STOP and saves the retentive data. Then the CPU and Windows are shut down. The system must be restarted manually. MODE = 2: Restart CPU 150xS The CPU goes to STOP and saves the retentive data. Then the CPU is shut down and restarted. MODE = 3: Restart Windows The CPU remains in RUN. Windows is restarted (as of TIA Portal V14, MODE 3 is only retained for backward compatibility. We recommend using MODE 4 or MODE 5). MODE = 4: Windows is shut down and restarted correctly. The CPU remains in RUN. MODE = 5: Restart Windows (Comparable with MODE 3; Exception: MODE 5 should only be used if Windows has crashed).
COM-MENT	Input	STRING	I, Q, M, D, L	With Mode = 1, 3 and 4, you can specify a reason for the restart. The reason is output in the Windows Event Log.
Ret_Val	Return	WORD	Q, M, D, L	Ret_Val = 0: No faults Ret_Val = 8090: The value passed to MODE is not supported. Ret_Val = 8091: Windows does not respond to the call of the Shut_Down instruction (only valid for Mode 3 and 4). Ret_Val = 8092: Please contact SIMATIC Customer Support if this error occurs (only valid for Mode 3 and 4).

RE_TRIGR: Restart cycle monitoring time

Description

The "Restart cycle monitoring time" instruction is used to restart the cycle monitoring time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

The instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Calling the instruction

The following call conditions apply:

- The following applies to a CPU of the S7-1200 series:
  In firmware versions <= 2.2, you can call the instruction only in a program cycle organization block of priority 1. This corresponds to the lowest priority of all organization blocks. If the instruction is called in an organization block with a higher priority, the instruction is not executed and the result (BR bit, enable output ENO) is always "0".
  In firmware versions >= 2.2, you can call the instruction regardless of the priority in all organization blocks.
- The following applies to a CPU of the S7-1500 series:
  You can call the instruction regardless of the priority in all organization blocks.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and thus end the program execution. The effects at a transition from RUN to STOP depend on the CPU configuration.

When the result of logic operation (RLO) at the input of the instruction is "1", the CPU changes to STOP mode and the program execution is ended. The signal state at the output of the instruction is not evaluated.

When the RLO is "0" at the input of the instruction, then the instruction will not be executed.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a program block. This normally involves programming or access errors. If the system reports an error during the execution of the program block, detailed information is stored in the operand at the ERROR output for the first error to occur during the execution of the block since the last execution of the instruction.

Only operands of the "ErrorStruct" system data type can be specified at the ERROR output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the program block, the error information about the next error to occur is output by the instruction only after the first error that occurred has been remedied.

Note

Output ERROR

The ERROR output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO is only set if the enable input EN returns signal state "1" and error information is present. If one of these conditions does not apply, then the remaining program execution is not affected by the instruction.

Note

Activation of the local error handling

As soon as you insert the instruction in the program code of a program block, the local error handling is activated and default system reactions are ignored when errors occur.

Options for error handling

You can find an overview of the options for error handling here: Overview of mechanisms for error handling (Page 138)

For a detailed example of local error handling that contains several error handling options, refer to: Example of how to handle program execution errors (Page 167)

Parameters

The following table shows the parameters of the "Get error locally" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
ERROR	Output	ErrorStruct	D, L	Error information

Data type "ErrorStruct"

The "ErrorStruct" data type can be inserted in a global data block or in the block interface. You can insert the data type more then once if you assign a different name for the data structure each time. The data structure and the name of individual structure elements cannot be changed. If you save the error information in a global data block, it can also be read out by other program blocks.

The following table shows the structure of the "ErrorStruct" data type:

Structure component	Data type	Description
ERROR_ID	WORD	Error ID
FLAGS	BYTE	Shows whether the error occurred during a program block call. 16#01: Error during a program block call 16#00: No error during a program block call
REACTION	BYTE	Default reaction: 0: Ignore (write error) 1: Continue with substitute value "0" (read error) 2: Skip instruction (system error)

Structure component		Data type	Description
CODE_ADDRESS		CREF	Information about the address and type of the program block
	BLOCK_TYPE	BYTE	Program block type in which the error occurred: 1: Organization block (OB) 2: Function (FC) 3: Function block (FB)
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand
	AREA	BYTE	(C) Memory area: L: 16#40...16#7F, 16#86, 16#87, 16#8E, 16#8F, 16#C0...16#FF I: 16#81 Q: 16#82 M: 16#83 DB: 16#40, 16#84, 16#85, 16#8A, 16#8B PI: 16#01 PQ: 16#02 Technological objects: 16#04
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

The following table shows the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment

ID* (hexadecimal)	ID* (decimal)	Description
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid reference used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2577	9591	The block property "Parameter passing via registers" is not selected.
2576	9590	Error in the local data distribution
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

Example

The following example shows how the instruction works:

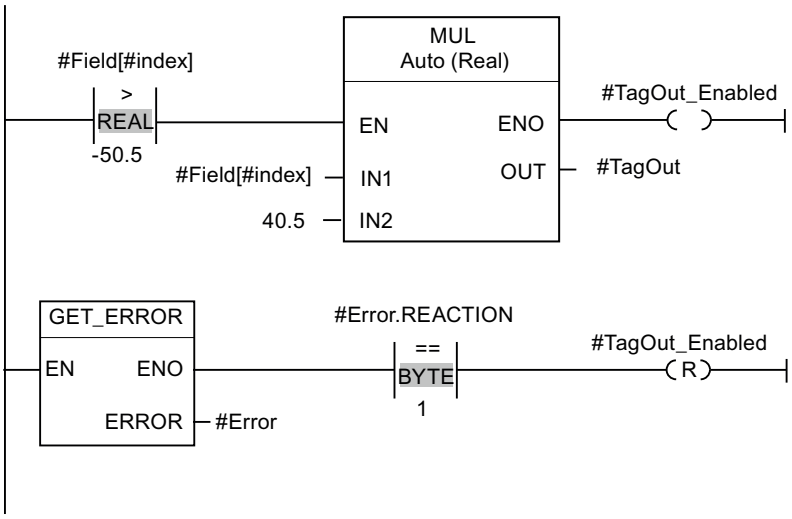

An error occurred accessing the #Field[#index] tag. The enable output ENO of the "Multiply" instruction and the #TagOut_Enabled operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error locally" instruction is evaluated with the comparison instruction "Equal". If the "#Error.REACTION" structure component has the value "1", this involves a read/access error and the #TagOut_Enabled output is reset.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

Basic information on LAD (Page 7121)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the error ID of the first error that occurred in the tag is stored at the ID output.

Only operands of the WORD data type can be specified at the ID output. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The ID output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO of the instruction "Get error ID locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Options for error handling

You can find an overview of the options for error handling here: Overview of mechanisms for error handling (Page 138)

For a detailed example of local error handling that contains several error handling options, refer to: Example of how to handle program execution errors (Page 167)

Parameters

The following table shows the parameters of the "Get error ID locally" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
ID	Output	WORD	I, Q, M, D, L	Error ID

Parameter ID

The following table shows the values that can be output at the ID parameter:

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid reference used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used

ID* (hexadecimal)	ID* (decimal)	Description
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

Example

The following example shows how the instruction works:

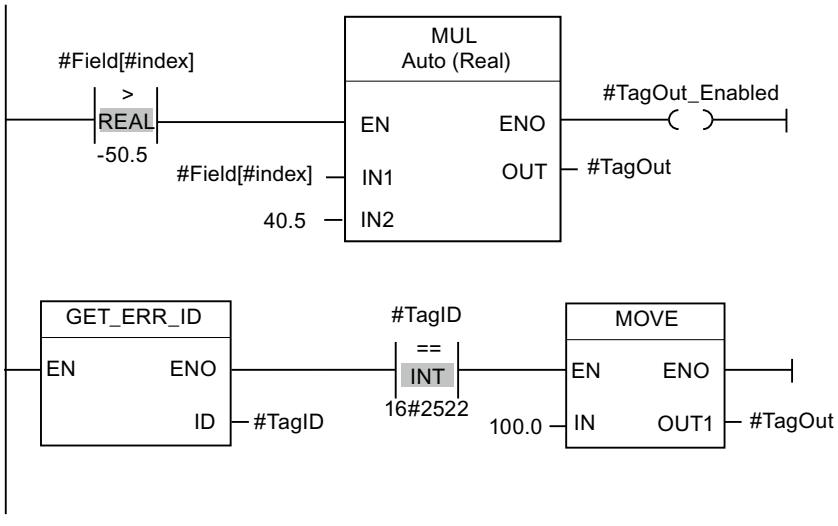

An error occurred accessing the #Field[#index] tag. The enable output ENO of the "Multiply" instruction and the #TagOut_Enabled operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error ID locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error ID locally" instruction is evaluated with the comparison instruction "Equal". If the #TagID operand returns the ID 2522, this involves a read/access error and the value "100.0" is written to the #TagOut output.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

Basic information on LAD (Page 7121)

INIT_RD: Initialize all retain data (S7-1500)

Description

You can use the "Initialize all retain data" instruction to reset the retain data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution would exceed the program cycle duration.

Parameters

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	If the input "REQ" has the signal state "1", all retain data are reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.

You can find additional information on valid data types under "See also".

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.

Error code* (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

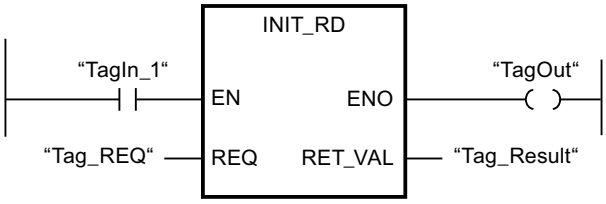

If the operands "TagIn_1" and "Tag_REQ" have signal state "1", the instruction is executed. The retain data of all data blocks, bit memories and SIMATIC timers and counters are reset If the instruction is executed without errors, the ENO enable output has the signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 883)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

WAIT: Configure time delay (S7-1500)

Description

The "Configure time delay" instruction pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to +32767 microseconds (µs). The smallest possible delay time depends on the CPU and corresponds to the execution time of the instruction.

The execution of the instruction can be interrupted by higher priority events and does not return any error information.

Note

Negative delay time

If you specify a negative delay time at parameter WT, the enable output ENO, or the RLO and the BR bit, will return the signal state FALSE. A negative delay time does not affect the CPU. The following instructions linked to enable output ENO are not executed in LAD or FBD.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

Example of the influence of the planned delay time

In the following example, you will see the influence of the delay time of the instruction "WAIT" in various scenarios.

The following figure is a schematic representation of the scenarios:

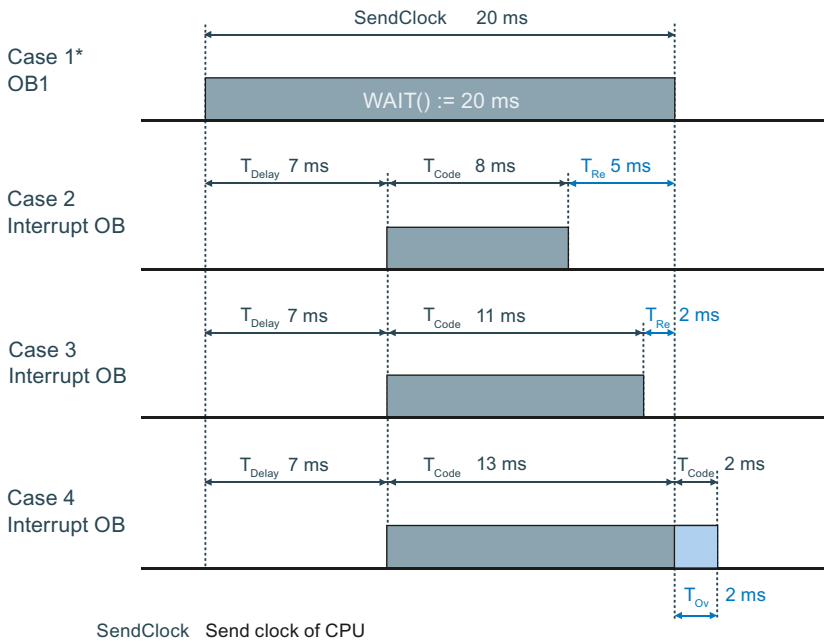

SendClock	Send clock of CPU
T_{Delay}	Delay time of the interrupt OB
T_{Code}	Time of program execution
T_{Re}	Remaining time
T_{Ov}	Excess time by which the send clock is increased

* The OB1 with the instruction "Wait" is also part of cases 2, 3 and 4 and needs to be taken into account.

Remaining time = The period between the end of the planned delay time (via "WAIT") until the end of the interrupt OB

Excess time = The period between the end of the interrupt OB and the end of the planned delay time (via "WAIT").

Case 1:

The "WAIT" instruction is called in an OB1. The "WAIT" instruction can be interrupted by higher-priority OBs or higher-priority processes (e.g. System Threads). However, the delay time of the "WAIT" instruction is neither changed nor deferred.

Cases 2 and 3:

Processing of the program in OB1 is resumed after a time delay of 20 ms. This time delay is calculated by calling the "WAIT" instruction in OB1 (see OB1 with WAIT). Within these 20 ms, an interrupt OB can run its own program code. The send clock of the CPU is not changed.

Case 4:

Processing of the program in OB1 is resumed after the higher-priority process has ended. The 20 ms delay time in OB1 has elapsed, but the higher-priority process still needs to be ended. The send clock of the CPU is increased.

Note

Order of execution of system or communication processes (System Threads)

The System Threads usually use the priority "15". There are also System Threads with a higher priority than "26", but these processes cause a lower CPU load. System Threads are not displayed in the figure.

Runtime measurement of the OB1 using the "RT_INFO" instruction:

Case 2: 20 ms - 8 ms - System Threads = <12 ms. Send clock: ~20 ms.

Case 3: 20 ms - 11 ms - System Threads - <9 ms. Send clock: ~20 ms.

Case 4: 20 ms - 15 ms - System Threads - <7 ms. Send clock: ~22 ms.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

RUNTIME: Measure program runtime

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and write the value to the IN-OUT parameter MEM. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the IN-OUT parameter MEM as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the

command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The "Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overruns, the instruction returns values <= 0.0. This can occur up to once per minute for S7-1200 CPUs with firmware version <V4.2. Ignore these runtime values.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Parameter

The following table lists the parameters of the "Measure program runtime" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MEM	InOut	LREAL	I, Q, M, D, L	The content is for internal purposes only.
RET_VAL	Output	LREAL	I, Q, M, D, L	Returns the measured runtime in seconds

Example

The following example shows how the instruction works based on the runtime calculation of a program block:

Network 1:

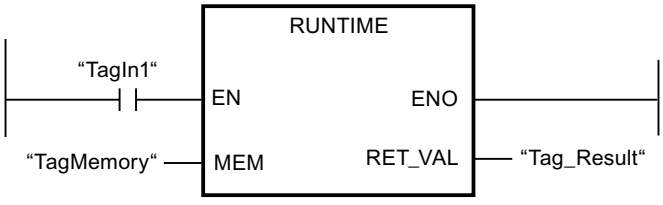

Network 2:

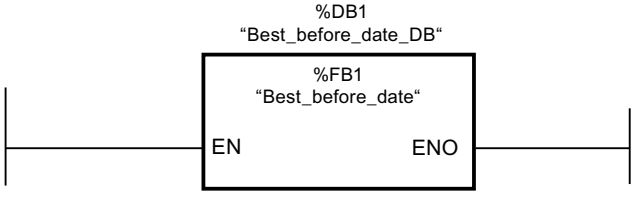

Network 3:

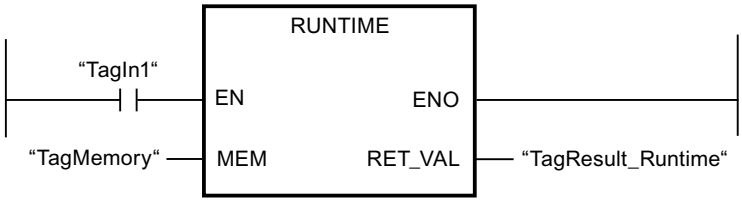

When the "TagIn1" operand in network 1 has the signal state "1", the instruction is executed. The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called in network 2.

When the FB1 program block has been executed and the "TagIn1" operand has the signal state "1", the instruction in network 3 is executed. The second call of the instruction calculates the runtime of the program block and writes the result to the output RET_VAL.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

You can find a more detailed example of how you can measure the total cycle time of a program in the Siemens Industry Online Support under the following entry ID: 87668055 (https://support.industry.siemens.com/cs/ww/en/view/87668055)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

Word logic operations

AND: AND logic operation

Description

You can use the "AND logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by AND logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are logically ANDed. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with AND logic (ANDed). The result is stored in the OUT output.

The result bit has signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "AND logic operation" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P, or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P, or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P, or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

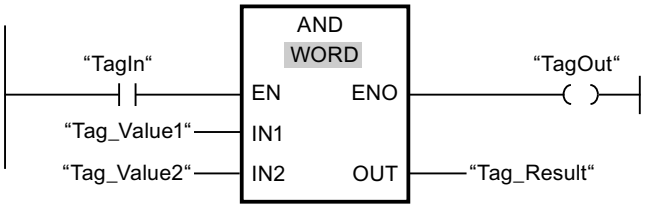

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0000 0000 0000 0101

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ANDed. The result is mapped bit-for-bit and output in operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

OR: OR logic operation

Description

You can use the "OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are combined by OR logic operation. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with OR logic (ORed). The result is stored in the OUT output.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "OR logic operation" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P, or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P, or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P, or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

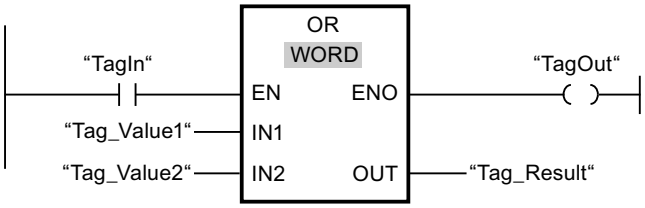

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1111

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ORed. The result is mapped bit-for-bit and output in operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

XOR: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by EXCLUSIVE OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are combined by EXCLUSIVE OR logic operation. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with EXCLUSIVE OR logic. The result is stored in the OUT output.

The result bit has signal state "1" when one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P, or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P, or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P, or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

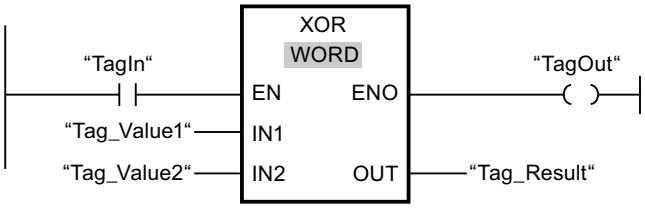

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1010

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are combined by EXCLUSIVE OR logic. The result is mapped bit-for-bit and output in operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

INVERT: Create ones complement

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits at input IN. When the instruction is processed, the value at input IN is linked to EXCLUSIVE OR by a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit number). This inverts the signal state of the individual bits that are then stored at output OUT.

Parameters

The following table shows the parameters of the instruction "Create ones complement":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Bit strings, integers	I, Q, M, D, L, P	Ones complement of the value at input IN

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

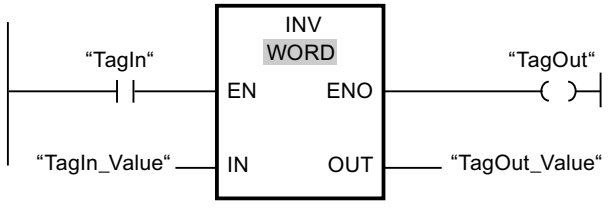

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	W#16#000F	W#16#7E
OUT	TagOut_Value	W#16#FFF0	W#16#81

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction inverts the signal state of the individual bits at input "TagIn_Value" and writes the result to output "TagOut_Value". The enable output ENO and the output "TagOut" are set to signal state "1".

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value that is specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. When the value at the IN input is greater than 31, a modulo 32 instruction is executed.

Parameters

The following table shows the parameters of the "Decode" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	UINT	I, Q, M, D, L, P, or constant	Position of the bit in the output value which is set.
OUT	Output	Bit strings	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

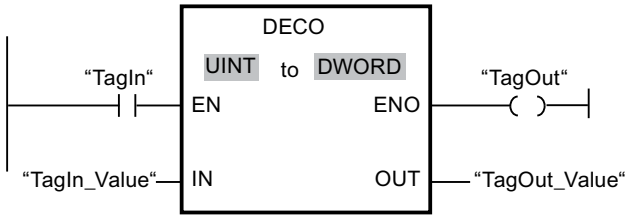

The following figure shows how the instruction works using specific operand values:

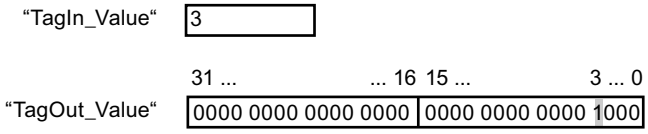

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction reads bit number "3" from the value of the operand "TagIn_Value" at the input and sets the third bit to the value of the operand "TagOut_Value" at the output.

If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

ENCO: Encode

Description

You can use the "Encode" instruction to read the bit number of the least significant bit in the input value and to send it to the OUT output.

The "Encode" instruction selects the least significant bit of the value at the IN input and writes its bit number to the tag in the OUT output.

Parameters

The following table shows the parameters of the "Encode" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L, P, or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

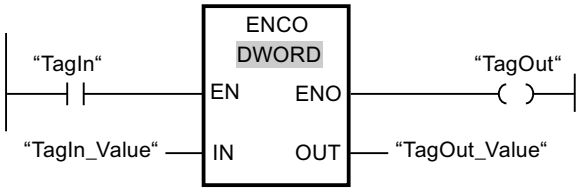

The following figure shows how the instruction works using specific operand values:

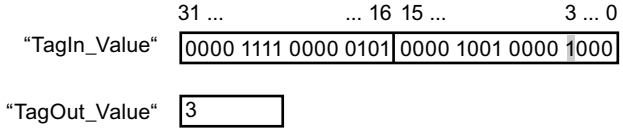

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction selects the least significant bit at the "TagIn_Value" input and writes bit position "3" to the tag in the "TagOut_Value" output.

If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

SEL: Select

Description

Depending on a switch (G input), the "Select" instruction selects one of the inputs, IN0 or IN1 and copies its content to the OUT output. When the input G has signal state "0", the value at the input IN0 is moved. When the input G has signal state "1", the value at the input IN1 is moved to the output OUT.

All tags at all parameters must have the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
G	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, TOD, CHAR, WCHAR, DATE	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, LDT, CHAR, WCHAR, DATE	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	First input value

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN1	Input	Bit strings, integers, floating-point numbers, timers, TOD, CHAR, WCHAR, DATE	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, LDT, CHAR, WCHAR, DATE	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, TOD, CHAR, WCHAR, DATE	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, LDT, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

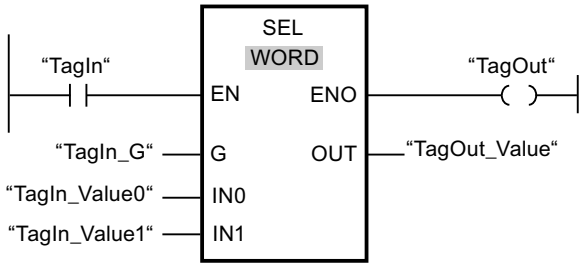

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If the "TagIn" operand has signal state "1", the instruction is executed. Based on the signal state at the "TagIn_G" input, the value at the "TagIn_Value0" or "TagIn_Value1" input is selected and moved to the "TagOut_Value" output. If the instruction is executed without errors, the enable output ENO has signal state "1" and the output "TagOut" is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

MUX: Multiplex

Description

You can use the instruction "Multiplex" to copy the content of a selected input to output OUT. The number of selectable inputs of the instruction box can be expanded. You can declare a maximum of 32 inputs.

The inputs are numbered automatically in the box. Numbering starts at IN0 and is incremented continuously with each new input. You can use the K parameter to define the input whose content is to be copied to the OUT output. If the value of the parameter K is greater than the number of available inputs, the content of the ELSE parameter is copied to the OUT output and the ENO enable output is assigned the signal state "0".

The "Multiplex" instruction can only be executed if the tags have the same data type in all inputs and in the OUT output. The K parameter is an exception, since only integers can be specified for it.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The input at the K parameter is located outside the available inputs. This applies regardless of whether the ELSE input is used. The value at the OUT output remains changed.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the "Multiplex" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
K	Input	Integers	Integers	I, Q, M, D, L, P, or constant	Specifies the input whose content is to be copied. • If K = 0 => Parameter IN0 • If K = 1 => Parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	Second input value
INn	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	Optional input values

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
ELSE	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	Specifies the value to be copied when K > n.
OUT	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Output to which the value is to be copied.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

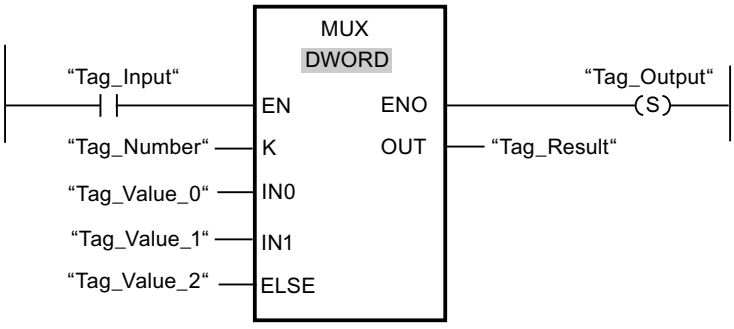

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
K	Tag_Number	1
IN0	Tag_Value_0	DW#16#00000000
IN1	Tag_Value_1	DW#16#003E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#003E4A7D

If the "Tag_Input" operand has signal state "1", the instruction is executed. Depending on the value of the operand "Tag_Number", the value at input "Tag_Value_1" is copied and assigned to the operand at output "Tag_Result". If the instruction is executed without errors, the "ENO" and "Tag_Output" enable outputs are set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

DEMUX: Demultiplex

Description

You can use the instruction "Demultiplex" to copy the content of the input IN to a selected output. The number of selectable outputs can be extended in the instruction box. The outputs are numbered automatically in the box. Numbering starts at OUT0 and continues consecutively with each new output. You can use the K parameter to define the output to which the content of the IN input is to be copied. The other outputs are not changed. If the value of the parameter K is greater than the number of available outputs, then the content of input IN is copied to the parameter ELSE and the enable output ENO is assigned the signal state "0".

The instruction "Demultiplex" can only be executed if the tags at the input IN and at all outputs are of the same data type. The K parameter is an exception, since only integers can be specified for it.

The enable output ENO is reset if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the instruction "Demultiplex":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
K	Input	Integers	Integers	I, Q, M, D, L, P, or constant	Specifies the output to which the input value (IN) is copied. • If K = 0 => Parameter OUT0 • If K = 1 => Parameter OUT1, etc.
IN	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	Input value
OUT0	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	First output
OUT1	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Second output

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
OUTn	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Optional outputs
ELSE	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Output to which the input value (IN) at K > n is copied.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:

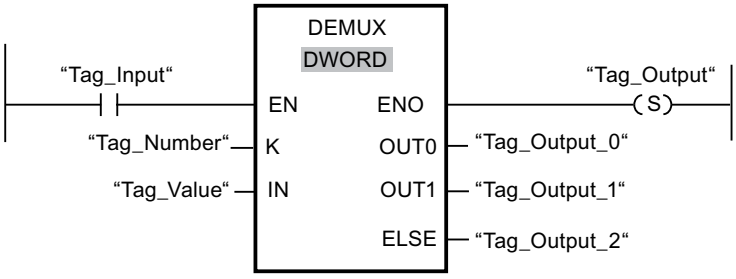

The following tables show how the instruction works using specific operand values:

Table 4-1 Input values of the "Demultiplex" instruction before network execution

Parameter	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

Table 4-2 Output values of the "Demultiplex" instruction after network execution

Parameter	Operand	Values	
OUT0	Tag_Output_0	Unchanged	Unchanged
OUT1	Tag_Output_1	DW#16#FFFFFFFF	Unchanged
ELSE	Tag_Output_2	Unchanged	DW#16#003E4A7D

If the "Tag_Input" has signal state "1", the instruction is executed. Depending on the value of the operand "Tag_Number", the value at input "IN" is copied to the corresponding output.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

Shift and rotate

SHR: Shift right

Description

You can use the "Shift right" instruction to shift the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure show how the content of an operand of integer data type is shifted four bit positions to the right:

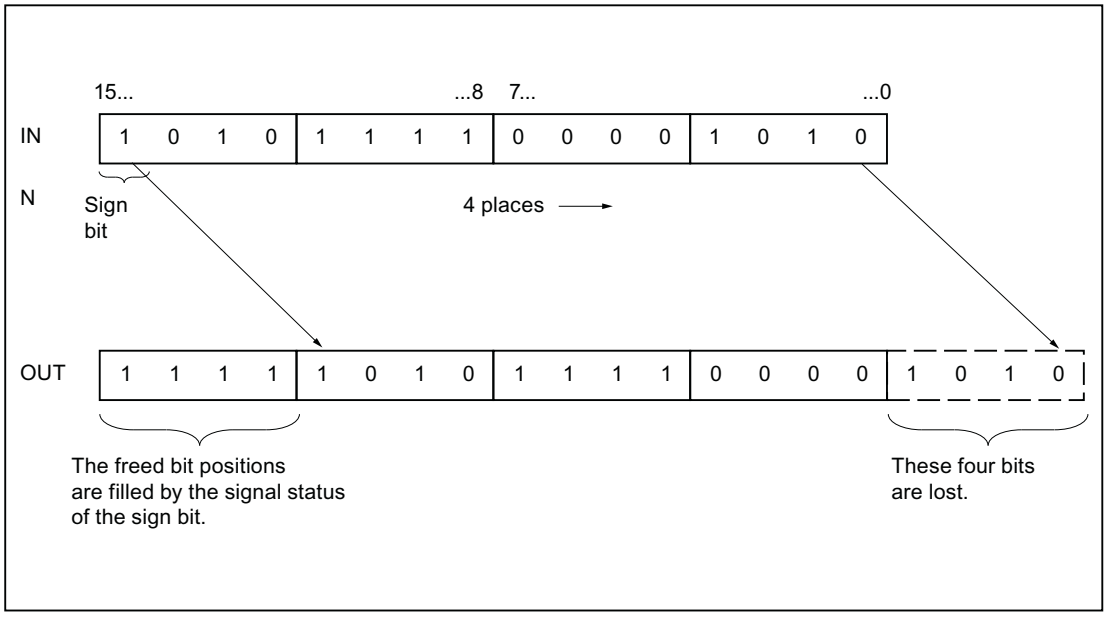

Parameters

The following table shows the parameters of the "Shift right" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

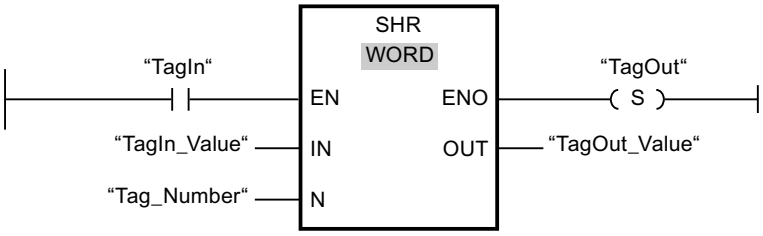

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If operand "TagIn" has the signal state "1", the "Shift right" instruction is executed. The content of operand "TagIn_Value" is shifted three bit positions to the right. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

SHL: Shift left

Description

You can use the "Shift left" instruction to shift the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure shows how the content of an operand of the WORD data type is shifted by six bit positions to the left:

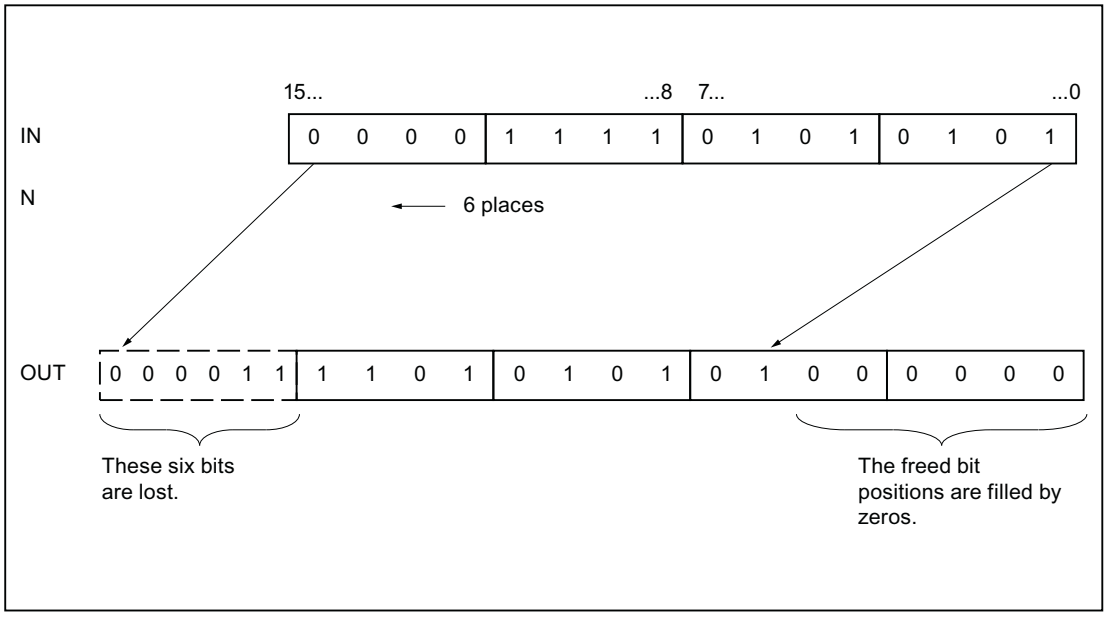

Parameters

The following table shows the parameters of the "Shift left" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

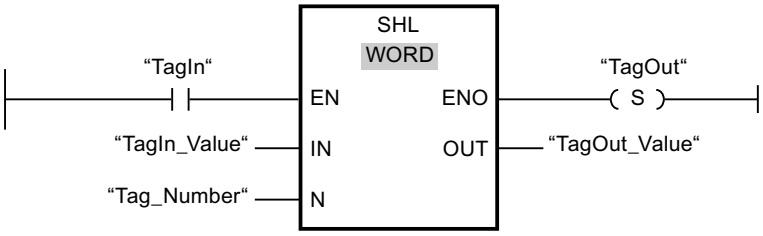

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If operand "TagIn" has the signal state "1", the "Shift left" instruction is executed. The content of operand "TagIn_Value" is shifted four bit positions to the left. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

ROR: Rotate right

Description

You can use the "Rotate right" instruction to rotate the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of available bit positions, the operand value at input IN is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of DWORD data type is rotated three positions to the right:

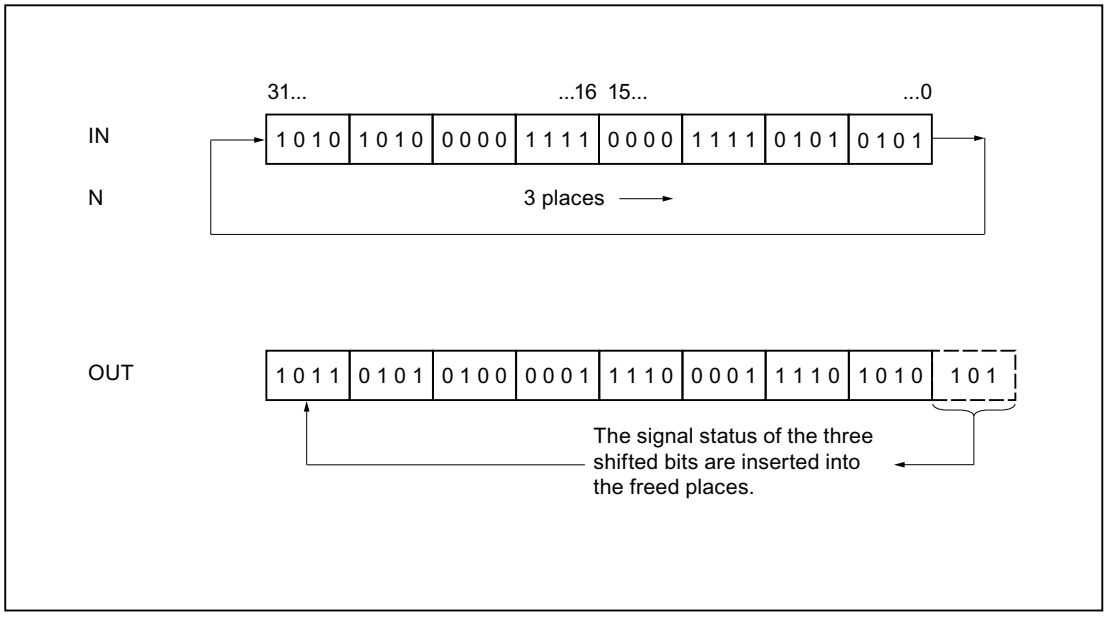

Parameters

The following table shows the parameters of the instruction "Rotate right":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

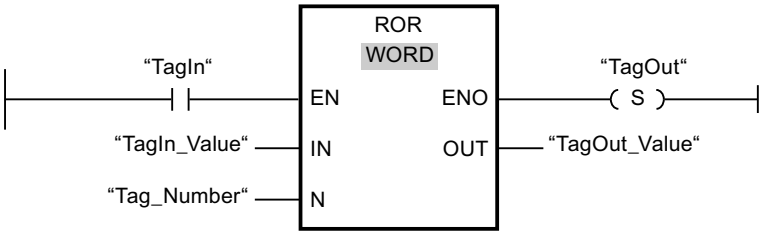

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

If operand "TagIn" has the signal state "1", the "Rotate right" instruction is executed. The content of operand "TagIn_Value" is rotated five bit positions to the right. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

ROL: Rotate left

Description

You can use the "Rotate left" instruction to rotate the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of available bit positions, the operand value at input IN is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of DWORD data type is rotated three positions to the left:

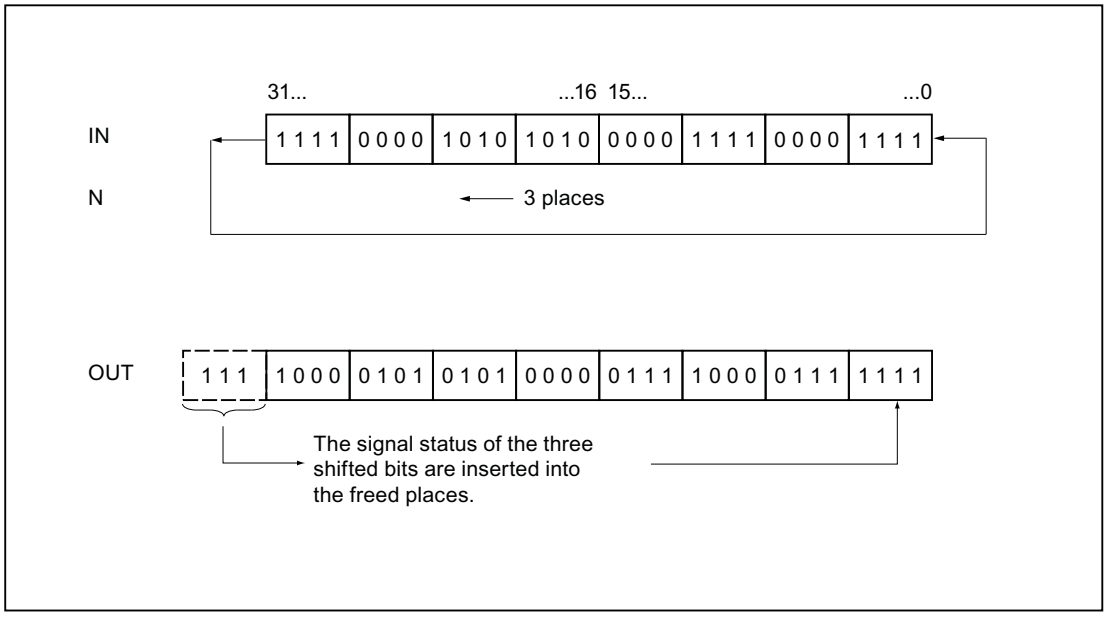

Parameters

The following table shows the parameters of the instruction "Rotate left":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

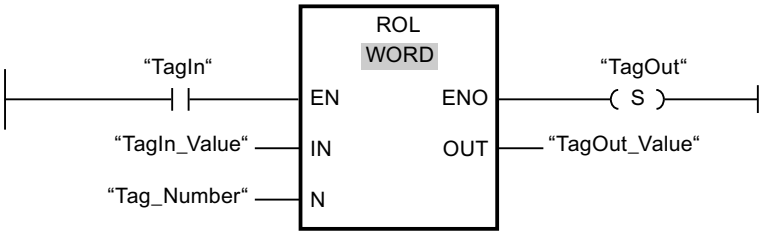

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	1010 1000 1111 0110
N	Tag_Number	5
OUT	TagOut_Value	0001 1110 1101 0101

If input "TagIn" has the signal state "1", the instruction "Rotate left" is executed. The content of operand "TagIn_Value" is rotated five bit positions to the left. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

Legacy (S7-1500)

DRUM: Implement sequencer (S7-1500)

Description

You can use the "Implement sequencer" instruction to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this

calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). When a bit of the configurable mask has a signal state of "1", the OUT_VAL value sets/resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit at the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit at the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L or constant	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Maximum step number (for example: LST_STEP = 16#08; a maximum of 8 steps is possible.)
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L or constant	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)

Parameter	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P or constant	Current numerical value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P or constant	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L or constant	Preset counter value for each step [1 to 16] where one clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

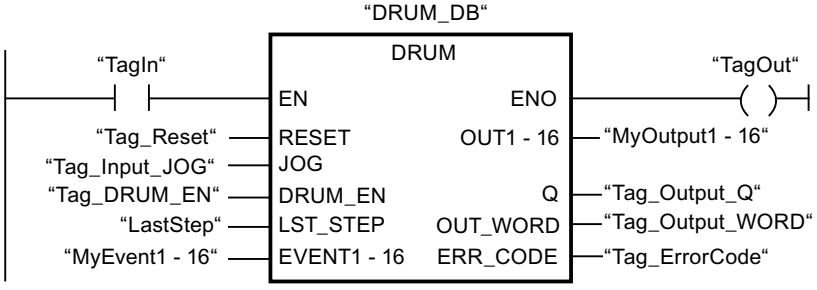

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameter	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A

Parameter	Address	Value
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE

Parameter	Address	Value
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE

Parameter	Operand	Address	Value
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on LAD (Page 7121)

DCAT: Discrete control-timer alarm (S7-1500)

Description

You can use the "Discrete control-timer alarm" to count the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET is < PT) OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state of the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".
- When the signal state of the parameters CMD and CMD_HIS is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state at the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the CMD, CMD_HIS and O_FB parameters are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CMD	Input	BOOL	I, Q, M, D, L or constant	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.

Parameter	Declaration	Data type	Memory area	Description
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
CMD_HIS	Static	BOOL	D, L or constant	CMD history bit

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The CMD_HIS parameter of the instance data block is set to the signal state "1" and the ET parameter is reset to "0".

Note

You can initialize static parameters in the data block.

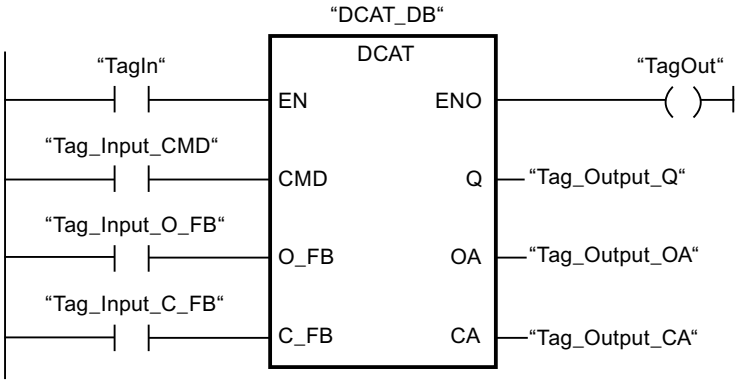

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE

Parameter	Operand	Value
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

MCAT: Motor control-timer alarm (S7-1500)

Description

The "Motor control-timer alarm" instruction is used to accumulate the time from the point at which one of the command inputs (opening or closing) is switched on. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameters	Declaration	Data type	Memory space	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
O_CMD	Input	BOOL	I, Q, M, D, L or constant	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L or constant	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L or constant	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	A signal state of "0" indicates an error condition.
ET	Static	DINT	D, L or constant	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
O_HIS	Static	BOOL	D, L or constant	"Open" history bit
C_HIS	Static	BOOL	D, L or constant	"Close" history bit

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened

Input parameters								Output parameters								
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped
Legend:																
INC	Add the time difference (ms) since the last processing of the FB to ET															
PT	PT is set to the same value as ET															
X	Cannot be used															
<PT	ET < PT															
>=PT	ET >= PT															
If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case: OO = FALSE CO = FALSE OA = FALSE CA = FALSE ET = PT Q = TRUE																

The "Motor control-timer alarm" instruction returns no error information.

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

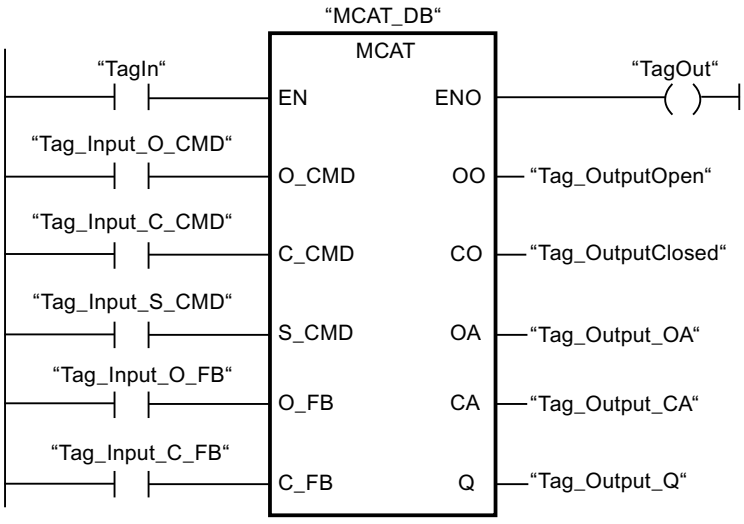

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE

Parameter	Operand	Value
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

IMC: Compare input bits with the bits of a mask (S7-1500)

Description

You can use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise, the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

SMC: Compare scan matrix (S7-1500)

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P or constant	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on LAD (Page 7121)

LEAD_LAG: Lead and lag algorithm (S7-1500)

Description

You can use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the instruction "Lead and lag algorithm" is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the instruction "Lead and lag algorithm" in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two operations together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher

frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed. Constants can also be specified on the IN parameter.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time Constants can also be specified on the SAMPLE_T parameter.
OUT	Output	REAL	I, Q, M, D, L	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in in the same unit as sample time.
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

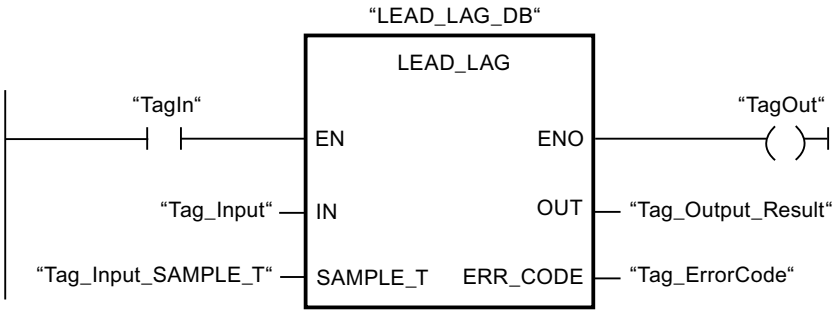

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on LAD (Page 7121)

SEG: Create bit pattern for seven-segment display (S7-1500)

Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)
0000	00111111	0
0001	00000110	1
0010	01011011	2
0011	01001111	3
0100	01100110	4
0101	01101101	5
0110	01111101	6
0111	00000111	7
1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	B
1100	00111001	C
1101	01011110	D
1110	01111001	E
1111	01110001	F

Seven-segment display

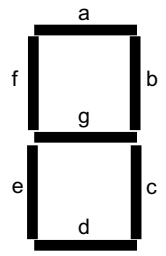

Parameter

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P, or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:

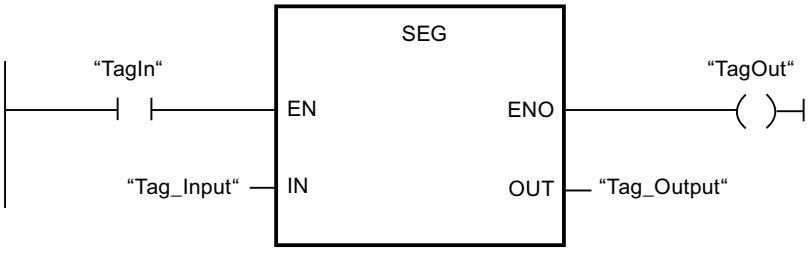

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 Display: 1234

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

BCDCPL: Create tens complement (S7-1500)

Description

You can use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L, P, or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:

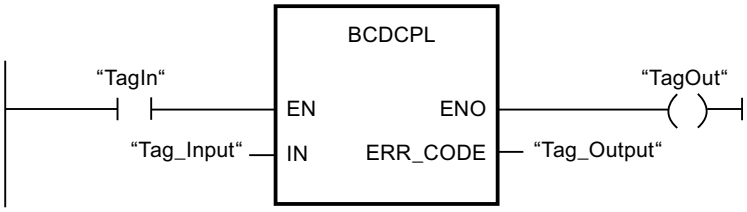

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".		

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on LAD (Page 7121)

BITSUM: Count number of set bits (S7-1500)

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output on the RET_VAL parameter.

Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P, or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of bits to be set

Example

The following example shows how the instruction works:

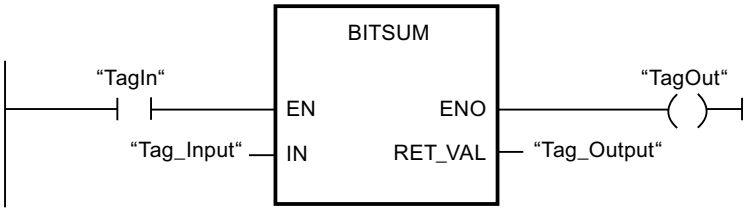

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".		

For more information and the program code to the above named example, refer to: Sample Library for Instructions (Page 593)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

4.1.3.2 FBD

Bit logic operations

&: AND logic operation

Description

You can use the instruction "AND logic operation" to query the signal states of two or more specified operands and evaluate them according to the AND truth table.

If the signal state of all the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of one of the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

Note

Instructions displayed as "inactive" at the operands

Under the following conditions:

- The instruction "AND logic operation" is interconnected at its inputs with multiple comparator operations (e.g. CMP ==) with the data type String, WString or Variant.
- The result "FALSE" for the instruction "AND logic operation" is already fulfilled by comparator operations of the same type (data type String, WString or Variant) with the signal state "FALSE".

Result: In online mode, the following comparator operations (data type String, WString or Variant) of the interconnection is shown as "inactive".

Example:

Operand 1: CMP == (data type String) returns the signal state "TRUE". CMP == is shown as "active".

Operand 2: CMP == (data type INT) returns the signal state "TRUE". CMP == is shown as "active".

Operand 3: CMP == (data type String) returns the signal state "FALSE". CMP == is shown as "active".

Operand 4: CMP == (data type String) returns the signal state "TRUE". CMP == is shown as "inactive".

Result of the "AND logic operation": Signal state "FALSE".

Parameters

The following table shows the parameters of the instruction "AND logic operation":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:

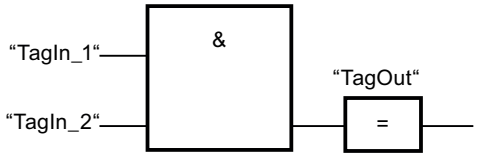

Output "TagOut" is set, when the signal state of the operands "TagIn_1" and "TagIn_2" is "1" and reset when the state of the operands "TagIn_1" and "TagIn_2" is "0".

See also

AND truth table (Page 944)

Example of detecting the direction of a conveyor belt (Page 7234)

Example of controlling room temperature (Page 7242)

Overview of the valid data types (Page 231)

Adding additional inputs and outputs to FBD elements (Page 7218)

Insert input (Page 949)

Example of detecting the fill level of a storage area (Page 7236)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

FBD programming examples (Page 7232)

AND truth table

Results of the logic operation

The following table shows the results that arise from the AND logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	1	1
0	1	0
1	0	0
0	0	0

See also

&: AND logic operation (Page 942)

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

>=1: OR logic operation

Description

You can use the instruction "OR logic operation" to query the signal states of two or more specified operands and evaluate them according to the OR truth table.

If the signal state of one of the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of all the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

Note

Instructions displayed as "inactive" at the operands

Under the following conditions:

- The instruction "OR logic operation" is interconnected at its inputs with multiple comparator operations (e.g. CMP ==) with the data type String, WString or Variant.
- The result ("TRUE") for the instruction "OR logic operation" is already fulfilled by the comparator operation of the same type (data type String, WString or Variant) with the signal state "TRUE".

Result: In online mode, the following comparator operations (data type String, WString or Variant) of the interconnection is shown as "inactive".

Example:

Operand 1: CMP == (data type String) returns the signal state "FALSE". CMP == is shown as "active".

Operand 2: CMP == (data type String) returns the signal state "TRUE". CMP == is shown as "active".

Operand 3: CMP == (data type String) returns the signal state "TRUE". CMP == is shown as "inactive".

Operand 4: CMP == (data type INT) returns the signal state "TRUE". CMP == is shown as "active".

Result of the "OR logic operation": Signal state "TRUE".

Parameters

The following table shows the parameters of the instruction "OR logic operation":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:

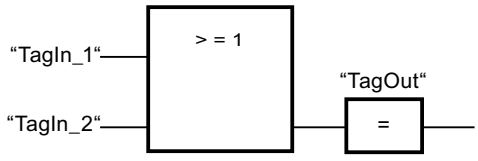

Output "TagOut" is set, when the signal state of the operands "TagIn_1" or "TagIn_2" is "1".

See also

OR truth table (Page 946)
Example of controlling a conveyor belt (Page 7232)
Adding additional inputs and outputs to FBD elements (Page 7218)
Overview of the valid data types (Page 231)
Insert input (Page 949)
Inserting FBD elements (Page 7198)
Editing FBD elements (Page 7215)
Inserting operands in FBD instructions (Page 7222)
FBD programming examples (Page 7232)

OR truth table

Results of the logic operation

The following table shows the results that arise from the OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	1
0	0	0

See also

>=1: OR logic operation (Page 944)
Overview of the valid data types (Page 231)
Inserting FBD elements (Page 7198)
Editing FBD elements (Page 7215)
Inserting operands in FBD instructions (Page 7222)

X: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" instruction to query the result of a signal state query according to the EXCLUSIVE OR truth table.

With an "EXCLUSIVE OR logic operation" instruction, the signal state is "1" when the signal state of one of the two specified operands is "1". When more than two operands are queried, the common RLO is "1" if an odd number of the queried operands returns the result "1".

Parameters

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:

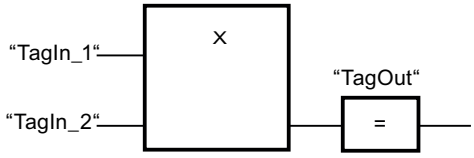

Output "TagOut" is set when the signal state of the operands "TagIn_1" and "TagIn_2" is "1". When both operands return the signal state "1" or "0", the output "TagOut" is reset.

See also

EXCLUSIVE OR truth table (Page 948)

Adding additional inputs and outputs to FBD elements (Page 7218)

Overview of the valid data types (Page 231)

Insert input (Page 949)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

EXCLUSIVE OR truth table

Results of the logic operation

The following table shows the results that arise from the EXCLUSIVE OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	0
0	0	0

The following table shows the results that arise from the EXCLUSIVE OR logic operation of three operands:

Signal state of the first operand	Signal state of the second operand	Signal state of the third operand	Result of the logic operation
1	0	0	1
0	1	1	0
0	1	0	1
1	0	1	0
0	0	1	1
1	1	0	0
1	1	1	1
0	0	0	0

See also

X: EXCLUSIVE OR logic operation (Page 946)

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

Insert input

Description

The "Insert input" instruction is used to add an input to the box of one of the following instructions:

- "AND logic operation"
- ""OR logic operation"
- ""EXCLUSIVE OR logic operation"

You can query the signal state of several operands by the extension of an instruction box.

Parameters

The following table shows the parameters of the "Insert input" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:

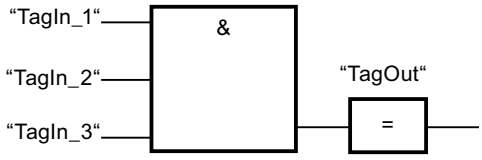

The box of the "AND logic operation" instruction was extended by an additional input at which the signal state of the operand "TagIn_3" is queried. The "TagOut" output is set, when the operands "TagIn_1" and "TagIn_2" and "TagIn_3" return signal state "1".

See also

&: AND logic operation (Page 942)

>=1: OR logic operation (Page 944)

X: EXCLUSIVE OR logic operation (Page 946)

Overview of the valid data types (Page 231)

Inserting operands in FBD instructions (Page 7222)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO).

Example

The following example shows how the instruction works:

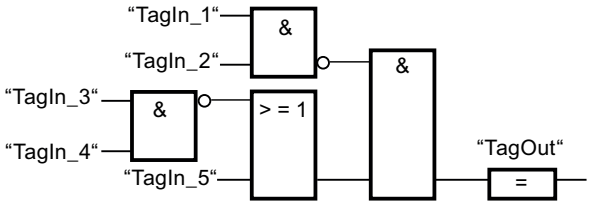

The "TagOut" output is set when the following conditions are fulfilled:

- The input "TagIn_1" and/or "TagIn_2" has signal state "0".
- The input "TagIn_3" and/or "TagIn_4" has signal state "0" or the input "TagIn_5" has signal state "1".

See also

Overview of the valid data types (Page 231)

Example of detecting the direction of a conveyor belt (Page 7234)

Example of detecting the fill level of a storage area (Page 7236)

Example of controlling room temperature (Page 7242)

FBD programming examples (Page 7232)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

=: Assignment

Description

You can use the instruction "Assignment" to set the bit of a specified operand. If the result of logic operation (RLO) at the box input has the signal state "1", the specified operand is set to signal state "1". If the signal state at the box input is "0", the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the box input is assigned directly to the operand above the assignment box.

Parameters

The following table shows the parameters of the instruction "Assignment":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:

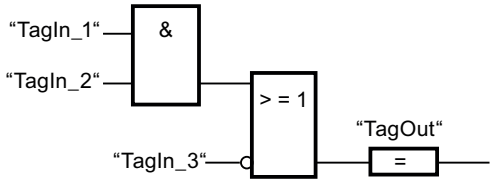

The operand "TagOut" is set at the output of the "Assignment" instruction when one of the following conditions is fulfilled:

- The inputs "TagIn_1" and "TagIn_2" have the signal state "1".
- The signal state at the input "TagIn_3" is "0".

See also

Overview of the valid data types (Page 231)

Example of detecting the fill level of a storage area (Page 7236)

Example of controlling room temperature (Page 7242)

FBD programming examples (Page 7232)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

/=: Negate assignment

Description

The instruction "Negate assignment" inverts the result of logic operation (RLO) and assigns this to the operand above the box. If the RLO at the input of the box is "1", the binary operand is reset. If the RLO at the input of the box is "0", the operand is set to signal state "1".

Parameters

The following table shows the parameters of the instruction "Negate assignment":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the negated RLO is assigned.

Example

The following example shows how the instruction works:

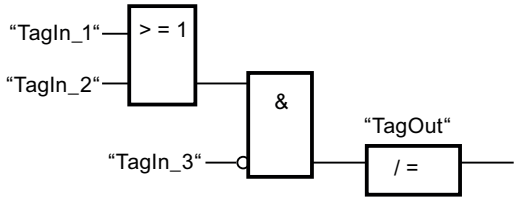

The operand "TagOut" is reset when the following conditions are fulfilled:

- The operand "TagIn_1" or "TagIn_2" has the signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

R: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is reset to "0". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

Parameters

The following table shows the parameters of the "Reset output" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Operand that is reset if RLO = "1".

Example

The following example shows how the instruction works:

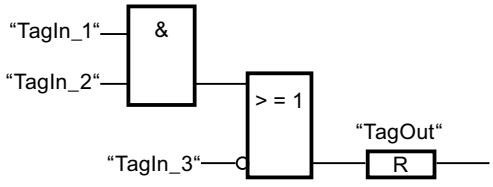

Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Example of controlling a conveyor belt (Page 7232)

Example of detecting the direction of a conveyor belt (Page 7234)

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

FBD programming examples (Page 7232)

S: Set output

Description

You can use the instruction "Set output" to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is set to "1". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

Parameters

The following table shows the parameters of the instruction "Set output":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:

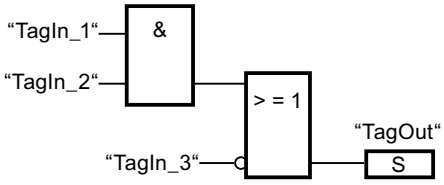

The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 231)

Example of detecting the direction of a conveyor belt (Page 7234)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

FBD programming examples (Page 7232)

SET_BF: Set bit field

Description

You use the "Set bit field" instruction to set several bits starting from a certain address.

You specify the number of bits to be set with the value of the constant at the N input. The address of the first bit to be set is defined by <Operand>. If the value of <Operand> may not be greater than the number of bits in a selected byte. If this value is greater, the instruction is not executed and the error message "Range violation for index <Operand1>" is displayed. The bits remain set until they are explicitly reset by another instruction.

The instruction is only executed if the enable input EN has the signal state "1".

Bit fields of the type PLC data type, STRUCT or ARRAY

With structures of the type PLC data type, STRUCT or ARRAY, the number of bits contained in the structure represents the maximum number of bits that can be reset:

- If you specify the value "20" at the N input, for example, and the structure only contains 10 bits, then only these 10 bits are set.
- If you specify the value "5" at the N input, for example, and the structure contains 10 bits, then exactly 5 bits are set.

Parameters

The following table shows the parameters of the "Set bit field" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
N	Input	UINT	Constant	Constant	Number of bits to be set
<Operand>	Output	BOOL	I, Q, M With a DB or an IDB, an element of an ARRAY[..] of BOOL	I, Q, M With a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be set.

Example

The following example shows how the instruction works:

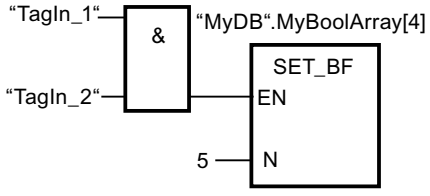

If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are set starting at the address of the operand "MyDB".MyBoolArray[4].

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

RESET_BF: Reset bit field

Description

You use the "Reset bit field" instruction to reset several bits starting from a certain address.

You specify the number of bits to be reset with the value of the constant at the N input. The address of the first bit to be reset is defined by <Operand>. If the value of the constant may not be greater than the number of bits in a selected byte. If this value is greater, the instruction is not executed and the error message "Range violation for index <Operand 1>" is displayed. The bits remain set until they are explicitly reset by another instruction.

The instruction is only executed if the enable input EN has the signal state "1".

Bit fields of the type PLC data type, STRUCT or ARRAY

With structures of the type PLC data type, STRUCT or ARRAY, the number of bits contained in the structure represents the maximum number of bits that can be set:

- If you specify the value "20" at the N input, for example, and the structure only contains 10 bits, then only these 10 bits are reset.
- If you specify the value "5" at the N input, for example and the structure contains 10 bits, then exactly 5 bits are reset.

Parameters

The following table shows the parameters of the "Reset bit field" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
N	Input	UINT	Constant	Constant	Number of bits to be reset
<Operand>	Output	BOOL	I, Q, M With a DB or an IDB, an element of an ARRAY[..] of BOOL	I, Q, M With a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be reset.

Example

The following example shows how the instruction works:

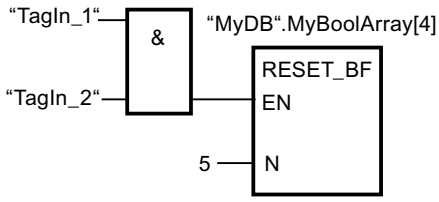

If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are reset starting at the address of the operand "MyDB".MyBoolArray[4].

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

SR: Set/reset flip-flop

Description

You can use the "Set/reset flip-flop" instruction to set or reset the bit of a specified operand based on the signal state of the inputs S and R1. If the signal state is "1" at input S and "0" at input R1, the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Parameters

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
S	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable setting
R1	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable resetting

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is set or reset
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:

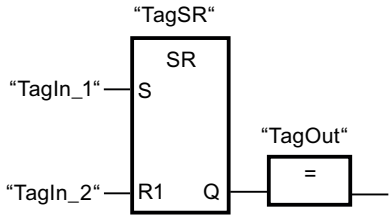

The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

RS: Reset/set flip-flop

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Parameters

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
R	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable resetting
S1	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable setting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is reset or set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:

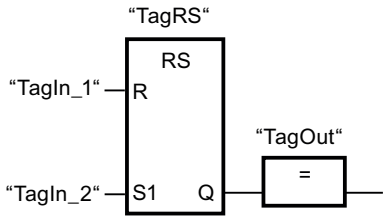

The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

P: Scan operand for positive signal edge

Description

You can use the "Scan operand for positive signal edge" instruction to determine whether there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

The following figure shows the change in the signal state in case of a negative and a positive signal edge:

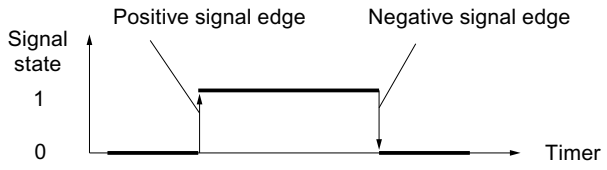

The positive signal edge is queried each time the instruction executes. When a positive signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved.

Example

The following example shows how the instruction works:

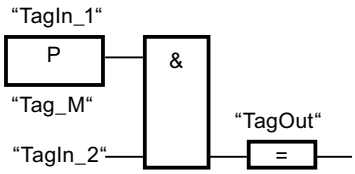

The "TagOut" output is set when the following conditions are fulfilled:

- There is a rising edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 231)

Example of detecting the direction of a conveyor belt (Page 7234)

FBD programming examples (Page 7232)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

N: Scan operand for negative signal edge

Description

You can use the "Scan operand for negative signal edge" instruction to determine whether there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

The following figure shows the change in the signal state in case of a negative and a positive signal edge:

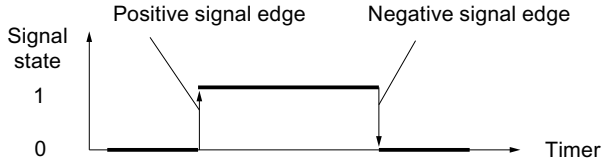

The negative signal edge is queried each time the instruction executes. When a negative signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved.

Example

The following example shows how the instruction works:

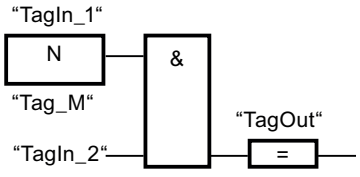

The "TagOut" output is set when the following conditions are fulfilled:

- There is a falling edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Inserting operands in FBD instructions (Page 7222)

Editing FBD elements (Page 7215)

P=: Set operand on positive signal edge

Description

You can use the instruction "Set operand on positive signal edge" to set a specified operand (<Operand2>) when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand1>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

The positive signal edge is queried each time the instruction executes. When a positive edge is detected, <Operand2> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

You specify the operand (<Operand2>) to be set in the operand placeholder above the instruction. Specify the edge memory bit (<Operand1>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the instruction "Set operand on positive signal edge":

Parameters	Declaration	Data type	Memory area	Description
<Operand2>	Output	BOOL	I, Q, M, D, L	Operand which is set when there is a positive signal edge.
<Operand1>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows the parameters of the instruction:

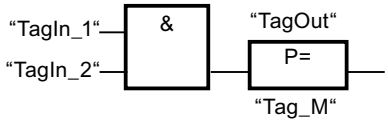

The "TagOut" output is set for one program cycle, when the signal state at the input of the instruction box switches from "0" to "1" (positive signal edge). In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

N=: Set operand on negative signal edge

Description

You can use the instruction "Set operand on negative signal edge" to set a specified operand (<Operand1>) when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand2>). If the instruction detects a change in the RLO from "1" to "0", there is a negative signal edge.

The negative signal edge is queried each time the instruction executes. When a negative signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

You specify the operand (<Operand1>) to be set in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the instruction "Set operand on negative signal edge":

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set when there is a negative signal edge.
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows how the instruction works:

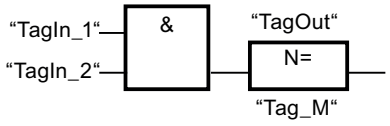

The operand "TagOut" is set for one program cycle if the signal state at the input of the instruction box changes from "1" to "0" (negative signal edge). In all other cases, the operand "TagOut" has the signal state "0".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Inserting operands in FBD instructions (Page 7222)

Editing FBD elements (Page 7215)

P_TRIG: Scan RLO for positive signal edge

Description

Use the instruction "Scan RLO for positive signal edge" to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

The positive signal edge is queried each time the instruction executes. As soon as a positive single edge is detected, the output Q of the instruction returns the signal state "1" for the length of a program cycle. In all other cases, the output returns the signal state "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the instruction "Scan RLO for positive signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

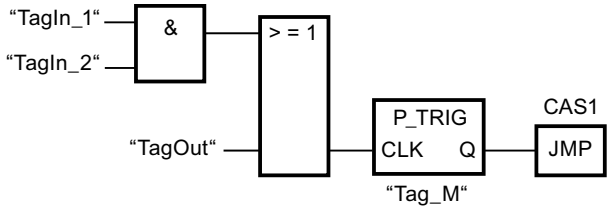

The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

N_TRIG: Scan RLO for negative signal edge

Description

Use the instruction "Scan RLO for negative signal edge" to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query saved in the edge memory bit (<Operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative signal edge.

The negative signal edge is queried each time the instruction executes. As soon as a negative single edge is detected, the output Q of the instruction returns the signal state "1" for the length of a program cycle. In all other cases, the signal state at the output of the instruction is "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

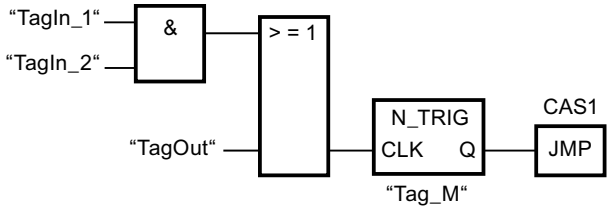

The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

R_TRIG: Detect positive signal edge

Description

With the "Detect positive signal edge" instruction, you can detect a state change from "0" to "1" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "0" to "1", a positive signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

Parameters

The following table shows the parameters of the "Detect positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

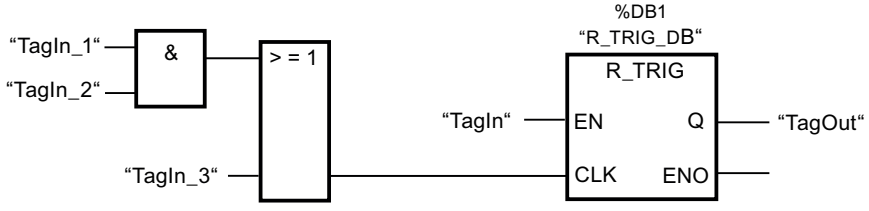

The previous state of the tag at the CLK input is stored in the "R_TRIG_DB" tag. If a change in the signal state from "0" to "1" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

F_TRIG: Detect negative signal edge

Description

With the "Detect negative signal edge" instruction, you can detect a state change from "1" to "0" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "1" to "0", a negative signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

Parameters

The following table shows the parameters of the "Detect negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

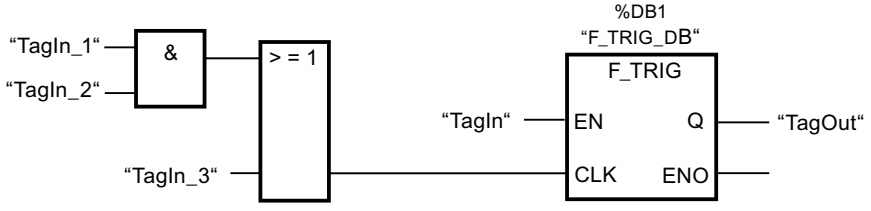

The previous state of the tag at the CLK input is stored in the "F_TRIG_DB" tag. If a change in the signal state from "1" to "0" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

Timer operations

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the Q output for a programmed duration. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. Output Q is set for the duration PT, regardless of the subsequent course of the input signal. Even when a new positive signal edge is detected, the signal state of the Q output is not affected as long as the PT duration is running.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of the time duration PT is reached. If the configured time duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TP_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Generate pulse" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T,C, P or constant	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Pulse output
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current timer value

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:

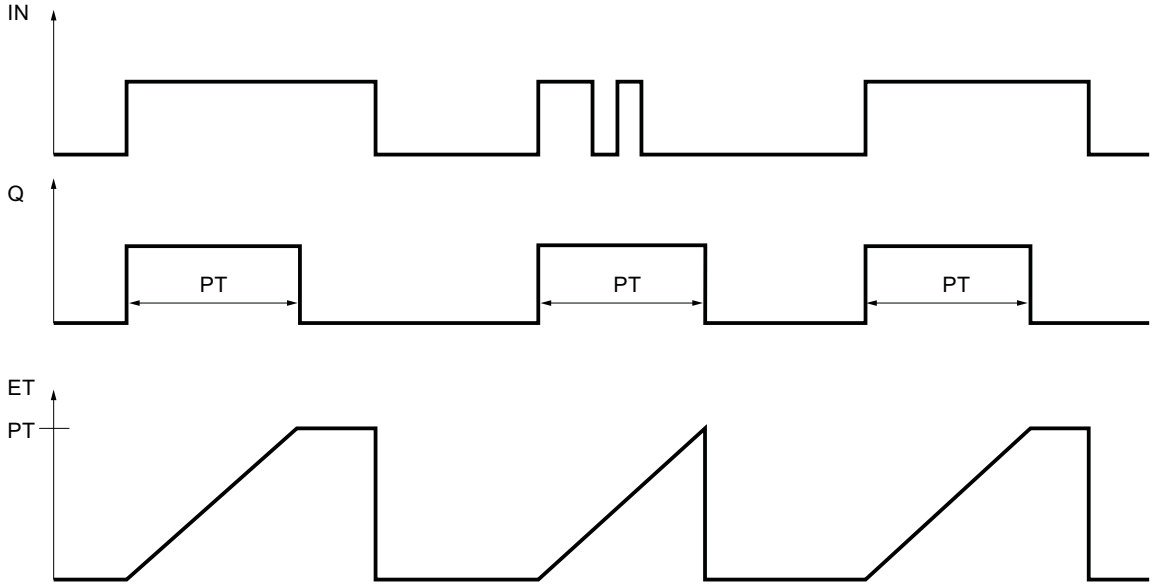

Example

The following example shows how the instruction works:

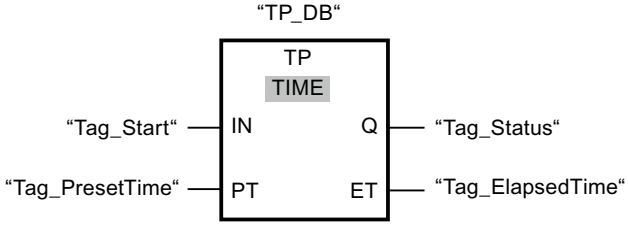

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Start	Signal transition "0" => "1"
PT	Tag_PresetTime	T#10s
Q	Tag_Status	TRUE
ET	Tag_ElapsedTime	from T#0s => T#10s

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTime" operand. When the time expires, the "Tag_Status" operand is reset to signal state "0".

See also

Overview of the valid data types (Page 231)

Example of controlling room temperature (Page 7242)

Instances (Page 62)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

TON: Generate on-delay

Description

You can use the instruction "Generate on-delay" to delay setting of the Q output by the time configured with the PT time. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the time PT has elapsed, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

The current time value can be queried at the output ET. The timer value starts at T#0s and ends when the value of the time duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TON_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T,C, P or constant	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current timer value

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:

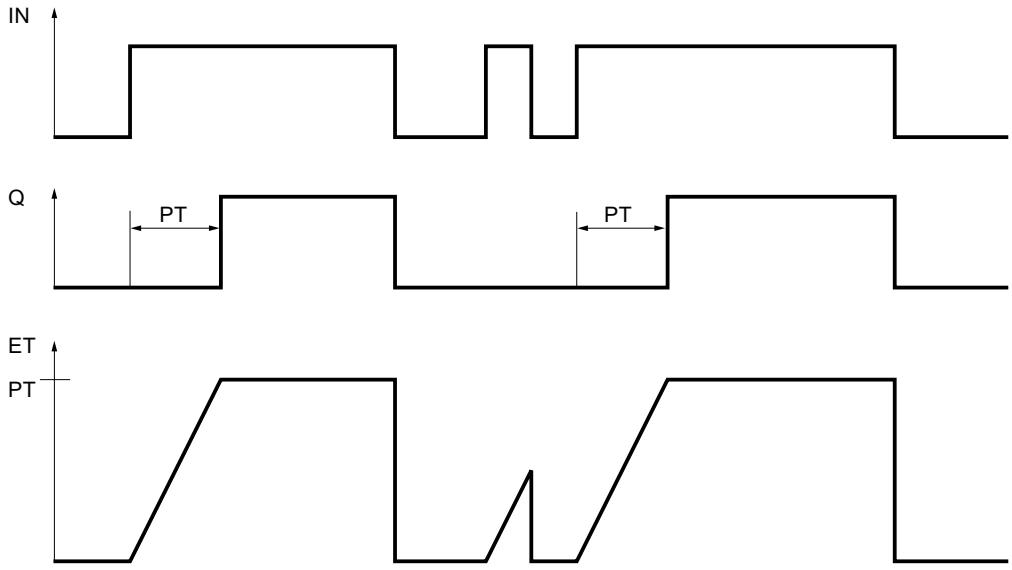

Example

The following example shows how the instruction works:

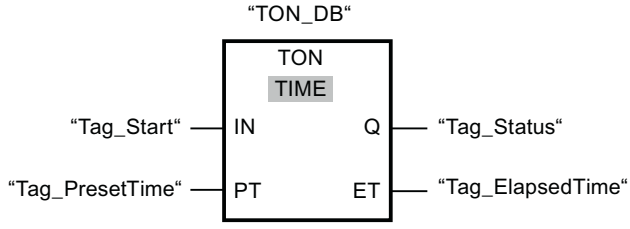

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Start	Signal transition "0" => "1"
PT	Tag_PresetTime	T#10s
Q	Tag_Status	FALSE; after 10 s => TRUE
ET	Tag_ElapsedTime	from T#0s => T#10s

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. When the time duration expires, the "Tag_Status" operand is set to the signal state "1". The Tag_Status operand remains set to "1" as long as the Tag_Start operand has signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand. When the signal state at the operand "Tag_Start" changes from "1" to "0", the "Tag_Status" operand is reset.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on FBD (Page 7185)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay setting of the Q output by the time configured with the PT time. The output Q is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). When the signal state at input IN changes back to "0" (positive signal edge), the configured time duration PT starts. Output Q remains set as long as the time duration PT is running. When the PT time duration expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q continues to be "1".

The current time value can be queried at the output ET. The timer value starts at T#0s and ends when the value of the time duration PT is reached. When the PT time duration expires, the ET output remains set to the current value until input IN changes back to "1". If input IN changes to "1" before the time duration PT has expired, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC Timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TOF_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Generate off-delay" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T,C, P or constant	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is reset when the timer PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current timer value

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate off-delay" instruction:

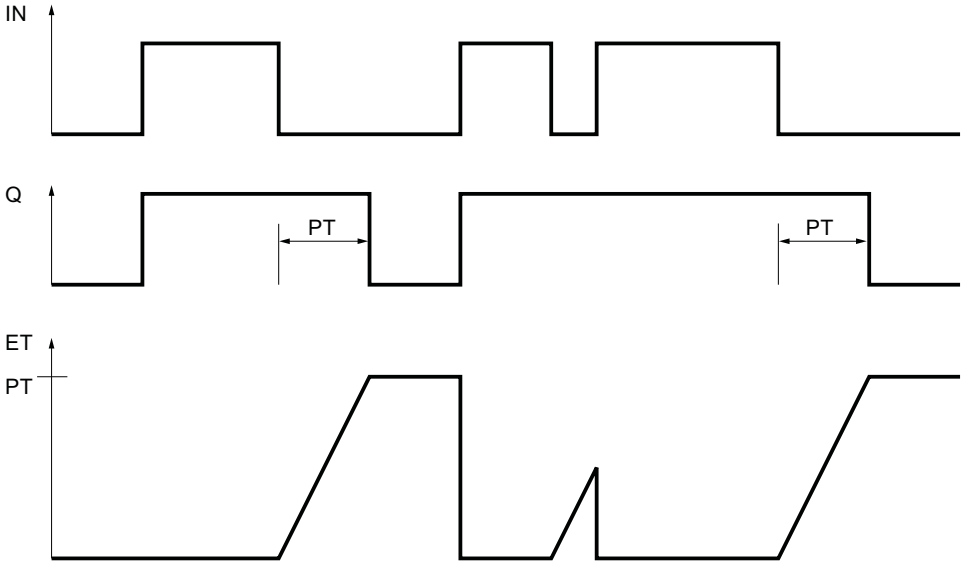

Example

The following example shows how the instruction works:

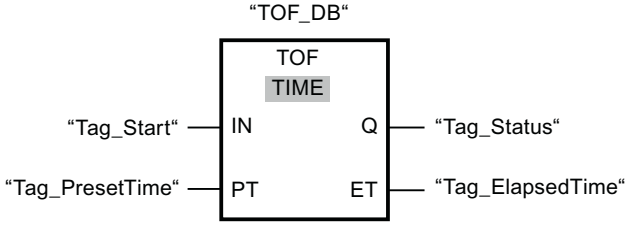

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Start	Signal transition "0" => "1"; signal transition "1" => "0"
PT	Tag_PresetTime	T#10s
Q	Tag_Status	TRUE
ET	Tag_ElapsedTime	from T#10s => T#0s

When the signal state of the operand "Tag_Start" changes from "0" to "1", the "Tag_Status" operand is set to signal state "1". When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set to TRUE. When the time has expired, the "Tag_Status" operand is reset to FALSE. The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on FBD (Page 7185)

TONR: Time accumulator

Description

The instruction "Time accumulator" is used to accumulate time values within a period set by the parameter PT. The instruction is executed and the configured time duration PT is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive edge). While the time PT is running, the time values are accumulated that are recorded when the IN input has signal state "1". The accumulated time is written to the ET output and can be queried there. When the current time value PT is reached, the output Q has the signal state "1". Output Q remains set at "1", even when the signal state at input IN changes to "0".

The R input resets the outputs ET and Q regardless of the signal state at the start input.

Each call of the "Time accumulator" instruction must be assigned an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TONR_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T,C, P or constant	Start input
R	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Reset input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Maximum duration of time recording. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current timer value

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:

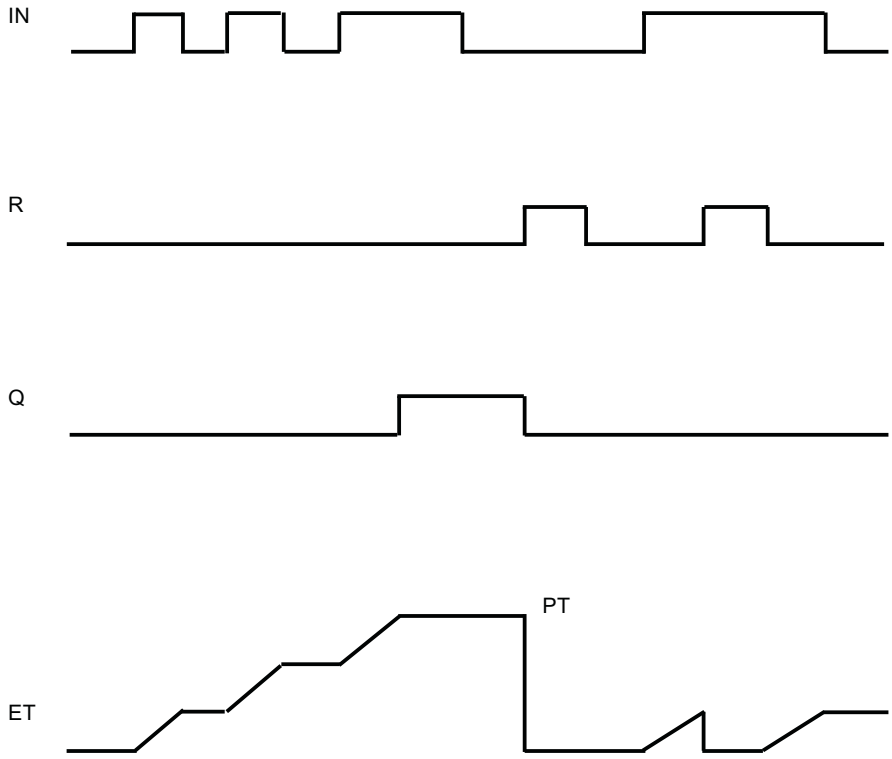

Example

The following example shows how the instruction works:

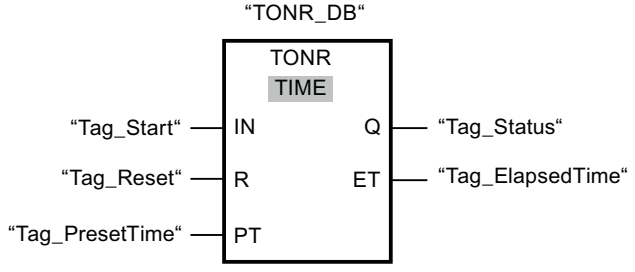

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Start	Signal transition "0" => "1"
PT	Tag_PresetTime	T#10s

Parameters	Operand	Value
Q	Tag_Status	FALSE; after 10 s => TRUE
ET	Tag_ElapsedTime	Signal transition from "0" => "1" The time T#10s expires. After 5 s signal transition from "1" => "0": The time at the "Tag_Elapsed-Time" operand remains at T#5s. After approx. 2 s, renewed signal transition from "0" => "1": The time at the "Tag_Elapsed-Time" operand continues to run at T#5s.

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. The time continues to run as long as the "Tag_Start" operand has the signal state "1". When the signal state of the "Tag_Start" operand changes from "1" to "0", the time stops and the current time value at the Tag_ElapsedTime operand is recorded. When the signal state at the "Tag_Start" operand changes again from "0" to "1", the time continues to run starting at the time value that was recorded at the transition from "1" to "0". When the time value specified for the PT parameter is reached, the "Tag_Status" operand is set to the signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on FBD (Page 7185)

TP: Start pulse timer

Description

Use the instruction "Start pulse timer" to start an IEC Timer with a specified duration as pulse. The IEC Timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC Timer runs for the specified time duration regardless of the subsequent course of the RLO. The expiry of the IEC Timer is also not affected by the detection of a new rising edge. As long as the IEC Timer is running, the querying of the timer status for "1" returns the signal state "1". When the IEC Timer has expired, the timer status returns the signal state "0".

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER or TP_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- The IEC_Timer structure is updated when the instruction is called. The time value at the ET output is only updated if the ET or Q outputs are scanned (e.g. "MyTimer".Q or "MyTimer".ET).
- When the assigned timer is accessed.

The current timer status is saved in the structure components Q of the IEC Timer. You can query the timer status with the help of a binary logic operation.

The execution of the instruction "Start pulse timer" assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

The following table shows the parameters of the instruction "Start pulse timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TP_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME	D, L	IEC Timer which is started.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction:

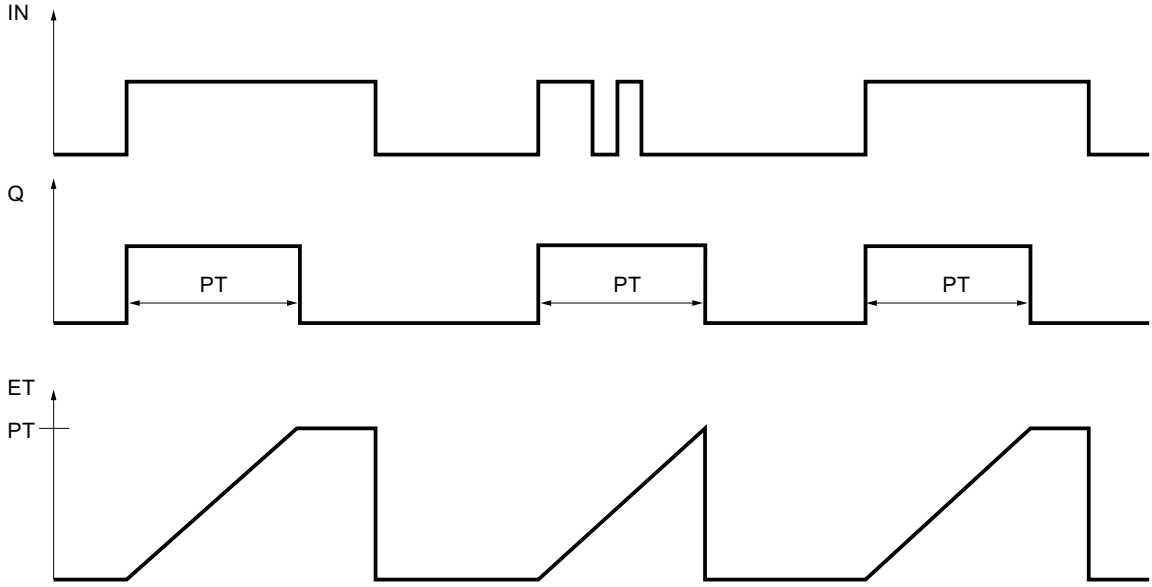

Example

The following example shows how the instruction works:

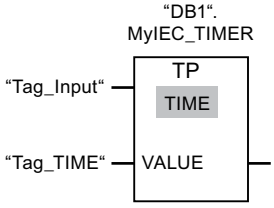

The instruction "Start pulse timer" is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The timer "DB1".MyIEC_TIMER is started for the time stored in the operand "TagTime".

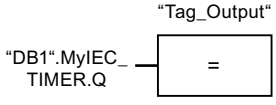

As long as the timer "DB1".MyIEC_TIMER is running, the timer status ("DB1".MyIEC_TIMER.Q) has signal state "1" and the operand "Tag_Output" is set. When the IEC Timer has expired, the signal state of the time status changes back to "0" and the "Tag_Output" operand is reset.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on FBD (Page 7185)

TON: Start on-delay timer

Description

Use the "Start on-delay timer" instruction to start an IEC Timer with a specified duration as on-delay. The IEC Timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC Timer runs for the specified time duration. The output returns the signal state "1" if the RLO at the input of the instruction has the signal state "1". If the RLO changes to "0" before the end of the timer, the running IEC Timer is reset. The query of the timer status for "1" returns the signal state "0". The IEC Timer restarts when the next rising signal edge is detected at the input of the instruction.

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start on-delay timer" stores its data in a structure of the data type IEC_TIMER or TON_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start on-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- The IEC_Timer structure is updated when the instruction is called. The time value at the ET output is only updated if the ET or Q outputs are scanned (e.g. "MyTimer".Q or "MyTimer".ET).
- When the assigned timer is accessed.

The current timer status is saved in the structure components ET of the IEC Timer. You can query the timer status with the help of a binary logic operation.

The execution of the "Start on-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

The following table shows the parameters of the instruction "Start on-delay timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TON_TIME	IEC_TIMER, IEC_LTIMER, TON_TIME, TON_LTIME	D, L	IEC Timer which is started.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction:

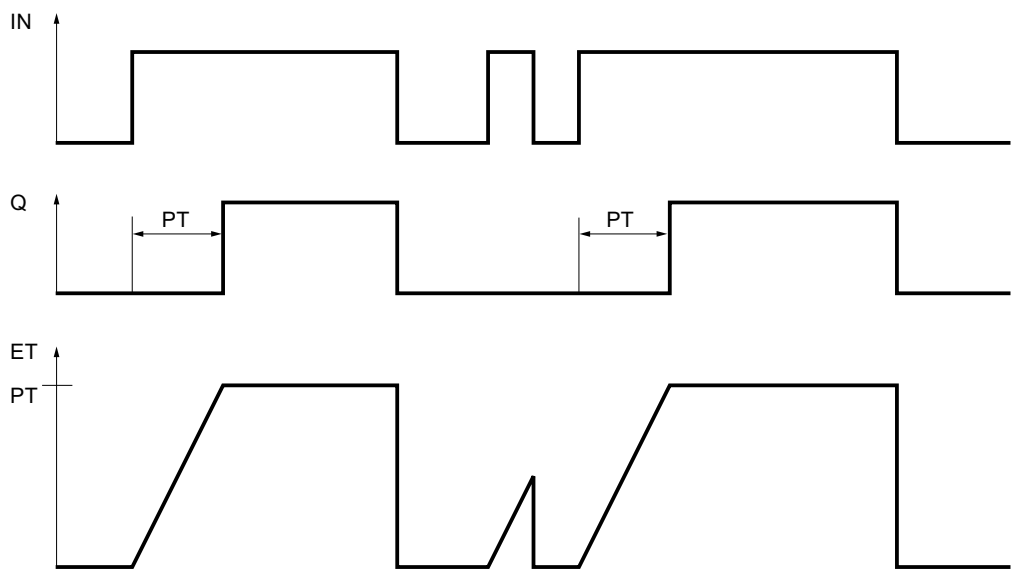

Example

The following example shows how the instruction works:

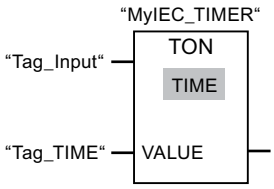

The "Start on-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The "MyIEC_TIMER" timer is started for the time stored in the "Tag_TIME" operand.

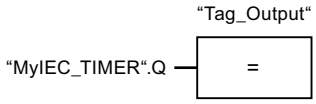

If the timer "MyIEC_TIMER" has expired and the operand "Tag_Input" has the signal state "1", querying the timer status ("MyIEC_TIMER".Q) returns signal state "1" and the "Tag_Output" operand is set. When the signal state of the operand "Tag_Input" changes to "0", the querying of the timer status returns the signal state "0" and the operand "Tag_Output" is reset.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on FBD (Page 7185)

TOF: Start off-delay timer

Description

Use the "Start off-delay timer" instruction to start an IEC Timer with a specified duration as off-delay. The query of the timer status for "1" returns the signal state "0" if the result of the logic operation (RLO) at the input of the instruction has the signal state "1". When the RLO changes from "1" to "0" (negative signal edge), the IEC Timer starts with the specified time duration. The timer status remains at signal state "1" as long as the IEC Timer is running. When the timer has run out and the RLO at the input of the instruction has the signal state "0", the timer status is set to the signal state "0". If the RLO changes to "1" before the end of the timer, the running IEC Timer is reset and the timer status remain at the signal state "1".

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start off-delay timer" stores its data in a structure of the data type IEC_TIMER or TOF_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start off-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- The IEC_Timer structure is updated when the instruction is called. The time value at the ET output is only updated if the ET or Q outputs are scanned (e.g. "MyTimer".Q or "MyTimer".ET).
- When the assigned timer is accessed.

The current timer status is saved in the structure components ET of the IEC Timer. You can query the timer status with the help of a binary logic operation.

The execution of the "Start off-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

The following table shows the parameters of the instruction "Start off-delay timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TOF_TIME	IEC_TIMER, IEC_LTI-MER, TOF_TIME, TOF_LTIME	D, L	IEC Timer which is started.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction:

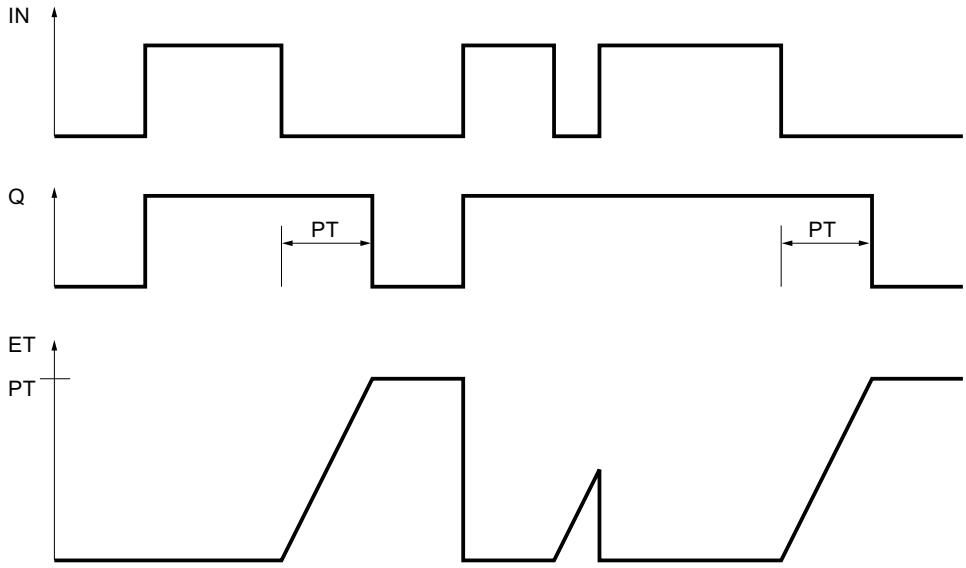

Example

The following example shows how the instruction works:

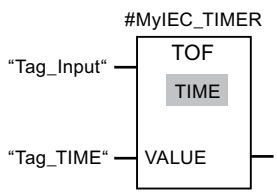

The "Start off-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "1" to "0". The #MyIEC_TIMER timer is started for the time stored in the operand "Tag_TIME".

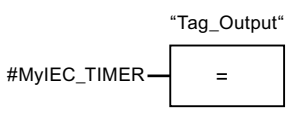

As long as timer #MyIEC_TIMER is running, the query of the time status (#MyIEC_TIMER.Q) returns the signal state "1" and operand "Tag_Output" is set. If the timer has expired and the operand "Tag_Input" has the signal state "0", the query of the timer status returns the signal state "0". If the signal state of the operand "Tag_Input" changes to "1" before timer #MyIEC_TIMER expires, the timer is reset. When the signal state of the operand "Tag_Input" is "1", the query of the timer status returns the signal state "1".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on FBD (Page 7185)

TONR: Time accumulator

Description

You can use the "Time accumulator" instruction to record how long the signal is at the input of instruction "1". The instruction is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The time is recorded as long at the RLO is "1". If the RLO changes to "0", the instruction is halted. If the RLO changes back to "1", the time recording is continued. The query of the timer status for "1" returns the signal state "1" if the recorded time exceeds the value of the specified time duration and the RLO at the input of coil is "1".

The timer status and the currently expired timer can be reset to "0" using the "Reset timer" instruction.

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER or TONR_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- The IEC_Timer structure is updated when the instruction is called. The time value at the ET output is only updated if the ET or Q outputs are scanned (e.g. "MyTimer".Q or "MyTimer".ET).
- When the assigned timer is accessed.

The current timer status is saved in the structure components ET of the IEC Timer. You can query the timer status with the help of a binary logic operation.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed only at the end of the network.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TONR_TIME	IEC_TIMER, IEC_LTIMER, TONR_TIME, TONR_LTIME	D, L	IEC Timer which is started.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction:

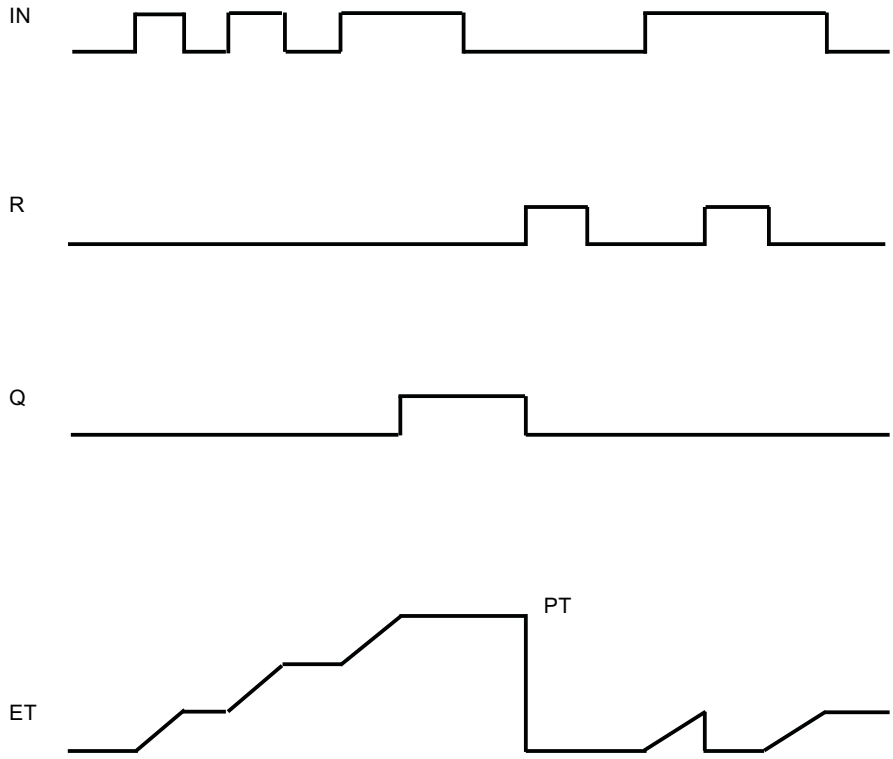

Example

The following example shows how the instruction works:

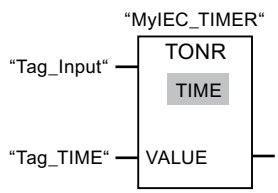

The "Time accumulator" instruction is executed if there is a positive signal edge in the RLO. The time is recorded as long as the operand "Tag_Input" has the signal state "1".

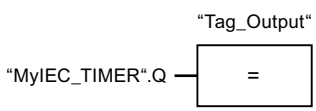

If the recorded time exceeds the value of the operand "Tag_TIME", then the query of the timer status ("MyIEC_TIMER".Q) will return the signal state "1" and the operand "Tag_Output" will be set.

See also

Overview of the valid data types (Page 231)

RT: Reset timer (Page 993)

Instances (Page 62)

Basic information on FBD (Page 7185)

RT: Reset timer

Description

You can use the "Reset timer" instruction to reset an IEC Timer to "0". You specify the IEC Timer to be reset by entering the name of the data block that contains the structure of the IEC Timer in the placeholder above the instruction.

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". When the instruction is executed the structure components of the IEC Timer are reset to "0" in the specified data block. If the RLO at box input is "0", the instruction is not executed.

The instruction does not influence the RLO. The RLO at the box input is transferred directly to the box output.

You must assign a IEC Timer declared in the program to the "Reset timer" instruction.

The instruction data is updated only when the instruction is called and not each time the assigned IEC Timer is accessed. The query of the data is only identical from the call of the instruction to the next call of the instruction.

Parameters

The following table shows the parameters of the instruction "Reset timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<IEC Timer>	InOut	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC Timer, which is reset.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

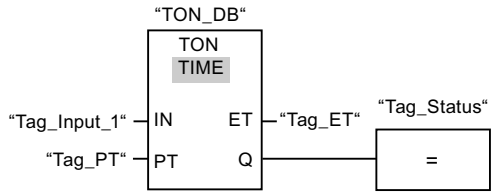

The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC Timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".

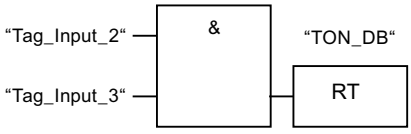

If the operands "Tag_Input_2" and "Tag_Input_3" have the signal state "1", the "Reset timer" instruction is executed and the IEC Timer stored in the data block "TON_DB" is reset.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on FBD (Page 7185)

PT: Load time duration

Description

Use the "Load time duration" instruction to set the time duration of an IEC Timer. The instruction is executed in every cycle when the result of logic operation (RLO) at the input of the instruction has the signal state "1". The instruction writes the specified time duration to the structure of the specified IEC Timer.

Note

If the specified IEC Timer is running during the execution, the instruction overwrites the current time duration of the specified IEC Timer. As a result, the timer status of the IEC Timer can change.

You must assign a IEC Timer declared in the program to the "Load time duration" instruction.

The instruction data is updated when the instruction is called and each time the assigned IEC Timer is accessed. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Parameters

The following table shows the parameters of the instruction "Load time duration":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Time duration
<IEC Timer>	InOut	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC Timer, the duration of which is set.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

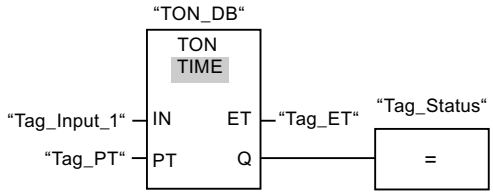

The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC Timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".

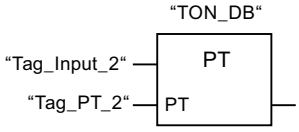

The "Load time duration" instruction is executed when the operand "Tag_Input_2" has the signal state "1". The instruction writes the time duration "Tag_PT_2" in the instance data block "TON_DB" and at the same time overwrites the value of the operand "Tag_PT" within the data block. As a result, the signal state of the timer status can change at the next query or upon access to "MyTimer".Q or "MyTimer".ET.

Note

The "Tag_Input_2" is executed as pulse flag in order that the time duration is loaded only throughout one program cycle.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on FBD (Page 7185)

Legacy (S7-1500)

S_PULSE: Assign pulse timer parameters and start (S7-1500)

Description

The "Assign pulse timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as long as the signal state at input S is "1". If the signal state at input S changes to "0" before the programmed duration expires, the timer is stopped. In this case, the signal state at output Q is "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1", the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign pulse timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign pulse timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign pulse timer parameters and start" instruction:

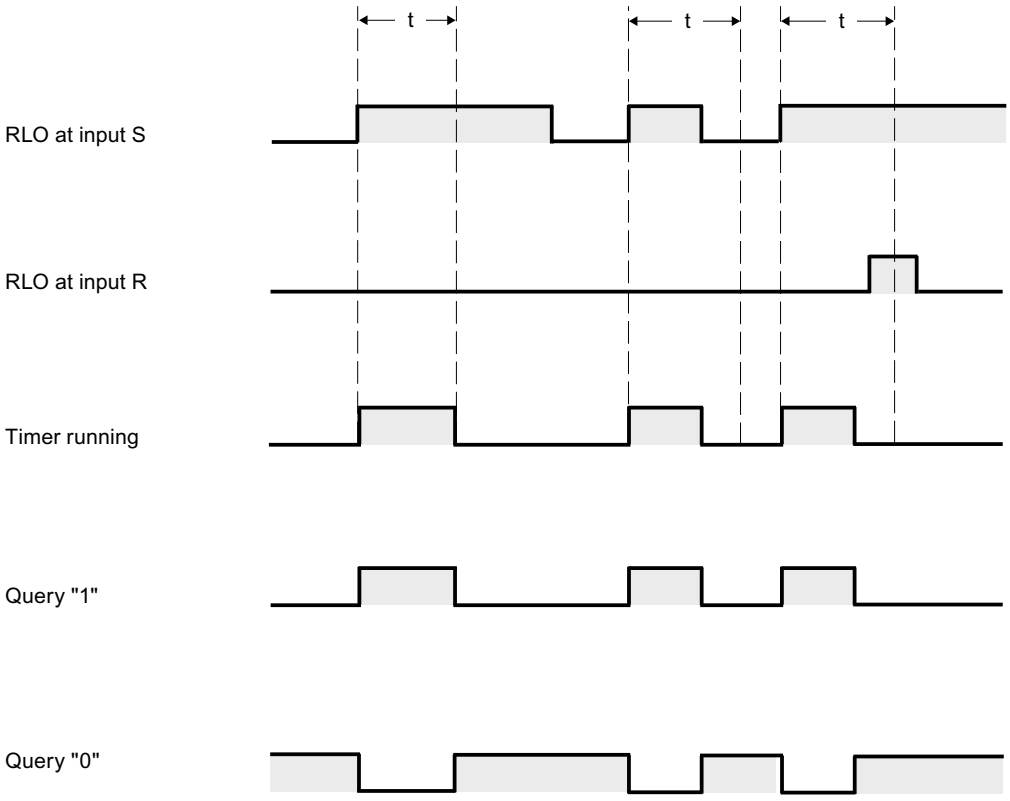

t = programmed time

Example

The following example shows how the instruction works:

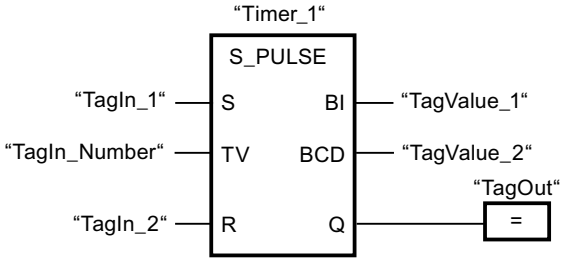

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer runs for the time value of the "TagIn_Number" operand as long as the "TagIn_1" operand has the signal state "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer "Timer_1" is stopped. The "TagOut" operand is reset to signal state "0".

The "TagOut" operand has the signal state "1" as long as the timer is running and the "TagIn_1" operand has the signal state "1". When the time has expired or is reset, the "TagOut" operand is reset to "0".

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on FBD (Page 7185)

S_PEXT: Assign extended pulse timer parameters and start (S7-1500)

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". As long as the timer is running, output Q has the signal state "1". When the timer has expired, output Q is reset to "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input TV.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1", the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign extended pulse timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign extended pulse timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:

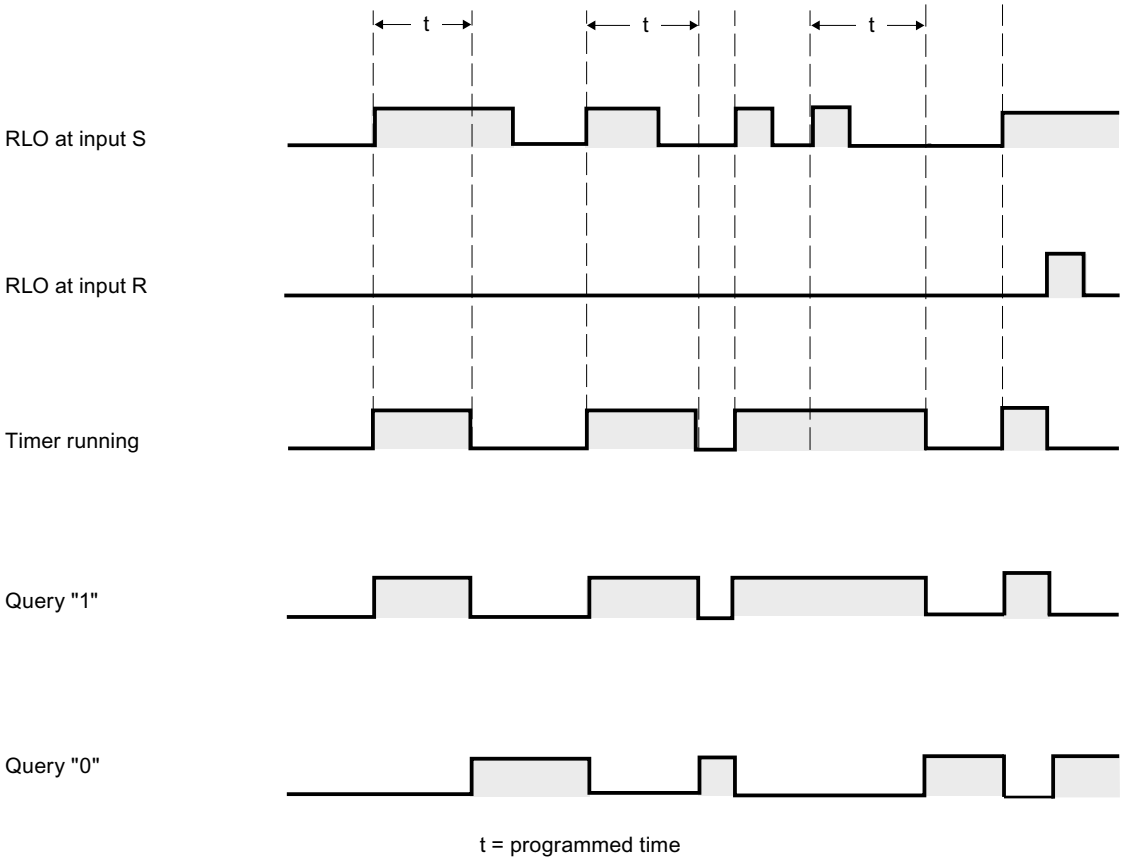

Example

The following example shows how the instruction works:

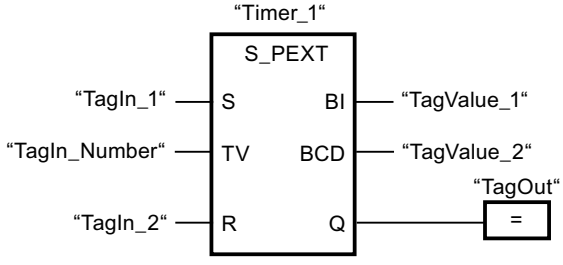

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer runs for the time value of the "TagIn_Number" operand without being affected by a negative edge at the S input. If the signal state at the "TagIn_1" operand changes from "0" to "1" before the timer expires, the timer is restarted.

The "TagOut" operand has the signal state "1" as long as the timer is running. When the time has expired or is reset, the "TagOut" operand is reset to "0".

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on FBD (Page 7185)

S_ODT: Assign on-delay timer parameters and start (S7-1500)

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as long as the signal state at input S is "1". If the timer expires correctly and input S still has signal state "1", output Q returns signal state "1". If the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the time is running and the signal state at input R changes from "0" to "1", the current time value and the time base are also set to zero. In this case, the signal state at output Q is "0". The timer is reset if the signal state is "1" at the R input even if the timer is not running and the RLO at input S is "1".

The "Assign on-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign on-delay timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:

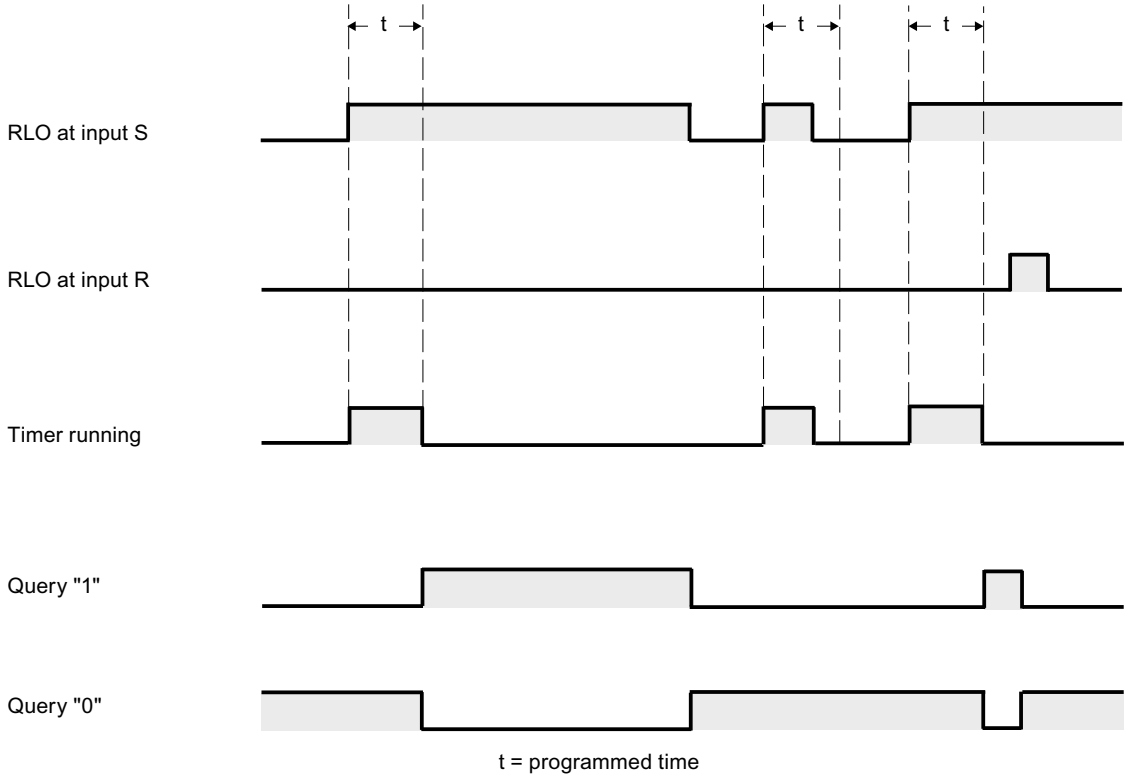

Example

The following example shows how the instruction works:

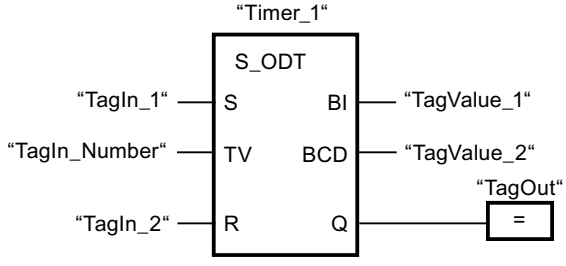

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". If the timer expires and the signal state of the operand is "1", the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. The "TagOut" operand has the signal state "0".

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on FBD (Page 7185)

S_ODTS: Assign retentive on-delay timer parameters and start (S7-1500)

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". If the timer expires, the "Q" output returns signal state "1" regardless of the signal state at input "S". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input (TV).

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0" regardless of the signal state at start input S. In this case, the signal state at output Q is "0".

The "Assign retentive on-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign retentive on-delay timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:

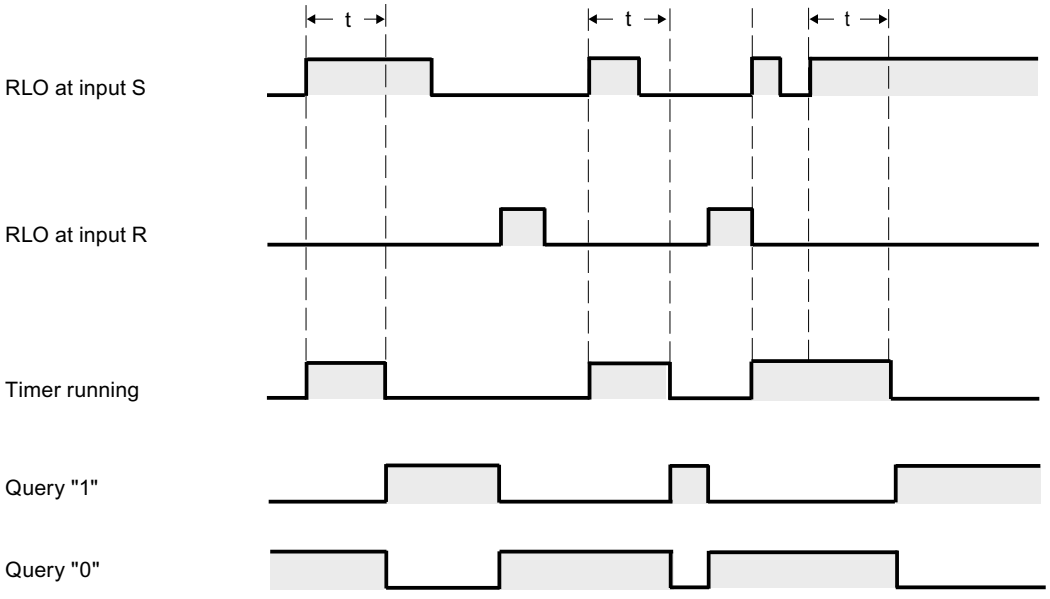

t = programmed time

Example

The following example shows how the instruction works:

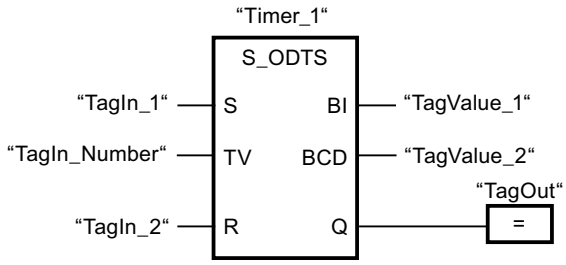

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand, even when the signal state of the "TagIn_1" operand changes to "0". When the time expires, the "TagOut" operand is reset to "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is restarted.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on FBD (Page 7185)

S_OFFDT: Assign off-delay timer parameters and start (S7-1500)

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a transition from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV). As long as the timer is running or input S returns signal state "1", output Q has signal state "1". If the timer expires and the signal state is "0", output Q is reset to signal state "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at input S.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0". In this case, the signal state at output Q is "0".

The "Assign off-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign off-delay timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:

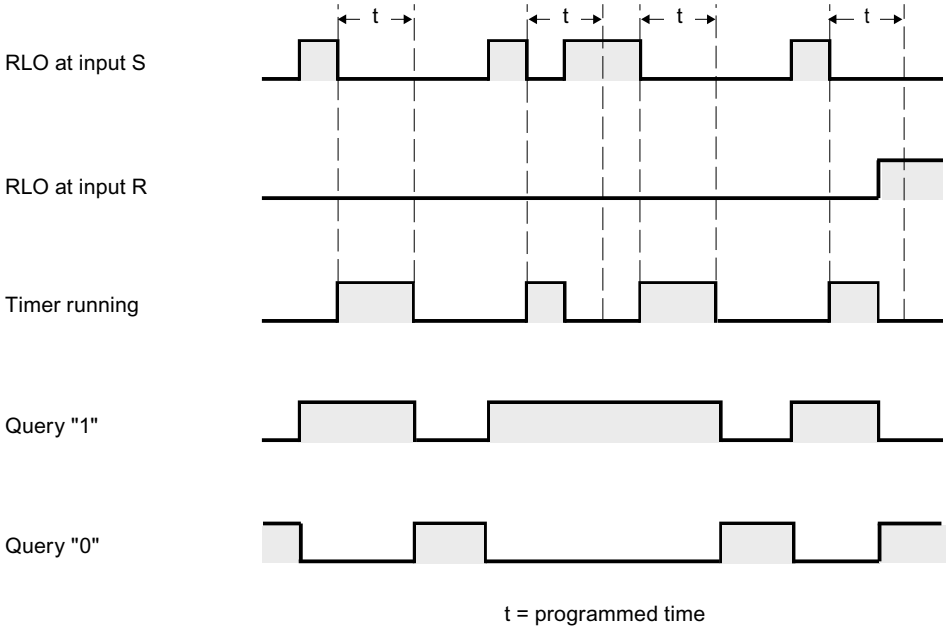

Example

The following example shows how the instruction works:

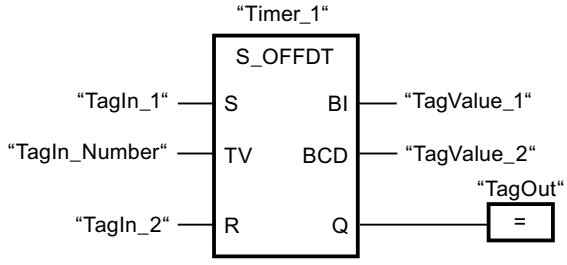

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "1" to "0". The timer expires with the value of operand "TagIn_Number". The "TagOut" operand is set to "1" if the timer is running or the "TagIn_1" operand has the signal state "0". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is reset.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on FBD (Page 7185)

SP: Start pulse timer (S7-1500)

Description

The instruction "Start pulse timer" starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs with the specified duration as long as the RLO has the signal state "1". As long as the timer is running, the query of timer status "1" returns the signal state "1". If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer stops. In this case, the query for timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction "Start pulse timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

Network 1:

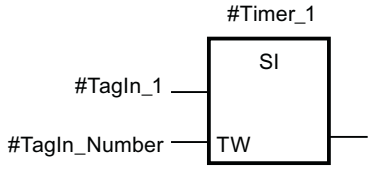

"#Timer_1" starts when the signal state of the "#TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "#TagIn_Number" operand as long as the signal state of the "#TagIn_1" operand is "1". If the signal state at the "#TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped.

Network 2:

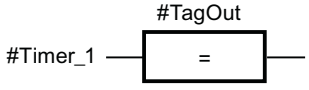

As long as the timer is running, the "#TagOut" operand returns signal state "1".

Network 3:

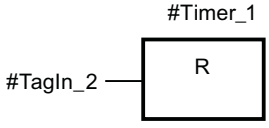

A signal state change of the operand "#TagIn_2" from "0" to "1" resets the timer, which stops the timer and sets the current time value to "0".

The figure below shows the timer diagram of the example:

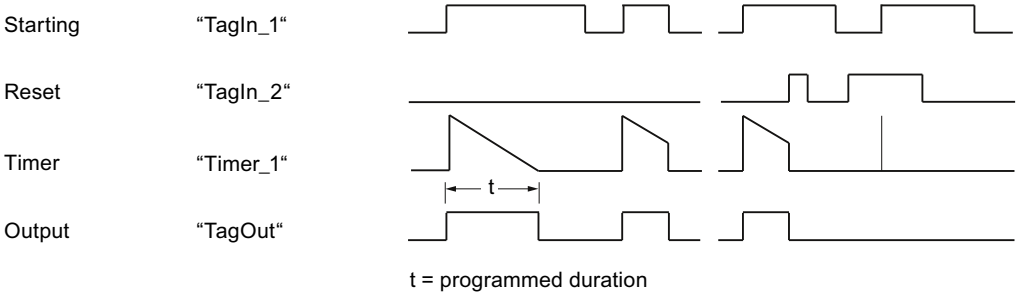

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basic information on FBD (Page 7185)

SE: Start extended pulse timer (S7-1500)

Description

The "Start extended pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period even when the RLO changes to signal state "0". As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is restarted with the programmed time period. The querying of the timer status for "1" returns the signal state "0" when the timer expires.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 1506)

Parameters

The following table shows the parameters of the "Start extended pulse timer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

Example

The following example shows how the instruction works:

Network 1:

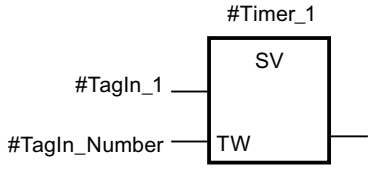

"#Timer_1" starts when the signal state of the "#TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the operand "#TagIn_Number" without being affected by a negative signal edge in the RLO.

Network 2:

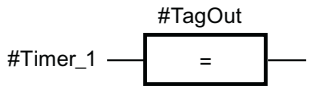

As long as the timer is running, the "#TagOut" operand returns signal state "1".

Network 3:

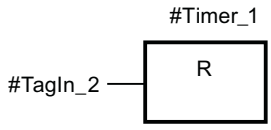

If the signal state of the operand "#TagIn_1" changes again from "0" to "1" before the timer expires, the timer is restarted.

The figure below shows the timer diagram of the example:

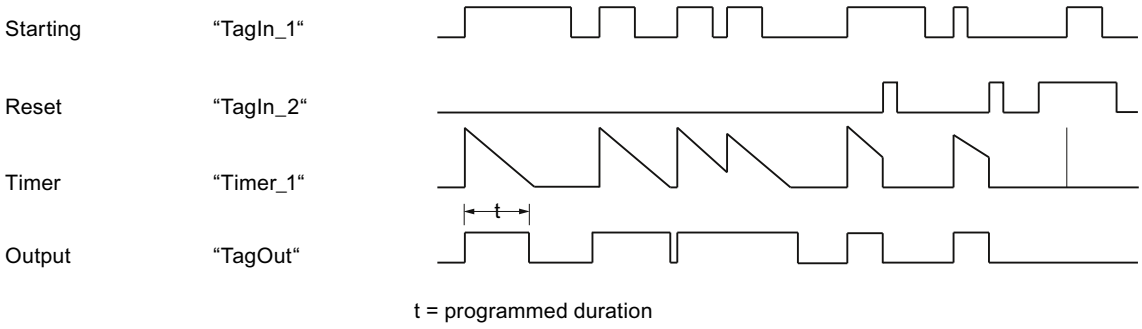

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

SD: Start on-delay timer (S7-1500)

Description

The "Start on-delay timer" instruction starts a programmed timer when the signal state "1" is detected at the start input. As long as this signal state remains "1", the timer expires with the specified duration. When the timer expires and the signal state at the start input is still "1", the query for timer status also returns "1". If the signal at the start input is "0", the timer is reset. In this case, querying the timer status returns the signal state "0". As soon as the signal at the start input changes to "1" again, the timer starts running again.

The signal state of the timer output is identical to the signal state of the start input. The start input is directly interconnected to the output and is not linked to the timer.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 1506)

Parameters

The following table shows the parameters of the instruction "Start on-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

Network 1:

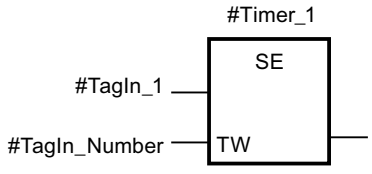

"#Timer_1" starts when the signal state of the "#TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "#TagIn_Number". If the signal state of the "#TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is reset.

Network 2:

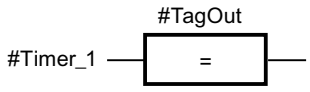

When the timer has expired, the #TagIn_1 operand at the start input has signal state "1" and the timer was not reset, the #TagOut operand is "1".

Network 3:

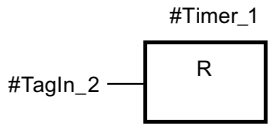

When the signal state of the #TagIn_2 operand has signal state "1", the timer #Timer_1 and the output #TagOut are reset.

If you want to restart #Timer_1, the signal state at the #TagIn_2 operand must be "0" and the signal state at the start input #TagIn_1 must change from "0" to "1".

The figure below shows the timer diagram of the example:

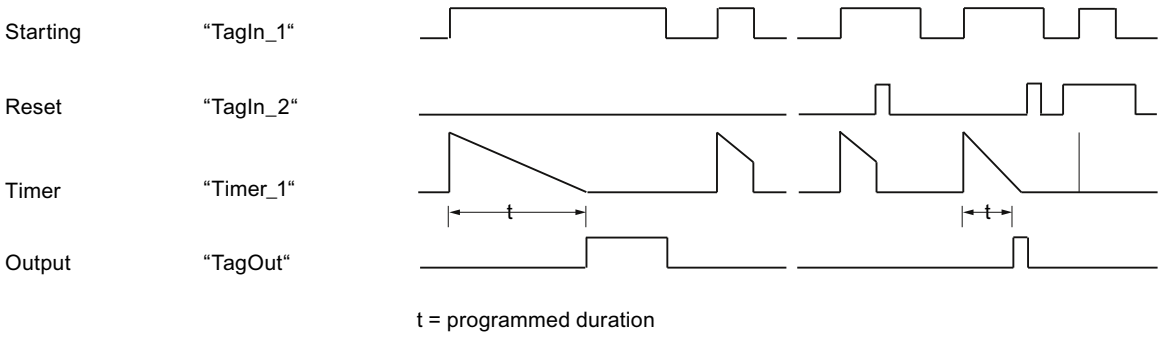

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

SS: Start retentive on-delay timer (S7-1500)

Description

The "Start retentive on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period even when the RLO changes to signal state "0". When the timer expires, the query for timer status for "1" returns the signal state "1". When the timer expires, the timer can only be restarted if it is explicitly reset.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start retentive on-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 1506)

Parameters

The following table shows the parameters of the instruction "Start retentive on-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

Network 1:

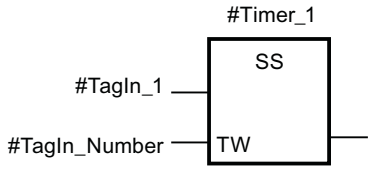

"#Timer_1" starts when the signal state of the "#TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "#TagIn_Number".

Network 2:

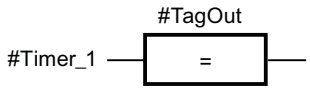

When the time expires, the "#TagOut" operand is reset to "1".

Network 3:

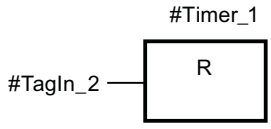

If the signal state at the "#TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is restarted. If the signal state of the operand "#TagIn_2" is "1", the timer "#Timer_1" is reset, which stops the timer and sets the current time value to "0".

The figure below shows the timer diagram of the example:

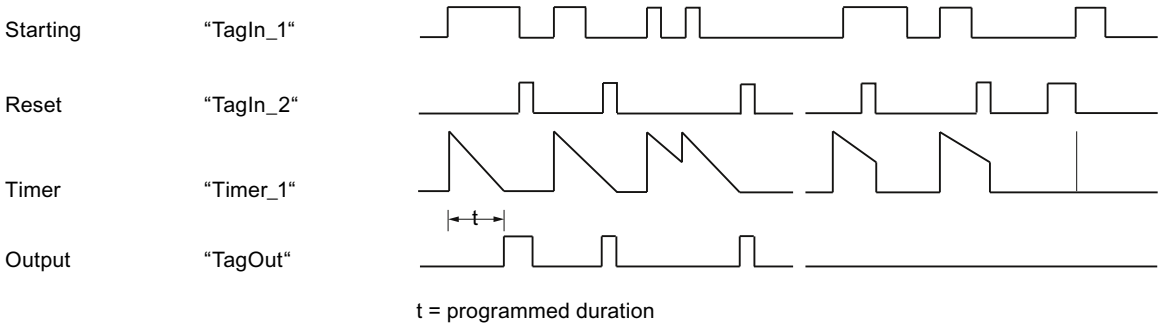

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

SF: Start off-delay timer (S7-1500)

Description

The "Start off-delay timer" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at the start input. The timer expires with the specified duration. As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is reset. The timer is always restarted when the RLO changes from "1" to "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The queries of the timer status for "1" return the query result "1" if the signal state of the result of logic operation is "1" at the execution of the instruction. If the RLO is "0", queries of the timer status for "1" return the query result "0".

The "Start off-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 1506)

Parameters

The following table shows the parameters of the instruction "Start off-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

Network 1:

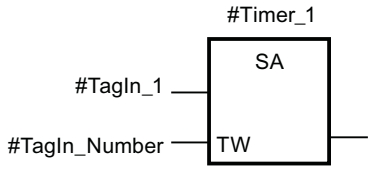

"#Timer_1" starts when the signal state of the "#TagIn_1" operand changes from "1" to "0". The timer expires with the value of operand "#TagIn_Number".

Network 2:

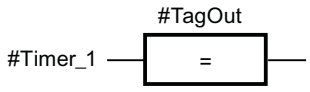

As long as the timer is running, the "#TagOut" operand is set to "1".

Network 3:

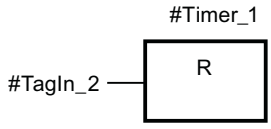

If the signal state at the "#TagIn_1" operand changes from "1" to "0" while the timer is running, the timer is restarted. If the signal state of the operand "#TagIn_2" is "1", the timer "#Timer_1" is reset, which stops the timer and sets the current time value to "0".

The figure below shows the timer diagram of the example:

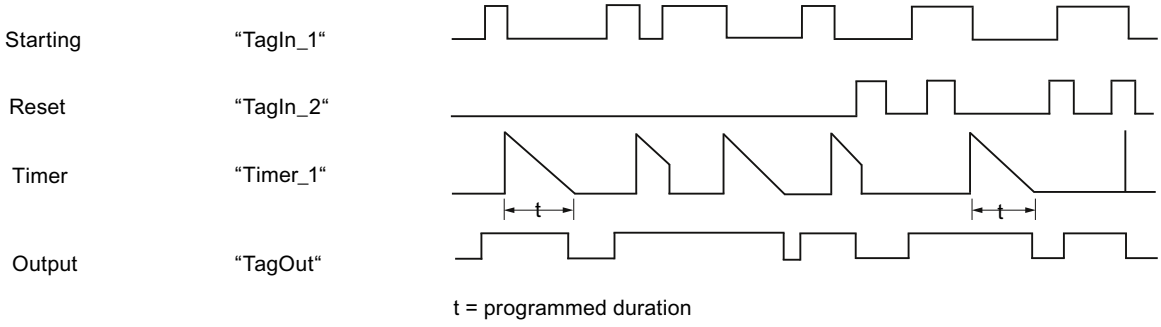

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

Counter operations

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the output CV. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can query the counter status in the Q output. The signal state at the Q output is determined by the PV parameter. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is reset to "0" and saved to an edge memory bit when the signal state at input R changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Reset input
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

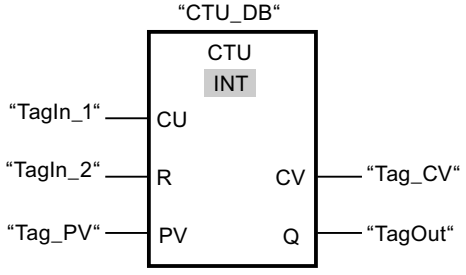

When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CV" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the specified data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is greater than

or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output returns the signal state "0".

See also

Overview of the valid data types (Page 231)

Setting retentivity in an instance data block (Page 7037)

Setting the retentivity of local tags (Page 6971)

Instances (Page 62)

Basic information on FBD (Page 7185)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can query the counter status in the Q output. If the current counter value is less than or equal to "0", the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is set to the value of the PV parameter and saved to a edge memory bit when the signal state at the LD input changes from "0" to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
LD	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

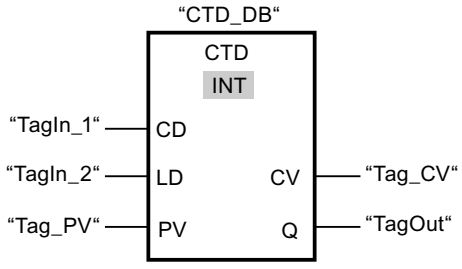

When the signal state of the "TagIn_1" operand changes from "0" to "1", the instruction is executed and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the counter value is decremented until the low limit of the specified data type (INT = -32768) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is less than or equal to "0". In all other cases, the "TagOut" output returns the signal state "0".

See also

Overview of the valid data types (Page 231)

Setting retentivity in an instance data block (Page 7037)

Setting the retentivity of local tags (Page 6971)

Instances (Page 62)

Basic information on FBD (Page 7185)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the current counter value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV output. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter and stored in a edge memory bit. As long as the LD input has signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The counter value is set to "0" and stored in an edge memory bit when the signal state at input R changes to "1". As long as the R input has signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0". You can also specify a constant for the PV parameter.

You can query the status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Load input

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value at which the QU output is set / value to which the CV output is set when LD = 1.
QU	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Up-counter status
QD	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Down-counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

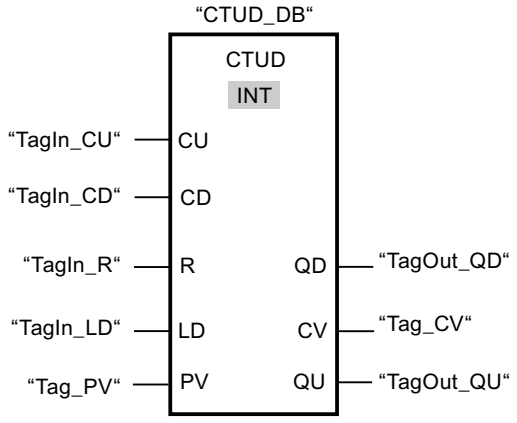

If the signal state at the "TagIn_CU" or "TagIn_CD" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_CU" input, the current counter value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_CD" input, the counter value is decremented by one and stored at the "Tag_CV" output. When there is a positive edge at the CU input, the counter value is incremented until it reaches the high limit (INT = 32767). If input CD has a positive signal edge, the counter value is decremented until it reaches the low limit (INT = -32768).

The "TagOut_GU" output has signal state "1" as long as the current counter value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut_QU" output returns the signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current counter value is less than or equal to "0". In all other cases, the "TagOut_QD" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Example of detecting the fill level of a storage area (Page 7236)

Setting retentivity in an instance data block (Page 7037)

Setting the retentivity of local tags (Page 6971)

Instances (Page 62)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

Legacy (S7-1500)

S_CU: Assign parameters and count up (S7-1500)

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The count is incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CU is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
<counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L, T, C or constant	Count up input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

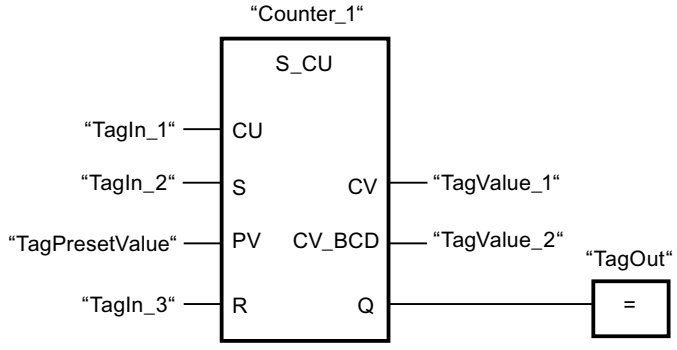

When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. When the signal state at the "TagIn_2" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

S_CD: Assign parameters and count down (S7-1500)

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The count is decremented until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CD is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count down" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input

Parameters	Declaration	Data type	Memory area	Description
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexa-decimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

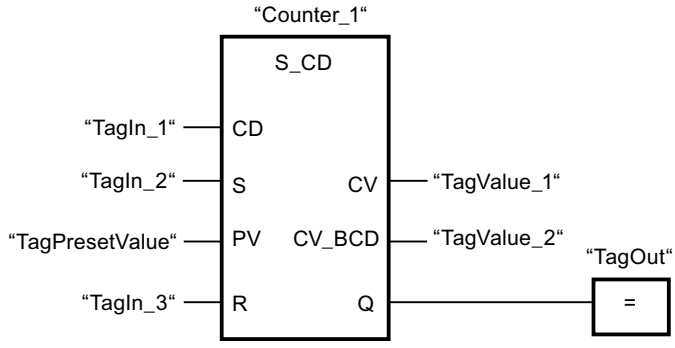

When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decremented by one. When the signal state at the "TagIn_2" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

S_CUD: Assign parameters and count up / down (S7-1500)

Description

You can use the "Assign parameters and count up / down" instruction to increment or decrement the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. If there is a positive signal edge at the CU and CD inputs in one program cycle, the counter value remains unchanged.

The counter value is incremented until the high limit of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of "0" is reached, the counter value is no longer decremented.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO is "1" at the inputs CU and CD, the counter counts accordingly in the next scan cycle, even if no change in the signal edge is detected.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU, CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up / down" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count up / down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L, T, C or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L, T, C or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)

Parameters	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexa-decimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

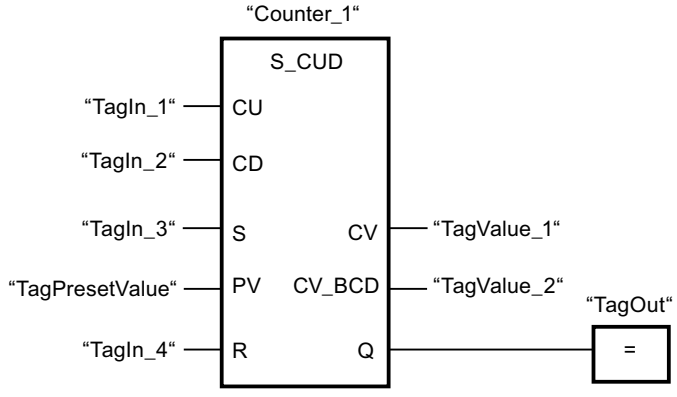

If the signal state at the input "TagIn_1" or at the input "TagIn_2" changes from "0" to "1" (positive edge), the instruction "Assign parameters and count up / down is executed. When there is a positive signal edge at the "TagIn_1" input and the current counter value is less than "999", the counter value is incremented by one. When there is a positive signal edge at the "TagIn_2" input and the current counter value is greater than "0", the counter value is decremented by one.

When the signal state at the "TagIn_3" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_4" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

SC: Set counter value (S7-1500)

Description

You can use the "Set counter value" instruction to set the value of a counter. The instruction is executed when the result of logic operation (RLO) at the start input of the instruction changes from "0" to "1". When the instruction is executed, the counter is set to the specified counter value.

The "Set counter value" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table lists the parameters of the "Set counter value" instruction:

Parameters	Declaration	Data type	Memory area	Description
<operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
PV	Input	WORD	I, Q, M, D, L or constant	Value with which the counter is preset in the BCD format. (C#0 to C#999)
<counter>	InOut/Input	COUNTER	C	Counter that is preset.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

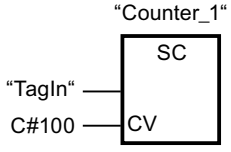

the counter "Counter_1" starts with the value "100" when the signal state of the "TagIn" operand changes from "0" to "1".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

CU: Count up (S7-1500)

Description

You use the "Count up" instruction to increment the value of the specified counter by the count of one on a rising edge at the start input. The count is incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

The "Count up" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
<operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
<counter>	InOut/Input	COUNTER	C	Counter whose value is incremented.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Network 1

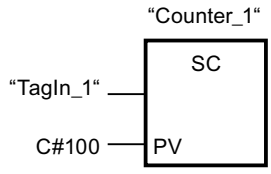

Network 2

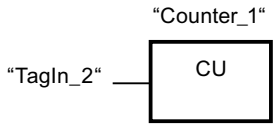

Network 3

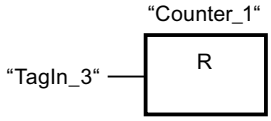

When the signal state of the operand "TagIn_1" changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of "Counter_1" is incremented by the count of one when the signal state of the "TagIn_2" operand changes from "0" to "1".

If the "TagIn_3" operand returns signal state "1", the value of "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

CD: Count down (S7-1500)

Description

You use the "Count down" instruction to decrement the value of the specified counter by the count of one on a rising edge at the start input. The count is decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer changed on a positive signal edge.

The "Count down" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
<counter>	InOut/Input	COUNTER	C	Counter whose value is decremented.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Network 1

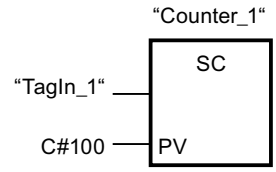

Network 2

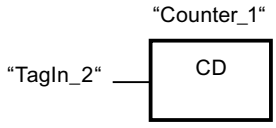

Network 3

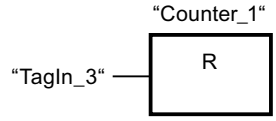

When the signal state of the operand "TagIn_1" changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of "Counter_1" is incremented by the count of one when the signal state of the "TagIn_2" operand changes from "0" to "1".

If the "TagIn_3" operand returns signal state "1", the value of "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to query whether the value at input IN1 is equal to the value at input IN2.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width of the operands must be the same.

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, then instead of the instruction "CMP ==: Equal" we recommend using the instruction "IN_RANGE: Value within range".

If floating-point numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

The special bit patterns of invalid floating-point numbers (NaN) that are the outcome of undefined results (e.g. root of -1) are not comparable. This means that if one of the operands has the value NaN, the instruction "CMP ==: Equal" has the result FALSE.

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table shows examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'aa'	'aaa'	0

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP ==: Equal" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP ==: Equal" will become inactive.

Result:

For the instruction "CMP ==: Equal" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP ==: Equal" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP ==: Equal" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

Bit patterns of invalid timers, date and times, e.g. DT#2015-13-33-25:62:99.999_999_999, cannot be compared. This means that if one of the operands has an invalid value, the instruction "CMP ==: Equal" has the result FALSE.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Comparison of hardware data types

To be able to compare the operands of the PORT data type, you need to select the WORD data type from the drop-down list in the instruction box.

If you want to compare the two data types, you must first create a tag of data type HW_ANY in the "Temp" section of the block interface and then copy the LADDR (from data type HW_DEVICE) to the tag. It is then possible to compare HW_ANY and HW_IO.

Comparison of structures

Note

Availability of comparison of structures

The option to compare structures is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

You can compare the values of two structured operands when both tags are of the same structure data type. When structures are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check. An exception is comparisons in which one of the two operands is a VARIANT or an ANY. If the data type is not yet known when the program is created, you can use the VARIANT data type. In this case you can also compare the operand with a structured tag of any data type. You can also compare two tags of the data type VARIANT or ANY with each other.

You can select the data type VARIANT for the comparison of structures from the drop-down list in the instruction box. Tags from the following data types are possible:

- PLC data type (UDT)
- STRUCT (either the structure of the STRUCT data type must be contained in a PLC data type or the two structures to be compared must be two elements of an ARRAY of STRUCT. Instance data blocks and tags of anonymous structures are not permitted.)
- Data blocks derived from a PLC data type.
- Tag to which ANY is pointing
- Tag to which VARIANT is pointing

The following requirements must be met to compare two tags of the data type ARRAY with VARIANT selected:

- The elements must each have the same data type.
- The two ARRAYs must have the same dimension.
- All dimensions must have the same number of elements. The exact ARRAY limits do not have to match.

Note

ARRAY of BOOL

When you compare two operands of the data type ARRAY of BOOL with each other, you have selected the data type VARIANT from the drop-down list, and the number of elements cannot be divided by 8, the fill bits are compared as well. This may affect the comparison result.

If an element within the structures that you want to compare is an invalid STRING/WSTRING, an invalid time or date or an invalid floating-point number, the comparison result in the RLO (result of logic operation) will return the signal state "0".

The table below shows an example of a comparison of structures:

<Operand1>			**<Operand2>**			**RLO of the instruction**
Tag of data type A <PLC data type>		Tag value	Tag of data type A <PLC data type>		Tag value	1
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

<Operand1>			**<Operand2>**			**RLO of the instruction**
Tag of data type A <PLC data type>		Tag value	Tag of data type B <PLC data type>		Tag value	0
	BOOL	FALSE		BOOL	TRUE	
	INT	2		INT	3	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	VARIANT (supplied with tag of data type A)		Tag value	1
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

Parameters

The following table shows the parameters of the instruction "Equal":

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time, ARRAY of <data type> with fixed and variable ARRAY limits, STRUCT, VARIANT, ANY, PLC data type	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time, ARRAY of <data type> with fixed and variable ARRAY limits, STRUCT, VARIANT, ANY, PLC data type	I, Q, M, D, L, P or constant	Second value to compare
As detailed above, the data types ARRAY, STRUCT (in a PLC data type), VARIANT, ANY and PLC data type (UDT) are only available as of firmware version 2.0 or 4.2.				

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

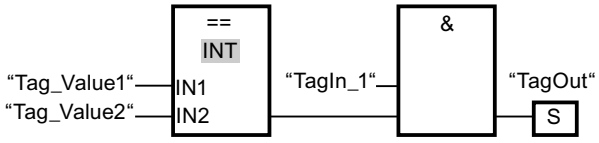

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" = "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to query whether the value at input IN1 is not equal to the value at input IN2.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width of the operands must be the same.

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, then instead of the instruction "CMP <>: Not equal" we recommend using the instruction "OUT_RANGE: Value outside range".

If floating-point numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

The special bit patterns of invalid floating-point numbers (NaN) that are the outcome of undefined results (e.g. root of -1) are not comparable. This means that if one of the operands has the value NaN, the instruction "CMP <>: Not equal" has the result FALSE.

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table shows examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0
'aa'	'aaa'	1

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP <>: Not equal" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP <>: Not equal" will become inactive.

Result:

For the instruction "CMP <>: Not equal" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP <>: Not equal" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP <>: Not equal" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

Bit patterns of invalid timers, date and times, e.g. DT#2015-13-33-25:62:99.999_999_999, cannot be compared. This means that if one of the operands has an invalid value, the instruction "CMP <>: Not equal" has the result FALSE.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Comparison of hardware data types

To be able to compare the operands of the PORT data type with the "Not equal" instruction, you need to select the WORD data type from the drop-down list of the instructions box.

If you want to compare the two hardware data types HW_IO and HW_DEVICE, you must first create a tag of data type HW_ANY in the section "Temp" of the block interface and then copy the LADDR (of data type HW_DEVICE) to the tag. It is then possible to compare HW_ANY and HW_IO.

Comparison of structures

Note

Availability of comparison of structures

The option to compare structures is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

You can compare the values of two structured operands when both tags are of the same structure data type. When structures are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check. An exception is comparisons in which one of the two operands is a VARIANT or an ANY. If the data type is not yet known when the program is created, you can use the VARIANT data type. In this case you can also compare the operand with a structured tag of any data type. You can also compare two tags of the data type VARIANT or ANY with each other.

You can select the data type VARIANT for the comparison of structures from the drop-down list in the instruction box. The following data types are possible:

- PLC data type
- STRUCT (the structure of the data type STRUCT must be contained in a PLC data type (UDT) or the two structures to be compared are two elements of an ARRAY of STRUCT. Anonymous structures are not permitted.)
- Tag to which ANY is pointing
- Tag to which VARIANT is pointing

The following requirements must be met to compare two tags of the data type ARRAY with VARIANT selected:

- The elements must each have the same data type.
- The two ARRAYs must have the same dimension.
- All dimensions must have the same number of elements. The exact ARRAY limits do not have to match.

Note

ARRAY of BOOL

When you compare two operands of the data type ARRAY of BOOL with each other, you have selected the data type VARIANT from the drop-down list, and the number of elements cannot be divided by 8, the fill bits are compared as well. This may affect the comparison result.

If an element within the structures that you want to compare is an invalid STRING/WSTRING, an invalid time or date or an invalid floating-point number, the comparison result in the RLO (result of logic operation) will return the signal state "0".

The table below shows an example of a comparison of structures:

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type A <PLC data type>		Tag value	0
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type B <PLC data type>		Tag value	1
	BOOL	FALSE		BOOL	TRUE	
	INT	2		INT	3	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	VARIANT (supplied with tag of data type A)		Tag value	0
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

Parameters

The following table shows the parameters of the "Not equal" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time, ARRAY of <data type> with fixed and variable ARRAY limits, STRUCT, VARIANT, ANY, PLC data type	I, Q, M, D, L, P, or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time, ARRAY of <data type> with fixed and variable ARRAY limits, STRUCT, VARIANT, ANY, PLC data type	I, Q, M, D, L, P, or constant	Second value to compare
As detailed above, the data types ARRAY, STRUCT (in a PLC data type), VARIANT, ANY and PLC data type (UDT) are only available as of firmware version 2.0 or 4.2.				

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

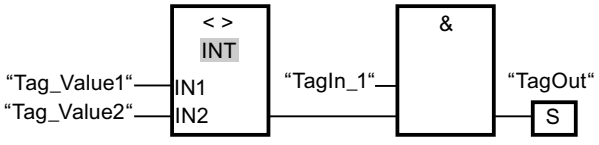

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <> "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to query whether the value at input IN1 is greater or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table shows examples of string comparisons:

IN1	IN2	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'Hello World'	'Hello World'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'AAA'	'a'	0

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >=: Greater or equal" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >=: Greater or equal" will become inactive.

Result:

For the instruction "CMP >=: Greater or equal" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >=: Greater or equal" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP >=: Greater or equal" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is greater (more recent) than or equal to the timer at input IN2.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Parameters

The following table shows the parameters of the "Greater or equal" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

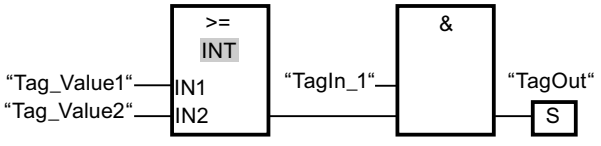

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" >= "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Example of detecting the fill level of a storage area (Page 7236)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to query whether the value at input IN1 is less or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table shows examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'Hello World'	'Hello World'	1
'HelloWorld'	'Hello World'	0
'BB'	'AA'	0
'AAA'	'AA'	0

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP <=: Less or equal" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP <=: Less or equal" will become inactive.

Result:

For the instruction "CMP <=: Less or equal" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP <=: Less or equal" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP <=: Less or equal" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is smaller (less recent) than or equal to the timer at input IN2.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Parameters

The following table shows the parameters of the "Less or equal" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

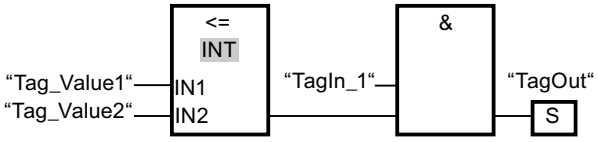

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <= "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

CMP >: Greater than

Description

You can use the "Greater than" instruction to query whether the value at input IN1 is greater than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table shows examples of string comparisons:

IN1	IN2	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'AA'	'aa'	0
'AAA'	'a'	0

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >: Greater than" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >: Greater than" will become inactive.

Result:

For the instruction "CMP >: Greater than" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >: Greater than" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP >: Greater than" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is greater (more recent) than the timer at input IN2.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Parameters

The following table shows the parameters of the "Greater than" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

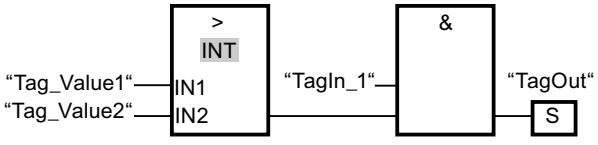

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" > "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

CMP <: Less than

Description

You can use the "Less than" instruction to query whether the value at input IN1 is less than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'BB'	'AA'	0
'AAA'	'AA'	0

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP <: Less than" (with data type STRING, WSTRING or VARIANT), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP <: Less than" will become inactive.

Result:

For the instruction "CMP <: Less than" (data type STRING, WSTRING or VARIANT), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP <: Less than" (data type STRING, WSTRING or VARIANT) is displayed. The instruction "CMP <: Less than" is shown as grayed out in the network in the inactive status.

Comparison of timers, date and time

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is less (less recent) than the timer at input IN2.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Parameters

The following table shows the parameters of the "Less than" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers, character strings, timers, date and time	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

Example

The following example shows how the instruction works:

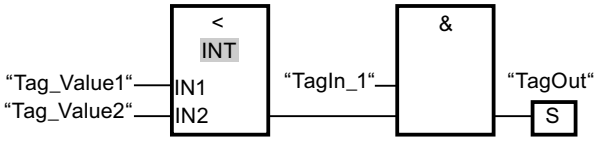

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" < "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Example of detecting the fill level of a storage area (Page 7236)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

IN_RANGE: Value within range

Description

You can use the "Value within range" instruction to query whether of the value at VAL input is within a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value within range" instruction compares the value at input VAL with the values of the inputs MIN and MAX and sends the result to the box output. If the value at input VAL satisfies the comparison MIN <= VAL or VAL <=MAX, the box output has the signal state "1". If the comparison is not fulfilled, the signal state is "0" at the box output.

The comparison function can only execute if the values to be compared are of the same data type.

Parameters

The following table shows the parameters of the "Value within range" instruction:

Parameters	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:

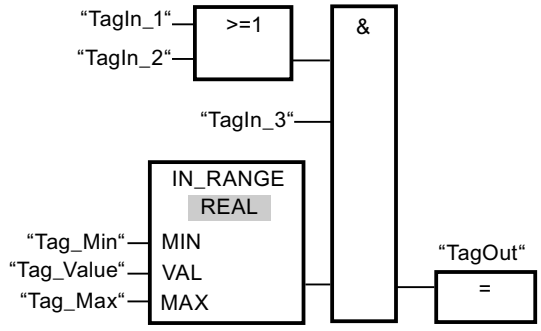

Output "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1" or "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".
- The value of the operand "Tag_Value" is within the value range that is specified by the current values of the operands "Tag_Min" and "Tag_Max" (MIN <= VAL or VAL <= MAX).

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

OUT_RANGE: Value outside range

Description

You can use the "Value outside range" instruction to query whether of the value at input VAL is outside a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value outside range" instruction compares the value at input VAL with the values of the inputs MIN and MAX and sends the result to the box output. If the value at input VAL satisfies the comparison MIN > VAL or VAL >MAX, the box output has the signal state "1". The box output has the signal state "1" if a specified operand of data type REAL has an invalid value.

The box output returns the signal state "0", if the value at input VAL does not satisfy the MIN > VAL or VAL > MAX condition.

The comparison function can only execute if the values to be compared are of the same data type.

Parameters

The following table shows the parameters of the "Value outside range" instruction:

Parameters	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

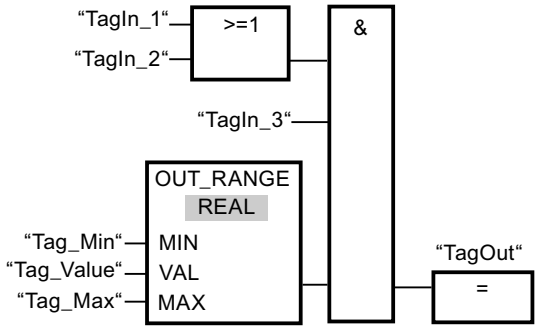

Output "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".
- The value of the operand "Tag_Value" is outside the value range that is specified by the values of the operands "Tag_Min" and "Tag_Max" (MIN > VAL or VAL > MAX).

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

OK: Check validity

Description

You can use the "Check validity" instruction to check if the value of an operand (<Operand>) is a valid floating-point number. The check is performed in every program cycle. If the operand value at the time of the query is a valid floating-point number, the output box will return the signal state"1". In all other cases, the signal state at the output of the "Check validity" instruction is "0".

You can use the "Check validity" instruction together with the EN mechanism. If you connect the instruction box to an EN enable input, the enable input is set only when the result of the validity query of the value is positive. You can use this function to ensure that an instruction is enabled only when the value of the specified operand is a valid floating-point number.

Parameters

The following example shows how the "Check validity" instruction works:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be checked.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:

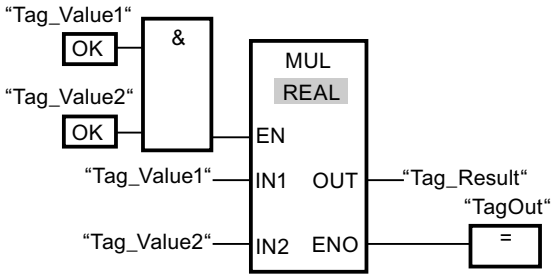

When the values of the operands "Tag_Value1" and "Tag_Value2" show valid floating-point numbers, the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied with the value of operand "Tag_Value2". The product of the multiplication is then written in the operand "Tag_Result". If the instruction is executed without errors, the TagOut operand at the "ENO" enable output is set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

NOT_OK: Check invalidity

Description

You can use the "Check invalidity" instruction to check if the value of an operand (<Operand>) is an invalid floating-point number. The check is performed in every program cycle. If the operand value at the time of the query is a valid floating-point number, then the output box will return the signal state "1". In all other cases, the signal state on the output box is "0".

Parameters

The following table shows the parameters of the instruction "Check invalidity":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be checked.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:

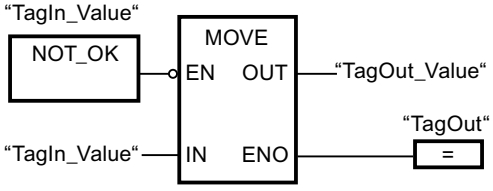

When the value of operand "TagIn_Value" is an invalid floating-point number, the instruction "Move value" is not executed. The "TagOut" operand at the ENO enable output is set to the signal state "0".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

VARIANT

EQ_Type: Compare data type for EQUAL with the data type of a tag

Description

You can use the "Compare data type for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Equal".

The tag at the IN1 parameter must have the VARIANT data type. The tag at the IN2 parameter can be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CMP ==". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CMP ==: Equal (Page 1037)

Parameters

The following table shows the parameters of the "Compare data type for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN1	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
IN2	Input	Bit sequences, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Second operand
OUT	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of comparison

Example

The following example shows how the instruction works:

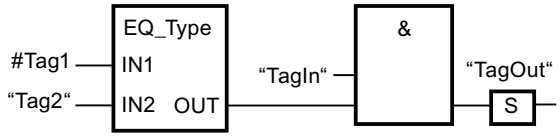

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag1 operand is equal to "Tag2".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

NE_Type: Compare data type for UNEQUAL with the data type of a tag

Description

You can use the "Compare data type for UNEQUAL with the data type of a tag" instruction to query the data type which a tag does not have to which a VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Not equal".

The tag at the IN1 parameter must have the VARIANT data type. The tag at the IN2 parameter can be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CMP <>". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CMP <>: Not equal (Page 1042)

Parameters

The following table shows the parameters of the "Compare data type for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN1	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
IN2	Input	Bit sequences, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Second operand
OUT	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of comparison

Example

The following example shows how the instruction works:

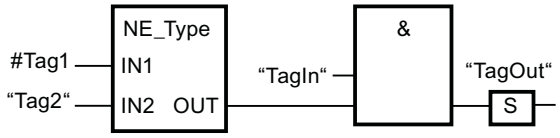

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag1 operand is not equal to "Tag2".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

EQ_ElemType: Compare data type of an ARRAY element for EQUAL with the data type of a tag

Description

You can use the "Compare data type of an ARRAY element for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Equal".

The tag at the IN1 parameter must have the VARIANT data type. The tag at the IN2 parameter can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CMP ==". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CMP ==: Equal (Page 1037)

Parameters

The following table shows the parameters of the "Compare data type of an ARRAY element for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN1	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
IN2	Input	Bit sequences, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Second operand
OUT	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of comparison

Example

The following example shows how the instruction works:

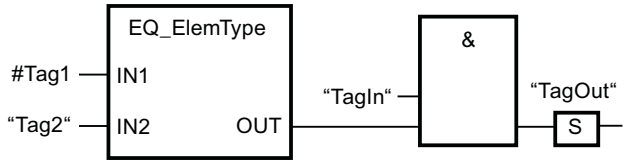

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag1 operand is equal to "Tag2".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

NE_ElemType: Compare data type of an ARRAY element for UNEQUAL with the data type of a tag

Description

You can use the "Compare data type of an ARRAY element for UNEQUAL with the data type of a tag" instruction to query which data type a tag does not have to which a VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Not equal".

The tag at the IN1 parameter must have the VARIANT data type. The tag at the IN2 parameter can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CMP <>". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CMP <>: Not equal (Page 1042)

Parameters

The following table shows the parameters of the "Compare data type of an ARRAY element for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN1	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
IN2	Input	Bit sequences, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Second operand
OUT	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of comparison

Example

The following example shows how the instruction works:

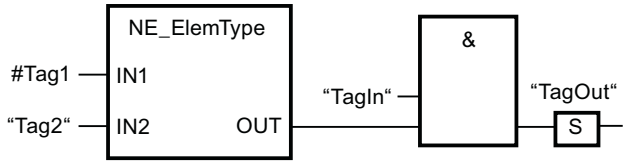

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag1 operand is not equal to "Tag2".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

IS_NULL: Check for EQUALS NULL pointer

Description

You can use the instruction "Check for EQUALS NULL pointer" to query whether the VARIANT or the reference points to a NULL pointer and therefore does not point to an object.

The <operand> must have the data type VARIANT or REF_TO <data type>.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "0" even if the ANY pointer is NULL.

Parameters

The following table shows the parameters of the "Check for EQUALS NULL pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<operand>	Input	VARIANT or REF_TO <data type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand that is compared for EQUALS NULL
OUT	Output	BOOL	I, Q, M, D, L	Result of comparison

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

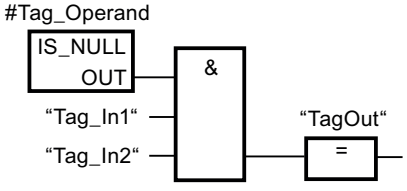

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "Tag_In1" and "Tag_In2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand operand does not point to an object.

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

NOT_NULL: Check for UNEQUALS NULL pointer

Description

You can use the instruction "Check for UNEQUALS NULL pointer" to query whether the VARIANT or the reference does not point to a NULL pointer and therefore points to an object.

The <operand> must have the data type VARIANT or REF_TO <data type>.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "1" even if the ANY pointer is NULL.

Parameters

The following table shows the parameters of the "Check for UNEQUALS NULL pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<operand>	Input	VARIANT or REF_TO <data type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand that is compared for UNEQUALS NULL
OUT	Output	BOOL	I, Q, M, D, L	Result of comparison

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

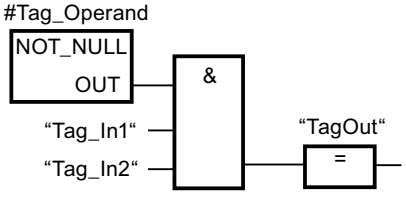

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "Tag_In1" and "Tag_In2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the #Tag_Operand operand points to an object.

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

IS_ARRAY: Check for ARRAY

Description

You can use the "Check for ARRAY" instruction to query whether the VARIANT points to a tag of the ARRAY data type.

The <operand> must have the VARIANT data type.

Parameters

The following table shows the parameters of the "Check for ARRAY" instruction:

Parameters	Declaration	Data type	Memory area	Description
<operand>	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand that is queried for ARRAY
OUT	Output	BOOL	I, Q, M, D, L	Result of comparison

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

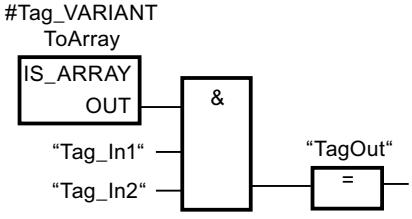

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "Tag_In1" and "Tag_In2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_VARIANTToArray" operand is of the ARRAY data type.

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

EQ_TypeOfDB: Compare data type of an indirectly addressed DB for EQUAL with a data type

Description

The "Compare data type of an indirectly addressed DB for EQUAL with a data type" instruction is used to query which data type of the data block has that the tag of the DB_ANY data type addresses. You compare the data type of the DB addressed by the tag at the IN1 parameter either with the data type of another tag or directly with a data type at the IN2 parameter for "Equal".

The tag at the IN1 parameter must have the DB_ANY data type. The tag at the IN2 parameter, for example, can be a PLC data type, a system data type, an axis or an FB.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN1	Input	DB_ANY	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
IN2	Input	TYPE_ID	I, Q, M, D, L	I, Q, M, D, L, P	Second operand
OUT	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of comparison

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

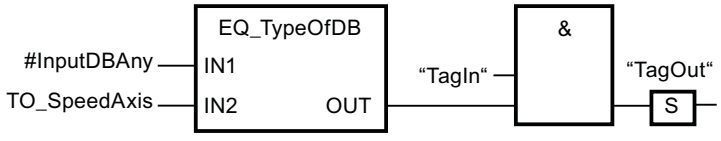

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the data type of the #InputDBAny operand addressed DB is equal with the TO_SpeedAxis data type.

The "TagOut" output is not set when the following conditions are fulfilled:

- The number of the data block is "0".
- The data block does not exist.
- The data block is an ARRAY DB.

See also

Overview of the valid data types (Page 231)

Using the DB_ANY data type (Page 204)

Basic information on FBD (Page 7185)

NE_TypeOfDB: Compare data type of an indirectly addressed DB for UNEQUAL with a data type

Description

The "Compare data type of an indirectly addressed DB for UNEQUAL with a data type" instruction is used to query which data type the data block does not have, that the tag of the DB_ANY data type addresses. You compare the data type of the DB addressed by the tag at the IN1 parameter either with the data type of another tag or directly with the data type at the IN2 parameter for "Not equal".

The tag at the IN1 parameter must have the DB_ANY data type. The tag at the IN2 parameter, for example, can be a PLC data type, a system data type, an axis or an FB.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN1	Input	DB_ANY	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		First operand
IN2	Input	TYPE_ID	I, Q, M, D, L	I, Q, M, D, L, P	Second operand
OUT	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of comparison

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

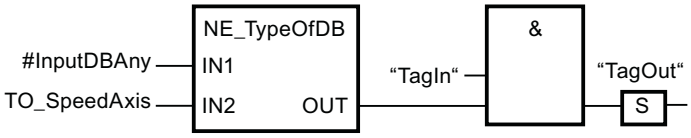

The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the data type of the #InputDBAny operand addressed DB is unequal with the TO_SpeedAxis data type.

The "TagOut" output is not set when the following conditions are fulfilled:

- The number of the data block is "0".
- The data block does not exist.
- The data block is an ARRAY DB.

See also

Overview of the valid data types (Page 231)

Using the DB_ANY data type (Page 204)

Basic information on FBD (Page 7185)

Math functions

CALCULATE: Calculate

Description

The "Calculate" instruction is used to define and execute an expression for the calculation of mathematical operations or complex logic operations depending on the selected data type.

You can select the data type of the instruction from the "???" drop-down list of the instruction box. Depending on the selected data type, you can combine the functionality of specific instructions to execute a complex calculation. The expression to be calculated is specified via a dialog you can open via the "Calculator" icon at the top of the instruction box. The expression can contain the names of the input parameters and the syntax of the instructions. It is not permitted to specify operand names or operand addresses.

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box.

The values of the inputs are use to execute the specified expression. Not all defined inputs have to be used in the expression. The result of the instruction is transferred to the box output OUT.

Note

If one of the mathematical operations fails in the expression, then no result is transferred to the OUT output and the ENO enable output has the signal state "1".

If, in the expression, you use inputs that are not available in the box, these inputs are automatically inserted. Provided that there are no gaps in the numbering of the inputs that are to be newly defined in the expression. You cannot, for example, use the input IN4 in the expression if the input IN3 is not defined.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result or an interim result of the "Calculate" instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.
- An error occurred during the execution of one of the instructions specified in the expression.

The following table shows the instructions that, depending on the selected data type, can be combined and executed in the expression of the "Calculate" instruction:

Data type	Instruction	Syntax	Example
Bit strings	AND: AND logic operation	AND	IN1 AND IN2 OR IN3
	OR: OR logic operation	OR	
	XOR: EXCLUSIVE OR logic operation	XOR	
	INV: Create ones complement	NOT	
	SWAP: Swap [1)]	SWAP	
Integers	ADD: Add	+	(IN1 + IN2) * IN3; (ABS(IN2)) * (ABS(IN1))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	MOD: Return remainder of division	MOD	
	INV: Create ones complement	NOT	
	NEG: Create twos complement	-(in1)	
	ABS: Form absolute value	ABS( )	

Data type	Instruction	Syntax	Example
Floating-point numbers	ADD: Add	+	((SIN(IN2) * SIN(IN2) + (SIN(IN3) * SIN(IN3)) / IN3)); (SQR(SIN(IN2)) + (SQR(COS(IN3)) / IN2))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	EXPT: Exponentiate	**	
	ABS: Form absolute value	ABS( )	
	SQR: Form square	SQR( )	
	SQRT: Form square root	SQRT( )	
	LN: Form natural logarithm	LN( )	
	EXP: Form exponential value	EXP( )	
	FRAC: Return fraction	FRAC( )	
	SIN: Form sine value	SIN( )	
	COS: Form cosine value	COS( )	
	TAN: Form tangent value	TAN( )	
	ASIN: Form arcsine value	ASIN( )	
	ACOS: Form arccosine value	ACOS( )	
	ATAN: Form arctangent value	ATAN( )	
	NEG: Create twos complement	-(in1)	
	TRUNC: Truncate numerical value	TRUNC( )	
	ROUND: Round numerical value	ROUND( )	
	CEIL: Generate next higher integer from floating-point number	CEIL( )	
	FLOOR: Generate next lower integer from floating-point number	FLOOR( )	
[1] Not possible for data type BYTE.			

Parameters

The following table shows the parameters of the instruction "Calculate":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First available input
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second available input

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
INn	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Additionally inserted inputs
OUT	Output	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Output to which the end result is to be transferred.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

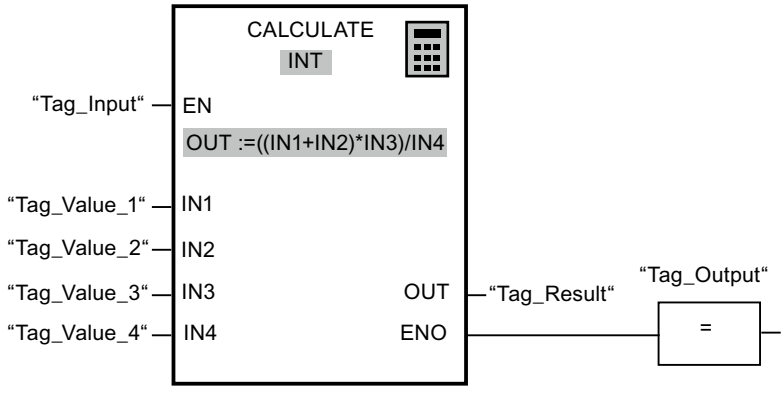

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value_1	4
IN2	Tag_Value_2	4
IN3	Tag_Value_3	3
IN4	Tag_Value_4	2
OUT	Tag_Result	12

If the "Tag_Input" has the signal state "1", the instruction is executed. The value of operand "Tag_Value_1" is added to the value of operand "Tag_Value_2". The sum is multiplied with the value of the operand "Tag_Value_3". The product is divided by the value of the operand "Tag_Value_4". The quotient is transferred as end result to the operand "Tag_Result" at the OUT output of the instruction. If no errors occur during the execution of the individual instructions, output ENO and the operand "Tag_Output" are set to "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Example of calculating an equation (Page 7240)

Basic information on FBD (Page 7185)

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 to the value at input IN2 and query the sum at output OUT (OUT := IN1+IN2).

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are added. The sum is stored at output OUT.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Add":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second number to be added
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values, which are added.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Sum

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:

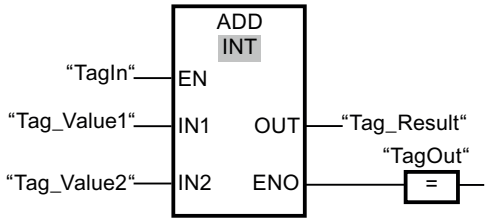

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at output OUT (OUT := IN1-IN2).

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Subtract":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Subtrahend
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Difference

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

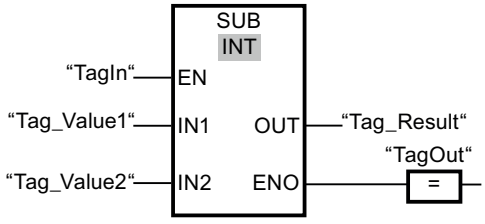

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at input IN1 with the value at input IN2 and query the total at output OUT (OUT := IN1*IN2).

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are multiplied. The product is stored at the OUT output.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Multiply":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for multiplication
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for multiplication
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values, which are multiplied.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Product

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

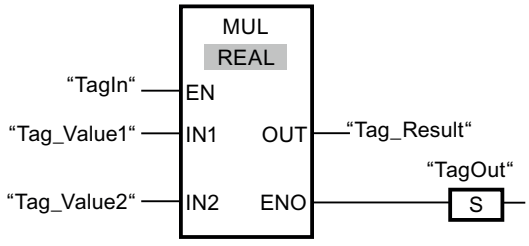

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is multiplied with the value of operand "Tag_Value2". The result of the multiplication is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT (OUT := IN1/IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Note

Division by the value "0"

The division of a dividend (IN1) by a divisor (IN2) with the value "0" sets the enable output (ENO) to signal state "TRUE".

On division by "0", the quotient value (OUT) is influenced by the data type as follows:

- The data type is INT or LREAL: Quotient value (OUT) is "0".
- The data type is REAL: Quotient value (OUT) is the maximum value (2143289344).

Parameters

The following table shows the parameters of the "Divide" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Dividend

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Quotient value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

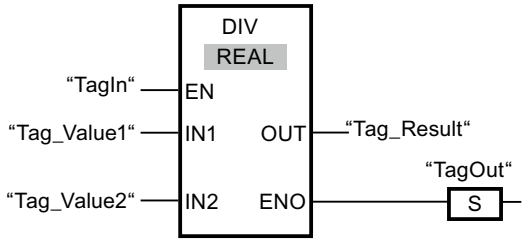

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The result of the division is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at input IN1 by the value at input IN2 and query the remainder of division at output OUT.

Parameters

The following table shows the parameters of the "Return remainder of division" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers	I, Q, M, D, L, P	I, Q, M, D, L, P	Remainder of division

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

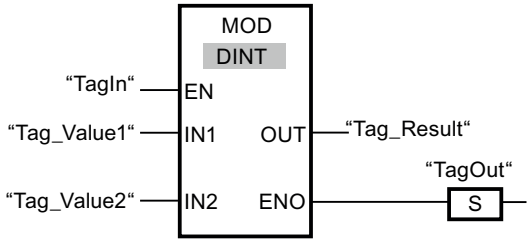

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder of division is stored in operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at input IN and query the result at output OUT. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

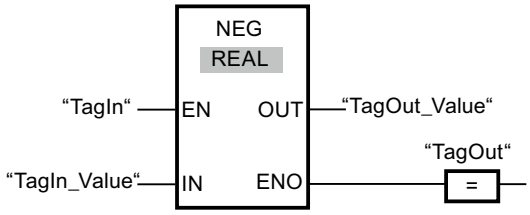

If the "TagIn" operand has signal state "1", the instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is stored at output "TagOut_Value". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

INC: Increment

Description

You can use the "Increment" instruction to change the value of the operand at parameter IN/OUT to the next higher value and query the result.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Increment" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	I, Q, M, D, L	Value to be incremented.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

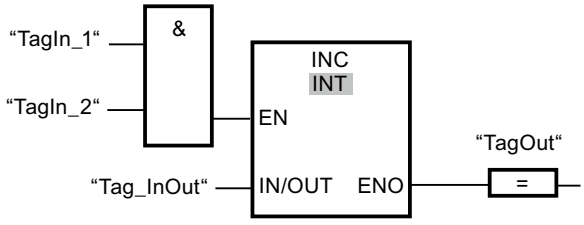

If the operands TagIn_1 and TagIn_2 have the signal state "1", the value of the operand "Tag_InOut" is incremented by one and the output "TagOut" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

DEC: Decrement

Description

You can use the "Decrement" instruction to change the value of the operand at parameter IN/OUT to the next lower value and query the result.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Decrement":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	I, Q, M, D, L	Value to be decremented.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

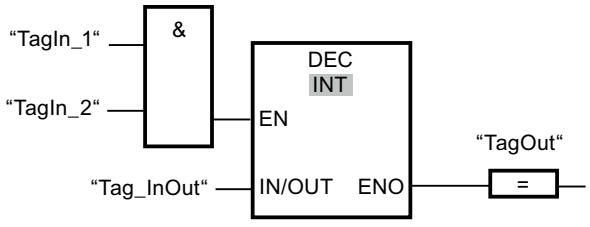

If the operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of the operand "Tag_InOut" is decremented by one and the output "TagOut" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

ABS: Form absolute value

Description

You can use the "Form absolute value" instruction to calculate the absolute value of the value specified at input IN. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Form absolute value":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Absolute value of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

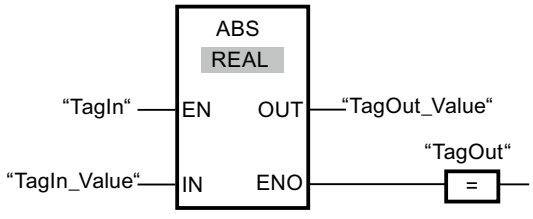

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the available inputs and writes the lowest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

In its initial state the instruction contains at least two inputs (IN1 and IN2) and no more than 100 inputs.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
INn	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Additionally inserted inputs whose values are to be compared
OUT	Output	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P	I, Q, M, D, L, P	Result
If the IEC check is not enabled, you can also use tags of the data type TIME, LTIME, TOD, LTOD, DATE and LDT by selecting a bit string of the same length as the data type. (e.g. instead of TIME => DINT, UDINT or DWORD = 32 bits)					

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

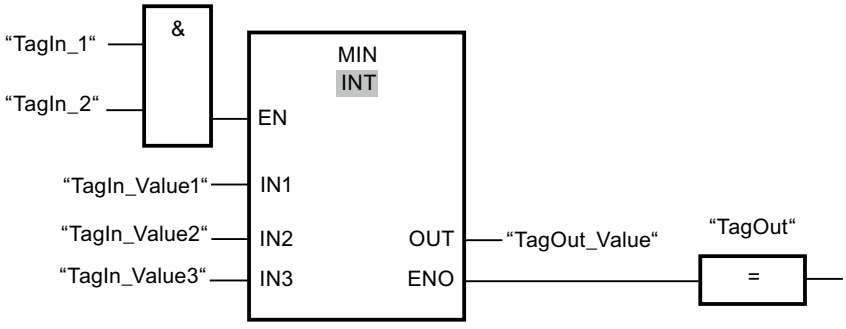

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the available inputs and writes the highest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

In its initial state the instruction contains at least two inputs (IN1 and IN2) and no more than 100 inputs.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
INn	Input	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Additionally inserted inputs whose values are to be compared.
OUT	Output	Integers, floating-point numbers, DTL, DT	I, Q, M, D, L, P	I, Q, M, D, L, P	Result
If the IEC check is not enabled, you can also use tags of the data type TIME, LTIME, TOD, LTOD, DATE and LDT by selecting a bit string of the same length as the data type. (e.g. instead of TIME => DINT, UDINT or DWORD = 32 bits)					

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

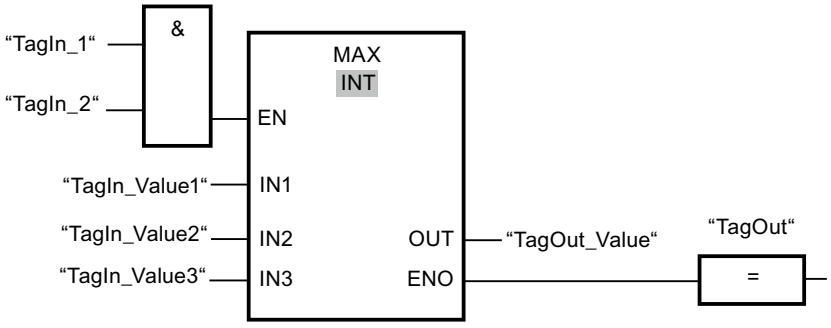

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN, output OUT is set to the value of the input MN. If the high limit MX is exceeded, output OUT is set to the value of the input MX.

If the value at the MN input is greater than at the MX input, the result is the value specified at the IN parameter and the enable output ENO is "0".

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN input is greater than the value at the MX input.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
MN	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL, DT	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DTL, DT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Low limit
IN	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL, DT	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DTL, DT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
MX	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL, DT	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DTL, DT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	High limit
OUT	Output	Integers, floating-point numbers, TIME, TOD, DATE, DTL, DT	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DTL, DT	I, Q, M, D, L, P	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.						

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

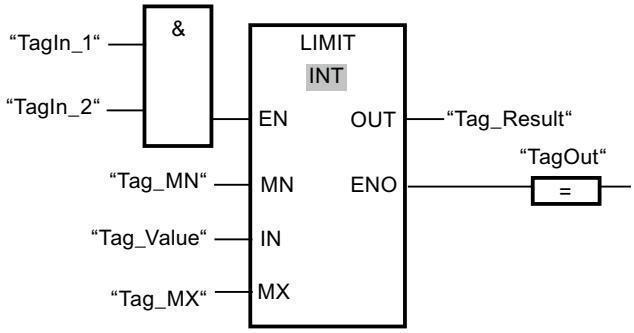

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value of the operand "Tag_Value" is less than the low limit, the value of the operand "Tag_MN" is copied to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

SQR: Form square

Description

The instruction "Form square" is used to square the value of a floating-point number at input IN and to write the result to output OUT.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form square":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Square of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

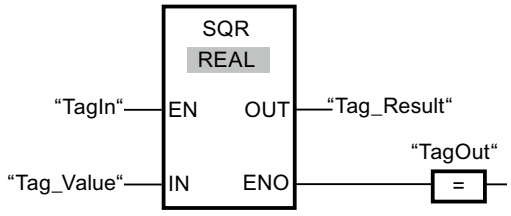

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction squares the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

SQRT: Form square root

Description

The instruction "Form square root" is used form the square root from the value of the floating-point number at input IN and to write the result to output OUT. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number. If the value at input IN is "0", the result is also "0".

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form square root":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	I, Q, M, D, L	Square root of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

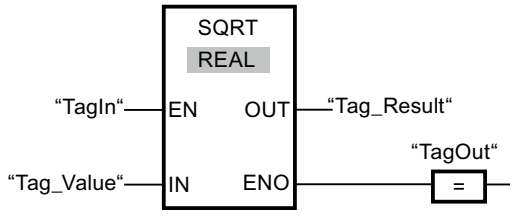

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the square root of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) of the value at input IN. The result is sent to output OUT and can be queried there. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form natural logarithm":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Natural logarithm of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

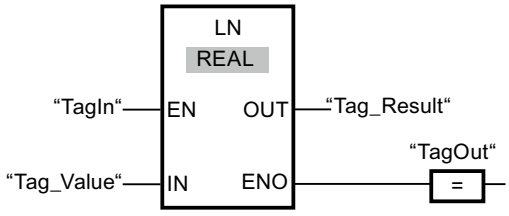

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction forms the natural logarithm of the value at input "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

EXP: Form exponential value

Description

You can use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value specified at input IN. The result is provided at output OUT and can be queried there (OUT = e^{IN}).

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form exponential value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Exponential value of input value IN

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

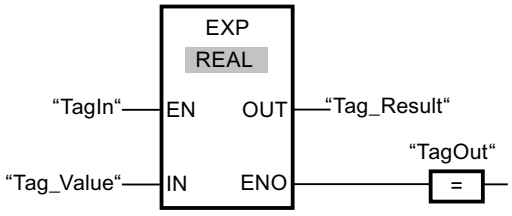

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction forms the exponent from base e and the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

SIN: Form sine value

Description

You can use the "Form sine value" instruction to calculate the sine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form sine value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Size of the angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Sine of the specified angle

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

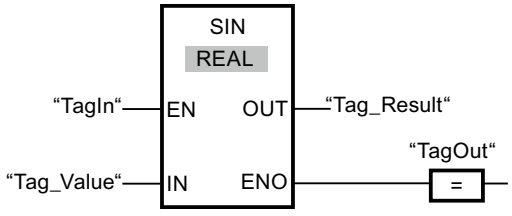

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the sine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

COS: Form cosine value

Description

You can use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form cosine value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Size of the angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Cosine of the specified angle

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

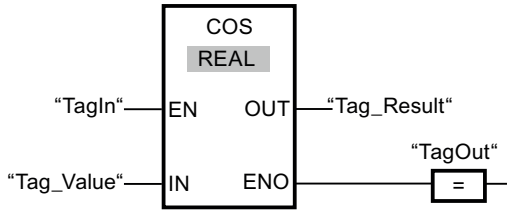

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the cosine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

TAN: Form tangent value

Description

You can use the "Form tangent value" instruction to calculate the tangent of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form tangent value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Tangent of the specified angle

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

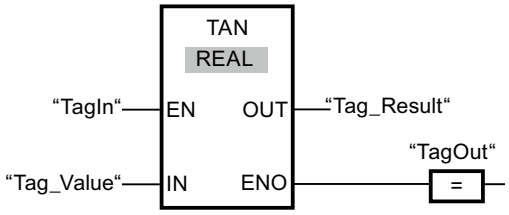

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

If the operand "TagIn" has the signal state "1", the "Form tangent value" instruction is executed. The instruction calculates the tangent of the angle specified at input "Tag_Value" and stores the result at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from the sine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

The following table shows the parameters of the instruction "Form arcsine value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Size of the angle in radians

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

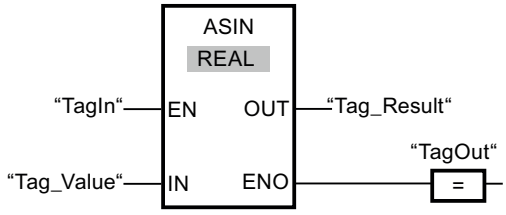

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from the cosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is output in radians at output OUT and can range in value from 0 to +π.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

The following table shows the parameters of the instruction "Form arccosine value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Size of the angle in radians

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

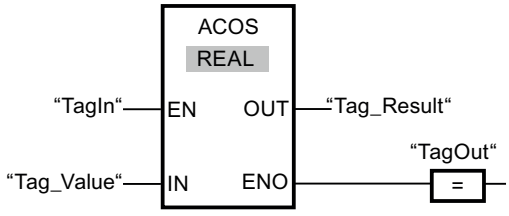

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from the tangent value specified at input IN, which corresponds to this value. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) at input IN. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form arctangent value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Tangent value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Size of the angle in radians

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

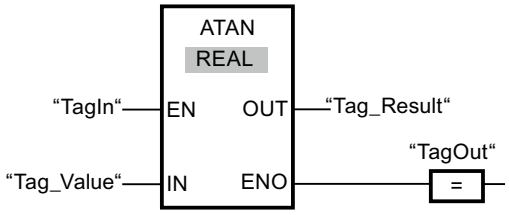

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398 (π/4)

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Invalid floating-point numbers (Page 253)

Basic information on FBD (Page 7185)

FRAC: Return fraction

Description

You can use the "Return fraction" instruction to determine the decimal places of the value at the IN input. The result of the query is stored at output OUT and can be queried there. If the input IN has e.g. the value 123.4567, the output OUT has the value 0.4567.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors occur during the execution of the instruction, for example there is no valid floating-point number at the input.

Parameters

The following table shows the parameters of the "Return fraction" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value, whose decimal places are to be determined.
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Decimal pla-ces of the input value at input IN

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

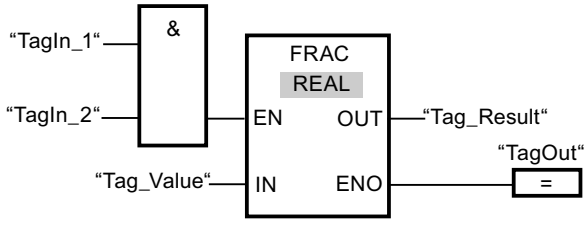

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	2.555
OUT	Tag_Result	0.555

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is started. The decimal places from the value at the operand "Tag_Value" are copied to the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

EXPT: Exponentiate

Description

You can use the "Exponentiate" instruction to raise the value at the input IN1 by a power specified with the value at input IN2. The result of the instruction is stored at output OUT and can be queried there (OUT = $IN1^{IN2}$).

The value at input IN1 must be a valid floating-point number. Integers are also allowed for setting the input IN2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors occur during the instruction processing, for example, if there is an overflow.

Parameters

The following table shows the parameters of the instruction "Exponentiate":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Base value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value with which the base value is exponentiated
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

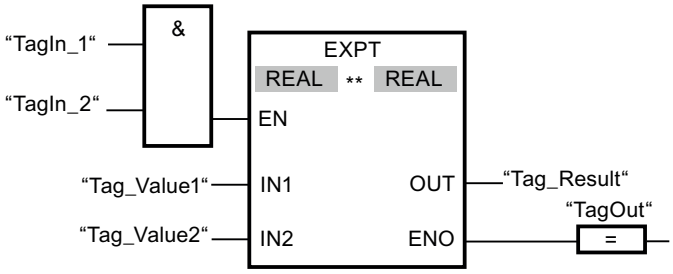

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Exponentiate" instruction is executed. The value of operand "Tag_Value1" is raised by the power of the value of the operand "Tag_Value2". The result is stored at output "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of ascending address.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data type at the IN parameter does not correspond to the specified data type at the OUT1 parameter.

The following table lists the possible transfers for the S7-1200 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE ,TOD, CHAR
WORD	WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD, CHAR

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
DWORD	DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
USINT	USINT, UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
INT	INT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UINT	UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UDINT	UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LREAL
TIME	TIME	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME
DATE	DATE	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, DATE
TOD	TOD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, CHAR, Character of a string[1)]
WCHAR	WCHAR	BYTE, WORD, DWORD, CHAR, WCHAR, character of a character string[1)]
Character of a string[1)]	Character of a string	CHAR, WCHAR, character of a string
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER

The following table lists the possible transfers for the S7-1500 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE ,TOD, LTOD, CHAR
WORD	WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, S5TIME, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
DWORD	DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
LWORD	LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LREAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
USINT	USINT, UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
INT	INT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UINT	UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD. LTOD
DINT	DINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UDINT	UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
LINT	LINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
ULINT	ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LWORD, LREAL
S5TIME	S5TIME	WORD, S5TIME
TIME	TIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME
LTIME	LTIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTIME
DATE	DATE	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, DATE
DT	DT	DT

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
LDT	LDT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LDT
TOD	TOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TOD
LTOD	LTOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, LWORD, CHAR, Character of a string[1)]
WCHAR	WCHAR	BYTE, WORD, DWORD, LWORD, CHAR, WCHAR, Character of a string[1)]
Character of a string[1)]	Character of a string	CHAR, WCHAR, character of a string
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
COUNTER	COUNTER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	TIMER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_LTIMER	IEC_LTIMER	IEC_LTIMER
IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNT-ER	IEC_LCOUNTER	IEC_LCOUNTER
IEC_UL-COUNTER	IEC_ULCOUNTER	IEC_ULCOUNTER
REF()	There is no data type conversion. The transferred tag must correspond exactly to the target data type.	

[1)] You can also use the "Move value" instruction to transfer individual characters of a string to operands of the CHAR or WCHAR data type. The number of the character to be transferred is specified in square brackets next to the operand name. "MyString[2]", for example, transfers the second character of the "MyString" string. The transfer of the operand of the data type CHAR or WCHAR to the individual character of a string is also possible. You can also replace a specific character of a string with the character of another string.

[2)] Transferring entire arrays (ARRAY) is possible only when the array components of the operands at input IN and at output OUT1 are of the same data type.

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at input IN is less than the bit length of the data type at output OUT1, the higher-order bits of the destination value will be overwritten with zeros.

In its initial state, the instruction box contains 1 output (OUT1). The number of outputs can be extended. The added outputs are numbered in ascending order on the box. During the execution of the instruction, the content of the operand at the input IN is transferred to all available outputs. The instruction box cannot be extended if structured data types (DTL, STRUCT, ARRAY) or characters of a string are transferred.

You can also use the "Move block" (MOVE_BLK) and "Move block uninterruptible" (UMOVE_BLK) instructions to move operands of the ARRAY data type. Operand of the data type STRING or WSTRING can be copied with the instruction "Move character string" (S_MOVE).

Parameters

The following table shows the parameters of the "Move value" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	Bit strings, integers, floating-point numbers, timers, date and time, CHAR, WCHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, timers, date and time, CHAR; WCHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data type, PLC data type (UDT)	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Element used to overwrite the destination address.
OUT1	Output	Bit strings, integers, floating-point numbers, timers, date and time, CHAR, WCHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, timers, date and time, CHAR; WCHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data type, PLC data type (UDT)	I, Q, M, D, L	I, Q, M, D, L	Destination address

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

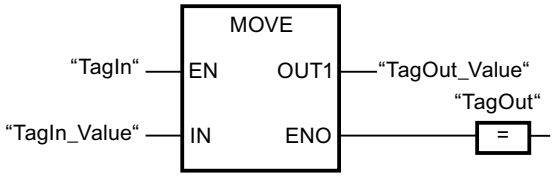

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If the "TagIn" operand returns signal state TRUE, the instruction is executed. The instruction moves the contents of the "TagIn_Value" operand to the "TagOut_Value" operand. If the instruction is executed without errors, the ENO and "TagOut" enable outputs are set to signal state TRUE.

Note

You can find more information on the MOVE instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Removing instruction inputs and outputs (Page 7219)

MOVE_BLK: Move block (Page 1125)

UMOVE_BLK: Move block uninterruptible (Page 1131)

S_MOVE: Move character string (Page 2124)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

Adding additional inputs and outputs to FBD elements (Page 7218)

Deserialize: Deserialize

Description

You can use the "Deserialize" instruction to convert back the sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type> and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type> is located must have the ARRAY of BYTE or ARRAY of CHAR data type and be declared with standard access in version 1.0. Optimized memory areas also permitted as of version 2.0. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion. If the memory area is filled using the instruction "Serialize" and it is possible that fill bytes were inserted, their conversion is not taken into account.

It is recommended to define the low limit of the ARRAY with "0", because then the index within ARRAY corresponds to the value of the POS parameter, e.g. ARRAY[0] = POS 0. The description and example below have been formed based on this.

The instruction enables you to convert multiple sequential representations of converted data back to their original state step-by-step.

If you only want to convert back a single sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type>, you can also directly use the instruction "TRCV: Receive data via communication connection".

Size of the memory area

The alignment rules mean that simple structures in the optimized memory area do not contain filling bytes. This results in a structure in the optimized memory area being smaller than in the standard memory area. ARRAYs of structures and structures in which other structures are located contain filling bytes. There is no general rule as to the memory area in which a composed structure requires more space.

Applies to CPUs of the S7-1500 series:

For a block with the block property "Optimized block access", the BOOL has a length of 1 byte. A structure that consists largely of the data type BOOL can therefore be larger in the optimized memory area than in the standard memory area. Composed structures with a low proportion of BOOL data types are smaller in the optimized memory area than in the standard memory area.

Optimized memory area

To deserialize larger structures, you can declare the memory area for sequential representation with optimized access for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. The content of the sequential representation remains unchanged, as for a standard memory area. Only symbolic access to the bytes in the ARRAY is possible.

Parameters

The following table shows the parameters of the "Deserialize" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_ARRAY	Input	VARIANT ARRAY[*] of BYTE [1)]	D, block interface of an FB (the sections Input, Output, Static and Temp are possible.)	Pointer to an ARRAY of BYTE or ARRAY of CHAR in which the data stream is saved.
DEST_VARIABLE	InOut	VARIANT	D, block interface of an FB	Pointer to a tag of the STRUCT, ARRAY or PLC data type (UDT) to be deserialized.

Parameters	Declaration	Data type	Memory area	Description
POS	InOut	DINT	I, Q, M, D, L	The operand at the POS parameter stores the index of the first byte based on the number of bytes occupied by the converted customer data. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information
[1] Possible for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The tag at the SRC_ARRAY parameter is not in a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains a ZERO pointer.
8151	No valid reference at the SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8154	Invalid data type at the SRC_ARRAY parameter
8250	A NULL pointer was transferred at parameter DEST_ARRAY.
8251	No valid reference at the DEST_VARIABLE parameter
8382	The value at the POS parameter is outside the limits of the array.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

For a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0, the following error code has a different meaning:

Error code* (W#16#...)	Explanation
8136	Access to the memory area at the SRC_ARRAY parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface of an FB or FC.
BufferPos	DINT	In the "Temp" section of the block interface of an FB or FC.
Error	INT	
Label	STRING[4]	

The following table shows the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

The following example shows how the instruction works:

Network 1:

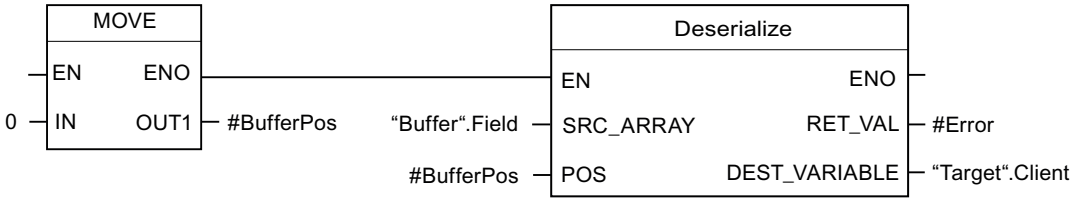

The "Move value" instruction moves the value "0" to the #BufferPos operand. The "Deserialize" instruction deserializes the sequential representation of the customer data from the "Buffer" tag and writes it to the "Target" tag. The #BufferPos operand stores the index of the first byte based on the number of bytes occupied by the converted customer data.

Network 2:

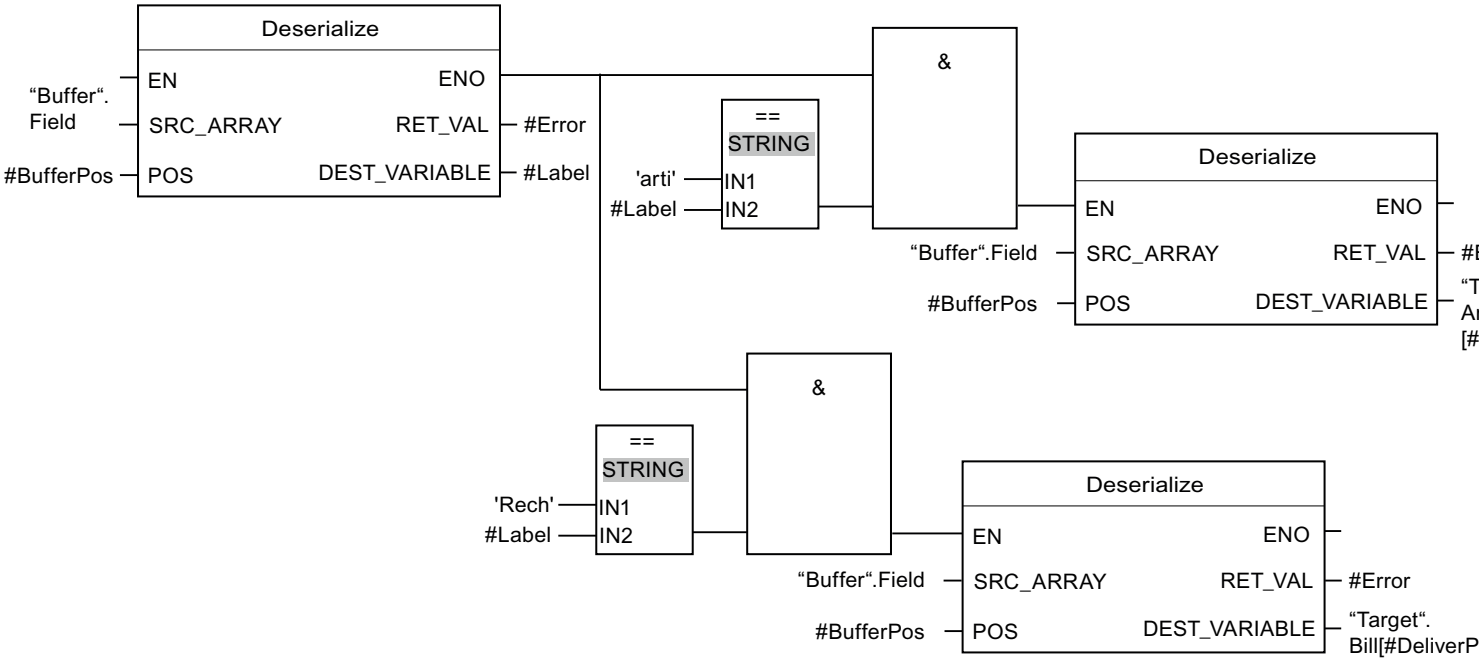

The "Deserialize" instruction deserializes the sequential representation of the separator sheet, which was stored in the sequential representation after the customer data, from the "Buffer" tag and writes the characters to the #Label operand. The characters are compared using comparison instructions "arti" and "Bill". If the comparison for "arti" = TRUE, these are article data that have been deserialized and written to the "Target" tag. If the comparison for "Bill" = TRUE, these are billing data that have been deserialized and written to the "Target" tag.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on VARIANT (Page 324)

Basics of PLC data types (UDT) (Page 273)

Structure of an ARRAY tag (Page 287)

Structure of a STRUCT tag (Page 280)

Structure of a STRING tag (Page 266)

Basic information on FBD (Page 7185)

Serialize: Serialize

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT), STRUCT or ARRAY of <data type> to a sequential representation without losing parts of their structure.

You use the instruction to temporarily save multiple structured data items from your program in a buffer, which should preferably be in a global data block, and send them to another CPU. The memory area in which the converted data is stored must have the ARRAY of BYTE or ARRAY of CHAR data type and be declared with standard access in version 1.0. Optimized data is also permitted as of version 2.0. Fill data of the source data area is undefined in the target array. These can be fill bytes or fill bits of a data area (e.g. ARRAY, STRUCT or PLC data type (UDT)) as well as the characters of a string currently not in use.

The capacity of the standard memory area is 64 KB. Structures that are larger than 64 KB in accordance with the rules for standard memory areas cannot be serialized if the operand at the DEST_ARRAY parameter is located in a standard memory area.

It is recommended to define the low limit of the ARRAY with "0", because then the index within ARRAY corresponds to the value of the POS parameter, e.g. ARRAY[0] = POS 0. The description and example below have been formed based on this.

The operand at the POS parameter contains information about the number of bytes used by the converted data.

If you want to send a single PLC data type (UDT), STRUCT or ARRAY of <data type>, you can directly call the instruction "TSEND: Send data via communication connection".

Note

Comparison of structures

To compare structures, they have to be serialized first. Use the "CMP" instruction instead.

You can find additional information in the section "Comparator".

Size of the memory area

The alignment rules mean that simple structures in the optimized memory area do not contain filling bytes. This results in a structure in the optimized memory area being smaller than in the standard memory area. ARRAYs of structures and structures in which other structures are located contain filling bytes. There is no general rule as to the memory area in which a composed structure requires more space.

Applies to CPUs of the S7-1500 series:

For a block with the block property "Optimized block access", the length of the data type BOOL depends on the data type that follows. This means if a BYTE follows, for example, the data type BOOL has a length of 1 byte. When a WORD follows, for example, the data type BOOL has a length of 2 bytes. A structure that consists largely of the data type BOOL can therefore be larger in the optimized memory area than in the standard memory area. Composed structures with a low proportion of BOOL data types are smaller in the optimized memory area than in the standard memory area.

We therefore recommend that the source data area for the serialization starts with the large data types and ends with Boolean elements. This greatly reduces the filling with filling bits.

Optimized memory area

To serialize larger structures, you can declare the memory area with optimized access for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. The sequential representation remains unchanged, as for a standard memory area.

Parameters

The following table shows the parameters of the instruction "Serialize":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_VARIABLE	Input	VARIANT	D, block interface of an FB	Pointer to a tag of the STRUCT, ARRAY or PLC data type (UDT) to be serialized.
DEST_ARRAY	InOut	VARIANT	D, block interface of an FB (the sections Input, Output, Static and Temp are possible.)	Pointer to an ARRAY of BYTE or ARRAY of CHAR in which the generated data stream is saved.
POS	InOut	DINT	I, Q, M, D, L	The operand at the POS parameter stores the index of the first byte based on the total number of bytes that the converted customer data has occupied. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains a ZERO pointer.
8151	No valid reference at the SRC_VARIABLE parameter
8236	The tag at the SRC_ARRAY parameter is not in a block with standard access.
8250	A NULL pointer was transferred at parameter DEST_ARRAY
8251	No valid reference at the DEST_ARRAY parameter

Error code* (W#16#...)	Explanation
8253	The tag at the parameter DEST_ARRAY does not provide enough memory space for the content of the tag at the parameter SRC_VARIABLE. Due to the input value of the tag at parameter POS the available memory space is reduced. The input value of the parameter POS decides where to begin within the tag at the parameter DEST_ARRAY.
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at the POS parameter is outside the limits of the array.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

For a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0, the following error code has a different meaning:

Error code* (W#16#...)	Explanation
8236	Access to the memory area at the DEST_ARRAY parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

Network 1:

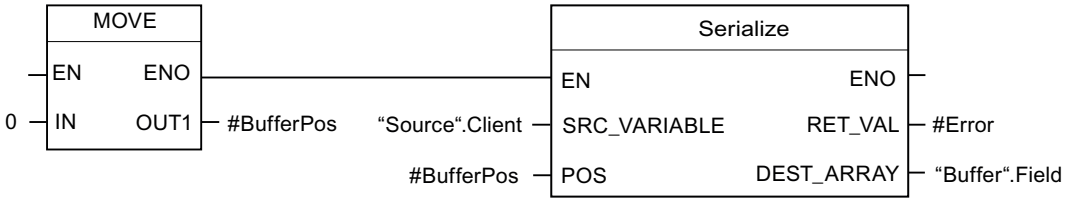

The "Move value" instruction moves the value "0" to the #BufferPos operand. The "Serialize" instruction serializes the customer data from the "Source" tag and writes the sequential representation to the "Buffer" tag. The index of the next unwritten byte of the "Buffer".Field operand is saved in the #BufferPos operand.

Network 2:

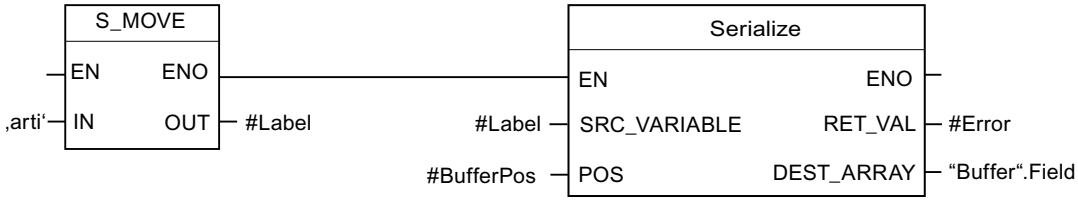

A kind of separator sheet is inserted now to make it easier to deserialize the sequential representation later. The "Move character string" instruction moves the "arti" characters to the #Label operand. The "Serialize" instruction writes these characters after the customer data to the "Buffer" tag. The value of the #BufferPos operand is incremented accordingly.

Network 3:

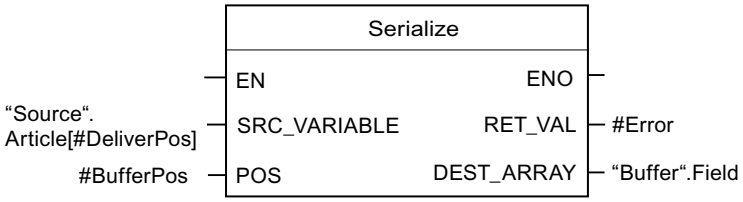

The "Serialize" instruction serializes the data of a specific article, which is calculated in runtime, from the "Source" tag and writes it in sequential representation to the "Buffer" tag after the "arti" characters.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table shows the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on VARIANT (Page 324)

Basics of PLC data types (UDT) (Page 273)

Structure of an ARRAY tag (Page 287)

Structure of a STRUCT tag (Page 280)

Structure of a STRING tag (Page 266)

Basic information on FBD (Page 7185)

MOVE_BLK: Move block

Description

You use the "Move block" instruction to move the content of a memory area (source range) to another memory area (target range). The number of elements to be moved to the target range is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source range and the target range have the same data type.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Note

Use of ARRAYs

The instruction copies the contents from the defined element n elements (n = depending on the value at the parameter COUNT) from the source range to the target range, starting at the specified index.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Number of elements to be copied from the source range to the target range
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	The first element of the target range to which the contents of the source range are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.						

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

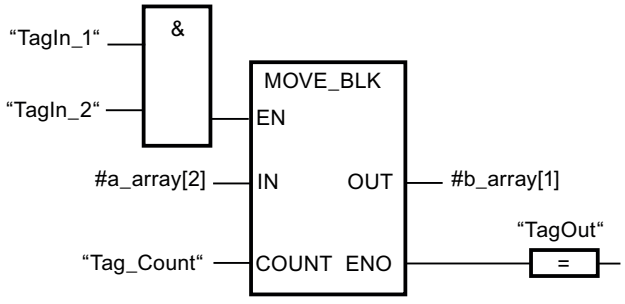

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The data type of the "a_array" operand is ARRAY [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the "b_array" operand is ARRAY [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

Note

You can find more information on the MOVE_BLK instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

MOVE_BLK_VARIANT: Move block

Description

You use the "Move block" instruction to move the content of a memory area (source range) to another memory area (target range). You can copy a complete ARRAY or elements of an ARRAY to another ARRAY of the same data type. The size (number of elements) of source and target ARRAY can be different. You can copy multiple or single elements within an ARRAY.

The number of elements that are to be copied must not exceed the selected source range or target range.

If you use the instruction at the time the block is being created, the ARRAY does not yet have to be known, as the source and the target are transferred per VARIANT.

Counting at the parameters SRC_INDEX and DEST_INDEX always begins with the low limit "0", regardless of the later declaration of the ARRAY.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is copied than is made available.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input [2)]	VARIANT (which points to an ARRAY or an individual ARRAY element), ARRAY of <Data_type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L or constant	Number of elements which are copied Assign the value "1" to the parameter COUNT, if no ARRAY is specified at parameter SRC or at parameter DEST.
SRC_INDEX	Input	DINT	I, Q, M, D, L or constant	Defines the first element to be copied: • The SRC_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter SRC, the integer at parameter SRC_INDEX specifies the first element within the source area from which it is to be copied. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter SRC or only one single element of an ARRAY is specified, then assign the value "0" at parameter SRC_INDEX.

Parameters	Declaration	Data type	Memory area	Description
DEST_INDEX	Input	DINT	I, Q, M, D, L or constant	Defines the start of the destination memory area: • The DEST_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter DEST, the integer at parameter DEST_INDEX specifies the first element within the target range that is to be copied into. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter DEST, then assign the value "0" at parameter DEST_INDEX.
DEST	Output [1)]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Destination area into which the contents of the source block are copied.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
1) The DEST parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface. 2) With the SRC parameter, the data types BOOL and ARRAY of BOOL are not permitted.				

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data types do not correspond. Instead of an ARRAY of Struct, use an ARRAY of PLC data type (UDT).
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at the SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value

Error code* (W#16#...)	Explanation
8382	The value at the SRC_INDEX parameter is less than zero.
8383	The value at the SRC_INDEX is outside the high limit of the ARRAY.
8482	The value at the DEST_INDEX parameter is less than zero.
8483	The value at the DEST_INDEX is outside the high limit of the ARRAY.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at the DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

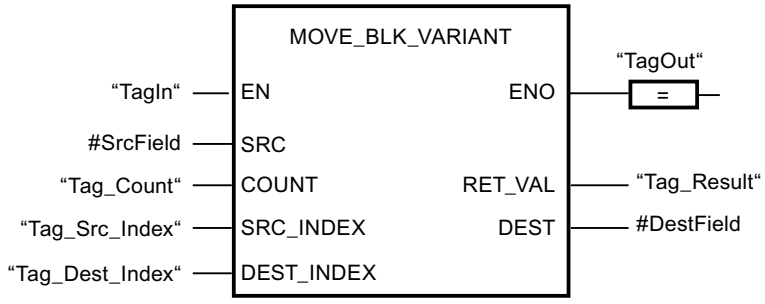

The following table shows how the instruction works using specific operand values:

Parameters	Declaration in the block interface	Operand	Value
SRC	Input	#SrcField	The local operand #SrcField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[0..10] of "MOVE_UDT")
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3

Parameters	Declaration in the block interface	Operand	Value
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	The local operand #DestField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[10..20] of "MOVE_UDT")

If the "TagIn" operand returns signal state "1", the instruction is executed. Two elements are moved from the source range, starting from the fourth element of the ARRAY [0..10] of MOVE_UDT, to the target range. The copies are pasted in the ARRAY [10..20] of MOVE_UDT starting from the fourth element. If the instruction is executed without errors, the enable output ENO has the signal state TRUE and the "TagOut" output is set.

Note

You can find more information on the MOVE_BLK_VARIANT instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

VariantGet: Read out VARIANT tag value (Page 1172)

Example of moving data (Page 333)

Basic information on VARIANT (Page 324)

Basic information on FBD (Page 7185)

UMOVE_BLK: Move block uninterruptible

Description

You use the "Move block uninterruptible" instruction to move the content of a memory area (source range) to another memory area (target range). The instruction cannot be interrupted. The number of elements to be moved to the target range is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source range and the target range have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Note

Use of ARRAYs

The instruction copies the contents from the defined element n elements (n = depending on the value at the parameter COUNT) from the source range to the target range, starting at the specified index.

You use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN [1)]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	The first element of the source area that is being copied

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Number of elements to be copied from the source range to the target range
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	The first element of the target range to which the contents of the source range are being copied
[1] The specified data types can only be used as elements of an array structure.						

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

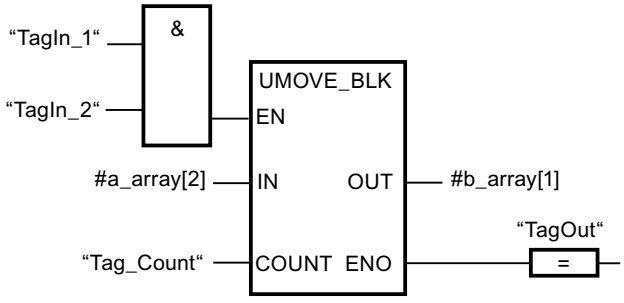

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The data type of the "a_array" operand is ARRAY[0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the "b_array" operand is ARRAY[0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element.

The move operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

Note

You can find more information on the UMOVE_BLK instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (target range) with the value of the IN input. The target range is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is moved to the target range as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source range and the target range have the same data type.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The maximum number of elements changed is the number of elements in the ARRAY or structure. If you copy more data than there are elements at the OUT output, you will get an unintended result.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Note

Use of ARRAYs

The instruction reads the content from the selected element from the source range and copies this content n-times (n = depending on the value at the parameter COUNT) to the destination, starting at the specified index.

Filling structures

As well as the elements of an ARRAY, you can also fill multiple elements of a structure (STRUCT, PLC data type) with the same value. The structure whose elements you want to fill must only contain individual elements of the same elementary data type. The structure itself can, however, be embedded in another structure.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Element used to fill the target range

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Number of repeated move operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	Address in target range from which filling starts

For additional information on valid data types, refer to "See also".

Example with an ARRAY

The following example shows how the instruction works when you want to fill an ARRAY:

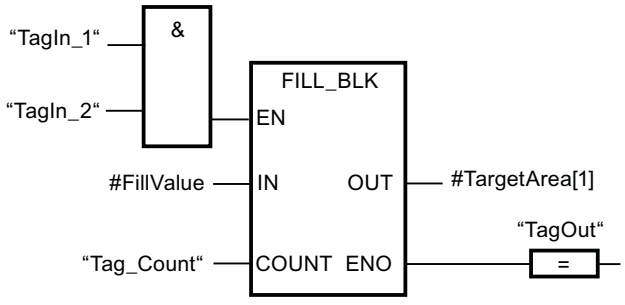

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	FillValue	The data type of the operand is INT.
COUNT	Tag_Count	3
OUT	TargetArea	The data type of the operand TargetArea is ARRAY[1..5] of INT. It consists of five elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction copies the value of the operand #FillValue to the #TargetArea output tag three times, starting with the first element. If the instruction is executed without errors, the TagOut operand at the "ENO" enable output is set to signal state "1".

Examples with a structure

The following examples show how the instruction works when you want to fill a structure:

Create a global data block with the following elements:

Data_block_1					Data type
MyStruct1					STRUCT
	Member_1				INT
	Member_2				INT
	Member_3				INT
	Member_4				INT
MyStruct2					STRUCT
	SubArray				ARRAY[1..2] of STRUCT
		SubArray[1]			STRUCT
			NestedStruct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT
		SubArray[2]			STRUCT
			Nested-Struct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT

Generate the following program code to address the MyStruct1 tag:

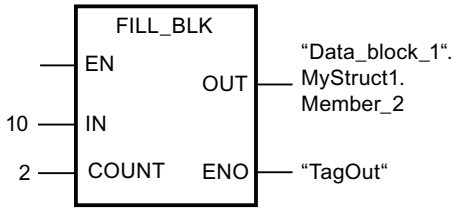

Generate the following program code to address the MyStruct2 tag:

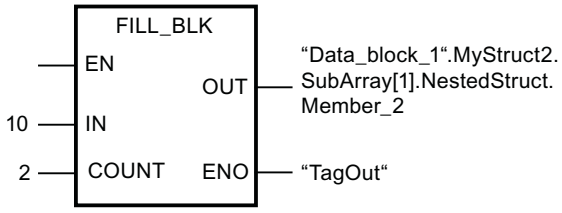

In each of the two examples, the value 10 of the IN parameter is copied twice to the operand at the OUT parameter, starting with the element Member_2. This means that the value 10 is copied to the elements Member_2 and Member_3. The other two elements, Member_1 and Member_4, are not changed. If the instruction is executed without errors, the TagOut operand is set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

UFILL_BLK: Fill block uninterruptible

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (target range) with the value of the IN input. The instruction cannot be interrupted. The target range is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is moved to the target range as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source range and the target range have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The maximum number of elements changed is the number of elements in the ARRAY or structure. If you copy more data than there are elements at the OUT output, you will get an unintended result.

When a ARRAY of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Note

Use of ARRAYs

The instruction reads the content from the selected element from the source range and copies this content n-times (n = depending on the value at the parameter COUNT) to the destination, starting at the specified index.

You use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Filling structures

As well as the elements of an ARRAY, you can also fill multiple elements of a structure (STRUCT, PLC data type) with the same value. The structure whose elements you want to fill must only contain individual elements of the same elementary data type. The structure itself can, however, be embedded in another structure.

Parameters

The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Element used to fill the target range
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Number of repeated move operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	D, L	Address in target range from which filling starts

For additional information on valid data types, refer to "See also".

Example with an ARRAY

The following example shows how the instruction works when you want to fill an ARRAY:

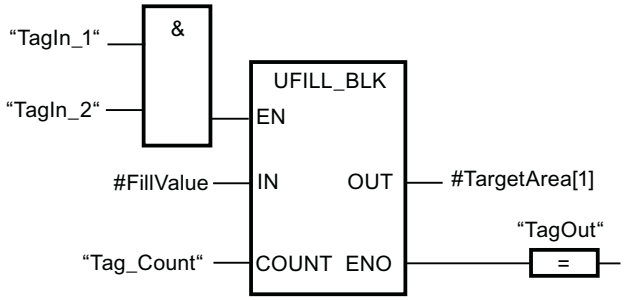

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	FillValue	The data type of the operand is INT.
COUNT	Tag_Count	3
OUT	TargetArea	The data type of the operand TargetArea is ARRAY[1..5] of INT. It consists of five elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction copies the value of the operand #FillValue to the #TargetArea output tag three times, starting with the first element. The move operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the TagOut operand at the "ENO" enable output is set to signal state "1".

Examples with a structure

The following examples show how the instruction works when you want to fill a structure:

Create a global data block with the following elements:

Data_block_1		Data type
MyStruct1		STRUCT
	Member_1	INT
	Member_2	INT
	Member_3	INT
	Member_4	INT
MyStruct2		STRUCT

Data_block_1					Data type
	SubArray				ARRAY[1..2] of STRUCT
		SubArray[1]			STRUCT
			NestedStruct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT
		SubArray[2]			STRUCT
			Nested-Struct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT

Generate the following program code to address the MyStruct1 tag:

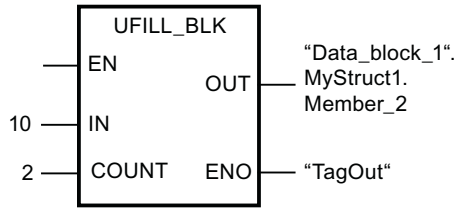

Generate the following program code to address the MyStruct2 tag:

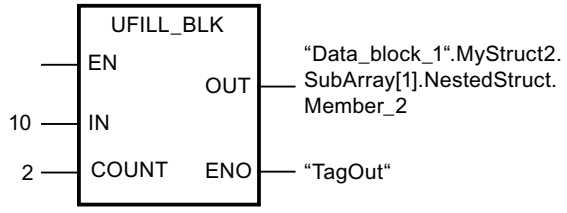

In each of the two examples, the value 10 of the IN parameter is copied twice to the operand at the OUT parameter, starting with the element Member_2. This means that the value 10 is copied to the elements Member_2 and Member_3. The other two elements, Member_1 and Member_4, are not changed. If the instruction is executed without errors, the TagOut operand is set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

SCATTER: Parse the bit sequence into individual bits

Description

The "Parse the bit sequence into individual bits" instruction parses a tag of the BYTE, WORD, DWORD or LWORD data type into individual bits and saves them in an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements.

Note

Multi-dimensional ARRAY of BOOL

With the "Parse the bit sequence into individual bits" instruction, the use of a multidimensional ARRAY of BOOL is not permitted.

Note

Length of the ARRAY, STRUCT or PLC data type

The ARRAY, the anonymous STRUCT or the PLC data type must have exactly the number of elements that is specified by the bit sequence.

This means for the data type BYTE, for example, the ARRAY, STRUCT or the PLC data type must have exactly 8 elements (WORD = 16, DWORD = 32 and LWORD = 64).

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

This way you can, for example, parse a status word, and read and change the status of the individual bits using the index. Using GATHER, you can merge the bits once again into a bit sequence.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- If the ARRAY, STRUCT or PLC data type does not provide enough BOOL elements.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD	I, Q, M, D, L	Bit sequence that is parsed. The values must not be located in the I/O area or in the DB of a technology object.
OUT	Output	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	I, Q, M, D, L	ARRAY, STRUCT or PLC data type in which the individual bits are stored

You can find additional information on valid data types under "See also".

Example with an ARRAY

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceWord		WORD
EnableOut	Output	BOOL
DestinationArray		ARRAY[0..15] of BOOL

The following example shows how the instruction works:

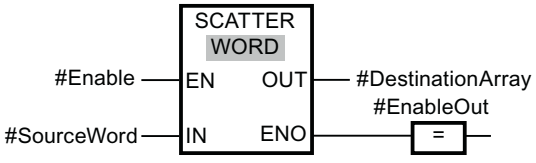

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceWORD	WORD (16 bits)
OUT	DestinationArray	The operand "DestinationArray" is of the data type ARRAY[0..15] of BOOL. It consists of 16 elements and is therefore just as large as the WORD that is to be parsed.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The operand #SourceWord of the data type WORD is parsed into its individual bits

(16) and assigned to the individual elements of the #DestinationArray operand. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

Example with a PLC data type (UDT)

Create the following PLC data type "myBits":

myBits		Name	Data type
1		GeneralError	Bool
2		DeviceError	Bool
3		CommError	Bool
4		myError1	Bool
5		myError2	Bool
6		myError3	Bool
7		myError4	Bool
8		myError5	Bool
9		myError6	Bool
10		myError7	Bool
11		myError8	Bool
12		myError9	Bool
13		myError10	Bool
14		myError11	Bool
15		myError12	Bool
16		myError13	Bool

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceWord		WORD
EnableOut	Output	BOOL
DestinationUDT		"myBits"

The following example shows how the instruction works:

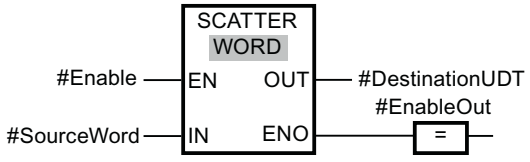

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceWord	WORD (16 bits)
OUT	DestinationUDT	The "DestinationUDT" operand has the PLC data type (UDT). It consists of 16 elements and is therefore just as large as the WORD that is to be parsed.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The #SourceWord operand of the data type WORD is parsed into its individual bits (16) and assigned to the individual elements of the #DestinationArray operand. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

SCATTER_BLK: Parse elements of an ARRAY of bit sequence into individual bits

Description

The "Parse elements of an ARRAY of bit sequence into individual bits" instruction parses one or more elements of an ARRAY of BYTE, WORD, DWORD or LWORD into individual bits and saves them in an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements. At the COUNT_IN parameter you specify how many elements of the source ARRAY are going to be parsed. The source ARRAY at the IN parameter may have more elements than specified at the COUNT_IN parameter. The ARRAY of BOOL, the anonymous STRUCT or the PLC data type must have sufficient elements to save the bits of the parsed bit sequences. However, the destination memory area may also be larger.

Note

Multi-dimensional ARRAY of BOOL

If the ARRAY is a multi-dimensional ARRAY of BOOL, the filling bits of the dimensions contained are counted as well even if they were not explicitly declared.

Example 1: An ARRAY[1..10,0..4,1..2] of BOOL is handled like an ARRAY[1..10,0..4,1..8] of BOOL or like an ARRAY[0..399] of BOOL.

Example 2: At the IN parameter, an ARRAY[0..5] of WORD (sourceArrayWord[2]) is interconnected. The COUNT_IN parameter has the value "3". At the OUT parameter, an ARRAY[0..1,0..5,0..7] of BOOL (destinationArrayBool[0,0,0]) is interconnected. Both the array at the IN parameter and at the OUT parameter has a size of 96 bits. The ARRAY of WORD is parsed into 48 individual bits.

Note

If the ARRAY low limit of the target ARRAY is not "0", note the following:

For performance reasons the index must always start at a BYTE, WORD, DWORD or LWORD limit. This means the index must be calculated starting at the low limit of the ARRAY. The following formula is used as basis for this calculation:

Valid indices = ARRAY low limit + n(number of bit sequences) × number of bits of the desired bit sequence

For an ARRAY[-2..45] of BOOL and the bit sequence WORD the calculation looks as follows:

- Valid index (-2) = -2 + 0 × 16
- Valid index (14) = -2 + 1 × 16
- Valid index (30) = -2 + 2 × 16

You can find an example described below.

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

This way you can, for example, parse status words, and read and change the status of the individual bits using the index. Using GATHER, you can merge the bits once again into a bit sequence.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- The source ARRAY has fewer elements than specified at the COUNT_IN parameter.
- The index of the destination ARRAY does not start at a BYTE, WORD, DWORD or LWORD limit. In this case, no result is written to the ARRAY of BOOL.
- The ARRAY[*] of BOOL, STRUCT or PLC data type does not provide the required number of elements. In this case as many bit sequences as possible are parsed and written to the ARRAY of BOOL, the anonymous STRUCT or the PLC data type. The remaining bit sequences are no longer taken into account.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Element of an ARRAY[*] of <bit sequence>	Element of an ARRAY[*] of <bit sequence>	I, Q, M, D, L	ARRAY of <bit sequence> that is parsed. The values must not be located in the I/O area or in the DB of a technology object.
COUNT_IN	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Counter for the number of elements of the source ARRAY that are going to be parsed. The value must not be in the I/O area or in the database of a technology object.
OUT	Output	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	I, Q, M, D, L	ARRAY, STRUCT or PLC data type in which the individual bits are stored

You can select the required bit sequence from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example of a destination ARRAY with the low limit "0"

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceArrayWord		ARRAY[0..5] of WORD
CounterInput		UDINT
EnableOut	Output	BOOL
DestinationArrayBool		ARRAY[0..95] of BOOL

The following example shows how the instruction works:

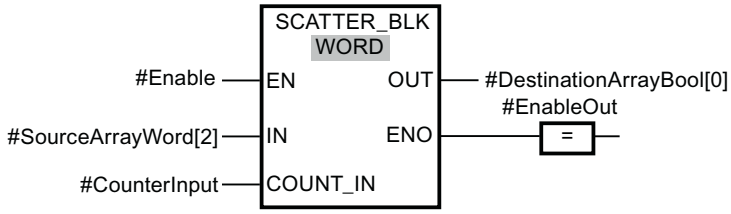

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayWord[2]	ARRAY[0..5] of WORD (96 bits can be parsed.)
COUNT_IN	CounterInput = 3	UDINT3 (3 WORDs or 48 bits are to be parsed. This means at least 48 bits must be available in the destination ARRAY.)
OUT	DestinationArrayBool[0]	The operand "DestinationArray-Bool" is of the data type AR-RAY[0..95] of BOOL. This means it provides 96 BOOL ele-ments.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The 3rd, 4th and 5th WORD of the #SourceArrayWord operand is parsed into its individual bits (48) and as of the 1st element assigned to the individual elements of the #DestinationArrayBool operand. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

Example of a destination ARRAY with the low limit "-2"

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceArrayWord		ARRAY[0..5] of WORD
CounterInput		UDINT
EnableOut	Output	BOOL
DestinationArrayBool		ARRAY[-2..93] of BOOL

The following example shows how the instruction works:

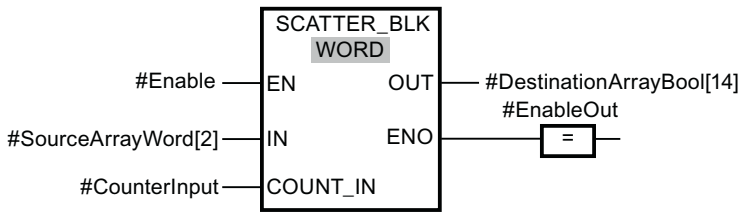

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayWord[2]	ARRAY[0..5] of WORD (96 bits can be parsed.)
COUNT_IN	CounterInput = 3	UDINT3 (3 WORDs or 48 bits are to be parsed. This means at least 48 bits must be available in the destination ARRAY.)
OUT	DestinationArrayBool[14]	The operand "DestinationArray-Bool" is of the data type AR-RAY[-2..93] of BOOL. This means it provides 96 BOOL ele-ments.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The 3rd, 4th and 5th WORD of the #SourceArrayWord operand is parsed into its individual bits (48) and as of the 16th Element assigned to the individual elements of the #DestinationArrayBool operand. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO. The remaining 32 bits are not written.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

GATHER: Merge individual bits into a bit sequence

Description

The "Merge individual bits into a bit sequence" instruction merges the bits from an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements into a bit sequence. The bit sequence is saved in a tag of the data type BYTE, WORD, DWORD or LWORD.

Note

Multi-dimensional ARRAY of BOOL

With the "Merge individual bits into a bit sequence" instruction, the use of a multidimensional ARRAY of BOOL is not permitted.

Note

Length of the ARRAY, STRUCT or PLC data type

The ARRAY, STRUCT or the PLC data type must have exactly the number of elements that is specified by the bit sequence.

This means for the data type BYTE, for example, the ARRAY, anonymous STRUCT or the PLC data type must have exactly 8 elements (WORD = 16, DWORD = 32 and LWORD = 64).

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- If the ARRAY, the anonymous STRUCT or the PLC data type (UDT) has fewer or more BOOL elements than specified by the bit sequence, they are not transferred and the ENO returns signal state "0". The ENO also returns "0" if fewer than the necessary number of bits is available.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	I, Q, M, D, L	ARRAY, STRUCT or PLC data type, the bits of which are merged into a bit sequence. The values must not be located in the I/O area or in the DB of a technology object.
OUT	Output	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD	I, Q, M, D, L	Merged bit sequence, saved in a tag

You can select the required bit sequence from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example with an ARRAY

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceArray		ARRAY[0..15] of BOOL
EnableOut	Output	BOOL
DestinationWord		WORD

The following example shows how the instruction works:

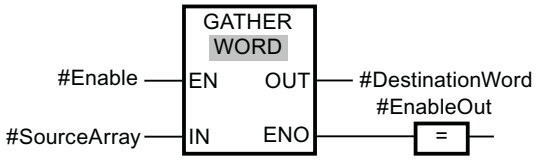

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArray	The operand "SourceArray" is of the data type ARRAY[0..15] of BOOL. It consists of 16 elements and is therefore just as large as the WORD in which the bits are to be merged.
OUT	DestinationWord	WORD (16 bits)

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The bits of the #SourceArray operand are merged into a WORD. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

Example with a PLC data type (UDT)

Create the following PLC data type "myBits":

myBits		
	Name	Data type
1	GeneralError	Bool
2	DeviceError	Bool
3	CommError	Bool
4	myError1	Bool
5	myError2	Bool
6	myError3	Bool
7	myError4	Bool
8	myError5	Bool
9	myError6	Bool
10	myError7	Bool
11	myError8	Bool
12	myError9	Bool
13	myError10	Bool
14	myError11	Bool
15	myError12	Bool
16	myError13	Bool

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceUDT		"myBits"
EnableOut	Output	BOOL
DestinationWord		WORD

The following example shows how the instruction works:

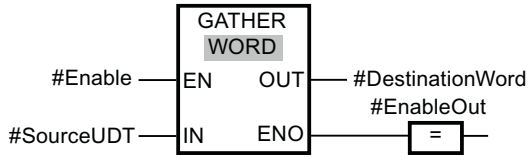

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceUDT	The "SourceUDT" operand has the PLC data type (UDT). It consists of 16 elements and is therefore just as large as the WORD in which the bits are to be merged.
OUT	DestinationWord	WORD (16 bits)

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. The bits of the #SourceUDT operand are merged into a WORD. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

GATHER_BLK: Merge individual bits into multiple elements of an ARRAY of bit sequence

Description

The "Merge individual bits into multiple elements of an ARRAY of bit sequence" instruction merges the bits from an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements into one or multiple elements of an ARRAY of <bit sequence>. At the COUNT_OUT parameter you specify how many elements of the destination ARRAY are going to be written. With this step you also implicitly specify how many elements of the ARRAY of BOOL, the anonymous STRUCT or the PLC data type are required. The destination ARRAY at the OUT parameter may have more elements than specified at the COUNT_OUT parameter. The ARRAY of <bit sequence> must have sufficient elements to save the bits that are going to be merged. However, the destination ARRAY may also be larger.

Note

Multi-dimensional ARRAY of BOOL

If the ARRAY is a multi-dimensional ARRAY of BOOL, the filling bits of the dimensions contained are counted as well even if they were not explicitly declared.

Example 1: An ARRAY[1..10,0..4,1..2] of BOOL is handled like an ARRAY[1..10,0..4,1..8] of BOOL or like an ARRAY[0..399] of BOOL.

Example 2: At the OUT parameter, an ARRAY[0..5] of WORD (sourceArrayWord[2]) is interconnected. The COUNT_IN parameter has the value "3". At the IN parameter, an ARRAY[0..1,0..5,0..7] of BOOL (destinationArrayBool[0,0,0]) is interconnected. Both the array at the IN parameter and at the OUT parameter has a size of 96 bits. 48 individual bits are merged from the ARRAY of BOOL.

Note

If the ARRAY low limit of the source ARRAY is not "0", note the following:

For performance reasons the index must always start at a BYTE, WORD, DWORD or LWORD limit. This means the index must be calculated starting at the low limit of the ARRAY. The following formula is used as basis for this calculation:

Valid indices = ARRAY low limit + n(number of bit sequences) × number of bits of the desired bit sequence

For an ARRAY[-2..45] of BOOL and the bit sequence WORD the calculation looks as follows:

- Valid index (-2) = -2 + 0 × 16
- Valid index (14) = -2 + 1 × 16
- Valid index (30) = -2 + 2 × 16

You can find an example described below.

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- The index of the source ARRAY does not start at a BYTE, WORD, DWORD or LWORD limit. In this case, no result is written to the ARRAY of <bit sequence>.
- The ARRAY[*] of <bit sequence> does not provide the required number of elements. In this case as many bit sequences as possible are merged and written to the ARRAY of <bit sequence>. The remaining bits are no longer taken into account.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	I, Q, M, D, L	ARRAY of BOOL, STRUCT or PLC data type whose bits are merged (source ARRAY) The values must not be located in the I/O area or in the DB of a technology object.

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
COUNT_OUT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Counter how many elements of the target ARRAY are to be described. The value must not be in the I/O area or in the database of a technology object.
OUT	Output	Element of an ARRAY[*] of <bit sequence>	Element of an ARRAY[*] of <bit sequence>	I, Q, M, D, L	ARRAY of <bit sequence> to which the bits are saved (destination ARRAY)

You can select the required bit sequence from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example of a source ARRAY with the low limit "0"

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceArrayBool		ARRAY[0..95] of BOOL
CounterOutput		UDINT
EnableOut	Output	BOOL
DestinationArrayWord		ARRAY[0..5] of WORD

The following example shows how the instruction works:

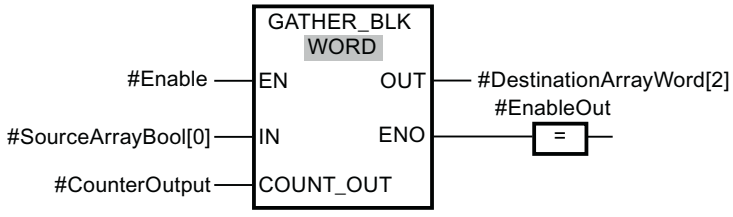

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayBool[0]	The operand "SourceArrayBool" is of the data type ARRAY[0..95] of BOOL. This means it provides 96 BOOL elements that can merged into words once again.
COUNT_OUT	CounterOutput = 3	UDINT3 (3 words are to be written. This means 48 bits must be available in the source ARRAY.)
OUT	DestinationArrayWord[2]	The operand "DestinationArrayWord" is of the data type ARRAY[0..5] of WORD. This means 6 WORD elements are available.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. As of the 1st element of the #SourceArrayBool operand, 48 bits are merged in the #DestinationArrayWord operand. The starting point in the destination ARRAY is the 3rd element. This means that the first 16 bits are written into the 3rd word, the second 16 nits into the 4th word and the third 16 bits into the 5th word of the destination ARRAY. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

Example of a source ARRAY with the low limit "-2"

Create the following tags in the block interface:

Tag	Section	Data type
Enable	Input	BOOL
SourceArrayBool		ARRAY[-2..93] of BOOL
CounterOutput		UDINT
EnableOut	Output	BOOL
DestinationArrayWord		ARRAY[0..5] of WORD

The following example shows how the instruction works:

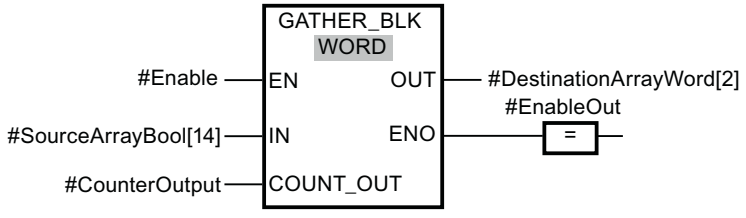

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayBool[14]	The operand "SourceArrayBool" is of the data type ARRAY[-2..93] of BOOL. Because the starting point is the 16th element, only 80 BOOL elements that can be merged into words again are available.
COUNT_OUT	CounterOutput = 3	UDINT3 (3 words are to be written. This means 48 bits must be available in the source ARRAY.)
OUT	DestinationArrayWord[2]	The operand "DestinationArrayWord" is of the data type ARRAY[0..5] of WORD. This means 6 WORD elements are available.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. Starting with the 16th element of the #SourceArrayBool operand, 48 bits are merged into the #DestinationArrayWord operand. The starting point in the destination ARRAY is the 3rd element. This means the first 16 bits of the source ARRAY are ignored. The second 16 Bits are written into the 3rd word, the third 16 bits into the 4th word and the fourth 16 bits into the 5th word of the destination ARRAY. The remaining 64 bits of the source ARRAY are not taken into account either. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

AssignmentAttempt: Attempt assignment of a VARIANT to a reference (S7-1500)

Description

With the "AssignmentAttempt" instruction, you attempt to assign a VARIANT tag to a reference tag. The data type of a reference tag is specified at the time of the declaration, while the data type of a VARIANT tag is determined during runtime. An implicit data type conversion is not permitted with reference tags. To assign the two data types to each other therefore, use the assignment attempt.

With the assignment attempt, a check is made at runtime to find out whether the VARIANT tag is of the correct data type. If this is the case, the instruction is executed. If the instruction is executed successfully, there is a valid reference in the target tag, otherwise a NULL.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL		Enable input
ENO	Output	BOOL		Enable output
SRC	Input	VARIANT	• Block interface of an FC: Input, Output, InOut, Temp • Block interface of an FB: Input, InOut, Temp	Pointer to the source tag whose address will be read out
DST	Output	Reference to: • Bit sequences, except BOOL, • Integers, • Floating-point numbers, • Character strings, • PLC data types (UDT), • System data types (SDT), • ARRAYs of the named data types	• Block interface of an FC: Input, Output, Temp, Return • Block interface of an FB: Temp	Reference to which the address of the source tag will be transferred

You will find additional information on valid data types under "See also".

Rules

The following rules apply to the assignment attempt: VARIANT tags that do not match these rules return the value "NULL" at runtime.

- The VARIANT must point to an address in the optimized memory area.
- The VARIANT must not point to an address in a temporary memory area.
- If you want to assign a VARIANT to a reference to an ARRAY the following rules apply:
  - The VARIANT tag must point to an ARRAY whose limits exactly match with those of the declared reference. A VARIANT tag which points to an ARRAY [0..9], does not match a tag REF_TO ARRAY[1..10].
  - Moreover you should compile the blocks that form the value of the VARIANT tag once in a CPU of the S7-1500 series, firmware version V2.5.

Example

The following example shows how the instruction works:

In the interface of the block, the VARIANT "myVariant" and the reference tag "myReference" were declared.

	Name	Data type
	Input	
	Output	
	InOut	
	myVariant	Variant
	Static	
	Temp	
	myReference	REF_TO Int
	Constant	

In the program code an attempt is made to assign "myVariant" to the reference tag "myReference". If "myVariant" has the data type "Int" at runtime, "myReference" contains a valid reference to the source tag of the VARIANT, otherwise NULL. When the assignment was successful, the enable output "ENO" has the signal state "1" and the value "10" can be written to the target tag.

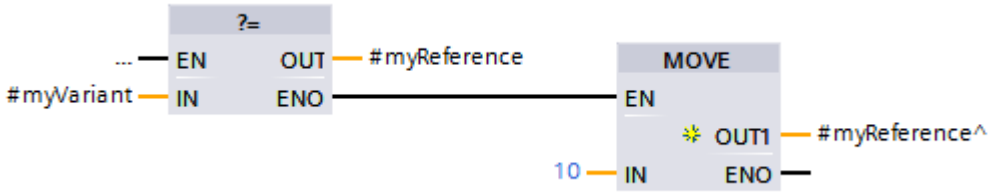

See also

Assignment attempt from VARIANT to a reference (Page 314)

Basic information about references (Page 300)

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

Sample Library for Instructions (Page 593)

SWAP: Swap

Description

You can use the instruction "Swap" to change the order of the bytes within the tag at input IN and query the result at output OUT.

The following figure shows how the bytes of a DWORD data type operand are swapped using the "Swap" instruction:

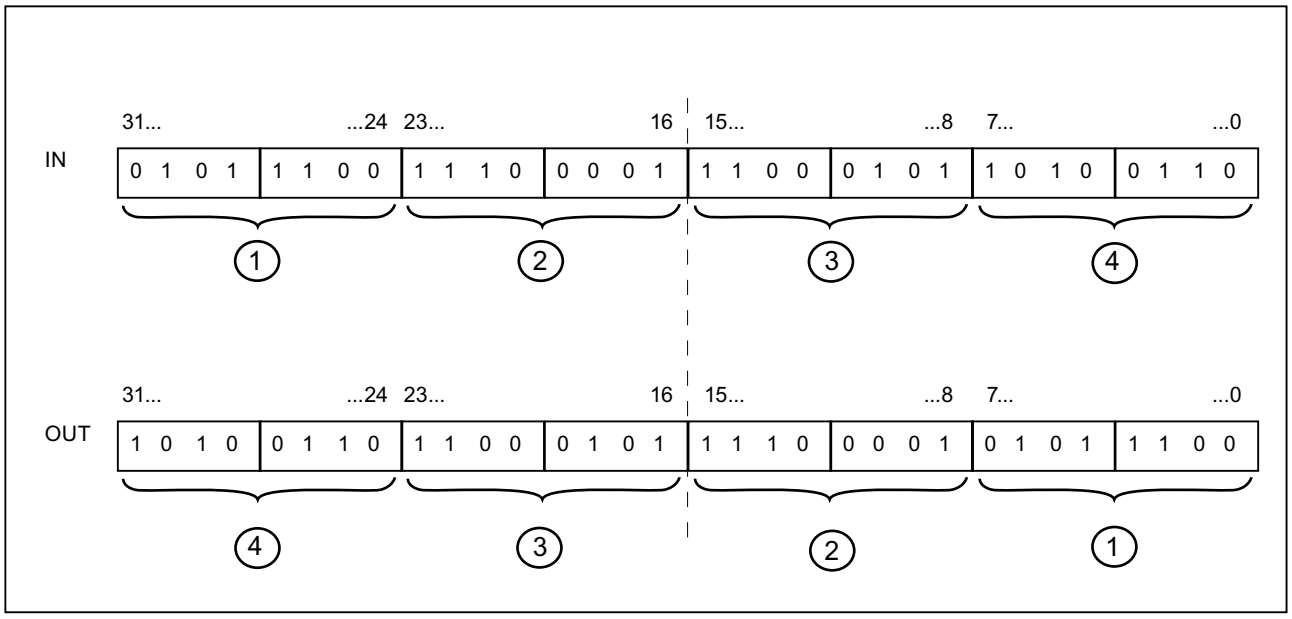

Parameters

The following table shows the parameters of the "Swap" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Operand whose bytes are swapped.
OUT	Output	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

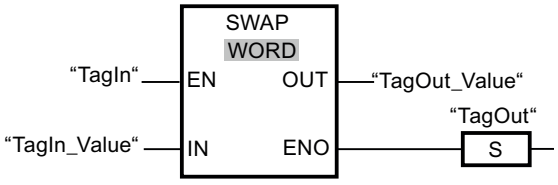

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 0101 0101
OUT	TagOut_Value	0101 0101 0000 1111

If the "TagIn" operand has signal state "1", the instruction is executed. The arrangement of the bytes is changed and stored in the operand "TagOut_Value".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

ARRAY DB (S7-1500)

ReadFromArrayDB: Read from ARRAY data block (S7-1500)

Description

The instruction "Read from ARRAY data block" is used to read the element from a data block of the ARRAY DB block type to which the index references, and write the value of the element to the target range.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

Enable output ENO returns the signal state FALSE if one of the following conditions applies:

- Enable input EN returns the signal state FALSE.
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Read from ARRAY data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB that is read. The specification can be a constant, a global tag or an indexed value.
VALUE	Output [1]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Value that is read and output
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
1) The VALUE parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, write protected, or is located in load memory.
8135	The ARRAY data block contains invalid values.
8154	The data block has the incorrect data type.
8282	The value at the INDEX parameter is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8452	Code generation error

Error code* (W#16#...)	Explanation
8453	There are two possible causes of the error: • The size of the VALUE parameter does not match the element length in the ARRAY data block. • The two tags are not in a memory area with optimized access. For additional information on the memory area access types, refer to: Basics of block access (Page 55)
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

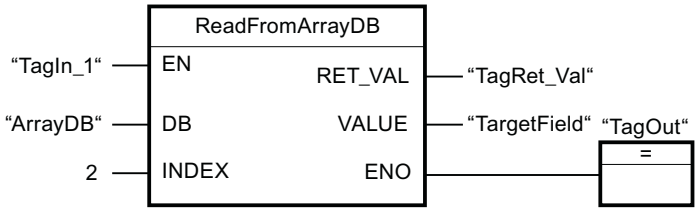

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type Array [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	TargetField	The "TargetField" operand is a global tag of the data type INT.

If the "TagIn1" operand returns signal state "1", the instruction is executed. The 2nd element is read from "ArrayDB" and written to the "TargetField" operand. If the instruction is executed without errors, the enable output ENO has the signal state TRUE and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Example of the use of ARRAY data blocks (Page 44)

Basic information on VARIANT (Page 324)

Basic information on FBD (Page 7185)

WriteToArrayDB: Write to ARRAY data block (S7-1500)

Description

The instruction "Write to ARRAY data block" is used to write the element to which the index references to a data block of the ARRAY DB block type.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

Enable output ENO returns the signal state FALSE if one of the following conditions applies:

- Enable input EN returns the signal state FALSE.
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Write to ARRAY data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written. The specification can be a constant, a global tag or an indexed value.
VALUE	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Value to be written
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, or is located in load memory.
8134	The data block is write protected.
8135	The data block is not an ARRAY data block.
8154	The data block has the incorrect data type.
8282	The value at the INDEX parameter is outside the limits of the ARRAY.
8350	The data type VARIANT at parameter VALUE provides the value "0".
8352	Code generation error
8353	There are two possible causes of the error: • The size of the VALUE parameter does not match the element length in the ARRAY data block. • The two tags are not in a memory area with optimized access. For additional information on the memory area access types, refer to: Basics of block access (Page 55)
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

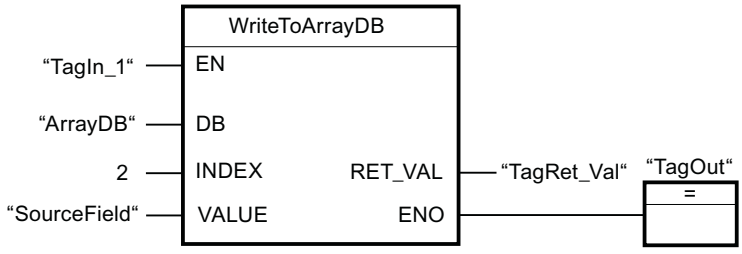

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type Array [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	SourceField	The "SourceField" operand is a global tag of the data type INT.

If the "TagIn1" operand returns signal state "1", the instruction is executed. The value of the "SourceField" operand is written to the 2nd element of the ARRAY DB. If the instruction is executed without errors, the enable output ENO has the signal state TRUE and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Example of the use of ARRAY data blocks (Page 44)

Basic information on VARIANT (Page 324)

Basic information on FBD (Page 7185)

ReadFromArrayDBL: Read from ARRAY data block in load memory (S7-1500)

Description

The instruction "Read from ARRAY data block in load memory" is used to read the element ti which the index references from a data block of the ARRAY DB block type in the load memory and write it to the target range.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Read from ARRAY data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Begin with the reading of the ARRAY DB
DB [1]	Input	DB_ANY	I, Q, M, D, L	ARRAY data block that is read.
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB that is read. The specification can be a constant, a global tag or an indexed value.
VALUE [1]	InOut	VARIANT	D (element of a global data block) L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Pointer to the DB in the work memory that is to be read and the value of which is to be written. No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the ERROR parameter.
[1] The data blocks must be created with the "Optimized" block property.				

Parameter ERROR

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at the INDEX parameter is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error

Error code* (W#16#...)	Explanation
8753	The size of the parameter does not match the element length in the ARRAY data block.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL" and "WRIT_DBL" in the corresponding descriptions of the instructions.

Example

The following example shows how the instruction works:

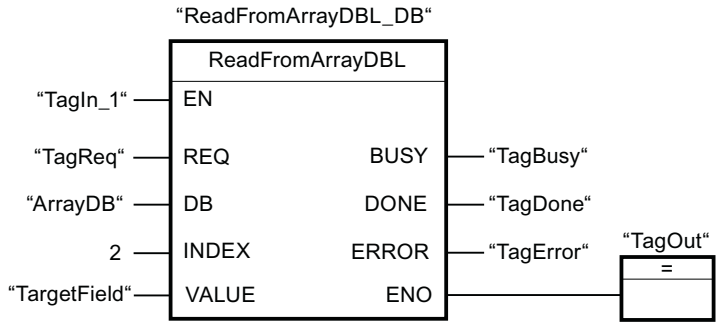

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type ARRAY [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	TargetField	The "TargetField" operand is a global tag of the data type INT.
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq" operand, the instruction is executed. The 2nd element is read from "ArrayDB" and output at the "VALUE" parameter. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. If the instruction is executed without errors, the enable output ENO has the signal state TRUE and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand has the signal state TRUE.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

WRIT_DBL: Write to data block in the load memory (Page 2718)

READ_DBL: Read from data block in the load memory (Page 2714)

Example of the use of ARRAY data blocks (Page 44)

Basic information on VARIANT (Page 324)

Difference between synchronous and asynchronous instructions (Page 596)

Basic information on FBD (Page 7185)

WriteToArrayDBL: Write to ARRAY data block in load memory (S7-1500)

Description

The instruction "Write to ARRAY data block in load memory" is used to write the element to which the index references to a data block of the ARRAY DB block type in load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Write to ARRAY data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Start writing to the array DB
DB [1)]	Input	DB_ANY	I, Q, M, D, L	ARRAY data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written. The specification can be a constant, a global tag or an indexed value.

Parameter	Declaration	Data type	Memory area	Description
VALUE [1]	Input	VARIANT	D (element of a global data block) L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Pointer to the DB in the work memory that is to be read and the value of which is to be written. No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the ERROR parameter.
[1] The data blocks must be created with the "Optimized" block property.				

Parameter ERROR

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8234	The data block is write protected.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at the INDEX parameter is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL" and "WRIT_DBL" in the corresponding descriptions of the instructions.

Example

The following example shows how the instruction works:

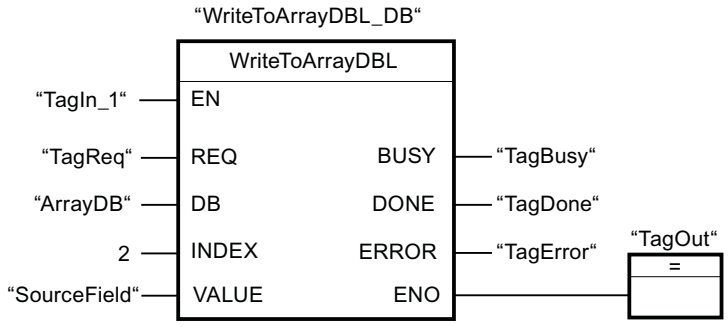

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type ARRAY [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	SourceField	The "SourceField" operand is a global tag of the data type INT.
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq" operand, the instruction is executed. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written to the 2nd element of "ArrayDB". If the instruction is executed without errors, the enable output ENO has the signal state TRUE and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand has the signal state TRUE.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

WRIT_DBL: Write to data block in the load memory (Page 2718)

READ_DBL: Read from data block in the load memory (Page 2714)

Example of the use of ARRAY data blocks (Page 44)

Basic information on VARIANT (Page 324)

Difference between synchronous and asynchronous instructions (Page 596)

Basic information on FBD (Page 7185)

VARIANT

VariantGet: Read out VARIANT tag value

Description

You can use the "Read out VARIANT tag value" instruction to read the value of the tag to which the VARIANT at the SRC parameter points and write it in the tag at the DST parameter.

The SRC parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the DST parameter.

The data type of the tag specified at the DST parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT instruction: Move block" instruction. For additional information, refer to "See also".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameters

The following table shows the parameters of the "Read out VARIANT tag value" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
SRC	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		Tag to be read
DST	Output	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

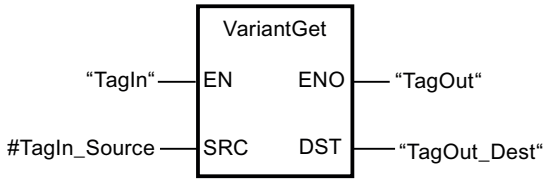

If the "TagIn" operand has signal state "1", the instruction is executed. The value of the tag to which the VARIANT at the "#TagIn_Source" operand points, is read and written to the "TagOut_Dest" operand.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on VARIANT (Page 324)

MOVE_BLK_VARIANT: Move block (Page 1127)

Basic information on FBD (Page 7185)

VariantPut: Write VARIANT tag value

Description

You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter to the memory at the DST parameter to which the VARIANT points.

The DST parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT instruction: Move block" instruction. For additional information, refer to "See also".

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameters

The following table shows the parameters of the instruction "Write VARIANT tag value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
SRC	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Tag to be read
DST	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

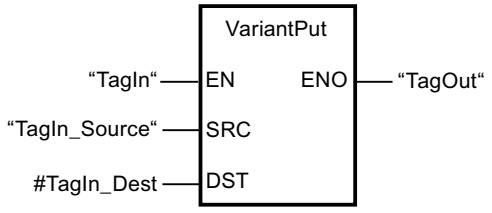

If the "TagIn" operand returns signal state "1", the instruction is executed. The value of the "TagIn_Source" operand is written to the tag to which the VARIANT at the #TagIn_Dest operand points.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on VARIANT (Page 324)

MOVE_BLK_VARIANT: Move block (Page 1127)

Basic information on FBD (Page 7185)

CountOfElements: Get number of ARRAY elements

Description

You use the "Get number of ARRAY elements" instruction to query how many ARRAY elements a tag to which VARIANT points has.

If it is a single-dimensional ARRAY, the number of ARRAY elements is output as the result. (The difference between the high and low limit +1). If it is a multi-dimensional ARRAY, the number of all dimensions is output as the result.

If you want to query the elements of an ARRAY DB, you should use the instructions "ReadFromArrayDB" or "WriteFromArrayDB", because here a more precise error evaluation for the number of elements is possible.

Note

Instances

The VARIANT pointer can point to no instance and therefore also to no multi-instance or an ARRAY of multi-instances.

Note

ARRAY within a data block

If you want to query the number of the elements of an ARRAY which is located in a data block, the block attribute "Data block write-protected in the device" must not be activated in this block. Otherwise, the RET_VAL parameter returns the result "0", irrespective of how many elements the ARRAY contains.

The result is also "0" if the VARIANT tag is not an ARRAY.

If the VARIANT points to an ARRAY of BOOL, the fill elements are included in the count. (For example, 8 is returned for an ARRAY[0..1] of BOOL).

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The VARIANT tag is not an ARRAY. (The result is "0".)

Parameters

The following table shows the parameters of the "Get number of ARRAY elements" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Tag to be queried
RET_VAL	Output	UDINT	I, Q, M, D, L	Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

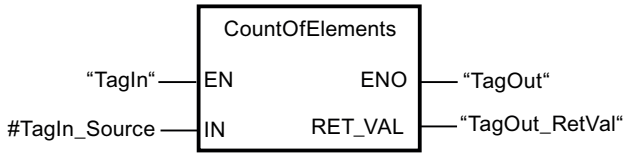

If the "TagIn" operand returns signal state "1", the instruction is executed. The number of ARRAY elements of the tag to which the VARIANT at the #TagIn_Source operand points is read and output at the "TagOut_RetVal" operand.

See also

Basic information on ARRAY (Page 283)

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on VARIANT (Page 324)

Basic information on FBD (Page 7185)

ARRAY[*]

LOWER_BOUND: Read out low ARRAY limit

Description

You can declare tags of the data type ARRAY[*] in the block interface of a function block or function. For these local tags, you can read out the limits of the ARRAY. You will need to specify the required dimension at the DIM parameter.

The "Read out ARRAY low limit" instruction is used to read out the variable low limit of the ARRAY.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The dimension specified at input DIM does not exist.

Note

Availability of the instruction

The instruction is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

Parameters

The following table shows the parameters of the instruction "Read out ARRAY low limit":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
ARR	Input	ARRAY[*]	FB => Section InOut FC => Sections Input and InOut	ARRAY of which the variable low limit is to be read.
DIM	Input	UDINT	I, Q, M, D, L, P or constant	Dimension of the ARRAY of which the variable low limit is to be read.
OUT	Output	DINT	I, Q, M, D, L, P	Result

Example

The following example shows how the instruction works:

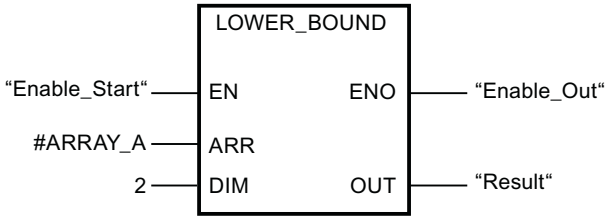

If the "Enable_Start" operand returns signal state "1", the instruction is executed. It reads out the variable low limit of the ARRAY #ARRAY_A from the second dimension. If the instruction is executed without errors, the result is written to operand "Enable_Out" and operand "Result" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on ARRAY (Page 283)

Basic information on FBD (Page 7185)

UPPER_BOUND: Read out high ARRAY limit

Description

You can declare tags of the data type ARRAY[*] in the block interface of a function block or function. For these local tags, you can read out the limits of the ARRAY. You will need to specify the required dimension at the DIM parameter.

The instruction "Read out ARRAY high limit" is used to read out the variable high limit of the ARRAY.

Enable output ENO returns the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The dimension specified at input DIM does not exist.

Note

Availability of the instruction

The instruction is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

Parameters

The following table shows the parameters of the instruction "Read out ARRAY high limit":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
ARR	Input	ARRAY[*]	FB => Section In-Out FC => Sections Input and InOut	ARRAY of which the variable high limit is to be read.
DIM	Input	UDINT	I, Q, M, D, L, P or constant	Dimension of the ARRAY of which the variable high limit is to be read.
OUT	Output	DINT	I, Q, M, D, L, P	Result

Example

The following example shows how the instruction works:

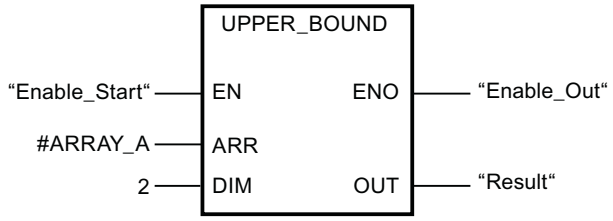

If the "Enable_Start" operand returns signal state "1", the instruction is executed. It reads out the variable high limit of the ARRAY #ARRAY_A from the second dimension. If the instruction is executed without errors, the result is written to operand "Enable_Out" and operand "Result" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on ARRAY (Page 283)

Basic information on FBD (Page 7185)

Legacy (S7-1500)

FieldRead: Read field (S7-1500)

Description

You can use the "Read field" instruction to read out a specific component from the field specified in the MEMBER parameter and transfer its content to the tag in the VALUE parameter. You use the parameter INDEX to define the index of the field components that are to be read. At the parameter MEMBER you specify the first component of the field to be read.

The data types of the field component at parameter MEMBER, the index and the tags at parameter VALUE must correspond to the data type of the instruction "Read field" because implicit conversion is not possible.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The field component specified at the parameter INDEX is not defined in the field specified at the parameter MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Read field":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
INDEX	Input	DINT	DINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Index of field components whose content is read out
MEMBER	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR and WCHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD, CHAR and WCHAR as components of an ARRAY tag	D, L	D, L	First component of the field from which will be read
VALUE	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD CHAR, WCHAR	I, Q, M, D, L, P	I, Q, M, D, L, P	Operand to which the contents of the field component are transferred.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

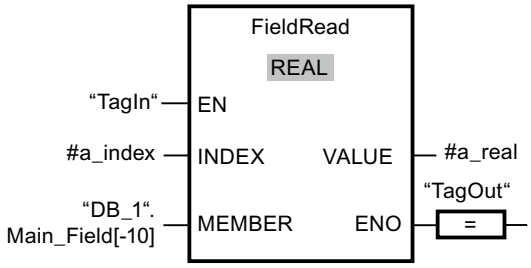

The following table shows how the instruction works using concrete operand values:

Parameters	Tag	Value
INDEX	a_index	4
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the data block "DB_1"
VALUE	a_real	Component with index 4 of the "Main_Field[-10..10] of REAL" field

The field component with index 4 is read from the "Main_Field[-10...10] of REAL" field and written to the "a_real" tag. The field component to be read is defined by the value at the parameter INDEX.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

FieldWrite: Write field (S7-1500)

Description

The "Write field" instruction is used to transfer the content of the tag at the VALUE input to a specific component of the field at the MEMBER output. You use the value at the INDEX input to specify the index of the field component that is described. At the MEMBER output, enter the first component of the field which is to be written to.

The data types of the field component at parameter MEMBER, the index and the tags at parameter VALUE must correspond to the data type of the instruction "Read field" because implicit conversion is not possible.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The field component specified at the input INDEX is not defined in the field specified at the output MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Write field":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
INDEX	Input	DINT	DINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Index of field component that is written with the content of VALUE
VALUE	Input	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD CHAR, WCHAR	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Operand whose contents are copied
MEMBER	Output	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, CHAR and WCHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD, CHAR and WCHAR as components of an ARRAY tag	D, L	D, L	First component of the field to which the content of VALUE is written.

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

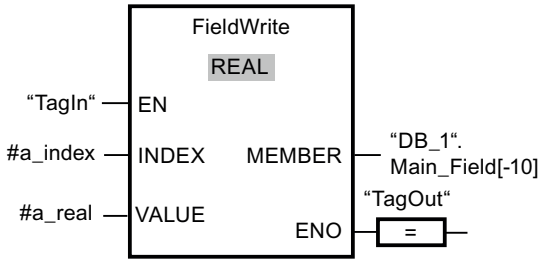

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
INDEX	a_index	4
VALUE	a_real	10.54
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the data block "DB_1"

The value "10.54" of the "a_real" tag is written to the field component with index 4 of the "Main_Field[-10 ... 10] of REAL" field. The index of the field component to which the content of the tag "a_real" is transferred is specified by the value at the input INDEX.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

BLKMOV: Move block (S7-1500)

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can use this tag in memory areas "with optimized block access".

The following figure shows the principle of the move operation:

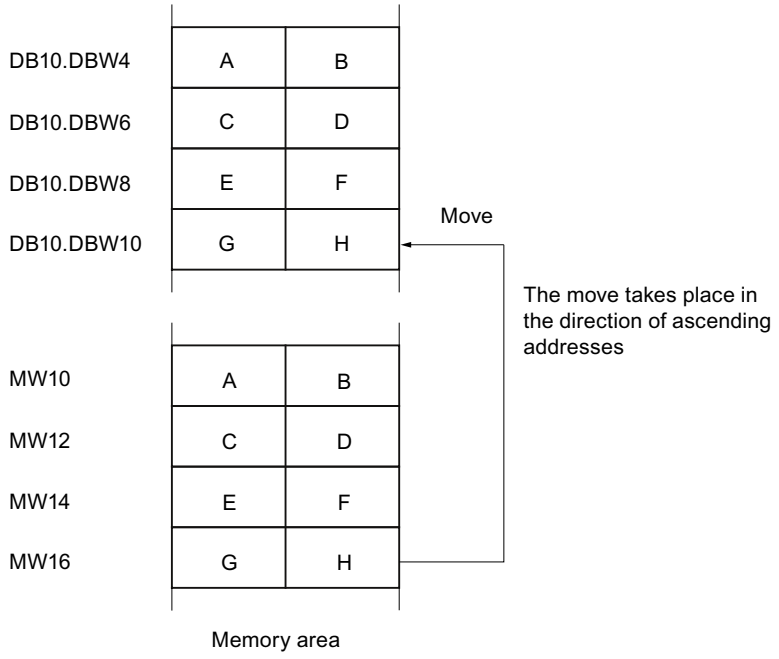

Consistency of source and destination data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes to the SRCBLK and DSTBLK parameters.

Parameters

The following table shows the parameters of the instruction "Move block":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
DSTBLK	Output [1)]	VARIANT	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the DSTBLK parameter.

Error code* (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

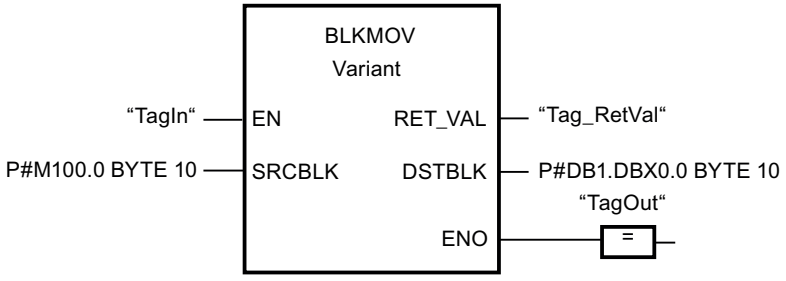

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1230)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

UBLKMOV: Move block uninterruptible (S7-1500)

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

The move operation cannot be interrupted by other operating system activities. As a result the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can use this tag in memory areas "with optimized block access".

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
DSTBLK	Output [1)]	VARIANT	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
8091	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

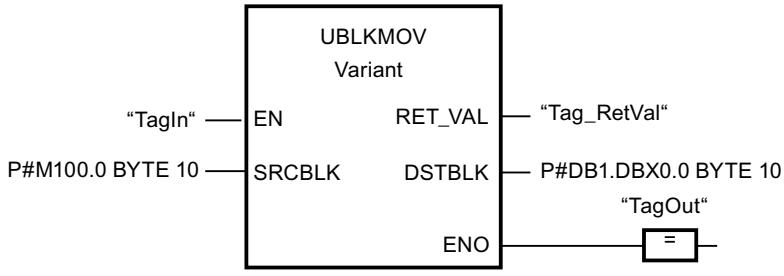

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1230)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

FILL: Fill block (S7-1500)

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can use this tag in memory areas "with optimized block access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK instruction. Fill block" instruction.

The following figure shows the principle of the move operation:

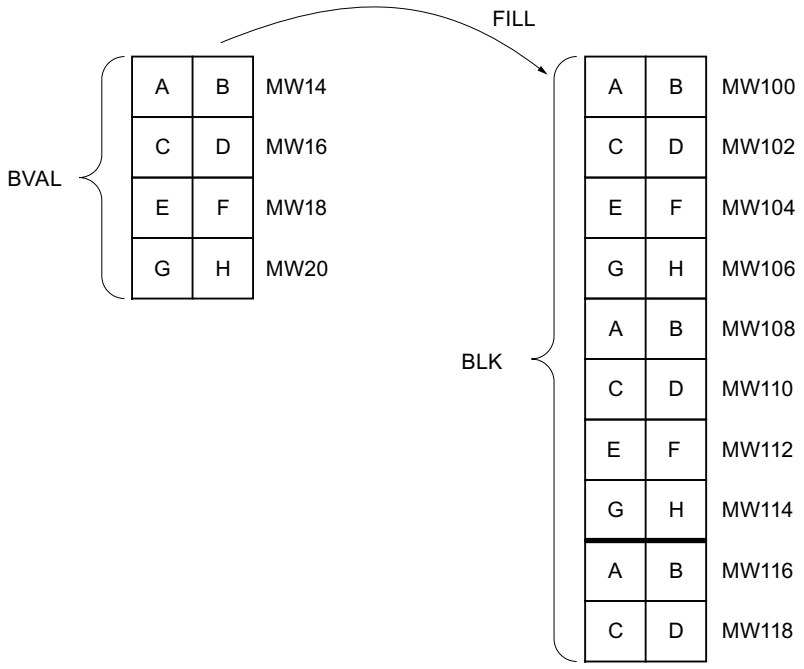

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is data type STRING, the instruction writes the entire string including the administration information.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
BVAL	Input	VARIANT	I, Q, M, D, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
BLK	Output [1)]	VARIANT	I, Q, M, D, L, P	Specification of the memory area that will be filled with the content of the source area.
1) The BLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the BVAL parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the BLK parameter.

Error code* (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

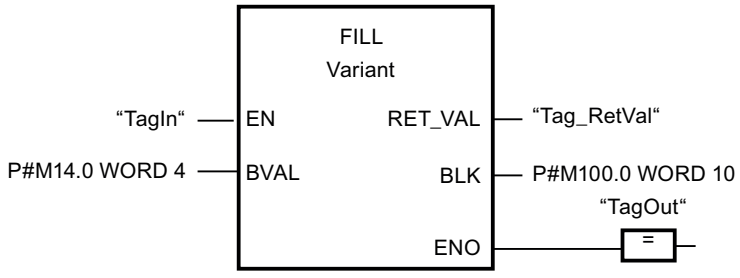

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1230)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types set in the instruction box. The converted value is output at the OUT output.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- Errors, such as an overflow, occur during execution.

Conversion options for bit strings

The bit strings BYTE and WORD cannot be selected in the instruction box. However, it is possible to specify an operand of data type DWORD or LWORD at a parameter of the instruction if the lengths of the input and output operands match. The operand will be interpreted and implicitly converted from the data type of a bit string according to the data type of the input or output parameter. The data type DWORD is for example interpreted as DINT/UDINT and LWORD as LINT/ULINT. These conversion options are also available to you when "IEC check" is enabled.

Note

For CPUs of the S7-1500 series: The data types DWORD and LWORD can only be converted to or from data type REAL or LREAL.

During the conversion, the bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If no errors occur during the conversion, the signal state of enable output ENO = 1; if an error occurs during processing, the signal state of the enable output ENO = 0.

Parameters

The following table shows the parameters of the "Convert value" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value to be converted.
OUT	Output	Bit strings, integers, floating-point numbers, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the conversion

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

Example

The following example shows conversion of an integer (16 bit) to another integer (32 bit):

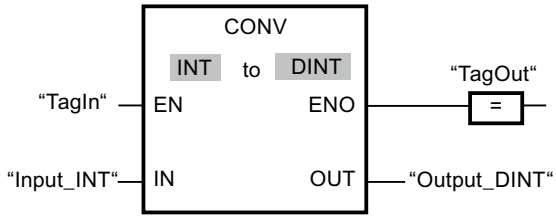

The following example shows conversion of a byte (8 bit) to the integer SINT (8 bit):

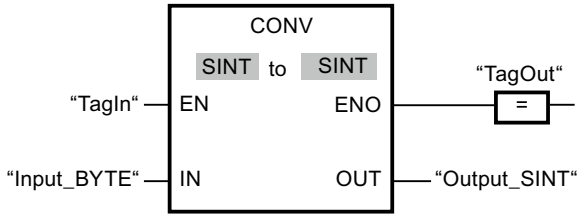

The following example shows the conversion of a byte (8 bit) to the unsigned integer USINT (8 bit):

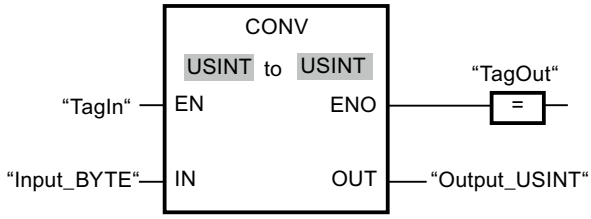

The conversions are possible, because the length of the two operands is the same.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Data type conversion for S7-1200 (Page 489)

Basic information on FBD (Page 7185)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to the nearest integer. If the input value is exactly between an even and odd number, then the even number is converted. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value to be rounded.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the rounding

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

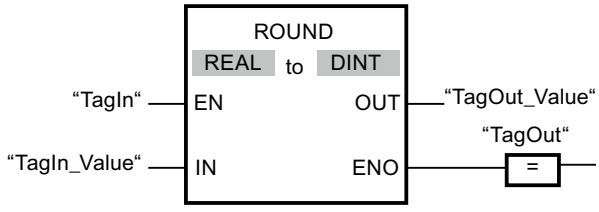

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Explicit conversions (Page 511)

Basic information on FBD (Page 7185)

CEIL: Generate next higher integer from floating-point number

Description

You can use the "Generate next higher integer from floating-point number" instruction to round the value at input IN to the next higher integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next higher integer. The result of the instruction is output at the OUT output and can be queried there. The output value can be greater than or equal to the input value.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next higher integer from floating-point number":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result with the next higher integer

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

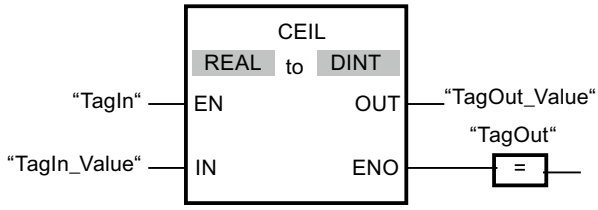

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next higher integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Explicit conversions (Page 511)

Basic information on FBD (Page 7185)

FLOOR: Generate next lower integer from floating-point number

Description

You can use the "Generate next lower integer from floating-point number" instruction to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is provided at output OUT and can be queried there. The output value can be less than or equal to the input value.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next lower integer from floating-point number":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result with the next lower integer

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

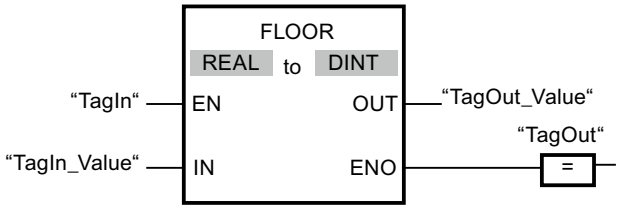

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and displayed at output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Explicit conversions (Page 511)

Basic information on FBD (Page 7185)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at input IN. The value at input IN is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to output OUT without decimal places.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Truncate numerical value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result with integer part of the floating-point number

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

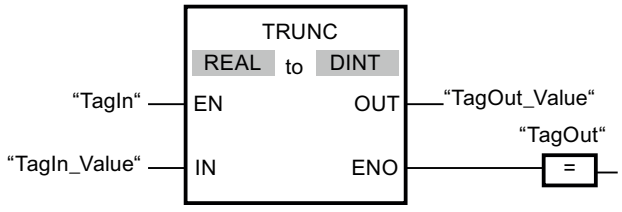

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The integer part of the floating-point number at input "TagIn_Value" is converted to an integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Explicit conversions (Page 511)

Basic information on FBD (Page 7185)

SCALE_X: Scale

Description

You can use the "Scale" instruction to scale the value at the VALUE input by mapping it to a specified value range. When the instruction "Scale" is executed, the floating-point value at input VALUE is scaled to the value range, which is defined by the parameters MIN and MAX. The result of the scaling is an integer, which is stored at output OUT.

The following figure shows an example of how values can be scaled:

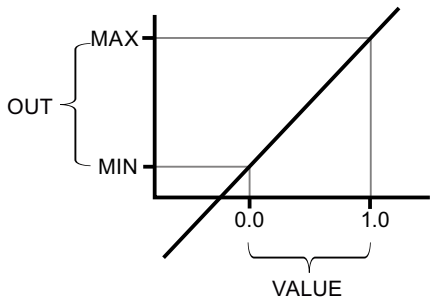

The "Scale" instruction works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input MIN is greater than or equal to the value at input MAX.

- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value at input VALUE is NaN (Not a number = result of an invalid arithmetic operation).

Note

For more information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	I, Q, M, D, L	Result of scaling

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

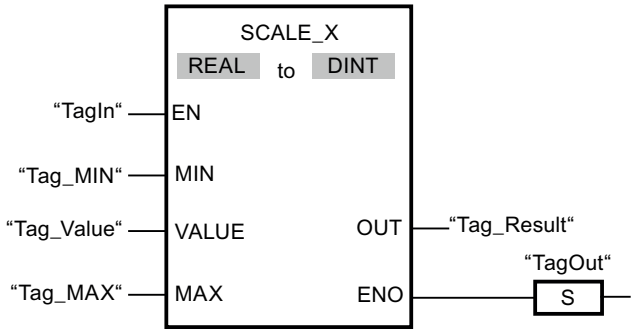

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5
MAX	Tag_MAX	30
OUT	Tag_Result	20

If the "TagIn" operand has signal state "1", the instruction is executed. The value at input "Tag_Value" is scaled to the range of values defined by the values at inputs "Tag_MIN" and "Tag_MAX". The result is stored at output "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

NORM_X: Normalize

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the parameters MIN and MAX to define the limits of a value range that is applied to the scale. Depending on the location of the normalized value in this value range, the result at output OUT is calculated and stored as a floating-point number. If the value to be normalized is equal to the value at input MIN, output OUT returns the value "0.0". If the value to be normalized is equal to the value at input MAX, output OUT returns the value "1.0".

The following figure shows an example of how values can be normalized:

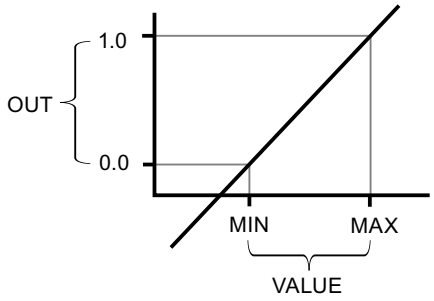

The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input MIN is greater than or equal to the value at input MAX.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- The value at input VALUE is NaN (result of an invalid arithmetic operation).

Note

For more information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the instruction "Normalize":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
MIN [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Low limit of the value range

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
VALUE [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be normalized.
MAX [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Floating-point numbers	I, Q, M, D, L	I, Q, M, D, L	Result of the normalization
[1] If you use constants in these three parameters, you only need to declare one of them.					

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

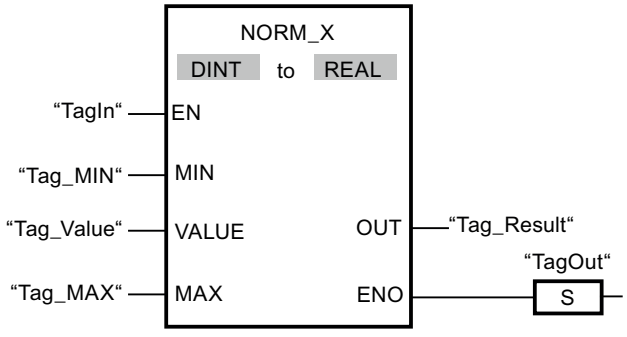

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	20
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

If the "TagIn" operand has signal state "1", the instruction is executed. The value at input "Tag_Value" is assigned to the range of values defined by the values at inputs "Tag_MIN" and "Tag_MAX". The tag value at input "Tag_Value" is normalized corresponding to the defined value range. The result is stored as a floating-point number at output "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

Legacy (S7-1500)

SCALE: Scale (S7-1500)

Description

Use the "Scale" instruction to convert the integer in the IN parameter into a floating-point number, which can be scaled in physical units between a low limit value and a high limit value. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible in the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case, the constant "K1" has the value -27648.0 and the constant "K2" the value +27648.0.
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case, the constant "K1" has the value 0.0 and the constant "K2" the value +27648.0.

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN	Input	INT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Indicates if the value at IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	I, Q, M, D, L, P	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the "K2" constant or less than the value of the "K1" constant.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

The following example shows how the instruction works:

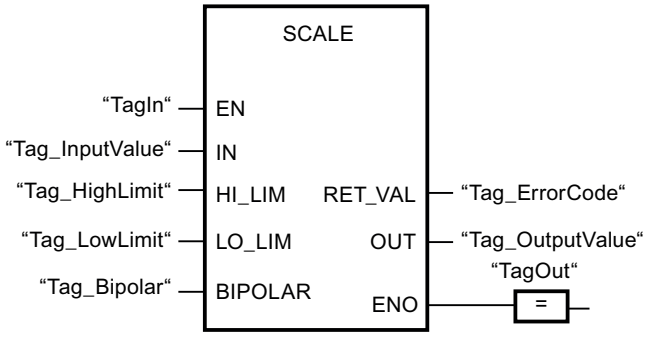

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1230)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

UNSCALE: Unscale (S7-1500)

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case, the constant "K1" has the value -27648.0 and the constant "K2" the value +27648.0.
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case, the constant "K1" has the value 0.0 and the constant "K2" the value +27648.0.

When the value at parameter IN is not within the limits defined by HI_LIM and LO_LIM, an error is output and the result is set to the nearest limit.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	I, Q, M, D, L, P	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally"
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

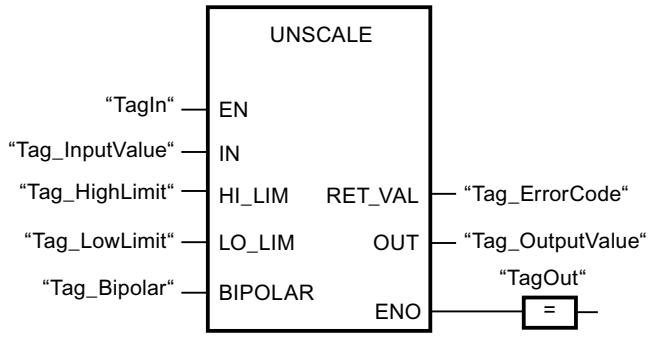

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

GET_ERR_ID: Get error ID locally (Page 1230)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

Program control operations

JMP: Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil can occur in a network.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:

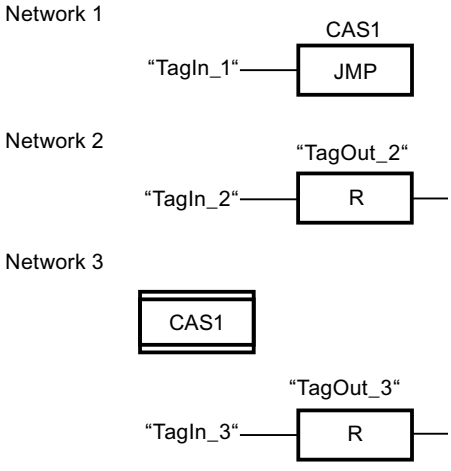

If the "TagIn_1" operand has signal state "1", the instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

JMPN: Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil can occur in a network.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of the logic operation RLO at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:

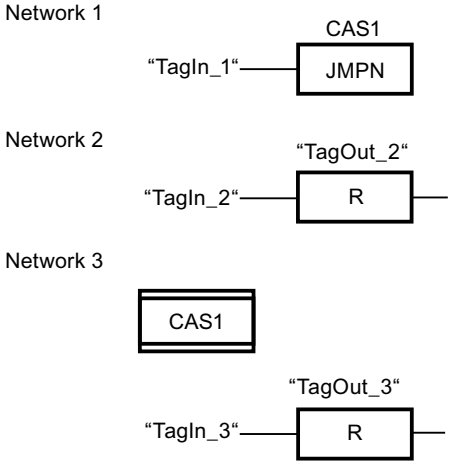

If the "TagIn_1" operand has signal state "0", the instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

LABEL: Jump label

Description

The jump label identifies a destination network in which the execution of the program can be resumed after the execution of a jump instruction.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block. You can declare up to 32 jump labels when you use a CPU S7-1200 and a maximum of 256 jump labels when you use a CPU S7-1500.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

Adhere to the following grammatical rules for jump labels:

- Letters (a to z, A to Z)
- A combination of letters and numbers. Check that the order is correct, i.e. first the letters, then the numbers (a - z, A - Z, 0 - 9).
- Special characters or a combination of letters and numbers must not be used in reverse order, i.e. first the numbers and then the letters (0-9, a - z, A - Z).

Example

The following example shows how the instruction works:

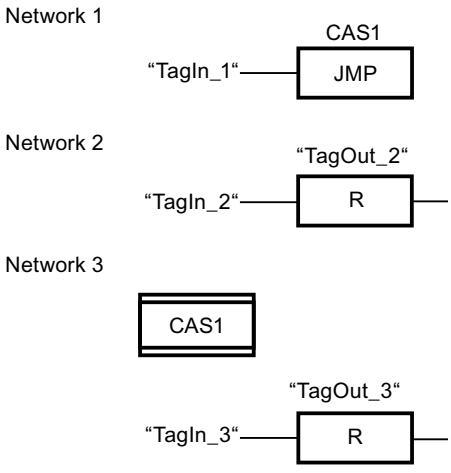

If the "TagIn_1" operand has signal state "1", the instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

JMP_LIST: Define jump list

Description

You can use the "Define jump list" instruction to define several conditional jumps and continue the program execution in a specific network depending on the value of the K parameter.

You define the jumps with jump labels (LABEL), which you specify at the outputs of the instruction box. The number of outputs can be expanded in the instruction box. You can declare up to 32 outputs when you use a CPU S7-1200 and a maximum of 256 outputs when you use a CPU S7-1500.

The numbering of the outputs begins with the value "0" and is continued in ascending order with each new output. Only jump labels can be specified at the outputs of the instruction. Instructions or operands cannot be specified.

The value of the K parameter specifies the number of the output and thus the jump label where the program execution is to be resumed. If the value in the K parameter is greater than the number of available outputs, the program execution is resumed in the next network of the block.

The "Define jump list" instruction is only executed if the signal state is "1" at the EN enable input.

Parameters

The following table shows the parameters of the "Define jump list" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, L, D or constant	Enable input
K	Input	UINT	I, Q, M, L, D or constant	Specifies the number of the output and thus the jump that is to be made.
DEST0	-	-	-	First jump label
DEST1	-	-	-	Second jump label
DESTn	-	-	-	Optional jump labels

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

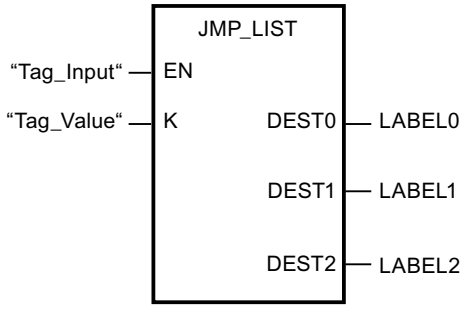

The following table shows how the instruction works using specific operand values:

Parameters	Operand/jump label	Value
K	Tag_Value	1
DEST0	LABEL0	Jump to the network that is identified with the jump label "LABEL0".
DEST1	LABEL1	Jump to the network that is identified with the jump label "LABEL1".
DEST2	LABEL2	Jump to the network that is identified with the jump label "LABEL2".

If the "Tag_Input" operand has signal state "1", the instruction is executed. The execution of the program is continued according to the value of the operand "Tag_Value" in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

SWITCH: Jump distributor

Description

You can use the "Jump distributor" instruction to define multiple program jumps to be executed depending on the result of one or more comparison instructions.

You specify the value to be compared in the K parameter. This value is compared with the values that are provided by the various inputs. You can select the comparison method for each individual input. The availability of the various comparison instructions depends on the data type of the instruction.

The following table shows the comparison instructions that are available depending on the selected data type:

Data type		Instruction	Syntax
S7-1200	S7-1500		
Bit strings	Bit strings	Equal	==
		Not equal	<>
Integers, floating-point numbers, TIME, DATE, TOD	Integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	Equal	==
		Not equal	<>
		Greater or equal	>=
		Less or equal	<=
		Greater than	>
		Less than	<

You can select the data type of the instruction from the "???" drop-down list of the instruction box. If you select a comparison instruction and the data type of the instruction is not yet defined, then the "???" drop-down list will only list those data types that are permitted for the selected comparison instruction.

Execution of the instruction begins with the first comparison and runs until a comparison condition is met. If a comparison condition is met, the subsequent comparison conditions are not considered. If none of the specified comparison conditions are fulfilled, the jump is executed at output ELSE. If not jump label is defined at output ELSE, the linear execution of the program is not interrupted but instead continued in the next network.

In its initial state the instruction box contains at least 2 outputs (DEST0 and DEST1). The number of outputs can be extended. The numbering of the outputs begins with the value "0" and is continued in ascending order with each new output. Specify jump labels (LABEL) at the outputs of the instruction. Instructions or operands cannot be specified at the outputs of the instruction.

An input is automatically inserted to each additional output. The jump programmed at an output is executed if the comparison condition of the corresponding input is fulfilled.

Parameters

The following table shows the parameters of the instruction "Jump distributor":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	Enable input
K	Input	UINT	UINT	I, Q, M, D, L or constant	Specifies the value to be compared.
<Comparison values>	Input	Bit strings, integers, floating-point numbers, TIME, DATE, TOD	Bit strings, integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	I, Q, M, D, L or constant	Input values with which the value of the parameter K is compared.
DEST0	-	-	-	-	First jump label
DEST1	-	-	-	-	Second jump label
DEST(n)	-	-	-	-	Optional jump labels: • S7-1200: n = 2 to 32 • S7-1500: n = 2 to 256
ELSE	-	-	-	-	Program jump which is executed if none of the comparison conditions are fulfilled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

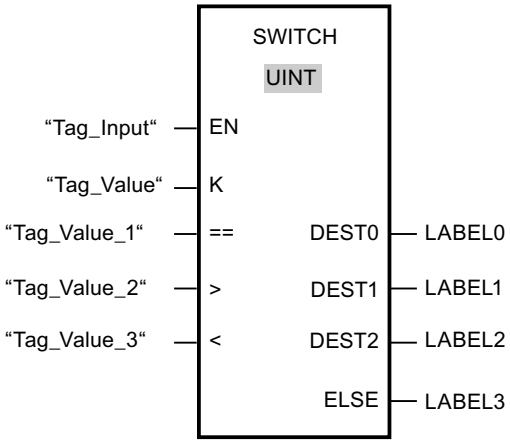

The following table shows how the instruction works using specific operand values:

Parameters	Operand/jump label	Value
K	Tag_Value	23
==	Tag_Value_1	20
>	Tag_Value_2	21
<	Tag_Value_3	19
DEST0	LABEL0	Jump to jump label "LABEL0", if the value of parameter K is equal to 20.
DEST1	LABEL1	Jump to jump label "LABEL1", if the value of parameter K is greater than 21.
DEST2	LABEL2	Jump to jump label "LABEL2", if the value of parameter K is less than 19.
ELSE	LABEL3	Jump to jump label "LABEL3", if the none of the comparison conditions are fulfilled.

If the operand "Tag_Input" changes to signal state "1", the instruction is executed. The execution of the program is continued in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

RET: Return

Description

You can use the "Return" instruction to stop the execution of a program block conditionally or absolutely. The signal state of the return value (operand) is mapped in the enable output ENO of the calling program block when the program block is exited.

There are three ways that the execution of a program block can be ended:

Termination of the program block	Description
Without a call of the instruction	The program block is exited after the execution of the last network. The enable output ENO of the calling program block is set to signal state "1".
Call of the instruction with preceding logic operation (see example)	If the preceding logic operation is fulfilled, the program execution in the currently called program block is ended and the program block is exited. (Conditional program block end) The program execution is continued in the calling program block (e.g. an OB) after the program block call. The enable output ENO of the calling program block corresponds to the operand.
Call of the instruction without preceding logic or the instruction is directly connected to the left power rail.	The program block is exited unconditionally. (Absolute program block end) The enable output ENO of the calling program block corresponds to the operand.

If an organization block (OB) is ended, another program block is selected by the execution level system and started or further executed.

- If the OB program cycle is ended, it is restarted.
- If an OB is ended that has interrupted another program block (e.g. an interrupt OB), then the interrupted program block (e.g. program cycle OB) is further executed.

Note

RET instruction in relation with JMP and JMPN

If a network already contains the "JMP: Jump if RLO = 1" or "JMPN: Jump if RLO = 0 instruction, the RET: Return" instruction must not be used. Only one jumping coil may be used in a network

---

Possible settings of the return value (operand)

The return value of the instruction can assume the following values:

- Ret (RLO, corresponds to the result of logic operation RLO. The signal state "1" is supplied to the enable output ENO of the calling program block because the RET instruction is only then executed as a conditional instruction when the condition is TRUE.)
- Ret True or Ret False (the respective value of the constant, TRUE or FALSE, is supplied to the calling program block.)
- Ret Value (the value of the Boolean tag <Operand> is supplied to the calling program block.)

To set the return value of the instruction, click the small yellow triangle of the instruction and select the desired value in the drop-down list.

The following table shows the status of the calling program block when the instruction is programmed in a network within the called program block:

RLO	Return value	ENO of the calling program block
1	RLO	1
	TRUE	1
	FALSE	0
	<Operand> Boolean tag with possible memory areas I, Q, M, D, L, T, and C	<Operand>
0	RLO	The program execution is continued in the next network of the called program block.
	TRUE	
	FALSE	
	<Operand>	

Example

The following example shows how the instruction works:

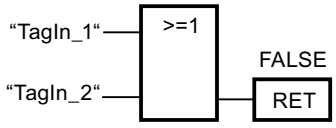

If one of the operands "TagIn_1" or "TagIn_2" have signal state "1", the instruction is executed. Program execution is ended in the called program block and continues in the calling program block. The enable output ENO of the calling program block is reset to signal state "0".

See also

Overview of the valid data types (Page 231)

Example of the use of the EN/ENO mechanism in FBD (Page 150)

Basic information on FBD (Page 7185)

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether passwords configured for the CPU are legitimized or not. In this way, you can prevent legitimized connections, even if the correct password is known.

When the instruction is called and the REQ parameter has the signal state "0", the currently set state is displayed at the output parameters. If changes have been made to the input parameters, these are not transferred to the output parameters.

When the instruction is called and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD).

- If signal state "0" is present, authorization via password is not allowed.
- If signal state "1" is present, the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Passwords that are not configured must have the signal state TRUE at the input and return the signal state TRUE at the output. The fail-safe password can only be configured for an F-CPU and must therefore always be interconnected with the signal state TRUE in a standard CPU. If the instruction returns an error, the call has no effect and the previous lock is still in effect.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the mode selector to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Plugging an empty memory card (transfer module or program card) into an S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called once again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access from HMI systems if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing unintentional lockout on an S7-1500 CPU

The settings can be made on the front panel of the CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode selector to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing unintentional lockout on an S7-1200 CPU

Because the S7-1200 CPU does not have a mode switch, protection is disabled with POWER OFF - POWER ON. This means that it is possible and advisable to prevent accidental lockout with the help of specific program sequences within your program.

To do so, program a time control either by means of a cyclic interrupt OB or a timer in the Main OB (OB 1). This gives you the option of calling the "Limit and enable password legitimation" instruction in the respective OB (e.g., OB 1 or OB 35) relatively quickly after a transition from POWER OFF to POWER ON and the associated disabling of the protection. Call the instruction in the startup OB (OB 100) to keep the time window in which the instruction is inactive and there are no limitations for password legitimation relatively small. This procedure will give you the best possible protection from unauthorized access.

If there has been an accidental lockout, you can skip the call in the startup OB (by querying an input parameter, for example) and you have the set time (for example, 10 seconds up to 1 minute) to establish a connection to the CPU before the lock becomes active once again.

If there is no timer in your program code and a lockout has occurred, insert an empty transfer card or an empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then download the user program once again from STEP 7 to the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

WARNING

Inserting an empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Once you have pulled the transfer card, the content of the transfer card is available in the internal load memory. Make sure that the card does not include a program at this point.

WARNING

Inserting an empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. Once you have pulled the previously empty program card, the internal load memory is empty.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200 CPU: The lock is disabled and the instruction must be called once again in the program (e.g., in the startup OB). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Parameters

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password

Parameters	Declaration	Data type	Memory area	Description
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported
80D0	The password for fail-safe is not configured. The signal state must be TRUE for standard CPUs.
80D1	The read/write access is not configured
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on FBD (Page 7185)

PC systems

SHUT_DOWN: Shut down target system

Description

You can use the instruction "SHUT_DOWN: Shutdown target system" to shut down the PC-based automation system and restart the S7 software controller CPU 150xS or Windows on the PC-based automation system.

The instruction is available in the "Instructions" task card under Basic instructions > Program control > Runtime control.

A restart is, for example, advisable in the following situations:

- An industrial UPS (uninterruptible power supply) reports a power failure via a digital input.
- Windows stops responding or shows a "blue screen".
- Too many error OBs are called in the user program.

Parameter

The following table shows the parameters of the "Shutdown target system" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	MODE = 1: Shut down CPU 150xS and Windows The CPU goes to STOP and saves the retentive data. Then the CPU and Windows are shut down. The system must be restarted manually. MODE = 2: Restart CPU 150xS The CPU goes to STOP and saves the retentive data. Then the CPU is shut down and restarted. MODE = 3: Restart Windows The CPU remains in RUN. Windows is restarted (as of TIA Portal V14, MODE 3 is only retained for backward compatibility. We recommend using MODE 4 or MODE 5). MODE = 4: Windows is shut down and restarted correctly. The CPU remains in RUN. MODE = 5: Restart Windows (Comparable with MODE 3; Exception: MODE 5 should only be used if Windows has crashed).
COM-MENT	Input	STRING	I, Q, M, D, L	With Mode = 1, 3 and 4, you can specify a reason for the restart. The reason is output in the Windows Event Log.
Ret_Val	Return	WORD	Q, M, D, L	Ret_Val = 0: No faults Ret_Val = 8090: The value passed to MODE is not supported. Ret_Val = 8091: Windows does not respond to the call of the Shut_Down instruction (only valid for Mode 3 and 4). Ret_Val = 8092: Please contact SIMATIC Customer Support if this error occurs (only valid for Mode 3 and 4).

RE_TRIGR: Restart cycle monitoring time

Description

The "Restart cycle monitoring time" instruction is used to restart the cycle monitoring time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

The instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Calling the instruction

The following call conditions apply:

- The following applies to a CPU of the S7-1200 series:
  In firmware versions <= 2.2, you can call the instruction only in a program cycle organization block of priority 1. This corresponds to the lowest priority of all organization blocks. If the instruction is called in an organization block with a higher priority, the instruction is not executed and the result (BR bit, enable output ENO) is always "0".
  In firmware versions >= 2.2, you can call the instruction regardless of the priority in all organization blocks.
- The following applies to a CPU of the S7-1500 series:
  You can call the instruction regardless of the priority in all organization blocks.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

STP: Exit program

Description

The instruction "Exit program" instruction is used to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

When the (RLO) at the input of the instruction is "1", the CPU changes to STOP mode and the execution of the program is terminated. The signal state at the output of the instruction is not evaluated.

If the RLO at the input of the instruction is "0", then the instruction will not be executed.

Parameters

The instruction "Exit program" has no parameters.

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a program block. This normally involves programming or access errors. If the system reports an error during the execution of the program block, detailed information is stored in the operand at the ERROR output for the first error to occur during the execution of the block since the last execution of the instruction.

Only operands of the "ErrorStruct" system data type can be specified at the ERROR output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the program block, the error information about the next error to occur is output by the instruction only after the first error that occurred has been remedied.

Note

Output ERROR

The ERROR output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO is only set if the enable input EN returns signal state "1" and error information is present. If one of these conditions does not apply, then the remaining program execution is not affected by the instruction.

Note

Activation of the local error handling

As soon as you insert the instruction in the program code of a program block, the local error handling is activated and default system reactions are ignored when errors occur.

Options for error handling

You can find an overview of the options for error handling here: Overview of mechanisms for error handling (Page 138)

For a detailed example of local error handling that contains several error handling options, refer to: Example of how to handle program execution errors (Page 167)

Parameters

The following table shows the parameters of the "Get error locally" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
ERROR	Output	ErrorStruct	D, L	Error information

Data type "ErrorStruct"

The "ErrorStruct" data type can be inserted in a global data block or in the block interface. You can insert the data type more then once if you assign a different name for the data structure each time. The data structure and the name of individual structure elements cannot be changed. If you save the error information in a global data block, it can also be read out by other program blocks.

The following table shows the structure of the "ErrorStruct" data type:

Structure component		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows whether the error occurred during a program block call. 16#01: Error during a program block call 16#00: No error during a program block call
REACTION		BYTE	Default reaction: 0: Ignore (write error) 1: Continue with substitute value "0" (read error) 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of the program block
	BLOCK_TYPE	BYTE	Program block type in which the error occurred: 1: Organization block (OB) 2: Function (FC) 3: Function block (FB)
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand

Structure component		Data type	Description
	AREA	BYTE	(C) Memory area: L: 16#40...16#7F, 16#86, 16#87, 16#8E, 16#8F, 16#C0...16#FF I: 16#81 Q: 16#82 M: 16#83 DB: 16#40, 16#84, 16#85, 16#8A, 16#8B PI: 16#01 PQ: 16#02 Technological objects: 16#04
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

The following table shows the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid reference used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.

ID* (hexadecimal)	ID* (decimal)	Description
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

Example

The following example shows how the instruction works:

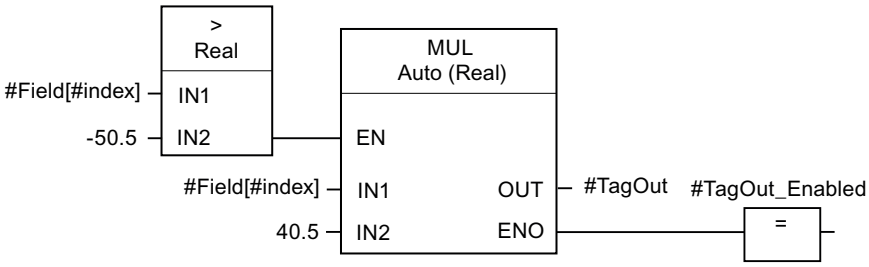

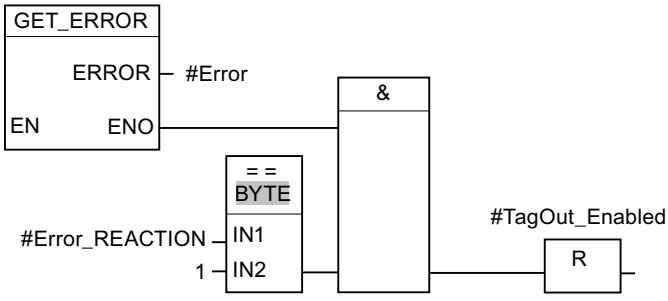

An error occurred accessing the #Field[#index] tag. The enable output ENO of the "Multiply" instruction and the #TagOut_Enabled operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error locally" instruction is evaluated with the comparison instruction "Equal". If the "#Error.REACTION" structure component has the value "1", this involves a read/access error and the #TagOut_Enabled output is reset.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the error ID of the first error that occurred in the tag is stored at the ID output.

Only operands of the WORD data type can be specified at the ID output. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The ID output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO of the instruction "Get error ID locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Options for error handling

You can find an overview of the options for error handling here:Overview of mechanisms for error handling (Page 138)

For a detailed example of local error handling that contains several error handling options, refer to: Example of how to handle program execution errors (Page 167)

Parameters

The following table shows the parameters of the "Get error ID locally" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
ID	Output	WORD	I, Q, M, D, L	Error ID

Parameter ID

The following table shows the values that can be output at the ID parameter:

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid reference used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input

ID* (hexadecimal)	ID* (decimal)	Description
2943	10563	Write error: Output
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

Example

The following example shows how the instruction works:

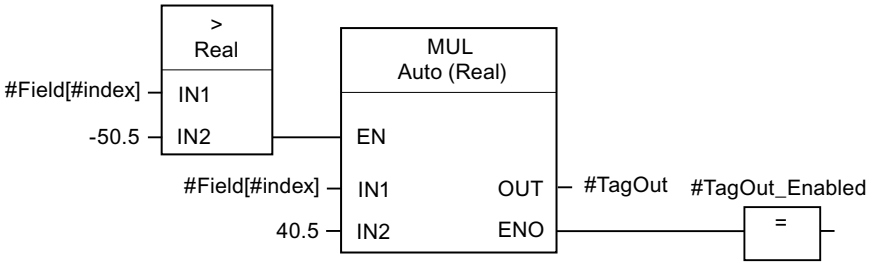

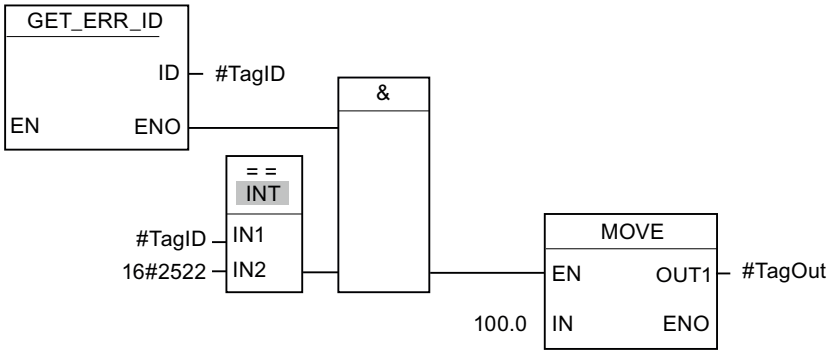

An error occurred accessing the #Field[#index] tag. The enable output ENO of the "Multiply" instruction and the #TagOut_Enabled operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error ID locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error ID locally" instruction is evaluated with the comparison instruction "Equal". If the #TagID operand returns the ID 2522, this involves a read/access error and the value "100.0" is written to the #TagOut output.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

INIT_RD: Initialize all retain data (S7-1500)

Description

The "Initialize all retain data" instruction is used to reset the retentive data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution exceeds the program cycle duration.

Parameters

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	If the input "REQ" has the signal state "1", all retentive data are reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

The following example shows how the instruction works:

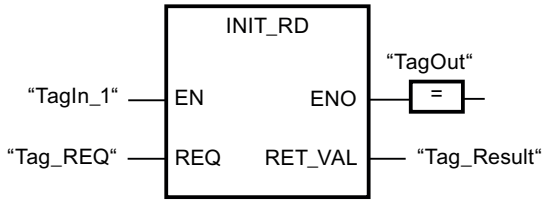

If the operands "TagIn_1" and "Tag_REQ" have signal state "1", the instruction is executed. All retentive data of all data blocks, bit memories and SIMATIC timers and counters are reset. If the instruction is executed without errors, the ENO enable output has the signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1230)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

WAIT: Configure time delay (S7-1500)

Description

The "Configure time delay" instruction pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to +32767 microseconds (µs). The smallest possible delay time depends on the CPU and corresponds to the execution time of the instruction.

The execution of the instruction can be interrupted by higher priority events and does not return any error information.

Note

Negative delay time

If you specify a negative delay time at parameter WT, the enable output ENO, or the RLO and the BR bit, will return the signal state FALSE. A negative delay time does not affect the CPU. The following instructions linked to enable output ENO are not executed in LAD or FBD.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

Example of the influence of the planned delay time

In the following example, you will see the influence of the delay time of the instruction "WAIT" in various scenarios.

The following figure is a schematic representation of the scenarios:

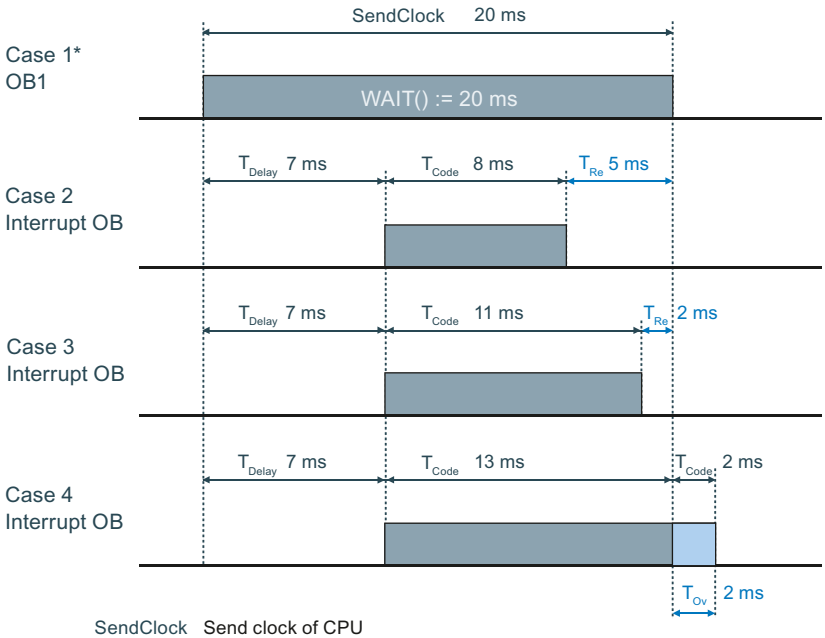

SendClock Send clock of CPU

T_{Delay} Delay time of the interrupt OB

T_{Code} Time of program execution

T_{Re} Remaining time

T_{Ov} Excess time by which the send clock is increased

* The OB1 with the instruction "Wait" is also part of cases 2, 3 and 4 and needs to be taken into account.

Remaining time = The period between the end of the planned delay time (via "WAIT") until the end of the interrupt OB

Excess time = The period between the end of the interrupt OB and the end of the planned delay time (via "WAIT").

Case 1:

The "WAIT" instruction is called in an OB1. The "WAIT" instruction can be interrupted by higher-priority OBs or higher-priority processes (e.g. System Threads). However, the delay time of the "WAIT" instruction is neither changed nor deferred.

Cases 2 and 3:

Processing of the program in OB1 is resumed after a time delay of 20 ms. This time delay is calculated by calling the "WAIT" instruction in OB1 (see OB1 with WAIT). Within these 20 ms, an interrupt OB can run its own program code. The send clock of the CPU is not changed.

Case 4:

Processing of the program in OB1 is resumed after the higher-priority process has ended. The 20 ms delay time in OB1 has elapsed, but the higher-priority process still needs to be ended. The send clock of the CPU is increased.

Note

Order of execution of system or communication processes (System Threads)

The System Threads usually use the priority "15". There are also System Threads with a higher priority than "26", but these processes cause a lower CPU load. System Threads are not displayed in the figure.

Runtime measurement of the OB1 using the "RT_INFO" instruction:

Case 2: 20 ms - 8 ms - System Threads = <12 ms. Send clock: ~20 ms.

Case 3: 20 ms - 11 ms - System Threads - <9 ms. Send clock: ~20 ms.

Case 4: 20 ms - 15 ms - System Threads - <7 ms. Send clock: ~22 ms.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

RUNTIME: Measure program runtime

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured

runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and write the value to the IN-OUT parameter MEM. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the IN-OUT parameter MEM as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The "Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overruns, the instruction returns values <= 0.0. This can occur up to once per minute for S7-1200 CPUs with firmware version <V4.2. Ignore these runtime values.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Parameters

The following table lists the parameters of the "Measure program runtime" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MEM	InOut	LREAL	I, Q, M, D, L	The content is for internal purposes only.
RET_VAL	Output	LREAL	I, Q, M, D, L	Returns the measured runtime in seconds

Example

The following example shows how the instruction works based on the runtime calculation of a program block:

Network 1:

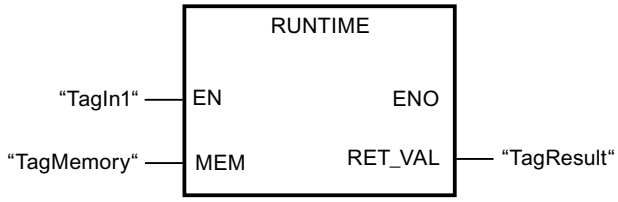

Network 2:

Network 3:

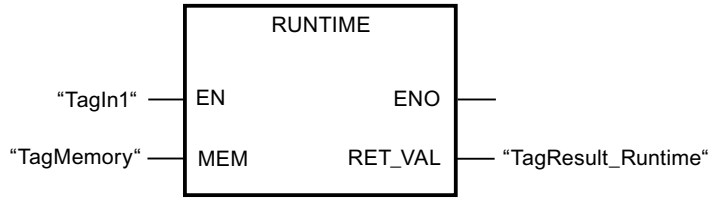

When the "TagIn1" operand in network 1 has the signal state "1", the instruction is executed. The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called in network 2.

When the FB1 program block has been executed and the "TagIn1" operand has the signal state "1", the instruction in network 3 is executed. The second call of the instruction calculates the runtime of the program block and writes the result to the output RET_VAL.

You can find a more detailed example of how you can measure the total cycle time of a program in the Siemens Industry Online Support under the following entry ID: 87668055 (https://support.industry.siemens.com/cs/ww/en/view/87668055)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

Word logic operations

AND: AND logic operation

Description

You can use the instruction "AND logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by AND logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by AND logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by AND logic. The result is stored at output "OUT".

The result bit has the signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "AND logic operation":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

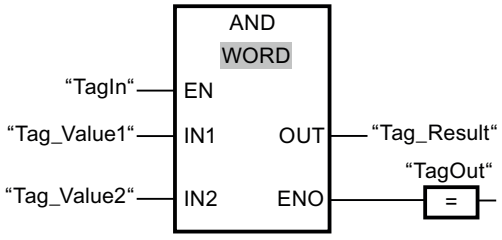

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0000 0000 0000 0101

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is linked by AND to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

OR: OR logic operation

Description

You can use the instruction "OR logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended in the instruction box. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by OR logic. The result is stored at output "OUT".

The result bit has the signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "OR logic operation":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

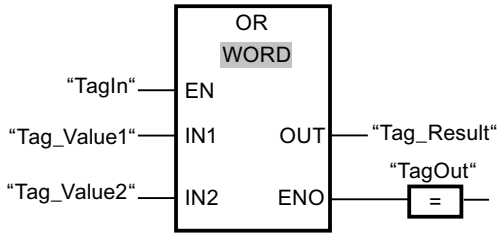

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1111

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is linked by OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

XOR: EXCLUSIVE OR logic operation

Description

You can use the instruction "EXCLUSIVE OR logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by EXCLUSIVE OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by EXCLUSIVE OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended in the instruction box. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by EXCLUSIVE OR logic. The result is stored at output "OUT".

The result bit has the signal state "1" when one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

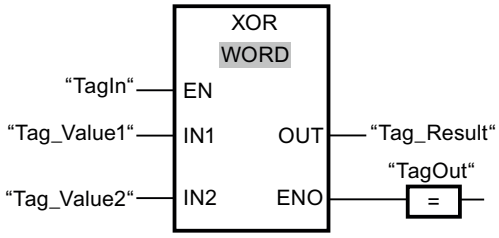

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1010

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is linked by EXCLUSIVE OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

INVERT: Create ones complement

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits at input IN. When the instruction is processed, the value at input IN is linked to EXCLUSIVE OR by a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit number). This inverts the signal state of the individual bits that are then stored at output OUT.

Parameters

The following table shows the parameters of the instruction "Create ones complement":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Bit strings, integers	I, Q, M, D, L, P	I, Q, M, D, L, P	Ones complement of the value at input IN

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

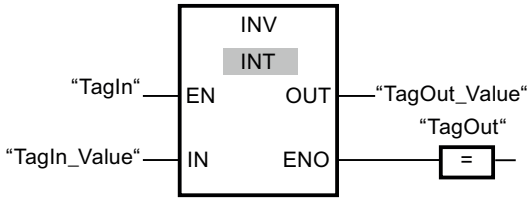

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	W#16#000F	W#16#7E
OUT	TagOut_Value	W#16#FFF0	W#16#81

If operand "TagIn" has the signal state "1", the "Create ones complement" instruction is executed. The instruction inverts the signal state of the individual bits at input "TagIn_Value" and writes the result to output "TagOut_Value". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. If the value at input IN is greater than 31, a modulo 32 instruction is executed.

Parameters

The following table shows the parameters of the "Decode" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	UINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set.
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

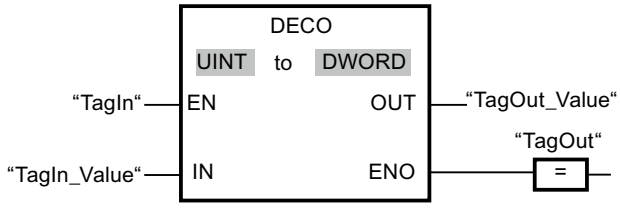

The following figure shows how the instruction works using concrete operand values:

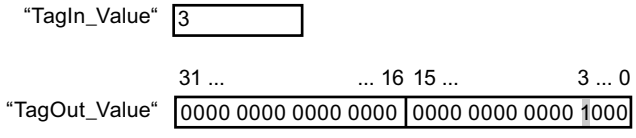

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction reads bit number "3" from the value of the operand "TagIn_Value" and sets the third bit to the value of the operand "TagOut_Value".

If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

ENCO: Encode

Description

The instruction "Encode" is used to read the bit number of the lowest bit in the input value and output it to the output OUT.

The instruction "Encode" selects the least significant bit of the value at the IN input and writes its bit number to the tag in the output OUT.

Parameters

The following table shows the parameters of the instruction "Encode":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

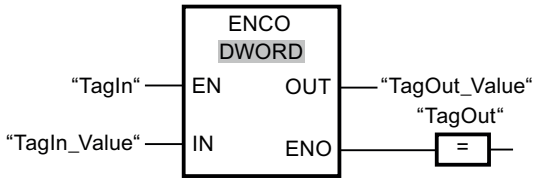

The following figure shows how the instruction works using concrete operand values:

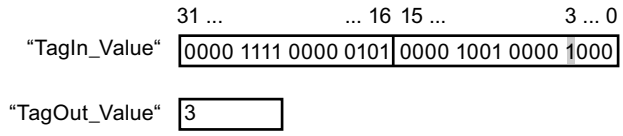

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction selects bit position "3" as the least significant bit at input "TagIn_Value" and writes the value "3" to the tag at the output "TagOut_Value.

If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

SEL: Select

Description

Depending on the signal state at switch (input G), the "Select" instruction selects one of the inputs IN0 or IN1 and moves its content to the output OUT. When the input G has the signal state "0", the value at the input IN0 is moved. When the input G has the signal state "1", the value at the input IN1 is moved to the output OUT.

The instruction can only be executed if the enable input EN has the signal state "1" and the tags of all parameters are of the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
G	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Switch

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN0	Input	Bit strings, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, DATE, TDT, CHAR, WCHAR	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, DATE, TDT, CHAR, WCHAR	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, DATE, TDT, CHAR, WCHAR	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

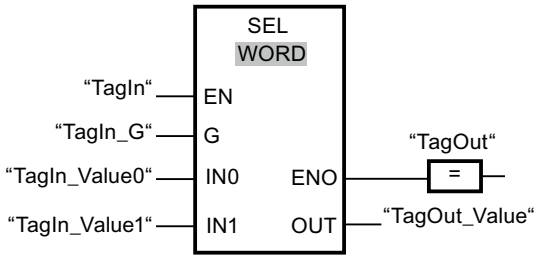

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If the "TagIn" operand has signal state "1", the instruction is executed. Based on the signal state at the input "TagIn_G", the value at input "TagIn_Value0" or "TagIn_Value1" is selected and copied to output "TagOut_Value". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

MUX: Multiplex

Description

You can use the instruction "Multiplex" to copy the content of a selected input to output OUT. In its initial state the instruction box contains at least 2 inputs (IN0 and IN1). The number of selectable inputs of the instruction box can be expanded. You can declare a maximum of 32 inputs.

The inputs are numbered automatically in the box. Numbering starts at IN0 and is incremented continuously with each new input. You use the K parameter to define the input whose content is to be copied to the OUT output. If the value of the parameter K is greater than the number of available inputs, the content of the parameter ELSE is copied to output OUT and enable output ENO is assigned the signal state "0".

The "Multiplex" instruction can only be executed if the tags have the same data type in all inputs and in the OUT output. The K parameter is an exception, since only integers can be specified for it.

The enable output ENO is reset if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The input at the K parameter is located outside the available inputs. This is the reaction regardless of whether the ELSE input will be used or not. The value at the OUT output remains unchanged.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the "Multiplex" instruction:

Parameters	Declaring	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
K	Input	Integers	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Specifies the input whose content is to be copied. • If K = 0 => Parameter IN0 • If K = 1 => Parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value

Parameters	Declaring	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
INn	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
ELSE	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Specifies the value to be copied when K > n
OUT	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Output to which the value is to be copied

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

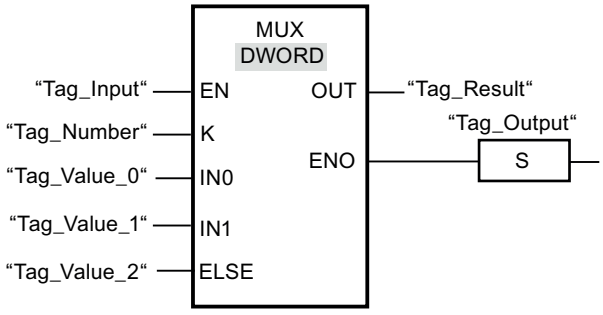

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
K	Tag_Number	1
IN0	Tag_ValueI_0	DW#16#00000000
IN1	Tag_Value_1	DW#16#003E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#003E4A7D

If the "Tag_Input" operand has signal state "1", the instruction is executed. Depending on the value of the operand "Tag_Number", the value at input "Tag_Value_1" is copied and assigned to the operand at output "Tag_Result". If no errors occur during the execution of the instruction, the outputs ENO and "Tag_Output" are set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

DEMUX: Demultiplex

Description

You can use the instruction "Demultiplex" to copy the content of the input IN to a selected output. In its initial state the instruction box contains at least 2 outputs (OUT0 and OUT1). The number of selectable outputs can be extended in the instruction box. The outputs are numbered automatically in the box. Numbering starts at OUT0 and continues consecutively with each new output. You use the K parameter to define the output to which the content of the IN input is to be copied. The other outputs are not changed. If the value of the parameter K is greater than the number of available outputs, then the content of input IN is copied to the parameter ELSE and the enable output ENO is assigned the signal state "0".

The instruction "Demultiplex" can only be executed if the tags at the input IN and at all outputs are of the same data type. The K parameter is an exception, since only integers can be specified for it.

The enable output ENO is reset if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the instruction "Demultiplex":

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
K	Input	Integers	Integers	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Specifies the output to which the input value (IN) is copied. • If K = 0 => Parameter OUT0 • If K = 1 => Parameter OUT1, etc.
IN	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P, or constant	I, Q, M, D, L, P, or constant	Input value
OUT0	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	First output
OUT1	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Second output

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
OUTn	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Optional outputs
ELSE	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, DATE	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Output to which the input value (IN) at K > n is copied.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:

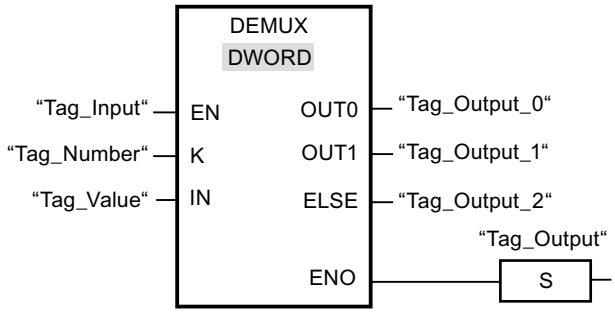

The following tables show how the instruction works using specific operand values:

Table 4-3 Input values of the "Demultiplex" instruction before network execution

Parameter	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

Table 4-4 Output values of the "Demultiplex" instruction after network execution

Parameter	Operand	Values	
OUT0	Tag_Output_0	Unchanged	Unchanged
OUT1	Tag_Output_1	DW#16#FFFFFFFF	Unchanged
ELSE	Tag_Output_2	Unchanged	DW#16#003E4A7D

If the "Tag_Input" input has the signal state"1", the "Demultiplex" instruction is executed. Depending on the value of the operand "Tag_Number", the value at input "IN" is copied to the corresponding output.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

Shift and Rotate

SHR: Shift right

Description

You can use the "Shift right" instruction to shift the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be moved.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value at the input N is greater than the number of available bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure show how the content of an operand of integer data type is shifted four bit positions to the right:

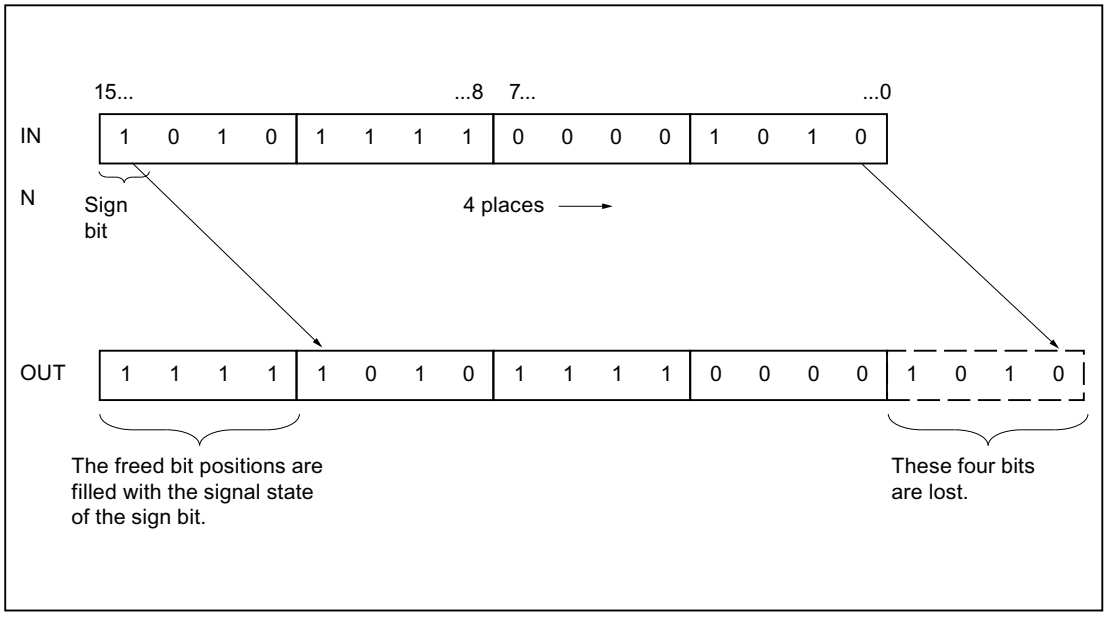

Parameters

The following table shows the parameters of the instruction "Shift right":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

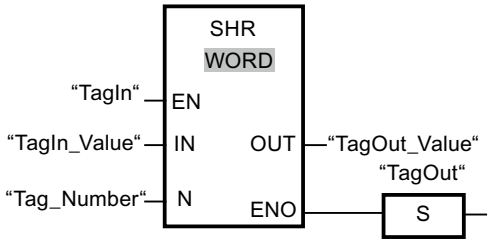

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If the "TagIn" operand has signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is shifted three bit positions to the right. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

SHL: Shift left

Description

You can use the "Shift left" instruction to shift the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be moved.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value at the input N is greater than the number of available bit positions, the operand value at input IN is shifted to the left by the available number of bit positions.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure shows how the content of an operand of the WORD data type is shifted by six bit positions to the left:

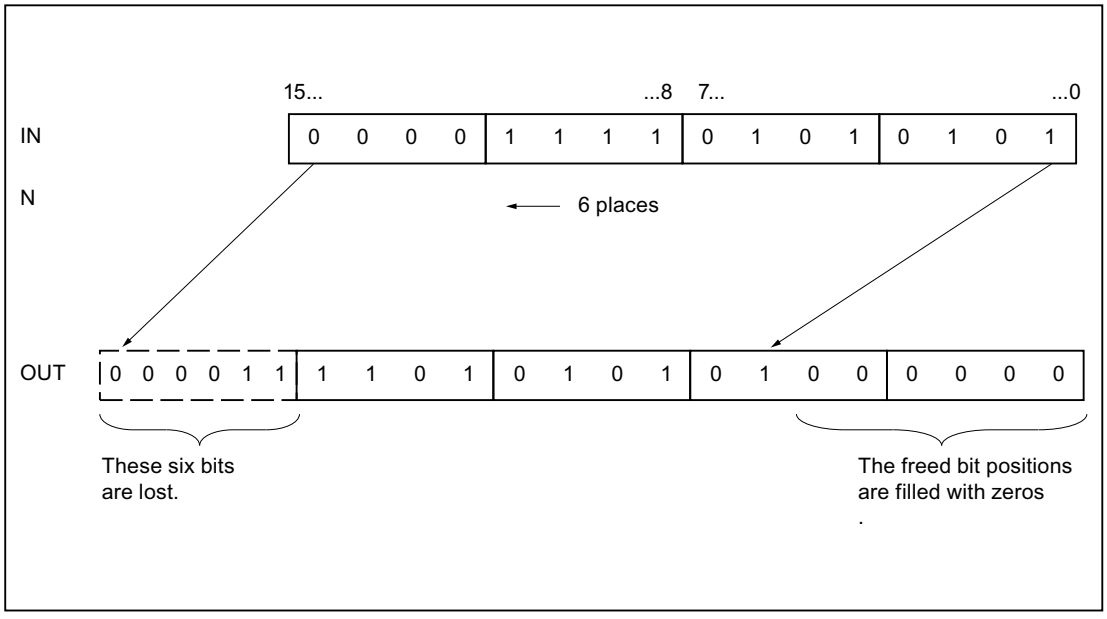

Parameters

The following table shows the parameters of the instruction "Shift left":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be shifted.
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted.
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

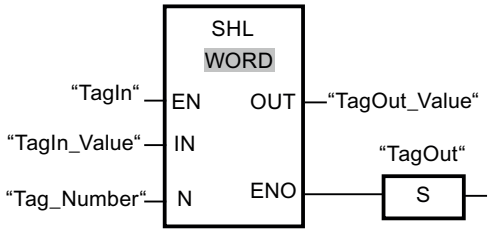

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If the "TagIn" operand has signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is shifted four bit positions to the left. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

ROR: Rotate right

Description

You can use the "Rotate right" instruction to rotate the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating on the left-hand side are filled true-to-position with the bit positions that are pushed out from the left-hand side.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value of the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:

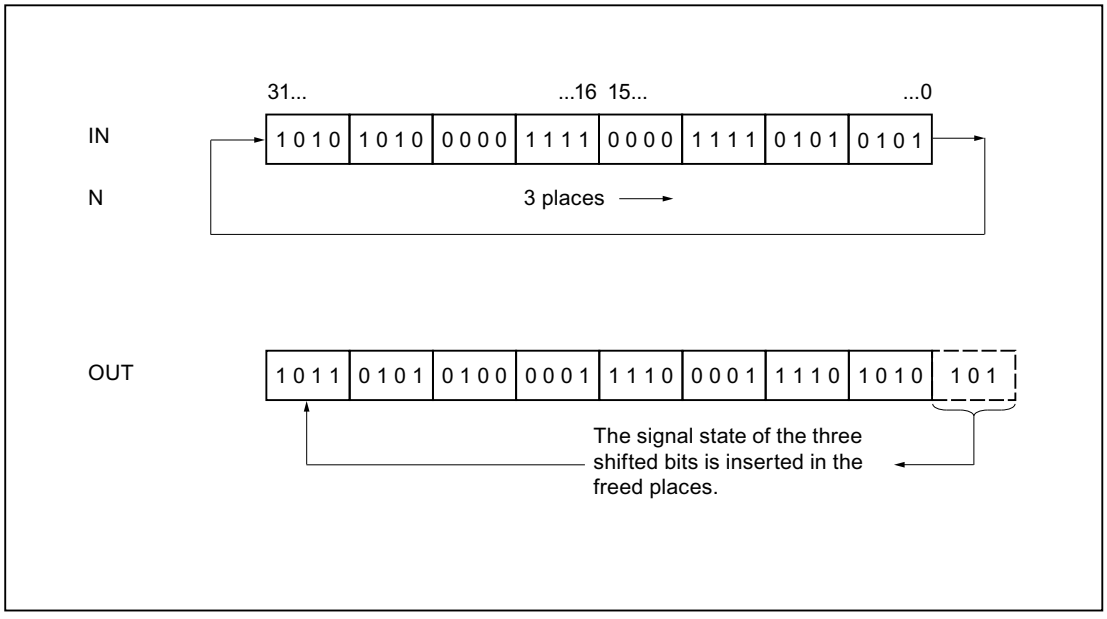

Parameters

The following table shows the parameters of the instruction "Rotate right":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

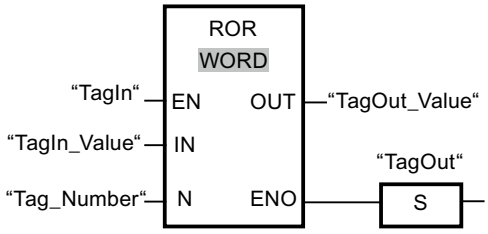

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

If the "TagIn" operand has signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the right. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

ROL: Rotate left

Description

You can use the "Rotate left" instruction to rotate the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating on the right-hand side are filled true-to-position with the bit positions that are pushed out from the left-hand side.

If the value at the input N is "0", the value at input IN is copied to the operand at output OUT.

If the value of the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:

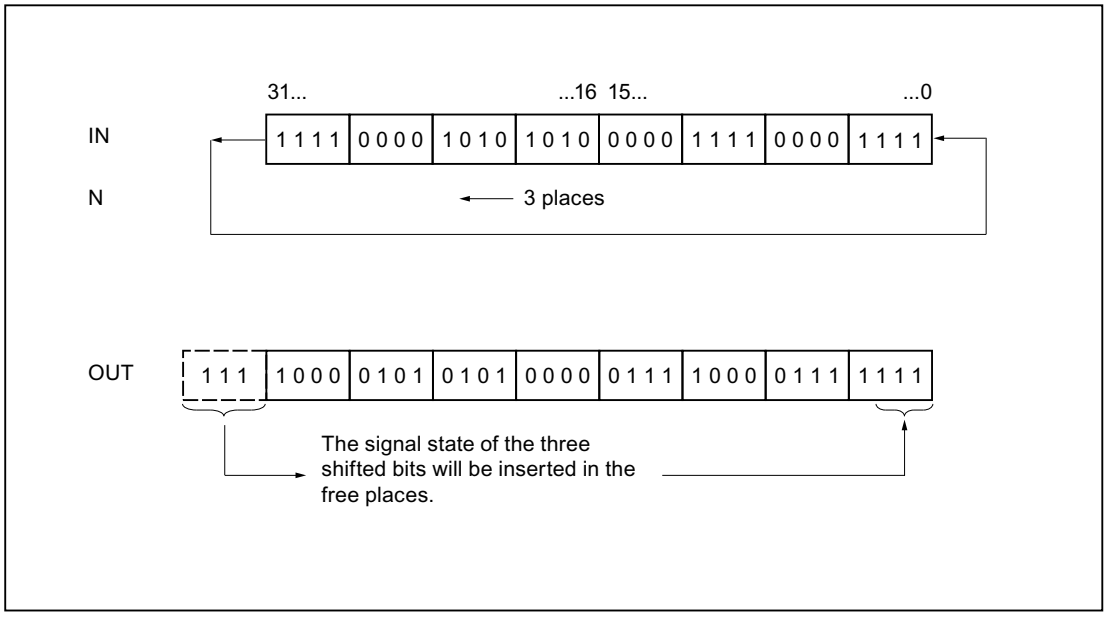

Parameters

The following table shows the parameters of the instruction "Rotate left":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be rotated.
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated.
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

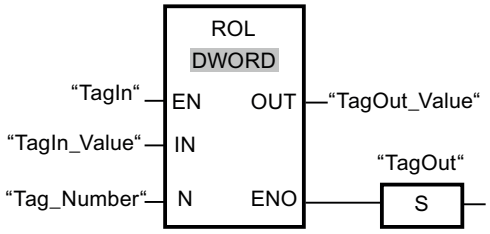

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	1010 1000 1111 0110
N	Tag_Number	5
OUT	TagOut_Value	0001 1110 1101 0101

If the "TagIn" has the signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the left. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

Legacy (S7-1500)

DRUM: Implement sequencer (S7-1500)

Description

You can use the "Implement sequencer" instruction to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the

current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). When a bit of the configurable mask has a signal state of "1", the OUT_VAL value sets/resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit at the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit at the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L or constant	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Maximum step number (for example: LST_STEP = 16#08; a maximum of 8 steps is possible.)
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L or constant	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.

Parameter	Declaration	Data type	Memory area	Description
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L or constant	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P or constant	Current numerical value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P or constant	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L or constant	Preset counter value for each step [1 to 16] where one clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

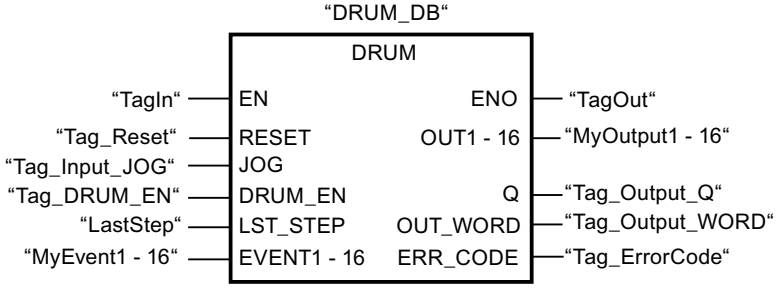

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A

Parameter	Address	Value
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE

Parameter	Address	Value
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE

Parameter	Operand	Address	Value
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on FBD (Page 7185)

DCAT: Discrete control-timer alarm (S7-1500)

Description

You can use the "Discrete control-timer alarm" to count the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET is < PT) OA and CA are influenced as follows:
  - The Q and CMD_HIS parameters are set to "1".
  - The ET, OA, and CA parameters are reset to "0".
- When the signal state of the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".
- When the signal state of the CMD and CMD_HIS parameters is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state at the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the CMD, CMD_HIS and O_FB parameters are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the CMD, CMD_HIS and C_FB parameters return signal state "0", the time difference (ms) since the last execution of the instruction is added to the value of the ET parameter. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L or constant	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L or constant	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms.
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
CMD_HIS	Static	BOOL	D, L or constant	CMD history bit

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The CMD_HIS parameter of the instance data block is set to the signal state "1" and the ET parameter is reset to "0".

Note

You can initialize static parameters in the data block.

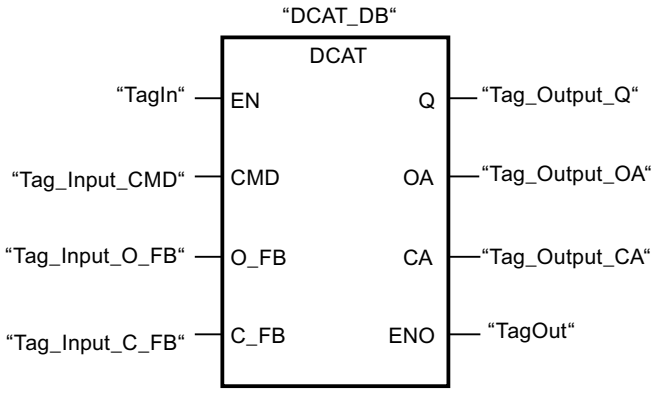

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 231)
Basics of the EN/ENO mechanism (Page 141)
Basic information on FBD (Page 7185)

MCAT: Motor control-timer alarm (S7-1500)

Description

The "Motor control-timer alarm" instruction is used to accumulate the time from the point at which one of the command inputs (opening or closing) is switched on. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

The "Motor control-timer alarm" instruction returns no error information.

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameters	Declaration	Data type	Memory space	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
O_CMD	Input	BOOL	I, Q, M, D, L or constant	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L or constant	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L or constant	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	A signal state of "0" indicates an error condition.
ET	Static	DINT	D, L or constant	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time

Parameters	Declaration	Data type	Memory space	Description
O_HIS	Static	BOOL	D, L or constant	"Open" history bit
C_HIS	Static	BOOL	D, L or constant	"Close" history bit

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

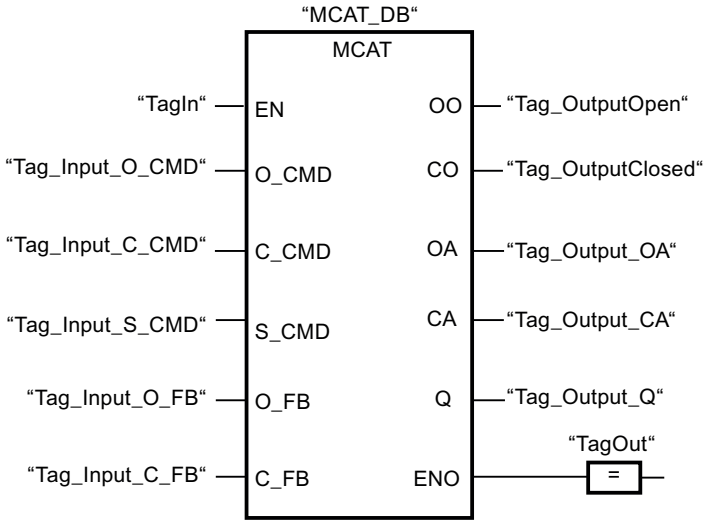

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

IMC: Compare input bits with the bits of a mask (S7-1500)

Description

You can use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.

Parameter	Declaration	Data type	Memory area	Description
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on FBD (Page 7185)

SMC: Compare scan matrix (S7-1500)

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found, the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written to the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information

Parameter	Declaration	Data type	Memory area	Description
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P or constant	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on FBD (Page 7185)

LEAD_LAG: Lead and lag algorithm (S7-1500)

Description

You can use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the instruction "Lead and lag algorithm" is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and

SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the instruction "Lead and lag algorithm" in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

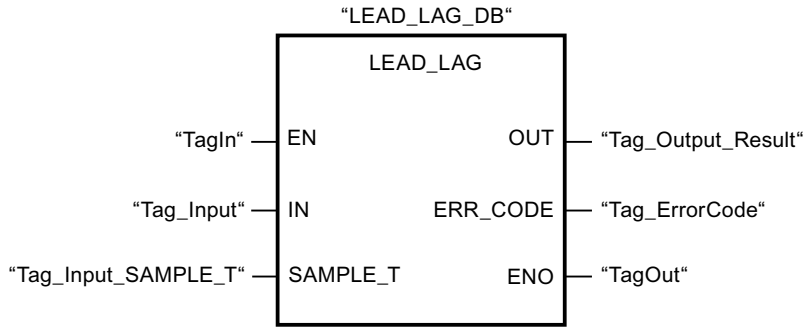

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameters	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameters	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameters	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on FBD (Page 7185)

SEG: Create bit pattern for seven-segment display (S7-1500)

Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	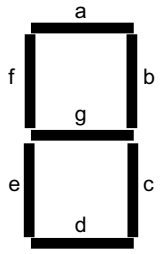
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

Parameters

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:

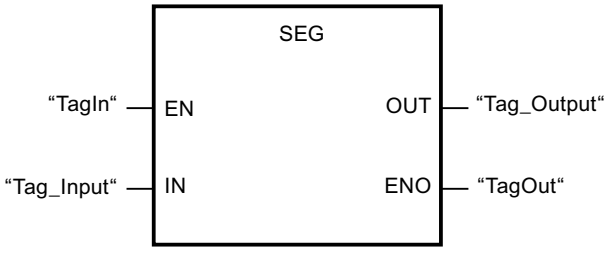

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

BCDCPL: Create tens complement (S7-1500)

Description

You can use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:

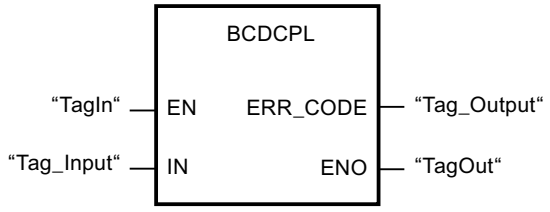

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on FBD (Page 7185)

BITSUM: Count number of set bits (S7-1500)

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output at the RET_VAL parameter.

Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C or constant	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted.
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of bits to be set

Example

The following example shows how the instruction works:

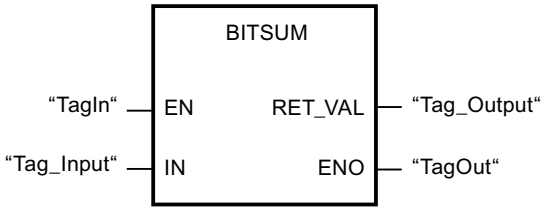

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Basic information on FBD (Page 7185)

4.1.3.3 STL (S7-1500)

Bit logic operations (S7-1500)

R_TRIG: Detect positive signal edge (S7-1500)

Description

With the "Detect positive signal edge" instruction, you can detect a state change from "0" to "1" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "0" to "1", a positive signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

Parameters

The following table shows the parameters of the "Detect positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

```
STL                              Explanation
CALL R_TRIG, "R_TRIG_DB"         // The instruction is called.
CLK := "TagIn"                   // Positive signal edge is detected.
Q := "TagOut"                    // Signal state "1" on positive signal edge.
```

The previous state of the tag at the CLK input is stored in the "R_TRIG_DB" tag. If a change is the signal state from "0" to "1" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

F_TRIG: Detect negative signal edge (S7-1500)

Description

With the "Detect negative signal edge" instruction, you can detect a state change from "1" to "0" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "1" to "0", a negative signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

Parameters

The following table shows the parameters of the "Detect negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

```
STL                              Explanation
CALL F_TRIG, "F_TRIG_DB"         // The instruction is called.
CLK := "TagIn"                   // Negative signal edge is detected.
Q := "TagOut"                    // Signal state "1" on negative signal edge.
```

The previous state of the tag at the CLK input is stored in the "F_TRIG_DB" tag. If a change in the signal state from "1" to "0" is detected in the "TagIn" operand, the "TagOut" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

Timer operations (S7-1500)

TP: Generate pulse (S7-1500)

Description

You can use the "Generate pulse" instruction to set the Q output for a programmed duration. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. The Q parameter is set for the time PT, regardless of the subsequent changes in the input signal. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the time duration PT is active.

You can query the current time value at the ET output. The time value starts at T#0s and ends when the value of the time PT is reached. When the time PT has elapsed and the signal state at input IN is "0", the ET output is reset.

Use the "Call block" (CALL) instruction in the program code to call the "Generate pulse" instruction.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME or TP_LTIME in the "Static" section of a block (for example, #MyTP_TIMER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens.

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate pulse" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Parameters

The following table shows the parameters of the "Generate pulse" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Pulse output
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current timer value

Pulse timing diagram

The following figure shows the behavior of the "Generate pulse" instruction after the start:

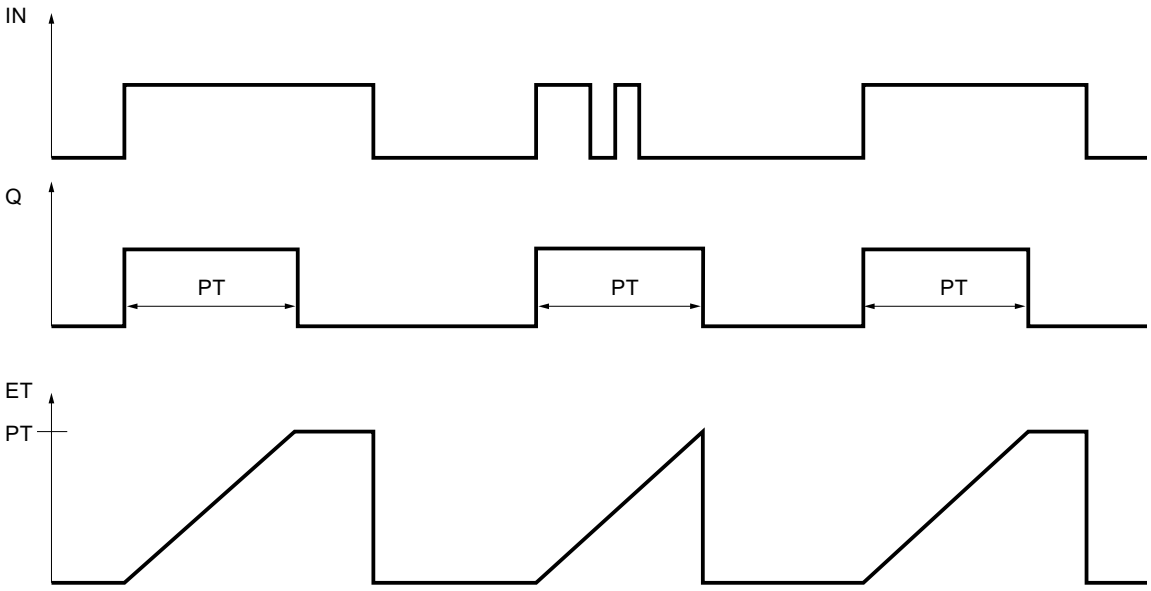

Example

The following example shows how the instruction works:

STL	Explanation
`CALL TP, "TP_DB"`	`// The instruction is called. The "TP_DB" data block is assigned to the instruction.` `// Select the required data type from the "???" drop-down list.`
`IN := "Tag_Start"`	`// The instruction is executed on a rising signal edge of the operand.`
`PT := "Tag_PresetTIME"`	`// Duration of the pulse`
`Q := "Tag_Output"`	`// The operand is set for the duration that was specified by the "Tag_PresetTIME" operand.`

STL	Explanation
ET := "Tag_ElapsedTIME"	// Current timer value

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

TON: Generate on-delay (S7-1500)

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the programmed time PT. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the PT duration has expired, the Q parameter returns signal state "1". The Q parameter remains set as long as the start input IN is still "1". When the signal state at the start input changes from "1" to "0", the Q parameter is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The time value starts at T#0s and ends when the value of the time PT is reached. The ET parameter is reset as soon as the signal state of the IN parameter changes to "0".

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

In the program code, you call the "Generate on-delay" instruction with the "Call block" (CALL) instruction.

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or TON_LTIME in the "Static" section of a block (for example, #MyTON_TIMER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens.

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Parameters

The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Signal state that is delayed by the time PT.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current timer value

Pulse timing diagram

The following figure shows the behavior of the "Generate on-delay" instruction after the start:

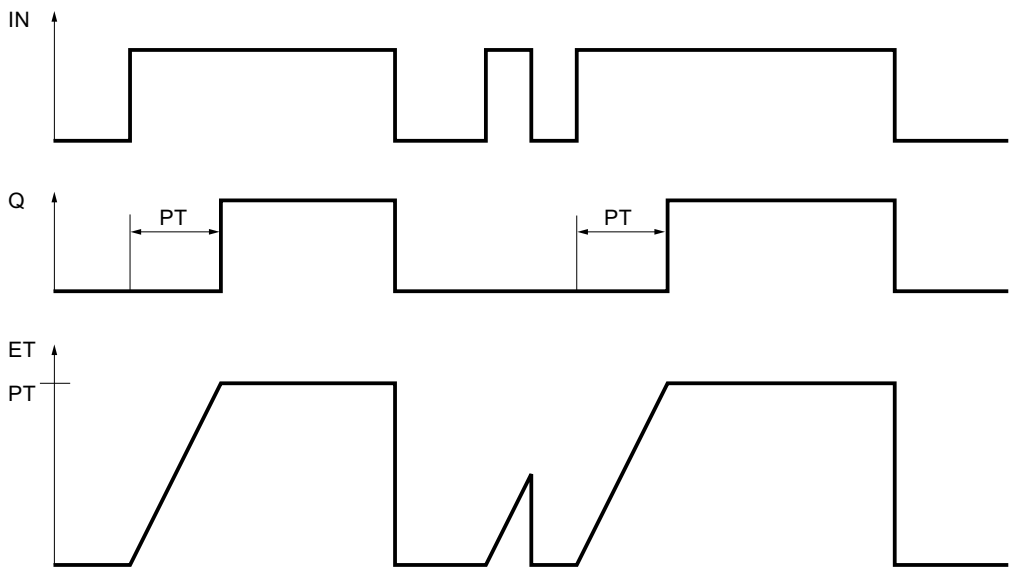

Example

The following example shows how the instruction works:

STL	Explanation
CALL "TON", "TON_DB"	// The instruction is called. The "TON_DB" data block is assigned to the instruction.
	// Select the required data type from the "???" drop-down list.
IN := "Tag_Start"	// The instruction is executed on a rising signal edge of the operand.
PT := "Tag_PresetTIME"	// Specifies the time by which the rising signal edge of the IN parameter is delayed.
Q := "Tag_Output"	// The operand is set if the duration PT that was specified by the "Tag_PresetTIME" tag has expired.
	// The Q parameter remains set as long as the "Tag_Start" variable is "1".
	// When the signal state at the start input changes from "1" to "0", the operand of the Q parameter is reset.
ET := "Tag_ElapsedTIME"	// Current timer value

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

TOF: Generate off-delay (S7-1500)

Description

You can use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed time PT. The Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state at the IN parameter changes back to "0", the programmed time PT starts. The Q parameter remains set as long as time PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before time PT expires, the timer is reset. The signal state of the Q parameter remains set to "1".

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time PT is reached. When the time PT expires, the ET parameter

remains set to the current value until the IN parameter changes back to "1". If input IN changes to "1" before the time PT elapses, the ET output is reset to the value T#0s.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

In the program code, you call the "Generate off-delay" instruction with the "Call block" (CALL) instruction.

Each call of the "Generate off-delay" instruction must be assigned to an IEC Timer in which the instruction data is stored. An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or TOF_LTIME in the "Static" section of a block (for example, #MyTOF_TIMER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens.

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Parameters

The following table shows the parameters of the "Generate off-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Signal state that is delayed by the time PT.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current timer value

Pulse timing diagram

The following figure shows the behavior of the "Generate off-delay" instruction after the start:

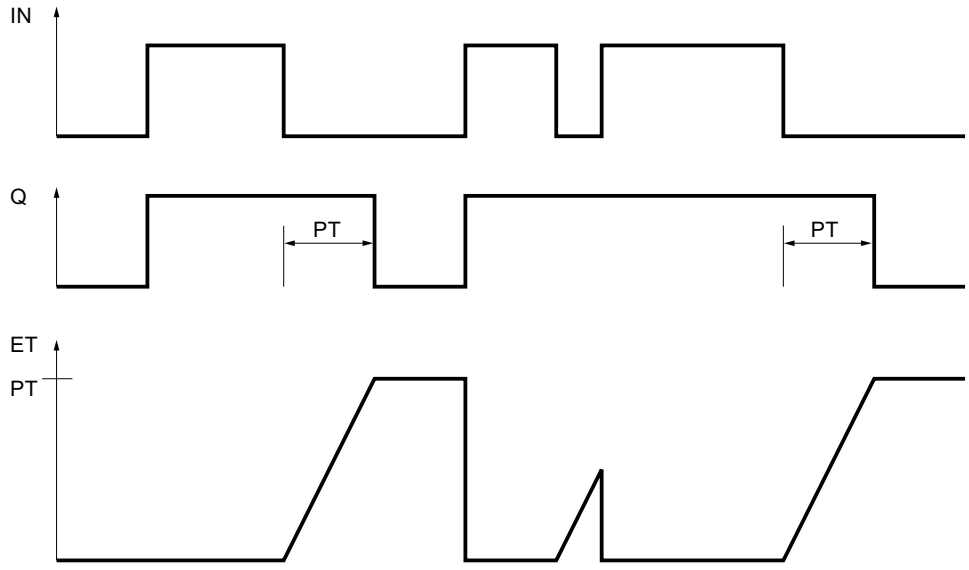

Example

The following example shows how the instruction works:

STL	Explanation
CALL TOF, "TOF_DB"	// The instruction is called. The "TOF_DB" data block is assigned to the instruction.
	// Select the required data type from the "???" drop-down list.
IN := "Tag_Start"	// The instruction is executed on a rising signal edge of the operand.
PT := "Tag_PresetTIME"	// Specifies the time by which the falling signal edge of the IN parameter is delayed.
Q := "Tag_Output"	// The operand is set when the instruction is started by a rising signal edge of the IN parameter.
	// When the value at the IN parameter changes from "1" to "0", the "Tag_Output" operand remains set as long as the time specified by the "Tag_PresetTIME" tag is running.
	// When the time of the PT parameter expires, the operand is reset.
ET := "Tag_ElapsedTIME"	// Current timer value

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

TONR: Time accumulator (S7-1500)

Description

The instruction "Time accumulator" is used to accumulate time values within a period set by the parameter PT. When the signal state at the IN input changes from "0" to "1" (positive signal edge), the instruction is executed and time PT starts. While the time PT is running, the time values are accumulated that are recorded when the IN input has signal state "1". The accumulated time is written to the ET output and can be queried there. When the duration PT expires, the output Q has signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes from "1" to "0" (negative signal edge).

The R input resets the ET and Q outputs regardless of the signal state of the start input.

In the program code, you call the "Time accumulator" instruction with the "Call block" (CALL) instruction.

Each call of the "Time accumulator" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or TONR_LTIME in the "Static" section of a block (for example, #MyTONR_TIMER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens.

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Maximum duration of time recording The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Output that is set when time PT expires.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Accumulated time

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:

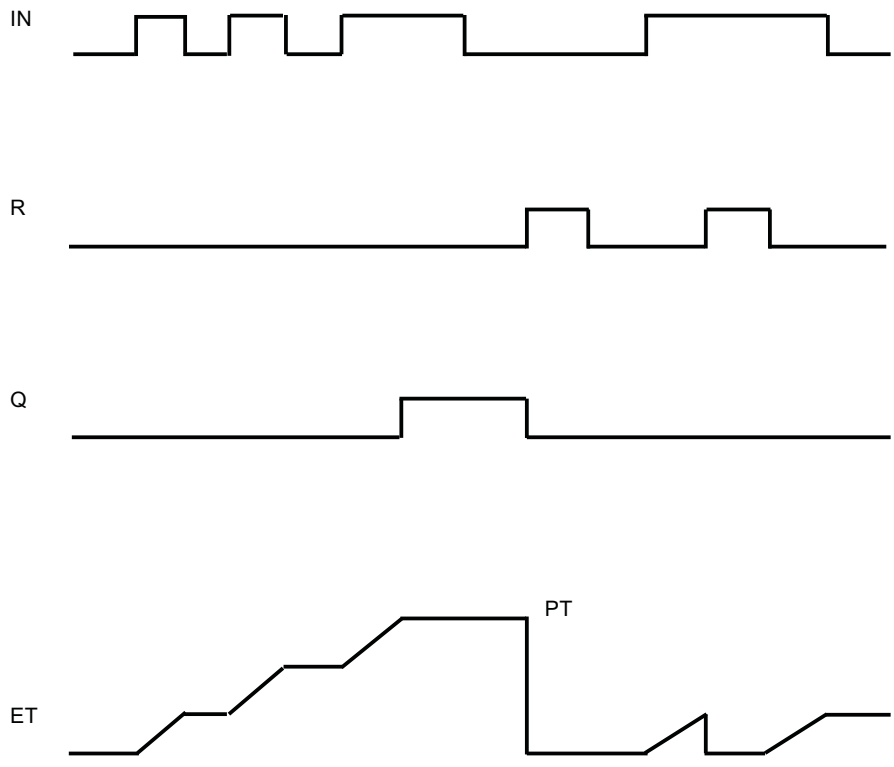

Example

The following example shows how the instruction works:

STL	Explanation
CALL TONR, "TONR_DB"	// The instruction is called. The "TONR_DB" data block is assigned to the instruction.
	// Select the required data type from the "???" drop-down list.
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand and the time duration at the PT input is started.
	While the time PT is running, the time values are accumulated that are recorded when the IN input has signal state "1".
R := "Tag_Reset"	// The R input resets the outputs ET and Q regardless of the signal state at the start input.
PT := "Tag_PresetTIME"	// Specifies how long the time values are accumulated.
Q := "Tag_Output"	// The operand is set if the time at PT has expired.
	// The Q parameter remains set to "1", even when the signal state at the IN parameter changes from "1" to "0" (negative signal edge).
ET := "Tag_ElapsedTIME"	// The accumulated time is written to output ET and can be queried there.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

RESET_TIMER: Reset timer (S7-1500)

Description

You can use the "Reset timer" instruction to reset an IEC timer to "0". The structure components of the timer in the specified data block are reset to "0".

The instruction does not influence the RLO. At the TIMER parameter, the "Reset timer" instruction is assigned an IEC timer declared in the program.

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

Parameters

The following table shows the parameters of the "Reset timer" instruction:

Parameters	Declaration	Data type	Memory area	Description
TIMER	Output	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer that is reset

You can select the data type of the instruction from the "???" drop-down list.

Example

The following example shows how the instruction works:

STL	Explanation
CALL TON, "TON_DB"	// The instruction is called. The "TON_DB" instance data block is assigned to the instruction.
	// Select the required data type from the "???" drop-down list.
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand "Tag_Start".
PT := "Tag_PresetTIME"	// The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PresetTIME".
Q := "Tag_Output"	// The operand "Tag_Output" is set if the time duration PT specified by the operand "Tag_PresetTIME" has expired.
	// The parameter Q remains set as long as the "Tag_Start" operand still has the signal state "1".
	// When the signal state at the start input changes from "1" to "0", the operand of the Q parameter is reset.
ET := "Tag_ElapsedTIME"	// Current timer value
A "Tag_Input_1"	// When the operand "Tag_Input_1" and
A "Tag_Input_2"	// The operand "Tag_Input_2" returns the signal state "1",
CALL RESET_TIMER	// The "Reset timer" instruction is called.
	// Select the required data type from the "???" drop-down list.
TIMER := "TON_DB"	// IEC timer is reset.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

PRESET_TIMER: Load time duration (S7-1500)

Description

You can use the "Load time duration" instruction to set the time for an IEC timer. The instruction is executed in each cycle. The instruction writes the specified time to the structure of the specified IEC timer.

You assign an IEC timer declared in the program to the "Load time duration" instruction.

The instruction data is updated only when the instruction is called and each time the assigned IEC timer is accessed. The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Note

If the specified IEC timer is running while the instruction executes, the instruction overwrites the current time of the specified IEC timer. This can change the timer status of the IEC timer.

Parameters

The following table shows the parameters of the "Load time duration" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs
<IEC timer>	Output	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer whose duration is set

You can select the data type of the instruction from the "???" drop-down list.

Example

The following example shows how the instruction works:

STL	Explanation
CALL TON, "TON_DB"	// The instruction is called. The "TON_DB" instance data block is assigned to the instruction.
	// Select the required data type from the "???" drop-down list.
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand "Tag_Start".
PT := "Tag_PresetTIME"	// The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PresetTIME".
Q := "Tag_Output"	// The operand "Tag_Output" is set if the time duration PT specified by the operand "Tag_PresetTIME" has expired.
	// The parameter Q remains set as long as the "Tag_Start" operand still has the signal state "1".
	// When the signal state at the start input changes from "1" to "0", the operand of the Q parameter is reset.
ET := "Tag_ElapsedTIME"	// Current timer value
A "Tag_Input_1"	// When the operand "Tag_Input_1" and
A "Tag_Input_2"	// The operand "Tag_Input_2" returns the signal state "1",
CALL PRESET_TIMER	// The "Time accumulator" instruction is called.
	// Select the required data type from the "???" drop-down list.
PT := "Tag_PresetTIME_new"	// The instruction writes the time duration "Tag_PresetTIME_new" in the instance data block "TON_DB", thereby overwriting the time value of the operand "Tag_PresetTIME" within the instance data block. The signal state of the timer status may therefore change at the next query or when "MyTimer".Q or "MyTimer".ET are accessed.
TIMER := "TON_DB"	// IEC timer is reset.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

Counter operations (S7-1500)

CTU: Count up (S7-1500)

Description

You can use the "Count up" instruction to increment the value at the CV parameter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the parameter CV is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the output CV. When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current counter value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the signal state of the R parameter is "1", the signal state of the CU parameter has no effect on the instruction.

In the program code, you call the "Count up" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "1" set for the R parameter of the instruction. If instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Instance data block of system data type IEC_<Counter> (shared DB / single instance)

The system data type of the instance data block is derived from the data type of the instruction:

Data type of the instruction	System data type of the instance data block (shared DB)
SINT / USINT	IEC_SCOUNTER / IEC_USCOUNTER
INT / UINT	IEC_COUNTER / IEC_UCOUNTER
DINT / UDINT	IEC_DCOUNTER / IEC_UDCOUNTER
LINT / ULINT	IEC_LCOUNTER / IEC_ULCOUNTER

When you set up the IEC Counter in a single instance, the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive.

For additional information on setting retentivity in an instance data block, refer to "See also".

Local tag (multi-instance)

The data type of the local tag is derived from the data type of the instruction:

Data type of the instruction	Data type of the local tag
SINT / USINT	CTU_SINT / CTU_USINT / IEC_SCOUNTER / IEC_USCOUNTER
INT / UINT	CTU_INT / CTU_UINT / IEC_COUNTER / IEC_UCOUNTER
DINT / UDINT	CTU_DINT / CTU_UDINT / IEC_DCOUNTER / IEC_UDCOUNTER
LINT / ULINT	CTU_LINT / CTU_ULINT / IEC_LCOUNTER / IEC_ULCOUNTER

When you set up the IEC Counter as multi-instance in a function block with "optimized block access", it is defined as retentive in the block interface.

Declaring IEC Counters

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens. You can then declare an IEC Counter as follows:

- Single instance: Declaration of an instance data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Multi-instance: Declaration as a local tag of the type CTU_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTU_COUNTER)

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

Example

The following example shows how the instruction works:

STL	Explanation
CALL CTU, "CTU_DB"	// The instruction is called. The "CTU_DB" data block is assigned to the instruction.
	// Select the required data type from the "???" drop-down list.
CU := "Tag_StartCTU"	// When the signal state of the "Tag_StartCTU" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one.
	// The counter value is incremented until the high limit of INT = 32767 is reached.
R := "Tag_ResetCounter"	// When the signal state of the "Tag_ResetCounter" operand changes to "1", the "Tag_CounterValue" operand is reset to "0".
PV := "Tag_PresetValue"	// The operand determines the value at which the operand of the Q parameter is set.
Q := "Tag_CounterStatus"	// The operand is set as long as the current counter value is greater than or equal to the value of the PV parameter.
CV := "Tag_CounterValue"	// Current counter value

See also

Overview of the valid data types (Page 231)

Setting retentivity in an instance data block (Page 7037)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

CTD: Count down (S7-1500)

Description

The "Count down" instruction is used to decrement the value at the parameter CV. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the signal state of the LD parameter is "1", the signal state of the CD parameter has no effect on the instruction.

In the program code, you call the "Count down" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

The operating system resets the instances of the "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case the desired initial value for the CV parameter is specified in the PV parameter. If instances of the "Count down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Instance data block of system data type IEC_<Counter> (shared DB / single instance)

The system data type of the instance data block is derived from the data type of the instruction:

Data type of the instruction	System data type of the instance data block (shared DB)
SINT / USINT	IEC_SCOUNTER / IEC_USCOUNTER
INT / UINT	IEC_COUNTER / IEC_UCOUNTER
DINT / UDINT	IEC_DCOUNTER / IEC_UDCOUNTER
LINT / ULINT	IEC_LCOUNTER / IEC_ULCOUNTER

When you set up the IEC Counter in a single instance, the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive.

For additional information on setting retentivity in an instance data block, refer to "See also".

Local tag (multi-instance)

The data type of the local tag is derived from the data type of the instruction:

Data type of the instruction	Data type of the local tag
SINT / USINT	CTD_SINT / CTD_USINT / IEC_SCOUNTER / IEC_USCOUNTER
INT / UINT	CTD_INT / CTD_UINT / IEC_COUNTER / IEC_UCOUNTER
DINT / UDINT	CTD_DINT / CTD_UDINT / IEC_DCOUNTER / IEC_UDCOUNTER
LINT / ULINT	CTD_LINT / CTD_ULINT / IEC_LCOUNTER / IEC_ULCOUNTER

When you set up the IEC Counter as multi-instance in a function block with "optimized block access", it is defined as retentive in the block interface.

Declaring IEC Counters

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens. You can then declare an IEC Counter as follows:

- Single instance: Declaration of an instance data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Multi-instance: Declaration as a local tag of the type CTD_<Data_type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTD_COUNTER)

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L or constant	Count input
LD	Input	BOOL	I, Q, M, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

Example

The following example shows how the instruction works:

STL	Explanation
CALL CTD, "CTD_DB"	// The instruction is called. The "CTD_DB" data block is assigned to the instruction.
	// Select the required data type from the "???" drop-down list.
CD := "Tag_StartCTD"	// When the signal state of the "Tag_StartCTD" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterValue" operand is decremented by one.
	// The counter value of the CV parameter is decremented until the low limit of INT = -32768 is reached.
LD := "Tag_LoadPV"	// When the signal state of the "Tag_LoadPV" operand changes to "1", the "Tag_CounterValue" operand is set to the value of the "Tag_PresetValue" operand.
PV := "Tag_PresetValue"	// Specifies the value to which the counter is set when the signal state is "1" in the LD parameter.
Q := "Tag_CounterStatus"	// The operand is set when the current counter value is less than or equal to zero.
CV := "Tag_CounterValue"	// Current counter value

See also

Overview of the valid data types (Page 231)

Setting retentivity in an instance data block (Page 7037)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

CTUD: Count up and down (S7-1500)

Description

Use the "Count up and down" instruction to increment or decrement the counter value at the CV parameter. When the signal state at the CU parameter changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored in the CV parameter. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the counter value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in a program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV parameter. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state of the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has signal state "1", the signal state of the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0". You can also specify a constant for the PV parameter.

You can query the status of the down counter at the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

In the program code, you call the "Count up and down" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

The operating system resets the instances of the "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a startup OB:

- When used as up counter, the value at the R parameter must be set to "1".
- When used as down counter, the value at the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Instance data block of system data type IEC_<Counter> (shared DB / single instance)

The system data type of the instance data block is derived from the data type of the instruction:

Data type of the instruction	System data type of the instance data block (shared DB)
SINT / USINT	IEC_SCOUNTER / IEC_USCOUNTER
INT / UINT	IEC_COUNTER / IEC_UCOUNTER
DINT / UDINT	IEC_DCOUNTER / IEC_UDCOUNTER
LINT / ULINT	IEC_LCOUNTER / IEC_ULCOUNTER

When you set up the IEC Counter in a single instance, the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive.

For additional information on setting retentivity in an instance data block, refer to "See also".

Local tag (multi-instance)

The data type of the local tag is derived from the data type of the instruction:

Data type of the instruction	Data type of the local tag
SINT / USINT	CTUD_SINT / CTUD_USINT / IEC_SCOUNTER / IEC_USCOUNTER
INT / UINT	CTUD_INT / CTUD_UINT / IEC_COUNTER / IEC_UCOUNTER
DINT / UDINT	CTUD_DINT / CTUD_UDINT / IEC_DCOUNTER / IEC_UDCOUNTER
LINT / ULINT	CTUD_LINT / CTUD_ULINT / IEC_LCOUNTER / IEC_ULCOUNTER

When you set up the IEC Counter as multi-instance in a function block with "optimized block access", it is defined as retentive in the block interface.

Declaring IEC Counters

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens. You can then declare an IEC Counter as follows:

- Single instance: Declaration of an instance data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Multi-instance: Declaration as a local tag of the type CTUD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTUD_COUNTER)

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the QU output is set / value to which the CV output is set when LD = 1.
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

Example

The following example shows how the instruction works:

STL	Explanation
CALL CTUD, "CTUD_DB"	// The instruction is called. The "CTUD_DB" data block is assigned to the instruction. // Select the required data type from the "???" drop-down list.
CU := "Tag_StartCTU"	// When the signal state of the "Tag_StartCTU" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one. // The counter value is incremented on a rising signal edge of the CU parameter until it reaches the high limit of INT = 32767.

STL	Explanation
CD := "Tag_StartCTD"	// When the signal state of the "Tag_StartCTD" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterValue" operand is decremented by one.
	// The counter value of the CV parameter is decremented until the low limit of -32768 is reached.
R := "Tag_ResetCounter"	// When the signal state of the "Tag_ResetCounter" operand changes to "1", the "Tag_CounterValue" operand is reset to "0".
LD := "Tag_LoadPV"	// When the signal state of the "Tag_LoadPV" operand changes to "1", the "Tag_CounterValue" operand is set to the value of the "Tag_PresetValue" operand.
PV := "Tag_PresetValue"	// Specifies the value to which the counter is set when the signal state is "1" in the LD parameter.
QU := "Tag_CounterStatus"	// The operand is set as long as the current counter value is greater than or equal to the value of the PV parameter.
QD := "Tag_CounterStatus"	// The operand is set when the current counter value is less than or equal to zero.
CV := "Tag_CounterValue"	// Current counter value

See also

Overview of the valid data types (Page 231)

Setting retentivity in an instance data block (Page 7037)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

Comparator operations (S7-1500)

CompType: Compare tag structured data types (S7-1500)

Description

You can use the "Compare tag structured data types" instruction to determine if the first comparison value of a structured tag (IN1) is equal to or not equal to a second comparison value of another structured tag (IN2).

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1" at the OUT parameter. If the comparison condition is not fulfilled, the instruction returns RLO "0".

Comparison of floating-point numbers

When floating-point numbers are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check.

The special bit patterns of invalid floating-point numbers (NaN) that are the outcome of undefined results (e.g. root of -1) are not comparable. This means that if one of the operands has the value NaN, both the "CompType EQ" and "CompType NE" instructions return FALSE as the result.

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table shows examples of EQ string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'aa'	'aaa'	0

The following table shows examples of NE string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0
'aa'	'aaa'	1

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Comparison of timers, date and time

Bit patterns of invalid timers, date and times, e.g. DT#2015-13-33-25:62:99.999_999_999, cannot be compared. This means that if one of the operands has an invalid value, both the "CompType EQ" and "CompType NE" instructions return FALSE as the result.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

Note

Availability of comparison of structures

The option to compare structures is available for a CPU of the S7-1500 series as of firmware version >= 2.0.

You can compare the values of two structured operands when both tags are of the same structure data type. When structures are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check. An exception is comparisons in which one of the two operands is a VARIANT or an ANY. If the data type is not yet known when the program is created, you can use the VARIANT data type. In this case you can also compare the operand with a structured tag of any data type. You can also compare two tags of the data type VARIANT or ANY with each other.

The following data types are possible:

- PLC data type
- STRUCT (the structure of the data type STRUCT must be contained in a PLC data type (UDT) or the two structures to be compared are two elements of an ARRAY of STRUCT. Anonymous structures are not permitted.)
- Tag to which ANY is pointing
- Tag to which VARIANT is pointing

The following requirements must be met to compare two tags of the data type ARRAY:

- The elements must each have the same data type.
- The two ARRAYs must have the same dimension.
- All dimensions must have the same number of elements. The exact ARRAY limits do not have to match.

The tables below show examples of a comparison of structures for "Equal":

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type A <PLC data type>		Tag value	1
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type B <PLC data type>		Tag value	0
	BOOL	FALSE		BOOL	TRUE	
	INT	2		INT	3	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	VARIANT (supplied with tag of data type A)		Tag value	1
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

The tables below show examples of a comparison of structures for "Not equal":

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type A <PLC data type>		Tag value	0
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type B <PLC data type>		Tag value	1
	BOOL	FALSE		BOOL	TRUE	
	INT	2		INT	3	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	VARIANT (supplied with tag of data type A)		Tag value	0
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

Parameters

The following table shows the parameters of the instruction "Compare tag structured data types":

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Binary numbers, integers, floating-point numbers, character strings, timers, date and time, ARRAY of <data type> with fixed and variable ARRAY limits, STRUCT, VARIANT, ANY, PLC data type	I, Q, M, D, L, P	First comparison value
IN2	Input	Binary numbers, integers, floating-point numbers, character strings, timers, date and time, ARRAY of <data type> with fixed and variable ARRAY limits, STRUCT, VARIANT, ANY, PLC data type	I, Q, M, D, L, P	Second value to compare
OUT	Output	BOOL	I, Q, M, D, L	Result of the instruction
As detailed above, the data types ARRAY, STRUCT (in a PLC data type), VARIANT, ANY and PLC data type (UDT) are only available as of firmware version 2.0.				

Example

The following example shows how the instruction works:

```
STL                                Explanation
CALL CompType                      // The instruction is called.
                                   // Select the required function. Either "EQ" or
                                   "NE" from the drop-down list.
IN1 := "Tag_Operand1"              // First comparison value
IN2 := "Tag_Operand2"              // Second comparison value
OUT := "Tag_Result"                // The "Tag_Result" output returns the signal state
                                   "1" if the condition of the comparison instruction
                                   is fulfilled. i.e. the "Tag_Operand1" operand is
                                   equal to "Tag_Operand2".
```

See also

Overview of the valid data types (Page 231)

Basic information on VARIANT (Page 324)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

VARIANT (S7-1500)

EQ_Type: Compare data type for EQUAL with the data type of a tag (S7-1500)

Description

You can use the "Compare data type for EQUAL with the data type of a tag" instruction to query the data type of a tag to which the VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Equal".

The tag at the IN1 parameter must have the VARIANT data type. The tag at the IN2 parameter can be an elementary data type or a PLC data type.

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CompType". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CompType: Compare tag structured data types (Page 1311)

Parameters

The following table shows the parameters of the "Compare data type for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	First operand
IN2	Input	Binary numbers, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L, P	Second operand
RET_VAL	Output	BOOL	I, Q, M, D, L	Result of the instruction

Example

The following example shows how the instruction works:

```
STL                                  Explanation
CALL EQ_Type                         // The instruction is called.
IN1 := #Tag_Operand1                 // First operand to be compared
IN2 := "Tag_Operand2"                // Second operand to be compared
RET_VAL := "Tag_Result"              // The "Tag_Result" output returns the signal
                                     state "1" if the condition of the comparison in-
                                     struction is fulfilled. i.e. the #Tag_Operand1
                                     operand is not equal to "Tag_Operand2".
```

See also

Overview of the valid data types (Page 231)

Basic information on VARIANT (Page 324)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

NE_Type: Compare data type for UNEQUAL with the data type of a tag (S7-1500)

Description

You can use the "Compare data type for UNEQUAL with the data type of a tag" instruction to query the data type which a tag does not have to which a VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Not equal".

The tag at the IN1 parameter must have the VARIANT data type. The tag at the IN2 parameter can be an elementary data type or a PLC data type.

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CompType". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CompType: Compare tag structured data types (Page 1311)

Parameters

The following table shows the parameters of the "Compare data type for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	First operand
IN2	Input	Binary numbers, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L, P	Second operand
RET_VAL	Output	BOOL	I, Q, M, D, L	Result of the instruction

Result

The following example shows how the instruction works:

STL	Explanation
CALL NE_Type	// The instruction is called.
IN1 := #Tag_Operand1	// First operand to be compared
IN2 := "Tag_Operand2"	// Second operand to be compared
RET_VAL := "Tag_Result"	// The "Tag_Result" output returns the signal state "1" if the condition of the comparison instruction is fulfilled. i.e. the #Tag_Operand1 operand is not equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 231)

Basic information on VARIANT (Page 324)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

EQ_ElemType: Compare data type of an ARRAY element for EQUAL with the data type of a tag (S7-1500)

Description

You can use the "Compare data type of an ARRAY element for EQUAL with the data type of a tag" instruction to query the data type of a tag to which the VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Equal".

The tag at the IN1 parameter must have the VARIANT data type. The tag at the IN2 parameter can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CompType". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CompType: Compare tag structured data types (Page 1311)

Parameters

The following table shows the parameters of the "Compare data type of an ARRAY element for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	First operand
IN2	Input	Binary numbers, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L, P	Second operand
RET_VAL	Output	BOOL	I, Q, M, D, L	Result of the instruction

Result

The following example shows how the instruction works:

STL	Explanation
`CALL EQ_ElemType`	`// The instruction is called.`
`IN1 := #Tag_Operand1`	`// First operand to be compared`
`IN2 := "Tag_Operand2"`	`// Second operand to be compared`
`RET_VAL := "Tag_Result"`	`// The "Tag_Result" output returns the signal state "1" if the condition of the comparison instruction is fulfilled. i.e. the #Tag_Operand1 operand is not equal to "Tag_Operand2".`

See also

Overview of the valid data types (Page 231)

Basic information on VARIANT (Page 324)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

NE_ElemType: Compare data type of an ARRAY element for UNEQUAL with the data type of a tag (S7-1500)

Description

You can use the "Compare data type of an ARRAY element for UNEQUAL with the data type of a tag" instruction to query the data type which the tag does not have to which the VARIANT points. You are comparing the data type of the tag at IN1 parameter, which you declared in the block interface, with the data type of a tag at IN2 parameter for "Not equal".

The tag at the IN1 parameter must have the VARIANT data type. The tag at the IN2 parameter can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

Comparison of timers, date and time

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparison of structures

For the comparison of structures, you can use the instruction "CompType". Anonymous structures cannot generally be compared unless they are part of the same ARRAY.

For additional information on the comparison of structures, please refer to: CompType: Compare tag structured data types (Page 1311)

Parameters

The following table shows the parameters of the "Compare data type of an ARRAY element for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	First operand
IN2	Input	Binary numbers, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L, P	Second operand
RET_VAL	Output	BOOL	I, Q, M, D, L	Result of the instruction

Result

The following example shows how the instruction works:

STL	Explanation
`CALL NE_ElemType`	`// The instruction is called.`
`IN1 := #Tag_Operand1`	`// First operand to be compared`
`IN2 := "Tag_Operand2"`	`// Second operand to be compared`
`RET_VAL := "Tag_Result"`	`// The "Tag_Result" output returns the signal state "1" if the condition of the comparison instruction is fulfilled. i.e. the #Tag_Operand1 operand is not equal to "Tag_Operand2".`

See also

Overview of the valid data types (Page 231)

Basic information on VARIANT (Page 324)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

IS_NULL: Check for EQUALS NULL pointer (S7-1500)

Description

You can use the instruction "Check for EQUALS NULL pointer" to query whether the VARIANT or the reference points to a NULL pointer and therefore does not point to an object.

The tag at the parameter OPERAND must have the data type VARIANT or REF_TO >data type>.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "0" even if the ANY pointer is NULL.

Parameters

The following table shows the parameters of the "Check for EQUALS NULL pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
OPERAND	Input	VARIANT or REF_TO <data type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand that is compared for EQUALS NULL
RET_VAL	Output	BOOL	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL IS_NULL	// The instruction is called.
OPERAND := #Tag_Operand	// Operand to be compared
RET_VAL := "Tag_Result"	// The "Tag_Result" output returns the signal state "1" if the condition of the comparison instruction is fulfilled. i.e. the #Tag_Operand operand does not point to an object.

See also

Overview of the valid data types (Page 231)

Basic information on VARIANT (Page 324)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

NOT_NULL: Check for UNEQUALS NULL pointer (S7-1500)

Description

You can use the instruction "Check for UNEQUALS NULL pointer" to query whether the VARIANT or the reference does not point to a NULL pointer and therefore points to an object.

The tag at the parameter OPERAND must have the data type VARIANT or REF_TO >data type>.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "1" even if the ANY pointer is NULL.

Parameters

The following table shows the parameters of the "Check for UNEQUALS NULL pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
OPERAND	Input	VARIANT or REF_TO <data type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand that is compared for UNEQUALS NULL
RET_VAL	Output	BOOL	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                                  Explanation
CALL NOT_NULL                        // The instruction is called.
OPERAND := #Tag_Operand              // Operand to be compared
RET_VAL := "Tag_Result"              // The "Tag_Result" output returns the signal
                                     state "1" if the condition of the comparison in-
                                     struction is fulfilled. i.e. the #Tag_Operand
                                     operand does not point to an object.
```

See also

Overview of the valid data types (Page 231)

Basic information on VARIANT (Page 324)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

IS_ARRAY: Check for ARRAY (S7-1500)

Description

You can use the "Check for ARRAY" instruction to query whether the VARIANT points to a tag of the ARRAY data type.

The tag at the OPERAND parameter has the data type VARIANT. The result of the query is output at the RET_VAL parameter.

Parameters

The following table shows the parameters of the "Check for ARRAY" instruction:

Parameters	Declaration	Data type	Memory area	Description
OPERAND	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand that is queried for ARRAY
RET_VAL	Output	BOOL	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`CALL IS_ARRAY`	`// The instruction is called.`
`OPERAND := #Tag_Input`	`// Operand to be queried at ARRAY`
`RET_VAL := "Tag_Result"`	`// The "Tag_Result" output returns the signal state "1" if the condition of the comparison instruction is fulfilled. // In other words, if the VARIANT at the #Tag_Input operand points to an ARRAY.`

See also

Overview of the valid data types (Page 231)

Basic information on VARIANT (Page 324)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

EQ_TypeOfDB: Compare data type of an indirectly addressed DB for EQUAL with a data type (S7-1500)

Description

The "Compare data type of an indirectly addressed DB for EQUAL with a data type" instruction is used to query which data type of the data block has that the tag of the DB_ANY data type addresses. You compare the data type of the DB addressed by the tag at the IN1 parameter either with the data type of another tag or directly with a data type at the IN2 parameter for "Equal".

The tag at the IN1 parameter must have the DB_ANY data type. The tag at the IN2 parameter, for example, can be a PLC data type, a system data type, an axis or an FB.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DB_ANY	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	First operand
IN2	Input	TYPE_ID	I, Q, M, D, L, P	Second operand
RET_VAL	Output	BOOL	I, Q, M, D, L	Result of comparison

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

STL	Description
CALL EQ_TypeOfDB	// Call the instruction
IN1 := #InputDBAny	// Operand to query for DB_ANY
IN2 := TO_SpeedAxis	// The condition of the comparison instruction is fulfilled if the data type of the #InputDBAny operand addressed DB is equal with the data type TO_SpeedAxis.
RET_VAL := "TagOut"	// Result of comparison

The "TagOut" output is not set when the following conditions are fulfilled:

- The number of the data block is "0".
- The data block does not exist.
- The data block is an ARRAY DB.

See also

Overview of the valid data types (Page 231)

Using the DB_ANY data type (Page 204)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

NE_TypeOfDB: Compare data type of an indirectly addressed DB for UNEQUAL with a data type (S7-1500)

Description

The "Compare data type of an indirectly addressed DB for UNEQUAL with a data type" instruction is used to query which data type the data block does not have, that the tag of the DB_ANY data type addresses. You compare the data type of the DB addressed by the tag at the IN1 parameter either with the data type of another tag or directly with the data type at the IN2 parameter for "Not equal".

The tag at the IN1 parameter must have the DB_ANY data type. The tag at the IN2 parameter, for example, can be a PLC data type, a system data type, an axis or an FB.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DB_ANY	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	First operand
IN2	Input	TYPE_ID	I, Q, M, D, L, P	Second operand
RET_VAL	Output	BOOL	I, Q, M, D, L	Result of comparison

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

```
STL                                    Description
CALL NE_TypeOfDB                       // Call the instruction
IN1 := #InputDBAny                     // Operand to query for DB_ANY
```

```
STL                                        Description
IN2 := TO_SpeedAxis                        // The condition of the comparison in-
                                           struction is fulfilled if the data type
                                           of the #InputDBAny operand addressed DB
                                           is unequal with the data type TO_SpeedAx-
                                           is.
RET_VAL := "TagOut"                        // Result of comparison
```

The "TagOut" output is not set when the following conditions are fulfilled:

- The number of the data block is "0".
- The data block does not exist.
- The data block is an ARRAY DB.

See also

Overview of the valid data types (Page 231)

Using the DB_ANY data type (Page 204)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

Math functions (S7-1500)

MIN: Get minimum (S7-1500)

Description

The "Get minimum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the lowest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT parameter is invalid if one of the following conditions is met:

- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second input value
IN3	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Third input value

Parameters	Declaration	Data type	Memory area	Description
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result
If the IEC check is not activated, you can also use tags with the data types TIME, LTIME, TOD, LTOD, DATE and LDT by selecting a bit string or integer with the same length as the data type of the instruction (e.g. instead of TIME => DINT, UDINT or DWORD = 32 bits).				

You can select the data type for the parameters INn and OUT from the "???" drop-down list.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL MIN                     // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
IN1 := "TagIn_Value1"        // First input value is compared.
IN2 := "TagIn_Value2"        // Second input value to be compared.
IN3 := "TagIn_Value3"        // Third input value to be compared.
OUT := "Tag_Minimum"         The "Tag_Minimum" operand is written with the value of
                             the "TagIn_Value1" operand because this has the lowest
                             value.
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Minimum	12222

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

MAX: Get maximum (S7-1500)

Description

The "Get maximum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the highest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is fulfilled:

- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second input value
IN3	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Third input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result
If the IEC check is not activated, you can also use tags with the data types TIME, LTIME, TOD, LTOD, DATE and LDT by selecting a bit string or integer with the same length as the data type of the instruction (e.g. instead of TIME => DINT, UDINT or DWORD = 32 bits).				

You can select the data type for the parameters INn and OUT from the "???" drop-down list.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL MAX                     // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
IN1 := "TagIn_Value1"        // First input value is compared.
IN2 := "TagIn_Value2"        // Second input value to be compared.
IN3 := "TagIn_Value3"        // Third input value to be compared.
OUT := "Tag_Maximum"         // The "Tag_Maximum" operand is written with the value
                             of the "TagIn_Value2" operand because this has the
                             highest value.
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Maximum	14444

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

LIMIT: Set limit value (S7-1500)

Description

You can use the "Set limit value" instruction to limit the value at the IN input to the values at the inputs MN and MX. If the value at the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the IN input value is below the low limit MN, the OUT output is set to the value of the MN input. If the high limit MX is exceeded, the OUT output is set to the value of the MX input.

The instruction is only executed if all the tags at the inputs are of the same data type.

If the value at the MN input is greater than at the MX input, the result is the value specified at the IN parameter.

The value of the OUT output is invalid if one of the following conditions is fulfilled:

- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN parameter is greater than the value of the MX parameter.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameters	Declaration	Data type	Memory area	Description
MN	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Low limit
IN	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Input value
MX	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	High limit

Parameters	Declaration	Data type	Memory area	Description
OUT	Output	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

You can select the data type for the parameters MN, IN, MX and OUT from the "???" drop-down list.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL LIMIT                   // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
MN := "Tag_LowLimit"         // Low limit
IN := "Tag_InputValue"       // Input value
MX := "Tag_HighLimit"        // High limit
OUT := "Tag_Result"          // The "Tag_Result" operand is written with the value
                             of the "Tag_LowLimit" operand because the value of the
                             "Tag_InputValue" operand is outside the defined limit
                             range.
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_LowLimit	12000
IN	Tag_InputValue	8000
MX	Tag_HighLimit	16000
OUT	Tag_Result	12000

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

Move operations (S7-1500)

MOVE: Move value (S7-1500)

Description

You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT output. The operands at input IN and at output OUT must have the identical data type.

The following table shows the possible transfers:

Source (IN)	Destination (OUT)
BYTE	BYTE
WORD	WORD
DWORD	DWORD
LWORD	LWORD
SINT	SINT
USINT	USINT
INT	INT
UINT	UINT
DINT	DINT
UDINT	UDINT
LINT	LINT
ULINT	ULINT
REAL	REAL
LREAL	LREAL
S5TIME	S5TIME
TIME	TIME
LTIME	LTIME
DATE	DATE
DT	DT
LDT	LDT
TOD	TOD
LTOD	LTOD
DTL	DTL
CHAR	CHAR
WCHAR	WCHAR
Character of a string[1)]	Character of a string
ARRAY	ARRAY
STRUCT	STRUCT
PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER
IEC_LTIMER	IEC_LTIMER
IEC_SCOUNTER	IEC_SCOUNTER

Source (IN)	Destination (OUT)
IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER
IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNTER	IEC_DCOUNTER
IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNTER	IEC_LCOUNTER
IEC_ULCOUNTER	IEC_ULCOUNTER
DB_ANY	DB_ANY
REF()	There is no data type conversion. The transferred tag must correspond exactly to the target data type.

[1)] You can also use the "Move value" instruction to transfer individual characters of a string to operands of the CHAR or WCHAR data type. The number of the character to be transferred is specified in square brackets next to the operand name. "MyString[2]", for example, transfers the second character of the "MyString" string. It is also possible to transfer from operands of the data type CHAR to the individual characters of a string. You can also replace a specific character of a string with the character of another string.

You can use the "Move block" (MOVE_BLK), "Move block uninterruptible" (UMOVE_BLK) and "Move block" (MOVE_BLK_VARIANT) instructions to move operands of the ARRAY data type. You can move operands of the STRING data type with the instruction "Move character string" (S_MOVE).

Parameters

The following table shows the parameters of the "Move value" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings, integers, floating-point numbers, timers, date and time, strings, ARRAY, IEC data types, PLC data type (UDT), DB_ANY	I, Q, M, D, L or constant	Element used to overwrite the destination address
OUT	Output	Bit strings, integers, floating-point numbers, timers, date and time, strings, ARRAY, IEC data types, PLC data type (UDT), DB_ANY	I, Q, M, D, L	Destination address

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL MOVE                    // The instruction is called.
IN := "TagIn_LREAL"          // The contents of the operand "TagIn_LREAL" are moved.
OUT := "TagOut_LREAL"        // And transferred to the operand "TagOut_LREAL".
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_LREAL	1.23456789098e55
OUT	TagOut_LREAL	1.23456789098e55

Note

You can find more information on the MOVE instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

Deserialize: Deserialize (S7-1500)

Description

You can use the "Deserialize" instruction to convert back the sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type> and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type> is located must have the ARRAY of BYTE or ARRAY of CHAR data type and be declared with standard access in version 1.0. Optimized memory areas also permitted as of version 2.0. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion. If the memory area is filled using the instruction "Serialize" and it is possible that fill bytes were inserted, their conversion is not taken into account.

It is recommended to define the low limit of the ARRAY with "0", because then the index within ARRAY corresponds to the value of the POS parameter, e.g. ARRAY[0] = POS 0. The description and example below have been formed based on this.

The instruction enables you to convert multiple sequential representations of converted data back to their original state step-by-step.

If you only want to convert back a single sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type>, you can also directly use the instruction "TRCV: Receive data via communication connection".

Size of the memory area

The alignment rules mean that simple structures in the optimized memory area do not contain filling bytes. This results in a structure in the optimized memory area being smaller than in the standard memory area. ARRAYs of structures and structures in which other structures are located contain filling bytes. There is no general rule as to the memory area in which a composed structure requires more space.

Applies to CPUs of the S7-1500 series:

For a block with the block property "Optimized block access", the BOOL has a length of 1 byte. A structure that consists largely of the data type BOOL can therefore be larger in the optimized memory area than in the standard memory area. Composed structures with a low proportion of BOOL data types are smaller in the optimized memory area than in the standard memory area.

Optimized memory area

To deserialize larger structures, you can declare the memory area for sequential representation with optimized access for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. The content of the sequential representation remains unchanged, as for a standard memory area. Only symbolic access to the bytes in the ARRAY is possible.

Parameters

The following table shows the parameters of the "Deserialize" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC_ARRAY	Input	VARIANT ARRAY[*] of BYTE [1)]	D, block interface of an FB (the sections Input, Output, Static and Temp are possible.)	Pointer to an ARRAY of BYTE or ARRAY of CHAR in which the data stream is saved.
DEST_VARIABLE	InOut	VARIANT	D, block interface of an FB	Pointer to a tag of the STRUCT, ARRAY or PLC data type (UDT) to be deserialized.
POS	InOut	DINT	I, Q, M, D, L	The operand at the POS parameter stores the index of the first byte based on the number of bytes occupied by the converted customer data. The POS parameter is calculated zero-based.

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	Error information
[1] Possible for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The tag at the SRC_ARRAY parameter is not in a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains a ZERO pointer.
8151	No valid reference at the SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8154	Invalid data type at the SRC_ARRAY parameter
8250	A NULL pointer was transferred at parameter DEST_ARRAY.
8251	No valid reference at the DEST_VARIABLE parameter
8382	The value at the POS parameter is outside the limits of the array.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

For a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0, the following error code has a different meaning:

Error code* (W#16#...)	Explanation
8136	Access to the memory area at the SRC_ARRAY parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface of an FB or FC.
BufferPos	DINT	In the "Temp" section of the block interface of an FB or FC.
Error	INT	
Label	STRING[4]	
cmp	BOOL	

The following table shows the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

The following example shows how the instruction works:

```
STL                                     Explanation
L 0                                     // The value "0" is loaded to accumulator
                                        1.
T #BufferPos                            // The value "0" is transferred to the
                                        #BufferPos operand.

CALL Deserialize                        // The instruction is called.
SRC_ARRAY := "Buffer".Field             // The customer data from the "Buffer"
                                        tag are deserialized.
RET_VAL := #Error                       // Error information
DEST_VARIABLE := "Target".Client        // The deserialized customer data are
                                        written to the "Target" tag.
POS := #BufferPos                       // The #BufferPos operand stores the in-
                                        dex of the first byte based on the number
                                        of bytes occupied by the converted custom-
                                        er data.

AN BR                                   // If an error occurs during the execu-
JC end                                  tion of the instruction, the processing
                                        of the program jumps to the end.

CALL Deserialize                        // The instruction is called.
SRC_ARRAY := "Buffer".Field             // The separator sheet from the "Buffer"
                                        tag is deserialized. The separator sheet
                                        was inserted in the form of 'arti' or
                                        'Bill' characters to allow better differ-
                                        entiation of the customer data.
RET_VAL := #Error                       // Error information
```

STL	Explanation
DEST_VARIABLE := #Label	// The deserialized characters of the separator sheet are written to the "#Label" operand.
POS := #BufferPos	// The number of bytes used by the converted characters is stored in the #BufferPos operand.
AN BR JC end	// If an error occurs during the execution of the instruction, the processing of the program jumps to the end.
CALL S_COMP	// The instruction is called. // Select the required data type from the "???" drop-down list.
IN1 := 'arti' IN2 := #Label	// Comparison of characters 'arti' with characters that were stored in the #Label operand.
OUT := #cmp	// Result of the instruction
A #cmp JCN noarticle	// If the comparison returns the signal state "0", the characters in the #Label operand are not 'arti' characters and the processing of the program jumps to the "noarticle" jump label.
CALL Deserialize	// If the result of the comparison was "1", the instruction is called.
SRC_ARRAY := "Buffer".Field	// The article data are deserialized.
RET_VAL := #Error	// Error information
DEST_VARIABLE := "Target".Article[#DeliverPos]	// The deserialized article data are written to the "Target" tag.
POS := #BufferPos	// The number of bytes used by the converted article data is stored in the "#BufferPos" operand.
AN BR JC end BE	// If no error occurred during processing, you have reached the block end.
noarticle: NOP 0	// If the comparison for 'arti' = "0", processing of the program continues here.
CALL S_COMP	// The instruction is called. // Select the required data type from the "???" drop-down list.

STL	Explanation
IN1 := 'Bill' IN2 := #Label	// Comparison of characters 'Bill' with characters that were stored in the #Label operand.
OUT := #cmp	// Result of the instruction
A #cmp JCN end	// If the comparison returns the signal state "1", the characters in the #Label operand are 'Bill' characters and processing of the program continues.
CALL Deserialize	// If the result of the comparison was "1", the instruction is called.
SRC_ARRAY := "Buffer".Field	// The bill data are deserialized.
RET_VAL := #Error	// Error information
DEST_VARIABLE := "Target".Bill[#Deliver-Pos]	// The deserialized customer data are written to the "Target" tag.
POS := #BufferPos	// The number of bytes used by the converted article data is stored in the "#BufferPos" operand.
AN BR JC end BE	// If no error occurred during processing, you have reached the block end.
end: CLR	// A jump to jump label "end" takes place in the case of an error.
SAVE	// The RLO is saved in the BR bit.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

Basic information on VARIANT (Page 324)

Basics of PLC data types (UDT) (Page 273)

Structure of an ARRAY tag (Page 287)

Structure of a STRUCT tag (Page 280)

Structure of a STRING tag (Page 266)

STL Basics (Page 7245)

Serialize: Serialize (S7-1500)

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT), STRUCT or ARRAY of <data type> to a sequential representation without losing parts of their structure.

You use the instruction to temporarily save multiple structured data items from your program in a buffer, which should preferably be in a global data block, and send them to another CPU. The memory area in which the converted data is stored must have the ARRAY of BYTE or ARRAY of CHAR data type and be declared with standard access in version 1.0. Optimized data is also permitted as of version 2.0. Fill data of the source data area is undefined in the target array. These can be fill bytes or fill bits of a data area (e.g. ARRAY, STRUCT or PLC data type (UDT)) as well as the characters of a string currently not in use.

The capacity of the standard memory area is 64 KB. Structures that are larger than 64 KB in accordance with the rules for standard memory areas cannot be serialized if the operand at the DEST_ARRAY parameter is located in a standard memory area.

It is recommended to define the low limit of the ARRAY with "0", because then the index within ARRAY corresponds to the value of the POS parameter, e.g. ARRAY[0] = POS 0. The description and example below have been formed based on this.

The operand at the POS parameter contains information about the number of bytes used by the converted data.

If you want to send a single PLC data type (UDT), STRUCT or ARRAY of <data type>, you can directly call the instruction "TSEND: Send data via communication connection".

Size of the memory area

The alignment rules mean that simple structures in the optimized memory area do not contain filling bytes. This results in a structure in the optimized memory area being smaller than in the standard memory area. ARRAYs of structures and structures in which other structures are located contain filling bytes. There is no general rule as to the memory area in which a composed structure requires more space.

Applies to CPUs of the S7-1500 series:

For a block with the block property "Optimized block access", the length of the data type BOOL depends on the data type that follows. This means if a BYTE follows, for example, the data type BOOL has a length of 1 byte. When a WORD follows, for example, the data type BOOL has a length of 2 bytes. A structure that consists largely of the data type BOOL can therefore be larger in the optimized memory area than in the standard memory area. Composed structures with a low proportion of BOOL data types are smaller in the optimized memory area than in the standard memory area.

We therefore recommend that the source data area for the serialization starts with the large data types and ends with Boolean elements. This greatly reduces the filling with filling bits.

Optimized memory area

To serialize larger structures, you can declare the memory area with optimized access for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. The sequential representation remains unchanged, as for a standard memory area.

Parameters

The following table shows the parameters of the instruction "Serialize":

Parameters	Declaration	Data type	Memory area	Description
SRC_VARIABLE	Input	VARIANT	D, block interface of an FB	Pointer to a tag of the STRUCT, ARRAY or PLC data type (UDT) to be serialized.
DEST_ARRAY	InOut	VARIANT	D, block interface of an FB (the sections Input, Output, Static and Temp are possible.)	Pointer to an ARRAY of BYTE or ARRAY of CHAR in which the generated data stream is saved.
POS	InOut	DINT	I, Q, M, D, L	The operand at the POS parameter stores the index of the first byte based on the total number of bytes that the converted customer data has occupied. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains a ZERO pointer.
8151	No valid reference at the SRC_VARIABLE parameter
8236	The tag at the SRC_ARRAY parameter is not in a block with standard access.
8250	A NULL pointer was transferred at parameter DEST_ARRAY.
8251	No valid reference at the DEST_ARRAY parameter
8253	The tag at the parameter DEST_ARRAY does not provide enough memory space for the content of the tag at the parameter SRC_VARIABLE. Due to the input value of the tag at parameter POS the available memory space is reduced. The input value of the parameter POS decides where to begin within the tag at the parameter DEST_ARRAY.
8254	Invalid data type at the DEST_ARRAY parameter

Error code* (W#16#...)	Explanation
8382	The value at the POS parameter is outside the limits of the array.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

For a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0, the following error code has a different meaning:

Error code* (W#16#...)	Explanation
8236	Access to the memory area at the DEST_ARRAY parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
L 0	// The value "0" is loaded to accumulator 1.
T #BufferPos	// The value "0" is transferred to the #BufferPos operand.
CALL Serialize	// The instruction is called.
SRC_VARIABLE := "Source".Client	// The customer data from the "Source" tag are serialized.
RET_VAL := #Error	// Error information
DEST_ARRAY := "Buffer".Field	// The sequential representation is written to the "Buffer" tag.
POS := #BufferPos	// The index of the next unwritten byte of the "Buffer".Field is stored in the #BufferPos operand.
CALL S_MOVE	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
IN := 'arti' OUT := #Label	// The 'arti' characters are copied to the #Label operand.
CALL Serialize	// The instruction is called.
SRC_VARIABLE := #Label	/ / The characters are serialized.
RET_VAL := #Error	// Error information
DEST_ARRAY := "Buffer".Field	// The sequential representation is written to the "Buffer".Field tag after the customer data.
POS := #BufferPos	// The value of the #BufferPos operand is incremented accordingly.

STL	Explanation
CALL Serialize	// The instruction is called.
SRC_VARIABLE := "Source".Article[#DeliverPos]	// The data of a specific article from the "Source" tag are serialized.
RET_VAL := #Error	// Error information
DEST_ARRAY := "Buffer".Field	// The sequential representation is written to the "Buffer" tag after the 'arti' characters.
POS := #BufferPos	// Number of bytes

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table shows the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

Basic information on VARIANT (Page 324)

Basics of PLC data types (UDT) (Page 273)

Structure of an ARRAY tag (Page 287)
Structure of a STRUCT tag (Page 280)
Structure of a STRING tag (Page 266)
STL Basics (Page 7245)

MOVE_BLK: Move block (S7-1500)

Description

You can use the "Move block" instruction to move the content of a memory area (source range) to another memory area (target range). The number of elements to be moved to the target range is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at the IN parameter.

The instruction can only be executed if the source range and the target range have the same data type.

The value at the OUT output is invalid if more data is moved than is made available at the IN parameter or OUT parameter.

Note

Use of ARRAYs

The instruction copies the contents from the defined element n elements (n = depending on the value at the parameter COUNT) from the source range to the target range, starting at the specified index.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be copied from the source range to the target range
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	The first element of the target range to which the contents of the source range are being copied
[1] The specified data types can only be used as elements of an Array structure.				

You can select the data type of the instruction from the "???" drop-down lists.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL MOVE_BLK	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
IN := #a_array[2]	// First element to be moved.
COUNT := "Tag_Count"	// Number of elements to be moved from the source range to the target range.
OUT := #b_array[1]	// First element to which is moved.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The data type of the "a_array" operand is ARRAY[0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the "b_array" operand is ARRAY[0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element.

Note

You can find more information on the MOVE_BLK instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

MOVE_BLK_VARIANT: Move block (S7-1500)

Description

You can use the "Move block" instruction to move the content of a memory area (source range) to another memory area (target range). You can copy a complete ARRAY or elements of an ARRAY to another ARRAY of the same data type. The size (number of elements) of source and destination ARRAY may be different. You can copy multiple or single elements within an ARRAY.

The number of elements that are to be copied must not exceed the selected source range or target range.

If you use the instruction at the time the block is being created, the ARRAY does not yet have to be known, as the source and the target are transferred per VARIANT.

Counting at the parameters SRC_INDEX and DEST_INDEX always begins with the low limit "0", regardless of the later declaration of the ARRAY.

The instruction is not executed if more data is copied than is made available.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC	Input [2)]	VARIANT (which points to an ARRAY or an individual ARRAY element), ARRAY of <Data_type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L or constant	Number of elements which are copied Assign the value "1" to the parameter COUNT, if no ARRAY is specified at parameter SRC or at parameter DEST.

Parameters	Declaration	Data type	Memory area	Description
SRC_INDEX	Input	DINT	I, Q, M, D, L or constant	Defines the first element to be copied: • The SRC_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter SRC, the integer at parameter SRC_INDEX specifies the first element within the source area from which it is to be copied. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter SRC or only one single element of an ARRAY is specified, then assign the value "0" at parameter SRC_INDEX.
DEST_INDEX	Input	DINT	I, Q, M, D, L or constant	Defines the start of the destination memory area: • The DEST_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter DEST, the integer at parameter DEST_INDEX specifies the first element within the target range that is to be copied into. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter DEST, then assign the value "0" at parameter DEST_INDEX.
DEST	Output [1)]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Destination area into which the contents of the source block are copied.

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
1) The DEST parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface. 2) With the SRC parameter, the data types BOOL and ARRAY of BOOL are not permitted.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data types do not correspond. Instead of an ARRAY of Struct, use an ARRAY of PLC data type (UDT).
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at the SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value
8382	The value at the SRC_INDEX parameter is less than zero.
8383	The value at the SRC_INDEX is outside the high limit of the ARRAY.
8482	The value at the DEST_INDEX parameter is less than zero.
8483	The value at the DEST_INDEX is outside the high limit of the ARRAY.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at the DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                                  Explanation
CALL MOVE_BLK_VARIANT                // The instruction is called.
SRC := #SrcField                     // First element to moved
COUNT := "Tag_Count"                 // Number of elements to be moved from the
                                     source range to the target range.
```

```
STL                                    Explanation
SRC_INDEX := "Tag_Src_Index"           // First element at the SRC parameter to be
                                       moved
DEST_INDEX := "Tag_Dest_Index"         // First element at the DEST parameter to be
                                       moved
DEST := #DestField                     // The first element of the source area to be
                                       moved.
RET_VAL := "Tag_Result"                // Error information
```

The following table shows how the instruction works using specific operand values:

Parameters	Declaration in the block interface	Operand	Value
SRC	Input	#SrcField	The local operand #SrcField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[0..10] of "MOVE_UDT")
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	The local operand #DestField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[10..20] of "MOVE_UDT")

Two elements are moved from the source range, starting from the fourth element of the ARRAY [0..10] of MOVE_UDT, to the target range. The copies are pasted in the ARRAY [10..20] of MOVE_UDT starting from the fourth element.

Note

You can find more information on the MOVE_BLK_VARIANT instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

Basic information on VARIANT (Page 324)

STL Basics (Page 7245)

UMOVE_BLK: Move block uninterruptible (S7-1500)

Description

You use the "Move block uninterruptible" instruction to move the content of a memory area (source range) to another memory area (target range). The instruction cannot be interrupted. The number of elements to be moved to the target range is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source range and the target range have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

The value at the OUT output is invalid if more data is moved than is made available at the IN parameter or OUT parameter.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Note

Use of ARRAYs

The instruction copies the contents from the defined element n elements (n = depending on the value at the parameter COUNT) from the source range to the target range, starting at the specified index.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be copied from the source range to the target range
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	The first element of the target range to which the contents of the source range are being copied
[1] The specified data types can only be used as elements of an Array structure.				

You can select the data type of the instruction from the "???" drop-down lists.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`CALL UMOVE_BLK`	`// The instruction is called.`
	`// Select the required data type from the "???" drop-down list.`
`IN := #a_array[2]`	`// First element to be moved.`
`COUNT := "Tag_Count"`	`// Number of elements to be moved from the source range to the target range.`
`OUT := #b_array[1]`	`// First element to which is moved.`

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The data type of the "a_array" operand is ARRAY[0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the "b_array" operand is ARRAY[0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The move operation cannot be interrupted by other operating system activities.

Note

You can find more information on the UMOVE_BLK instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

FILL_BLK: Fill block (S7-1500)

Description

You can use the "Fill block" instruction to fill a memory area (target range) with the value of the IN input. The target range is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is moved to the target range as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source range and the target range have the same data type.

The maximum number of elements changed is the number of elements in the ARRAY or structure. If you copy more data than there are elements at the OUT output, you will get an unintended result.

Note

Use of ARRAYs

The instruction reads the content from the selected element from the source range and copies this content n-times (n = depending on the value at the parameter COUNT) to the destination, starting at the specified index.

Filling structures

As well as the elements of an ARRAY, you can also fill multiple elements of a structure (STRUCT, PLC data type) with the same value. The structure whose elements you want to fill must only contain individual elements of the same elementary data type. The structure itself can, however, be embedded in another structure.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L, P or constant	Element used to fill the target range
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of repeated move operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	Address in target range from which filling starts

You can select the data type of the instruction from the "???" drop-down lists.

For additional information on valid data types, refer to "See also".

Example with an ARRAY

The following example shows how the instruction works when you want to fill an ARRAY:

STL	Explanation
CALL FILL_BLK	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
IN := #FillValue	// Value to be copied
COUNT := "Tag_Count"	// Number of repeated copy operations
OUT := #TargetArea[1]	// The target range is filled from the first element of the ARRAY #TargetArea[1..5] of INT

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	FillValue	The data type of the operand is INT.
COUNT	Tag_Count	3
OUT	TargetArea	The data type of the operand TargetArea is ARRAY[1..5] of INT. It consists of five elements of data type INT.

The instruction copies the value of the operand #FillValue to the #TargetArea output tag three times, starting with the first element.

Examples with a structure

The following examples show how the instruction works when you want to fill a structure:

Create a global data block with the following elements:

Data_block_1					Data type
MyStruct1					STRUCT
	Member_1				INT
	Member_2				INT
	Member_3				INT
	Member_4				INT
MyStruct2					STRUCT
	SubArray				ARRAY[1..2] of STRUCT
		SubArray[1]			STRUCT
			NestedStruct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT
		SubArray[2]			STRUCT
			Nested-Struct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT

Generate the following program code to address the MyStruct1 tag:

STL	Explanation
CALL FILL_BLK	// The instruction is called.
	// Select the required data type from the "???" drop-down list.

STL	Explanation
IN := 10	// Value to be copied
COUNT := 2	// Number of repeated copy operations
OUT := "Data_block_1".MyStruct1.Member_2	// The structure is filled from Member_2. This means that Member_1 and Member_4 are not changed.

Generate the following program code to address the MyStruct2 tag:

STL	Explanation
CALL FILL_BLK	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
IN := 10	// Value to be copied
COUNT := 2	// Number of repeated copy operations
OUT := "Data_block_1".MyStruct2.SubArray[1].Member_2	// The structure is filled from Member_2. This means that Member_1 and Member_4 are not changed.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

UFILL_BLK: Fill block uninterruptible (S7-1500)

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (target range) with the value of the IN input. The instruction cannot be interrupted. The target range is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is moved to the target range as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source range and the target range have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The maximum number of elements changed is the number of elements in the ARRAY or structure. If you copy more data than there are elements at the OUT output, you will get an unintended result.

Note

Use of ARRAYs

The instruction reads the content from the selected element from the source range and copies this content n-times (n = depending on the value at the parameter COUNT) to the destination, starting at the specified index.

You use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Filling structures

As well as the elements of an ARRAY, you can also fill multiple elements of a structure (STRUCT, PLC data type) with the same value. The structure whose elements you want to fill must only contain individual elements of the same elementary data type. The structure itself can, however, be embedded in another structure.

Parameters

The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L, P or constant	Element used to fill the target range
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of repeated move operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	Address in target range from which filling starts

You can select the data type of the instruction from the "???" drop-down lists.

For additional information on valid data types, refer to "See also".

Example with an ARRAY

The following example shows how the instruction works when you want to fill an ARRAY:

```
STL                          Explanation
CALL UFILL_BLK               // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
   IN := #FillValue          // Value to be copied
   COUNT := "Tag_Count"      // Number of repeated copy operations
   OUT := #TargetArea[1]     // The target range is filled from the first element
                             of the ARRAY #TargetArea[1..5] of INT
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	FillValue	The data type of the operand is INT.
COUNT	Tag_Count	3
OUT	TargetArea	The data type of the operand TargetArea is ARRAY[1..5] of INT. It consists of five elements of data type INT.

The instruction copies the value of the operand #FillValue to the #TargetArea output tag three times, starting with the first element. The move operation cannot be interrupted by other operating system activities.

Examples with a structure

The following examples show how the instruction works when you want to fill a structure:

Create a global data block with the following elements:

Data_block_1		Data type
MyStruct1		STRUCT
	Member_1	INT
	Member_2	INT
	Member_3	INT
	Member_4	INT
MyStruct2		STRUCT

Data_block_1					Data type
	SubArray				ARRAY[1..2] of STRUCT
		SubArray[1]			STRUCT
			NestedStruct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT
		SubArray[2]			STRUCT
			Nested-Struct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT

Generate the following program code to address the MyStruct1 tag:

STL	Explanation
CALL UFILL_BLK	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
IN := 10	// Value to be copied
COUNT := 2	// Number of repeated copy operations
OUT := "Data_block_1".My-Struct1.Member_2	// The structure is filled from Member_2. This means that Member_1 and Member_4 are not changed.

Generate the following program code to address the MyStruct2 tag:

STL	Explanation
CALL UFILL_BLK	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
IN := 10	// Value to be copied
COUNT := 2	// Number of repeated copy operations
OUT := "Data_block_1".My-Struct2.	// The structure is filled from Member_2. This means that Member_1 and Member_4 are not changed.
SubArray[1].Member_2	

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

SCATTER: Parse the bit sequence into individual bits (S7-1500)

Description

The "Parse the bit sequence into individual bits" instruction parses a tag of the BYTE, WORD, DWORD or LWORD data type into individual bits and saves them in an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements.

Note

Multi-dimensional ARRAY of BOOL

With the "Parse the bit sequence into individual bits" instruction, the use of a multidimensional ARRAY of BOOL is not permitted.

Note

Length of the ARRAY, STRUCT or PLC data type

The ARRAY, the anonymous STRUCT or the PLC data type must have exactly the number of elements that is specified by the bit sequence.

This means for the data type BYTE, for example, the ARRAY, STRUCT or the PLC data type must have exactly 8 elements (WORD = 16, DWORD = 32 and LWORD = 64).

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1500 series as of firmware version 2.1.

This way you can, for example, parse a status word, and read and change the status of the individual bits using the index. Using GATHER, you can merge the bits once again into a bit sequence.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BYTE, WORD, DWORD, LWORD	I, Q, M, D, L	Bit sequence that is parsed. The values must not be located in the I/O area or in the DB of a technology object.
OUT	Output	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	I, Q, M, D, L	ARRAY, STRUCT or PLC data type in which the individual bits are stored

You can find additional information on valid data types under "See also".

Example with an ARRAY

Create the following tags in the block interface:

Tag	Section	Data type
SourceWord	Input	WORD
DestinationArray	Output	ARRAY[0..15] of BOOL

The following example shows how the instruction works:

STL

```
CALL SCATTER                          // Call the instruction
                                      // Select the required data type from the
                                      "???" drop-down list.
    IN := #SourceWord                 The operand #SourceWord of the data type
    OUT := #DestinationArray          WORD is parsed into its individual bits
                                      (16) and assign the individual elements
                                      of the #DestinationArray operand.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceWord	WORD (16 bits)
OUT	DestinationArray	The operand "DestinationArray" is of the data type ARRAY[0..15] of BOOL. It consists of 16 elements and is therefore just as large as the WORD that is to be parsed.

Example with a PLC data type (UDT)

Create the following PLC data type "myBits":

myBits

	Name	Data type
1	GeneralError	Bool
2	DeviceError	Bool
3	CommError	Bool
4	myError1	Bool
5	myError2	Bool
6	myError3	Bool
7	myError4	Bool
8	myError5	Bool
9	myError6	Bool
10	myError7	Bool
11	myError8	Bool
12	myError9	Bool
13	myError10	Bool
14	myError11	Bool
15	myError12	Bool
16	myError13	Bool

Create the following tags in the block interface:

Tag	Section	Data type
SourceWord	Input	WORD
DestinationUDT	Output	"myBits"

The following example shows how the instruction works:

STL

```
CALL SCATTER                          // Call the instruction
                                      // Select the required data type from the
                                      "???" drop-down list.
    IN := #SourceWord                 // The #SourceWord operand of the data
    OUT := #DestinationUDT            type WORD is parsed into its individual
                                      bits (16) and assigned to the individual
                                      elements of the #DestinationArray operand.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceWord	WORD (16 bits)
OUT	DestinationUDT	The "DestinationUDT" operand has the PLC data type (UDT). It consists of 16 elements and is therefore just as large as the WORD that is to be parsed.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

SCATTER_BLK: Parse elements of an ARRAY of bit sequence into individual bits (S7-1500)

Description

The "Parse elements of an ARRAY of bit sequence into individual bits" instruction parses one or more elements of an ARRAY of BYTE, WORD, DWORD or LWORD into individual bits and saves them in an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements. At the COUNT_IN parameter you specify how many elements of the source ARRAY are going to be parsed. The source ARRAY at the IN parameter may have more elements than specified at the COUNT_IN parameter. The ARRAY of BOOL, the anonymous STRUCT or the PLC data type must have sufficient elements to save the bits of the parsed bit sequences. However, the destination memory area may also be larger.

Note

Multi-dimensional ARRAY of BOOL

If the ARRAY is a multi-dimensional ARRAY of BOOL, the filling bits of the dimensions contained are counted as well even if they were not explicitly declared.

Example 1: An ARRAY[1..10,0..4,1..2] of BOOL is handled like an ARRAY[1..10,0..4,1..8] of BOOL or like an ARRAY[0..399] of BOOL.

Example 2: At the IN parameter, an ARRAY[0..5] of WORD (sourceArrayWord[2]) is interconnected. The COUNT_IN parameter has the value "3". At the OUT parameter, an ARRAY[0..1,0..5,0..7] of BOOL (destinationArrayBool[0,0,0]) is interconnected. Both the array at the IN parameter and at the OUT parameter has a size of 96 bits. The ARRAY of WORD is parsed into 48 individual bits.

Note

If the ARRAY low limit of the target ARRAY is not "0", note the following:

For performance reasons the index must always start at a BYTE, WORD or DWORD limit. This means the index must be calculated starting at the low limit of the ARRAY. The following formula is used as basis for this calculation:

Valid indices = ARRAY low limit + n(number of bit sequences) × number of bits of the desired bit sequence

For an ARRAY[-2..45] of BOOL and the bit sequence WORD the calculation looks as follows:

- Valid index (-2) = -2 + 0 × 16
- Valid index (14) = -2 + 1 × 16
- Valid index (30) = -2 + 2 × 16

You can find an example described below.

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1500 series as of firmware version 2.1.

This way you can, for example, parse status words, and read and change the status of the individual bits using the index. Using GATHER, you can merge the bits once again into a bit sequence.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Element of an ARRAY[*] of <bit sequence>	I, Q, M, D, L	ARRAY of <bit sequence> that is parsed. The values must not be located in the I/O area or in the DB of a technology object.
COUNT_IN	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Counter for the number of elements of the source ARRAY that are going to be parsed. The value must not be in the I/O area or in the database of a technology object.
OUT	Output	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	I, Q, M, D, L	ARRAY, STRUCT or PLC data type in which the individual bits are stored

You can select the required bit sequence from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example of a destination ARRAY with the low limit "0"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayWord	Input	ARRAY[0..5] of WORD
CounterInput		UDINT
DestinationArrayBool	Output	ARRAY[0..95] of BOOL

The following example shows how the instruction works:

STL

```
CALL SCATTER_BLK                          // Call the instruction
                                          // Select the required data type from the
                                          "???" drop-down list.
    In := #SourceArrayWord[2]             // The 3rd, 4th and 5th WORD of the #Sour-
                                          ceArrayWord operand is parsed into its in-
                                          dividual bits (48).
    COUNT_IN := #CounterInput             // 3 words are parsed
```

```
STL
    OUT := DestinationArrayBool[0]          // Assign to the individual elements of
                                            the #DestinationArrayBool operand begin-
                                            ning with the 1st element.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayWord	ARRAY[0..5] of WORD (96 bits can be parsed.)
COUNT_IN	CounterInput = 3	UDINT3 (3 WORDs or 48 bits are to be parsed. This means at least 48 bits must be available in the destination ARRAY.)
OUT	DestinationArrayBool[0]	The operand "DestinationArrayBool" is of the data type ARRAY[0..95] of BOOL. This means it provides 96 BOOL elements.

Example of a destination ARRAY with the low limit "-2"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayWord	Input	ARRAY[0..5] of WORD
CounterInput		UDINT
DestinationArrayBool	Output	ARRAY[-2..93] of BOOL

The following example shows how the instruction works:

```
STL
CALL SCATTER_BLK                        // Call the instruction
                                        // Select the required data type from the
                                        "???" drop-down list.
    In := #SourceArrayWord[2]           // The 3rd, 4th and 5th WORD of the #Sour-
                                        ceArrayWord operand is parsed into its in-
                                        dividual bits (48).
    COUNT_IN := #CounterInput           // 3 words are parsed
    OUT := DestinationArrayBool[14]     // And starting with the 16th Element as-
                                        signed to the individual elements of the
                                        #DestinationArrayBool operand.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayWord[2]	ARRAY[0..5] of WORD (96 bits can be parsed.)
COUNT_IN	CounterInput = 3	UDINT3 (3 WORDs or 48 bits are to be parsed. This means at least 48 bits must be available in the destination ARRAY.)
OUT	DestinationArrayBool[14]	The operand "DestinationArray-Bool" is of the data type AR-RAY[-2..93] of BOOL. This means it provides 96 BOOL ele-ments.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

GATHER: Merge individual bits into a bit sequence (S7-1500)

Description

The "Merge individual bits into a bit sequence" instruction merges the bits from an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements into a bit sequence. The bit sequence is saved in a tag of the data type BYTE, WORD, DWORD or LWORD.

Note

Multi-dimensional ARRAY of BOOL

With the "Merge individual bits into a bit sequence" instruction, the use of a multidimensional ARRAY of BOOL is not permitted.

Note

Length of the ARRAY, STRUCT or PLC data type

The ARRAY, STRUCT or the PLC data type must have exactly the number of elements that is specified by the bit sequence.

This means for the data type BYTE, for example, the ARRAY, anonymous STRUCT or the PLC data type must have exactly 8 elements (WORD = 16, DWORD = 32 and LWORD = 64).

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1500 series as of firmware version 2.1.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	I, Q, M, D, L	ARRAY, STRUCT or PLC data type, the bits of which are merged into a bit sequence. The values must not be located in the I/O area or in the DB of a technology object.
OUT	Output	BYTE, WORD, DWORD, LWORD	I, Q, M, D, L	Merged bit sequence, saved in a tag

You can select the required bit sequence from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example with an ARRAY

Create the following tags in the block interface:

Tag	Section	Data type
SourceArray	Input	ARRAY[0..15] of BOOL
DestinationWord	Output	WORD

The following example shows how the instruction works:

STL

```
CALL GATHER                                  // Call the instruction
                                             // Select the required data type from the
                                             "???" drop-down list.
    IN := #SourceArray                       // The bits of the #SourceArray operand
                                             are merged into a WORD.
    OUT := #DestinationWord                  // The 16 merged bits are saved in the
                                             #DestinationWord operand.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArray	The operand "SourceArray" is of the data type ARRAY[0..15] of BOOL. It consists of 16 elements and is therefore just as large as the WORD in which the bits are to be merged.
OUT	DestinationWord	WORD (16 bits)

Example with a PLC data type (UDT)

Create the following PLC data type "myBits":

myBits

	Name	Data type
1	GeneralError	Bool
2	DeviceError	Bool
3	CommError	Bool
4	myError1	Bool
5	myError2	Bool
6	myError3	Bool
7	myError4	Bool
8	myError5	Bool
9	myError6	Bool
10	myError7	Bool
11	myError8	Bool
12	myError9	Bool
13	myError10	Bool
14	myError11	Bool
15	myError12	Bool
16	myError13	Bool

Create the following tags in the block interface:

Tag	Section	Data type
SourceUDT	Input	"myBits"
DestinationWord	Output	WORD

The following example shows how the instruction works:

STL

```
CALL GATHER                                  // Call the instruction
                                             // Select the required data type from the
                                             "???" drop-down list.
    IN := #SourceUDT                         // The bits of the #SourceUDT operand are
                                             merged into a WORD.
    OUT := #DestinationWord                  // The 16 merged bits are saved in the
                                             #DestinationWord operand.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceUDT	The "SourceUDT" operand has the PLC data type (UDT). It consists of 16 elements and is therefore just as large as the WORD in which the bits are to be merged.
OUT	DestinationWord	WORD (16 bits)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

GATHER_BLK: Merge individual bits into multiple elements of an ARRAY of bit sequence (S7-1500)

Description

The "Merge individual bits into multiple elements of an ARRAY of bit sequence" instruction merges the bits from an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements into one or multiple elements of an ARRAY of <bit sequence>. At the COUNT_OUT parameter you specify how many elements of the destination ARRAY are going to be written. With this step you also implicitly specify how many elements of the ARRAY of BOOL, the anonymous STRUCT or the PLC data type are required. The destination ARRAY at the OUT parameter may have more elements than specified at the COUNT_OUT parameter. The ARRAY of <bit sequence> must have sufficient elements to save the bits that are going to be merged. However, the destination ARRAY may also be larger.

Note

Multi-dimensional ARRAY of BOOL

If the ARRAY is a multi-dimensional ARRAY of BOOL, the filling bits of the dimensions contained are counted as well even if they were not explicitly declared.

Example 1: An ARRAY[1..10,0..4,1..2] of BOOL is handled like an ARRAY[1..10,0..4,1..8] of BOOL or like an ARRAY[0..399] of BOOL.

Example 2: At the OUT parameter, an ARRAY[0..5] of WORD (sourceArrayWord[2]) is interconnected. The COUNT_IN parameter has the value "3". At the IN parameter, an ARRAY[0..1,0..5,0..7] of BOOL (destinationArrayBool[0,0,0]) is interconnected. Both the array at the IN parameter and at the OUT parameter has a size of 96 bits. 48 individual bits are merged from the ARRAY of BOOL.

Note

If the ARRAY low limit of the source ARRAY is not "0", note the following:

For performance reasons the index must always start at a BYTE, WORD or DWORD limit. This means the index must be calculated starting at the low limit of the ARRAY. The following formula is used as basis for this calculation:

Valid indices = ARRAY low limit + n(number of bit sequences) × number of bits of the desired bit sequence

For an ARRAY[-2..45] of BOOL and the bit sequence WORD the calculation looks as follows:

- Valid index (-2) = -2 + 0 × 16
- Valid index (14) = -2 + 1 × 16
- Valid index (30) = -2 + 2 × 16

You can find an example described below.

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1500 series as of firmware version 2.1.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	I, Q, M, D, L	ARRAY of BOOL, STRUCT or PLC data type whose bits are merged (source ARRAY) The values must not be located in the I/O area or in the DB of a technology object.
COUNT_OUT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Counter how many elements of the target ARRAY are to be described. The value must not be in the I/O area or in the database of a technology object.
OUT	Output	Element of an ARRAY[*] of <bit sequence>	I, Q, M, D, L	ARRAY of <bit sequence> to which the bits are saved (destination ARRAY)

You can find additional information on valid data types under "See also".

Example of a source ARRAY with the low limit "0"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayBool	Input	ARRAY[0..95] of BOOL
CounterOutput		UDINT
DestinationArrayWord	Output	ARRAY[0..5] of WORD

The following example shows how the instruction works:

STL

```
CALL GATHER_BLK                          // Call the instruction
                                         // Select the required data type from the
                                         "???" drop-down list.
IN := SourceArrayBool[0]                 // As of the 1st element of the #SourceAr-
                                         rayBool operand, 48 bits are merged in
                                         the #DestinationArrayWord operand.
COUNT_OUT := #CounterOutput              // 3 words are merged
OUT := DestinationArrayWord[2]           // The starting point in the destination
                                         ARRAY is the 3rd element. This means that
                                         the first 16 bits are written into the
                                         3rd word, the second 16 nits into the 4th
                                         word and the third 16 bits into the 5th
                                         word of the destination ARRAY.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayBool[0]	The operand "SourceArrayBool" is of the data type ARRAY[0..95] of BOOL. This means it provides 96 BOOL elements that can merged into words once again.
COUNT_OUT	CounterOutput = 3	UDINT3 (3 words are to be written. This means 48 bits must be available in the source ARRAY.)
OUT	DestinationArrayWord[2]	The operand "DestinationArrayWord" is of the data type ARRAY[0..5] of WORD. This means 6 WORD elements are available.

Example of a source ARRAY with the low limit "-2"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayBool	Input	ARRAY[-2..93] of BOOL
CounterOutput		UDINT
DestinationArrayWord	Output	ARRAY[0..5] of WORD

The following example shows how the instruction works:

STL

```
CALL GATHER_BLK                          // Call the instruction
                                         // Select the required data type from the
                                         "???" drop-down list.
IN := SourceArrayBool[14]                // Starting with the 16th element of the
                                         #SourceArrayBool operand, 48 bits are
                                         merged into the #DestinationArrayWord op-
                                         erand.
COUNT_OUT := #CounterOutput              // 3 words are merged
OUT := DestinationArrayWord[2]           // The starting point in the destination
                                         ARRAY is the 3rd element. This means the
                                         first 16 bits of the source ARRAY are ig-
                                         nored. The second 16 Bits are written in-
                                         to the 3rd word, the third 16 bits into
                                         the 4th word and the fourth 16 bits into
                                         the 5th word of the destination ARRAY.
                                         The remaining 64 bits of the source ARRAY
                                         are not taken into account either.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayBool[14]	The operand "SourceArrayBool" is of the data type ARRAY[-2..93] of BOOL. Because the starting point is the 16th element, only 80 BOOL elements that can be merged into words again are available.
COUNT_OUT	CounterOutput = 3	UDINT3 (3 words are to be written. This means 48 bits must be available in the source ARRAY.)
OUT	DestinationArrayWord[2]	The operand "DestinationArrayWord" is of the data type ARRAY[0..5] of WORD. This means 6 WORD elements are available.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

AssignmentAttempt: Attempt assignment of a VARIANT to a reference (S7-1500)

Description

With the "AssignmentAttempt" instruction, you attempt to assign a VARIANT tag to a reference tag. The data type of a reference tag is specified at the time of the declaration, while the data type of a VARIANT tag is determined during runtime. An implicit data type conversion is not permitted with reference tags. To assign the two data types to each other therefore, use the assignment attempt.

With the assignment attempt, a check is made at runtime to find out whether the VARIANT tag is of the correct data type. If this is the case, the instruction is executed. If the instruction is executed successfully, there is a valid reference in the target tag, otherwise a NULL.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC	Input	VARIANT	• Block interface of an FC: Input, Output, InOut, Temp • Block interface of an FB: Input, InOut, Temp	Pointer to the source tag whose address will be read out
DST	Output	Reference to: • Bit sequences, except BOOL, • Integers, • Floating-point numbers, • Character strings, • PLC data types (UDT), • System data types (SDT), • ARRAYs of the named data types	• Block interface of an FC: Input, Output, Temp • Block interface of an FB: Temp	Reference to which the address of the source tag will be transferred

You will find additional information on valid data types under "See also".

Rules

The following rules apply to the assignment attempt: VARIANT tags that do not match these rules return the value "NULL" at runtime.

- The VARIANT must point to an address in the optimized memory area.
- The VARIANT must not point to an address in a temporary memory area.
- If you want to assign a VARIANT to a reference to an ARRAY the following rules apply:
  - The VARIANT tag must point to an ARRAY whose limits exactly match with those of the declared reference. A VARIANT tag which points to an ARRAY [0..9], does not match a tag REF_TO ARRAY[1..10].
  - Moreover you should compile the blocks that form the value of the VARIANT tag once in a CPU of the S7-1500 series, firmware version V2.5.

Example

The following example shows how the instruction works:

In the interface of the block, the VARIANT "myVariant" and the reference tag "myReference" were declared.

		Name	Data type
1		Input	
2		Output	
3		Tag_Out	Bool
4		InOut	
5		myVariant	Variant
6		Static	
7		Temp	
8		myReference	REF_TO Int
9		Constant	

In the program code an attempt is made to assign "myVariant" to the reference tag "myReference". If "myVariant" has the data type "Int" at runtime, "myReference" contains a valid reference to the source tag of the VARIANT, otherwise NULL. If the next query for "NOT_NULL" is true, it means that the assignment was successful and the "Tag_Out" output parameter can be written.

```
1        CALL  AssignmentAttempt
2           IN  :=#myVariant
3           OUT :=#myReference
4        CALL  NOT_NULL
5           OPERAND :=#myReference
6           Ret_Val :=#Tag_Out
7
```

See also

Basic information about references (Page 300)

Assignment attempt from VARIANT to a reference (Page 314)

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

ARRAY DB (S7-1500)

ReadFromArrayDB: Read from ARRAY data block (S7-1500)

Description

The instruction "Read from ARRAY data block" is used to read the element from a data block of the ARRAY DB block type to which the index references, and write the value of the element to the target range.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

Parameters

The following table shows the parameters of the "Read from ARRAY data block" instruction:

Parameters	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB that is read. The specification can be a constant, a global tag or an indexed value.
VALUE	Output [1)]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Value that is read and output

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
1) The VALUE parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, write protected, or is located in load memory.
8135	The ARRAY data block contains invalid values.
8154	The data block has the incorrect data type.
8282	The value at the INDEX parameter is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8452	Code generation error
8453	There are two possible causes of the error: • The size of the VALUE parameter does not match the element length in the ARRAY data block. • The two tags are not in a memory area with optimized access. For additional information on the memory area access types, refer to: Basics of block access (Page 55)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
`CALL ReadFromArrayDB`	`// Calling the instruction`
`DB := "ArrayDB"`	`// Data block from which data is read.`
`INDEX := "ArrayDB".THIS[2]`	`// The 2nd element is read from the data block.`
`RET_VAL := "TagRet_Val"`	`// Error information`
`VALUE := "TargetField"`	`// The read value is provided at the parameter VALUE.`

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type Array [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	TargetField	The "TargetField" operand is a global tag of the data type INT.

The 2nd element is read from "ArrayDB" and written to the "TargetField" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Example of the use of ARRAY data blocks (Page 44)

STL Basics (Page 7245)

WriteToArrayDB: Write to ARRAY data block (S7-1500)

Description

The instruction "Write to ARRAY data block" is used to write the element to which the index references to a data block of the ARRAY DB block type.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

Parameters

The following table shows the parameters of the "Write to ARRAY data block" instruction:

Parameters	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written. The specification can be a constant, a global tag or an indexed value.

Parameters	Declaration	Data type	Memory area	Description
VALUE	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Value to be written
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, or is located in load memory.
8134	The data block is write protected.
8135	The data block is not an ARRAY data block.
8154	The data block has the incorrect data type.
8282	The value at the INDEX parameter is outside the limits of the ARRAY.
8350	The data type VARIANT at parameter VALUE provides the value "0".
8352	Code generation error
8353	There are two possible causes of the error: • The size of the VALUE parameter does not match the element length in the ARRAY data block. • The two tags are not in a memory area with optimized access. For additional information on the memory area access types, refer to: Basics of block access (Page 55)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                                    Explanation
CALL WriteToArrayDB                    // Calling the instruction
DB := "ArrayDB"                        // Data block to which data is written
```

STL	Explanation
INDEX := 2	// The value is written to the element [2] of the data block.
RET_VAL := "TagRet_Val"	// Error information
VALUE := "SourceField"	// The value that is written into the data block.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type Array [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	SourceField	The "SourceField" operand is a global tag of the data type INT.

The value of the "SourceField" operand is written to the 2nd element of the ARRAY DB.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Example of the use of ARRAY data blocks (Page 44)

STL Basics (Page 7245)

ReadFromArrayDBL: Read from array data block in load memory (S7-1500)

Description

The instruction "Read from ARRAY data block in load memory" is used to read the element ti which the index references from a data block of the ARRAY DB block type in the load memory and write it to the target range.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

Parameters

The following table shows the parameters of the "Read from ARRAY data block in load memory" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Begin with the reading of the ARRAY DB
DB [1]	Input	DB_ANY	I, Q, M, D, L	ARRAY data block that is read.
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB that is read. The specification can be a constant, a global tag or an indexed value.
VALUE [1]	InOut	VARIANT	D (element of a global data block) L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Pointer to the DB in the work memory that is to be read and the value of which is to be written. No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the ERROR parameter.
[1] The data blocks must be created with the "Optimized" block property.				

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at the INDEX parameter is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".

Error code* (W#16#...)	Explanation
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL" and "WRIT_DBL" in the corresponding descriptions of the instructions.

Example

The following example shows how the instruction works:

```
STL                                          Explanation
CALL ReadFromArrayDBL, "ReadFromAr-          // Calling the instruction
rayDBL_DB"
REQ := "TagReg"                              // The instruction executes on a positive
                                             signal edge.
DB := "ArrayDB"                              // Data block from which data is read.
INDEX := 2                                   // The element [2] is read from the data
                                             block.
VALUE := "TargetField"                       // The read value is provided at the pa-
                                             rameter VALUE.
BUSY := "TagBusy"                            // Indicates whether the instruction is
                                             still in process.
DONE := "TagDone"                            // Indicates whether the instruction is
                                             closed.
ERROR := "TagError"                          // Error information
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type ARRAY [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	TargetField	The "TargetField" operand is a global tag of the data type INT.
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

The instruction is executed when a positive signal edge is detected at the "TagReq" operand. The 2nd element is read from "ArrayDB" and output at the "VALUE" parameter. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. After the instruction has been processed, the "TagDone" operand has the signal state TRUE.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

READ_DBL: Read from data block in the load memory (Page 2714)

WRIT_DBL: Write to data block in the load memory (Page 2718)

Example of the use of ARRAY data blocks (Page 44)

Difference between synchronous and asynchronous instructions (Page 596)

STL Basics (Page 7245)

WriteToArrayDBL: Write to array data block in load memory (S7-1500)

Description

The instruction "Write to ARRAY data block in load memory" is used to write the element to which the index references to a data block of the ARRAY DB block type in load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

Parameters

The following table shows the parameters of the "Write to ARRAY data block in load memory" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Start writing to the ARRAY DB
DB [1)]	Input	DB_ANY	I, Q, M, D, L	ARRAY data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written. The specification can be a constant, a global tag or an indexed value.

Parameters	Declaration	Data type	Memory area	Description
VALUE [1]	Input	VARIANT	D (element of a global data block) L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Pointer to the DB in the work memory that is to be read and the value of which is to be written. No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The ARRAY DB is still being written
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the ERROR parameter.
[1] The data blocks must be created with the "Optimized" block property.				

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8234	The data block is write protected.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at the INDEX parameter is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL" and "WRIT_DBL" in the corresponding descriptions of the instructions.

Example

The following example shows how the instruction works:

STL	Explanation
CALL WriteToArrayDBL, "WriteToArrayDBL_DB"	// Calling the instruction
REQ := "TagReg"	// The instruction executes on a positive signal edge.
DB := "ArrayDB"	// Data block from which data is read.
INDEX := 2	// The element [2] is read from the data block.
VALUE := "SourceField"	// The read value is provided at the parameter VALUE.
BUSY := "TagBusy"	// Indicates whether the instruction is still in process.
DONE := "TagDone"	// Indicates whether the instruction is closed.
ERROR := "TagError"	// Error information

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type ARRAY [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	SourceField	The "SourceField" operand is a global tag of the data type INT.
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

The instruction is executed when a positive signal edge is detected at the "TagReq" operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written to the 2nd element of "ArrayDB". After the instruction has been processed, the "TagDone" operand has the signal state TRUE.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

READ_DBL: Read from data block in the load memory (Page 2714)

WRIT_DBL: Write to data block in the load memory (Page 2718)

Example of the use of ARRAY data blocks (Page 44)

Difference between synchronous and asynchronous instructions (Page 596)

STL Basics (Page 7245)

Read / Write access (S7-1500)

PEEK: Read memory address (S7-1500)

Description

The "Read memory address" instruction is used to read a memory address from a standard memory area without specifying a data type.

Use of memory area 16#84: DB

If you use a data block as memory area and this is not yet known during the creation of the program code you have the option of using the EN/ENO mechanism to recognize possible access errors:

```
CALL PEEK
AREA := 16#84
DBNUMBER := 1
BYTEOFFSET := 2
RET_VAL := #Peeker

JBI end
L 16#ffff
T #Peeker
end: NOP 0
```

Parameters

The following table shows the parameters of the "Read memory address" instruction:

Parameters	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#1: I/O input
DBNUMBER	Input	DINT, DB_ANY	I, Q, M, D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D	Address to read from Only the 16 least significant bits are used.
RET_VAL	Output	Bit strings	I, Q, M, D	Result of the instruction

You can select the data type of the instruction from the "???" drop-down lists.

Note

If you read the memory address from the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL PEEK                    // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
AREA := "Tag_Area"           // Data block area selected
DBNUMBER := "Tag_DBNumber"   // Number of the data block
BYTEOFFSET := "Tag_Byte"     // Address to read from
RET_VAL := "Tag_Result"      // Result
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
RET_VAL	Tag_Result	Byte value "20"

The instruction reads the value of address "20" from the "Tag_Byte" operand at data block "5" and returns the result at the "Tag_Result" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

PEEK_BOOL: Read memory bit (S7-1500)

Description

The "Read memory bit" instruction is used to read a memory bit from a standard memory area without specifying a data type.

Use of memory area 16#84: DB

If you use a data block as memory area and this is not yet known during the creation of the program code you have the option of using the EN/ENO mechanism to recognize possible access errors:

```
CALL PEEK_BOOL
AREA := 16#84
DBNUMBER := 1
BYTEOFFSET := 2
BITOFFSET := 1
RET_VAL := #Peeker_BOOL

SPBI end
L 16#ffff
T #Peeker_BOOL
end: NOP 0
```

Parameters

The following table shows the parameters of the "Read memory bit" instruction:

Parameters	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#1: I/O input
DBNUMBER	Input	DINT, DB_ANY	I, Q, M, D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D	Address to read from Only the 16 least significant bits are used.
BITOFFSET	Input	INT	I, Q, M, D	Bit to be read from
RET_VAL	Output	BOOL	I, Q, M, D	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list.

Note

If you read the memory bit from the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL PEEK_BOOL               // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
AREA := "Tag_Area"           // Data block area selected
DBNUMBER := "Tag_DBNumber"   // Number of the data block
BYTEOFFSET := "Tag_Byte"     // Address to read from
BITOFFSET := "Tag_Bit"       // Bit to be read from
RET_VAL := "Tag_Result"      // Result
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
RET_VAL	Tag_Result	3

The instruction reads the value of memory bit "3" from the "Tag_Bit" operand at byte "20" of data block "5" and returns the result at the "Tag_Result" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

POKE: Write memory address (S7-1500)

Description

The "Write memory address" instruction is used to write a memory address to a standard memory area without specifying a data type.

Use of memory area 16#84: DB

If you use a data block as memory area and this is not yet known during the creation of the program code you have the option of using the EN/ENO mechanism to recognize possible access errors.

Parameters

The following table shows the parameters of the "Write memory address" instruction:

Parameters	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#2: I/O output
DBNUMBER	Input	DINT, DB_ANY	I, Q, M, D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D	Address to be written Only the 16 least significant bits are used.
VALUE	Input	Bit strings	I, Q, M, D	Value to be written

You can select the data type of the instruction from the "???" drop-down lists.

Note

If you write the memory address to the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL POKE                    // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
AREA := "Tag_Area"           // Data block area selected
DBNUMBER := "Tag_DBNumber"   // Number of the data block
BYTEOFFSET := "Tag_Byte"     // Address to read from
VALUE := "Tag_Value"         // Value to be written
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
VALUE	Tag_Value	16#11

The instruction overwrites the memory address "20" in the data block "5" with value "16#11".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

POKE_BOOL: Write memory bit (S7-1500)

Description

The "Write memory bit" instruction is used to write a memory bit to a standard memory area without specifying a data type.

Use of memory area 16#84: DB

If you use a data block as memory area and this is not yet known during the creation of the program code you have the option of using the EN/ENO mechanism to recognize possible access errors.

Parameters

The following table shows the parameters of the "Write memory bit" instruction:

Parameters	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#2: I/O output
DBNUMBER	Input	DINT, DB_ANY	I, Q, M, D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D	Address to be written Only the 16 least significant bits are used.
BITOFFSET	Input	INT	I, Q, M, D	Bit to be written
VALUE	Input	BOOL	I, Q, M, D	Value to be written

You can select the data type of the instruction from the "???" drop-down list.

Note

If you write the memory bit to the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL POKE_BOOL               // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
AREA := "Tag_Area"           // Data block area selected
DBNUMBER := "Tag_DBNumber"   // Number of the data block
BYTEOFFSET := "Tag_Byte"     // Address to read from.
BITOFFSET := "Tag_Bit"       // Bit to read from
VALUE := "Tag_Value"         // Value to be written
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
VALUE	Tag_Value	M0.0

The instruction overwrites the memory bit "3" in data block "5" in byte "20" with the value "M0.0".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

POKE_BLK: Write memory area (S7-1500)

Description

The "Write memory area" instruction is used to write a memory area to a different standard memory area without specifying a data type.

Use of memory area 16#84: DB

If you use a data block as memory area and this is not yet known during the creation of the program code you have the option of using the EN/ENO mechanism to recognize possible access errors.

Parameters

The following table shows the parameters of the "Write memory area" instruction:

Parameters	Declaration	Data type	Memory area	Description
AREA_SRC	Input	BYTE	I, Q, M, D	The following areas can be selected in the source memory area: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB
DBNUMBER_SRC	Input	DINT, DB_ANY [1]	I, Q, M, D	Number of the data block in the source memory area, if AREA = DB, otherwise "0"
BYTEOFFSET_SRC	Input	DINT	I, Q, M, D	Address in the source memory area to be written Only the 16 least significant bits are used.
AREA_DEST	Input	BYTE	I, Q, M, D	The following areas can be selected in the destination memory area: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB
DBNUMBER_DEST	Input	DINT, DB_ANY [1]	I, Q, M, D	Number of the data block in the destination memory area, if AREA = DB, otherwise "0" Only the 16 least significant bits are used.
BYTEOFFSET_DEST	Input	DINT	I, Q, M, D	Address in the destination memory area to be written Only the 16 least significant bits are used.
COUNT	Input	DINT	I, Q, M, D	Number of bytes which are moved
[1] The data types of the parameters DBNUMBER_SRC and DBNUMBER_DEST must have the same data type. In other words, both tags must be either data type DINT or data type DB_ANY.				

You can select the data type of the instruction from the "???" drop-down list.

Note

If you write the memory address to the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

```
STL                              Explanation
CALL POKE_BLK                    // The instruction is called.
                                 // Select the required data type from the "???" drop-
                                 down list.
AREA_SRC := "Tag_Source_Area"    // Data block area selected in source memory area
DBNUMBER_SRC :=                  // Number of the data block in the source memory area
"Tag_Source_DBNumber"
BYTEOFFSET_SRC :=                // Address to read from in source memory area
"Tag_Source_Byte"
AREA_DEST :=                     // Data block area selected in destination memory area
"Tag_Source_Area"
DBNUMBER_DEST :=                 // Number of the data block in the destination memory
"Tag_Source_DBNumber"            area
BYTEOFFSET_DEST :=               // Address to read from in destination memory area
"Tag_Source_Byte"
COUNT := "Tag_Count"             // Number of moved bytes
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
AREA_SRC	Tag_Source_Area	16#84
DBNUMBER_SRC	Tag_Source_DBNumber	5
BYTEOFFSET_SRC	Tag_Source_Byte	20
AREA_DEST	Tag_Destination_Area	16#83
DBNUMBER_DEST	Tag_Destination_DBNumber	0
BYTEOFFSET_DEST	Tag_Destination_Byte	30
COUNT	Tag_Count	100

The instruction writes 100 byte from data block "5" starting with address "20" in the memory area of the bit memory starting at address "30".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

READ_LITTLE: Read data in little endian format (S7-1500)

Description

The instruction "Read data in little endian format" is used to read data from a memory area and to write this to a single tag in the little endian byte sequence. With the little endian format, the byte with the least significant bits is saved first, which means at the lowest memory address.

The parameters SRC_ARRAY and DEST_VARIABLE are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at parameter DEST_VARIABLE has to be an elementary data type. The VARIANT at parameter SRC_ARRAY points to the memory area from which is read, and this must be an ARRAY of BYTE.

You can also use an actual parameter with variable ARRAY index for the parameters SRC_ARRAY and DEST_VARIABLE.

The operand at the POS parameter determines the position in the memory area at which reading starts.

Note

Reading tag of data type VARIANT or BOOL

If you want to read a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to read a tag of the data type BOOL, use a "Slice access".

Parameters

The following table shows the parameters of the "Read data in little endian format" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC_ARRAY	Input	ARRAY of BYTE	I, Q, M, D, L	Memory area to be read from
OUT	Output	INT	I, Q, M, D, L	Error information
DEST_VARIABLE	Output	Bit strings, integers, floating-point numbers, timers, LDT, TOD, LTOD, DATE, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L	Read value
POS	InOut	DINT	I, Q, M, D, L	Determines the position at which the reading starts. The POS parameter is calculated zero-based.

Parameter OUT

The following table shows the meaning of the values of the OUT parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

STL	Explanation
CALL READ_LITTLE	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
SRC_ARRAY := #SourceField	// The integer 1_295_788_826 is read from the memory area #SourceField.
OUT := "Tag_Error"	// Error information
DEST_VARIABLE := #DINTVari-able	// The integer is written to the #DINTVariable operand in little endian format. The data type at the DEST_VAR-IABLE parameter specifies how many bytes are read.
POS := #TagPos	// The quantity 4 is saved at the "#TagPos" operand.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
SRC_ARRAY	#SourceField	ARRAY[0..3] of BYTE := 16#1A, 16#2B, 16#3C, 16#4D
DEST_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
POS	#TagPos	0 => 4

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

WRITE_LITTLE: Write data in little endian format (S7-1500)

Description

You use the "Write data in little endian format" instruction to write the data of a single tag in the little endian byte sequence to a memory area. With the little endian format, the byte with the least significant bits is saved first, which means at the lowest memory address.

The parameters SRC_VARIABLE and DEST_ARRAY are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at the SRC_VARIABLE parameter must point to an elementary data type. The VARIANT at the DEST_ARRAY parameter points to the memory area to which the data is written, and this must be an ARRAY of BYTE.

You can also use an actual parameter with variable ARRAY index for the parameters SRC_ARRAY and DEST_VARIABLE.

The operand at the POS parameter determines the position in the memory area at which writing starts.

Note

Writing tag of data type VARIANT or BOOL

If you want to write a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to write a tag of the data type BOOL, use a "Slice access".

Parameters

The following table shows the parameters of the "Write data in little endian format" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC_VARIABLE	Input	Bit strings, integers, floating-point numbers, timers, LDT, TOD, LTOD, DATE, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L	Tag whose data are written
OUT	Output	INT	I, Q, M, D, L	Error information
DEST_ARRAY	InOut	ARRAY of BYTE	I, Q, M, D, L	Memory area to which the data are written
POS	InOut	DINT	I, Q, M, D, L	Determines the position at which the writing starts. The POS parameter is calculated zero-based.

Parameter OUT

The following table shows the meaning of the values of the OUT parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

STL	Explanation
CALL WRITE_LITTLE	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
SRC_VARIABLE := #DINTVariable	// The integer 1_295_788_826 from the #DINTVariable operand is written in little endian format to the memory area #TargetField. The data type at the SRC_VARIABLE parameter specifies how many bytes are written.
OUT := "Tag_Error"	// Error information
DEST_ARRAY := #TargetField	// Memory area
POS := #TagPos	// The quantity 4 is saved at the "#TagPos" operand.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
SRC_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
DEST_ARRAY	#TargetField	ARRAY[0..10] of BYTE = 16#1A, 16#2B, 16#3C, 16#4D
POS	#TagPos	0 => 4

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

READ_BIG: Read data in big endian format (S7-1500)

Description

The instruction "Read data in little endian format" is used to read data from a memory area and to write this to a single tag in the big endian byte sequence. With the big endian format, the byte with the most significant bits is saved first, which means at the lowest memory address.

The parameters SRC_ARRAY and DEST_VARIABLE are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at parameter DEST_VARIABLE has to be an elementary data type. The VARIANT at the SRC_ARRAY parameter points to the memory area to be read from, and this must be an ARRAY of BYTE.

You can also use an actual parameter with variable ARRAY index for the parameters SRC_ARRAY and DEST_VARIABLE.

The operand at the POS parameter determines the position in the memory area at which reading starts.

Note

Reading tag of data type VARIANT or BOOL

If you want to read a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to read a tag of the data type BOOL, use a "Slice access".

Parameters

The following table shows the parameters of the "Read data in big endian format" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC_ARRAY	Input	ARRAY of BYTE	I, Q, M, D, L	Memory area to be read from
OUT	Output	INT	I, Q, M, D, L	Error information
DEST_VARIABLE	Output	Bit strings, integers, floating-point numbers, timers, LDT, TOD, LTOD, DATE, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L	Read value
POS	InOut	DINT	I, Q, M, D, L	Determines the position at which the reading starts. The POS parameter is calculated zero-based.

Parameter OUT

The following table shows the meaning of the values of the OUT parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

STL	Explanation
CALL READ_BIG	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
SRC_ARRAY := #SourceField	// The integer 439_041_101 is read from the memory area #SourceField.
OUT := "Tag_Error"	// Error information
DEST_VARIABLE := #DINTVariable	// The integer is written to the #DINTVariable operand in big endian format. The data type at the DEST_VARIABLE parameter specifies how many bytes are read.
POS := #TagPos	// The quantity 4 is saved at the "#TagPos" operand.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
SRC_ARRAY	#SourceField	ARRAY[0..10] of BYTE := 16#1A, 16#2B, 16#3C, 16#4D
DEST_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
POS	#TagPos	0 => 4

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

WRITE_BIG: Write data in big endian format (S7-1500)

Description

You use the "Write data in big endian format" instruction to write the data of a single tag in the big endian byte sequence to a memory area. With the big endian format, the byte with the most significant bits is saved first, which means at the lowest memory address.

The parameters SRC_VARIABLE and DEST_ARRAY are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at the SRC_VARIABLE parameter must point to an elementary data type. The VARIANT at the DEST_ARRAY parameter points to the memory area to which the data is written, and this must be an ARRAY of BYTE.

You can also use an actual parameter with variable ARRAY index for the parameters SRC_ARRAY and DEST_VARIABLE.

The operand at the POS parameter determines the position in the memory area at which writing starts.

Note

Writing tag of data type VARIANT or BOOL

If you want to write a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to write a tag of the data type BOOL, use a "Slice access".

Parameters

The following table shows the parameters of the "Write data in big endian format" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC_VARIABLE	Input	Bit strings, integers, floating-point numbers, timers, LDT, TOD, LTOD, DATE, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L	Tag whose data are written
OUT	Output	INT	I, Q, M, D, L	Error information
DEST_ARRAY	InOut	ARRAY of BYTE	I, Q, M, D, L	Memory area to which the data are written
POS	InOut	DINT	I, Q, M, D, L	Determines the position at which the writing starts. The POS parameter is calculated zero-based.

Parameter OUT

The following table shows the meaning of the values of the OUT parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

STL	Explanation
CALL WRITE_BIG	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
SRC_VARIABLE := #DINTVariable	// The integer 439_041_101 from the #DINTVariable operand is written in big endian format to the memory area #TargetField. The data type at the SRC_VARIABLE parameter specifies how many bytes are written.
OUT := "Tag_Error"	// Error information
DEST_ARRAY := #TargetField	// Memory area
POS := #TagPos	// The quantity 4 is saved at the "#TagPos" operand.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
SRC_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
DEST_ARRAY	#TargetField	ARRAY[0..10] of BYTE = 16#1A, 16#2B, 16#3C, 16#4D
POS	#TagPos	0 => 4

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

VARIANT (S7-1500)

VariantGet: Read out VARIANT tag value (S7-1500)

Description

You can use the "Read out VARIANT tag value" instruction to read the value of the tag to which the VARIANT at the SRC parameter points and write it in the tag at the DST parameter.

The SRC parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the DST parameter.

The data type of the tag specified at the DST parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT instruction: Move block" instruction. For additional information, refer to "See also".

Parameters

The following table shows the parameters of the "Read out VARIANT tag value" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Tag to be read
DST	Output	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`CALL VariantGet`	`// The instruction is called`
`SRC := #TagIn_Source`	`// The value of the tag to which the VARIANT at the #TagIn_Source operand points is read.`
`DST := "TagOut_Dest"`	`// The content of the tag is written to the "TagOut_Dest" operand.`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on VARIANT (Page 324)

STL Basics (Page 7245)

VariantPut: Write VARIANT tag value (S7-1500)

Description

You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter to the memory at the DST parameter to which the VARIANT points.

The DST parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT instruction: Move block" instruction. For additional information, refer to "See also".

Parameters

The following table shows the parameters of the instruction "Write VARIANT tag value":

Parameters	Declaration	Data type	Memory area	Description
SRC	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L, P	Tag to be read
DST	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                              Explanation
CALL VariantPut                  // The instruction is called.
SRC := "TagIn_Source"            // The value of the "TagIn_Source" operand is read...
DST := #TagIn_Dest               // and written to the tag to which the VARIANT at
                                 the #TagIn_Dest operand points.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on VARIANT (Page 324)

MOVE_BLK_VARIANT: Move block (Page 1347)

STL Basics (Page 7245)

CountOfElements: Get number of ARRAY elements (S7-1500)

Description

You use the "Get number of ARRAY elements" instruction to query how many ARRAY elements a tag to which VARIANT points has.

If it is a single-dimensional ARRAY, the number of ARRAY elements is output as the result. (The difference between the high and low limit +1). If it is a multi-dimensional ARRAY, the number of all dimensions is output as the result.

If you want to query the elements of an ARRAY DB, you should use the instructions "ReadFromArrayDB" or "WriteFromArrayDB", because here a more precise error evaluation for the number of elements is possible.

Note

Instances

The VARIANT pointer can point to no instance and therefore also to no multi-instance or an ARRAY of multi-instances.

Note

ARRAY within a data block

If you want to query the number of the elements of an ARRAY which is located in a data block, the block attribute "Data block write-protected in the device" must not be activated in this block. Otherwise, the RET_VAL parameter returns the result "0", irrespective of how many elements the ARRAY contains.

The result is also "0" if the VARIANT tag is not an ARRAY.

If the VARIANT points to an ARRAY of BOOL, the fill elements are included in the count. (For example, 8 is returned for an ARRAY[0..1] of BOOL).

Parameters

The following table shows the parameters of the "Get number of ARRAY elements" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Tag to be queried
RET_VAL	Output	UDINT	I, Q, M, D, L	Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`CALL CountOfElements`	`// The instruction is called.`
`IN := #TagIn_Source`	`// The number of ARRAY elements of the tag to which the VARIANT at the #TagIn_Source operand points is read...`
`RET_VAL := "TagOut_RetVal"`	`// and output at the "TagOut_RetVal" operand.`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on VARIANT (Page 324)

STL Basics (Page 7245)

ARRAY[*] (S7-1500)

LOWER_BOUND: Read out low ARRAY limit (S7-1500)

Description

You can declare tags of the data type ARRAY[*] in the block interface of a function block or function. For these local tags, you can read out the limits of the ARRAY. You will need to specify the required dimension at the DIM parameter.

The "Read out ARRAY low limit" instruction is used to read out the variable low limit of the ARRAY.

Note

Availability of the instruction

The instruction is available for a CPU of the S7-1500 series as of firmware version >= 2.0.

Parameters

The following table shows the parameters of the instruction "Read out ARRAY low limit":

Parameters	Declaration	Data type	Memory area	Description
ARR	Input	ARRAY[*]	FB => Section In-Out FC => Sections Input and InOut	ARRAY of which the variable low limit is to be read.
DIM	Input	UDINT	I, Q, M, D, L, P or constant	Dimension of the ARRAY of which the variable low limit is to be read.
OUT	Output	DINT	I, Q, M, D, L, P	Result

Example

The following example shows how the instruction works:

STL	Explanation
CALL LOWER_BOUND	// The instruction is called.
ARR := #ARRAY_A	// The variable low limit of the ARRAY #ARRAY_A is to be read.
DIM := 2	// The variable low limit of the second dimension is to be read.
OUT := "Result"	// Result of the instruction

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on ARRAY (Page 283)

STL Basics (Page 7245)

UPPER_BOUND: Read out high ARRAY limit (S7-1500)

Description

You can declare tags of the data type ARRAY[*] in the block interface of a function block or function. For these local tags, you can read out the limits of the ARRAY. You will need to specify the required dimension at the DIM parameter.

The instruction "Read out ARRAY high limit" is used to read out the variable high limit of the ARRAY.

Note

Availability of the instruction

The instruction is available for a CPU of the S7-1500 series as of firmware version >= 2.0.

Parameters

The following table shows the parameters of the instruction "Read out ARRAY high limit":

Parameters	Declaration	Data type	Memory area	Description
ARR	Input	ARRAY[*]	FB => Section In-Out FC => Sections Input and InOut	ARRAY of which the variable high limit is to be read.
DIM	Input	UDINT	I, Q, M, D, L, P or constant	Dimension of the ARRAY of which the variable high limit is to be read.
OUT	Output	DINT	I, Q, M, D, L, P	Result

Example

The following example shows how the instruction works:

STL	Explanation
`CALL UPPER_BOUND`	`// The instruction is called.`
`ARR := #ARRAY_A`	`// The variable high limit of the ARRAY #ARRAY_A is to be read.`
`DIM := 2`	`// The variable high limit of the second dimension is to be read.`
`OUT := "Result"`	`// Result of the instruction`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on ARRAY (Page 283)

STL Basics (Page 7245)

Legacy (S7-1500)

BLKMOV: Move block (S7-1500)

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can also use this tag in memory areas "with optimized block access".

The following figure shows the principle of the move operation:

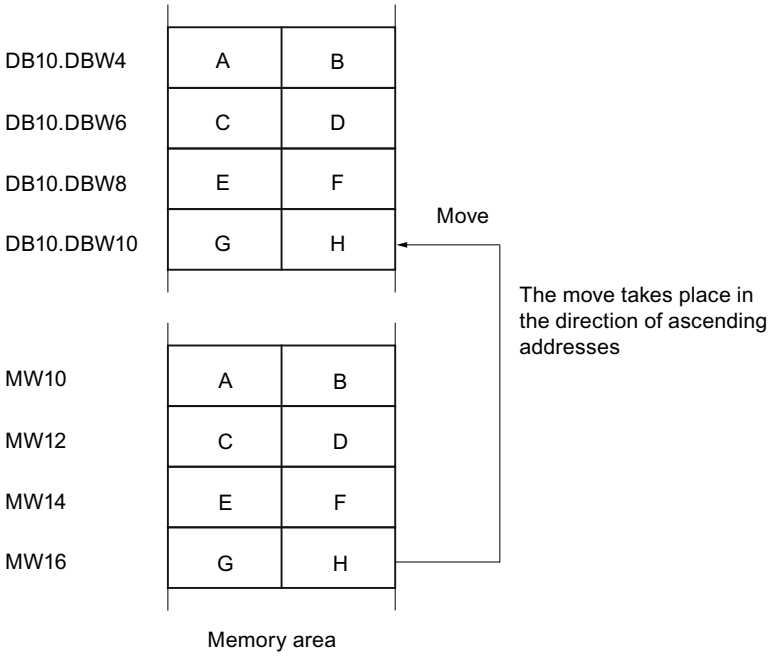

Consistency of source and destination data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1]	VARIANT	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL BLKMOV                  // The "Move block" instruction is called.
SRCBLK := P#M100.0 BYTE 10   // 10 Bytes starting from MB100 are moved.
RET_VAL := "Tag_ErrorCode"   // If an error occurs during the move operation, its
                             error code is output in the variable "Tag_ErrorCode".
DSTBLK := P#M200.0 BYTE 10   // The moved bytes are written to the 10 bytes start-
                             ing from MB200
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1441)

STL Basics (Page 7245)

UBLKMOV: Move block uninterruptible (S7-1500)

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can also use this tag in memory areas "with optimized block access".

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as the area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1]	VARIANT	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source or target area is only available in the load memory.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the DSTBLK parameter.

Error code* (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
CALL UBLKMOV	// The "Move block uninterruptible" instruction is called.
SRCBLK := P#M100.0 BYTE 10	// 10 Bytes starting from MB100 are moved.
RET_VAL := "Tag_ErrorCode"	// If an error occurs during the move operation, its error code is output in the variable "Tag_ErrorCode".
DSTBLK := P#M200.0 BYTE 10	// The moved bytes are written to the 10 bytes starting from MB200

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1441)

STL Basics (Page 7245)

FILL: Fill block (S7-1500)

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can also use this tag in memory areas "with optimized block access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK instruction: Fill block" instruction.

The following figure shows the principle of the move operation:

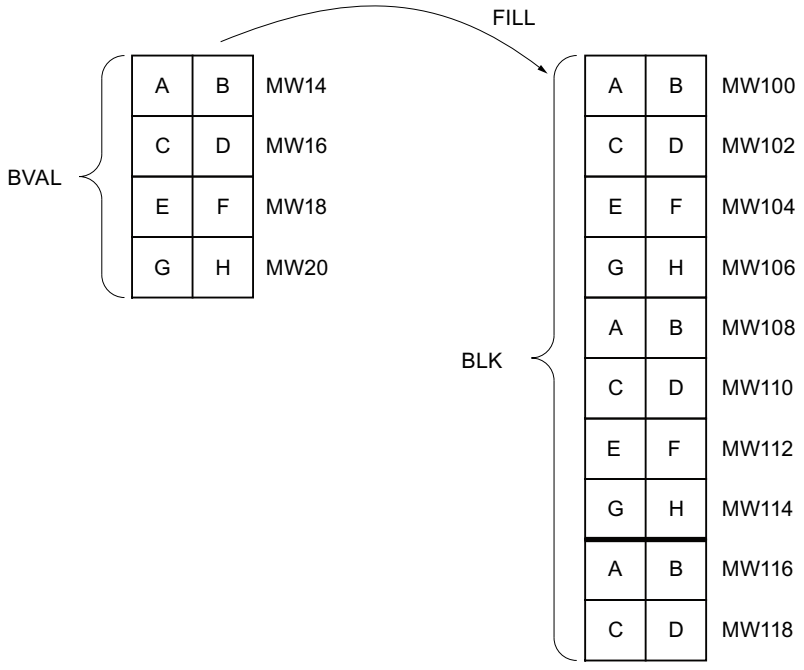

Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory

- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
BVAL	Input	VARIANT	I, Q, M, D, L, P	Specification of the memory area (source area) with the content used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
BLK	Output [1)]	VARIANT	I, Q, M, D, L, P	Specification of the memory area that is filled with the content of the source area.
1) The BLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the BVAL parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the BLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
`CALL FILL`	`// The "Fill block" instruction is called.`
`BVAL := P#M14.0 WORD 4`	`// The area from MW14 to MW20 is moved`
`RET_VAL := "Tag_ErrorCode"`	`// Error information`
`BLK := P#M100.0 WORD 10`	`// The memory area from MW100 to MW118 is filled with the content of 4 words that are contained in the memory area on the BVAL parameter`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1441)

STL Basics (Page 7245)

Conversion operations (S7-1500)

SCALE_X: Scale (S7-1500)

Description

You can use the "Scale" instruction to scale the value at the VALUE input by mapping it to a specified value range. When the "Scale" instruction is executed, the floating-point value at the VALUE input is scaled to the value range that was defined by the MIN and MAX parameters. The result of the scaling is an integer, which is stored in the RET_VAL output.

The following figure shows an example of how values can be scaled:

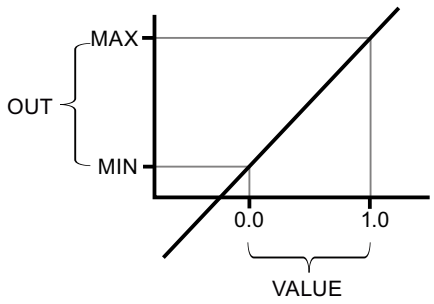

The "Scale" instruction works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

Note

For more information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameters	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
RET_VAL	Output	Integers, floating-point numbers	I, Q, M, D, L	Result of scaling

You can select the data type of the instruction from the "???" drop-down lists.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SCALE_X                 // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
MIN := "Tag_Minimum"         // Low limit
VALUE := "Tag_Value"         // Floating-point value
MAX := "Tag_Maximum"         // High limit
RET_VAL := "Tag_Result"      // Result of the instruction
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MIN	Tag_Minimum	10
VALUE	Tag_Value	0.5
MAX	Tag_Maximum	30
OUT	Tag_Result	20

The value at the "Tag_Value" input is scaled to the range of values defined by the values at the "Tag_Minimum" and "Tag_Maximum" inputs. The result is stored in the "Tag_Result" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Declaring global constants (Page 7089)

STL Basics (Page 7245)

NORM_X: Normalize (S7-1500)

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. The result at the RET_VAL output is calculated and stored as a floating-point number depending on the location of the value to be normalized within this value range. If the value to be normalized is equal to the value at the MIN input, the OUT output has the value "0.0". If the value to be normalized is equal to the value at input MAX, output OUT returns the value "1.0".

The following figure shows an example of how values can be normalized:

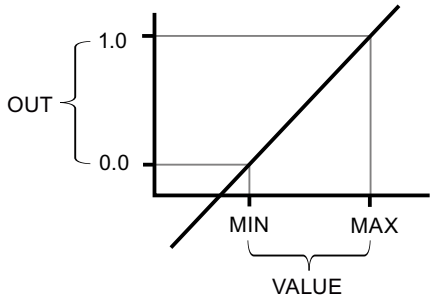

The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

Note

For more information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the instruction "Normalize":

Parameters	Declaration	Data type	Memory area	Description
MIN [1]	Input	Integers, floating-point numbers	I, Q, M, D, L	Low limit of the value range
VALUE [1]	Input	Integers, floating-point numbers	I, Q, M, D, L	Value to be normalized
MAX [1]	Input	Integers, floating-point numbers	I, Q, M, D, L	High limit of the value range
RET_VAL	Output	Floating-point numbers	I, Q, M, D, L	Result of the normalization
[1] If you use constants in these three parameters, you only need to declare one of them.				

You can select the data type of the instruction from the "???" drop-down lists.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL NORM_X                  // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
MIN := "Tag_Minimum"         // Low limit
VALUE := "Tag_Value"         // Floating-point value
MAX := "Tag_Maximum"         // High limit
RET_VAL := "Tag_Result"      // Result of the instruction
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MIN	Tag_Minimum	10
VALUE	Tag_Value	20
MAX	Tag_Maximum	30
RET_VAL	Tag_Result	0.5

The value at the "Tag_Value" input is mapped to the range of values that were defined by the values at the "Tag_Minimum" and "Tag_Maximum" inputs. The tag value at the "Tag_Value" input is normalized to the defined value range. The result is stored as a floating-point number in the "Tag_Result" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Declaring global constants (Page 7089)

STL Basics (Page 7245)

VARIANT (S7-1500)

VARIANT_TO_DB_ANY: Convert VARIANT to DB_ANY (S7-1500)

Description

You use the "Convert VARIANT to DB_ANY" instruction to query the data block number that the operand that is specified at the IN parameter addresses. This can be an instance data block or an ARRAY data block. The operand at the IN parameter has the data type VARIANT, which means you do not need to know the data type of the data block whose number is to be queried when the program is created. The data block number is read during runtime and written to the operand specified at the RET_VAL parameter.

Requirements

If the requirements are met, the instruction is executed. If the requirements are not met, "0" is output as data block number.

The output tag...	homed...	Conversion options
VARIANT	... a data block which is an instance data block of a PLC data type or a system data type (SDT).	The output tag can be converted to a data block number.
VARIANT	... a data block which is an ARRAY DB.	The output tag can be converted to a data block number.
VARIANT	... a tag with an elementary data type.	It is not possible to convert the output tag to a database number because a data block can never comprise only one elementary data type.
VARIANT	... a structure within a data block.	It is not possible to convert the output tags to a database number because it is only a part within the data block.

Parameters

The following table shows the parameters of the "Convert VARIANT to DB_ANY" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Tag to be read. (The function value of the "DB_ANY_TO_VARIANT" instruction). You can use a local or global tag at the IN parameter.
RET_VAL	Output	DB_ANY	I, Q, M, D, L	Result: Number of the DB
ERR	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

ERR parameter

The following table shows the meaning of the values of the ERR parameter:

Error code* (W#16#...)	Explanation
0000	No error
252C	The VARIANT data type at the IN parameter has the value "0" and the CPU changes to STOP mode.
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8131	The data block does not exist, is too short, or is located in load memory.
8132	The data block is too short and not an ARRAY data block.
8150	The data type VARIANT at parameter IN provides the value "0". To receive this error message, the "Handle errors within block" block property must be activated. Otherwise the CPU changes to STOP mode and sends the error code 16#252C.
8153	The VARIANT data type at the IN parameter does not point to the start of an ARRAY data block or the length of the VARIANT does not match that of the data block.
8154	The data block has the incorrect data type.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
`CALL VARIANT_TO_DB_ANY`	`// The instruction is called.`
`IN := #tempVARIANT`	`// The number of a data block that is specified at the tempVARIANT operand is read. Because the operand has the data type VARIANT, you do not need to know the data type of the tag during program creation.`

```
STL                                      Explanation
RET_VAL := "OutputDBNumber"              // The number is written to the "OutputDB-
                                         Number" tag which has the data type
                                         DB_ANY.
ERR := "Tag_Error"                       // Error information
```

The following table shows how the instruction works using specific operand values:

Parameters	Declaration in the block interface	Operand	Value
IN	Input	tempVARIANT	-
RET_VAL	Output	OutputDBNumber	11

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on VARIANT (Page 324)

Using the DB_ANY data type (Page 204)

STL Basics (Page 7245)

DB_ANY_TO_VARIANT: Convert DB_ANY to VARIANT (S7-1500)

Description

The "Convert DB_ANY to VARIANT" instruction is used to generate a VARIANT tag from a data block that meets the requirements listed below. The operand at the IN parameter has the data type DB_ANY, which means that the data block does not have to be known when the program is created. The data block number is read during runtime.

Requirements

If the requirements are met, the instruction is executed. If the requirements are not met or the data block does not exist, the value NULL is output at the RET_VAL parameter. All other accesses with the RET_VAL tag fail.

The input tag of data type ...	homed...	Conversion options
DB_ANY	...a data block which is an instance data block of a PLC data type or a system data type (SDT).	Conversion is possible
DB_ANY	...a data block which is an ARRAY DB.	Conversion is possible
DB_ANY	...a data block which is an instance data block of a function block or a global data block.	Conversion is not possible

Parameters

The following table shows the parameters of the "Convert DB_ANY to VARIANT" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DB_ANY	I, Q, M, D, L	Data block whose number is read. You can use a local or global tag at the IN parameter.
RET_VAL	Output [1]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Number of the data block
ERR	Output	INT	I, Q, M, D, L	Error information
1) The RET_VAL parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

ERR parameter

The following table shows the meaning of the values of the ERR parameter:

Error code* (W#16#...)	Explanation
0000	No error
8130	The number of the data block is "0"
8131	The data block does not exist, is too short, or is located in load memory.
8132	The data block is too short and not an ARRAY data block.
8134	The data block is write protected.
8154	The data block has the incorrect data type.
8155	The data block has an unknown data type. [1]
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

[1] The reason for the error code #8155 to be output is:

A PLC data type (UDT1) was declared and a data block (DB2) of the data type "UDT1" was created. The tag table contains a tag (3) of the data type DB_ANY. Next the instruction "DB_ANY_TO_VARIANT" was called in a program block (4) and supplied with the tag (3) at the IN parameter. During execution the instruction "DB_ANY_TO_VARIANT" returns the error code 16#8155.

Follow these steps to resolve the error code:

1. Create a function (FC5) and declare a tag of the data type VARIANT at the InOut interface.
2. Create an additional function (FC6) and call FC5 in it.
3. Create a tag (7) of the data type "UDT1" in FC6 in the Temp interface.

4. Supply the InOut interface of the FC5 with tag (7).
5. Compile and download the two blocks (FC5 and FC6) to your CPU. You do not need to call these blocks (FC5 and FC6) in the user program.

Result:

The error code 16#8155 is no longer output because the user program is now familiar with the data type.

This procedure is not necessary if you call one of the two instructions "VariantGet" or "VariantPut" after calling the instruction "DB_ANY_TO_VARIANT".

Example

The following example shows how the instruction works:

STL	Explanation
`CALL DB_ANY_TO_VARIANT`	`// The instruction is called.`
`IN := "InputDB"`	`// The number of any data block that is specified at the "InputDB" operand is used to generate a tag of data type VARIANT that addresses the data block. Because the operand at the IN parameter has the DB_ANY data type, the data block that will be used during runtime does not have to be known when the program is created (neither the name nor the number of the data block).`
`RET_VAL := #tempVARIANT`	`// Because the operand at the RET_VAL parameter has the data type VARIANT, you do not need to know the data type of the data block when the program is created.`
`ERR := "Tag_Error"`	`// Error information`

The following table shows how the instruction works using specific operand values:

Parameters	Declaration in the block interface	Operand	Value
IN	Input	InputDB	11
RET_VAL	Temp	tempVARIANT	-

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on VARIANT (Page 324)

Using the DB_ANY data type (Page 204)

STL Basics (Page 7245)

Legacy (S7-1500)

SCALE: Scale (S7-1500)

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number that can be scaled in physical units between a low limit value and a high limit value. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case, the constant "K1" has the value -27648.0 and the constant "K2" the value +27648.0.
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case, the constant "K1" has the value 0.0 and the constant "K2" the value +27648.0.

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled inversely proportionate to the input value.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1".
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                              Explanation
CALL SCALE                       // The instruction is called.
IN := "Tag_InputValue"           // Specifies the value to be converted and scaled
HI_LIM := "Tag_HighLimit"        // High limit
LO_LIM := "Tag_LowLimit"         // Low limit
BIPOLAR := "Tag_Bipolar"         // Indicates whether the value at the IN parameter is
                                 to be interpreted as bipolar or unipolar
RET_VAL := "Tag_ErrorCode"       // Error information
OUT := "Tag_OutputValue"         // Result of the instruction
```

The following table shows the values of the various operands before execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	50.03978588

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1441)

Evaluating errors with output parameter RET_VAL (Page 162)

STL Basics (Page 7245)

UNSCALE: Unscale (S7-1500)

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case, the constant "K1" has the value -27648.0 and the constant "K2" the value +27648.0.
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case, the constant "K1" has the value 0.0 and the constant "K2" the value +27648.0.

When the value at parameter IN is not within the limits defined by HI_LIM and LO_LIM, an error is output and the result is set to the nearest limit.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL UNSCALE                 // The instruction is called.
IN := "Tag_InputValue"       // Specifies the value to be unscaled
HI_LIM := "Tag_HighLimit"    // High limit
LO_LIM := "Tag_LowLimit"     // Low limit
BIPOLAR := "Tag_Bipolar"     // Indicates whether the value at the IN parameter is
                             to be interpreted as bipolar or unipolar
RET_VAL := "Tag_ErrorCode"   // Error information
```

```
STL                          Explanation
OUT := "Tag_OutputValue"     // Result of the instruction
```

The following table shows the values of the various operands before execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	22

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

GET_ERR_ID: Get error ID locally (Page 1441)

Evaluating errors with output parameter RET_VAL (Page 162)

STL Basics (Page 7245)

Program control operations (S7-1500)

Runtime control (S7-1500)

ENDIS_PW: Limit and enable password legitimation (S7-1500)

Description

You use the "Limit and enable password legitimation" instruction to specify whether passwords configured for the CPU are legitimized or not. In this way, you can prevent legitimized connections, even if the correct password is known.

When you call the instruction and the REQ parameter has the signal state "0", the currently set state is displayed at the output parameters. If changes have been made to the input parameters, these are not transferred to the output parameters.

When the instruction is called and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD).

- If signal state "0" is present, authorization via password is not allowed.
- If signal state "1" is present, the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Passwords that are not configured must have the signal state TRUE at the input and return the signal state TRUE at the output. The fail-safe password can only be configured for an F-CPU and must therefore always be interconnected with the signal state TRUE in a standard CPU. If the instruction returns an error, the call has no effect and the previous lock is still in effect.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the mode selector to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Plugging an empty memory card (transfer card or program card) into an S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called once again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access from HMI systems if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing accidental lockout of an S7-1500 CPU

The settings can be made on the front panel of the CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode switch to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing accidental lockout of an S7-1200 CPU

Because the S7-1200 CPU does not have a mode selector, protection is disabled with POWER OFF - POWER ON. This means that it is possible and advisable to prevent accidental lockout with the help of specific program sequences within your program.

To do so, program a time control either by means of a cyclic interrupt OB or a timer in the Main OB (OB 1). This gives you the option of calling the "Limit and enable password legitimation" instruction in the respective OB (e.g., OB 1 or OB 35) relatively quickly after a transition from POWER OFF to POWER ON and the associated disabling of the protection. Call the instruction in the startup OB (OB 100) to keep the time window in which the instruction is inactive and there are no limitations for password legitimation relatively small. This procedure will give you the best possible protection from unauthorized access.

If there has been an accidental lockout, you can skip the call in the startup OB (by querying an input parameter, for example) and you have the set time (for example, 10 seconds up to 1 minute) to establish a connection to the CPU before the lock becomes active once again.

If there is no timer in your program code and a lockout has occurred, insert an empty transfer card or an empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then download the user program once again from STEP 7 to the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

WARNING

Inserting an empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Once you have pulled the transfer card, the content of the transfer card is available in the internal load memory. Make sure that the card does not include a program at this point.

WARNING
Inserting an empty program card When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property. Make sure that the program card is empty. The internal load memory is copied to the empty program card. Once you have pulled the previously empty program card, the internal load memory is empty.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200 CPU: The lock is disabled and the instruction must be called once again in the program (e.g., in the startup OB). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Parameters

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed

Parameters	Declaration	Data type	Memory area	Description
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported
80D0	The password for fail-safe is not configured. The signal state must be TRUE for standard CPUs.
80D1	The read/write access is not configured
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

STL Basics (Page 7245)

Declaring global constants (Page 7088)

PC systems (S7-1500)

SHUT_DOWN: Shut down target system (S7-1500)

Description

You can use the instruction "SHUT_DOWN: Shutdown target system" to shut down the PC-based automation system and restart the S7 software controller CPU 150xS or Windows on the PC-based automation system.

The instruction is available in the "Instructions" task card under Basic instructions > Program control > Runtime control.

A restart is, for example, advisable in the following situations:

- An industrial UPS (uninterruptible power supply) reports a power failure via a digital input.
- Windows stops responding or shows a "blue screen".
- Too many error OBs are called in the user program.

Parameter

The following table shows the parameters of the "Shutdown target system" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	MODE = 1: Shut down CPU 150xS and Windows The CPU goes to STOP and saves the retentive data. Then the CPU and Windows are shut down. The system must be restarted manually. MODE = 2: Restart CPU 150xS The CPU goes to STOP and saves the retentive data. Then the CPU is shut down and restarted. MODE = 3: Restart Windows The CPU remains in RUN. Windows is restarted (as of TIA Portal V14, MODE 3 is only retained for backward compatibility. We recommend using MODE 4 or MODE 5). MODE = 4: Windows is shut down and restarted correctly. The CPU remains in RUN. MODE = 5: Restart Windows (Comparable with MODE 3; Exception: MODE 5 should only be used if Windows has crashed).
COM-MENT	Input	STRING	I, Q, M, D, L	With Mode = 1, 3 and 4, you can specify a reason for the restart. The reason is output in the Windows Event Log.
Ret_Val	Return	WORD	Q, M, D, L	Ret_Val = 0: No faults Ret_Val = 8090: The value passed to MODE is not supported. Ret_Val = 8091: Windows does not respond to the call of the Shut_Down instruction (only valid for Mode 3 and 4). Ret_Val = 8092: Please contact SIMATIC Customer Support if this error occurs (only valid for Mode 3 and 4).

RE_TRIGR: Restart cycle monitoring time (S7-1500)

Description

The "Restart cycle monitoring time" instruction is used to restart the cycle monitoring time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

If the instruction is called in a block with a higher priority, such as a hardware interrupt or diagnostic error interrupt, the instruction is not executed and the enable output ENO returns signal state "0".

The instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Calling the instruction

You can call the instruction regardless of the priority in all organization blocks.

Parameters

The "Restart cycle monitoring time" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

STP: Exit program (S7-1500)

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Parameters

The "Exit program" instruction has no parameters and returns no error information.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

GET_ERROR: Get error locally (S7-1500)

Description

The "Get error locally" instruction is used to query the occurrence of errors within a program block. This normally involves programming or access errors. If the system reports an error during the execution of the program block, detailed information is stored in the operand at the OUT output for the first error to occur during the execution of the block since the last execution of the instruction.

Only operands of the "ErrorStruct" system data type can be specified at the OUT output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the program block, the error information about the next error to occur is output by the instruction only after the first error that occurred has been remedied.

Note

Output OUT

The OUT output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.

Note

Activation of the local error handling

As soon as you insert the instruction in the program code of a program block, the local error handling is activated and default system reactions are ignored when errors occur.

Options for error handling

You can find an overview of the options for error handling here: Overview of mechanisms for error handling (Page 138)

For a detailed example of local error handling that contains several error handling options, refer to: Example of how to handle program execution errors (Page 167)

Parameters

The following table shows the parameters of the "Get error locally" instruction:

Parameters	Declaration	Data type	Memory area	Description
OUT	Output	ErrorStruct	D, L	Error information

Data type "ErrorStruct"

The "ErrorStruct" data type can be inserted in a global data block or in the block interface. You can insert the data type more then once if you assign a different name for the data structure each time. The data structure and the name of individual structure elements cannot be changed. If you save the error information in a global data block, it can also be read out by other program blocks.

The following table shows the structure of the "ErrorStruct" data type:

Structure component		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows whether the error occurred during a program block call. 16#01: Error during a program block call 16#00: No error during a program block call
REACTION		BYTE	Default reaction: 0: Ignore (write error) 1: Continue with substitute value "0" (read error) 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of the program block
	BLOCK_TYPE	BYTE	Program block type in which the error occurred: 1: Organization block (OB) 2: Function (FC) 3: Function block (FB)
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand
	AREA	BYTE	(C) Memory area: L: 16#40...16#7F, 16#86, 16#87, 16#8E, 16#8F, 16#C0...16#FF I: 16#81 Q: 16#82 M: 16#83 DB: 16#40, 16#84, 16#85, 16#8A, 16#8B PI: 16#01 PQ: 16#02 Technological objects: 16#04

Structure component		Data type	Description
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

The following table shows the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid reference used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:

```
STL                          Explanation
LABEL: L #Field[#index]      // Access to the #Field[#index] tag
L 40.5                       // Load value 40.5
*R                           // Multiply value of the tag by the value 40.5.
T #TagOut                    // Transfer product to the operand #TagOut.

CALL GET_ERROR               // The instruction is called.
OUT := #Error                // If an error occurs, detailed information is stored in
                             the #Error operand.

T #Error.REACTION            // The structure component #Error.REACTION is read.
L 1                          // Load value 1
==I                          // The value of the structure component is compared with
                             the value 1.
JC LABEL                     // If the comparison of the two values is positive, the
                             program execution returns to the jump label LABEL.
```

An error occurred accessing the #Field[#index] tag. The #TagOut operand returns the signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, we recommend that you program the "Get error locally" instruction after the multiplication to get the error. The error information supplied by the "Get error locally" instruction is evaluated with a comparison. If the #Error.REACTION structure component has the value "1", there is a read/access error and the program execution begins again at the jump label LABEL.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

STL Basics (Page 7245)

GET_ERR_ID: Get error ID locally (S7-1500)

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the error ID of the first error that occurred in the tag is stored at the RET_VAL output.

Only operands of the WORD data type can be specified at the RET_VAL output. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The RET_VAL output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.

The output of the "Get error ID locally" instruction is only set if error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Options for error handling

You can find an overview of the options for error handling here: Overview of mechanisms for error handling (Page 138)

For a detailed example of local error handling that contains several error handling options, refer to: Example of how to handle program execution errors (Page 167)

Parameters

The following table shows the parameters of the "Get error ID locally" instruction:

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Output	WORD	I, Q, M, D, L	Error ID

RET_VAL parameter

The following table shows the values that can be output at the RET_VAL parameter:

RET_VAL* (Hexadecimal)	RET_VAL* (Decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range

RET_VAL* (Hexadecimal)	RET_VAL* (Decimal)	Description
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid reference used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:

```
STL                          Explanation
L #Field[#index]             // Access to the #Field[#index] tag
L 40.5                       // Load value 40.5
*R                           // Multiply value of the tag by the value 40.5.
T #TagOut                    // Transfer product to the operand #TagOut.

CALL GET_ERR_ID              // The instruction is called.
RET_VAL := #TagID            // If an error occurs, an error ID is stored in the #Tag-
                             ID operand.

T #TagID                     // The error ID at the #TagID operand is read.
```

```
STL                          Explanation
L 16#2522                    // Load value 16#2522
==I                          // The value of the error ID is compared with the value
                             of 16#2522.

CALL MOVE                    // The instruction is called.
IN := 100.0                  // The value 100.0 is copied to the #TagOut operand.
OUT := #TagOut               // Output of the instruction
```

An error occurred accessing the #Field[#index] tag. The #TagOut operand returns the signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, we recommend that you program the "Get error ID locally" instruction after the multiplication to get the error. The error ID supplied by the "Get error ID locally" instruction is evaluated with a comparison. If the #TagID operand returns ID 16#2522, there is a read/access error and the value "100.0" is moved to the #TagOut output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

STL Basics (Page 7245)

INIT_RD: Initialize all retain data (S7-1500)

Description

The "Initialize all retain data" instruction is used to reset the retentive data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution exceeds the program cycle duration.

Parameters

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	If the input "REQ" has the signal state "1", all retentive data are reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL INIT_RD                 // The instruction is called.
REQ := "Tag_REQ"             // If the operand "Tag_REQ" has the signal state "1",
                             all retentive data are reset.
RET_VAL := "Tag_RET_VAL"     // Error code
```

If the "Tag_REQ" operand has signal state "1", the instruction is executed. All retentive data of all data blocks, bit memories and SIMATIC timers and counters are reset.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1441)

Evaluating errors with output parameter RET_VAL (Page 162)

STL Basics (Page 7245)

WAIT: Configure time delay (S7-1500)

Description

You can use the "Configure time delay" instruction to pause the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to 32767 microseconds (µs). The smallest possible delay time depends on the CPU and corresponds to the execution time of the instruction.

The execution of the instruction can be interrupted by higher priority events and does not return any error information.

Note

Negative delay time

If you specify a negative delay time at parameter WT, the enable output ENO, or the RLO and the BR bit, will return the signal state FALSE. A negative delay time does not affect the CPU. The following instructions linked to enable output ENO are not executed in LAD or FBD.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

Example of the influence of the planned delay time

In the following example, you will see the influence of the delay time of the instruction "WAIT" in various scenarios.

The following figure is a schematic representation of the scenarios:

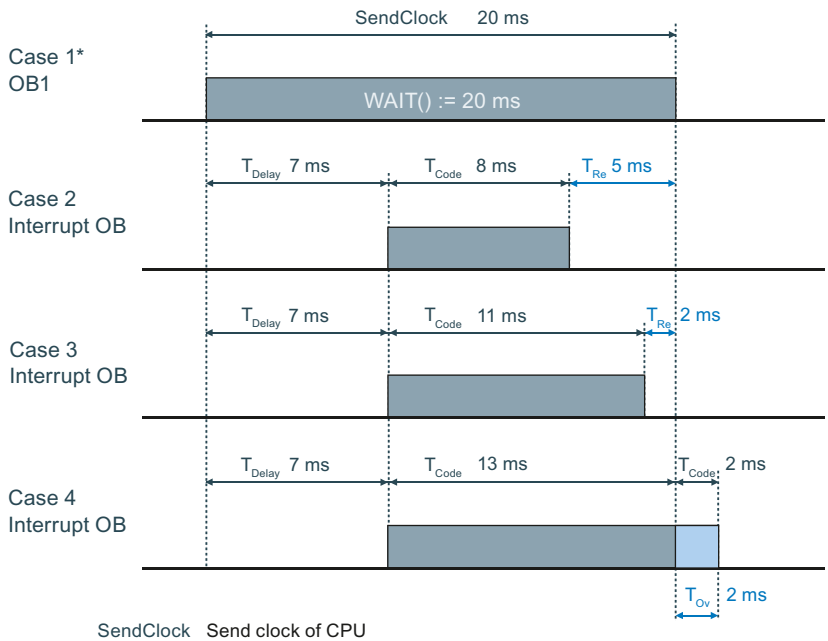

SendClock	Send clock of CPU
T_{Delay}	Delay time of the interrupt OB
T_{Code}	Time of program execution
T_{Re}	Remaining time
T_{Ov}	Excess time by which the send clock is increased

* The OB1 with the instruction "Wait" is also part of cases 2, 3 and 4 and needs to be taken into account.

Remaining time = The period between the end of the planned delay time (via "WAIT") until the end of the interrupt OB

Excess time = The period between the end of the interrupt OB and the end of the planned delay time (via "WAIT").

Case 1:

The "WAIT" instruction is called in an OB1. The "WAIT" instruction can be interrupted by higher-priority OBs or higher-priority processes (e.g. System Threads). However, the delay time of the "WAIT" instruction is neither changed nor deferred.

Cases 2 and 3:

Processing of the program in OB1 is resumed after a time delay of 20 ms. This time delay is calculated by calling the "WAIT" instruction in OB1 (see OB1 with WAIT). Within these 20 ms, an interrupt OB can run its own program code. The send clock of the CPU is not changed.

Case 4:

Processing of the program in OB1 is resumed after the higher-priority process has ended. The 20 ms delay time in OB1 has elapsed, but the higher-priority process still needs to be ended. The send clock of the CPU is increased.

Note

Order of execution of system or communication processes (System Threads)

The System Threads usually use the priority "15". There are also System Threads with a higher priority than "26", but these processes cause a lower CPU load. System Threads are not displayed in the figure.

Runtime measurement of the OB1 using the "RT_INFO" instruction:

Case 2: 20 ms - 8 ms - System Threads = <12 ms. Send clock: ~20 ms.

Case 3: 20 ms - 11 ms - System Threads - <9 ms. Send clock: ~20 ms.

Case 4: 20 ms - 15 ms - System Threads - <7 ms. Send clock: ~22 ms.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

RUNTIME: Measure program runtime (S7-1500)

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and write the value to the IN-OUT parameter MEM. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the IN-OUT parameter MEM as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the

command sequence according to the internal counter frequency and writes it to the output RET_VAL.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Parameters

The following table lists the parameters of the "Measure program runtime" instruction:

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Output	LREAL	I, Q, M, D, L	Returns the measured runtime in seconds
MEM	InOut	LREAL	I, Q, M, D, L	The content is for internal purposes only.

For additional information on valid data types, refer to "See also".

Example

The following example shows the how the instruction works based on the runtime calculation of a program block:

```
STL                              Explanation
CALL RUNTIME                     // The instruction is called.
   RET_VAL := "TagResult"        // Intermediate result
   MEM := "TagMemory"            // Saves the starting point of the runtime measurement.

CALL "Best_before_date",         // The "Best_before_date" block is called and pro-
"Best_before_date_DB"            cessed.

CALL RUNTIME                     // The instruction is called a second time.
   RET_VAL := "TagResult"        // Result of the instruction
   MEM := "TagMemory"            // Uses the start point of the runtime measurement as
                                 basis for the runtime measurement.
```

The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called.

When the program block FB1 has been processed, the instruction is executed a second time. The second call of the instruction calculates the runtime of the program block and writes the result to the "TagResult" operand on the RET_VAL output.

You can find a more detailed example of how you can measure the total cycle time of a program in the Siemens Industry Online Support under the following entry ID: 87668055 (https://support.industry.siemens.com/cs/ww/en/view/87668055)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

Word logic operations (S7-1500)

DECO: Decode (S7-1500)

Description

The instruction "Decode" sets a bit specified by the input value in the output value.

The "Decode" instruction reads the value of the IN parameter and sets the bit in the OUT parameter whose bit position corresponds to the value read. The other bits in the output value are filled with zeroes. When the value at the IN parameter is greater than 31, a modulo-32 operation is executed.

Parameters

The following table shows the parameters of the "Decode" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	UINT	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set
OUT	Output	Bit strings	I, Q, M, D, L, P	Output value

You can select the data type of the OUT parameter from the "???" drop-down list.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL DECO                    // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
IN := "Tag_Input"            // Input value that specifies the bit to be set in the
                             output value
OUT := "Tag_Output"          // Result
```

The following table shows how the instruction functions using specific values:

Tag_Input	3	
	31 16	15 ... 3 ... 0
Tag_Output	0000 0000 0000 0000	0000 0000 0000 1000

The instruction reads bit number "3" from the value of the operand "Tag_Input" and sets the third bit to the value of the operand "Tag_Output".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

ENCO: Encode (S7-1500)

Description

The instruction "Encode" is used to read the bit number of the least significant set bit in the input value and to output it at the parameter OUT.

The "Encode" instruction selects the least significant bit of the value at the IN parameter and writes this bit number to the operand at the OUT parameter. If the IN parameter has the value DW#16#00000001 or DW#16#00000000, the value "0" is output at the OUT parameter.

Parameters

The following table shows the parameters of the "Encode" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

You can select the data type of the IN parameter from the "???" drop-down list.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL ENCO                    // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
IN := "Tag_Input"            // Input value the least significant bit of which is
                             read.
OUT := "Tag_Output"          // Bit number of least significant bit set in the in-
                             put value
```

The following table shows how the instruction functions using specific values:

	31 16	15 ... 3 ... 0
“Tag_Input“	0000 1111 0000 0101	0000 1001 0000 1000
“Tag_Output“	3	

The instruction selects the least significant set bit of the variable "Tag_Input" and writes its bit position "3" to the variable "Tag_Output".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

SEL: Select (S7-1500)

Description

Depending on a switch (G input), the "Select" instruction selects one of the inputs, IN0 or IN1 and copies its content to the OUT output. When the G input has signal state "0", the value at the IN0 input is moved. If input G has signal state "1", the value at input IN1 is copied to output OUT.

The instruction is only executed if the tags of all parameters are of the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

Parameters	Declaration	Data type	Memory area	Description
G	Input	BOOL	I, Q, M, D, L or constant	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P or constant	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	Result

You can select the data type of the parameters INn and OUT from the "???" drop-down list.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SEL                     // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
G := "Tag_Input_G"           // Signal state of the switch
IN0 := "Tag_Input0"          // First input value
IN1 := "Tag_Input1"          // Second input value
OUT := "Tag_Output"          // Value of the selected input
```

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value
G	Tag_Input_G	1
IN0	Tag_Input0	W#16#0000
IN1	Tag_Input1	W#16#FFFF
OUT	Tag_Output	W#16#FFFF

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

Legacy (S7-1500)

DRUM: Implement sequencer (S7-1500)

Description

You use the "Implement sequencer" instruction to assign the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD) with the programmed values of the OUT_VAL parameter of the relevant step. The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the

current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). When a bit of the configurable mask has a signal state of "1", the OUT_VAL value sets/resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L or constant	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Maximum step number (for example: LST_STEP = 16#08; a maximum of 8 steps is possible.)
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L or constant	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.

Parameters	Declaration	Data type	Memory area	Description
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L or constant	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Preset first step of the sequencer (1 to 16)
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P or constant	Remaining processing time for the current step
DTBP	Static	WORD	I, Q, M, D, L, P or constant	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L or constant	System time of the previous call
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L or constant	Preset counter value for each step [1 to 16] where one clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

STL	Explanation
CALL DRUM, "DRUM_DB"	// The "Implement sequencer" instruction is called and the instance data block "DRUM_DB" is created.
RESET := "Tag_Reset"	// Reset input
JOG := "Tag_Input_Jog"	// On a positive signal edge, the instruction advances to the next step.
DRUM_EN := "Tag_Input_DrumEN"	// The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP := "Tag_Number_Last-Step"	// Number of the last programmed step
EVENT1 := "MyTag_Event_1"	// Event bit 1
EVENT2 := "MyTag_Event_2"	// Event bit 2
EVENT3 := "MyTag_Event_3"	// Event bit 3
EVENT4 := "MyTag_Event_4"	// Event bit 4
EVENT5 := "MyTag_Event_5"	// Event bit 5
EVENT6 := "MyTag_Event_6"	// Event bit 6
EVENT7 := "MyTag_Event_7"	// Event bit 7
EVENT8 := "MyTag_Event_8"	// Event bit 8
EVENT9 := "MyTag_Event_9"	// Event bit 9
EVENT10 := "MyTag_Event_10"	// Event bit 10
EVENT11 := "MyTag_Event_11"	// Event bit 11
EVENT12 := "MyTag_Event_12"	// Event bit 12
EVENT13 := "MyTag_Event_13"	// Event bit 13
EVENT14 := "MyTag_Event_14"	// Event bit 14
EVENT15 := "MyTag_Event_15"	// Event bit 15
EVENT16 := "MyTag_Event_16"	// Event bit 16
OUT1 := "MyTag_Output_1"	// Output bit 1
OUT2 := "MyTag_Output_2"	// Output bit 2
OUT3 := "MyTag_Output_3"	// Output bit 3
OUT4 := "MyTag_Output_4"	// Output bit 4
OUT5 := "MyTag_Output_5"	// Output bit 5
OUT6 := "MyTag_Output_6"	// Output bit 6
OUT7 := "MyTag_Output_7"	// Output bit 7
OUT8 := "MyTag_Output_8"	// Output bit 8
OUT9 := "MyTag_Output_9"	// Output bit 9
OUT10 := "MyTag_Output_10"	// Output bit 10

```
STL                              Explanation
OUT11 := "MyTag_Output_11"       // Output bit 11
OUT12 := "MyTag_Output_12"       // Output bit 12
OUT13 := "MyTag_Output_13"       // Output bit 13
OUT14 := "MyTag_Output_14"       // Output bit 14
OUT15 := "MyTag_Output_15"       // Output bit 15
OUT16 := "MyTag_Output_16"       // Output bit 16
Q := "Tag_Output_Q"              // The signal state "1" indicates that the time for
                                 the last step has elapsed
OUT_WORD := "Tag_OutputWord"     // Word address to which the sequencer writes the out-
                                 put values
ERR_CODE := "Tag_ErrorCode"      // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE

Parameters	Address	Value
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE

Parameters	Address	Value
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameters	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE

Parameters	Operand	Address	Value
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

STL Basics (Page 7245)

DCAT: Discrete control-timer alarm (S7-1500)

Description

You can use the "Discrete control-timer alarm" to count the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

The "Discrete control-timer alarm" instruction has the following effects on the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET is < PT) OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".

- When the signal state of the CMD and CMD_HIS parameters is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state at the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- When the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameters	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L or constant	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L or constant	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms

Parameters	Declaration	Data type	Memory area	Description
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
CMD_HIS	Static	BOOL	D, L or constant	CMD history bit

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The CMD_HIS parameter of the instance data block is set to the signal state "1" and the ET parameter is reset to "0".

Note

You can initialize static parameters in the data block.

```
STL                          Explanation
CALL DCAT, "DCAT_DB"         // The "Discrete control-timer alarm" instruction is
                             called and the instance data block "DCAT_DB" is cre-
                             ated.
CMD := "Tag_Input_CMD"       // Opening or closing of device
O_FB := "Tag_Input_O_FB"     // Feedback when opening
C_FB := "Tag_Input_C_FB"     // Feedback when closing
Q := "Tag_Output_Q"          // Shows the status of the parameter CMD
OA := "Tag_Output_OA"        // Alarm output when opening
CA := "Tag_Output_CA"        // Alarm output when closing
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

MCAT: Motor control-timer alarm (S7-1500)

Description

The "Motor control-timer alarm" instruction is used to accumulate the time from the point at which one of the command inputs (opening or closing) is switched on. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameters	Declaration	Data type	Memory space	Description
O_CMD	Input	BOOL	I, Q, M, D, L or constant	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L or constant	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L or constant	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing

Parameters	Declaration	Data type	Memory space	Description
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition
ET	Static	DINT	D, L or constant	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
O_HIS	Static	BOOL	D, L or constant	"Open" history bit
C_HIS	Static	BOOL	D, L or constant	"Close" history bit

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT

Input parameters		Output parameters
>=PT	ET >= PT	
If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore not possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case: OO = FALSE CO = FALSE OA = FALSE CA = FALSE ET = PT Q = TRUE		

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL MCAT, "MCAT_DB"             // The "Motor control-timer alarm" instruction is
                                 called and the instance data block "MCAT_DB" is cre-
                                 ated.
O_CMD := "Tag_Input_O_CMD"       // "Open" command input
C_CMD := "Tag_Input_C_CMD"       // "Close" command input
S_CMD := "Tag_Input_S_CMD"       // "Stop" command input
O_FB := "Tag_Input_O_FB"         // Feedback when opening
C_FB := "Tag_Input_C_FB"         // Feedback when closing
OO := "Tag_OutputOpen"           // "Open" output
CO := "Tag_OutputClosed"         // "Close" output
OA := "Tag_Output_OA"            // Alarm output when opening
CA := "Tag_Output_CA"            // Alarm output when closing
Q := "Tag_Output_Q"              // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE

Parameters	Operand	Value
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

IMC: Compare input bits with the bits of a mask (S7-1500)

Description

You can use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.

Parameters	Declaration	Data type	Memory area	Description
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example all 16 input bits are compared with the mask for step 2. The signal state of the OUT parameter is set to TRUE, since the input bits match the mask for step 2.

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL IMC, "IMC_DB"               // The instruction "Compare input bits with the bits
                                 of a mask" is called and the instance data block
                                 "IMC_DB" is created.
IN_BIT0 := "Tag_Input_BIT0"      // Input bit 0
IN_BIT1 := "Tag_Input_BIT1"      // Input bit 1
IN_BIT2 := "Tag_Input_BIT2"      // Input bit 2
IN_BIT3 := "Tag_Input_BIT3"      // Input bit 3
IN_BIT4 := "Tag_Input_BIT4"      // Input bit 4
IN_BIT5 := "Tag_Input_BIT5"      // Input bit 5
IN_BIT6 := "Tag_Input_BIT6"      // Input bit 6
IN_BIT7 := "Tag_Input_BIT7"      // Input bit 7
IN_BIT8 := "Tag_Input_BIT8"      // Input bit 8
IN_BIT9 := "Tag_Input_BIT9"      // Input bit 9
IN_BIT10 := "Tag_Input_BIT10"    // Input bit 10
IN_BIT11 := "Tag_Input_BIT11"    // Input bit 11
IN_BIT12 := "Tag_Input_BIT12"    // Input bit 12
IN_BIT13 := "Tag_Input_BIT13"    // Input bit 13
IN_BIT14 := "Tag_Input_BIT14"    // Input bit 14
IN_BIT15 := "Tag_Input_BIT15"    // Input bit 15
CMP_STEP := "Tag_CMP_STEP"       // Step number of the mask
OUT := "Tag_Output"              // Indicates whether or not a match was found.
ERR_CODE := "Tag_ErrorCode"      // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE

Parameters	Operand	Value
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
CMP_STEP	Tag_CMP_STEP	B#16#02
OUT	Tag_Output	FALSE
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the instance data block "IMC_DB" of the instruction:

Parameters	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output	TRUE
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)
Basic information on the status word (Page 186)
Switching display formats in the program status (Page 7698)
STL Basics (Page 7245)

SMC: Compare scan matrix (S7-1500)

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.

Parameters	Declaration	Data type	Memory area	Description
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.

Parameters	Declaration	Data type	Memory area	Description
LAST	Static	BYTE	I, Q, M, D, L, P or constant	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In this example, all 16 input bits are compared with the mask for steps 0 through 5 until a match is found. Only the masks for steps 0 through 2 are scanned, since the mask for step 2 matches the input bit.

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL SMC, "SMC_DB"               // The "Compare scan matrix" instruction is called and
                                 the instance data block "SMC_DB" is created.
IN_BIT0 := "Tag_Input_BIT0"      // Input bit 0
IN_BIT1 := "Tag_Input_BIT1"      // Input bit 1
IN_BIT2 := "Tag_Input_BIT2"      // Input bit 2
IN_BIT3 := "Tag_Input_BIT3"      // Input bit 3
IN_BIT4 := "Tag_Input_BIT4"      // Input bit 4
IN_BIT5 := "Tag_Input_BIT5"      // Input bit 5
IN_BIT6 := "Tag_Input_BIT6"      // Input bit 6
IN_BIT7 := "Tag_Input_BIT7"      // Input bit 7
```

```
STL                              Explanation
IN_BIT8 := "Tag_Input_BIT8"    // Input bit 8
IN_BIT9 := "Tag_Input_BIT9"    // Input bit 9
IN_BIT10 := "Tag_Input_BIT10"  // Input bit 10
IN_BIT11 := "Tag_Input_BIT11"  // Input bit 11
IN_BIT12 := "Tag_Input_BIT12"  // Input bit 12
IN_BIT13 := "Tag_Input_BIT13"  // Input bit 13
IN_BIT14 := "Tag_Input_BIT14"  // Input bit 14
IN_BIT15 := "Tag_Input_BIT15"  // Input bit 15
OUT := "Tag_Output"            // Indicates whether or not a match was found.
OUT_STEP := "Tag_Output_STEP"  // Contains the step number with the relevant mask.
ERR_CODE := "Tag_ErrorCode"    // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
OUT	Tag_Output	FALSE
OUT_STEP	Tag_Output_STEP	B#16#00
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the instance data block "SMC_DB" of the instruction:

Parameters	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE

Parameters	Address	Value
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE
LAST	DB84	B#16#05

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output	TRUE
OUT_STEP	Tag_Output_STEP	B#16#02
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

STL Basics (Page 7245)

LEAD_LAG: Lead and lag algorithm (S7-1500)

Description

You can use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the instruction "Lead and lag algorithm" in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL LEAD_LAG, "LEAD_LAG_DB"     // The "Lead and lag algorithm" instruction is called
                                 and the instance data block "LEAD_LAG_DB" is created.
IN := "Tag_Input"                // The input value of the current sample time (cycle
                                 time) to be processed.
SAMPLE_T := "Tag_In-             // Sample time
put_SAMPE_T"
OUT := "Tag_Output_Result"       // Result of the instruction
ERR_CODE := "Tag_ErrorCode"      // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameters	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameters	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameters	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

STL Basics (Page 7245)

SEG: Create bit pattern for seven-segment display (S7-1500)

Description

The "Create bit pattern for seven-segment display" instruction converts each of the four hexadecimal digits in the specified source word (IN) into four equivalent codes for a 7-segment display and writes these to the double word of the output (OUT).

The "Create bit pattern for seven-segment display" does not recognize any error conditions.

The relationship between the input hexadecimal values and the output bit patterns is as follows:

Digit	-gfedcba	Display	
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	Seven-segment display

0011	01001111	3
0100	01100110	4
0101	01101101	5
0110	01111101	6
0111	00000111	7
1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	B
1100	00111001	C
1101	01011110	D
1110	01111001	E
1111	01110001	F

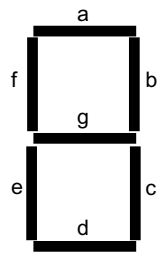

Parameters

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	IN	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits.
OUT	OUT	DWORD	I, Q, M, D, L, P	Destination bit pattern with four bytes.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`CALL SEG`	`// The "Create bit pattern for seven-segment display" instruction is called.`
`IN := "Tag_Input"`	`// Source word`
`OUT := "Tag_Output"`	`// Bit pattern for the seven-segment display`

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16#065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

BCDCPL: Create tens complement (S7-1500)

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL BCDCPL                  // The instruction is called.
IN := "Tag_Input"            // 7-digit BCD number
ERR_CODE := "Tag_Output"     // Tens complement of the specified BCD number
```

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

STL Basics (Page 7245)

BITSUM: Count number of set bits (S7-1500)

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output at the RET_VAL parameter.

Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL BITSUM                  // The instruction is called.
IN := "Tag_Input"            // Operand whose set bits are counted.
RET_VAL := "Tag_Output"      // Number of set bits of the operand "Tag_Input".
```

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

STL Basics (Page 7245)

STL Mnemonic (S7-1500)

Bit logic operations (S7-1500)

A: AND logic operation (S7-1500)

Description

Use the instruction "AND logic operation" to query whether the signal state of a binary operand is "1" and to AND the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand.

After execution of the instruction the RLO will be "1" if both linked signal states are also "1". If one of the linked signal states is "0", the RLO is "0" after executing the instruction.

If you use the instruction multiple times consecutively, all query results must be "1" for the common result of the logic operation to also be "1".

You can also use the instruction "AND logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "AND logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
A “Tag_Input_1“	Signal state 1: Normally open contact
A “Tag_Input_2“	Signal state 1: Normally open contact
= “Tag_Output“	Signal state 1: Coil
Shows closed switch	

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

STL programming examples (Page 7280)

AN: Negated AND logic operation (S7-1500)

Description

Use the instruction "Negated AND logic operation" to query whether the signal state of a binary operand is "0" and to AND the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. If the operand has signal state "0," the result of the query is "1". If the operand has signal state "1," the result of the query is "0".

After the instruction executes, the RLO is "1" if the result of the query and the RLO before the execution of the instruction have signal state "1".

After the instruction executes, the RLO is "0" if the result of the query or the RLO before the execution of the instruction have signal state "0".

You can also use the instruction "Negated AND logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "Negated AND logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program		Relay circuit diagram	
		Power rail	
A	“Tag_Input_1“	Signal state 0:	Normally open contact
AN	“Tag_Input_2“	Signal state 1:	Normally closed contact
=	“Tag_Output“	Signal state 0:	Coil

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

O: OR logic operation (S7-1500)

Description

Use the instruction "OR logic operation" to query whether the signal state of a binary operand is "1" and to OR the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand.

After the instruction executes, the RLO is "1" if one of the linked signal states is "1". If both linked signal states are "0", the RLO is "0" after the instruction executes.

If you use the instruction multiple times consecutively, it is sufficient that one of the results of the query is "1" for the common result of the logic operation to also be "1".

You can also use the instruction "OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
O “Tag_Input_1“	Signal state 1 normally open contact (“Tag_Input_1“) / Signal state 0 normally open contact (“Tag_Input_2“)
O “Tag_Input_2“	
= “Tag_Output“	Signal state 1: Coil
	Shows closed switch

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

ON: Negated OR logic operation (S7-1500)

Description

Use the instruction "Negated OR logic operation" to query whether the signal state of a binary operand is "0" and to OR the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. If the operand has signal state "0," the result of the query is "1". If the operand has signal state "1," the result of the query is "0".

After the instruction executes, the RLO is "1" if the result of the query or the RLO before the execution of the instruction have signal state "1". If both linked signal states are "0", the RLO is also "0" after the instruction executes.

If you use the instruction multiple times consecutively, it is sufficient that one of the results of the query is "1" for the common result of the logic operation to also be "1".

You can also use the instruction "Negated OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "Negated OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is checked

Example

The following example shows how the instruction works:

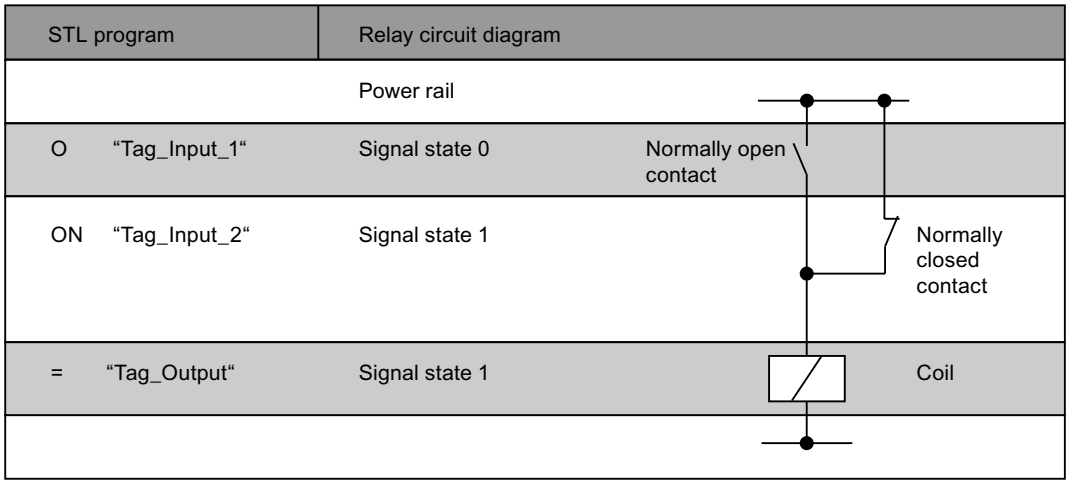

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

X: EXCLUSIVE OR logic operation (S7-1500)

Description

Use the instruction "EXCLUSIVE OR logic operation" to query whether the signal state of a binary operand is "1" and to XOR the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand. After the instruction EXECUTES, the RLO is "1" if the linked signal states are not equal. The RLO is "0" if both signal states are the same.

If, for example, the RLO before execution of the instruction is "1" and the queried operand is "1", the RLO is reset to "0". If the RLO is "1" and the operand has signal state "0", the RLO is set to "1" after the instruction executes.

If you use the instruction several times consecutively, it is sufficient that one of the query results is "1" for the common RLO to be "1" as well. If all query results are "1" or "0" the RLO is reset to "0".

You can also use the instruction "EXCLUSIVE OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
X "Tag_Input_1"	Contact
X "Tag_Input_2"	Contact
= "Tag_Output"	Coil

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

XN: Negated EXCLUSIVE OR logic operation (S7-1500)

Description

Use the "Negated EXCLUSIVE OR logic operation" instruction to query whether the signal state of a binary operand is "0" and to OR the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. After the instruction EXECUTES, the RLO is "1" if the linked signal states are not equal. The RLO is "0" if both signal states are the same.

If, for example, the queried operand has signal state "0," the result of the query is "1". XORing the query result with an RLO "1" results in signal state "0". The logic operation with an RLO "0" results in signal state "1".

You can also use the "Negated EXCLUSIVE OR logic operation" instruction to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "Negated EXCLUSIVE OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program		Relay circuit diagram
		Power rail
X	“Tag_Input_1“	Contact
XN	“Tag_Input_2“	Contact
=	“Tag_Output“	Coil

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

O: OR logic operation of AND functions (S7-1500)

Description

Use the instruction "OR logic operation of AND functions" to execute several AND logic operations before an OR logic operation. The overall result of the AND logic operations is saved temporarily and ORed with further signal queries.

Example

The following example shows how the instruction works:

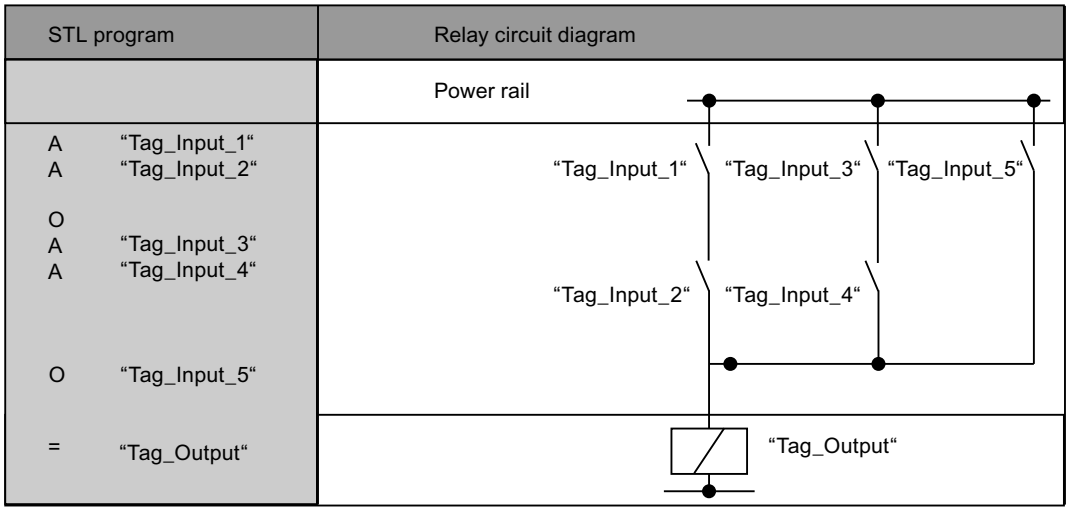

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

A(: AND logic operation with branch (S7-1500)

Description

Use the instruction "AND logic operation with branch" to execute the instructions of a parenthesized expression before an AND logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. The saved RLO is ANDed with the result of logic operation of the complete nested expression. If additional signal queries follow after the nested expression, these are also ANDed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

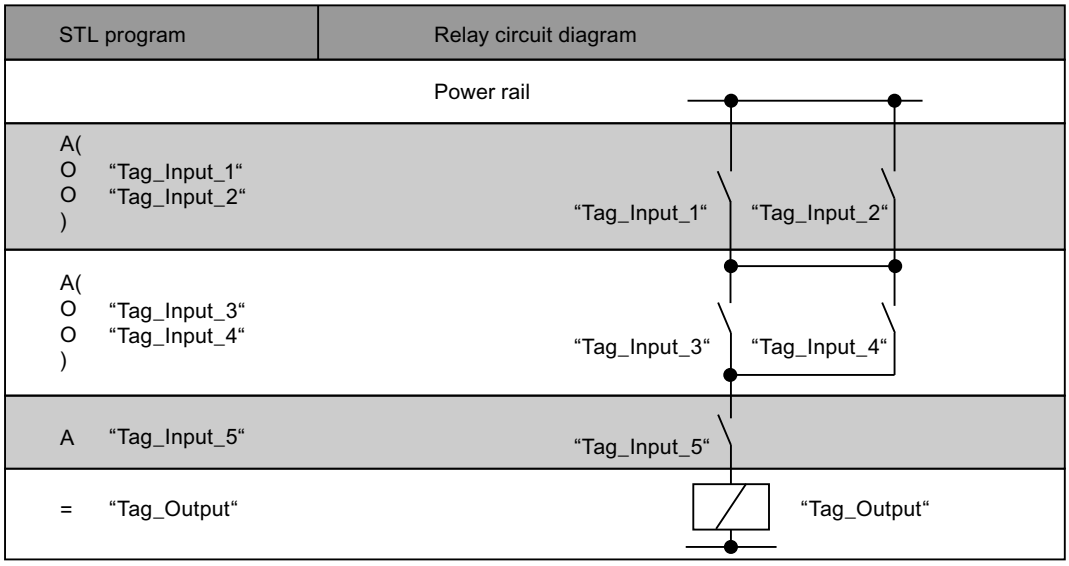

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

AN(: Negated AND logic operation with branch (S7-1500)

Description

Use the "Negated AND logic operation with branch" instruction to execute the instructions of a parenthesized expression before an AND logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. After the instructions in the parenthesized expression have executed, the RLO of the complete parenthesized expression is negated and ANDed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also ANDed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

STL	Explanation
AN(	// Start of the nested expression
	// AND the RLO of the parenthesized expression and negate the complete result
O "Tag_Input_1"	// Query the signal state of the operand for "1" and OR with the current RLO.
O "Tag_Input_2"	// Query the signal state of the operand for "1" and OR with the current RLO.
)	// End of the nested expression
AN "Tag_Input_3"	// Query the signal state of the operand for "0" and AND with the current RLO.
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

O(: OR logic operation with branch (S7-1500)

Description

Use the instruction "OR logic operation with branch" to execute the instructions of a parenthesized expression before an OR logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. After the instructions contained in the parenthesized expression have executed, the saved RLO is ORed with the RLO of the complete parenthesized expression. If additional signal queries follow after the parenthesized expression, these are also ORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
O(	// Start of the nested expression
	// OR the RLO of the nested expression

STL	Explanation
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

ON(: Negated OR logic operation with branch (S7-1500)

Description

Use the "Negated OR logic operation with branch" instruction to execute the instructions of a parenthesized expression before an OR logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. After the instructions in the parenthesized expression have executed, the RLO of the complete parenthesized expression is negated and ORed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also ORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
ON(	// Start of the nested expression
	// Link RLO of the nested expression by OR logic and negate the complete result.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

X(: EXCLUSIVE OR logic operation with branch (S7-1500)

Description

Use the instruction "EXCLUSIVE OR logic operation with branch" to execute the instructions of a parenthesized expression before an EXCLUSIVE OR logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. After the instructions in the parenthesized expression have executed, the saved RLO is XORed with the RLO of the complete parenthesized expression. If additional signal queries follow after the parenthesized expression, these are also XORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

STL	Explanation
X(	// Start of the nested expression
	// Link RLO of the nested expression by EXCLUSIVE OR logic
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
X "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

XN(: Negated EXCLUSIVE OR logic operation with branch (S7-1500)

Description

Use the "Negated EXCLUSIVE OR logic operation with branch" instruction to execute the instructions of a parenthesized expression before an EXCLUSIVE OR logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. After the instructions contained in the parenthesized expression have been executed, the RLO of the complete parenthesized expression will be negated and EXCLUSIVE ORed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also XORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

STL	Explanation
`XN(`	`// Start of the nested expression`
	`// Link RLO of the nested expression by EXCLUSIVE OR logic and negate the complete result`
`A "Tag_Input_1"`	`// Query the signal state of the operand for "1" and AND with the current RLO.`
`A "Tag_Input_2"`	`// Query the signal state of the operand for "1" and AND with the current RLO.`
`)`	`// End of the nested expression`
`XN "Tag_Input_3"`	`// Query the signal state of the operand for "1" and AND with the current RLO.`
`= "Tag_Output"`	`// Assign signal state of the RLO to the operand.`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

): Close branch (S7-1500)

Description

Use the "Close branch" instruction to specify the end of a parenthesized expression. After processing of the instruction the complete result of the nested expression is linked according to the instruction that was used to open the nested expression.

Use the following instructions to open a nested expression:

- A(: AND logic operation with branch
- AN(: Negated AND logic operation with branch
- O(: OR logic operation with branch
- ON(: Negated OR logic operation with branch
- X(: EXCLUSIVE OR logic operation with branch
- XN(: Negated EXCLUSIVE OR logic operation with branch

After the "Close branch" instruction, the CPU sets the binary result BR to the signal state that it had before the parenthesized expression.

Example

The following example shows how the instruction works:

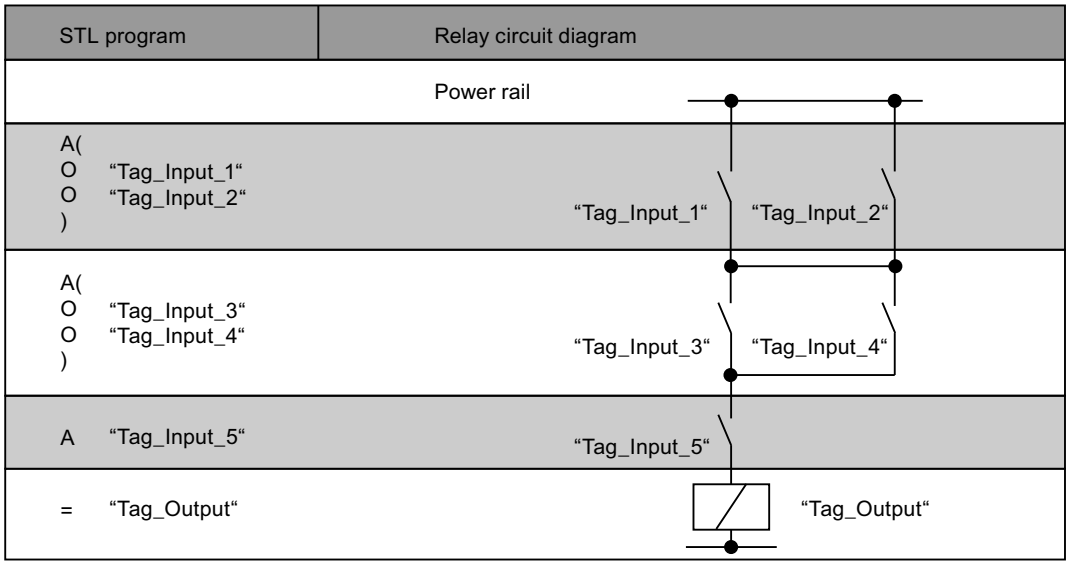

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

=: Assignment (S7-1500)

Description

Use the "Assignment" instruction to assign the signal state of a result of logic operation (RLO) saved in the CPU to a specified operand. If the RLO has signal state "1", the operand is set. If the signal is "0", the operand is reset to "0".

Parameters

The following table shows the parameters of the "Assignment" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the current RLO is assigned.

Example

The following example shows how the instruction works:

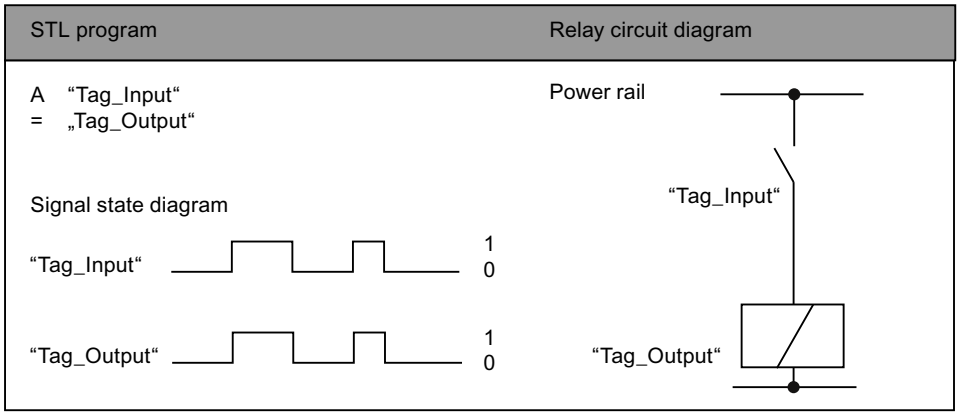

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

R: Reset (S7-1500)

Description

Use the "Reset" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed when the current result of logic operation (RLO) is "1". After execution of the instruction the specified operand is reset to "0". If there is a current RLO of "0", the signal state of the specified operand remains unchanged.

Parameters

The following table shows the parameters of the "Reset" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is reset when RLO = "1".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input_1"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
A "Tag_Input_2"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
R "Tag_Output"               // Reset the operand to "0" if RLO is "1"
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

S: Set (S7-1500)

Description

Use the "Set" instruction to set the signal state of a specified operand to "1".

The instruction is only executed when the current result of logic operation (RLO) is "1". After the instruction executes, the specified operand is set to "1". If there is a current RLO of "0", the signal state of the specified operand remains unchanged.

Parameters

The following table shows the parameters of the "Set" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is set when RLO = "1".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
S "Tag_Output"	// Sets the operand to "1" if RLO = "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

NOT: Invert RLO (S7-1500)

Description

Use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). You can use the "Invert RLO" instruction anywhere, even within a logic operation.

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the "Tag_Input_1" operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the "Tag_Input_2" operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the "Tag_Input_3" operand for "1" and AND with the current RLO.
NOT	// Invert signal state of the RLO
= "Tag_Output"	// Assign signal state of the RLO to the "Tag_Output" operand.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Input_1	1	1	1	0	0	0	1	0
Tag_Input_2	0	1	1	0	1	1	0	1
Tag_Input_3	0	0	1	0	0	1	1	1
Tag_Output	1	1	0	1	1	1	1	1

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

SET: Set RLO to 1 (S7-1500)

Description

Use the "Set RLO to 1" instruction to set the current result of logic operation (RLO) to signal state "1".

Example

The following example shows how the instruction works:

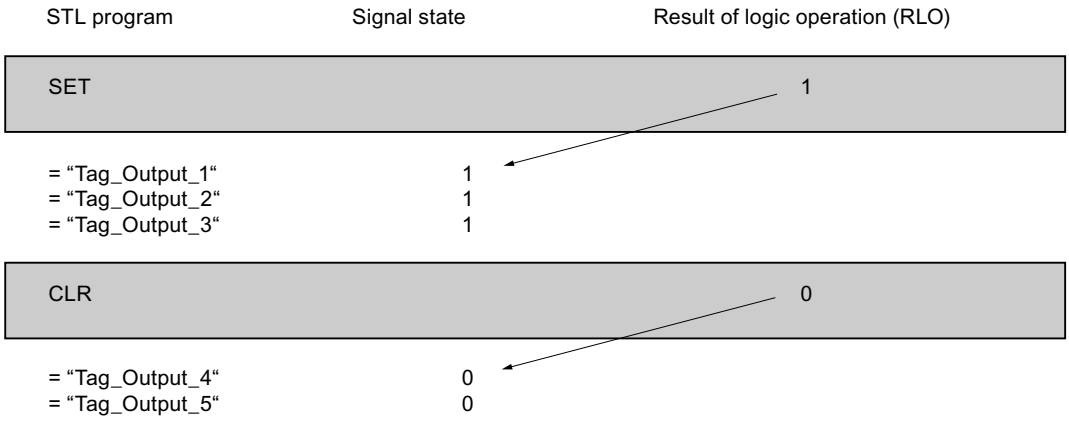

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

CLR: Reset RLO to 0 (S7-1500)

Description

Use the "Reset RLO to 0" instruction to set the current result of logic operation (RLO) to signal state "0".

Example

The following example shows how the instruction works:

STL program	Signal state	Result of logic operation (RLO)
SET		1
= “Tag_Output_1“	1	
= “Tag_Output_2“	1	
= “Tag_Output_3“	1	
CLR		0
= “Tag_Output_4“	0	
= “Tag_Output_5“	0	

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

SAVE: Save RLO in BR bit (S7-1500)

Description

Use the "Save RLO in BR bit" instruction to save the result of logic operation (RLO) in the binary result (BR). When it executes, the instruction transfers the signal state of the current result of logic operation to the status bit BR. The instruction operates independent of the conditions and does not affect any other status bits.

Note

After executing the "Save RLO in BR bit" instruction, the BR bit can be modified again by the following instructions in the same or lower-level blocks.

Use the "Save RLO in BR bit" instruction to check the execution status of a block. If, for example, you use the instruction "Save RLO in BR bit" at the end of a block, the BR bit is set to the signal state of the current result of logic operation of the block.

The BR bit helps to implement the EN/ENO mechanism for block calls. For more information on this topic, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state of the Tag_Input operand for "1".
SAVE	// Transfer signal state of the RLO to the BR bit.
BEC	// End of block if RLO = "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

FN: Scan RLO for negative signal edge (S7-1500)

Description

Use the "Scan RLO for negative signal edge" instruction to query if a "1" to "0" change exists in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query saved in the edge memory bit (<Operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative signal edge.

The following figure shows the change in the RLO in case of a negative or positive signal edge:

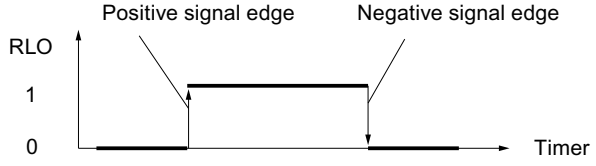

The negative signal edge is queried each time the instruction executes. When a negative signal edge is detected, the RLO is set to signal state "1" for the length of one program cycle after the edge evaluation. In all other cases, the signal state of the RLO is "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the previous query is saved.

Example

The following example shows how the instruction works:

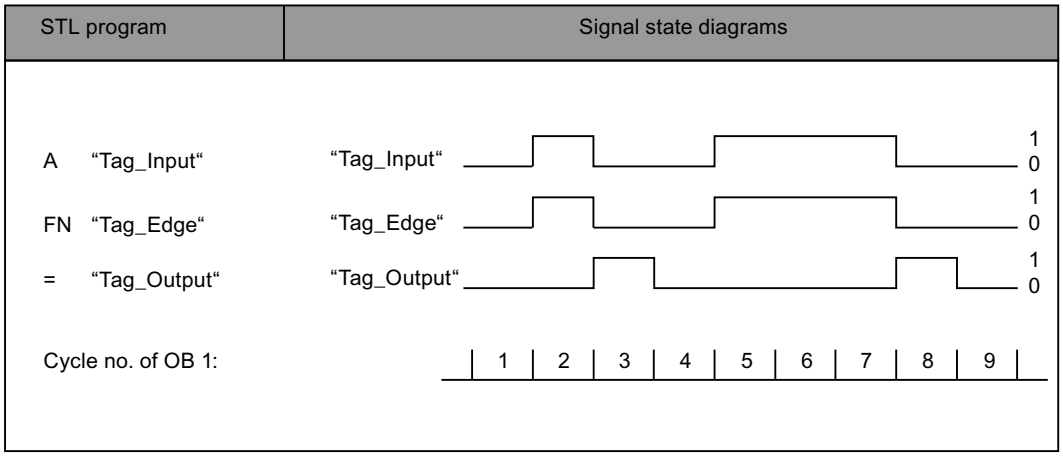

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

FP: Scan RLO for positive signal edge (S7-1500)

Description

Use the "Scan RLO for positive signal edge" instruction to query if a "0" to "1" change exists in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query saved in the edge memory bit (<Operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

The following figure shows the change in the RLO in case of a negative or positive signal edge:

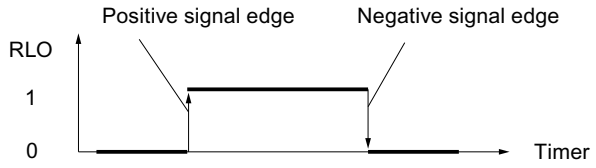

The positive signal edge is queried each time the instruction executes. When a positive signal edge is detected, the RLO is set to signal state "1" for the length of one program cycle after the edge evaluation. In all other cases, the signal state of the RLO is "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the instruction "Scan RLO for positive signal edge":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the previous query is saved.

Example

The following example shows how the instruction works:

STL program	Signal state diagrams

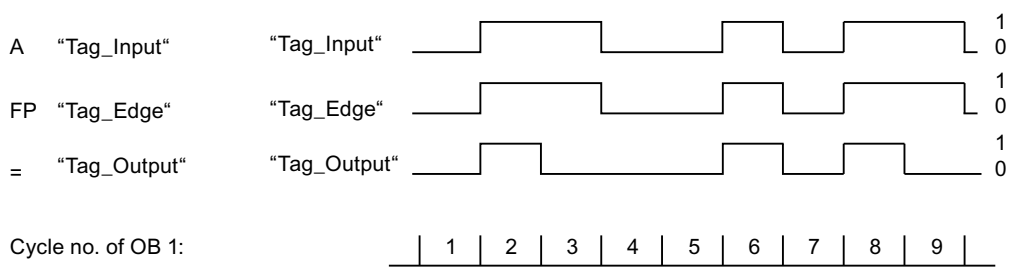

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

Timer operations (S7-1500)

FR: Enable timer (S7-1500)

Description

Use the instruction "Enable timer" to restart a timer. The instruction is executed with a positive edge and resets the internal edge trigger flag for the starting of a timer. During the processing of an instruction for starting the timer with a result of logic operation "1", the timer restarts even if no positive signal edge is present before the start instruction. If the result of logic operation is "0" before a start instruction, the resetting of the internal edge trigger flag does not affect the timer.

A change of the result of logic operation to "0" before the "Enable timer" instruction has no affect after the restart of the timer.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Enable timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Enabled timer

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_EnableInput"	// Query operand for "1" and link with the current RLO according to AND.
FR "MyTimer"	// If the signal edge is positive, enable timer.
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

L: Load timer value (Page 1506)

STL Basics (Page 7245)

L: Load timer value (S7-1500)

Description

With the "Load timer value" instruction you load the timer value of a specified timer in accumulator 1. The dual-coded timer value is loaded without the timebase. Zeros are written to accumulator 1 instead of the timebase.

After the loading, the value in accumulator 1 corresponds to a positive number of the data type INT.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

The following figure is an example of how the time value is loaded in the accumulator:

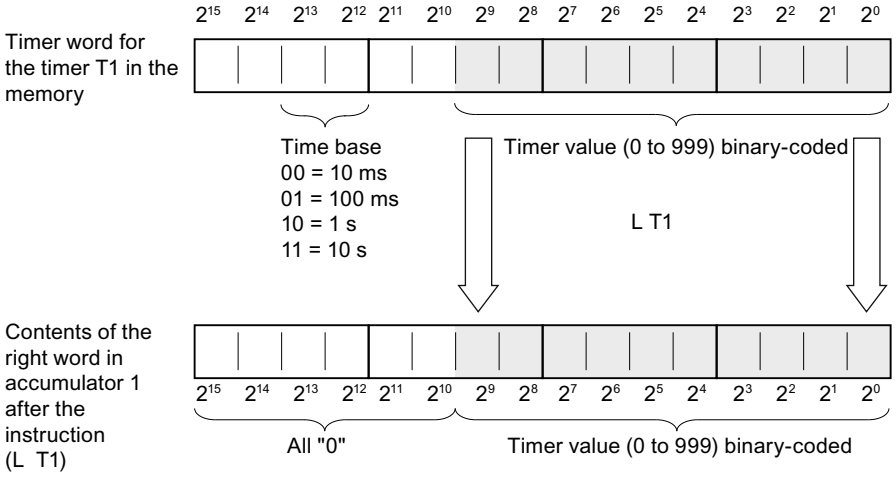

Parameters

The following table shows the parameters of the "Load timer value" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer whose current time value is loaded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_EnableInput"`	`// Query operand for "1" and link with the current RLO according to AND.`
`FR "MyTimer"`	`// If edge is positive, enable timer.`
`A "Tag_StartInput"`	`// Query operand for "1" and link with the current RLO according to AND.`
`L S5T#10s`	`// Load period of time (10s)`
`SP "MyTimer"`	`// Start timer as pulse if there is a positive edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".`
`A "Tag_Reset"`	`// Query operand for "1" and link with the current RLO according to AND.`
`R "MyTimer"`	`// If RLO = "1", reset timer.`
`A "MyTimer"`	`// Query status of timer for "1".`
`= "Tag_TimerStatus"`	`// Assign timer status to the operand.`
`L "MyTimer"`	`// Load current time value of timer in accumulator 1.`

STL	Explanation
`T "Tag_TimerValue"`	`// Transfer current time value to operand "Tag_TimerValue".`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

LC: Load BCD-coded timer value (S7-1500)

Description

With the "Load BCD-coded timer value" instruction you load a BCD-coded timer value of a specified timer that exists dual coded in accumulator 1. During loading the timebase is also transferred to Accumulator 1.

After loading, the value in accumulator 1 corresponds to a time duration in S5TIME format. The left word in accumulator 1 is padded with zeros.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

The following figure shows how the time value is loaded in the accumulator:

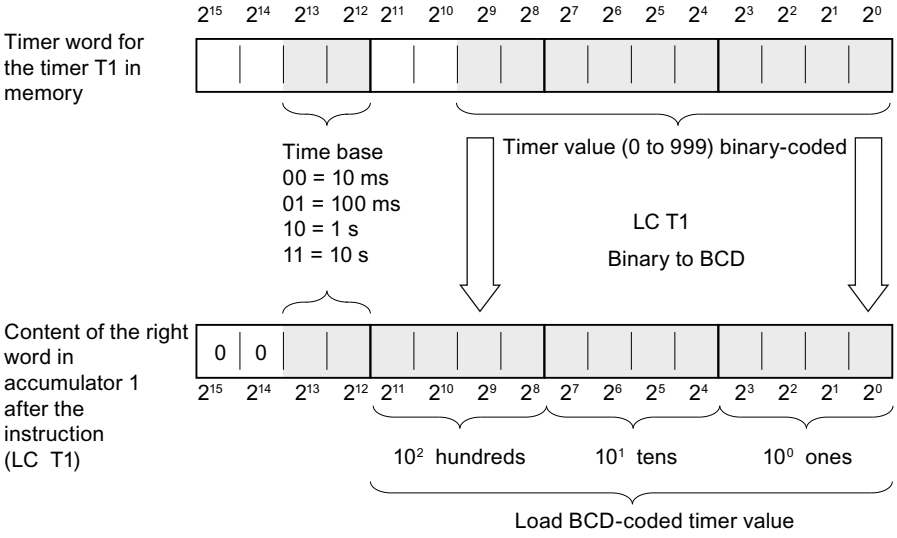

Parameters

The following table shows how the parameters of the "Load BCD-coded timer value" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer whose current BCD-coded timer value is loaded.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_EnableInput"	// Query operand for "1" and link with the current RLO according to AND.
FR "MyTimer"	// If edge is positive, enable timer.
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start timer as pulse if there is a positive edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
LC "MyTimer"	// Load current BCD-coded timer value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

L: Load timer value (Page 1506)

STL Basics (Page 7245)

R: Reset timer (S7-1500)

Description

You can use the "Reset timer" instruction to reset a specified timer to "0". The instruction is executed when the current result of logic operation (RLO) is "1". As long as the RLO is "1" before the instruction, the specified timer returns the value "0". The instruction also resets the time value and the time basis of the programmed period of time to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

Note

The "Reset timer" instruction does not reset the internal edge memory bit. To reset the internal edge memory bit, program the "Enable timer" instruction or execute an instruction to start the time with the signal state "0".

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Reset timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is reset

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Reset"                // Query operand for "1" and link with the current RLO
                             according to AND.
R "MyTimer"                  // If RLO = "1", reset timer.
A "MyTimer"                  // Query status of timer for "1".
= "Tag_TimerStatus"          // Assign timer status to the operand.
L "MyTimer"                  // Load current time value of timer in accumulator 1.
T "Tag_TimerValue"           // Transfer current time value to operand "Tag_Time-
                             rValue".
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

L: Load timer value (Page 1506)

STL Basics (Page 7245)

SP: Start pulse timer (S7-1500)

Description

You can use the "Start pulse timer" instruction to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1, as long as the result of logic operation (RLO) before the instruction remains "1". If the RLO changes to "0" before the end of the period of time, the timer is stopped. The queries of the timer status for "1" return the query result "1", as long as the period of time is running.

The timer can be restarted with a positive signal edge before the "Start pulse timer" instruction or using the "Enable timer" instruction. This restart can be effected provided the timer is not reset.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start pulse timer" instruction, the time value is counted down according to the timebase. The timer runs until the counter value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during the execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Start pulse timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer that is started as pulse.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:

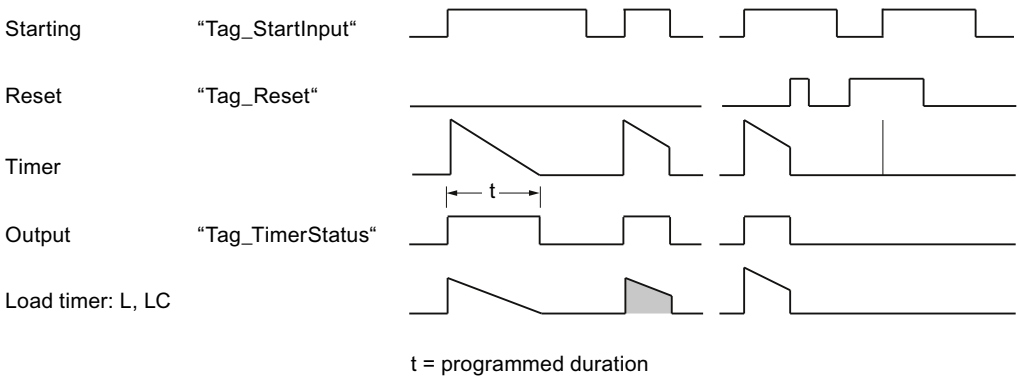

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

L: Load timer value (Page 1506)

STL Basics (Page 7245)

SE: Start extended pulse timer (S7-1500)

Description

You can use the "Start extended pulse timer" to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1, as long as the result of logic operation changes to "0" before the instruction. The queries of the timer status for "1" return the query result "1", as long as the period of time is running.

With each positive signal edge, the instruction restarts the timer with the programmed time duration, even if the timer has still not run down.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start extended pulse timer" instruction, the time value is counted down according to the timebase. The timer runs until the counter value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during the execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 1506)

Parameters

The following table shows the parameters of the "Start extended pulse timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as extended pulse

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_StartInput"`	`// Query operand for "1" and link with the current RLO according to AND.`
`L S5T#10s`	`// Load period of time (10s)`

STL	Explanation
SE "MyTimer"	// Start timer as extended pulse if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:

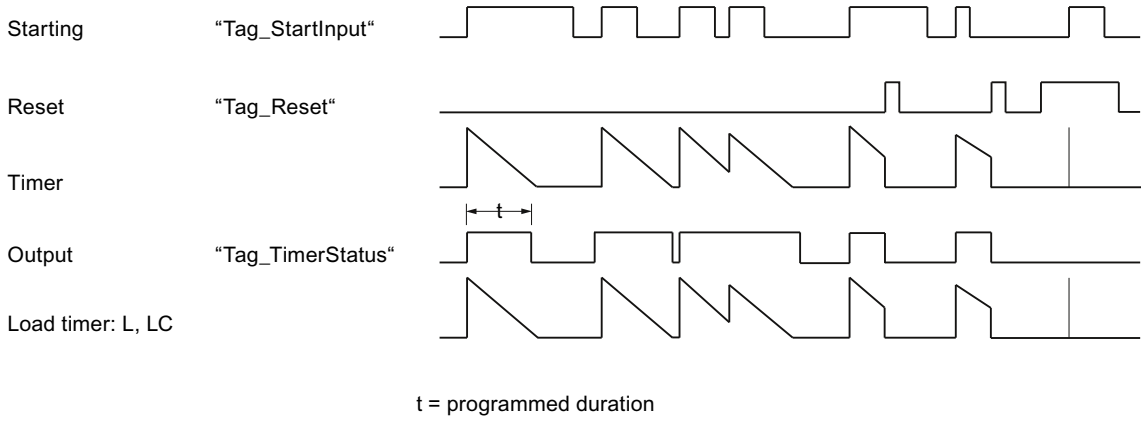

See also

Overview of the valid data types (Page 231)
Basic information on the status word (Page 186)
STL programming examples (Page 7280)
STL Basics (Page 7245)

SD: Start on-delay timer (S7-1500)

Description

The "Start on-delay timer" instruction starts a programmed timer when the signal state "1" is detected at the start input. As long as this signal state remains "1", the timer expires with the duration specified in accumulator 1. When the timer expires and the signal state at the start input is still "1", the query for timer status also returns "1". If the signal at the start input is "0", the timer is reset. In this case, querying the timer status returns the signal state "0". As soon as the signal at the start input changes to "1" again, the timer starts running again.

The signal state of the timer output is identical to the signal state of the start input. The start input is directly interconnected to the output and is not linked to the timer.

The duration in accumulator 1 is made up internally of a time value and a time base. When the instruction starts, the programmed time value counts down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during the execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Start on-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as on delay

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_StartInput"`	`// Query operand for "1" and link with the current RLO according to AND.`
`L S5T#10s`	`// Load period of time (10s)`

STL	Explanation
SD "MyTimer"	// Start timer as an on delay if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to the operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:

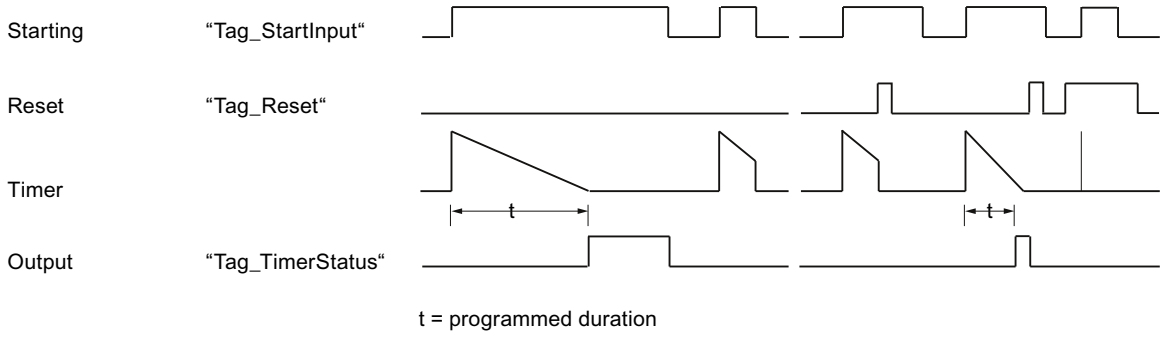

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

L: Load timer value (Page 1506)

STL Basics (Page 7245)

SS: Start retentive on-delay timer (S7-1500)

Description

You can use the "Start retentive on-delay timer" instruction to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs with the period of time specified in accumulator 1, even if the result of logic operation (RLO) changes to "0" before the instruction. When the time has run out, queries of the timer status for "1" return the query result "1". The query result is not hereby affected by the signal state of the current result of logic operation before the instruction.

With each positive edge, the instruction restarts the timer with the programmed period of time, even if the time has still not run down.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start retentive on-delay timer" instruction, the time value is counted down according to the time base. The timer runs until the counter value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during the execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 1506)

Parameters

The following table shows the parameters of the "Start retentive on-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer that is started as retentive on delay

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_StartInput"           // Query operand for "1" and link with the current RLO
                             according to AND.
```

STL	Explanation
L S5T#10s	// Load period of time (10s)
SS "MyTimer"	// Start retentive on-delay timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to the operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:

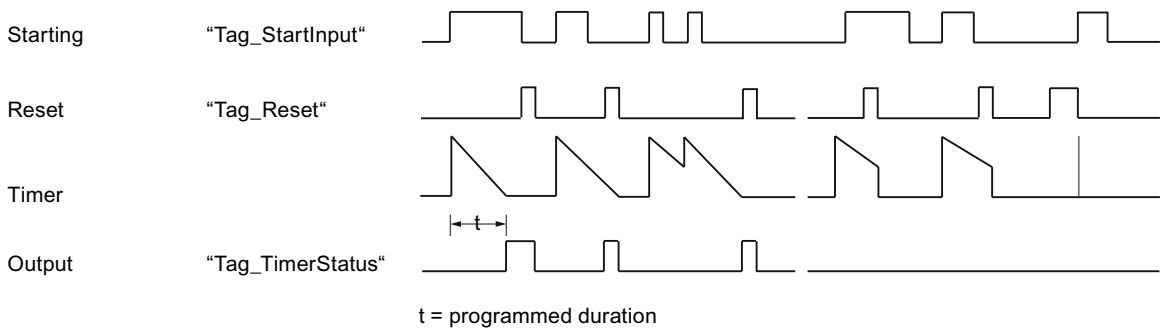

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

SF: Start off-delay timer (S7-1500)

Description

You can use the "Start off-delay timer" instruction to start a specified timer on a positive signal edge. If a negative signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1. If the result of logic operation (RLO) changes to "1" before the end of the period of time, the timer is reset. The timer is only restarted if a negative signal edge is detected during the processing of the instruction.

The queries of the timer status for "1" return the query result "1" if the signal state of the result of logic operation is "1" at the execution of the instruction or if the programmed time is running. If the time does not elapse or the RLO is "0", queries of the timer status for "1" return the query result "0".

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start off-delay timer" instruction, the time value is counted down according to the timebase. The timer runs until the counter value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during the execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 1506)

Parameters

The following table shows the parameters of the "Start off-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as off delay

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_StartInput"`	`// Query operand for "1" and link with the current RLO according to AND.`
`L S5T#10s`	`// Load period of time (10s)`
`SF "MyTimer"`	`// Start timer as an off delay if there is a negative signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".`
`A "Tag_Reset"`	`// Query operand for "1" and link with the current RLO according to AND.`
`R "MyTimer"`	`// If RLO = "1", reset timer and stop timing`
`A "MyTimer"`	`// Query status of timer for "1".`

STL	Explanation
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example.

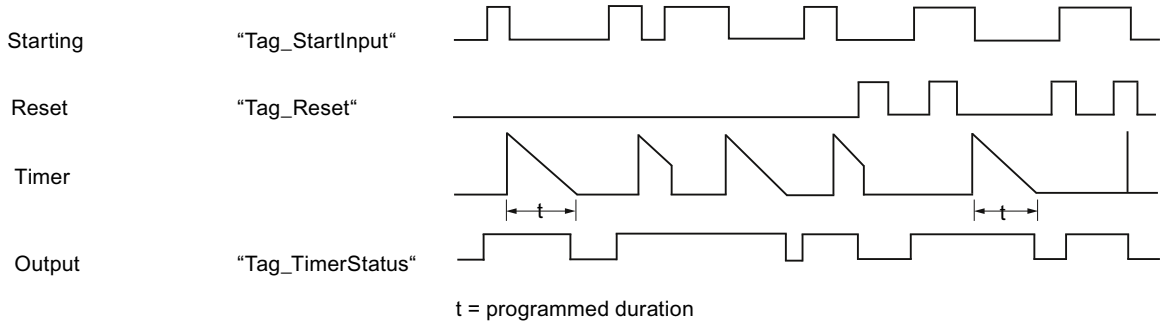

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

Counter operations (S7-1500)

FR: Enable counter (S7-1500)

Description

Use the "Enable counter" instruction to execute the instructions "Set counter", "Count up" and "Count down" without the presence of a positive signal edge. During processing, the instruction resets the internal edge flag to the signal state "0". The query before the corresponding count instruction must continue to have signal state "1".

The "Enable counter" instruction is executed when the result of logic operation (RLO) before the instruction changes from "0" to "1".

A counter does not have to be enabled for the count instruction to be executed.

Parameters

The following table shows the parameters of the "Enable counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter that is enabled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input_1"              // Query signal state for "1" and AND with the current
                             RLO.
FR "MyCounter"               // Enable counter
A "Tag_Input_2"              // Query signal state for "1" and AND with the current
                             RLO.
CD "MyCounter"               // Count down
A "MyCounter"                // Query counter status
= "Tag_Output"               // Assign query result to the operand.
```

In the example the counter ("MyCounter") is enabled when the signal state of the operand "Tag_Input_1" changes from "0" to "1". When the operand "Tag_Input_2" has signal state "1", the count value is decremented by one when the counter is enabled. When the signal state of the address "Tag_Input_2" is "0", the count value remains unchanged. The count value is also decremented by one regardless of the counter enabling, when the signal state of the address "Tag_Input_2" changes from "0" to "1".

The query of the counter status has the result "1" when the counter reading is greater than zero. If the counter reading is equal to zero, the result of the query is "0".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

L: Load counter (S7-1500)

Description

With the "Load counter" instruction you transfer the current value of a specified counter in binary coding to the right word of accumulator 1. The loaded counter is then available as a 16-bit integer for additional processing.

The following figure shows how the instruction works:

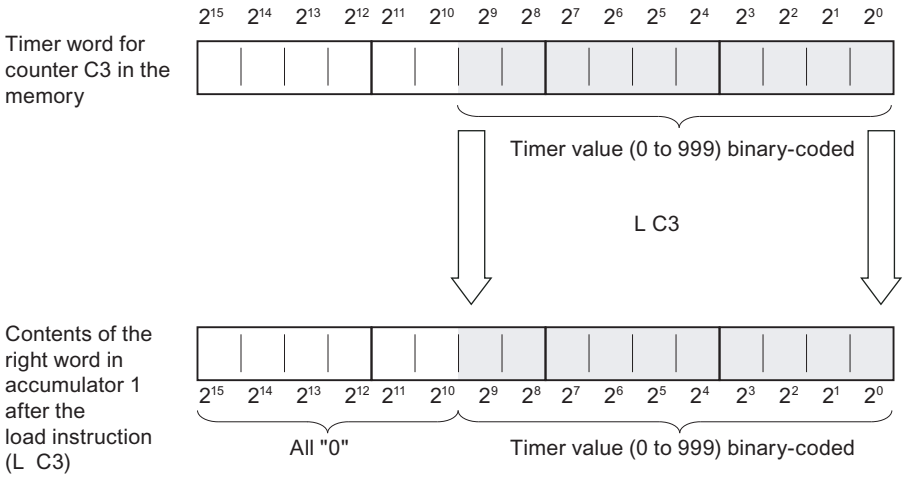

Parameters

The following table shows the parameters of the "Load counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose current value is loaded.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "MyCounter"                // Load current counter value binary-coded in accumu-
                             lator 1
L "Tag_Value"                // Move the counter value to accumulator 2.
                             // Load value of the operand "Tag_Value" in accumula-
                             tor 1
>=I                          // Query whether the count value is greater than or
                             equal to the value of the address "Tag_Value"
= "Tag_Output"               // Assign query result to the operand.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

LC: Load BCD-coded counter value (S7-1500)

Description

With the "Load BCD-coded counter value" instruction you transfer the current value of a specified BCD-coded counter to the right word of accumulator 1. The loaded counter value is then available in BCD form for additional processing.

The following figure shows how the "Load BCD-coded counter value" instruction works:

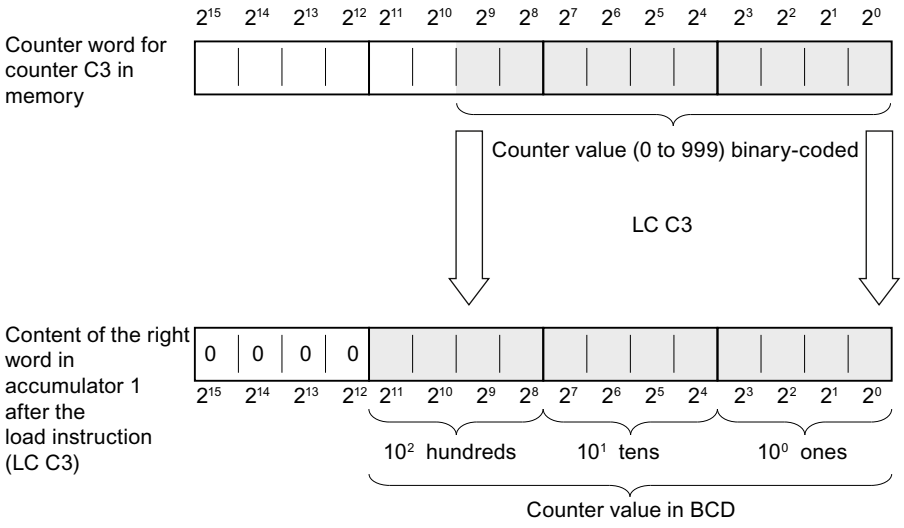

Parameters

The following table shows the parameters of the "Load BCD-coded counter value" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose current value is loaded.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`LC "MyCounter"`	`// Load current BCD-coded counter value in accumulator 1.`
`T "Tag_Output"`	`// Transfer current counter value to the address "Tag_Output" and save.`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

R: Reset counter (S7-1500)

Description

Use the "Reset counter" instruction to reset the current counter value of a specified counter to "0".

The instruction is only executed when the result of logic operation (RLO) before the instruction has signal state "1". As long as the RLO before the "Reset counter" instruction is "1", the queries of the counter status for "1" return the result "0".

When a counter is reset, the internal edge memory bit for the instructions "Set counter", "Count up" and "Count down" is not reset.

Parameters

The following table shows the parameters of the "Reset counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose count value is reset to "0".

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input"                // Query signal state for "1" and AND with the current
                             RLO.
R "MyCounter"                // Reset counter
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

S: Set counter (S7-1500)

Description

Use the "Set counter" instruction to set a specified counter to the value contained in accumulator 1 at the time of the query.

The "Set counter" instruction is executed when the result of logic operation before the instruction changes from "0" to "1". The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

To set a counter, accumulator 1 must contain a valid count value. Valid count values are values in BCD format with three decades. Only positive count values are permitted; the count instruction cannot process negative count values. If a valid BCD value is present in accumulator 1, the status bit OV has signal state "0". If there is no valid BCD value, the status bit OV has signal state "1". No synchronous error handling is started.

Parameters

The following table shows the parameters of the "Set counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter that is set.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_Input"`	`// Query signal state for "1" and AND with the current RLO.`
`L "Tag_CountValue"`	`// Load count value in accumulator 1.`
`S "MyCounter"`	`// Set counter to the value that is in accumulator 1`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

CU: Count up (S7-1500)

Description

Use the "Count up" instruction to increment the current counter value by one when the result of logic operation (RLO) before the instruction changes from "0" to "1". The counter value continues to be incremented for each positive signal edge until the high limit of "999" is reached. When the limit is reached the RLO before the instruction no longer has any affect on the count value.

A positive signal edge is always required to execute the "Count up" instruction. The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

Parameters

The following table shows the parameters of the "Count up" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter, whose count value is incremented by one on a positive edge.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_Input"`	`// Query signal state for "1" and AND with the current RLO.`
`CU "MyCounter"`	`// Increment count value by one on a positive edge.`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

CD: Count down (S7-1500)

Description

Use the "Count down" instruction to decrement the current counter value by one when the result of logic operation (RLO) before the instruction changes from "0" to "1". The count value continues to be decremented for each positive signal edge until the low limit of "0" is reached. When the limit is reached the RLO before the instruction no longer has any affect on the count value. Counting does not occur when count value is negative.

A positive signal edge is always required to execute the "Count down" instruction. The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

Parameters

The following table shows the parameters of the "Count down" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter, whose count value is decremented by one on a positive edge.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state for "1" and AND with the current RLO.
CD "MyCounter"	// Decrement counter value by one on a positive edge.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

Comparator operations (S7-1500)

? I: Compare 16-bit integers (S7-1500)

Description

With the "Compare 16-bit integers" instruction, you compare the content of the right word in accumulator 2 with the content of the right word in accumulator 1. The contents of accumulators 1 and 2 are interpreted as 16-bit integers (INT). The instruction performs the comparison depending on the specified comparison function.

Note

Unsigned data types (e.g. UINT) are not supported.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==I	Compare 16-bit integers for "equivalence"
<>I	Compare 16-bit integers for "non-equivalence"
>I	Compare 16-bit integers for "greater than"
<I	Compare 16-bit integers for "less than"
>=I	Compare 16-bit integers for "greater than or equal to"
<=I	Compare 16-bit integers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". When the condition of the comparison is not met, the RLO is set to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==I	0	1	0
<>I	1	0	1
>I	1	0	0
<I	0	0	1
>=I	1	1	0
<=I	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function you can evaluate the result of logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input_1"              // Load value of the operand "Tag_Input_1" in accumu-
                             lator 1.
L "Tag_Input_2"              // Shift the contents of accumulator 1 (Tag_Input_1)
                             to accumulator 2.
                             // Load value of the operand "Tag_Input_2" in accumu-
                             lator 1.
>I                           // Compare whether the value of the operand "Tag_In-
                             put_1" is greater than the value of the operand
                             "Tag_Input_2".
= "Tag_Output"               // Assign result of the comparison to the operand
                             "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	10	10	5
Tag_Input_2	5	10	10
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

? D: Compare 32-bit integers (S7-1500)

Description

Use the instruction "Compare 32-bit integers" to compare the contents of accumulator 2 (ACCU 2) with the contents of accumulator 1 (ACCU 1). The contents of accumulators 1 and 2 are interpreted as 32-bit integers (DINT). The instruction performs the comparison depending on the specified comparison function.

Note

Unsigned data types (e.g. UDINT) are not supported.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==D	Compare 32-bit integers for "equivalence"
<>D	Compare 32-bit integers for "non-equivalence"

Comparison function	Meaning
>D	Compare 32-bit integers for "greater than"
<D	Compare 32-bit integers for "less than"
>=D	Compare 32-bit integers for "greater than or equal to"
<=D	Compare 32-bit integers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". If the condition of the comparison is not met, the RLO is reset to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==D	0	1	0
<>D	1	0	1
>D	1	0	0
<D	0	0	1
>=D	1	1	0
<=D	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function you can evaluate the result of logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input_1"`	`// Load value of the operand "Tag_Input_1" in accumulator 1.`
`L "Tag_Input_2"`	`// Contents of accumulator 1 (Tag_Input_1) shifted to accumulator 2.`
	`// Load value of the operand "Tag_Input_2" in accumulator 1.`
`>D`	`// Compare whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".`
`= "Tag_Output"`	`// Assign result of the comparison to the operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	150 000	150 000	100 000
Tag_Input_2	100 000	150 000	150 000
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

? R: Compare floating-point numbers (S7-1500)

Description

Use the instruction "Compare floating-point numbers" to compare the contents of accumulator 2 (ACCU 2) with the contents of accumulator 1 (ACCU 1). The contents of accumulators 1 and 2 are interpreted as floating-point numbers. For the comparison, accumulators 1 and 2 must contain valid floating-point numbers. The instruction performs the comparison depending on the specified comparison function.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==R	Compare floating-point numbers for "equivalence"
<>R	Compare floating-point numbers for "non-equivalence"
>R	Compare floating-point numbers for "greater than"
<R	Compare floating-point numbers for "less than"
>=R	Compare floating-point numbers for "greater than or equal to"
<=R	Compare floating-point numbers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". If the condition of the comparison is not met, the RLO is reset to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==R	0	1	0
<>R	1	0	1
>R	1	0	0
<R	0	0	1
>=R	1	1	0
<=R	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

Note

During a comparison of invalid floating-point numbers, the CPU sets the result of logic operation to "0" and the status bits CC 0, CC 1, OVand OS to "1".

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function you can evaluate the result of logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input_1"`	`// Load value of the operand "Tag_Input_1" in accumulator 1.`
`L "Tag_Input_2"`	`// Shift the contents of accumulator 1 (Tag_Input_1) to accumulator 2.`
	`// Load value of the operand "Tag_Input_2" in accumulator 1`
`>R`	`// Compare whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".`
`= "Tag_Output"`	`// Assign result of the comparison to the operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	10 515	10 515	5 232
Tag_Input_2	5 232	10 515	10 515
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

Math functions (S7-1500)

Integers (S7-1500)

+I: Add integers (16 bit) (S7-1500)

Description

Use the instruction "Add integers (16 bit)" to add the values in the right words of accumulators 1 and 2. The instruction interprets these values as 16-bit integers.

The instruction saves the sum in the right word of accumulator 1. The left word of accumulator 1 remains unchanged.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for addition.
L "Tag_Value_2"	// Load second value for addition.
+I	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

-I: Subtract integers (16 bit) (S7-1500)

Description

With the "Subtract integers (16 bit)" instruction you subtract the value of the right word of accumulator 1 from the value of the right word of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as 16-bit integers.

The instruction saves the difference in the right word of accumulator 1. The left word of accumulator 1 remains unchanged.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load second value for addition in accumulator 1.

STL	Explanation
-I	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

*I: Multiply integers (16 bit) (S7-1500)

Description

Use the instruction "Multiply integers (16 bit)" to multiply the values in the right words of accumulators 1 and 2. The instruction interprets these values as 16-bit integers.

The instruction saves the product in accumulator 1 as 32-bit integer (DINT).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*I	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

/I: Divide integers (16 bit) (S7-1500)

Description

With the "Divide integers (16 bit)" instruction you divide the contents of the right word in accumulator 2 by the contents of the right word in accumulator 1. The instruction interprets the values of the accumulators as 16-bit integers.

The instruction provides two results: the quotients and the remainder of the division. The quotient is the whole number result of the division. The instruction saves this in the right word of accumulator 1. The instruction saves the remainder of the division in the left word of accumulator 1. If the dividend is negative, the remainder of the division is also negative.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value as quotient and in each case the value zero as remainder. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load dividend in accumulator 1.`
`L "Tag_Value_2"`	`// Move dividend to accumulator 2.`
	`// Load divisor in accumulator 1.`
`/I`	`// Divide the value of accumulator 2 by the value of accumulator 1.`
`T "Tag_Result"`	`// Transfer the result to operand "Tag_Result".`

Before processing

The following table shows the contents of the right words in accumulators 1 and 2 before executing the "Divide integers (16 bit)" instruction:

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the "Divide integers (16 bit)" instruction:

Accumulator 1	Value
Bit 0 to 15 (right word)	3
Bit 16 to 31 (left word)	1

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL programming examples (Page 7280)

STL Basics (Page 7245)

+D: Add double integers (32 bit) (S7-1500)

Description

Use the "Add double integers (32 bit)" instruction to add the values in accumulators 1 and 2. The instruction interprets these values as 32-bit integers. The instruction saves the sum in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load first value for addition.
L "Tag_Value_2"              // Load second value for addition.
+D                           // Add values
T "Tag_Result"               // Transfer sum to the operand "Tag_Result".
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

-D: Subtract double integers (32 bit) (S7-1500)

Description

With the "Subtract double integers (32 bit)" instruction you subtract the value of accumulator 1 from the value of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as 32-bit double integers. The instruction saves the result of the subtraction in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load the second value of subtraction into accumulator 1.
-D	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

*D: Multiply double integers (32 bit) (S7-1500)

Description

Use the "Multiply double integers (32 bit)" instruction to multiply the values in accumulators 1 and 2. The instruction interprets these values as 32-bit integers. The instruction saves the product in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*D	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

/D: Divide double integers (32 bit) (S7-1500)

Description

With the "Divide double integers (32 bit)" instruction you divide the value of accumulator 2 by the value of accumulator 1. The instruction interprets the values of accumulators 1 and 2 as 32-bit double integers. The instruction saves the result of the division (quotient) in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value zero as quotient. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
/D	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer quotients to the operand "Tag_Result".

Before processing

The following table shows the contents of accumulators 1 and 2 before executing the "Divide double integers (32 bit)" instruction:

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the "Divide double integers (32 bit)" instruction:

Accumulator	Value
Accumulator 1	3

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

+: Add constants (S7-1500)

Description

With the "Add constants" instruction, you add the value of a specified constant to the contents of accumulator 1. You can specify the constant value as 16-bit integer or 32-bit double integer. The instruction stores the result in accumulator 1. When adding 16-bit integers, the instruction only influences the right word of accumulator 1.

To execute the addition of a 16-bit integer as DINT math function, enter "L#" before the constant. The "Add constants" instruction permits the use of negative numbers.

The instruction does not affect any status bits.

Parameters

The following table shows the parameters of the "Add constants" instruction:

Parameters	Data type	Description
<constant>	INT, DINT	Constants to be added

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1".
+ 25	// Increment the value of accumulator 1 by 25.
T "Tag_Result_1"	// Transfer contents of accumulator 1 to the operand "Tag_Result_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2" in accumulator 1
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
+ -100	// Decrement the value of accumulator 1 by 100.
<I	// Check whether the value of accumulator 2 is less than the value of accumulator 1
JC NEXT	// If the result of the query is "1", jump to jump label NEXT.
	// If the result of the query is "0", execute the next instruction.
T "Tag_Result_2"	Transfer contents of accumulator 1 to the operand "Tag_Result_2".
L "Tag_Value_4"	// Load value of the operand "Tag_Value_4".
+ L#20	// Increment content of accumulator 1 by 20.
	// The addition is done as a DINT calculation
T "Tag_Result_3"	// Transfer contents of accumulator 1 to the operand "Tag_Result_3".
NEXT: L "Tag_Value_5"	// Jump label "NEXT"
	// Load value of the operand "Tag_Value_5".
+D	// Add values of accumulators 1 and 2.
T "Tag_Result_4"	// Transfer sum to the operand "Tag_Result_4".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

INC: Increment (S7-1500)

Description

Use the "Increment" instruction to increment the value in accumulator 1 by the value specified as parameter of the instruction. The specified value can be in a value range from 0 to 255. The value is incremented only in the right byte of accumulator 1. There is no transfer to the byte located on the left.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The content of accumulator 2 remains unchanged.

If the parameter value is outside the permitted range of value, the result will be corrupted without generating an error message.

Parameters

The following table shows the parameters of the "Increment" instruction:

Parameters	Format	Description
<constant>	8-bit integer	Value by which the value is incremented.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load value of the operand in accumulator 1.
INC 5                        // Increment value of accumulator 1 by 5.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

DEC: Decrement (S7-1500)

Description

Use the "Decrement" instruction to decrement the value in accumulator 1 by the value specified as parameter of the instruction. The specified value can be in a value range from 0 to 255. The value is decremented only in the right byte of accumulator 1. There is no transfer to the byte located on the left.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The content of accumulator 2 remains unchanged.

If the parameter value is outside the permitted range of value, the result will be corrupted without generating an error message.

Parameters

The following table shows the parameters of the "Decrement" instruction:

Parameters	Format	Description
<constant>	8-bit integer	Value by which the value is decremented.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load value of the operand in accumulator 1.
DEC 5                        // Decrement the value of accumulator 1 by 5.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

MOD: Return remainder of division (S7-1500)

Description

Use the "Return remainder of division" instruction to divide the value of accumulator 2 by the value of accumulator 1 and save the remainder of the division.

The instruction interprets the values of accumulators 1 and 2 as 32-bit integers. The instruction stores the remainder of the division in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the remainder of division is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction returns the value zero as result. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
MOD	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer remainder of division to the operand "Tag_Result".

Before processing

The following table shows the contents of accumulators 1 and 2 before executing the instruction "Return remainder of division":

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the instruction "Return remainder of division":

Accumulator	Value
Accumulator 1	1

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

Floating-point numbers (S7-1500)

+R: Add floating-point numbers (S7-1500)

Description

Use the "Add floating-point numbers" instruction to add the values contained in accumulators 1 and 2. The instruction interprets these values as floating-point numbers. The instruction saves the sum in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for addition.
L "Tag_Value_2"	// Load second value for addition.
+R	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

-R: Subtract floating-point numbers (S7-1500)

Description

With the "Subtract floating-point numbers" instruction you subtract the value of accumulator 1 from the value of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as floating-point numbers. The instruction stores the difference in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.

STL	Explanation
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load the second value of subtraction into accumulator 1.
-R	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

*R: Multiply floating-point numbers (S7-1500)

Description

Use the "Multiply floating-point numbers" instruction to multiply the values contained in accumulators 1 and 2. The instruction interprets these values as floating-point numbers. The instruction saves the product in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*R	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

/R: Divide floating-point numbers (S7-1500)

Description

With the "Divide floating-point numbers" instruction you divide the value of accumulator 2 by the value of accumulator 1. The instruction interprets the values of accumulators 1 and 2 as floating-point numbers. The instruction saves the result of the division (quotient) in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value zero as quotient. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
/R	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer quotients to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

ABS: Form absolute value (S7-1500)

Description

Use the "Form absolute value" instruction to form the absolute value of the value in accumulator 1. The instruction interprets the contents of accumulator 1 as a floating-point number and sets the sign of the mantissa to "0". The result is stored in accumulator 1.

The "Form absolute value" instruction does not affect any status bits.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load floating-point number into accumulator 1
ABS	// Form absolute value
T "Tag_Result"	// Transfer the result to operand "Tag_Result".

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	-1.5E+02
Tag_Result	1.5E+02

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

Advanced floating-point numbers (S7-1500)

SQR: Form square (S7-1500)

Description

Use the "Form square" instruction to calculate the square of a floating-point number located in accumulator 1. The result is stored in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SQR	// Form square
	// Store result in accumulator 1.

STL	Explanation
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

SQRT: Form square root (S7-1500)

Description

Use the "Form square root" instruction to form the square root of the floating-point number located in accumulator 1. The result is stored in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value less than zero or an invalid floating-point number, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

If accumulator 1 has the value "0", the instruction returns the value "0".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
SQRT	// Get square root

STL	Explanation
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

EXP: Form exponential value (S7-1500)

Description

Use the "Form exponential value" to form the exponent from the base e (e = 2.718282) and the floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
EXP	// Form exponential value
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

LN: Form natural logarithm (S7-1500)

Description

Use the "Form natural logarithm" instruction to calculate the natural logarithm of the value in accumulator 1 to base e (e = 2.718282).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value less than or equal to zero, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
LN	// Form natural logarithm.
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

SIN: Form sine value (S7-1500)

Description

Use the "Form sine value" instruction to calculate the sine of the angle in accumulator 1. The angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SIN	// Form sine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

ASIN: Form arcsine value (S7-1500)

Description

Use the "Form arcsine value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number in a value range from "-1" to "1".

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from $-\pi/2$ to $+\pi/2$ (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ASIN	// Form arcsine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

COS: Form cosine value (S7-1500)

Description

Use the "Form cosine value" instruction to calculate the cosine of the angle in accumulator 1. The size of the angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
COS	// Form cosine value
	// Store result in accumulator 1.

STL	Explanation
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

ACOS: Form arccosine value (S7-1500)

Description

Use the "Form arccosine value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number in a value range from "-1" to "1".

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from "0" to "π" (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ACOS	// Form arccosine value
	// Store result in accumulator 1.

STL	Explanation
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

TAN: Form tangent value (S7-1500)

Description

Use the "Form tangent value" instruction to calculate the tangent of the angle in accumulator 1. The size of the angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
TAN	// Form tangent value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.

STL	Explanation
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

ATAN: Form arctangent value (S7-1500)

Description

Use the "Form arctangent value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number.

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from $-\pi/2$ to $+\pi/2$ (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ATAN	// Form arctangent value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.

STL	Explanation
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

STL Basics (Page 7245)

Load and transfer (S7-1500)

Load (S7-1500)

L: Load (S7-1500)

Description

With the "Load" instruction you load the content of a specified operand in accumulator 1. The operand that is to be loaded can be addressed as a byte, word or double word in one of the following memory areas:

- Process image of the inputs and outputs (I, Q)
- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)
- Pointer
- I/O (PI)
- Timer (T)
- Counter (C)

The memory area of accumulator 1 is organized in bytes and is 32 bits wide.

The instruction writes the content of an operand that is to be loaded in byte format right aligned in accumulator 1. The remaining bytes in accumulator 1 are filled with "0".

The instruction writes the content of the operand to be loaded in word format to the right word of accumulator 1. The higher addressed byte is hereby transferred into the right byte (bits 0 to 7) of accumulator 1. The lower addressed byte is written to the left of it. The remaining bytes in the left word of accumulator 1 are filled with "0".

The instruction writes the content of the operand to be loaded in double word format to the 32 bits of accumulator 1. The highest addressed byte is hereby transferred into the right byte (bits 0 to 7) of accumulator 1. The lowest addressed byte is written in bits 24 to 31.

The following table shows, as an example, how the contents of accumulator 1 are changed by the loading of operands in byte, word and double word format:

Instruction	Accumulator 1							
	31 . . .	. . . 24	23 . . .	. . . 16	15 . . .	. . . 8	7 . . .	. . . 0
Load a byte: L MB10	0000	0000	0000	0000	0000	0000	<MB10>	
Load a word: L MW10	0000	0000	0000	0000	<MB10>		<MB11>	
Load a double word: L MD10	<MB10>		<MB11>		<MB12>		<MB13>	

During the loading of the specified operand the previous contents of accumulator 1 are moved to accumulator 2. The "Load" instruction transfers the complete contents of accumulator 1 to accumulator 2. The previous content of accumulator 2 is lost during this.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Parameters

The following table shows the parameters of the "Load" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, Timers, REAL, DATE, TOD, CHAR	I, Q, PI, M, L, DB, DI, T, C, pointer, parameter	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1" in accumu-
                             lator 1
L "Tag_Value_2"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_2" in accumu-
                             lator 1
*R                           // Multiply the values of accumulator 1 and 2.
                             // Store product in accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in accumu-
                             lator 1
```

STL	Explanation
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL programming examples (Page 7280)

L STW: Load status word in accumulator 1 (S7-1500)

Description

Use the "Load status word in accumulator 1" instruction to load the status word in accumulator 1. When it executes, the instruction writes the individual status bits in the right word of accumulator 1 The remaining bits of accumulator 1 are filled with "0".

The following table shows the contents of the accumulator after executing the "Load status word in accumulator 1" instruction:

	Accumulator 1									
Bit number	31-9	8	7	6	5	4	3	2	1	0
Content	0	BR	CC 1	CC 0	OV	OS	0	0	RLO	0

The instruction is always executed independently of the signal state of the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
L STW	// Load status word in accumulator 1.
T "Tag_STW"	// Transfer the contents of accumulator 1 (status bits) to the operand "Tag_STW".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

LAR1: Load AR1 with contents of accumulator 1 (S7-1500)

Description

With the "Load AR1 with contents of accumulator 1" instruction you load address register 1 (AR1) with the contents of accumulator 1. The value of accumulator 1 must correspond to the format of a POINTER.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
L P#10.0	// Load pointer (P#10.0) in accumulator 1.
LAR1	// Load address register 1 with contents of accumulator 1.
L MW [AR1,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR1,P#2.1]	// Check whether signal state of bit I12.1 is "1" and AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

LAR1 <D>: Load AR1 with double word or area pointer (S7-1500)

Description

Use the "Load AR1 with double word or area pointer" instruction to load address register 1 (AR1) with the contents of a double word or an area internal or area crossing pointer. The contents of the double word must match the format of an area pointer.

The pointer and the double word can address one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Parameters

The following table shows the parameters of the "Load AR1 with double word or area pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Input	DWORD, POINTER	D, M, L	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.0	// Load address register 1 with contents of the area internal pointer P#10.0.
L MW [AR1,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR1,P#2.1]	// Check whether signal state of bit I12.1 is "1" and AND with current RLO.

STL	Explanation
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".
LAR1 MD20	// Load address register 1 with contents of the MD20 (MD20 = P#30.0)
A I [AR1,P#2.1]	// Check whether signal state of bit I32.1 is "1" and AND with current RLO.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

LAR1 AR2: Load AR1 with contents of AR2 (S7-1500)

Description

Use the "Load AR1 with contents of AR2" instruction to load the contents of address register 2 (AR2) into address register 1 (AR1).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
LAR2 P#20.0	// Load address register 2 with contents of the area internal pointer P#20.0.
LAR1 AR2	// Copy contents of address register 2 into address register 1.
L MW [AR1,P#4.0]	// Load MW24 in accumulator 1.
T "Tag_2"	// Transfer accumulator 1 to "Tag_2".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

LAR2: Load AR2 with contents of accumulator 1 (S7-1500)

Description

With the "Load AR2 with contents of accumulator 1" instruction you load address register 2 (AR2) with the contents of accumulator 1. The value of accumulator 1 must correspond to the format of a POINTER.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
L P#10.0	// Load pointer (P#10.0) in accumulator 1.
LAR2	// Load address register 2 with contents of accumulator 1.
L MW [AR2,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR2,P#2.1]	// Check whether signal state of bit I12.1 is "1" and AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

LAR2 <D>: Load AR2 with double word or area pointer (S7-1500)

Description

Use the "Load AR2 with double word or area pointer" instruction to load address register 2 (AR2) with the contents of a double word or an area-internal or cross-area pointer. The contents of the double word must match the format of an area pointer (POINTER).

The pointer and the double word can address one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Parameters

The following table shows the parameters of the "Load AR2 with double word or area pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Input	DWORD, POINTER	D, M, L	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

STL	Explanation
LAR2 P#10.0	// Load address register 2 with contents of the area internal pointer P#10.0.
L MW [AR2,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR2,P#2.1]	// Check whether signal state of bit I12.1 is "1" and AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".
LAR2 MD20	// Load address register 2 with contents of the MD20 (MD20 = P#30.0)
A I [AR2,P#2.1]	// Check whether signal state of bit I32.1 is "1" and AND with current RLO.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

Transfer (S7-1500)

T: Transfer (S7-1500)

Description

Use the "Transfer" instruction to transfer the contents of accumulator 1 in byte, word or double word format to a specified operand. The contents of accumulator 1 are not changed by this instruction. The specified operand must be addressed in one of the following memory areas:

- Process image of the inputs and outputs (I, Q)
- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)
- I/O (PQ)

The number of bytes to be transferred depends on the format of the specified operand. If the operand exists in byte format, the instruction transfers the contents of the bits 0 to 7 of accumulator 1.

If the operand exists in word format, the instruction transfers the contents of the word that is right-aligned in accumulator 1. The right byte of accumulator 1 is transferred to the higher addressed byte of the operand. The bits 8 to 15 of the accumulator 1 are transferred to the lower byte of the operand.

If the specified operand exists in double word format, the instruction transfers the entire content of accumulator 1. The right byte of accumulator 1 is transferred to the highest addressed byte of the operand. The bits 24 to 31 of accumulator 1 are transferred to the lowest addressed byte of the operand.

The following table shows an example of how the contents of accumulator 1 are transferred in byte, word and double word format:

Instruction	Accumulator 1			
Bits	31 24	23 16	15 8	7 0
Contents of accumulator 1	byte (n)	byte (n+1)	byte (n+2)	byte (n+3)
Transfer of a byte				byte (n)

Instruction	Accumulator 1			
Transfer of a word			byte (n)	byte (n+1)
Transferring a double word	byte (n)	byte (n+1)	byte (n+2)	byte (n+3)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Parameters

The following table shows the parameters of the "Transfer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, Timers, REAL, DATE, TOD, CHAR	I, Q, PQ, M, L, DB, DI	Operand to which the contents of accumulator 1 are transferred.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result_1"	// Transfer product (contents of Accumulator 1) to the operand "Tag_Result_1".
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
T "Tag_Result_2"	// Transfer sum (contents of Accumulator 1) to the operand "Tag_Result_2"

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL programming examples (Page 7280)

T STW: Transfer accumulator 1 to status word (S7-1500)

Description

Use the "Transfer accumulator 1 to status word" instruction to transfer bits 0 to 8 of accumulator 1 to the status word.

The following table shows the bits in accumulator 1 that will overwrite status bits with processing of the instruction:

	Accumulator 1								
Bit number	8	7	6	5	4	3	2	1	0
Status bits	BR	CC 1	CC 0	OV	OS	0	0	RLO	0

The instruction is always executed independently of the signal state of the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_STW"`	`// Load value of the operand "Tag_STW" in accumulator 1.`
`T STW`	`// Transfer bits 0 to 8 of accumulator 1 to status word`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

CAR: Switch AR1 and AR2 (S7-1500)

Description

Use the instruction "Switch AR1 and AR2" to switch the contents of address registers 1 (AR1) and 2 (AR2).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
`LAR1 P#10.0`	`// Load address register 1 with contents of the area internal pointer P#10.0.`
`LAR2 P#20.0`	`// Load address register 2 with contents of the area internal pointer P#20.0.`
`CAR`	`// Switch contents of the address registers.`
`L MD [AR1,P#2.0]`	`// Load MD22 in accumulator 1.`
`T MD [AR2,P#2.0]`	`// Transfer MD12 too accumulator 2.`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

TAR1: Transfer AR1 to accumulator 1 (S7-1500)

Description

With the "Transfer AR1 to accumulator 1" instruction you transfer the contents of address register 1 (AR1) to accumulator 1. The content of accumulator 1 is moved to accumulator 2 during this.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#15.0	// Load contents of the area-internal pointer P#15.0 into address register 1.
TAR1	// Transfer contents of address register 1 (P#15.0) to accumulator 1.
T "Tag_Pointer"	// Transfer contents of accumulator 1 to the operand "Tag_Pointer".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

TAR1 <D>: Transfer AR1 to double word (S7-1500)

Description

Use the "Transfer AR1 to double word" instruction to transfer the contents of address register 1 (AR1) to a double word. The double word must hereby be addressed in one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Parameters

The following table shows the parameters of the "Transfer AR1 to double word" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Output	DWORD	D, M, L	Operand to which the contents of accumulator 1 are transferred.

Example

The following example shows how the instruction works:

STL	Explanation
TAR1 %DBD20	// Transfer contents of address register 1 to data double word DBD20.
TAR1 %DID30	// Transfer contents of address register 1 to instance double word DID30.
TAR1 %LD18	// Transfer contents of address register 1 to local data double word LD18.
TAR1 %MD24	// Transfer contents of address register 1 to memory double word MD24.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

TAR1 AR2: Transfer AR1 to AR2 (S7-1500)

Description

Use the "Transfer AR1 to AR2" instruction to copy the contents of address register 1 (AR1) to address register 2 (AR2).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.4	// Load contents of the area-internal pointer P#10.4 into address register 1.
TAR1 AR2	// Transfer contents of address register 1 to address register 2.
A I [AR2,P#3.2]	// Check whether signal state of bit I13.6 is "1" and AND with current RLO.
= "Tag1"	// Set operand "Tag_1" to "1"

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

TAR2: Transfer AR2 to accumulator 1 (S7-1500)

Description

With the "Transfer AR2 to accumulator 1" instruction you transfer the contents of address register 2 (AR2) to accumulator 1. The content of accumulator 1 is moved to accumulator 2 during this.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
`LAR2 P#20.0`	`// Load address register 2 with contents of the area internal pointer P#20.0.`
`TAR2`	`// Transfer contents of address register 2 (P#20.0) to accumulator 1.`
`T "Tag_Pointer"`	`// Transfer contents of accumulator 1 to the operand "Tag_Pointer".`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

TAR2 <D>: Transfer AR2 to double word (S7-1500)

Description

Use the "Transfer AR2 to double word" instruction to transfer the contents of address register 2 (AR2) to a double word. The double word must hereby be addressed in one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Parameters

The following table shows the parameters of the "Transfer AR2 to double word" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Output	DWORD	D, M, L	Operand to which the contents of accumulator 2 are transferred.

Example

The following example shows how the instruction works:

STL	Explanation
TAR2 %DBD20	// Transfer contents of address register 2 to data double word DBD20.
TAR2 %DID30	// Transfer contents of address register 2 to instance double word DID30.
TAR2 %LD18	// Transfer contents of address register 2 to local data double word LD18.
TAR2 %MD24	// Transfer contents of address register 2 to memory double word MD24.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

Conversion operations (S7-1500)

BTI: Convert BCD to integer (16 bit) (S7-1500)

Description

Use the "Convert BCD to integer (16 bit)" instruction to convert the value in the right word of accumulator 1 to a 16-bit integer. The instruction interprets the word that is to be converted as a three-digit binary coded decimal (BCD).

The bits 1 to 11 of accumulator 1 indicate the value that will be converted. Values in the range from "-999" to "+999" are permitted.

The sign of the result value is read from bit 15 of accumulator 1. If the signal state of the bit is "0", the sign is positive. If the signal state of the bit is "1," the sign is negative. Bits 12 through 14 are not used in the conversion.

If the BCD code contains a BCD error, the CPU sets the OV and the OS bit in the status word.

The result of the conversion is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load BCD in accumulator 1.`
`BTI`	`// Convert BCD to 16-bit integer.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:

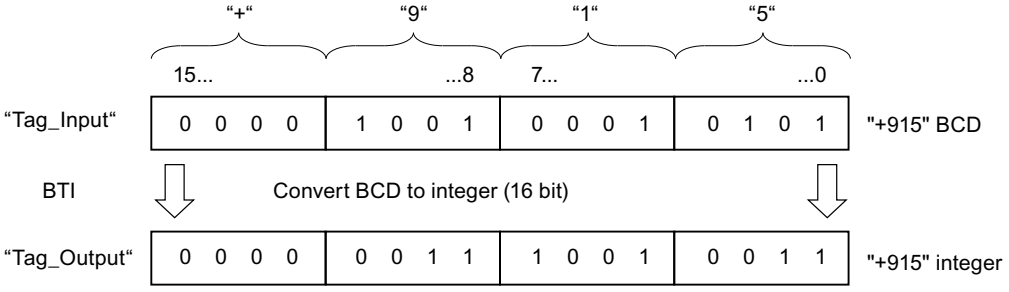

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

ITB: Convert integer (16 bit) to BCD (S7-1500)

Description

Use the "Convert integer (16 bit) to BCD" instruction to convert the value in the right word of accumulator 1 to a three-digit binary-coded decimal (BCD). The instruction interprets the value that is to be converted as a 16-bit integer.

The bits 0 to 11 of accumulator 1 indicate the value that will be converted. Values in the range from "-999" to "+999" are permitted. If the value to be converted is outside this range, the bits OV and OS are set to signal state "1". In this case the conversion is not executed.

The sign of the result value is read from bits 12 through 15 of accumulator 1. If the signal state of the bits is "0", the sign is positive. If the signal state of all four bits is "1", the sign is negative.

The result of the conversion is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load integer (16 bit) in accumulator 1.`
`ITB`	`// Convert integer to BCD.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:

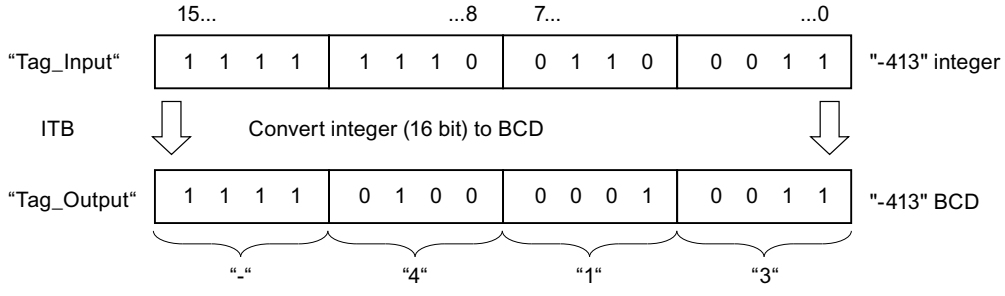

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

BTD: Convert BCD to integer (32 bit) (S7-1500)

Description

You can use the "Convert BCD to integer (32 bit)" to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a seven-digit binary coded binary coded decimal (BCD).

The bits 0 to 27 of accumulator 1 indicate the value that will be converted. Values in the range from "-9999999" through "+9999999" are permitted.

The sign of the result value is read from bit 31 of accumulator 1. If the signal state of the bit is "0", the sign is positive. If the signal state of the bit is "1," the sign is negative. Bits 28 through 30 are not used in the conversion.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

If the BCD code contains a BCD error, the CPU sets the OV and the OS bit in the status word.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load BCD in accumulator 1.`
`BTD`	`// BCD is converted to 32-bit integer.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:

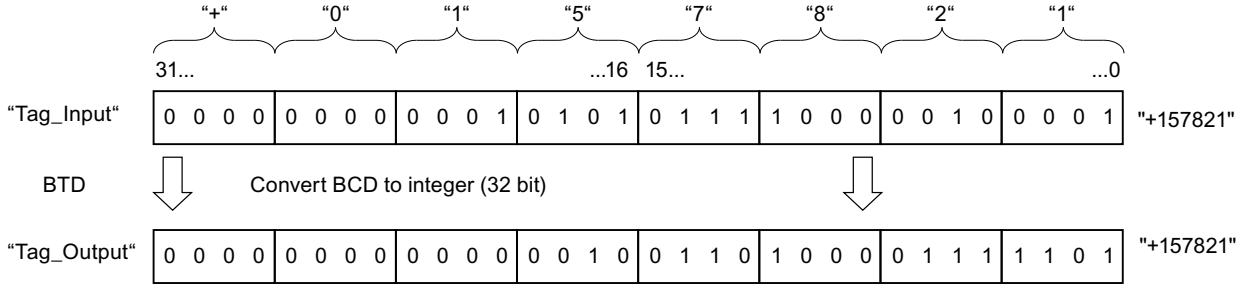

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

ITD: Convert integer (16 bit) to integer (32 bit) (S7-1500)

Description

Use the "Convert integer (16 bit) to integer (32 bit)" instruction to convert the value in the right word of accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a 16-bit integer.

Bits 0 through 15 are not used in the conversion. The bits 16 through 31 of the result value are filled with the signal state of the bit 15.

The result value is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load integer (16 bit) in accumulator 1.
ITD                          // Convert integer (16 bit) to integer (32 bit).
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Accumulator 1							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Before processing	XXXX	XXXX	XXXX	XXXX	1111	1111	1111	0110
After processing	1111	1111	1111	1111	1111	1111	1111	0110
	X: The signal state of the bits is irrelevant for the conversion							

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

DTB: Convert integer (32 bit) to BCD (S7-1500)

Description

Use the "Convert integer (32 bit) to BCD" instruction to convert the value in accumulator 1 to a seven-digit binary coded decimal (BCD). The instruction interprets the value that is to be converted as a 32-bit integer.

The bits 0 to 27 of accumulator 1 indicate the value that will be converted. Values in the range from "-9999999" through "+9999999" are permitted. If the value to be converted is outside this range, the bits OV and OS are set to signal state "1". In this case the conversion is not executed.

The sign of the result value is read from bits 28 through 31 of accumulator 1. If the signal state of the bits is "0", the sign is positive. If the signal state of all four bits is "1", the sign is negative.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load integer (32 bit) to accumulator 1`
`DTB`	`// Convert integer (32 bit) to BCD`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:

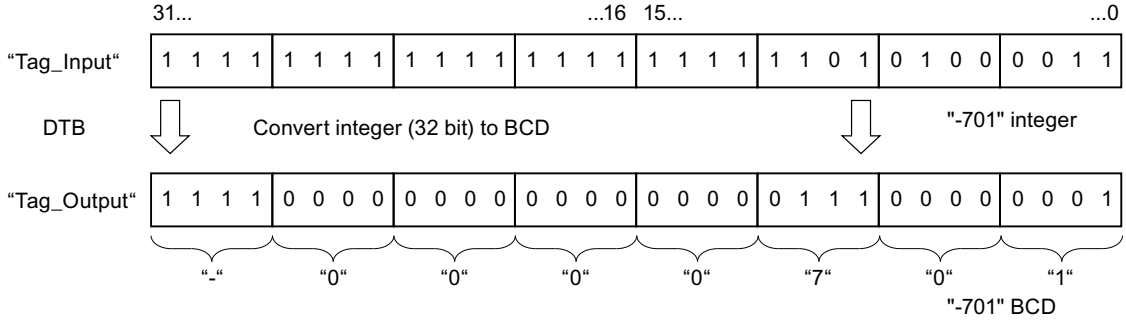

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

DTR: Convert integer (32 bit) to floating-point number (S7-1500)

Description

Use the "Convert integer (32 bit) to floating-point number" instruction to convert the value in accumulator 1 to a floating-point number. The instruction interprets the value that is to be converted as a 32-bit integer.

Because a 32-bit integer has a higher accuracy than a floating-point number, the result of the conversion is rounded off to the next representable number.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load integer (32 bit) to accumulator 1`
`DTR`	`// Convert integer (32 bit) to floating-point number.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:

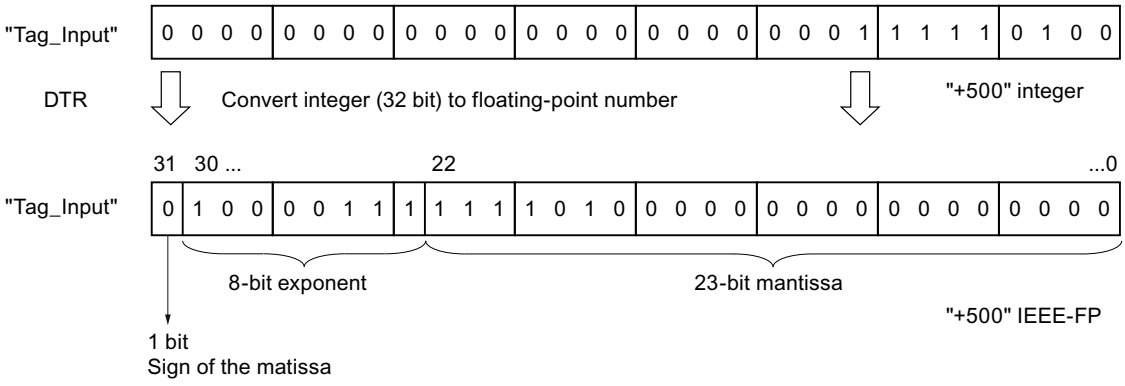

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

INVI: Create ones complement integer (16 bit) (S7-1500)

Description

Use the "Create ones complement integer (16 bit)" instruction to negate the value in the right word of accumulator 1 bit for bit.

When executed, the instruction inverts the signal state of each individual bit in the right word of accumulator 1. The ones are replaced with zeros and the zeros with ones.

The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value in accumulator 1
INVI                         // Create ones complement
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0110	0011	1010	1110
Tag_Output	1001	1100	0101	0001

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

INVD: Create ones complement double integer (32 bit) (S7-1500)

Description

Use the "Create ones complement double integer (32 bit)" to negate the value in accumulator 1 bit for bit.

During execution, the instruction inverts the signal state of each individual bit in accumulator 1. The ones are replaced with zeros and the zeros with ones.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value in accumulator 1
INVD                         // Create ones complement
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0110	1111	1000	1100	0110	0011	1010	1110
Tag_Output	1001	0000	0111	0011	1001	1100	0101	0001

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

NEGI: Negate integer (16 bit) (S7-1500)

Description

Use the "Negate integer (16 bit)" instruction to change the sign of the value in the right word of accumulator 1 by forming the twos complement. The instruction interprets the value of accumulator 1 as a 16-bit integer.

The processing of the instruction is equal to a multiplication with "-1". The result of the instruction is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

The "Negate integer (16 bit)" instruction affects the status bits CC 0, CC 1, OV and OS. The following table shows how the instruction influences the status bit depending on the result:

Result:	CC 1	CC 0	OV	OS
+1 to +32 767	1	0	0	-
0	0	0	0	-

Result:	CC 1	CC 0	OV	OS
-1 to -32 767	0	1	0	-
(-) 32 768	0	1	1	1

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value in accumulator 1
NEGI                         // Negate integer (16 bit)
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0101	1101	0011	1000
Tag_Output	1010	0010	1100	1000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

NEGD: Negate integer (32 bit) (S7-1500)

Description

Use the "Negate integer (32 bit)" instruction to change the sign of the value in accumulator 1 by forming the twos complement. The instruction interprets the value of accumulator 1 as a 32-bit integer.

The processing of the instruction is equal to a multiplication with "-1". The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The "Negate integer (32 bit)" instruction affects the status bits CC 0, CC 1, OV and OS. The following table shows how the instruction influences the status bit depending on the result:

Result	CC 1	CC 0	OV	OS
+1 to +2147483647	1	0	0	-
0	0	0	0	-

Result	CC 1	CC 0	OV	OS
-1 to -2147483647	0	1	0	-
(-) 2147483648	0	1	1	1

Example

The following example shows how the instruction works:

```
STL                         Explanation
L "Tag_Input"               // Load value in accumulator 1
NEGD                        // Negate integer (32 bit).
T "Tag_Output"              // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Status	Value							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0101	1111	0110	0100	0101	1101	0011	1000
Tag_Output	1010	0000	1001	1011	1010	0010	1100	1000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

NEGR: Negate floating-point number (S7-1500)

Description

The instruction "Negate floating-point number" inverts the sign of the mantissa and is equal to a multiplication by "-1".

The instruction interprets the value of accumulator 1 as a floating-point number. The sign of the mantissa is in bit 31 of the floating-point number.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The instruction does not affect any status bits.

Example

The following example shows how the instruction works:

```
STL                         Explanation
L "Tag_Input"               // Load floating-point number into accumulator 1
NEGR                        // Negate floating-point number.
T "Tag_Output"              // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Input	1.5E+02
Tag_Output	-1.5E+02

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

CAW: Switch bytes in the right word of accumulator 1 (S7-1500)

Description

Use the "Switch bytes in the right word of accumulator 1" instruction to switch the sequence of the two right bytes in the right word of accumulator 1.

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Bytes in accumulator 1			
Before processing	Value A	Value B	Value C	Value D
After processing	Value A	Value B	Value D	Value C

The result of the instruction is saved in the right word of accumulator 1. The bytes in the left word of accumulator 1 are not influenced by the instruction and remain unchanged.

Example

The following example shows how the instruction works:

```
STL                         Explanation
L "Tag_Input"               // Load value in accumulator.
```

```
STL                          Explanation
CAW                          // Switch bytes in the right word of accumulator 1.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Input	0000	1111	0000	1111
Tag_Output	0000	1111	1111	0000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

CAD: Switch all bytes in accumulator 1 (S7-1500)

Description

Use the "Switch all bytes in accumulator 1" instruction to switch the sequence of the bytes in accumulator 1.

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Bytes in accumulator 1			
Before processing	Value A	Value B	Value C	Value D
After processing	Value D	Value C	Value B	Value A

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value of the operand in accumulator 1.
CAD                          // Switch bytes of accumulator 1.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Input	1111	0000	0000	1111	0000	0000	1111	1111
Tag_Output	1111	1111	0000	0000	0000	1111	1111	0000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

RND: Round numerical value (S7-1500)

Description

Use the "Round numerical value" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and round this to the next integer.

If the floating-point number is exactly between an even and odd number, the even number is selected as result.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load floating-point number into accumulator 1
RND                          // Round floating-point number to an integer.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	101.5	-101.5
Tag_Output	102	-102

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

TRUNC: Truncate numerical value (S7-1500)

Description

Use the "Truncate numerical value" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value to be converted as a floating-point number and returns its whole number as result.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load floating-point number into accumulator 1
TRUNC                        // Convert floating-point number to an integer of data
                             type DINT.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	101.5	-101.5
Tag_Output	101	-101

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

RND+: Generate next higher integer from floating-point number (S7-1500)

Description

Use the "Generate next higher integer from floating-point number" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and rounds this number to the next higher integer. The result of the instruction is a number of the data type DINT that is greater than or equal to the floating-point number to be converted.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Example

The following example shows how the instruction works:

```
STL                         Explanation
L "Tag_Input"               // Load floating-point number into accumulator 1
RND+                        // Round floating-point number to next higher integer.
T "Tag_Output"              // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	100.5	-100.5
Tag_Output	101	-100

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

RND-: Generate next lower integer from floating-point number (S7-1500)

Description

Use the "Generate next lower integer from floating-point number" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and rounds this number to the next lower integer. The result of the instruction is a number of the data type DINT that is less than or equal to the floating-point number to be converted.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load floating-point number into accumulator 1
RND-                         // Round floating-point number to next lower integer.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	100.5	-100.5
Tag_Output	100	-101

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

Program control operations (S7-1500)

Jumps (S7-1500)

Jump label (S7-1500)

Description

Use a jump label to specify the place in the program at which the program execution should resume after a jump. The name of the jump label can consist of up to 128 letters, numbers or underscores.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block. Each jump label can jump to several locations. You can declare a maximum of 256 jump labels.

Both forward and backward jumps are possible.

Adhere to the following grammatical rules for jump labels:

- Letters (a to z, A to Z)
- A combination of letters and numbers. Check that the order is correct, i.e. first the letters, then the numbers (a - z, A - Z, 0 - 9).
- Special characters or a combination of letters and numbers must not be used in reverse order, i.e. first the numbers and then the letters (0-9, a - z, A - Z).

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".
JC MyLABEL	// With a scan result "1" jump to jump label "MyLABEL" and continue with program processing // If the result of the query is "0", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output"	// Load contents of accumulator 1 in operand "Tag_Output".

STL	Explanation
MyLABEL: A "Tag_Input_4"	// If there was a jump, continue program execution at this point. // Query whether operand "Tag_Input_4" is "1"

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JU: Unconditional jump (S7-1500)

Description

Use the "Unconditional jump" instruction to interrupt the linear program execution and continue at the point that is marked by the specified jump label. The instruction is always executed independently of conditions.

The "Unconditional jump" instruction does not affect the status bits.

Parameters

The following table shows the parameters of the "Unconditional jump" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".
JC MyLABEL_1	// With a scan result "1" jump to jump label "MyLABEL_1" and continue with program processing // If the result of the query is "0", continue linear program execution.

STL	Explanation
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output_1"	// Load contents of accumulator 1 in operand "Tag_Output_1".
JU MyLABEL_2	// Jump to jump label "MyLABEL_2" and resume execution of the program there.
MyLABEL_1:	// Jump label
L "Tag_Input_4"	// Load content of the operand "Tag_Input_4" in accumulator 1.
T "Tag_Output_2"	// Load contents of accumulator 1 in operand "Tag_Output_2".
MyLABEL_2: A "Tag_Input_5"	// Jump label "MyLABEL_2"
	// Query whether operand "Tag_Input_5" is "1"

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL programming examples (Page 7280)

JC: Jump if RLO = 1 (S7-1500)

Description

Use the "Jump if RLO = 1" instruction to interrupt linear program execution depending on the result of logic operation and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed when the current RLO is "1". If the current RLO before the instruction is "0", the jump is not executed and the program is resumed with the next instruction.

The "Jump if RLO = 1" instruction sets the RLO to "1" both if the condition is satisfied or not satisfied.

Parameters

The following table shows the parameters of the "Jump if RLO = 1" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".
JC MyLABEL	// If RLO = "1", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "0", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output_1"	// Load contents of accumulator 1 in operand "Tag_Output_1".
MyLABEL: L "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Load content of the operand "Tag_Input_4" in accumulator 1.
T "Tag_Output_2"	// Transfer contents of accumulator 1 to the operand "Tag_Output_2".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL programming examples (Page 7280)

JCN: Jump if RLO = 0 (S7-1500)

Description

Use the "Jump if RLO = 0" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed when the current RLO is "0". If the current RLO before the instruction is "1", the jump is not executed and the program is resumed with the next instruction.

The "Jump if RLO = 0" instruction sets the RLO to "1" both if the condition is satisfied or not satisfied.

Parameters

The following table shows the parameters of the "Jump if RLO = 0" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input_1"              // Query whether the value of operand "Tag_Input_1" is
                             "1" and AND.
A "Tag_Input_2"              // Query whether the value of operand "Tag_Input_2" is
                             "1" and AND.
JCN MyLABEL                  // If RLO = "0", jump to jump label "MyLABEL" and con-
                             tinue program execution there.
                             // If the RLO = "1", execute the next instruction.
L "Tag_Input_3"              // Load content of the operand "Tag_Input_3" in accu-
                             mulator 1.
T "Tag_Output"               // Load contents of accumulator 1 in operand "Tag_Out-
                             put".
MyLABEL: A "Tag_Input_4"     // If there was a jump, continue program execution at
                             this point.
                             // Query whether the value of operand "Tag_Input_4" is
                             "1" and AND with the RLO
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL programming examples (Page 7280)

JCB: Jump if RLO = 1 and save RLO (S7-1500)

Description

Use the "Jump if RLO = 1 and save RLO" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label. At the same time, the signal state of the current RLO is copied to the binary result (BR).

The jump to the specified jump label is only executed when the current RLO is "1". In this case the instruction set the binary result to "1".

If the current RLO before the instruction is "0", the jump is not executed and the program is resumed with the next instruction. In this case the instruction assigns the signal state "0" to the binary result.

The "Jump if RLO = 1 and save RLO" instruction sets the RLO to "1" both if the condition is satisfied or not satisfied.

Parameters

The following table shows the parameters of the "Jump if RLO = 1 and save RLO" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input_1"`	`// Load first comparison value.`
`L "Tag_Input_2"`	`// Load second comparison value.`
`>I`	`// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".`
`JCB MyLABEL`	`// Copy current RLO to BR.`
	`// If RLO = "1", jump to jump label "MyLABEL" and continue program execution there.`
	`// If the RLO = "0", execute the next instruction.`
`L "Tag_Input_3"`	`// Load content of the operand "Tag_Input_3" in accumulator 1.`
`T "Tag_Output_1"`	`// Load contents of accumulator 1 in operand "Tag_Output_1".`
`MyLABEL: L "Tag_Input_4"`	`// If there was a jump, continue program execution at this point.`
	`// Load content of the operand "Tag_Input_4" in accumulator 1.`
`T "Tag_Output_2"`	`// Load contents of accumulator 1 in operand "Tag_Output_2".`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JNB: Jump if RLO = 0 and save RLO (S7-1500)

Description

Use the "Jump if RLO = 0 and save RLO" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label. At the same time, the signal state of the current result of logic operation is copied to the binary result (BR).

The jump to the specified jump label is only executed when the current RLO is "0". In this case the instruction sets the binary result to "0".

If the current RLO before the instruction is "1", the jump is not executed and the program is resumed with the next instruction. In this case the instruction assigns the signal state "1" to the binary result.

The "Jump if RLO = 0 and save RLO" instruction sets the RLO to "1" both if the condition is satisfied or not satisfied.

Parameters

The following table shows the parameters of the "Jump if RLO = 0 and save RLO" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query whether the value of operand "Tag_Input_1" is "1" and AND.
A "Tag_Input_2"	// Query whether the value of operand "Tag_Input_2" is "1" and AND.
JNB MyLABEL	// Copy current RLO to BR.
	// If RLO = "0", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "1", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output"	// Transfer contents of accumulator 1 to the operand "Tag_Output".

STL	Explanation
MyLABEL: A "Tag_Input_4"	// If there was a jump, continue program execution at this point. // Query whether the value of operand "Tag_Input_4" is "1" and AND with the RLO

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JBI: Jump if BR = 1 (S7-1500)

Description

Use the "Jump if BR = 1" instruction to interrupt linear program execution depending on the binary result and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit BR is "1". If the status bit BR is "0", the jump is not executed and the program resumes with the next instruction.

Parameters

The following table shows the parameters of the "Jump if BR = 1" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
SET	// Set RLO at the start of block to "1"
SAVE	// Transfer signal state of the RLO to the BR bit.
....	// Any program
AN OV	// Query OV bit at end of block.
SAVE	// If there is an overflow, set BR bit to "0".
	// Set BR to "1" if there is no overflow.
JBI END	// If BR = "1", jump to jump label END.
	// If BR = "0", execute the next instruction.

```
STL                          Explanation
R "Tag_Output_1"             // Reset operand "Tag_Output_1" to "0".
END: S "Tag_Output_2"        // Set operand "Tag_Output_2" to "1"
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JNBI: Jump if BR = 0 (S7-1500)

Description

Use the "Jump if BR = 0" instruction to interrupt the linear program execution depending on the binary result (BR) and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit BR is "0". If the status bit BR before the instruction is "1", the jump is not executed and the program resumes with the next instruction.

Parameters

The following table shows the parameters of the "Jump if BR = 0" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
SET                          // Set RLO at the start of block to "1"
SAVE                         // Transfer signal state of the RLO to the BR bit.
....                         // Any program
AN OV                        // Query OV bit at end of block.
SAVE                         // If there is an overflow, set BR bit to "0".
                             // Set BR to "1" if there is no overflow.
JNBI END                     // If BR = "0", jump to jump label END.
                             // If BR = "1", execute the next instruction.
S "Tag_Output"               // Set operand "Tag_Output" to "1"
```

STL	Explanation
END: BE	// End block

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JO: Jump if OV = 1 (S7-1500)

Description

Use the "Jump if OV = 1" instruction to interrupt linear program execution depending on the status bit OV and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit OV is "1". The status bit OV is set if, for example, the result of a calculation is outside the permitted range or an invalid value is returned during the comparison of floating-point numbers.

If the status bit OV is "0", the jump is not executed and the program continues with the next instruction.

Parameters

The following table shows the parameters of the "Jump if OV = 1" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value
L "Tag_Value_2"	// Load value
*I	// Multiply values
JO OVER	// If the status bit OV has signal state "1", jump to jump label "OVER" and continue program execution there. // If the signal state of the status bit OV is "0", execute the next instruction.

STL	Explanation
T "Tag_Result"	// Transfer the result of the multiplication to the operand "Tag_Result".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
OVER: SET	// Continue program execution at this point if the condition (OV = "1") is satisfied.
	// Set RLO to signal state "1"
R "Tag_Output"	// Reset operand "Tag_Output" to "0".
NEXT: A "MyTag_1"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_1" is "1" and AND.
A "MyTag_2"	// Query whether the value of operand "MyTag_2" is "1" and AND.
S "Tag_Output_2"	// If RLO = 1, set the "Tag_Output_2" operand to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JOS: Jump if OS = 1 (S7-1500)

Description

Use the "Jump if OS = 1" instruction to interrupt linear program execution depending on the status bit OS and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit OS is "1". The status bit OS is always set when a number range overflow sets the status bit OV to "1". In contrast to status bit OV, the status bit OS remains set even if a result is subsequently in the permitted number range.

If the status bit OS is "0", the jump is not executed and the program continues with the next instruction.

Parameters

The following table shows the parameters of the "Jump if OS = 1" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value
L "Tag_Value_2"	// Load value
*I	// Multiply loaded values.
	// Store product in accumulator 1.
L "Tag_Value_3"	// Load value
+I	// Add loaded value to the product.
	// Store product in accumulator 1.
L "Tag_Value_4"	// Load value
-I	// Subtract loaded value from the calculated sum.
	// Store product in accumulator 1.
JOS OVER	// If there is an overflow in one of the three preceding instructions, the status bit OS is set to "1"
	// If the signal state of the status bit OS is "1", the program resumes at the jump label "OVER".
	// If the signal state of the status bit OS is "0", the next instruction is executed
T "Tag_Result"	// Transfer the result of the entire calculation to the operand "Tag_Result"
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
OVER: SET	// Continue program execution at this point if the condition (OV = "1") is satisfied.
	// Set RLO to signal state "1"
R "Tag_Output"	// Reset operand "Tag_Output" to "0".
NEXT: A "MyTag_1"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_1" is "1" and AND.
A "MyTag_2"	// Query whether the value of operand "MyTag_2" is "1" and AND.
S "Tag_Output_2"	// If RLO = 1, set the "Tag_Output_2" operand to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JZ: Jump if the result is zero (S7-1500)

Description

Use the "Jump if the result is zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bits CC 0 and CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, accumulator 1 has the value "0".
- A negative overflow occurs when executing the "Add integers (16 bit)" (+I) or "Add double integers (32 bit)" (+D) instruction.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- The content of accumulator 2 is equal to the content of accumulator 1 after the execution of a comparison instruction.
- The content of accumulator 1 is zero after the execution of a word logic operation.
- The value of the last shifted out bit is "0" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is zero" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load value of the operand "Tag_Value".
SRW 1                        // Shift the contents of the right word in accumulator
                             1 one place to the right.
JZ ZERO                      // If the signal state of the last bit shifted is "0",
                             jump to jump label "OVER" and continue with program
                             processing from there
                             // If the signal state of the last bit shifted is "1",
                             execute the next instruction.
L "MyTag_1"                  // Load value of the operand "MyTag_1" in accumulator
                             1
INC 1                        // Add right-hand byte of accumulator 1 with the value
                             "1".
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
```

STL	Explanation
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
ZERO: L "MyTag_2"	// Jump label "ZERO"
	// Load value of the operand "MyTag_2" in accumulator 1
INC 1	// Add right-hand byte of accumulator 1 with the value "1".
T "Tag_Result_2"	// Transfer the result to operand "Tag_Result_2".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JN: Jump if the result is not zero (S7-1500)

Description

Use the "Jump if the result is not zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal states of the status bits CC 0 and CC 1 are different. The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is not equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when executing a math instruction with integers (+I, -I, *I, /I, +D, -D, *D, /D, NEGI, NEGD).
- A positive or negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The content of accumulator 2 is not equal to the content of accumulator 1 after the execution of a comparison instruction.

- The content of accumulator 1 is not zero after the execution of a word logic operation.
- The value of the last shifted out bit is "1" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is not zero" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Valu_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
XOW                          // Apply EXCLUSIVE OR logic operation
JN NOZERO                    // If the value in accumulator 1 is not equal to zero,
                             the jump to the "NOZERO" jump label is performed and
                             program processing continues from there.
                             // If the value in accumulator 1 is equal to zero, the
                             next instruction will be processed.
AN "MyTag_1"                 // Query whether the value of operand "MyTag_1" is "0"
                             and AND.
S "Tag_Output_1"             // If RLO is "1", set the operand "Tag_Output_1" to
                             "1".
JU NEXT                      // Jump to jump label "NEXT" and resume execution of
                             the program there.
NOZERO: AN "MyTag_2"         // Jump label "NOZERO"
                             // Query whether the value of operand "MyTag_2" is "0"
                             and AND.
S "Tag_Output_2"             // If RLO is "1", set the operand "Tag_Output_2" to "1"
NEXT: A "MyTag_3"            // Jump label "NEXT"
                             // Query whether the value of operand "MyTag_3" is "1"
                             and AND.
A "MyTag_4"                  // Query whether the value of operand "MyTag_4" is "1"
                             and AND.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JP: Jump if the result is greater than zero (S7-1500)

Description

Use the "Jump if the result is greater than zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is executed only if the signal state of status bit CC 0 is "0" and CC 1 is "1". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is greater than zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (*I, /I, *D, /D).
- A positive overflow occurs when processing an arithmetic instruction with floating-point numbers.
- After the execution of a comparison instruction, the content of accumulator 2 is greater than the content of accumulator 1.
- The contents of accumulator 1 are not zero after the execution of a word logic operation.
- The value of the last shifted out bit is "1" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is greater than zero" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
```

STL	Explanation
-I	// Subtract value of the operand "Tag_Value_2" from the value of the operand "Tag_Value_1".
JP POSITIVE	// At a positive result value in accumulator "1" jump to jump label "POSITIVE" and continue program processing from there
	// If the result in accumulator 1 is negative, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO is "1", set the operand "Tag_Output_1" to "1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
POSITIVE: AN "MyTag_2"	// Jump label "POSITIVE"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO is "1", set the operand "Tag_Output_2" to "1".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JM: Jump if the result is less than zero (S7-1500)

Description

Use the "Jump if the result is less than zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is executed only if the signal state of status bit CC 0 is "1" and CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is less than zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (*I, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (+I, -I, +D, -D, NEGI, NEGD).
- A negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- After the execution of a comparison instruction, the content of accumulator 2 is less than the content of accumulator 1.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is less than zero" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
-I                           // Subtract value of the operand "Tag_Value_2" from
                             the value of the operand "Tag_Value_1".
JM NEGATIVE                  // At a negative result value in accumulator "1" jump
                             to jump label "NEGATIVE" and continue program process-
                             ing from there
                             // If the result in accumulator 1 is positive, execute
                             the next instruction.
AN "MyTag_1"                 // Query whether the value of operand "MyTag_1" is "0"
                             and AND.
S "Tag_Output_1"             // If RLO is "1", set the operand "Tag_Output_1" to
                             "1".
JU NEXT                      // Jump to jump label "NEXT" and resume execution of
                             the program there.
NEGATIVE: AN "MyTag_2"       // Jump label "NEGATIVE"
                             // Query whether the value of operand "MyTag_2" is "0"
                             and AND.
S "Tag_Output_2"             // If RLO is "1", set the operand "Tag_Output_2" to
                             "1".
```

STL	Explanation
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JPZ: Jump if the result is greater than or equal to zero (S7-1500)

Description

Use the "Jump if the result is greater than or equal to zero" instruction to interrupt linear program execution depending on the status bit CC 0 and continue it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bit CC 0 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is greater than or equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (*I, /I, *D, /D).
- A positive overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- After the execution of a comparison instruction, the content of accumulator 2 is greater than or equal to the content of accumulator 1.
- An instruction for word logic operation was executed.
- A shift instruction was executed.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is greater than or equal to zero" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
-I                           // Subtract value of the operand "Tag_Value_2" from
                             the value of the operand "Tag_Value_1".
JPZ REGULAR                  // At a positive result value or null in accumulator
                             "1" jump to jump label "REGULAR" and continue program
                             processing from there
                             // If the result in accumulator 1 is negative, execute
                             the next instruction.
AN "MyTag_1"                 // Query whether the value of operand "MyTag_1" is "0"
                             and AND.
S "Tag_Output_1"             // If RLO is "1", set the operand "Tag_Output_1" to
                             "1".
JU NEXT                      // Jump to jump label "NEXT" and resume execution of
                             the program there.
REGULAR: AN "MyTag_2"        // Jump label "REGULAR"
                             // Query whether the value of operand "MyTag_2" is "0"
                             and AND.
S "Tag_Output_2"             // If RLO is "1", set the operand "Tag_Output_2" to
                             "1".
NEXT: A "MyTag_3"            // Jump label "NEXT"
                             // Query whether the value of operand "MyTag_3" is "1"
                             and AND.
A "MyTag_4"                  // Query whether the value of operand "MyTag_4" is "1"
                             and AND.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JMZ: Jump if the result is less than or equal to zero (S7-1500)

Description

Use the "Jump if the result is less than or equal to zero" instruction to interrupt linear program execution depending on the status bit CC 1 and continue it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bit CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is less than or equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, *I, +D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (+I, -I, +D, -D, NEGI, NEGD).
- A negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- After the execution of a comparison instruction, the content of accumulator 2 is less than or equal to the content of accumulator 1.
- The value of accumulator 1 is equal to zero after processing a word logic.
- The value of the last shifted bit is equal to zero after processing a write instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is less than or equal to zero" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
-I                           // Subtract value of the operand "Tag_Value_2" from
                             the value of the operand "Tag_Value_1".
```

STL	Explanation
JMZ MyLABEL	// At a negative result value or null in accumulator "1" jump to jump label "MyLABEL" and continue program processing from there
	// If the result in accumulator 1 is positive, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO = "1", set the operand "Tag_Output_1" to "1"
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL: AN "MyTag_2"	// Jump label "MyLABEL"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO="1" set the operand "Tag_Output_2" to "1"
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JUO: Jump if the result is invalid (S7-1500)

Description

Use the "Jump if the result is invalid" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bit CC 0 and CC 1 is "1". The situation occurs when one of the following conditions applies:

- It is divided by zero with a mathematical instruction (/I, /D, MOD).
- When processing an arithmetic instruction with floating-point numbers an overflow occurs and the result value is an invalid floating-point number.
- When processing a comparative instruction with floating-point numbers an invalid floating-point number was used or occurred as a result value.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is invalid" instruction:

Parameters	Description
<Jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
/I                           // Divide value of the operand "Tag_Value_1" by value
                             of the operand "Tag_Value_2".
JUO ERROR                    // At a division by zero, jump to jump label "ERROR"
                             and continue with program processing from there
                             // Otherwise execute next instruction.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand
                             "Tag_Result".
A "MyTag_1"                  // Query whether the value of operand "MyTag_1" is "1"
                             and AND.
R "MyTag_1"                  // If RLO = "1", reset the operand "MyTag_1" to "0"
JU NEXT                      // Jump to jump label "NEXT" and resume execution of
                             the program there.
ERROR: AN "MyTag_1"          // Jump label "ERROR"
                             // Query whether the value of operand "MyTag_1" is "0"
                             and AND.
S "MyTag_1"                  // If RLO is "1", set the operand "MyTag_1" to "1".
NEXT: A "MyTag_3"            // Jump label "NEXT"
                             // Query whether the value of operand "MyTag_3" is "1"
                             and AND.
A "MyTag_4"                  // Query whether the value of operand "MyTag_4" is "1"
                             and AND.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

JL: Define jump list (S7-1500)

Description

Use the instruction "Define jump list" to program a list that consists of several entries of the instruction "Unconditional jump" (JU). The list begins directly after the instruction "Define jump list" and may contain up to 255 entries. The numbering of the jump functions in the list begins at zero. The jump list must be programmed without any gaps. The end of the jump list is marked by a jump label that is specified for the "Define jump list" instruction.

Which jump function of the list executes, depends on the value in the right byte of accumulator 1. If, for example, the accumulator 1 has the value "0", the first jump function executes. If the value in accumulator 1 is "1", the second jump function executes. If the value in accumulator 1 is greater than the number of list entries, the "Define jump list" instruction references the end of the list.

The "Jump distributor" instruction is processed independently of conditions and does not affect the status bits.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load jump number in accumulator 1.
JL END	// Start of the jump list
JU MyLABEL_1	// If the value is "0", execute in accumulator 1 and jump to jump label "MyLABEL_1".
JU MyLABEL_2	// If the value is "1", execute in accumulator 1 and jump to jump label "MyLABEL_2"
JU MyLABEL_3	// If the value is "2", execute in accumulator 1 and jump to jump label "MyLABEL_3".
END: L "MyTag_1"	// End of the jump list
	// Load content of the operand "MyTag_1" in accumulator 1.
L "MyTag_2"	// Load content of the operand "MyTag_2" in accumulator 1.
+I	// Add values
T "Tag_Output_2"	// Load contents of accumulator 1 in operand "Tag_Output_2".
MyLABEL_1: ...	// Jump label "MyLABEL_1"
....	// Any program
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL_2: ...	// Jump label "MyLABEL_2"
....	// Any program

STL	Explanation
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL_3: ...	// Jump label "MyLABEL_3"
....	// Any program
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
NEXT: ...	// Jump label "NEXT"

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL programming examples (Page 7280)

LOOP: Loop (S7-1500)

Description

Use the "Loop" instruction to program loops in the program.

The instruction interprets the right word of accumulator 1 as an unsigned 16-bit integer in the range from 0 to 65535. During execution the instruction first decrements the content of accumulator 1 by one. If the value in accumulator 1 after the decrement is not zero, the jump is executed at the specified jump label. If the value is zero, the jump is not executed and the processing of the program is resumed with the next instruction. The value in accumulator 1 therefore specifies the number of program loops that have to be executed. This number has to be saved in a loop counter.

The <number> operand contains the number of the loop cycles. The <loop counter> operand contains the number of the loop cycles still to be executed. At the end of the program loop, the content of the loop counter is loaded in accumulator 1 and decremented by one by the instruction. The jump to the jump label at the start of the program loop is executed if the value in the accumulator is not equal to zero after the decrement.

The "Loop" instruction does not affect any status bits.

> **WARNING**
>
> **Presetting the loop counter**
>
> If the loop counter is not preset or is preset with the value "0", the loop counter is decremented further and further into the negative, resulting in an endless loop that can change the CPU to STOP mode.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load number of time loop will execute into the right word of accumulator 1.
START: T "Tag_Counter"	// Start of the jump list
	// Transfer contents of accumulator 1 to the loop counter.
L "MyTag_1"	// Load value of the operand "MyTag_1".
L "MyTag_2"	// Load value of the operand "MyTag_2".
*D	// Multiply values
T "MyTag_1"	// Transfer the result of the multiplication to the operand "MyTag_1".
L "Tag_Counter"	// Load value of the loop counter into accumulator 1.
LOOP START	// Decrement value in accumulator 1 by one.
	// Jump to the start of the program loop if the value is not equal to zero.
	// If the value is zero, execute the next instruction.
L "MyTag_2"	// Load value of the operand "MyTag_2".
L 100	// Load the value 100.
>I	// Compare whether the value of the operand "MyTag_2" is greater than 200.
= "MyTag_3"	// Write comparison result to the operand "MyTag_3".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

Data blocks (S7-1500)

OPN: Open data block in DB register (S7-1500)

Description

You can open a global data block (DB) with the "Open data block in DB register" instruction. The number of the data block is transferred to the DB register. Subsequent DB commands access the relevant blocks depending on the content of the register.

The "Open data block in DB register" instruction is processed without any conditions and does not affect either the result of the logic operation or the accumulator contents.

Parameters

The following table shows the parameters of the "Open data block in DB register" instruction:

Operand	Declaration	Data block type	Description
<data block>	-	DB	Data block that will be opened.

Example

The following example shows how the instruction works:

STL	Explanation
`OPN "GlobalDataBlock"`	`// Open global data block and transfer the number of the block to the DB register.`
`L %DBW0`	`// Load data word DBW0 of the opened global data block into accumulator 1.`
`T "MyTag"`	`// Transfer contents of accumulator 1 to the operand "MyTag_1".`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

OPNDI: Open data block in DI register (S7-1500)

Description

You can open any data block with the "Open data block in DI register" instruction. The number of the data block is transferred to the DI register. Subsequent DI commands access the relevant blocks depending on the content of the register.

In the symbolic addressing of a local formal parameter from the block interface, you always access the data block whose number you specify when you call the block.

The "Open data block in DI register" instruction is processed without any conditions and does not affect either the result of the logic operation or the accumulator contents.

Parameters

The following table shows the parameters of the "Open data block in DI register" instruction:

Operand	Declaration	Data block type	Description
<data block>	-	DI	Data block that will be opened.

Example

The following example shows how the instruction works:

```
STL                                  Explanation
OPNDI "DataBlock"                    // Open data block and transfer the number of
                                     the data block to the DI register.
L %DIW0                              // Load data word DIW0 of the opened data
                                     block into accumulator 1.
T "MyTag"                            // Transfer contents of accumulator 1 to the
                                     operand "MyTag".
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

CDB: Swap data block register (S7-1500)

Description

Use the "Swap data block register" instruction to swap the contents of the data block register. The instruction executes regardless of conditions and does not affect the status bits.

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block and transfer the number of the data block to the DB register.
OPNDI "DataBlock"	// Open data block and transfer the number of the data block to the DI register.
CDB	// Swap data block register
	// The DB register points to "DataBlock" and the DI register points to "GlobalDataBlock".
L %DIW0	// Load DW0 from "GlobalDataBlock" into accumulator 1.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

L DBLG: Load the length of a global data block into accumulator 1 (S7-1500)

Description

With the "Load the length of a global data block into accumulator 1" instruction you load the length of a global data block opened by means of the data block register into accumulator 1. The previous content of accumulator 1 is moved to accumulator 2.

If no global data block was opened via the data block register prior to the processing of the instruction, the value "0" is loaded into accumulator 1. The length of the data block is equal to the number of the data bytes.

The status bits are not influenced by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block and transfer the number of the data block to the DB register.
L DBLG	// Load the length of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
<D	// Compare whether the length of the data block is less than the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

L DBNO: Load the number of a global data block into accumulator 1 (S7-1500)

Description

With the "Load the number of a global data block into accumulator 1" instruction you load the number of a global data block opened by means of the data block register into accumulator 1. The previous content of accumulator 1 is moved to accumulator 2.

If no global data block was opened via the data block register prior to the processing of the instruction, the value "0" is loaded into accumulator 1.

The status bits are not influenced by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block and transfer the number of the data block to the DB register.
L DBNO	// Load the number of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value

STL	Explanation
==I	// Compare whether the number of the data block is equal to the value of the operand "My-Tag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

L DILG: Load the length of an instance data block into accumulator 1 (S7-1500)

Description

With the "Load the length of an instance data block into accumulator 1" instruction you load the length of an instance data block opened by means of the data block register into accumulator 1. The previous content of accumulator 1 is moved to accumulator 2.

If no instance data block was opened via the data block register prior to the instruction executing, the value "0" is loaded into accumulator 1. The length of the data block is equal to the number of the data bytes.

The status bits are not influenced by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
OPNDI "InstanceDataBlock"	// Open data block and transfer the number of the data block to the DI register.
L DILG	// Load the length of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
<I	// Compare whether the length of the data block is less than the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

L DINO: Load the number of an instance data block into accumulator 1 (S7-1500)

Description

With the "Load the number of an instance data block into accumulator 1" instruction you load the number of an instance data block opened by means of the data block register into accumulator 1. The previous content of accumulator 1 is moved to accumulator 2.

If no instance data block was opened via the data block register prior to the instruction executing, the value "0" is loaded into accumulator 1.

The status bits are not influenced by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
`OPNDI "InstanceDataBlock"`	`// Open data block and transfer the number of the data block to the DI register.`
`L DINO`	`// Load the number of the opened data block into accumulator 1.`
`L "MyTag_1"`	`// Load comparison value`
`==I`	`// Compare whether the number of the data block is equal to the value of the operand "MyTag_1".`
`= "MyTag_2"`	`// Assign query result to the "MyTag_2" operand.`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

Code blocks (S7-1500)

BE: Block end (S7-1500)

Description

You can use the "Block end" instruction to end the block currently being processed and go to the point in the program where the block is called. The program processing is then resumed with the instruction that is located directly after the block call.

The "Block end" instruction is executed independent of conditions. If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SIN	// Form sine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand "Tag_Result".
BE	// End block

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

BEC: Conditional block end (S7-1500)

Description

You can use the "Conditional block end" instruction to end the block currently being processed depending on the result of logic operation (RLO) and to go to point in the program where the block is called.

If the RLO is "1", the instruction is executed. The block currently being processed is terminated and the program processing is resumed in the calling block. The program processing is then resumed with the instruction that is located directly after the block call.

If the RLO is "0" when the instruction is processed, the instruction is not executed. In this case the CPU sets the RLO to "1" and executes the next instruction.

If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load value of the operand "Tag_Value_1" in accumulator 1`
`L "Tag_Value_2"`	`// Shift the contents of accumulator 1 to accumulator 2.`
	`// Load value of the operand "Tag_Value_2" in accumulator 1`
`>I`	`// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.`
`A "Tag_Input"`	`// Query the value of operand "Tag_Input" for "1" and AND with the RLO.`
`BEC`	`// End block if condition satisfied (RLO = "1").`
	`// If the condition is not satisfied(RLO = "0"), execute the next instruction.`
`T "Tag_Result"`	`// Transfer contents of accumulator 1 to the operand "Tag_Result".`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

BEU: Unconditional block end (S7-1500)

Description

You can use the "Unconditional block end" instruction to end the block currently being processed and go to the point in the program where the block is called. The program processing is then resumed with the instruction that is located directly after the block call.

The "Unconditional block end" instruction is executed independent of conditions and can be programmed multiple times within a block. If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A "Tag_Input"	// Query the value of operand "Tag_Input" for "1" and AND with the RLO.
JC NEXT	// If the condition is satisfied (RLO = "1"), continue program execution at jump label"NEXT".
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
BEU	// End block
NEXT: T "Tag_Result"	// Jump label "NEXT"
	// Transfer contents of accumulator 1 to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL programming examples (Page 7280)

CALL: Call block (S7-1500)

Description

Use the "Call block" instruction to call the following block types in the program:

- Functions
- Function blocks

The "Call block" instruction executes without any conditions. After processing of the instruction, program execution is resumed in the called block.

You can feed a called block with data. The data is transferred via the block parameters. The parameters of the called block are listed according to the call instruction in the called block. You can assign the required current parameters to these parameters. During execution of the "Call block" instruction, the data is transferred to the called block. By transferring the data, the content of the status word and the address and data block registers are changed.

Those parameters of a function block that are not fed with parameters, retain their current value. All parameters must be fed when functions are called. If a called block has no parameters, no parameter list is displayed.

If the called block requires an instance data block, give it to the block, separated by a comma when called. The specified data block must be created before the call.

When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

Note

The "Conditional block call" instruction can only be used when the block property "Parameter passing via registers" is disabled.

Example

The following example shows how the instruction works:

```
STL                                        Explanation
CALL "MyFunction"                          // Call function "MyFunction"
      Input_1 := "Tag_Input_1",            // Assign actual parameters
      Input_2 := "Tag_Input_2"
      Output_1 := "Tag_Output_1"
      Output_2 := "Tag_Output_1"
CALL "MyFunctionBlock", "MyFB_DB"          // Call function block "MyFunctionBlock"
      Value_1 := "Tag_Value_1"             // Assign actual parameters
      Value_2 := "Tag_Value_2"
      Output := "Tag_Output"
```

```
STL                                      Explanation
CALL "LIMIT"                             // Call "Set limit value" instruction
      MN := "Tag_LowLimit"               // Assign actual parameters
      IN := "Tag_InputValue"
      MX := "Tag_HighLimit"
      OUT := "Tag_Output"
CALL "CTU", "CTU_DB"                     // Call "Count up" counter
      CU := "Tag_StartCTU"               // Assign actual parameters
      R := "Tag_ResetCounter"
      PV := "Tag_PresetValue"
      Q := "Tag_CounterStatus"
      CV := "Tag_CounterValue"
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

CC: Conditional block call (S7-1500)

Description

Use the "Conditional block call" instruction to call functions (FCs) and function blocks (FBs) depending on the result of logic operation (RLO).

The instruction is only executed if the current result of logic operation (RLO) before execution of the instruction is "1". After processing of the instruction, program execution is resumed in the called block. When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

If there is a block change, the status bit OS is reset to "0". The status bits CC 0, CC 1 and OV are affected depending on the calling block.

Note

The "Conditional block call" instruction can only be used when the block property "Parameter passing via registers" is enabled. However, this lowers performance.

The contents of the accumulators and the address registers are not changed by the "Conditional block call" instruction.

If the current RLO is "0", the instruction as well as the block call are not executed and the RLO is set to "1".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the value of operand "Tag_Input_1" for "1" and AND with the current RLO.
CC "MyFunction"	// If condition is satisfied (RLO = "1"), call the function MyFunction.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
A "Tag_Input_2"	// Query the value of operand "Tag_Input_2" for "1" and AND with the current RLO.
CC "MyFunctionBlock"	// If condition is satisfied (RLO = "1"), call the function block MyFunctionBlock.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Overview of block properties (Page 6921)

Parameter assignment via registers during block call in STL (Page 7247)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

UC: Unconditional block call (S7-1500)

Description

You can use the "Unconditional block call" instruction to call functions (FCs) and function blocks (FBs) that have no parameters and no instance data blocks.

The instruction is executed regardless of conditions. After processing of the instruction, program execution is resumed in the called block. When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

If there is a block change, the status bit OS is reset to "0". The status bits CC 0, CC 1 and OV are affected depending on the calling block.

Note

The "Unconditional block call" instruction can only be used when the block property "Parameter passing via registers" is enabled. However, this lowers performance.

The contents of the accumulators and the address registers are not changed by the "Unconditional block call" instruction.

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_Input_1"`	`// Query the value of operand "Tag_Input_1" for "1" and AND with the current RLO.`
`UC "MyFunction"`	`// Call function "MyFunction".`
`A "Tag_Input_2"`	`// Query the value of operand "Tag_Input_2" for "1" and AND with the current RLO.`
`UC "MyFunctionBlock"`	`// Call function block "MyFunctionBlock".`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Overview of block properties (Page 6921)

Parameter assignment via registers during block call in STL (Page 7247)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

Word logic operations (S7-1500)

AW: AND logic operation word by word (S7-1500)

Description

Use the instruction "AND logic operation word by word" to AND the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" if both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

The following table shows how the result is formed by an AND word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	0	0	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the "AND logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is ANDed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into the right word of
                             accumulator 1.
AW W#16#F6B5                 // AND the value of the right word in accumulator 1
                             with the constant (W#16#F6B5).
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into the right word of
                             accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" into the
                             right word of accumulator 1.
AW                           // AND the value of the right word in accumulator 2
                             ("Tag_Value_2") with the value of the right word in
                             accumulator 1 ("Tag_Value_3")
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	0101	0000	0011	0001
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	0100	1000	0000	0010

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Setting the status bit for word logic operations (Page 190)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL programming examples (Page 7280)

OW: OR logic operation word by word (S7-1500)

Description

Use the instruction "OR logic operation word by word" to OR the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both bits of the logic operation have signal state "0", the corresponding result bit is reset to "0".

The following table shows how the result is formed by an OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the "OR logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is ORed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into the right word of
                             accumulator 1.
OW W#16#F6B5                 // OR the value of the right word in accumulator 1
                             with the constant (W#16#F6B5).
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into the right word of
                             accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" into the
                             right word of accumulator 1.
OW                           // OR the value of the right word in accumulator 2
                             ("Tag_Value_2") with the value of the right word in
                             accumulator 1 ("Tag_Value_3")
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	1111	1111	1011	1111
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	1111	1110	1011	1011

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Setting the status bit for word logic operations (Page 190)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

STL programming examples (Page 7280)

XOW: EXCLUSIVE OR logic operation word by word (S7-1500)

Description

Use the instruction "EXCLUSIVE OR logic operation word by word" to EXCLUSIVE OR the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" if the signal state of the two bits to be linked is different. If both of the bits of the logic operation have the same signal state, the corresponding result bit is reset to "0".

The following table shows how the result is formed by an EXCLUSIVE OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	0

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is XORed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into the right word of
                             accumulator 1.
XOW W#16#F6B5                // XOR the value of the right word in accumulator 1
                             with the constant (W#16#F6B5).
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into the right word of
                             accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" into the
                             right word of accumulator 1.
XOW                          // XOR the value of the right word in accumulator 2
                             ("Tag_Value_2") with the value of the right word in
                             accumulator 1 ("Tag_Value_3")
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	1010	1111	1000	1110
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	1011	0110	1011	1001

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Setting the status bit for word logic operations (Page 190)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

AD: AND logic operation double word by double word (S7-1500)

Description

You can use the instruction "AND logic operation double word by double word" to AND the contents of accumulator 1 with the contents of accumulator 2 or with the value of a specified constant, bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" if both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

The following table shows how the result is formed by an AND word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	0	0	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the instruction "AND logic operation double word by double word":

Parameters	Data type	Description
<constant>	DWORD	Value that is ANDed with the value in accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
AD DW#16#39C657AC            // AND the value in accumulator 1 with the constant
                             (DW#16#39C657AC).
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in accumu-
                             lator 1.
```

STL	Explanation
AD	// AND the value of accumulator 2 Tag_Value_2 with the value of accumulator 1 ("Tag_Value_3")
	// Store result in accumulator 1.
T "Tag_Result_2"	// Transfer the result to operand "Tag_Result_2".

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39 C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Re-sult_1	0001	1001	0100	0100	0001	0101	0010	1000
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Re-sult_2	0010	0001	0100	0010	0101	0101	0010	1010

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Setting the status bit for word logic operations (Page 190)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

OD: OR logic operation double word by double word (S7-1500)

Description

You can use the instruction "OR logic operation double word by double word" to OR the contents of accumulator 1 with the contents of accumulator 2 or with the value of a specified constant, bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both bits of the logic operation have signal state "0", the corresponding result bit is reset to "0".

The following table shows how the result is formed by an OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the "OR logic operation double word by double word" instruction:

Parameters	Data type	Description
<constant>	DWORD	Value that is ORed with the value in accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
OD DW#16#39C657AC            // OR the value in accumulator 1 with the constant
                             (DW#16#39C657AC).
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in accumu-
                             lator 1.
OD                           // OR the value of accumulator 2 Tag_Value_2 with the
                             value of accumulator 1 ("Tag_Value_3")
                             // Store result in accumulator 1
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39 C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Re-sult_1	0111	1111	1110	0110	1101	1111	1011	1111

Operand	Value							
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Re-sult_2	0111	1101	1100	0111	0101	1111	1110	1111

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Setting the status bit for word logic operations (Page 190)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

XOD: EXCLUSIVE OR logic operation double word by double word (S7-1500)

Description

You can use the instruction "EXCLUSIVE OR logic operation double word by double word" to XOR the contents of accumulator 1 with the contents of accumulator 2 bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" if the signal state of the two bits to be linked is different. If both of the bits of the logic operation have the same signal state, the corresponding result bit is reset to "0".

The following table shows how the result is formed by an EXCLUSIVE OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	0

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation double word by double word" instruction:

Parameters	Data type	Description
<constant>	DWORD	Value that is XORed with the value in accumulator 1

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
XOD DW#16#39C657AC           // XOR the value in accumulator 1 with the constant
                             (DW#16#39C657AC).
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in accumu-
                             lator 1.
XOD                          // XOR the value of accumulator 2 Tag_Value_2 with the
                             value of accumulator 1 ("Tag_Value_3")
                             // Store result in accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39 C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Re-sult_1	0110	0110	1010	0010	1100	1010	1001	0111
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Re-sult_2	0101	1100	1000	0101	0000	1010	1000	0111

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Setting the status bit for word logic operations (Page 190)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

Shift and rotate (S7-1500)

Shift (S7-1500)

SSI: Shift word by word with sign (S7-1500)

Description

Use the instruction "Shift word by word with sign" to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the right. During the shifting, positions that become free are padded with the signal state of bit 15 (sign bit for INT numbers). The bits 16 to 31 of accumulator 1 remain unchanged. The instruction is executed regardless of the RLO. The status bit CC1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SSI <number>: The number of shifts is specified by the operand <number>. Values between 0 and 15 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SSI: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with the signal state of bit 15.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SSI 6                        // Shift bits 0 to 15 of accumulator 1 six places to
                             the right.
                             // Fill the now empty bit places with the state of bit
                             15.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 3                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SSI                          // Shift bits 0 to 15 of accumulator 1 three places to
                             the right.
                             // Fill the now empty bit places with the state of bit
                             15.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1111	1110	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	0000	1011	1010	0101

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

SSD: Shift double word by double word with sign (S7-1500)

Description

Use the instruction "Shift double word by double word with sign" to shift the complete contents of accumulator 1 bit by bit to the right. During the shifting, positions that become free are padded with the signal state of bit 31 (sign bit for DINT numbers). The instruction is executed regardless of the RLO. The status bit CC1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SSD <number>: The number of shifts is specified by the operand <number>. Values between 0 and 31 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SSD: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  For values greater than 31, all bits of accumulator 1 are padded with the signal state of bit 31.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Parameters

The following table shows the parameters of the instruction "Shift double word by double word with sign":

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT,	Number of bit places by which the value will be shifted.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
```

STL	Explanation
SSD 7	// Shift bits 0 to 31 of accumulator 1 seven places to the right.
	// Fill the now empty bit places with the state of bit 31.
T "Tag_Result_1"	// Transfer contents of accumulator 1 to the operand.
L 4	// Load number of places shifted into accumulator 1
L "Tag_Value_2"	// Move number of places shifted to the right byte of accumulator 2.
	// Load value of the operand in accumulator 1.
SSD	// Shift bits 0 to 31 of accumulator 1 four places to the right.
	// Fill the now empty bit places with the state of bit 31.
T "Tag_Result_2"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	1000	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1111	1111	0001	1110	1100	1000	1011	1010
Tag_Value_2	0010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	0010	1000	1010	0010	1001	1011	1100

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

SLW: Shift left word by word (S7-1500)

Description

Use the "Shift left word by word" instruction to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the left. The bit places that become free are padded with zeros. The bits 16 to 31 of accumulator 1 remain unchanged. The instruction is executed regardless of the RLO. The status bit CC1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SLW <Number>: The number of shifts is specified by the operand <number>. Values between 0 and 15 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SLW: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Parameters

The following table shows the parameters of the "Shift left word by word" instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SLW 5                        // Shift bits 0 to 15 of accumulator 1 five places to
                             the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SLW                          // Shift bits 0 to 15 of accumulator 1 four places to
                             the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1010	0111	0110	0000

Operand	Value							
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	1101	0010	1011	0000

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

SRW: Shift right word by word (S7-1500)

Description

Use the "Shift right word by word" instruction to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the right. The bit places that become free are padded with zeros. The bits 16 to 31 of accumulator 1 remain unchanged. The instruction is executed regardless of the RLO. The status bit CC1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SRW <Number>: The number of shifts is specified by the operand <number>. Values between 0 and 15 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SRW: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Parameters

The following table shows the parameters of the "Shift right word by word" instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SRW 6                        // Shift bits 0 to 15 of accumulator 1 six places to
                             the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SRW                          // Shift bits 0 to 15 of accumulator 1 four places to
                             the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Re-sult_1	0101	1111	0110	0100	0000	0001	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Re-sult_2	0101	1111	0110	0100	0000	0101	1101	0010

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

SLD: Shift left double word by double word (S7-1500)

Description

Use the "Shift left double word by double word" instruction to shift the complete contents of accumulator 1 bit by bit to the left. The bit places that become free are padded with zeros. The instruction is executed regardless of the RLO. The status bit CC1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SLD <number>: The number of shifts is specified by the operand <number>. Values between 0 and 31 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SLD: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  For values greater than 31, all bits of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Parameters

The following table shows the parameters of the "Shift left double word by double word" instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SLD 5                        // Shift bits 0 to 31 of accumulator 1 five places to
                             the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SLD                          // Shift bits 0 to 31 of accumulator 1 four places to
                             the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1110	1100	1000	1011	1010	0111	0110	0000
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	1000	1010	0010	1001	1011	1100	1101	0000

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

SRD: Shift right double word by double word (S7-1500)

Description

Use the instruction "Shift right double word by double word" to shift the complete contents of accumulator 1 bit by bit to the right. The bit places that become free are padded with zeros. The instruction is executed regardless of the RLO. The status bit CC1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SRD <number>: The number of shifts is specified by the operand <number>. Values between 0 and 31 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SRD: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  For values greater than 31, all bits of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Parameters

The following table shows the parameters of the "Shift right double word by double word" instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SRD 7                        // Shift bits 0 to 31 of accumulator 1 seven places to
                             the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SRD                          // Shift bits 0 to 31 of accumulator 1 four places to
                             the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0000	0000	1011	1110	1100	1000	1011	1010
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	1010	1000	1010	0010	1001	1011	1100

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

Rotate (S7-1500)

RLD: Rotate left double word by double word (S7-1500)

Description

Use the "Rotate left double word by double word" instruction to rotate the complete contents of accumulator 1 bit by bit to the left. During processing the instruction shifts bit 0 to 31 of accumulator 1 bit by bit to the left and pads the places that become free during writing with the pushed out bit places. The instruction is executed regardless of the RLO. The status bit CC1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be rotated:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- RLD <number>: The number of rotations is specified by the operand <number>. Values between 0 and 32 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- RLD: The number of rotations is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  For values greater than 32, the rotation number is calculated with a modulo division by 32. If, for example, the rotate number is 34, the bits of accumulator 1 are rotated by 2 bit places.

If the specified rotate number is 32, the contents of accumulator 1 remain unchanged.

If the specified rotate number is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Parameters

The following table shows the parameters of the "Rotate left double word by double word" instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places to be rotated.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
RLD 4                        // Rotate bits 0 to 31 of accumulator 1 four places to
                             the left
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 6                          // Load number of places to be rotated in accumulator
                             1.
L "Tag_Value_2"              // Move number of places to be rotated to right byte
                             of accumulator 2
                             // Load value of the operand in accumulator 1.
RLD                          // Rotate bits 0 to 31 of accumulator 1 six places to
                             the left
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	**0101**	1111	0110	0100	0101	1101	0011	1011
Tag_Re-sult_1	1111	0110	0100	0101	1101	0011	1011	**0101**
Tag_Value_2	**1010**	**10**00	1010	0010	1001	1011	1100	1101
Tag_Re-sult_2	0010	1000	1010	0110	1111	0011	01**10**	**1010**

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

RRD: Rotate right double word by double word (S7-1500)

Description

Use the "Rotate right double word by double word" instruction to rotate the complete contents of accumulator 1 bit by bit to the right. During processing the instruction shifts bits 0 to 31 of accumulator 1 bit by bit to the right and pads the places that become free during writing with the pushed out bit places. The instruction is executed regardless of the RLO. The status bit CC1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be rotated:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- RRD <number>: The number of rotations is specified by the operand <number>. Values between 0 and 32 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- RRD: The number of rotations is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  For values greater than 32, the rotation number is calculated with a modulo division by 32. If, for example, the rotate number is 34, the bits of accumulator 1 are rotated by 2 bit places.

If the specified rotate number is 32, the contents of accumulator 1 remain unchanged.

If the specified rotate number is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Parameters

The following table shows the parameters of the instruction "Rotate right double word by double word":

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places to be rotated.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
RRD 4                        // Rotate bits 0 to 31 of accumulator 1 four places to
                             the right.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 6                          // Load number of places to be rotated in accumulator
                             1.
L "Tag_Value_2"              // Move number of places to be rotated to right byte
                             of accumulator 2
                             // Load value of the operand in accumulator 1.
RRD                          // Rotate bits 0 to 31 of accumulator 1 six places to
                             the right.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	**1011**
Tag_Result_1	**1011**	0101	1111	0110	0100	0101	1101	0011
Tag_Value_2	1010	1000	1010	0010	1001	1011	11**00**	**1101**
Tag_Result_2	**0011**	**01**10	1010	0010	1000	1010	0110	1111

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

RLDA: Rotate left by status bit CC 1 (S7-1500)

Description

You can use the "Rotate left by status bit CC 1" instruction to shift the contents of accumulator 1 one bit to the left. The bit places (0) that become free during shifting are padded with the signal state of status bit CC 1. Status bit CC 1 receives the signal state of the shifted out bit (31).

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0".

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value"`	`// Load value of the operand in accumulator 1.`
`RLDA`	`// Rotate bits 0 to 31 of accumulator 1 one place to the left`
	`// Fill bit 0 with the signal state of status bit CC 1.`
	`// Write the shifted-out bit to the status bit CC 1.`
`T "Tag_Result"`	`// Transfer contents of accumulator 1 to the operand.`

The following table shows how the instruction works using specific operand values:

Operand	CC 1	Value							
Tag_Value	**1**	**0**101	1111	0110	0100	0101	1101	0011	1011
Tag_Result	**0**	1011	1110	1100	1000	1011	1010	0111	011**1**

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

RRDA: Rotate right by status bit CC 1 (S7-1500)

Description

Use the "Rotate right by status bit CC 1" instruction to shift the contents of accumulator 1 one bit to the right. The bit places (31) that become free during shifting are padded with the signal state of status bit CC 1. Status bit CC 1 receives the signal state of the shifted out bit (0).

The instruction is executed regardless of the RLO. It resets the status bits CC0 and OV to "0".

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value"`	`// Load value of the operand in accumulator 1.`
`RRDA`	`// Rotate bits 0 to 31 of accumulator 1 one place to the right`
	`// Fill bit 31 with the signal state of status bit CC 1.`
	`// Write the shifted-out bit to the status bit CC 1.`
`T "Tag_Result"`	`// Transfer contents of accumulator 1 to the operand.`

The following table shows how the instruction works using specific operand values:

Operand	CC 1	Value							
Tag_Value	1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result	0	1010	1111	1011	0010	0010	1110	1001	1101

Status bit CC 1 has the value "1" because the bit 0 of the "Tag_Value" operand also has the value "1".

See also

Overview of the valid data types (Page 231)

Querying and setting status bits in STL (Page 187)

STL Basics (Page 7245)

Additional instructions (S7-1500)

Accumulator (S7-1500)

TAK: Swap content of accumulators 1 and 2 (S7-1500)

Description

Use the instruction "Swap content of accumulators 1 and 2" to switch the contents accumulator 1 with the contents of accumulator 2.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
JC NEXT	// If condition is satisfied (RLO = "1"), jump to jump label NEXT.
	// If the condition is not satisfied (RLO = "0"), execute the next instruction.
TAK	// Swap the contents of accumulators 1 and 2.
NEXT: -I	// Jump label "NEXT"
	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Output"	// Transfer contents of accumulator 1 to the operand "Tag_Output".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

PUSH: Shift content to the next higher accumulator (S7-1500)

Description

Use the instruction "Shift content to the next higher accumulator" to move the contents of accumulator 1 to accumulator 2.

The content of accumulator 1 is not influenced by the instruction and remains unchanged after processing the instruction. The content of accumulator 2 will be lost.

The following table shows the contents of accumulators 1 to 2 before and after the executing the instruction:

Status	Accumulator	
	1	2
Before processing	Value A	Value B
After processing	Value A	Value A

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
PUSH                         // Shift contents of accumulators 1 to 2 to the next
                             higher accumulator.
L "Tag_Value_2"              // Load value of the operand in accumulator 1.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

POP: Shift content to the next lower accumulator (S7-1500)

Description

Use the instruction "Shift content to the next lower accumulator" to move the contents of accumulator 2 to accumulator 1.

The content of accumulator 2 is not influenced by the instruction and remains unchanged after processing the instruction. The content of accumulator 1 will be lost.

The following table shows the contents of accumulators 1 to 2 before and after the executing the instruction:

Status	Accumulator	
	1	2
Before processing	Value A	Value B
After processing	Value B	Value B

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

```
STL                          Explanation
T "Tag_Value_1"              // Transfer contents of accumulator 1 to the operand
                             "Tag_Value_1".
POP                          // Shift the contents of accumulator 2 to lower-order
                             accumulator 1.
T "Tag_Value_2"              // Transfer contents of accumulator 1 to the operand
                             "Tag_Value_2".
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

Address register (S7-1500)

+AR1: Add accumulator 1 to AR1 (S7-1500)

Description

With the "Add accumulator 1 to AR1" instruction you add a value to the content of address register 1. The type of pointer located in address register 1 and the operand range remain as they are.

The following options are available for specifying the value to be added:

- Specification via a constant: The instruction adds the value of the constant to address register 1. The value of the constant must correspond to the format of a range-internal pointer (POINTER).
- Specification via the value of the right word in the accumulator 1: The instruction interprets the value of the right word in accumulator 1 as a 16-bit integer and extends it with the correct sign to 24 bits. The instruction then adds value of accumulator 1 to address register 1. Values from -32768 to +32767 are permitted.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Parameters

The following table shows the parameters of the "Add accumulator 1 to AR1" instruction:

Parameters	Data type	Description
<constant>	POINTER	Value that is added to address register 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
+AR1 P#10.0                  // Add pointer to address register 1.
L "Tag_Value"                // Load value of the operand "Tag_Value" into the
                             right word of accumulator 1.
+AR1                         // Add value of accumulator 1 to address register 1.
TAR1 %MD24                   // Transfer contents of address register 1 to double
                             word MD24.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

+AR2: Add accumulator 1 to AR2 (S7-1500)

Description

With the "Add accumulator 1 to AR2" instruction you add a value to the content of address register 2. The type of pointer located in address register 2 and the operand range remain as they are.

The following options are available for specifying the value to be added:

- Specification via a constant: The instruction adds the value of the constant to address register 2. The value of the constant must correspond to the format of a range-internal pointer (POINTER).
- Specification via the value of the right word in the accumulator 1: The instruction interprets the value of the right word in accumulator 1 as a 16-bit integer and extends it with the correct sign to 24 bits. The instruction then adds value of accumulator 1 to address register 2. Values from -32768 to +32767 are permitted.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Note

Address register AR2 is used when processing multi-instances. If you program the "Add accumulator 1 to AR2" instruction, you must first "save" the contents of address register 2 and reload it later.

Parameters

The following table shows the parameters of the "Add accumulator 1 to AR2" instruction:

Parameters	Data type	Description
<constant>	POINTER	Value that is added to address register 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
+AR2 P#10.0                  // Add pointer to address register 2.
L "Tag_Value"                // Load value of the operand "Tag_Value" into the
                             right word of accumulator 1.
+AR2                         // Add value of accumulator 1 to address register 2.
TAR2 %MD24                   // Transfer contents of address register 1 to double
                             word MD24.
```

See also

Overview of the valid data types (Page 231)
Basic information on the status word (Page 186)
Inserting STL instructions (Page 7259)
Editing STL instructions (Page 7277)
Instances (Page 62)
Querying and setting status bits in STL (Page 187)

Null instructions (S7-1500)

BLD: Program display (null instruction) (S7-1500)

Description

The instruction "Program display (null instruction)" does not execute any function and does not affect the status bits. The instruction is used to recognize code sequences during a parameter transfer or with LAD/FBD networks. It is created automatically when a LAD or FBD program is displayed in STL. The parameter value is the number of the instruction and is generated by the programming device.

Parameters

The following table shows the parameters of the "Program display (null instruction)" instruction:

Parameters	Data type	Description
<number>	WORD	Number of the instruction

See also

Overview of the valid data types (Page 231)
Basic information on the status word (Page 186)
Inserting STL instructions (Page 7259)
Editing STL instructions (Page 7277)
Instances (Page 62)
Querying and setting status bits in STL (Page 187)

NOP 0: Null instruction (S7-1500)

Description

If "Null instruction" has a parameter 0, it does not execute any function and does not affect the status bits. The instruction code contains a bit pattern with 16 zeros. The instruction is only relevant for the programming device when a program is displayed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A	// Query the signal state of the operand for "1" and AND with the current RLO.
JC NEXT	// If condition is satisfied (RLO = "1"), jump to jump label NEXT.
	// If the condition is not satisfied (RLO = "0"), execute the next instruction.
TAK	// Swap the contents of accumulators 1 and 2.
NEXT: NOP 0	// Jump label "NEXT"

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Querying and setting status bits in STL (Page 187)

NOP 1: Null instruction (S7-1500)

Description

If "Null instruction" has a parameter 1, it does not execute any function and does not affect the status bits. The instruction code contains a bit pattern with 16 ones. The instruction is only relevant for the programming device when a program is displayed.

See also

Overview of the valid data types (Page 231)
Basic information on the status word (Page 186)
Inserting STL instructions (Page 7259)
Editing STL instructions (Page 7277)
Instances (Page 62)
Querying and setting status bits in STL (Page 187)

4.1.3.4 SCL

Bit logic operations

R_TRIG: Detect positive signal edge

Description

With the "Detect positive signal edge" instruction, you can detect a state change from "0" to "1" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "0" to "1", a positive signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Incoming signal, the edge of which is to be queried
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

```
SCL
"R_TRIG_DB"(CLK := "TagIn",
            Q => "TagOut");
```

The previous state of the tag at the CLK input is stored in the "R_TRIG_DB" tag. If a change is the signal state from "0" to "1" is detected in the "TagIn_1" and "TagIn_2" operands or in the "TagIn_3" operand, the "TagOut_Q" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

F_TRIG: Detect negative signal edge

Description

With the "Detect negative signal edge" instruction, you can detect a state change from "1" to "0" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "1" to "0", a negative signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.

In all other cases, the signal state at the output of the instruction is "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Incoming signal, the edge of which is to be queried
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

SCL

```
"F_TRIG_DB"(CLK := "TagIn",
            Q => "TagOut");
```

The previous state of the tag at the CLK input is stored in the "F_TRIG_DB" tag. If a change in the signal state from "1" to "0" is detected in the "TagIn" operand, the "TagOut" output has signal state "1" for one cycle.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Timer operations

Calling IEC timers

Description

You have the option of declaring the IEC timer as single or multiple instance and calling it in the program code.

The following syntax options are available for the declaration of an IEC timer as multi-instance within a structure in the block interface:

IEC timer as ARRAY element

Declaration in the block interface:

Static	
MyARRAY	Array[0..1] of IEC_TIMER
MyARRAY[0]	IEC_TIMER
MyARRAY[1]	IEC_TIMER

Program code:

SCL

```
#MyARRAY[1].TOF(IN := <Operand>, PT := <Operand>)
```

Declaration in the block interface:

Static	
MyARRAY	Array[0..1] of TON_TIME
MyARRAY[0]	TON_TIME
MyARRAY[1]	TON_TIME

Program code:

SCL

```
#MyARRAY[#index](IN := <Operand>, PT := <Operand>)
```

IEC timer within an anonymous structure

Declaration in the block interface:

Static	
MyStruct	Struct
FirstTime	IEC_TIMER
PT	Time
ET	Time
IN	Bool
Q	Bool

Program code:

SCL
```
#MyStruct.FirstTime.TOF(IN := <Operand>, PT := <Operand>)
```

Declaration in the block interface:

Static	
MyStruct	Struct
FirstTime	TON_TIME
PT	Time
ET	Time
IN	Bool
Q	Bool

Program code:

SCL
```
#MyStruct.FirstTime(IN := <Operand>, PT := <Operand>)
```

IEC timer in the global data block

Declaration in the data block:

MyGlobalDB

	Name	Datentyp
1	Static	
2	Timer	"MyUDT"
3	FirstTime	IEC_TIMER
4	PT	Time
5	ET	Time
6	IN	Bool
7	Q	Bool

Program code:

SCL
```
"MyGlobalDB".Timer.FirstTime.TOF(IN := <Operand>, PT := <Operand>)
```

Declaration in the data block:

MyGlobalDB

	Name	Datentyp
1	Static	
2	Timer	"MyUDT"
3	FirstTime	IEC_TIMER
4	SecondTime	TON_TIME
5	PT	Time
6	ET	Time
7	IN	Bool
8	Q	Bool

Program code:

SCL
```
"MyGlobalDB".Timer.SecondTime(IN := <Operand>, PT := <Operand>)
```

IEC timer as elements in the block interface

Declaration in the block interface:

Static	
Timer	"MyUDT"
FirstTime	IEC_TIMER
PT	Time
ET	Time
IN	Bool
Q	Bool

Program code:

SCL
```
#Timer.FirstTime.TOF(IN := <Operand>, PT := <Operand>)
```

Declaration in the block interface:

Static	
Timer	"MyUDT"
FirstTime	IEC_TIMER
SecondTime	TON_TIME
PT	Time
ET	Time
IN	Bool
Q	Bool

Program code:

SCL
```
#Timer.SecondTime(IN := <Operand>, PT := <Operand>)
```

IEC timer within an ARRAY DB

Declaration in the ARRAY DB:

MyARRAYDB

	Name	Datentyp
1	MyARRAYDB	Array[0..1] of "MyUDT"
2	MyARRAYDB[0]	"MyUDT"
3	FirstTime	IEC_TIMER
4	PT	Time
5	ET	Time
6	IN	Bool
7	Q	Bool

Program code:

SCL

```
"MyARRAYDB"."THIS"[0].FirstTime.TOF(IN := <Operand>, PT := <Operand>)
```

Declaration in the ARRAY DB:

MyARRAYDB			
		Name	Datentyp
1		MyARRAYDB	Array[0..1] of "MyUDT"
2		MyARRAYDB[0]	"MyUDT"
3		FirstTime	IEC_TIMER
4		SecondTime	TON_TIME
5		PT	Time
6		ET	Time
7		IN	Bool
8		Q	Bool

Program code:

SCL

```
"MyARRAYDB"."THIS"[0].SecondTime(IN := <Operand>, PT := <Operand>)
```

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the parameter Q for a programmed duration. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. The Q parameter is set for the time PT, regardless of the subsequent changes in the input signal. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the time duration PT is active.

You can query the current time value set in the ET parameter. The timer value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT is reached and the signal state at the IN parameter is "0", the ET parameter is reset.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored.

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

For information on calling IEC timers within structures (multi-instance), please refer to: Calling IEC timers (Page 1661)

For CPUs of the S7-1200 series

An IEC timer is a structure of the data type IEC_TIMER or TP_TIME that you can declare as follows:

- Declaration of an instance data block of system data type IEC_TIMER (for example, "MyIEC_TIMER_DB")
- Declaration as local tag of the data type TP_TIME in the "Static" section of a program block (for example, #MyIEC_TIMER_Instance)

For CPUs of the S7-1500 series

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of an instance data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_LTIMER_DB")
- Declaration as local tag of the data type TP_TIME or TP_LTIME in the "Static" section of a program block (for example, #MyIEC_LTIMER_Instance)

IEC timer as instance data block of system data type IEC_<Timer> (Shared DB)

You can declare the IEC timer as a data block as follows:

<IEC_Timer_DB>.TP();

IEC timer as local tag from the block interface (multi-instance)

You can declare the IEC timer as a local tag as follows:

#myLocal_Timer();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L, P	Duration of the pulse. The value of the PT parameter must be positive.

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
Q	Output	BOOL	BOOL	I, Q, M, D, L, P	Operand that is set for the PT duration.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L, P	Current timer value

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:

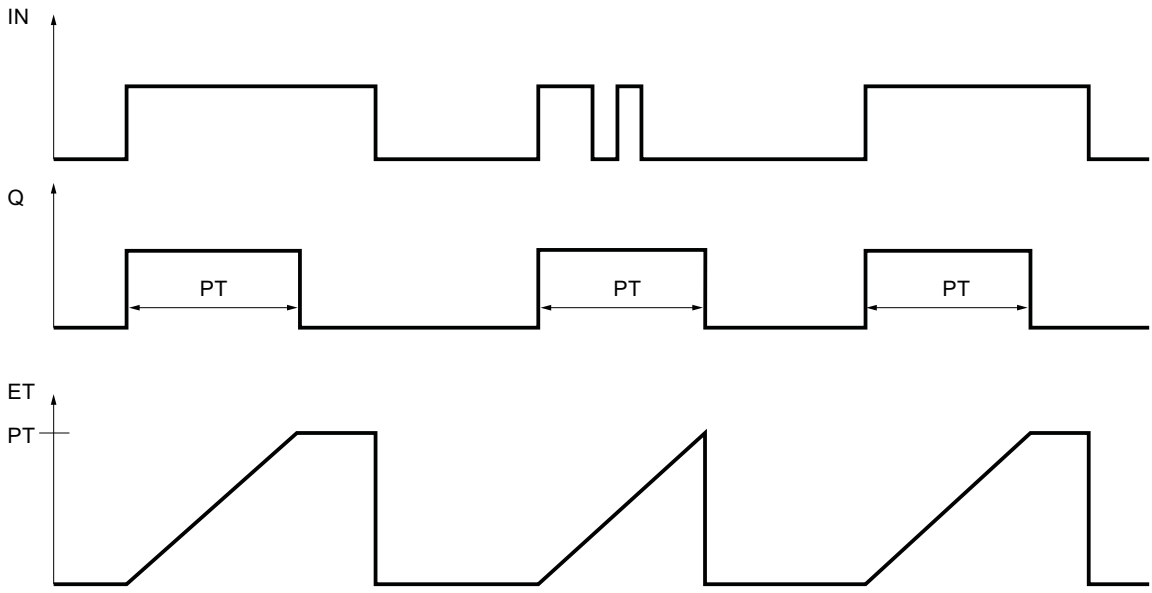

Example

The following example shows how the instruction works:

SCL

```
"TP_DB".TP(IN := "Tag_Start",
           PT := "Tag_PresetTime",
           Q => "Tag_Status",
           ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay the setting of the Q parameter for the programmed duration PT. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the PT duration has expired, the Q parameter returns signal state "1". The Q parameter remains set as long as the start input is still "1". If the signal state of the IN parameter changes from "1" to "0", the parameter Q will be reset. The timer function is restarted when a new positive signal edge is detected at the IN parameter.

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. The ET parameter is reset as soon as the signal state of the IN parameter changes to "0".

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

For information on calling IEC timers within structures (multi-instance), please refer to: Calling IEC timers (Page 1661)

For CPUs of the S7-1200 series

An IEC timer is a structure of the data type IEC_TIMER or TON_TIME that you can declare as follows:

- Declaration of an instance data block of system data type IEC_TIMER (for example, "MyIEC_TIMER_DB")
- Declaration as local tag of the data type TON_TIME in the "Static" section of a program block (for example, #MyIEC_TIMER_Instance)

For CPUs of the S7-1500 series

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of an instance data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER_DB")
- Declaration as local tag of the data type TON_TIME or TON_LTIME in the "Static" section of a program block (for example, #MyIEC_TIMER_Instance)

IEC timer as instance data block of system data type IEC_<Timer> (Shared DB)

You can declare the IEC timer as a data block as follows:

<IEC_Timer_DB>.TON();

IEC timer as local tag from the block interface (multi-instance)

You can declare the IEC timer as a local tag as follows:

#myLocal_Timer();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L, P	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L, P	Operand that is set when the timer PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L, P	Current timer value

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:

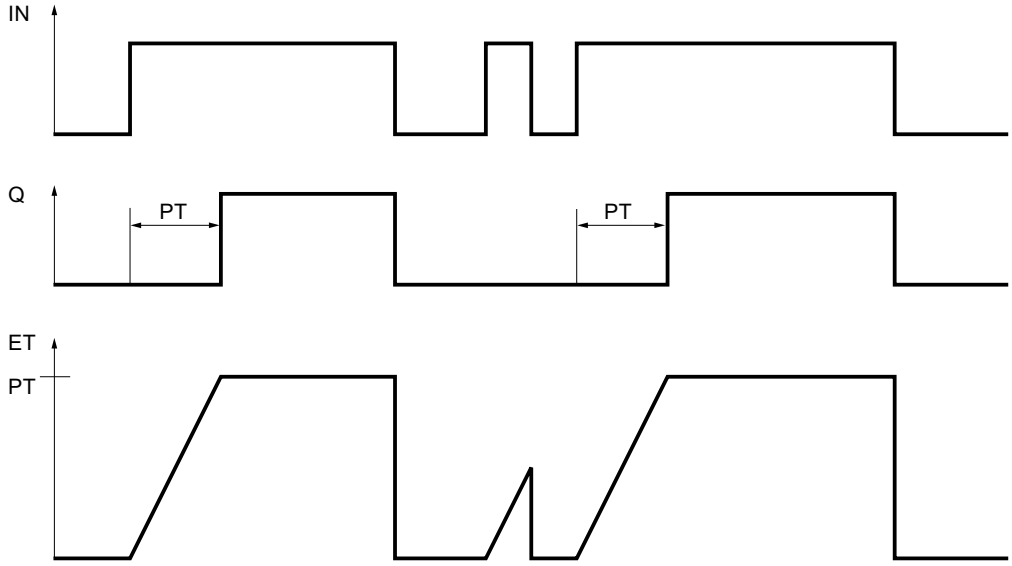

Example

The following example shows how the instruction works:

SCL

```
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. When the time duration expires, the "Tag_Status" operand is set to the signal state "1". The Tag_Status operand remains set to "1" as long as the Tag_Start operand has signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand. When the signal state at the operand "Tag_Start" changes from "1" to "0", the "Tag_Status" operand is reset.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay the resetting of the Q parameter for the programmed duration PT. The Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state at the IN parameter changes back to "0", the programmed time PT starts. The Q parameter remains set as long as time PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before time PT expires, the timer is reset. The signal state of the Q parameter remains set to "1".

The current time value can be queried in the ET parameter. The timer value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1". If the IN parameter changes to "1" before the time PT has expired, the ET parameter is reset to the value T#0s.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate off-delay" instruction must be assigned to an IEC Timer in which the instruction data is stored.

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

For information on calling IEC timers within structures (multi-instance), please refer to: Calling IEC timers (Page 1661)

For CPUs of the S7-1200 series

An IEC timer is a structure of the data type IEC_TIMER or TOF_TIME that you can declare as follows:

- Declaration of an instance data block of system data type IEC_TIMER (for example, "MyIEC_TIMER_DB")
- Declaration as local tag of the data type TOF_TIME in the "Static" section of a program block (for example, #MyIEC_TIMER_Instance)

For CPUs of the S7-1500 series

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of an instance data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER_DB")
- Declaration as local tag of the data type TOF_TIME or TOF_LTIME in the "Static" section of a program block (for example, #MyIEC_TIMER_Instance)

IEC timer as instance data block of system data type IEC_<Timer> (Shared DB)

You can declare the IEC timer as a data block as follows:

<IEC_Timer_DB>.TOF();

IEC timer as local tag from the block interface (multi-instance)

You can declare the IEC timer as a local tag as follows:

#myLocal_Timer();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L, P	Duration of the off delay. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L, P	Operand that is reset when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L, P	Current timer value

Pulse timing diagram

The following figure shows the pulse diagram of the "Generate off-delay" instruction:

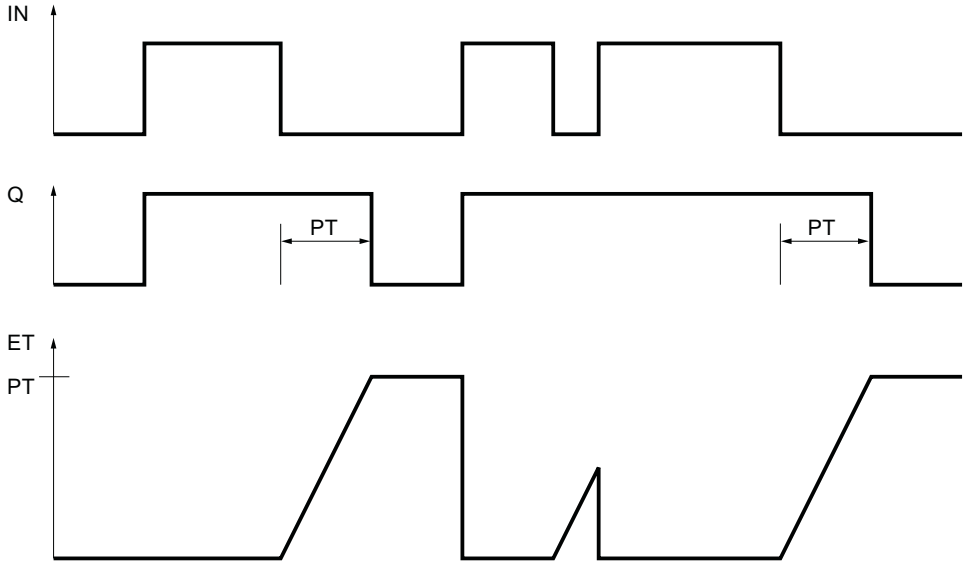

Example

The following example shows how the instruction works:

SCL

```
"TOF_DB".TOF(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

With a change in the signal state of the "Tag_Start" operand from "0" to "1", the "Tag_Status" operand is set. When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set. When the time has expired, the "Tag_Status" operand is reset. The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

TONR: Time accumulator

Description

The instruction "Time accumulator" is used to accumulate time values within a period set by the parameter PT. When the signal state of the IN parameter changes to "1", the instruction executes and the time duration PT starts. While the time duration PT is running, the time values that are recorded when the IN parameter has signal state "1" are accumulated. The accumulated time is output in the ET parameter and can be queried there. When the time duration PT is reached, the Q parameter has signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes to "0".

The R parameter resets the ET and Q parameters regardless of the signal state at the IN parameter.

Each call of the "Time accumulator" instruction must be assigned an IEC timer in which the instruction data is stored.

The instruction data is updated in the following cases:

- When the instruction is called, if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

For information on calling IEC timers within structures (multi-instance), please refer to: Calling IEC timers (Page 1661)

For CPUs of the S7-1200 series

An IEC Timer is a structure of the data type IEC_TIMER or TONR_TIME that you can declare as follows:

- Declaration of an instance data block of system data type IEC_TIMER (for example, "MyIEC_TIMER_DB")
- Declaration as local tag of the data type TONR_TIME in the "Static" section of a program block (for example, #MyIEC_TIMER_Instance)

For CPUs of the S7-1500 series

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of an instance data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER_DB")
- Declaration as local tag of the data type TONR_TIME or TONR_LTIME in the "Static" section of a program block (for example, #MyIEC_TIMER_Instance)

IEC timer as instance data block of system data type IEC_<Timer> (Shared DB)

You can declare the IEC timer as a data block as follows:

<IEC_Timer_DB>.TONR();

IEC timer as local tag from the block interface (multi-instance)

You can declare the IEC timer as a local tag as follows:

#myLocal_Timer();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I, Q, M, D, L, P	Start input
R	Input	BOOL	BOOL	I, Q, M, D, L, P	Reset of the ET and Q parameters
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L, P	Maximum duration of time recording. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L, P	Operand that remains set when the timer PT has expired.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L, P	Accumulated time

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:

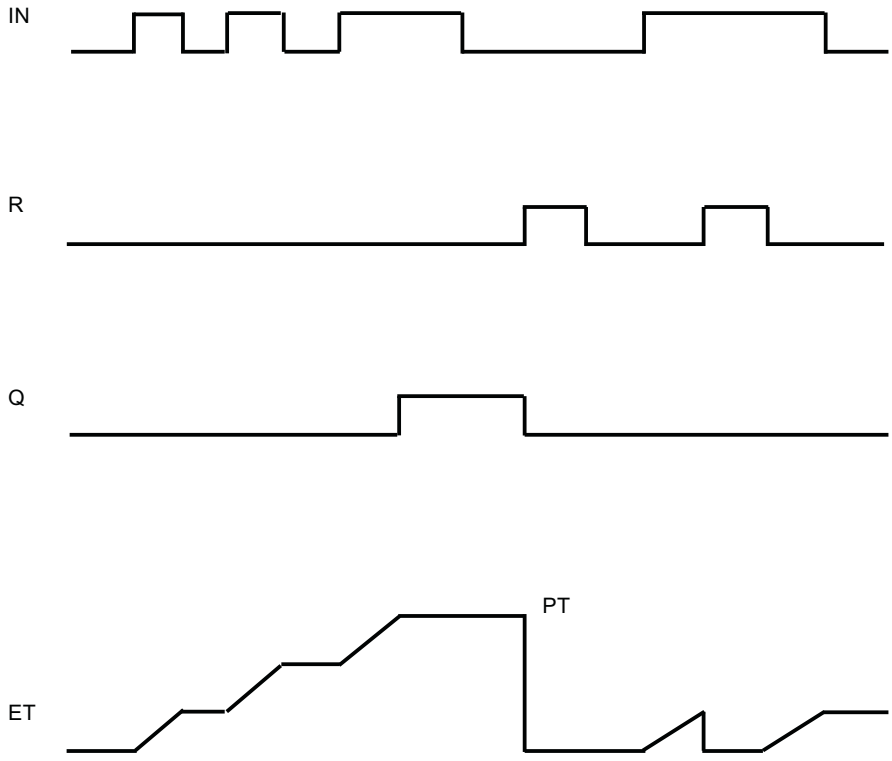

Example

The following example shows how the instruction works:

SCL

```
"TONR_DB".TONR(IN := "Tag_Start",
               R := "Tag_Reset",
               PT := "Tag_PresetTime",
               Q => "Tag_Status",
               ET => "Tag_Time");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. While the time is running, the time values are accumulated that are recorded when the "Tag_Start" operand has signal state "1". The accumulated time is stored in the "Tag_Time" operand. When the time value specified for the PT parameter is reached, the "Tag_Status" operand is set to the signal state "1". The current time value is stored in the "Tag_Time" operand.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

RESET_TIMER: Reset timer

Description

You can use the "Reset timer" instruction to reset an IEC timer to "0". The structure components of the timer in the specified data block are reset to "0".

The instruction does not influence the RLO. At the TIMER parameter, the "Reset timer" instruction is assigned an IEC timer declared in the program. The instruction must be programmed in an IF instruction. The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer that is reset

Example

The following example shows how the instruction works:

SCL

```
IF #started = false THEN
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
#started := true;
END_IF;

IF "TON_DB".ET < T#25s THEN
```

SCL

```
RESET_TIMER(TIMER := "TON_DB");
#started := false;
END_IF;
```

When the tag #started has the signal state "0", the instruction "Generate on-delay" is executed when there is positive signal edge on the operand "Tag_Start". The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PresetTime". The operand "Tag_Status" is set if the time duration specified by the operand "Tag_PresetTime" has expired. The parameter Q remains set as long as the operand "Tag_Start" still has the signal state "1". When the signal state of the start input changes from "1" to "0", the operand on the Q parameter is reset.

If the expired time of the IEC timer "TON_DB" is less than 25s, the "Reset timer" instruction is executed and the timer stored in the "TON_DB" instance data block is reset.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

PRESET_TIMER: Load time duration

Description

You can use the "Load time duration" instruction to set the time for an IEC timer. The instruction is executed in every cycle when the result of logic operation (RLO) at the input of the instruction has the signal state "1". The instruction writes the specified time to the structure of the specified IEC timer.

Note

If the specified IEC timer is running while the instruction executes, the instruction overwrites the current time of the specified IEC timer. This can change the timer status of the IEC timer.

You assign an IEC timer declared in the program to the "Load time duration" instruction.

The instruction data is updated only when the instruction is called and each time the assigned IEC timer is accessed. The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L	Duration with which the IEC timer runs.
<IEC timer>	Output	IEC_TIM-ER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIM-ER, IEC_LTI-MER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer whose duration of which is set.

Example

The following example shows how the instruction works:

SCL

```
IF #started = false THEN
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
#started := true;
#preset = true
END_IF;

IF "TON_DB".ET < T#10s AND #preset = true THEN
PRESET_TIMER(PT := T#25s,
             TIMER := "TON_DB");
#preset := false;
END_IF;
```

When the tag #started has the signal state "0" and the operand "Tag_Start" has a positive signal edge, the instruction "Generate on-delay" is executed. The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PresetTime". The operand "Tag_Status" is set if the time duration PT specified by the operand "Tag_PresetTime" has expired. The parameter Q remains set as long as the operand

"Tag_Start" still has the signal state "1". When the signal state of the start input changes from "1" to "0", the operand on the Q parameter is reset.

The instruction "Load time duration" is executed when the expired time of the IEC timer "TON_DB" is less than 10s and the tag #preset has the signal state "1". The instruction writes the time duration that is specified at the parameter PT in the instance data block "TON_DB", thereby overwriting the time value of the operand "Tag_PresetTime" within the instance data block. The signal state of the timer status may therefore change at the next query or when "TON_DB".Q or "TON_DB".ET are accessed.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Legacy (S7-1500)

S_PULSE: Assign pulse timer parameters and start (S7-1500)

Description

The "Assign pulse timer parameters and start" instruction starts the time programmed in the T_NO parameter when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) of the S parameter. The timer runs for the programmed time (TV) as long as the signal state of the S parameter is "1".

When the signal state of the S parameter changes to "0" before the programmed time has expired, the timer is stopped and the Q parameter is reset to "0".

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

Parameter Q returns signal state "1" as long as the timer is running and the signal state at parameter S is "1". When the signal state of the S parameter changes to "0" before the programmed time has expired, the Q parameter returns signal state "0". If the timer is reset by parameter R or if the timer has expired then parameter Q also returns signal state "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign pulse timer parameters and start" instruction:

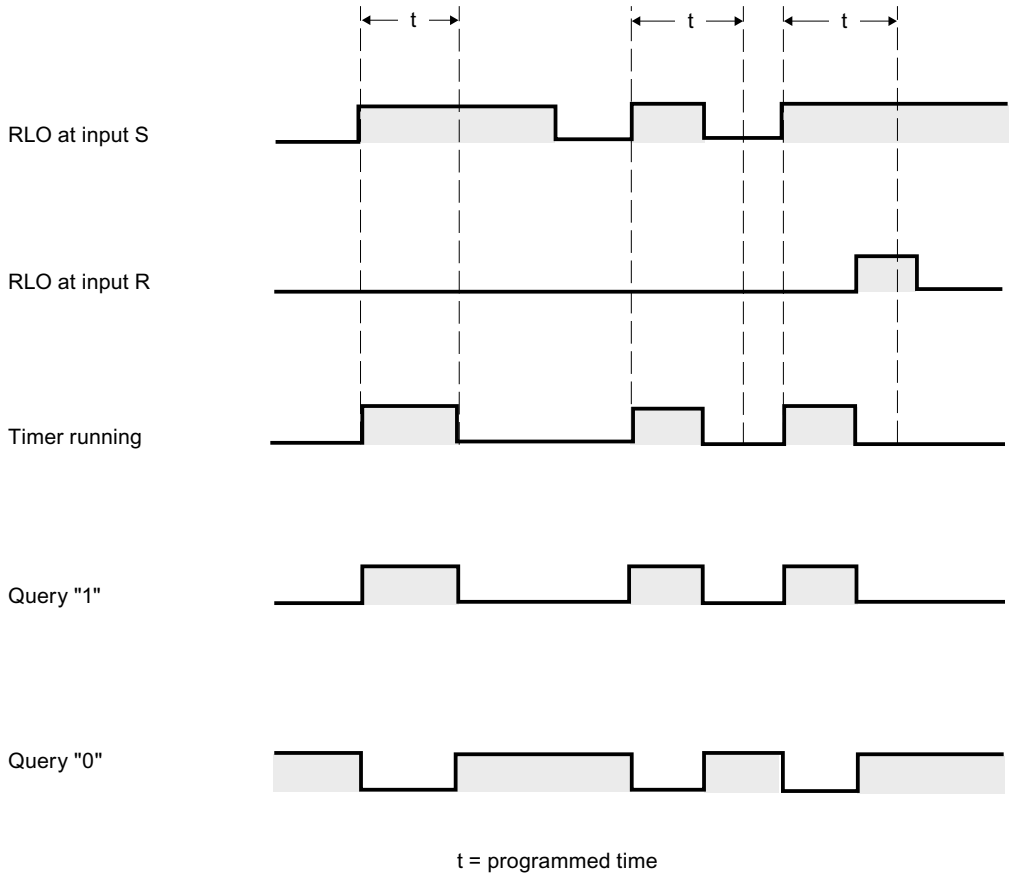

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_PULSE(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q => "Tag_Status",
                        BI => "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer counts down with the time value of the operand "Tag_Number" until operand "Tag_1" returns signal state "1".

If the signal state at the S parameter changes to "0" before the programmed time has elapsed, the "Tag_Status" operand is reset to "0". If the timer is reset by the R parameter or if the timer has expired, the "Tag_Status" operand also returns signal state "0".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basics of SCL (Page 7293)

S_PEXT: Assign extended pulse timer parameters and start (S7-1500)

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) even if the signal state at the S parameter changes to "0". As long as the timer runs, parameter Q returns the signal state "1".

When the timer has expired, parameter Q is reset to "0". If the signal state at the S parameter changes from "0" to "1" while the timer is running, the timer is restarted with the time programmed in the TV parameter.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the timer is running and the signal state at parameter R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running then signal state "1" at parameter R has no effect.

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:

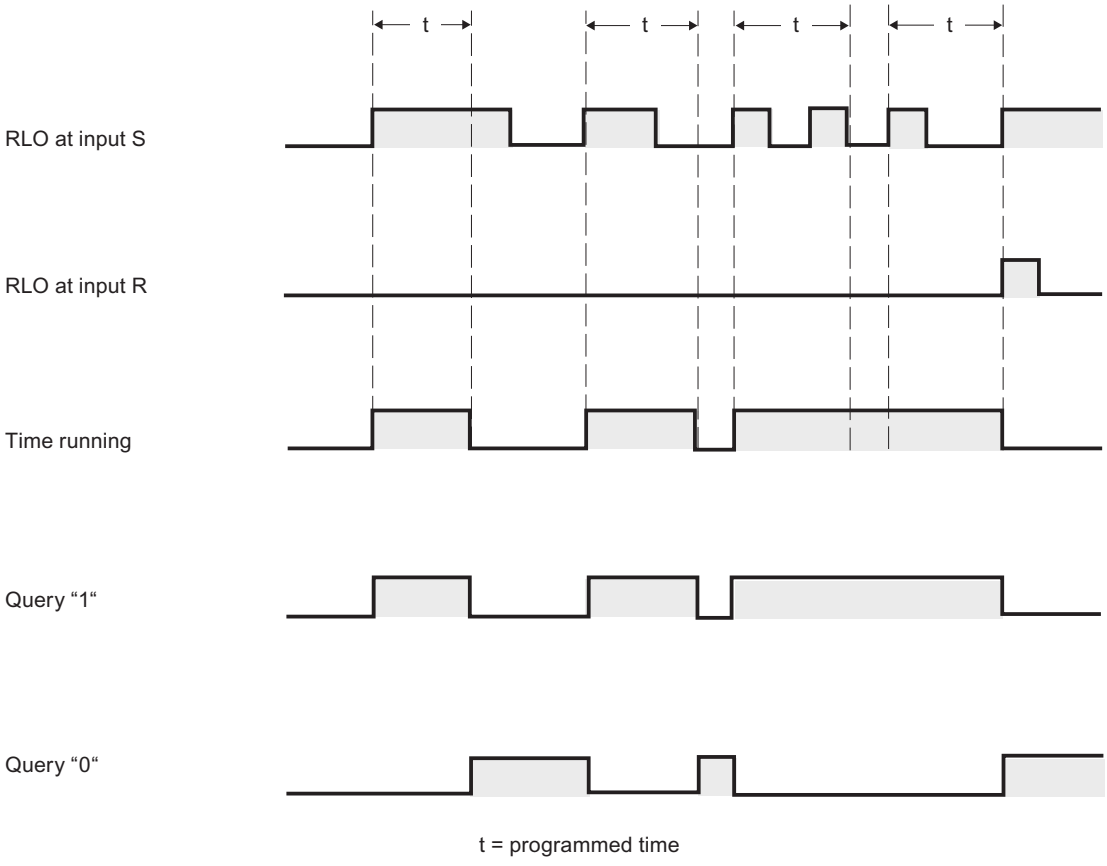

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := S_PEXT(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q => "Tag_Status",
                       BI => "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". As long as the timer runs, operand "Tag_Status" returns the signal state "1". When the timer has expired, operand "Tag_Status" is reset to "0". If the signal state at the S input changes from "0" to "1" while the timer is running, the timer is restarted with the time "Tag_Number".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basics of SCL (Page 7293)

S_ODT: Assign on-delay timer parameters and start (S7-1500)

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer as on-delay when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) as long as the signal state of the S parameter is "1".

If the timer expires correctly and parameter S still has signal state "1" then parameter Q returns signal state "1". If the signal state at the S parameter changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the time is running and the signal state at input R changes from "0" to "1" then the current time value and the time base are also set to zero. In this case, the signal state at parameter Q is "0". The timer is reset if the signal state at the R parameter is "1", even if the timer is not running and the result of logic operation (RLO) at the S parameter is "1".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:

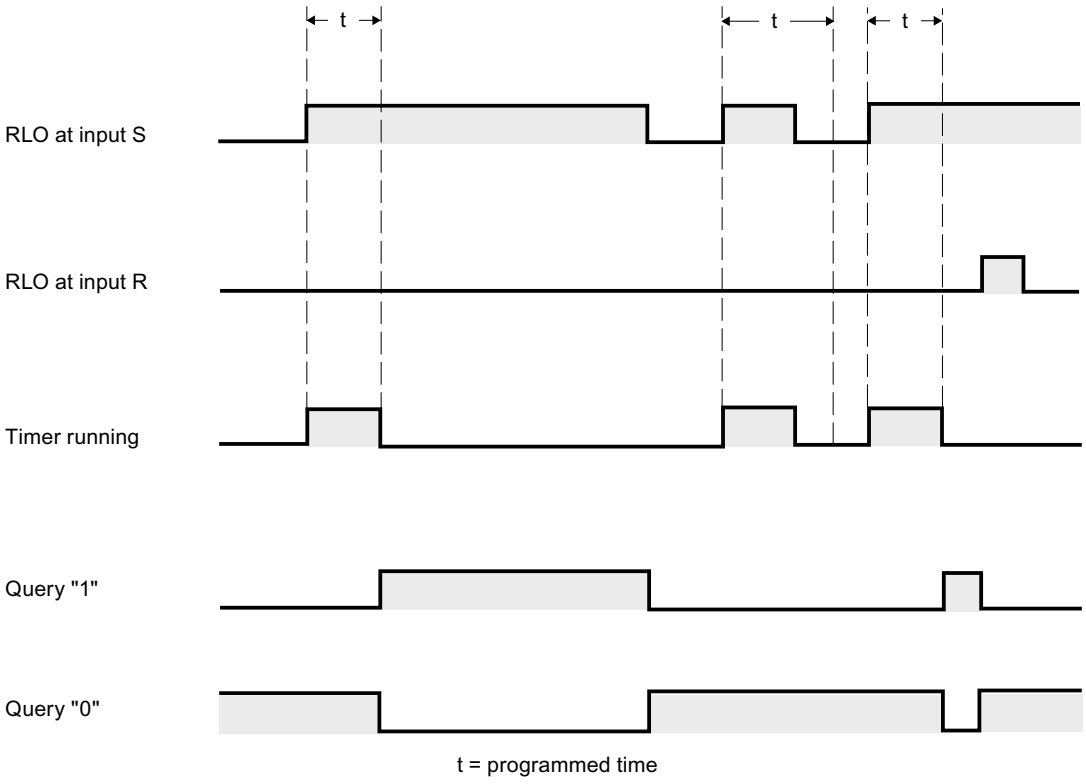

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_ODT(T_NO := "Timer_1",
                      S := "Tag_1",
                      TV := "Tag_Number",
                      R := "Tag_Reset",
                      Q => "Tag_Status",
                      BI => "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number" as long as the signal state of operand "Tag_1" is "1".

If the timer expires correctly and operand "Tag_Status" has signal state "1" then operand "Tag_Status" is reset to "1". If the signal state of the "Tag_1" operand changes from "1" to "0" while the timer is running, the timer is stopped. In this case, operand "Tag_Status" returns signal state "0".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basics of SCL (Page 7293)

S_ODTS: Assign retentive on-delay timer parameters and start (S7-1500)

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) even if the signal state at the S parameter changes to "0".

When the timer expires, the "Q" parameter returns signal state "1" regardless of the signal state of the "S" parameter. If the signal state at the S parameter changes from "0" to "1" while the timer is running, the timer is restarted with the programmed time TV.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

Signal state "1" at parameter R resets the current time value and time base to "0", independent of the signal state at parameter S. In this case, the signal state at parameter Q is "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value

Parameters	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:

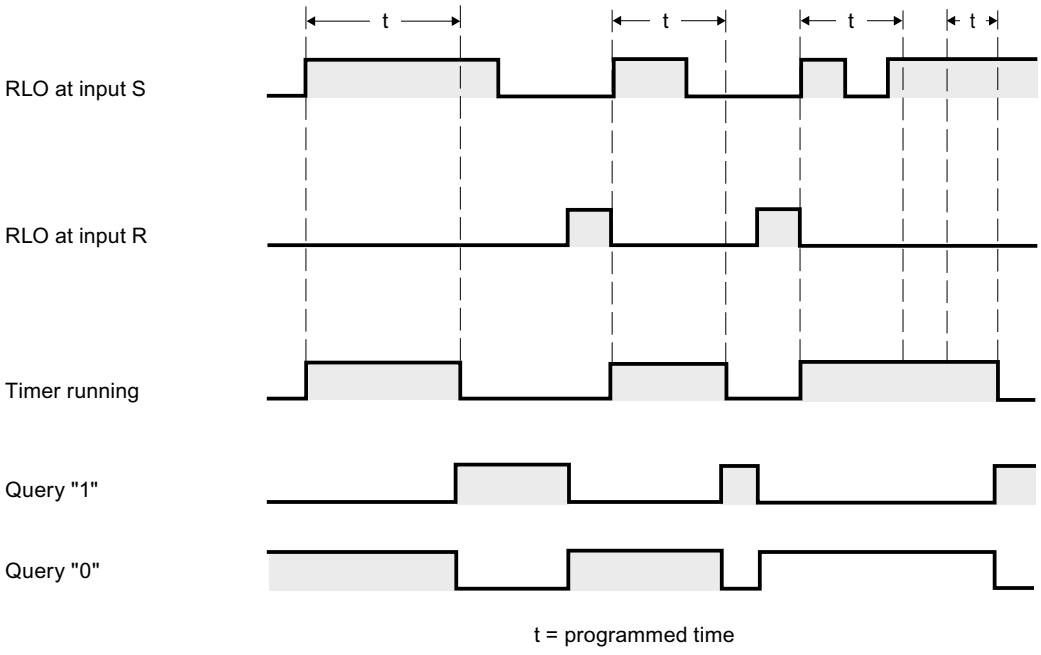

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_ODTS(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q => "Tag_Status",
                       BI => "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number".

If the timer expires, operand "Tag_Status" returns signal state "1" independent of the signal state of operand "Tag_1". If the signal state of the "Tag_1" operand changes from "0" to "1" while the timer is running, the timer is restarted with the time "Tag_Number".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basics of SCL (Page 7293)

S_OFFDT: Assign off-delay timer parameters and start (S7-1500)

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a negative signal edge is detected at the S parameter. The timer runs for the programmed time (TV). As long as the timer is running or parameter S returns signal state "1", then parameter Q has signal state "1".

If the timer expires and the signal state is "0" then parameter Q is reset to signal state "0". If the signal state at parameter S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at parameter S.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

Signal state "1" at parameter R resets the current time value and time base to "0". In this case, the signal state at parameter Q is "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:

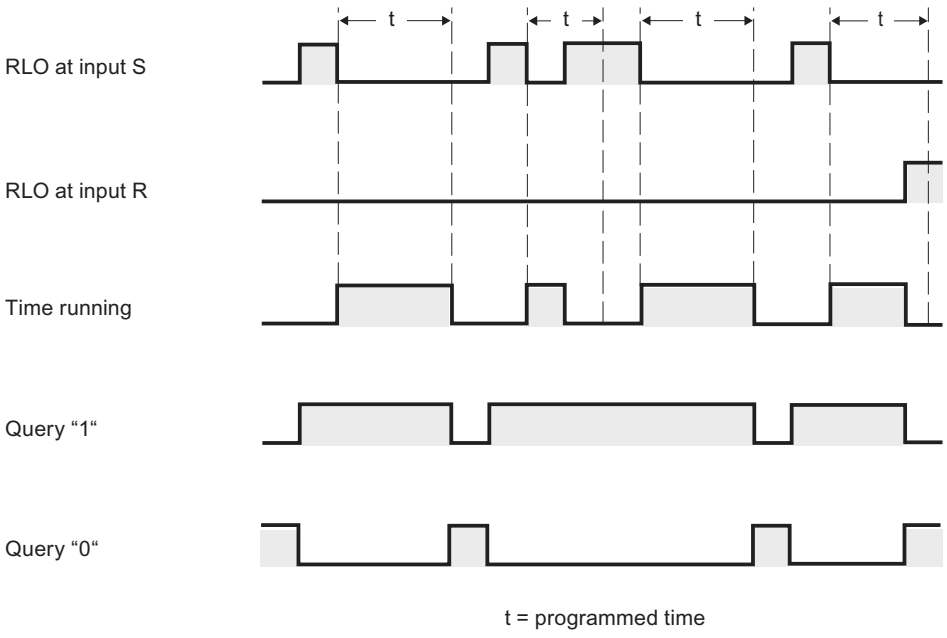

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_OFFDT(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q => "Tag_Status",
                        BI => "Tag_Value");
```

The timer "Timer_1" is started when the signal state of the operand "Tag_1" changes from "1" to "0". The timer runs for the duration "Tag_Number". As long as the timer is running or operand "Tag_1" returns signal state "1", then operand "Tag_Status" has signal state "1".

If the timer expires and the signal state of operand "Tag_1" is "0" then operand "Tag_Status" is reset to signal state "0". If the signal state of the "Tag_1" operand changes from "0" to "1" while the timer is running, the timer is reset. The timer is only restarted after a falling edge is detected at parameter S.

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 1506)

Basics of SCL (Page 7293)

Counter operations

Calling IEC counters

Description

You have the option of declaring the IEC counter as single or multiple instance and calling it in the program code.

The following syntax options are available for the declaration of an IEC counter as multi-instance within a structure in the block interface:

IEC counter as ARRAY element

Declaration in the block interface:

	Static	
	MyARRAY	Array[0..1] of IEC_COUNTER
	MyARRAY[0]	IEC_COUNTER
	MyARRAY[1]	IEC_COUNTER

Program code:

SCL
```
#MyARRAY[1].CTU(CU := <Operand>, PV := <Operand>)
```

Declaration in the block interface:

Static	
MyARRAY	Array[0..1] of CTU_INT
MyARRAY[0]	CTU_INT
MyARRAY[1]	CTU_INT

Program code:

SCL
```
#MyARRAY[#index](CU := <Operand>, PV := <Operand>)
```

IEC counter within an anonymous structure

Declaration in the block interface:

Static	
MyStruct	Struct
FirstTime	IEC_COUNTER
CU	Bool
CD	Bool
R	Bool
LD	Bool
QU	Bool
QD	Bool
PV	Int
CV	Int

Program code:

SCL
```
#MyStruct.FirstTime.CTU(CU := <Operand>, PV := <Operand>)
```

Declaration in the block interface:

Static	
MyStruct	Struct
FirstTime	CTU_INT
CU	Bool
CD	Bool
R	Bool
LD	Bool
QU	Bool
QD	Bool
PV	Int
CV	Int

Program code:

SCL

```
#MyStruct.FirstTime(CU := <Operand>, PV := <Operand>)
```

IEC counter in the global data block

Declaration in the data block:

MyGlobalDB		
	Name	Datentyp
1	Static	
2	Counter	"MyUDT"
3	FirstCount	IEC_COUNTER
4	CU	Bool
5	CD	Bool
6	R	Bool
7	LD	Bool
8	QU	Bool
9	QD	Bool
10	PV	Int
11	CV	Int

Program code:

SCL

```
"MyGlobalDB".Counter.FirstCount.CTU(CU := <Operand>, PV := <Operand>)
```

Declaration in the data block:

MyGlobalDB		
	Name	Datentyp
1	Static	
2	Counter	"MyUDT"
3	FirstCount	IEC_COUNTER
4	SecondCount	CTU_INT
5	CU	Bool
6	CD	Bool
7	R	Bool
8	LD	Bool
9	QU	Bool
10	QD	Bool
11	PV	Int
12	CV	Int

Program code:

SCL

```
"MyGlobalDB".Counter.SecondCount(CU := <Operand>, PV := <Operand>)
```

IEC counter as element in the block interface

Declaration in the block interface:

Static	
Counter	"MyUDT"
FirstCount	IEC_COUNTER
CU	Bool
CD	Bool
R	Bool
LD	Bool
QU	Bool
QD	Bool
PV	Int
CV	Int

Program code:

SCL

```
#Counter.FirstCount.CTU(CU := <Operand>, PV := <Operand>)
```

Declaration in the block interface:

Static	
Counter	"MyUDT"
FirstCount	IEC_COUNTER
SecondCount	CTU_INT
CU	Bool
CD	Bool
R	Bool
LD	Bool
QU	Bool
QD	Bool
PV	Int
CV	Int

Program code:

SCL

```
#Counter.SecondCount(CU := <Operand>, PV := <Operand>)
```

IEC counter within an ARRAY DB

Declaration in the ARRAY DB:

MyARRAYDB		
	Name	Datentyp
1	MyARRAYDB	Array[0..1] of "MyUDT"
2	MyARRAYDB[0]	"MyUDT"
3	FirstCount	IEC_COUNTER
4	CU	Bool
5	CD	Bool
6	R	Bool
7	LD	Bool
8	QU	Bool
9	QD	Bool
10	PV	Int
11	CV	Int

Program code:

SCL

```
"MyARRAYDB"."THIS"[0].FirstCount.CTU(CU := <Operand>, PV := <Operand>)
```

Declaration in the ARRAY DB:

MyARRAYDB		
	Name	Datentyp
1	MyARRAYDB	Array[0..1] of "MyUDT"
2	MyARRAYDB[0]	"MyUDT"
3	FirstCount	IEC_COUNTER
4	SecondCount	CTU_INT
5	CU	Bool
6	CD	Bool
7	R	Bool
8	LD	Bool
9	QU	Bool
10	QD	Bool
11	PV	Int
12	CV	Int

Program code:

SCL

```
"MyARRAYDB"."THIS"[0].SecondCount(CU := <Operand>, PV := <Operand>)
```

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at the CV parameter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the parameter CV is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit of the data type specified at the CV parameter. When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current counter value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the signal state of the R parameter is "1", the signal state of the CU parameter has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For CPUs of the S7-1200 series

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT

For CPUs of the S7-1500 series

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT

You can declare an IEC counter as follows:

- Declaration of an instance data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER_DB")
- Declaration as local tag of the data type CTUD_<DatenType> in the "Static" section of a program block (for example, #MyIEC_COUNTER_Instance)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive.

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

IEC counter as data block of system data type IEC_<Counter> (Shared DB)

You can declare the IEC counter as a data block as follows:

<IEC_Counter_DB>.CTU();

IEC counter as local tag from the block interface (multi-instance)

You can declare the IEC counter as a local tag as follows:

#myLocal_Counter();

For information on calling IEC counters within structures (multi-instance), please refer to: Calling IEC counters (Page 1691)

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, P	Reset input
PV	Input	Integers	I, Q, M, D, L, P	Value at which the Q output is set
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

Example

The following example shows how the instruction works:

SCL

```
"IEC_COUNTER_DB".CTU(CU := "Tag_Start",
                     R := "Tag_Reset",
                     PV := "Tag_PresetValue",
                     Q => "Tag_Status",
                     CV => "Tag_CounterValue");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the specified data type (INT = 32767) is reached.

The "Tag_Status" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" operand. In all other cases, the "Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 231)

Setting retentivity in an instance data block (Page 7037)

Basics of SCL (Page 7293)

CTD: Count down

Description

The "Count down" instruction is used to decrement the value at the parameter CV. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the signal state of the LD parameter is "1", the signal state of the CD parameter has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For CPUs of the S7-1200 series

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT

For CPUs of the S7-1500 series

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT

You can declare an IEC counter as follows:

- Declaration of an instance data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER_DB")
- Declaration as local tag of the data type CTD_<DatenType> in the "Static" section of a program block (for example, #MyIEC_COUNTER_Instance)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive.

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

IEC counter as data block of system data type IEC_<Counter> (Shared DB)

You can declare the IEC counter as a data block as follows:

<IEC_Counter_DB>.CTD();

IEC counter as local tag from the block interface (multi-instance)

You can declare the IEC counter as a local tag as follows:

#myLocal_Counter();

For information on calling IEC counters within structures (multi-instance), please refer to: Calling IEC counters (Page 1691)

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	Integers	I, Q, M, D, L, P	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

Example

The following example shows how the instruction works:

```
SCL
"IEC_SCOUNTER_DB".CTD(CD := "Tag_Start",
                      LD := "Tag_Load",
                      PV := "Tag_PresetValue",
                      Q => "Tag_Status",
                      CV => "Tag_CounterValue");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the instruction is executed and the value of the "Tag_CounterValue" operand is decremented by one. With each additional positive signal edge, the counter value is decremented until it reaches the low limit of the specified data type (-128).

The operand "Tag_Status" has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 231)

Setting retentivity in an instance data block (Page 7037)

Basics of SCL (Page 7293)

CTUD: Count up and down

Description

Use the "Count up and down" instruction to increment or decrement the counter value at the CV parameter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the counter value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in a program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV parameter. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state of the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has signal state "1", the signal state of the CU and CD parameters has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to

signal state "1". In all other cases, the signal state of the QU parameter is "0". You can also specify a constant for the PV parameter.

You can query the status of the down counter at the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For CPUs of the S7-1200 series

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT

For CPUs of the S7-1500 series

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT

You can declare an IEC counter as follows:

- Declaration of an instance data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER_DB")
- Declaration as local tag of the data type CTUD_<DatenType> in the "Static" section of a program block (for example, #MyCTUD_COUNTER_Instance)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive.

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

IEC counter as data block of system data type IEC_<Counter> (Shared DB)

You can declare the IEC counter as a data block as follows:

<IEC_Counter_DB>.CTUD();

IEC counter as local tag from the block interface (multi-instance)

You can declare the IEC counter as a local tag as follows:

#myLocal_Counter();

For information on calling IEC counters within structures (multi-instance), please refer to: Calling IEC counters (Page 1691)

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	Integers	I, Q, M, D, L, P	Value at which the QU output is set / value to which the CV output is set when LD = 1.
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

Example

The following example shows how the instruction works:

SCL

```
"IEC_COUNTER_DB".CTUD(CU := "Tag_Start1",
                      CD := "Tag_Start2",
                      LD := "Tag_Load",
                      R := "Tag_Reset",
                      PV := "Tag_PresetValue",
                      QU => "Tag_CU_Status",
                      QD => "Tag_CD_Status",
                      CV => "Tag_CounterValue");
```

If the "Tag_Start1" operand has a positive signal edge in the signal state, the current counter value is incremented by one and stored in the "Tag_CounterValue" operand. If the "Tag_Start2" operand has a positive signal edge in the signal state, the counter value is decremented by one and is also stored in the "Tag_CounterValue" operand. The counter value is incremented on the positive signal edge of the CU parameter until it reaches the high limit of the specified data type (INT). If the CD parameter has a positive signal edge, the counter value is decremented until it reaches the low limit of the specified data type (INT).

The operand "Tag_CU_Status" has signal state "1" as long as the current counter value is greater than or equal to the value of the operand "Tag_PresetValue". In all other cases, the "Tag_CU_Status" output has signal state "0".

The operand "Tag_CD_Status" has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_CD_Status" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Setting retentivity in an instance data block (Page 7037)

Basics of SCL (Page 7293)

Legacy (S7-1500)

S_CU: Assign parameters and count up (S7-1500)

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is provided at the parameter CV. The counter value is incremented until the limit of "999" is reached. When the limit value is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state of the S parameter changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if the result of logic operation (RLO) at the CU input is "1", the counter counts once in the next cycle, even when no signal edge change is detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has the signal state "1", a change in the signal state of the parameters CU and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
C_NO	Input	COUNTER, INT	C	Counter operations The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
S	Input	BOOL	I, Q, M, D, L	Input for presetting the counter

Parameters	Declaration	Data type	Memory area	Description
PV	Input	WORD	I, Q, M, D, L	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	I, Q, M, D, L	Reset input
Q	Output	BOOL	I, Q, M, D, L	Status of the counter
CV	Output	WORD	I, Q, M, D, L	Current counter value
Function value		WORD	I, Q, M, D, L	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CU(C_NO := "Counter_1",
                     CU := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

If the signal state of the "Tag_Start" parameter changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. If the signal state of the "Tag_1" input changes from "0" to "1", the counter value in BCD format is set to the value of the "Tag_PresetValue" operand. The counter value is reset to "0" when the "Tag_Reset" operand has signal state "1".

The current counter value is stored in hexadecimal form in the operand "Tag_Value".

The "Tag_Status" output has the signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

S_CD: Assign parameters and count down (S7-1500)

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the current counter value is decreased by one. The current counter value is provided at the parameter CV. The counter value is decreased until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decreased on a positive signal edge.

When the signal state of the S parameter changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if the result of logic operation (RLO) at the CD parameter is "1", the counter counts once in the next cycle, even when no signal edge change is detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has the signal state "1", a change in the signal state of the parameters CD and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
C_NO	Input	COUNTER, INT	C	Counter operations The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L	Count down input
S	Input	BOOL	I, Q, M, D, L	Input for presetting the counter
PV	Input	WORD	I, Q, M, D, L	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	I, Q, M, D, L	Reset input
Q	Output	BOOL	I, Q, M, D, L	Status of the counter
CV	Output	WORD	I, Q, M, D, L	Current counter value
Function value		WORD	I, Q, M, D, L	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CD(C_NO := "Counter_1",
                     CD := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decreased by one. If the signal state of the "Tag_1" operand changes from "0" to "1", the counter value in BCD format is set to the value of the "Tag_PresetValue" operand. The counter value is reset to "0" when the "Tag_Reset" operand has signal state "1".

The current counter value is stored in the "Tag_Value" operand.

The "Tag_Status" operand returns signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

S_CUD: Assign parameters and count up / down (S7-1500)

Description

You can use the "Assign parameters and count up / down" instruction to increment and decrement the value of a counter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the counter value is decreased by one. The current counter value is provided at the parameter CV. If there is a positive signal edge at the parameters CU and CD in one program cycle, the counter value remains unchanged.

The counter value is incremented until the high limit of "999" is reached. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit "0" is reached, the counter value is not decremented any further.

When the signal state of the S parameter changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if the result of logic operation (RLO) of the CU and CD parameters is "1", the counter will count once in the next cycle, even if no signal edge change was detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the parameter R has the signal state "1", processing of the signal state of the parameters CU, CD and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
C_NO	Input	COUNTER, INT	C	Counter operations The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
S	Input	BOOL	I, Q, M, D, L	Input for presetting the counter
PV	Input	WORD	I, Q, M, D, L	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	I, Q, M, D, L	Reset input
Q	Output	BOOL	I, Q, M, D, L	Status of the counter
CV	Output	WORD	I, Q, M, D, L	Current counter value (hexadecimal)
Function value		WORD	I, Q, M, D, L	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CD(C_NO := "Counter_1",
                     CU := "Tag_CU",
                     CD := "Tag_CD",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

When a positive signal edge is detected in the signal state of the "Tag_CU" operand and the current counter value is less than "999", the counter value is incremented by one. When a positive signal edge is detected in the signal state of the "Tag_CD" operand and the current counter value is greater than "0", the counter value is decremented by one.

If the signal state of the "Tag_1" operand changes from "0" to "1", the counter value in BCD format is set to the value of the "Tag_PresetValue" operand. The counter value is reset to "0" when the "Tag_Reset" operand has signal state "1".

The current counter value is stored in the "Tag_Value" operand.

The "Tag_Status" operand returns signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Comparator operations

TypeOf: Check data type of a VARIANT tag

Description

You can use the "Check data type of a VARIANT tag" instruction to query the data type of a tag to which a VARIANT tag is pointing. You can compare the data type of the that you have declared in the block interface either to the data type of another tag or directly with a data type to determine whether they are "Equal" or "Not equal".

The comparison operand can be an elementary data type or a PLC data type.

You can only use the "Check data type of a VARIANT tag" instruction within an IF or CASE instruction.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Binary numbers, integers, floating-point numbers, times, date and time, character strings, VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand to query

For additional information on valid data types, refer to "See also".

Example

The following example shows the comparison with another tag:

SCL
```
IF TypeOf(#TagVARIANT) = TypeOf("TagBYTE") THEN
...;
END_IF;
```

The following example shows the comparison with a data type:

SCL
```
IF TypeOf(#TagVARIANT) = BYTE THEN
...;
END_IF;
```

If the operand to which the VARIANT #TagVARIANT is pointing has the BYTE data type, the comparison condition has been met.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Example of programming a queue (FIFO) (Page 336)

Basic information on VARIANT (Page 324)

Basics of SCL (Page 7293)

TypeOfElements: Check data type of an ARRAY element of a VARIANT tag

Description

You can use the "Check data type of an ARRAY element of a VARIANT tag" instruction to query the data type of a tag to which a VARIANT tag is pointing. You compare the data type of a tag to the data type of the tag that you have declared in the block interface to determine whether they are "Equal" or "Not equal".

The operand must have the VARIANT data type. The comparison operand can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand to query

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

SCL

```
IF TypeOfElements("Tag_Variant") = TypeOF("GlobalDB".Product[1]) THEN
"Tag_Variant" := "GlobalDB".Product[1] * 3;
END_IF;
```

The following table shows how the instruction works using specific operand values:

Operand	Value
"GlobalDB".Product[1]	1.5
Tag_Variant	4.5

If the tag to which the VARIANT points and the "GlobalDB".Product[1] operand have the REAL data type, the "GlobalDB".Product[1] operand is then multiplied by 3 and the result is written to the "Tag_Variant" operand.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Example of programming a queue (FIFO) (Page 336)

Basic information on VARIANT (Page 324)

Basics of SCL (Page 7293)

IS_ARRAY: Check for ARRAY

Description

You can use the "Check for ARRAY" instruction to query whether the VARIANT points to a tag of the ARRAY data type.

You can only use the "Check for ARRAY" instruction within an IF instruction.

Parameters

The following table shows the parameters of the "Check for ARRAY" instruction:

Parameters	Declaration	Data type	Memory area	Description
<operand>	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand that is queried for ARRAY
Function value		UDINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF IS_ARRAY(#Tag_VARIANTToArray) THEN
"Tag_Result" := CountOfElements(#Tag_VARIANTToArray);
END_IF;
```

If the tag to which the VARIANT points is an ARRAY, the number of ARRAY elements is output.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Example of programming a queue (FIFO) (Page 336)

Basic information on VARIANT (Page 324)

Basics of SCL (Page 7293)

TypeOfDB: Query data type of a DB (S7-1500)

Description

The "Query data type of a DB" instruction is used to query which data type the data block has that the tag of the DB_ANY data type addresses. You can compare the data type of the DB addressed by the tag <Operand> either with the data type of another tag or directly with a data type for "Equal" or "Not equal".

The tag must have the DB_ANY data type. The comparison operand, for example, can be a PLC data type, a system data type, an axis or an FB.

You can only use the "Query data type of a DB" instruction within an IF or CASE instruction.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	DB_ANY	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Operand to query

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

SCL

```
IF TypeOfDB(#InputDBAny) = TO_SpeedAxis THEN
"TagOut" := 1;
END_IF;
```

The "TagOut" output is set if the data type of the #InputDBAny operand addressed DB is equal to the TO_SpeedAxis data type.

The "TagOut" output is not set when the following conditions are fulfilled:

- The number of the data block is "0".
- The data block does not exist.
- The data block is an ARRAY DB.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Using the DB_ANY data type (Page 204)

Basics of SCL (Page 7293)

Math functions

ABS: Form absolute value

Description

Use the "Form absolute value" instruction to calculate the absolute value of an input value and to save the result in the specified operands.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<expression>	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Input value
Function value		SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := ABS("Tag_Value");
"Tag_Result2" := ABS("Tag_Value1"*"Tag_Value2");
```

The absolute value of the input value is returned in the format of the input value as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	-2
Tag_Result1	2
Tag_Value1	4
Tag_Value2	-1
Tag_Result2	4

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values of the available inputs and returns the lowest value as the result.

A minimum of two and a maximum of 32 inputs can be specified at the instruction.

The result is invalid if any of the following conditions are met:

- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Second input value
INn	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Additionally inserted inputs whose values are to be compared
Function value		Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Result of the instruction
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := MIN(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	12222
IN2	Tag_Value2	14444
IN3	Tag_Value3	13333
Function value	Tag_Result	12222

The instruction compares the values of the available inputs and copies the lowest value ("Tag_Value1") to operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values of the available inputs and returns the greatest value as the result.

A minimum of two and a maximum of 32 inputs can be specified at the instruction.

The result is invalid if any of the following conditions are met:

- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Second input value
INn	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Additionally inserted inputs whose values are to be compared
Function value		Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Result of the instruction
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MAX(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	12 222
IN2	Tag_Value2	14 444
IN3	Tag_Value3	13 333
Function value	Tag_Result	14 444

The instruction compares the values of the specified operands and copies the greatest value ("Tag_Value2") to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

LIMIT: Set limit value

Description

The "Set limit value" instruction limits the value of the parameter IN to the values of the parameters MN and MX. The value of the parameter MN may not be greater than the value of the parameter MX.

If the value of the IN parameter fulfills the condition MN <= IN <= MX, it is returned as the result of the instruction. If the condition is not fulfilled and the IN input value is less than the MN low limit, the value of the MN parameter is returned as the result. If the high limit MX is exceeded, the value of the MX parameter is returned as the result.

If the value at the MN input is greater than at the MX input, the result is the value specified at the IN parameter and the enable output ENO is "0".

The instruction is only executed if the operands of all parameters are of the same data type.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
MN	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Low limit
IN	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Input value
MX	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	High limit
Function value		Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	I, Q, M, D, L, P	Result of the instruction
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := LIMIT(MN := "Tag_Minimum",
                      IN := "Tag_Value",
                      MX := "Tag_Maximum");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_Minimum	12 000
IN	Tag_Value	8 000
MX	Tag_Maximum	16 000
Function value	Tag_Result	12 000

The value of operand "Tag_Value" is compared with the values of operands "Tag_Minimum" and "Tag_Maximum". Because the value of the operand "Tag_Value" is less than the low limit value, the value of operand "Tag_Minimum" will be copied to operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SQR: Form square

Description

Use the "Form square" instruction to square the input value and save the result in the specified operand.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := SQR("Tag_Value");
"Tag_Result2" := SQR((SQR("Tag_Value1"))*"Tag_Value2");
```

The square of the input value is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	6.25
Tag_Value1	6.0
Tag_Value2	2.0
Tag_Result2	5184.0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SQRT: Form square root

Description

Use the "Form square root" instruction to calculate the square root of the input value and save the result in the specified operand. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number. If the input value is "0", the result is also "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := SQRT("Tag_Value");
"Tag_Result2" := SQRT((SQR("Tag_Value1"))+"Tag_Value2");
```

The square root of the input value is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	4.0
Tag_Result1	2.0
Tag_Value1	3.0
Tag_Value2	16.0
Tag_Result2	5.0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e=2.718282) from the input value. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := LN("Tag_Value");
"Tag_Result2" := LN("Tag_Value1"+"Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	0.916
Tag_Value1	1.5
Tag_Value2	3.2
Tag_Result2	1.548

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

EXP: Form exponential value

Description

The "Form exponential value" instruction calculates the exponent from the base e (e = 2.718282) and the input value and saves the result in the specified operand.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Exponential value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := EXP("Tag_Value");
"Tag_Result2" := EXP("Tag_Value1"/"Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	20.5
Tag_Result1	799 902 200

Operand	Value
Tag_Value1	15.5
Tag_Value2	30.2
Tag_Result2	1.671

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the input value. The input value must be given in radians.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value (size of an angle in radians)
Function value		Floating-point numbers	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SIN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796 (π/2)
Tag_Result	1.0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the input value. The input value must be given in radians.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value (size of an angle in radians)
Function value		Floating-point numbers	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := COS("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796 (π/2)
Tag_Result	0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the sine of the input value. The input value must be given in radians.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value (size of an angle in radians)
Function value		Floating-point numbers	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := TAN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+3.141593 (π)
Tag_Result	0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from a sine value, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in radians and can range in value from -π/2 to +π/2.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Sine value
Function value		Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ASIN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+1.570796 (π/2)

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from a cosine value, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in radians and can range in value from 0 to +π.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Cosine value
Function value		Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ACOS("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	0
Tag_Result	+1.570796 (π/2)

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from a tangent value, which corresponds to this value. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) as input value. The calculated angle size is given in radians and can range in value from -π/2 to +π/2.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Tangent value
Function value		Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ATAN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+0.785398 (π/4)

See also

Overview of the valid data types (Page 231)

Invalid floating-point numbers (Page 253)

Basics of SCL (Page 7293)

FRAC: Return fraction

Description

The result of the instruction "Return fraction" returns the decimal places of a value. Input value 123.4567, for example, returns the value 0.4567.

Use the following syntax to change the data type of the instruction:

FRAC_<data type>();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
_<Data type>		Floating-point numbers Default: REAL	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Floating-point numbers	I, Q, M, D, L, P	Decimal places of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := FRAC("Tag_Value");
"Tag_Result2" := FRAC_LREAL("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	2.555	-1.4421
Tag_Result1	0.555	-0.4421

See also

Overview of the valid data types (Page 231)

Invalid floating-point numbers (Page 253)

Basics of SCL (Page 7293)

Move operations

Deserialize: Deserialize

Description

You can use the "Deserialize" instruction to convert back the sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type> and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type> is located must have the ARRAY of BYTE or ARRAY of CHAR data type and be declared with standard access in version 1.0. Optimized memory areas also permitted as of version 2.0. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion. If the memory area is filled using the instruction "Serialize" and it is possible that fill bytes were inserted, their conversion is not taken into account.

It is recommended to define the low limit of the ARRAY with "0", because then the index within ARRAY corresponds to the value of the POS parameter, e.g. ARRAY[0] = POS 0. The description and example below have been formed based on this.

The instruction enables you to convert multiple sequential representations of converted data back to their original state step-by-step.

If you only want to convert back a single sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type>, you can also directly use the instruction "TRCV: Receive data via communication connection".

Size of the memory area

The alignment rules mean that simple structures in the optimized memory area do not contain filling bytes. This results in a structure in the optimized memory area being smaller than in the standard memory area. ARRAYs of structures and structures in which other structures are located contain filling bytes. There is no general rule as to the memory area in which a composed structure requires more space.

Applies to CPUs of the S7-1500 series:

For a block with the block property "Optimized block access", the BOOL has a length of 1 byte. A structure that consists largely of the data type BOOL can therefore be larger in the optimized memory area than in the standard memory area. Composed structures with a low proportion of BOOL data types are smaller in the optimized memory area than in the standard memory area.

Optimized memory area

To deserialize larger structures, you can declare the memory area for sequential representation with optimized access for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. The content of the sequential representation remains unchanged, as for a standard memory area. Only symbolic access to the bytes in the ARRAY is possible.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC_ARRAY	Input	VARIANT ARRAY[*] of BYTE [1]	D, block interface of an FB (the sections Input, Output, Static and Temp are possible.)	Pointer to an ARRAY of BYTE or ARRAY of CHAR in which the data stream is saved.
DEST_VARIABLE	InOut	VARIANT	D, block interface of an FB	Pointer to a tag of the STRUCT, ARRAY or PLC data type (UDT) to be deserialized.
POS	InOut	DINT	I, Q, M, D, L	The operand at the POS parameter stores the index of the first byte based on the number of bytes occupied by the converted customer data. The POS parameter is calculated zero-based.
Function value		INT	I, Q, M, D, L	Error information
[1] Possible for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The tag at the SRC_ARRAY parameter is not in a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains a ZERO pointer.
8151	No valid reference at the SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8154	Invalid data type at the SRC_ARRAY parameter
8250	A NULL pointer was transferred at parameter DEST_ARRAY.
8251	No valid reference at the DEST_VARIABLE parameter
8382	The value at the POS parameter is outside the limits of the array.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

For a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0, the following error code has a different meaning:

Error code* (W#16#...)	Explanation
8136	Access to the memory area at the SRC_ARRAY parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface of an FB or FC.
BufferPos	DINT	In the "Temp" section of the block interface of an FB or FC.
Error	INT	
Label	STRING[4]	

The following table shows the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

The following example shows how the instruction works:

SCL

```
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                           DEST_VARIABLE := "Target".Client,
                           POS := #BufferPos);

#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                           DEST_VARIABLE := #Label,
                           POS := #BufferPos);
```

SCL

```
IF #Label = 'arti' THEN
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                           DEST_VARIABLE := "Target".Article[#DeliverPos],
                           POS := #BufferPos);
ELSIF #Label = 'Bill' THEN
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                           DEST_VARIABLE := "Target".Bill[#DeliverPos],
                           POS := #BufferPos);
;
ELSE
;
END_IF;
```

The "Deserialize" instruction deserializes the sequential representation of the customer data from the "Buffer" tag and writes it to the "Target" tag. The #BufferPos operand stores the index of the first byte based on the number of bytes occupied by the converted customer data.

The "Deserialize" instruction deserializes the sequential representation of the separator sheet, which was stored in the sequential representation after the customer data, from the "Buffer" tag and writes the characters to the #Label operand. The characters are compared using comparison instructions "arti" and "Bill". If the comparison for "arti" = TRUE, these are article data that have been deserialized and written to the "Target" tag. If the comparison for "Bill" = TRUE, these are billing data that have been deserialized and written to the "Target" tag.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of PLC data types (UDT) (Page 273)

Structure of an ARRAY tag (Page 287)

Structure of a STRUCT tag (Page 280)

Structure of a STRING tag (Page 266)

Basics of SCL (Page 7293)

Serialize: Serialize

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT), STRUCT or ARRAY of <data type> to a sequential representation without losing parts of their structure.

You use the instruction to temporarily save multiple structured data items from your program in a buffer, which should preferably be in a global data block, and send them to another CPU. The memory area in which the converted data is stored must have the ARRAY of BYTE or ARRAY of CHAR data type and be declared with standard access in version 1.0. Optimized

data is also permitted as of version 2.0. Fill data of the source data area is undefined in the target array. These can be fill bytes or fill bits of a data area (e.g. ARRAY, STRUCT or PLC data type (UDT)) as well as the characters of a string currently not in use.

The capacity of the standard memory area is 64 KB. Structures that are larger than 64 KB in accordance with the rules for standard memory areas cannot be serialized if the operand at the DEST_ARRAY parameter is located in a standard memory area.

It is recommended to define the low limit of the ARRAY with "0", because then the index within ARRAY corresponds to the value of the POS parameter, e.g. ARRAY[0] = POS 0. The description and example below have been formed based on this.

The operand at the POS parameter contains information about the number of bytes used by the converted data.

If you want to send a single PLC data type (UDT), STRUCT or ARRAY of <data type>, you can directly call the instruction "TSEND: Send data via communication connection".

Note

Comparison of structures

To compare structures, they have to be serialized first. Use a comparison expression instead.

You can find additional information here: Relational expressions (Page 7297)

Size of the memory area

The alignment rules mean that simple structures in the optimized memory area do not contain filling bytes. This results in a structure in the optimized memory area being smaller than in the standard memory area. ARRAYs of structures and structures in which other structures are located contain filling bytes. There is no general rule as to the memory area in which a composed structure requires more space.

Applies to CPUs of the S7-1500 series:

For a block with the block property "Optimized block access", the length of the data type BOOL depends on the data type that follows. This means if a BYTE follows, for example, the data type BOOL has a length of 1 byte. When a WORD follows, for example, the data type BOOL has a length of 2 bytes. A structure that consists largely of the data type BOOL can therefore be larger in the optimized memory area than in the standard memory area. Composed structures with a low proportion of BOOL data types are smaller in the optimized memory area than in the standard memory area.

We therefore recommend that the source data area for the serialization starts with the large data types and ends with Boolean elements. This greatly reduces the filling with filling bits.

Optimized memory area

To serialize larger structures, you can declare the memory area with optimized access for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. The sequential representation remains unchanged, as for a standard memory area.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC_VARIABLE	Input	VARIANT	D, block interface of an FB	Pointer to a tag of the STRUCT, ARRAY or PLC data type (UDT) to be serialized.
DEST_ARRAY	InOut	VARIANT	D, block interface of an FB (the sections Input, Output, Static and Temp are possible.)	Pointer to an ARRAY of BYTE or ARRAY of CHAR in which the generated data stream is saved.
POS	InOut	DINT	I, Q, M, D, L	The operand at the POS parameter stores the index of the first byte based on the total number of bytes that the converted customer data has occupied. The POS parameter is calculated zero-based.
Function value		INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains a ZERO pointer.
8151	No valid reference at the SRC_VARIABLE parameter
8236	The tag at the SRC_ARRAY parameter is not in a block with standard access.
8250	A NULL pointer was transferred at parameter DEST_ARRAY.
8251	No valid reference at the DEST_ARRAY parameter
8253	The tag at the parameter DEST_ARRAY does not provide enough memory space for the content of the tag at the parameter SRC_VARIABLE. Due to the input value of the tag at parameter POS the available memory space is reduced. The input value of the parameter POS decides where to begin within the tag at the parameter DEST_ARRAY.
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at the POS parameter is outside the limits of the array.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

For a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0, the following error code has a different meaning:

Error code* (W#16#...)	Explanation
8236	Access to the memory area at the DEST_ARRAY parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

SCL

```
#Tag_RetVal := Serialize(SRC_VARIABLE := "Source".Client,
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);

#Label := STRING_TO_WSTRING('arti');

#Tag_RetVal := Serialize(SRC_VARIABLE := #Label,
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);

#Tag_RetVal := Serialize(SRC_VARIABLE := "Source".Article[#DeliverPos],
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);
```

The "Serialize" instruction serializes the customer data from the "Source" tag and writes the sequential representation to the "Buffer" tag. The index of the next unwritten byte of the "Buffer".Field operand is saved in the #BufferPos operand.

A kind of separator sheet is inserted now to make it easier to deserialize the sequential representation later. The "Move character string" instruction moves the "arti" characters to the #Label operand. The "Serialize" instruction writes these characters after the customer data to the "Buffer" tag. The value of the #BufferPos operand is incremented accordingly.

The "Serialize" instruction serializes the data of a specific article, which is calculated in runtime, from the "Source" tag and writes it in sequential representation to the "Buffer" tag after the "arti" characters.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface

Operand	Data type	Declaration
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table shows the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of PLC data types (UDT) (Page 273)

Structure of an ARRAY tag (Page 287)

Structure of a STRUCT tag (Page 280)

Structure of a STRING tag (Page 266)

Basics of SCL (Page 7293)

MOVE_BLK: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source range) to another memory area (target range). The number of elements to be moved to the target range is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the first element in the source area.

The instruction can only be executed if the source range and the target range have the same data type.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at the IN parameter or OUT parameter.

Note

Use of ARRAYs

The instruction copies the contents from the defined element n elements (n = depending on the value at the parameter COUNT) from the source range to the target range, starting at the specified index.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P	Number of elements to be copied from the source range to the target range
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the target range to which the contents of the source range are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
MOVE_BLK(IN := #a_array[2],
         COUNT := "Tag_Count",
         OUT => #b_array[1]);
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The data type of the "a_array" operand is ARRAY[0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the "b_array" operand is ARRAY[0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element.

Note

You can find more information on the MOVE_BLK instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

MOVE_BLK_VARIANT: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source range) to another memory area (target range). You can copy a complete ARRAY or elements of an ARRAY to another ARRAY of the same data type. The size (number of elements) of source and destination ARRAY may be different. You can copy multiple or single elements within an ARRAY.

The number of elements that are to be copied must not exceed the selected source range or target range.

If you use the instruction at the time the block is being created, the ARRAY does not yet have to be known, as the source and the target are transferred per VARIANT.

Counting at the parameters SRC_INDEX and DEST_INDEX always begins with the low limit "0", regardless of the later declaration of the ARRAY.

The instruction is not executed if more data is copied than is made available.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC	Input [2)]	VARIANT (which points to an ARRAY or an individual ARRAY element), ARRAY of <Data_type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L	Number of elements which are copied Assign the value "1" to the parameter COUNT, if no ARRAY is specified at parameter SRC or at parameter DEST.
SRC_INDEX	Input	DINT	I, Q, M, D, L	Defines the first element to be copied: • The SRC_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter SRC, the integer at parameter SRC_INDEX specifies the first element within the source area from which it is to be copied. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter SRC or only one single element of an ARRAY is specified, then assign the value "0" at parameter SRC_INDEX.

Parameters	Declaration	Data type	Memory area	Description
DEST_INDEX	Input	DINT	I, Q, M, D, L	Defines the start of the destination memory area: • The DEST_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter DEST, the integer at parameter DEST_INDEX specifies the first element within the target range that is to be copied into. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter DEST, then assign the value "0" at parameter DEST_INDEX.
DEST	Output [1)]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Destination area into which the contents of the source block are copied.
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information
1) The DEST parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface. 2) With the SRC parameter, the data types BOOL and ARRAY of BOOL are not permitted.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data types do not correspond. Instead of an ARRAY of Struct, use an ARRAY of PLC data type (UDT).
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at the SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value
8382	The value at the SRC_INDEX parameter is less than zero.
8383	The value at the SRC_INDEX is outside the high limit of the ARRAY.
8482	The value at the DEST_INDEX parameter is less than zero.
8483	The value at the DEST_INDEX is outside the high limit of the ARRAY.

Error code* (W#16#...)	Explanation
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at the DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MOVE_BLK_VARIANT(SRC := #SrcField,
                                 COUNT := "Tag_Count",
                                 SRC_INDEX := "Tag_Src_Index",
                                 DEST_INDEX := "Tag_Dest_Index",
                                 DEST => #DestField);
```

The following table shows how the instruction works using specific operand values:

Parameters	Declaration in the block interface	Operand	Value
SRC	Input	#SrcField	The local operand #SrcField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[0..10] of "MOVE_UDT")
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	The local operand #DestField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[10..20] of "MOVE_UDT")

Two elements are moved from the source range, starting from the fourth element of the ARRAY [0..10] of MOVE_UDT, to the target range. The copies are pasted in the ARRAY [10..20] of MOVE_UDT starting from the fourth element.

Note

You can find more information on the MOVE_BLK_VARIANT instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

VariantGet: Read out VARIANT tag value (Page 1795)

Basics of SCL (Page 7293)

UMOVE_BLK: Move block uninterruptible

Description

You use the "Move block uninterruptible" instruction to move the content of a memory area (source range) to another memory area (target range). The instruction cannot be interrupted. The number of elements to be moved to the target range is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the first element in the source area.

The instruction can only be executed if the source range and the target range have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at the IN parameter or OUT parameter.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Note

Use of ARRAYs

The instruction copies the contents from the defined element n elements (n = depending on the value at the parameter COUNT) from the source range to the target range, starting at the specified index.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P	Number of elements to be copied from the source range to the target range
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	The first element of the target range to which the contents of the source range are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
UMOVE_BLK(IN := #a_array[2],
          COUNT := "Tag_Count",
          OUT => #b_array[1]);
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The data type of the "a_array" operand is ARRAY[0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the "b_array" operand is ARRAY[0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The move operation cannot be interrupted by other operating system activities.

Note

You can find more information on the UMOVE_BLK instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (target range) with the value of the IN input. The target range is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is moved to the target range as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source range and the target range have the same data type.

The maximum number of elements changed is the number of elements in the ARRAY or structure. If you copy more data than there are elements at the OUT output, you will get an unintended result.

Note

Use of ARRAYs

The instruction reads the content from the selected element from the source range and copies this content n-times (n = depending on the value at the parameter COUNT) to the destination, starting at the specified index.

Filling structures

As well as the elements of an ARRAY, you can also fill multiple elements of a structure (STRUCT, PLC data type) with the same value. The structure whose elements you want to fill must only contain individual elements of the same elementary data type. The structure itself can, however, be embedded in another structure.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Binary numbers, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P	Element used to fill the target range
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P	Number of repeated move operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	Address in target range from which filling starts

For additional information on valid data types, refer to "See also".

Example with an ARRAY

The following example shows how the instruction works when you want to fill an ARRAY:

SCL

```
FILL_BLK(IN := #FillValue,
         COUNT := "Tag_Count",
         OUT => #TargetArea[1]);
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	FillValue	The data type of the operand is INT.
COUNT	Tag_Count	3
OUT	TargetArea	The data type of the operand TargetArea is ARRAY[1..5] of INT. It consists of five elements of data type INT.

The instruction copies the value of the operand #FillValue to the #TargetArea output tag three times, starting with the first element.

Examples with a structure

The following examples show how the instruction works when you want to fill a structure:

Create a global data block with the following elements:

Data_block_1		Data type
MyStruct1		STRUCT
	Member_1	INT
	Member_2	INT
	Member_3	INT
	Member_4	INT
MyStruct2		STRUCT

Data_block_1					Data type
	SubArray				ARRAY[1..2] of STRUCT
		SubArray[1]			STRUCT
			NestedStruct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT
		SubArray[2]			STRUCT
			Nested-Struct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT

Generate the following program code to address the MyStruct1 tag:

SCL

```
FILL_BLK(IN := 10,
         COUNT := 2,
         OUT => "Data_block_1".MyStruct1.Member_2);
```

Generate the following program code to address the MyStruct2 tag:

SCL

```
FILL_BLK(IN := 10,
         COUNT := 2,
        OUT => "Data_block_1".MyStruct2.SubArray[1].NestedStruct.Member_2);
```

In each of the two examples, the value 10 of the IN parameter is copied twice to the operand at the OUT parameter, starting with the element Member_2. This means that the value 10 is copied to the elements Member_2 and Member_3. The other two elements, Member_1 and Member_4, are not changed.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

UFILL_BLK: Fill block uninterruptible

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (target range) with the value of the IN input. The instruction cannot be interrupted. The target range is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is moved to the target range as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source range and the target range have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The maximum number of elements changed is the number of elements in the ARRAY or structure. If you copy more data than there are elements at the OUT output, you will get an unintended result.

Note

Use of ARRAYs

The instruction reads the content from the selected element from the source range and copies this content n-times (n = depending on the value at the parameter COUNT) to the destination, starting at the specified index.

You use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Filling structures

As well as the elements of an ARRAY, you can also fill multiple elements of a structure (STRUCT, PLC data type) with the same value. The structure whose elements you want to fill must only contain individual elements of the same elementary data type. The structure itself can, however, be embedded in another structure.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	I, Q, M, D, L, P	Element used to fill the target range
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P	Number of repeated move operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, TOD, DATE, CHAR, WCHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, WCHAR, TOD, LTOD	D, L	Address in target range from which filling starts

For additional information on valid data types, refer to "See also".

Example with an ARRAY

The following example shows how the instruction works when you want to fill an ARRAY:

SCL

```
UFILL_BLK(IN := #FillValue,
          COUNT := "Tag_Count",
          OUT => #TargetArea[1]);
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	FillValue	The data type of the operand is INT.
COUNT	Tag_Count	3
OUT	TargetArea	The data type of the operand TargetArea is ARRAY[1..5] of INT. It consists of five elements of data type INT.

The instruction copies the value of the operand #FillValue to the #TargetArea output tag three times, starting with the first element. The move operation cannot be interrupted by other operating system activities.

Examples with a structure

The following examples show how the instruction works when you want to fill a structure:

Create a global data block with the following elements:

Data_block_1					Data type
MyStruct1					STRUCT
	Member_1				INT
	Member_2				INT
	Member_3				INT
	Member_4				INT
MyStruct2					STRUCT
	SubArray				ARRAY[1..2] of STRUCT
		SubArray[1]			STRUCT
			NestedStruct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT
		SubArray[2]			STRUCT
			Nested-Struct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT

Generate the following program code to address the MyStruct1 tag:

SCL

```
UFILL_BLK(IN := 10,
          COUNT := 2,
          OUT => "Data_block_1".MyStruct1.Member_2);
```

Generate the following program code to address the MyStruct2 tag:

SCL

```
UFILL_BLK(IN := 10,
          COUNT := 2,
          OUT => "Data_block_1".MyStruct2.SubArray[1].NestedStruct.Member_2);
```

In each of the two examples, the value 10 of the IN parameter is copied twice to the operand at the OUT parameter, starting with the element Member_2. This means that the value 10 is copied to the elements Member_2 and Member_3. The other two elements, Member_1 and Member_4, are not changed.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SCATTER: Parse the bit sequence into individual bits

Description

The "Parse the bit sequence into individual bits" instruction parses a tag of the BYTE, WORD, DWORD or LWORD data type into individual bits and saves them in an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements.

Note

Multi-dimensional ARRAY of BOOL

With the "Parse the bit sequence into individual bits" instruction, the use of a multidimensional ARRAY of BOOL is not permitted.

Note

Length of the ARRAY, STRUCT or PLC data type

The ARRAY, the anonymous STRUCT or the PLC data type must have exactly the number of elements that is specified by the bit sequence.

This means for the data type BYTE, for example, the ARRAY, STRUCT or the PLC data type must have exactly 8 elements (WORD = 16, DWORD = 32 and LWORD = 64).

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

This way you can, for example, parse a status word, and read and change the status of the individual bits using the index. Using GATHER, you can merge the bits once again into a bit sequence.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- If the ARRAY, STRUCT or PLC data type does not provide enough BOOL elements.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD	I, Q, M, D, L	Bit sequence that is parsed. The values must not be located in the I/O area or in the DB of a technology object.
OUT	Output	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	I, Q, M, D, L	ARRAY, STRUCT or PLC data type in which the individual bits are stored

You can find additional information on valid data types under "See also".

Example with an ARRAY

Create the following tags in the block interface:

Tag	Section	Data type
SourceWord	Input	WORD
DestinationArray	Output	ARRAY[0..15] of BOOL

The following example shows how the instruction works:

SCL

```
SCATTER(IN := #SourceWord,
        OUT => #DestinationArray);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceWord	WORD (16 bits)
OUT	DestinationArray	The operand "DestinationArray" is of the data type ARRAY[0..15] of BOOL. It consists of 16 elements and is therefore just as large as the WORD that is to be parsed.

The operand #SourceWord of the data type WORD is parsed into its individual bits (16) and assigned to the individual elements of the #DestinationArray operand.

Example with a PLC data type (UDT)

Create the following PLC data type "myBits":

myBits		
	Name	Data type
1	GeneralError	Bool
2	DeviceError	Bool
3	CommError	Bool
4	myError1	Bool
5	myError2	Bool
6	myError3	Bool
7	myError4	Bool
8	myError5	Bool
9	myError6	Bool
10	myError7	Bool
11	myError8	Bool
12	myError9	Bool
13	myError10	Bool
14	myError11	Bool
15	myError12	Bool
16	myError13	Bool

Create the following tags in the block interface:

Tag	Section	Data type
SourceWord	Input	WORD
DestinationUDT	Output	"myBits"

The following example shows how the instruction works:

SCL

```
SCATTER(IN := #SourceWord,
        OUT => #DestinationUDT);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceWord	WORD (16 bits)
OUT	DestinationUDT	The "DestinationUDT" operand has the PLC data type (UDT). It consists of 16 elements and is therefore just as large as the WORD that is to be parsed.

The #SourceWord operand of the data type WORD is parsed into its individual bits (16) and assigned to the individual elements of the #DestinationArray operand.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SCATTER_BLK: Parse elements of an ARRAY of bit sequence into individual bits

Description

The "Parse elements of an ARRAY of bit sequence into individual bits" instruction parses one or more elements of an ARRAY of BYTE, WORD, DWORD or LWORD into individual bits and saves them in an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements. At the COUNT_IN parameter you specify how many elements of the source ARRAY are going to be parsed. The source ARRAY at the IN parameter may have more elements than specified at the COUNT_IN parameter. The ARRAY of BOOL, the anonymous STRUCT or the PLC data type must have sufficient elements to save the bits of the parsed bit sequences. However, the destination memory area may also be larger.

Note

Multi-dimensional ARRAY of BOOL

If the ARRAY is a multi-dimensional ARRAY of BOOL, the filling bits of the dimensions contained are counted as well even if they were not explicitly declared.

Example 1: An ARRAY[1..10,0..4,1..2] of BOOL is handled like an ARRAY[1..10,0..4,1..8] of BOOL or like an ARRAY[0..399] of BOOL.

Example 2: At the IN parameter, an ARRAY[0..5] of WORD (sourceArrayWord[2]) is interconnected. The COUNT_IN parameter has the value "3". At the OUT parameter, an ARRAY[0..1,0..5,0..7] of BOOL (destinationArrayBool[0,0,0]) is interconnected. Both the array at the IN parameter and at the OUT parameter has a size of 96 bits. The ARRAY of WORD is parsed into 48 individual bits.

Note

If the ARRAY low limit of the target ARRAY is not "0", note the following:

For performance reasons the index must always start at a BYTE, WORD, DWORD or LWORD limit. This means the index must be calculated starting at the low limit of the ARRAY. The following formula is used as basis for this calculation:

Valid indices = ARRAY low limit + n(number of bit sequences) × number of bits of the desired bit sequence

For an ARRAY[-2..45] of BOOL and the bit sequence WORD the calculation looks as follows:

- Valid index (-2) = -2 + 0 × 16
- Valid index (14) = -2 + 1 × 16
- Valid index (30) = -2 + 2 × 16

You can find an example described below.

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

This way you can, for example, parse status words, and read and change the status of the individual bits using the index. Using GATHER, you can merge the bits once again into a bit sequence.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- The source ARRAY has fewer elements than specified at the COUNT_IN parameter.
- The index of the destination ARRAY does not start at a BYTE, WORD, DWORD or LWORD limit. In this case, no result is written to the ARRAY of BOOL.
- The ARRAY[*] of BOOL, STRUCT or PLC data type does not provide the required number of elements. In this case as many bit sequences as possible are parsed and written to the ARRAY of BOOL, the anonymous STRUCT or the PLC data type. The remaining bit sequences are no longer taken into account.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Element of an ARRAY[*] of <bit sequence>	Element of an ARRAY[*] of <bit sequence>	I, Q, M, D, L	ARRAY of <bit sequence> that is parsed. The values must not be located in the I/O area or in the DB of a technology object.
COUNT_IN	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Counter for the number of elements of the source ARRAY that are going to be parsed. The value must not be in the I/O area or in the database of a technology object.
OUT	Output	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	I, Q, M, D, L	ARRAY, STRUCT or PLC data type in which the individual bits are stored

You can find additional information on valid data types under "See also".

Example of a destination ARRAY with the low limit "0"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayWord	Input	ARRAY[0..5] of WORD
CounterInput		UDINT
DestinationArrayBool	Output	ARRAY[0..95] of BOOL

The following example shows how the instruction works:

SCL

```
SCATTER_BLK(IN := #SourceArrayWord[2],
            COUNT_IN := #CounterInput,
            OUT => #DestinationArrayBool[0]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayWord[2]	ARRAY[0..5] of WORD (96 bits can be parsed.)
COUNT_IN	CounterInput = 3	UDINT3 (3 WORDs or 48 bits are to be parsed. This means at least 48 bits must be available in the destination ARRAY.)
OUT	DestinationArrayBool[0]	The operand "DestinationArray-Bool" is of the data type AR-RAY[0..95] of BOOL. This means it provides 96 BOOL ele-ments.

The 3rd, 4th and 5th WORD of the #SourceArrayWord operand is parsed into its individual bits (48) and as of the 1st element assigned to the individual elements of the #DestinationArrayBool operand.

Example of a destination ARRAY with the low limit "-2"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayWord	Input	ARRAY[0..5] of WORD
CounterInput		UDINT
DestinationArrayBool	Output	ARRAY[-2..93] of BOOL

The following example shows how the instruction works:

SCL

```
SCATTER_BLK(IN := #SourceArrayWord[2],
            COUNT_IN := #CounterInput,
            OUT => #DestinationArrayBool[14]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayWord[2]	ARRAY[0..5] of WORD (96 bits can be parsed.)
COUNT_IN	CounterInput = 3	UDINT3 (3 WORDs or 48 bits are to be parsed. This means at least 48 bits must be available in the destination ARRAY.)
OUT	DestinationArrayBool[14]	The operand "DestinationArray-Bool" is of the data type AR-RAY[-2..93] of BOOL. This means it provides 96 BOOL ele-ments.

The 3rd, 4th and 5th WORD of the #SourceArrayWord operand is parsed into its individual bits (48) and as of the 16th Element assigned to the individual elements of the #DestinationArrayBool operand.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

GATHER: Merge individual bits into a bit sequence

Description

The "Merge individual bits into a bit sequence" instruction merges the bits from an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements into a bit sequence. The bit sequence is saved in a tag of the data type BYTE, WORD, DWORD or LWORD.

Note

Multi-dimensional ARRAY of BOOL

With the "Merge individual bits into a bit sequence" instruction, the use of a multidimensional ARRAY of BOOL is not permitted.

Note

Length of the ARRAY, STRUCT or PLC data type

The ARRAY, STRUCT or the PLC data type must have exactly the number of elements that is specified by the bit sequence.

This means for the data type BYTE, for example, the ARRAY, anonymous STRUCT or the PLC data type must have exactly 8 elements (WORD = 16, DWORD = 32 and LWORD = 64).

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- If the ARRAY, the anonymous STRUCT or the PLC data type (UDT) has fewer or more BOOL elements than specified by the bit sequence, they are not transferred and the ENO returns signal state "0". The ENO also returns "0" if fewer than the necessary number of bits is available.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	I, Q, M, D, L	ARRAY, STRUCT or PLC data type, the bits of which are merged into a bit sequence. The values must not be located in the I/O area or in the DB of a technology object.
OUT	Output	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD	I, Q, M, D, L	Merged bit sequence, saved in a tag

You can find additional information on valid data types under "See also".

Example with an ARRAY

Create the following tags in the block interface:

Tag	Section	Data type
SourceArray	Input	ARRAY[0..15] of BOOL
DestinationWord	Output	WORD

The following example shows how the instruction works:

SCL

```
GATHER(IN := #SourceArray,
       OUT => #DestinationWord);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArray	The operand "SourceArray" is of the data type ARRAY[0..15] of BOOL. It consists of 16 elements and is therefore just as large as the WORD in which the bits are to be merged.
OUT	DestinationWord	WORD (16 bits)

The bits of the #SourceArray operand are merged into a WORD.

Example with a PLC data type (UDT)

Create the following PLC data type "myBits":

myBits		
	Name	Data type
1	GeneralError	Bool
2	DeviceError	Bool
3	CommError	Bool
4	myError1	Bool
5	myError2	Bool
6	myError3	Bool
7	myError4	Bool
8	myError5	Bool
9	myError6	Bool
10	myError7	Bool
11	myError8	Bool
12	myError9	Bool
13	myError10	Bool
14	myError11	Bool
15	myError12	Bool
16	myError13	Bool

Create the following tags in the block interface:

Tag	Section	Data type
SourceUDT	Input	"myBits"
DestinationWord	Output	WORD

The following example shows how the instruction works:

SCL

```
GATHER(IN := #SourceUDT,
       OUT => #DestinationWord);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceUDT	The "SourceUDT" operand has the PLC data type (UDT). It consists of 16 elements and is therefore just as large as the WORD in which the bits are to be merged.
OUT	DestinationWord	WORD (16 bits)

The bits of the #SourceUDT operand are merged into a WORD.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

GATHER_BLK: Merge individual bits into multiple elements of an ARRAY of bit sequence

Description

The "Merge individual bits into multiple elements of an ARRAY of bit sequence" instruction merges the bits from an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements into one or multiple elements of an ARRAY of <bit sequence>. At the COUNT_OUT parameter you specify how many elements of the destination ARRAY are going to be written. With this step you also implicitly specify how many elements of the ARRAY of BOOL, the anonymous STRUCT or the PLC data type are required. The destination ARRAY at the OUT parameter may have more elements than specified at the COUNT_OUT parameter. The ARRAY of <bit sequence> must have sufficient elements to save the bits that are going to be merged. However, the destination ARRAY may also be larger.

Note

Multi-dimensional ARRAY of BOOL

If the ARRAY is a multi-dimensional ARRAY of BOOL, the filling bits of the dimensions contained are counted as well even if they were not explicitly declared.

Example 1: An ARRAY[1..10,0..4,1..2] of BOOL is handled like an ARRAY[1..10,0..4,1..8] of BOOL or like an ARRAY[0..399] of BOOL.

Example 2: At the OUT parameter, an ARRAY[0..5] of WORD (sourceArrayWord[2]) is interconnected. The COUNT_IN parameter has the value "3". At the IN parameter, an ARRAY[0..1,0..5,0..7] of BOOL (destinationArrayBool[0,0,0]) is interconnected. Both the array at the IN parameter and at the OUT parameter has a size of 96 bits. 48 individual bits are merged from the ARRAY of BOOL.

Note

If the ARRAY low limit of the source ARRAY is not "0", note the following:

For performance reasons the index must always start at a BYTE, WORD, DWORD or LWORD limit. This means the index must be calculated starting at the low limit of the ARRAY. The following formula is used as basis for this calculation:

Valid indices = ARRAY low limit + n(number of bit sequences) × number of bits of the desired bit sequence

For an ARRAY[-2..45] of BOOL and the bit sequence WORD the calculation looks as follows:

- Valid index (-2) = -2 + 0 × 16
- Valid index (14) = -2 + 1 × 16
- Valid index (30) = -2 + 2 × 16

You can find an example described below.

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1200 series as of firmware version >4.2, and for a CPU of the S7-1500 series as of firmware version 2.1.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- The index of the source ARRAY does not start at a BYTE, WORD, DWORD or LWORD limit. In this case, no result is written to the ARRAY of <bit sequence>.
- The ARRAY[*] of <bit sequence> does not provide the required number of elements. In this case as many bit sequences as possible are merged and written to the ARRAY of <bit sequence>. The remaining bits are no longer taken into account.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	I, Q, M, D, L	ARRAY of BOOL, STRUCT or PLC data type whose bits are merged (source ARRAY) The values must not be located in the I/O area or in the DB of a technology object.
COUNT_OUT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Counter how many elements of the target ARRAY are to be described. The value must not be in the I/O area or in the database of a technology object.
OUT	Output	Element of an ARRAY[*] of <bit sequence>	Element of an ARRAY[*] of <bit sequence>	I, Q, M, D, L	ARRAY of <bit sequence> to which the bits are saved (destination ARRAY)

You can find additional information on valid data types under "See also".

Example of a source ARRAY with the low limit "0"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayBool	Input	ARRAY[0..95] of BOOL
CounterOutput		UDINT
DestinationArrayWord	Output	ARRAY[0..5] of WORD

The following example shows how the instruction works:

SCL

```
GATHER_BLK(IN := #SourceArrayBool[0],
```

SCL

```
            COUNT_OUT := #CounterOutput,
            OUT => #DestinationArrayWord[2]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayBool[0]	The operand "SourceArrayBool" is of the data type ARRAY[0..95] of BOOL. This means it provides 96 BOOL elements that can merged into words once again.
COUNT_OUT	CounterOutput = 3	UDINT3 (3 words are to be written. This means 48 bits must be available in the source ARRAY.)
OUT	DestinationArrayWord[2]	The operand "DestinationArrayWord" is of the data type ARRAY[0..5] of WORD. This means 6 WORD elements are available.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. As of the 1st element of the #SourceArrayBool operand, 48 bits are merged in the #DestinationArrayWord operand. The starting point in the destination ARRAY is the 3rd element. This means that the first 16 bits are written into the 3rd word, the second 16 nits into the 4th word and the third 16 bits into the 5th word of the destination ARRAY. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

Example of a source ARRAY with the low limit "-2"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayBool	Input	ARRAY[-2..93] of BOOL
CounterOutput		UDINT
DestinationArrayWord	Output	ARRAY[0..5] of WORD

The following example shows how the instruction works:

SCL

```
GATHER_BLK(IN := #SourceArrayBool[14],
            COUNT_OUT := #CounterOutput,
            OUT => #DestinationArrayWord[2]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayBool[14]	The operand "SourceArrayBool" is of the data type ARRAY[-2..93] of BOOL. Because the starting point is the 16th element, only 80 BOOL elements that can be merged into words again are available.
COUNT_OUT	CounterOutput = 3	UDINT3 (3 words are to be written. This means 48 bits must be available in the source ARRAY.)
OUT	DestinationArrayWord[2]	The operand "DestinationArrayWord" is of the data type ARRAY[0..5] of WORD. This means 6 WORD elements are available.

If the operand #Enable returns the signal state "1" at the enable input EN, the instruction is executed. Starting with the 16th element of the #SourceArrayBool operand, 48 bits are merged into the #DestinationArrayWord operand. The starting point in the destination ARRAY is the 3rd element. This means the first 16 bits of the source ARRAY are ignored. The second 16 Bits are written into the 3rd word, the third 16 bits into the 4th word and the fourth 16 bits into the 5th word of the destination ARRAY. The remaining 64 bits of the source ARRAY are not taken into account either. If an error occurs during the execution of the instruction, the operand #EnableOut returns the signal state "0" at the enable output ENO.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

AssignmentAttempt: Attempt assignment of a VARIANT to a reference (S7-1500)

Description

With the "AssignmentAttempt" instruction, you attempt to assign a VARIANT tag to a reference tag. The data type of a reference tag is specified at the time of the declaration, while the data type of a VARIANT tag is determined during runtime. An implicit data type conversion is not permitted with reference tags. To assign the two data types to each other therefore, use the assignment attempt.

With the assignment attempt, a check is made at runtime to find out whether the VARIANT tag is of the correct data type. If this is the case, the instruction is executed. If the instruction is executed successfully, there is a valid reference in the target tag, otherwise a NULL.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC	Input	VARIANT	• Block interface of an FC: Input, Output, InOut, Temp • Block interface of an FB: Input, InOut, Temp	Pointer to the source tag whose address will be read out
DST	Output	Reference to: • Bit sequences, except BOOL, • Integers, • Floating-point numbers, • Character strings, • PLC data types (UDT), • System data types (SDT), • ARRAYs of the named data types	• Block interface of an FC: Input, Output, Temp • Block interface of an FB: Temp	Reference to which the address of the source tag will be transferred

You will find additional information on valid data types under "See also".

Rules

The following rules apply to the assignment attempt: VARIANT tags that do not match these rules return the value "NULL" at runtime.

- The VARIANT must point to an address in the optimized memory area.
- The VARIANT must not point to an address in a temporary memory area.
- If you want to assign a VARIANT to a reference to an ARRAY the following rules apply:
  - The VARIANT tag must point to an ARRAY whose limits exactly match with those of the declared reference. A VARIANT tag which points to an ARRAY [0..9], does not match a tag REF_TO ARRAY[1..10].
  - Moreover you should compile the blocks that form the value of the VARIANT tag once in a CPU of the S7-1500 series, firmware version V2.5.

Example

The following example shows how the instruction works:

In the interface of the block, the VARIANT "myVariant" and the reference tag "myReference" were declared.

	Name	Data type
	Input	
	Output	
	InOut	
	myVariant	Variant
	Static	
	Temp	
	myReference	REF_TO Int
	Constant	

In the program code an attempt is made to assign "myVariant" to the reference tag "myReference". If "myVariant" has the data type "Int" at runtime, "myReference" contains a valid reference to the source tag of the VARIANT, otherwise NULL. If the next query for "NOT_NULL" is true, it means that the assignment was successful and the value "10" can be written to the target tag.

```
1  #myReference ?= #myVariant;
2
3  IF #myReference <> NULL THEN
4     #myReference^ := 10;
5  END_IF;
```

See also

Basic information about references (Page 300)

Assignment attempt from VARIANT to a reference (Page 314)

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

SWAP: Swap

Description

You can use the "Swap" instruction to change the arrangement of the bytes of an input value and save the result in the specified operand.

The following figure shows how the bytes of an operand of the DWORD data type are swapped using the "Swap" instruction:

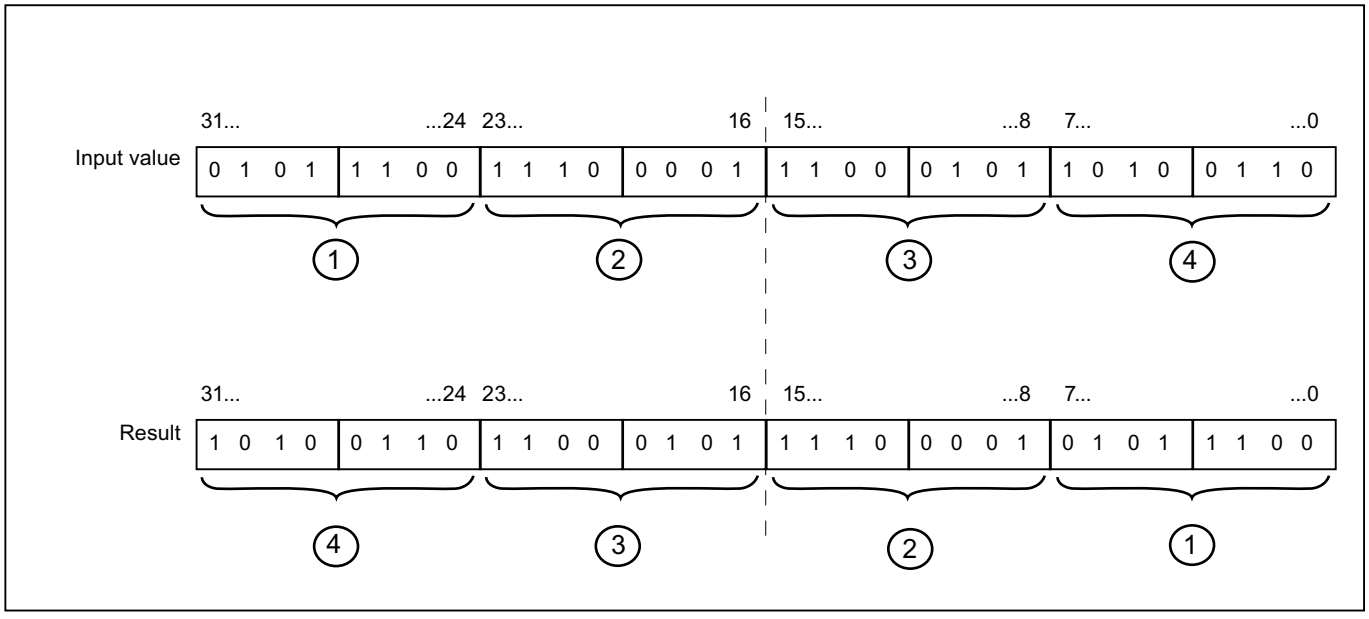

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Expression>	Input	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P	Input value
Function value		WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SWAP("Tag_Value");
```

The result of the instruction is returned as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	0000 1111 0101 0101
Tag_Result	0101 0101 0000 1111

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ARRAY DB (S7-1500)

ReadFromArrayDB: Read from array data block (S7-1500)

Description

The instruction "Read from ARRAY data block" is used to read the element from a data block of the ARRAY DB block type to which the index references, and write the value of the element to the target range.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P	Element in the DB that is read. The specification can be a constant, a global tag or an indexed value.
VALUE	Output [1)]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Value that is read and output
Function value (RET_VAL)		INT	I, Q, M, D, L	Result of the instruction
1) The VALUE parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, write protected, or is located in load memory.
8135	The ARRAY data block contains invalid values.
8154	The data block has the incorrect data type.
8282	The value at the INDEX parameter is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8452	Code generation error
8453	There are two possible causes of the error: • The size of the VALUE parameter does not match the element length in the ARRAY data block. • The two tags are not in a memory area with optimized access. For additional information on the memory area access types, refer to: Basics of block access (Page 55)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"TagResult" := ReadFromArrayDB(DB := "ArrayDB",
                               INDEX := 2,
                               VALUE => "TargetField");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type Array [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	TargetField	The "TargetField" operand is a global tag of the data type INT.

The 2nd element is read from "ArrayDB" and written to the "TargetField" operand.

See also

Overview of the valid data types (Page 231)

Example of the use of ARRAY data blocks (Page 44)

Basics of SCL (Page 7293)

WriteToArrayDB: Write to array data block (S7-1500)

Description

The instruction "Write to ARRAY data block" is used to write the element to which the index references to a data block of the ARRAY DB block type.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P	Element in the DB to which data is written. The specification can be a constant, a global tag or an indexed value.
VALUE	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Value to be written
Function value (RET_VAL)		INT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, or is located in load memory.

Error code* (W#16#...)	Explanation
8134	The data block is write protected.
8135	The data block is not an ARRAY data block.
8154	The data block has the incorrect data type.
8282	The value at the INDEX parameter is outside the limits of the ARRAY.
8350	The data type VARIANT at parameter VALUE provides the value "0".
8352	Code generation error
8353	There are two possible causes of the error: • The size of the VALUE parameter does not match the element length in the ARRAY data block. • The two tags are not in a memory area with optimized access. For additional information on the memory area access types, refer to: Basics of block access (Page 55)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"TagResult" := WriteToArrayDB(DB := "ArrayDB",
                              INDEX := 2,
                              VALUE := "SourceField");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type Array [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	SourceField	The "SourceField" operand is a global tag of the data type INT.

The value of the "SourceField" operand is written to the 2nd element of the ARRAY DB.

See also

Overview of the valid data types (Page 231)

Example of the use of ARRAY data blocks (Page 44)

Basics of SCL (Page 7293)

ReadFromArrayDBL: Read from array data block in load memory (S7-1500)

Description

The instruction "Read from ARRAY data block in load memory" is used to read the element ti which the index references from a data block of the ARRAY DB block type in the load memory and write it to the target range.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = "1": Begin with the reading of the ARRAY DB
DB [1]	Input	DB_ANY	I, Q, M, D, L	ARRAY data block that is read.
INDEX	Input	DINT	I, Q, M, D, L, P	Element in the DB that is read. The specification can be a constant, a global tag or an indexed value.
VALUE [1]	InOut	VARIANT	D (element of a global data block) L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Pointer to the DB in the work memory that is to be read and the value of which is to be written. No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the ERROR parameter.
[1] The data blocks must be created with the "Optimized" block property.				

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at the INDEX parameter is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL" and "WRIT_DBL" in the corresponding descriptions of the instructions.

Example

The following example shows how the instruction works:

SCL

```
"ReadFromArrayDBL_DB"(REQ := "TagReg",
                      DB := "ArrayDB",
                      INDEX := 2,
                      VALUE := "TargetField",
                      BUSY => "TagBusy",
                      DONE => "TagDone",
                      ERROR => "TagError");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type ARRAY [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	TargetField	The "TargetField" operand is a global tag of the data type INT.

Parameters	Operand	Value
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

The instruction is executed when a positive signal edge is detected at the "TagReq" operand. The 2nd element is read from "ArrayDB" and output at the "VALUE" parameter. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. After the instruction has been processed, the "TagDone" operand has the signal state TRUE.

See also

Overview of the valid data types (Page 231)

READ_DBL: Read from data block in the load memory (Page 2714)

WRIT_DBL: Write to data block in the load memory (Page 2718)

Example of the use of ARRAY data blocks (Page 44)

Difference between synchronous and asynchronous instructions (Page 596)

Basics of SCL (Page 7293)

WriteToArrayDBL: Write to array data block in load memory (S7-1500)

Description

The instruction "Write to ARRAY data block in load memory" is used to write the element to which the index references to a data block of the ARRAY DB block type in load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of <Data type>. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = "1": Start writing to the array DB
DB [1)]	Input	DB_ANY	I, Q, M, D, L	ARRAY data block to which data is written

Parameters	Declaration	Data type	Memory area	Description
INDEX	Input	DINT	I, Q, M, D, L, P	Element in the DB to which data is written. The specification can be a constant, a global tag or an indexed value.
VALUE [1]	Input	VARIANT	D (element of a global data block) L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Pointer to the DB in the work memory that is to be read and the value of which is to be written. No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the ERROR parameter.
[1] The data blocks must be created with the "Optimized" block property.				

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8234	The data block is write protected.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at the INDEX parameter is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL" and "WRIT_DBL" in the corresponding descriptions of the instructions.

Example

The following example shows how the instruction works:

SCL

```
"WriteToArrayDBL_DB"(REQ := "TagReg",
                     DB := "ArrayDB",
                     INDEX := 2,
                     VALUE := "SourceField",
                     BUSY => "TagBusy",
                     DONE => "TagDone",
                     ERROR => "TagError");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type ARRAY [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	SourceField	The "SourceField" operand is a global tag of the data type INT.
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

The instruction is executed when a positive signal edge is detected at the "TagReq" operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written to the 2nd element of "ArrayDB". After the instruction has been processed, the "TagDone" operand has the signal state TRUE.

See also

Overview of the valid data types (Page 231)

READ_DBL: Read from data block in the load memory (Page 2714)

WRIT_DBL: Write to data block in the load memory (Page 2718)

Example of the use of ARRAY data blocks (Page 44)

Difference between synchronous and asynchronous instructions (Page 596)

Basics of SCL (Page 7293)

Read/write access

PEEK: Read memory address

Description

The "Read memory address" instruction is used to read a memory address from a standard memory area without specifying a data type.

Use the following syntax to change the data type of the instruction:

PEEK_<data type>();

Use of memory area 16#84: DB

If you use a data block as memory area and this is not yet known during the creation of the program code you have the option of using the EN/ENO mechanism to recognize possible access errors:

SCL

```
#Peeker := PEEK(AREA := 16#84,
                DBNUMBER := 1,
                BYTEOFFSET := 2,
                ENO => ENO);
IF NOT ENO THEN;
     #Peeker := 16#ffff;
END_IF;
```

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#1: Peripheral input (S7-1500 only)
DBNUM-BER	Input	DINT, DB_ANY	D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D	Address to read from Only the 16 least significant bits are used.

Parameters	Declaration	Data type	Memory area	Description
_<data type>		Bit strings default: BYTE	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Bit strings	I, Q, M, D	Result of the instruction

Note

If you read the memory address from the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := PEEK(AREA := "Tag_Area",
                      DBNUMBER := "Tag_DBNumber",
                      BYTEOFFSET := "Tag_Byte");
```

SCL

```
"Tag_Result2" := PEEK_WORD(AREA := "Tag_Area",
                           DBNUMBER := "Tag_DBNumber",
                           BYTEOFFSET := "Tag_Byte");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
Function value	Tag_Result1	Byte value "20"
Function value	Tag_Result2	Word value "20"

The instruction reads the value of address "20" from the "Tag_Byte" operand at data block "5" and returns the result as a function value at the "Tag_Result" operand.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

PEEK_BOOL: Read memory bit

Description

The "Read memory bit" instruction is used to read a memory bit from a standard memory area without specifying a data type.

Use of memory area 16#84: DB

If you use a data block as memory area and this is not yet known during the creation of the program code you have the option of using the EN/ENO mechanism to recognize possible access errors:

SCL

```
#Peeker_BOOL := PEEK_BOOL(AREA := 16#84,
                          DBNUMBER := 1,
                          BYTEOFFSET := 2,
                          BITOFFSET := 1,
                          ENO => ENO);
IF NOT ENO THEN;
     #Peeker_BOOL := 0;
END_IF;
```

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#1: Peripheral input (S7-1500 only)
DBNUM-BER	Input	DINT, DB_ANY	D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D	Address to read from Only the 16 least significant bits are used.

Parameters	Declaration	Data type	Memory area	Description
BITOFFSET	Input	INT	I, Q, M, D	Bit to be read from
Function value		BOOL	I, Q, M, D	Result of the instruction

Note

If you read the memory bit from the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := PEEK_BOOL(AREA := "Tag_Area",
                          DBNUMBER := "Tag_DBNumber",
                          BYTEOFFSET := "Tag_Byte",
                          BITOFFSET := "Tag_Bit");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
Function value	Tag_Result	3

The instruction reads the value of memory bit "3" from the "Tag_Bit" operand at byte "20" of data block "5" and returns the result at the "Tag_Result" operand as function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

POKE: Write memory address

Description

The "Write memory address" instruction is used to write a memory address to a standard memory area without specifying a data type.

Use of memory area 16#84: DB

If you use a data block as memory area and this is not yet known during the creation of the program code you have the option of using the EN/ENO mechanism to recognize possible access errors:

SCL

```
POKE(AREA := 16#84,
     DBNUMBER := 1,
     BYTEOFFSET := 2,
     VALUE := BYTE#7,
     ENO => ENO);
```

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#2: Peripheral output (S7-1500 only)
DBNUM-BER	Input	DINT, DB_ANY	D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D	Address to be written Only the 16 least significant bits are used.
VALUE	Input	Bit sequen-ces	I, Q, M, D	Value to be written

Note

If you write the memory address to the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

SCL

```
POKE(AREA := "Tag_Area",
     DBNUMBER := "Tag_DBNumber",
     BYTEOFFSET := "Tag_Byte",
     VALUE := "Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
VALUE	Tag_Value	16#11

The instruction overwrites the memory address "20" in the data block "5" with value "16#11".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

POKE_BOOL: Write memory bit

Description

The "Write memory bit" instruction is used to write a memory bit to a standard memory area without specifying a data type.

Use of memory area 16#84: DB

If you use a data block as memory area and this is not yet known during the creation of the program code you have the option of using the EN/ENO mechanism to recognize possible access errors:

SCL

```
POKE(AREA := 16#84,
     DBNUMBER := 1,
     BYTEOFFSET := 2,
     BITOFFSET := 4,
     VALUE := TRUE,
     ENO => ENO);
```

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
AREA	Input	BYTE	I, Q, M, D	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#2: Peripheral output (S7-1500 only)
DBNUM-BER	Input	DINT, DB_ANY	D	Number of the data block if AREA = DB, otherwise "0"
BYTEOFF-SET	Input	DINT	I, Q, M, D	Address to be written Only the 16 least significant bits are used.
BITOFFSET	Input	INT	I, Q, M, D	Bit to be written
VALUE	Input	BOOL	I, Q, M, D	Value to be written

Note

If you write the memory bit to the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

SCL

```
POKE_BOOL(AREA := "Tag_Area",
          DBNUMBER := "Tag_DBNumber",
          BYTEOFFSET := "Tag_Byte",
          BITOFFSET := "Tag_Bit",
          VALUE := "Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
VALUE	Tag_Value	M0.0

The instruction overwrites the memory bit "3" in data block "5" in byte "20" with the value "M0.0".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

POKE_BLK: Write memory area

Description

The "Write memory area" instruction is used to write a memory area to a different standard memory area without specifying a data type.

Use of memory area 16#84: DB

If you use a data block as memory area and this is not yet known during the creation of the program code you have the option of using the EN/ENO mechanism to recognize possible access errors:

SCL

```
POKE_BLK(AREA_SRC := 16#84,
         DBNUMBER_SRC := 1,
         BYTEOFFSET_SRC := 2,
         AREA_DEST := 16#84,
         DBNUMBER_DEST := 11,
         BYTEOFFSET_DEST := 22,
         COUNT := 3,
         ENO => ENO);
```

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
AREA_SRC	Input	BYTE	I, Q, M, D	The following areas can be selected in the source memory area: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB
DBNUMBER_SRC	Input	DINT, DB_ANY [1)]	D	Number of the data block in the source memory area, if AREA = DB, otherwise "0"
BYTEOFF-SET_SRC	Input	DINT	I, Q, M, D	Address in the source memory area to be written Only the 16 least significant bits are used.

Parameters	Declaration	Data type	Memory area	Description
AREA_DEST	Input	BYTE	I, Q, M, D	The following areas can be selected in the destination memory area: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB
DBNUMBER_DEST	Input	DINT, DB_ANY [1]	D	Number of the data block in the destination memory area, if AREA = DB, otherwise "0"
BYTEOFFSET_DEST	Input	DINT	I, Q, M, D	Address in the destination memory area to be written Only the 16 least significant bits are used.
COUNT	Input	DINT	I, Q, M, D	Number of bytes which are copied
[1] The data types of the parameters DBNUMBER_SRC and DBNUMBER_DEST must have the same data type. In other words, both tags must be either data type DINT or data type DB_ANY.				

Note

If you write the memory address to the input, output or bit memory areas, you must assign the DBNUMBER parameter the value "0", as the instruction is invalid otherwise.

Example

The following example shows how the instruction works:

SCL

```
POKE_BLK(AREA_SRC := "Tag_Source_Area",
         DBNUMBER_SRC := "Tag_Source_DBNumber",
         BYTEOFFSET_SRC := "Tag_Source_Byte");
         AREA_DEST := "Tag_Destination_Area",
         DBNUMBER_DEST := "Tag_Destination_DBNumber",
         BYTEOFFSET_DEST := "Tag_Destination_Byte",
         COUNT := "Tag_Count");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
AREA_SRC	Tag_Source_Area	16#84
DBNUMBER_SRC	Tag_Source_DBNumber	5
BYTEOFFSET_SRC	Tag_Source_Byte	20
AREA_DEST	Tag_Destination_Area	16#83
DBNUMBER_DEST	Tag_Destination_DBNumber	0
BYTEOFFSET_DEST	Tag_Destination_Byte	30
COUNT	Tag_Count	100

The instruction writes 100 byte from data block "5" starting with address "20" in the memory area of the bit memory starting at address "30".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

READ_LITTLE: Read data in little endian format

Description

The instruction "Read data in little endian format" is used to read data from a memory area and to write this to a single tag in the little endian byte sequence. With the little endian format, the byte with the least significant bits is saved first, which means at the lowest memory address.

The parameters SRC_ARRAY and DEST_VARIABLE are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at parameter DEST_VARIABLE has to be an elementary data type. The VARIANT at the SRC_ARRAY parameter points to the memory area to be read from, and this must be an ARRAY of BYTE.

You can also use an actual parameter with variable ARRAY index for the parameters SRC_ARRAY and DEST_VARIABLE.

The operand at the POS parameter determines the position in the memory area at which reading starts.

Note

Reading tag of data type VARIANT or BOOL

If you want to read a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to read a tag of the data type BOOL, use a "Slice access".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200 V4.0 and higher	S7-1500		
SRC_ARRAY	Input	ARRAY of BYTE	ARRAY of BYTE	I, Q, M, D, L	Memory area to be read from
DEST_VARIABLE	Output	Bit strings, integers, floating-point numbers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L	Read value

Parameters	Declaration	Data type		Memory area	Description
		S7-1200 V4.0 and higher	S7-1500		
POS	InOut	DINT	DINT	I, Q, M, D, L	Determines the position at which the reading starts. The POS parameter is calculated zero-based.
Function value (RET_VAL)		INT	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

SCL

```
#TagResult := READ_LITTLE(SRC_ARRAY := #SourceField,
                          DEST_VARIABLE => #DINTVariable,
                          POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
SRC_ARRAY	#SourceField	ARRAY[0..3] of BYTE := 16#1A, 16#2B, 16#3C, 16#4D
DEST_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
POS	#TagPos	0 => 4

The instruction reads the integer 1_295_788_826 from the #SourceField memory area and writes it to the #DINTVariable operand in little endian format. The data type at the DEST_VARIABLE parameter specifies how many bytes are read. The quantity 4 is stored in the #TagPos operand.

See also

Overview of the valid data types (Page 231)

Deserialize: Deserialize (Page 1730)

Serialize: Serialize (Page 1733)

Addressing areas of a tag with slice access (Page 110)

Basics of SCL (Page 7293)

WRITE_LITTLE: Write data in little endian format

Description

You use the "Write data in little endian format" instruction to write the data of a single tag in the little endian byte sequence to a memory area. With the little endian format, the byte with the least significant bits is saved first, which means at the lowest memory address.

The parameters SRC_VARIABLE and DEST_ARRAY are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at the SRC_VARIABLE parameter must point to an elementary data type. The VARIANT at the DEST_ARRAY parameter points to the memory area to which the data is written, and this must be an ARRAY of BYTE.

You can also use an actual parameter with variable ARRAY index for the parameters SRC_ARRAY and DEST_VARIABLE.

The operand at the POS parameter determines the position in the memory area at which writing starts.

Note

Writing tag of data type VARIANT or BOOL

If you want to write a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to write a tag of the data type BOOL, use a "Slice access".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200 V4.0 and higher	S7-1500		
SRC_VARIABLE	Input	Bit strings, integers, floating-point numbers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L	Tag whose data are written
DEST_ARRAY	InOut	ARRAY of BYTE	ARRAY of BYTE	I, Q, M, D, L	Memory area to which the data are written
POS	InOut	DINT	DINT	I, Q, M, D, L	Determines the position at which the writing starts. The POS parameter is calculated zero-based.
Function value (RET_VAL)		INT	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

SCL

```
#TagResult := WRITE_LITTLE(SRC_VARIABLE := #DINTVariable,
                           DEST_ARRAY := #TargetField,
                           POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
SRC_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
DEST_ARRAY	#TargetField	ARRAY[0..10] of BYTE = 16#1A, 16#2B, 16#3C, 16#4D
POS	#TagPos	0 => 4

The instruction writes the integer 1_295_788_826 from the #DINTVariable operand to the #TargetField memory area in little endian format. The data type at the SRC_VARIABLE parameter specifies how many bytes are written. The quantity 4 is stored in the #TagPos operand.

See also

Overview of the valid data types (Page 231)

Deserialize: Deserialize (Page 1730)

Serialize: Serialize (Page 1733)

Addressing areas of a tag with slice access (Page 110)

Basics of SCL (Page 7293)

READ_BIG: Read data in big endian format

Description

The instruction "Read data in little endian format" is used to read data from a memory area and to write this to a single tag in the big endian byte sequence. With the big endian format, the byte with the most significant bits is saved first, which means at the lowest memory address.

The parameters SRC_ARRAY and DEST_VARIABLE are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at parameter DEST_VARIABLE has to be an elementary data type. The VARIANT at the SRC_ARRAY parameter points to the memory area to be read from, and this must be an ARRAY of BYTE.

You can also use an actual parameter with variable ARRAY index for the parameters SRC_ARRAY and DEST_VARIABLE.

The operand at the POS parameter determines the position in the memory area at which reading starts.

Note

Reading tag of data type VARIANT or BOOL

If you want to read a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to read a tag of the data type BOOL, use a "Slice access".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200 V4.0 and higher	S7-1500		
SRC_ARRAY	Input	ARRAY of BYTE	ARRAY of BYTE	I, Q, M, D, L	Memory area to be read from
DEST_VARIABLE	Output	Bit strings, integers, floating-point numbers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L	Read value
POS	InOut	DINT	DINT	I, Q, M, D, L	Determines the position at which the reading starts. The POS parameter is calculated zero-based.
Function value (RET_VAL)		INT	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

SCL

```
#TagResult := READ_BIG(SRC_ARRAY := #SourceField,
                       DEST_VARIABLE => #DINTVariable,
                       POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
SRC_ARRAY	#SourceField	ARRAY[0..10] of BYTE := 16#1A, 16#2B, 16#3C, 16#4D
DEST_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
POS	#TagPos	0 => 4

The instruction reads the integer 439_041_101 from the #SourceField memory area and writes it to the #DINTVariable operand in big endian format. The data type at the DEST_VARIABLE parameter specifies how many bytes are read. The quantity 4 is stored in the #TagPos operand.

See also

Overview of the valid data types (Page 231)

Deserialize: Deserialize (Page 1730)

Serialize: Serialize (Page 1733)

Addressing areas of a tag with slice access (Page 110)

Basics of SCL (Page 7293)

WRITE_BIG: Write data in big endian format

Description

You use the "Write data in big endian format" instruction to write the data of a single tag in the big endian byte sequence to a memory area. With the big endian format, the byte with the most significant bits is saved first, which means at the lowest memory address.

The parameters SRC_VARIABLE and DEST_ARRAY are of data type VARIANT. However, there are a few restrictions regarding the data type with which the parameters can be interconnected. The VARIANT at the SRC_VARIABLE parameter must point to an elementary data type. The VARIANT at the DEST_ARRAY parameter points to the memory area to which the data is written, and this must be an ARRAY of BYTE.

You can also use an actual parameter with variable ARRAY index for the parameters SRC_ARRAY and DEST_VARIABLE.

The operand at the POS parameter determines the position in the memory area at which writing starts.

Note

Writing tag of data type VARIANT or BOOL

If you want to write a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to write a tag of the data type BOOL, use a "Slice access".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200 V4.0 and higher	S7-1500		
SRC_VARIABLE	Input	Bit strings, integers, floating-point numbers, TOD, DATE, CHAR, WCHAR	Bit strings, integers, floating-point numbers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L	Tag whose data are written
DEST_ARRAY	InOut	ARRAY of BYTE	ARRAY of BYTE	I, Q, M, D, L	Memory area to which the data are written
POS	InOut	DINT	DINT	I, Q, M, D, L	Determines the position at which the writing starts. The POS parameter is calculated zero-based.
Function value (RET_VAL)		INT	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data type at the SRC_ARRAY parameter is not ARRAY of BYTE.
8382	The value at the POS parameter is outside the limits of the ARRAY.
8383	The value at the POS parameter is within the limits of the ARRAY but the size of the memory area exceeds the high limit of the ARRAY.

Example

The following example shows how the instruction works:

SCL

```
#TagResult := WRITE_BIG(SRC_VARIABLE := #DINTVariable,
                        DEST_ARRAY := #TargetField,
                        POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
SRC_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
DEST_ARRAY	#TargetField	ARRAY[0..10] of BYTE = 16#1A, 16#2B, 16#3C, 16#4D
POS	#TagPos	0 => 4

The instruction writes the integer 439_041_101 from the #DINTVariable operand to the #TargetField memory area in big endian format. The data type at the SRC_VARIABLE parameter specifies how many bytes are written. The quantity 4 is stored in the #TagPos operand.

See also

Overview of the valid data types (Page 231)

Deserialize: Deserialize (Page 1730)

Serialize: Serialize (Page 1733)

Addressing areas of a tag with slice access (Page 110)

Basics of SCL (Page 7293)

VARIANT

VariantGet: Read out VARIANT tag value

Description

You can use the "Read out VARIANT tag value" instruction to read the value of the tag to which the VARIANT at the SRC parameter points and write it in the tag at the DST parameter.

The SRC parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the DST parameter.

The data type of the tag specified at the DST parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT instruction: Move block" instruction. For additional information, refer to "See also".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
SRC	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		Tag to be read
DST	Output	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
VariantGet(SRC := #TagIn_Source,
           DST => "TagOut_Dest");
```

The value of the tag to which the VARIANT at the "#TagIn_Source" operand points, is read and written to the "TagOut_Dest" operand.

See also

Overview of the valid data types (Page 231)

MOVE_BLK_VARIANT: Move block (Page 1739)

Basics of SCL (Page 7293)

VariantPut: Write VARIANT tag value

Description

You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter to the memory at the DST parameter to which the VARIANT points.

The DST parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT instruction: Move block" instruction. For additional information, refer to "See also".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
SRC	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	I, Q, M, D, L	I, Q, M, D, L, P	Tag to be read
DST	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)		Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
VariantPut(SRC := "TagIn_Source",
           DST := #TagIn_Dest);
```

The value of the "TagIn_Source" operand is written to the tag to which the VARIANT at the #TagIn_Dest operand points.

See also

Overview of the valid data types (Page 231)

MOVE_BLK_VARIANT: Move block (Page 1739)

Basics of SCL (Page 7293)

CountOfElements: Get number of ARRAY elements

Description

You use the "Get number of ARRAY elements" instruction to query how many ARRAY elements a tag to which VARIANT points has.

If it is a single-dimensional ARRAY, the number of ARRAY elements is output as the result. (The difference between the high and low limit +1). If it is a multi-dimensional ARRAY, the number of all dimensions is output as the result.

If you want to query the elements of an ARRAY DB, you should use the instructions "ReadFromArrayDB" or "WriteFromArrayDB", because here a more precise error evaluation for the number of elements is possible.

Note

Instances

The VARIANT pointer can point to no instance and therefore also to no multi-instance or an ARRAY of multi-instances.

Note

ARRAY within a data block

If you want to query the number of the elements of an ARRAY which is located in a data block, the block attribute "Data block write-protected in the device" must not be activated in this block. Otherwise, the RET_VAL parameter returns the result "0", irrespective of how many elements the ARRAY contains.

The result is also "0" if the VARIANT tag is not an ARRAY.

If the VARIANT points to an ARRAY of BOOL, the fill elements are included in the count. (For example, 8 is returned for an ARRAY[0..1] of BOOL).

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Tag to be queried
Function value		UDINT	I, Q, M, D, L	Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

SCL

```
IF IS_ARRAY(#Tag_VARIANTToArray) THEN
"Tag_Result" := CountOfElements(#Tag_VARIANTToArray);
END_IF;
```

If the tag to which the VARIANT points is an ARRAY, the number of ARRAY elements is output.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ARRAY[*]

LOWER_BOUND: Read out low ARRAY limit

Description

You can declare tags of the data type ARRAY[*] in the block interface of a function block or function. For these local tags, you can read out the limits of the ARRAY. You will need to specify the required dimension at the DIM parameter.

The "Read out ARRAY low limit" instruction is used to read out the variable low limit of the ARRAY.

Note

Availability of the instruction

The instruction is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
ARR	Input	ARRAY[*]	FB => Section In-Out FC => Sections Input and InOut	ARRAY of which the variable low limit is to be read.
DIM	Input	UDINT	I, Q, M, D, L, P or constant	Dimension of the ARRAY of which the variable low limit is to be read.
Function value		DINT	I, Q, M, D, L, P	Result

Example

The following example shows how the instruction works:

SCL

```
"Result" := LOWER_BOUND(ARR := #ARRAY_A,
                        DIM := 2);
```

The instruction reads out the variable low limit of the ARRAY #ARRAY_A from the second dimension. If the instruction is executed without errors, the result is written to operand "Result".

See also

Overview of the valid data types (Page 231)

Basic information on ARRAY (Page 283)

Basics of SCL (Page 7293)

UPPER_BOUND: Read out high ARRAY limit

Description

You can declare tags of the data type ARRAY[*] in the block interface of a function block or function. For these local tags, you can read out the limits of the ARRAY. You will need to specify the required dimension at the DIM parameter.

The instruction "Read out ARRAY high limit" is used to read out the variable high limit of the ARRAY.

Note

Availability of the instruction

The instruction is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
ARR	Input	ARRAY[*]	FB => Section In-Out FC => Sections Input and InOut	ARRAY of which the variable high limit is to be read.
DIM	Input	UDINT	I, Q, M, D, L, P or constant	Dimension of the ARRAY of which the variable high limit is to be read.
Function value		DINT	I, Q, M, D, L, P	Result

Example

The following example shows how the instruction works:

SCL

```
"Result" := UPPER_BOUND(ARR := #ARRAY_A,
                        DIM := 2);
```

The instruction reads out the variable high limit of the ARRAY #ARRAY_A from the second dimension. If the instruction is executed without errors, the result is written to operand "Result".

See also

Overview of the valid data types (Page 231)

Basic information on ARRAY (Page 283)

Basics of SCL (Page 7293)

Legacy (S7-1500)

BLKMOV: Move block (S7-1500)

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can also use this tag in memory areas "with optimized block access".

The following figure shows the principle of the move operation:

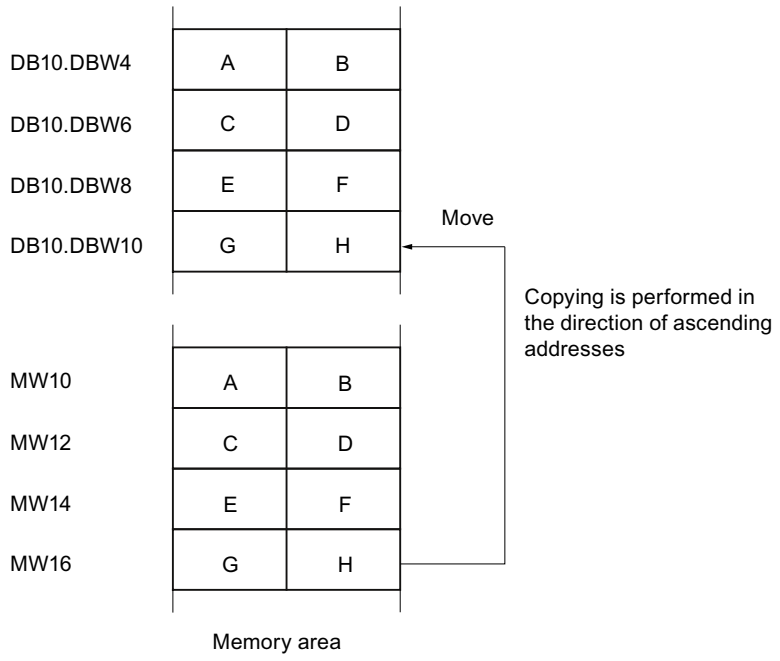

Consistency of source and destination data

Note that while the instruction "Move block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
DSTBLK	Output 1)	VARIANT	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
Function value (RET_VAL)		INT	I, Q, M, D, L, P	Error information
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_RetVal" := BLKMOV(SRCBLK := P#M100.0 BYTE 10,
                       DSTBLK => P#DB1.DBX0.0 BYTE 10);
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1857)

Basics of SCL (Page 7293)

UBLKMOV: Move block uninterruptible (S7-1500)

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can also use this tag in memory areas "with optimized block access".

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as the area length.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
DSTBLK	Output [1)]	VARIANT	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
Function value (RET_VAL)		INT	I, Q, M, D, L, P	Error information
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
8091	The source or target area is only available in the load memory.

Error code (W#16#...)	Explanation
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_RetVal" := UBLKMOV(SRCBLK := P#M100.0 BYTE 10,
                        DSTBLK => P#DB1.DBX0.0 BYTE 10);
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1857)

Basics of SCL (Page 7293)

FILL: Fill block (S7-1500)

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

You use VARIANT to define the source and destination areas.

Note

You can also define the source and destination areas using the ANY data type

If you use the ANY data type, you must observe the following in connection with the STRING data type:

- In the case of an assignment of STRING (source area) using ANY after STRING (destination area), the content of the STRING is copied to the destination area repeatedly until it is full.
  Source area: 'STEP7-SCL-TIA-Portal'
  Destination area: 'STEP7-SCL-TIA-PortalSTEP7-SCL-TIA-PortalSTEP7-SCL'
- In the case of an assignment of WSTRING (source area) using ANY after WSTRING (destination area), the entire WSTRING including type information is copied only once to the destination area.
  Source area: WSTRING#'STEP7-SCL-TIA-Portal'
  Destination area: WSTRING#'STEP7-SCL-TIA-Portal'

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can also use this tag in memory areas "with optimized block access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK instruction: Fill block" instruction.

The following figure shows the principle of the move operation:

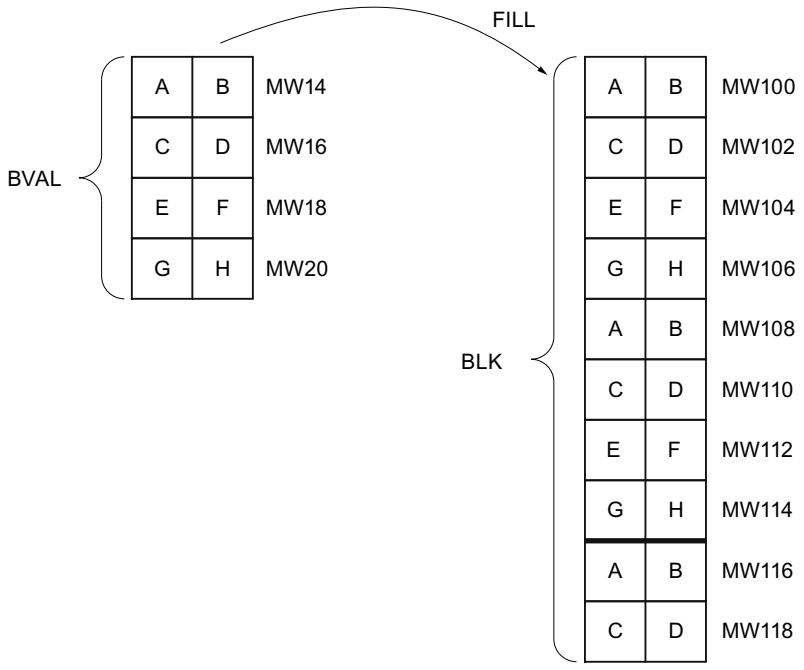

Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
BVAL	Input	VARIANT	I, Q, M, D, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
BLK	Output [1)]	VARIANT	I, Q, M, D, L, P	Specification of the memory area that will be filled with the content of the source area.
Function value (RET_VAL)		INT	I, Q, M, D, L, P	Error information
1) The BLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

BVAL parameter

When you transfer a structure as an input parameter, remember that the length of a structure is always based on an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8152	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the BVAL parameter.
8352	The WSTRING, WCHAR, BOOL, ARRAY of STRING, ARRAY of WSTRING and ARRAY of WCHAR data types are not supported at the BLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_RetVal" := FILL(BVAL := P#M14.0 WORD 4,
                     BLK => P#M100.0 WORD 10);
```

The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1857)

Basics of SCL (Page 7293)

Conversion operations

CONVERT: Convert value

Description

Use the "Convert value" instruction to program explicit conversions. When the instruction is inserted, the "CONVERT" dialog opens. You specify the source data type and the destination data type of the conversion in this dialog. The source value is read and converted to the specified destination data type.

Conversion options for bit strings

BYTE and WORD bit strings cannot be selected in the instruction box. However, it is possible to specify an operand of data type DWORD or LWORD at a parameter of the instruction if the lengths of the input and output operands match. The operand is then interpreted by the data type of a bit string according to the data type of the input or output parameter and is implicitly converted. The data type DWORD is for example interpreted as DINT/UDINT and LWORD as LINT/ULINT. These conversion options are also available to you when "IEC check" is enabled.

Note

For CPUs of the S7-1500 series: The data types DWORD and LWORD can only be converted to or from data type REAL or LREAL.

During the conversion, the bit pattern of the source value is transferred unchanged, right-justified, to the target data type. If no errors occur during the conversion, the signal state of enable output ENO = 1; if an error occurs during processing, the signal state of the enable output ENO = 0.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Source type>	Input	Binary numbers, integers, floating-point numbers, times, date and time, character strings, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted.
<Destination type>	Output	Binary numbers, integers, floating-point numbers, times, date and time, character strings, BCD16, BCD32	I, Q, M, D, L, P or constant	Result of the conversion

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_INT" := REAL_TO_INT("Tag_REAL");
```

The following table shows how the instruction works using specific operand values:

Operand	Data type	Value
Tag_REAL	REAL	20.56
Tag_INT	INT	21

During the conversion, the value of the "Tag_REAL" operand will be rounded to the nearest integer and saved in the "Tag_INT" operand.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ROUND: Round numerical value

Description

The "Round numerical value" instruction is used to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts it into an integer or floating-point number. If the input value is exactly between an even and odd number, the even number is selected.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value to be rounded.
Function value		Integers, floating-point numbers	I, Q, M, D, L, P	Result of the rounding

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ROUND("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	1.50000000	-1.50000000
Tag_Result	2	-2

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

CEIL: Generate next higher integer from floating-point number

Description

Use the "Generate next higher integer from floating-point number" instruction to round the value to the nearest integer. The instruction interprets the input value as floating-point number and converts it to the next higher integer. The function value can be greater than or equal to the input value.

Use the following syntax to change the data type of the instruction:

CEIL_<data type>();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
_<Data type>		Integers, floating-point numbers Default: DINT	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Integers, floating-point numbers	I, Q, M, D, L	Input value rounded up

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := CEIL("Tag_Value");
"Tag_Result2" := CEIL_REAL("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.5	-0.5
Tag_Result1	1	0
Tag_Result2	1.0	0.0

The result of the instruction is returned in the operand "Tag_Resultxy" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

FLOOR: Generate next lower integer from floating-point number

Description

Use the "Generate next lower integer from floating-point number" instruction to round the value of a floating point number to the next lower integer. The instruction interprets the input value as floating-point number and converts it to the next lower integer. The function value can be equal or less than the input value.

Use the following syntax to change the data type of the instruction:

FLOOR_<data type>();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
_<Data type>		Integers, floating-point numbers Default: DINT	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Integers, floating-point numbers	I, Q, M, D, L	Input value rounded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := FLOOR("Tag_Value");
"Tag_Result2" := FLOOR_REAL("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.5	-0.5
Tag_Result1	0	-1
Tag_Result2	0.0	-1.0

The result of the instruction is returned in the operand "Tag_Resultxy" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

TRUNC: Truncate numerical value

Description

The "Truncate numerical value" instruction is used to generate an integer from the input value without rounding. The instruction selects only the integer part of the input value and returns this part without decimal places as the function value.

Use the following syntax to change the data type of the instruction:

TRUNC_<data type>();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expression>	Input	Floating-point numbers	I, Q, M, D, L	Input value
_<Data type>		Integers, floating-point numbers Default: DINT	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Integers, floating-point numbers	I, Q, M, D, L	Integer component of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := TRUNC("Tag_Value1");
"Tag_Result2" := TRUNC("Tag_Value2"+"Tag_Value3");
"Tag_Result3" := TRUNC_SINT("Tag_Value4");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value1	-1.5
Tag_Result1	-1
Tag_Value2	2.1
Tag_Value3	3.2
Tag_Result2	5
Tag_Result3	2
Tag_Value4	2.4

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SCALE_X: Scale

Description

Use the "Scale" instruction to scale a floating-point number by mapping it to a specific value range. You specify the value range with the MIN and MAX parameters. The result of the scaling is an integer.

Use the following syntax to change the data type of the instruction:

SCALE_X_<data type>();

The following figure shows an example of how values can be scaled:

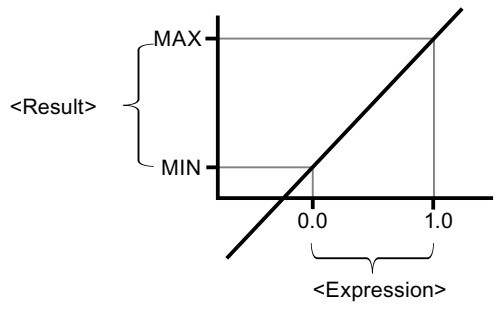

The "Scale" instruction works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

Note

For more information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L	High limit of the value range
_<Data type>		Integers, floating-point numbers Default: INT	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Integers, floating-point numbers	-	Result of scaling

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := SCALE_X(MIN := "Tag_Value1",
                         VALUE := "Tag_Real",
                         MAX := "Tag_Value2");
"Tag_Result2" := SCALE_X_REAL(MIN := "Tag_Value1",
                              VALUE := "Tag_Real",
                              MAX := "Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Real	0.5
Tag_Value1	10
Tag_Value2	30
Tag_Result1	20
Tag_Result2	20.0

See also

Overview of the valid data types (Page 231)

Declaring global constants (Page 7089)

Basics of SCL (Page 7293)

NORM_X: Normalize

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. The result at the OUT output is calculated and stored as a floating-point number depending on the location of the value to be normalized within this value range. If the value to be normalized equals the value at input MIN, the instruction returns the result "0.0". If the value to be normalized equals the value at input MAX, the instruction returns the result "1.0".

Use the following syntax to change the data type of the instruction:

NORM_X_<data type>();

The following figure shows an example of how values can be normalized:

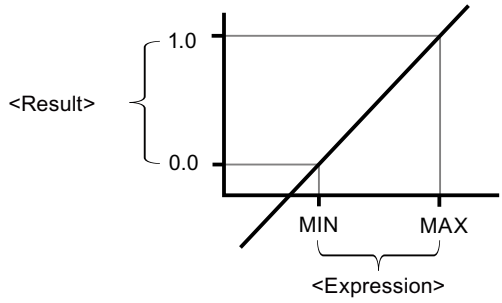

The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

Note

For more information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
MIN [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L	Low limit of the value range
VALUE [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L	Value to be normalized.
MAX [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L	High limit of the value range

Parameters	Declaration	Data type	Memory area	Description
_<Data type>		Floating-point numbers Default: REAL	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Floating-point numbers	I, Q, M, D, L	Result of the normalization
[1] If you use constants in these three parameters, you only need to declare one of them.				

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := NORM_X(MIN := "Tag_Value1",
                        VALUE := "Tag_InputValue",
                        MAX := "Tag_Value2");
"Tag_Result2" := NORM_X_LREAL(MIN := "Tag_Value1",
                              VALUE := "Tag_InputValue",
                              MAX := "Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InputValue	20
Tag_Value1	10
Tag_Value2	30
Tag_Result1	0.5
Tag_Result2	0.5

See also

Overview of the valid data types (Page 231)

Declaring global constants (Page 7089)

Basics of SCL (Page 7293)

REF: Create a reference to a tag (S7-1500)

Description

The keyword "REF()" is used to specify the tag to which a previously declared reference will point. As parameter, specify the tag to be referenced.

Note

Declaration of references

Before you use the keyword "REF()", you must first declare a reference in the block interface.

See also:

Declaring references (Page 303)

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Expression>	Input	Bit sequences, except BOOL, Integers, Floating-point numbers, Character strings, PLC data types (UDT), System data types (SDT), ARRAYs of the named data types	DB tag (optimized), Block interface of an FB (optimized)	Tag to which the reference will point
Function value		The data type of the function value corresponds to that of the declared reference tag.	Block interface of an FC: Input, Output, Temp Block interface of an FB: Temp	Address of the referenced tag

You will find additional information on valid data types under "See also".

Rules

The following rules apply to referencing:

- The data type of the tag that you specify in brackets must correspond exactly to the data type of the declared reference. This means that a reference with data type "REF_TO Int" can only point to a tag of data type "Int". No data type conversion is made.
- The tag that you specify as the parameter for "REF()" must be located in an optimized memory area.
- A reference must not refer to the following data:
  - Temporary data (TEMP)
  - Global tags from the PLC tag table
  - Block parameters
  - Constants
  - Write-protected tags

The following applies to references to arrays:

- The array limits and dimensions of the reference and the referenced tag must be identical.
- Array[*] is not supported.
- References to ARRAY DBs that are based on a PLC data type must be created as follows:
  `REF("my_ArrayDB_UDT".THIS)`
  `REF("my_ArrayDB_UDT"."THIS"[i])`

Example

In the interface of the block the temporary tags "#myRefInt", "#myRefType" and "#myRefArray" were declared.

Name	Data type
Input	
Output	
InOut	
Static	
a	Int
Temp	
myRefInt	REF_TO Int
myRefType	REF_TO "myUDT"
myRefARRAY	REF_TO Array[0..3] of Int
Constant	

In the program code these references are assigned concrete tags:

SCL

```
#myRefInt := REF(#a);
#myRefType := REF("myDB".myUDT);
#myRefType := REF("myArrayDB_UDT"."THIS"[1]);
#myRefARRAY := REF("myDB".myArray);
```

See also

Basic information about references (Page 300)

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

VARIANT

VARIANT_TO_DB_ANY: Convert VARIANT to DB_ANY

Description

You use the "Convert VARIANT to DB_ANY" instruction to query the data block number that the operand that is specified at the IN parameter addresses. This can be an instance data block or an ARRAY data block. The operand at the IN parameter has the data type VARIANT, which means you do not need to know the data type of the data block whose number is to be queried when the program is created. The data block number is read during runtime and written to the operand specified at the RET_VAL parameter.

Requirements

If the requirements are met, the instruction is executed. If the requirements are not met, "0" is output as data block number.

The output tag...	homed...	Conversion options
VARIANT	... a data block which is an instance data block of a PLC data type or a system data type (SDT).	The output tag can be converted to a data block number.
VARIANT	... a data block which is an ARRAY DB.	The output tag can be converted to a data block number.
VARIANT	... a tag with an elementary data type.	It is not possible to convert the output tag to a database number because a data block can never comprise only one elementary data type.
VARIANT	... a structure within a data block.	It is not possible to convert the output tags to a database number because it is only a part within the data block.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Tag to be read. (The function value of the "DB_ANY_TO_VARIANT" instruction). You can use a local or global tag at the IN parameter.
ERR	Output	INT	I, Q, M, D, L	Error information
Function value (RET_VAL)		DB_ANY	I, Q, M, D, L	Result: Number of the DB

For additional information on valid data types, refer to "See also".

ERR parameter

The following table shows the meaning of the values of the ERR parameter:

Error code* (W#16#...)	Explanation
0000	No error
252C	The VARIANT data type at the IN parameter has the value "0" and the CPU changes to STOP mode.
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8131	The data block does not exist, is too short, or is located in load memory.
8132	The data block is too short and not an ARRAY data block.
8150	The data type VARIANT at parameter IN provides the value "0". To receive this error message, the "Handle errors within block" block property must be activated. Otherwise the CPU changes to STOP mode and sends the error code 16#252C.
8153	The VARIANT data type at the IN parameter does not point to the start of an ARRAY data block or the length of the VARIANT does not match that of the data block.
8154	The data block has the incorrect data type.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"OutputDBNumber" := VARIANT_TO_DB_ANY(IN := #tempVARIANT,
                                      ERR := "Tag_Error");
```

The following table shows how the instruction works using specific operand values:

Parameters	Declaration in the block interface	Operand	Value
IN	Input	tempVARIANT	-
<function value>	Output	OutputDBNumber	11

The number of a data block that is specified at the tempVARIANT operand is read. Because the operand has the data type VARIANT, you do not need to know the data type of the tag during program creation. The number is written to the "OutputDBNumber" tag which has the data type DB_ANY.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

DB_ANY_TO_VARIANT: Convert DB_ANY to VARIANT

Description

The "Convert DB_ANY to VARIANT" instruction is used to generate a VARIANT tag from a data block that meets the requirements listed below. The operand at the IN parameter has the data type DB_ANY, which means that the data block does not have to be known when the program is created. The data block number is read during runtime.

Requirements

If the requirements are met, the instruction is executed. If the requirements are not met or the data block does not exist, the value NULL is output at the RET_VAL parameter. All other accesses with the RET_VAL tag fail.

The input tag of data type ...	homed...	Conversion options
DB_ANY	...a data block which is an instance data block of a PLC data type or a system data type (SDT).	Conversion is possible
DB_ANY	...a data block which is an ARRAY DB.	Conversion is possible
DB_ANY	...a data block which is an instance data block of a function block or a global data block.	Conversion is not possible

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DB_ANY	I, Q, M, D, L	Data block whose number is read. You can use a local or global tag at the IN parameter.
ERR	Output	INT	I, Q, M, D, L	Error information
Function value (RET_VAL) [1]		VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Number of the data block
1) The RET_VAL parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

ERR parameter

The following table shows the meaning of the values of the ERR parameter:

Error code* (W#16#...)	Explanation
0000	No error
8130	The number of the data block is "0"
8131	The data block does not exist, is too short, or is located in load memory.
8132	The data block is too short and not an ARRAY data block.
8134	The data block is write protected.
8154	The data block has the incorrect data type.
8155	The data block has an unknown data type. [1]
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

[1] The reason for the error code #8155 to be output is:

A PLC data type (UDT1) was declared and a data block (DB2) of the data type "UDT1" was created. The tag table contains a tag (3) of the data type DB_ANY. Next the instruction "DB_ANY_TO_VARIANT" was called in a program block (4) and supplied with the tag (3) at the IN parameter. During execution the instruction "DB_ANY_TO_VARIANT" returns the error code 16#8155.

Follow these steps to resolve the error code:

1. Create a function (FC5) and declare a tag of the data type VARIANT at the InOut interface.
2. Create an additional function (FC6) and call FC5 in it.
3. Create a tag (7) of the data type "UDT1" in FC6 in the Temp interface.

4. Supply the InOut interface of the FC5 with tag (7).
5. Compile and download the two blocks (FC5 and FC6) to your CPU. You do not need to call these blocks (FC5 and FC6) in the user program.

Result:

The error code 16#8155 is no longer output because the user program is now familiar with the data type.

This procedure is not necessary if you call one of the two instructions "VariantGet" or "VariantPut" after calling the instruction "DB_ANY_TO_VARIANT".

Example

The following example shows how the instruction works:

SCL

```
#tempVARIANT := DB_ANY_TO_VARIANT(IN := "InputDB",
                                  ERR := "Tag_Error");
```

The following table shows how the instruction works using specific operand values:

Parameters	Declaration in the block interface	Operand	Value
IN	Input	InputDB	11
<function value>	Temp	tempVARIANT	-

The number of any data block that is specified at the "InputDB" operand is used to generate a tag of data type VARIANT that addresses the data block. Because the operand at the IN parameter has the DB_ANY data type, the data block that will be used during runtime does not have to be known when the program is created (neither the name nor the number of the data block). Because the operand at the RET_VAL parameter has the data type VARIANT, you do not need to know the data type of the data block when the program is created.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Legacy (S7-1500)

SCALE: Scale (S7-1500)

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number that can be scaled in physical units between a low limit value and a high limit value. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the "K1" and "K2" constants are determined by the signal state on the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case, the constant "K1" has the value -27648.0 and the constant "K2" the value +27648.0.
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case, the constant "K1" has the value 0.0 and the constant "K2" the value +27648.0.

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled inversely proportionate to the input value.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
Function value (RET_VAL)		WORD	I, Q, M, D, L, P	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the parameter IN is greater than 27 648 or is less than 0 (unipolar) or -27 648 (bipolar).
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_ErrorCode" := SCALE(IN := "Tag_InputValue",
                         HI_LIM := "Tag_HighLimit",
                         LO_LIM := "Tag_LowLimit",
                         BIPOLAR := "Tag_Bipolar",
                         OUT => "Tag_Result");
```

The error information is returned in the operand "Tag_ErrorCode" as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_Result	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1857)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

UNSCALE: Unscale (S7-1500)

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case, the constant "K1" has the value -27648.0 and the constant "K2" the value +27648.0.
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case, the constant "K1" has the value 0.0 and the constant "K2" the value +27648.0.

When the value at parameter IN is not within the limits defined by HI_LIM and LO_LIM, an error is output and the result is set to the nearest limit.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
Function value (RET_VAL)		WORD	I, Q, M, D, L, P	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

```
"Tag_ErrorCode" := UNSCALE(IN := "Tag_InputValue",
                           HI_LIM := "Tag_HighLimit",
                           LO_LIM := "Tag_LowLimit",
                           BIPOLAR := "Tag_Bipolar",
                           OUT => "Tag_Result");
```

The error information is returned in the operand "Tag_ErrorCode" as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_Result	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

GET_ERR_ID: Get error ID locally (Page 1857)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

Program control operations

IF: Run conditionally

Description

The instruction "Run conditionally" branches the program flow depending on a condition. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Parameters

Depending on the type of branch, you can program the following forms of the instruction:

- Branch through IF:
  IF <condition> THEN <instructions>
  END_IF;

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the execution of the program continues with the next instruction after the END_IF.

- Branch through IF and ELSE:
  IF <condition> THEN <instructions1>
  ELSE <Instructions0>
  END_IF;

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the instructions programmed after the ELSE are executed. Then the execution of the program continues with the next instruction after the END_IF.

- Branch through IF, ELSIF and ELSE:
  IF <condition1> THEN <instructions1>
  ELSIF <condition2> THEN <instruction2>
  ELSE <Instructions0>
  END_IF;

If the first condition (<Condition1>) is satisfied, the instructions (<Instructions1>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If the first condition is not satisfied, the second condition (<Condition2>) is checked. If the second condition (<Condition2>) is fulfilled, the instructions (<Instructions2>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If none of the conditions are fulfilled, the instructions (<Instructions0>) after ELSE are executed followed by the execution of the program after END_IF.

You can nest as many combinations of ELSIF and THEN as you like within the IF instruction. The programming of an ELSE branch is optional.

The syntax of the IF instruction consists of the following parts:

Parameters	Data type	Memory area	Description
<Condition>	BOOL	I, Q, M, D, L	Expression to be evaluated
<Instructions>	-		Instructions to be executed with satisfied condition. An exception are instructions programmed after the ELSE. These are executed if no condition within the program loop is satisfied.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF "Tag_1" = 1
THEN "Tag_Value" := 10;
ELSIF "Tag_2" = 1
THEN "Tag_Value" := 20;
ELSIF "Tag_3" = 1
THEN "Tag_Value" := 30;
ELSE "Tag_Value" := 0;
END_IF;
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_1	1	0	0	0
Tag_2	0	1	0	0
Tag_3	0	0	1	0
Tag_Value	10	20	30	0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

CASE: Create multiway branch

Description

The instruction "Create multiway branch" executes one of several instruction sequences depending on the value of a numerical expression.

The value of the expression must be an integer. When the instruction is executed, the value of the expression is compared with the values of several constants. If the value of the

expression agrees with the value of a constant, the instructions programmed directly after this constant are executed. The constants can assume the following values:

- An integer (for example, 5)
- A range of integers (for example 15...20)
- An enumeration consisting of integers and ranges (for example, 10, 11, 15...20)

You can declare the instruction as follows:

CASE <Tag> OF

<Constant1>: <Instructions1>;

<Constant2>: <Instructions2>;

<ConstantX>: <InstructionsX>; // X >= 3

ELSE <Instructions0>;

END_CASE;

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
<Tag>	Integers	I, Q, M, D, L	Value which is compared to the programmed constant values.
<Constant>	Integers	-	Constant values which form the condition for the execution of an instruction sequence. The constants can assume the following values: • An integer (for example, 5) • A range of integers (for example 15...20) • An enumeration consisting of integers and ranges (for example, 10, 11, 15...20)
<Instruction>	-	-	Any instructions which are executed if the value of the expression agrees with the value of a constant. An exception are instructions programmed after the ELSE. These instructions are executed if the values do not agree.

For additional information on valid data types, refer to "See also".

If the value of the expression agrees with the value of the first constant (<Constant1>), the instructions (<Instructions1>) which are programmed directly after the first constant are executed. Program execution subsequently resumes after the END_CASE.

If the value of the expression does not agree with the value of the first constant (<Constant1>), this value is compared to the value of the constant which is programmed next. In this way, the CASE instruction is executed until the values agree. If the value of the expression does not correspond to any of the programmed constant values, the instructions (<Instructions0>) which are programmed after the ELSE are executed. ELSE is an optional part of the syntax and can be omitted.

The CASE instruction can also be nested by replacing an instruction block with CASE. END_CASE represents the end of the CASE instruction.

Example

The following example shows how the instruction works:

```
SCL
CASE "Tag_Value" OF
     0 :
          "Tag_1" := 1;
     1,3,5 :
          "Tag_2" := 1;
     6...10 :
          "Tag_3" := 1;
     16,17,20...25 :
          "Tag_4" := 1;
     ELSE
          "Tag_5" := 1;
END_CASE;
```

The following table shows how the instruction works using specific operand values:

Operand	Values				
Tag_Value	0	1, 3 , 5	6, 7, 8, 9, 10	16, 17, 20, 21, 22, 23, 24, 25	2
Tag_1	1	-	-	-	-
Tag_2	-	1	-	-	-
Tag_3	-	-	1	-	-
Tag_4	-	-	-	1	-
Tag_5	-	-	-	-	1
1: The operand is set to the signal state "1". -: The signal state of the operand remains unaltered.					

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

FOR: Run in counting loop

Description

The instruction "Run in counting loop" causes repeated execution of a program loop until a loop variable lies within a specified value range.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Note

Information on the number of runs and the run tag

The number of runs cannot be changed while the program is running.

For performance reasons, the run tag should be declared in the block interface in the "Temp" section. It is not possible to change the run tag from within the loop.

The following example program would therefore generate a syntax error and cannot be compiled:

```
FOR #i := 1 TO 10 DO
#i := #i + 1;
END_FOR;
```

Limits for FOR statements

To program "safe" FOR statements that do not run endlessly, observe the following rules and limits:

FOR <Run_tag> := <Start_value> TO <End_value> BY <Increment> DO <Instructions>;

END_FOR;

If ...	... then	Note
Start value < End value	End value < (PMAX increment)	Run tag runs in positive direction
Start value > End value AND increment < O	End value > (NMAX increment)	Run tag runs in negative direction

Limits

Different limits apply to the possible data types:

Data type	PMAX	NMAX
Run tag of type SINT	127	-128
Run tag of type INT	32767	-32768
Run tag of type DINT	2147483647	-2147483648
Run tag of type LINT	9223372036854775807	-9223372036854775808

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type		Memory area	Description
	S7-1200	S7-1500		
<Run tag>	SINT, INT, DINT	SINT, INT, DINT, LINT	I, Q, M, D, L	Operand whose value is evaluated with the loop execution. The data type of the loop variable determines the data type of the other parameters.
<Start value>	SINT, INT, DINT	SINT, INT, DINT, LINT	I, Q, M, D, L	Expression whose value is allocated at the start of the loop execution of the loop variables.
<End value>	SINT, INT, DINT	SINT, INT, DINT, LINT	I, Q, M, D, L	Expression whose value defines the last run of the program loop. The value of the loop variable is checked before each loop: • End value not reached: The instructions according to DO are executed • End value is reached: The FOR loop is executed one last time • End value exceeded: The FOR loop is completed An alteration to the end value is not permitted during execution of the instruction.
<Increment>	SINT, INT, DINT	SINT, INT, DINT, LINT	I, Q, M, D, L	Expression by whose value the loop variable is increased (positive increment) or decreased (negative increment) after each loop. Specification of the increment is optional. If no increment is given, the value of the loop variable is increased by 1 after each loop. An alteration of the increment is not permitted during execution of the instruction.
<Instructions>	-	-		Instructions which are carried out with each loop, as long as the value of the loop variable lies within the value range. The value range is defined by the start and end values.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
FOR i := 2 TO 8 BY 2
    DO "a_array[i] := "Tag_Value"*"b_array[i]";
END_FOR;
```

The operand "Tag_Value" is multiplied with the elements (2, 4, 6, 8) of the ARRAY tag "b_array". The result is read in to the elements (2, 4, 6, 8) of the ARRAY tag "a_array".

See also

CONTINUE: Recheck loop condition (Page 1841)

EXIT: Exit loop immediately (Page 1842)

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

WHILE: Run if condition is met

Description

The instruction "Run if condition is met" causes a program loop to be repeatedly executed until the implementation condition is satisfied. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

You can declare the instruction as follows:

WHILE <Condition> DO <Instructions>;

END_WHILE;

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
<Condition>	BOOL	I, Q, M, D, L	Expression which is evaluated before each loop.
<Instructions>	-		Instructions to be executed with satisfied condition. If the condition has not been satisfied, program execution continues after END_WHILE.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
WHILE
      "Tag_Value1" <> "Tag_Value2"
      DO "Tag_Result"
      := "Tag_Input";
END_WHILE;
```

As long as the values of the operands "Tag_Value1" and "Tag_Value2" do not match, the value of the operand "Tag_Input" is allocated to the operand "Tag_Result".

See also

EXIT: Exit loop immediately (Page 1842)

CONTINUE: Recheck loop condition (Page 1841)

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

REPEAT: Run if condition is not met

Description

The instruction "Run if condition is not met" causes a program loop to be repeatedly executed until a termination condition is met. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

The instructions are executed once, even if the termination condition is fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

You can declare the instruction as follows:

REPEAT <instructions>;

UNTIL <condition> END_REPEAT;

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
<Instructions>	-		Instructions that are executed as long as the programmed condition has the value FALSE. The instructions are executed once, even if the termination condition is fulfilled.
<Condition>	BOOL	I, Q, M, D, L	Expression which is evaluated after each loop. If the expression has the value FALSE, the program loop is executed once again. If the expression has the value TRUE, the program loop continues after END_REPEAT.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
REPEAT "Tag_Result"
        := "Tag_Value";
UNTIL "Tag_Error"
END_REPEAT;
```

As long as the value of the operand "Tag_Error" has the signal state "0", the value of the operand "Tag_Value" is allocated to the operand "Tag_Result".

See also

CONTINUE: Recheck loop condition (Page 1841)

EXIT: Exit loop immediately (Page 1842)

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

CONTINUE: Recheck loop condition

Description

The "Recheck loop condition" instruction ends the current program run of a FOR, WHILE or REPEAT loop.

After execution of the instruction, the conditions for the continuation of the program loop are evaluated again. The instruction affects the program loop which directly contains the instruction.

Example

The following example shows how the instruction works:

SCL

```
FOR i
     := 1 TO 15 BY 2 DO
     IF (i < 5) THEN
     CONTINUE;
END_IF;
      "DB10".Test[i] := 1;
END_FOR;
```

For additional information on valid data types, refer to "See also".

If the condition i < 5 is satisfied, the subsequent value allocation ("DB10".Test[i] := 1) will not be executed. The run variable (i) is increased by the increment of "2" and checked to see whether its current value lies in the programmed value range. If the run variable lies in the value range, the IF condition is evaluated again.

If the condition i < 5 is not satisfied, the subsequent value allocation ("DB10".Test[i] := 1) will be executed and a new loop started. In this case, the run variable is also increased by the increment "2" and checked.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

EXIT: Exit loop immediately

Description

The instruction "Exit loop immediately" cancels the execution of a FOR, WHILE or REPEAT loop at any point regardless of conditions. The execution of the program is continued after the end of the loop (END_FOR, END_WHILE, END_REPEAT).

The instruction affects the program loop which directly contains the instruction.

Example

The following example shows how the instruction works:

SCL

```
FOR i := 15 TO 1 BY -2 DO
IF (i < 5)
THEN EXIT;
END_IF;
"DB10".Test[i] := 1;
END_FOR;
```

For additional information on valid data types, refer to "See also".

If the condition i < 5 is satisfied, then the execution of the loop will be canceled. Program execution resumes after the END_FOR.

If the condition i < 5 is not satisfied, the subsequent value allocation ("DB10".Test[i] :=1) will be executed and a new loop started. The run tag (i) is decreased by the increment of "-2" and it is checked whether its current value lies in the programmed value range. If the (i) run variable lies within the value range, the IF condition is evaluated again.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

GOTO: Jump

Description

Use the instruction "Jump" to resume the execution of a program at a given point marked with a jump label.

The jump labels and the instruction "Jump" must be in the same block. The name of a jump label can only be assigned once within a block. Each jump label can be the target of several jump instructions.

A jump from the "outside" into a program loop is not permitted, but a jump from a loop to the "outside" is possible.

Adhere to the following grammatical rules for jump labels:

- Letters (a to z, A to Z)
- A combination of letters and numbers. Check that the order is correct, i.e. first the letters, then the numbers (a - z, A - Z, 0 - 9).
- Special characters or a combination of letters and numbers must not be used in reverse order, i.e. first the numbers and then the letters (0-9, a - z, A - Z).

You can declare the instruction as follows:

GOTO <jump label>;

...

<Jump label>: <Instructions>

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Description
<jump label>	-	Jump label to be jumped to
<instructions>	-	Instructions which are executed after the jump.

Example

The following example shows how the instruction works:

SCL

```
CASE "Tag_Value" OF
1 : GOTO MyLABEL1;
2 : GOTO MyLABEL2;
3 : GOTO MyLABEL3;
ELSE GOTO MyLABEL4;
END_CASE;
MyLABEL1: "Tag_1" := 1;
MyLABEL2: "Tag_2" := 1;
MyLABEL3: "Tag_3" := 1;
MyLABEL4: "Tag_4" := 1;
```

Depending on the value of the "Tag_Value" operand, the execution of the program will resume at the point identified by the corresponding jump label. If the operand "Tag_Value" has the value 2, for example, program execution will resume at the jump label "MyLABEL2". The program line identified by the jump label "MyLABEL1" will be skipped in this case.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

RETURN: Exit block

Description

The instruction "Exit block" exits the program execution in the currently edited block and continues in the calling block.

The instruction can be omitted at the end of the block.

Example

The following example shows how the instruction works:

SCL

```
IF "Tag_Error" <>0 THEN RETURN;
END_IF;
```

If the signal state of the "Tag_Error" operand is not zero, execution of the program ends in the block currently being processed.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

(*...*): Insert a comment section

Description

You can use the "Insert a comment section" instruction to add a comment section. The text inside the parentheses "(*...*)" is handled as a comment.

Example

The following example shows how the instruction works:

SCL

```
(*This is a comment section.*)
```

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

REGION: Structure program code

Description

You can use the instruction "Structure program code" to structure the program code in an SCL block by dividing it into different areas.

For additional information on working with areas, please refer to:Working with regions (Page 7331)

You can declare the instruction as follows:

REGION <Name>

<Instructions>

END_REGION

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
REGION	-	-	Keyword with which the area starts
<Name>	-	-	Text designating the REGION
<Instructions>	-	-	Program code that is preceded and succeeded by the REGION
END_REGION	-	-	Keyword with which the area ends

Example

The following example shows how the instruction works:

SCL

```
REGION Feeder System
     // Structures the source code for the feeder system
   IF "Variable_1" = 0 THEN
      "Variable_2" := 10;
   END_IF;
END_REGION
```

The Feeder System area comprises the complete program code.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether passwords configured for the CPU are legitimized or not. In this way, you can prevent legitimized connections, even if the correct password is known.

When you call the instruction and the REQ parameter has the signal state "0", the currently set state is displayed at the output parameters. If changes have been made to the input parameters, these are not transferred to the output parameters.

When the instruction is called and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD).

- If signal state "0" is present, authorization via password is not allowed.
- If signal state "1" is present, the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Passwords that are not configured must have the signal state TRUE at the input and return the signal state TRUE at the output. The fail-safe password can only be configured for an F-CPU and must therefore always be interconnected with the signal state TRUE in a standard CPU. If the instruction returns an error, the call has no effect and the previous lock is still in effect.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the mode selector to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Plugging an empty memory card (transfer card or program card) into an S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called once again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access from HMI systems if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing accidental lockout of an S7-1500 CPU

The settings can be made on the front panel of the CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode switch to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing accidental lockout of an S7-1200 CPU

Because the S7-1200 CPU does not have a mode selector, protection is disabled with POWER OFF - POWER ON. This means that it is possible and advisable to prevent accidental lockout with the help of specific program sequences within your program.

To do so, program a time control either by means of a cyclic interrupt OB or a timer in the Main OB (OB 1). This gives you the option of calling the "Limit and enable password legitimation" instruction in the respective OB (e.g., OB 1 or OB 35) relatively quickly after a transition from POWER OFF to POWER ON and the associated disabling of the protection. Call the instruction in the startup OB (OB 100) to keep the time window in which the instruction is inactive and there are no limitations for password legitimation relatively small. This procedure will give you the best possible protection from unauthorized access.

If there has been an accidental lockout, you can skip the call in the startup OB (by querying an input parameter, for example) and you have the set time (for example, 10 seconds up to 1 minute) to establish a connection to the CPU before the lock becomes active once again.

If there is no timer in your program code and a lockout has occurred, insert an empty transfer card or an empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then download the user program once again from STEP 7 to the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

WARNING

Inserting an empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Once you have pulled the transfer card, the content of the transfer card is available in the internal load memory. Make sure that the card does not include a program at this point.

WARNING

Inserting an empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. Once you have pulled the previously empty program card, the internal load memory is empty.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200 CPU: The lock is disabled and the instruction must be called once again in the program (e.g., in the startup OB). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password

Parameters	Declaration	Data type	Memory area	Description
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
Function value (RET_VAL)		WORD	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported
80D0	The password for fail-safe is not configured. The signal state must be TRUE for standard CPUs.
80D1	The read/write access is not configured

Error code* (W#16#...)	Explanation
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ENDIS_PW(REQ := 0,
                         F_PWD := 0,
                         FULL_PWD := 0,
                         R_PWD := 1,
                         HMI_PWD := 0,
                         F_PWD_ON => "Status_F_PWD",
                         FULL_PWD_ON => "Status_FULL_PWD",
                         R_PWD_ON => "Status_R_PWD",
                         HMI_PWD_ON => "Status_HMI_PWD");
```

The following table shows how the instruction works using specific operand values:

Operand	Data type	Value
REQ	BOOL	0
F_PWD	BOOL	0
FULL_PWD	BOOL	0
R_PWD	BOOL	1
HMI_PWD	BOOL	0
Status_F_PWD	BOOL	0
Status_FULL_PWD	BOOL	0
Status_R_PWD	BOOL	1
Status_HMI_PWD	BOOL	0

The instruction is executed because the REQ operand has the signal state "1". The R_PWD operand has the signal state "1", which means that read access is allowed when the assigned password is entered. The R_PWD_ON status operand also has the signal state "1", thereby signaling that the R_PWD operand is activated.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

PC systems

SHUT_DOWN: Shut down target system

Description

You can use the instruction "SHUT_DOWN: Shutdown target system" to shut down the PC-based automation system and restart the S7 software controller CPU 150xS or Windows on the PC-based automation system.

The instruction is available in the "Instructions" task card under Basic instructions > Program control > Runtime control.

A restart is, for example, advisable in the following situations:

- An industrial UPS (uninterruptible power supply) reports a power failure via a digital input.
- Windows stops responding or shows a "blue screen".
- Too many error OBs are called in the user program.

Parameter

The following table shows the parameters of the "Shutdown target system" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	MODE = 1: Shut down CPU 150xS and Windows The CPU goes to STOP and saves the retentive data. Then the CPU and Windows are shut down. The system must be restarted manually. MODE = 2: Restart CPU 150xS The CPU goes to STOP and saves the retentive data. Then the CPU is shut down and restarted. MODE = 3: Restart Windows The CPU remains in RUN. Windows is restarted (as of TIA Portal V14, MODE 3 is only retained for backward compatibility. We recommend using MODE 4 or MODE 5). MODE = 4: Windows is shut down and restarted correctly. The CPU remains in RUN. MODE = 5: Restart Windows (Comparable with MODE 3; Exception: MODE 5 should only be used if Windows has crashed).
COM-MENT	Input	STRING	I, Q, M, D, L	With Mode = 1, 3 and 4, you can specify a reason for the restart. The reason is output in the Windows Event Log.
Ret_Val	Return	WORD	Q, M, D, L	Ret_Val = 0: No faults Ret_Val = 8090: The value passed to MODE is not supported. Ret_Val = 8091: Windows does not respond to the call of the Shut_Down instruction (only valid for Mode 3 and 4). Ret_Val = 8092: Please contact SIMATIC Customer Support if this error occurs (only valid for Mode 3 and 4).

RE_TRIGR: Restart cycle monitoring time

Description

The "Restart cycle monitoring time" instruction is used to restart the cycle monitoring time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

The instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Calling the instruction

The following call conditions apply:

- The following applies to a CPU of the S7-1200 series:
  In firmware versions <= 2.2, you can call the instruction only in a program cycle organization block of priority 1. This corresponds to the lowest priority of all organization blocks. If the instruction is called in an organization block with a higher priority, the instruction is not executed and the result (BR bit, enable output ENO) is always "0".
  In firmware versions >= 2.2, you can call the instruction regardless of the priority in all organization blocks.
- The following applies to a CPU of the S7-1500 series:
  You can call the instruction regardless of the priority in all organization blocks.

Parameters

The "Restart cycle monitoring time" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a program block. This normally involves programming or access errors. If the system reports an error during the execution of the program block, detailed information is output at the <Operand> on the first error to occur during the execution of the block since the last execution of the instruction.

The error information can only be saved in operands of the "ErrorStruct" system data type. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the program block, the error information about the next error to occur is output by the instruction only after the first error that occurred has been remedied.

Note

<Operand>

The <Operand> is only changed if error information is present. To set the operand back to "0" after handling the error, you have the following options:

- Declare the operand in the "Temp" section of the block interface.
- Reset the operand to "0" before calling the instruction.

Note

Activation of the local error handling

As soon as you insert the instruction in the program code of a program block, the local error handling is activated and default system reactions are ignored when errors occur.

Options for error handling

You can find an overview of the options for error handling here: Overview of mechanisms for error handling (Page 138)

For a detailed example of local error handling that contains several error handling options, refer to: Example of how to handle program execution errors (Page 167)

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
<Operand>	ErrorStruct	D, L	Information about errors that have occurred

Data type "ErrorStruct"

The "ErrorStruct" data type can be inserted in a global data block or in the block interface. You can insert the data type more then once if you assign a different name for the data structure each time. The data structure and the name of individual structure elements cannot be changed. If you save the error information in a global data block, it can also be read out by other program blocks.

The following table shows the structure of the "ErrorStruct" data type:

Structure component		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows whether the error occurred during a program block call. 16#01: Error during a program block call 16#00: No error during a program block call
REACTION		BYTE	Default reaction: 0: Ignore (write error) 1: Continue with substitute value "0" (read error) 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of the program block
	BLOCK_TYPE	BYTE	Program block type in which the error occurred: 1: Organization block (OB) 2: Function (FC) 3: Function block (FB)
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand
	AREA	BYTE	(C) Memory area: L: 16#40...16#7F, 16#86, 16#87, 16#8E, 16#8F, 16#C0...16#FF I: 16#81 Q: 16#82 M: 16#83 DB: 16#40, 16#84, 16#85, 16#8A, 16#8B PI: 16#01 PQ: 16#02 Technological objects: 16#04

Structure component		Data type	Description
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

The following table shows the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid reference used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:

SCL

```
LABEL: #TagOut := #Field[#index] * REAL#40.5;

GET_ERROR(#Error);

IF #Error.REACTION = 1 THEN
   GOTO LABEL;
   ;
END_IF;
```

An error occurred accessing the #Field[#index] tag. The #TagOut operand returns the signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, we recommend that you program the "Get error locally" instruction after the multiplication to get the error. The error information supplied by the "Get error locally" instruction is evaluated with a comparison. If the #Error.REACTION structure component has the value "1", there is a read/access error and the program execution begins again at the jump label LABEL.

See also

Switching display formats in the program status (Page 7698)

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

Basics of SCL (Page 7293)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the instruction outputs the error ID of the first error that occurred.

The error ID can only be saved in operands of the WORD data type. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The <Operand> is only changed if error information is present. To set the operand back to "0" after handling the error, you have the following options:

- Declare the operand in the "Temp" section of the block interface.
- Reset the operand to "0" before calling the instruction.

The output of the "Get error ID locally" instruction is only set if error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Options for error handling

You can find an overview of the options for error handling here: Overview of mechanisms for error handling (Page 138)

For a detailed example of local error handling that contains several error handling options, refer to: Example of how to handle program execution errors (Page 167)

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
Function value	WORD	D, L	Error ID

Error ID

The following table shows the values that can be output:

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING

ID* (hexadecimal)	ID* (decimal)	Description
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid reference used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:

SCL

```
#TagOut := #Field[#index] * REAL#40.5;

#TagID := GET_ERR_ID();

IF #TagID = 16#2522 THEN
   MOVE_BLK(IN := #TagArrayIn[0],
            COUNT := 1,
            OUT => #TagArrayOut[1]);
END_IF;
```

An error occurred accessing the #Field[#index] tag. The #TagOut operand returns the signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, we recommend that you program the "Get error ID locally" instruction after the multiplication to get the error. The error ID supplied by the "Get error ID locally" instruction is evaluated with a comparison. If the #TagID operand returns ID 16#2522, there is a read/access error and the value "100.0" of the #TagArrayIn[0] operand is written to the #TagArrayOut[1] operand.

See also

Switching display formats in the program status (Page 7698)

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

Basics of SCL (Page 7293)

INIT_RD: Initialize all retain data (S7-1500)

Description

The "Initialize all retain data" instruction is used to reset the retentive data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution exceeds the program cycle duration.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L	If the input "REQ" has the signal state "1", all retentive data are reset.
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally"
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := INIT_RD("Tag_REQ");
```

If the operand "Tag_REQ" has the signal state "1", the instruction is executed. All retentive data of all data blocks, bit memories and SIMATIC timers and counters are reset.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1857)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

WAIT: Configure time delay (S7-1500)

Description

The "Configure time delay" instruction pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to 32767 microseconds (µs). The smallest possible delay time depends on the CPU and corresponds to the execution time of the instruction.

The execution of the instruction can be interrupted by higher priority events and does not return any error information.

Note

Negative delay time

If you specify a negative delay time at parameter WT, the enable output ENO, or the RLO and the BR bit, will return the signal state FALSE. A negative delay time does not affect the CPU. The following instructions linked to enable output ENO are not executed in LAD or FBD.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

Example of the influence of the planned delay time

In the following example, you will see the influence of the delay time of the instruction "WAIT" in various scenarios.

The following figure is a schematic representation of the scenarios:

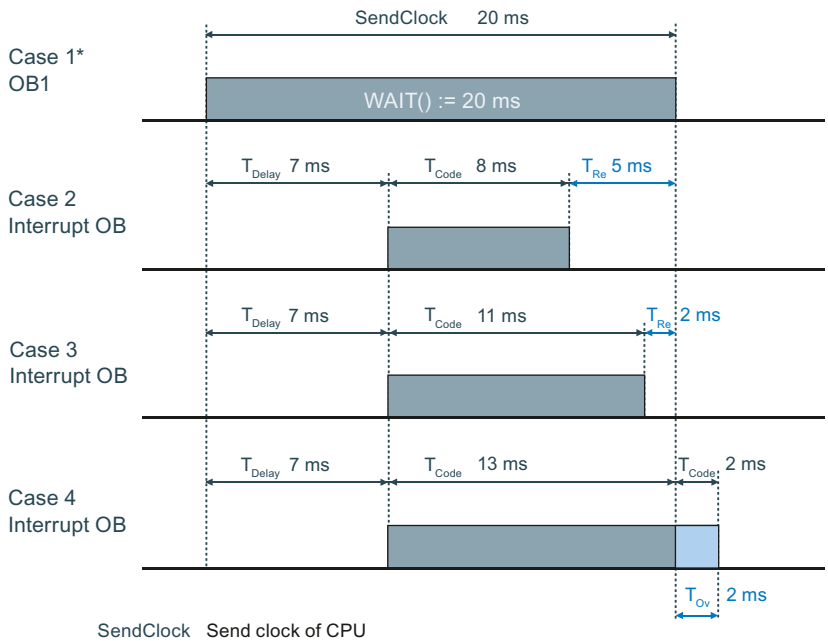

SendClock	Send clock of CPU
T_{Delay}	Delay time of the interrupt OB
T_{Code}	Time of program execution
T_{Re}	Remaining time
T_{Ov}	Excess time by which the send clock is increased

* The OB1 with the instruction "Wait" is also part of cases 2, 3 and 4 and needs to be taken into account.

Remaining time = The period between the end of the planned delay time (via "WAIT") until the end of the interrupt OB

Excess time = The period between the end of the interrupt OB and the end of the planned delay time (via "WAIT").

Case 1:

The "WAIT" instruction is called in an OB1. The "WAIT" instruction can be interrupted by higher-priority OBs or higher-priority processes (e.g. System Threads). However, the delay time of the "WAIT" instruction is neither changed nor deferred.

Cases 2 and 3:

Processing of the program in OB1 is resumed after a time delay of 20 ms. This time delay is calculated by calling the "WAIT" instruction in OB1 (see OB1 with WAIT). Within these 20 ms, an interrupt OB can run its own program code. The send clock of the CPU is not changed.

Case 4:

Processing of the program in OB1 is resumed after the higher-priority process has ended. The 20 ms delay time in OB1 has elapsed, but the higher-priority process still needs to be ended. The send clock of the CPU is increased.

Note

Order of execution of system or communication processes (System Threads)

The System Threads usually use the priority "15". There are also System Threads with a higher priority than "26", but these processes cause a lower CPU load. System Threads are not displayed in the figure.

Runtime measurement of the OB1 using the "RT_INFO" instruction:

Case 2: 20 ms - 8 ms - System Threads = <12 ms. Send clock: ~20 ms.

Case 3: 20 ms - 11 ms - System Threads - <9 ms. Send clock: ~20 ms.

Case 4: 20 ms - 15 ms - System Threads - <7 ms. Send clock: ~22 ms.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

RUNTIME: Measure program runtime

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and writes the value to the in/out parameter. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the in/out parameter as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The "Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overruns, the instruction returns values <= 0.0. This can occur up to once per minute for S7-1200 CPUs with firmware version <V4.2. Ignore these runtime values.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	InOut	LREAL	I, Q, M, D, L	The content is for internal purposes only.
Function value		LREAL	I, Q, M, D, L	Returns the measured runtime in seconds

Example

The following example shows the how the instruction works based on the runtime calculation of a program block:

SCL

```
"Tag_Result" := RUNTIME("Tag_Memory");

"Best_before_date_DB" ();

"Tag_Result" := RUNTIME("Tag_Memory");
```

The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called.

When the program block FB1 has been processed, the instruction is executed a second time. The second call of the instruction calculates the runtime of the program block and writes the result to the output "Tag_Result".

You can find a more detailed example of how you can measure the total cycle time of a program in the Siemens Industry Online Support under the following entry ID: 87668055 (https://support.industry.siemens.com/cs/ww/en/view/87668055)

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Word logic operations

DECO: Decode

Description

The instruction "Decode" sets a bit specified by the input value in the output value.

The instruction "Decode" reads the value of the parameter IN and sets the bit in the output value, whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. If the value of the IN parameter is greater than 31, a modulo 32 instruction is executed.

Use the following syntax to change the data type of the instruction:

DECO_<data type>();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	UINT	I, Q, M, D, L, P	Position of the bit in the output value which is set.
_<Data type>		Bit strings default: DWORD	-	Data type of the function value: 1. You can specify the data type of the instruction explicitly using "_". 2. If you do not specify the data type explicitly, it will be determined by the utilized tags or type-coded constants. 3. If you neither specify the data type explicitly nor specify defined tags or type-coded constants, the default data type will be used.
Function value		Bit strings	I, Q, M, D, L, P	Current output value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := DECO(IN := "Tag_Value");
"Tag_Result2" := DECO_BYTE(IN := "Tag_Value2");
```

The following figure shows how the instruction works using specific operand values:

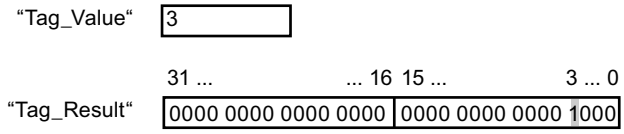

The instruction reads the number "3" from the value of the operand "Tag_Value" and sets the third bit to the value of the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ENCO: Encode

Description

The instruction "Encode" reads the bit number of the lowest-value bit set in the input value and issues this as a result.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P	Input value
Function value		INT	I, Q, M, D, L, P	Bit number of the bit in the input value that is read out.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ENCO(IN := "Tag_Value");
```

The following figure shows how the instruction works using specific operand values:

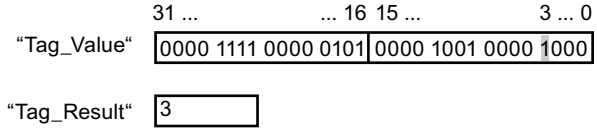

The instruction reads the lowest-value set bit of the operand "Tag_Value" and writes the bit position "3" in the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SEL: Select

Description

The instruction "Select" selects one of the parameters IN0 or IN1 depending on a switch (G parameter) and issues its content as a result. When the parameter G has the signal status "0", the value at parameter IN0 is moved. When the parameter G has the signal status "1", the value at parameter IN1 is moved and returned as a function value.

The instruction is only executed if the tags of all parameters have the same data type class.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
G	Input	BOOL	BOOL	I, Q, M, D, L	Switch
IN0	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, timers, strings, DATE, TOD, LTOD, DT, LDT	I, Q, M, D, L, P	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, timers, strings, DATE, TOD, LTOD, DT, LDT	I, Q, M, D, L, P	Second input value
Function value		Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, timers, strings, DATE, TOD, LTOD, DT, LDT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SEL(G := "Tag_Value",
                    IN0 := "Tag_0",
                    IN1 := "Tag_1");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0	1
Tag_0	W#16#0000	W#16#4C
Tag_1	W#16#FFFF	D#16#5E
Tag_Result	W#16#0000	D#16#5E

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

MUX: Multiplex

Description

The "Multiplex" instruction copies the value of a selected input parameter and issues it. You can use the parameter K to determine the number of the input parameter whose value will be moved. Numbering starts at IN0 and is incremented continuously with each new input. You can declare a maximum of 32 inputs.

Numerical data types and time data types are permitted at the inputs. All tags with assigned parameters must be of the same data type.

The function value is invalid if any of the following requirements are met:

- Errors occurred during execution of the instruction.
- If the input at the K parameter is located outside the available inputs and the INELSE input is not used, the function value is invalid.

Note

K parameter has a negative integer

If a tag with a valid data type is specified at the input parameters and the K parameter has a negative integer, the value of the tag is changed.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
K	Input	Integers	Integers	I, Q, M, D, L, P	Specifies the parameter whose content is to be transferred. • If K = 0 => Parameter IN0 • If K = 1 => Parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	I, Q, M, D, L, P	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	I, Q, M, D, L, P	Second input value
INn	Input	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	I, Q, M, D, L, P	Optional input values

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
INELSE	Input	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	I, Q, M, D, L, P	Specifies the value to be copied when K <> n.
Function value		Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MUX(K := "Tag_Number",
                    IN0 := "Tag_1",
                    IN1 := "Tag_2",
                    INELSE := "Tag_3");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Number	1	4
Tag_1	DW#16#00000000	DW#16#00000000
Tag_2	DW#16#003E4A7D	DW#16#003E4A7D
Tag_3	DW#16#FFFF0000	DW#16#FFFF0000
Tag_Result	DW#16#003E4A7D	DW#16#FFFF0000

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

DEMUX: Demultiplex

Description

The "Demultiplex" instruction transfers the value of the input parameter IN to a selected output parameter. The selection of the input parameter takes place independent of the parameter value K. The K parameter specifies the output parameter number to which the value of the input parameter IN is transferred. The other output parameters are not changed. Numbering starts at OUT0 and continues consecutively with each new output. You can declare a maximum of 32 output parameters.

If the value of the K parameter is greater than the number of output parameters, the content of the IN input parameter is copied to the OUTELSE output parameter and the signal state "0" is assigned to the enable output ENO.

The function value is invalid if any of the following requirements are met:

- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Note

K parameter < 0 or K > available outputs

If you specify at the K parameter a value that is outside the available outputs (K < 0 or K > available outputs), the value of the IN input parameter is output at the OUTELSE output parameter.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
K	Input	Integers	Integers	I, Q, M, D, L, P	Specifies the output to which the input value (IN) will be copied. • If K = 0 => Parameter OUT0 • If K = 1 => Parameter OUT1, etc.
IN	Input	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	Input value
OUT0	Output	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	First output
OUT1	Output	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	Second output

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
OUTn	Output	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	Optional outputs
OUTELSE	Output	Binary numbers, integers, floating-point numbers, timers, STRING, CHAR, WCHAR, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P	Output to which the value at input IN is copied if K > n.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
DEMUX(K := "Tag_Number",
      IN := "Tag_Value",
      OUT0 := "Tag_1",
      OUT1 := "Tag_2",
      OUTELSE := "Tag_3");
```

The following tables show how the instruction works using specific operand values:

Input values of the "Demultiplex" instruction before the network execution

Parameters	Operand	Values	
K	Tag_Number	2	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

Output values of the "Demultiplex" instruction after the network execution

Parameters	Operand	Values	
OUT0	Tag_1	Unchanged	Unchanged
OUT1	Tag_2	DW#16#FFFFFFFF	Unchanged
OUTELSE	Tag_3	Unchanged	DW#16#003E4A7D

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Shift and Rotate

SHR: Shift right

Description

The "Shift right" instruction shifts the contents of the IN parameter bit-by-bit to the right and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of available bit places, then the value of the IN parameter is shifted to the right by the available number of bit places.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure shows how the content of an integer data type operand is shifted by four bit positions to the right:

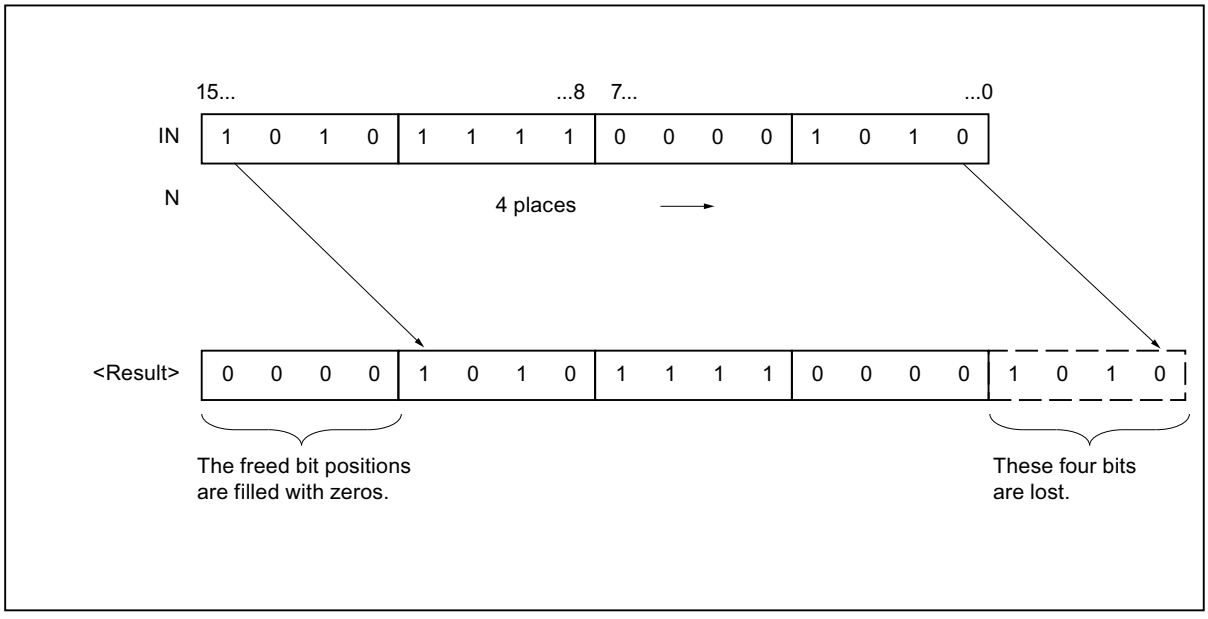

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Number of bits by which the value (IN) is shifted
Function value		Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SHR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	3
Function value	Tag_Result	0000 0111 1111 0101

The content of the "Tag_Value" operand is shifted by three bit positions to the right. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SHL: Shift left

Description

The "Shift left" instruction shifts the contents of the IN parameter bit-by-bit to the left and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of bit places, the value of the IN parameter is shifted to the left by the available number of bit places.

The bit positions freed by the shift are filled with zeros in the result value.

The following figure shows how the content of an operand of the WORD data type is shifted six bit positions to the left:

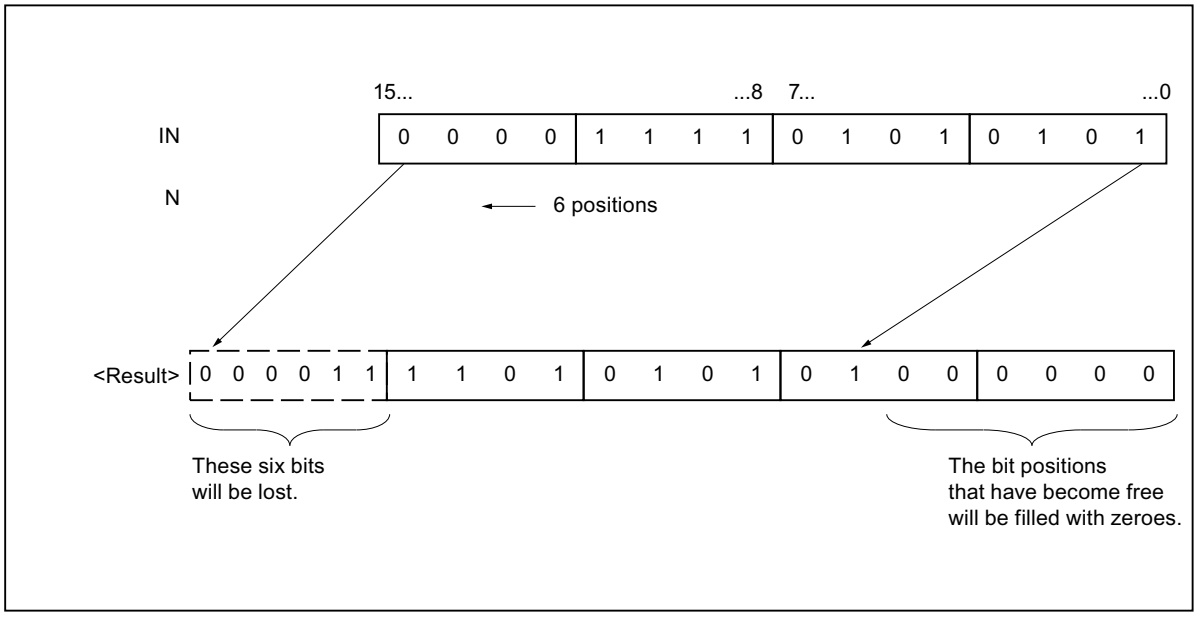

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Number of bits by which the value (IN) is shifted
Function value		Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SHL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	4
Function value	Tag_Result	1111 1010 1111 0000

The value of the "Tag_Value" operand is shifted by four bit places to the left. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ROR: Rotate right

Description

The "Rotate right" instruction rotates the content of the IN parameter bit-by-bit to the right and assigns the result to the specified operand. The parameter N is used to specify the number of bit places by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:

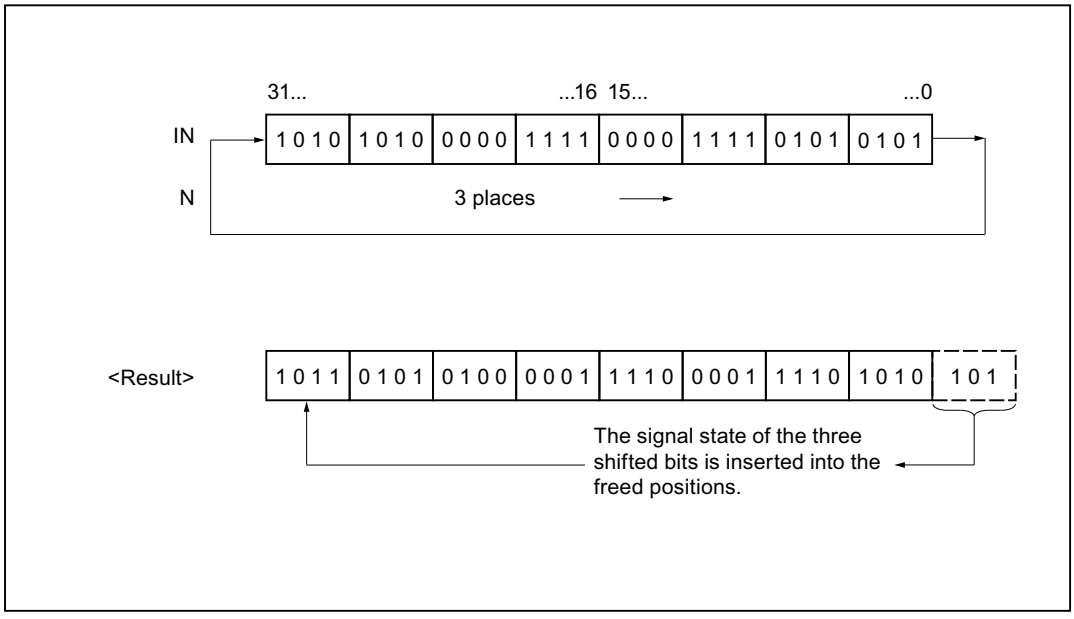

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Number of bit positions by which the value (IN) is rotated
Function value		Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ROR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0000 1111 1001 0101
N	Tag_Number	5
Function value	Tag_Result	1010 1000 0111 1100

The content of the "Tag_Value" operand is rotated by five bit positions to the right. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ROL: Rotate left

Description

The "Rotate left" instruction rotates the contents of the IN parameter bit-by-bit to the left and returns it as a function value. The parameter N is used to specify the number of bit places by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:

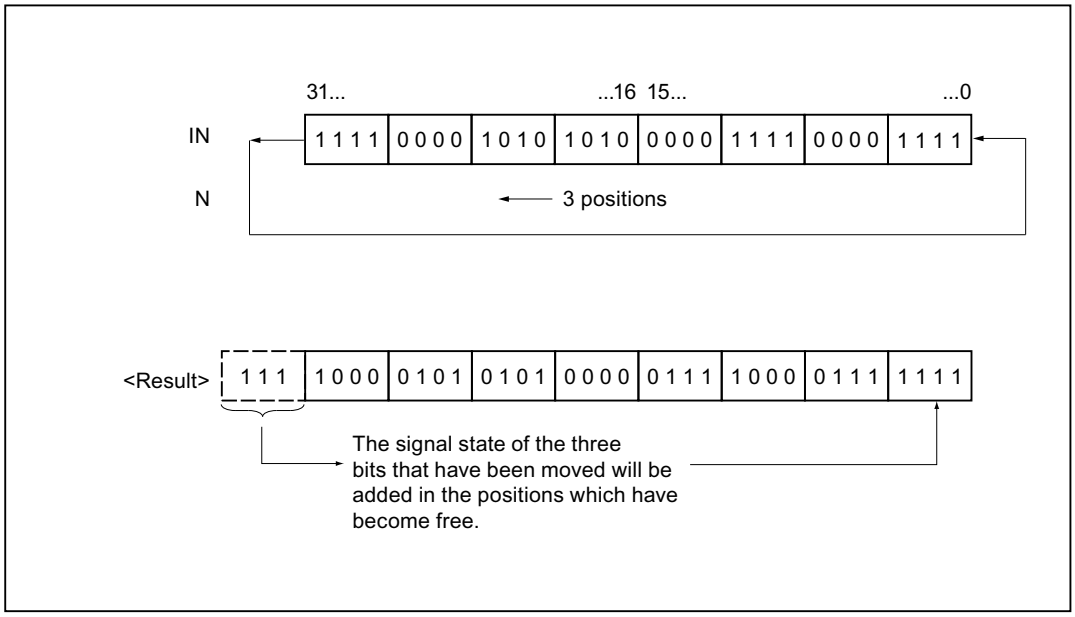

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Number of bit positions by which the value (IN) is rotated
Function value		Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ROL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1010 1000 1111 0110
N	Tag_Number	5
Function value	Tag_Result	0001 1110 1101 0101

The content of the operand "Tag_Value" is rotated five bit places to the left. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Legacy (S7-1500)

DRUM: Implement sequencer (S7-1500)

Description

You can use the "Implement sequencer" instruction to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). When a bit of the configurable mask has a signal state of "1", the OUT_VAL value sets/resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L	The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L	Maximum step number (for example: LST_STEP = 16#08; a maximum of 8 steps is possible.)
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P	Current counter value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L	Previous system time
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L	Preset counter value for each step [1 to 16] where one clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

SCL

```
"DRUM_DB"(RESET := "Tag_Reset"
          JOG := "Tag_Input_Jog"
          DRUM_EN := "Tag_Input_DrumEN"
          LST_STEP := "Tag_Number_LastStep"
          EVENT1 := "MyTag_Event_1"
          EVENT2 := "MyTag_Event_2"
          EVENT3 := "MyTag_Event_3"
          EVENT4 := "MyTag_Event_4"
          EVENT5 := "MyTag_Event_5"
          EVENT6 := "MyTag_Event_6"
          EVENT7 := "MyTag_Event_7"
          EVENT8 := "MyTag_Event_8"
          EVENT9 := "MyTag_Event_9"
          EVENT10 := "MyTag_Event_10"
          EVENT11 := "MyTag_Event_11"
          EVENT12 := "MyTag_Event_12"
          EVENT13 := "MyTag_Event_13"
          EVENT14 := "MyTag_Event_14"
          EVENT15 := "MyTag_Event_15"
```

SCL

```
EVENT16 := "MyTag_Event_16"
OUT1 => "MyTag_Output_1"
OUT2 => "MyTag_Output_2"
OUT3 => "MyTag_Output_3"
OUT4 => "MyTag_Output_4"
OUT5 => "MyTag_Output_5"
OUT6 => "MyTag_Output_6"
OUT7 => "MyTag_Output_7"
OUT8 => "MyTag_Output_8"
OUT9 => "MyTag_Output_9"
OUT10 => "MyTag_Output_10"
OUT11 => "MyTag_Output_11"
OUT12 => "MyTag_Output_12"
OUT13 => "MyTag_Output_13"
OUT14 => "MyTag_Output_14"
OUT15 => "MyTag_Output_15"
OUT16 => "MyTag_Output_16"
Q => "Tag_Output_Q"
OUT_WORD => "Tag_OutputWord"
ERR_CODE => "Tag_ErrorCode");
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE

Parameters	Address	Value
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameters	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

DCAT: Discrete control-timer alarm (S7-1500)

Description

You can use the "Discrete control-timer alarm" to count the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET is < PT) OA and CA are influenced as follows:
  - The Q and CMD_HIS parameters are set to "1".
  - The ET, OA, and CA parameters are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".
- When the signal state of the parameters CMD and CMD_HIS is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state at the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- When the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	D, L	Preset timer value, where one clock pulse = 1 ms.
PREV_TIME	Static	DWORD	D, L	Previous system time
CMD_HIS	Static	BOOL	D, L	CMD history bit

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The CMD_HIS parameter of the instance data block is set to the signal state "1" and the ET parameter is reset to "0".

Note

You can initialize static parameters in the data block.

SCL

```
"DCAT_DB"(CMD := "Tag_Input_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          Q => "Tag_Output_Q",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA");
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

MCAT: Motor control-timer alarm (S7-1500)

Description

The "Motor control-timer alarm" instruction is used to accumulate the time from the point at which one of the command inputs (opening or closing) is switched on. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory space	Description
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	A signal state of "0" indicates an error condition.
ET	Static	DINT	D, L	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	D, L	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L	Previous system time
O_HIS	Static	BOOL	D, L	"Open" history bit
C_HIS	Static	BOOL	D, L	"Close" history bit

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close

Input parameters								Output parameters								
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore not possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

SCL

```
"MCAT_DB"(O_CMD := "Tag_Iput_O_CMD",
          C_CMD := "Tag_Input_C_CMD",
          S_CMD := "Tag_Input_S_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          OO => "Tag_OutputOpen",
          CO => "Tag_OutputClosed",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA",
          Q => "Tag_Output_Q");
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

IMC: Compare input bits with the bits of a mask (S7-1500)

Description

You can use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.

Parameters	Declaration	Data type	Memory area	Description
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

SMC: Compare scan matrix (S7-1500)

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.

Parameters	Declaration	Data type	Memory area	Description
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

LEAD_LAG: Lead and lag algorithm (S7-1500)

Description

You can use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the instruction "Lead and lag algorithm" in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of output OUT so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P	The input value of the current sample time (cycle time) to be processed. Constants can also be specified at the IN parameter.
SAMPLE_T	Input	INT	I, Q, M, D, L, P	Sample time Constants can also be specified at the SAMPLE_T parameter.
OUT	Output	REAL	I, Q, M, D, L	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L	Error information

Parameters	Declaration	Data type	Memory area	Description
LD_TIME	Static	REAL	I, Q, M, D, L, P	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P	Previous output

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

SCL

```
"LEAD_LAG_DB"(IN := "Tag_Input",
              SAMPLE_T := "Tag_Input_SAMPLE_T",
              OUT => "Tag_Output_Result",
              ERR_CODE => "Tag_ErrorCode");
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameters	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameters	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameters	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

SEG: Create bit pattern for seven-segment display (S7-1500)

Description

The "Create bit pattern for seven-segment display" instruction is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a 7-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	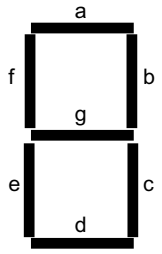
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display
Function value		VOID		Empty function value

Example

The following example shows how the instruction works:

SCL

```
SEG(IN := "Tag_Input",
    OUT => "Tag_Output");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
		Hexadecimal	Binary
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW16#065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

BCDCPL: Create tens complement (S7-1500)

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the operand. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Bit strings	I, Q, M, D, L, P	7-digit BCD number
Function value		DWORD	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := BCDCPL("Tag_Input");
```

The following table shows how the instruction functions using specific values:

Operand	Value*
Tag_Input	DW#16#01234567
Tag_Result	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

BITSUM: Count number of set bits (S7-1500)

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that are set to the signal state "1".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	DWORD	I, Q, M, D, L, P	Operand whose set bits are counted
Function value		INT	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := BITSUM("Tag_Input");
```

The following table shows how the instruction functions using specific values:

Operand	Value*
Tag_Input	DW#16#12345678
Tag_Result	W#16#000D (13 bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)
Switching display formats in the program status (Page 7698)
Basics of SCL (Page 7293)

4.1.3.5 GRAPH (S7-1500)

GRAPH sequencer (S7-1500)

Step and transition (S7-1500)

Description

You can use the "Step and transition" structure element to insert a step and transition simultaneously in your sequencer.

See also:

Step (Page 1905)
Transition (Page 1906)

See also

Step (Page 1905)
Transition (Page 1906)
Sequence end (Page 1907)
Jump to step (Page 1907)
Alternative branch (Page 1908)
Simultaneous branch (Page 1908)
Close branch (Page 1909)
Steps and transitions (Page 7375)

Step (S7-1500)

Description

Steps are used to subdivide complex automation tasks into straightforward subtasks that can then be accomplished using actions. The individual steps are organized in sequencers so that every step can be executed in the specified order in the program execution. Each step must be assigned a unique name and number.

For the execution to actually be performed, the step must be activated by one of the following conditions:

- The step has been defined as an initial step.
- The transition of the previous step has been fulfilled.
- The step has been called by an event-dependent action.

Once all actions have been executed, the step becomes inactive again.

Steps in which no actions are programmed are called empty steps. Such an empty step reacts like an active step and the subsequent transition is always fulfilled.

See also

Basics on steps (Page 7375)

Step and transition (Page 1905)

Transition (Page 1906)

Sequence end (Page 1907)

Jump to step (Page 1907)

Alternative branch (Page 1908)

Simultaneous branch (Page 1908)

Close branch (Page 1909)

Transition (S7-1500)

Description

Transitions are located between the steps and contain the conditions for advancing from one step to the next (step-enabling). When the step-enabling conditions of a transition are fulfilled, the next step becomes active and its actions are executed. You can program the conditions of a transition either in LAD or FBD.

See also

Basics on transitions (Page 7377)

Step and transition (Page 1905)

Step (Page 1905)

Sequence end (Page 1907)

Jump to step (Page 1907)

Alternative branch (Page 1908)

Simultaneous branch (Page 1908)

Close branch (Page 1909)

Sequence end (S7-1500)

Description

You can use the "Sequence end" structure element to stop a sequencer or a branch. In a simultaneous branch, a transition must precede the sequence end.

Note

If all branches of a sequencer are completed with a sequence end, the sequencer can be restarted by the "INIT_SQ" parameter or using the "Initialize" button in the "Sequence control" pane of the "Testing" task card.

See also

Elements of a sequencer (Page 7372)

Step and transition (Page 1905)

Step (Page 1905)

Transition (Page 1906)

Jump to step (Page 1907)

Alternative branch (Page 1908)

Simultaneous branch (Page 1908)

Close branch (Page 1909)

Jump to step (S7-1500)

Description

You can use a jump to continue the program execution at any step within the GRAPH function block. Jumps can be inserted at the end of the main branch or an alternative branch to enable cyclic processing of the sequencer. The jump and jump destination are represented in the sequencer as arrows, in which case the return transition is specified for the jump destination.

See also

Elements of a sequencer (Page 7372)

Step and transition (Page 1905)

Step (Page 1905)

Transition (Page 1906)

Sequence end (Page 1907)

Alternative branch (Page 1908)

Simultaneous branch (Page 1908)

Close branch (Page 1909)

Alternative branch (S7-1500)

Description

You can use alternative branches to program OR branches. This means that you insert branches that start with a transition after a step. The corresponding branch is executed depending on which transition is satisfied first. If several transitions are satisfied at the same time, the transition farthest to the left has the highest priority and the corresponding branch is executed. Alternative branches end once again with a transition.

You can program up to 125 alternative branches in a sequencer.

See also

Elements of a sequencer (Page 7372)

Step and transition (Page 1905)

Step (Page 1905)

Transition (Page 1906)

Sequence end (Page 1907)

Jump to step (Page 1907)

Simultaneous branch (Page 1908)

Close branch (Page 1909)

Simultaneous branch (S7-1500)

Description

You can use simultaneous branches to program AND branches. This means that you use one transition to activate multiple steps, the actions of which are then executed. Simultaneous branches always start with a step.

The subsequent transitions of the simultaneous branches are located on the main branch, in which case the various simultaneous branches can be connected to various points of the main branch. Note that branches that you join together in a transition only advance to the next step when all branches are completely executed.

You can program up to 249 simultaneous branches in a sequencer.

See also

Elements of a sequencer (Page 7372)

Step and transition (Page 1905)

Step (Page 1905)
Transition (Page 1906)
Sequence end (Page 1907)
Jump to step (Page 1907)
Alternative branch (Page 1908)
Close branch (Page 1909)

Close branch (S7-1500)

Description

You can use the "Close branch" element to close simultaneous and alternative branches to their parent branch. This is necessary if you do not want to complete the branch with a jump or a sequence end. In a simultaneous branch, you can insert "Close branch" only after a step.

See also

Elements of a sequencer (Page 7372)
Step and transition (Page 1905)
Step (Page 1905)
Transition (Page 1906)
Sequence end (Page 1907)
Jump to step (Page 1907)
Alternative branch (Page 1908)
Simultaneous branch (Page 1908)

GRAPH actions (S7-1500)

Timer operations (S7-1500)

TP: Generate pulse (S7-1500)

Description

You use the "Generate pulse" instruction to set the Q output for a programmed duration. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. The Q parameter is set for the time PT, regardless of the subsequent changes in the input signal. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the time duration PT is active.

You can query the current time value set in the ET parameter. The timer value starts at T#0s and ends when the value of time PT is reached. When time PT is reached and the signal state of the IN parameter is "0", the ET parameter is reset.

Note

If the time is not called in the program because it is skipped, for example, the ET parameter returns a constant value as soon as the time has expired.

Each call of the "Generate pulse" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Pulse output
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current timer value

You can select the data type of the instruction from the "???" drop-down list.

Pulse timing diagram

The following figure shows the pulse diagram of the "Generate pulse" instruction after the start:

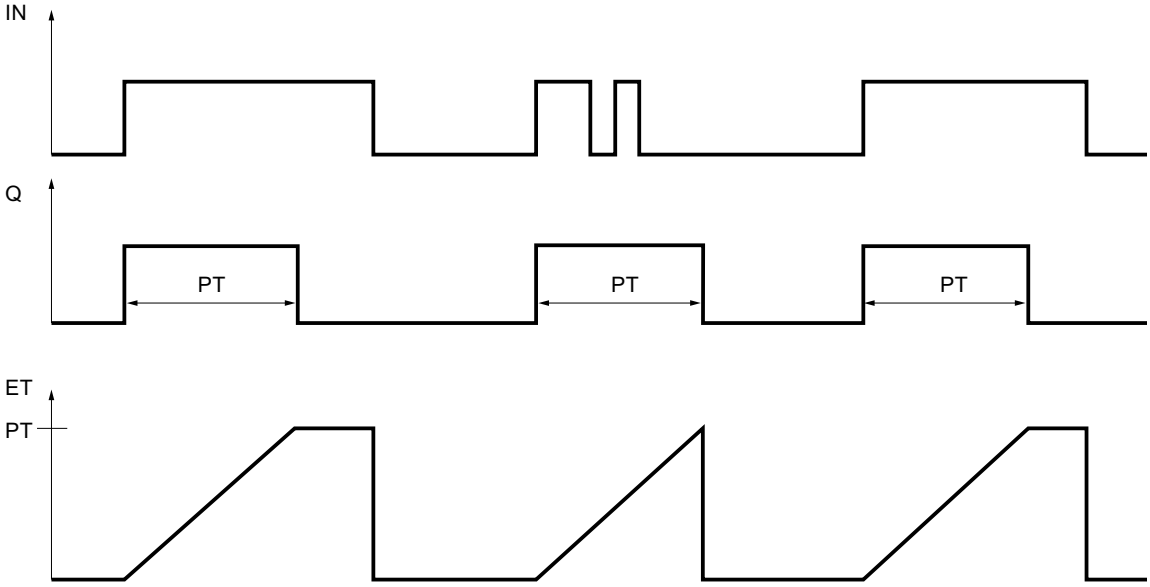

Example

The following example shows how the instruction works:

GRAPH

```
CALL TP TIME, "IEC_TP_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTIME" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

TON: Generate on-delay (S7-1500)

Description

You use the "Generate on-delay" instruction to delay setting of the Q output by the programmed time PT. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the time PT has elapsed, the Q parameter has the signal state "1". The Q parameter remains set as long as the start input IN is still "1". When the signal state at the start input changes from "1" to "0", the Q parameter is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of time PT is reached. The ET parameter is reset as soon as the signal state of the IN parameter changes to "0".

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate on-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Signal state that is delayed by the time PT.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current timer value

You can select the data type of the instruction from the "???" drop-down list.

Pulse timing diagram

The following figure shows the behavior of the "Generate on-delay" instruction after the start:

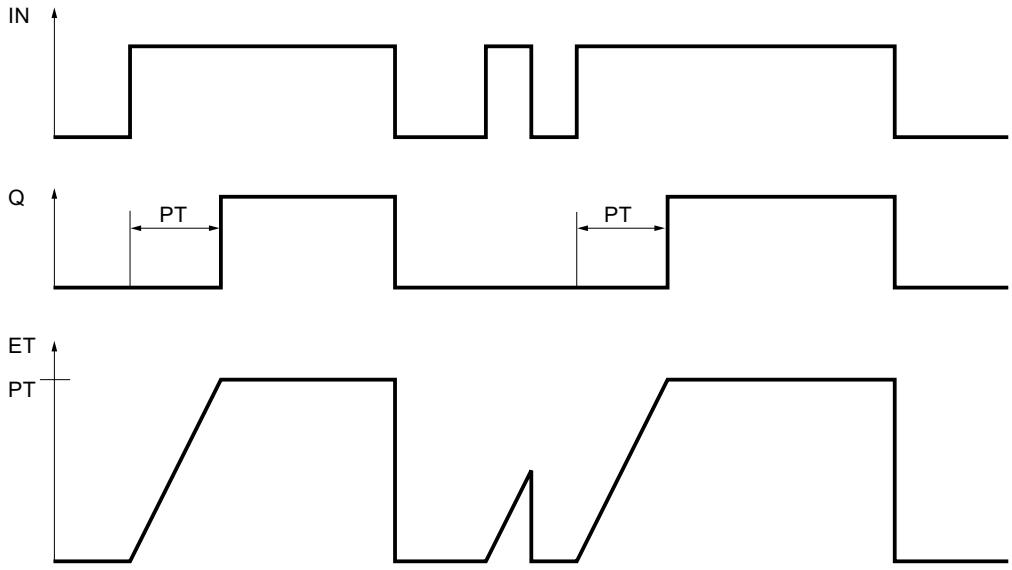

Example

The following example shows how the instruction works:

GRAPH

```
CALL TON TIME, "IEC_TON_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. When the time expires, the "Tag_Status" operand is set to the signal state "1". The "Tag_Status" operand remains set to "1" as long as the "Tag_Start" operand has signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand. When the signal state of the "Tag_Start" operand changes from "1" to "0", the "Tag_Status" operand is reset.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

TOF: Generate off-delay (S7-1500)

Description

You use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed time PT. The Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state at the IN parameter changes back to "0", the programmed time PT starts. The Q parameter remains set as long as time PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before time PT expires, the timer is reset. The signal state of the Q parameter remains set to "1".

The current time value can be queried in the ET parameter. The timer value starts at T#0s and ends when the value of time PT is reached. When the time PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1". If the IN input changes to "1" before time PT has expired, the ET output is reset to the value T#0s.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate off-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.

Parameter	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L, P	Signal state that is delayed by the time PT.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current timer value

You can select the data type of the instruction from the "???" drop-down list.

Pulse timing diagram

The following figure shows the behavior of the "Generate off-delay" instruction after the start:

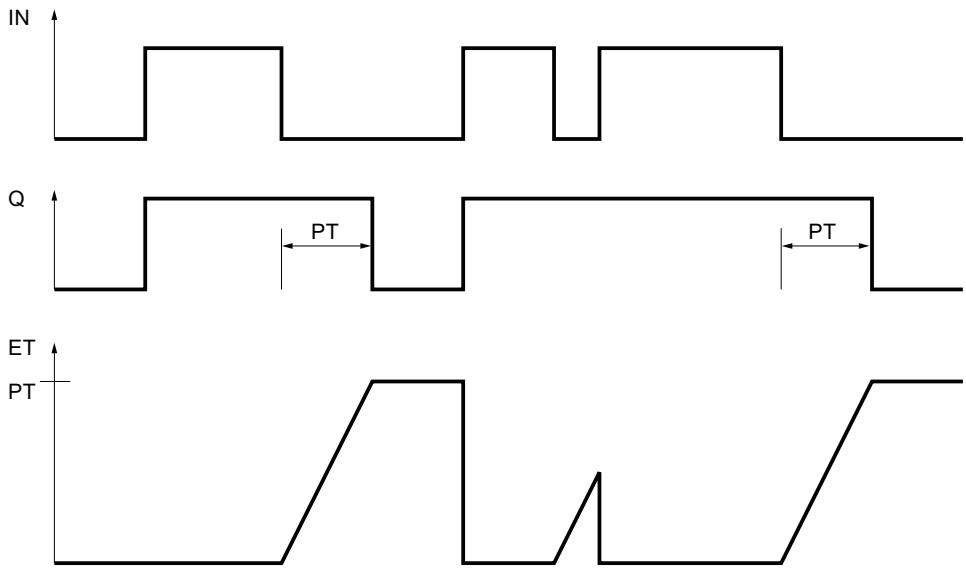

Example

The following example shows how the instruction works:

GRAPH

```
CALL TOF TIME, "IEC_TOF_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Tag_Status" operand is set. When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set. When the time has expired, the "Tag_Status" operand is reset. The current time value is stored in the "Tag_ElapsedTIME" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

TONR: Time accumulator (S7-1500)

Description

The instruction "Time accumulator" is used to accumulate time values within a period set by the parameter PT. When the signal state at the IN input changes from "0" to "1" (positive signal edge), the instruction is executed and time PT starts. While the time PT is running, the time values are accumulated that are recorded when the IN input has signal state "1". The accumulated time is written to the ET output and can be queried there. When the duration PT expires, the output Q has signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes from "1" to "0" (negative signal edge).

The R input resets the ET and Q outputs regardless of the signal state of the start input.

Each call of the "Time accumulator" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated in the following cases:

- When the instruction is called if the ET or Q outputs are interconnected. If the outputs are not interconnected, the current time value at output ET is not updated.
- At an access to the Q or ET outputs.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Maximum duration of time recording The value of the PT parameter must be positive.

Parameters	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L, P	Output that is set when time PT expires.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Accumulated time

You can select the data type of the instruction from the "???" drop-down list.

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:

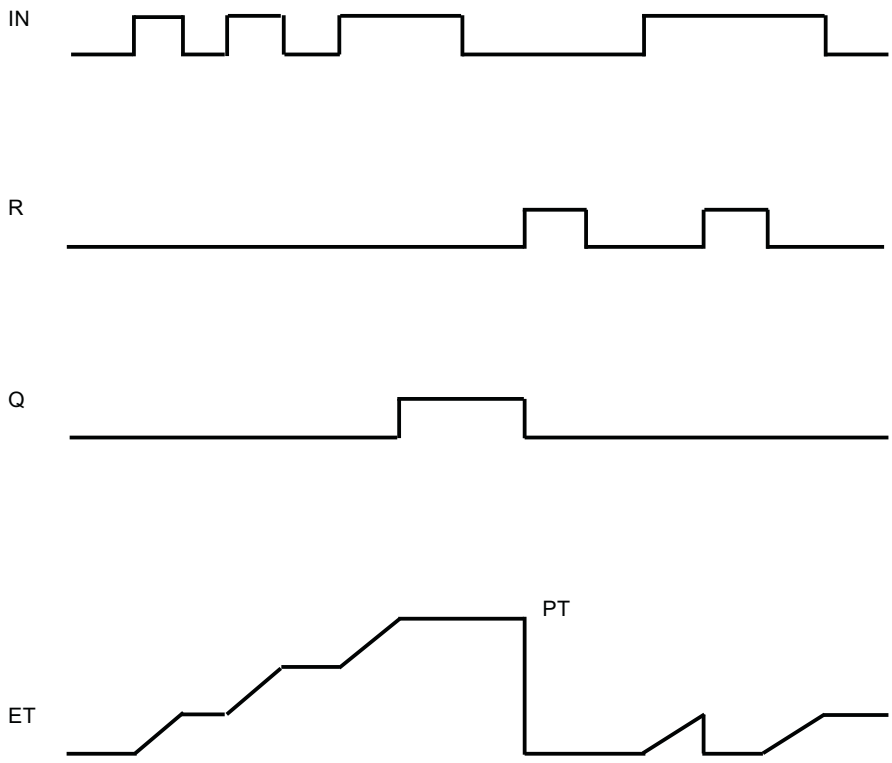

Example

The following example shows how the instruction works:

GRAPH

```
CALL TONR TIME, "IEC_TONR_DB"
      (IN := "Tag_Start"
       R := "Tag_Reset"
       PT := "Tag_PresetTIME"
       Q => "Tag_Status"
       ET => "Tag_ElapsedTIME"
       )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. While the time is running, the time values are accumulated

that are recorded when the "Tag_Start" operand has signal state "1". The accumulated time is stored in the "Tag_ElapsedTIME" operand. When the time value specified for the PT parameter is reached, the "Tag_Status" operand is set to the signal state "1". The current time value is stored in the "Tag_ElapsedTIME" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

Counter operations (S7-1500)

CTU: Count up (S7-1500)

Description

You can use the "Count up" instruction to increment the value at the CV parameter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the parameter CV is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the CV output. When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current counter value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the signal state of the R parameter is "1", the signal state of the CU parameter has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

Data block of system data type IEC_<Counter> (Shared DB)

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

Local tag

- CTU_SINT / CTU_USINT
- CTU_INT / CTU_UINT
- CTU_DINT / CTU_UDINT
- CTU_LINT / CTU_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data_type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTU_COUNTER)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list.

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTU INT, "IEC_CTU_DB"
      (CU := "Tag_Start"
       R := "Tag_ResetCOUNTER"
       PV := "Tag_PresetValue"
       Q => "Tag_Status"
       CV => "Tag_CounterValue"
       )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until it reaches the high limit of the (INT =32767) data type.

The value of the PV parameter is adopted as the limit for determining the "Tag_Status" output. The "Tag_Status" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" operand. In all other cases, the "Tag_Status" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Setting retentivity in an instance data block (Page 7037)

Setting the retentivity of local tags (Page 6971)

Basic information on GRAPH (Page 7369)

CTD: Count down (S7-1500)

Description

The "Count down" instruction is used to decrement the value at the parameter CV. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the signal state of the LD parameter is "1", the signal state of the CD parameter has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

Data block of system data type IEC_<Counter> (Shared DB)

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

Local tag

- CTD_SINT / CTD_USINT
- CTD_INT / CTD_UINT
- CTD_DINT / CTD_UDINT
- CTD_LINT / CTD_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data_type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTD_COUNTER)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L or constant	Count input
LD	Input	BOOL	I, Q, M, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list.

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTD INT, "IEC_CTD_DB"
     (CD := "Tag_Start"
      LD := "Tag_LoadPV"
      PV := "Tag_PresetValue"
      Q => "Tag_Status"
      CV => "Tag_CounterValue"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count down" instruction is executed and the value of the "Tag_CounterValue" output is decremented by one. With each additional positive signal edge, the counter value is decremented until the low limit of the specified data type (INT = -32768) is reached.

The "Tag_Status" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_Status" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Setting retentivity in an instance data block (Page 7037)

Setting the retentivity of local tags (Page 6971)

Basic information on GRAPH (Page 7369)

CTUD: Count up and down (S7-1500)

Description

You can use the "Count up and down" instruction to increment and decrement the value of the CV parameter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in a program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV parameter. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state of the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has signal state "1", the signal state of the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0".

You can query the status of the down counter at the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

Data block of system data type IEC_<Counter> (Shared DB)

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

Local tag

- CTUD_SINT / CTUD_USINT
- CTUD_INT / CTUD_UINT

- CTUD_DINT / CTUD_UDINT
- CTUD_LINT / CTUD_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data_type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTUD_COUNTER)

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multi-instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the QU output is set / value to which the CV output is set when LD = 1.
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	Integers, CHAR, WCHAR, DATE	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list.

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTUD INT, "IEC_CTUD_DB"
      (CU := "Tag_StartCTU"
       CD := "Tag_StartCTD"
       R := "Tag_ResetCOUNTER"
       LD := "Tag_LoadPV"
       PV := "Tag_PresetValue"
       QU => "Tag_CounterStatusUP"
       QD => "Tag_CounterStatusDOWN"
       CV => "Tag_CounterValue"
       )
```

When the signal state of the "Tag_StartCTU" or "Tag_StartCTD" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "Tag_StartCTU" input, the current counter value is incremented by one and stored in the "Tag_CounterValue" output. When there is a positive signal edge at the "Tag_StartCTD" input, the counter value is decremented by one and stored in the "Tag_CounterValue" output. When there is a positive signal edge at the CU input, the counter value is incremented until it reaches the high limit of 32767. When there is a positive signal edge at the CD input, the counter value is decremented until it reaches the low limit of INT = -32768.

The "Tag_CounterStatusUP" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" input. In all other cases, the "Tag_CounterStatusUP" output has signal state "0".

The "Tag_CounterStatusDOWN" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_CounterStatusDOWN" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Setting retentivity in an instance data block (Page 7037)

Setting the retentivity of local tags (Page 6971)

Basic information on GRAPH (Page 7369)

Math functions (S7-1500)

NEG: Create twos complement (S7-1500)

Description

You use the "Create twos complement" instruction to change the sign of the operand value. For example, if the value is positive, the negative equivalent of this value is output.

The value of the OUT output is invalid if the result of the instruction is outside the range permitted for the data type specified at the OUT output (only valid for integers).

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := NEG_REAL("Tag_InValue")
```

The instruction changes the sign of the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

ABS: Form absolute value (S7-1500)

Description

You use the "Form absolute value" instruction to calculate the absolute value of the operand value.

The value of the result is invalid if a floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ABS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	-6.234
Tag_OutValue	6.234

The instruction calculates the absolute value of the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

MIN: Get minimum (S7-1500)

Description

You use the "Get minimum" instruction to compare the values of the available IN1 and IN2 inputs and write the lowest value in the OUT output.

The value of the OUT output is invalid if one of the following conditions is met:

- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MIN INT
      (IN1 := "TagIn_Value1"
       IN2 := "TagIn_Value2"
       OUT => "Tag_Minimum"
       )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
OUT	Tag_Minimum	12222

The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to the "Tag_Minimum" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

MAX: Get maximum (S7-1500)

Description

You use the "Get maximum" instruction to compare the values of the IN1 and IN2 inputs and write the highest value to the OUT output.

The value of the OUT output is invalid if one of the following conditions is met:

- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MAX INT
      (IN1 := "TagIn_Value1"
       IN2 := "TagIn_Value2"
       OUT => "Tag_Maximum"
       )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
OUT	Tag_Maximum	14444

The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to the "Tag_Maximum" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

LIMIT: Set limit value (S7-1500)

Description

You use the "Set limit value" instruction to limit the value of the IN input to the values of the MN and MX inputs. If the value of the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not met and the IN input value is below the low limit MN, the OUT output is set to the value of the MN input. If the high limit MX is exceeded, the OUT output is set to the value of the MX input.

If the value at the MN input is greater than at the MX input, the result is the value specified at the IN parameter. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is met:

- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value of the MN parameter is greater than the value of the MX parameter.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
MN	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Low limit
IN	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Input value
MX	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	High limit
OUT	Output	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL LIMIT INT
      (MN := "Tag_LowLimit"
       IN := "Tag_InputValue"
       MX := "Tag_HighLimit"
       OUT => "Tag_Result"
       )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_LowLimit	12000
IN	Tag_InputValue	8000
MX	Tag_HighLimit	16000
OUT	Tag_Result	12000

The value of the "Tag_InputValue" operand is compared with the values of the "Tag_LowLimit" and "Tag_HighLimit" operands. Since the value of the "Tag_InputValue" operand is less than the low limit, the value of the "Tag_LowLimit" operand is copied to the "Tag_Result" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

SQR: Form square (S7-1500)

Description

The "Form square" instruction is used to square the value of the floating-point number of the operand and write the result to the output.

If the operand contains an invalid floating-point number, the value of the result is invalid.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Square of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SQR("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	5.0
Tag_OutValue	25.0

The instruction squares the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

SQRT: Form square root (S7-1500)

Description

Use the "Form square root" instruction to calculate the square root of the input value and save the result in the specified operand. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number. If the value of the operand is "0", then the result is also "0".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L	Square root of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SQRT("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	25.0
Tag_OutValue	5.0

The instruction calculates the square root of the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Invalid floating-point numbers (Page 253)

Basic information on GRAPH (Page 7369)

LN: Form natural logarithm (S7-1500)

Description

You use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e=2.718282) from the input value. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := LN("Tag_InValue")
```

The instruction calculates the natural logarithm of the value in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Invalid floating-point numbers (Page 253)

Basic information on GRAPH (Page 7369)

EXP: Form exponential value (S7-1500)

Description

You use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2,718282) and the value of the operand.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Exponential value of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := EXP("Tag_InValue")
```

The instruction calculates the exponent from base e and the value of the "Tag_InValue" operand and outputs the result (e^{IN}) in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Invalid floating-point numbers (Page 253)

Basic information on GRAPH (Page 7369)

SIN: Form sine value (S7-1500)

Description

You use the "Form sine value" instruction to calculate the sine of an angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the value of the operand is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Sine of the specified angle

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SIN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+1.570796 (π/2)
Tag_OutValue	1.0

The instruction calculates the sine of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

COS: Form cosine value (S7-1500)

Description

You use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the value of the operand is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Cosine of the specified angle

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := COS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+1.570796 (π/2)
Tag_OutValue	0

The instruction calculates the cosine of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

TAN: Form tangent value (S7-1500)

Description

You use the "Form tangent value" instruction to calculate the tangent of an angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the value of the operand is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Tangent of the specified angle

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := TAN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+3.141593 (π)
Tag_OutValue	0

The instruction calculates the tangent of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

ASIN: Form arcsine value (S7-1500)

Description

You use the "Form arcsine value" instruction to calculate the size of the angle that corresponds to the sine value specified in the operand. Only valid floating-point numbers within the range -1 to +1 can be specified for the operand. The calculated angle size can range in value from -π/2 to +π/2.

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Sine value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ASIN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1.0
Tag_OutValue	+1.570796 (π/2)

The instruction calculates the size of the angle corresponding to the sine value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

ACOS: Form arccosine value (S7-1500)

Description

You use the "Form arccosine value" instruction to calculate the size of the angle that corresponds to the cosine value specified in the operand. Only valid floating-point numbers within the range -1 to +1 can be specified for the operand. The calculated angle size can range in value from 0 to +π.

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Cosine value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ACOS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0
Tag_OutValue	+1.570796 (π/2)

The instruction calculates the size of the angle corresponding to the cosine value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

ATAN: Form arctangent value (S7-1500)

Description

You use the "Form arctangent value" instruction to calculate the size of the angle that corresponds to the tangent value specified in the operand. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) at the operand. The calculated angle size can range in value from -π/2 to +π/2.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Tangent value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ATAN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1.0
Tag_OutValue	+0.785398 (π/4)

The instruction calculates the size of the angle corresponding to the tangent value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Invalid floating-point numbers (Page 253)

Basic information on GRAPH (Page 7369)

FRAC: Return fraction (S7-1500)

Description

You use the "Return fraction" instruction to determine the decimal places of the value in the operand. For example, if the operand contains the value 123.4567, the result returns the value 0.4567.

The value of the result is invalid if the value of the operand is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value, whose decimal places are determined
RET_VAL	Output	Floating-point numbers	I, Q, M, D, L, P	Decimal places of the input value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL FRAC REAL
      (IN := "Tag_InValue"
       RET_VAL => "Tag_OutValue"
       )
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	2.555
Tag_OutValue	0.555

The instruction copies the decimal places from the value of the "Tag_InValue" operand to the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

Move operations (S7-1500)

Deserialize: Deserialize (S7-1500)

Description

You can use the "Deserialize" instruction to convert back the sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type> and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type> is located must have the ARRAY of BYTE or ARRAY of CHAR data type and be declared with standard access in version 1.0. Optimized memory areas also permitted as of version 2.0. The capacity of the standard memory area is 64 KB. Make sure that there is enough memory space prior to the conversion. If the memory area is filled using the instruction "Serialize" and it is possible that fill bytes were inserted, their conversion is not taken into account.

It is recommended to define the low limit of the ARRAY with "0", because then the index within ARRAY corresponds to the value of the POS parameter, e.g. ARRAY[0] = POS 0. The description and example below have been formed based on this.

The instruction enables you to convert multiple sequential representations of converted data back to their original state step-by-step.

If you only want to convert back a single sequential representation of a PLC data type (UDT), STRUCT or ARRAY of <data type>, you can also directly use the instruction "TRCV: Receive data via communication connection".

Size of the memory area

The alignment rules mean that simple structures in the optimized memory area do not contain filling bytes. This results in a structure in the optimized memory area being smaller than in the standard memory area. ARRAYs of structures and structures in which other structures are located contain filling bytes. There is no general rule as to the memory area in which a composed structure requires more space.

Applies to CPUs of the S7-1500 series:

For a block with the block property "Optimized block access", the BOOL has a length of 1 byte. A structure that consists largely of the data type BOOL can therefore be larger in the optimized memory area than in the standard memory area. Composed structures with a low proportion of BOOL data types are smaller in the optimized memory area than in the standard memory area.

Optimized memory area

To deserialize larger structures, you can declare the memory area for sequential representation with optimized access for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. The content of the sequential representation remains unchanged, as for a standard memory area. Only symbolic access to the bytes in the ARRAY is possible.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC_ARRAY	Input	VARIANT	D, block interface of an FB (the sections Input, Output, Static and Temp are possible.)	Pointer to an ARRAY of BYTE or ARRAY of CHAR in which the data stream is saved.
DEST_VARIABLE	InOut	VARIANT	D, block interface of an FB	Pointer to a tag of the STRUCT, ARRAY or PLC data type (UDT) to be deserialized.
POS	InOut	DINT	I, Q, M, D, L	The operand at the POS parameter stores the index of the first byte based on the number of bytes occupied by the converted customer data. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The tag at the SRC_ARRAY parameter is not in a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains a ZERO pointer.
8151	No valid reference at the SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8154	Invalid data type at the SRC_ARRAY parameter
8250	A NULL pointer was transferred at parameter DEST_ARRAY.
8251	No valid reference at the DEST_VARIABLE parameter
8382	The value at the POS parameter is outside the limits of the array.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

For a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0, the following error code has a different meaning:

Error code* (W#16#...)	Explanation
8136	Access to the memory area at the SRC_ARRAY parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL Deserialize VARIANT
     (SRC_ARRAY := "Buffer".Field
      DEST_VARIABLE => "Target".Client
      POS := #BufferPos
      RET_VAL => #Error
      )
```

The instruction deserializes the sequential representation of the customer data from the "Buffer" tag and writes it to the "Target" tag. The #BufferPos operand stores the index of the first byte based on the number of bytes occupied by the converted customer data.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on VARIANT (Page 324)

Basics of PLC data types (UDT) (Page 273)

Structure of an ARRAY tag (Page 287)

Structure of a STRUCT tag (Page 280)

Structure of a STRING tag (Page 266)

Basic information on GRAPH (Page 7369)

Serialize: Serialize (S7-1500)

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT), STRUCT or ARRAY of <data type> to a sequential representation without losing parts of their structure.

You use the instruction to temporarily save multiple structured data items from your program in a buffer, which should preferably be in a global data block, and send them to another CPU. The memory area in which the converted data is stored must have the ARRAY of BYTE or ARRAY of CHAR data type and be declared with standard access in version 1.0. Optimized data is also permitted as of version 2.0. Fill data of the source data area is undefined in the target array. These can be fill bytes or fill bits of a data area (e.g. ARRAY, STRUCT or PLC data type (UDT)) as well as the characters of a string currently not in use.

The capacity of the standard memory area is 64 KB. Structures that are larger than 64 KB in accordance with the rules for standard memory areas cannot be serialized if the operand at the DEST_ARRAY parameter is located in a standard memory area.

It is recommended to define the low limit of the ARRAY with "0", because then the index within ARRAY corresponds to the value of the POS parameter, e.g. ARRAY[0] = POS 0. The description and example below have been formed based on this.

The operand at the POS parameter contains information about the number of bytes used by the converted data.

If you want to send a single PLC data type (UDT), STRUCT or ARRAY of <data type>, you can directly call the instruction "TSEND: Send data via communication connection".

Size of the memory area

The alignment rules mean that simple structures in the optimized memory area do not contain filling bytes. This results in a structure in the optimized memory area being smaller than in the standard memory area. ARRAYs of structures and structures in which other structures are located contain filling bytes. There is no general rule as to the memory area in which a composed structure requires more space.

Applies to CPUs of the S7-1500 series:

For a block with the block property "Optimized block access", the length of the data type BOOL depends on the data type that follows. This means if a BYTE follows, for example, the data type BOOL has a length of 1 byte. When a WORD follows, for example, the data type BOOL has a length of 2 bytes. A structure that consists largely of the data type BOOL can therefore be larger in the optimized memory area than in the standard memory area. Composed structures with a low proportion of BOOL data types are smaller in the optimized memory area than in the standard memory area.

We therefore recommend that the source data area for the serialization starts with the large data types and ends with Boolean elements. This greatly reduces the filling with filling bits.

Optimized memory area

To serialize larger structures, you can declare the memory area with optimized access for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. The sequential representation remains unchanged, as for a standard memory area.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
SRC_VARIABLE	Input	VARIANT	D, block interface of an FB	Pointer to a tag of the STRUCT, ARRAY or PLC data type (UDT) to be serialized.
DEST_ARRAY	InOut	VARIANT	D, block interface of an FB (the sections Input, Output, Static and Temp are possible.)	Pointer to an ARRAY of BYTE or ARRAY of CHAR in which the generated data stream is saved.
POS	InOut	DINT	I, Q, M, D, L	The operand at the POS parameter stores the index of the first byte based on the total number of bytes that the converted customer data has occupied. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

You can find additional information on valid data types under "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains a ZERO pointer.
8151	No valid reference at the SRC_VARIABLE parameter
8236	The tag at the SRC_ARRAY parameter is not in a block with standard access.
8250	A NULL pointer was transferred at parameter DEST_ARRAY.
8251	No valid reference at the DEST_ARRAY parameter
8253	The tag at the parameter DEST_ARRAY does not provide enough memory space for the content of the tag at the parameter SRC_VARIABLE. Due to the input value of the tag at parameter POS the available memory space is reduced. The input value of the parameter POS decides where to begin within the tag at the parameter DEST_ARRAY.
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at the POS parameter is outside the limits of the array.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

For a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0, the following error code has a different meaning:

Error code* (W#16#...)	Explanation
8236	Access to the memory area at the DEST_ARRAY parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL Serialize VARIANT
     (SRC_VARIABLE := "Source".Client
      DEST_ARRAY => "Buffer".Field
      POS := #BufferPos
      RET_VAL => #Error
      )
```

The instruction serializes the customer data from the "Source" tag and writes the sequential representation to the "Buffer" tag. The index of the next unwritten byte of the "Buffer".Field operand is saved in the #BufferPos operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on VARIANT (Page 324)

Basics of PLC data types (UDT) (Page 273)

Structure of an ARRAY tag (Page 287)

Structure of a STRUCT tag (Page 280)

Structure of a STRING tag (Page 266)

Basic information on GRAPH (Page 7369)

MOVE_BLK: Move block (S7-1500)

Description

You use the "Move block" instruction to move the content of a memory area (source range) to another memory area (target range). The number of elements to be moved to the target range is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at the IN parameter.

The instruction can only be executed if the source range and the target range have the same data type.

The value at the output OUT is invalid if more data is copied than is made available at the input IN or at the output OUT.

Note

Use of ARRAYs

The instruction copies the contents from the defined element n elements (n = depending on the value at the parameter COUNT) from the source range to the target range, starting at the specified index.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be copied from the source range to the target range
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	The first element of the target range to which the contents of the source range are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.				

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MOVE_BLK INT_UINT
     (IN := #a_array[2]
      COUNT := "Tag_Count"
      OUT => #b_array[1]
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the "a_array" operand is ARRAY[0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the "b_array" operand is ARRAY[0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element.

Note

You can find more information on the MOVE_BLK instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

MOVE_BLK_VARIANT: Move block (S7-1500)

Description

You use the "Move block" instruction to move the content of a memory area (source range) to another memory area (target range). You can copy a complete ARRAY or elements of an ARRAY to another ARRAY of the same data type. The size (number of elements) of source and destination ARRAY may be different. You can copy multiple or single elements within an ARRAY.

The number of elements that are to be copied must not exceed the selected source range or target range.

If you use the instruction at the time the block is being created, the ARRAY does not yet have to be known, as the source and the target are transferred per VARIANT.

Counting at the parameters SRC_INDEX and DEST_INDEX always begins with the low limit "0", regardless of the later declaration of the ARRAY.

The instruction is not executed if more data is copied than is made available.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC	Input [2)]	VARIANT (which points to an ARRAY or an individual ARRAY element), ARRAY of <Data_type>	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L or constant	Number of elements which are copied Assign the value "1" to the parameter COUNT, if no ARRAY is specified at parameter SRC or at parameter DEST.
SRC_INDEX	Input	DINT	I, Q, M, D, L or constant	Defines the first element to be copied: • The SRC_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter SRC, the integer at parameter SRC_INDEX specifies the first element within the source area from which it is to be copied. Independent of the declared ARRAY limits. • If no Array is specified at parameter SRC or only one single element of an ARRAY is specified, then assign the value "0" at parameter SRC_INDEX.

Parameter	Declaration	Data type	Memory area	Description
DEST_INDEX	Input	DINT	I, Q, M, D, L or constant	Defines the start of the destination memory area: • The DEST_INDEX parameter is calculated zero-based. If an ARRAY is specified at parameter DEST, the integer at parameter DEST_INDEX specifies the first element within the target range that is to be copied into. Independent of the declared ARRAY limits. • If no ARRAY is specified at parameter DEST, then assign the value "0" at parameter DEST_INDEX.
DEST	Output [1)]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Destination area into which the contents of the source block are copied.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
1) The DEST parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface. 2) With the SRC parameter, the data types BOOL and ARRAY of BOOL are not permitted.				

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The data types do not correspond. Instead of an ARRAY of Struct, use an ARRAY of PLC data type (UDT).
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at the SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value

Error code* (W#16#...)	Explanation
8382	The value at the SRC_INDEX parameter is less than zero.
8383	The value at the SRC_INDEX is outside the high limit of the ARRAY.
8482	The value at the DEST_INDEX parameter is less than zero.
8483	The value at the DEST_INDEX is outside the high limit of the ARRAY.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at the DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL MOVE_BLK_VARIANT
     (SRC := #SrcField
      COUNT := "Tag_Count"
      SRC_INDEX := "Tag_Src_Index"
      DEST_INDEX := "Tag_Dest_Index"
      DEST => #DestField
      RET_VAL := "Tag_Result"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Declaration in the block interface	Operand	Value
SRC	Input	#SrcField	The local operand #SrcField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[0..10] of "MOVE_UDT")
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3

Parameter	Declaration in the block interface	Operand	Value
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	The local operand #DestField uses a PLC data type that was still unknown at the time when the block was programmed. (ARRAY[10..20] of "MOVE_UDT")

Two elements are moved from the source range, starting from the fourth element of the ARRAY [0..10] of MOVE_UDT, to the target range. The copies are pasted in the ARRAY [10..20] of MOVE_UDT starting from the fourth element.

Note

You can find more information on the MOVE_BLK_VARIANT instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Example of moving data (Page 333)

Basic information on VARIANT (Page 324)

Basic information on GRAPH (Page 7369)

UMOVE_BLK: Move block uninterruptible (S7-1500)

Description

You use the "Move block uninterruptible" instruction to move the content of a memory area (source range) to another memory area (target range). The instruction cannot be interrupted. The number of elements to be moved to the target range is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at the IN parameter.

The instruction can only be executed if the source range and the target range have the same data type.

Note

Move uninterruptible

The move operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

The value at the output OUT is invalid if more data is copied than is made available at the input IN or at the output OUT.

You use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be copied from the source range to the target range
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	The first element of the target range to which the contents of the source range are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.				

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL UMOVE_BLK INT_UINT
      (IN := #a_array[2]
       COUNT := "Tag_Count"
       OUT => #b_array[1]
```

GRAPH
)

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the "a_array" operand is ARRAY[0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the "b_array" operand is ARRAY[0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The move operation cannot be interrupted by other operating system activities.

Note

You can find more information on the UMOVE_BLK instruction in the following article in the Siemens Industry Online Support:

In STEP 7 (TIA Portal) how do you copy memory areas and structured data from one data block to another?

https://support.industry.siemens.com/cs/ww/en/view/42603881 (https://support.industry.siemens.com/cs/ww/en/view/42603881)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

FILL_BLK: Fill block (S7-1500)

Description

You can use the "Fill block" instruction to fill a memory area (target range) with the value of the IN input. The target range is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is moved to the target range as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source range and the target range have the same data type.

The maximum number of elements changed is the number of elements in the ARRAY or structure. If you copy more data than there are elements at the OUT output, you will get an unintended result.

Note

Use of ARRAYs

The instruction reads the content from the selected element from the source range and copies this content n-times (n = depending on the value at the parameter COUNT) to the destination, starting at the specified index.

Filling structures

As well as the elements of an ARRAY, you can also fill multiple elements of a structure (STRUCT, PLC data type) with the same value. The structure whose elements you want to fill must only contain individual elements of the same elementary data type. The structure itself can, however, be embedded in another structure.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L, P or constant	Element used to fill the target range
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of repeated move operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	Address in target range from which filling starts

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example with an ARRAY

The following example shows how the instruction works when you want to fill an ARRAY:

GRAPH

```
CALL FILL_BLK INT_UINT
     (IN := #FillValue
      COUNT := "Tag_Count"
```

GRAPH

```
OUT => #TargetArea[1]
)
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	FillValue	The data type of the operand is INT.
COUNT	Tag_Count	3
OUT	TargetArea	The data type of the operand TargetArea is ARRAY[1..5] of INT. It consists of five elements of data type INT.

The instruction copies the value of the operand #FillValue to the #TargetArea output tag three times, starting with the first element.

Examples with a structure

The following examples show how the instruction works when you want to fill a structure:

Create a global data block with the following elements:

Data_block_1					Data type
MyStruct1					STRUCT
	Member_1				INT
	Member_2				INT
	Member_3				INT
	Member_4				INT
MyStruct2					STRUCT
	SubArray				ARRAY[1..2] of STRUCT
		SubArray[1]			STRUCT
			NestedStruct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT
		SubArray[2]			STRUCT
			Nested-Struct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT

Generate the following program code to address the MyStruct1 tag:

GRAPH

```
CALL FILL_BLK INT_UINT
     (IN := 10
      COUNT := 2
      OUT => "Data_block_1".MyStruct1.Member_2
      )
```

Generate the following program code to address the MyStruct2 tag:

GRAPH

```
CALL FILL_BLK INT_UINT
     (IN := 10
      COUNT := 2
      OUT => "Data_block_1".MyStruct2.SubArray[1].NestedStruct.Member_2
      )
```

In each of the two examples, the value 10 of the IN parameter is copied twice to the operand at the OUT parameter, starting with the element Member_2. This means that the value 10 is copied to the elements Member_2 and Member_3. The other two elements, Member_1 and Member_4, are not changed.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

UFILL_BLK: Fill block uninterruptible (S7-1500)

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (target range) with the value of the IN input. The instruction cannot be interrupted. The target range is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is moved to the target range as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source range and the target range have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The maximum number of elements changed is the number of elements in the ARRAY or structure. If you copy more data than there are elements at the OUT output, you will get an unintended result.

Note

Use of ARRAYs

The instruction reads the content from the selected element from the source range and copies this content n-times (n = depending on the value at the parameter COUNT) to the destination, starting at the specified index.

You use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Filling structures

As well as the elements of an ARRAY, you can also fill multiple elements of a structure (STRUCT, PLC data type) with the same value. The structure whose elements you want to fill must only contain individual elements of the same elementary data type. The structure itself can, however, be embedded in another structure.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	I, Q, M, D, L, P or constant	Element used to fill the target range
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of repeated move operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR, WCHAR	D, L	Address in target range from which filling starts

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example with an ARRAY

The following example shows how the instruction works when you want to fill an ARRAY:

GRAPH

```
CALL UFILL_BLK INT_UINT
```

GRAPH

```
(IN := #FillValue
 COUNT := "Tag_Count"
 OUT => #TargetArea[1]
 )
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	FillValue	The data type of the operand is INT.
COUNT	Tag_Count	3
OUT	TargetArea	The data type of the operand TargetArea is ARRAY[1..5] of INT. It consists of five elements of data type INT.

The instruction copies the value of the operand #FillValue to the #TargetArea output tag three times, starting with the first element. The move operation cannot be interrupted by other operating system activities.

Examples with a structure

The following examples show how the instruction works when you want to fill a structure:

Create a global data block with the following elements:

Data_block_1					Data type
MyStruct1					STRUCT
	Member_1				INT
	Member_2				INT
	Member_3				INT
	Member_4				INT
MyStruct2					STRUCT
	SubArray				ARRAY[1..2] of STRUCT
		SubArray[1]			STRUCT
			NestedStruct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT
		SubArray[2]			STRUCT
			Nested-Struct		STRUCT
				Member_1	INT
				Member_2	INT
				Member_3	INT
				Member_4	INT

Generate the following program code to address the MyStruct1 tag:

GRAPH

```
CALL UFILL_BLK INT_UINT
    (IN := 10
     COUNT := 2
     OUT => "Data_block_1".MyStruct1.Member_2
      )
```

Generate the following program code to address the MyStruct2 tag:

GRAPH

```
CALL UFILL_BLK INT_UINT
     (IN := 10
      COUNT := 2
      OUT => "Data_block_1".MyStruct2.SubArray[1].NestedStruct.Member_2
      )
```

In each of the two examples, the value 10 of the IN parameter is copied twice to the operand at the OUT parameter, starting with the element Member_2. This means that the value 10 is copied to the elements Member_2 and Member_3. The other two elements, Member_1 and Member_4, are not changed.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

SCATTER: Parse the bit sequence into individual bits (S7-1500)

Description

The "Parse the bit sequence into individual bits" instruction parses a tag of the BYTE, WORD, DWORD or LWORD data type into individual bits and saves them in an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements.

Note

Multi-dimensional ARRAY of BOOL

With the "Parse the bit sequence into individual bits" instruction, the use of a multidimensional ARRAY of BOOL is not permitted.

Note

Length of the ARRAY, STRUCT or PLC data type

The ARRAY, the anonymous STRUCT or the PLC data type must have exactly the number of elements that is specified by the bit sequence.

This means for the data type BYTE, for example, the ARRAY, STRUCT or the PLC data type must have exactly 8 elements (WORD = 16, DWORD = 32 and LWORD = 64).

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1500 series as of firmware version 2.1.

This way you can, for example, parse a status word, and read and change the status of the individual bits using the index. Using GATHER, you can merge the bits once again into a bit sequence.

The ENO enable output returns the signal state "0" if the ARRAY does not provide enough BOOL elements.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BYTE, WORD, DWORD, LWORD	I, Q, M, D, L	Bit sequence that is parsed. The values must not be located in the I/O area or in the DB of a technology object.
OUT	Output	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	I, Q, M, D, L	ARRAY, STRUCT or PLC data type in which the individual bits are stored

You can find additional information on valid data types under "See also".

Example with an ARRAY

Create the following tags in the block interface:

Tag	Section	Data type
SourceWord	Input	WORD
DestinationArray	Output	ARRAY[0..15] of BOOL

The following example shows how the instruction works:

GRAPH

```
CALL SCATTER
    (IN := #SourceWord
     OUT => #DestinationArray
     )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceWord	WORD (16 bits)
OUT	DestinationArray	The operand "DestinationArray" is of the data type ARRAY[0..15] of BOOL. It consists of 16 elements and is therefore just as large as the WORD that is to be parsed.

The #SourceWord operand of the WORD data type is parsed into its individual bits (16) and assigned to the individual elements of the #DestinationArray operand.

Example with a PLC data type (UDT)

Create the following PLC data type "myBits":

myBits

	Name	Data type
1	GeneralError	Bool
2	DeviceError	Bool
3	CommError	Bool
4	myError1	Bool
5	myError2	Bool
6	myError3	Bool
7	myError4	Bool
8	myError5	Bool
9	myError6	Bool
10	myError7	Bool
11	myError8	Bool
12	myError9	Bool
13	myError10	Bool
14	myError11	Bool
15	myError12	Bool
16	myError13	Bool

Create the following tags in the block interface:

Tag	Section	Data type
SourceWord	Input	WORD
DestinationUDT	Output	"myBits"

The following example shows how the instruction works:

GRAPH

```
CALL SCATTER
     (IN := #SourceWord
      OUT => #DestinationUDT
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceWord	WORD (16 bits)
OUT	DestinationUDT	The "DestinationUDT" operand has the PLC data type (UDT). It consists of 16 elements and is therefore just as large as the WORD that is to be parsed.

The #SourceWord operand of the WORD data type is parsed into its individual bits (16) and assigned to the individual elements of the #DestinationUDT operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

SCATTER_BLK: Parse elements of an ARRAY of bit sequence into individual bits (S7-1500)

Description

The "Parse elements of an ARRAY of bit sequence into individual bits" instruction parses one or more elements of an ARRAY of BYTE, WORD, DWORD or LWORD into individual bits and saves them in an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements. At the COUNT_IN parameter you specify how many elements of the source ARRAY are going to be parsed. The source ARRAY at the IN parameter may have more elements than specified at the COUNT_IN parameter. The ARRAY of BOOL, the anonymous STRUCT or the PLC data type must have sufficient elements to save the bits of the parsed bit sequences. However, the destination memory area may also be larger.

Note

Multi-dimensional ARRAY of BOOL

If the ARRAY is a multi-dimensional ARRAY of BOOL, the filling bits of the dimensions contained are counted as well even if they were not explicitly declared.

Example 1: An ARRAY[1..10,0..4,1..2] of BOOL is handled like an ARRAY[1..10,0..4,1..8] of BOOL or like an ARRAY[0..399] of BOOL.

Example 2: At the IN parameter, an ARRAY[0..5] of WORD (sourceArrayWord[2]) is interconnected. The COUNT_IN parameter has the value "3". At the OUT parameter, an ARRAY[0..1,0..5,0..7] of BOOL (destinationArrayBool[0,0,0]) is interconnected. Both the array at the IN parameter and at the OUT parameter has a size of 96 bits. The ARRAY of WORD is parsed into 48 individual bits.

Note

If the ARRAY low limit of the target ARRAY is not "0", note the following:

For performance reasons the index must always start at a BYTE, WORD, DWORD or LWORD limit. This means the index must be calculated starting at the low limit of the ARRAY. The following formula is used as basis for this calculation:

Valid indices = ARRAY low limit + n(number of bit sequences) × number of bits of the desired bit sequence

For an ARRAY[-2..45] of BOOL and the bit sequence WORD the calculation looks as follows:

- Valid index (-2) = -2 + 0 × 16
- Valid index (14) = -2 + 1 × 16
- Valid index (30) = -2 + 2 × 16

You can find an example described below.

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1500 series as of firmware version 2.1.

This way you can, for example, parse status words, and read and change the status of the individual bits using the index. Using GATHER, you can merge the bits once again into a bit sequence.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- The source ARRAY has fewer elements than specified at the COUNT_IN parameter.
- The index of the destination ARRAY does not start at a BYTE, WORD, DWORD or LWORD limit. In this case, no result is written to the ARRAY of BOOL.
- The ARRAY[*] of BOOL, STRUCT or PLC data type does not provide the required number of elements. In this case as many bit sequences as possible are parsed and written to the ARRAY of BOOL, the anonymous STRUCT or the PLC data type. The remaining bit sequences are no longer taken into account.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Element of an ARRAY[*] of <bit sequence>	I, Q, M, D, L	ARRAY of <bit sequence> that is parsed. The values must not be located in the I/O area or in the DB of a technology object.
COUNT_IN	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Counter for the number of elements of the source ARRAY that are going to be parsed. The value must not be in the I/O area or in the database of a technology object.
OUT	Output	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	I, Q, M, D, L	ARRAY, STRUCT or PLC data type in which the individual bits are stored

You can select the required bit sequence from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example of a destination ARRAY with the low limit "0"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayWord	Input	ARRAY[0..5] of WORD
CounterInput		UDINT
DestinationArrayBool	Output	ARRAY[0..95] of BOOL

The following example shows how the instruction works:

GRAPH

```
CALL SCATTER_BLK
     (IN := #SourceArrayWord[2]
      COUNT_IN := #CounterInput
      OUT => #DestinationArrayBool[0]
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayWord[2]	ARRAY[0..5] of WORD (96 bits can be parsed.)
COUNT_IN	CounterInput = 3	UDINT3 (3 WORDs or 48 bits are to be parsed. This means at least 48 bits must be available in the destination ARRAY.)
OUT	DestinationArrayBool[0]	The operand "DestinationArray-Bool" is of the data type AR-RAY[0..95] of BOOL. This means it provides 96 BOOL ele-ments.

The 3rd, 4th and 5th WORD of the #SourceArrayWord operand is parsed into its individual bits (48) and as of the 1st element assigned to the individual elements of the #DestinationArrayBool operand.

Example of a destination ARRAY with the low limit "-2"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayWord	Input	ARRAY[0..5] of WORD
CounterInput		UDINT
DestinationArrayBool	Output	ARRAY[-2..93] of BOOL

The following example shows how the instruction works:

GRAPH

```
CALL SCATTER_BLK
     (IN := #SourceArrayWord[2]
```

GRAPH

```
COUNT_IN := #CounterInput
OUT => #DestinationArrayBool[14]
)
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayWord[2]	ARRAY[0..5] of WORD (96 bits can be parsed.)
COUNT_IN	CounterInput = 3	UDINT3 (3 WORDs or 48 bits are to be parsed. This means at least 48 bits must be available in the destination ARRAY.)
OUT	DestinationArrayBool[14]	The operand "DestinationArray-Bool" is of the data type AR-RAY[-2..93] of BOOL. This means it provides 96 BOOL ele-ments.

The 3rd, 4th and 5th WORD of the #SourceArrayWord operand is parsed into its individual bits (48) and as of the 16th element assigned to the individual elements of the #DestinationArrayBool operand. The remaining 32 bits are not written.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

GATHER: Merge individual bits into a bit sequence (S7-1500)

Description

The "Merge individual bits into a bit sequence" instruction merges the bits from an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements into a bit sequence. The bit sequence is saved in a tag of the data type BYTE, WORD, DWORD or LWORD.

Note

Multi-dimensional ARRAY of BOOL

With the "Merge individual bits into a bit sequence" instruction, the use of a multidimensional ARRAY of BOOL is not permitted.

Note

Length of the ARRAY, STRUCT or PLC data type

The ARRAY, STRUCT or the PLC data type must have exactly the number of elements that is specified by the bit sequence.

This means for the data type BYTE, for example, the ARRAY, anonymous STRUCT or the PLC data type must have exactly 8 elements (WORD = 16, DWORD = 32 and LWORD = 64).

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1500 series as of firmware version 2.1.

The ENO enable output returns the signal state "0", if the ARRAY, anonymous STRUCT or the PLC data type has fewer or more BOOL elements than specified by the bit sequence. The BOOL elements are not transferred then. The ENO also returns "0" if fewer than the necessary number of bits is available.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	ARRAY[*] of BOOL, STRUCT or PLC data type *: 8, 16, 32 or 64 elements	I, Q, M, D, L	ARRAY, STRUCT or PLC data type, the bits of which are merged into a bit sequence. The values must not be located in the I/O area or in the DB of a technology object.
OUT	Output	BYTE, WORD, DWORD, LWORD	I, Q, M, D, L	Merged bit sequence, saved in a tag

You can select the required bit sequence from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example with an ARRAY

Create the following tags in the block interface:

Tag	Section	Data type
SourceArray	Input	ARRAY[0..15] of BOOL
DestinationWord	Output	WORD

The following example shows how the instruction works:

GRAPH

```
CALL GATHER
    (IN := #SourceArray
     OUT => #DestinationWord
    )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArray	The operand "SourceArray" is of the data type ARRAY[0..15] of BOOL. It consists of 16 elements and is therefore just as large as the WORD in which the bits are to be merged.
OUT	DestinationWord	WORD (16 bits)

The bits of the #SourceArray operand are merged into a WORD.

Example with a PLC data type (UDT)

Create the following PLC data type "myBits":

myBits

	Name	Data type
1	GeneralError	Bool
2	DeviceError	Bool
3	CommError	Bool
4	myError1	Bool
5	myError2	Bool
6	myError3	Bool
7	myError4	Bool
8	myError5	Bool
9	myError6	Bool
10	myError7	Bool
11	myError8	Bool
12	myError9	Bool
13	myError10	Bool
14	myError11	Bool
15	myError12	Bool
16	myError13	Bool

Create the following tags in the block interface:

Tag	Section	Data type
SourceUDT	Input	"myBits"
DestinationWord	Output	WORD

The following example shows how the instruction works:

GRAPH

```
CALL GATHER
    (IN := #SourceUDT
     OUT => #DestinationWord
    )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceUDT	The "SourceUDT" operand has the PLC data type (UDT). It consists of 16 elements and is therefore just as large as the WORD in which the bits are to be merged.
OUT	DestinationWord	WORD (16 bits)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

GATHER_BLK: Merge individual bits into multiple elements of an ARRAY of bit sequence (S7-1500)

Description

The "Merge individual bits into multiple elements of an ARRAY of bit sequence" instruction merges the bits from an ARRAY of BOOL, an anonymous STRUCT or a PLC data type exclusively with Boolean elements into one or multiple elements of an ARRAY of <bit sequence>. At the COUNT_OUT parameter you specify how many elements of the destination ARRAY are going to be written. With this step you also implicitly specify how many elements of the ARRAY of BOOL, the anonymous STRUCT or the PLC data type are required. The destination ARRAY at the OUT parameter may have more elements than specified at the COUNT_OUT parameter. The ARRAY of <bit sequence> must have sufficient elements to save the bits that are going to be merged. However, the destination ARRAY may also be larger.

Note

Multi-dimensional ARRAY of BOOL

If the ARRAY is a multi-dimensional ARRAY of BOOL, the filling bits of the dimensions contained are counted as well even if they were not explicitly declared.

Example 1: An ARRAY[1..10,0..4,1..2] of BOOL is handled like an ARRAY[1..10,0..4,1..8] of BOOL or like an ARRAY[0..399] of BOOL.

Example 2: At the OUT parameter, an ARRAY[0..5] of WORD (sourceArrayWord[2]) is interconnected. The COUNT_IN parameter has the value "3". At the IN parameter, an ARRAY[0..1,0..5,0..7] of BOOL (destinationArrayBool[0,0,0]) is interconnected. Both the array at the IN parameter and at the OUT parameter has a size of 96 bits. 48 individual bits are merged from the ARRAY of BOOL.

Note

If the ARRAY low limit of the source ARRAY is not "0", note the following:

For performance reasons the index must always start at a BYTE, WORD, DWORD or LWORD limit. This means the index must be calculated starting at the low limit of the ARRAY. The following formula is used as basis for this calculation:

Valid indices = ARRAY low limit + n(number of bit sequences) × number of bits of the desired bit sequence

For an ARRAY[-2..45] of BOOL and the bit sequence WORD the calculation looks as follows:

- Valid index (-2) = -2 + 0 × 16
- Valid index (14) = -2 + 1 × 16
- Valid index (30) = -2 + 2 × 16

You can find an example described below.

Note

Availability of the instruction

The instruction can be used with a CPU of the S7-1500 series as of firmware version 2.1.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- The index of the source ARRAY does not start at a BYTE, WORD, DWORD or LWORD limit. In this case, no result is written to the ARRAY of <bit sequence>.
- The ARRAY[*] of <bit sequence> does not provide the required number of elements. In this case as many bit sequences as possible are merged and written to the ARRAY of <bit sequence>. The remaining bits are no longer taken into account.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Element of an ARRAY[*] of BOOL, STRUCT or PLC data type	I, Q, M, D, L	ARRAY of BOOL, STRUCT or PLC data type whose bits are merged (source ARRAY) The values must not be located in the I/O area or in the DB of a technology object.
COUNT_OUT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L	Counter how many elements of the target ARRAY are to be described. The value must not be in the I/O area or in the database of a technology object.
OUT	Output	Element of an ARRAY[*] of <bit sequence>	I, Q, M, D, L	ARRAY of <bit sequence> to which the bits are saved (destination ARRAY)

You can find additional information on valid data types under "See also".

Example of a source ARRAY with the low limit "0"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayBool	Input	ARRAY[0..95] of BOOL
CounterOutput		UDINT
DestinationArrayWord	Output	ARRAY[0..5] of WORD

The following example shows how the instruction works:

GRAPH

```
CALL GATHER_BLK
    (IN := #SourceArrayBool[0]
     COUNT_OUT := #CounterOutput
     OUT => #DestinationArrayWord[2]
    )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayBool[0]	The operand "SourceArrayBool" is of the data type ARRAY[0..95] of BOOL. This means it provides 96 BOOL elements that can merged into words once again.
COUNT_OUT	CounterOutput = 3	UDINT3 (3 words are to be written. This means 48 bits must be available in the source ARRAY.)
OUT	DestinationArrayWord[2]	The operand "DestinationArrayWord" is of the data type ARRAY[0..5] of WORD. This means 6 WORD elements are available.

As of the 1st element of the #SourceArrayBool operand, 48 bits are merged in the #DestinationArrayWord operand. The starting point in the destination ARRAY is the 3rd element. This means that the first 16 bits are written into the 3rd word, the second 16 nits into the 4th word and the third 16 bits into the 5th word of the destination ARRAY.

Example of a source ARRAY with the low limit "-2"

Create the following tags in the block interface:

Tag	Section	Data type
SourceArrayBool	Input	ARRAY[-2..93] of BOOL
CounterOutput		UDINT
DestinationArrayWord	Output	ARRAY[0..5] of WORD

The following example shows how the instruction works:

GRAPH

```
CALL GATHER_BLK
    (IN := #SourceArrayBool[14]
     COUNT_OUT := #CounterOutput
     OUT => #DestinationArrayWord[2]
    )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Data type
IN	SourceArrayBool[14]	The operand "SourceArrayBool" is of the data type ARRAY[-2..93] of BOOL. Because the starting point is the 16th element, only 80 BOOL elements that can be merged into words again are available.
COUNT_OUT	CounterOutput = 3	UDINT3 (3 words are to be written. This means 48 bits must be available in the source ARRAY.)
OUT	DestinationArrayWord[2]	The operand "DestinationArrayWord" is of the data type ARRAY[0..5] of WORD. This means 6 WORD elements are available.

Starting with the 16th element of the #SourceArrayBool operand, 48 bits are merged into the #DestinationArrayWord operand. The starting point in the destination ARRAY is the 3rd element. This means the first 16 bits of the source ARRAY are ignored. The second 16 Bits are written into the 3rd word, the third 16 bits into the 4th word and the fourth 16 bits into the 5th word of the destination ARRAY. The remaining 64 bits of the source ARRAY are not taken into account either.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

SWAP: Swap (S7-1500)

Description

You use the "Swap" instruction to change the order of the bytes of the operand.

Depending on the data type used, you can swap all bytes in accumulator 1 or only the bytes in the right word of accumulator 1.

The following figure shows how the bytes of the operand of data type DWORD are swapped:

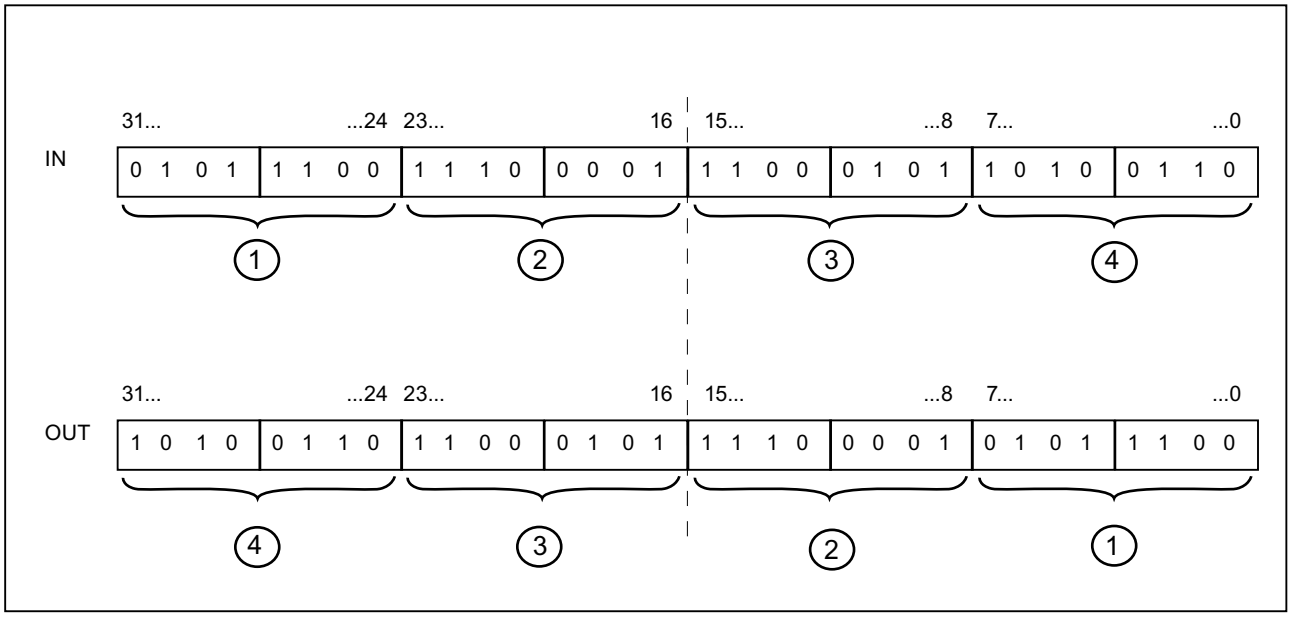

Order of bytes in the right word of accumulator 1

For data type WORD, you swap the bytes in the right word of accumulator 1.

The following table shows the contents of accumulator 1 before and after execution of the instruction:

Status	Bytes in accumulator 1			
Before execution	Value A	Value B	Value C	Value D
After execution	Value A	Value B	Value D	Value C

The result of the instruction is saved in the right word of accumulator 1. The bytes in the left word of accumulator 1 are not influenced by the instruction and remain unchanged.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	WORD, DWORD, LWORD	I, Q, M, D, L or constant	Operand whose bytes are swapped
<Result>	Output	WORD, DWORD, LWORD	I, Q, M, D, L	Result

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SWAP("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_InValue	0000	1111	0000	1111
Tag_OutValue	0000	1111	1111	0000

Swapping all bytes in accumulator 1:

For data type DWORD, you swap all bytes in accumulator 1.

The following table shows the contents of accumulator 1 before and after execution of the instruction:

Status	Bytes in accumulator 1			
Before execution	Value A	Value B	Value C	Value D
After execution	Value D	Value C	Value B	Value A

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_InValue	1111	0000	0000	1111	0000	0000	1111	1111
Tag_OutValue	1111	1111	0000	0000	0000	1111	1111	0000

The bytes in the "Tag_InValue" operand are swapped and stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

ARRAY DB (S7-1500)

ReadFromArrayDB: Read from array data block (S7-1500)

Description

The instruction "Read from ARRAY data block" is used to read data from a data block of the ARRAY DB block type and write it to a destination area.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB that is read. The specification can be a constant, a global tag or an indexed value.
VALUE	Output [1)]	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Value that is read and output
RET_VAL	Output	INT	I, Q, M, D, L, P	Result of the instruction
1) The VALUE parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, write protected, or is located in load memory.
8135	The ARRAY data block contains invalid values.
8154	The data block has the incorrect data type.
8282	The value at the INDEX parameter is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".
8452	Code generation error

Error code* (W#16#...)	Explanation
8453	There are two possible causes of the error: • The size of the VALUE parameter does not match the element length in the ARRAY data block. • The two tags are not in a memory area with optimized access. For additional information on the memory area access types, refer to: Basics of block access (Page 55)
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL ReadFromArrayDB
     (DB := "ArrayDB"
      INDEX := 2
      VALUE => "TargetField"
      RET_VAL => "TagResult"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type Array [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	TargetField	The "TargetField" operand is a global tag of the data type INT.

The element is read from "ArrayDB" and written to the "TargetField" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Evaluating errors with output parameter RET_VAL (Page 162)

Example of the use of ARRAY data blocks (Page 44)

Basic information on GRAPH (Page 7369)

WriteToArrayDB: Write to array data block (S7-1500)

Description

The instruction "Write to ARRAY data block" is used to write data to a data block of the ARRAY DB block type.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written. The specification can be a constant, a global tag or an indexed value.
VALUE	Input	VARIANT	L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Value to be written
RET_VAL	Output	INT	I, Q, M, D, L, P	Result of the instruction

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
80B5	The copying was interrupted.
8132	The data block does not exist, is too short, or is located in load memory.
8134	The data block is write protected.
8135	The data block is not an ARRAY data block.
8154	The data block has the incorrect data type.
8282	The value at the INDEX parameter is outside the limits of the ARRAY.
8350	The data type VARIANT at parameter VALUE provides the value "0".
8352	Code generation error

Error code* (W#16#...)	Explanation
8353	There are two possible causes of the error: • The size of the VALUE parameter does not match the element length in the ARRAY data block. • The two tags are not in a memory area with optimized access. For additional information on the memory area access types, refer to: Basics of block access (Page 55)
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL WriteToArrayDB
     (DB := "ArrayDB"
      INDEX := 2
      VALUE := "SourceField"
      RET_VAL => "TagResult"
      )
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type Array [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	SourceField	The "SourceField" operand is a global tag of the data type INT.

The value of the "SourceField" operand is written to the 2nd element of the ARRAY DB.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Evaluating errors with output parameter RET_VAL (Page 162)

Example of the use of ARRAY data blocks (Page 44)

Basic information on GRAPH (Page 7369)

ReadFromArrayDBL: Read from array data block in load memory (S7-1500)

Description

The instruction "Read from ARRAY data block in load memory" is used to read data from a data block of the ARRAY DB block type in the load memory and write it to a destination area.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Begin with the reading of the ARRAY DB
DB [1]	Input	DB_ANY	I, Q, M, D, L	ARRAY data block that is read.
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB that is read. The specification can be a constant, a global tag or an indexed value.
VALUE [1]	InOut	VARIANT	D (element of a global data block) L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Pointer to the DB in the work memory that is to be read and the value of which is to be written. No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the ERROR parameter.
[1] The data blocks must be created with the "Optimized" block property.				

Parameter ERROR

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at the INDEX parameter is outside the limits of the ARRAY.
8750	The data type VARIANT at parameter VALUE provides the value "0".
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL" and "WRIT_DBL" in the corresponding descriptions of the instructions.

Example

The following example shows how the instruction works:

GRAPH

```
CALL ReadFromArrayDBL DB_ANY, "ReadFromArrayDBL_DB"
     (REQ := "TagReg"
      DB := "ArrayDB"
      INDEX := 2
      VALUE := "TargetField"
      BUSY => "TagBusy"
      DONE => "TagDone"
      ERROR => "TagError"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type ARRAY [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"

Parameter	Operand	Value
VALUE	TargetField	The "TargetField" operand is a global tag of the data type INT.
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

The instruction is executed when a positive signal edge is detected at the "TagReq" operand. The 2nd element is read from "ArrayDB" and output at the "VALUE" parameter. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. After the instruction has been processed, the "TagDone" operand has the signal state TRUE.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Evaluating errors with output parameter RET_VAL (Page 162)

Example of the use of ARRAY data blocks (Page 44)

Difference between synchronous and asynchronous instructions (Page 596)

Basic information on GRAPH (Page 7369)

WriteToArrayDBL: Write to array data block in load memory (S7-1500)

Description

The instruction "Write to ARRAY data block in load memory" is used to write data to a data block of the ARRAY DB block type in the load memory.

An ARRAY data block is a data block that consists of exactly one ARRAY of [Data type]. The elements of the ARRAY can be of PLC data type or any other elementary data type. Counting of the ARRAY always begins with the low limit "0".

If the ARRAY data block has been designated with the block attribute "Only store in load memory", it will only be stored in the load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Start writing to the array DB
DB [1]	Input	DB_ANY	I, Q, M, D, L	ARRAY data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written. The specification can be a constant, a global tag or an indexed value.
VALUE [1]	Input	VARIANT	D (element of a global data block) L (The declaration is possible in the "Input", "InOut" and "Temp" sections of the block interface.)	Pointer to the DB in the work memory that is to be read and the value of which is to be written. No local constants or tags from the TEMP section must be used.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the ERROR parameter.
[1] The data blocks must be created with the "Optimized" block property.				

Parameter ERROR

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	The element data type stored in the ARRAY data block does not match the element data type transferred in the VARIANT.
8230	The data block number is incorrect.
8231	The data block does not exist.
8232	The data block is too short, or is not located in load memory.
8234	The data block is write protected.
8235	The data block is not an ARRAY DB.
8254	The data block has the incorrect data type.
8382	The value at the INDEX parameter is outside the limits of the ARRAY.
8450	The data type VARIANT at parameter VALUE provides the value "0".

Error code* (W#16#...)	Explanation
8751	Code generation error
8752	Code generation error
8753	The size of the VALUE parameter does not match the element length in the ARRAY data block.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL" and "WRIT_DBL" in the corresponding descriptions of the instructions.

Example

The following example shows how the instruction works:

GRAPH

```
CALL WriteToArrayDBL DB_ANY, "WriteToArrayDBL_DB"
     (REQ := "TagReg"
      DB := "ArrayDB"
      INDEX := 2
      VALUE := "SourceField"
      BUSY => "TagBusy"
      DONE => "TagDone"
      ERROR => "TagError"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The "ArrayDB" operand is an ARRAY DB of the data type ARRAY [0 to 10] of INT.
INDEX	2	2nd element of the "ArrayDB"
VALUE	SourceField	The "SourceField" operand is a global tag of the data type INT.
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

The instruction is executed when a positive signal edge is detected at the "TagReq" operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written to the 2nd element of "ArrayDB". After the instruction has been processed, the "TagDone" operand has the signal state TRUE.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Evaluating errors with output parameter RET_VAL (Page 162)

Example of the use of ARRAY data blocks (Page 44)

Difference between synchronous and asynchronous instructions (Page 596)

Basic information on GRAPH (Page 7369)

Legacy (S7-1500)

BLKMOV: Move block (S7-1500)

Description

You use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can also use this tag in memory areas "with optimized block access".

The following figure shows the principle of the move operation:

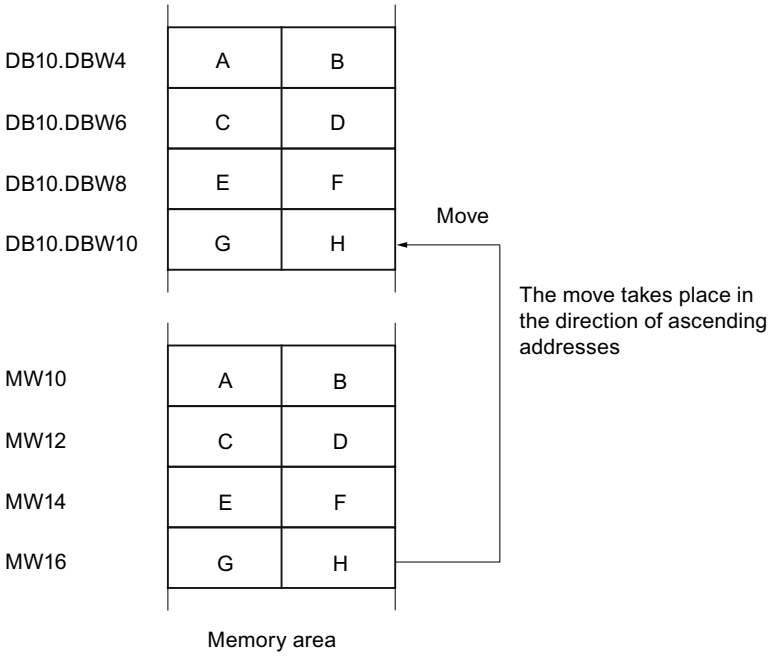

Consistency of source and destination data

Make sure that the source data remains unchanged during execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is of data type STRING, the characters that are actually contained in the character string will be moved. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	VARIANT [1]	I, Q, M, D, L, P	Specifies the memory area to which the block is moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is only available in the load memory.
8152	The WSTRING, WCHAR, BOOL, and ARRAY of STRING data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR, BOOL, and ARRAY of STRING data types are not supported at the DSTBLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL BLKMOV VARIANT
     (SRCBLK := P#M100.0 BYTE 10
      RET_VAL => "Tag_ErrorCode"
      DSTBLK => P#DB1.DBX0.0 BYTE 10
      )
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_ErrorCode" tag.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

GET_ERR_ID: Get error ID locally (Page 2020)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on GRAPH (Page 7369)

UBLKMOV: Move block uninterruptible (S7-1500)

Description

You use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can also use this tag in memory areas "with optimized block access".

Memory areas

You use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of data type STRING. If only the source area is of data type STRING, the characters that are actually contained in the character string will be moved. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as the area length.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1]	VARIANT	I, Q, M, D, L, P	Specifies the memory area to which the block is moved (destination area).
1) The DSTBLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source or destination area is only available in the load memory.
8152	The WSTRING, WCHAR, BOOL, and ARRAY of STRING data types are not supported at the SRCBLK parameter.
8352	The WSTRING, WCHAR, BOOL, and ARRAY of STRING data types are not supported at the DSTBLK parameter.

Error code* (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally"
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL UBLKMOV VARIANT
     (SRCBLK := P#M100.0 BYTE 10
      RET_VAL => "Tag_ErrorCode"
      DSTBLK => P#DB1.DBX0.0 BYTE 10
      )
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_ErrorCode" tag.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

GET_ERR_ID: Get error ID locally (Page 2020)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on GRAPH (Page 7369)

FILL: Fill block (S7-1500)

Description

You use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used within memory areas in which the "Optimized block access" attribute is not activated. This applies to data blocks (DBs), organization blocks (OBs), functions (FCs), bit memory (M), inputs (I), and outputs (Q).

If a tag of the instruction has been declared with the retentivity setting "Set in IDB", however, you can also use this tag in memory areas "with optimized block access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK instruction: Fill block" instruction.

The following figure shows the principle of the move operation:

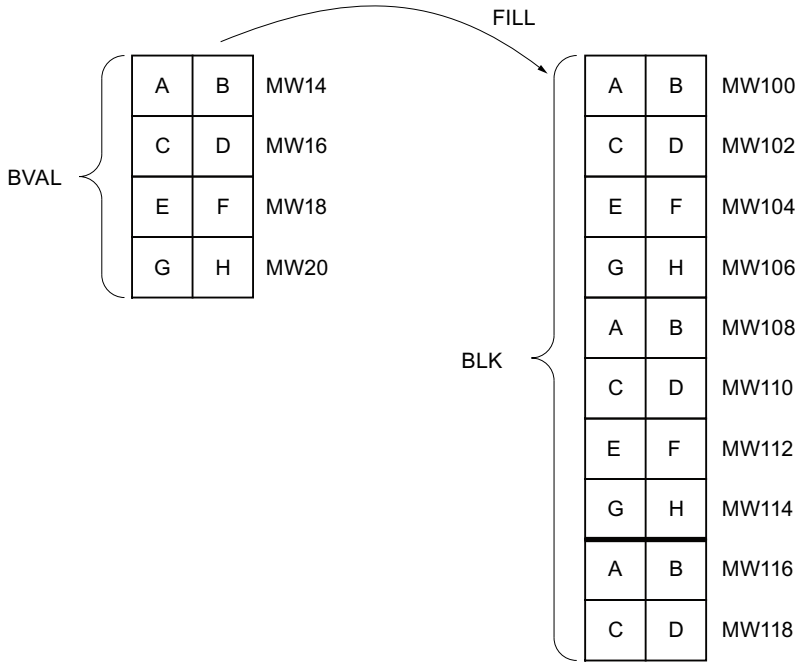

Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Make sure that the source data remain unchanged during execution of the "Fill block" instruction, since otherwise consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory

- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
BVAL	Input	VARIANT	I, Q, M, D, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
BLK	Output [1)]	VARIANT	I, Q, M, D, L, P	Specification of the memory area that will be filled with the content of the source area.
1) The BLK parameter is declared as Output, since the data flow into the tag. However, the tag itself must be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is only available in the load memory.
8152	The WSTRING, WCHAR, BOOL, and ARRAY of STRING data types are not supported at the BVAL parameter.
8352	The WSTRING, WCHAR, BOOL, and ARRAY of STRING data types are not supported at the BLK parameter.
General error information	See also: "GET_ERR_ID: Get error ID locally"
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL FILL VARIANT
     (BVAL := P#M14.0 WORD 4
      RET_VAL => "Tag_ErrorCode"
      BLK => P#M100.0 WORD 10
      )
```

The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

GET_ERR_ID: Get error ID locally (Page 2020)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on GRAPH (Page 7369)

Conversion operations (S7-1500)

CONVERT: Convert value (S7-1500)

Description

You use the "Convert value" instruction to read the content of the operand and convert it according to the assigned data types.

For information on possible conversions, refer to section "Explicit conversion" under "See also".

The value of the result is invalid if errors, such as an overflow, occur during processing.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Binary numbers, integers, floating-point numbers, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted.
<Result>	Output	Binary numbers, integers, floating-point numbers, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data types of the instruction from the "???" drop-down lists.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := INT_TO_DINT("Tag_InValue")
```

The content of the "Tag_InValue" operand is read and converted to an integer (32-bit). The result is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

Explicit conversions (Page 405)

ROUND: Round numerical value (S7-1500)

Description

You use the "Round numerical value" instruction to round the value of the operand to the nearest integer. The instruction interprets the value as a floating-point number and converts it into to the next integer. If the input value is exactly between an even and odd number, the even number is selected.

The value of the result is invalid if errors, such as an overflow, occur during processing.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value to be rounded.
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result of rounding

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ROUND("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	1.50000000	-1.50000000
Tag_OutValue	2	-2

The floating-point number in the "Tag_InValue" operand is rounded to the nearest even integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

CEIL: Generate next higher integer from floating-point number (S7-1500)

Description

You use the "Generate next higher integer from floating-point number" instruction to round the value of the operand to the next higher integer. The instruction interprets the value as a floating-point number and converts this number to the next higher integer. The result can be greater than or equal to the input value.

The value of the result is invalid if errors, such as an overflow, occur during processing.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value as floating-point number
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next higher integer

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := CEIL("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	0.50000000	-0.50000000
Tag_OutValue	1	0

The floating-point number in the "Tag_InValue" operand is rounded to the next higher integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

FLOOR: Generate next lower integer from floating-point number (S7-1500)

Description

You use the "Generate next lower integer from floating-point number" instruction to round the value of the operand to the next lower integer. The instruction interprets the value as a floating-point number and converts this number to the next lower integer. The result can be less than or equal to the input value.

The value of the result is invalid if errors, such as an overflow, occur during processing.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value as floating-point number
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next lower integer

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := FLOOR("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	0.50000000	-0.50000000
Tag_OutValue	0	-1

The floating-point number in the "Tag_InValue" operand is rounded to the next lower integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

TRUNC: Truncate numerical value (S7-1500)

Description

You use the "Truncate numerical value" instruction to form an integer from the value of the operand. The value is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and outputs it without decimal places as the result.

The value of the result is invalid if errors, such as an overflow, occur during processing.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value as floating-point number
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L	Result with integer part of the floating-point number

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := TRUNC("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	1.50000000	-1.50000000
Tag_OutValue	1	-1

The integer part of the floating-point number in the "Tag_InValue" operand is converted to an integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

SCALE_X: Scale (S7-1500)

Description

You use the "Scale" instruction to scale the value of the VALUE parameter by mapping it to a specified value range. When the instruction is executed, the floating-point value in the VALUE input is scaled to the value range that was defined by the MIN and MAX parameters. The result of the scaling is an integer, which is stored in the OUT output.

The following figure shows an example of how values can be scaled:

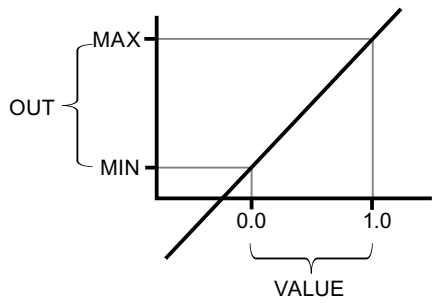

The "Scale" instruction works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

The value of the result is invalid if any of the following conditions are met:

- The value of the MIN parameter is greater than or equal to the value of the MAX parameter.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value of the VALUE parameter is NaN (Not a Number = result of an invalid arithmetic operation).

Note

For additional information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
RET_VAL	Output	Integers, floating-point numbers	I, Q, M, D, L	Result of scaling

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SCALE_X INT_REAL
     (MIN := "Tag_Minimum"
      VALUE := "Tag_Value"
      MAX := "Tag_Maximum"
      RET_VAL => "Tag_OutputValue"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_Minimum	10
VALUE	Tag_Value	0.5
MAX	Tag_Maximum	30
RET_VAL	Tag_ReturnValue	20

The value of the "Tag_Value" parameter is scaled to the range of values defined by the values of the "Tag_Minimum" and "Tag_Maximum" parameters. The result is stored in the "Tag_ReturnValue" parameter.

See also

Overview of the valid data types (Page 231)

Basic information on GRAPH (Page 7369)

NORM_X: Normalize (S7-1500)

Description

You use the "Normalize" instruction to normalize the value of the VALUE parameter by mapping it to a linear scale. You use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. The result is calculated based on the position of the value to be normalized within this value range and stored as a floating-point number in the OUT parameter. If the value to be normalized is equal to the value of the MIN parameter, the OUT parameter has the value "0.0". If the value to be normalized is equal to the value of the MAX parameter, the OUT parameter has the value "1.0".

The following figure shows an example of how values can be normalized:

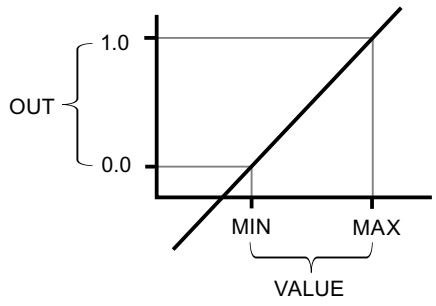

The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

The value of the result is invalid if any of the following conditions are met:

- The value of the MIN parameter is greater than or equal to the value of the MAX parameter.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value of the VALUE parameter is NaN (Not a Number = result of an invalid arithmetic operation).

Note

For additional information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
MIN [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Value to be normalized
MAX [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
RET_VAL	Output	Floating-point numbers	I, Q, M, D, L	Result of the normalization
[1] If you use constants in these three parameters, you only need to declare one of them.				

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL NORM_X INT_REAL
      (MIN := "Tag_Minimum"
       VALUE := "Tag_Value"
       MAX := "Tag_Maximum"
       RET_VAL => "Tag_OutputValue"
       )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_Minimum	10
VALUE	Tag_Value	20
MAX	Tag_Maximum	30
RET_VAL	Tag_ReturnValue	0.5

The value of the "Tag_Value" parameter is assigned to the range of values that were defined with the "Tag_Minimum" and "Tag_Maximum" parameters. The tag value in the "Tag_Value" parameter is normalized according the defined value range. The result is stored as a floating-point number in the "Tag_ReturnValue" parameter.

See also

Overview of the valid data types (Page 231)

Basic information on GRAPH (Page 7369)

Legacy (S7-1500)

SCALE: Scale (S7-1500)

Description

You use the "Scale" instruction to convert the integer of the IN parameter to a floating-point number that can be scaled in physical units between a low limit and a high limit. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and lies in a value range between -27 648 and 27 648. In this case the constant "K1" has the value "-27 648.0" and the constant "K2" the value "+27 648.0".
- Signal state "0": It is assumed that the value of the IN parameter is unipolar and lies in a value range between 0 and 27 648. In this case the constant "K1" has the value "0.0" and the constant "K2" the value "+27 648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit (LO_LIM) and an error is output.

When the specified low limit is greater than the high limit (LO_LIM > HI_LIM), the result is scaled inversely proportional to the input value.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit

Parameter	Declaration	Data type	Memory area	Description
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"
General error information	See also: "GET_ERR_ID: Get error ID locally"
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL SCALE
     (IN := "Tag_InputValue"
      HI_LIM := "Tag_HighLimit"
      LO_LIM := "Tag_LowLimit"
      BIPOLAR := "Tag_Bipolar"
      RET_VAL => "Tag_ErrorCode"
      OUT => "Tag_OutputValue"
      )
```

The following table shows the values of the various operands before execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0

Parameter	Operand	Value
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	50.03978588

The "Tag_InputValue" operand indicates the value that is to be converted and scaled. The high and low limits are defined by the "Tag_HighLimit" and "Tag_LowLimit" operands. You use operand "Tag_Bipolar" = TRUE to specify that the value of the IN parameter is to be interpreted as bipolar. The result of the instruction is output at the "Tag_OutputValue" operand.

See also

Overview of the valid data types (Page 231)

GET_ERR_ID: Get error ID locally (Page 2020)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

UNSCALE: Unscale (S7-1500)

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and lies in a value range between -27 648 and 27 648. In this case the constant "K1" has the value "-27 648.0" and the constant "K2" the value "+27 648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and lies in a value range between 0 and 27 648. In this case the constant "K1" has the value "0.0" and the constant "K2" the value "+27 648.0".

When the value at parameter IN is not within the limits defined by HI_LIM and LO_LIM, an error is output and the result is set to the nearest limit.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).

Error code (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally"
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

GRAPH

```
CALL UNSCALE
      (IN := "Tag_InputValue"
       HI_LIM := "Tag_HighLimit"
       LO_LIM := "Tag_LowLimit"
       BIPOLAR := "Tag_Bipolar"
       RET_VAL => "Tag_ErrorCode"
       OUT => "Tag_OutputValue"
       )
```

The following table shows the values of the various operands before execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	22

The "Tag_InputValue" operand indicates the value that is to be converted and unscaled. The high and low limits are defined by the "Tag_HighLimit" and "Tag_LowLimit" operands. You use operand "Tag_Bipolar" = TRUE to specify that the value of the IN parameter is to be interpreted as bipolar. The result of the instruction is output at the "Tag_OutputValue" operand.

See also

Overview of the valid data types (Page 231)

GET_ERR_ID: Get error ID locally (Page 2020)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on GRAPH (Page 7369)

Program control operations (S7-1500)

Runtime control (S7-1500)

ENDIS_PW: Limit and enable password legitimation (S7-1500)

Description

You use the "Limit and enable password legitimation" instruction to specify whether passwords configured for the CPU are legitimized or not. In this way, you can prevent legitimized connections, even if the correct password is known.

When you call the instruction and the REQ parameter has the signal state "0", the currently set state is displayed at the output parameters. If changes have been made to the input parameters, these are not transferred to the output parameters.

When the instruction is called and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD):

- If signal state "0" is present, authorization via password is not allowed.
- If signal state "1" is present, the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited, except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Passwords that are not configured must have the signal state TRUE at the input and return the signal state TRUE at the output. The fail-safe password can only be configured for an F-CPU and must therefore always be interconnected with the signal state TRUE in a standard CPU. If the instruction returns an error, the call has no effect and the previous lock is still in effect.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".

- Set the mode selector to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Plugging an empty memory card (transfer module or program card) into an S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called once again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access from HMI systems if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing unintentional lockout on an S7-1500 CPU

The settings can be made on the front panel of the CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode selector to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing unintentional lockout on an S7-1200 CPU

Because the S7-1200 CPU does not have a mode switch, protection is disabled with POWER OFF - POWER ON. This means that it is possible and advisable to prevent accidental lockout with the help of specific program sequences within your program.

To do so, program a time control either by means of a cyclic interrupt OB or a timer in the Main OB (OB 1). This gives you the option of calling the "Limit and enable password legitimation" instruction in the respective OB (e.g., OB 1 or OB 35) relatively quickly after a transition from POWER OFF to POWER ON and the associated disabling of the protection. Call the instruction in the startup OB (OB 100) to keep the time window in which the instruction is inactive and there are no limitations for password legitimation relatively small. This procedure will give you the best possible protection from unauthorized access.

If there has been an accidental lockout, you can skip the call in the startup OB (by querying an input parameter, for example) and you have the set time (for example, 10 seconds up to 1 minute) to establish a connection to the CPU before the lock becomes active once again.

If there is no timer in your program code and a lockout has occurred, insert an empty transfer card or an empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then download the user program once again from STEP 7 to the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

WARNING

Inserting an empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Once you have pulled the transfer card, the content of the transfer card is available in the internal load memory. Make sure that the card does not include a program at this point.

WARNING

Inserting an empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. Once you have pulled the previously empty program card, the internal load memory is empty.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200 CPU: The lock is disabled and the instruction must be called once again in the program (e.g., in the startup OB). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password

Parameter	Declaration	Data type	Memory area	Description
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported
80D0	The password for fail-safe is not configured. The signal state must be TRUE for standard CPUs.
80D1	The read/write access is not configured

Error code* (W#16#...)	Explanation
80D2	The read access is not configured
80D3	The HMI access is not configured
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

RE_TRIGR: Restart cycle monitoring time (S7-1500)

Description

The "Restart cycle monitoring time" instruction is used to restart the cycle monitoring time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

If the instruction is called in a block with a higher priority, such as a hardware interrupt or diagnostic error interrupt, the instruction is not executed and the enable output ENO returns signal state "0".

The instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Calling the instruction

You can call the instruction regardless of the priority in all organization blocks.

Parameters

The "Restart cycle monitoring time" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 231)

Basic information on GRAPH (Page 7369)

STP: Exit program (S7-1500)

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Parameters

The "Exit program" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 231)

Basic information on GRAPH (Page 7369)

GET_ERROR: Get error locally (S7-1500)

Description

The "Get error locally" instruction is used to query the occurrence of errors within a program block. This normally involves programming or access errors. If the system reports an error during the execution of the program block, detailed information is stored in the operand at the OUT output for the first error to occur during the execution of the block since the last execution of the instruction.

Only operands of the "ErrorStruct" system data type can be specified at the OUT output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the program block, the error information about the next error to occur is output by the instruction only after the first error that occurred has been remedied.

Note

Output OUT

The OUT output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.

Note

Activation of the local error handling

As soon as you insert the instruction in the program code of a program block, the local error handling is activated and default system reactions are ignored when errors occur.

Options for error handling

You can find an overview of the options for error handling here: Overview of mechanisms for error handling (Page 138)

For a detailed example of local error handling that contains several error handling options, refer to: Example of how to handle program execution errors (Page 167)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
OUT	Output	ErrorStruct	D, L	Error information

Data type "ErrorStruct"

The "ErrorStruct" data type can be inserted in a global data block or in the block interface. You can insert the data type more then once if you assign a different name for the data structure each time. The data structure and the name of individual structure elements cannot be changed. If you save the error information in a global data block, it can also be read out by other program blocks.

The following table shows the structure of the "ErrorStruct" data type:

Structure component		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows whether the error occurred during a program block call. 16#01: Error during a program block call 16#00: No error during a program block call
REACTION		BYTE	Default reaction: 0: Ignore (write error) 1: Continue with substitute value "0" (read error) 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of the program block
	BLOCK_TYPE	BYTE	Program block type in which the error occurred: 1: Organization block (OB) 2: Function (FC) 3: Function block (FB)
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command

Structure component		Data type	Description
POINTER_NUMBER_LOCA-TION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand
	AREA	BYTE	(C) Memory area: L: 16#40...16#7F, 16#86, 16#87, 16#8E, 16#8F, 16#C0...16#FF I: 16#81 Q: 16#82 M: 16#83 DB: 16#40, 16#84, 16#85, 16#8A, 16#8B PI: 16#01 PQ: 16#02 Technological objects: 16#04
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

The following table shows the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid reference used
2534	9524	Block number error FC
2535	9525	Block number error FB
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist

ID* (hexadecimal)	ID* (decimal)	Description
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

Basic information on GRAPH (Page 7369)

GET_ERR_ID: Get error ID locally (S7-1500)

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the error ID of the first error that occurred in the tag is stored at the RET_VAL output.

Only operands of the WORD data type can be specified at the RET_VAL output. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The RET_VAL output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.

The output of the "Get error ID locally" instruction is only set if error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Options for error handling

You can find an overview of the options for error handling here: Overview of mechanisms for error handling (Page 138)

For a detailed example of local error handling that contains several error handling options, refer to: Example of how to handle program execution errors (Page 167)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	WORD	I, Q, M, D, L	Error ID

Parameter RET_VAL

The following table shows the values that can be output at the RET_VAL parameter:

Error code* (hexadecimal)	Error code* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid reference used
2534	9524	Block number error FC
2535	9525	Block number error FB
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist

Error code* (hexadecimal)	Error code* (decimal)	Description
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Use of the instructions GET_ERROR and GET_ERR_ID (Page 166)

Basic information on GRAPH (Page 7369)

INIT_RD: Initialize all retain data (S7-1500)

Description

You use the "Initialize all retain data" instruction to reset the retentive data of all data blocks, bit memories, and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution exceeds the program cycle duration.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	If the input "REQ" has the signal state "1", all retentive data are reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally"
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL INIT_RD
     (REQ := "Tag_REQ"
      RET_VAL => "Tag_RET_VAL"
      )
```

If the "Tag_REQ" operand has signal state "1", the instruction is executed. All retentive data of all data blocks, bit memories, and SIMATIC timers and counters are reset.

See also

Overview of the valid data types (Page 231)

GET_ERR_ID: Get error ID locally (Page 2020)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

WAIT: Configure time delay (S7-1500)

Description

You can use the "Configure time delay" instruction to pause the program execution for a specific period of time. You indicate the period of time in microseconds in the WT parameter.

You can configure time delays from -32768 up to 32767 microseconds (µs). The smallest possible delay time depends on the CPU and corresponds to the execution time of the instruction.

The execution of the instruction can be interrupted by higher priority events and does not return any error information.

Note

Negative delay time

If you specify a negative delay time at parameter WT, the enable output ENO, or the RLO and the BR bit, will return the signal state FALSE. A negative delay time does not affect the CPU. The following instructions linked to enable output ENO are not executed in LAD or FBD.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

Example of the influence of the planned delay time

In the following example, you will see the influence of the delay time of the instruction "WAIT" in various scenarios.

The following figure is a schematic representation of the scenarios:

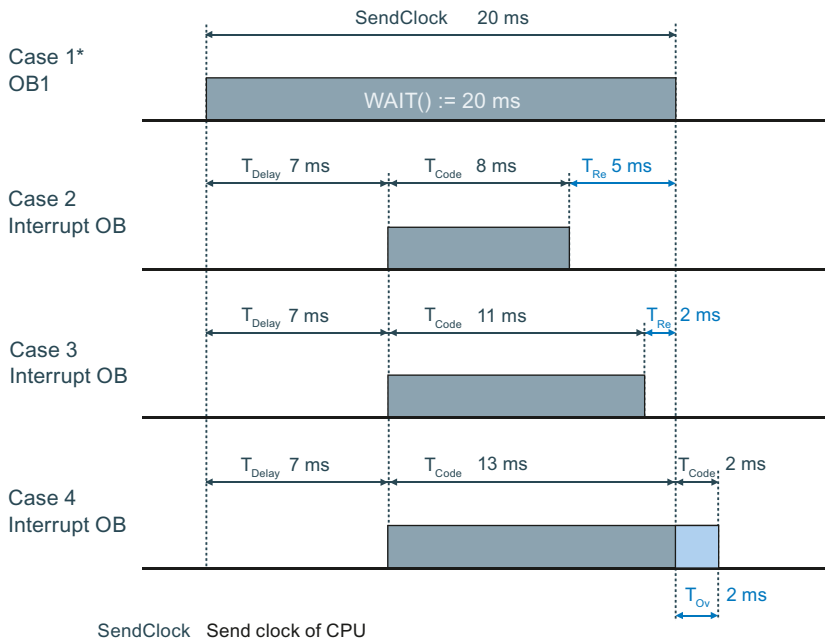

SendClock	Send clock of CPU
T_{Delay}	Delay time of the interrupt OB
T_{Code}	Time of program execution
T_{Re}	Remaining time
T_{Ov}	Excess time by which the send clock is increased

* The OB1 with the instruction "Wait" is also part of cases 2, 3 and 4 and needs to be taken into account.

Remaining time = The period between the end of the planned delay time (via "WAIT") until the end of the interrupt OB

Excess time = The period between the end of the interrupt OB and the end of the planned delay time (via "WAIT").

Case 1:

The "WAIT" instruction is called in an OB1. The "WAIT" instruction can be interrupted by higher-priority OBs or higher-priority processes (e.g. System Threads). However, the delay time of the "WAIT" instruction is neither changed nor deferred.

Cases 2 and 3:

Processing of the program in OB1 is resumed after a time delay of 20 ms. This time delay is calculated by calling the "WAIT" instruction in OB1 (see OB1 with WAIT). Within these 20 ms, an interrupt OB can run its own program code. The send clock of the CPU is not changed.

Case 4:

Processing of the program in OB1 is resumed after the higher-priority process has ended. The 20 ms delay time in OB1 has elapsed, but the higher-priority process still needs to be ended. The send clock of the CPU is increased.

Note

Order of execution of system or communication processes (System Threads)

The System Threads usually use the priority "15". There are also System Threads with a higher priority than "26", but these processes cause a lower CPU load. System Threads are not displayed in the figure.

Runtime measurement of the OB1 using the "RT_INFO" instruction:

Case 2: 20 ms - 8 ms - System Threads = <12 ms. Send clock: ~20 ms.

Case 3: 20 ms - 11 ms - System Threads - <9 ms. Send clock: ~20 ms.

Case 4: 20 ms - 15 ms - System Threads - <7 ms. Send clock: ~22 ms.

See also

Overview of the valid data types (Page 231)

Basic information on GRAPH (Page 7369)

Word logic operations (S7-1500)

NOT (INV): Inverting (S7-1500)

Description

You use the instruction "Inverting" to invert the signal status of the bits of the operand. When the instruction is executed, the value of the operand is linked by EXCLUSIVE OR to a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit numbers). This inverts the signal state of the individual bits that are then output as the result.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Bit strings, integers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Bit strings, integers	I, Q, M, D, L, P	Ones complement of the operand value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"TagOut_Value" := NOT("TagIn_Value")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
TagIn_Value	W#16#000F
TagOut_Value	W#16#FFF0

The instruction inverts the signal state of the individual bits at "TagIn_Value" operand and writes the result to the "TagOut_Value" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

DECO: Decode (S7-1500)

Description

You use the "Decode" instruction to set a bit in the output value that is specified by the input value .

The "Decode" instruction reads the value of the IN parameter and sets the bit in the OUT parameter whose bit position corresponds to the value read. The other bits in the output value are filled with zeroes. When the value of the IN parameter is greater than 31, a modulo-32 operation is executed.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	UINT	I, Q, M, D, L, P or constant	Position of the bit in the output value that is set.
OUT	Output	Bit strings	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL DECO UINT_DWORD
      (IN := "Tag_Input"
       OUT => "Tag_Output"
      )
```

The following figure shows how the instruction functions using specific values:

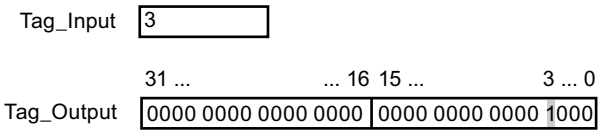

The instruction reads bit number "3" from the value of the "Tag_Input" operand of the input and sets the third bit in the value of the "Tag_Output" operand of the output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

ENCO: Encode (S7-1500)

Description

You use the "Encode" instruction to read the bit number of the least significant set bit in the input value and output it in the OUT parameter.

The "Encode" instruction selects the least significant bit of the value in the IN parameter and writes its bit number in the operands in the OUT parameter. If the IN parameter has the value DW#16#00000001 or DW#16#00000000, the value "0" is output in the OUT parameter.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

```
CALL ENCO DWORD
     (IN := "Tag_Input"
      OUT => "Tag_Output"
     )
```

The following figure shows how the instruction functions using specific values:

	31 16	15 ... 3 ... 0
"Tag_Input"	0000 1111 0000 0101	0000 1001 0000 1000
"Tag_Output"	3	

The instruction selects the least significant set bit of the "Tag_Input" tag and writes its bit position "3" to the "Tag_Output" tag.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

SEL: Select (S7-1500)

Description

You use the "Select" instruction to select, dependent on a switch (G input), one of two inputs, IN0 or IN1, and move its content to the OUT output. If the G input has the signal state "0", the value at the IN0 input is moved. If the G input has the signal state "1", the value of the IN1 input is moved to the OUT output.

The instruction is only executed if the tags of all parameters have the same data type.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
G	Input	BOOL	I, Q, M, D, L or constant	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SEL CHAR
     (G := "Tag_Input_G"
      IN0 := "Tag_Input0"
      IN1 := "Tag_Input1"
      OUT => "Tag_Output"
     )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value
G	Tag_Input_G	1
IN0	Tag_Input0	W#16#0000
IN1	Tag_Input1	W#16#FFFF
OUT	Tag_Output	W#16#FFFF

Based on the signal state of the "Tag_Input_G" input, the value of the "Tag_Input0" or "Tag_Input1" input is selected and moved to the "Tag_Output" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

MUX: Multiplex (S7-1500)

Description

You use the "Multiplex" instruction to copy the content of a selected input to the RET_VAL output. The number of selectable inputs can be extended to a maximum of 32. Numbering starts at IN0 and continues consecutively with each new input. You use the K parameter to define the input whose content is to be copied to the RET_VAL output. If the value of the K parameter is greater than the number of available inputs, the content of the INELSE parameter is copied to the RET_VAL output.

The "Multiplex" instruction can only be executed if the tags in all inputs and in the RET_VAL output have the same data type. The K parameter is an exception, since only integers can be specified for it.

The value of the RET_VAL parameter is invalid if one of the following conditions is met:

- The input at the K parameter is located outside the available inputs. This reaction is independent of whether the input INELSE is used or not. The value at the output RET_VAL remains unchanged and the enable ouput ENO is set to "0".
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
K	Input	Integers	I, Q, M, D, L or constant	Specifies the input whose content is to be copied.
IN0	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L or constant	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L or constant	Second input value
INn	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L or constant	Optional input values

Parameter	Declaration	Data type	Memory area	Description
INELSE	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L or constant	Specifies the value that is copied when K > n.
RET_VAL	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L	Output to which the value is copied.

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MUX SINT_SINT
      (K := "Tag_Number"
       IN0 := "Tag_Value_1"
       IN1 := "Tag_Value_2"
       INELSE := "Tag_Value_3"
       RET_VAL => "Tag_Result"
       )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
K	Tag_Number	1
IN0	Tag_Value_1	DW#16#00000000
IN1	Tag_Value_2	DW#16#3E4A7D
INELSE	Tag_Value_3	DW#16#FFFF0000
RET_VAL	Tag_Result	DW#16#3E4A7D

Depending on the value of the "Tag_Number" operand, the value of the "Tag_Value_1" input is copied and output to the operand in the "Tag_Result" output.

See also

Overview of the valid data types (Page 231)

Basic information on GRAPH (Page 7369)

DEMUX: Demultiplex (S7-1500)

Description

You use the "Demultiplex" instruction to copy the content of the IN input to a selected output. The number of selectable outputs can be extended to a maximum of 32. Numbering starts at OUT0 and continues consecutively with each new output. You use the K parameter to define the output to which the content of the IN input is to be copied. The other outputs are not changed. If the value of the K parameter is greater than the number of available outputs, the content of the IN input is copied to the OUTELSE parameter.

The "Demultiplex" instruction can only be executed if the tags in the IN input and in all outputs have the same data type. The K parameter is an exception, since only integers can be specified for it.

The value of the OUTELSE output is invalid if one of the following conditions is met:

- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
K	Input	Integers	I, Q, M, D, L or constant	Specifies the output to which the input value (IN) is copied.
IN	Input	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L or constant	Input value
OUT0	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L	First output
OUT1	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L	Second output

Parameter	Declaration	Data type	Memory area	Description
OUTn	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L	Optional outputs
OUTELSE	Output	Binary numbers, integers, floating-point numbers, timers, CHAR, WCHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L	Output to which the input value (IN) is copied when K > n.

You can select the data type of the instruction from the "???" drop-down list.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL DEMUX SINT_SINT
      (K := "Tag_Number"
       IN := "Tag_Value"
       OUT0 => "Tag_Output_1"
       OUT1 => "Tag_Output_2"
       OUTELSE => "Tag_Output_3"
      )
```

The following tables show how the instruction works using specific operand values:

Table 4-5 Input values of the "Demultiplex" instruction before network execution:

Parameter	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#3E4A7D

Table 4-6 Output values of the "Demultiplex" instruction after network execution:

Parameter	Operand	Values	
OUT0	Tag_Output_1	Unchanged	Unchanged
OUT1	Tag_Output_2	DW#16#FFFFFFFF	Unchanged
OUTELSE	Tag_Output_3	Unchanged	DW#16#3E4A7D

Based on the value of the "Tag_Number" operand, the value of the IN input is copied to the corresponding output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

Shift and Rotate (S7-1500)

SHR: Shift right (S7-1500)

Description

You use the "Shift right" instruction to shift the content of Operand1 to the right bitwise. You use Operand2 to specify the number of bit positions the specified value will be shifted.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is shifted to the right by the available number of bit positions.

When unsigned values are shifted, the free bit positions in the left area of Operand1 are filled by zeroes. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure shows how the content of Operand1 of data type INT is shifted by four bit positions to the right:

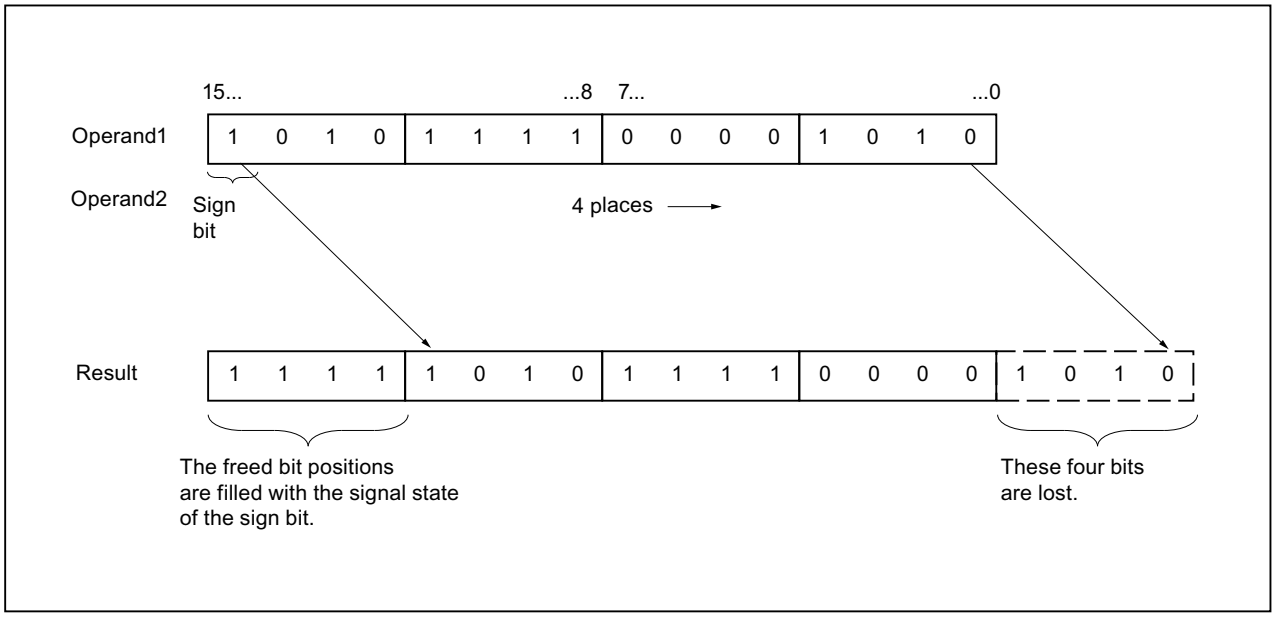

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted
<Operand2>	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SHR_INT("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0011 1111 1010 1111
Tag_Number	3
Tag_OutValue	0000 0111 1111 0101

The content of the "Tag_InValue" operand is shifted three bit positions to the right. The result is output in the "Tag_OutValue" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

SHL: Shift left (S7-1500)

Description

You use the "Shift left" instruction to shift the content of Operand1 to the left bitwise. You use Operand2 to specify the number of bit positions the specified value will be shifted.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is shifted to the left by the available number of bit positions.

The bit positions in the right part of Operand1 freed by shifting are filled with zeros.

The following figure shows how the content of Operand1 of data type WORD is shifted by six bit positions to the left:

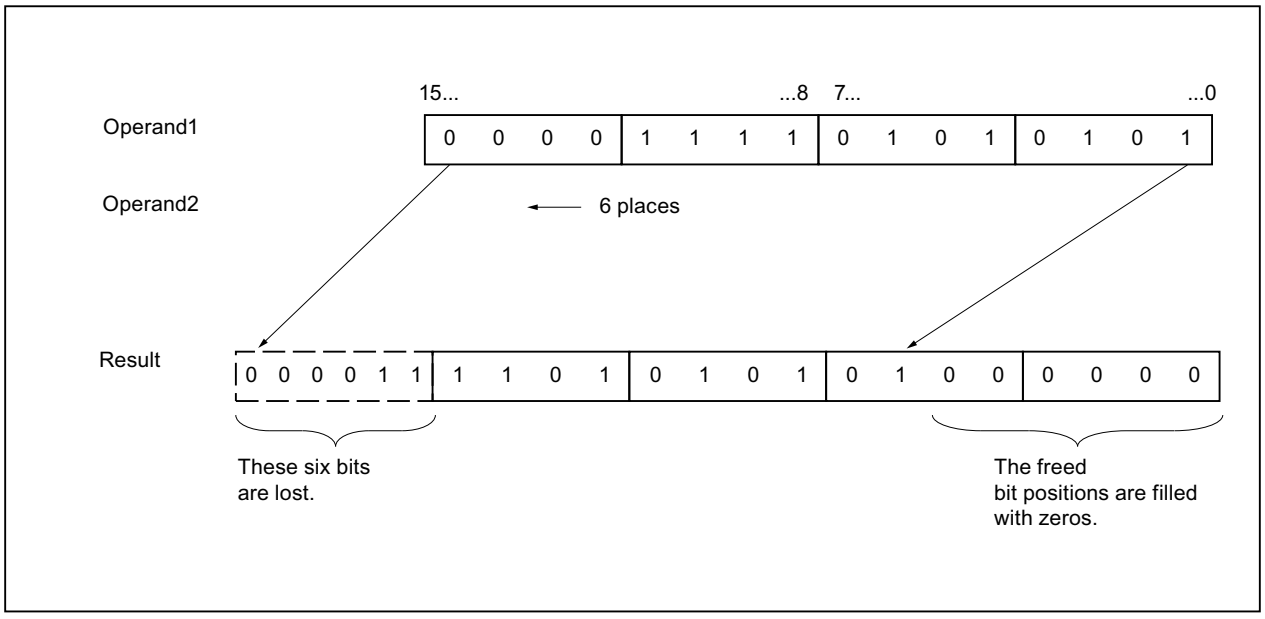

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted
<Operand2>	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SHL_WORD("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0011 1111 1010 1111
Tag_Number	4
Tag_OutValue	1111 1010 1111 0000

The content of the "Tag_InValue" operand is shifted four bit positions to the left and output as the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

ROR: Rotate right (S7-1500)

Description

You use the "Rotate right" instruction to rotate the content of Operand1 to the right bitwise. You use Operand2 to specify the number of bit positions the specified value will be rotated.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is rotated by the specified number of bit positions.

The bit positions freed by rotating are filled with the bit positions that are pushed out.

The following figure shows how the content of an operand of data type DWORD is rotated three bit positions to the right:

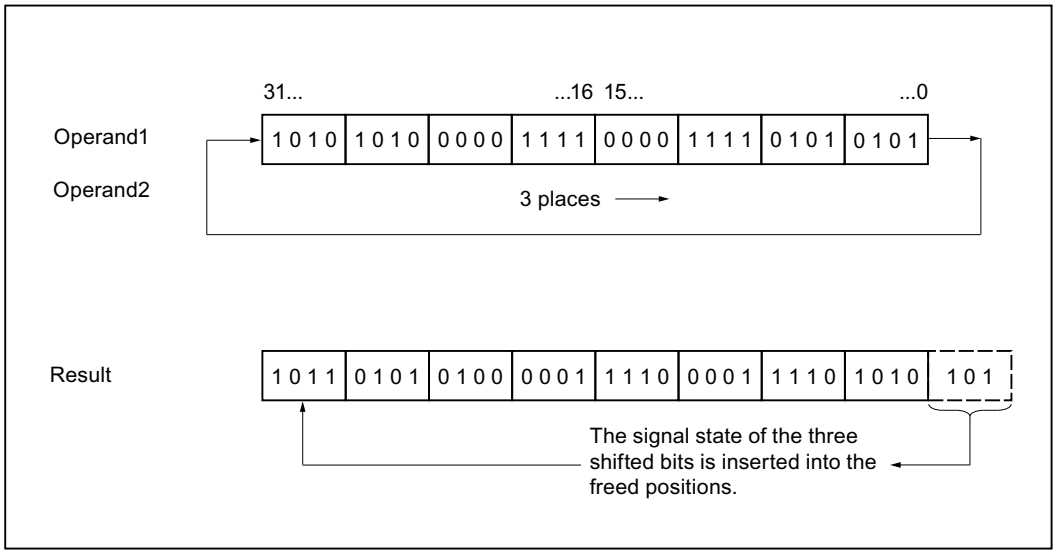

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L or constant	Value to be rotated
<Operand2>	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ROR("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1010 1010 0000 1111 0000 1111 0101 0101
Tag_Number	5
Tag_OutValue	1010 1101 0101 0000 0111 1000 0111 1010

The content of the "Tag_InValue" operand is rotated five bit positions to the right and output as the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

ROL: Rotate left (S7-1500)

Description

You use the "Rotate left" instruction to rotate the content of Operand1 to the left bitwise. You use Operand2 to specify the number of bit positions the specified value will be rotated.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is rotated by the specified number of bit positions.

The bit positions freed by rotating are filled with the bit positions that are pushed out.

The following figure shows how the content of an operand of data type DWORD is rotated three bit positions to the left:

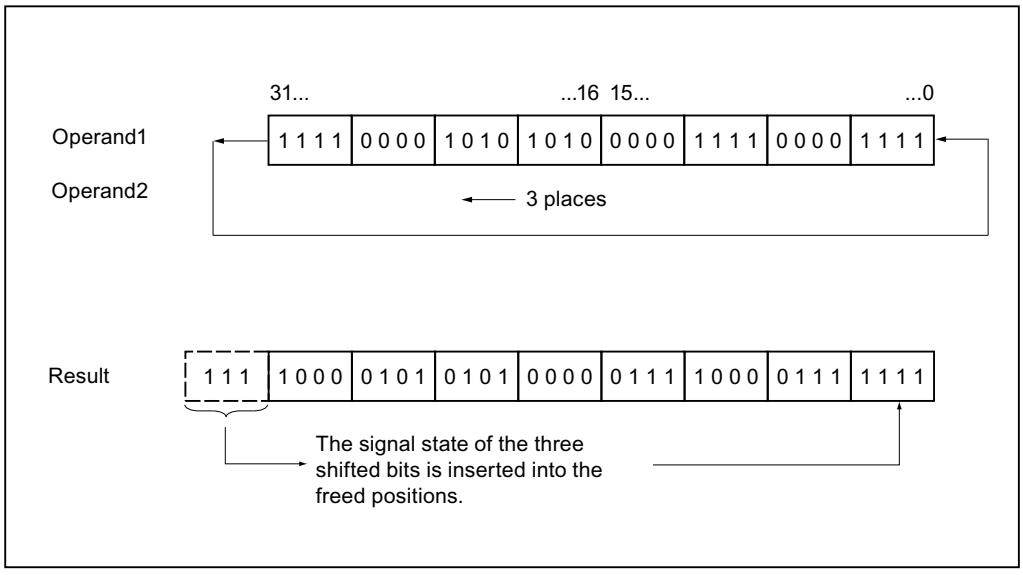

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L or constant	Value to be rotated
<Operand2>	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ROL("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1111 0000 1010 1010 0000 1111 0000 1111
Tag_Number	5
Tag_OutValue	0001 0101 0100 0001 1110 0001 1111 1110

The content of the "Tag_InValue" operand is rotated five bit positions to the left and output as the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

Legacy (S7-1500)

DRUM: Implement sequencer (S7-1500)

Description

You use the "Implement sequencer" instruction to assign the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD) with the programmed values of the OUT_VAL parameter of the relevant step. The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). When a bit of the configurable mask has a signal state of "1", the OUT_VAL value sets/resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L or constant	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Maximum step number (for example: LST_STEP = 16#08; a maximum of 8 steps is possible.)
EVENT(i) 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L or constant	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L or constant	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Preset first step of the sequencer (1 to 16)
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P or constant	Remaining processing time for the current step
DTBP	Static	WORD	I, Q, M, D, L, P or constant	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L or constant	System time of the previous call
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L or constant	Preset counter value for each step [1 to 16] where one clock pulse = 1 ms.

Parameter	Declaration	Data type	Memory area	Description
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L or constant	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value at the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value at the LST_STEP parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL DRUM, "DRUM_DB"
     (RESET := "Tag_Reset"
      JOG := "Tag_Input_Jog"
      DRUM_EN := "Tag_Input_Drum_EN"
      LST_STEP := "Tag_Number_LastStep"
      EVENTn := "MyTag_Event_n"
      OUTn => "MyTag_Output_n"
      Q => "Tag_Output_Q"
      OUT_WORD => "Tag_OutputWord"
      ERR_CODE => "Tag_ErrorCode"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameter	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE

Parameter	Address	Value
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE

Parameter	Operand	Address	Value
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

DCAT: Discrete control-timer alarm (S7-1500)

Description

You can use the "Discrete control-timer alarm" instruction to accumulate the time from the point at which theCMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET is < PT) OA and CA are influenced as follows:
  - The Q and CMD_HIS parameters are set to "1".
  - The ET, OA, and CA parameters are reset to "0".
- When the signal state at the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".
- When the signal state at the CMD and CMD_HIS parameters is "1" and the O_FB parameter is set to "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state at the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value at the CMD_HIS parameter is reset to the value of the CMD parameter.
- If the signal state of the CMD, CMD_HIS, and O_FB parameters is set to "1" and the C_FB parameter has the value "0", the signal state of the OA parameter is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.

- If the CMD, CMD_HIS, and C_FB parameters have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the ET parameter. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the CA parameter is reset to "1". If the value at the PT parameter is not exceeded, the CA parameter has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the CMD, CMD_HIS, and O_FB parameters have the signal state "0" and the C_FB parameter is set to "1", the CA parameter is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction supplies no error information.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L or constant	The signal state "0" indicates the "close" command. The signal state "1" indicates the "open" command.
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the CMD parameter
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms.
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms.
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
CMD_HIS	Static	BOOL	D, L or constant	CMD history bit

Example

In the following example, the CMD parameter changes from "0" to "1". After the execution of the instruction, the Q parameter is set to "1" and the two alarm outputs OA and CA have the signal state "0". The CMD_HIS parameter of the instance data block is set to the signal state "1" and the ET parameter is reset to "0".

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL DCAT, "DCAT_DB"
     (CMD := "Tag_Input_CMD"
      O_FB := "Tag_Input_O_FB"
      C_FB := "Tag_Input_C_FB"
      Q => "Tag_Output_Q"
      OA => "Tag_Output_OA"
      CA => "Tag_Output_CA"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the "DCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the "DCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

MCAT: Motor control-timer alarm (S7-1500)

Description

You use the "Motor control-timer alarm" instruction to accumulate the time from the point at which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open

Input parameters								Output parameters								
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore not possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
O_CMD	Input	BOOL	I, Q, M, D, L or constant	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L or constant	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L or constant	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing

Parameter	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
O_HIS	Static	BOOL	D, L or constant	"Open" history bit
C_HIS	Static	BOOL	D, L or constant	"Close" history bit

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL MCAT, "MCAT_DB"
     (O_CMD := "Tag_Input_O_CMD"
      C_CMD := "Tag_Input_C_CMD"
      S_CMD := "Tag_Input_S_CMD"
      O_FB := "Tag_Input_O_FB"
      C_FB := "Tag_Input_C_FB"
      OO => "Tag_OutputOpen"
      CO => "Tag_OutputClosed"
      OA => "Tag_Output_OA"
      CA => "Tag_Output_CA"
      Q => "Tag_Output_Q"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE

Parameter	Operand	Value
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

IMC: Compare input bits with the bits of a mask (S7-1500)

Description

You use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the CMP_VAL[x,0] mask, where "x" indicates the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise, the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

In the following example all 16 input bits are compared with the mask for step 2. The signal state of the OUT parameter is set to TRUE, since the input bits match the mask for step 2.

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL IMC, "IMC_DB"
     (IN_BIT0 - 15 := "Tag_Input_BITn"
      CMP_STEP := "Tag_CMP_STEP"
      OUT => "Tag_Output"
```

GRAPH

```
ERR_CODE => "Tag_ErrorCode"
)
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
CMP_STEP	Tag_CMP_STEP	B#16#02
OUT	Tag_Output	FALSE
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the "IMC_DB" instance data block of the instruction:

Parameter	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE

Parameter	Address	Value
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output	TRUE
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

SMC: Compare scan matrix (S7-1500)

Description

You use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the CMP_VAL[x,0] mask, where "x" indicates the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.

Parameter	Declaration	Data type	Memory area	Description
LAST	Static	BYTE	I, Q, M, D, L, P or constant	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

In this example, all 16 input bits are compared with the mask for steps 0 through 5 until a match is found. Only the masks for steps 0 through 2 are scanned, since the mask for step 2 matches the input bit.

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL SMC, "SMC_DB"
     (IN_BIT0 - 15 := "Tag_Input_BITn"
      OUT => "Tag_Output"
      OUT_STEP => "Tag_Output_STEP"
      ERR_CODE => "Tag_ErrorCode"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
OUT	Tag_Output	FALSE
OUT_STEP	Tag_Output_STEP	B#16#00
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the "SMC_DB" instance data block of the instruction:

Parameter	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE
LAST	DB84	B#16#05

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output	TRUE
OUT_STEP	Tag_Output_STEP	B#16#02
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

LEAD_LAG: Lead and lag algorithm (S7-1500)

Description

You use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and error information is output in the ERR_CODE parameter.

You use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two operations together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL LEAD_LAG, "LEAD_LAG_DB"
     (IN := "Tag_Input"
      SAMPLE_T := "Tag_Input_SAMPLE_T"
      OUT => "Tag_Output_Result"
      ERR_CODE => "Tag_ErrorCode"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the "LEAD_LAD_DB" instance data block of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

SEG: Create bit pattern for seven-segment display (S7-1500)

Description

The "Create bit pattern for seven-segment display" instruction converts each of the four hexadecimal digits in the specified source word (IN) into four equivalent codes for a 7-segment display and writes these to the double word of the output (OUT).

The instruction does not detect any error conditions.

The relationship between the input hexadecimal values and the output bit patterns is as follows:

Digit	-gfedcba	Display	Seven-segment display
0000	00111111	0	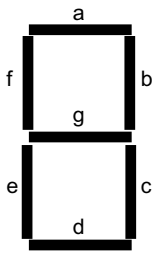
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Destination bit pattern with four bytes

Example

The following example shows how the instruction works:

GRAPH

```
CALL SEG
      (IN := "Tag_Input"
```

GRAPH

```
OUT => "Tag_Output"
)
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 Display: 1234

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

BCDCPL: Create tens complement (S7-1500)

Description

You use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:

GRAPH

```
CALL BCDCPL
      (IN := "Tag_Input"
       ERR_CODE => "Tag_Output"
       )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

BITSUM: Count number of set bits (S7-1500)

Description

You use the "Count number of set bits" instruction to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified in the IN parameter. The result of the instruction is output at the RET_VAL parameter.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

Example

The following example shows how the instruction works:

GRAPH

```
CALL BITSUM
      (IN := "Tag_Input"
       RET_VAL => "Tag_Output"
       )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 bits)
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Basic information on GRAPH (Page 7369)

GRAPH LAD instructions (S7-1500)

General LAD instructions in GRAPH (S7-1500)

Description

Different LAD instructions are available to you when programming in GRAPH.

The following table shows an overview of the instructions and their availability:

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
General		
Insert network	x	
Empty box	x	x
Open branch	x	x
Close branch	x	x
-|: Insert input	x	
Bit logic operations		
---| |---: Normally open contact (Page 599)	x	x
---| / |---: Normally closed contact (Page 601)	x	x
--|NOT|--: Invert RLO (Page 602)	x	x
---( )---: Assignment (Page 603)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
---(/)---: Negate assignment (Page 604)	x	
---(R)---: Reset output (Page 604)	x	
---(S)---: Set output (Page 605)	x	
SET_BF: Set bit field (Page 606)	x	
RESET_BF: Reset bit field (Page 607)	x	
SR: Set/reset flip-flop (Page 609)	x	
RS: Reset/set flip-flop (Page 610)	x	
--|P|--: Scan operand for positive signal edge (Page 611)	x	
--|N|--: Scan operand for negative signal edge (Page 613)	x	
--(P)--: Set operand on positive signal edge (Page 614)	x	
--(N)--: Set operand on negative signal edge (Page 616)	x	
P_TRIG: Scan RLO for positive signal edge (Page 617)	x	
N_TRIG: Scan RLO for negative signal edge (Page 618)	x	
R_TRIG: Detect positive signal edge (Page 619)	x	
F_TRIG: Detect negative signal edge (Page 620)	x	
Timers		
IEC timers		
TP: Generate pulse (Page 622)	x	
TON: Generate on-delay (Page 624)	x	
TOF: Generate off-delay (Page 627)	x	
TONR: Time accumulator (Page 630)	x	
---(TP)---: Start pulse timer (Page 633)	x	
---(TON)---: Start on-delay timer (Page 636)	x	
---(TOF)---: Start off-delay timer (Page 639)	x	
---(TONR)---: Time accumulator (Page 641)	x	
---(RT)---: Reset timer (Page 644)	x	
---(PT)---: Load time duration (Page 645)	x	
SIMATIC timers (S7-1500)		
S_PULSE: Assign pulse timer parameters and start (Page 647)	x	
S_PEXT: Assign extended pulse timer parameters and start (Page 650)	x	
S_ODT: Assign on-delay timer parameters and start (Page 653)	x	
S_ODTS: Assign retentive on-delay timer parameters and start (Page 655)	x	
S_OFFDT: Assign off-delay timer parameters and start (Page 658)	x	
Counters		
IEC counters		
CTU: Count up (Page 670)	x	
CTD: Count down (Page 672)	x	
CTUD: Count up and down (Page 675)	x	
SIMATIC counters (S7-1500)		
S_CU: Assign parameters and count up (Page 678)	x	
S_CD: Assign parameters and count down (Page 680)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
S_CUD: Assign parameters and count up / down (Page 682)	x	
Comparator operations		
CMP ==: Equal (Page 687)	x	x
CMP <>: Not equal (Page 693)	x	x
CMP >=: Greater or equal (Page 698)	x	x
CMP <=: Less or equal (Page 700)	x	x
CMP >: Greater than (Page 703)	x	x
CMP <: Less than (Page 705)	x	x
IN_RANGE: Value within range (Page 708)	x	
OUT_RANGE: Value outside range (Page 709)	x	
---| OK |---: Check validity (Page 711)	x	
---| NOT_OK |---: Check invalidity (Page 712)	x	
EQ_Type: Compare data type for EQUAL with the data type of a tag (Page 713)	x	
NE_Type: Compare data type for UNEQUAL with the data type of a tag (Page 714)	x	
EQ_ElemType: Compare data type of an ARRAY element for EQUAL with the data type of a tag (Page 716)	x	
NE_ElemType: Compare data type of an ARRAY element for UNEQUAL with the data type of a tag (Page 718)	x	
IS_NULL: Query for EQUALS ZERO pointer (Page 719)	x	
NOT_NULL: Query for UNEQUALS ZERO pointer (Page 721)	x	
IS_ARRAY: Check for ARRAY (Page 722)	x	
EQ_TypeOfDB: Compare DB_ANY data type with the data type of a tag for EQUAL (Page 723)	x	
NE_TypeOfDB: Compare DB_ANY data type with the data type of a tag for NOT EQUAL (Page 724)	x	
CMP>T: Greater than step activation time (Page 2072)		x
CMP>U: Greater than uninterrupted step activation time (Page 2073)		x
Math functions		
CALCULATE: Calculate (Page 726)	x	
ADD: Add (Page 729)	x	
SUB: Subtract (Page 730)	x	
MUL: Multiply (Page 732)	x	
DIV: Divide (Page 733)	x	
MOD: Return remainder of division (Page 734)	x	
NEG: Create twos complement (Page 736)	x	
INC: Increment (Page 737)	x	
DEC: Decrement (Page 738)	x	
ABS: Form absolute value (Page 739)	x	
MIN: Get minimum (Page 741)	x	
MAX: Get maximum (Page 742)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
LIMIT: Set limit value (Page 744)	x	
SQR: Form square (Page 746)	x	
SQRT: Form square root (Page 748)	x	
LN: Form natural logarithm (Page 749)	x	
EXP: Form exponential value (Page 750)	x	
SIN: Form sine value (Page 751)	x	
COS: Form cosine value (Page 753)	x	
TAN: Form tangent value (Page 754)	x	
ASIN: Form arcsine value (Page 755)	x	
ACOS: Form arccosine value (Page 757)	x	
ATAN: Form arctangent value (Page 758)	x	
FRAC: Return fraction (Page 759)		
EXPT: Exponentiate (Page 761)	x	
Move operations		
MOVE: Move value (Page 762)	x	
Deserialize: Deserialization (Page 767)	x	
Serialize: Serialization (Page 771)	x	
FieldRead: Read field (Page 831)	x	
FieldWrite: Write field (Page 833)	x	
MOVE_BLK: Move block (Page 776)	x	
SCATTER: Parse the bit sequence into individual bits (Page 792)	x	
SCATTER_BLK: Parse elements of an ARRAY of bit sequence into individual bits (Page 796)	x	
GATHER: Merge individual bits into a bit sequence (Page 800)	x	
GATHER_BLK: Merge individual bits into multiple elements of an ARRAY of bit sequence (Page 804)	x	
MOVE_BLK_VARIANT: Move block (Page 778)	x	
UMOVE_BLK: Move block uninterruptible (Page 783)	x	
FILL_BLK: Fill block (Page 785)	x	
UFILL_BLK: Fill block uninterruptible (Page 789)	x	
?= Assignment attempt (Page 808)	x	
SWAP: Swap (Page 811)	x	
ReadFromArrayDB: Read from the array data block (Page 812)	x	
WriteToArrayDB: Write to the array data block (Page 815)	x	
ReadFromArrayDBL: Read from the array data block in the load memory (Page 817)	x	
WriteToArrayDBL: Write to array data block in the load memory (Page 820)	x	
VariantGet: Read VARIANT tag value (Page 823)	x	
VariantPut: Write VARIANT tag value (Page 825)	x	
CountOfElements: Get number of ARRAY elements (Page 826)	x	
LOWER_BOUND: Read ARRAY low limit (Page 828)	x	
UPPER_BOUND: Read ARRAY high limit (Page 829)	x	

LAD instruction	Perma-nent in-struction	Transition/ Supervision/ Interlock
BLKMOV: Move block (S7-1500) (Page 835)	x	
UBLKMOV: Move block uninterruptible (S7-1500) (Page 838)	x	
FILL: Fill block (S7-1500) (Page 841)	x	
Conversion operations		
CONVERT: Convert value (Page 845)	x	
ROUND: Round numerical value (Page 847)	x	
CEIL: Generate next higher integer from floating-point number (Page 848)	x	
FLOOR: Generate next lower integer from floating-point number (Page 850)	x	
TRUNC: Truncate numerical value (Page 851)	x	
SCALE_X: Scale (Page 852)	x	
NORM_X: Normalize (Page 855)	x	
SCALE: Scale (S7-1500) (Page 857)	x	
UNSCALE: Unscale (S7-1500) (Page 859)	x	
Program control operations		
Runtime control		
ENDIS_PW: Limit and enable password legitimation (Page 871)	x	
RE_TRIGR: Restart cycle monitoring time (Page 877)	x	
STP: Exit program (Page 878)	x	
GET_ERROR: Get error locally (Page 879)	x	
GET_ERR_ID: Get error ID locally (Page 883)	x	
INIT_RD: Initialize all retain data (Page 886)	x	
WAIT: Configure time delay (Page 887)	x	
RUNTIME: Runtime measurement (Page 890)	x	
Word logic operations		
AND: AND logic operation (Page 892)	x	
OR: OR logic operation (Page 894)	x	
XOR: EXCLUSIVE OR logic operation (Page 895)	x	
INV: Create ones complement (Page 897)	x	
DECO: Decode (Page 898)	x	
ENCO: Encode (Page 899)	x	
SEL: Select (Page 901)	x	
MUX: Multiplex (Page 903)	x	
DEMUX: Demultiplex (Page 906)	x	
Shift and rotate		
SHR: Shift right (Page 909)	x	
SHL: Shift left (Page 911)	x	
ROR: Rotate right (Page 913)	x	
ROL: Rotate left (Page 915)	x	
Additional instructions		
DRUM: Implement sequencer (Page 917)	x	
DCAT: Discrete control-timer alarm (Page 923)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
MCAT: Motor control-timer alarm (Page 926)	x	
IMC: Compare input bits with the bits of the mask (Page 930)	x	
SMC: Compare scan matrix (Page 932)	x	
LEAD_LAG: Lead and lag algorithm (Page 934)	x	
SEG: Create bit pattern for seven-segment display (Page 937)	x	
BCDCPL: Create tens complement (Page 939)	x	
BITSUM: Count number of set bits (Page 940)	x	

CMP>T: Greater than step activation time (S7-1500)

Description

You use the "Greater than step activation time" instruction to monitor the total duration of a step in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >T: Greater than step activation time" (with data type STRING, WSTRING), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >T: Greater than step activation time" becomes inactive.

Result:

For the instruction "CMP >T: Greater than step activation time" (data type STRING, WSTRING), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >T: Greater than step activation time" (data type STRING, WSTRING) is displayed again. The instruction "CMP >T: Greater than step activation time" is shown as grayed out in the network in the inactive status.

Parameters

The following table shows the parameters of the "Greater than step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Current or last activation time of the step
<Operand2>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

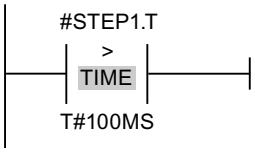

As long as the activation time of #STEP1.T is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

CMP>U: Greater than uninterrupted step activation time (S7-1500)

Description

You use the "Greater than uninterrupted step activation time" instruction to monitor the duration of a step, minus any faults, in supervision conditions. The times of any events or faults are not recorded and only the pure step duration is monitored.

In the programmed condition, the total activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled,

the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >U: Greater than uninterrupted step activation time" (with data type STRING, WSTRING), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >U: Greater than uninterrupted step activation time" becomes inactive.

Result:

For the instruction "CMP >U: Greater than uninterrupted step activation time" (data type STRING, WSTRING), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >U: Greater than uninterrupted step activation time" (data type STRING, WSTRING) is displayed again. The instruction "CMP >U: Greater than uninterrupted step activation time" is shown as grayed out in the network in the inactive status.

Parameters

The following table shows the parameters of the "Greater than uninterrupted step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Current or last activation time of the step minus the faults
<Operand2>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

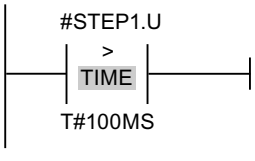

As long as the activation time of #STEP1.U minus possible faults is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

CMP>T_MAX: Greater than maximum step activation time (S7-1500)

Description

You use the "Greater than maximum step activation time" instruction to monitor the maximum total duration of a step in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with the maximum duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >T_MAX: Greater than maximum step activation time" (with data type STRING, WSTRING), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >T_MAX: Greater than maximum step activation time" becomes inactive.

Result:

For the instruction "CMP >T_MAX: Greater than maximum step activation time" (data type STRING, WSTRING), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >T_MAX: Greater than maximum step activation time" (data type STRING, WSTRING) is displayed again. The instruction "CMP >T_MAX: Greater than maximum step activation time" is shown as grayed out in the network in the inactive status.

Parameter

The following table shows the parameters of the "Greater than maximum step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Current or last activation time of the step
<Operand2>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

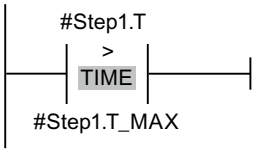

As long as the activation time of #STEP1.T is less than the maximum step activation time of #Step1.T_MAX, the RLO is "0". As soon as the maximum step activation time is exceeded, the RLO changes to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basics of step time monitoring (Page 7465)

Programming step time monitoring (Page 7466)

Basic information on GRAPH (Page 7369)

CMP>T_WARN: Greater than warning time (S7-1500)

Description

You use the "Greater than warning time" instruction to monitor the duration of a step and issue a warning when the time is exceeded in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1" and a warning is output. If the comparison condition is not fulfilled, the instruction returns RLO "0".

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >T_WARN: Greater than warning time" (with data type STRING, WSTRING), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >T_WARN: Greater than warning time" becomes inactive.

Result:

For the instruction "CMP >T_WARN: Greater than warning time" (data type STRING, WSTRING), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >T_WARN: Greater than warning time" (data type STRING, WSTRING) is displayed again. The instruction "CMP >T_WARN: Greater than warning time" is shown as grayed out in the network in the inactive status.

Parameters

The following table lists the parameters of the "Greater than warning time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Current or last activation time of the step
<Operand2>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

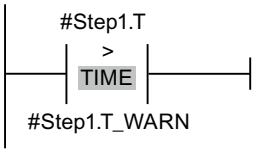

As long as the activation time of #Step1.T does not exceed the warning time of #Step1.T_WARN, the RLO is "0". As soon as the warning time is exceeded, the RLO changes to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basics of step time monitoring (Page 7465)

Programming step time monitoring (Page 7466)

Basic information on GRAPH (Page 7369)

GRAPH FBD instructions (S7-1500)

General FBD instructions in GRAPH (S7-1500)

Description

Different FBD instructions are available to you when programming in GRAPH.

The following table shows an overview of the instructions and their availability:

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
General		
New network	x	
Empty box	x	x
Branch: Open branch	x	x
-|: Insert input (Page 949)	x	x
-o|: Invert RLO (Page 950)	x	x
Bit logic operations		
&: AND logic operation (Page 942)	x	x
>=1: OR logic operation (Page 944)	x	x
X: EXCLUSIVE OR logic operation (Page 946)	x	
=: Assignment (Page 950)	x	
/=: Negate assignment (Page 951)	x	
R: Reset output (Page 952)	x	
S: Set output (Page 953)	x	
SET_BF: Set bit field (Page 954)	x	
RESET_BF: Reset bit field (Page 956)	x	
SR: Set/reset flip-flop (Page 957)	x	
RS: Reset/set flip-flop (Page 958)	x	
P: Scan operand for positive signal edge (Page 960)	x	

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
N: Scan operand for negative signal edge (Page 961)	x	
P=: Set operand on positive signal edge (Page 963)	x	
N=: Set operand on negative signal edge (Page 964)	x	
P_TRIG: Scan RLO for positive signal edge (Page 965)	x	
N_TRIG: Scan RLO for negative signal edge (Page 966)	x	
R_TRIG: Detect positive signal edge (Page 968)	x	
F_TRIG: Detect negative signal edge (Page 969)	x	
Timers		
IEC timers		
TP: Generate pulse (Page 970)	x	
TON: Generate on-delay (Page 973)	x	
TOF: Generate off-delay (Page 976)	x	
TONR: Time accumulator (Page 979)	x	
TP: Start pulse timer (Page 982)	x	
TON: Start on-delay timer (Page 985)	x	
TOF: Start off-delay timer (Page 987)	x	
TONR: Time accumulator (Page 990)	x	
RT: Reset timer (Page 993)	x	
PT: Load time duration (Page 994)	x	
SIMATIC Timers		
S_PULSE: Assign pulse timer parameters and start (Page 996)	x	
S_PEXT: Assign extended pulse timer parameters and start (Page 999)	x	
S_ODT: Assign on-delay timer parameters and start (Page 1002)	x	
S_ODTS: Assign retentive on-delay timer parameters and start (Page 1005)	x	
S_OFFDT: Assign off-delay timer parameters and start (Page 1007)	x	
Counters		
IEC counters		
CTU: Count up (Page 1019)	x	
CTD: Count down (Page 1022)	x	
CTUD: Count up and down (Page 1025)	x	
SIMATIC Counters		
S_CU: Assign parameters and count up (Page 1028)	x	
S_CD: Assign parameters and count down (Page 1030)	x	
S_CUD: Assign parameters and count up / down (Page 1032)	x	
Comparator operations		
CMP==: Equal (Page 1037)	x	x
CMP <>: Not equal (Page 1042)	x	x
CMP>=: Greater or equal (Page 1047)	x	x
CMP<=: Less or equal (Page 1049)	x	x
CMP>: Greater than (Page 1051)	x	x

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
CMP<: Less than (Page 1054)	x	x
IN_RANGE: Value within range (Page 1056)	x	
OUT_RANGE: Value outside range (Page 1057)	x	
---| OK |---: Check validity (Page 1059)	x	
---| NOT_OK |---: Check invalidity (Page 1060)	x	
EQ_Type: Compare data type for EQUAL with the data type of a tag (Page 1061)	x	
NE_Type: Compare data type for UNEQUAL with the data type of a tag (Page 1062)	x	
EQ_ElemType: Compare data type of an ARRAY element for EQUAL with the data type of a tag (Page 1064)	x	
NE_ElemType: Compare data type of an ARRAY element for UNEQUAL with the data type of a tag (Page 1066)	x	
IS_NULL: Query for EQUALS ZERO pointer (Page 1067)	x	
NOT_NULL: Query for UNEQUALS ZERO pointer (Page 1069)	x	
IS_ARRAY: Check for ARRAY (Page 1070)	x	
EQ_TypeOfDB: Compare DB_ANY data type with the data type of a tag for EQUAL (Page 1071)	x	
NE_TypeOfDB: Compare DB_ANY data type with the data type of a tag for NOT EQUAL (Page 1072)	x	
CMP>T: Greater than step activation time (Page 2083)		x
CMP>U: Greater than uninterrupted step activation time (Page 2084)		x
Math functions		
CALCULATE: Calculate (Page 1073)	x	
ADD: Add (Page 1077)	x	
SUB: Subtract (Page 1078)	x	
MUL: Multiply (Page 1079)	x	
DIV: Divide (Page 1081)	x	
MOD: Return remainder of division (Page 1082)	x	
NEG: Create twos complement (Page 1084)	x	
INC: Increment (Page 1085)	x	
DEC: Decrement (Page 1086)	x	
ABS: Form absolute value (Page 1087)	x	
MIN: Get minimum (Page 1089)	x	
MAX: Get maximum (Page 1090)	x	
LIMIT: Set limit value (Page 1092)	x	
SQR: Form square (Page 1094)	x	
SQRT: Form square root (Page 1096)	x	
LN: Form natural logarithm (Page 1097)	x	
EXP: Form exponential value (Page 1098)	x	
SIN: Form sine value (Page 1099)	x	
COS: Form cosine value (Page 1101)	x	

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
TAN: Form tangent value (Page 1102)	x	
ASIN: Form arcsine value (Page 1103)	x	
ACOS: Form arccosine value (Page 1105)	x	
ATAN: Form arctangent value (Page 1106)	x	
FRAC: Return fraction (Page 1107)	x	
EXPT: Exponentiate (Page 1109)	x	
Move operations		
MOVE: Move value (Page 1110)	x	
Deserialize: Deserialization (Page 1116)	x	
Serialize: Serialization (Page 1120)	x	
FieldRead: Read field (Page 1179)	x	
FieldWrite: Write field (Page 1181)	x	
MOVE_BLK: Move block (Page 1125)	x	
SCATTER: Parse the bit sequence into individual bits (Page 1142)	x	
SCATTER_BLK: Parse elements of an ARRAY of bit sequence into individual bits (Page 1145)	x	
GATHER: Merge individual bits into a bit sequence (Page 1149)	x	
GATHER_BLK: Merge individual bits into multiple elements of an ARRAY of bit sequence (Page 1153)	x	
MOVE_BLK_VARIANT: Move block (Page 1127)	x	
UMOVE_BLK: Move block uninterruptible (Page 1131)	x	
FILL_BLK: Fill block (Page 1134)	x	
UFILL_BLK: Fill block uninterruptible (Page 1138)	x	
?= Assignment attempt (Page 1157)	x	
SWAP: Swap (Page 1159)	x	
ReadFromArrayDB: Read from the array data block (Page 1161)	x	
WriteToArrayDB: Write to the array data block (Page 1164)	x	
ReadFromArrayDBL: Read from the array data block in the load memory (Page 1166)	x	
WriteToArrayDBL: Write to array data block in the load memory (Page 1169)	x	
VariantGet: Read VARIANT tag value (Page 1172)	x	
VariantPut: Write VARIANT tag value (Page 1173)	x	
CountOfElements: Get number of ARRAY elements (Page 1175)	x	
LOWER_BOUND: Read ARRAY low limit (Page 1176)	x	
UPPER_BOUND: Read ARRAY high limit (Page 1178)	x	
BLKMOV: Move block (S7-1500) (Page 1183)	x	
UBLKMOV: Move block uninterruptible (S7-1500) (Page 1186)	x	
FILL: Fill block (S7-1500) (Page 1189)	x	
Conversion operations		
CONVERT: Convert value (Page 1192)	x	
ROUND: Round numerical value (Page 1194)	x	

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
CEIL: Generate next higher integer from floating-point number (Page 1196)	x	
FLOOR: Generate next lower integer from floating-point number (Page 1197)	x	
TRUNC: Truncate numerical value (Page 1199)	x	
SCALE_X: Scale (Page 1200)	x	
NORM_X: Normalize (Page 1202)	x	
VARIANT_TO_DB_ANY: Convert VARIANT to DB_ANY	x	
DB_ANY_TO_VARIANT: Convert DB_ANY to VARIANT	x	
SCALE: Scale (S7-1500) (Page 1205)	x	
UNSCALE: Unscale (S7-1500) (Page 1207)	x	
Program control operations		
Runtime control		
ENDIS_PW: Limit and enable password legitimation (Page 1218)	x	
RE_TRIGR: Restart cycle monitoring time (Page 1224)	x	
STP: Exit program (Page 1225)	x	
GET_ERROR: Get error locally (Page 1226)	x	
GET_ERR_ID: Get error ID locally (Page 1230)	x	
INIT_RD: Initialize all retain data (Page 1233)	x	
WAIT: Configure time delay (Page 1234)	x	
RUNTIME: Runtime measurement (Page 1236)	x	
Word logic operations		
AND: AND logic operation (Page 1239)	x	
OR: OR logic operation (Page 1240)	x	
XOR: EXCLUSIVE OR logic operation (Page 1242)	x	
INV: Create ones complement (Page 1243)	x	
DECO: Decode (Page 1245)	x	
ENCO: Encode (Page 1246)	x	
SEL: Select (Page 1247)	x	
MUX: Multiplex (Page 1249)	x	
DEMUX: Demultiplex (Page 1252)	x	
Shift and rotate		
SHR: Shift right (Page 1255)	x	
SHL: Shift left (Page 1257)	x	
ROR: Rotate right (Page 1259)	x	
ROL: Rotate left (Page 1261)	x	
Additional instructions		
DRUM: Implement sequencer (Page 1263)	x	
DCAT: Discrete control-timer alarm (Page 1269)	x	
MCAT: Motor control-timer alarm (Page 1273)	x	
IMC: Compare input bits with the bits of the mask (Page 1276)	x	
SMC: Compare scan matrix (Page 1278)	x	

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
LEAD_LAG: Lead and lag algorithm (Page 1280)	x	
SEG: Create bit pattern for seven-segment display (Page 1283)	x	
BCDCPL: Create tens complement (Page 1285)	x	
BITSUM: Count number of set bits (Page 1286)	x	

CMP>T: Greater than step activation time (S7-1500)

Description

You can use the "Greater than step activation time" instruction to monitor the total duration of a step in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >T: Greater than step activation time" (with data type STRING, WSTRING), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >T: Greater than step activation time" becomes inactive.

Result:

For the instruction "CMP >T: Greater than step activation time" (data type STRING, WSTRING), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >T: Greater than step activation time" (data type STRING, WSTRING) is displayed again. The instruction "CMP >T: Greater than step activation time" is shown as grayed out in the network in the inactive status.

Parameters

The following table shows the parameters of the "Greater than step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Current or last activation time of the step
IN2	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

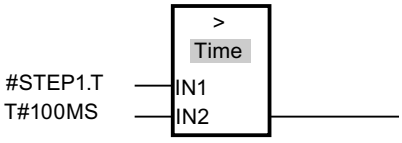

As long as the activation time of #STEP1.T is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

CMP>U: Greater than uninterrupted step activation time (S7-1500)

Description

You use the "Greater than uninterrupted step activation time" instruction to monitor the duration of a step, minus any faults, in supervision conditions. The times of any events or faults are not recorded and only the pure step duration is monitored.

In the programmed condition, the total activation time of a step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled,

the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >U: Greater than uninterrupted step activation time" (with data type STRING, WSTRING), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >U: Greater than uninterrupted step activation time" becomes inactive.

Result:

For the instruction "CMP >U: Greater than uninterrupted step activation time" (data type STRING, WSTRING), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >U: Greater than uninterrupted step activation time" (data type STRING, WSTRING) is displayed again. The instruction "CMP >U: Greater than uninterrupted step activation time" is shown as grayed out in the network in the inactive status.

Parameters

The following table shows the parameters of the "Greater than uninterrupted step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Current or last activation time of the step minus the faults
IN2	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

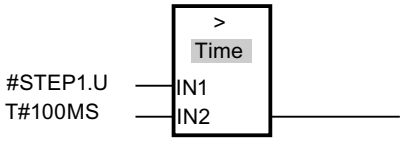

As long as the activation time of #STEP1.U minus possible faults is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basic information on GRAPH (Page 7369)

CMP>T_MAX: Greater than maximum step activation time (S7-1500)

Description

You use the "Greater than maximum step activation time" instruction to monitor the maximum total duration of a step in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with the maximum duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >T_MAX: Greater than maximum step activation time" (with data type STRING, WSTRING), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >T_MAX: Greater than maximum step activation time" becomes inactive.

Result:

For the instruction "CMP >T_MAX: Greater than maximum step activation time" (data type STRING, WSTRING), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >T_MAX: Greater than maximum step activation time" (data type STRING, WSTRING) is displayed again. The instruction "CMP >T_MAX: Greater than maximum step activation time" is shown as grayed out in the network in the inactive status.

Parameters

The following table shows the parameters of the "Greater than maximum step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Current or last activation time of the step
IN2	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

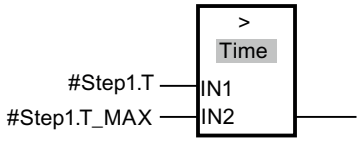

As long as the activation time of #STEP1.T is less than the maximum step activation time of #Step1.T_MAX, the RLO is "0". As soon as the maximum step activation time is exceeded, the RLO changes to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basics of step time monitoring (Page 7465)

Programming step time monitoring (Page 7466)

Basic information on GRAPH (Page 7369)

CMP>T_WARN: Greater than warning time (S7-1500)

Description

You use the "Greater than warning time" instruction to monitor the duration of a step and issue a warning when the time is exceeded in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the

comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1" and a warning is output. If the comparison condition is not fulfilled, the instruction returns RLO "0".

Note

Status still displayed despite "inactive" instruction

Under the following requirements:

- Prior to the instruction "CMP >T_WARN: Greater than warning time" (with data type STRING, WSTRING), a condition (e.g. of a NO contact) is queried in the network.
- "Monitoring on" is switched on.
- A new result of the condition resets the network to FALSE. The instruction "CMP >T_WARN: Greater than warning time" becomes inactive.

Result:

For the instruction "CMP >T_WARN: Greater than warning time" (data type STRING, WSTRING), the previous status is still shown in the network.

When you switch "Monitoring on" off and on again or scroll to a different network, the correct status of the instruction "CMP >T_WARN: Greater than warning time" (data type STRING, WSTRING) is displayed again. The instruction "CMP >T_WARN: Greater than warning time" is shown as grayed out in the network in the inactive status.

Parameters

The following table lists the parameters of the "Greater than warning time" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Current or last activation time of the step
IN2	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

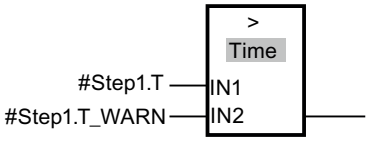

As long as the activation time of #Step1.T does not exceed the warning time of #Step1.T_WARN, the RLO is "0". As soon as the warning time is exceeded, the RLO changes to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 186)

Basics of step time monitoring (Page 7465)

Programming step time monitoring (Page 7466)

Basic information on GRAPH (Page 7369)

4.1.4 Extended instructions

4.1.4.1 Date and time-of-day

T_COMP: Compare time tags (S7-1500)

Description

This instruction is used to compare the contents of two tags of the "Timers" or "Date and time" data type.

The instruction supports the comparison of the following data types: DATE, TIME, LTIME, TOD (TIME_OF_DAY), LTOD (LTIME_OF_DAY), DT (DATE_AND_TIME), LDT (DATE_AND_LTIME), DTL, S5Time.

The data types must be the same length and format to carry out the comparison.

The comparison result is output at the OUT parameter as a return value. For this purpose, the parameter OUT is set to "1" if the condition used for the comparison is satisfied.

The following comparison options are available:

Symbol	Description
EQ	The return value has the signal state “1" if the time is the same at the parameters IN1 and IN2.
NE	The return value has the signal state “1" if the time at the parameter IN1 is not the same as that at the parameter IN2.
GE	The return value has the signal state "1" if the time at parameter IN1 is greater (more recent) than or equal to the time at parameter IN2 .
LE	The return value has the signal state "1" if the time at parameter IN1 is smaller (less recent) than or equal to the time at parameter IN2.
GT	The return value has the signal state "1" if the time at parameter IN1 is greater (more recent) than the time at parameter IN2.
LT	The return value has the signal state "1" if the time at parameter IN1 is smaller (older) than the time at parameter IN2 .

Parameters

The following table shows the parameters of the "T_COMP" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DATE, TIME, LTIME, TOD, LTOD, DT, LDT, DTL, S5Time	I, Q, M, D, L, P or constant	First value to be compared.
IN2	Input	DATE, TIME, LTIME, TOD, LTOD, DT, LDT, DTL, S5Time	I, Q, M, D, L, P or constant	Second value to be compared.
OUT	Output	BOOL	I, Q, M, D, L, P	Return value

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you use the "Greater or equal" comparison option to compare two times with the LTIME data type.

Create three tags in a global data block for storing the data.

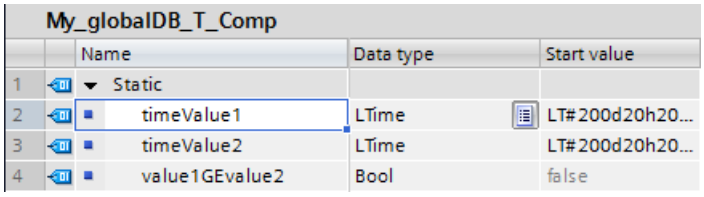

Interconnect the parameters of the instruction as follows. Select the "GE" comparison option.

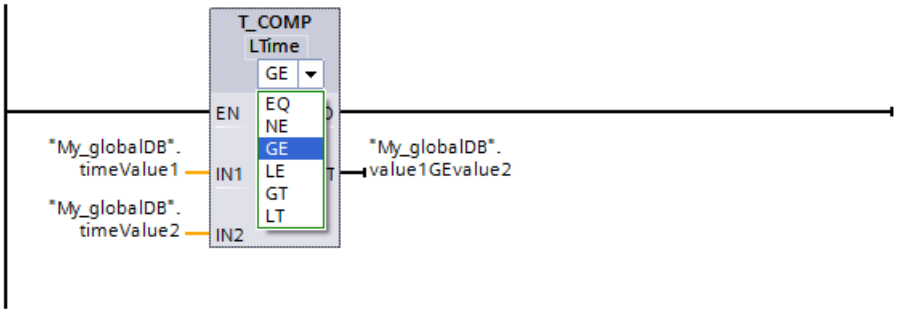

Since the time of the first value to be compared ("timeValue1") is greater than or equal to the second value ("timeValue2"), the return value ("value1GEvalue2") shows the signal state "TRUE".

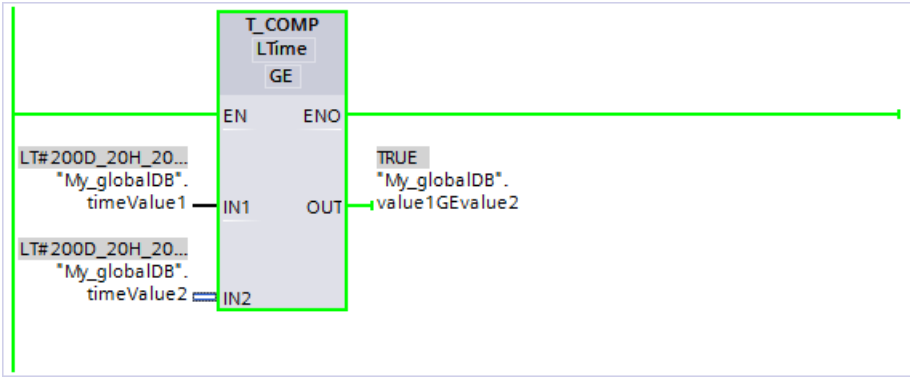

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

T_CONV: Convert times and extract (S7-1200, S7-1500)

Description

You use the instruction "T_CONV" to convert the data type of the IN input parameter to the data type that is output at the OUT output. You select the data formats for the conversion from the instruction boxes of the input and output.

Parameters

The following table shows the parameters of the instruction "T_CONV". If an input and output parameter of the same data type is used, the instruction copies the corresponding value.

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Integers, TIME, date and time*	WORD, integers, timers, date and time*	I, Q, M, D, L, P or constant	Value to be converted
OUT	Return	Integers, TIME, date and time*	WORD, integers, timers, date and time*	I, Q, M, D, L, P	Result of the conversion
* The range of supported data types depends on the CPU. Please refer to the overviews of valid data types for information on the data types supported by the S7-1200 and S7-1500 modules.					

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you convert a time with the DATE AND TIME data type to a time with the LTIME OF DAY data type.

Create two tags in a global data block for storing the data.

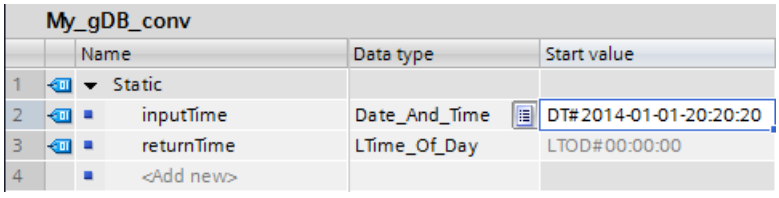

Interconnect the parameters of the instruction as follows and select the data types.

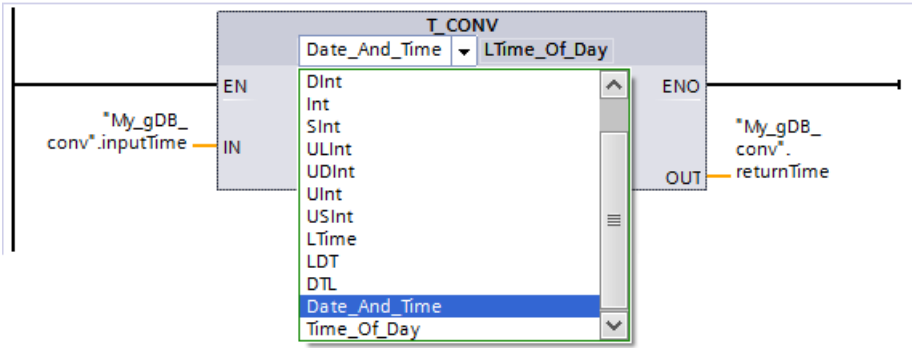

The value to be converted ("inputTime") is output at the output parameter as a new value ("returnTime"). The information for the date is lost.

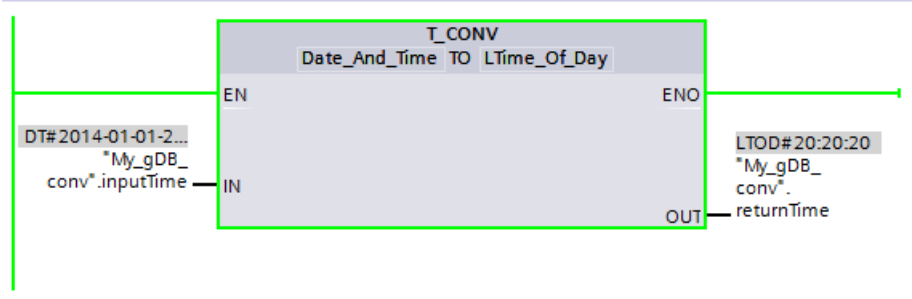

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

T_ADD: Add times (S7-1200, S7-1500)

Description

You use this instruction to add the time information in the IN1 input to the time information in the IN2 input. You can query the result in the OUToutput parameter. You can add the following formats:

- Addition of a time period to another time period.
  Example: Addition of a TIME data type to another TIME data type.
- Addition of a time period to a time.
  Example: Addition of a TIME data type to the DTL data type.

The data type for the value at input parameter IN1 and output parameter OUT is defined by the selection in the instruction boxes of the input and output. You can only specify time information in TIME format in the IN2 input parameter (for S7-1500 modules, also LTIME).

Note

Overflow/underflow

The largest possible value of the OUT output parameter may be exceeded or the smallest possible value undershot during the calculation. Evaluate the enable output ENO to detect this.

Parameters

The following tables show the parameters of the "T_ADD" instruction, according to the possible conversions:

Table 4-7 Addition of a time period to another time period

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Second number to be added
OUT	Return	DINT, DWORD, TIME, TOD	TIME, LTIME,	I, Q, M, D, L, P	Result of addition The data type selection depends on the data types selected for the IN1 and IN2 input parameters.

Table 4-8 Addition of a time period to a time

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DTL, TOD	DT, TOD, LTOD, LDT, DTL	I, Q, M, D, L, P or constant	First number to be added For LTIME at parameter IN2, only LTOD, LDT or DTL can be used.
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Second number to be added
OUT	Return	DINT, DWORD, TIME, TOD, UDINT, DTL	DT, DTL, LDT, TOD, LTOD	I, Q, M, D, L, P	Result of addition The data type selection depends on the data types selected for the IN1 and IN2 input parameters.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you add a time duration with the TIME data type to a time of day with the TOD data type.

Create three tags in a global data block for storing the data.

My_gDB_t_add			
	Name	Data type	Start value
1	Static		
2	timeValTOD	Time_Of_Day	TOD#20:20:20
3	timeValTIME	Time	T#60ms
4	valueTimeResult	Time_Of_Day	TOD#00:00:00

Interconnect the parameters of the instruction as follows. You select the data types for time of day ("timeValTOD") and time period ("timeValTIME").

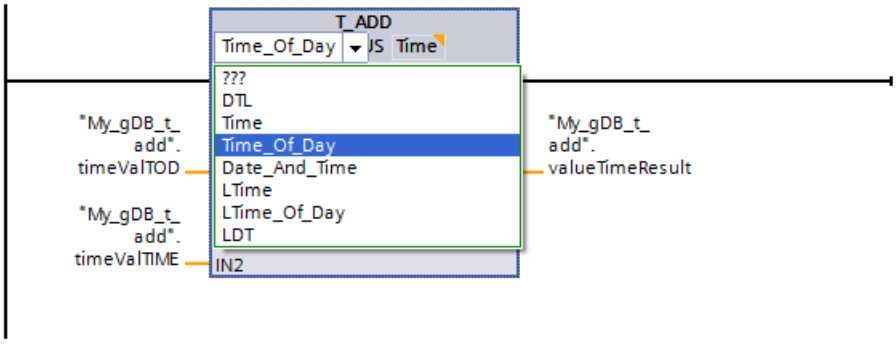

The time of day ("timeValTOD") and the time period ("timeValTIME") are added and the result is displayed as a time of day at output parameter OUT ("valueTimeResult").

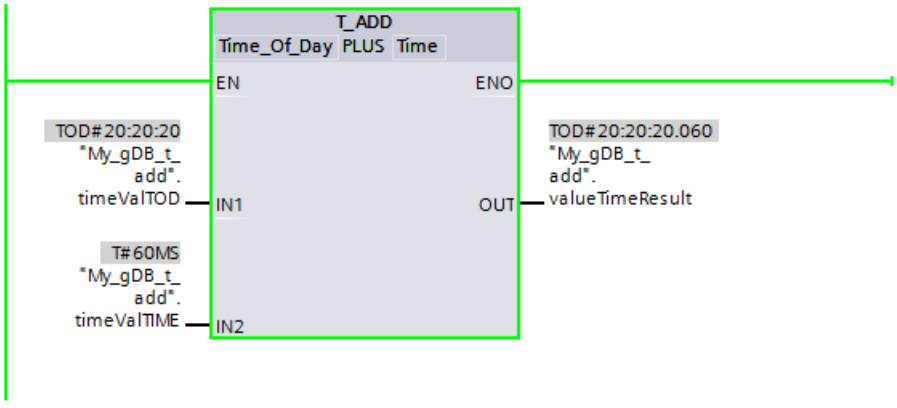

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

T_SUB: Subtract times (S7-1200, S7-1500)

Description

You use this instruction to subtract the time information in the IN2 input parameter from the time information in the IN1 input parameter. You can query the difference at the OUT output parameter. You can subtract the following formats:

- Subtraction of a time period from another time period
  Example: Subtraction of a time period of the data type TIME from another time period of the data type TIME. The result can be output to a tag with the TIME format.
- Subtraction of a time period from a time
  Example: Subtraction of a time period of the data type TIME from a time of the data type DTL. The result can be output to a tag with the DTL format.

You decide the formats of the values in the IN1 input parameter and the OUT output parameter by selecting the data types for the input and output parameters of the instruction.

Note

Overflow/underflow

The largest possible value of the OUT output parameter may be exceeded or the smallest possible value undershot during the calculation. Evaluate the enable output ENO to detect this.

Parameters

The following tables show the parameters of the "T_SUB" instruction, according to the possible conversions:

Table 4-9 Subtraction of a time period from another time period

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Minuend
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Subtrahend
OUT	Return	DINT, DWORD, TIME, TOD, UDINT	TIME, LTIME	I, Q, M, D, L, P	Result of subtraction

Table 4-10 Subtraction of a time period from a time

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DTL, TOD	TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Minuend For LTIME at parameter IN2, only LTOD, LDT or DTL are used.
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Subtrahend
OUT	Return	DTL, DINT, DWORD, TIME, TOD, UDINT	TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P	Result of subtraction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you subtract a time period with the TIME data type from a time of day with the TOD data type.

Create three tags in a global data block for storing the data.

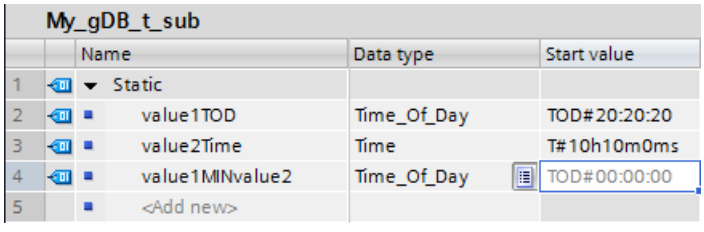

Interconnect the parameters of the instruction as follows. You select the data type for the time of day ("value1TOD") and the data type for the time period ("value2Time").

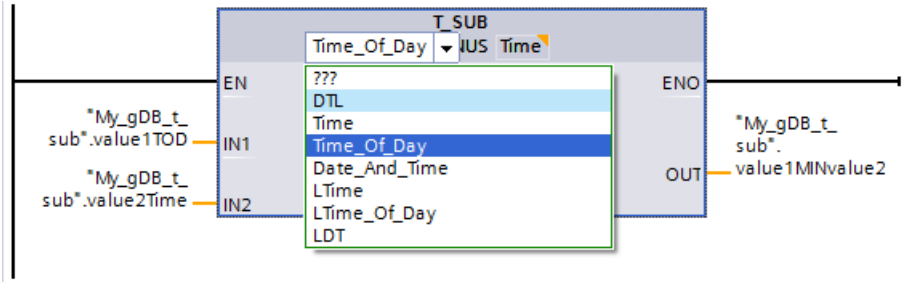

The time of day ("value1TOD") and the time period ("value2Time") are subtracted and the result is displayed as a time of day at output parameter OUT ("value1MINvalue2").

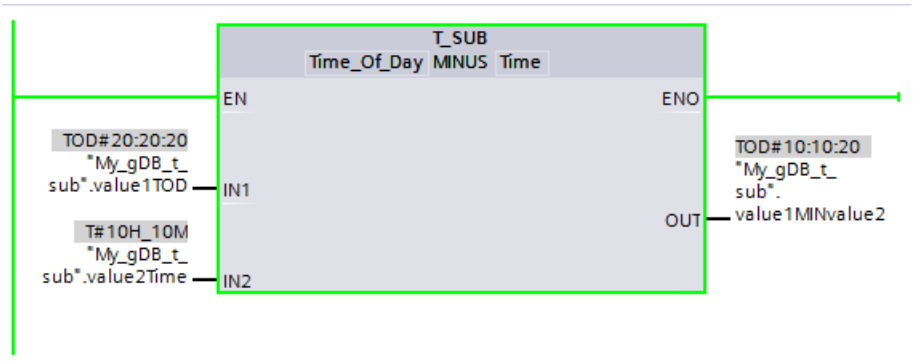

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

T_DIFF: Time difference (S7-1200, S7-1500)

Description

You use this instruction to subtract the time information in the IN2 input parameter from the time information in the IN1 input parameter. The result is sent at output parameter OUT.

- If the time information at the IN2 input parameter is greater than the time information at the IN1 input parameter, the result is output as a negative value at the OUT output parameter.
- If the result of the subtraction is outside the TIME number range, the enable output ENO is = "0". Depending on the data types used, you get either a truncated result or "0" (0:00).
  - If you select the minuend and subtrahend parameters of the DTL data type, the result has the TIME data type. It may not be greater than 24 days, otherwise the enable output ENO = "0" and the result is "0".
  - If you select the minuend and subtrahend parameters of the LDT data type, this restriction is avoided.

Parameters

The following table shows the parameters of the "T_DIFF" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DTL, DATE, TOD	DTL, DATE, DT, TOD, LTOD, LDT	I, Q, M, D, L, P or constant	Minuend
IN2	Input	DTL, DATE, TOD	DTL, DATE, DT, TOD, LTOD, LDT	I, Q, M, D, L, P or constant	Subtrahend
OUT	Return	TIME, INT	TIME, LTIME, INT	I, Q, M, D, L, P	Difference of input parameters

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

Use of the "T_DIFF instruction in SCL

In SCL, the data type TIME is preset for the OUT parameter (see also: Basic information on the data types of SCL instructions (Page 7339) If you want to use the data type INT for OUT, you need to extend the instruction name T_DIFF by adding "_INT" (see also: Changing the data type of an SCL instruction (Page 7340)).

Example

In the following example, you calculate the difference between two times of day with the TOD data type. You specify the difference in the "TIME" data type.

Create three tags in a global data block for storing the data.

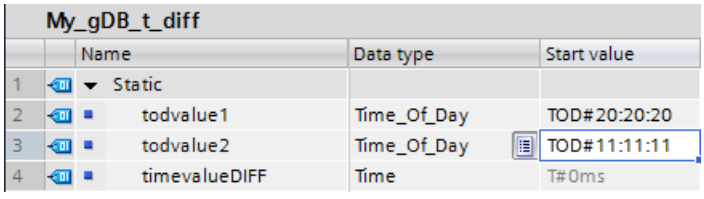

My_gDB_t_diff

	Name	Data type	Start value
1	Static		
2	todvalue1	Time_Of_Day	TOD#20:20:20
3	todvalue2	Time_Of_Day	TOD#11:11:11
4	timevalueDIFF	Time	T#0ms

Interconnect the parameters of the instruction as follows and select the data types. With the first selection option, you specify the data type of the points in time. With the second selection option, you specify the data type of the difference.

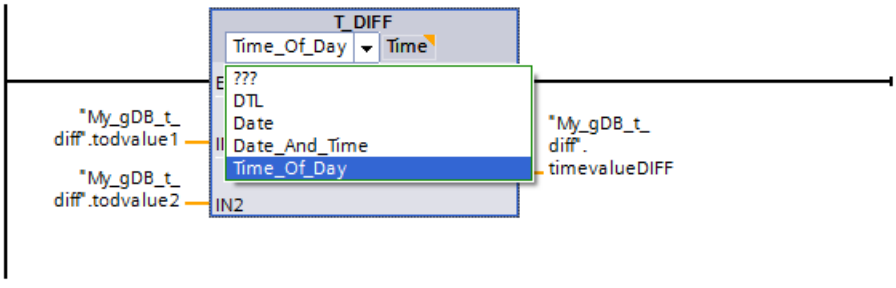

The first time of day ("todvalue1") and the second time of day ("todvalue2") are subtracted and the difference is displayed as a time period at output parameter OUT ("timevalueDIFF").

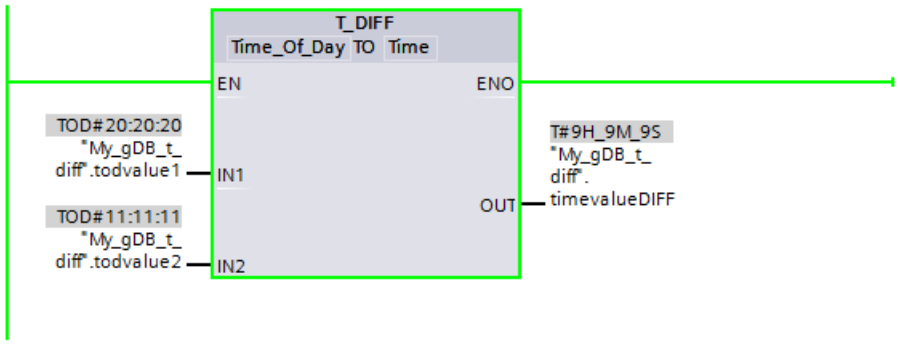

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

T_COMBINE: Combine times (S7-1200, S7-1500)

Description

The instruction combines the value of a date with the value of a time, resulting in a combined date and time value.

- The date is input at the input parameter IN1. Use a value between 1990-01-01 and 2089-12-31 for the data type DATE (this is not checked).
- The time is input at the IN2 input value (TOD/LTOD data type).
- The combined data type for the date and time value is output at the OUT output value.

Parameters

The following table shows the parameters of the "T_COMBINE" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DATE	DATE	I, Q, M, D, L, P or constant	Input tag of the date
IN2	Input	TOD	TOD, LTOD	I, Q, M, D, L, P or constant	Input tag of the time
OUT	Return	DTL	DT, DTL, LDT	I, Q, M, D, L, P	Return value of the date and time

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

Use of T_COMBINE in SCL programs

When you drag the instruction "T_COMBINE" from the "Instructions" task card to the SCL programming window, the internal instruction CONCAT_DATE_TOD or CONCAT_DATE_LTOD is displayed there.

Example

In the following example, you combine a time of day with the TOD data type and a date with the DATE data type. You specify the return value in the DT data type.

Create three tags in a global data block for storing the data.

My_gDB_t_comb			
	Name	Data type	Start value
1	Static		
2	valueDATE	Date	D#2014-01-01
3	valueTOD	Time_Of_Day	TOD#20:22:20
4	combTIME	Date_And_Time	DT#1990-01-01-00:00:00

Interconnect the parameters of the instruction as follows and select the data types. With the first selection option, you specify the data type of the time of day ("valueTOD"). With the second selection option, you specify the data type of the return value ("combTIME").

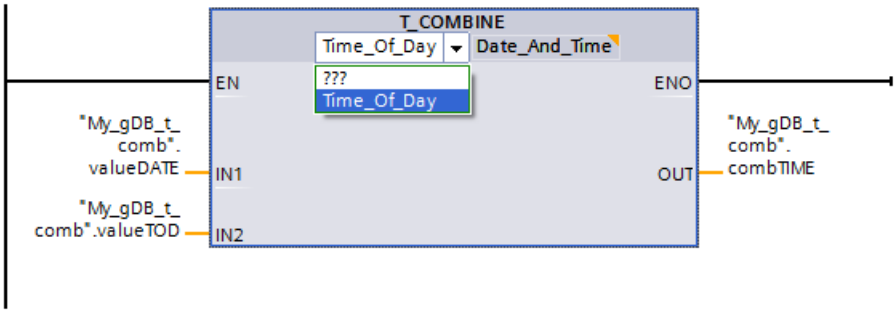

The date ("valueDATE") is expanded by the specification of the time of day ("valueTOD") and the return value is displayed at output parameter OUT ("combTIME").

My_gDB_t_comb				
	Name	Data type	Start value	Monitor value
1	Static			
2	valueDATE	Date	D#2014-01-01	D#2014-1-1
3	valueTOD	Time_Of_Day	TOD#20:22:20	TOD#20:22:20
4	combTIME	Date_And_Time	DT#1990-01-01-0	DT#2014-01-01-20:22:20

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

Time-of-day functions

WR_SYS_T: Set time-of-day (S7-1200, S7-1500)

Description

You use this instruction to set the date and time-of-day (module time) of the CPU clock. Enter the date and time-of-day at the input parameter IN . The value must be in the following range:

- With DT: min. DT#1990-01-01-0:0:0, max. DT#2089-12-31-23:59:59.999
- With LDT: min. LDT#1970-01-01-0:0:0.000000000, max. LDT#2200-12-31-23:59:59.999999999
- With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31-23:59:59.999999999

You can query whether errors have occurred during execution of the instruction in the RET_VAL output parameter.

The "WR_SYS_T" instruction is not used to pass information about the local time zone or daylight saving time.

Note

Module time of CPU clock

The CPU clock interprets the module time as the coordinated universal time (UTC). Accordingly, the module time is always stored without the factors "local time zone" or "summer time" in the CPU clock. The CPU clock then calculates the local time of the CPU clock based on the module time.

The module time of the CPU clock is used as template for all time processes starting from the CPU. Examples of use:

- Calculation of local time of CPU clock based on module time
- Representation of the module time in local time under "Online & Diagnostics"
- Block properties
- Entries in the diagnostics buffer of the CPU

Parameter

The following table shows the parameters of the "WR_SYS_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	DTL	DT **, DTL, LDT	I, Q, M, D, L, P or constant *	Date and time
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Status of the instruction
* The data types DT and DTL are not used for the following memory areas: input, output and bit memory. ** When using the data type DT the milliseconds are not transferred to the CPU.					

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0000	No error
8080	Error in date
8081	Error in time
8082**	Invalid value specified for the month (byte 2 in DTL format).
8083**	Invalid value specified for the day (byte 3 in DTL format).
8084**	Invalid value specified for the hour (byte 5 in DTL format).
8085**	Invalid value specified for the minute (byte 6 in DTL format).
8086**	Invalid value specified for the second (byte 7 in DTL format).
8087**	Invalid value specified for the nanosecond (bytes 8 to 11 in DTL format).
80B0	The real-time clock has failed
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	
** Only if the date and time are specified at the parameter IN in DTL format	

Example

In the following example, you set the date and time of the CPU clock. The data type used is DATE AND TIME.

Create three tags in a global data block for storing the data.

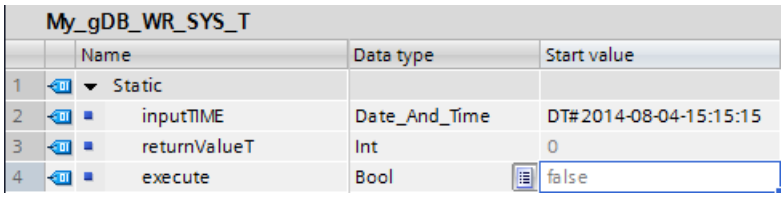

Interconnect the parameters of the instruction as follows. Select the DATE AND TIME data type.

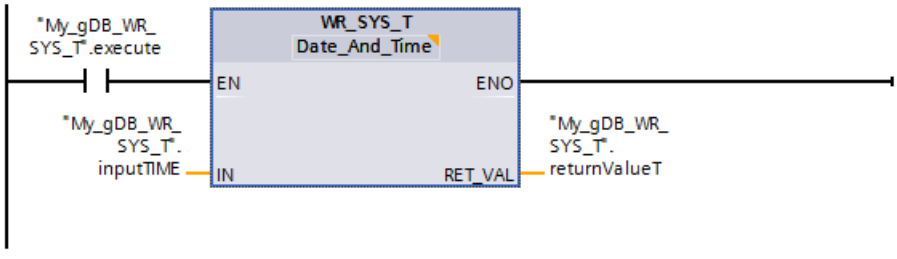

If the normally open contact ("execute") supplies the signal state "TRUE", the "WR_SYS_T" instruction is executed. The module time of the CPU clock is overwritten with the time to be set ("inputTIME"). The output parameter RET_VAL ("returnValueT") indicates that processing took place without errors.

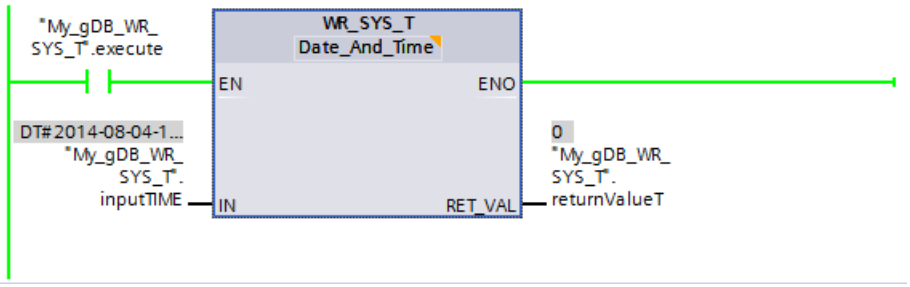

You can establish whether the new module time ("inputTIME") has been correctly received by the CPU clock as follows:

- Using the display of an S7-1500 CPU: Navigate on the CPU display to "Settings > Date & Time > General".
- Using TIA Portal: Read out the module time of the CPU clock using the "RD_SYS_T (Page 2103)" instruction.
- Using TIA Portal: Navigate to the "Online & Diagnostics" entry of the CPU, and open the "Functions > Set time of day" tab.
  The Coordinated Universal Time (UTC) is set for the module time of the CPU clock. Central European Time is set as local time in the TIA Portal in this example. Accordingly, an hour is added to the time to be set ("inputTIME") in the "Online & Diagnostics" entry in the TIA Portal. Another hour is added when daylight saving time is used. The calculated local time is output in 12-hour clock format.

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

See also

Switching display formats in the program status (Page 7698)

RD_SYS_T: Read time-of-day (S7-1200, S7-1500)

Description

You use this instruction to read the current date and current time-of-day (module time) of the CPU clock.

The read dates are output at the OUT output parameter of the instruction. The provided value does not include information about the local time zone or daylight saving time.

Note

Module time of CPU clock

The CPU clock interprets the module time as the coordinated universal time (UTC). Accordingly, the module time is always stored without the factors "local time zone" or "summer time" in the CPU clock. The CPU clock then calculates the local time of the CPU clock based on the module time.

The module time of the CPU clock is used as template for all time processes starting from the CPU. Examples of use:

- Calculation of local time of CPU clock based on module time
- Representation of the module time in local time under "Online & Diagnostics"
- Block properties
- Entries in the diagnostics buffer of the CPU

You can query whether errors have occurred during execution of the instruction in the RET_VALoutput.

Parameters

The following table shows the parameters of the "RD_SYS_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Status of the instruction
OUT	Output	DTL	DT, DTL, LDT	I, Q, M, D, L, P *	Date and time of CPU
* The data types DT and DTL cannot be used for the memory areas: input, output and bit memory.					

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0000	No error
8081	The data read out cannot be saved because it is outside the permissible range for the selected data type of OUT parameter. • With DT: min. DT#1990-01-01-0:0:0, max. DT#2089-12-31-23:59:59.999 • With LDT: min. LDT#1970-01-01-0:0:0.000000000, max. LDT#2262-04-11-23:47:16.854775807 • With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2262-04-11-23:47:16.854775807
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

In the following example, you read out the module time of the CPU clock. The data type used is DATE AND TIME.

Create two tags in a global data block for storing the data.

My_gDB_RD_SYS_T

	Name	Data type	Start value
1	Static		
2	outputTIME	Date_And_Time	DT#1990-01-01-00:00:00
3	returnValue	Int	0

Interconnect the parameters of the instruction as follows. Select the DATE AND TIME data type.

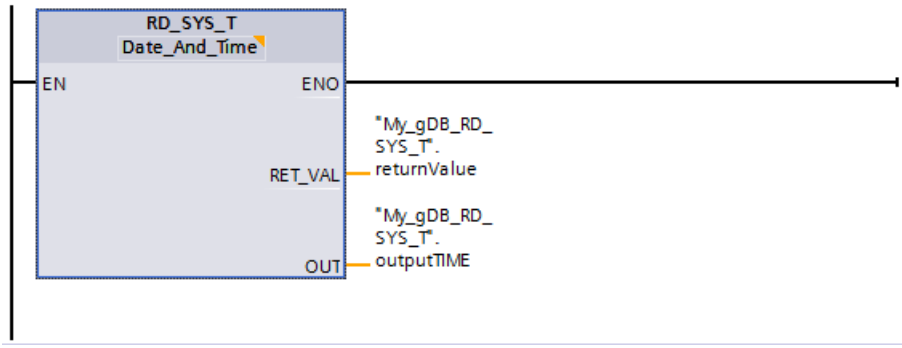

The module time of the CPU clock is read out and displayed at output parameter OUT ("outputTIME"). The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

My_gDB_RD_SYS_T

	Name	Data type	Start value	Monitor value
1	Static			
2	outputTIME	Date_And_Time	DT#1990-01-01-00:00:00	DT#2014-08-04-15:15:15
3	returnValue	Int	0	0

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

See also

Switching display formats in the program status (Page 7698)

RD_LOC_T: Read local time (S7-1200, S7-1500)

Description

You use this instruction to read the current local time from the CPU clock and output this at the OUT output. Information on the time zone and the start of daylight saving time and standard time, which you have set in the configuration of the CPU clock, is used to output the local time.

Parameters

The following table shows the parameters of the "RD_LOC_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Status of the instruction
OUT	Output	DTL	DT, LDT, DTL	I, Q, M, D, L, P *	Local time
* The data types DT and DTL cannot be used for the memory areas: input, output and bit memory.					

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0000	No error
0001	No error. Local time is output as daylight saving time.
8080	Local time cannot be read.
8081	The current local time cannot be saved because it is outside the permissible range for the selected data type of OUT parameter. • With DT: min. DT#1990-01-01-0:0:0, max. DT#2089-12-31-23:59:59.999 • With LDT: min. LDT#1970-01-01-0:0:0.000000000, max. LDT#2262-04-11-23:47:16.854775807 • With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2262-04-11-23:47:16.854775807
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

In the following example, you read out the local time of the CPU clock. The data type used is DATE AND TIME.

Create two tags in a global data block for storing the data.

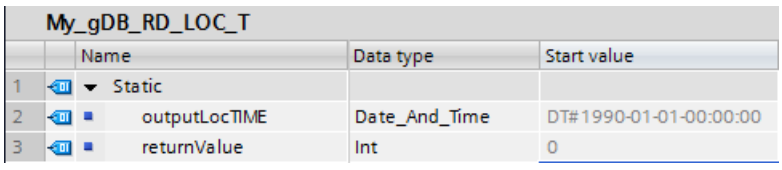

Interconnect the parameters of the instruction as follows. Select the DATE AND TIME data type.

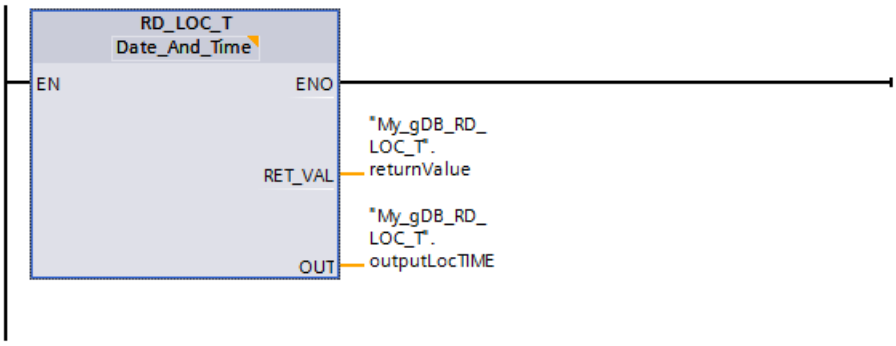

The local time of the CPU clock is read out and displayed at output parameter OUT ("outputLocTIME"). Output parameter RET_VAL ("returnValue") indicates that processing took place without errors and the local time is output as daylight saving time with this call.

My_gDB_RD_LOC_T

	Name	Data type	Start value	Monitor value
1	Static			
2	outputLocTIME	Date_And_Time	DT#1990-01-01-00:00:00	DT#2014-08-04-16:15:15.001
3	returnValue	Int	0	1

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

See also

Switching display formats in the program status (Page 7698)

WR_LOC_T: Write local time (S7-1200, S7-1500)

Description

The instruction "WR_LOC_T" is used to set the date and time of the CPU clock. Enter the date and time-of-day as local time at the input parameter LOCTIME.

The value must be in the following range:

- With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31 23:59:59.999999999
- With LDT: min. LDT#1970-01-01-0:0:0.000000000, max. LDT#2200-12-31 23:59:59.999999999

The granularity of the time information for local time and system time is product-specific and is at least one millisecond. Input values at the LOCTIME parameter which are less than those supported by the CPU are rounded up during system time calculation.

You can query whether errors have occurred during execution of the instruction in the RET_VAL output parameter.

Parameters

The following table shows the parameters of the "WR_LOC_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
LOCTIME	Input	DTL	DTL, LDT	I, Q, M, D, L, P or constant *	Local time
DST	Input	BOOL	BOOL	I, Q, M, D, L, P, T, C or constant	**D**aylight **S**aving **T**ime Is only evaluated during the "double hour" at changeover to standard time. • TRUE = daylight saving time (first hour) • FALSE = standard time (second hour)
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Error message (see "RET_VAL parameter")
* The data type DTL cannot be used for the memory areas: input, output and bit memory.					

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter DST (Daylight Saving Time)

The parameter DST is evaluated only during the changeover from daylight saving to standard time:

- Use of DST during changeover from daylight saving to standard time
  The clock is set back by [1] one hour at 3 a.m. This means there are two hours between 02:00:00:000000000 AM and 02:59:59:999999999 AM.
  For all times for LOCTIME that are between 02:00:00:000000000 AM and 02:59:59:999999999 AM, you therefore need to declare whether the time is before or after the time changeover. The DST parameter is used for this:
  - With DST=TRUE, the time is in the first of the two hours, i.e. still in daylight saving time.
  - With DST=FALSE, the time is in the second of the two hours, i.e. in standard time.

  For all times for LOCTIME that are outside the double hour, DST is irrelevant.
- Changeover from standard to daylight saving time
  The parameter DST is not relevant for the changeover from standard to daylight saving time.
  The clock is set forward by [1] one hour at 2 a.m. This means there is no hour between 02:00:00:000000000 AM and 02:59:59:999999999 AM.
  If you specify a time within this hour at LOCTIME, the error code W#16#8089 is output.

[1]The description here assumes Central European Time (CET). The changeover time for CET is 2:00 AM (start of daylight saving), or 3:00 AM (end of daylight saving). Different changeover times may apply correspondingly to other time zones.

Parameter RET_VAL

Error code* (W#16#....)	Description
0000	No error.
8080	The LOCTIME parameter has an invalid value.
8081	Time value specified at the LOCTIME parameter is outside the valid value range: • With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31 23:59:59.999999999 • With LDT: min. LDT#1970-1-1-0:0:0.000000000, max. LDT#2200-12-31 23:59:59.999999999
8082**	Invalid value specified for the month (byte 2 in DTL format).
8083**	Invalid value specified for the day (byte 3 in DTL format).
8084**	Invalid value specified for the hour (byte 5 in DTL format).
8085**	Invalid value specified for the minute (byte 6 in format DTL).
8086**	Invalid value specified for the second (byte 7 in DTL format).
8087**	Invalid value specified for the nanosecond (byte 8 to 11 in DTL format).
8089	Time value does not exist (hour already passed upon changeover to daylight saving time).
80B0	The real-time clock has failed.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	
** Only for local time information at the LOCTIME parameter in DTL format.	

Example

In the following example, you set the local time of the CPU clock. The data type used is DTL.

Create four tags in a global data block for storing the data.

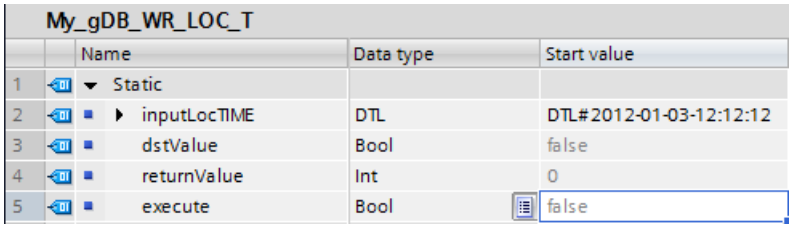

My_gDB_WR_LOC_T

	Name	Data type	Start value
1	Static		
2	inputLocTIME	DTL	DTL#2012-01-03-12:12:12
3	dstValue	Bool	false
4	returnValue	Int	0
5	execute	Bool	false

Interconnect the parameters of the instruction as follows. Select the DTL data type.

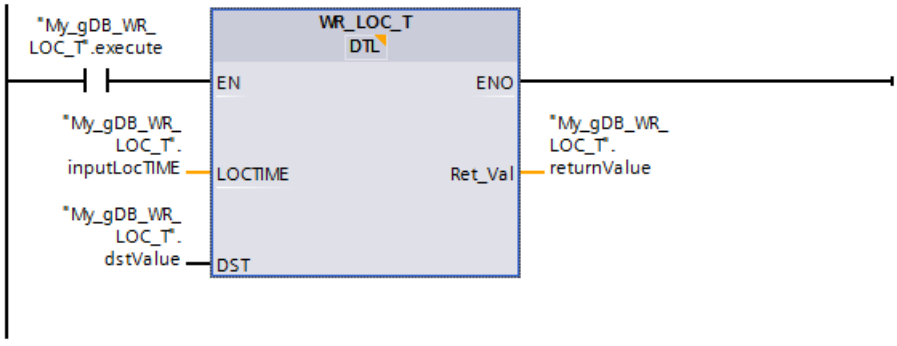

If the normally open contact ("execute") supplies the signal state "TRUE", the "WR_LOC_T" instruction is executed. The local time of the CPU clock is overwritten with the time to be set ("inputLocTIME"). The output parameter RET_VAL ("returnValue") indicates that the process took place without errors. Input parameter DST ("dstValue") specifies that the time information refers to standard time. This parameter is only relevant for the "double hour" (see description "Parameter DST").

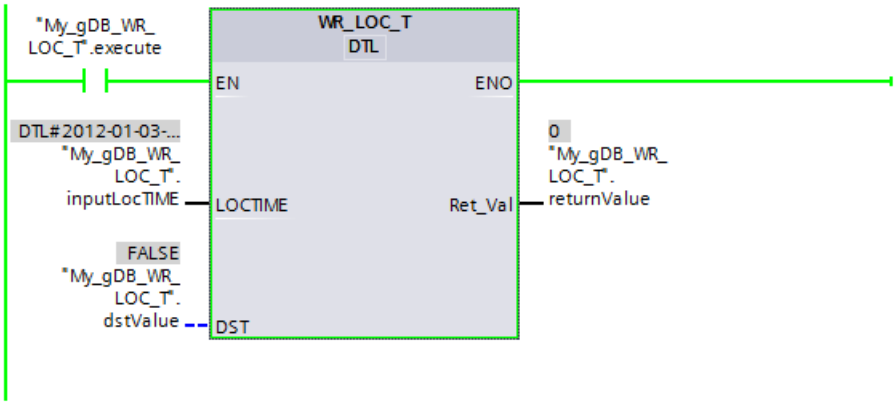

You can establish whether the new local time ("inputLocTIME") has been correctly received by the CPU clock as follows:

- Using the display of an S7-1500 CPU: Navigate on the CPU display to "Settings > Date & Time > General".
- Using TIA Portal: Read out the local time of the CPU clock using the "RD_LOC_T (Page 2105)" instruction.
- Using TIA Portal: Navigate to the "Online & Diagnostics" entry of the CPU, and open the "Functions > Set time of day" tab.
  The local time is output in a 12-hour clock format.

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

See also

Switching display formats in the program status (Page 7698)

SET_TIMEZONE: Set time zone (S7-1200, S7-1500)

Description

Use the instruction "SET_TIMEZONE" to set the parameter for the local time zone and the daylight saving / standard time changeover.

The settings which are carried out with the instruction "SET_TIMEZONE" correspond with the settings for the time-of-day in the properties of the CPU. Define the corresponding parameters in the system data type TimeTransformationRule for execution of the instruction "SET_TIMEZONE".

The local time is calculated based on the system time using the settings for the time zone and the daylight saving / standard time changeover. The system time of the CPU is the UTC time. The system time is used exclusively for communication within the system.

Note

Calling the "SET_TIMEZONE" instruction

The instruction "SET_TIMEZONE" writes data internally to the load memory of the CPU. Accordingly, the change in time zone also applies after a power failure. Calling the instruction again is unnecessary.

Call the instruction once every time the time zone has to be changed. For example, it is advisable to call "SET_TIMEZONE" in a startup OB.

Note

Use with CPUs of the S7-1500 series

The "SET_TIMEZONE" instruction can only be used with a CPU of the S7-1500 series starting from firmware version V1.7.

Parameters

The following table shows the parameters of the "SET_TIMEZONE" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, P or constant	Setting of parameters that are stored at parameter TimeZone.
TimeZone	Input	TimeTransformationRule	D, L	You interconnect the TimeTransformationRule system data type (see below) at parameter TimeZone.
DONE	Output	BOOL	I, Q, M, D, L, P	Status parameter: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L, P	Status parameter: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L, P	Status parameter: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L, P	Detailed error and status information is output at the parameter STATUS. The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter TimeZone

You store the parameters for the local time zone and the daylight saving/standard time changeover in the TimeTransformationRule system data type

You create the TimeTransformationRule by entering TimeTransformationRule as the data type in a data block or the local interface of a function block.

The structure of the TimeTransformationRule is as follows:

Name	Data type	Description
TimeTransformationRule	STRUCT	

Name	Data type	Description
Bias	INT	Time difference between the local time and the system time (UTC) in minutes. The value must be between -720 and +780 minutes (-12 to +13 hours). The value (UTC -12 to +13 hours) corresponds to the time zones that you specify in the properties of the CPU.
DaylightBias	INT	Time difference between standard and daylight saving time in minutes. The value must be between 0 and 120 minutes. • The value "0" deactivates the changeover between daylight saving time and standard time. The values for "DaylightStart..." and "StandardStart..." are set to "0". Only the value of Bias is evaluated (time difference local time/ system time). • If the value is not "0", all tags of the TimeTransformationRule structure are evaluated. If an entry is invalid, error code 808F is output at parameter STATUS.
Specification of the time of day for changeover to daylight saving time. The following times always refer to the local time.		
DaylightStartMonth	USINT	Month in which change is made to daylight saving time: 1 = January 2 = February 3 = March ... 12 = December
DaylightStartWeek	USINT	Week in which change is made to daylight saving time. 1 = First occurrence of the weekday in the month ... 5 = Last occurrence of the weekday in the month
DaylightStartWeek-day	USINT	Weekday on which change is made to daylight saving time: 1 = Sunday ... 7 = Saturday
DaylightStartHour	USINT	Hour in which change is made to daylight saving time
DaylightStartMinute	USINT	Minute in which change is made to daylight saving time
Specification of the time of day for changeover to standard time. The following times always refer to the local time.		
StandardStartMonth	USINT	Month in which change is made to standard time: 1 = January 2 = February 3 = March ... 12 = December
StandardStartWeek	USINT	Week in which change is made to standard time: 1 = First occurrence of the weekday in the month ...

Name		Data type	Description
			5 = Last occurrence of the weekday in the month
	StandardStartWeek-day	USINT	Weekday on which change is made to standard time: 1 = Sunday ... 7 = Saturday
	StandardStartHour	USINT	Hour in which change is made to standard time
	StandardStartMinute	USINT	Minute in which change is made to standard time
	TimeZoneName	STRING[80]	Not used: The configured character string is ignored and not written to the CPU-internal data.

Parameter STATUS

Error code* (W#16#....)	Description
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
808F	The structure, content or data types of the TimeTransformationRule at the TimeZone parameter are invalid or inconsistent.
80C3	Temporary resource error: The CPU is currently processing the maximum possible number of simultaneous block calls. "SET_TIMEZONE" cannot be executed until at least one of the block calls is finished.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

In the following example, you set the parameters for the local time zone and the daylight saving/standard time changeover. You are advised to call the example in a startup OB.

Create the following for storing the data in a global data block: The "timezone" structure (with the TimeTransformationRule data type) and five additional tags.

Note

If you want to call the example in a startup OB: Set the start value of the tag "execute" to "TRUE".

SLI_gDB_SET_TIMEZONE			
	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	timezone	TimeTransformatio...	
4	Bias	Int	120
5	DaylightBias	Int	60
6	DaylightStartMonth	USInt	3
7	DaylightStartWeek	USInt	5
8	DaylightStartWeekday	USInt	7
9	DaylightStartHour	USInt	2
10	DaylightStartMinute	USInt	0
11	StandardStartMonth	USInt	10
12	StandardStartWeek	USInt	5
13	StandardStartWeekday	USInt	1
14	StandardStartHour	USInt	3
15	StandardStartMinute	USInt	0
16	TimeZoneName	String[80]	'My_GMT+'
17	modeDONE	Bool	false
18	modeBUSY	Bool	false
19	modeERROR	Bool	false
20	statusTime	Word	16#0
21	memErrStatus	Word	16#0

Create a local tag for the interconnection of the output parameter DONE.

9	statDone	Bool	false	Non-retain

Network 1: Interconnect the parameters of the "SET_TIMEZONE" instruction as follows.

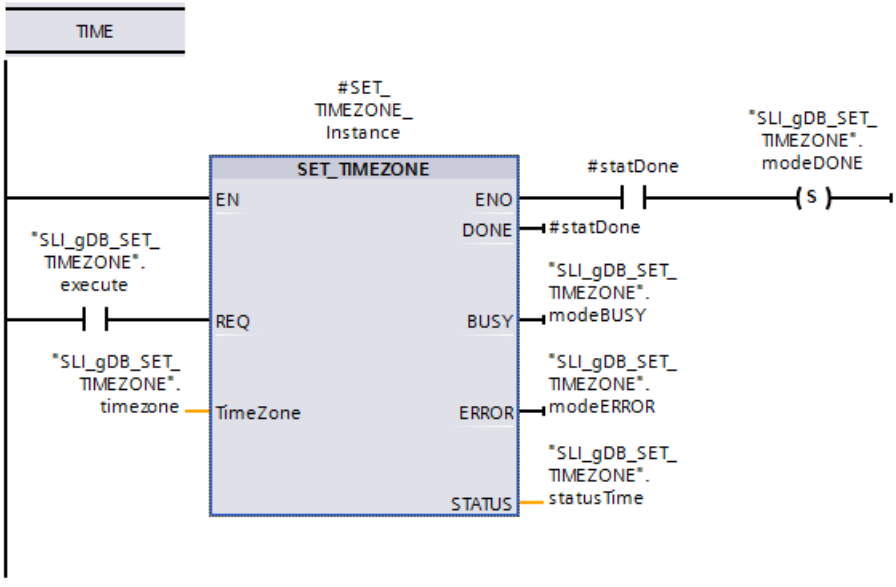

Also, create the following additional interconnections:

Network 2: In case of error, you save the status in the "memErrStatus" tag.

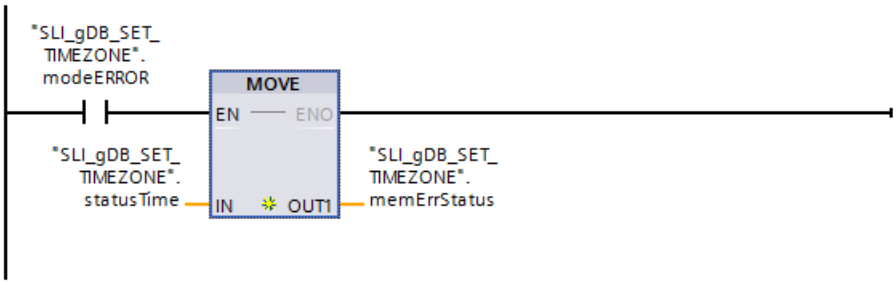

Network 3: After successful processing, have the input parameter REQ ("execute") automatically reset as follows.

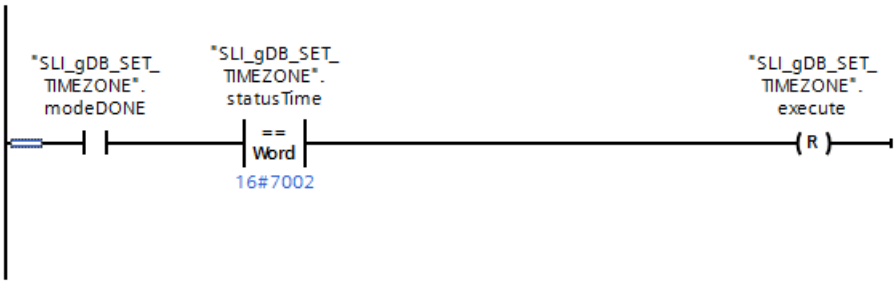

Network 4: To ensure that the instruction "SET_TIMEZONE" is fully processed when called in a startup OB, use the instructions JMP (in network 4) and Label (see network 1).

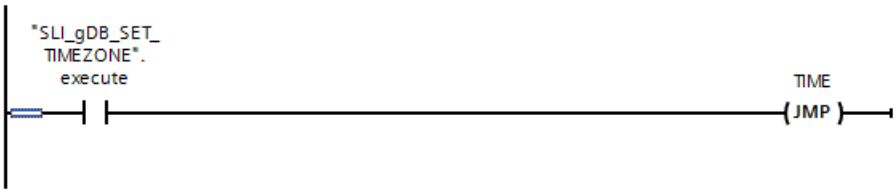

Note

If you want to call the example in a startup OB, the CPU for processing the example must be in startup. Set the CPU to STOP mode before you load the example to the CPU. Then set the CPU to RUN mode.

If the input parameter REQ ("execute) returns the signal state "TRUE", the data for the time zone of the CPU clock is overwritten by the data to be set ("timezone"). That also means that:

- Output parameter BUSY ("modeBUSY") goes to signal state "TRUE". After processing, output parameter BUSY shows the value "FALSE" and output parameter DONE ("#statDone") shows the value "TRUE". The success status is saved to the "modeDONE" tag.
- The output parameter STATUS ("statusTime") indicates how processing is running*. (*The job processing begins (value "7001") and the instruction is then shown as already active (value "7002").)
- The output parameter ERROR ("modeERROR") indicates that processing is running without errors (signal state is "FALSE").

SLI_gDB_SET_TIMEZONE				
	Name	Data type	Start value	Monitor value
1	Static			
2	execute	Bool	false	FALSE
3	timezone	TimeTransformatio...		
4	Bias	Int	120	120
5	DaylightBias	Int	60	60
6	DaylightStartMonth	USInt	3	3
7	DaylightStartWeek	USInt	5	5
8	DaylightStartWeekday	USInt	7	7
9	DaylightStartHour	USInt	2	2
10	DaylightStartMinute	USInt	0	0
11	StandardStartMonth	USInt	10	10
12	StandardStartWeek	USInt	5	5
13	StandardStartWeekday	USInt	1	1
14	StandardStartHour	USInt	3	3
15	StandardStartMinute	USInt	0	0
16	TimeZoneName	String[80]	'My_GMT+'	'My_GMT+'
17	modeDONE	Bool	false	TRUE
18	modeBUSY	Bool	false	FALSE
19	modeERROR	Bool	false	FALSE
20	statusTime	Word	16#0	16#7000
21	memErrStatus	Word	16#0	16#0000

Note: The "SET_TIMEZONE" instruction is level-triggered. The instruction is executed only when the input parameter REQ ("execute") returns the signal state "TRUE".

After successful processing of SET_TIMEZONE ("modeDONE" is "TRUE"), the input parameter REQ ("execute") is automatically reset to the value "FALSE". The value "16#7002" is queried for "statusTime" for this purpose. After the reset, the status value "16#7000" is output.

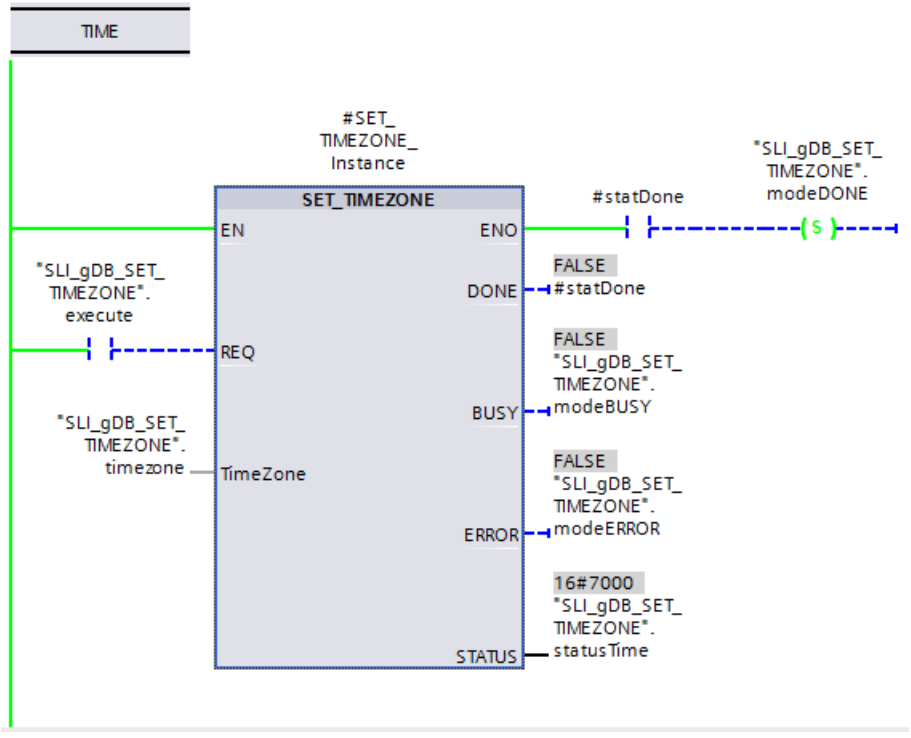

You can establish whether the data to be set ("timezone") has been correctly received by the CPU clock as follows:

- Using the display of an S7-1500 CPU: Navigate on the CPU display to "Settings > Date & Time > Daylight Saving Time".
- Using TIA Portal: Read out the local time of the CPU clock using the "RD_LOC_T (Page 2105)" instruction.
- Using TIA Portal: Read out the module time of the CPU clock using the "RD_SYS_T (Page 2103)" instruction.

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

SNC_RTCB: Synchronize slave clocks (S7-1500)

Definition: Synchronization of slave clocks

The synchronization of clock slaves refers to the transfer of the date and time-of-day from the clock master of a bus segment to all clock slaves of this bus segment.

Description

You use this instruction to synchronize all slave clocks present locally on the CPU or locally on a bus segment independent of the set synchronization interval (the external interface of a CP or CM belongs to a different bus segment). Successful synchronization is only possible if "SNC_RTCB" is called on a CPU whose real-time clock was assigned as the master clock for at least one bus segment.

Parameters

The following table shows the parameters of the "SNC_RTCB" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred during synchronization.
0001	The existing clock was not assigned the master clock function for any of the bus segments.

Error code* (W#16#...)	Explanation
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

TIME_TCK: Read system time (S7-1500)

Description

With the "TIME_TCK" instruction, you read the system time of the CPU. The system time is a time counter that counts from 0 to a maximum of 2147483647 ms. In case of an overflow, the system time is counted again starting with "0". The time scale and the accuracy of the system time is 1 ms. The system time is only influenced by the operating modes of the CPU. You can use the system time, for example, to measure the duration of processes by comparing the results of two "TIME_TCK" calls. The instruction does not provide any error information.

The following table provides an overview of how the system time changes depending on the operating modes of the CPU.

Operating mode	System time ...
Startup	... is constantly updated.
RUN	
STOP	... is stopped and retains the current value.
Warm restart	... is deleted and restarts with "0".

Parameters

The following table shows the parameters of the "TIME_TCK" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	TIME	I, Q, M, D, L	The RET_VAL parameter contains the read system time in the range from 0 to 2^{31} -1 ms.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you read out the system time of the CPU. You specify the return value in the TIME data type.

Create one tag for storing the data in a global data block.

My_gDB_TIME_TCK			
	Name	Data type	Start value
1	Static		
2	outputCPUtimer	Time	T#0ms

Interconnect the parameters of the instruction as follows.

The system time of the CPU is read out and displayed at output parameter RET_VAL ("outputCPUtimer").

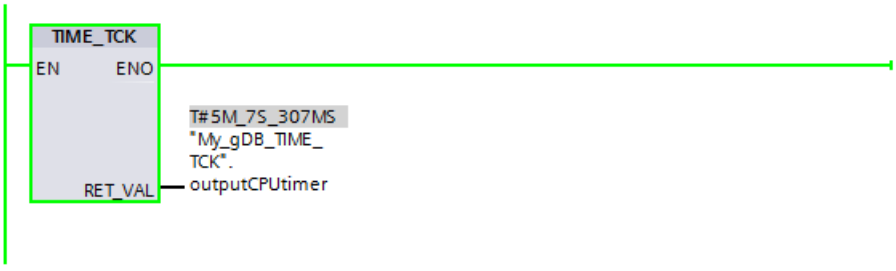

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

See also

Switching display formats in the program status (Page 7698)

RTM: Runtime meters (S7-1200, S7-1500)

Description

You can use this instruction to set, start, stop, and read out a 32-bit operating hours counter of your CPU.

Ensure that the operating hours counter can also be stopped or restarted during execution of the user program, which may render the saved values incorrect.

Parameters

The following table shows the parameters of the "RTM" instruction:

Parameter	Declaration	Data type	Memory area	Description
NR	Input	RTM	I, Q, M, D, L or constant	Number of the operating hours counter Numbering starts with 0. For information on the number of operating hours counters of your CPU, refer to the technical specifications.
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID: • 0: Read out (the status is then written to CQ and the current value to CV). After the operating hours counter has reached (2^31) - 1 hours, it stops at the highest value that can be displayed and outputs an "Overflow" error message. • 1: start (at the last counter value) • 2: stop • 4: set to the value specified in the parameter PV • 5: set to the value specified in the parameter PV and start • 6: set to the value specified in the parameter PV and stop
PV	Input	DINT	I, Q, M, D, L or constant	New value for the operating hours counter
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
CQ	Output	BOOL	I, Q, M, D, L	Status of the operating hours counter (1: running)
CV	Output	DINT	I, Q, M, D, L	Current value of the operating hours counter

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the operating hours counter
8081	A negative value was passed to the PV parameter.
8082	Overflow of the operating hours counter
8091	The MODE input parameter contains an invalid value.
General error information	See also: Evaluating errors with GET_ERR_ID (Page 1857)

Example

In the following example, you set the operating hours counter of the CPU and read out the value after an hour.

Create six tags in a global data block for storing the data.

My_gDB_RTM			
	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	in_processValue	DInt	5
4	returnValue	Int	0
5	statusRTM	Bool	false
6	currentValue	DInt	0
7	comandMODE	Byte	5

Interconnect the parameters of the instruction as follows. At input parameter NR, specify the number of the operating hours counter of the CPU.

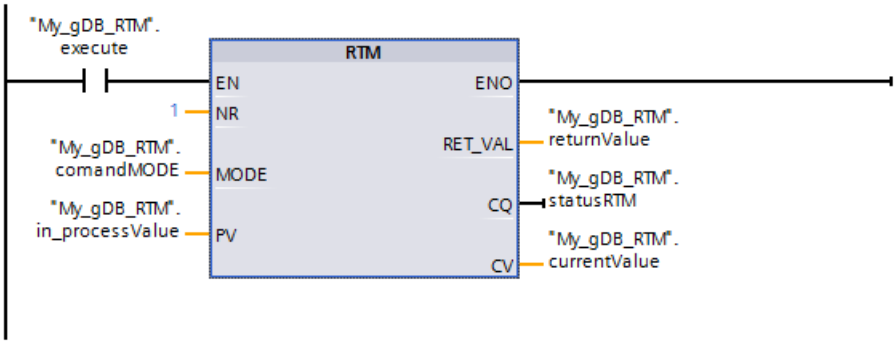

If the normally open contact ("execute") supplies the signal state "TRUE", the "RTM" instruction is executed. The operating hours counter of the CPU is set to the value to be set ("in_processValue") and started. After the start of the operating hours counter, set the value of input parameter MODE ("comandMODE") to "0". (To do so, click the parameter and select "Modify operand > Set to 0".) As a result, the "RTM" instruction only reads the current value ("currentValue") of the operating hours counter without changing it. Output parameter CQ ("statusRTM") indicates after the start of the operating hours counter that the operating hours counter is running (value is "TRUE"). The output parameter RET_VAL ("returnValue") indicates that processing is running without errors.

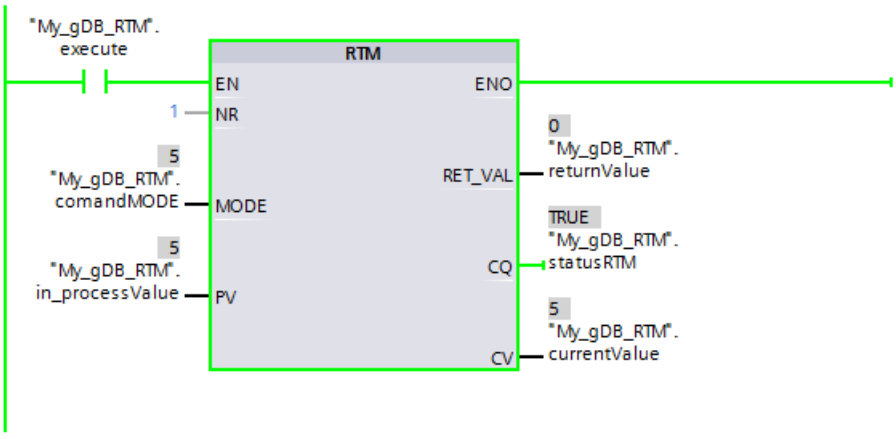

After an hour, output parameter CV ("currentValue") indicates the value "6".

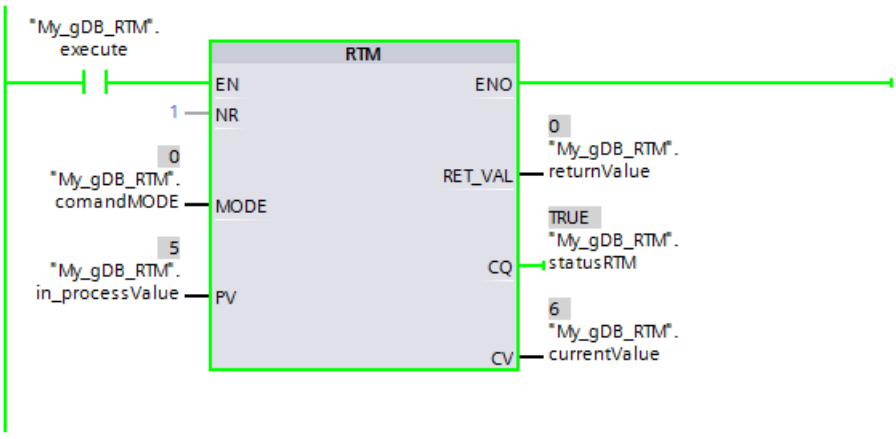

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

4.1.4.2 String + Char

S_MOVE: Move character string (S7-1200, S7-1500)

Description

You can use this instruction to write the content of a character string (W)STRING from parameter IN to the data area that you specify at parameter OUT.

To copy tags of data type ARRAY, use the "MOVE_BLK" and "UMOVE_BLK" instructions.

Parameters

The following table shows the parameters of the "S_MOVE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Source string
OUT	Output	STRING, WSTRING	D, L	Destination string

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you copy the content of a character string from input parameter IN to another character string at output parameter OUT. The data type used is STRING.

Create two tags in a global data block for storing the data.

My_gDB_s_move

		Name	Data type	Start value
1		Static		
2		stringValueIN	String	'cf6a6a'
3		stringValueOUT	String	''

Interconnect the parameters of the instruction as follows.

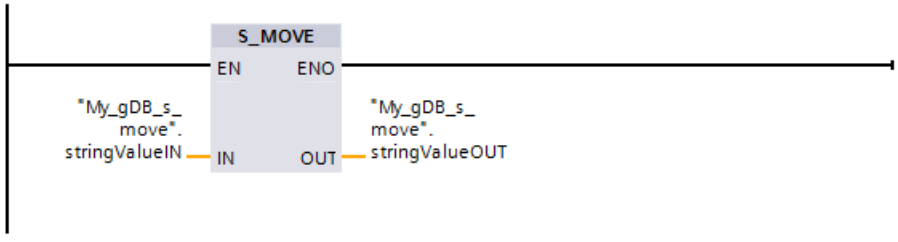

The result of the character string to be copied ("stringValueIN") is output at output parameter OUT ("stringValueOUT").

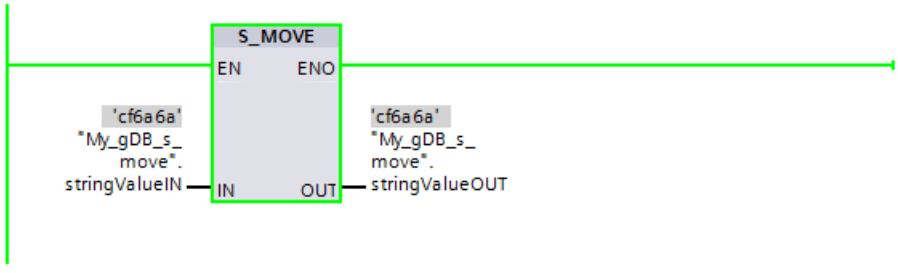

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

S_COMP: Compare character strings (S7-1500)

Description

The instruction compares the contents of two tags in the (W)STRING format and outputs the result of the comparison as a return value. The tags that are to be compared will be interconnected at the IN1 and IN2 inputs. You can only assign a symbolically defined tag for the input parameters.

Use the instruction box to select the comparison condition. If the comparison condition (for example, greater than or equal to) is satisfied, the signal state is set to "1" at the output parameter OUT .

Select one of the following comparison conditions:

Symbol	Description
EQ	The return value has the signal state "1" if the string at the parameter IN1 is equal to the string at the parameter IN2 .
NE	The return value has the signal state "1" if the string at the parameter IN1 is not equal to the string at the parameter IN2 .
GT (1)	The return value has the signal state "1" if the string at the parameter IN1 is greater than the string at the parameter IN2 .
LT (1)	The return value has the signal state "1" if the string at the parameter IN1 is less than the string at the parameter IN2 .
GE (1)	The return value has the signal state "1" if the string at the parameter IN1 is greater than or equal to the string at the parameter IN2 .
LE (1)	The return value has the signal state "1" if the string at the parameter IN1 is less than or equal to the string at the parameter IN2 .
(1) The characters are compared by their ASCII code (for example, 'a' is greater than 'A'), starting from the left. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.	

Parameters

The following table shows the parameters of the "S_COMP" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING, WSTRING*	D, L or constant	Input tag in the STRING / WSTRING format.
IN2	Input	STRING, WSTRING*	D, L or constant	Input tag in the STRING / WSTRING format.
OUT	Output	BOOL	I, Q, M, D, L	Result of comparison
* Define the maximum length of the character string if you use the data type STRING / WSTRING in the interface declaration for a temporary tag (you will find further information in the description of the data type).				

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you use the "Equal" comparison option to compare two character strings with the "STRING" data type.

Create three tags in a global data block for storing the data.

My_gDB_s_comp			
	Name	Data type	Start value
1	Static		
2	stringvalue1	String	'ThisIsAVeryLongStringWithSomeNumbers646'
3	stringvalue2	String	'ThisIsAVeryLongStringWithSomeNumbers646'
4	svalue1EQsvalue2	Bool	false

Interconnect the parameters of the instruction as follows. Select the STRING data type and the EQ comparison option.

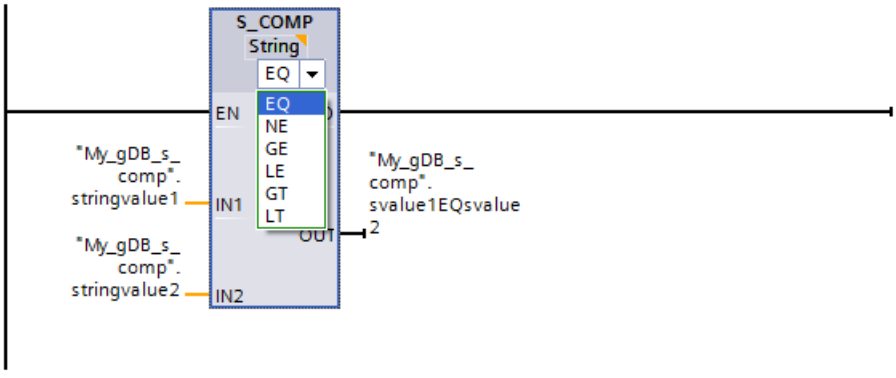

Since the value of the first character string to be compared ("stringvalue1") is equal to the second character string ("stringvalue2"), the comparison result ("svalue1EQsvalue2") shows the signal state "TRUE".

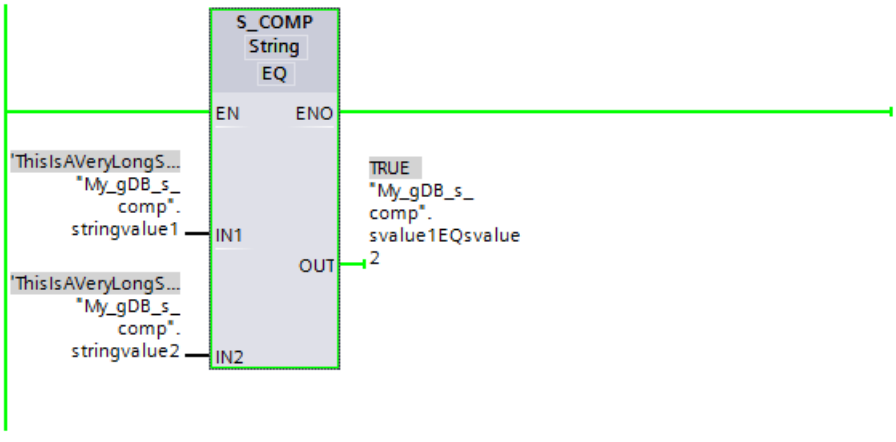

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

S_CONV: Convert character string (S7-1200, S7-1500)

Description

You use this instruction to convert the value at the IN input to the data format you have specified in the OUT output. You decide the output format of the conversion by selecting a data type for the OUT output parameter.

The following conversions are possible:

- Conversion of a string
- Conversion of a numerical value or character to a string
- Conversion of a character to a character

Note

Explicit conversions

You can find more information on explicit conversions here:

- For an S7-1200-CPU: Explicit conversions (Page 511)
- For an S7-1500-CPU: Explicit conversions (Page 405)

Conversion of a string

- To a numerical value (integer or floating point):
  The conversion is performed for all characters of the character string specified in the IN input parameter. Permitted characters are the digits "0" to "9", the decimal point, and the plus and minus signs. The first character of the string may be a valid number or a sign. Leading spaces and exponential notations are ignored.
- To a character:
  When a character string is converted to a character, the first character of the character string is transferred to parameter OUT.
- To a string

Conversion of a numerical value or character to a string

- You decide the format of the numeric value to be converted by selecting a data type for the IN input. Specify a valid tag of the OUT data type at the (W)STRING output. The length of the character string after conversion depends on the value at the IN input.
- The conversion result is saved as a string starting at the third byte. The first byte of the string records the maximum length and the second byte the actual length of the character string. Positive numeric value are output without a sign.
- If the numeric value 0, which exists as an INT or UINT data type, is converted to a string (e.g. INT_TO_STRING(0)), the string in the result has a length of 6 characters.

- When a numeric value is converted to a string, the first characters of the string are padded with spaces. The number of spaces depends on the length of the numerical value.

  Note

  SCL-specific information

  When a numerical value is converted to a string, the first characters of the string are **not** padded with spaces.

  If you want the first character of the string to be padded with spaces, use the "VAL_STRG" instruction.

- When a (W)CHAR character is converted, the character is written at the first position of the character string.

Note

Exponential notation during conversion from floating-point numbers

Do not use exponential notation ("e" or "E") for the conversion from floating-point numbers with the instruction "S_CONV". Instead, use the "STRG_VAL" instruction for the conversion of floating-point numbers with exponential notation. Select the input format of the exponential notation via the FORMAT parameter of the instruction.

Parameters

The following tables show the parameters of the "S_CONV" instruction, according to the possible conversions:

Table 4-11 Parameters for converting a character string to a numeric value:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Value to be converted
OUT	Output	CHAR, WCHAR, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL	I, Q, M, D, L	Result of the conversion

Table 4-12 Parameters for converting a character string to a character string:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Value to be converted
OUT	Output	STRING, WSTRING	D, L	Result of the conversion (possible conversions: STRING to WSTRING and vice-versa)

Table 4-13 Parameters for converting a numerical value or character to a character string:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	CHAR, WCHAR, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL	I, Q, M, D, L or constant	Value to be converted
OUT	Output	STRING, WSTRING	D, L	Result of the conversion

Table 4-14 Parameters for converting a character to a character:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	CHAR, WCHAR	I, Q, M, D, L or constant	Value to be converted
OUT	Output	CHAR, WCHAR	I, Q, M, D, L	Result of the conversion (possible conversions: CHAR to WCHAR and vice-versa)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, convert a number with the INT data type to a character string with the STRING data type.

Create two tags in a global data block for storing the data.

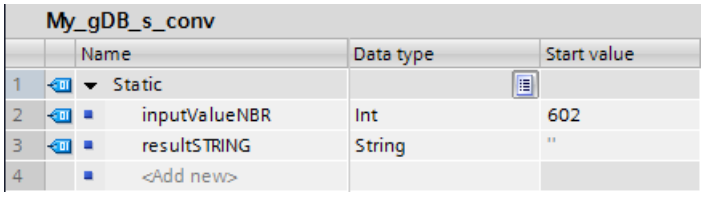

My_gDB_s_conv

	Name	Data type	Start value
1	Static		
2	inputValueNBR	Int	602
3	resultSTRING	String	''
4	<Add new>		

Interconnect the parameters of the instruction as follows. In doing so, select the data types. With the first selection option, you specify the data type of the value to be converted ("inputValueNBR"). With the second selection option, you specify the data type of the character string to be generated ("resultSTRING").

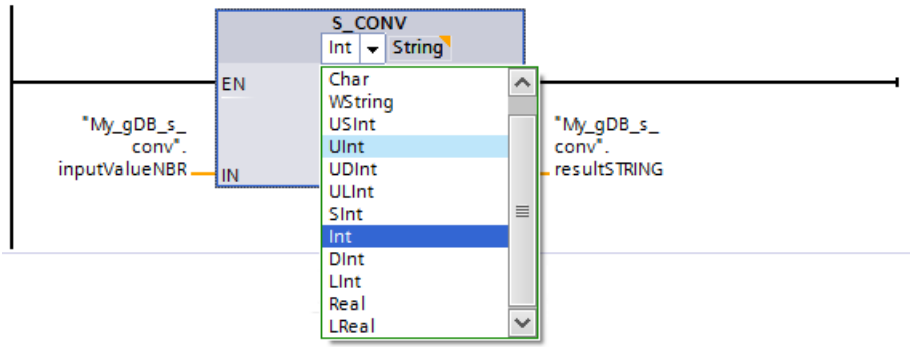

The value to be converted ("inputValueNBR") is converted to the output format. Blank spaces are written to the empty places at the start of the character string. The result of the conversion is output at output parameter OUT ("resultSTRING") as a string.

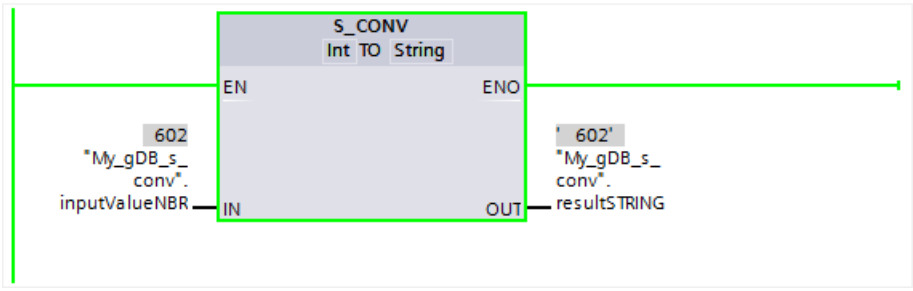

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

See also

STRG_VAL: Convert character string to numerical value (Page 2131)

STRG_VAL: Convert character string to numerical value (S7-1200, S7-1500)

Description

The "STRG_VAL" instruction converts a character string to an integer or a floating-point number:

- You specify the character string to be converted in the IN input parameter.
- You define the format of the output value by selecting a data type for the OUT output parameter.

Permitted characters for the conversion are the digits "0" to "9", the decimal point, the decimal comma, notations "E" and "e", and the plus and minus characters. The conversion is canceled if a invalid character is found.

Parameters

The following table shows the parameters of the "STRG_VAL" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	STRING, WSTRING	STRING, WSTRING	D, L or constant	Number string to be converted
FORMAT	Input	WORD	WORD	I, Q, M, D, L, P or constant	Input format of the characters
P	Input	UINT	UINT	I, Q, M, D, L, P or constant	Reference to the first character to be converted (first character = 1, the value "0" or a value > length of the string is invalid)
OUT	Output	USINT, SINT, UINT, INT, UDINT, DINT, REAL, LREAL	USINT, SINT, UINT, INT, UDINT, DINT, ULINT, LINT, REAL, LREAL	I, Q, M, D, L, P	Result of the conversion

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter FORMAT

You use the FORMAT parameter to specify how the characters of a character string are to be interpreted. Exponential values are also converted and represented with the "STRG_VAL" instruction.

The following table shows the possible values of the FORMAT parameter and their meaning:

Value (W#16#....)	Notation	Decimal representation
0000	Decimal fraction	"."
0001		","
0002	Exponential	"."
0003		","
0004 to FFFF	Invalid values	

Parameter P

The conversion starts at the character whose position you specified in the P parameter. If, for example, the value "1" is specified in the P parameter, the conversion starts at the first character of the specified character string.

Example

The following table shows examples of the conversion of a character string to a numeric value:

IN (STRING)	FORMAT (W#16#....)	OUT (data type)	OUT (value)	ENO status
'123'	0000	INT/DINT	123	1
'-00456'	0000	INT/DINT	-456	1
'123.45'	0000	INT/DINT	123	1
'+2345'	0000	INT/DINT	2345	1
'00123AB'	0000	INT/DINT	123	1
'123'	0000	REAL	123.0	1
'-00456'	0001	REAL	-456.0	1
'+00456'	0001	REAL	456.0	1
'123.45'	0000	REAL	123.45	1
'123.45'	0001	REAL	12345.0	1
'123,45'	0000	REAL	12345.0	1
'123,45'	0001	REAL	123.45	1
'.00123AB'	0001	REAL	123.0	1
'1.23e-4'	0000	REAL	1.23	1
'1.23E-4'	0000	REAL	1.23	1
'1.23E-4'	0002	REAL	1.23E-4	1
'12,345.67'	0000	REAL	12345.67	1
'12,345.67'	0001	REAL	12.345	1
'3.4e39'	0002	REAL	W#16#7F800000	0
'-3.4e39'	0002	REAL	W#16#FF800000	0
'1.1754943e-38'	0002	REAL	0.0	1
'12345'	-/-	SINT	0	0
'A123'	-/-	-/-	0	0
''	-/-	-/-	0	0
'++123'	-/-	-/-	0	0
'+-123'	-/-	-/-	0	0

Example

In the following example, you convert a number string with the STRING data type to a floating-point number with the REAL data type. The result will have a length of 32 bits and may have a sign due to the REAL data type.

Create four tags in a global data block for storing the data.

My_gDB_STRG_VAL			
	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'567.12'
3	resultSformat	Word	16#0001
4	pointerSTRG	UInt	1
5	outputVAL	Real	0.0

Interconnect the parameters of the instruction as follows. Using the left selection option, select the data type for the character string. Using the right selection option, select the data type for the floating-point number.

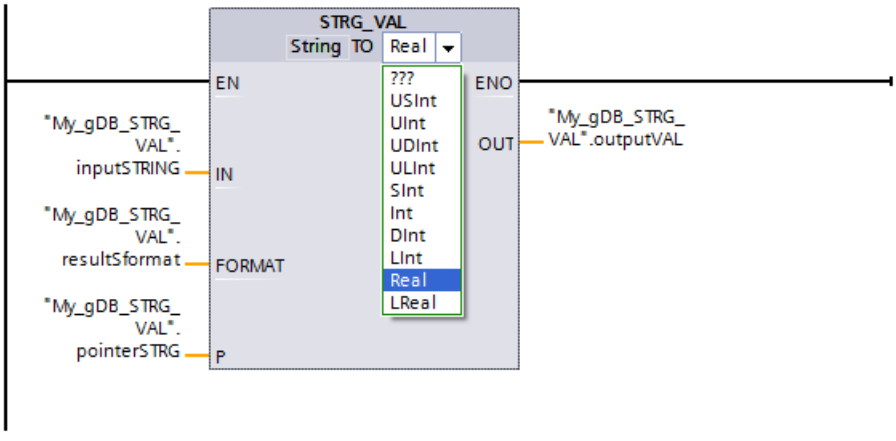

The number string is converted starting from the first character according to the value "1" of parameter P ("pointerSTRG"). The period in the number string is interpreted as a thousands delimiter due to the value "0001" of parameter FORMAT ("resultSformat"). (The comma is the decimal separator for the value "0001".) The value to be converted ("inputSTRING") is output at output parameter OUT ("outputVAL") as a floating point number.

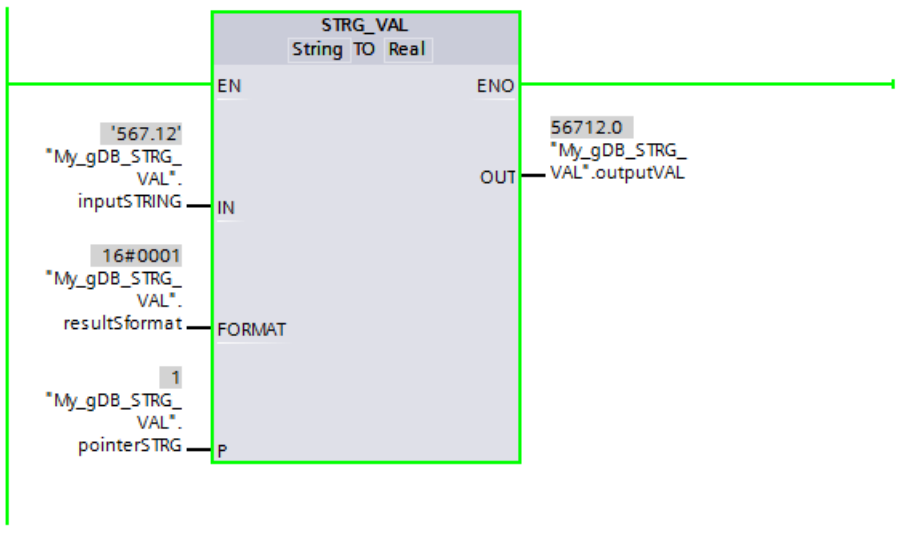

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

VAL_STRG: Convert numerical value to character string (S7-1200, S7-1500)

Description

The "VAL_STRG" instruction is used to convert a numerical value into a character string.

- You specify the value to be converted in the IN input parameter. You decide the format of the numeric value by selecting a data type.
- You query the result of the conversion in the OUT output parameter.

Permitted characters for the conversion are the digits "0" to "9", the decimal point, the decimal comma, notations "E" and "e", and the plus and minus characters. The conversion is interrupted by invalid characters.

Parameter

The following table shows the parameters of the "VAL_STRG" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	USINT, SINT, UINT, INT, UDINT, DINT, REAL, LREAL	USINT, SINT, UINT, INT, UDINT, DINT, ULINT, LINT, REAL, LREAL	I, Q, M, D, L, P or constant	Value to be converted
SIZE	Input	USINT	USINT	I, Q, M, D, L, P or constant	Number of character positions
PREC	Input	USINT	USINT	I, Q, M, D, L, P or constant	Number of decimal places
FORMAT	Input	WORD	WORD	I, Q, M, D, L, P or constant	Output format of the characters
P	InOut	UINT	UINT	I, Q, M, D, L, P or constant	Character starting at which the result is written.
OUT	Output	STRING, WSTRING	STRING, WSTRING	D, L	Result of the conversion

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter P

With the P parameter, you specify the character in the string starting at which the result is written. If, for example, the value "2" is specified in the P parameter, the converted value is saved starting at the second character of the string.

SIZE and P parameters

Use the SIZE parameter to specify how many characters of the character string will be written. This is counted starting from the character specified in the P parameter. If the output value is shorter than the specified length, the result is written to the character string right-justified. The empty character positions are filled with blanks.

Parameter FORMAT

Use the FORMAT parameter to specify how the numerical value is interpreted during conversion and written to the character string. You can specify only tags of the FORMAT data type in the USINT parameter.

The following table shows the possible values of the FORMAT parameter and their meaning:

Value (W#16#....)	Notation	Sign	Decimal representation
0000	Decimal fraction	"-"	"."
0001			","
0002	Exponential		"."
0003			","
0004	Decimal fraction	"+" and "-"	"."
0005			","
0006	Exponential		"."
0007			","
0008 to FFFF	Invalid values		

Note

Relevance of values 2, 3, 6 and 7 of the FORMAT parameter

The values 2, 3, 6 and 7 of the FORMAT parameter are only relevant if the parameter IN has data type REAL or LREAL.

Parameter PREC

Use the PREC parameter to define the number of decimal places when converting floating-point numbers. A maximum precision of seven numbers is supported for numerical values of the REAL data type. If the value to be converted is an integer, you use the PREC parameter to specify the position where the decimal point will be placed.

Example

The following table shows examples of the conversion of numeric values to a character string if the parameters OUT contains an empty string before the call of VAL_STRG.

IN (value)	IN (data type)	P	SIZE	FORMAT (W#16#....)	PREC	OUT (STRING)	ENO status
123	UINT	16	10	0000	0	`xxxxxxx123 C`	1
0	UINT	16	10	0000	2	`xxxxxx0.00 C`	1

IN (value)	IN (data type)	P	SIZE	FORMAT (W#16#....)	PREC	OUT (STRING)	ENO status
12345678	UDINT	16	10	0000	3	x12345.678 C	1
12345678	UDINT	16	10	0001	3	x12345.678 C	1
123	INT	16	10	0004	0	xxxxxx+123 C	1
-123	INT	16	10	0004	0	xxxxxx-123 C	1
-0.00123	REAL	16	10	0004	4	xxx-0.0012 C	1
-0.00123	REAL	16	10	0006	4	-1.2300E-3 C	1
-Inf [1]	REAL	16	10	-/-	4	xxxxxx-INF C	0
+Inf [2]	REAL	16	10	-/-	4	xxxxxx+INF C	0
NaN [3]	REAL	16	10	-/-	4	xxxxxxxNaN C	0
12345678	UDINT	16	6	-/-	3	xxxxxxxxxx C	0

"x" represents blanks
[1]-Inf: Floating-point number representing a negative infinite value.
[2]+Inf: Floating-point number representing a positive infinite value.
[3]NaN: Value returned as the result of invalid math operations.

Example

The following table shows examples of the conversion of numeric values to a character string when the parameter OUT contains an empty string prior to calling VAL_STRG.

In all cases, the parameters IN, SIZE, PREC, FORMAT and P have the same values: IN=123, SIZE=16, PREC=0, FORMAT=4, P=10. Only the default of OUT is different.

- OUT contains a string that is shorter than P.
  The CPU fills OUT after the existing string with spaces up to the (P-1)th position. As of position P OUT It is written taking into account leading spaces.

OUT before the call of VAL_STRG:	ABCDEFxxxxxxxxxx xxxxxxxxxxxxxxxx
OUT after the call of VAL_STRG:	ABCDEFxxxxxxxxxx xxxxxx123xxxxxxx
"x" represents a space	

- OUT contains a string that is longer than P+SIZE
  As of position P the CPU fills OUT with SIZE characters taking into account the leading spaces.

OUT before the call of VAL_STRG:	ABCDEFGHIJKLMNOP QRSTUVWXYZABCDEF
OUT after the call of VAL_STRG:	ABCDEFGHIxxxxxxx xxxxxx123ZABCDEF
"x" represents a space	

- OUT contains a string that is longer than P but shorter than P+SIZE
  As of position P the CPU fills OUT with SIZE characters taking into account the leading spaces.

OUT before the call of VAL_STRG:	ABCDEFGHIJKLMNOP xxxxxxxxxxxxxxxx
OUT after the call of VAL_STRG:	ABCDEFGHIxxxxxxx xxxxxx123xxxxxxx
"x" represents a space	

Example

In the following example, you convert a floating-point number with the REAL data type to a character string with the STRING data type.

Create six tags in a global data block for storing the data.

My_gDB_VAL_STRG			
	Name	Data type	Start value
1	Static		
2	inputVAL	Real	-567.12
3	pointer2STRG	UInt	16
4	sizeSTRG	USInt	10
5	precVAL	USInt	3
6	outputSTRING	String	''
7	resultV2Sformat	Word	16#0004

Interconnect the parameters of the instruction as follows. Select the data types:

- The left selection option is for the value to be converted.
- The right selection option is for the character string to be generated.

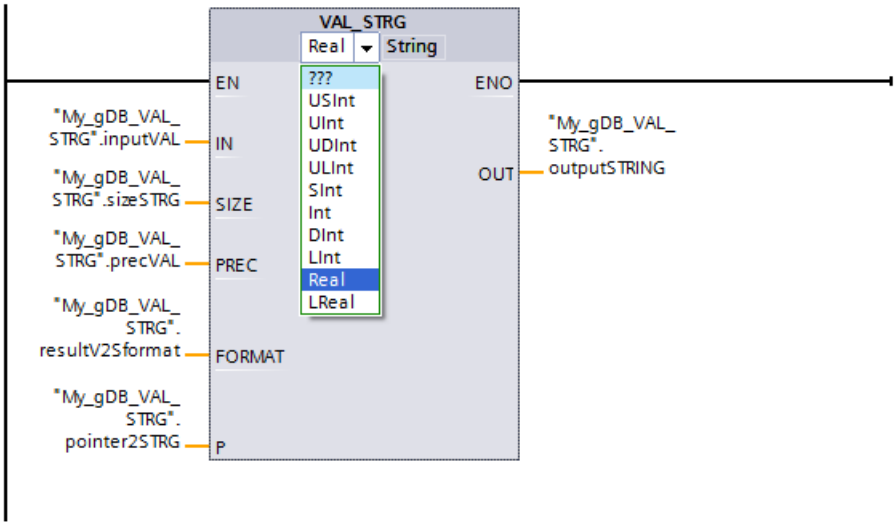

The character string is written starting from the 16th character according to the value "16" of parameter P" ("pointer2STRG") . Starting from this point, the character string is 10 characters long, according to the "10" value of the SIZE parameter ("sizeSTRG"). The point of the value to be converted ("inputVAL") is interpreted as a decimal separator due to the value "0004" of parameter FORMAT ("resultV2Sformat"). Three decimal places are written to the string according to the value "3" of parameter PREC ("precVAL"). The sign of the value to be converted is stored as a character in the string and prefixes the numbers. The remaining 10 characters of the string are written as spaces in front of the sign. The character string is output at output parameter OUT ("outputSTRING").

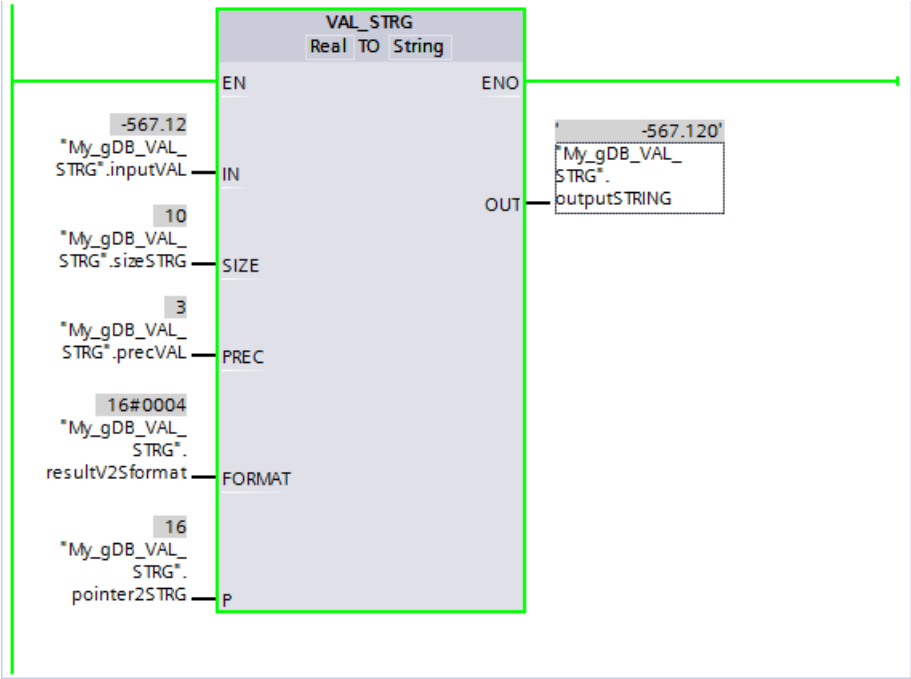

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

Strg_TO_Chars: Convert character string to Array of CHAR (S7-1200, S7-1500)

Description

With the "Strg_TO_Chars" instruction, you copy a character string of the data type STRING to an Array of CHAR or Array of BYTE or a character string of the data type WSTRING to an Array of WCHAR or Array of WORD. Only ASCII characters are valid for the copy process.

- Specify the character string at the input parameter STRG.
- The characters are written at the parameter CHARS to a Array of CHAR / BYTE / WCHAR / WORD data type.
  - The number of characters in the destination field must be at least the same as the number copied from the source string.
  - If the destination field contains fewer characters than the source string, the characters are written up to the maximum length of the destination field.
  - If the character string contains a "$00" or W#16#0000 character, this has no effect on the copy operation (see graphic).
  - The number of copied characters is output at parameter CNT.
- Use the parameter PCHARS to indicate from which position writing will begin in the destination field.
  - Example: If writing should take place starting from the third position, use the value "2" at parameter PCHARS:

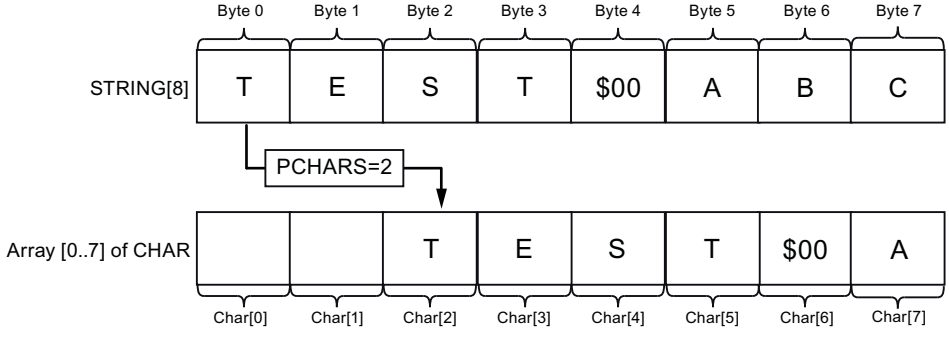

  - The default for PCHARS is "0". When PCHARS = 0, the low index limit of the array is used (e.g. CHAR[0] for an Array [0..5] of CHAR). This also applies if the low limit of the array is negative (e.g. CHAR[-5] for an Array [-5..5] of CHAR).

Note

Use of the instruction with S7-1200 V2.0

S7-1200 up to Version 2.0 only supports Array [0 .. n] of CHAR / BYTE. Negative index limits (e.g. Array [-3..2] of CHAR) are not permissible. This restriction is not checked by the software.

Parameters

The following table shows the parameters of the "Strg_TO_Chars" instruction:

Parameter	Declaration	Data type	Memory area	Description
STRG	Input	STRING, WSTRING	D, L or constant	Source of the copy operation
PCHARS	Input	DINT	I, Q, M, D, L, P or constant	Position in the structure Array of (W)CHAR / BYTE / WORD starting from which the characters of the character string are written.
CHARS	InOut	VARIANT	D, L	Destination of the copy operation Copy the characters to a structure of the Array of (W)CHAR/ BYTE/ WORD data type.
CNT	Output	UINT	I, Q, M, D, L, P	Number of moved characters

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you copy characters of a character string with the STRING data type to a structure of the Array of CHAR data type.

Create four tags in a global data block for storing the data.

My_gDB_Strg_TO_Chars

	Name	Data type	Start value
1	Static		
2	inputSTRG	String[10]	'cf7a7a#'
3	pointerCHARS	DInt	2
4	myarrayCHARS	Array[0..9] of Char	
5	myarrayCHARS[0]	Char	' '
6	myarrayCHARS[1]	Char	' '
7	myarrayCHARS[2]	Char	' '
8	myarrayCHARS[3]	Char	' '
9	myarrayCHARS[4]	Char	' '
10	myarrayCHARS[5]	Char	' '
11	myarrayCHARS[6]	Char	' '
12	myarrayCHARS[7]	Char	' '
13	myarrayCHARS[8]	Char	' '
14	myarrayCHARS[9]	Char	' '
15	countCHARS	UInt	0

Interconnect the parameters of the instruction as follows and select the data type of the character string.

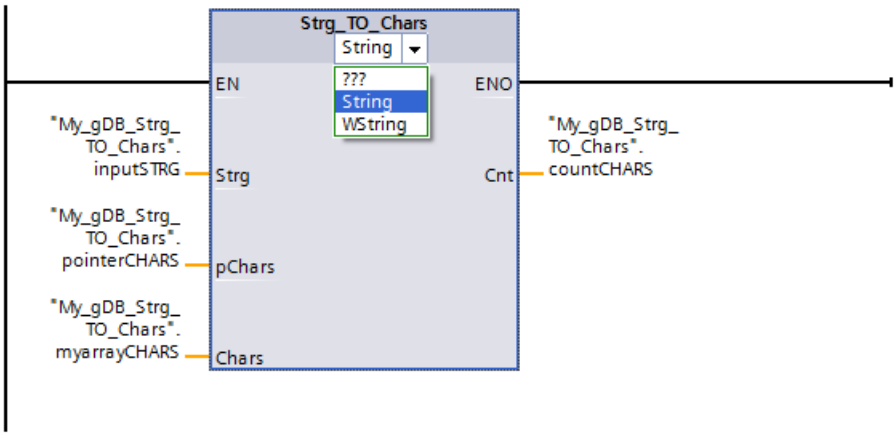

A structure made up of individual characters is created due to the Array of CHAR data type. The CHARS structure ("myarrayCHARS") is ten characters long (Array ... [0..9]). According to the value "2" of parameter PCHARS ("pointerCHARS"), writing begins starting from the third character in the structure ("0" and "1" are empty, "2" contains the first character of the character string ("inputSTRG")). Once the characters of the string ("inputSTRG") are written to the structure ("myarrayCHARS"), the last character of the structure to be created is written empty. The number of characters of the character string that are moved is output at output parameter CNT ("countCHARS").

My_gDB_Strg_TO_Chars

	Name	Data type	Start value	Monitor value
1	Static			
2	inputSTRG	String[10]	'cf7a7a#'	'cf7a7a#'
3	pointerCHARS	DInt	2	2
4	myarrayCHARS	Array[0..9] of Char		
5	myarrayCHARS[0]	Char	' '	' '
6	myarrayCHARS[1]	Char	' '	' '
7	myarrayCHARS[2]	Char	' '	'c'
8	myarrayCHARS[3]	Char	' '	'f'
9	myarrayCHARS[4]	Char	' '	'7'
10	myarrayCHARS[5]	Char	' '	'a'
11	myarrayCHARS[6]	Char	' '	'7'
12	myarrayCHARS[7]	Char	' '	'a'
13	myarrayCHARS[8]	Char	' '	'#'
14	myarrayCHARS[9]	Char	' '	' '
15	countCHARS	UInt	0	7

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

Chars_TO_Strg: Convert Array of CHAR to character string (S7-1200, S7-1500)

Description

With the "Chars_TO_Strg" instruction you copy characters from an Array of CHAR or Array of BYTE to a character string of the data type STRING or from an ARRAY of WCHAR or Array of WORD to a character string of the data type WSTRING. For the copy operation only ASCII characters are valid.

- Specify the characters of the Array of (W)CHAR / BYTE / WORD that will be copied to a character string at the input parameter CHARS.
- The characters are written at the parameter STRG to a (W)STRING data type.
  - The number of characters in the string is at least the same number as were copied from the source field.
  - If the character string is shorter than the number of characters in the source field, the characters are written up to the maximum length of the string.
  - If the Array of CHAR / BYTE contains a "$00" character or if the Array of WCHAR / WORD contains a W#16#0000 character, the copy operation is only carried out up to the corresponding position (see graphic).
- Using the PCHARS parameter, specify the position of the source field as of which the characters will be copied. PCHARS = 0 is the default value and always specifies the lower index limit of the array even if this is negative.
  - Example: if copying is to start with the third character of the source field, use the value "2" at parameter PCHARS:

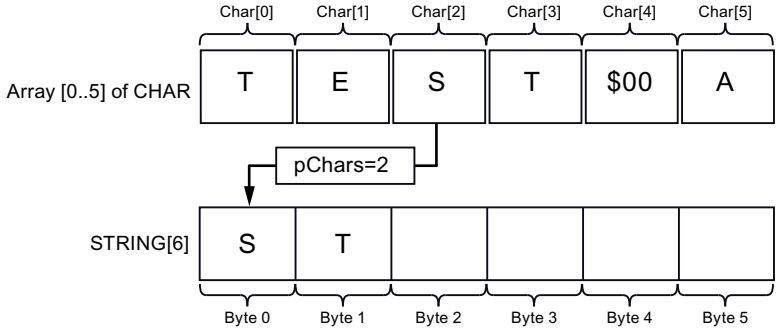

  - If an index is specified at parameter PCHARS and the index is not contained in the copy source (e.g. "7" at Array [0..5] of CHAR) then the instruction is not executed.

Note

Use of the instruction with S7-1200 V2.0

S7-1200 up to Version 2.0 only supports Array [0 .. n] of CHAR / BYTE. Negative index limits (e.g. Array [-3..2] of CHAR) are not permissible. This restriction is not checked by the software.

Parameters

The following table shows the parameters of the "Chars_TO_Strg" instruction:

Parameter	Declaration	Data type	Memory area	Description
CHARS	Input	VARIANT	D, L	Source of the copy operation Array of (W)CHAR / BYTE / WORD from which the characters are copied.
PCHARS	Input	DINT	I, Q, M, D, L, P or constant	Position in the Array of (W)CHAR / BYTE / WORD starting at which the characters are copied.
CNT	Input	UINT	I, Q, M, D, L, P or constant	Number of characters to be copied. Use "0" to copy all characters.
STRG	Output	STRING, WSTRING	D, L	Destination of the copy operation Character string with the (W)STRING data type. Observe the maximum length of the data types: • STRING: 254 characters • WSTRING: 254 characters (default) / 16382 characters (maximum) When using WSTRING, note that you must define a length > 254 characters explicitly with square brackets (e.g., WSTRING[16382]).

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you copy characters of a structure of the Array of CHAR data type to a character string with the STRING data type.

Create four tags in a global data block for storing the data.

My_gDB_Chars_TO_Strg

	Name	Data type	Start value
1	Static		
2	inputArrayCHARS	Array[0..9] of Char	
3	inputArrayCHARS[0]	Char	'M'
4	inputArrayCHARS[1]	Char	'y'
5	inputArrayCHARS[2]	Char	'S'
6	inputArrayCHARS[3]	Char	'7'
7	inputArrayCHARS[4]	Char	'P'
8	inputArrayCHARS[5]	Char	'L'
9	inputArrayCHARS[6]	Char	'C'
10	inputArrayCHARS[7]	Char	' '
11	inputArrayCHARS[8]	Char	' '
12	inputArrayCHARS[9]	Char	' '
13	pointerCHARS	DInt	2
14	outputSTRG	String	''
15	countCHARS	UInt	0

Interconnect the parameters of the instruction as follows and select the data type of the character string.

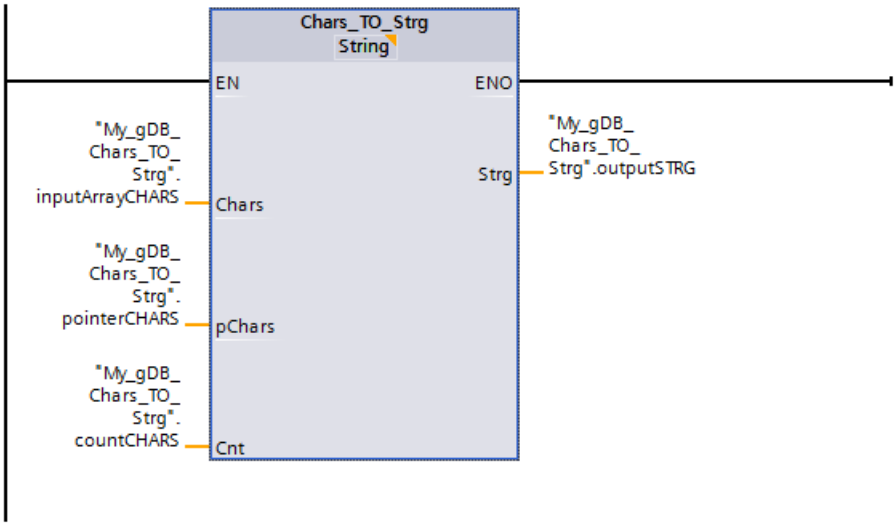

The CHARS structure ("inputArrayCHARS") is ten characters long (Array ... [0..9]). According to the value "2" of parameter PCHARS ("pointerCHARS"), characters are copied starting from the third position of the structure to the character string ("outputSTRG"). Starting from position "2", all characters of the structure ("inputArrayCHARS") are copied to the character string ("outputSTRG"), since parameter CNT ("countCHARS") has the value "0".

My_gDB_Chars_TO_Strg

	Name	Data type	Start value	Monitor value
1	Static			
2	inputArrayCHARS	Array[0..9] of Char		
3	inputArrayCHARS[0]	Char	'M'	'M'
4	inputArrayCHARS[1]	Char	'y'	'y'
5	inputArrayCHARS[2]	Char	'S'	'S'
6	inputArrayCHARS[3]	Char	'7'	'7'
7	inputArrayCHARS[4]	Char	'P'	'P'
8	inputArrayCHARS[5]	Char	'L'	'L'
9	inputArrayCHARS[6]	Char	'C'	'C'
10	inputArrayCHARS[7]	Char	' '	' '
11	inputArrayCHARS[8]	Char	' '	' '
12	inputArrayCHARS[9]	Char	' '	' '
13	pointerCHARS	DInt	2	2
14	outputSTRG	String	''	'S7PLC '
15	countCHARS	UInt	0	0

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

MAX_LEN: Determine the length of a character string (S7-1200, S7-1500)

Description

A tag of the (W)STRING data type contains two lengths: the maximum length and the current length (this is the number of currently valid characters).

- The maximum length of the character string is specified for each tag in the STRING keyword in square brackets. The number of bytes occupied by a string is 2 greater than the maximum length.
- The maximum length of the character string is specified for each tag in the WSTRING keyword in square brackets. The number of words occupied by a character string is 2 greater than the maximum length.
- The current length represents the number of the character places actually used. The current length must be less than or equal to the maximum length.

You use the "MAX_LEN" instruction to determine the maximum length of the character string specified at the IN input parameter This is output as numerical value at the output par OUT.

Note

Checking the character string

The "MAX_LEN" instruction does not check the character string at the parameter "IN".

If errors occur during processing of the instruction, the value "0" is output at the OUT parameter.

Note

Reading out the current length

You use the "LEN (Page 2165)" instruction to read out the current length of a string.

Parameter

The following table shows the parameters of the "MAX_LEN" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	• STRING • WSTRING	D, L or constant	Character string
OUT	Return	• INT • DINT	I, Q, M, D, L, P	Maximum number of characters

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you determine the maximum length of a character string with the STRING data type.

Create two tags in a global data block for storing the data.

My_gDB_MAX_LEN			
	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	stringMLengthOUT	Int	0

Interconnect the parameters of the instruction as follows.

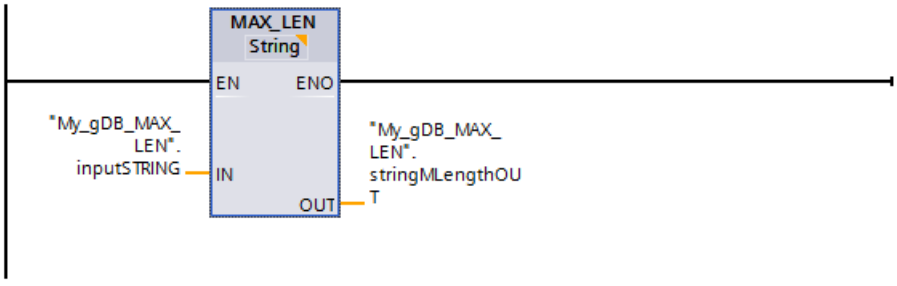

The maximum length of the specified character string ("inputSTRING") is determined and returned at output parameter OUT ("stringMLengthOUT") as a numerical value.

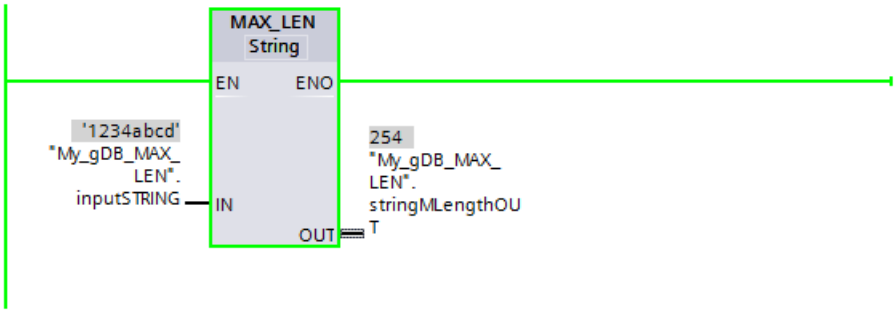

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

JOIN: Join multiple strings (S7-1500)

Description

The "JOIN" instruction connects multiple strings into an array.

To convert multiple strings into a single one, the instruction provides the following functions:

- **Format selection**
  You can use the first bit at the Mode parameter to specify whether the sequence of the source strings for an array should be in the format CSV or FSR.
  In the following example, two source strings are specified by the two columns of the table. The maximum number of characters amounts to 4 characters for the first source string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth.

1963	1974
Miller	Jackson
John	Peter
Roadname	VeryLongRoadna

  – With CSV (Comma Separated Values), the contents of the source strings are written consecutively into the destination array and separated with a delimiter. (See examples below.)
  – With FSR (Fixed Size Records), a certain number of characters is defined in the destination array for each source string. If the characters of a source string do not need the space reserved for them in the destination array, the corresponding array elements are filled with delimiters. However, if the number of characters in a source string is greater than the space reserved for them, the corresponding array elements are filled from the front, and the excess characters of the source string are truncated (see example below).

- **Selection of the delimiter for the source strings**
  The RecSeparator parameter is used to select the delimiter that will be used for each string. Select a character based on the content of the input strings at the SrcStruct parameter. If the input strings contain a comma within a string, for example, do not use the comma as a delimiter. The data type that you use for the delimiter must correspond to the destination array at the DstArray parameter As a result, the delimiters can also be written to the array.

- **Selection of the delimiter for the end of all source strings**
  You can use the third bit of the Mode parameter to specify whether an additional character will be written as a delimiter at the end of the copied characters in the destination array (DstArray parameter). At the EndSeparator parameter, specify the character that will be used as a delimiter. Make sure to use a character other than the one set for the RecSeparator parameter (delimiter for individual strings). If the two delimiters are indistinguishable, you will obtain unexpected results if you reverse the conversion using the "SPLIT" instruction.

- **Selection of the source strings**
  You specify the source strings at the SrcStruct parameter. For the data type, you can use Array of STRING or Array of WSTRING or structures which only contain the STRING or WSTRING data type. This also applies to user data types or nested structures. You can use these as long as they exclusively contain the data type STRING or WSTRING respectively.

- **Specification of the number of joined strings**
  If you use an Array of STRING or Array of WSTRING at the SrcStruct parameter (source string) (no nested structures), you can use the Count parameter to specify the number of source strings that are joined to form a single string. If you use a data type other than Array of (W)STRING at the SrcStruct parameter, the Count parameter is ignored. It is only possible to connect a larger part of an array in this way.

- **Selection of a destination area for writing the array**
  You use the Array of (W)CHAR data type at the DstArray parameter. In this case, it is not possible to use the STRING or WSTRING data types, because the length is limited to 254 characters or 256 bytes for STRING.
- **Index of the position in the array (DestArray target parameter)**
  The conversion starts at this position. The instruction reads the Position parameter to determine the position at which the conversion is completed. This enables follow-up calls of the instruction for filling the array.

Parameter

The following table shows the parameters of the "JOIN" instruction:

Parameter	Declaration	Data type	Memory area	Description
Mode	Input	DWORD	I, Q, M, D, L or constant	Specifies how the merger to string is performed (see Mode" parameter).
RecSeparator	Input	VARIANT	I, Q, M, D, L	Delimiter for the source strings • With CSV: Character that is used as a delimiter for the individual strings. • With FSR: Character that is used as a fill character for the individual strings.
EndSeparator	Input	VARIANT	I, Q, M, D, L	Delimiter for the end of the conversion Delimiter that is written at the end of the characters if bit 3 = 1 has been set for the Mode parameter.
SrcStruct	Input	VARIANT	I, Q, M, D, L	Pointer to the source strings.
Count	Input	UDINT	I, Q, M, D, L or constant	Number of strings that were joined. The Count parameter can only be used when an Array of (W)STRING has been used at the SrcStruct parameter.
DestArray	InOut	VARIANT	I, Q, M, D, L	Area in which the characters are written after the conversion. Use the Array [0 .. x] of CHAR/WCHAR data type at the DestArray parameter. You can set the length (x) of the array based on the length of the source strings at the SrcStruct parameter.
Position	InOut	UDINT	I, Q, M, D, L	Index the position in the total string
Ret_Val	Return	INT	I, Q, M, D, L	Status of instruction (see table "RET_VAL parameter").

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter Mode

Bit	Bit value "0"	Bit value "1"	Description
0	CSV format (Comma Separated Values)	FSR format (Fixed Size Record)	Selection of the format: • With CSV, the source strings are separated by a delimiter in the destination array. • With FSR, the source strings are written into the destination array with the fill characters defined at the RecSeparator parameter.
1	-	-	Irrelevant for the "JOIN" instruction.
2	-	-	Reserved (irrelevant for bit value)
3	Write no additional delimiters.	Write character defined by the EndSeparator parameter at the end of the read characters.	Select whether an additional character will be written as a delimiter at the end of the characters in the array (DestArray parameter).
4	-	-	Irrelevant for the "JOIN" instruction.

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error.
8190	Selection at the Mode parameter is not supported.
8x20	Source strings invalid.
8x53	VARIANT points to data structure that is too short.
8x54	Invalid data type
8082	The value at parameter Count is greater than the number of source character strings in SrcStruct.
8xB4	Different data types at parameters SrcStruct (source) and DestArray (destination) or at separators (parameters RecSeparator and EndSeparator).
80B5	Buffer overflow with the instruction. The characters are not completely output at parameter DestArray, or the value at parameter Position is outside of DestArray.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)

* Note the following for the error codes:

- The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".
- The "x" in the second position of the listed error codes stands for the parameter that caused the error.
  Example: Error code 8352 hex = error occurred at 3rd Parameter (EndSeparator), see Parameter table.
- If the error cannot be clearly assigned to a specific parameter, "0" is output.
  Example: The CHAR data type is used for the delimiter (RecSeparator parameter). WCHAR is used as data type for the array at the DestArray parameter. Error code 80B4 is output in this case.

Two examples for the JOIN instruction when the destination ARRAY should have the format CSV

- First example:
  The following source strings are available:
  - `1963`
  - `Miller`
  - `John`
  - `Roadname`

  If you select "," as the delimiter, calling JOIN provides the following destination array:
  `1963,Miller,John,Roadname`
- Second example:
  The following source strings are available:
  - `1974`
  - `Jackson`
  - `Peter`
  - `VeryLongRoadname`

  If you select "," as the delimiter, calling JOIN provides the following destination array:
  `1974,Jackson,Peter,VeryLongRoadname`

Two examples for the JOIN instruction when the destination ARRAY should have the format FSR

- First example:
  The following source strings are available:
  - `1963`
  - `Miller`
  - `John`
  - `Roadname`

  The number of reserved characters in the destination array amounts to 4 characters for the first source string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth.
  If you select "," as the fill characters, calling JOIN provides the following destination array:
  `1963Miller,,,,,,,John,,,,,,Roadname,,,,,,`
- Second example:
  The following source strings are available:
  - `1974`
  - `Jackson`
  - `Peter`
  - `VeryLongRoadname`

  The number of reserved characters in the destination array amounts to 4 characters for the first source string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth.
  If you select "," as the fill characters, calling JOIN provides the following destination array:
  `1974Jackson,,,,,,Peter,,,,,VeryLongRoadna`

See also

Switching display formats in the program status (Page 7698)

SPLIT: Splitting an array of characters into multiple strings (S7-1500)

Description

The "SPLIT" instruction converts an array (Array of CHAR / WCHAR) into several separate strings (Array of STRING / WSTRING or structure).

To convert the array into several strings, specify the following information:

- **Selection of array to be read**
  You specify the array to be read at the SrcArray parameter. Make sure that the data types used for the input and output parameters match any other parameters used. If you use an array with the CHAR data type at the SrcArray parameter, for example, you must use the CHAR data type for the delimiter (Rec/EndSeparator) and the structure must only contain strings with the STRING data type at the DestStruct parameter.
- **Format selection for the source array**
  You use the first bit of the Mode parameter to specify whether the array to be read has the CSV or FSR format.
  - With CSV (Comma Separated Values), characters in the source array that belong together are separated by a delimiter from the next characters that belong together.
    Two examples of a source array:
    `1963,Miller,John,CitynameA,Roadname`
    `1974,Jackson,Peter,CitynameB,VeryLongRoadname`
  - With FSR (Fixed Size Records), a certain number of characters is defined for each item of logical information of the source array. Each item of information should fit into the space defined for it. When information does not need the space defined for it, it is filled with delimiters.
    Two examples of source arrays, whereby the length of the information is 4 characters for the first information item (year of birth), 13 characters for the second (last name), 10 characters for the third (first name), 9 characters for the fourth (city) and 16 characters for the fifth (street):
    `1963Miller,,,,,,,John,,,,,,CitynameARoadname,,,,,,,,`
    `1974Jackson,,,,,,Peter,,,,,CitynameBVeryLongRoadname`
- **Delimiter used for the array to be read**
  - If the array to be read has CSV format, use the RecSeparator parameter to specify which delimiter was used.
  - If the array to be read has FSR format, use the RecSeparator parameter to specify which fill character was used.
- **Selection of the delimiter for the end of the total string**
  Specify the delimiter after which the reading of the array should stop at the EndSeparator parameter. The "SPLIT" instruction stops at this position and outputs the string that has been found. Note that the EndSeparator delimiter is evaluated with higher priority than the RecSeparator delimiter. If the delimiter at the EndSeparator parameter was used in a string to be read (between two RecSeparator delimiters), all content after the EndSeparator delimiter is ignored.
- **Specification of the position starting at which the array is read**
  The conversion starts at this position in the array and the instruction reads the Position parameter to determine the position at which the conversion is completed. This enables follow-up calls of the instruction for filling the various strings at the DestStruct parameter.
- **Outputting the number of read strings**
  If you use an Array of STRING at the DestStruct parameter, you can use the Count parameter to output the number of read strings. Only strings with content are counted. If you use a data type other than Array of STRING at the DestStruct parameter, "0" is output at the Count parameter.

Parameters

The following table shows the parameters of the "SPLIT" instruction:

Parameter	Declaration	Data type	Memory area	Description
Mode	Input	DWord	I, Q, M, D, L or constant	Specifies how the split into multiple strings is performed (see "Mode" parameter).
RecSeparator	Input	Variant	I, Q, M, D, L	Delimiter or fill character • With CSV: Character that was used in the array to be read to identify the individual strings. • With FSR: Character that has been used as a fill character in the array to be read.
EndSeparator	Input	Variant	I, Q, M, D, L	Delimiter used to define the end of the total string in the array to be read.
SrcArray	Input	Variant	I, Q, M, D, L	Pointer to the array to be read (Array of CHAR/WCHAR)
DestStruct	InOut	Variant	I, Q, M, D, L	Structure which includes the converted strings (Array of STRING / WSTRING).
Position	InOut	UDInt	I, Q, M, D, L	Position from which the array is to be read at the SrcArray parameter.
Ret_Val	Return	Int	I, Q, M, D, L	Result of instruction execution / error code (see table "Ret_Val parameter")
Count	Output	UDInt	I, Q, M, D, L	Number of strings that were found.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter "Mode"

Bit	Bit value "0"	Bit value "1"	Description
0	CSV format (Comma Separated Values)	FSR format (Fixed Size Record)	Basic mode: Selection of CSV or FSR
1	• With CSV: Additional characters cause an error. • With FSR: Additional fill characters remain in string.	• With CSV: Additional characters are ignored. • With FSR: Additional fill characters are removed.	Use bit 1 to specify how you want to deal with additional characters: • With CSV: – If the bit is set, additional characters that do not fit in the destination string are ignored. Example: The instruction writes to a string with a length of 16 characters (STRING[16] data type). No delimiter is contained in the source after the first 16 characters. If bit 1 is set, the extra characters are ignored and the instruction continues to read the array. – If the bit is not set, the instruction stops in such a case and generates an error message at the Ret_Val parameter. • With FSR: – If the bit is set, the fill characters located to the right of characters containing information are not written to the destination character string when the source array is transferred to the destination character strings (see example). – If the bit is not set, the fill characters located to the right of characters containing information are written to the destination character string when the source array is transferred to the destination character strings (see example).
2	-	-	Reserved for use in future versions.
3	EndSeparator is not counted.	EndSeparator is counted.	Specification of whether EndSeparator is counted for Position parameter.
4	Leave unwritten strings (STRING) at length.	Set unwritten strings (STRING) to length "0".	Specification of whether unused strings (STRING) at the DestStruct parameter are set to the length "0".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error.
0001	Additional characters are ignored.
0002	Only with CSV: • No separator EndSeparator was found in the array to be read (specified with SrcArray). Conversion was canceled for another reason, for example because DestStruct is completely full. • RecSeparator and EndSeparator are identical.
8190	Selection at the Mode parameter is not supported.
8x20	Source strings invalid.
8x53	VARIANT points to data structure that is too short.
8x54	Invalid data type

Error code* (W#16#...)	Explanation
8882	Position not in permissible range
8x84	Additional characters found.
8xB4	Different data types at parameters SrcArray (source) and DestStruct (destination) or at separators (parameters RecSeparator and EndSeparator).
80B5	The array SrcArray to be read is too small.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)

* Note the following for the error codes:

- The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".
- The "x" in the second position of the listed error codes stands for the parameter that caused the error.
  Example: Error code 8352 hex = error occurred at 3rd parameter (EndSeparator), see Parameter table.
- If the error cannot be clearly assigned to a specific parameter, "0" is output.
  Example: The CHAR data type is used for the delimiter (RecSeparator parameter). WCHAR is used as data type for the array at the DestArray parameter. Error code 80B4 is output in this case.

Two examples for the SPLIT instruction when the ARRAY to be read has the CSV format

In the first step, the source array is split into strings as specified by the delimiter (RecSeparator parameter, e.g. ",").

In the second step, the resulting strings after the split are stored in destination strings. You can use the DestStruct parameter to specify their length.

If the strings produced by the split have more characters than can fit in the destination string, the behavior of SPLIT after this point depends on the Mode parameter. With Mode=2#00, this results in an error: The associated error code is returned and no other destination strings are

filled. With Mode=2#10, the surplus characters are ignored and the next destination string is filled with the characters after the next delimiter.

- First example:
  In this example, there are these two source arrays:
  - `1963,Miller,John,Roadname`
  - `1974,Jackson,Peter,VeryLongRoadname`

  The length of the destination string amounts to 4 characters for the first destination string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth. The first call for SPLIT produces the following destination strings with Mode=2#10:
  - `1963`
  - `Miller`
  - `John`
  - `Roadname`

  The second call for SPLIT produces the following destination strings with Mode=2#10:
  - `1974`
  - `Jackson`
  - `Peter`
  - `VeryLongRoadna`
- Second example:
  In this example, the logical information in the source array is not in the order expected. There are these two source arrays:
  - `1963,Miller,Roadname,John`
  - `1974,Jackson,VeryLongRoadname,Peter`

  As in the first example, the length of the destination string amounts to 4 characters for the first destination string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth.
  The first call for SPLIT produces the following destination strings with Mode=2#10:
  - `1963`
  - `Miller`
  - `Roadname`
  - `John`

  The second call for SPLIT produces the following destination strings with Mode=2#10:
  - `1974`
  - `Jackson`
  - `VeryLongRo`
  - `Peter`

Two examples for the SPLIT instruction when the ARRAY to be read has format FSR

The source array is split into strings as you specified in the DestStruct parameter for the length of the destination string.

With regard to the fill characters (RecSeparator parameter e.g. ","), the behavior of SPLIT depends on the Mode parameter: With Mode=2#01, the fill characters are entered in the destination strings; with Mode=2#11, they are not entered.

- First example:
  In this example, there are these two source arrays:
  - `1963Miller,,,,,,,John,,,,,,Roadname,,,,,,`
  - `1974Jackson,,,,,,Peter,,,,,VeryLongRoadname`

  The length of the destination string amounts to 4 characters for the first destination string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth. The first call for SPLIT produces the following destination strings with Mode=2#01:
  - `1963`
  - `Miller,,,,,,,`
  - `John,,,,,,`
  - `Roadname,,,,,,`

  The second call for SPLIT produces the following destination strings with Mode=2#01:
  - `1974`
  - `Jackson,,,,,,`
  - `Peter,,,,,`
  - `VeryLongRoadna`
- Second example:
  In this example, the logical information in the source array is not in the order expected. There are these two source arrays:
  - `1963Miller,,,,,,,Roadname,,,,,,John,,,,,,`
  - `1974Jackson,,,,,,VeryLongRoadnamePeter,,,,,`

  As in the first example, the length of the destination string amounts to 4 characters for the first destination string, 13 characters for the second, 10 characters for the third and 14 characters for the fourth.
  The first call for SPLIT produces the following destination strings with Mode=2#11:
  - `1963`
  - `Miller`
  - `Roadname`
  - `,,,,John`

  Explanation: The SPLIT instruction lays templates over the source array, so to speak. The length of these templates is determined by the length of the destination strings. During transfer to the destination strings, only those commas to the right of characters with information content are interpreted as fill characters. The fill characters are not entered with Mode=2#11. In the first call, the commas are to the right of the "John" fill characters. The commas to the left of "John", however, are not fill characters, because the relevant "John" characters follow before the end of the corresponding destination string is reached. Therefore, the commas to the left of "John" appear in the destination string.
  The second call for SPLIT produces the following destination strings with Mode=2#11:

- 1974
- Jackson
- VeryLongRo
- adnamePeter

See also

Switching display formats in the program status (Page 7698)

JOIN: Join multiple strings (Page 2147)

ATH: Convert ASCII string to hexadecimal number (S7-1200, S7-1500)

Description

You use the instruction "ATH" to convert the ASCII character string specified at the IN input parameter into a hexadecimal number. The result of the conversion is output to the OUT output parameter.

- You can reference the following data types using the pointer at the IN parameter (ASCII): STRING, WSTRING, CHAR, BYTE, Array of CHAR, Array of BYTE, WCHAR, Array of WCHAR, Array of WORD.
- You can reference the following data types using the pointer at the OUT parameter (hexadecimal): Array of CHAR, Array of BYTE, Array of WORD, STRING, BYTE, CHAR, WORD, INT, DWORD, DINT, SINT, USINT, UINT, UDINT. Only with S7-1500: Array of WCHAR, WSTRING, WCHAR, ULINT, LINT, LWORD

You specify the number of ASCII characters to be converted with the N parameter. A maximum of 32767 valid ASCII characters can be converted. Only digits "0" to "9", upper case letters "A" to "F", and lower case letters "a" to "f" can be interpreted. All other characters are converted to zeros.

Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output word length is only half of the input word length. The ASCII characters are converted and positioned in the output in the same order as they are read in. If there is an odd number of ASCII characters, the hexadecimal number is padded with zeros in the nibble to the right of the last converted hexadecimal number.

Parameter

The following table shows the parameters of the "ATH" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	D, L	Pointer to ASCII character string
N	Input	INT	I, Q, M, D, L or constant	Number of ASCII characters to be converted
RET_VAL	Return	WORD	I, Q, M, D, L	Status of the instruction
OUT	Output	VARIANT	I, Q, M, D, L	Hexadecimal number

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#....)*	Description
0000	No error
0007	Invalid character. Only the following ASCII characters may be used: Digits "0" to "9", upper case letters "A" to "F", lower case letters "a" to "f".
8182	Input buffer is too small for data in the N parameter.
8120	Invalid format in the IN parameter.
8151	Non-supported data type in the IN parameter.
8482	Output buffer is too small for data in the N parameter.
8420	Invalid format in the OUT parameter.
8451	Non-supported data type in the OUT parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

ASCII characters and hexadecimal values

The following table shows the ASCII characters and the corresponding hexadecimal values:

ASCII character	ASCII-coded hexadecimal value	Hexadecimal digit
"0"	30	0
"1"	31	1
"2"	32	2
"3"	33	3
"4"	34	4
"5"	35	5
"6"	36	6
"7"	37	7
"8"	38	8
"9"	39	9
"A"	41	A
"B"	42	B
"C"	43	C
"D"	44	D
"E"	45	E
"F"	46	F

Example

The following table shows examples of the conversion of ASCII character strings to hexadecimal numbers:

Data type of IN	IN	N	OUT	ENO status
STRING, WSTRING	'0a23'	4	• WORD#16#0A23 • UINT#10#2595	1
STRING, WSTRING	‘123aFx1a23’	10	• LWORD#16#123AF01A_23000000 • ULINT#10#1 313 626 236 378 939 392	0
STRING, WSTRING	‘a23’	3	WORD#16#A230	1
STRING, WSTRING	‘0a23’	4	• WORD#[16#0A23, 16#0000] • BYTE#[16#0A, 16#23]	1
STRING, WSTRING	‘aFbE’	1	• CHAR#‘A’ • WCHAR#‘A’	1
STRING, WSTRING	‘0a23’	4	• CHAR#[‘0’, ‘A’, ‘2’, ‘3’] • WCHAR#[‘0’, ‘A’, ‘2’, ‘3’]	1
STRING, WSTRING	‘123aFC1a23’	10	• STRING#‘123AFC1A23’ • WSTRING#‘123AFC1A23’	1
ArrayOfCHAR, ArrayOfWCHAR	[‘a’, ‘B’, ‘E’, ‘3’]	4	WORD#16#ABE3	1
ArrayOfBYTE	[16#39, 16#32, 16#35]	3	WORD#16#9250	1
ArrayOfWORD	[16#1234, 16#0032, 16#0031, 16#0000]	3	WORD#16#0210	0
ArrayOfWORD	[16#0031, 16#0032, 16#0033, 16#0034]	4	WORD#16#1234	1
ArrayOfWORD	[16#AB31, 16#32AF, 16#4333, 16#0034]	4	WORD#16#0004	0
ArrayOfWORD	[16#0031, 16#0032, 16#0033, 16#0034]	4	[16#1234, 16#0000, 16#0000, 16#0000]	1
ArrayOfWORD	[16#0034, 16#0035, 16#0036, 16#0037, 16#0041, 16#0042, 16#0043, 16#0044]	8	DWORD#16#4567_ABCD	1

See also

Switching display formats in the program status (Page 7698)

HTA: Convert hexadecimal number to ASCII string (S7-1200, S7-1500)

Description

You use the instruction "HTA" to convert the hexadecimal number specified at the IN input into an ASCII character string. The result of the conversion is stored at the address specified in the OUT parameter.

- You can reference the following data types using the pointer in the IN parameter (hexadecimal): Array of CHAR, Array of BYTE, STRING, BYTE, CHAR, WORD, Array of WORD, INT, DWORD, Array of DWORD, DINT, SINT, USINT, UINT, UDINT. Only with S7-1500: Array of WCHAR, WSTRING, WCHAR, ULINT, LINT, LWORD, Array of LWORD
- You can reference the following data types using the pointer in the OUT parameter (ASCII): STRING, WSTRING, Array of CHAR, Array of WCHAR, Array of BYTE, Array of WORD

You specify the number of hexadecimal bytes to be converted with the N parameter. Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output value is twice as long as the input value. Each nibble of the hexadecimal number is converted to a character while maintaining the original order.

A maximum of 32767 characters are written to the ASCII character string. The result of the conversion is represented by the digits "0" to "9" and upper-case letters "A" to "F".

If the complete result of the conversion cannot be displayed in the OUT parameter, the result is will only be partially written to the parameter.

Parameter

The following table shows the parameters of the "HTA" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	I, Q, M, D, L	Start address of the hexadecimal digits
N	Input	UINT	I, Q, M, D, L or constant	Number of hexadecimal bytes to be converted
RET_VAL	Return	WORD	I, Q, M, D, L	Error message
OUT	Output	VARIANT	D, L	Address at which the result is stored.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0000	No error
8182	Input buffer is too small for data in the N parameter.
8120	Invalid format in the IN parameter.
8151	Non-supported data type in the IN parameter.

Error code* (W#16#....)	Description
8482	Output buffer is too small for data in the N parameter.
8420	Invalid format in the OUT parameter.
8451	Non-supported data type in the OUT parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

ASCII characters and hexadecimal values

The following table shows the ASCII characters and the corresponding hexadecimal values:

Hexadecimal digit	ASCII-coded hexadecimal value	ASCII character
0	30	"0"
1	31	"1"
2	32	"2"
3	33	"3"
4	34	"4"
5	35	"5"
6	36	"6"
7	37	"7"
8	38	"8"
9	39	"9"
A	41	"A"
B	42	"B"
C	43	"C"
D	44	"D"
E	45	"E"
F	46	"F"

Example

The following table shows examples of the conversion of hexadecimal numbers to ASCII character strings:

IN	N	OUT	Data type of OUT	ENO status
• WORD#16#0123 • UINT#10#291	2	'0123'	STRING, WSTRING	1
• DWORD#16#123AF012 • INT#10#305852434	4	'123AF012'	STRING, WSTRING	1
WORD#16#ABE3	2	CHAR#['A', 'B', 'E', '3']	ArrayOfCHAR, ArrayOfW-CHAR	1
WORD#16#9250	2	[16#39, 16#32, 16#35, 16#30]	ArrayOfBYTE	1
ArrayOfWORD#[16#123A, 16#0000, 16#ABCD, 16#0000]	2	[16#0031, 16#0032, 16#0033, 16#0041]	ArrayOfWORD	1

IN	N	OUT	Data type of OUT	ENO status
WORD#16#1234	2	[16#0031, 16#0032, 16#0033, 16#0034]	ArrayOfWORD	1
DWORD#16#4567ABCD	4	[16#0034, 16#0035, 16#0036, 16#0037, 16#0041, 16#0042, 16#0043, 16#0044]	ArrayOfWORD	1
ArrayOfD-WORD#[16#A800_0037, 16#0000_0041]	1	‘A8’	STRING, WSTRING	1
• CHAR#[‘0’, ‘A’, ‘2’, ‘3’] • WCHAR#[‘0’, ‘A’, ‘2’, ‘3’]	2	‘3041’	STRING, WSTRING	1
• STRING#‘123AFC1A23’ • WSTRING#‘123AFC1A23’	5	‘3132334146’	STRING, WSTRING	1
• Char#‘A‘ • Byte#16#41	1	‘41’	STRING, WSTRING	1

See also

Switching display formats in the program status (Page 7698)

LEN: Determine the length of a character string (S7-1200, S7-1500)

Description

A tag of the (W)STRING data type contains two lengths: the maximum length and the current length (this is the number of currently valid characters).

- The maximum length of the character string is specified for each tag in the STRING keyword in square brackets. The number of bytes occupied by a string is 2 greater than the maximum length.
- The maximum length of the character string is specified for each tag in the WSTRING keyword in square brackets. The number of words occupied by a character string is 2 greater than the maximum length.
- The current length represents the number of the character places actually used. The current length must be less than or equal to the maximum length.

You use the instruction "LEN" to query the current length of the character string specified at the IN input parameter. This is output as numerical value at the output par OUT. An empty string (‘’) has the length zero.

If errors occur during processing of the instruction, the value "0" is output at the OUT parameter.

Note

Reading out the maximum length

You use the "MAX_LEN (Page 2146)" instruction to read out the maximum length of a string.

Parameter

The following table shows the parameters of the "LEN" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
OUT	Return	INT	I, Q, M, D, L	Number of valid characters

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you determine the length of a character string with the STRING data type.

Create two tags in a global data block for storing the data.

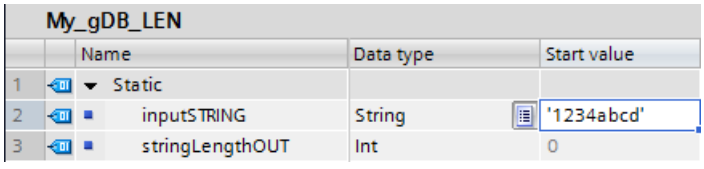

Interconnect the parameters of the instruction as follows.

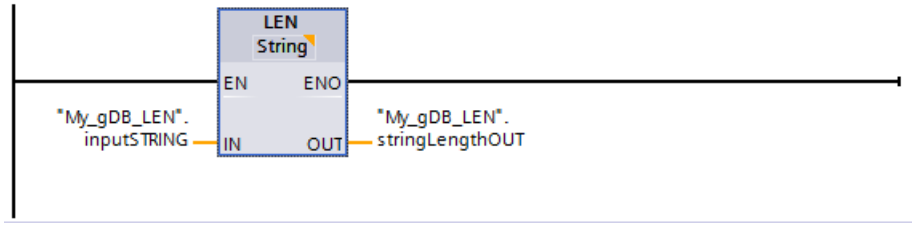

The number of actually occupied characters of the character string ("inputSTRING") is determined and returned as a numerical value at output parameter "OUT" ("stringLengthOUT").

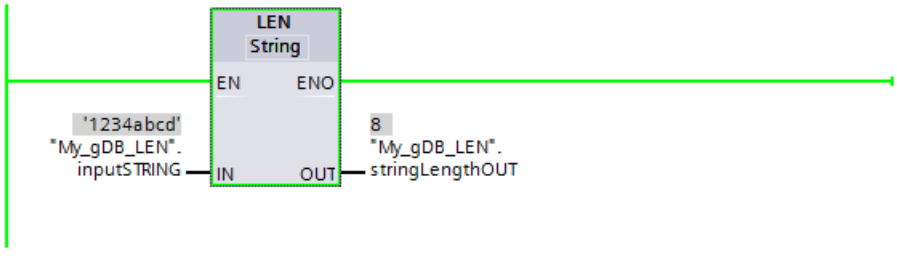

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

CONCAT: Combine character strings (S7-1200, S7-1500)

Description

You use the instruction "CONCAT" to combine the character string at the IN1 input parameter with the character string at the IN2 input parameter. The result is output at the OUT output parameter in (W)STRING format. If the resulting character string is longer than the tag specified in the OUT output parameter, then the resulting character string will be limited to the available length.

If errors occur during processing of the instruction and it is possible to write the OUT output parameter, an empty string will be output.

Parameters

The following table shows the parameters of the "CONCAT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING, WSTRING	D, L or constant	Character string
IN2	Input	STRING, WSTRING	D, L or constant	Character string
OUT	Return	STRING, WSTRING	D, L	Resulting character string

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you connect two character strings with the STRING data type.

Create three tags in a global data block for storing the data.

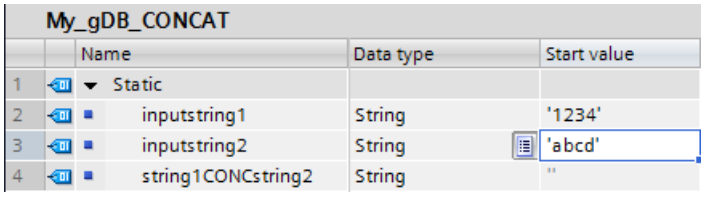

Interconnect the parameters of the instruction as follows.

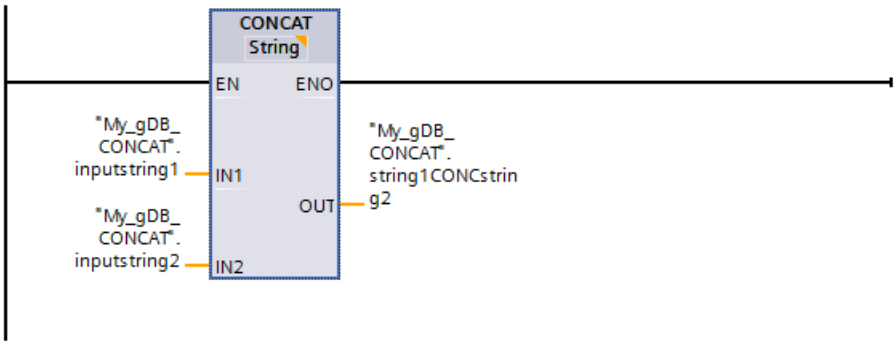

The characters of the second character string ("inputstring2") are appended to the first character string ("inputstring1") and the result is output at output parameter OUT ("string1CONCstring2").

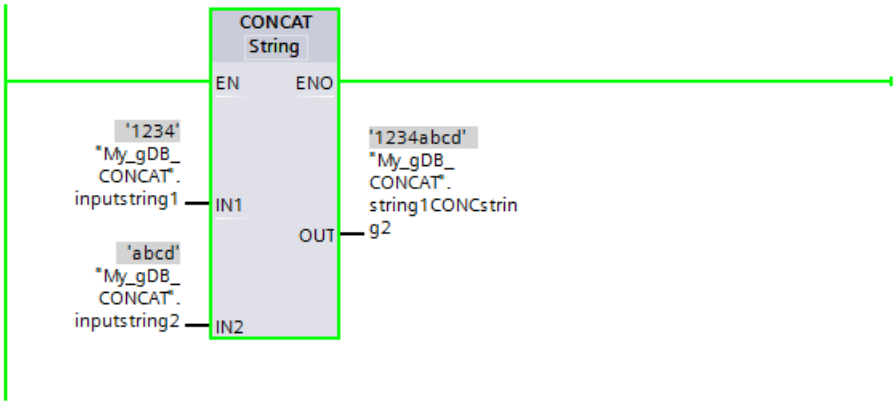

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

LEFT: Read the left character of a character string (S7-1200, S7-1500)

Description

You use the instruction "LEFT" to extract a partial string beginning with the first character of the string at the IN input parameter. You specify the number of characters to be extracted in the L parameter. The extracted characters are output at the OUT output parameter in (W)STRING format.

If the number of characters to be extracted is greater than the current length of the character string, the OUT output parameter returns the input character string as a result. If the L parameter contains the value "0" or the input value is an empty string, an empty string will be returned. If the value in the L parameter is negative, an empty string will be output.

If errors occur during processing of the instruction and it is possible to write the OUT output parameter, an empty string will be output.

Parameters

The following table shows the parameters of the "LEFT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be extracted
OUT	Return	STRING, WSTRING	D, L	Extracted partial string

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you extract a partial character string with the STRING data type from a character string starting from the first character.

Create three tags in a global data block for storing the data.

My_gDB_LEFT

		Name	Data type	Start value
1		Static		
2		inputSTRING	String	'1234abcd'
3		extractNumber	Int	4
4		outputExtractSTRING	String	''

Interconnect the parameters of the instruction as follows.

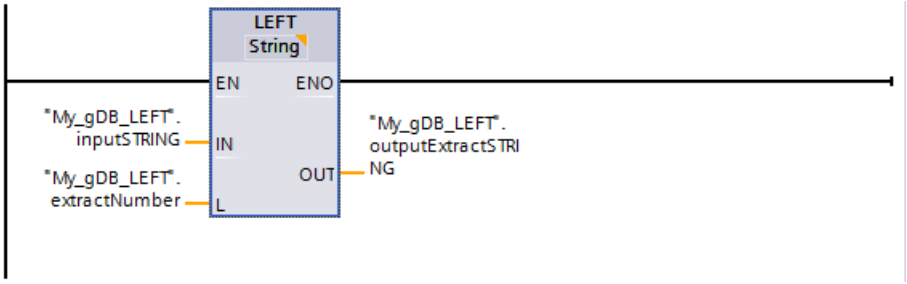

According to the value "4" of parameter L ("extractNumber"), you extract a four-character partial character string from the character string ("inputSTRING") starting from the first character from the left. The extracted partial character string is output at output parameter OUT ("outputExtractSTRING").

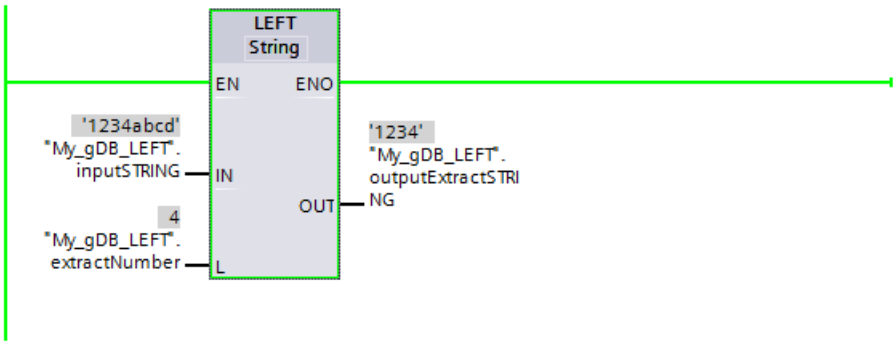

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

RIGHT: Read the right characters of a character string (S7-1200, S7-1500)

Description

You use this instruction to extract the last L character in a character string in the input parameter IN. You specify the number of characters to be extracted in the L parameter. The extracted characters are output at the OUT output parameter in (W)STRING format.

If the number of characters to be extracted is greater than the current length of the character string, the OUT output parameter returns the input character string as a result. If the L parameter contains the value "0" or the input value is an empty string, an empty string will be returned. If the value in the L parameter is negative, an empty string will be output.

If errors occur during processing of the instruction and it is possible to write the OUT output parameter, an empty string is output.

Parameters

The following table shows the parameters of the "RIGHT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be extracted
OUT	Return	STRING, WSTRING	D, L	Extracted partial string

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you extract a partial character string with the STRING data type from a character string starting from the last character.

Create three tags in a global data block for storing the data.

My_gDB_RIGHT			
	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	extractNumber	Int	4
4	outputExtractSTRING	String	''

Interconnect the parameters of the instruction as follows.

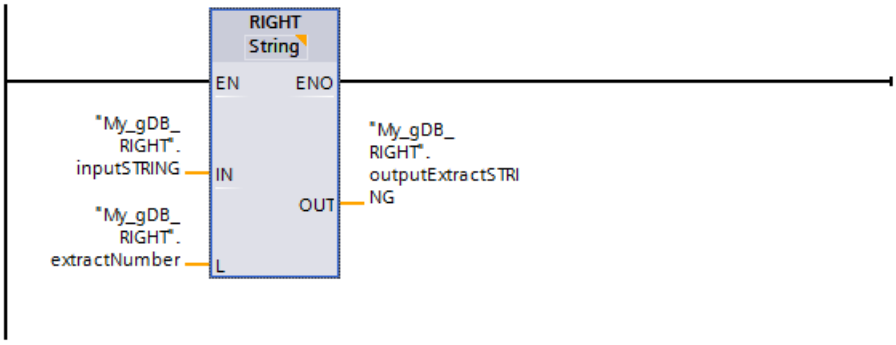

According to the value "4" of input parameter L ("extractNumber"), you extract a four-character partial character string from the character string ("inputSTRING") starting from the first character from the right. The extracted partial character string is output at output parameter OUT ("outputExtractSTRING").

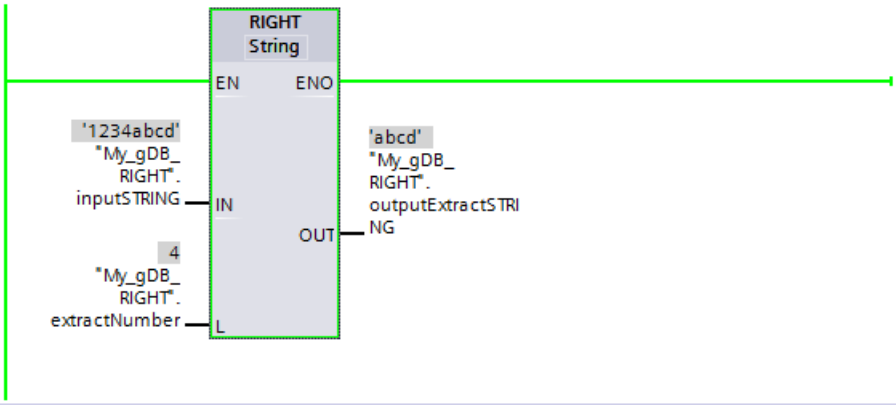

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

MID: Read middle characters of a character string (S7-1200, S7-1500)

Description

You use this instruction to extract a portion of the character string in the IN input parameter. With the P parameter, you specify the position of the first character to be extracted. With the L parameter, you define the length of the character string to be extracted. The extracted partial character string is output to the OUT output parameter.

Note the following rules when executing the instruction:

- If the number of characters to be extracted exceeds the current length of the character string in the IN input parameter, a partial character string will be output. The partial character string starts at the P character string and is continued until the end of the character string.
- If the character position specified in the P parameter falls outside the current character string length in the IN input parameter, an empty character string will be output in the OUT output parameter.
- If the value of the P or L parameter equals zero or is negative, an empty character string will be output in the OUT output parameter.

If errors occur during processing of the instruction and it is possible to write the OUT output parameter, an empty string will be output.

Parameters

The following table shows the parameters of the "MID" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Length of the string to be extracted
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of the first character to be extracted (first character = 1)
OUT	Return	STRING, WSTRING	D, L	Extracted partial string

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you extract a partial character string with the STRING data type from the middle of a character string.

Create four tags in a global data block for storing the data.

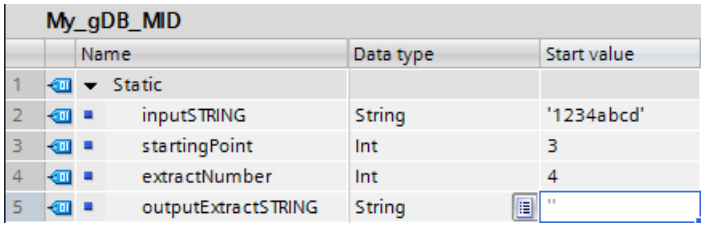

My_gDB_MID

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	startingPoint	Int	3
4	extractNumber	Int	4
5	outputExtractSTRING	String	''

Interconnect the parameters of the instruction as follows.

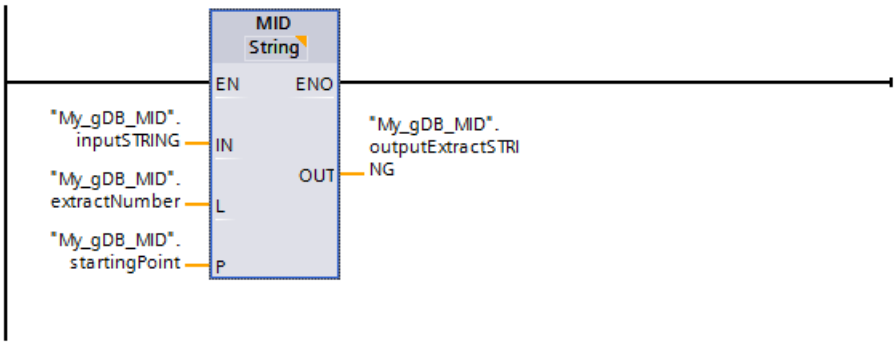

According to the value "4" of input parameter L ("extractNumber"), you extract a four-character partial character string from the character string ("inputSTRING"). The extraction starts from the third character ("startingPoint" has the value "3") of the character string ("inputSTRING"). The extracted partial character string is output at output parameter OUT ("outputExtractSTRING").

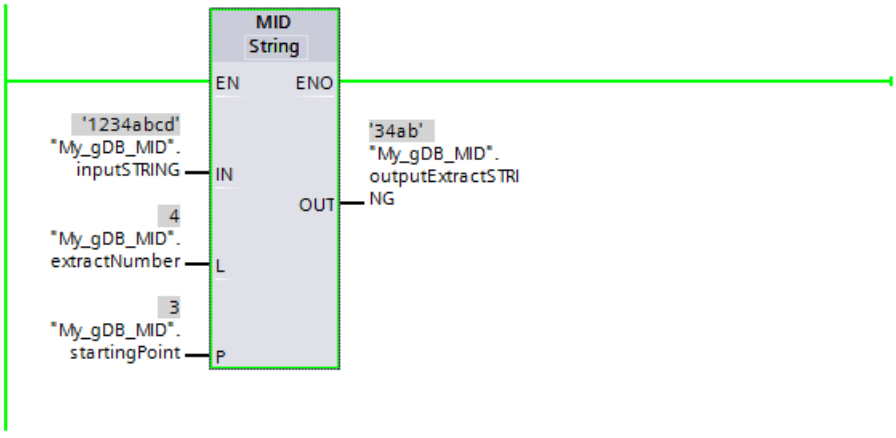

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

DELETE: Delete characters in a character string (S7-1200, S7-1500)

Description

You use this instruction to delete a portion of the character string in the IN input parameter. With the P parameter, you specify the position of the first character to be deleted. You specify the number of characters to be deleted in the L parameter. The remaining partial character string is output at the OUT output parameter in (W)STRING format.

Note the following rules when executing the instruction:

- If the value in the P parameter is less than or equals zero, an empty character string will be output in the OUT output parameter.
- If the value in the P parameter is greater than the current length of the character string in the IN input, the input character string will be returned in the OUT output parameter.

- If the value in the L parameter equals zero, the input character string will be returned in the OUT output parameter.
- If the number of characters to be deleted at the L parameter is greater than the length of the character string at the IN input parameter, the characters starting at the position specified by the P parameter are deleted. The string resulting from this is output.
- If the value in the L parameter is negative, an empty character string will be output.

If errors occur during processing of the instruction and it is possible to write the OUT output parameter, an empty string will be output.

Parameters

The following table shows the parameters of the "DELETE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING, WSTRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be deleted
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of first character to be deleted
OUT	Return	STRING, WSTRING	D, L	Resulting character string

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you delete characters from a character string with the STRING data type.

Create four tags in a global data block for storing the data.

My_gDB_DELETE

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	startingPoint	Int	3
4	deleteNumber	Int	4
5	outputResidualSTRING	String	''

Interconnect the parameters of the instruction as follows.

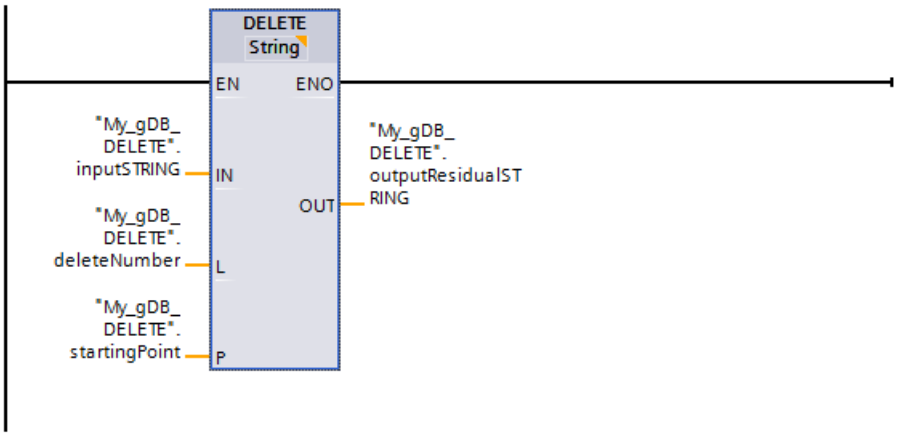

According to the value "4" of input parameter L ("deleteNumber"), you delete four characters from the character string ("inputSTRING") starting from the third character ("startingPoint" has the value "3"). The remaining character string is output at output parameter OUT ("outputResidualSTRING").

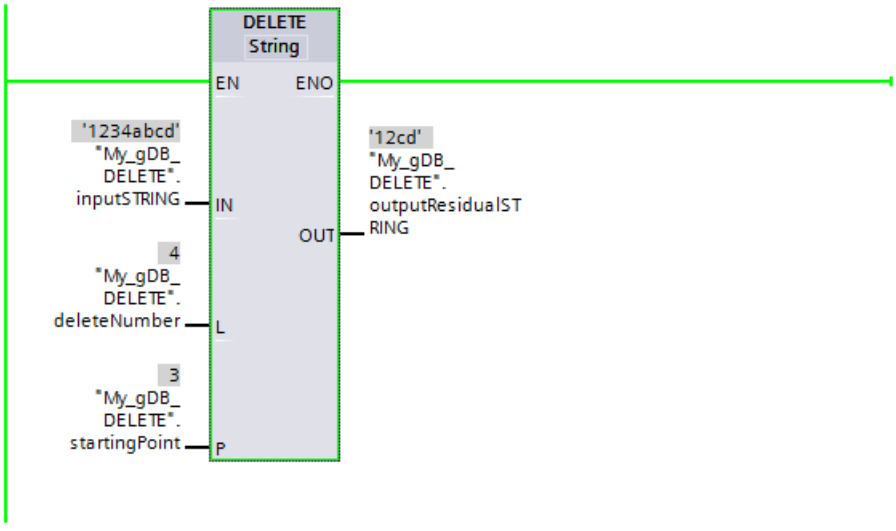

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

INSERT: Insert characters in a character string (S7-1200, S7-1500)

Description

You use this instruction to insert the character string in the IN2 input parameter to the character string in the IN1 input parameter. With the P parameter, you specify the position of the character starting at which the characters are inserted. The result is output at the OUT output parameter in (W)STRING format.

Note the following rules when executing the instruction:

- If the value in the P parameter exceeds the current length of the character string in the IN1 input parameter, the character string of the IN2 input parameter will be appended to the character string of the IN1 input parameter.
- If the value at the P parameter is zero, the character string at the OUT parameter followed by the character string at the IN2 parameter will be output in the IN1 parameter.
- If the value in the P parameter is negative, an empty character string will be output in the OUT output parameter.
- If the resulting character string is longer than the tag specified in the OUT output parameter, the resulting character string will be limited to the available length.

Parameters

The following table shows the parameters of the "INSERT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING, WSTRING	D, L or constant	Character string
IN2	Input	STRING, WSTRING	D, L or constant	String to insert
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Insert position
OUT	Return	STRING, WSTRING	D, L	Resulting character string

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you insert a character string in another character string. The data type used is STRING.

Create four tags in a global data block for storing the data.

My_gDB_INSERT

	Name	Data type	Start value
1	Static		
2	input1_STRING	String	'1234'
3	input2_STRING	String	'abcd'
4	startingPoint	Int	3
5	outputStrg1AndStrg2	String	''

Interconnect the parameters of the instruction as follows.

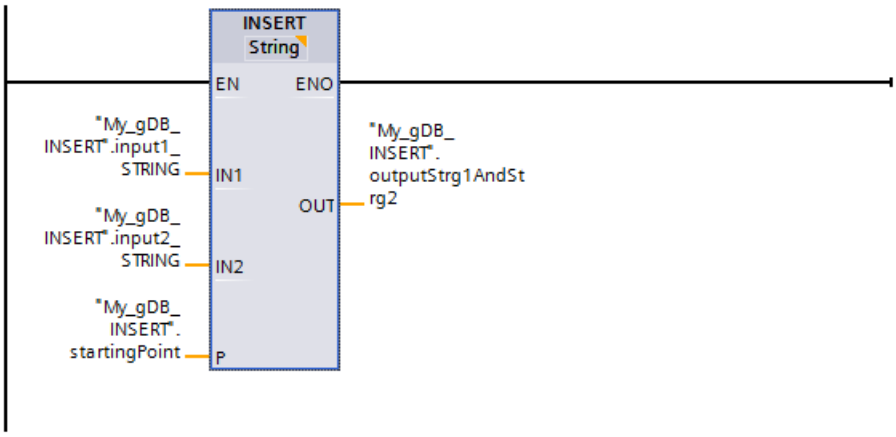

The characters of the second character string ("input2_STRING") are inserted in the first character string ("input1_STRING"). According to the value "3" of parameter P ("startingPoint"), characters are inserted after the third character of the first character string ("input1_STRING"). The result is output at output parameter OUT ("outputStrg1AndStrg2").

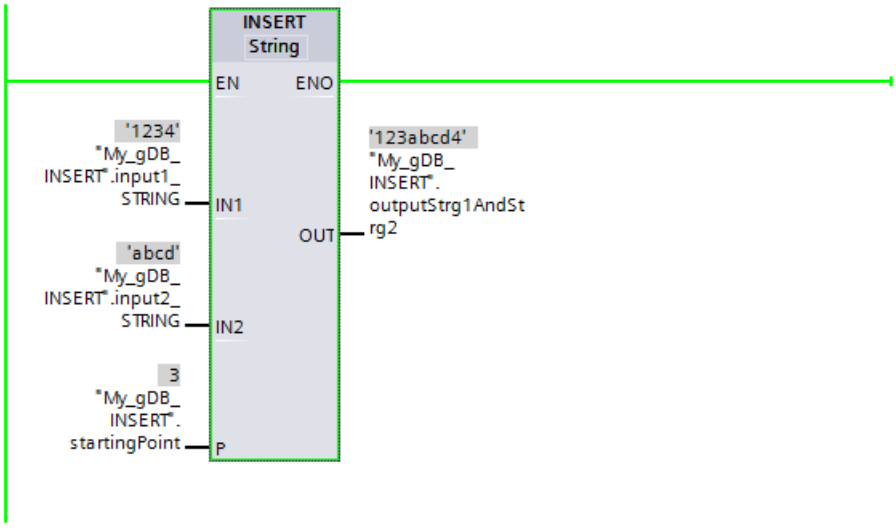

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

REPLACE: Replace characters in a character string (S7-1200, S7-1500)

Description

You use this instruction to replace a portion of the character string in the IN1 input with the character string in the IN2 input parameter. You specify the position of the first character to be replaced in the P parameter. You specify the number of characters to be replaced in the L parameter. The result is output at the OUT output parameter in (W)STRING format.

Note the following rules when executing the instruction:

- If the value in the P parameter is less than or equals zero, an empty character string will be output in the OUT output parameter.
- If the value in the L parameter is less than zero, an empty character string will be output in the OUT output parameter.
- If P equals one, the character string in the IN1 input will be replaced beginning with (and including) the first character.
- If the value in the P parameter exceeds the current length of the character string in the IN1 input parameter, the character string of the IN2 input parameter will be appended to the character string of the IN1 input parameter.
- If the resulting character string is longer than the tag specified in the OUT output parameter, the resulting character string will be limited to the available length.
- If the value at parameter L is zero, characters are not replaced but inserted. Similar conditions apply as with the instruction INSERT. See also: INSERT: Insert characters in a character string (Page 2175)

Parameters

The following table shows the parameters of the "REPLACE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING, WSTRING	D, L or constant	String with characters to be replaced.
IN2	Input	STRING, WSTRING	D, L or constant	String with characters to be inserted.
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be replaced
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of first character to be replaced
OUT	Return	STRING, WSTRING	D, L	Resulting character string

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you replace part of a character string with another character string. The data type used is STRING.

Create five tags in a global data block for storing the data.

My_gDB_REPLACE			
	Name	Data type	Start value
1	Static		
2	input1_STRING	String	'1234'
3	input2_STRING	String	'abcd'
4	replaceNumber	Int	2
5	startingPoint	Int	3
6	outputNewSTRING	String	''

Interconnect the parameters of the instruction as follows.

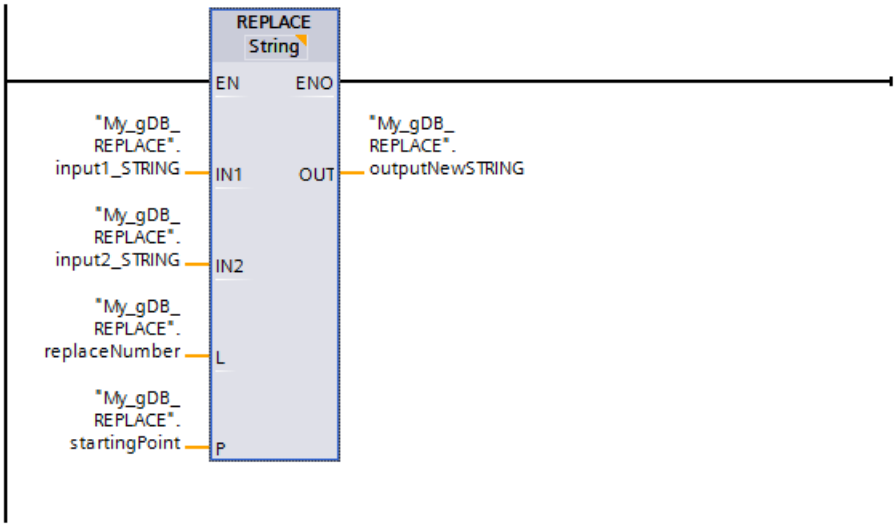

The second character string ("input2_STRING") is added to the first character string ("input1_STRING") starting from the third character ("startingPoint" has the value "3"). According to the value "2" of parameter L ("replaceNumber"), the third and fourth characters of the first character string ("input1_STRING") are replaced in the process. The result is output at output parameter OUT ("outputNewSTRING").

My_gDB_REPLACE				
	Name	Data type	Start value	Monitor value
1	Static			
2	input1_STRING	String	'1234'	'1234'
3	input2_STRING	String	'abcd'	'abcd'
4	replaceNumber	Int	2	2
5	startingPoint	Int	3	3
6	outputNewSTRING	String	''	'12abcd'

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

FIND: Find characters in a character string (S7-1200, S7-1500)

Description

You use this instruction to search through the character string at the IN1 input parameter for a specific character string.

- You specify the value to be searched for in the IN2 input parameter. The search is made from left to right.
- The position of the first occurrence is output in the OUT output parameter. If the search returns no match, the value "0" will be output in the OUT output parameter.

If an invalid character is specified at the IN2 parameter or if an error occurs during processing, the value "0" is output at the OUT parameter.

Parameters

The following table shows the parameters of the "FIND" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING, WSTRING	D, L or constant	String searched through
IN2	Input	STRING, WSTRING	D, L or constant	Character string that is searched for
OUT	Return	INT	I, Q, M, D, L	Character position

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you search for a character string in another character string. The data type used is STRING.

Create three tags in a global data block for storing the data.

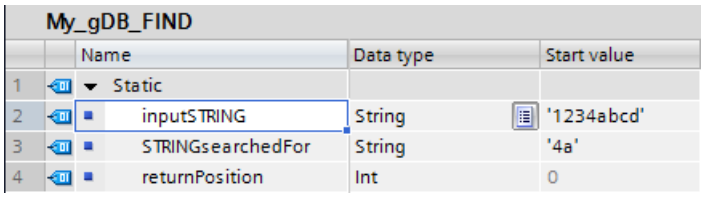

Interconnect the parameters of the instruction as follows.

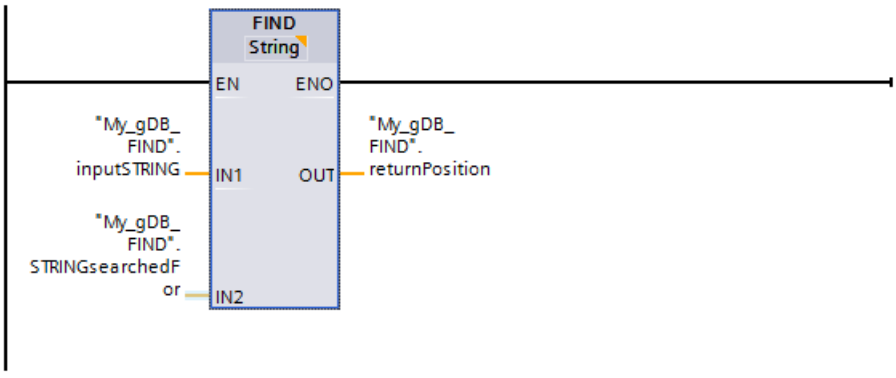

The first character string ("inputSTRING") searches for the value "4a" of the second character string ("STRINGsearchedFor"). The character position ("4") at which the searched character string begins is output at output parameter OUT ("returnPosition").

My_gDB_FIND

	Name	Data type	Start value	Monitor value
1	Static			
2	inputSTRING	String	'1234abcd'	'1234abcd'
3	STRINGsearchedFor	String	'4a'	'4a'
4	returnPosition	Int	0	4

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

Runtime information (S7-1200, S7-1500)

GetSymbolName: Read out a tag on the input parameter (S7-1200, S7-1500)

Description

With the "GetSymbolName" instruction, you use the input parameter of a block to read out the name of a tag interconnected at the VARIABLE parameter.

Note

Negated tag and the input parameter of a block

An empty string is always returned if you use "GetSymbolName" for the input parameter of a block at which a negated tag is interconnected.

Note

Increased code memory requirement

The use of GetSymbolName leads to an increased code memory requirement. The more frequently the instruction is called and the longer the actual symbols are, the greater the memory requirement.

If a block is used multiple times in the project and called each time by a different tag, you can use the "GetSymbolName" instruction to evaluate the name of the calling tag. The process value of the tag is irrelevant in this case.

- You specify the input parameters of the block interface at the VARIABLE parameter. Use only an interface parameter for this parameter and no PLC or data block tags.
- To limit the length of the read tag name, use the SIZE parameter. If the name has been truncated, this is indicated by the characters "..." (Unicode character 16#2026) appears at the end of the name. Note that this character has the length 1.
  An example to illustrate the meaning of the SIZE parameter. The following tag name is read out via the block interface: "MyPLCTag" (The double quotes at the start and end belong to the name.)

SIZE	GetSymbolName returns	Explanation
1	'...'	• First character of WSTRING:' • Identifier that the name was truncated: ... • Last character of WSTRING:'
2	'"...'	• First character of WSTRING:' • The first character of the name and identifier that the name was truncated:"... • Last character of WSTRING:'
3	'"M...'	• First character of WSTRING:' • The first two characters of the name and identifier that the name was truncated:"... "M... • Last character of WSTRING:'
6	'"MyPL...'	• First character of WSTRING:' • The first five characters of the name and identifier that the name was truncated: "MyPL... • Last character of WSTRING:'
0	'"MyPLCTag"'	• First character of WSTRING:' • All characters of the name: "MyPLCTag" • Last character of WSTRING:'

- The name read is output at the OUT parameter.

Parameter

The following table shows the parameters of the "GetSymbolName" instruction:

Parameter	Declaration	Data type	Memory area	Description
VARIABLE	Input	PARAMETER	Parameter sections Input, Output, InOut	Selection of the local interface to which you want to read out the name of the input parameter supply.
SIZE	Input	DINT	I, Q, M, D, L	Limits the number of characters output at the OUT parameter. • SIZE > 0: GetSymbolName returns the first SIZE characters of the name. • SIZE = 0: GetSymbolName returns the entire name. • SIZE < 0: GetSymbolName returns the last SIZE characters of the name.
OUT	Return	WSTRING	D, L	Output of the tag name of the input parameters supply

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you read out the name of a tag that is interconnected via the input parameter of a block.

Create two tags in a global data block for storing the data.

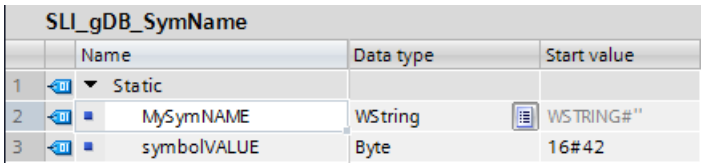

Create an FC "SLI_FC_GetSymbolName". Create the following local tag in the block.

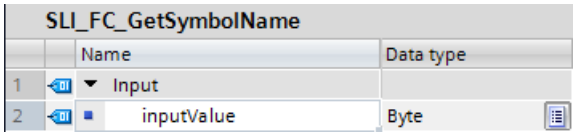

Call the instruction "GetSymbolName" in the FC and interconnect "GetSymbolName" as follows.

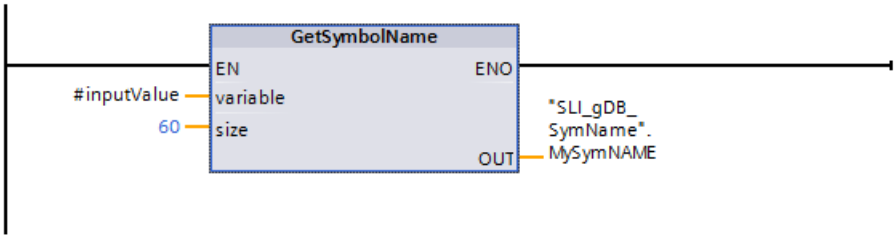

Create an FC "SLI_FC_FCCall_GetSymbolName". In the FC, call the FC "SLI_FC_GetSymbolName" and interconnect this as follows:

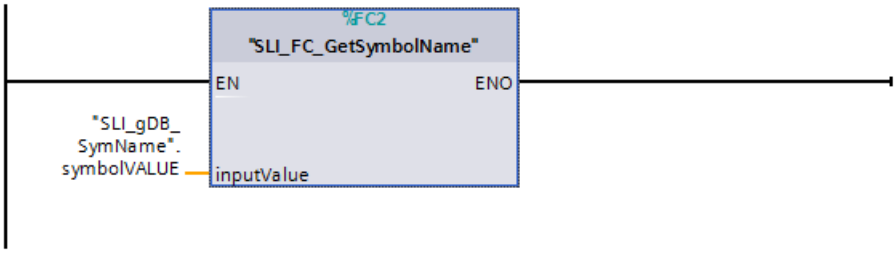

The "GetSymbolName" instruction is executed in the "SLI_FC_GetSymbolName" block. Input parameter inputValue of the "SLI_FB_GetSymbolName" block is examined for its interconnection using input parameter VARIABLE of the instruction. In doing so, the "symbolVALUE" tag is read out and output as a character string at output parameter OUT ("MySymNAME"). According to the value of input parameter SIZE, the length of the character string is limited to 60 characters.

SLI_gDB_SymName

	Name	Data type	Start value	Monitor value
1	Static			
2	MySymNAME	WString	WSTRING#''	WSTRING#'"SLI_gDB_SymName".symbolVALUE'
3	symbolVALUE	Byte	16#42	16#42

Program code

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

See also

Switching display formats in the program status (Page 7698)

Querying GetSymbolPath: actual parameter at beginning of a call path (S7-1200, S7-1500)

Description

Use the instruction "GetSymbolPath" to read the names of the parameters transferred at the beginning of a call path and passed via multiple block calls.

Note

Increased code memory requirement

The use of GetSymbolPath leads to an increased code memory requirement. The more frequently the instruction is called and the longer the actual symbols are, the greater the memory requirement.

- At the parameter VARIABLE of the instruction, you specify the formal parameter names from which the name is to be read when the interconnected actual parameter is called during runtime.
  - If the tag of a data block is used to supply the input parameters, this tag - consisting of the name of the DB and access path to the structured tag, is output.
  - If a PLC tag is used to supply the input parameters, the name of the PLC tag is output.
  - If a constant is used to supply the input parameters, this constant is output.
- To limit the length of the read tag name, use the SIZE parameter. If the name has been truncated, this is indicated by the character "..." (Unicode character 16#2026) at the end of the name. Note that this character has the length 1.
  An example to illustrate the meaning of the SIZE parameter. The following tag name is read out by the block interface: "MyPLCTag" (The double quotes at the start and end belong to the name.)

SIZE	GetSymbolPath returns	Explanation
1	'...'	• First character of WSTRING:' • Identifier that the name was truncated: ... • Last character of WSTRING:'
2	'"...'	• First character of WSTRING:' • The first character of the name and identifier that the name was truncated:"... • Last character of WSTRING:'
3	'"M...'	• First character of WSTRING:' • The first two characters of the name and identifier that the name was truncated:"... "M... • Last character of WSTRING:'
6	'"MyPL...'	• First character of WSTRING:' • The first five characters of the name and identifier that the name was truncated: "MyPL... • Last character of WSTRING:'
0	'"MyPLCTag"'	• First character of WSTRING:' • All characters of the name: "MyPLCTag" • Last character of WSTRING:'

Parameter

The following table shows the parameters of the "GetSymbolPath" instruction:

Parameter	Declaration	Data type	Memory area	Description
VARIABLE	Input	PARAMETER	Parameter sections Input, Output, InOut	Selection of the formal parameter for which you want to read out the name of the corresponding actual parameter at the beginning of the call path.
SIZE	Input	DINT	I, Q, M, D, L or constant	Limits the number of characters output at the OUT parameter. • SIZE > 0: GetSymbolPath returns the first SIZE characters of the name. • SIZE = 0: GetSymbolPath returns the entire name. • SIZE < 0: GetSymbolPath returns the last SIZE characters of the name.
OUT	Output	WSTRING	D, L	Output of the tag name of the input parameters supply.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

The following example shows the use of GetSymbolPath over several call levels:

- The FB_Level_1 block is called from the organization block OB1. In turn, the FB_Level_2 block is called from this.
- GetSymbolPath is executed in the FB_Level_2 block to read the name of the parameter whose value is passed at the beginning of the call path to REQ2.
- At the REQ1 parameter, the MyStarterBit tag is interconnected as actual parameter. The bit is located in the MySTRUCT structure in the MyDatablock data block.
  This information is read by GetSymbolPath and output at the OUT parameter.

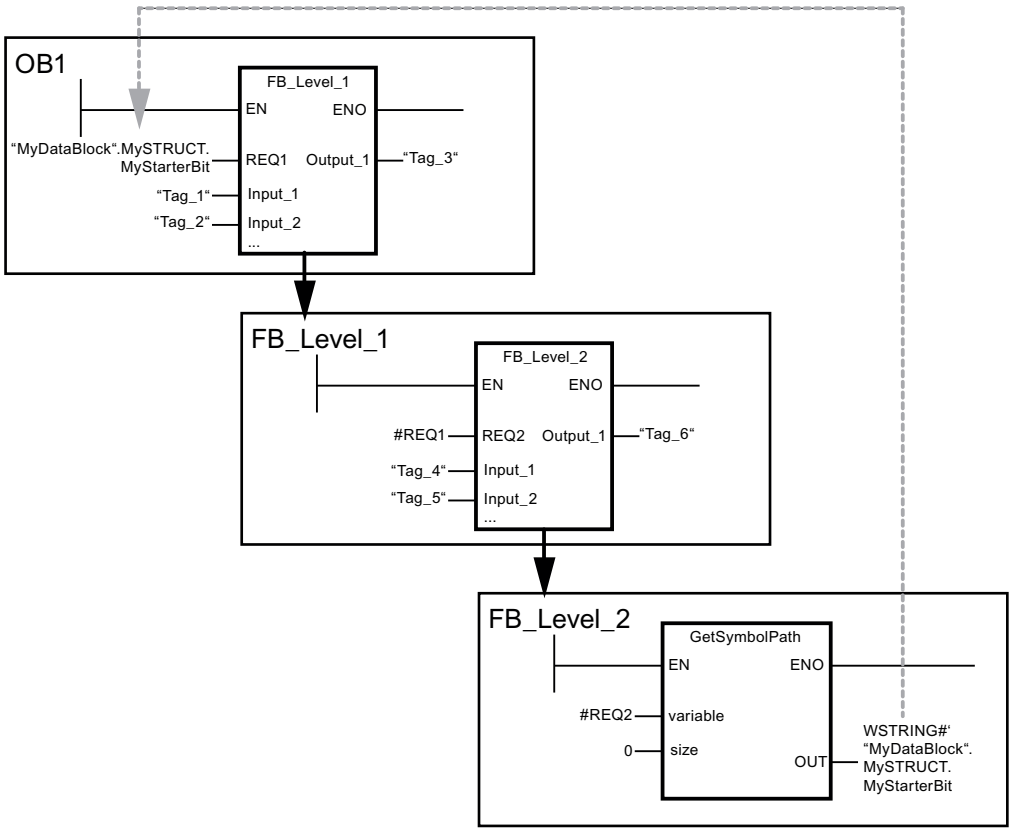

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

See also

Switching display formats in the program status (Page 7698)

GetInstanceName: Read out name of the read instance (S7-1200, S7-1500)

Description

You can use the "GetInstanceName" instruction to read the name of the instance data block within a function block.

Note

Increased code memory requirement

The use of GetInstanceName leads to an increased code memory requirement. It becomes even greater the more frequently the instruction is called, the longer the name of the instance data block is and the less you limit the length.

- To limit the length of the read instance name, use the SIZE parameter. If the name has been truncated, this is indicated by the character "..." (Unicode character 16#2026) at the end of the name. Note that this character has the length 1.
  An example to illustrate the meaning of the SIZE parameter. The following instance name is read out via the block interface: "Level1_DB" (The double quotes at the start and end belong to the name.)

SIZE	Returns GetInstanceName	Explanation
1	'...'	• First character of WSTRING:' • Identifier that the name was truncated: ... • Last character of WSTRING:'
2	'"...'	• First character of WSTRING:' • The first character of the name and identifier that the name was truncated:"... • Last character of WSTRING:'
3	'"L...'	• First character of WSTRING:' • The first two characters of the name and identifier that the name was truncated:"... "L... • Last character of WSTRING:'
6	'"Leve...'	• First character of WSTRING:' • The first five characters of the name and identifier that the name was truncated: "Leve... • Last character of WSTRING:'
0	'"Level1_DB"'	• First character of WSTRING:' • All characters of the name: "Level1_DB" • Last character of WSTRING:'

- The name of the instance data block is written to the OUT parameter. If the name of the instance data block is longer than the maximum length of WSTRING, the name is truncated.

Parameter

The following table shows the parameters of the "GetInstanceName" instruction:

Parameter	Declaration	Data type	Memory area	Description
SIZE	Input	DINT	I, Q, M, D, L or constant	Limits the number of characters output at the OUT parameter. • SIZE > 0: GetInstanceName returns the first SIZE characters of the name. • SIZE = 0: GetInstanceName returns the entire name. • SIZE < 0: GetInstanceName returns the last SIZE characters of the name.
OUT	Output	WSTRING	D, L	Read name of the instance data block

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you read out the name of an instance data block.

Create two tags in a global data block for storing the data.

SLI_gDB_GetInstanceName			
	Name	Data type	Start value
1	Static		
2	limitSIZE	DInt	0
3	outputInstName	WString	WSTRING#''

Create an FB "SLI_FB_GetInstanceName". Interconnect the parameters of the "GetInstanceName" instruction as follows.

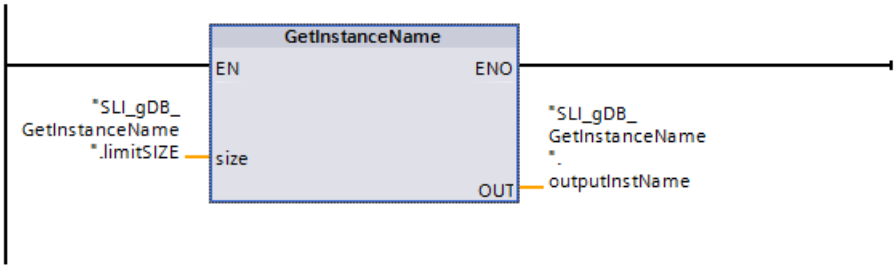

The "GetInstanceName" instruction is executed in the "SLI_FB_GetInstanceName" block. By means of the "GetInstanceName" instruction, the associated instance data block of the "SLI_FB_GetInstanceName" block is determined and output as a character string at output parameter OUT ("outputInstName"). According to the value "0" of parameter SIZE ("limitSIZE"), the length of the character string is unlimited.

SLI_gDB_GetInstanceName				
	Name	Data type	Start value	Monitor value
1	Static			
2	limitSIZE	DInt	0	0
3	outputInstName	WString	WSTRING#''	WSTRING#'"SLI_FB_GetInstanceName_DB"'

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

See also

Switching display formats in the program status (Page 7698)

GetInstancePath: Query composite global name of the block instance (S7-1200, S7-1500)

Description

You use the "GetInstancePath" instruction to read the composed global name of the block instance within a function block. The composed global name of the block instance is the path of the complete call hierarchy when multi-instances are used.

Note

Increased code memory requirement

The use of GetInstancePath leads to an increased code memory requirement. The longer the name and the higher the number of parameters, the greater the memory requirement.

In the following example, the "GetInstancePath" instruction is called in the FB_Level_3 function block.

- The FB_Level_3 function block stores its data in the calling FB_Level_2 function block.
- The FB_Level_2 function block in turn stores its data in the calling FB_Level_1 function block.
- The FB_Level_1 function block in turn stores its data in its instance data block IDB_LEVEL_1. Through the use of multi-instances, the instance data block of FB_Level_1 contains all data of the three function blocks.

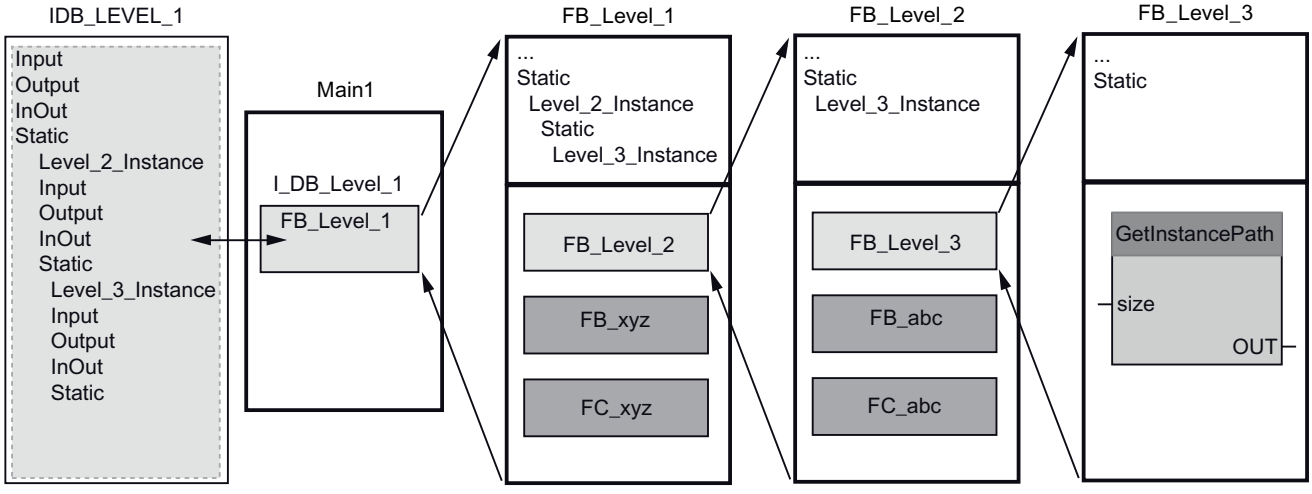

The "GetInstancePath" instruction returns the following values for this example:

SIZE	GetInstancePath returns	Explanation
1	'...'	• First character of WSTRING:' • Identifier that the name was truncated: ... • Last character of WSTRING:'
2	'"...'	• First character of WSTRING:' • The first character of the name and identifier that the name was truncated:"... • Last character of WSTRING:'
3	'"I...'	• First character of WSTRING:' • The first two characters of the name and identifier that the name was truncated:"... "I... • Last character of WSTRING:'
6	'"IDB_...'	• First character of WSTRING:' • The first five characters of the name and identifier that the name was truncated: "IDB_... • Last character of WSTRING:'
0	'"IDB_LEVEL_1".Level_2_ Instance.Level_3_Instance'	• First character of WSTRING:' • All characters of the name: "IDB_LEVEL_1".Level_2_Instance.Level_3_Instance • Last character of WSTRING:'

Note

Use of "GetInstancePath" in function blocks with single instance

If the function block in which you call "GetInstancePath" saves data in its own instance data block, the name of the single instance is output as the global name. The result at parameter OUT corresponds in this case to the ""GetInstanceName (Page 2187)" instruction.

Parameter

The following table shows the parameters of the "GetInstancePath" instruction:

Parameter	Declaration	Data type	Memory area	Description
SIZE	Input	DINT	I, Q, M, D, L or constant	Limits the number of characters output at the OUT parameter. • SIZE > 0: GetInstancePath returns the first SIZE characters of the name. • SIZE = 0: GetInstancePath returns the entire name. • SIZE < 0: GetInstancePath returns the last SIZE characters of the name.
OUT	Output	WSTRING	D, L	Read global name of the block instance. If the global name of the block instance is longer than the maximum length of WSTRING (254 characters), the name is truncated.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following you read out the assembled global names of the block instance.

Create one tag for storing the data in a global data block.

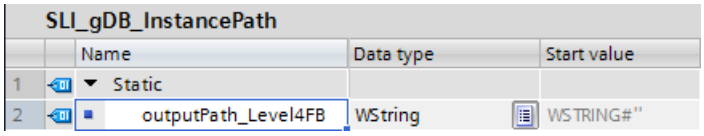

Create an FB "SLI_FB_GIP_Level4" and call the instruction "GetInstancePath" in it.

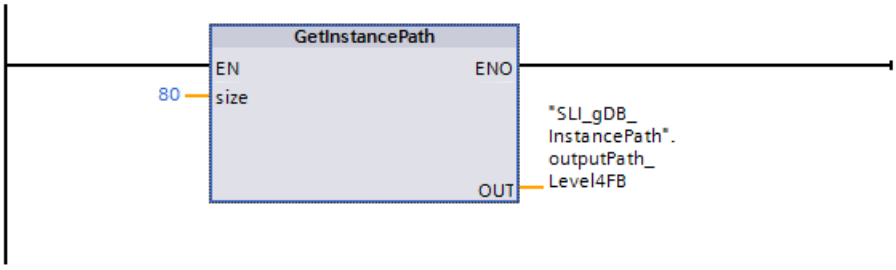

Create an FB "SLI_FB_GIP_Level3" and call the instruction "SLI_FB_GIP_Level4" in it.

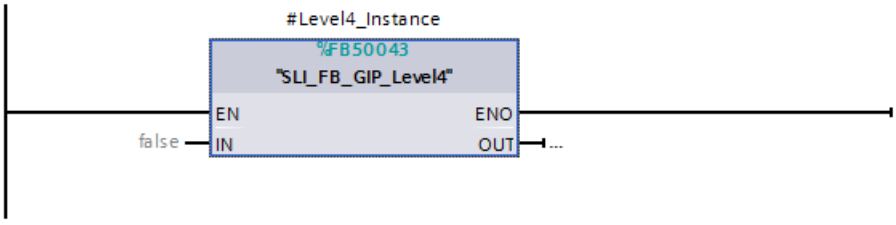

Create an FB "SLI_FB_GIP_Level2" and call the instruction "SLI_FB_GIP_Level3" in it.

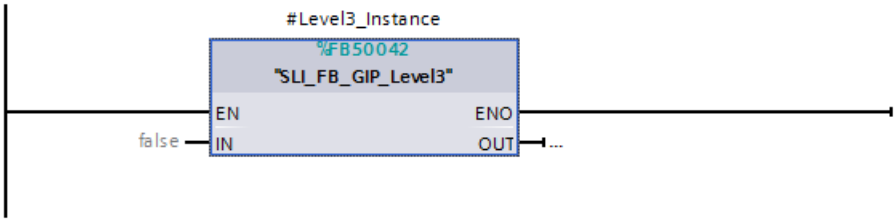

Create an FB "SLI_FB_GIP_Level1" and call the instruction "SLI_FB_GIP_Level2" in it.

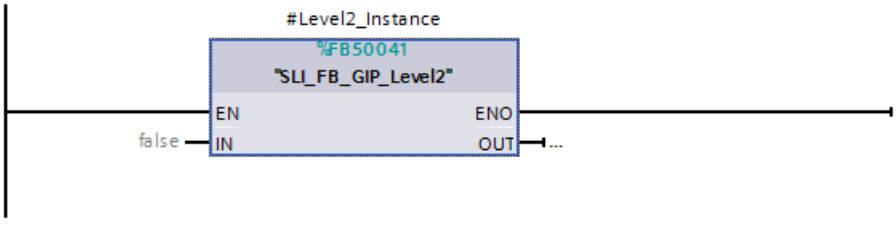

Create the following local tags in the function block "SLI_FB_GIP_Level1".

SLI_FB_GIP_Level1

		Name	Data type	Default value
1		Input		
2		IN	Bool	false
3		inputValue	Byte	16#0
4		Output		
5		OUT	Bool	false
6		InOut		
7		<Add new>		
8		Static		
9		Level2_Instance	"SLI_FB_GIP_Level2"	

Create a cycle OB (OB123" and call the instruction "SLI_FB_GIP_Level1" in it. Interconnect the function block as follows.

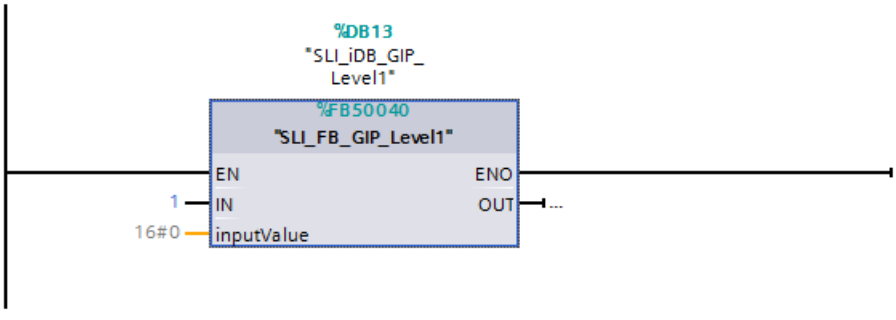

The instruction "GetInstancePath" is executed in the OB123. The instruction "GetInstancePath" reads out the instances of all function blocks, from call of instruction in the FB "SLI_FB_GIP_Level4" to the call in the OB123. The names of the instances are output as strings at the output parameter OUT ("outputPath_Level4FB") of the instruction "GetInstancePath". The length of the character string is limited to 80 characters, according to the value of input parameter SIZE of the operation "GetInstancePath".

SLI_gDB_InstancePath

		Name	Data t..	St...	Monitor value
1		Static			
2		outputPath_Level4FB	W...	WSTR	WSTRING#'"SLI_iDB_GIP_Level1".Level2_Instance.Level3_Instance.Level4_Instance'

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

GetBlockName: Read out name of the block (S7-1200, S7-1500)

Description

You can use the "GetBlockName" instruction to read the name of the block in which the instruction is called.

- If the length of the block name is to be limited to a certain number of characters, specify the max. length at the SIZE parameter. If the name has been truncated, this is indicated by the character "..." (Unicode character 16#2026) at the end of the name. Note that this character has the length 1.
  An example to illustrate the meaning of the SIZE parameter. The following block name is read out: "Level1_gbn" (The double quotes at the start and end belong to the name.)

SIZE	GetBlockName returns	Explanation
1	'...'	• First character of WSTRING:' • Identifier that the name was truncated: ... • Last character of WSTRING:'
2	'"...'	• First character of WSTRING:' • The first character of the name and identifier that the name was truncated:"... • Last character of WSTRING:'
3	'"L...'	• First character of WSTRING:' • The first two characters of the name and identifier that the name was truncated:"... "L... • Last character of WSTRING:'
6	'"Leve...'	• First character of WSTRING:' • The first five characters of the name and identifier that the name was truncated: "Leve... • Last character of WSTRING:'
0	'"Level1_gbn"'	• First character of WSTRING:' • All characters of the name: "Level1_gbn" • Last character of WSTRING:'

- The name of the block is written at the RET_VAL parameter. If the name of the block is longer than the maximum length of WSTRING, the name is truncated.

Parameter

The following table shows the parameters of the "GetBlockName" instruction:

Parameter	Declaration	Data type	Memory area	Description
SIZE	Input	UINT	I, Q, M, D, L or constant	Limits the number of characters output at the RET_VAL parameter. • SIZE > 0: GetBlockName returns the first SIZE characters of the name. • SIZE = 0: GetBlockName returns the entire name. • SIZE < 0: GetBlockName returns the last SIZE characters of the name.
RET_VAL	Output	WSTRING	D, L	Read name of the instance data block

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

In the following example, you read out the block name.

Create two tags in a global data block for storing the data.

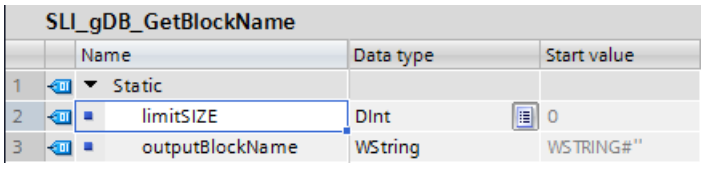

Interconnect the parameters of the instruction as follows.

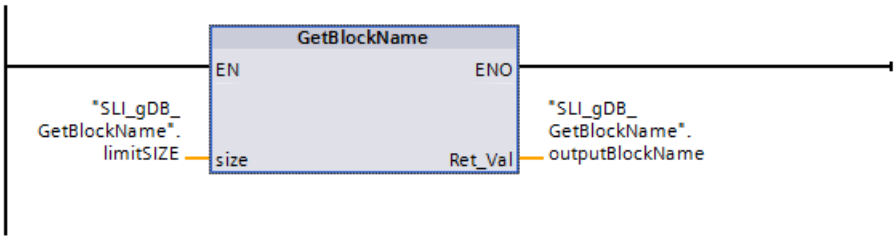

The "GetBlockName" instruction is executed in the "SLI_FC_GetBlockName" block. By means of the "GetBlockName" instruction, the name of the "SLI_FC_GetBlockName" block is read out and output as a character string at output parameter RET_VAL ("outputBlockName"). According to the value "0" of parameter SIZE ("limitSIZE"), the length of the character string is unlimited.

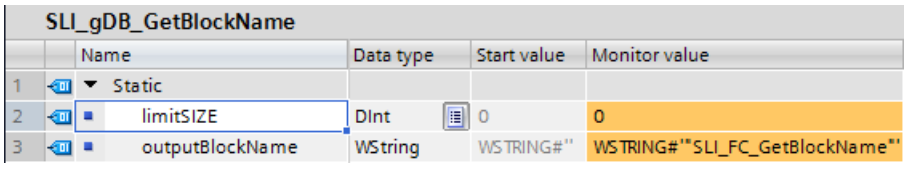

Program code

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

See also

Switching display formats in the program status (Page 7698)

4.1.4.3 Process image (S7-1500)

UPDAT_PI: Update the process image inputs (S7-1500)

Description

With the instruction, you update a partial image partition of the inputs defined in the configuration.

If you have configured repeated signaling of I/O access errors for the system-side process image update, then the selected process image will be updated constantly.

Otherwise, the update can only be performed when the selected process image partition is not updated by the system; in other words if you have not assigned this process image partition to an interrupt OB.

Note

Each logical address that you have assigned to a process image partition of the inputs per configuration no longer belongs to the OB 1 process image of the inputs.

When you update a process image partition with "UPDAT_PI", you must not perform a simultaneous update with the "SYNC_PI (Page 2203)" instruction.

System-side updating of the OB 1 process image of the inputs and the process image partitions of inputs that you have assigned to an interrupt OB takes place independently of "UPDAT_PI" calls.

Parameter

The following table shows the parameters of the "UPDAT_PI" instruction:

Parameter	Declaration	Data type	Memory area	Description
PART	Input	PIP	I, Q, M, D, L or constant	Number of the process image partition of the inputs to be updated. Maximum value range (depending on the CPU): 1 <= n <= 31 (means process image partition n).
RET_VAL	Return	INT	I, Q, M, D, L, P	Error information
FLADDR	Output	WORD	I, Q, M, D, L, P	Address of the first byte to cause an error if an access error has occurred.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for PART parameter
8091	The specified process image partition was not yet defined.
8092	The process image partition is being updated by the system and must not be used for the PART parameter.
8093	The process image partition update is being processed in another OB.
80A0	An access error was detected when accessing the I/O.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Note

If you use the instruction for process image partitions of DP standard slaves, the error codes of the "DPRD_DAT (Page 2338)" instruction are also possible. This is the case if you have defined consistency areas greater than 32 bytes.

Example

In the following example, you update the process image partition "1" of the inputs defined in the configuration.

Create four tags in a global data block for storing the data.

SLI_gDB_UPDAT_PI

		Name	Data type	Start value
1		Static		
2		part	PIP	1
3		returnValue	Int	0
4		errAddress	Word	16#0
5		updateOk	Bool	false

Interconnect the parameters of the instruction as follows.

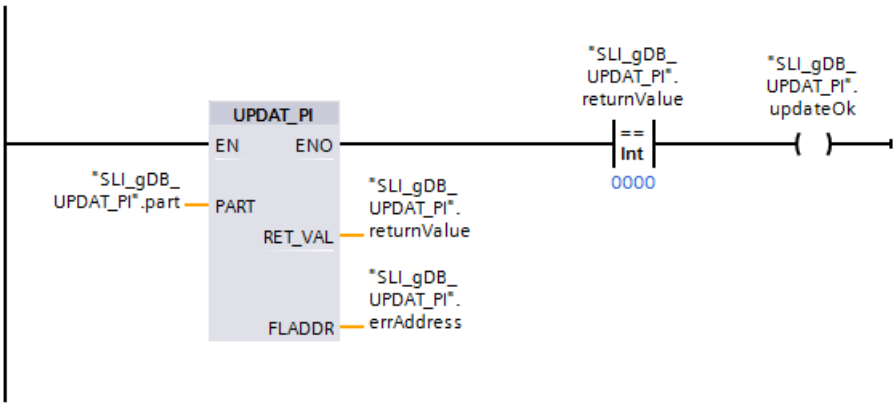

You create an input module (DI) for your CPU, and configure the process image partition "1" under "Properties > Input 0 - 15 > I/O addresses". You also do not assign the process image partition to any OB.

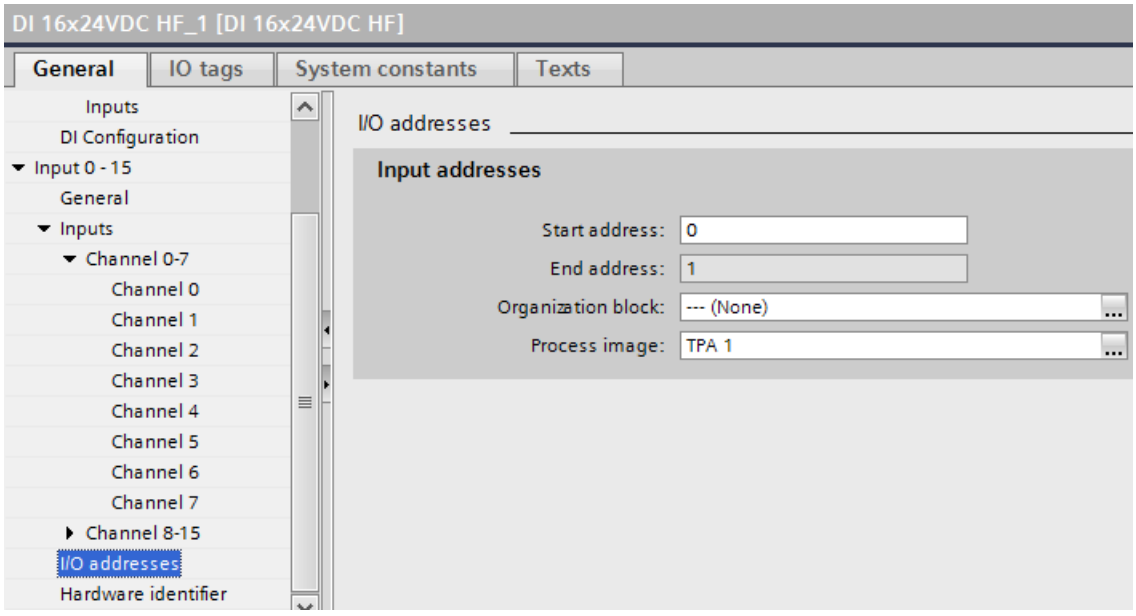

The number of the process image partition that is to be updated is stored in the input parameter PART ("part") of the instruction "UPDAT_PI". The specified process image partition is updated when the instruction "UPDAT_PI" is called.

The output parameter RET_VAL ("returnValue") indicates that processing took place without errors. Processing without errors sets the coil ("updateOk") to "TRUE". Because no access

error occurs, the error address ("errAddress") of the instruction "UPDAT_PI" shows the value "16#0000".

SLI_gDB_UPDAT_PI

		Name	Data type	Start value	Monitor value
1		Static			
2		part	PIP	1	1
3		returnValue	Int	0	0
4		errAddress	Word	16#0	16#0000
5		updateOk	Bool	false	TRUE

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

UPDAT_PO: Update the process image outputs (S7-1500)

Description

You use the instruction to transfer the signal states of a process image partition of the outputs defined by configuration to the output modules.

If you have specified a consistency range for the selected process image partition, corresponding data is transferred as consistent data to the respective I/O module.

Note

Each logical address you have assigned to a process image partition of the outputs during configuration no longer belongs to the OB 1 process image of the outputs.

Outputs that you update with "UPDAT_PO" must not be updated simultaneously with the "SYNC_PO (Page 2205)" instruction.

The OB 1 process image of the outputs and the process image partitions of the outputs that you have assigned to an interrupt OB are transferred by the system to the output modules independently of "UPDAT_PO" calls.

Parameter

The following table shows the parameters of the "UPDAT_PO" instruction:

Parameter	Declaration	Data type	Memory area	Description
PART	Input	PIP	I, Q, M, D, L or constant	Number of the process image partition of the outputs to be transferred. Maximum value range (depending on the CPU): 1 <= n <= 31 (means process image partition n)
RET_VAL	Return	INT	I, Q, M, D, L, P	Error information
FLADDR	Output	WORD	I, Q, M, D, L, P	Address of the first byte to cause an error if an access error has occurred.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for PART parameter
8091	The specified process image partition was not yet defined.
8092	The process image partition is being updated by the system and must not be used for the PART parameter.
8093	The process image partition update is being processed in another OB.
80A0	An access error was detected when accessing the I/O.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Note

If you use the instruction for process image partitions of DP standard slaves, the error codes of the "DPWR_DAT (Page 2340)" instruction are also possible. This is the case if you have defined consistency areas greater than 32 bytes.

Example

In the following example, you update the process image partition "1" of the outputs defined in the configuration.

Create four tags in a global data block for storing the data.

SLI_gDB_UPDAT_PO

		Name	Data type	Start value
1		Static		
2		part	PIP	1
3		returnValue	Int	0
4		errAddress	Word	16#0
5		updateOk	Bool	false

Interconnect the parameters of the instruction as follows.

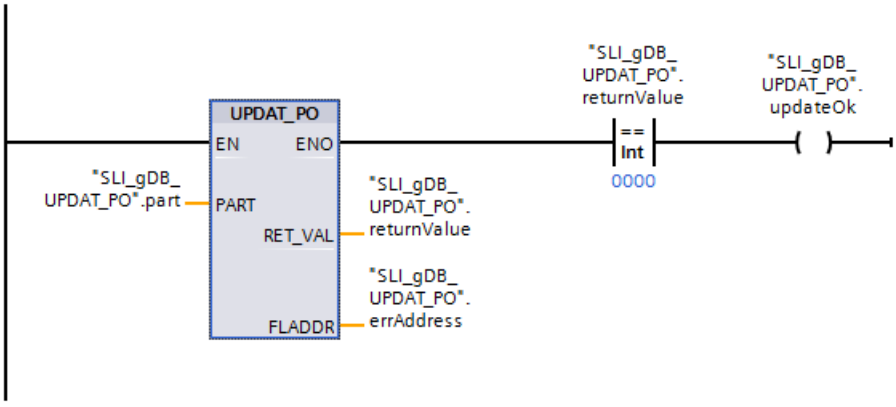

You create an output module (DQ) for your CPU, and configure the process image partition "1" under "Properties > Output 0 - 15 > I/O addresses". You also do not assign the process image partition to any OB.

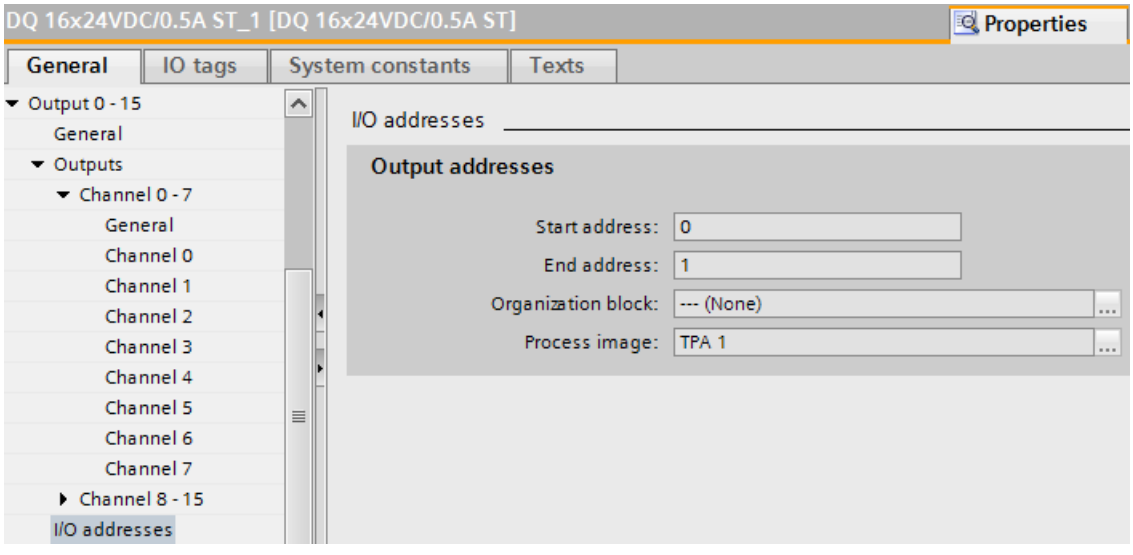

The number of the process image partition that is to be updated is stored in the input parameter PART ("part") of the instruction "UPDAT_PO". The specified process image partition is updated when the instruction "UPDAT_PO" is called.

The output parameter RET_VAL ("returnValue") indicates that processing took place without errors. Processing without errors sets the coil ("updateOk") to "TRUE". Because no access error occurs, the error address ("errAddress") of the instruction "UPDAT_PO" shows the value "16#0000".

SLI_gDB_UPDAT_PO

	Name	Data type	Start value	Monitor value
1	Static			
2	part	PIP	1	1
3	returnValue	Int	0	0
4	errAddress	Word	16#0	16#0000
5	updateOk	Bool	false	TRUE

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

SYNC_PI: Synchronize the process image inputs (S7-1500)

Description

The "SYNC_PI" is used to update the process image partition of inputs in isochronous mode. A user program linked to a DP cycle or PN send cycle uses this instruction to update input data acquired in a process image partition of the inputs isochronously and consistently.

Call

"SYNC_PI" is interruptible and is only be called in OBs 61, 62, 63 and 64.

Note

The call of the "SYNC_PI" instruction is only permitted in OBs 61 to 64 if you have assigned the affected process image partition to the associated OB in the HW configuration. A process image partition that you update with "SYNC_PI" must not be updated simultaneously with the "UPDAT_PI (Page 2197)" instruction.

Parameter

The following table shows the parameters of the "SYNC_PI" instruction:

Parameter	Declaration	Data type	Memory area	Value range	Description
PART	Input	PIP	I, Q, M, D, L or constant	1 to 31	Number of the process image partition of the inputs to be updated isochronously.
RET_VAL	Return	INT	I, Q, M, D, L, P	-	Error information
FLADDR	Output	WORD	I, Q, M, D, L, P		Address of the first byte to cause an error if an access error has occurred.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)*	Explanation
0000	No error occurred.
0001	Consistency warning. The update of the process image partition was distributed over two DP or PN cycles. However, the data in one submodule were consistently transferred.
8090	Illegal value at PART parameter or updating the specified process image partition of the inputs is not permitted in this OB. The specified process image partition was possibly not yet defined. The process image partition of the inputs was not updated.
8092	The process image partition is being updated by the system and must not be used for the PART parameter.
8093	The process image partition update is being processed in another OB.
80A0	During updating an access error was detected. The affected inputs were set to "0".
80A1	The update time is after the permitted access window. The process image partition of the inputs was not updated. The DP or PN cycle is too short to ensure enough time for processing the instruction. You must therefore increase the TDP (also known as T_DC), Ti and To timers.
80A2	Access error with consistency warning In the case of update of the specified process image partition an access error with simultaneous consistency warning was detected. • The data of the incorrect inputs was not read by the I/O. In the process image partition of the inputs, the affected inputs are set to "0". • The update of the process image partition of the not involved by an access error input data was distributed over two DP and PN cycles.
80C1	The update time is before the permitted access window. The process image partition of the inputs was not updated.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Note

If you use the "SYNC_PI" instruction for process image partitions of DP standard slaves, error codes of the "DPRD_DAT" instruction are also possible. This is the case if you have defined consistency areas greater than 32 bytes.

See also

Switching display formats in the program status (Page 7698)

DPRD_DAT: Read consistent data of a DP standard slave (Page 2338)

SYNC_PO: Synchronize the process image outputs (S7-1500)

Description

The instruction "SYNC_PO" is used to update a process image partition of outputs in isochronous mode. An application program linked to a DP cycle or a PN send cycle can use this instruction for the consistent isochronous transfer of the computed output data of a process image partition from outputs to the I/O devices.

Call

"SYNC_PO" is interruptible and should only be called in OBs 61, 62, 63 and 64.

Note

The call of the "SYNC_PO" instruction in OBs 61 to 64 is only permitted if you have assigned the affected process image partition to the associated OB in the HW configuration. A process image partition that you update with "SYNC_PO" must not be updated simultaneously with the "UPDAT_PO (Page 2200)" instruction.

Parameter

The following table shows the parameters of the "SYNC_PO" instruction:

Parameter	Declaration	Data type	Memory area	Value range	Description
PART	Input	PIP	I, Q, M, D, L or constant	1 to 31	Number of the process image partition of the outputs to be updated isochronously.
RET_VAL	Return	INT	I, Q, M, D, P	-	If an error occurs while the instruction is being executed, the return value contains an error code.
FLADDR	Output	WORD	I, Q, M, D, L, P	-	Address of the first byte that has caused the error.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)*	Explanation
0000	No error occurred.
0001	Consistency warning. The update of the process image partition was distributed over two DP or PN cycles. However, the data in one submodule were consistently transferred.
8090	Illegal value at PART parameter or updating of the specified process image partition of the outputs is not permitted in this OB. The specified process image partition was possibly not yet defined. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
8092	The process image partition is being updated by the system and is not used for the PART parameter.
8093	The process image partition update is being processed in another OB.

Error code (W#16#...)*	Explanation
80A0	In the case of update of the specified process image partition of the outputs an access error was detected. Incorrect outputs were not transferred to the I/O. During process image partition of the outputs, these outputs remain unchanged.
80A1	Access error with consistency warning In the case of update of the specified process image partition of the outputs an access error with simultaneous consistency warning was detected. • The data of the incorrect outputs were not transferred to the I/O. During process image partition of the outputs, the involved outputs remain unchanged. • The update of the process image partition of the not involved by an access error input data was distributed over two DP and PN cycles.
80A2	The update time is after the permitted access window. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
80C1	The update time is before the permitted access window. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Note

If you use the "SYNC_PO" instruction for process image partitions of DP standard slaves, error codes of the "DPWR_DAT" instruction are also possible. This is the case if you have defined consistency areas greater than 32 bytes.

See also

DPWR_DAT: Write consistent data of a DP standard slave (Page 2340)

Switching display formats in the program status (Page 7698)

4.1.4.4 Distributed I/O

Overview of the types of data records

Introduction

The data record as property of the hardware corresponds to the summary of data that reflects certain properties of the hardware. Each hardware can have numerous data records as properties. The data records can hereby be distinguished based on their use and their scope of information.

Below you can familiarize yourself with the types of data records and you will be guided through working with data records by assigning modules, systems and instructions.

The following information applies to the S7-1200 and S7-1500 automation systems as well as the ET 200MP and ET 200SP distributed I/O systems.

Types of data records

The following types of data records exist:

Types		Uses
Parameter data records	a) for input modules	Parameter data records contain the properties of the input/output function of the relevant I/O module. With the data record, you, for example, read out a wire break or change the input delay. Parameter data records can be read and written.
	b) for technology modules or modules with the use of the technology function	
	a) for output modules	
Control data records	a) to configure a central or distributed setup	The control data record contains the hardware configuration of the system or setup and allows reconfiguration of a module or an IO system via the user program. The control data record can be read and written.
	b) to reconfigure an IO system	
I&M data records	a) for central setup or distributed setup via PROFINET IO	The I&M data records contain the identification and maintenance data of a module (I&M = Identification & Maintenance). The data differs as follows: • Identification data: – depends on vendor data – read-only – You can, for example, read out the serial number. • Maintenance data – depends on the system and the use of the device – can be read and written. – can be used to change the installation location of the module, for example.
	b) for central setup or distributed setup via PROFIBUS DP	
Diagnostics data records and configuration data records	a) for distributed setup via PROFINET IO	Diagnostics data records and configuration data records each have the same structure and vary only in content and size. The data records can be distinguished as follows: • Diagnostics data records: – The diagnostics data records contain the diagnostics and monitoring data of a module. – Channel diagnostics data records are generated if a channel is in an error state and / or has triggered an interrupt. A diagnostics data record of length 0 is returned if there is no fault/error. – The structure and size of vendor-specific diagnostics data records depend on the vendor. – Diagnostics data records are read-only. • Configuration data records: – The configuration data records contain the data of the actual configuration or preset configuration of a module. – Configuration data records are readable (actual configuration, preset configuration) and writable (preset configuration).
	b) for distributed setup via PROFIBUS DP	

Note

The recipe data records used in WinCC are not properties of the module and have nothing to do with the data records being discussed here.

Use of a data record in the proect

Before you use a data record in you project, please read the relevant manual for the data record structure and any requirements. The "Additional information" section references the respective manual.

Parameter data records

Note

Reading out channels

The data records 0..X and 64..64+X contain channel-specific parameters. Accordingly for example the data record 0 contains the information of channel 0.

The data record 128 contains the information of all channels of the relevant module from the hardware configuration.

The RD_DPAR instruction can only be executed on an S7-1500 CPU or an ET 200 distributed I/O.

Type:	Parameter data records (a)				
Relates to the module:	Input module				
Location:	Central setup (S7-1500) or ET 200MP		S7-1200 or ET 200SP (*)		
Instructions:	RDREC, WRREC	RD_DPAR	RDREC, WRREC		RD_DPAR (not for S7-1200)
Data record (DR):	DR 0.X (DR = channel; X = the last channel of the input module)	DR 128	DR 128	*For ET 200SP Namur: DR 0.X (DR = channel; X = the last channel of the input module) or DR 128	DR 128
HW ID:	Use HW ID of the input module				
Example:	For example, S7-1500 DI16x24VDC HF --> channels 0..15; (e.g., use DR 13). See also "Use the program example of the parameter data record with WRREC & RDREC (Page 2234)".	For example, S7-1500 DI16x24VDC HF --> DR 128. See also "Program example of RD_DPAR (Page 2421)".	1st example: Use input module of an S7-1212C CPU --> HW ID of CPU, DR 128 and RDREC / WRREC. 2nd example: Use ET 200SP DI8x24VDC HF --> HW ID of DI module, DR 128 and RDREC / WRREC or RD_DPAR. See also "Program example of RD_DPAR (Page 2421)".		

Type:	Parameter data records (b)		
Relates to the module:	Technology modules or module with the use of the technology function		
Location:	Central setup (S7-1500 or S7-1200) or distributed setup (ET 200MP or ET 200SP)		
Instructions:	RDREC, WRREC		RD_DPAR (not for S7-1200)
Data record (DR):	DR 128	Alternatively, for DI modules of S7-1500: DR 0.X (DR = channel; X = the last channel of the input module)	DR 128
HW ID:	Use HW ID of the affected module / I/O module.		
Example:	1) Example: CPU S7-1511 PN; high speed counter of the S7-1500 DI16x24VDC HF --> for example, DR 1; use HW ID of the DI module (HSC). Use WRREC/RDREC. 2) Example: TM Count 2x24V --> DR 128; use HW ID of the TM. Use WRREC / RDREC or RD_DPAR. 3) Example: High speed counter of an S7-1200 CPU --> DR 128; use HW ID of CPU and WRREC / RDREC.		

Type:	Parameter data records (c)				
Relates to the module:	Output module				
Location:	Central setup (S7-1500) or ET 200MP		S7-1200 or ET 200SP (*)		
Instructions:	RDREC, WRREC	RD_DPAR	RDREC, WRREC		RD_DPAR (not for S7-1200)
Data record (DR):	DR 64.64 + X (DR 64 = channel 0; X = the last channel of the output module)	DR 128	DR 128	*For ET 200SP Namur: DS 64..64 + X (DR 64 = channel 0; X = the last channel of the output module) or DR 128	DR 128
HW ID:	Use HW ID of the output module.				
Example:	For example, DQ16x24VDC/0.5A ST --> channels 0..15; (e.g. for channel 13 use the DR 77).	For example, DQ16x24VDC/0.5A ST --> DR 128.	For example, ET 200SP DQ8x24VDC/0.5A ST --> DR 128.		

Note

Further data records

There may be more data records. E.g. for the input modules:

- With the ET 200SP AI RTD there is the DR 237 for deactivating the wire break check.
- With the ET 200SP AI Energy Meter ST there is the DR 142 for reading the measured data.

Note

Interface module with PROFIBUS DP

If you use an interface module with PROFIBUS DP, you cannot read out parameter data records "0" and "1" from the downstream input/output module. Instead read out the content of the diagnostics data records "0" and "1".

For example with output modules the "parameter data record 0" is the "DR 64" and the "parameter data record 1"is the "DR 65".

Additional information

You can find additional information on the parameter data record in the manual for the relevant I/O module (e.g. S7-1500 DI16x24VDC HF (https://support.industry.siemens.com/cs/ww/en/view/59193001)).

Control data records

Type:	Control data records			
Relates to the module:	All modules of the setup that are to be used			IO system and required IO devices
Location:	Central setup (S7-1200, S7-1500)	Distributed setup (e.g. ET 200MP, ET 200SP)		PROFINET IO
Instructions:	WRREC, RDREC	WRREC, RDREC	RDREC	ReconfigIOSystem
Data record (DR):	DR 196	DR 196	DR 197 (feedback data record)	DR "CTRLREC" (parameter of ReconfigIOSystem)
HW ID:	Use HW ID 33.	Use HW ID of the head [Head] of the IM of the distributed I/O.		Use HW ID of the PROFINET interface (IO controller). Specify the corresponding HW IDs via CTRLREC according to the connections and IO devices that are to be set up.
Example:	For example, for CPU S7-1511-1 PN with DI16x24VDC HF. See also "Use the program example of the control data record with WRREC & RDREC (Page 2244)".	For example, DI16x24VDC HF, ET 200MP IM 155-5 PN ST --> use the ID of "<IO-device_V2.0[Head]>". The real setup contains an additional DI16x24VDC HF, which is not to be considered during the configuration.		For example, connect and activate an S7-1500 CPU (IO controller) with an ET 200MP (IO device) and an ET 200SP (as optional IO device). See also "Program example for ReconfigIOSystem (Page 2320)".

Additional information

For more information on the control data record 196, refer to the system manual of the relevant automation system (e.g. "S7-1500 automation system (https://support.industry.siemens.com/cs/ww/en/view/59191792)").

For more information on the control data record 197, refer to the system manual of the relevant interface module (e.g. "ET 200MP IM 155-5 PN ST (https://support.industry.siemens.com/cs/ww/en/view/59193106)").

You can find more information on the control data record "CTRLREC" with the instruction "ReconfigIOSystem: Reconfigure IO system (Page 2315)".

I&M data records

Note the following use of the I&M data records:

- The data record "I&M0" ("16#AFF0") is used to transfer general information about the module or device.
- The data record "I&M1" ("16#AFF1") is used to transfer the function and installation location of the module or device.
- The data record "I&M2" ("16#AFF2") is used to transfer the installation date of the module or device.
- The data record "I&M3" ("16#AFF3") is used to transfer individual additional information about the module or device.

Note

The data record "I&M0" ("16#AFF0") is supported by every module. All other I&M data records are optional.

Type:	I&M data records				
Relates to the module:	Device / module				
Location:	Central setup or distributed setup via PROFINET IO or PROFIBUS DP				
Instructions:	RDREC, WRREC				
Data record (DR):	DR 16#AFF0	DR 16#AFF1	DR 16#AFF2	DR 16#AFF3	Alternatively, for distributed setup via PROFIBUS DP: DR 255
HW ID:	Use HW ID of the module.				
Example:	1) Example: For CPU S7-1511-1 PN --> DR 16#AFF0; use the HW ID of the CPU. See also "Program example of I&M data record with read RDREC (Page 2216)".				2) Example: For ET 200SP with IM 155-6 DP ST --> DR 255; use the HW ID of the IM.

Additional information

You can find additional information on the relevant I&M data record:

- in the system manual of the relevant automation system (e.g. "S7-1500 automation system (https://support.industry.siemens.com/cs/ww/en/view/59191792)").
- In the programming manual "SIMATIC NET PROFINET IO-Base user programming interface (https://support.industry.siemens.com/cs/ww/en/view/26435491)".

Diagnostics data records and configuration data records

Type:	Diagnostics data records and configuration data records			
Relates to the module:	Device/module or submodule in the IO system			
Location:	Distributed setup (PROFINET IO, PROFIBUS DP)			
Instructions:	RDREC, WRREC	RDREC		
Data record (DR):	Configuration (actual/preset):	Channel diagnostics:	Manufacturer-specific diagnostics via interface module:	
	Use DR for the relevant module. See also the addressing level and DR for "PROFINET IO (https://support.industry.siemens.com/cs/ww/en/view/19289930)".		For PROFINET IO: See Interface module	For PROFIBUS DP: DR 0, 1
HW ID:	Use hardware ID of the relevant module (HW ID with "Hw_SubModule" data type).		Use hardware ID of the relevant module (HW ID with "Hw_SubModule" data type).	
Example:	For example, IO controller S7-1513-1 PN; use HW ID of the module on the slot level for the IO device (ET 200MP IM 155-5 PN HF). Read out the preset configuration with DR 16#C001.	For example, IO device (ET 200MP IM 155-5 PN HF) with DQ module causes a wire break. Use the hardware ID of the header module of the IO device. IO controller is S7-1513-1 PN. Use DR 16#E00A for channel diagnostics. See also "Read program example for diagnostic data record with RDREC (Page 2220)".	For example, PROFIBUS DP: ET 200MP with IM 155-5 DP ST; Read DI module --> HW ID ("Hw_SubModule" data type) of the DI module and use DR 1 for diagnostics.	

Additional information

The addressing level, the setup and the important data records for diagnostics and configuration of PROFINET IO are available at:

- in the programming manual "From PROFIBUS DP to PROFINET IO (https://support.industry.siemens.com/cs/ww/en/view/19289930)".

You can find additional and general information on channel diagnostics data records:

- in the online help of STEP 7 in "Diagnostics of PROFINET devices".
- in the system manual for PROFINET (for example"PROFINET with STEP 7 as of V14 (https://support.industry.siemens.com/cs/ww/en/view/49948856)").

You can find more information on manufacturer-specific diagnostics data records:

- in the manual for the relevant interface module (for example, ET 200MP IM 155-5 DP ST (https://support.industry.siemens.com/cs/ww/en/view/78324181)).

See also

Sample Library for Instructions (Page 593)

RDREC: Read data record (Page 2213)

WRREC: Write data record (Page 2232)

FAQ page of the Library of PLC Datatypes (https://support.industry.siemens.com/cs/ww/en/view/109482396)

RDREC: Read data record

RDREC: Read data record (S7-1200, S7-1500)

Description

You use the instruction "RDREC" to read the data record with the number INDEX from the module addressed using the ID. This may be a module in a central rack or a distributed module (PROFIBUS DP or PROFINET IO).

- Use the parameter ID to select from which module (DP/PROFINET IO) a data record is to be read. Use only the hardware identifier (HW ID) of the module for the ID parameter. You can find the hardware identifier of the module in "PLC tags > System constants". For example, for an S7-1500 DI module: This is shown as <Name of DI module> with the data type "Hw_SubModule".
- Use the INDEX parameter to select which data record is read from the module. Which data records can be read with which data record number depends on the module. You will find additional information about this in the documentation for the respective module.
- The data records readable via "RDREC" have a different length. Use MLEN to specify the maximum number of data record bytes you want to read. If length "0" is selected at the parameter MLEN, the complete data record is written at the parameter RECORD.
- Select the target range RECORD at least MLEN bytes long. If you read out the full data record with MLEN=0, use the maximum length of the data record for RECORD. The structure (configuration, data types and length) that you use at the parameter RECORD, also depends on which data record is read out by which module.
- The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the target range RECORD. In this case, the LEN output parameter contains the length of the read data in bytes.
- If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR. In this case, the output parameter STATUS contains the error information.

Note

The interface of the "RDREC" instruction is identical to that in the "RDREC" FB defined in the standard "PROFIBUS and PROFINET Guideline Communication Function Blocks on PROFIBUS DP and PROFINET IO".

Functional description

"RDREC" is an instruction that works asynchronously, which means its execution extends over multiple calls. You start the data record transfer by calling "RDREC" with REQ = 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS. The two central bytes of STATUS correspond to the RET_VAL output parameter of the instructions that operate asynchronously.

See also: Difference between synchronous and asynchronous instructions (Page 596)

The transfer of the data record is complete when the output parameter BUSY has the value FALSE .

Calling RDREC in the SYNCUP system state of an S7-1500-R/H system

When you call the RDREC instruction in the SYNCUP system state of an S7-1500-R/H system, the error codes 0x7001 and 0x7002 are always returned. Even if an error occurs, for example, no other error code is provided when the data record source does not exist. In this case, the correct error code is only supplied after the SYNCUP has finished.

If you call the RDREC instruction in the SYNCUP system state of an S7-1500-R/H system, no data record request is triggered. The data record request is not triggered until you call the instruction again after the SYNCUP has ended. It is therefore recommended to always call RDREC cyclically.

Parameter

The following table shows the parameters of the "RDREC" instruction:

Parameter	Declaration	Data type*	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T**, C** or constant	REQ = 1: Transfer data record
ID	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the hardware module (DP/PROFINET IO) The number is assigned automatically and is stored in the properties of the module or of the interface in the hardware configuration.
INDEX	Input	DINT	I, Q, M, D, L or constant	Data record number
MLEN	Input	UINT	I, Q, M, D, L or constant	Maximum length in bytes of the data record information to be read
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information
LEN	Output	UINT	I, Q, M, D, L	Length of the read data record information

Parameter	Declaration	Data type*	Memory area	Description
RECORD	InOut	VARIANT	I, Q, M, D, L	Target range for the data record read. If you are using the NREF system data type for the target range in a data block with optimized access, no values are written to the target range.
* There is no implicit conversion in STL, which is why the range of valid data types may be limited. During programming in STL, note the permissible data types in each case in the tooltip of the parameter ** For S7-1500 only.				

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

If you use "RDREC" to read a data record for PROFINET IO, then negative values in the INDEX, MLEN, and LEN parameters will be interpreted as an unsigned 16-bit integer.

Parameter STATUS

For interpretation of the STATUS parameter, see Parameter STATUS (Page 2276).

Special case for PC systems: Determining Windows status

The following scenario will help you determine the Windows status of your PC system:

- Call the instruction "RDREC" (SFB52) with a cyclical OB (e.g. OB1).

To determine the Windows status with the "RDREC" instruction, a data record index (0x7FFE) is defined for the software controller.

The "RDREC" instruction is called with the following input parameters:

Parameter	Declaration	Data type	Memory area	Value/Description
(HIC) ID	Input	HW_IO	I, Q, M, D, L or constant	59 (Host Interface RID)
INDEX	Input	DINT	I, Q, M, D, L or constant	0x7FFE
MLEN	Input	UINT	I, Q, M, D, L or constant	9
RECORD	InOut	VARIANT	I, Q, M, D, L	Address for each area of 9 bytes

The data record for the Windows status is output by the eighth byte and can have the following content:

- 0x01: Windows running
- 0x02: Windows not running
- 0x03: Bluescreen

Note

In case of a system crash, Windows no longer responds and shows a blue screen which indicates the reason for and the time of the system crash. By default, the blue screen remains visible permanently.

The system is stopped after the system error and you can encounter the following scenarios:

- Windows no longer sends signals to the configured software CPU
- No call of OB82
- A diagnostic entry is not created

Examples

You can find an example here: Program example of I&M data record with read RDREC (Page 2216).

You can find another example here: Read program example for diagnostic data record with RDREC (Page 2220).

You can find additional information and the program code for the examples here: Sample Library for Instructions (Page 593).

See also

Basics of block access (Page 55)

Overview of the types of data records (Page 2206)

Program example of I&M data record with read RDREC

Introduction

In the following example, you read out the I&M0 data record of a CPU S7-1500.

For more information on data records, refer to the Overview of the types of data records (Page 2206).

Requirement

Create nine tags and a structure in a global data block for storing the data.

You read the hardware identifier of the CPU in “PLC tags > System constants. The CPU should be displayed there as <Local> with the data type “Hw_SubModule“. You store the hardware identifier via the input parameter ID ("id").

SLI_gDB_RDREC_I-M			
	Name	Data type	Start value
1	Static		
2	start	Bool	false
3	id	HW_IO	49
4	dataRecNbr	DInt	16#AFF0
5	maxLen	UInt	0
6	checkedValid	Bool	false
7	busy	Bool	false
8	error	Bool	false
9	status	DWord	16#0
10	len	UInt	0
11	recordIM0	Struct	
12	BlockType	Word	16#0
13	BlockLength	UInt	0
14	BlockVersion_HL	Word	16#0
15	VendorID_HL	Word	16#0
16	OrderID	Array[0..19] of Byte	
17	IM_SerialNbr	Array[0..15] of Byte	
18	IM_HWrev	UInt	0
19	IM_SWrev	Struct	
20	IM_RevCounter	UInt	0
21	IM_ProfileID	UInt	0
22	IM_ProfileSpecType	Word	16#0
23	IM_Version	Word	16#0
24	IM_Supported	Word	16#0

Create the following local tags in an FB.

memErrStatus	DWord	16#0	Set in IDB

Interconnect parameters

Network 1: Interconnect the parameters of the "RDREC" instruction as follows.

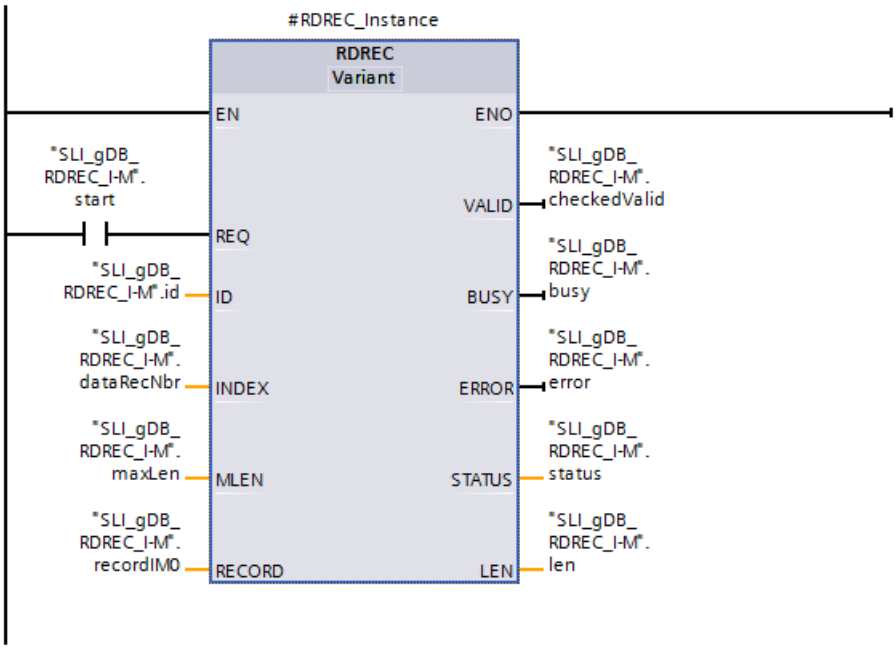

Network 2: To save the status if an error occurs, interconnect the “status“ variable as follows

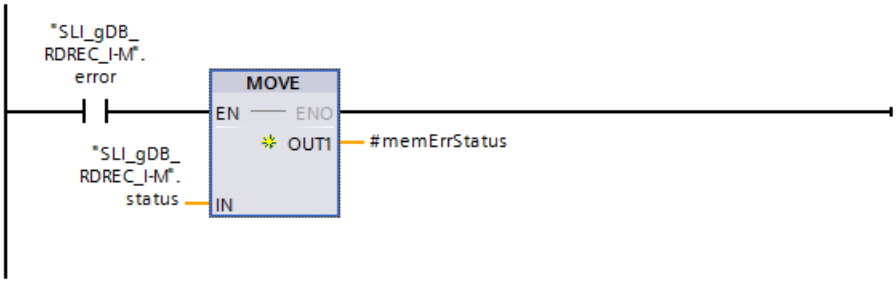

Result

When the input parameter REQ ("start") returns the signal state "TRUE", the "RDREC" instruction is started. Via the input parameter ID ("id") the instruction "RDREC" calls the module. With the input parameter INDEX ("dataRecNbr") the I&M0 data record of the CPU is called.

Over several calls the "RDREC" instruction reads out the I&M0 data record and saves the data record using the parameter RECORD ("recordIM0").

Among other things the I&M0 data record contains the following information.

- The vendor name ("VendorID_HL")
- The article (“OrderID“) & device number (“IM_SerialNbr“)

- Hardware ("IM_HWrev") & software information ("IMSWrev")
- IM version ("IM_Version") and the bandwidth supported I&Mdata records ("IM_Supported")

recordIM0	Struct		
BlockType	Word	16#0	16#0020
BlockLength	UInt	0	56
BlockVersion_HL	Word	16#0	16#0100
VendorID_HL	Word	16#0	16#002A
OrderID	Array[0..19] of Byte		
IM_SerialNbr	Array[0..15] of Byte		
IM_HWrev	UInt	0	3
IM_SWrev	Struct		
Refix	Byte	16#0	16#56
FunctEnh	Byte	16#0	16#01
BugFix	Byte	16#0	16#05
InternalChange	Byte	16#0	16#00
IM_RevCounter	UInt	0	0
IM_ProfileID	UInt	0	0
IM_ProfileSpecType	Word	16#0	16#0000
IM_Version	Word	16#0	16#0101
IM_Supported	Word	16#0	16#001E

The length of the read data record is stored using the output parameter LEN ("len"). According to the value of the input parameter MLEN ("maxLen"), has the value “0“ the data record information that is to be read is unlimited.

During processing the output parameter BUSY ("busy") is set to "TRUE" and VALID ("checkedValid") to "FALSE". A successful reading result ("checkedValid" to "TRUE") is not output until processing is complete. The block status is displayed at the output parameter STATUS ("status"). Processing takes place without errors.

SLI_gDB_RDREC_I-M

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	id	HW_IO	49	16#0031
4	dataRecNbr	DInt	16#AFF0	45040
5	maxLen	UInt	0	0
6	checkedValid	Bool	false	TRUE
7	busy	Bool	false	FALSE
8	error	Bool	false	FALSE
9	status	DWord	16#0	16#0000_0000
10	len	UInt	0	60
11	recordIM0	Struct		

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

RDREC: Read data record (Page 2213)

Read program example for diagnostic data record with RDREC

Introduction

In the following example, you use RDREC to read the diagnostic data record "16#E00A" of an IO device. When a wire break is triggered, the error information is output as a diagnostic data record.

For more information on data records, refer to the Overview of the types of data records (Page 2206).

Note

Number of diagnostic data records

Each slot of a module triggering an error generates its own data record for the IO device. Diagnostics can thus contain multiple records.

The example is intended to read multiple hardware errors (channel errors) of multiple modules and copy them to the appropriate structure of a memory array. The memory arrays ("recordA", "recordB") can hold a maximum of six data records with a maximum of 16 channels each based on a specified setting.

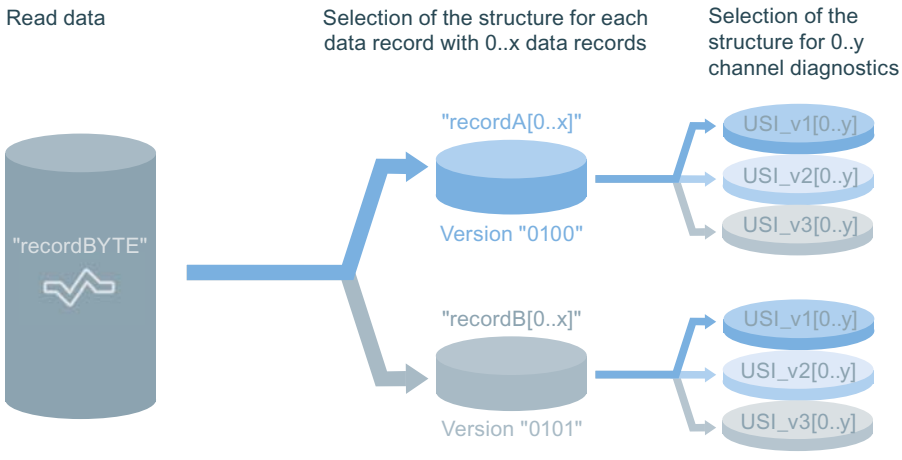

Requirements

Hardware installed

The following hardware is required:

- An IO controller (e.g. S7-1513-1 PN)
- An IO device (e.g. ET 200MP IM 155-5 PN HF)
- One or more digital output modules of the HF type (for example, DQ 4x24VDC HF)

Note

Trigger wire break

Make sure that the channels of the DQ module that are to trigger the wire break are not wired.

Note

Procedure for configuration of the hardware

Configure the hardware as follows:

1. Connect the CPU and IO device over PROFINET and assign the CPU to the IO device as an IO controller.
2. Open the path "Properties > Output 0-x > Outputs > Channel 0-y" for the respective DQ module.
3. Select each channel to be configured and select the "Wire break" option in the ""Diagnostics" area.

PLC data types created

You need the following PLC data types for the transfer and storage of data records:

- For the body content of the "A"/"0100" structure:

Rd_0100body

	Name	Data type	Default value
1	BlockType	Word	16#0
2	BlockLength	UInt	0
3	BlockVersion_HL	Word	16#0
4	SlotNumber	UInt	0
5	SubSlotNumber	UInt	0
6	ChannelLocation	Word	16#0
7	ChannelProperties	Array[0..15] of Bool	
8	UserStructureIdentifier	Word	16#0

- For the body content of the "B"/"0101" structure:

Rd_0101body

	Name	Data type	Default value
1	BlockType	Word	16#0
2	BlockLength	UInt	0
3	BlockVersion_HL	Word	16#0
4	API	DWord	16#0
5	SlotNumber	UInt	0
6	SubSlotNumber	UInt	0
7	ChannelLocation	Word	16#0
8	ChannelProperties	Array[0..15] of Bool	
9	UserStructureIdentifier	Word	16#0

- For the structure of type 1 in accordance with the value from "UserStructureIdentifier": Type 1 of the structure is used for basic channel diagnostics.

USI_v1

	Name	Data type	Default value
1	ChannelNumber	UInt	0
2	ChannelProperties	Array[0..15] of Bool	
3	ChannelErrorType	Word	16#0

- For the structure of type 2 in accordance with the value from "UserStructureIdentifier": Type 2 of the structure is used for extended channel diagnostics.

USI_v2

	Name	Data type	Default value
1	ChannelNumber	UInt	0
2	ChannelProperties	Array[0..15] of Bool	
3	ChannelErrorType	Word	16#0
4	ExtChErrorType	Word	16#0
5	ExtChAddValue	DWord	16#0

- For the structure of type 3 in accordance with the value from "UserStructureIdentifier": Type 3 of the structure is used for manufacturer-specific channel diagnostics. The configuration of the structure is not defined in this example. For this example, a length of 28 bytes is assumed for type 3.

USI_v3

	Name	Data type	Default value
1	ChannelDiagData3	Array[0..27] of Byte	

- For the summary of the "A"/"0100" structure:

RECdiag_0100

	Name	Data type
1	Body0100	"Rd_0100body"
2	USI_DetailsType1	Array[0..15] of "USI_v1"
3	USI_DetailsType2	Array[0..15] of "USI_v2"
4	USI_DetailsType3	Array[0..15] of "USI_v3"

- For the summary of the "B"/"0101" structure:

RECdiag_0101

	Name	Data type
1	Body0101	"Rd_0101body"
2	USI_DetailsType1	Array[0..15] of "USI_v1"
3	USI_DetailsType2	Array[0..15] of "USI_v2"
4	USI_DetailsType3	Array[0..15] of "USI_v3"

Note

Size of the arrays

If you want to change the array sizes (for read memory, data record or channels), you need to do the following:

- For channels: Adapt the size of the arrays ("Array[0..y] of USI_v..") in the PLC data types ("RECdiag_0100", "RECdiag_0101"). Adapt the "maxChannelPos" value in the gDB.
- For data records: Adapt the size of the arrays ("recordA", "recordB") in the gDB. Adapt the "maxRecordPos" value in the gDB.
- For the read memory ("recordBYTE"): Adapt the size of the array ("recordBYTE") and the value of "maxPosition" in the gDB. Note that the diagnostics data record "16#E00A" can have a maximum total of 4176 bytes.

Watch table created

To trigger a wire break, use the "Force" function of a watch table.

If you force the value "16#FFFF" for the "%QW0" output, for example, all channels with configured wire break using the output trigger a hardware error.

Storage of data

Create the following tags and structures for storing data in a global data block (gDB):

SLI_gDB_RDREC_Diag			
	Name	Data type	Start value
1	Static		
2	id	HW_IO	262
3	dataRecNbr	DInt	16#E00A
4	startRead	Bool	false
5	maxLength	UInt	0
6	maxPosition	UInt	4175
7	recordBYTE	Array[0..4175] of Byte	
8	readLength	UInt	0
9	checkedValid	Bool	false
10	busy	Bool	false
11	error	Bool	false
12	memErrStatus	DWord	16#0
13	countCalls	UInt	0
14	maxChannelPos	UInt	15
15	maxRecordPos	UInt	5
16	recordA	Array[0..5] of "RECdiag_0100"	
17	recordB	Array[0..5] of "RECdiag_0101"	
18	data_sorting_done	Bool	false
19	outputOverflow	Struct	
20	Too_many_bytes	Bool	false
21	Too_many_records	Bool	false
22	Too_many_channels	Bool	false
23	different_structs_used	Bool	false
24	countRecs	UInt	0
25	chosenStruct	UInt	0
26	posForRec	UInt	0
27	posForChannel	UInt	0
28	resetOps	Struct	
29	reset	Bool	false

You read the hardware identifier of the head of the IO device and store the hardware identifier using the "id" tag. You can find the hardware identifier for the header module with "PLC tags > System constants" and with an entry such as <Head~IO_Device>. The data type used is "HW_SubModule".

FC "SLI_FC_start_RDiag": Interconnect parameters

To start the "RDREC" instruction, based on a hardware fault, create the FC "SLI_FC_start_RDiag".

Create the following interconnection in the FC.

You call the FC "SLI_FC_start_RDiag" in a diagnostic interrupt OB (OB82).

FC "SLI_FC_reset_RDiag": Interconnect parameters

To reset process values after an error, create the FC "SLI_FC_reset_RDiag".

Create the following local tag in the FC.

Temp	
tempReset	Bool

Create the following interconnections in the FC.

Network 1: To reset the process values, create the following interconnections. Part 1:

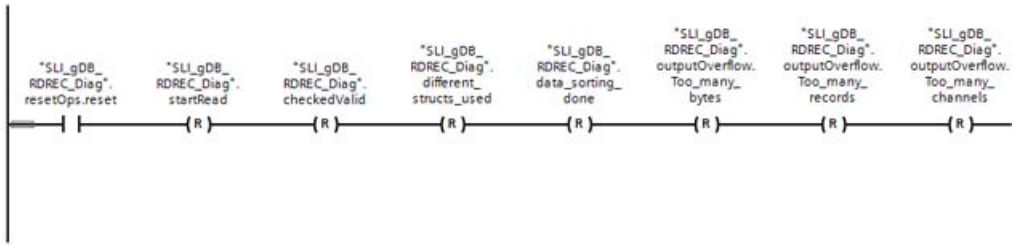

Part 2:

Part 3:

Part 4:

Network 2: To reset the "reset" tag, create the following interconnections.

FB "SLI_FB_chooseStruct_RDiag": Interconnect parameters

You create the FB "SLI_FB_chooseStruct_RDiag" to select the appropriate structure for the data record and to copy the data to the appropriate location in the array.

You can find additional information in the "Result" section ("Network 6") or in the comment field of the networks (see "Program code").

FB "SLI_FB_RDREC_Diag": Interconnect parameters

You create the FB "SLI_FC_start_RDiag" to call and process the "RDREC" instruction and all other subsequent processes.

Create the following local tags in the FB.

Static			
statusExe	DWord	16#0	Nicht re...
checkExe	Bool	false	Nicht rema...
lengthOut	UInt	0	Nicht rema...
RDREC_Instance00	RDREC		

Create the following interconnections in the FB:

Network 1: Call the FC "SLI_FC_reset_RDiag" and create the "Label" instruction with the name "RDREC".

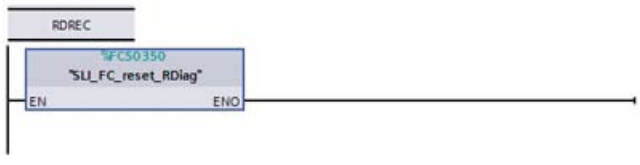

Network 2: Interconnect the parameters of the "RDREC" instruction as follows.

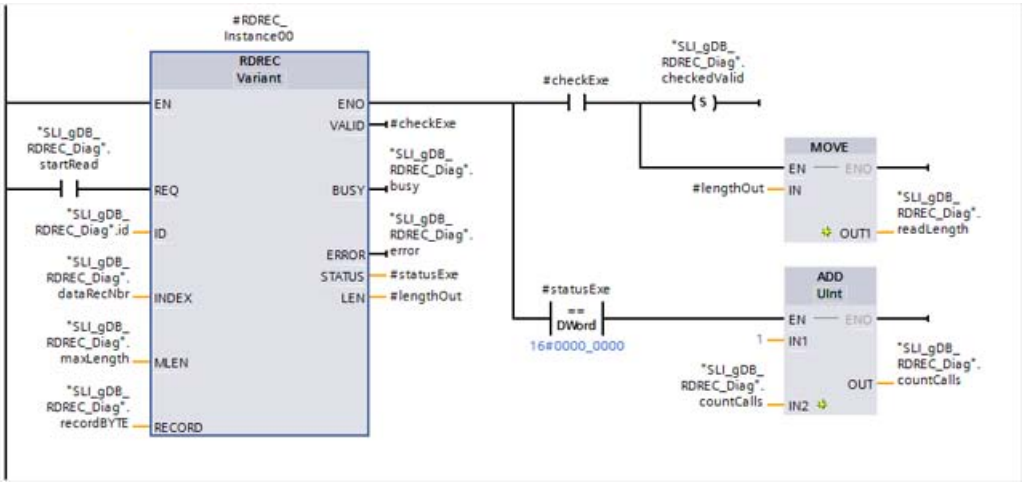

Network 3: To save the status if a RDREC error occurs, interconnect the parameter as follows

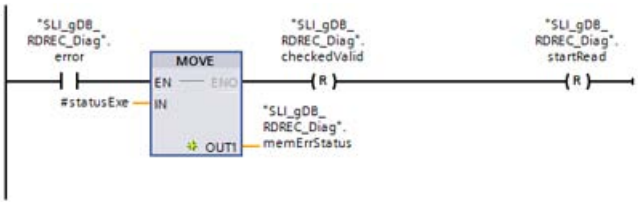

Network 4: To end the processing of RDREC, create the following interconnections.

Network 5: To jump back to network 1, create the following interconnections.

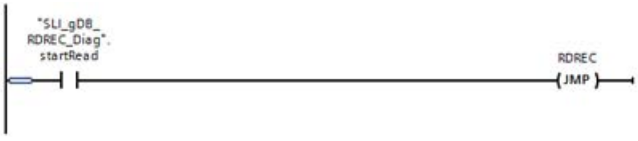

Network 6: Call the FB "SLI_FB_chooseStruct_RDiag" as follows.

You call the FB "SLI_FB_RDREC_Diag" in a cyclic OB (OB1).

Event FB "SLI_FB_RDREC_Diag"

Network 1:

If the NO contact ("reset") has the "TRUE" signal state, the process values of the FB "SLI_FB_chooseStruct_RDiag" and the process values of the global data block are reset.

Network 2:

When the input parameter REQ ("startRead") returns the signal state "TRUE", the "RDREC" instruction is started.

Note

Calling "startRead"

In accordance with the call by OB82, "startRead" is set to "TRUE" every time a hardware error is detected. The successful calls of RDREC can be counted using the "countCalls" tag. After three triggered channel errors, the "countCalls" value is "3".

The "RDREC" instruction calls the header module of the IO device using the input parameter ID ("id"). The input parameter INDEX ("dataRecNbr") is used to call the "16#E00A" diagnostics data record of the IO device.

The "RDREC" instruction reads the diagnostic data over several calls and saves the data read (x data records) with the RECORD parameter ("recordBYTE"). The length of the read data is recorded in the LEN output parameter ("#lengthOut") and saved for further use with the "readLength" tag. In accordance with the value of the input parameter MLEN ("maxLength" has the value "0"), the length of the data to be read is not restricted.

During processing the output parameter BUSY ("busy") is set to "TRUE" and VALID ("checkedValid") to "FALSE". A successful reading result ("checkedValid" set to "TRUE") is not output until processing is complete; it is used by the FB "SLI_FB_chooseStruct_RDiag" as a start condition. To end processing of the FB "SLI_FB_chooseStruct_RDiag", the "checkedValid" tag is set to "FALSE".

The block status is displayed at the output parameter STATUS ("#statusExe"). In the example, processing takes place without errors.

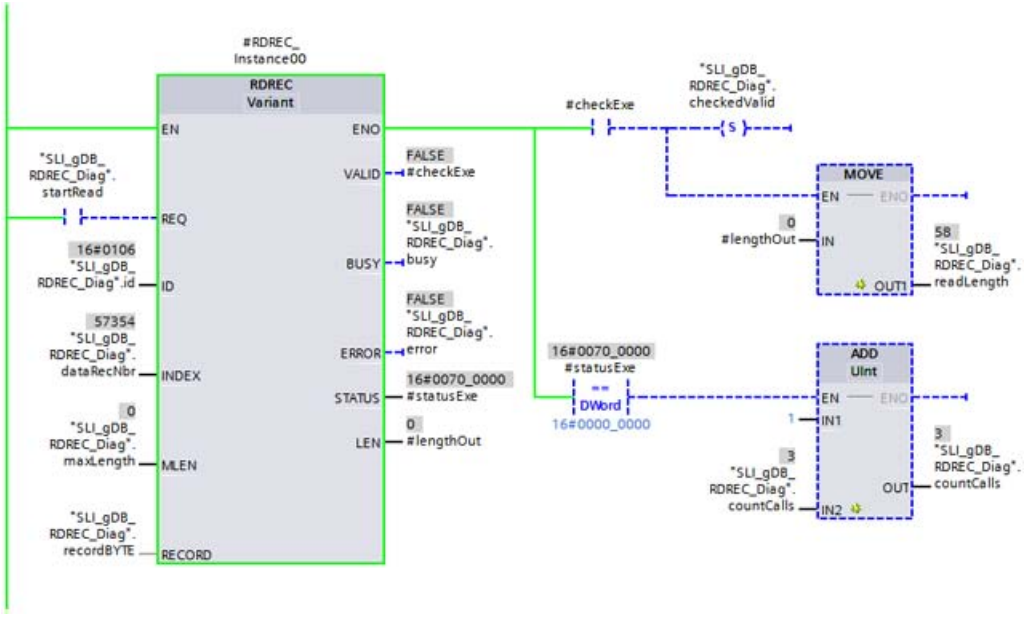

Network 3:

When the NO contact ("error") supplies the signal state "TRUE", the status ("#statusExe") is saved if an error occurs ("memErrStatus"). The "checkedValid" and "startRead" tags are also reset. The processing of RDREC stops. The processing of the FB "SLI_FB_chooseStruct_RDiag" cannot be started.

Network 4:

When RDREC has been fully processed ("#statusExe" has the value "16#0070_0200"), "startRead" is set to "FALSE". The processing of RDREC stops.

Network 5:

If the NO contact ("startRead") has the signal state "TRUE", processing jumps to network 1 (with the "JMP" instruction) in accordance with the "RDREC" label.

Network 6:

If the "checkedValid" tag has the signal state "TRUE" and the "reset" tag has the signal state "FALSE", the sorting process is started in the FB "SLI_FB_chooseStruct_RDiag".

In the FB "SLI_FB_chooseStruct_RDiag", the structure type of the respective data record and the structure type corresponding to the UserStructureIdentifier are selected. The data of the data record is copied to the correct location in the appropriate array. In addition, processing may be repeated several times according to the size of the respective data record. The information of one or more channels is read in this case.

If the "length of the data record + preceding lengths " ("#allRecLEN") is less than the length of all read data ("readLength") at the end of processing the data record content, processing of the FB starts again from the top. Another data record is read, its contents compared and its data is copied to the correct location of the appropriate array.

The sorting process is completed by setting the local tags ("#readComplete", "#dbComplete", "#limitReach") and the global tag "checkedValid" to "FALSE".

The sorting process is documented based in the following tags:

- "data_sorting_done": This tag indicates that processing of the FB "SLI_FB_chooseStruct_RDiag" is complete and has ended. The "checkedValid" tag is reset, which means processing of the FB cannot be started again.
- "outputOverflow": This structure contains three tags, each of which records a different violation of array limits. For example, more data records might have been read than the corresponding array ("recordA" or "recordB") is capable of storing.
- "different_structs_used": If both "recordA" and "recordB" are used to store the diagnostic data records, the "different_structs_used" tag returns the value "TRUE".
- "countRecs": This tag counts the copied data records.
- "chooseStruct": This tag indicates the structure used last. The value "4" indicates that a data record was last stored in the "recordB" structure with the "USI Type 1" form.
- "posForRec": This tag stores the most recently used location in the array of the data records ("recordA" or "recordB"). The value "1" indicates that a data record was last stored in "recordB[1]" (see "chooseStruct").
- "posForChannel": This tag stores the most recently used location in the array of the channels ("USI_DetailsType[x]"). The value "0" indicates that a channel was stored in "recordB[1]/USI_DetailsType[0]" (see "chooseStruct" and "posForRec").

SLI_gDB_RDREC_Diag

	Name	Data type	Start value	Monitor value
1	Static			
2	id	HW_IO	262	16#0106
3	dataRecNbr	DInt	16#E00A	57354
4	startRead	Bool	false	FALSE
5	maxLength	UInt	0	0
6	maxPosition	UInt	4175	4175
7	recordBYTE	Array[0..4175] of B...		
8	readLength	UInt	0	58
9	checkedValid	Bool	false	FALSE
10	busy	Bool	false	FALSE
11	error	Bool	false	FALSE
12	memErrStatus	DWord	16#0	16#0000_0000
13	countCalls	UInt	0	3
14	maxChannelPos	UInt	15	15
15	maxRecordPos	UInt	5	5
16	recordA	Array[0..5] of "RECd...		
17	recordB	Array[0..5] of "RECd...		
18	data_sorting_done	Bool	false	TRUE
19	outputOverflow	Struct		
20	different_structs_used	Bool	false	FALSE
21	countRecs	UInt	0	2
22	chosenStruct	UInt	0	4
23	posForRec	UInt	0	1
24	posForChannel	UInt	0	0
25	resetOps	Struct		
26	reset	Bool	false	FALSE

Below you can see the data record with the structure "B"/"0101" at position "0" ("recordB[0]") with two read channels.

The value "16#8000" from "ChannelLocation" shows diagnostics on the sub-module level. According to the value "16#0006" of "ChannelErrorType", there is a wire break at the channels 0 and 1 (see "ChannelNumber") of submodule 1 (see "SubSlotNumber"), of the module in slot 2 (see "SlotNumber").

17	recordB	Array[0..5] of "RECd...		
18	recordB[0]	"RECdiag_0101"		
19	Body0101	"Rd_0101body"		
20	BlockType	Word	16#0	16#0010
21	BlockLength	UInt	0	28
22	BlockVersion_HL	Word	16#0	16#0101
23	API	DWord	16#0	16#0000_0000
24	SlotNumber	UInt	0	2
25	SubSlotNumber	UInt	0	1
26	ChannelLocation	Word	16#0	16#8000
27	ChannelProperties	Array[0..15] of Bool		
28	UserStructureIdentifier	Word	16#0	16#8000
29	USI_DetailsType1	Array[0..15] of "USI...		
30	USI_DetailsType1[0]	"USI_v1"		
31	ChannelNumber	UInt	0	0
32	ChannelProperties	Array[0..15] of Bool		
33	ChannelErrorType	Word	16#0	16#0006
34	USI_DetailsType1[1]	"USI_v1"		
35	ChannelNumber	UInt	0	1
36	ChannelProperties	Array[0..15] of Bool		
37	ChannelProperties[0]	Bool	false	FALSE
38	ChannelProperties[1]	Bool	false	FALSE
39	ChannelProperties[2]	Bool	false	FALSE
40	ChannelProperties[3]	Bool	false	TRUE
41	ChannelProperties[4]	Bool	false	FALSE
42	ChannelProperties[5]	Bool	false	FALSE
43	ChannelProperties[6]	Bool	false	TRUE
44	ChannelProperties[7]	Bool	false	FALSE
45	ChannelProperties[8]	Bool	false	TRUE

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

RDREC: Read data record (Page 2213)

WRREC: Write data record

WRREC: Write data record (S7-1200, S7-1500)

Description

The instruction "WRREC" is used to transfer the RECORD data record to the component addressed using ID. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

- Use the ID parameter to select the module for which a data record is to be written. Use only the hardware identifier (HW ID) of the module for the ID parameter. You can find the hardware identifier of the module in "PLC tags > System constants". For example, for an S7-1500 DI module: This is shown as <Name of DI module> with the data type "Hw_SubModule".
- Use the INDEX parameter to select which data record is written. The module determines which data records can be written with which data record number. You can find additional information on this in the manual for the relevant module.
- Use LEN to specify the length of the data record to be transmitted in bytes. Select the source area RECORD at least LEN bytes long.
- The value TRUE at output parameter DONE indicates that the data record has been successfully transferred.
- If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR. In this case, the output parameter STATUS contains the error information.

Note

The interface of the "WRREC" instruction is identical to that in the "WRREC" FB defined in the standard "PROFIBUS and PROFINET Guideline Communication Function Blocks on PROFIBUS DP and PROFINET IO".

Functional description

"WRREC" is an asynchronous instruction. Processing takes place across several calls. You start the data record transfer by calling "WRREC" with REQ = 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS. The two central bytes of STATUS correspond to the RET_VAL output parameter of the instructions that operate asynchronously.

See also: Difference between synchronous and asynchronous instructions (Page 596)

Note that you must assign the same value to the actual parameter of RECORD for all "WRREC" calls that belong to one and the same job. The same applies to the actual parameters of LEN.

The transfer of the data record is complete when the output parameter BUSY has the value FALSE.

Calling WRREC in the SYNCUP system state of an S7-1500-R/H system

When you call the WRREC instruction in the SYNCUP system state of an S7-1500-R/H system, the error codes 0x7001 and 0x7002 are always returned. Even if an error occurs, for example, no other error code is provided when the data record target does not exist. In this case, the correct error code is only supplied after the SYNCUP has finished.

If you call the WRREC instruction in the SYNCUP system state of an S7-1500-R/H system, no data record request is triggered. The data record request is not triggered until you call the instruction again after the SYNCUP has ended. It is therefore recommended to always call WRREC cyclically.

Parameter

The following table shows the parameters of the "WRREC" instruction:

Parameter	Declaration	Data type**	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T*, C* or constant	REQ = 1: Transfer data record
ID	Input	HW_IO	I, Q, M, D, L or constant	ID number of the hardware component (DP/PROFINET IO) The number is assigned automatically and is stored in the properties of the component or of the interface in the hardware configuration.
INDEX	Input	DINT	I, Q, M, D, L or constant	Data record number
LEN	Input	UINT	I, Q, M, D, L or constant	(hidden) Maximum length of the data record to be transferred in bytes
DONE	Output	BOOL	I, Q, M, D, L	Data record was transferred
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the writing process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information For interpretation of the STATUS parameter, see Parameter STATUS (Page 2276).
RECORD	InOut	VARIANT	I, Q, M, D, L	Data record

* For S7-1500 only.

** There is no implicit conversion in STL, which is why the range of valid data types may be limited. During programming in STL, please note the valid data types. These are displayed in the parameter tooltip.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

If you use "WRREC" to write a data record for PROFINET IO, negative values in the INDEX and LEN parameters will be interpreted as an unsigned 16-bit integer.

Parameter STATUS

For interpretation of the STATUS parameter, see Parameter STATUS (Page 2276).

Examples

You can find an example here: Use the program example of the parameter data record with WRREC & RDREC (Page 2234).

You can find another example here: Use the program example of the control data record with WRREC & RDREC (Page 2244).

You can find additional information and the program code for the examples here: Sample Library for Instructions (Page 593).

See also

Overview of the types of data records (Page 2206)

Use the program example of the parameter data record with WRREC & RDREC

Introduction

In the following example you read the parameter data record of a digital input module via RDREC and rewrite the data record via WRREC. When doing this you change the input delay.

You can find additional information on the data records in the Overview of the types of data records (Page 2206).

Requirement

Create the following tags for storing the data in a global data block:

- two tags for general use and seven tags for WRREC:

SLI_gDB_WRREC_Para			
	Name	Data type	Start value
1	Static		
2	id	HW_IO	257
3	dataRecNbr	DInt	3
4	startWrite	Bool	false
5	maxWriteLength	UInt	4
6	writeCheckedValid	Bool	false
7	writeBusy	Bool	false
8	writeError	Bool	false
9	writeStatus	DWord	16#0
10	wrMemErrStat	DWord	16#0

- a structure for the data record at WRREC:

11	writeParaRecord	Struct	
12	Version	USInt	16
13	LengthPerChannel	USInt	2
14	ClearanceDiagnosis	Array[0..7] of Bool	
15	InputDelay_ProcessEvent	Array[0..7] of Bool	
16	InputDelay_ProcessEvent[0]	Bool	1
17	InputDelay_ProcessEvent[1]	Bool	false
18	InputDelay_ProcessEvent[2]	Bool	false
19	InputDelay_ProcessEvent[3]	Bool	false
20	InputDelay_ProcessEvent[4]	Bool	false
21	InputDelay_ProcessEvent[5]	Bool	false
22	InputDelay_ProcessEvent[6]	Bool	false
23	InputDelay_ProcessEvent[7]	Bool	false

- eight tags for RDREC:

24	startRead	Bool	false
25	maxReadLength	UInt	4
26	readLength	UInt	0
27	readCheckedValid	Bool	false
28	readBusy	Bool	false
29	readError	Bool	false
30	readStatus	DWord	16#0
31	reMemErrStat	DWord	16#0

- a structure for the data record at RDREC:

32	readParaRecord	Struct	
33	Version	USInt	0
34	LengthPerChannel	USInt	0
35	ClearanceDiagnosis	Array[0..7] of Bool	
36	InputDelay_ProcessEvent	Array[0..7] of Bool	
37	InputDelay_ProcessEvent[0]	Bool	false
38	InputDelay_ProcessEvent[1]	Bool	false
39	InputDelay_ProcessEvent[2]	Bool	false
40	InputDelay_ProcessEvent[3]	Bool	false
41	InputDelay_ProcessEvent[4]	Bool	false
42	InputDelay_ProcessEvent[5]	Bool	false
43	InputDelay_ProcessEvent[6]	Bool	false
44	InputDelay_ProcessEvent[7]	Bool	false

Create the following local tags in an "SLI_FB_WRREC_Para" FB.

7	Static		
8	WRREC_Instance	WRREC	
9	RDREC_Instance	RDREC	
10	readCheck	Bool	false
11	writeCheck	Bool	false
12	readLength	UInt	0
13	readBYTE	Struct	
14	writeBYTE	Struct	

Interconnect parameters: "SLI_itemisePara_WRREC_Para" FC

To transfer the values of a bit sequence (0..7) create the FC "SLI_itemisePara_WRREC_Para".

Create the following local tags in it.

SLI_itemisePara_WRREC_Para		
	Name	Data type
1	Input	
2	in0	Bool
3	in1	Bool
4	in2	Bool
5	in3	Bool
6	in4	Bool
7	in5	Bool
8	in6	Bool
9	in7	Bool
10	Output	
11	out0	Bool
12	out1	Bool
13	out2	Bool
14	out3	Bool
15	out4	Bool
16	out5	Bool
17	out6	Bool
18	out7	Bool

Network 1: According to the example of network 1, create interconnections for all bits (0..7).

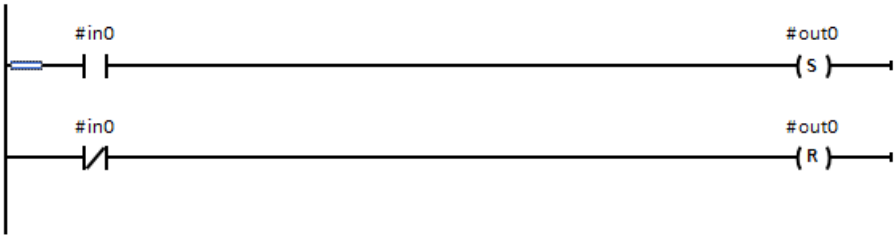

Interconnect parameters: "SLI_FB_WRREC_Para" FB

Create the following interconnections in the "SLI_FB_WRREC_Para" FB.

Network 1: Interconnect the parameters of the "RDREC" instruction as follows.

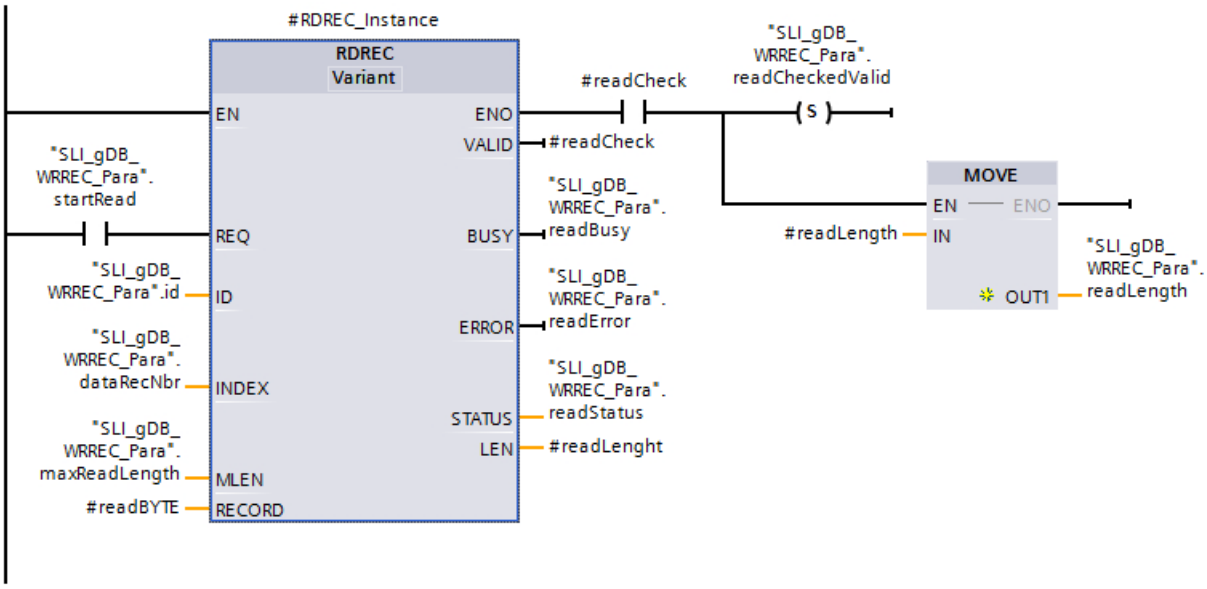

Network 2: Below interconnect all tags of the structure “#readBYTE“ with the structure “readParaRecord“. Part 1 of the network:

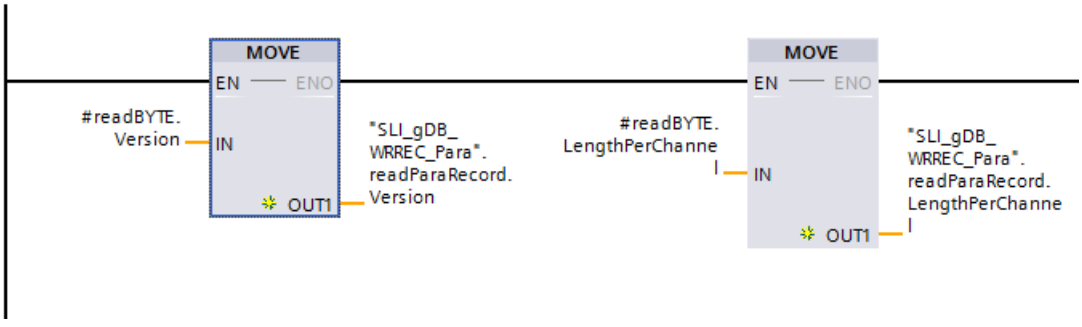

Part 2 of the network: Interconnect the tags "ClearanceDiagnosis" and "InputDelay_ProcessEvent" according to the example below.

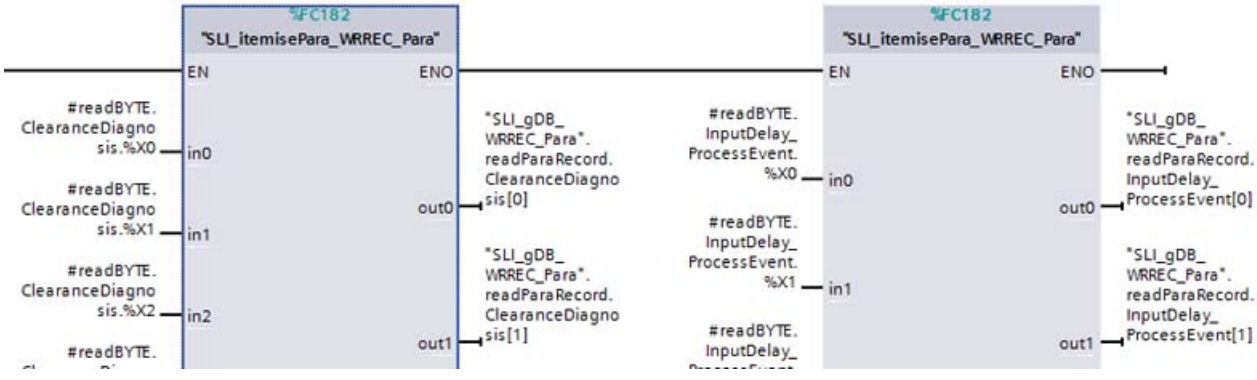

Network 3: Then interconnect the tags for the error situation of RDREC.

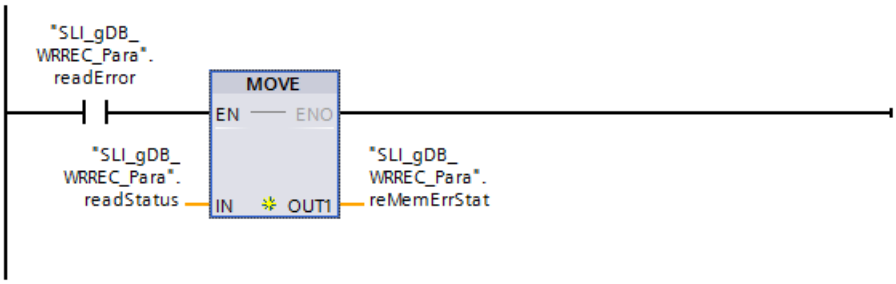

Network 4: Then interconnect the tags for ending RDREC.

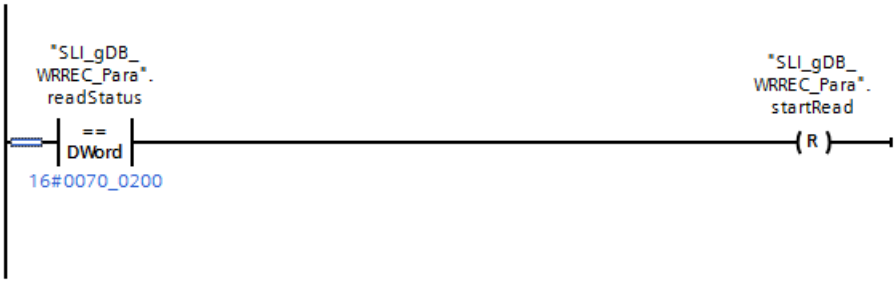

Network 5: Below interconnect all tags of the structure “writeParaRecord“ with the structure “#writeBYTE“. Part 1 of the network:

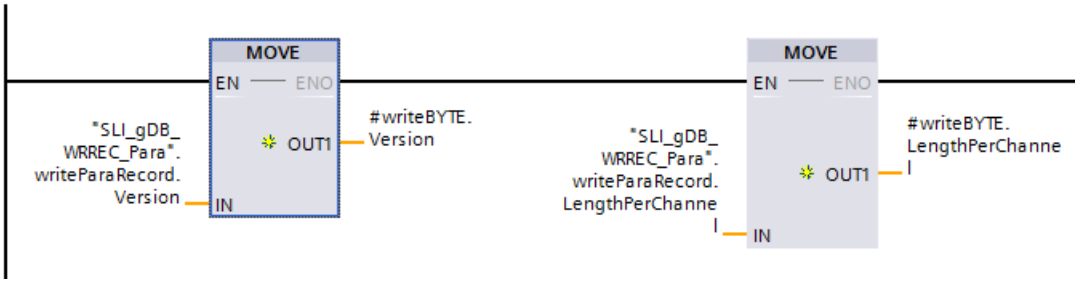

Part 2 of the network:

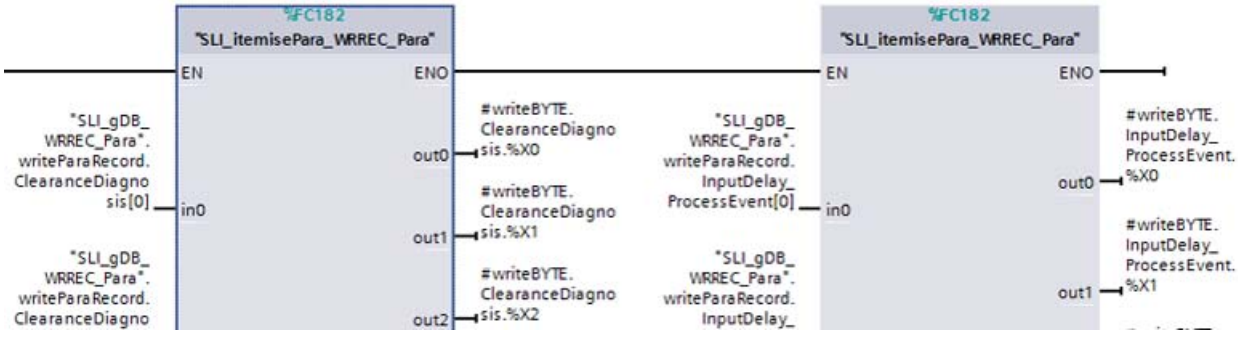

Network 6: Interconnect the parameters of the "WRREC" instruction as follows.

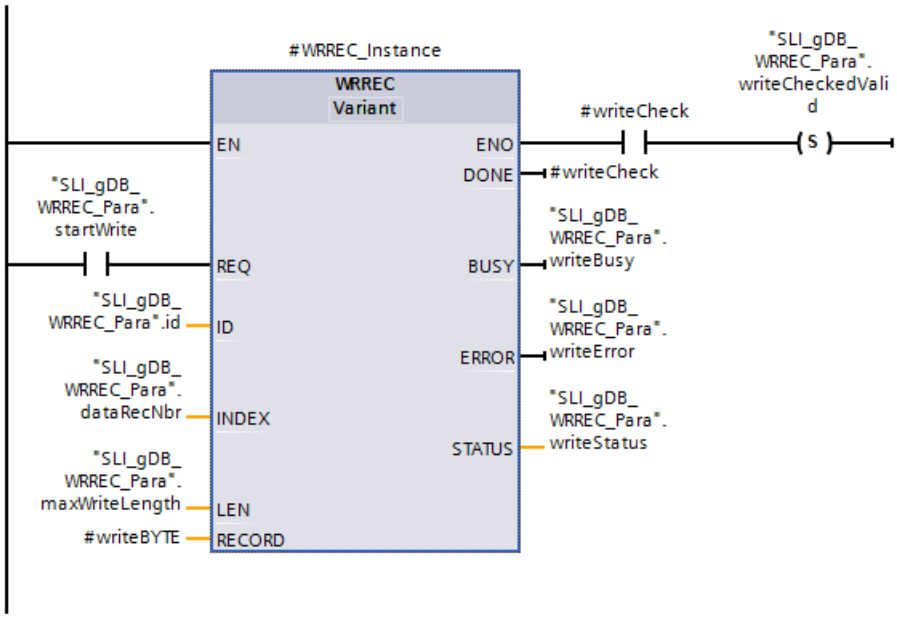

Network 7: Then interconnect the tags for the error situation of WRREC.

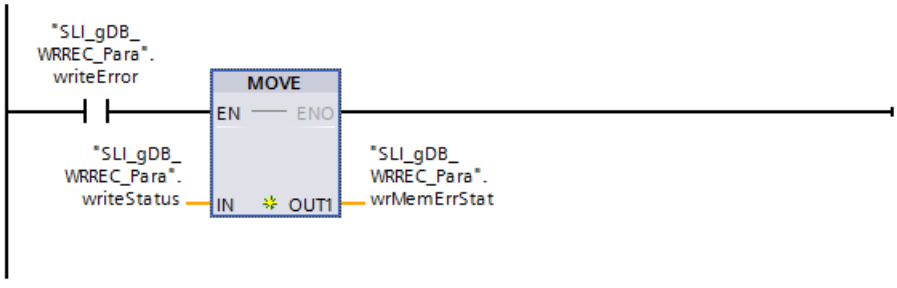

Network 8: Then interconnect the tags for ending WRREC and calling RDREC again.

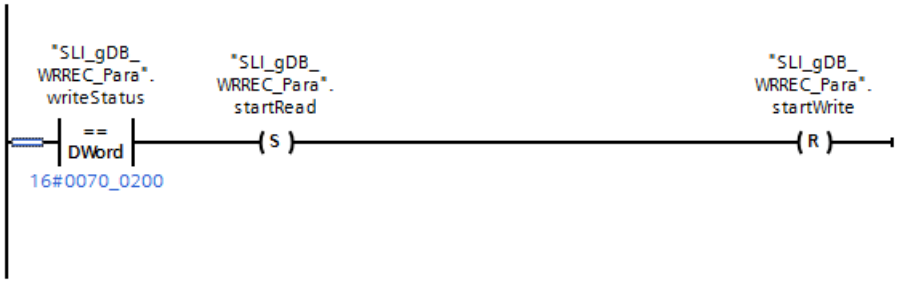

Result of network 1:

You create an input module and read the hardware ID in the properties of the input module. You store the hardware identifier via the input parameter ID ("id"). Using the input parameter INDEX ("dataRecNbr") the data record with the number "3" of the module is called. The data record corresponds to the channel with the same name.

SLI_gDB_WRREC_Para

	Name	Data type	Start value	Monitor value
1	Static			
2	id	HW_IO	257	16#0101
3	dataRecNbr	DInt	3	3

When the input parameter REQ ("startRead") returns the signal state "TRUE", the "RDREC" instruction is started. Via the input parameter ID ("id") the instruction "RDREC" calls the input module.

Over several calls the "RDREC" instruction reads out the data record and saves the data record using the parameter RECORD ("#readBYTE"). The length of the read data record is stored using the output parameter LEN ("#readLength"). According to the value of the input parameter MLEN ("maxReadLength"), the data record information that is to be read is limited to 4 bytes.

During processing the output parameter BUSY ("readBusy") is set to "TRUE" and VALID ("#readCheck") to "FALSE".

A successful reading result is output as follows when processing is complete.

- The output parameter VALID ("#readCheck") is set to "TRUE" and sets the tag "readCheckedValid" permanently to "TRUE".
- In addition to this the value of the local tag "#readLength" is stored permanently in the tag "readLength".

The block status is displayed at the output parameter STATUS ("readStatus"). Processing takes place without errors.

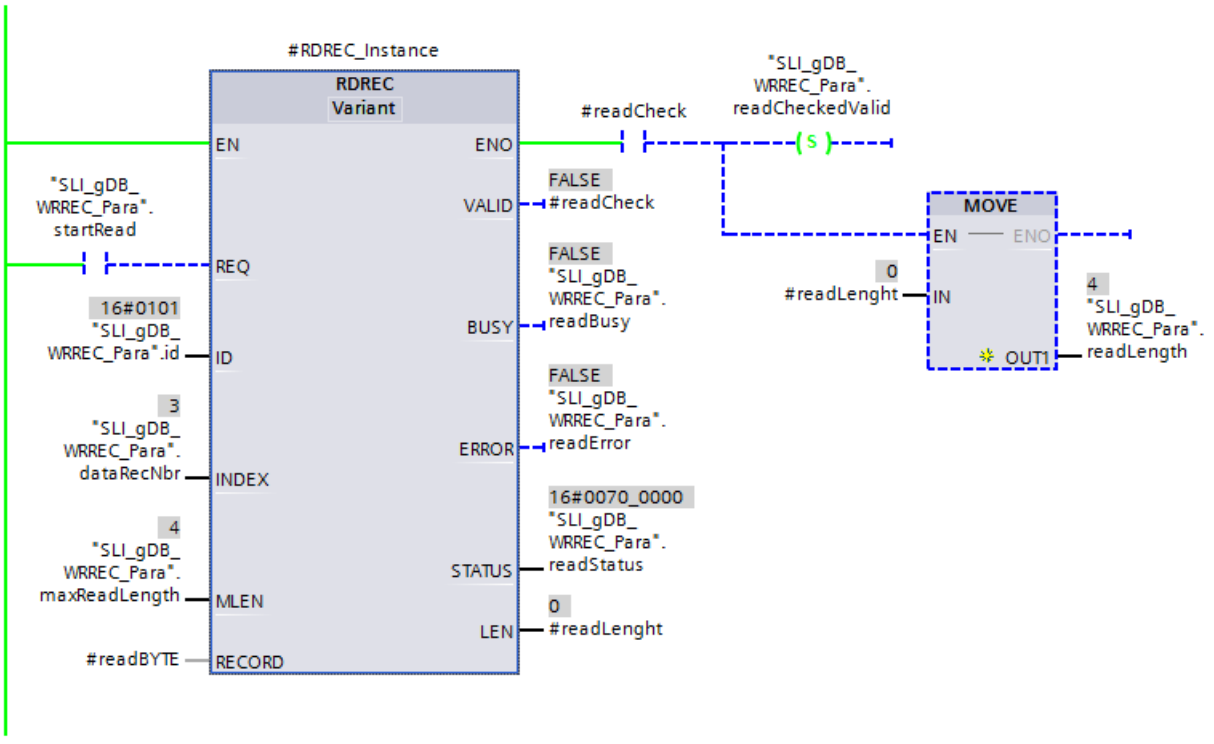

Result of network 2:

Using MOVE instructions and the FC SLI_itemisePara_WRREC_Para, the structure "#readBYTE" transfers the values of the tags to the tags of the structure "readParaRecord" with the same names. To improve understanding, the tags of the “readParaRecord“ structure are represented by descriptive names. The structure is described in the device manual of the module. For example "InputDelay_ProcessEvent" indicates an input delay of 3.2 ms. For the status of the channel (“ClearanceDiagnosis“) no disruption is output (bits 0 to 8 are “FALSE“.

32	readParaRecord	Struct		
33	Version	USInt	0	16
34	LengthPerChannel	USInt	0	2
35	ClearanceDiagnosis	Array[0..7] of Bool		
36	InputDelay_ProcessEvent	Array[0..7] of Bool		
37	InputDelay_ProcessEvent[0]	Bool	false	FALSE
38	InputDelay_ProcessEvent[1]	Bool	false	TRUE
39	InputDelay_ProcessEvent[2]	Bool	false	TRUE
40	InputDelay_ProcessEvent[3]	Bool	false	FALSE
41	InputDelay_ProcessEvent[4]	Bool	false	FALSE
42	InputDelay_ProcessEvent[5]	Bool	false	FALSE
43	InputDelay_ProcessEvent[6]	Bool	false	FALSE
44	InputDelay_ProcessEvent[7]	Bool	false	FALSE

Result of network 3:

When “readError“ supplies the signal status “TRUE“, the value of the tag “readStatus“ is stored in the tag “reMemErrStat“.

Result of network 4:

When the instruction “RDREC“ has been executed, “readStatus“ supplies the value “16#0070_0200“. After this a resetter sets the tag “startRead“ to “FALSE“

Result of network 5:

Using MOVE instructions and the FC SLI_itemisePara_WRREC_Para, the structure "writeParaRecord" transfers the values of the tags to the tags of the structure "#writeBYTE" with the same names. To improve understanding, the tags of the “writeParaRecord“ structure are represented by descriptive names.

Result of network 6:

When the input parameter REQ ("startWrite") returns the signal state "TRUE", the "WRREC" instruction is started.

The new data for the data record is stored at the RECORD (“#writeBYTE“) parameter. Over several calls, the instruction “WRREC“ writes the new data to the data record "3". According to the value of the input parameter LEN ("maxWriteLength"), the data record information that is to be written is limited to 4 bytes.

During processing the output parameter BUSY ("writeBusy") is set to "TRUE" and DONE ("#writeCheck") to "FALSE".

A successful writing result is output as follows when processing is complete.

- The output parameter DONE ("#writeCheck") is set to "TRUE".
- When "#writeCheck" supplies the signal state "TRUE" the tag "writeCheckedValid" is set permanently to "TRUE".

The block status is displayed at the output parameter STATUS ("writeStatus"). Processing takes place without errors.

4		startWrite	Bool	false	FALSE
5		maxWriteLength	UInt	4	4
6		writeCheckedValid	Bool	false	TRUE
7		writeBusy	Bool	false	FALSE
8		writeError	Bool	false	FALSE
9		writeStatus	DWord	16#0	16#0070_0000
10		wrMemErrStat	DWord	16#0	16#0000_0000

Result of network 7:

When “writeError“ supplies the signal status “TRUE“, the value of the tag “writeStatus“ is stored in the tag “wrMemErrStat“.

Result of network 8:

When the instruction “WRREC“ has been executed, “writeStatus“ supplies the value “16#0070_0200“. After this the resetter sets the tag “startWrite“ to “FALSE“ The tag “startRead“ is set to “TRUE“ after which the instruction “RDREC“ is executed again. The structure “readParaRecord“ shows the newly written values of the data record. Since in “InputDelay_ProcessEvent“ the bit 0 supplies the signal state “TRUE“, the channel 3 now has an input delay off 0.1 ms.

32	readParaRecord	Struct		
33	Version	USInt	0	16
34	LengthPerChannel	USInt	0	2
35	ClearanceDiagnosis	Array[0..7] of Bool		
36	InputDelay_ProcessEvent	Array[0..7] of Bool		
37	InputDelay_ProcessEvent[0]	Bool	false	TRUE
38	InputDelay_ProcessEvent[1]	Bool	false	FALSE
39	InputDelay_ProcessEvent[2]	Bool	false	FALSE
40	InputDelay_ProcessEvent[3]	Bool	false	FALSE
41	InputDelay_ProcessEvent[4]	Bool	false	FALSE
42	InputDelay_ProcessEvent[5]	Bool	false	FALSE
43	InputDelay_ProcessEvent[6]	Bool	false	FALSE
44	InputDelay_ProcessEvent[7]	Bool	false	FALSE

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

WRREC: Write data record (Page 2232)

RDREC: Read data record (Page 2213)

Use the program example of the control data record with WRREC & RDREC

Introduction

In the following example, you write the control data record 196 using WRREC and check this by reading the data record out with RDREC. For the configuration select the central setup of an S7-1500

You can find additional information on the data records in Overview of the types of data records (Page 2206).

Requirement

Note

Enabling configuration control

To be able to write the control data record, the configuration control (option handling) of the CPU must be enabled. You will find these in “CPU properties > Configuration control”.

Create the following tags for storing the data in a global data block:

- three tags for general use and six for WRREC:

SLI_gDB_WRREC_Control			
	Name	Data type	Start value
1	Static		
2	id	HW_IO	33
3	dataRecNbr	DInt	196
4	maxLength	UInt	0
5	configDone	Bool	false
6	checkedValid_WR	Bool	false
7	busyWR	Bool	false
8	errorWR	Bool	false
9	statusWR	DWord	16#0
10	memErrStatusWR	DWord	16#0

- a structure for the data at WRREC (data record 196):

11	recConfigCtrl_WR	Struct	
12	BlockLength	USInt	8
13	BlockID	USInt	196
14	Version	USInt	4
15	Subversion	USInt	0
16	Slot_0	USInt	0
17	Slot_1	USInt	1
18	Slot_2	USInt	2
19	Slot_3	USInt	3

The data record is structured as follows:

– "BlockLength" indicates the length of the data record in Byte.

– "BlockID" indicates the data record number.

– "Version" indicates the type of structure. A central structure is used for the example. In a central structure "Subversion" has the value "0".

– "Slot_0" to "Slot_.." correspond to the slots used. The values of the slots indicate the numbering to be used.

- six tags for RDREC:

20	startRead	Bool	false
21	checkedValid_RD	Bool	false
22	lengthRD	UInt	0
23	busyRD	Bool	false
24	errorRD	Bool	false
25	statusRD	DWord	16#0

- a structure for the data at RDREC:

26	recConfigCtrl_RD	Struct	
27	BlockLength	USInt	0
28	BlockID	USInt	0
29	Version	USInt	0
30	Subversion	USInt	0
31	Slot_0	USInt	0
32	Slot_1	USInt	0
33	Slot_2	USInt	0
34	Slot_3	USInt	0

The instruction “WRREC“ for writing the control data recording must be called in a startup OB. In the startup OB create the following local tags for passing on the data.

4	Temp		
5	checkWR	Bool	
6	statusWR	DWord	

Create the following local tags in an FB. Call the FB in a program cycle OB (OB1).

7	Static			
8	RDREC_Instance	RDREC		
9	memErrStatus	DWord	16#0	Set in IDB
10	checkRD	Bool	false	Non-retain
11	lengthRD	UInt	0	Non-retain

Interconnecting parameters in the startup OB

Network 1: Interconnect the parameters of the "WRREC" instruction as follows.

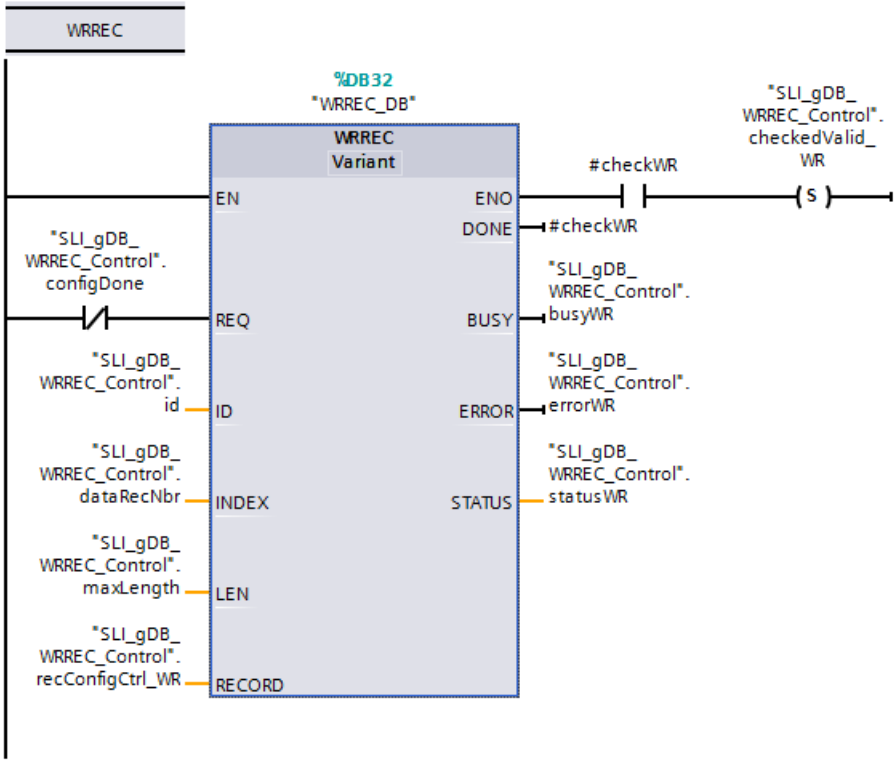

Network 2: Then interconnect the parameters for the error situation of WRREC.

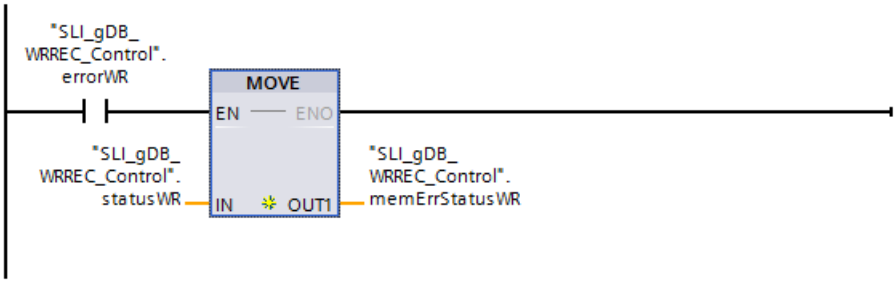

Network 3: Then interconnect the parameters for ending WRREC.

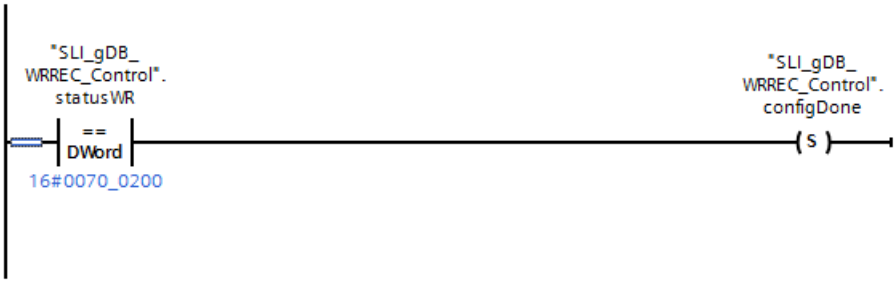

Network 4: This is followed by interconnecting the parameters for requesting the repeated call of WRREC.

Interconnect parameters in the FB

Network 1: Interconnect the parameters of the "RDREC" instruction as follows.

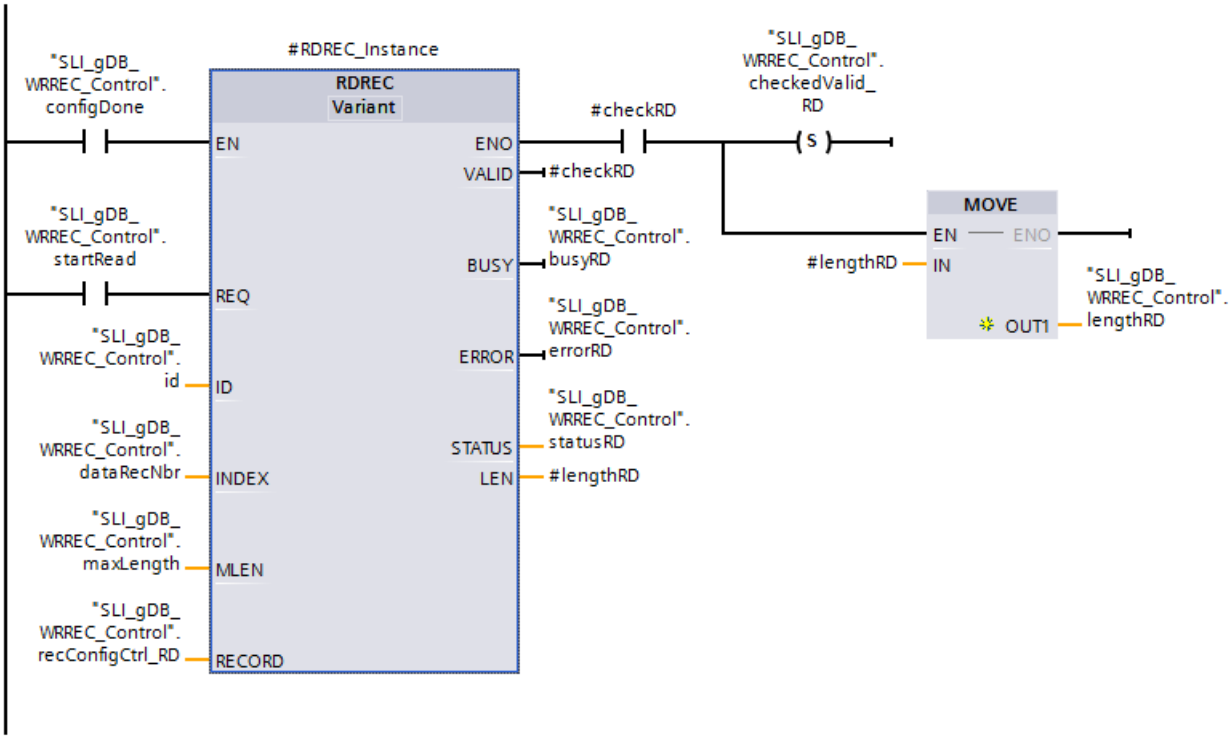

Network 2: Then interconnect the parameters for the error situation of RDREC.

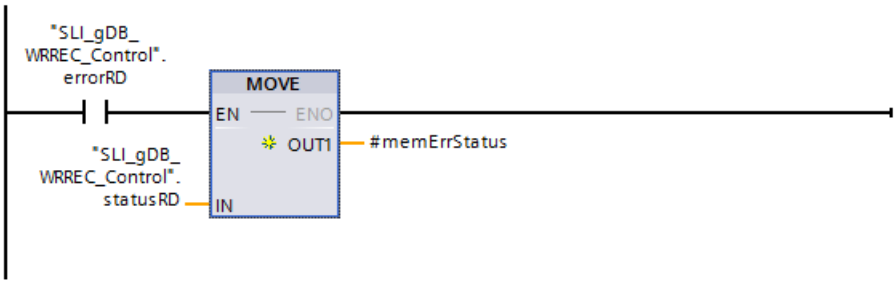

Network 3: Then interconnect the parameters for ending RDREC.

General parameters

You store the hardware identifier of the configuration of the CPU (“33“) via the input parameter ID (“id“). You store the control data record ("196") using the input parameter INDEX ("dataRecNbr"). According to the value of the tag maxLength" (with the value "0") the amount of data record information to be taken into account is unlimited.

Note

Loading in the CPU and processing of the program in startup.

1. To load the program, set the CPU to "STOP" mode.
2. Set the project to "Online" mode.
3. Set the CPU to "RUN" mode. The "Startup" mode is worked through.

In "Startup” mode the CPU works through the program sections located in the startup OB.

Startup OB: Result of network 1

Since the input parameter REQ supplies the signal state "TRUE", the "WRREC" instruction is started. The normally closed contact (“configDone“) has the signal state "FALSE".

The data for the control data record is stored at the RECORD (“recConfigCtrl_WR“) parameter. To improve understanding, the tags of the “recConfigCtrl_WR“ structure are represented by clear names and have the same structure as described in the system manual..

11	recConfigCtrl_WR	Struct		
12	BlockLength	USInt	8	8
13	BlockID	USInt	196	196
14	Version	USInt	4	4
15	Subversion	USInt	0	0
16	Slot_0	USInt	0	0
17	Slot_1	USInt	1	1
18	Slot_2	USInt	2	2
19	Slot_3	USInt	3	3

Over several calls, the instruction “WRREC“ writes the new data to the control data record. During processing of WRREC the output parameter BUSY ("busyWR") is set to "TRUE" and DONE ("#checkWR") to "FALSE".

A successful writing result is output as follows when processing is complete.

- The output parameter DONE ("#checkWR") is set to "TRUE" and sets the tag "checkedValid_WR" permanently to "TRUE".

The block status is displayed at the output parameter STATUS ("statusWR"). Processing takes place without errors.

SLI_gDB_WRREC_Control

	Name	Data type	Start value	Monitor value
1	Static			
2	id	HW_IO	33	16#0021
3	dataRecNbr	DInt	196	196
4	maxLength	UInt	0	0
5	configDone	Bool	false	TRUE
6	checkedValid_WR	Bool	false	TRUE
7	busyWR	Bool	false	FALSE
8	errorWR	Bool	false	FALSE
9	statusWR	DWord	16#0	16#0000_0000
10	memErrStatusWR	DWord	16#0	16#0000_0000

Startup OB: Result of network 2

If an error occurs when processing WRREC:

When “errorWR“ supplies the signal status “TRUE“, the value of the tag “statusWR“ is stored in the tag “memErrStatusWR“.

Startup OB: Result of network 3

When the instruction “WRREC“ has been executed, “statusWR“ supplies the value “16#0070_0200“. The tag "configDone" is then set to "TRUE". This ends the processing of WRREC.

Startup OB: Result of network 4

As long as the processing of the "WRREC" instruction has not yet ended ("configDone" has the value"FALSE" and "busyWR" has the value "TRUE") processing returns to network 1.

FB: Result of network 1

After processing the program sections located in the startup OB the CPU is in the “RUN“ mode.

When the input parameter EN ("configDone") supplies the signal state "TRUE", the "RDREC" instruction is released for processing. When the input parameter REQ ("startRead") returns the signal state "TRUE", the "RDREC" instruction is started.

Over several calls the "RDREC" instruction reads out the control data record and saves the data record using the parameter RECORD ("recConfigCtrl_RD").

26	recConfigCtrl_RD	Struct		
27	BlockLength	USInt	0	8
28	BlockID	USInt	0	196
29	Version	USInt	0	4
30	Subversion	USInt	0	0
31	Slot_0	USInt	0	0
32	Slot_1	USInt	0	1
33	Slot_2	USInt	0	2
34	Slot_3	USInt	0	3

The length of the read data record is stored using the output parameter LEN ("#readLength"). During processing of RDREC the output parameter BUSY ("busyRD") is set to "TRUE" and VALID ("#checkRD") to "FALSE".

A successful reading result is output as follows when processing is complete.

- The output parameter VALID ("#checkRD") is set to "TRUE". #checkRD then sets the tag "checkedValid_RD" permanently to "TRUE".
- In addition to this the value of the local tag "#lengthRD" is stored permanently in the tag "lengthRD".

The block status is displayed at the output parameter STATUS ("statusRD"). Processing takes place without errors.

20	startRead	Bool	false	FALSE
21	checkedValid_RD	Bool	false	TRUE
22	lengthRD	UInt	0	8
23	busyRD	Bool	false	FALSE
24	errorRD	Bool	false	FALSE
25	statusRD	DWord	16#0	16#0070_0000

FB: Result of network 2

If an error occurs when processing RDREC:

When “errorRD“ supplies the signal status “TRUE“, the value of the tag “statusRD“ is stored in the local tag “#memErrStatus“.

FB: Result of network 3

When the instruction “RDREC“ has been executed and has checked the data, “#checkRD“ supplies the value “TRUE“. The tag "startRead" is then set to "FALSE". Execution of the “RDREC“ instruction is then ended.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

WRREC: Write data record (Page 2232)

RDREC: Read data record (Page 2213)

GETIO: Read process image (S7-1200, S7-1500)

Description

You use the instruction "GETIO" to consistently read out all inputs of a DP standard slave/ PROFINET IO device. The instruction "GETIO" calls the instruction "DPRD_DAT (Page 2338)". If there was no error during the data transmission, the data that has been read is entered in the destination area indicated by INPUTS .

The following applies to the destination area:

- If you have configured a length of exactly one byte for the selected component, you must use a single byte as the destination area.
- If you have configured a length of multiple bytes for the selected component, you must use an ARRAY of BYTE as destination area whose length is identical to the number of configured bytes.

Only use BYTE or ARRAY of BYTE as data type of the destination area. In particular, do not use an ARRAY of STRUCT. If you are using an ARRAY of STRUCT, you will create gaps in the destination area.

If you read from a DP standard slave with a modular configuration or with several DP identifiers, you only access the data of one component/DP identifier at the configured start address with a "GETIO" call.

Parameter

The following table shows the parameters of the "GETIO" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUB-MODULE	I, Q, M, D, L or constant	Hardware ID of the DP standard slave / PROFINET IO device.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPRD_DAT (Page 2338)" in the form DW#16#40xxxx00.
LEN	Output	INT	I, Q, M, D, L	Amount of data read in bytes
INPUTS	InOut	VARIANT	I, Q, M, D, L	Destination area for the read data. It must have the same length as the range that you configured for the selected DP standard slave / PROFINET IO device. Only the data type BYTE or ARRAY of BYTE is valid.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

See also: DPRD_DAT: Read consistent data of a DP standard slave (Page 2338)

Example

You can find the example here: Program example for GETIO & SETIO (Page 2254).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

SETIO: Transfer process image (S7-1200, S7-1500)

Description

Transfer the data using the SETIO instruction. The source area of the transfer is defined using the OUTPUTS parameter. The data is transferred in a consistent state to the addressed DP standard slave
/ PROFINET IO device. If you have configured the relevant address area of the DP standard slave / PROFINET IO device as a consistent range in a process image, the data is transferred to the process image. "SETIO" calls the "DPWR_DAT (Page 2340)" instruction during this transfer.

The source range must have the same length that you configured for the selected component.

In the case of a DP standard slave / PROFINET IO device with modular configuration or with several DP identifiers, you can only access one DP identifier / component per "SETIO" call.

Parameter

The following table shows the parameters of the "SETIO" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUBMOD-ULE	I, Q, M, D, L or con-stant	Hardware ID of the DP standard slave / PROFINET IO device.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPWR_DAT (Page 2340)" in the form DW#16#40xxxx00.
OUTPUTS	InOut	VARIANT	I, Q, M, D, L	Source range for the data to be written. It must have the same length as the range that you configured for the selected DP standard slave / PROFINET IO device. Only the data type BYTE is valid.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

See also: DPWR_DAT: Write consistent data of a DP standard slave (Page 2340)

Example

You will find the example here: Program example for GETIO & SETIO (Page 2254).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

Program example for GETIO & SETIO

Introduction

In the following example, you read out the channels of a digital input module via GETIO and write new values to the channels of a digital output module via SETIO. You then read out the changes via GETIO and via the digital input module.

Requirements

- An IO device/DP slave is created with a digital input module and a digital output module.

  Note

  Change with SETIO, read out changes with GETIO

  The digital input module must be connected to the digital output module in order that the written value can be read via SETIO using GETIO

- To store the data, create nine tags and the two structures "readinputs" and "writeOutputs" in a global data block.
  Use the tag "id_Read" to store the hardware identifier of the digital input module. Use the tag "id_Write" to store the hardware identifier of the digital output module.

  Note

  Find hardware identifier

  Use "PLC tags > System constants" to find the hardware identifier. For example, for the digital input module, search for "Local~<Name of the DI module>", with the data type "HW_Submodule". The associated cell "value" contains the hardware identifier.

SLI_gDB_GETIO

	Name	Data type	Start value
1	Static		
2	id_Read	HW_SUBMODULE	257
3	executeRead	Bool	false
4	readInputs	Struct	
5	Channel_0-7	Array[0..7] of Bool	
6	Channel_8-15	Array[0..7] of Bool	
7	Channel_16-23	Array[0..7] of Bool	
8	Channel_24-31	Array[0..7] of Bool	
9	statusRead	DWord	16#0
10	lengthRead	Int	0
11	doneRead	Bool	false
12	id_Write	HW_SUBMODULE	258
13	executeWrite	Bool	false
14	writeOutputs	Struct	
15	Channel_0-7	Array[0..7] of Bool	
16	Channel_8-15	Array[0..7] of Bool	
17	Channel_16-23	Array[0..7] of Bool	
18	Channel_24-31	Array[0..7] of Bool	
19	statusWrite	DWord	16#0
20	doneWrite	Bool	false

- To forward the data, create the following local tags in an "SLI_FB_GETIO_SETIO" FB.

7	Static			
8	GETIO_Instance	GETIO		
9	SETIO_Instance	SETIO		
10	readBYTE	Struct		Non-retain
11	Channel_0-7	Byte	16#0	Non-retain
12	Channel_8-15	Byte	16#0	Non-retain
13	Channel_16-23	Byte	16#0	Non-retain
14	Channel_24-31	Byte	16#0	Non-retain
15	writeBYTE	Struct		Non-retain
16	Channel_0-7	Byte	16#0	Non-retain
17	Channel_8-15	Byte	16#0	Non-retain
18	Channel_16-23	Byte	16#0	Non-retain
19	Channel_24-31	Byte	16#0	Non-retain

- To change the channel states, set the following bits (channel 1 and 2) in the global data block to "TRUE":

14	writeOutputs	Struct	
15	Channel_0-7	Array[0..7] of Bool	
16	Channel_0-7[0]	Bool	false
17	Channel_0-7[1]	Bool	1
18	Channel_0-7[2]	Bool	1
19	Channel_0-7[3]	Bool	false
20	Channel_0-7[4]	Bool	false
21	Channel_0-7[5]	Bool	false
22	Channel_0-7[6]	Bool	false
23	Channel_0-7[7]	Bool	false

Interconnect parameters: "SLI_itemiseByte_gsIO" FC

To transfer the values of a bit sequence (0..7) create the FC "SLI_itemiseByte_gsIO".

Create the following local tags in it.

SLI_itemiseByte_gsIO		
	Name	Data type
1	Input	
2	in0	Bool
3	in1	Bool
4	in2	Bool
5	in3	Bool
6	in4	Bool
7	in5	Bool
8	in6	Bool
9	in7	Bool
10	Output	
11	out0	Bool
12	out1	Bool
13	out2	Bool
14	out3	Bool
15	out4	Bool
16	out5	Bool
17	out6	Bool
18	out7	Bool

Network 1: According to the example of network 1, create interconnections for all bits (0..7).

Interconnect parameters: "SLI_FB_GETIO_SETIO" FB

Create the following interconnections in the "SLI_FB_GETIO_SETIO" FB. On the IO device/ DP slave, then call the FB "SLI_FB_GETIO_SETIO" in a program cycle OB (OB1).

Network 1: Interconnect the parameters of the "GETIO" instruction as follows.

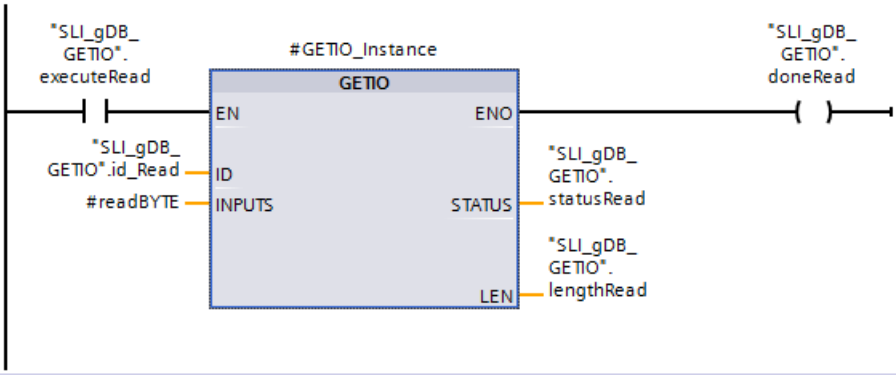

Network 2: Close the instruction "GETIO" as follows.

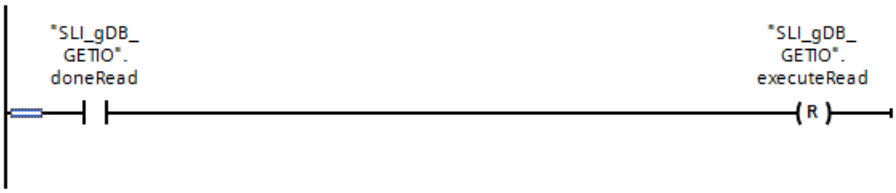

Network 3: Create the interconnections according to the following example to convert the channel values of GETIO. All inputs and outputs (0..7) of the "SLI_itemiseByte_gsIO" FC are interconnected in this case.

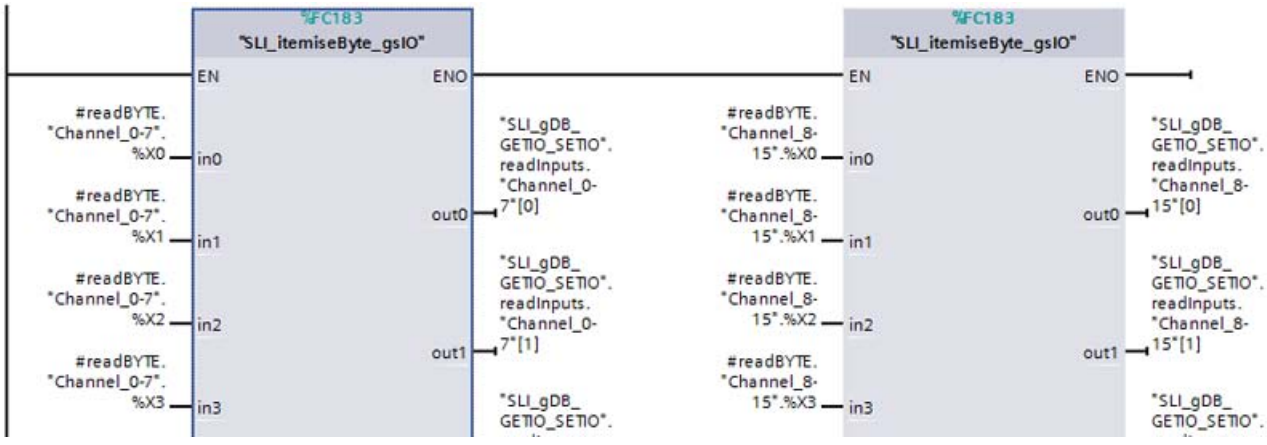

Network 4: Create the interconnections according to the following example to convert the channel values of SETIO. All inputs and outputs (0..7) of the "SLI_itemiseByte_gsIO" FC are interconnected in this case.

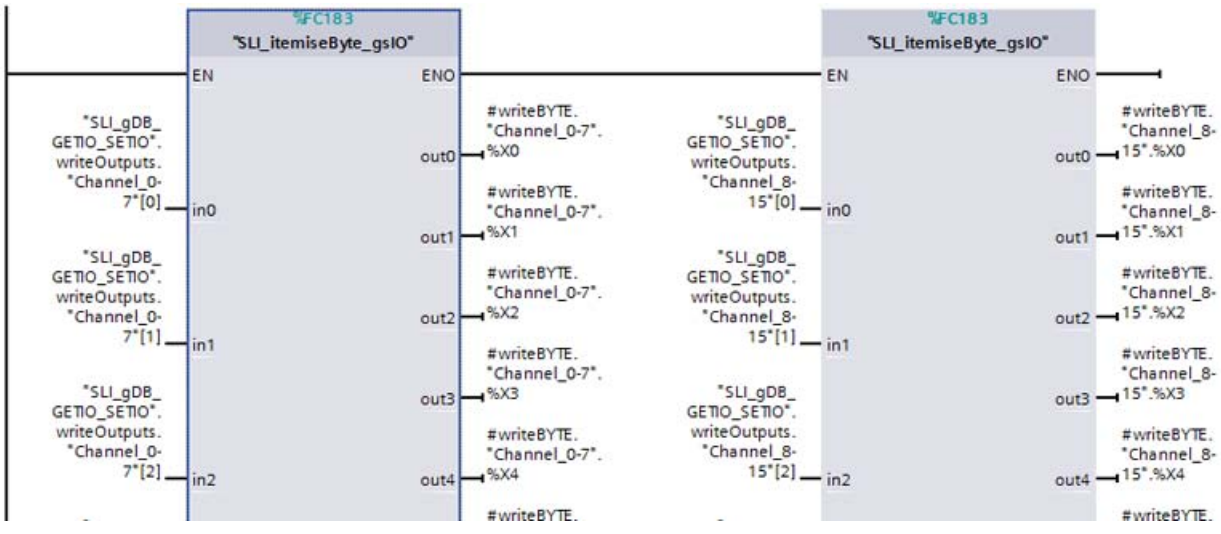

Network 5: Interconnect the parameters of the "SETIO" instruction as follows.

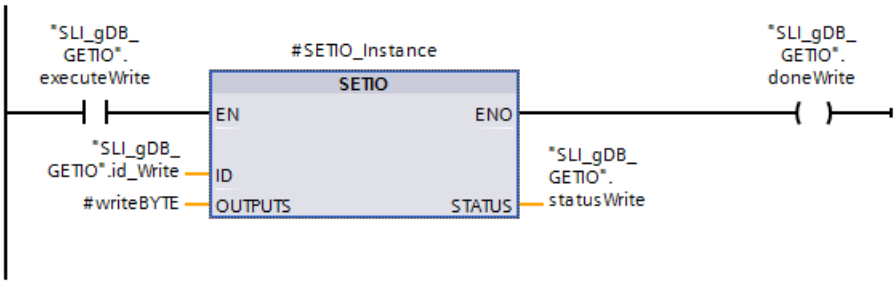

Network 6: Close the instruction "SETIO" as follows.

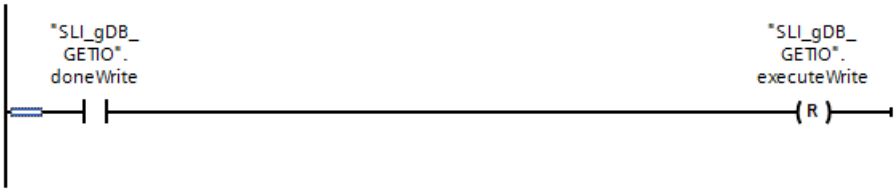

Result of network 1 (GETIO)

When the input parameter EN ("executeRead") returns the signal state "TRUE", the "GETIO" instruction is started. The instruction "GETIO" calls the input module via the input parameter ID ("id_Read"). The channels of the input module are read out via the input parameter INPUTS ("#readBYTE"). The length of the read channels in BYTE is recorded at the output parameter LEN ("lengthRead"). The value "2" references 16 input channels.

The block status is displayed at the output parameter STATUS ("statusRead"). Processing takes place without errors.

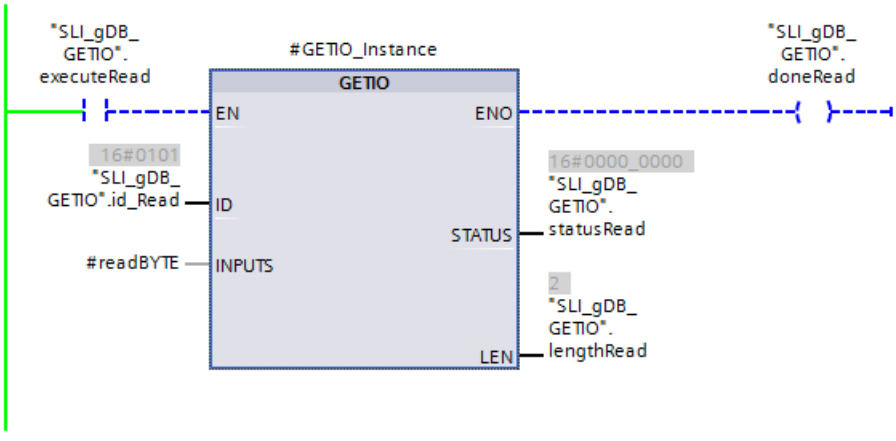

Result of network 2 (GETIO)

When the instruction "GETIO" has been executed, the output parameter ENO ("doneRead") returns the signal state "TRUE". After this, the resetter sets the tag "executeRead" to "FALSE" and ends the processing of "GETIO".

Result of network 3 (GETIO)

Using MOVE instructions and the FC SLI_itemiseByte_gsIO, the structure "#readBYTE" transfers the values of the tags to the tags of the structure "readInputs". The values of the channels 0 to 7 and 8 to 15 are hereby converted from BYTE to array of BOOL.

To improve understanding, the tags of the "readInputs" structure are represented by descriptive names.

Result of network 4 (SETIO)

Using MOVE instructions and the FC SLI_itemiseByte_gsIO, the structure "writeOutputs" transfers the values of the tags to the tags of the structure "#writeBYTE". The values of the channels 0 to 7 and 8 to 15 are converted from array of BOOL to BYTE.

To improve understanding, the tags of the "writeOutputs" structure are represented by descriptive names.

Result of network 5 (SETIO)

When the input parameter EN ("executeWrite") returns the signal state "TRUE", the "SETIO" instruction is started. The instruction "SETIO" calls the output module via the input parameter ID ("id_Write"). New values are written to the channels of the output module via the input parameter OUTPUTS ("#writeBYTE").

The block status is displayed at the output parameter STATUS ("statusWrite"). Processing takes place without errors.

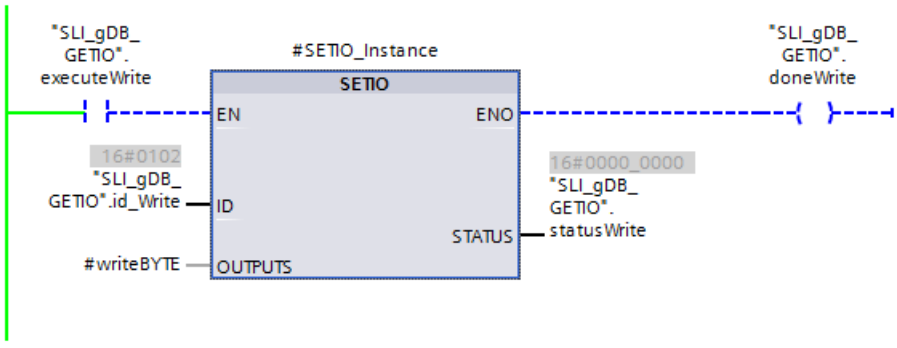

Result of network 6 (SETIO)

When the instruction "SETIO" has been executed, the output parameter ENO ("doneWrite") returns the signal state "TRUE". After this, the resetter sets the tag "executeWrite" to "FALSE" and ends the processing of "SETIO".

Result of reprocessing of GETIO

If the tag "executeRead" is reset to "TRUE", the structure "readInputs" shows the new values on the channels of the input module after the execution of GETIO. The channels 1 and 2 ("Channel_0-7[1]..[2]") now return the signal state "TRUE".

4	readInputs	Struct		
5	Channel_0-7	Array[0..7] of Bool		
6	Channel_0-7[0]	Bool	false	FALSE
7	Channel_0-7[1]	Bool	false	TRUE
8	Channel_0-7[2]	Bool	false	TRUE
9	Channel_0-7[3]	Bool	false	FALSE
10	Channel_0-7[4]	Bool	false	FALSE
11	Channel_0-7[5]	Bool	false	FALSE
12	Channel_0-7[6]	Bool	false	FALSE
13	Channel_0-7[7]	Bool	false	FALSE
14	Channel_8-15	Array[0..7] of Bool		
15	Channel_16-23	Array[0..7] of Bool		
16	Channel_24-31	Array[0..7] of Bool		

Note

Changing the signal state of a channel

If a value is already assigned to a channel on which a signal state is to be changed (for example by a PLC tag), this pre-assignment can make a change by SETIO invalid. Make sure that the signal state of a channel to be changed is not already determined by other factors.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

GETIO: Read process image (Page 2252)

SETIO: Transfer process image (Page 2253)

GETIO_PART: Read process image area (S7-1200, S7-1500)

Description

You use the instruction "GETIO_PART" to consistently read out a related part of the inputs of an IO module. "GETIO_PART" calls the instruction "DPRD_DAT (Page 2338)".

Use the ID input parameter to select the IO module by means of the hardware ID.

You use the OFFSET and LEN parameters to specify the portion of the process image area to be read. If the input area spanned by OFFSET and LEN is not completely covered by the module, the block returns the error code DW#16#40**80B7**00.

The length of the destination area must be larger than or equal to the amount of bytes to be read:

- If there was no error during the data transmission, ERROR receives the value FALSE. The data that is read is written to the destination area defined at the INPUTS parameter.
- If there was an error during the data transmission, ERROR receives the value TRUE. The STATUS parameter receives the error information from "DPRD_DAT".
- If the destination area is greater than LEN, then the first LEN bytes of the destination area will be written. ERROR receives the value FALSE.

Parameter

The following table shows the parameters of the "GETIO_PART" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUBMOD-ULE	I, Q, M, D, L or constant	Hardware identifier of the module
OFFSET	Input	INT	I, Q, M, D, L or constant	Number of the first byte to be read in the process image for the component (smallest possible value: 0).
LEN	Input	INT	I, Q, M, D, L or constant	Number of bytes to be read.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPRD_DAT" in the form DW#16#40xxxx00, if ERROR = TRUE.
ERROR	Output	BOOL	I, Q, M, D, L	Error display: ERROR = TRUE if an error occurs when "DPRD_DAT" is called.
INPUTS	InOut	VARIANT	I, Q, M, D, L	Destination area for read data: If the destination area is greater than LEN, the first LEN bytes of the destination area are written.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

See "RET_VAL" parameter of the "DPRD_DAT (Page 2338)" instruction.

Example

You will find the example here: Program example for GETIO_PART & SETIO_PART (Page 2264).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

SETIO_PART: Transfer process image area (S7-1200, S7-1500)

Description

You can use the "SETIO_PART" instruction to consistently write data from the source area spanned by OUTPUTS to the outputs of an IO module. "SETIO_PART" calls the instruction "DPWR_DAT (Page 2340)".

With the input parameter ID you select the I/O module based on the hardware identified. With the parameters OFFSET and LEN you specify the portion of the process image area to be written for the component addressed by ID. If the output area spanned by OFFSET and LEN is not completely covered by the module, the block returns the error code DW#16#4080B700.

The length of the destination area must be larger than or equal to the amount of bytes to be read:

- If there was no error during the data transmission, ERROR receives the value FALSE.
- If there was an error during the data transmission, ERROR receives the value TRUE, and STATUS receives the error information of "DPWR_DAT".
- If the source area is greater than LEN, the first LEN bytes from OUTPUTS are transferred. ERROR receives the value FALSE.

Parameters

The following table shows the parameters of the "SETIO_PART" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUB-MODULE	I, Q, M, D, L or constant	Hardware identifier of the IO module.
OFFSET	Input	INT	I, Q, M, D, L or constant	Number of the first byte to be written in the process image for the component (smallest possible value: 0).
LEN	Input	INT	I, Q, M, D, L or constant	Number of bytes to be written.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPWR_DAT" in the form DW#16#40xxxx00, if ERROR = TRUE.
ERROR	Output	BOOL	I, Q, M, D, L	Error display: ERROR = TRUE if an error occurs when "DPWR_DAT" is called.
OUTPUTS	InOut	VARIANT	I, Q, M, D, L	Source range for the data to be written: If the source area is greater than LEN, the first LEN bytes are transferred from OUT-PUTS.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

See instruction "DPWR_DAT (Page 2340)".

Example

You will find the example here: Program example for GETIO_PART & SETIO_PART (Page 2264).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

Program example for GETIO_PART & SETIO_PART

Introduction

In the following example, you read out the channels of a digital input module via GETIO_PART and write new values to some of the channels of a digital output module via SETIO_PART. You then read out the changes via GETIO_PART and via the digital input module.

Requirements

- An IO device/DP slave is created with a digital input module and a digital output module.

  Note

  Change with SETIO_PART, read out changes with GETIO_PART

  The digital input module must be connected to the digital output module in order that the written value can be read via SETIO_PART using GETIO_PART

- To store the data, create 14 tags and the two structures "readinputs" and "writeOutputs" in a global data block.
  Use the tag "id_Read" to store the hardware identifier of the digital input module. Use the tag "id_Write" to store the hardware identifier of the digital output module.

  Note

  Find hardware identifier

  Use "PLC tags > System constants" to find the hardware identifier. For example, for the digital input module, search for "Local~<Name of the DI module>", with the data type "HW_Submodule". The associated cell "value" contains the hardware identifier.

SLI_gDB_GETIO_SETIO_PART

	Name	Data type	Start value
1	Static		
2	id_Read	HW_SUBMODULE	257
3	executeRead	Bool	false
4	firstByteRead	Int	0
5	maxLengthRead	Int	2
6	readInputs	Struct	
7	Channel_0-7	Array[0..7] of Bool	
8	Channel_8-15	Array[0..7] of Bool	
9	Channel_16-23	Array[0..7] of Bool	
10	Channel_24-31	Array[0..7] of Bool	
11	errorRead	Bool	false
12	statusRead	DWord	16#0
13	doneRead	Bool	false
14	id_Write	HW_SUBMODULE	258
15	executeWrite	Bool	false
16	firstByteWrite	Int	0
17	maxLengthWrite	Int	1
18	writeOutputs	Struct	
19	Channel_0-7	Array[0..7] of Bool	
20	Channel_8-15	Array[0..7] of Bool	
21	Channel_16-23	Array[0..7] of Bool	
22	Channel_24-31	Array[0..7] of Bool	
23	errorWrite	Bool	false
24	statusWrite	DWord	16#0
25	doneWrite	Bool	false

- To change the channel states, set the following bits (for BYTE 0 and BYTE 1) to "TRUE" in the global data block:

14	writeOutputs	Struct	
15	Channel_0-7	Array[0..7] of Bool	
16	Channel_0-7[0]	Bool	false
17	Channel_0-7[1]	Bool	1
18	Channel_0-7[2]	Bool	1
19	Channel_0-7[3]	Bool	false
20	Channel_0-7[4]	Bool	false
21	Channel_0-7[5]	Bool	false
22	Channel_0-7[6]	Bool	false
23	Channel_0-7[7]	Bool	false
24	Channel_8-15	Array[0..7] of Bool	
25	Channel_8-15[0]	Bool	false
26	Channel_8-15[1]	Bool	1

- To forward the data, create the following local tags in an "SLI_FB_GETIO_SETIO_PART" FB.

7	Static			
8	GETIO_PART_Instance	GETIO_PART		
9	SETIO_PART_Instance	SETIO_PART		
10	readBYTE	Struct		Non-retain
11	writeBYTE	Struct		Non-retain

Interconnect parameters: "SLI_itemiseByte_gpspIO" FC

To transfer the values of a bit sequence (0..7) create the FC "SLI_itemiseByte_gpspIO".

Create the following local tags in it.

SLI_itemiseByte_gpspIO		
	Name	Data type
1	Input	
2	in0	Bool
3	in1	Bool
4	in2	Bool
5	in3	Bool
6	in4	Bool
7	in5	Bool
8	in6	Bool
9	in7	Bool
10	Output	
11	out0	Bool
12	out1	Bool
13	out2	Bool
14	out3	Bool
15	out4	Bool
16	out5	Bool
17	out6	Bool
18	out7	Bool

Network 1: According to the example of network 1, create interconnections for all bits (0..7).

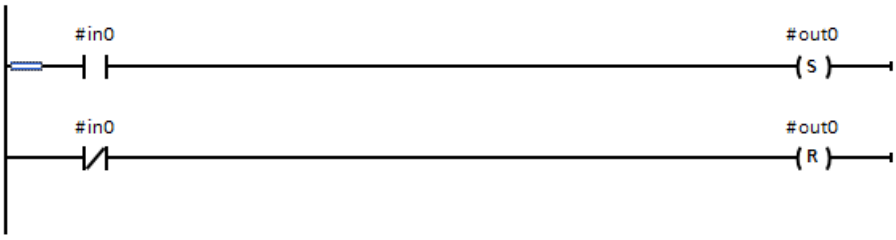

Interconnect parameters: "SLI_FB_GETIO_SETIO_PART" FB

Create the following interconnections in the "SLI_FB_GETIO_SETIO_PART" FB. On the IO device/DP slave, then call the FB "SLI_FB_GETIO_SETIO_PART" in a program cycle OB (OB1).

Network 1: Interconnect the parameters of the "GETIO_PART" instruction as follows.

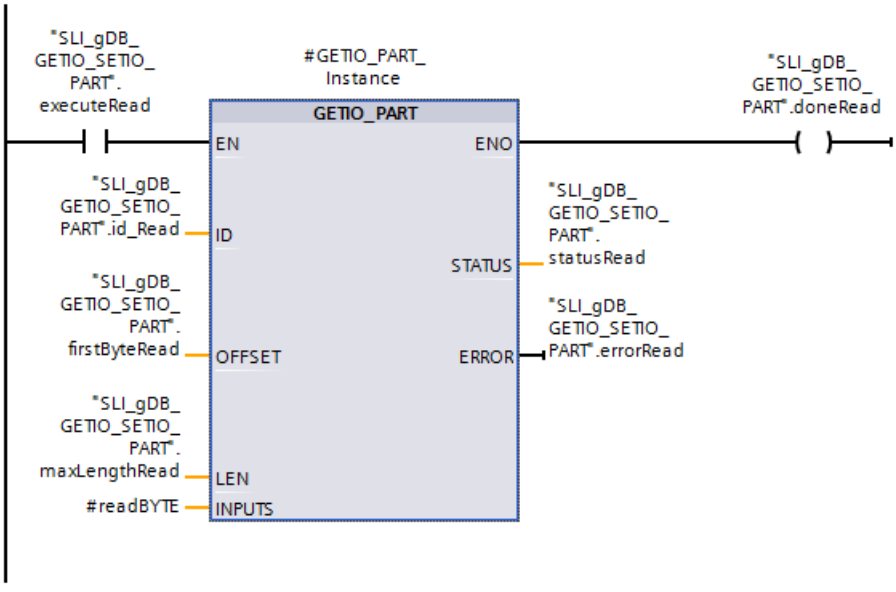

Network 2: Close the instruction "GETIO_PART" as follows.

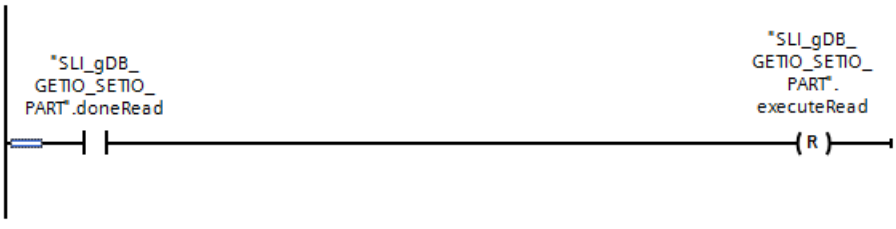

Network 3: Create the interconnections according to the following example to convert the channel values of GETIO_PART. All inputs and outputs (0..7) of the "SLI_itemiseByte_gpspIO" FC are interconnected in this case.

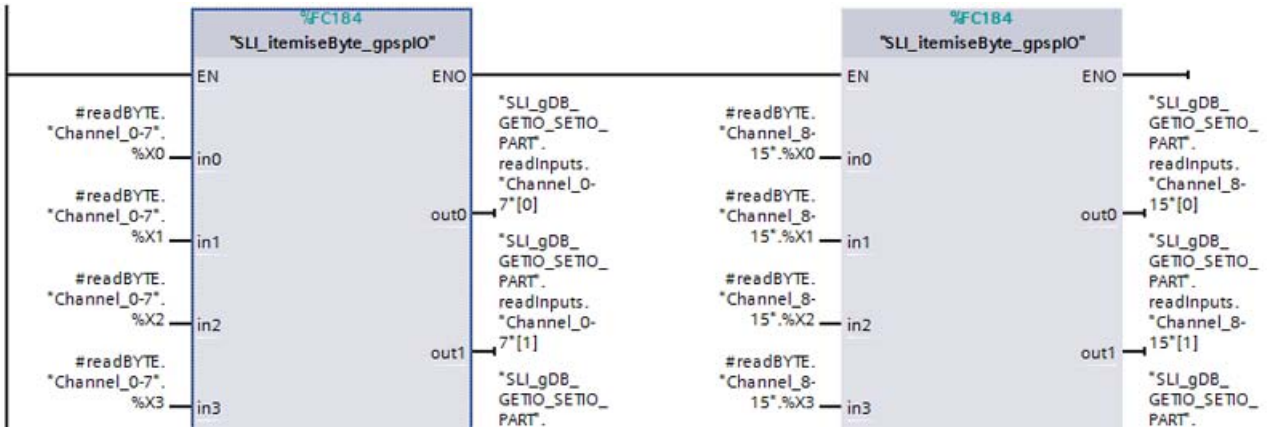

Network 4: Create the interconnections according to the following example to convert the channel values of SETIO_PART. All inputs and outputs (0..7) of the "SLI_itemiseByte_gpspIO" FC are interconnected in this case.

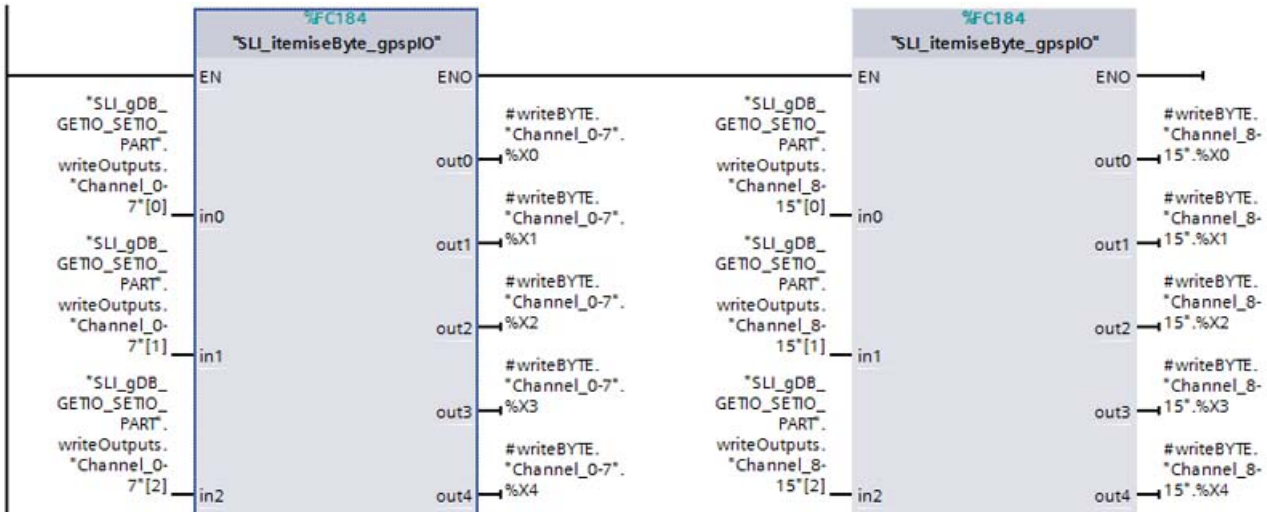

Network 5: Interconnect the parameters of the "SETIO_PART" instruction as follows.

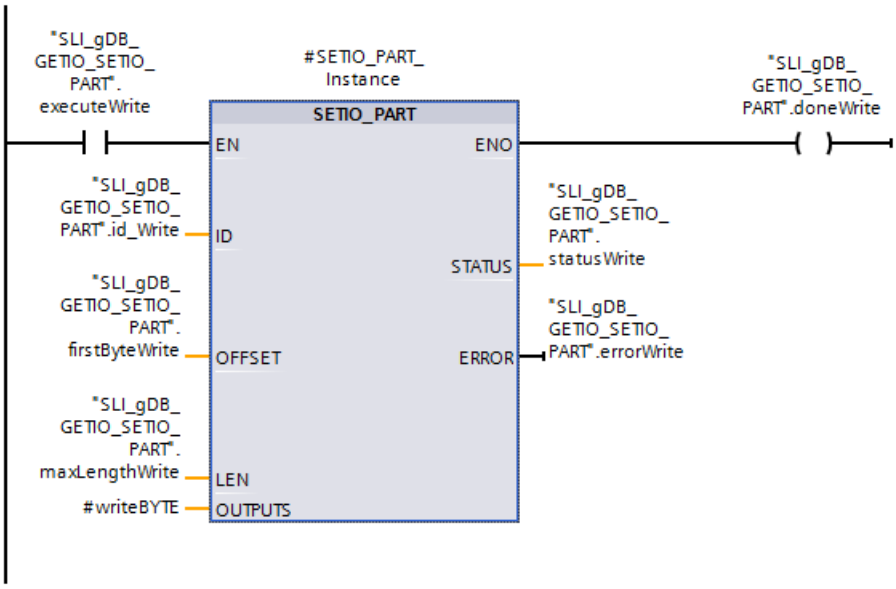

Network 6: Close the instruction "SETIO_PART" as follows.

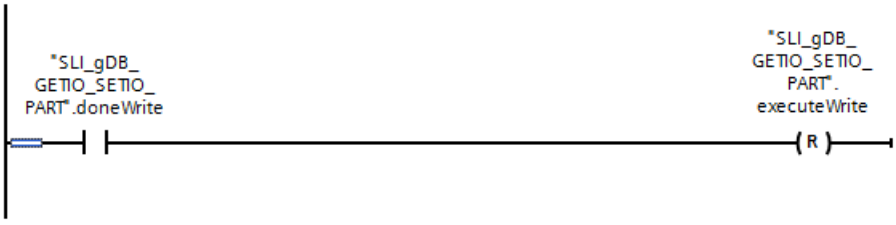

Result of network 1 (GETIO_PART)

When the input parameter EN ("executeRead") returns the signal state "TRUE", the "GETIO_PART" instruction is started. The instruction "GETIO_PART" calls the input module via the input parameter ID ("id_Read"). The channels of the input module are read out via the input parameter INPUTS ("#readBYTE").

Through the input parameter OFFSET ("firstByteRead") with the value "0", the instruction "GETIO_PART" starts to read channels of the input module starting from the "BYTE 0". Two BYTE (16 channels) are read out according to the value "2" of the input parameter LEN ("maxLengthRead"). In this example, this value corresponds to the channels 0 to 15.

The block status is displayed at the output parameter STATUS ("statusRead") and ERROR ("errorRead"). Processing takes place without errors.

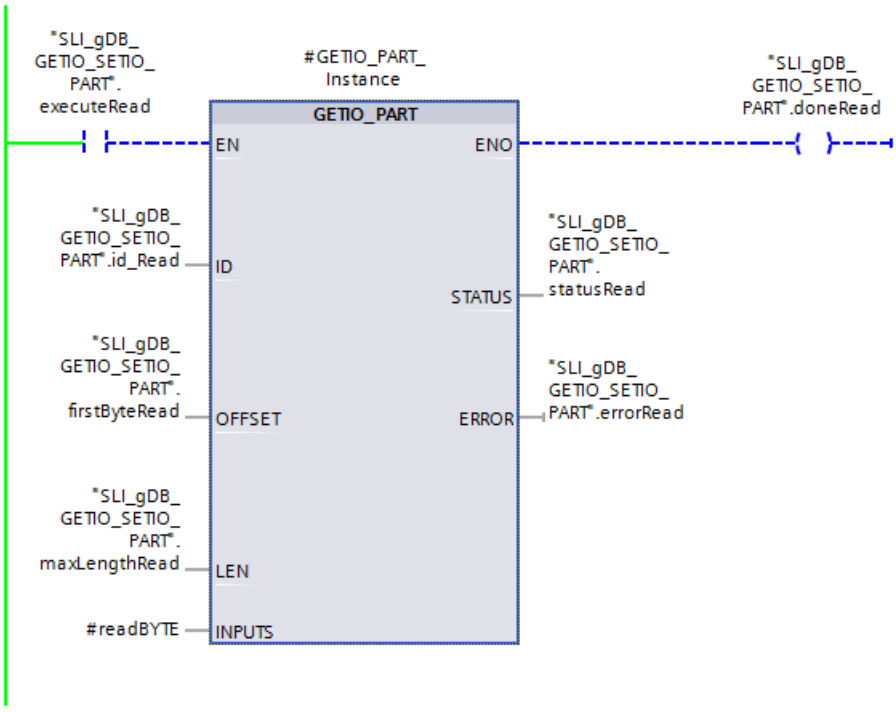

Result of network 2 (GETIO_PART)

When the instruction "GETIO_PART" has been executed, the output parameter ENO ("doneRead") returns the signal state "TRUE". After this, the resetter sets the tag "executeRead" to "FALSE" and ends the processing of "GETIO_PART".

SLI_gDB_GETIO_SETIO_PART

	Name	Data type	Start value	Monitor value
1	Static			
2	id_Read	HW_SUBMODULE	257	16#0101
3	executeRead	Bool	false	FALSE
4	firstByteRead	Int	0	0
5	maxLengthRead	Int	2	2
6	readInputs	Struct		
7	errorRead	Bool	false	FALSE
8	statusRead	DWord	16#0	16#0000_0000
9	doneRead	Bool	false	FALSE

Result of network 3 (GETIO_PART)

Using MOVE instructions and the FC SLI_itemiseByte_gpspIO, the structure "#readBYTE" transfers the values of the tags to the tags of the structure "readInputs". The values of the channels 0 to 7 and 8 to 15 are hereby converted from BYTE to array of BOOL.

To improve understanding, the tags of the "readInputs" structure are represented by descriptive names.

Result of network 4 (SETIO_PART)

Using MOVE instructions and the FC SLI_itemiseByte_gpspIO, the structure "writeOutputs" transfers the values of the tags to the tags of the structure "#writeBYTE". The values of the channels 0 to 7 and 8 to 15 are converted from array of BOOL to BYTE.

To improve understanding, the tags of the "writeOutputs" structure are represented by descriptive names.

Result of network 5 (SETIO_PART)

When the input parameter EN ("executeWrite") returns the signal state "TRUE", the "SETIO_PART" instruction is started. The instruction "SETIO_PART" calls the output module via the input parameter ID ("id_Write"). New values are written to the channels of the output module via the input parameter OUTPUTS ("#writeBYTE").

Through the input parameter OFFSET ("firstByteWrite") with the value "0", the instruction "SETIO_PART" starts to write to channels of the output module starting from the "BYTE 0". Only one BYTE (8 channels) is written according to the value "1" of the input parameter LEN ("maxLengthWrite"). In this example, this value corresponds to the channels 0 to 7. The channel 9 set to the value "TRUE" (see "writeOutputs", "Channel_8-15[1]") is thus not taken into consideration during writing to the channels of the output module.

The block status is displayed at the output parameter STATUS ("statusWrite") and ERROR ("errorWrite"). Processing takes place without errors.

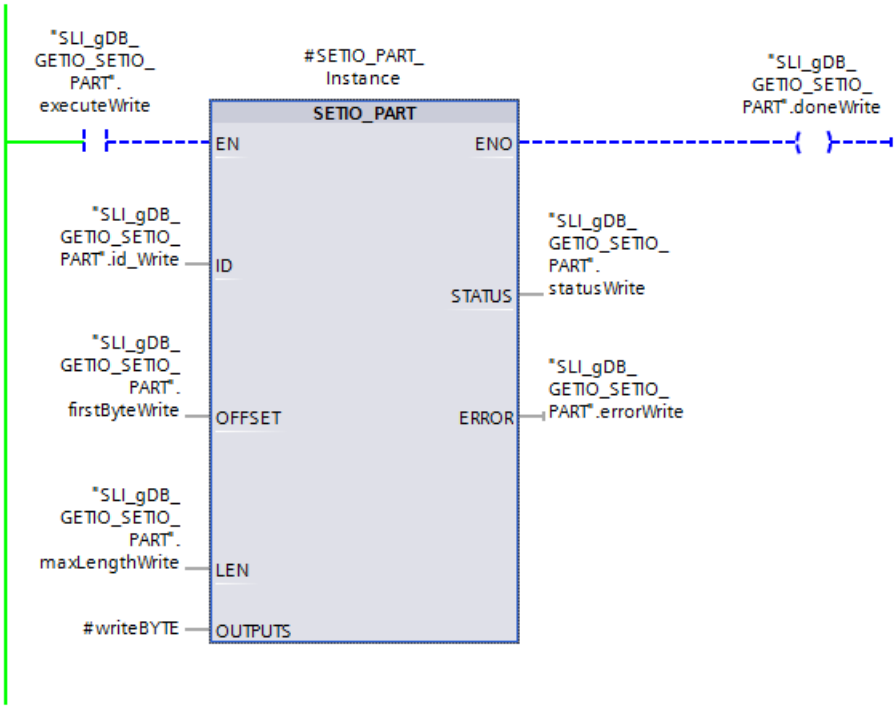

Result of network 6 (SETIO_PART)

When the instruction "SETIO_PART" has been executed, the output parameter ENO ("doneWrite") returns the signal state "TRUE". After this, the resetter sets the tag "executeWrite" to "FALSE" and ends the processing of "SETIO_PART".

10	id_Write	HW_SUBMODULE	258	16#0102
11	executeWrite	Bool	false	FALSE
12	firstByteWrite	Int	0	0
13	maxLengthWrite	Int	1	1
14	writeOutputs	Struct		
15	errorWrite	Bool	false	FALSE
16	statusWrite	DWord	16#0	16#0000_0000
17	doneWrite	Bool	false	FALSE

Result of reprocessing of GETIO_PART

If the tag "executeRead" is reset to "TRUE", the structure "readInputs" shows the new values on the channels of the input module after the execution of GETIO_PART. A change affects only the channels 0 to 7 according to the values of the tags "firstByteWrite" and "maxLengthWrite". The channels 1 and 2 ("Channel_0-7[1]..[2]") now return the signal state "TRUE". The signal state of channel 9 ("Channel_8-15[1]") is unchanged.

6	readInputs	Struct		
7	Channel_0-7	Array[0..7] of Bool		
8	Channel_0-7[0]	Bool	false	FALSE
9	Channel_0-7[1]	Bool	false	TRUE
10	Channel_0-7[2]	Bool	false	TRUE
11	Channel_0-7[3]	Bool	false	FALSE
12	Channel_0-7[4]	Bool	false	FALSE
13	Channel_0-7[5]	Bool	false	FALSE
14	Channel_0-7[6]	Bool	false	FALSE
15	Channel_0-7[7]	Bool	false	FALSE
16	Channel_8-15	Array[0..7] of Bool		
17	Channel_8-15[0]	Bool	false	FALSE
18	Channel_8-15[1]	Bool	false	FALSE

Note

Changing the signal state of a channel

If a value is already assigned to a channel on which a signal state is to be changed (for example by a PLC tag), this pre-assignment can make a change by SETIO_PART invalid. Make sure that the signal state of a channel to be changed is not already determined by other factors.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

GETIO_PART: Read process image area (Page 2261)

SETIO_PART: Transfer process image area (Page 2263)

RALRM: Receive interrupt

Description RALRM (S7-1200, S7-1500)

Description

The instruction receives an interrupt with all corresponding information from an I/O module (centralized configuration) or from a module of a DP slave or PROFINET IO device. This information is made available at its output parameters.

The information in the output parameters contains the start information of the called OB as well as information of the interrupt source.

The configuration of the data structure of the target range AINFO corresponds to that for a central configuration of the PROFINET IO data structure.

Call "RALRM" only within the interrupt OB started by the CPU operating system as a result of the I/O interrupt that is to be examined.

Note

If you call "RALRM" in an OB whose start event is not an I/O interrupt, the instruction will provide correspondingly reduced information in its outputs.
Make sure to use different instance DBs when you call "RALRM" in different OBs. If you evaluate data resulting from an "RALRM" call outside of the associated interrupt OB, use a separate instance DB per OB start event.

Note

The interface of the "RALRM" instruction is identical to that in the "RALRM" FB defined in the standard "PROFIBUS and PROFINET Guideline Communication Function Blocks on PROFIBUS DP and PROFINET IO".

Calling RALRM

"RALRM" can be called in three operating modes (MODE parameter). These are explained in the table below.

MODE	RALRM ...
0	... shows the component that triggered the interrupt in the output parameter ID and sets the output parameter NEW to TRUE.
1	... writes all output parameters, regardless of the component triggering the interrupt.
2	... checks whether the component specified in the F_ID input parameter has triggered the interrupt. • If not, NEW = FALSE • If yes, NEW = TRUE and all other output parameters are written.

Parameters

The following table shows the parameters of the "RALRM" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module. The number is assigned automatically and is stored in the hardware configuration properties of the component or interface.
MLEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the interrupt information to be received in bytes. With MLEN = 0, all data specified at the AINFO parameter are read.
NEW	Output	BOOL	I, Q, M, D, L	A new interrupt was received.
STATUS (Page 2276)	Output	DWORD	I, Q, M, D, L	Error code
ID	Output	HW_IO	I, Q, M, D, L	Hardware identifier of the module from which the interrupt was received.
LEN	Output	UINT	I, Q, M, D, L	Length of the received interrupt information
TINFO (Page 2279)	InOut	VARIANT	D, L	Target range for OB start and management information
AINFO (Page 2297)	InOut	VARIANT	I, Q, M, D, L	Target range for header information and additional interrupt information For AINFO , provide a length of at least MLEN bytes.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

If you select a target range (Page 2304)TINFO or AINFO that is too short, RALRM does not enter the full information.

Special case for PC systems: Determining Windows status

The following scenario will help you determine the Windows status of your PC system:

- Call the instruction "RALRM" (SFB54) in OB82.

The "RALRM" instruction is called with the following input parameters:

Parameter	Declaration	Data type	Memory area	Value/Description
MODE	Input	INT	I, Q, M, D, L or constant	2
F_ID	Input	HW_IO	I, Q, M, D, L or constant	59 (Host Interface RID)
MLEN	UINT	UINT	I, Q, M, D, L or constant	0
TINFO	InOut	VARIANT	D, L	Address for each area of 54 bytes
AINFO	InOut	VARIANT	I, Q, M, D, L	Address for each area of 54 bytes

In the case of a software controller, you can also use RALRM to receive information on the status of Windows.
Use bytes 50 and 51 from the parameter "AINFO" to receive detailed information on the status of Windows. If byte 50 shows the value W#16#40m, you can read out the following detailed information on the window status in byte 51:

- 0xA1: Bluescreen
- 0xA2: Windows running
- 0xA3: Windows not running

Note

In case of a system crash, Windows no longer responds and shows a blue screen which indicates the reason for and the time of the system crash. By default, the blue screen remains visible permanently.

The system is stopped after the system error and you can encounter the following scenario:

- Windows no longer sends signals to the configured software CPU

Example

You can find the example here: Program example for RALRM (Page 2305).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

Parameter STATUS (S7-1200, S7-1500)

Description

The STATUS output parameter contains error information. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	• B#16#00, if no error • Function ID from DPV1-PDU: In the event of an error, B#16#80 is output (in the event of an error reading a data record B#16#DE and writing a data record B#16#DF). If no DPV1 protocol element is used, then B#16#C0 will be output.
STATUS[2]	Error_Decode	Location of the error ID
STATUS[3]	Error_Code_1	Error ID
STATUS[4]	Error_Code_2	Manufacturer-specific extension of the error ID

Field element STATUS[2]

STATUS[2] can have the following values:

Error_Decode (B#16#...)	Source	Meaning
00 to 7F	CPU	No error or no warning
80	DPV1	Error according to IEC 61158-6
81 to 8F	CPU	B#16#8x shows an error in the xth call parameter of the instruction.
FE, FF	DP profile	Profile-specific error

Field element STATUS[3]

STATUS[3] can have the following values:

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
00	00		No error, no warning
70	00	reserved, reject	Initial call; no active data record transfer
	01	reserved, reject	Initial call; data record transfer has started
	02	reserved, reject	Intermediate call; data record transfer already active
80	81		The system data type at parameter TINFO does not match the call environment of the instruction. The used system data type must match the organization block in the user program (example: for a time-delay interrupt OB you need the system data type TI_Delay).

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	90	reserved, pass	• Invalid logical start address • RDREC And WRREC instructions with an S7-1500-R/H CPU: No access to the data record because the partner CPU is in the STOP operating mode.
	92	reserved, pass	Illegal type for VARIANT pointer
	93	reserved, pass	The DP component addressed via ID or F_ID is not configured.
	96		The "RALRM (Page 2273)" does not supply the OB start information, management information, header information, or additional interrupt information. For OBs 4x, 55, 56, 57, 82 and 83, you use the "DPNRM_DG (Page 2353)" instruction to read the current diagnostic frame of the relevant DP slave asynchronously (address information from OB start information).
	A0	read error	Negative acknowledgment when reading the module
	A1	write error	Negative acknowledgment when writing to the module
	A2	module failure	DP protocol error at layer 2 (e.g., slave failure or bus problems)
	A3	reserved, pass	General communication error or IO device / DP slave is unreachable
	A4	reserved, pass	Communication on PBUS+ disrupted
	A5	reserved, pass	–
	A7	reserved, pass	DP slave or module is occupied (temporary error
	A8	version conflict	DP slave or module reports non-compatible versions
	A9	feature not supported	Function is not supported by DP slave or module
	AA to AF	user specific	DP slave or module reports a manufacturer-specific error in its application. Please check the documentation from the manufacturer of the DP slave or module.
	B0	invalid index	Data record not known in module Illegal data record number ≥ 256
	B1	write length error	Error at length specification: • With "RALRM (Page 2273)": length error in AINFO (Page 2297) • With "RDREC (Page 2213)": length error in MLEN • With "WRREC (Page 2232)": length error in LEN
	B2	invalid slot	• The configured slot is not occupied. • For PROFINET IO and PROFIBUS DP: IO device / DP slave is unreachable
	B3	type conflict	Actual module type does not match specified module type
	B4	invalid area	DP slave or module reports access to an invalid area

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	B5	state conflict	DP slave or module not ready
	B6	access denied	DP slave or module denies access
	B7	invalid range	DP slave or module reports an invalid range for a parameter or value
	B8	invalid parameter	DP slave or module reports an invalid parameter
	B9	invalid type	DP slave or module reports an invalid type With "RDREC (Page 2213)": buffer too small (subsets cannot be read) With "WRREC (Page 2232)": buffer too small (subsets cannot be written)
	BA		S7-1500-R/H-CPU: The primary CPU fails while reading or writing a data record. The new primary CPU acknowledges the request with 0x80BA. The data record request is not repeated on the new primary CPU.
	BB to BF	user specific	DP slave or module reports a manufacturer-specific error when accessing. Please check the documentation from the manufacturer of the DP slave or module.
	C0	read constrain conflict	With "WRREC (Page 2232)": The data is only written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You only write the data online with PG/PC. With "RDREC (Page 2213)": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. Therefore, it is only possible to read the data online with PG/PC.
	C1	write constrain conflict	The data of the previous write job on the module for the same data record has not yet been processed by the module.
	C2	resource busy	The module is currently processing the maximum possible number of jobs for a CPU.
	C3	resource unavailable	The required operating resources are currently occupied.
	C4		Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.
	C5		DP slave or module not available.
	C6		Data record transfer was canceled due to priority class cancellation
	C7		Job aborted due to warm or cold restart on the DP master
	C8 to CF		DP slave or module reports a manufacturer-specific resource error. Please check the documentation from the manufacturer of the DP slave or module.
	Dx	user specific	DP slave specific. Refer to the description of the DP slave.
	Ex	user specific	Manufacturer-specific. Please check the documentation of the module.

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
81	00 to FF		Error in the initial call parameter (with "RALRM (Page 2273)": MODE
	00		Illegal operating mode
82	00 to FF		Error in the second call parameter
:	:		:
88	00 to FF		Error in the eighth call parameter (with "RALRM (Page 2273)": TINFO (Page 2279))
	01		Wrong syntax ID
	23		Configuration limits exceeded or destination area too small
	24		Wrong range ID
	32		DB/DI no. out of user range
	3A		DB/DI no. is NULL for area ID DB/DI or specified DB/DI does not exist
89	00 to FF		Error in the ninth call parameter (with "RALRM (Page 2273)": AINFO (Page 2297))
	01		Wrong syntax ID
	23		Configuration limits exceeded or destination area too small
	24		Wrong range ID
	32		DB/DI no. out of user range
	3A		DB/DI no. is NULL for area ID DB/DI or specified DB/DI does not exist
8A	00 to FF		Error in the 10th call parameter
:	:		:
8F	00 to FF		Error in the 15th call parameter
FE, FF	00 to FF		Profile-specific error

Parameter TINFO (S7-1200, S7-1500)

Data structure of the destination area TINFO

The data structure of the destination area TINFO contains the start information of the organization block in which "RALRM" is currently being called.

The TINFO destination area can contain the start information with standard access or optimized access. The format of the start information in the TINFO destination area must always match the start information of the corresponding organization block.

- The start information of an OB with standard access is always stored in the first 20 bytes of the "Temp" section in the block interface. Use the "TI_Classic" data structure for this.
- The start information of an OB with optimized access is always written to the "Input" section. The start information is stored in data structures specific to the organization block. Use the data structure specific to the organization block.

Changing the block access (standard/optimized) also changes the block interface.

The following table provides an overview of the data structures that are used depending on the organization block at the TINFO parameter.

Name of the data structure	S7-1200 CPUs as of version	S7-1500 CPUs as of version	Data structure used for:
Data structure for organization blocks with standard access			
TI_Classic	-	V1	Organization blocks without optimized block access.
Data structure for organization blocks with optimized block access			
TI_ProgramCycle	V2	V1	Cycle OB (program cycle)
TI_Startup	V2	V1	Startup OB (startup)
TI_Delay	V2	V1	Time delay interrupt
TI_Cyclic	V2	V1	Cyclic interrupt OB
TI_HWInterrupt	V2	V1	Hardware interrupt OB
TI_HWInterrupt_Extended	-	V2.0	Hardware interrupt OB
TI_TimeError	V2	V1	Time error OB
TI_DiagnosticInterrupt	V2	V1	Diagnostic error interrupt OB
TI_PlugPullModule	V2	V1	Pull/plug of modules OB
TI_StationFailure	V2	V1	Rack or station failure OB
TI_ProgIOAccessError	V2	V1	• Programming error OB • I/O access error OB
TI_TimeOfDay	V2	V1	Time-of-day interrupt OB
TI_SynchCycle	-	V1	Synchronous cycle interrupt OB
TI_Submodule	V2	V1	• Status interrupt OB • Update interrupt OB • OB for manufacturer or profile-specific interrupt
TI_CPURedundancyError_V2	-	V2.6	CPU redundancy error OB

Data structure for organization blocks with standard access

The following shows the TI_Classic data structure:

Parameter	Data type	Byte	Description
TI_Classic			Data structure for organization blocks without optimized block access.

Parameter	Data type	Byte	Description
Bytes 0 to 19: Start information of the OB in which "RALRM" is currently being called.*			
EV_CLASS	BYTE	0	Event class Example OB1: • Bits 0-3: ID of the event (1 = incoming event) • Bits 4-7: Event class (1 = Event class 1)
EV_NUM	BYTE	1	Event number (depending on OB type) Example OB1 (SCAN_1): • SCAN_1 = 1 for first call • SCAN_1 = 3 for all additional calls
PRIORITY	BYTE	2	Priority class
NUM	BYTE	3	OB number
TYP2_3	BYTE	4	Additional information Different information is stored in the BYTES "TYP2_3" and "TYPE 1" depending on the OB type used. What these are is described in the documentation for the organization blocks. Example (OB1): • TYP2_3: OB1_RESERVED_1 (reserved) • TYP1: OB1_RESERVED_2 (reserved)
TYP1	BYTE	5	
ZI1	WORD	6 to 7	Additional information Different information is stored in "ZI1" depending on the OB type used. What these are is described in the documentation for the organization blocks. Example (OB1): • ZI1: OB1_PREV_CYCLE (runtime of previous cycle in ms)
ZI2_3	DWORD	8 to 11	Additional information Different information is stored in "ZI2_3" depending on the OB type used. What these are is described in the documentation for the organization blocks. Example (OB1): • ZI2: OB1_MIN_CYCLE (minimum cycle time (ms) since the last startup) • ZI3: OB1_MAX_CYCLE (maximum cycle time (ms) since the last startup)
OB_DATE_TIME	DATE_AND_TIME (DT)	12 to 19	Date and time-of-day when the OB was called.
Bytes 20 and 21: Address information			
address	WORD	20 and 21	Address information like S7-300/400 CPUs: • In a central configuration, the rack number (0-31): 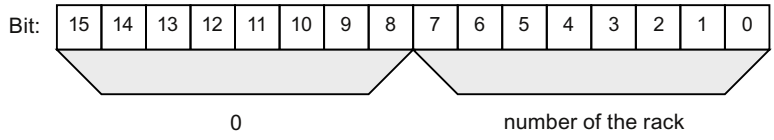 • For a distributed configuration with PROFIBUS DP: – DP master system ID (1-31) – Station number (0-127).

Parameter	Data type	Byte	Description
			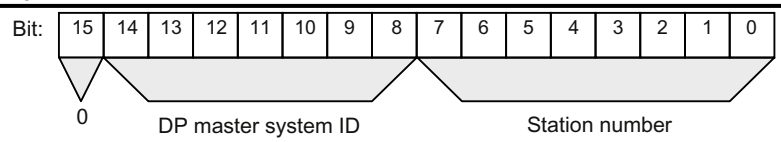 • In a distributed configuration with PROFINET IO: – The last two positions in the PROFINET IO system ID (0-15). To obtain the complete PROFINET IO system ID, you must add 100 (decimal) to it. – Station number (0-2047). 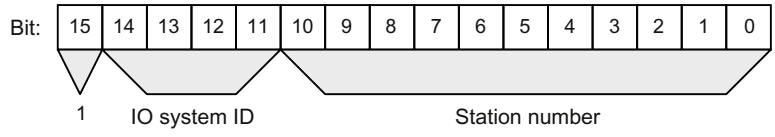
Bytes 22 to 31: Management information			
slv_prfl	BYTE	22	Slave profile like S7-300/400 CPUs: • In a central configuration: 0 (data record 0 or data record 1) • In a distributed configuration: – Bit 0 to 3: Slave type - 0000: DP (structure data record 0) - 0001: DPS7 (structure data record 0 or data record 1) - 0010: DPS7 V1 (structure data record 0 or data record 1) - 0011: DPV1 (structure according to PROFIBUS DP standard) - 0100 – 0111: Reserved - 1000: PROFINET IO (structure according to PROFINET IO standard) - from 1001: Reserved – Bit 4 to 7: Profile type (reserved)
intr_type	BYTE	23	Interrupt info type like S7-300/400 CPUs: • In a central configuration: 0 • In a distributed configuration: – Bit 0 to 3: Interrupt info type - 0000: Interrupt comes from a configured remote module - 0001: Interrupt comes from a non-DPV1 slav / non IO device or a slot that is not configured - 0010: Interrupt generated in the CPU - from 0011: Reserved – Bit 4 to 7: Structure version - 0000: Initial - from 0001: Reserved
flags1	BYTE	24	Flags of the PROFIBUS DP master interface module / PROFINET IO controller interface module like S7-300/400 CPUs: • In a central configuration: 0 • In a distributed configuration: – Bit 0 = 0: Interrupt originating from an integrated interface module (PROFINET IO or PROFIBUS DP)

Parameter		Data type	Byte	Description
				– Bit 0 = 1: Interrupt originating from an external interface module (PROFINET IO or PROFIBUS DP) – Bit 1 to 7: Reserved
	flags2	BYTE	25	Flags of the PROFIBUS DP master interface module / PROFINET IO controller interface module like S7-300/400 CPUs: • In a central configuration: 0 • For a distributed configuration with PROFIBUS DP: – Bit 0: EXT_DIAG_FLAG from the diagnostics message frame, or 0 if this bit does not exist in the interrupt The bit is 1 if the DP slave is faulty. – Bit 1 to 7: Reserved • In a distributed configuration with PROFINET IO: – Bit 0: ARDiagnosisState or 0 if there is no information in the interrupt. The bit is 1 if the IO device is faulty. – Bit 1 to 7: Reserved
	id	UINT	26 and 27	Management information • In a central configuration: 0 • For a distributed configuration with PROFIBUS DP: PROFIBUS ID number as unique identifier of the PROFIBUS DP slave • In a distributed configuration with PROFINET IO: PROFINET IO device ID number as unique identifier of the PROFINET IO device
	manufactur-er	UINT	28 and 29	Manufacturer ID (only in a distributed configuration with PROFINET IO).
	instance	UINT	30 and 31	Ident number of the instance (only in a distributed configuration with PROFINET IO).
*The start information depends on the OB used. You can find the start information of each OB type at the interface or in the documentation of the OB.				

Data structures for organization blocks with optimized block access

The data structures for organization blocks with optimized block access have the following format:

- Bytes 0 to 3: Format of the start information, class and number of the called OB (same structure for all data structures).
- Bytes 4 to 19: Optimized start information (structure depends on the OB type). The data in bytes 4 to 19 correspond to the structure and content of the corresponding OB interface.
- Bytes 20 to 31: In addition, the address and management information for certain OBs. The data in bytes 20 to 31 correspond to the data of 20 to 31 bytes of the TI_Classic data structure.

The format of the data structures is described in the following tables.

Table 4-15 **Cycle OB: Data structure TI_ProgramCycle**

Parameter		Data type	Byte	Description
TI_ProgramCycle				Data structure for cycle OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=1)
	OB_Nr	UINT	2	OB number (1...32767).
	Initial_Call	BOOL	4	= TRUE in the first call of this OB: • Transition from STOP or HOLD to RUN • After reloading
	Remanence	BOOL	5	= TRUE, if retentive data are available.

Table 4-16 **Startup OB: Data structure TI_Startup**

Parameter		Data type	Byte	Description
TI_Startup				Data structure for startup OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=100)
	OB_Nr	UINT	2	OB number (1...32767).
	LostRetentive	BOOL	4	=TRUE, if the contents of retentive data areas have been lost.
	LostRTC	BOOL	5	=TRUE, if the time-of-day of the real-time clock has been lost.

Table 4-17 **Time delay interrupt OB: Data structure TI_Delay**

Parameter		Data type	Byte	Description
TI_Delay				Data structure for time delay interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=20)
	OB_Nr	UINT	2	OB number (1...32767).
	Sign	WORD	4	User ID: Input parameter SIGN from the call of the "SRT_DINT (Page 2469)" instruction.

Table 4-18 **Cyclic interrupt OB: Data structure TI_Cyclic**

Parameter		Data type	Byte	Description
TI_Cyclic				Data structure for cyclic interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=30)
	OB_Nr	UINT	2	OB number (1...32767).
	Initial_Call	BOOL	4	= TRUE in the first call of this OB • At the transition from STOP or HOLD to RUN • After reloading
	Event_Count	INT	6	Number of discarded start events since the last start of this OB.

Table 4-19 **Hardware interrupt OB: Data structure TI_HWInterrupt**

Parameter		Data type	Byte	Description
TI_HWInterrupt				Data structure for hardware interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=40)
	OB_Nr	UINT	2	OB number (1...32767).
	LADDR	HW_IO	4	Hardware identifier of the module that triggers the hardware interrupt.
	USI	WORD	6	Identifier for future extensions (not user-relevant).
	IChannel	USINT	8	Number of the channel that triggered the hardware interrupt.
	EventType	BYTE	9	Identifier for the event type associated with the event triggering the interrupt (e.g., rising edge). You can find this ID in the description of the respective module.
	address	WORD	20	See "address" parameter for the TI_Cassic. data structure
	slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
	intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
	flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
	flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
	id	UNIT	26	See "id" parameter for the TI_Cassic data structure.
	manufacturer	UNIT	28	See "manufacturer" parameter for the TI_Cassic data structure.
	instance	UNIT	30	See "instance" parameter for the TI_Cassic data structure.

Table 4-20 **Hardware interrupt OB: Data structure TI_HWInterrupt_Extended**

Parameter		Data type	Byte	Description
TI_HWInterrupt_Extended				Data structure for hardware interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=40)
	OB_Nr	UINT	2	OB number (1...32767).
	LADDR	HW_IO	4	Hardware identifier of the module that triggers the hardware interrupt.
	USI	WORD	6	Identifier for future extensions (not user-relevant).
	IChannel	USINT	8	Number of the channel that triggered the hardware interrupt.
	EventType	BYTE	9	Identifier for the event type associated with the event triggering the interrupt (e.g., rising edge). You can find this ID in the description of the respective module.
	PointAddr	DWORD	12	• In the case of digital modules: Bit array containing the inputs on the module that triggered the hardware interrupt • In the case of analog modules: Bit array with the information as to which channel has exceeded which limit • In the case of CPs or IMs: Interrupt status of the module (not user-relevant)
	address	WORD	20	See "address" parameter for the TI_Cassic. data structure
	slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
	intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
	flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
	flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
	id	UNIT	26	See "id" parameter for the TI_Cassic data structure.
	manufacturer	UNIT	28	See "manufacturer" parameter for the TI_Cassic data structure.
	instance	UNIT	30	See "instance" parameter for the TI_Cassic data structure.

Table 4-21 **Time error OB: Data structure TI_TimeError**

Parameter		Data type	Byte	Description
TI_TimeError				Data structure for time error OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=80)
	OB_Nr	UINT	2	OB number (1...32767).
	Csg_OBnr	OB_ANY	4	Number of the OB that was being executed when the time error occurred.
	Fault_ID	BYTE	7	Error code. Possible values: • B#16#01: Cycle time exceeded • B#16#02: The requested OB is still being processed. • B#16#05: Expired time-of-day interrupt due to time jump. • B#16#06: Expired time-of-day interrupt upon re-entry into RUN after HOLD. • B#16#07: Overflow of the OB request buffer for the current priority class. • B#16#08: Isochronous mode interrupt - time error. • B#16#09: Interrupt loss due to high interrupt load • B#16#0B: Technology synchronization interrupt - time error
	Csg_Prio	UNIT	8	Priority of the OB that was being executed when the time error occurred.

Table 4-22 **Diagnostic interrupt OB: Data structure TI_DiagnosticInterrupt**

Parameter	Data type	Byte	Description
TI_DiagnosticInterrupt			Data structure for diagnostic interrupt OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=82)
OB_Nr	UINT	2	OB number (1...32767).
LADDR	HW_ANY	4	Hardware identifier of the hardware object that triggered the diagnostic interrupt.
IO_State	WORD	6	Status of the hardware object: • Bit 0: Good • Bit 1: Disabled • Bit 2: Maintenance required • Bit 3: Maintenance demanded • Bit 4: Error • Bit 5: Not accessible • Bit 6: Qualified • Bit 7: Not available
Channel	UINT	8	Channel number
MultiError	BOOL	10	=TRUE, if there is more than one error.
address	WORD	20	See "address" parameter for the TI_Cassic. data structure
slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
id	UINT	26	See "id" parameter for the TI_Cassic data structure.
manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Table 4-23 **Pull/plug interrupt OB: Data structure** TI_PlugPullModule

Parameter	Data type	Byte	Description
TI_PlugPullModule			Data structure for pull/plug interrupt OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=83)
OB_Nr	UINT	2	OB number (1...32767).
LADDR	HW_IO	4	Hardware identifier of the affected module or submodule
Event_Class	BYTE	6	• B#16#38: (Sub)module plugged • B#16#39: (Sub)module pulled or not responding
Fault_ID	BYTE	7	Error code The following table shows the events that cause the start of the pull/plug interrupt OB. • With Event_Class = B#16#38 - module/submodule plugged: – B#16#54: Submodule plugged and matches configured submodule – B#16#55: Submodule plugged, but does not match configured submodule – B#16#56: Submodule plugged, but error in module parameter assignment – B#16#57: Submodule or module plugged, but with a fault or maintenance – B#16#58: Submodule access error corrected – B#16#61: Module inserted • With Event_Class = B#16#39 - module/submodule pulled or not responding: – B#16#51: Module pulled – B#16#54: Submodule pulled – B#16#61: Module pulled or not responding
address	WORD	20	See "address" parameter for the TI_Cassic. data structure
slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
id	UINT	26	See "id" parameter for the TI_Cassic data structure.
manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Table 4-24 **Rack error OB: Data structure TI_StationFailure**

Parameter	Data type	Byte	Description
TI_StationFailure			Data structure for rack error OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=86)
OB_Nr	UINT	2	OB number (1...32767).
LADDR	HW_Device	4	Hardware identifier of the defective hardware object.
Event_Class	BYTE	6	• B#16#32: Activation of a station with the "D_ACT_DP" instruction • B#16#33: Deactivation of a station with the "D_ACT_DP" instruction • B#16#38: Outgoing event • B#16#39: Incoming event
Fault_ID	BYTE	7	Error code The error code is used to output the event that caused the start of the rack error OB. • Event_Class = B#16#39, FAULT_ID = C3: Distributed I/O: Failure of a DP master system • Event_Class = B#16#38, FAULT_ID = C4: Failure of a DP station • Event_Class = B#16#39, FAULT_ID = CA: PROFINET IO system failure • Event_Class = B#16#39/38, FAULT_ID = CB: PROFINET IO station failure/ station return • Event_Class = B#16#38, FAULT_ID = CC: PROFINET IO station return with problem or maintenance • Event_Class = B#16#38, FAULT_ID = CD: PROFINET IO station return, expected configuration different from actual configuration • Event_Class = B#16#38, FAULT_ID = CE: PROFINET IO station return, error in module parameter assignment • Event_Class = B#16#32/33, FAULT_ID = CF: Activation/deactivation of a PROFINET IO device with the "D_ACT_DP" instruction • Event_Class = B#16#39/38, FAULT_ID = F8: Failure/return of some of the submodules of a PROFINET I-device • Event_Class = B#16#38, FAULT_ID = F9: Return of some of the submodules of a PROFINET I-device with a device configuration difference
address	WORD	20	See "address" parameter for the TI_Cassic. data structure
slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
id	UINT	26	See "id" parameter for the TI_Cassic data structure.
manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Table 4-25 **Programming error OB / I/O access error OB: Data structure TI_ProgIOAccessError**

Parameter	Data type	Byte	Description*
TI_ProgIOAccessError			Data structure for programming error OB and I/O access error OB

Parameter	Data type	Byte	Description*
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class: • =120 for programming error OB • =120 for I/O access error OB
OB_Nr	UINT	2	OB number (1...32767).
BlockNr	UINT	4	Number of the block in which the programming error occurred.
Reaction	USINT	6	• 0: Ignore error • 1: Substitute bad value • 2: Skip command
Fault_ID	BYTE	7	Error code • B#16#21: BCD conversion error • B#16#22: Area length error when reading • B#16#23: Area length error when writing • B#16#24: Area error when reading • B#16#25: Area error when writing • B#16#26: Error in timer no. • B#16#27: Error in counter no. • B#16#28: Read access to a byte, word or double word with a pointer, the bit address of which is not 0 • B#16#29: Write access to a byte, word or double word with a pointer, the bit address of which is not 0 • B#16#:30: Write access to a write-protected global DB • B#16#31: Write access to a write-protected instance DB • B#16#32: DB number error when accessing a global DB • B#16#33: DB number error when accessing an instance DB • B#16#34: Number error during FC call • B#16#35: Number error during FB call • B#16#42: I/O access error, reading • B#16#43: I/O access error, writing • B#16#3A: Access to a DB that is not loaded; the DB number is located in the permitted area • B#16#3C: Access to an FC that is not loaded; the FC number is within the permissible range • B#16#3D: Access to an instruction (SFC) that is not loaded; the SFC number is within the permissible range • B#16#3E: Access to an FB that is not loaded; the FB number is within the permissible range • B#16#3F: Access to an SFB that is not available; the SFB number is within the permissible range
BlockType	USINT	8	Type of block in which the error has occurred:

Parameter		Data type	Byte	Description*
				• 1: OB • 2: FC • 3: FB • 4: SFC • 5: SFB • 6: DB
	Area	USINT	9	Memory area in which the incorrect access occurred: • Local data: B#16#40 to 4E, 86, 87, 8E, 8F, C0 to CE • Process image input: B#16#01 • Process image output: B#16#02 • Technology DB: B#16#04 • I: B#16#81 • Q: B#16#82 • M: B#16#83 • DB: B#16#84, 85, 8A, 8B
	DBNr	DB_ANY	10	DB no. if AREA = DB (global DB or instance DB) Only relevant for programming error OB.
	Csg_OBNr	OB_ANY	12	OB number: • 121: Programming error OB • 122: I/O access error OB
	Csg_Prio	USINT	14	OB priority
	Width	USINT	15	Type of access during which the error occurred: • Bit: B#16#00 • Byte: B#16#01 • Word: B#16#02 • DWord: B#16#03 • LWord: B#16#04
* Only certain output values are possible depending on the OB (I/O access error OB or programming error OB) from which information is read.				

Table 4-26 **Time-of-day interrupt OB: Data structure TI_TimeOfDay**

Parameter		Data type	Byte	Description
TI_TimeOfDay				Data structure for time-of-day interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=10)
	OB_Nr	UINT	2	OB number (1...32767).
	CaughtUp	BOOL	4	=TRUE if an OB call was executed subsequently because time was adjusted ahead
	SecondTime	BOOL	5	=TRUE if an OB was called a second time because time was adjusted back. Note: SecondTime is set only once in situations in which the time is set back.

Table 4-27 **Isochronous mode interrupt OB: Data structure TI_SynchCycle**

Parameter		Data type	Byte	Description
TI_SynchCycle				Data structure for isochronous mode interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=61)
	OB_Nr	UINT	2	OB number (1...32767).
	Initial_Call	BOOL	4	= TRUE in the first call of this OB: • Transition from STOP or HOLD to RUN • After reloading
	IO _System	USINT	5	Number of the distributed I/O system triggering the interrupt
	Event_Count	INT	6	• = n: Number of lost cycles • = -1: An unknown number of cycles has been lost (e.g., because cycle has changed).
	PIP_Input	BOOL	10	=TRUE: The associated process image of the inputs is up-to-date
	PIP_Output	BOOL	11	=TRUE: The associated process image of the outputs was transferred to the outputs in good time after the last cycle
	SyncCycle-Time	LTIME	16	Calculated cycle time

Table 4-28 **Status interrupt OB / update interrupt OB for manufacturer or profile-specific interrupt: Data structure TI_Submodule**

Parameter		Data type	Byte	Description
TI_Submodule				Data structure for status interrupt OB / update interrupt OB for manufacturer or profile-specific interrupt
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class • =55 for status interrupt OB • =56 for update interrupt OB • =57 for manufacturer or profile-specific interrupt OB
	OB_Nr	UINT	2	OB number (1...32767).
	LADDR	HW_IO	4	Hardware address of the component triggering the interrupt
	Slot	UINT	6	Slot number of the component triggering the interrupt
	Specifier	WORD	8	Interrupt specifier from the interrupt frame
	address	WORD	20	See "address" parameter for the TI_Cassic. data structure
	slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
	intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
	flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
	flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
	id	UINT	26	See "id" parameter for the TI_Cassic data structure.
	manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
	instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Table 4-29 **CPU redundancy error OB: Data structure TI_CPURedundancyError_V2**

Parameters		Data type	Byte	Description
TI_CPURedundancyError_V2				Data structure for CPU redundancy error OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=72)
	OB_Nr	UINT	2	OB number (1...32767)
	LADDR	HW_ANY	4	Hardware identifier of the defective hardware object
	Event_Class	BYTE	6	Event class and identifiers (B#16#73: Incoming event)
	Fault_ID	BYTE	7	Error code

Parameter AINFO (S7-1200, S7-1500)

Data structure of the destination area AINFO with interrupts from PROFIBUS DP

Byte	Meaning	
0 to 3	Header information, for exact description, see below	
4 to 63	Additional interrupt information: data for the respective interrupt:	
	Distributed:	ARRAY[0] to ARRAY[59]

Structure of the header information with interrupts from PROFIBUS DP

Byte	Data type	Meaning		
0	BYTE	Length of the received interrupt information in bytes		
		Central:	4 to 224	
		Distributed:	4 to 63	
1	BYTE	Central:	Reserved	
		Distributed:	ID for the interrupt type	
			1: 2: 3: 4: 5: 6: 31	Diagnostics interrupt Process interrupt Pull interrupt Plug interrupt Status interrupt Update interrupt Failure of an expansion device, DP master system or DP station
			32 to 126:	Manufacturer-specific interrupt
2	BYTE	Slot number of the component that triggered the interrupt		
3	BYTE	Central:	Reserved	
		Distributed:	Specifier	
			Bits 0 and 1:	0: No further information; 1: Slot faulty 2: Slot not faulty
			Bit 2:	Add_Ack
			Bits 3 to 7:	Sequence number

Data structure of the destination area AINFO with interrupts from PROFINET IO or central I/O devices

Byte	Meaning
0 to 25	Header information, for exact description, see below
26 to 1431	Additional interrupt information: Standardized diagnostics data for the respective interrupt: ARRAY[0] to ARRAY[1405] Note: The additional interrupt information may also be omitted.

Structure of the header information with interrupts from PROFINET IO or central I/O devices

Byte	Data type	Meaning
0 and 1	WORD	• Bits 0 to 7: Block type • Bits 8 to 15: Reserved
2 and 3	WORD	Block length
4 and 5	WORD	Version: • Bits 0 to 7: low byte • Bits 8 to 15: high byte
6 and 7	WORD	ID for interrupt type: • 1: Diagnostics interrupt (incoming or outgoing) • 2: Process interrupt • 3: Remove module interrupt • 4: Insert module interrupt • 5: Status interrupt • 6: Update interrupt • 7: Redundancy interrupt • 8: Controlled by supervisor • 9: Released by supervisor • 10: Configured module not inserted • 11: Return of the sub-module • 12: Diagnostics interrupt (outgoing) • 13: Slave-to-slave connection alarm • 14: Neighborhood change alarm • 15: Clock synchronization message (bus end) • 16: Clock synchronization alarm (device end) • 17: Network component alarm • 18: Time synchronization alarm (on bus side) • 19 to 29: Reserved • 30: Upload-and-Retrieval-Alarm • 31: Reserved • 32 to 127: Manufacturer-specific interrupt • 128 to 65535: Reserved
8 to 11	DWORD	API (Application Process Identifier)
12 to 13	WORD	Slot number of the component triggering the interrupt (value range 0 to 65535)

Byte	Data type	Meaning
14 to 15	WORD	Submodule slot number of the component triggering the interrupt (value range 0 to 65535)
16 to 19	DWORD	Module identification; specific information on the source of the interrupt
20 to 23	DWORD	Submodule identification; specific information on the source of the interrupt
24 to 25	WORD	Interrupt specifier: • Bits 0 to 10: Sequence number (value range 0 to 2047) • Bit 11: Channel diagnostics: 0: No channel diagnostics available 1: Channel diagnostic information exists • Bit 12: Status of manufacturer-specific diagnostics: 0: No manufacturer-specific status information available 1: Manufacturer-specific status information available • Bit 13: Status of diagnostics for sub-module: 0: No status information available, all errors have been corrected 1: At least one item of channel diagnostics and/or status information is available • Bit 14: Reserved • Bit 15: Application relationship diagnosis state: – 0: None of the modules configured within this AR are reporting diagnostic information – 1: At least one of the modules configured in this AR is reporting diagnostic information

Structure of the additional interrupt information for interrupts from PROFINET IO or central I/O

The additional interrupt information for PROFINET IO depends on the format identifier. It can comprise multiple data blocks with the same or different format identifier. The following format identifiers are available:

- W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics
2 to n	BYTE	See manufacturer's manual.

- W#16#8000: Channel diagnostics
  Channel diagnostics is output in blocks of 6 bytes each. The additional interrupt information (without format identifier) is only output for disrupted channels.

Byte	Data type	Meaning	
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8000: Channel diagnostics	
2 to 3	WORD	Channel number of the component triggering the interrupt (value range: 0 to 65535): • W#16#0000 to W#16#7FFF: Channel number of the interface module/sub-module • W#16#8000: The generic substitute for the whole sub-module • W#16#8001 to W#16#FFFF: Reserved	
4	BYTE	Bits 0 to 2:	Reserved
		Bits 3 to 4:	Type of error: • 0: Reserved • 1: Incoming error • 2: Outgoing error • 3: Outgoing error, other errors present
		Bits 5 to 7:	Type of channel: • 0: Reserved • 1: Input channel • 2: Output channel • 3: Input/output channel

Byte	Data type	Meaning
5	BYTE	Data format: • B#16#00: Free data format • B#16#01: Bit • B#16#02: 2-bit • B#16#03: 4-bit • B#16#04: Byte • B#16#05: Word • B#16#06: Double word • B#16#07: 2 double words • B#16#08 to B#16#FF: Reserved

Byte	Data type	Meaning
6 to 7	WORD	Type of error: • W#16#0000: Reserved • W#16#0001: Short-circuit • W#16#0002: Undervoltage • W#16#0003: Overvoltage • W#16#0004: Overload • W#16#0005: Overtemperature • W#16#0006: Wire break • W#16#0007: High limit exceeded • W#16#0008: Low limit value exceeded • W#16#0009: Error • W#16#000A: Simulation active • W#16#000B to W#16#000E: Reserved • W#16#000F to W#16#001F: Manufacturer-specific • W#16#0020 to W#16#00FF: Reserved • W#16#0100 to w#16#7FFF: Manufacturer-specific • W#16#8000: Data transfer not possible • W#16#8001: Error at partner • W#16#8002: Media redundancy error • W#16#8003: Synchronization loss • W#16#8004: Isochronous mode error • W#16#8005: Multicast CR error • W#16#8006: Reserved • W#16#8007: Fiber-optic error • W#16#8008: Function failure of network component • W#16#8009: Time error • W#16#800A: DFP problem (DFP: dynamic frame packing) • W#16#800B: MRPD problem (MRPD: media redundancy with planned duplication) • W#16#800C: System redundancy error • W#16#800D to W#16#FFFF: Reserved Not all channels support every error type. For detailed information, refer to the description of the diagnostic data for the specific device.

Note

The section from "channel number" to "error type" can occur from 0 to n times.

- W#16#8001: MULTIPLE (different types of diagnostics information are transmitted). In this case, the additional interrupt information is transmitted as blocks of variable length.

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8001: Manufacturer-specific diagnostics and/or channel diagnostics
2 to 3	WORD	Block type
4 to 5	WORD	Block length
6	BYTE	Version: high byte
7	BYTE	Version: low byte
8 to 11	DWORD	API (only if low byte of version = 1)
12 to 13	WORD	Slot number
14 to 15	WORD	Subslot number
16 to 17	WORD	Channel number
18 to 19	WORD	Channel properties
20 to 21	WORD	Format identifier: • W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics • W#16#8000: Channel diagnostics • W#16#8002: Extended channel diagnostics • W#16#8003: Stepped extended channel diagnostics • W#16#8004 to W#16#80FF: Reserved
22 to n	BYTE	Data depend on the format identifier

Note

The section starting from "block type" can occur from 1 to n times.

- W#16#8002: Extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8002
2 to 3	Channel number
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information

- W#16#8003: Stepped extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8003
2 to 3	Channel number
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information
14 to 17	Qualified channel qualifier

- W#16#8100: Maintenance information

Byte	Meaning
0 to 1	Format identifier W#16#8100
2 to 3	Block type
4 to 5	Block length
6 to 7	Block version
8 to 9	Reserved Note: These two bytes are missing with central modules. These exist with distributed modules (according to the PROFINET standard).
10 to 13	Maintenance status

Note

You can find more detailed information about the structure of the additional alarm information in the *Programming Manual SIMATIC PROFINET IO from PROFIBUS DP to PROFINET IO* and the current version of IEC 61158-6-10-1.

Destination area TINFO and AINFO (S7-1200, S7-1500)

Destination area TINFO and AINFO

Depending on the respective OB in which "RALRM (Page 2273)" is called, the destination areas TINFO and AINFO are only partially written. Refer to the table below to find out which information is entered respectively.

Interrupt type	OB class	TINFO OB start information	TINFO Management information	AINFO Header information	AINFO Additional interrupt information	
Process interrupt	4x	Yes	Yes	Yes	Central:	No
					Distributed:	As supplied by PROFIBUS DP slave/PROFINET IO device
Status interrupt	55	Yes	Yes	Yes	Yes	Yes
Update interrupt	56	Yes	Yes	Yes	Yes	Yes
Manufacturer-specific interrupt	57	Yes	Yes	Yes	Yes	Yes

Interrupt type	OB class	TINFO OB start information	TINFO Management information	AINFO Header information	AINFO Additional interrupt information	
I/O redundancy error	70	Yes	Yes	No	No	No
Diagnostics interrupt	82	Yes	Yes	Yes	Central:	Structure according to PROFINET IO standard
					Distributed:	As supplied by PROFIBUS DP slave/PROFINET IO device
Insert/remove interrupt	83	Yes	Yes	Yes	Central:	No
					Distributed:	As supplied by PROFIBUS DP slave/PROFINET IO device
Special form of the remove module interrupt: Controlled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Special form of the insert module interrupt: Enabled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Unconfigured module inserted	83	Yes	Yes	Yes	PROFINET IO only	
Rack failure/ station failure	86	Yes	Yes	No	No	
... all other OBs		Yes	No	No	No	

Program example for RALRM

Introduction

In the following example, you generate a wire break via an input module and receive the interrupt, including associated information. The example applies to PROFINET IO and/or an S7-1500 CPU.

Note

Place the example

The example must be stored in the setup in which the input module is to generate an interrupt (wire break).

Requirements

- An input module is created and the hardware identifier is read out in the properties of the input module. You store the hardware identifier via the input parameter F_ID ("alarmCheckID").
- To generate an interrupt, select the option "Wire break" for the input channels. This option is available under "Module parameters > Channel template > Inputs".

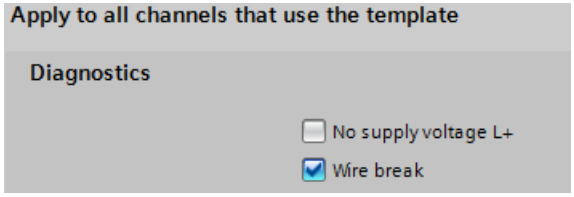

- To store the data, create seven tags and the structures "TI_Diagnostic_Interrupt" and "Additional_Diag" in a global data block. Create the "TI_Diagnostic_Interrupt" structure with the "TI_DiagnosticInterrupt" data type.

SLI_gDB_RALRM

	Name	Data type	Start value
1	Static		
2	alarmMode	Int	2
3	alarmCheckID	HW_IO	257
4	alarmStatus	DWord	16#0
5	alarmFromID	HW_IO	16#0
6	alarmReceived	Bool	false
7	alarmMaxLength	UInt	0
8	alarmLength	UInt	0
9	TI_Diagnostic_Interrupt	TI_DiagnosticInterrupt	
10	Additional_Diag	Struct	
11	header	Struct	
12	BlockType	Word	16#0
13	BlockLength	UInt	0
14	Version	Word	16#0
15	InterruptType	UInt	0
16	API	DWord	16#0
17	SlotNumber	UInt	0
18	SubSlotNumber	UInt	0
19	ModulSpecInfo	DWord	16#0
20	SubmodulSpecInfo	DWord	16#0
21	InterruptSpecifier	Word	16#0
22	addInfo	Struct	
23	FormatIdentifier	Word	16#0
24	ChannelNumber	Word	16#0
25	ChannelErrorType	Byte	16#0
26	DataFormat	Byte	16#0
27	ErrorCode	Word	16#0

Interconnect parameters

Call the instruction "RALRM" in a diagnostic interrupt OB (OB82). Interconnect the parameters of the instruction as follows.

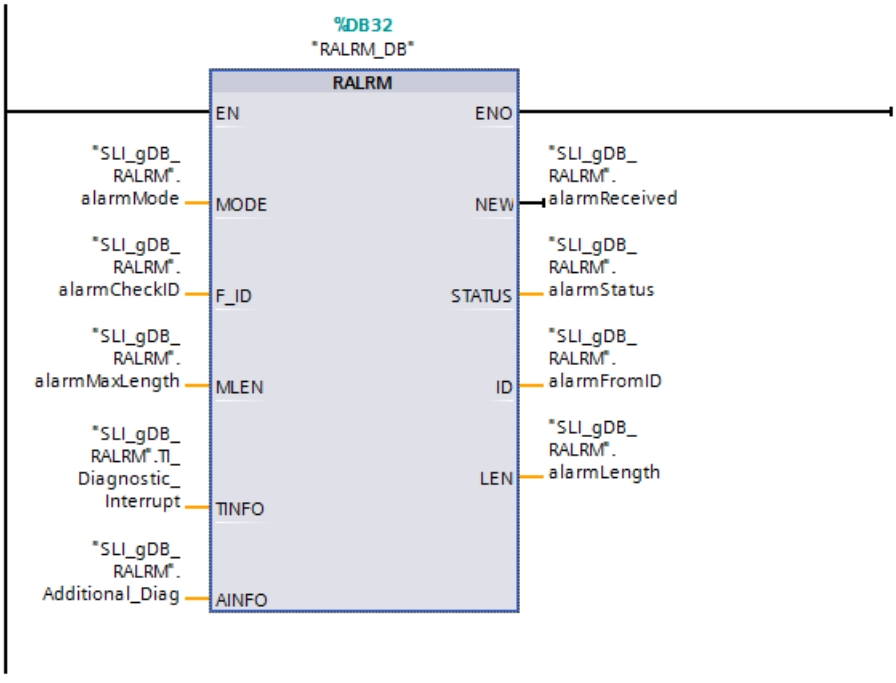

Result of RALRM

Note

Trigger wire break

To trigger the interrupt, pull one of the power cables of the input module's inputs and then reconnect the cable. Alternatively, you can also pull a potential bridge of the input module and reconnect it.

When you generate a wire break, the input module generates an interrupt. The diagnostic interrupt OB is then called and the instruction "RALRM" started.

The instruction "RALRM" calls the input module via the input parameter F_ID ("alarmCheckID"). The instruction "RALRM" checks whether the interrupt originates from the input module according to the value "2" of the input parameter MODE ("alarmMode"). If the interrupt originates from the input module ("alarmCheckID" and "alarmFromID" are identical), the output parameter NEW ("alarmReceived") is set to "TRUE" and the interrupt information is stored via the parameters TINFO ("TI_Diagnostivc_Interrupt") and AINFO ("Additional_Diag"). The hardware identifier of the module generating the interrupt (the input module) is displayed at the output parameter ID ("alarmFromID").

The length of the received interrupt information is recorded at the output parameter LEN ("alarmLength"). According to the value of the input parameter MLEN ("alarmMaxLength" has the value “0"), the information that is to be read for AINFO ("Additional_Diag") is unlimited.

The block status is displayed at the output parameter STATUS ("alarmStatus"). Processing takes place without errors.

SLI_gDB_RALRM				
	Name	Data type	Start value	Monitor value
1	Static			
2	alarmMode	Int	2	2
3	alarmCheckID	HW_IO	257	16#0101
4	alarmStatus	DWord	16#0	16#0000_0000
5	alarmFromID	HW_IO	16#0	16#0101
6	alarmReceived	Bool	false	TRUE
7	alarmMaxLength	UInt	0	0
8	alarmLength	UInt	0	76

The following sections contain the evaluation for TINFO and AINFO.

Details about parameter TINFO ("TI_Diagnostivc_Interrupt")

The OB start information and management information are stored at the parameter TINFO ("TI_Diagnostivc_Interrupt").

The generated interrupt includes the following information:

- OB start information: The OB82 uses optimized start information ("SI_Format" with the value "254").
- Management information:
  - The input module (see "LADDR") with the status "Error" (see the value of "IO_State") is read out.
  - An interrupt was recently recorded for the channel "15" (see the value of "Channel").
  - Multiple errors were recorded ("MutliError" with the value "TRUE").
  - An interrupt was recently recorded for station "0" (see the value of "address").
  - A structure was recorded according to the diagnostics data record 0 (see the value of "slv_prfl").
  - The interrupt was triggered by a configured, distributed module (see the value of "intr_type").

9	TI_Diagnostic_Interrupt	TI_DiagnosticInter...		
10	SI_Format	USInt	0	254
11	OB_Class	USInt	82	82
12	OB_Nr	UInt	0	82
13	LADDR	HW_ANY	16#0	16#0101
14	IO_State	Word	16#0	16#0010
15	Channel	UInt	0	15
16	MultiError	Bool	false	TRUE
17	address	Word	16#0	16#0000
18	slv_prfl	Byte	16#0	16#00
19	intr_type	Byte	16#0	16#00
20	flags1	Byte	16#0	16#00
21	flags2	Byte	16#0	16#00
22	id	UInt	0	0
23	manufacturer	UInt	0	0
24	instance	UInt	0	0

Details about parameter AINFO ("Additional_Diag")

The header information and additional information about the interrupt is stored at the parameter AINFO ("Additional_Diag").

Note

Structure and scope of the interrupt information

This example is intended to show reception of a single hardware fault (wire break). Accordingly, only the interrupt information about one of the relevant channels is displayed when a multierror occurs.

- To call the additional information for each channel individually, use a multiple call of RALRM: For each call, query the channel ("TI_Diagnostivc_Interrupt.Channel") and store the information of the tags "ChannelErrorType" to "ErrorCode" for each channel. See also "FAQ for channel diagnostics with RALRM (https://support.industry.siemens.com/cs/ww/en/view/109480387)".
- Depending on the application, note the structure to be stored at the parameter AINFO (Page 2297).

The generated interrupt includes the following information:

- The OB used is a diagnostic interrupt OB ("InterruptType" with the value "1").
- The interrupt is reported for a module at the slot "2" ("SlotNumber") and for the channels 8 to 15 (BYTE 1, "SubSlotNumber").
- At least one item of channel diagnostics and/or status information is available for the submodule ("InterruptSpecifier" with the value "16#2000").
- In this example, the structure of the channel diagnostics is used ("FormatIdentifier" with the value "16#8000").
- The channel number of the component that triggered the interrupt is assigned to the submodule ("ChannelNumber" with the value "16#0008").
- An incoming error for an input channel ("ChannelErrorType" with the value "16#28") is diagnosed.

- The data format used is a bit ("DataFormat" with the value "16#01").
- A wire break was diagnosed ("ErrorCode" with the value "16#0006").

25	Additional_Diag	Struct		
26	header	Struct		
27	BlockType	Word	16#0	16#0002
28	BlockLength	UInt	0	72
29	Version	Word	16#0	16#0100
30	InterruptType	UInt	0	1
31	API	DWord	16#0	16#0000_0000
32	SlotNumber	UInt	0	2
33	SubSlotNumber	UInt	0	1
34	ModulSpecInfo	DWord	16#0	16#0000_0000
35	SubmodulSpecInfo	DWord	16#0	16#0000_0000
36	InterruptSpecifier	Word	16#0	16#2000
37	addInfo	Struct		
38	FormatIdentifier	Word	16#0	16#8000
39	ChannelNumber	Word	16#0	16#0008
40	ChannelErrorType	Byte	16#0	16#28
41	DataFormat	Byte	16#0	16#01
42	ErrorCode	Word	16#0	16#0006

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Additional example

A more complex example is available here: "FAQ for channel diagnostics with RALRM (https://support.industry.siemens.com/cs/ww/en/view/109480387)".

D_ACT_DP: Activate/deactivate DP slaves (S7-1200, S7-1500)

Description

With the instruction "D_ACT_DP", you can deactivate and activate configured DP slaves/PROFINET IO devices in a targeted manner. In addition, you can determine whether each assigned DP slave or PROFINET IO device is currently activated or deactivated.

Note

S7-1200 CPUs

In S7-1200 CPUs you can only use the instruction D_ACT_DP for PROFINET IO devices. You cannot use it for PROFIBUS DP slaves.

You cannot deactivate/activate an IE/PB Link PN IO type of gateway using the instruction "D_ACT_DP". If you nevertheless use "D_ACT_DP" on the gateway named, the CPU returns

the value W#16#8093 (there is no hardware object that can be activated or deactivated for the address specified in LADDR).

As of firmware version V1.8 of the S7-1500 CPUs the following applies: You can use the "D_ACT_DP" instruction to deactivate/activate DP slaves that are connected to an IE/PB Link PN IO type of gateway.

The instruction cannot be used on PROFIBUS PA field devices that are connected by a DP/PA link to a DP master system.

Note

Several runs through the cycle control point are required to perform the deactivation or activation job. Therefore, you do not wait for the end of such a job in a programmed loop.

Functional description

"D_ACT_DP" is an asynchronous instruction. Processing takes place across several calls. You start the job by calling "D_ACT_DP" with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596).

Application

If you configure DP slaves/PROFINET IO devices in a CPU which are not actually present or not currently required, the CPU will nevertheless continue to access these DP slaves/PROFINET IO devices at regular intervals. After the slaves are deactivated, further CPU accessing will stop. With PROFIBUS DP, the fastest possible DP bus cycle can be achieved in this manner. The corresponding error events no longer occur.

Examples

From a machine OEM's point of view, there are numerous device options possible in series production of machines. However, each delivered machine includes only one combination of selected options.

Every one of these possible machine options is configured as a DP slave/PROFINET IO device by the manufacturer in order to create and maintain a common user program having all possible options. Use "D_ACT_DP" to deactivate all DP slaves/PROFINET IO devices not present at machine startup.

A similar situation exists for machine tools having numerous tooling options available but actually using only a few of them at any given time. These tools are implemented as DP slaves/PROFINET IO devices. With "D_ACT_DP", the user program activates the tools currently needed and deactivates those required later.

Identification of a job

If you have started a deactivation or activation job and you call "D_ACT_DP" again before the job is complete, the behavior of the instruction depends on whether or not the new call involves the same job. If the input parameter LADDR matches, the call will be interpreted as follow-on call.

Deactivating DP slaves/PROFINET IO devices

When you deactivate a DP slave/PROFINET IO device with "D_ACT_DP", its process outputs are set to the configured substitute values or to 0 (safe state). The assigned DP master/PROFINET IO controller does not continue to address this component. Deactivated DP slaves/PROFINET IO devices are not identified as faulty or missing by the error LEDs on the DP master/PROFINET IO controller or CPU.

The process image inputs of deactivated DP slaves/PROFINET IO devices is updated with 0. Therefore, it is treated just like failed DP slaves/PROFINET IO devices.

If you directly access the user data of a previously deactivated DP slave/PROFINET IO device from your program, the I/O access error OB is called and the corresponding start event is entered in the diagnostics buffer. If you attempt to access a deactivated DP slave/PROFINET IO device via an instruction (such as "RD_REC (Page 2332)"), you will receive the same error information in RET_VAL as for an unavailable DP slave/PROFINET IO device.

Deactivating a DP slave/PROFINET IO device does not start the program error OB, even if its inputs or outputs belong to the system-side process image to be updated.

If a DP station/PNIO station fails after you have deactivated it with "D_ACT_DP", the operating system does not detect the failure.

Applies to PROFIBUS DP: If you wish to deactivate DP slaves functioning as transmitters in slave-to-slave communication, we recommend that you first deactivate the receivers (listeners) that detect which input data the transmitter is transferring to its DP master. Deactivate the transmitter only after you have performed this step.

Activating DP slaves/PROFINET IO devices

When you reactivate a DP slave/PROFINET IO device with "D_ACT_DP", this component is configured and assigned parameters by the associated DP master/PROFINET IO controller (as with the return of a failed DP station/PROFINET IO station). This activation is complete when the component is able to transfer user data.

Activating a DP slave/PROFINET IO device does not start the program error OB, even if its inputs or outputs belong to the system-side process image to be updated.

If you try to activate a DP slave/PROFINET IO device with "D_ACT_DP" that cannot be accessed (for example, because it was physically separated from the bus), the instruction returns the error code W#16#80A7 after expiration of the configured parameter assignment time for distributed I/O. The DP slave/PROFINET IO device is activated and the fact that the activated DP slave/PROFINET IO device cannot be accessed results in a corresponding display in the system diagnostics.

If the DP slave/PROFINET IO device is accessible again afterwards, this results in standard system behavior (for example, a call of the OB configured for this purpose).

Note

Activating a DP slave/PROFINET IO device may be time-consuming. If you want to cancel a currently running activation job, start "D_ACT_DP" with the same value for LADDR and MODE =2. You repeat the call for "D_ACT_DP" with MODE = 2 until the successful cancellation of the activation job is displayed with RET_VAL =0.

If you wish to activate DP slaves which take part in the slave-to-slave communication, we recommend that you first activate the transmitters and then the receivers (listeners).

Parameters

The following table shows the parameters of the "D_ACT_DP" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Level-triggered control parameter REQ=1: Run activation or deactivation
MODE	Input	USINT	I, Q, M, D, L or constant	Job identifier Possible values: • 0: Request information on whether the addressed component is activated or deactivated (output via RET_VAL parameter) • 1: Activate the DP slave/PROFINET IO device • 2: Deactivate the DP slave/PROFINET IO device
LADDR	Input	HW_DEVICE	I, Q, M, D, L or constant	Hardware identifier of the DP slave (HW_DPSlave)/PROFINET IO device (HW_Device) The number can be taken from the properties of the DP slave / PROFINET IO device in the Network view or from the "System constants" tab of the standard tag table. If both the identifier for the device diagnostics as well as the ID for operating state transitions are both specified there, you must use the code for the device diagnostics.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	Active code: • BUSY=1: The job is still active. • BUSY=0: The job was terminated.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	Job completed without error.
0001	The DP slave/PROFINET IO device is active (this error code is possible only with MODE = 0.)
0002	The DP slave/PROFINET IO device is deactivated (this error code is possible only with MODE = 0.)
7000	First call with REQ= 0: The job specified with LADDR is not active; BUSY has the value "0".
7001	First call with REQ= 1. The job specified with LADDRwas triggered. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant). The activated job is still active; BUSY has the value "1".
8090	• You have not configured a module with the address specified in LADDR. • You operate your CPU as I-slave / I-device and you have specified an address of this I-slave/I-device in LADDR.
8092	The deactivation of the currently addressed DP slave/PROFINET IO device (MODE=2) cannot be canceled by being activated (MODE=1). Activate the component at a later time.
8093	The address specified in LADDR does not belong to any DP slave/PROFINET IO device that can be activated or deactivated or the MODE parameter is unknown.
8094	You have attempted to activate a device which is potential partner for a tool change port. However, another device is already activated on this tool change port at this time. The activated device will remain activated.
80A0	Error during the communication between the CPU and the IO controller.
80A3	The DP master/PROFINET IO controller concerned does not support this function.
80A4	The CPU does not support this function for external DP masters/PROFINET IO controller.
80A7	A timeout occurred during activation: The remote device is unreachable or you have set the parameter assignment time for central and distributed I/O too short. The status of the remote device is "activated", but it is not reachable.
80AA	Activation with errors in the DP slave/PROFINET IO device: Differences in the configuration
80AB	Activation with errors in the DP slave/PROFINET IO device: Parameter assignment error
80AC	Activation with errors in the DP slave/PROFINET IO device: Maintenance required
80C3	• Temporary resource error: The CPU is currently processing the maximum possible activation and deactivation jobs (8). • The CPU is busy receiving a modified configuration. Currently you can not activate/deactivate DP slaves/PROFINET IO devices.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

ReconfigIOSystem: Reconfigure IO system

ReconfigIOSystem: Reconfigure IO system (S7-1500)

Description

You can use the "ReconfigIOSystem" instruction to transfer a data record to the PROFINET interface of the CPU. This data record ("CTRLREC" parameter) contains the following information for controlling the configuration:

- A list of optional IO devices to be activated
- A list of the partner ports to be specified, if you have set the "Partner is set in the user program" option in the port properties of IO devices

Note

The "ReconfigIOSystem" instruction uses the "D_ACT_DP" instruction internally in MODE 1 and MODE 3 to enable/disable IO devices. Therefore, observe the rules and notes in the description of this instruction.

See also "D_ACT_DP: Disable/enable distributed I/O devices (Page 2310)".

Dependencies between instruction versions

Version V1.1 of instruction "ReconfigIOSystem" requires version V1.1 of the instruction "WRREC".

Version V1.0 of instruction "ReconfigIOSystem" requires version V1.1 of instruction "WRREC".

Functional description

"ReconfigIOSystem" is an asynchronous instruction. Processing takes place across several calls. You start the job by calling "ReconfigIOSystem" with REQ=1.

The output parameters STATUS and BUSY indicate the status of the job.

Parameters

The following table shows the parameters of the "ReconfigIOSystem" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Edge-triggered control parameter REQ=1: Perform data transfer
MODE	Input	UINT	I, Q, M, D, L or constant	You control how the instruction works with the MODE parameter. You can find a detailed functional description in the table below. Possible values: • 1: Disable all IO devices of the IO system for the conversion phase • 2: Reconfigure the IO system according to the data record settings (CTRLREC) • 3: Enable all IO devices of the IO system again after reconfiguration
LADDR	Input	HW_INTERFACE	I, Q, M, D, L or constant	Hardware identifier of the PROFINET interface (IO controller)
CTRLREC	Input	VARIANT	I, Q, M, D, L	Data record for controlling the actual configuration of the IO system
DONE	Output	BOOL	I, Q, M, D, L	0: Instruction not yet completed 1: Instruction completed
BUSY	Output	BOOL	I, Q, M, D, L	Active code: 0: Instruction active 1: Instruction completed
ERROR	Output	BOOL	I, Q, M, D, L	0: Instruction completed without error 1: Instruction completed with error
STATUS	Output	DWORD	I, Q, M, D, L	Result/error code
ERR_INFO	Output	WORD	I, Q, M, D, L	HW ID of the most recently determined IO device that caused an error.

You can find additional information on valid data types under "Overview of valid data types (Page 231)".

Parameter MODE

The following values are possible for the MODE input parameter:

MODE	Description
1	All IO devices of the IO system can be disabled by calling the instruction with mode 1. The "ReconfigIOSystem" instruction uses the "D_ACT_DP" instruction internally. "ReconfigIOSystem" returns errors that are detected by D_ACT_DP in the following output parameters: • STATUS (error code) • ERR_INFO (hardware identifier of the IO device causing the error). In STATUS and ERR_INFO, the CPU enters the last determined error/HW identifier and in so doing overwrites an existing error code. For this reason, additional errors can be present besides the entered error.
2	For controlling the actual configuration of the IO system, the instruction transfers the data record to the PROFINET interface, which is addressed with LADDR. See the following section for information on the structure of the data record. See the following section about the STATUS parameter for error analysis.
3	All non-optional IO devices in the IO system and optional IO devices that are listed in the control data record CTRLREC are enabled. The optional IO devices that are **not** listed in the CTRLREC data record remain disabled. If IO devices that are part of docking units are listed in the CTRLREC control data record, the PN IO system reacts as follows: • IO devices of the docking units remain disabled when ReconfigIOSystem is called with Mode 3. This reaction corresponds to the reaction of a configuration without configuration-controlled IO devices. IO devices of docking units are disabled by default and must be enabled in the user program.

Structure of the control data record

You can use the control data record ("CTRLREC" parameter) to inform the PROFINET interface of the CPU which optionally configured IO devices are available in the real IO system configuration and which port interconnections are to be set.

This requires a configuration that allows IO system configuration to be adapted:

- The IO devices listed in the control data record must be enabled as "Optional IO device" (IO device properties: PROFINET interface [X1] > Advanced options > Interface options).
- The port interconnections listed in the control data record are only possible when the partner port is set to "Partner is set by the user program" for the appropriate ports.

The data type of parameter CTRLREC is "VARIANT". The "CTRLREC" control data record must have the following structure:

- Until version V1.1 of ReconfigIOSystem: Array of elements of the Word data type (for numerical addressing)
- As of version V1.2 of ReconfigIOSystem: Array of elements of the UInt data type (for symbolic or numerical addressing) or array of elements of the Word data type (for numerical addressing)

Below, the basic structural composition of "CTRLREC" is described with the Word element data type.

You read the required hardware identifiers directly in the "System constants" tab in the network view or in the device view. To do this, select the object in the network view (IO device) or in the device view (PROFINET interface).

Recommendation: Use the symbolic addressing of the HW identifiers by using the names of the HW identifiers.

Name	Data type	Comment
Version_High, Version_Low	Word	Version of the control data record(High byte: 01 Low byte: 00
Number_of_opt_Devices_used	Word	Number of optional IO devices that are used in the real IO system configuration. Optional IO devices that are not listed below remain disabled.
Activate_opt_Device_1	UINT / Hw_Device	The optional IO devices that are present in the real configuration must be listed with their hardware identifiers. Use the system constant of the IO device object. Example: The IO device object has the name "IO-Device_4~IO-Device" and its type is "Hw_Device".
Activate_opt_Device_2	UINT / Hw_Device	2nd optional IO device (e.g. with value 262)
...	...	...
Activate_opt_Device_n	UINT / Hw_Device	nth optional IO device (e.g. with value 282)
Number_of_Port_Interconnections_used	Word	Number of port interconnections listed below. If you do not specify port interconnections, enter "0". For all the ports for which you have configured "Partner is set by the user program" and that are not listed below, the CPU uses the "Any partner" setting.
Port_Interconnection_1_Local	UINT / Hw_Interface	1st port interconnection, HW identifier of local port Use the system constant of the port object. Example: The port object has the name "IO-Device_2~PROFINET_interface~Port_2" and its type is "Hw_Interface".
Port_Interconnection_1_Remote	UINT / Hw_Interface	1st port interconnection, HW identifier of partner port
Port_Interconnection_2_Local	UINT / Hw_Interface	2nd port interconnection, HW identifier of local port
Port_Interconnection_2_Remote	UINT / Hw_Interface	2nd port interconnection, HW identifier of partner port
...	...	...
Port_Interconnection_n_Local	UINT / Hw_Interface	nth Port interconnection, hardware identifier local port
Port_Interconnectio_n_Remote	UINT / Hw_Interface	nth Port interconnection, hardware identifier local port

Parameter STATUS

The STATUS output parameter contains error information. You can find a detailed list of possible error codes in the next section. If STATUS is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	B#16#DF: Error writing data record, otherwise B#16#00.
STATUS[2]	Error_Decode	B#16#80 is output for an error (corresponding to IEC 61158-6 in the context of reading and writing data records).

Field element	Name	Meaning
STATUS[3]	Error_Code_1	B#16#AA in the case of an error in the structure of the data record
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension: • B#16#00 Error in the data record (e.g. incorrect padding byte values (not equal to 0) • B#16#01 Reserved • B#16#02 At least one station number for an IO device in the data record is invalid (not configured, points to a non-optional IO device or has the value 0 for an IO controller) • B#16#03 At least one partner port specified in the data record is invalid: Examples: – Subslot address of a partner port is not available – Partner port is configured incorrectly (the correct setting is: "Partner is set by the user program") – Partner port belongs to an IO device that is disabled • B#16#10 Version of the CTRLREC control data record invalid (specified version is not supported) • B#16#11 The number of optional IO devices to be activated in the CTRLREC control data record is not supported • B#16#12 The number of specified interconnections ("Partner set in the user program") in the CTRLREC control data record is not supported • B#16#13 Internal conversion of the hardware identifier to device number failed. The ERR_INFO output parameter contains the hardware identifier of the device causing the error. • B#16#14 Consistency error: The length of the CTRLREC control data record does not match the information in the control data record. Example: 20 optional IO devices are specified, but the control data record only has a length of 10 bytes.

Error codes (STATUS parameter)

Error code	Description
16#0000_0000	Job completed without error
16#0070_0000	No job active
16#0070_0100	First call of the instruction
16#0070_0200	Following call of the instruction (instruction is still running, BUSY = 1)
16#0080_8000	MODE is not supported
16#0084_5100	Incorrect data type of CRTLREC data record. Use an array of UInt or an array of Word.
16#0080_9100	The LADDR parameter does not address any PROFINET interface (does not exist or wrong type, e.g. PROFIBUS interface). The PROFINET interface does not support configuration control of IO systems.
16#0080_Cx00	Temporary error, e.g., due to temporary lack of resources.

Error code	Description
16#DF80_AAxy	Error in the structure of the data record (MODE 2). For the meaning of "xy", see the definition of STATUS[4] Error_Code_2 above.
16#DF80_B6xy	Configuration control not possible because either no optional IO device was configured or no port was assigned with "Partner is set in the user program". This configuration is a prerequisite for calling the instruction. "xy" is irrelevant.
16#0080_9400 16#0080_A000 16#0080_A700 16#0080_AA00 16#0080_AB00 16#0080_AC00	Forwarded error codes of the internally called D_ACT_DP instruction. For the meaning of these error codes, see "D_ACT_DP (Page 2310)". The hardware identifiers of IO devices causing errors are entered in ERR_INFO (entry is consecutively overwritten by subsequent errors). If more than one IO device is involved, online diagnostics with STEP 7 is recommended. With IRT configuration: The device numbers of the IO devices should follow from the IO controller in ascending order of their topological interconnection, see here.

Example

You will find the example here: Program example for ReconfigIOSystem (Page 2320).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

Program example for ReconfigIOSystem

Introduction

In the following example, you configure an IO system via the program using the control data record "CTRLREC". The target configuration should consist of an S7-1500 CPU, an IO device (for example ET 200MP) and an optional IO device (for example, ET 200SP). For this purpose, disable an existing IO system, transfer the configuration values to the PROFINET interface of the CPU and enable the IO devices.

You can find additional information on the data records in Overview of the types of data records (Page 2206).

Requirements

- An S7-1500 CPU has been created.
- Two ET 200 have been created.
- A PROFINET connection has been created and the hardware settings have been made:
  - IO device (for example, ET 200MP IM 155-5 PN HF)
  - optional IO device (for example, ET 200SP IM 155-6 PN ST
  - IO system is set up by the program

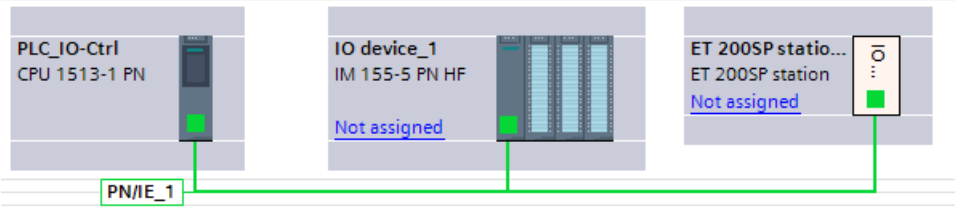

Note

Hardware settings for an IO system to be set up by the program

Proceed as follows:

1. Connect the two ET 200 via PROFINET with the CPU.
2. Assign the two ET 200 as IO controllers to the CPU. Then open the CPU properties.
3. Open the entry "PROFINET interface [X1] > Advanced options > Port X1 P1 > Port interconnection > Partner port". Select the option "Set Partner by user program". Do the same for "Port X1 P2".
4. Open the properties of ET 200MP (...> Port X1 P1) and select the option "Set Partner by user program" (see step 3).
5. Open the properties of ET 200SP (...> Port X1 P1) and perform the following settings:
   - Select the option "Set Partner by user program" (see step 3).
   - Open "PROFINET interface [X1] > Advanced options > Interface options" and select the option "Optional IO device".

Storage of data

To store the data, create the two structures "operateMode" and "ctrlRec" and 13 additional tags in a global data block.

The structure "ctrlRec" contains the control data record "CTRLREC".

SLI_gDB_ReconfigIOSystem

	Name	Data type	Start value
1	Static		
2	start	Bool	false
3	startDone	Bool	false
4	reset	Bool	false
5	getExecute	Bool	false
6	operateMode	Struct	
7	value	UInt	1
8	Disable_all_IO_devices_done	Bool	false
9	Reconfigure_IO_system_done	Bool	false
10	Enable_all_IO_devices_done	Bool	false
11	laddr_interface	HW_INTERFACE	64
12	ctrlRec	Struct	
13	Version	Word	16#0100
14	Number_of_opt_Devices_used	Word	1
15	Activate_opt_Device_1	HW_DEVICE	264
16	Number_of_Port_Interconnections_used	Word	2
17	Port_Interconnection_1_Local	HW_INTERFACE	65
18	Port_Interconnection_1_Remote	HW_INTERFACE	259
19	Port_Interconnection_2_Local	HW_INTERFACE	66
20	Port_Interconnection_2_Remote	HW_INTERFACE	273
21	status	DWord	16#0
22	busy	Bool	false
23	done	Bool	false
24	error	Bool	false
25	errorDevice	Word	16#0
26	memErr	Bool	false
27	memErrDevice	Word	16#0
28	memErrStatus	DWord	16#0

Note

You have to specify the values of the hardware identifiers ("laddr_interface", "ctrlRec") according to your system.

For example, determine the hardware identifier of the interface of the CPU ("laddr_interface") as follows:

In the CPU, open the entry "PLC tags > Show all tags > System constants" and search for the name <Local~PROFINET-Interface_1> with the data type "Hw_Interface". The associated cell "value" contains the hardware identifier.

Note

Pay attention to the interfaces used.

- The Port X1 P1 of the IO device (ET 200MP/"<IO_device_1>") is connected with the port X1 P1 of the CPU ("<Local>").
- The Port X1 P1 of the optional IO device (ET 200SP/"<IO_device_2>") is connected with the port X1 P2 of the CPU ("<Local>"). You also require the hardware identifier of the device (data type "Hw_Device") from the optional IO device.

35	Local	Hw_SubModule	257	
36	Local~Common	Hw_SubModule	50	
37	Local(1)	Hw_SubModule	49	
38	OB_Main	OB_PCYCLE	1	
39	Local~Display	Hw_SubModule	54	
40	Local~MC	Hw_SubModule	51	
41	Local~Exec	Hw_SubModule	52	
42	Local~PROFINET_interface_1	Hw_Interface	64	-->laddr
43	Local~PROFINET_interface_1~Port_1	Hw_Interface	65	-->Port 1 local
44	Local~PROFINET_interface_1~Port_2	Hw_Interface	66	-->Port 2 local
45	Local~PROFINET_IO-System	Hw_IoSystem	258	
46	IO_device_1~Proxy	Hw_SubModule	267	
47	IO_device_1~IODevice	Hw_Device	265	
48	IO_device_1~PROFINET_interface	Hw_Interface	268	
49	IO_device_1~PROFINET_interface~Port_1	Hw_Interface	259	-->Port 1 remote
50	IO_device_1~PROFINET_interface~Port_2	Hw_Interface	260	
51	IO_device_1~Head	Hw_SubModule	261	
52	IO_device_1~DI_16x24VDC_HF_1	Hw_SubModule	262	
53	IO_device_1~DQ_8x24VDC_2A_HF_1_1	Hw_SubModule	263	
54	IO_device_2~Proxy	Hw_SubModule	270	
55	IO_device_2~IODevice	Hw_Device	264	-->Used as optional
56	IO_device_2~Head	Hw_SubModule	271	IO device
57	IO_device_2~PROFINET_interface	Hw_Interface	272	
58	IO_device_2~PROFINET_interface~Port_1	Hw_Interface	273	-->Port 2 remote
59	IO_device_2~PROFINET_interface~Port_2	Hw_Interface	274	
60	IO_device_2~DQ_8x24VDC_0_5A_HF_1	Hw_SubModule	276	

Interconnecting parameters in the FC "SLI_FC_start_reset_ReconfigIOS"

To start or reset the instruction "ReconfigIOSystem", create the FC "SLI_FC_start_reset_ReconfigIOS". To pass on the data, create the following local tags in the FC.

SLI_FC_start_reset_ReconfigIOS

	Name	Data type
1	Input	
2	start	Bool
3	reset	Bool
4	Output	
5	<Add new>	
6	InOut	
7	startDone	Bool
8	Temp	
9	tempStart	Bool
10	resetDone	Bool

Network 1: To set the input parameter REQ ("getExecute"), interconnect the tag "start".

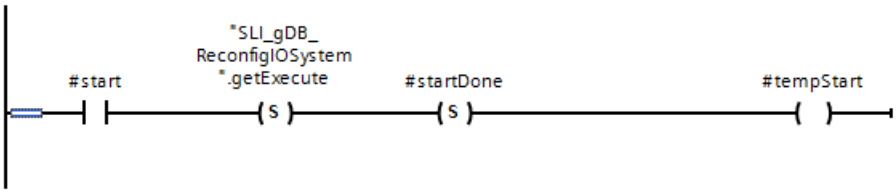

Network 2: To reset "start", create the following interconnections.

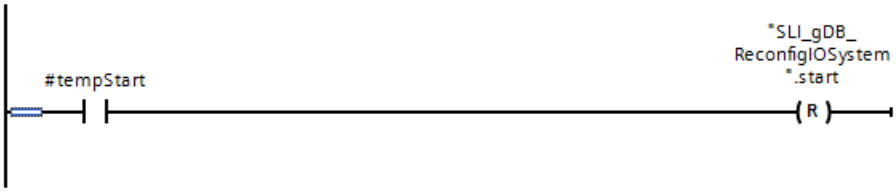

Network 3: To reset all set process values, create the following interconnections. Part 1 of the network:

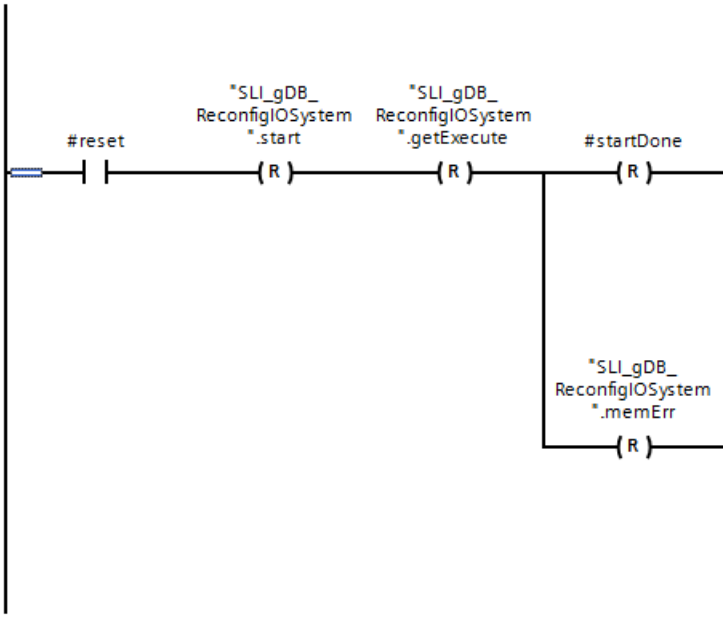

Part 2 of the network:

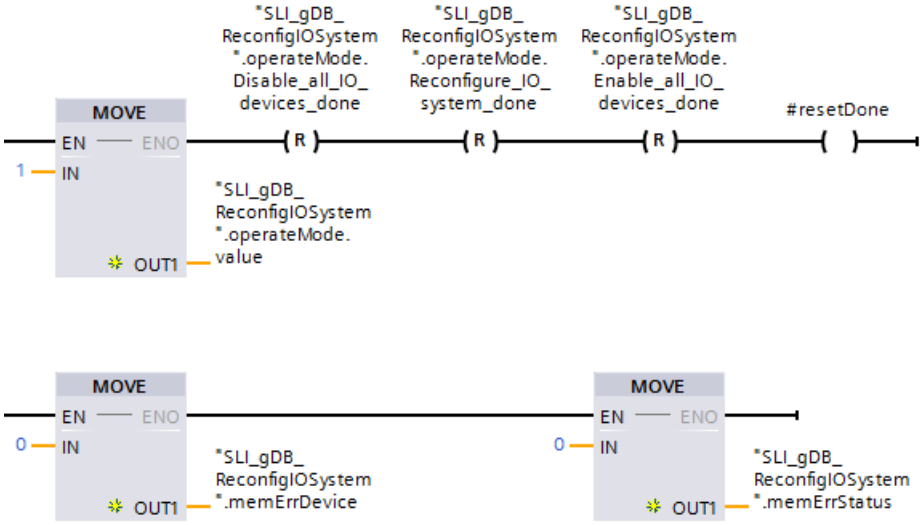

Network 4: To reset "reset", create the following interconnections.

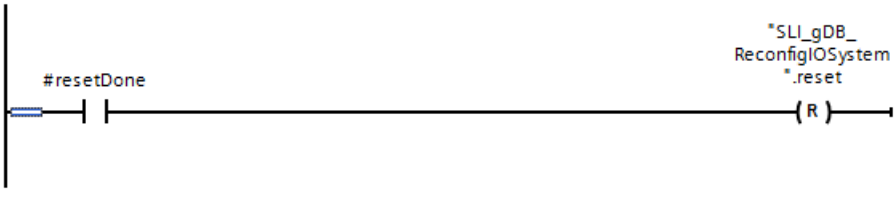

Interconnecting parameters in the FC "SLI_FC_switchMode_ReconfigIOS"

To safely reconfigure the IO system, run through all three modes of "ReconfigIOSystem". You need to set an automatism for mode switchover in the FC "SLI_FC_switchMode_ReconfigIOS" so that you do not have to switch the modes of the "ReconfigIOSystem" instruction yourself.

Network 1: To save the information in the case of error, create the following interconnections.

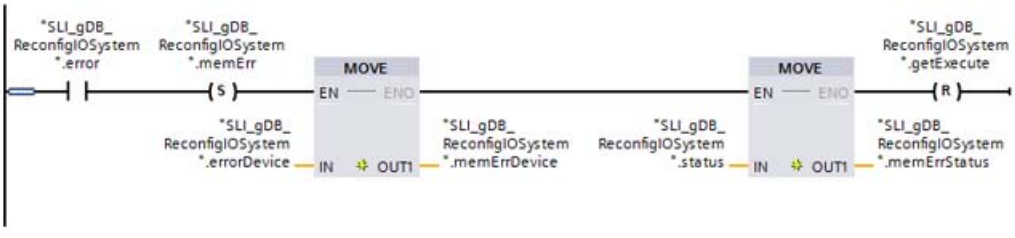

Network 2: To automatically start the instruction "ReconfigIOSystem" in mode 2 or mode 3, create the following interconnections.

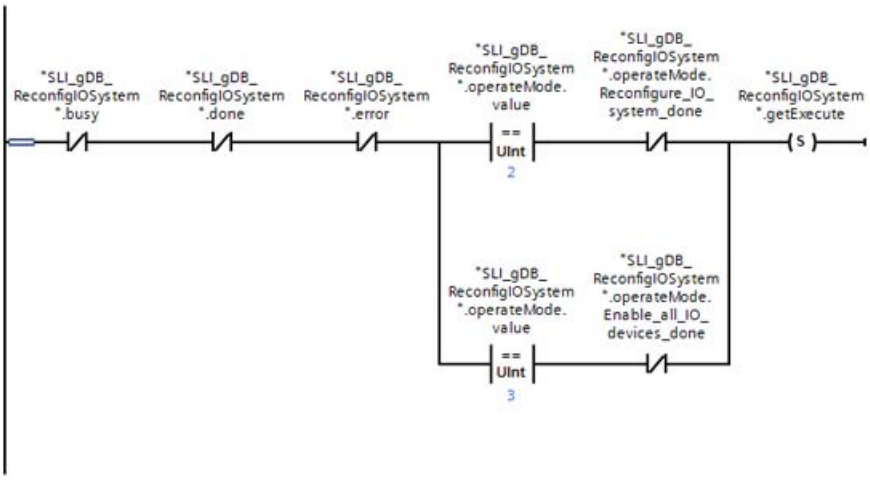

Network 3: Record the success of the processing of ReconfigIOSystem in mode 1 as follows.

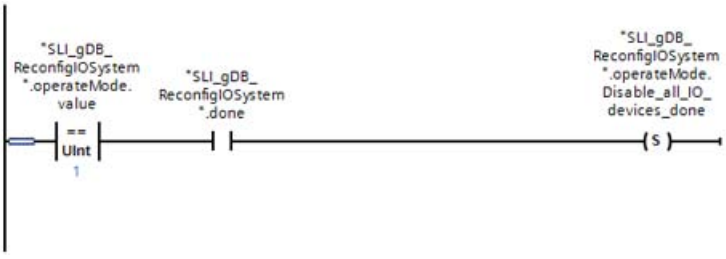

Network 4: To safely switch from mode 1 to mode 2, create the following interconnections.

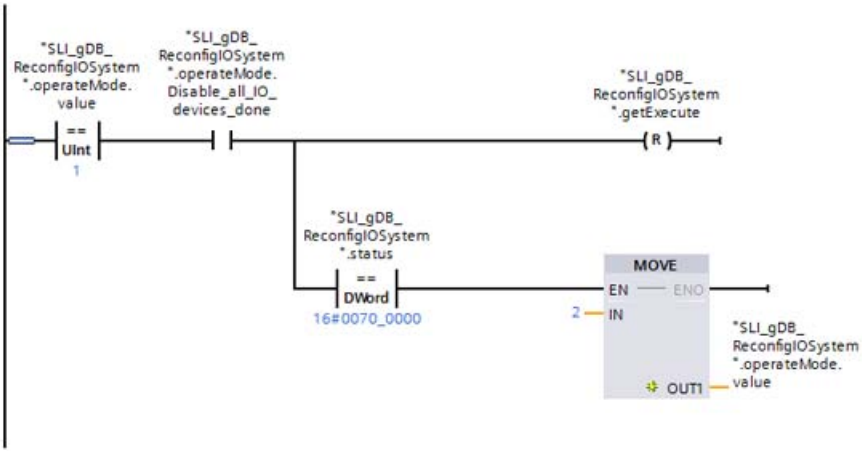

Network 5: Record the success of the processing of ReconfigIOSystem in mode 2 as follows.

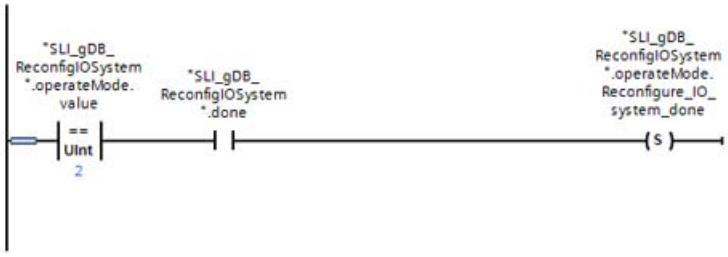

Network 6: To safely switch from mode 2 to mode 3, create the following interconnections.

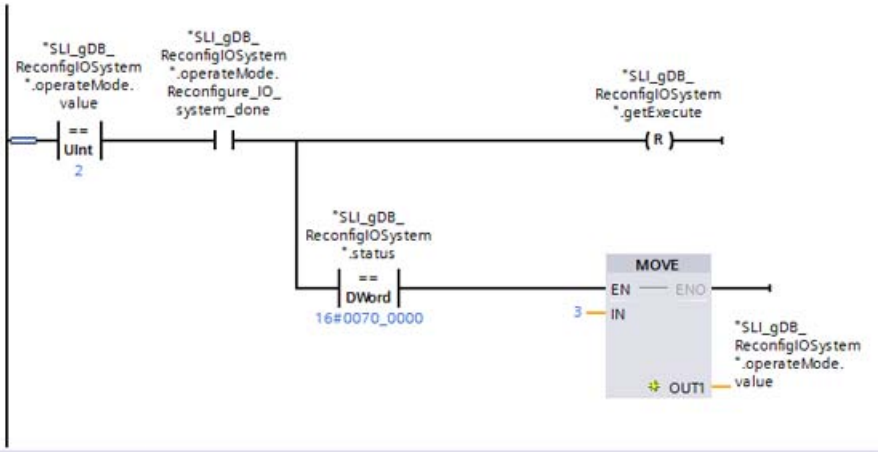

Network 7: Record the success of the processing of ReconfigIOSystem in mode 3 as follows.

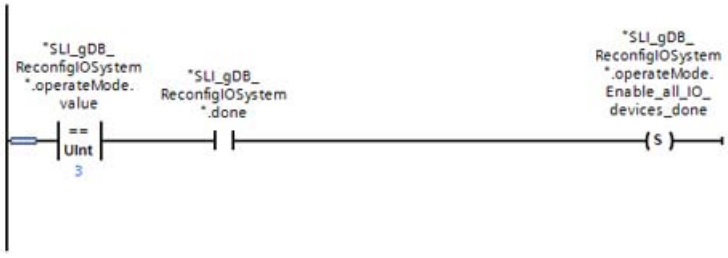

Network 8: To end the processing of ReconfigIOSystem, create the following interconnections.

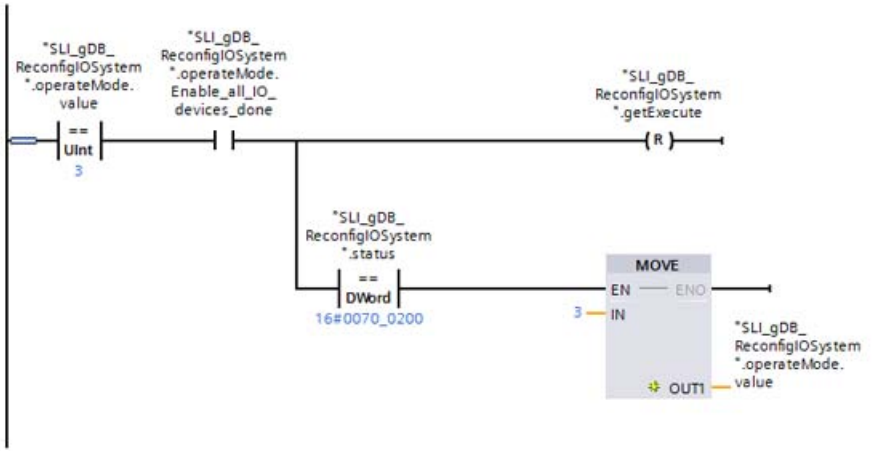

Note

Alternatively, you can switch from mode 3 to mode 1 after processing. For this purpose, enter the value "1" for the instruction "MOVE".

Interconnecting parameters in the FB "SLI_FB_ReconfigIOSystem"

Call the instruction "ReconfigIOSystem" in an FB ("SLI_FB_ReconfigIOSystem"). To pass on the data, create the following local tags in the FB. Create the tags "#recordAoWord" and "#recordStruct" as AT constructs. The tags must be the same size (as "ctrlRec").

7		Static			
8		ReconfigIOSystem_Instance_1	ReconfigIOSyste...		
9		recordAoWord	Array[0..7] of Word		Set in IDB
10		recordStruct AT "recordAoWord"	Struct		Set in IDB
11		Version	Word		Set in IDB
12		Number_of_opt_Devices_used	Word		Set in IDB
13		Activate_opt_Device_1	HW_DEVICE		Set in IDB
14		Number_of_Port_Interconnections_...	Word		Set in IDB
15		Port_Interconnection_1_Local	HW_INTERFACE		Set in IDB
16		Port_Interconnection_1_Remote	HW_INTERFACE		Set in IDB
17		Port_Interconnection_2_Local	HW_INTERFACE		Set in IDB
18		Port_Interconnection_2_Remote	HW_INTERFACE		Set in IDB

Network 1: To transfer the data record ("ctrlRec"), create the following interconnection.

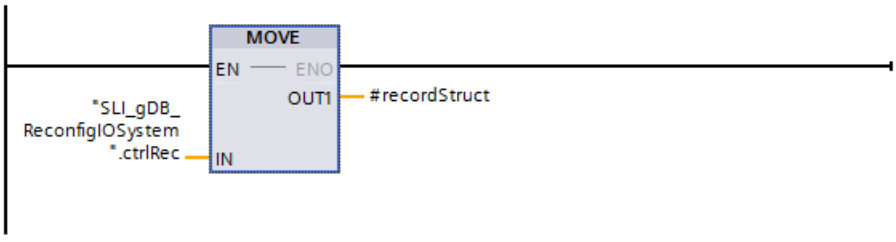

Network 2: Interconnect the FC "SLI_FC_start_reset_ReconfigIOS" as follows.

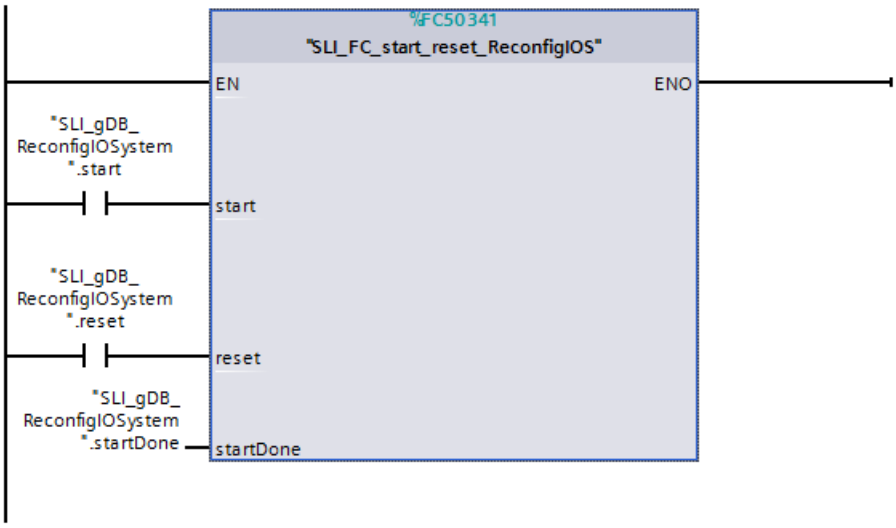

Network 3: Interconnect the instruction "ReconfigIOSystem" as follows.

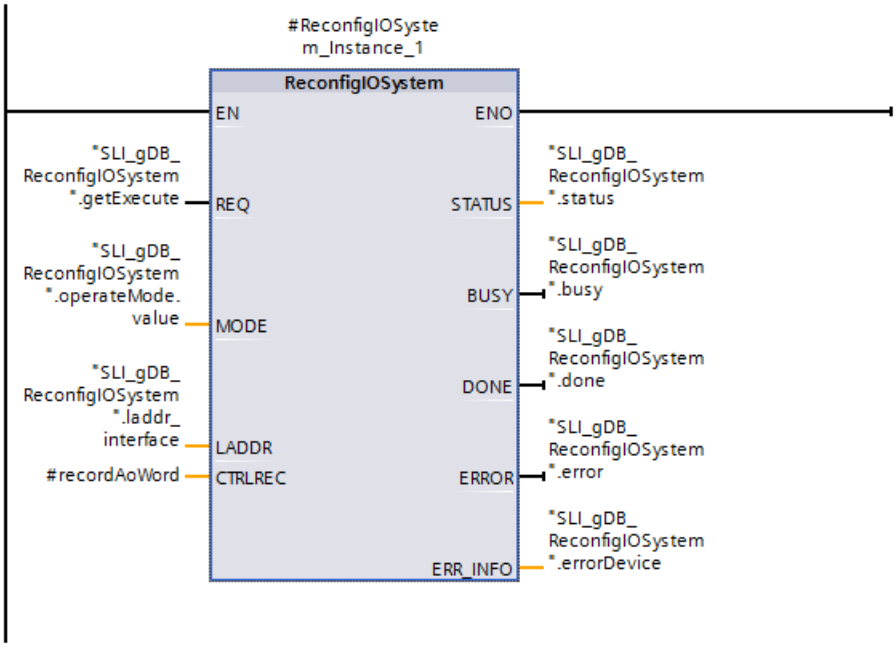

Network 4: Interconnect the FC "SLI_FC_switchMode_ReconfigIOS" as follows.

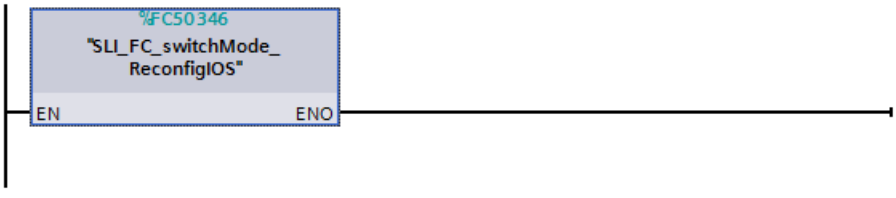

Call the FB ("SLI_FB_ReconfigIOSystem") in an OB1.

Result (FB "SLI_FB_ReconfigIOSystem")

FB Network 1:

The values of the data record ("ctrlRec") are transferred to an Array of WORD ("#recordAoWord") via a MOVE instruction. To improve understanding, the tags of the "ctrlRec" and "#recordStruct" structures are represented by descriptive names.

FB Network 2:

When the input parameter start ("start") returns the signal state "TRUE", the tag "getExecute" and the output parameter startDone ("startDone") are set to "TRUE". The processing of the instruction "ReconfigIOSystem" is started by the tag "getExecute" at "TRUE".

When the input parameter reset ("reset") returns the signal state "TRUE", the start command of "ReconfigIOSystem", the process values of the processing and the information of the fault scenario are reset.

FB Network 3:

When the input parameter REQ ("getExecute") returns the signal state "TRUE", the "ReconfigIOSystem" instruction is started. The instruction "ReconfigIOSystem" calls the interface of the CPU via the input parameter LADDR ("laddr_interface").

The mode for the operating mode is transferred to the instruction "ReconfigIOSystem" via the input parameter MODE ("operateMode.value"). ReconfigIOSystem disables all existing IO systems of the CPU according to the start value "1". The block status is displayed at the output parameter STATUS ("status"). In the example, processing takes place without errors.

The "ReconfigIOSystem" instruction only calls the data record at the input parameter CTRLREC ("#recordAoWord") in mode 2. The configuration values for the IO system of ReconfigIOSystem to be set up are communicated to the CPU (IO controller).

In mode 3, the "ReconfigIOSystem" instruction enables the IO system. The switchover of the input parameters REQ ("getExecute") and MODE ("operateMode.value") is implemented in network 4 ("SLI_FC_switchMode_ReconfigIOS").

SLI_gDB_ReconfigIOSystem				
	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	FALSE
3	startDone	Bool	false	TRUE
4	reset	Bool	false	FALSE
5	getExecute	Bool	false	FALSE
6	operateMode	Struct		
7	value	UInt	1	3
8	Disable_all_IO_devices_done	Bool	false	TRUE
9	Reconfigure_IO_system_done	Bool	false	TRUE
10	Enable_all_IO_devices_done	Bool	false	TRUE
11	laddr_interface	HW_INTERFACE	64	16#0040
12	ctrlRec	Struct		
13	Version	Word	16#0100	16#0100
14	Number_of_opt_Devices_used	Word	1	16#0001
15	Activate_opt_Device_1	HW_DEVICE	264	16#0108
16	Number_of_Port_Interconnections_used	Word	2	16#0002
17	Port_Interconnection_1_Local	HW_INTERFACE	65	16#0041
18	Port_Interconnection_1_Remote	HW_INTERFACE	259	16#0103
19	Port_Interconnection_2_Local	HW_INTERFACE	66	16#0042
20	Port_Interconnection_2_Remote	HW_INTERFACE	273	16#0111
21	status	DWord	16#0	16#0070_0000
22	busy	Bool	false	FALSE
23	done	Bool	false	FALSE
24	error	Bool	false	FALSE
25	errorDevice	Word	16#0	16#0000
26	memErr	Bool	false	FALSE
27	memErrDevice	Word	16#0	16#0000
28	memErrStatus	DWord	16#0	16#0000_0000

FB Network 4:

If the instruction "ReconfigIOSystem" was successfully processed in mode 1 (the tag "done" has the value "TRUE"), the tag "operateMode.Disable_all_IO_devices_done" is set to "TRUE". As a result, the input parameter REQ ("getExecute") is then reset to "FALSE". As soon as ReconfigIOSystem is inactive ("status" has the value "16#0070_0000"), the value "2" is passed to the tag "operateMode.value". ReconfigIOSystem is then in mode 2.

In mode 2, the input parameter REQ ("getExecute") is automatically set to "TRUE" if the following applies to the instruction "ReconfigIOSystem":

- ReconfigIOSystem is inactive ("busy" and "done" is "FALSE").
- No error occurs ("error" is "FALSE").
- The success state of mode 2 is not reached ("operateMode.Reconfigure_IO_system_done" is "FALSE").

If the instruction "ReconfigIOSystem" was successfully processed in mode 2 (the tag "done" has the value "TRUE"), the tag "operateMode.Reconfigure_IO_system_done" is set to "TRUE". Afterwards, the scenario is similar to with mode 1.

- The "getExecute" tag is reset to "FALSE".
- ReconfigIOSystem ("operateMode.value") is set to mode 3.

In mode 3 the input parameter REQ ("getExecute") is automatically set to "TRUE" if the same applies to the "ReconfigIOSystem" instruction as for mode 2 - with one exception:

- The success state of mode 3 is not reached ("operateMode.Enable_all_IO_devices_done" is "FALSE").

The success status in mode 3 (the tag "done" has the value "TRUE") is queried in the same way as for mode 2. After the successful processing ReconfigIOSystem remains in mode 3 ("operateMode.value" retains the value).

After the completed processing of the three modes, the "ReconfigIOSystem" instruction is inactive ("status" has the value "16#0070_0000"). The IO system is set up: An S7-1500 CPU with an IO device and an optional IO device. Both IO devices are activated.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

ReconfigIOSystem: Reconfigure IO system (Page 2315)

Others

RD_REC: Read data record from I/O (S7-1500)

Description

Use the instruction to read the data record with the number RECNUM from the addressed module. You start the read process by assigning the value "1" to the input parameter REQ during the call. If the read process is executed immediately, the instruction returns the value "0" at the output parameter BUSY . If BUSY has the value "1", reading is not yet complete.

See also: Difference between synchronous and asynchronous instructions (Page 596). The data record read is entered in the target range spanned by the RECORD parameter, providing the data transfer was free of errors.

Note

When you read out a data record with a number greater than one from an FM or a CP you have purchased prior to February 1997 (below referred to as "old modules"), "RD_REC" responds differently than it does in new modules. This special situation is covered in the section "Using old S7-300 FMs and CPs with data record numbers >1" (see below).

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not read any data records from the I/O modules in the user program with "RD_REC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: Set the interface of the DP master to "DPV1".

Parameters

The following table shows the parameters of the "RD_REC" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. In addition: Length of the actually transferred data record in bytes (possible values: +1 to +240), if the target range is greater than the transferred data record and if no error occurred during the transfer.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
RECORD	Output	VARIANT	I, Q, M, D, L	Target range for the data record read. With asynchronous execution of "RD_REC", make sure that the actual parameters of RECORD have the same length information for all calls. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs, the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is illegal for S7-300 CPUs and leads to an error message in the user program.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RECORD

Note

If you want to ensure that the entire data record is always read, select a target range with a length of 241 bytes. If the data transfer is without error, RET_VAL contains the actual data record length.

Using old S7-300 FMs and CPs with data record numbers > 1

If you want to use the instruction "RD_REC" to read out a data record with a number greater than one from an old S7-300 FM or old S7-300 CP, remember the following points:

- If the target range is greater than the actual length of the required data record, no data is entered in RECORD . RET_VAL is written with W#16#80B1.
- If the target range is less than the actual length of the required data record, the CPU reads as many bytes beginning at the start of the record as are specified in the length information of RECORD and enters this number of bytes in RECORD . RET_VAL is written with "0".
- If the length specified in RECORD is the same as the actual length of the required data record, the CPU reads the data record and enters it in RECORD . RET_VAL is written with "0".

Parameter RET_VAL

- If an error occurred while the function was being executed, the return value contains an error code.
- If no error occurred during the transfer, RET_VAL contains the following:
  - 0, if the entire target range was filled with data from the selected data record (the data record can also be incomplete).
  - The length of the actually transferred data record in bytes (possible values: +1 to +240), if the target range is greater than the transferred data record.

Note

If the general error W#16#8745 occurs, this only indicates that access to at least one byte of the process image was blocked. The data record was read by the module correctly and written to the I/O memory area.

When looking at the "real" error information (error codes W#16#8xyz) in the following table, two different cases are described:

- Temporary errors (error codes W#16#80A2 to 80A3, 80Cx):
  It is possible that this type of error is eliminated without user action. For this reason, it is advisable to call the instruction again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#809x, 80A0, 80A1, 80Bx):
  This type of error does not correct itself. Once you have eliminated the error, the instruction needs to be called again. Example of a permanent error: Wrong length specification in RECORD (W#16#80B1).

Note

If you transfer data records to a DPV1 slave with "WR_REC (Page 2337)" or if you read data records from a DPV1 slave with RD_REC and if this DPV1 slave operates in DPV1 mode, the DP master evaluates the error information it received from the slave as follows:

If the error information lies within the range from W#16#8000 to W#16#80FF or W#16#F000 to W#16#FFFF, the DP master passes the error information to the instruction. If the error information is outside this range, the DP master passes the value W#16#80A2 to the instruction and suspends the slave.

For a description of the error information originating from DPV1 slaves, refer to STATUS[3] Parameter STATUS (Page 2276).

Parameter RET_VAL for WR_REC and RD_REC

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer started; BUSY has the value 1.	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value 1.	Distributed I/O
8090	Address specified at parameter ADDR is invalid.	-
8092	A type other than BYTE is specified in the ANY reference.	-
8093	This instruction is not permitted for the module selected by means of LADDR and IOID .	-
80A0	Negative acknowledgment when reading from the module: Module was removed during the reading process or is defective	With "RD_REC" only
80A1	Negative acknowledgment when writing to the module: Module was removed during the writing process or is defective	With "WR_REC (Page 2337)" only
80A2	• DP protocol error at layer 2 (for example, slave failure or bus problems) • For ET 200S, data record cannot be read in DPV0 mode.	Distributed I/O
80A3	DP protocol error with user interface/user	Distributed I/O
80A4	Communication on PROFIBUS disrupted.	-

Error code* (W#16#...)	Explanation	Restriction
80B0	• Instruction not possible for module type. • The module does not recognize the data record. • Data record number 241 not permitted. • With "WR_REC (Page 2337)", data records 0 and 1 are not permitted.	-
80B1	The length specified in parameter RECORD is incorrect.	• With "WR_REC (Page 2337)": Length incorrect • With "RD_REC" (only when using old S7-300 FMs and S7-300 CPs): specified length > data record length • With DPNRM_DG: specified length < data record length
80B2	The configured slot is not occupied.	-
80B3	Actual module type does not match specified module type	-
80B5	DP slave or module is not ready.	-
80B7	DP slave or module reports an invalid range for a parameter or value.	With "RD_REC" only
80C0	With "WR_REC (Page 2337)": The data is only written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You can only write the data online with PG/PC. With "RD_REC": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. In this case, you can only read the data online with PG/PC. With "DPNRM_DG (Page 2353)": No diagnostics data available.	With "WR_REC (Page 2337)" or "RD_REC" or "DPNRM_DG (Page 2353)"
80C1	The data of the previous write job on the module for the same data record has not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	-
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available.	Distributed I/O
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O
80Ex	Manufacturer-specific. Please check the documentation of the module.	-

Error code* (W#16#...)	Explanation	Restriction
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)	-
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 7698)

WR_REC: Write data record to I/O (S7-1500)

Description

Use the instruction "WR_REC" to transfer the data record RECORD to the addressed module.

The data to be transferred are read from the RECORD parameter during the first call. If the transfer of the data record takes longer than the duration of one call, the contents of the RECORD parameter are no longer relevant for the subsequent instruction calls (for the same job).

You start the writing process by assigning the value "1" to the input parameter REQ during the call. If the writing process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY. If BUSY has the value "1", writing is not yet complete.

Parameters

The following table shows the parameters of the instruction "WR_REC":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	REQ = 1: Write request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number
RECORD	Input	VARIANT	I, Q, M, D, L	Data record
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

See also: RD_REC: Read data record from I/O (Page 2332)

Note

If the general error W#16#8544 occurs, it only indicates that access to at least one byte of the I/O memory area containing the data record was denied. The data transfer was continued.

DPRD_DAT: Read consistent data of a DP standard slave (S7-1200, S7-1500)

Description

Use the instruction "DPRD_DAT" to read out consistent data from an I/O module.

The instruction can be used for the central modules as well as for the DP standard slaves and PROFINET IO devices.

You require "DPRD_DAT" because you can only read out a maximum of four continuous bytes using the load commands that access the I/O or the process image input table. If required, you can also read consistent data via the process image of the inputs. Refer to the related documentation to find out if your CPU supports this functionality. For additional information on consistent data of a DP standard slave/PROFINET IO device, refer to Section "Data consistency (Page 3442)".

If necessary, use the instruction "DPRD_DAT" also for a data area of 1 byte or larger. For information on the maximum length of the data, refer to the documentation of your CPU (e.g. 64 bytes for an S7-1214).

- Use the parameter LADDR to select the module of the DP normslave/PROFINET IO device. If an access error occurs, the error code W#16#8090 is output.
- Use the RECORD parameter to define the target range for the read data:
  - The target range has to be at least as long as the inputs of the selected module. If the target range is longer than the inputs of the module, it is written from the front with the values of the inputs; with S7-1200 CPUs the remaining bytes of the target range remain unchanged, with S7-1500 CPUs the remaining bytes are filled with zeros. If you read from a DP standard slave with a modular configuration or with several DP identifiers, you can only access the data of a module of the configured hardware identifier per "DPRD_DAT" call. If the selected target range is too small, the error code RET_VAL is output at the 80B1 parameter.
  - The following data types can be used: BOOL, BYTE, CHAR, WCHAR, WORD, LWORD, DWORD, INT, UINT, USINT, SINT, LINT, ULINT, DINT, UDINT, REAL, LREAL. The use of these data types in a data structure of the type ARRAY or STRUCT is also permissible.
  - The STRING and WSTRING data types are not supported.
  - PLC data types (UDT) and system data types are supported.
- If there was no error during the data transmission, the data that has been read is entered in the target range defined at the RECORD parameter.

Parameters

The following table shows the parameters of the instruction "DPRD_DAT":

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Hardware ID of the module from which the data is to be read. The hardware ID can be found in the properties of the module in the device view or system constants.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	VARIANT	I, Q, M, D, L	Target range for the user data that were read.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	• You have not configured a module for the specified hardware identifier or • You have ignored the restriction concerning the length of consistent data, or • you have not specified a hardware identifier as an address at parameter LADDR .
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.
8093	No DP module/PROFINET IO device from which you can read consistent data exists for the hardware identifier specified in LADDR . This error code also occurs if the module addressed by means of LADDR does not have inputs.
80A0	An access error was detected when accessing the I/O.
80B1	The length of the specified target range at parameter RECORD is shorter than the configured user data length.
80C0	The data have not been read yet.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Parameter STATUS (Page 2276)

Switching display formats in the program status (Page 7698)

DPWR_DAT: Write consistent data of a DP standard slave (S7-1200, S7-1500)

Description

Use the instruction "DPWR_DAT" to transfer the data at the parameters RECORD consistently to the addressed module of the central module or of the DP standard slave/PROFINET IO device and if necessary to the process image (i.e. if the affected address range of the DP standard slaves is projected as consistency range in a process image).

You require "DPWR_DAT" because you can only write a maximum of four continuous bytes using the transfer commands that access the I/O or process image output. If required, also write consistent data via the process image output. Refer to the related documentation to find out if your CPU supports this functionality. Do not use both possibilities concurrently when writing consistent data: Either use "DPWR_DAT" or write via the process image output table. For additional information on consistent data of a DP standard slave/PROFINET IO device, refer to Section "Data consistency (Page 3442)". If the DP standard slave has a modular design, you only access one module of the DP slave.

⚠ CAUTION
I/O access When using "DPWR_DAT", avoid accessing I/O areas that have process image partitions with OB6x connections (isochronous mode interrupts) assigned to them.

If necessary, the instruction "DPRD_DAT" can also be used for a data area of 1 byte or larger. For information on the maximum length of the data, refer to the documentation of your CPU (e.g. 64 bytes for an S7-1214).

- Use the LADDR parameter to select the DP standard slave / PROFINET IO device. If an access error occurs on the addressed module, the error code 8090 is output.
- Use the RECORD parameter to define the source range of the data to be written:
  - The source range has to be at least as long as the outputs of the selected module. Only the outputs are transferred, the other bytes are not considered. If the source area at parameter RECORD is longer than the outputs of the configured module, only the data up to the maximum length of the outputs is transferred. If the source area at parameter RECORD is shorter than the outputs of the configured module, the error code 80B1 is output.
  - The following data types can be used: BOOL, BYTE, CHAR, WCHAR, WORD, LWORD, DWORD, INT, UINT, USINT, SINT, LINT, ULINT, DINT, UDINT, REAL, LREAL. The use of these data types in a data structure of the type ARRAY or STRUCT is also permissible.
  - The STRING and WSTRING data types are not supported.
  - PLC data types (UDT) and system data types are supported.

The data transfer is synchronous. Once the instruction is finished, the write process is completed.

Parameters

The following table shows the parameters of the instruction "DPWR_DAT":

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Hardware ID of the module to which the data is to be written. The hardware ID can be found in the properties of the module in the device view or system constants.
RECORD	Input	VARIANT	I, Q, M, D, L	Source area for the user data to be written.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	• You have not configured a module for the specified hardware identifier or • You have ignored the restriction concerning the length of consistent data, or • you have not specified a hardware identifier at parameter LADDR .
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.
8093	No DP module /PROFINET IO device to which you can write consistent data exists at the HW ID specified in LADDR . This error code also occurs if the DP standard slave / PROFINET IO device addressed via LADDR does not have outputs.
80A1	Access error detected while I/O devices were being accessed.
80B1	The length of the specified source range at the RECORD parameter is shorter than the outputs of the configured DP standard slave / PROFINET IO device.
80C1	The data of the previous write job have not yet been processed by the DP standard slave / PROFINET IO device.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Parameter STATUS (Page 2276)

iDevice / iSlave (S7-1200, S7-1500)

RCVREC: Receive data record (S7-1200, S7-1500)

Description

An I-device can receive a data record from a superordinate controller. The receipt takes place in the user program with the instruction "RCVREC" (receive record).

The instruction has the following operating modes:

- Check whether the I-device has a request for a data record receipt.
- Make the data record available to the output parameters.
- Sending an answer to the superordinate controller.

You can determine the operating mode executed by the instruction using the input parameter MODE (see below).

The I-device must be in the RUN or STARTUP mode.

With MLEN, you specify the maximum number of bytes you want to receive. Select the destination area RECORD at least MLEN bytes long.

If a data record was received (MODE=1 or MODE=2), the output parameter NEW will indicate that the data record was stored in RECORD. Note that RECORD has a sufficient length. The output parameter LEN contains the actual length of the data record received in bytes.

Set CODE1 and CODE2 to zero for the positive answer to the superordinate controller. If the received data record is rejected, enter Error Code 1 in CODE1 and Error Code 2 in CODE2 of the negative answer at the superordinate controller.

Note

If the I-device has received a request for a data record receipt, you must recognize the delivery of this request within a certain duration. After recognition, you must send an answer to the superordinate controller within this time period. Otherwise the I-device will experience a timeout error which causes the operating system of the I-device to send a negative answer to the superordinate controller. You can find the value for the duration in the description of PRVREC (Page 2344).

The output parameter STATUS receives the error information after the occurrence of an error.

Operating modes

You can determine the operating mode of the instruction "RCVREC" with the input parameter MODE. This step will be explained in the following table.

MODE	Meaning
0	Check whether a request for a data record receipt exists If a data record from a higher-level controller exists on the I-device, the instruction will only write the output parameters NEW, SLOT, SUBSLOT, INDEX and LEN. If you call the instruction several times with MODE=0, then the output parameter will only refer to one and the same request.
1	Receiving a data record for any subslot of the I-device If a data record from a superordinate controller exists on the I-device for any subslot of the I-device, the instruction will write the output parameter and transfer the data record to the parameter RECORD.
2	Receiving a data record for a specific subslot of the I-device If a data record from a superordinate controller exists on the I-device for a specific subslot of the I-device, the instruction will write the output parameter and transfer the data record to the parameter RECORD.
3	Sending a positive answer to the superordinate controller The instruction checks the request of the superordinate controller to receive a data record, accepts the existing data record and sends a positive acknowledgment to the superordinate controller.
4	Sending a negative answer to the superordinate controller The instruction checks the request of the superordinate controller to receive a data record, rejects the existing data record and sends a negative acknowledgment to the superordinate controller. Enter the reason for the rejection in the input parameters CODE1 and CODE2.

Note

After the receipt of a data record (NEW=1) you must call "RCVREC" twice to ensure complete processing. You must do this in the following order:

- First call with MODE=1 or MODE=2 (must be completed within one second after request was received, otherwise the instruction of the requesting IO controller is acknowledged as negative, which means parameter STATUS (field element STATUS[3] corresponds to Error_code_1): C2 (Resource busy))
- Second call with MODE=3 or MODE=4 (must be completed within 10 s after call with MODE=1 or MODE=2 . Once the wait time has expired, the higher-level IO controller receives the following error in the STATUS parameter of its requesting instruction (RDREC) : STATUS parameter (field element STATUS[3] corresponds to Error_code_1): A7 (Application busy))

Parameters

The following table shows the parameters of the "RCVREC" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	HW_SUB-MODULE	I, Q, M, D, L or constant	Subslot in the transfer area of the I-device for the data record to be received (only relevant for MODE=2). The high word is always set to zero.

Parameter	Declaration	Data type	Memory area	Description
MLEN	Input	INT*	I, Q, M, D, L or constant	Maximum length of the data record to be received in bytes
CODE1	Input	BYTE	I, Q, M, D, L or constant	"0" (for MODE=3) and/or Error Code 1 (for MODE=4)
CODE2	Input	BYTE	I, Q, M, D, L or constant	"0" (for MODE=3) and/or Error Code 2 (for MODE=4)
NEW	Output	BOOL	I, Q, M, D, L	• MODE=0: New data record was received • MODE=1 or 2: Data record was transferred to RECORD
STATUS	Output	DWORD	I, Q, M, D, L	Error information
SLOT	Output	HW_SUB-MODULE	I, Q, M, D, L	Identical to F_ID
SUBSLOT	Output	HW_SUB-MODULE	I, Q, M, D, L	Identical to F_ID
INDEX	Output	UINT	I, Q, M, D, L	Number of the data record received
LEN	Output	UINT	I, Q, M, D, L	Length of the data record received
RECORD	InOut	VARIANT	I, Q, M, D, L	Destination area for the data record received.
* Use the data type UINT in the STL programming language.				

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

For interpretation of the STATUS parameter, refer to section: Parameter STATUS (Page 2276)

PRVREC: Make data record available (S7-1200, S7-1500)

Description

An I-device can receive a request from a superordinate controller to make a data record available. The I-device makes the data record available in the user program with the instruction "PRVREC" (provide record).

The instruction has the following operating modes:

- Check whether the I-device has a request for making a data record available.
- Transfer the requested data record to the superordinate controller.
- Sending an answer to the superordinate controller.

You can determine the operating mode executed by the instruction using the input parameter MODE (see below).

The instruction PRVREC can only be executed in the I-device in the RUN or STARTUP mode.

If the I-device is in STOP mode, PRVREC cannot be executed and the higher-level IO controller receives the following error in the STATUS of its requesting instruction (RDREC):

STATUS parameter (field element STATUS[3] corresponds to Error_code_1): B5 (State conflict).

When the I-device is in STARTUP or RUN, the I-device has 1 second after receiving the read request to respond with PRVREC(MODE=1, …) or PRVREC(MODE=2, …). Otherwise the error C2 (Resource busy) is signaled to the higher-level IO controller.

After calling PRVREC (Mode=1/2), the I-device has 10 seconds to conclude the instruction with PRVREC (MODE=3) or PRVREC (MODE=4). Otherwise the error A7 (Application busy) is signaled to the higher-level IO controller.

Enter the maximum number of bytes the data record to be sent should have with LEN. Select the destination area RECORD at least LEN bytes long.

If a request to make a data record available exists, (MODE=0), the output parameter NEW will be set to TRUE.

If the request for making a data record available is accepted, write RECORD for the positive answer to the superordinate controller with the requested data record and write zero for CODE1 and CODE2. If the request for making a data record available is to be rejected, enter the negative answer to the superordinate controller in Error Code 1 of the CODE1 and in Error Code 2 of theCODE2.

The output parameter STATUS receives the error information after the occurrence of an error.

Operating modes

You can determine the operating mode of the instruction "PRVREC" with the input parameter MODE. This step will be explained in the following table.

MODE	Meaning
0	Check whether a request for making a data record available exists If a request from a higher-level controller for making a data record available exists on the I-device, the instruction will only write the output parameters NEW, SLOT, SUBSLOT, INDEX and RLEN. If you call the instruction several times with MODE=0, then the output parameter will only refer to one and the same request.
1	Receiving a request for making a data record available for any subslot of the I-device If such a request from a superordinate controller for any subslot of the I-device exists on the I-device, the instruction will write the output parameter.
2	Receiving a request for making a data record available for a specific subslot of the I-device If such a request from a superordinate controller for a specific subslot of the I-device exists on the I-device, the instruction will write the output parameter.
3	Make the data record available and send a positive answer to the superordinate controller The instruction checks the request of the superordinate controller to make a data record available, makes the request data record available to RECORD and sends a positive acknowledgement to the superordinate controller.
4	Sending a negative answer to the superordinate controller The instruction checks the request of the superordinate controller to make a data record available, rejects this request and sends a negative acknowledgement to the superordinate controller. Enter the reason for the rejection in the input parameters CODE1 and CODE2.

Note

After the receipt of a request (NEW=1) you must call the instruction twice to ensure complete processing. You must do this in the following order:

- First call with MODE=1 or MODE=2 (must be completed within one second after request was received, otherwise the instruction of the requesting IO controller is acknowledged as negative, which means parameter STATUS (field element STATUS[3] corresponds to Error_code_1): C2 (Resource busy))
- Second call with MODE=3 or MODE=4 (must be completed within 10 s after call with MODE=1 or MODE=2 . Once the wait time has expired, the higher-level IO controller receives the following error in the STATUS parameter of its requesting instruction (RDREC) : STATUS parameter (field element STATUS[3] corresponds to Error_code_1): A7 (Application busy))

Tip

Program the first and second call directly one after the other.

Background: After the first call of the instruction with Mode 1 or Mode 2, the CPU cannot process an additional instance of the instruction, which means no additional data record job. Only after the second call was edited with Mode 3 or Mode 4, can the CPU process a different data record job. If an additional instance of the instruction is called after the first call with Mode 1 or Mode 2, the status returns the value 0xC080C300 (= error, Resource unavailable).

Example: You are using the function Asset Management data record as well as the PROFIenergy function in the I-device user program. Asset Management uses the instruction PRVREC and PROFIenergy uses the instruction PE_I_DEV. PE_I_DEV internally also uses the instructions PRVREC and RCVREC.

Parameters

The following table shows the parameters of the "PRVREC" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	HW_SUB-MODULE	I, Q, M, D, L or constant	Subslot in the transfer area of the I-device for the data record to be sent (only relevant for MODE=2). The high word is always set to zero. When you call PRVREC to supply an asset management data record to the IO controller (data record no. 0xF880), you must set F_ID = 0. The device is addressed with F_ID = 0. (Only relevant for MODE=2. Please note: When calling with MODE=0, the value 0 is returned for INDEX=0xF880 by PRVREC for SLOT and SUBSLOT.)
CODE1	Input	BYTE	I, Q, M, D, L or constant	"0" (for MODE=3) and/or Error Code 1 (for MODE=4)

Parameter	Declaration	Data type	Memory area	Description
CODE2	Input	BYTE	I, Q, M, D, L or constant	"0" (for MODE=3) and/or Error Code 2 (for MODE=4)
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the data record to be sent in bytes
NEW	Output	BOOL	I, Q, M, D, L	The new data record was requested by the superordinate controller.
STATUS	Output	DWORD	I, Q, M, D, L	Error information
SLOT	Output	HW_SUB-MODULE	I, Q, M, D, L	Identical to F_ID
SUBSLOT	Output	HW_SUB-MODULE	I, Q, M, D, L	Identical to F_ID
INDEX	Output	UINT	I, Q, M, D, L	Number of the data record to be sent
RLEN	Output	UINT	I, Q, M, D, L	Length of the data record to be sent
RECORD	InOut	VARIANT	I, Q, M, D, L	Data record made available

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

For interpretation of the STATUS parameter, refer to section: Parameter STATUS (Page 2276)

PROFIBUS

DPSYC_FR: Synchronize DP slaves / Freeze inputs (S7-1500)

Description

You use the instruction to synchronize one or more groups of DP slaves.

The function involves sending one of the control commands below, or a combination of them, to the relevant groups:

- SYNC (simultaneous output and freezing of output states on the DP slaves)
- UNSYNC (cancels the SYNC control command)
- FREEZE (freeze the input states on the DP slaves and read in the frozen inputs)
- UNFREEZE (cancels the FREEZE control command)

Before you send the control commands listed above, you must assign the DP slaves to groups per configuration. You must know which DP slave is assigned to which group, with which number and know the reactions of the various groups to SYNC/FREEZE.

Note

Note that the control commands SYNC and FREEZE also remain valid when you perform a warm/cold restart.

Please note also that you may initiate only one SYNC/UNSYNC job or only one FREEZE/UNFREEZE job at a time.

Functional description

"DPSYC_FR" is an asynchronous instruction. Processing takes place across several calls. You start the job by calling "DPSYC_FR" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596).

Identification of a job

If you have triggered a SYNC/FREEZE job and called "DPSYC_FR" again before the first job was completed, the response of the instruction depends on whether the new call is for the same job. If the input parameters LADDR, GROUP and MODE match, the call is interpreted as a follow-on call.

Writing outputs of DP modules

The writing of outputs of DP modules is triggered as follows:

- By transfer commands to the DP I/O
- By writing the process image of the outputs to the modules (by the operating system at the end of OB 1 or by calling the instruction "UPDAT_PO (Page 2200)")
- Calling the "DPWR_DAT (Page 2340)" instruction.

In normal operation, the DP master transfers the output bytes cyclically (within the cycle of the PROFIBUS DP bus) to the outputs of the DP slaves.

If you want to have certain output data (possibly distributed over several slaves) applied to the outputs to the process at exactly the same time, you can send the SYNC control command to the relevant DP master using the "DPSYC_FR" instruction.

What are the effects of SYNC?

With the SYNC control command, the DP slaves of the selected groups are switched to Sync mode. The DP master transfers the current output data and instructs the DP slaves involved to freeze their outputs. With the following output message frames, the DP slaves enter the output data in an internal buffer and the state of the outputs remains unchanged.

Following each SYNC control command, the DP slaves of the selected groups apply the output data of their internal buffer to the outputs on the process.

The outputs are only updated cyclically again when you send the UNSYNC control command using the "DPSYC_FR" instruction.

Note

If the DP slaves of the selected group(s) are not currently connected to the network or have failed when the control command was sent, they will not be switched to SYNC mode. This information will not be communicated in the return value of the instruction.

Reading input data of DP modules

The input data of the DP modules are read as follows:

- Using load commands to the DP I/O
- When the process image of the inputs is updated (by the operating system at the start of OB 1 or by calling the "UPDAT_PI (Page 2197)" instruction)
- By calling the "DPRD_DAT (Page 2338)" instruction.

In normal operation, the DP master receives this input data cyclically (within the cycle of the PROFIBUS DP bus) from its DP slaves and makes them available to the CPU.

If you want to have certain input data (possibly distributed over several slaves) to be read from the process at exactly the same time, send the FREEZE control command to the relevant DP master using the "DPSYC_FR" instruction.

What are the effects of FREEZE?

With the FREEZE control command, the DP slaves involved are switched to Freeze mode. The DP master instructs the DP slaves involved to freeze the current state of the inputs. It then transfers the frozen data to the input area of the CPU.

Following each FREEZE control command, the DP slaves freeze the state of their inputs again.

The DP master receives the current state of the inputs cyclically again after you have sent the UNFREEZE control command with the "DPSYC_FR" instruction.

Note

If the DP slaves of the selected group(s) are not currently connected to the network or have failed when the control command has been sent, they will not be switched to FREEZE mode. This information will not be communicated in the return value of the instruction.

Data consistency

Because the DPSYC_FR functions are asynchronous and can be interrupted by higher priority classes, you should make sure that the process images are consistent with the actual inputs and outputs of the I/O when using the "DPSYC_FR" instruction.

This is guaranteed if you comply with the following consistency rules:

- Define suitable process image partitions for the "SYNC outputs" and the "FREEZE inputs" (only possible on the S7-400). Call the "UPDAT_PO (Page 2200)" instruction immediately before each first call of a SYNC job. Call the "UPDAT_PI (Page 2197)" instruction immediately after the respective last call of a FREEZE job.
- As an alternative: Use only direct I/O access for outputs involved in a SYNC job and for inputs involved in a FREEZE job. Do not write to these outputs when a SYNC job is active and do not read in these inputs when a FREEZE job is active.

Use of DPWR_DAT and DPRD_DAT

If you use the "DPWR_DAT (Page 2340)" instruction, it must be complete before you send a SYNC job for the outputs involved.

If you use the "DPRD_DAT (Page 2338)" instruction, it must be complete before you send a FREEZE job for the inputs involved.

Startup and "DPSYC_FR"

The user alone must take responsibility for sending the SYNC and FREEZE control commands in the startup OBs.

If you want the outputs of one or more groups to be in SYNC mode when the user program starts, you must initialize these outputs during startup and completely execute the "DPSYC_FR" instruction with the SYNC control command.

If you want the inputs of one or more groups to be in FREEZE mode when the user program starts, you must execute the "DPSYC_FR" instruction with the FREEZE control command completely for these inputs during startup.

Parameters

The following table shows the parameters of the "DPSYC_FR" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Level-triggered control parameter REQ = 1: Triggering of the SYNC/FREEZE job
LADDR	Input	HW_INTERFACE	I, Q, M, D, L or constant	Hardware identifier of the DP master interface The number can be taken from the properties of the DP master interface in the Network view or from the "System constants" tab of the standard tag table.
GROUP	Input	BYTE	I, Q, M, D, L or constant	Group selection Bit 0 = 1: group 1 selected Bit 1 = 1: group 2 selected : Bit 7 = 1: group 8 selected You can select several groups per job. The value B#16#0 is invalid.

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID (coding complies with EN 50 170 Volume 2, PROFIBUS) Bit 0: reserved (value 0) Bit 1: reserved (value 0) Bit 2: • = 1: UNFREEZE will be executed • = 0: no meaning Bit 3: • = 1: FREEZE will be executed • = 0: no meaning Bit 4: • = 1: UNSYNC will be executed • = 0: no meaning Bit 5: • = 1: SYNC will be executed • = 0: no meaning Bit 6: reserved (value 0) Bit 7: reserved (value 0) Possible values: • with exactly one ID per job: – B#16#04 (UNFREEZE) – B#16#08 (FREEZE) – B#16#10 (UNSYNC) – B#16#20 (SYNC) • with more than one ID per job: – B#16#14 (UNSYNC, UNFREEZE) – B#16#18 (UNSYNC, FREEZE) – B#16#24 (SYNC, UNFREEZE) – B#16#28 (SYNC, FREEZE)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. Make sure that you evaluate RET_VAL each time the block has been executed.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The SYNC/FREEZE job is not yet complete.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. For a description of this error information, refer to STATUS[3], STATUS (Page 2276) parameter.

Error code* (W#16#...)	Explanation
0000	Job completed without error.
7000	First call with REQ= 0: The job specified with LADDR , GROUP and MODE is not active; BUSY has the value "0".
7001	First call with REQ= 1. The job specified with LADDR , GROUP and MODE was triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant). The activated SYNC/FREEZE job is still running; BUSY has the value 1.
8090	The module selected with LADDR is not a DP master.
8093	This instruction is not permitted for the module selected with LADDR (configuration or version of the DP master).
8094	GROUP parameter is incorrect
8095	MODE parameter is incorrect
80A4	Communication on PROFIBUS disrupted.
80B0	The group selected with GROUP is not configured.
80B1	The group selected with GROUP is not assigned to this CPU.
80B2	The SYNC job specified with MODE is not permitted on the group selected with GROUP.
80B3	The FREEZE job specified with MODE is not permitted on the group selected with GROUP .
80C2	Temporary shortage of resources on the DP master: The DP master is currently processing the maximum number of jobs for a CPU.
80C3	This SYNC/UNSYNC job cannot be activated at present because only one SYNC/UNSYNC job can be triggered at a time. Check your user program.
80C4	This FREEZE/UNFREEZE job cannot be activated at present because only one FREEZE/UNFREEZE job can be triggered at a time. Check your user program.
80C5	Short circuit directly at DP interface
80C6	Job aborted due to I/O disconnection by CPU
80C7	Job aborted due to warm or cold restart on the DP master
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

DPNRM_DG: Read diagnostics data from a DP slave (S7-1200, S7-1500)

Description

You use the "DPNRM_DG" instruction to read the current diagnostic data of a DP slave in the form specified in EN 50 170 Volume 2, PROFIBUS.

Refer to the following table for the basic structure of the slave diagnostic data and to the manuals of the DP slaves for further information.

Byte	Meaning
0	Station status 1
1	Station status 2
2	Station status 3
3	Master station number
4	Vendor ID (high byte)
5	Vendor ID (low byte)
6 ...	Additional slave-specific diagnostic information

The data that has been read is entered in the destination area indicated by RECORD following without error data transfer. You start the read process by assigning the value "1" to the REQ input parameter when the "DPNRM_DG" instruction is called.

Functional description

The read process is executed asynchronously. It can span several calls. The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596)

Parameters

The following table shows the parameters of the "DPNRM_DG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_DPSLAVE	I, Q, M, D, L or constant	Hardware identifier of the DP slave Note: The identifier must be specified in hexadecimal form. For example, hardware identifier 1022 means: LADDR:=W#16#3FE.
RET_VAL	Return	DINT, INT, LREAL, REAL	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error has occurred, the length of the data actually transferred is entered in RET_VAL .

Parameter	Declaration	Data type	Memory area	Description
RECORD	Output	VARIANT	I, Q, M, D, L	Destination area for the diagnostic data that were read. Only the data type BYTE is valid. The minimum length of the data record to be read or the destination area is 6. The maximum length of the data record to be read is 240. For standard slaves, which have a diagnostic data length of more than 240 bytes up to a maximum of 244 bytes, the first 240 bytes are transferred to the destination area. Then, the corresponding overflow bit is set in the data.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

You can find information on data type conversion in the different programming languages under "Overview of data type conversions (Page 555)".

Parameter RECORD

The CPU evaluates the actual length of the read diagnostic data:

If the length specified for RECORD

- is less than the number of data bytes supplied, the data is discarded. The corresponding error code is entered in RET_VAL.
- is greater than or equal to the number of data bytes supplied, the data is accepted in the destination area and the actual length is entered in RET_VAL as a positive value.

Note

You must ensure that the actual parameters of RECORD correspond in all calls belonging to a job.

A job is uniquely identified by the LADDR input parameter.

Standard slaves with more than 240 bytes of diagnostic data

With standard slaves on which the number of standard diagnostic data is between 241 and 244 bytes, note the following points:

If the length specified for RECORD

- is less than 240 bytes, the data is discarded and the corresponding error code is entered in RET_VAL.
- is greater than or equal to 240 bytes, the first 240 bytes of the standard diagnostic data is transferred to the destination area and the overflow bit is set in the data.

Parameter RET_VAL

- If an error occurred while the function was being executed, the return value contains an error code.
- If no errors have occurred during data transfer, RET_VAL contains the length of the data read in bytes as a positive number.

Note

The amount of data read in a DP slave depends on its diagnostics status.

For the evaluation of the error information of the RET_VAL parameter, refer to the following table.

Error code (W#16#...)	Explanation	Restriction
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".	-
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Address specified at parameter LADDR is invalid.	-
8093	This instruction is not permitted for the module selected by means of LADDR and IOID.	-
80A2	• DP protocol error at layer 2 (for example, slave failure or bus problems) • For ET 200S, data record cannot be read in DPV0 mode.	Distributed I/O
80A3	DP protocol error with user interface/user	Distributed I/O
80A4	Communication on PROFIBUS disrupted.	Distributed I/O
80B0	• Instruction not possible for module type. • The module does not recognize the data record. • Data record number 241 not permitted. • With "WR_REC (Page 2232)", data records 0 and 1 are not permitted.	-
80B1	The length specified in parameter RECORD is incorrect.	specified length < data record length
80B2	The configured slot is not occupied.	-
80B3	Actual module type does not match specified module type	-
80C0	There are no diagnostic data available.	-
80C1	The data of the previous write job on the module for the same data record has not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	-

Error code (W#16#...)	Explanation	Restriction
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available.	Distributed I/O
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)	-
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 7698)

Overview of data type conversions (Page 555)

DP_TOPOL: Determine topology for DP master system (S7-1500)

Description

You use the instruction to trigger the topology determination for a selected DP master system. Calling the instruction will address all diagnostics repeaters on a DP master system.

Note

The topology can only be determined for one DP master system at a time.

Topology determination is the prerequisite for the detailed display of the error location if line errors occur. After configuration and after every change to the physical configuration of a DP master system, you must repeat the topology determination with "DP_TOPOL".

Changes to the physical configuration are:

- Changes in line lengths
- Adding or removing stations or components with repeater function
- Changing station addresses

If an error is reported by a diagnostics repeater, "DP_TOPOL" writes the DPR and DPRI outputs for the duration of one "DP_TOPOL pass. If errors are reported by multiple diagnostics repeaters of the selected DP master system, "DP_TOPOL" writes information on the first error-reporting diagnostics repeater to DPR and DPRI . You can read out the entire diagnostic information on the programming device or using the "DPNRM_DG (Page 2353)" instruction. If no diagnostics repeater reports an error, the DPR and DPRI outputs have the value "0".

If you want to repeat a topology determination after an error occurs, you must first reset "DP_TOPOL". This is done by calling "DP_TOPOL" with REQ=0 and R=1.

Functional description

"DP_TOPOL" is an asynchronous instruction. Processing takes place across several calls. You start determining the bus topology by calling "DP_TOPOL" with REQ=1 . If you want to cancel the process, call "DP_TOPOL" with R=1 .

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596)

Note

Determining the topology may take several minutes.

Identification of a job

The input parameter DP_ID uniquely specifies a job.

If you call "DP_TOPOL" again before the topology is discovered, the manner in which the instruction reacts depends on whether the new call involves the same job: If the DP_ID parameter matches a job that has not yet been completed, then the call is interpreted as a follow-up call. The value W#16#7002 is entered in RET_VAL. On the other hand, if another job is involved, the CPU rejects it.

Parameters

The following table shows the parameters of the "DP_TOPOL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ=1: Trigger topology determination
R	Input	BOOL	I, Q, M, D, L or constant	R=1: Cancel topology determination
DP_ID	Input	HW_IOSYSTEM	I, Q, M, D, L or constant	DP master system ID of those master systems for which the topology will be determined
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: Topology determination is not yet complete.
DPR	Output	BYTE	I, Q, M, D, L	PROFIBUS address of the diagnostics repeater reporting the error.
DPRI	Output	BYTE	I, Q, M, D, L	Measuring segment diagnostics repeater reporting the error: • Bit 0 = 1: Temporary faults on segment DP2 • Bit 1 = 1: Permanent faults on segment DP2 • Bit 4 = 1: Temporary faults on segment DP3 • Bit 5 = 1: Permanent faults on segment DP3

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

The "real" error information (error codes W#16#8xyz) can be divided into two classes:

- Temporary errors (error codes W#16#80A2 to 80A4, 80C3, 80C5):
  It is possible that this type of error is eliminated without user action. It is advisable to call "DP_TOPOL" again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#8082, 80B0, 80B2):
  With this error type, you must resolve the error. A new call of "DP_TOPOL" is required once you have eliminated the error. Example of a permanent error: The DP master/CPU does not support this service. (W#16#80B0).

Error code* (W#16#...)	Explanation
0000	Job completed without error.
7000	First call with REQ=0. Topology determination is not triggered. BUSY has the value "0".
7001	First call with REQ=1. The request to execute topology determination was sent. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Topology determination is not yet complete. BUSY has the value "1".
7010	You have attempted to cancel topology determination. But there is no running job with the specified DP_ID. BUSY has the value "0".
7011	First call with R=1. The cancellation of the topology determination was triggered. BUSY has the value "1".
7012	Intermediate call: The cancellation of the topology determination is not yet complete. BUSY has the value "1".
7013	Final call: The topology determination was cancelled. BUSY has the value "0".
8082	No DP master system is configured with the specified DP_ID.
80A2	Error during topology determination; for more detailed information, refer to output parameters DPR and DPRI.
80A3	Error during topology determination: Monitoring time has elapsed (timeout).
80A4	Communication on PROFIBUS disrupted.
80B0	The DP master/CPU does not support this service.
80B2	Error during topology determination: No diagnostics repeater was found at the selected DP master system.
80C3	Resources required are currently in use. Possible cause: You have initiated a second topology determination (only one topology determination is permitted at one time).
80C5	The DP master system is currently not available.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

ASi (S7-1500)

ASI_CTRL: Control AS-i master behavior (S7-1500)

Description of ASI_CTRL (S7-1500)

Description

Using the instruction "ASI_CTRL", the behavior of the AS-i master is controlled from the user program of the PLC. The instruction processes the command protocol automatically. It also enables parameter assignment on SIMATIC AS-i masters and reading out information data. The functions and operation of the command interface are detailed in the manual for the AS-i master.

Both centrally inserted AS-i masters and distributed AS-i masters via PROFIBUS DP are supported. Combinations with PROFINET IO (e.g. IE/PB Link PN IO) are also possible.

The schematic diagram below shows the functions of the "ASI_CTRL" instruction:

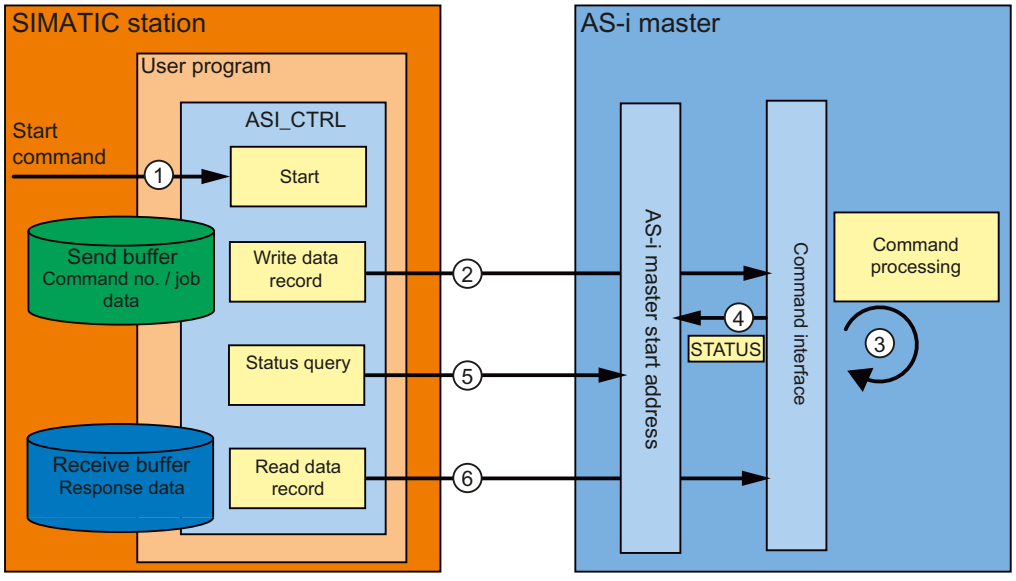

① Start of processing at the REQ parameter.

② The program sends the required command to the AS-i master via the instruction "RDREC".

③ The AS-i master processes the command.

④ The current status of the AS-i master is stored in the input area of the binary data (logical start address).

⑤ The instruction "ASI_CTRL" cyclically queries and evaluates the 4 status bits.

⑥ Once command processing is complete, the command job is completed with "RDREC". Depending on the command, the data field of "RDREC" may contain the response data for the command or additional status information.

Dependencies between instruction versions

Version V1.3 of instruction "ASI_CTRL" requires version V1.1 of instruction "WRREC".

Version V1.2 of instruction "ASI_CTRL" requires version V1.1 of instruction "WRREC".

Differences in the command call between IE/ AS-i Link and DP/AS-i Links

There are significant differences in how a controller exchanges commands with an AS-i master.

- **IE/AS-i Link** (PROFINET) used the data record interface. The different commands are called with either "Write data record" ("WRREC" instruction) or "Read data record" ("RDREC" instruction) by various data record numbers.
- **DP/AS-i Links** (PROFIBUS) use the command interface. All commands are called by data record number 2 with "Write data record" ("WRREC" instruction) plus "Read data record" ("RDREC" instruction). The type of command is defined by the data content in the write job.

Operation of the "ASI_CTRL" instruction

The instruction "ASI_CTRL" is an asynchronous function block. Processing takes place across several calls.

- A job is started with REQ = TRUE.
- The job status is displayed via the BUSY output parameters and the two central bytes of the output parameter STATUS.
- The BUSY parameter is set during job processing. Upon the first call, STATUS is assigned the value 00700100_H. Upon all subsequent calls for this job, it is assigned the value 00700200_H. When the job is complete, the result is output at the parameters DONE or ERROR.
  - If no errors have occurred, DONE is set. For jobs with response data from the AS-i master, this data is provided in the specified receive buffer. In such cases, the STATUS parameter also displays the volume of data supplied in bytes. For jobs without response data, the value 00000000_H is entered in STATUS.
  - If an error occurs during job processing, ERROR is set. The content of the receive buffer is invalid in such a case. An error code is entered in the STATUS parameter for a more detailed description of the error which has occurred.

Number of command calls

If you use the instruction "ASI_CTRL" to send commands, you may not simultaneously send other commands to the same AS-i master with "RDREC (Page 2213)" or "WRREC (Page 2232)". This also applies to multiple calls of the instruction for the same AS-i master.

The instruction "ASI_CTRL" cannot be run with interruptions. Calls therefore must not be programmed in program priority classes which interrupt each other (for example with calls in OB 1 and in OB 35).

Parameters

The following table shows the parameters of the "ASI_CTRL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, constant	REQ = TRUE starts a new job unless a job is already in progress. No edge evaluation takes place.
LADDR	Input	HW_IO	I, Q, M, D, L, constant	Hardware identifier of the AS-i master. You can obtain the address from the module properties.
SD	Input	VARIANT	I, Q, M, D, L	Send buffer The parameter refers to a memory area in which the command is to be specified (see "ASi commands (Page 2363)").
RD	Input	VARIANT	I, Q, M, D, L	Receive buffer This buffer is only relevant for commands which deliver answer data. The parameter refers to a memory area in which a command response is stored (see "ASi commands (Page 2363)").
DONE	Output	BOOL	I, Q, M, D, L	DONE = TRUE: Job completed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = TRUE: Job in process.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = TRUE: Job aborted with error.
STATUS	Output	DWORD	I, Q, M, D, L	Job status / error code See the "STATUS parameter" description.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

LADDR, SD and RD parameters

The parameters LADDR, SD and RD must not be changed in any block cycle when a job is in progress: they must remain constant.

Parameter STATUS

The following table shows the possible STATUS displays dependent upon DONE and ERROR.

DONE	BUSY	ERROR	STATUS	Meaning
0	0	0	00700000_H	First call with REQ = FALSE; no active job.
0	1	0	00700100_H	First call with REQ = TRUE; job has started.
0	1	0	00700200_H	Subsequent call (REQ irrelevant); job is still in progress.
1	0	0	00000000_H	Job completed without errors. No response data.
1	0	0	$0000xx00_H$	Job completed without errors. Number of xx bytes of response data.
0	0	1	$C0818400_H$	Data type of formal operand RD invalid.

DONE	BUSY	ERROR	STATUS	Meaning
0	0	1	C0818500 H	Error in communication with AS-i master (incorrect address configured at the LADDR parameter).
0	0	1	C0838100 H	Incorrect AS-i slave address.
0	0	1	C0838200 H	AS-i slave is not enabled (not in LAS).
0	0	1	C0838300 H	Error at the AS interface (the SD parameter may have been set too small).
0	0	1	C0838400 H	The command is not valid with the current AS-i master status.
0	0	1	C0838500 H	There is an AS-i slave with the address "0".
0	0	1	C0838600 H	The AS-i slave has invalid configuration data (I/O or ID codes).
0	0	1	C083A100 H	The AS-i slave addressed has not been found at the AS interface.
0	0	1	C083A200 H	There is an AS-i slave with the address "0".
0	0	1	C083A300 H	There is already an AS-i slave with the new address at the AS interface.
0	0	1	C083A400 H	The AS-i slave address cannot be deleted.
0	0	1	C083A500 H	The AS-i slave address cannot be set.
0	0	1	C083A600 H	The AS-i slave address cannot be permanently saved.
0	0	1	C083A700 H	Error during reading of the extended ID1 code.
0	0	1	C083A800 H	The target address is not plausible (e.g. a B slave address has been used for a standard slave).
0	0	1	C083B100 H	A length error has occurred during string transfer.
0	0	1	C083B200 H	A protocol error has occurred during string transfer.
0	0	1	C083F800 H	Unknown job number or job parameter.
0	0	1	C083F900 H	The AS-i master has detected an EEPROM error.

See also

RDREC: Read data record (Page 2213)

WRREC: Write data record (Page 2232)

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 883)

Difference between synchronous and asynchronous instructions (Page 596)

ASi commands (S7-1500)

Description

The command interface allows the controller and AS-i master to exchange parameter assignment and information data.

These commands

- provide the complete functionality of the M4 master profile of the AS-i master specifications.
- enable the AS-i master to be completely configured from the controller.

Note

AS-i commands supported

Please see the manual of the relevant AS-i master for the AS-i commands supported and a detailed description.

General structure of the send buffer

The general structure of the send buffer for commands and job data is set out in the table below. The area for the command number must always be filled. The number of bytes for the job data can be found in the description of the command (see AS-i master documentation). In this case, "q" refers to the start address of the send buffer.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Meaning							
q + 0	Command number							
q + 1	Job data							
q + 2	Job data							
q + ...	Job data							

General structure of the receive buffer

The general structure of the receive buffer for the command response data is set out in the table below. The number of bytes for the response data depends on the command. Some commands do not return response data, and therefore only require the definition of a virtual receive buffer which is not filled with data. In this case, "n" refers to the start address of the receive buffer.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Meaning							
n + 0	Response data							
n + 1	Response data							
n + 2	Response data							
n + ...	Response data							

NOTICE
Memory areas can be overwritten If the receive buffer of the "ASI_CTRL" instruction is too short, neighboring memory areas may be overwritten. The length specified in the ANY pointer of the RD parameter when the instruction "ASI_CTRL" is called is irrelevant. The required length for the receive buffer can be found in the description of the command. The following applies for command numbers 39_H, 41_H, 42_H, 43_H and 44_H: The receive buffer must be 221 bytes long (bytes 0 to 220), even if the command returns less data. Depending on the command, the highest bytes in the receive buffer may be overwritten with zero values by the AS-i master.

AS-i commands

A selection of possible AS-i commands is set out in the table below.

Name	Parameter	Return	Coding
Set_permanent_parameter (Set_Permanent_Parameter)	Slave address, parameter		00_H
Get_permanent_parameter (Get_Permanent_Parameter)	Slave address	Parameter	01_H
Write_parameter (Write_Parameter)	Slave address, parameter	Parameter echo	02_H
Read_parameter (Read_Parameter)	Slave address	Parameter value	03_H
Store_actual_parameters (Store_Actual_Parameters)			04 H
Set_configuration_data	Slave address, configuration		25 H
Get_configuration_data	Slave address	Set configuration data	26 H
Store_actual_configuration (Store_Actual_Configuration)			07_H
Get_actual_configuration	Slave address	Actual configuration data	28_H
Configure_LPS	LPS		29_H
Set_offline_mode	Mode		$0A_H$
Select_auto-program	Mode		$0B_H$
Set_mode	Mode		$0C_H$
Change_AS–iSlave_address (Change_AS–iSlave_Address)	Address1, Address2		$0D_H$
Get_AS-iSlave_status	Slave address	Error record of the AS-i slave	$0F_H$
Read_lists_and_flags		LDS, LAS, LPS, flags	30_H

Name	Parameter	Return	Coding
Get_overall_configuration		Actual configuration data, current parameters, LAS, flags	39 $_H$
Set_overall_configuration	Overall configuration		3A $_H$
Write_parameter_list	Parameter list		3C $_H$
Read_parameter_echo_list		Parameter echo list	33 $_H$
Write_CTT2_request	Slave address CTT2 string	CTT2 string	44 $_H$
Read_version_identifier		Version string	14 $_H$
Read_AS-i_slave	Slave address	ID code	17 $_H$
Read_AS-i_slave_extended_ID1	Slave address	Extended ID1 code	37 $_H$
Write_AS-iSlave_extended_ID1	Extended ID1 code		3F $_H$
Read_AS-iSlave_extended_ID2	Slave address	Extended ID2 code	38 $_H$
Read_AS-iSlave_IO	Slave address	I/O configuration	18 $_H$
Read_I/O_error_list		LPF	3E $_H$
Write_AS–i-slave_parameter_string	Slave address, parameter string		40 $_H$
Read_AS–iSlave_parameter_string	Slave address	Parameter string	41 $_H$
Read_AS-iSlave_ID_string	Slave address	ID string	42 $_H$
Read_AS-iSlave_diagnostic_string	Slave address	Diagnostic string	43 $_H$
Read_AS-i_line_error_counter			4A $_H$
Read_and_clear_AS-i_line_error_counter			4B $_H$
Read_AS-iSlave_error_counter	Slave address		4C $_H$
Read_and_clear_AS-iSlave_error_counter	Slave address		4D $_H$
Additional command for DP/ AS-i F-Link:			
AS–i_status/diag_of_F_slaves		Status/diagnostics of all AS-i safe slaves	51 $_H$

Note

Re-initializing the AS-i master command interface

Another command which is not mentioned in the table is command 77 $_H$. This call serves to re-initialize the command interface of the AS-i master. Any command which the AS-i master specified is currently processing will be terminated.

As of version V2.1.20 of DP/AS-i LINK Advanced, command 0E $_H$ is also available. This call is used to release or block the ground fault monitoring function for a line.

4.1.4.5 PROFIenergy (S7-1200, S7-1500)

Description of PROFIenergy (S7-1200, S7-1500)

PROFIenergy

PROFIenergy is a manufacturer- and device-neutral profile for energy management with PROFINET. To reduce electricity consumption during breaks in production and unplanned interruptions, it is possible to shut down equipment in a coordinated and centralized manner using PROFIenergy.

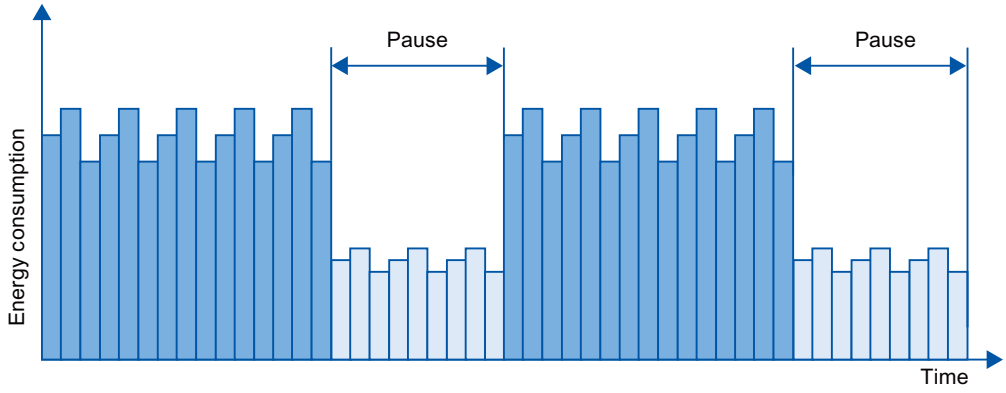

The PROFINET devices/power modules are switched off via special commands in the user program of the PROFINET IO controller. No additional hardware is required. The PROFINET devices interpret the PROFIenergy commands directly.

PROFIenergy controller (PE controller)

The PE controller is a PLC that activates or deactivates the idle state of lower-level devices. Specific production components or complete production lines are deactivated and reactivated via the user program. Commands (e.g. "Start_Pause" and "End_Pause") are sent to the lower-level device using corresponding instructions (function blocks). The commands are sent via the PROFINET communication protocol.

PROFIenergy entity (PE entity)

The PE entity receives the PROFIenergy commands of the PE controller and executes these accordingly (for example, by returning a measured value or activating an energy saving mode). Implementation of the PE entity in a PROFIenergy-capable device is device- and manufacturer-specific.

The PE entity can, for example, be executed:

- Within the proxy of a submodule: The PE commands are valid for the addressed submodule and any existing lower-level submodules.
- Within the proxy of a module: The PE commands are valid for different submodules within the module.

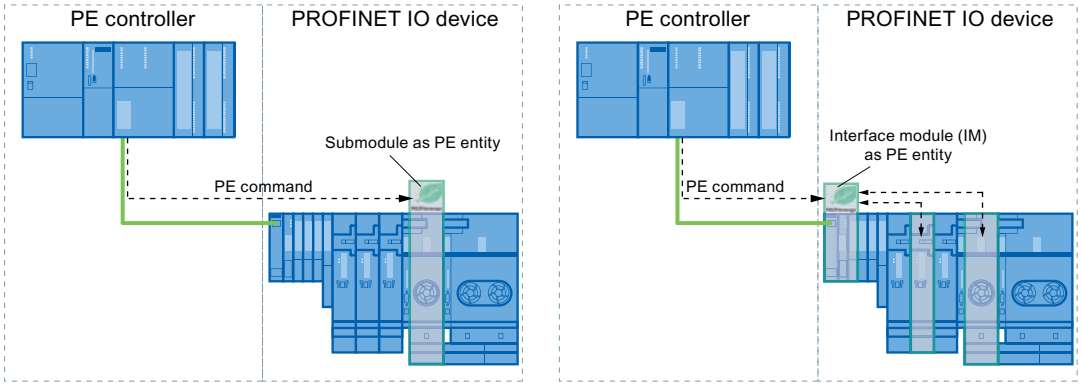

- For networked submodules without a proxy function: The PE commands in this case are only valid for the respective submodule.

PROFIenergy instructions

- Instructions for IO controller
  - The "PE_START_END (Page 2368)" instruction represents the simplest way to activate or deactivate the idle state of the PROFINET devices (PROFIenergy commands "Start_Pause" and "End_Pause"). This is done using a positive and negative signal edge in the instruction.
  - The "PE_CMD (Page 2373)" instruction allows you to transmit all PROFIenergy commands, including "Start_Pause" and "End_Pause". With other commands, you can query the current status of the PROFINET devices or the behavior during breaks, for example.
  - The instruction "PE_DS3_Write_ET200S (Page 2378)" is used to define the switching characteristic of up to 8 slots of ET 200S. The instruction is not a PROFIenergy instruction; however, it supplements the PROFIenergy functions for an ET 200S.
- Instruction for iDevices
  The "PE_I_DEV (Page 2403)" instruction allows you to implement PROFIenergy on iDevices as well. The instruction receives PROFIenergy commands on the iDevice and forwards these to the user program for processing. After processing the command, the user program calls the "PE_I_DEV (Page 2403)" instruction again in order to send the acknowledgement to the IO controller. For these replies, each command offers you a corresponding auxiliary block that supplies the response data to the instruction "PE_I_DEV (Page 2403)".

PROFIenergy commands (PE commands)

The PE controller transmits PE commands to the PE entity. The PE command can be either a control command to switch a PE entity to the energy-saving mode, or a command to read a status or measured value:

- PI commands for control
  PROFIenergy supports two control commands that can be executed using either the "PE_Start_End (Page 2368)" instruction or the "PE_CMD (Page 2373)" instruction:
  - Start_Pause: Starting a suitable energy-saving mode (PE Energy-saving mode)
  - End_Pause: Exiting the energy-saving mode (switch to PE_ready_to_operate mode)
- PI commands for reading a status or measured value
  Certain condition information can be read by the control system using the following status commands with the instruction "PE_CMD (Page 2373)":
  - PE_Identify: Reading the list of PE commands supported by the PE entity.
  - PEM_Status: Reading the currently active mode of a PE entity (e.g. PE_ready_to_operate).
  - Query_Modes: Output of an overview of all supported energy-saving modes, including the time and energy information
  - Query_Measurement: Output of the measured values of a PE entity

Example applications

Examples for use of PROFIenergy instructions can be found in the Industry Online Support in the entry "PROFIenergy - Saving Energy with SIMATIC S7 (http://support.automation.siemens.com/WW/view/en/41986454)".

See also

Service and Support (http://support.automation.siemens.com/)

IO controller (S7-1200, S7-1500)

PE_START_END: Start and exit energy-saving mode (S7-1200, S7-1500)

Description

The instruction "PE_START_END" is used to start and exit the energy-saving mode of a specified PE entity (e.g. ET 200S).

The instruction "PE_START_END" is used in the PE controller, preferably when only field devices from which energy data does not need to be read out are connected to the corresponding PE devices. Alternatively, the instruction "PE_CMD (Page 2373)" can be used to read energy data.

The energy-saving modes are configured in the user program of the PE controller. On completion of the "PE_START_END" instruction, the PE entity reports its current energy-saving mode and outputs this data at parameter PE_MODE_ID.

Write and read jobs of the "PE_START_END" instruction.

The instruction "PE_START_END" sends a PROFIenergy command internally as a write job to the PE entity using "WRREC (Page 2232)". "PE_START_END" then waits for acknowledgment from the PE entity. The acknowledgment data record is read with the instruction "RDREC (Page 2213)" every 100 milliseconds. Until acknowledgment is received from the PE entity, the function repeats the read job for 10 seconds at intervals of 100 milliseconds. The response data of the PE entity is also read with the instruction "RDREC (Page 2213)".

Below is a flowchart of the write and read jobs:

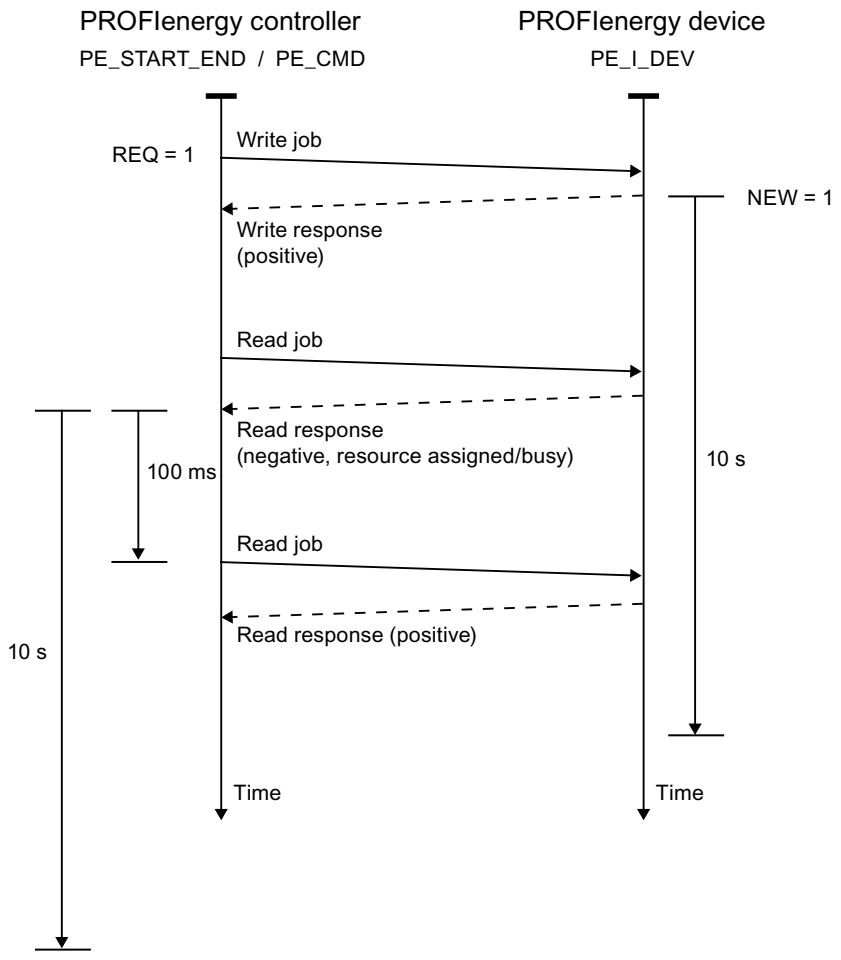

Parameter

The following table shows the parameters of the "PE_START_END" instruction:

Parameter	Declaration	Data type	Memory area	Description
START	Input	BOOL	I, Q, M, D, L or constant	Transmitting the PE command "Start_Pause" to the PE entity with the address set at parameter ID.
END	Input	BOOL	I, Q, M, D, L or constant	Transmitting the PE command "End_Pause" to the PE entity with the address set at parameter ID.
ID	Input	HW_ SUBMODULE	I, Q, M, D, L or constant	Address of the PE entity Use the hardware ID of the head module for a PROFINET IO device. Refer to the system constants of the assigned IO controller for the hardware ID. The name of the head module is composed of the name of the IO device and the suffix [Head] (example: "IO_Device_1[Head]"). If the PE entity is an I-device, use the hardware identifier of a transfer area instead.
PAUSE_TIME	Input	TIME	I, Q, M, D, L, P or constant	Planned pause duration. • Range: T#1MS to T#24D20H31M23S647MS • Start value: T#0MS
VALID	Output	BOOL	I, Q, M, D, L	PE command successfully sent.
BUSY	Output	BOOL	I, Q, M, D, L	PE command still being processed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred during processing. The error message is output at the STATUS parameter.
STATUS	Output	DWORD	I, Q, M, D, L, P	Block status / error number (see "STATUS parameter")
PE_MODE_ID	Output	BYTE	I, Q, M, D, L, P	Identification number of energy-saving mode (energy-saving level for the duration of the pause).

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter PAUSE_TIME

The parameter PAUSE_TIME is used to set the default duration of the energy-saving period at the PE entity. The PE entity checks whether the pause period is of sufficient length and whether it can be implemented. The minimum pause duration (Time_min_Pause) must be greater than the sum of the times the device needs to switch to energy-saving mode (Time_to_Pause) and to switch back to operating mode (Time_to_Operate).

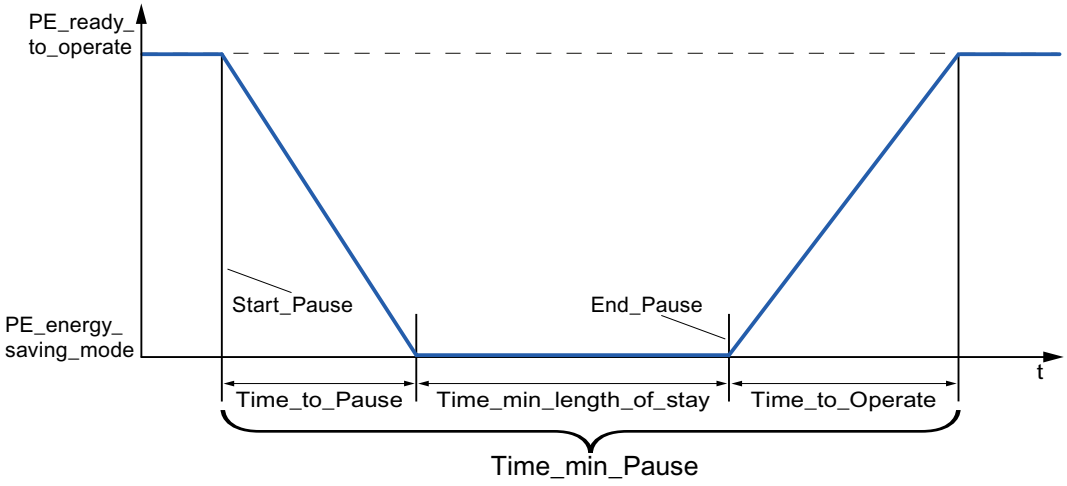

ET 200S checks whether the scheduled pause duration is greater than or equal to the minimum pause duration (PM-E_Pause_Min) saved on the ET 200S. This is fixed at 10 seconds. If a shorter pause is used, the power modules (PM-E) of the ET 200S will remain on.

The module does not switch back on automatically at the end of the pause; it remains in OFF mode until the "END" command is sent. This prevents uncoordinated switch-on, which can cause undesired peak loads.

Parameter STATUS

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error while reading the data record • B#16#**DF**: Error while writing the data record • B#16#**C0**: Error message caused by the instruction or the communication instructions "RDREC (Page 2213)" and "WRREC (Page 2232)" used internally.
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific • **FE**: DP/PNIO profiles - PROFIenergy-specific error

Field element	Name	Meaning
STATUS[3]	Error_Code_1	Error ID • When Error_Decode = **80**: – **80**: Simultaneous positive edge at the input parameters START and END. – **81**: Length conflict at the CMD_PARAM and CMD_PARAM_LEN parameters. – **82-8F**: Other error messages (reserved) • When Error_Decode = **FE**: – **01**: "Service Request ID" invalid – **02**: Incorrect "Request_Reference" – **03**: "Modifier" invalid – **04**: "Data Structure Identifier RQ" invalid – **05**: "Data Structure Identifier RS" invalid – **06**: "PE energy-saving modes" are not supported – **07**: "Response" is too long (maximum transmissible length exceeded) – **08**: "Count" invalid – **50**: No suitable "energy mode" is available. – **51**: Time value specified is not supported. – **52**: "PE_Mode_ID" invalid – **53**: No switch to "PE energy-saving mode" because the device is in operation – **54**: Function not available at this time. Wrong device configured or incorrect setup. – **55** to **FF**: Reserved
STATUS[4]	Error_Code_2	Manufacturer-specific extension of the error ID

Note

Error messages of the instructions RDREC and WRREC

The instruction "PE_START_END" uses the instructions "WRREC (Page 2232)" and "RDREC (Page 2213)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 2276) parameter for an explanation of the error codes of the "WRREC (Page 2232)" and "RDREC (Page 2213)" instructions.

See also

Description of PROFIenergy (Page 2366)

Difference between synchronous and asynchronous instructions (Page 596)

PE_CMD: Start and exit energy-saving mode / Read out status information (S7-1200, S7-1500)

Description

The instruction "PE_CMD" is used in the PE controller to start or terminate a pause in the energy-saving mode in the PE entity. Additional information and energy measurements are also read out from a PE entity using "PE_CMD".

The instruction is best used with PE controllers with assigned PE devices to which field devices are connected from which energy measurements are to be read out. If this is not the case, the instruction "PE_START_END (Page 2368)" can also be used to start and end the pauses.

Transfer of the PROFIenergy commands (PE commands)

The instruction "PE_CMD" transfers a PROFIenergy command to a PE entity.

The instruction can also be used if additional commands are to be added to the PROFIenergy profile in future. The commands which can be used with the current PROFIenergy profile are listed in the description of the CMD and CMD_MODIFIER parameters (see the "CMD and CMD_MODIFIER parameters" table).

- The individual PE commands which are transferred to the PE entity using the instruction are assigned defined "Service_Request_IDs". The Service_Request_IDs 01 to 05 and 16 are assigned at the CMD parameter.
- The two PE commands 04 (Query_Modes) and 16 (Query_Measurement) are defined in more detail with the CMD_MODIFIER parameter.
- Additional values are transferred at the CMD_PARA parameter for individual PE commands (see the description of the individual PE commands). The parameter CMD_PARA_LEN defines the length of the data at the parameter CMD_PARA.

The commands are transferred without plausibility test. The response data of the PE entity is saved to the RESPONSE_DATA data area that is addressed by VARIANT pointer (for information about the response frame contents, refer to the description of the respective PE command).

Write and read jobs of the "PE_CMD" instruction.

The instruction "PE_CMD" uses "WRREC (Page 2232)" internally to transmit a PROFIenergy command as a write job to the PE entity. "PE_CMD" then waits for the acknowledgment from the PE entity. The acknowledgment data record is read with the instruction "RDREC (Page 2213)" every 100 milliseconds. Until acknowledgment is received from the PE entity, the function repeats the read job for 10 seconds at intervals of 100 milliseconds. The response data of the PE entity is also read with the instruction "RDREC (Page 2213)".

Below is a flowchart of the write and read jobs:

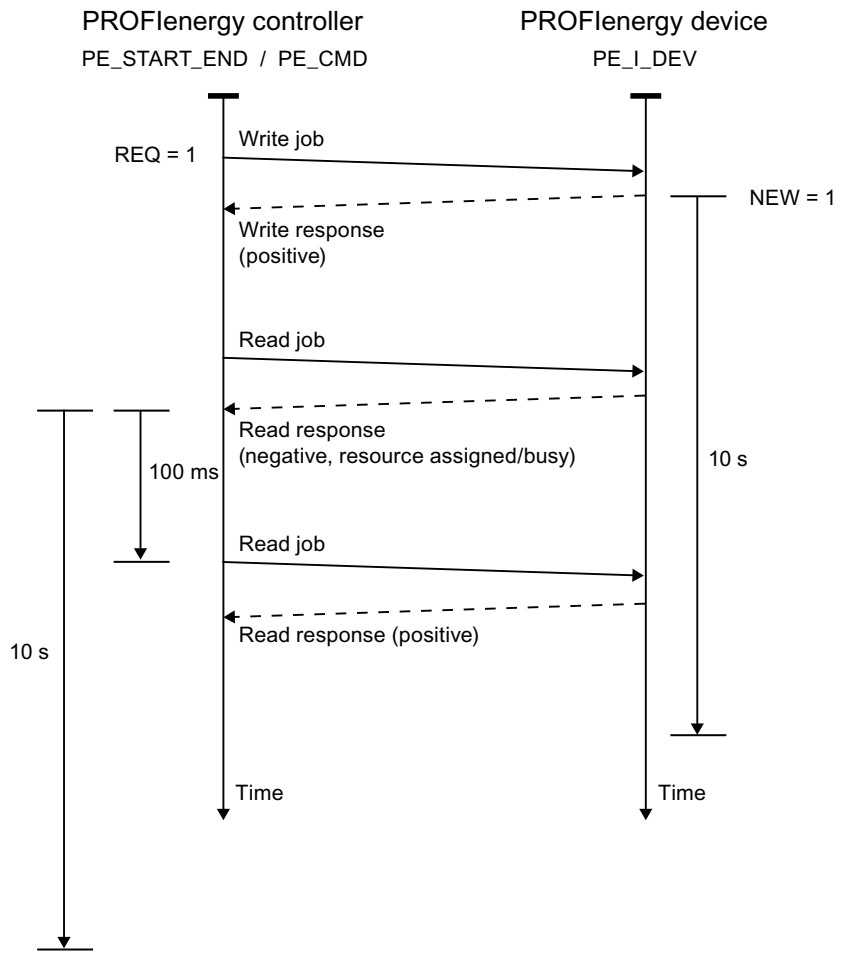

Parameter

The following table shows the parameters of the "PE_CMD" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts transferring the PE command upon a positive edge.
ID	Input	HW_ SUBMODULE	I, Q, M, D, L or constant	Address of the PE entity Use the hardware ID of the head module for a PROFINET IO device. Refer to the system constants of the assigned IO controller for the hardware ID. The name of the head module is composed of the name of the IO device and the suffix [Head] (example: "IO_Device_1[Head]"). If the PE entity is an I-device, use the hardware identifier of a transfer area instead.

Parameter	Declaration	Data type	Memory area	Description
CMD	Input	BYTE	I, Q, M, D, L, P or constant	Service-Request-ID of the PROFIenergy command in accordance with the PROFIenergy profile (see "CMD and CMD_MODIFIER parameters"). Further service request IDs are possible following extensions of the PROFIenergy profile.
CMD_MODIFIER	Input	BYTE	I, Q, M, D, L, P or constant	PROFIenergy sub-command (only when CMD=3 or CMD=16, see "CMD and CMD_MODIFIER parameters") Further sub-commands are possible following extensions of the PROFIenergy profile.
CMD_PARA	Input	VARIANT	I, Q, M, D, L	Parameters for the PE commands: • Get mode: PE_mode_ID • Get measurement values: List of Measurement_Ids The complete Service Data Request is entered.
CMD_PARA_LEN	Input	INT	I, Q, M, D, L, P or constant	The actual length of the command parameters (<= length defined in CMD_PARA; is verified by the instruction).
CMD_DATA_STRUCTURE_IDENTIFIER_RQ	Input	BYTE	I, Q, M, D, L, P or constant	Value of the Data Structure Identifier from the standard Note: The parameter CMD_DATA_STRUCTURE_IDENTIFIER_RQ exists as of instruction version V2.0.
RESPONSE_DATA	InOut	VARIANT	I, Q, M, D, L	PROFIenergy information May be complete response frame including block header, depending on the command. Note: If the buffer is too small, only the number of bytes which is specified in the VARIANT pointer will be entered.
VALID	Output	BOOL	I, Q, M, D, L	Command successfully sent.
BUSY	Output	BOOL	I, Q, M, D, L	Command still being processed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred during processing.
STATUS	Output	DWORD	I, Q, M, D, L	Block status / error number (see "STATUS parameter"):

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameters CMD and CMD_MODIFIER

CMD	CMD_ MODIFIER	PROFIenergy command	Description
01	0	Start_Pause (Page 2393)	Start energy-saving mode or switch to a different energy-saving mode.
02	0	End_Pause (Page 2393)	End energy-saving mode.
03	1	Query_Modes - List of energy-saving modes (Page 2394)	Output the energy-saving modes supported.
	2	Query_Modes - Get mode (Page 2395)	Output attributes of the energy-saving mode currently enabled.
04	0	PEM_Status (Page 2397)	Query status of energy-saving mode.
05	0	PE_Identify (Page 2399)	Read out the number and description of PE commands supported.
16	1	Query_Measurement - Get_Measurement_List (Page 2399)	List of measured values supported by the PE entity.
	2	Query_Measurement - Get_Measurement_Values (Page 2402)	Output the measured values of the PE entity.

Parameter STATUS

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error while reading the data record • B#16#**DF**: Error while writing the data record • B#16#**C0**: Error message from the internal communication instructions "RDREC (Page 2213)" and "WRREC (Page 2232)".
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific • **FE**: DP/PNIO profiles - PROFIenergy-specific error

Field element	Name	Meaning
STATUS[3]	Error_Code_1	Error ID • When Error_Decode = **80**: – **81**: Length conflict at the CMD_PARA and CMD_PARA_LEN parameters, or the maximum data record length (4095 bytes) was exceeded. – **82-8F**: Other error messages (reserved) • When Error_Decode = **FE**: – **01**: "Service Request ID" invalid – **02**: Incorrect "Request_Reference" – **03**: "Modifier" invalid – **04**: "Data Structure Identifier RQ" invalid – **05**: "Data Structure Identifier RS" invalid – **06**: "PE energy-saving modes" are not supported – **07**: "Response" is too long (maximum transmissible length exceeded) – **08**: "Count" invalid – **50**: A suitable energy-saving mode (energy mode) is not available. – **51**: Time value specified is not supported. – **52**: "PE_Mode_ID" invalid
STATUS[4]	Error_Code_2	Manufacturer-specific extension of the error ID

Note

Error messages of the instructions RDREC and WRREC

The instruction "PE_CMD" uses the instructions "WRREC (Page 2232)" and "RDREC (Page 2213)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 2276) parameter for an explanation of the error codes of the "WRREC (Page 2232)" and "RDREC (Page 2213)" instructions.

See also

Description of PROFIenergy (Page 2366)

Difference between synchronous and asynchronous instructions (Page 596)

PE_DS3_Write_ET200S: Set power module switching behavior (S7-1200, S7-1500)

Description

The instruction "PE_DS3_Write_ET200S" sends basic settings for power module switching behavior to the ET 200S. The switching behavior of up to 8 slots in the ET 200S (e.g. for power modules) is defined with the instruction "PE_DS3_Write_ET200S".

Note

This instruction is not part of the PROIenergy profile but instead supplements SIMATIC-specific functions.

Parameter

The following table shows the parameters of the "PE_DS3_Write_ET200S" instruction:

Parameter	Declaration	Data type	Memory area	Description
ENABLE	Input	BOOL	I, Q, M, D, L or constant	A positive edge triggers the transfer of the data record. Transfer the data record again following voltage OFF/ON.
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Address of the ET 200S Accept the address from the hardware configuration.
SLOT_NO_X	Input	INT	I, Q, M, D, L, P or constant	Slot number of switchable power module X.
FUNC_X	Input	INT	I, Q, M, D, L, P or constant	Function of the module in this slot. Use the parameter FUNC_X to define the switching behavior of the PM-E (power module of the ET 200S): • FALSE: – With "PAUSE_START": -No effect on PM-E -PM-E remains on – With "PAUSE_STOP": -Switches PM_E back on • TRUE: – With "PAUSE_START": -Switches PM_E off – With "PAUSE_STOP": -Switches PM-E back on
BUSY	Output	BOOL	I, Q, M, D, L	Transfer not yet complete.
DONE	Output	BOOL	I, Q, M, D, L	Transfer completed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Transfer completed with error.
STATUS	Output	DWORD	I, Q, M, D, L, P	Error number (see STATUS parameter of the instruction "PE_Start_End (Page 2368)")

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

See also

Description of PROFIenergy (Page 2366)

Difference between synchronous and asynchronous instructions (Page 596)

PE_WOL: Starting and stopping energy-saving mode via WakeOnLan (S7-1200, S7-1500)

Description PE_WOL (S7-1200, S7-1500)

Description

The "PE_WOL" instruction sends the PROFIenergy commands "Start_Pause" and "End_Pause" to multiple PROFIenergy-enabled devices in the PROFINET I/O systems.

Multiple PE devices can be coordinated via the instruction provided that the PE devices support the "Wake on LAN" function over a UDP connection.

The "PE_WOL" instruction can only be executed on a CPU with and integrated Ethernet interface. This CPU should be able to load blocks with a size of approx. 400 KB. If this CPU is connected via an Ethernet CP, it is not possible to use the block with PROFINET I/O systems.

The instruction "PE_WOL" is executed asynchronously.

The "PE_WOL" instruction only supports devices with a device number lower than 256. Devices with a higher device number are not taken into account.

Definition: Wake on LAN

Using Wake on LAN, deactivated devices can be reactivated through the receipt of a special Ethernet packet.

For this procedure to work, the data processing equipment must have a network controller that is capable of receiving such a packet.

This packet (Magic Packet™) has a special format. It contains the MAC address of the network adapter 15 times.

Selection of devices

The devices are selected using the user data block at the parameter PENERGY (Type: "PE_PLUS"). The user DB represents the database for processing multiple devices.

Prior to initialization of "PE_WOL", you must store at least the following information in the user DB:

- ID of the PROFINET I/O system
- Data of the connection which is used for "Wake On LAN".

- Port number which is used for "Wake On LAN".
- For each device
  - Pause time (PauseTime)
  - Switching of the device to the PE_SLEEP_MODE (EnableSleep)

You initialize the "PE_WOL" instruction with the COM_RST parameter. The jobs stored in the user DB are processed one after the other after initialization.

The graphic below illustrates how the PE command "Start_Pause" is transmitted to several devices:

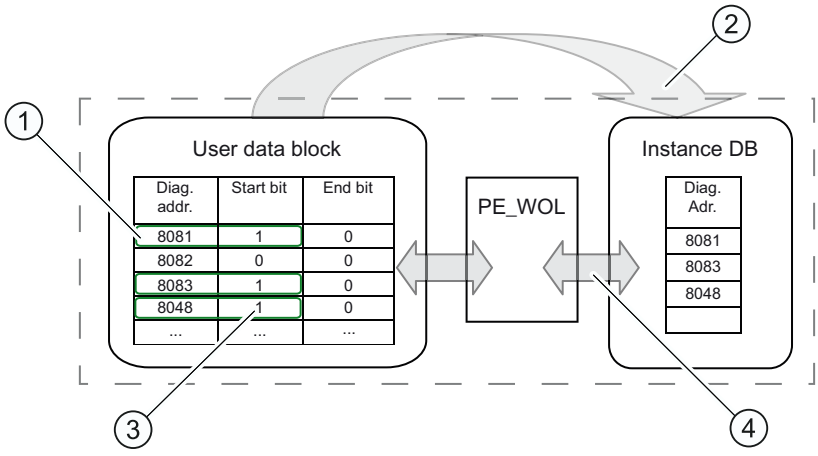

(1) Step 1: The bit "CmdStartPause" of the devices to be shut down is set to "1" by the user.

(2) Step 2: The diagnostic addresses of the devices to be shut down (CmdStartPause = "1") are linked to the queue.

(3) Step 3: The bit "CmdStartPause" is reset automatically once the jobs have been linked.

(4) Step 4: The instruction "PE_WOL" processes the jobs as soon as they are linked.

A PROFIenergy command "CmdStartPause" or "CmdEndPause" can be sent for all recognized devices in the PROFINET IO system via the START and END parameters.

The status of the job processing as well as possible errors during processing are output by the STATUS parameter.

Operation of the instruction using the user DB

The instruction "PE_WOL" can only be operated via the user DB. A basic procedure applies in this case:

1. Selection of the command to be executed for a device:
   - START_PAUSE ("CmdStartPause" in user DB)
   - ENDE_PAUSE ("CmdEndPause" in user DB)
   - UPDATE_STATUS ("CmdUpdateStatus" in user DB)
2. Setting the bit for updating ("Update" in the header of the user DB)
   At least one CPU cycle should pass with an "Update" = False between two updates; otherwise, edge detection cannot be guaranteed.

Prioritizing the PE commands

The graphic below shows the chronological sequence of three possible commands.

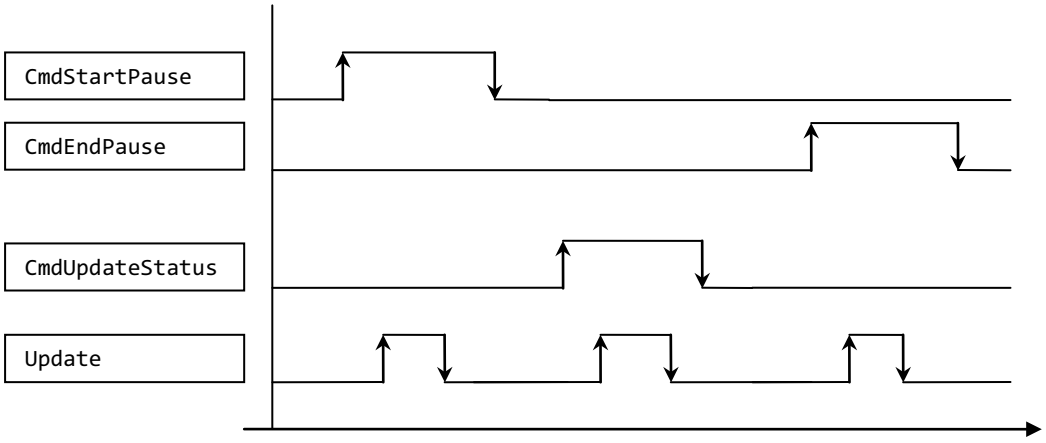

These are processed one after the other, regardless of whether the previous command call was successful or completed with an error.

Only one command is executed if you set two commands, such as "CmdEndPause" and "CmdUpdateStatus" simultaneously. There is a prioritization within the block:

- The command "CmdStartPause" has the highest priority and is therefore always executed if selected.
- The command "CmdEndPause" has the second highest priority.
- The command "CmdUpdateStatus" has the lowest priority.

If all three commands are set at the same time, the commands that are not executed remain preselected. In this case, the next rising edge executes the next command.

Parameters

The following table shows the parameters of the "PE_WOL" instruction:

Parameter	Declaration	Data type	Memory area	Description
COM_RST (Page 2382)	Input	BOOL	I, Q, M, D, L	Resets the block and performs a re-initialization. As long as True is set here, the initialization is started but not yet fully completed. Only the falling edge continues the initialization and switches to the normal operating mode after initialization.
START (Page 2383)	Input	BOOL	I, Q, M, D, L	A rising edge performs a "CmdStartPause" PROFIenergy command for all detected devices that support this function.
END (Page 2384)	Input	BOOL	I, Q, M, D, L	A rising edge performs a "CmdEndPause" PROFIenergy command for all detected devices that support this function.

Parameter	Declaration	Data type	Memory area	Description
PENERGY (Page 2384)	InOut	PE_PLUS	D	Pointer to the user DB that contains the database for processing multiple devices.
STATUS (Page 2389)	Output	DWORD	I, Q, M, D, L	Status/error number for the current status of the instruction (see "STATUS parameter").

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

See also

Difference between synchronous and asynchronous instructions (Page 596)

Parameter COM_RST (S7-1200, S7-1500)

Sequence of the initialization routine

You start the initialization of the "PE_WOL" instruction with the COM_RST parameter.

The following flow diagram shows the initialization routine.

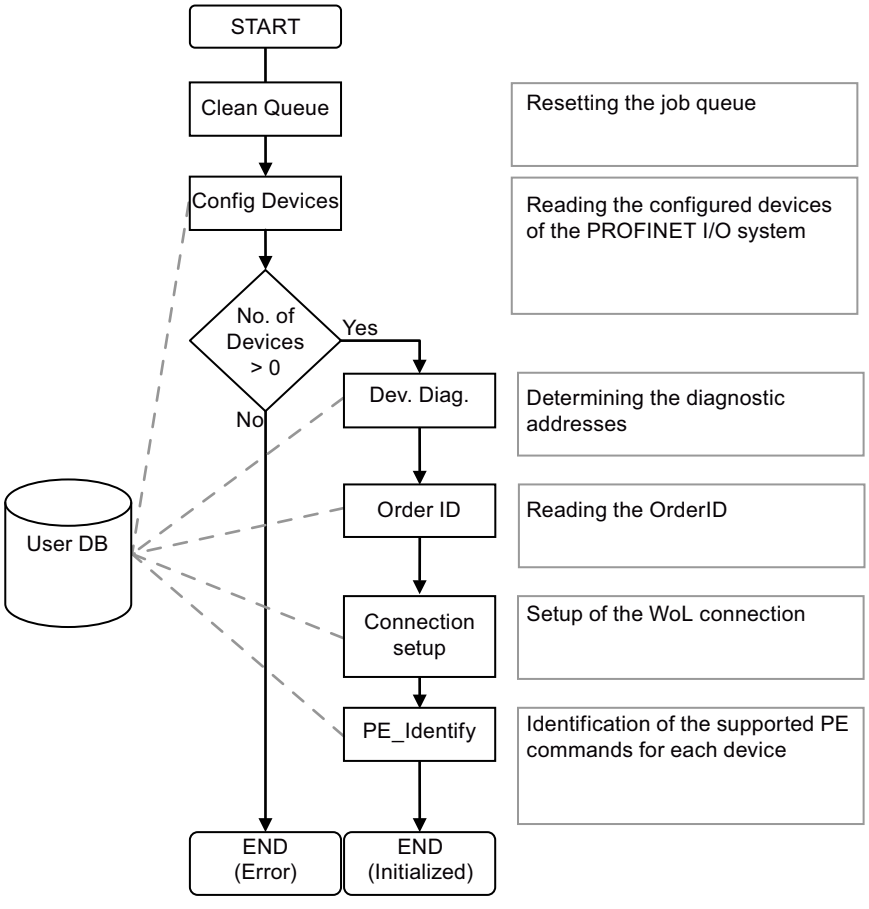

Parameter START (S7-1200, S7-1500)

Sequence of the CmdStartPause command

The flow diagram shows the internally used functions and interaction with a device when executing the CmdStartPause command.

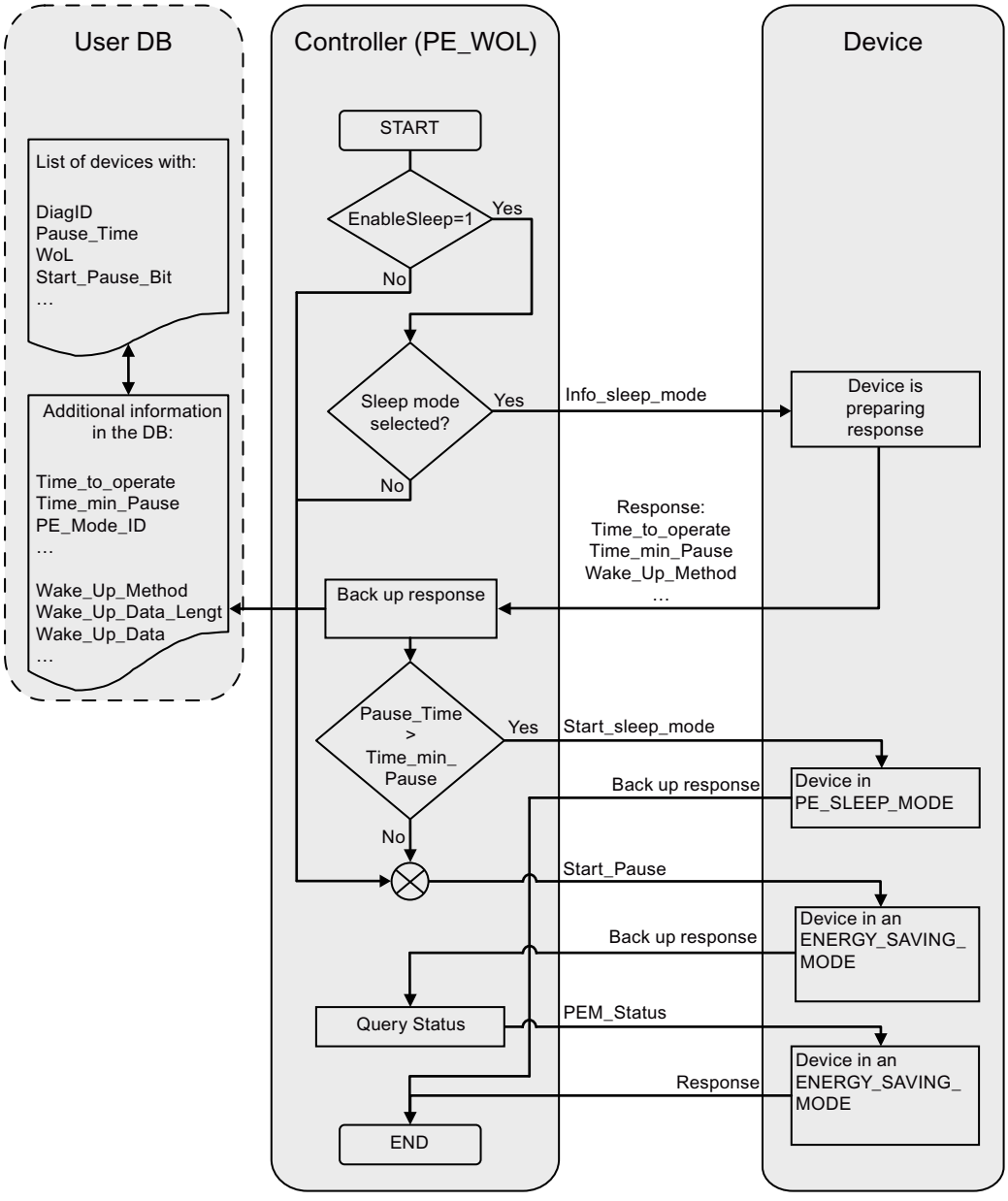

Parameter END (S7-1200, S7-1500)

Sequence of the CmdEndPause command

The flow diagram shows the internally used functions and interaction with a device when executing the CmdEndPause command.

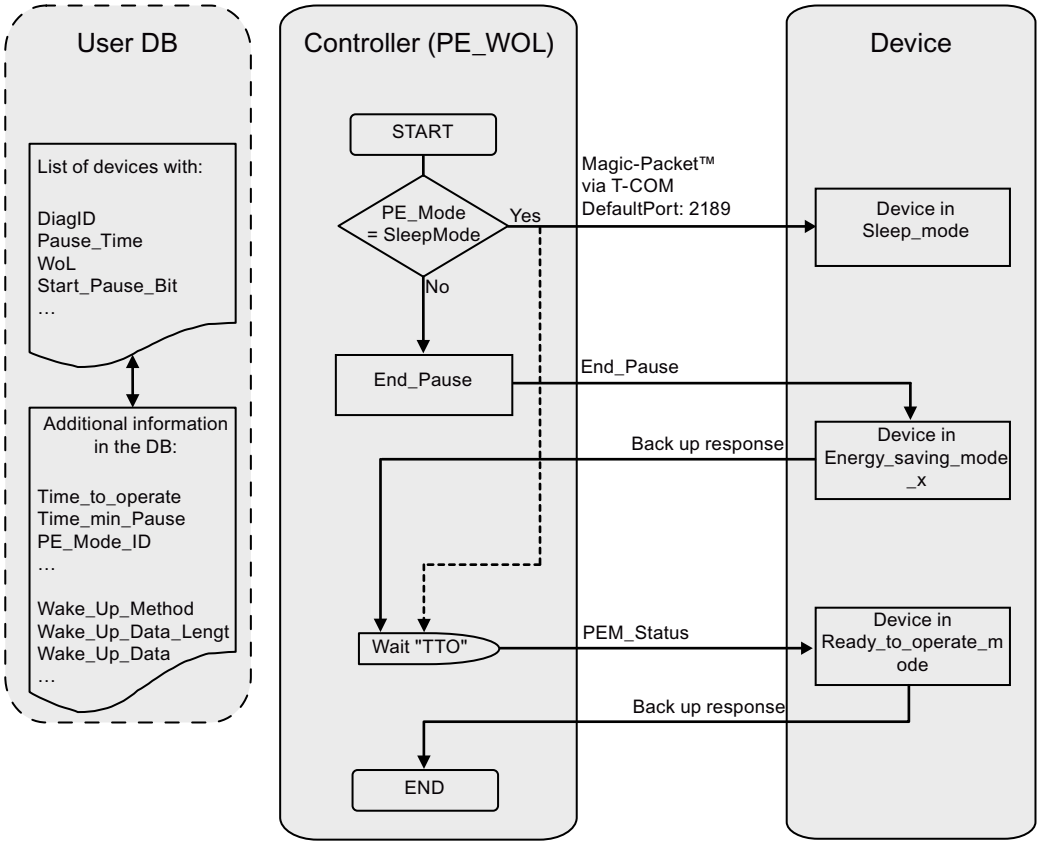

PENERGY parameter (S7-1200, S7-1500)

Data block at the PENERGY parameter

The user DB for the "PE_WOL" PROFIenergy instruction represents a data base for processing multiple devices.

The data block is usually divided into two parts. These are:

- Header 110 bytes
- Device section for a maximum of 256 devices each with 100 bytes (Device). These include:
  - Device-specific data (Device)
  - PROFIenergy-specific data (PE)
  - Data for the job processing (Task)
  - User data (UserData)

The data block operates with optimized access.

Connection parameter "Connection"

The "PE_WOL" instruction reserves a connection resource in the area of "Open User Communication". This is used as a UDP connection. The following parameters must be defined in the data block for this:

- Connection ID ("Connection.id" parameter)
  The connection ID is an integer within the range of 1 to 4095. It is used to identify the communication resources allocated by the firmware, for example the send and receive buffers.
  The connection ID must be unique throughout the CPU.
- Port number that is used for the "Wake on LAN" function ("Header.PortNo" parameter)
  Number of the UDP port via which a "Wake On LAN" packet is sent. These port numbers are part of the communication resources, which are identified and provided by the firmware through the connection ID. The default setting for port 2189 used here is currently not assigned by IANA. The port number is transferred to the connection configuration and applied to parameter "Connection.local_tsap_id[1]". The port number and the length of the remote port are defined using parameters "Connection.rem_tsap_id[1]" and "Connection.rem_tsap_id_LEN" and must be assigned manually.

Structure of the data block

The data block has the following structure:

Name	Data type	Offset	Comment
Header	PE_HEADER	-	Header information
Update (1)	BOOL	-	Signal for indicating a change in the data area. • True= indicates a change by the user. • False = indicates the application of changes.
Initialized	BOOL	-	Signal indicating the completed initialization. • True=Initialization finished. • False= indicates that the block is not initialized.
LinkUp	BOOL	-	Indicates successful configuration of the Ethernet interface. • True=interface ready for use. • False=interface not yet configured.
LinkDown	BOOL	-	Indicates an unconfigured interface. • True=interface is not yet configured. • False=interface is currently being configured or is now configured.
PROFINET_ID (1)	INT	-	ID of the PROFINET I/O system
Reserved	ARRAY [1..37] OF BYTE	-	Reserved
LastDeviceID	INT	-	Contains the highest Device-ID in this PROFINET I/O system.
PortNo (1)	WORD	-	Port number that is used for the "Wake on LAN" function (default= 2189).
Connection	TCON_Param	-	Contains the connection configuration of the "Wake on LAN" connection.

Name	Data type	Offset	Comment
BLOCK_LENGTH	UInt	-	Length of the structure (always B#16#40).
ID (1)	CONN_OUC	-	Connection ID
CONNECTION_TYPE (1)	USINT	-	Connection type (UDP = B#16#13)
ACTIVE_EST (1)	BOOL	-	Active connection establishment (always passive for UDP)
LOCAL_DEVICE_ID (1)	USINT	-	Contains the interface ID (CPU dependent)
LOCAL_TSAP_ID_LEN (1)	USINT	-	Contains the length in bytes of the own/local UDP port
REM_SUBNET_ID_LEN (1)	USINT	-	Unused (always B#16#00)
REM_STADDR_LEN (1)	USINT	-	Contains the length of the remote IP address or B#16#00.
REM_TSAP_ID_LEN (1)	USINT	-	Contains the length in bytes of the remote UDP port. The length specification must be entered manually.
NEXT_STADDR_LEN (1)	USINT	-	Contains the length of the default router address (not relevant).
LOCAL_TSAP_ID (1)	ARRAY[1..16] OF BYTE	-	Contains your own local port number. The value is applied from parameter PortNo during initialization.
REM_SUBNET_ID (1)	ARRAY[1..6] OF USINT	-	Unused (always B#16#00)
REM_STADDR (1)	ARRAY[1..6] OF USINT	-	Contains the remote IP address.
REM_TSAP_ID (1)	ARRAY[1..16] OF BYTE	-	Contains the remote UDP port number. The port number must be entered manually.
NEXT_STADDR (1)	ARRAY[1..6] OF BYTE	-	Irrelevant
SPARE (1)	WORD	-	
Device	ARRAY[1..256] OF PE_DEVICE	-	Array of devices
Device	PE_DEV	-	Contains data for each device
DeviceID	HW_DEVICE	-	Hardware identifier of the device. Assigned by the hardware configuration.
PE_EntityID	HW_IO	-	Hardware identifier of the PROFIenergy entity. Assigned by the hardware configuration.
MACAdr	ARRAY[1..6] OF BYTE	-	Contains the MAC address of the device.
IPAdr	ARRAY[1..4] OF BYTE	-	Contains the IP address of the device.
OrderID_MxLen	Byte	-	Contains the maximum length of the OrderID.
OrderId_ActLen	Byte		Contains the actual length of the OrderID.
OrderID_Data	ARRAY[1..20] OF CHAR		Contains the OrderID of the device.
PE	PE_PE	-	PROFIenergy-specific data.

Name	Data type	Offset	Comment
ModeID	BYTE	-	PE_MODE_ID according to PROFIenergy specification.
Result	BYTE	-	PE ErrorCode according to PROFIenergy specification.
PauseTime (1)	TIME	-	Contains the idle time in ms.
TimeToPause	TIME	-	Contains the time it takes for the device to go into Pause mode.
TimeToOperate	TIME	-	Contains the time it takes for the device to go into operating mode.
MinSleepTime	TIME	-	Contains the minimum time of the device in PE_SLEEP_MODE.
SleepToOperate	TIME	-	Contains the time it takes for the device to go from PE_SLEEP_MODE into operating mode.
StatusOperate	BOOL	-	Indicates the operating mode of the device.
StatusPause	BOOL	-	Indicates the Pause mode of the device.
StatusSleep	BOOL	-	Indicates the PE_SLEEP_MODE of the device.
StatusTransitOK	BOOL	-	Indicates the transition from one energy state to another.
StatusInTransit	BOOL	-	Indicates a current state transition.
StatusTransitNOK	BOOL	-	Indicates that the state change was not successful.
StatusError	BOOL	-	Indicates an error with the device.
StatusRetryEx	BOOL	-	Indicates an unsuccessful execution of a command. No more attempts will be made to execute this command.
CmdStartPause (1)	BOOL	-	Places a START_PAUSE command for this device in the queue.
CmdEndPause (1)	BOOL	-	Places a END_PAUSE command for this device in the queue.
CmdUpdateStatus (1)	BOOL	-	Places a PEM_STATUS command for this device in the queue.
EnableSleep (1)	BOOL	-	Allows PE_SLEEP_MODE for this device. • True= device should go to PE_SLEEP_MODE if idle time is sufficient • False=device may not go into PE_SLEEP_MODE.
Services	WORD	-	Shows all supported PROFIenergy services.
UserData (2)	ARRAY[1..24] OF BYTE	-	User-defined data
Task	PE_TASK	-	Job processing

Name	Data type	Offset	Comment
Cmd	BYTE	-	Internal bits for job processing
CmdJ	BYTE	-	Internal bits for job processing
TimeStart	BOOL	-	Starts a delay time.
TimeStarted	BOOL	-	The delay time has just started.
TimeDone	BOOL	-	Shows the expiration of the delay time.
Done	BOOL	-	Indicates that this job has been completed.
DelayedCmd	BOOL	-	Indicates that a delayed command is still pending.
IsV1_0	BOOL	-	Indicates that this device is a Spec. V1.0 device.
IsWakeOnLAN	BOOL	-	Indicates that this device is woken up by "Wake On LAN".
RetryCount	BYTE	-	Repetition counter for PE_COMMANDS
Duration	TIME	-	Contains the value for the delay in ms.
StartTime	TIME	-	Contains the start time for the delay time.
MachineState	INT	-	Contains the internal status of the job.

(1) To be filled in by the user.
(2) Available for use by the user.

Parameter STATUS (S7-1200, S7-1500)

Parameter STATUS

The output value at the STATUS parameter is divided into three areas:

- Bits 31 to 24: MESSAGE
- Bits 23 to 16: LOCATION
- Bits 15 to 0: INFORMATION

You can learn the meaning of each error code for the three areas in the following tables:

Table 4-30 Possible values for MESSAGE

Error code (W#16#...)	Description
00	No error.
50	Instruction initialized.
51	Determining configuration of the PROFINET I/O system.
52	The instruction could not determine any configured devices in the PROFINET I/O system.
53	Determining the logical addresses of configured devices.
54	Reading interface information out of the devices.
55	Determining I&M data (data record 0 only) of the configured devices.
56	Configuring PROFINET interface for sending the "Wake on LAN" MagicPaket™ via UDP.
57	Determining PROFIenergy capabilities of the connected devices.
62	Invalid PROFINET I/O system ID detected. The causative number is displayed in the INFORMATION field.

Error code (W#16#...)	Description
70	The instruction is initiated and processing jobs. The value in the INFORMATION field contains the number of currently active jobs.
80	The instruction has lost the initialization during processing of jobs. This generally occurs when either the instance DB or the user DB has been reloaded.
FF	Unknown error has occurred.

Table 4-31 Possible values for LOCATION

Error code (W#16#...)	Description
00	The instruction is either not initialized or it is idle.
70	The instruction is waiting for jobs.
71	The instruction has appended a job to the job list.
72	The instruction is preparing to send a job.
73	The instruction is sending a job to a device.
74	The instruction is waiting for a response from the device.
75	The instruction is evaluating the response of the device.
76	The instruction is removing the job from the job list.
FF	Unknown error has occurred.

Table 4-32 Possible values for INFORMATION

Error code (W#16#...)	Description
0000	No additional information available and no jobs active.
0001 -00FF	1 – 255 jobs are currently being processed.
7000	Re-initialization with COM_RST started but not yet completed.
8001	Error at 1st parameter
8002	Error at 2nd parameter
8003	Error at 3rd parameter
8004	Error at 4th parameters
8005	Error at 5th parameter This error is reported if there is no interconnection or an invalid interconnection to the user DB. Possible causes: • The user DB is too small. • The user DB is write-protected. • There is no user DB in the RAM. • The user DB is invalid for the employed CPU.

Error code (W#16#...)	Description
8085 to 80CE	Error when configuring the connection. The error messages of the internally used TCON instruction are output. The description of the error messages can be found in the table for the STATUS (Page 3541) parameter.
8100	An attempt has been made to place more than the maximum of 256 possible jobs. This is a temporary error, which is resolved by completing a few jobs. The placed jobs are not accepted and must be placed again.
8200	An attempt was made to send an invalid or an unsupported PROFIenergy command (PE_COMMAND).
8400	Connection ID outside the permitted range. Initialization aborted. Check the ID of the connection configuration for "Wake on LAN". See data block at parameter PENERGY > Header > Connection > ID.
84xx	A communication error has occurred. The number of the device that generated the error is output to "xx".
85xx	An error has been returned from device xx. The number of the device that generated the error is output to "xx".
8600	The requested wake-up method is currently not supported.
FFFF	Unknown error has occurred.

See also

PENERGY parameter (Page 2384)

PROFIenergy commands (S7-1200, S7-1500)

Structure of the message frames (S7-1200, S7-1500)

Structure of the response frame in accordance with the PROFIenergy profile

The table below shows the basic structure of the response frame in accordance with the PROFIenergy profile. The response frame consists of a general section (Header) and a specific section (Service Data Response). The contents of the specific section of the response frame can be found in the description of the relevant PROFIenergy command.

Block definition	Attributes	Value	Data type	Description
BlockHeader	BlockType	801_{hex}	WORD	
	BlockLength		WORD	Number of bytes without consideration of the fields BlockType and BlockLength.
	BlockVersionHigh	1_{hex}	BYTE	
	BlockVersionLow	0_{hex}	BYTE	

Block definition	Attributes	Value	Data type	Description
Response Header	Service_Request_ID	1_{hex} to FF $_{hex}$	BYTE	ID of the PI command executed. The ID of the PE command processed by the PE entity is returned in the response frame: • 01: Start_Pause • 02: End_Pause • 03: Query_Modes • 04: PEM_Status • 05: PE_Identify • 06 to 09: Reserved • 16: Query_Measurement • 11 to CF: Reserved • D0 to FF: Manufacturer-specific
	Request_Reference	1_{hex} to FF $_{hex}$	BYTE	Unique number to identify the query/response pair (returned by the server in the response).
Service Header Response	Status	1_{hex} to FF $_{hex}$	BYTE	Information whether or not the PI command was executed: • 00: Reserved • 01: Completed • 02: Completed with error • 03: Data incomplete • 04 to CF: Reserved • D0 to FF: Depends on the Service_Request_ID
	Data_Structure_ Identifier_RS	1_{hex} to FF $_{hex}$	BYTE	• 00: Reserved • 01 to FF: Data structure depends on the Service_Request_ID • 0xFF - Error
Service Data Response				Depends on the Service-Request-ID: • For the service request IDs, see the CMD and CMD_MODIFIER parameters of the instruction "PE_CMD (Page 2373)". • The specific contents of the response frame can be found in the description of the PE command (see for example the "Start_Pause (Page 2393)" command).

PI Command "Start_Pause" (S7-1200, S7-1500)

Description

Use the PE command "Start_Pause" to start the energy-saving mode. The command Start_Pause can be used to:

- Switch the PE from "ready" state (PE_ready_to_operate) to an energy-saving mode (PE_energy_saving_mode).
- The PE entity can automatically switch between the energy-saving modes.
  Energy consumption can increase or decrease when you switch energy-saving mode.

Calling the PI command "Start_Pause"

The command "Start_Pause" is called with the instruction "PE_CMD (Page 2373)" with the following parameters:

Parameter	Value	Description
CMD	1	Call of PE command "Start_Pause".
CMD_MODIFIER	0	There are no additional specifications of the command call for command "Start_Pause".
CMD_PARA_LEN	4	Length of the CMD_PARA parameter of 4 bytes.
CMD_PARA	VARIANT	VARIANT pointer to the value for "Pause_Time" (TIME).

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see instruction "PE_CMD (Page 2373)"):

Attribute	Value	Data type	Description
PE_Mode_ID	1 to 255	BYTE	Identification number of the energy-saving mode
Reserved	0	BYTE	-

PI Command "End_Pause" (S7-1200, S7-1500)

Description

The PE command "End_Pause" is used to end the energy-saving mode of the PE entity.

Calling the PE command "End_Pause"

The command "End_Pause" is called with the instruction "PE_CMD (Page 2373)" with the following parameters:

Parameter	Value	Description
CMD	2	Call of PE command "End_Pause".
CMD_MODIFIER	0	There are no additional specifications of the command call for command "End_Pause".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 2373)"):

Attribute	Value	Data type	Description
Time_to_operate	-	DWORD	Expected time it takes to switch to the "PE_ready_to_operate" mode.

PI command "Query_modes" - "List_Energy_Saving_Modes" (S7-1200, S7-1500)

Description

Use PE command "Query_modes" and sub-command (modifier) "List_Energy_Saving_Modes" to output all energy-saving modes (PE_Mode_ID) supported by the PE entity.

The query result is written in the form of a response frame to the data block referenced by theRESPONSE_DATA parameter.

Calling the PE command "Query_modes" - "List_Energy_Saving_Modes"

The command "List_Energy_Saving_Modes" is called with the instruction "PE_CMD (Page 2373)" with the following parameters:

Parameter	Value	Description
CMD	3	Call of PE command "Query_modes".
CMD_MODIFIER	1	Specification of the command call: Select the sub-command "List_Energy_Saving_Modes" to output the number and types of energy-saving modes supported.
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 2373)"):

Attribute	Value	Data type	Description
Number_of_PE_ Mode_IDs	1	BYTE	The number of energy-saving modes.
PE_Mode_IDs	-	Array [...] of BYTE	Array with the IDs of the supported energy-saving modes. The meaning of the individual IDs depends on the PE entity.

PI command "Query_modes" - "Get_Mode" (S7-1200, S7-1500)

Description

You use the PE command "Query_modes" and the sub-command (modifier) "Get_Mode" to output the attributes of the energy-saving mode which is currently enabled.

Calling the PE command "Query_modes" - "Get_Mode"

The call of the command with the instruction "PE_CMD" occurs with the following parameters:

Parameter	Value	Description
CMD	3	Call of the PE command "Query_modes"
CMD_MODIFIER	2	Specification of the command call: Select the sub-command "Get_Mode" to output the status of the mode which is currently enabled
CMD_PARA_LEN	1	Length of the CMD_PARA parameter of 1 byte
CMD_PARA	VARIANT	VARIANT pointer to the value for PE_MODE_ID

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 2373)"):

Attribute	Value	Data type	Description
PE_Mode_ID	• 0 "PE_power_off" mode • 1...254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	ID of the energy-saving mode currently enabled
PE_Mode_Attributes	Bit 0: • = 0: Only static energy consumer and time values are available. • = 1: Dynamic energy consumer and time values are available. Bit 1 to 7: • Reserved	BYTE	
Time_min_Pause [1]	Time difference without date	DWORD	Minimum pause time for the PI mode. The minimum pause time is the sum of the values of the following attributes: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay See the description "PAUSE_TIME parameter" of the instruction "PE_START_END: Start and exit energy-saving mode (Page 2368)".
Time_to_Pause [1]	Time difference without date	DWORD	Switch off time: Duration from call of energy-saving mode to start of energy-saving mode (transition duration of PE_ready_to_operate to PE_energy_saving_mode). The switch-off time depends on the PE entity.
Time_to_operate [1]	Time difference without date	DWORD	Switch-on time: Duration of the transition from energy-saving mode (PE_energy_saving_mode) to ready to operate mode (PE_ready_to_operate). The PE entity calculates the duration dynamically in the output operation.
Time_min_length_of_stay [1]		DWORD	Minimum duration of the energy-saving mode in the PE entity
Time_max_length_of_stay [1]		DWORD	Maximum duration of the energy-saving mode in the PE entity
Mode_Power_Consumption [2]		REAL	Power consumption of the PE entity in active energy-saving mode Unit: kW

Attribute	Value	Data type	Description
Energy_ Consumption_ to_pause [2]		REAL	Energy consumption of the PE entity during transition from ready-to-operate mode (PE_ready_to_operate) to energy-saving mode (PE_energy_saving_mode) Unit: kWh
Energy_ Consumption_ to_operate [2]		REAL	Energy consumption of the PE entity during transition from energy-saving mode (PE_energy_saving_mode) to ready-to-operate mode (PE_ready_to_operate) Unit: kWh
[1] If the duration is infinite, 0xFFFFFFFF is output. If the duration is zero, "0" is output. [2] If the data for energy and power consumption by the PE entity is not defined, "0.0" is output as value.			

PI command "PEM_Status" (S7-1200, S7-1500)

Description

The PE command "PEM_Status" is used to query the status of a PE entity energy-saving mode that is currently enabled.

Calling the PE command "PEM_Status"

The command "PEM_Status" is called with the instruction "PE_CMD" with the following parameters:

Parameter	Value	Description
CMD	4	Call of PE command "PEM_Status".
CMD_MODIFIER	0	There are no other specifications of the command call for command "PEM_Status".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 2373)"):

Attribute	Value	Data type	Description
PE_Mode_ID_ Source	• 0 "PE_power_off" mode • 1 to 254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	Mode of the PE entity before a PE command is sent.
PE_Mode_ID_ Destination	• 0 "PE_power_off" mode • 1 to 254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	Mode of the PE entity after a PE command is executed.
Time_to_operate	Time difference without date	DWORD	Switch-on time: Duration of the transition from energy-saving mode (PE_energy_saving_mode) to ready to operate mode (PE_ready_to_operate). The PE entity calculates the duration dynamically during output.
Remaining_time_to_ destination	Time difference without date	DWORD	Remaining time to switch to another mode.
Mode_Power_ Consumption		REAL	Power consumption of the PE entity in active energy-saving mode. Unit: kW
Energy_ Consumption_ to_Destination		REAL	Energy consumption for the current PI transition Unit: kWh
Energy_ Consumption_ to_operate		REAL	Energy consumption of the PE entity during transition from energy-saving mode (PE_energy-saving mode) to ready-to-operate mode (PE_ready_to_operate) Unit: kWh

PI command "PE_Identify" (S7-1200, S7-1500)

Description

The PE command "PE_Identify" is used to read out the number and description of PE commands supported by the PE entity. The number of specific commands supported depends on the PE entity. As PE_Identify is itself a PE command, at least three supported PE commands are output for a positive response: Start_Pause, End_Pause and PE_Identify.

Call of the PE command "PE_Identify"

The command "PE_Identify" is called with the instruction "PE_CMD (Page 2373)" with the following parameters:

Parameter	Value	Description
CMD	5	Call of the command "PE_Identify".
CMD_MODIFIER	0	There are no other specifications of the command call for command "PE_Identify".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	Irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 2373)"):

Attribute	Value	Data type	Description
Count [1]	6	BYTE	Number of supported PROFIenergy commands
Start_Pause	1	BYTE	First PE command supported (Service_Request_ID)
End_Pause	2	BYTE	...
Query_Modes	3	BYTE	...
PEM_Status	4	BYTE	...
PE_Identify	5	BYTE	...
Query_Measurement	16	BYTE	Last PE command supported (Service_Request_ID)
[1] The number of commands supported is manufacturer-specific and depends on the PE entity used. When all 6 PE commands are supported, the values given are an example of a response frame.			

PI Command "Query_Measurement" - "Get_Measurement_list" (S7-1200, S7-1500)

Description

Use the PE command "Query_Measurement" and sub-command (modifier) "Get_measurement_list" to query the specific measured values supported by the PE entity. The supported measured values are output as a list in the data block referenced by the RESPONSE_DATA parameter.

Calling the PE command "Query_Measurement" - "Get_Measurement_list"

The command is called with the instruction "PE_CMD (Page 2373)" with the following parameters:

Parameter	Value	Description
CMD	16	Call of the command "Query_Measurement".
CMD_MODIFIER	1	Specification of the command call: Select the sub-command "Get_Measurement_List" to output a list of supported measured values.
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 2373)"):

Attribute	Value	Data type	Description
Count	-	BYTE	Number of measurement IDs
reserved	-	BYTE	
...			
Measurement_ID	-	WORD	First supported Measurement_ID The Measurement_ID is manufacturer-specific. For more information, refer to the manual of the respective PE entity.
Accuracy_Domain	-	BYTE	See "accuracy domain" table.
Accuracy_Class	-	BYTE	See "accuracy classes" table.
Range	-	REAL	Specifies the scale end value of the measured value (only for accuracy domain 1). The attribute range uses the same unit as defined using attribute Measurement_ID (only one unit is used for each Measurement_ID).
...			
Measurement_ID	-	WORD	Last supported Measurement_ID
Accuracy_Domain	-	BYTE	See "accuracy domain" table.
Accuracy_Class	-	BYTE	See "accuracy classes" table.
Range	-	REAL	Specifies the scale end value of the measurement value (only for accuracy domain 1). The attribute range uses the same unit as defined using attribute Measurement_ID (only one unit is used for each Measurement_ID).

Accuracy domains

Accuracy domain	Description
0	Reserved
1	The accuracy deviation is output in percentages of the scale end value. The percentage of the possible deviation is divided into accuracy classes (see table: Accuracy classes of the accuracy domains 1 and 2).

Accuracy domain	Description
2	The accuracy deviation is output in percentages of the current measurement value. The percentage of the possible deviation is divided into accuracy classes (see table: Accuracy classes of the accuracy domains 1 and 2).
3	The measuring accuracy adheres to the standard IEC 61557-12. The function performance classes for performance measurement and monitoring devices (PMD) without external sensors, and the system performance classes for PMD with external sensors, are coded as set out in the "Accuracy classes of accuracy domain 3" table.
4	The entry of the accuracy adheres to the standard EN 50470-3, chapter 8 (see also table: Accuracy classes of accuracy domain 4).

Accuracy classes

Table 4-33 Accuracy classes of the accuracy domains 1 and 2

Accuracy class	0	1	2	3	4	5	6	7	8
Meaning	Reserved	0.01%	0.02%	0.05%	0.1%	0.2%	0.5%	1%	1.5%

Accuracy class	9	10	11	12	13	14	15	>15
Meaning	2%	2.5%	3%	5%	10%	20%	>20%	Not defined

Table 4-34 Accuracy classes of accuracy domain 3

Accuracy class	0	1	2	3	4	5	6	7	8
Meaning	Reserved	0,02	0,05	0,1	0,2	0,5	1	1,5	2

Accuracy class	9	10	11	12	13	14	>13
Meaning	2,5	3	5	10	20	20%	Not defined

Table 4-35 Accuracy classes of accuracy domain 4

Accuracy class	0	1	2	3	4	5	6	>7
Meaning	Reserved	0,5	1,0	1,5	2,0	2,5	3,0	Not defined

PI Command "Query_Measurement" - "Get_Measurement_values" (S7-1200, S7-1500)

Description

Use the PE command "Query_Measurement" and sub-command (modifier) "Get_measurement_values" to output a list of measured values supported by the PE entity. The measured values are output as a list in the data block referenced by the RESPONSE_DATA parameter.

Calling the PE command "Query_Measurement" - "Get_Measurement_values"

The command is called with the instruction "PE_CMD (Page 2373)" with the following parameters:

Parameter	Value	Description
CMD	16	Call of the command "Query_Measurement".
CMD_MODIFIER	2	Specification of the command call: Select the command "Get_Measurement_Values" to output a list of supported measurement values.
CMD_PARA_LEN	0	Depends on the number of measurement values. The length of the parameters results from the attribute count and the sum of the lengths of the attributes for the transferred measurement values.
CMD_PARA	VARIANT	VARIANT pointer to data structure with list of measured values to be queried (see "CMD_PARA parameter").

Parameter CMD_PARA

The structure specified with the VARIANT pointer at the CMD_PARA parameter must have the following setup:

Attribute	Value	Data type	Description
Count	-	BYTE	Number of measured values (Measurement-IDs)
reserved	0	BYTE	Not used
Measurement_ID	-	WORD	First measured value queried
...			
Measurement_ID	-	WORD	Last measured value queried

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 2373)"):

Attribute	Value	Data type	Description
Count [1]	-	BYTE	Number of measured values (Measurement-IDs)
reserved	0	BYTE	Not used
Length_of_Structure	2 to 65535	WORD	Length of the structure in bytes

Attribute	Value	Data type	Description
Measurement_Data_Structure_ID	1 = simple value	BYTE	Defines the following structure.
Measurement_ID	0 to 65535	WORD	ID of the supported measurement value.
Status_of_ Measurement_Value	1 to 3	BYTE	Status of the measurement value: • 1: Valid • 2: Not supported • 3: Invalid
Transmission_Data_Type	-	REAL	
End_of_demand	-	TOD	Optional time stamping with data type TimeOfDay.
...			
Length_of_Structure	-	WORD	Length of the structure in bytes
Measurement_Data_Structure_ID	-	BYTE	Defines the following structure.
Measurement_ID	-	WORD	ID of the supported measurement value.
Status_of_Measurement_Value	-	BYTE	Status of the measurement value: • 1: Valid • 2: Not supported • 3: Invalid
Transmission_Data_Type	-	REAL	
End_of_demand	-	TOD	Optional time stamping with data type TimeOfDay.
[1] If the data length of the measurement values queried exceeds the size of the PDU (Protocol Data Unit) of the protocol layer, the data will not be completely transferred and only the supported measurement values will be output.			

iDevice / iSlave (S7-1200, S7-1500)

PE_I_DEV: Control PROFIenergy commands in the I-Device (S7-1200, S7-1500)

Description

The instruction "PE_I_DEV" is used to process the PROFIenergy profile in the intelligent IO device (iDevice).

- The instruction "PE_I_DEV" is called cyclically by the user program of the iDevice and receives all PROFIenergy commands.
- The PROFIenergy response is generated by configuring an auxiliary block. The response in the pause is fully programmable. The response data must be provided within 10 seconds; otherwise, "State conflict 0x80B5" will occur at the STATUS parameter of the instruction in the IO controller.

Note: The functions are executed by the firmware in a standard PROFIenergy-compatible IO device such as the ET 200S.

No specific knowledge of the PROFIenergy standard is required to use the instruction.

PROFIenergy auxiliary blocks (PE auxiliary blocks)

The PE auxiliary blocks are used to generate the response frame. Enter the response data (in plain text) at the input parameters of the relevant block.

- For each PROFIenergy command, there is a corresponding auxiliary block for a positive response:
  - PE command "Start_Pause": PE_Start_RSP (Page 2408)
  - PE command "End_Pause": PE_End_RSP (Page 2409)
  - PE command "Query_modes" - "List_Energy_Saving_Modes": PE_List_Modes_RSP (Page 2411)
  - PE command "Query_modes" - "Get_Mode": PE_Get_Mode_RSP (Page 2412)
  - PE command "PEM_Status": PE_PEM_Status_RSP (Page 2414)
  - PE command "PE_Identify": PE_Identify_RSP (Page 2415)
  - PE command "Query_Measurement" - "Get_Measurement_list": PE_Measurement_List_RSP (Page 2417)
  - PE command "Query_Measurement" - "Get_Measurement_values": PE_Measurement_Value_RSP (Page 2418)
- Irrespective of the PROFIenergy command, there is an additional shared auxiliary block for a negative response (see PE_Error_RSP (Page 2407)).

Interconnection of the auxiliary blocks

The instruction "PE_I_DEV" and the auxiliary blocks are coordinated. Some of the parameters are simply interconnected. The diagram below shows which of the parameters need to be interconnected.

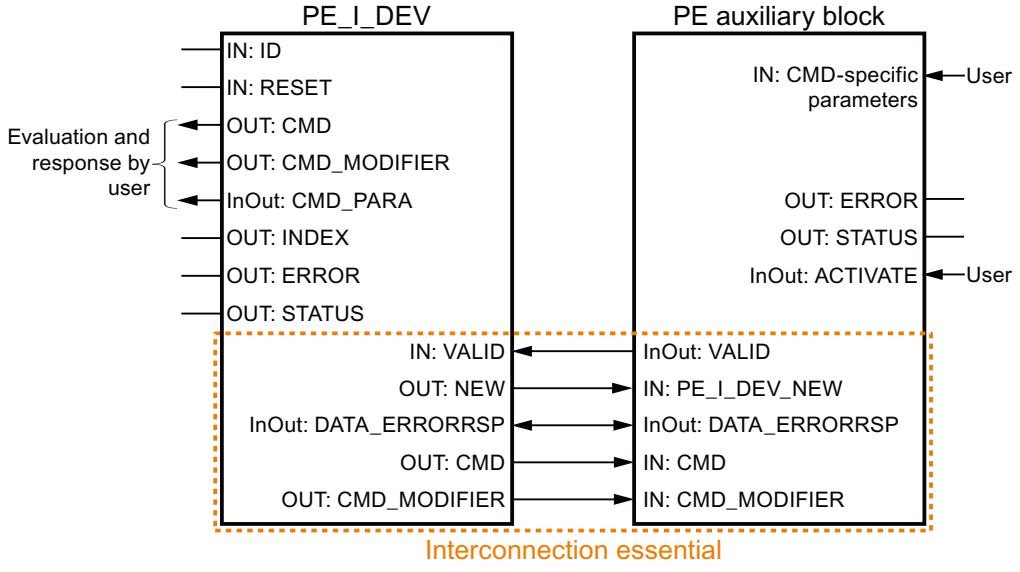

Parameters

The following table shows the parameters of the instruction "PE_I_DEV":

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L or constant	Resets the instruction.
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Address of the transfer area, which is supplied in the IO controller with the data for PROFIenergy. The hardware identifier can be found in the system constants.
VALID	Input	BOOL	I, Q, M, D, L or constant	The response data for the PROFIenergy controller is ready and can be sent.
CMD_PARA	Output	VARIANT	I, Q, M, D, L	Parameters for: • Get mode: PE_mode_ID • Get measurement values: List of Measurement_IDs (list of IDs of the tags to be read; either one tag or multiple tags may be read at any given time). Maximum length: 234 bytes Note: With S7-1200 the pointer must not reference 64-bit data types (in other words not the data types LWORD, LINT and ULINT).
DATA_ERRORRSP	InOut	VARIANT	I, Q, M, D, L	Pointer to the data area containing the acknowledgment data for the PROFIenergy controller. This must correspond to the pointer which is also used with the auxiliary blocks. Note: With S7-1200 the pointer must not reference 64-bit data types (in other words not the data types LWORD, LINT and ULINT).
INDEX	Output	INT	I, Q, M, D, L	Data record number of the PROFIenergy record (set at 0x80A0)
CMD	Output	INT	I, Q, M, D, L	Service-Request-ID of the PROFIenergy command in accordance with the PROFIenergy profile (see "CMD and CMD_MODIFIER parameters"). Further PE commands (Service-Request-IDs) are possible following extensions of the PROFIenergy profile.
CMD_MODIFIER	Output	INT	I, Q, M, D, L	PROFIenergy sub-command: • Only when CMD=3 or CMD=16; see "CMD and CMD_MODIFIER parameters". • For all other commands: "0". Further sub-commands are possible following extensions of the PROFIenergy profile.
NEW	Output	BOOL	I, Q, M, D, L	New data available from the PROFIenergy controller.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Command completed with error.
STATUS	Output	DWORD	I, Q, M, D, L	Error information (see "STATUS parameter").

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

CMD and CMD_MODIFIER parameters

CMD	CMD_ MODIFIER	PROFIenergy command	Description
01	0	Start_Pause	Start energy-saving mode or switch to a different energy-saving mode.
02	0	End_Pause	End energy-saving mode.
03	1	Query_Modes - List energy saving Modes	Output the energy-saving modes supported.
	2	Query_Modes - Get Mode	Output attributes of the energy-saving mode currently enabled.
04	0	PEM_Status	Query status of energy-saving mode.
05	0	PE_Identify	Read out the number and description of PE commands supported.
16	1	Query_Measurement - Get_Measurement_List	List of measured values supported by the PE entity.
	2	Query_Measurement - Get_Measurement_Values	Output of the measured values of the PE entity.

STATUS parameter

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error while reading the data record • B#16#**DF**: Error while writing the data record • B#16#**C0**: Error message from the "PE_I_DEV" instruction or the internally used "PRVREC (Page 2344)" and "RCVREC (Page 2342)" communication instructions.
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific
STATUS[3]	Error_Code_1	Error ID (when Error_Decode = **80**): • **B1**: Write length error (error in write length or insufficient length information on the data type VARIANT).
STATUS[4]	Error_Code_2	For PROFINET errors: Output of IO controller error message If no PROFINET error has occurred, the value in STATUS[4] = "0".

Note

Error messages of the "PRVREC" and "RCVREC" instructions

The "PE_I_DEV" instruction uses the "PRVREC (Page 2344)" and "RCVREC (Page 2342)" instructions for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

For the meaning of the error codes of the "PRVREC" and "RCVREC" instructions, see the description of the corresponding STATUS (Page 2276) parameter.

See also

WRREC: Write data record (Page 2232)

RDREC: Read data record (Page 2213)

Description of PROFIenergy (Page 2366)

Auxiliary blocks of the instruction PE_I_DEV (S7-1200, S7-1500)

PE_Error_RSP: Generate negative answer to command (S7-1200, S7-1500)

Description

The auxiliary block "PE_Error_RSP" (Response with failure) generates a negative response if the command requested is not supported (either in general or temporarily). Response generation does not depend on the command requested.

Parameter

The following table shows the parameters of the "PE_Error_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block is processed if the parameter value is set to "1".
ERROR_CODE	Input	BYTE	I, Q, M, D, L or constant	Error number
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 2403)" instruction.

Parameter	Declaration	Data type	Memory area	Description
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block sets the parameter once the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 2403)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_Start_RSP: Generate answer to command at start of pause (S7-1200, S7-1500)

Description

The auxiliary block "PE_Start_RSP" (Start Pause) generates the response to the PE command Start_Pause (Page 2393). The instruction returns the energy-saving mode to which the device has switched (PE_MODE_ID parameter).

If the response differs depending on the length of the pause, you can return this fact in the feedback on the energy-saving mode (PE_Mode_ID = 1 for a brief pause, PE_Mode_ID = 2 for a longer pause, etc.).

Parameter

The following table shows the parameters of the "PE_Start_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block is processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 2403)" instruction.

Parameter	Declaration	Data type	Memory area	Description
PE_MODE_ID	Input	BYTE	I, Q, M, D, L or constant	PE mode that the process assumes
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 2403)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block sets the parameter once the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 2403)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_End_RSP: Generate answer to command at end of pause (S7-1200, S7-1500)

Description

The auxiliary block "PE_End_RSP" generates the response to the PE command End_Pause (Page 2393). To switch from the current mode to "Ready_To_Operate" mode, the time required is returned as response.

Parameter

The following table shows the parameters of the "PE_End_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block is processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 2403)" instruction.
Time_to_Operate	Input	DWORD	I, Q, M, D, L or constant	Time required to switch from the current mode to "Ready_To_Operate".
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 2403)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block sets the parameter once the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 2403)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_List_Modes_RSP: Generate queried energy savings modes as answer (S7-1200, S7-1500)

Description

The auxiliary block "PE_List_Modes_RSP" generates the response to the PE command List_Energy_Saving_Modes (Page 2394). The response generated includes the number and the IDs of energy-saving modes supported.

Parameter

The following table shows the parameters of the "PE_List_Modes_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block is processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 2403)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Number_of_PE_Mode_IDs	Input	BYTE	I, Q, M, D, L or constant	Number of energy-saving modes supported. Permitted values: 1 to 254
PE_MODE_ID	Input	VARIANT	I, Q, M, D, L	Points to the area in which the IDs of the energy-saving modes supported (PE_Mode_ID) are stored. Valid range: 1 to 254.
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 2403)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block sets the parameter once the response data for the PROFIenergy controller is available and ready for transmission.

Parameter	Declaration	Data type	Memory area	Description
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 2403)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_Get_Mode_RSP: Generate queried energy data as answer (S7-1200, S7-1500)

Description

The auxiliary block "PE_Get_Mode_RSP" generates the response to the command Get_Mode (Page 2395). The response frame contains the time, performance, or energy data of an energy-saving mode.

Parameters

The following table shows the parameters of the "PE_Get_Mode_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block is processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 2403)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
PE_Mode_ID	Input	BYTE	I, Q, M, D, L or constant	ID of the energy-saving mode currently enabled.
PE_Mode_Attributes	Input	BYTE	I, Q, M, D, L or constant	Additional information regarding energy saving mode

Parameter	Declaration	Data type	Memory area	Description
Time_min_Pause	Input	DWORD	I, Q, M, D, L or constant	Minimum pause duration for this PE energy-saving mode. This is the sum of the following three parameters: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay
Time_to_Pause	Input	DWORD	I, Q, M, D, L or constant	Time between the edge at the START parameter (see "PE_I_DEV (Page 2403)") and the requested PE energy-saving mode being reached.
Time_to_Operate	Input	DWORD	I, Q, M, D, L or constant	Maximum switch-on time to "PE_ready_to_operate". The "Time_to_operate" parameter can be used directly for the relevant calculations. The value may be a static maximum, or be calculated dynamically by the PE entity.
Time_min_Lenght_of_stay	Input	DWORD	I, Q, M, D, L or constant	Minimum dwell time of the PE entity in this PE mode.
Time_max_Lenght_of_stay	Input	DWORD	I, Q, M, D, L or constant	Maximum dwell time of the PE entity in this PE mode.
Mode_Power_Consumption	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the current PE mode in [kW].
Energy_Consum_to_Pause	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from "PE_ready_to_operate" to the current PE mode in [kWh].
Energy_Consum_to_operate	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from the current PE mode to "PE_ready_to_operate" in [kWh].
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 2403)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block sets the parameter once the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 2403)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_PEM_Status_RSP: Generate PEM status as answer (S7-1200, S7-1500)

Description

The auxiliary block "PE_PEM_Status_RSP" generates the response to the command PEM_Status (Page 2397).

Parameter

The following table shows the parameters of the "PE_PEM_Status_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block is processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 2403)" instruction.
PE_MODE_ID_ Source	Input	BYTE	I, Q, M, D, L or constant	Source and Destination for PEM_STATUS. Values: • 0x00: PE_POWER_OFF • 0x01 – 0xFE: manufacturer-specific • 0xFF: PE_READY_TO_OPERATE
PE_MODE_ID_ Destination	Input	BYTE	I, Q, M, D, L or constant	
Time_to_Operate[1]	Input	DWORD	I, Q, M, D, L or constant	Maximum switch-on time to "PE_ready_to_operate". Time_to_operate" can be used directly for the relevant calculations. The value may be a static maximum, or be calculated dynamically by the PE entity.
Remaining_time_ to_destination[1]	Input	DWORD	I, Q, M, D, L or constant	Optional: Time remaining until the requested PE mode. Dynamic value or static maximum value
Mode_Power_ Consumption[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the current PE mode in [kW].

Parameter	Declaration	Data type	Memory area	Description
Energy_ Consumption_ to_Destination[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the time until the requested PE mode in [kW].
Energy_ Consumption_ to_operate[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from the current PE mode to "PE_ready_to_operate" in [kWh].
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 2403)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block sets the parameter once the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 2403)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

[1] If the time period is unlimited, the maximum value "0xFFFFFFFF" can be specified. If the time period is "Zero", "0x00" can be used.

[2] If no energy consumption value has been defined, "0.0" can be specified.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_Identify_RSP: Generate supported PROFIenergy commands as answer (S7-1200, S7-1500)

Description

The auxiliary block "PE_Identify_RSP" generates the response to the command PE_Identify (Page 2399). In the response to the command, specify which PROFIenergy commands are supported. Please note that PE_IDENTIFY is itself a PE command and has to be included in the response.

Parameter

The following table shows the parameters of the "PE_Identify_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block is processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 2403)" instruction.
Start_Pause	Input	BOOL	I, Q, M, D, L or constant	One parameter for each of the relevant PROFIenergy commands: • 0: PE command is not supported • 1: PE command is supported
End_Pause	Input	BOOL	I, Q, M, D, L or constant	
Query_Modes	Input	BOOL	I, Q, M, D, L or constant	
PEM_Status	Input	BOOL	I, Q, M, D, L or constant	
PE_Identify	Input	BOOL	I, Q, M, D, L or constant	
Query_ Measurement	Input	BOOL	I, Q, M, D, L or constant	
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 2403)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block sets the parameter once the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 2403)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_Measurement_List_RSP: Generate list of supported measured values as answer (S7-1200, S7-1500)

Description

The auxiliary block "PE_Measurement_List_RSP" generates the response to the command Get_measurement_list (Page 2399). In the response, specify which measured values (Measurement-IDs) are supported.

Parameter

The following table shows the parameters of the "PE_Measurement_List_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block is processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 2403)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Count	Input	BYTE	I, Q, M, D, L or constant	Number of measured values supported (measurement IDs)
Measurement_List	Input	VARIANT	D	Pointer to the array with the Measurement_IDs supported. For information on the structure of the array in accordance with the PROFIenergy profile, see: PI Command "Query_Measurement" - "Get_Measurement_list" (Page 2399)

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 2403)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block sets the parameter once the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 2403)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_Measurement_Value_RSP: Generate queried measured values as answer (S7-1200, S7-1500)

Description

The auxiliary block "PE_Measurement_Value_RSP" generates the response to the command Get_measurement_values (Page 2402). In the response, return the values of the requested measurements.

Parameter

The following table shows the parameters of the "PE_Measurement_Value_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block is processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 2403)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Count	Input	BYTE	I, Q, M, D, L or constant	Number of measured values (Measurement_Values).
Measurement_Values	Input	VARIANT	D	Pointer to the array with the measured values (Measurement_IDs). For information on the structure of the array in accordance with the PROFIenergy profile, see PI Command "Query_Measurement" - "Get_Measurement_values" (Page 2402)
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 2403)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 2403)" instruction. The auxiliary block sets the parameter once the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 2403)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

4.1.4.6 Module parameter assignment (S7-1500)

Writing and reading data records (S7-1500)

Principle

Some modules have a write-only system data area to which your program can transfer data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see following table).

Modules can also have a system data area. From your program, you have read-access only to the system data area. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see following table).

Note

There are modules that have both system data areas. These are physically separate areas and the only thing they have in common is their logical structure.

Write-only system data area

The following table shows the structure of the write-only system data area. This table also shows how the permitted size of the individual data records and which instructions can be used to write them.

Data record number	Contents	Size	Can be written with instruction
0	Parameter	-	WR_DPARM (Page 2431)
1	Parameter	-	WR_DPARM (Page 2431)
2 to 127	User data	Each ≤ 240 bytes	WR_DPARM (Page 2431) WR_REC (Page 2337)
128 to 240	Parameter	Each ≤ 240 bytes	WR_DPARM (Page 2431) WR_REC (Page 2337)

Read-only system data area

The following table shows the structure of the read-only system data area. This table also shows the permitted size of the individual data records and which instructions can be used to read them.

Data record number	Contents	Size	Can be read with instruction
0	Module-specific diagnostics data (set as standard for the entire system)	4 bytes	RD_REC (Page 2332)
1	Channel-specific diagnostics data (incl. data record 0)	4 to 220 Bytes	RD_REC (Page 2332)
2 to 127	User data	Each ≤ 240 bytes	RD_REC (Page 2332)
128 to 240	Diagnostics data	Each ≤ 240 bytes	RD_REC (Page 2332)

System resources

If you start several asynchronous data record transfers one after the other with only short intervals between them, the allocation of system resources by the operating system ensures that all the jobs are executed and that they do not interfere with each other.

If the limits of the system resources are reached, this is indicated in RET_VAL. You remedy this temporary error situation by repeating the job.

The maximum number of "simultaneously" active jobs of a single instruction type depends on the CPU.

RD_DPAR: Read module data record (S7-1500)

Description

You use the instruction to read the data record with the number INDEX of the addressed component from the configured system data. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the target range RECORD. In this case, the LEN output parameter contains the length of the read data in bytes.

If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR . In this case, the output parameter STATUS contains the error information.

Note

Information about the data records

For information on the data record numbers and the data record structure, refer to the product manual for the relevant module.

Functional description

The "RD_DPAR" instruction works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling "RD_DPAR" with REQ = 1.

The output parameter BUSY and bytes 2 and 3 of the output parameter STATUS show the status of the job. Bytes 2 and 3 of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Difference between synchronous and asynchronous instructions (Page 596)

The transfer of the data record is complete when the output parameter BUSY has the value FALSE.

Parameter

The following table shows the parameters of the "RD_DPAR" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module (e.g. input module). The number is assigned automatically and is stored in the properties of the module under "HW identifier" (e.g. "Input 0 - 15 > HW identifier").
INDEX	Input	INT	I, Q, M, D, L or constant	Data record number
RECORD	InOut	VARIANT	I, Q, M, D, L	Target range for the read data record
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The job is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.
STATUS	Output	DWORD	I, Q, M, D, L	• Call ID (bytes 2 and 3) or error code • Byte 1: B#16#00, if no error. Otherwise function ID from DPV1-PDU: In the case of error for data record reading B#16#DE, in the case of error for data record writing B#16#DF. If no DPV1 protocol element is used: B#16#C0. • Byte 4: Manufacturer-specific error ID extension
LEN	Output	INT	I, Q, M, D, L	Length of the read data record information

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.	-
8093	Address specified at parameter LADDR is invalid.	-
80B0	Instruction for module type not possible, or module does not recognize the data record.	-
80B1	The length of the data record to be sent is incorrect.	-
80C3	The required resources (memory, etc.) are occupied at this time.	
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)	-
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".		

Example

In the following example, you read the complete parameter data record of a digital input module.

Create nine tags in a global data block for storing the data and the structure "record". (You can find the organization of the structure below.)

SLI_gDB_RD_DPAR

		Name	Data type	Start value
1		Static		
2		start	Bool	false
3		laddr	HW_IO	257
4		dataRecNbr	Int	128
5		checkedValid	Bool	false
6		busy	Bool	false
7		error	Bool	false
8		status	DWord	16#0
9		lengthRec	Int	0
10		record	Struct	
11		Mode_C0_Bool	Array[0..7] of Bool	

Interconnect parameters: "SLI_itemiseByte_RD_DPAR" FC

To transfer the values of a bit sequence (0..7) create the FC "SLI_itemiseByte_RD_DPAR".

Create the following local tags in it.

	Name	Data type
1	Input	
2	in0	Bool
3	in1	Bool
4	in2	Bool
5	in3	Bool
6	in4	Bool
7	in5	Bool
8	in6	Bool
9	in7	Bool
10	Output	
11	out0	Bool
12	out1	Bool
13	out2	Bool
14	out3	Bool
15	out4	Bool
16	out5	Bool
17	out6	Bool
18	out7	Bool

Network 1: According to the example of network 1, create interconnections for all bits (0..7).

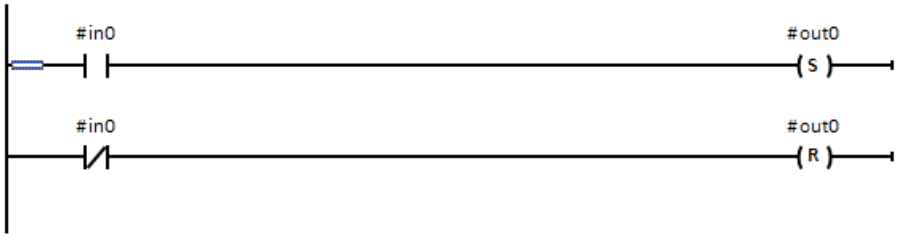

Interconnect parameters: "SLI_FB_RD_DPAR" FB

Create the following local tags in the "SLI_FB_RD_DPAR" FB.

7	Static	
8	RD_DPAR_Instance	RD_DPAR

Network 1: Interconnect the parameters of the "RD_DPAR" instruction as follows.

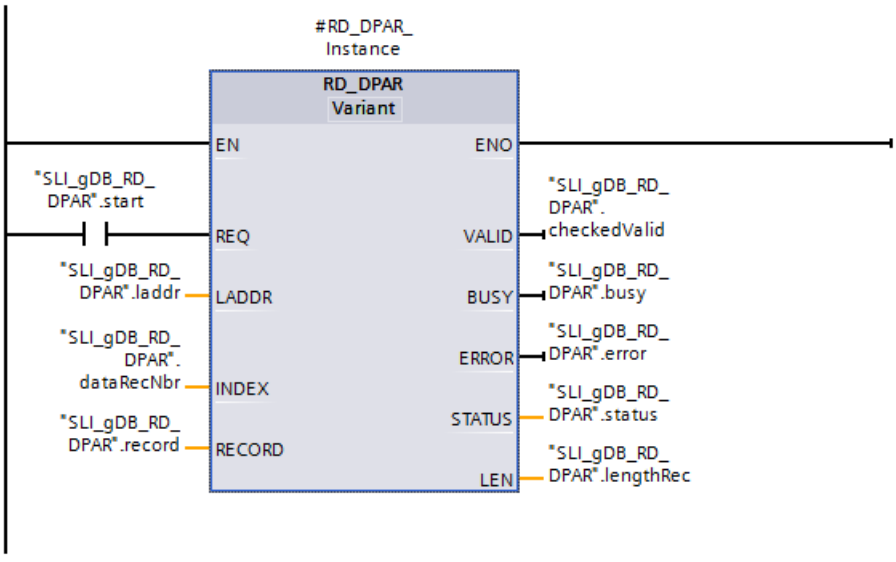

Network 2: To convert the tag, interconnect the "Mode_Channel0" tag as follows. All inputs and outputs (0..7) of the "SLI_itemiseByte_RD_DPAR" FC are interconnected in this case.

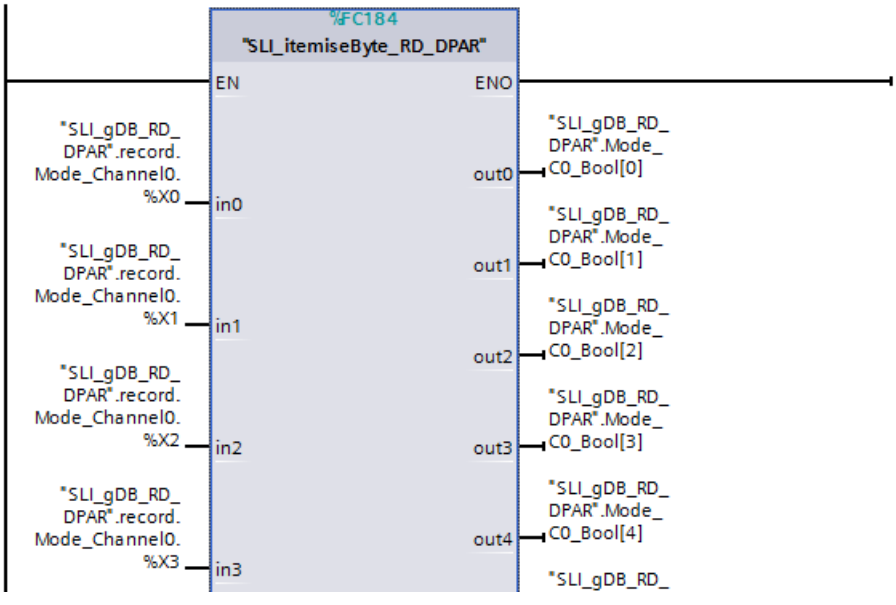

Hardware identifier

You create an input module and read the hardware ID in the properties of the input module. You store the hardware identifier via the input parameter LADDR ("laddr").

Result (FB "SLI_FB_RD_DPAR")

When the input parameter REQ ("start") returns the signal state "TRUE", the "RD_DPAR" instruction is started. Via the input parameter LADDR ("laddr") the instruction "RD_DPAR" calls the module. Using the input parameter INDEX ("dataRecNbr") the data record with the number "128" of the module is called. During several calls the "RD_DPAR" instruction reads out the data record and saves the data record using the parameter RECORD ("record").

The length of the read data record is stored using the output parameter LEN ("lengthRec"). According to the value of the input parameter MLEN ("maxLen"), has the value "0“ the data record information that is to be read is unlimited.

During processing the output parameter BUSY ("busy") is set to "TRUE" and VALID ("checkedValid") to "FALSE". A successful reading result ("checkedValid" to "TRUE") is not output until processing is complete. The block status is displayed at the output parameter STATUS ("status"). Processing takes place without errors.

SLI_gDB_RD_DPAR

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	laddr	HW_IO	257	16#0101
4	dataRecNbr	Int	128	128
5	checkedValid	Bool	false	TRUE
6	busy	Bool	false	FALSE
7	error	Bool	false	FALSE
8	status	DWord	16#0	16#0000_0000
9	lengthRec	Int	0	34
10	record	Struct		
11	Mode_C0_Bool	Array[0..7] of Bool		

The data record 128 contains the parameter information of all channels of the module. The value "2" of the tag "ByteOfDataRec" relates to the length of the data per channel. It also provides information on the status and the operating mode.

Excerpt of the read data record up to channel 1:

record	Struct		
Version	USInt	0	16
ByteOfDataRec	USInt	0	2
Diag_Channel0	Byte	16#0	16#00
Mode_Channel0	Byte	16#0	16#06
Diag_Channel1	Byte	16#0	16#00
Mode_Channel1	Byte	16#0	16#06

For a more detailed explanation of the data of the data record 128 convert the BYTE values to a bit string. Here an example of the operating mode of the channel 0: Since bits 1 and 2 are set to "TRUE" the channel 0 has an input delay of 3.2 ms. You can find the explanation of the structure of the parameter data record or the channel in the device manual of the module.

11	Mode_C0_Bool	Array[0..7] o...		
12	Mode_C0_Bool[0]	Bool	false	FALSE
13	Mode_C0_Bool[1]	Bool	false	TRUE
14	Mode_C0_Bool[2]	Bool	false	TRUE
15	Mode_C0_Bool[3]	Bool	false	FALSE
16	Mode_C0_Bool[4]	Bool	false	FALSE
17	Mode_C0_Bool[5]	Bool	false	FALSE
18	Mode_C0_Bool[6]	Bool	false	FALSE
19	Mode_C0_Bool[7]	Bool	false	FALSE

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

RD_DPARM: Read data record from configured system data (Page 2429)

Switching display formats in the program status (Page 7698)

Overview of the types of data records (Page 2206)

RD_DPARA: Read module data record asynchronously (S7-1500)

Description

You use the instruction to read the data record with the number RECNUM of a selected module from the configured system data. The read data record is entered in the target range defined by the parameter RECORD .

Note

Information about the data records

For information on the data record numbers and the data record structure, refer to the product manual for the relevant module.

Functional description

The "RD_DPARA" instruction works asynchronously, that is, its execution extends over multiple calls. You start the reading process by calling the instruction with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596)

Parameters

The following table shows the parameters of the "RD_DPARA" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module (e.g. input module). The number is assigned automatically and is stored in the properties of the module under "HW identifier" (e.g. "Input 0 - 15 > HW identifier").
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error occurred during the transmission, the following cases are distinguished: • RET_VAL contains the length of the actually read data record in bytes if the target range is larger than the read data record. • RET_VAL contains "0" if the length of the read data record is equal to the length of the target range.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The job is not yet complete.
RECORD	Output	VARIANT	I, Q, M, D, L	Target range for the data record read.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.	-
8093	Address specified at parameter LADDR is invalid.	-
80B0	Instruction for module type not possible, or module does not recognize the data record.	-
80B1	The length of the data record to be sent is incorrect. With RD_DPARM (Page 2429): The target range spanned by RECORD is too short.	-
80C3	The required resources (memory, etc.) are occupied at this time.	
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)	-
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 7698)

RD_DPARM: Read data record from configured system data (S7-1500)

Use

Note

Restriction on use

The "RD_DPARM" instruction is only intended for the migration from S7-300/400 CPUs to S7-1500 CPUs. In all other cases, use the "RD_DPARA" instruction.

Description

You use instruction to read the data record with the number RECNUM of the addressed module from the configured system data. The read data record is entered into the target range spanned by the RECORD parameter.

Note

Information about the data records

For information on the data record numbers and the data record structure, refer to the product manual for the relevant module.

Functional description

The "RD_DPARM" instruction works asynchronously, that is, its execution extends over multiple calls.

The job status is indicated by the output parameter RET_VAL.

See also: Difference between synchronous and asynchronous instructions (Page 596)

Parameters

The following table shows the parameters of the "RD_DPARM" instruction:

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module (e.g. input module). The number is assigned automatically and is stored in the properties of the module under "HW identifier" (e.g. "Input 0 - 15 > HW identifier").

Parameter	Declaration	Data type	Memory area	Description
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	Length of the data record read in bytes if the read data record fits in the target range and no error occurred in the transfer. If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	VARIANT	I, Q, M, D, L	Target range for the data record read.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7001	First call: Data transfer started	Distributed I/O
7002	Intermediate call: Data transfer already active	Distributed I/O
8090	Specified logical base address invalid.	-
8092	A type other than BYTE is specified in the VARIANT reference at the RECORD parameter.	-
8093	This instruction is not valid for the module selected with LADDR.	-
80B0	Instruction for module type not possible, or module does not recognize the data record	-
80B1	The target range spanned by RECORD is too short.	-
80C3	The required resources (memory, etc.) are occupied at this time.	-
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)	-
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 7698)

WR_DPARM: Transfer data record (S7-1500)

Description

You use the instruction "WR_DPARM" to transfer the data record with the number RECNUM from the configuration data to the addressed module. The data record can be static or dynamic.

Note

Information about the data records

For information on the data record numbers and the data record structure, refer to the product manual for the relevant module.

Parameters

The following table shows the parameters of the "WR_DPARM" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Write request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module (e.g. input module). The number is assigned automatically and is stored in the properties of the module under "HW identifier" (e.g. "Input 0 - 15 > HW identifier").
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8093	Address specified at parameter LADDR is invalid.	-

Error code* (W#16#...)	Explanation	Restriction
80A1	Negative acknowledgment during sending of data record to module (module defective or unplugged during sending).	-
80A2	DP protocol error at layer 2, possible hardware/interface error in DP slave.	Distributed I/O
80A3	DP protocol error at user interface/user.	Distributed I/O
80A9	This function is not supported.	-
80B0	Instruction for module type not possible, or module does not recognize the data record.	-
80B1	The length of the data record to be sent is incorrect.	-
80B2	The configured slot is not occupied.	-
80B3	Actual module type does not correspond to specified module type.	-
80B6	Access denied	-
80C1	The data of the previous write job on the module for the same data record has not yet been processed by the module.	-
80C3	The required resources (memory, etc.) are occupied at this time.	
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available or deactivated.	Distributed I/O
80C6	Data record transfer cancelled due to priority class abort (restart or background).	Distributed I/O
80D0	There is no entry for the module.	-
80D1	The data record number is not configured for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-
80D5	The data record is static.	-
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)	-
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 7698)

Difference between synchronous and asynchronous instructions (Page 596)

4.1.4.7 Interrupts

ATTACH: Attach an OB to an interrupt event (S7-1200, S7-1500)

Description

You use the instruction "ATTACH" to assign an organization block (OB) to a hardware interrupt event.

- You enter the symbolic or numeric name of the organization block in the OB_NR parameter. This will then be assigned to the event specified in the EVENT parameter.
- You select the hardware interrupt event at the EVENT parameter. Hardware interrupt events already generated are listed in the PLC tags under "System constants".

If the event in the EVENT parameter occurs following error-free execution of the "ATTACH" instruction, the organization block in the OB_NR parameter will be called and its program executed.

With the ADD parameter, you specify whether previous assignments of the organization block to other events should be canceled or retained. If the ADD parameter has the value "0", the existing assignments will be replaced by the current assignment.

Hardware interrupt events

Hardware interrupts can be used if events are not pending long enough. They react to these during runtime. Each hardware interrupt can be assigned to hardware interrupt OBs. These OBs contain the reaction to a particular event.

Hardware interrupts can be created for different events. Examples for this are:

- The detection of rising or falling edges for digital inputs.
- Violation of defined low and high limits for analog inputs.
- External reset, overflow/underflow, direction reversal etc. for high-speed counters.

Functional principle

Each hardware interrupt can be assigned to a hardware interrupt OB, which is placed in the queue for processing when the hardware interrupt event occurs. The assignment of OB and event can occur during configuration or at runtime:

- To assign an event to an OB during configuration, select a hardware interrupt OB for an event in the hardware configuration under "Hardware interrupts".
- Use the ATTACH instruction to make the assignment during runtime. The assignment of event and hardware interrupt OB is made via the EVENT and OB_NR parameters.

Parameters

The following table shows the parameters of the "ATTACH" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_ATT	I, Q, M, D, L or constant	Organization block (numbers up to 32767 are supported.)
EVENT	Input	EVENT_ATT	I, Q, M, D, L or constant	Hardware interrupt event to be assigned to the OB. The hardware interrupt event must first be enabled in the hardware device configuration for inputs or high-speed counters.
ADD	Input	BOOL	I, Q, M, D, L or constant	Effects on previous assignments: • ADD=0 (default): This event replaces all previous event assignments for this OB. • ADD=1: This event is added to the previous event assignments for this OB.
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0	No error
8090	OB does not exist
8091	OB is incorrect type
8093	Event does not exist
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you assign a hardware interrupt OB (OB40) to a created hardware interrupt event.

Requirements:

You create an input module (DI) and configure a hardware interrupt event for a positive edge under "Properties > Inputs 0-15 > ... > Channel 0".

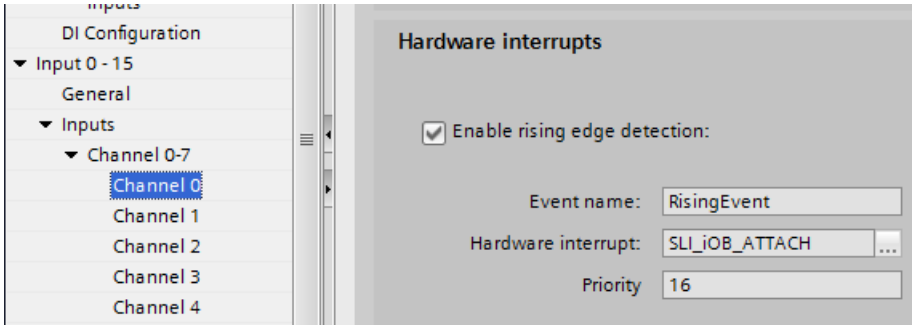

The following hardware ID of the configured hardware interrupt event is stored at the input parameter EVENT ("event").

General | IO tags | System constants | Texts

Name	Type	Hardware identifier
RisingEvent	Event_HwInt	16#C0000101
Local~DI_16x24VDC_HF_1	Hw_SubModule	257

Data storage

Create nine tags in a global data block for storing the data. You store the number of the OB to be assigned (OB40) with the input parameter OB_NR ("obNbr") of the instruction "ATTACH".

SLI_gDB_ATTACH

	Name	Data type	Start value
1	Static		
2	start	Bool	false
3	obNbr	OB_ATT	40
4	add	Bool	0
5	event	EVENT_ATT	16#C0000101
6	returnValue	Int	0
7	testValue1	Int	123
8	testValue2	Int	0
9	signal	Bool	0
10	scannedSignal	Bool	false

To call the hardware interrupt event, you create a tag table with a PLC tag (address "I0.0").

SLI_Var_ATTACH

	Name	Data type	Address
1	SLI_ATTACH_force	Bool	%I0.0

Interconnection

Interconnect the parameters of the instruction as follows.

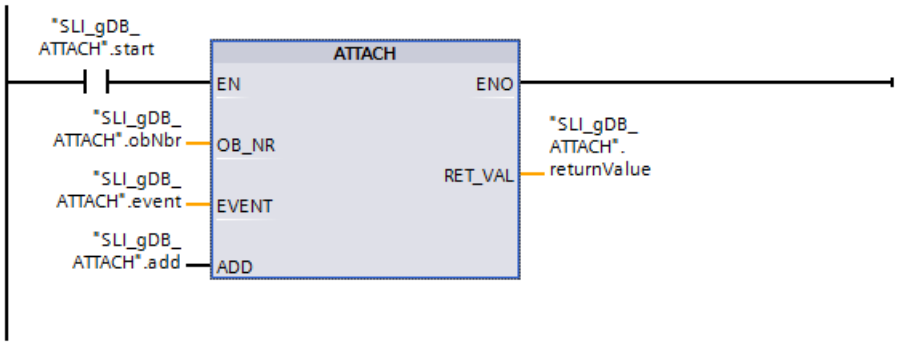

Interconnect the PLC tag as follows to call the hardware interrupt event.

Network 2:

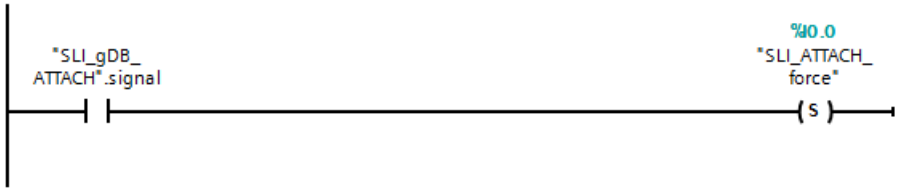

Network 3:

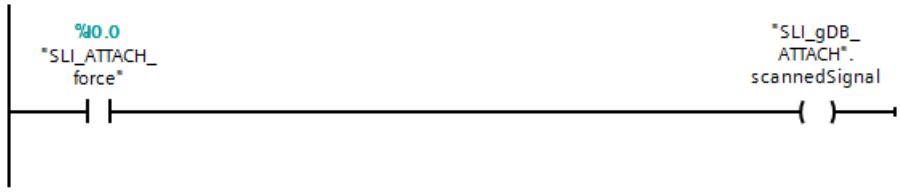

To reset, interconnect "testValue2" and "SLI_ATTACH_force" as follows.

Network 4:

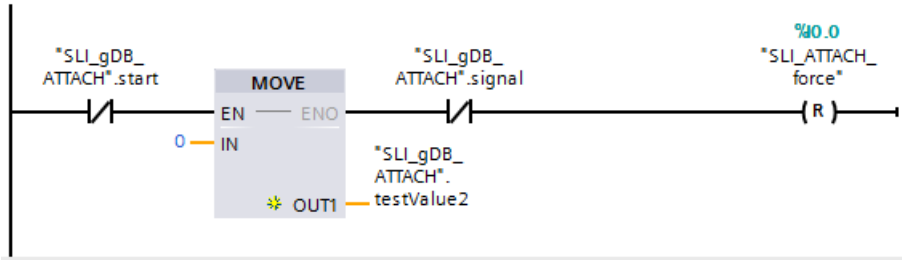

Create the following interconnection in OB40:

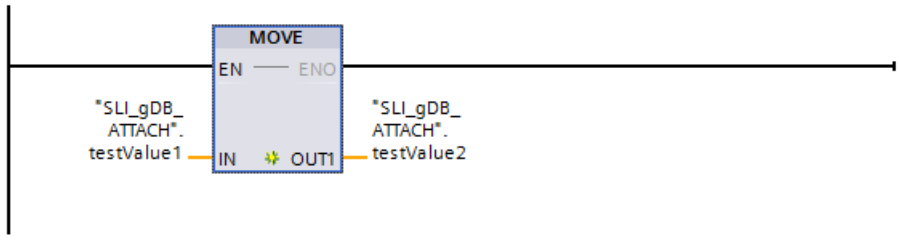

Processing

When the normally open contact ("start") supplies the signal state "TRUE", the "ATTACH" instruction is executed and the event is assigned to the OB. With the value "FALSE" of the input parameter ADD ("add"), the event replaces all previous event assignments for OB40. The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

When the normally open contact ("signal") in network 2 supplies the signal state "TRUE", the PLC tag "SLI_ATTACH_force" is set to the signal state "TRUE" and OB40 is executed. While OB40 is being called, the value of "testValue1" is transferred to "testValue2".

When the normally open contacts ("start" and "signal") supply the signal state "FALSE", the values of "testValue2" and "SLI_ATTACH_force" are reset in network 4.

SLI_gDB_ATTACH

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	obNbr	OB_ATT	40	40
4	add	Bool	0	FALSE
5	event	EVENT_ATT	16#C0000101	16#C000_0101
6	returnValue	Int	0	0
7	testValue1	Int	123	123
8	testValue2	Int	0	123
9	signal	Bool	0	TRUE
10	scannedSignal	Bool	false	TRUE

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

DETACH: Detach an OB from an interrupt event (S7-1200, S7-1500)

Description

You use this instruction to cancel the existing assignment of an organization block to one or more hardware interrupt events during runtime.

You enter the symbolic or numeric name of the organization block in the OB_NR parameter. This assignment to the event specified in the EVENT parameter will be canceled.

- If you have selected an individual hardware interrupt event at the EVENT parameter, the assignment of the OB to this hardware interrupt event is cancelled. All other currently existing assignments remain active. You can select an individual hardware interrupt event using the drop-down list of the operand placeholder.
- If you have not selected a hardware interrupt event, all the events currently assigned to this OB_NR organization block are separated.

Parameters

The following table shows the parameters of the "DETACH" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_ATT	I, Q, M, D, L or constant	Organization block (numbers up to 32767 are supported.)
EVENT	Input	EVENT_ATT	I, Q, M, D, L or constant	Hardware interrupt event
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0	No error
1	No assignment exists (warning)
8090	OB does not exist
8091	OB has incorrect type
8093	Event does not exist
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you assign a hardware interrupt event to a hardware interrupt OB (OB41), and dissolve the assignment.

Requirements:

A hardware interrupt OB (OB41 with the name "SLI_iOB_DETACH") has already been created. You create an input module (DI) and configure a hardware interrupt event for a positive edge under "Properties > Inputs 0-15 > ... > Channel 1".

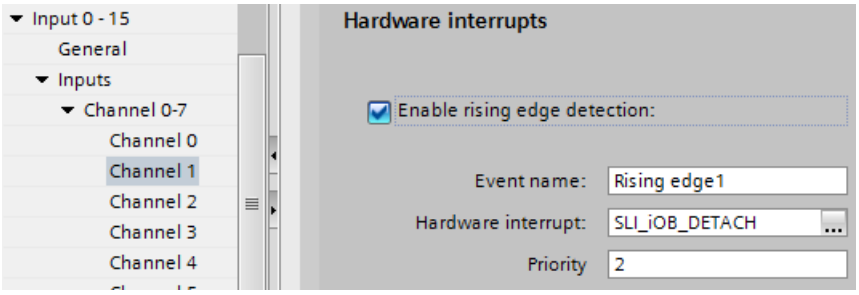

The following hardware ID of the configured hardware interrupt event ("Rising edge 1") is stored at the input parameter EVENT ("event").

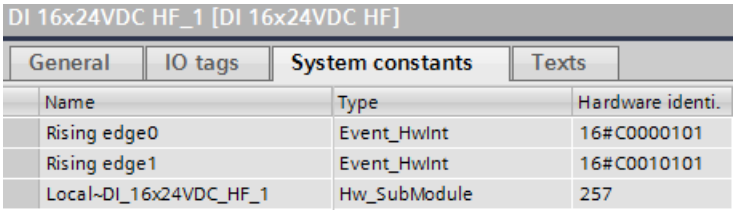

Name	Type	Hardware identi.
Rising edge0	Event_HwInt	16#C0000101
Rising edge1	Event_HwInt	16#C0010101
Local~DI_16x24VDC_HF_1	Hw_SubModule	257

Data storage:

Create ten tags in a global data block for storing the data. You store the number of the OB to be assigned (OB41) with the input parameter OB_NR ("obNbr") of the instructions "ATTACH" and "DETACH".

SLI_gDB_DETACH

	Name	Data type	Start value
1	Static		
2	startAttach	Bool	false
3	add	Bool	0
4	retValAtt	Int	0
5	obNbr	OB_ATT	41
6	event	EVENT_ATT	16#C0010101
7	retValDet	Int	0
8	testValue1	Int	125
9	testValue2	Int	0
10	signal	Bool	0
11	scannedSignal	Bool	false

To call the hardware interrupt event, you create a tag table with a PLC tag (address "I1.0").

SLI_Var_DETACH

	Name	Data type	Address
1	SLI_DETACH_event	Bool	%I1.0

Interconnection:

To call the instructions, create an FB ("SLI_FB_DETACH").

Network 1: Interconnect the parameters of the "ATTACH" instruction as follows.

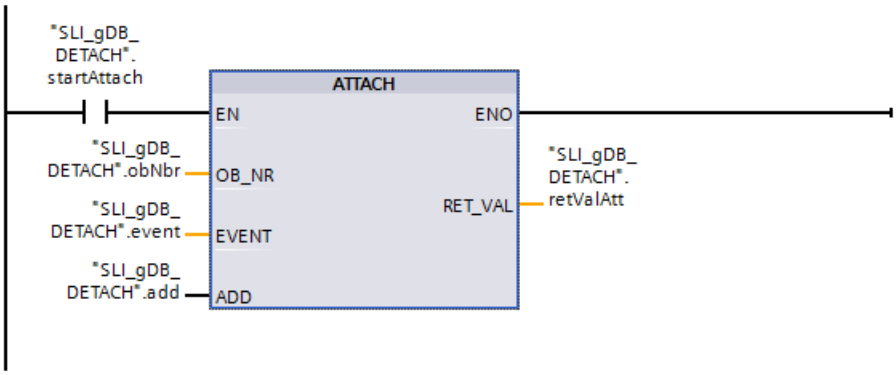

Network 2: Interconnect the PLC tag as follows to call the hardware interrupt event.

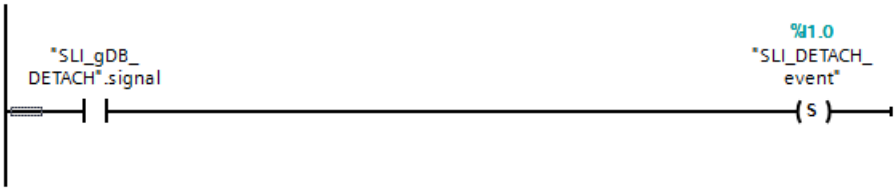

Network 3: You forward the call of the hardware interrupt event as follows.

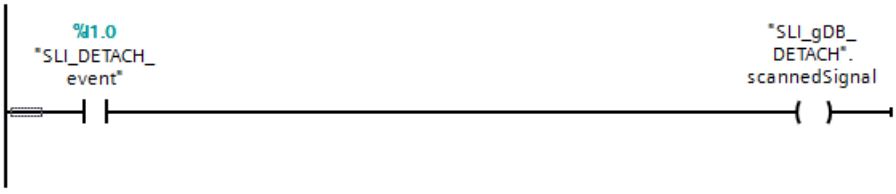

Network 4: Interconnect the parameters of the "DETACH" instruction as follows.

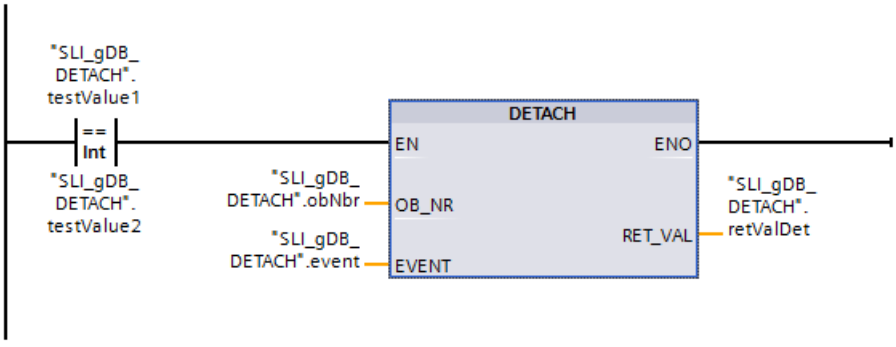

Network 5: To reset, interconnect "testValue2" and "SLI_DETACH_event" as follows.

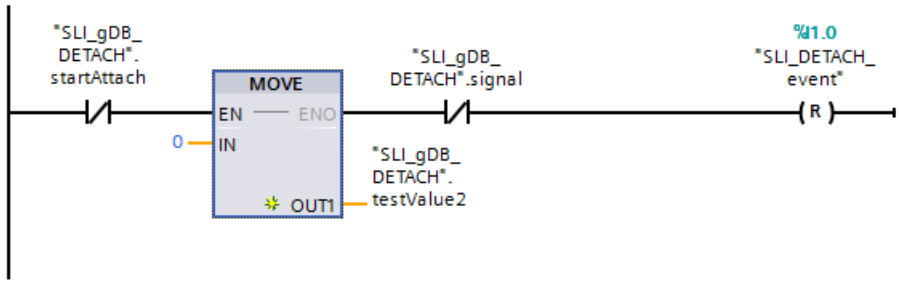

Create the following interconnection in OB41:

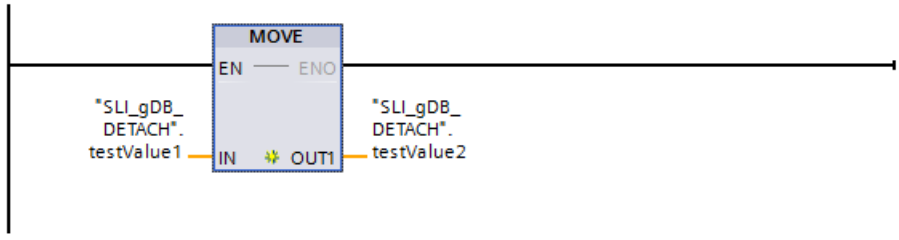

Processing

When the normally open contact ("startAttach") supplies the signal state "TRUE", the "ATTACH" instruction is executed and the event is assigned to OB41. With the value "FALSE" of the input parameter ADD ("add"), the event replaces all previous event assignments for OB41. The output parameter RET_VAL ("retValAtt") indicates that processing took place without errors.

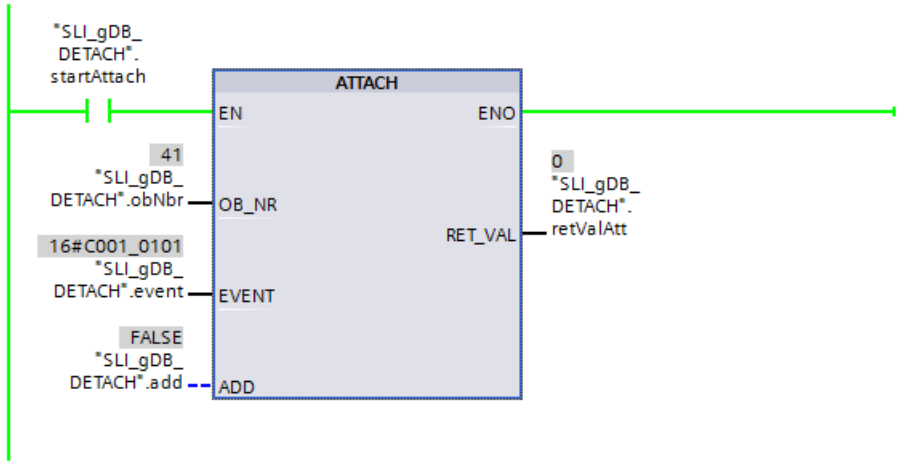

When the normally open contact ("signal") in network 2 supplies the signal state "TRUE", the hardware interrupt event ("SLI_DETACH_event") is set to "TRUE". The OB41 is called and the value of "testValue1" is transferred to "testValue2" in OB41. As long as the hardware interrupt event returns the signal state "TRUE", the OB41 is called and displayed in network 3 by the coil ("scannedSignal").

SLI_gDB_DETACH				
	Name	Data type	Start value	Monitor value
1	Static			
2	startAttach	Bool	false	TRUE
3	add	Bool	0	FALSE
4	retValAtt	Int	0	0
5	obNbr	OB_ATT	41	41
6	event	EVENT_ATT	16#C0010101	16#C001_0101
7	retValDet	Int	0	0
8	testValue1	Int	125	125
9	testValue2	Int	0	125
10	signal	Bool	0	TRUE
11	scannedSignal	Bool	false	TRUE

When "testValue1" and "testValue2" supply the same value, the instruction "DETACH" is executed and the event is separated from OB41. The output parameter RET_VAL ("retValDet") indicates that processing took place without errors.

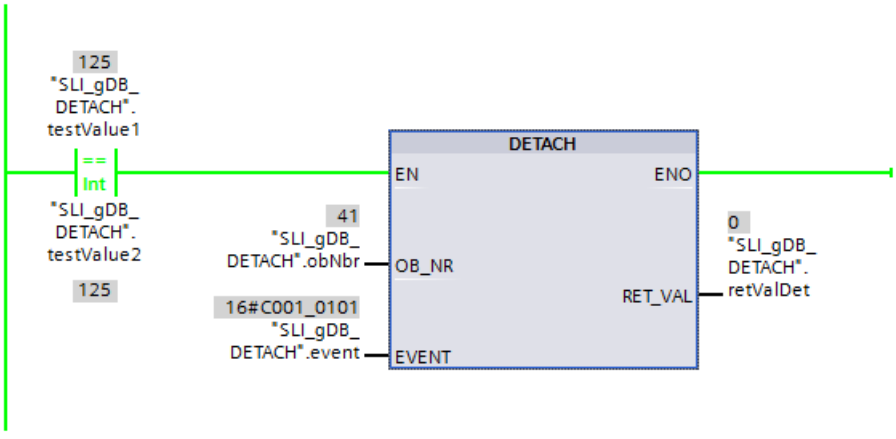

When the tag "startAttach" supplies the signal state "FALSE", the values of "testValue2" and the PLC tag are reset in network 5.

You can find additional information and the program code for the above-named example here:Sample Library for Instructions (Page 593)

See also

Switching display formats in the program status (Page 7698)

Cyclic interrupt

SET_CINT: Set cyclic interrupt parameters (S7-1200, S7-1500)

Description

You use this instruction to set the parameters for a cyclic interrupt OB. The start time for a cyclic interrupt OB is generated from the respective time interval of the OB and the phase offset.

- The time interval of an OB is the interval at which the OB is periodically called. For example, if the time interval is 100 µs, the OB will be called every 100 µs during program execution.
- The phase offset is a time interval by which the call of a cyclic interrupt OB is offset. You can use the phase offset to process low priority organization blocks in a precise time base.

If the OB does not exist or if the time interval used is not supported, a corresponding error alarm is output in the RET_VAL parameter.

A time interval in the CYCLE parameter of "0" means that the OB will not be called.

Functional description

If a lower priority OB and a higher priority OB are called in the same time interval, the lower priority OB will only be called once the higher priority OB has been executed. The call time for the lower priority OB can be offset according to the length of time to execute the higher-priority OB.

OB call without phase offset

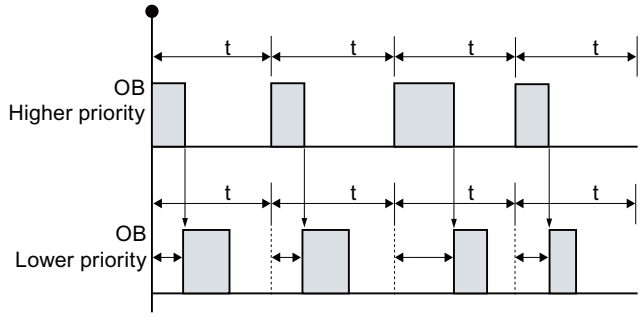

If a phase offset is configured for the lower priority OB and the phase offset is greater than the current execution time of the respective higher priority OB, then the block will be called in a fixed time base.

OB call with phase offset

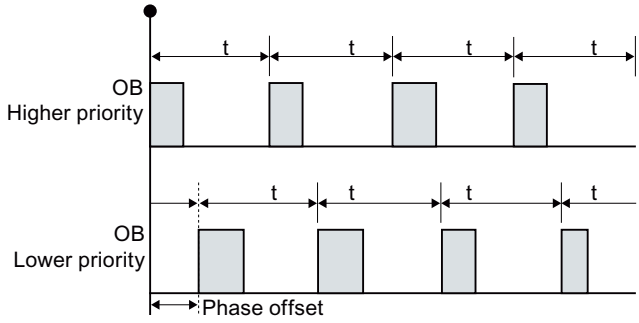

Parameters

The following table shows the parameters of the "SET_CINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_CYCLIC	I, Q, M, D, L or constant	OB number (<32768)
CYCLE	Input	UDINT	I, Q, M, D, L or constant	Time interval in microseconds
PHASE	Input	UDINT	I, Q, M, D, L or constant	Phase offset in microseconds
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0	No error
8090	OB does not exist or is of the wrong type
8091	Incorrect time interval
8092	Incorrect phase offset
80B2	No event assigned to OB
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for cyclic interrupt functions (Page 2446).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

QRY_CINT: Query cyclic interrupt parameters (S7-1200, S7-1500)

Description

The instruction QRY_CINT returns the following parameters of a cyclic interrupt OB:

- Cycle time
- Phase offset
- Status of the cyclic interrupt (enabled, delayed, expired, etc.)

The cyclic interrupt OB is identified using the OB_NR parameter.

The values of the queried cyclic interrupt parameters correspond to those at the time the "QRY_CINT" instruction is executed.

Parameters

The following table shows the parameters of the "QRY_CINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_CYCLIC (INT)	I, Q, M, D, L or constant	OB number (<32768) or symbolic addressing via the name of the OB (e.g. OB_MyOB)
CYCLE	Output	UDINT	I, Q, M, D, L	Cycle time in microseconds
PHASE	Output	UDINT	I, Q, M, D, L	Phase offset
STATUS	Output	WORD	I, Q, M, D, L	Status of the cyclic interrupt: • Bit 0 to bit 4: see parameter STATUS • Other bits: Always "0"
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

Bit	Value	Meaning
0	0	Not used (always "0").
1	0	The cyclic interrupt is enabled.
	1	The cyclic interrupt is delayed.
2	0	The cyclic interrupt is not enabled or has expired.
	1	The cyclic interrupt is enabled.
3	0	Not used (always "0").

Bit	Value	Meaning
4	0	An OB with the specified number does not exist.
	1	An OB with the specified number exists.
Other bits		Not used (always "0").

Parameter RET_VAL

If an error occurs, the relevant error code is displayed in the RET_VAL parameter and the STATUS parameter is set to "0".

Error code* (W#16#....)	Description
0	No error.
8090	OB does not exist or is of the wrong type
80B2	No result assigned to OB.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for cyclic interrupt functions (Page 2446).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Program example for cyclic interrupt functions

Introduction

In the following example, you read the phase and the cycle time of a cyclic interrupt OB (OB31) and set a new cycle time.

The OB31 has the following defaults:

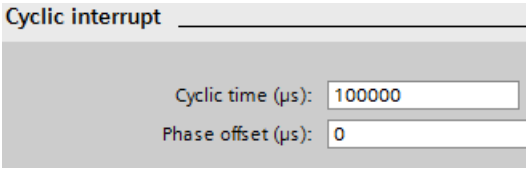

Requirement

- To call, create an OB31.
- Create twelve tags in a global data block for storing the data.

SLI_gDB_SET_CINT			
	Name	Data type	Start value
1	Static		
2	executeSet	Bool	false
3	obNbr	OB_CYCLIC	31
4	setCycle	UDInt	1000
5	setPhase	UDInt	0
6	returnValueSet	Int	0
7	testTime	Time	T#0ms
8	testTimeLimit	Time	T#2m
9	executeQry	Bool	false
10	returnValueQry	Int	0
11	itemiseStatus	Struct	
12	Cyclic_interrupt_enabled	Bool	false
13	Cyclic_interrupt_not_expired	Bool	false
14	OB_with_obNbr_exists	Bool	false
15	qryCycle	UDInt	0
16	qryPhase	UDInt	0

Interconnect parameters: "SLI_itemiseStatus_CINT" FC

To evaluate the status information, create an FC "SLI_itemiseStatus_CINT".

Create the following local tag in it.

SLI_itemiseStatus_CINT		
	Name	Data type
1	Input	
2	inWord	Word

Create interconnections for all relevant bits.

Network 1:

Network 2:

Network 3:

Interconnect parameters: "SLI_FB_SET_CINT" FB

To call the instructions, create an FB ("SLI_FB_SET_CINT"). Create the following local tag in the function block "SLI_FB_SET_CINT".

7		Static		
8		statWord	Word	16#0

Network 1: Interconnect the parameters of the "QRY_CINT" instruction as follows.

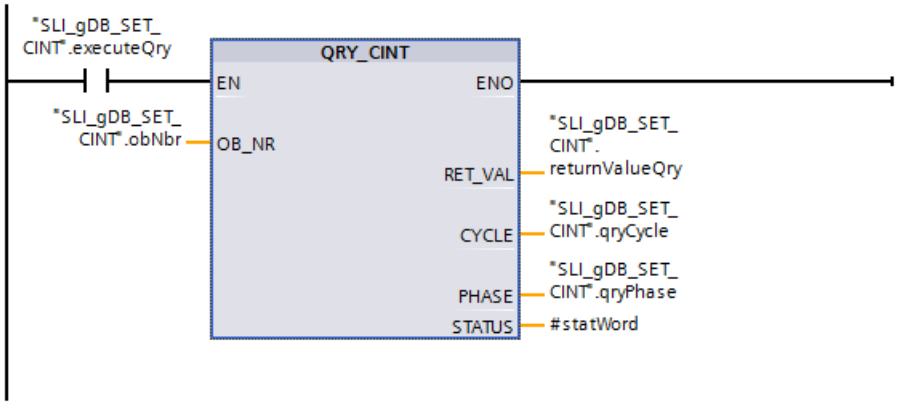

Network 2: Interconnect the parameters of the "SET_CINT" instruction as follows.

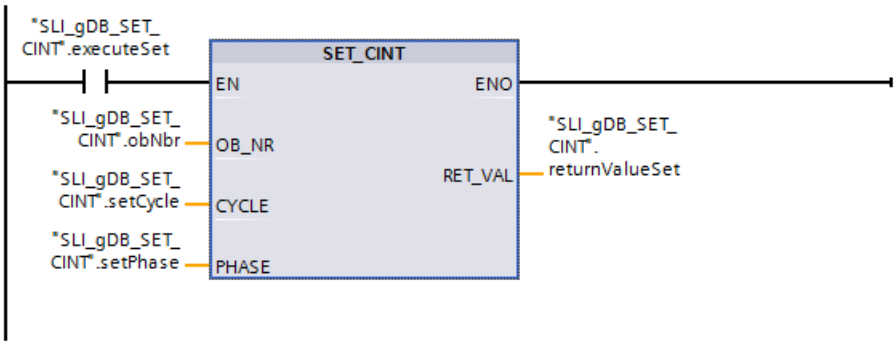

Network 3: Interconnect the parameter of the "SLI_itemiseStatus_CINT" FC as follows.

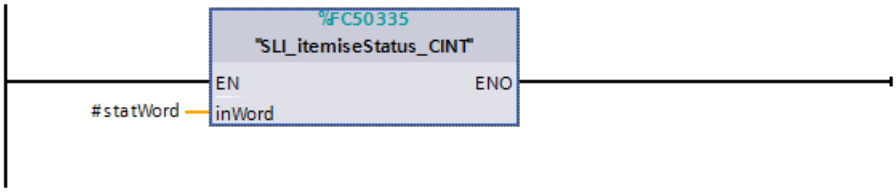

Interconnect parameters: OB31

Create the following interconnections in OB31.

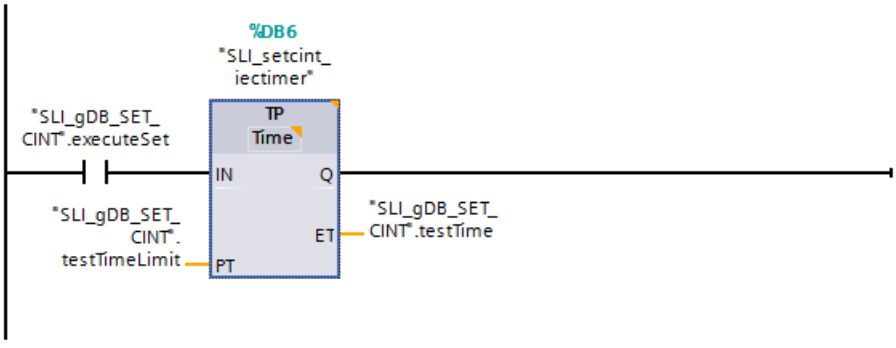

Result of "QRY_CINT"

The number of the OB to be called (OB31) is stored with the input parameter OB_NR ("obNbr") of the instructions "QRY_CINT" and "SET_CINT".

When the normally open contact ("executeQry") supplies the signal state "TRUE", the "QRY_CINT" instruction is executed and the data of OB31 is read.

The cycle time is output at the output parameter CYCLE ("qryCycle"). The phase offset is output at the output parameter PHASE ("qryPhase").
The output parameter STATUS ("#statWord") displays the status of OB31. The output parameter RET_VAL ("returnValueQry") indicates that processing is taking place without errors.

Thee current status information ("#statWord") of the OB31 is interpreted in network 3 via the FC "SLI_itemiseStatus_CINT" and output via the structure "itemiseStatus".

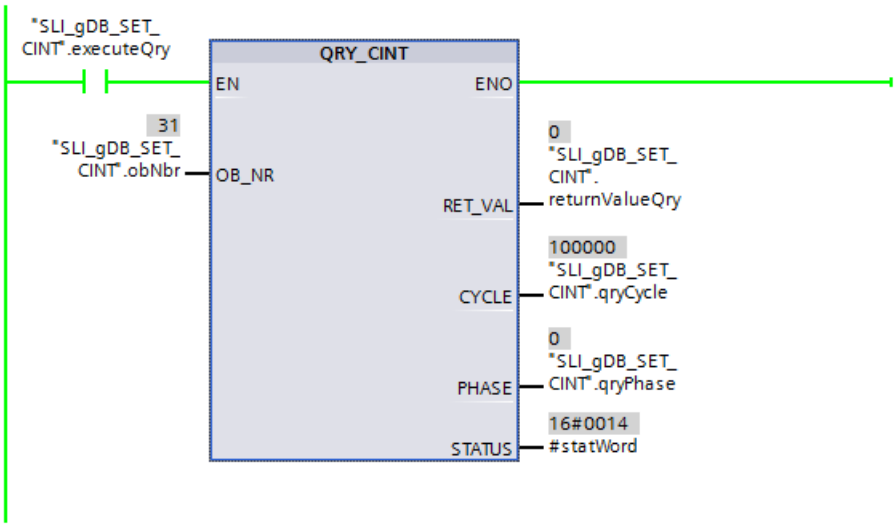

The value of the tag "#statWord" is mapped in the structure "itemiseStatus".

12	itemiseStatus	Struct		
13	Cyclic_interrupt_enabled	Bool	false	TRUE
14	Cyclic_interrupt_not_expired	Bool	false	TRUE
15	OB_with_obNbr_exists	Bool	false	TRUE

Result of "SET_CINT"

When the normally open contact ("executeSet") supplies the signal state "TRUE", the "SET_CINT" instruction is executed. A new cycle time is transferred to OB31 with the input parameter CYCLE ("setCycle") of the instruction "SET_CINT". The value "0" is stored as phase offset at the input parameter PHASE ("setPhase").

The output parameter RET_VAL ("returnValueSet") indicates that processing took place without errors.

SLI_gDB_SET_CINT

	Name	Data type	Start value	Monitor value
1	Static			
2	executeSet	Bool	false	TRUE
3	obNbr	OB_CYCLIC	31	31
4	setCycle	UDInt	1000	1000
5	setPhase	UDInt	0	0
6	returnValueSet	Int	0	0
7	testTime	Time	T#0ms	T#14S_487MS
8	testTimeLimit	Time	T#2m	T#2M
9	executeQry	Bool	false	TRUE
10	returnValueQry	Int	0	0
11	itemiseStatus	Struct		
12	Cyclic_interrupt_enabled	Bool	false	TRUE
13	Cyclic_interrupt_not_expired	Bool	false	TRUE
14	OB_with_obNbr_exists	Bool	false	TRUE
15	qryCycle	UDInt	0	1000
16	qryPhase	UDInt	0	0

After execution of the instruction "SET_CINT", OB31 retains the new cyclic interrupt data until STOP of the CPU. After the execution, you can set the normally open contact ("executeSet") to the signal state "FALSE" once again. The default properties of OB31 are not affected by the execution of "SET_CINT".

Result of the OB31

When the normally open contact ("executeSet") supplies the signal state "TRUE", an IEC timer "TP" is started in OB31. The IEC timer stops after 2 minutes.

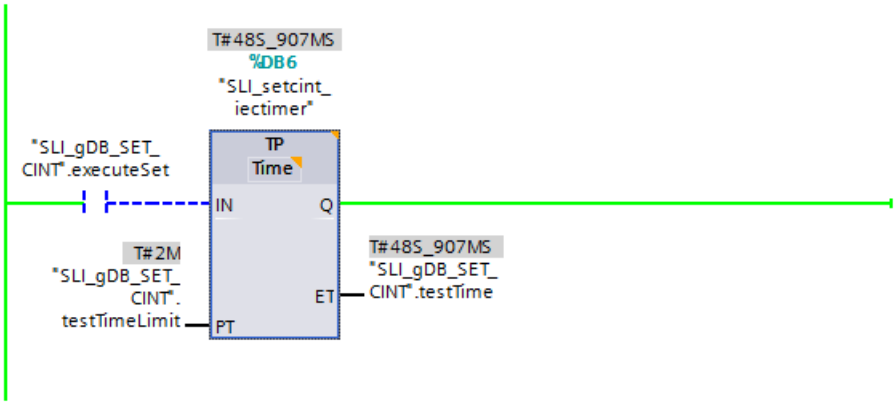

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Time-of-day interrupt

SET_TINT: Set time-of-day interrupt (S7-1500)

Description

The instruction "SET_TINT" is used to set the start data and time of the time-of-day interrupt organization blocks from the user program, without having to make settings in the hardware configuration.

- Enter the number of the time-of-day interrupt-OBs at the parameter OB_NR for which you would like to set the start date and the time-of-day.
- Use the parameter SDT and PERIOD to specify when and how often the time-of-day interrupt OB should be called:
  - One-time call: Enter the date and the time-of-day at the parameter SDT. Use the value "0" at parameter PERIOD.
  - Repeated call: Enter the date and the time-of-day of the first call at the parameter SDT. Use the parameter PERIOD to define the time interval in which subsequent calls of the OBs should take place.

Observe the following when setting the start date and start time:

- The date and the time-of-day specification at the parameter SDT refers to the system time.
- The seconds and milliseconds specification is ignored and set to "0" at the start time.
- Only the starting date days 1, 2, ... 28 are possible if you want to specify a monthly time-of-day interrupt OBs. This restriction prevents skipping the monthly call (e.g. 30-day months or in February).
  The setting "end of the month" (W#16#2001) can be used at the parameter PERIOD as an alternative to the 29th 30th and the 31st of the month.

After setting the time-of-day interrupt with "SET_TINT" it still must be activated with the instruction "ACT_TINT".

Note

Additional information about time-of-day interrupt OBs

Other special features for the use of time-of-day interrupt OBs can be found in the descriptions of the organization blocks of the respective CPU:

For the S7-1200: AUTOHOTSPOT

For the S7-1500: AUTOHOTSPOT

The settings at the parameters SDT and PERIOD coincide with the settings for time-of-day interrupt in the properties of the time-of-day interrupt OBs.

Parameters

The following table shows the parameters of the "SET_TINT" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OBs • The numbers 10 to 17 are available for the time-of-day interrupt OBs. • An OB number starting with 123 can also be assigned as an alternative. The OB number is displayed in the program block folder and in the system constants.
SDT	Input	DT	D, L or constant	Start date and start time

Parameters	Declaration	Data type	Memory area	Description
PERIOD	Input	WORD	I, Q, M, D, L or constant	Execution intervals from the starting point SDT to: • W#16#0000 = single execution • W#16#0201 = once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the actual parameter of RET_VAL contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR (no time-of-day interrupt OB addressed).
8091	Error at the parameter SDT (invalid date and time-of-day specification).
8092	Incorrect input at the parameter PERIOD.
80A1	The set start time is in the past. This error code only occurs at PERIOD = W#16#0000.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for time-of-day interrupt functions (Page 2460).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

ACT_TINT: Enable time-of-day interrupt (Page 2457)

SET_TINTL: Set time-of-day interrupt (S7-1200, S7-1500)

Description

The instruction "SET_TINTL" is used to set the start data and time of the time-of-day interrupt organization blocks from the user program, without having to make settings in the hardware configuration.

- Enter the number of the time-of-day interrupt-OBs at the parameter OB_NR for which you would like to set the start date and the time-of-day.
- Use the parameter SDT and PERIOD to specify when and how often the time-of-day interrupt OB should be called:
  - One-time call: Enter the date and the time-of-day at the parameter SDT. Use the value "0" at parameter PERIOD.
  - Repeated call: Enter the date and the time-of-day of the first call at the parameter SDT. Use the parameter PERIOD to define the time interval in which subsequent calls of the OBs should take place.
- Use the parameter LOCAL to select whether the time specified at the parameter SDT refers to the local time or the system time.
- With the ACTIVATE parameter, you specify when the settings made for the organization block are to be applied:
  - ACTIVATE = true: The settings made are applied directly.
  - ACTIVATE = false: The settings are applied only after "ACT_TINT (Page 2457)" is called:

Observe the following when setting the start date and start time:

- The seconds and milliseconds specification is ignored and set to "0" at the start time.
- Only the starting date days 1, 2, ... 28 are possible if you want to specify a monthly time-of-day interrupt OBs. This restriction prevents skipping the monthly call (e.g. 30-day months or in February).
  The setting "end of the month" (W#16#2001) can be used at the parameter PERIOD as an alternative to the 29th 30th and the 31st of the month.

Please observe the following during the use of local time:

- Changeover from daylight saving to standard time: When calling time-of-day interrupt organization blocks with a start time within the second hour during changeover from daylight saving time to standard time, also use a time-delay interrupt during the first hour of the time changeover.
- Changeover from standard to daylight saving time: If you specify the skipped hour as the time-of-day for the daylight saving time changeover day, the error code 16#8091 will be output at a single execution (PERIOD = W#16#0000).

Note

Additional information about time-of-day interrupt OBs

Other special features for the use of time-of-day interrupt OBs can be found in the descriptions of the organization blocks of the respective CPU:

For the S7-1200: AUTOHOTSPOT

For the S7-1500: AUTOHOTSPOT

The settings at the parameters SDT and PERIOD coincide with the settings for time-of-day interrupt in the properties of the time-of-day interrupt OBs.

Parameters

The following table shows the parameters of the "SET_TINTL" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OBs • The numbers 10 to 17 are available for the time-of-day interrupt OBs. • An OB number starting with 123 can also be assigned as an alternative. The OB number is displayed in the program block folder and in the system constants.
SDT	Input	DTL	D, L or constant	Start date and start time
LOCAL	Input	BOOL	I, Q, M, D, L or constant	• true: Use local time • false: Use system time
PERIOD	Input	WORD	I, Q, M, D, L or constant	Execution intervals from the starting point SDT to: • W#16#0000 = single execution • W#16#0201 = Once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end
ACTIVATE	Input	BOOL	I, Q, M, D, L or constant	• true: Set and activate the time-of-day interrupt • false: Set the time-of-day interrupt and activate only at call of "ACT_TINT (Page 2457)"
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the actual parameter of RET_VAL contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR (no time-of-day interrupt OB addressed).
8091	Error at the parameter SDT (invalid date and time-of-day specification).
8092	Incorrect input at the parameter PERIOD.
80A1	The set start time is in the past. This error code only occurs at PERIOD = W#16#0000.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

CAN_TINT: Cancel time-of-day interrupt (S7-1200, S7-1500)

Description

The instruction "CAN_TINT" is used to delete the start date and time of a specified time-of-day interrupt organization block. This deactivates the time-of-day interrupt, and the organization block is no longer called.

If you want to reuse the time-of-day interrupt, the start time must be reset (instruction "SET_TINTL (Page 2454)" or "SET_TINT (Page 2451)").

After this, you must reactivate the time-of-day interrupt:

- If you use the instruction "SET_TINT (Page 2451)" or "SET_TINTL (Page 2454)" with parameter ACTIVE=false to set the time-of-day interrupt, call "ACT_TINT (Page 2457)".
- By using the instruction "SET_TINTL (Page 2454)" you can also activate the time-of-day interrupt directly with the parameter ACTIVE=true.

Parameters

The following table shows the parameters of the "CAN_TINT" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OB, whose starting date and time-of-day are to be deleted.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the actual parameter of RET_VAL contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR.
80A0	No starting date/time-of-day defined for the affected time-of-day interrupt OB.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for time-of-day interrupt functions (Page 2460).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

ACT_TINT: Enable time-of-day interrupt (S7-1200, S7-1500)

Description

The instruction "ACT_TINT" is used to activate a time-of-day interrupt organization block from the user program. A prerequisite for the execution of the instruction is that the starting date and time-of-day are already set for the time-of-day interrupt OB.

Parameters

The following table shows the parameters of the "ACT_TINT" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OBs • The numbers 10 to 17 are available for the time-of-day interrupt OBs. • An OB number starting with 123 can also be assigned as an alternative. The OB number is displayed in the program block folder and in the system constants.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the actual parameter of RET_VAL contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR (no time-of-day interrupt OB addressed).
80A0	Start date and time-of day not set for the relevant time-of-day interrupt OB.
80A1	The activated time is in the past. If the time-of-day interrupt is to be executed only once, only this error occurs.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for time-of-day interrupt functions (Page 2460).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

QRY_TINT: Query status of time-of-day interrupt (S7-1200, S7-1500)

Description

You can use this instruction to display the status of a time-of-day interrupt organization block in the STATUS output parameter.

Parameters

The following table shows the parameters of the "QRY_TINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OB whose status is queried The OB number is displayed in the program block folder and in the system constants.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the actual parameter of RET_VAL contains an error code.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-of-day interrupt (see below)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred. Exception: Status message "0" on bit 4 (no OB with this number created).
8090	Error at the parameter OB_NR. Possible causes: • The value at the parameter OB_NR is not an OB number (<1 or > 32767) supported by the CPU. • The OB number does not address a time-of-day interrupt OB.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Parameter STATUS

If an error occurs (see RET_VAL parameter), "0" is output in the STATUS parameter.

Bit	Value	Meaning
0	Always "0"	Irrelevant
1	0	The time-of-day interrupt is enabled.

Bit	Value	Meaning
	1	The time-of-day interrupt is disabled.
2	0	Time-of-day interrupt is not activated or has elapsed.
	1	The time-of-day interrupt is activated.
4	0	An OB with an OB number as specified at OB_NR parameter does not exist.
	1	An OB with an OB number as specified at OB_NR parameter does exist.
6	0	The time-of-day interrupt is based on the system time
	1	The time-of-day interrupt is based on the local time
Other		Always "0"

Example

You can find the example here: Program example for time-of-day interrupt functions (Page 2460).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Program example for time-of-day interrupt functions

Introduction

In the following example, you read the status of a time-of-day interrupt OB (OB10), and set a start time as well as the call frequency. Once OB10 has been started, disable OB10 once again.

The OB10 has the following defaults:

Requirement

Create 13 tags and a structure in a global data block for storing the data.

SLI_gDB_SET_TINT			
	Name	Data type	Start value
1	Static		
2	obNbr	OB_TOD	10
3	executeQry	Bool	false
4	itemiseStatus	Struct	
5	T-o-d_interrupt_enabled	Bool	false
6	T-o-d_interrupt_activated	Bool	false
7	OB_with_obNbr_exists	Bool	false
8	T-o-d_interrupt_on_system_time	Bool	false
9	T-o-d_interrupt_on_local_time	Bool	false
10	returnValueQry	Int	0
11	executeSet	Bool	false
12	startDateTime	Date_And_Time	DT#1990-01-01-0
13	presetTime	Time	T#5M
14	periodOfExecute	Word	16#0201
15	returnValueSet	Int	0
16	executeAct	Bool	false
17	returnValueAct	Int	0
18	executeCan	Bool	false
19	returnValueCan	Int	0
20	testCurrCount	Int	0

Interconnect parameters: "SLI_itemiseStatus_TINT" FC

To evaluate the status information, create an FC "SLI_itemiseStatus_TINT".

Create the following local tag in it.

SLI_itemiseStatus_TINT		
	Name	Data type
1	Input	
2	inWord	Word

Create interconnections for all relevant bits.

Network 1:

Network 2:

Network 3:

Network 4:

Network 5:

Interconnect parameters: "SLI_retrievetime_TINT" FC

To read the date (system time) and define the start time of OB10, create an FC "SLI_retrievetime_TINT". Create the following local tags in the FC.

7		Temp	
8		statTIME	Date_And_Time
9		statRetVal	Int

Network 1: Interconnect the parameters as follows. "RD_SYS_T" reads the system time. Use "T_ADD" to add 5 minutes to the system time.

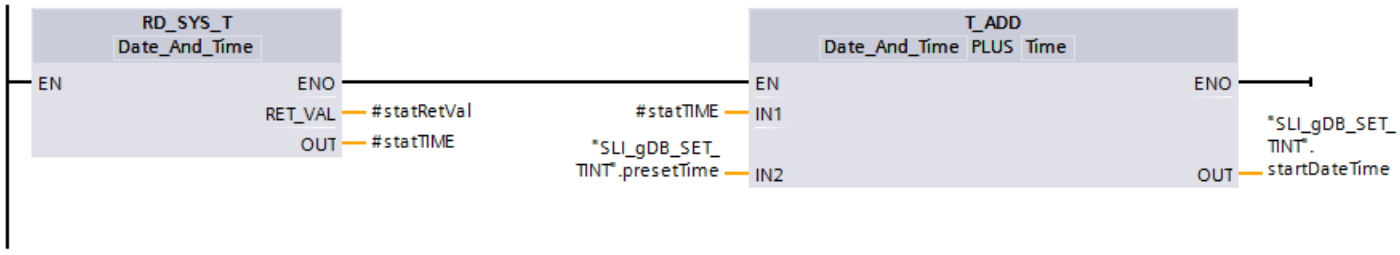

Interconnect parameters: "SLI_FB_SET_TINT" FB

To call the instructions, create an FB ("SLI_FB_SET_TINT"). Create the following local tags in the function block "SLI_FB_SET_TINT".

7		Static		
8		statMem	Bool	false
9		statWord	Word	16#0

Network 1: Interconnect the parameters of the "QRY_TINT" instruction as follows.

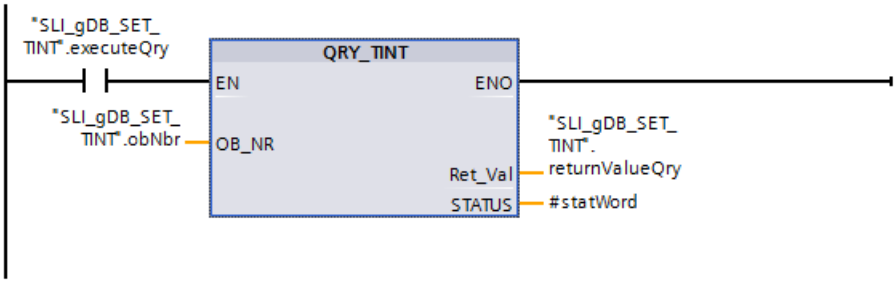

Network 2: You call the FC "SLI_retrievetime_TINT" in network 2 of the FB.

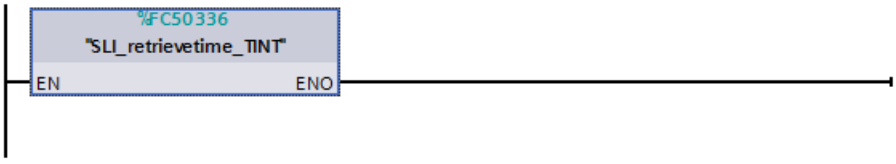

Network 3: Interconnect the parameters of the "SET_TINT" instruction as follows.

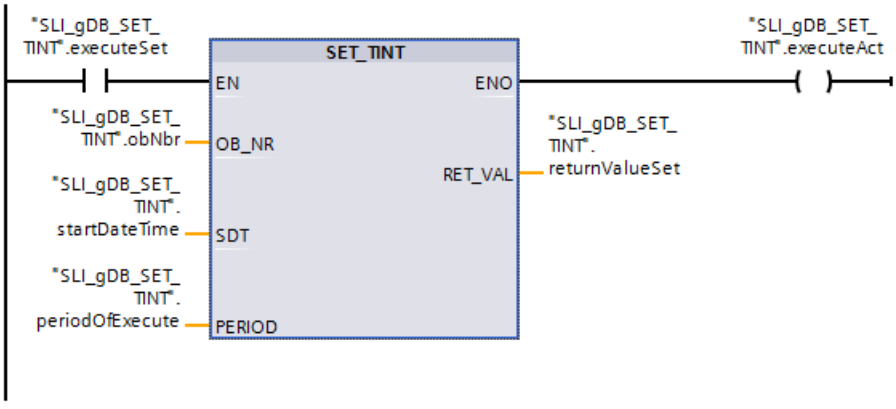

Network 4: Interconnect the parameters of the "ACT_TINT" instruction as follows.

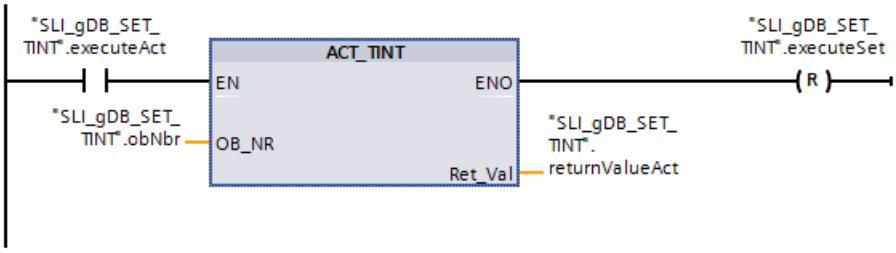

Network 5: Interconnect the parameters of the "CAN_TINT" instruction as follows.

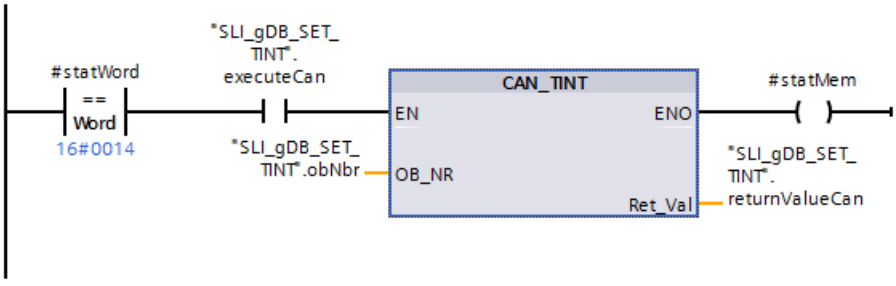

Network 6: Interconnect the parameters that are to be reset with the execution of "CAN_TINT" as follows.

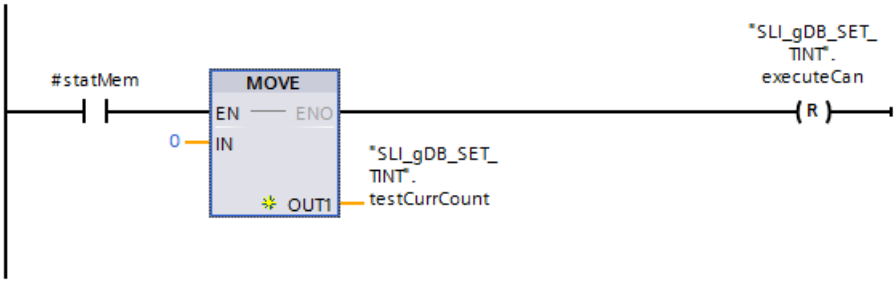

Network 7: Interconnect the parameter of the "SLI_itemiseStatus_TINT" FC as follows.

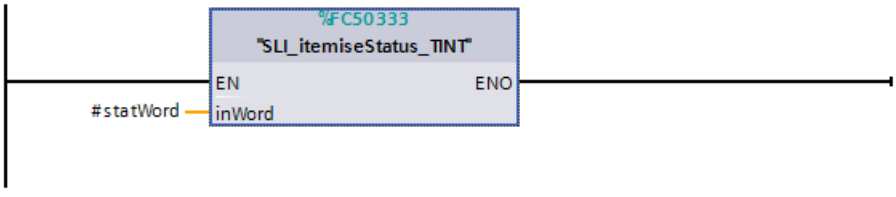

Interconnect parameters: OB10

Create the following interconnections in OB10.

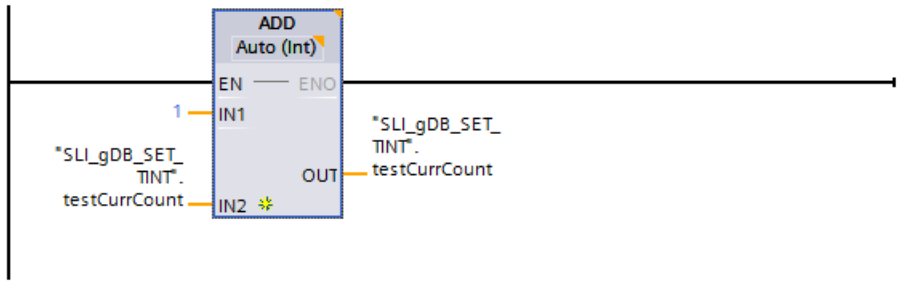

Result of "QRY_TINT"

The instructions receive the number of the OB to be called (OB10) with the input parameter OB_NR ("obNbr").

Network1:

When the normally open contact ("executeQry") supplies the signal state "TRUE", the "QRY_TINT" instruction is executed and the data of OB10 is read. The output parameter STATUS ("#statWord") displays the status of OB10. The output parameter RET_VAL ("returnValueQry") indicates that processing is taking place without errors.

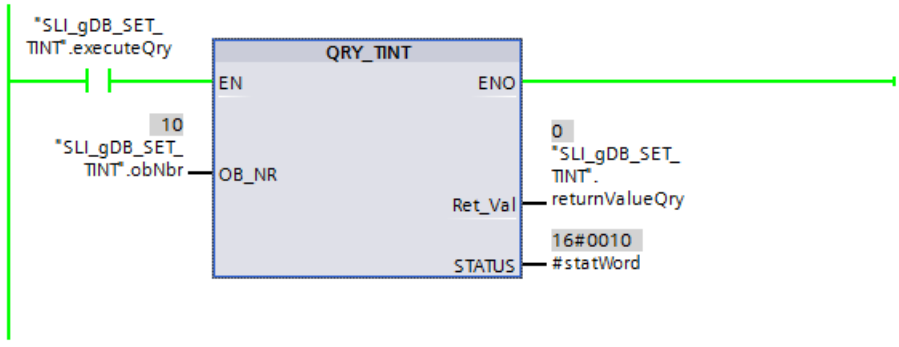

The value of the tag "#statWord" is mapped in the structure "itemiseStatus".

4	itemiseStatus	Struct		
5	T-o-d_interrupt_enabled	Bool	false	TRUE
6	T-o-d_interrupt_activated	Bool	false	FALSE
7	OB_with_obNbr_exists	Bool	false	TRUE
8	T-o-d_interrupt_on_system_time	Bool	false	TRUE
9	T-o-d_interrupt_on_local_time	Bool	false	FALSE

Result of "SET_TINT"

Network 3: When the normally open contact ("executeSet") supplies the signal state "TRUE", the "SET_TINT" instruction is executed. The input parameter SDT ("startDateTime") transfers the defined start time (see Network 2) to OB10. "Every minute" (value "16#0201") is stored as call frequency at the input parameter PERIOD ("periodOfExecute").

The output parameter RET_VAL ("returnValueSet") indicates that processing took place without errors.

The execution of SET_TINT sets the coil ("executeAct") to "TRUE" and the instruction "ACT_TINT" is executed in network 4.

Result of "ACT_TINT"

Network 4: The instruction "ACT_TINT" activates OB10. The output parameter RET_VAL ("returnValueAct") indicates that processing took place without errors. The reset ("executeSet") in the signal state "TRUE" automatically ends the execution of the SET_TINT instruction and therefore also the execution of the ACT_TINT instruction.

Note

After execution of the instruction "SET_TINT", OB10 retains the new call data until STOP of the CPU. The default properties of OB10 are not affected by the execution of the instruction "SET_TINT".

Result of the OB10

With each call of OB10, the instruction "ADD" is executed and the tag "testCurrCount" is incremented by "1". OB10 is called (according to the value of "periodOfExecute") once every minute.

SLI_gDB_SET_TINT

	Name	Data type	Start value	Monitor value
1	Static			
2	obNbr	OB_TOD	10	10
3	executeQry	Bool	false	TRUE
4	itemiseStatus	Struct		
5	T-o-d_interrupt_enabled	Bool	false	TRUE
6	T-o-d_interrupt_activated	Bool	false	TRUE
7	OB_with_obNbr_exists	Bool	false	TRUE
8	T-o-d_interrupt_on_system_time	Bool	false	TRUE
9	T-o-d_interrupt_on_local_time	Bool	false	FALSE
10	returnValueQry	Int	0	0
11	executeSet	Bool	false	FALSE
12	startDateTime	Date_And_Time	DT#1990-01-01-0	DT#2016-11-30-17...
13	presetTime	Time	T#5M	T#5M
14	periodOfExecute	Word	16#0201	16#0201
15	returnValueSet	Int	0	0
16	executeAct	Bool	false	FALSE
17	returnValueAct	Int	0	0
18	executeCan	Bool	false	FALSE
19	returnValueCan	Int	0	0
20	testCurrCount	Int	0	5

Result of "CAN_TINT"

Network 5: When OB10 is set and enabled ("#statWord" has the value "16#0014"), the comparator supplies the signal state "TRUE" and CAN_TINT can be executed.

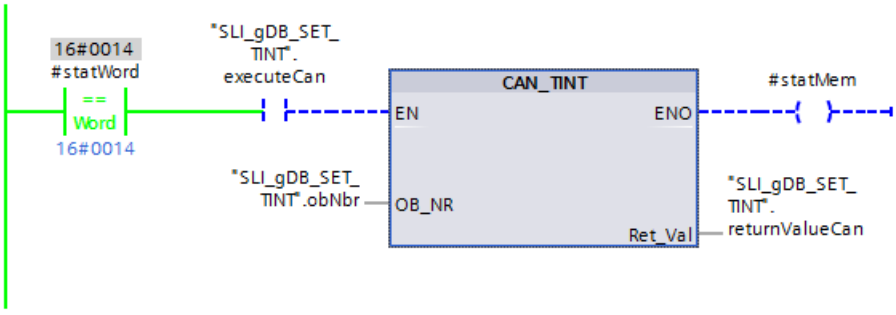

When the normally open contact ("executeCan") supplies the signal state "TRUE", the "CAN_TINT" instruction is executed. The instruction disables OB10 and deletes the call data of OB10 set with SET_TINT. The output parameter RET_VAL ("returnValueCan") indicates that processing took place without errors.

The execution of CAN_TINT sets the coil ("#statMem") to "TRUE" and network 6 is executed.

Result of the reset

Network 6: When the normally open contact ("#statMem") supplies the signal state "TRUE", the tag "testCurrCount" is reset to the value "0" with the instruction "MOVE". Reset ("executeCan") in the signal state "TRUE" also automatically ends the execution of the instruction "CAN_TINT".

Result of the status information "#statWord"

The FC "SLI_itemiseStatus_TINT" is called in network 7. The significant bits of the tag "#statWord" are evaluated and output as Boolean tags with plain-text name. When a tag of the structure "itemiseStatus" supplies the signal state "TRUE", its statement is correct.

Additional information

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Time-delay interrupt

Using time-delay interrupts

Definition

After you have called the "SRT_DINT (Page 2469)" instruction, the operating system generates an interrupt after the specified delay time has elapsed. The configured time-delay interrupt OB is called.

Prerequisites for the call

Before a time-delay interrupt can be called by the operating system, the following conditions must be met:

- The time-delay interrupt OB must be started by the "SRT_DINT (Page 2469)" instruction.
- The time-delay interrupt OB must not be deselected during configuration.
- The time-delay interrupt OB must exist in the CPU.

Purpose of the instructions "SRT_DINT", "CAN_DINT" and "QRY_DINT"

You use the instructions to

- Start time-delay interrupts ("SRT_DINT (Page 2469)")
- Cancel time-delay interrupts ("CAN_DINT (Page 2471)")
- Query time-delay interrupts ("QRY_DINT (Page 2471)")

Effects on the time-delay interrupt

The following table lists a number of different situations and explains the effect they have on a time-delay interrupt.

If ...	and ...	Then ...
A time-delay interrupt is started (by calling "SRT_DINT (Page 2469)")	The time-delay interrupt has already started,	The delay time is overwritten; the time-delay interrupt is started again.
	The time-delay interrupt OB does not exist at the time of the call,	a diagnostic buffer entry is made and the CPU continues to run.
	The interrupt is started in a startup OB and the delay time elapses before the CPU changes to RUN,	The call of the time-delay interrupt OB is delayed until the CPU is in RUN mode.
The delay time has elapsed,	A previously started time-delay interrupt OB is still being executed,	The operating system generates a time error (calls OB 80). If OB 80 does not exist, the CPU continues to run).

Response to warm restart and cold restart

During a warm restart or a cold restart, all the time-delay interrupt settings made in the user program by means of instructions are cleared.

Starting in a startup OB

A time-delay interrupt can be started in a startup OB. Two conditions must be satisfied to call the time-delay OB:

- The delay time must have elapsed.
- The CPU must be in the RUN mode.

If the delay time has elapsed and the CPU is not yet in the RUN mode, the time-delay interrupt OB call is delayed until the CPU is in RUN mode. The time-delay interrupt OB is then called before the first instruction in OB Main [OB 1] is executed.

SRT_DINT: Start time-delay interrupt (S7-1200, S7-1500)

Description

The instruction "SRT_DINT" is used to start a time-delay interrupt, which calls a time-delay interrupt OB after the expiry of the delay time specified at the parameter DTIME. The time delay interrupt OB is executed once for the time delay interrupt.

If the time-delay interrupt has not been executed yet and the instruction "SRT_DINT" is called again, the existing time-delay interrupt is deleted and a new time-delay interrupt is started.

Note

Delay time

If the delay time used is less than or equal to the currently used CPU cycle time and SRT_DINT is called cyclically, the time delay interrupt OB is executed with each CPU cycle.

Make sure that you select a delay time that is greater than the CPU cycle time.

With the SIGN parameter, you can enter an identifier that identifies the start of the time-delay interrupt. The values of the DTIME and SIGN parameters are shown in the start information of the organization blocked called.

Accuracy

The maximum time between the call of the "SRT_DINT" instruction and the start of the time-delay interrupt OB is one millisecond longer than the configured delay time, provided that no interruption events delay the call.

Parameters

The following table shows the parameters of the "SRT_DINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	I, Q, M, D, L or constant	Number of the OB to be executed after a delay time
DTIME	Input	TIME	I, Q, M, D, L or constant	Delay time (1 to 60000 ms) You can realize longer times, for example, by using a counter in a time-delay interrupt OB.
SIGN	Input	WORD	I, Q, M, D, L or constant	Identifier that appears when the time-delay interrupt OB is called in the start event information of the OB.
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
8090	Incorrect parameter OB_NR
8091	Incorrect parameter DTIME
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for time-delay interrupt functions (Page 2473).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

CAN_DINT: Cancel time-delay interrupt (S7-1200, S7-1500)

Description

You use this instruction to cancel a started time-delay interrupt and, thus, also cancel the call of the time-delay interrupt OB that is executed after the configured delay time. You specify the number of the organization block whose call is to be canceled in the OB_NR parameter.

Parameters

The following table shows the parameters of the "CAN_DINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	I, Q, M, D, L or constant	Number of the OB whose call will be canceled
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
8090	Incorrect parameter OB_NR
80A0	Time-delay interrupt has not started.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for time-delay interrupt functions (Page 2473).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

QRY_DINT: Query time-delay interrupt status (S7-1200, S7-1500)

Description

The instruction "QRY_DINT" is used to query the status of the time-delay interrupt.

Parameters

The following table shows the parameters of the "QRY_DINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	I, Q, M, D, L or constant	Number of the OB whose status is being queried.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the actual parameter of RET_VAL contains an error code. The value "0" is displayed in the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-delay interrupt, see following table.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

Bit	Value	Meaning
0	0	Irrelevant
1	0	Time-delay interrupt is enabled by the operating system.
	1	Time-delay interrupt is disabled.
2	0	Time-delay interrupt is not activated or has elapsed.
	1	Time-delay interrupt is activated.
3	-	-
4	0	Time-delay interrupt OB with the specified number does not exist.
	1	Time-delay interrupt OB with the specified number exists.
Other bits		Always "0"

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Incorrect information in the OB_NR parameter
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for time-delay interrupt functions (Page 2473).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Program example for time-delay interrupt functions

Introduction

In the following example, you read the status of a time-delay interrupt OB (OB20), and set a new delay time and start or disable OB20.

Requirement

Create 10 tags and a structure in a global data block for storing the data.

SLI_gDB_SRT_DINT

	Name	Data type	Start value
1	Static		
2	obNbr	OB_DELAY	20
3	executeQry	Bool	false
4	itemiseStatus	Struct	
5	Time-delay_interrupt_enabled	Bool	false
6	Time-delay_interrupt_activated	Bool	false
7	OB_with_obNbr_exists	Bool	false
8	returnValueQry	Int	0
9	executeSrt	Bool	false
10	delayTime	Time	T#6000ms
11	sign	Word	16#0001
12	returnValueSrt	Int	0
13	executeCan	Bool	false
14	returnValueCan	Int	0
15	testCurrCount	Int	0

Interconnect parameters: "SLI_itemiseStatus_DINT" FC

To evaluate the status information, create an FC "SLI_itemiseStatus_DINT".

Create the following local tag in it.

SLI_itemiseStatus_DINT

	Name	Data type
1	Input	
2	inWord	Word

Create interconnections for all relevant bits.

Network 1:

Network 2:

Network 3:

Interconnect parameters: "SLI_FB_SRT_DINT" FB

To call the instructions, create an FB "SLI_FB_SRT_DINT". Create the following local tags in the function block "SLI_FB_SRT_DINT".

7	Static		
8	statWord	Word	16#0
9	statMem	Bool	false

Network 1: Interconnect the parameters of the "QRY_DINT" instruction as follows.

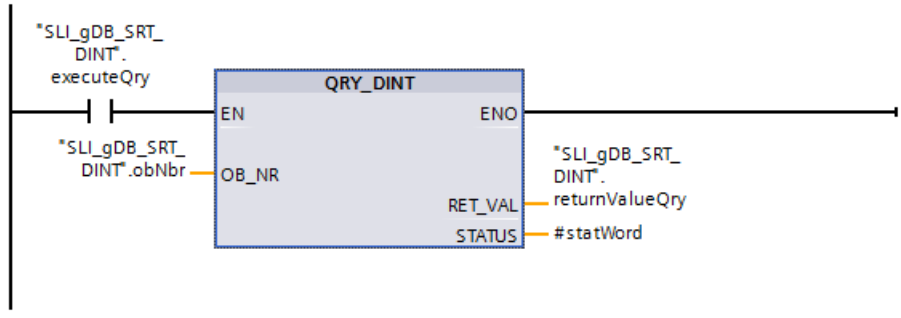

Network 2: Interconnect the parameters of the "SRT_DINT" instruction as follows.

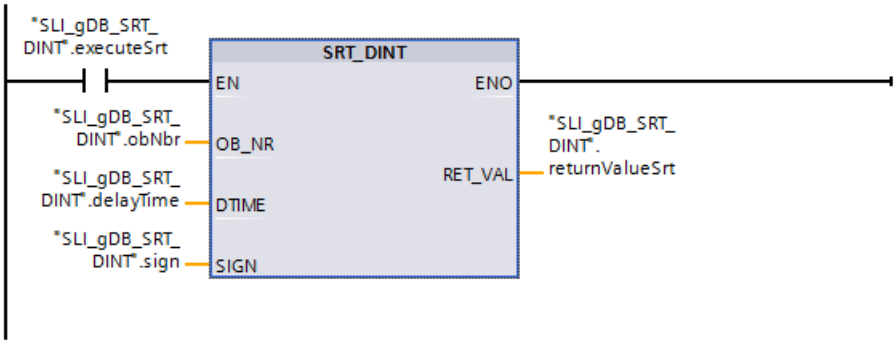

Network 3: Interconnect the parameters of the "CAN_DINT" instruction as follows.

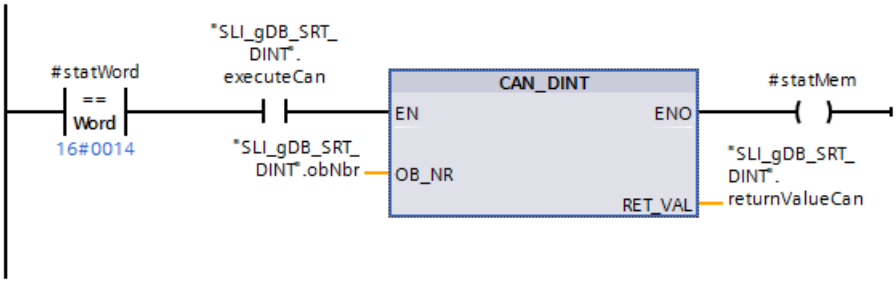

Network 4: Interconnect the parameters that are to be reset with the execution of "CAN_DINT" as follows.

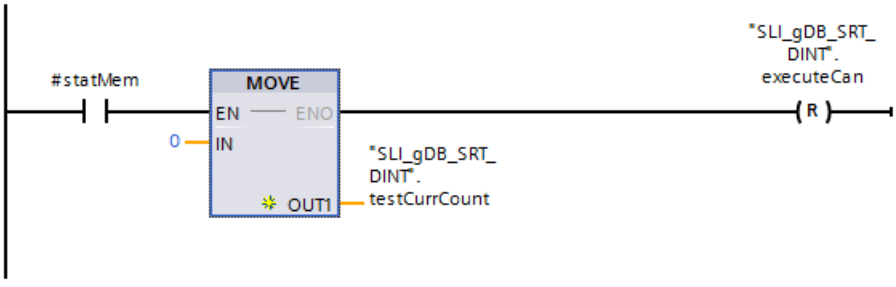

Network 5: Interconnect the parameter of the "SLI_itemiseStatus_DINT" FC as follows.

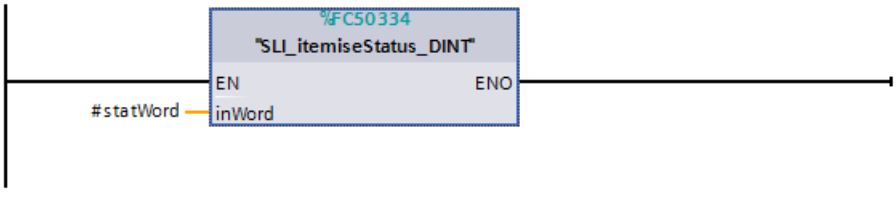

You call the FB "SLI_FB_SRT_DINT" in a cyclic OB (OB1).

Interconnect parameters: OB20

Create the following interconnections in OB20.

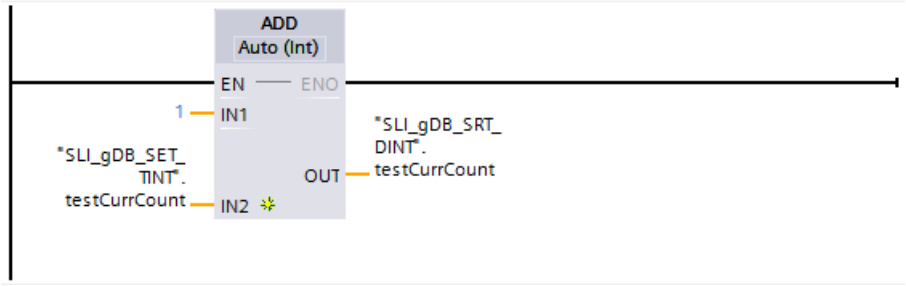

Result of "QRY_DINT"

The instructions receive the number of the OB to be called (OB20) with the input parameter OB_NR ("obNbr").

Network 1:

When the normally open contact ("executeQry") supplies the signal state "TRUE", the "QRY_DINT" instruction is executed and the data of OB20 is read. The output parameter STATUS ("#statWord") displays the status of OB20. The output parameter RET_VAL ("returnValueQry") indicates that processing is taking place without errors.

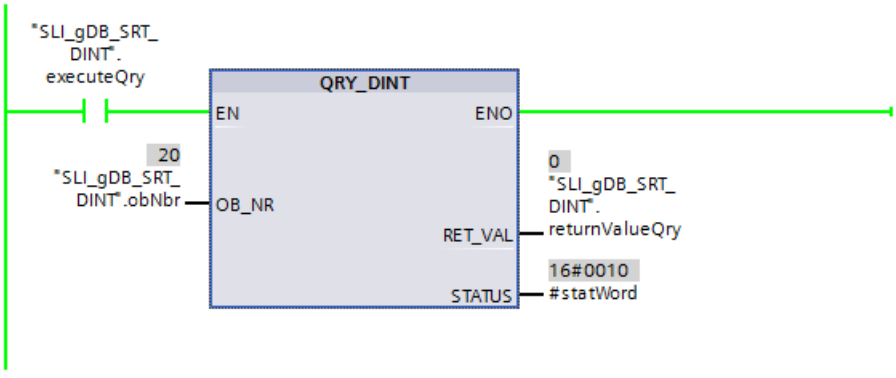

The value of the tag "#statWord" is mapped in the structure "itemiseStatus".

4	itemiseStatus	Struct		
5	Time-delay_interrupt_enabled	Bool	false	TRUE
6	Time-delay_interrupt_activated	Bool	false	FALSE
7	OB_with_obNbr_exists	Bool	false	TRUE

Result of "SRT_DINT"

Network 2: When the normally open contact ("executeSrt") supplies the signal state "TRUE", the "SRT_DINT" instruction is executed. The input parameter DTIME ("delayTime") transfers the delay time (6 s) to OB20 and OB20 is enabled.

Note

Its delay time starts running at the time OB20 is enabled. OB20 is then called unless it has received the delay time again beforehand in a new CPU cycle.

The selected identifier (value "16#0001") to be output with the execution of OB20 is stored at the input parameter SIGN ("sign"). The identifier is shown in the start event information of OB20. The output parameter RET_VAL ("returnValueSrt") indicates that processing took place without errors.

If the NO contact ("executeSrt") returns the signal state "FALSE" after successful processing of SRT_DINT, the delay time is not transferred again to the OB20. As soon as the delay time has expired, OB20 is executed once.

Note

The described reaction of the OB20 in combination with the EN parameter only applies when the currently used CPU cycle time is less than the delay time of the OB20.

Result of the OB20

With each call of OB20, the instruction "ADD" is executed and the tag "testCurrCount" is incremented by "1". OB20 is called only once after expiration of the delay time defined with SRT_DINT. After it has been called, OB20 is once again "disabled".

SLI_gDB_SRT_DINT

	Name	Data type	Start value	Monitor value
1	Static			
2	obNbr	OB_DELAY	20	20
3	executeQry	Bool	false	TRUE
4	itemiseStatus	Struct		
5	Time-delay_interrupt_enabled	Bool	false	TRUE
6	Time-delay_interrupt_activated	Bool	false	FALSE
7	OB_with_obNbr_exists	Bool	false	TRUE
8	returnValueQry	Int	0	0
9	executeSrt	Bool	false	FALSE
10	delayTime	Time	T#6000ms	T#6S
11	sign	Word	16#0001	16#0001
12	returnValueSrt	Int	0	0
13	executeCan	Bool	false	FALSE
14	returnValueCan	Int	0	0
15	testCurrCount	Int	0	1

Result of "CAN_DINT"

Network 3: As long as OB20 is set and enabled ("#statWord" has the value "16#0014"), the comparator returns the signal state "TRUE" and CAN_DINT can be executed.

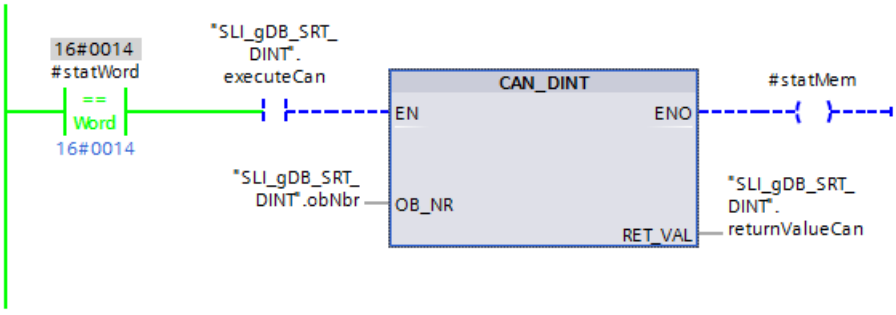

When the normally open contact ("executeCan") supplies the signal state "TRUE", the "CAN_DINT" instruction is executed. The instruction disables OB20 and deletes the set call data of OB20.
The output parameter RET_VAL ("returnValueCan") indicates that processing took place without errors.

The execution of CAN_DINT sets the coil ("#statMem") to "TRUE" and network 4 is executed.

Result of the reset

Network 4: When the normally open contact ("#statMem") supplies the signal state "TRUE", the tag "testCurrCount" is reset to the value "0" with the instruction "MOVE". Reset ("executeCan") in the signal state "TRUE" also automatically ends the execution of the instruction "CAN_DINT".

Result of the status information "#statWord"

The FC "SLI_itemiseStatus_DINT" is called in network 7. The significant bits of the tag "#statWord" are evaluated and output as Boolean tags with plain-text name. When a tag of the structure "itemiseStatus" supplies the signal state "TRUE", its statement is correct.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Synchronous errors (S7-1500)

Mask synchronous error events (S7-1500)

Introduction

Synchronous errors are programming and access errors. Such errors occur as a result of programming with incorrect operand areas or numbers, or incorrect addresses. **Masking** these synchronous errors means the following:

- Masked synchronous errors do not trigger an error OB call and do not lead to a programmed alternative reaction.
- The CPU "records" the masked errors that have occurred in an error status register.

Masking is carried out by calling the "MSK_FLT (Page 2485)" instruction.

Unmasking errors means canceling a previously set mask and clearing the corresponding bit in the event status register of the current priority class. Masking is canceled as follows:

- By calling the "DMSK_FLT (Page 2486)" instruction.
- Once the current priority class has been completed.

If an error event occurs after it has been unmasked, the operating system will start the associated error OB.

You can use the "READ_ERR (Page 2487)" instruction to read the masked errors that have occurred.

Note

Regardless of masking or unmasking of the error events for the S7-1500, the error event is entered in the diagnostic buffer and the group error LED of the CPU lights up.

Handling errors in general

Different actions are possible if programming and I/O area access errors occur in a user program:

- You can program an error OB that is called by the operating system when the corresponding error occurs.
- You can disable the error OB call individually for each priority class. If an error of this type occurs in the priority class, the CPU does not change to STOP in this case. The CPU enters the error in an error register. From this entry, however, you do not recognize when or how often this error has occurred.

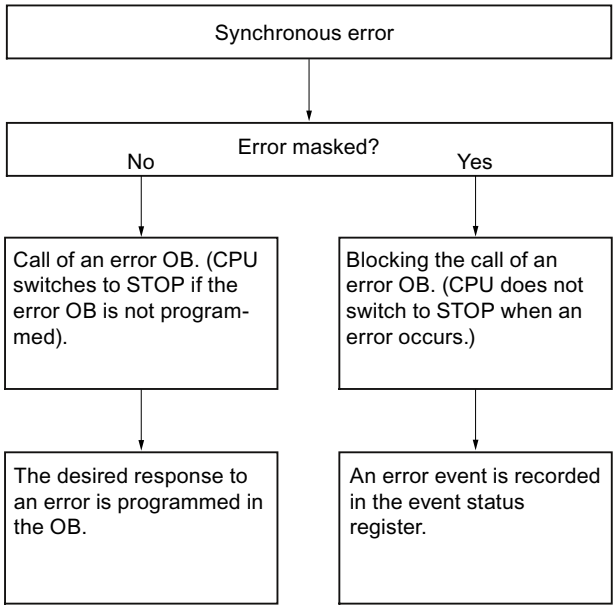

Synchronous errors are assigned to a particular bit pattern known as the **error filter**. You can find this error filter in the input and output parameters of the "MSK_FLT (Page 2485)", "DMSK_FLT (Page 2486)", and "READ_ERR (Page 2487)" instructions.

Synchronous errors are divided into **programming and access errors** which you can mask in two error masks. The error filters are illustrated in the following figures.

Programming error filter

The following figure shows the bit pattern of the error filter for programming errors. The error filter for the programming errors is available in the "PRGFLT_..." parameters (see below "Programming errors, Low-Word" or "Programming errors, High-Word").

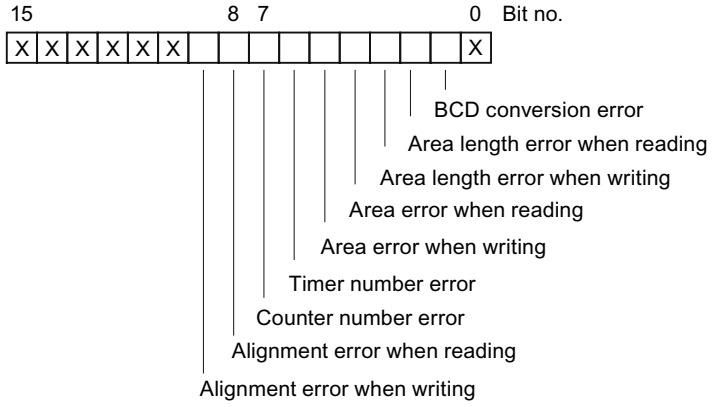

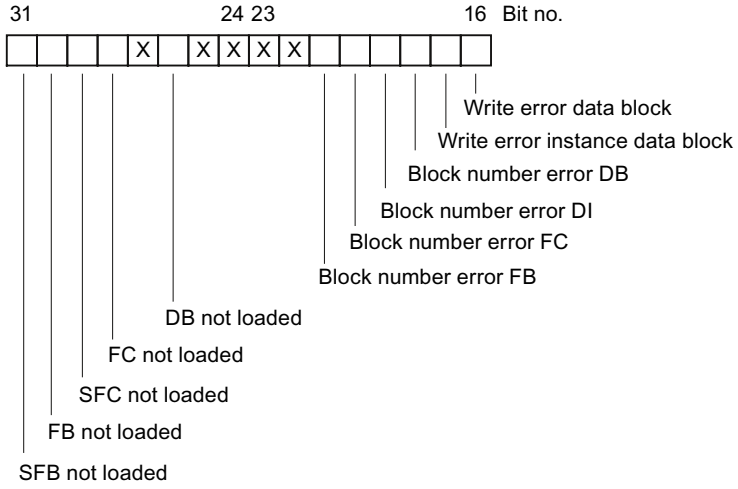

Legend: X Not relevant

The non-relevant bits "x" are set to "0" for the input and output parameters of the instructions "MSK_FLT (Page 2485)", "DMSK_FLT (Page 2486)", "READ_ERR (Page 2487)".

Access error mask

The following figure shows the bit pattern of the error mask for access errors. The error mask for access errors is located in the parameters ACCFLT_...

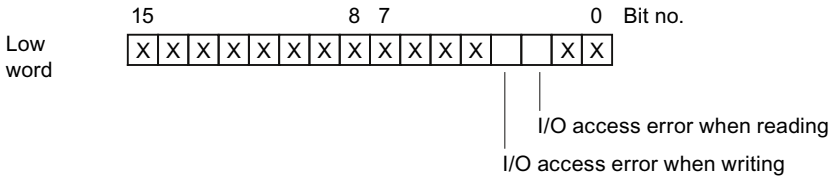

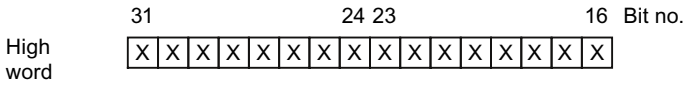

Legend: X Not relevant

Example: The following figure shows how the low word of the error mask for access errors appears with all masked errors.

- As input parameter for "MSK_FLT (Page 2485)"
- As output parameter for "MSK_FLT (Page 2485)"

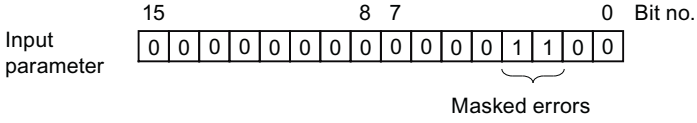

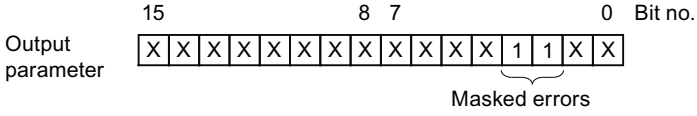

Legend:
- X Not relevant
- 0 Not masked
- 1 Masked

Programming error low word

The following table lists the errors assigned to the low word of the error filter for programming errors. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Possible cause of error
BCD conversion error	2521	The value to be converted is not a BCD number (e.g. 5E8).
Area length error when reading	2522	An addressed operand is not located entirely within the possible operand range. Example: MW 320 should be read although the memory area is only 256 bytes long.
Area length error when writing	2523	An addressed operand is not located entirely within the possible operand range. Example: A value should be written to MW 320 although the memory area is only 256 bytes long.
Area error when reading	2524	An incorrect range identifier for the operand was specified during indirect, overlapping range addressing. Example: • Correct: LAR1 P#E 12.0 L W[AR1, P#0.0] • Incorrect: LAR1 P#12.0 L W[AR1, P#0.0] The area error is reported during this operation.
Area error when writing	2525	An incorrect range identifier for the operand was specified during indirect, overlapping range addressing. Example: • Correct: LAR1 P#E 12.0 T W[AR1, P#0.0] • Incorrect: LAR1 P#12.0 T W[AR1, P#0.0] The area error is reported during this operation.
Timer number error (not implemented with the S7-1500)	2526	A non-existent timer is accessed. Example: • SI T [MW 0] with MW 0 = 129; the timer 129 should be started although there are only 128 timers available.
Counter number error (not implemented with the S7-1500)	2527	A non-existent counter is accessed. • Example: ZV Z [MW 0] with MW 0 = 600; the counter 600 is accessed although only 512 counters are available.

Error	Event ID (W#16#...)	Possible cause of error
Alignment error when reading	2528	A byte, word or double word operand with a bit address ≠ 0 is addressed. Example: • Correct: LAR1 P#M12.0 L B[AR1, P#0.0] • Incorrect: LAR1 P#M12.4 L B[AR1, P#0.0]
Alignment error when writing	2529	A byte, word or double word operand with a bit address ≠ 0 is addressed. Example: • Correct: LAR1 P#M12.0 T B[AR1, P#0.0] • Incorrect: LAR1 P#M12.4 T B[AR1, P#0.0]

Programming error high word

The following table lists the errors assigned to the high word of the error filter for programming errors. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Possible cause of error
Write error data block	2530	The data block to be written to is write-protected.
Write error instance data block	2531	The instance data block to be written to is write-protected.
Block number error DB	2532	The number of the data block is greater than the highest permissible number.
Block number error DI	2533	The number of the instance data block is greater than the highest permissible number.
Block number error FC	2534	The number of a called function (FC) is greater than the highest permissible number.
Block number error FB	2535	The number of a called function block (FB) is greater than the highest permissible number.
DB not loaded	253A	The data block is not loaded.
Instruction not loaded	253C to 253F	The instruction to be called is not loaded.

Access error

The following table contains the errors that are assigned to the error mask for access errors. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Possible cause of error
I/O access error when reading	2942	• The address in the I/O area has no assigned signal module. • Access to this I/O area was not acknowledged within the specified module monitoring time (timeout).
I/O access error when writing	2943	

MSK_FLT: Mask synchronous error events (S7-1500)

Description

You use the instruction to control the reaction of the CPU to synchronous errors. This is done by masking the respective synchronous errors (for error filters, see Mask synchronous error events (Page 2479)). When you call "MSK_FLT", you mask the synchronous errors in the current priority class.

If you set individual bits of the synchronous error filter to "1" in the input parameters, other bits that were set previously retain their value of "1". You obtain new error filters that you read out using the output parameters. The synchronous errors you have masked do not call an OB but are simply entered in an error register. You read the event status register with the instruction "READ_ERR (Page 2487)".

Parameters

The following table shows the parameters of the instruction "MSK_FLT":

Parameter	Declaration	Data type	Memory area	Description
PRGFLT_SET_MASK	Input	DWORD	I, Q, M, D, L or constant	Programming error to be masked
ACCFLT_SET_MASK	Input	DWORD	I, Q, M, D, L or constant	Access error to be masked
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_MASKED	Output	DWORD	I, Q, M, D, L	Masked programming errors
ACCFLT_MASKED	Output	DWORD	I, Q, M, D, L	Masked access errors

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	None of the errors were already masked.
0001	At least one of the errors was already masked. Nevertheless, the other errors will be masked.
-	General error information See also: Getting error ID locally with GetErrorID (Page 1857)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for synchronous error events (Page 2488).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

DMSK_FLT: Unmask synchronous error events (S7-1500)

Description

You use the instruction to unmask the errors masked with "MSK_FLT (Page 2485)". To do this, you must set the corresponding bits of the error filter to "1" in the input parameters. With the "DMSK_FLT" call, you unmask the corresponding synchronous errors of the current priority class. At the same time, the queried entries are cleared in the error register. You read out the new error filters using the output parameters.

Parameters

The following table shows the parameters of the instruction "DMSK_FLT":

Parameter	Declaration	Data type	Memory area	Description
PRGFLT_RESET_MASK	Input	DWORD	I, Q, M, D, L or constant	Programming errors to be unmasked
ACCFLT_RESET_MASK	Input	DWORD	I, Q, M, D, L or constant	Access errors to be unmasked
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_MASKED	Output	DWORD	I, Q, M, D, L	Still masked programming errors
ACCFLT_MASKED	Output	DWORD	I, Q, M, D, L	Still masked access errors

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	All specified errors were unmasked.
0001	At least one of the errors was not masked. Nevertheless, the other errors will be unmasked.
-	General error information See also: GET_ERR_ID: Get error ID locally (Page 1857)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for synchronous error events (Page 2488).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

READ_ERR: Read out event status register (S7-1500)

Description

You use the instruction to read the error register. The structure of the error register corresponds to that of the programming and access error masks, which you can program as input parameters with "MSK_FLT (Page 2485)" and "DMSK_FLT (Page 2486)".

In the input parameters, you enter the synchronous errors you want to read from the error register. When you call "READ_ERR", you read the required entries from the error register and at the same time clear these entries.

The error register contains information that tells you which of the masked synchronous errors in the current priority class occurred at least once. If a bit is set, this means that the corresponding masked synchronous error occurred at least once.

Parameters

The following table shows the parameters of the instruction "READ_ERR":

Parameter	Declaration	Data type	Memory area	Description
PRGFLT_QUERY	Input	DWORD	I, Q, M, D, L or constant	Query programming error
ACCFLT_QUERY	Input	DWORD	I, Q, M, D, L or constant	Query access error
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_CLR	Output	DWORD	I, Q, M, D, L	Programming errors that occurred
ACCFLT_CLR	Output	DWORD	I, Q, M, D, L	Access errors that occurred

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	All queried errors are masked.
0001	At least one of the queried errors is not masked.
-	General error information See also: Getting error ID locally with GetErrorID (Page 1857)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for synchronous error events (Page 2488).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Program example for synchronous error events (S7-1500)

Introduction

In the example below, mask the synchronous error events (program errors) with the instruction "MSK_FLT", unmask a part of the synchronous error events with the instruction "DMSK_FLT" and read out the event status register with the instruction "READ_ERR". When the program error is unmasked, call a program error OB.

Storage of data

Create the following structures and tags for storing the data in a global data block.

- Tags for MSK_FLT:

SLI_gDB_MSK_FLT

	Name	Data type	Start value
1	Static		
2	maskedProgErrs	DWord	16#0
3	maskedAccessErrs	DWord	16#0
4	MSK_FLT	Struct	
5	execute	Bool	false
6	progErrMask_Details	Struct	
7	accessErrMask_Details	Struct	
8	progErrSetMask	DWord	16#0
9	accessErrSetMask	DWord	16#0
10	returnValue	Int	0

- MSK_FLT / content of the structure "progErrMask_Details":

6	progErrMask_Details	Struct	
7	BCD_conversion_error	Bool	false
8	Area_length_error_when_reading	Bool	1
9	Area_length_error_when_writing	Bool	1
10	Range_error_when_reading	Bool	1
11	Range_error_when_writing	Bool	1

You can find the complete organization of the structure here: Mask synchronous error events (Page 2479).

- MSK_FLT / content of the structure "accessErrMask_Details":

7	accessErrMask_Details	Struct	
8	IO_access_error_when_reading	Bool	false
9	IO_access_error_when_writing	Bool	false

- Tags for DMSK_FLT:

11	DMSK_FLT	Struct	
12	execute	Bool	false
13	progErrResetMask	DWord	16#0000_000C
14	accessErrResetMask	DWord	16#0
15	returnValue	Int	0

- Tags for READ_ERR:

16	READ_ERR	Struct	
17	busyReading	Bool	false
18	peOB_unmasked_active	Bool	false
19	returnValue	Int	0
20	occuredMskProgErrs	DWord	16#0
21	occuredMskAccessErrs	DWord	16#0
22	occProgErrMask_Details	Struct	
23	occAccErrMask_Details	Struct	

The organization of the structures "occProgErrMask_Details" and "occAccErrMask_Details" is similar to the organization of the structures "progErrMask_Details" and "accessErrMask_Details".

- Tags for the test result:

24	testErrorEvent	Struct	
25	index	Int	0
26	field	Array[0..10] of Real	
27	result	Real	0.0
28	startOut	Bool	false

Interconnect parameters: "SLI_FB_genErr_SyncErrEv" FB

Create an "SLI_FB_genErr_SyncErrEv" FB to generate a program error.

Create the following local tag in it.

SLI_FB_genErr_SyncErrEv			
	Name	Data type	Default value
1	Input		
2	index[0..10]	Int	0

Create the following interconnections in the FB.

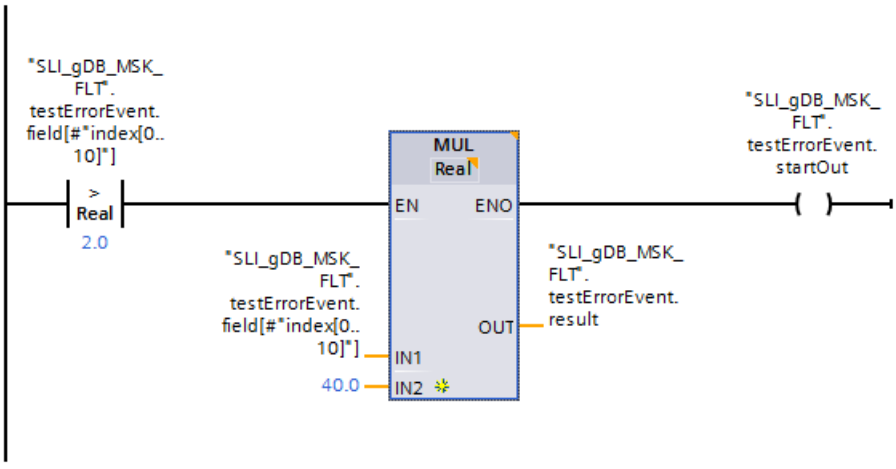

Interconnect parameters: "SLI_itemiseDW_SyncErrEv" FC

To transfer a bit sequence in the size of a DWORD, create the FC "SLI_itemiseDW_SyncErrEv".

Create the following local tags (inputs and outputs from 1 to 31) in it.

- Excerpt of the inputs:

SLI_itemiseDW_SyncErrEv

	Name	Data type
1	Input	
2	in1	Bool
3	in2	Bool
4	in3	Bool
5	in4	Bool
6	in5	Bool
7	in6	Bool
8	in7	Bool
9	in8	Bool

- Excerpt of the outputs:

21	in31	Bool
22	Output	
23	out1	Bool
24	out2	Bool
25	out3	Bool
26	out4	Bool
27	out5	Bool
28	out6	Bool
29	out7	Bool
30	out8	Bool
39	out31	Bool

Create the following interconnections for the inputs and outputs in the FC based on the example of network 1.

Network 1:

Interconnect parameters: "SLI_mask_SyncErrEv" FC

To convert the masks to be set for program and access errors, create the FC "SLI_mask_SyncErrEv".

Create the following local tags in it.

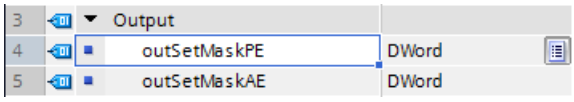

Network 1:

Interconnect the parameters of the "SLI_itemiseDW_SyncErrEv" FC according to the following example.

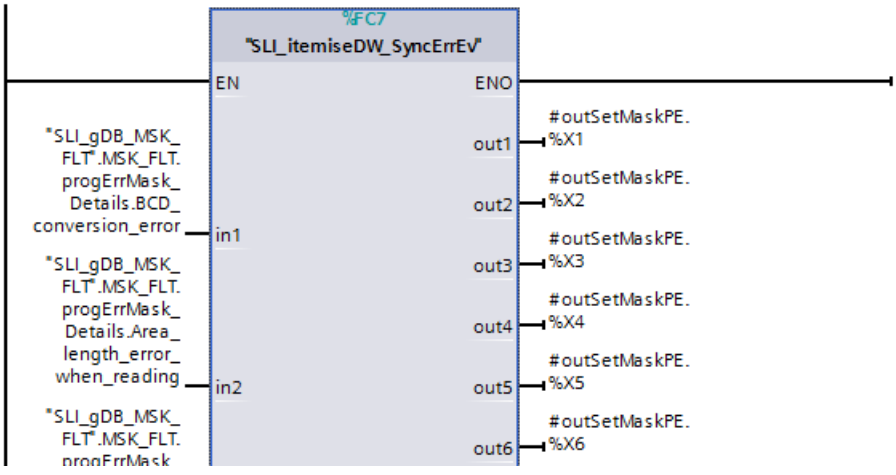

You can find the complete organization of the structure of the program errors ("progErrMask_Details") here: Mask synchronous error events (Page 2479).

Network 2:

Interconnect bit 2 of the DWORD of the access errors as follows.

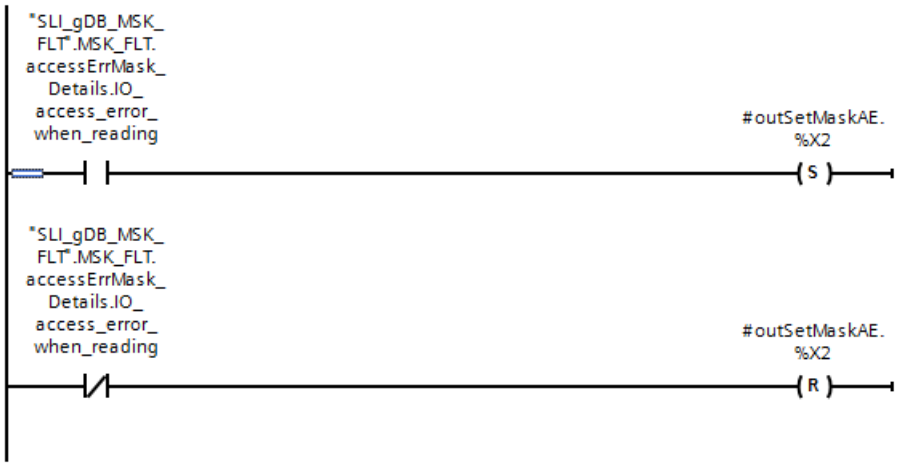

Network 3:

Interconnect bit 3 of the DWORD of the access errors as follows.

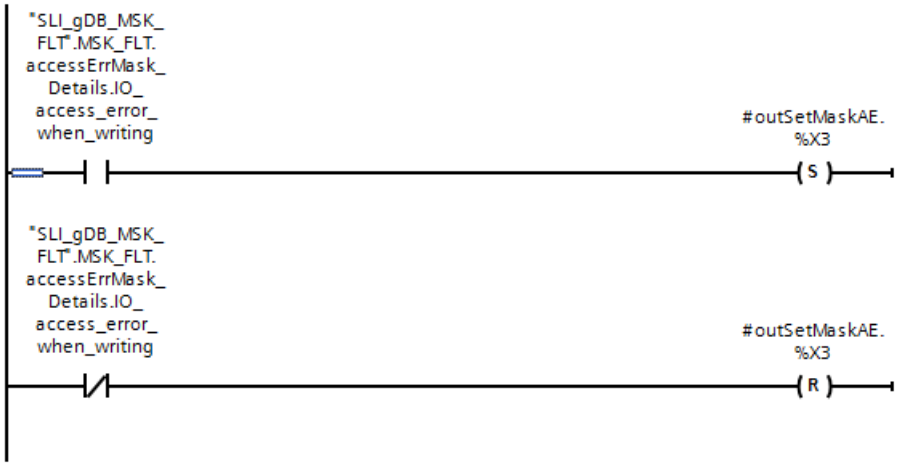

Interconnect parameters: "SLI_occured_SyncErrEv" FC

To convert read program and access errors, create the FC "SLI_occured_SyncErrEv".

Create the following local tags in it.

SLI_occured_SyncErrEv		
	Name	Data type
1	Input	
2	inReadMaskPE	DWord
3	inReadMaskAE	DWord

Network 1:

Interconnect the parameters of the "SLI_itemiseDW_SyncErrEv" FC according to the following example.

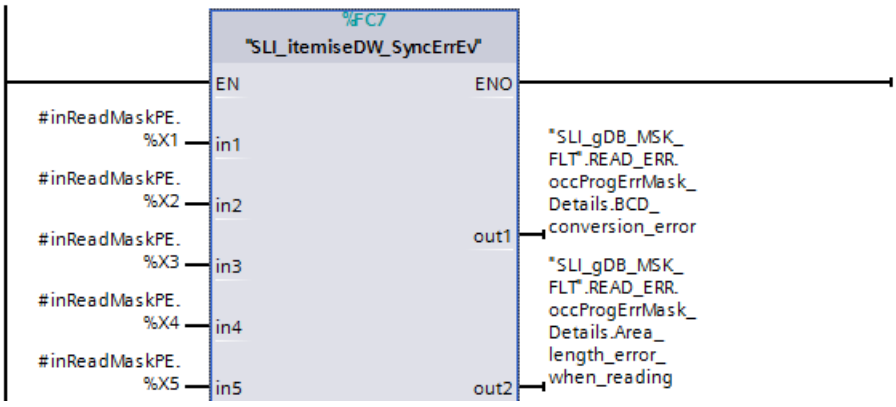

You can find the complete organization of the structure of the program errors ("occProgErrMask_Details") here: Mask synchronous error events (Page 2479).

Network 2:

Interconnect bit 2 of the DWORD of the access errors as follows.

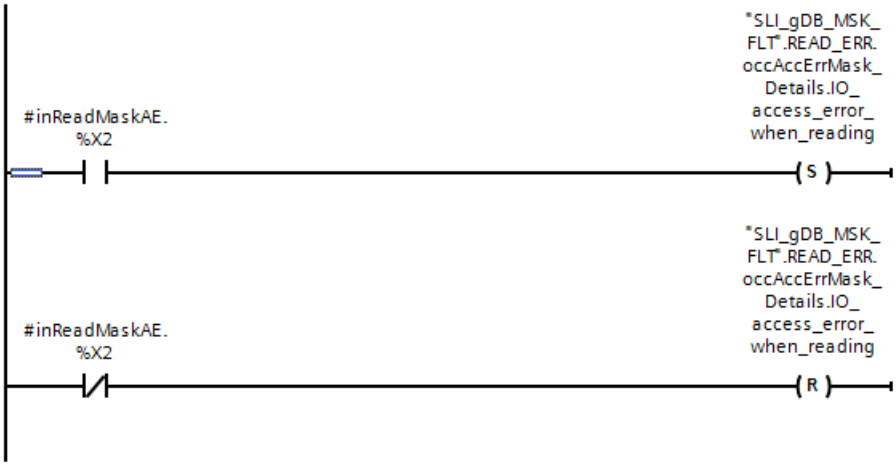

Network 3:

Interconnect bit 3 of the DWORD of the access errors as follows.

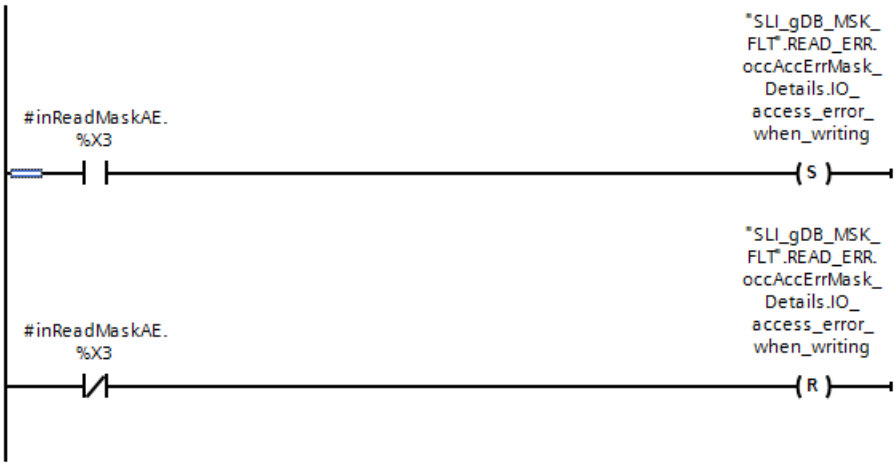

Interconnect parameters: "SLI_peOB_SyncErrEV" OB

Create the OB "SLI_peOB_SyncErrEV" in case of a program error in the unmasked state.

Network 1:

Create the following interconnection in the OB.

Interconnect parameters: "SLI_FC_MSK_FLT" FC

To call the program parts, create the FC "SLI_FC_MSK_FLT".

Network 1:

Interconnect the parameters of the FC "SLI_mask_SyncErrEv" as well as the tags "peOB_unmased_active", "MSK_FLT.execute" and "maskedProgErrs" as follows.

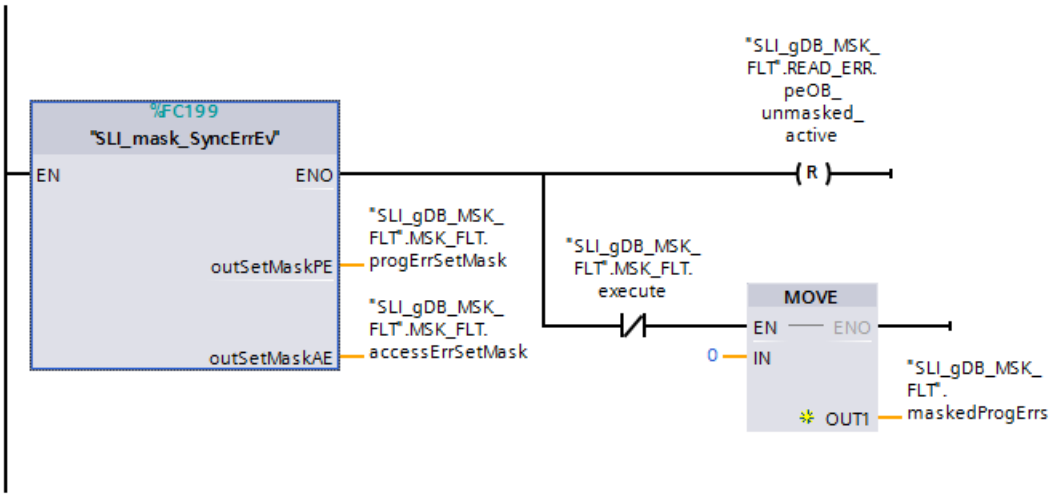

Network 2:

Interconnect the parameters of the "MSK_FLT" instruction as follows.

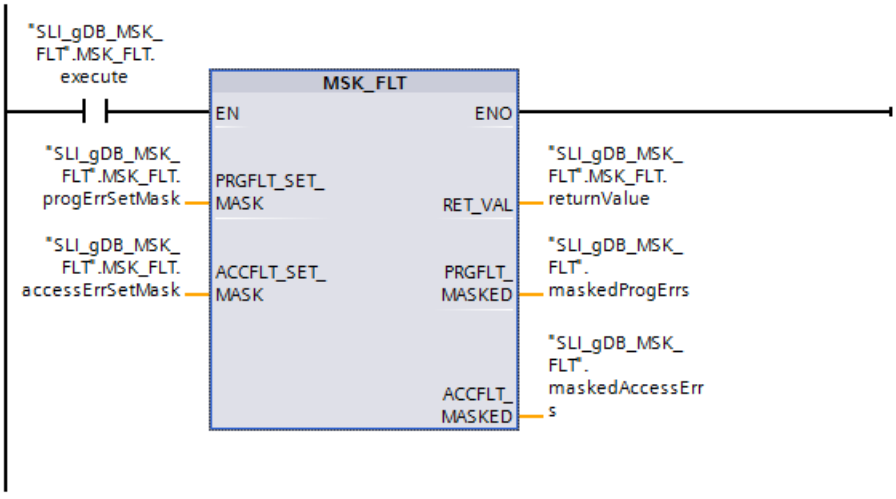

Network 3:

Interconnect the parameters of the "DMSK_FLT" instruction as follows.

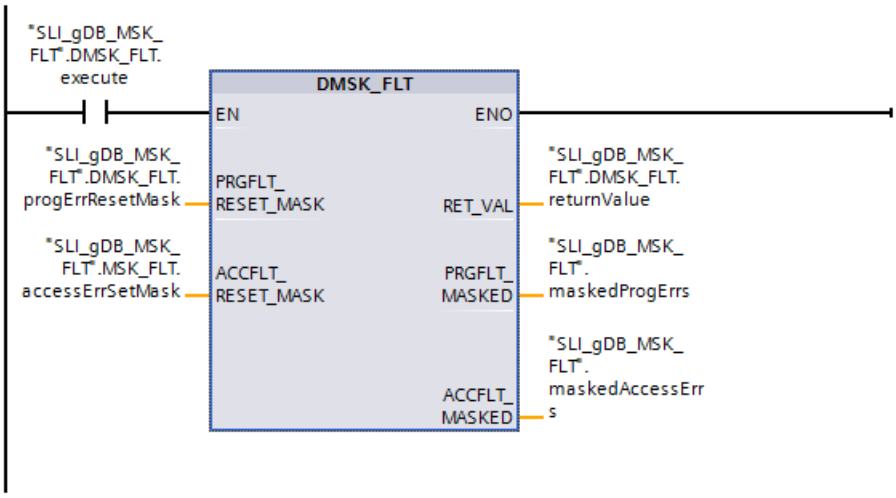

Network 4:

Interconnect the parameters of the "SLI_FB_genErr_SyncErrEv" FC as follows.

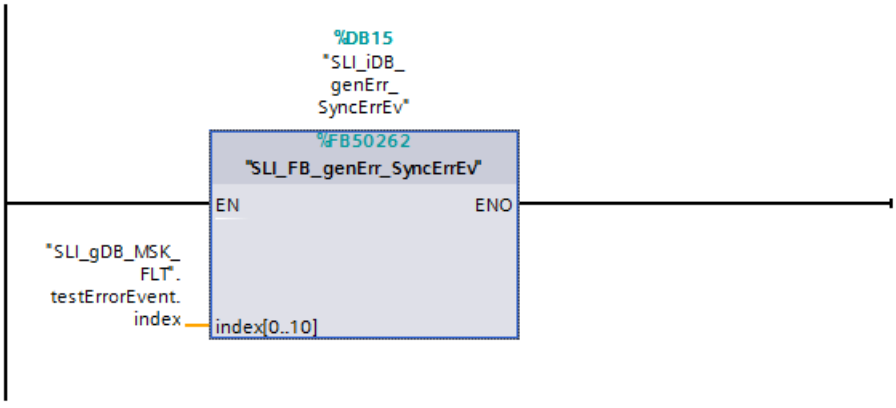

Network 5:

Interconnect the parameters of the "READ_ERR" instruction as follows.

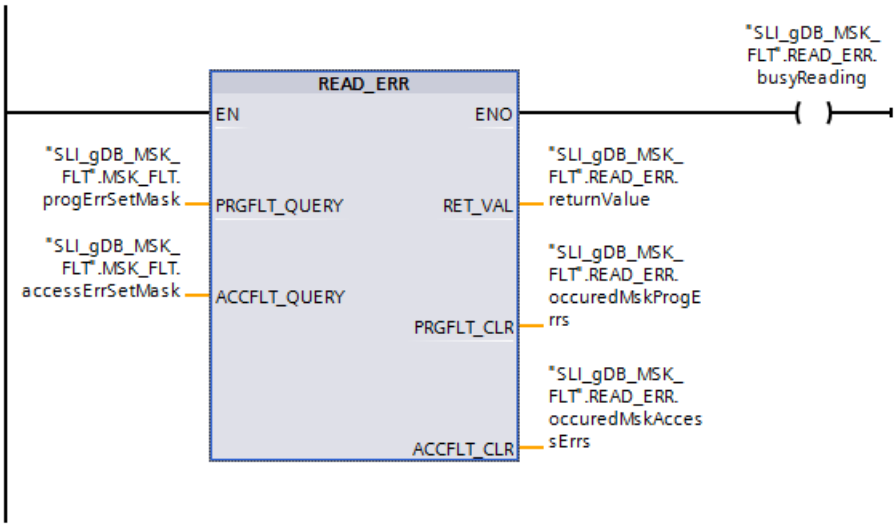

Network 6:

Interconnect the parameters of the "SLI_occured_SyncErrEv" FC as follows.

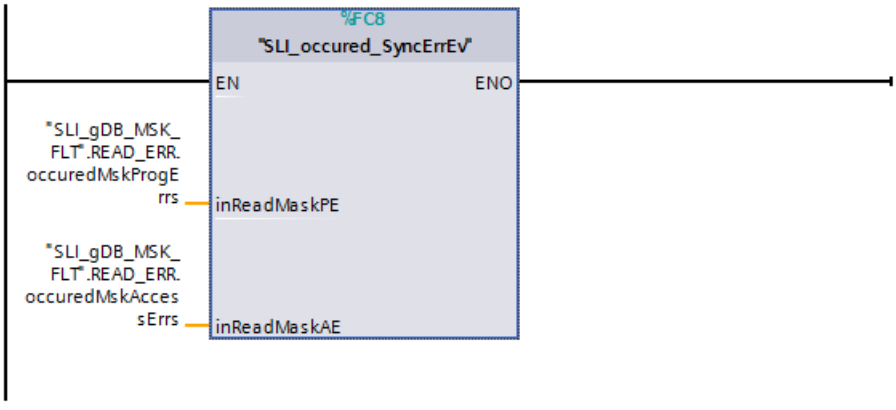

Result of OB "SLI_peOB_SyncErrEv"

When a program error was triggered and is not masked by a filter, the program error OB "SLI_peOB_SyncErrEv" is executed.

The tag "READ_ERR.peOB_unmasked_active" is then set to "TRUE" in network 1 of the OB.

Result of FC "SLI_FC_MSK_FLT"

Network 1 (FC "SLI_mask_SyncErrEv"):

By calling the FC "SLI_mask_SyncErrEv", the structures for program errors and access errors are converted into the data type DWORD ("progErrSetMask" and "accessErrSetMask").

Reset the value because the value "TRUE" of the tag "READ_ERR.peOB_unmasked_active" is still displayed even after calling the program error OB. If the instruction "MSK_FLT" is not executed successfully ("MSK_FLT.execute" has the value "FALSE"), the tag "maskedProgErrs" is reset.

Network 2 ("MSK_FLT"):

When the normally open contact ("MSK_FLT.execute") supplies the signal state "TRUE", the "MSK_FLT" instruction is executed.

The synchronous error events that have to be controlled are specified according to the tags "progErrMask_Details" and "accessErrMask_Details":

- You specify the program errors you want to mask with "progErrMask_Details".
- No access errors ("accessErrMask_Details") were masked in the example.

The synchronous error events that have to be controlled are stored in the data type DWORD with the input parameters PRGFLT_SET_MASK ("MSK_FLT.progErrSetMask") and ACCFLT_SET_MASK ("MSK_FLT.accessErrSetMask").

The output parameter RET_VAL ("MSK_FLT.returnValue") indicates that none of the errors was already masked. The currently masked synchronous error events are displayed at the output parameters "PRGFLT_MASKED" ("maskedProgErrs") and ACCFLT_MASKED ("maskedAccessErrs").

2	maskedProgErrs	DWord	16#0	16#0000_003C
3	maskedAccessErrs	DWord	16#0	16#0000_0000
4	MSK_FLT	Struct		
5	execute	Bool	false	TRUE
6	progErrMask_Details	Struct		
7	accessErrMask_Details	Struct		
8	progErrSetMask	DWord	16#0	16#0000_003C
9	accessErrSetMask	DWord	16#0	16#0000_0000
10	returnValue	Int	0	0

Network 3 ("DMSK_FLT"):

When the normally open contact ("DMSK_FLT.execute") supplies the signal state "TRUE", the "DMSK_FLT" instruction is executed. The synchronous error events that have to be unmasked are stored in the data type DWORD with the input parameters PRGFLT_RESET_MASK ("DMSK_FLT.progErrResetMask") and ACCFLT_RESET_MASK ("DMSK_FLT.accessErrResetMask").

Output parameter RET_VAL ("DMSK_FLT.returnValue") indicates that all specified errors were unmasked.

11	DMSK_FLT	Struct		
12	execute	Bool	false	TRUE
13	progErrResetMask	DWord	16#0000_000C	16#0000_000C
14	accessErrResetMask	DWord	16#0	16#0000_0000
15	returnValue	Int	0	0

The currently masked synchronous error events are displayed at the output parameters "PRGFLT_MASKED" ("maskedProgErrs") and ACCFLT_MASKED ("maskedAccessErrs").

2	maskedProgErrs	DWord	16#0	16#0000_0030
3	maskedAccessErrs	DWord	16#0	16#0000_0000

Network 4 (FB "SLI_FB_genErr_SyncErrEv"):

When the tag "testErrorEvent.index" returns a value greater than "10", a program error is triggered.

Network 5 ("READ_ERR"):

The instruction "READ_ERR" reads the event status register. The synchronous error events that are to be read are stored in the data type DWORD with the input parameters PRGFLT_QUERY ("READ_ERR.progErrSetMask") and ACCFLT_QUERY ("READ_ERR.accessErrSetMask").

- Case 1 - The program error was not triggered. No masks are set:
  The output parameter RET_VAL ("READ_ERR.returnValue") indicates that at least one of the queried errors is not masked.
  The currently occurring error events are displayed at the output parameters "PRGFLT_CLR" ("READ_ERR.occuredMskProgErrs") and ACCFLT_CLR ("READ_ERR.occuredMskAccessErrs").

16	READ_ERR	Struct		
17	busyReading	Bool	false	TRUE
18	peOB_unmasked_active	Bool	false	FALSE
19	returnValue	Int	0	1
20	occuredMskProgErrs	DWord	16#0	16#0000_0000
21	occuredMskAccessErrs	DWord	16#0	16#0000_0000
22	occProgErrMask_Details	Struct		
23	occAccErrMask_Details	Struct		

- Case 2 - The program error was triggered and is not masked:
  MSK_FLT, DMSK_FLT are inactive. The program error OB is executed.

16	READ_ERR	Struct		
17	busyReading	Bool	false	TRUE
18	peOB_unmasked_active	Bool	false	TRUE
19	returnValue	Int	0	1
20	occuredMskProgErrs	DWord	16#0	16#0000_0000
21	occuredMskAccessErrs	DWord	16#0	16#0000_0000
22	occProgErrMask_Details	Struct		
23	occAccErrMask_Details	Struct		
24	testErrorEvent	Struct		
25	index	Int	0	11
26	field	Array[0..10] of Real		
27	result	Real	0.0	0.0
28	startOut	Bool	false	FALSE

- Case 3 - The triggered program error is masked:
  MSK_FLT is active. DMSK_FLT and the program error OB are inactive.

16	READ_ERR	Struct		
17	busyReading	Bool	false	TRUE
18	peOB_unmasked_active	Bool	false	FALSE
19	returnValue	Int	0	0
20	occuredMskProgErrs	DWord	16#0	16#0000_0004
21	occuredMskAccessErrs	DWord	16#0	16#0000_0000
22	occProgErrMask_Details	Struct		
23	occAccErrMask_Details	Struct		
24	testErrorEvent	Struct		
25	index	Int	0	11
26	field	Array[0..10] of Real		
27	result	Real	0.0	0.0
28	startOut	Bool	false	FALSE

The following program error occurred according to the structure "occProgErrMask_Details":

22	occProgErrMask_Details	Struct		
23	BCD_conversion_error	Bool	false	FALSE
24	Area_length_error_when_reading	Bool	false	TRUE
25	Area_length_error_when_writing	Bool	false	FALSE
26	Range_error_when_reading	Bool	false	FALSE
27	Range_error_when_writing	Bool	false	FALSE

- Case 4 - The triggered program error is unmasked:
  MSK_FLT, DMSK_FLT and the program error OB are active.

16	READ_ERR	Struct		
17	busyReading	Bool	false	TRUE
18	peOB_unmasked_active	Bool	false	TRUE
19	returnValue	Int	0	1
20	occuredMskProgErrs	DWord	16#0	16#0000_0000
21	occuredMskAccessErrs	DWord	16#0	16#0000_0000
22	occProgErrMask_Details	Struct		
23	occAccErrMask_Details	Struct		
24	testErrorEvent	Struct		
25	index	Int	0	11
26	field	Array[0..10] of Real		
27	result	Real	0.0	0.0
28	startOut	Bool	false	FALSE

Network 6 (FC "SLI_occured_SyncErrEv"):

By calling the FC "SLI_occured_SyncErrEv", the read program errors ("occuredMskProgErrs") and access errors ("occuredMskAccessErrs") are converted into the structures "occProgErrMask_Details" and "occAccErrMask_Details".

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

MSK_FLT: Mask synchronous error events (Page 2485)

DMSK_FLT: Unmask synchronous error events (Page 2486)

READ_ERR: Read out event status register (Page 2487)

Asynchronous error event

DIS_IRT: Disable interrupt event (S7-1500)

Description

The instruction "DIS_IRT" is used to disable the processing of new interrupts and asynchronous error events. Disabling means that if an interrupting event occurs, the operating system of the CPU reacts as follows:

- It **neither** calls an interrupt OB nor an asynchronous error OB,
- **nor** does it trigger the normal reaction if an interrupt OB or asynchronous error OB is not programmed.

If you disable interrupts and asynchronous error events, this remains in effect for all priority classes. The disable is only canceled with the instruction "EN_IRT (Page 2503)" or by a warm or a cold restart.

Note

Remember that when you program the "DIS_IRT" instruction, all interrupts that occur will be discarded.

Parameters

The following table shows the parameters of the "DIS_IRT" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Specifies which interrupts and asynchronous errors are disabled.
OB_NR	Input	INT	I, Q, M, D, L or constant	OB number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter MODE

MODE (B#16#...)	Meaning
00	All newly occurring interrupts and asynchronous error events are disabled (with the exception of the programming error and the error during direct access to I/O data and the Motion Control OBs). Assign the value "0" to the OB_NR parameter. Entries continue to be made in the diagnostics buffer.
01	All newly occurring events belonging to a specified interrupt class are disabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Process interrupts: 40 • Interrupts for DPV1: 50 • Synchronous cycle interrupt: 60 • Redundancy error interrupts: 70 • Asynchronous error interrupts: 80 Entries continue to be made in the diagnostics buffer.
02	All new occurrences of a specified interrupt are disabled. You designate the interrupt using the OB number. Entries continue to be made in the diagnostics buffer.

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	The OB_NR input parameter contains an invalid value.
8091	The MODE input parameter contains an invalid value.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for DIS_IRT & EN_IRT (Page 2504).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

EN_IRT: Enable interrupt event (S7-1500)

Description

You use the instruction to enable the processing of new interrupts and asynchronous error events that you have previously disabled with the "DIS_IRT (Page 2501)" instruction. This means that if an interrupt event occurs, the CPU operating system reacts in one of the following ways:

- It calls an interrupt OB or asynchronous error OB.
  or
- it triggers the specified reaction if an interrupt OB or asynchronous error OB is not programmed.

Parameters

The following table shows the parameters of the "EN_IRT" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Specifies which interrupts and asynchronous error events will be enabled.
OB_NR	Input	INT	I, Q, M, D, L or constant	OB number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter MODE

MODE	Meaning
0	All newly occurring interrupts and asynchronous error events are enabled with the exception of the programming error and the error during direct access to I/O data and the Motion Control OBs.
1	All newly occurring events belonging to a specified interrupt class are enabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Process interrupts: 40 • Interrupts for DPV1: 50 • Synchronous cycle interrupt: 60 • Redundancy error interrupts: 70 • Asynchronous error interrupts: 80
2	All newly occurring events of a specified interrupt are enabled. You designate the interrupt using the OB number.

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	The OB_NR input parameter contains an invalid value.
8091	The MODE input parameter contains an invalid value.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for DIS_IRT & EN_IRT (Page 2504).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

Program example for DIS_IRT & EN_IRT

Introduction

In the following example you disable processing of a cyclic interrupt OB (OB33) with the help of the "DIS_IRT" instruction and enable OB33 once again with the instruction "EN_IRT".

Storage of data

Create the following structures and tags for storing the data in a global data block.

SLI_gDB_AsyncEvent

	Name	Data type	Start value
1	Static		
2	DIS_IRT	Struct	
3	execute	Bool	false
4	done	Bool	false
5	mode	Byte	16#02
6	obNbr	Int	33
7	returnValue	Int	0
8	EN_IRT	Struct	
9	execute	Bool	false
10	done	Bool	false
11	mode	Byte	16#02
12	obNbr	Int	33
13	returnValue	Int	0
14	testOB	Struct	
15	execute	Bool	false
16	testTime	Time	T#0ms
17	testTimeLimit	Time	T#2m
18	setOut	Bool	false

"SLI_ciOB_AsyncEvent" OB

You create an "SLI_ciOB_AsyncEvent" OB and monitor its behavior during disabling/enabling.

Network 1: You now call the instruction "TP" in the OB and create the following interconnections.

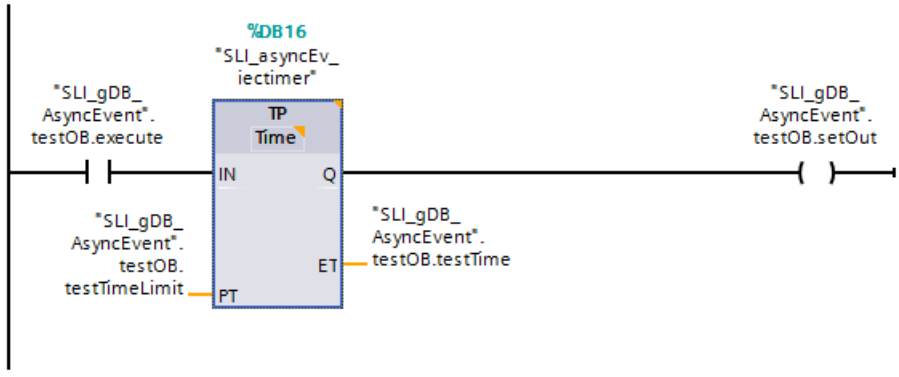

Network 2: Reset the tag "testOB.execute" as follows:

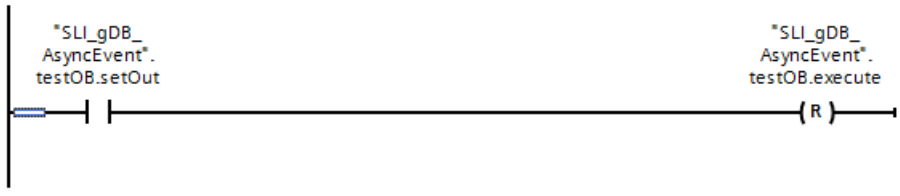

"SLI_FC_AsyncEvent" FC

To call the program parts, create the FC "SLI_FC_AsyncEvent".

Create the following local tags in the FC.

7		Temp	
8		tempDIS	Bool
9		tempEN	Bool

Network 1: Interconnect the parameters of the "DIS_IRT" instruction as follows.

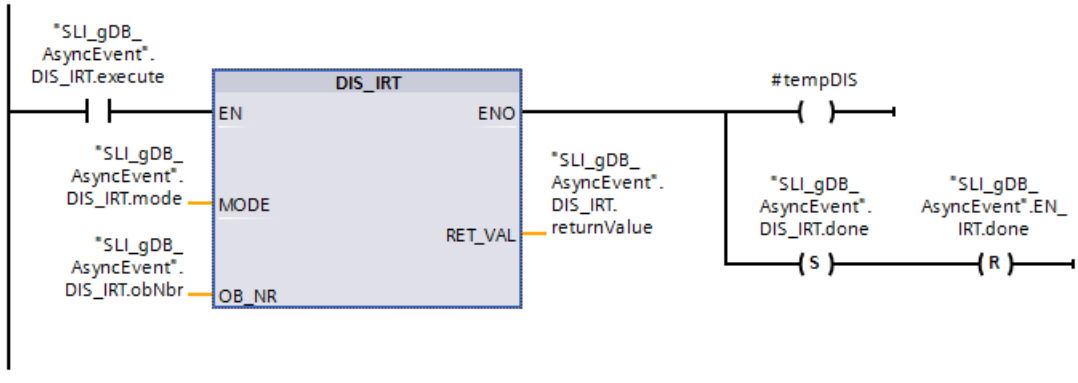

Network 2: Reset the tag "DIS_IRT.execute" as follows:

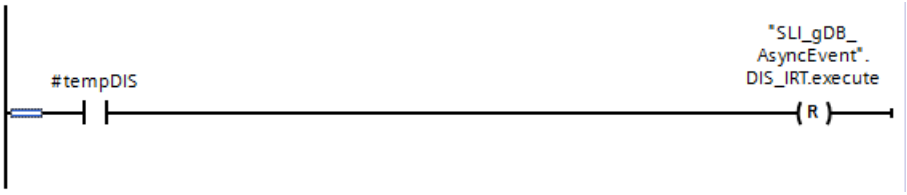

Network 3: Interconnect the parameters of the "EN_IRT" instruction as follows.

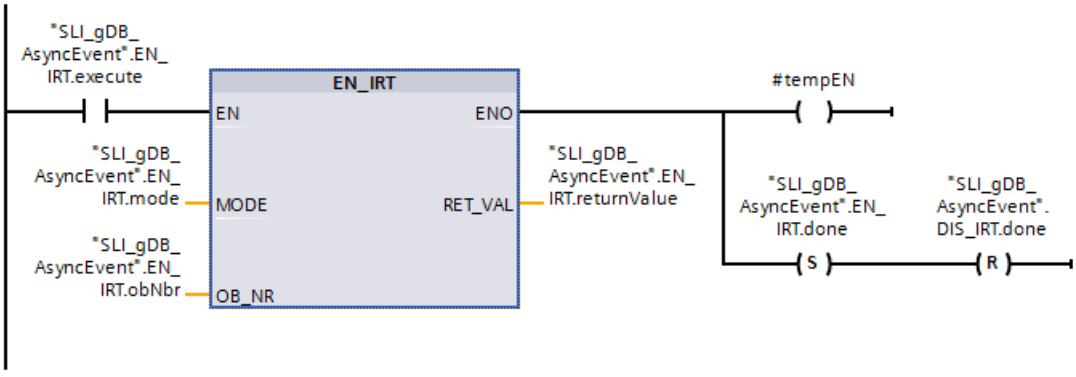

Network 4: Reset the tag "EN_IRT.execute" as follows:

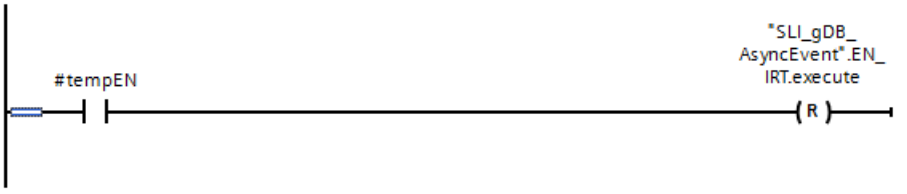

Result of OB "SLI_ciOB_AsyncEvent"

If the normally open contact ("testOB.execute") supplies the signal state "TRUE" once, the "TP" instruction is executed. The tag "testOB.execute" is automatically reset to the signal state "FALSE". The value for the "testTime" tag is incremented until it corresponds to the value of the "testTimeLimit" tag.

SLI_gDB_AsyncEvent

	Name	Data type	Start value	Monitor value
1	Static			
2	DIS_IRT	Struct		
3	execute	Bool	false	FALSE
4	done	Bool	false	FALSE
5	mode	Byte	16#02	16#02
6	obNbr	Int	33	33
7	returnValue	Int	0	0
8	EN_IRT	Struct		
9	execute	Bool	false	FALSE
10	done	Bool	false	FALSE
11	mode	Byte	16#02	16#02
12	obNbr	Int	33	33
13	returnValue	Int	0	0
14	testOB	Struct		
15	execute	Bool	false	FALSE
16	testTime	Time	T#0ms	T#19S_100MS
17	testTimeLimit	Time	T#2m	T#2M
18	setOut	Bool	false	TRUE

Result of FC "SLI_FC_AsyncEvent"

Network 1:

If the normally open contact ("DIS_IRT.execute") supplies the signal state "TRUE", the "DIS_IRT" instruction is executed. The operating mode "2" to be used is specified according to the input parameter MODE ("DIS_IRT.mode"). OB33 is addressed and its processing disabled according to the input parameter OB_NR ("DIS_IRT.obNbr").

The output parameter RET_VAL ("DIS_IRT.returnValue") indicates that processing in the example is taking place without errors. You save the processing of "DIS_IRT" in the tag "DIS_IRT.done". You also reset the "EN_IRT.done" tag.

The value of the "testOB.testTime" tag is no longer incremented. Nevertheless, the IEC timer of the "TP" instruction continues to count in the background.

SLI_gDB_AsyncEvent				
	Name	Data type	Start value	Monitor value
1	Static			
2	DIS_IRT	Struct		
3	execute	Bool	false	FALSE
4	done	Bool	false	TRUE
5	mode	Byte	16#02	16#02
6	obNbr	Int	33	33
7	returnValue	Int	0	0
8	EN_IRT	Struct		
9	execute	Bool	false	FALSE
10	done	Bool	false	FALSE
11	mode	Byte	16#02	16#02
12	obNbr	Int	33	33
13	returnValue	Int	0	0
14	testOB	Struct		
15	execute	Bool	false	FALSE
16	testTime	Time	T#0ms	T#28S_500MS
17	testTimeLimit	Time	T#2m	T#2M
18	setOut	Bool	false	TRUE

Network 2:

If the normally open contact ("#tempDIS") supplies the signal state "TRUE", the "DIS_IRT.execute" tag is reset.

Network 3:

If the normally open contact ("EN_IRT.execute") supplies the signal state "TRUE", the "EN_IRT" instruction is executed. The operating mode "2" to be used is specified according to the input parameter MODE ("EN_IRT.mode"). OB33 is addressed and its processing enabled once again according to the input parameter OB_NR ("EN_IRT.obNbr").

The output parameter RET_VAL ("EN_IRT.returnValue") indicates that processing in the example is taking place without errors. You save the processing of "EN_IRT" in the tag "EN_IRT.done". You also reset the "DIS_IRT.done" tag.

The value of the "testOB.testTime" tag is still incremented or changed according to the IEC timer of the "TP" instruction.

SLI_gDB_AsyncEvent

	Name	Data type	Start value	Monitor value
1	Static			
2	DIS_IRT	Struct		
3	execute	Bool	false	FALSE
4	done	Bool	false	FALSE
5	mode	Byte	16#02	16#02
6	obNbr	Int	33	33
7	returnValue	Int	0	0
8	EN_IRT	Struct		
9	execute	Bool	false	FALSE
10	done	Bool	false	TRUE
11	mode	Byte	16#02	16#02
12	obNbr	Int	33	33
13	returnValue	Int	0	0
14	testOB	Struct		
15	execute	Bool	false	FALSE
16	testTime	Time	T#0ms	T#1M_40S_800MS
17	testTimeLimit	Time	T#2m	T#2M
18	setOut	Bool	false	TRUE

Network 4:

If the normally open contact ("#tempEN") supplies the signal state "TRUE", the "EN_IRT.execute" tag is reset.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

DIS_IRT: Disable interrupt event (Page 2501)

EN_IRT: Enable interrupt event (Page 2503)

DIS_AIRT: Delay execution of higher priority interrupts and asynchronous error events (S7-1200, S7-1500)

Description

You use "DIS_AIRT" to delay the processing of interrupt OBs whose priority are higher than the priority of the current organization block.

You can call "DIS_AIRT" multiple times in an organization block. The "DIS_AIRT" calls are counted by the operating system. Processing is delayed more and more each time "DIS_AIRT" is executed. To cancel a delay, you must execute the "EN_AIRT (Page 2510)" instruction. The processing delay applies until the number of executions of "EN_AIRT (Page 2510)" matches the number of calls of "DIS_AIRT" or the current OB has been worked through.

Each call of "DIS_AIRT" increases the number of delays; the current value is made available to you at the RET_VAL parameter.

Parameters

The following table shows the parameters of the "DIS_AIRT" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of delays

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Return value	Description
n	"n" shows the number of processing delays, i.e., the number of DIS_AIRT calls, after the instruction is completed. Use EN_AIRT (Page 2510) to clear the delays.

Example

You can find the example here: Program example for DIS_AIRT & EN_AIRT (Page 2511).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

EN_AIRT: Enable execution of higher priority interrupts and asynchronous error events (S7-1200, S7-1500)

Description

You use "EN_AIRT" to enable processing of organization blocks when interrupts occur that have been delayed by the "DIS_AIRT (Page 2509)" instruction.

When "EN_AIRT" is executed, you cancel a processing delay that was registered by the operating system when "DIS_AIRT (Page 2509)" was called. To cancel all delays, the number of "EN_AIRT" executions must be equal to the number of "DIS_AIRT (Page 2509)" calls. Example: You have called "DIS_AIRT (Page 2509)" five times. As a result, the processing is also delayed five times. After this, you must call the "EN_AIRT" instruction five times in order to cancel all five delays.

In the EN_AIRT parameter of the "EN_AIRT" instruction, you query the number of interrupt delays that have not yet been enabled after the execution of "RET_VAL". The value "0" in the RET_VAL parameter means that all delays enabled by "DIS_AIRT (Page 2509)" have been cancelled.

Parameters

The following table shows the parameters of the "EN_AIRT" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of configured delays

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Return value/ Error code* (W#16#...)	Description
n	"n" shows the number of processing delays that are not yet enabled after the instruction is completed. If n = 0, interrupt handling is first enabled again.
8080	Although the interrupt processing was already enabled, the function was called again.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for DIS_AIRT & EN_AIRT (Page 2511).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

Program example for DIS_AIRT & EN_AIRT

Introduction

In the following example you delay processing of a cyclic interrupt OB or of interrupt OBs with a higher priority with the help of the instructions "DIS_AIRT" and "EN_AIRT". A test program is not interrupted by an interrupt when you call the instructions.

You use the test program to save the time and diagnostics data in a data memory.

- To do this, you read out the CPU time with the instruction "TIME_TCK" and the diagnostics data with the instruction "GET_DIAG" in a cyclic interrupt OB.
- In a cyclic OB (OB1), you also read out the system time of the CPU clock with the instruction "RD_SYS_T".

Storage of data

Create the PLC data type "testAIRT" for the data memory.

testAIRT

		Name	Data type	Default value
1		errorHW	Bool	false
2		outCPUtimer	Time	T#0ms
3		outputTIME	Date_And_Time	DT#1990-01-01-0

Create the following structures and tags for storing the data in a global data block.

- For DIS_AIRT, EN_AIRT, and time recording:

SLI_gDB_delayAIRT			
	Name	Data type	Start value
1	Static		
2	startOB1	Bool	false
3	startOB_ci	Bool	false
4	delay	Struct	
5	executeDel	Bool	false
6	executeEn	Bool	false
7	nbrOfDelays	Int	0
8	timeData	Struct	
9	retValRDsys	Int	0
10	outCPUtimer	Time	T#0ms

- For diagnostics:

Note

Find hardware identifier ("laddr")

Use "PLC tags > System constants" to find the hardware identifier. For an S7-1500 CPU: Find the entry "Local~Common" with the data type "HW_Submodule". The associated cell "value" contains the hardware identifier.

11	diag	Struct	
12	mode	UInt	2
13	laddr	HW_ANY	50
14	returnValue	Int	0
15	countDiagDetails	UInt	0
16	getDIAG	DNN	
17	SubordinateState	UInt	0
18	SubordinateIOState	Word	16#0
19	DNNmode	Word	16#0
20	itemiseIOState	Struct	
21	No_maintenance_required	Bool	false
22	The module_or_device_is_disabled	Bool	false
23	Maintenance_required	Bool	false
24	Maintenance_demanded	Bool	false
25	Error	Bool	false
26	The_module_or_the_device_can_not_be_reached_from_the_CPU	Bool	false
27	Inputs_or_outputs_are_not_available	Bool	false

- For the data memory:

28	dataStorage	Array[0..3] of "testAIRT"	
29	pos	Int	0

Interconnect parameters: in the FC "SLI_FC_itemise_IOState_AIRT"

To further evaluate the diagnostics data of the "GET_DIAG" instruction, create an FC "SLI_FC_itemise_IOState_AIRT".

Create the following local tags in the FC.

SLI_FC_itemise_IOState_AIRT		
	Name	Data type
1	Input	
2	ComponentStateDetail	Word
3	Output	
4	_____No_maintenance_required	Bool
5	_____The module_or_device_is_disabled	Bool
6	_____Maintenance_required	Bool
7	_____Maintenance_demanded	Bool
8	Error	Bool
9	_____The_module_or_the_device_can_not_be_reached_from_the_CPU	Bool
10	_____Inputs_or_outputs_are_not_available	Bool

Network 1: You create the following interconnections.

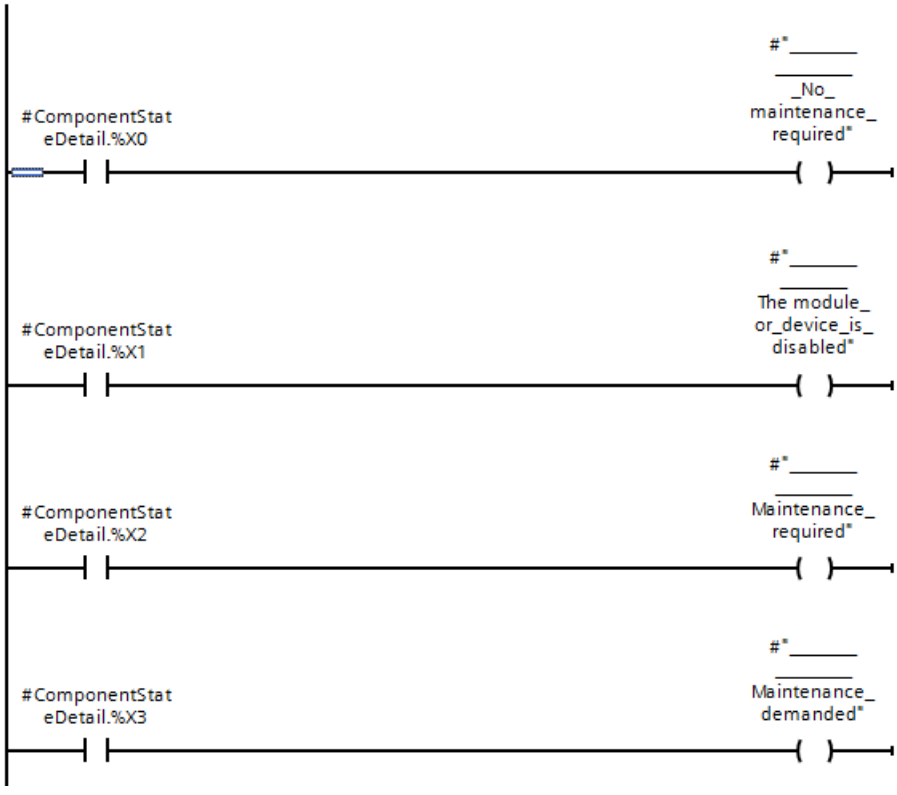

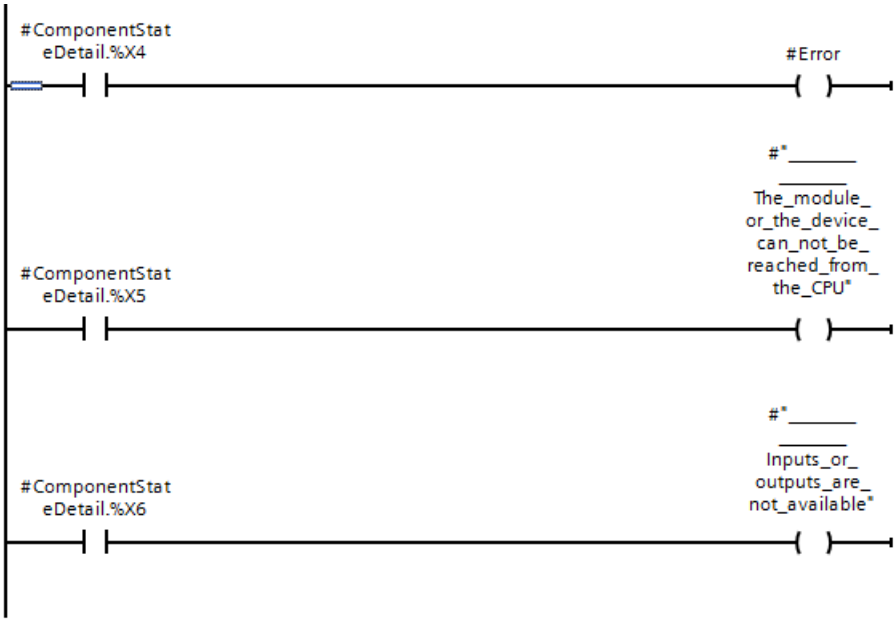

Interconnect parameters: in the "SLI_ciOB_diagProg_AIRT" OB

You create a cyclic interrupt OB "SLI_ciOB_diagProg_AIRT" and monitor its behavior during the delay.

You create the following local tag in the OB.

4		Temp	
5		tempReset	Bool

Network 1: You now call the instruction "GET_DIAG" in the OB and create the following interconnections.

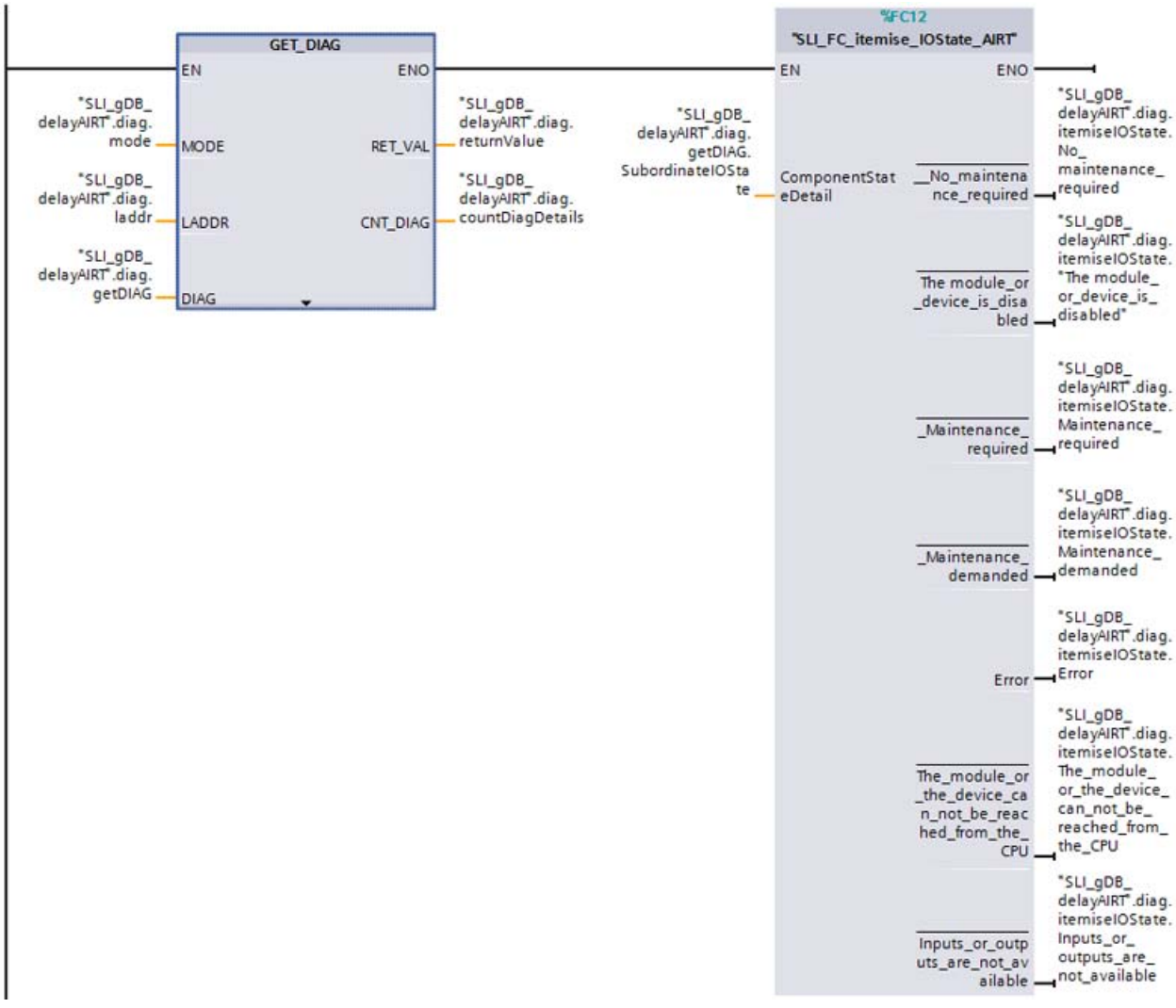

Network 2: You record the CPU time as follows and increment the "pos" tag according to the following condition.

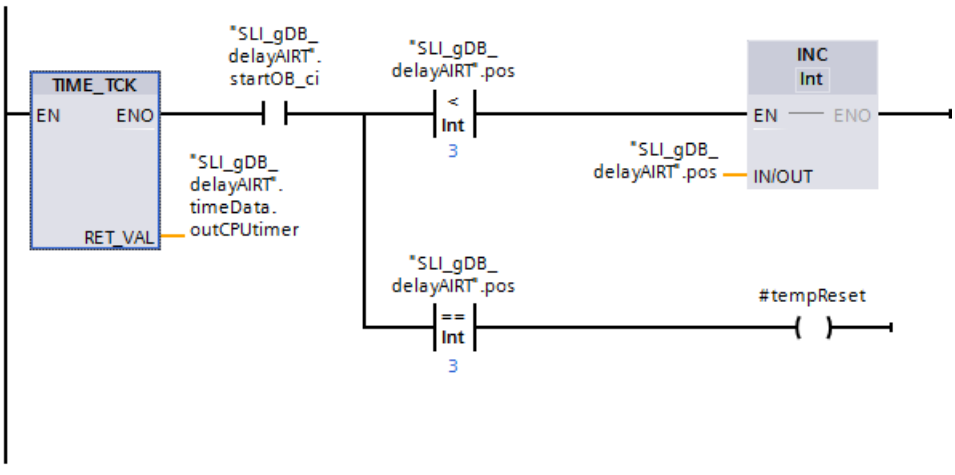

Network 3: Reset the tag "startOB_ci" as follows:

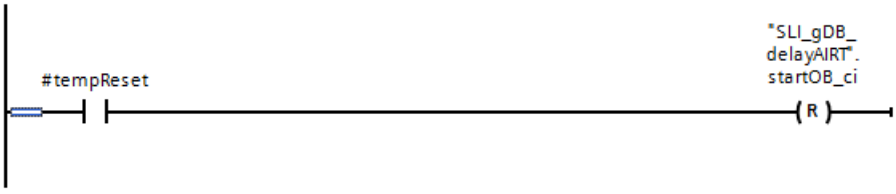

Interconnect parameters: in the FC "SLI_FC_saveDiag_AIRT"

To read the system time and merge the time and diagnostics data, create the FC "SLI_FC_saveDiag_AIRT".

Create the following local tags in the FC.

SLI_FC_saveDiag_AIRT		
	Name	Data type
1	Input	
2	start	Bool
3	Output	
4	<Add new>	
5	InOut	
6	<Add new>	
7	Temp	
8	tempReset	Bool

Network 1: You create the following interconnections.

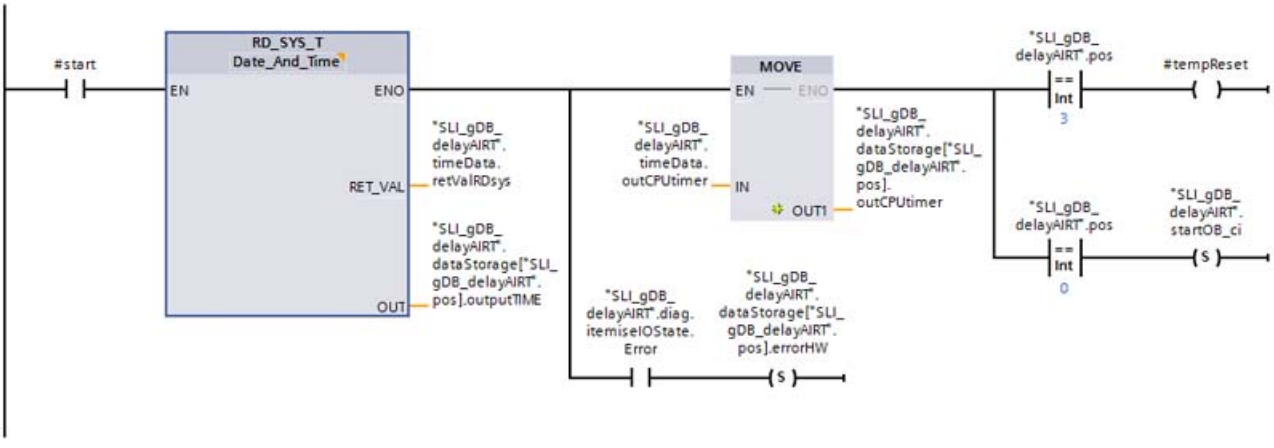

Network 2: Reset the tag "startOB1" as follows:

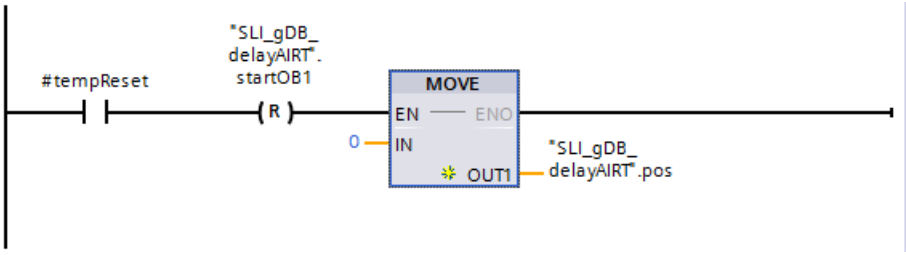

Interconnect parameters: in the FC "SLI_FC_delayAIRT"

To call the test program, create the FC "SLI_FC_delayAIRT".

Network 1: Interconnect the parameters of the "DIS_AIRT" instruction as follows.

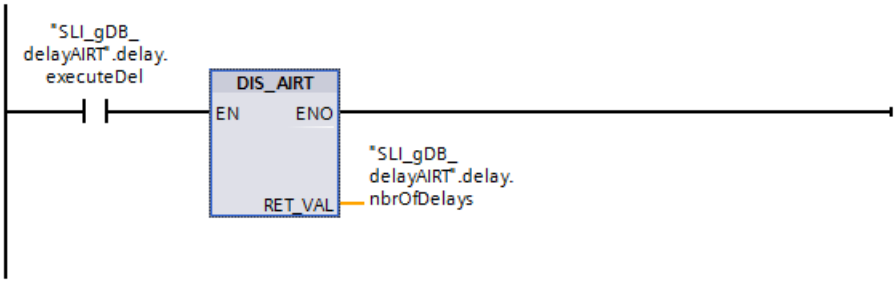

Network 2: Interconnect the FC "SLI_FC_saveDiag_AIRT" as follows.

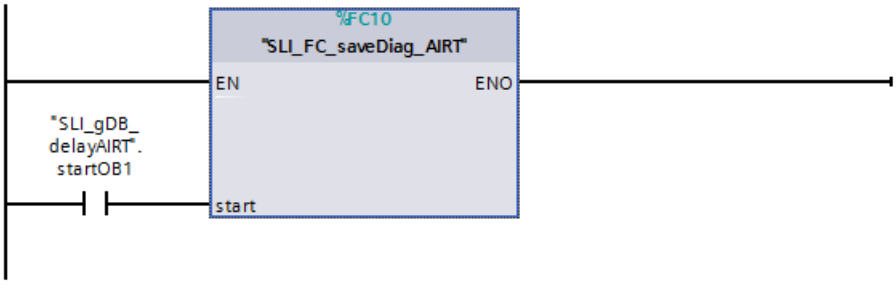

Network 3: Interconnect the parameters of the "EN_AIRT" instruction as follows.

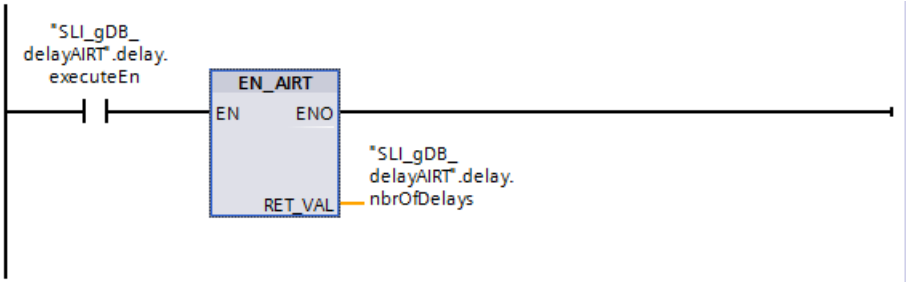

Procedure for starting the test scenario

To start the test program, follow these steps:

Note

If you do not want to delay processing of the cyclic interrupt OB ("SLI_ciOB_diagProg_AIRT"), skip steps 1 and 2.

1. Set the tag "delay.executeDel" to "TRUE".
2. Set the tag "delay.executeEn" to "TRUE".
3. Set the tag "startOB1" to "TRUE". Afterwards, the tag "startOB_ci" is set automatically. The "startOB1" tag and the "startOB_ci" tag are automatically reset after filling the data memory.

Procedure for stopping the processing of DIS_AIRT and EN_AIRT

To stop the delayed processing of the cyclic interrupt OB ("SLI_ciOB_diagProg_AIRT"), follow these steps:

1. Set the tag "delay.executeEn" to "FALSE".
2. Set the tag "delay.executeDel" to "FALSE".

Result of OB "SLI_ciOB_diagProg_AIRT"

The cyclic interrupt OB ("SLI_ciOB_diagProg_AIRT") is called every 100 ms. It reads the diagnostics data with the instruction "GET_DIAG" and outputs the CPU time.

If the normally open contact ("startOB_ci") supplies the signal state "TRUE" and the "pos" tag returns a value greater than "3", the "pos" tag is incremented. If the "pos" tag returns the value "3", the "startOB_ci" tag is reset.

SLI_gDB_delayAIRT

	Name	Data type	Start value	Monitor value
1	Static			
2	startOB1	Bool	false	FALSE
3	startOB_ci	Bool	false	FALSE
4	delay	Struct		
5	executeDel	Bool	false	FALSE
6	executeEn	Bool	false	FALSE
7	nbrOfDelays	Int	0	0
8	timeData	Struct		
9	retValRDsys	Int	0	0
10	outCPUtimer	Time	T#0ms	T#5M_26S_588MS

If execution of the cyclic interrupt OB ("SLI_ciOB_diagProg_AIRT") is not delayed, the values of the data memory appear as below.

28	dataStorage	Array[0..3] o...		
29	dataStorage[0]	"testAIRT"		
30	errorHW	Bool	false	FALSE
31	outCPUtimer	Time	T#0ms	T#2M_9S_388MS
32	outputTIME	Date_And_Time	DT#1990-01-01-0	DT#2017-01-20-09:58:24.384
33	dataStorage[1]	"testAIRT"		
34	errorHW	Bool	false	FALSE
35	outCPUtimer	Time	T#0ms	T#2M_9S_488MS
36	outputTIME	Date_And_Time	DT#1990-01-01-0	DT#2017-01-20-09:58:24.484
37	dataStorage[2]	"testAIRT"		
38	errorHW	Bool	false	FALSE
39	outCPUtimer	Time	T#0ms	T#2M_9S_588MS
40	outputTIME	Date_And_Time	DT#1990-01-01-0	DT#2017-01-20-09:58:24.585
41	dataStorage[3]	"testAIRT"		
42	errorHW	Bool	false	FALSE
43	outCPUtimer	Time	T#0ms	T#2M_9S_688MS
44	outputTIME	Date_And_Time	DT#1990-01-01-0	DT#2017-01-20-09:58:24.587
45	pos	Int	0	0

Result of FC "SLI_FC_delayAIRT"

Network 1:

If the normally open contact ("delay.executeDel") supplies the signal state "TRUE", the "DIS_AIRT" instruction is executed. Processing of the cyclic interrupt OB ("SLI_ciOB_diagProg_AIRT") or of an interrupt OB with higher priority is delayed. The output parameter RET_VAL ("delay.nbrOfDelays") indicates that a delay is active in the calling OB (OB1).

Network 2:

If the normally open contact ("startOB1") supplies the signal state "TRUE", the test program of the FC "SLI_FC_saveDiag_AIRT" is executed.

In the FC "SLI_FC_saveDiag_AIRT":

The system time is read with the instruction "RD_SYS_T" and stored in the data memory "dataStorage". The cell of the data memory that is to be used is determined by the "pos" tag.

The CPU time ("outCPUtimer") is also stored in the data memory. If the "diag.itemiseIOState.Error" tag has the signal state "TRUE", the "errorHW" tag is set to "TRUE" in the data memory.

The following is decided in the next steps:

- If "pos" has the value "0": The value of the "startOB_ci" tag is set to "TRUE". This means the "pos" tag can be incremented in the cyclic interrupt OB ("SLI_ciOB_diagProg_AIRT").
- If "pos" has the value "3": The tags "startOB1" and "pos" are reset.

If execution of the cyclic interrupt OB ("SLI_ciOB_diagProg_AIRT") is delayed, the values of the data memory appear as below.

28	dataStorage	Array[0..3] o...		
29	dataStorage[0]	"testAIRT"		
30	errorHW	Bool	false	FALSE
31	outCPUtimer	Time	T#0ms	T#25M_52S_688MS
32	outputTIME	Date_And_Time	DT#1990-01-01-0	DT#2017-01-20-10:22:07.684
33	dataStorage[1]	"testAIRT"		
34	errorHW	Bool	false	FALSE
35	outCPUtimer	Time	T#0ms	T#25M_52S_788MS
36	outputTIME	Date_And_Time	DT#1990-01-01-0	DT#2017-01-20-10:22:07.784
37	dataStorage[2]	"testAIRT"		
38	errorHW	Bool	false	FALSE
39	outCPUtimer	Time	T#0ms	T#25M_52S_888MS
40	outputTIME	Date_And_Time	DT#1990-01-01-0	DT#2017-01-20-10:22:07.884
41	dataStorage[3]	"testAIRT"		
42	errorHW	Bool	false	FALSE
43	outCPUtimer	Time	T#0ms	T#25M_52S_988MS
44	outputTIME	Date_And_Time	DT#1990-01-01-0	DT#2017-01-20-10:22:07.887
45	pos	Int	0	0

Network 3:

If the normally open contact ("delay.executeEn") supplies the signal state "TRUE", the "EN_AIRT" instruction is executed. Processing of the cyclic interrupt OB ("SLI_ciOB_diagProg_AIRT") or of an interrupt OB with higher priority is enabled once again. The output parameter RET_VAL ("delay.nbrOfDelays") indicates that a delay is not active in the calling OB (OB1). Processing in the example takes place without errors.

SLI_gDB_delayAIRT

	Name	Data type	Start value	Monitor value
1	Static			
2	startOB1	Bool	false	FALSE
3	startOB_ci	Bool	false	FALSE
4	delay	Struct		
5	executeDel	Bool	false	TRUE
6	executeEn	Bool	false	TRUE
7	nbrOfDelays	Int	0	0
8	timeData	Struct		
9	retValRDsys	Int	0	0
10	outCPUtimer	Time	T#0ms	T#27M_36S_988MS

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

DIS_AIRT: Delay execution of higher priority interrupts and asynchronous error events (Page 2509)

EN_AIRT: Enable execution of higher priority interrupts and asynchronous error events (Page 2510)

4.1.4.8 Alarms

Program_Alarm: Generate program alarm with associated values (S7-1500)

Description

The instruction "Generate program alarm with associated values" monitors a signal and on a signal change at the parameterSIG generates a program alarm (for definition, see also: Introduction to alarm configuration (Page 7493)). An incoming program alarm is generated with a signal change from 0 to 1 and an outgoing program alarm is generated with a signal change from 1 to 0. The program alarm is triggered synchronously to program execution.

You can append up to ten associated values to the program alarm at the parametersSD_i (0 ≤ i ≤10). The associated values are detected at the time of signal change at parameter SIG and assigned to the program alarm. For more information on configuring associated values, refer to the following sections: Insert associated values in alarms (Page 7514), Examples of associated values (Page 7517).

Each incoming or outgoing alarm is assigned a time stamp:

- The current system time of the PLC when the signal change occurs is used by default (default value at the TIMESTAMP parameter).
- If you wish to specify a different time stamp, you can create one at the parameter TIMESTAMP.
  The time value must always be specified in system time (i.e. UTC) as this is the time used for time synchronization throughout the plant.
- If an alarm is to be time-stamped with a local time, a conversion module, which converts the local time to system time, must be connected in series. This is the only way to make sure that the time stamp is shown properly in the alarm display.

To use the current system time of the CPU, set the TIMESTAMP parameter to its default value (LDT#1970-01-01-00:00:00.0).

Total size of the associated values for program messages

The alarm procedure of the S7-1500 CPU allows a maximum of 512 bytes for associated values (SD parameters) of the instruction "Program_Alarm".

You will only see a warning regarding this number in the TIA Portal <= V14 during compilation. There is a stricter check as of TIA Portal V14 SP1. If this number is exceeded, an error is output during compilation to rule out error scenarios during runtime. This check from now on no longer refers to the actual size of the SD parameters at the time the instruction "Program_Alarm" is called, but to the maximum possible size of the SD parameters. This means you must decide beforehand with what length you declare the character string tags you are transferring, as these take up a large number of bytes.

This must be taken into account in the migration of projects from version <= V14 to version V14 SP1 Update 1.

Call of the instruction "Generate program alarm with associated values"

Call the instruction only in a function block (FB). The block is processed synchronously. An alarm is triggered as soon as the block is exited. If there is an error during processing, an error code is output.

Once the instruction has been inserted in the FB, a multi-instance of the data type "Program_Alarm" is created in the "Static" section of the block interface. Choose any name for this multi-instance in the dialog which appears. It is also the name of the program alarm.

Finally, add the parameters of the instruction in line with your requirements (see the "Parameters" table).

Configuring the program alarm

The program alarm settings are displayed in the "Properties" window when you select the name of the program alarm in the "Static" section or in the network of the FB. Select the alarm class, priority, etc. and edit the alarm text.

The settings that are made here can be edited in the project tree. To do this, open the "Alarms" tab under "PLC supervisions & alarms"; in it open the "Program alarms" tab. All existing program alarms are then displayed in the "Alarm types" table.

Parameters

The following table shows the parameters of the instruction "Generate program alarm with associated values":

Parameter	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L, T, C or constant	The signal to be monitored. • Positive signal edge: An incoming program alarm is generated • Negative signal edge: An outgoing program alarm is generated
TIMESTAMP	Input	LDT	M, D, L or constant	This parameter is used to assign an alarm a time stamp, for example, from an input signal with a distributed stamp. The time value must always be specified in system time (i.e. UTC), as this is the time used for time synchronization throughout the plant. • "Not assigned" means that the system time of the CPU will be used as the interrupt time stamp when a signal change occurs (default). • Any system time input will be used as the interrupt time stamp when a signal change occurs. Note: If an interrupt is to be time-stamped with a local time, a conversion module must be connected in series to convert the local time to system time. This is the only way to make sure that the time stamp is shown correctly in the interrupt display. Note: When acknowledging the alarm, the PLC system time is always used as time stamp.
SD_i	Input	VARIANT	I, Q, M, D, L	i-th associated value (1 ≤ i ≤ 10) You can use binary numbers, integers, floating-point numbers, or character strings as associated values.

Parameter	Declaration	Data type	Memory area	Description
Error	Output	BOOL	I, Q, M, D, L	Status parameter Error Error = TRUE indicates that an error has occurred during processing. The possible error cause is displayed at the Status parameter.
Status	Output	WORD	I, Q, M, D, L	Status parameter Status Display of error information (see "Error and Status parameters").

You can find additional information on valid data types under "See also".

Error and Status parameters

The following table contains all specific error information that is output via the Error and Status parameters.

Error	Status*	Explanation
0	0000	No error, or the instruction was not processed because there was no signal edge at the parameter SIG.
1	0085	"Only information" alarm type
1	8001	Invalid static alarm information
1	8002	No valid static alarm information
1	8004	Maximum size of 512 bytes reached for the associated values of the alarm.
1	8005	A positive signal edge is pending at parameter SIG and there is still an alarm pending which has not yet been acknowledged.
1	8007	Outgoing alarm which was not preceded by an incoming alarm.
1	8087	Static alarms are deactivated.
1	8089	Alarm is too long.
1	80Ax	The SD_i parameter has an invalid value.
1	80C1	The CPU does not generate any alarms at this time because initialization routines are running (this is the case after download in RUN, for example). Try again later.
1	80C2	The maximum number of alarms per time unit has been sent. Try again later.
1	80C3	All dynamic alarm instances are being used. Try again later.
1	80C4	Alarm is output and cannot be overwritten. Try again later.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".		

Example

In the following example, you generate a program alarm with associated value for a signal change.

Create one tag in a global data block for storing the signal value to be monitored.

Program_Alarm_Tags			
	Name	Data type	Start value
1	Static		
2	CreateAlarm	Bool	false

The instruction is called in a function block. Create four parameters for the function block for interconnection of the instruction.

ProgramAlarm			
	Name	Data type	Default value
1	Input		
2	<Add new>		
3	Output		
4	<Add new>		
5	InOut		
6	<Add new>		
7	Static		
8	MyStaticText	String	'MyStaticText'
9	Program_Alarm_1	Program_Alarm	
10	Temp		
11	OUTPUT_Error	Bool	
12	OUTPUT_Status	Word	

Interconnect the parameters of the instruction as follows.

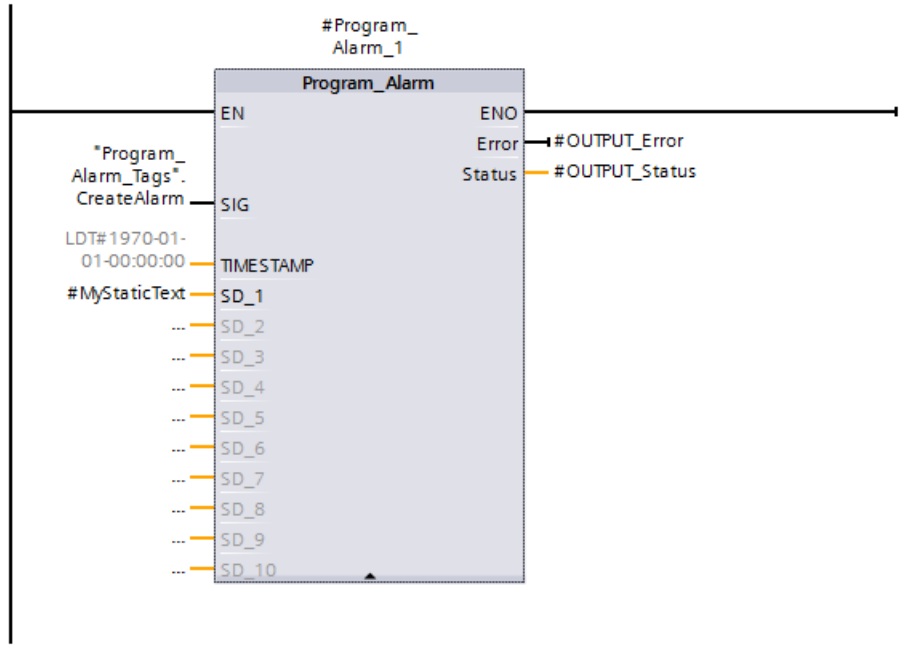

Call the function block in an OB.

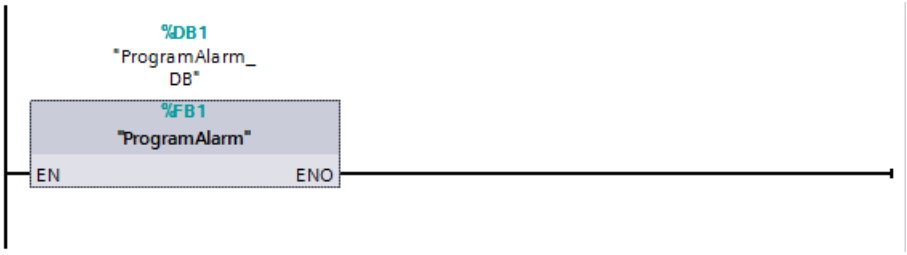

A PLC alarm is generated automatically when the instruction is created. Open the PLC supervisions & alarms dialog and select the alarm to be edited in the Alarms > Program alarms tab. Create the alarm text, including two key words in the Properties > Basic settings of the alarm.

Note: If you right-click in the text box, you can insert a key word, tag, or text list.

As a result of the character sequence "@1%s@", the value of parameter SD_1 ("#MyStaticText") is read out and output as a character string.

See also:

- Insert associated values in alarms (Page 7514)
- Examples of associated values (Page 7517)
- Keywords for alarm texts (Page 7519)

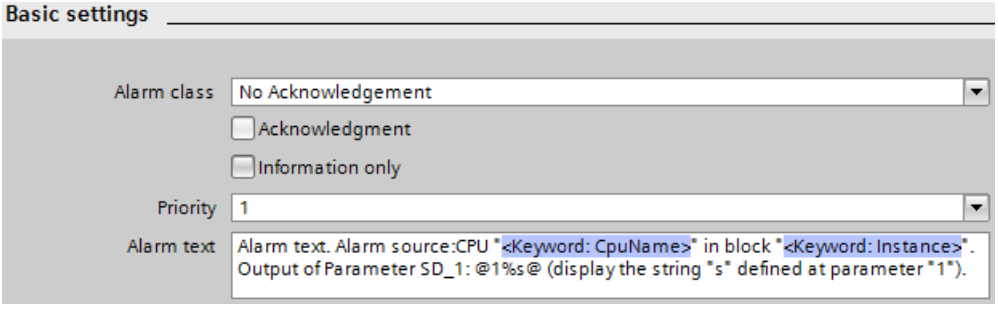

If parameter SIG ("CreateAlarm") has the signal state "TRUE", the PLC alarm is output. At output parameter STATUS ("OUTPUT_Status"), the value "0001" indicates that a signal change has occurred. Then, it is indicated that no further processing is taking place (the value is "0000"). At parameter SD_1 ("#MyStaticText"), the associated value of the PLC alarm is output.

A time stamp can be transferred to parameter TIMESTAMP. If not interconnected, parameter TIMESTAMP outputs the local time of the CPU clock. At output parameter ERROR ("OUTPUT_Error"), it is indicated that the processing of the instruction was error-free.

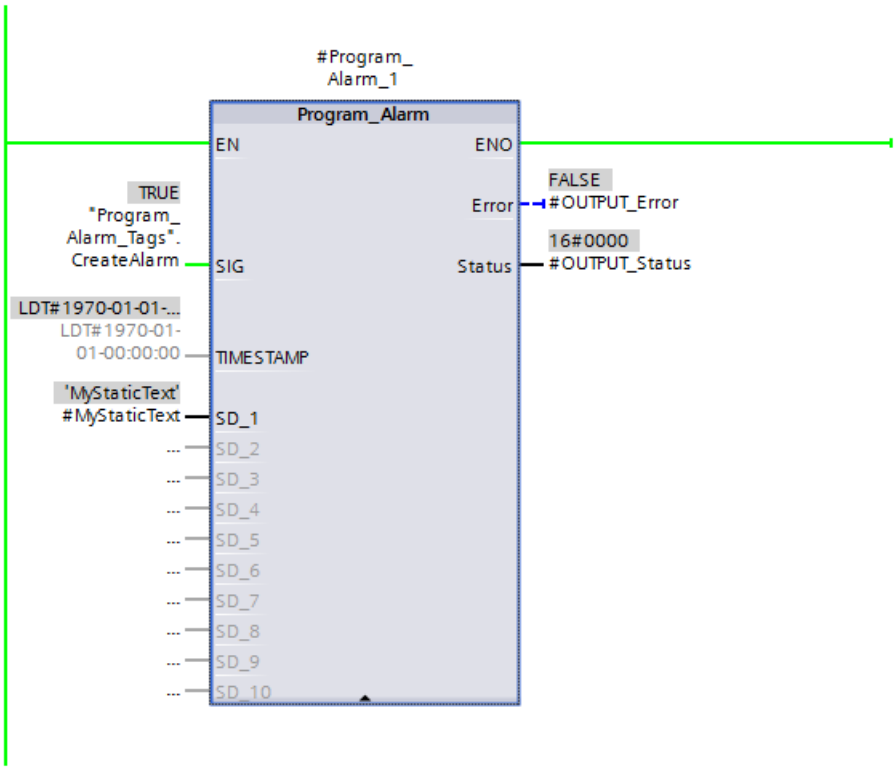

Use the web server of the CPU, for example, to output the PLC alarm. The following is necessary in order to use the web server:

- The web server must be activated in the CPU settings.

Use the Internet browser to open the Web server (via the IP address of the CPU) and log on in the Web server menu. The CPU outputs the alarm text as long as the signal being monitored ("CreateAlarm") supplies the value "TRUE".

Time	Message text
02:32:09.226 pm	Alarm text. Alarm source:CPU "PLC_2" in block "ProgramAlarm_DB". Output of Par

Program code

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

Example

You can find a detailed application example in the Siemens Industry Online Support (https://support.industry.siemens.com/cs/document/98210758?dti=0&lc=en-WW).

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Get_AlarmState: Output alarm status (S7-1500)

Description

Use the instruction "Output alarm status" to output the alarm status of a program alarm.

There are three possible alarm statuses:

- Incoming
- Outgoing
- Acknowledged

The output of the alarm status always refers to a program alarm that was created using the instruction "Generate program alarm with associated values".

The program alarm is selected with the Alarm input parameter. Specify the instance DB of the instruction "Generate program alarm with associated values" at the parameter Alarm.

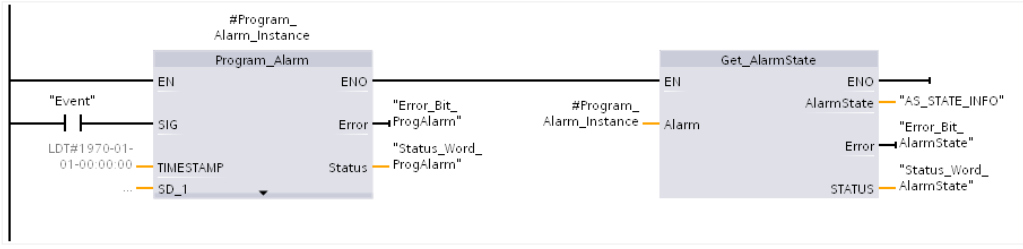

The alarm status is returned in a byte via the AlarmState output parameter. The meaning of the individual bits is shown in the following figure:

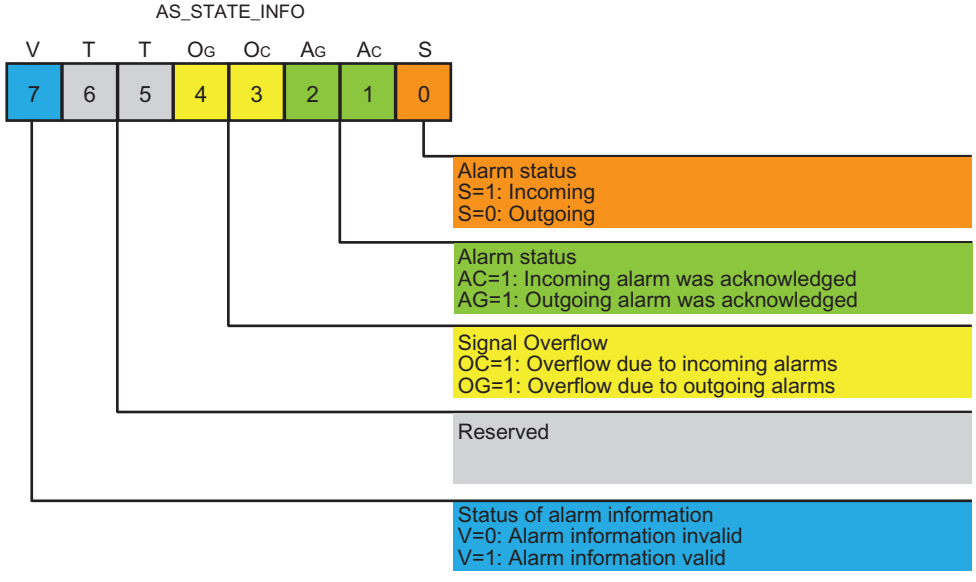

The execution status of the instruction is shown with the output parameters Error and STATUS.

Parameters

The following table lists the parameters of the instruction "Output alarm status":

Parameter	Declaration	Data type	Memory area	Description
Alarm	Input	ALARM_BASE	D	Instance of the instruction "Generate program alarm with associated values" • Alarm.Messagetype = Alarm_AP, the the bit Ac has either the signal status 0 or 1 and the bit Ag has the signal status 1 – not active: 0x86 (1000 0110) – active/not acknowledged: 0x85 (1000 0101) – active/acknowledged: 0x87 (1000 0111) – outgoing/not acknowledged: 0x84 (1000 0100) • Alarm.Messagetype = Notify_AP, the bits Ac and Ag both have the signal state 1. – not active: 0x86 (1000 0110) – active: 0x87 (1000 0111) • Alarm.Messagetype = Inforeport_AP, the bits Ac and Ag both have the signal status 1 – not active: 0x86 (1000 0110) – (temporarily) active: 0x87 (1000 0111) If the alarm is not active, bit S always has the signal state 0.
AlarmState	Output	BYTE	I, Q, M, D, L	Status of the alarm as a bit array

Parameter	Declaration	Data type	Memory area	Description
Error	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy STATUS to a free data area.

You can find additional information on valid data types under "See also".

Parameter STATUS

Error code* (W#16#...)	Explanation
8001	Invalid static alarm instance
8002	ID of the alarm is invalid
8003	No active alarms within the alarm class • Alarm_AP: The alarm is outgoing and acknowledged. • Notify_AP: The alarm is outgoing. • Inforeport Bit V is set to signal status 0.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you output the alarm status of a program alarm.

Create four tags in a global data block for storing the data.

My_gDB_Get_AlarmState

	Name	Data type	Start value
1	Static		
2	AS_statusBYTE	Byte	16#0
3	outputERROR	Bool	false
4	outputSTATUS	Word	16#0
5	CreateAlarm	Bool	false

The "Get_AlarmState" instruction is called together with the "Program_Alarm" instruction in a function block. Create four parameters for the function block for interconnection of the instructions.

7	Static		
8	MyStaticText	String	'MyStaticText'
9	Program_Alarm_2	Program_Alarm	
10	Temp		
11	OUTPUT_Error	Bool	
12	OUTPUT_Status	Word	

Interconnect the parameters of the instructions as follows.

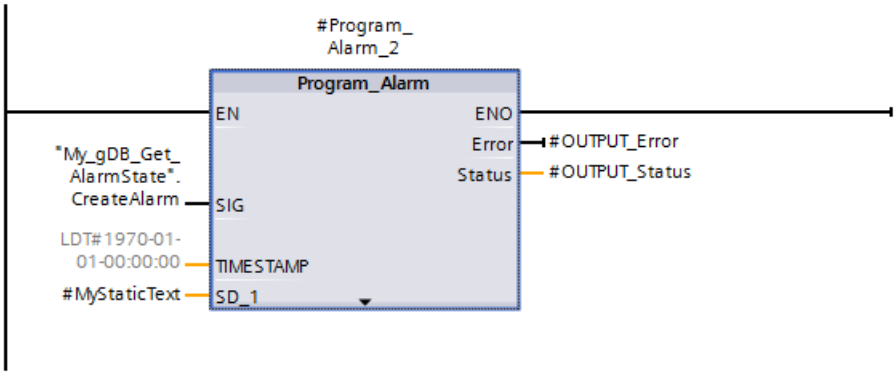

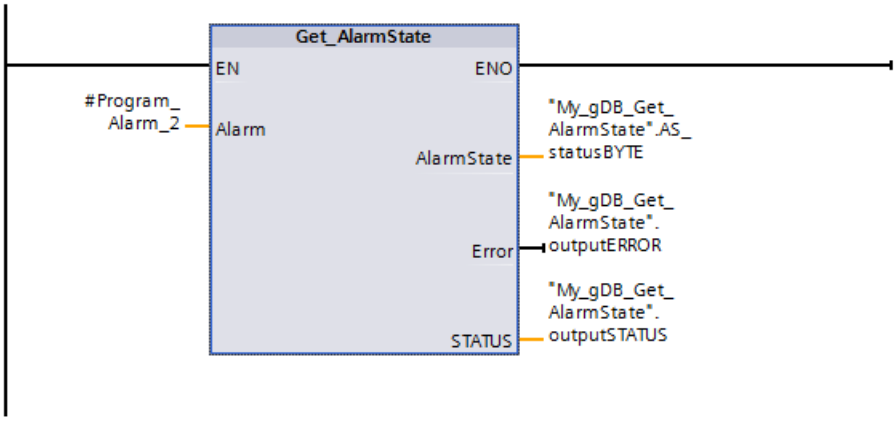

Call the function block in an OB.

A PLC alarm is generated automatically when the "Program_Alarm" instruction is created. Open the PLC supervisions & alarms dialog and select the alarm to be edited in the Alarms > Program alarms tab. Create the alarm text, including two key words.

Select the following settings for the alarm.

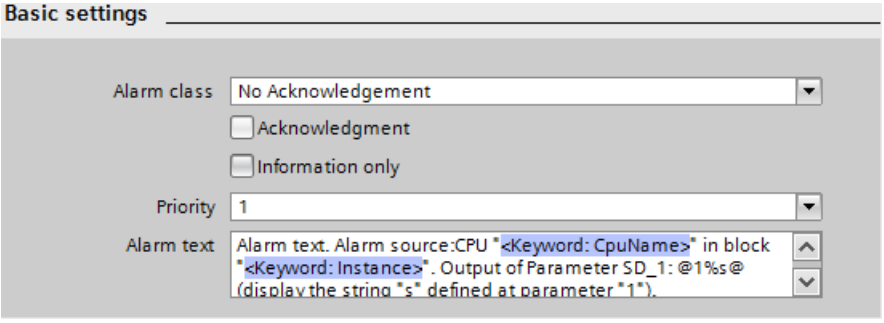

If input parameter SIG ("CreateAlarm") of the "Program_Alarm" instruction has the signal state "TRUE", the PLC alarm is output.

In the case of the "Get_AlarmState" instruction, the following occurs: The PLC alarm is communicated to the "Get_AlarmState" instruction using input parameter ALARM. At output parameter AlarmState ("AS_statusBYTE"), it is indicated that the alarm corresponding to the alarm class "No Acknowledgement" is active.

At output parameters ERROR ("outputERROR") and STATUS ("outputSTATUS"), it is indicated that the processing of the instruction is proceeding error-free.

My_gDB_Get_AlarmState

	Name	Data type	Start value	Monitor value
1	Static			
2	AS_statusBYTE	Byte	16#0	16#85
3	outputERROR	Bool	false	FALSE
4	outputSTATUS	Word	16#0000	16#0000
5	CreateAlarm	Bool	false	TRUE

For the output of the PLC alarm, use the web server of the CPU.

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593)

Example

ou can find a detailed application example in the Siemens Industry Online Support. (https://support.industry.siemens.com/cs/document/98210758?dti=0&lc=en-WW)

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Program_Alarm: Generate program alarm with associated values (Page 2522)

Creating and editing alarms (Page 7511)

Gen_UsrMsg: Generate user diagnostic alarms

Description

You use the "Gen_UsrMsg" instruction to generate an alarm that is entered in the diagnostic buffer.

You use the Mode parameter to select whether an incoming or an outgoing alarm is to be generated:

- When Mode = 1: Create an incoming alarm.
- When Mode = 2: Create an outgoing alarm.
- Regardless of whether an incoming or outgoing alarm is generated, the alarm always has the attribute "Information only".

The entry in the diagnostic buffer is created synchronously. Alarm transmission is asynchronous.

If an error occurs during the execution of an instruction, it is output via the parameter RET_VAL.

Content of the alarm

The content of the alarm is defined with a text list:

- Specify the text list you want to use with the parameter TextListID. For this purpose, open the dialog "PLC alarm text lists" in the project navigation. Show the column "ID" in the dialog "Text lists". Apply the ID at the parameter TextListID.
- Use the parameter TextID to select the text list entry you want to write in the diagnostic buffer. For this purpose select an entry from the "Text lists entries" dialog by applying a number from the columns "Range from / range to" at the parameter TextID. The same number must be used from both the "Range from" and "Range to" columns for the text list entry.
- Additional information about text lists can be found here: AUTOHOTSPOT

Defining associated values

New associated values to be added to the alarm are defined in the text list entry:

- Associated values are defined by adding the following information in the text list entry: @<No. of the associated value><Element type><Format specification>@
- Use the system data type AssocValues to specify which associated value is added when generating the alarm.
- Additional information on the structure of associated values can be found here: Structure of associated values (Page 7508)

Parameters

The following table shows the parameters of the "Gen_UsrMsg" instruction:

Parameter	Declaration	Data type	Memory area	Description
Mode	Input	UInt	I, Q, M, D, L or constant	Parameters for selecting the status of the alarm: • 1: incoming alarm • 2: outgoing alarm
TextID	Input	UInt	I, Q, M, D, L or constant	ID of the text list entry that should be used for the alarm text.
TextListID	Input	UInt	I, Q, M, D, L or constant	ID of the text list that contains text list entry.
Ret_Val	Return	Int	I, Q, M, D, L	Error code of the instruction.
AssocValues	InOut	VARIANT	D, L	Pointer to the system data type AssocValues that allows you to define the associated values.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter AssocValues

Use the system data type AssocValues to define which associated values will be sent. A maximum of 8 associated values are possible. The structure of the system data type is created by entering the data type "AssocValues" as a data block.

The associated values are selected by entering the numbers of the associated values for the parameters Value[x]. Note the following:

- The values for TextID and TextListID are treated by the instruction as the associated values to be sent. As a result, "1" and "2" are already assigned as numbers for addressing associated values. Do not use the numbers "1" or "2" to address associated values.
- Address the associated value at parameter Value [1] as number "3", at parameter Value [2] as number "4", and so forth.

Byte	Parameter	Data type	Start value	Description	Number of the associated value
0..1	Value[1]	UINT	0	First associated value of the alarm.	3
2..3	Value[2]	UINT	0	Second associated value of the alarm.	4
4..5	Value[3]	UINT	0	...	5
6..7	Value[4]	UINT	0	...	6
8..9	Value[5]	UINT	0	...	7
10..11	Value[6]	UINT	0	...	8
12..13	Value[7]	UINT	0	...	9
14..15	Value[8]	UINT	0	Eighth associated value of the alarm.	10

Parameter RET_VAL

The following table contains all specific error information that is output via the RET_VAL parameter.

Error code* (W#16#...)	Explanation
0000	No error
8080	Value in the Mode parameter is not supported.
80C1	Resource bottleneck due to too many parallel calls.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you generate an alarm that is entered in the diagnostic buffer.

Create five tags and an "AssocValues" structure (with the AssocValues data type) in a global data block for storing the data.

My_gDB_Gen_UsrMsg			
	Name	Data type	Start value
1	Static		
2	myMODE	UInt	1
3	myTEXT	UInt	3
4	myLIST	UInt	512
5	returnValue	Int	0
6	execute	Bool	false
7	AssocValues	AssocValues	
8	Value	Array[1..8] of UInt	
9	Value[1]	UInt	33
10	Value[2]	UInt	411
11	Value[3]	UInt	578
12	Value[4]	UInt	6122
13	Value[5]	UInt	722
14	Value[6]	UInt	8111
15	Value[7]	UInt	9829
16	Value[8]	UInt	10457

Interconnect the parameters of the instruction as follows.

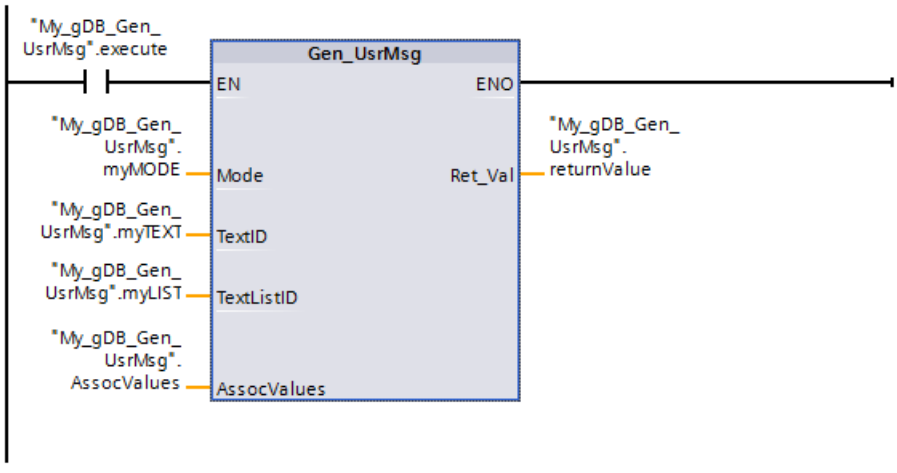

Use the "Text lists" entry to create a Text list and a Text list entry for the alarm. Apply the ID of the text list at parameter TextListID ("myLIST"). Apply the ID (Range ...) of the text list entry at parameter TextID ("myTEXT"). Assign the alarm parameters as follows.

Text lists

Name	Id	Selection	Comment
USER_1	512	Decimal	

Text list entries of USER_1

Range from	Range to	Entry
3	3	This is a user generated message. Return Value[3]: @5I%6d@

If the normally open contact ("execute") supplies the signal state "TRUE", the "Gen_UsrMsg" instruction is executed. According to the value of parameter Mode ("myMODE"), an incoming alarm is generated. The alarm to be output is made known to the instruction using parameters TextListID ("myLIST") and TextID ("myTEXT"). The associated values of the alarm are transferred using parameter AssocValues ("AssocValues").

When the alarm is generated, the character string "@5I%6d@" contained in the alarm text is interpreted as follows:

- The associated value with the number "5" is read out in the INT data type. The number corresponds to parameter Value[3] of the "AssocValues" structure.
- The associated value is output as a decimal number. The decimal number is limited to six places.

At output parameter Ret_Val ("returnValue"), it is indicated that the processing of the instruction was error-free.

My_gDB_Gen_UsrMsg

	Name	Data type	Start value	Monitor value
1	Static			
2	myMODE	UInt	1	1
3	myTEXT	UInt	3	3
4	myLIST	UInt	512	512
5	returnValue	Int	0	0
6	execute	Bool	false	FALSE
7	AssocValues	AssocValues		
8	Value	Array[1..8] of UInt		
9	Value[1]	UInt	33	33
10	Value[2]	UInt	411	411
11	Value[3]	UInt	578	578
12	Value[4]	UInt	6122	6122
13	Value[5]	UInt	722	722
14	Value[6]	UInt	8111	8111
15	Value[7]	UInt	9829	9829
16	Value[8]	UInt	10457	10457

For the output of the alarm, for a CPU of the S7-1500 series, open the entry "Online & Diagnostics > Diagnostic buffer".

Details on event: 1 of 1000 Event ID: 16# 2D2:0003

Description: User message: This is a user generated message. Return Value[3]: 578

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Example

You will find a detailed application example in the Siemens Industry Online Support (https://support.industry.siemens.com/cs/document/98210758?dti=0&lc=en-WW).

See also

Switching display formats in the program status (Page 7698)

Creating and editing alarms (Page 7511)

Get_Alarm: Read pending alarm (S7-1500)

Description

Using the "Read pending alarms" instruction, you can log on for alarms generated in the user program and copy an incoming or outgoing alarm to a referenced data block. This allows you to pass on for example alarms as a STRING to a master computer.

When you call the instruction with the "OperateMode = 1" and log on with this, you have to specify at least one or more display classes at the input "DispClassNr". After logging on no more alarms are lost either incoming or outgoing.

The alarms are processed sequentially, starting with the oldest alarm. They are read out and stored in a data block in the CPU. You can then forward the data, e.g. using the instruction "TSEND_C", to a master computer.

The instruction operates asynchronously and therefore avoids long runtimes of the function block "Get_Alarm [FB702]" that could lead to a cycle overflow. The instruction must be called recurrently in a cycle organization block.

Because the function block "Get_Alarm [FB702]" needs up to 11 calls per requested language for each retrieved alarm, the signal state "1" can be present at the Busy parameter for up to 33 calls. Retrieval of an alarm is not complete until the Busy parameter once again has the signal state "0".

This results in the recommended call time of the "Get_Alarm [FB702]" between 20 ms and 200 ms. If the call time exceeds 1.8 seconds, the "Get_Alarm [FB702]" may log off automatically. As a result of the logoff, no more alarms can be read out.

In the following cases, you need to log on the instruction again using the input "OperateMode = 1":

- After automatic logoff because the function block "Get_Alarm [FB702]" is called too rarely.
- When the operating mode changes from "STOP" > "RUN".
- As soon as you change existing alarm configurations, such as the Program_Alarm or ProDiag, the instruction automatically logs off.

You can use the instruction for alarms from the following areas:

- ProDiag
- System diagnostics
- GRAPH
- Program_Alarm
- Motion Control

Note

CPU property "Central alarm management in the PLC"

If you do not enable the CPU property "Central alarm management in the PLC" in the device configuration, only the IDs of the alarm text fields are output instead of the alarm texts (alarm text, information text and additional alarm texts 1 - 9).

Parameter

The following table shows the parameters of the instruction Read pending alarm":

Parameter	Declaration	Data type	Memory area	Description
EN *)	Input	BOOL	I, Q, M, D, L	Enable input
ENO *)	Output	BOOL	I, Q, M, D, L	Enable output
Operate-Mode	Input	BYTE	I, Q, M, D, L	There are three possible values: • 1: Logon • 2: Read out data • 3: Logoff
DataMode	Input	BYTE	I, Q, M, D, L	There are four different possible values: • 0: Only the column titles are read out (Producer-ID, ID_1, ID_2, Prio, Timestamp, State) • 1: The column headers and the alarm text is read out including associated values. • 2: The column headers, the alarm text including associated values and the information text are read out. • 3: The column headers, the alarm text including associated values, the information text and the additional alarm texts are read out.

Parameter	Declaration	Data type	Memory area	Description
DispClassNr (Display class number)	Input	ARRAY[0..31] of UINT	I, Q, M, D, L	You can define up to 32 different display classes: • 0 ... 16: User-defined display classes that can be set on the HMI device • 257: System diagnostics • 259: Security If you do not use the parameter, all display classes are reported. A change of the display class is only taken into consideration if mode 1 is set at the parameter mode.
Lcid (Local ID)	Input	UDINT	I, Q, M, D, L	There are different possible values: • 0: All languages • 1: First language with the lowest LC number • 2: First language plus second language with the second lowest LC number • 3: All languages • or: Enter the exact LC number to select a specific language.
Data	InOut	"AlarmData"	I, Q, M, D, L	Target range for the alarm data that was read. Alarm text structure from the PLC data type "AlarmData" according to the value at the "DataMode" parameter.
DataReady	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No data is available. • 1: The data is available. The data is valid until the next call of the instruction. Therefore, it must be processed in the same program cycle, e.g. using a move instruction.
Busy	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: The execution of the instruction has not yet been started or the asynchronous processing was completed. Additional information on the processing status can be found in the STATUS parameter. • 1: Execution of the instruction not yet complete Further entries exist
Error	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.

Parameter	Declaration	Data type	Memory area	Description
Init	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No action necessary. • 1: A new logon is required. The alarm data received so far are no longer valid. This may be necessary in the following situations: • You have logged on. • The CPU changes from "STOP" mode to "RUN" mode. • Alarms were re-configured and the changes were made by means of "Download to device" in "RUN" mode, for example, a change in the alarm text structure.
Status	Output	WORD	I, Q, M, D, L	Error and status information
*): The enable input EN and the enable output ENO are only visible in the programming languages LAD and FBD.				

Parameter Status

The following table shows the meaning of the values of the Status parameter:

Status code* (W#16#...)	Explanation
0000	No error
0001	New data is available.
8001	The value at parameter OperateMode is invalid.
8002	The Data parameter has the wrong data type.
8003	The Lcid parameter is invalid or a language selected with the Lcid parameter is not loaded.
8004	You have already logged on via the OperateMode = 1 parameter in the user program to read the pending messages. You are only allowed and required to log on again if you log out using the OperateMode = 3, have re-configured alarm or performed a STOP > RUN transition.
8005	The value at parameter DataMode is invalid.
8085	• You did not specify a display class when you logged on. • The instruction was logged off during the reconfiguration of supervisions (ProDiag). • Automatic logoff because the function block "Get_Alarm [FB702]" is called too rarely.
80C5	The processing of the instruction is not yet completed. Please try again later.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

PLC data type "AlarmData"

The PLC data type "AlarmData" reflects the structure of an alarm.

Name	Data type	Default value	Description
ProducerID	UINT	0	There are different possible values: • 0: Program_Alarm • 4: System diagnostics • 5: Standard Motion Control (T-CPU) • 6: Security (for example, false login) • 7: SINUMERIK • 8: GRAPH • 9: ProDiag
ID_1	UINT	0	Alarm ID This ID is also displayed in the alarm editor.
ID_2	UINT	0	ID of the alarm runtime instance This ID is an internal range ID that is significant for ProDiag and sequencers: • In ProDiag, it is the supervision ID. • In a sequencer, it is the step number. For the "Program_Alarm" instruction the ID is 0, and it has no significance for system diagnostics.
PRIO	UINT	0	Priorities: 0 to 16
TimeStamp	LDT	DT#1970-01-01-0:0:0.000000000	Time stamp The time stamp is based on UTC (Universal Time Coordinated) or the system time.
State	BYTE	16#0	Status parameter: • 0: Outgoing • 1: Incoming
LC_ID	ARRAY[0..2] of UINT	-	Returns the value that is specified at the input "Lcid": • 0: All languages • 1: First language with the lowest LC number • 2: First language plus second language with the second lowest LC number • 3: All languages or: Enter the exact LC number to select a specific language.
AlarmText	ARRAY[0..2] of WSTRING	-	Alarm text with up to 254 characters (508 bytes). If the text is longer, it is truncated at 251 characters, for example WSTRING#'Meld... .
InfoText	ARRAY[0..2] of WSTRING	-	Information text
AddText_1 - 9	ARRAY[0..2] of WSTRING	-	Additional text 1 - 9

Example

You can find the example here: Program example for Get_Alarm & Ack_Alarms - Part 1 (Page 2545).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Insert associated values in alarms (Page 7514)

Examples of associated values (Page 7517)

Introduction to alarm configuration (Page 7493)

Ack_Alarms: Acknowledge alarms (S7-1500)

Description

You can use the "Acknowledge alarms" instruction to acknowledge all incoming and outgoing alarms in your program code with a single command.

The instruction is called as soon as there is a positive signal edge either at the enable input EN (LAD / FBD) or as a preceding logic operation (STL / SCL), and the value = 1 is set at the MODE parameter. The instruction operates asynchronously and therefore avoids long runtimes of the function block "Ack_Alarms [FB701]" that could lead to a cycle overflow. The CPU acknowledges all incoming and outgoing alarms and informs all HMI devices about this acknowledgment. Any pending alarms on the displays disappear as soon as they "went out".

In each case, only one acknowledgment (max. 100 alarms requiring acknowledgment) can be carried out at the same time. If additional alarms are pending, the next 100 alarms requiring acknowledgment will be confirmed automatically.

The instruction can also be used for ProDiag supervision alarms.

Parameters

The following table shows the parameters of the instruction "Acknowledge alarms":

Parameter	Declaration	Data type	Memory area	Description
EN *)	Input	BOOL	I, Q, M, D, L	Enable input
ENO *)	Output	BOOL	I, Q, M, D, L	Enable output
MODE	Input	UINT	I, Q, M, D, L	There are two different possible values: • 0: The instruction is not executed. • 1: Confirm all alarms. All active alarms that can be confirmed are confirmed.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Error and status information
*): The enable input EN and the enable output ENO are only visible in the programming languages LAD and FBD.				

Parameter STATUS

The following table shows the meaning of the values of the STATUS parameter:

Status code* (W#16#...)	Explanation
0000	No error
7001	No error occurred and the next 100 alarms requiring acknowledgment will be confirmed.
8001	The value at parameter MODE is invalid.
80C5	A confirmation process is already running. Please try again later.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

You will find the example here: Program example for Get_Alarm & Ack_Alarms - Part 1 (Page 2545).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Insert associated values in alarms (Page 7514)

Examples of associated values (Page 7517)

Introduction to alarm configuration (Page 7493)

Program example for Get_Alarm & Ack_Alarms

Program example for Get_Alarm & Ack_Alarms - Part 1

Introduction

In the following example, you collect all alarms of the CPU 1 (S7-1500 as of firmware V2.0) and save the alarms. You then use a programmed connection to the CPU 2 (S7-1500) to send the alarms (in abbreviated form; in PLC data type "rcvGetAlarm") from CPU 1 to CPU 2. You also save the alarms on CPU 2.

In the first part of the programming example for Get_Alarm and Ack_Alarms, you deal with the requirements and interconnection of the parameters.

The results and the procedure for the connection are described here: Program example for Get_Alarm & Ack_Alarms - Part 2 (Page 2562).

Note

Size of the memory arrays for the alarms

The memory arrays for the alarms ("alarmStorage" and "rcvdStorage") can hold up to 16 alarms when configured as described.

If you want to change the memory size, you must change the array size and the maximum value "maxPosition".

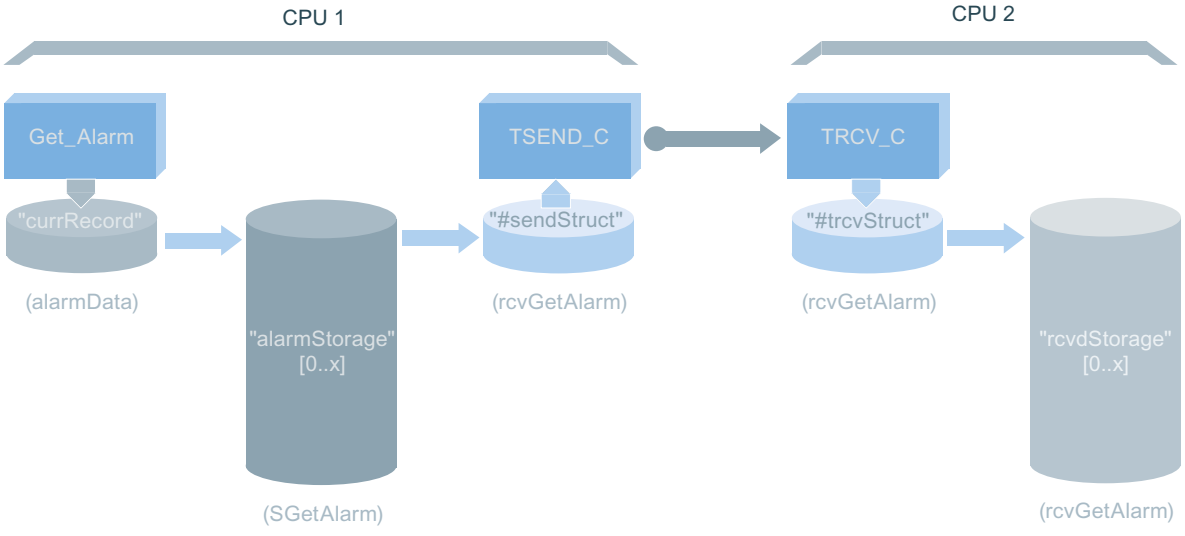

Requirements for a programmed connection

You create two CPUs (S7-1500) and connect the CPUs with each other over PROFINET. You do not assign parameters to the connection.

Note

Using Get_Alarm and Ack_Alarms

The CPU 1 (for the call of Get_Alarm and Ack_Alarms) must have firmware v2.0 or higher. In addition, the CPU property "Central alarm management in the PLC (Page 2538)" must be activated for the CPU 1.

Note

Permitting read and write access

By setting a low protection level under Properties > Protection of the CPU, you ensure that read and write access is permitted.

CPU 1: Program requirements

Creating PLC data types

You need the following PLC data types for the transfer and storage of alarms:

- You need "SGetAlarm" to save the alarms, including information about the current read loss

SGetAlarm

	Name	Data type	Default value
1	countReadLoss	UDInt	0
2	alarm	AlarmData	

- You need "rcvGetAlarm" to transfer the alarms over TSEND_C.

rcvGetAlarm

	Name	Data type	Default value
1	countReadLoss	UDInt	0
2	ProducerID	UInt	0
3	ID_1	UInt	0
4	ID_2	UInt	0
5	PRIO	UInt	0
6	TimeStamp	LDT	LDT#1970-01-01-
7	State	Byte	16#0
8	LCID	Array[0..2] of UInt	
9	AlarmText	WString[254]	WSTRING#''

Storage of data

Create the following structures and tags for storing the data in a global data block ("SLI_gDB_GAalarm").

- For parameter assignment of TSEND_C:

SLI_gDB_GAalarm			
	Name	Data type	Start value
1	Static		
2	TSEND_C	Struct	
3	start	Bool	false
4	stop	Bool	false
5	comControl	Bool	false
6	done	Bool	false
7	busy	Bool	false
8	error	Bool	false
9	status	Word	16#0
10	memErrStatus	Word	16#0

- For parameter assignment of Get_Alarm:

SLI_gDB_GAalarm			
	Name	Data type	Start value
11	GetAlarm	Struct	
12	start	Bool	false
13	stop	Bool	false
14	check	Struct	
15	anew	Bool	false
16	operateMode	Struct	
17	value	Byte	16#0
18	Logon_done	Bool	false
19	ReadData_ready	Bool	false
20	ReadData_active	Bool	false
21	Logoff_done	Bool	false
22	outputMode	Byte	16#3
23	languageCodeID	UDInt	0
24	currRecord	AlarmData	
25	currReadLoss	UDInt	0
26	dataReady	Bool	false
27	initiateLogin	Bool	false
28	busy	Bool	false
29	status	Word	16#0
30	memError	Bool	false
31	memErrStatus	Word	16#0

- As part of the "GetAlarm" structure for saving the alarms in the memory array:

32	maxPosition	UInt	15
33	alarmStorage	Array[0..15] of "SGetAlarm"	
34	limitReach	Bool	false
35	savingDone	Bool	false
36	initiateSending	Bool	false
37	countAlarms	UInt	0
38	posForAlarm	UInt	0

- For parameter assignment of Ack_Alarms:

40	AckAlarm	Struct	
41	executeAck	Bool	false
42	error	Bool	false
43	status	Word	16#0

ProDiag alarm

Create three tags in a global data block ("SLI_gDB_testAlarm_GAA") for a test scenario.

SLI_gDB_testAlarm_GAA

	Name	Data type	Start value
1	Static		
2	startPump	Bool	false
3	closeValve	Bool	false
4	valveOut	Bool	false

Create the following rule in the global data block ("SLI_gDB_testAlarm_GAA") to trigger a ProDiag alarm. To do so, open the global data block > the properties of the tag "startPump" > Supervisions > Add new supervision.

General

Type of supervision:	Operand		
Supervised tag:	"SLI_gDB_testAlarm_GAA".startPump	Trigger:	True
Delay time:	T#0ms		
Condition 1:	"SLI_gDB_testAlarm_GAA".closeValve	C1 trigger:	True
Condition 2:		C2 trigger:	True
Condition 3:		C3 trigger:	True
Category:	2: Warning		
Subcategory 1:			
Subcategory 2:			

CPU 2: Program requirements

Creating PLC data type

Create the PLC data type "rcvGetAlarm" as described for CPU 1.

Storage of data

Create the following structures and tags for storing the data in a global data block ("SLI_gDB_trcv_GAA").

SLI_gDB_trcv_GAA			
	Name	Data type	Start value
1	Static		
2	TRCV_C	Struct	
3	start	Bool	false
4	comControl	Bool	false
5	done	Bool	false
6	busy	Bool	false
7	error	Bool	false
8	status	Word	16#0
9	memErrStatus	Word	16#0
10	rcvLen	UDInt	0
11	dataReceived	Bool	false
12	GetAlarm	Struct	
13	maxPosition	UInt	15
14	rcvdStorage	Array[0..15] of "rcvGetAlarm"	
15	limitReach	Bool	false
16	savingDone	Bool	false
17	countAlarms	UInt	0
18	posForAlarm	UInt	0

CPU 1: FB "SLI_FB_send_GAA" - Interconnect parameters

Create the FB "SLI_FB_send_GAA" to set up the connection between CPU 1 and CPU 2 and data transfer.

Create the following local tags in the FB.

SLI_FB_send_GAA				
	Name	Data type	Default value	Retain
1	Input			
2	start	Bool	false	Non-ret...
3	connect	Bool	false	Non-retain
4	Output			
5	<Add new>			
6	InOut			
7	done	Bool	false	Non-retain
8	Static			
9	posForAlarm	UInt	0	Set in IDB
10	sendStruct	"rcvGetAlarm"		Non-retain
11	initiateSending	Bool	false	Set in IDB
12	do	Bool	false	Non-retain
13	do2	Bool	false	Non-retain
14	Temp			
15	tempDone	Bool		
16	Constant			
17	<Add new>			

In the FB, you use:

- Interconnections for canceling ("TSEND_C.stop") the data transfer.
- Interconnections for starting ("#do2") the data transfer.
- Interconnections to initiate a new data transfer ("#initiateSending", "#do").
- The instruction "TSEND_C" for connection establishment and data transfer.

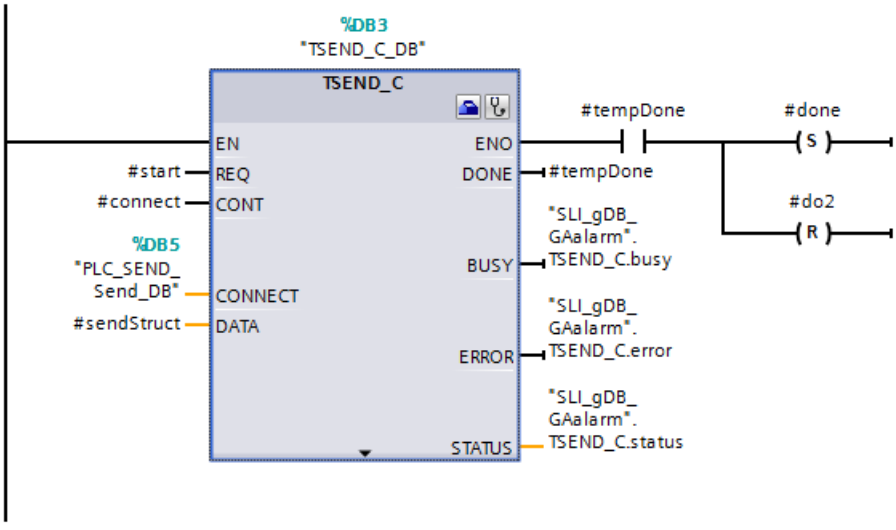

Note

Interconnection of the input parameter CONNECT of the "TSEND_C" instruction

To interconnect the input parameter CONNECT, open the wizard of the "TSEND_C" instruction with "Properties > Configuration".

Make the following settings for the connection of TSEND_C:

- End point Select the CPUs using the drop-down list. Interface, subnet and address are entered automatically.
- Connection data: Use the drop-down list and the selection "New" to create one data block per CPU. This data block is required for storing the connection data. The name of the data block can be freely selected for each CPU (communication partner).
- Select the connection type "TCP" and the configuration type "Use program block". These settings set up an Ethernet connection with the "TCP" protocol and as programmed connection.
- You enter a freely selected connection ID. Use a connection ID that has not already been assigned.
- You enter a value for the partner port. The value must be >=2000.

Additional information on the configuration of TSEND_C (Page 3482) is available here: Program example for send functions (Page 3506).

Additional information on setup and interconnections of the FB "SLI_FB_send_GAA" is available in the program code of the example in the Sample Library for Instructions (Page 593).

CPU 1: FC "SLI_FC_switchMode_GAA" - Interconnect parameters

You use the FC "SLI_FC_switchMode_GAA" to control the "Get_Alarm" instruction.

Create the following local tags in the FC.

SLI_FC_switchMode_GAA		
	Name	Data type
1	Input	
2	start	Bool
3	stop	Bool
4	Output	
5	<Add new>	
6	InOut	
7	mode2Reinit	Bool
8	Temp	
9	tempStart	Bool
10	tempStop	Bool
11	Constant	
12	<Add new>	
13	Return	
14	SLI_FC_switchMode_GAA	Void

Create the following interconnections in the FC.

Network 1: To start the processing of Get_Alarm, create the following interconnections.

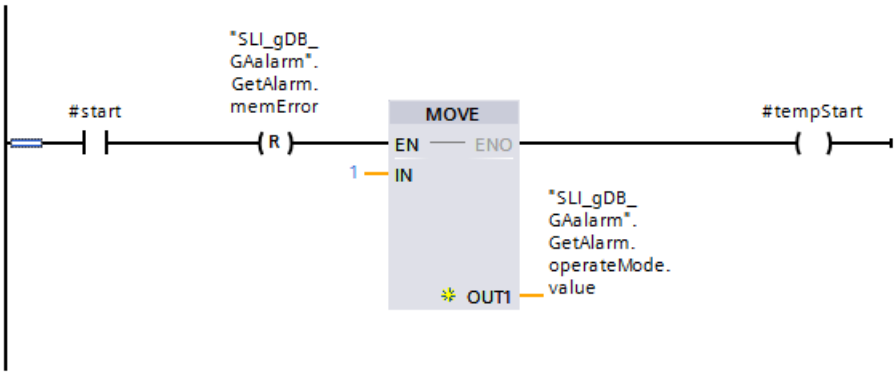

Network 2: To start the processing of Get_Alarm, create the following interconnections (Part 2).

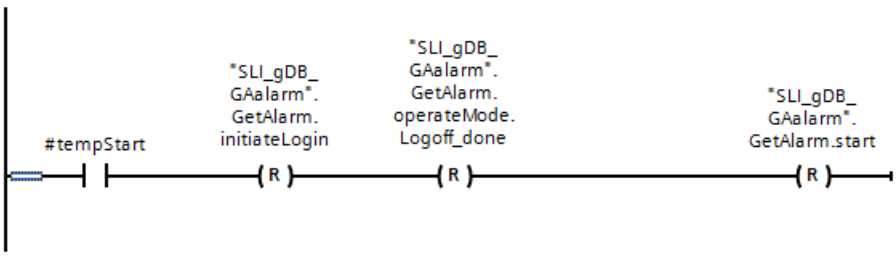

Network 3: For login of Get_Alarm, create the following interconnections.

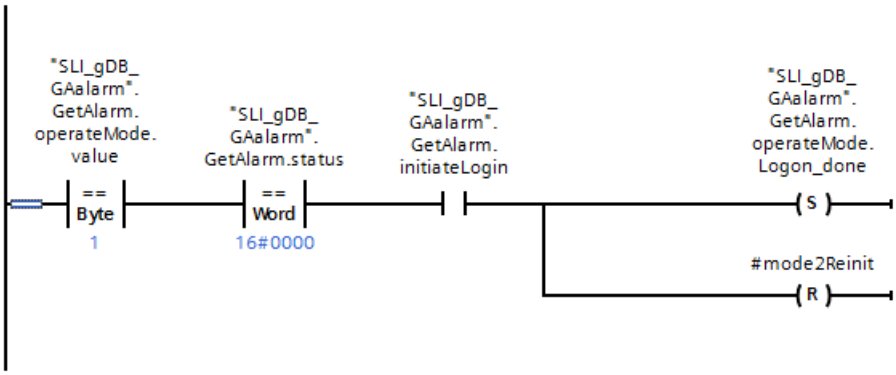

Network 4: To switch to the "Read" mode of Get_Alarm, create the following interconnections.

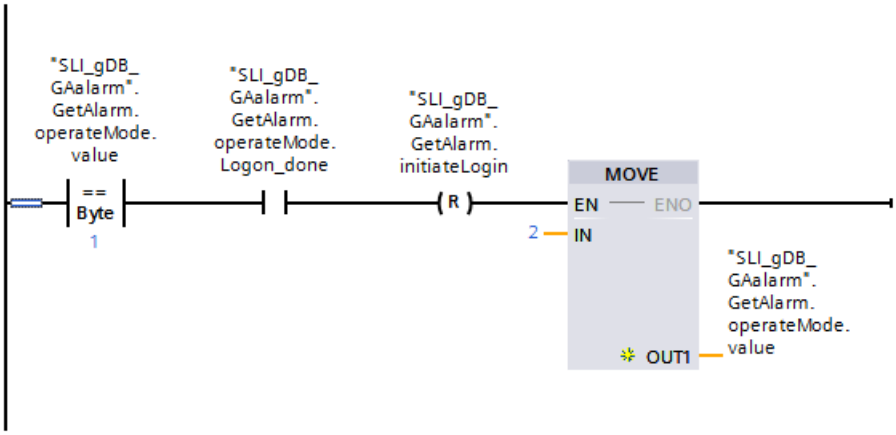

Network 5: To process the "Read" mode of Get_Alarm, create the following interconnections.

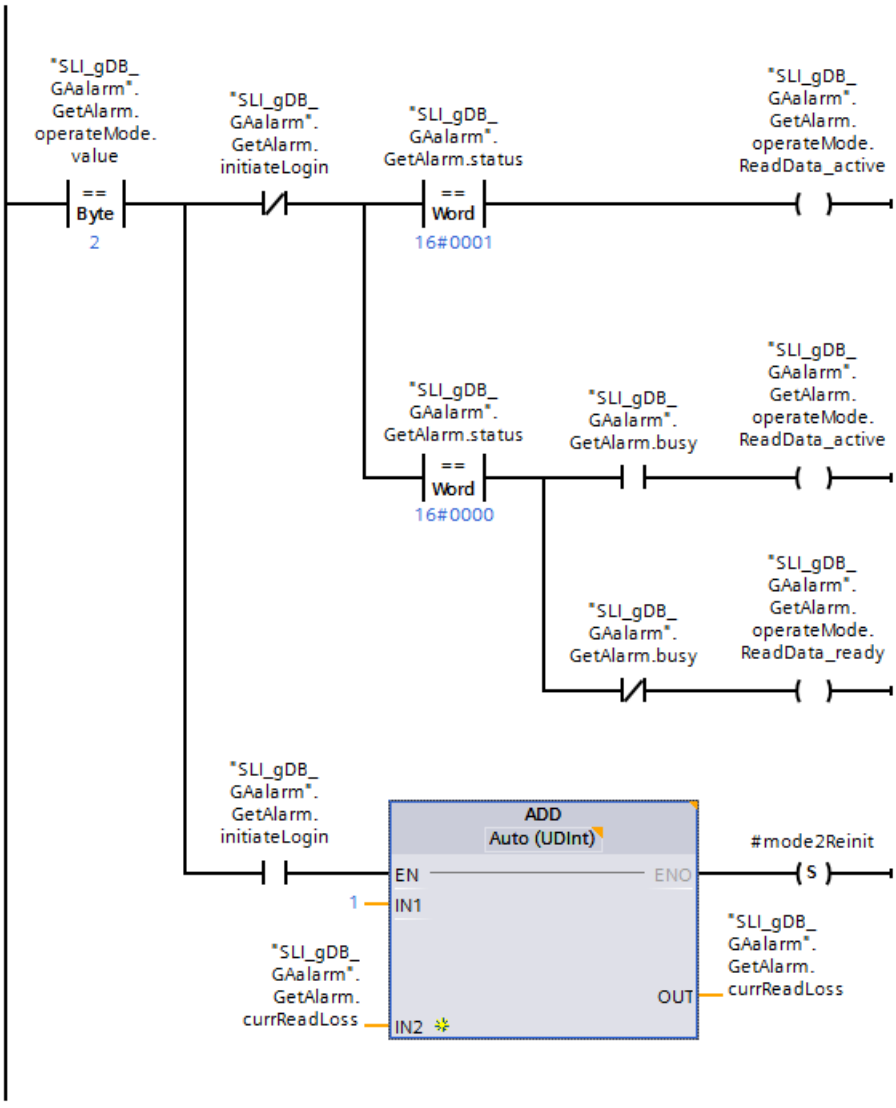

Network 6: To process the "Read" mode of Get_Alarm, create the following interconnections (Part 2).

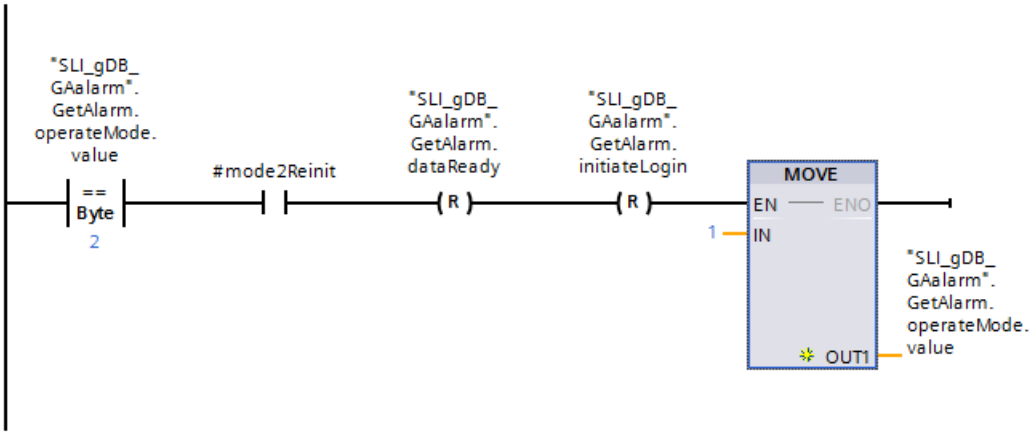

Network 7: To end the processing of Get_Alarm, create the following interconnections.

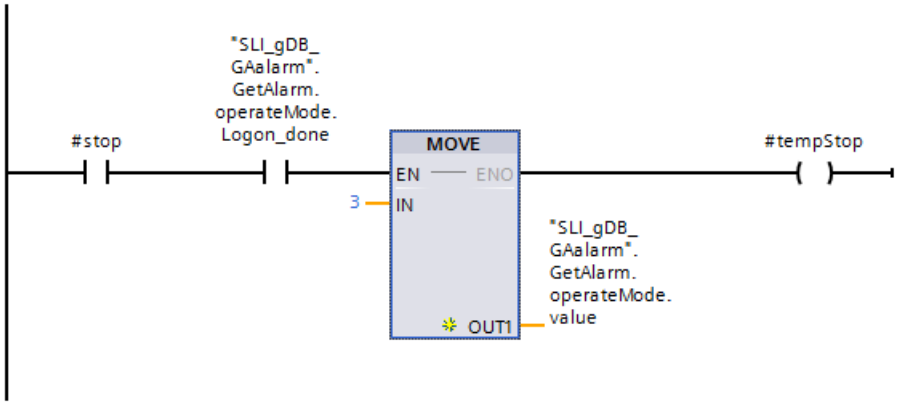

Network 8: To end the processing of Get_Alarm, create the following interconnections (Part 2).

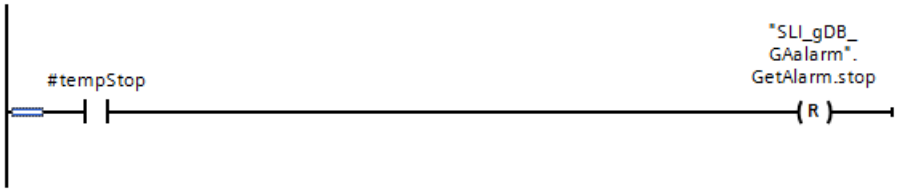

Network 9: To end the processing of Get_Alarm, create the following interconnections (Part 3).

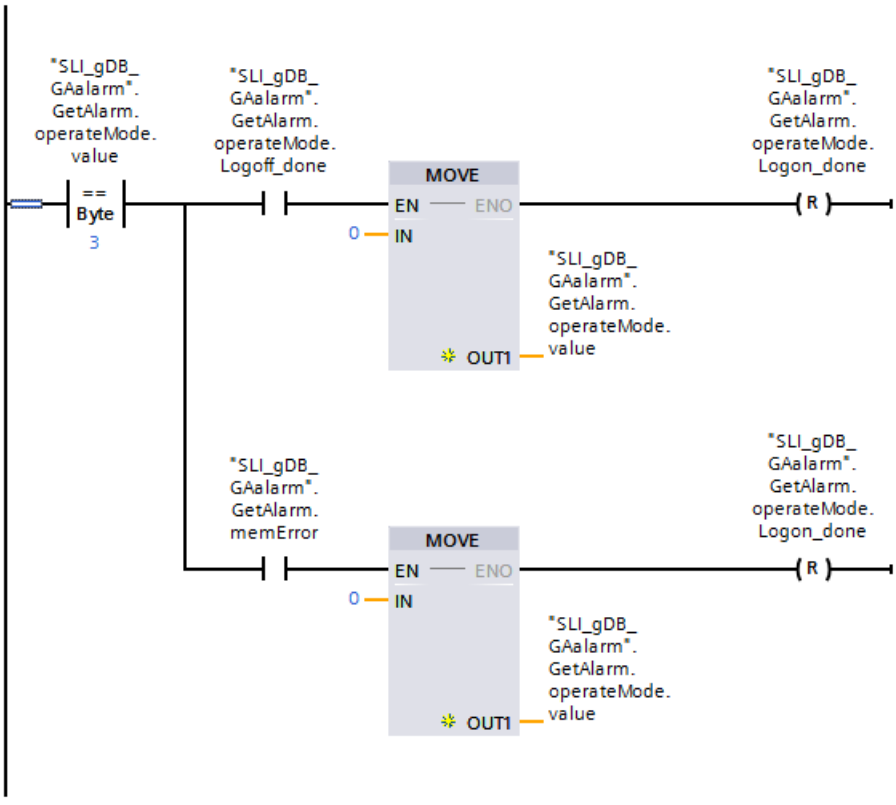

CPU 1: FB "SLI_FB_saveAlarms_GAA" - Interconnect parameters

You create the FB "SLI_FB_saveAlarms_GAA" to copy the alarms to the memory array "GetAlarm.alarmStorage".

Create the following local tags in the FB.

7	Static			
8	posForAlarm	UInt	0	Set in IDB
9	do	Bool	false	Set in IDB
10	do2	Bool	false	Set in IDB
11	nothing	Bool	false	Set in IDB
12	newRound	Bool	false	Set in IDB
13	dbComplete	Bool	false	Set in IDB
14	allDone	Bool	false	Set in IDB

Create the following interconnections in the FB.

Network 1: The copying of alarms to the memory array "GetAlarm.alarmStorage" is decided as follows:

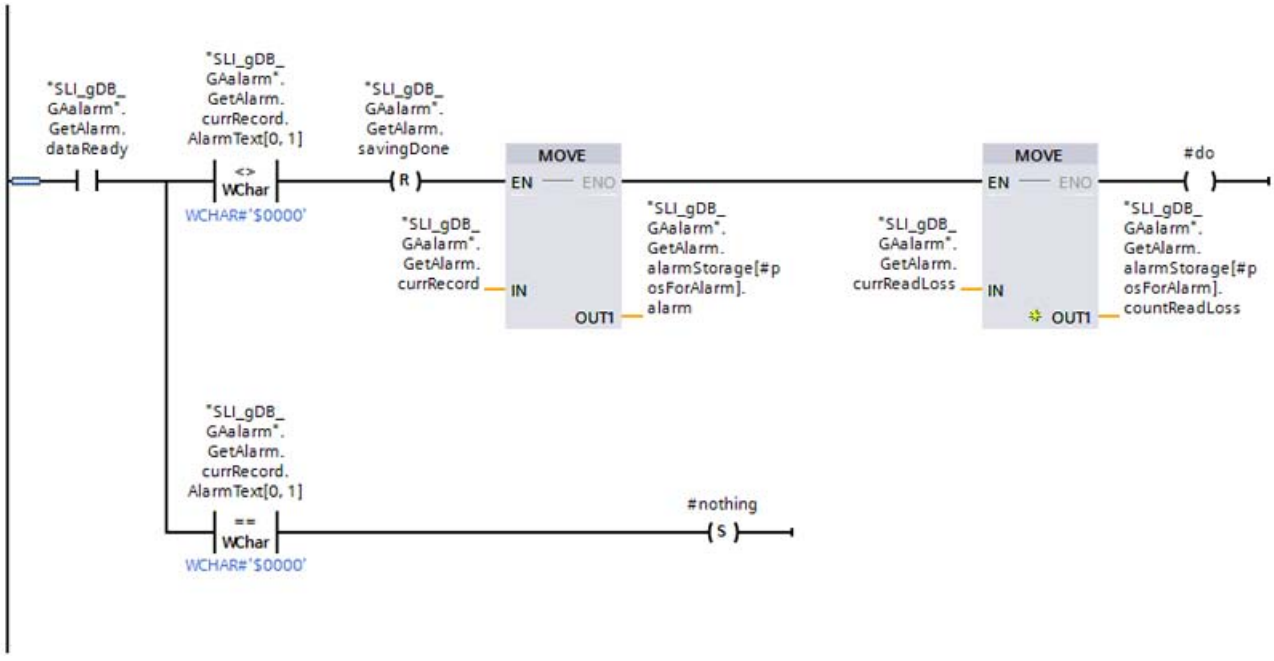

Network 2: To save the cell number of the memory array used last and to count the alarms, create the following interconnections.

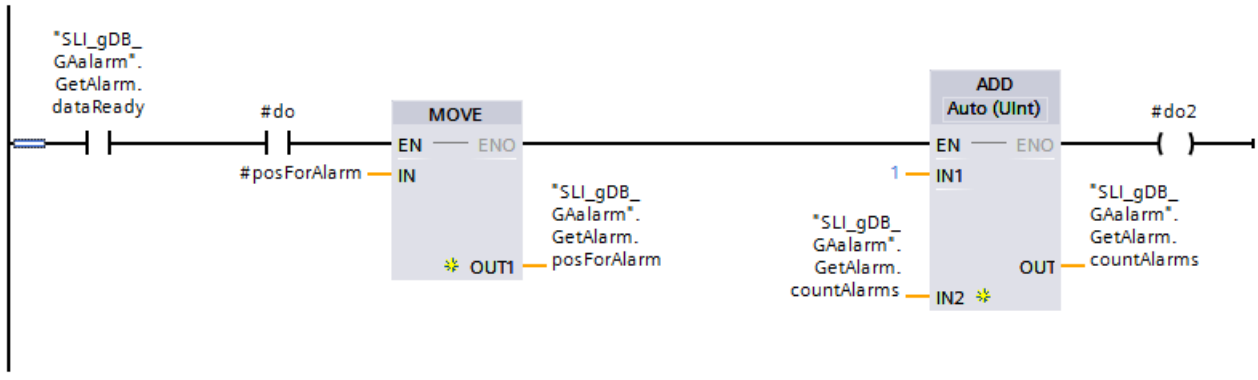

Network 3: Incrementing or resetting the cell number of the memory array is decided as follows:

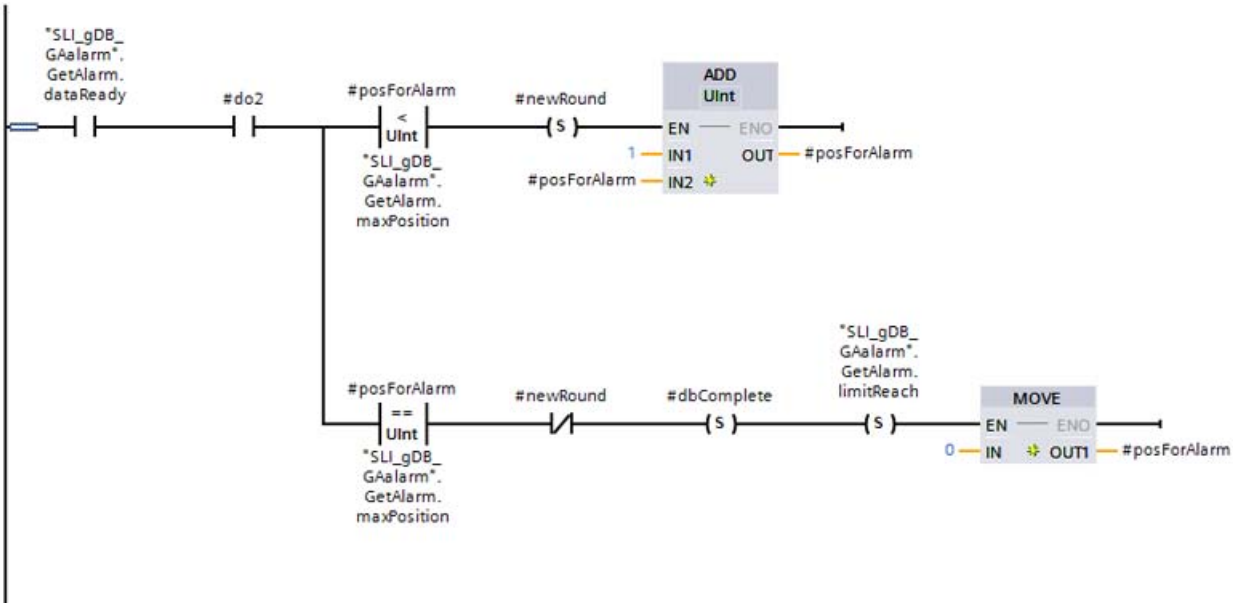

Network 4: To save the success when copying to the memory array, create the following interconnections.

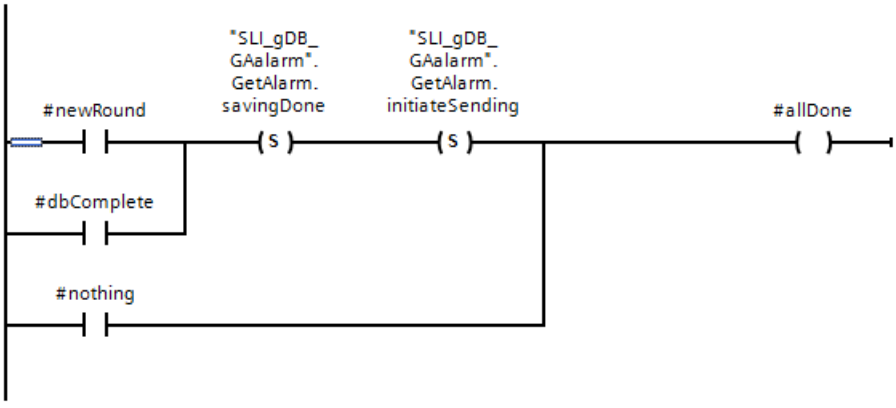

Network 5: To end the processing of SLI_FB_saveAlarms_GAA, create the following interconnections.

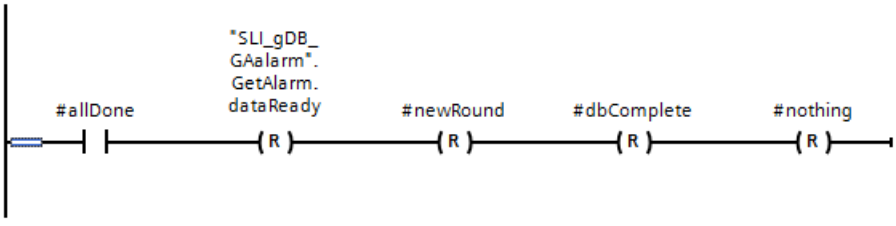

CPU 1: FB "SLI_FBCall_GetAlarm_AckA" - Interconnect parameters

You create the FB "SLI_FBCall_GetAlarm_AckA" for the following purposes:

- To call and modify Get_Alarm
- To call the blocks for copying and send processes.
- To call Ack_Alarms.

Create the following local tags in the FB.

7	Static			
8	acknowledge	UInt	1	Non-ret...
9	dataReady	Bool	false	Non-retain
10	initiate	Bool	false	Non-retain
11	error	Bool	false	Non-retain
12	SLI_FB_send_GAA_Instance	"SLI_FB_send_GAA"		

Create the following interconnections in the FB.

Network 1: Interconnect the parameters of the "SLI_FB_send_GAA" function block as follows.

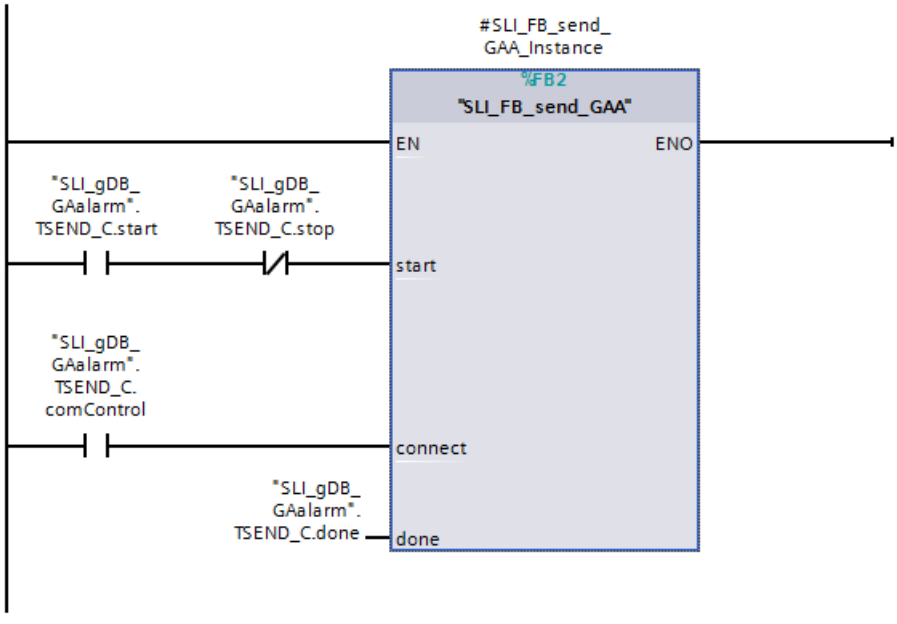

Network 2: Interconnect the parameters of the "SLI_FC_switchMode_GAA" FC as follows.

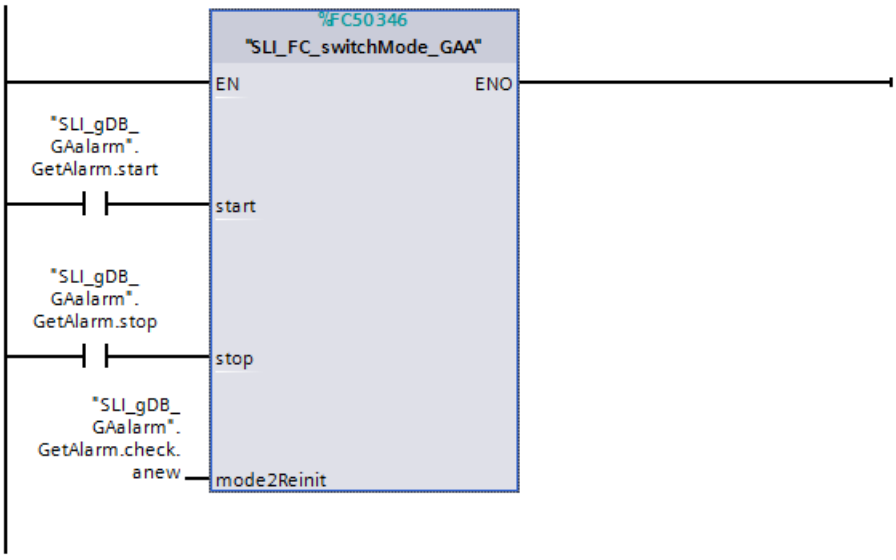

Network 3: Interconnect the parameters of the "Get_Alarm" instruction as follows.

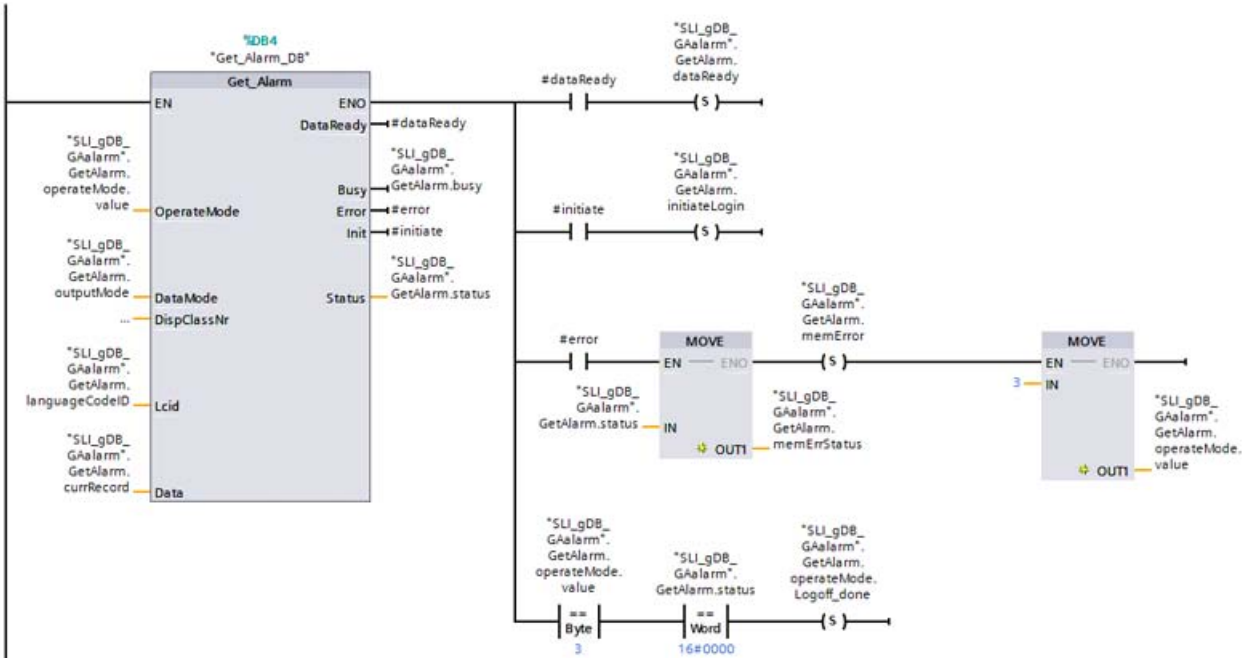

Network 4: Interconnect the parameters of the "SLI_FB_saveAlarms_GAA" function block as follows.

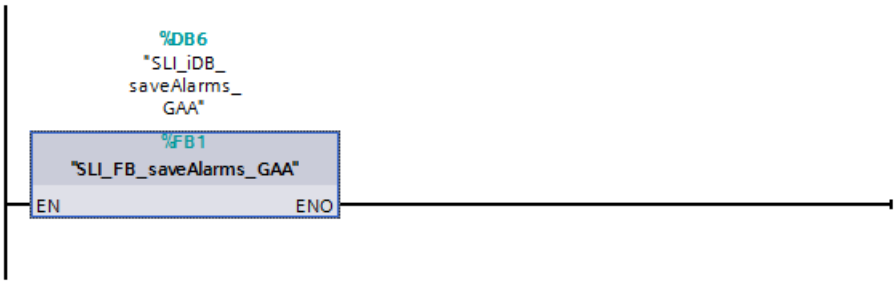

Network 5: Create the following interconnections to test the ProDiag alarms.

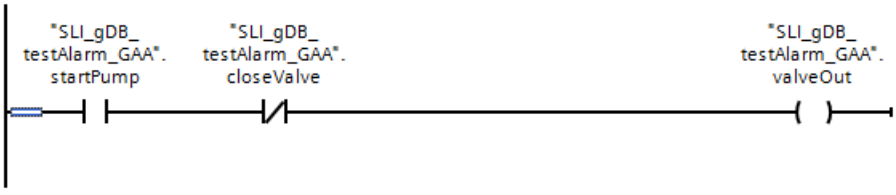

Network 6: Interconnect the parameters of the "Ack_Alarms" instruction as follows.

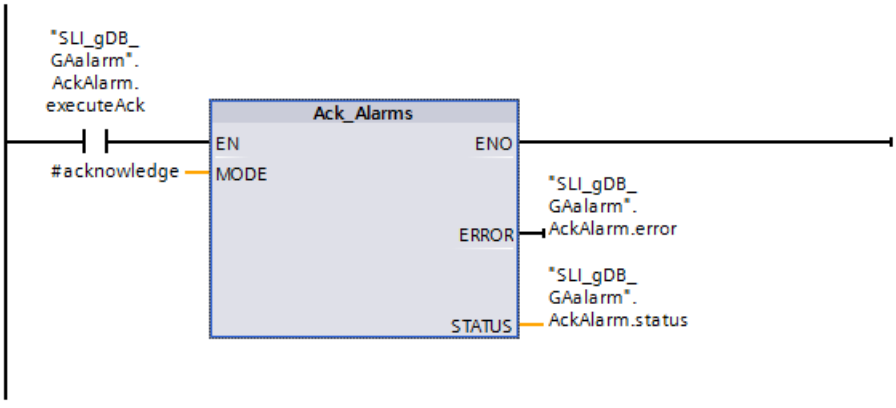

CPU 2: FB "SLI_FB_rcvdAlarms_GAA" - Interconnect parameters

You create the FB "SLI_FB_rcvdAlarms_GAA" to copy the alarms to the memory array "GetAlarm.rcvdStorage". The setup and interconnections are similar to those of the FB "SLI_FB_saveAlarms_GAA".

Additional information on setup and interconnections of the FB "SLI_FB_rcvdAlarms_GAA" is available in the program code of the example in the Sample Library for Instructions (Page 593).

CPU 2: FB "SLI_FBCall_trcv_GAA" - Interconnect parameters

You create the FB "SLI_FBCall_trcv_GAA" to call and process TRCV_C as well as all subsequent processes.

Create the following local tags in the FB.

7	Static			
8	TRCV_C_Instance	TRCV_C		
9	trcvStruct	"rcvGetAlarm"		Non-retain

Create the following interconnections in the FB.

Network 1: Interconnect the parameters of the "TRCV_C" instruction as follows. Configure the CONNECT parameter of TRCV_C using the TRCV_C wizard.

Additional information on the configuration of TRCV_C (Page 3497) is available here: Program example for send functions (Page 3506).

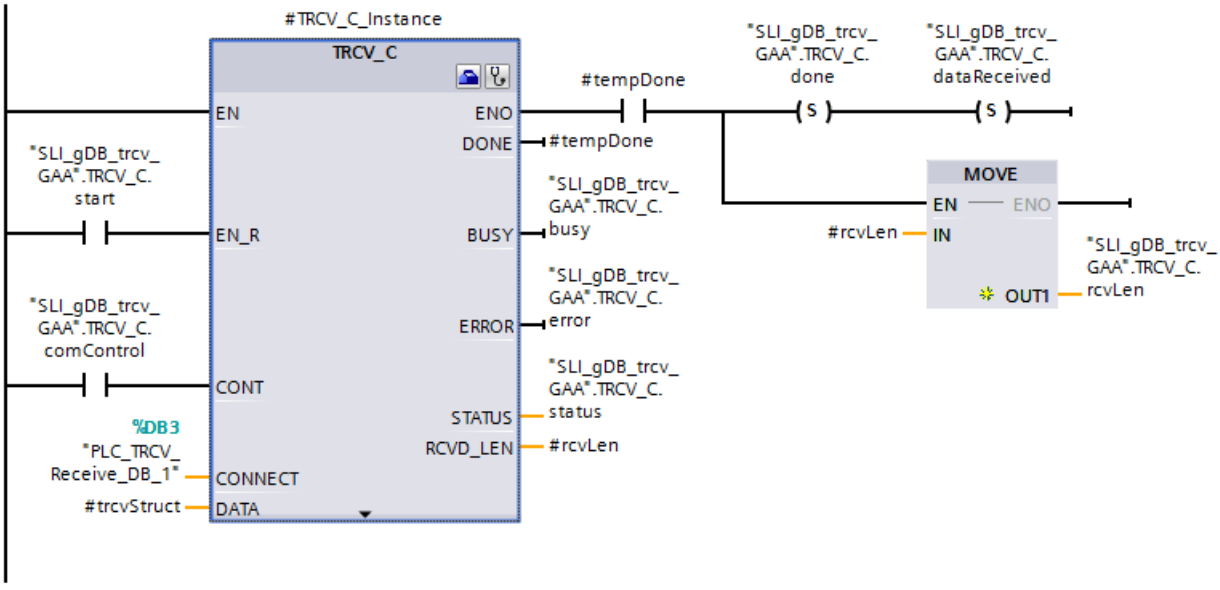

Network 2: To save the status if a TRCV_C error occurs, interconnect the parameter as follows.

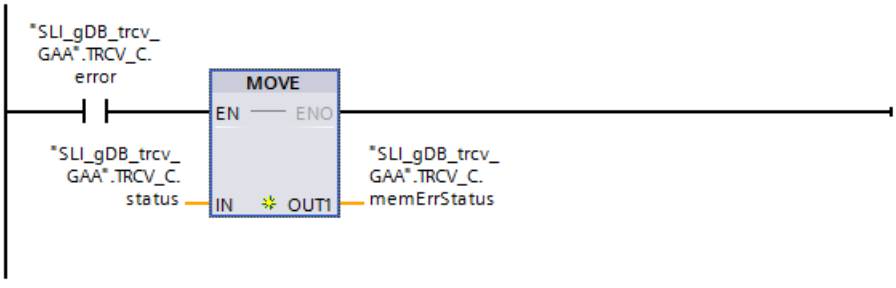

Network 3: Interconnect the parameters of the "SLI_FB_rcvdAlarms_GAA" function block as follows.

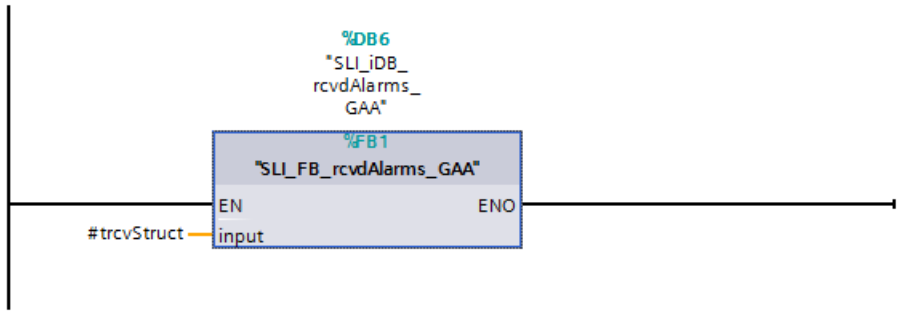

Results and procedure

The results and the procedure for the communication connection are described here: Program example for Get_Alarm & Ack_Alarms - Part 2 (Page 2562).

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Ack_Alarms: Acknowledge alarms (Page 2543)

Program example for Get_Alarm & Ack_Alarms - Part 2

Introduction

In the second part of the program example for Get_Alarm and Ack_Alarms, you will become familiar with the results of the configuration and the ProDiag alarm.

The requirements and the interconnection of the parameters are available here: Program example for Get_Alarm & Ack_Alarms - Part 1 (Page 2545).

Procedure for connection setup between CPU 1 and CPU 2

Observe the order of the action steps:

1. For CPU 2, change the tag "TRCV_C.start" to "TRUE".
2. For CPU 2, change the tag "TRCV_C.comControl" to "TRUE".
3. For CPU 1, change the tag "TSEND_C.start" to "TRUE".
4. For CPU 1, change the tag "TSEND_C.comControl" to "TRUE".

Procedure for disconnection of CPU 1 and CPU 2

Observe the order of the action steps:

1. For CPU 1, change the tag "TSEND_C.stop" to "TRUE".
2. For CPU 2, change the tag "TRCV_C.start" to "FALSE".
3. For CPU 2, change the tag "TRCV_C.comControl" to "FALSE".
4. For CPU 1, change the tag "TSEND_C.start" to "FALSE" and the tag "TSEND_C.stop" to "FALSE".
5. For CPU 1, change the tag "TSEND_C.comControl" to "FALSE".

Result of CPU 1: FB "SLI_FBCall_GetAlarm_AckA" - Network 1

If the normally open contact ("TSEND_C.start") supplies the signal state "TRUE", the "TSEND_C" instruction is started. When the input parameter CONNECT ("TSEND_C.comControl") returns the signal state "TRUE", the "TSEND_C" instruction creates a communication connection between CPU 1 and CPU 2.

Successful transfer of the data record (#sendStruct) is indicated by the output parameter DONE ("TSEND_C.done") with the signal state "TRUE" ("TSEND_C.status has the value "0000"). After transmitting the data record (#sendStruct) to CPU 2, the communication connection continues to be monitored (status "7004").

If the normally open contact ("TSEND_C.stop") supplies the signal state "TRUE", the "TSEND_C" instruction is stopped.

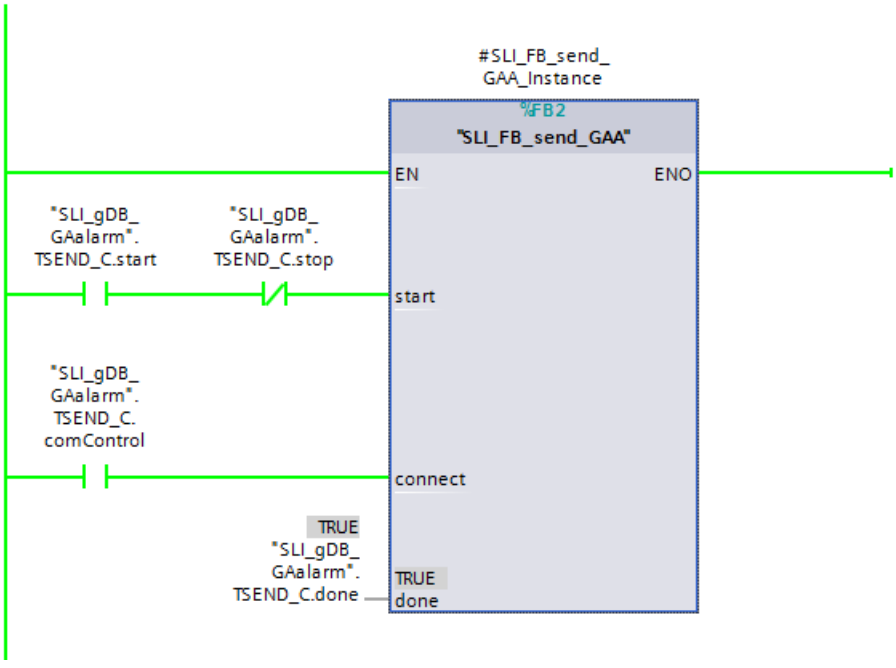

Additional information on the operating principle of TSEND_C (Page 3482) and TRCV_C (Page 3497) is available here: Program example for send functions (Page 3506).

Result of CPU 1: FB "SLI_FBCall_GetAlarm_AckA" - Network 2

If the normally open contact ("GetAlarm.start") supplies the signal state "TRUE", the "Get_Alarm" instruction is started. For this purpose, the "Get_Alarm" instruction is called in mode 1 ("GetAlarm.operateMode.value") and then changes to mode 2. After it has been processed, the tag "GetAlarm.start" is automatically reset to "FALSE".

If the normally open contact ("GetAlarm.stop") supplies the signal state "TRUE", the "Get_Alarm" instruction is stopped. For this purpose, the "Get_Alarm" instruction is called in mode 3 ("GetAlarm.operateMode.value") and then changes to inactivity (mode 0). A requirement for the stop command is successful logon of Get_Alarm ("GetAlarm.operateMode.Logon_done" is "TRUE"). After it has been processed, the tag "GetAlarm.stop" is automatically reset to "FALSE".

Note

Mode 0

When the value "0" is used at the parameter OPERATEMODE ("GetAlarm.operateMode.value") the instruction "Get_Alarm" outputs a parameter assignment error as the status. The reason: The instruction "Get_Alarm" does not know a mode 0. However, the value "0" at the parameter OPERATEMODE is necessary in this example to hold the instruction "Get_Alarm" in a state of complete inactivity. Otherwise the instruction "Get_Alarm" would have to permanently pass through one of the modes (log on or log off).

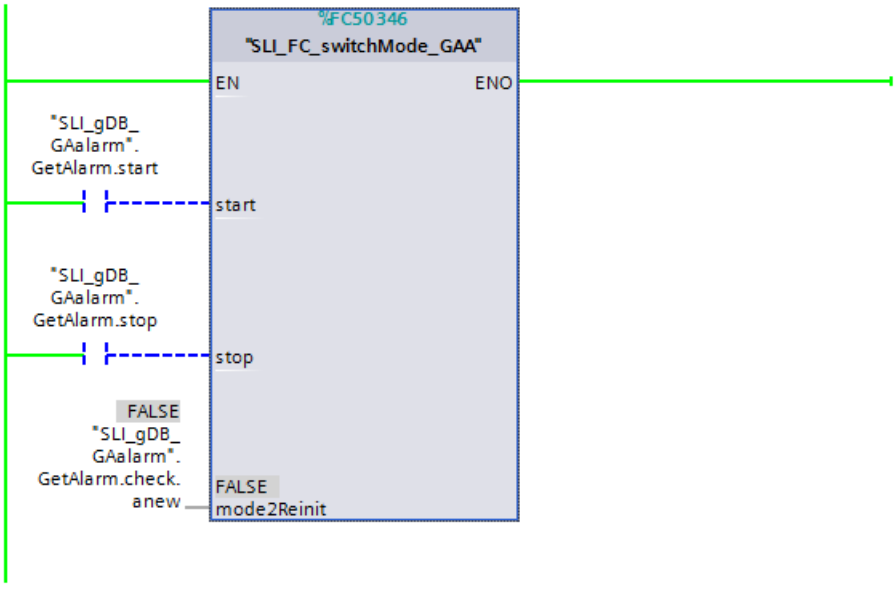

The success state of the processed mode of Get_Alarm is recorded:

- "GetAlarm.operateMode.Logon_done" is "TRUE" = mode 1 was completed successfully.
- "GetAlarm.operateMode.ReadData_ready" is "TRUE" = mode 2 is set. Get_Alarm is ready to read.
- "GetAlarm.operateMode.ReadData_active" is "TRUE" = Get_Alarm is in mode 2 and busy reading the data.
- "GetAlarm.operateMode.Logoff_done" is "TRUE" = mode 3 was completed successfully.

Note

Unintentional log off of Get_Alarm

When Get_Alarm is logged off in mode 2, the output parameter INIT ("GetAlarm.initiateLogin") returns the signal state "TRUE". This event is counted as read loss ("GetAlarm.currReadLoss") in FC "SLI_FC_switchMode_GAA" and the (INOUT) parameter MODE2REINIT ("GetAlarm.check.anew") of "SLI_FC_switchMode_GAA" is set to "TRUE". This results in a reset of the tags "GetAlarm.dataReady" and "GetAlarm.initiateLogin" and the "Get_Alarm" instruction is automatically restarted in mode 1. Afterwards Get_Alarm is once again ready to read in mode 2.

18		currReadLoss	UDInt	0	0
19		dataReady	Bool	false	FALSE
20		initiateLogin	Bool	false	FALSE
21		busy	Bool	false	FALSE
22		status	Word	16#0	16#0000
23		memError	Bool	false	FALSE
24		memErrStatus	Word	16#0	16#0000

Result of CPU 1: FB "SLI_FBCall_GetAlarm_AckA" - Network 3

When the input parameter OPERATEMODE ("GetAlarm.operateMode.value") shows the value "2", the "Get_Alarm" instruction reads all the alarms of CPU 1. For this purpose, the respective alarm is recorded at the input parameter DATA ("GetAlarm.currRecord").

SLI_gDB_GAalarm

	Name	Data type	Start value	Monitor value
1	Static			
2	TSEND_C	Struct		
3	GetAlarm	Struct		
4	start	Bool	false	FALSE
5	stop	Bool	false	FALSE
6	check	Struct		
7	anew	Bool	false	FALSE
8	operateMode	Struct		
9	value	Byte	16#0	16#02
10	Logon_done	Bool	false	TRUE
11	ReadData_ready	Bool	false	TRUE
12	ReadData_active	Bool	false	FALSE
13	Logoff_done	Bool	false	FALSE
14	outputMode	Byte	16#3	16#03
15	languageCodeID	UDInt	0	0
16	currRecord	AlarmData		

The signal state "TRUE" at the output parameter DATAREADY ("#dataReady") indicates a new alarm. Because the value of "#dataReady" (as well as that of "#initiate") is only set for one cycle, the status is saved with the tag "GetAlarm.dataReady".

When the output parameter INIT ("#initiate") returns the signal state "TRUE" in mode 2, Get_Alarm was logged off and must be logged on again. See the note on network 2. In mode 1, "#initiate" / "GetAlarm.initiateLogin" shows the successful logon.

In case of an error, the status of Get_Alarm ("GetAlarm.status") is saved with the tags "GetAlarm.memErrStatus" and "GetAlarm.memErr", and Get_Alarm is set to mode 3 to log off. When Get_Alarm is to be logged off and no error occurs ("GetAlarm.status" has the value "16#0000"), logoff is saved through "GetAlarm.operateMode.Logoff_done" with the value "TRUE".

3	GetAlarm	Struct		
4	start	Bool	false	FALSE
5	stop	Bool	false	FALSE
6	check	Struct		
7	anew	Bool	false	FALSE
8	operateMode	Struct		
9	value	Byte	16#0	16#00
10	Logon_done	Bool	false	FALSE
11	ReadData_ready	Bool	false	FALSE
12	ReadData_active	Bool	false	FALSE
13	Logoff_done	Bool	false	TRUE
14	outputMode	Byte	16#3	16#03

The input parameter DATAMODE ("GetAlarm.outputMode") with the value "16#03" specifies that the alarms are read out completely (with all information). The input parameter DISPCLASSNR is not connected. This results in Get_Alarm reading all the display classes. The input parameter LCID ("GetAlarm.languageCodeID") with the value "0" specifies that all language versions of alarms should be read.

Result of CPU 1: FB "SLI_FBCall_GetAlarm_AckA" - Network 4

Under the following conditions, the currently read alarm ("GetAlarm.currRecord") as well as the read losses counted until then ("GetAlarm.currReadLoss") are saved as a combined data record in the memory array "GetAlarm.alarmStorage":

- "GetAlarm.dataReady" returns the signal state "TRUE". This means Get_Alarm has read a new alarm.
- In addition, the data record of the read alarm ("GetAlarm.currRecord") is not empty. Its alarm text is checked for this reason.

26	alarmStorage	Array[0..15] of "SG...		
27	alarmStorage[0]	"SGetAlarm"		
28	countReadL...	UDInt	0	0
29	alarm	AlarmData		
30	Produce...	UInt	0	9
31	ID_1	UInt	0	53
32	ID_2	UInt	0	1
33	PRIO	UInt	0	0
34	TimeSta...	LDT	LDT#1970-01-01-(	LDT#2016-06-22-0...
35	State	Byte	16#0	16#01
36	LCID	Array[0..2] of UInt		
37	LCID...	UInt	0	0
38	LCID...	UInt	0	0
39	LCID...	UInt	0	0
40	AlarmText	Array[0..2] of WStri...		
41	Alar...	WString[254]	WSTRING#''	WSTRING#'Warning ...

Next, the position used ("#posForAlarm") is saved in the memory array "GetAlarm.alarmStorage" in the global tag "GetAlarm.posForAlarm". In addition, the alarms are counted and the current value is recorded with the global tag "GetAlarm.countAlarms".

The position information ("#posForAlarm") is incremented so that a new alarm can be saved in a different cell in the memory array. A requirement for incrementing is that the limit of the memory array ("GetAlarm.maxPositon") is not exceeded. When the limit is exceeded, this fact is noted ("GetAlarm.limitReach") and the position information ("#posForAlarm") is set to the value "0". Any subsequent alarms would therefore overwrite old alarm entries in the memory array with this setting.

Successful copying is noted in the tag "GetAlarm.savingDone". The "GetAlarm.initiateSending" tag is set to initiate subsequent sending of the alarm.

At the end of processing of "SLI_FB_saveAlarms_GAA", the tag "GetAlarm.dataReady" (and local tags) is reset to "FALSE". The program of "SLI_FB_saveAlarms_GAA" is only processed again when "GetAlarm.dataReady" returns the value "TRUE".

25			maxPosition	UInt	15	15
26			alarmStorage	Array[0..15] of "SG...		
27			limitReach	Bool	false	FALSE
28			savingDone	Bool	false	TRUE
29			initiateSending	Bool	false	FALSE
30			countAlarms	UInt	0	1
31			posForAlarm	UInt	0	0

Result of CPU 1: FB "SLI_FBCall_GetAlarm_AckA" - Network 5

When the normally open contact ("startPump") and the normally closed contact ("closeValve") return the signal state "TRUE", the configured ProDiag alarm is triggered.

Get_Alarm reads out the alarm and saves it in the assigned tag ("GetAlarm.currRecord"). In network 4, the alarm is subsequently copied to the memory array ("GetAlarm.alarmStorage").

17	currRecord	AlarmData		
18	ProducerID	UInt	0	9
19	ID_1	UInt	0	53
20	ID_2	UInt	0	1
21	PRIO	UInt	0	0
22	TimeStamp	LDT	LDT#1970-01-01-(	LDT#2016-06-22-0...
23	State	Byte	16#0	16#01
24	LCID	Array[0..2] of UInt		
25	LCID[0]	UInt	0	0
26	LCID[1]	UInt	0	0
27	LCID[2]	UInt	0	0
28	AlarmText	Array[0..2] of WStri...		
29	AlarmText[0]	WString[254]	WSTRING#''	WSTRING#'Warning ...
30	AlarmText[1]	WString[254]	WSTRING#''	WSTRING#''
31	AlarmText[2]	WString[254]	WSTRING#''	WSTRING#''

Result of CPU 1: FB "SLI_FBCall_GetAlarm_AckA" - Network 1 (data transmission)

When the tag "GetAlarm.initiateSending" returns the signal state "TRUE", the saved position ("GetAlarm.posForAlarm") is transferred in the memory array of a local tag with the same name ("#posForAlarm") and the command is forwarded for a new data transfer. The "#initiateSending" tag is set to "TRUE".

If, in addition, no error is signaled and TSEND_C is signaled as busy ("TSEND_C.error" and "TSEND_C.busy" are "FALSE"), the respective alarm ("GetAlarm.alarmStorage[..]") is transferred in abbreviated form to the structure "#sendStruct". If no stop command ("TSEND_C.stop" to "TRUE") was given, the tag "TSEND_C.start" is set to "TRUE".

The start command for data transmission ("TSEND_C.start" is "TRUE") sends the abbreviated alarm ("#sendStruct") to CPU 2 over multiple cycles.

Result of CPU 1: FB "SLI_FBCall_GetAlarm_AckA" - Network 6

If the normally open contact ("AckAlarm.executeAck") supplies the signal state "TRUE", the "Ack_Alarms" instruction is started. According to the value "1" of the input parameter MODE ("#acknowledge"), Ack_Alarms acknowledges all active alarms. Processing in the example takes place without errors ("AckAlarm.error" is "FALSE" and "AckAlarm.status" has the value "16#0000").

To view the confirmations, use an HMI device or the Web server of the CPU.

Result of CPU 2: FB "SLI_FBCall_trcv_GAA" - Network 1

If the normally open contact ("TRCV_C.start") supplies the signal state "TRUE", the "TRCV_C" instruction is started. When the input parameter CONNECT ("TRCV_C.comControl") supplies the signal state "TRUE", the "TRCV_C" instruction creates a communication connection between CPU 2 and CPU 1.

Successful transfer of the data record (#trcvStruct) is indicated by the output parameter DONE ("TRCV_C.done" and "TRCV_C.dataReceived") with the signal state "TRUE" ("TRCV_C.status has the value "0000"). After transmitting the data record (#trcvStruct) to CPU 2, the communication connection continues to be monitored and TRCV_C is ready to receive (status

"7006"). The length in Byte of the actually transmitted data record is detected with the output parameter RCVD_LEN ("TRCV_C.rcvLen").

SLI_gDB_trcv_GAA				
	Name	Data type	Start value	Monitor value
1	Static			
2	TRCV_C	Struct		
3	start	Bool	false	TRUE
4	comControl	Bool	false	TRUE
5	done	Bool	false	TRUE
6	busy	Bool	false	TRUE
7	error	Bool	false	FALSE
8	status	Word	16#0	16#7006
9	memErrStatus	Word	16#0	16#0000
10	rcvLen	UDInt	0	540
11	dataReceived	Bool	false	FALSE
12	GetAlarm	Struct		
13	maxPosition	UInt	15	15
14	rcvdStorage	Array[0..15] of "rcv...		
15	limitReach	Bool	false	FALSE
16	savingDone	Bool	false	FALSE
17	countAlarms	UInt	0	0
18	posForAlarm	UInt	0	0

Additional information on the operating principle of TSEND_C (Page 3482) and TRCV_C (Page 3497) is available here: Program example for send functions (Page 3506).

Result of CPU 2: FB "SLI_FBCall_trcv_GAA" - Network 2

If the normally open contact ("TRCV_C.error") returns the signal state "TRUE", the status ("TRCV_C.status") of the error is saved ("TRCV_C.memErrStatus").

Result of CPU 2: FB "SLI_FBCall_trcv_GAA" - Network 3

An alarm is saved in the memory array "GetAlarm.rcvdStorage" under the following conditions:

- "TRCV_C.dataReceived" returns the signal state "TRUE". This means a new data record was sent to TRCV_C.
- In addition, the new received data record (#trcvStruct) is not empty.

12	GetAlarm	Struct		
13	maxPosition	UInt	15	15
14	rcvdStorage	Array[0..15] of "rcv...		
15	rcvdStorage[0]	"rcvGetAlarm"		
16	countReadL...	UDInt	0	0
17	ProducerID	UInt	0	9
18	ID_1	UInt	0	53
19	ID_2	UInt	0	1
20	PRIO	UInt	0	0
21	TimeStamp	LDT	LDT#1970-01-01-(	LDT#2016-06-22-0...
22	State	Byte	16#0	16#01
23	LCID	Array[0..2] of UInt		
24	AlarmText	WString[254]	WSTRING#''	WSTRING#'Warning ...

Next, the position used ("#posForAlarm") is saved in the memory array "GetAlarm.rcvdStorage" in the global tag "GetAlarm.posForAlarm". In addition, the alarms are counted and the current value is recorded with the global tag "GetAlarm.countAlarms".

The position information ("#posForAlarm") is incremented so that a new alarm can be saved in a different cell in the memory array. A requirement for incrementing is that the limit of the memory array ("GetAlarm.maxPositon") is not exceeded. When the limit is exceeded, this fact is noted ("GetAlarm.limitReach") and the position information ("#posForAlarm") is set to the value "0". Any subsequent alarms would therefore overwrite old alarm entries in the memory array with this setting.

Successful copying is noted in the tag "GetAlarm.savingDone".

At the end of processing of "SLI_FB_rcvdAlarms_GAA", the tag "TRCV_C.dataReceived" (and local tags) is reset to "FALSE". The program of "SLI_FB_rcvdAlarms_GAA" is only processed again when "TRCV_C.dataReceived" returns the value "TRUE".

SLI_gDB_trcv_GAA

	Name	Data type	Start value	Monitor value
1	Static			
2	TRCV_C	Struct		
3	start	Bool	false	TRUE
4	comControl	Bool	false	TRUE
5	done	Bool	false	TRUE
6	busy	Bool	false	TRUE
7	error	Bool	false	FALSE
8	status	Word	16#0	16#7006
9	memErrStatus	Word	16#0	16#0000
10	rcvLen	UDInt	0	540
11	dataReceived	Bool	false	FALSE
12	GetAlarm	Struct		
13	maxPosition	UInt	15	15
14	rcvdStorage	Array[0..15] of "rcv...		
15	limitReach	Bool	false	FALSE
16	savingDone	Bool	false	TRUE
17	countAlarms	UInt	0	1
18	posForAlarm	UInt	0	0

Part 1 of the program example

The requirements and the interconnection of the parameters are available here: Program example for Get_Alarm & Ack_Alarms - Part 1 (Page 2545).

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

4.1.4.9 Diagnostics

RD_SINFO: Read current OB start information (S7-1200, S7-1500)

Description

You use the instruction "RD_SINFO" to read the start information

- of the last OB called that has not yet been completely executed, and
- of the last startup OB started.

There is no time stamp in either case. If the call is in OB 100, OB 101 or OB 102, two identical start information messages will be returned.

Parameter

The following table shows the parameters of the "RD_SINFO" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Error information
TOP_SI	Output	VARIANT	D, L	Start information of the current OB
START_UP_SI	Output	VARIANT	D, L	Start information of the startup OB last started

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

SDTs of the TOP_SI parameter

The following table shows the possible SDTs for the TOP_SI parameter:

Organization blocks (OB)	System data types (SDT)	System data type numbers
Any	SI_classic	592
	SI_none	593
ProgramCycleOB	SI_ProgramCycle	594
TimeOfDayOB	SI_TimeOfDay	595
TimeDelayOB	SI_Delay	596
CyclicOB	SI_Cyclic	597
ProcessEventOB	SI_HWInterrupt	598
	SI_HWInterrupt_Extended	616
ProfileEventOB StatusEventOB UpdateEventOB	SI_Submodule	601
SynchronousCycleOB	SI_SynchCycle	602
IOredundancyErrorOB	SI_IORedundancyError	604
CPUredundancyErrorOB	SI_CPURedundancyError	605
TimeErrorOB	SI_TimeError	606

Organization blocks (OB)	System data types (SDT)	System data type numbers
DiagnosticErrorOB	SI_DiagnosticInterrupt	607
PullPlugEventOB	SI_PlugPullModule	608
PeripheralAccessErrorOB	SI_AccessError	609
RackStationFailureOB	SI_StationFailure	610
ServoOB	SI_Servo	611
IpoOB	SI_Ipo	612
StartupOB	SI_Startup	613
ProgrammingErrorOB IOaccessErrorOB	SI_ProgIOAccessError	614
CPURedundancyErrorOB	SI_CPURedundancyError_V2	617

SDTs of the START_UP_SI parameter

The following table shows the possible SDTs for the START_UP_SI parameter:

System data types (SDT)	System data type numbers
SI_classic	592
SI_none	593
SI_Startup	613

Structures

The following tables set out the meaning of the structure elements of the individual structures:

Table 4-36 SI_classic structure

Structure element	Data type	Description
EV_CLASS	BYTE	• Bits 0 to 3: Event ID • Bits 4 to 7: Event class
EV_NUM	BYTE	Event number
PRIORITY	BYTE	Priority class number (Meaning of B#16#FE: OB not available or disabled or cannot be started in current operating mode)
NUM	BYTE	OB number
TYP2_3	BYTE	Data ID 2_3: Identifies the information entered in ZI2_3
TYP1	BYTE	Data ID 1: Identifies the information entered in ZI1
ZI1	WORD	Additional information 1
ZI2_3	DWORD	Additional information 2_3

Table 4-37 SI_none structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)

Table 4-38 SI_ProgramCycle structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 1	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
Remanence	BOOL	For OB_Class = 1

Table 4-39 SI_TimeOfDay structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 10	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
CaughtUp	BOOL	For OB_Class = 10
SecondTime	BOOL	For OB_Class = 10

Table 4-40 SI_Delay structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 20	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Sign	WORD	For OB_Class = 20

Table 4-41 SI_Cyclic structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 30	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92

Table 4-42 SI_HWInterrupt structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 40	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
USI	WORD	For OB_Class = 40
IChannel	USINT	For OB_Class = 40
EventType	BYTE	For OB_Class = 40

Table 4-43 Structure SI_HWInterrupt_Extended

Structure element	Data type	Byte	Description
SI_Format	USINT	0	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 40	1	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	2	OB number (1 ... 32767)
LADDR	HW_IO	4	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
USI	WORD	6	For OB_Class = 40
IChannel	USINT	8	For OB_Class = 40
EventType	BYTE	9	For OB_Class = 40
PointAddr	DWORD	12	For OB_Class = 40

Table 4-44 SI_Submodule structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92

Structure element	Data type	Description
Slot	UINT	For OB_Class = 55, 56, 57
Specifier	WORD	For OB_Class = 55, 56, 57

Table 4-45 SI_SynchCycle structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 61	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
PIP_Input	BOOL	For OB_Class = 61, 91, 92
PIP_Output	BOOL	For OB_Class = 61, 91, 92
IO_System	USINT	For OB_Class = 61, 91, 92
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92
SyncCycleTime	LTIME	Calculated cycle time

Table 4-46 SI_IORedundancyError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 70	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_ANY	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86

Table 4-47 SI_CPURedundancyError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 72	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Switch_Over	BOOL	For OB_Class = 72

Table 4-48 SI_TimeError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 80	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86
Csg_OBnr	OB_ANY	For OB_Class = 80
Csg_Prio	UINT	For OB_Class = 80

Table 4-49 SI_DiagnosticInterrupt structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 82	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
IO_State	WORD	For OB_Class = 82
LADDR	HW_ANY	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Channel	UINT	For OB_Class = 82
MultiError	BOOL	For OB_Class = 82

Table 4-50 SI_PlugPullModule structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 83	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86

Table 4-51 SI_AccessError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 85	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86

Structure element	Data type	Description
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86
IO_Addr	UINT	For OB_Class = 85
IO_LEN	UINT	For OB_Class = 85

Table 4-52 SI_StationFailure structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 86	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86

Table 4-53 SI_Servo structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 91	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
PIP_Input	BOOL	For OB_Class = 61, 91, 92
PIP_Output	BOOL	For OB_Class = 61, 91, 92
IO_System	USINT	For OB_Class = 61, 91, 92
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92
Synchronous	BOOL	

Table 4-54 SI_Ipo structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 92	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
PIP_Input	BOOL	For OB_Class = 61, 91, 92
PIP_Output	BOOL	For OB_Class = 61, 91, 92
IO_System	USINT	For OB_Class = 61, 91, 92

Structure element	Data type	Description
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92
Reduction	UINT	For OB_Class = 92

Table 4-55 SI_Startup structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 100	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LostRetentive	BOOL	For OB_Class = 100
LostRTC	BOOL	For OB_Class = 100

Table 4-56 SI_ProgIOAcessError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
BlockNr	UINT	For OB_Class = 121, 122
Reaction	USINT	For OB_Class = 121, 122
Fault_ID	BYTE	For OB_Class = 121, 122
BlockType	USINT	For OB_Class = 121, 122
Area	USINT	For OB_Class = 121, 122
DBNr	DB_ANY	For OB_Class = 121, 122
Csg_OBNr	OB_ANY	For OB_Class = 121, 122
Csg_Prio	USINT	For OB_Class = 121, 122
Width	USINT	For OB_Class = 121, 122

Table 4-57 SI_CPURedundancyError_V2 structure

Structural element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 72	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_ANY	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86

Note

If this was created with the block property "Standard, the structure elements specified for the SI_classic structure are identical in content to the temporary tags of an OB.

Note, however, that temporary tags of the individual OBs can have different names and different data types. Also note that the call interface of each OB includes additional information regarding the date and the time of the OB request.

Bits 4 to 7 of the EV_CLASS structure element contain the event class. The following values are possible here:

- 1: Start events from standard OBs
- 2: Start events from synchronous error OBs
- 3: Start events from asynchronous error OBs

The PRIORITY structure element supplies the priority class belonging to the current OB.

Apart from these two elements, NUM is also relevant. NUM contains the number of the current OB or the startup OB that was started last.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
8081	Start information of the current OB does not correspond to the specified system data type
8083	Start information of the last startup OB started does not correspond to the specified system data type

Example

OB 80 is the OB that was called last and that has not yet been completely processed and OB 100 is the start-up OB that was started last.

The following table shows the assignment between the structure elements of the TOP_SI parameter of the "RD_SINFO" instruction and the associated local tags of OB 80.

TOP_SI structure element	Data type	OB 80 - Associated local tag	Data type
EV_CLASS	BYTE	OB80_EV_CLASS	BYTE
EV_NUM	BYTE	OB80_FLT_ID	BYTE
PRIORITY	BYTE	OB80_PRIORITY	BYTE
NUM	BYTE	OB80_OB_NUMBR	BYTE
TYP2_3	BYTE	OB80_RESERVED_1	BYTE
TYP1	BYTE	OB80_RESERVED_2	BYTE
ZI1	WORD	OB80_ERROR_INFO	WORD

TOP_SI structure element	Data type	OB 80 - Associated local tag	Data type
ZI2_3	DWORD	OB80_ERR_EV_CLASS	BYTE
		OB80_ERR_EV_NUM	BYTE
		OB80_OB_PRIORITY	BYTE
		OB80_OB_NUM	BYTE

The following table shows the assignment between the structure elements of the START_UP_SI parameter of the "RD_SINFO" instruction and the associated local tags of OB 100.

START_UP_SI structure element	Data type	OB 100 - Local tag	Data type
EV_CLASS	BYTE	OB100_EV_CLASS	BYTE
EV_NUM	BYTE	OB100_STRTUP	BYTE
PRIORITY	BYTE	OB100_PRIORITY	BYTE
NUM	BYTE	OB100_OB_NUMBR	BYTE
TYP2_3	BYTE	OB100_RESERVED_1	BYTE
TYP1	BYTE	OB100_RESERVED_2	BYTE
ZI1	WORD	OB100_STOP	WORD
ZI2_3	DWORD	OB100_STRT_INFO	DWORD

Example

In the following example, you read out the start information of the program cycle OB that was last called and the startup OB of the CPU that was last started.

Create three tags in a global data block for storing the data.

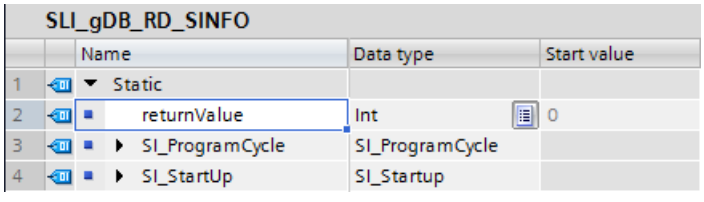

Call the "RD_SINFO" instruction in an OB1. Interconnect the parameters of the "RD_SINFO" instruction as follows.

Use only one OB for the CPU, for the example.

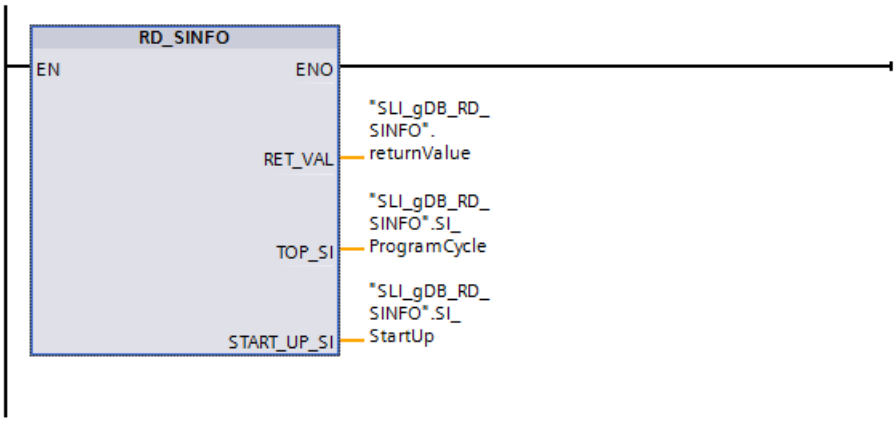

When the input parameter EN returns the signal state "TRUE", the "RD_SINFO" instruction is started. The instruction "RD_SINFO" reads the start information of the last called program cycle OB (OB1) and outputs this at the output parameter TOP_SI ("SI_ProgrammCycle"). Start information of the startup OB that was
last started is output at the output parameter START_UP_SI ("SI_Startup").

According to the value of the parameter "SI_Format", OB1 uses optimized start information. In addition, the class of the OB ("OB_Class") and the assigned number of the OB ("OB_Nr") are read out. No number ("OB_Nr" with the value "0") is read out for the startup OB ("SI_Startup"), because no startup OB is used by the CPU. In addition, standard values ("SI_Format", "OB_Class") are displayed for the startup OB.

The output parameter RET_VAL ("returnValue") indicates that the processing of the example is taking place without errors (the value is "0").

SLI_gDB_RD_SINFO

	Name	Data type	Start value	Monitor value
1	Static			
2	returnValue	Int	0	0
3	SI_ProgramCycle	SI_ProgramCycle		
4	SI_Format	USInt	0	254
5	OB_Class	USInt	1	1
6	OB_Nr	UInt	0	1
7	Initial_Call	Bool	false	FALSE
8	Remanence	Bool	false	TRUE
9	SI_StartUp	SI_Startup		
10	SI_Format	USInt	0	254
11	OB_Class	USInt	100	100
12	OB_Nr	UInt	0	0
13	LostRetentive	Bool	false	FALSE
14	LostRTC	Bool	false	FALSE

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Evaluating errors with output parameter RET_VAL (Page 162)

RT_INFO: Read out runtime statistics (S7-1500)

Description

Use the "RT_INFO" instruction to generate statistics on the runtime of specific organization blocks, communication or the user program.

Use the MODE parameter to select the information to be output:

- MODE 1 to 3 return the data on the runtime of a specific organization block whose number was specified at the OB parameter.
- MODE 10 and 11 measure over a time period of 1 s and return the mean value of the percentage of total runtime used by higher-priority OBs (MODE 10) or mean value of the percentage of total runtime used by communication (MODE 11).
- MODE 20 and 21 return the percentages of the runtime allotted to communication and the user program for the last program cycle.
- MODE 23 to 25 output the shortest, longest, and current cycle time of the user program.
- MODE 30 to 32 return data on configured settings of the user program.
- MODE 50 to 55 return data on the SYNCUP system state of an S7-1500-R/H system

All measurements are started once again when the CPU transitions from startup to RUN.

The runtime of an OB is not available until the first processing of the OB has been completed. The cycle time is not available until the first cycle has been completed.
If you call RT_INFO before the first measurement has been completed, the preset value of the associated measurement is entered in INFO and the value 0 is assigned to RET_VAL. The following table lists the preset values for MODE = 1, 2, 3, 23, 24 and 25.

MODE	Preset value
1	0
2	0
3	0x FFFF FFFF FFFF FFFF
23	0
24	0x FFFF FFFF FFFF FFFF
25	Parameterized minimum cycle time

Parameters

The following table shows the parameters of the "RT_INFO" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	Use the MODE parameter to select the information you want to read out (see table "MODE parameter").
OB	Input	OB_ANY	I, Q, M, D, L or constant	Use the OB parameter to select the OB for which you want to read out information.

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Error information (see " RET_VAL parameter").
INFO	InOut	VARIANT	I, Q, M, D, L	Pointer to the area to which the read data is to be written. The data type required for INFO depends on the MODE parameter (see table "MODE parameter").

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter MODE

The following table shows which information is returned for the values at the MODE parameter.

MODE (decimal)	Description	Note	Value at OB parameter	Data type of INFO	Available as of CPU version
1	Runtime of a specific OB	The runtime of an OB is defined as that period in which the CPU processes the commands of this OB. The time required to process higher-priority OBs and communication tasks that interrupt the OB is therefore not included. If, therefore, you want to determine the complete period between the processing of the first and the last command of the OB - meaning including the processing of higher-priority OBs and of communication tasks that interrupt the OB - use the instruction "RUNTIME".	OB number	LTIME	S7-1500 V1.5
2	Maximum runtime of a specific OB		OB number	LTIME	S7-1500 V1.5
3	Minimum runtime of a specific OB		OB number	LTIME	S7-1500 V1.5
10	Mean value of the percentage of total runtime used by higher-priority OBs The runtime of all OBs used in the user program with a higher priority than cyclical program OBs is output (ProgramCycle). These usually include all OB types except startup OBs. The CPU determines which OBs can be used and their priority. This information is included in the basic chapters on programming.	Averaging takes place over a time period of 1 s.	Not relevant	UINT	S7-1500 V1.5

MODE (decimal)	Description	Note	Value at OB parameter	Data type of INFO	Available as of CPU version
11	Mean value of the percentage of total runtime used by communication The percentage of communication processes in the total runtime of the user program is output.	Averaging takes place over a time period of 1 s. The returned communication percentage can be greater than you have specified in the CPU parameter "Cycle load due to communication". This can be the case if you have specified a minimum cycle time and this is not fully required by the user program. The remaining time that is available can then be used again for communication processes.	Not relevant	UINT	S7-1500 V1.5
20	Identical to MODE 10 except that the evaluation is based on the last program cycle executed.	• If the cycle time is too high (> 1 second), the calculation is not performed. The instruction outputs the value 65535 (0xFFFF). • If the cycle time is too low (< 1 millisecond), the calculation cannot be performed. In this case, the last cycle with a duration of at least one millisecond is evaluated. By assigning a minimum cycle time in the CPU properties, you can prevent cycle times from falling below 1 ms. • For MODE 21: The returned communication percentage can be greater than you have specified in the CPU parameter "Cycle load due to communication". This can be the case if you have specified a minimum cycle time and this is not fully required by the user program. The remaining time that is available can then be used again for communication processes.	Not relevant	UINT	S7-1500 V1.7
21	Identical to MODE 11 except that the evaluation is based on the last program cycle executed.		Not relevant	UINT	S7-1500 V1.7
23	Longest cycle time Duration of the longest cycle since the last transition from STOP to RUN.	The times correspond to the "Cycle times measured" values in the "Cycle time" dialog of the TIA Portal. You open the dialog using Online & Diagnostics > Diagnostics > Cycle time.	Not relevant	LTIME	S7-1500 V1.7
24	Shortest cycle time Duration of the shortest cycle since the last transition from STOP to RUN.		Not relevant	LTIME	S7-1500 V1.7
25	Current/last cycle time Duration of the last cycle.		Not relevant	LTIME	S7-1500 V1.7

MODE (decimal)	Description	Note	Value at OB parameter	Data type of INFO	Available as of CPU version
30	Cycle monitoring time Maximum permitted duration of the CPU program. If the cycle time exceeds the cycle monitoring time, the CPU goes to STOP or calls a time error OB.	The times correspond to the "Configured cycle time" values in the "Cycle time" dialog of the TIA Portal. You open the dialog using Online & Diagnostics > Diagnostics > Cycle time.	Not relevant	LTIME	S7-1500 V1.5
31	Output of the configured minimum cycle time for the user program. If a minimum cycle time has been assigned in the CPU properties, the operating system delays the start of the new cycle until the minimum cycle time has been reached.		Not relevant	LTIME	S7-1500 V1.5
32	Output of the configured maximum communication load as a percentage	The specification of the percentage of the cycle load allotted to communication is made in the CPU properties under "Communication load".	Not relevant	UINT	S7-1500 V1.5
50	Number of static data to be transferred in the SYNCUP system state	-	Not relevant	LINT	S7-1500 V2.6
51	Number of static data already transferred in the SYNCUP system state	-	Not relevant	LINT	S7-1500 V2.6
52	Number of dynamic data to be transferred in the SYNCUP system state	-	Not relevant	LINT	S7-1500 V2.6
53	Number of dynamic data already transferred in the SYNCUP system state	-	Not relevant	LINT	S7-1500 V2.6
54	Maximum delay between primary CPU and backup CPU that the backup CPU can make up for	-	Not relevant	LTIME	S7-1500 V2.6
55	Actual delay between primary CPU and backup CPU in the catch-up phase of the backup CPU	-	Not relevant	LTIME	S7-1500 V2.6

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0	No error
8080	Value at the MODE parameter is not supported.

Error code* (W#16#...)	Explanation
8081	Organization block selected at OB parameter does not exist in the user program.
8092	Selected MODE parameter is not supported by this CPU version.
80C3	Insufficient resources. Try to call the instruction again at a later time.
8452	Data type at INFO parameter is incorrect. Check whether the correct data type was selected depending on the MODE parameter.

Example

In the following example, you read out the runtime of a cycle organization block.

Create four tags in a global data block for storing the data.

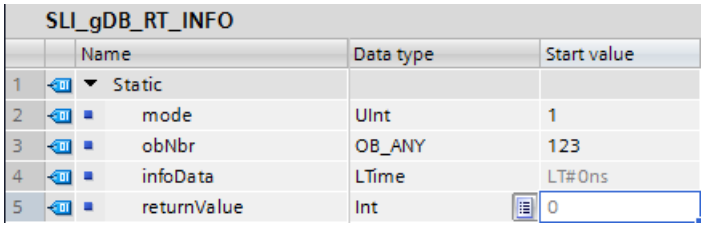

SLI_gDB_RT_INFO

		Name	Data type	Start value
1		Static		
2		mode	UInt	1
3		obNbr	OB_ANY	123
4		infoData	LTime	LT#0ns
5		returnValue	Int	0

Interconnect the parameters of the instruction as follows.

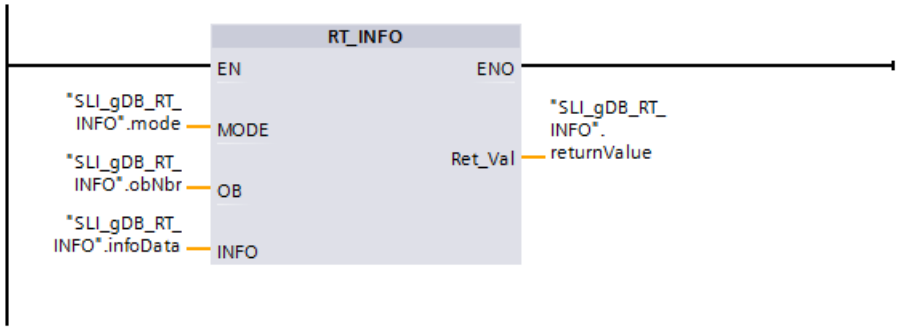

- Create a new block of type "Program cycle". Specify the OB number at the OB parameter.
- Enter "1" (read out runtime of a particular OB) at the MODE parameter.
- Specify a tag with the data type LTIME at the INFO parameter ("infoData" in this case).
- Specify a tag with the data type INT at the RET_VAL parameter to output the error messages of the instruction.

After calling the instruction, the currently measured runtime is written to the "infoData" tag.

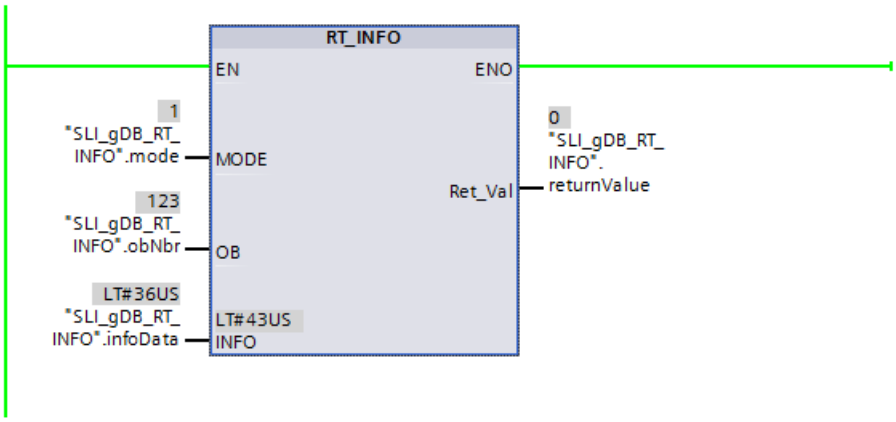

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Evaluating errors with output parameter RET_VAL (Page 162)

LED: Read LED status (S7-1200, S7-1500)

Description

You can use the "LED" instruction to read out the status (e.g., "On" or "Off") of a particular module LED.

- With the LADDR parameter, you address the CPU or the interface.
- With the LED parameter, you select the module LED whose current status is read out using the instruction.
- The RET_VAL parameter outputs the status of the selected LED when the instruction is called. Depending on the LED selected, only certain status information can be displayed. For example, some LEDs have only one color. Refer to the hardware documentation of the respective module for information on the possible statuses of a particular LED.

Parameters

The following table shows the parameters of the "LED" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Hardware identifier of the CPU or interface The number is assigned automatically and is stored in the properties of the CPU or interface in the hardware configuration (*CPU name* + *~Common*).
LED	Input	UINT	I, Q, M, D, L or constant	Identification number of the LED: • 1: STOP/RUN • 2: ERROR • 3: MAINT (maintenance) • 4: Redundant • 5: Link (green) • 6: Rx/Tx (yellow)
RET_VAL	Return	INT	I, Q, M, D, L	Status of LED

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

RET_VAL	Description
0 to 9	LED status: • 0 = LED does not exist or status information not available • 1 = Permanently switched off • 2 = Color 1 (e.g., for LED STOP/RUN: green) permanently ON • 3 = Color 2 (e.g., for LED STOP/RUN: orange) permanently ON • 4 = Color 1 flashing at 2 Hz • 5 = Color 2 flashing at 2 Hz • 6 = Colors 1 and 2 flashing alternately at 2 Hz • 7 = LED is active, color 1 • 8 = LED is active, color 2 • 9 = LED does not exist or status information not available For reasons of compatibility, ENO is set to FALSE for this value of RET_VAL.
8086	The R/H system is not in the RUN-Redundant system state. Therefore, the data of the partner CPU is not available.
8091	Hardware component addressed using the LADDR parameter is not available.
8092	A hardware component was addressed with the LADDR parameter that does not return the required information.
8093	The identification number specified at the LED parameter is not defined.
80Bx	The CPU specified in the LADDR parameter does not support the "LED" instruction.

Example

In the following example, you read out the status of the CPU LED.

Create three tags in a global data block for storing the data.

My_gDB_LED			
	Name	Data type	Start value
1	Static		
2	myLADDR	HW_IO	50
3	myLED	UInt	1
4	returnValue	Int	0

Interconnect the parameters of the instruction as follows.

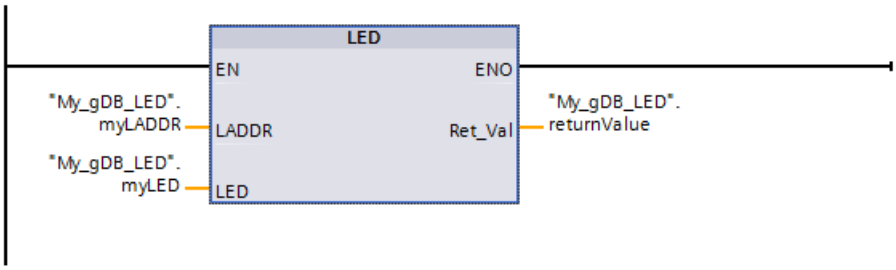

The HW identifier of the CPU is made known to the "LED" instruction through parameter LADDR ("myLADDR"). The CPU LED to be monitored is made known through parameter LED ("myLED"). The status of the CPU LED (STOP/RUN) is queried. If the CPU is placed in RUN operating mode from STOP, the value "6" (green and orange alternating) is indicated at output parameter RET_VAL ("returnValue"). Then, the value "2" (permanently green) is indicated as the LED status ("returnValue").

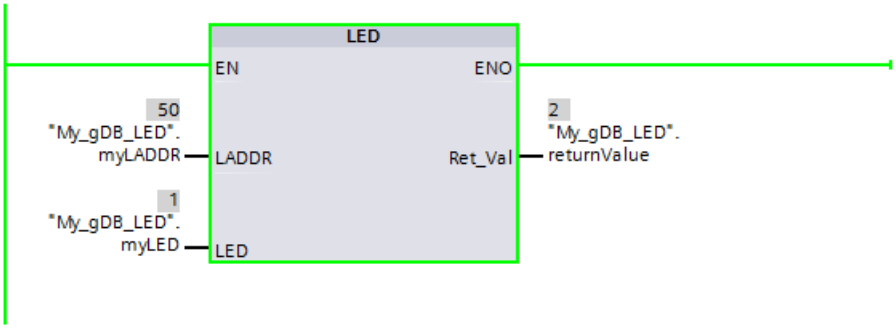

Note: The meaning of the colors for the STOP/RUN LED are as follows:

Color	Meaning
Red	STOP
Green	RUN
Alternating green and orange	The CPU is loading
Alternating red and green	Program processing is running

Additional information on the meaning of the LED colors can be found in the hardware description of the CPU.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Get_IM_Data: Reading identification and maintenance data (S7-1200, S7-1500)

Description

The "Get_IM_Data" instruction reads the identification and maintenance data (I&M) data from a device. Use the LADDR parameter to select the device from which you want to read the I&M via the hardware identifier.

The IM_TYPE parameter is used to select the data that is read out via the instruction:

- IM_TYPE = 0: I&M 0 data
  The I&M 0 data are the device-specific basic information of a device and contain information such as the manufacturer ID, order number, serial number and the hardware and firmware version. Only read access is possible to the I&M 0 data. The information is also displayed in the TIA Portal with the "Online & Diagnostics" view of a device.
- IM_TYPE = 11: I&M 1 data from the parameter assignment data of the CPU
  The I&M 1 data contain a functional description of the device and the location designation. i.e. the information about how the device is designated within the plant.
- IM_TYPE = 12: I&M 2 data from the parameter assignment data of the CPU
  The I&M 2 data includes the date of installation, i.e. the information about when the device was installed in the plant.
- IM_TYPE = 13: I&M 3 data from the parameter assignment data of the CPU
  The I&M 3 data contains additional information about the installed device. The additional information is free text and is assigned as desired.

The I&M data read is written to the addressed area defined at the DATA parameter.

The execution status of the read job is displayed via the BUSY, DONE, ERROR output parameters and the two middle bytes of the STATUS output parameter.

Definition: Identification and maintenance data (I&M)

Identification and maintenance (I&M) data refers to information stored in a module, which assists you in checking the plant configuration, locating hardware changes in a plant and eliminating errors.

- Identification data (I-data) is static information about a device that can only be read.
- Maintenance data (M-data) refers to plant-dependent information, such as the installation location or date. Maintenance data are created during configuration and written to the module.

Parameters

The following table shows the parameters of the "Get_IM_Data" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware ID of the device The number is assigned automatically and is stored in the properties of the device or in the hardware configuration.
IM_TYPE	Input	UINT	I, Q, M, D, L or constant	Identification and maintenance data number Possible values: • 0: I&M 0 data • 11: I&M 1 data from the parameter assignment data of the CPU • 12: I&M 2 data from the parameter assignment data of the CPU • 13: I&M 3 data from the parameter assignment data of the CPU
DATA	InOut	VARIANT	I, Q, M, D, L	Area for storing the read identification and maintenance data (see below).
DONE	Output	BOOL	I, Q, M, D, L	The instruction was executed successfully. The I&M data were transferred to the DATA parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Execution of the instruction completed or not yet started. • 1: Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display it, therefore, copy STATUS to a free data area.

You can find more information on the data types in "Overview of the valid data types (Page 231)".

Parameter DATA for I&M 0 data

Use an array (ARRAY of BYTE) or a special data structure to store the I&M 0 data:

- If you address an array (ARRAY of BYTE) at parameter DATA, the read I&M 0 data is copied as a byte sequence to DATA. If the addressed array is longer than the read data, the unneeded bytes are filled with zeros.
- The following "IM0_Data" structure can also be used at the DATA parameter for I&M 0 data:

Parameter		Data type	Bytes	Description
Manufacturer_ID		UINT	2	Manufacturer ID (e.g. "42" for SIEMENS)
Order_ID		CHAR[20]	20	Order number
Serial_Number		CHAR[16]	16	Serial number
Hardware_Revision		UINT	2	Hardware revision
Software_Revision		STRUCT	4	Firmware revision
	Type	CHAR	1	-
	Functional	USInt	1	-
	Bugfix	USInt	1	-
	Internal	USInt	1	-
Revision_Counter		UINT	2	Revision counter
Profile_ID		UINT	2	Profile
Profile_Specific_Type		UINT	2	Device class
IM_Version		WORD	2	I&M version
I&M_Supported		WORD	2	I&M data (I&M 0-I&M 4) supported at the device end

If a different data type is used at the DATA parameter, the STATUS parameter outputs error code 8093.

DATA parameter for I&M 1, I&M 2, and I&M 3 data

Use a string (STRING), an array (ARRAY of CHAR/BYTE) or a data structure (STRUCT) to store the I&M data:

- If you address a string (STRING data type) at the the DATA parameter, the length of the string automatically adjusts the length of the read I&M data (up to 254 characters).
- If you address a data structure (ARRAY of CHAR/BYTE or STRUCT) a the DATA parameter, the I&M data read are written to the individual components of the data type used. If the addressed data structure is longer than the read data, the remaining components are filled with zeros.
- If you crate a data structure with a STRUCT data type at the DATA parameter, use a data block without optimized block access (see category "Attributes" of the block properties).

If a data type other than STRING, ARRAY of BYTE/CHAR or STRUCT is used at the DATA parameter, the STATUS parameter generates error code 8093.

Note

Additional information about I&M data

You can find more information about I&M data, for example, on the PROFIBUS & PROFINET International websites (link: http://www.profibus.com (http://www.profibus.com)).

Parameter STATUS

Error code* (W#16#...)	Description
0	No error
7001	First call of the asynchronous instruction "Get_IM_Data". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
7002	Additional call of the asynchronous instruction "Get_IM_Data". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
8086	The R/H system is not in the RUN-Redundant system state. Therefore, the data of the partner CPU is not available.
8091	The device addressed at the LADDR parameter does not exist.
8092	LADDR addresses a device that does not support the output of the I&M data.
8093	Data type at the DATA parameter is not supported.
80A1	LADDR addresses a device that does not support the output of I&M 1, I&M 2 or I&M 3 data.
80B1	No valid data available. (Either the I&M data is not supported by the (sub)module or you have not configured I&M data for this device.)
80B2	Invalid value at the IM_TYPE parameter or the selected IM_TYPE is not supported by the CPU or the addressed device.
80C3	Maximum of 10 parallel executions of the instances exceeded: • Avoid more than 10 parallel calls. • Repeat the call at a later time, as this is a temporary error.
8752	The memory area specified at the DATA parameter is too small to store all the I&M data. The read I&M data are stored only up to the maximum length of the specified memory area.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you read out the IM0 data of an S7-1500 CPU. The IM0 data are the basic information of a device and contain information such as the manufacturer ID, order number, serial number and the hardware and firmware version.

Create four tags and a structure with the IM0_Data data type in a global data block for storing the IM0 data. You can assign any name to the structure ("MyPLC_IM_Data") in this case.

SLI_gDB_Get_IM_Data			
	Name	Data type	Start value
1	Static		
2	imData	IM0_Data	
3	done	Bool	false
4	busy	Bool	false
5	error	Bool	false
6	status	Word	16#0

Enter the hardware identifier of the CPU at the LADDR parameter. The hardware identifier uniquely identifies the product. To determine the hardware identifier of your CPU, open the PLC tag table and the System constants tab. Then, search for the CPU in the Name column. The associated value is the hardware identifier that you enter at the LADDR parameter.

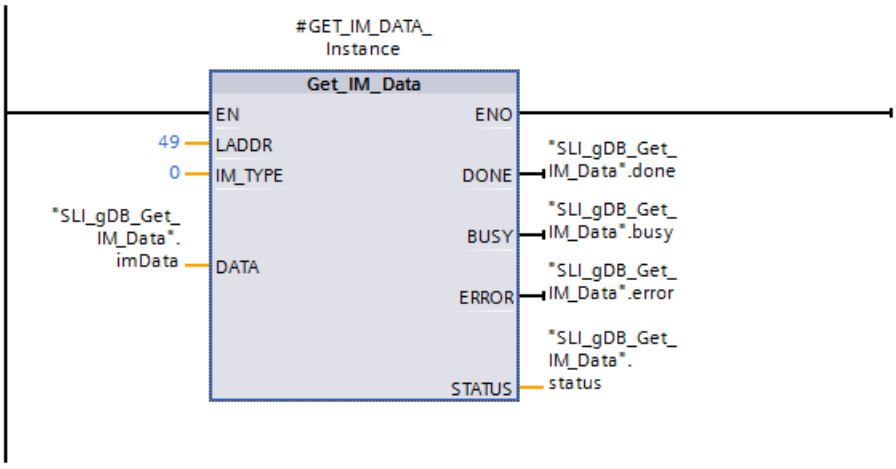

Once the instruction is successfully executed, the IM0 data is written to the data block.

SLI_gDB_Get_IM_Data				
	Name	Data type	Start value	Monitor value
1	Static			
2	imData	IM0_Data		
3	Manufacturer_ID	UInt	0	42
4	Order_ID	String[20]	''	'6ES7 511-1AK00-0...
5	Serial_Number	String[16]	''	'S C-DOS710132013'
6	Hardware_Revision	UInt	0	3
7	Software_Revision	IM0_Version		
8	Type	Char	' '	'V'
9	Functional	USInt	0	1
10	Bugfix	USInt	0	5
11	Internal	USInt	0	0
12	Revision_Counter	UInt	0	0
13	Profile_ID	UInt	0	0
14	Profile_Specific_Ty...	UInt	0	0
15	IM_Version	Word	16#0	16#0101
16	IM_Supported	Word	16#0	16#001E
17	done	Bool	false	TRUE
18	busy	Bool	false	FALSE
19	error	Bool	false	FALSE
20	status	Word	16#0	16#0000

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Difference between synchronous and asynchronous instructions (Page 596)

GET_NAME: Read out name of an IO device or a DP slave (S7-1200, S7-1500)

Description

The instruction "GET_NAME" reads out the name of an IO device or a DP slave. The name is displayed in the network view and in the properties of the IO device or the DP slave. The following screenshot shows this using an IO device as example.

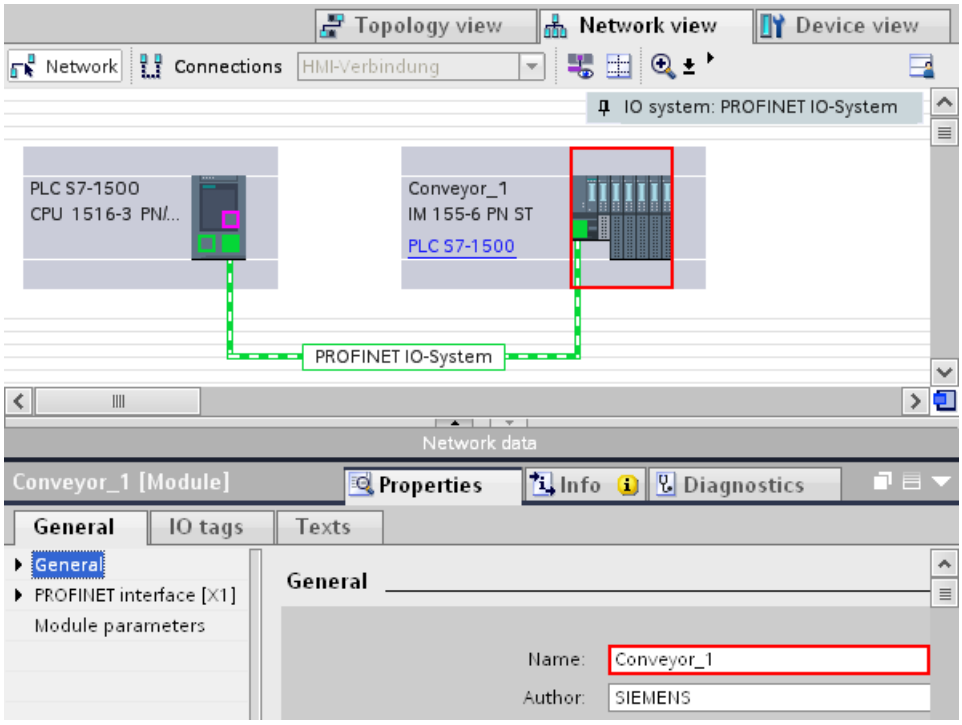

You select the IO device or the DP slave by using the hardware identifier of the PROFINET IO system or the DP master system (at the LADDR parameter) and the device number of the IO device or the PROFIBUS address of the DP slave (STATION_NR parameter).

After the instruction has been executed successfully, the name of the IO device or the DP slave is written to the area addressed with the DATA parameter.

The name that is read depends on the type of IO device:

- For a DP slave or IO device, the name of the head module is output.
- For an I-slave or I-device, the name of the interface module is output.
- The name of the interface is output for an HMI panel.
- The name of the interface module is output for a PC station.
- The name of the Device Access Point (DAP) is displayed (name of the interface or head module) for GSD devices.

The length of the name is output at the LEN parameter. If the name is longer than the area specified at the DATA parameter, only that section which corresponds to the maximum length of the addressed area will be written.

The maximum length for a name is 128 characters.

Note

Name of the CPU readout (Version 1.1)

If you assign a "0" at each of the parameters LADDR and STATION_NR the instruction will output the name of the CPU.

For the R/H system in the RUN-Redundant mode, the instruction returns the name of the primary CPU.

Parameters

The following table shows the parameters of the "GET_NAME" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IOSYSTEM	I, Q, M, D, L or constant	Hardware identifier of the PROFINET IO or DP master system. The number is taken from the system constants or the properties of the PROFINET IO system or DP master system.
STATION_NR	Input	UINT	I, Q, M, D, L or constant	Device number of the PROFINET IO device or PROFIBUS address of the DP slave. This number is displayed in the network view and in the properties of the IO device or the DP slave.
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the area to which the name is written.
DONE	Output	BOOL	I, Q, M, D, L	The instruction was executed successfully. The name of the IO device or the DP slave was transferred to the area at the DATA parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Execution of the instruction complete. • 1: Execution of the instruction not yet complete.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
LEN	Output	DINT	I, Q, M, D, L	Length of the name of the IO device or the DP slave (number of characters).
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, therefore, copy STATUS to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

Error code* (W#16#...)	Explanation
0	No error
7001	First call of the asynchronous instruction "GET_NAME". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
7002	Additional call of the asynchronous instruction "GET_NAME". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
8090	The hardware identifier specified at the LADDR parameter does not exist in the project.
8092	The value at the LADDR parameter does not address a PROFINET IO system or a DP master system.
8093	Instruction does not support data type at the DATA parameter.
8095	• Device number (STATION_NR parameter) does not exist in the selected PROFINET IO system or does not address an IO device. • PROFIBUS address (STATION_NR parameter) does not exist in the selected DP master system.
80B1	The CPU used does not support the instruction.
80C3	Temporary resource error: The CPU is currently processing the maximum possible number of simultaneous block calls. "GET_NAME" cannot be executed until at least one of the block calls is finished.
8852	The area specified at the DATA parameter is too short for the full name of the IO device or the DP slave. The name will be written up to the maximum possible length. To read the full name, use a longer data area at the DATA parameter. The area must have at least as many characters as there are at the LEN parameter.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

The following example shows how you can read out the station name of a HMI panel.

Configuration of the HMI panel:

- The HMI panel with the station name "HMI_IO-Device" is created in the network view and assigned to the same PROFINET IO system as the CPU.
- The operating mode "IO device" was activated in the properties of the hardware configuration for the HMI panel and the CPU was assigned as the IO controller.
- The device number "20" assigned in the properties under "Ethernet addresses".

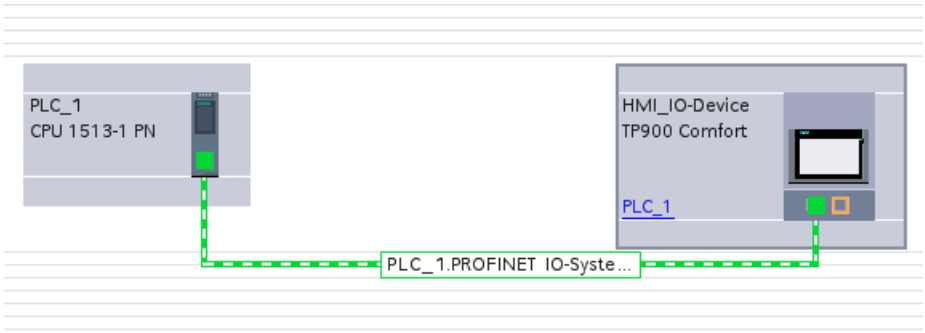

Create the following tags for storing the data in a global data block:

SLI_gDB_GET_NAME

		Name	Data type	Start value
1		Static		
2		moduleName	String	''
3		done	Bool	false
4		busy	Bool	false
5		error	Bool	false
6		lengthOfData	DInt	0
7		status	Word	16#0

Assigning parameters to the instruction "GET_NAME":

- Enter the hardware identifier of the IO system (262) at the input parameter LADDR.
- Enter the device number of the HMI panel (2) at the input parameter STATION_NR.
- Interconnect the remaining parameters of the "GET_NAME" instruction as follows.

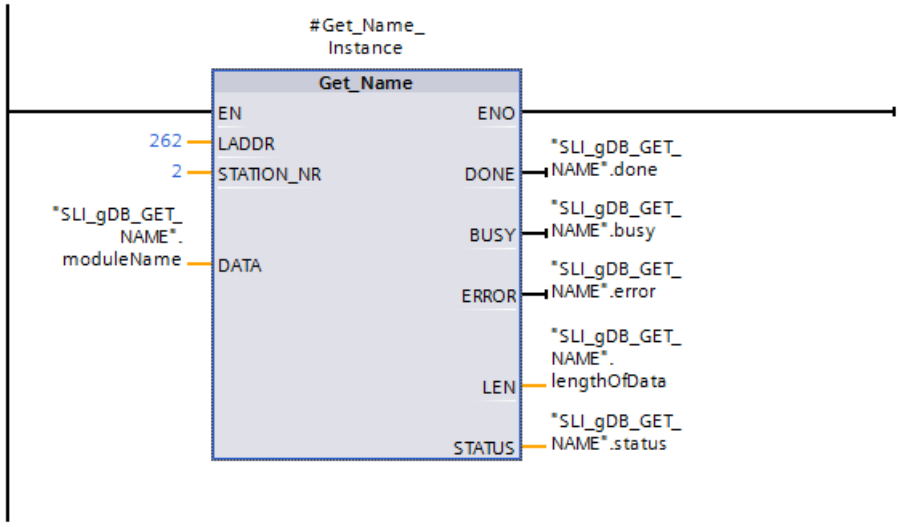

Execution of the instruction:

- After execution of the instruction, the name of the station of the HMI panel (HMI_IO-Device) is stored at parameter DATA ("moduleName").
- The number of characters of the name is output (13) at the parameter LEN.
- As GET_NAME was not stopped after the successful reading of the IO device, output parameter BUSY ("busy") shows the value "TRUE" and output parameter STATUS ("status") shows the value "16#7002".

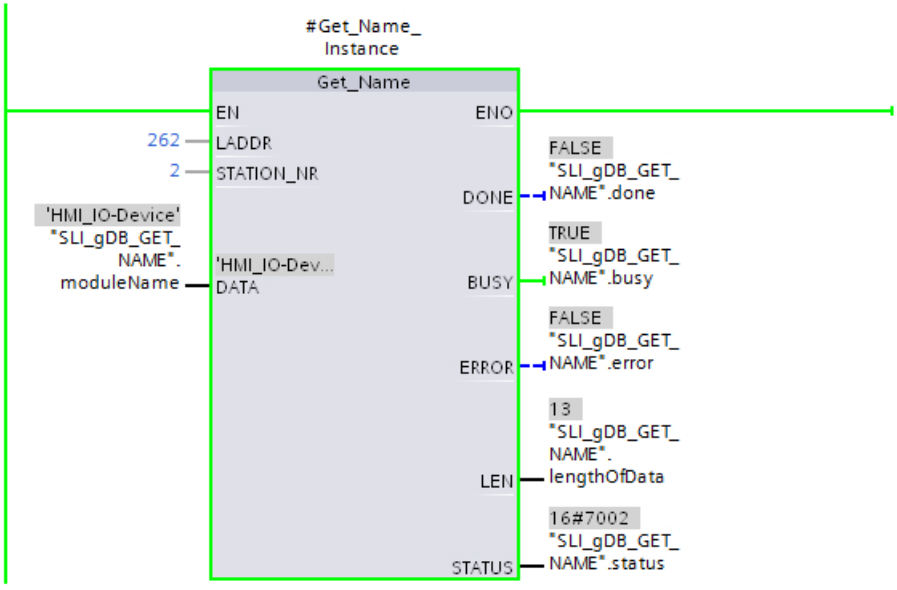

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

GetStationInfo: Read information of an IO device (S7-1200, S7-1500)

Description

You can use the "GetStationInfo" instruction to read information of a PROFINET IO device. The instruction also enables you to read the information of an IO device located in a lower-level IO system (connected via CP/CM).

The IO device is addressed via the hardware identifier of the station at the LADDR parameter. The hardware identifier is displayed in the "Devices & Networks" view in the station properties.

Use the MODE parameter to select the information to be read.

At the DATA parameter, specify the data area to which the read address data will be written. For storing the IP address, use the "IF_CONF_v4" structure. For storing the MAC address, use the "IF_CONF_MAC" structure.

Enable reading of the address data using the REQ control parameter. This requires the IO device to be accessible.

The execution status of the read job is displayed via the BUSY, DONE, ERROR output parameters and the STATUS output parameter.

Note

Address the IO device using only the hardware identifier of the station

The station, the IO device and PROFINET interface have their own hardware identifier. Use only the hardware identifier of the station for the "GetStationInfo" instruction.

If a PROFINET interface is addressed via the LADDR parameter, for example, the address data is not read and the 8092 error code is generated.

To read the address data of an integrated PROFINET interface or a CM/CP in the central configuration, use the "RDREC" instruction.

Parameters

The following table shows the parameters of the "GetStationInfo" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter Request Activates the reading of the information with REQ = "1".
LADDR	Input	HW_DEVICE	I, Q, M, D, L or constant	Hardware identifier of the station of the IO device The number is taken from the properties of the station in the Network view or from the "System constants" tab of the default tag table.
DETAIL	Input	HW_SUB-MODULE	I, Q, M, D, L or constant	The DETAIL parameter is not used. Leave the parameter unconnected.
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of address data to be read: • MODE = 1: Address parameter according to IPv4 (S7-1500 CPUs as of firmware version V1.1) • MODE = 2: MAC address (S7-1500 CPUs as of firmware version V1.5)
DATA	InOut	VARIANT	D, L	Pointer to the area to which the address data of the IO device is written. Use the "IF_CONF_v4" structure for MODE = 1 and the "IF_CONF_MAC" structure for MODE = 2.
DONE	Output	BOOL	I, Q, M, D, L	The instruction was executed successfully. The address data was transferred to the DATA parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Execution of the instruction complete. • 1: Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy STATUS to a free data area.

You can find more information on the data types in "Overview of the valid data types (Page 231)".

Parameter DATA

- Use the "IF_CONF_v4" structure at the DATA parameter to store the address parameter according to IPv4.

Byte	Parameter	Data type	Start value	Description
0 … 1	Id	UINT	30	ID of the "IF_CONF_v4" structure.
2 … 3	Length	UNIT	18	Length of data read in BYTE.
4 ... 5	Mode	UNIT	0	Not relevant for the "GetStationInfo" instruction (left at "0").
6 ... 9	InterfaceAddress	ARRAY [1..4] of BYTE	-	IP address of the IO device in the format IP_V4, e.g. for 192.168.3.10: • addr[1] = 192 • addr[2] = 168 • addr[3] = 3 • addr[4] = 10
10 … 13	SubnetMask	ARRAY [1..4] of BYTE	-	Subnet mask of the IO device in the format IP_V4, e.g. for 255.255.255.0: • addr[1] = 255 • addr[2] = 255 • addr[3] = 255 • addr[4] = 0
14 ... 17	DefaultRouter	ARRAY [1..4] of BYTE	-	IP address of the router in the format IP_V4, e.g. for 192.168.3.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 3 • addr[4] = 1

- Use the "IF_CONF_MAC" structure at parameter DATA for storing the MAC address.

Byte	Parameter	Data type	Start value	Description
0 … 1	Id	UINT	3	ID of the "IF_CONF_MAC" structure.
2 … 3	Length	UNIT	12	Length of data read in BYTE.
4 ... 5	Mode	UNIT	0	Not relevant for the "GetStationInfo" instruction (left at "0").
6 ... 11	MACAddress	ARRAY [1..6] of BYTE	-	MAC address of the IO device, e.g. for 08-00-06-12-34-56 • Mac[1] = 8 • Mac[2] = 0 • Mac[3] = 6 • Mac[4] = 12 • Mac[5] = 34 • Mac[6] = 56

Parameter STATUS

Error code* (W#16#...)	Explanation
0	No error
7000	No job in progress.
7001	First call of the asynchronous instruction "GetStationInfo". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
7002	Additional call of the asynchronous instruction "GetStationInfo". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
8080	Value at the MODE parameter is not supported.
8090	The hardware identifier specified at the LADDR parameter is not configured.
8092	The LADDR parameter does not address a PROFINET IO device.
8093	Invalid data type at the DATA parameter.
80A0	Requested information is not read.
80C0	Addressed IO device is not reachable.
80C3	The maximum number of simultaneous calls of the "GetStationInfo" instruction (10 instances) has been reached.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

Below, you use the „GetStationInfo“ instruction to read the IP address data of an IO device and write the information to a data block. The IP address data includes the IP address, subnet mask and (if used) the address data of the router.

The instruction is executed on the IO controller and reads the IP address information of a lower-level IO device (an ET200MP in this example).

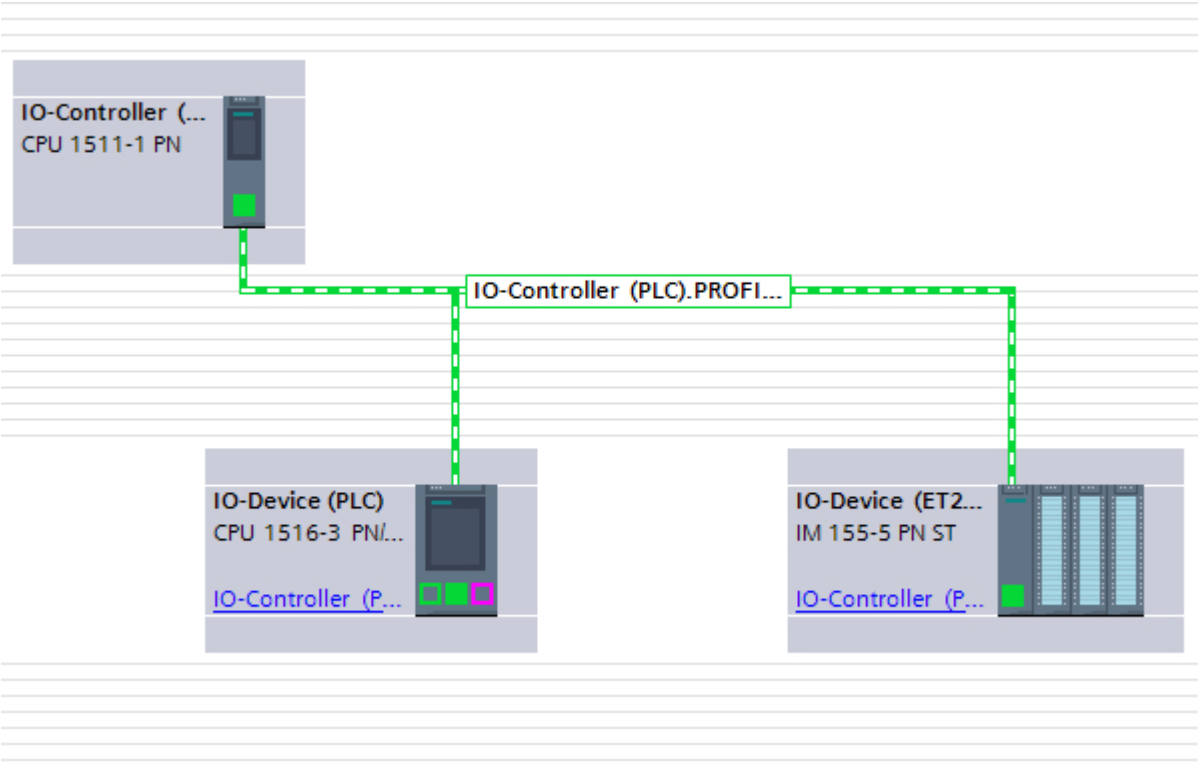

Create five tags and a structure with the IF_CONF_v4 data type in a global data block for storing the address data. Assign any name to the structure (in this case "IP_Address").

SLI_gDB_GetStationInfo

		Name	Data type	Start value
1		Static		
2		execute	Bool	false
3		IP_address	IF_CONF_v4	
4		done	Bool	false
5		busy	Bool	false
6		error	Bool	false
7		status	Word	16#0

Then call the "GetStationInfo" instruction:

- You use the IF_CONF_v4 structure at the DATA parameter.
- Enter the hardware identifier of the IO device at the LADDR parameter. The hardware identifier uniquely identifies the product. To determine the hardware identifier of your IO device, open the PLC tag table and the System constants tab. Then, search for the device in the Name column and for the „Hw_Device“ in the Data type column. The associated value is the hardware identifier that you enter at the LADDR parameter.
- Select "1" (read address parameters according to IPv4) for the MODE parameter.

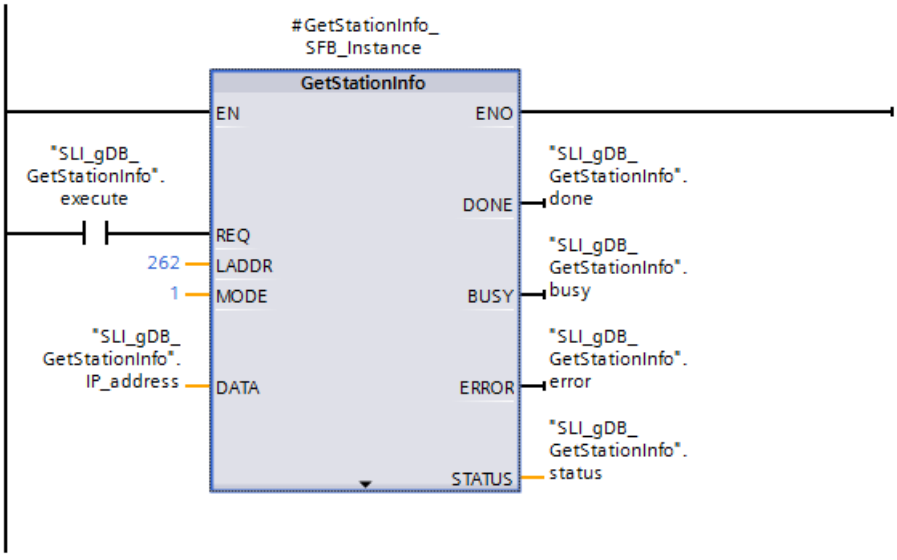

You start reading the address data with REQ= 1 (TRUE). Once the instruction is successfully executed, the IP address data is written to the data block.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Difference between synchronous and asynchronous instructions (Page 596)

GetChecksum: Read out checksum (S7-1200, S7-1500)

Description

The "GetChecksum" instruction reads the checksum of a group of objects. The objects are selected with the parameter "Scope".

Note

Generation of the checksum for blocks

There are differences between standard and safety blocks in terms of the generation of the checksum. You can find additional information in the help for the STEP 7 Safety option package.

Parameter

The following table shows the parameters of the "GetChecksum" instruction:

Parameter	Declaration	Data type	Description
Scope	Input	UINT	With the Scope parameter, you select the objects for which you want to read out the checksum: • 1: Standard blocks (i.e. no safety blocks) • 2: Safety blocks • 3: Text lists
Range (hidden)	Input	UINT	"Range" must have the value 0.
Subrange (hidden)	Input	UINT	"Subrange" must have the value 0.
Done	Output	BOOL	The instruction was executed successfully. The checksum was transferred to the Checksum parameter.
Busy	Output	BOOL	Status parameter • 0: Execution of the instruction complete. • 1: Execution of the instruction not yet complete.
Error	Output	BOOL	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the "Status" parameter.

Parameter	Declaration	Data type	Description
Status	Output	WORD	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy STATUS to a free data area.
Checksum	InOut	VARIANT	Checksum The "Checksum" parameter must point to a tag of the data type ARRAY[0..7] of BYTE; unused bytes are set to 0.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter "Status"

Error code* (W#16#...)	Explanation
0	No error
7001	First call: Generation of the checksum initiated
7002	Intermediate call: Generation of the checksum already active
8080	Invalid value of the "Scope" parameter
8081	The "Range" parameter is not supported in this firmware version and must have the value 0.
8082	The "Subrange" parameter is not supported in this firmware version and must have the value 0.
8091	The "GetChecksum" instruction is not supported by the CPU, or the addressed objects are not available, for example because there is no failsafe program on the CPU.
8852	The tag to which the "Checksum" parameter points has the wrong data type.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you read out the checksum of the standard blocks of a S7-1500 CPU.

Create the following tags for storing the data in a global data block:

SLI_gDB_GetChecksum

	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	scope	UInt	1
4	checksum	Array[0..7] of Byte	
5	busy	Bool	false
6	done	Bool	false
7	error	Bool	false
8	memErrStatus	Word	16#0

Create an FB. Create the following local tags in the FB.

7	Static		
8	GetChecksum_Instance	GetChecksum	
9	status	Word	16#0

Network 1: Interconnect the parameters of the "GetChecksum" instruction as follows.

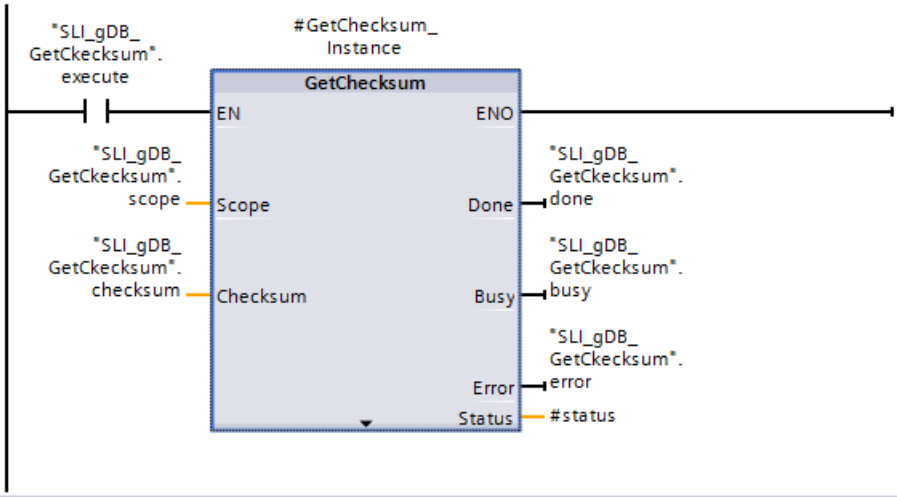

Network 2: If an error occurs, save the status as follows.

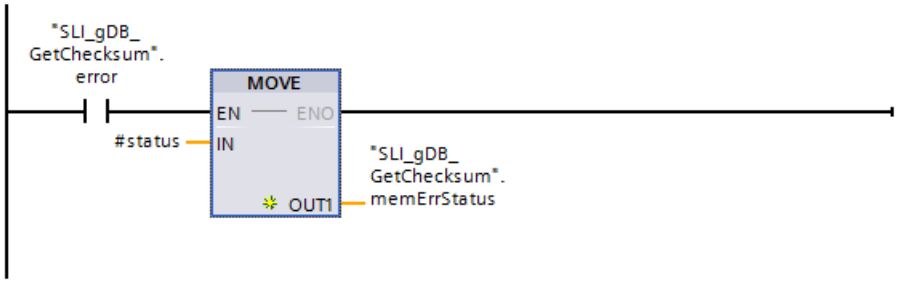

If the normally open contact ("execute") supplies the signal state "TRUE", the "GetChecksum" instruction is executed. The checksum of the standard blocks is read according to the value "1" of the input parameter SCOPE ("scope"). The checksum is output at the CHECKSUM ("checksum") parameter. The success status of GetChecksum is displayed at the output parameter DONE ("done").

Output parameters STATUS ("#status" and "memErrStatus") and ERROR ("error") indicate that the processing in the example was completed without error.

SLI_gDB_GetCkecksum

	Name	Data type	Start value	Monitor value
1	Static			
2	execute	Bool	false	TRUE
3	scope	UInt	1	1
4	checksum	Array[0..7] of Byte		
5	checksum[0]	Byte	16#0	16#F3
6	checksum[1]	Byte	16#0	16#7A
7	checksum[2]	Byte	16#0	16#5F
8	checksum[3]	Byte	16#0	16#9F
9	checksum[4]	Byte	16#0	16#F3
10	checksum[5]	Byte	16#0	16#35
11	checksum[6]	Byte	16#0	16#22
12	checksum[7]	Byte	16#0	16#CC
13	busy	Bool	false	FALSE
14	done	Bool	false	TRUE
15	error	Bool	false	FALSE
16	memErrStatus	Word	16#0	16#0000

You can also find the checksum in the CPU properties.

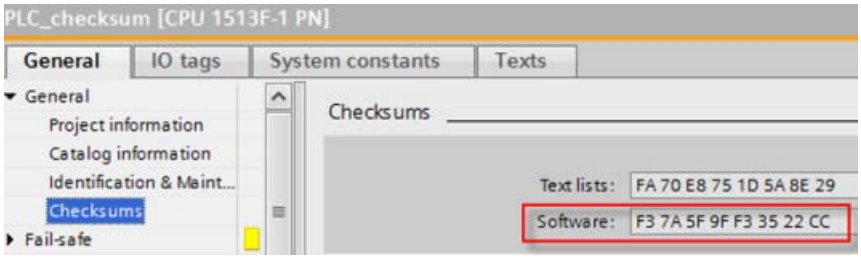

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Comparison of PLC programs based on checksums (Page 7588)

GetSMCinfo: Reading out information about the SIMATIC memory card (S7-1500)

Description

Use the instruction "GetSMCinfo" to retrieve information about the inserted SIMATIC memory card. The information to be read out is selected with the parameter "Mode":

- Mode = 0, 1, 2, 3: Permissible for standard CPUs
- Mode = 10, ... 13 and 20, ... 23: Permissible for S7-1500-R/H CPUs
  The Mode values 10, ... 13 refer to the CPU with redundancy ID 1.
  The Mode values 20, ... 23 refer to the CPU with redundancy ID 2.

Enable reading of the address data via the REQ control parameter. The output parameters Done, Busy, Error and Status indicate the status of the job.

Dependency of the information that can be read out on the system status of the R/H system

- Memory size

System state and role of the CPU, if applicable	Pairing state of the CPUs	Permissible mode values
RUN-Solo, primary CPU	Not paired	• Redundancy ID=1: Mode=10 (own SMC) • Redundancy ID=2: Mode=20 (own SMC)
	Paired	• Redundancy ID=1: – Mode=10 (own SMC) or – Mode=20 (SMC of partner CPU) • Redundancy ID=2: – Mode=10 (SMC of partner CPU) or – Mode=20 (own SMC)
SYNCUP, primary CPU	-	• Redundancy ID=1: – Mode=10 (own SMC) or – Mode=20 (SMC of partner CPU) • Redundancy ID=2: – Mode=10 (SMC of partner CPU) or – Mode=20 (own SMC)
RUN-Redundant	-	• Redundancy ID=1: – Mode=10 (own SMC) or – Mode=20 (SMC of partner CPU) • Redundancy ID=2: – Mode=10 (SMC of partner CPU) or – Mode=20 (own SMC)

- Memory in use

System state and role of the CPU, if applicable	Pairing state of the CPUs	Permissible mode values
RUN-Solo, primary CPU	irrelevant	• Redundancy ID=1: Mode=11 (own SMC) • Redundancy ID=2: Mode=21 (own SMC)
SYNCUP, primary CPU	-	Depending on the start time of GetSMCinfo and the synchronization of the two CPUs, the information can either be read or not. If it can be read: • Redundancy ID=1: – Mode=11 (own SMC) or – Mode=21 (SMC of partner CPU) • Redundancy ID=2: – Mode=11 (SMC of partner CPU) or – Mode=21 (own SMC)
RUN-Redundant	-	• Redundancy ID=1: – Mode=11 (own SMC) or – Mode=21 (SMC of partner CPU) • Redundancy ID=2: – Mode=11 (SMC of partner CPU) or – Mode=21 (own SMC)

Note

Determination of the occupied memory space of memory cards of different memory sizes

If you use memory cards with different memory sizes in the two CPUs of an R/H system, the determination of the occupied memory space can lead to different results despite identical memory content. The reason for this is the different cluster size of the memory cards.

- Percentage of service life used

System state and role of the CPU, if applicable	Pairing state of the CPUs	Permissible mode values
RUN-Solo, primary CPU	irrelevant	• Redundancy ID=1: Mode=12 (own SMC) • Redundancy ID=2: Mode=22 (own SMC)
SYNCUP, primary CPU	-	Depending on the start time of GetSMCinfo and the synchronization of the two CPUs, the information can either be read or not. If it can be read: • Redundancy ID=1: – Mode=12 (own SMC) or – Mode=22 (SMC of partner CPU) • Redundancy ID=2: – Mode=12 (SMC of partner CPU) or – Mode=22 (own SMC)
RUN-Redundant	-	• Redundancy ID=1: – Mode=12 (own SMC) or – Mode=22 (SMC of partner CPU) • Redundancy ID=2: – Mode=12 (SMC of partner CPU) or – Mode=22 (own SMC)

- Configured threshold of service life

System state and role of the CPU, if applicable	Pairing state of the CPUs	Permissible mode values
RUN-Solo, primary CPU	irrelevant	• Redundancy ID=1: – Mode=13 (own SMC) or – Mode=23 (SMC of partner CPU) • Redundancy ID=2: – Mode=13 (SMC of partner CPU) or – Mode=23 (own SMC)
SYNCUP, primary CPU	-	• Redundancy ID=1: – Mode=13 (own SMC) or – Mode=23 (SMC of partner CPU) • Redundancy ID=2: – Mode=13 (SMC of partner CPU) or – Mode=23 (own SMC)
RUN-Redundant	-	• Redundancy ID=1: – Mode=13 (own SMC) or – Mode=23 (SMC of partner CPU) • Redundancy ID=2: – Mode=13 (SMC of partner CPU) or – Mode=23 (own SMC)

Parameter

The following table shows the parameters of the instruction "GetSMCinfo":

Parameter	Declaration	Data type	Description
REQ	Input	BOOL	Control parameter Request Activates the reading of the information with REQ = "1".
Mode	Input	UINT	Use the Mode parameter to select the information about the SIMATIC memory card that you want to read out: • 0: Memory size in KB (1 KB = 1024 bytes) • 1: Assigned memory space in KB • 2: Maintenance information: Previously used up portion of the service life in % Note: – With the S7-1500 software controllers and S7-PLCSIM, 0x00 is always entered in Info for Mode=2 (no maintenance information available). – With SIMATIC Memory Cards, which do not generally support this information or which cannot supply this information with the current firmware of your CPU, 0xFF is entered in Info. • 3: Set proportion of the service life as a percentage after which the CPU creates a diagnostics buffer entry and activates the maintenance LED. Note: – This information is read from the memory card, but instead from the planning data. If you have disabled the generation of diagnostics interrupt, 0xFF is entered in Info. – With the S7-1500 software controllers and S7-PLCSIM, 0xFF is always entered in Info for Mode=3. • 10 or 20: Corresponds to Mode 0 with standard CPUs – Mode 10: Memory size of the SMC in CPU with redundancy ID 1 – Mode 20: Memory size of the SMC in CPU with redundancy ID 2 • 11 or 21: Corresponds to Mode 1 with standard CPUs – Mode 11: Memory in use of the SMC in CPU with redundancy ID 1 – Mode 21: Memory in use of the SMC in CPU with redundancy ID 2 • 12 or 22: Corresponds to Mode 2 with standard CPUs – Mode 12: Percentage of service life used for CPU with redundancy ID 1 – Mode 22: Percentage of service life used for CPU with redundancy ID 2 • 13 or 23: Corresponds to Mode 3 with standard CPUs – Mode 13: Configured threshold of service life for CPU with redundancy ID 1 – Mode 23: Configured threshold of service life for CPU with redundancy ID 2

Parameter	Declaration	Data type	Description
Done	Output	BOOL	1: The instruction has been executed successfully. The information that was read is transferred to the Info parameter.
Busy	Output	BOOL	Status parameter • 0: Execution of the instruction complete. • 1: Execution of the instruction not yet complete.
Error	Output	BOOL	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the Status parameter.
Status	Output	WORD	Error code
Info	InOut	UDINT	Buffer for the information that was read.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

Determining the I&M 0 data

The I&M 0 data of the SIMATIC memory card cannot be determined with the GetSMCinfo instruction. For this use the instruction "Get_IM_Data (Page 2590)".

Parameter Status

Error code* (W#16#...)	Explanation
0	No error.
7000	No job in progress.
7001	First call: Trigger job (Busy = 1, Done = 0).
7002	Intermediate call: Job already active (Busy = 1, Done = 0).
8080	Invalid value of the "Mode" parameter
8081 (only S7-1200)	No SIMATIC memory card inserted.
8086	The R/H system is not in the RUN-Redundant system state. Therefore, the data of the partner CPU is not available.
8092	No data available, for example, because "GetSMCinfo" is not supported by the CPU.
80C3	The maximum number of simultaneous calls of the "GetSMCinfo" instruction has been reached.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you determine the memory size of the SIMATIC memory card used.

Create the following tags for storing the data in a global data block:

SLI_gDB_GetSMCinfo			
	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	mode	UInt	0
4	getInfo	UDInt	0
5	done	Bool	false
6	busy	Bool	false
7	error	Bool	false
8	status	Word	16#0

Create an FB. In the status area of the FB, create a local tag"#done" with the Bool data type.

Interconnect the parameters of the "GetSMCinfo" instruction as follows.

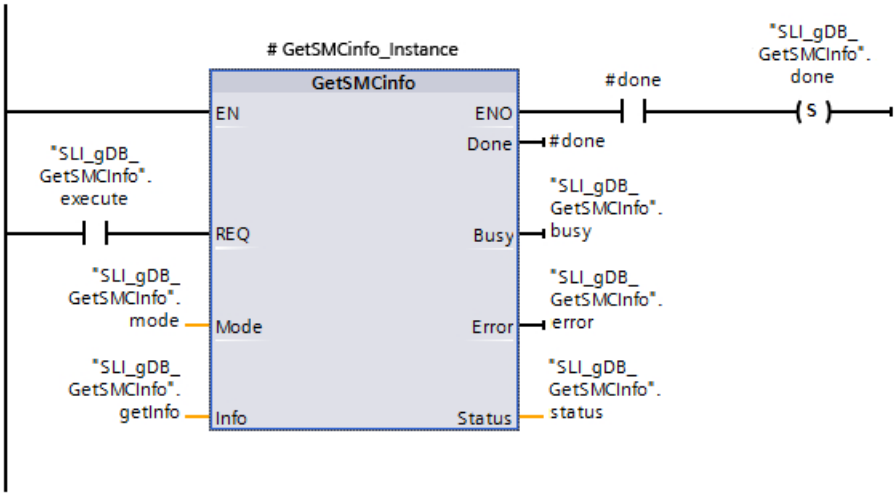

The GetSMCinfo instruction is executed only when the input parameter REQ ("execute") returns the signal state "TRUE". The mode to be used for reading out the SIMATIC Memory Card is stored in the input parameter MODE ("mode"). The memory size of the SIMATIC memory card is read according to the value "0" and displayed in KB at parameter INFO ("getInfo"). The success status of GetSMCinfo is displayed at the output parameter DONE ("#done") and stored in the tag "done".

Output parameters STATUS ("status") and ERROR ("error") indicate that the processing in the example was completed without error.

SLI_gDB_GetSMCinfo				
	Name	Data type	Start value	Monitor value
1	Static			
2	execute	Bool	false	TRUE
3	mode	UInt	0	0
4	getInfo	UDInt	0	2025008
5	done	Bool	false	TRUE
6	busy	Bool	false	FALSE
7	error	Bool	false	FALSE
8	status	Word	16#0	16#0000

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

GetClockStatus: Read out status of the CPU clock (S7-1500)

Description

You use the instruction "GetClockStatus" to read the following information from the internal CPU clock:

- Information on whether you have enabled time synchronization via NTP server in the properties of your CPU (Bit1 of the ClockStatus parameter).
- Information on whether time synchronizations have been missed (Bit0 of the ClockStatus parameter).
- Information on whether daylight saving or standard time is active (Bit2 of the ClockStatus parameter).

Parameters

The following table shows the parameters of the "GetClockStatus" instruction:

Parameters	Declaration	Data type	Description
ClockStatus	Output	WORD	Status of the internal CPU clock: • Bit0: Information on whether time synchronizations have been missed – Bit0 = 1: One or more time synchronizations have been missed. – Bit0 = 0: No time synchronization has been missed. Note on evaluating Bit0: You may only evaluate Bit0 when time synchronization is enabled, i.e. when Bit1 = 1 Bit0 is reset to 0 in the following cases: – After a CPU restart – After the time is set Please note that the earliest change to 1 with an S7-1500 CPU is ca. three minutes after a missed time synchronization. • Bit1 = 1: Time synchronization is enabled • Bit2 – = 1: Daylight saving is active – = 0: Standard time is active • Bits 3 to 15: 0 (reserved)
Ret_Val	Output	INT	Return value

You can find additional information on valid data types under: Overview of the valid data types (Page 231)

Parameter Ret_Val

Error code* (W#16#...)	Explanation
0	No error
8092	No data available, for example because "GetClockStatus" is not supported by your CPU.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

DeviceStates: Read module status information in an IO system (S7-1200, S7-1500)

Description

You use the instruction "DeviceStates" to query specific status information for all modules in an IO system, which means:

- Either for all IO devices in a PROFINET IO system
- Or for all DP slaves in a DP master system

The Boolean value that is output indicates the modules to which the selected status applies. For example, read out which IO devices are currently disabled in a PROFINET IO system.

Information is also displayed as to whether the status information to be read applies to at least one of the IO devices or DP slaves.

The instruction can be called in a cyclic OB as well as in an interrupt OB (e.g. OB82 - diagnostic interrupt).

Parameters

The following table shows the parameters of the "DeviceStates" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IOSYSTEM	I, Q, M, L or constant	Hardware identifier of the PROFINET IO or DP master system (see description below)
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of status information to be read (see description below)
RET_VAL	Return	INT	I, Q, M, D, L	Status of instruction (see description below)
STATE	InOut	VARIANT	I, Q, M, D, L	Buffer for status of the IO devices or the DP slaves (see description below)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter LADDR

You select the PROFINET IO or DP master system at the LADDR parameter by means of the hardware identifier.

The hardware identifier is available:

- Either in the Network view of the properties of the PROFINET IO or DP master system.
- Or in the PLC tag table in the listed system constants with the data type HW_IOSYSTEM.

Parameter MODE

You use the MODE parameter to read out status information. You can read out one of the following status information items for the entire PROFINET IO or DP master system:

- 1: IO devices/DP slaves are configured
- 2: IO devices/DP slaves are faulty
- 3: IO devices/DP slaves are disabled
- 4: IO devices/DP slaves exist
- 5: IO devices/DP slaves for which a problem has occurred. For example:
  - Maintenance demanded or recommended
  - Not accessible
  - Not available
  - Error occurred

Parameter STATE

With the STATE parameter, the status of the IO devices/DP slaves that was selected with the MODE parameter is output.

If the status selected using MODE applies to an IO device/DP slave, the following bits are set to "1" at the STATE parameter:

- Bit 0 = 1: Group display. The bit n of at least one IO device/DP slave was set to "1".
- Bit n = 1: The status selected with MODE applies to the IO device/DP slave.
  - With a PROFINET IO system, bit n corresponds to the device number of the respective IO device (see properties of the PROFINET interface in the device view and network view)
  - With a PROFIBUS DP system, bit n corresponds to the PROFIBUS address of the DP slave (see properties of the DP slave in the device view and network view)

Use "BOOL" or "Array of BOOL" as data type:

- To only output the bit for group display of the status information, you can use the data type BOOL at the STATE parameter.
- To output the status information for all IO devices/DP slaves, use Array of BOOL with the following length:
  - With PROFINET IO system: 1024 bits
  - With DP master system: 128 bits

Parameter RET_VAL

Error code* (W#16#...)	Description
0	No error
8091	Hardware identifier of the LADDR parameter does not exist. Check (for example, in the system constants) whether the value for LADDR exists in the project.
8092	LADDR does not address a PROFINET IO or DP master system.
8093	Invalid data type at the STATE parameter.
80B1	Instruction "DeviceStates" is not supported by the CPU.
80B2	The selected MODE parameter is not supported by the used CPU for the IO system specified in the LADDR parameter.
8452	The complete status information does not fit in the tag configured in the STATE parameter. Note: When the tag configured at STATE is checked for adequate field length, the CountOfElements instruction is called. This counts the fill elements when the data type VARIANT points to an Array of BOOL; with an Array [0...120] of BOOL e.g. 128 is determined as the field length. This has the consequence that the error code W#16#8452 is only returned by DeviceStates when the number of field elements you have set plus the number of fill elements created by the CPU falls below the value 1024 or 128
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example - Reading out the presence of IO devices in the PROFINET IO master system

In the following example, you query the existence of IO devices in an IO system. The IO system consists of two CPUs of the S7-1500 series. The "PLC_14" CPU will contain the program, including the "DeviceStates" instruction. The "PLC_13" CPU will be configured as an IO device.

In the "PLC_14" CPU: Create three tags and a "mySTATE" structure (with the Array of BOOL data type) in a global data block for storing the data.

My_gDB_DeviceStates

	Name	Data type	Start value
1	Static		
2	myLADDR	HW_IOSYSTEM	258
3	myMODE	UInt	4
4	returnValue	Int	0
5	mySTATE	Array[0..1023] ...	
6	mySTATE[0]	Bool	false
7	mySTATE[1]	Bool	false
8	mySTATE[2]	Bool	false
9	mySTATE[3]	Bool	false
10	mySTATE[4]	Bool	false
11	mySTATE[5]	Bool	false
12	mySTATE[6]	Bool	false

In the "PLC_14" CPU: The instruction is called in a cyclic OB. Interconnect the parameters of the instruction as follows.

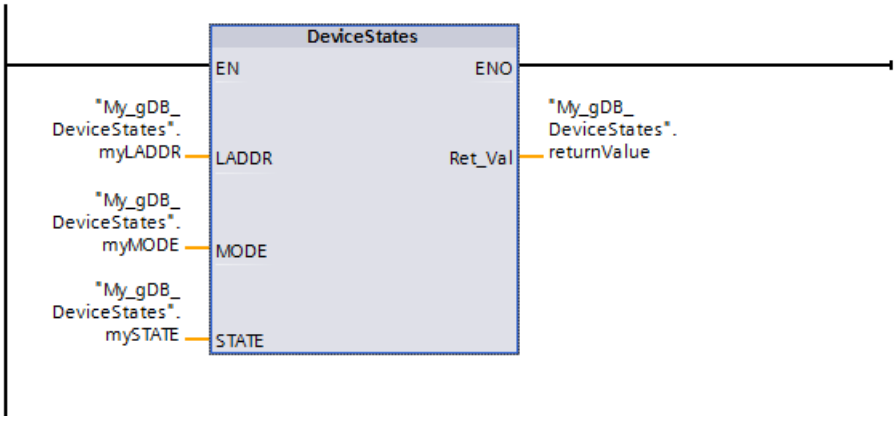

In the "PLC_13" CPU: Set up this CPU "PLC_13" as an IO device using the CPU properties. The IO device receives the device number 1.

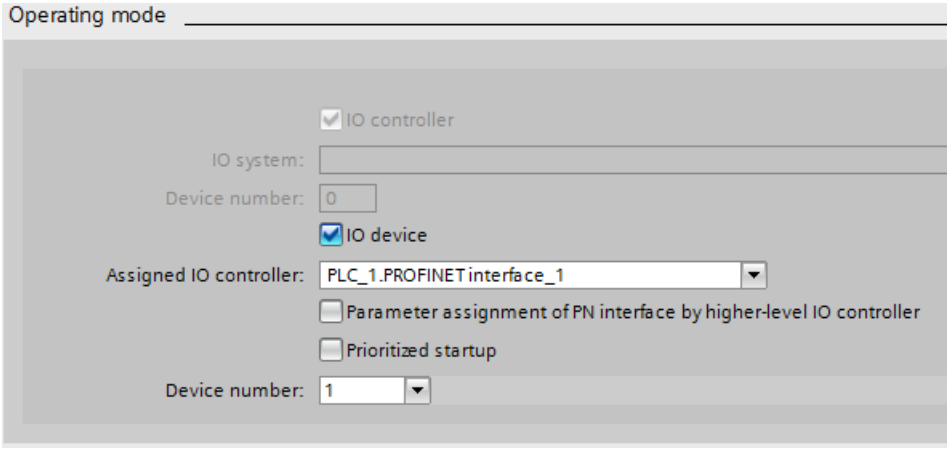

An IO system is displayed in the Network view.

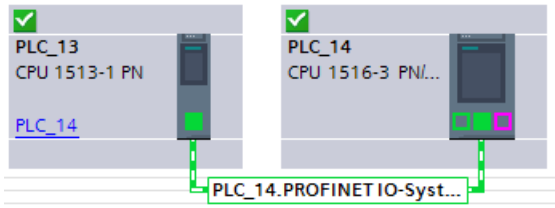

In the "PLC_14" CPU: The HW identifier of the IO system is made known to the "DeviceStates" instruction through parameter LADDR ("myLADDR"). According to the value "4" of parameter MODE ("myMODE"), the IO system is searched for IO devices.

At parameter STATE ("mySTATE"), the presence is output for the IO devices (based on the value of parameter MODE). Bit 0 serves as a group value and indicates that IO devices are present. Bit 1 indicates that the IO device with the device number 1 is present.

The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

My_gDB_DeviceStates

	Name	Data type	Start value	Monitor value
1	Static			
2	myLADDR	HW_IOSYSTEM	258	16#0102
3	myMODE	UInt	4	4
4	returnValue	Int	0	0
5	mySTATE	Array[0..1023] ...		
6	mySTATE[0]	Bool	false	TRUE
7	mySTATE[1]	Bool	false	TRUE
8	mySTATE[2]	Bool	false	FALSE
9	mySTATE[3]	Bool	false	FALSE
10	mySTATE[4]	Bool	false	FALSE
11	mySTATE[5]	Bool	false	FALSE
12	mySTATE[6]	Bool	false	FALSE

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Example - Reading the faulty stations of a PROFINET IO master system

A PROFINET IO system consists of 4 IO devices with the device numbers 1, 2, 3 and 4. The IO device with number 2 is faulty.

The instruction "DeviceStates" is executed for the PROFINET IO system with MODE = 2 (faulty/not faulty).

The following bits are set in the STATE parameter:

- Bit 0 = 1: A fault exists for at least one of the IO devices.
- Bit 1 = 0: IO device with device number 1 is not faulty.
- Bit 2 = 1: IO device with device number 2 is faulty.
- Bit 3 = 0: IO device with device number 3 is not faulty.
- Bit 4 = 0: IO device with device number 4 is not faulty.

- Bit 5 = 0: Irrelevant
- Bit 6 = 0: Irrelevant
- ...

Example - Reading the faulty stations of a PROFIBUS DP master system

A DP master system consists of 4 DP slaves with the PROFIBUS addresses 3, 4, 5 and 6. The DP slave with address 4 is faulty.

The instruction "DeviceStates" is executed for the DP master system with MODE = 2 (faulty/not faulty).

The following bits are set in the STATE parameter:

- Bit 0 = 1: A fault exists for at least one of the DP slaves.
- Bit 1 = 0: Irrelevant
- Bit 2 = 0: Irrelevant
- Bit 3 = 0: DP slave with address 3 is not faulty.
- Bit 4 = 1: DP slave with address 4 is faulty.
- Bit 5 = 0: DP slave with address 5 is not faulty.
- Bit 6 = 0: DP slave with address 6 is not faulty.
- Bit 7 = 0: Irrelevant
- Bit 8 = 0: Irrelevant
- ...

See also

Switching display formats in the program status (Page 7698)

ModuleStates: Read module status information of a module (S7-1200, S7-1500)

Description

You can use the "ModuleStates" instruction to read the status information of the modules of a PROFINET IO device or PROFIBUS DP slave.

The Boolean value that is output indicates the modules to which the selected status applies. You can, for example, read which modules are currently disabled in a PROFINET IO device.

Information is also displayed as to whether the status information to be read applies to at least one of the modules.

The instruction can be called in a cyclic OB as well as in an interrupt OB (e.g. OB82 - diagnostic interrupt).

Parameters

The following table shows the parameters of the "ModuleStates" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_DEVICE	I, Q, M, D, L or constant	Hardware identifier of the station (see description below)
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of module status information to be read (see description below)
RET_VAL	Return	INT	I, Q, M, D, L	Status of instruction (see description below)
STATE	InOut	VARIANT	I, Q, M, D, L	Buffer for the module status (see description below)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter LADDR

You select the IO device or the DP slave at the LADDR parameter by means of the hardware identifier of the station.

The hardware identifier is available:

- Either in the network view of the properties of the IO device station or DP slave station.
- Or in the PLC tag table for the listed system constants with the data type HW_DEVICE (for an IO device) or with the data type HW_DPSLAVE (for a DP slave).

Parameter MODE

You use the MODE parameter to read out status information. One of the following status information items can be read for the modules:

- 1: Modules are configured
- 2: Modules are faulty
- 3: Modules are disabled
- 4: Modules exist
- 5: There is a problem in the modules. For example:
  - Maintenance demanded or recommended
  - Not accessible
  - Not available
  - Error occurred

Parameter STATE

The STATE parameter outputs the status of the modules selected with the MODE parameter.

If the status selected using MODE applies to a module, the following bits are set to "1":

- Bit 0 = 1: Group display. The bit n of at least one module was set to "1".
- Bit n = 1: The status selected with MODE applies to the module in slot n-1 (example: bit 3 = slot 2).

Use "BOOL" or "Array of BOOL" as data type:

- To only output the bit for group display of the status information, use the BOOL data type at the STATE parameter.
- To output the status information for all modules, use Array of BOOL with a length of 128 bits.

Parameter RET_VAL

Error code* (W#16#...)	Description
0	No error
8091	Hardware identifier of the LADDR parameter does not exist. Check (for example, in the system constants) whether the value for LADDR exists in the project.
8092	LADDR does not address an IO device or DP slave.
8093	Invalid data type at the STATE parameter.
80B1	The instruction "ModuleStates" is not supported by the CPU.
80B2	The selected MODE parameter is not supported by the used CPU for the IO device/the DP slave in the LADDR parameter.
8452	The complete status information does not fit in the tag configured in the STATE parameter. Note: When the tag configured at STATE is checked for adequate field length, the CountOfElements instruction is called. This counts the fill elements when the data type VARIANT points to an Array of BOOL; with an Array [0...120] of BOOL e.g. 128 is determined as the field length. This has the consequence that the error code W#16#8452 is only returned by ModuleStates when the number of field elements you have set plus the number of fill elements created by the CPU falls below the value 128
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you query the existence of modules of a PROFINET IO device. The IO system consists of two CPUs of the S7-1500 series. The "PLC_14" CPU will contain the program, including the "ModuleStates" instruction. The "PLC_13" CPU will be configured as an IO device.

In the "PLC_14" CPU: Create three tags and a "mySTATE" structure (with the Array of BOOL data type) in a global data block for storing the data.

My_gDB_ModuleStates			
	Name	Data type	Start value
1	Static		
2	myLADDR	HW_DEVICE	260
3	myMODE	UInt	4
4	returnValue	Int	0
5	mySTATE	Array[0..127] of Bool	
6	mySTATE[0]	Bool	false
7	mySTATE[1]	Bool	false
8	mySTATE[2]	Bool	false
9	mySTATE[3]	Bool	false
10	mySTATE[4]	Bool	false
11	mySTATE[5]	Bool	false
12	mySTATE[6]	Bool	false

In the "PLC_14" CPU: The instruction is called in a cyclic OB. Interconnect the parameters of the instruction as follows.

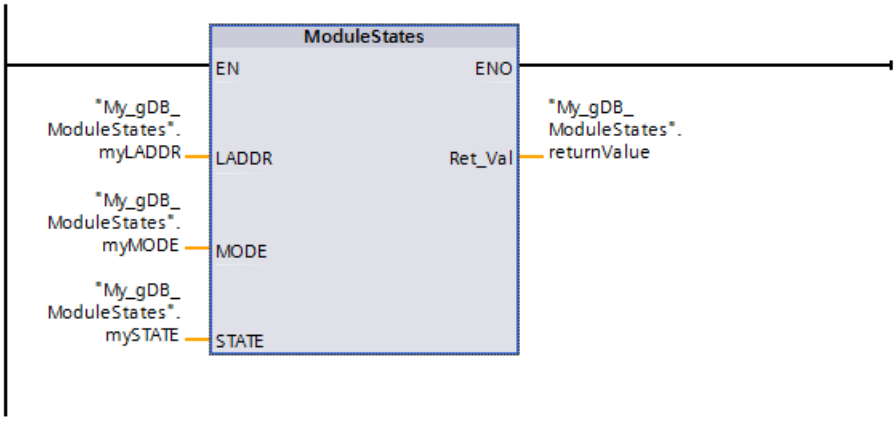

In the "PLC_13" CPU: Set up this CPU "PLC_13" as an IO device using the CPU properties.

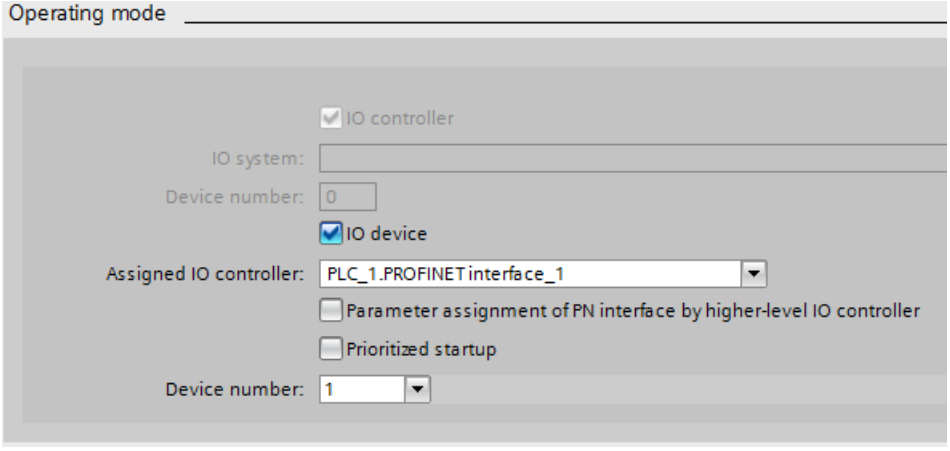

An IO system is displayed in the Network view.

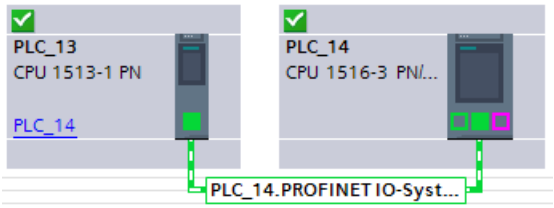

In the "PLC_14" CPU: A module is located in slot 1 of the IO device.

In the "PLC_14" CPU: The HW identifier of the IO device is made known to the "ModuleStates" instruction through parameter LADDR ("myLADDR"). According to the value "4" of parameter MODE ("myMODE"), the IO device is searched for modules.

At parameter STATE ("mySTATE"), the presence is output for the modules (based on the value of parameter MODE) . Bit 0 serves as a group value and indicates that modules are present. Bit 2 indicates that a module is present in slot 1.

The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

My_gDB_ModuleStates

	Name	Data type	Start value	Monitor value
1	Static			
2	myLADDR	HW_DEVICE	260	16#0104
3	myMODE	UInt	4	4
4	returnValue	Int	0	0
5	mySTATE	Array[0..127] of Bool		
6	mySTATE[0]	Bool	false	TRUE
7	mySTATE[1]	Bool	false	FALSE
8	mySTATE[2]	Bool	false	TRUE
9	mySTATE[3]	Bool	false	FALSE
10	mySTATE[4]	Bool	false	FALSE
11	mySTATE[5]	Bool	false	FALSE
12	mySTATE[6]	Bool	false	FALSE

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Example

An IO device contains 4 modules in slots 1 to 4. The module in slot 2 is faulty.

The instruction "ModuleStates" is executed for the IO device with MODE = 2 (faulty/not faulty).

The following bits are set in the STATE parameter:

- Bit 0 = 1: A fault exists for at least one module.
- Bit 1 = 0: Slot number 0 (used by IO device)
- Bit 2 = 0: Module in slot number 1 is not faulty.
- Bit 3 = 1: Module in slot number 2 is faulty.
- Bit 4 = 0: Module in slot number 3 is not faulty.
- Bit 5 = 0: Module in slot number 4 is not faulty.
- Bit 6 = 0: Irrelevant
- Bit 7 = 0: Irrelevant

See also

Switching display formats in the program status (Page 7698)

GEN_DIAG: Generate diagnostics information (S7-1500)

Description

The "GEN_DIAG" instruction generates diagnostics information for hardware components from other manufacturers for use in TIA Portal diagnostics. The GSD(GSDL/GSDML) file supplied by the manufacturer must first be installed before the instruction can be used.

The instruction generates all diagnostic events (including those for maintenance).

- Use the LADDR parameter to select the hardware component for which you want to generate a diagnostic event.
- Use the MODE parameter to specify whether the event is to be outgoing or incoming.
- Use the DiagEvent parameter to define the diagnostic event in the DiagnosticDetail structure. When you define a tag at the DiagEvent parameter, the structure is created automatically in the local interface of the block.

The diagnostics information is provided synchronously. Diagnostics information transfer and alarm output are asynchronous.

NOTICE
Failsafe-specific errors messages are not valid If you define failsafe-specific diagnostics information at the DiagEvent parameter, the instruction will check this information and output the error code 80A1.

Parameters

The following table shows the parameters of the "GEN_DIAG" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_ANY	I, Q, M, D, L or constant	Identification number of the hardware component
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of incoming/outgoing information: • 1: The diagnostic event specified is an incoming event • 2: The diagnostic event specified is an outgoing event • 3: All diagnostic events are outgoing. There are therefore no hardware component faults (green diagnostics symbol). The DiagEvent parameter is not evaluated when MODE = 3.
DiagEvent	InOut	DiagnosticDetail	L, D	Specifies the diagnostic event (see "DiagEvent parameter").
RET_VAL	Return	INT	I, Q, M, D, L	Status of instruction / error message (see "RET_VAL parameter")

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter DiagEvent

The structure DiagnosticDetail is a system data type for specifying the diagnostic event. Its structure is as follows:

Parameter		Data type	Description
DiagnosticDetail		Struct	
	ChannelInfo	WORD	Channel properties (0...7)
	ALID	UINT	Local ID of the alarm. The ID uniquely identifies the alarm.
	TextID	UINT	ID of an alarm text in a text list.
	ChannelNumber	UINT	Manufacturer-specific channel number (0x0000 — 0x7FFF)
	Addval_0	DWORD	Placeholder for additional information The value / list of values depends on the connection error.
	TextID2	UINT	Texts for CPU response (mode, OB calls, etc.).
	LADDR	HW_ANY	Identical to the LADDR parameter.
	TextListId	UINT	• 0: No text list • ≠0: ID of the text list
	ChannelDirection	UINT	• 0000: Irrelevant • FFF1: Input • FFF2: Output • FFF3: Input/Output
	Addval_1	DWORD	Placeholder for additional information on the channel error (depends on the GSD file). For channel error types, see also: IEC 61158 (PROFINET IO Type 10 and PROFIBUS DP Type 3).

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error
8080	Value in the MODE parameter is not supported.
8090	Identification for hardware component not available at the LADDR parameter.
8091	Diagnostics information cannot be generated for the hardware component addressed at the LADDR parameter.
80A1	• Content of the DiagnosticsDetail structure at the DiagEvent parameter is invalid or inconsistent. • Failsafe-specific diagnostics information defined at the DiagEvent parameter (not valid).
80A4	Hardware component addressed cannot be accessed.
80C1	Insufficient resources for parallel execution.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

GET_DIAG: Read diagnostic information (S7-1200, S7-1500)

Description

The "GET_DIAG" instruction is used to read the diagnostic information of a hardware component. The hardware component is selected with the parameter LADDR. With the MODE parameter, you select which diagnostic information is read. The S7-1200 CPU ignores the parameter LADDR when MODE=0.

Parameters

The following table shows the parameters of the "GET_DIAG" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	Use the MODE parameter to select which diagnostics data is output.
LADDR	Input	HW_ANY (UINT)	I, Q, M, L or constant	Hardware ID of the hardware component
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction
CNT_DIAG	Output	UINT	I, Q, M, D, L	Reserved (always "0").
DIAG	InOut	VARIANT	I, Q, M, D, L	Diagnostic information corresponds to the selected mode, see table below
DETAIL	InOut	VARIANT	I, Q, M, D, L	Parameter is hidden. Do not use this parameter!

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter MODE

Depending on the value at the MODE parameter, different diagnostic data is output at the DIAG, CNT_DIAG and DETAIL output parameters.

MODE	Description	DIAG	CNT_DIAG
0	Output of all supported diagnostic information for a hardware component as DWORD, where Bit X=1 indicates that mode X is supported.	Bits of the DWORD data type: • Bit 0 = 1: MODE 0 is supported • Bit 1 = 1: MODE 1 is supported • Bit 2 = 1: MODE 2 is supported • Bits 3 to 31: Not relevant With MODE=0, the S7-1200 CPU ignores the parameter LADDR.	0
1	Output of the diagnostic status of the addressed hardware component.	Structure DIS (see below for description): • MaintenanceState • ComponentStateDetail • OwnState • IOState • OperatingState	0
2	Output of the status of all subordinate components of the addressed hardware component.	Structure DNN (see below for description): • SubordinateState • SubordinateIOState • DNNmode	0

DIS structure

With parameter MODE = 1, the diagnostic information is output in accordance with the DIS structure. In this case, enter the system data type "DIS" as data type for the tag declaration.

Note

Determining the OperatingState

When determining the OperatingState of a standard CPU, you need to select LADDR = 52 (hardware identifier of CPUExecUnit).

When determining the OperatingState of a CPU of an R/H system, you need to select LADDR = 65147 (hardware identifier of Local1~HCPUredCtrl) or LADDR = 65347 (hardware identifier of Local2~HCPUredCtrl).

When determining the system state of an R/Y system, you need to select LADDR = 34 (hardware identifier of Local1~RHSystem).

The following table shows the meaning of the individual parameter values.

Parameter	Data type	Value	Description	Rule for forming the parameter
MaintenanceS-tate	DWORD	Enum		MaintenanceState depends on Component-StateDetail.
		0	Good	• Bit 15 of ComponentStateDetail is always 1. • Bits 11 to 14 of ComponentStateDetail can have the value 0 or 1. • Bit 3 of ComponentStateDetail can be 1. • All other bits of ComponentStateDetail are 0.
		1	The hardware component is disabled.	• If bit 16 of ComponentStateDetail is 1 • If bit 17 of ComponentStateDetail is 1
		2	-	-
		3	-	-
		4	-	-
		5	Maintenance required	If bit 4 of ComponentStateDetail is 1
		6	Maintenance demanded	If bit 5 of ComponentStateDetail is 1
		7	Error	If bit 6 of ComponentStateDetail is 1
		8	Status unknown / error in higher-level hardware component	• If bits 11 to 14 of ComponentStateDetail have the value 3 • If bits 7 to 10 of ComponentStateDetail have the value 3 or 4
		9	-	-
		10	Inputs/outputs are not available.	• If bits 11 to 14 of ComponentStateDetail have the value 2 • If bits 7 to 10 of ComponentStateDetail have the value 1 or 2 • If bit 18 of ComponentStateDetail is 1 • If bit 19 of ComponentStateDetail is 1

Parameter	Data type	Value	Description	Rule for forming the parameter
Component-StateDetail	DWORD	Bit array	Status: • Bit 0 to 15: Status of the hardware component • Bit 16 to 31: Status of the CPU	-
		0 to 2 (Enum)	Additional information: • 0: No additional information • 1: Transfer not permitted	-
		3	Bit 3 = 1 if at least one qualified diagnostics is available	-
		4	Bit 4 = 1: Maintenance required for at least one channel or the component	-
		5	Bit 5 = 1: Maintenance demanded for at least one channel or the component	-
		6	Bit 6 = 1: Error in at least one channel or the component	-
		7 to 10 (Enum)	• 0: Own - Hardware component is available • 1: Application Ready Pending - No I/O access • 2: Locked by Superordinated - No access to hardware component • 3: Locked by IOController - No access to hardware component • 4: Locked by IOSupervisor - No access to hardware component	-
		11 to 14 (Enum)	• 0 (bit 11 = 0, bit 12 = 0, bit 13 = 0, bit 14 = 0): OK - The correct element exists. • 1 (bit 11 = 1, bit 12 = 0, bit 13 = 0, bit 14 = 0): Substitute - The element was replaced by a compatible element. • 2 (bit 11 = 0, bit 12 = 1, bit 13 = 0, bit 14 = 0): Wrong - The element was replaced by an incompatible element (The element is reachable, but not ready for operation.). • 3 (bit 11 = 1, bit 12 = 1, bit 13 = 0, bit 14 = 0): No Submodule - The element does not exist.	-
		15	Reserved (always "1").	-
		16 to 31	Status information for modules generated by the CPU: • Bit 16 = 1: Submodule or module disabled • Bit 17 = 1: CiR operation active	-

Parameter	Data type	Value	Description	Rule for forming the parameter
			• Bit 18 = 1: Input not available • Bit 19 = 1: Output not available • Bit 20 = 1: Overflow diagnostics buffer • Bit 21 = 1: Diagnostics not available • Bit 22 = 1: Partial device failure (with PNIO for Shared iDevices) • Bit 23 - 31: Reserved (always 0)	
OwnState	UINT	Enum	Maintenance status	OwnState depends on MaintenanceState.
		0	Good	If MaintenanceState has the value 0.
		1	Disabled	If MaintenanceState has the value 1.
		2	Maintenance required	If MaintenanceState has the value 5.
		3	Maintenance demanded	If MaintenanceState has the value 6.
		4	Error	If MaintenanceState has the value 7.
		5	Not accessible	If MaintenanceState has the value 8.
		6	Diagnostic status unknown	Default value for components that do not return diagnostic events.
		7	IO not available - Inputs/outputs are not available.	If MaintenanceState has the value 10.

Parameter	Data type	Value	Description	Rule for forming the parameter
IOState	WORD	Bit array	I/O status of the hardware component	One or more bits can be set. IOState depends on ComponentStateDetail.
		Bit 0 = 1	Good	• Bit 15 of ComponentStateDetail is always 1. • Bits 11 to 14 of ComponentStateDetail can have the value 0 or 1. • Bit 3 of ComponentStateDetail can be 1. • All other bits of ComponentStateDetail are 0. If bit 0 = 1, bits 1 to 5 and bit 7 = 0.
		Bit 1 = 1	Disabled	• If bit 16 of ComponentStateDetail is 1 • If bit 17 of ComponentStateDetail is 1 If bit 1 = 1, all other bits are 0.
		Bit 2 = 1	Maintenance required	If bit 4 of ComponentStateDetail is 1
		Bit 3 = 1	Maintenance demanded	If bit 5 of ComponentStateDetail is 1
		Bit 4 = 1	Error	If bit 6 of ComponentStateDetail is 1
		Bit 5 = 1	Hardware component not reachable - Hardware component cannot be accessed.	• If bits 11 to 14 of ComponentStateDetail have the value 3 • If bits 7 to 10 of ComponentStateDetail have the value 3 or 4 If bit 5 = 1, all other bits are 0.
		Bit 6 = 1:	Qualified: Bit 6 = 1 if at least one qualified diagnostics is available	If bit 3 of ComponentStateDetail is 1 Bit 0, 2, or 3 is set in this case.
		Bit 7 = 1:	I/O data not available - I/O data not available	• If bits 11 to 14 of ComponentStateDetail have the value 2 • If bits 7 to 10 of ComponentStateDetail have the value 1 or 2 • If bit 18 of ComponentStateDetail is 1 • If bit 19 of ComponentStateDetail is 1
		8 to 14	Reserved (always = 0)	-
		Bit 15	Network/hardware fault • S7-1200: Reserved (always = 0) • S7-1500: If bit 4 = 1 or bit 5 = 1: – Bit 15 = 0: Network error – Bit 15 = 1: Hardware error	-

Parameter	Data type	Value	Description	Rule for forming the parameter
OperatingState	UINT	Enum		-
		0	Not Supported - for I/O modules OperatingState always has the value 0.	-
		1	STOP / firmware update	-
		2	STOP / Memory reset	-
		3	STOP / self start	-
		4	STOP	-
		5	Memory reset	-
		6	Startup	-
		7	-	-
		8	RUN	-
		9	RUN-Redundant	-
		10	HOLD	-
		11	-	-
		12	-	-
		13	DEFECT (Note: can only be seen in diagnostics buffer entries)	-
		14	-	-
		15	De-energized (Note: can only be seen in diagnostics buffer entries)	-
		16	CiR	-
		17	STOP without ODIS	-
		18	RUN ODIS	-
		19	PgmTest	-
		20	RunPgmTest (state of primary CPU when the backup CPU is in test mode)	-
		21	Run-Syncup	-
		22	SYNCUP (only backup CPU in SYNCUP system state)	-
		31	State of partner CPU unknown (for example when the partner CPU is not available)	-
		32	-	-
		33	STOP system state	-
		34	Reserved	-
		35	STARTUP system state	-
		36	Reserved	-
		37	RUN-Solo system state	-
		38	SYNCUP system state	-
		39	Reserved	-
		40	RUN-Redundant system state	-

DNN structure

With parameter MODE = 2, the diagnostic information details are output in accordance with the DNN structure. In this case, enter the system data type "DNN" as data type for the tag declaration.

The following table shows the meaning of the individual parameter values.

Parameter	Data type	Value	Description
SubordinateState	UINT	Enum	Status of the subordinate component (see parameter OwnState of the DIS structure)
SubordinateIOState	WORD	Bitarray	Status of the inputs and outputs of the subordinate component (see parameter IO State of the DIS structure)
DNNmode	WORD	Bitarray	• Bit 0 = 0: Diagnostics enabled • Bit 0 = 1: Diagnostics disabled • Bit 1 to 15: Reserved

Parameter RET_VAL

Error code* (W#16#...)	Description
0	No error
n	The data area at the DETAIL parameter is too small. Not all details of the diagnostics data are output.
8080	Value in the MODE parameter is not supported.
8081	Type at the DIAG parameter is not supported with the selected mode (parameter MODE).
8082	Type at the DETAIL parameter is not supported with the selected mode (parameter MODE).
8090	LADDR does not exist
80C1	Insufficient resources for parallel execution.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you read out the diagnostic information of a CPU.

Create four tags and a "myDIAG" structure (with the DIS data type) in a global data block for storing the data.

My_gDB_GET_DIAG			
	Name	Data type	Start value
1	Static		
2	diagMODE	UInt	1
3	myLADDR	HW_ANY	50
4	returnValue	Int	0
5	CountDiagDetails	UInt	0
6	myDIAG	DIS	
7	MaintainanceState	DWord	16#0
8	ComponentStateDetail	DWord	16#0
9	OwnState	UInt	0
10	IOState	Word	16#0
11	OperatingState	UInt	0

Interconnect the parameters of the instruction as follows.

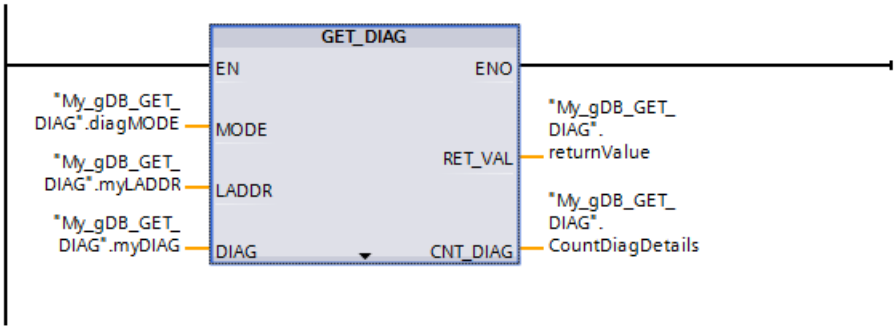

The HW identifier of the CPU is made known to the "GET_DIAG" instruction through parameter LADDR ("myLADDR"). According to the value "1" of parameter MODE ("diagMODE"), the following applies:

- The instruction reads the status of the addressed hardware object (of the CPU).
- At parameter DIAG ("myDIAG"), the diagnostic information is output in a structure (DIS data type).

To understand the diagnostic information, the hex values must be converted to binary code. At parameter DIAG ("myDIAG"), the following is indicated:

- MaintenanceState: According to the value "0", the CPU requires no maintenance.
- ComponentStateDetail: According to the hex value "0000_8000", bit 15 is active.
- OwnState: According to the value "0", no fault has occurred.
- IOState: According to the hex value "0001", there is no maintenance required.
- OperatingState: Outputs "0".

The output parameter RET_VAL ("returnValue") indicates that processing took place without errors. At output parameter CNT_DIAG ("CountDiagDetails"), it is indicated that "0" diagnostic details of parameter DETAIL have been output.

My_gDB_GET_DIAG

	Name	Data type	Start value	Monitor value
1	Static			
2	diagMODE	UInt	1	1
3	myLADDR	HW_ANY	50	50
4	returnValue	Int	0	0
5	CountDiagDetails	UInt	0	0
6	myDIAG	DIS		
7	MaintainanceState	DWord	16#0	16#0000_0000
8	ComponentStateDetail	DWord	16#0	16#0000_8000
9	OwnState	UInt	0	0
10	IOState	Word	16#0	16#0001
11	OperatingState	UInt	0	0

Note: For example, you could individually read out bit 3 (channel diagnostics yes/no) from the ComponentStateDetail tag.

- Address the bit as follows: ComponentStateDetail.%X3

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Example - Diagnostics of an IO device

In the following example, you read out the diagnostic information of an IO device. For diagnostics, you use the device view of the IO device and the instruction GET_DIAG.

Diagnostics in the device view

One of the IO device modules signals an error. The device status changes and is displayed as follows:

The module and the head module of the IO device show a group status in the device view. An error is therefore displayed for both.

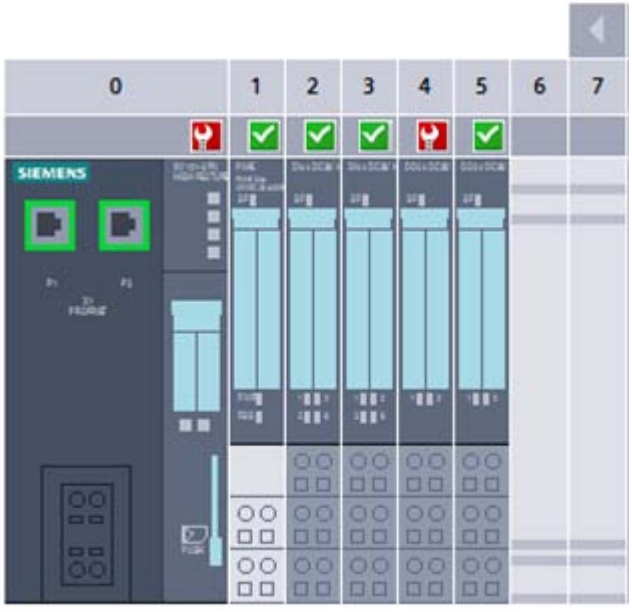

Diagnostics with the GET_DIAG instruction

To read out the diagnostic information of the IO device, address the head module of the IO device (via LADDR).

- In mode 1 (MODE has the value 1), you read out the status of the IO device.
- In mode 2 (MODE has the value 2), you read out the status of the subordinate modules.

Call MODE 1: Self-diagnostics	Call MODE 2: Summary of the subordinate diagnostics
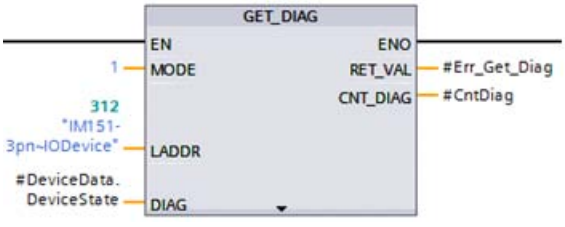	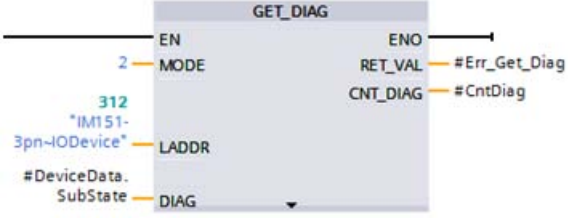

LADDR (HW ID) = IM151-3pn~IODevice
DIAG (DIS structure) = #DeviceData.DeviceState
DIS.OwnState = 0 (no error)
DIS.IOState = bit sequence 0000 0000 0000 0001
	LADDR (HW ID) = IM151-3pn~IODevice
	DIAG (DNN structure) = #DeviceData.SubState
	DNN.SubordinateState = 4 (error)
	DNN.SubordianteIOState = bit sequence 1000 0000 0001 0001

Result

Unlike the device view, the GET_DIAG instruction does not return the group status of the IO device. The error in the subordinate module is not read until mode 2.

See also

Switching display formats in the program status (Page 7698)

4.1.4.10 Pulse (S7-1200)

CTRL_PWM: Pulse-width modulation (S7-1200)

Description

The "CTRL_PWM" instruction is used to enable and disable a pulse output supported by the CPU using the software.

- Enter the hardware ID of the pulse generator that you want to control with the instruction at the input PWM.
- When the bit in the ENABLE input of the instruction is set, the pulse output is enabled.
  - If ENABLE has a value of TRUE the pulse generator will generate a pulse with the properties defined in the device configuration.
  - When the bit in the ENABLE input is reset or the CPU changes to STOP, the pulse output is disabled and no more pulses are generated.

BUSY always has the value of FALSE for S7-1200 since S7-1200 activates the pulse generator if the instruction "CTRL_PWM" is executed.

When the EN enable input has signal state "1" and no errors have occurred during execution of the instruction, the enable output ENO is set.

Note

Use of the force table for PWM and PTO

Digital inputs and outputs that are used for PWM and PTO cannot be forced. Digital inputs and outputs that were assigned via device configuration are not controlled by either the force table or the watch table.

Note

Process image of PTO/PWM outputs

Do not use PTO/PWM outputs in the process image (for example, for access in the user program, for online functions or in the HMI). The update rate of the process image is much slower than the rate of the signal changes. The display in the process image therefore does not reflect the signal flow.

Requirement

A prerequisite for the correct execution of the instruction is that the specified pulse generator is activated in the hardware configuration.

Open the properties of the module in the device view. Go to "Pulse generators (PTO/PWM)", open the desired PTO/PWM and activate the function the "Activate this pulse generator" under "General".

Go to "Assigning parameters" and set the Pulse options.

Note

Pulse output parameters are assigned exclusively in the device configuration and not using the "CTRL_PWM" instruction. Any change of parameters that is intended to have an effect on the CPU must therefore be made while the CPU is in STOP mode. Changing the duration of the pulse is an exception.

Changing the pulse duration from the user program

The pulse duration setting made in the dialog "Pulse options" can be changed from the user program.

The value set for "Initial pulse duration" is written in the output bytes of the pulse generator. The start address and the end address are displayed in the properties of the pulse generator under "I/O-addresses".

To change the pulse duration, write the desired values in the output word address specified in the device configuration.

Example:

- A value of 500 (decimal) is used for the "Initial pulse duration". The start address of the PTO/PWM is "1000" and the end address is "1001".
- The binary value of "0000000111110100" (=500 decimal) is written in both of the output bytes.
  - Start address (AB1000): 0000_0001 (BIN)
  - End address (AB1001): 1111_0100 (BIN)

Please note that the pulse duration always depends on the parameters set for the Pulse duration format (hundredths, thousandths, ...).

Change the cycle time from the user program.

Select the check box "Permit change to cycle time in runtime" in the "Pulse options" of the corresponding pulse generator. When this check box is selected, the first two output bytes contain the pulse duration and the third to sixth output byte contains the cycle time.

During the run time of the pulse generator, you can change the value of the double word at the end of allocated memory of the outputs. This changes the cycle time of the PWM signal.

Example: Select the check box so that the CPU assigns PWM1 six output bytes, and select AB1008 to AB1013 for this. After loading the program into the CPU and starting the pulse generator, you can change the pulse duration by writing AW1008 and the cycle time by writing AD1010.

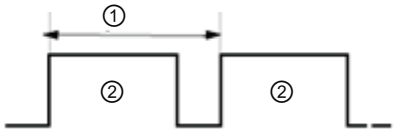

Figure 4-1 PWM_Example

① Cycle time
② Pulse duration

Parameters

The following table shows the parameters of the "CTRL_PWM" instruction:

Parameter	Declaration	Data type	Memory area	Description
PWM	Input	HW_PWM	I, Q, M, D, L or constant	Hardware ID of the pulse generator The hardware ID can be found in the properties of the pulse generator in the Device view. The hardware IDs of the pulse generators are also listed in the system constants.
ENABLE	Input	BOOL	I, Q, M, D, L or constant	The pulse output is enabled when ENABLE = TRUE and disabled when ENABLE = FALSE.
BUSY	Output	BOOL	I, Q, M, D, L	Processing status
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction (see below).

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

Error code* (W#16#...)	Description
0	No errors
80A1	Hardware ID of the pulse generator is invalid.
80D0	Pulse generator with the specified hardware ID is not activated. Activate the pulse generator in the CPU properties under "Pulse generators (PTO/PWM)".
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

CTRL_PTO: Output a pulse sequence with a preset frequency (S7-1200)

Description

The "CTRL_PTO" instruction makes a pulse sequence available to you with a preset frequency. For this you do not require an axis DB of a technological object.

Requirement

The use of the "CTRL_PTO" instruction requires an activated pulse generator. You activate this in the hardware configuration as well as selecting the signal type. You need to specify the pulse generator at the "PTO" parameter and set the "REQ" parameter to TRUE.

Note

Using a PLC pulse generator

At any one time a pulse generator can only be used by one instruction.

Note

Use of the force table with PWM and PTO

Digital inputs and outputs that are used for PWM and PTO cannot be forced. Digital inputs and outputs that were assigned via the device configuration are not controlled by either the force table or the watch table.

Note

Process image of PTO/PWM outputs

Do not use PTO/PWM outputs in the process image (for example, for access in the user program, for online functions or in the HMI). The update rate of the process image is much slower than the rate of the signal changes. The display in the process image therefore does not reflect the signal flow.

Parameter

The following table shows the parameters of the "CTRL_PTO" instruction:

Parameter	Declaration	Data type	Description
REQ	Input	BOOL	• REQ=1: Sets the frequency of the pulse generator to the value of FREQUENCY REQ=1 and FREQUENCY=0: Deactivates the pulse generator • REQ=0: No change at pulse generator
PTO	Input	HW_PTO	Hardware identifier of the pulse generator This can be taken from the properties of the pulse generator in the device view or the list of system constants.
FREQUEN-CY	Input	UDINT	Frequency of the pulse sequence to be output in Hz
DONE	Output	BOOL	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	Processing status Since the S7-1200 enables the pulse generator when the "CTRL_PTO" instruction is executed, BUSY always has the value FALSE with the S7-1200.

Parameter	Declaration	Data type	Description
ERROR	Output	BOOL	Status parameter • 0: No error • 1: An error occurred during execution of the instruction.
STATUS	Output	WORD	Status of the instruction (see below)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter Status

Error code* (W#16#...)	Explanation
0	No error
8090	The pulse generator with the specified hardware ID is already being used.
8091	Range exceeded for the "FREQUENCY" parameter; for more detailed information, refer to the manual.
80A1	The PTO parameter does not address a hardware ID of a pulse generator.
80D0	The pulse generator with the specified hardware ID is not activated or does not have the "PTO" property. Activate the pulse generator in the CPU properties under "Pulse generators (PTO/PWM)" and select a signal type "PTO".
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

4.1.4.11 Recipes and data logging

Recipe functions

Recipe functions - overview

Description

With the instructions "RecipeExport" and "RecipeImport" you transfer recipe data

- from a recipe data block to a CSV file on the memory card of the CPU ("RecipeExport")
- from a CSV file on the memory card of the CPU to a recipe data block ("RecipeImport")

Note

Service life of memory cards

The service life of memory cards is limited. Additional information is available here: AUTOHOTSPOT

Protection of recipe data

The instructions "RecipeExport" and RecipeImport" are not suitable for applications in which special protection is necessary for the recipe data. For higher protection of the data use the recipe functions of HMI.

Position of the recipe data block with "RecipeExport"

The data block with the recipe data exported with "RecipeExport" can be located in work or load memory. If the data block is both in the work memory and in the load memory, the values from the work memory are used.

Delimiters

You can use different delimiters to separate the row values in the recipe. Note that if you open the CSV file with the exported recipe data, for example in Excel, commas can be replaced by periods and vice versa. The reason for this is that different languages use different decimal separators. This can have an effect when importing CSV files. The following convention applies for "RecipeImport"; the first detected possible delimiter is assumed to be the delimiter for all values of a table row.

- Example 1: If a semicolon is detected as the first delimiter, during import it is assumed that the semicolon is used as the delimiter for all further values. As the decimal delimiter with a value of the type REAL, a comma is then used.
- Example 2: If a comma is detected as the first delimiter, during import it is assumed that the comma is used as the delimiter for all further values. As the decimal delimiter with a value of the type REAL, a period is then used.

If the recipe data record contains a character string, make sure that the character string does not contain the delimiter you are using. With character strings, also make sure that they do not contain any control characters that specify a line break (with ASCII: LF, CR LF, CR).

Note: Spaces and tabs are not used as delimiters. These lead to error code 80B2 during import.

Note

Asynchronous processing of the recipe functions

Because the recipe functions are executed asynchronously, you must keep the data in the source area consistent until the DONE parameter or the ERROR parameter changes to the value "1".

RecipeExport: Exporting recipes (S7-1200, S7-1500)

Description

The "RecipeExport" instruction exports the recipe data from a data block to a CSV file in the load memory.

With S7-1500 CPUs, the DB that is to be exported can be located either in the work memory or in the load memory. If the data block is in both the work memory and the load memory, the values from the work memory are exported. For S7-1200 CPUs, the DB to be exported must be present in the load memory.

The CSV file is created in the load memory. For S7-1500 CPUs, this is the memory card; for S7-1200 CPUs the internal or external load memory (if a memory card is present).

During the export, the CSV file is created in the "Recipes" folder in the main directory of the load memory. The name of the data block is used as file name of the created CSV file. If a CSV file with an identical name already exists, it is overwritten during the export.

Each recipe data record generates a line in the CSV file. These lines have the following structure:

- The first line contains the "Index" entry and the name of the recipe components.
- Each additional line contains the number of the recipe data record (index) and the associated components.

When you take the data types of the recipe DB, the maximum size of a data record is 5000 bytes. If this is exceeded, the error code W#16#8092 is returned. This restriction to 5000 bytes also applies to the size of the header. If the header is greater than 5000 bytes, it is exported up to the 5000 byte limit with an S7-1500 CPU and the export is continued with the first recipe data record. An S7-1200 CPU rejects the export entirely in this case and returns the error code W#16#8092.

The S7-1200 CPUs also monitor the number of columns. These must not exceed 255 (whereby the index column is not counted).

If the recipe DB is a DB with standard access and consists of more components than are permitted (for example of two ARRAYs and one structure), only the first ARRAY is exported and STATUS is set to W#16#0000.

Functional description

The export is triggered by the "REQ" parameter. The BUSY parameter is set to "1" during the export. After the execution of the instruction, BUSY is reset to "0" and the completion of the operation is indicated with "1" at the DONE parameter.

If an error occurs during execution, this is signaled by the parameters ERROR and STATUS.

Parameters

The following table shows the parameters of the "RecipeExport" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Control parameter REQUEST: Activates the export on a rising edge.
RECIPE_DB	InOut	VARIANT	D	Pointer to the recipe data block. For information on the structure of the data block, refer to: Structure of a recipe DB (Page 2654)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: The instruction is not executed. • 1: The instruction is executed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See the "STATUS" parameter table.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

Error code* (W#16#...)	Explanation
0	No error occurred
7000	No job processing active
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8090	File name of the CSV file contains invalid characters. The file name of the CSV file is identical to the name of the data block.
8091	The data structure referenced by RECIPE_DB cannot be processed or the number of data columns in a S7-1200 CPU is greater than 255. Some information may have been written to the CSV file.
8092	Data structure at RECIPE_DB parameter exceeds 5000 bytes. Additionally for S7-1200 CPUs: The size of the header is greater than 5000 bytes. The export is rejected.
80B3	Insufficient memory space on the memory card or the internal load memory.
80B4	The memory card is write-protected.
80B6	With (S7-1200 CPUs only) attribute "Save only in load memory", the recipe data block is not activated.

Error code* (W#16#...)	Explanation
80C0	CSV files is temporarily locked.
80C1	Recipe data block temporarily disabled.
80C3	Too few resources
8624	RECIPE_DB points to an impermissible area, for example to local data.
8652	The tag at RECIPE_DB is insufficient. No information was written to the CSV file.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You will find the example here: Example program for recipe functions (Page 2657).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Difference between synchronous and asynchronous instructions (Page 596)

RecipeImport: Importing recipes (S7-1200, S7-1500)

Description

The "RecipeImport" instruction imports the recipe data from a CSV file of the load memory into the data block at the RECIPE_DB parameter.

For S7-1500 CPUs, the CSV file is on the memory card; for S7-1200 CPUs, it is in the internal or external load memory (if a memory card is present).

The values in the data block are overwritten in the process. With S7-1500 CPUs, the DB to which the recipe data is to be imported can be located either in the work memory or in the load memory. If it is in both the work memory and the load memory, the values are imported to the data block located in the work memory. For S7-1200 CPUs, the DB into which the recipe data is to be imported must be located in the load memory.

The recipe is only maintained permanently if it is imported to a load memory DB. If it is imported to a work memory DB but you need it permanently, you must transfer the recipe to a load memory DB after the import with the help of the WRIT_DBL instruction.

Note the following when importing the CSV file:

- The CSV file must be located in the "Recipes" directory of the load memory.
- The name of the CSV file must match the name of the data block at the RECIPE_DB parameter.

- Each line of the CSV file corresponds to one recipe data record. The first line (header) of the CSV file contains the name of the recipe components (see also: Structure of a recipe DB (Page 2654)). The first line is ignored completely during the import. The names of the recipe components in the CSV file and the data block are not synchronized during the import, and the number of entries is also irrelevant.
- In each case the first value in each line of the CSV file is the index number of the recipe. The individual recipes are imported in the order of the index. For this, the index in the CSV file has to be in ascending order and may contain no gaps (if this is not the case, the error message 80B0 is output at the STATUS parameter).
- The CSV file may not contain more recipe data records than provided for in the data block. The maximum number of data records is indicated by the array limits in the data block.
- If a value of the CSV file does not fit in the associated data type of the DB, the import is rejected. If necessary, character strings are truncated after their maximum permissible length.
  If the data type in the DB is BOOL type, an S7-1500 CPU accepts all integer values. The value 1 is then set to TRUE and all other values are set to FALSE. An S7-1200 CPU accepts only the values 0 and 1, all other values are rejected.
- If the recipe DB is a DB with standard access and consists of more components than are permitted (for example of two ARRAYs and one structure), the recipe data is only imported into the first ARRAY and STATUS is set to W#16#0000.

Functional description

The import is triggered by the "REQ" parameter. The BUSY parameter is set to "1" during the import. After the execution of the instruction, BUSY is reset to "0" and the completion of the operation is indicated with "1" at the DONE parameter.

If an error occurs during execution, this is signaled by the parameters ERROR and STATUS.

Parameters

The following table shows the parameters of the "RecipeImport" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Control parameter REQUEST: Activates the import on a positive edge.
RECIPE_DB	InOut	VARIANT	D	Pointer to the recipe data block. For information on the structure of the data block, refer to: Structure of a recipe DB (Page 2654)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: The instruction is not executed. • 1: The instruction is executed.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See the "STATUS" parameter table.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

Error code* (W#16#...)	Explanation
0	No error occurred
7000	No job processing active
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8090	The file name contains invalid characters.
8091	The data structure referenced with RECIPE_DB cannot be process, for example, because the data block does not exist.
8092	No matching CSV file found for the import. Possible cause: The name of the CSV file does not match the name of the recipe DB.
80C0	CSV file is temporarily locked.
80C1	Recipe data block is temporarily locked.
80B0	Numbering in the index of the CSV file is not continuous, not ascending or exceeds the maximum number (array limit) in the data block.
80B1	Structure of the recipe data block and the CSV file do not match: The CSV file contains too many fields.
80B2	Structure of the recipe data block and the CSV file do not match: The CSV file contains too few fields. Note: The lines of the CSV file are read one after the other. If a line of the CSV file contains more characters than expected by the CPU, the error code W#16#80B2 is returned regardless of how many fields were detected.
80B6	With (S7-1200 CPUs only) attribute "Save only in load memory", the recipe data block is not activated.
80C3	Too few resources
80D0 +n	Structure of the recipe data block and the CSV file do not match: Data type in field n does not match (n<=46).
80FF	Structure of the recipe data block and the CSV file do not match: Data type in field n does not match (n>46).
8624	RECIPE_DB points to an impermissible area, for example to local data.
8630	Target range is write-protected, for example a write-protected DB.
8652	The tag at RECIPE_DB is insufficient. The data block was not written.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Example program for recipe functions (Page 2657).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Difference between synchronous and asynchronous instructions (Page 596)

Structure of a recipe DB

Introduction

The following sections describes the structure of a recipe DB based on a simple example. The recipe DB consists of five data records, three of which are used. The fourth and fifth data records are left free for later expansions. Each data record contains one recipe, which is made up of the recipe name and eight ingredients.

product-name	water	barley	wheat	hops	yeast	wa-terTmp	mashTmp	mash-Time	QTest
Pils	10	9	3	280	39	40	30	100	0
Lager	10	9	3	150	33	50	30	120	0
BlackBeer	10	9	3	410	47	60	30	90	1
Not_used	0	0	0	0	0	0	0	0	0
Not_used	0	0	0	0	0	0	0	0	0

Structure of the recipe data block

The recipe data are implemented as follows in a global data block:

- The template for all recipes is the PLC data type "Beer_Recipe" with the recipe components "procutname", "water", etc. with the corresponding data types.
- In a global data block, the PLC data type is used as Array [1.. 5] of "Beer_Recipe". The array limits (1 to 5 in this case) indicate the maximum number of recipes that the DB may contain.
- The values for the recipe components are added as start value in the data block.
- The global DB is interconnected with the instruction via the InOut parameter RECIPE_DB.

Recipe_DB				
	Name	Data type	Offset	Start value
1	Static			
2	Products	Array [1 .. 5] of "Beer_Recipe"	...	
3	Products[1]	"Beer_Recipe"	...	
4	Products[2]	"Beer_Recipe"	...	
5	Products[3]	"Beer_Recipe"	...	
6	productname	String[20]	...	'BlackBeer'
7	water	UInt	...	10
8	barley	UInt	...	9
9	wheat	UInt	...	3
10	hops	UInt	...	410
11	yeast	UInt	...	47
12	waterTmp	UInt	...	60
13	mashTmp	UInt	...	30
14	mashTime	UInt	...	90
15	QTest	UInt	...	1
16	Products[4]	"Beer_Recipe"	...	
17	Products[5]	"Beer_Recipe"	...	

Exporting to CSV file

After the execution of the "RecipeExport (Page 2649)" instruction the data of the DB are written to a CSV file with the following structure:

Recipe_DB.csv

```
index,productname,water,barley,wheat,hops,yeast,waterTmp,mashTmp,mashTime,QTest
1,”Pils”,10,9,3,280,39,40,30,100,0
2,”Lager”,10,9,3,150,33,50,30,120,0
3,”BlackBeer”,10,9,3,410,47,60,30,90,1
4,”Not_used”,0,0,0,0,0,0,0,0,0
5,”Not_used”,0,0,0,0,0,0,0,0,0
```

Display in Excel

The CSV file can be opened in Excel to make reading and editing easier. If the commas are not recognized as separators when you open the file, use the Excel import function to output the data in structured form:

	A	B	C	D	E	F	G	H	I	J	K
1	index	product	water	barley	wheat	hops	yeast	waterTmp	mashTmp	mashTime	QTest
2	1	"Pils"	10	9	3	280	39	40	30	100	0
3	2	"Lager"	10	9	3	150	33	50	30	120	0
4	3	"BlackBeer"	10	9	3	410	47	60	30	90	1
5	4	"Not_used"	0	0	0	0	0	0	0	0	0
6	5	"Not_used"	0	0	0	0	0	0	0	0	0

Editing the CSV file

The CSV file is uploaded to the PC/programming device by means of the Web server and edited. After editing, the modified file can be reloaded in the CPU. You must delete the existing CSV file for this.

You can use the "RecipeImport (Page 2651)" instruction to re-import the modified data from the CSV file into the data block.

Note that the modified data must still be compatible with the data block. In other words:

- No changes may be made to the structure in the table (for example, by adding ingredients in a new column).
- If you add data records to the file, make sure when importing into the data block that the array limits through which you specify the maximum number of data records correspond to at least the number of data records.
- An index is automatically generated during export to the CSV file. If you create additional data records, add consecutive index numbers accordingly.
- The values in the table cells must correspond to the data types used in the data block with respect to format as well as length.
  - Example 1: If INT data type was used in the data block, use only integers in the table.
  - Example 2: If SINT data type was used in the data block, use only integers with the values -128 to +127 in the table.

  Always keep in mind the permitted data types and data areas as listed in the table below when you make changes in the table.

Data type		Format	Note
Floating-point numbers	LReal	+9.9999999999999E+999	Always written exponentially
	Real	+9.9999999E+99	Always written exponentially
Signed integers	LInt	+9999999999999999999	Value range of signed integers: -9223372036854775808 .. +9223372036854775807
	DInt	+9999999999	Value range of integers: -2147483648 to +2147483647
	Int	+99999	Value range of integers: -32768 to +32767
	SInt	+999	Value range of integers: -128 to +127
Unsigned integers	ULInt	+99999999999999999999	Value range of integers: 0 to +18446744073709551615
	UDInt	+9999999999	Value range of integers: 0 to +4294967295
	UInt	+99999	Value range of integers: 0 to +65535
	USInt	+999	Value range of integers: 0 to +255
Binary numbers	LWord	+99999999999999999999	Value range of integers: 0 to +18446744073709551615
	DWord	+9999999999	Value range of integers: 0 to +4294967295
	Word	+99999	Value range of integers: 0 to +65535
	Byte	+999	Value range of integers: 0 to +255
	Bool	9	Value range: 0 or 1

Data type		Format	Note
Date and time	LTIME	dddddd:hh:mm:ss. 999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	TIME	hhh:mm:ss.999	ISO format with milliseconds
	S5TIME	hhh:mm:ss.999	ISO format with milliseconds
	LDT	YYYY-MM-DD hh:mm:ss. 999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	DTL	YYYY-MM-DD hh:mm:ss. 999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	DT	YYYY-MM-DD hh:mm:ss.999	ISO format with milliseconds
	DATE	YYYY-MM-DD	ISO format
	LTime_Of_ Day	hh:mm:ss.999_999_999	ISO format
	TOD	hh:mm:ss.999	ISO format
Characters	WString	"abcd"	• Character string inside double quotation marks. When a recipe DB is imported, the double quotation marks may be omitted. • Current length • A character string of data type WString consists of elements of the WChar data type. When a recipe DB is exported, the contents of characters of the WChar data type are limited to 16#FF (example: 16#1255 is changed to 16#55).
	String	"abcd"	• Character string inside double quotation marks. When a recipe DB is imported, the double quotation marks may be omitted. • Current length
	WChar	"a"	Individual character inside double quotation marks. When a recipe DB is imported, the double quotation marks may be omitted. When a recipe DB is exported, the contents of characters of the WChar data type are limited to 16#FF (example: 16#1255 is changed to 16#55).
	Char	"a"	Individual character inside double quotation marks. When a recipe DB is imported, the double quotation marks may be omitted.

Example program for recipe functions

In the following example, you export the recipe data from a data block (recipe DB) to a CSV file and import new recipe data into the data block.

Requirements

As the template for the recipe data create a PLC data type "Beer_Recipe" an.

Beer_Recipe			
	Name	Data type	Default value
1	productname	String[20]	''
2	water	UInt	0
3	barley	UInt	0
4	wheat	UInt	0
5	hops	UInt	0
6	yeast	UInt	0
7	waterTmp	UInt	0
8	mashTmp	UInt	0
9	mashTime	UInt	0
10	QTest	UInt	0

To store all recipe data records, create a recipe DB. You will find further information on creating a recipe DB in "Structure of a recipe DB (Page 2654)".

Recipe_DB			
	Name	Data type	Start value
1	Static		
2	Products	Array[1..5] of "Beer_Recipe"	
3	Products[1]	"Beer_Recipe"	
4	Products[2]	"Beer_Recipe"	
5	Products[3]	"Beer_Recipe"	
6	productname	String[20]	'Blackbeer'
7	water	UInt	10
8	barley	UInt	9
9	wheat	UInt	3
10	hops	UInt	410
11	yeast	UInt	47
12	waterTmp	UInt	60
13	mashTmp	UInt	30
14	mashTime	UInt	90
15	QTest	UInt	1
16	Products[4]	"Beer_Recipe"	
17	Products[5]	"Beer_Recipe"	

To store the data for the interconnection, create twelve tags in a global data block.

My_gDB_Recipefunctions			
	Name	Data type	Start value
1	Static		
2	expREQ	Bool	false
3	expDONE	Bool	false
4	expBUSY	Bool	false
5	expERROR	Bool	false
6	expSTATUS	Word	16#0
7	impREQ	Bool	false
8	impDONE	Bool	false
9	impBUSY	Bool	false
10	impERROR	Bool	false
11	impSTATUS	Word	16#0
12	outputErr_EXP	Byte	0
13	outputErr_IMP	Byte	0

To display and store a CSV file, use the Web server of the CPU. In order to use the web server and theCSV file, the following is necessary:

- The web server must be activated in the CPU settings.
- A user profile with the Web server properties of the CPU must be set up with the following settings:
  - The user must be able to read files.
  - The user must be able to write/delete files.

Interconnection of the parameters

The instructions are called in a function block.

Network 1: Interconnect the parameters of the "RecipeExport" instruction as follows.

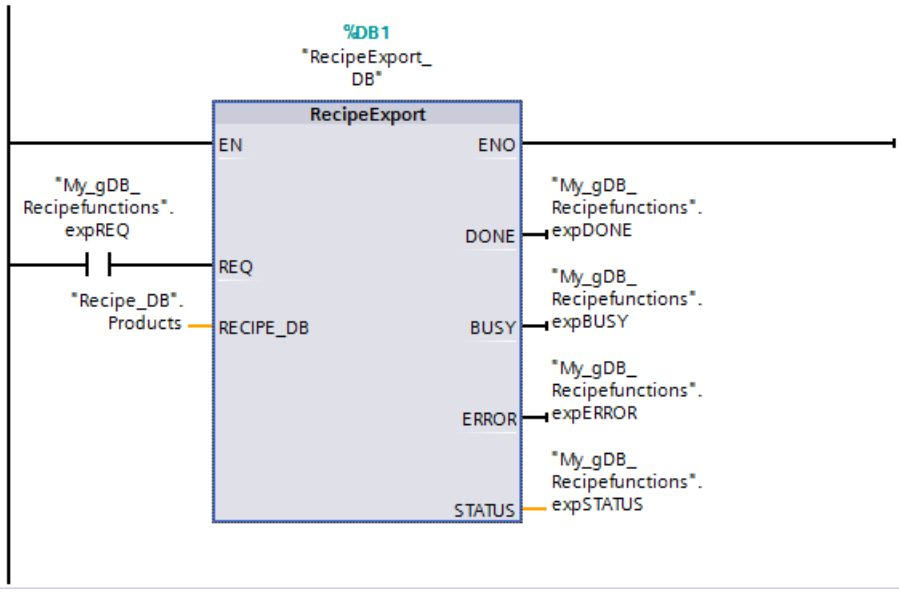

Network 3: Interconnect the parameters of the "RecipeImport" instruction as follows.

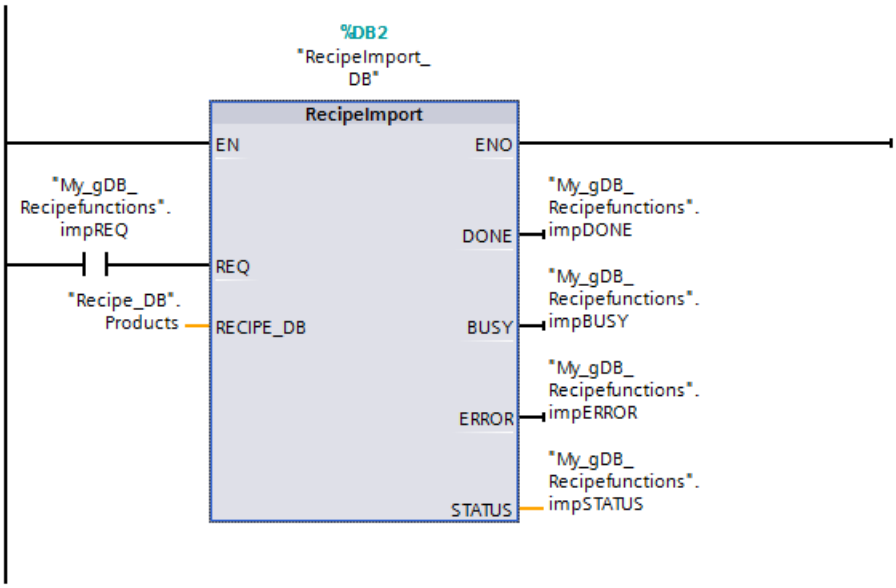

Network 2 and 4: Since the parameters STATUS ("expSTATUS" and "impSTATUS") are only valid for one cycle, record the parameters with a "MOVE" instruction. At the output parameters OUT1 ("outputErr_EXP" or "outputErr_IMP") output the status in case of an error.

Interconnect the parameters in network 2 as follows.

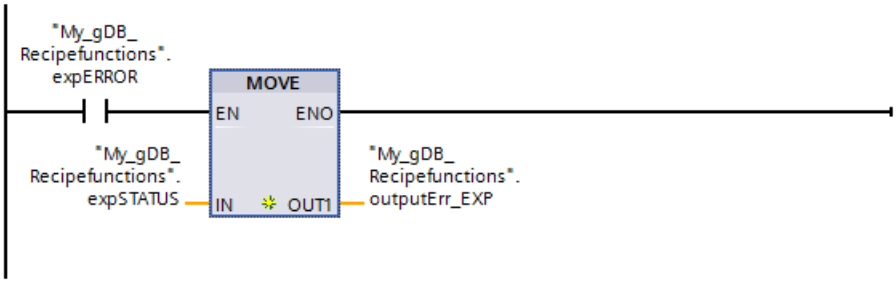

Interconnect the parameters in network 4 as follows.

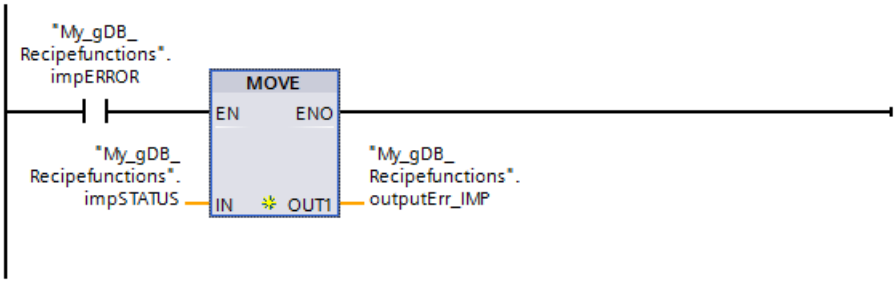

"RecipeExport" instruction

The recipe DB to be exported needs to be interconnected at the RECIPE_DB parameter. If the REQ ("expREQ") parameter has the signal status "TRUE" export is started. From the data of the recipe DB, a CSV file is generated and stored in the recipe folder in the CPU memory. At output parameter STATUS ("expSTATUS") it is displayed that a signal change has taken place. Then, it is indicated that no further processing is taking place (the value is "7000").

At output parameter ERROR ("expERROR"), it is indicated that the processing of the instruction was error-free.

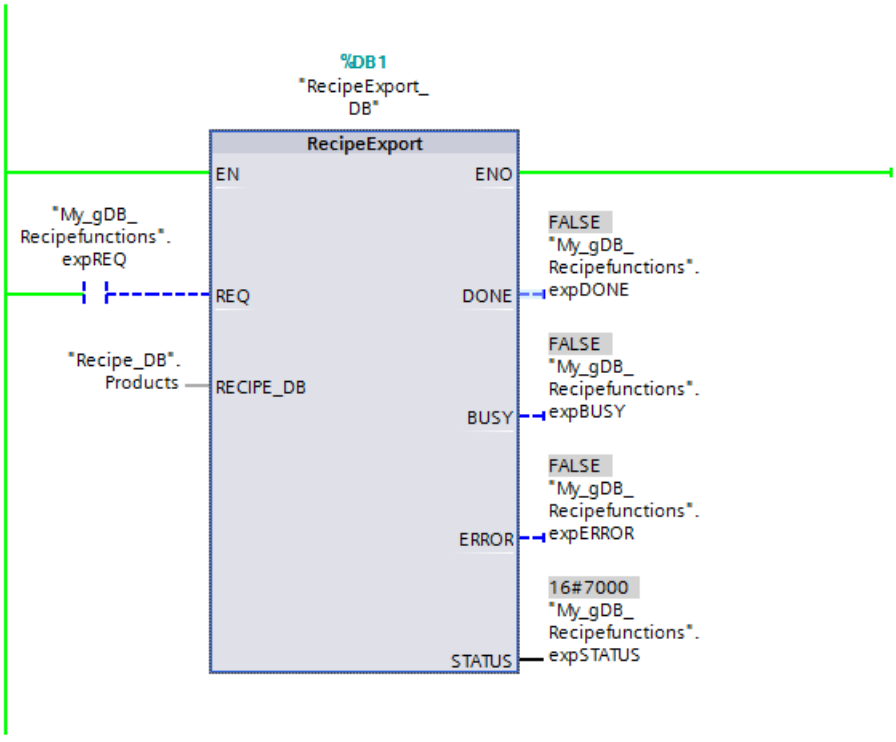

"RecipeImport" instruction

The recipe DB to be imported needs to be interconnected at the RECIPE_DB parameter. If the REQ ("impREQ") parameter has the signal status "TRUE" export is started. In the existing recipe DB, all the recipe data is loaded from the corresponding CSV file-that is located in the recipe folder of the CPU memory. At output parameter STATUS ("impSTATUS") it is displayed that a signal change has taken place. Then, it is indicated that no further processing is taking place (the value is "7000").

At output parameter ERROR ("impERROR"), it is indicated that the processing of the instruction was error-free.

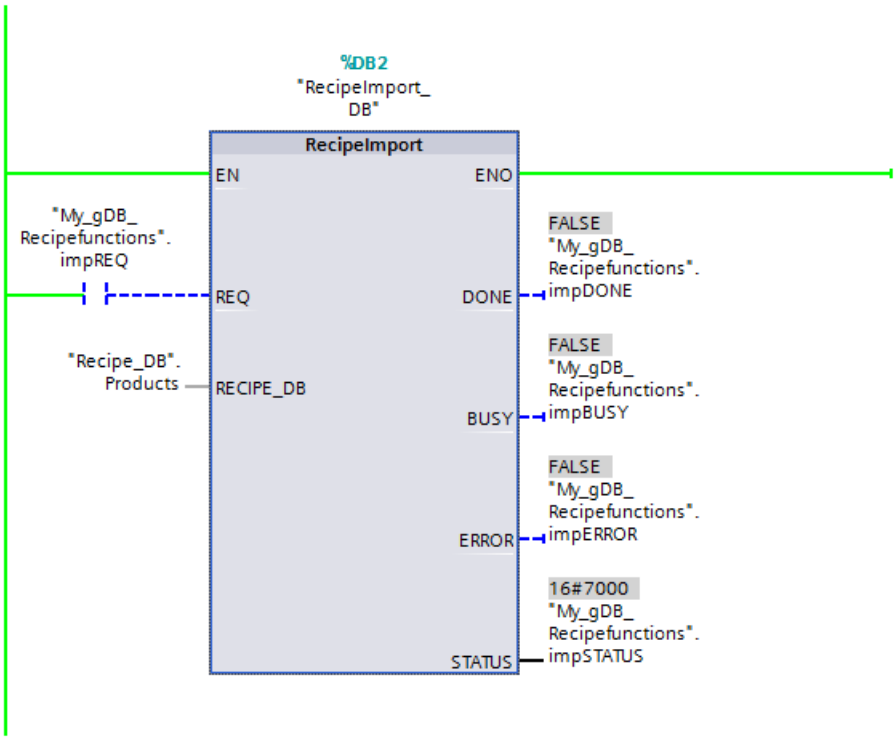

Displaying and saving the CSV file

Opening the Web server

Use the Internet browser to open the Web server (via the IP address of the CPU) and log on in the Web server menu.

Via the "File browser" tab open the "Recipes" folder in the CPU memory.

The CPU outputs the CSV file as soon as the "RecipeExport" instruction executes. To have the current status of the CSV file displayed, click on the "Update automatically" button. Below the function is displayed as "On".

Editing the CSV file

To edit the CSV file, you have the following options:

- You open the CSV file in the Web server.
- You save the CSV file on the PG/PC.
  - **Note**: To be able to load the edited CSV file from the PC in the CPU memory: First delete the CSV file with the same name in the CPU memory

/Recipes/

Name	Size	Changed	Delete	Rename
..				
Recipe_DB.csv	511	03:09:32 pm 10/06/2014		

  - After editing, load the CSV file in the CPU memory using the Web server.

Directory operations:

Browse... No file selected. Upload file

Changing component values during runtime

To change component values of the recipe DB during runtime (e.g. Recipe_DB.Product[4]) use a watch table.

	i	Name	...	Display format	Monitor value	Modify value
1		"Recipe_DB".Products[4].productname		String	'Not_used'	'Mix'
2		"Recipe_DB".Products[4].water		DEC	0	10
3		"Recipe_DB".Products[4].barley		DEC	0	2
4		"Recipe_DB".Products[4].wheat		DEC	0	2
5		"Recipe_DB".Products[4].hops		DEC	0	4
6		"Recipe_DB".Products[4].yeast		DEC	0	30
7		"Recipe_DB".Products[4].waterTmp		DEC	0	60
8		"Recipe_DB".Products[4].mashTmp		DEC	0	60
9		"Recipe_DB".Products[4].mashTime		DEC	0	10
10		"Recipe_DB".Products[4].QTest		DEC	0	0

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

Data logging

Data logging - Overview

Saving process values

The data logging instructions are used in the user program to save process values to data logs. Data logs can be saved on the Memory Card (MC) or in the internal load memory. The data logs are saved in CSV format (Comma Separated Value).

NOTICE
Special characters in character strings If you use character strings within the data log, it is strongly recommended that you do not use the delimiter of the CSV file within these character strings. If you ignore this recommendation, opening the CSV file e.g. with MS Excel may lead to problems. Instead of the delimiter of the CSV file, use a different character such as "-", "/" or the space " ". Also the character that indicates the end of a character string and CR andLF are special characters in a character string. They are not handled correctly by the instructions and cause problems.

The data types are converted into a character string according to the following rules:

Data type		Format	Note
Floating-point numbers	LReal	+9.9999999999999E+999	Always written exponentially
	Real	+9.9999999E+99	Always written exponentially
Signed integers	LInt	+9999999999999999999	Value range of signed integers: -9223372036854775808 .. +9223372036854775807
	DInt	+9999999999	Value range of integers: -2147483648 to +2147483647
	Int	+99999	Value range of integers: -32768 to +32767
	SInt	+999	Value range of integers: -128 to +127
Unsigned integers	ULInt	+99999999999999999999	Value range of integers: 0 to +18446744073709551615
	UDInt	+9999999999	Value range of integers: 0 to +4294967295
	UInt	+99999	Value range of integers: 0 to +65535
	USInt	+999	Value range of integers: 0 to +255
Binary numbers	LWord	+99999999999999999999	Value range of integers: 0 to +18446744073709551615
	DWord	+9999999999	Value range of integers: 0 to +4294967295
	Word	+99999	Value range of integers: 0 to +65535
	Byte	+999	Value range of integers: 0 to +255
	Bool	9	Value range: 0 or 1

Data type		Format	Note
Date and time	LTIME	dddddd:hh:mm:ss. 999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	TIME	hhh:mm:ss.999	ISO format with milliseconds
	S5TIME	hhh:mm:ss.999	ISO format with milliseconds
	LDT	YYYY-MM-DD hh:mm:ss. 999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	DTL	YYYY-MM-DD hh:mm:ss. 999_999_999	ISO format with milliseconds, microseconds and nanoseconds
	DT	YYYY-MM-DD hh:mm:ss.999	ISO format with milliseconds
	DATE	YYYY-MM-DD	ISO format
	LTime_Of_ Day	hh:mm:ss.999_999_999	ISO format
	TOD	hh:mm:ss.999	ISO format
Characters	WString	"abcd"	• Character string inside double quotation marks • The current length is padded with spaces up to the maximum length. • A character string of data type WString consists of elements of the WChar data type. When a data log is saved, the contents of characters of the WChar data type are limited to 16#FF (example: 16#1255 is changed to 16#55).
	String	"abcd"	• Character string inside double quotation marks • The current length is padded with spaces up to the maximum length.
	WChar	"a"	Individual character inside double quotation marks. When a data log is saved, the contents of characters of the WChar data type are limited to 16#FF (example: 16#1255 is changed to 16#55).
	Char	"a"	Individual character inside double quotation marks

The data logging instructions are used in your program to create or open a data log, to write an entry and to close the data log file.

WARNING

Runtime of the data logging instructions

The conditions specified below slow down the data logging instructions. This effect becomes stronger with the growing size and number of data logging files.

- Large file size of the data log file
- Large number of data log files on the memory card
- Simultaneous access to a log file by the Web server

The runtime of some data logging instructions is not typically impaired by the size and number of data logs (e.g. "DataLogWrite"); the instructions "DataLogCreate", "DataLogNewFile", "DataLogClear" and "DataLogOpen", on the other hand, show the behavior described.

Note

Data logging instructions with which data is transferred

Because the data logging functions are executed asynchronously, you must keep the data in the source area consistent until the DONE parameter or the ERROR parameter changes to the value "1".

You decide by the creation of the data buffer which program values are stored in the data log during the creation of the data buffer. The data buffer is used as a memory for new data log entries. New values have to be written to the buffer before "DataLogWrite (Page 2683)" is called. During execution of the "DataLogWrite (Page 2683)" instruction, data is written from the buffer to a data log record.

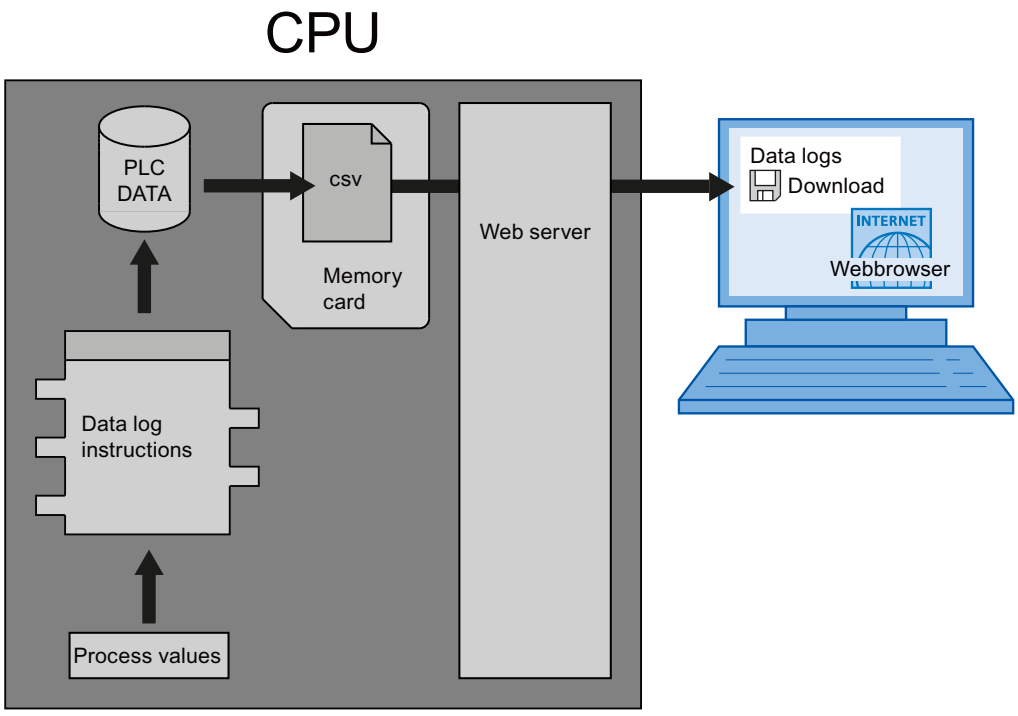

Data log files can be copied to the PC as follows:

- If the PROFINET interface is connected to the PC, you use a Web browser to access the data logs via the Web server. The CPU can be in RUN or STOP mode for this. If the CPU is in "RUN" mode, the program continues running while the Web server is transferring data.
- If there is a memory card in the CPU, you can remove this card and insert it into a standard slot for SD (Secure Digital) cards or MMC (MultiMediaCard) cards on a PC or programming device. Use File Manager to transfer the data log files from the memory card to the PC. When you remove the memory card, the CPU goes to "STOP".

Note

Service life of memory cards

The service life of memory cards is limited. Additional information is available here: AUTOHOTSPOT

Properties of the data log

Writing of the data records of a data log is carried out in accordance with the principle of a ring buffer. New data records are added until the maximum number of data records is reached (RECORD parameter). The next data record then overwrites the "oldest" data record of the data log.

If you want to prevent data records from being overwritten, use the "DataLogNewFile (Page 2688)" instruction to create a new data log file based on the current data log. New data records are then written into the new data log.

Creating data logs

With the "DataLogCreate (Page 2669)" instruction, you can create a new data log file in the ""\DataLogs" directory of the load memory.

- The name assigned at the NAME parameter is the designation for the data log and is also used as the file name for the CSV file. The file is stored in the directory "DataLogs".
- The block parameter DATA determines the data buffer for the new data log object and the columns and data types in the data log. The columns and data types of a data record in the data log are determined by the elements of the structure declaration or array declaration of this data buffer. Each element of a structure or of an array corresponds to a column in a line in the data log.
- You use the HEADER block parameter to assign a header text in the header to each column.
- The "DataLogCreate (Page 2669)" instruction returns an ID. This ID is used by the other data logging instructions as a reference for the created data log.

Opening data logs

You use the instruction "DataLogOpen" (S7-1200 and S7-1500) to open an existing data log on the memory card. A data log must be open before you can write new data records to it.

Data log opens automatically when you execute the instructions "DataLogCreate (Page 2669)" and "DataLogNewFile (Page 2688)".

Up to 10 data logs can be open simultaneously. The data log to be opened can be selected using the ID or name of the data log.

- If you specify the ID and the name of the data log in the ID and NAME parameters, respectively, the data log will be identified based on the ID. The data log name will not be compared.
- If you select the data log using the parameter NAME and specify 0 as the ID, the ID belonging to the data log will be displayed in the ID parameter when you open the data log.
- If you select the data log using the ID parameter and no name is specified, the name will be displayed in the NAME parameter when the data log is opened.

You use the MODE parameter to specify whether the data records of the data log are to be deleted upon opening.

You can run a consistency check for "DataLogOpen (S7-1500).

Writing to data logs

A data log must be open ("DataLogOpen (Page 2677)" instruction) before a data record can be written to a data log. The "DataLogWrite (Page 2683)" instruction writes a data record to the data log.

Note

Changing the content of data log files

As data log files can be huge (500,000 KB), the content of these files cannot be checked either during switch-on or before execution of "DataLogWrite". You must not change the content outside the CPU as this can result in a corrupt file.

Closing data logs

With the "DataLogClose (Page 2685)" instruction, you close and open data log. You select the data log using the ID parameter.

The data log is closed automatically when the CPU switches to STOP and upon restart.

Deleting data logs

You use the instruction "DataLogDelete (Page 2687)" (S7-1500) to delete a data log file from the memory card. The data log and the data records it contains can only be deleted if the log was created with the instruction "DataLogCreate (Page 2669)".

Select the data log to be deleted using the NAME and ID parameters. The ID parameter is evaluated first. If there is a data log with the relevant ID, the NAME parameter will not be

evaluated. If the value "0" is used at the ID parameter, a value with the data type STRING must be used at the NAME parameter.

Note

Deleting data logs on the memory card

You can copy data logs from a memory card using a PC card reader, but you must not change them or delete them using a PC card reader.

In addition to the data log instructions, the use of a Web server is recommended to display, download (copy) and delete data logs.

Calling the file system of the memory card in the file Explorer directly brings with it the risk that you accidentally delete or change data log or other system files so that a file becomes damaged or the memory card becomes unusable. Changes to data logs on the memory card that you made outside the PLC have the effect POWER ON time of the PLC is increased.

Clearing data logs

You use the instruction "DataLogClear (Page 2681)" (S7-1500) to delete all the data records in an existing data log. The instruction does not delete the optional header of the CSV file (see the description of the HEADER parameter of the instruction "DataLogCreate (Page 2669)").

You use the ID parameter to select the data log whose data records are to be deleted. Before you can delete data records, the data log must be open.

New file for data logs

You use the instruction "DataLogNewFile (Page 2688)" (S7-1200) or "DataLogTypedNewFile (Page 2690)" (S7-1500) to create a new data log with the same properties as an existing data log. This allows the contents of an existing data log to be retained.

When called, the instruction creates a new data log on the memory card or in the internal load memory with the name defined at the NAME parameter. You use the ID parameter to specify the ID of the old data log whose properties you want to apply to the new data log. The ID of the new data log is then output via the ID parameter.

You specify the file size of the new data log with the RECORDS parameter of the instruction.

You can run a consistency check for "DataLogTypedNewFile (Page 2690)" (S7-1500).

DataLogCreate: Create data log (S7-1200, S7-1500)

Description

With the "DataLogCreate" instruction you create a data log.

The data log is saved on the memory card or in the internal load memory in the directory "\DataLogs". The amount of data that is stored in a data log is dependent on the available space on the memory card or the storage space in the internal load memory of the CPU used. The maximum size of the generated CSV file is 500,000,000 bytes for S7-1200 CPUs, 1,000,000,000 bytes for S7-1500 CPUs.

You specify the maximum number of data records that is stored in a data log in the RECORDS parameter. When the specified maximum number of data records in the data log is reached, the oldest data record is overwritten. To avoid overwriting of existing data records, use the "DataLogNewFile (Page 2688)" instruction. The instruction is used to create a new data log with the same structure when the number specified at the RECORDS parameter is reached (return value 1 at the STATUS parameter of the "DataLogWrite (Page 2683)" instruction). The data records are then saved in the new data log.

You specify the name for the data log in the NAME parameter. The data log is created in CSV (Comma Separated Value) format. With the HEADER parameter, you create an (optional) header for the data log.

Once the data log is created, it is opened automatically. This means that data can be written.

Parameters

The following table shows the parameters of the "DataLogCreate" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction The data log is created with a rising edge at the parameter REQ.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Maximum number of data records in the data log If the "DataLogWrite (Page 2683)" instruction writes more records than specified in this parameter, the oldest record is overwritten.
FORMAT	Input	UInt	I, Q, M, L, D or constant	Data format: • 0: Internal (not supported) • 1: CSV (Comma separated values)
TIMESTAMP	Input	UInt	I, Q, M, L, D or constant	Time-stamping; for details, see below The extra columns in the header are automatically added if time-stamping is activated.
NAME	Input	VARIANT	L, D	Name of the data log The specified name is also used as a file name for the csv file. With S7-1200 CPUs. the following restrictions apply to name of the data log: • The name must not be longer than 35 characters. • All ASCII characters between 0x20 and 0x7E are permitted with the exception of \ ' / " : ; [ ] | = . * ? < > With S7-1500 CPUs. the following restrictions apply to name of the data log: • The name must not be longer than 55 characters. • The following characters are permitted: 0 ... 9, a ... z, A ... Z as well as - and _

Parameter	Declaration	Data type	Memory area	Description
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log (only output) The ID of the data log is required for further data logging instructions to address the created data log.
HEADER	InOut	VARIANT	D	Header of the data log (optional) The parameter is hidden after the instruction is added. The header is written as the first line in the CSV file.
DATA	InOut	VARIANT	D	Pointer to the data structure that is written as data record when executing the "DataLogWrite (Page 2683)" instruction.
DONE	Output	BOOL	I, Q, M, L, D	Status parameter: • 0: Processing not yet complete. • 1: Processing of instruction finished successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Status parameter: • 0: Processing of the instruction has not started, completed, or canceled. • 1: Processing of the instruction is in progress.
ERROR	Output	BOOL	I, Q, M, L, D	Status parameter: • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Detailed status information: Detailed error and status information is output at the parameter STATUS. The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter HEADER

The HEADER parameter is a VARIANT pointer to a data block that defines a header for the CSV file (header). The header is always the first line in the CSV file representation.

- The header preset using HEADER is transferred to the CSV file. Here there is no check as to whether it is consistent with the remainder of the data log. In particular, there is no check whether the number of columns separated by commas matches the number of data columns.
- A STRING, Array of BYTE, or Array of CHAR data type can be used for the individual column names. When the STRING data type is used, the length is limited to 254 bytes. When the Array of BYTE and Array of CHAR are used, 5000 elements are possible.

If you do not specify a value in the HEADER parameter or HEADER is located in the load memory, no header is created.

For the values 1, 3 and 5 of TIMESTAMP the following text is inserted at the start of the header: "Record, Date, UTC Time,".

For the values 2 and 4 of TIMESTAMP the following text is inserted at the start of the header: "Record, Date, Loc Time,".

Parameter TIMESTAMP

The following table shows the possible values of TIMESTAMP.

Value	Format of DATE_AND_TIME	Time source	Note
0	-	No time stamp	Obligatory
1	• S7 1200: mm/dd/yyyy, hh:mm:ss • S7 1500: yyyy-mm-dd, hh:mm:ss.999	System time	Obligatory
2	• S7-1200 CPUs: mm/dd/yyyy, hh:mm:ss • S7-1500 CPUs as of firmware version V2.6: mm/dd/yyyy, hh:mm:ss[.999] (24-hour time, [milliseconds optional])	Local time	Optional
3	S7-1500 CPUs as of firmware version V2.6: mm/dd/yyyy, hh:mm:ss[.999] (24-hour time, [milliseconds optional])	System time	Optional
4	S7-1500 CPUs as of firmware version V2.6: yyyy-mm-dd, hh:mm:ss.[999] (24-hour time, [milliseconds optional])	Local time	Optional
5	S7-1500 CPUs as of firmware version V2.6: yyyy-mm-dd, hh:mm:ss.[999] (24-hour time, [milliseconds optional])	System time	Optional

Parameter DATA

The DATA parameter is a VARIANT pointer to an elementary data type, a structure or an array in a data block. An element of a structure or an array corresponds to a column in the data log with a specific data type.

Note the following when creating the data block:

- Normally, the number of columns matches the number of columns defined in the HEADER parameter. This is however not stipulated.
- Each element of the structure or the array corresponds to a column entry in the CSV file. Therefore, no nesting structures (STRUCT in STRUCT) may be used when using the data type STRUCT. However the use of arrays within a data type is permitted STRUCT with the following exception: Arrays whose elements are of the data type "PLC data type (UDT)" are not permitted.
- The data structure can contain up to 256 elements. If there are more than 256 elements, error code 8C52 is output at the STATUS parameter.
- The tags of the data block can be set as retentive or non-retentive tags. However, the retentivity setting must be the same for all tags of the data block.

Parameter STATUS (S7-1200)

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	The maximum number of simultaneously active DataLogCreate instructions has already been reached.
8090	Invalid file name (see description of the NAME parameter).
8093	Data log already exists.
8097	The length of the file exceeds the limitation of the file system or the CPU-specific limitation.
80A2	Write error returned from file system.
80B2	Maximum number of data logs exceeded
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
80C1	Too many data logs are open.
8253	The RECORDS parameter has an invalid value.
8353	Invalid format selection
8453	Invalid time stamp
8B24	Invalid assignment at parameter HEADER (for example: points to local memory).
8B51	Impermissible data type at parameter HEADER or length exceeds the maximum size.
8C24	Invalid assignment at parameter DATA (for example: points to local memory).
8C51	Impermissible data type at parameter DATA / data structure cannot be used.
8C52	Structure at the parameter DATA contains more than 256 elements.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Parameter STATUS (S7-1500)

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
8070	The maximum number of simultaneously active DataLogCreate instructions has already been reached.
8090	Invalid file name (see description of the NAME parameter).
8091	"NAME" parameter is not a string.
8093	Data log already exists.
8097	The length of the file exceeds the limitation of the file system or the CPU-specific limitation.
80A2	Write error returned from file system.
80B2	Maximum number of data logs exceeded
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
80C1	Too many data logs are open.
8253	The RECORDS parameter has an invalid value.
8353	Invalid format selection.
8453	Invalid time stamp.
8B24	Invalid assignment at parameter HEADER (for example: points to local memory).
8B51	Impermissible data type at parameter HEADER or length exceeds the maximum size.
8C24	Invalid assignment at parameter DATA (for example: points to local memory).
8C51	Impermissible data type at parameter DATA / data structure cannot be used.
8C52	Structure at the parameter DATA contains more than 256 elements.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, a simple data log is created with a time stamp and three process values.

Tags in the global data block

The values for the input parameters of the data log are stored in the global data block "DataLogDB":

- DataLogName (String): The tag contains the name of the data log which is also used as the file name for the CSV file.
- DataLogID (DInt): The ID of the data log is written in this tag when the instruction is called.
  - The ID is automatically assigned by the instruction.
  - Use the tag "DataLogID" in other DataLog instructions in order to address this data log.

- MyHeader (String): This tag contains the header of the data log, i.e. the column headings for the process values.
- MyData (Struct): The tag contains the three process values that are written in the data log. Each time a record is written (instruction "DataLogWrite (Page 2683)") the current values are written in a new record.

DataLogDB

	Name	Data type	Start value	Retain
1	Static			
2	DataLogName	String	'MyDataLog'	
3	DataLogID	DInt	0	
4	MyHeader	String	'Value1,Value2,Value3'	
5	MyData	Struct		
6	ProcessValue1	Int	2	
7	ProcessValue2	Int	3	
8	ProcessValue3	Int	4	

Calling the "DataLogCreate" instruction

The instruction is called with the following input parameters:

- REQ (BOOL): The data log is created when REQ = "TRUE".
- RECORD (3): A maximum of three data records can be written in the data log. Thereafter, the oldest data record is overwritten.
- FORMAT (1): The data log is created as a CSV file.
- TIMESTAMP (1): Enabled: Two additional columns are automatically created (date and time) for the data log. The current time stamp is written in the data record each time "DataLogWrite (Page 2683)" is executed.
- NAME (VARIANT): Pointer to the tag "DataLogName" in the data block "DataLogDB".
- ID (DWORD): Pointer to the tag "DataLogID" in the data block "DataLogDB" (output only).
- HEADER (VARIANT): Pointer to the tag "MyHeader" in the data block "DataLogDB".
- DATA (VARIANT): Pointer to the tag "MyData" in the data block "DataLogDB".

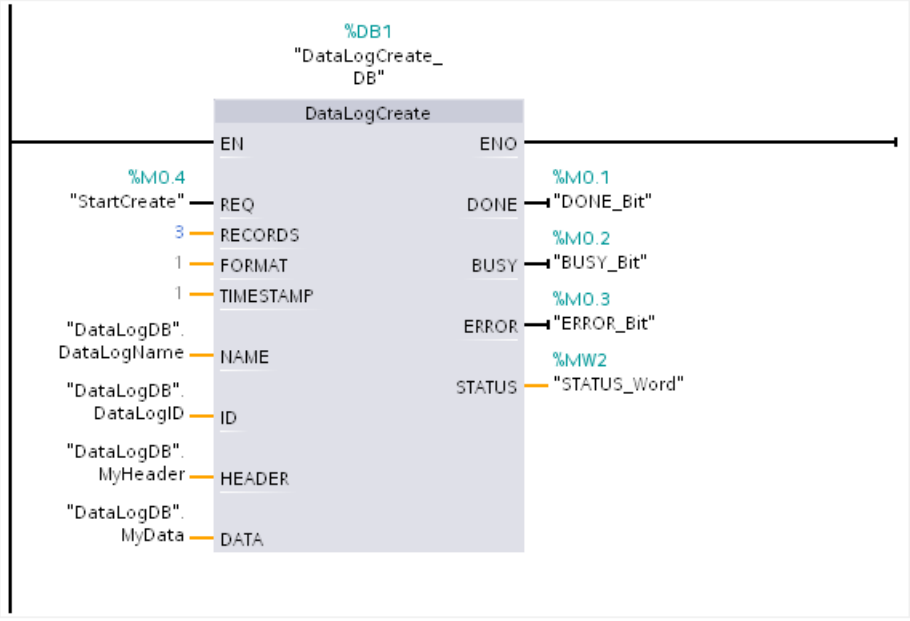

Readout of the data log via the web server

You read out the created data log with the web server:

- Activate the web server in the properties of the CPU. The authorization "...to read files" must be activated for the web server in order to readout the data log.
- You can access the web server via the Internet browser by entering the IP address of the CPU as a URL.
- Under "Filebrowser" the directory "\DataLogs" was automatically created which contains the data log. Alternatively, you can also look in the menu under "DataLogs".
- The DataLog only contains one entry "//END" if the instruction "DataLogWrite (Page 2683)" has still not been executed. The first data record is written after the initial execution of "DataLogWrite (Page 2683)".

	A	B	C	D	E	F
1	SeqNo	Date	Time	ProcessValue1	ProcessValue2	ProcessValue3
2	1	01.07.2013	12:33:42.917	2	3	4
3						
4						
5						

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Data logging - Overview (Page 2664)

Difference between synchronous and asynchronous instructions (Page 596)

DataLogOpen: Open data log

DataLogOpen: Open data log (S7-1200)

Description

You use the "DataLogOpen" instruction to open an existing data log on the memory card. To write new data records to the data log, a data log must be open.

Data log opens automatically when you execute the instructions "DataLogCreate (Page 2669)" and "DataLogNewFile (Page 2688)".

Up to 10 data logs can be open simultaneously. The data log to be opened can be selected using the ID or name of the data log.

- If you specify the ID and the name of the data log in the ID and NAME parameters, respectively, the data log will be identified based on the ID. The data log name will not be compared.
- If you select the data log using the parameter NAME and specify 0 as the ID, the ID belonging to the data log will be displayed in the ID parameter when you open the data log.
- If you select the data log using the ID parameter and no name is specified, the name will be displayed in the NAME parameter when the data log is opened.

You use the MODE parameter to specify whether the data records of the data log are to be deleted upon opening.

Parameters

The following table shows the parameters of the "DataLogOpen" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execution of the instruction upon a rising edge.
MODE	Input	UInt	I, Q, M, L, D or constant	Mode for opening the data log: • MODE= "0" Data records of the data log are retained • MODE= "1" Data records of the data log are deleted, but the header is retained
NAME	Input	VARIANT	L, D	(File) name of the data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log.
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, L, D	Execution of instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Error code* (W#16#...)	Description
0	No errors.
2	Warning: Data log file has already been opened by this application.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	The maximum number of simultaneously active DataLogOpen instructions has already been reached.
8090	There are inconsistencies between the data log definition and existing data log data.
8091	A data type other than String was used at the NAME parameter.
8092	Data log does not exist.
80B4	The memory card or data log file is write-protected.
80C0	Access currently not possible. This may be the case for example when loading during RUN.
80C1	Too many files open.
The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Example program for working with data logs (Page 2692).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Data logging - Overview (Page 2664)

Difference between synchronous and asynchronous instructions (Page 596)

DataLogOpen: Open data log (S7-1500)

Description

You use the "DataLogOpen" instruction to open an existing data log on the memory card. A data log must be open to write new data records to it.

Data log opens automatically when you execute the instructions "DataLogCreate (Page 2669)" and "DataLogNewFile (Page 2688)".

Up to 10 data logs can be open simultaneously. The data log to be opened can be selected using the ID or name of the data log.

- If you specify the ID and the name of the data log in the ID and NAME parameters, respectively, the data log will be identified based on the ID. The data log name will not be compared.
- If you select the data log using the parameter NAME and specify 0 as the ID, the ID belonging to the data log will be displayed in the ID parameter when you open the data log.
- If you select the data log using the ID parameter and no name is specified, the name will be displayed in the NAME parameter when the data log is opened.

You use the MODE parameter to specify whether the data records of the data log are to be deleted upon opening.

The parameter DATA enables a consistency check between the data log to be opened and the definition of the data log of the instruction "DataLogCreate (Page 2669)". A check is made as to whether the data type of the DATA parameter is identical to the data type of "DataLogCreate". If this is not the case, the error code W#16#8090 is output at the STATUS parameter.

Parameters

The following table shows the parameters of the "DataLogOpen" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
MODE	Input	UInt	I, Q, M, L, D or constant	Mode for opening the data log: • MODE= "0" Data records of the data log are retained • MODE= "1" Data records of the data log are deleted, but the header is retained
NAME	Input	VARIANT	L, D	(File) name of the data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log.
DATA	InOut	VARIANT	D	With consistency check: Pointer to the data area at the DATA parameter of the "DataLogCreate (Page 2669)" instruction.
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Error code* (W#16#...)	Description
0	No errors.
2	Warning: Data log file has already been opened by this application.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	The maximum number of simultaneously active DataLogOpen instructions has already been reached.

Error code* (W#16#...)	Description
8090	Data types inconsistent. The data log at parameter ID uses different data types than specified at parameter DATA.
8091	A data type other than String was used at the NAME parameter.
8092	Data log does not exist.
80B4	The memory card or data log file is write-protected.
80C0	Access currently not possible. This may be the case for example when loading during RUN.
80C1	Too many files open.
The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You will find the example here: Example program for working with data logs (Page 2692).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Data logging - Overview (Page 2664)

Difference between synchronous and asynchronous instructions (Page 596)

DataLogClear: Empty data log (S7-1200, S7-1500)

Description

The "DataLogClear" instruction deletes all data records in an existing data log. The instruction does not delete the optional header of the CSV file (see the description of the HEADER parameter of the instruction "DataLogCreate (Page 2669)").

You use the ID parameter to select the data log whose data records will be deleted.

Requirement

Before you can delete data records, the data log must be open (see "DataLogOpen (Page 2677) instruction").

Parameters

The following table shows the parameters of the "DataLogClear" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
ID	InOut	DWORD	I, Q, M, D, L	Object ID of the data log
DONE	Output	BOOL	I, Q, M, D, L	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output at the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

Error code* (W#16#...)	Explanation
0000	No error.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	The maximum number of simultaneously active DataLogClear instructions has already been reached.
8080	The data log file selected with the ID parameter is not processed by the "DataLogClear" instruction.
8092	Data log does not exist.
80A2	Write error signaled back by the file system.
80B0	Data log is not open.
80B4	The memory card is write-protected.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

Data logging - Overview (Page 2664)

Difference between synchronous and asynchronous instructions (Page 596)

DataLogWrite: Write data log (S7-1200, S7-1500)

Description

With the "DataLogWrite" instruction you write a data record into an existing data log. With the ID parameter you select the data log into which the data record is to be written. To create a new data record, the data log must be open. The instruction creates a new data record in the format that was specified in the DATA parameter when the data log was created.

Before calling the "DataLogWrite" instruction, transfer the data to the tag that you interconnected at the DATA parameter of the "DataLogCreate" instruction. When the "DataLogWrite" instruction is executed, the transferred data is copied to the data log.

NOTICE
Data log data loss when the power supply to the CPU is interrupted If the power supply is interrupted during execution of the "DataLogWrite" instruction, the data record to be transferred is lost.

Parameters

The following table shows the parameters of the "DataLogWrite" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS (S7-1200)

Error code* (W#16#...)	Description
0	No errors
0001	Last possible data record created at the end of the file. Creating another data record will overwrite an older data record.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	The maximum number of simultaneously active DataLogWrite instructions has already been reached.
8090	Data log definition does not match the existing data log.
8092	Data log does not exist.
8093	The source address for the data log has been changed (e.g. by download).
80A2	Write error signaled back by the file system.
80B0	Data log is not open.
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Parameter STATUS (S7-1500)

Error code* (W#16#...)	Description
0	No errors
0001	Last possible data record created at the end of the file. Creating another data record will overwrite an older data record.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
8070	The maximum number of simultaneously active DataLogWrite instructions has already been reached.
8090	Data log definition does not match the existing data log.
8092	Data log does not exist.
8093	The source address for the data log has been changed (e.g. by download).
80A2	Write error signaled back by the file system.
80B0	Data log is not open.
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You will find the example here: Example program for working with data logs (Page 2692).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Data logging - Overview (Page 2664)

Difference between synchronous and asynchronous instructions (Page 596)

DataLogClose: Close data log (S7-1200, S7-1500)

Description

With the "DataLogClose" instruction, you close an open data log. You select the data log using the ID parameter.

Note

Closing data logs automatically

The data log is closed automatically when the CPU goes to STOP or if there is a restart.

Parameters

The following table shows the parameters of the "DataLogClose" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execute function on a rising edge.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS (S7-1200)

Error code* (W#16#...)	Description
0	No errors
1	Data log is not open
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	The maximum number of simultaneously active DataLogClose instructions has already been reached.
8092	Data log does not exist.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Parameter STATUS (S7-1500)

Error code* (W#16#...)	Description
0	No errors
1	Data log is not open
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
8070	The maximum number of simultaneously active DataLogClose instructions has already been reached.
8092	Data log does not exist.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Example program for working with data logs (Page 2692).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

DataLogDelete: Delete data log (S7-1200, S7-1500)

Description

You use the "DataLogDelete" instruction to delete a data log file from the memory card. The data log and the data records it contains are only deleted if they were created with the "DataLogCreate" or "DataLogNewFile" instruction.

Parameters

The following table shows the parameters of the "DataLogDelete" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
NAME	Input	VARIANT	L, D	File name of the data log
DELFILE	Input	BOOL	I, Q, M, D, L or constant	• 0: Data log is retained. • 1: Data log is deleted.
ID	InOut	DWORD	I, Q, M, D, L	Object ID of the data log
DONE	Output	BOOL	I, Q, M, D, L	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Deletion of the data log is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output at the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameters NAME and ID

Select the data log to be deleted using the NAME and ID parameters. The ID parameter is evaluated first. If there is a data log with the relevant ID, the NAME parameter will not be evaluated. If the value "0" is used at the ID parameter, a value with the data type STRING must be used at the NAME parameter.

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	The maximum number of simultaneously active DataLogDelete instructions has already been reached.
8091	A data type other than STRING is being used at the NAME parameter.
8092	Data log does not exist.
80A2	Write error signaled back by the file system.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You will find the example here: Example program for working with data logs (Page 2692).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

DataLogCreate: Create data log (Page 2669)

Difference between synchronous and asynchronous instructions (Page 596)

DataLogNewFile: Data log in new file

DataLogNewFile: Data log in new file (S7-1200)

Description

With the "DataLogNewFile" instruction you create a new data log with the same properties as an already existing data log. This allows the contents of an existing data log to be retained.

When called, the instruction creates a new data log on the memory card or in the internal load memory with the name defined at the NAME parameter. You use the ID parameter to specify the ID of the old data log whose properties you want to apply to the new data log. The ID of the new data log is then output via the ID parameter.

You specify the file size of the new data log with the RECORDS parameter of the instruction.

Once the new data log is created, it is opened automatically. This means that data can be written.

Parameters

The following table shows the parameters of the "DataLogNewFile" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execution of the instruction upon a rising edge.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Number of data records in the data log.
NAME	Input	VARIANT	L, D	File name of the new data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log • In: ID of the existing data log • Out: ID of the new data log
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	The maximum number of simultaneously active DataLogNewFile instructions has already been reached.
8090	Invalid file name.
8091	Data type at parameter NAME is not STRING.
8092	Source data log does not exist.
8093	New data log already exists.
8097	File length exceeds the file system limit.

Error code* (W#16#...)	Description
80A0	Data types inconsistent. The data log at parameter ID uses different data types than specified at parameter DATA.
80A2	Write error signaled back by the file system.
80B2	Maximum number of data logs exceeded or data log resources used up
80B3	Load memory not sufficient.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
80C1	Too many files open.
8253	The RECORDS parameter has an invalid value.
The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Example program for working with data logs (Page 2692).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

DataLogNewFile: Data log in new file (S7-1500)

Description

With the "DataLogNewFile" instruction you create a new data log with the same properties as an already existing data log. This allows the contents of an existing data log to be retained.

When called, the instruction creates a new data log on the memory card or in the internal load memory with the name defined at the NAME parameter. You use the ID parameter to specify the ID of the old data log whose properties you want to apply to the new data log. The ID of the new data log is then output via the ID parameter.

You specify the file size of the new data log with the RECORDS parameter of the instruction.

The parameter DATA enables a consistency check between the new data log to be created and the definition of the data log of the instruction "DataLogCreate (Page 2669)". A check is made as to whether the data type of the DATA parameter is identical to the data type of "DataLogCreate". If this is not the case, the error code W#16#80A0 is output at the STATUS parameter.

Once the new data log is created, it is opened automatically. This means that data can be written.

Parameters

The following table shows the parameters of the "DataLogNewFile" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Number of data records in the data log.
NAME	Input	VARIANT	L, D	File name of the new data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log • In: ID of the existing data log • Out: ID of the new data log
DATA	InOut	VARIANT	D	Data type for consistency check
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	The maximum number of simultaneously active DataLogNewFile instructions has already been reached.
8090	Invalid file name
8091	Data type at parameter NAME is not STRING.
8092	Source data log does not exist.
8093	New data log already exists.
8097	The length of the file exceeds the limitation of the file system or the CPU-specific limitations.

Error code* (W#16#...)	Description
80A0	Data types inconsistent. The data log at parameter ID uses different data types than specified at parameter DATA.
80A2	Write error signaled back by the file system.
80B2	Maximum number of data logs exceeded or data log resources used up
80B3	Load memory not sufficient.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
80C1	Too many files open.
8253	The RECORDS parameter has an invalid value.
The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You will find the example here: Example program for working with data logs (Page 2692).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

Example program for working with data logs

Introduction

In the following example program you create a data log, write one or more records into the data log, close and open the data log and create a new file based on the data log. A possibility to delete data logs is also provided in the example. The example also offers the possibility to edit up to eight data logs.

For detailed information about individual instructions, open the corresponding help description using the corresponding links.

General notes on using data logs

- The data log created is automatically opened when the "DataLogCreate" and "DataLogNew" instructions are executed.
- Data logs are automatically closed after the CPU switches from RUN to STOP or after a restart of the CPU.
- For the "DataLogWrite" instruction to be executed, the data log must be open.
- For S7-1200 CPUs, a maximum of eight data logs can be open simultaneously. For S7-1500 CPUs, a maximum of ten data logs can be open simultaneously.

Requirements

You carry out the following settings in the CPU properties:

- In "PROFINET interface > Access to the web server" you activate the access to the web server.
- Activate the web server for the module under the "Web server" entry.
- In the “User administration“ section create a new user with read, write and delete rights for files.

Storage of data

The names and ID for the data logs are required to call up and edit a data log. Create a memory for data log entries so that these data do not get lost. You define the structure of a data log entry via a PLC data type "stackDataLog".

stackDataLog

	Name	Data type	Default value
1	name	String	''
2	ID	DWord	16#0
3	DLclosed	Bool	false

Create the following tags for storing the data in a global data block (SLI_gDB_Datalogging). Among them are also the memories for data log entries and the variables to control the memory.

Note: If you want to also physically delete data logs from the load memory of the CPU, use the "DLdelete.deleteMode" tag with the value "TRUE".

SLI_gDB_DataLogging

	Name	Data type	Start value	Retain
1	Static			
2	DLcreate	Struct		
3	execute	Bool	false	
4	busy	Bool	false	
5	error	Bool	false	
6	memErrStatus	Word	16#0	
7	dlogCreated	Bool	false	
8	DLdelete	Struct		
9	execute	Bool	false	
10	deleteMode	Bool	false	
11	busy	Bool	false	
12	error	Bool	false	
13	memErrStatus	Word	16#0	
14	dlogDeleted	Bool	false	
15	name	String	'myDataLogEE1'	
16	newName	String	'myNewDataLogEE1'	
17	logID	DWord	16#0	
18	logHeader	String	'Count,Temperature,Pressure'	
19	myData	Struct		
20	count	Int	0	
21	temperature	Real	0.0	
22	pressure	Real	0.0	
23	maxPosEntry	UInt	7	
24	dataLogEntries	Array[0..7] of "stackDataLog"		✓
25	nextPosInStack	UInt	0	✓
26	callEntry	UInt	0	

The DB ("SLI_gDB_Datalogging") is used to create (DataLogCreate (Page 2669)) or to delete (DataLogDelete (Page 2687)) a data log. The three entries of the structure "myData" are used as process values: count, temerature and pressure. In the data block, these three values are stored temporarily and then transferred with the "DataLogWrite (Page 2683)" instruction to a data log as a data record.

A data record is composed later from the following entries:

- The data record number (assigned automatically)
- The date (assigned automatically when "1" is used for DataLogCreate at the TIMESTAMP parameter).
- The time of day (assigned automatically when "1" is used for DataLogCreate at the TIMESTAMP parameter).
- The current value of "count" from the "myData" structure.
- The current value of "temperature" from the "myData" structure.
- The current value of "pressure" from the "myData" structure.

Use the global data block "SLI_gDB_DataLogW" to make available the process values for writing a data record DataLogWrite (Page 2683)).

SLI_gDB_DataLogW			
Name	Data type	Start value	Retain
1 Static			
2 DLwrite	Struct		
3 execute	Bool	false	
4 preset	Struct		
5 count	Int	5	
6 scaledTemp	Real	25.7	
7 scaledPressure	Real	45.0	
8 busy	Bool	false	
9 error	Bool	false	
10 status	Word	16#0	
11 memErrStatus	Word	16#0	
12 memDone	Bool	false	
13 DLclose	Struct		
14 execute	Bool	false	
15 busy	Bool	false	
16 error	Bool	false	
17 status	Word	16#0	
18 memDone	Bool	false	
19 DLopen	Struct		
20 execute	Bool	false	
21 busy	Bool	false	
22 error	Bool	false	
23 status	Word	16#0	
24 memDone	Bool	false	
25 DLnewfile	Struct		
26 execute	Bool	false	
27 busy	Bool	false	
28 error	Bool	false	
29 status	Word	16#0	
30 newFileCreated	Bool	false	

FC "SLI_FC_saveEntry_DataLog": Interconnecting parameters

Use an FC "SLI_FC_saveEntry_DataLog“ to save the name and ID of a data log. Create the following local tags in it.

SLI_FC_saveEntry_DataLog	
Name	Data type
1 Input	
2 name	String
3 logID	DWord
4 Output	
5 <Add new>	
6 InOut	
7 <Add new>	
8 Temp	
9 nextPosInStack	UInt
10 maxPosEntry	UInt
11 tempName	String
12 tempID	DWord

Network 1: Via the input parameters "#name" and "#logID" you transfer the values of the tags "name" and "logID“.

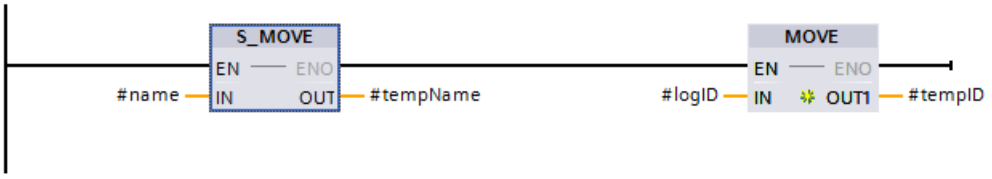

Network 2: The "nextPosInstack" tag specifies the storage location to be used for a data log entry in the "dataLogEntries“ array. The "maxPosEntry" tag specifies the last cell in the "dataLogEntries“ array. Interconnect the values of the "nextPosInstack" and "maxPosEntry“ tags for further processing as follows.

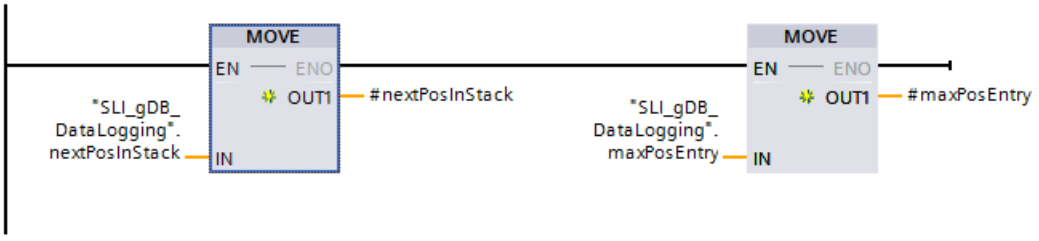

Network 3: You store the name and ID in the following way in the "dataLogEntries“ array.

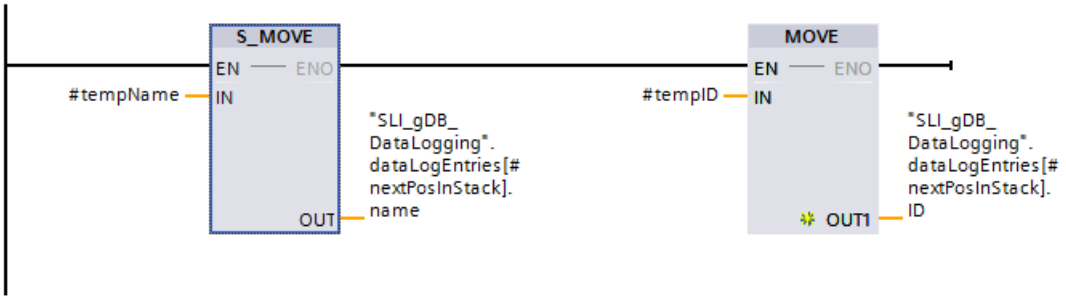

Network 4: To ensure the storage of a new data log in a different entry of the memory, increment the tag "nextPosInStack".

Note: When the storage limit ("#maxPosEntry") has been reached, the "nextPosInStack" tag is not incremented, but reset to the value "0". This means that the data of a new data log would overwrite the data of an old data log.

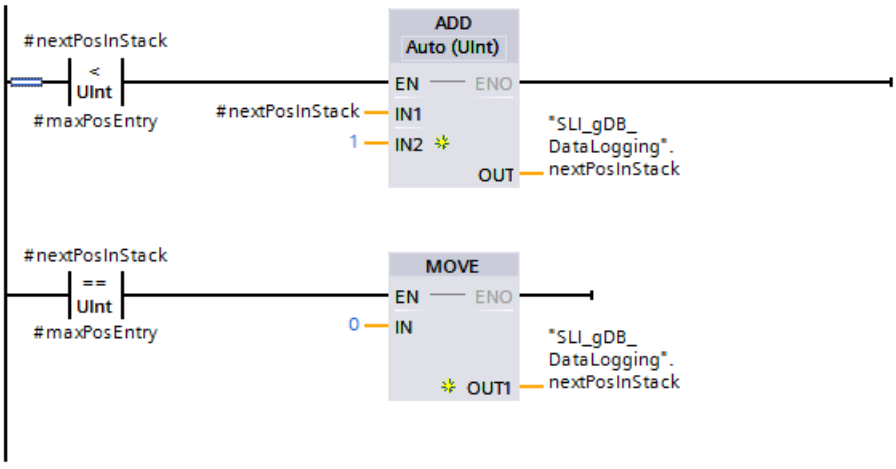

FC "SLI_FC_callEntry_DataLog": Interconnecting parameters

To call up the name and ID of a selected data log entry create an FC "SLI_FC_callEntry_DataLog". Create the following local tags in it. You specify the selection of a data log entry via the "callEntry“ tag.

SLI_FC_callEntry_DataLog

		Name	Data type
1		Input	
2		callEntry	UInt
3		Output	
4		name	String
5		logID	DWord
6		InOut	
7		<Add new>	
8		Temp	
9		tempEntry	UInt
10		tempName	String
11		tempID	DWord

Network 1: Interconnect the values of the "callEntry“ tags for further processing as follows.

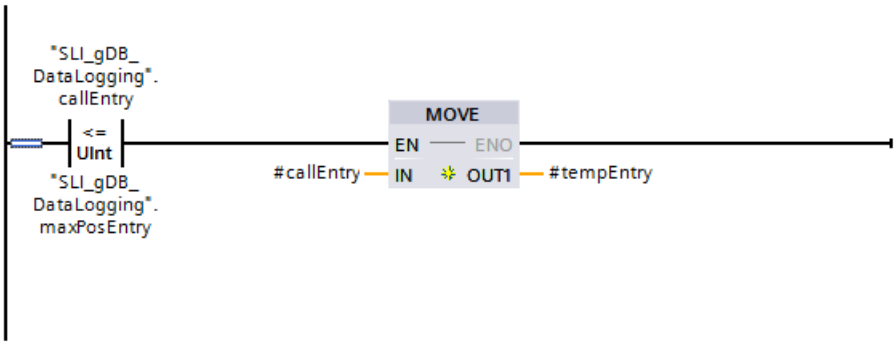

Network 2: The name and ID of the created data log are called as follows.

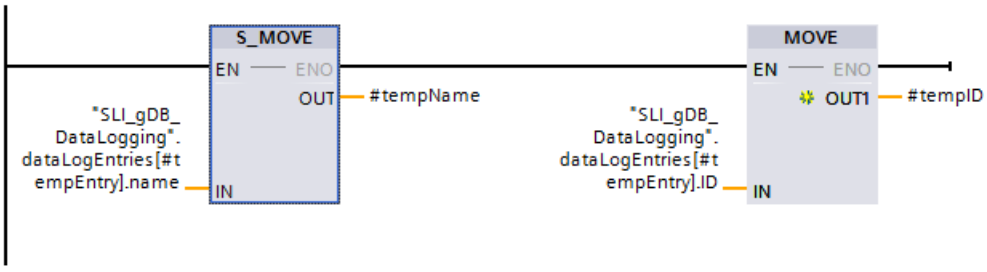

Network 3: Via the output parameters "#name" and "#logID" you transfer the values of the tags "dataLogEntries.name" and "dataLogEntries.ID“.

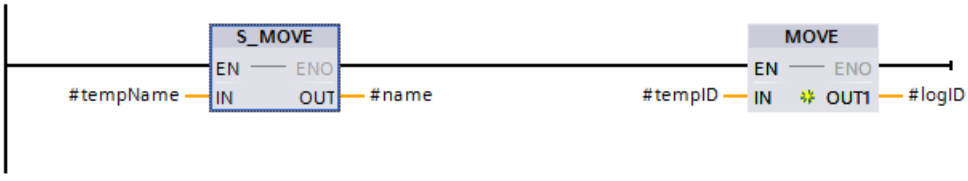

FC "SLI_FC_deleteEntry_DataLog": Interconnecting parameters

To delete the data of a data log entry use an FC "SLI_FC_deleteEntry_DataLog". Create the following local tags in it.

SLI_FC_deleteEntry_DataLog

		Name	Data type	Default value
1		Input		
2		deleteEntry	UInt	
3		Output		
4		<Add new>		
5		InOut		
6		<Add new>		
7		Temp		
8		tempEntry	UInt	
9		Constant		
10		EMPTYSTRING	String	''

Network 1: Subsequently ensure that the temporary tag "#tempEntry" has a value.

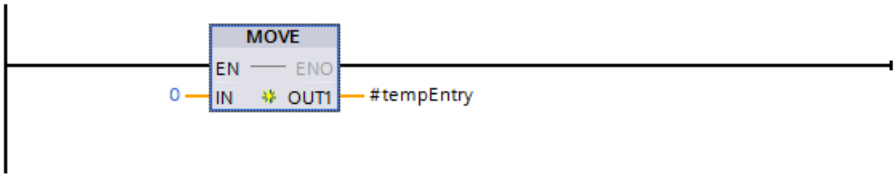

Network 2: If the value of the "callEntry" tag does not exceed the value of the "maxPosEntry" tag, the following is to happen:

- The value of the "callEntry" tag is transferred at the input parameter #deleteEntry. The data log entry to be deleted is selected via the tag "callEntry".
- The tags "dataLogEntries.name", "dataLogEntries.ID" and "dataLogEntries.DLclosed" of the selected data log entry are reset.

The interconnections have the following appearance.

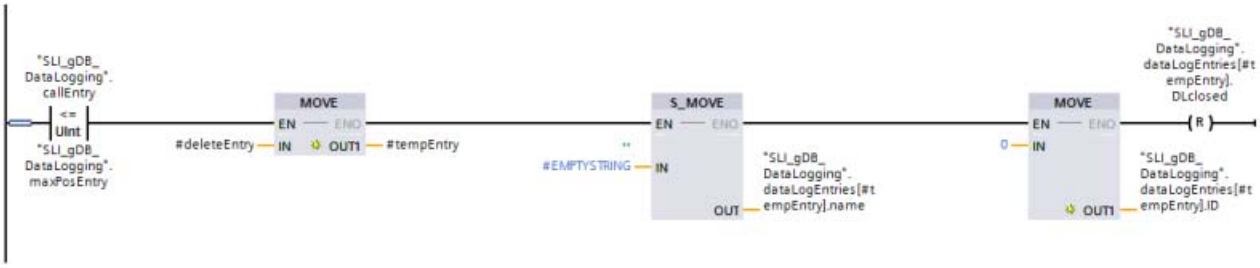

FC "SLI_FC_memCloseEntry_DataLog": Interconnecting parameters

To save the status (open or closed) of the data log use an FC "SLI_FC_memCloseEntry_DataLog". Create the following local tags in it.

SLI_FC_memCloseEntry_DataLog		
	Name	Data type
1	Input	
2	callEntry	UInt
3	memClose	Bool
4	Output	
5	<Add new>	
6	InOut	
7	<Add new>	
8	Temp	
9	tempEntry	UInt
10	tempClose	Bool

Network 1: To select the data log entry transfer the value of the input parameter "#callEntry". In addition you transfer the state of the data log via the input parameter "#memClose".

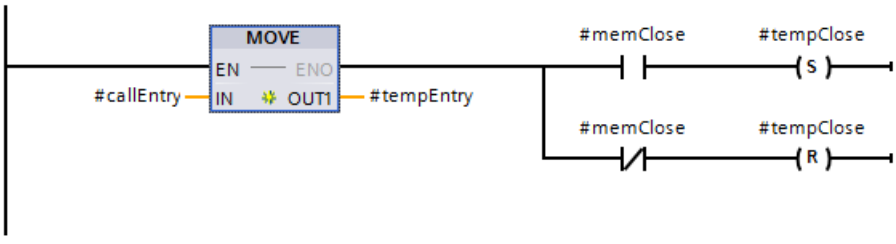

Network 2: Save the status of the data log in the suitable data log entry. The interconnections have the following appearance.

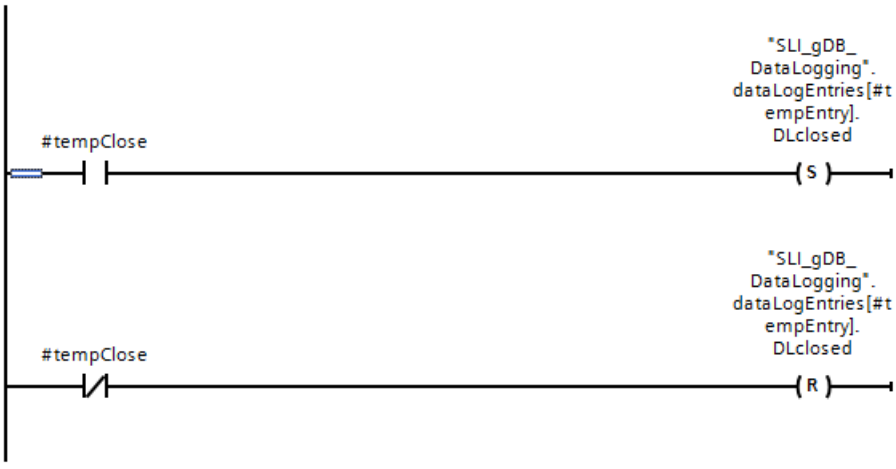

FB "SLI_FB_DataLogging": Interconnecting parameters

Create an FB "SLI_FB_DataLogging“ as the central block of the example program. Create the following local tags in the FB.

7	Static			
8	createDone	Bool	false	Non-retain
9	createStatus	Word	16#0	Non-retain
10	deleteDone	Bool	false	Non-retain
11	deleteStatus	Word	16#0	Non-retain
12	writeDone	Bool	false	Non-retain
13	executeClose	Bool	false	Non-retain
14	closeDone	Bool	false	Non-retain
15	openDone	Bool	false	Non-retain
16	memOpenDone	Bool	false	Non-retain
17	newFileDone	Bool	false	Non-retain
18	newFileStatus	Word	16#0	Non-retain
19	stateMem	Bool	false	Non-retain
20	stateMem2	Bool	false	Non-retain
21	stateMem3	Bool	false	Non-retain
22	stateMem4	Bool	false	Non-retain
23	stateMem5	Bool	false	Non-retain
24	stateMem6	Bool	false	Non-retain
25	DataLogCreate_Instan..	DataLogCreate		
26	DataLogWrite_Instance	DataLogWrite		
27	DataLogClose_Instance	DataLogClose		
28	DataLogOpen_Instance	DataLogOpen		
29	DataLogNewFile_Insta..	DataLogNewFile		
30	DataLogDelete_Instan..	DataLogDelete		
31	Temp			
32	<Add new>			
33	Constant			
34	EMPTYSTRING	String	''	

Create the following interconnections in the FB. You call the FB in a program cycle OB (OB1).

Network 1: A positive edge at the input parameter REQ ("DLcreate.execute") of the instruction DataLogCreate (Page 2669) starts the creation of the data log.

The output parameter DONE ("#createDone") of DataLogCreate is only valid for one cycle. Therefore, save its value in the "DLcreate.dlogCreated“ tag. The "DLcreate.execute" tag is reset in the process.

In case of an error the status ("DLcreate.status") in the "DLcreate.memErrStatus" tag is saved.

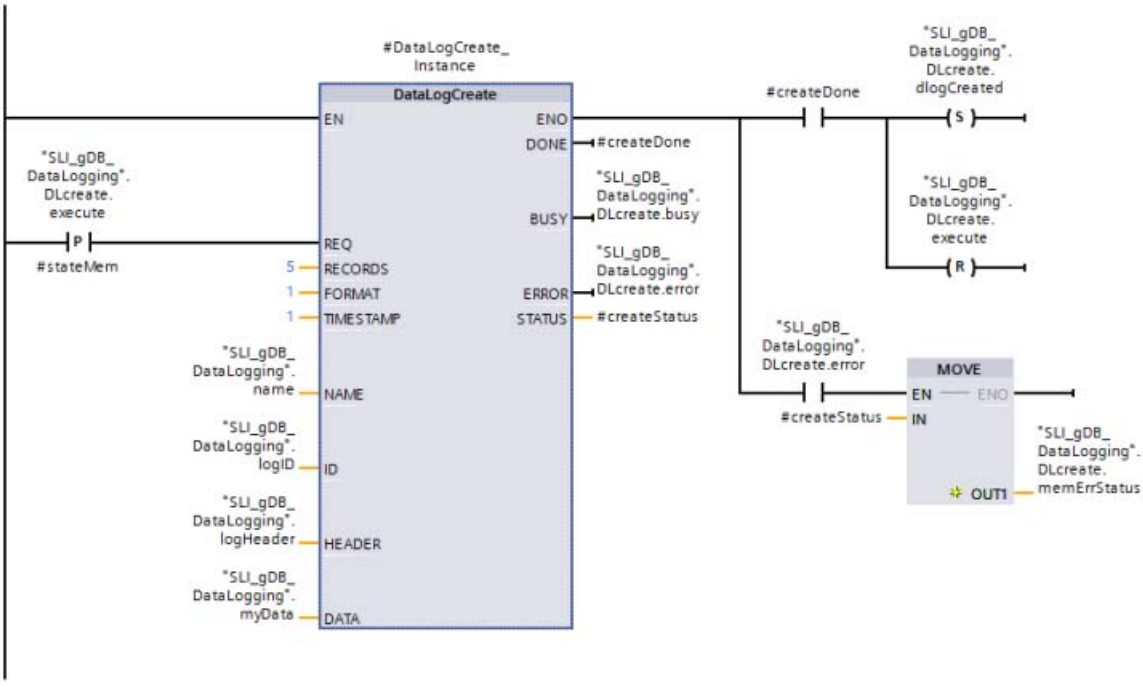

Network 2: You save the name and the ID of the data log by using the FC "SLI_FC_saveEntry_DataLog". Storing is carried out if the parameter DONE ("#createDone") has the signal state "TRUE".

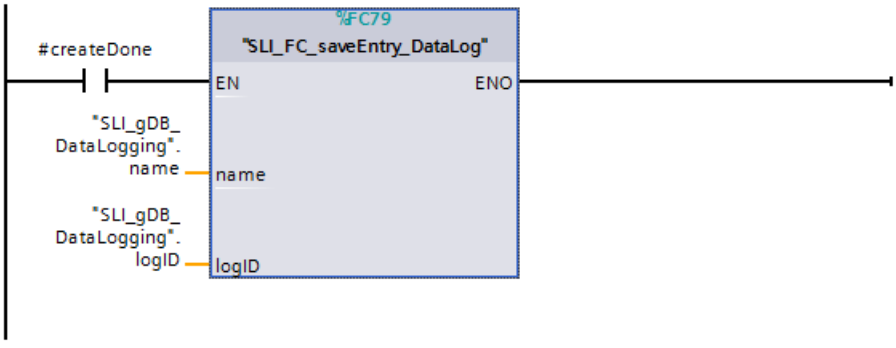

Network 3: A rising edge triggers the moment at which new process values are stored in the myData structure. This step is used to store the desired process values temporarily in the data block "SLI_gDB_DataLogging" .

In addition the desired data log is called up by using the FC "SLI_FC_callEntry_DataLog“. By means of the "callEntry" tag the data log is selected on the basis of its data log entry in the "dataLogEntries" array.

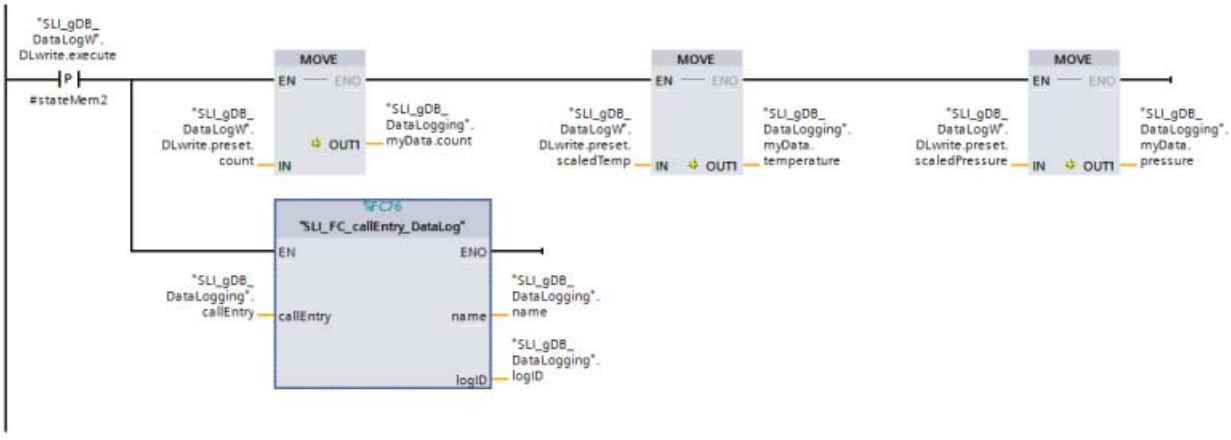

Network 4: If a data log was created and the value of the "logID" tag is not “0”, the input EN of DataLogWrite (Page 2683) is set. The reason for the query is that a generation process extends over several cycles and must be completed in order for a write operation to be performed. Writing is triggered for a data record by a rising edge at the input parameter REQ ("DLwrite.execute").

The success status of DataLogWrite ("#writeDone" is "TRUE“) is stored via the "DLwrite.memDone" tag. The "DLwrite.execute" tag is reset in the process.

In case of an error the status ("DLwrite.status") in the "DLwrite.memErrStatus" tag is saved. If the status is "16#0001“, the instruction "DataLogWrite" automatically initiates closing of the data log ("DLclose.execute" is "TRUE").

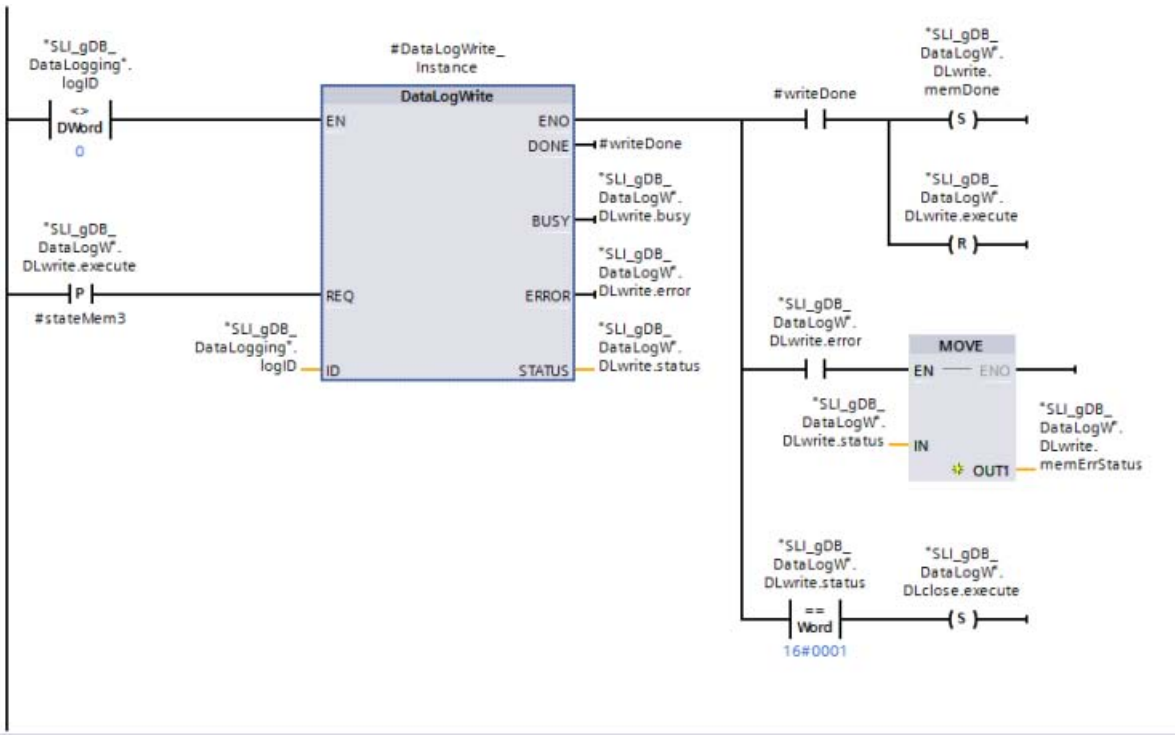

Note

Number of data records in the data log

The data log is created for five data records (see Network 1).

This means:

- After five data records the value "0001" is output at the instruction “DataLogWrite (Page 2683)“ at the parameter STATUS ("DLwrite.status").
- The last possible data record at the end of the file is created. When another data record is created, an older data record is overwritten.

Network 5: Before closing a data log select the data log to be closed (via "callEntry"). When the normally open contact ("DLclose.execute") supplies the signal state "TRUE", the desired data log is called on the basis of its entry in the “dataLogEntries“ array. In addition "#executeClose" is set to "TRUE". Create the following interconnections to this purpose.

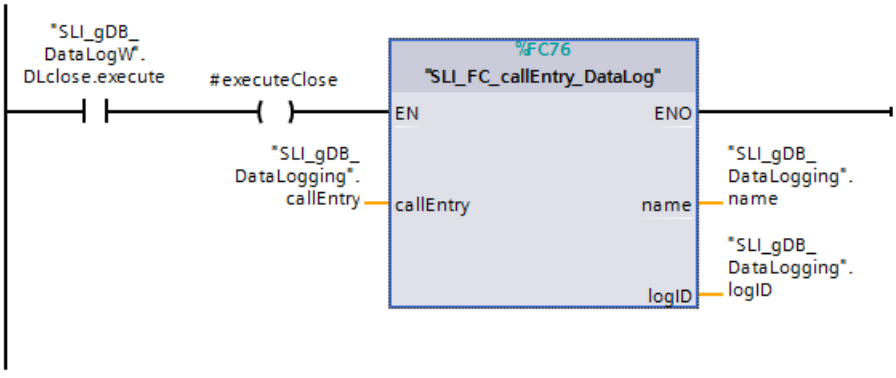

Network 6: Close the data log after the last data record has been written or processing of the data log is finished.

To this purpose the REQ ("#executeClose") input that executes the DataLogClose (Page 2685) instruction is set. When the data log is closed, no more data records are written.

The success status of DataLogClose ("#closeDone" is "TRUE") is stored via the "DLclose.memDone" tag.

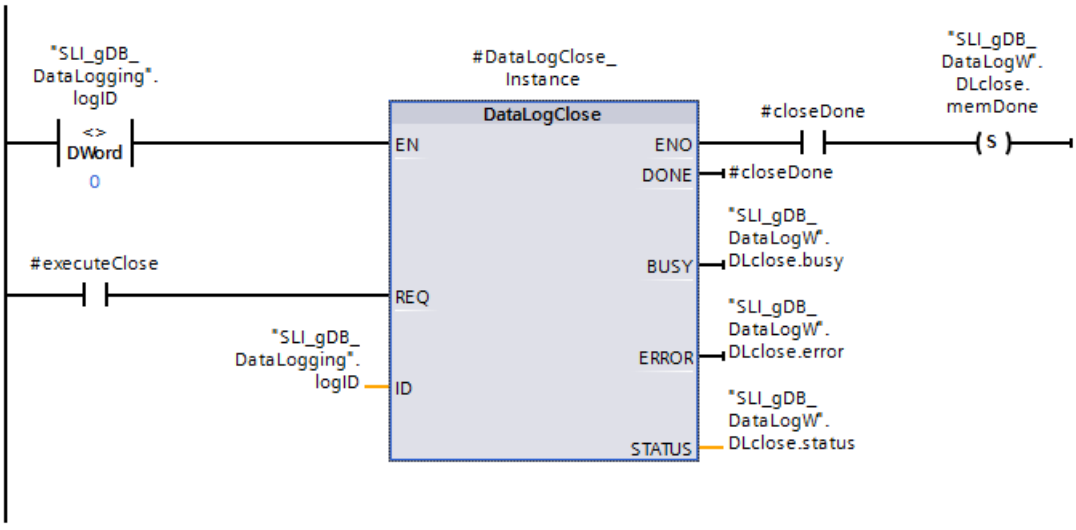

Network 7: Have the "DLclose.execute" tag reset automatically as follows.

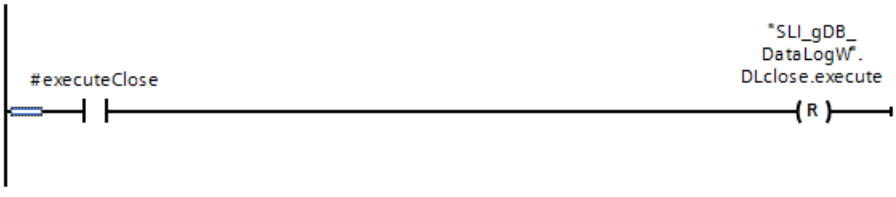

Network 8: On the basis of the following interconnection you store the “closed” status ("DLclose.memDone" is "TRUE") of the data logs in the data log entry ("dataLogEntries.DLclosed" is "TRUE").

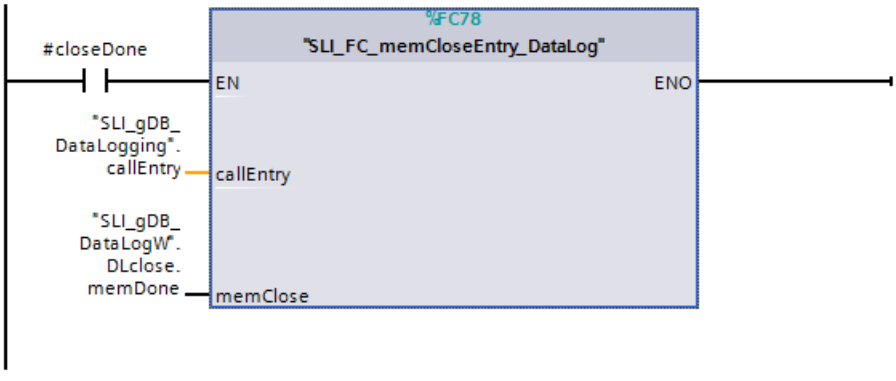

Network 9: Before opening a data log select the data log to be opened (via "callEntry"). When the normally open contact ("DLopen.execute") supplies the signal state "TRUE", the desired data log is called on the basis of its entry in the “dataLogEntries“ array. Create the following interconnections to this purpose.

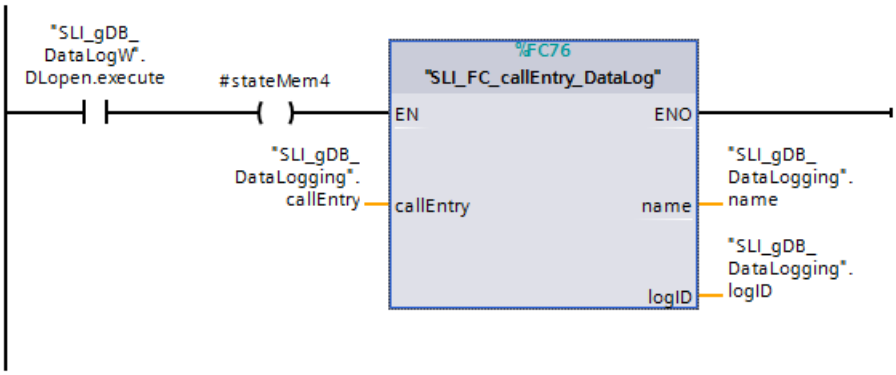

Network 10: The data log must be opened again with the DataLogOpen (Page 2679) instruction in order to write a data record again at a later time. If another data record is written with DataLogWrite (Page 2683) after this, the oldest data record is always the one that is overwritten.

The success status of DataLogOpen ("#openDone" is "TRUE") is stored via the "DLopen.memDone" tag. The negated result of "#openDone" is saved via the "#memOpenDone“ tag.

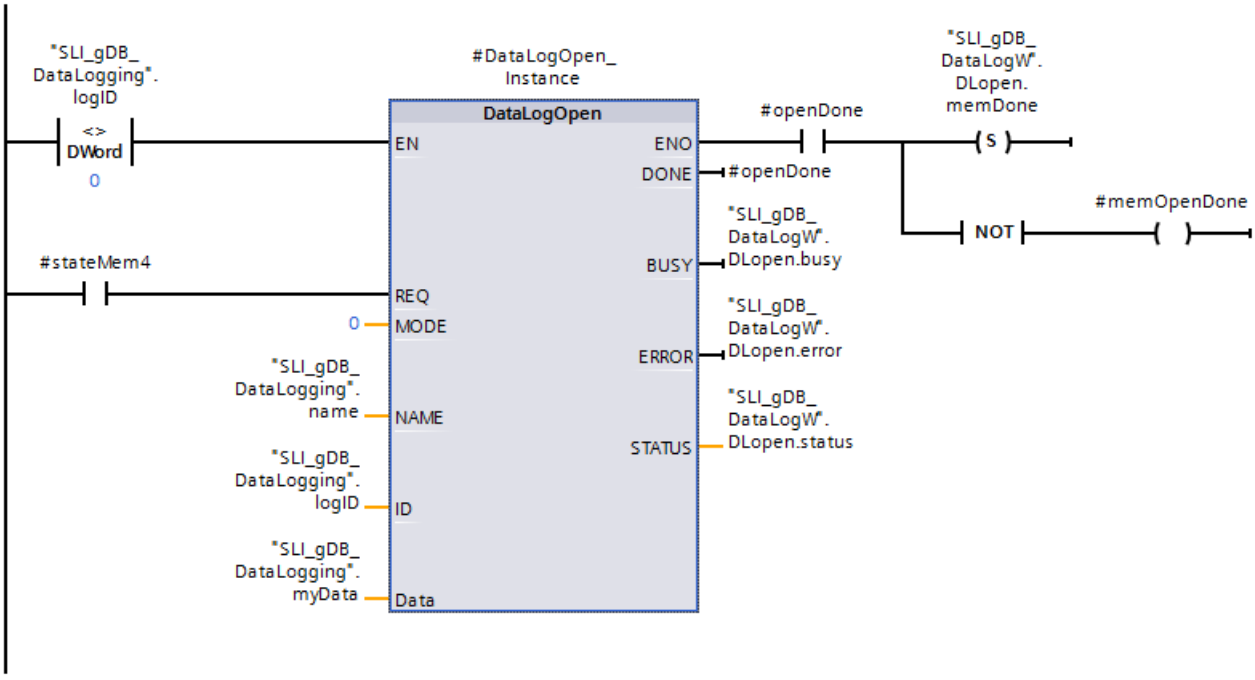

Network 11: Have the "DLopen.execute" tag reset automatically as follows.

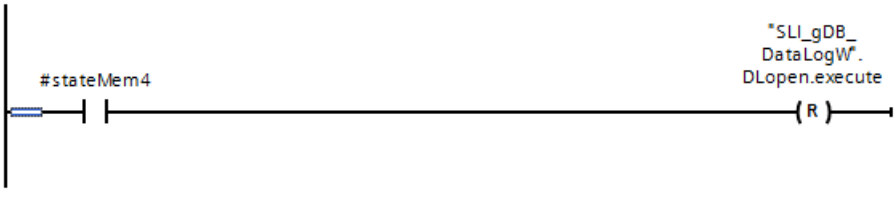

Network 12: On the basis of the following interconnection you store the “open” status of the data log in the data log entry ("#memClose" or "dataLogEntries.DLclosed" is "FALSE").

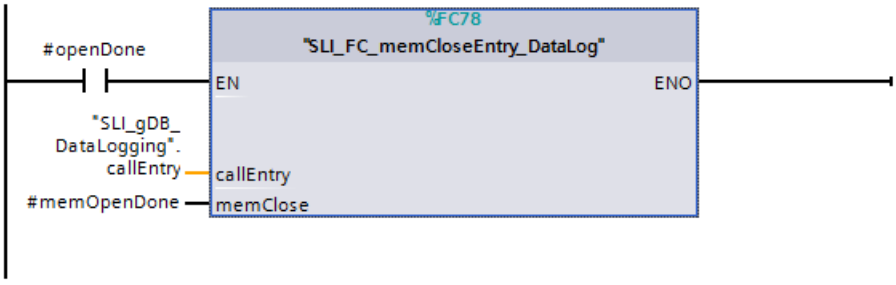

Network 13: To create a new file on the basis of a data log you first have to select the data log to be used (via "callEntry"). When the normally open contact ("DLnewfile.execute") supplies the signal state "TRUE", the desired data log is called on the basis of its entry in the “dataLogEntries“ array. Create the following interconnections to this purpose.

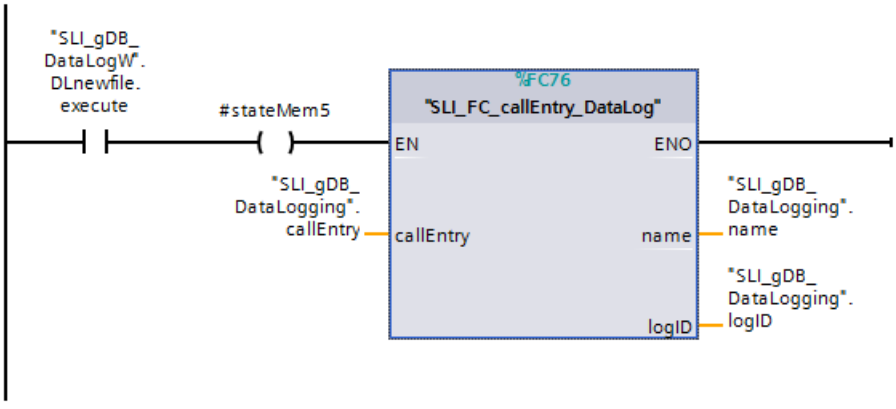

Network 14: If you do not want to overwrite the older data records, you can use the “DataLogNewFile (Page 2690)“ instruction to create a new data log with the same structure. To do this, enter the ID of the existing data log whose structure you want to copy at the ID parameter of the instruction. Once the “DataLogNewFile“ instruction has been executed, a new and unique ID value for the new data log is assigned.

The creation process is triggered by a rising edge at the input parameter REQ ("#stateMem5"). The success status of DataLogNewFile ("#newFileDone" is "TRUE") is stored via the "DLnewfile.memDone" tag. In case of an error the status ("#newFileStatus") in the "DLnewfile.memErrStatus" tag is saved.

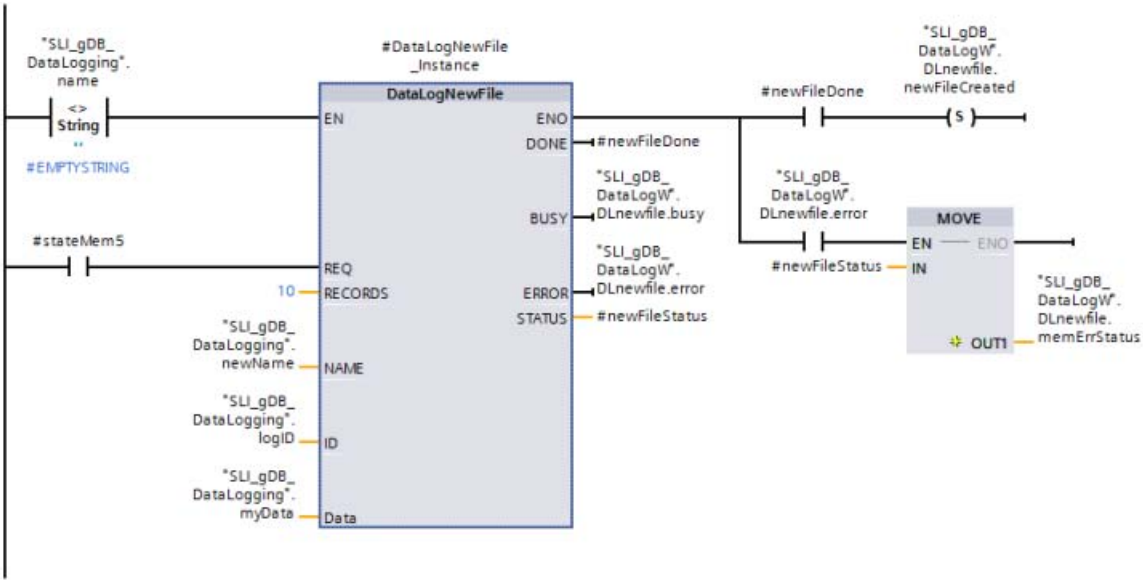

Note

Call over several cycles

Note that the call for DataLogNewFile (Page 2690) also extends over several cycles. For example, you could prevent the premature execution of the instruction "DataLogNewFile" by sampling the BUSY parameter of the DataLogWrite (Page 2683) instruction.

Network 15: Have the "DLnewfile.execute" tag reset automatically as follows.

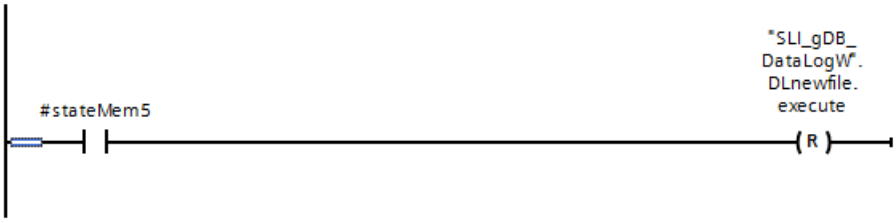

Network 16: You save the name and the ID of the data log by using the FC "SLI_FC_saveEntry_DataLog". Storing is carried out if the parameter DONE ("#newFileDone") has the signal state "TRUE".

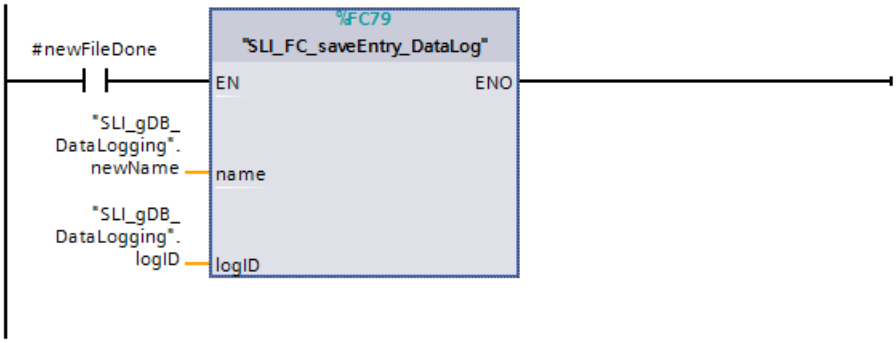

Network 17: To delete data log you first have to select the data log to be deleted (via "callEntry"). When the normally open contact ("DLdelete.execute") supplies the signal state "TRUE", the desired data log is called on the basis of its entry in the “dataLogEntries“ array. Create the following interconnections to this purpose.

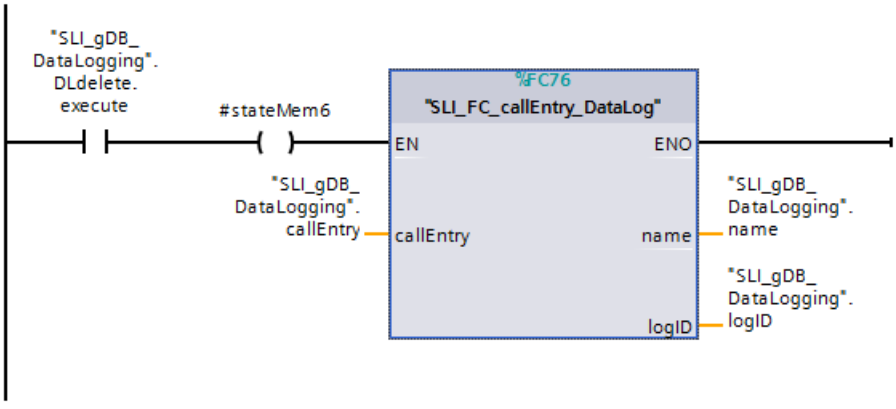

Network 18: Delete data logs that are no longer required by using the instruction "DataLogDelete (Page 2687)".

The deleting process for a data log is triggered by a rising edge at the input parameter REQ ("#stateMem6"). The success status of DataLogDelete (DONE is "TRUE") is stored via the "DLdelete.dlogDeleted" tag. In case of an error the status ("#deleteStatus") in the "DLdelete.memErrStatus" tag is saved.

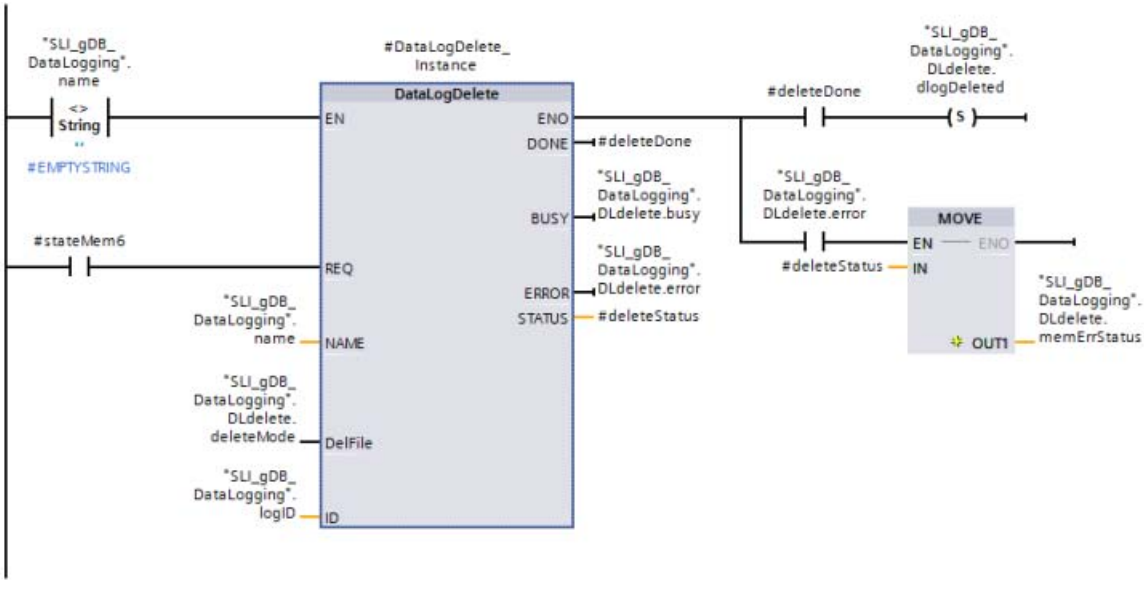

Network 19: Have the "DLdelete.execute" tag reset automatically as follows.

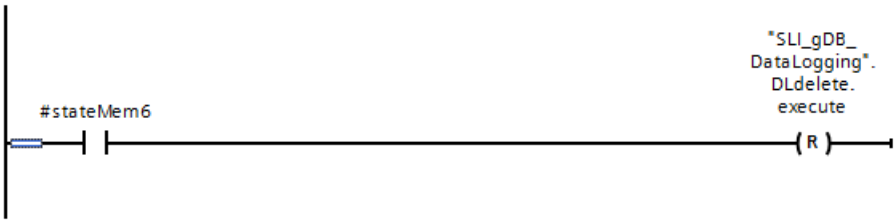

Network 20: You delete the entry of the data log by using the FC "SLI_FC_deleteEntry_DataLog". Deleting is carried out if the parameter DONE ("#deleteDone") has the signal state "TRUE".

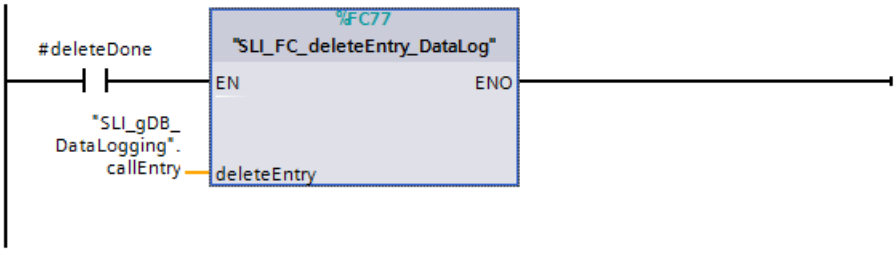

Result

Opening of written data via the web server

The data logs created with the example program are displayed via the web server. To do this, open the Web server using an Internet browser and open the "\DataLogs" directory.

Contents of the CSV files

- "5" is set as the maximum number of data records when creating the data log with the DataLogCreate instruction. If this number is not exceeded, all written data records are included in the data log.

	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	1	9/30/2010	20:26:56	1	9.86E+01	3.52E+01
3	2	9/30/2010	20:28:43	2	1.00E+02	3.73E+01
4	3	9/30/2010	20:29:03	3	9.99E+01	3.68E+01
5	4	9/30/2010	20:29:21	4	9.95E+01	3.64E+01
6	5	9/30/2010	20:30:19	5	9.92E+01	3.74E+01
7						

- If another data record is added to this, the oldest data record (Record 1) is overwritten.

	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	6	9/30/2010	20:32:03	6	9.86E+01	3.58E+01
3	2	9/30/2010	20:28:43	2	1.00E+02	3.73E+01
4	3	9/30/2010	20:29:03	3	9.99E+01	3.68E+01
5	4	9/30/2010	20:29:21	4	9.95E+01	3.64E+01
6	5	9/30/2010	20:30:19	5	9.92E+01	3.74E+01
7						

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

4.1.4.12 Data block functions

CREATE_DB: Create data block (S7-1200, S7-1500)

Description

The "CREATE_DB" instruction is used to create a new data block in the load and/or work memory.

The instruction "CREATE_DB" does not change the checksum of the user program.

A data block that was only generated in the work memory has the following properties:

- After a memory reset or POWER OFF / POWER ON this block no longer exists.
- When loading or when there is a STOP-RUN transition, its content remains unchanged.

Number of the data block

The data block created is assigned the smallest free number from the range defined by the LOW_LIMIT (low limit) and UP_LIMIT (high limit) parameters. This number is output at the DB_NUM parameter. The DB is generated if a number is found in the permissible range of the CPU.

To create a DB with a specific number, enter the same number for the high and low limit of the range to be specified. If a DB with the same number already exists in the work memory and/ or load memory, or if the DB exists as a copied version, the instruction will be terminated and an error message will be generated at the RET_VAL parameter.

Start values of the data block

The SRCBLK parameter is used to define start values for the DB that is to be created. The SRCBLK parameter is a pointer to a DB or a DB area from which you apply the start values. The DB addressed at the SRCBLK parameter must have been generated with standard access ("Optimized block access" attribute disabled).

- If the area specified at the SRCBLK parameter is larger than the DB generated, the values up to the length of the DB generated will be applied as start values.
- If the area specified at the SRCBLK parameter is smaller than the DB generated, the remaining values will be filled with "0".

To ensure data consistency, you must not change this data area while "CREATE_DB" is being executed (which means as long as the BUSY parameter has the value TRUE).

Note

SRCBLK points to a TEMP tag

TEMP tags of standard blocks are located in the temporary local data area. A reference in SRCBLK to a TEMP tag therefore results in the value 80A3 (the data block at the SRCBLK parameter was not created with standard access) being output as RET_VAL.

Functional description

The "CREATE_DB" instruction works asynchronously. Processing takes place across several calls. You start the job by calling "CREATE_DB" with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596)

Parameters

The following table shows the parameters of the "CREATE_DB" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Level-triggered control parameter "request to activate" REQ = 1: Request to generate the data block
LOW_LIMIT	Input	UINT	I, Q, M, D, L or constant	Low limit of the range for the assignment of a DB number. The smallest possible number is 60000.
UP_LIMIT	Input	UINT	I, Q, M, D, L or constant	High limit of the area used by "CREATE_DB" to assign a number to your DB (largest possible DB number: 60999)
COUNT	Input	UDINT	I, Q, M, D, L or constant	The count value specifies the number of bytes which you want to reserve for the DB generated. The number of bytes must be an even number. The maximum length is 65534 bytes.

Parameter	Declaration	Data type	Memory area	Description
ATTRIB	Input	BYTE	I, Q, M, D, L or constant	You use the first 4 bits of the byte at parameter ATTRIB to define the properties of the data block *:
				• Bit 0 = 0: Attribute "Only store in load memory" is not set. • Bit 0 = 1: Attribute "Only store in load memory" is set. With this setting, the DB takes up no space in the work memory and is not included in the program. The DB cannot be accessed with bit commands. When bit 0 = 1, the selection for bit 2 is irrelevant. To ensure compatibility with STEP 7 V5.x, bits 0 and 3 must be considered together (see below).
				• Bit 1 = 0: Attribute "Data block write-protected in the device" is not set. • Bit 1 = 1: Attribute "Data block write-protected in the device" is set.
				• Bit 2 = 0: DB is retentive (only for DBs generated in the load and in the work memory). The DB is regarded as retentive if at least one value has been set as retentive. • Bit 2 = 1: DB is not retentive Retentivity is not supported with DBs that are only stored in load memory or only in the work memory. If you call the "CREATE_DB" instruction with one of the two combinations "retentive and only load memory" or "retentive and only work memory" the DB to be generated will not be marked as retentive.
				• Bit 3 = 0: Creation of the DB either in the load memory or in the work memory (selection using bit 0, see above) • Bit 3 = 1: Creation of the DB both in the load memory and in the work memory (bit 0 irrelevant) To ensure compatibility with STEP 7 V5.x, bits 0 and 3 must be used in combination. When bit 3 = 1, bit 0 is irrelevant.
				Bit 0 / **Bit 3** / **DB generation** 0 / 0 / In work memory only 1 / 0 / In load memory only Irrelevant / 1 / Work and load memory
				• Bit 4 = 0 - No start values specified (input values at the SRCBLK parameter will be ignored). • Bit 4 = 1 - Specify start values (values correspond to the DB addressed by the SRCBLK parameter).
SRCBLK	Input	VARIANT	D	Pointer to the data block whose values will be used to initialize the data block to be generated.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The process is not yet complete.
DB_NUM	Output	DB_DYN (UINT)	I, Q, M, D, L	Number of the DB created.
* The properties selected here correspond to the attributes in the properties of a data block.				

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
0081	The destination area is greater than the source area. The source area is written completely to the destination area. The remaining bytes of the destination area remain unchanged.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source area is larger than the destination area. The complete destination area is written. The remaining bytes of the source area are ignored.
8092	The "Create data block" function is currently unavailable because • The "Compress user memory" function is currently active. • The maximum number of blocks on your CPU has already been reached.
8093	No data block or a data block that is not in the work memory is specified for the SRCBLK parameter.
8094	An invalid value was specified at parameter ATTRIB.
80A1	DB number error: • The numbers in the specified range (LOW_LIMIT to UP_LIMIT) are assigned or are not in the permissible range of the CPU • Low limit > high limit
80A2	DB length error: • The length is "0" • The length is an odd number • The length is greater than permitted by the CPU
80A3	The data block at the SRCBLK parameter was not created with standard access.
80B1	There is no DB number free.
80B2	Insufficient free memory available.
80B4	The memory card is write-protected.
80BB	Not enough load memory.
80C3	The maximum number of simultaneously active "CREATE_DB" instructions has already been reached.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for CREATE functions (Page 2729).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

READ_DBL: Read from data block in the load memory (S7-1200, S7-1500)

Description

With the instruction, you copy a DB or an area of a DB in load memory (Micro Memory Card) to the data area of a destination DB. The destination data block must be runtime-relevant, which means it must not be created with the Only store in load memory attribute. The content of the load memory is not changed during the copy process.

To ensure data consistency, you must not change the destination area while "READ_DBL" is being executed (i.e., as long as the BUSY parameter has the value TRUE).

The parameters SRCBLK and DSTBLK (source and destination blocks) must both be VARIANT data type. The following restrictions hereby apply:

- Valid for standard memory areas: If you want to interconnect a parameter of VARIANT data type (source or destination area) with a tag of BOOL data type or an ARRAY of BOOL, you have the following options:
  - You can address it symbolically
    Example: Parameter SRCBLK: "Data_block".myArray
  - You can address it absolutely with help of the ANY pointer. However, you must note that the specified length of the area must be divisible by 8, or the instruction will otherwise not be executed.
    Example: Parameter SRCBLK: P#DB123.DBX456.0 BOOL 1000
- For a VARIANT pointer of type STRING, the length must be equal to 1.
- The source and destination blocks must have been created with the same block access, i.e. both must use either the "Optimized" or "Standard" access type.
- When the source and target DB blocks have optimized access, you must not use the STRUCT data type.

Note

"READ_DBL" is processed asynchronously and is therefore, it is not suitable for frequent (or cyclic) reading of tags in the load memory.

Once started, a job is always completed. If the maximum number of simultaneously active "READ_DBL" instructions is reached and you call "READ_DBL" once again at this time in a priority class having higher priority, error code W#16#80C3 will be returned. Consequently it does not make sense to restart the high-priority job right away.

Functional description

The "READ_DBL" instruction works asynchronously. Processing takes place across several calls. You start the job by calling "READ_DBL" with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596)

Parameters

The following table shows the parameters of the "READ_DBL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
SRCBLK	Input	VARIANT	D	Pointer to data block in the load memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
DSTBLK	Output	VARIANT	D	Pointer to the data block in the work memory that is written to

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
0081	The destination area is greater than the source area. The source area is written completely to the destination area; the remaining bytes of the destination area will not be changed.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8x51	Note: This error code is only for S7-1200 CPUs. Data type error in the data block.
8081	The source area is larger than the destination area. The complete destination area is written. The remaining bytes of the source area are ignored.
8082	• Destination DB type different from source DB type (optimized/standard access). • The destination and source DB were both created with optimized access. You have, however, used structures with the destination and source DB that lead to a different arrangement of the data when compiled. (Note: Use UDTs instead.)
8093	Note: This error code is only for S7-1500 CPUs. No data block or a data block that is not in the work memory is specified for the DSTBLK parameter.
80B1	Note: This error code is only for S7-1500 CPUs. At parameter SRCBLK, only data blocks that are located in the load memory are permitted.

Error code* (W#16#...)	Description
8xB1	Note: This error code is only for S7-1200 CPUs. No data block is specified for the SRCBLK parameter, or the data block specified there is not a load memory object.
80B4	DSTBLK points to a DB with F-attribute (that must not be written)
8xC0	Note: This error code is only for S7-1200 CPUs. The destination DB is currently being processed by another instruction or a communication function.
80C3	The maximum number of simultaneously active "READ_DBL" instructions has already been reached.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you read out the contents of a DB in the load memory to a DB in the work memory.

To store the data for the interconnection, create three tags in a global data block.

SLI_gDB_READ_DBL

	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	busy	Bool	false
4	returnValue	Int	0

To store the send data, create two global data blocks with three tags each of the INT data type.

- For receive data in the load memory:

SourceDB_LoadMemory

	Name	Data type	Offset	Start value
1	Static			
2	MySourceValue1	Int	0.0	111
3	MySourceValue2	Int	2.0	222
4	MySourceValue3	Int	4.0	333

- For receive data in the work memory:

TargetDB

	Name	Data type	Offset	Start value
1	Static			
2	MyTargetValue1	Int	0.0	0
3	MyTargetValue2	Int	2.0	0
4	MyTargetValue3	Int	4.0	0

Create the following interconnections in an FB "SLI_FB_READ_DBL".

Network 1: Interconnect the parameters of the "READ_DBL" instruction as follows.

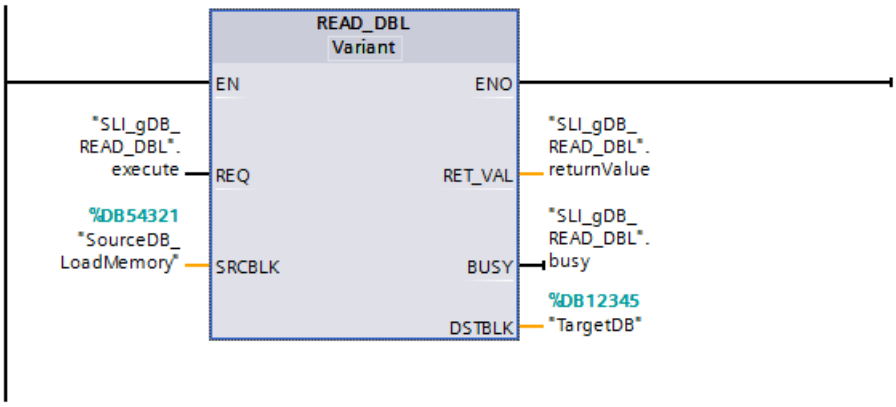

Network 2: You can stop the processing of READ_DBL as follows.

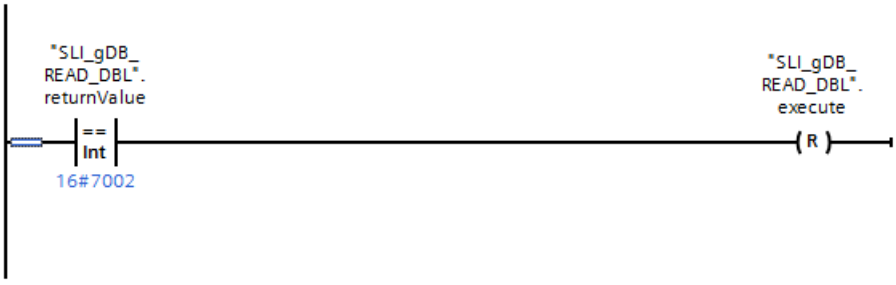

If the normally open contact ("execute") supplies the signal state "TRUE", the "READ_DBL" instruction is executed. The DB is stored in the load memory with the input parameter SRCBLK ("SourceDB_LoadMemory").
The DB is stored in the work memory with the output parameter DSTBLK ("TargetDB"). The instruction "READ_DBL" calls the data to be transferred in the load memory (SRCBLK) and reads the data in the work memory (DSTBLK).

The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 1857)

WRIT_DBL: Write to data block in the load memory (S7-1200, S7-1500)

Description

The instruction "WRIT_DBL" is used to transfer the contents of a DB or a DB area from the work memory to a DB or a DB area in the load memory (Micro Memory Card). The source DB must be runtime-relevant, which means it must not be created with the Only store in load memory attribute.

Note

Start value behavior during "Upload from device"

The start values that you changed with the "WRIT_DBL" instruction are lost when you perform the "Upload from device" action.

To ensure data consistency, it is not permitted to change the source range while "WRIT_DBL" is being executed (i.e., as long as the BUSY parameter has the value TRUE).

The parameters SRCBLK and DSTBLK (source and destination blocks) must both be VARIANT data type. The following restrictions hereby apply:

- For a VARIANT pointer of type BOOL, the length must be divisible by 8.
- For a VARIANT pointer of type STRING, the length must be equal to 1.
- The source and destination blocks must have been created with the same block access, i.e. both must use "Optimized block access" or optimized access must be disabled for both.
- When the source and target DB blocks have optimized access, you must not use the STRUCT data type.

The "WRIT_DBL" instruction does not change the checksum of the user program if you write a DB that was created using an instruction. However, when a loaded DB is written, the first entry in this DB changes the checksum of the user program.

Note

"WRIT_DBL" is not suitable for frequent (or cyclical) writing of tags in the load memory. This is because the memory card technology limits the number of write accesses that can be made to a memory card.

Functional description

The "WRIT_DBL" instruction works asynchronously. Processing takes place across several calls. You start the job by calling "WRIT_DBL" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596)

Parameters

The following table shows the parameters of the "WRIT_DBL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Write request
SRCBLK	Input	VARIANT	D	Pointer to the DB in the work memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.
DSTBLK	Output	VARIANT	D	Pointer to the data block in the load memory that is written to

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
0081	The destination area is greater than the source area. The source area is written completely to the destination area; the remaining bytes of the destination area will not be changed.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8051	Note: This error code is only for S7-1200 CPUs. Type error data block.
8251	Note: This error code is only for S7-1200 CPUs. Data type error in the source DB
8551	Note: This error code is only for S7-1200 CPUs. Data type error in the target DB
8081	The source area is larger than the destination area. The complete destination area is written. The remaining bytes of the source area are ignored.
8082	Note: This error code is only for S7-1500 CPUs. • Destination DB type different from source DB type (optimized/non-optimized access). • The destination and source DB were both created with optimized access. You have, however, used structures with the destination and source DB that lead to a different arrangement of the data when compiled. (Note: Use UDTs instead.)
8093	Note: This error code is only for S7-1500 CPUs. No data block or a data block that is not in the work memory is specified for the SRCBLK parameter.
80B1	Note: This error code is only for S7-1500 CPUs. At parameter DSTBLK, only data blocks that are located in the load memory are permitted.

Error code* (W#16#...)	Description
82B1	Note: This error code is only for S7-1200 CPUs. No data block is specified for the SRCBLK parameter.
85B1	Note: This error code is only for S7-1200 CPUs. For parameter DSTBLK, no data block is specified or the data block specified there is not in the load memory.
80B4	• The memory card is write-protected. • DB with F-attribute must not be read.
80BB	Insufficient load memory available.
82C0	Note: This error code is only for S7-1200 CPUs. The source DB is currently being processed by another instruction or a communication instruction.
85C0	Note: This error code is only for S7-1200 CPUs. The destination DB is currently being processed by another instruction or a communication instruction.
80C3	The maximum number of simultaneously active "WRIT_DBL" instructions has already been reached.
General error codes	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you transfer the contents of a DB in the work memory to a DB in the load memory. To verify successful transfer to the DB, read the contents in another DB in the work memory.

To store the data for the interconnection, create five tags in a global data block.

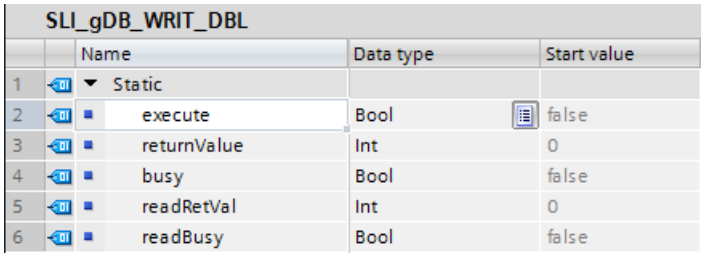

SLI_gDB_WRIT_DBL

	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	returnValue	Int	0
4	busy	Bool	false
5	readRetVal	Int	0
6	readBusy	Bool	false

To store the data to be transmitted, create three global data blocks with one "myData" tag each (Array of INT data type):

- A DB for send data in the work memory (SLI_srcDB_WRIT_DBL).

SLI_srcDB_WRIT_DBL			
	Name	Data type	Start value
1	Static		
2	myData	Array[0..3] of Int	
3	myData[0]	Int	1
4	myData[1]	Int	2
5	myData[2]	Int	3
6	myData[3]	Int	4

- A DB for receive data in the load memory (SLI_dstDB_WRIT_DBL).
- A DB for receive data in the work memory (SLI_finDB_WRIT_DBL).

Network 1:

Interconnect the parameters of the "WRIT_DBL" instruction as follows.

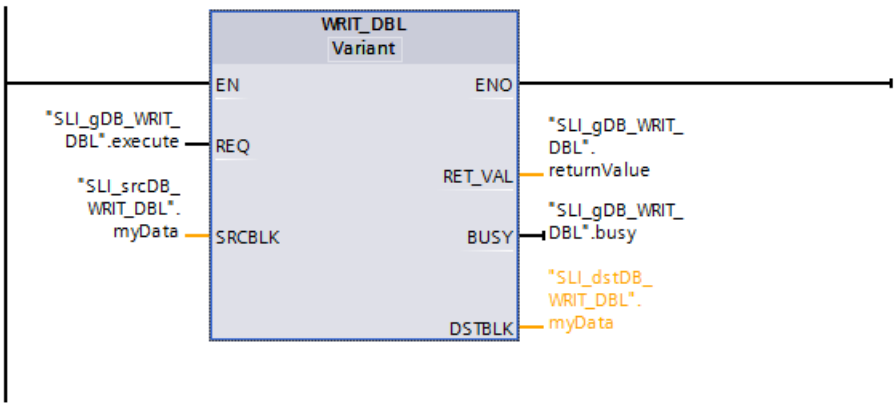

Network 2:

Interconnect the parameters of the "READ_DBL" instruction as follows.

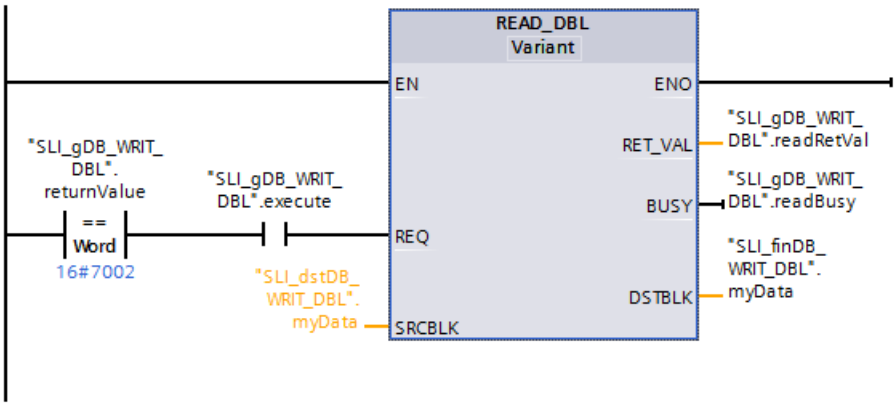

Network 3:

Reset the tag "execute" as follows:

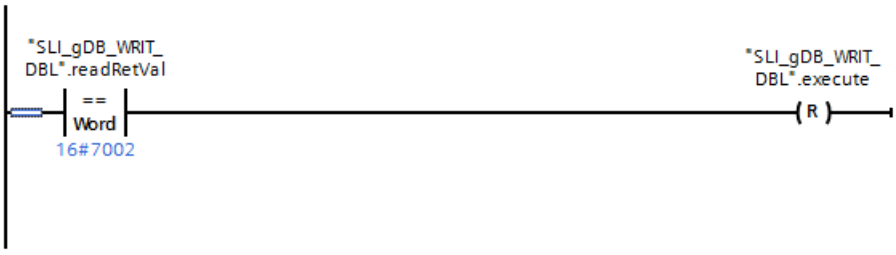

"WRIT_DBL" instruction:

If the normally open contact ("execute") supplies the signal state "TRUE", the "WRIT_DBL" instruction is executed. The DB is stored in the work memory with the input parameter SRCBLK ("SLI_srcDB_WRIT_DBL.myData"). The DB is stored in the load memory with the output parameter DSTBLK ("SLI_dstDB_WRIT_DBL.myData").

The "WRIT_DBL" instruction calls the data to be transferred in the work memory (SRCBLK) and writes the data to the required location in the load memory (DSTBLK). The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

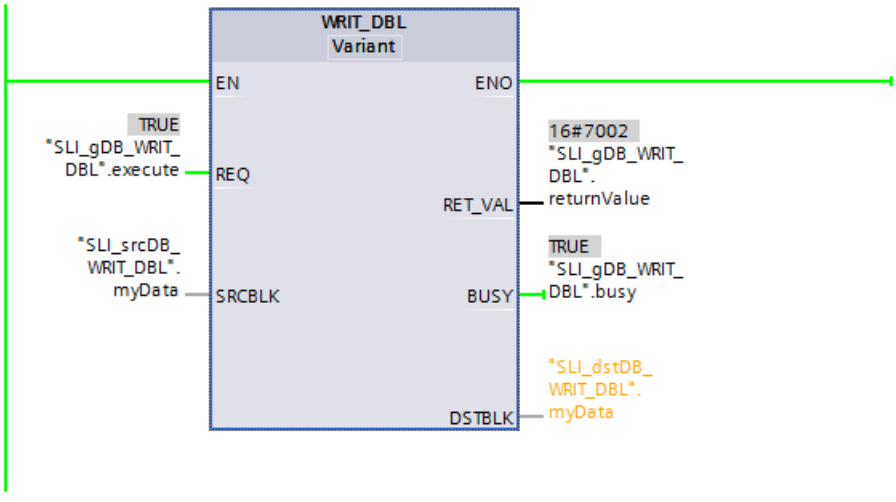

"READ_DBL" instruction:

When processing of WRIT_DBL took place without errors and the normally open contact ("execute") supplies the signal state "TRUE", the "READ_DBL" instruction is executed. The DB is stored in the load memory with the input parameter SRCBLK ("SLI_dstDB_WRIT_DBL.myData"). The DB for the receive data is stored with the output parameter DSTBLK ("SLI_finDB_WRIT_DBL.myData").

The "READ_DBL" instruction calls the data to be transferred in the load memory (SRCBLK) and reads the data at the required location in the work memory (DSTBLK). The output parameter RET_VAL ("readRetVal") indicates that processing took place without errors.

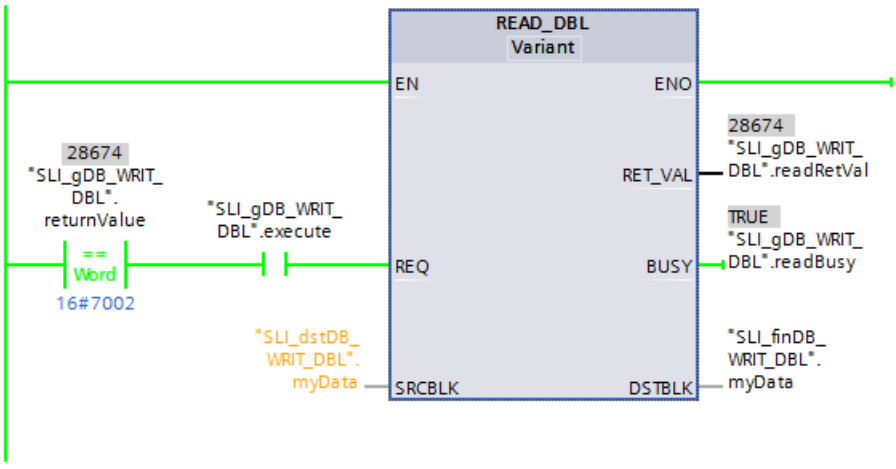

The DB for the receive data in the work memory (SLI_finDB_WRIT_DBL) displays the transmitted data.

SLI_finDB_WRIT_DBL

		Name	Data type	Start value	Monitor value
1		Static			
2		myData	Array[0..3] o...		
3		myData[0]	Int	0	1
4		myData[1]	Int	0	2
5		myData[2]	Int	0	3
6		myData[3]	Int	0	4

The tag "execute" is reset after the successful processing of READ_DBL (see Network 3). The processing of READ_DBL and WRIT_DBL is stopped.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

ATTR_DB: Read data block attribute (S7-1200, S7-1500)

Description

You use the instruction "ATTR_DB" to obtain information about a data block (DB) located in the work memory of the CPU. The instruction determines the attributes set at the ATTRIB parameter for the DB selected.

The length cannot be read out for data blocks with optimized access and data blocks that are only in load memory. In these cases, the DB_LENGTH parameter has the value "0".

Do not apply ATTR_DB to data blocks with optimized access and activated memory reserve.

Do not read out the data blocks for Motion Control with the "ATTR_DB" instruction. The error code 80B2 is output for this.

Parameters

The following table shows the parameters of the "ATTR_DB" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Request to read block attributes
DB_NUM-BER	Input	DB_ANY	I, Q, M, D, L or constant	Number of the DB to be tested
RET_VAL	Output	INT	I, Q, M, D, L	Error information
DB_LENGTH	Output	UDINT	I, Q, M, D, L	• Number of data bytes which the selected DB contains. • "0" for data blocks with optimized access and data blocks that are only in load memory.
ATTRIB	Output	BYTE	I, Q, M, D, L	DB properties:
				• Bit 0*= 0: Attribute "Only store in load memory" is not set. • Bit 0*= 1: Attribute "Only store in load memory" is set.
				• Bit 1 = 0: Attribute "Data block write-protected in the device" is not set. • Bit 1 = 1: Attribute "Data block write-protected in the device" is set.
				If bit 0 = 1, then bit 2 is irrelevant and gets the value 1. • Bit 2 = 0: Retentive - The DB is regarded as retentive if at least one value has been set as retentive. • Bit 2 = 1: Not retentive - The complete DB is not retentive.
				• Bit 3*= 0: The DB is either in the load memory (bit 0 = 1) or in the work memory (bit 0 = 0). • Bit 3*= 1: The DB is generated in both the load and the work memory
* The relationship between bit 0 and bit 3 is explained in the parameters of the instruction "CREATE_DB: Create data block (Page 2710)".				

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
80A1	Error in input parameter DB_NUMBER: the actual parameter selected • Is "0" • Is greater than the maximum permitted DB number for the CPU used.
80B1	The DB with the specified number does not exist on the CPU.
80B2	Data blocks of Motion Control technology objects cannot be read with the "ATTR_DB" instruction.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you read the status of a data block.

You create a DB with a "myData" tag (Array[0..3] of INT data type).

SLI_srcDB_ATTR_DB

	Name	Data type	Offset	Start value
1	Static			
2	myData	Array[0..3] of Int	0.0	
3	myData[0]	Int	0.0	1
4	myData[1]	Int	2.0	2
5	myData[2]	Int	4.0	3
6	myData[3]	Int	6.0	4

For standard access, disable optimized data access in the DB properties.

Attributes

- [ ] Only store in load memory
- [ ] Data block write-protected in the device
- [ ] Optimized block access

To store the data for the interconnection, create four tags in a global data block.

SLI_gDB_ATTR_DB

	Name	Data type	Start value
1	Static		
2	startAttr	Bool	false
3	attrRetVal	Int	0
4	readLenByte	UDInt	0
5	readAttrBool	Array[0..7] of Bool	

Interconnect parameters: "SLI_itemiseByte_ATTR_DB" FC

To better evaluate the value of the output parameter ATTRIB, create the FC "SLI_itemiseByte_ATTR_DB".

Create the following local tag in it.

SLI_itemiseByte_ATTR_DB		
	Name	Data type
1	Input	
2	inByte	Byte

Network 1: Create interconnections for all bits (0..7) of the local tag according to the example of network 1.

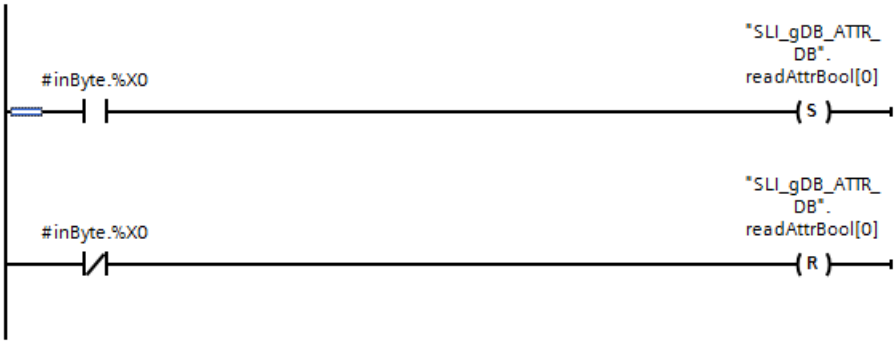

Interconnect parameters: "SLI_FB_ATTR_DB" FB

Create an FB "SLI_FB_ATTR_DB" for the call of ATTR_DB. Create the following local tag in it.

7	Static		
8	statAttByte	Byte	16#0

Network 1: Interconnect the parameters of the "ATTR_DB" instruction as follows.

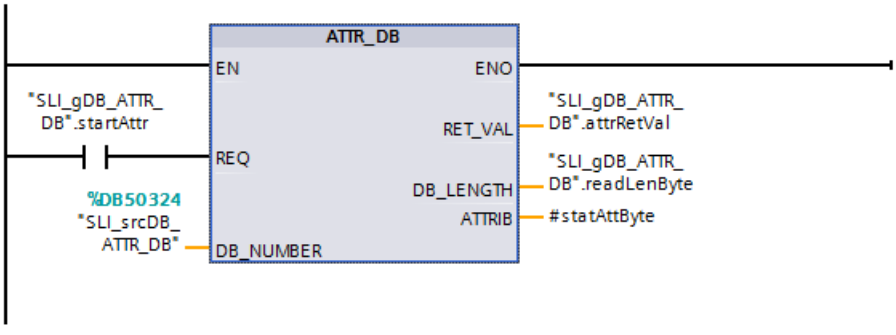

Network 2: Interconnect the FC "SLI_itemiseByte_ATTR_DB" as follows.

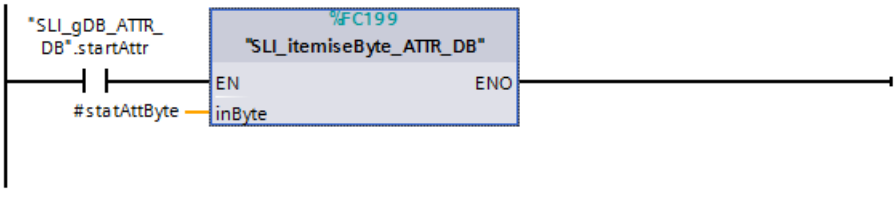

Result of FB "SLI_FB_ATTR_DB"

If the normally open contact ("startAttr") supplies the signal state "TRUE", the "ATTR_DB" instruction is executed. The number of the DB to be evaluated is stored with the input parameter DB_NUMBER ("SLI_srcDB_ATTR_DB"). The "ATTR_DB" instruction calls the DB based on the number and outputs its status at the output parameter ATTRIB ("#statAttrByte"). The length of the read data (4 INT) of the data block in BYTE (value "8") is output at the output parameter DB_LENGTH ("readLenByte").

The hexadecimal value of the output parameter ATTRIB is converted into a bit string using FC "SLI_itemiseByte_ATTR_DB". The bit sequence "00110000" (Bit 0 to 7) means:

- The DB exists not only in the load memory.
- The DB is not write-protected.
- The DB is not retentive.
- The DB is generated in both the load and the work memory.

The output parameter RET_VAL ("attrRetVal") indicates that processing took place without errors.

SLI_gDB_ATTR_DB

	Name	Data type	Start value	Monitor value
1	Static			
2	startAttr	Bool	false	TRUE
3	attrRetVal	Int	0	0
4	readLenByte	UDInt	0	8
5	readAttrBool	Array[0..7] of Bool		
6	readAttrBool[0]	Bool	false	FALSE
7	readAttrBool[1]	Bool	false	FALSE
8	readAttrBool[2]	Bool	false	TRUE
9	readAttrBool[3]	Bool	false	TRUE
10	readAttrBool[4]	Bool	false	FALSE
11	readAttrBool[5]	Bool	false	FALSE
12	readAttrBool[6]	Bool	false	FALSE
13	readAttrBool[7]	Bool	false	FALSE

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

DELETE_DB: Delete data block (S7-1200, S7-1500)

Description

You use the instruction "DELETE_DB" to delete a data block (DB) that was created from the user program by calling the instruction "CREATE_DB (Page 2710)".

If the data block was not created with "CREATE_DB", the error code W#16#80B5 is output at the RET_VAL parameter.

The selected data block is not deleted immediately, but rather at the cycle control point after execution of the cycle OB.

Functional description

The "DELETE_DB" instruction works asynchronously. Processing takes place across several calls. You start the interrupt transfer by calling the instruction with REQ = 1.

Output parameter BUSY and bytes 2 and 3 of output parameter RET_VAL show the status of the job.

When the output parameter BUSY has the value FALSE, the deletion of the data block is completed.

Parameters

The following table shows the parameters of the "DELETE_DB" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ =1: Request to delete the DB with the number in the DB_NUMBER parameter
DB_NUMBER	Input	UINT	I, Q, M, D, L or constant	Number of the DB to be deleted
RET_VAL	Output	INT	I, Q, M, D, L	Error information (see "RET_VAL parameter")
BUSY	Output	BOOL	I, Q, M, D, L	BUSY =1: The process is not yet complete.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
80A1	Error in input parameter DB_NUMBER: • The value at the parameter is "0". • The value at the parameter is greater than the maximum permitted DB number for the CPU used.
80B1	The DB with the specified number does not exist on the CPU.
80B4	The DB cannot be deleted because the memory card of the CPU is write-protected.
80B5	The DB was not created using "CREATE_DB".
80BB	Not enough load memory.
80C3	The "Delete a DB" function cannot be executed at this time due to a temporary resource bottleneck.

Error code* (W#16#...)	Explanation
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for CREATE functions (Page 2729).

You can find additional information and the program code for the above named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Difference between synchronous and asynchronous instructions (Page 596)

Program example for CREATE functions

Introduction

In the following example, you create a DB in the load memory, read its contents, and then delete the DB.

Requirement

Create a global DB ("SLI_srcDB_CREATE_DB") to serve as a template for the new DB.

SLI_srcDB_CREATE_DB

	Name	Data type	Offset	Start value
1	Static			
2	myData	Array[0..3] of Int	0.0	
3	myData[0]	Int	0.0	1
4	myData[1]	Int	2.0	2
5	myData[2]	Int	4.0	3
6	myData[3]	Int	6.0	4
7	readData	Array[0..3] of Int	8.0	
8	readData[0]	Int	0.0	0
9	readData[1]	Int	2.0	0
10	readData[2]	Int	4.0	0
11	readData[3]	Int	6.0	0

The properties of the DB are as follows.

Attributes

- [ ] Only store in load memory
- [ ] Data block write-protected in the device
- [ ] Optimized block access

To store the data for the interconnection, create 14 tags in a global data block ("SLI_gDB_CREATE_DB").

SLI_gDB_CREATE_DB

	Name	Data type	Start value
1	Static		
2	startCreate	Bool	false
3	lowLim	UInt	60003
4	hiLim	UInt	60003
5	setLenByte	UDInt	8
6	setAttrBool	Array[0..7] of Bool	
7	setAttrBool[0]	Bool	1
8	setAttrBool[1]	Bool	0
9	setAttrBool[2]	Bool	1
10	setAttrBool[3]	Bool	0
11	setAttrBool[4]	Bool	1
12	setAttrBool[5]	Bool	false
13	setAttrBool[6]	Bool	false
14	setAttrBool[7]	Bool	false
15	createRetVal	Int	0
16	createBusy	Bool	false
17	memNbrNewDB	UInt	0
18	startRead	Bool	false
19	rdblRetVal	Int	0
20	rdblBusy	Bool	false
21	startDelete	Bool	false
22	delRetVal	Int	0
23	delBusy	Bool	false

To store the data, also create the following local tags in the FB "SLI_FB_CREATE_DB".

7	Static			
8	statAttByte	Byte	16#0	Non-ret...
9	nbrNewDB	UInt	0	Non-retain
10	Temp			
11	tempCreate	Bool		
12	tempRead	Bool		
13	tempDelete	Bool		

Interconnect parameters: "SLI_itemiseByte_CRDfunctions" FC

To convert the values of the "setAttrBool" tag into the "#statAttByte" tag, create the FC "SLI_itemiseByte_CRDfunctions".

Create the following local tag in it.

Network 1: Create interconnections for all bits (0..7) of the local tag according to the example of network 1.

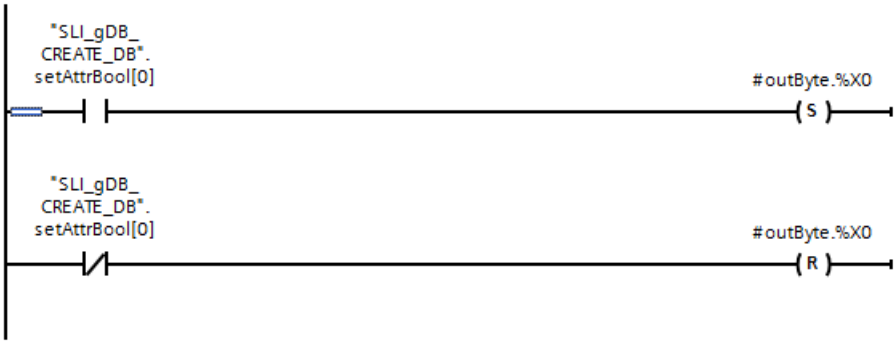

Interconnect parameters: "SLI_FB_CREATE_DB" FB

Network 1: Interconnect the parameters of the "SLI_itemiseByte_CRDfunctions" FC as follows.

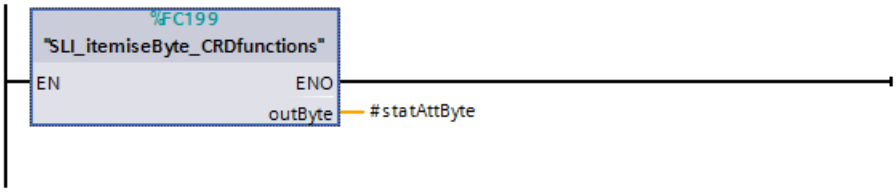

Network 2: Interconnect the parameters of the "CREATE_DB" instruction as follows.

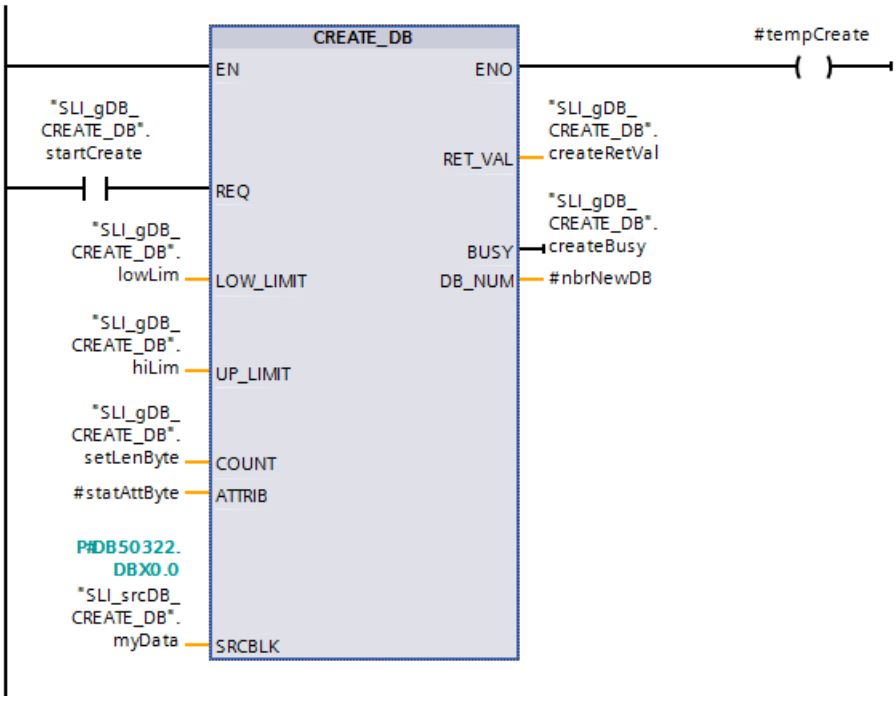

Network 3: To store the number of the newly created DB, create the following interconnections.

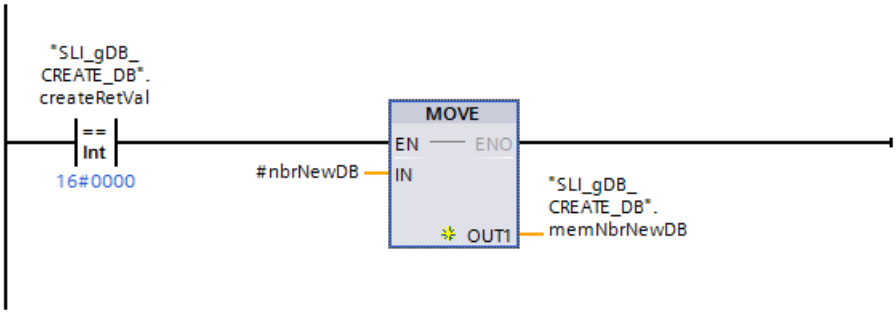

Network 4: To end the execution of CREATE_DB once again, create the following interconnections.

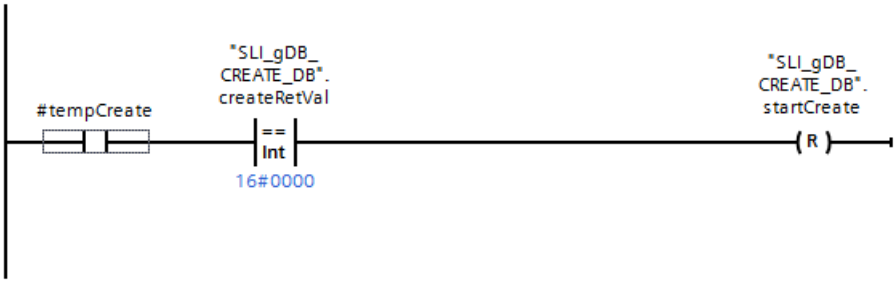

Network 5: Interconnect the parameters of the "READ_DBL" instruction as follows.

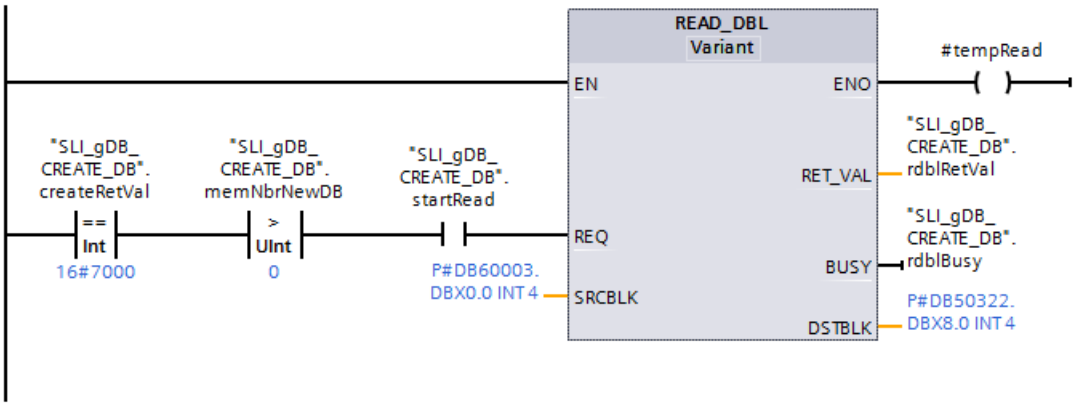

Network 6: To end the execution of READ_DBL once again, create the following interconnections.

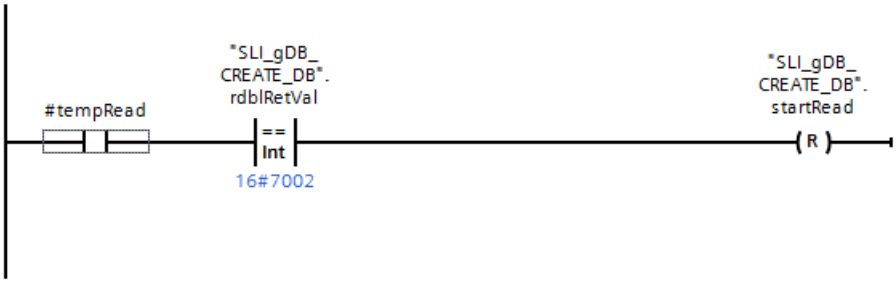

Network 7: Interconnect the parameters of the "DELETE_DB" instruction as follows.

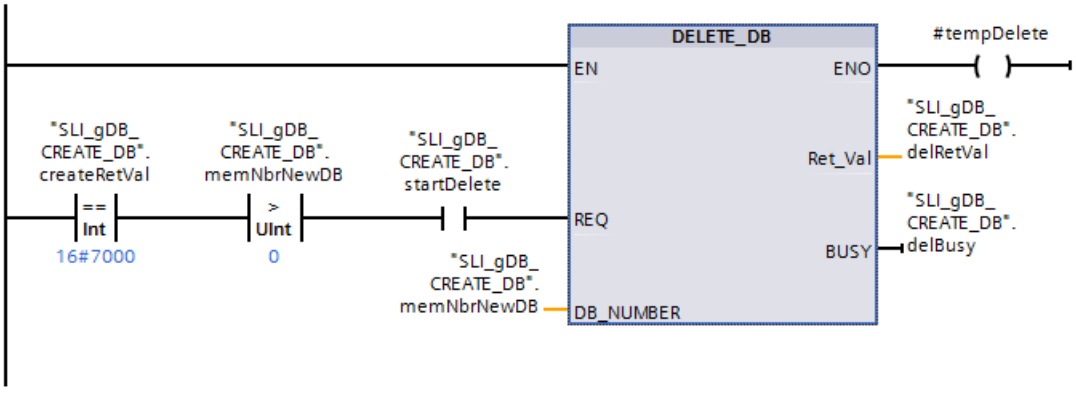

Network 8: To end the execution of DELETE_DB once again, create the following interconnections.

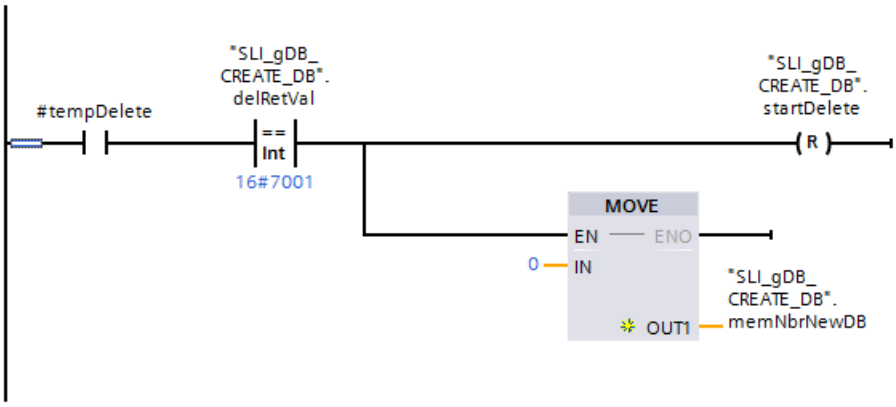

Result of network 1

With an AT construct, the properties ("setAttrBool") stored in the Array of BOOL data type(#statAttBool) are converted into BYTE (#statAttByte) and stored.

Result of network 2 - "CREATE_DB" instruction

If the normally open contact ("startCreate") supplies the signal state "TRUE", the "CREATE_DB" instruction is executed. According to the input parameter COUNT ("setLenByte"), a new DB with a length of 8 bytes is created in the process. The number of the DB that is to be created is specified with the input parameter LOW_LIMIT ("lowLim") or UP_LIMIT ("hiLim"). The properties (in BYTE) with which the DB is to be created are stored at the input parameter ATTR ("#statAttByte"). The DB is to be created only in the load memory and filled with defined contents. A DB is stored at the input parameter SRCBLK ("SLI_srcDB_CREATE_DB.myData") to serve as template for the contents of the new DB.

The output parameter RET_VAL ("CreateRetVal") indicates that processing took place without errors. The instruction "CREATE_DB" remains inactive after its execution ("createRetVal" has the value 16#7000"). The number of the new DB is briefly output at the end of the execution at the output parameter DB_NUM ("#nbrNewDB").

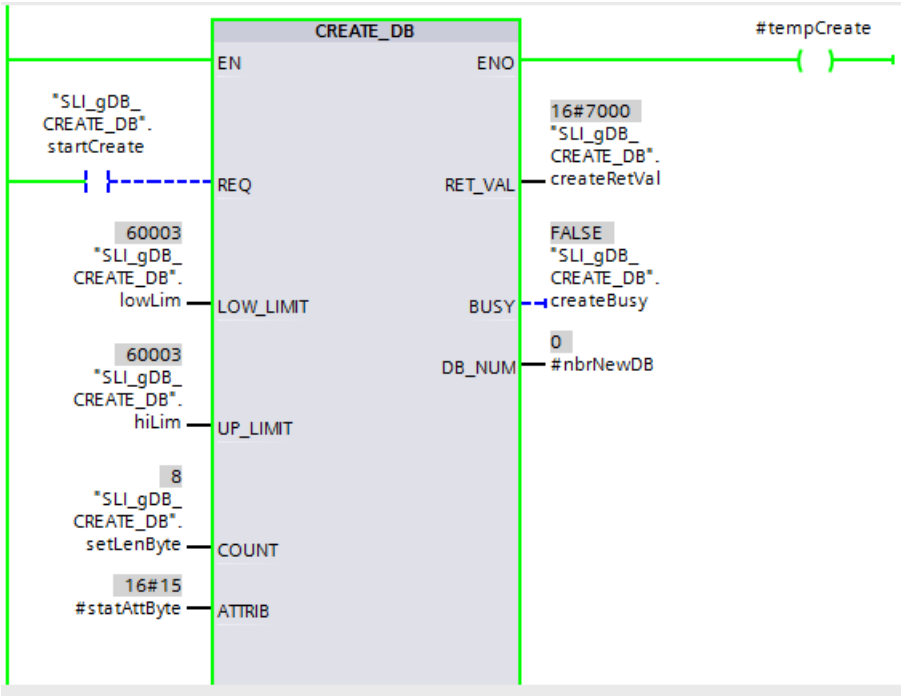

Result of network 3

When processing of CREATE_DB took place without errors, "createRetVal" has the value "16#0000". As a result, the number ("#nbrNewDB") of the DB is stored in the "memNbrNewDB" tag.

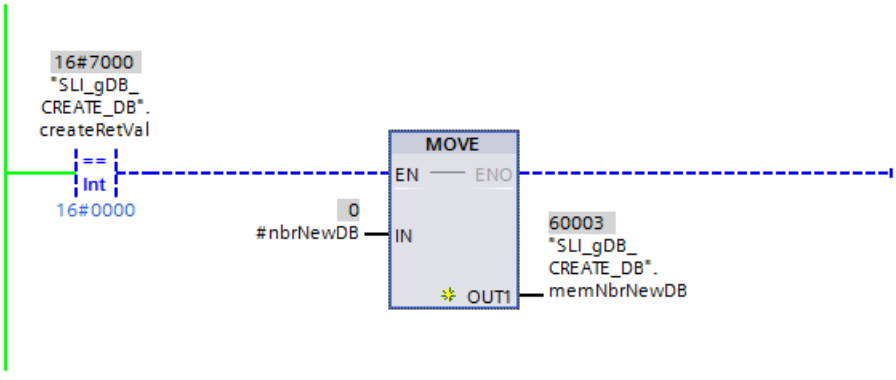

Result of network 4

When CREATE_DB at the ENO parameter ("#tempCreate") supplies the signal state "TRUE", the interconnection of network 4 is active. In addition, when processing of CREATE_DB took place without errors, the reset ("startCreate") is set to "TRUE" and the execution of CREATE_DB is ended.

SLI_gDB_CREATE_DB

	Name	Data type	Start value	Monitor value
1	Static			
2	startCreate	Bool	false	FALSE
3	lowLim	UInt	60003	60003
4	hiLim	UInt	60003	60003
5	setLenByte	UDInt	8	8
6	setAttrBool	Array[0..7] of Bool		
7	createRetVal	Int	0	28672
8	createBusy	Bool	false	FALSE
9	memNbrNewDB	UInt	0	60003
10	startRead	Bool	false	FALSE
11	rdblRetVal	Int	0	28672
12	rdblBusy	Bool	false	FALSE
13	startDelete	Bool	false	FALSE
14	delRetVal	Int	0	28672
15	delBusy	Bool	false	FALSE

Result of network 5 - "READ_DBL" instruction

The following preconditions exist for executing READ_DBL:

- CREATE_DB is inactive ("createRetVal" has the value "16#7000").
- CREATE_DB has created a new DB whose number was stored ("memNbrNewDB" has a value > 0).

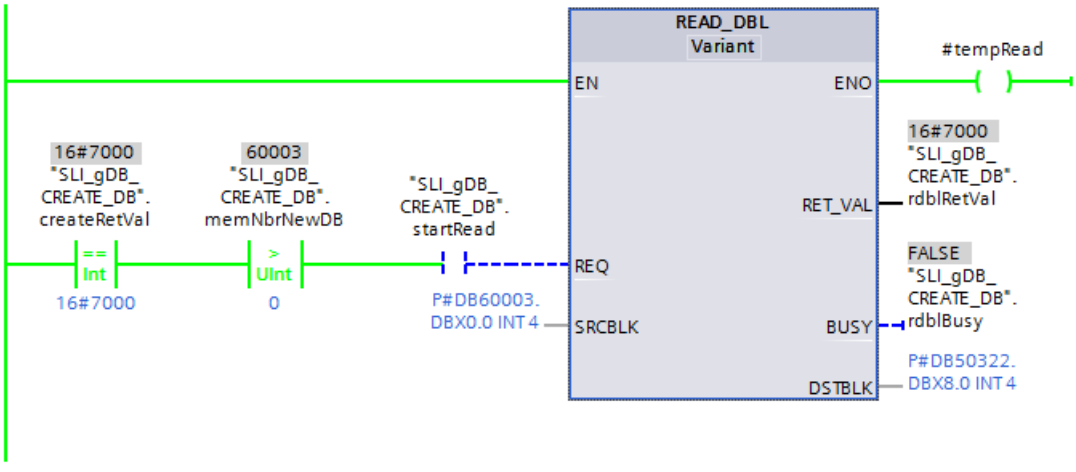

If the normally open contact ("startRead") supplies the signal state "TRUE", the "READ_DBL" instruction is executed. The newly created DB is referenced in the load memory with the input parameter SCRBLK. The DB is referenced in the work memory ("SLI_srcDB_CREATE_DB.readData") with the output parameter DSTBLK. The "READ_DBL"

instruction calls the data to be transferred in the load memory (SCRBLK) and reads the data at the required location in the work memory (DSTBLK).

The output parameter RET_VAL ("rdblRetVal") indicates that processing took place without errors. The transferred data is displayed for the DB in the work memory ("SLI_srcDB_CREATE_DB.readData").

SLI_srcDB_CREATE_DB

	Name	Data type	Offset	Start value	Monitor value
1	Static				
2	myData	Array[0..3] of Int	0.0		
3	myData[0]	Int	0.0	1	1
4	myData[1]	Int	2.0	2	2
5	myData[2]	Int	4.0	3	3
6	myData[3]	Int	6.0	4	4
7	readData	Array[0..3] of Int	8.0		
8	readData[0]	Int	0.0	0	1
9	readData[1]	Int	2.0	0	2
10	readData[2]	Int	4.0	0	3
11	readData[3]	Int	6.0	0	4

Result of network 6

When READ_DBL at the ENO parameter ("#tempRead") supplies the signal state "TRUE", the interconnection of network 6 is active. In addition, when processing of READ_DBL is complete, the reset ("startRead") is set to "TRUE" and the execution of READ_DBL is ended.

Result of network 7 - "DELETE_DB" instruction

The following preconditions exist for executing DELETE_DB:

- CREATE_DB is inactive ("createRetVal" has the value "16#7000").
- CREATE_DB has created a new DB whose number was stored ("memNbrNewDB" has a value > 0).

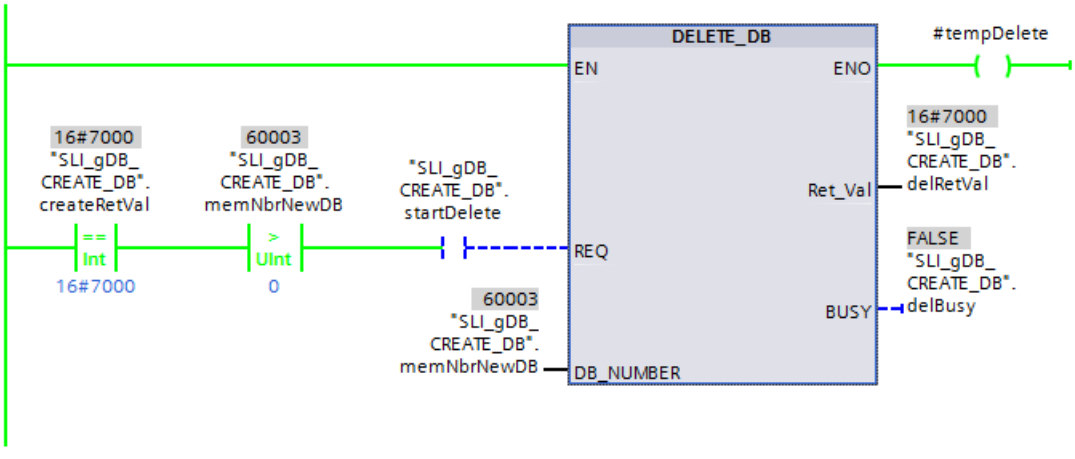

If the normally open contact ("startDelete") supplies the signal state "TRUE", the "DELETE_DB" instruction is executed. The newly created DB is referenced based on its

number with the input parameter DB_NUMBER ("memNbrNewDB"). The "DELETE_DB" instruction deletes the newly created DB.

The output parameter RET_VAL ("deleteRetVal") indicates that processing took place without errors.

Result of network 8

When DELETE_DB at the ENO parameter ("#tempRead") supplies the signal state "TRUE", the interconnection of network 8 is active. In addition, when processing of DELETE_DB is complete, the reset ("startDelete") is set to "TRUE" and the execution of DELETE_DB is ended.

The previously created block is deleted after the execution of DELETE_DB The "memNbrNewDB" tag is set to "0" in order to prevent a command being issued to delete an already deleted block.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

4.1.4.13 Addressing

Instructions for address conversion

Description

There are different options for addressing a module (IO address, hardware identifier, slot).

You can convert the address data with the following instructions:

- GEO2LOG: Determine hardware identifier from slot (Page 2739)
- LOG2GEO: Determine slot from hardware identifier (Page 2742)
- LOG2MOD: Determine the hardware identifier from addressing of STEP 7 V5.5 SPx (Page 2744)
- IO2MOD: Determine hardware identifier from an IO address (Page 2745)
- RD_ADDR: Determine IO addresses from the hardware identifier (Page 2748)

The following instructions are supported in addition for migrated projects:

- GEO_LOG: Determine hardware identifier from slot (Page 2751)
- LOG_GEO: Determine slot from hardware identifier (Page 2753)
- RD_LGADR: Determine IO addresses from the hardware identifier (Page 2754)
- GADR_LGC: Determine hardware identifier from slot and offset in the user data address area (Page 2756)
- LGC_GADR: Determine slot from hardware identifier (Page 2757)

Type of address conversion

The figure below shows which instruction runs which address conversion.

Name	Type	IO address(es)	Hardware identifier	Slot
GEO2LOG	SFC			
LOG2GEO	SFC			
LOG2MOD	SFC			
IO2MOD	SFC			
RD_ADDR	SFC			
GEO_LOG	FC			
LOG_GEO	FC			
RD_LGADR	FC			
GADR_LGC	FC			
LGC_GADR	FC			

GEO2LOG: Determine hardware identifier from slot (S7-1200, S7-1500)

Description

You use the "GEO2LOG" instruction to determine hardware identifier based on slot information that you define using the system data type GEOADDR.

Depending on the type of hardware you define at the parameter HWTYPE the following information is evaluated from the other parameters GEOADDR:

- When HWTYPE = 1 (IO system):
  - Only IOSYSTEM is evaluated. The other parameters of GEOADDR are not taken into consideration.
  - The hardware identifier of the IO system is output.
- When HWTYPE = 2 (IO device):
  - IOSYSTEM and STATION are evaluated. The other parameters of GEOADDR are not taken into consideration.
  - The hardware identifier of the IO device is output.

- With HWTYPE = 4 (module):
  - IOSYSTEM, STATION and SLOT are evaluated. The SUBSLOT parameter of GEOADDR is not taken into consideration.
  - The hardware identifier of the module is output.
- With HWTYPE = 5 (submodule):
  - All parameters of GEOADDR are evaluated.
  - The hardware identifier of the submodule is output.

The AREA parameter of the GEOADDR system data type is not evaluated.

Parameters

The following table shows the parameters of the "GEO2LOG" instruction:

Parameter	Declaration	Data type	Memory area	Description
GEOADDR	Input	VARIANT	D, L	Pointer to the structure of the GEOADDR system data type. The system data type contains the slot information from which the hardware ID is determined. See also: System data type GEOADDR (Page 2750)
RET_VAL	Return	INT	I, Q, M, D, L	Output of error information.
LADDR	Output	HW_ANY	I, Q, M, D, L	Hardware identifier of the assembly or the module. The number is automatically assigned and is stored in the properties in the hardware configuration.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8091	Invalid value for in GEOADDR for HWTYPE.
8094	Invalid value for in GEOADDR for IOSYSTEM.
8095	Invalid value for in GEOADDR for STATION.
8096	Invalid value for in GEOADDR for SLOT.
8097	Invalid value for in GEOADDR for SUBSLOT.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you determine the hardware ID of the CPU based on the slot information.

Create four tags in a global data block for storing the data.

SLI_gDB_GEO2LOG

	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	geoAddr	GEOADDR	
4	HWTYPE	UInt	4
5	AREA	UInt	0
6	IOSYSTEM	UInt	0
7	STATION	UInt	0
8	SLOT	UInt	1
9	SUBSLOT	UInt	0
10	returnValue	Int	0
11	laddr	HW_ANY	16#0

Interconnect the parameters of the instruction as follows.

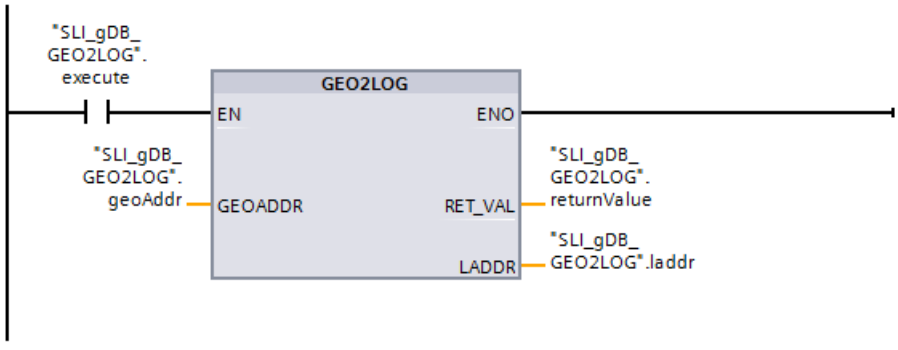

If the normally open contact ("execute") supplies the signal state "TRUE", the "GEO2LOG" instruction is executed. The slot information is stored in the input parameter GEOADDR ("geoAddr"). The instruction "GEO2LOG" calls the hardware ID for a module (HWTYPE with the value "4") of the central controller (IOSYSTEM with the value "0") in the rack "0" (value of STATION), in slot "1" (value of SLOT).

The determined hardware ID of the local component of the CPU is displayed at the output parameter LADDR ("laddr"). The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

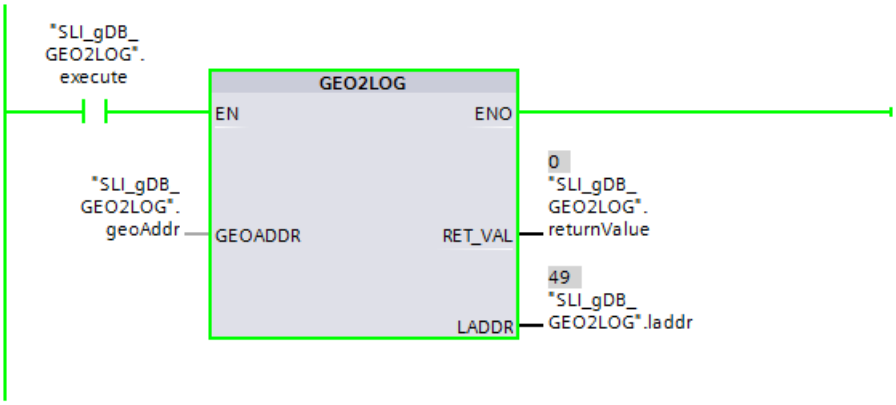

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Instructions for address conversion (Page 2738)

LOG2GEO: Determine slot from hardware identifier (S7-1200, S7-1500)

Description

You use the "LOG2GEO" instruction to determine the module slot belonging to a hardware identifier.

Parameters

The following table shows the parameters of the "LOG2GEO" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_ANY	I, Q, M, D, L or constant	Hardware identifier of the module whose slot you want to find. The hardware ID is assigned automatically and is stored in the properties of the module in the hardware configuration and the system constants.
RET_VAL	Output	INT	I, Q, M, D, L	Output of error information.
GEOADDR	InOut	VARIANT	D, L	Pointer to the GEOADDR system data type. The slot information is written in the system data type GEOADDR. See also: System data type GEOADDR (Page 2750)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8090	The address specified at the LADDR parameter is invalid.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you determine the slot information of an input module based on the hardware ID.

Create four tags in a global data block for storing the data.

SLI_gDB_LOG2GEO

	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	laddr	HW_ANY	257
4	geoAddr	GEOADDR	
5	HWTYPE	UInt	0
6	AREA	UInt	0
7	IOSYSTEM	UInt	0
8	STATION	UInt	0
9	SLOT	UInt	0
10	SUBSLOT	UInt	0
11	returnValue	Int	0

Interconnect the parameters of the instruction as follows.

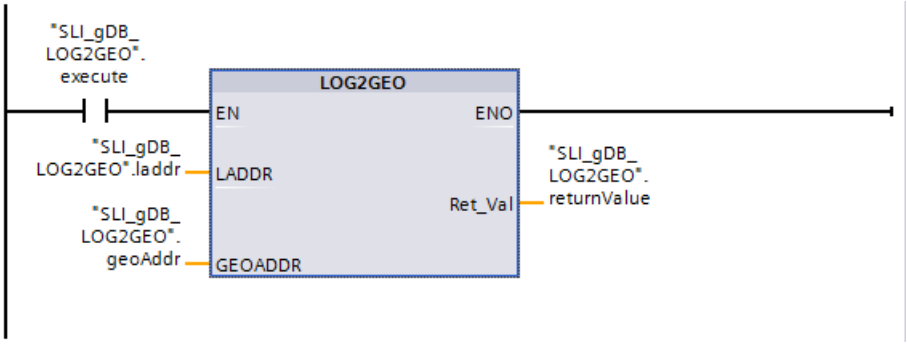

If the normally open contact ("execute") supplies the signal state "TRUE", the "LOG2GEO" instruction is executed. The hardware ID of the input module is stored with the input parameter LADDR ("laddr"). The instruction "LOG2GEO" calls the slot information of the input module using the hardware ID.

The determined slot information of the input module is displayed at the output parameter GEOADDR ("geoAddr"). A module (HWTYPE with the value "4") of the central controller (IOSYSTEM with the value "0") in the rack "0" (value of STATION), in slot "2" (value of SLOT) is determined.

The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

SLI_gDB_LOG2GEO

	Name	Data type	Start value	Monitor value
1	Static			
2	execute	Bool	false	TRUE
3	laddr	HW_ANY	257	16#0101
4	geoAddr	GEOADDR		
5	HWTYPE	UInt	0	4
6	AREA	UInt	0	0
7	IOSYSTEM	UInt	0	0
8	STATION	UInt	0	0
9	SLOT	UInt	0	2
10	SUBSLOT	UInt	0	0
11	returnValue	Int	0	0

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

LOG2MOD: Determine the hardware identifier from addressing of STEP 7 V5.5 SPx (S7-1500)

Description

You use the "LOG2MOD" instruction to determine the hardware identifier for an IO (sub)module from the addressing of STEP 7 5.5 SPx (IO data address or diagnostic address).

The hardware identifier is used at the LADDR input parameter for addressing of various instructions. Convert the addressing parameters from STEP 7 5.5 SPx by calling "LOG2MOD" beforehand.

Parameter

The following table shows the parameters of the "LOG2MOD" instruction:

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Identifier of the address area as in STEP 7 5.5 SPx: • B#16#00: Bit15 of ADDR specifies whether an input (Bit15=0) or output address (Bit15=1) exists. • B#16#54= Peripheral input (PI) • B#16#55= Peripheral output (PQ)
ADDR	Input	WORD	I, Q, M, D, L or constant	Logical address of the IO data of the module as offset (corresponding addressing in STEP 7 5.5 SPx) or diagnostic address.

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Error code of the instruction.
HWID	Output	HW_IO	I, Q, M, D, L	Determined hardware identifier of the IO (sub)module.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8093	• Specified address is not used by any hardware components. • Specified value at IOID parameter is invalid.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

IO2MOD: Determine hardware identifier from an IO address (S7-1200, S7-1500)

Description

The "IO2MOD" instruction determines the hardware identifier of the module from an IO address (I, Q, PI, PQ) of a module.

Enter the IO address at the ADDR parameter.

- If a series of IO addresses is used at this parameter, only the first address is evaluated to determine the hardware identifier.

- If the first address is correctly specified, the length for the address specification at the ADDR parameter is of no significance.

- If an address area is used that encompasses several modules or unused addresses, the hardware identifier of the first module can also be determined.

- If no IO address of a module is specified at parameter ADDR, the error code 8090 is output at parameter RET_VAL.

Note

Input of IO address in SCL

In SCL, you do not program using the I/O access ID "%QWx:P". In this case, use the symbolic tag name or the absolute address in the process image.

Parameters

The following table shows the parameters of the "IO2MOD" instruction:

Parameter	Declaration	Data type	Memory area	Description
ADDR	Input	VARIANT	I, Q	IO address (I, Q, PI, PQ) within a module. Make sure that slice access is not used for the parameter ADDR. If this is the case, incorrect values are output at the LADDR parameter.
RET_VAL	Return	INT	I, Q, M, D, L	Error code of the instruction.
LADDR	Output	HW_IO	I, Q, M, D, L	Determined hardware identifier (logical address) of the IO module.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8090	IO address specified at ADDR parameter is not used by any hardware component.
8092	Invalid data type used at parameter ADDR (for example, WCHAR or WSTRING).
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you determine the hardware ID of an input module based on an IO address.

Create a PLC tag in a tag table to read the hardware ID.

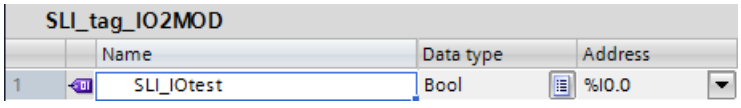

Create three tags in a global data block for storing the data.

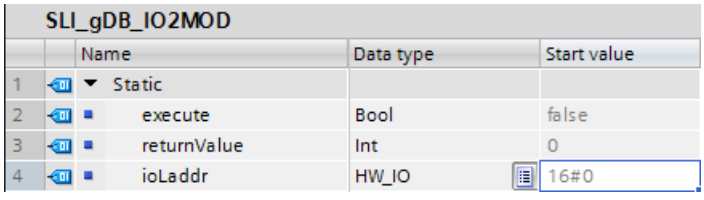

Interconnect the parameters of the instruction as follows.

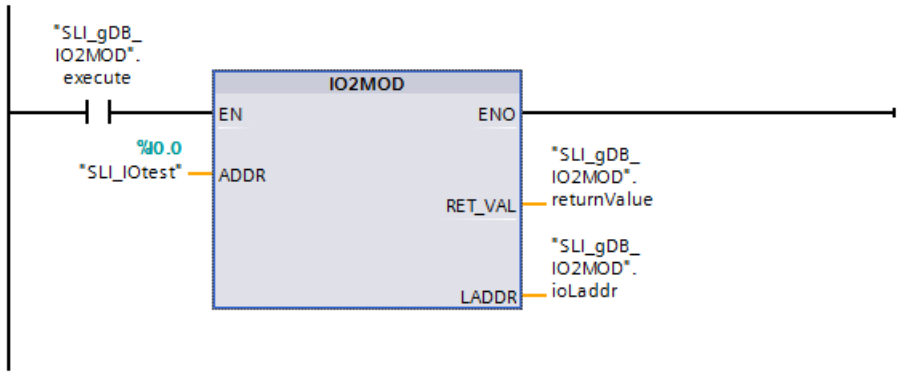

If the normally open contact ("execute") supplies the signal state "TRUE", the "IO2MOD" instruction is executed. The IO address to be localized is transmitted with the input parameter ADDR ("SLI_IOtest"). The "IO2MOD" instruction calls the IO address and reads the input module.

The determined hardware ID of the input module is displayed at the output parameter LADDR ("ioLaddr"). The output parameter RET_VAL ("returnValue") indicates that processing took place without errors.

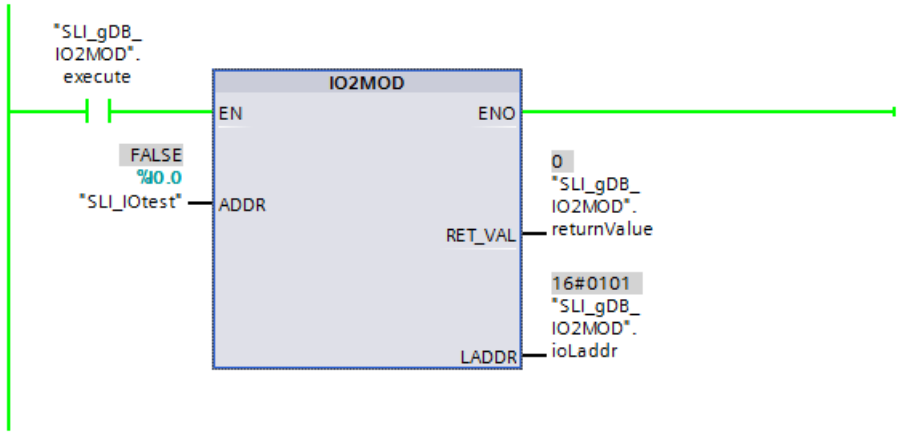

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

RD_ADDR: Determine IO addresses from the hardware identifier (S7-1200, S7-1500)

Description

The "RD_ADDR" instruction determines the length and the start address of the inputs or outputs based on the hardware identifier of a sub(module).

- Use the LADDR parameter to select the input or output module based on the hardware identifier.
- The following output parameters are used depending on whether it is an input module or output module:
  - In the case of an input module the determined values are output at the PIADDR and PICOUNT parameters.
  - In the case of an output module the determined values are output at the PQADDR and PQCOUNT parameters.
- The PIADDR and PQADDR parameters each contain the start address of the I/O addresses of the module.
- The PICOUNT and PQCOUNT parameters each contain the number of bytes of the inputs our outputs (1 byte for 8 inputs/outputs, 2 bytes for 16 inputs/outputs).

Note

Determining the IO addresses with packed addresses of an ET200

With packed addresses of an ET200. the first module of the pack group returns all addresses.

For the other modules, the following is output with the PIADDR and PQADDR parameters:

- with PROFINET as address "0"
- with PROFIBUS as address "0" In addition, the error code W#16#8090 is output.

For the number of bytes of the inputs and outputs (parameters PICOUNT and PQCOUNT), "0" is output in each case.

Parameter

The following table shows the parameters of the "RD_ADDR" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the (sub)module.
RET_VAL	Return	INT	I, Q, M, D, L	Error code of the instruction.
PIADDR	Output	UDINT	I, Q, M, D, L	Start address of the input module.
PICOUNT	Output	UINT	I, Q, M, D, L	Number of bytes of the inputs.
PQADDR	Output	UDINT	I, Q, M, D, L	Start address of the output module.
PQCOUNT	Output	UINT	I, Q, M, D, L	Number of bytes of the outputs.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8090	Hardware identifier of the module at the LADDR parameter is invalid.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

Example

In the following example, you determine the length and start address of the inputs in an input module based on the hardware ID.

Create seven tags in a global data block for storing the data.

SLI_gDB_RD_ADDR

	Name	Data type	Start value
1	Static		
2	executeRead	Bool	false
3	readLaddr	HW_IO	257
4	readRetVal	Int	0
5	piAddr	UDInt	0
6	piCount	UInt	0
7	pqAddr	UDInt	0
8	pqCount	UInt	0

Interconnect the parameters of the instruction as follows.

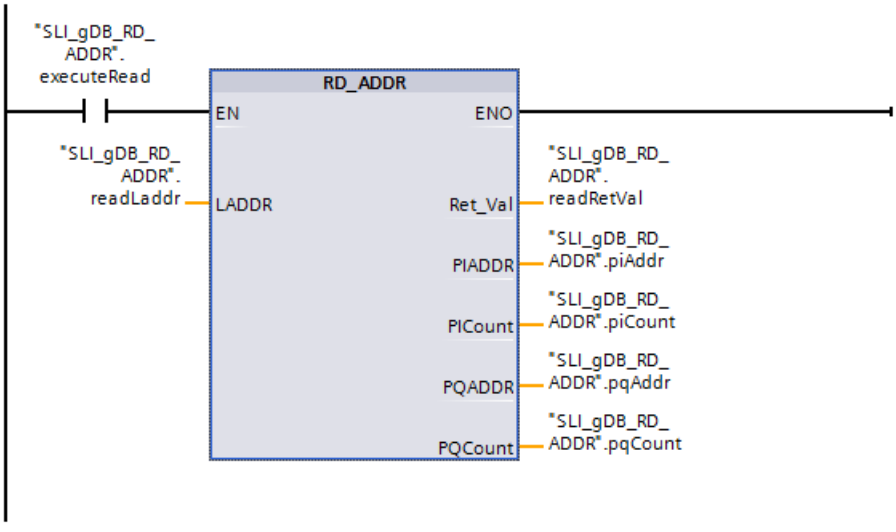

If the normally open contact ("executeRead") supplies the signal state "TRUE", the "RD_ADDR" instruction is executed. The hardware ID of the input module is stored with the input parameter LADDR ("readLaddr"). The instruction "RD_ADDR" determines the input module using the hardware ID and reads the length of the inputs and start address.

The determined start address is displayed at the output parameter PIADDR ("piAddr"). The number of bytes of the inputs is displayed at the output parameter PICOUNT ("piCount").

The output parameter RET_VAL ("readRetVal") indicates that processing took place without errors.

SLI_gDB_RD_ADDR

	Name	Data type	Start value	Monitor value
1	Static			
2	executeRead	Bool	false	TRUE
3	readLaddr	HW_IO	257	16#0101
4	readRetVal	Int	0	0
5	piAddr	UDInt	0	0
6	piCount	UInt	0	2
7	pqAddr	UDInt	0	0
8	pqCount	UInt	0	0

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

System data type GEOADDR

Geographical address

The system data type GEOADDR contains the geographical address of a module, i.e. the slot information.

- Geographical address for PROFINET IO
  For PROFINET IO the geographical address was composed of the ID of the PROFINET IO system, the device number, the slot number and the submodule (if a sub-module is used).
- Geographical address for PROFIBUS DP
  For PROFIBUS DP, the geographical address consists of the ID of the DP master system, the station number and the slot number.

The slot information of the modules can be found in the hardware configuration of each module.

System data type GEOADDR

If you enter "GEOADDR" as the data type in a data block, the structure GEOADDR is automatically created.

The system data type GEOADDR has the following structure:

Parameter name		Data type	Description
GEOADDR		STRUCT	
	HWTYPE	UINT	Hardware type: • 1: IO system (PROFINET/PROFIBUS) • 2: IO device/DP slave • 3: Rack • 4: Module • 5: Submodule If a hardware type is not supported by the instruction, a HWTYPE "0" is output.
	AREA	UINT	Area ID: • 0 = CPU • 1 = PROFINET IO • 2 = PROFIBUS DP • 3 = AS-i
	IOSYSTEM	UINT	PROFINET IO system (0=central unit in the rack)
	STATION	UINT	• Number of the rack if area identifier AREA = 0 (central module). • Station number if area identifier AREA > 0.
	SLOT	UINT	Slot number
	SUBSLOT	UINT	Number of the submodule. This parameter has the value "0" if no submodule is available or can be plugged.

Legacy

GEO_LOG: Determine hardware identifier from slot (S7-1500)

Description

The corresponding module slot of a signal module is known. Use the "GEO_LOG" instruction to determine the corresponding hardware identifier of the module from this.

Parameter

The following table shows the parameters of the "GEO_LOG" instruction:

Parameter	Declaration	Data type	Memory area	Description
MASTER	Input	INT	I, Q, M, D, L or constant	Area ID: • 0, if the slot is located in a centralized configuration. • 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	Input	INT	I, Q, M, D, L or constant	• If MASTER = 0: Number of the rack • If MASTER > 0: Station number of the field device
SLOT	Input	INT	I, Q, M, D, L or constant	Slot number
SUBSLOT	Input	INT	I, Q, M, D, L or constant	Number of the submodule. If no submodule exists or can be inserted SUBSLOT must have the value "0".
RET_VAL	Return	INT	I, Q, M, D, L	Error information
LADDR	Output	HW_IO	I, Q, M, D, L	Hardware identifier of the module

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8094	No subnet was configured with the specified SUBNETID .
8095	Illegal value for STATION parameter
8096	Illegal value for SLOT parameter
8097	Illegal value for SUBSLOT parameter.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

Instructions for address conversion (Page 2738)

LOG_GEO: Determine slot from hardware identifier (S7-1500)

Description

You use the "LOG_GEO" instruction to determine the module slot belonging to a hardware identifier.

Parameters

The following table shows the parameters of the "LOG_GEO" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module for which the slot is to be determined.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
AREA	Output	INT	I, Q, M, D, L	The area ID indicates how the remaining output parameters are to be interpreted: • 0: central device • 2: PROFIBUS DP / PROFINET IO
MASTER	Output	INT	I, Q, M, D, L	With AREA = 0: • 0: If the slot is located in one of the racks (central device). With AREA = 2: • 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	Output	INT	I, Q, M, D, L	• With MASTER = 0: Number of the rack • With MASTER > 0: Station number of the field device
SLOT	Output	INT	I, Q, M, D, L	Slot number
SUBSLOT	Output	INT	I, Q, M, D, L	Number of the submodule
OFFSET	Output	INT	I, Q, M, D, L	The OFFSET parameter is not output by the instruction (always "0").

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

Instructions for address conversion (Page 2738)

RD_LGADR: Determine IO addresses from the hardware identifier

Description

Based on the hardware identifier, you can determine the IO addresses of a module, a central submodule or a submodule for PNIO and DP slaves with the "RD_LGADR" instruction.

- You can specify the hardware ID of the submodule at the LADDR parameter.
- The addresses are written to the PEADDR and PAADDR parameters is ascending order.
  - For an input module, only the PEADDR parameter is written. For an output module, the PAADDR parameter is written.
  - An Array of WORD is used to store the addresses in each case.
- The number of addresses is output at the PECOUNT parameter (for an input module) and PACOUNT parameter (for an output module).

Note

Description of the boxes PEADDR and PAADDR

In PEADDR or PAADDR only as many addresses are entered as are returned (This number is entered in PECOUNT or PACOUNT). Addresses of earlier RD_LGADR-calls located behind the currently returned addresses are retained; they must not be evaluated in the current call.

Note

Determining the IO addresses with packed addresses of an ET200

With packed addresses of an ET200. the first module of the pack group returns all addresses.

For the other modules, the following is output with the PEADDR and PAADDR parameters:

- with PROFINET as address "0"
- with PROFIBUS as address "0" In addition, the error code W#16#8090 is output.

For the number of bytes of the inputs and outputs (parameters PECOUNT and PACOUNT), "0" is output in each case.

Parameters

The following table shows the parameters of the "RD_LGADR" instruction:

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Is not evaluated
LADDR	Input	HW_ANY	I, Q, M, D, L or constant	Hardware identifier of the module or the sub-module.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PEADDR	Output	ANY	I, Q, M, D, L	Field for the PI addresses with the Array of WORD data type Note: The field has to lie in the non-optimized memory area.
PECOUNT	Output	INT	I, Q, M, D, L	Number of returned PI addresses
PAADDR	Output	ANY	I, Q, M, D, L	Field for the PQ addresses with the Array of WORD data type Note: The field has to lie in the non-optimized memory area.
PACOUNT	Output	INT	I, Q, M, D, L	Number of returned PQ addresses

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Specified hardware identifier is invalid
80A0	Error in the output parameter PEADDR: The data type of the array elements is not WORD.
80A1	Error in the output parameter PAADDR: The data type of the array elements is not WORD.
80A2	Error in the output parameter PEADDR: The specified array does not accommodate all the logical addresses.

Error code (W#16#...)	Explanation
80A3	Error in the output parameter PAADDR: The specified array does not accommodate all the logical addresses.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)

See also

Switching display formats in the program status (Page 7698)

Instructions for address conversion (Page 2738)

GADR_LGC: Determine hardware identifier from slot and offset in the user data address area (S7-1500)

Description

You use the "GADR_LGC" instruction to determine the hardware identifier of a signal module. The hardware identifier is determined from the module slot and the offset in the user data address area of the module.

Note

Restriction on use

The "GADR_LGC" instruction cannot be used for modules behind gateways (e.g., IE/PB link). Use the "GEO2LOG" instruction instead.

Parameters

The following table shows the parameters of the "GADR_LGC" instruction:

Parameter	Declaration	Data type	Memory area	Description
SUBNETID	Input	BYTE	I, Q, M, D, L or constant	Area ID: • 0: If the slot is located in the central module • 1 to 32: DP master system ID of the corresponding distributed I/O system if the slot is in a distributed I/O device. • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
RACK	Input	WORD	I, Q, M, D, L or constant	• Number of the rack, if area identifier is 0 • Device number of the distributed I/O device if the area identifier > 0
SLOT	Input	WORD	I, Q, M, D, L or constant	Slot no.

Parameter	Declaration	Data type	Memory area	Description
SUBSLOT	Input	BYTE	I, Q, M, D, L or constant	Irrelevant
SUBADDR	Input	WORD	I, Q, M, D, L or constant	Offset in the user data address area of the module
RET_VAL	Return	INT	I, Q, M, D, L	Error information
IOID	Output	BYTE	I, Q, M, D, L	The IOID output parameter is not written (always "0").
LADDR	Output	HW_MOD-ULE	I, Q, M, D, L	Hardware identifier of the module

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8094	No subnet was configured with the specified SUBNETID .
8095	Illegal value for RACK parameter.
8096	Illegal value for SLOT parameter.
8098	Illegal value for SUBADDR parameter.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

Instructions for address conversion (Page 2738)

LGC_GADR: Determine slot from hardware identifier (S7-1500)

Description

You use the "LGC_GADR" instruction to determine the module slot belonging to a hardware identifier.

Note

Restriction on use

The "LGC_GADR" instruction cannot be used for modules behind gateways (e.g., IE/PB link). Use the "LOG2GEO" instruction instead.

Parameters

The following table shows the parameters of the "LGC_GADR" instruction:

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Is not evaluated.
LADDR	Input	HW_MOD-ULE	I, Q, M, D, L or constant	Hardware identifier of the module
RET_VAL	Return	INT	I, Q, M, D, L	Error information
AREA	Output	BYTE	I, Q, M, D, L	The area ID indicates how the remaining output parameters are to be interpreted: • 0: Central module • 2: PROFIBUS DP
RACK	Output	WORD	I, Q, M, D, L	Rack number: • With central module (AREA= 0): – Rack number • For PROFIBUS DP (AREA=2): – Low byte: Station number – High byte: DP master system ID
SLOT	Output	WORD	I, Q, M, D, L	Slot number: • With central module (AREA= 0): – Slot number • For PROFIBUS DP (AREA=2): – Slot no. in the station
SUBADDR	Output	WORD	I, Q, M, D, L	Is not output (always "0").

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid
8093	This instruction is not permitted for the module selected by the parameters IOID and LADDR .
General error information	See also: GET_ERR_ID: Get error ID locally (Page 883)
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

Instructions for address conversion (Page 2738)

4.1.4.14 File handling

FileReadC: Read file from Memory Card (S7-1500)

Description

The "FileReadC" instruction is used to read data from a file located on the memory card and write it to a destination area on the CPU. You specify the file by entering its name and complete path.

Enable reading from the file using the REQ control parameter. The output parameters Done, Busy, Error and Status indicate the status of the job.

The following rules apply for the file name and path specification:

- The file name must not be longer than 55 characters.
- The following characters are permitted for the folder and file name: "0" to "9", "a" to "z", "A" to "Z", "-", "_"
- The file name must not contain more than one dot ("."); the dot separates the name from the file extension. The file name must contain at least one character. A file extension is not necessary.
- The path name must not start with "/", "\" or ".".
- The path name must not contain any ".."

Examples of valid paths and file names: "UserFiles/Lift16_DataBase.txt", "UserFiles/2017-04-13_ErrorLog.bin"

You can read up to 16 MB (16,777,216 bytes) by executing "FileReadC"; the data is read segment by segment. The segment size is device-specific; an S7-1500-CPU, for example, uses 32 KB (32,768 byte) blocks. If the number of data items to be read is greater than the segment size, you will need to call the instruction multiple times in your program. For consistency reasons, you should therefore not access the data read until after you call the instruction for the last time.

The parameters "Offset" and "Length" specify the length of the items of data to be read. The resource occupied is released again once the read process is complete.

Parameters

The following table shows the parameters of the instruction "FileReadC":

Parameters	Declaration	Data type	Description
REQ	Input	BOOL	Control parameter Request Enables reading a file from the memory card with a rising edge.
Name	Input	STRING	Name of the file to be read incl. complete path
Offset	Input	UDINT	Byte offset after which the file is to be read
Length	Input	UDINT	Length of the area to be read in bytes Length = 0 means that the maximum possible number of data items per call is read (for a S7-1500-CPU, 32 KB or the size of the "Data" parameter)

Parameters	Declaration	Data type	Description
Done	Output	BOOL	1: The instruction has been executed successfully. The information read has been transferred to the "Data" parameter.
Busy	Output	BOOL	Status parameter • 0: Execution of the instruction completed or not yet started. • 1: Execution of the instruction not yet complete.
Error	Output	BOOL	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the "Status" parameter.
Status	Output	WORD	Error code
ResultLength	Output	UDINT	Length of data read in bytes
Data	InOut	VARIANT	Destination area for data read Permitted data types: BYTE and Array of BYTE

You can find additional information on valid data types under Overview of the valid data types (Page 231).

Parameter Status

Error code* (W#16#...)	Explanation
0	No error
0081	Warning: Offset + Length is greater than the length of the file. Data is read from "Offset" to the end of the file. "Data" contains fewer data items than requested; "ResultLength" is shorter than "Length"; data past "ResultLength" in "Data" remains unchanged.
8091	Path does not exist or is is invalid.
8092	The "Name" parameter is not of the data type "STRING“, is too long or contains invalid characters.
8093	The "Offset" parameter points past the end of the file to be read.
8094	The "Length" parameter is larger than the maximum permitted value. For an S7-1500 CPU, for example, the maximum value permitted is 16 MB, i.e. 16,777,216 bytes.
80A1	Read error; the destination area specified by the "Data" parameter may be partly overwritten.
80B1	The destination area specified by the "Data" parameter is shorter than the length required at the "Length" parameter.
80C0	The file cannot be accessed (write-protected or blocked by another process).
80C3	The maximum number of simultaneously active FileReadC instructions has already been reached.
8A24	The “Data“ points to an impermissible area, for example to the load memory or the local data.
8A30	The target range is write-protected, for example a write-protected DB.
8A51	Invalid data type of the "Data" parameter.
8A52	The tag at the Data parameter is insufficient. A part of the source area data may have been written.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, please refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

FileWriteC: Write file on the memory card (S7-1500)

Description

The "FileWriteC" instruction is used to write data located in a source area on the CPU to a file located in the "UserFiles" folder on the memory card. You specify this file by entering its name and complete path. If the file does not exist, it is created by the CPU in the "UserFiles" folder. The "Offset" parameter must be padded with zeros in this case, otherwise W#16#8093 is output at the "Status" parameter. If the "UserFiles" folder does not exist either, it is also created by the CPU; subfolders are not created – in this case, W#16#8091 is output at the "Status" parameter.

Enable writing to the file using the "REQ" control parameter. The output parameters "Done", "Busy", "Error" and "Status" indicate the status of the job.

The following rules apply for the file name and path specification:

- The file name must not be longer than 55 characters.
- The following characters are permitted for the folder and file name: "0" to "9", "a" to "z", "A" to "Z", "-", "_"
- The file name must not contain more than one dot ("."); the dot separates the name from the file extension. The file name must contain at least one character. A file extension is not necessary.
- The path name must not start with "/", "\" or ".".
- The path name must not contain any ".."

Examples of valid paths and file names: "UserFiles/Lift16_DataBase.txt", "UserFiles/2017-04-13_ErrorLog.bin"

You can write up to 16 MB (16,777,216 bytes) by executing "FileWriteC"; the data is written segment by segment. The segment size is device-specific; an S7-1500-CPU, for example, uses 32 KB (32,768 byte) blocks. If the number of data items to be written is greater than the segment size, you will need to call the instruction multiple times in your program. For consistency reasons, you should therefore not access the data written until after you call the instruction for the last time. If the available file is too short, it is extended to the required size.

The parameters "Offset" and "Length" specify the location in the file to which the data is to be written. The resource occupied is released again once the write process is complete.

The "FileWriteC" instruction only starts the write operation if the following condition is fulfilled: "Offset" + "Length" <= 16 MB

Parameters

The following table shows the parameters of the instruction "FileWriteC":

Parameters	Declaration	Data type	Description
REQ	Input	BOOL	Control parameter Request Enables writing a file on the memory card with a rising edge.
Name	Input	STRING	Name of the file to be written incl. complete path
Offset	Input	UDINT	Byte offset from which the file is to be written

Parameters	Declaration	Data type	Description
Length	Input	UDINT	Length of the area to be written in bytes "Length" = 0 means that the entire source area specified with the "Data" parameter is written.
Done	Output	BOOL	1: The instruction has been executed successfully.
Busy	Output	BOOL	Status parameter • 0: Execution of the instruction completed or not yet started. • 1: Execution of the instruction not yet complete.
Error	Output	BOOL	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the "Status" parameter.
Status	Output	WORD	Error code
ResultLength	Output	UDINT	Length of data written in bytes
Data	InOut	VARIANT	Source area Permitted data types: BYTE and Array of BYTE

You can find additional information on valid data types under Overview of the valid data types (Page 231).

Parameter Status

Error code* (W#16#...)	Explanation
0	No error
8091	Path does not exist or is is invalid.
8092	The "Name" parameter is not of the data type "STRING“, is too long or contains invalid characters.
8093	• The "Offset" parameter points past the end of the file to be written. • The creation of the file is rejected because "Offset" is greater than zero.
8094	• "Length" is greater than the maximum permitted value. For an S7-1500 CPU, for example, the maximum value permitted is 16 MB, i.e. 16,777,216 bytes. • "Length" + "Offset" is greater than the maximum permitted value.
80A1	Write error; the data in the file on the memory card may be partly overwritten.
80B1	The source area specified by the "Data" parameter is shorter than the length required at the "Length" parameter.
80B3	There is not enough space on the memory card or in the internal load memory.
80B4	The memory card or file is write-protected.
80C0	The file cannot be accessed.
80C3	The maximum number of simultaneously active FileWriteC instructions has already been reached.
8A24	The “Data“ points to an impermissible area, for example to the load memory or the local data.
8A51	Invalid data type of the "Data" parameter.
8A52	The tag at the Data parameter is insufficient. A part of the source area data may have been written.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, please refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

4.1.4.15 R/H system

RH_CTRL: Influencing sequences in R/H systems (S7-1500)

Description

With the instruction "RH_CTRL", you can influence R/H systems as follows:

- You can disable the system state "SYNCUP". The lock is in force until you revoke it with the "RH_CTRL" instruction or until the R/H system goes into "STOP" mode.
- You can re-enable the system state "SYNCUP".

Functional description

The "RH_CTRL" instruction works asynchronously. Processing takes place across several calls. You start the job by calling "RH_CTRL" with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596)

Parameter

The following table shows the parameters of the instruction "RH_CTRL":

Parameter	Declaration	Data type	Description
REQ	Input	BOOL	Control parameter Request Enables the execution of the instruction when REQ = "1".
Mode	Input	BYTE	Using the "Mode" parameter, select how you want to influence the R/H system: • 3: Disable system state SYNCUP • 4: Re-enable system state SYNCUP
Submode	Input	WORD	Hidden parameter. It must have the value 0.
Ret_Val	Output	INT	Error code
Busy	Output	BOOL	Status parameter • 0: Execution of the instruction complete. • 1: Execution of the instruction not yet complete.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Ret_Val parameter

Error code* (W#16#...)	Explanation
0	No error.
1	• with mode = 3: The SYNCUP system state is already disabled. • with mode = 4: The SYNCUP system state is already enabled.
7000	No job in progress (REQ = 0).
7001	REQ = 1 at first call: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value 1.
8082	Disabling the "SYNCUP" system state is no longer possible, because the R/H system is already in this system state.
8090	Invalid value of the "Mode" parameter.
8091	Invalid value of the "Submode" parameter. It must have the value 0.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

RH_GetPrimaryID: Determining the redundancy ID of the primary CPU (S7-1500)

Description

The "RH_GetPrimaryID" instruction is used to determine the redundancy ID of the primary CPU.

You can use this instruction in all system states even if the pairing of the two R or H CPUs has not yet been performed.

Parameter

The following table shows the parameters of the instruction "RH_GetPrimaryID":

Parameter	Declaration	Data type	Description
Ret_Val	Output	INT	Return value

You can find additional information on valid data types under: Overview of the valid data types (Page 231)

Ret_Val parameter

Error code* (W#16#...)	Explanation
1	The CPU with redundancy ID 1 is the primary CPU.
2	The CPU with redundancy ID 2 is the primary CPU.
8091	No data available, for example because "RH_GetPrimaryID" is not supported by your CPU.
* The error codes in the program editor are displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

4.1.5 Technology

4.1.5.1 High-speed counters (S7-1200)

CTRL_HSC: Control high-speed counters (S7-1200)

Parameters

Parameter	Declaration	Data type	Memory area	Description
EN	INPUT	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	OUTPUT	BOOL	I, Q, M, D, L	Enable output
HSC	INPUT	HW_HSC	I, Q, M or constant	Hardware address of the high-speed counter (HW-ID)
DIR	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new count direction (see NEW_DIR)
CV	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new count value (see NEW_CV)
RV	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new reference value (see NEW_RV)
PERIOD	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new period of a frequency measurement (see NEW_PERIOD)
NEW_DIR	INPUT	INT	I, Q, M, D, L or constant	Count direction loaded when DIR = TRUE.

Parameter	Declaration	Data type	Memory area	Description
NEW_CV	INPUT	DINT	I, Q, M, D, L or constant	Count value loaded when CV = TRUE.
NEW_RV	INPUT	DINT	I, Q, M, D, L or constant	Reference value loaded when RV = TRUE.
NEW_PERIOD	INPUT	INT	I, Q, M, D, L or constant	Period of the frequency measurement loaded when PERIOD = TRUE.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Processing status *
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status of the operation

* With a high-speed counter in the CPU or in the signal board, the BUSY parameter always has the value 0.

Description

With the "Control high-speed counter" instruction, you can make parameter settings and control the high-speed counters supported by the CPU by loading new values into the counter. Execution of the instruction requires that a high-speed counter to be controlled is enabled. You cannot execute multiple "Control high-speed counter" instructions simultaneously in the program for a given high-speed counter.

You can load the following parameter values into a high-speed counter using the "Control high-speed counter" instruction:

- Count direction (NEW_DIR): The count direction defines whether a high-speed counter counts up or down. The count direction is defined by the following values at the NEW_DIR input: 1 = up, -1 = down.
  A change to the count direction with the "Control high-speed counter" instruction is only possible when direction control is set in the parameters by the program. The count direction specified at the NEW_DIR input is loaded into a high-speed counter when the bit at the DIR input is set.
- Count value (NEW_CV): The count value is the initial value at which a high-speed counter starts counting. The count value can be in the range -2147483648 to 2147483647.
  The count value specified at the NEW_CV input is loaded into a high-speed counter when the bit at the CV input is set.
- Reference value (NEW_RV): You can compare the reference value with the current counter value to trigger an alarm. Similar to the counter value, the reference value can be in the range -2147483648 to 2147483647.
  The reference value specified at the NEW_RV input is loaded into a high-speed counter when the bit at the RV input is set.
- Period of the frequency measurement (NEW_PERIOD): The period of the frequency measurement is specified by the following values at the NEW_PERIOD input: 10 = 0.01s, 100 = 0.1s, 1000 = 1s.
  The time period can be updated if the "Measure frequency" function for the specified high-speed counter is configured. The time period specified at the NEW_PERIOD input is loaded into a high-speed counter when the bit at the PERIOD input is set.

The "Control high-speed counter" instruction is only executed if the signal state at the EN input is "1".

The ENO enable output is set only when the EN enable input has signal state "1" and no errors occur during execution of the operation.

When inserting the "Control high-speed counter" instruction, an instance data block is created in which the operation data is saved.

STATUS parameter

At the STATUS output, you can query whether errors occurred during execution of the "Control high-speed counter" instruction. The following table shows the meaning of the values output at the STATUS output:

Error code (hexadecimal)	Description
0	No error
80A1	Hardware identifier of the high-speed counter invalid
80B1	Count direction (NEW_DIR) invalid
80B2	Count value (NEW_CV) invalid
80B3	Reference value (NEW_RV) invalid
80B4	Period of the frequency measurement (NEW_PERIOD) invalid
80C0	Multiple access to the high-speed counter
80D0	The high-speed counter (HSC) is not enabled in the CPU hardware configuration.

CTRL_HSC_EXT: Control high-speed counter (extended) (S7-1200)

Parameters

Parameters	Declaration	Data type	Memory area	Description
EN	INPUT	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	OUTPUT	BOOL	I, Q, M, D, L	Enable output
HSC	INPUT	HW_HSC	I, Q, M or constant	Hardware address of the high-speed counter (HW-ID)
CTRL	INOUT	VARIANT	M, D	Use of a system data type (SDT)
DONE	OUTPUT	BOOL	I, Q, M, D, L	Feedback after successful processing instruction
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Processing status
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Feedback for faulty processing of the instruction
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status of the operation

Description

With the "Control high-speed counter (extended)" instruction, you can control and assign parameters to the high-speed counters supported by the CPU via the software by loading new values into the counters. Execution of the instruction requires that a high-speed counter to be controlled is enabled. You cannot execute multiple "Control high-speed counter (extended)" instructions simultaneously in the program for a given high-speed counter.

The "Control high-speed counter (extended)" instruction is only executed if the signal state at the EN input is "1". As long as the operation is executing, the bit at the BUSY output is set. Once the operation has executed completely, the bit at the BUSY output is reset.

The ENO enable output is set only when the EN enable input has signal state "1" and no errors occur during execution of the operation.

When the "Control high-speed counter (extended)" instruction is inserted, an instance data block is created in which the operation data is saved.

Using the system data type HSC_Period

The "Control high-speed counter (extended)" instruction supports the system data type SDT 381 "HSC_Period" for the period measurement.

The "HSC_Period" data type corresponds to an HSC that is configured for the "Period" operating mode. The CTRL_HSC_EXT instruction offers program access to the number of input pulses across a specified measuring interval. This instruction allows the calculation of the period between the input pulses with a resolution in the nanosecond range.

Byte	Parameters	Declaration	Data type	Description
0 … 3	Elapsed-Time	OUT	UDINT	Time between the rising edges of Edge_Count
4 … 7	EdgeCount	OUT	UDINT	Number of the rising edges within the Elapsed_Time. If Edge_Count = 0, the Elapsed_Time is the time since the last rising edge.
8.0	EnHSC	IN	BOOL	Use as an enable input via gate control: • FALSE: Measurement stopped • TRUE: Measurement enabled
8.6	EnPeriod	IN	BOOL	Update of the period • FALSE: No update • TRUE: Update period
10 … 11	NewPeriod	IN	INT	Interval of the period measurement in milliseconds. Permissible values are 10, 100 and 1000.

ElapsedTime specifies the time in nanoseconds between the last counter events of consecutive measuring intervals. If no counter events occurred during a measuring interval, ElapsedTime outputs the accumulated time since the last counter event. ElapsedTime has a range of 0 to 4,294,967,280 nanoseconds (0x0000 0000 to 0xFFFF FFF0). The return value 4,294,967,295 (0xFFFF FFFF) signals that a period overflow has occurred. An overflow indicates that the period between two pulse edges exceeds 4.295 seconds and that the period cannot be calculated with this instruction. The values from 0xFFFF FFF1 to 0xFFFF FFFE are reserved.

EdgeCount outputs the number of counter events received during the measuring interval. The period can only be calculated if the value of EdgeCount is greater than zero. If ElapsedTime is either 0 (no input pulses received) or OR 0xFFFF FFFF (period overflow), EdgeCount is not valid.

If EdgeCount is valid, calculate the period in nanoseconds by using the following formula: Period = ElapsedTime/EdgeCount

The calculated period value is a mean value from time periods of all the pulses that occur during the measuring interval. If the period of an incoming pulse is greater than the measuring interval (10, 100 or 1000 ms), the period calculation requires several measuring intervals.

The following figures show examples of the period measurement with the instruction:

Example 1: Multiple Counting Events in a Measurement Interval

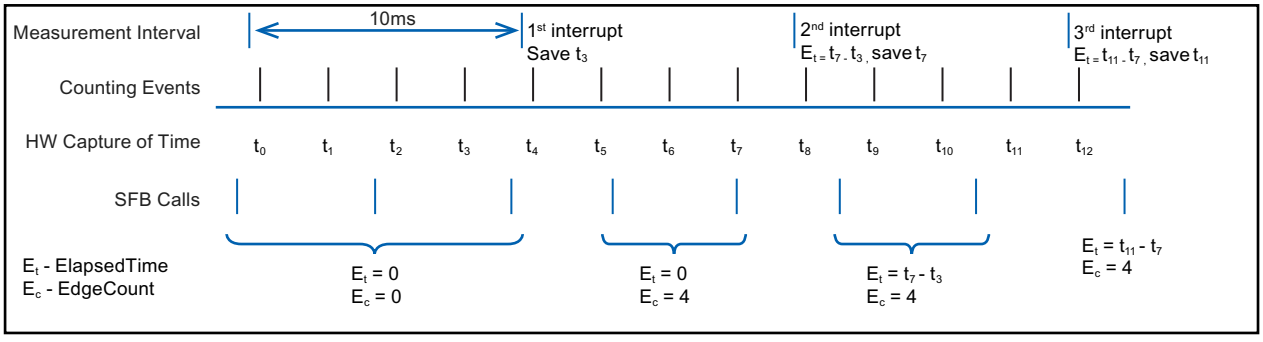

Example 2: Zero and One Counting Events in Multiple Measurement Intervals

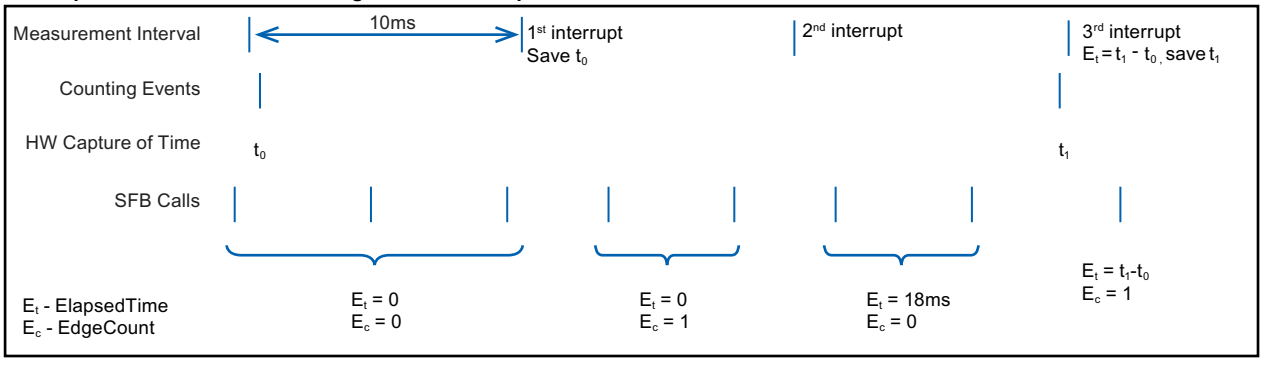

Rules:
1. If $E1_t = 0$, the period is invalid
2. Else, Period = E_t / E_c

STATUS parameter

At the STATUS output, you can query whether errors occurred during execution of the "Control high-speed counter (extended)" instruction. The following table shows the meaning of the values output at the STATUS output:

Error code (hexadecimal)	Description
0	No error
80A1	Hardware identifier of the high-speed counter invalid

Error code (hexadecimal)	Description
80C0	Multiple access to the high-speed counter
80D0	The high-speed counter (HSC) is not enabled in the CPU hardware configuration.

4.1.5.2 Counting and measurement (S7-1500) (S7-1500)

High_Speed_Counter (S7-1500)

High_Speed_Counter (S7-1500)

Description High_Speed_Counter (S7-1500)

Description

The High_Speed_Counter instruction is used to control the technology module counting and measuring functions via the user program.

Call

The instruction High_Speed_Counter must be called once per counter, either cyclically or in a time-controlled program. The call is not permitted in an event-controlled interrupt program.

Operating principle

Counter value: The counter value is available at the output parameter CountValue. The counter value is updated at every call of the High_Speed_Counter instruction.

Measured value: The technology module updates the measured value asynchronously to the instruction call based on the configured update time. The measured value last determined by the technology module is updated at the output parameter MeasuredValue each time the instruction is called.

The measured value and the counter value are available in parallel in the feedback interface.

Capture: The output parameter CaptureStatus = TRUE indicates a valid Capture value at the output parameter CapturedValue.

- A Capture value is captured under the following conditions:
  - A digital input has the parameter assignment "Capture"
  - CaptureEnable = TRUE
  - Edge at digital input with the Capture function
- The output parameter CaptureStatus is reset by a negative edge at the input parameter CaptureEnable.

Synchronization: The output parameter SyncStatus = TRUE indicates that a synchronization has occurred.

- The counter value is synchronized under the following conditions:
  - A digital input has the parameter assignment "Synchronization" **or** the incremental encoder has the parameter assignment "Synchronization at signal N"
  - SyncEnable = TRUE
  - The static tag SyncUpDirection (or SyncDownDirection) = TRUE
  - Edge at the digital input with the synchronization function **or** positive edge of the signal N at the encoder input
- The output parameter SyncStatus is reset by a negative edge at
  - the input parameter SyncEnable or
  - the static tag SyncDownDirection or
  - the static tag SyncUpDirection

Parameter changes via the user program

Proceed as follows to modify parameters using the user program:

1. Check based on the respective Set tag to determine whether the technology object is ready for the parameter change (Set tag = FALSE) or whether a change job is still running (Set tag = TRUE).
   The following Set tags in UserCmdFlags are available for this in the static tags of the technology object instance DB:
   - SetReferenceValue0
   - SetReferenceValue1
   - SetUpperLimit
   - SetLowerLimit
   - SetCountValue
   - SetStartValue
   - SetNewDirection
2. If the technology object is ready for the parameter change, modify the relevant static tag. The following static tags of the technology object instance DB are available for this:
   - NewReferenceValue0 / NewReferenceValue0_M (for SetReferenceValue0)
   - NewReferenceValue1 / NewReferenceValue1_M (for SetReferenceValue1)
   - NewUpperLimit
   - NewLowerLimit
   - NewCountValue
   - NewStartValue
   - NewDirection

3. Set the relevant Set tag for execution of the change command.
4. Use the output parameter Error to check whether an error has occurred.
   If no errors have occurred and the Set tag has been automatically reset by the technology object, the parameter change was successful.

Note

Changed counting limit

If the new high counting limit is less than the current counter value, the counter value is set to the low counting limit or the start value according on the parameter assignment. If the new low counting limit is greater than the current counter value, the counter value is set to the high counting limit or the start value according to the parameter assignment.

Operating mode (High_Speed_Counter V3.0 or higher)

Configure the operating mode in the technology object under "Behavior of DQ0".

The operating mode is indicated by the output parameter CompareMeasuredValue:

State	Description
FALSE	Operating mode "Use count value as reference" The comparison functions work with the counter value. The following static variables are specifically used in this operating mode: • NewReferenceValue0 • NewReferenceValue1 • CurReferenceValue0 • CurReferenceValue1 The four specific static variables of operating mode "Use measured value as reference" are ignored.
TRUE	Operating mode "Use measured value as reference": The comparison functions work with the measured value. The following static variables are specifically used in this operating mode: • NewReferenceValue0_M • NewReferenceValue1_M • CurReferenceValue0_M • CurReferenceValue1_M The four specific static variables of operating mode "Use count value as reference" are ignored.

Acknowledgment of events

You acknowledge signaled events using the positive edge of the input parameter EventAck . EventAck must stay set until the technology object has reset the status bits of the following events of the count channel:

- CompResult0
- CompResult1

- ZeroStatus
- PosOverflow
- NegOverflow

Status of the digital inputs (TM Count and TM PosInput)

You can obtain the status of the digital inputs with the static tags StatusDI0, StatusDI1 or StatusDI2.

Status of the digital inputs (Compact CPU)

You can obtain the status of the digital inputs with the static tags StatusDI0 and StatusDI1. When a digital input of the Compact CPU is not used for a counter, you can use it via the user program.

Use of digital outputs by the user program (TM Count and TM PosInput)

You can set the digital outputs with the High_Speed_Counter instruction,

- If the "Use by user program" setting is configured for "Set output".
- If the "After set command from CPU until comparison value" setting is configured for "Set output".
- if you set the corresponding static tag ManualCtrlDQm (temporary overwrite).

The static tags SetDQ0 and SetDQ1 only have an effect in these cases. In the first and third case, DQm follows the value of SetDQm. In the second case, DQm is set with a positive edge of SetDQm. DQm is reset when the counter value corresponds to the comparison value or at a negative edge of SetDQm.

Use digital outputs by user program (Compact CPU)

You can set the DQ1 digital output with the High_Speed_Counter instruction.

- If the "Use by user program" setting is configured for "Set output".
- If the "After set command from CPU until comparison value" setting is configured for "Set output".
- if you set the corresponding static tag ManualCtrlDQ1 (temporary overwrite).

The static tag SetDQ1 only has an effect in these cases. In the first and third case, DQ1 follows the value of SetDQ1. In the second case, DQ1 is set and reset with a positive edge of SetDQ1 when the counter value corresponds to the comparison value or at a negative edge of SetDQ1.

Note

Before you can set a physical digital output of the Compact CPU with the High_Speed_Counter instruction, you have to assign the DQ1 signal to the desired digital output.

You can set the DQ0 signal using the High_Speed_Counter instruction with the static tag StatusDQ0.

- If the "Use by user program" setting is configured for "Set output".
- If the "After set command from CPU until comparison value" setting is configured for "Set output".
- If you set the static tag ManualCtrlDQ0 (temporary overwrite).

The static tag SetDQ0 only has an effect in these cases. In the first and third case, DQ0 follows the value of SetDQ0. In the second case, StatusDQ0 is set and reset with an edge (positive or negative) by SetDQ0 and is reset when the counter value corresponds to the comparison value.

Note

The digital output DQ0 is not available as a physical output.

Reaction to error

If an error has occurred during the call of the instruction or in the technology module, the output parameter Error is set. More detailed error information can be read at the output parameter ErrorID (Page 2779).

Eliminate the cause of the error and acknowledge the error message by setting the input parameter ErrorAck. When no more errors are pending, the technology object resets the output parameter Error . No new error is signaled until you acknowledge the previous error.

Changing the count direction

The count direction can only be changed by the user program if "Pulse (A)" is configured as the signal type. In all other cases, the count direction is determined by the input signals of the technology module. The count direction is controlled by the static tag NewDirection:

- +1: Upward count direction
- -1: Downward count direction

To execute the change command, you need to set the static tag SetNewDirection = TRUE.

High_Speed_Counter input parameters (S7-1500)

Parameter	Declaration	Data type	Default	Description
SwGate	INPUT	BOOL	FALSE	Control software gate: • Positive edge: Software gate opens • Negative edge: Software gate closes Together with the hardware gate, the SwGate enables the internal gate.
CaptureEnable	INPUT	BOOL	FALSE	Enable Capture function After the enable, a Capture event occurs at the next configured edge at the relevant digital input. A negative edge at CaptureEnable resets the output parameter CaptureStatus. A negative edge at CaptureEnable resets the enable even if no Capture event has occurred. Irrespective of CaptureEnable the last value is retained at the output parameter CapturedValue until the next Capture event.
SyncEnable	INPUT	BOOL	FALSE	Enable synchronization The direction enabled for synchronization is indicated in the static tags SyncUpDirection and SyncDownDirection. A negative edge at SyncEnable resets the output parameter SyncStatus .
ErrorAck	INPUT	BOOL	FALSE	High_Speed_Counter up to V3.0: A positive edge acknowledges the status reported. High_Speed_Counter as of V3.1: A high level acknowledges the error status reported.
EventAck	INPUT	BOOL	FALSE	A positive edge resets the following output parameters: • CompResult0 • CompResult1 • ZeroStatus • PosOverflow • NegOverflow
SetCountValue	INOUT	BOOL	FALSE	A positive edge starts the transfer of the new counter value in the static tag NewCountValue to the technology module. The counter value takes effect immediately after the transfer.

High_Speed_Counter output parameters (S7-1500)

Parameter	Declaration	Data type	Default	Description
StatusHW	OUTPUT	BOOL	FALSE	Status bit technology module: The module is configured and ready for operation. The module data is valid.
StatusGate	OUTPUT	BOOL	FALSE	Status bit: Internal gate is released if parameter is set
StatusUp	OUTPUT	BOOL	FALSE	Status bit: The last count pulse incremented the counter and took place no more than 0.5 s ago
StatusDown	OUTPUT	BOOL	FALSE	Status bit: The last count pulse decremented the counter and took place no more than 0.5 s ago

Parameter	Declaration	Data type	Default	Description
CompResult0	OUTPUT	BOOL	FALSE	Status bit: Compare event for DQ0 occurred (status change) based on the selected comparison condition. If the counter value is set to the start value in operating mode "Use count value as reference", CompResult0 is not set. The positive edge of the input parameter EventAck is used to reset CompResult0 .
CompResult1	OUTPUT	BOOL	FALSE	Status bit: Compare event for DQ1 occurred (status change) based on the selected comparison condition. If the counter value is set to the start value in operating mode "Use count value as reference", CompResult1 is not set. The positive edge of the input parameter EventAck is used to reset CompResult1 .
SyncStatus	OUTPUT	BOOL	FALSE	Status bit: Synchronization event occurred If the input parameter SyncEnable is set, the configured edge sets the status bit SyncStatus at the respective digital input . SyncStatus is reset by a negative edge at • SyncEnable (input parameter) or • SyncUpDirection (static tag) or • SyncDownDirection (static tag)
CaptureStatus	OUTPUT	BOOL	FALSE	Status bit: Capture event occurred, the output parameter CapturedValue has a valid Capture value If the input parameter CaptureEnable is set, the configured edge sets the status bit CaptureStatus. at the respective digital input. You reset CaptureStatus using the negative edge of the input parameter CaptureEnable .
ZeroStatus	OUTPUT	BOOL	FALSE	Status bit: CountValue has reached the value "0" irrespective of the count direction You reset ZeroStatus using the positive edge of the input parameter EventAck.
PosOverflow	OUTPUT	BOOL	FALSE	Status bit: CountValue has violated high counting limit in the positive direction You reset PosOverflow using the positive edge of the input parameter EventAck.
NegOverflow	OUTPUT	BOOL	FALSE	Status bit: CountValue has violated low counting limit in the negative direction You reset NegOverflow using the positive edge of the input parameter EventAck.
Error	OUTPUT	BOOL	FALSE	An error has occurred. Refer to the output parameter ErrorID for the cause of the error.
ErrorID	OUTPUT	WORD	0	The ErrorID (Page 2779) parameter displays the number of the error message. ErrorID = 0000_H: There is no error.
CountValue	OUTPUT	DINT	0	Current counter value

Parameter	Declaration	Data type	Default	Description
CapturedValue	OUTPUT	DINT	0	The last acquired Capture value The value is retained until the next Capture event, irrespective of the input parameter CaptureEnable. If a new Capture event has occurred, CaptureStatus is set and is reset by you using the negative edge of the input parameter CaptureEnable .
MeasuredValue	OUTPUT	REAL	0.0	Current measured value for frequency, period duration or velocity (depending on configuration)
CompareMeasuredValue[1]	OUTPUT	BOOL	FALSE	Status bit: FALSE: Operating mode "Use count value as reference"; comparison functions work with counter value TRUE: Operating mode "Use measured value as reference"; comparison functions work with measured value

[1] Available for High_Speed_Counter version V3.0 or higher

High_Speed_Counter static variables (S7-1500)

Tag	Data type	Default	Access	Description
NewCountValue	DINT	L#0	Write	New counter value
NewReferenceValue0	DINT	L#0	Write	New comparison value 0 in the operating mode "Use count value as reference"
NewReferenceValue1	DINT	L#10	Write	New comparison value 1 in the operating mode "Use count value as reference"
NewReferenceValue0_M[1]	REAL	L#0.0	Write	New comparison value 0 in operating mode "Use measured value as reference"
NewReferenceValue1_M[1]	REAL	L#10.0	Write	New comparison value 1 in operating mode "Use measured value as reference"
NewUpperLimit	DINT	L#2147483647	Write	New high counting limit
NewLowerLimit	DINT	L#-2147483648	Write	New low counting limit
NewStartValue	DINT	L#0	Write	New start value
NewDirection	INT	0	Write	New count direction: +1: Upward count direction -1: Downward count direction
CurReferenceValue0	DINT	L#0	Read	Current comparison value 0 in operating mode "Use count value as reference"
CurReferenceValue1	DINT	L#10	Read	Current comparison value 1 in operating mode "Use count value as reference"
CurReferenceValue0_M[1]	REAL	L#0.0	Read	Current comparison value 0 in operating mode "Use measured value as reference"
CurReferenceValue1_M[1]	REAL	L#10.0	Read	Current comparison value 1 in operating mode "Use measured value as reference"
CurUpperLimit	DINT	L#2147483647	Read	Current high counting limit
CurLowerLimit	DINT	L#-2147483648	Read	Current low counting limit
CurStartValue	DINT	L#0	Read	Current start value

Tag		Data type	Default	Access	Description
AdditionalErrorID		DWORD	W#16#0000	Read	Error information of an internal instruction, e.g.,RDREC
UserCmdFlags		STRUCT	-		
	SetNewDirection	BOOL	FALSE	Write	Set new count direction
	SetUpperLimit	BOOL	FALSE	Write	Set high counting limit
	SetLowerLimit	BOOL	FALSE	Write	Set low counting limit
	SetReferenceValue0	BOOL	FALSE	Write	Set comparison value 0
	SetReferenceValue1	BOOL	FALSE	Write	Set comparison value 1
	SetStartValue	BOOL	FALSE	Write	Set start value
	SyncDownDirection	BOOL	TRUE	Write	Enable synchronization in downward count direction
	SyncUpDirection	BOOL	TRUE	Write	Enable synchronization in upward count direction
	SetDQ0	BOOL	FALSE	Write	Set digital output DQ0
	SetDQ1	BOOL	FALSE	Write	Set digital output DQ1
	ManualCtrlDQ0	BOOL	FALSE	Write	Enable setting of digital output DQ0: TRUE: • SetDQ0 sets DQ0 • Control bit TM_CTRL_DQ0 = FALSE FALSE: • Setting not enabled • Control bit TM_CTRL_DQ0 = TRUE
	ManualCtrlDQ1	BOOL	FALSE	Write	Enable setting of digital output DQ1: TRUE: • SetDQ1 sets DQ1 • Control bit TM_CTRL_DQ1 = FALSE FALSE: • Setting not enabled • Control bit TM_CTRL_DQ1 = TRUE
UserStatusFlags		STRUCT	-		
	StatusDI0	BOOL	FALSE	Read	Current status of digital input DI0
	StatusDI1	BOOL	FALSE	Read	Current status of digital input DI1
	StatusDI2	BOOL	FALSE	Read	Current status of digital input DI2
	StatusDQ0	BOOL	FALSE	Read	Current status of digital output DQ0
	StatusDQ1	BOOL	FALSE	Read	Current status of digital output DQ1

[1] Available for High_Speed_Counter version V3.0 or higher

Error codes of parameter ErrorID (S7-1500)

Error code (W#16#...)	Description
0000	No error
Error messages from technology module	
80A1	POWER_ERROR from feedback interface: Incorrect supply voltage L+
80A2	ENC_ERROR from feedback interface: Incorrect encoder signal
80A3	LD_ERROR from feedback interface: Error when loading via control interface
Error messages of the instruction High_Speed_Counter	
80B1	Invalid count direction
80B4	For operating mode "Use count value as reference", the following applies: New low counting limit does not meet the following conditions: • Low counting limit < high counting limit • Low counting limit <= comparison value/start value For operating mode "Use measured value as reference", the following applies: New low counting limit does not meet the following conditions: • Low counting limit < high counting limit • Low counting limit <= start value
80B5	For operating mode "Use count value as reference", the following applies: New high counting limit does not meet the following conditions: • Low counting limit < high counting limit • High counting limit >= comparison value/start value For operating mode "Use measured value as reference", the following applies: New high counting limit does not meet the following conditions: • Low counting limit < high counting limit • High counting limit >= start value
80B6	New start value does not meet the following condition: • Low counting limit <= start value <= high counting limit
80B7	New counter value does not meet the following condition: • Low counting limit <= counter value <= high counting limit
80B8	For operating mode "Use count value as reference", the following applies: New comparison value 0 does not meet the following conditions: • Low counting limit <= comparison value 0 <= high counting limit • Comparison value 0 < comparison value 1 For operating mode "Use measured value as reference", the following applies: New comparison value 0 does not meet the following conditions: • Comparison value 0 < comparison value 1

Error code (W#16#...)	Description
80B9	For operating mode "Use count value as reference", the following applies: New comparison value 1 does not meet the following conditions: • Low counting limit <= comparison value 1 <= high counting limit • Comparison value 0 < comparison value 1 For operating mode "Use measured value as reference", the following applies: New comparison value 1 does not meet the following conditions: • Comparison value 0 < comparison value 1
80C0	Instruction High_Speed_Counter was called multiple times with the same instance (DB)
80C1	Communication with technology module failed (read data records): Error information of internal instruction RDREC saved in static tag AdditionalErrorID
80C2	Communication with technology module failed (write data records): Error information of internal instruction WRREC saved in static tag AdditionalErrorID
80C3	Access to input data (feedback interface) failed: Error information of internal instruction GETIO_PART saved in static tag AdditionalErrorID
80C4	Access to output data (control interface) failed: Error information of internal instruction SETIO_PART saved in static tag AdditionalErrorID
80C5	Reading of the current start information of the OB failed: Error information of internal instruction RD_SINFO saved in static tag AdditionalErrorID
80C6	Failed to get I/O addresses of the technology module: Error information of internal instruction RD_ADDR saved in static tag AdditionalErrorID
80C7	Module not inserted or no supply voltage L+

SSI_Absolute_Encoder (S7-1500)

SSI_Absolute_Encoder (S7-1500)

Description SSI_Absolute_Encoder (S7-1500)

Description

The SSI_Absolute_Encoder instruction is used to control the position input and measuring functions of the technology module TM PosInput via the user program.

Call

The instruction must be called once per channel, either cyclically or in a time-controlled program. The call is not permitted in an event-controlled interrupt program.

Operating principle

Position value: The position value is available at the output parameter PositionValue. The position value is updated every time the instruction is called.

Measured value: The technology module updates the measured value asynchronously to the instruction call based on the configured update time. The measured value last determined by the technology module is updated at the output parameter MeasuredValue each time the instruction is called.

Measured value and position value are available in parallel as output parameters.

Instead of a measured value, the complete SSI frame can be returned at the output parameter CompleteSSIFrame. Either MeasuredValue or CompleteSSIFrame is valid, depending on the parameter assignment.

Capture: The output parameter CaptureStatus = TRUE indicates a valid Capture value at the output parameter CapturedValue.

- A Capture value is captured under the following conditions:
  - A digital input has the parameter assignment "Capture"
  - CaptureEnable = TRUE
  - Edge at digital input with the Capture function
- The output parameter CaptureStatus is reset by a negative edge at the input parameter CaptureEnable.

Parameter changes via the user program

Proceed as follows to modify parameters using the user program:

1. Check the relevant Set tag to establish whether the technology object is ready for the parameter change (Set tag = FALSE) or whether a change job is still running (Set tag = TRUE).
   The following Set tags in UserCmdFlags are available for this in the static tags of the technology object instance DB:
   - SetReferenceValue0
   - SetReferenceValue1
2. If the technology object is ready for the parameter change, modify the relevant static tag.
   The following static tags of the technology object instance DB are available for this:
   - NewReferenceValue0 / NewReferenceValue0_M (for SetReferenceValue0)
   - NewReferenceValue1 / NewReferenceValue1_M (for SetReferenceValue1)
3. Set the relevant Set tag for execution of the change job.
4. Use the output parameter Error to check whether an error has occurred.
   If no errors have occurred and the Set tag has been automatically reset by the technology object, the parameter change was successful.

Operating mode

Configure the operating mode in the technology object under "Behavior of DQ0".

The operating mode is indicated by the output parameters CompareMeasuredValue:

State	Description
FALSE	Operating mode "Use position value (SSI absolute value) as reference": The comparison functions work with the position value. The following static variables are specifically used in this operating mode: • NewReferenceValue0 • NewReferenceValue1 • CurReferenceValue0 • CurReferenceValue1 The four specific static variables of operating mode "Use measured value as reference" are ignored.
TRUE	Operating mode "Use measured value as reference": The comparison functions work with the measured value. The following static variables are specifically used in this operating mode: • NewReferenceValue0_M • NewReferenceValue1_M • CurReferenceValue0_M • CurReferenceValue1_M The four specific static tags of operating mode "Use position value (SSI absolute value) as reference" are ignored.

Acknowledgment of events

You acknowledge signaled events using the positive edge of the input parameter EventAck . EventAck must stay set until the technology object has reset the status bits of the following events of the count channel:

- CompResult0
- CompResult1
- ZeroStatus
- PosOverflow
- NegOverflow

Status of the digital inputs

You obtain the status of the digital inputs via the static tags StatusDI0 and StatusDI1.

Using digital outputs with user program

In the following cases you can set the digital outputs via the instruction:

Case	Description
The setting "Use by user program" is configured for "Set output".	The respective digital output DQM follows the value of SetDQm.
The setting "After set command from CPU until comp. value" is configured for "Set output".	The respective digital output DQm is set with a negative edge of SetDQm. DQm is reset when the position value corresponds to the comparison value or at a negative edge of SetDQm.
Set the respective static variable ManualCtrlDQm (temporary overwrite).	The respective digital output DQM follows the value of SetDQm.

Reaction to error

If an error has occurred during the call of the instruction or in the technology module, the output parameter Error is set. More detailed error information can be read at the output parameter ErrorID (Page 2786).

Eliminate the cause of the error and acknowledge the error message by setting the input parameter ErrorAck. When no more errors are pending, the technology object resets the output parameter Error . No new error is signaled until you acknowledge the previous error.

Input parameter SSI_Absolute_Encoder (S7-1500)

Parameter	Declaration	Data type	Default	Description
CaptureEnable	INPUT	BOOL	FALSE	Enable Capture function After the enable, a Capture event occurs at the next configured edge at the relevant digital input. A negative edge at CaptureEnable resets the output parameter CaptureStatus. A negative edge at CaptureEnable resets the enable even if no Capture event has occurred. Irrespective of CaptureEnable the last value is retained at the output parameter CapturedValue until the next Capture event.
ErrorAck	INPUT	BOOL	FALSE	SSI_Absolute_Encoder V1.0: A positive edge acknowledges the error status reported. SSI_Absolute_Encoder as of V2.0: A high level acknowledges the error status reported.
EventAck	INPUT	BOOL	FALSE	A positive edge resets the following output parameters: • CompResult0 • CompResult1 • ZeroStatus • PosOverflow • NegOverflow

Output parameter SSI_Absolute_Encoder (S7-1500)

Parameter	Declaration	Data type	Default	Description
StatusHW	OUTPUT	BOOL	FALSE	Status bit technology module: The module is configured and ready for operation. The module data is valid.
StatusUp	OUTPUT	BOOL	FALSE	Status bit: Last position value has changed in the positive direction and took place no more than 0.5 s ago.
StatusDown	OUTPUT	BOOL	FALSE	Status bit: Last position value has changed in the negative direction and took place no more than 0.5 s ago.
CompResult0	OUTPUT	BOOL	FALSE	Status bit: Comparison event for DQ0 (change of state) occurs as a result of the selected comparison condition. The positive edge of the input parameter EventAck is used to reset CompResult0 .
CompResult1	OUTPUT	BOOL	FALSE	Status bit: Comparison event for DQ1 (change of state) occurs as a result of the selected comparison condition. The positive edge of the input parameter EventAck is used to reset CompResult1 .
CaptureStatus	OUTPUT	BOOL	FALSE	Status bit: Capture event occurred, the output parameter CapturedValue has a valid Capture value If the input parameter CaptureEnable is set, the configured edge sets the status bit CaptureStatus. at the respective digital input. You reset CaptureStatus using the negative edge of the input parameter CaptureEnable .
CapturedValue	OUTPUT	DINT	0	The last acquired Capture value The value is retained until the next Capture event, irrespective of the input parameter CaptureEnable. If a new Capture event has occurred, CaptureStatus is set and is reset by you using the negative edge of the input parameter CaptureEnable .
ZeroStatus	OUTPUT	BOOL	FALSE	Status bit: PositionValue has reached or exceeded the value "0" irrespective of the count direction. You reset ZeroStatus using the positive edge of the input parameter EventAck.
PosOverflow	OUTPUT	BOOL	FALSE	Status bit: PositionValue has exceeded the high limit of the position value range of the encoder in the positive direction. You reset PosOverflow using the positive edge of the input parameter EventAck.
NegOverflow	OUTPUT	BOOL	FALSE	Status bit: PositionValue has exceeded the low limit of the position value range of the encoder in the negative direction. You reset NegOverflow using the positive edge of the input parameter EventAck.
Error	OUTPUT	BOOL	FALSE	An error has occurred. Refer to the output parameter ErrorID for the cause of the error.
ErrorID	OUTPUT	WORD	0	The ErrorID (Page 2786) parameter displays the number of the error message. ErrorID = 0000_H: There is no error.
PositionValue	OUTPUT	DINT	0	Current position value

Parameter	Declaration	Data type	Default	Description
MeasuredValue	OUTPUT	REAL	0.0	Current measured value for frequency, period duration or velocity (depending on configuration) Either MeasuredValue or CompleteSSIFrame is valid, depending on the parameter assignment in the technology object under "Measured value".
CompleteSSIFrame	OUTPUT	DWORD	0	Last received complete SSI frame (least significant 32 bits) Either MeasuredValue or CompleteSSIFrame is valid, depending on the parameter assignment in the technology object under "Measured value".
CompareMeasuredValue	OUTPUT	BOOL	FALSE	Status bit: FALSE: Operating mode "Use position value (SSI absolute value) as reference"; position value is used as reference TRUE: Operating mode "Use measured value as reference"; measured value is used as reference

Static tags SSI_Absolute_Encoder (S7-1500)

Tag		Data type	Default	Access	Description
NewReferenceValue0		DINT	L#0	Write	New comparison value 0 in operating mode "Use position value (SSI absolute value) as reference"
NewReferenceValue1		DINT	L#10	Write	New comparison value 1 in operating mode "Use position value (SSI absolute value) as reference"
NewReferenceValue0_M		REAL	L#0.0	Write	New comparison value 0 in operating mode "Use measured value as reference"
NewReferenceValue1_M		REAL	L#10.0	Write	New comparison value 1 in operating mode "Use measured value as reference"
CurReferenceValue0		DINT	L#0	Read	Current comparison value 0 in operating mode "Use position value (SSI absolute value) as reference"
CurReferenceValue1		DINT	L#10	Read	Current comparison value 1 in operating mode "Use position value (SSI absolute value) as reference"
CurReferenceValue0_M		REAL	L#0.0	Read	Current comparison value 0 in operating mode "Use measured value as reference"
CurReferenceValue1_M		REAL	L#10.0	Read	Current comparison value 1 in operating mode "Use measured value as reference"
AdditionalErrorID		DWORD	W#16#0000	Read	Error information of an internal instruction, e.g.,RDREC
UserCmdFlags		STRUCT	-		
	SetReferenceValue0	BOOL	FALSE	Write	Set comparison value 0
	SetReferenceValue1	BOOL	FALSE	Write	Set comparison value 1
	SetDQ0	BOOL	FALSE	Write	Set digital output DQ0
	SetDQ1	BOOL	FALSE	Write	Set digital output DQ1

Tag		Data type	Default	Access	Description
	ManualCtrlDQ0	BOOL	FALSE	Write	Enable setting of digital output DQ0. TRUE: SetDQ0 sets DQ0[1] FALSE: Setting not enabled
	ManualCtrlDQ1	BOOL	FALSE	Write	Enable setting of digital output DQ1: TRUE: SetDQ1 sets DQ1[1] FALSE: Setting not enabled
UserStatusFlags		STRUCT	-		
	StatusDI0	BOOL	FALSE	Read	Current status of digital input DI0
	StatusDI1	BOOL	FALSE	Read	Current status of digital input DI1
	StatusDQ0	BOOL	FALSE	Read	Current status of digital output DQ0
	StatusDQ1	BOOL	FALSE	Read	Current status of digital output DQ1

[1] The instruction sets the TM_CTRL_DQm bit to FALSE. in the control interface of the module. The static tag SetDQm acts on the control bit SET_DQm.

Error codes of parameter ErrorID (S7-1500)

Error code (W#16#...)	Description
0000	No error
Error messages from technology module	
80A1	POWER_ERROR from feedback interface: Incorrect supply voltage L+
80A2	ENC_ERROR from feedback interface: Incorrect encoder signal
80A3	LD_ERROR from feedback interface: Error when loading via control interface
Error messages of the instruction SSI_Absolute_Encoder	
80B8	New comparison value 0 does not meet the following conditions: • Low counting limit <= comparison value 0 <= high counting limit • Comparison value 0 < comparison value 1
80B9	New comparison value 1 does not meet the following conditions: • Low counting limit <= comparison value 1 <= high counting limit • Comparison value 0 < comparison value 1
80C0	Instruction was called multiple times with the same instance (DB)
80C1	Communication with technology module failed (read data records): Error information of internal instruction RDREC saved in static tag AdditionalErrorID
80C2	Communication with technology module failed (write data records): Error information of internal instruction WRREC saved in static tag AdditionalErrorID
80C5	Reading of the current start information of the OB failed: Error information of internal instruction RD_SINFO saved in static tag AdditionalErrorID
80C6	Failed to get I/O addresses of the technology module: Error information of internal instruction RD_ADDR saved in static tag AdditionalErrorID
80C7	Module not inserted or no supply voltage L+

4.1.5.3 PID Control (S7-1200, S7-1500)

PID_Compact (S7-1200, S7-1500)

New features of PID_Compact (S7-1200, S7-1500)

PID_Compact V2.4

- **Initialization of the integral action**
  PID_Compact now initializes the integral action if you use OverwriteInitialOutputValue together with inverted control logic.
  If you have been using OverwriteInitialOutputValue together with inverted control logic up to now, please note that the sign of the output value changes with PID_Compact V2.4.

PID_Compact V2.3

- **Response of the output value when switching from "Inactive" operating mode to "Automatic mode"**
  The new option IntegralResetMode = 4 was added and defined as default. With IntegralResetMode = 4, the integral action is automatically pre-assigned when switching from "Inactive" operating mode to "Automatic mode" so that a control deviation results in a jump of the output value with identical sign.
- **Initialization of the integral action in automatic mode**
  The integral action can be initialized in automatic mode with the tags OverwriteInitialOutputValue and PIDCtrl.PIDInit. This simplifies the use of PID_Compact for override controls.

PID_Compact V2.2

- **Use with S7-1200**
  As of PID_Compact V2.2, the instruction with V2 functionality can also be used on S7-1200 with firmware version 4.0 or higher.

PID_Compact V2.0

- **Reaction to error**
  The reaction to error has been completely overhauled. PID_Compact now reacts in a more fault-tolerant manner in the default setting. This reaction is set when copying PID_Compact V1.X from an S7-1200 CPU to an S7-1500 CPU.

NOTICE
Your system may be damaged. If you use the default setting, PID_Compact remains in automatic mode when the process value limits are exceeded. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

  The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. Use ErrorAck to acknowledge the errors and warnings without restarting the controller or clearing the integral action. Switching operating modes no longer clears errors that are no longer pending.
  You can configure the reaction to error with SetSubstituteOutput and ActivateRecoverMode.
- **Substitute output value**
  You can configure a substitute output value that is to be output if an error occurs.
- **Switching the operating mode**
  You specify the operating mode at the Mode in/out parameter and use a rising edge at ModeActivate to start the operating mode. The sRet.i_Mode tag has been omitted.
- **Multi-instance capability**
  You can call up PID_Compact as multi-instance DB. No technology object is created in this case and no parameter assignment interface or commissioning interface is available. You must assign parameters for PID_Compact directly in the multi-instance DB and commission it via a watch table.
- **Startup characteristics**
  The operating mode specified at the Mode parameter is also started on a falling edge at Reset and during a CPU cold restart, if RunModeByStartup = TRUE.
- **ENO characteristics**
  ENO is set depending on the operating mode.
  If State = 0, then ENO = FALSE.
  If State ≠ 0, then ENO = TRUE.
- **Setpoint value specification during tuning**
  You configure the permitted fluctuation of the setpoint during tuning at the CancelTuningLevel tag.
- **Value range for output value limits**
  The value 0.0 no longer has to fall within the output value limits.
- **Pre-assigning the integral action**
  Using the tags IntegralResetMode and OverwriteInitialOutputValue, you can determine the pre-assignment of the integral action when switching from "Inactive" operating mode to "Automatic mode".

- **Switching a disturbance variable on**
  You can switch a disturbance variable on at the Disturbance parameter.
- **Default value of PID parameters**
  The following default settings have been changed:
  - Proportional action weighting (PWeighting) from 0.0 to 1.0
  - Derivative action weighting (DWeighting) from 0.0 to 1.0
  - Coefficient for derivative delay (TdFiltRatio) from 0.0 to 0.2
- **Renaming tags**
  The static tags have been given new names that are compatible with PID_3Step.

PID_Compact V1.2

- **Manual mode on CPU startup**
  If ManualEnable = TRUE when the CPU starts, PID_Compact starts in manual mode. A rising edge at ManualEnable is not necessary.
- **Pretuning**
  If the CPU is switched off during pretuning, pretuning starts again when the CPU is switched back on.

PID_Compact V1.1

- **Manual mode on CPU startup**
  When the CPU starts up, PID_Compact only switches to manual mode with a rising edge at ManualEnable. Without rising edge, PID_Compact starts in the last operating mode in which ManualEnable was FALSE.
- **Reaction to reset**
  A rising edge at Reset resets the errors and warnings and clears the integral action. A falling edge at Reset triggers a switchover to the most recently active operating mode.
- **Default of process value high limit**
  The default value of r_Pv_Hlm has been changed to 120.0.
- **Monitoring the sampling time**
  - An error is no longer output when the current sampling time is ≥ 1.5 x current mean value or when the current sampling time is ≤ 0.5 x current mean value. The sampling time may deviate much more in automatic mode.
  - PID_Compact is compatible with FW, V2.0 or higher.
- **Access to tags**
  The following tags can now be used in the user program.
  - i_Event_SUT
  - i_Event_TIR
  - r_Ctrl_Ioutv
- **Troubleshooting**
  PID_Compact now outputs the correct pulses when the shortest ON time is not equal to the shortest OFF time.

Compatibility with CPU and FW (S7-1200, S7-1500)

The following table shows which version of PID_Compact can be used on which CPU.

CPU	FW	PID_Compact
S7-1200	V4.2 or higher	V2.3 V2.2 V1.2
	V4.0 to V4.1	V2.2 V1.2
	V3.x	V1.2 V1.1
	V2.x	V1.2 V1.1
	V1.x	V1.0
S7-1500	as of V2.5	V2.4 V2.3 V2.2 V2.1 V2.0
	V2.0 and V2.1	V2.3 V2.2 V2.1 V2.0
	V1.5 to V1.8	V2.2 V2.1 V2.0
	V1.1	V2.1 V2.0
	V1.0	V2.0

CPU processing time and memory requirement PID_Compact V2.x (S7-1200, S7-1500)

CPU processing time

Typical CPU processing times of the PID_Compact technology object as of Version V2.0, depending on CPU type.

CPU	Typ. CPU processing time PID_Compact V2.x
CPU 1211C ≥ V4.0	300 µs
CPU 1215C ≥ V4.0	300 µs
CPU 1217C ≥ V4.0	300 µs
CPU 1505S ≥ V1.0	45 µs
CPU 1510SP-1 PN ≥ V1.6	85 µs
CPU 1511-1 PN ≥ V1.5	85 µs

CPU 1512SP-1 PN ≥ V1.6	85 µs
CPU 1516-3 PN/DP ≥ V1.5	50 µs
CPU 1518-4 PN/DP ≥ V1.5	4 µs

Memory requirement

Memory requirement of an instance DB of the PID_Compact technology object as of Version V2.0.

Memory requirement	Memory requirement of the instance DB of PID_Compact V2.x
Load memory requirement	Approx. 12000 bytes
Total work memory requirement	788 bytes
Retentive work memory requirement	44 bytes

PID_Compact V2 (S7-1200, S7-1500)

Description of PID_Compact V2 (S7-1200, S7-1500)

Description

The PID_Compact instruction provides a PID controller with integrated tuning for actuators with proportional action.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Substitute output value with error monitoring

For a more detailed description of the operating modes, see the State parameter.

PID algorithm

PID_Compact is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation:

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
T_D	Derivative action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
c	Derivative action weighting

Block diagram of PID_Compact

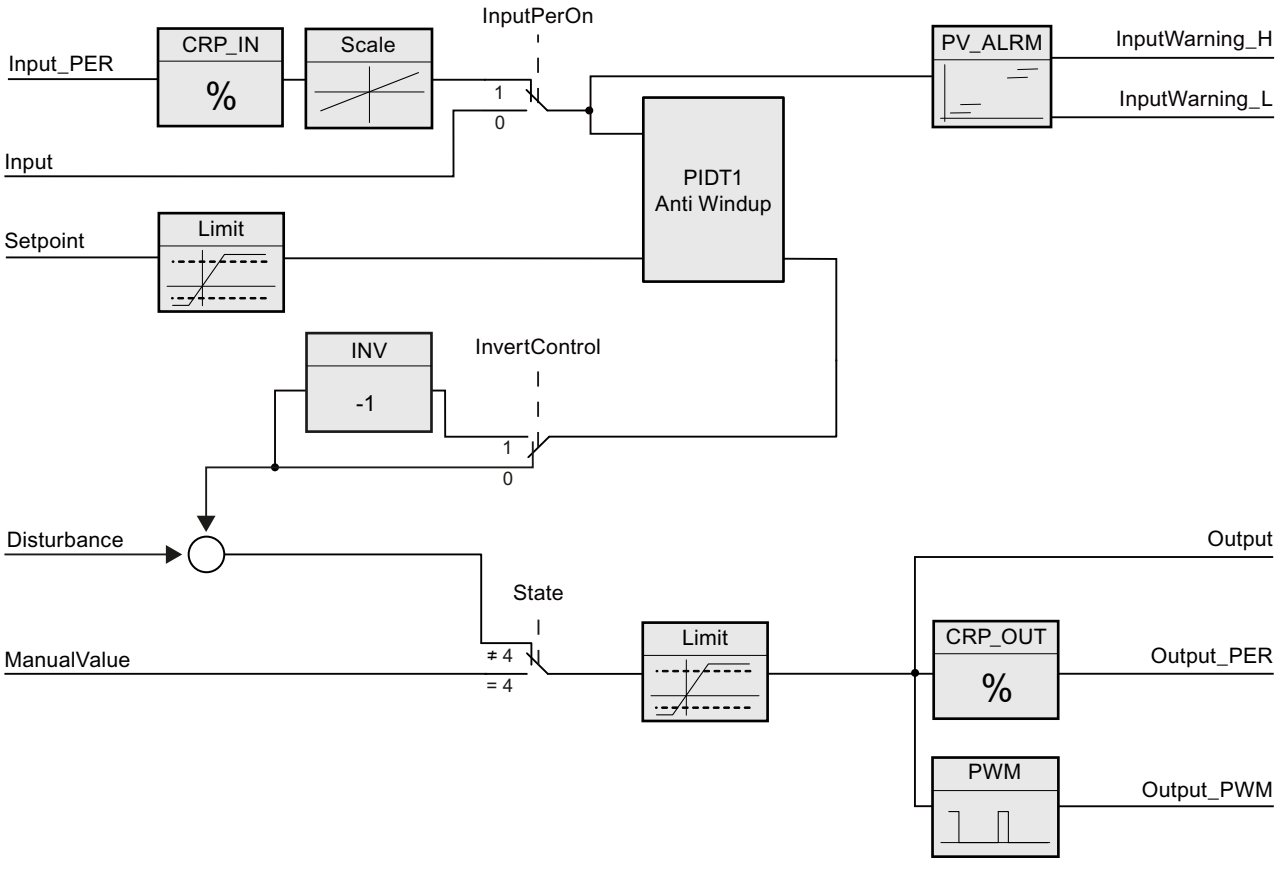

Block diagram of PIDT1 with anti-windup

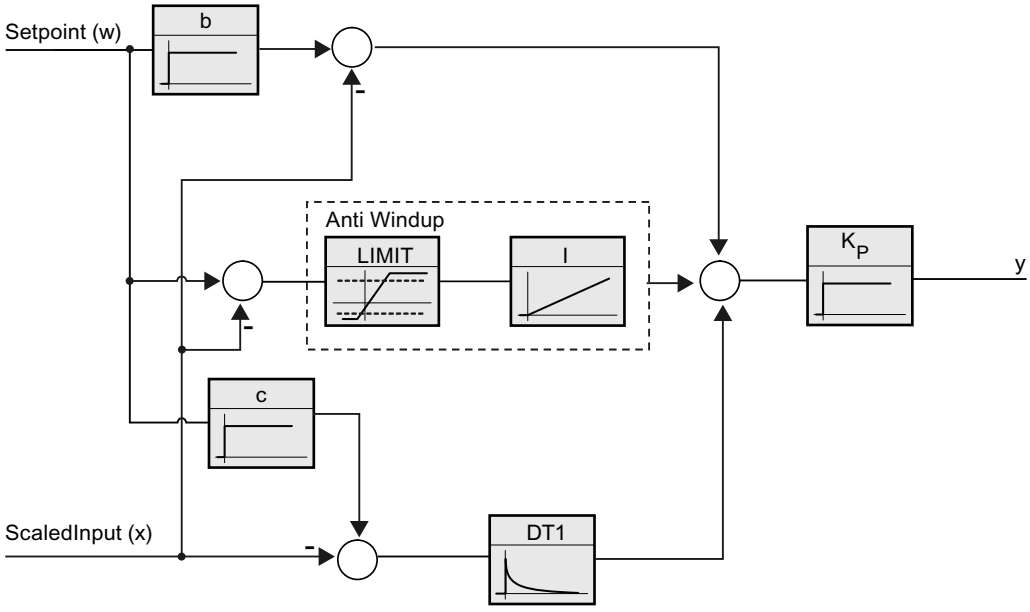

Call

PID_Compact is called in the constant time scale of a cycle interrupt OB.

If you call PID_Compact as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_Compact directly in the multi-instance DB and commission it via a watch table.

Download to device

The actual values of retentive variables are only updated when you download PID_Compact completely.

AUTOHOTSPOT

Startup

When the CPU starts up, PID_Compact starts in the operating mode that is saved in the Mode in/out parameter. To switch to "Inactive" operating mode during startup, set RunModeByStartup = FALSE.

Reaction to error

In automatic mode and during commissioning, the reaction to error depends on the SetSubstituteOutput and ActivateRecoverMode variables. In manual mode, the reaction is independent of SetSubstituteOutput and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the reaction additionally depends on the error that occurred.

SetSubstitu-teOutput	ActivateReco-verMode	Configuration editor > output value > Set Output to	Reaction
Not relevant	FALSE	Zero (inactive)	Switch to "Inactive" mode (State = 0) The value 0.0 0 is transferred to the actuator.
FALSE	TRUE	Current output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The current output value is transferred to the actuator while the error is pending.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The value at SubstituteOutput is transferred to the actuator while the error is pending.

In manual mode, PID_Compact uses ManualValue as output value, unless ManualValue is invalid. If ManualValue is invalid, SubstituteOutput is used. If ManualValue and SubstituteOutput are invalid, Config.OutputLowerLimit is used.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.

Mode of operation of PID_Compact V2 (S7-1200, S7-1500)

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warning = 0040h), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID_Compact automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID_Compact checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. Output, ManualValue and SubstituteOutput are limited to these values. The output value limits must match the control logic.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0%
Output_PER	-100.0 to 100.0%
Output_PWM	0.0 to 100.0%

Rule:

OutputUpperLimit > OutputLowerLimit

Note

Use with two or more actuators

PID_Compact is not suitable for use with two or more actuators (for example, in heating/cooling applications), because different actuators need different PID parameters to achieve a good control response. Use PID_Temp for applications with two actuators acting in opposite directions.

Substitute output value

In the event of an error, PID_Compact can output a substitute output value that you define at the SubstituteOutput tag. The substitute output value must be within the output value limits.

Monitoring signal validity

The values of the following parameters are monitored for validity when used:

- Setpoint
- Input
- Input_PER
- Disturbance
- ManualValue
- SubstituteOutput
- Output
- Output_PER
- Output_PWM

Monitoring of the sampling time PID_Compact

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_Compact instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (Error = 0800h).

The error occurs during tuning if:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

The error occurs in automatic mode if:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID_Compact in OB1. You must then accept a lower control quality due to the deviating sampling time.

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

If you use Output_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The cycle time should be at least 10 times the PID algorithm sampling time.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_Compact does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

Input parameters of PID_Compact V2 (S7-1200, S7-1500)

Table 4-58

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A tag of the user program is used as source for the process value. If you are using parameter Input, then Config.InputPer-On = FALSE must be set.

Parameter	Data type	Default	Description
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using parameter Input_PER, then Config.InputPerOn = TRUE must be set.
Disturbance	REAL	0.0	Disturbance variable or precontrol value
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge activates "manual mode", while State = 4, Mode remains unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog. • A TRUE -> FALSE edge activates the operating mode that is specified by Mode. We recommend that you change the operating mode using ModeActivate only.
ManualValue	REAL	0.0	Manual value This value is used as the output value in manual mode. Values from Config.OutputLowerLimit to Config.OutputUpperLimit are permitted.
ErrorAck	BOOL	FALSE	• FALSE -> TRUE edge ErrorBits and Warning are reset.
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Switch to "Inactive" mode – ErrorBits and Warnings are reset. • As long as Reset = TRUE, – PID_Compact remains in "Inactive" mode (State = 0). – You cannot change the operating mode with Mode and ModeActivate or ManualEnable. – You cannot use the commissioning dialog. • TRUE -> FALSE edge – If ManualEnable = FALSE, PID_Compact switches to the operating mode that is saved in Mode. – If Mode = 3, the integral action is treated as configured with the tag IntegralResetMode.
ModeActivate	BOOL	FALSE	• FALSE -> TRUE edge PID_Compact switches to the operating mode that is saved in the Mode parameter.

Output parameters of PID_Compact V2 (S7-1200, S7-1500)

Table 4-59

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
The "Output", "Output_PER", and "Output_PWM" outputs can be used concurrently.			
Output	REAL	0.0	Output value in REAL format
Output_PER	INT	0	Analog output value
Output_PWM	BOOL	FALSE	Pulse-width-modulated output value The output value is formed by by variable On and Off times.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit). The setpoint is limited to Config.SetpointUpperLimit .
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached (Setpoint ≤ Config.SetpointLowerLimit). The setpoint is limited to Config.SetpointLowerLimit .
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 2808) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Substitute output value with error monitoring
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 2812) shows which error messages are pending. ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck.

In/out parameters of PID_Compact V2 (S7-1200, S7-1500)

Table 4-60

Parameter	Data type	Default	Description
Mode	INT	4	At Mode, specify the operating mode to which PID_Compact is to switch. Options are: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode The operating mode is activated by: • Rising edge at ModeActivate • Falling edge at Reset • Falling edge at ManualEnable • Cold restart of CPU if RunModeByStartup = TRUE Mode is retentive. A detailed description of the operating modes can be found in Parameters State and Mode V2 (Page 2808).

See also

Parameters State and Mode V2 (Page 2808)

Static tags of PID_Compact V2 (S7-1200, S7-1500)

Note

Change the tags identified with (1) only in "Inactive" mode to prevent malfunction of the PID controller.

Tag	Data type	Default	Description
IntegralResetMode	INT	Up to V2.2: 1, V2.3 or higher: 4	The IntegralResetMode V2 tag (Page 2816) determines how the integral actionPIDCtrl.IntegralSum is pre-assigned when switching from "Inactive" operating mode to "Automatic mode". This setting only works for one cycle. Options are: • IntegralResetMode = 0: Smooth • IntegralResetMode = 1: Delete • IntegralResetMode = 2: Hold • IntegralResetMode = 3: Pre-assign • IntegralResetMode = 4: Like setpoint change (only for PID_Compact with version ≥ 2.3)
OverwriteInitialOutputValue	REAL	0.0	If one of the following conditions is met, the integral action PIDCtrl.IntegralSum is pre-assigned automatically as if Output = OverwriteInitialOutputValue in the previous cycle: • IntegralResetMode = 3 when switching from "Inactive" operating mode to "Automatic mode". • TRUE -> FALSE edge at parameter Reset and parameter Mode = 3 • PIDCtrl.PIDInit = TRUE in "Automatic mode" (available as of PID_Compact version 2.3)
RunModeByStartup	BOOL	TRUE	Activate operating mode at Mode parameter after CPU restart If RunModeByStartup = TRUE, PID_Compact starts in the operating mode saved in the Mode parameter after CPU startup. If RunModeByStartup = FALSE, PID_Compact remains in "Inactive" mode after CPU startup.
LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
ActivateRecoverMode	BOOL	TRUE	The Tag ActivateRecoverMode V2 (Page 2814) determines the reaction to error.
Warning	DWORD	0	Tag Warning V2 (Page 2816) shows the warnings since Reset = TRUE or ErrorAck =TRUE. Warning is retentive.
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)
CurrentSetpoint	REAL	0.0	CurrentSetpoint always displays the current setpoint. This value is frozen during tuning.

Tag	Data type	Default	Description
CancelTuningLevel	REAL	10.0	Permissible fluctuation of setpoint during tuning. Tuning is not canceled until: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel
SubstituteOutput	REAL	0.0	Substitute output value When the following conditions are met, the substitute output value is used: • An error has occurred in automatic mode. • SetSubstituteOutput = TRUE • ActivateRecoverMode = TRUE
SetSubstituteOutput	BOOL	TRUE	If SetSubstituteOutput = TRUE and ActivateRecover-Mode = TRUE, the substitute output value configured is output as long as an error is pending. If SetSubstituteOutput = FALSE and ActivateRecover-Mode = TRUE, the actuator remains at the current output value as long as an error is pending. If ActivateRecoverMode = FALSE, SetSubstituteOutput is not effective. If SubstituteOutput is invalid (ErrorBits = 20000h), the substitute output value cannot be output.
Config.InputPerOn(1)	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.InvertControl(1)	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.InputUpperLimit(1)	REAL	120.0	High limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). This pre-assignment ensures that an error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_Compact reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit(1)	REAL	0.0	Low limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. InputLowerLimit < InputUpperLimit

Tag	Data type	Default	Description
Config.InputUpperWarning(1)	REAL	3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning(1)	REAL	-3.402822e +38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit(1)	REAL	100.0	High limit of output value For details, see OutputLowerLimit OutputUpperLimit > OutputLowerLimit
Config.OutputLowerLimit(1)	REAL	0.0	Low limit of output value For Output and Output_PER, the range of values from -100.0 to +100.0, including zero, is valid. At -100.0, Output_PER = -27648; at +100.0, Output_PER = 27648. For Output_PWM, the value range 0.0 to +100.0 applies. The output value limits must match the control logic. OutputLowerLimit < OutputUpperLimit
Config.SetpointUpperLimit(1)	REAL	3.402822e+38	High limit of setpoint If you configure SetpointUpperLimit outside the process value limits, the configured process value absolute high limit is used as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit(1)	REAL	-3.402822e +38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime(1)	REAL	0.0	The minimum ON time of the pulse width modulation in seconds is rounded to MinimumOnTime = n×CycleTime.Value
Config.MinimumOffTime(1)	REAL	0.0	The minimum OFF time of the pulse width modulation in seconds is rounded to MinimumOffTime = n×CycleTime.Value

Tag	Data type	Default	Description
Config.InputScaling.UpperPointIn(1)	REAL	27648.0	Scaling Input_PER high Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.LowerPointIn(1)	REAL	0.0	Scaling Input_PER low Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.UpperPointOut(1)	REAL	100.0	Scaled high process value Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.LowerPointOut(1)	REAL	0.0	Scaled low process value Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
CycleTime.StartEstimation	BOOL	TRUE	If CycleTime.StartEstimation = TRUE, the automatic determination of the cycle time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If CycleTime.EnEstimation = TRUE, the PID_Compact sampling time is calculated. If CycleTime.EnEstimation = FALSE, the PID_Compact sampling time is not calculated and you need to correct the configuration of CycleTime.Value manually.
CycleTime.EnMonitoring	BOOL	TRUE	If CycleTime.EnMonitoring = FALSE, the PID_Compact sampling time is not monitored. If it is not possible to execute PID_Compact within the sampling time, no error (ErrorBits=0800h) is output and PID_Compact does not switch to "Inactive" mode.
CycleTime.Value(1)	REAL	0.1	PID_Compact sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time [s]
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time [s]
CtrlParamsBackUp.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	1.0	Saved proportional action weighting factor
CtrlParamsBackUp.DWeighting	REAL	1.0	Saved derivative action weighting factor
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm

Tag	Data type	Default	Description
PIDSelfTune.SUT.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If SUT.CalculateParams = TRUE, the parameters for pretuning are recalculated according to these properties. This enables you to change the parameter calculation method without having to repeat controller tuning. SUT.CalculateParams is set to FALSE after the calculation.
PIDSelfTune.SUT.TuneRule	INT	0	Methods used to calculate parameters during pretuning: • SUT.TuneRule = 0: PID according to Chien, Hrones and Reswick • SUT.TuneRule = 1: PI according to Chien, Hrones and Reswick
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning: • State = 0: Initialize pretuning • State = 100: Calculate the standard deviation • State = 200: Find the point of inflection • State = 9900: Pretuning successful • State = 1: Pretuning not successful
PIDSelfTune.TIR.RunIn	BOOL	FALSE	With the RunIn tag, you can specify that fine tuning can also be performed without pretuning. • RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If the requirements for pretuning are not met, PID_Compact reacts as if RunIn = TRUE. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_Compact switches to the mode from which tuning was started. • RunIn = TRUE The pretuning is skipped. PID_Compact tries to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. RunIn is set to FALSE after fine tuning.
PIDSelfTune.TIR.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If TIR.CalculateParams = TRUE, the parameters for fine tuning are recalculated according to these properties. This enables you to change the parameter calculation method without having to repeat controller tuning. TIR.CalculateParams is set to FALSE after the calculation.

Tag	Data type	Default	Description
PIDSelfTune.TIR.TuneRule	INT	0	Methods used to calculate parameters during fine tuning: • TIR.TuneRule = 0: PID automatic • TIR.TuneRule = 1: PID fast • TIR.TuneRule = 2: PID slow • TIR.TuneRule = 3: Ziegler-Nichols PID • TIR.TuneRule = 4: Ziegler-Nichols PI • TIR.TuneRule = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of fine tuning: • State = -100 Fine tuning is not possible. Pretuning will be performed first. • State = 0: Initialize fine tuning • State = 200: Calculate the standard deviation • State = 300: Attempt to reach setpoint • State = 400: Attempt to reach setpoint with existing PID parameters (if pretuning was successful) • State = 500: Determine oscillation and calculate parameters • State = 9900: Fine tuning successful • State = 1: Fine tuning not successful
PIDCtrl.IntegralSum(1)	REAL	0.0	Current integral action
PIDCtrl.PIDInit	BOOL	FALSE	PIDCtrl.PIDInit is available as of PID_Compact version 2.3. If PIDCtrl.PIDInit = TRUE in "Automatic mode", the integral action PIDCtrl.IntegralSum is pre-assigned automatically as if Output = OverwriteInitialOutputValue in the previous cycle. This can be used for a AUTOHOTSPOT.
Retain.CtrlParams.Gain(1)	REAL	1.0	Active proportional gain To invert the control logic, use the Config.InvertControl tag. Negative values at Gain also invert the control logic. We recommend you use only InvertControl to set the control logic. The control logic is also inverted if InvertControl = TRUE and Gain < 0.0. Gain is retentive.
Retain.CtrlParams.Ti(1)	REAL	20.0	• CtrlParams.Ti > 0.0: Active integral action time • CtrlParams.Ti = 0.0: Integral action is deactivated Ti is retentive.
Retain.CtrlParams.Td(1)	REAL	0.0	• CtrlParams.Td > 0.0: Active derivative action time • CtrlParams.Td = 0.0: Derivative action is deactivated Td is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.TdFiltRatio(1)	REAL	0.2	Active derivative delay coefficient The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient • 0.0: Derivative action is effective for one cycle only and therefore almost not effective. • 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant. • > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed. TdFiltRatio is retentive.
Retain.CtrlParams.PWeighting(1)	REAL	1.0	Active proportional action weighting The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Proportional action for setpoint change is fully effective • 0.0: Proportional action for setpoint change is not effective The proportional action is always fully effective when the process value is changed. PWeighting is retentive.
Retain.CtrlParams.DWeighting(1)	REAL	1.0	Active derivative action weighting The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Derivative action is fully effective upon setpoint change • 0.0: Derivative action is not effective upon setpoint change The derivative action is always fully effective when the process value is changed. DWeighting is retentive.
Retain.CtrlParams.Cycle(1)	REAL	1.0	Active sampling time of the PID algorithm CtrlParams.Cycle is calculated during tuning and rounded to an integer multiple of CycleTime.Value. CtrlParams.Cycle is used as time period of the pulse width modulation. Cycle is retentive.

See also

Tag ActivateRecoverMode V2 (Page 2814)

Tag Warning V2 (Page 2816)

Changing the PID_Compact V2 interface (S7-1200, S7-1500)

The following table shows what has changed in the PID_Compact instruction interface.

PID_Compact V1	PID_Compact V2	Change
Input_PER	Input_PER	Data type from Word to Int
	Disturbance	New
	ErrorAck	New
	ModeActivate	New
Output_PER	Output_PER	Data type from Word to Int
Error	ErrorBits	Renamed
	Error	New
	Mode	New
sb_RunModeByStartup	RunModeByStartup	Function
	IntegralResetMode	
	OverwriteInitialOutputValue	New
	SetSubstituteOutput	New
	CancelTuningLevel	New
	SubstituteOutput	New

The following table shows which variables have been renamed.

PID_Compact V1.x	PID_Compact V2
sb_GetCycleTime	CycleTime.StartEstimation
sb_EnCyclEstimation	CycleTime.EnEstimation
sb_EnCyclMonitoring	CycleTime.EnMonitoring
sb_RunModeByStartup	RunModeByStartup
si_Unit	PhysicalUnit
si_Type	PhysicalQuantity
sd_Warning	Warning
sBackUp.r_Gain	CtrlParamsBackUp.Gain
sBackUp.r_Ti	CtrlParamsBackUp.Ti
sBackUp.r_Td	CtrlParamsBackUp.Td
sBackUp.r_A	CtrlParamsBackUp.TdFiltRatio
sBackUp.r_B	CtrlParamsBackUp.PWeighting
sBackUp.r_C	CtrlParamsBackUp.DWeighting
sBackUp.r_Cycle	CtrlParamsBackUp.Cycle
sPid_Calc.r_Cycle	CycleTime.Value
sPid_Calc.b_RunIn	PIDSelfTune.TIR.RunIn
sPid_Calc.b_CalcParamSUT	PIDSelfTune.SUT.CalculateParams
sPid_Calc.b_CalcParamTIR	PIDSelfTune.TIR.CalculateParams
sPid_Calc.i_CtrlTypeSUT	PIDSelfTune.SUT.TuneRule
sPid_Calc.i_CtrlTypeTIR	PIDSelfTune.TIR.TuneRule
sPid_Calc.r_Progress	Progress

PID_Compact V1.x	PID_Compact V2
sPid_Cmpt.r_Sp_Hlm	Config.SetpointUpperLimit
sPid_Cmpt.r_Sp_Llm	Config.SetpointLowerLimit
sPid_Cmpt.r_Pv_Norm_IN_1	Config.InputScaling.LowerPointIn
sPid_Cmpt.r_Pv_Norm_IN_2	Config.InputScaling.UpperPointIn
sPid_Cmpt.r_Pv_Norm_OUT_1	Config.InputScaling.LowerPointOut
sPid_Cmpt.r_Pv_Norm_OUT_2	Config.InputScaling.UpperPointOut
sPid_Cmpt.r_Lmn_Hlm	Config.OutputUpperLimit
sPid_Cmpt.r_Lmn_Llm	Config.OutputLowerLimit
sPid_Cmpt.b_Input_PER_On	Config.InputPerOn
sPid_Cmpt.b_LoadBackUp	LoadBackUp
sPid_Cmpt.b_InvCtrl	Config.InvertControl
sPid_Cmpt.r_Lmn_Pwm_PPTm	Config.MinimumOnTime
sPid_Cmpt.r_Lmn_Pwm_PBTm	Config.MinimumOffTime
sPid_Cmpt.r_Pv_Hlm	Config.InputUpperLimit
sPid_Cmpt.r_Pv_Llm	Config.InputLowerLimit
sPid_Cmpt.r_Pv_HWrn	Config.InputUpperWarning
sPid_Cmpt.r_Pv_LWrn	Config.InputLowerWarning
sParamCalc.i_Event_SUT	PIDSelfTune.SUT.State
sParamCalc.i_Event_TIR	PIDSelfTune.TIR.State
sRet.i_Mode	sRet.i_Mode has been omitted. The operating mode is changed using Mode and ModeActivate.
sRet.r_Ctrl_Gain	Retain.CtrlParams.Gain
sRet.r_Ctrl_Ti	Retain.CtrlParams.Ti
sRet.r_Ctrl_Td	Retain.CtrlParams.Td
sRet.r_Ctrl_A	Retain.CtrlParams.TdFiltRatio
sRet.r_Ctrl_B	Retain.CtrlParams.PWeighting
sRet.r_Ctrl_C	Retain.CtrlParams.DWeighting
sRet.r_Ctrl_Cycle	Retain.CtrlParams.Cycle

Parameters State and Mode V2 (S7-1200, S7-1500)

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID_Compact switches to the operating mode saved in the Mode in-out parameter.

When the CPU is switched on or switches from Stop to RUN mode, PID_Compact starts in the operating mode that is saved in the Mode parameter. To leave PID_Compact in "Inactive" mode, set RunModeByStartup = FALSE.

Meaning of values

State / Mode	Description of operating mode
0	Inactive In "Inactive" operating mode, the output value 0.0 is always output, regardless of Config.OutputUpperLimit and Config.OutputLowerLimit. Pulse width modulation is off.
1	Pretuning The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning. Pretuning requirements: • Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) • ManualEnable = FALSE • Reset = FALSE • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * | Config.InputUpperLimit - Config.InputLowerLimit| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint and the process value lie within the configured limits. The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp. The controller switches to automatic mode following successful pretuning. If pretuning is unsuccessful, the switchover of the operating mode is dependent on ActivateRecoverMode. The phase of pretuning is indicated with PIDSelfTune.SUT.State.

State / Mode	Description of operating mode
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are recalculated based on the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning. PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp. Requirements for fine tuning: • No disturbances are expected. • The setpoint and the process value lie within the configured limits. • ManualEnable = FALSE • Reset = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started from: • Automatic mode (State = 3) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_Compact controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual mode (State = 4) If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. If the process value for pretuning is already too near the setpoint or PIDSelfTune.TIR.RunIn = TRUE, an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start. The controller switches to automatic mode following successful fine tuning. If fine tuning is unsuccessful, the switchover of the operating mode is dependent on ActivateRecoverMode. The "Fine tuning" phase is indicated with PIDSelfTune.TIR.State.
3	Automatic mode In automatic mode, PID_Compact corrects the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate. The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor. The ActivateRecoverMode tag is taken into consideration in automatic mode.

State / Mode	Description of operating mode
4	Manual mode In manual mode, you specify a manual output value in the ManualValue parameter. You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only. The switchover from manual mode to automatic mode is bumpless. Manual mode is also possible when an error is pending.
5	Substitute output value with error monitoring The control algorithm is deactivated. The SetSubstituteOutput tag determines which output value is output in this operating mode. • SetSubstituteOutput = FALSE: Last valid output value • SetSubstituteOutput = TRUE: Substitute output value You cannot activate this operating mode using Mode = 5. In the event of an error, it is activated instead of "Inactive" operating mode if all the following conditions are met: • Automatic mode (Mode = 3) • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode is effective. As soon as the errors are no longer pending, PID_Compact switches back to automatic mode.

ENO characteristics

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

Automatic switchover of operating mode during commissioning

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID_Compact automatically switches the operating mode in the event of an error. The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started

Cycle no.	Mode	State	Action
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. At the start of pretuning or fine tuning, PID_Compact has saved the value of State in the Mode in/out parameter. PID_Compact therefore switches to the operating mode from which tuning was started.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode.

See also

Output parameters of PID_Compact V2 (Page 2798)

Parameter ErrorBits V2 (S7-1200, S7-1500)

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0003h, for example, indicates that the errors 0001h and 0002h are pending simultaneously.

In manual mode, PID_Compact uses ManualValue as output value. The exception is Errorbits = 10000h.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0004	Error during fine tuning. Oscillation of the process value could not be maintained. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0008	Error at start of pretuning. The process value is too close to the setpoint. Start fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
0010	The setpoint was changed during tuning. You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0020	Pretuning is not permitted during fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact remains in fine tuning mode.
0080	Error during pretuning. The output value limits are not configured correctly or the actual value does not react as expected. Check whether the limits of the output value are configured correctly and match the control logic. Also make sure that the actual value does not oscillate strongly before starting pretuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0100	Error during fine tuning resulted in invalid parameters. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0400	Calculation of output value failed. Check the PID parameters. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0800	Sampling time error: PID_Compact is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter. If this error occurred during simulation with PLCSIM, see the notes under AUTOHOTSPOT.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
10000	Invalid value at ManualValue parameter. Value has an invalid number format. If ActivateRecoverMode = TRUE before an error occurred, PID_Compact uses SubstituteOutput as the output value. As soon as you specify a valid value in ManualValue, PID_Compact uses it as the output value.

ErrorBits (DW#16#...)	Description
20000	Invalid value at SubstituteOutput tag. Value has an invalid number format. PID_Compact uses the output value low limit as the output value. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_Compact switches back to automatic mode.
40000	Invalid value at Disturbance parameter. Value has an invalid number format. If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_Compact remains in automatic mode. If pretuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_Compact switches to the operating mode saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled.

Tag ActivateRecoverMode V2 (S7-1200, S7-1500)

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

Automatic mode

NOTICE
Your system may be damaged. If ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode even if there is an error and the process limit values are exceeded. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

ActivateRecoverMode	Description
FALSE	PID_Compact automatically switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_Compact switches between the calculated output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more of the following errors occur, PID_Compact stays in automatic mode: • 0001h: The "Input" parameter is outside the process value limits. • 0800h: Sampling time error • 40000h: Invalid value at parameter Disturbance. If one or more of the following errors occur, PID_Compact switches to "Substitute output value with error monitoring" mode: • 0002h: Invalid value at Input_PER parameter. • 0200h: Invalid value at Input parameter. • 0400h: Calculation of output value failed. • 1000h: Invalid value at Setpoint parameter. If the following error occurs, PID_Compact switches to "Substitute output value with error monitoring" mode and moves the actuator to Config.OutputLowerLimit: • 20000h: Invalid value at SubstituteOutput tag. Value has an invalid number format. This characteristics are independent of SetSubstituteOutput. As soon as the errors are no longer pending, PID_Compact switches back to automatic mode.

Pretuning and fine tuning

ActivateRecoverMode	Description
FALSE	PID_Compact automatically switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If the following error occurs, PID_Compact remains in the active mode: • 0020h: Pretuning is not permitted during fine tuning. The following errors are ignored: • 10000h: Invalid value at ManualValue parameter. • 20000h: Invalid value at SubstituteOutput tag. When any other error occurs, PID_Compact cancels the tuning and switches to the mode from which tuning was started.

Manual mode

ActivateRecoverMode is not effective in manual mode.

Tag Warning V2 (S7-1200, S7-1500)

If several warnings are pending simultaneously, the values of the Warning tag are displayed with binary addition. The display of warning 0003h, for example, indicates that the warnings 0001h and 0002h are pending simultaneously.

Warning (DW#16#....)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRule = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
1000	The substitute output value cannot be reached because it is outside the output value limits.

The following warnings are deleted as soon as the cause is eliminated:

- 0001h
- 0004h
- 0008h
- 0040h
- 0100h

All other warnings are cleared with a rising edge at Reset or ErrorAck.

IntegralResetMode V2 tag (S7-1200, S7-1500)

The IntegralResetMode tag determines how the integral action PIDCtrl.IntegralSum is pre-assigned:

- When switching from "Inactive" operating mode to "Automatic mode"
- With edge TRUE -> FALSE at parameter Reset and parameter Mode = 3

This setting only works for one cycle and is only effective if the integral action is activated (Retain.CtrlParams.Ti > 0.0 tag).

IntegralReset-Mode	Description
0	Smooth The value of PIDCtrl.IntegralSum is pre-assigned so that the switchover is bumpless, which means "Automatic mode" starts with the output value = 0.0 (parameter Output) and there is no jump of the output value regardless of the control deviation (setpoint – actual value).
1	Delete We recommend setting the weighting of the proportional action (Retain.CtrlParams.PWeighting) to 1.0 if this option is used. The value of PIDCtrl.IntegralSum is deleted. Any control deviation will cause a jump change of the output value. The direction of the output value jump depends on the configured weighting of the proportional action (Retain.CtrlParams.PWeighting tag) and the control deviation: • Proportional action weighting = 1.0: Output value jump and control deviation have identical signs. Example: If the actual value value is smaller than the setpoint (positive control deviation), the output value jumps to a positive value. • Proportional action weighting < 1.0: For large control deviations, the output value jump and control deviation have identical signs. Example: If the actual value is much smaller than the setpoint (positive control deviation), the output value jumps to a positive value. For small control deviations, the output value jump and control deviation have different signs. Example: If the actual value value is just below the setpoint (positive control deviation), the output value jumps to a negative value. This is usually not desirable, because it results in a temporary increase in the control deviation. The smaller the configured weighting of the proportional action, the greater the control deviation must be to receive an output value jump with identical sign. We recommend setting the weighting of the proportional action (Retain.CtrlParams.PWeighting) to 1.0 when this option is used. Otherwise, you may experience the undesirable behavior described for small control deviations. Alternatively, you can also use IntegralResetMode = 4. This option guarantees identical signs of the output value jump and control deviation independent of the configured weighting of the proportional action and the control deviation.
2	Hold The value of PIDCtrl.IntegralSum is not changed. You can define a new value using the user program.
3	Pre-assign The value of PIDCtrl.IntegralSum is automatically pre-assigned as if Output = OverwriteInitialOutputValue in the last cycle.
4	Like setpoint change (only for PID_Compact with version ≥ 2.3) The value of PIDCtrl.IntegralSum is automatically pre-assigned so that a similar output value jump results as for a PI controller in automatic mode in case of a setpoint change from the current actual value to the current setpoint. Any control deviation will cause a jump of the output value. Output value jump and control deviation have identical signs. Example: If the actual value value is smaller than the setpoint (positive control deviation), the output value jumps to a positive value. This is independent of the configured weighting of the proportional action and the control deviation.

If IntegralResetMode is assigned a value outside the valid value range, PID_Compact behaves as with the pre-assignment of IntegralResetMode:

- PID_Compact up to V2.2: IntegralResetMode = 1
- PID_Compact V2.3 and higher: IntegralResetMode = 4

All statements made above regarding the sign of the output value jump are based on a normal control logic (Config.InvertControl = FALSE tag). With an inverted control logic (Config.InvertControl = TRUE), the output value jump will have a reverse sign.

Sample program for PID_Compact (S7-1200, S7-1500)

In the following example, you are controlling temperature values with the technology object of the instruction "PID_Compact". The temperature values are simulated based on a block which simulates a delay element of the third order (PT3 element). The PID parameters of the technology object can be set automatically via the pretuning.

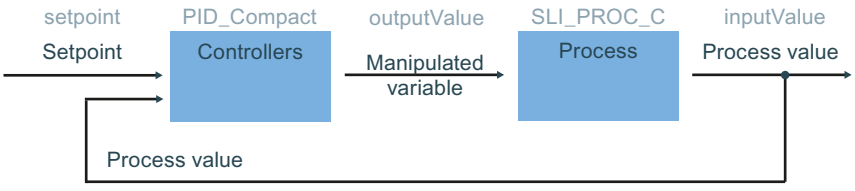

Data storage

Create seven tags in a global data block for storage of the interconnection data.

SLI_gDB_PID_Compact

		Name	Data type	Start value
1		Static		
2		setpoint	Real	75.0
3		inputValue	Real	0.0
4		outputValue	Real	0.0
5		state	Int	0
6		error	Bool	false
7		errorBits	DWord	16#0
8		resetAll	Bool	false

Interconnection of the parameters

You call the following interconnections in a cyclic interrupt OB.

Network 1: You interconnect the parameters of the instruction "PID_Compact" as follows.

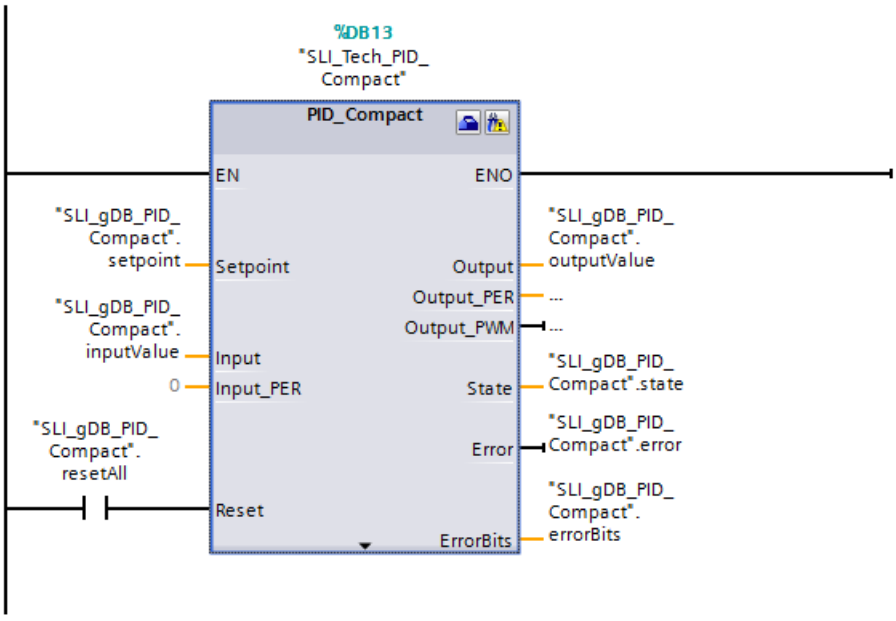

Network 2: You interconnect the parameters of the block simulating the temperature values "SLI_PROC_C" as follows.

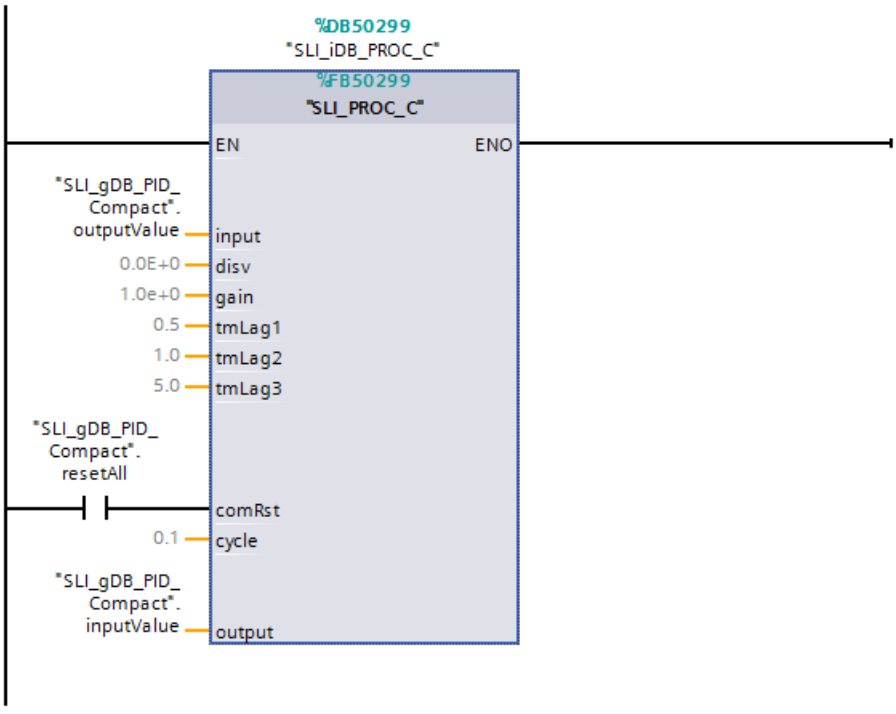

Technology object

You configure the technology object with the properties of the instruction "PID_Compact" or by using the path Technology object > Configuration. The controller type and the input/output parameters are important for the example. With the controller type, you make a preselection for the unit of the value to the controlled. In this example, "Temperature" with the unit "°C" is used as controller type. The parameters of the "PID_Compact" are already interconnected with global tags. Therefore, the information on use of the parameters Input and Output is sufficient.

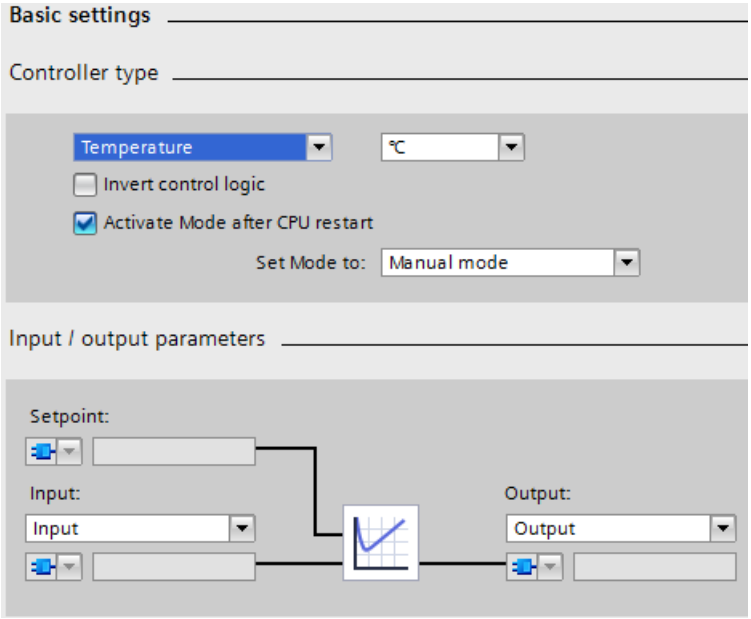

Procedure for starting the control

After the download to the CPU the PID_Compact is in manual mode with manual value 0.0. To start the control, follow these steps:

1. Open the Commissioning of the technology object "SLI_Tech_PID_Compact".
2. Click the "Start" button in the "Measurement" area.

   Measurement starts and PID_Compact can be activated.

3. Pretuning is selected.
   Click the "Start" button in the "Tuning mode" area.
   A pretuning is performed. The PID parameters are automatically adjusted to the process. After the completion of the pretuning PID_Compact switches to automatic mode.

Note

Alternative to start PID_Compact

Alternatively, you can switch PID_Compact to automatic mode in the "Online status of controller" area with the "Stop PID_Compact" / "Start PID_Compact" without pretuning. In this case the controller uses default values for the PID parameters and shows a worse controller behavior for the application case.

Procedure for stopping control

To stop and exit PID_Compact and the program, follow these steps:

1. Click the "Stop PID_Compact" button in the technology object "SLI_Tech_PID_Compact" in the "Online status of controller" area.

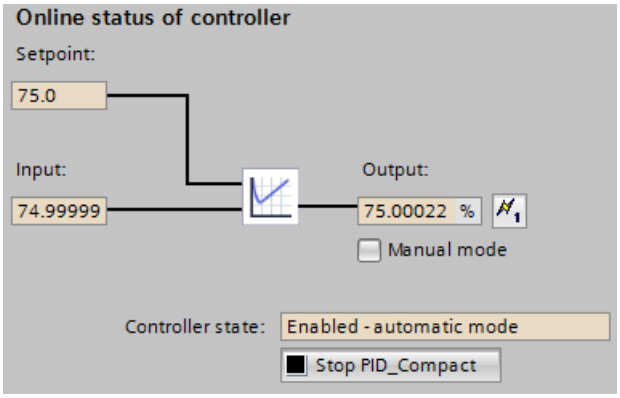

   The instruction "PID_Compact" exits the control and outputs the value "0.0" as manipulated variable.

2. Click the "Stop" button in the "Measurement" area.
3. To set the process value immediately to the value "0.0", follow these steps:
   In the block "SLI_OB_PID_Compact", set the "resetAll" tag to the value "TRUE", and then to the value "FALSE".

"PID_Compact" instruction

The setpoint for the temperature that is to be controlled is specified at the parameter Setpoint ("setpoint"). The control is started when the instruction "PID_Compact" was started with the technology object. The instruction "PID_Compact" outputs a manipulated variable at the output parameter Output ("outputValue"). The process value of the temperature is transferred to the instruction "PID_Compact" with the input parameter Input ("inputValue").

The instruction "PID_Compact" adjusts the manipulated variable ("outputValue") depending on the history of the difference between setpoint ("setpoint") and process value ("inputValue") . The process is repeated so that the process value ("inputValue") approaches the setpoint ("setpoint") through the manipulated variable ("outputValue").

The current operating mode of the instruction "PID_Compact" is displayed at the output parameter State ("state"). After pretuning (the value of "state" is "1"), the PID_Compact switches to automatic mode (the value is "3").

The output parameter Error ("error") currently shows that no error is pending. The output parameter ErrorBits ("errorBits") provides information on the error type in case of error. If an error occurs, this can be acknowledged in the technology object, in the optimization status area with the "ErrorAck" button.

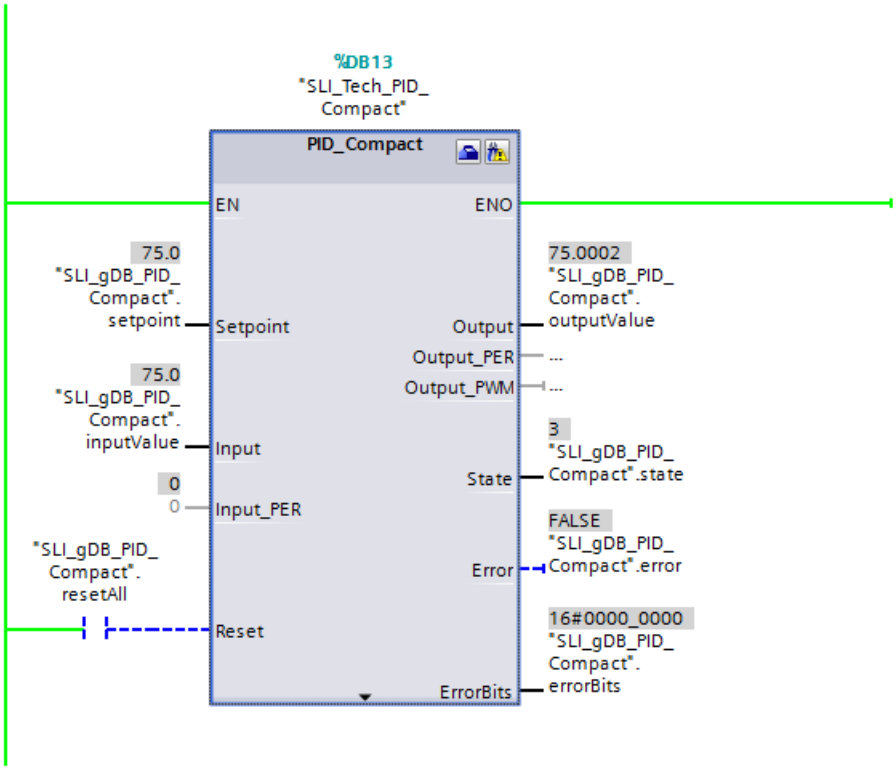

"SLI_PROC_C" block

The "SLI_PROC_C" block simulates the process value ("inputValue") of the rising temperature of a plant. The block "SLI_PROC_C" contains the manipulated variable of the controller ("outputValue) and simulates the temperature behavior of the process. This temperature is fed as process value ("inputValue") into the controller.

A change in the values of the "resetAll" tag (of the comRst parameter) has the following effects:

Parameter comRst ("resetAll")	The instruction "PID_Compact" is running	The instruction "PID_Compact" was stopped
comRst ("resetAll") remains set to the value "FALSE"	The "SLI_PROC_C" block outputs a new process value ("inputValue") based on a manipulated variable ("outputValue").	The "SLI_PROC_C" block does not receive a manipulated variable > "0.0", but it still outputs a new process value > "0.0".
comRst ("resetAll"): Change from "FALSE" to the value "TRUE"	Both manipulated variable ("outputValue") and output process value ("inputValue") are reset to "0.0".	The output process value ("inputValue") / the temperature of the "SLI_PROC_C" block is reset to "0.0".
comRst ("resetAll"): Change from "TRUE" to the value "FALSE"	Temperature control starts again.	The output process value / the temperature ("inputValue") remains "0.0".

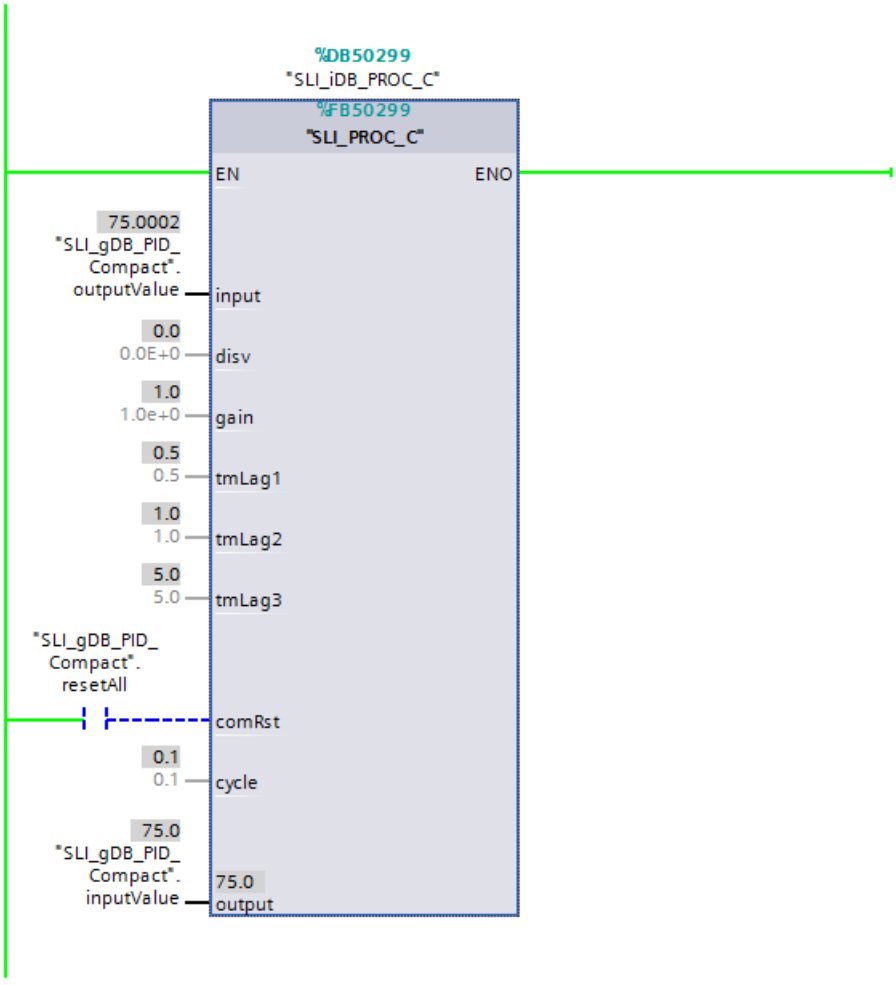

Program code

You can find additional information about the program code for the above-named example under the keyword "Sample Library for Instructions".

PID_Compact V1 (S7-1200)

Description of PID_Compact V1 (S7-1200)

Description

The PID_Compact instruction provides a PID controller with integrated tuning for automatic and manual mode.

Call

PID_Compact is called in the constant interval of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

Download to device

The actual values of retentive tags are only updated when you download PID_Compact completely.

AUTOHOTSPOT

Startup

At the startup of the CPU, PID_Compact starts in the operating mode that was last active. To retain PID_ Compact in "Inactive" mode, set sb_RunModeByStartup = FALSE.

Monitoring of the sampling time PID_Compact

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_Compact instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. If the current sampling time deviates too much from this mean value, Error = 0800 hex occurs and PID_Compact switches to "Inactive" mode.

PID_Compact, Version 1.1 or higher is set to "Inactive" mode during controller tuning under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

In automatic mode, PID_Compact, Version 1.1 or higher, is set to "Inactive" mode under the following conditions:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

During controller tuning and in automatic mode, PID_Compact 1.0 is set to "Inactive" operating mode under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

- Current sampling time >= 1.5 x current mean value
- Current sampling time <= 0.5 x current mean value

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

PID algorithm

PID_Compact is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The following equation is used to calculate the output value.

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient ($T1 = a \times T_D$)
	Derivative action time
c	Derivative action weighting

Block diagram of PID_Compact

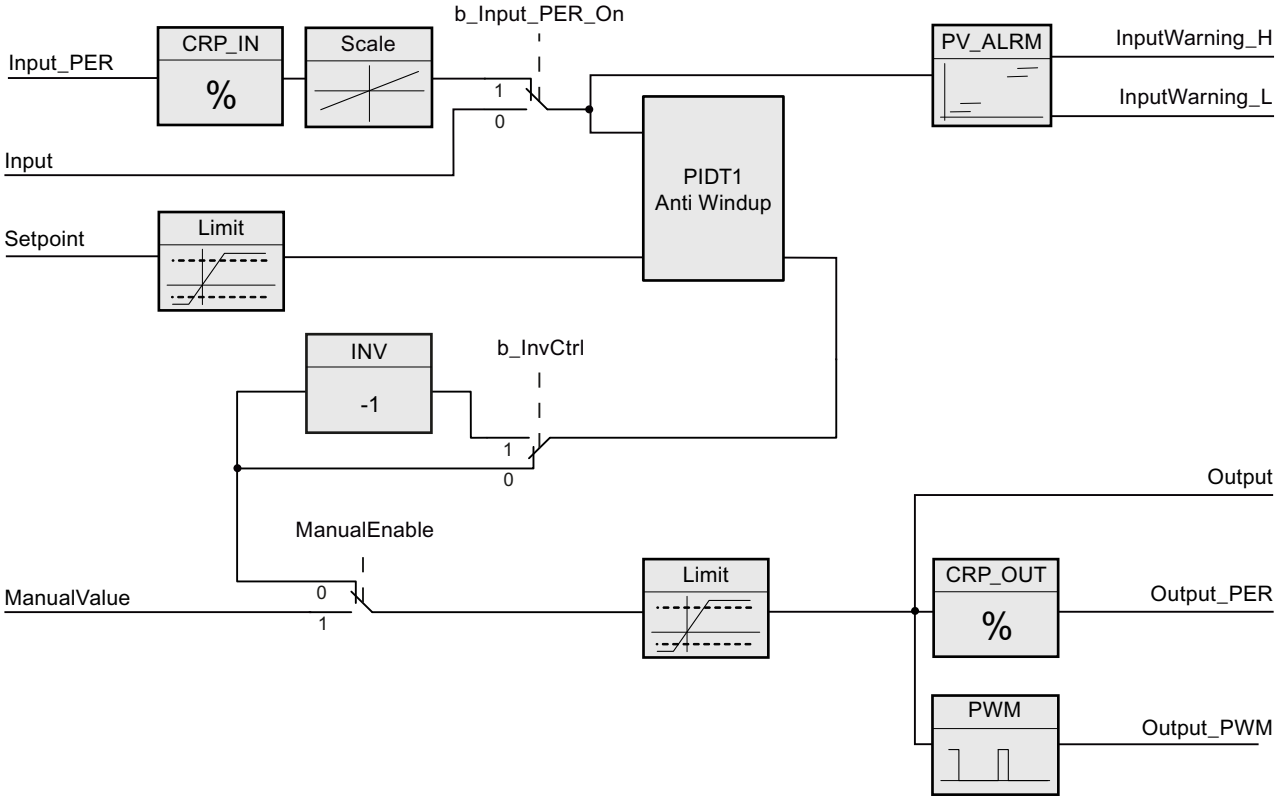

Block diagram of PIDT1 with anti-windup

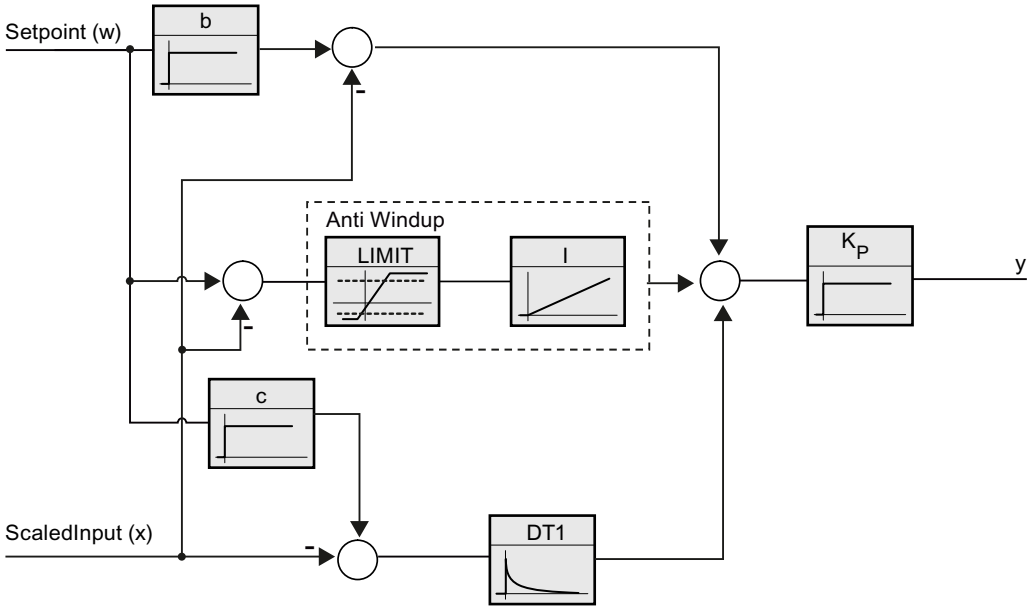

Reaction to error

If errors occur, they are output in parameter Error, and PID_Compact changes to "Inactive" mode. Reset the errors using the Reset parameter.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_Compact does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

Input parameters of PID_Compact V1 (S7-1200)

Table 4-61

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A variable of the user program is used as source for the process value. If you are using parameter Input, then sPid_Cmpt.b_Input_PER_On = FALSE must be set.
Input_PER	WORD	W#16#0	Analog input as the source of the process value If you are using parameter Input_PER, then sPid_Cmpt.b_Input_PER_On = TRUE must be set.
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge selects "Manual mode", while State = 4, sRet.i_Mode remains unchanged. • A TRUE -> FALSE edge selects the most recently active operating mode, State =sRet.i_Mode A change of sRet.i_Mode will not take effect during ManualEnable = TRUE. The change of sRet.i_Mode will only be considered upon a TRUE -> FALSE edge at ManualEnable . **PID_Compact V1.2 und PID_Compact V1.0** If at start of the CPU ManualEnable = TRUE, PID_Compact starts in manual mode. A rising edge (FALSE > TRUE) at ManualEnable is not necessary. **PID_Compact V1.1** At the start of the CPU, PID_Compact only switches to manual mode with a rising edge (FALSE->TRUE) at ManualEnable . Without rising edge, PID_Compact starts in the last operating mode in which ManualEnable was FALSE.
ManualValue	REAL	0.0	Manual value This value is used as the output value in manual mode.
Reset	BOOL	FALSE	The Reset parameter (Page 2837) restarts the controller.

Output parameters of PID_Compact V1 (S7-1200)

Table 4-62

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Output of the scaled process value
Outputs "Output", "Output_PER", and "Output_PWM" can be used concurrently.			
Output	REAL	0.0	Output value in REAL format
Output_PER	WORD	W#16#0	Analog output value
Output_PWM	BOOL	FALSE	Pulse-width-modulated output value The output value is formed by minimum On and Off times.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the setpoint absolute high limit is reached. The setpoint in the CPU is limited to the configured setpoint absolute high limit. The configured process value absolute high limit is the default for the setpoint high limit. If you set sPid_Cmpt.r_Sp_Hlm to a value within the process value limits, this value is used as the setpoint high limit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the setpoint absolute low limit has been reached. In the CPU, the setpoint is limited to the configured setpoint absolute low limit. The configured process value absolute low limit is the default setting for the setpoint low limit. If you set sPid_Cmpt.r_Sp_Llm to a value within the process value limits, this value is used as the setpoint low limit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 2833) shows the current operating mode of the PID controller. To change the operating mode, use variable sRet.i_Mode. • State = 0: Inactive • State = 1: pretuning • State = 2: fine tuning • State = 3: Automatic mode • State = 4: Manual mode
Error	DWORD	W#16#0	The Error parameter (Page 2836) indicates the error messages. Error = 0000: No error pending.

Static tags of PID_Compact V1 (S7-1200)

Note

You must not change tags that are not listed. These are used for internal purposes only.

Change the tags identified with (1) only in "Inactive" mode to prevent malfunction of the PID controller. "Inactive" mode is forced by setting the "sRet.i_Mode" tag to "0".

Table 4-63

Tag	Data type	Default	Description
sb_GetCycleTime	BOOL	TRUE	If sb_GetCycleTime = TRUE, the automatic determination of the cycle time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
sb_EnCyclEstimation	BOOL	TRUE	If sb_EnCyclEstimation = TRUE, the PID_Compact sampling time is not monitored.
sb_EnCyclMonitoring	BOOL	TRUE	If sb_EnCyclMonitoring = FALSE, the PID_Compact sampling time is not monitored. If it is not possible to execute PID_Compact within the sampling time, an 0800 error is not output and PID_Compact does not change to "Inactive" mode.
sb_RunModeByStartup	BOOL	TRUE	Activate Mode after CPU restart If sb_RunModeByStartup = FALSE, the controller will remain inactive after a CPU startup. After a CPU startup and if sb_RunModeByStartup = TRUE, the controller will return to the most recently active operating mode.
si_Unit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
si_Type	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
sd_Warning	DWORD	DW#16#0	Variable sd_warning (Page 2838) displays the warnings generated since the reset, or since the last change of the operating mode.
sBackUp.r_Gain	REAL	1.0	Saved proportional gain You can reload values from the sBackUp structure with sPid_Cmpt.b_LoadBackUp = TRUE.
sBackUp.r_Ti	REAL	20.0	Saved integral action time [s]
sBackUp.r_Td	REAL	0.0	Saved derivative action time [s]
sBackUp.r_A	REAL	0.0	Saved derivative delay coefficient
sBackUp.r_B	REAL	0.0	Saved proportional action weighting factor
sBackUp.r_C	REAL	0.0	Saved derivative action weighting factor
sBackUp.r_Cycle	REAL	1.0	Saved sampling time of PID algorithm
sPid_Calc.r_Cycle(1)	REAL	0.1	Sampling time of the PID_Compact instruction r_Cycle is determined automatically and is usually equivalent to the cycle time of the calling OB.

Tag	Data type	Default	Description
sPid_Calc.b_RunIn	BOOL	FALSE	• b_RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If the requirements for pretuning are not met, PID_Compact reacts as if b_RunIn = TRUE. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_Compact switches to "Inactive" mode. • b_RunIn = TRUE The pretuning is skipped. PID_3Compact tries to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. b_RunIn is set to FALSE after fine tuning.
sPid_Calc.b_CalcParamSUT	BOOL	FALSE	The parameters for pretuning will be recalculated if b_CalcParamSUT = TRUE. This enables you to change the parameter calculation method without having to repeat controller tuning. b_CalcParamSUT is set to FALSE after the calculation.
sPid_Calc.b_CalcParamTIR	BOOL	FALSE	The parameters for fine tuning will be recalculated if b_CalcParamTIR = TRUE. This enables you to change the parameter calculation method without having to repeat controller tuning. b_CalcParamTIR will be set to FALSE after calculation.
sPid_Calc.i_CtrlTypeSUT	INT	0	Methods used to calculate parameters during pretuning: • i_CtrlTypeSUT = 0: PID according to Chien, Hrones and Reswick • i_CtrlTypeSUT = 1: PI according to Chien, Hrones and Reswick
sPid_Calc.i_CtrlTypeTIR	INT	0	Methods used to calculate parameters during fine tuning: • i_CtrlTypeTIR = 0: PID automatic • i_CtrlTypeTIR = 1: PID fast • i_CtrlTypeTIR = 2: PID slow • i_CtrlTypeTIR = 3: Ziegler-Nichols PID • i_CtrlTypeTIR = 4: Ziegler-Nichols PI • i_CtrlTypeTIR = 5: Ziegler-Nichols P
sPid_Calc.r_Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)

Tag	Data type	Default	Description
sPid_Cmpt.r_Sp_Hlm(1)	REAL	+3.402822e +38	High limit of setpoint If you configure sPid_Cmpt.r_Sp_Hlm outside the process value limits, the configured process value absolute high limit is used as the setpoint high limit. If you set sPid_Cmpt.r_Sp_Hlm within the process value limits, this value is used as the setpoint high limit.
sPid_Cmpt.r_Sp_Llm(1)	REAL	-3.402822e +38	Low limit of the setpoint If you set sPid_Cmpt.r_Sp_Llm outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. If you set sPid_Cmpt.r_Sp_Llm within the process value limits, this value is used as the setpoint low limit.
sPid_Cmpt.r_Pv_Norm_IN_1(1)	REAL	0.0	Scaling Input_PER low Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_IN_2(1)	REAL	27648.0	Scaling Input_PER high Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_OUT_1(1)	REAL	0.0	Scaled low process value Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_OUT_2(1)	REAL	100.0	Scaled high process value Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Lmn_Hlm(1)	REAL	100.0	Output value high limit for output parameter "Output"
sPid_Cmpt.r_Lmn_Llm(1)	REAL	0.0	Low output value limit for output parameter "Output"
sPid_Cmpt.b_Input_PER_On(1)	BOOL	TRUE	If b_Input_PER_On = TRUE, then parameter Input_PER is used. If b_Input_PER_On = FALSE, then parameter Input is used.
sPid_Cmpt.b_LoadBackUp	BOOL	FALSE	Activate the back-up parameter set. If an optimization has failed, you can reactivate the previous PID parameters by setting this bit.
sPid_Cmpt.b_InvCtrl(1)	BOOL	FALSE	Invert control logic With b_InvCtrl = TRUE, an increasing control deviation causes a reduction in the output value.
sPid_Cmpt.r_Lmn_Pwm_PPTm(1)	REAL	0.0	The minimum ON time of the pulse width modulation in seconds is rounded to r_Lmn_Pwm_PPTm = r_Cycle or r_Lmn_Pwm_PPTm = n*r_Cycle

Tag	Data type	Default	Description
sPid_Cmpt.r_Lmn_Pwm_PBTm(1)	REAL	0.0	The minimum OFF time of the pulse width modulation in seconds is rounded to r_Lmn_Pwm_PBTm = r_Cycle or r_Lmn_Pwm_PBTm = n*r_Cycle
sPid_Cmpt.r_Pv_Hlm(1)	REAL	120.0	High limit of the process value At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and the PID_Compact switches to "Inactive" mode. r_Pv_Hlm > r_Pv_Llm
sPid_Cmpt.r_Pv_Llm(1)	REAL	0.0	Low limit of the process value r_Pv_Llm < r_Pv_Hlm
sPid_Cmpt.r_Pv_HWrn(1)	REAL	+3.402822e +38	Warning high limit of the process value If you set r_Pv_HWrn outside the process value limits, the configured process value absolute high limit is used as the warning high limit. If you configure r_Pv_HWrn within the process value limits, this value is used as the warning high limit. r_Pv_HWrn > r_Pv_LWrn r_Pv_HWrn ≤ r_Pv_Hlm
sPid_Cmpt.r_Pv_LWrn(1)	REAL	-3.402822e +38	Warning low limit of the process value If you set r_Pv_LWrn outside the process value limits, the configured process value absolute low limit is used as the warning low limit. If you configure r_Pv_LWrn within the process value limits, this value is used as the warning low limit. r_Pv_LWrn < r_Pv_HWrn r_Pv_LWrn ≥ r_Pv_LWrn
sPidCalc.i_Ctrl_IOutv(1)	REAL	0.0	Current integral action
sParamCalc.i_Event_SUT	INT	0	Variable i_Event_SUT (Page 2839) indicates the current phase of "pretuning":
sParamCalc.i_Event_TIR	INT	0	Variable i_Event_TIR (Page 2839) indicates the current phase of "fine tuning":
sRet.i_Mode	INT	0	The operating mode is changed edge-triggered. The following operating mode is enabled on a change to • i_Mode = 0: "Inactive" mode (controller stop) • i_Mode = 1: "Pretuning" mode • i_Mode = 2: "Fine tuning" mode • i_Mode = 3: "Automatic" mode • i_Mode = 4: "Manual" mode i_Mode is retentive.
sRet.r_Ctrl_Gain(1)	REAL	1.0	Active proportional gain Gain is retentive.

Tag	Data type	Default	Description
sRet.r_Ctrl_Ti(1)	REAL	20.0	• r_Ctrl_Ti > 0.0: Active integral action time • r_Ctrl_Ti = 0.0: Integral action is deactivated r_Ctrl_Ti is retentive.
sRet.r_Ctrl_Td(1)	REAL	0.0	• r_Ctrl_Td > 0.0: Active derivative action time • r_Ctrl_Td = 0.0: Derivative action is deactivated r_Ctrl_Td is retentive.
sRet.r_Ctrl_A(1)	REAL	0.0	Active derivative delay coefficient r_Ctrl_A is retentive.
sRet.r_Ctrl_B(1)	REAL	0.0	Active proportional action weighting r_Ctrl_B is retentive.
sRet.r_Ctrl_C(1)	REAL	0.0	Active derivative action weighting r_Ctrl_C is retentive.
sRet.r_Ctrl_Cycle(1)	REAL	1.0	Active sampling time of the PID algorithm r_Ctrl_Cycle is calculated during tuning and rounded to an integer multiple of r_Cycle. r_Ctrl_Cycle is used as time period of the pulse width modulation. r_Ctrl_Cycle is retentive.

Parameters State and sRet.i_Mode V1 (S7-1200)

Correlation of the parameters

The State parameter indicates the current operating mode of the PID controller. You cannot modify the State parameter.

You need to modify the sRet.i_Mode tag to change the operating mode. This also applies when the value for the new operating mode is already in sRet.i_Mode. First set sRet.i_Mode = 0 and then sRet.i_Mode = 3. Provided the current operating mode of the controller supports this change, State is set to the value of sRet.i_Mode.

When PID_Compact automatically switches the operating mode, the following applies: State ! = sRet.i_Mode.

Examples:

- Successful pretuning
  State = 3 and sRet.i_Mode = 1
- Error
  State = 0 and sRet.i_Mode remains at the same value, e.g sRet.i_Mode = 3
- ManualEnalbe = TRUE
  State = 4 and sRet.i_Mode remain at the previous value, for example, sRet.i_Mode = 3

Note

You wish to repeat successful fine tuning without exiting automatic mode with i_Mode = 0.

Setting sRet.i_Mode to an invalid value such as 9999 for one cycle has no effect on State. Set Mode = 2 in the next cycle. You can generate a change to sRet.i_Mode without first switching to "inactive" mode.

Meaning of values

State / sRet.i_Mode	Description of the operating mode
0	Inactive The controller is switched off. The controller was in "inactive" mode before pretuning was performed. The PID controller will change to "inactive" mode when running if an error occurs or if the "Deactivate controller" icon is clicked in the commissioning window.
1	Pretuning The pretuning determines the process response to a jump of the output value and searches for the point of inflection. The optimized PID parameters are calculated as a function of the maximum rate of rise and dead time of the controlled system. Pretuning requirements: • The controller is in inactive mode or manual mode • ManualEnable = FALSE • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * |sPid_Cmpt.r_Pv_Hlm - sPid_Cmpt.r_Pv_Llm| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint may not be changed during pretuning. The higher the stability of the process value, the easier it is to calculate the PID parameters and increase precision of the result. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. PID parameters are backed up before they are recalculated and can be reactivated with sPid_Cmpt.b_LoadBackUp. There is a change to automatic mode following successful pretuning and to "inactive" mode following unsuccessful pretuning. The phase of pretuning is indicated with Tag i_Event_SUT V1 (Page 2839).

State / sRet.i_Mode	Description of the operating mode
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized based on the amplitude and frequency of this oscillation. The differences between the process response during pretuning and fine tuning are analyzed. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning. PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. PID parameters are backed up before they are recalculated and can be reactivated with sPid_Cmpt.b_LoadBackUp. Requirements for fine tuning: • No disturbances are expected. • The setpoint and the process value lie within the configured limits. • The setpoint may not be changed during fine tuning. • ManualEnable = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started in: • Automatic mode (State = 3) Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning. PID_Comact will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual (State = 4) mode If the requirements for pretuning are met, pretuning is started. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode. An attempt is made to reach the setpoint with a minimum or maximum output value if the process value for pretuning is already too near the setpoint or sPid_Calc.b_RunIn = TRUE. This can produce increased overshoot. The controller will change to "automatic mode" after successfully completed "fine tuning" and to "inactive" mode if "fine tuning" has not been successfully completed. The "Fine tuning" phase is indicated with Tag i_Event_TIR V1 (Page 2839).

State / sRet.i_Mode	Description of the operating mode
3	Automatic mode In automatic mode, PID_Compact corrects the controlled system in accordance with the parameters specified. The controller changes to automatic mode if one the following conditions is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Change of variable sRet.i_Mode to the value 3. After CPU startup or change from Stop to RUN mode, PID_Compact will start in the most recently active operating mode. To retain PID_Compact in "Inactive" mode, set sb_RunModeByStartup = FALSE.
4	Manual mode In manual mode, you specify a manual output value in the ManualValue parameter. This operating mode is enabled if sRet.i_Mode = 4, or at the rising edge on ManualEnable. If ManualEnable changes to TRUE, only State will change. sRet.i_Mode will retain its current value. PID_Compact will return to the previous operating mode upon a falling edge at ManualEnable. The change to automatic mode is bumpless.

See also

Output parameters of PID_Compact V1 (Page 2828)

Tag i_Event_SUT V1 (Page 2839)

Tag i_Event_TIR V1 (Page 2839)

Parameter Error V1 (S7-1200)

If several errors are pending simultaneously, the values of the error codes are displayed with binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending simultaneously.

Error (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > sPid_Cmpt.r_Pv_Hlm or • Input < sPid_Cmpt.r_Pv_Llm You cannot move the actuator again until you eliminate the error.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input.
0004	Error during fine tuning. Oscillation of the process value could not be maintained.
0008	Error at start of pretuning. The process value is too close to the setpoint. Start fine tuning.
0010	The setpoint was changed during tuning.
0020	Pretuning is not permitted in automatic mode or during fine tuning.
0080	Error during pretuning. The output value limits are not configured correctly or the actual value does not react as expected. Check whether the limits of the output value are configured correctly and match the control logic. Also make sure that the actual value does not oscillate strongly before starting pretuning.

Error (DW#16#...)	Description
0100	Error during tuning resulted in invalid parameters.
0200	Invalid value at "Input" parameter: Value has an invalid number format.
0400	Calculation of output value failed. Check the PID parameters.
0800	Sampling time error: PID_Compact is not called within the sampling time of the cyclic interrupt OB. If this error occurred during simulation with PLCSIM, see the notes under AUTOHOTSPOT.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format.

See also

Output parameters of PID_Compact V1 (Page 2828)

Parameter Reset V1 (S7-1200)

The response to Reset = TRUE depends on the version of the PID_Compact instruction.

Reset response PID_Compact V.1.1 or higher

A rising edge at Reset triggers a change to "Inactive" mode; errors and warnings are reset and the integral action is deleted. A falling edge at Reset triggers a change to the most recently active operating mode. If automatic mode was active before, the integral action is pre-assigned in such a way that the switchover is bumpless.

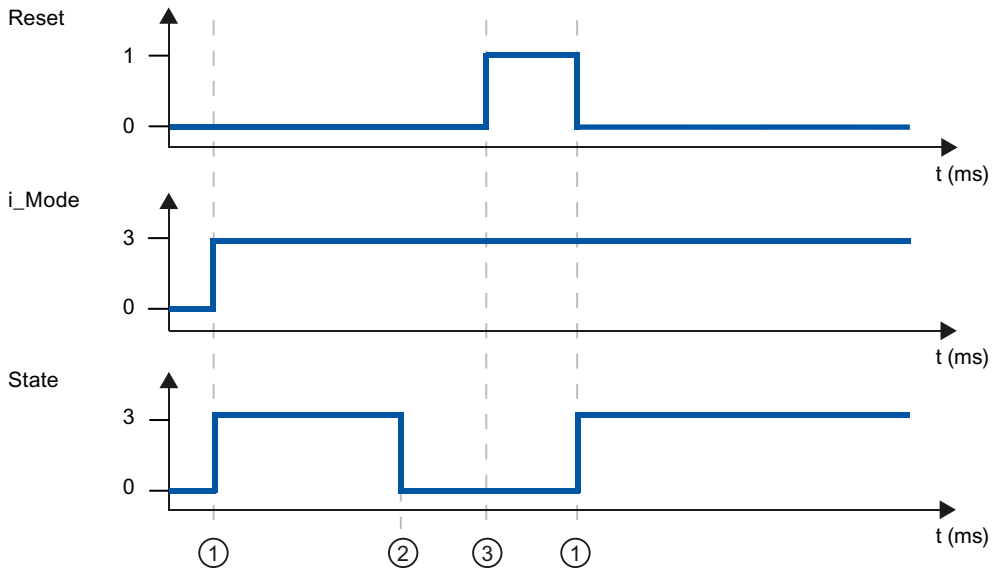

① Activation
② Error
③ Reset

Reset response PID_Compact V.1.0

A rising edge at Reset triggers a change to "Inactive" mode; errors and warnings are reset and the integral action is deleted. The controller is not reactivated until the next edge at i_Mode.

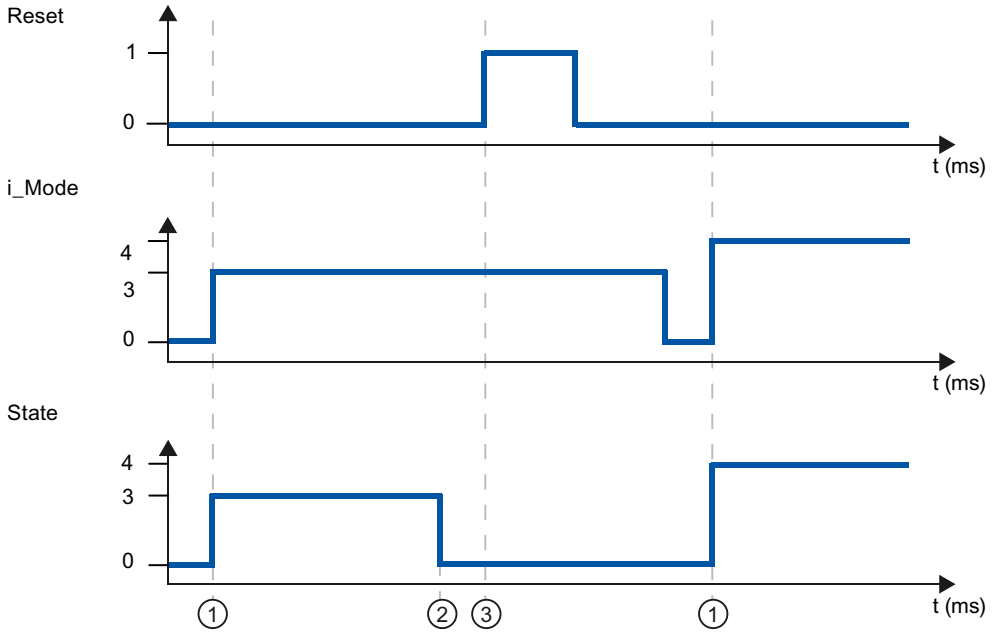

① Activation
② Error
③ Reset

Tag sd_warning V1 (S7-1200)

If several warnings are pending, the values of variable sd_warning are displayed by means of binary addition. The display of warning 0003, for example, indicates that the warnings 0001 and 0002 are also pending.

sd_warning (DW#16#....)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0002	Oscillation increased during fine tuning.
0004	The setpoint was outside the set limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. The PID parameters were instead calculated using the "i_CtrlTypeTIR = 3" method.
0010	The operating mode could not be changed because ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.

The following warnings are deleted as soon as the cause is dealt with:

- 0004
- 0020
- 0040

All other warnings are cleared with a rising edge at Reset.

Tag i_Event_SUT V1 (S7-1200)

i_Event_SUT	Name	Description
0	SUT_INIT	Initialize pretuning
100	SUT_STDABW	Calculate the standard deviation
200	SUT_GET_POI	Find the point of inflection
9900	SUT_IO	Pretuning successful
1	SUT_NIO	Pretuning not successful

See also

Static tags of PID_Compact V1 (Page 2828)

Parameters State and sRet.i_Mode V1 (Page 2833)

Tag i_Event_TIR V1 (S7-1200)

i_Event_TIR	Name	Description
-100	TIR_FIRST_SUT	Fine tuning is not possible. Pretuning will be executed first.
0	TIR_INIT	Initialize fine tuning
200	TIR_STDABW	Calculate the standard deviation
300	TIR_RUN_IN	Attempt to reach the setpoint
400	TIR_CTRLN	Attempt to reach the setpoint with the existing PID parameters (if pretuning has been successful)
500	TIR_OSZIL	Determine oscillation and calculate parameters
9900	TIR_IO	Fine tuning successful
1	TIR_NIO	Fine tuning not successful

See also

Static tags of PID_Compact V1 (Page 2828)

Parameters State and sRet.i_Mode V1 (Page 2833)

PID_3Step (S7-1200, S7-1500)

New features of PID_3Step (S7-1200, S7-1500)

PID_3Step V2.3

- As of PID_3Step Version 2.3 the monitoring and limiting of the travel time can be deactivated with Config.VirtualActuatorLimit = 0.0.

PID_3Step V2.2

- Use with S7-1200
  As of PID_3Step V2.2, the instruction with V2 functionality can also be used on S7-1200 with firmware version 4.0 or higher.

PID_3Step V2.0

- Reaction to error
  The reaction to ActivateRecoverMode = TRUE has been completely overhauled.
  PID_3Step reacts in a more fault tolerant manner in the default setting.

NOTICE
Your system may be damaged. If you use the default setting, PID_3Step remains in automatic mode even if the process value limits are exceeded. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

  You use the ErrorAck input parameter to acknowledge the errors and warnings without restarting the controller or clearing the integral action.
  Switching operating modes does not acknowledge errors that are no longer pending.
- Switching the operating mode
  You specify the operating mode at the Mode in/out parameter and use a positive edge at ModeActivate to start the operating mode. The Retain.Mode tag has been omitted.
  The transition time measurement can no longer be started with GetTransitTime.Start, but only with Mode = 6 and a positive edge at ModeActivate.
- Multi-instance capability
  You can call up PID_3Step as multi-instance DB. No technology object is created in this case and no parameter assignment interface or commissioning interface is available. You must assign parameters for PID_3Step directly in the multi-instance DB and commission it via a watch table.
- Startup characteristics
  The operating mode specified at the Mode parameter is also started on a negative edge at Reset and during a CPU cold restart, if RunModeByStartup = TRUE.

- **ENO characteristics**
  ENO is set depending on the operating mode.
  If State = 0, then ENO = FALSE.
  If State ≠ 0, then ENO = TRUE.
- **Manual mode**
  The Manual_UP and Manual_DN input parameters no longer function as edge-triggered parameters. Edge-triggered manual mode continues to be possible using the ManualUpInternal and ManualDnInternal tags.
  In "Manual mode without endstop signals" (Mode = 10), the endstop signals Actuator_H and Actuator_L are ignored even though they are activated.
- **Default value of PID parameters**
  The following default settings have been changed:
  - Proportional action weighting (PWeighting) from 0.0 to 1.0
  - Derivative action weighting (DWeighting) from 0.0 to 1.0
  - Coefficient for derivative delay (TdFiltRatio) from 0.0 to 0.2
- **Limiting of motor transition time**
  You configure the maximum percentage of the motor transition time that the actuator will travel in one direction in the Config.VirtualActuatorLimit tag.
- **Setpoint value specification during tuning**
  You configure the permitted fluctuation of the setpoint during tuning at the CancelTuningLevel tag.
- **Switching a disturbance variable on**
  You can switch a disturbance variable on at the Disturbance parameter.
- **Troubleshooting**
  If the endstop signals are not activated (ActuatorEndStopOn = FALSE), ScaledFeedback is determined without Actuator_H or Actuator_L.

PID_3Step V1.1

- **Manual mode on CPU startup**
  If ManualEnable = TRUE when the CPU starts, PID_3Step starts in manual mode. A positive edge at ManualEnable is not necessary.
- **Reaction to error**
  The ActivateRecoverMode tag is no longer effective in manual mode.
- **Troubleshooting**
  The Progress tag is reset following successful tuning or transition time measurement.

Compatibility with CPU and FW (S7-1200, S7-1500)

The following table shows which version of PID_3Step can be used on which CPU.

CPU	FW	PID_3Step
S7-1200	V4.2 or higher	V2.3 V2.2 V1.1
	V4.0 to V4.1	V2.2 V1.1
	V3.x	V1.1 V1.0
	V2.x	V1.1 V1.0
	V1.x	-
S7-1500	V2.0 or higher	V2.3 V2.2 V2.1 V2.0
	V1.5 to V1.8	V2.2 V2.1 V2.0
	V1.1	V2.1 V2.0
	V1.0	V2.0

CPU processing time and memory requirement PID_3Step V2.x (S7-1200, S7-1500)

CPU processing time

Typical CPU processing times of the PID_3Step technology object as of Version V2.0, depending on CPU type.

CPU	Typ. CPU processing time PID_3Step V2.x
CPU 1211C ≥ V4.0	410 µs
CPU 1215C ≥ V4.0	410 µs
CPU 1217C ≥ V4.0	410 µs
CPU 1505S ≥ V1.0	50 µs
CPU 1510SP-1 PN ≥ V1.6	120 µs
CPU 1511-1 PN ≥ V1.5	120 µs
CPU 1512SP-1 PN ≥ V1.6	120 µs
CPU 1516-3 PN/DP ≥ V1.5	65 µs
CPU 1518-4 PN/DP ≥ V1.5	5 µs

Memory requirement

Memory requirement of an instance DB of the PID_3Step technology object as of Version V2.0.

Memory requirement	Memory requirement of the instance DB of PID_3Step V2.x
Load memory requirement	Approx. 15000 bytes
Total work memory requirement	1040 bytes
Retentive work memory requirement	60 bytes

PID_3Step V2 (S7-1200, S7-1500)

Description of PID_3Step V2 (S7-1200, S7-1500)

Description

You use the PID_3Step instruction to configure a PID controller with self tuning for valves or actuators with integrating behavior.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Approach substitute output value
- Transition time measurement
- Error monitoring
- Approach substitute output value with error monitoring
- Manual mode without endstop signals

For a more detailed description of the operating modes, see the State parameter.

PID algorithm

PID_3Step is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation:

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Δy	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
T_D	Derivative action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
c	Derivative action weighting

Block diagram without position feedback

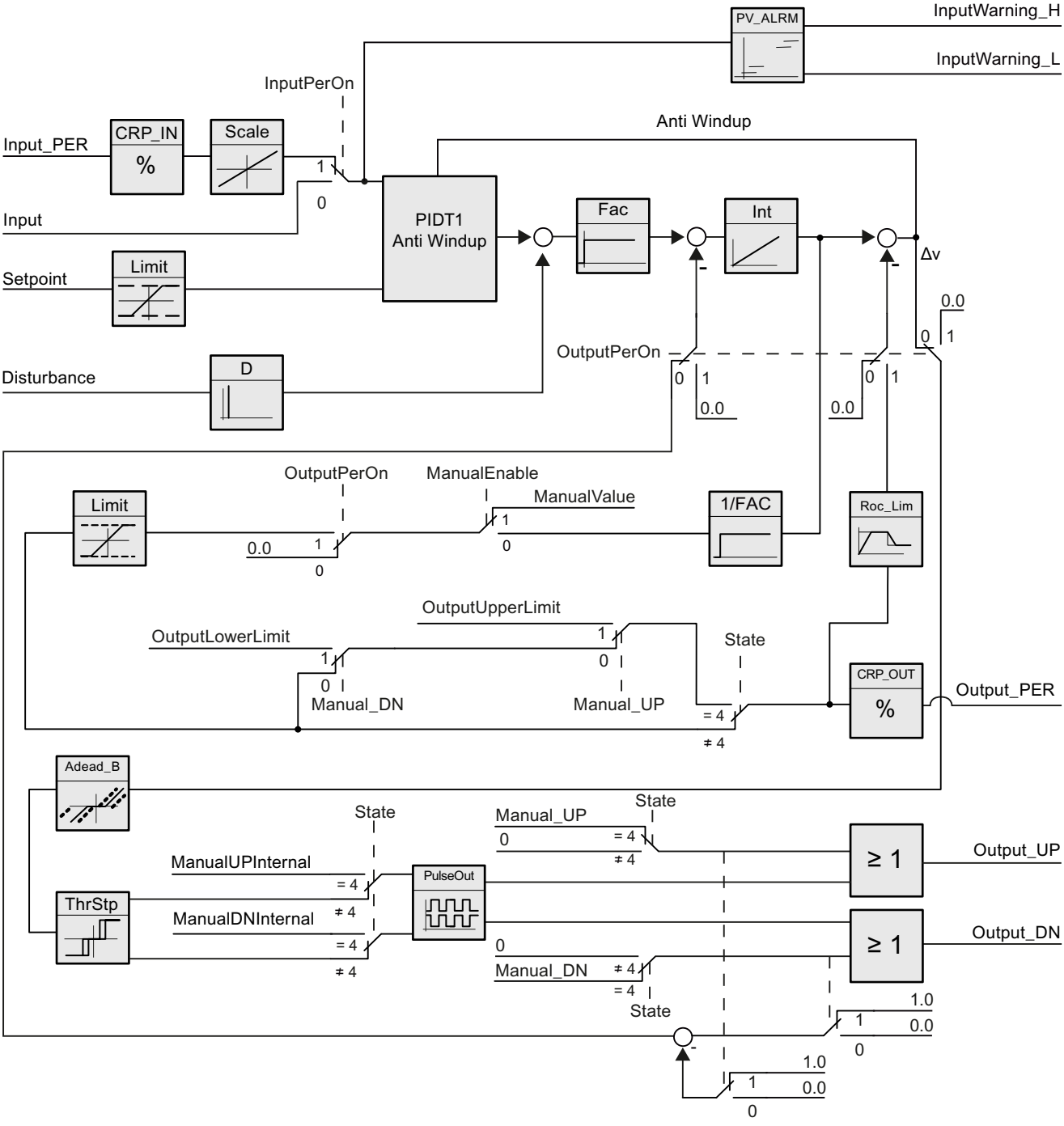

Block diagram with position feedback

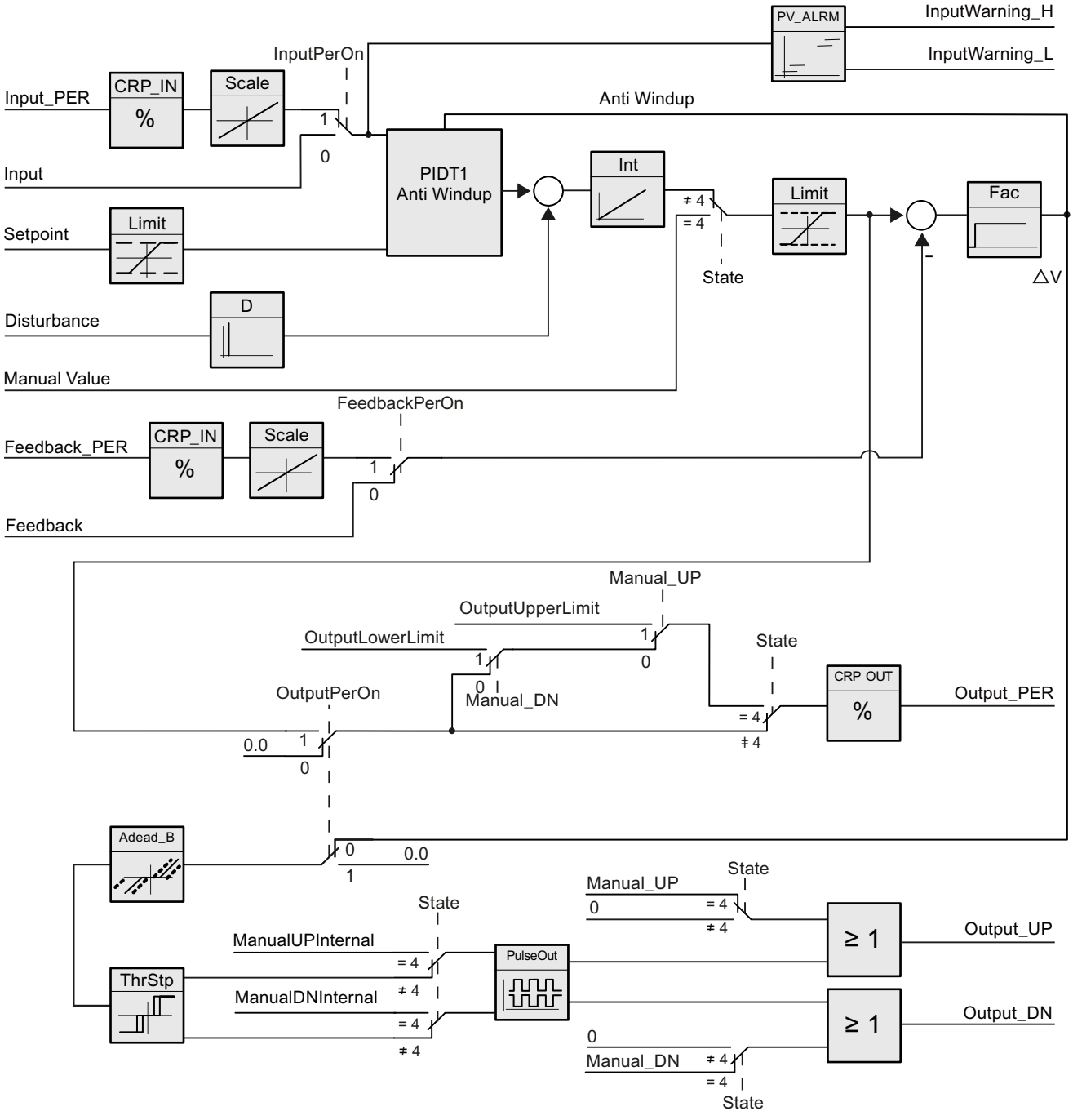

Block diagram of PIDT1 with anti-windup

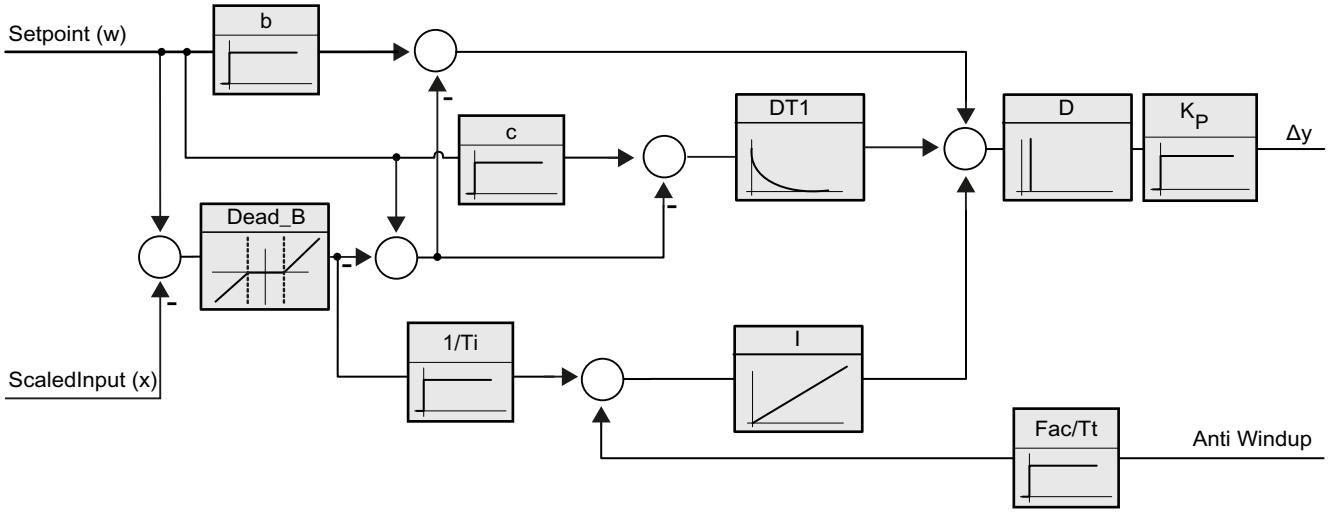

Call

PID_3Step is called in the constant time scale of a cycle interrupt OB.

If you call PID_3Step as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_3Step directly in the multi-instance DB and commission it via a watch table.

Download to device

The actual values of retentive tags are only updated when you download PID_3Step completely.

AUTOHOTSPOT

Startup

When the CPU starts up, PID_3Step starts in the operating mode that is saved in the Mode in/out parameter. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

Reaction to error

In automatic mode and during commissioning, the reaction to error depends on the ErrorBehaviour and ActivateRecoverMode tags. In manual mode, the reaction is independent of ErrorBehaviour and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the reaction additionally depends on the error that occurred.

ErrorBe-haviour	ActivateReco-verMode	Configuration editor > actuator setting > Set Output to	Reaction
FALSE	FALSE	Current output value	Switch to "Inactive" mode (State = 0) The actuator remains in the current position.
FALSE	TRUE	Current output value while error is pending	Switch to "Error monitoring" mode (State = 7) The actuator remains in the current position while the error is pending.
TRUE	FALSE	Substitute output value	Switch to "Approach substitute output value" mode (State = 5) The actuator moves to the configured substitute output value. Switch to "Inactive" mode (State = 0) The actuator remains in the current position.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Approach substitute output value with error monitoring" mode (State = 8) The actuator moves to the configured substitute output value. Switch to "Error monitoring" mode (State = 7)

In manual mode, PID_3Step uses ManualValue as output value, unless the following errors occur:

- 2000h: Invalid value at Feedback_PER parameter.
- 4000h: Invalid value at Feedback parameter.
- 8000h: Error during digital position feedback.

You can only change the position of the actuator with Manual_UP and Manual_DN, not with ManualValue.

The Error parameter indicates whether an error has occurred in this cycle. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.

See also

Parameters State and Mode V2 (Page 2864)

Parameter ErrorBits V2 (Page 2869)

Mode of operation of PID_3Step V2 (S7-1200, S7-1500)

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warning = 0040h), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID_3Step automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID_3Step checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. The output value limits must be within "Low endstop" and "High endstop".

- High endstop: Config.FeedbackScaling.UpperPointOut
- Low endstop: Config.FeedbackScaling.LowerPointOut

Rule:

UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit ≥ LowerPointOut

The valid values for "High endstop" and "Low endstop" depend upon:

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
FALSE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
FALSE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
TRUE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%

If OutputPerOn = FALSE and FeedbackOn = FALSE, you cannot limit the output value. Output_UP and Output_DN are then reset upon Actuator_H = TRUE or Actuator_L = TRUE. If endstop signals are also not present, Output_UP and Output_DN are reset after a travel time of Config.VirtualActuatorLimit × Retain.TransitTime/100. As of PID_3Step Version 2.3 the monitoring and limiting of the travel time can be deactivated with Config.VirtualActuatorLimit = 0.0.

The output value is 27648 at 100% and -27648 at -100%. PID_3Step must be able to completely close the valve.

Note

Use with two or more actuators

PID_3 Step is not suitable for use with two or more actuators (for example, in heating/cooling applications), because different actuators need different PID parameters to achieve a good control response.

Substitute output value

If an error has occurred, PID_3Step can output a substitute output value and move the actuator to a safe position that is specified in the SavePosition tag. The substitute output value must be within the output value limits.

Monitoring signal validity

The values of the following parameters are monitored for validity when used:

- Setpoint
- Input
- Input_PER
- Input_PER
- Feedback
- Feedback_PER
- Disturbance
- ManualValue
- SavePosition
- Output_PER

Monitoring the PID_3Step sampling time

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_3Step instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (ErrorBits = 0800h).

The error occurs during tuning if:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

The error occurs in automatic mode if:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID_3Step in OB1. You must then accept a lower control quality due to the deviating sampling time.

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_3Step are executed at every call.

Measuring the motor transition time

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The actuator is moved in one direction for a maximum time of Config.VirtualActuatorLimit × Retain.TransitTime/100. PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ. You can measure the motor transition time during commissioning. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior. You should therefore configure the motor transition time with the value that the motor requires to move the actuator from the closed to the opened state.

If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID_Compact instead.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_3Step does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

Changing the PID_3Step V2 interface (S7-1200, S7-1500)

The following table shows what has changed in the PID_3Step instruction interface.

PID_3Step V1	PID_3Step V2	Change
Input_PER	Input_PER	Data type from Word to Int
Feedback_PER	Feedback_PER	Data type from Word to Int
	Disturbance	New
Manual_UP	Manual_UP	Function
Manual_DN	Manual_DN	Function
	ErrorAck	New
	ModeActivate	New
Output_PER	Output_PER	Data type from Word to Int
	ManualUPInternal	New
	ManualDNInternal	New
	CancelTuningLevel	New
	VirtualActuatorLImit	New
Config.Loadbackup	Loadbackup	Renamed
Config.TransitTime	Retain.TransitTime	Renamed and retentivity added
GetTransitTime.Start		Replaced by Mode and ModeActivate
SUT.CalculateSUTPar-ams	SUT.CalculateParams	Renamed
SUT.TuneRuleSUT	SUT.TuneRule	Renamed
TIR.CalculateTIRParams	TIR.CalculateParams	Renamed
TIR.TuneRuleTIR	TIR.TuneRule	Renamed
Retain.Mode	Mode	Function Declaration of static for in-out parameters

Input parameters of PID_3Step V2 (S7-1200, S7-1500)

Table 4-64

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A tag of the user program is used as source for the process value. If you are using parameter Input, then Config.InputPer-On = FALSE must be set.
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using parameter Input_PER, then Config.InputPer-On = TRUE must be set.
Actuator_H	BOOL	FALSE	Digital position feedback of the valve for the high endstop If Actuator_H = TRUE, the valve is at the high endstop and is no longer moved towards this direction.

Parameter	Data type	Default	Description
Actuator_L	BOOL	FALSE	Digital position feedback of the valve for the low endstop If Actuator_L = TRUE, the valve is at the low endstop and is no longer moved towards this direction.
Feedback	REAL	0.0	Position feedback of the valve If you are using parameter Feedback, then Config.Feedback-PerOn = FALSE must be set.
Feedback_PER	INT	0	Analog position feedback of a valve If you are using parameter Feedback_PER, then Config.Feed-backPerOn = TRUE must be set. Feedback_PER is scaled based on the tags: • Config.FeedbackScaling.LowerPointIn • Config.FeedbackScaling.UpperPointIn • Config.FeedbackScaling.LowerPointOut • Config.FeedbackScaling.UpperPointOut
Disturbance	REAL	0.0	Disturbance variable or precontrol value
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge activates "manual mode", while State = 4, Mode remains unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog. • A TRUE -> FALSE edge activates the operating mode that is specified by Mode. We recommend that you change the operating mode using ModeActivate only.
ManualValue	REAL	0.0	In manual mode, the absolute position of the valve is specified. ManualValue is only evaluated if you are using Output_PER, **or** if position feedback is available.
Manual_UP	BOOL	FALSE	• Manual_UP = TRUE The valve is opened even if you are using Output_PER or a position feedback. The valve is no longer moved if the high endstop has been reached. See also Config.VirtualActuatorLimit • Manual_UP = FALSE If you are using Output_PER or a position feedback, the valve is moved to ManualValue. Otherwise, the valve is no longer moved. If Manual_UP and Manual_DN are set to TRUE simultaneously, the valve is not moved.
Manual_DN	BOOL	FALSE	• Manual_DN = TRUE The valve is closed even if you are using Output_PER or a position feedback. The valve is no longer moved if the low endstop has been reached. See also Config.VirtualActuatorLimit • Manual_DN = FALSE If you are using Output_PER or a position feedback, the valve is moved to ManualValue. Otherwise, the valve is no longer moved.

Parameter	Data type	Default	Description
ErrorAck	BOOL	FALSE	• FALSE -> TRUE edge ErrorBits and Warning are reset.
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Switch to "Inactive" mode – ErrorBits and Warnings are reset. • As long as Reset = TRUE, – PID_3Step remains in "Inactive" mode (State = 0). – You cannot change the operating mode with Mode and ModeActivate or ManualEnable. – You cannot use the commissioning dialog. • TRUE -> FALSE edge – If ManualEnable = FALSE, PID_3Step switches to the operating mode that is saved in Mode. – If Mode = 3, the integral action is treated as configured with the tag IntegralResetMode.
ModeActivate	BOOL	FALSE	• FALSE -> TRUE edge PID_3Step switches to the operating mode that is saved in the Mode parameter.

Output parameters of PID_3Step V2 (S7-1200, S7-1500)

Table 4-65

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
ScaledFeedback	REAL	0.0	Scaled position feedback For an actuator without position feedback, the position of the actuator indicated by ScaledFeedback is very imprecise. ScaledFeedback may only be used for rough estimation of the current position in this case.
Output_UP	BOOL	FALSE	Digital output value for opening the valve If Config.OutputPerOn = FALSE, the Output_UP parameter is used.
Output_DN	BOOL	FALSE	Digital output value for closing the valve If Config.OutputPerOn = FALSE, the Output_DN parameter is used.
Output_PER	INT	0	Analog output value If Config.OutputPerOn = TRUE, Output_PER is used. Use Output_PER if you are using a valve as actuator which is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V or 4...20 mA. The value at Output_PER corresponds to the target position of the valve, e.g. Output_PER = 13824, when the valve is to be opened by 50%.

Parameter	Data type	Default	Description
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit). The setpoint is limited to Config.SetpointUpperLimit .
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached (Setpoint ≤ Config.SetpointLowerLimit). The setpoint is limited to Config.SetpointLowerLimit .
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 2864) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Approach substitute output value • State = 6: Transition time measurement • State = 7: Error monitoring • State = 8: Approach substitute output value with error monitoring • State = 10: Manual mode without endstop signals
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 2869) shows which error messages are pending. ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck.

See also

Parameters State and Mode V2 (Page 2864)

Parameter ErrorBits V2 (Page 2869)

In/out parameters of PID-3Step V2 (S7-1200, S7-1500)

Table 4-66

Parameter	Data type	Default	Description
Mode	INT	4	At the Mode parameter, you specify the operating mode to which PID_3Step is to switch. Options are: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode • Mode = 6: Transition time measurement • Mode = 10: Manual mode without endstop signals The operating mode is activated by: • Rising edge at ModeActivate • Falling edge at Reset • Falling edge at ManualEnable • Cold restart of CPU if RunModeByStartup = TRUE Mode is retentive. A detailed description of the operating modes can be found in Parameters State and Mode V2 (Page 2864).

Static tags of PID_3Step V2 (S7-1200, S7-1500)

Note

Change the tags identified with (1) only in "Inactive" mode to prevent malfunction of the PID controller.

Tag	Data type	Default	Description
ManualUpInternal	BOOL	FALSE	In manual mode, each rising edge opens the valve by 5% of the total control range or for the duration of the minimum motor transition time. ManualUpInternal is only evaluated if you are not using Output_PER or a position feedback. This tag is used in the commissioning dialog.
ManualDnInternal	BOOL	FALSE	In manual mode, every rising edge closes the valve by 5% of the total control range or for the duration of the minimum motor transition time. ManualDnInternal is only evaluated if you are not using Output_PER or position feedback. This tag is used in the commissioning dialog.
ActivateRecoverMode	BOOL	TRUE	The ActivateRecoverMode V2 (Page 2872) tag determines the reaction to error.

Tag	Data type	Default	Description
RunModeByStartup	BOOL	TRUE	Activate operating mode at Mode parameter after CPU restart If RunModeByStartup = TRUE, PID_3Step starts in the operating mode saved in the Mode parameter after CPU startup. If RunModeByStartup = FALSE, PID_3Step remains in "Inactive" mode after CPU startup.
LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature
ErrorBehaviour	BOOL	FALSE	If ErrorBehaviour = FALSE and an error has occurred, the valve stays at its current position and the controller switches directly to "Inactive" or "Error monitoring" mode. If ErrorBehaviour = TRUE and an error occurs, the actuator moves to the substitute output value and only then switches to "Inactive" or "Error monitoring" mode. If the following errors occur, you can no longer move the valve to a configured substitute output value. • 2000h: Invalid value at Feedback_PER parameter. • 4000h: Invalid value at Feedback parameter. • 8000h: Error during digital position feedback. • 20000h: Invalid value at SavePosition tag.
Warning	DWORD	DW#16#0	The Warning tag (Page 2864) shows the warnings since Reset = TRUE or ErrorAck =TRUE. Warning is retentive. Cyclic warnings (for example, process value warning) are shown until the cause of the warning is removed. They are automatically deleted once their cause has gone. Non-cyclic warnings (for example, point of inflection not found) remain and are deleted like errors.
SavePosition	REAL	0.0	Substitute output value If ErrorBehaviour = TRUE, the actuator is moved to a position that is safe for the plant when an error occurs. As soon as the substitute output value has been reached, PID_3Step switches the operating mode according to ActivateRecoverMode.
CurrentSetpoint	REAL	0.0	Currently active setpoint. This value is frozen at the start of tuning.
CancelTuningLevel	REAL	10.0	Permissible fluctuation of setpoint during tuning. Tuning is not canceled until: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)
Config.InputPerOn(1)	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.OutputPerOn(1)	BOOL	FALSE	If OutputPerOn = TRUE, the Output_PER parameter is used. If OutputPerOn = FALSE, the Ouput_UP and Output_DN parameters are used.

Tag	Data type	Default	Description
Config.InvertControl(1)	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.FeedbackOn(1)	BOOL	FALSE	If FeedbackOn = FALSE, a position feedback is simulated. Position feedback is generally activated when Feedback-On = TRUE.
Config.FeedbackPerOn(1)	BOOL	FALSE	FeedbackPerOn is only effective when FeedbackOn = TRUE. If FeedbackPerOn = TRUE, the analog input is used for the position feedback (Feedback_PER parameter). If FeedbackPerOn = FALSE, the Feedback parameter is used for the position feedback.
Config.ActuatorEndStopOn(1)	BOOL	FALSE	If ActuatorEndStopOn = TRUE, the digital position feedback Actuator_L and Actuator_H are taken into consideration.
Config.InputUpperLimit(1)	REAL	120.0	High limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit(1)	REAL	0.0	Low limit of the process value InputLowerLimit < InputUpperLimit
Config.InputUpperWarning(1)	REAL	+3.40282 2e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning(1)	REAL	-3.40282 2e+38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit(1)	REAL	100.0	High limit of output value The following value range is permitted: UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit For more details, see OutputLowerLimit.

Tag	Data type	Default	Description
Config.OutputLowerLimit(1)	REAL	0.0	Low limit of output value The following value range is permitted: OutputUpperLimit > OutputLowerLimit ≥ LowerPointOut When using Output_PER, an output value limit of -100% corresponds to the value Output_PER = -27648; 100% correspond to the value Output_PER = 27648. If OutputPerOn = FALSE and FeedbackOn = FALSE, OutputLowerLimit and OutputUpperLimit are not evaluated. Output_UP and Output_DN are then reset at Actuator_H = TRUE or Actuator_L = TRUE (if ActuatorEndStopOn = TRUE) or after a travel time of Config.VirtualActuatorLimit * Retain.TransitTime/100 (if ActuatorEndStopOn = FALSE).
Config.SetpointUpperLimit(1)	REAL	+3.402822e+38	High limit of setpoint If you set SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is preassigned as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit(1)	REAL	-3.402822e+38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured absolute process value low limit is preassigned as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime(1)	REAL	0.0	Minimum ON time Minimum time in seconds for which the servo drive must be switched on. Config.MinimumOnTime is only effective if Output_UP and Output_DN are used (Config.OutputPerOn = FALSE).
Config.MinimumOffTime(1)	REAL	0.0	Minimum OFF time Minimum time in seconds for which the servo drive must be switched off. Config.MinimumOffTime is only effective if Output_UP and Output_DN are used (Config.OutputPerOn = FALSE).
Config.VirtualActuatorLimit(1)	REAL	150.0	If all the following conditions have been satisfied, the actuator is moved in one direction for the maximum period of VirtualActuatorLimit × Retain.TransitTime/100 and the warning 2000h is output: • Config.OutputPerOn = FALSE • Config.ActuatorEndStopOn = FALSE • Config.FeedbackOn = FALSE If Config.OutputPerOn = FALSE and Config.ActuatorEndStopOn = TRUE or Config.FeedbackOn = TRUE, only the warning 2000h is output. If Config.OutputPerOn = TRUE, VirtualActuatorLimit is not taken into consideration. As of PID_3Step Version 2.3 the monitoring and limiting of the travel time can be deactivated with Config.VirtualActuatorLimit = 0.0.

Tag	Data type	Default	Description
Config.InputScaling.UpperPointIn(1)	REAL	27648.0	Scaling Input_PER high Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointIn(1)	REAL	0.0	Scaling Input_PER low Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.UpperPointOut(1)	REAL	100.0	Scaled high process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointOut(1)	REAL	0.0	Scaled low process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.FeedbackScaling.UpperPointIn(1)	REAL	27648.0	Scaling Feedback_PER high Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointIn(1)	REAL	0.0	Scaling Feedback_PER low Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.UpperPointOut(1)	REAL	100.0	High endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure. The permitted value range is determined by the configuration. • FeedbackOn = FALSE: UpperPointOut = 100.0 • FeedbackOn = TRUE: UpperPointOut = 100.0 or 0.0 UpperPointOut ≠ LowerPointOut
Config.FeedbackScaling.LowerPointOut(1)	REAL	0.0	Low endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure. The permitted value range is determined by the configuration. • FeedbackOn = FALSE: LowerPointOut = 0.0 • FeedbackOn = TRUE: LowerPointOut = 0.0 or -100.0 LowerPointOut ≠ UpperPointOut
GetTransitTime.InvertDirection	BOOL	FALSE	If InvertDirection = FALSE, the valve is fully opened, closed, and then reopened in order to determine the valve transition time. If InvertDirection = TRUE, the valve is fully closed, opened, and then closed again.

Tag	Data type	Default	Description
GetTransitTime.SelectFeedback	BOOL	FALSE	If SelectFeedback = TRUE, then Feedback_PER, or Feedback is taken into consideration in the transition time measurement. If SelectFeedback = FALSE, then Actuator_H and Actuator_L are taken into consideration in the transition time measurement.
GetTransitTime.State	INT	0	Current phase of the transition time measurement • State = 0: Inactive • State = 1: Open valve completely • State = 2: Close valve completely • State = 3: Move valve to target position (NewOutput) • State = 4: Transition time measurement successfully completed • State = 5: Transition time measurement canceled
GetTransitTime.NewOutput	REAL	0.0	Target position for transition time measurement with position feedback The target position must be between "High endstop" and "Low endstop". The difference between NewOutput and ScaledFeedback must be at least 50% of the permissible control range.
CycleTime.StartEstimation	BOOL	TRUE	If StartEstimation = TRUE, the measurement of the PID_3Step sampling time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If EnEstimation = TRUE, the PID_3Step sampling time is calculated. If CycleTime.EnEstimation = FALSE, the PID_3Step sampling time is not calculated and you need to correct the configuration of CycleTime.Value manually.
CycleTime.EnMonitoring	BOOL	TRUE	If EnMonitoring = TRUE, the PID_3Step sampling time is monitored. If it is not possible to execute PID_3Step within the sampling time, the error 0800h is output and the operating mode is switched. ActivateRecoverMode and ErrorBehaviour determine which operating mode is switched to. If EnMonitoring = FALSE, the PID_3Step sampling time is not monitored, the error 0800h is not output, and the operating mode is not switched.
CycleTime.Value(1)	REAL	0.1	PID_3Step sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain
CtrlParamsBackUp.Ti	REAL	20.0	Saved integration time in seconds
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time in seconds
CtrlParamsBackUp.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	1.0	Saved proportional action weighting
CtrlParamsBackUp.DWeighting	REAL	1.0	Saved derivative action weighting
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm in seconds

Tag	Data type	Default	Description
CtrlParamsBackUp.Input-DeadBand	REAL	0.0	Saved dead band width of the control deviation
PIDSelfTune.SUT.Calculate-Params	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRule. CalculateParams is set to FALSE following calculation.
PIDSelfTune.SUT.TuneRule	INT	1	Methods used to calculate parameters during pretuning: • SUT.TuneRule = 0: PID fast I • SUT.TuneRule = 1: PID slow I • SUT.TuneRule = 2: Chien, Hrones and Reswick PID • SUT.TuneRule = 3: Chien, Hrones, Reswick PI • SUT.TuneRule = 4: PID fast II • SUT.TuneRule = 5: PID slow II
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning: • State = 0: Initialize pretuning • State = 50: Determine start position without position feedback • State = 100: Calculate the standard deviation • State = 200: Find the point of inflection • State = 300: Determine the rise time • State = 9900: Pretuning successful • State = 1: Pretuning not successful
PIDSelfTune.TIR.RunIn	BOOL	FALSE	With the RunIn tag, you can specify that fine tuning can also be performed without pretuning. • RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_3Step switches to the mode from which tuning was started. • RunIn = TRUE The pretuning is skipped. PID_3Step attempts to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start. RunIn is set to FALSE after fine tuning.
PIDSelfTune.TIR.Calculate-Params	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRule. CalculateParams is set to FALSE following calculation.

Tag	Data type	Default	Description
PIDSelfTune.TIR.TuneRule	INT	0	Methods used to calculate parameters during fine tuning: • TIR.TuneRule = 0: PID automatic • TIR.TuneRule = 1: PID fast • TIR.TuneRule = 2: PID slow • TIR.TuneRule = 3: Ziegler-Nichols PID • TIR.TuneRule = 4: Ziegler-Nichols PI • TIR.TuneRule = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of fine tuning: • State = -100 Fine tuning is not possible. Pretuning will be performed first. • State = 0: Initialize fine tuning • State = 200: Calculate the standard deviation • State = 300: Attempt to reach the setpoint with the maximum or minimum output value • State = 400: Attempt to reach the setpoint with existing PID parameters (if pretuning was successful) • State = 500: Determine oscillation and calculate parameters • State = 9900: Fine tuning successful • State = 1: Fine tuning not successful
Retain.TransitTime(1)	REAL	30.0	Motor transition time in seconds Time in seconds the actuating drive requires to move the valve from the closed to the opened state. TransitTime is retentive.
Retain.CtrlParams.SetByUser(1)	BOOL	FALSE	If SetByUser = FALSE, the PID parameters are determined automatically and PID_3Step operates with a dead zone at the output value. The dead band width is calculated during tuning on the basis of the standard deviation of the output value and saved in Retain.CtrlParams.OutputDeadBand. If SetByUser = TRUE, the PID parameters are entered manually and PID_3 Step operates without a dead zone at the output value. Retain.CtrlParams.OutputDeadBand = 0.0 SetByUser is retentive.
Retain.CtrlParams.Gain(1)	REAL	1.0	Active proportional gain To invert the control logic, use the Config.InvertControl tag. Negative values at Gain also invert the control logic. We recommend you use only InvertControl to set the control logic. The control logic is also inverted if InvertControl = TRUE and Gain < 0.0. Gain is retentive.
Retain.CtrlParams.Ti(1)	REAL	20.0	• Ti > 0.0: Active integration time in seconds • Ti = 0.0: Integral action is deactivated Ti is retentive.
Retain.CtrlParams.Td(1)	REAL	0.0	• Td > 0.0: Active derivative action time in seconds • Td = 0.0: Derivative action is deactivated Td is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.TdFiltRatio(1)	REAL	0.2	Active derivative delay coefficient The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient • 0.0: Derivative action is effective for one cycle only and therefore almost not effective. • 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant. • > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed. TdFiltRatio is retentive.
Retain.CtrlParams.PWeighting(1)	REAL	1.0	Active proportional action weighting The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Proportional action for setpoint change is fully effective • 0.0: Proportional action for setpoint change is not effective The proportional action is always fully effective when the process value is changed. PWeighting is retentive.
Retain.CtrlParams.DWeighting(1)	REAL	1.0	Active derivative action weighting The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Derivative action is fully effective upon setpoint change • 0.0: Derivative action is not effective upon setpoint change The derivative action is always fully effective when the process value is changed. DWeighting is retentive.
Retain.CtrlParams.Cycle(1)	REAL	1.0	Active sampling time of PID algorithm in seconds, rounded to an integer multiple of the cycle time of the calling OB. Cycle is retentive.
Retain.CtrlParams.InputDeadBand(1)	REAL	0.0	Dead band width of the control deviation InputDeadBand is retentive.

See also

Parameters State and Mode V2 (Page 2864)

Tag ActivateRecoverMode V2 (Page 2872)

Parameters State and Mode V2 (S7-1200, S7-1500)

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID_3Step switches to the operating mode saved in the Mode in-out parameter.

When the CPU is switched on or switches from Stop to RUN mode, PID_3Step starts in the operating mode that is saved in the Mode parameter. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

Meaning of values

State	Description of operating mode
0	Inactive The controller is switched off and no longer changes the valve position.
1	Pretuning The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning. Pretuning requirements: • The motor transition time has been configured or measured. • Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) • ManualEnable = FALSE • Reset = FALSE • The setpoint and the process value lie within the configured limits. The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher as compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp. The controller switches to automatic mode following successful pretuning. If pretuning is unsuccessful, the switchover of operating mode is dependent on ActivateRecoverMode and ErrorBehaviour. The pretuning phase is indicated with the SUT.State tag.

State	Description of operating mode
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are recalculated based on the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning. PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel The PID parameters are backed up before fine tuning. They can be reactivated with LoadBackUp. Requirements for fine tuning: • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. • ManualEnable = FALSE • Reset = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started from: • Automatic mode (State = 3) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual mode (State = 4) If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. If PIDSelfTune.TIR.RunIn = TRUE, pretuning is skipped and an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. The controller switches to automatic mode following successful fine tuning. If fine tuning is unsuccessful, the switchover of operating mode is dependent on ActivateRecoverMode and ErrorBehaviour. The fine tuning phase is indicated with the TIR.State tag.
3	Automatic mode In automatic mode, PID_3Step controls the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate. The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor. The ActivateRecoverMode tag is taken into consideration in automatic mode.

State	Description of operating mode
4	Manual mode In manual mode, you specify manual output values in the Manual_UP and Manual_DN parameters or ManualValue parameter. Whether or not the actuator can be moved to the output value in the event of an error is described in the ErrorBits parameter. You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only. The switchover from manual mode to automatic mode is bumpless. Manual mode is also possible when an error is pending.
5	Approach substitute output value This operating mode is activated in the event of an error, if Errorbehaviour = TRUE and ActivateRecoverMode = FALSE.. PID_3Step moves the actuator to the substitute output value and then switches to "Inactive" mode.
6	Transition time measurement The time that the motor needs to completely open the valve from the closed condition is determined. This operating mode is activated when Mode = 6 and ModeActivate = TRUE is set. If endstop signals are used to measure the transition time, the valve will be opened completely from its current position, closed completely, and opened completely again. If GetTransitTime.InvertDirection = TRUE, this behavior is inverted. If position feedback is used to measure the transition time, the actuator will be moved from its current position to a target position. The output value limits are not taken into consideration during the transition time measurement. The actuator can travel to the high or the low endstop.
7	Error monitoring The control algorithm is switched off and no longer changes the valve position. This operating mode is activated instead of "Inactive" mode in the event of an error. All the following conditions must be met: • Automatic mode (Mode = 3) • Errorbehaviour = FALSE • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 2872) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.
8	Approach substitute output value with error monitoring This operating mode is activated instead of "approach substitute output value" mode when an error occurs. PID_3Step moves the actuator to the substitute output value and then switches to "error monitoring" mode. All the following conditions must be met: • Automatic mode (Mode = 3) • Errorbehaviour = TRUE • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 2872) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.
10	Manual mode without endstop signals The endstop signals are **not** taken into consideration, even though Config.ActuatorEndStopOn = TRUE. The output value limits are not taken into consideration. Otherwise, PID_3Step behaves the same as in manual mode.

ENO characteristics

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

Automatic switchover of operating mode during commissioning

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID_3Step automatically switches the operating mode in the event of an error. The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. At the start of transition time measurement, pretuning, or fine tuning, PID_3Step saved the value of State in the Mode in/out parameter. PID_3Step therefore switches to the operating mode from which transition time measurement or tuning was started.

If ActivateRecoverMode = FALSE, "Inactive" or "Approach substitute output value" mode is activated.

Automatic switchover of operating mode after transition time measurement

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated after successful transition time measurement.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode after successful transition time measurement.

Automatic switchover of operating mode in automatic mode

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.

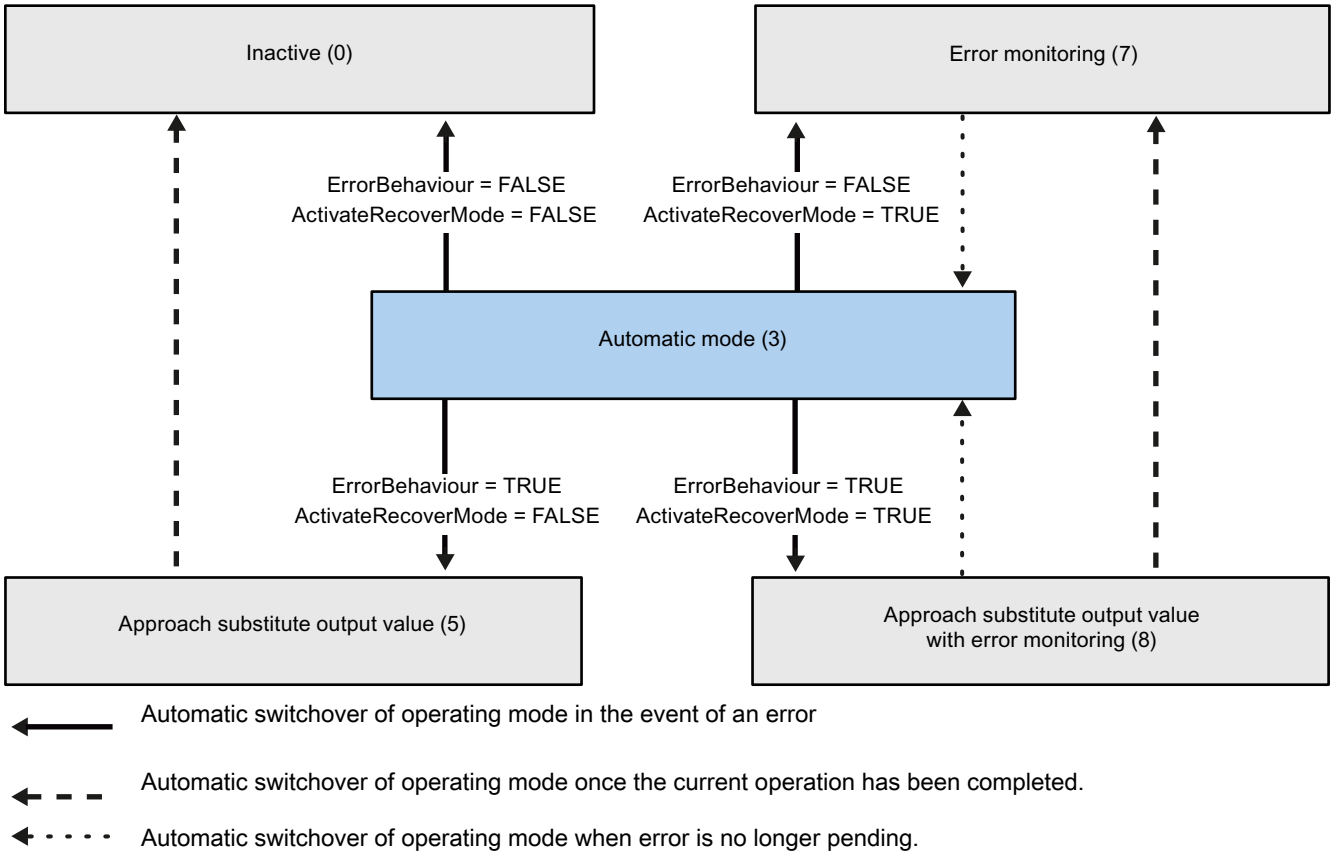

See also

Tag ActivateRecoverMode V2 (Page 2872)

Parameter ErrorBits V2 (Page 2869)

Parameter ErrorBits V2 (S7-1200, S7-1500)

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0003h, for example, indicates that the errors 0001h and 0002h are pending simultaneously.

If there is a position feedback, PID_3Step uses ManualValue as output value in manual mode. The exception is Errorbits = 10000h.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
0004	Error during fine tuning. Oscillation of the process value could not be maintained. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0010	The setpoint was changed during tuning. You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0020	Pretuning is not permitted during fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step remains in fine tuning mode.
0080	Error during pretuning. The output value limits are not configured correctly or the actual value does not react as expected. Check whether the limits of the output value are configured correctly and match the control logic. Also make sure that the actual value does not oscillate strongly before starting pretuning. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0100	Error during fine tuning resulted in invalid parameters. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
0400	Calculation of output value failed. Check the PID parameters. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
0800	Sampling time error: PID_3Step is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter. If this error occurred during simulation with PLCSIM, see the notes under AUTOHOTSPOT.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
2000	Invalid value at Feedback_PER parameter. Check whether an error is pending at the analog input. The actuator cannot be moved to the substitute output value and remains in its current position. In manual mode, you can change the position of the actuator only with Manual_UP and Manual_DN, and not with ManualValue. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
4000	Invalid value at Feedback parameter. Value has an invalid number format. The actuator cannot be moved to the substitute output value and remains in its current position. In manual mode, you can change the position of the actuator only with Manual_UP and Manual_DN, and not with ManualValue. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
8000	Error during digital position feedback. Actuator_H = TRUE and Actuator_L = TRUE. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. In order to move the actuator from this state, you must deactivate the "Actuator endstop" (Config.ActuatorEndStopOn = FALSE) or switch to manual mode without endstop signals (Mode = 10). If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
10000	Invalid value at ManualValue parameter. Value has an invalid number format. The actuator cannot be moved to the manual value and remains in its current position. Specify a valid value in ManualValue or move the actuator in manual mode with Manual_UP and Manual_DN.
20000	Invalid value at SavePosition tag. Value has an invalid number format. The actuator cannot be moved to the substitute output value and remains in its current position.
40000	Invalid value at Disturbance parameter. Value has an invalid number format. If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_3Step remains in automatic mode. If pretuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_3Step switches to the operating mode saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled. The error has no effect during transition time measurement.

Tag ActivateRecoverMode V2 (S7-1200, S7-1500)

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

NOTICE
Your system may be damaged. If ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode even if the process limit values are exceeded. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

Automatic mode

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" mode. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_3Step switches between the calculated output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more of the following errors occur, PID_3Step stays in automatic mode: • 0001h: The "Input" parameter is outside the process value limits. • 0800h: Sampling time error • 40000h: Invalid value at Disturbance parameter. If one or more of the following errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at Input_PER parameter. • 0200h: Invalid value at Input parameter. • 0400h: Calculation of output value failed. • 1000h: Invalid value at Setpoint parameter. If one or more of the following errors occur, PID_3Step can no longer move the actuator: • 2000h: Invalid value at Feedback_PER parameter. • 4000h: Invalid value at Feedback parameter. • 8000h: Error during digital position feedback. • 20000h: Invalid value at SavePosition tag. Value has an invalid number format. The characteristics are independent of ErrorBehaviour. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

Pretuning, fine tuning, and transition time measurement

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" mode. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate. The controller changes to "Inactive" mode after successful transition time measurement.
TRUE	If the following error occurs, PID_3Step remains in the active mode: • 0020h: Pretuning is not permitted during fine tuning. The following errors are ignored: • 10000h: Invalid value at ManualValue parameter. • 20000h: Invalid value at SavePosition tag. When any other error occurs, PID_3Step cancels the tuning and switches to the mode from which tuning was started.

Manual mode

ActivateRecoverMode is not effective in manual mode.

See also

Static tags of PID_3Step V2 (Page 2856)

Parameters State and Mode V2 (Page 2864)

Tag Warning V2 (S7-1200, S7-1500)

If several warnings are pending simultaneously, their values are displayed with binary addition. The display of warning 0005h, for example, indicates that the warnings 0001h and 0004h are pending simultaneously.

Warning (DW#16#...)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRule = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
0400	The transition time cannot be measured because the actuator settings do not match the selected measuring method.
0800	The difference between the current position and the new output value is too small for transition time measurement. This can produce incorrect results. The difference between the current output value and new output value must be at least 50% of the entire control range.
1000	The substitute output value cannot be reached because it is outside the output value limits.
2000	The actuator was moved in one direction for longer than Config.VirtualActuatorLimit × Retain.TransitTime. Check whether the actuator has reached an endstop signal.

The following warnings are deleted as soon as the cause is eliminated:

- 0001h
- 0004h
- 0008h
- 0040h
- 0100h
- 2000h

All other warnings are cleared with a rising edge at Reset or ErrorAck.

PID_3Step V1 (S7-1200)

Description PID_3Step V1 (S7-1200)

Description

You use the PID_3Step instruction to configure a PID controller with self tuning for valves or actuators with integrating behavior.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Approach substitute output value
- Transition time measurement
- Approach substitute output value with error monitoring
- Error monitoring

For a more detailed description of the operating modes, see the State parameter.

PID algorithm

PID_3Step is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The following equation is used to calculate the output value.

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right]$$

Δy	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient ($T1 = a \times T_D$)
T_D	Derivative action time
c	Derivative action weighting

Block diagram without position feedback

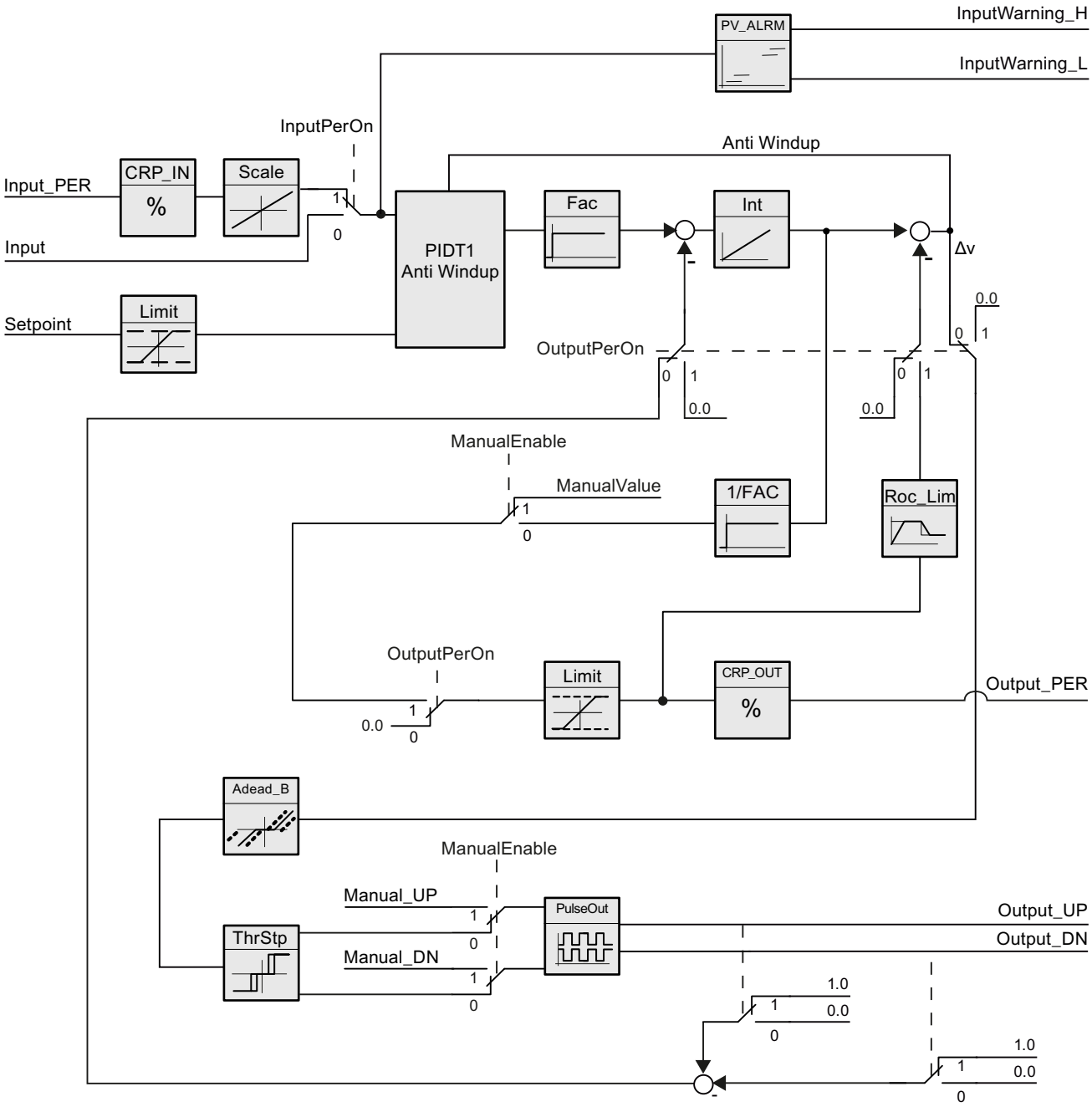

Block diagram with position feedback

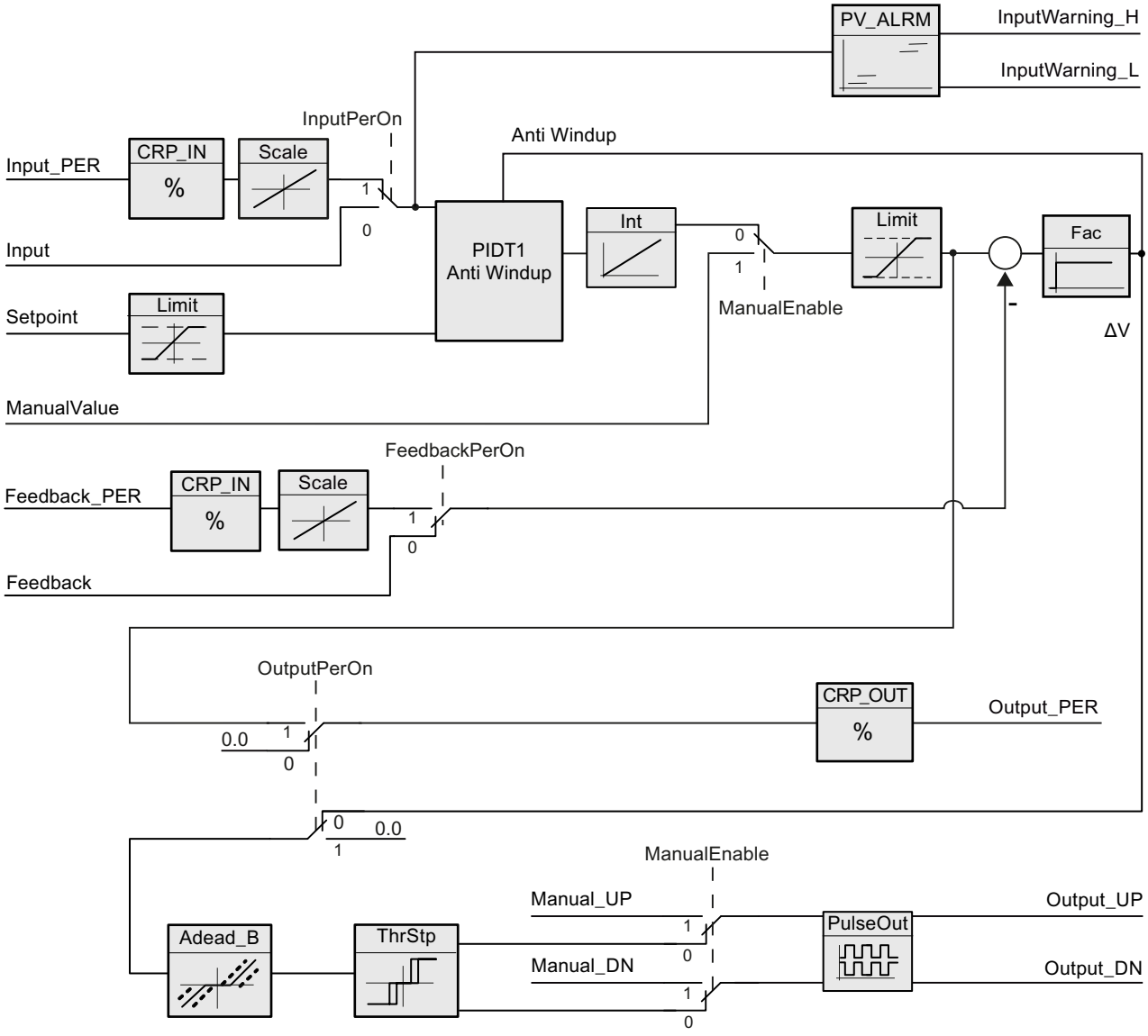

Block diagram of PIDT1 with anti-windup

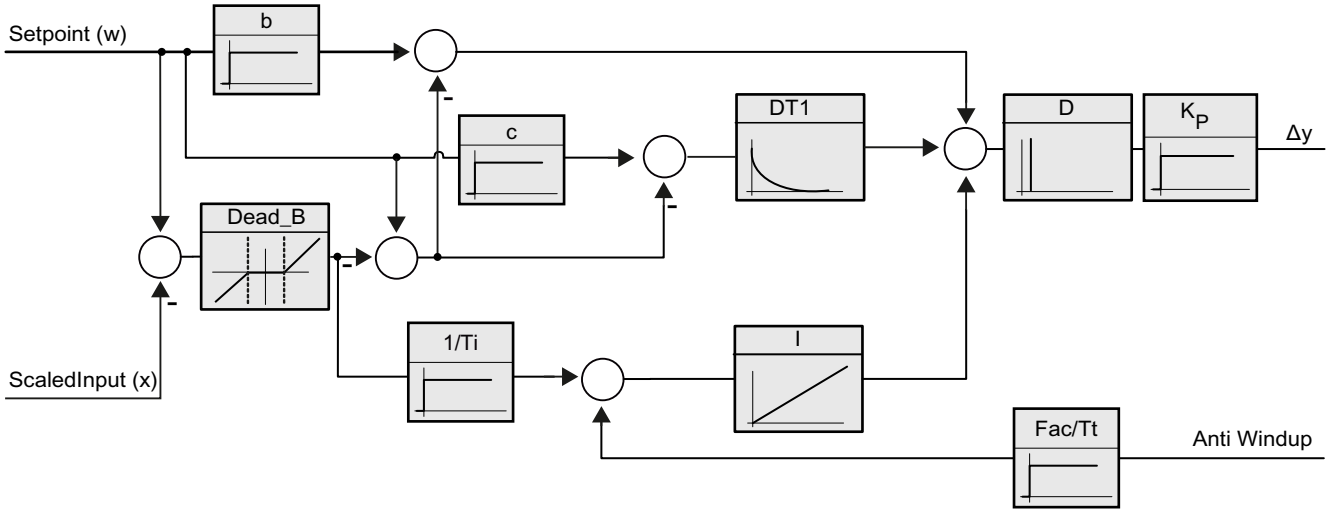

Call

PID_3Step is called in a constant time interval of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

Download to device

The actual values of retentive tags are only updated when you download PID_3Step completely.

AUTOHOTSPOT

Startup

At the startup of the CPU, PID_3Step starts in the operating mode that was last active. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

Reaction to error

If errors occur, these are output in the Error parameter. You configure the reaction of PID_3Step using the ErrorBehaviour and ActivateRecoverMode tags.

ErrorBe-haviour	ActivateReco-verMode	Actuator setting configuration Set Output to	Reaction
0	FALSE	Current output value	Switch to "Inactive" mode (Mode = 0)
0	TRUE	Current output value while er-ror is pending	Switch to "Error monitoring" mode (Mode = 7)

ErrorBe-haviour	ActivateReco-verMode	Actuator setting configuration Set Output to	Reaction
1	FALSE	Substitute output value	Switch to "Approach substitute output value" mode (Mode = 5) Switch to "Inactive" mode (Mode = 0)
1	TRUE	Substitute output value while error is pending	Switch to "Approach substitute output value with error monitoring" mode (Mode = 8) Switch to "Error monitoring" mode (Mode = 7)

The ErrorBits parameter shows which errors have occurred.

See also

Parameter State and Retain.Mode V1 (Page 2891)

Parameter ErrorBits V1 (Page 2898)

Operating principle PID_3Step V1 (S7-1200)

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0001hex).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warnings = 0040hex), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID_3Step automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID_3Step checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. The output value limits must be within "Low endstop" and "High endstop".

- High endstop: Config.FeedbackScaling.UpperPointOut
- Low endstop: Config.FeedbackScaling.LowerPointOut

Rule:

UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit ≥ LowerPointOut

The valid values for "High endstop" and "Low endstop" depend upon:

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
FALSE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
FALSE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	FALSE	FALSE	Cannot be set (100.0%)	Cannot be set (100.0%)
TRUE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%

If OutputPerOn = FALSE and FeedbackOn = FALSE, you cannot limit the output value. The digital outputs are reset with Actuator_H = TRUE or Actuator_L = TRUE, or after a travel time amounting to 110% of the motor transition time.

The output value is 27648 at 100% and -27648 at -100%. PID_3Step must be able to completely close the valve. Therefore, zero must be included in the output value limits.

Note

Use with two or more actuators

PID_3 Step is not suitable for use with two or more actuators (for example, in heating/cooling applications), because different actuators need different PID parameters to achieve a good control response.

Substitute output value

If an error has occurred, PID_3Step can output a substitute output value and move the actuator to a safe position that is specified in the SavePosition tag. The substitute output value must be within the output value limits.

Monitoring signal validity

The values of the following parameters are monitored for validity:

- Setpoint
- Input
- Input_PER
- Feedback
- Feedback_PER
- Output

Monitoring the PID_3Step sampling time

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_3Step instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (ErrorBits = 0800 hex).

PID_3Step is set to "Inactive" mode during tuning under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

In automatic mode, PID_3Step is set to "Inactive" mode under the following conditions:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_3Step are executed at every call.

Measuring the motor transition time

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The maximum time that the actuator is moved in one direction is 110% of the motor transition time. PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ. You can measure the motor transition time during commissioning. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior. You should therefore configure the motor transition time with the value that the motor requires to move the actuator from the closed to the opened state.

If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID_Compact instead.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_3Step does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

PID_3Step V1 input parameters (S7-1200)

Table 4-67

Parameters	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A tag of the user program is used as source for the process value. If you are using parameter Input, then Config.InputPer-On = FALSE must be set.
Input_PER	WORD	W#16#0	An analog input is used as the source of the process value. If you are using parameter Input_PER, then Config.InputPer-On = TRUE must be set.
Actuator_H	BOOL	FALSE	Digital position feedback of the valve for the high endstop If Actuator_H = TRUE, the valve is at the high endstop and is no longer moved towards this direction.
Actuator_L	BOOL	FALSE	Digital position feedback of the valve for the low endstop If Actuator_L = TRUE, the valve is at the low endstop and is no longer moved towards this direction.
Feedback	REAL	0.0	Position feedback of the valve If you are using parameter Feedback, then Config.Feedback-PerOn = FALSE must be set.
Feedback_PER	WORD	W#16#0	Analog position feedback of a valve If you are using parameter Feedback_PER, then Config.Feed-backPerOn = TRUE must be set. Feedback_PER is scaled based on the tags: • Config.FeedbackScaling.LowerPointIn • Config.FeedbackScaling.UpperPointIn • Config.FeedbackScaling.LowerPointOut • Config.FeedbackScaling.UpperPointOut
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge selects "Manual mode", while State = 4, Retain.Mode remains unchanged. • A TRUE -> FALSE edge selects the most recently active operating mode A change of Retain.Mode will not take effect during ManualEn-able = TRUE. The change of Retain.Mode will only be consid-ered upon a TRUE -> FALSE edge at ManualEnable . **PID_3Step V1.1**If ManualEnable = TRUE when the CPU starts, PID_3Step starts in manual mode. A rising edge (FALSE > TRUE) at ManualEnable is not necessary. **PID_3Step V1.0** At the start of the CPU, PID_3Step only switches to manual mode with a rising edge (FALSE->TRUE) at ManualEnable . Without rising edge, PID_3Step starts in the last operating mode in which ManualEnable was FALSE.
ManualValue	REAL	0.0	In manual mode, you specify the absolute position of the valve. ManualValue will only be evaluated if you are using OutputPer, **or** if position feedback is available.

Parameters	Data type	Default	Description
Manual_UP	BOOL	FALSE	In manual mode, every rising edge opens the valve by 5% of the total control range, or for the duration of the minimum motor transition time. Manual_UP is evaluated only if you are not using Output_PER and there is no position feedback available.
Manual_DN	BOOL	FALSE	In manual mode, every rising edge closes the valve by 5% of the total control range, or for the duration of the minimum motor transition time. Manual_DN is evaluated only if you are not using Output_PER and there is no position feedback available.
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Switch to "Inactive" mode – ErrorBits and Warning are reset – Intermediate controller values are reset (PID parameters are retained) • TRUE -> FALSE edge – Change in most recent active mode – If automatic mode was active before, switchover to automatic mode is bumpless.

PID_3Step V1 output parameters (S7-1200)

Table 4-68

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
ScaledFeedback	REAL	0.0	Scaled position feedback For an actuator without position feedback, the position of the actuator indicated by ScaledFeedback is very imprecise. ScaledFeedback may only be used for rough estimation of the current position in this case.
Output_UP	BOOL	FALSE	Digital output value for opening the valve If Config.OutputPerOn = FALSE, the Output_UP parameter is used.
Output_DN	BOOL	FALSE	Digital output value for closing the valve If Config.OutputPerOn = FALSE, the Output_DN parameter is used.
Output_PER	WORD	W#16#0	Analog output value If Config.OutputPerOn = TRUE, Output_PER is used. Use Output_PER if you are using a valve as actuator which is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V or 4...20 mA. The value at Output_PER corresponds to the target position of the valve, e.g. Output_PER = 13824, when the valve is to be opened by 50%.

Parameter	Data type	Default	Description
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached. In the CPU, the setpoint is limited to the configured absolute setpoint high limit. The configured absolute process value high limit is the default for the setpoint high limit. If you configure Config.SetpointUpperLimit to a value within the process value limits, this value is used as the setpoint high limit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached. In the CPU, the setpoint is limited to the configured absolute setpoint low limit. The configured absolute process value low limit is the default setting for the setpoint low limit. If you configure Config.SetpointLowerLimit to a value within the process value limits, this value is used as the setpoint low limit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 2891) shows the current operating mode of the PID controller. You change the operating mode with the Retain.Mode tag. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Approach substitute output value • State = 6: Transition time measurement • State = 7: Error monitoring • State = 8: Approach substitute output value with error monitoring
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 2898) indicates the error messages.

See also

Parameter State and Retain.Mode V1 (Page 2891)

Parameter ErrorBits V1 (Page 2898)

PID_3Step V1 static tags (S7-1200)

Note

You must not change tags that are not listed. These are used for internal purposes only.

Change the tags identified with (1) only in "Inactive" mode to prevent malfunction of the PID controller. "Inactive" mode is forced by setting the "Retain.Mode" tag to "0".

Table 4-69

Tag	Data type	Default	Description
ActivateRecoverMode	BOOL	TRUE	The ActivateRecoverMode tag (Page 2900) determines the reaction to error.
RunModeByStartup	BOOL	TRUE	Activate Mode after CPU restart If RunModeByStartup = TRUE, the controller returns to the last active operating mode after a CPU restart. If RunModeByStartup = FALSE, the controller remains inactive after a CPU restart.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
ErrorBehaviour	INT	0	If ErrorBehaviour = 0 and an error has occurred, the valve stays at its current position and the controller switches directly to "Inactive" or "Error monitoring" mode. If ErrorBehaviour = 1 and an error occurs, the actuator moves to the substitute output value and only then switches to "Inactive" or "Error monitoring" mode. If the following errors occur, you can no longer move the valve to a configured substitute output value. • 2000h: Invalid value at Feedback_PER parameter. • 4000h: Invalid value at Feedback parameter. • 8000h: Error during digital position feedback.
Warning	DWORD	DW#16#0	The Warning tag (Page 2891) displays the warnings generated since a Reset or since the last operating mode switchover. Cyclic warnings (for example, process value warning) are shown until the cause of the warning is removed. They are automatically deleted once their cause has gone. Non-cyclic warnings (for example, point of inflection not found) remain and are deleted like errors.
SavePosition	REAL	0.0	Substitute output value If ErrorBehaviour = 1 and an error occurs, the actuator moves to a safe position for the plant and only then switches to "Inactive" mode.
CurrentSetpoint	REAL	0.0	Currently active setpoint. This value is frozen at the start of tuning.
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)
Config.InputPerOn(1)	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.OutputPerOn(1)	BOOL	FALSE	If OutputPerOn = TRUE, the Output_PER parameter is used. If OutputPerOn = FALSE, the Ouput_UP and Output_DN parameters are used.
Config.LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE.
Config.InvertControl(1)	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.FeedbackOn(1)	BOOL	FALSE	If FeedbackOn = FALSE, a position feedback is simulated. Position feedback is generally activated when FeedbackOn = TRUE.

Tag	Data type	Default	Description
Config.FeedbackPerOn(1)	BOOL	FALSE	FeedbackPerOn is only effective when FeedbackOn = TRUE. If FeedbackPerOn = TRUE, the analog input is used for the position feedback (Feedback_PER parameter). If FeedbackPerOn = FALSE, the Feedback parameter is used for the position feedback.
Config.ActuatorEndStopOn(1)	BOOL	FALSE	If ActuatorEndStopOn = TRUE, the digital position feedback Actuator_L and Actuator_H are taken into consideration.
Config.InputUpperLimit(1)	REAL	120.0	High limit of the process value At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit(1)	REAL	0.0	Low limit of the process value InputLowerLimit < InputUpperLimit
Config.InputUpperWarning(1)	REAL	+3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning(1)	REAL	-3.402822e+38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit(1)	REAL	100.0	High limit of output value The following value range is permitted: UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit For more details, see OutputLowerLimit.
Config.OutputLowerLimit(1)	REAL	0.0	Low limit of output value The following value range is permitted: OutputUpperLimit > OutputLowerLimit ≥ LowerPointOut When using Output_PER, an output value limit of -100% corresponds to the value Output_PER = -27648; 100% correspond to the value Output_PER = 27648. If OutputPerOn = FALSE and FeedbackOn = FALSE, OutputLowerLimit and OutputUpperLimit are not evaluated. Output_UP and Output_DN are then reset at Actuator_H = TRUE or Actuator_L = TRUE (if ActuatorEndStopOn = TRUE) or after a travel time of 110% * Config.TransitTime (if ActuatorEndStopOn = FALSE).

Tag	Data type	Default	Description
Config.SetpointUpperLimit(1)	REAL	+3.402822e+38	High limit of setpoint If you set SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is preassigned as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit(1)	REAL	-3.402822e+38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured absolute process value low limit is preassigned as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime(1)	REAL	0.0	Minimum ON time Minimum time in seconds for which the servo drive must be switched on. Config.MinimumOnTime is only effective if Output_UP and Output_DN are used (Config.OutputPerOn = FALSE).
Config.MinimumOffTime(1)	REAL	0.0	Minimum OFF time Minimum time in seconds for which the servo drive must be switched off. Config.MinimumOffTime is only effective if Output_UP and Output_DN are used (Config.OutputPerOn = FALSE).
Config.TransitTime(1)	REAL	30.0	Motor transition time Time in seconds the actuating drive requires to move the valve from the closed to the opened state.
Config.InputScaling.UpperPointIn(1)	REAL	27648.0	Scaling Input_PER high Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointIn(1)	REAL	0.0	Scaling Input_PER low Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.UpperPointOut(1)	REAL	100.0	Scaled high process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointOut(1)	REAL	0.0	Scaled low process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.FeedbackScaling.UpperPointIn(1)	REAL	27648.0	Scaling Feedback_PER high Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.

Tag	Data type	Default	Description
Config.FeedbackScaling.LowerPointIn(1)	REAL	0.0	Scaling Feedback_PER low Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.UpperPointOut(1)	REAL	100.0	High endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure. The permitted value range is determined by the configuration. • FeedbackOn = FALSE: UpperPointOut = 100.0 • FeedbackOn = TRUE: UpperPointOut = 100.0 or 0.0 UpperPointOut ≠ LowerPointOut
Config.FeedbackScaling.LowerPointOut(1)	REAL	0.0	Low endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure. The permitted value range is determined by the configuration. • FeedbackOn = FALSE: LowerPointOut = 0.0 • FeedbackOn = TRUE: LowerPointOut = 0.0 or -100.0 LowerPointOut ≠ UpperPointOut
GetTransitTime.InvertDirection	BOOL	FALSE	If InvertDirection = FALSE, the valve is fully opened, closed, and then reopened in order to determine the valve transition time. If InvertDirection = TRUE, the valve is fully closed, opened, and then closed again.
GetTransitTime.SelectFeedback	BOOL	FALSE	If SelectFeedback = TRUE, then Feedback_PER, or Feedback is taken into consideration in the transition time measurement. If SelectFeedback = FALSE, then Actuator_H and Actuator_L are taken into consideration in the transition time measurement.
GetTransitTime.Start	BOOL	FALSE	If Start = TRUE, the transition time measurement is started.
GetTransitTime.State	INT	0	Current phase of the transition time measurement • State = 0: Inactive • State = 1: Open valve completely • State = 2: Close valve completely • State = 3: Move valve to target position (NewOutput) • State = 4: Transition time measurement successfully completed • State = 5: Transition time measurement canceled
GetTransitTime.NewOutput	REAL	0.0	Target position for transition time measurement with position feedback The target position must be between "High endstop" and "Low endstop". The difference between NewOutput and ScaledFeedback must be at least 50% of the permissible control range.

Tag	Data type	Default	Description
CycleTime.StartEstimation	BOOL	TRUE	If StartEstimation = TRUE, the measurement of the PID_3Step sampling time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If EnEstimation = TRUE, the PID_3Step sampling time is calculated.
CycleTime.EnMonitoring	BOOL	TRUE	If EnMonitoring = TRUE, the PID_3Step sampling time is monitored. If it is not possible to execute PID_3Step within the sampling time, the error 0800h is output and the operating mode is switched. ActivateRecoverMode and ErrorBehaviour determine which operating mode is switched to. If EnMonitoring = FALSE, the PID_3Step sampling time is not monitored, the error 0800h is not output, and the operating mode is not switched.
CycleTime.Value(1)	REAL	0.1	PID_3Step sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser. You can reload values from the CtrlParamsBackUp structure with Config.LoadBackUp = TRUE.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time
CtrlParamsBackUp.TdFiltRatio	REAL	0.0	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	0.0	Saved proportional action weighting
CtrlParamsBackUp.DWeighting	REAL	0.0	Saved derivative action weighting
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm
CtrlParamsBackUp.InputDeadBand	REAL	0.0	Saved dead zone width of the control deviation
PIDSelfTune.SUT.CalculateSUTParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateSUTParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRuleSUT. CalculateSUTParams is set to FALSE following calculation.
PIDSelfTune.SUT.TuneRuleSUT	INT	1	Methods used to calculate parameters during pretuning: • TuneRuleSUT = 0: PID fast I • TuneRuleSUT = 1: PID slow I • TuneRuleSUT = 2: Chien, Hrones and Reswick PID • TuneRuleSUT = 3: Chien, Hrones, Reswick PI • TuneRuleSUT = 4: PID fast II • TuneRuleSUT = 5: PID slow II
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning:

Tag	Data type	Default	Description
PIDSelfTune.TIR.RunIn	BOOL	FALSE	• RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_3Step switches to "Inactive" mode. • RunIn = TRUE The pretuning is skipped. PID_3Step attempts to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start. RunIn is set to FALSE after fine tuning.
PIDSelfTune.TIR.Calculate-TIRParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateTIRParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRuleTIR. CalculateTIRParams is set to FALSE following calculation.
PIDSelfTune.TIR.TuneRule-TIR	INT	0	Methods used to calculate parameters during fine tuning: • TuneRuleTIR = 0: PID automatic • TuneRuleTIR = 1: PID fast • TuneRuleTIR = 2: PID slow • TuneRuleTIR = 3: Ziegler-Nichols PID • TuneRuleTIR = 4: Ziegler-Nichols PI • TuneRuleTIR = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of "fine tuning":
Retain.Mode	INT	0	A change to the value of Retain.Mode initiates a switch to another operating mode. The following operating mode is enabled upon a change of Mode to: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode • Mode = 5: Approach substitute output value • Mode = 6: Transition time measurement • Mode = 7: Error monitoring • Mode = 8: Approach substitute output value with error monitoring Mode is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.SetByUser(1)	BOOL	FALSE	If SetByUser = FALSE, the PID parameters are determined automatically and PID_3Step operates with a dead zone at the output value. The dead zone width is calculated during tuning on the basis of the standard deviation of the output value and saved in Retain.CtrlParams.OutputDeadBand. If SetByUser = TRUE, the PID parameters are entered manually and PID_3 Step operates without a dead zone at the output value. Retain.CtrlParams.OutputDeadBand = 0.0 SetByUser is retentive.
Retain.CtrlParams.Gain(1)	REAL	1.0	Active proportional gain Gain is retentive.
Retain.CtrlParams.Ti(1)	REAL	20.0	• Ti > 0.0: Active integral action time • Ti = 0.0: Integral action is deactivated Ti is retentive.
Retain.CtrlParams.Td(1)	REAL	0.0	• Td > 0.0: Active derivative action time • Td = 0.0: Derivative action is deactivated Td is retentive.
Retain.CtrlParams.TdFiltRatio(1)	REAL	0.0	Active derivative delay coefficient TdFiltRatio is retentive.
Retain.CtrlParams.PWeighting(1)	REAL	0.0	Active proportional action weighting PWeighting is retentive.
Retain.CtrlParams.DWeighting(1)	REAL	0.0	Active derivative action weighting DWeighting is retentive.
Retain.CtrlParams.Cycle(1)	REAL	1.0	Active sampling time of PID algorithm in seconds, rounded to an integer multiple of the cycle time of the calling OB. Cycle is retentive.
Retain.CtrlParams.InputDeadBand(1)	REAL	0.0	Dead zone width of the control deviation InputDeadBand is retentive.

See also

Parameter State and Retain.Mode V1 (Page 2891)

Tag ActivateRecoverMode V1 (Page 2900)

Parameter State and Retain.Mode V1 (S7-1200)

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

To switch from one operating mode to another, you must change the Retain.Mode tag. This also applies when the value for the new operating mode is already in Retain.Mode. For example, set Retain.Mode = 0 first and then Retain.Mode = 3. Provided the current operating mode of the controller permits this switchover, State will be set to the value of Retain.Mode.

When PID_3Step automatically switches from one operating mode to another, the following applies: State != Retain.Mode.

Examples:

- After successful pretuning
  State = 3 and Retain.Mode = 1
- In the event of an error
  State = 0 and Retain.Mode remain at the previous value, for example, Retain.Mode = 3
- ManualEnalbe = TRUE
  State = 4 and Retain.Mode remain at the previous value, e.g., Retain.Mode = 3

Note

You want, for example, to repeat successful fine tuning without exiting automatic mode with Mode = 0.

Setting Retain.Mode to an invalid value such as 9999 for one cycle has no effect on State. Set Mode = 2 in the next cycle. In this way, you can generate a change to Retain.Mode without first switching to "Inactive" mode.

Meaning of values

State / Retain.Mode	Description
0	Inactive The controller is switched off and no longer changes the valve position.
1	Pretuning The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The optimized PID parameters are calculated as a function of the maximum rate of rise and dead time of the controlled system. Pretuning requirements: • State = 0 or State = 4 • ManualEnable = FALSE • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher as compared to the noise. Before the PID parameters are recalculated, they are backed up and can be reactivated with Config.LoadBackUp. The setpoint is frozen in the CurrentSetpoint tag. The controller switches to automatic mode following successful pretuning and to "Inactive" mode following unsuccessful pretuning. The pretuning phase is indicated with the SUT.State tag.

State / Retain.Mode	Description
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned based on the amplitude and frequency of this oscillation. The differences between the process response during pretuning and fine tuning are analyzed. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before fine tuning. They can be reactivated with Config.LoadBackUp. The setpoint is frozen in the CurrentSetpoint tag. Requirements for fine tuning: • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. • ManualEnable = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started from: • Automatic mode (State = 3) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual mode (State = 4) Pretuning is always started first. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. If PIDSelfTune.TIR.RunIn = TRUE, pretuning is skipped and an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. The controller switches to automatic mode following successful fine tuning. If fine tuning was not successful, the controller switches to "Inactive" mode. The fine tuning phase is indicated with the TIR.State tag.
3	Automatic mode In automatic mode, PID_3Step controls the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Changing the Retain.Mode tag to the value 3. When the CPU is switched on or switches from Stop to RUN mode, PID_3Step starts in the most recently active operating mode. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE. The ActivateRecoverMode tag is taken into consideration in automatic mode.

State / Retain.Mode	Description
4	Manual mode In manual mode, you specify manual output values in the Manual_UP and Manual_DN parameters or ManualValue parameter. Whether or not the actuator can be moved to the output value in the event of an error is described in the ErrorBits parameter. This operating mode is enabled if Retain.Mode = 4, or on a rising edge at ManualEnable. If ManualEnable changes to TRUE, only State changes. Retain.Mode retains its current value. On a falling edge at ManualEnable, PID_3Step returns to the previous operating mode. The switchover to automatic mode is bumpless. **PID_3Step V1.1** Manual mode is always possible in the event of an error. **PID_3Step V1.0** Manual mode is dependent on the ActivateRecoverMode tag in the event of an error.
5	Approach substitute output value This operating mode is activated in the event of an error or when Reset = TRUE if Errorbehaviour = 1 and ActivateRecoverMode = FALSE.. PID_3Step moves the actuator to the substitute output value and then switches to "Inactive" mode.
6	Transition time measurement The time that the motor needs to completely open the valve from the closed condition is determined. This operating mode is activated when GetTransitTime.Start = TRUE is set. If endstop signals are used to measure the transition time, the valve will be opened completely from its current position, closed completely, and opened completely again. If GetTransitTime.InvertDirection = TRUE, this behavior is inverted. If position feedback is used to measure the transition time, the actuator will be moved from its current position to a target position. The output value limits are not taken into consideration during the transition time measurement. The actuator can travel to the high or the low endstop.
7	Error monitoring The control algorithm is switched off and no longer changes the valve position. This operating mode is activated instead of "Inactive" mode in the event of an error. All the following conditions must be met: • Mode = 3 (automatic mode) • Errorbehaviour = 0 • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 2900) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.
8	Approach substitute output value with error monitoring This operating mode is activated instead of "Approach substitute output value" mode in the event of an error. PID_3Step moves the actuator to the substitute output value and then switches to "Error monitoring" mode. All the following conditions must be met: • Mode = 3 (automatic mode) • Errorbehaviour = 1 • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 2900) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

Automatic switchover of operating mode during commissioning

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour on the switchover of operating mode from transition time measurement, pretuning, and fine tuning modes.

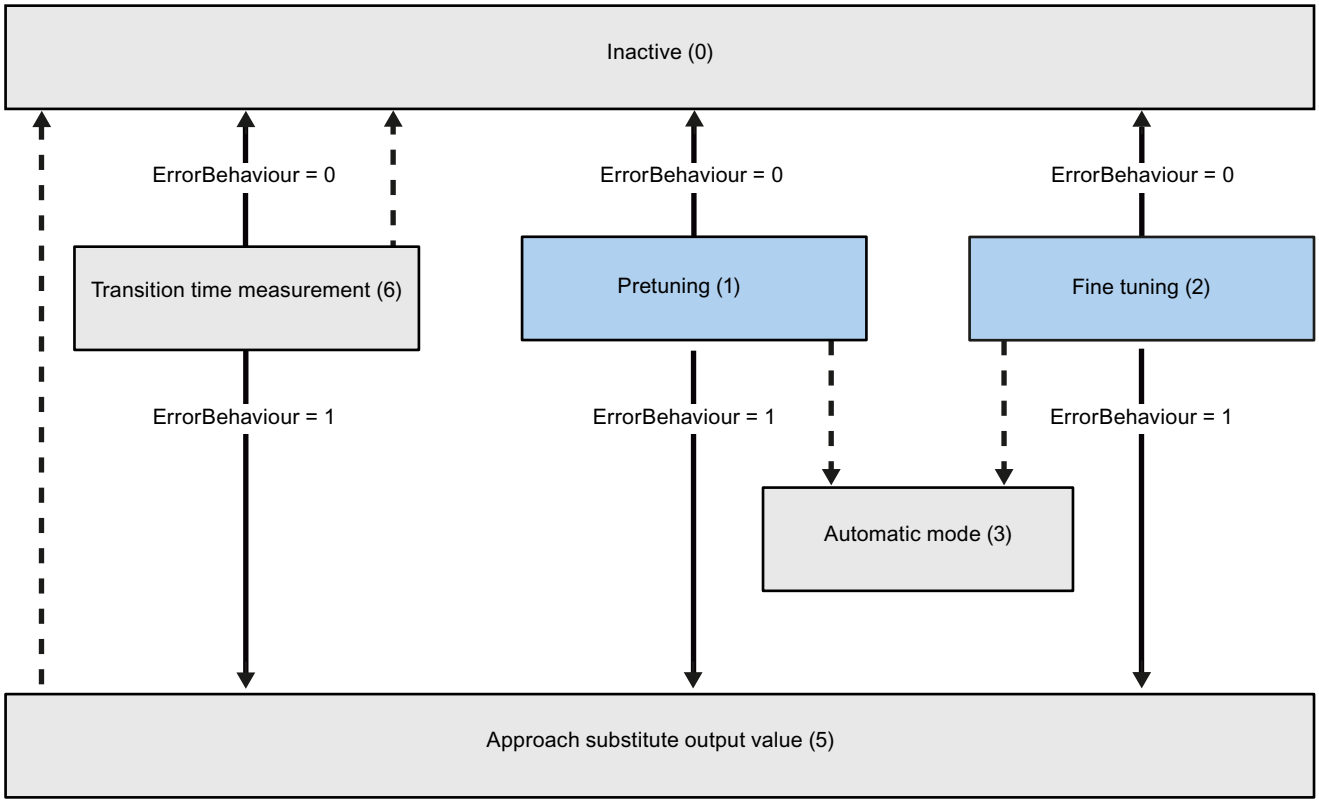

Automatic switchover of operating mode in the event of an error

Automatic switchover of operating mode once the current operation has been completed.

Automatic switchover of operating mode in automatic mode (PID_3Step V1.1)

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.

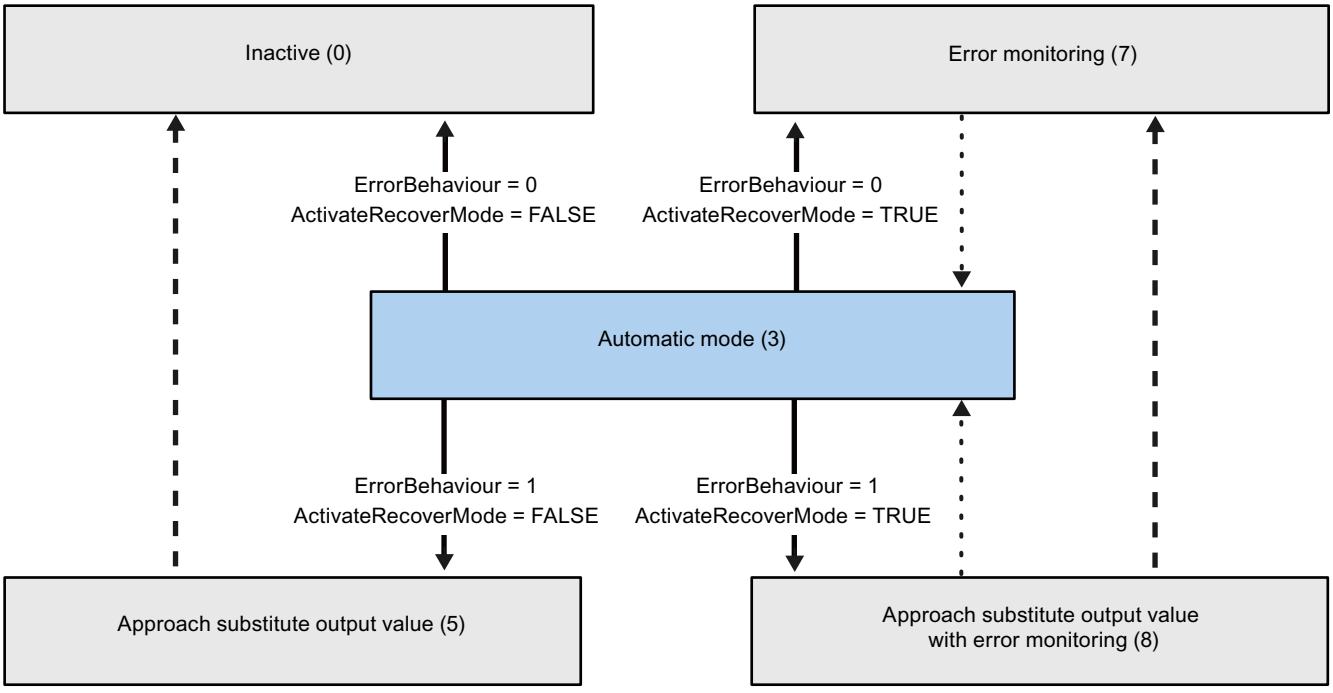

Automatic switchover of operating mode in the event of an error

Automatic switchover of operating mode once the current operation has been completed.

Automatic switchover of operating mode in automatic and manual modes (PID_3Step V1.0)

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.

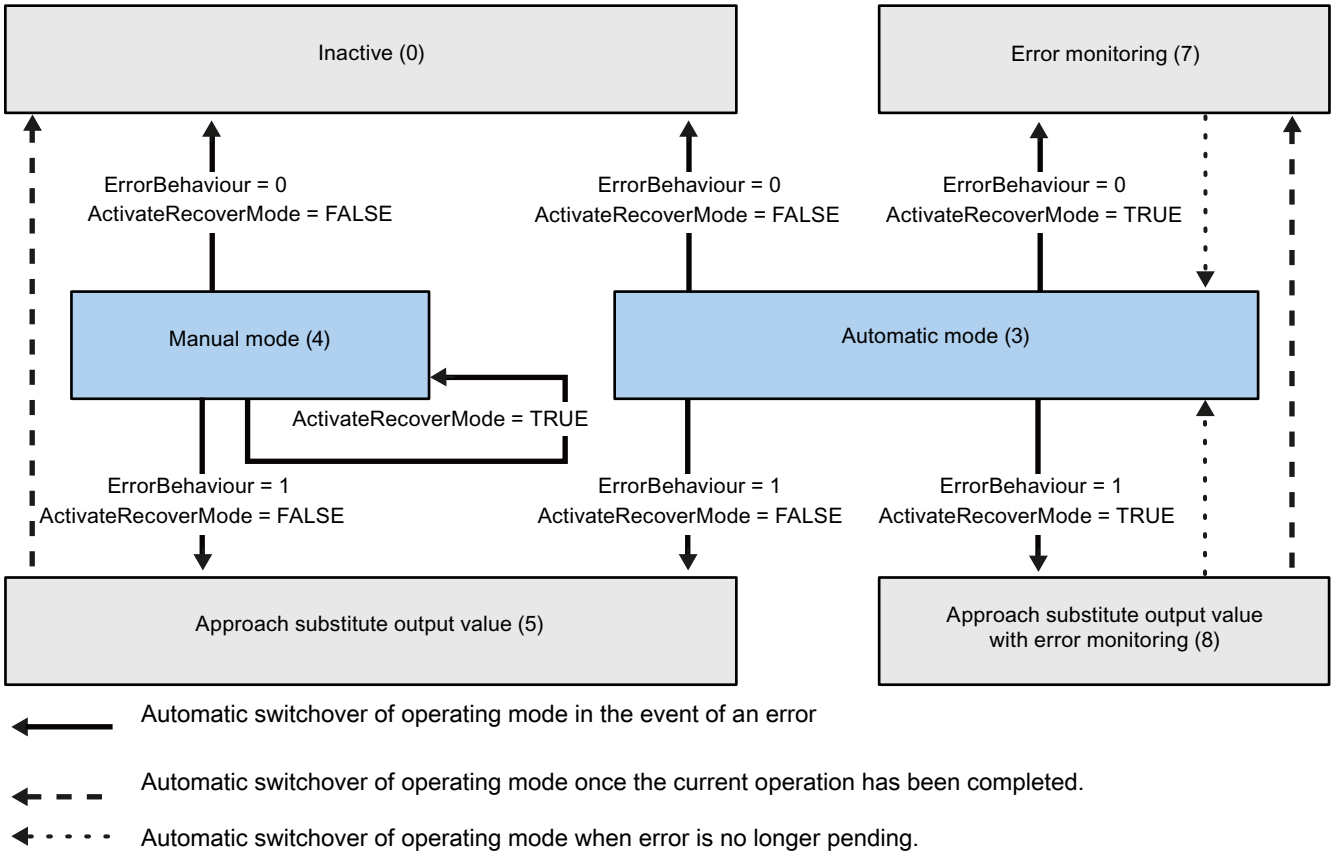

See also

Tag ActivateRecoverMode V1 (Page 2900)

Parameter ErrorBits V1 (Page 2898)

Parameter ErrorBits V1 (S7-1200)

If several errors are pending simultaneously, the values of the error codes are displayed with binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending simultaneously.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If ActivateRecoverMode = TRUE and ErrorBehaviour = 1, the actuator moves to the substitute output value. If ActivateRecoverMode = TRUE and ErrorBehaviour = 0, the actuator stops in its current position. If ActivateRecoverMode = FALSE, the actuator stops in its current position. PID_3Step V1.1 You can move the actuator in manual mode. PID_3Step V1.0 Manual mode is not possible in this state. You cannot move the actuator again until you eliminate the error.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
0004	Error during fine tuning. Oscillation of the process value could not be maintained.
0020	Pretuning is not permitted in automatic mode or during fine tuning.
0080	Error during pretuning. The output value limits are not configured correctly or the actual value does not react as expected. Check whether the limits of the output value are configured correctly and match the control logic. Also make sure that the actual value does not oscillate strongly before starting pretuning.
0100	Error during fine tuning resulted in invalid parameters.
0200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
0400	Calculation of output value failed. Check the PID parameters.
0800	Sampling time error: PID_3Step is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode. If this error occurred during simulation with PLCSIM, see the notes under AUTOHOTSPOT.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.

ErrorBits (DW#16#...)	Description
2000	Invalid value at Feedback_PER parameter. Check whether an error is pending at the analog input. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. You must deactivate position feedback (Config. FeedbackOn = FALSE) to move the actuator from this state. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
4000	Invalid value at Feedback parameter. Value has an invalid number format. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. You must deactivate position feedback (Config. FeedbackOn = FALSE) to move the actuator from this state. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
8000	Error during digital position feedback. Actuator_H = TRUE and Actuator_L = TRUE. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. In order to move the actuator from this state, you must deactivate the "Actuator endstop" (Config.ActuatorEndStopOn = FALSE). If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.

Parameter Reset V1 (S7-1200)

A rising edge at Reset triggers a change to "Inactive" mode, and errors and warnings are reset. A falling edge at Reset triggers a change to the most recently active operating mode. If automatic mode was active before, switchover to automatic mode is bumpless.

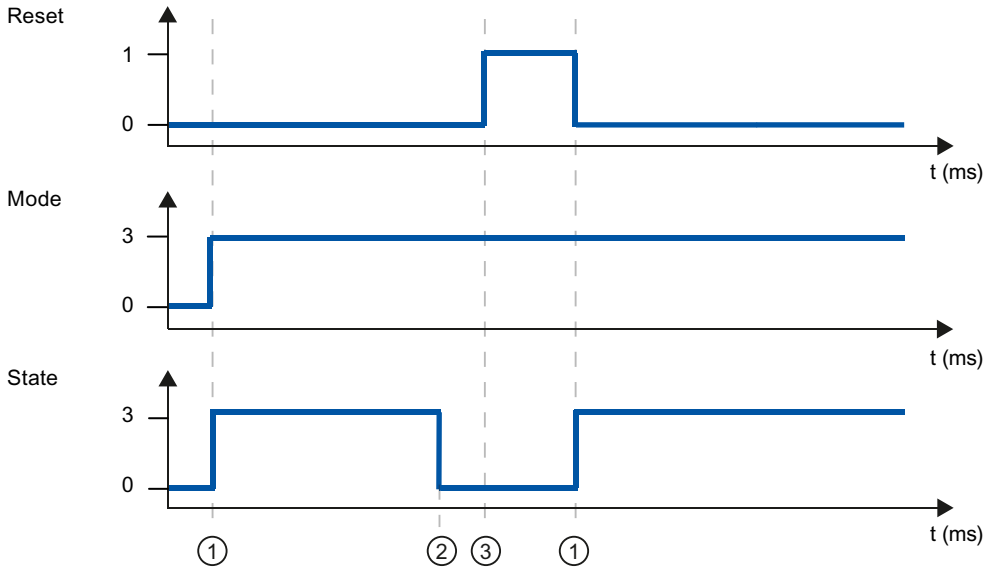

① Activation
② Error
③ Reset

Tag ActivateRecoverMode V1 (S7-1200)

The effect of the ActivateRecoverMode variable depends on the version of the PID_3Step.

Behavior in version 1.1

The ActivateRecoverMode variable determines the behavior in the event of an error in automatic mode. ActivateRecoverMode is not effective during pretuning, fine tuning and transition time measurement.

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" operating mode. The controller is activated by a reset or a change in Retain.Mode.
TRUE	If errors occur frequently in automatic mode, this setting has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, PID_3Step **cannot** approach the configured substitute output value. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

Behavior in version 1.0

The ActivateRecoverMode variable determines the behavior in the event of an error in automatic and manual mode. ActivateRecoverMode is not effective during pretuning, fine tuning and transition time measurement.

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" operating mode. The controller is activated by a reset or a change in Retain.Mode.
TRUE	**Errors in automatic mode** If errors occur frequently in automatic mode, this setting has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, PID_3Step **cannot** approach the configured substitute output value. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode. **Errors in manual mode** If one or more of the following errors occur, PID_3Step stays in manual mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, you **cannot** move the valve to a suitable position.

See also

PID_3Step V1 static tags (Page 2884)

Parameter State and Retain.Mode V1 (Page 2891)

Tag Warning V1 (S7-1200)

If several warnings are pending simultaneously, their values are displayed with binary addition. The display of warning 0003, for example, indicates that the warnings 0001 and 0002 are pending simultaneously.

Warning (DW#16#...)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0002	Oscillation increased during fine tuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. The PID parameters were instead calculated using the TuneRuleTIR = 3 method.
0010	The operating mode could not be changed because ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Retain.Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The rule used for tuning produces an incorrect result, or is not supported.
0400	Method selected for transition time measurement not suitable for actuator. The transition time cannot be measured because the actuator settings do not match the selected measuring method.
0800	The difference between the current position and the new output value is too small for transition time measurement. This can produce incorrect results. The difference between the current output value and new output value must be at least 50% of the entire control range.
1000	The substitute output value cannot be reached because it is outside the output value limits.

The following warnings are deleted as soon as the cause is eliminated:

- 0004
- 0020
- 0040
- 0100

All other warnings are cleared with a rising edge at Reset.

Tag SUT.State V1 (S7-1200)

SUT.State	Name	Description
0	SUT_INIT	Initialize pretuning
50	SUT_TPDN	Determine start position without position feedback
100	SUT_STDABW	Calculate the standard deviation
200	SUT_GET_POI	Find the point of inflection
300	SUT_GET_RISETM	Determine the rise time

SUT.State	Name	Description
9900	SUT_IO	Pretuning successful
1	SUT_NIO	Pretuning not successful

Tag TIR.State V1 (S7-1200)

TIR.State	Name	Description
-100	TIR_FIRST_SUT	Fine tuning is not possible. Pretuning will be executed first.
0	TIR_INIT	Initialize fine tuning
200	TIR_STDABW	Calculate the standard deviation
300	TIR_RUN_IN	Attempt to reach the setpoint with the maximum or minimum output value
400	TIR_CTRLN	Attempt to reach the setpoint with the existing PID parameters (if pretuning has been successful)
500	TIR_OSZIL	Determine oscillation and calculate parameters
9900	TIR_IO	Fine tuning successful
1	TIR_NIO	Fine tuning not successful

PID_Temp (S7-1200, S7-1500)

New features of PID_Temp (S7-1200, S7-1500)

PID_Temp V1.1

- **Response of the output value on switchover from "Inactive" operating mode to "Automatic mode"**
  The new option IntegralResetMode = 4 was added and defined as default. With IntegralResetMode = 4, the integral action is automatically pre-assigned when switching from "Inactive" operating mode to "Automatic mode" so that a control deviation results in a jump of the PID output value with identical sign.
- **Initialization of the integral action in automatic mode**
  The integral action can be initialized in automatic mode with the tags OverwriteInitialOutputValue and PIDCtrl.PIDInit. This simplifies the use of PID_Temp for override controls.

Compatibility with CPU and FW (S7-1200, S7-1500)

The following table shows which version of PID_Temp can be used on which CPU.

CPU	FW	PID_Temp
S7-1200	V4.2 or higher	V1.1 V1.0
	V4.1	V1.0

CPU	FW	PID_Temp
S7-1500	V2.0 or higher	V1.1 V1.0
	V1.7 to V1.8	V1.0

CPU processing time and memory requirement PID_Temp V1 (S7-1200, S7-1500)

CPU processing time

Typical CPU processing times of the PID_Temp technology object as of Version 1.0, depending on CPU type.

CPU	Typ. CPU processing time PID_Temp V1
CPU 1211C ≥ V4.1	580 µs
CPU 1215C ≥ V4.1	580 µs
CPU 1217C ≥ V4.1	580 µs
CPU 1505S ≥ V1.0	50 µs
CPU 1510SP-1 PN ≥ V1.7	130 µs
CPU 1511-1 PN ≥ V1.7	130 µs
CPU 1512SP-1 PN ≥ V1.7	130 µs
CPU 1516-3 PN/DP ≥ V1.7	75 µs
CPU 1518-4 PN/DP ≥ V1.7	6 µs

Memory requirement

Memory requirement of an instance DB of the PID_Temp technology object as of Version V1.0.

	Memory requirement of the instance DB of PID_Temp V1
Load memory requirement	Approx. 17000 bytes
Total work memory requirement	1280 bytes
Retentive work memory requirement	100 bytes

PID_Temp (S7-1200, S7-1500)

Description of PID_Temp (S7-1200, S7-1500)

Description

The PID_Temp instruction provides a PID controller with integrated tuning for temperature processes. PID_Temp can be used for pure heating or heating/cooling applications.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Substitute output value with error monitoring

For a more detailed description of the operating modes, see the State parameter.

PID algorithm

PID_Temp is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation (control zone and dead zone deactivated):

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

The table below shows the meaning of the icons used in the equation and in the subsequent figures.

Icon	Description	Associated parameters of the PID_Temp instruction
y	Output value of the PID algorithm	-
K_p	Proportional gain	Retain.CtrlParams.Heat.Gain Retain.CtrlParams.Cool.Gain CoolFactor
s	Laplace operator	-
b	Proportional action weighting	Retain.CtrlParams.Heat.PWeighting Retain.CtrlParams.Cool.PWeighting
w	Setpoint	CurrentSetpoint
x	Process value	ScaledInput
T_I	Integral action time	Retain.CtrlParams.Heat.Ti Retain.CtrlParams.Cool.Ti
T_D	Derivative action time	Retain.CtrlParams.Heat.Td Retain.CtrlParams.Cool.Td
a	Derivative delay coefficient (derivative delay T1 = a × T_D)	Retain.CtrlParams.Heat.TdFiltRatio Retain.CtrlParams.Cool.TdFiltRatio
c	Derivative action weighting	Retain.CtrlParams.Heat.DWeighting Retain.CtrlParams.Cool.DWeighting
DeadZone	Dead zone width	Retain.CtrlParams.Heat.DeadZone Retain.CtrlParams.Cool.DeadZone
ControlZone	Control zone width	Retain.CtrlParams.Heat.ControlZone Retain.CtrlParams.Cool.ControlZone

PID_Temp block diagram

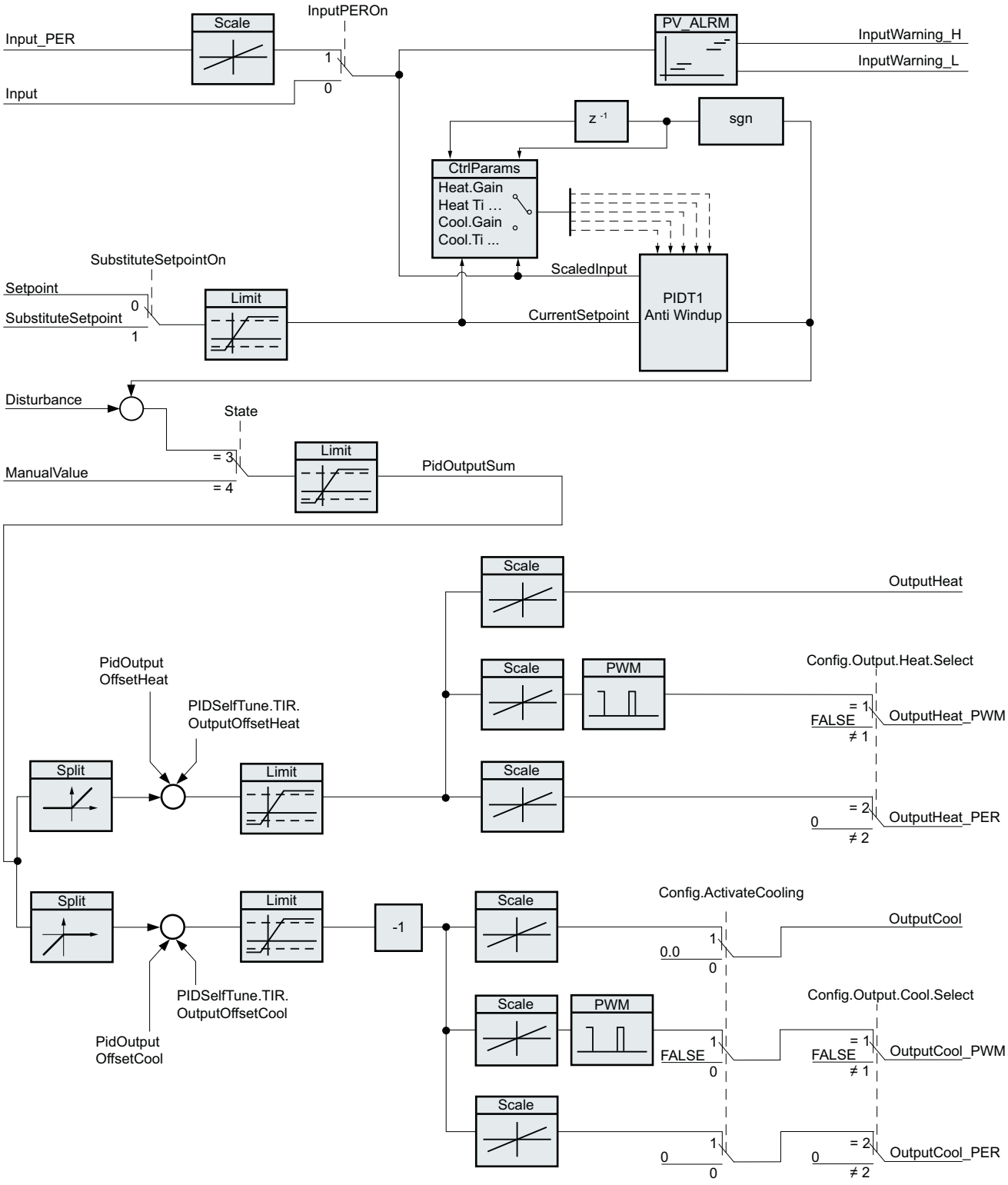

Block diagram of PIDT1 with anti-windup

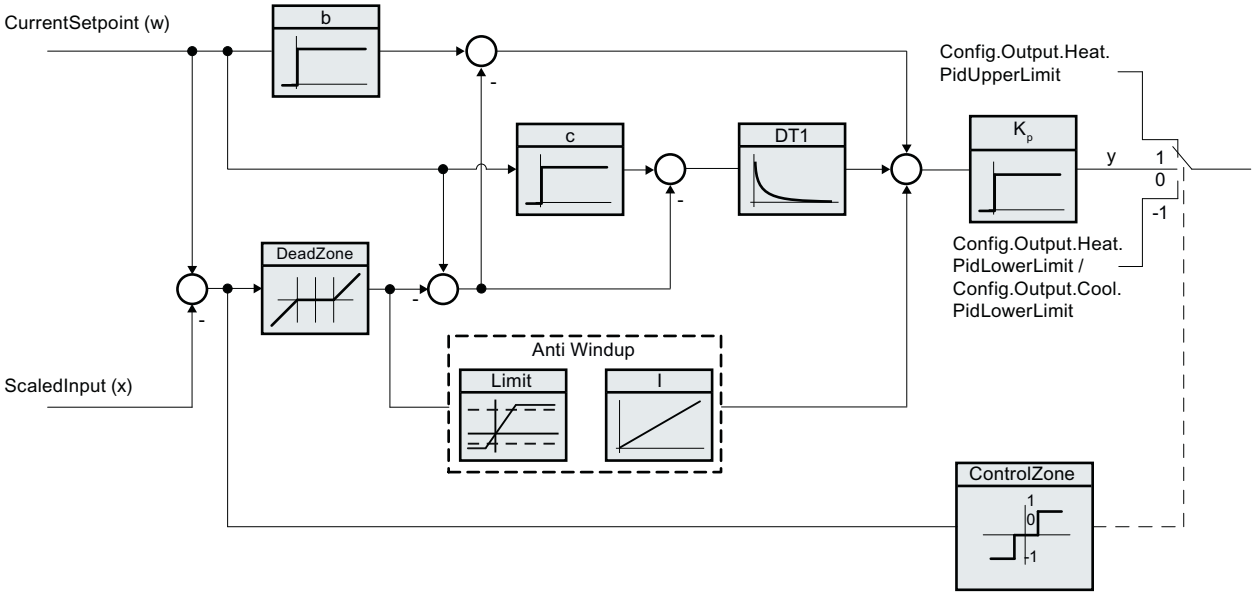

Call

PID_Temp is called in the constant time scale of a cyclic interrupt OB.

If you call PID_Temp as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_Temp directly in the multi-instance DB and commission it via a watch table.

Download to device

The process values of retentive tags are only updated when you download PID_Temp completely.

Download technology object to device

Startup

When the CPU starts up, PID_Temp starts in the operating mode that is saved in the Mode in/out parameter. To switch to "Inactive" operating mode during startup, set RunModeByStartup = FALSE.

Reaction to error

The behavior in the case of an error is determined by the tags SetSubstituteOutput and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the behavior also depends on the error that occurred.

SetSubstituteOutput	ActivateRecoverMode	Configuration editor > Basic settings of output > Set PidOutputSum to	Reaction
Not relevant	FALSE	Zero (inactive)	Switch to "Inactive" (State = 0) mode The output value of the PID algorithm and all outputs for heating and cooling are set to 0. The scaling of the outputs for heating and cooling is not active.
FALSE	TRUE	Current value for error while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The current output value is transferred to the actuator while the error is pending.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The value at SubstituteOutput is transferred to the actuator while the error is pending.

In manual mode, PID_Temp uses ManualValue as output value, unless ManualValue is invalid.

- If ManualValue is invalid, SubstituteOutput is used.
- If ManualValue and SubstituteOutput are invalid, Config.Output.Heat.PidLowerLimit is used.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.

Mode of operation of PID_Temp (S7-1200, S7-1500)

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0000001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warning = 0000040h), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID_Temp automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller area. PID_Temp checks whether this area is within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Substitute setpoint

You can specify a substitute setpoint at the SubstituteSetpoint tag and activate it with SubstituteSetpointOn = TRUE. In this way, you can temporarily specify the setpoint directly, for example for a slave controller in a cascade, without having to change the user program. The limits set for the setpoint also apply to the substitute setpoint.

Heating and cooling

With the default setting, PID_Temp only uses the outputs for heating (OutputHeat, OutputHeat_PWM, OutputHeat_PER). The output value of the PID algorithm (PidOutputSum) is scaled and output at the outputs for heating. You specify with Config.Output.Heat.Select if OutputHeat_PWM or OutputHeat_PER is calculated. OutputHeat is always calculated.

With Config.ActivateCooling = TRUE, you can also activate the outputs for cooling (OutputCool, OutputCool_PWM, OutputCool_PER). Positive output values of the PID algorithm (PidOutputSum) are scaled and output at the outputs for heating. Negative output values of the PID algorithm are scaled and output at the outputs for cooling. You specify with Config.Output.Cool.Select if OutputCool_PWM or OutputCool_PER is calculated. OutputCool is always calculated.

Two methods are available to calculate the PID output value with activated cooling:

- Cooling factor (Config.AdvancedCooling = FALSE):
  The output value calculation for cooling takes place with the PID parameters for heating, taking into consideration the configurable cooling factor Config.CoolFactor. This method is suitable if the heating and cooling actuators have a similar time response but different gains. If this method is selected, pretuning and fine tuning for cooling as well as the PID parameter set for cooling are not available. You can only execute the tuning for heating.
- PID parameter switching (Config.AdvancedCooling = TRUE):
  The output value calculation for cooling takes place by means of a separate PID parameter set. Based on the calculated output value and the control deviation, the PID algorithm decides whether the PID parameter for heating or cooling is used. This method is suitable if the heating and cooling actuator have different time responses and different gains. Pretuning and fine tuning for cooling are only available if this method is selected.

Output value limits and scaling

Depending on the operating mode, the PID output value (PidOutputSum) is calculated automatically by the PID algorithm or defined by the manual value (ManualValue) or the configured substitute output value (SubstituteOutput).

The PID output value is limited according to the configuration:

- If cooling is deactivated (Config.ActivateCooling = FALSE), Config.Output.Heat.PidUpperLimit is the high limit and Config.Output.Heat.PidLowerLimit the low limit.
- If cooling is activated (Config.ActivateCooling = TRUE), Config.Output.Heat.PidUpperLimit is the high limit and Config.Output.Cool.PidLowerLimit the low limit.

The PID output value is scaled and output at the outputs for heating and cooling. Scaling can be defined separately for each output and is specified in the structures Config.Output.Heat or Config.Output.Cool with 2 value pairs each:

Output	Value pair	Parameter
OutputHeat	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high output value (heating) Config.Output.Heat.UpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low output value (heating) Config.Output.Heat.LowerScaling
OutputHeat_PWM	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high PWM output value (heating) Config.Output.Heat.PwmUpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low PWM output value (heating) Config.Output.Heat.PwmLowerScaling
OutputHeat_PER	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high analog output value (heating) Config.Output.Heat.PerUpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low analog output value (heating) Config.Output.Heat.PerLowerScaling
OutputCool	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high output value (cooling) Config.Output.Cool.UpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low output value (cooling) Config.Output.Cool.LowerScaling

Output	Value pair	Parameter
OutputCool_PWM	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high PWM output value (cooling) Config.Output.Cool.PwmUpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low PWM output value (cooling) Config.Output.Cool.PwmLowerScaling
OutputCool_PER	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high analog output value (cooling) Config.Output.Cool.PerUpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low analog output value (cooling) Config.Output.Cool.PerLowerScaling

If cooling is activated (Config.ActivateCooling = TRUE), Config.Output.Heat.PidLowerLimit must have the value 0.0.

Config.Output.Cool.PidUpperLimit must always have the value 0.0.

Example:

Output scaling when using output OutputHeat (cooling deactivated; Config.Output.Heat.PidLowerLimit may be unequal to 0.0):

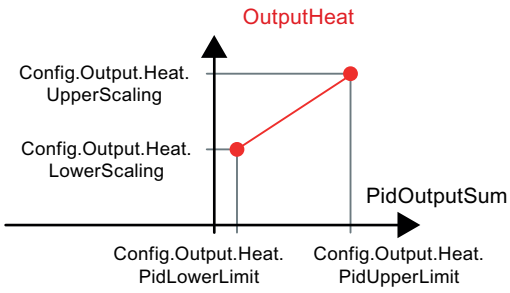

Example:

Output scaling when using output OutputHeat_PWM and OutputCool_PER (cooling activated; Config.Output.Heat.PidLowerLimit must be 0.0):

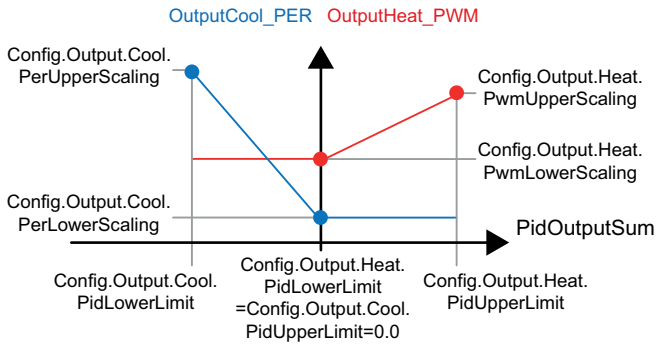

With the exception of the "Inactive" operating mode, the value at an output is always located between its scaled high output value and scaled low output value, for example, for OutputHeat always between Config.Output.Heat.UpperScaling and Config.Output.Heat.LowerScaling.

If you want to limit the value at the associated output, you therefore have to adapt these scaling values as well.

Cascading

PID_Temp supports you when you use cascade control (see: AUTOHOTSPOT).

Substitute output value

In the event of an error, PID_Temp can output a substitute output value that you define at the SubstituteOutput tag. The substitute output value must be within the limits for the PID output value. The values at the outputs for heating and cooling resulting from the substitute output value are the result of the configured output scaling.

Monitoring signal validity

The values of the following parameters are monitored for validity when used:

- Setpoint
- SubstituteSetpoint
- Input
- Input_PER
- Disturbance
- ManualValue
- SubstituteOutput
- PID parameters in the structures Retain.CtrlParams.Heat and Retain.CtrlParams.Cool.

Monitoring the sampling time PID_Temp

Ideally, the sampling time is equivalent to the cycle time of the cyclic interrupt OB. The PID_Temp instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (Error = 0000800h).

The error occurs during tuning if:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

The error occurs in automatic mode if:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID_Temp in OB1. You must then accept a lower control quality due to the deviating sampling time.

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time of the cyclic interrupt OB (sampling time PID_Temp). All other functions of the PID_Temp are executed at every call.

If cooling and PID parameter switching are activated, PID_Temp uses a separate sampling time of the PID algorithm for heating and cooling. In all other configurations, only the sampling time of the PID algorithm for heating is used.

If you use OutputHeat_PWM or OutputCool_PWM, the sampling time of the PID algorithm is used as time period of the pulse width modulation. The accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The cycle time should be no more than a tenth of the PID algorithm sampling time.

If the PID algorithm sampling time and thus the time period of the pulse width modulation is very high when you use OutputHeat_PWM or OutputCool_PWM, you can define a deviating shorter period duration at the Config.Output.Heat.PwmPeriode or Config.Output.Cool.PwmPeriode parameters to improve the smoothness of the process value.

Control logic

PID_Temp can be used for heating or heating/cooling applications and always works with normal control logic.

An increase of the PID output value (PidOutputSum) is intended to increase the process value. The values at the outputs for heating and cooling resulting from the PID output value are the result of the configured output scaling.

An inverted control logic or negative proportional gain are not supported.

If you only need an output value for your application in which an increase is to reduce the process value (for example, discharge control), you can use PID_Compact with inverted control logic.

Input parameters of PID_Temp (S7-1200, S7-1500)

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode Valid range of values: Config.SetpointUpperLimit ≥ Setpoint ≥ Config.SetpointLowerLimit Config.InputUpperLimit ≥ Setpoint ≥ Config.InputLowerLimit
Input	REAL	0.0	A tag of the user program is used as source for the process value. If you are using the Input parameter, Config.InputPerOn = FALSE must be set.
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using the Input_PER parameter, Config.InputPerOn = TRUE must be set.
Disturbance	REAL	0.0	Disturbance variable or precontrol value
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge activates "Manual mode", whileState = 4, Mode remains unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog. • A TRUE -> FALSE edge activates the operating mode that is specified by Mode. We recommend that you change the operating mode using Mode and ModeActivate only.
ManualValue	REAL	0.0	Manual value This value is used in manual mode as PID output value (PidOutputSum). The values at the outputs for heating and cooling resulting from this manual value are the result of the configured output scaling (structures Config.Output.Heat and Config.Output.Cool). For controllers with activated cooling output (Config.ActivateCooling = TRUE), define: • a positive manual value to output the value at the outputs for heating • a negative manual value to output the value at the outputs for cooling The permitted value range is determined by the configuration. • Cooling output deactivated (Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ ManualValue ≥ Config.Output.Heat.PidLowerLimit • Cooling output activated (Config.ActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ ManualValue ≥ Config.Output.Cool.PidLowerLimit
ErrorAck	BOOL	FALSE	• FALSE -> TRUE edge ErrorBits and Warning are reset.

Parameter	Data type	Default	Description
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Switch to "Inactive" mode – ErrorBits and Warning are reset. • As long as Reset = TRUE, – PID_Temp remains in "Inactive" mode (State = 0). – you cannot change the operating mode with Mode and ModeActivate or ManualEnable – You cannot use the commissioning dialog. • TRUE -> FALSE edge – If ManualEnable = FALSE, PID_Temp switches to the operating mode that is saved in Mode. – If Mode = 3 (automatic mode), the integral action is treated as configured with the tag IntegralResetMode.
ModeActi-vate	BOOL	FALSE	• FALSE -> TRUE edge PID_Temp switches to the operating mode that is saved at the Mode input.

Output parameters of PID_Temp (S7-1200, S7-1500)

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
OutputHeat	REAL	0.0	Output value (heating) in REAL format The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Heat.PidUpperLimit, Config.Output.Heat.UpperScaling and Config.Output.Heat.PidLowerLimit, Config.Output.Heat.LowerScaling and output in REAL format at OutputHeat. OutputHeat is always calculated.
OutputCool	REAL	0.0	Output value (cooling) in REAL format The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Cool.PidUpperLimit, Config.Output.Cool.LowerScaling and Config.Output.Cool.PidLowerLimit, Config.Output.Cool.UpperScaling and output in REAL format at OutputCool. OutputCool is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE).
Outpu-tHeat_PER	INT	0	Analog output value (heating) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Heat.PidUpperLimit, Config.Output.Heat.PerUpperScaling and Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PerLowerScaling and output as analog value at OutputHeat_PER. OutputHeat_PER is only calculated if Config.Output.Heat.Select = 2.

Parameter	Data type	Default	Description
Output-Cool_PER	INT	0	Analog output value (cooling) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Cool.PidUpperLimit, Config.Output.Cool.PerLowerScaling and Config.Output.Cool.PidLowerLimit, Config.Output.Cool.PerUpperScaling and output as analog value at OutputCool_PER. OutputCool_PER is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE) and Config.Output.Cool.Select = 2.
OutputHeat_PWM	BOOL	FALSE	Pulse-width modulated output value (heating) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Heat.PidUpperLimit, Config.Output.Heat.PwmUpperScaling and Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PwmLowerScaling and output as pulse-width modulated value (variable switch on and switch off times) at OutputHeat_PWM. OutputHeat_PWM is only calculated if Config.Output.Heat.Select = 1.
Output-Cool_PWM	BOOL	FALSE	Pulse-width modulated output value (cooling) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Cool.PidUpperLimit, Config.Output.Cool.PwmLowerScaling and Config.Output.Cool.PidLowerLimit, Config.Output.Cool.PwmUpperScaling and output as pulse-width modulated value (variable switch on and switch off times) at OutputCool_PWM. OutputCool_PWM is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE) and Config.Output.Cool.Select = 1.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit) or Setpoint ≥ Config.InputUpperLimit. The setpoint high limit is the minimum of Config.SetpointUpperLimit and Config.InputUpperLimit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit is reached (Setpoint ≤ Config.SetpointLowerLimit) or Setpoint ≤ Config.InputLowerLimit. The setpoint low limit is the maximum of Config.SetpointLowerLimit and Config.InputLowerLimit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit (ScaledInput ≥ Config.InputUpperWarning).
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit (ScaledInput ≤ Config.InputLowerWarning).
State	INT	0	The PID_Temp state and mode parameters (Page 2950) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. For pretuning and fine tuning, you specify with Heat.EnableTuning and Cool.EnableTuning whether tuning takes place for heating or cooling. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Substitute output value with error monitoring

Parameter	Data type	Default	Description
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The PID_Temp ErrorBits parameter (Page 2958) shows the pending error messages. ErrorBits is retentive and is reset with a rising edge at Reset or ErrorAck.

In/out parameters of PID_Temp V2 (S7-1200, S7-1500)

Parameter	Data type	Default	Description
Mode	INT	4	At Mode, specify the operating mode to which PID_Temp is to switch. Options are: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode The operating mode is activated by: • Rising edge at ModeActivate • Falling edge at Reset • Falling edge at ManualEnable • Cold restart of CPU if RunModeByStartup = TRUE For pretuning and fine tuning, you specify with Heat.EnableTuning and Cool.EnableTuning whether tuning takes place for heating or cooling. Mode is retentive. A detailed description of the operating modes can be found in State and Mode parameters (Page 2950).
Master	DWORD	DW#16#0	Interface for cascade control If this PID_Temp instance is used as slave controller in a cascade (Config.Cascade.IsSlave = TRUE), assign the Master parameter at the instruction call with the Slave parameter of the master controller. Example: Call of a slave controller "PID_Temp_2" with master controller "PID_Temp_1" in SCL: ---------- `"PID_Temp_2"(Master := "PID_Temp_1".Slave, Setpoint := "PID_Temp_1".OutputHeat);` ---------- You use this interface to exchange slave controller information about operating mode, limit and substitute setpoint with your master controller. Keep in mind that the call of the master controller has to take place before the call of the slave controller in the same cyclic interrupt OB. Assignment: • Bits 0 to 15: Unassigned • Bits 16 to 23 – Limit counter: A slave controller whose output value is limited increments this counter. Depending on the configured number of slaves (Config.Cascade.CountSlaves) and of the anti-windup mode (Config.Cascade.AntiWindUpMode), the master controller reacts accordingly. • Bit 24 – Automatic mode of the slave controllers: TRUE, if all slave controllers are in automatic mode • Bit 25 – Substitute setpoint of the slave controllers: TRUE, if a slave controller has activated the substitute setpoint (SubstituteSetpointOn = TRUE)

Parameter	Data type	Default	Description
Slave	DWORD	DW#16#0	Interface for cascade control You use this interface to exchange slave controller information about operating mode, limit and substitute setpoint with your master controller. See description of Master parameter

See also

PID_Temp state and mode parameters (Page 2950)

PID_Temp static tags (S7-1200, S7-1500)

Note

Change the tags identified with (1) only in "Inactive" mode to prevent malfunction of the PID controller.

Tag	Data type	Default	Description
IntegralResetMode	Int	V1.0: 1, V1.1 or higher: 4	The IntegralResetMode tag (Page 2966) determines how the integral action PIDCtrl.IOutputOld is pre-assigned when switching from "Inactive" operating mode to "Automatic mode". This setting only works for one cycle. • IntegralResetMode = 0: Smooth • IntegralResetMode = 1: Delete • IntegralResetMode = 2: Hold • IntegralResetMode = 3: Pre-assign • IntegralResetMode = 4: Like setpoint change (only for PID_Temp with version ≥ 1.1)
OverwriteInitialOutputValue	REAL	0.0	If one of the following conditions is met, the integral action PIDCtrl.IOutputOld is pre-assigned automatically as if PIDOutputSum = OverwriteInitialOutputValue in the previous cycle: • IntegralResetMode = 3 when switching from "Inactive" operating mode to "Automatic mode" • TRUE -> FALSE edge at parameter Reset and parameter Mode = 3 • PIDCtrl.PIDInit = TRUE in "Automatic mode" (available as of PID_Temp version 1.1)
RunModeByStartup	BOOL	TRUE	Activate operating mode at Mode parameter after CPU restart • If RunModeByStartup = TRUE, PID_Temp starts in the operating mode saved in the Mode parameter after CPU startup. • If RunModeByStartup = FALSE, PID_Temp remains in "Inactive" mode after CPU startup.

Tag	Data type	Default	Description
LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded from the CtrlParamsBackUp structure. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE. The acceptance is bumpless.
SetSubstituteOutput	BOOL	TRUE	Selection of the output value while an error is pending (State = 5): • If SetSubstituteOutput = TRUE and ActivateRecoverMode = TRUE, the configured substitute output value SubstituteOutput is output as PID output value as long as an error is pending. • If SetSubstituteOutput = FALSE and ActivateRecoverMode = TRUE, the actuator remains at the current PID output value as long as an error is pending. • If ActivateRecoverMode = FALSE, SetSubstituteOutput is not effective. • If SubstituteOutput is invalid (ErrorBits = 0020000h), the substitute output value cannot be output. In this case, the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit) is used as PID output value.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF. This parameter is used for display in the editors and does not influence the control algorithm.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature. This parameter is used for display in the editors and does not influence the control algorithm.
ActivateRecoverMode	BOOL	TRUE	The ActivateRecoverMode tag determines the reaction to error.
Warning	DWORD	0	The Warning tag shows the warnings since Reset = TRUE or ErrorAck =TRUE. Warning is retentive.
Progress	REAL	0.0	Progress of current tuning phase as a percentage (0.0 - 100.0)
CurrentSetpoint	REAL	0.0	CurrentSetpoint always displays the currently effective setpoint. This value is frozen during tuning.
CancelTuningLevel	REAL	10.0	Permissible fluctuation of setpoint during tuning. Tuning is not canceled until: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel

Tag	Data type	Default	Description
SubstituteOutput	REAL	0.0	The substitute output value is used as PID output value as long as the following conditions are met: • One or more errors are pending in automatic mode for which ActivateRecoverMode is in effect • SetSubstituteOutput = TRUE • ActivateRecoverMode = TRUE The values at the outputs for heating and cooling resulting from the substitute output value are the result of the configured output scaling (structures Config.Output.Heat and Config.Output.Cool). For controllers with activated cooling output (Config.ActivateCooling = TRUE), define: • a positive substitute output value to output the value at the outputs for heating • a negative substitute output value to output the value at the outputs for cooling The permitted value range is determined by the configuration. • Cooling output deactivated (Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ SubstituteOutput ≥ Config.Output.Heat.PidLowerLimit • Cooling output activated (Config.ActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ SubstituteOutput ≥ Config.Output.Cool.PidLowerLimit
PidOutputSum	REAL	0.0	PID output value PidOutputSum displays the output value of the PID algorithm. Depending on the operating mode, it is either calculated automatically or defined by the manual value or the configured substitute output value. The values at the outputs for heating and cooling resulting from the PID output value are the result of the configured output scaling (structures Config.Output.Heat and Config.Output.Cool). The PidOutputSum is limited as defined in the configuration. • Cooling output deactivated (Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ PidOutputSum ≥ Config.Output.Heat.PidLowerLimit • Cooling output activated (ConfigActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ PidOutputSum ≥ Config.Output.Cool.PidLowerLimit

Tag	Data type	Default	Description
PidOutputOffsetHeat	REAL	0.0	Offset of the PID output value heating PidOutputOffsetHeat is added to the value that results from PidOutputSum for the heating branch. Enter a positive value for PidOutputOffsetHeat to receive a positive offset at the outputs for heating. The resulting values at the outputs for heating are the result of the configured output scaling (Config.Output.Heat structure). This offset can be used for actuators which need a fixed minimum value, for example, fans with minimum speed.
PidOutputOffsetCool	REAL	0.0	Offset of the PID output value cooling PidOutputOffsetCool is added to the value that results from PidOutputSum for the cooling branch. Enter a negative value for PidOutputOffsetCool to receive a positive offset at the outputs for cooling. The resulting values at the outputs for cooling are the result of the configured output scaling (Config.Output.Cool structure). This offset can be used for actuators which need a fixed minimum value, for example, fans with minimum speed.
SubstituteSetpointOn	BOOL	FALSE	Activates the substitute setpoint as controller setpoint. • FALSE = the Setpoint parameter is used. • TRUE = the SubstituteSetpoint parameter is used as setpoint SubstituteSetpointOn can be used to specify the setpoint of a slave controller in a cascade directly without having to change the user program.
SubstituteSetpoint	REAL	0.0	Substitute setpoint If SubstituteSetpointOn = TRUE, the SubstituteSetpoint parameter is used as setpoint. Valid range of values: Config.SetpointUpperLimit ≥ SubstituteSetpoint ≥ Config.SetpointLowerLimit, Config.InputUpperLimit ≥ SubstituteSetpoint ≥ Config.InputLowerLimit
DisableCooling	BOOL	FALSE	DisableCooling = TRUE deactivates the cooling branch for heating/cooling controllers (Config.ActivateCooling = TRUE) in Automatic mode by setting PidOutputSum to 0.0 as low limit. PidOutputOffsetCool and the output scaling for the cooling outputs remain active. DisableCooling can be used for tuning of multi-zone applications to temporarily deactivate the cooling branch as long as all controllers have not completed their tuning yet. This parameter is set/reset by the user manually and is not automatically reset by the PID_Temp instruction.

Tag	Data type	Default	Description
AllSlaveAutomaticState	BOOL	FALSE	If this PID_Temp instance is used as master controller in a cascade (Config.Cascade.IsMaster = TRUE), AllSlaveAutomaticState = TRUE indicates that all slave controllers are in automatic mode. Tuning, manual mode or automatic mode of the master controller can only be executed accurately if all slave controllers are in automatic mode. AllSlaveAutomaticState is only determined if you interconnect the master controller and slave controller with the Master and Slave parameters. For details, see the Master parameter.
NoSlaveSubstituteSetpoint	BOOL	FALSE	If this PID_Temp instance is used as master controller in a cascade (Config.Cascade.IsMaster = TRUE), NoSlaveSubstituteSetpoint = TRUE indicates that no slave controller has activated its substitute setpoint. Tuning, manual mode or automatic mode of the master controller can only be executed accurately if no slave controller has activated its substitute setpoint. NoSlaveSubstituteSetpoint is only determined if you interconnect the master controller and slave controller with the Master and Slave parameters. For details, see the Master parameter.
Heat.EnableTuning	BOOL	TRUE	Enabling of tuning for heating Heat.EnableTuning must be set for the following tunings (at the same time or prior to the start with Mode and ModeActivate): • Pretuning heating • Pretuning heating and cooling • Fine tuning heating This parameter is not automatically reset by the PID_Temp instruction.
Cool.EnableTuning	BOOL	FALSE	Enabling of tuning for cooling Cool.EnableTuning must be set for the following tunings (simultaneously with or prior to the start with Mode and ModeActivate): • Pretuning cooling • Pretuning heating and cooling • Fine tuning cooling Only effective if the cooling output and PID parameter switching are activated ("Config.ActivateCooling" = TRUE and "Config.AdvancedCooling" = TRUE). This parameter is not automatically reset by the PID_Temp instruction.
Config.InputPerOn(1)	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used for detecting the process value. If InputPerOn = FALSE, the Input parameter is used.

Tag	Data type	Default	Description
Config.InputUpperLimit(1)	REAL	120.0	High limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. If the limit is exceeded, an error is output and the reaction is determined by ActivateRecoverMode. At the I/O input, the process value can be a maximum of 18% higher than the nominal range (overrange). This means the limit cannot be exceeded when you use an I/O input with the pre-setting for high limit and process value scaling. When pretuning is started, the difference between high and low limit of the process value is checked to determine whether the distance between setpoint and process value meets the necessary requirements. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit(1)	REAL	0.0	Low limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. If the limit is undershot, an error is output and the reaction is determined by ActivateRecoverMode. InputLowerLimit < InputUpperLimit
Config.InputUpperWarning(1)	REAL	3.402822e+38	Warning high limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. If the limit is exceeded, a warning is output at the Warning parameter. • If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. • If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning
Config.InputLowerWarning(1)	REAL	-3.402822e+38	Warning low limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. If the limit is undershot, a warning is output at the Warning parameter. • If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. • If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning

Tag	Data type	Default	Description
Config.SetpointUpperLimit(1)	REAL	3.402822e +38	High limit of setpoint Setpoint and SubstituteSetpoint are monitored to ensure adherence to this limit. If the limit is exceeded, a warning is output at the Warning parameter. • If you configure SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is used as the setpoint high limit. • If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit. SetpointUpperLimit > SetpointLowerLimit
Config.SetpointLowerLimit(1)	REAL	-3.402822e +38	Low limit of the setpoint Setpoint and SubstituteSetpoint are monitored to ensure adherence to this limit. If the limit is undershot, a warning is output at the Warning parameter. • If you set SetpointLowerLimit outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. • If you configure SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit. SetpointLowerLimit < SetpointUpperLimit
Config.ActivateCooling(1)	BOOL	FALSE	Activate cooling output • Config.ActivateCooling = FALSE Only the outputs for heating are used. • Config.ActivateCooling = TRUE The outputs for heating and cooling are used. If you are using the cooling output, the controller must not be configured as master controller (Config.Cascade.IsMaster must be FALSE) .

Tag	Data type	Default	Description
Config.AdvancedCooling(1)	BOOL	TRUE	Method for heating/cooling • Cooling factor (Config.AdvancedCooling = FALSE) The output value calculation for cooling takes place with the PID parameters for heating (Retain.CtrlParams.Heat structure) taking into consideration the configurable cooling factor Config.CoolFactor. This method is suitable if the heating and cooling actuators have a similar time response but different gains. Pretuning and fine tuning for cooling are not available when you select this method. You can only execute the tuning for heating. • PID parameter switching (Config.AdvancedCooling = TRUE) The output value calculation for cooling takes place by means of a separate PID parameter set (Retain.CtrlParams.Cool structure). This method is suitable if the heating and cooling actuator have different time responses and different gains. Pretuning and fine tuning for cooling are only available when you select this method (Mode = 1 or 2, Cool.EnableTuning = TRUE). Config.AdvancedCooling is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE).
Config.CoolFactor(1)	REAL	1.0	Cooling factor If Config.AdvancedCooling = FALSE, Config.CoolFactor is considered as factor in the calculation of the output value for cooling. This allows different gains of heating and cooling actuators to be taken into account. Config.CoolFactor is not set automatically or adjusted during tuning. You must correctly configure Config.CoolFactor manually with the ratio "heating actuator gain/cooling actuator gain". Example: Config.CoolFactor = 2.0 means that the gain of the heating actuator is twice as high as the gain of the cooling actuator. Config.CoolFactor is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and cooling factor is selected as method for heating/cooling (Config.AdvancedCooling = FALSE). Config.CoolFactor > 0.0
Config.InputScaling.UpperPointIn(1)	REAL	27648.0	Scaling Input_PER high Input_PER is scaled based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). UpperPointIn > LowerPointIn

Tag	Data type	Default	Description
Config.InputScaling.LowerPoin-tIn(1)	REAL	0.0	Scaling Input_PER low Input_PER is scaled based on the two value pairs UpperPoint-Out, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). LowerPointIn < UpperPointIn
Config.InputScaling.UpperPoint-Out(1)	REAL	100.0	Scaled high process value Input_PER is scaled based on the two value pairs UpperPoint-Out, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). UpperPointOut > LowerPointOut
Config.InputScaling.LowerPoint-Out(1)	REAL	0.0	Scaled low process value Input_PER is scaled based on the two value pairs UpperPoint-Out, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). LowerPointOut < UpperPointOut
Config.Output.Heat.Select(1)	INT	1	Selecting the output value for heating Config.Output.Heat.Select specifies which outputs are used for heating: • Heat.Select = 0 - OutputHeat is used • Heat.Select = 1 - OutputHeat and OutputHeat_PWM are used • Heat.Select = 2 -OutputHeat and OutputHeat_PER are used Outputs that are not used are not calculated and remain at their default value.

Tag	Data type	Default	Description
Config.Output.Heat.PwmPeriode(1)	REAL	0.0	Period duration of the pulse width modulation (PWM) for heating (OutputHeat_PWM output) in seconds: • Heat.PwmPeriode = 0.0 The sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as period duration of the PWM. • Heat.PwmPeriode > 0.0 The value is rounded off to an integer multiple of the PID_Temp sampling time (CycleTime.Value) and used as period duration of the PWM. This setting can be used to improve the smoothing of the process value with a long sampling time of the PID algorithm. The value must meet the following conditions: – Heat.PwmPeriode ≤ Retain.CtrlParams.Heat.Cycle, – Heat.PwmPeriode > Config.Output.Heat.MinimumOnTime – Heat.PwmPeriode > Config.Output.Heat.MinimumOffTime
Config.Output.Heat.PidUpperLimit(1)	REAL	100.0	High limit of the PID output value for heating The PID output value (PidOutputSum) is limited to the high limit. Heat.PidUpperLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for heating: • Heat.UpperScaling for OutputHeat • Heat.PwmUpperScaling for OutputHeat_PWM • Heat.PerUpperScaling for OutputHeat_PER If you want to limit the value at the associated output, you must also adjust these scaling values. Heat.PidUpperLimit > Heat.PidLowerLimit

Tag	Data type	Default	Description
Config.Output.Heat.PidLowerLimit(1)	REAL	0.0	Low limit of the PID output value for heating For controllers with **deactivated** cooling output (Config.ActivateCooling = FALSE), the PID output value (PidOutputSum) is limited to this low limit. For controllers with **activated** cooling output (Config.ActivateCooling = TRUE), the value must be 0.0. Heat.PidLowerLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for heating: • Heat.LowerScaling for OutputHeat • Heat.PwmLowerScaling for OutputHeat_PWM • Heat.PerLowerScaling for OutputHeat_PER If you want to limit the value at the associated output, you must also adjust these scaling values. The permitted value range is determined by the configuration. • Cooling output deactivated (Config.ActivateCooling = FALSE): Heat.PidLowerLimit < Heat.PidUpperLimit • Cooling output activated (Config.ActivateCooling = TRUE): Heat.PidLowerLimit = 0.0
Config.Output.Heat.UpperScaling(1)	REAL	100.0	Scaled high output value for heating Heat.UpperScaling and Heat.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for heating (OutputHeat). The OutputHeat value is always located between Heat.UpperScaling and Heat.LowerScaling. Heat.UpperScaling ≠ Heat.LowerScaling
Config.Output.Heat.LowerScaling(1)	REAL	0.0	Scaled low output value for heating Heat.LowerScaling and Heat.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for heating (OutputHeat). The OutputHeat value is always located between Heat.UpperScaling and Heat.LowerScaling. Heat.UpperScaling ≠ Heat.LowerScaling
Config.Output.Heat.PwmUpperScaling(1)	REAL	100.0	Scaled high PWM output value for heating Heat.PwmUpperScaling and Heat.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for heating (OutputHeat_PWM). The OutputHeat_PWM value is always located between Heat.PwmUpperScaling and Heat.PWMLowerScaling. Heat.PwmUpperScaling is only effective if OutputHeat_PWM is selected as output for heating (Heat.Select = 1) 100.0 ≥ Heat.PwmUpperScaling ≥ 0.0 Heat.PwmUpperScaling ≠ Heat.PwmLowerScaling

Tag	Data type	Default	Description
Config.Output.Heat.PwmLowerScaling(1)	REAL	0.0	Scaled low PWM output value for heating Heat.PwmLowerScaling and Heat.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for heating (OutputHeat_PWM). The OutputHeat_PWM value is always located between Heat.PwmUpperScaling and Heat.PwmLowerScaling. Heat.PwmLowerScaling is only effective if OutputHeat_PWM is selected as output for heating (Heat.Select = 1) 100.0 ≥ Heat.PwmLowerScaling ≥ 0.0 Heat.PwmUpperScaling ≠ Heat.PwmLowerScaling
Config.Output.Heat.PerUpperScaling(1)	REAL	27648.0	Scaled high analog output value for heating Heat.PerUpperScaling and Heat.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for heating (OutputHeat_PER). The OutputHeat_PER value is always located between Heat.PerUpperScaling and Heat.PerLowerScaling. Heat.PerUpperScaling is only effective if OutputHeat_PER is selected as output for heating (Heat.Select = 2) 32511.0 ≥ Heat.PerUpperScaling ≥ -32512.0 Heat.PerUpperScaling ≠ Heat.PerLowerScaling
Config.Output.Heat.PerLowerScaling(1)	REAL	0.0	Scaled low analog output value for heating Heat.PerLowerScaling and Heat.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for heating (OutputHeat_PER). The OutputHeat_PER value is always located between Heat.PerUpperScaling and Heat.PerLowerScaling. Heat.PerLowerScaling is only effective if OutputHeat_PER is selected as output for heating (Heat.Select = 2) 32511.0 ≥ Heat.PerLowerScaling ≥ -32512.0 Heat.PerUpperScaling ≠ Heat.PerLowerScaling
Config.Output.Heat.MinimumOnTime(1)	REAL	0.0	Minimum on time of the pulse width modulation for heating (OutputHeat_PWM output) A PWM pulse is never shorter than this value. The value is rounded off to: Heat.MinimumOnTime = n × CycleTime.Value Heat.MinimumOnTime is only effective if the output for heating OutputHeat_PWM is selected (Heat.Select = 1)". 100000.0 ≥ Heat.MinimumOnTime ≥ 0.0

Tag	Data type	Default	Description
Config.Output.Heat.MinimumOff-Time(1)	REAL	0.0	Minimum off time of the pulse width modulation for heating (OutputHeat_PWM output) A PWM pause is never shorter than this value. The value is rounded off to: Heat.MinimumOffTime = n × CycleTime.Value Heat.MinimumOffTime is only effective if the output for heating OutputHeat_PWM is selected (Heat.Select = 1)". 100000.0 ≥ Heat.MinimumOffTime ≥ 0.0
Config.Output.Cool.Select(1)	INT	1	Selecting the output value for cooling Config.Output.Cool.Select specifies which outputs are used for cooling: • Cool.Select = 0 - OutputCool is used • Cool.Select = 1 -OutputCool and OutputCool_PWM are used • Cool.Select = 2 - OutputCool and OutputCool_PER are used Outputs that are not used are not calculated and remain at their default value. Only effective if the cooling output is activated (Config.Activa-teCooling = TRUE).

Tag	Data type	Default	Description
Config.Output.Cool.PwmPeriode(1)	REAL	0.0	Period duration of the pulse width modulation for cooling (OutputCool_PWM output) in seconds: • Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = FALSE: sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as period duration of the PWM. • Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = TRUE: The sampling time of the PID algorithm for cooling (Retain.CtrlParams.Cool.Cycle) is used as period duration of the PWM. • Cool.PwmPeriode > 0.0: The value is rounded off to an integer multiple of the PID_Temp sampling time (CycleTime.Value) and used as period duration of the PWM. This setting can be used to improve the smoothing of the process value with a long sampling time of the PID algorithm. The value must meet the following conditions: – Cool.PwmPeriode ≤ Retain.CtrlParams.Cool.Cycle or Retain.CtrlParams.Heat.Cycle – Cool.PwmPeriode > Config.Output.Cool.MinimumOnTime – Cool.PwmPeriode > Config.Output.Cool.MinimumOffTime Only effective if the cooling output is activated (Config.ActivateCooling = TRUE).
Config.Output.Cool.PidUpperLimit(1)	REAL	0.0	High limit of the PID output value for cooling The value must be 0.0. Cool.PidUpperLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for cooling: • Cool.LowerScaling for OutputCool • Cool.PwmLowerScaling for OutputCool_PWM • Cool.PerLowerScaling for OutputCool_PER If you want to limit the value at the associated output, you must also adjust these scaling values. Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). Cool.PidUpperLimit = 0.0

Tag	Data type	Default	Description
Config.Output.Cool.PidLowerLimit(1)	REAL	-100.0	Low limit of the PID output value for cooling For controllers with activated cooling output (Config.ActivateCooling = TRUE), the PID output value (PidOutputSum) is limited to this low limit. Cool.PidLowerLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for cooling: • Cool.UpperScaling for OutputCool • Cool.PwmUpperScaling for OutputCool_PWM • Cool.PerUpperScaling for OutputCool_PER If you want to limit the value at the associated output, you must also adjust these scaling values. Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). Cool.PidLowerLimit < Cool.PidUpperLimit
Config.Output.Cool.UpperScaling(1)	REAL	100.0	Scaled high output value for cooling Cool.UpperScaling and Cool.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for cooling (OutputCool). The OutputCool value is always located between Cool.UpperScaling and Cool.LowerScaling. Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). Cool.UpperScaling ≠ Cool.LowerScaling
Config.Output.Cool.LowerScaling(1)	REAL	0.0	Scaled low output value for cooling Cool.LowerScaling and Cool.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for cooling (OutputCool). The OutputCool value is always located between Cool.UpperScaling and Cool.LowerScaling. Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). Cool.UpperScaling ≠ Cool.LowerScaling
Config.Output.Cool.PwmUpperScaling(1)	REAL	100.0	Scaled high PWM output value for cooling Cool.PwmUpperScaling and Cool.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for cooling (OutputCool_PWM). The OutputCool_PWM value is always located between Cool.PwmUpperScaling and Cool.PwmLowerScaling. Cool.PwmUpperScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PWM is selected as output for cooling (Cool.Select = 1). 100.0 ≥ Cool.PwmUpperScaling ≥ 0.0 Cool.PwmUpperScaling ≠ Cool.PwmLowerScaling

Tag	Data type	Default	Description
Config.Output.Cool.PwmLowerScaling(1)	REAL	0.0	Scaled low PWM output value for cooling Cool.PwmLowerScaling and Cool.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for cooling (OutputCool_PWM). The OutputCool_PWM value is always located between Cool.PwmUpperScaling and CoolPwm.LowerScaling. Cool.PwmLowerScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PWM is selected as output for cooling (Cool.Select = 1). 100.0 ≥ Cool.PwmLowerScaling ≥ 0.0 Cool.PwmUpperScaling ≠ Cool.PwmLowerScaling
Config.Output.Cool.PerUpperScaling(1)	REAL	27648.0	Scaled high analog output value for cooling Cool.PerUpperScaling and Cool.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for cooling (OutputCool_PER). The OutputCool_PER value is always located between Cool.PerUpperScaling and Cool.PerLowerScaling. Cool.PerUpperScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PER is selected as output for cooling (Cool.Select = 2). 32511.0 ≥ Cool.PerUpperScaling ≥ -32512.0 Cool.PerUpperScaling ≠ Cool.PerLowerScaling
Config.Output.Cool.PerLowerScaling(1)	REAL	0.0	Scaled low analog output value for cooling Cool.PerLowerScaling and Cool.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for cooling (OutputCool_PER). The OutputCool_PER value is always located between Cool.PerUpperScaling and Cool.PerLowerScaling. Cool.PerLowerScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PER is selected as output for cooling (Cool.Select = 2). 32511.0 ≥ Cool.PerLowerScaling ≥ -32512.0 Cool.PerUpperScaling ≠ Cool.PerLowerScaling
Config.Output.Cool.MinimumOnTime(1)	REAL	0.0	Minimum on time of the pulse width modulation for cooling (OutputCool_PWM output) A PWM pulse is never shorter than this value. The value is rounded off to: Cool.MinimumOnTime = n × CycleTime.Value Cool.MinimumOnTime is only effective if the output for cooling OutputCool_PWM is selected (Cool.Select = 1). Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). 100000.0 ≥ Cool.MinimumOnTime ≥ 0.0

Tag	Data type	Default	Description
Config.Output.Cool.MinimumOff-Time(1)	REAL	0.0	Minimum off time of the pulse width modulation for cooling (OutputCool_PWM output) A PWM pause is never shorter than this value. The value is rounded off to: Cool.MinimumOffTime = n × CycleTime.Value Cool.MinimumOffTime is only effective if the output for cooling OutputCool_PWM is selected (Cool.Select = 1). Only effective if the cooling output is activated (Config.Activa-teCooling = TRUE). 100000.0 ≥ Cool.MinimumOffTime ≥ 0.0
If you are using PID_Temp in a cascade, the master controller and slave controller exchange information via the Master and Slave parameters. You need to make the interconnection. For details, see the Master parameter.			
Config.Cascade.IsMaster(1)	BOOL	FALSE	The controller is master in a cascade and provides the slave setpoint. Set IsMaster = TRUE if you are using this PID_Temp instance as master controller in a cascade. A master controller defines the setpoint of a slave controller with its output. A PID_Temp instance can be master controller and slave controller at the same time. If the controller is used as master controller, the cooling output must be deactivated (Config.ActivateCooling = FALSE).
Config.Cascade.IsSlave(1)	BOOL	FALSE	The controller is slave in a cascade and receives its setpoint from the master. Set IsSlave = TRUE if you are using this PID_Temp instance as slave controller in a cascade. A slave controller receives its setpoint (Setpoint parameter) from the output of its master controller (OutputHeat parame-ter). A PID_Temp instance can be master controller and slave controller at the same time.
Config.Cascade.AntiWindUp-Mode(1)	INT	1	Anti-windup behavior in the cascade Options are: • Anti-windup = 0 The AntiWindUp functionality is deactivated. The master controller does not respond to the limit of its slave controllers. • Anti-windup = 1 The integral action of the master controller is reduced in the ratio "Slaves in limit" to "Number of slaves" ("CountSlaves" parameter). This reduces the effects of the limitation on the control behavior. • Anti-windup = 2 The integral action of the master controller is held as soon as a slave controller is in the limit. Only effective if the controller is configured as master controller (Config.Cascade.IsMaster = TRUE).

Tag	Data type	Default	Description
Config.Cascade.CountSlaves(1)	INT	1	Number of subordinate slaves Here you enter the number of directly subordinate slave controllers which receive their setpoint from this master controller. Only effective if the controller is configured as master controller (Config.Cascade.IsMaster = TRUE). 255 ≥ CountSlaves ≥ 1
CycleTime.StartEstimation	BOOL	TRUE	If CycleTime.EnEstimation = TRUE, CycleTime.StartEstimation = TRUE starts automatic determination of the PID_Temp sampling time (cycle time of the calling OB). CycleTime.StartEstimation = FALSE is set once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If CycleTime.EnEstimation = TRUE, the PID_Temp sampling time is determined automatically. If CycleTime.EnEstimation = FALSE, the sampling time PID_Temp is not determined automatically and must be configured correctly manually with CycleTime.Value.
CycleTime.EnMonitoring	BOOL	TRUE	If CycleTime.EnMonitoring = FALSE, the PID_Temp sampling time is not monitored. If PID_Temp cannot be executed within the sampling time, no error (ErrorBits=0000800h) is output and PID_Temp does not respond as configured with ActivateRecoverMode.
CycleTime.Value(1)	REAL	0.1	PID_Temp sampling time (cycle time of the calling OB) in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.			
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser
CtrlParamsBackUp.Heat.Gain	REAL	1.0	Saved proportional gain for heating
CtrlParamsBackUp.Heat.Ti	REAL	20.0	Saved integral action time for heating in seconds
CtrlParamsBackUp.Heat.Td	REAL	0.0	Saved derivative action time for heating in seconds
CtrlParamsBackUp.Heat.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient for heating
CtrlParamsBackUp.Heat.PWeighting	REAL	1.0	Saved weighting of the proportional action for heating
CtrlParamsBackUp.Heat.DWeighting	REAL	1.0	Saved weighting of the derivative action for heating
CtrlParamsBackUp.Heat.Cycle	REAL	1.0	Saved sampling time of the PID algorithm for heating in seconds
CtrlParamsBackUp.Heat.ControlZone	REAL	3.402822e+38	Saved control zone width for heating
CtrlParamsBackUp.Heat.DeadZone	REAL	0.0	Saved dead zone width for heating
CtrlParamsBackUp.Cool.Gain	REAL	1.0	Saved proportional gain for cooling
CtrlParamsBackUp.Cool.Ti	REAL	20.0	Saved integral action time for cooling in seconds
CtrlParamsBackUp.Cool.Td	REAL	0.0	Saved derivative action time for cooling in seconds
CtrlParamsBackUp.Cool.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient for cooling

Tag	Data type	Default	Description
CtrlParamsBackUp.Cool.PWeighting	REAL	1.0	Saved proportional action weighting factor for cooling
CtrlParamsBackUp.Cool.DWeighting	REAL	1.0	Saved derivative action weighting factor for cooling
CtrlParamsBackUp.Cool.Cycle	REAL	1.0	Saved sampling time of the PID algorithm for cooling in seconds
CtrlParamsBackUp.Cool.ControlZone	REAL	3.402822e+38	Saved control zone width for cooling
CtrlParamsBackUp.Cool.DeadZone	REAL	0.0	Saved dead zone width for cooling
PIDSelfTune.SUT.CalculateParamsHeat	BOOL	FALSE	The properties of the heating branch of the controlled system are saved during pretuning for heating. If SUT.CalculateParamsHeat = TRUE, the PID parameters for heating (Retain.CtrlParams.Heat structure) are recalculated on the basis of these properties. This enables you to change the parameter calculation method (PIDSelfTune.SUT.TuneRuleHeat parameter) without having to repeat the tuning. SUT.CalculateParamsHeat is set to FALSE after the calculation. Only possible if the pretuning was successful (SUT.ProcParHeatOk = TRUE).
PIDSelfTune.SUT.CalculateParamsCool	BOOL	FALSE	The properties of the cooling branch of the controlled system are saved during tuning for cooling. If SUT.CalculateParamsCool = TRUE, the PID parameters for cooling (Retain.CtrlParams.Cool structure) are recalculated on the basis of these properties. This enables you to change the parameter calculation method (PIDSelfTune.SUT.TuneRuleCool parameter) without having to repeat the tuning. SUT.CalculateParamsCool is set to FALSE after the calculation. Only possible if the pretuning was successful (SUT.ProcParCoolOk = TRUE). Only effective if Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE.
PIDSelfTune.SUT.TuneRuleHeat	INT	2	Method for PID parameter calculation with pretuning for heating Options are: • SUT.TuneRuleHeat = 0: PID according to CHR • SUT.TuneRuleHeat = 1: PI according to CHR • SUT.TuneRuleHeat = 2: PID for temperature processes according to CHR (results in a slower and rather asymptomatic control response with lower overshoot than SUT.TuneRuleHeat = 0) (CHR = Chien, Hrones and Reswick) Only with SUT.TuneRuleHeat = 2 is the control zone Retain.CtrlParams.Heat.ControlZone automatically set during pretuning for heating.

Tag	Data type	Default	Description
PIDSelfTune.SUT.TuneRuleCool	INT	2	Method for PID parameter calculation with pretuning for cooling Options are: • SUT.TuneRuleCool = 0: PID according to CHR • SUT.TuneRuleCool = 1: PI according to CHR • SUT.TuneRuleCool = 2: PID for temperature processes according to CHR (results in a slower and rather asymptomatic control response with lower overshoot than SUT.TuneRuleCool = 0) (CHR = Chien, Hrones and Reswick) Only with SUT.TuneRuleCool = 2 is the control zone Retain.CtrlParams.Cool.ControlZone automatically set during pretuning for cooling. SUT.TuneRuleCool is only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE).
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning: • State = 0: Initialize pretuning • State = 100: Calculate standard deviation for heating • State = 200: Calculate standard deviation for cooling • State = 300: Determine point of inflection for heating • State = 400: Determine point of inflection for cooling • State = 500: Set heating to setpoint after reaching point of inflection • State = 600: Set cooling to setpoint after reaching point of inflection • State = 700: Compare efficiency of the heating actuator and cooling actuator • State = 800: Heating and cooling activated • State = 900: Cooling activated • State = 1000: Determine delay time after switching off heating • State = 9900: Pretuning successful • State = 1: Pretuning not successful
PIDSelfTune.SUT.ProcParHeatOk	BOOL	FALSE	TRUE: The calculation of the process parameters for pretuning heating was successful. This tag is set during tuning. It must be TRUE for calculation of the PID parameters for heating.
PIDSelfTune.SUT.ProcParCoolOk	BOOL	FALSE	TRUE: The calculation of the process parameters for pretuning cooling was successful. This tag is set during tuning. It must be TRUE for calculation of the PID parameters for cooling.

Tag	Data type	Default	Description
PIDSelfTune.SUT.AdaptDelay-Time	INT	0	The AdaptDelayTime tag determines the adaptation of the delay time for heating at the operating point (for "Pretuning heating" and "Pretuning heating and cooling"). Options are: • SUT.AdaptDelayTime = 0: No adaptation of delay time. The SUT.State = 1000 phase is skipped. This option results in a shorter tuning time than with SUT.AdaptDelayTime = 1. • SUT.AdaptDelayTime = 1: Adaptation of the delay time to the setpoint in SUT.State = 1000 phase by switching off heating temporarily. This option results in a longer tuning time than withSUT.AdaptDelayTime = 0. It can improve the control response if the process behavior depends significantly on the operating point (non-linearity). This option should not be used for multi-zone applications with strong thermal connections.
PIDSelfTune.SUT.CoolingMode	INT	0	The CoolingMode tag determines the manipulated variable output to determine the cooling parameters (for pretuning heating and cooling). Options are: • SUT.CoolingMode = 0: Switch off heating and switch on cooling after reaching the setpoint. The SUT.State = 700 phase is skipped. Phase SUT.State = 500 is followed by phase SUT.State = 900. This option can improve the control response if the gain of the cooling actuator is low compared to the gain of the heating actuator. It results in a shorter tuning time than with SUT.CoolingMode = 1 or 2. • SUT.CoolingMode = 1: Switch on cooling in addition to heating after reaching the setpoint. The SUT.State = 700 phase is skipped. Phase SUT.State = 500 is followed by phase SUT.State = 800. This option can improve the control response if the gain of the cooling actuator is high compared to the gain of the heating actuator. • SUT.CoolingMode = 2: After heating up to the setpoint, a decision is automatically made in phase SUT.State = 700 as to whether heating is switched off. Phase SUT.State = 500 is followed by phase SUT.State = 700 and then SUT.State = 800 or SUT.State = 900. This option requires more time than options 0 and 1.

Tag	Data type	Default	Description
PIDSelfTune.TIR.RunIn	BOOL	FALSE	Use the RunIn tag to specify the sequence of fine tuning during start from automatic mode. • RunIn = FALSE If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint (TIR.State = 500 or 600). Only then will fine tuning start. • RunIn = TRUE PID_Temp tries to reach the setpoint with minimum or maximum output value (TIR.State = 300 or 400). This can produce increased overshoot. Fine tuning then starts automatically. RunIn is set to FALSE after fine tuning. During start of fine tuning from Inactive or Manual mode, PID_Temp reacts as described under RunIn = TRUE.
PIDSelfTune.TIR.CalculateParam-sHeat	BOOL	FALSE	The properties of the heating branch of the controlled system are saved during fine tuning for heating. If TIR.CalculateParamsHeat= TRUE, the PID parameters for heating (Retain.CtrlParams.Heat structure) are recalculated on the basis of these properties. This enables you to change the parameter calculation method (PIDSelfTune.TIR.TuneRuleHeat parameter) without having to repeat the tuning. TIR.CalculateParamsHeat is set to FALSE after the calculation. Only possible if fine tuning heating was successful beforehand (TIR.ProcParHeatOk = TRUE).
PIDSelfTune.TIR.CalculateParamsCool	BOOL	FALSE	The properties of the cooling branch of the controlled system are saved during fine tuning for cooling. If TIR.CalculateParamsCool= TRUE, the PID parameters for cooling (Retain.CtrlParams.Cool structure) are recalculated on the basis of these properties. This enables you to change the parameter calculation method (PIDSelfTune.TIR.TuneRuleCool parameter) without having to repeat the tuning. TIR.CalculateParamsCool is set to FALSE after the calculation. Only possible if fine tuning cooling was successful beforehand (TIR.ProcParCoolOk = TRUE). Only effective if Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE

Tag	Data type	Default	Description
PIDSelfTune.TIR.TuneRuleHeat	INT	0	Method for parameter calculation during fine tuning for heating Options are: • TIR.TuneRuleHeat = 0: PID automatic • TIR.TuneRuleHeat = 1: PID fast (faster control response with higher amplitudes of the output value than with TIR.TuneRuleHeat = 2) • TIR.TuneRuleHeat = 2: PID slow (slower control response with lower amplitudes of the output value than with TIR.TuneRuleHeat = 1) • TIR.TuneRuleHeat = 3: ZN PID • TIR.TuneRuleHeat = 4: ZN PI • TIR.TuneRuleHeat = 5: ZN P (ZN=Ziegler-Nichols) To be able to repeat the calculation of the PID parameters for heating with TIR.CalculateParamsHeat and TIR.TuneRule-Heat = 0, 1 or 2, the previous fine tuning also has to have been executed with TIR.TuneRuleHeat = 0, 1 or 2. If this is not the case, TIR.TuneRuleHeat = 3 is used. The recalculation of the PID parameters for heating with TIR.CalculateParamsHeat and TIR.TuneRuleHeat = 3, 4 or 5 is always possible.
PIDSelfTune.TIR.TuneRuleCool	INT	0	Method for parameter calculation during fine tuning for cooling Options are: • TIR.TuneRuleCool = 0: PID automatic • TIR.TuneRuleCool = 1: PID fast (faster control response with higher amplitudes of the output value than with TIR.TuneRuleCool = 2) • TIR.TuneRuleCool = 2: PID slow (slower control response with lower amplitudes of the output value than with TIR.TuneRuleCool = 1) • TIR.TuneRuleCool = 3: ZN PID • TIR.TuneRuleCool = 4: ZN PI • TIR.TuneRuleCool = 5: ZN P (ZN=Ziegler-Nichols) To be able to repeat the calculation of the PID parameters for cooling with TIR.CalculateParamsCool and TIR.TuneRuleCool = 0, 1 or 2, the previous fine tuning also has to have been executed with TIR.TuneRuleCool = 0, 1 or 2. If this is not the case, TIR.TuneRuleCool = 3 is used. The recalculation of the PID parameters for cooling with TIR.CalculateParamsCool and TIR.TuneRuleCool = 3, 4 or 5 is always possible. Only effective if the cooling output and PID parameter switch-ing are activated (ConfigActivateCooling = TRUE and Con-fig.AdvancedCooling = TRUE).

Tag	Data type	Default	Description
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of "fine tuning": • State = 0: Initialize fine tuning • State = 100: Calculate standard deviation for heating • State = 200: Calculate standard deviation for cooling • State = 300: Attempting to reach setpoint for heating with two-step control using heating • State = 400: Attempting to reach setpoint for cooling with two-step control using cooling • State = 500: Attempting to reach setpoint for heating with PID control • State = 600: Attempting to reach setpoint for cooling with PID control • State = 700: Calculate standard deviation for heating • State = 800: Calculate standard deviation for cooling • State = 900: Determine oscillation and calculate parameters for heating • State = 1000: Determine oscillation and calculate parameters for cooling • State = 9900: Fine tuning successful • State = 1: Fine tuning not successful
PIDSelfTune.TIR.ProcParHeatOk	BOOL	FALSE	TRUE: The calculation of the process parameters for fine tuning heating was successful. This tag is set during tuning. It must be met for calculation of the PID parameters for heating.
PIDSelfTune.TIR.ProcParCoolOk	BOOL	FALSE	TRUE: The calculation of the process parameters for fine tuning cooling was successful. This tag is set during tuning. It must be met for calculation of the PID parameters for cooling.

Tag	Data type	Default	Description
PIDSelfTune.TIR.OutputOffset-tHeat	REAL	0.0	Tuning offset heating of the PID output value TIR.OutputOffsetHeat is added to the value that results from PidOutputSum for the heating branch. To receive a positive offset at the outputs for heating, define a positive value for TIR.OutputOffsetHeat. The resulting values at the outputs for heating are the result of the configured output scaling (Struktur Config.Output.Heat). This tuning offset can be used in controllers with activated cooling output and PID parameter switching (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE) for fine tuning cooling. If the outputs for cooling are not active at the setpoint that is to be tuned (PidOutputSum > 0.0), fine tuning cooling is not possible. In this case, define a positive tuning offset heating which is greater than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning. This step increases the values at the outputs for heating and activates the outputs for cooling (PidOutputSum < 0.0). Fine tuning cooling is now possible. When fine tuning is complete, TIR.OutputOffsetHeat is reset to 0.0. Major changes at TIR.OutputOffsetHeat in one step can result in temporary overshoots. Config.Output.Heat.PidUpperLimit ≥ PIDSelfTune.TIR.OutputOffsetHeat ≥ Config.Output.Heat.PidLowerLimit
PIDSelfTune.TIR.OutputOffset-Cool	REAL	0.0	Tuning offset cooling of the PID output value TIR.OutputOffsetCool is added to the value that results from PidOutputSum for the cooling branch. To receive a positive offset at the outputs for cooling, define a negative value for TIR.OutputOffsetCool. The resulting values at the outputs for cooling are the result of the configured output scaling (Struktur Config.Output.Coool). This tuning offset can be used in controllers with activated cooling output (Config.ActivateCooling = TRUE) for fine tuning heating. If the outputs for heating are not active at the setpoint that is to be tuned (PidOutputSum < 0.0), fine tuning heating is not possible. In this case, define a negative tuning offset cooling which is less than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning. This step increases the values at the outputs for cooling and activates the outputs for heating (PidOutputSum > 0.0). Fine tuning heating is now possible. When fine tuning is complete, TIR.OutputOffsetCool is reset to 0.0. Major changes at TIR.OutputOffsetCool in one step can result in temporary overshoots. Config.Output.Cool.PidUpperLimit ≥ PIDSelfTune.TIR.OutputOffsetCool ≥ Config.Output.Cool.PidLowerLimit

Tag	Data type	Default	Description
PIDSelfTune.TIR.WaitForControlIn	BOOL	FALSE	Waiting with fine tuning after reaching the setpoint If TIR.WaitForControlIn = TRUE, fine tuning waits in between reaching the setpoint (TIR.State = 500 or 600) and calculation of the standard deviation (TIR.State = 700 or 800) until a FALSE -> TRUE edge is given at TIR.FinishControlIn. TIR.WaitForControlIn can be used for simultaneous fine tuning of several controllers in multi-zone applications to synchronize tuning of the individual zones. It ensures that all zones have reached their setpoints before the actual tuning starts. The influence of thermal connections between the zones on tuning can be reduced in this way. TIR.WaitForControlIn is only effective if fine tuning is started from automatic mode with PIDSelfTune.TIR.RunIn = FALSE.
PIDSelfTune.TIR.ControlInReady	BOOL	FALSE	If TIR.WaitForControlIn = TRUE, PID_Temp sets TIR.ControlInReady = TRUE as soon as the setpoint has been reached and waits with additional tuning steps until a FALSE -> TRUE edge is given at TIR.FinishControlIn.
PIDSelfTune.TIR.FinishControlIn	BOOL	FALSE	If TIR.ControlInReady = TRUE, a FALSE -> TRUE edge at TIR.FinishControlIn stops the wait and fine tuning resumes.
PIDCtrl.IOutputOld(1)	REAL	0.0	Integral action in last cycle
PIDCtrl.PIDInit	BOOL	FALSE	PIDCtrl.PIDInit is available as of PID_Temp version 1.1. If PIDCtrl.PIDInit = TRUE in "Automatic mode", the integral action PIDCtrl.IOutputOld is pre-assigned automatically as if PidOutputSum = OverwriteInitialOutputValue in the previous cycle. This can be used for a AUTOHOTSPOT.
Retain.CtrlParams.SetByUser(1)	BOOL	FALSE	If the PID parameters are entered manually in the configuration editor, SetByUser = TRUE. This parameter is used for display in the editors and does not influence the control algorithm. SetByUser is retentive.
Retain.CtrlParams.Heat.Gain(1)	REAL	1.0	Active proportional gain for heating Heat.Gain is retentive. Heat.Gain ≥ 0.0
Retain..CtrlParams.Heat.Ti(1)	REAL	20.0	Active integral action time for heating in seconds The integral action for heating is switched off with Heat.CtrlParams.Ti = 0.0. Heat.Ti is retentive. 100000.0 ≥ Heat.Ti ≥ 0.0
Retain.CtrlParams.Heat.Td(1)	REAL	0.0	Active derivative action time for heating in seconds The derivative action for heating is switched off with Heat.CtrlParams.Td = 0.0. Heat.Td is retentive. 100000.0 ≥ Heat.Td ≥ 0.0

Tag	Data type	Default	Description
Retain.CtrlParams.Heat.TdFiltRatio(1)	REAL	0.2	Active derivative delay coefficient for heating The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient • 0.0: Derivative action is effective for one cycle only and therefore almost not effective. • 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant. • > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed. Heat.TdFiltRatio is retentive. Heat.TdFiltRatio ≥ 0.0
Retain.CtrlParams.Heat.PWeighting(1)	REAL	1.0	Active weighting of the proportional action for heating The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Proportional action for setpoint change is fully effective • 0.0: Proportional action for setpoint change is not effective The proportional action is always fully effective when the process value is changed. Heat.PWeighting is retentive. 1.0 ≥ Heat.PWeighting ≥ 0.0
Retain.CtrlParams.Heat.DWeighting(1)	REAL	1.0	Active weighting of the derivative action for heating The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Derivative action is fully effective upon setpoint change • 0.0: Derivative action is not effective upon setpoint change The derivative action is always fully effective when the process value is changed. Heat.DWeighting is retentive. 1.0 ≥ Heat.DWeighting ≥ 0.0
Retain.CtrlParams.Heat.Cycle(1)	REAL	1.0	Active sampling time of the PID algorithm for heating in seconds CtrlParams.Heat.Cycle is calculated during tuning and rounded to an integer multiple of CycleTime.Value. If Config.Output.Heat.PwmPeriode = 0.0, Heat.Cycle is used as period duration of the pulse width modulation for heating. If Config.Output.Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = FALSE, Heat.Cycle is used as period duration of the pulse width modulation for cooling. Heat.Cycle is retentive. 100000.0 ≥ Heat.Cycle > 0.0

Tag	Data type	Default	Description
Retain.CtrlParams.Heat.Control-Zone(1)	REAL	3.402822e +38	Active control zone width for heating The control zone for heating is switched off with Heat.Control-Zone = 3.402822e+38. Heat.ControlZone is only set automatically during pretuning heating or pretuning heating and cooling if PIDSelf-Tune.SUT.TuneRuleHeat = 2 is selected as method of the parameter calculation. For controllers with deactivated cooling output (Config.ActivateCooling = FALSE) or controllers with activated cooling output and cooling factor (Config.AdvancedCooling = FALSE), the control zone is symmetrically located between Setpoint – Heat.ControlZone and Setpoint + Heat.ControlZone. For controllers with activated cooling output and PID parameter switching (Config.ActivateCooling = TRUE, Config.Advanced-Cooling = TRUE), the control zone is located between Setpoint – Heat.ControlZone and Setpoint + Cool.ControlZone. Heat.ControlZone is retentive. Heat.ControlZone > 0.0
Retain.CtrlParams.Heat.Dead-Zone(1)	REAL	0.0	Active dead zone width for heating (see AUTOHOTSPOT) The dead zone for heating is switched off with Heat.DeadZone = 0.0. Heat.DeadZone is not set automatically or adjusted during tuning. You must correctly configure Heat.DeadZone manually. When the dead zone is switched on, the result can be a permanent control deviation (deviation between setpoint and process value). This can have a negative effect on fine tuning. For controllers with deactivated cooling output (Config.ActivateCooling = FALSE) or controllers with activated cooling output and cooling factor (Config.AdvancedCooling = FALSE), the dead zone is symmetrically located between Setpoint – Heat.DeadZone and Setpoint + Heat.DeadZone. For controllers with activated cooling output and PID parameter switching (Config.ActivateCooling = TRUE, Config.Advanced-Cooling = TRUE), the dead zone is located between Setpoint – Heat.DeadZone and Setpoint + Cool.DeadZone. Heat.DeadZone is retentive. Heat.DeadZone ≥ 0.0
Retain.CtrlParams.Cool.Gain(1)	REAL	1.0	Active proportional gain for cooling Cool.Gain is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). Cool.Gain ≥ 0.0

Tag	Data type	Default	Description
Retain.CtrlParams.Cool.Ti(1)	REAL	20.0	Active integral action time for cooling in seconds The integral action for cooling is switched off with Cool.CtrlParams.Ti = 0.0. Cool.Ti is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). 100000.0 ≥ Cool.Ti ≥ 0.0
Retain.CtrlParams.Cool.Td(1)	REAL	0.0	Active derivative action time for cooling in seconds The derivative action for cooling is switched off with Cool.CtrlParams.Td = 0.0. Cool.Td is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). 100000.0 ≥ Cool.Td ≥ 0.0
Retain.CtrlParams.Cool.TdFiltRatio(1)	REAL	0.2	Active derivative delay coefficient for cooling The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient • 0.0: Derivative action is effective for one cycle only and therefore almost not effective. • 0.5: This value has proved useful in practice for controlled systems with one dominant time constant. • > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed. Cool.TdFiltRatio is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). Cool.TdFiltRatio ≥ 0.0
Retain.CtrlParams.Cool.PWeighting(1)	REAL	1.0	Active weighting of the proportional action for cooling The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Proportional action for setpoint change is fully effective • 0.0: Proportional action for setpoint change is not effective The proportional action is always fully effective when the process value is changed. Cool.PWeighting is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). 1.0 ≥ Cool.PWeighting ≥ 0.0

Tag	Data type	Default	Description
Retain.CtrlParams.Cool.DWeighting[(1)]	REAL	1.0	Active weighting of the derivative action for cooling The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Derivative action is fully effective upon setpoint change • 0.0: Derivative action is not effective upon setpoint change The derivative action is always fully effective when the process value is changed. Cool.DWeighting is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). 1.0 ≥ Cool.DWeighting ≥ 0.0
Retain.CtrlParams.Cool.Cycle[(1)]	REAL	1.0	Active sampling time of the PID algorithm for cooling in seconds CtrlParams.Cool.Cycle is calculated during tuning and rounded off to an integer multiple of CycleTime.. If Config.Output.Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = TRUE, Cool.Cycle is used as period duration of the pulse width modulation for cooling. If Config.Output.Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = FALSE, Heat.Cycle is used as period duration of the pulse width modulation for cooling. Cool.Cycle is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). 100000.0 ≥ Cool.Cycle > 0.0

Tag	Data type	Default	Description
Retain.CtrlParams.Cool.Control-Zone(1)	REAL	3.402822e+38	Active control zone width for cooling The control zone for cooling is switched off with Cool.Control-Zone = 3.402822e+38. Cool.ControlZone is only set automatically during pretuning cooling or pretuning heating and cooling if PIDSelf-Tune.SUT.TuneRuleCool = 2 is selected as method of the parameter calculation. Cool.ControlZone is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). Cool.ControlZone > 0.0
Retain.CtrlParams.Cool.Dead-Zone(1)	REAL	0.0	Active dead zone width for cooling (see AUTOHOTSPOT) The dead zone for cooling is switched off with Cool.DeadZone = 0.0. Cool.DeadZone is not set automatically or adjusted during tuning. You must correctly configure Cool.DeadZone manually. When the dead zone is switched on, the result can be a permanent control deviation (deviation between setpoint and process value). This can have a negative effect on fine tuning. Cool.DeadZone is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). Cool.DeadZone ≥ 0.0

See also

PID_Temp ActivateRecoverMode tag (Page 2960)

PID_Temp Warning tag (Page 2963)

PID_Temp state and mode parameters (S7-1200, S7-1500)

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID_Temp switches to the operating mode saved in the Mode in-out parameter.

Heat.EnableTuning and Cool.EnableTuning specify for pretuning and fine tuning, if tuning takes place for heating or cooling.

If the CPU is switched on or switches from Stop to RUN mode, PID_Temp starts in the operating mode that is saved in the Mode parameter. To leave PID_Temp in "Inactive" mode, set RunModeByStartup = FALSE.

Meaning of values

State / Mode	Description of operating mode
0	**Inactive** The following output values are output in "Inactive" mode: • 0.0 as PID output value (PidOutputSum) • 0.0 as output value for heating (OutputHeat) and output value for cooling (OutputCool) • 0 as analog output value for heating (OutputHeat_PER) and analog output value for cooling (OutputCool_PER) • FALSE as PWM output value for heating (OutputHeat_PWM) and PWM output value for cooling (OutputCool_PWM) This does not depend on the configured output value limits and scaling in the structures Config.Output.Heat and Config.Output.Cool.

State / Mode	Description of operating mode
1	**Pretuning** The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning. PID_Temp offers different pretuning types depending on the configuration: • Pretuning heating: A jump change is output at the output value heating, the PID parameters for heating are calculated (Retain.CtrlParams.Heat structure), and control to the setpoint then takes place in automatic mode. If the process behavior strongly depends on the operating point, an adaptation of the delay time can be activated at the setpoint with PIDSelfTune.SUT.AdaptDelayTime. • Pretuning heating and cooling: A jump is output at the output value heating. As soon as the process value is close to the setpoint, a jump change is output at the output value cooling. The PID parameters for heating (Retain.CtrlParams.Heat structure) and cooling (Retain.CtrlParams.Cool structure) are calculated. Then, control to the setpoint takes place in automatic mode. If the process behavior strongly depends on the operating point, an adaptation of the delay time can be activated at the setpoint with PIDSelfTune.SUT.AdaptDelayTime. Depending on the effect of the cooling actuator compared to the heating actuator, the quality of tuning can be influenced by whether or not the heating and cooling outputs are operated simultaneously during tuning. You can specify this with PIDSelfTune.SUT.CoolingMode. • Pretuning cooling: A jump change is output at the output value cooling and the PID parameters for cooling are calculated (Struktur Retain.CtrlParams.Cool). Then, control to the setpoint takes place in automatic mode. If you want to tune the PID parameters for heating and cooling, you can expect a better control response with "Pretuning heating" followed by "Pretuning cooling" rather than with "Pretuning heating and cooling". However, carrying out pretuning in two steps takes more time. **General requirements for pretuning:** • The PID_Temp instruction is called in a cyclic interrupt OB. • Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) • ManualEnable = FALSE • Reset = FALSE • The setpoint and the process value lie within the configured limits. **Requirements for pretuning heating:** • Heat.EnableTuning = TRUE • Cool.EnableTuning = FALSE • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * |Config.InputUpperLimit - Config.InputLowerLimit| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint is greater than the process value.

State / Mode	Description of operating mode
	Setpoint > Input **Requirements for pretuning heating and cooling:** • Heat.EnableTuning = TRUE· • Cool.EnableTuning = TRUE • The cooling output is activated (Config.ActivateCooling = TRUE). • The PID parameter switching is activated (Config.AdvancedCooling = TRUE). • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * |Config.InputUpperLimit - Config.InputLowerLimit| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint is greater than the process value. Setpoint > Input **Requirements for pretuning cooling:** • Heat.EnableTuning = FALSE· • Cool.EnableTuning = TRUE· • The cooling output is activated (Config.ActivateCooling = TRUE). • The PID parameter switching is activated (Config.AdvancedCooling = TRUE). • A "pretuning heating" or "pretuning heating and cooling" has been successful (PIDSelfTune.SUT.ProcParHeatOk = TRUE), if possible at the same setpoint. • The process value must be close to the setpoint. |Setpoint - Input| < 0.05 * |Config.InputUpperLimit - Config.InputLowerLimit| The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" or "Manual mode". The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel The method for calculation of the PID parameters can be specified separately for heating and cooling with PIDSelfTune.SUT.TuneRuleHeat and PIDSelfTune.SUT.TuneRuleCool. Before the PID parameters are recalculated, they are backed up in the CtrlParamsBackUp structure and can be reactivated with LoadBackUp. After successful pretuning, the switch is made to automatic mode. After unsuccessful pretuning, the switch to the mode is determined by ActivateRecoverMode. The phase of pretuning is indicated with PIDSelfTune.SUT.State.

State / Mode	Description of operating mode
2	**Fine tuning** Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning. PID_Temp automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. PID_Temp offers different fine tuning types depending on the configuration: • Fine tuning heating: PID_Temp generates an oscillation of the process value with periodic changes at the output value heating and calculates the PID parameters for heating (Struktur Retain.CtrlParams.Heat). • Fine tuning cooling: PID_Temp generates an oscillation of the process value with periodic changes at the output value cooling and calculates the PID parameters for cooling (Struktur Retain.CtrlParams.Cool). **Temporary tuning offset for heating/cooling controllers** If PID_Temp is used as heating/cooling controller (Config.ActivateCooling = TRUE), the PID output value (PidOutputSum) at the setpoint must meet the following requirements for a process value oscillation to be generated and fine tuning to be successful: • Positive PID output value for fine tuning heating • Negative PID output value for fine tuning cooling If this requirement is not met, you can define a temporary offset for fine tuning which is output at the output with the opposite effect: • Offset for cooling output (PIDSelfTune.TIR.OutputOffsetCool) with fine tuning heating. Define a negative tuning offset cooling which is less than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning. • Offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) with fine tuning cooling. Define a positive tuning offset heating which is greater than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning. The defined offset is balanced by the PID algorithm so that the process value remains at the setpoint. The height of the offset allows the PID output value to be adapted correspondingly so that it fulfills the requirement mentioned above. To avoid larger overshoots of the process value when defining the offset, it can also be increased in several steps. If PID_Temp exits the fine tuning mode, the tuning offset is reset. Example for definition of an offset for fine tuning cooling: • Without offset: – Setpoint = Process value (ScaledInput) = 80°C – PID output value (PidOutputSum) = 30.0 – Output value heating (OutputHeat) = 30.0 – Output value cooling (OutputCool) = 0.0 Oscillation of the process value around the setpoint cannot be generated with the cooling output alone. Fine tuning would fail here. • With definition of an offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) = 80.0 – Setpoint = process value (ScaledInput) = 80°C – PID output value (PidOutputSum) = -50.0

State / Mode	Description of operating mode
	– Output value heating (OutputHeat) = 80.0 – Output value cooling (OutputCool) = -50.0 By defining an offset for the heating output, the cooling output can now create an oscillation of the process value around the setpoint. Fine tuning can now be carried out successfully. **General requirements for fine tuning:** • The PID_Temp instruction is called in a cyclic interrupt OB. • No disturbances are expected. • The setpoint and the process value lie within the configured limits. • The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint. When the dead zone is switched on, the result can be a permanent control deviation (deviation between setpoint and actual value). This can have a negative effect on fine tuning. • ManualEnable = FALSE • Reset = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode **Requirements for fine tuning heating:** • Heat.EnableTuning = TRUE • Cool.EnableTuning = FALSE • If PID_Temp is configured as heating/cooling controller (Config.ActivateCooling = TRUE), the heating output must be active at the operating point at which tuning is to take place (PidOutputSum > 0.0 (see tuning offset)). **Requirements for fine tuning cooling:** • Heat.EnableTuning = FALSE • Cool.EnableTuning = TRUE • The cooling output is activated (Config.ActivateCooling = TRUE). • The PID parameter switching is activated (Config.AdvancedCooling = TRUE) • The cooling output must be active at the operating point at which tuning is to take place (PidOutputSum < 0.0 (see tuning offset)). **The course of fine tuning is determined by the mode from which it is started:** • Automatic mode (State = 3) with PIDSelfTune.TIR.RunIn = FALSE (default) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_Temp controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) with PIDSelfTune.TIR.RunIn = TRUE Attempts are made to reach the setpoint with the minimum or maximum output value: – with minimum or maximum output value heating for fine tuning heating – With minimum or maximum output value cooling for fine tuning cooling. This can produce increased overshoot. Fine tuning starts when the setpoint is reached.

State / Mode	Description of operating mode
	If the setpoint cannot be reached, PID_Temp does not automatically abort tuning. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel The method for calculation of the PID parameters can be specified separately for heating and cooling with PIDSelfTune.TIR.TuneRuleHeat and PIDSelfTune.TIR.TuneRuleCool. Before the PID parameters are recalculated, they are backed up in the CtrlParamsBackUp structure and can be reactivated with LoadBackUp. The controller changes to automatic mode after successful fine tuning. After unsuccessful fine tuning, the switch to the mode is determined by ActivateRecoverMode. The "Fine tuning" phase is indicated with PIDSelfTune.TIR.State.
3	**Automatic mode** In automatic mode, PID_Temp corrects the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is met: • Pretuning successfully completed • Fine tuning successfully completed • Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate. The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor. The ActivateRecoverMode tag is taken into consideration in automatic mode.
4	**Manual mode** In manual mode, you specify a manual PID output value in the ManualValue parameter. The values at the outputs for heating and cooling resulting from this manual value are the result of the configured output scaling. You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only. The switchover from manual mode to automatic mode is bumpless. The ActivateRecoverMode tag is taken into consideration in manual mode.
5	**Substitute output value with error monitoring** The control algorithm is deactivated. The SetSubstituteOutput tag determines which PID output value (PidOutputSum) is output in this operating mode. • SetSubstituteOutput = FALSE: Last valid PID output value • SetSubstituteOutput = TRUE: Substitute output value (SubstituteOutput) You cannot activate this operating mode using Mode = 5. In the event of an error, it is activated instead of "Inactive" operating mode if all the following conditions are met: • Automatic mode (State = 3) • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode is effective. As soon as the errors are no longer pending, PID_Temp switches back to automatic mode.

ENO characteristics

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

Automatic switchover of operating mode during commissioning

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID_Temp automatically switches the operating mode in the event of an error.

The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. When you start pretuning or fine tuning, PID_Temp has saved the value of State in the Mode in-out parameter. This means PID_Temp switches to the mode from which tuning was started.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode.

See also

Output parameters of PID_Temp (Page 2916)

In/out parameters of PID_Temp V2 (Page 2919)

PID_Temp ErrorBits parameter (S7-1200, S7-1500)

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0000003h, for example, indicates that the errors 0000001h and 0000002h are pending simultaneously.

ErrorBits (DW#16#...)	Description
0000000	There is no error.
0000001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode. If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0000002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode. If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0000004	Error during fine tuning. Oscillation of the process value could not be maintained. If PID_Temp is used as heating-cooling controller (Config.ActivateCooling = TRUE), to be able to generate actual value oscillation, the PID output value (PidOutputSum) at the setpoint must be • positive for fine tuning heating • negative for fine tuning cooling If this requirement is not met, use the tuning offsets (PIDSelfTune.TIR.OutputOffsetCool and PIDSelf-Tune.TIR.OutputOffsetHeat tags), see AUTOHOTSPOT. If ActivateRecoverMode was = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0000008	Error at start of pretuning. The process value is too close to the setpoint or greater than the setpoint. Start fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0000010	The setpoint was changed during tuning. You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0000020	Pretuning is not permitted during fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp remains in fine tuning mode.
0000040	Error during pretuning. Cooling could not reduce the process value. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
0000100	Error during fine tuning resulted in invalid parameters. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0000200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode. If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0000400	Calculation of output value failed. Check the PID parameters. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0000800	Sampling time error: PID_Temp is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode. If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter. If this error occurred during simulation with PLCSIM, see the notes under AUTOHOTSPOT.
0001000	Invalid value at "Setpoint" parameter or "SubstituteSetpoint": Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode. If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.
0010000	Invalid value at ManualValue parameter. Value has an invalid number format. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp remains in manual mode and uses SubstituteOutput as PID output value. As soon as you specify a valid value in ManualValue, PID_Temp uses it as the PID output value.
0020000	Invalid value at SubstituteOutput tag. Value has an invalid number format. PID_Temp remains in the "Substitute output value with error monitoring" mode or manual mode and uses the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit) as PID output value. As soon as you specify a valid value in SubstituteOutput, PID_Temp uses it as the PID output value.

ErrorBits (DW#16#...)	Description
0040000	Invalid value at Disturbance parameter. Value has an invalid number format. If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_Temp remains in automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled.
0200000	Error in master in the cascade: Slaves are not in automatic mode or have activated substitute setpoint and prevent tuning of the master. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0400000	Pretuning heating is not permitted while cooling is active. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0800000	The process value must be close to the setpoint to start pretuning cooling. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
1000000	Error at start of tuning: Heat.EnableTuning and Cool.EnableTuning are not set or do not match the configuration. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
2000000	Pretuning cooling requires successful pretuning heating. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
4000000	Error at start of fine tuning: Heat.EnableTuning and Cool.EnableTuning must not be set simultaneously. If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
8000000	Error during calculation of the PID parameters resulted in invalid parameters. The invalid parameters are discarded and the original PID parameters are retained unchanged. We can distinguish between the following cases: • If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode. • If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode. • If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.

PID_Temp ActivateRecoverMode tag (S7-1200, S7-1500)

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

Automatic mode and manual mode

NOTICE
Your system may be damaged. If ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode or in manual mode even if there is an error and the process limit values are exceeded. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

ActivateRecoverMode	Description
FALSE	PID_Temp switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	**Automatic mode** If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_Temp switches between the calculated PID output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or several of the following errors occur and automatic mode was active before the error occurred, PID_Temp remains in automatic mode: • 0000001h: The "Input" parameter is outside the process value limits. • 0000800h: Sampling time error • 0040000h: Invalid value at Disturbance parameter. • 8000000h: Error during calculation of the PID parameters If one or several of the following errors occur and automatic mode was active before the error occurred, PID_Temp switches to "Substitute output value with error monitoring" mode: • 0000002h: Invalid value at Input_PER parameter. • 0000200h: Invalid value at Input parameter. • 0000400h: Calculation of output value failed. • 0001000h: Invalid value at Setpoint parameter or SubstituteSetpoint. As soon as the errors are no longer pending, PID_Temp switches back to automatic mode. If the following error occurs in "Substitute output value with error monitoring" mode, PID_Temp sets the PID output value to Config.Output.Heat.PidLowerLimit as long as this error is pending: • 0020000h: Invalid value at SubstituteOutput tag. Value has an invalid number format. This behavior is independent of SetSubstituteOutput.
	Manual mode If one or several errors occur and manual mode was active before the error occurred, PID_Temp remains in manual mode. If the following error occurs in manual mode, as long as this error is pending, PID_Temp sets the PID output value to SubstituteOutput: • 0010000h: Invalid value at ManualValue parameter. Value has an invalid number format. If the error 0010000h is pending in manual mode and the following error occurs, PID_Temp sets the PID output value to Config.Output.Heat.PidLowerLimit as long as this error is pending: • 0020000h: Invalid value at SubstituteOutput tag. Value has an invalid number format. This behavior is independent of SetSubstituteOutput.

Pretuning and fine tuning

ActivateRecoverMode	Description
FALSE	PID_Temp switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If the following error occurs, PID_Temp remains in the active mode: • 0000020h: Pretuning is not permitted during fine tuning. The following errors are ignored: • 0010000h: Invalid value at ManualValue parameter. • 0020000h: Invalid value at SubstituteOutput tag. When any other error occurs, PID_Temp cancels the tuning and switches to the mode from which tuning was started.

PID_Temp Warning tag (S7-1200, S7-1500)

If several warnings are pending simultaneously, the values of the Warning tag are displayed with binary addition. If the warning 0000003h is displayed, for example, the warnings 0000001h and 0000002h are pending simultaneously.

Warning (DW#16#....)	Description
0000000	No warning pending.
0000001	The point of inflection was not found during pretuning.
0000004	The setpoint was limited to the configured limits.
0000008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the method TIR.TuneRuleHeat = 3 or TIR.TuneRuleCool = 3.
0000010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0000020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0000040	The process value exceeded one of its warning limits.
0000080	Invalid value at Mode. The operating mode is not switched.
0000100	The manual value was limited to the limits of the PID output value.
0000200	The specified rule for tuning is not supported. No PID parameters are calculated.
0001000	The substitute output value cannot be reached because it is outside the output value limits.
0004000	The specified selection of the output value for heating and/or cooling is not supported. Only the output OutputHeat or OutputCool is used.
0008000	Invalid value at PIDSelfTune.SUT.AdaptDelayTime. The default value 0 is used.
0010000	Invalid value at PIDSelfTune.SUT.CoolingMode. The default value 0 is used.
0020000	The activation of cooling (Config.ActivateCooling tag) is not supported by the controller that is used as master (Config.Cascade.IsMaster tag). PID_Temp works as heating controller. Set the Config.ActivateCooling tag to FALSE.
0040000	Invalid value at Retain.CtrlParams.Heat.Gain, Retain.CtrlParams.Cool.Gain oder Config.CoolFactor. PID_Temp supports only positive values for proportional gain (heating and cooling) and cooling factor. Automatic mode remains active with PID output value 0.0. The integral component is stopped.

The following warnings are deleted as soon as the cause has been remedied or you repeat the action with valid parameters:

- 0000001h
- 0000004h
- 0000008h
- 0000040h
- 0000100h

All other warnings are cleared with a rising edge at Reset or ErrorAck.

PwmPeriode tag (S7-1200, S7-1500)

If the PID algorithm sampling time (Retain.CtrlParams.Heat.Cycle or Retain.CtrlParams.Heat.Cycle) and thus the time period of the pulse width modulation is very high when you use OutputHeat_PWM or OutputCool_PWM, you can define a deviating shorter time period at the Config.Output.Heat.PwmPeriode or Config.Output.Cool.PwmPeriode parameters to improve the smoothness of the process value.

Time period of the pulse width modulation at OutputHeat_PWM

Time period of the PWM at output OutputHeat_PWM depending on Config.Output.Heat.PwmPeriode:

- Heat.PwmPeriode = 0.0 (default)
  The sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as time period of the PWM.
- Heat.PwmPeriode > 0.0
  The value is rounded off to an integer multiple of the PID_Temp sampling time (CycleTime.Value) and used as time period of the PWM.
  The value must meet the following conditions:
  - Heat.PwmPeriode ≤ Retain.CtrlParams.Heat.Cycle
  - Heat.PwmPeriode > Config.Output.Heat.MinimumOnTime
  - Heat.PwmPeriode > Config.Output.Heat.MinimumOffTime

Time period of the pulse width modulation at OutputCool_PWM

Time period of the PWM at output OutputCool_PWM depending on Config.Output.Cool.PwmPeriode and the method for heating/cooling:

- Cool.PwmPeriode = 0.0 and cooling factor (Config.AdvancedCooling = FALSE):
  The sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as time period of the PWM.
- Cool.PwmPeriode = 0.0 and PID parameter switching (Config.AdvancedCooling = TRUE):
  The sampling time of the PID algorithm for cooling (Retain.CtrlParams.Cool.Cycle) is used as time period of the PWM.
- Cool.PwmPeriode > 0.0:
  The value is rounded off to an integer multiple of the PID_Temp sampling time (CycleTime.Value) and used as time period of the PWM.
  The value must meet the following conditions:
  - Cool.PwmPeriode ≤ Retain.CtrlParams.Cool.Cycle or Retain.CtrlParams.Heat.Cycle
  - Cool.PwmPeriode > Config.Output.Cool.MinimumOnTime
  - Cool.PwmPeriode > Config.Output.Cool.MinimumOffTime

Config.Output.Cool.PwmPeriode is only effective if the cooling output is activated (Config.ActivateCooling =TRUE).

When you use PwmPeriode, the accuracy of the PWM output signal is determined by the relationship of PwmPeriode to the PID_Temp sampling time (cycle time of the OB). PwmPeriode should be at least 10 times the PID_Temp sampling time.

If the sampling time of the PID algorithm is not an integer multiple of PwmPeriode, each last period of the PWM within the sampling time of the PID algorithm is extended accordingly.

Example for OutputHeat_PWM

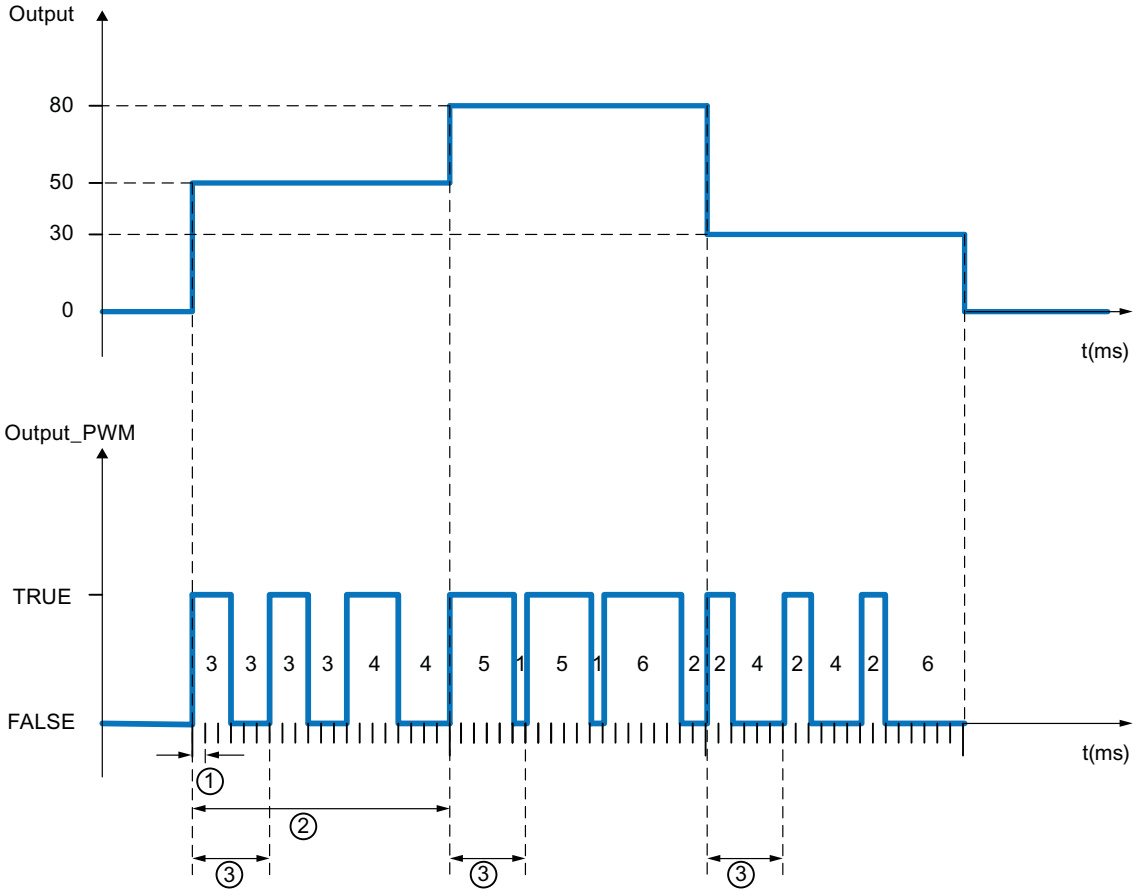

① PID_Temp sampling time = 100.0 ms (cycle time of the calling cyclic interrupt OB, CycleTime.Value tag)

② PID algorithm sampling time = 2000.0 ms (Retain.CtrlParams.Heat.Cycle tag)

③ Time period of the PWM for heating = 600.0 ms (Config.Output.Heat.PwmPeriode tag)

IntegralResetMode tag (S7-1200, S7-1500)

The IntegralResetMode tag determines how the integral action PIDCtrl.IOutputOld is pre-assigned:

- When switching from "Inactive" operating mode to "Automatic mode"
- With edge TRUE -> FALSE at parameter Reset and parameter Mode = 3

This setting only works for one cycle and is only effective if the integral action is activated (Retain.CtrlParams.Heat.Ti and Retain.CtrlParams.Cool.Ti > 0.0 tags).

IntegralReset-Mode	Description
0	Smooth The value of PIDCtrl.IOutputOld is pre-assigned so that the switchover is bumpless, which means "Automatic mode" starts with the output value = 0.0 (parameter PidOutputSum) and there is no jump of the output value regardless of the control deviation (setpoint – process value).
1	Delete We recommend setting the weighting of the proportional action (Retain.CtrlParams.Heat.PWeighting and Retain.CtrlParams.Cool.PWeighting tags) to 1.0 if this option is used. The value of PIDCtrl.IOutputOld is deleted. Any control deviation will cause a jump of the PID output value. The direction of the output value jump depends on the active weighting of the proportional action (Retain.CtrlParams.Heat.PWeighting and Retain.CtrlParams.Cool.PWeighting tags) and the control deviation: • Active proportional action weighting = 1.0: Output value jump and control deviation have identical signs. Example: If the process value value is smaller than the setpoint (positive control deviation), the PID output value jumps to a positive value. • Active proportional action weighting < 1.0: For large control deviations, the PID output value jump and control deviation have identical signs. Example: If the process value is much smaller than the setpoint (positive control deviation), the PID output value jumps to a positive value. For small control deviations, the PID output value jump and control deviation have different signs. Example: If the process value is just below the setpoint (positive control deviation), the PID output value jumps to a negative value. This is usually not desirable, because it results in a temporary increase in the control deviation. The smaller the configured weighting of the proportional action, the greater the control deviation must be to receive a PID output value jump with identical sign. We recommend setting the weighting of the proportional action (Retain.CtrlParams.Heat.PWeighting and Retain.CtrlParams.Cool.PWeighting tags) to 1.0 when this option is used. Otherwise, you may experience the undesirable behavior described for small control deviations. Alternatively, you can also use IntegralResetMode = 4. This option guarantees identical signs of the PID output value jump and control deviation independent of the configured weighting of the proportional action and the control deviation.
2	Hold The value of PIDCtrl.IOutputOld is not changed. You can define a new value using the user program.
3	Pre-assign The value of PIDCtrl.IOutputOld is automatically pre-assigned as if PidOutputSum = OverwriteInitialOutputValue in the last cycle.
4	Like setpoint change (only for PID_Temp with version ≥ 1.1) The value of PIDCtrl.IOutputOld is automatically pre-assigned so that a similar PID output value jump results as for a PI controller in automatic mode in case of a setpoint change from the current process value to the current setpoint. Any control deviation will cause a jump of the PID output value. The PID output value jump and control deviation have identical signs. Example: If the process value value is smaller than the setpoint (positive control deviation), the PID output value jumps to a positive value. This is independent of the configured weighting of the proportional action and the control deviation.

If IntegralResetMode is assigned a value outside the valid value range, PID_Temp behaves as with the pre-assignment of IntegralResetMode:

- PID_Temp up to V1.0: IntegralResetMode = 1
- PID_Temp as of V1.1: IntegralResetMode = 4

PID basic functions (S7-1500)

Differences compared to CONT_C S7-300/400 (S7-1500)

The CONT_C instruction is created with optimized block access.

All parameters are retentive. The retentivity cannot be changed. The actual values of retentive tags are only updated when you download CONT_C completely.

AUTOHOTSPOT

Otherwise, the behavior of CONT_C on an S7-1500 CPU is exactly the same as on an S7-300, S7-400 CPU.

Description CONT_C (Page 5560)

How CONT_C works (Page 5561)

CONT_C block diagram (Page 5563)

Input parameter CONT_C (Page 5564)

Output parameters CONT_C (Page 5566)

Differences compared to CONT_S S7-300/400 (S7-1500)

The CONT_S instruction is created with optimized block access. S

All parameters are retentive. The retentivity cannot be changed. The actual values of retentive tags are only updated when you download CONT_S completely.

AUTOHOTSPOT

Otherwise, the behavior of CONT_S on an S7-1500 CPU is exactly the same as on an S7-300, S7-400 CPU.

Description CONT_S (Page 5566)

Mode of operation CONT_S (Page 5567)

Block diagram CONT_S (Page 5569)

Input parameters CONT_S (Page 5570)

Output parameters CONT_S (Page 5571)

Differences compared to PULSEGEN S7-300/400 (S7-1500)

The PULSEGEN instruction is created with optimized block access.

All parameters are retentive. The retentivity cannot be changed. The actual values of retentive tags are only updated when you download PULSEGEN completely.

AUTOHOTSPOT

The behavior of PULSEGEN on an S7-1500 CPU is exactly the same as on an S7-300, S7-400 CPU.

Description PULSEGEN (Page 5571)

Mode of operation PULSEGEN (Page 5573)

Mode of operation PULSEGEN (Page 5576)

Three-step control (Page 5576)

Two-step control (Page 5578)

Input parameters PULSEGEN (Page 5580)

Output parameter PULSEGEN (Page 5581)

Differences compared to TCONT_CP S7-300/400 (S7-1500)

The TCONT_CP instruction is created with optimized block access.

All parameters are retentive. The retentivity cannot be changed. The actual values of retentive tags are only updated when you download TCONT_CP completely.

AUTOHOTSPOT

Otherwise, the behavior of TCONT_CP on an S7-1500 CPU is exactly the same as on an S7-300, S7-400 CPU.

Description TCONT_CP (Page 5581)

Mode of operation TCONT_CP (Page 5582)

Operating principle of the pulse generator (Page 5591)

Block diagram TCONT_CP (Page 5594)

Input parameters TCONT_CP (Page 5595)

Output parameters TCONT_CP (Page 5596)

In/out parameters TCONT_CP (Page 5597)

Static variables TCONT_CP (Page 5597)

Parameter STATUS_H (Page 5602)

Parameters STATUS_D (Page 5603)

Differences compared to TCONT_S S7-300/400 (S7-1500)

The TCONT_S instruction is created with optimized block access.

All parameters are retentive. The retentivity cannot be changed. The actual values of retentive tags are only updated when you download TCONT_S completely.

AUTOHOTSPOT

Otherwise, the behavior of TCONT_S on an S7-1500 CPU is exactly the same as on an S7-300, S7-400 CPU.

Description TCONT_S (Page 5603)

Mode of operation TCONT_S (Page 5604)
Block diagram TCONT_S (Page 5608)
Input paramters TCONT_S (Page 5609)
Output parameters TCONT_S (Page 5610)
In/out parameters TCONT_S (Page 5610)
Static variables TCONT_S (Page 5610)

Polyline (S7-1200, S7-1500)

Compatibility with CPU and FW (S7-1200, S7-1500)

The following table shows which version of Polyline can be used on which CPU:

CPU	FW	Polyline
S7-1200	V4.2 or higher	V1.0
S7-1500	V2.0 or higher	V1.0

Description Polyline (S7-1200, S7-1500)

Description

The Polyline instruction maps input value Input onto output value Output using a characteristic curve. The characteristic curve is defined as a polyline with maximum 50 points. Linear interpolation is performed between the points. You can fit the polyline to the desired characteristic curve using the number and configuration of the points.

The Polyline instruction can be used, for example, to linearize non-linear characteristics of sensor or actuators.

Interpolation calculation

Polyline calculates the output value at the Output parameter for the input value at the Input parameter that lies between point values x_i and x_{i+1} with a linear interpolation. The linear interpolation is calculated according to the following formula:

$$Output = \frac{(Input - x_i)}{(x_{i+1} - x_i)} (y_{i+1} - y_i) + y_i$$

With parameter Reset = TRUE, an alternative output value can also be specified using the SubstituteOutput parameter.

Polyline data

The value pairs for the polyline are contained in the Static area of the instruction.

Note

- The minimum number of value pairs to be configured is 2.
- The maximum number of value pairs to be configured is 50.
- For a valid configuration, the x values must be specified in ascending order.

To allow the polyline data to be changed without the changes taking effect immediately, the value pairs of the polyline are duplicated and contained in the following structures:

- UserData
  The polyline data in this structure can be edited.
  Make use of this structure to specify or change the polyline data. Changes in this structure do not affect the interpolation calculation until the check and duplication of the data to the WorkingData structure is initiated. This happens by setting Validate = TRUE or automatically during the first processing of Polyline after the change of operating state of the CPU from STOP to RUN.
  The preassignment of values in this structure does not represent a valid configuration. To use the values for the interpolation calculation, change the tags to valid values.
- WorkingData
  The polyline data in this structure cannot be edited. This data is used for the interpolation calculation. Do not manually change the data in this structure.

Both structures have the same data type and thus the same content:

- NumberOfUsedPoints
  Number of points used for the interpolation calculation.
- Point
  The array with 50 elements contains value pairs of points Point[i].x and Point[i].y with index "i" from 1 to 50.

The following figure shows a polyline with four points.

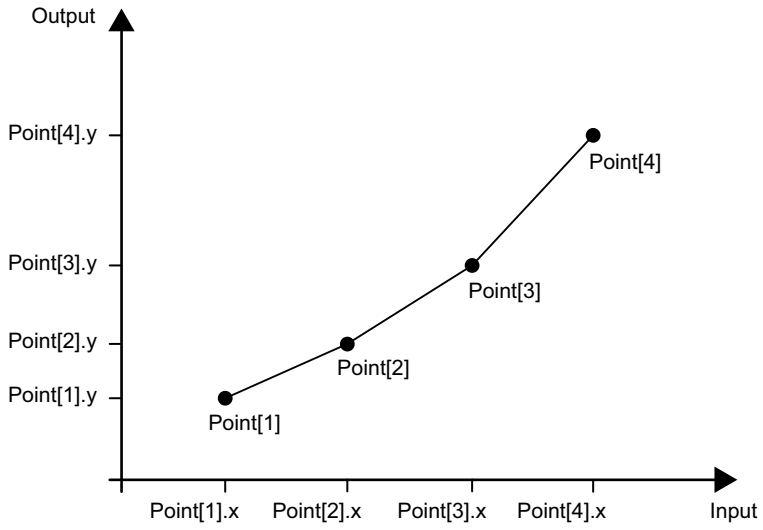

Call

Polyline is called as a single-instance DB in an OB. In an FB, Polyline can be called as both a single-instance DB and a multi-instance DB and as a parameter-instance DB.

When the instruction is called, no technology object is created. No parameter assignment interface or commissioning interface is available. You assign the Polyline parameters directly in the instance DB and commission Polyline using a watch table of the user program in the CPU or HMI.

Startup

The tags in the UserData and WorkingData structures are not retentive. These tags are initialized with the start values after each change of operating state of the CPU from STOP to RUN.

If you change the actual values in the UserData structure in online mode and these values are to be retained after the change of operating state of the CPU from STOP to RUN, back up these values in the start values of the data block.

At the first call of the Polyline instruction after the change of operating state of the CPU from STOP to RUN, the data in the UserData structure is automatically checked for validity. If the check is successful, the data is transferred to the WorkingData structure.

Reaction to error

The Polyline instruction detects different errors that may occur during interpolation calculation. The result of the interpolation calculation can be output at the output despite a pending error. If an error prevents correct calculation of the interpolation result, a substitute output value is output at the output.

You specify the substitute output value that is output if an error occurs that prevents correct calculation of the interpolation result as follows at the ErrorMode tag:

- If the ErrorMode tag is set to 0, Polyline outputs the input value at the Input parameter.
- If the ErrorMode tag is set to 1, Polyline outputs the substitute output value. You assign the substitute output value at the SubstituteOutput parameter.
- If the ErrorMode tag is set to 2, Polyline outputs the last valid output value at the Output parameter.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error is set to FALSE. The ErrorBits parameter shows which errors have occurred. ErrorBits is retentive and is reset only by a positive edge at the Reset or ErrorAck parameter.

Operating principle Polyline (S7-1200, S7-1500)

Polyline data

To change the polyline, you edit the values in the UserData structure. The values are then checked for validity and transferred to the WorkingData structure. Only in the WorkingData structure are the values used for the interpolation calculation.

The values are checked and transferred when

- You set the Validate parameter to TRUE while the Reset parameter is set to FALSE.
- Polyline is called for the first time after the change of operating state of the CPU from STOP to RUN while the Reset parameter is set to FALSE.
  If Polyline has already been called, for example in OB100, another automatic check of the values is not performed at the subsequent calls.

If the polyline data in the UserData structure is invalid, the previous polyline data in the WorkingData structure remains unchanged and a corresponding error message is output. If the check was performed for the first time, no valid values are available in the WorkingData structure and a corresponding error message is output. In this case, the Output parameter is specified with the substitute output value that you configure with the ErrorMode tag.

The check and transfer of the values from the UserData structure requires more CPU processing time than the interpolation calculation. In time-critical applications, the first execution of Polyline can be in startup OB 100. In this way, the time-consuming one-time check and transfer of the polyline data can be completed before the cyclic application program sections.

Validity of the polyline data

When the values in the UserData structure are checked, they must meet the following conditions so that a valid polyline is available for the interpolation calculation:

- 2 ≤ UserData.NumberOfUsedPoints ≤ 50
- UserData.Point[j].x < UserData.Point[j+1].x with index j = 1.. (UserData.NumberOfUsedPoints – 1)
- -3.402823e+38 ≤ UserData.Point[i].x ≤ 3.402823e+38 with index i = 1..UserData.NumberOfUsedPoints
- -3.402823e+38 ≤ UserData.Point[i].y ≤ 3.402823e+38 with index i = 1..UserData.NumberOfUsedPoints
- UserData.Point[i].x and UserData.Point[i].y are valid REAL values (≠ NaN) with index i = 1..UserData.NumberOfUsedPoints

If one or more conditions are not met during the check, the values in the UserData structure are not transferred to the WorkingData structure. A corresponding error message is output at the ErrorBits (Page 2977) parameter.

The preassignment of values in the UserData structure does not represent a valid configuration. Change the tags to valid values so that the tags can be used for the interpolation calculation.

Note

If more than the maximum number of 50 points are needed for your application, use two or more instances of Polyline.

Calculating the output value

If the input value at the Input parameter is below the first x-value or above the last x-value of the utilized points, configure the preassignment of the Output parameter with the following settings at the OutOfRangeMode tag:

- OutOfRangeMode = 0
  The output value is extrapolated with the slope of the first or last two points.

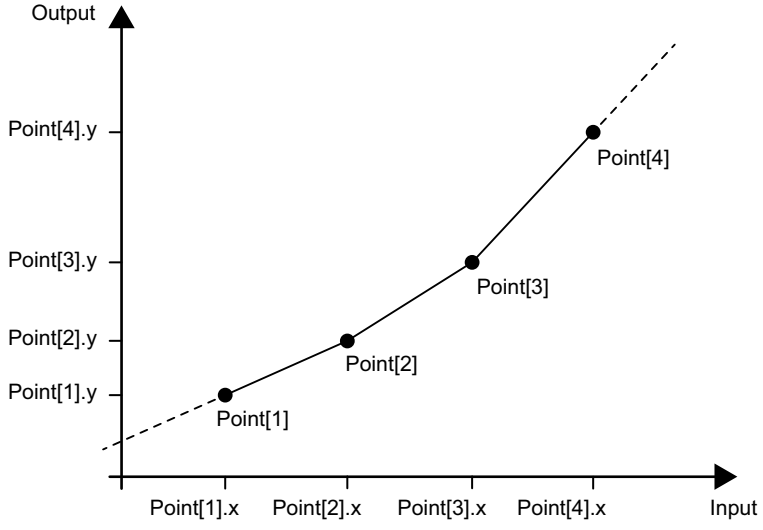

  If the OutOfRangeMode tag lies outside the permissible value range of 0 to 1, the default preassignment 0 becomes effective.

- OutOfRangeMode = 1
  The output value is limited to the y-value of the first or last point.

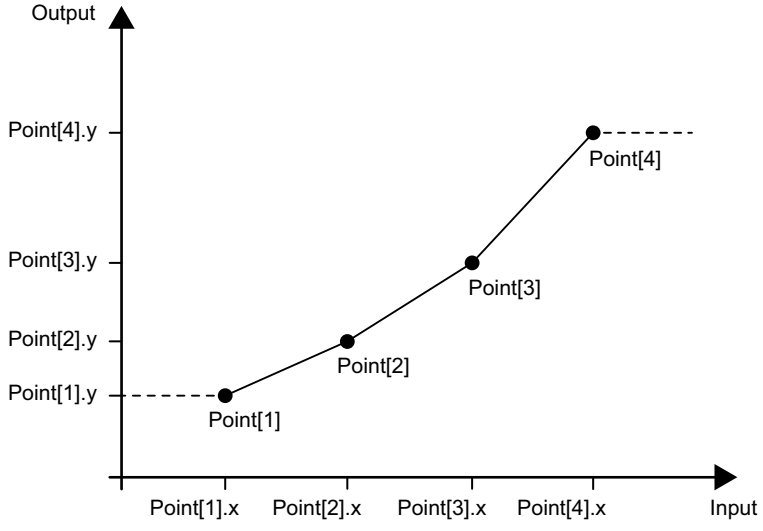

The Output parameter has a permissible value range of a REAL data type of -3.402823e+38 to 3.402823e+38. The output value at the Output parameter is checked for validity each time the Polyline instruction is executed. If the interpolation calculation yields an invalid REAL value, the output value is replaced with the setting at the ErrorMode tag.

Enable behavior EN/ENO

If one of the following conditions is met, enable output ENO is set to FALSE.

- Enable input EN is set to TRUE and the Output parameter is specified by a substitute output value in case of error messages ErrorBits ≥ 16#0001_0000.
- Enable input EN is set to FALSE.

If enable input EN is set to TRUE and the Output parameter can be determined as follows, enable output ENO is set to TRUE:

- With valid interpolation calculation when Reset = FALSE.

or

- With valid SubstituteOutput parameter when Reset = TRUE.

Currently utilized points

The NextXIndex index outputs the index of the next higher x-value for the current input value. You can use this to determine the points that are being used for the current interpolation calculation.

WorkingData.Point[NextXIndex-1].x < Input ≤ WorkingData.Point[NextXIndex].x

Example:

- If the value of the Input parameter is between WorkingData.Point[3].x and WorkingData.Point[4].x, the NextXIndex tag has the value 4.
- If the value of the Input parameter is less than WorkingData.Point[1].x, the NextXIndex tag has the value 1.
- If the value of the Input parameter is greater than WorkingData.Point[WorkingData.NumberOfUsedPoints].x and is thus greater than the last x-value of the polyline, the NextXIndex tag has the value of the WorkingData.NumberOfUsedPoints + 1 tag. Consequently, maximum permissible value of the NextXIndex tag is 51.

Input parameters of Polyline (S7-1200, S7-1500)

Parameter	Data type	Default	Description
Input	REAL	0.0	Input value
Substitu-teOutput	REAL	0.0	SubstituteOutput is used as the substitute output value when • Reset = TRUE or • An error with error message ErrorBits ≥ 16#0001_0000 prevents correct calculation of the interpolation result, and the configured value of ErrorMode is 1 .
Validate	BOOL	FALSE	If Validate is set to TRUE, the polyline data in UserData is checked for validity and transferred to WorkingData.

Parameter	Data type	Default	Description
ErrorAck	BOOL	FALSE	Deletes the error messages • Edge FALSE -> TRUE ErrorBits is reset
Reset	BOOL	FALSE	Performs a restart of the instruction • Edge FALSE -> TRUE ErrorBits is reset. • As long as Reset is set to TRUE, the substitute output value SubstituteOutput is output at the output. • As long as Reset is set to FALSE, the interpolation calculation is performed.

Output parameters of Polyline (S7-1200, S7-1500)

Parameter	Data type	Default	Description
Output	REAL	0.0	Output value
Error	BOOL	FALSE	When Error is set to TRUE, at least one error is currently pending.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 2977) shows which error messages are pending. ErrorBits is retentive and is reset at a positive edge at Reset or ErrorAck .

Static tags of Polyline (S7-1200, S7-1500)

Tag	Data type	Default	Description
UserData	AuxFct_PointTable	-	Input are for polyline data The polyline data in the UserData structure can be edited. Changes in this structure do not affect the interpolation calculation until the check and duplication of the data to the WorkingData structure is initiated.
UserData.NumberOfUsedPoints	INT	0	Number of points used for the interpolation calculation Permissible value range: 2 to 50
UserData.Point	Array[1..50] of AuxFct_Point	-	Points for the interpolation calculation The array with 50 elements of data type AuxFct_Point contains the value pairs of the points.
UserData.Point[i]	AuxFct_Point	-	Point for the interpolation calculation An element with index "i" from the "Point" array.
UserData.Point[i].x	REAL	0.0	x-value of the point Permissible value range: Point[i].x < Point[i+1].x
UserData.Point[i].y	REAL	0.0	y-value of the point
WorkingData	AuxFct_PointTable	-	Display area of the currently active polyline data The polyline data in the WorkingData structure cannot be edited. It is used for the interpolation calculation.

Tag	Data type	Default	Description
WorkingData.NumberOfUsed-Points	INT	0	Number of points used for the interpolation calculation Permissible value range: 2 to 50
WorkingData.Point	Array[1..50] of AuxFct_Point	-	Points for the interpolation calculation The array with 50 elements of type AuxFct_Point contains the value pairs of the points.
WorkingData.Point[i]	AuxFct_Point	-	Point for the interpolation calculation An element with index "i" from the "Point" array.
WorkingData.Point[i].x	REAL	0.0	x-value of the point Permissible value range: Point[i].x < Point[i+1].x
WorkingData.Point[i].y	REAL	0.0	y-value of the point
ErrorMode	INT	0	Selection of the substitute output value following an error • 0 = Input • 1 = SubstituteOutput • 2 = Last valid output value Permissible value range: 0 to 2
OutOfRangeMode	INT	0	Selection of the output value if the input value lies outside the defined x-values • 0 = Maintain slope • 1 = y-value of the first/last point Permissible value range: 0 to 1
NextXIndex	INT	2	Index of the next x-value Used for monitoring the index of the points that are being used for the current interpolation calculation. The following condition applies: WorkingData.Point[NextXIndex-1].x < Input ≤ WorkingData.Point[NextXIndex].x Do not change this value manually.

ErrorBits parameter (S7-1200, S7-1500)

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 16#0000_0003, for example, indicates that the errors 16#0000_0001 and 16#0000_0002 are pending simultaneously.

For Polyline the errors output at the ErrorBits parameter are divided into two categories:

- Errors with error messages ErrorBits < 16#0001_0000
- Errors with error messages ErrorBits ≥ 16#0001_0000

Errors with error messages ErrorBits < 16#0001_0000

If one or more errors with error messages ErrorBits < 16#0001_0000 are pending, enable output ENO is not changed. The output value can be determined as follows despite this error:

- Interpolation calculation when Reset = FALSE.

or

- Output of SubstituteOutput when Reset = TRUE.

ErrorBits (DW#16#...)	Description
0000_0000	No error is pending.
0000_0001	The Output parameter was limited to -3.402823e+38 or +3.402823e+38. If the interpolation value is output at the output (Reset = FALSE and ErrorBits < 16#0001_0000), check the following tags used in the interpolation calculation: • Input • WorkingData.Point[i].x • WorkingData.Point[i].y If a substitute output value is output at the Output parameter when Reset = FALSE and ErrorBits ≥ 16#0001_0000, check the Input or SubstituteOutput parameter, depending on the value set at the Error-Mode tag. If Reset = TRUE, check the SubstituteOutput parameter. If the Output parameter is no longer limited, the error that is stored at the Error parameter is cleared.
0000_0002	One or more tags in the UserData structure have invalid values but valid polyline data is available in the WorkingData structure. The polyline data in the UserData structure was checked because the Validate parameter is set to TRUE while the Reset parameter is simultaneously set to FALSE. The polyline data in the UserData structure is not transferred to the WorkingData structure so that the changes made in the UserData structure will not become effective. The FB Polyline continues the interpolation calculation with the unchanged values in the WorkingData structure. Ensure that the following conditions are met when the Validate parameter is set to TRUE: 1. 2 ≤ UserData.NumberOfUsedPoints ≤ 50 2. UserData.Point[j].x < UserData.Point[j+1].x with index j = 1..(UserData.NumberOfUsedPoints - 1) 3. -3.402823e+38 ≤ UserData.Point[i].x ≤ 3.402823e+38 with index i = 1..UserData.NumberOfUsedPoints 4. -3.402823e+38 ≤ UserData.Point[i].y ≤ 3.402823e+38 with index i = 1..UserData.NumberOfUsedPoints 5. UserData.Point[i].x and UserData.Point[i].y are valid REAL values (≠ NaN) with index i = 1..UserData.NumberOfUsedPoints The error bit at the Error parameter is deleted when • The Validate parameter is set to FALSE. or • The Reset parameter is set to TRUE or • The Validate parameter is set to TRUE while the Reset parameter is set to FALSE and the five conditions for UserData named above are met.

Errors with error messages ErrorBits ≥ 16#0001_0000

If one or more errors with error messages ErrorBits ≥ 16#0001_0000 are pending, the output value cannot be determined as expected. The substitute output value is output instead. Enable output ENO is set to FALSE.

ErrorBits (DW#16#...)	Description
0001_0000	The SubstituteOutput or Input parameter that is being used as the output value has no valid REAL value. The output is set to 0.0. Possible error scenarios: • The Reset parameter is set to FALSE and an error with ErrorBits ≥ 16#0002_0000 is pending that prevents the interpolation calculation, and – the Input parameter has no valid REAL value, then tag ErrorMode = 0. – the SubstituteOutput parameter has no valid REAL value, then tag ErrorMode = 1. • The Reset parameter is set to TRUE while the SubstituteOutput parameter has no valid REAL value. The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The SubstituteOutput or Input parameter is changed to a valid REAL value. or • The SubstituteOutput or Input parameter is no longer being used as the output value. If the SubstituteOutput or Input parameter is changed to a valid REAL value, FB Polyline uses the SubstituteOutput or Input parameter again as the output value.
0002_0000	The Input parameter has no valid REAL value, while the interpolation calculation is to be performed (Reset = FALSE). The Output parameter is assigned the substitute output value that you configure at the ErrorMode tag. Note that the output value is set to 0.0 when tag ErrorMode = 0 is set because the value at the Input parameter is not being used as a substitute output value in this case. The NextXIndex tag is not updated as long as the Input parameter has an invalid REAL value The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The Input parameter is changed to a valid REAL value. or • The Reset parameter is set to TRUE. If the Input parameter is changed to a valid REAL value and the Reset parameter is set to FALSE, FB Polyline outputs the interpolation value again at the output.

ErrorBits (DW#16#...)	Description
0004_0000	The interpolation calculation yields an invalid REAL value for the Output parameter. Check the REAL values in the WorkingData structure for validity. The Output parameter is assigned the substitute output value that you configure at the ErrorMode tag. If you want to change the polyline data, first edit the UserData structure and then set parameter Validate = TRUE. Do not manually change the data in WorkingData structure. The NextXIndex tag can have an invalid value as long as this error is pending. The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The interpolation calculation yields valid values for the Output parameter again. or • The Reset parameter is set to TRUE. If the interpolation calculation yields valid REAL values again and parameter Reset = FALSE is set, FB Polyline outputs the interpolation value again at the output.

ErrorBits (DW#16#...)	Description
0008_0000	One or more tags in the UserData structure have invalid values while no valid polyline data is available in the WorkingData structure. The polyline data in the UserData structure were checked because • The Validate parameter is set to TRUE while the Reset parameter is set to FALSE. or • When Polyline is executed for the first time after the change of operating state of the CPU from RUN to STOP , the Reset parameter is set to FALSE . The polyline data in the UserData structure is not transferred to the WorkingData structure so that the values in the UserData structure will not become effective. FB Polyline does not output the interpolation value at the Output parameter because no valid polyline data is contained in the WorkingData structure. The Output parameter is assigned the substitute output value that is configure at the ErrorMode tag. Ensure that the following conditions are met when the Validate parameter is set to TRUE or when the Reset parameter is set to FALSE while Polyline is being executed for the first time after the change of operating state of the CPU from STOP to RUN : 1. 2 ≤ UserData.NumberOfUsedPoints ≤ 50 2. UserData.Point[j].x < UserData.Point[j+1].x with index j = 1..(UserData.NumberOfUsedPoints – 1) 3. -3.402823e+38 ≤ UserData.Point[i].x ≤ 3.402823e+38 with index i = 1..UserData.NumberOfUsedPoints 4. -3.402823e+38 ≤ UserData.Point[i].y ≤ 3.402823e+38 with index i = 1..UserData.NumberOfUsedPoints 5. UserData.Point[i].x and UserData.Point[i].y are valid REAL values (≠ NaN) with index i = 1..UserData.NumberOfUsedPoints Note that all tags in the UserData and WorkingData structures are not retentive. These tags are initialized with the start values after each change of operating state of the CPU from STOP to RUN. The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The Validate parameter is set to TRUE while the Reset parameter is set to FALSE and the five conditions for UserData named above are met. or • When Polyline is executed for the first time after the change of operating state of the CPU from STOP to RUN, the Reset parameter is set to FALSE and the five conditions for UserData named above are met. or • The Reset parameter is set to TRUE. If the five conditions for UserData named above are met and the following conditions are met, FB Polyline outputs the interpolation value at the output again. • The Validate parameter is set to TRUE while the Reset parameter is set to FALSE. or • When Polyline is executed for the first time after the change of operating state of the CPU from RUN to STOP, the Reset parameter is set to FALSE.

SplitRange (S7-1200, S7-1500)

Compatibility with CPU and FW (S7-1200, S7-1500)

The following table shows which version of SplitRange can be used on which CPU:

CPU	FW	SplitRange
S7-1200	V4.2 or higher	V1.0
S7-1500	V2.0 or higher	V1.0

SplitRange description (S7-1200, S7-1500)

Description

The SplitRange instruction converts the input value into an output value. The input value is located in the value range that is limited by Points.x1 and Points.x2. The output value is located in the value range that is limited by Points.y1 and Points.y2.

The following figure shows the relevant characteristic of an example configuration of the SplitRange instruction:

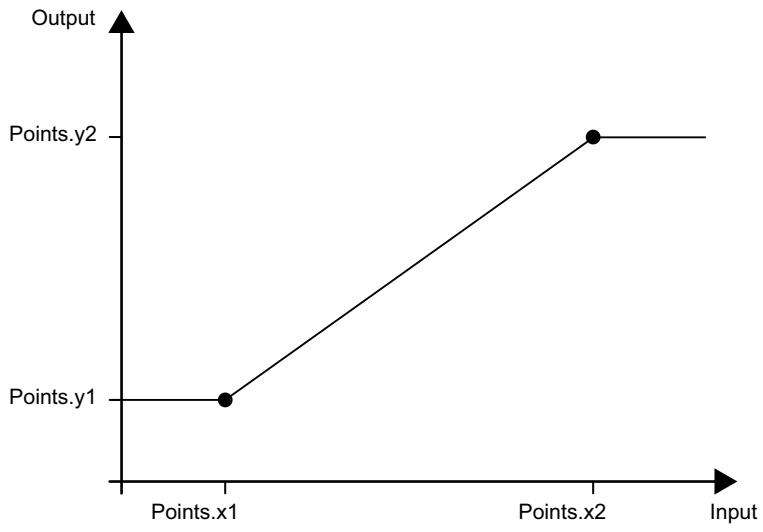

Use SplitRange when you need to control a process that is influenced by multiple actuators. SplitRange splits the output value range of the PID controller into multiple subranges. Assign a subrange to each actuator. The user program calls the block once per subrange. The input value of each SplitRange instance is connected to the output value of the PID controller.

The figure below shows an example of a control loop with two SplitRange instances and two actuators:

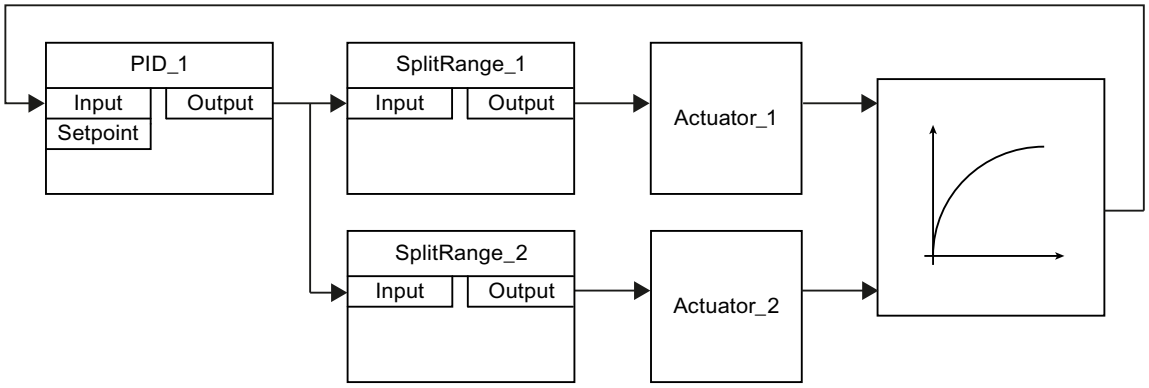

Validity of the SplitRange data

The value pairs in the Points structure define the input and output value range of SplitRange. The two value pairs are located in the static area of the block SplitRange.

SplitRange checks whether the following conditions are met for each call so that valid values are available for the calculation of the output value:

- Points.x1 < Points.x2
- Points.x1, Points.y1, Points.x2 and Points.y2 are within the permitted value range from -3.402823e+38 to 3.402823e+38
- Points.x1, Points.y1, Points.x2 and Points.y2 are valid REAL values (≠ NaN e.g. 16#7FFF_FFFF)

If one or more of these conditions are not met, correct calculation of the output value is not possible. A corresponding error message is output at the ErrorBits parameter.

The preassignment of the x and y values with 0.0 does not represent a valid configuration. Change the tags to valid values so that the tags can be used for the calculation of the output value.

Enable behavior EN/ENO

If one of the following conditions is met, enable output ENO is set to FALSE.

- Enable input EN is set to TRUE and the Output parameter is specified by a substitute output value in case of error messages ErrorBits ≥ 16#0001_0000.
- Enable input EN is set to FALSE.

If enable input EN is set to TRUE and the Output parameter can be determined as follows, enable output ENO is set to TRUE:

- With the valid calculation of the output value with Reset = FALSE.

or

- With valid SubstituteOutput parameter when Reset = TRUE.

Call

SplitRange is called as a single-instance DB in an OB. In an FB, SplitRange can be called as both a single-instance DB and a multi-instance DB and as a parameter-instance DB.

When the instruction is called, no technology object is created. No parameter assignment interface or commissioning interface is available. You assign the SplitRange parameters directly in the instance DB and commission SplitRange using a watch table of the user program in the CPU or HMI.

Startup

The tags in the static area of SplitRange are not retentive. These tags are initialized with the start values after each operating state transition of the CPU from STOP to RUN.

If you change the actual values in the Points structure in online mode and these values are to be retained after the operating state transition of the CPU from STOP to RUN, back up these values in the start values of the data block.

Reaction to error

The SplitRange instruction detects different errors that may occur during calculation of the output value. The result of the calculation can be output at the output despite a pending error. If an error prevents correct calculation of the output value, a substitute output value is output at the output.

You specify the substitute output value that is output if an error occurs that prevents correct calculation of the output value as follows at the ErrorMode tag:

- If the ErrorMode tag is set to 0, SplitRange outputs the input value at the Input parameter.
- If the ErrorMode tag is set to 1, SplitRange outputs the substitute output value. You assign the substitute output value at the SubstituteOutput parameter.
- If the ErrorMode tag is set to 2, SplitRange outputs the last valid output value at the Output parameter.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error is set to FALSE. The ErrorBits parameter shows which errors have occurred. ErrorBits is retentive and is reset only by a positive edge at the Reset or ErrorAck parameter.

SplitRange input parameters (S7-1200, S7-1500)

Parameter	Data type	Default	Description
Input	REAL	0.0	Input value
Substitu-teOutput	REAL	0.0	SubstituteOutput is used as the substitute output value when • Reset = TRUE or • An error with error message ErrorBits ≥ 16#0001_0000 prevents correct calculation of the output value, and the configured value of ErrorMode is 1.

Parameter	Data type	Default	Description
ErrorAck	BOOL	FALSE	Deletes the error messages • Edge FALSE -> TRUE ErrorBits is reset
Reset	BOOL	FALSE	Performs a restart of the instruction • Edge FALSE -> TRUE ErrorBits is reset. • As long as Reset is set to TRUE, the substitute output value SubstituteOutput is output at the output. • As long as Reset is set to FALSE, the calculation of the output value is performed.

SplitRange output parameters (S7-1200, S7-1500)

Parameter	Data type	Default	Description
Output	REAL	0.0	Output value
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 2986) shows which error messages are pending. ErrorBits is retentive and is reset at a positive edge at Reset or ErrorAck .
Error	BOOL	FALSE	When Error is set to TRUE, at least one error is currently pending.

SplitRange static tags (S7-1200, S7-1500)

Tag	Data type	Default	Description
Points	AuxFct_SplitRange_Points	-	Points data
Points.x1	REAL	0.0	x-value of point 1 Permissible value range: Points.x1 < Points.x2
Points.y1	REAL	0.0	y-value of point 1
Points.x2	REAL	0.0	x-value of point 2 Permissible value range: Points.x1 < Points.x2
Points.y2	REAL	0.0	y-value of point 2
ErrorMode	INT	0	Selection of the substitute output value following an error • 0 = Input • 1 = SubstituteOutput • 2 = Last valid output value Permissible value range: 0 to 2

ErrorBits parameter (S7-1200, S7-1500)

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 16#0000_0003, for example, indicates that the errors 16#0000_0001 and 16#0000_0002 are pending simultaneously.

For SplitRange, the errors output at the ErrorBits parameter are divided into two categories:

- Errors with error messages ErrorBits < 16#0001_0000
- Errors with error messages ErrorBits ≥ 16#0001_0000

Errors with error messages ErrorBits < 16#0001_0000

If one or more errors with error messages ErrorBits < 16#0001_0000 are pending, enable output ENO is not changed. The output value can be determined as follows despite this error:

- Calculation of output value based on the input value and the value pairs in the Points structure when Reset = FALSE.

Or

- Output of SubstituteOutput when Reset = TRUE.

ErrorBits (DW#16#...)	Description
0000_0000	No error is pending.
0000_0001	The Output parameter was limited to -3.402823e+38 or +3.402823e+38. If a substitute output value is output at the Output parameter when Reset = FALSE and ErrorBits ≥ 16#0001_0000, check the Input or SubstituteOutput parameter, depending on the value set at the Error-Mode tag. If Reset = TRUE, check the SubstituteOutput parameter. If the Output parameter is no longer limited, the error that is stored at the Error parameter is cleared.

Errors with error messages ErrorBits ≥ 16#0001_0000

If one or more errors with error messages ErrorBits ≥ 16#0001_0000 are pending, the output value cannot be determined as expected. The substitute output value is output instead. Enable output ENO is set to FALSE.

ErrorBits (DW#16#...)	Description
0001_0000	The SubstituteOutput or Input parameter that is being used as the output value has no valid REAL value. The output is set to 0.0. Possible error scenarios: • The Reset parameter is set to FALSE and an error with ErrorBits ≥ 16#0002_0000 is pending that prevents the calculation of the output value, and – the Input parameter has no valid REAL value, then tag ErrorMode = 0. – the SubstituteOutput parameter has no valid REAL value, then tag ErrorMode = 1. • The Reset parameter is set to TRUE while the SubstituteOutput parameter has no valid REAL value. The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The SubstituteOutput or Input parameter is changed to a valid REAL value. or • The SubstituteOutput or Input parameter is no longer being used as the output value. When the SubstituteOutput or Input parameter is changed to a valid REAL value, FB SplitRange uses the SubstituteOutput or Input parameter again as the output value.
0002_0000	The Input parameter has no valid REAL value while the calculation of the output value is being performed (Reset = FALSE). The Output parameter is assigned the substitute output value that you configure at the ErrorMode tag. Note that the output value is set to 0.0 when tag ErrorMode = 0 is set because the value at the Input parameter is not being used as a substitute output value in this case. The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The Input parameter is changed to a valid REAL value. or • The Reset parameter is set to TRUE. When the Input parameter is changed to a valid REAL value and the Reset parameter is set to FALSE, FB SplitRange outputs the calculated output value again at the output.

ErrorBits (DW#16#...)	Description
0004_0000	One or more tags in the Points structure have invalid values. Or the calculation of the output value yields an invalid REAL value for the Output parameter. The Output parameter is assigned the substitute output value that you configure at the ErrorMode tag. Ensure that the following conditions are met: 1. Points.x1 < Points.x2 2. Points.x1, Points.y1, Points.x2 and Points.y2 are within the permitted value range from -3.402823e +38 to 3.402823e+38 3. Points.x1, Points.y1, Points.x2 and Points.y2 are valid REAL values (≠ NaN e.g. 16#7FFF_FFFF) Note that all tags in the Points structure are not retentive. These tags are initialized with the start values after each operating state transition of the CPU from STOP to RUN. The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The three conditions listed above are met for the values in the Points structure. or • The Reset parameter is set to TRUE. When the three conditions listed above are met for the values in the Points structure and the Reset = FALSE parameter is set, FB SplitRange outputs the calculated output value again at the output.

RampFunction (S7-1200, S7-1500)

Compatibility with CPU and FW (S7-1200, S7-1500)

The following table shows which version of RampFunction can be used on which CPU:

CPU	FW	RampFunction
S7-1200	V4.2 or higher	V1.0
S7-1500	V2.0 or higher	V1.0

RampFunction description (S7-1200, S7-1500)

Description

The RampFunction instruction limits the slew rate of a signal. RampFunction outputs a signal jump at the input as ramp function of the output value.

Use the RampFunction to prevent signal jumps, for example, in the following cases:

- Between setpoint source and setpoint input of the controller to achieve a smoother response without influencing the disturbance reaction.
- Between the controller output and the actuator input to preserve the actuator, for example, a motor with gears or the process.

The following limits can be set for the slew rate:

- Increasing slew rate in positive value range
- Decreasing slew rate in positive value range
- Increasing slew rate in negative value range
- Decreasing slew rate in negative value range

In addition, the RampFunction instruction limits the output value to the high and low limit.

When the slew rate limit or the low or high limit are reached, RampFunction sets the associated output bit to TRUE.

Function chart

The following figure shows the RampFunction instruction and a function chart as an example:

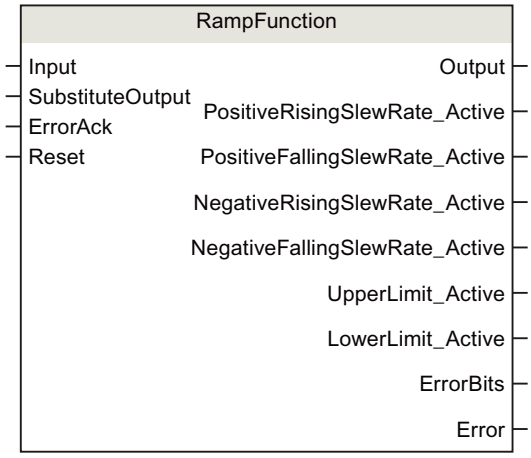

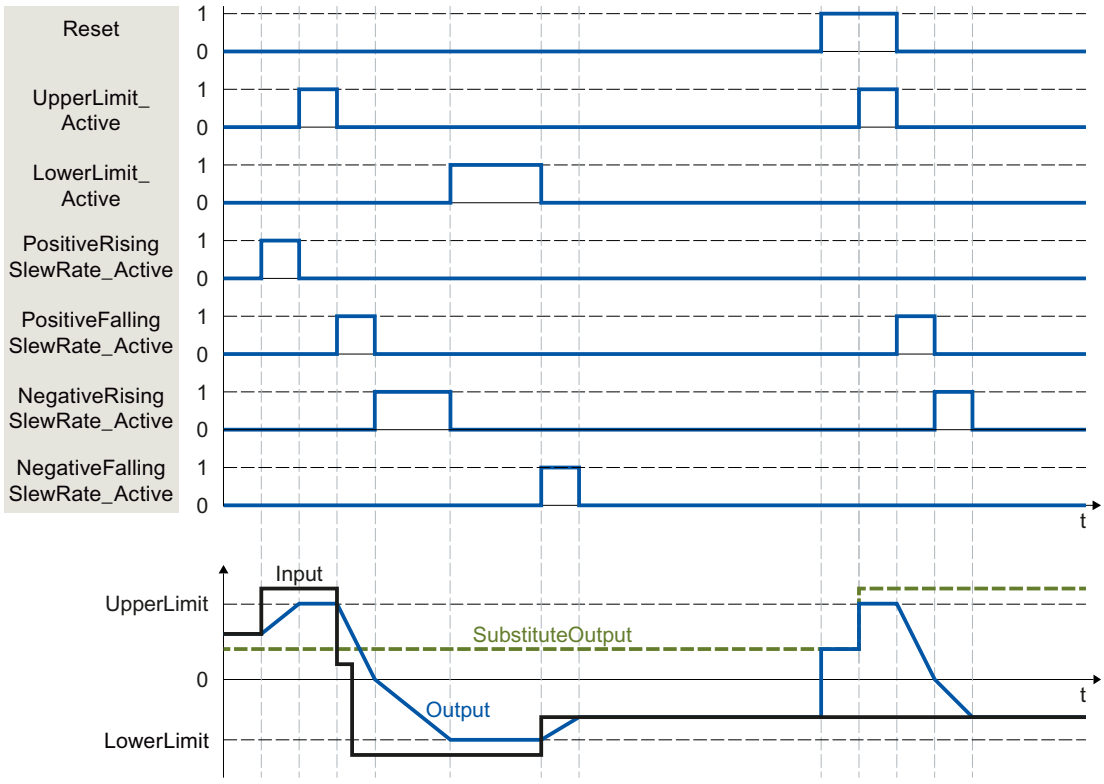

Call

RampFunction is called as a single-instance DB in an OB. In an FB, RampFunction can be called as both a single-instance DB and a multi-instance DB and as a parameter-instance DB.

When the instruction is called, no technology object is created. No parameter assignment interface or commissioning interface is available. You assign the RampFunction parameters

directly in the instance DB and commission RampFunction using a watch table of the user program in the CPU or HMI.

Startup

The tags in the static area of RampFunction are not retentive. These tags are initialized with the start values after each operating state transition of the CPU from STOP to RUN.

If you change the actual values of the limits in online mode and these values are to be retained after the operating state transition of the CPU, back up these values in the start values of the data block.

You specify the initialization value for the Output parameter at the StartMode tag. The initialization value is output at the Output parameter during the first call of RampFunction after the operating state transition of the CPU. For subsequent calls, RampFunction calculates the output value, starting from this initialization value, based on the input value and the slew rate limits.

The following table shows the dependency between the StartMode tag and the Output parameter. The values in the Output column are output at the Output parameter after the operating state transition of the CPU.

StartMode	Output	Example
0	Value of the Input parameter	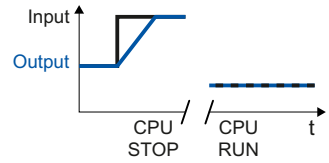
1	Value of the SubstituteOutput parameter	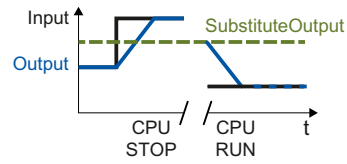
2	Remains unchanged. Output parameter is retentive.	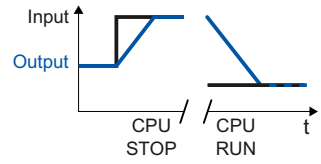
3	0.0	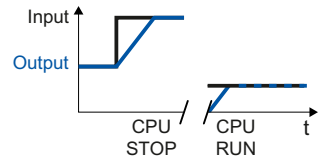

StartMode	Output	Example
4	Value of the LowerLimit tag	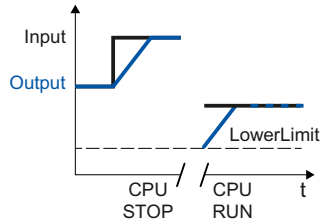
5	Value of the UpperLimit tag	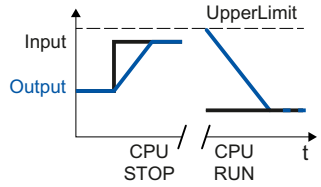

The following applies in addition for all values of the StartMode tag:

- When the values of the UpperLimit and LowerLimit tags are valid, the initialization value is limited to the value range of these tags. Only then is the initialization value output at the Output parameter.
- If the initialization value is invalid, the substitute output value is output at the Output parameter at the first call of the instruction after an operating state transition of the CPU. You configure the substitute output value at the ErrorMode tag. The substitute output value is limited by the value range of the tags UpperLimit and LowerLimit. For subsequent calls, the instruction calculates the output value starting from this substitute output value.
- The StartMode tag is only effective when the Reset = FALSE parameter is set at the first call of the instruction. If the Reset = TRUE parameter is set, the value of the SubstituteOutput parameter is output at the Output parameter.

Reaction to error

The RampFunction instruction detects different errors that may occur during calculation of the output value. The result of this calculation can be output at the output despite a pending error. If an error prevents correct calculation of the output value, a substitute output value is output at the output.

You specify the substitute output value that is output if an error occurs that prevents correct calculation of the output value at the ErrorMode tag.

The following table shows the dependency between the ErrorMode tag and the substitute output value that is output by the RampFunction at the Output parameter:

ErrorMode	Output
0	Value of the Input parameter
1	Value of the SubstituteOutput parameter
2	The last valid output value at the Output parameter
3	0.0
4	Value of the LowerLimit tag
5	Value of the UpperLimit tag

The following applies in addition for all values of the ErrorMode tag:

- If the substitute output value is not a valid REAL value, 0.0 is output as output value.
- When the values of the UpperLimit and LowerLimit tags are valid, the substitute output value is limited to the value range of these tags. Only then is the substitute output value output at the Output parameter.
- If an error is pending that prevents correct calculation of the output value, RampFunction changes at the Output parameter from the calculated output value to the substitute output value. A jump of the output value can occur, depending on the value of the ErrorMode tag.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error is set to FALSE. The ErrorBits parameter shows which errors have occurred. ErrorBits is retentive and is reset only by a positive edge at the Reset or ErrorAck parameter.

RampFunction mode of operation (S7-1200, S7-1500)

Limiting the slew rate

You can configure four limits for the slew rate of the input signal. The following factors determine which limit is currently in effect:

- Sign of the output value at the Output parameter
- Direction in which the amount of the output value changes at the Output parameter

The following table shows the effective tags for the slew rate limit depending on the Output parameter:

Output	Effective tag
Output ≥ 0 and |Output| rising	PositiveRisingSlewRate
Output ≥ 0 and |Output| falling	PositiveFallingSlewRate
Output < 0 and |Output| rising	NegativeRisingSlewRate
Output < 0 and |Output| falling	NegativeFallingSlewRate

The amount of the slew rate limits defines the maximum change of the output value per second.

Example:

The following scenario applies for the example:

- PositiveRisingSlewRate = 10.0
- Call time of RampFunction = 0.1 s
- Input > Output ≥ 0.0

Result:

The output value Output increases by 1.0 per call (10.0 per second) until the value at the Input parameter has been reached.

To disable the slew rate limit for one or more areas, set the corresponding tag to the value 3.402823e+38.

When the output value Output is currently limited by a slew rate limit, RampFunction sets the associated output bit to TRUE:

- PositiveRisingSlewRate_Active
- PositiveFallingSlewRate_Active
- NegativeRisingSlewRate_Active
- NegativeFallingSlewRate_Active

When the Reset parameter is set to TRUE, the slew rate limits are not in effect. This means jumps at the SubstituteOutput parameter result in jumps at the Output parameter.

RampFunction checks whether the following conditions for the tags PositiveRisingSlewRate, PositiveFallingSlewRate, NegativeRisingSlewRate and NegativeFallingSlewRate are met for each call:

- Values are within the permitted value range greater than 0.0 up to 3.402823e+38
- Values are valid REAL values (≠ NaN e.g. 16#7FFF_FFFF)

If one or more conditions are not met, the substitute output value is output at the Output parameter. A corresponding error message is output at the ErrorBits parameter.

Limiting the output value

The output value Output is always limited to the value range of the tags UpperLimit and LowerLimit as long as these tags have valid values.

When the output value Output is currently limited by this value range, RampFunction sets the associated output bit to TRUE:

- UpperLimit_Active
- LowerLimit_Active

The limit of the output value has a higher priority than the limit of the slew rate. Changes of the tags UpperLimit and LowerLimit therefore result in jumps of the output value Output, if this is required to observe the limits of the tags UpperLimit and LowerLimit. The limiting of the slew rate is not taken into account in this case.

Example:

If the UpperLimit is reduced from 100.0 to 80.0 while the values of the parameters Input and Output are 90.0, the output value Output jumps to 80.0. The output value Output jumps to 80.0 regardless whether or not it violates the configured limit for the slew rate.

RampFunction checks whether the following conditions are met for each call:

- LowerLimit < UpperLimit
- LowerLimit and UpperLimit are within the permitted value range from -3.402823e+38 to 3.402823e+38
- LowerLimit and UpperLimit are valid REAL values (≠ NaN e.g. 16#7FFF_FFFF)

If one or more conditions are not met, the substitute output value is output at the Output parameter. A corresponding error message is output at the ErrorBits parameter.

In addition, RampFunction checks for each call whether the output value Output has the permitted value range of a REAL data type from -3.402823e+38 to 3.402823e+38. If the

calculation of the output value yields an invalid REAL value, the substitute output value is output at the Output parameter. You configure the substitute output value at the ErrorMode tag.

Enable behavior EN/ENO

If one of the following conditions is met, enable output ENO is set to FALSE.

- Enable input EN is set to TRUE and the Output parameter is specified by a substitute output value in case of error messages ErrorBits ≥ 16#0001_0000.
- Enable input EN is set to FALSE.

If enable input EN is set to TRUE and the Output parameter can be determined as follows, enable output ENO is set to TRUE:

- With the signal limiting the slew rate at Reset = FALSE.

or

- With valid SubstituteOutput parameter when Reset = TRUE.

Measuring the cycle time automatically

To calculate the slew rate of the output value RampFunction needs the time that has expired since the last call of RampFunction.

The cycle time by default is measured automatically and output as of the second call at the CycleTime.Value tag. RampFunction measures the cycle time for each call of the instruction and can therefore be used in non-equidistant call cycles, e.g. in OB1.

If measurement of the cycle time has an invalid result, RampFunction calculates the current output value with the last valid cycle time. In addition, RampFunction outputs an error message at the ErrorBits parameter.

When you disable automatic measurement of the cycle time by setting the tag CycleTime.EnableMeasurement = FALSE, you must enter the cycle time manually at the CycleTime.Value tag. RampFunction checks the CycleTime.Value tag for validity at each call.

Automatic measurement of the cycle time with breakpoints

When breakpoints are active between two calls of RampFunction, automatic measurement of the cycle time results in the actual time that has expired between two calls. When one breakpoint is active, the CPU is in HOLD operating state.

Note

The active breakpoints extend the time period between two calls of RampFunction.

The longer the time period between two calls, the greater the maximum permitted change of the output value at the Output parameter.

Example:

The following scenario applies for the example:

- PositiveRisingSlewRate = 10.0
- Call time of RampFunction = 0.1 s
- Input > Output ≥ 0.0

Result without breakpoints:

The output value Output increases by 1.0 per call until the value at the Input parameter has been reached.

Result with an active breakpoint of ten seconds:

With the next call, the output value Output increases by 100.0.

If you do not need the calculation of the output value based on the actual time with active breakpoints, follow these steps:

- Disable automatic measurement of the cycle times by setting the tag CycleTime.EnableMeasurement = FALSE.
- Enter the cycle time manually at the CycleTime.Value tag.

RampFunction input parameters (S7-1200, S7-1500)

Parameter	Data type	Default	Description
Input	REAL	0.0	Input value
Substitu-teOutput	REAL	0.0	SubstituteOutput is used as the substitute output value when • Reset = TRUE or • An error with error message ErrorBits ≥ 16#0001_0000 prevents correct calculation of the output value, and the configured value of ErrorMode is 1.
ErrorAck	BOOL	FALSE	Deletes the error messages • Edge FALSE -> TRUE ErrorBits is reset
Reset	BOOL	FALSE	Performs a restart of the instruction • Edge FALSE -> TRUE ErrorBits is reset. • As long as Reset is set to TRUE, the substitute output value SubstituteOutput is output at the output. • As long as Reset is set to FALSE, the calculation of the output value is performed.

RampFunction output parameters (S7-1200, S7-1500)

Parameter	Data type	Default	Description
Output	REAL	0.0	Output value
PositiveRisingSlew-Rate_Active	BOOL	FALSE	When PositiveRisingSlewRate_Active = TRUE, the output value is currently limited by PositiveRisingSlewRate .

Parameter	Data type	Default	Description
PositiveFallingSlew-Rate_Active	BOOL	FALSE	When PositiveFallingSlewRate_Active = TRUE, the output value is currently limited by PositiveFallingSlewRate.
NegativeRising-SlewRate_Active	BOOL	FALSE	When NegativeRisingSlewRate_Active = TRUE, the output value is currently limited by NegativeRisingSlewRate.
NegativeFalling-SlewRate_Active	BOOL	FALSE	When NegativeFallingSlewRate_Active = TRUE, the output value is currently limited by NegativeFallingSlewRate.
UpperLimit_Active	BOOL	FALSE	When UpperLimit_Active = TRUE, the output value is currently limited by UpperLimit.
LowerLimit_Active	BOOL	FALSE	When LowerLimit_Active = TRUE, the output value is currently limited by LowerLimit.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 2999) shows which error messages are pending. ErrorBits is retentive and is reset at a positive edge at Reset or ErrorAck .
Error	BOOL	FALSE	When Error = TRUE, at least one error is currently pending.

RampFunction static tags (S7-1200, S7-1500)

Tag	Data type	Default	Description
PositiveRisingSlew-Rate	REAL	10.0	Limit for slew rate of the output value per second in positive range with rising amount With PositiveRisingSlewRate = 3.402823e+38, this slew rate limit is disabled. Permissible value range: > 0.0
PositiveFallingSlew-Rate	REAL	10.0	Limit for slew rate of the output value per second in positive range with falling amount With PositiveFallingSlewRate = 3.402823e+38, this slew rate limit is disabled. Permissible value range: > 0.0
NegativeRisingSlew-Rate	REAL	10.0	Limit for slew rate of the output value per second in negative range with rising amount With NegativeRisingSlewRate = 3.402823e+38, this slew rate limit is disabled. Permissible value range: > 0.0
NegativeFallingSlew-Rate	REAL	10.0	Limit for slew rate of the output value per second in negative range with falling amount With NegativeFallingSlewRate = 3.402823e+38, this slew rate limit is disabled. Permissible value range: > 0.0
UpperLimit	REAL	100.0	High limit of output value Permissible value range: > LowerLimit
LowerLimit	REAL	0.0	Low limit of output value Permissible value range: < UpperLimit

Tag	Data type	Default	Description
ErrorMode	INT	2	Selection of the substitute output value following an error • 0 = Input • 1 = SubstituteOutput • 2 = Last valid output value • 3 = 0.0 • 4 = LowerLimit • 5 = UpperLimit Permissible value range: 0 to 5
StartMode	INT	2	Selecting the output value for the first call of the instruction • 0 = Input • 1 = SubstituteOutput • 2 = Last output value • 3 = 0.0 • 4 = LowerLimit • 5 = UpperLimit Permissible value range: 0 to 5
CycleTime	AuxFct_Cycle-Time	-	Cycle time data
CycleTime.Value	REAL	0.1	Time interval between two calls of the instruction in seconds Permissible value range: > 0.0
CycleTime.Enable-Measurement	BOOL	TRUE	Enable automatic measurement of the cycle time

ErrorBits parameter (S7-1200, S7-1500)

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 16#0000_0003, for example, indicates that the errors 16#0000_0001 and 16#0000_0002 are pending simultaneously.

The RampFunction instruction divides errors that are output at the ErrorBits parameter in two categories:

- Errors with error messages ErrorBits < 16#0001_0000
- Errors with error messages ErrorBits ≥ 16#0001_0000

Errors with error messages ErrorBits < 16#0001_0000

If one or more errors with error messages ErrorBits < 16#0001_0000 are pending, enable output ENO is not changed. The output value can be determined as follows despite this error:

- Calculation of output value based on the input value and the limits for the slew rate when Reset = FALSE.

or

- Output of SubstituteOutput when Reset = TRUE.

ErrorBits (DW#16#...)	Description
0000_0000	No error is pending.
0000_0002	The measurement of the cycle time yields in an invalid value while the output value is being calculated (Reset = FALSE). If a valid value of the cycle time has already been measured, RampFunction calculates the output value based on the last value of the CycleTime.Value tag. If no valid value of the cycle time was previously measured, RampFunction still outputs the output value at the Output parameter that is configured with the StartMode tag. The error bit at the Error parameter is deleted when • The measurement of the cycle time yields a valid value or • The Reset parameter is set to TRUE

Errors with error messages ErrorBits ≥ 16#0001_0000

If one or more errors with error messages ErrorBits ≥ 16#0001_0000 are pending, the output value cannot be calculated as expected. The substitute output value is output instead. Enable output ENO is set to FALSE.

The output value limit remains active as long as the tags LowerLimit and UpperLimit have valid values.

The slew rate limit is no longer active. Jumps at the output value can occur in one of the following scenarios:

- When the error is detected, RampFunction switches from the calculated output value to the replacement output value. Whether a jump occurs depends on the value of the tag ErrorMode.
- The substitute output value is changed while it is active.

ErrorBits (DW#16#...)	Description
0001_0000	The SubstituteOutput or a different tag that is being used as output value has no valid REAL value. The output value at the Output parameter is therefore set to 0.0 and limited by the tags LowerLimit and UpperLimit. Possible error scenarios: • The Reset parameter is set to FALSE and an error with ErrorBits ≥ 16#0002_0000 is pending that prevents the calculation of the output value, and – the Input parameter has no valid REAL value, then tag ErrorMode = 0. – the SubstituteOutput parameter has no valid REAL value, then tag ErrorMode = 1. – the LowerLimit tag has no valid REAL value, then tag ErrorMode= 4. – the UpperLimit tag has no valid REAL value, then tag ErrorMode= 5. • The Reset parameter is set to TRUE while the SubstituteOutput parameter has no valid REAL value. The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The SubstituteOutput, Input, LowerLimit or UpperLimit parameter is changed to a valid REAL value. or • The SubstituteOutput, Input, LowerLimit or UpperLimit parameter is no longer being used as the output value. When the SubstituteOutput, Input, LowerLimit or UpperLimit parameter is changed to a valid REAL value, RampFunction outputs the value of this parameter again as the output value.
0002_0000	The Input parameter has no valid REAL value while the calculation of the output value is being performed (Reset = FALSE). The substitute output value is output at the Output parameter that is configured at the ErrorMode tag and is limited by the tags UpperLimit and LowerLimit. Note that the output value is set to 0.0 when tag ErrorMode = 0 is set because the value at the Input parameter is not being used as a substitute output value in this case. The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The Input parameter is changed to a valid REAL value. or • The Reset parameter is set to TRUE. When the Input parameter is changed to a valid REAL value and the Reset parameter is set to FALSE, RampFunction outputs the calculated output value again at the Output parameter.
0004_0000	The calculation of the output value yields an invalid REAL value at the Output parameter. The substitute output value is output at the Output parameter that is configured at the ErrorMode tag and is limited by the tags UpperLimit and LowerLimit. The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The calculation of the output value outputs valid values at the Output parameter again. or • The Reset parameter is set to TRUE. When the calculation of the output value yields valid REAL values again and the Reset parameter is set to FALSE, RampFunction outputs the calculated output value again at the Output parameter.

ErrorBits (DW#16#...)	Description
0008_0000	The LowerLimit or UpperLimit tag has an invalid value. The following value is output at the Output parameter, depending on the Reset parameter: • Reset = FALSE The substitute output value that is configured with the ErrorMode tag is output at the Output parameter. • Reset = TRUE The value of the SubstituteOutput parameter is output at the Output parameter. In both cases, the Ouput parameter is limited to the value range of the REAL data type from -3.402823e+38 to 3.402823e+38. Ensure that the following conditions are met: 1. LowerLimit < UpperLimit 2. LowerLimit and UpperLimit are within the permitted value range from -3.402823e+38 to 3.402823e+38 3. LowerLimit and UpperLimit are valid REAL values (≠ NaN e.g. 16#7FFF_FFFF) Note that the output value is set to 0.0 at the Output parameter when Reset = FALSE and • the LowerLimit tag has no valid REAL value, the tag ErrorMode = 4 or • the UpperLimit tag has no valid REAL value, then tag ErrorMode = 5. In this case, the values of the tags LowerLimit and UpperLimit are not output as substitute output values at the Output parameter because they are invalid. When the three conditions listed above for the LowerLimit and UpperLimit tags are met, the following applies: • The error bit at the Error parameter is cleared. • Enable output ENO is set to TRUE again. • RampFunction outputs the output value at the Output parameter again with – Reset = FALSE, by calculating and limiting the output value. – Reset = TRUE due to the limited value at the SubstituteOutput. parameter.
0010_0000	At least one of the following parameters has invalid values while the calculation of the output value is being performed (Reset = FALSE): 1. PositiveRisingSlewRate 2. PositiveFallingSlewRate 3. NegativeRisingSlewRate 4. NegativeFallingSlewRate The substitute output value is output at the Output parameter that is configured at the ErrorMode tag and is limited by the tags UpperLimit and LowerLimit. Ensure that the following conditions are met for all four parameters listed above: • The values are within the permitted value range greater than 0.0 up to 3.402823e+38 • The values are valid REAL values (≠ NaN e.g. 16#7FFF_FFFF) The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • The conditions listed above are met for the four parameters or • The Reset parameter is set to TRUE. When the conditions listed above are met for the four parameters and Reset is set to FALSE, RampFunction outputs the calculated output value at the Output parameter again.

ErrorBits (DW#16#...)	Description
0020_0000	The tag (configured with StartMode) for the initialization of the Output parameter at the first call of the instruction does not have a valid REAL value. The substitute output value is output with the first call of the instruction at the Output parameter that is configured at the ErrorMode tag and is limited by the tags LowerLimit and UpperLimit. For subsequent calls, RampFunction calculates the output value starting from this substitute output value. This error occurs at the first call of the instruction after the operating state transition of the CPU from STOP to RUN when Reset = FALSE and • The SubstituteOutput parameter has no valid REAL value with StartMode= 1. • The Output parameter has no valid REAL value with StartMode = 2. The following applies for the subsequent calls: • The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again. • RampFunction outputs the calculated output value at the Output parameter again.
0040_0000	The CycleTime.Value tag has an invalid value, while the calculation of the output value is being performed (Reset = FALSE). The substitute output value is output at the Output parameter that is configured at the ErrorMode tag and is limited by the tags UpperLimit and LowerLimit. Ensure that the following conditions are met: • 0.0 < CycleTime.Value < 3.402823e+38 • CycleTime.Value is a valid REAL value (≠ NaN e.g. 16#7FFF_FFFF) To automatically calculate the value of the CycleTime.Value tag, set the CycleTime.EnableMeasurement tag to TRUE. The error bit at the Error parameter is cleared and enable output ENO is set to TRUE again, when • the conditions listed above are met for the CycleTime.Value tag or • The Reset parameter is set to TRUE. When the conditions listed above are met for the CycleTime.Value tag and Reset is set to FALSE, RampFunction outputs the calculated output value at the Output parameter again.

4.1.5.4 Motion Control (S7-1200, S7-1500, S7-1500T)

S7-1200 Motion Control (S7-1200)

S7-1200 Motion Control as of V6 (S7-1200)

MC_Power (S7-1200)

MC_Power: Enable, disable axis as of V6 (S7-1200)

Description

The Motion Control instruction "MC_Power" enables or disables an axis.

Requirements

- The positioning axis technology object has been configured correctly.
- There is no pending enable-inhibiting error.

Override response

Execution of "MC_Power" cannot be aborted by a Motion Control command.

Disabling the axis (input parameter "Enable" = FALSE) aborts all Motion Control commands for the associated technology object in accordance with the selected "StopMode".

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Axis technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The axis is enabled.
				FALSE	All current jobs are interrupted in accordance with the "StopMode" configured. The axis is stopped and disabled.
StartMode	INPUT	INT	1	0	Enable positioning axis not position-controlled *)
				1	Enable positioning axis position-controlled *)
				*) This parameter is ignored when a positioning axis with PTO (Pulse Train Output) drive is used. The parameter initially takes effect when the positioning axis is enabled (Enable changes from FALSE to TRUE) and when the axis is enabled after successful acknowledgment of an interrupt that caused the axis to be disabled.	

Parameter	Declaration	Data type	Default value	Description	
StopMode	INPUT	INT	0	0	Emergency stop If a request to disable the axis is pending, the axis brakes at the configured emergency deceleration. The axis is disabled after reaching standstill.
				1	Immediate stop If a request to disable the axis is pending, this setpoint zero is output and the axis is disabled. The axis is braked depending on the configuration in the drive, and is brought to a standstill. With drive connection via PTO (Pulse Train Output): When you disable the axis, the pulse output is stopped with a frequency-dependent deceleration: • Output frequency ≥ 100 Hz Deceleration: max. 30 ms • Output frequency < 100 Hz Deceleration: 30 ms up to max. 1.5 s at 2 Hz
				2	Emergency stop with jerk control If a request to disable the axis is pending, the axis brakes at the configured emergency deceleration. If the jerk control is activated, the configured jerk is taken into account. The axis is disabled after reaching standstill.
Status	OUTPUT	BOOL	FALSE	Status of axis enable	
				FALSE	The axis is disabled. The axis does not execute Motion Control commands and does not accept any new commands (exception: MC_Reset command) For drive connection via PTO (Pulse Train Output): The axis is not homed. Upon disabling, the status does not change to FALSE until the axis reaches a standstill.
				TRUE	The axis is enabled. The axis is ready to execute Motion Control commands. Upon axis enabling, the status does not change to TRUE until the signal "Drive ready" is pending. If the "Drive ready" drive interface was not configured in the axis configuration, the status changes to TRUE immediately.
Busy	OUTPUT	BOOL	FALSE	TRUE	MC_Power" is active.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction "MC_Power" or in the associated technology object. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Note

If the axis is switched off due to an error, it will be enabled again automatically after the error has been eliminated and acknowledged. This requires that the input parameter "Enable" has retained the value TRUE during this process.

Enabling an axis with configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameters "StartMode" and "StopMode" with the desired value. Set the input parameter "Enable" to TRUE.
   The enable output for "Drive enabled" changes to TRUE to enable the power to the drive. The CPU waits for the "Drive ready" signal of the drive.
   When the "Drive ready" signal is available at the configured ready input of the CPU, the axis is enabled. The output parameter "Status" and the variable of the technology object <axis name>.StatusBits.Enable indicate the value TRUE.

Enabling an axis without configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameters "StartMode" and "StopMode" with the desired value. Set the input parameter "Enable" to TRUE. The axis is enabled. The output parameter "Status" and the variable of the technology object <axis name>.StatusBits.Enable indicate the value TRUE.

Disabling an axis

To disable an axis, you can follow the steps described below:

1. Bring the axis to a standstill.
   You can identify when the axis is at a standstill in the variable of the technology object <axis name>.StatusBits.StandStill.
2. Set input parameter "Enable" to FALSE after standstill is reached.
3. If output parameters "Busy" and "Status" and variable of technology object <axis name>.StatusBits.Enable indicate the value FALSE, disabling of the axis is complete.

See also

MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)

MC_Home: Home axes, set reference point as of V6 (Page 3010)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)

MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)

MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)

MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)
MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)
MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)
MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)
MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)
MC_Halt: Stop axis as of V6 (Page 3015)

MC_Power: Function chart as of V6 (S7-1200)

Function chart

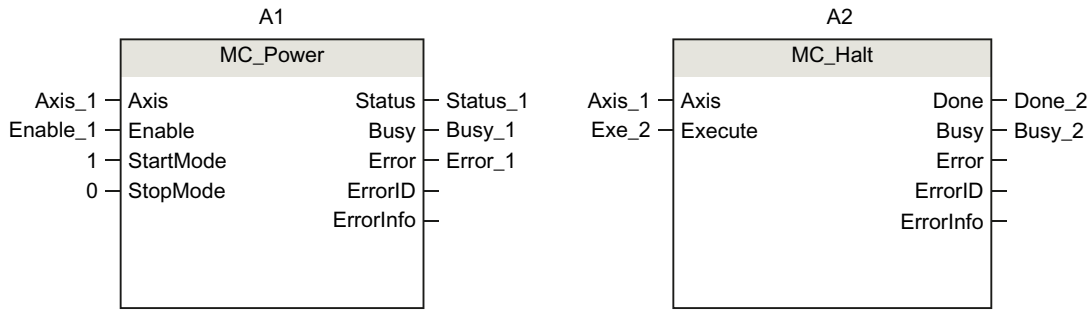

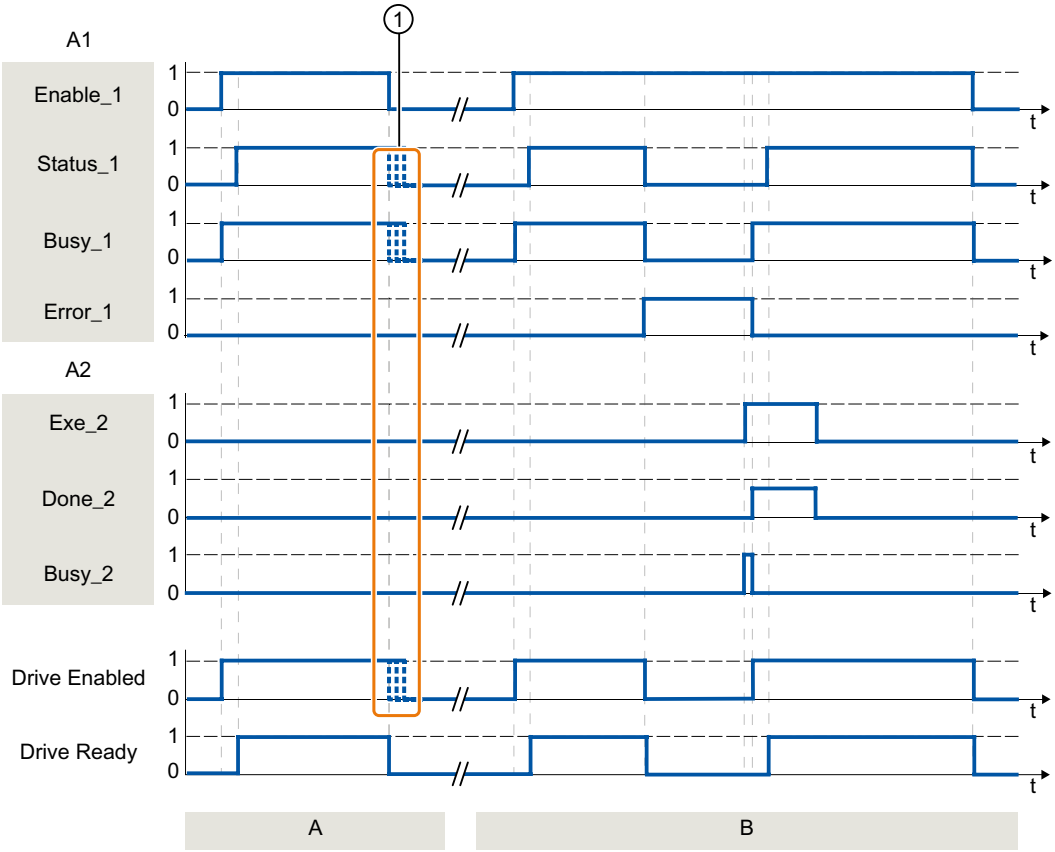

Section A	An axis is enabled and then disabled again. When the drive has signaled "Drive ready" back to the CPU, the successful enable can be read out via "Status_1".
Section B	Following an axis enable, an error has occurred that caused the axis to be disabled. The error is eliminated and acknowledged with "MC_Reset". The axis is then enabled again.
①	The exact end of the signals depends on the selected drive and the StopMode.

MC_Reset (S7-1200)

MC_Reset: Acknowledge fault, restart technology object as of V6 (S7-1200)

Description

Motion Control instruction "MC_Reset" can be used to acknowledge "Operating error with axis stop" and "Configuration error". The errors that require acknowledgment can be found in the "List of ErrorIDs and ErrorInfos" under "Remedy".

The axis configuration can be downloaded to the work memory after a download in RUN mode.

Requirements

- The positioning axis technology object has been configured correctly.
- The cause of a pending configuration error requiring acknowledgment has been eliminated (for example, acceleration in positioning axis technology object has been changed to a valid value).

Override response

The MC_Reset command cannot be aborted by any other Motion Control command.

The new MC_Reset command does not abort any other active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Restart	INPUT	BOOL	FALSE	TRUE	Download the axis configuration from the load memory to the work memory. The command can only be executed when the axis is disabled. Refer to the notes on Download to the CPU.
				FALSE	Acknowledges pending errors
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Acknowledging an error requiring acknowledgment with MC_Reset

To acknowledge an error, follow these steps:

1. Check the requirements indicated above.
2. Start the acknowledgment of the error with a rising edge at input parameter "Execute".
3. If output parameter "Done" indicates the value TRUE and tag of technology object <Axis name>.StatusBits.Error the value FALSE, the error has been acknowledged.

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)

MC_Home: Home axes, set reference point as of V6 (Page 3010)

MC_Halt: Stop axis as of V6 (Page 3015)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)

MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)

MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)

MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)

MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)

MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)

MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)

MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)

MC_Home (S7-1200)

MC_Home: Home axes, set reference point as of V6 (S7-1200)

Description

Motion Control instruction "MC_Home" is used to match the axis coordinates to the real, physical drive position. Homing is required for absolute positioning of the axis. The following types of homing can be executed:

- Active homing (Mode = 3)
  The homing procedure is executed automatically.
- Passive homing (Mode = 2)
  During passive homing, the "MC_Home" Motion Control instruction does not carry out any homing motion. The traversing motion required for this must be implemented by the user via other Motion Control instructions. When the homing switch is detected, the axis is homed.
- Direct homing absolute (Mode = 0)
  The current axis position is set to the value of parameter "Position".

- Direct homing relative (Mode = 1)
  The current axis position is offset by the value of parameter "Position".
- Absolute encoder adjustment relative (Mode = 6)
  The current axis position is offset by the value of parameter "Position".
- Absolute encoder adjustment absolute (Mode = 7)
  The current axis position is set to the value of parameter "Position".

Mode 6 and 7 can only be used with drives with an analog interface and PROFIdrive drive.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled. (Not valid for PROFIdrive drive/analog drive connection Mode = 0 or 1)
- No MC_CommandTable command may be active upon start with Mode = 0, 1, or 2.

Override response

The override response depends on the selected mode:

Mode = 0, 1, 6, 7

The MC_Home command cannot be aborted by any other Motion Control command.

The MC_Home command does not abort any active Motion Control commands. Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 2

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 2, 3

The new MC_Home command aborts the following active Motion Control command:

- MC_Home command Mode = 2

Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 3

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Home command aborts the following active Motion Control commands:

- MC_Home command Mode = 2, 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description
Axis	INPUT	TO_Axis	-	Axis technology object
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge
Position	INPUT	REAL	0.0	• Mode = 0, 2, and 3 Absolute position of axis after completion of the homing operation • Mode = 1 Correction value for the current axis position Limit values: -1.0E12 ≤ Position ≤ 1.0E12

Parameter	Declaration	Data type	Default value	Description	
Mode	INPUT	INT	0	Homing mode	
				0	Direct homing (absolute) New axis position is the position value of parameter "Position".
				1	Direct homing (relative) New axis position is the current axis position + position value of parameter "Position".
				2	Passive homing Homing according to the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
				3	Active homing Homing procedure in accordance with the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
				6	Absolute encoder adjustment (relative) The current axis position is offset by the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<AxisName>.StatusSensor.AbsEncoderOffset)
				7	Absolute encoder adjustment (absolute) The current axis position is set to the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<AxisName>.StatusSensor.AbsEncoderOffset)
Done	OUTPUT	BOOL	FALSE	TRUE	Command completed
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	
ReferenceMarkPosition	OUTPUT	REAL	0.0	Display of the position at which the technology object was homed (valid when "Done" = TRUE)	

Resetting the "Homed" status

The "Homed" status of a technology object (<Axis name>.StatusBits.HomingDone) is reset under the following conditions:

- **Drive connection via PTO (Pulse Train Output):**
  - Start an "MC_Home" command for active homing
    (After successful completion of the homing operation, the "Homed" status is set again.)
  - Disabling of axis by the "MC_Power" Motion Control instruction
  - Changeover between automatic mode and manual control
  - After POWER OFF → POWER ON of the CPU
  - After restart of CPU (RUN-STOP → STOP-RUN)
- **Technology objects with incremental actual values:**
  - Start an "MC_Home" command for active homing
    (After successful completion of the homing operation, the "Homed" status is set again.)
  - Error in the encoder system, or encoder failure
  - Restart of the technology object
  - After POWER OFF → POWER ON of the CPU
  - Memory reset
  - Modification of the encoder configuration
- **Technology objects with absolute actual values:**
  - Errors in the sensor system/encoder failure
  - Replacement of the CPU
  - Modification of the encoder configuration
  - Restoration of the CPU factory settings
  - Transfer of a different project to the controller

Homing an axis

To home the axis, follow these stops:

1. Check the requirements indicated above.
2. Provide the necessary input parameters with values and start the homing operation with a rising edge at input parameter "Execute".
3. If output parameter "Done" and technology object variable <axis name>.StatusBits.HomingDone indicate the value TRUE, homing is complete. The reference position can be taken from the <axis name>.ReferenceMarkPosition variable.

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)

MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)

MC_Halt: Stop axis as of V6 (Page 3015)
MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)
MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)
MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)
MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)
MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)
MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)
MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)
MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)

MC_Halt (S7-1200)

MC_Halt: Stop axis as of V6 (S7-1200)

Description

The "MC_Halt" Motion Control instruction stops all movements and brings the axis to a standstill with the configured deceleration. The standstill position is not defined.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_Halt command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Halt command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command

- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)

MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)

MC_Home: Home axes, set reference point as of V6 (Page 3010)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)

MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)

MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)

MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)

MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)

MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)

MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)

MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)

MC_Halt: Function chart as of V6 (S7-1200)

Function chart

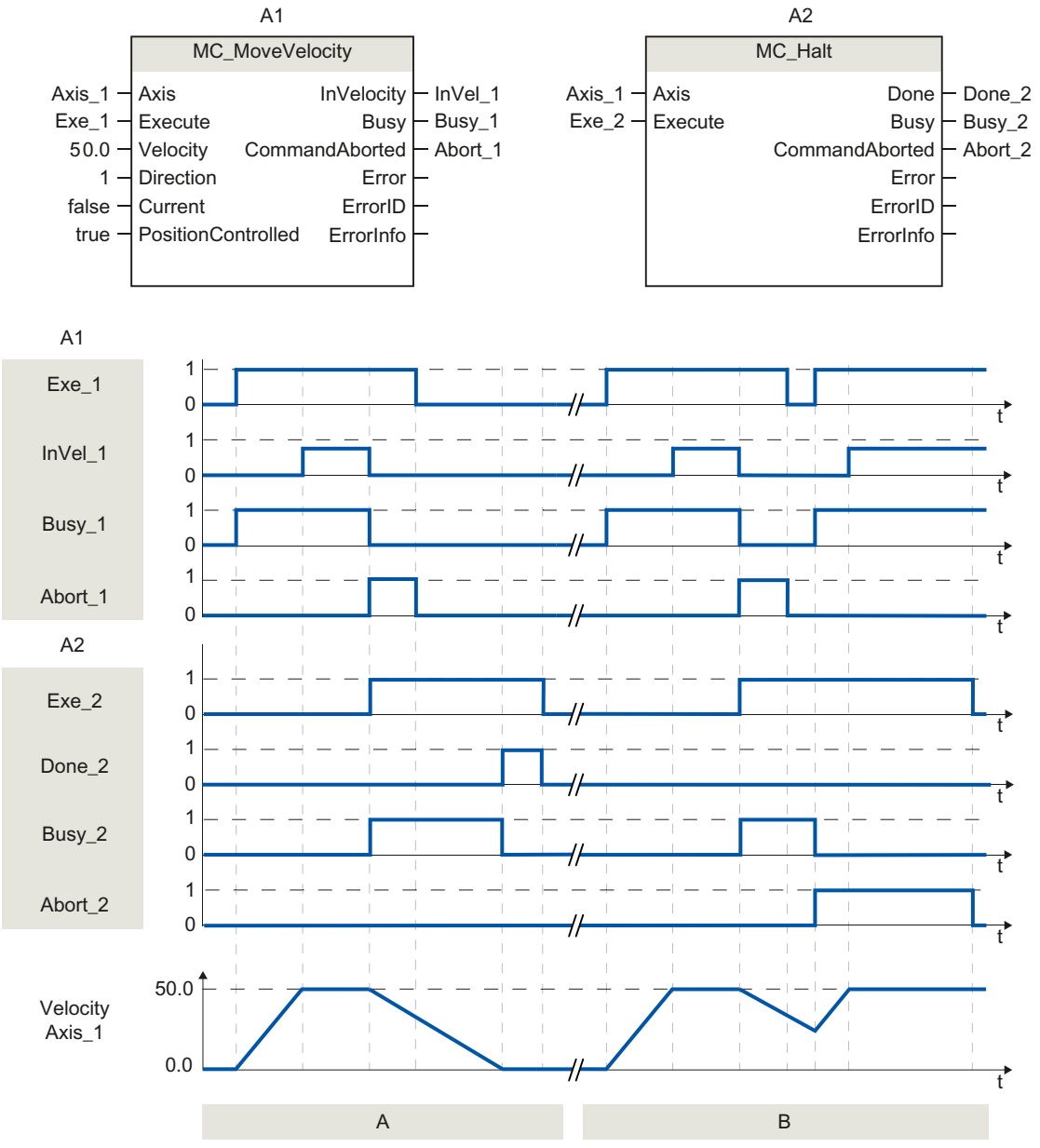

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

Section A	The axis is braked by an MC_Halt command until it comes to a standstill. The axis standstill is signaled via "Done_2".
Section B	While an MC_Halt command is braking the axis, this command is aborted by another motion command. The abort is signaled via "Abort_2".

MC_MoveAbsolute (S7-1200)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (S7-1200)

Description

The "MC_MoveAbsolute" Motion Control instruction starts an axis positioning motion to move it to an absolute position.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.
- The axis is homed.

Override response

The MC_MoveAbsolute command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveAbsolute command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command

- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_PositioningAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	Absolute target position Limit values: -1.0E12 ≤ Position ≤ 1.0E12	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the target position to be approached. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Direction	INPUT	INT	1	Motion direction of the axis Is only evaluated with "modulo" enabled. "Technology object > Configuration > Extended parameters > Modulo > Enable Modulo" Parameter is ignored with PTO axes.	
				0	The sign for the velocity ("Velocity" parameter) determines the motion direction.
				1	Positive direction (Target position is approached in a positive direction)
				2	Negative direction (Target position is approached in a negative direction)
				3	Shortest distance (Starting from the current position, the technology selects the shortest distance to the target position)
Done	OUTPUT	BOOL	FALSE	TRUE	Absolute target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)
MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)
MC_Home: Home axes, set reference point as of V6 (Page 3010)
MC_Halt: Stop axis as of V6 (Page 3015)
MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)
MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)
MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)
MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)
MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)
MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)
MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)

MC_MoveAbsolute: Function chart as of V6 (S7-1200)

Function chart

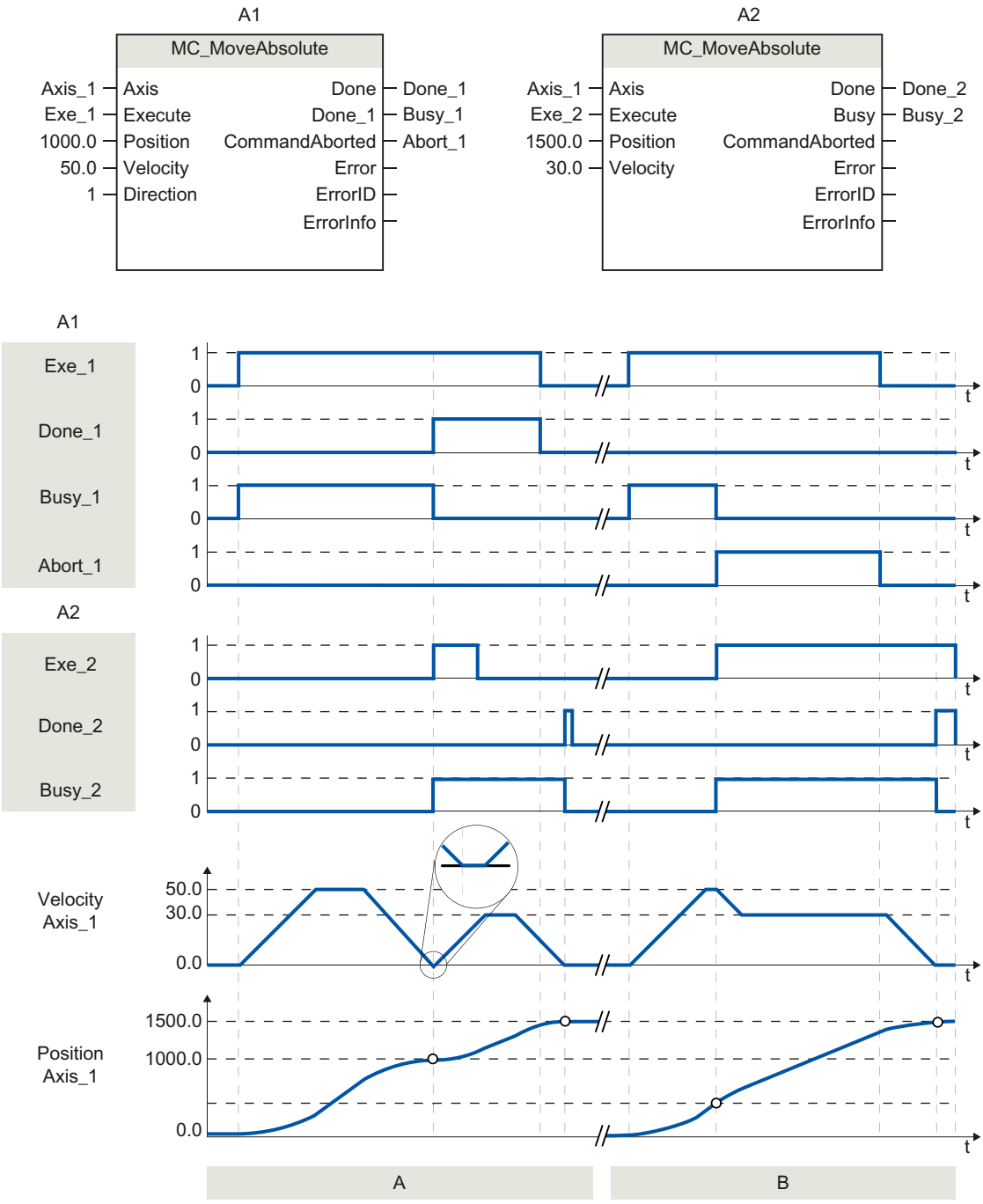

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

Section A	An axis is moved to absolute position 1000.0 with an MC_MoveAbsolute command. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveAbsolute command, with target position 1500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
Section B	An active MC_MoveAbsolute command is aborted by another MC_MoveAbsolute command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveRelative (S7-1200)

MC_MoveRelative: Relative positioning of axis as of V6 (S7-1200)

Description

The "MC_MoveRelative" Motion Control instruction starts a positioning motion relative to the start position.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveRelative command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveRelative command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command

- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning-Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Distance	INPUT	REAL	0.0	Travel distance for the positioning operation Limit values: -1.0E12 ≤ Distance ≤ 1.0E12	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the distance to be traveled. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)

MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)

MC_Home: Home axes, set reference point as of V6 (Page 3010)

MC_Halt: Stop axis as of V6 (Page 3015)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)

MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)

MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)

MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)

MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)

MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)

MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)

MC_MoveRelative: Function chart as of V6 (S7-1200)

Function chart

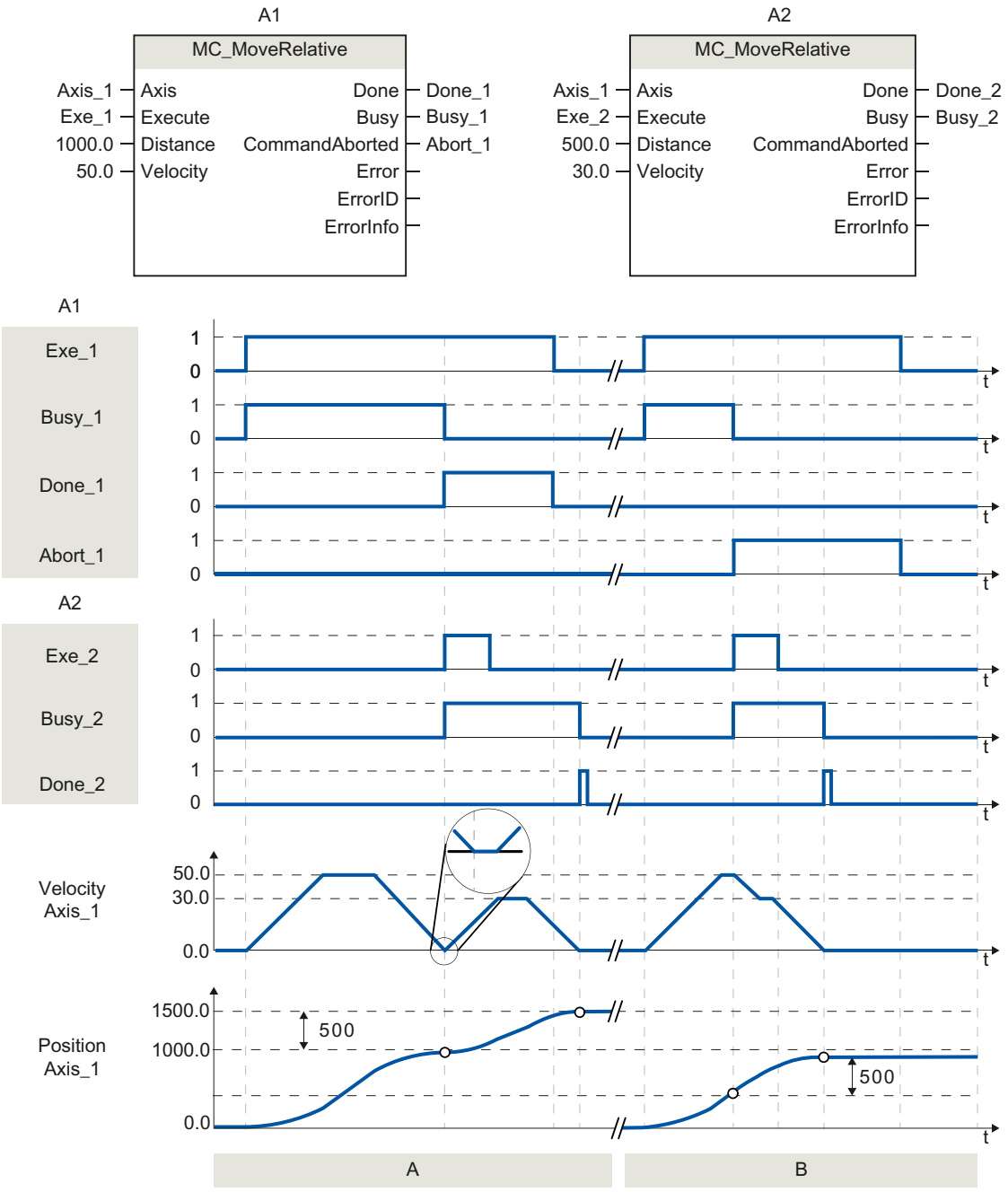

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

Section A	The axis is moved by an MC_MoveRelative command by the distance ("Distance") 1000.0. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveRelative command, with travel distance 500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
Section B	An active MC_MoveRelative command is aborted by another MC_MoveRelative command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity by the new distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveVelocity (S7-1200)

MC_MoveVelocity: Move axis at set velocity as of V6 (S7-1200)

Description

Motion control instruction "MC_MoveVelocity" moves the axis constantly at the specified velocity.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

MC_MoveVelocity can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveVelocity command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command

- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Velocity	INPUT	REAL	10.0	Velocity specification for axis motion Limit values: Start/stop velocity ≤ |Velocity ≤ maximum velocity (Velocity = 0.0 is permitted)	
Direction	INPUT	INT	0	Direction specification	
				0	Direction of rotation corresponds to the sign of the value in parameter "Velocity"
				1	Positive direction of rotation (The sign of the value in parameter "Velocity" is ignored)
				2	Negative direction of rotation (The sign of the value in parameter "Velocity" is ignored)
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	"Maintain current velocity" is deactivated. The values of parameters "Velocity" and "Direction" are used.
				TRUE	"Maintain current velocity" is activated. The values in parameters "Velocity" and "Direction" are not taken into account. When the axis resumes motion at the current velocity, the "InVelocity" parameter returns the value TRUE.
PositionControlled	INPUT	BOOL	TRUE	FALSE	Non position-controlled operation
				TRUE	Position-controlled operation
				The parameter applies as long as the "MC_MoveVelocity" command is being executed. After this, the setting of MC_Power applies again. This parameter is ignored when a PTO axis is used.	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	• "Current" = FALSE: The velocity specified in parameter "Velocity" was reached. • "Current" = TRUE: The axis travels at the current velocity at the start time.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.

Parameter	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Note

PLCopen Version 2.0

The Motion Control instruction "MC_MoveVelocity" is compatible to PLCopen Version 2.0 as of V4.

The "InVelocity" and "Busy" parameters show their status regardless of the "Execute" parameter until the command is overridden or stopped by an error. For more information, refer to the section " Tracking active commands.

Behavior with zero setpoint velocity (Velocity = 0.0)

An MC_MoveVelocity command with "Velocity" = 0.0 (such as an MC_Halt command) aborts active motion commands and stops the axis with the configured deceleration.

When the axis comes to a standstill, output parameter "InVelocity" indicates TRUE for at least one program cycle.

"Busy" indicates the value TRUE during the deceleration process and changes to FALSE together with "InVelocity". If parameter "Execute" = TRUE is set, "InVelocity" and "Busy" are latched.

When the "MC_MoveVelocity" command is started, status bit "SpeedCommand" is set in the technology object. Status bit "ConstantVelocity" is set upon axis standstill. Both bits are adapted to the new situation when a new motion command is started.

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)

MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)

MC_Home: Home axes, set reference point as of V6 (Page 3010)

MC_Halt: Stop axis as of V6 (Page 3015)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)

MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)

MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)

MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)

MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)

MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)

MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)

MC_MoveVelocity: Function chart as of V6 (S7-1200)

Function chart

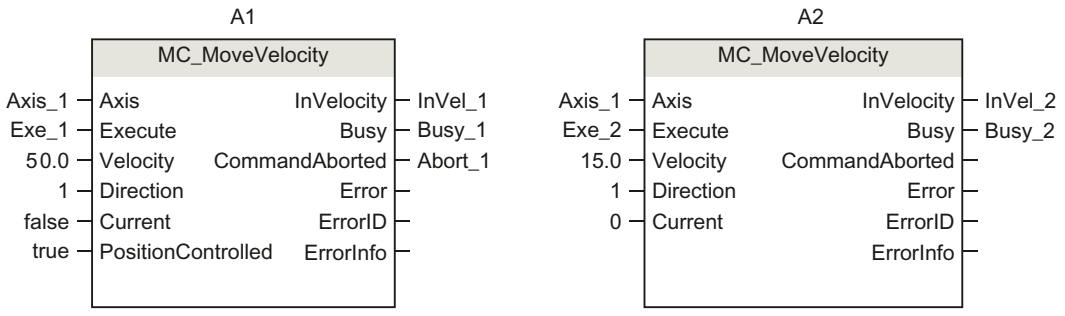

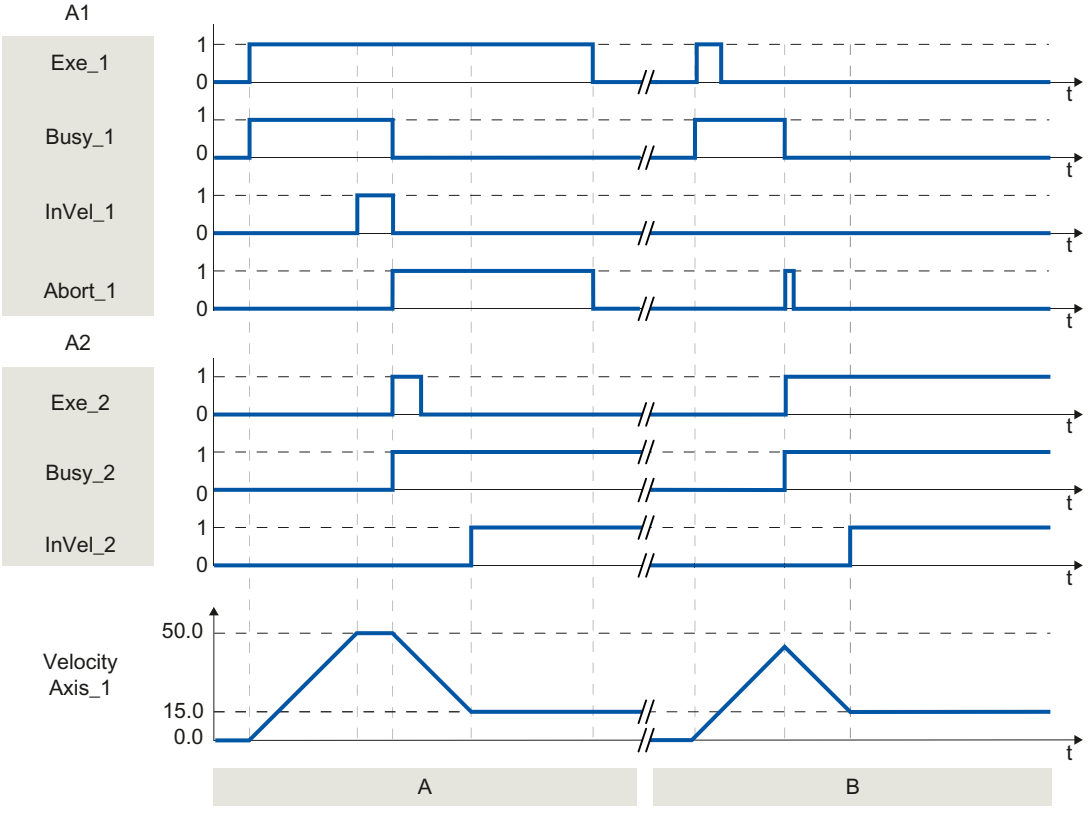

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

Section A	An active MC_MoveVelocity command signals via "InVel_1" that its target velocity has been reached. It is then aborted by another MC_MoveVelocity command. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.
Section B	An active MC_MoveVelocity command is aborted by another MC_MoveVelocity command prior to reaching its target velocity. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.

MC_MoveJog (S7-1200)

MC_MoveJog: Move axis in jog mode as of V6 (S7-1200)

Description

Motion control instruction "MC_MoveJog" moves the axis constantly at the specified velocity in jog mode. You use this Motion Control instruction, for example, for testing and commissioning purposes.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveJog command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveJog command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command

- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".	
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".	
If both parameters are simultaneously TRUE, the axis stops with the configured deceleration. An error is indicated in parameters "Error", "ErrorID", and "ErrorInfo".					
Velocity	INPUT	REAL	10.0	Preset velocity for jog mode Limit values: Start/stop velocity ≤ velocity ≤ maximum velocity	
PositionControlled	INPUT	BOOL	TRUE	FALSE	Non position-controlled operation
				TRUE	Position-controlled operation
				The parameter applies as long as the "MC_MoveJog" command is being executed. After this, the setting of MC_Power applies again. This parameter is ignored when a PTO axis is used.	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity specified in parameter "Velocity" was reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)

MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)

MC_Home: Home axes, set reference point as of V6 (Page 3010)

MC_Halt: Stop axis as of V6 (Page 3015)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)

MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)

MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)

MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)
MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)
MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)
MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)

MC_MoveJog: Function chart as of V6 (S7-1200)

Function chart

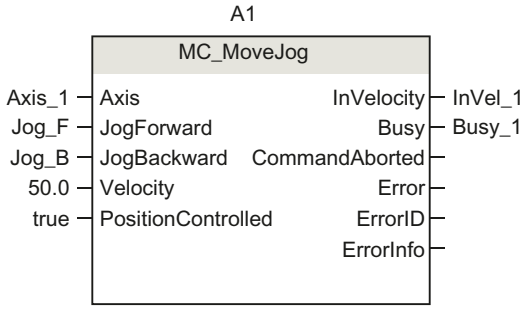

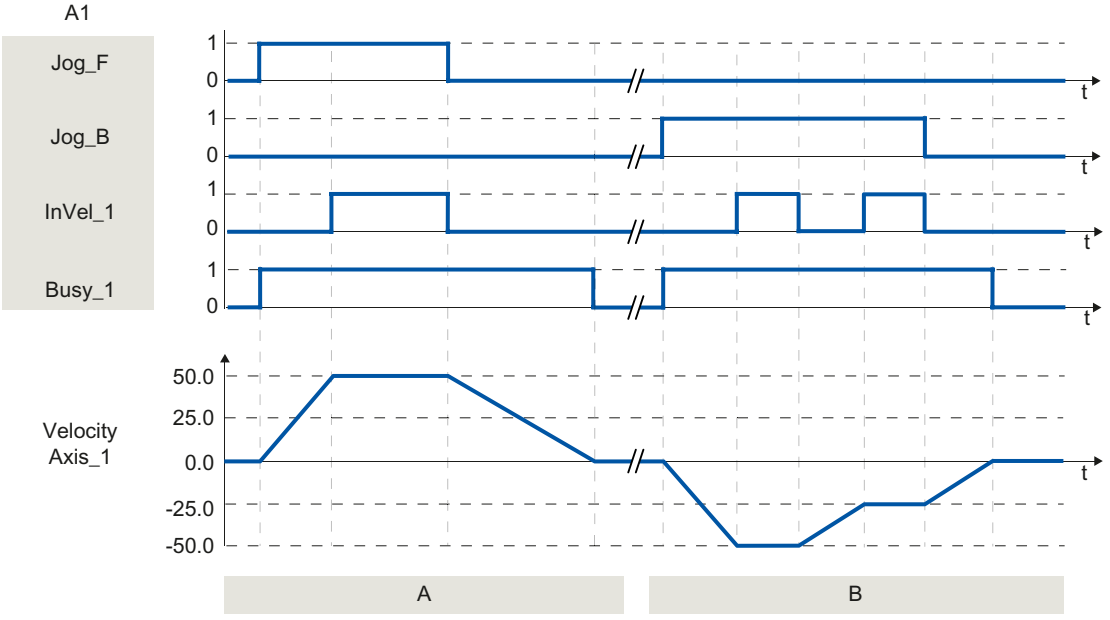

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

Section A	The axis is moved in the positive direction in jog mode via "Jog_F". When the target velocity 50.0 is reached, this is signaled via "InVel_1". After "Jog_F" is reset, the axis is braked to a standstill.
Section B	The axis is moved in the negative direction in jog mode via "Jog_B". When the target velocity -50.0 is reached, this is signaled via "InVel_1". When "Jog_B" is set, the value at parameter "Velocity" changes to 25.0. "InVel_1" is reset and the axis is braked. When the new target velocity -25.0 is reached, this is signaled via "InVel_1". After "Jog_B" is reset, the axis is braked to a standstill.

MC_CommandTable (S7-1200)

MC_CommandTable: Run axis commands as motion sequence as of V6 (S7-1200)

Description

The Motion Control instruction "MC_CommandTable" combines multiple individual axis control commands in one movement sequence. "MC_CommandTable" is available for axes with drive connection via PTO (Pulse Train Output).

Requirements

- The positioning axis technology object has been inserted and correctly configured.
- The drive is connected via PTO (Pulse Train Output).
- The command table technology object has been inserted and correctly configured.
- The axis is enabled.

Override response

The MC_CommandTable command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_CommandTable command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command

- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The active Motion Control command is canceled by the start of the first "Positioning Relative", "Positioning Absolute", "Velocity set point" or "Halt" command.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
CommandTable	INPUT	TO_CommandTable	-	Command table technology object	
Execute	INPUT	BOOL	FALSE	Command table start with positive edge	
StartStep	INPUT	INT	1	Defines the step at which the execution of the command table should begin Limit values: 1 ≤ StartStep ≤ EndStep	
EndStep	INPUT	INT	32	Defines the step up to which the execution of command table should take place Limit values: StartStep ≤ EndStep ≤ 32	
Done	OUTPUT	BOOL	FALSE	TRUE	Command table has been successfully executed
Busy	OUTPU	BOOL	FALSE	TRUE	The command table is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The command table was canceled by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command table. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	
CurrentStep	OUTPUT	INT	0	Step in command table currently being executed	
StepCode	OUTPUT	WORD	16#0000	User-defined numerical value / bit pattern of the step currently being executed	

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)

MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)

MC_Home: Home axes, set reference point as of V6 (Page 3010)

MC_Halt: Stop axis as of V6 (Page 3015)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)

MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)

MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)
MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)
MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)
MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)
MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)

MC_ChangeDynamic (S7-1200)

MC_ChangeDynamic: Change dynamic settings of axis as of V6 (S7-1200)

Description

Motion Control instruction "MC_ChangeDynamic" allows you to change the following settings of the axis:

- Change the ramp-up time (acceleration) value
- Change the ramp-down time (deceleration) value
- Change the emergency stop ramp-down time (emergency stop deceleration) value
- Change the smoothing time (jerk) value

For the effectiveness of the change, refer to the description of the tag.

Requirements

The positioning axis technology object has been configured correctly.

Override response

A MC_ChangeDynamic command cannot be aborted by any other Motion Control command.

A new MC_ChangeDynamic command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
ChangeRampUp	INPUT	BOOL	FALSE	TRUE	Change ramp-up time in line with input parameter "RampUpTime"
RampUpTime	INPUT	REAL	5.00	Time (in seconds) to accelerate axis from standstill to configured maximum velocity without jerk limit. The change will influence the tag <Axis name>. Config.DynamicDefaults.Acceleration. For the effectiveness of the change, refer to the description of this tag.	

Parameter	Declaration	Data type	Default value	Description	
ChangeRampDown	INPUT	BOOL	FALSE	TRUE	Change ramp-down time to correspond to input parameter "RampDownTime"
RampDownTime	INPUT	REAL	5.00	Time (in seconds) to decelerate axis from the configured maximum velocity to standstill without jerk limiter. The change will influence the tag <Axis name>. Config.DynamicDefaults.Deceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeEmergency	INPUT	BOOL	FALSE	TRUE	Change emergency stop ramp-down time in line with input parameter "EmergencyRampTime"
EmergencyRampTime	INPUT	REAL	2.00	Time (in seconds) to decelerate the axis from configured maximum velocity to standstill without jerk limiter in emergency stop mode. The change will influence the tag <Axis name>. Config.DynamicDefaults.EmergencyDeceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeJerkTime	INPUT	BOOL	FALSE	TRUE	Change smoothing time according to the input parameter "JerkTime"
JerkTime	INPUT	REAL	0.25	Smoothing time (in seconds) used for the axis acceleration and deceleration ramps The change will influence the tag <Axis name>. Config.DynamicDefaults.Jerk . For the effectiveness of the change, refer to the description of this tag.	
Done	OUTPUT	BOOL	FALSE	TRUE	The changed values have been written to the technology data block. The description of the tags will show when the change becomes effective.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Note

At the input parameters "RampUpTime", "RampDownTime", "EmergencyRampTime" and "JerkTime", values can be entered which exceed the admissible limits of the resulting parameters: "Acceleration", "Deceleration", "Emergency stop deceleration" and "Jerk".

Ensure that your inputs are within the valid range, taking into consideration the equations and limits in section "Dynamic".

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)

MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)

MC_Home: Home axes, set reference point as of V6 (Page 3010)

MC_Halt: Stop axis as of V6 (Page 3015)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)

MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)
MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)
MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)
MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)
MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)
MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)

MC_ReadParam (S7-1200)

MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (S7-1200)

Description

The Motion Control instruction "MC_ReadParam" enables continuous reading of motion data and status messages of an axis. The current value of the corresponding tags is determined at the start of the command.

The following motion data and status messages can be read:

- As of technology version V4:
  - Setpoint position of the axis
  - Setpoint and actual velocity of the axis
  - Current distance of axis from target position
  - Target position of the axis
- Additional as of technology version V5:
  - Actual position of the axis
  - Actual velocity of the axis
  - Current following error
  - Drive status
  - Encoder status
  - Status bits
  - Error bits

Requirements

The positioning axis technology object has been configured correctly.

Override response

A MC_ReadParam command cannot be aborted by any other Motion Control command.

A new MC_ReadParam command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Enable	INPUT	BOOL	FALSE	TRUE	Read the tag specified with the "Parameter" and store the value in the destination address specified with "Value".
				FALSE	Do not update assigned motion data
Parameter	INPUT	VARIANT (REAL)	-	VARIANT pointer to the value to be read. The following tags are permitted: • <Axis name>.Position • <Axis name>.Velocity • <Axis name>.ActualPosition • <Axis name>.ActualVelocity • <Axis name>.StatusPositioning.<Tag name> • <Axis name>.StatusDrive.<Tag name> • <Axis name>.StatusSensor.<Tag name> • <Axis name>.StatusBits.<Tag name> • <Axis name>.ErrorBits.<Tag name> The description of the tags named and the tag structures can be found in the Appendix AUTOHOTSPOT.	
Value	INOUT	VARIANT (REAL)	-	VARIANT pointer to the target tag or destination address to which the read value is to be written.	
Valid	OUTPUT	BOOL	FALSE	TRUE	The read value is valid.
				FALSE	The read value is invalid.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)

MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)

MC_Home: Home axes, set reference point as of V6 (Page 3010)

MC_Halt: Stop axis as of V6 (Page 3015)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)

MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)

MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)

MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)

MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)

MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)

MC_WriteParam: Write tag of positioning axis as of V6 (Page 3038)

MC_WriteParam (S7-1200)

MC_WriteParam: Write tag of positioning axis as of V6 (S7-1200)

Description

Motion Control instruction "MC_WriteParam" enables the writing of tags of the positioning axis technology object in the user program. In contrast to the value assignment of the tags in the user program, "MC_WriteParam" can also change values of read-only tags.

You can learn about the tags, the conditions under which they can be written and the time at which they take effect in the description of the technology object tags.

Requirements

- The positioning axis technology object has been configured correctly.
- To write tags that are read-only in the user program, the axis must be disabled.
- Tags whose change requires a restart cannot be written with "MC_WriteParam".

Override response

A MC_WriteParam command cannot be aborted by any other Motion Control command.

A new MC_WriteParam command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Parameter	INPUT	VARIANT (BOOL, INT, DINT, UDINT, REAL)	-	VARIANT pointer to the technology object tags positioning axis (destination address) to be written	
Value	INPUT	VARIANT (BOOL, INT, DINT, UDINT, REAL)	-	VARIANT pointer to the value to be written (source address)	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Value was written
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".

Parameter	Declaration	Data type	Default value	Description
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"

See also

MC_Power: Enable, disable axis as of V6 (Page 3004)

MC_Reset: Acknowledge fault, restart technology object as of V6 (Page 3009)

MC_Home: Home axes, set reference point as of V6 (Page 3010)

MC_Halt: Stop axis as of V6 (Page 3015)

MC_MoveAbsolute: Absolute positioning of axis as of V6 (Page 3018)

MC_MoveRelative: Relative positioning of axis as of V6 (Page 3022)

MC_MoveVelocity: Move axis at set velocity as of V6 (Page 3025)

MC_MoveJog: Move axis in jog mode as of V6 (Page 3029)

MC_CommandTable: Run axis commands as motion sequence as of V6 (Page 3032)

MC_ChangeDynamic: Change dynamic settings of axis as of V6 (Page 3034)

MC_ReadParam: Continuously read motion data of a positioning axis as of V6 (Page 3036)

S7-1200 Motion Control V4...5 (S7-1200)

MC_Power (S7-1200)

MC_Power: Enable, disable axis V4...5 (S7-1200)

Description

The Motion Control instruction "MC_Power" enables or disables an axis.

Requirements

- The positioning axis technology object has been configured correctly.
- There is no pending enable-inhibiting error.

Override response

Execution of "MC_Power" cannot be aborted by a Motion Control command.

Disabling the axis (input parameter "Enable" = FALSE) aborts all Motion Control commands for the associated technology object in accordance with the selected "StopMode".

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Axis technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The axis is enabled.
				FALSE	All current jobs are interrupted in accordance with the "StopMode" configured. The axis is stopped and disabled.
StopMode	INPUT	INT	0	0	Emergency stop If a request to disable the axis is pending, the axis brakes at the configured emergency deceleration. The axis is disabled after reaching standstill.
				1	Immediate stop If a request to disable the axis is pending, this setpoint zero is output and the axis is disabled. The axis is braked depending on the configuration in the drive, and is brought to a standstill. With drive connection via PTO (Pulse Train Output): When you disable the axis, the pulse output is stopped with a frequency-dependent deceleration: • Output frequency ≥ 100 Hz Deceleration: max. 30 ms • Output frequency < 100 Hz Deceleration: 30 ms up to max. 1.5 s at 2 Hz
				2	Emergency stop with jerk control If a request to disable the axis is pending, the axis brakes at the configured emergency deceleration. If the jerk control is activated, the configured jerk is taken into account. The axis is disabled after reaching standstill.
Status	OUTPUT	BOOL	FALSE	Status of axis enable	
				FALSE	The axis is disabled. The axis does not execute Motion Control commands and does not accept any new commands (exception: MC_Reset command). For drive connection via PTO (Pulse Train Output): The axis is not homed. Upon disabling, the status does not change to FALSE until the axis reaches a standstill.
				TRUE	The axis is enabled. The axis is ready to execute Motion Control commands. Upon axis enabling, the status does not change to TRUE until the signal "Drive ready" is pending. If the "Drive ready" drive interface was not configured in the axis configuration, the status changes to TRUE immediately.
Busy	OUTPUT	BOOL	FALSE	TRUE	"MC_Power" is active.

Parameter	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction "MC_Power" or in the associated technology object. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Note

If the axis is switched off due to an error, it will be enabled again automatically after the error has been eliminated and acknowledged. This requires that the input parameter "Enable" has retained the value TRUE during this process.

Enabling an axis with configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE.
   The enable output for "Drive enabled" changes to TRUE to enable the power to the drive. The CPU waits for the "Drive ready" signal of the drive.
   When the "Drive ready" signal is available at the configured ready input of the CPU, the axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Enabling an axis without configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE. The axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Disabling an axis

To disable an axis, you can follow the steps described below:

1. Bring the axis to a standstill.
   You can identify when the axis is at a standstill in the tag of the technology object <Axis name>.StatusBits.StandStill.
2. Set input parameter "Enable" to FALSE after standstill is reached.
3. If output parameters "Busy" and "Status" and tag of technology object <Axis name>.StatusBits.Enable indicate the value FALSE, disabling of the axis is complete.

See also

MC_Power: Function chart V4...5 (Page 3043)
MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)
MC_Home: Home axis, set home position V4...5 (Page 3045)
MC_Halt: Stop axis V4...5 (Page 3050)
MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)
MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)
MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)
MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)
MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)
MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)
MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)
MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)
S7-1200 Motion Control V1...3 (Page 3073)

MC_Power: Function chart V4...5 (S7-1200)

Function chart

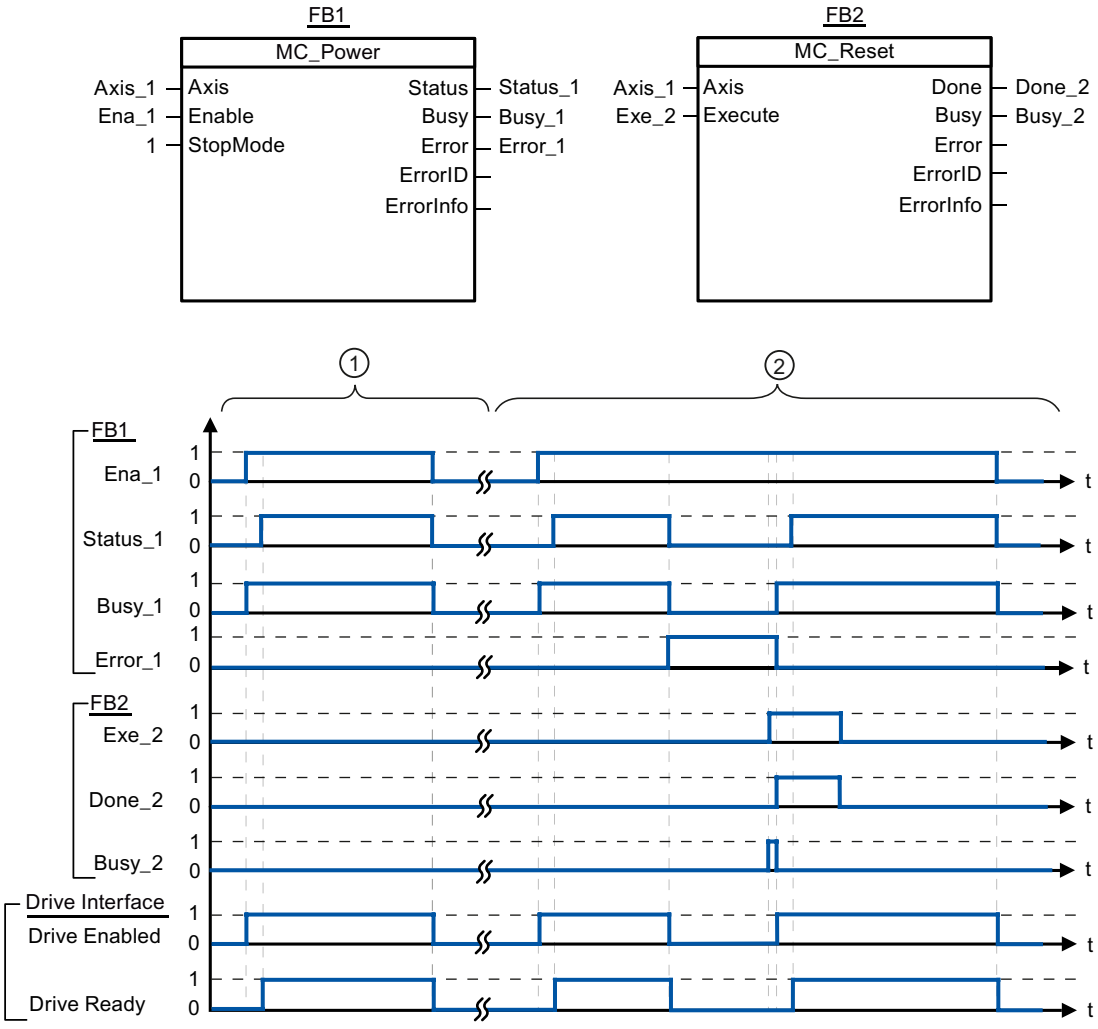

①	An axis is enabled and then disabled again. When the drive has signaled "Drive ready" back to the CPU, the successful enable can be read out via "Status_1".
②	Following an axis enable, an error has occurred that caused the axis to be disabled. The error is eliminated and acknowledged with "MC_Reset". The axis is then enabled again.

See also

MC_Power: Enable, disable axis V4...5 (Page 3039)

MC_Reset (S7-1200)

MC_Reset: Acknowledge fault, restart technology object V4...5 (S7-1200)

Description

Motion Control instruction "MC_Reset" can be used to acknowledge "Operating error with axis stop" and "Configuration error". The errors that require acknowledgment can be found in the "List of ErrorIDs and ErrorInfos" under "Remedy".

The axis configuration can be downloaded to the work memory after a download in RUN mode.

Requirements

- The positioning axis technology object has been configured correctly.
- The cause of a pending configuration error requiring acknowledgment has been eliminated (for example, acceleration in positioning axis technology object has been changed to a valid value).

Override response

The MC_Reset command cannot be aborted by any other Motion Control command.

The new MC_Reset command does not abort any other active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Restart	INPUT	BOOL	FALSE	TRUE	Download the axis configuration from the load memory to the work memory. The command can only be executed when the axis is disabled. Refer to the notes on Download to the CPU.
				FALSE	Acknowledges pending errors
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Acknowledging an error requiring acknowledgment with MC_Reset

To acknowledge an error, follow these steps:

1. Check the requirements indicated above.
2. Start the acknowledgment of the error with a rising edge at input parameter "Execute".
3. If output parameter "Done" indicates the value TRUE and tag of technology object <Axis name>.StatusBits.Error the value FALSE, the error has been acknowledged.

See also

MC_Power: Enable, disable axis V4...5 (Page 3039)

MC_Home: Home axis, set home position V4...5 (Page 3045)

MC_Halt: Stop axis V4...5 (Page 3050)

MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)

MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)

MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)

MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)

MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)

MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)

MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)

MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)

S7-1200 Motion Control V1...3 (Page 3073)

MC_Home (S7-1200)

MC_Home: Home axis, set home position V4...5 (S7-1200)

Description

Motion Control instruction "MC_Home" is used to match the axis coordinates to the real, physical drive position. Homing is required for absolute positioning of the axis. The following types of homing can be executed:

- Active homing (Mode = 3)
  The homing procedure is executed automatically.
- Passive homing (Mode = 2)
  During passive homing, the "MC_Home" Motion Control instruction does not carry out any homing motion. The traversing motion required for this must be implemented by the user via other Motion Control instructions. When the homing switch is detected, the axis is homed.
- Direct homing absolute (Mode = 0)
  The current axis position is set to the value of parameter "Position".

- Direct homing relative (Mode = 1)
  The current axis position is offset by the value of parameter "Position".
- Absolute encoder adjustment relative (Mode = 6)
  The current axis position is offset by the value of parameter "Position".
- Absolute encoder adjustment absolute (Mode = 7)
  The current axis position is set to the value of parameter "Position".

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.
- No MC_CommandTable command may be active upon start with Mode = 0, 1, or 2.

Override response

The override response depends on the selected mode:

Mode = 0, 1, 6, 7

The MC_Home command cannot be aborted by any other Motion Control command.

The MC_Home command does not abort any active Motion Control commands. Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 2

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 2, 3

The new MC_Home command aborts the following active Motion Control command:

- MC_Home command Mode = 2

Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 3

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Home command aborts the following active Motion Control commands:

- MC_Home command Mode = 2, 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description
Axis	INPUT	TO_Axis	-	Axis technology object
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge
Position	INPUT	REAL	0.0	• Mode = 0, 2, and 3 Absolute position of axis after completion of the homing operation • Mode = 1 Correction value for the current axis position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$

Parameter	Declaration	Data type	Default value	Description	
Mode	INPUT	INT	0	Homing mode	
				0	Direct homing (absolute) New axis position is the position value of parameter "Position".
				1	Direct homing (relative) New axis position is the current axis position + position value of parameter "Position".
				2	Passive homing Homing according to the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
				3	Active homing Homing procedure in accordance with the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
				6	Absolute encoder adjustment (relative) The current axis position is offset by the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<AxisName>.StatusSensor.AbsEncoderOffset)
				7	Absolute encoder adjustment (absolute) The current axis position is set to the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<AxisName>.StatusSensor.AbsEncoderOffset)
Done	OUTPUT	BOOL	FALSE	TRUE	Command completed
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Resetting the "Homed" status

The "Homed" status of a technology object (<Axis name>.StatusBits.HomingDone) is reset under the following conditions:

- **Drive connection via PTO (Pulse Train Output):**
  - Start an "MC_Home" command for active homing
    (After successful completion of the homing operation, the "Homed" status is set again.)
  - Disabling of axis by the "MC_Power" Motion Control instruction
  - Changeover between automatic mode and manual control
  - After POWER OFF -> POWER ON of the CPU
  - After CPU restart (RUN-STOP -> STOP-RUN)
- **Technology objects with incremental actual values:**
  - Start an "MC_Home" command for active homing
    (After successful completion of the homing operation, the "Homed" status is set again.)
  - Error in the encoder system, or encoder failure
  - Restart of the technology object
  - After POWER OFF → POWER ON of the CPU
  - Memory reset
  - Modification of the encoder configuration
- **Technology objects with absolute actual values:**
  - Errors in the sensor system/encoder failure
  - Replacement of the CPU
  - Modification of the encoder configuration
  - Restoration of the CPU factory settings
  - Transfer of a different project to the controller

Homing an axis

To home the axis, follow these stops:

1. Check the requirements indicated above.
2. Provide the necessary input parameters with values and start the homing operation with a rising edge at input parameter "Execute".
3. If output parameter "Done" and technology object tag <Axis name>.StatusBits.HomingDone indicate the value TRUE, homing is complete.

See also

MC_Power: Enable, disable axis V4...5 (Page 3039)

MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)

MC_Halt: Stop axis V4...5 (Page 3050)

MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)
MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)
MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)
MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)
MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)
MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)
MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)
MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)
S7-1200 Motion Control V1...3 (Page 3073)

MC_Halt (S7-1200)

MC_Halt: Stop axis V4...5 (S7-1200)

Description

The "MC_Halt" Motion Control instruction stops all movements and brings the axis to a standstill with the configured deceleration. The standstill position is not defined.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_Halt command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Halt command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command

- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_Halt: Function chart V4...5 (Page 3052)
MC_Power: Enable, disable axis V4...5 (Page 3039)
MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)
MC_Home: Home axis, set home position V4...5 (Page 3045)
MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)
MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)
MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)
MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)
MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)
MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)
MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)
MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)
S7-1200 Motion Control V1...3 (Page 3073)

MC_Halt: Function chart V4...5 (S7-1200)

Function chart

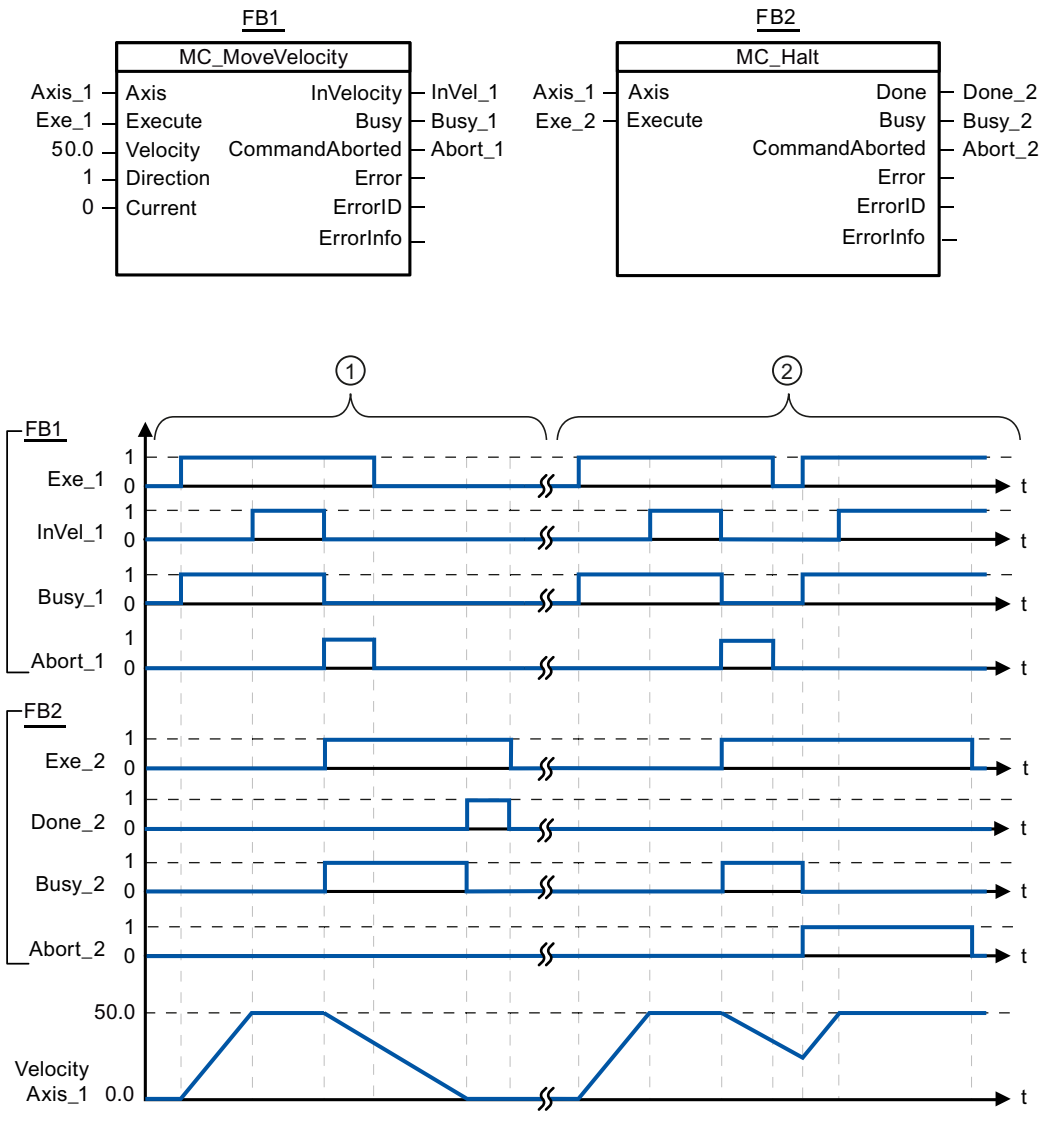

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is braked by an MC_Halt command until it comes to a standstill. The axis standstill is signaled via "Done_2".
②	While an MC_Halt command is braking the axis, this command is aborted by another motion command. The abort is signaled via "Abort_2".

See also

MC_Halt: Stop axis V4...5 (Page 3050)

MC_MoveAbsolute (S7-1200)

MC_MoveAbsolute: Absolute positioning of axis V4...5 (S7-1200)

Description

The "MC_MoveAbsolute" Motion Control instruction starts an axis positioning motion to move it to an absolute position.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.
- The axis is homed.

Override response

The MC_MoveAbsolute command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveAbsolute command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning-Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	Absolute target position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the target position to be approached. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Absolute target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_MoveAbsolute: Function chart V4...5 (Page 3055)

MC_Power: Enable, disable axis V4...5 (Page 3039)

MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)

MC_Home: Home axis, set home position V4...5 (Page 3045)

MC_Halt: Stop axis V4...5 (Page 3050)

MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)

MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)

MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)

MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)

MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)

MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)

MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)

S7-1200 Motion Control V1...3 (Page 3073)

MC_MoveAbsolute: Function chart V4...5 (S7-1200)

Function chart

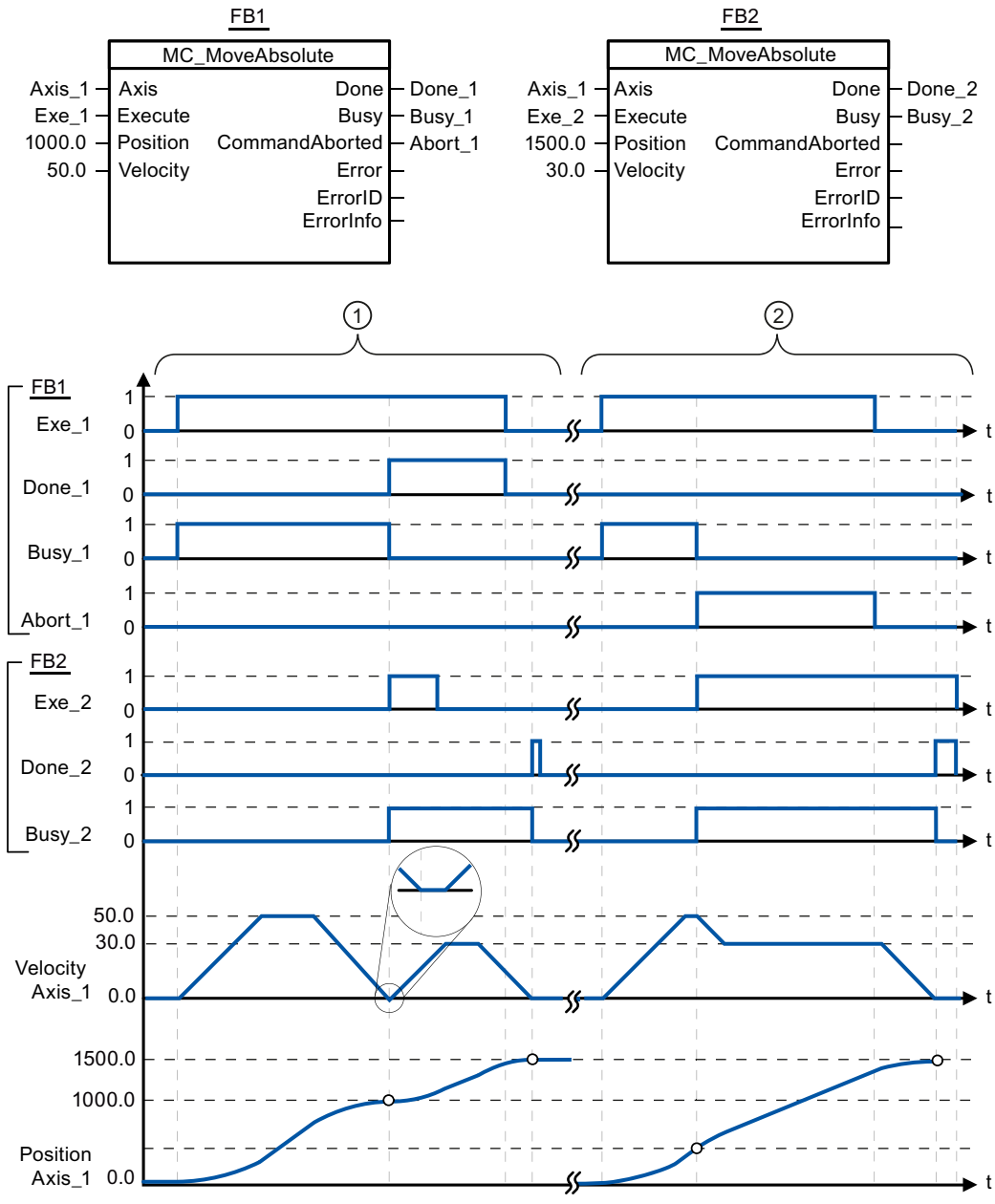

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An axis is moved to absolute position 1000.0 with an MC_MoveAbsolute command. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveAbsolute command, with target position 1500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveAbsolute command is aborted by another MC_MoveAbsolute command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)

MC_MoveRelative (S7-1200)

MC_MoveRelative: Relative positioning of axis V4...5 (S7-1200)

Description

The "MC_MoveRelative" Motion Control instruction starts a positioning motion relative to the start position.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveRelative command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveRelative command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command

- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning-Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Distance	INPUT	REAL	0.0	Travel distance for the positioning operation Limit values: $-1.0e^{12} \leq$ Distance $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the distance to be traveled. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_MoveRelative: Function chart V4...5 (Page 3058)

MC_Power: Enable, disable axis V4...5 (Page 3039)

MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)

MC_Home: Home axis, set home position V4...5 (Page 3045)

MC_Halt: Stop axis V4...5 (Page 3050)

MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)

MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)

MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)

MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)

MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)

MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)
MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)
S7-1200 Motion Control V1...3 (Page 3073)

MC_MoveRelative: Function chart V4...5 (S7-1200)

Function chart

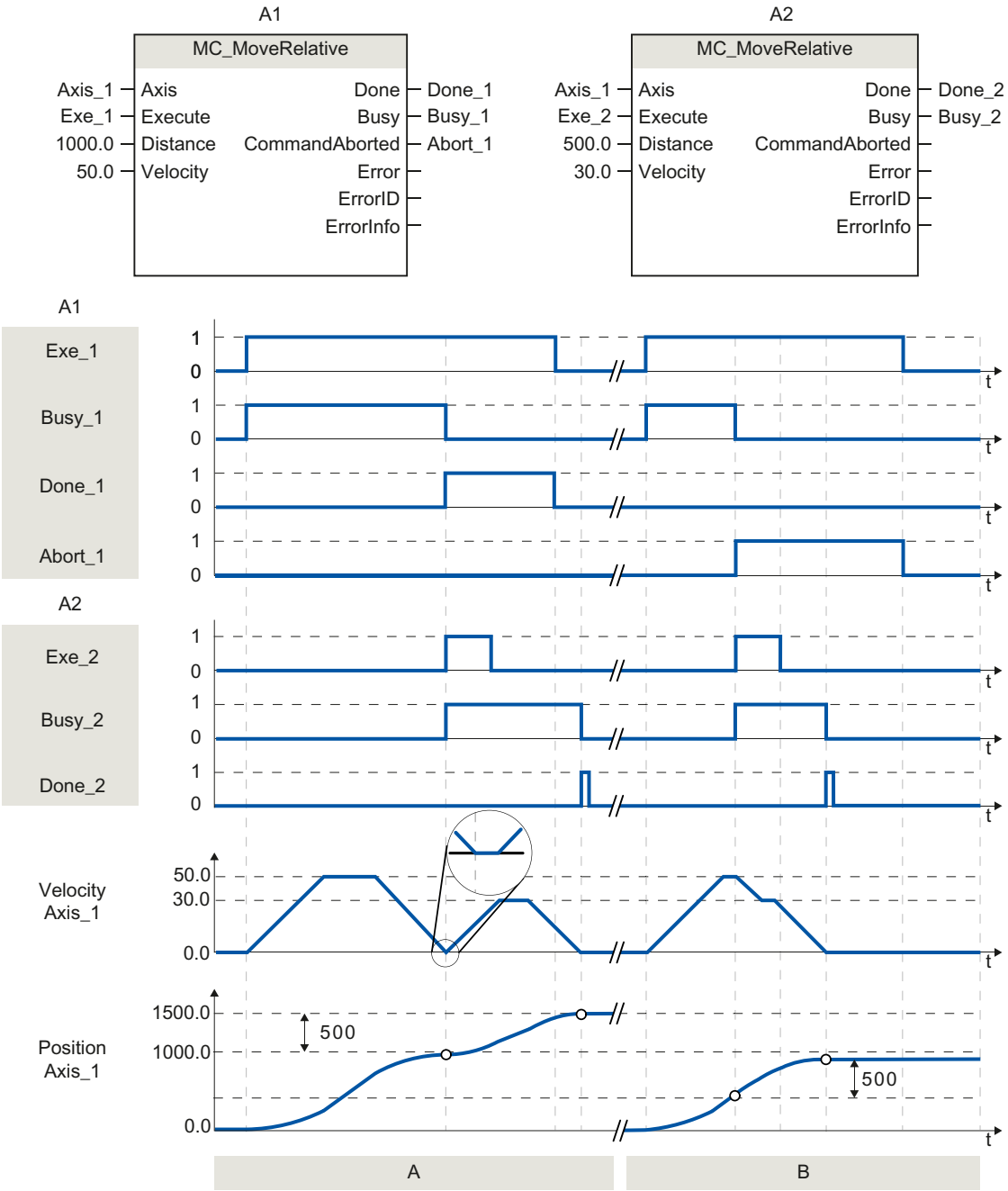

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	The axis is moved by an MC_MoveRelative command by the distance ("Distance") 1000.0. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveRelative command, with travel distance 500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveRelative command is aborted by another MC_MoveRelative command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity by the new distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)

MC_MoveVelocity (S7-1200)

MC_MoveVelocity: Move axis at set velocity V4...5 (S7-1200)

Description

Motion control instruction "MC_MoveVelocity" moves the axis constantly at the specified velocity.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

MC_MoveVelocity can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveVelocity command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Velocity	INPUT	REAL	10.0	Velocity specification for axis motion Limit values: Start/stop velocity ≤ |Velocity ≤ maximum velocity (Velocity = 0.0 is permitted)	
Direction	INPUT	INT	0	Direction specification	
				0	Direction of rotation corresponds to the sign of the value in parameter "Velocity"
				1	Positive direction of rotation (The sign of the value in parameter "Velocity" is ignored)
				2	Negative direction of rotation (The sign of the value in parameter "Velocity" is ignored)
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	"Maintain current velocity" is deactivated. The values of parameters "Velocity" and "Direction" are used.
				TRUE	"Maintain current velocity" is activated. The values in parameters "Velocity" and "Direction" are not taken into account. When the axis resumes motion at the current velocity, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUTPUT	BOOL	FALSE	TRUE	• "Current" = FALSE: The velocity specified in parameter "Velocity" was reached. • "Current" = TRUE: The axis travels at the current velocity at the start time.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed

Parameter	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Note

PLCopen Version 2.0

The Motion Control instruction "MC_MoveVelocity" is compatible to PLCopen Version 2.0 as of V4.

The "InVelocity" and "Busy" parameters show their status regardless of the "Execute" parameter until the command is overridden or stopped by an error. For more information, refer to the section " Tracking active commands.

Behavior with zero setpoint velocity (Velocity = 0.0)

An MC_MoveVelocity command with "Velocity" = 0.0 (such as an MC_Halt command) aborts active motion commands and stops the axis with the configured deceleration.

When the axis comes to a standstill, output parameter "InVelocity" indicates TRUE for at least one program cycle.

"Busy" indicates the value TRUE during the deceleration process and changes to FALSE together with "InVelocity". If parameter "Execute" = TRUE is set, "InVelocity" and "Busy" are latched.

When the "MC_MoveVelocity" command is started, status bit "SpeedCommand" is set in the technology object. Status bit "ConstantVelocity" is set upon axis standstill. Both bits are adapted to the new situation when a new motion command is started.

See also

MC_MoveVelocity: Function chart V4...5 (Page 3062)

MC_Power: Enable, disable axis V4...5 (Page 3039)

MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)

MC_Home: Home axis, set home position V4...5 (Page 3045)

MC_Halt: Stop axis V4...5 (Page 3050)

MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)

MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)

MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)

MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)

MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)
MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)
MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)
S7-1200 Motion Control V1...3 (Page 3073)

MC_MoveVelocity: Function chart V4...5 (S7-1200)

Function chart

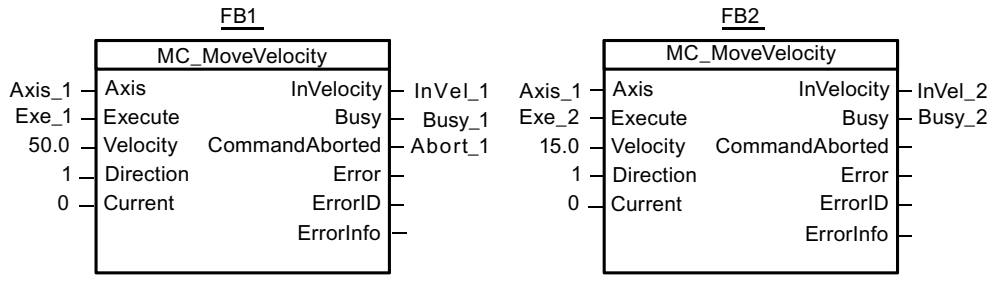

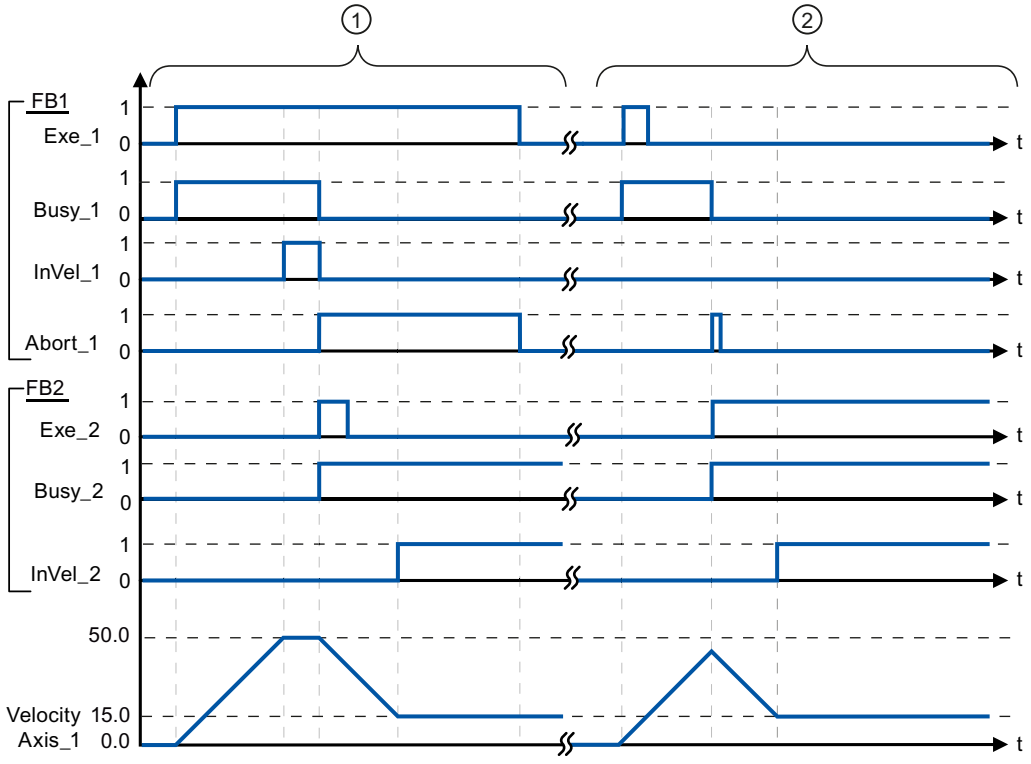

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An active MC_MoveVelocity command signals via "InVel_1" that its target velocity has been reached. It is then aborted by another MC_MoveVelocity command. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.
②	An active MC_MoveVelocity command is aborted by another MC_MoveVelocity command prior to reaching its target velocity. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.

See also

MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)

MC_MoveJog (S7-1200)

MC_MoveJog: Move axis in jog mode V4...5 (S7-1200)

Description

Motion control instruction "MC_MoveJog" moves the axis constantly at the specified velocity in jog mode. You use this Motion Control instruction, for example, for testing and commissioning purposes.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveJog command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveJog command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command

- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".	
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".	
If both parameters are simultaneously TRUE, the axis stops with the configured deceleration. An error is indicated in parameters "Error", "ErrorID", and "ErrorInfo".					
Velocity	INPUT	REAL	10.0	Preset velocity for jog mode Limit values: Start/stop velocity ≤ velocity ≤ maximum velocity	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity specified in parameter "Velocity" was reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_MoveJog: Function chart V4...5 (Page 3065)

MC_Power: Enable, disable axis V4...5 (Page 3039)

MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)

MC_Home: Home axis, set home position V4...5 (Page 3045)

MC_Halt: Stop axis V4...5 (Page 3050)

MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)

MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)

MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)

MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)

MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)

MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)
MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)
S7-1200 Motion Control V1...3 (Page 3073)

MC_MoveJog: Function chart V4...5 (S7-1200)

Function chart

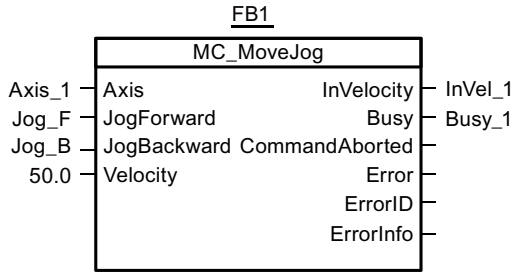

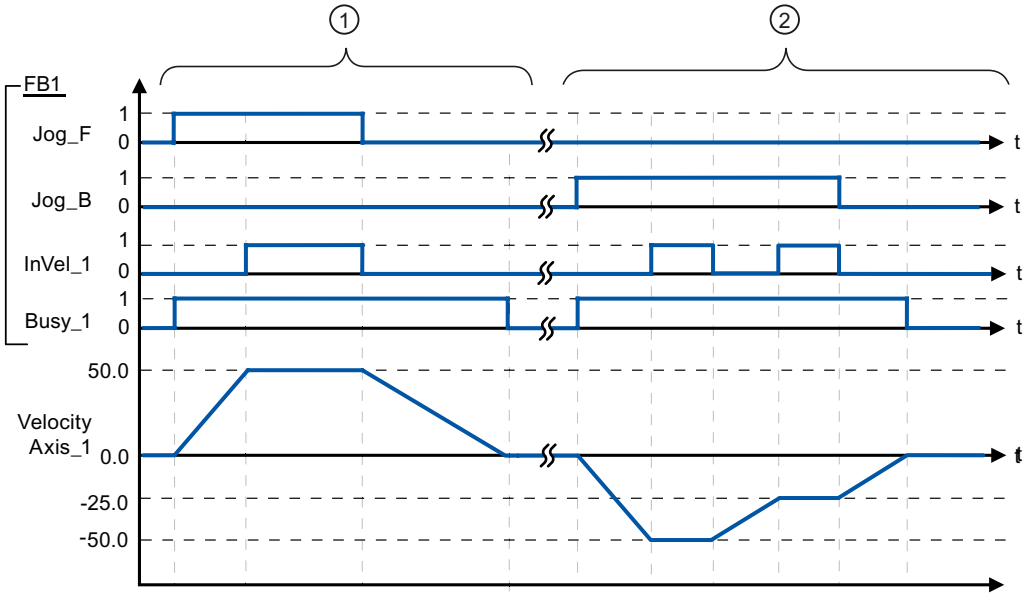

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is moved in the positive direction in jog mode via "Jog_F". When the target velocity 50.0 is reached, this is signaled via "InVel_1". After "Jog_F" is reset, the axis is braked to a standstill.
②	The axis is moved in the negative direction in jog mode via "Jog_B". When the target velocity -50.0 is reached, this is signaled via "InVel_1". When "Jog_B" is set, the value at parameter "Velocity" changes to 25.0. "InVel_1" is reset and the axis is braked. When the new target velocity -25.0 is reached, this is signaled via "InVel_1". After "Jog_B" is reset, the axis is braked to a standstill.

See also

MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)

MC_CommandTable (S7-1200)

MC_CommandTable: Run axis commands as motion sequence V4...5 (S7-1200)

Description

The Motion Control instruction "MC_CommandTable" combines multiple individual axis control commands in one movement sequence. "MC_CommandTable" is available for axes with drive connection via PTO (Pulse Train Output).

Requirements

- The positioning axis technology object has been inserted and correctly configured.
- The drive is connected via PTO (Pulse Train Output).
- The command table technology object has been inserted and correctly configured.
- The axis is enabled.

Override response

The MC_CommandTable command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_CommandTable command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The active Motion Control command is canceled by the start of the first "Positioning Relative", "Positioning Absolute", "Velocity set point" or "Halt" command.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
CommandTable	INPUT	TO_CommandTable	-	Command table technology object	
Execute	INPUT	BOOL	FALSE	Command table start with positive edge	
StartStep	INPUT	INT	1	Defines the step at which the execution of the command table should begin Limit values: 1 ≤ StartStep ≤ EndStep	
EndStep	INPUT	INT	32	Defines the step up to which the execution of command table should take place Limit values: StartStep ≤ EndStep ≤ 32	
Done	OUTPUT	BOOL	FALSE	TRUE	Command table has been successfully executed
Busy	OUTPU	BOOL	FALSE	TRUE	The command table is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The command table was canceled by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command table. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	
CurrentStep	OUTPUT	INT	0	Step in command table currently being executed	
StepCode	OUTPUT	WORD	16#0000	User-defined numerical value / bit pattern of the step currently being executed	

See also

MC_Power: Enable, disable axis V4...5 (Page 3039)

MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)

MC_Home: Home axis, set home position V4...5 (Page 3045)
MC_Halt: Stop axis V4...5 (Page 3050)
MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)
MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)
MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)
MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)
MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)
MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)
MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)
S7-1200 Motion Control V1...3 (Page 3073)

MC_ChangeDynamic (S7-1200)

MC_ChangeDynamic: Change dynamic settings of axis V4...5 (S7-1200)

Description

Motion Control instruction "MC_ChangeDynamic" allows you to change the following settings of the axis:

- Change the ramp-up time (acceleration) value
- Change the ramp-down time (deceleration) value
- Change the emergency stop ramp-down time (emergency stop deceleration) value
- Change the smoothing time (jerk) value

For the effectiveness of the change, refer to the description of the tag.

Requirements

The positioning axis technology object has been configured correctly.

Override response

A MC_ChangeDynamic command cannot be aborted by any other Motion Control command.

A new MC_ChangeDynamic command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description
Axis	INPUT	TO_SpeedAxis	-	Axis technology object
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge

Parameter	Declaration	Data type	Default value	Description	
ChangeRampUp	INPUT	BOOL	FALSE	TRUE	Change ramp-up time in line with input parameter "RampUpTime"
RampUpTime	INPUT	REAL	5.00	Time (in seconds) to accelerate axis from standstill to configured maximum velocity without jerk limit. The change will influence the tag <Axis name>. Config.DynamicDefaults.Acceleration. For the effectiveness of the change, refer to the description of this tag.	
ChangeRampDown	INPUT	BOOL	FALSE	TRUE	Change ramp-down time to correspond to input parameter "RampDownTime"
RampDownTime	INPUT	REAL	5.00	Time (in seconds) to decelerate axis from the configured maximum velocity to standstill without jerk limiter. The change will influence the tag <Axis name>. Config.DynamicDefaults.Deceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeEmergency	INPUT	BOOL	FALSE	TRUE	Change emergency stop ramp-down time in line with input parameter "EmergencyRampTime"
EmergencyRampTime	INPUT	REAL	2.00	Time (in seconds) to decelerate the axis from configured maximum velocity to standstill without jerk limiter in emergency stop mode. The change will influence the tag <Axis name>. Config.DynamicDefaults.EmergencyDeceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeJerkTime	INPUT	BOOL	FALSE	TRUE	Change smoothing time according to the input parameter "JerkTime"
JerkTime	INPUT	REAL	0.25	Smoothing time (in seconds) used for the axis acceleration and deceleration ramps The change will influence the tag <Axis name>. Config.DynamicDefaults.Jerk . For the effectiveness of the change, refer to the description of this tag.	
Done	OUTPUT	BOOL	FALSE	TRUE	The changed values have been written to the technology data block. The description of the tags will show when the change becomes effective.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Note

At the input parameters "RampUpTime", "RampDownTime", "EmergencyRampTime" and "JerkTime", values can be entered which exceed the admissible limits of the resulting parameters: "Acceleration", "Deceleration", "Emergency stop deceleration" and "Jerk".

Ensure that your inputs are within the valid range, taking into consideration the equations and limits in section "Dynamic".

See also

MC_Power: Enable, disable axis V4...5 (Page 3039)
MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)
MC_Home: Home axis, set home position V4...5 (Page 3045)
MC_Halt: Stop axis V4...5 (Page 3050)
MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)
MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)
MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)
MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)
MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)
MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)
MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)
S7-1200 Motion Control V1...3 (Page 3073)

MC_ReadParam (S7-1200)

MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (S7-1200)

Description

The Motion Control instruction "MC_ReadParam" enables continuous reading of motion data and status messages of an axis. The current value of the corresponding tags is determined at the start of the command.

The following motion data and status messages can be read:

- As of technology version V4:
  - Setpoint position of the axis
  - Setpoint and actual velocity of the axis
  - Current distance of axis from target position
  - Target position of the axis
- Additional as of technology version V5:
  - Actual position of the axis
  - Actual velocity of the axis
  - Current following error
  - Drive status
  - Encoder status
  - Status bits
  - Error bits

Requirements

The positioning axis technology object has been configured correctly.

Override response

A MC_ReadParam command cannot be aborted by any other Motion Control command.

A new MC_ReadParam command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Enable	INPUT	BOOL	FALSE	TRUE	Read the tag specified with the "Parameter" and store the value in the destination address specified with "Value".
				FALSE	Do not update assigned motion data
Parameter	INPUT	VARIANT (REAL)	-	VARIANT pointer to the value to be read. The following tags are permitted: • <Axis name>.Position • <Axis name>.Velocity • <Axis name>.ActualPosition • <Axis name>.ActualVelocity • <Axis name>.StatusPositioning.<Tag name> • <Axis name>.StatusDrive.<Tag name> • <Axis name>.StatusSensor.<Tag name> • <Axis name>.StatusBits.<Tag name> • <Axis name>.ErrorBits.<Tag name> The description of the tags named and the tag structures can be found in the Appendix AUTOHOTSPOT.	
Value	INOUT	VARIANT (REAL)	-	VARIANT pointer to the target tag or destination address to which the read value is to be written.	
Valid	OUTPUT	BOOL	FALSE	TRUE	The read value is valid.
				FALSE	The read value is invalid.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_Power: Enable, disable axis V4...5 (Page 3039)

MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)

MC_Home: Home axis, set home position V4...5 (Page 3045)

MC_Halt: Stop axis V4...5 (Page 3050)
MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)
MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)
MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)
MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)
MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)
MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)
MC_WriteParam: Write tag of positioning axis V4...5 (Page 3072)
S7-1200 Motion Control V1...3 (Page 3073)

MC_WriteParam (S7-1200)

MC_WriteParam: Write tag of positioning axis V4...5 (S7-1200)

Description

Motion Control instruction "MC_WriteParam" enables the writing of tags of the positioning axis technology object in the user program. In contrast to the value assignment of the tags in the user program, "MC_WriteParam" can also change values of read-only tags.

You can learn about the tags, the conditions under which they can be written and the time at which they take effect in the description of the technology object tags.

Requirements

- The positioning axis technology object has been configured correctly.
- To write tags that are read-only in the user program, the axis must be disabled.

Override response

A MC_WriteParam command cannot be aborted by any other Motion Control command.

A new MC_WriteParam command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description
Parameter	INPUT	VARIANT (BOOL, INT, DINT, UDINT*, REAL) *) as of V5	-	VARIANT pointer to the technology object tag positioning axis (destination address) to be written
Value	INPUT	VARIANT (BOOL, INT, DINT, UDINT*, REAL) *) as of V5	-	VARIANT pointer to the value to be written (source address)

Parameter	Declaration	Data type	Default value	Description	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Value was written
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

MC_Power: Enable, disable axis V4...5 (Page 3039)
MC_Reset: Acknowledge fault, restart technology object V4...5 (Page 3044)
MC_Home: Home axis, set home position V4...5 (Page 3045)
MC_Halt: Stop axis V4...5 (Page 3050)
MC_MoveAbsolute: Absolute positioning of axis V4...5 (Page 3053)
MC_MoveRelative: Relative positioning of axis V4...5 (Page 3056)
MC_MoveVelocity: Move axis at set velocity V4...5 (Page 3059)
MC_MoveJog: Move axis in jog mode V4...5 (Page 3063)
MC_CommandTable: Run axis commands as motion sequence V4...5 (Page 3066)
MC_ChangeDynamic: Change dynamic settings of axis V4...5 (Page 3068)
MC_ReadParam: Continuously read motion data of a positioning axis V4...5 (Page 3070)
S7-1200 Motion Control V1...3 (Page 3073)

S7-1200 Motion Control V1...3 (S7-1200)

MC_Power (S7-1200)

MC_Power: Enable, disable axis V1...3 (S7-1200)

Description

The Motion Control instruction "MC_Power" enables or disables an axis.

Requirements

- The axis technology object has been configured correctly.
- There is no pending enable-inhibiting error.

Override response

Execution of "MC_Power" cannot be aborted by a Motion Control command.

Disabling the axis (input parameter "Enable" = FALSE) aborts all Motion Control commands for the associated technology object in accordance with the selected "StopMode".

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Enable	INPUT	BOOL	FALSE	TRUE	Motion Control attempts to enable the axis.
				FALSE	All current jobs are interrupted in accordance with the "StopMode" configured. The axis is stopped and disabled.
StopMode	INPUT	INT	0	0	Emergency stop If a request to disable the axis is pending, the axis brakes at the configured emergency stop deceleration. The axis is disabled after reaching standstill.
				1	Immediate stop If a request to disable the axis is pending, this axis is disabled without deceleration. The pulse output is stopped immediately.
				2	Emergency stop with jerk control If a request to disable the axis is pending, the axis brakes at the configured emergency stop deceleration. If the jerk control is activated, the configured jerk is taken into account. The axis is disabled after reaching standstill.

Parameter	Declaration	Data type	Default value	Description	
Status	OUTPUT	BOOL	FALSE	Status of axis enable	
				FALSE	The axis is disabled. The axis does not execute Motion Control commands and does not accept any new commands (exception: MC_Reset command). The axis is not homed. Upon disabling, the status does not change to FALSE until the axis reaches a standstill.
				TRUE	The axis is enabled. The axis is ready to execute Motion Control commands. Upon axis enabling, the status does not change to TRUE until the signal "Drive ready" is pending. If the "Drive ready" drive interface was not configured in the axis configuration, the status changes to TRUE immediately.
Busy	OUTPUT	BOOL	FALSE	TRUE	"MC_Power" is active.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction "MC_Power" or in the associated technology object. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Note

If the axis is switched off due to an error, it will be enabled again automatically after the error has been eliminated and acknowledged. This requires that the input parameter "Enable" has retained the value TRUE during this process.

Enabling an axis with configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE.
   The enable output for "Drive enabled" changes to TRUE to enable the power to the drive.
   The CPU waits for the "Drive ready" signal of the drive.
   When the "Drive ready" signal is available at the configured Ready input of the CPU, the axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Enabling an axis without configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE. The axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Disabling an axis

To disable an axis, you can follow the steps described below:

1. Bring the axis to a standstill.
   You can identify when the axis is at a standstill in the tag of the technology object <Axis name>.StatusBits.StandStill.
2. Set input parameter "Enable" to FALSE after standstill is reached.
3. If output parameters "Busy" and "Status" and the tag of technology object <Axis name>.StatusBits.Enable indicate the value FALSE, disabling of the axis is complete.

See also

S7-1200 Motion Control V4...5 (Page 3039)

MC_Power: Function chart V1...3 (Page 3077)

MC_Reset: Acknowledge fault, restart technology object V1...3 (Page 3078)

MC_Home: Home axis, set home position V1...3 (Page 3079)

MC_Halt: Stop axis V1...3 (Page 3082)

MC_MoveAbsolute: Absolute positioning of axis V1...3 (Page 3086)

MC_MoveRelative: Relative positioning of axis V1...3 (Page 3089)

MC_MoveVelocity: Move axis at set velocity V1...3 (Page 3092)

MC_MoveJog: Move axis in jog mode V1...3 (Page 3096)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3099)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3101)

MC_Power: Function chart V1...3 (S7-1200)

Function chart

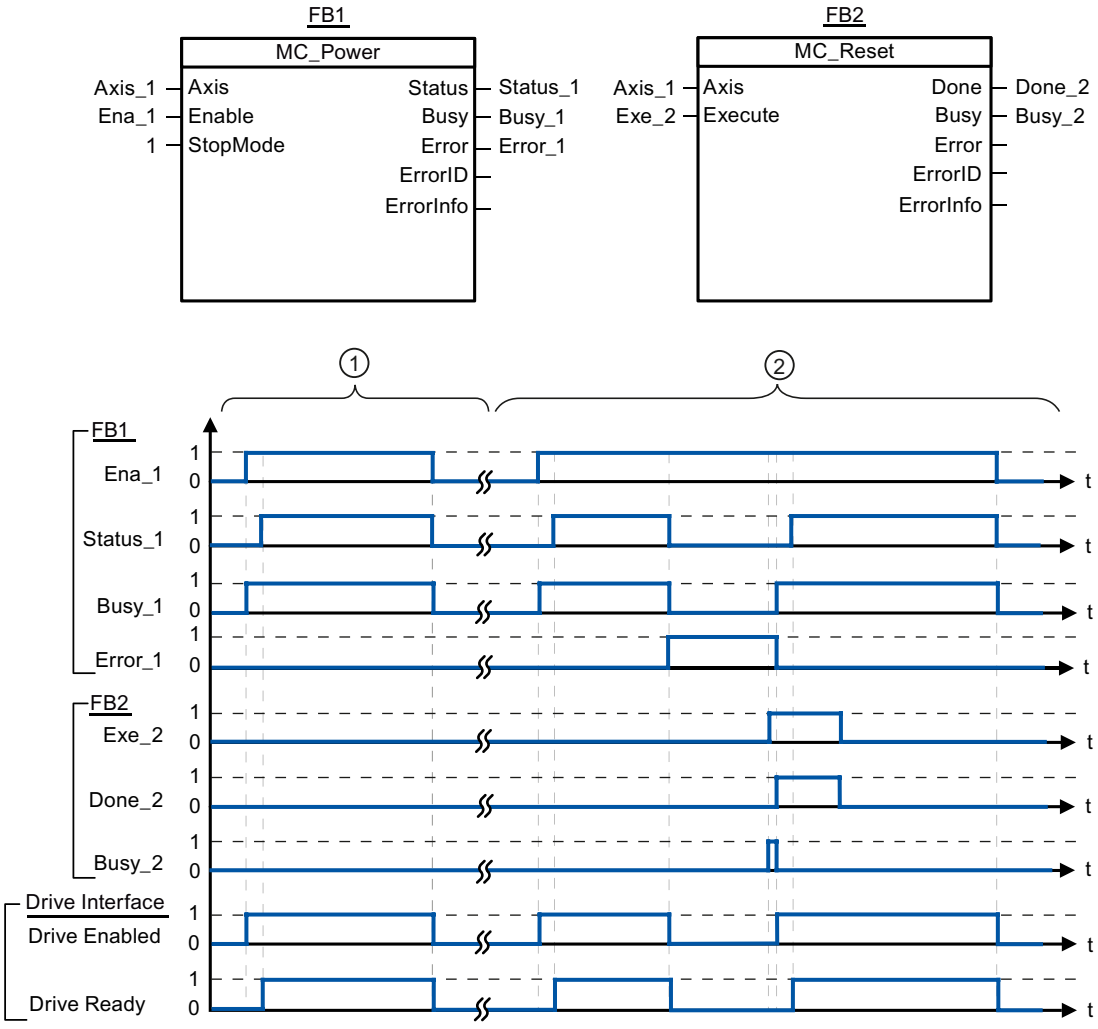

①	An axis is enabled and then disabled again. When the drive has signaled "Drive ready" back to the CPU, the successful enable can be read out via "Status_1".
②	Following an axis enable, an error has occurred that caused the axis to be disabled. The error is eliminated and acknowledged with "MC_Reset". The axis is then enabled again.

See also

MC_Power: Enable, disable axis V1...3 (Page 3073)

MC_Reset (S7-1200)

MC_Reset: Acknowledge fault, restart technology object V1...3 (S7-1200)

Description

Motion Control instruction "MC_Reset" can be used to acknowledge "Operating error with axis stop" and "Configuration error". The errors that require acknowledgement can be found in the "List of ErrorIDs and ErrorInfos" under "Remedy".

From version V3.0, the axis configuration can be downloaded to the work memory in RUN operating mode.

Requirements

- The axis technology object has been configured correctly.
- The cause of a pending configuration error requiring acknowledgement has been eliminated (for example, acceleration in positioning axis technology object has been changed to a valid value).

Override response

The MC_Reset command cannot be aborted by any other Motion Control command.

The new MC_Reset command does not abort any other active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Restart	INPUT	BOOL	FALSE	(From version V3.0)	
				TRUE	Download the axis configuration from the load memory to the work memory. The command can only be executed when the axis is disabled. Refer to the notes on Download to the CPU.
				FALSE	Acknowledges pending errors
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Acknowledging an error requiring acknowledgement with MC_Reset

To acknowledge an error, follow these steps:

1. Check the requirements indicated above.
2. Start the acknowledgement of the error with a rising edge at input parameter "Execute".
3. If output parameter "Done" indicates the value TRUE and tag of technology object <Axis name>.StatusBits.Error the value FALSE, the error has been acknowledged.

See also

S7-1200 Motion Control V4...5 (Page 3039)

MC_Power: Enable, disable axis V1...3 (Page 3073)

MC_Home: Home axis, set home position V1...3 (Page 3079)

MC_Halt: Stop axis V1...3 (Page 3082)

MC_MoveAbsolute: Absolute positioning of axis V1...3 (Page 3086)

MC_MoveRelative: Relative positioning of axis V1...3 (Page 3089)

MC_MoveVelocity: Move axis at set velocity V1...3 (Page 3092)

MC_MoveJog: Move axis in jog mode V1...3 (Page 3096)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3099)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3101)

MC_Home (S7-1200)

MC_Home: Home axis, set home position V1...3 (S7-1200)

Description

Motion Control instruction "MC_Home" is used to match the axis coordinates to the real, physical drive position. Homing is required for absolute positioning of the axis. The following types of homing can be executed:

- Active homing (Mode = 3)
  The homing procedure is executed automatically.
- Passive homing (Mode = 2)
  During passive homing, the "MC_Home" Motion Control instruction does not carry out any homing motion. The traversing motion required for this must be implemented by the user via other Motion Control instructions. When the homing switch is detected, the axis is homed.
- Direct homing absolute (Mode = 0)
  The current axis position is set to the value of parameter "Position".
- Direct homing relative (Mode = 1)
  The current axis position is offset by the value of parameter "Position".

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.
- No MC_CommandTable command may be active upon start with Mode = 0, 1, or 2.

Override response

The override response depends on the selected mode:

Mode = 0, 1

The MC_Home command cannot be aborted by any other Motion Control command.

The MC_Home command does not abort any active Motion Control commands. Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 2

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 2, 3

The new MC_Home command aborts the following active Motion Control command:

- MC_Home command Mode = 2

Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 3

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Home command aborts the following active Motion Control commands:

- MC_Home command Mode = 2, 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	• Mode = 0, 2, and 3 Absolute position of axis after completion of the homing operation • Mode = 1 Correction value for the current axis position Limit values: -1.0E+12 ≤ Position ≤ 1.0E+12	
Mode	INPUT	INT	0	Homing mode	
				0	Direct homing absolute New axis position is the position value of parameter "Position".
				1	Direct homing relative New axis position is the current axis position + position value of parameter "Position".
				2	Passive homing Homing according to the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
				3	Active homing Homing procedure in accordance with the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
Done	OUTPUT	BOOL	FALSE	TRUE	Command completed
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Note

Axis homing is lost under the following conditions:

- Disabling of axis by the "MC_Power" Motion Control instruction
- Changeover between automatic mode and manual control
- Upon start of active homing. After successful completion of the homing operation, axis homing is again available.
- After POWER OFF -> POWER ON of the CPU
- After CPU restart (RUN-STOP -> STOP-RUN)

Homing an axis

To home the axis, follow these stops:

1. Check the requirements indicated above.
2. Initialize the necessary input parameters with values, and start the homing operation with a rising edge at input parameter "Execute"
3. If output parameter "Done" and technology object tag <Axis name>.StatusBits.HomingDone indicate the value TRUE, homing is complete.

See also

S7-1200 Motion Control V4...5 (Page 3039)

MC_Power: Enable, disable axis V1...3 (Page 3073)

MC_Reset: Acknowledge fault, restart technology object V1...3 (Page 3078)

MC_Halt: Stop axis V1...3 (Page 3082)

MC_MoveAbsolute: Absolute positioning of axis V1...3 (Page 3086)

MC_MoveRelative: Relative positioning of axis V1...3 (Page 3089)

MC_MoveVelocity: Move axis at set velocity V1...3 (Page 3092)

MC_MoveJog: Move axis in jog mode V1...3 (Page 3096)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3099)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3101)

MC_Halt (S7-1200)

MC_Halt: Stop axis V1...3 (S7-1200)

Description

The "MC_Halt" Motion Control instruction stops all movements and brings the axis to a standstill with the configured deceleration. The standstill position is not defined.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_Halt command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Halt command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

S7-1200 Motion Control V4...5 (Page 3039)
MC_Halt: Function chart V1...3 (Page 3085)
MC_Power: Enable, disable axis V1...3 (Page 3073)
MC_Reset: Acknowledge fault, restart technology object V1...3 (Page 3078)
MC_Home: Home axis, set home position V1...3 (Page 3079)
MC_MoveAbsolute: Absolute positioning of axis V1...3 (Page 3086)
MC_MoveRelative: Relative positioning of axis V1...3 (Page 3089)
MC_MoveVelocity: Move axis at set velocity V1...3 (Page 3092)
MC_MoveJog: Move axis in jog mode V1...3 (Page 3096)
MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3099)
MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3101)

MC_Halt: Function chart V1...3 (S7-1200)

Function chart

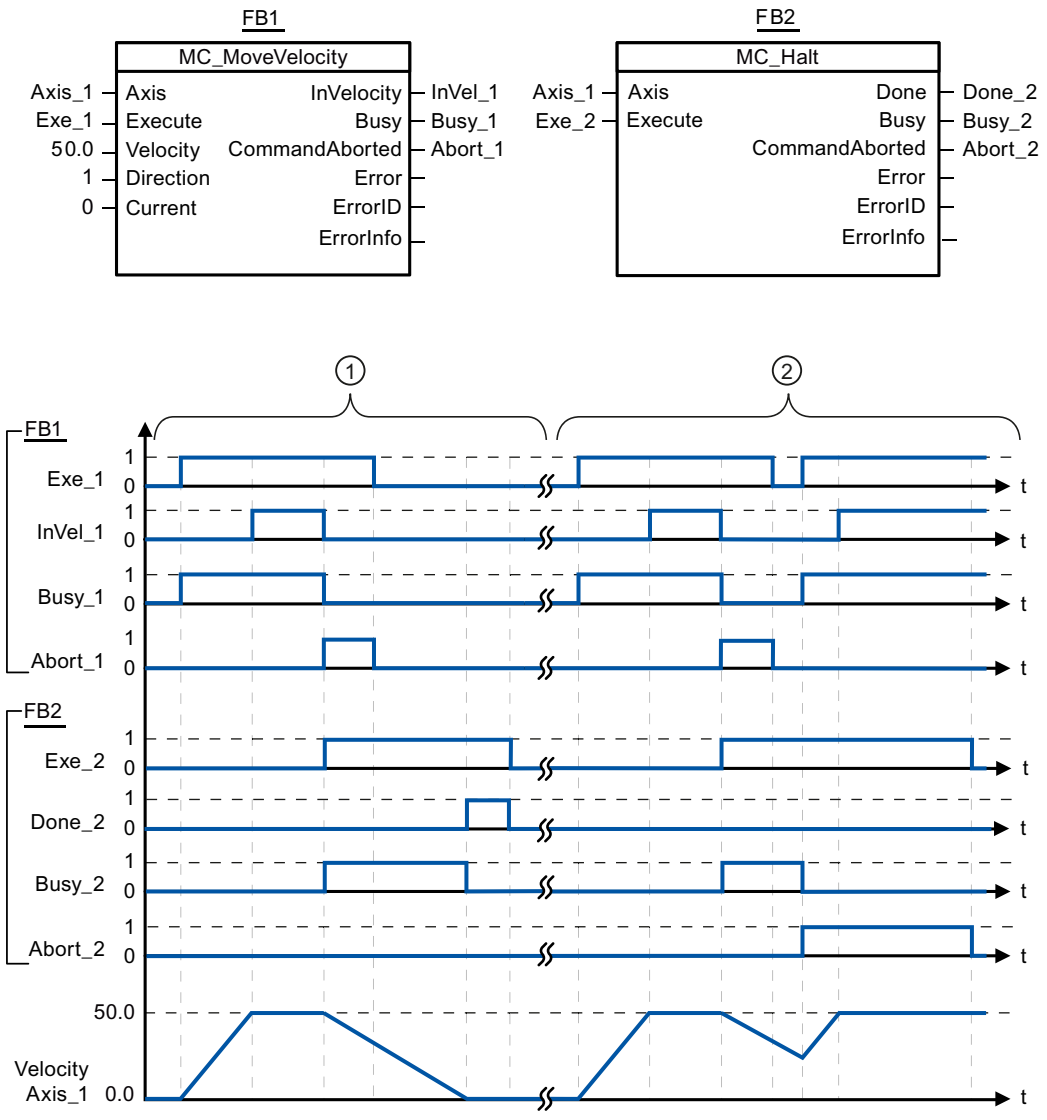

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is braked by an MC_Halt command until it comes to a standstill. The axis standstill is signaled via "Done_2".
②	While an MC_Halt command is braking the axis, this command is aborted by another motion command. The abort is signaled via "Abort_2".

See also

MC_Halt: Stop axis V1...3 (Page 3082)

MC_MoveAbsolute (S7-1200)

MC_MoveAbsolute: Absolute positioning of axis V1...3 (S7-1200)

Description

The "MC_MoveAbsolute" Motion Control instruction starts an axis positioning motion to move it to an absolute position.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.
- The axis is homed.

Override response

The MC_MoveAbsolute command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveAbsolute command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	Absolute target position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the target position to be approached. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Absolute target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

S7-1200 Motion Control V4...5 (Page 3039)

MC_MoveAbsolute: Function chart V1...3 (Page 3088)

MC_Power: Enable, disable axis V1...3 (Page 3073)

MC_Reset: Acknowledge fault, restart technology object V1...3 (Page 3078)

MC_Home: Home axis, set home position V1...3 (Page 3079)

MC_Halt: Stop axis V1...3 (Page 3082)

MC_MoveRelative: Relative positioning of axis V1...3 (Page 3089)

MC_MoveVelocity: Move axis at set velocity V1...3 (Page 3092)

MC_MoveJog: Move axis in jog mode V1...3 (Page 3096)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3099)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3101)

MC_MoveAbsolute: Function chart V1...3 (S7-1200)

Function chart

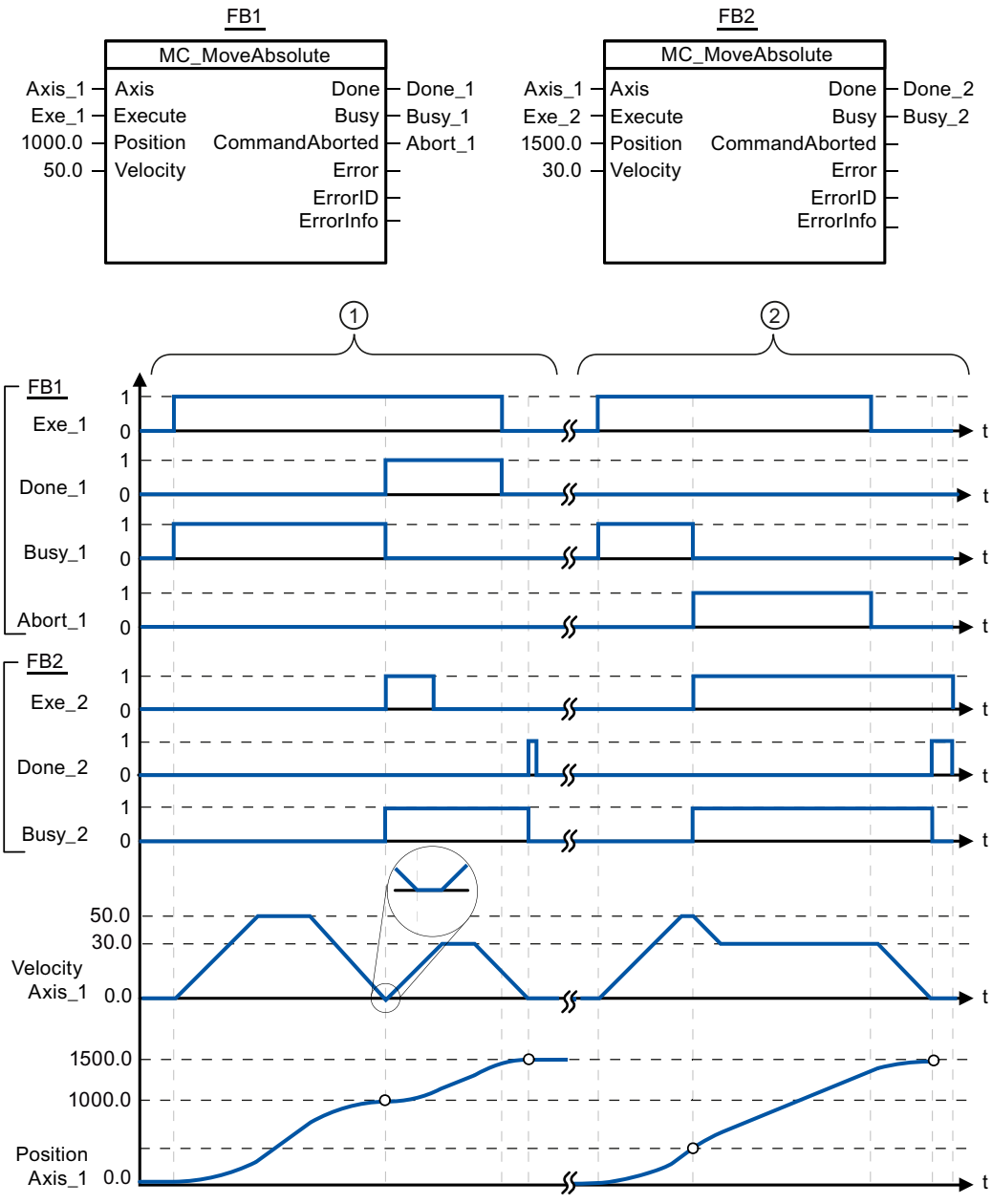

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An axis is moved to absolute position 1000.0 with an MC_MoveAbsolute command. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveAbsolute command, with target position 1500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveAbsolute command is aborted by another MC_MoveAbsolute command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveAbsolute: Absolute positioning of axis V1...3 (Page 3086)

MC_MoveRelative (S7-1200)

MC_MoveRelative: Relative positioning of axis V1...3 (S7-1200)

Description

The "MC_MoveRelative" Motion Control instruction starts a positioning motion relative to the start position.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveRelative command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveRelative command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command

- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Distance	INPUT	REAL	0.0	Travel distance for the positioning operation Limit values: $-1.0e^{12} \leq$ Distance $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the distance to be traveled. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

S7-1200 Motion Control V4...5 (Page 3039)

MC_MoveRelative: Function chart V1...3 (Page 3091)

MC_Power: Enable, disable axis V1...3 (Page 3073)

MC_Reset: Acknowledge fault, restart technology object V1...3 (Page 3078)

MC_Home: Home axis, set home position V1...3 (Page 3079)

MC_Halt: Stop axis V1...3 (Page 3082)

MC_MoveAbsolute: Absolute positioning of axis V1...3 (Page 3086)

MC_MoveVelocity: Move axis at set velocity V1...3 (Page 3092)

MC_MoveJog: Move axis in jog mode V1...3 (Page 3096)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3099)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3101)

MC_MoveRelative: Function chart V1...3 (S7-1200)

Function chart

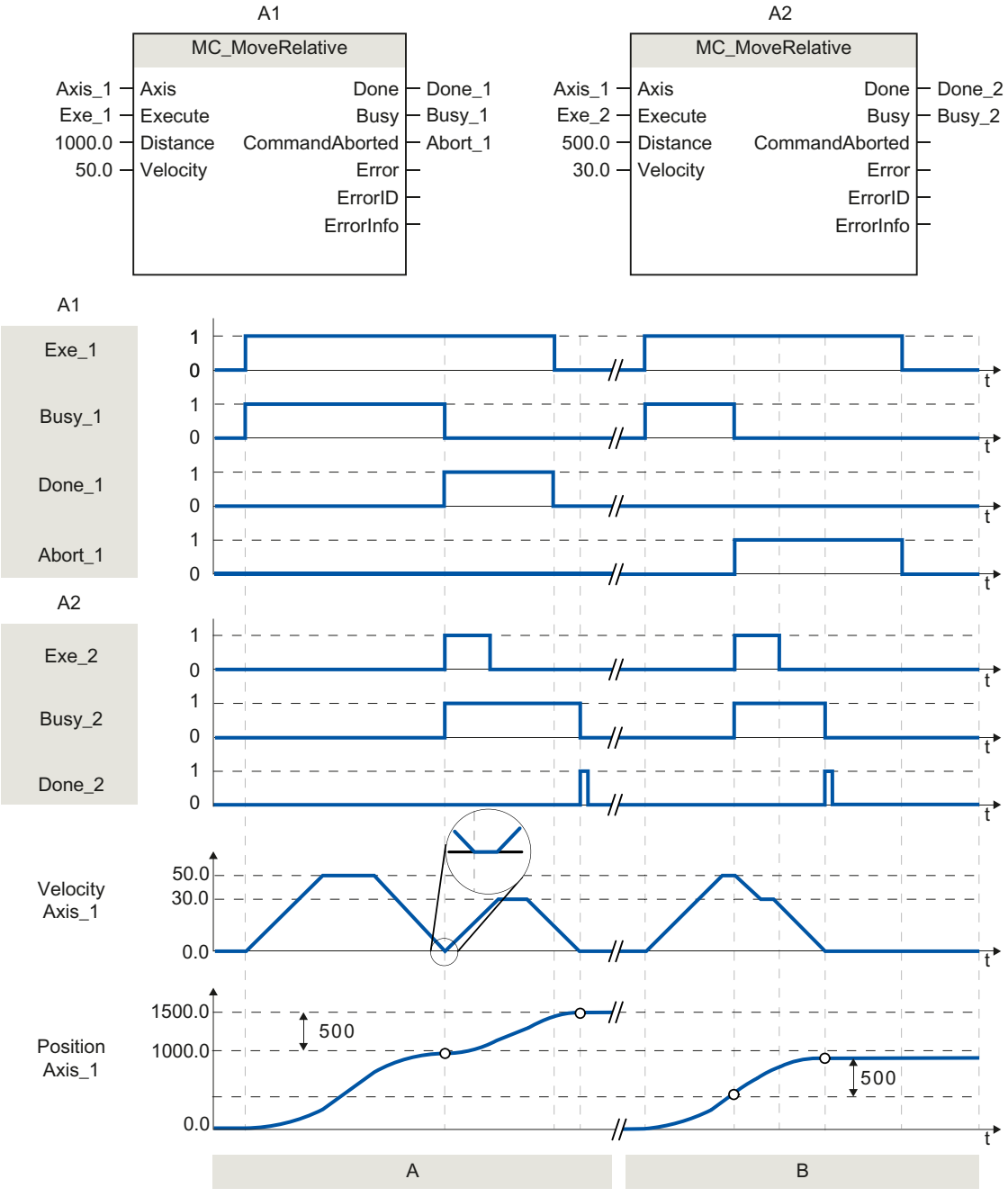

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	The axis is moved by an MC_MoveRelative command by the distance ("Distance") 1000.0. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveRelative command, with travel distance 500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveRelative command is aborted by another MC_MoveRelative command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity by the new distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveRelative: Relative positioning of axis V1...3 (Page 3089)

MC_MoveVelocity (S7-1200)

MC_MoveVelocity: Move axis at set velocity V1...3 (S7-1200)

Description

Motion control instruction "MC_MoveVelocity" moves the axis constantly at the specified velocity.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

MC_MoveVelocity can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveVelocity command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command

- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Velocity	INPUT	REAL	10.0	Velocity specification for axis motion Limit values: Start/stop velocity ≤ |Velocity ≤ maximum velocity (Velocity = 0.0 is permitted)	
Direction	INPUT	INT	0	Direction specification	
				0	Direction of rotation corresponds to the sign of the value in parameter "Velocity"
				1	Positive direction of rotation (The sign of the value in parameter "Velocity" is ignored)
				2	Negative direction of rotation (The sign of the value in parameter "Velocity" is ignored)
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	"Maintain current velocity" is deactivated. The values of parameters "Velocity" and "Direction" are used.
				TRUE	"Maintain current velocity" is activated. The values in parameters "Velocity" and "Direction" are not taken into account. When the axis resumes motion at the current velocity, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUTPUT	BOOL	FALSE	TRUE	• "Current" = FALSE: The velocity specified in parameter "Velocity" was reached. • "Current" = TRUE: The axis travels at the current velocity at the start time.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Behavior with zero setpoint velocity (Velocity = 0.0)

An MC_MoveVelocity command with "Velocity" = 0.0 (such as an MC_Halt command) aborts active motion commands and stops the axis with the configured deceleration.

When the axis comes to a standstill, output parameter "InVelocity" indicates TRUE for at least one program cycle.

"Busy" indicates the value TRUE during the deceleration process and changes to FALSE together with "InVelocity". If parameter "Execute" = TRUE is set, "InVelocity" and "Busy" are latched.

When the "MC_MoveVelocity" command is started, status bit "SpeedCommand" is set in the technology object. Status bit "ConstantVelocity" is set upon axis standstill. Both bits are adapted to the new situation when a new motion command is started.

See also

S7-1200 Motion Control V4...5 (Page 3039)

MC_MoveVelocity: Function chart V1...3 (Page 3095)

MC_Power: Enable, disable axis V1...3 (Page 3073)

MC_Reset: Acknowledge fault, restart technology object V1...3 (Page 3078)

MC_Home: Home axis, set home position V1...3 (Page 3079)

MC_Halt: Stop axis V1...3 (Page 3082)

MC_MoveAbsolute: Absolute positioning of axis V1...3 (Page 3086)

MC_MoveRelative: Relative positioning of axis V1...3 (Page 3089)

MC_MoveJog: Move axis in jog mode V1...3 (Page 3096)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3099)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3101)

MC_MoveVelocity: Function chart V1...3 (S7-1200)

Function chart

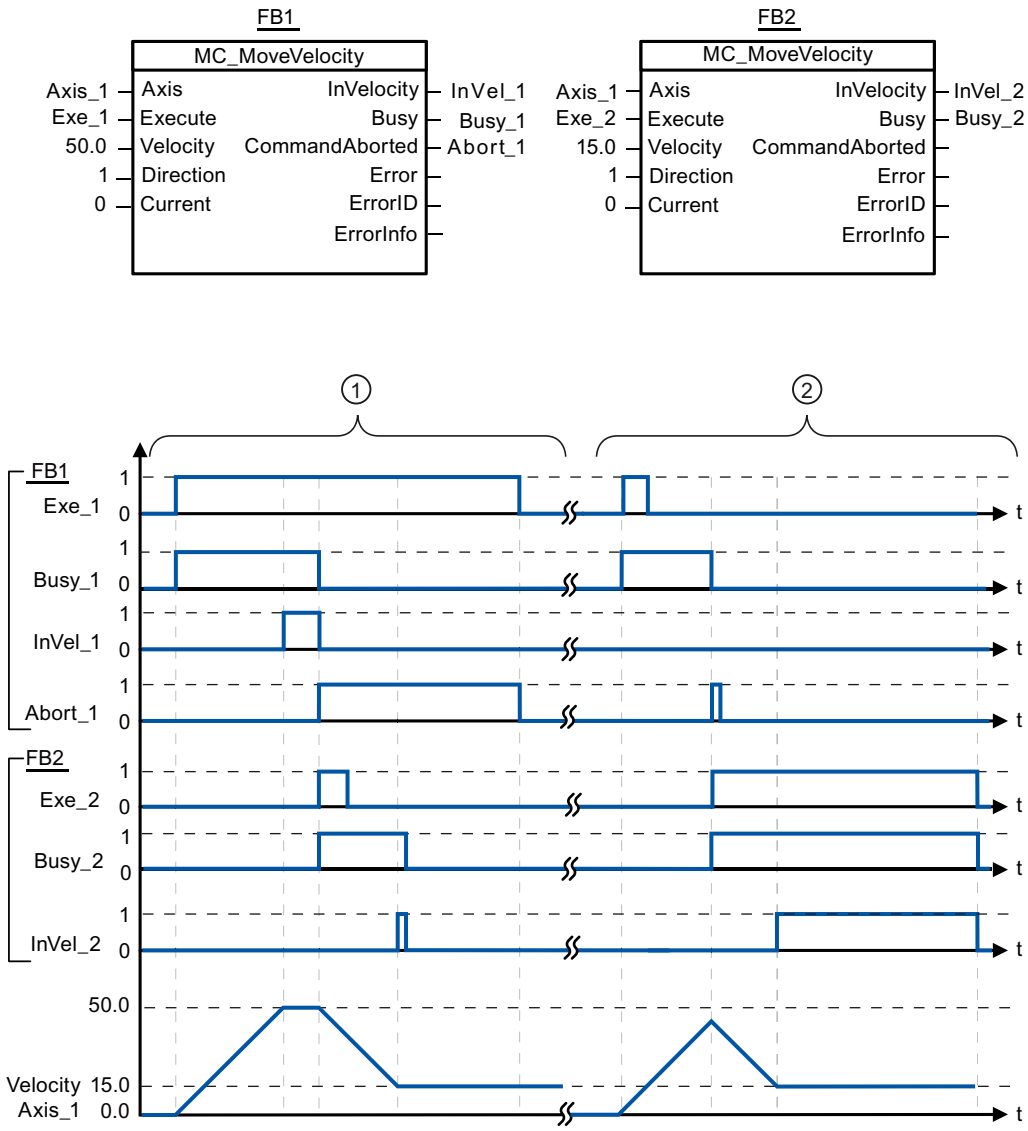

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An active MC_MoveVelocity command signals via "InVel_1" that its target velocity has been reached. It is then aborted by another MC_MoveVelocity command. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.
②	An active MC_MoveVelocity command is aborted by another MC_MoveVelocity command prior to reaching its target velocity. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.

See also

MC_MoveVelocity: Move axis at set velocity V1...3 (Page 3092)

MC_MoveJog (S7-1200)

MC_MoveJog: Move axis in jog mode V1...3 (S7-1200)

Description

Motion control instruction "MC_MoveJog" moves the axis constantly at the specified velocity in jog mode. You use this Motion Control instruction, for example, for testing and commissioning purposes.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveJog command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveJog command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".	
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".	
If both parameters are simultaneously TRUE, the axis stops with the configured deceleration. An error is indicated in parameters "Error", "ErrorID", and "ErrorInfo".					
Velocity	INPUT	REAL	10.0	Preset velocity for jog mode	
				Limit values, instruction version V1.0: Start/stop velocity ≤ |Velocity ≤ maximum velocity	
				Limits, instruction version V2.0: Start/stop velocity ≤ velocity ≤ maximum velocity	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity specified in parameter "Velocity" was reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution, the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

See also

S7-1200 Motion Control V4...5 (Page 3039)

MC_MoveJog: Function chart V1...3 (Page 3098)

MC_Power: Enable, disable axis V1...3 (Page 3073)

MC_Reset: Acknowledge fault, restart technology object V1...3 (Page 3078)

MC_Home: Home axis, set home position V1...3 (Page 3079)

MC_Halt: Stop axis V1...3 (Page 3082)

MC_MoveAbsolute: Absolute positioning of axis V1...3 (Page 3086)

MC_MoveRelative: Relative positioning of axis V1...3 (Page 3089)

MC_MoveVelocity: Move axis at set velocity V1...3 (Page 3092)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3099)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3101)

MC_MoveJog: Function chart V1...3 (S7-1200)

Function chart

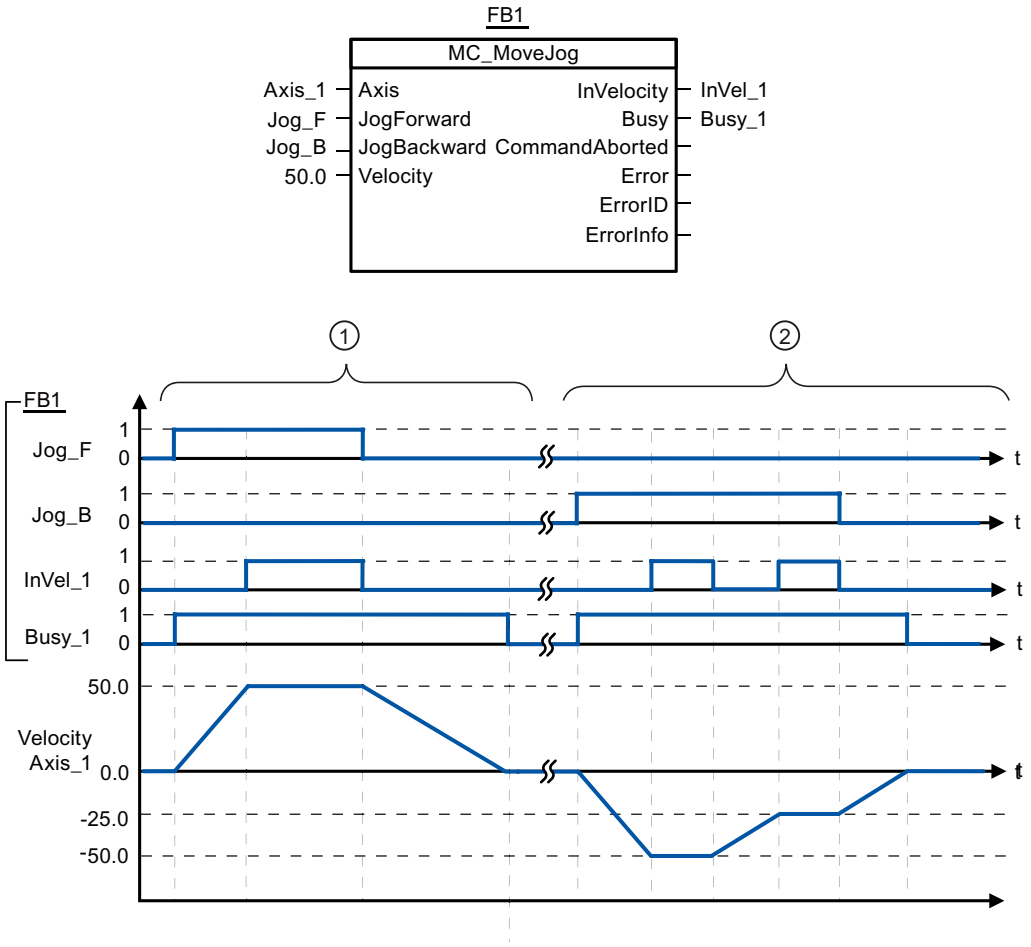

The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is moved in the positive direction in jog mode via "Jog_F". When the target velocity 50.0 is reached, this is signaled via "InVelo_1". After " Jog_F" is reset, the axis brakes again until it comes to a standstill.
②	The axis is moved in the negative direction in jog mode via "Jog_B". When the target velocity 50.0 is reached, this is signaled via "InVelo_1". After " Jog_B" is reset, the axis brakes again until it comes to a standstill.

See also

MC_MoveJog: Move axis in jog mode V1...3 (Page 3096)

MC_CommandTable (S7-1200)

MC_CommandTable: Run axis commands as motion sequence V2...3 (S7-1200)

Description

The Motion Control instruction "MC_CommandTable" combines multiple individual axis control commands in one movement sequence.

Requirements

- The axis technology object has been inserted in V2 and correctly configured.
- The command table technology object has been inserted and correctly configured.
- The axis is enabled.

Override response

The MC_CommandTable command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_CommandTable command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The active Motion Control command is cancelled by the start of the first "Positioning Relative", "Positioning Absolute", "Velocity set point" or "Halt" command.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
CommandTable	INPUT	TO_CommandTable_1	-	Command table technology object	
Execute	INPUT	BOOL	FALSE	Command table start with positive edge	
StartStep	INPUT	INT	1	Defines the step at which the execution of the command table should begin Limit values: 1 ≤ StartStep ≤ EndStep	
EndStep	INPUT	INT	32	Defines the step up to which the execution of command table should take place Limit values: StartStep ≤ EndStep ≤ 32	
Done	OUTPUT	BOOL	FALSE	TRUE	Command table has been successfully executed
Busy	OUTPU	BOOL	FALSE	TRUE	The command table is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The command table was cancelled by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command table. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	
CurrentStep	OUTPUT	INT	0	Step in command table currently being executed	
StepCode	OUTPUT	WORD	16#0000	User-defined numerical value / bit pattern of the step currently being executed	

See also

S7-1200 Motion Control V4...5 (Page 3039)

MC_Power: Enable, disable axis V1...3 (Page 3073)

MC_Reset: Acknowledge fault, restart technology object V1...3 (Page 3078)

MC_Home: Home axis, set home position V1...3 (Page 3079)

MC_Halt: Stop axis V1...3 (Page 3082)

MC_MoveAbsolute: Absolute positioning of axis V1...3 (Page 3086)

MC_MoveRelative: Relative positioning of axis V1...3 (Page 3089)

MC_MoveVelocity: Move axis at set velocity V1...3 (Page 3092)

MC_MoveJog: Move axis in jog mode V1...3 (Page 3096)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 3101)

MC_ChangeDynamic (S7-1200)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (S7-1200)

Description

Motion Control instruction "MC_ChangeDynamic" allows you to change the following settings of the axis:

- Change the ramp-up time (acceleration) value
- Change the ramp-down time (deceleration) value
- Change the emergency stop ramp-down time (emergency stop deceleration) value
- Change the smoothing time (jerk) value

For the effectiveness of the change, refer to the description of the tag.

Requirements

- The axis technology object has been inserted in Version V2.
- The axis technology object has been configured correctly.

Override response

A MC_ChangeDynamic command cannot be aborted by any other Motion Control command.

A new MC_ChangeDynamic command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
ChangeRampUp	INPUT	BOOL	FALSE	TRUE	Change ramp-up time in line with input parameter "RampUpTime"
RampUpTime	INPUT	REAL	5.00	Time (in seconds) to accelerate axis from standstill to configured maximum velocity without jerk limit. The change will influence the tag <Axis name>. Config.DynamicDefaults.Acceleration. For the effectiveness of the change, refer to the description of this tag.	
ChangeRampDown	INPUT	BOOL	FALSE	TRUE	Change ramp-down time in line with input parameter "RampDownTime"
RampDownTime	INPUT	REAL	5.00	Time (in seconds) to decelerate axis from the configured maximum velocity to standstill without jerk limiter. The change will influence the tag <Axis name>. Config.DynamicDefaults.Deceleration . For the effectiveness of the change, refer to the description of this tag.	

Parameter	Declaration	Data type	Default value	Description	
ChangeEmergency	INPUT	BOOL	FALSE	TRUE	Change emergency stop ramp-down time in line with input parameter "EmergencyRampTime"
EmergencyRampTime	INPUT	REAL	2.00	Time (in seconds) to decelerate the axis from configured maximum velocity to standstill without jerk limiter in emergency stop mode. The change will influence the tag <Axis name>. Config.DynamicDefaults.EmergencyDeceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeJerkTime	INPUT	BOOL	FALSE	TRUE	Change smoothing time according to the input parameter "JerkTime"
JerkTime	INPUT	REAL	0.25	Smoothing time (in seconds) used for the axis acceleration and deceleration ramps The change will influence the tag <Axis name>. Config.DynamicDefaults.Jerk . For the effectiveness of the change, refer to the description of this tag.	
Done	OUTPUT	BOOL	FALSE	TRUE	The changed values have been written to the technology data block. The description of the tags will show when the change becomes effective.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Note

At the input parameters "RampUpTime", "RampDownTime", "EmergencyRampTime" and "JerkTime", values can be entered which exceed the admissible limits of the resulting parameters: "Acceleration", "Deceleration", "Emergency stop deceleration" and "Jerk".

Please note the equations and limits in "Axis technology object" -> "Configuring the technology object" -> "Dynamics" and ensure that the values you input are within the valid range.

See also

S7-1200 Motion Control V4...5 (Page 3039)

MC_Power: Enable, disable axis V1...3 (Page 3073)

MC_Reset: Acknowledge fault, restart technology object V1...3 (Page 3078)

MC_Home: Home axis, set home position V1...3 (Page 3079)

MC_Halt: Stop axis V1...3 (Page 3082)

MC_MoveAbsolute: Absolute positioning of axis V1...3 (Page 3086)

MC_MoveRelative: Relative positioning of axis V1...3 (Page 3089)

MC_MoveVelocity: Move axis at set velocity V1...3 (Page 3092)

MC_MoveJog: Move axis in jog mode V1...3 (Page 3096)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 3099)

S7-1500 Motion Control (S7-1500, S7-1500T)

S7-1500 Motion Control V4 (S7-1500, S7-1500T)

MC_Power V4 (S7-1500, S7-1500T)

MC_Power: Enable, disable technology object V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_Power", a technology object is enabled or disabled.

Applies to

- Synchronous axis
- Positioning axis
- Speed axis
- External encoder

Requirement

- The technology object has been configured correctly.
- Cyclic BUS communication is established between controller and encoder ("<TO>.StatusSensor[n].CommunicationOK" = TRUE).
- Cyclic BUS communication is established between controller and drive ("<TO>.StatusDrive.CommunicationOK" = TRUE).

Override response

- An "MC_Power" job cannot be aborted by any other Motion Control job.
- An "MC_Power" job with parameter "Enable"TRUE enables a technology object but does not thereby abort any other Motion Control instructions.
- Disabling the technology object (parameter "Enable" = FALSE) aborts all Motion Control jobs on the corresponding technology object in accordance with the selected "StopMode". This process cannot be canceled by the user.

Parameters

The following table shows the parameters of Motion Control instruction "MC_Power":

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The technology object is enabled.
				FALSE	The technology object is disabled. All current jobs at the technology object are aborted in accordance with the configured "StopMode".
StartMode	INPUT	DINT	1	0	Enable positioning axis/synchronous axis not position-controlled
				1	Enable positioning axis/synchronous axis position-controlled
				The parameter initially takes effect when the positioning axis is enabled ("Enable" changes from "FALSE" to "TRUE") and when the axis is enabled after acknowledgment of an interrupt that caused the axis to be disabled. This parameter is ignored when a speed axis or an external encoder is used.	

Parameters	Declaration	Data type	Default value	Description	
StopMode	INPUT	INT	0	Not applicable to the technology object external encoder If you disable a technology object with a negative edge at parameter "Enable", the axis decelerates in accordance with the selected "StopMode".	
				0	Emergency stop When the technology object is disabled, the axis brakes to a standstill without jerk limit, using the emergency deceleration configured in "Technology object > Configuration > Extended parameters > Emergency stop ramp". The enable is then canceled. (<TO>.DynamicDefaults.EmergencyDeceleration)
				1	Immediate stop When a technology object is disabled, the setpoint zero is output, and the enable is canceled. The axis is braked to a standstill according to the configuration in the drive.
				2	Stop with maximum dynamic values When the technology object is disabled, the axis is braked to a standstill using the maximum deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic limits". The configured maximum jerk is hereby taken into account. The enable is then canceled. (<TO>.DynamicLimits.MaxDeceleration; <TO>.DynamicLimits.MaxJerk)
Status	OUTPUT	BOOL	FALSE	Technology object enable status	
				FALSE	Disabled • The technology object does not accept any Motion Control jobs. • Speed control and positioning control are not active. • The actual values of the technology object are not checked for validity.
				TRUE	Enabled • The enabled technology object accepts Motion Control jobs. • Speed control and positioning control are active. • The actual values of the technology object are valid.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.

Parameters	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction "MC_Power". The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Enabling technology objects

To enable a technology object, set the "Enable" parameter to "TRUE".

The following two cases are differentiated:

- Enable at a standstill
- Enable in motion of axis

Enable at a standstill

Depending on the "StartMode" parameter, the position is held ("StartMode" = 1) or the velocity setpoint zero is output ("StartMode" = 0). When the "Status" parameter shows the value "TRUE", the technology object is enabled.

Enable in motion of axis

If "StartMode" = 1, the position at the time of the setting of the "Enable" input takes effect as the position setpoint for the position controller. The axis is braked to a standstill and adjusted to the set position depending on the maximum deceleration configured under "Technology object > Configuration > Extended parameters > Dynamic limits". If monitoring operation or dynamic limits are hereby exceeded, this leads to corresponding alarm reactions.

If "StartMode" = 0, the axis is braked as much as possible by the specification of the velocity setpoint zero. Monitoring operations and dynamic limits are not active in this case.

When the "Status" parameter shows the value "TRUE", the technology object is enabled.

Note

Automatic enable after acknowledgment of a technology alarm

If the technology object is disabled due to a technology alarm, the technology object will be enabled again automatically after the cause has been eliminated and the alarm has been acknowledged. This requires the "Enable" parameter to have retained the value "TRUE" during this process.

Disabling technology objects

To disable a technology object, set the "Enable" parameter to "FALSE".

If an axis is in motion, it is braked to a standstill according to the selected "StopMode".

When the "Busy" and "Status" parameters show the value "FALSE", the disabling of the technology object is complete.

Drive connection by means of PROFIdrive

When a drive is connected using PROFIdrive, the setpoint, enable and drive status are transmitted via the PROFIdrive telegram.

- **Enable technology object and activate drive**
  With "Enable" = TRUE parameter, the technology object is enabled. The drive is enabled according to the PROFIdrive standard.
  When the "<TO>.StatusDrive.InOperation" tag shows the value "TRUE", the drive is ready to execute setpoints. The "Status" parameter is set to the value "TRUE".
- **Disable technology object and deactivate drive**
  With the "Enable" = FALSE parameter, the "Status" parameter is set to the value "FALSE", and the axis is braked according to the selected "StopMode". The drive is disabled according to the PROFIdrive standard.

Analog drive connection

The setpoint is output via an analog output. Optionally, you can configure an enabling signal via digital output (<TO>.Actor.Interface.EnableDriveOutput), and a readiness signal via digital input (<TO>.Actor.Interface.DriveReadyInput).

- **Enable technology object and activate drive**
  With "Enable" = TRUE parameter, the enable output ("Enable drive output") is set.
  When the drive returns the readiness signal via the ready input ("Drive ready input"), the "Status" parameter and the technology object's "<TO>.StatusDrive.InOperation" tag are set to "TRUE", and the setpoint is switched to the analog output.
- **Disable technology object and deactivate drive**
  With the "Enable" = FALSE parameter, the "Status" parameter is set to the value "FALSE", and the axis is braked according to the selected "StopMode". When the setpoint zero is reached, the enable output is set to "FALSE".

Additional information

Additional information on enabling and disabling technology objects and drives can be found in the appendix MC_Power Function Charts.

MC_Power: Function chart V4 (S7-1500, S7-1500T)

Function chart: Enabling a technology object and example of alarm response

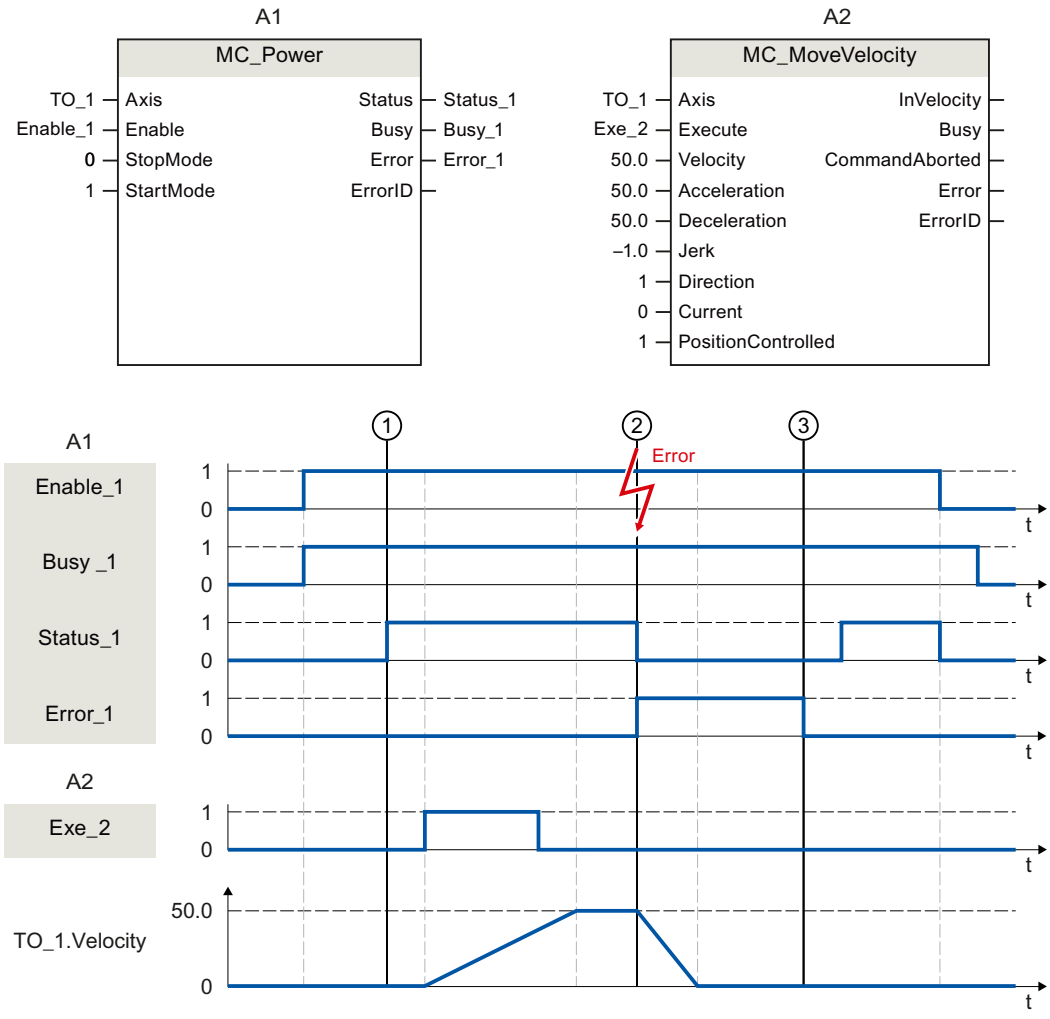

A technology object is enabled with "Enable_1= TRUE". The successful enable can be read from "Status_1" at time ①. The axis will then move with an "MC_MoveVelocity" job (A2). The velocity profile of the axis can be read from "TO_1.Velocity".

At time ② an error occurs in the technology object, which results in the disabling of the technology object (alarm response: remove enable). The axis is braked to a standstill according to the configuration in the drive. When the technology object is disabled, "Status_1" is reset. Since the axis was not disabled using "Enable_1" = FALSE, the selected "StopMode" does not apply. The cause of the error is corrected and the alarm is acknowledged at time ③.

Since "Enable_1" is still set, the technology object is enabled again. The successful enable can be read from "Status_1". Finally, the technology object is disabled with "Enable_1" = FALSE.

MC_Reset V4 (S7-1500, S7-1500T)

MC_Reset: Acknowledge alarms, restart technology object V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_Reset", you acknowledge all technology alarms that can be acknowledged in the user program. Acknowledgment also resets the "Error" and "Warning" bits in the technology data block.

With the Motion Control instruction "MC_Reset" with "Restart" = TRUE, you start reinitialization (restart) of technology objects. Upon restart of the technology object, the new configuration data are applied in the technology data block.

Applies to

- All technology objects

Requirement

- Positioning/synchronous axis, speed axis and external encoder technology objects:
  For a restart, the technology object must be disabled.
  ("MC_Power.Status" = FALSE and "MC_Power.Busy" = FALSE)

Override response

- Parameter "Restart" = FALSE:
  Processing of the "MC_Reset" instruction with parameter "Restart" = FALSE cannot be aborted by any other Motion Control job. The "MC_Reset" job does not abort any active motion control jobs.
- Parameter "Restart" = TRUE:
  Processing of the "MC_Reset" instruction with parameter "Restart" = TRUE cannot be aborted by any other Motion Control job.

Parameters

The following table shows the parameters of Motion Control instruction "MC_Reset":

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Object	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge

Parameter	Declaration	Data type	Default value	Description	
Restart	INPUT	BOOL	FALSE	TRUE	Reinitialization of the technology object and acknowledgment of pending technology alarms. The technology object is reinitialized with the configured start values.
				FALSE	Acknowledgment of queued technology alarms
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged. The restart has been executed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Acknowledging technology alarms

To acknowledge technology alarms, follow these steps:

1. Check the requirements indicated above.
2. Set the parameter "Restart" = FALSE.
3. Start the acknowledgment of the error with a positive edge at parameter "Execute".
   When the "Done" parameter shows the value "TRUE", the error has been acknowledged.

Note

Acknowledge with "Restart" = FALSE

If only the technology alarms are to be acknowledged, set "Restart" = FALSE. The technology object cannot be used during a restart.

Restarting a technology object

To restart a technology object, follow these steps:

1. Check the requirements indicated above.
2. Set the parameter "Restart" = TRUE.
3. Perform the restart with a positive edge at parameter "Execute".
   When the "Done" parameter shows the value "TRUE", the restart of the technology object is complete.

Additional information regarding the restart can be found in the section Restarting technology objects.

MC_Home V4 (S7-1500, S7-1500T)

MC_Home: Home technology object, set home position V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_Home", you create the relationship between the position in the technology object and the mechanical position. The actual position value in the technology object is assigned to a homing mark at the same time. This homing mark represents a known mechanical position.

Homing is performed according to the mode selected with the "Mode" parameter and the configuration under "Technology object > Configuration > Extended parameters > Homing".

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and ≥ V2.0 is available in section Version overview.

The preset values under "Technology object > Configuration > Extended parameters > Dynamic default values" are used for the dynamic values Acceleration, Deceleration and Jerk.

Applies to

- Synchronous axis
- Positioning axis
- External encoder

The following table shows which modes are possible with each of the technology objects:

Operating mode	Positioning axis/ synchronous axis with incremental encoder	Positioning axis/ synchronous axis with absolute encoder	External incremental encoder	External absolute encoder
Active homing ("Mode" = 3, 5)	X	-	-	-
Passive homing ("Mode" = 2, 8, 10)	X	-	X	-
Set actual position ("Mode" = 0)	X	X	X	X
Relative offset to the actual position ("Mode" = 1)	X	X	X	X
Absolute encoder adjustment ("Mode" = 6, 7)	-	X	-	X

Requirement

- The technology object has been configured correctly.
- "Mode" = 2, 3, 5, 8, 10
  The technology object is enabled.
- "Mode" = 0, 1, 6, 7, 8
  The encoder values are valid (<TO>.StatusSensor[n].State = 2).
- "Mode" = 6, 7
  The axis is in position-controlled mode.

Override response

The override response for "MC_Home" jobs is described in section Override response V4: Homing and motion jobs (Page 3251).

Parameters

The following table shows the parameters of Motion Control instruction "MC_Home":

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Position	INPUT	LREAL	0.0	The specified value is used according to the selected "Mode".	

Parameter	Declaration	Data type	Default value	Description	
Mode	INPUT	INT	0	Operating mode	
				0	Direct homing (absolute) The current position of the technology object is set to the value of parameter "Position".
				1	Direct homing (relative) The current position of the technology object is shifted by the value of parameter "Position".
				2	Passive homing (without reset) Functions like "Mode" 8, with the difference that the "homed" status is **not** reset when the function is enabled.
				3	Active homing The positioning axis/synchronous axis technology object performs a homing movement according to the configuration. After the completion of the motion, the axis is positioned at the value of the "Position" parameter.
				4	Reserved
				5	Active homing ("Position" parameter has no effect) The positioning axis/synchronous axis technology object performs a homing movement according to the configuration. After completion of the motion, the axis is positioned at the home position configured under "Technology object > Configuration > Extended parameters > Homing > Active homing". (<TO>.Homing.HomePosition)
				6	Absolute encoder adjustment (relative) The current position is shifted by the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<TO>.StatusSensor[n].AbsEncoderOffset)
				7	Absolute encoder adjustment (absolute) The current position is set to the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<TO>.StatusSensor[n].AbsEncoderOffset)

Parameter	Declaration	Data type	Default value	Description	
				8	Passive homing When the homing mark is detected, the actual value is set to the value of the "Position" parameter.
				9	Abort passive homing An active job for passive homing is aborted.
				10	Passive homing ("Position" parameter has no effect) When the homing mark is detected, the actual value is set to the home position configured under "Technology object > Configuration > Extended parameters > Homing > Passive homing". (<TO>.Homing.HomePosition)
ReferenceMarkPosition	OUTPUT	LREAL	0.0	Display of the position at which the technology object was homed (valid when "Done" = TRUE)	
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Resetting the "Homed" status

The "Homed" status of a technology object is reset under the following conditions (<TO>.StatusWord.X5 (HomingDone)):

- **Technology objects with incremental actual values:**
  - Starting an "MC_Home" job with "Mode" = 3, 5, 8, 10
    (After successful completion of the homing operation, the "Homed" status is reset.)
  - Error in the encoder system, or encoder failure
  - Restart of the technology object
  - After POWER OFF → POWER ON of the CPU
  - Memory reset
  - Modification of the encoder configuration
- **Technology objects with absolute actual values:**
  - Error in sensor system/encoder failure
  - Replacement of the CPU
  - Modification of the encoder configuration
  - Restoration of the CPU factory settings
  - Transfer of a different project into the controller

Homing a technology object with "Mode" = 1 … 8, 10

To home a technology object, follow these steps:

1. Check the requirements indicated above.
2. Specify the desired homing function in the "Mode" parameter.
3. Initialize the necessary parameters with values, and start the homing operation with a positive edge at the "Execute" parameter.
   When the "Done" parameter shows the value "TRUE", the "MC_Home" job has been completed according to the selected "Mode". The "Homed" status of the technology object is indicated under "Technology object > Diagnostics > Status and error bits > Motion status > Homed" (<TO>.StatusWord.X5 (HomingDone)).

Abort of a passive homing process with "Mode" = 9

With "Mode" = 9, the technology object is not homed. When an active "MC_Home" job for passive homing ("Mode" = 2, 8, 10) is overridden by another "MC_Home" job with "Mode" = 9, the running job is aborted with parameter "CommandAborted" = TRUE. The overriding job with "Mode" = 9 signals successful execution with parameter "Done" = TRUE.

Additional information

An option for the evaluation of the individual status bits can be found in the section Evaluating StatusWord, ErrorWord and WarningWord.

See also

Override response V4: Homing and motion jobs (Page 3251)

MC_Halt V4 (S7-1500, S7-1500T)

MC_Halt: Pause axis V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_Halt", you brake an axis until it comes to a standstill.

You define the dynamic behavior of the braking operation with parameters "Jerk" and "Deceleration".

Applies to

- Synchronous axis
- Positioning axis
- Speed axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The override response for "MC_Halt" jobs is described in section Override response V4: Homing and motion jobs (Page 3251).

Parameters

The following table shows the parameters of Motion Control instruction "MC_Halt":

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)

Parameters	Declaration	Data type	Default value	Description	
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	Constant-acceleration velocity profile; the specified jerk is used
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
AbortAcceleration	INPUT	BOOL	FALSE	FALSE	The current acceleration at the start of the job is reduced using the configured jerk. Afterwards, the deceleration builds up
				TRUE	The acceleration is set to 0.0 at the start of the job, and the deceleration immediately builds up.
Done	OUTPUT	BOOL	FALSE	TRUE	Velocity zero has been reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Braking an axis with "MC_Halt"

Proceed as follows to decelerate an axis to a standstill:

1. Check the requirements indicated above.
2. Supply the parameters "Deceleration", "Jerk" and "AbortAcceleration" with the desired values.
3. Start the "MC_Halt" job with a positive edge at parameter "Execute".
   The current motion state is indicated in "Busy", "Done" and "Error". The standstill of the axis is indicated under "Technology object > Diagnostics > Status and error bits > Motion status > Standstill" (<TO>.StatusWord.X7 (Standstill)).

Braking an axis with active force/torque limit

With active force/torque limiting the permitted torque may not be sufficient to move the calculated braking ramp. The "MC_Halt" job does not signal "Done" = TRUE.

To brake an axis with active force/torque limiting, use the Motion Control instruction "MC_MoveVelocity" with "MC_MoveVelocity.Velocity" = 0.0 and "MC_MoveVelocity.PositionControlled" = FALSE.

Additional information

An option for the evaluation of the individual status bits can be found in the section Evaluating StatusWord, ErrorWord and WarningWord.

See also

Override response V4: Homing and motion jobs (Page 3251)

MC_MoveVelocity: Move axis with velocity/speed setpoint V4 (Page 3128)

MC_Halt: Function chart V4 (S7-1500, S7-1500T)

Function chart: Stopping an axis and the overriding job characteristics

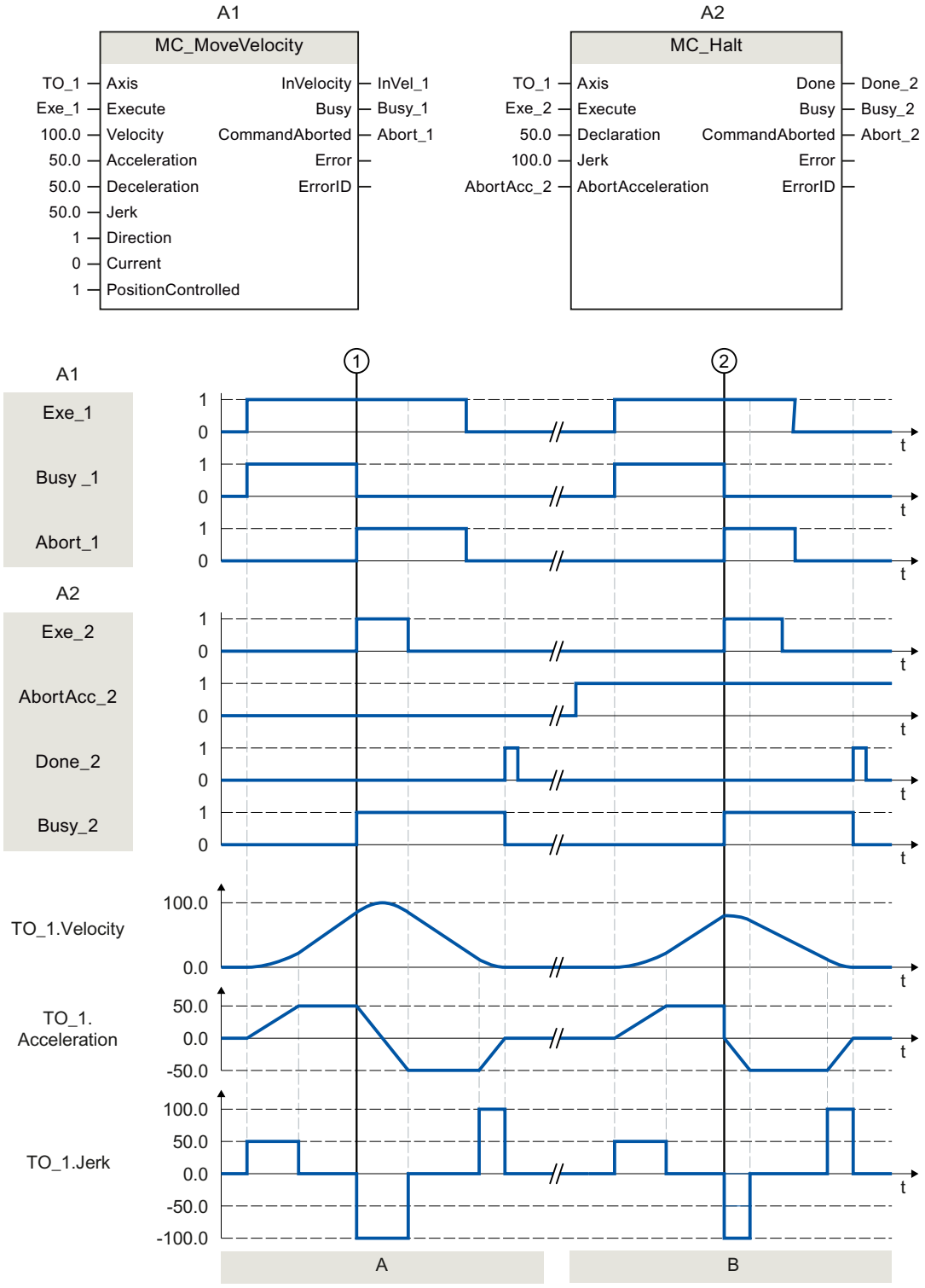

Section A	An axis is moved with an "MC_MoveVelocity" job (A1). At time ①, the "MC_MoveVelocity" job is overridden by an "MC_Halt" job (A2). The job abort is signaled via "Abort_1". With "AbortAcc_2" = FALSE, the current acceleration is reduced with the specified jerk. Afterward, the deceleration builds up and the axis is braked to a standstill. The completion of the "MC_Halt" job is reported via "Done_2".
Section B	The axis is moved with an "MC_MoveVelocity" job (A1). At time ②, the "MC_MoveVelocity" job is overridden by an "MC_Halt" job (A2). The job abort is signaled via "Abort_1". With "AbortAcc_2" = TRUE, the current acceleration is set to zero immediately and the deceleration builds up. The axis is braked to a standstill. The completion of the "MC_Halt" job is reported via "Done_2".

MC_MoveAbsolute V4 (S7-1500, S7-1500T)

MC_MoveAbsolute: Position axis absolutely V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MoveAbsolute", you can move an axis to an absolute position.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

Applies to

- Synchronous axis
- Positioning axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.
- The technology object is homed.

Override response

The override response for "MC_MoveAbsolute" jobs is described in section Override response V4: Homing and motion jobs (Page 3251).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveAbsolute":

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_PositioningAxis	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge

Parameters	Declaration	Data type	Default value	Description	
Position	INPUT	LREAL	0.0	Absolute target position	
Velocity	INPUT	LREAL	-1.0	Velocity setpoint for the positioning	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	Constant acceleration velocity profile The specified value is used.
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
Direction	INPUT	INT	1	Motion direction of the axis This parameter is only evaluated when the modulo function is enabled. "Technology object > Configuration > Basic parameters > Enable modulo	
				1	Positive direction
				2	Negative direction
				3	Shortest distance
Done	OUTPUT	BOOL	FALSE	TRUE	The target position has been reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.

Parameters	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Moving an axis to an absolute position

Proceed as follows to move an axis to an absolute position:

1. Check the requirements indicated above.
2. Specify the desired target position in the "Position" parameter.
3. Start the "MC_MoveAbsolute" job with a positive edge at parameter "Execute".
   The current motion state is indicated in "Busy", "Done" and "Error".

See also

Override response V4: Homing and motion jobs (Page 3251)

MC_MoveAbsolute: Function chart V4 (S7-1500, S7-1500T)

Function chart: Absolute positioning of an axis, and the response to an overriding job

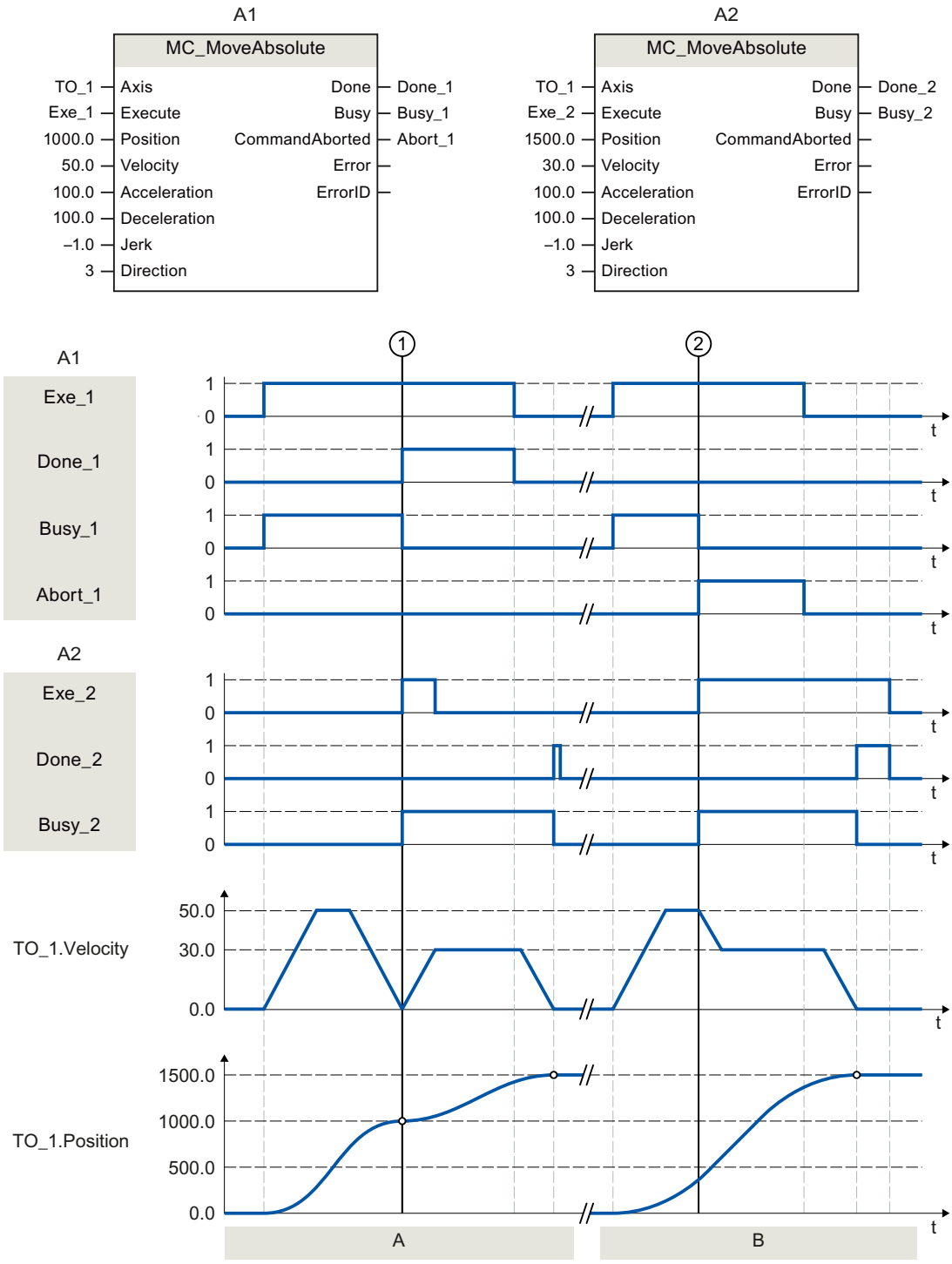

Section A	An axis is moved to absolute position 1000.0 with an "MC_MoveAbsolute" job (A1). When the axis reaches the target position, this is signaled at time ① via "Done_1". At this time ①, another "MC_MoveAbsolute" job (A2) with target position 1500.0 is started. When the axis reaches the target position 1500.0, this is signaled via "Done_2". Since "Exe_2" was previously reset, "Done_2" is applied only to one cycle.
Section B	An active "MC_MoveAbsolute" job (A1) is overridden at time ② by another "MC_MoveAbsolute" job (A2). The abort is signaled via "Abort_1". The axis is braked to the changed velocity and moved to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveRelative V4 (S7-1500, S7-1500T)

MC_MoveRelative: Position axis relatively V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MoveRelative", you move an axis relative to its position when execution of the job began.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

Applies to

- Synchronous axis
- Positioning axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The override response for "MC_MoveRelative" jobs is described in section Override response V4: Homing and motion jobs (Page 3251).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveRelative":

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning-Axis	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Distance	INPUT	LREAL	0.0	Distance for the positioning process (negative or positive)	

Parameter	Declaration	Data type	Default value	Description	
Velocity	INPUT	LREAL	-1.0	Velocity setpoint for the positioning	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	Constant acceleration velocity profile The specified value is used.
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
Done	OUTPUT	BOOL	FALSE	TRUE	The target position has been reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Moving an axis relative to the starting position

Proceed as follows to move an axis relative to the starting position:

1. Check the requirements indicated above.
2. Specify distance to be moved in the "Distance" parameter.
3. Start the "MC_MoveRelative" job with a positive edge at parameter "Execute".
   The current motion state is indicated in "Busy", "Done" and "Error".

See also

Override response V4: Homing and motion jobs (Page 3251)

MC_MoveRelative: Function chart V4 (S7-1500, S7-1500T)

Function chart: Relative positioning of an axis, and the overriding job characteristics

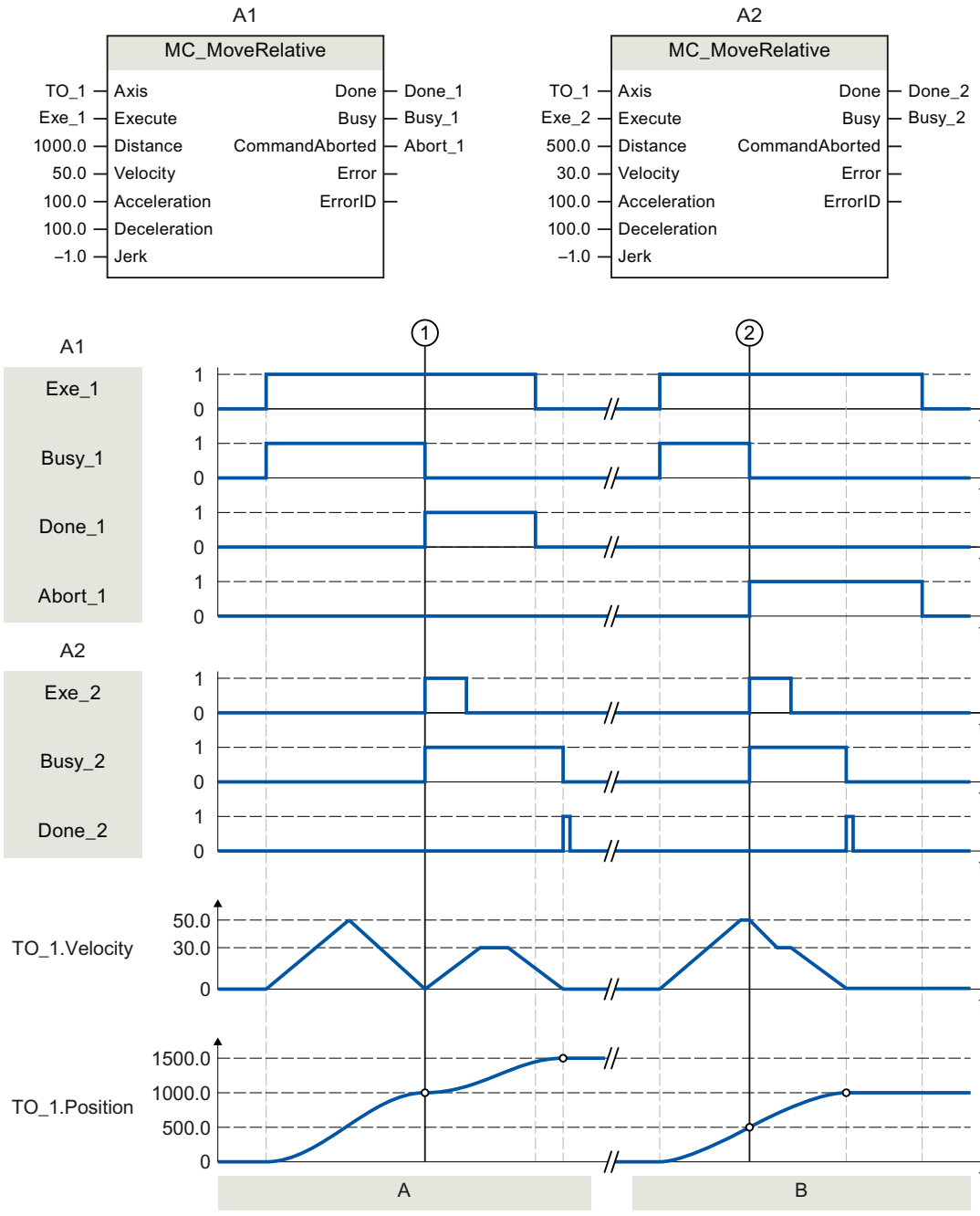

Section A	The axis is moved by an "MC_MoveRelative" job (A1) by the distance ("Distance") 1000.0 (the starting position here is 0.0). When the axis reaches the target position, this is signaled at time ① via "Done_1". At this time ①, another "MC_MoveRelative" job (A2) with distance 500.0 is started. When the new target position is reached, this is signaled via "Done_2". Since "Exe_2" was previously reset, "Done_2" is applied only to one cycle.
Section B	An active "MC_MoveRelative" job (A1) is overridden by another "MC_MoveRelative" job (A2). The abort is signaled at time ② via "Abort_1". The axis is then moved at the new velocity by the distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveVelocity V4 (S7-1500, S7-1500T)

MC_MoveVelocity: Move axis with velocity/speed setpoint V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MoveVelocity", you move an axis at constant velocity/speed.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

- Positioning axis/synchronous axis:
  A velocity is specified in the "Velocity" parameter.
- Speed axis:
  A speed is specified in the "Velocity" parameter.

Applies to

- Synchronous axis
- Positioning axis
- Speed axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The override response for "MC_MoveVelocity" jobs is described in section Override response V4: Homing and motion jobs (Page 3251).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveVelocity":

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Velocity	INPUT	LREAL	100.0	Velocity setpoint / speed setpoint for the motion ("Velocity" = 0.0 is permitted)	
Acceleration	INPUT	LREAL	-1.0	Acceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	Constant acceleration velocity profile The specified value is used.
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
Direction	INPUT	INT	0	Direction of rotation of the axis	
				0	The sign of the velocity specified at the "Velocity" parameter defines the direction of rotation.
				1	Positive direction of rotation The value of "Velocity" is used.
				2	Negative direction of rotation The value of "Velocity" is used.

Parameters	Declaration	Data type	Default value	Description	
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	Deactivated The values of parameters "Velocity" and "Direction" are taken into account.
				TRUE	activated The values at the parameters "Velocity" and "Direction" are not taken into account. The current velocity and direction at function start are retained. When the axis resumes motion at the velocity that was current at function start, the "InVelocity" parameter returns the value "TRUE".
PositionControlled	INPUT	BOOL	TRUE	FALSE	Non position-controlled operation
				TRUE	Position-controlled mode
				The parameter applies as long as the "MC_MoveVelocity" job is being executed. After this, the setting of the following job applies. This parameter is ignored when a speed axis is used.	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity setpoint/speed setpoint has been reached. A velocity setpoint/speed setpoint is output.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Behavior with setpoint velocity/speed zero ("Velocity" = 0.0)

An "MC_MoveVelocity" job with "Velocity" = 0.0 stops the axis with the configured deceleration. When the velocity setpoint/speed setpoint zero is reached, the parameter "InVelocity" will indicate the value "TRUE".

Under "Technology object > Diagnostics > Status and error bits > Motion status", "constant velocity" and "standstill" will be displayed (<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill)).

The parameters "InVelocity" and "Busy" show the value "TRUE", until the "MC_MoveVelocity" job is overridden by another Motion Control job.

Moving an axis with constant velocity/speed

Proceed as follows to move an axis with constant velocity/speed:

1. Check the requirements indicated above.
2. At the "Velocity" parameter, specify the velocity/speed, with which the axis should be moved.
3. Start the "MC_MoveVelocity" job with a positive edge at parameter "Execute".
   The current motion state is indicated in "Busy", "InVelocity" and "Error".
   If the "InVelocity" parameter shows the value "TRUE", the velocity/speed setpoint has been reached. The axis continues moving at this constant velocity. The parameters "InVelocity" and "Busy" show the value "TRUE", until the "MC_MoveVelocity" job is overridden by another Motion Control job.

Note

Response to a change in the override

If the velocity/speed is influenced during constant motion by a change in the override (<TO>.Override.Velocity), the "InVelocity" parameter is reset during the acceleration or deceleration. When the newly calculated velocity/speed is reached ("Velocity" × "Override" %), then "InVelocity" is set again.

Additional information

An option for the evaluation of the individual status bits can be found in the section Evaluating StatusWord, ErrorWord and WarningWord.

See also

Override response V4: Homing and motion jobs (Page 3251)

MC_MoveVelocity: Function chart V4 (S7-1500, S7-1500T)

Function chart: Moving an axis with velocity specification, and the response to an overriding job

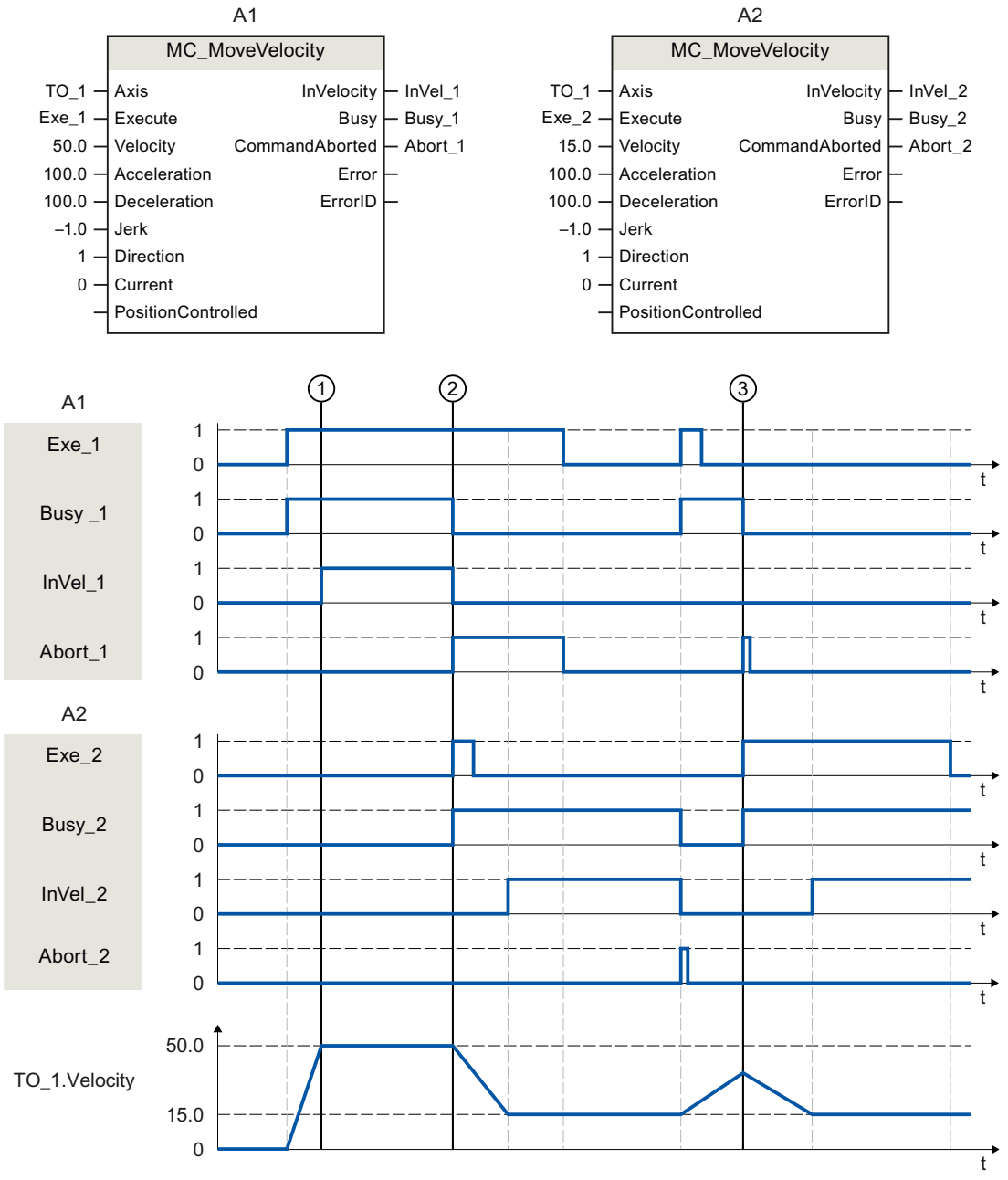

An "MC_MoveVelocity" job (A1) initiated via "Exe_1" accelerates the axis and signals at time ① via "InVel_1" that the velocity setpoint 50.0 has been reached.

At time ②, the job is overridden by another "MC_MoveVelocity" job (A2). The abort is signaled via "Abort_1". When the new velocity setpoint 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the constant velocity 15.0.

The running "MC_MoveVelocity" job (A2) is overridden by another "MC_MoveVelocity" job (A1). The abort is signaled via "Abort_2". The axis is accelerated to the new velocity setpoint 50.0. Before the velocity setpoint is reached, the current "MC_MoveVelocity" job (A1) is overridden at time ③ by another "MC_MoveVelocity" job (A2). The abort is signaled via "Abort_1". When the new velocity setpoint 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the constant velocity 15.0.

MC_MoveJog V4 (S7-1500, S7-1500T)

MC_MoveJog: Move axis in jog mode V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MoveJog", you move an axis in jog mode.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

- Positioning axis/synchronous axis:
  A velocity is specified in the "Velocity" parameter.
- Speed axis:
  A speed is specified in the "Velocity" parameter.

Applies to

- Synchronous axis
- Positioning axis
- Speed axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The override response for "MC_MoveJog" jobs is described in section Override response V4: Homing and motion jobs (Page 3251).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveJog":

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Technology object	
JogForward	INPUT	BOOL	FALSE	TRUE	As long as the parameter is "TRUE", the axis moves in the positive direction at the velocity specified in parameter "Velocity".
JogBackward	INPUT	BOOL	FALSE	TRUE	As long as the parameter is "TRUE", the axis moves in the negative direction at the velocity specified in parameter "Velocity".
Velocity	INPUT	LREAL	100.0	Velocity setpoint / speed setpoint for the motion	
				≥ 0.0	The specified value is used.
				< 0.0	The absolute value of the specified value is used.
Acceleration	INPUT	LREAL	-1.0	Acceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	Constant acceleration velocity profile The specified value is used.
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
PositionControlled	INPUT	BOOL	TRUE	FALSE	Non position-controlled operation
				TRUE	Position-controlled mode
				The parameter applies as long as the "MC_MoveJog" job is being executed. After this, the setting of the following job applies. This parameter is ignored when a speed axis is used.	

Parameter	Declaration	Data type	Default value	Description	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity setpoint/speed setpoint has been reached. A velocity setpoint/speed setpoint is output.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Behavior with setpoint velocity/speed zero ("Velocity" = 0.0)

An "MC_MoveJog" job with "Velocity" = 0.0 stops the axis with the configured deceleration. When the velocity setpoint/speed setpoint zero is reached, the parameter "InVelocity" will indicate the value "TRUE".

Under "Technology object > Diagnostics > Status and error bits > Motion status", "constant velocity" and "standstill" will be displayed (<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill)).

Moving an axis in jog mode

Proceed as follows to move an axis in jog mode:

1. Check the requirements indicated above.
2. Move the axis in the positive direction with "JogForward", or in the negative direction with "JogBackward".
   The current motion state is indicated in "Busy", "InVelocity" and "Error".
   If both "JogForward" and "JogBackward" are set to TRUE, the axis is braked at the last valid deceleration. The error 16#8007 (incorrect direction specification) is output.

Note

Response to a change in the override

If the velocity/speed is influenced during constant motion by a change in the override (<TO>.Override.Velocity), the "InVelocity" parameter is reset during the acceleration or deceleration. When the newly calculated velocity is reached ("Velocity" × "Override" %), then "InVelocity" is set again.

Additional information

An option for the evaluation of the individual status bits can be found in the section Evaluating StatusWord, ErrorWord and WarningWord.

See also

Override response V4: Homing and motion jobs (Page 3251)

MC_MoveJog: Function chart V4 (S7-1500, S7-1500T)

Function chart: Moving an axis in jog mode

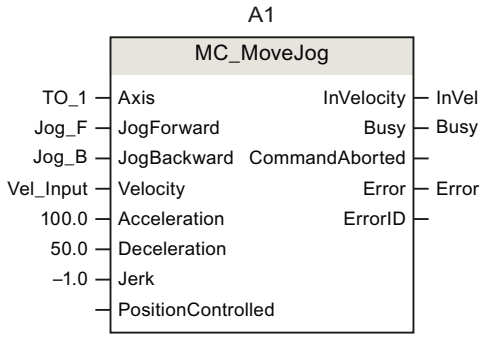

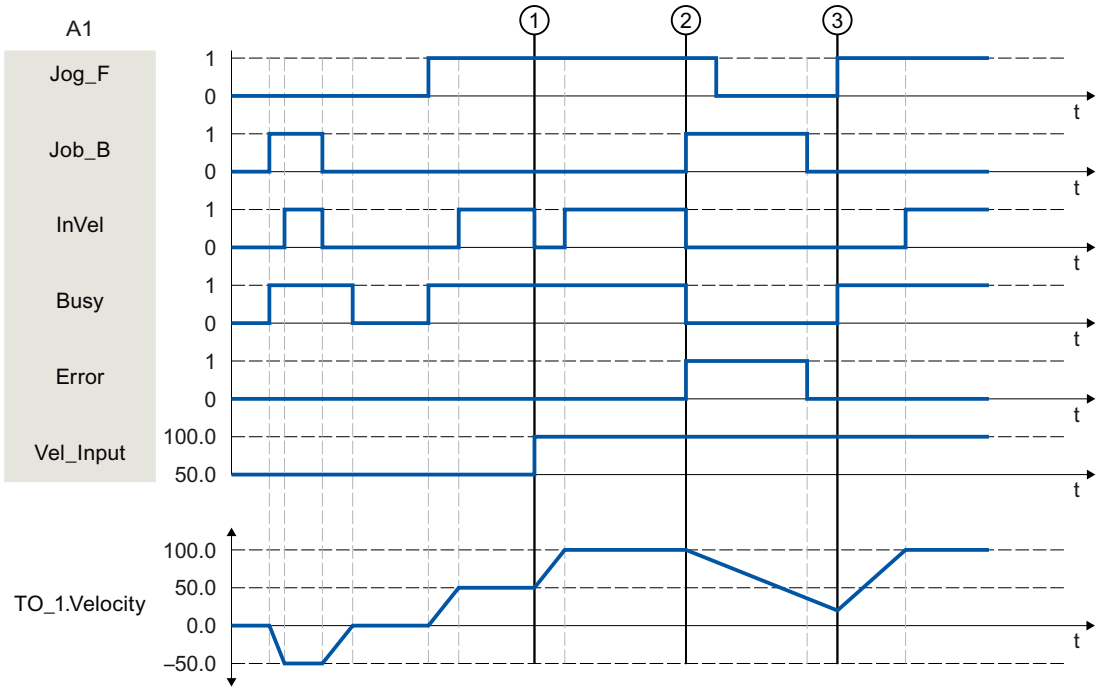

The axis is moved in the negative direction in jog mode via "Jog_B". When the velocity setpoint -50.0 is reached, this is signaled via "InVel" = TRUE. After "Jog_B" is reset, the axis is braked and brought to a standstill. Then the axis is moved in the positive direction via "Jog_F". When the velocity setpoint 50.0 is reached, this is signaled via "InVel" = TRUE.

At the time ①, if "Jog_F" is set, the velocity setpoint is changed to 100.0 by means of "Vel_Input". Alternatively, you can also change the velocity setpoint using the velocity override. "InVel" is reset. Axis is being accelerated. When the new velocity setpoint 100.0 is reached, this is signaled via "InVel" = TRUE.

If "Jog_F" is set, "Jog_B" is likewise set at time ②. If both "Jog_F" and "Jog_B" are set, then the axis is braked with the last applicable deceleration. An error is indicated via "Error", and the "ErrorID" of the error 16#8007 (incorrect direction specification) is output.

This error is resolved by resetting the two inputs "Jog_F" and "Jog_B".

During the braking ramp, "Jog_F" is set at time ③. The axis is accelerated to the last configured velocity. When the velocity setpoint 100.0 is reached, this is signaled via "InVel" = TRUE.

MC_MoveSuperimposed V4 (S7-1500, S7-1500T)

MC_MoveSuperimposed: Position axes overlapping V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MoveSuperimposed", you start a relative positioning motion that is superimposed on a active basic motion.

You define the dynamic behavior of the motion with parameters "VelocityDiff", "Jerk", "Acceleration" and "Deceleration" . The dynamic values are added to the values of the basic motion. The duration of the basic motion is not extended by a superimposed motion.

The dynamics of the total axis motion is the sum of the dynamic values of the basic motion and the superimposed motion.

The behavior of the total motion depends on the type of basic motion:

- The basic motion is a single-axis motion:
  - The maximum dynamics of the superimposed motion is the difference between the current dynamic values of the basic motion and the dynamic limits.
  - The entire motion is limited to the configured dynamic limits.
- The basic motion is a synchronous motion:
  - The maximum dynamics of the superimposed motion is the difference between the current dynamic values of the basic motion and the dynamic limits.
  - The synchronous motion of the following axis is not limited to the dynamic limits of the following axis.
  - An "MC_MoveSuperimposed" job on a leading axis in synchronous operation affects the leading axis and the following axis.
  - An "MC_MoveSuperimposed" job on a following axis in synchronous operation only affects the following axis.

It is always the dynamics of the total motion that is displayed in the technology data block and in the TIA Portal.

Applies to

- Synchronous axis
- Positioning axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The override response for "MC_MoveSuperimposed" jobs is described in section Override response V4: Homing and motion jobs (Page 3251).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveSuperimposed":

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning-Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Distance	INPUT	LREAL	0.0	Additional distance for the overlapping positioning process (negative or positive)	
VelocityDiff	INPUT	LREAL	-1.0	Maximum velocity deviation compared to the active motion	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)

Parameters	Declaration	Data type	Default value	Description	
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	Constant acceleration velocity profile The specified value is used.
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
Done	OUTPUT	BOOL	FALSE	TRUE	Superimposed positioning complete
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Starting superimposed positioning motion

To start a superimposed positioning motion with Motion Control instruction "MC_MoveSuperimposed", follow these steps:

1. Check the requirements indicated above.
2. Specify the additional distance to be moved in the "Distance" parameter.
3. Start the "MC_MoveSuperimposed" job with a positive edge at parameter "Execute". The current motion state is indicated in "Busy", "Done" and "Error".

See also

Override response V4: Homing and motion jobs (Page 3251)

MC_MoveSuperimposed: Function chart V4 (S7-1500, S7-1500T)

Function chart: Positioning axes overlapping

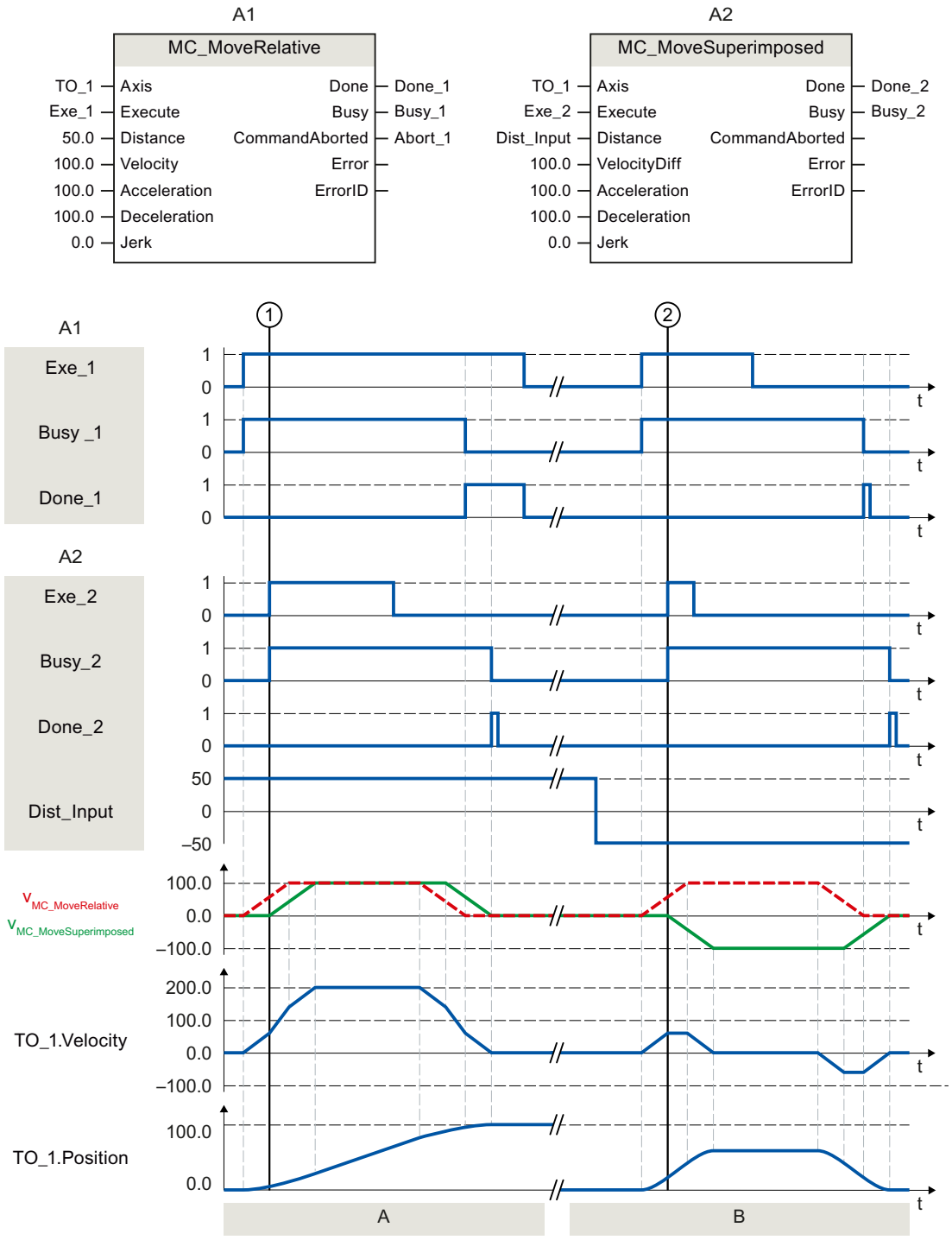

Section A	Using "Exe_1", an "MC_MoveRelative" job with a distance of 50.0 is initiated. At time ①, using "Exe_2", an "MC_MoveSuperimposed" job with a distance of 50.0 is initiated. The axis is moved with the added dynamic values of both jobs by the distance 50 + 50 = 100.0. When the axis reaches the target position, this is signaled via "Done_2".
Section B	Using "Exe_1", an "MC_MoveRelative" job with a distance of 50.0 is initiated. At time ②, using "Exe_2", a MC_MoveSuperimposed job with a distance of -50.0 is initiated. The axis reverses and is moved with the added dynamic values of both jobs by the distance 50.0 - 50.0 = 0.0. When the axis reaches the target position, this is signaled via "Done_2".

MC_SetSensor V4 (S7-1500T)

MC_SetSensor: Set alternative encoder as operationally active encoder V4 (S7-1500T)

Description

With the Motion Control instruction "MC_SetSensor", you switch over the encoder for closed loop position control of the axis.

The actual value of the addressed encoder can be adapted without switchover using parameter "Mode" 2 and 3.

Applies to

- Synchronous axis
- Positioning axis

Requirement

- The technology object and the alternative encoder have been configured correctly.
- No restart command and no "MC-Home" job running.

Override response

- An "MC_SetSensor" job is not aborted by any other Motion Control job.
- A new "MC_SetSensor" job does not abort any active motion control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_SetSensor":

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning-Axis	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge

Parameters	Declaration	Data type	Default value	Description	
Sensor	INPUT	INT	1	Number of the new encoder (1 to 4).	
Mode	INPUT	DINT	0	The mode determines the position alignment between the old encoder and new encoder.	
				0	Switch over encoder and transfer actual position to the new encoder With this encoder switchover, step changes in the positioning control are prevented. Bumpless switchover of the encoders is possible.
				1	Switch over encoder without aligning the actual position **Note** When closed loop position control is active, an additional difference of the two encoders acts as additional control deviation and can trigger a compensating motion.
				2	Transfer actual value The actual position is transferred to the encoder specified in the "Sensor" parameter.
				3	Transfer actual value The actual position of the "Reference encoder" ("ReferenceSensor" parameter) is transferred to the encoder specified in the "Sensor" parameter.
ReferenceSensor	INPUT	INT	1	Number of the reference encoder (see parameter "Mode" = 3)	
Done	OUTPUT	BOOL	FALSE	TRUE	Encoder for closed loop position control of the axis was switched over.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job has been aborted.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Changing to absolute encoder

When you switch the encoder to an absolute encoder and transfer the actual value ("Mode" = 2, 3), the actual value is calculated with the value of the absolute encoder and the absolute value offset. When switching to a different encoder, calculation of the actual value is canceled. The absolute encoder once again returns the absolute value + absolute value offset (<TO>.StatusSensor[n].AbsEncoderOffset) without calculation by the "MC_SetSensor" job.

Measuring input, output cam, cam track (S7-1500, S7-1500T)

MC_MeasuringInput V4 (S7-1500, S7-1500T)

MC_MeasuringInput: Start measuring once V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MeasuringInput", a one-time measuring is started.

With the one-time measuring, one or two edges can be detected with one measuring job. The position of the respective technology object axis or external encoder is assigned to the measuring event. The measurement result is indicated in the function block and in the technology data block and can be further processed in the user program. This completes the measuring job.

Another measuring job must be started again using "MC_MeasuringInput.Execute" = TRUE.

Applies to

- Measuring input

Requirement

- The technology object has been configured correctly.
- The encoder of the axis must have "valid" status (StatusSensor[1..4].State = valid). Otherwise, the measuring job is rejected in the function block with an error indication.
- Measuring using PROFIdrive is not possible during active or passive homing.

Override response

The override response for "MC_MeasuringInput" jobs is described in section Override response V4: Measuring input jobs (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MeasuringInput":

Parameters	Declaration	Data type	Default value	Description	
MeasuringInput	INPUT	TO_MeasuringInput	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge

Parameters	Declaration	Data type	Default value	Description	
Mode	INPUT	DINT	0	Type of measurement	
				0	Measurement of next positive edge
				1	Measurement of next negative edge
				2	Measurement of next two edges
				3	Measurement of two edges, beginning with the positive edge • Rising edge = "MeasuredValue1" (measured value 1) • Falling edge = "MeasuredValue2" (measured value 2)
				4	Measurement of two edges, beginning with the negative edge • Falling edge = "MeasuredValue1" • Rising edge = "MeasuredValue2"
MeasuringRange	INPUT	BOOL	FALSE	Acquisition of measured values Observe the time-related boundary conditions.	
				FALSE	Always acquire measured values
				TRUE	Acquire measured values only within the measuring range
StartPosition	INPUT	LREAL	0.0	Start position of the measuring range For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
EndPosition	INPUT	LREAL	0.0	End position of the measuring range For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
Done	OUTPUT	BOOL	FALSE	TRUE	The block has been completely processed. Measured values are valid.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The measuring job has been aborted.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	
MeasuredValue1	OUTPUT	LREAL	0.0	First measured value	
MeasuredValue2	OUTPUT	LREAL	0.0	Second measured value (for measurement of two edges)	

See also

Override response V4: Measuring input jobs (Page 3253)

MC_MeasuringInput: Function chart V4 (S7-1500, S7-1500T)

Function chart: Start measuring job once

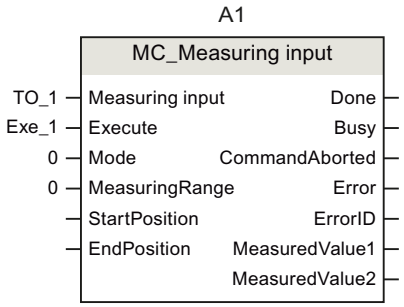

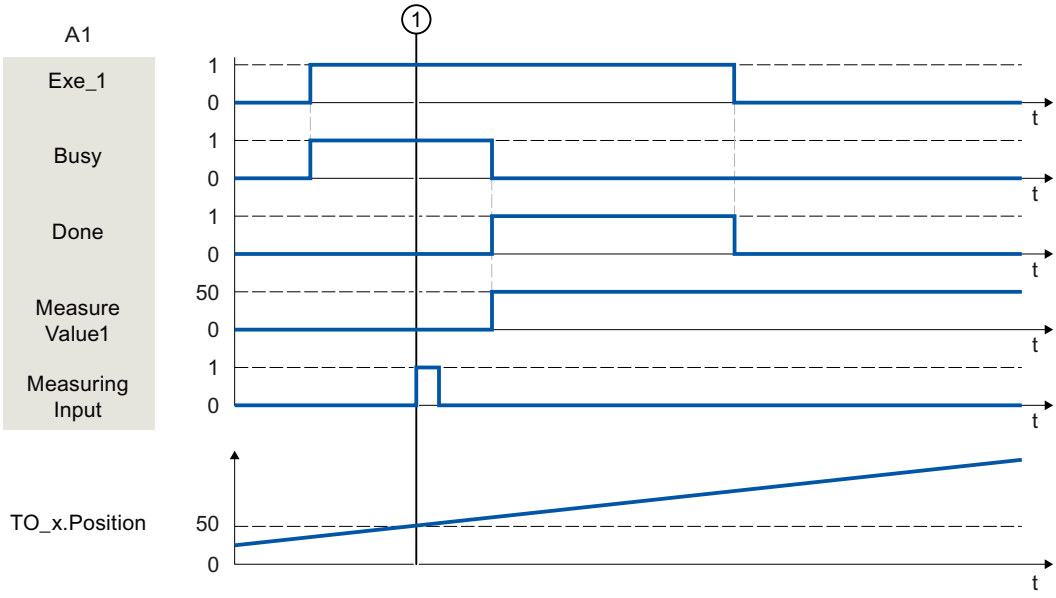

A "MC_MeasuringInput" job with "Mode" = 0 initiated using "Execute" performs a measurement of the next positive edge. A successfully completed measurement is signaled with "Done" = TRUE. The determined measured value ① (in example: 50) is output using "MeasuredValue1".

MC_MeasuringInputCyclic V4 (S7-1500, S7-1500T)

MC_MeasuringInputCyclic: Start cyclic measuring V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MeasuringInputCyclic", a cyclic measuring is started.

With cyclic measuring, up to two measuring events are detected by the system and the associated measuring positions displayed. The measurements are continued cyclically until they are ended per command.

Applies to

- Measuring input

Requirement

- The technology object has been configured correctly.
- Cyclic measuring is only possible when measuring using TM Timer DIDQ.
- The encoder of the axis operationally in effect must have "valid" status "valid" ("StatusSensor[1..4].State" = valid). Otherwise, the measuring job is rejected in the function block with an error indication.

Override response

The override response for "MC_MeasuringInputCyclic" jobs is described in section Override response V4: Measuring input jobs (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MeasuringInputCyclic":

Parameters	Declaration	Data type	Default value	Description	
MeasuringInput	INPUT	TO_MeasuringInput	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Mode	INPUT	DINT	0	Type of measurement	
				0	Measurement of positive edges
				1	Measurement of negative edges
				2	Measurement of both edges
MeasuringRange	INPUT	BOOL	FALSE	Acquisition of measured values Observe the time-related boundary conditions.	
				FALSE	Always acquire measured values
				TRUE	Acquire measured values only within the measuring range
StartPosition	INPUT	LREAL	0.0	Start position of the measuring range For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
EndPosition	INPUT	LREAL	0.0	End position of the measuring range For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The measuring job has been aborted.

Parameters	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	
MeasuredValue1	OUTPUT	LREAL		First measured value	
MeasuredValue2	OUTPUT	LREAL		Second measured value (when measuring multiple edges in one position control cycle)	
MeasuredValue1Counter	OUTPUT	UDINT	0	Count value for the first measured value	
MeasuredValue2Counter	OUTPUT	UDINT	0	Count value for the second measured value	
LostEdgeCounter1	OUTPUT	UDINT	0	Count value for the lost edges in the cycle clock of the first measured value acquisition	
LostEdgeCounter2	OUTPUT	UDINT	0	Count value for the lost edges in the cycle clock of the second measured value acquisition	

See also

Override response V4: Measuring input jobs (Page 3253)

MC_MeasuringInputCyclic: Function chart V4 (S7-1500, S7-1500T)

Function chart: Start cyclic measuring job

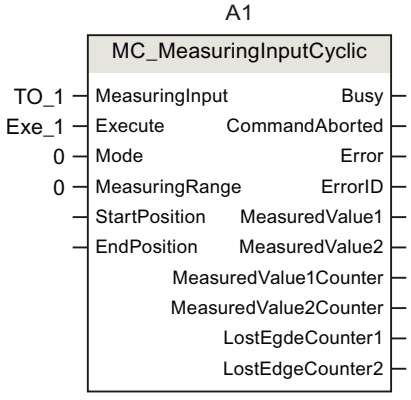

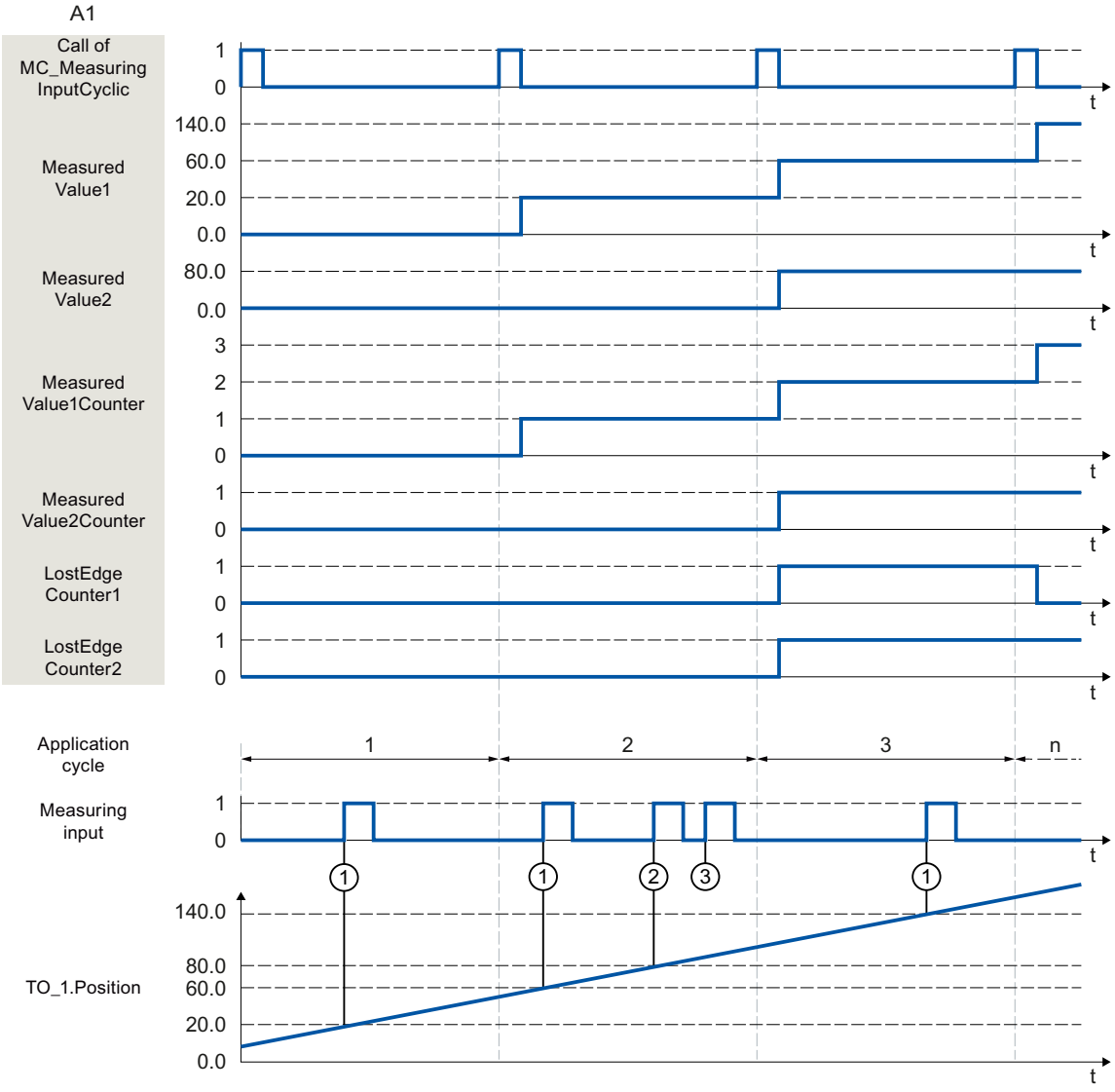

A "MC_MeasuringInputCyclic" command initiated using "Execute" (signal and start point not shown) (e.g. call in MC-PreServo [OB67]) with "Mode" = 0 and without specified measuring range "MeasuringRange = FALSE" performs a measurement of the positive edges. By calling the Motion Control instruction "MC_MeasuringInputCyclic" in MC-PreServo [OB67] or MC-PreServo [OB67], the measuring command as well as the measurement itself is called in the same application cycle.

The actual position determined at the time of the first positive edge ① in the position control cycle is output using "MeasuredValue1", and count tag "MeasuredValue1Counter" is incremented by "1".

The actual position determined at the time of the second positive edge ② in the position control cycle is output using "MeasuredValue2", and count tag "MeasuredValue2Counter" is incremented by "1".

If additional positive edges occur in the same position control cycle ③, these are recorded in LostEdgeCounter1 and LostEdgeCounter2.

MC_AbortMeasuringInput V4 (S7-1500, S7-1500T)

MC_AbortMeasuringInput: Abort active measuring V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_AbortMeasuringInput", an active one-time or cyclic measuring job is aborted.

Applies to

- Measuring input

Requirement

- The technology object has been configured correctly.

Override response

The override response for "MC_AbortMeasuringInput" jobs is described in section Override response V4: Measuring input jobs (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_AbortMeasuringInput":

Parameters	Declaration	Data type	Default value	Description	
MeasuringInput	INPUT	TO_MeasuringInput	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Done	OUTPUT	BOOL	0	The function block has been processed. The measuring job has been deactivated.	
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job has been aborted.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

Override response V4: Measuring input jobs (Page 3253)

MC_OutputCam V4 (S7-1500, S7-1500T)

MC_OutputCam: Activate/deactivate output cam V4 (S7-1500, S7-1500T)

Description

You use Motion Control instruction "MC_OutputCam" to activate the specified output cam.

Depending on the output cam type, the following input parameters are in effect:

- Distance output cam
  - "OnPosition"
  - "OffPosition"
- Time-based output cam
  - "OnPosition"
  - "Duration"

You define the operating mode and the effective direction of the output cam with parameters "Mode" and "Direction".

When "MC_OutputCam.Enable" = TRUE, the input parameters are always read and take effect at the next position control cycle.

When a technological alarm occurs, the output cam is processed again after error acknowledgment.

Applies to

- Output cam

Requirement

- The technology object has been configured correctly.
- The higher-level technology object must have a valid position.
- Setpoint output cams are not output for an axis in non-position-controlled operation.
- A setpoint output cam with the setting "MC_OutputCam.OnPosition" = 0 switches at position setpoint = 0.

Override response

The "MC_OutputCam" is canceled by:

- Disabling of the output cam with "MC_OutputCam.Enable" = FALSE
- Only one instance of the "MC_OutputCam" function block can be active on an output cam in the user program. A second instance of the "MC_OutputCam" function block on a currently active output cam is rejected with an error.

Parameters

The following table shows the parameters of Motion Control instruction "MC_OutputCam":

Parameters	Declaration	Data type	Default value	Description	
OutputCam	INPUT	TO_OutputCam	-	Technology object	
Enable	INPUT	BOOL	FALSE	FALSE	Output cam is disabled
				TRUE	Output cam is being processed
OnPosition	INPUT	LREAL	0.0	Start position of the output cam [unit of measurement of the interconnected technology object] For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
OffPosition	INPUT	LREAL	0.0	End position of the distance output cam [unit of measurement of the interconnected technology object] For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
Duration	INPUT	LREAL	0.0	Switch-on duration of the time-based output cam [unit of measurement: ms] The value for the switch-on duration of a time-based output cam must be greater than 0.0.	
Mode	INPUT	DINT	1	Operating mode	
				1	Standard Output cam functionality (output not inverted)
				2	Output cam functionality with inverted output
				3	Output cam always ON (while "Enable" = TRUE)
Direction	INPUT	DINT	1	Activation direction of the output cam	
				1	Positive direction
				2	Negative direction
				3	Both directions
CamOutput	OUTPUT	BOOL	-	Status display based on last call of Motion Control instruction "MC_OutputCam"	
Busy	OUTPUT	BOOL	FALSE	TRUE	The processing of the output cam is active.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

MC_OutputCam: Function chart V4 (S7-1500, S7-1500T)

Function chart: Activate the specified output cam type

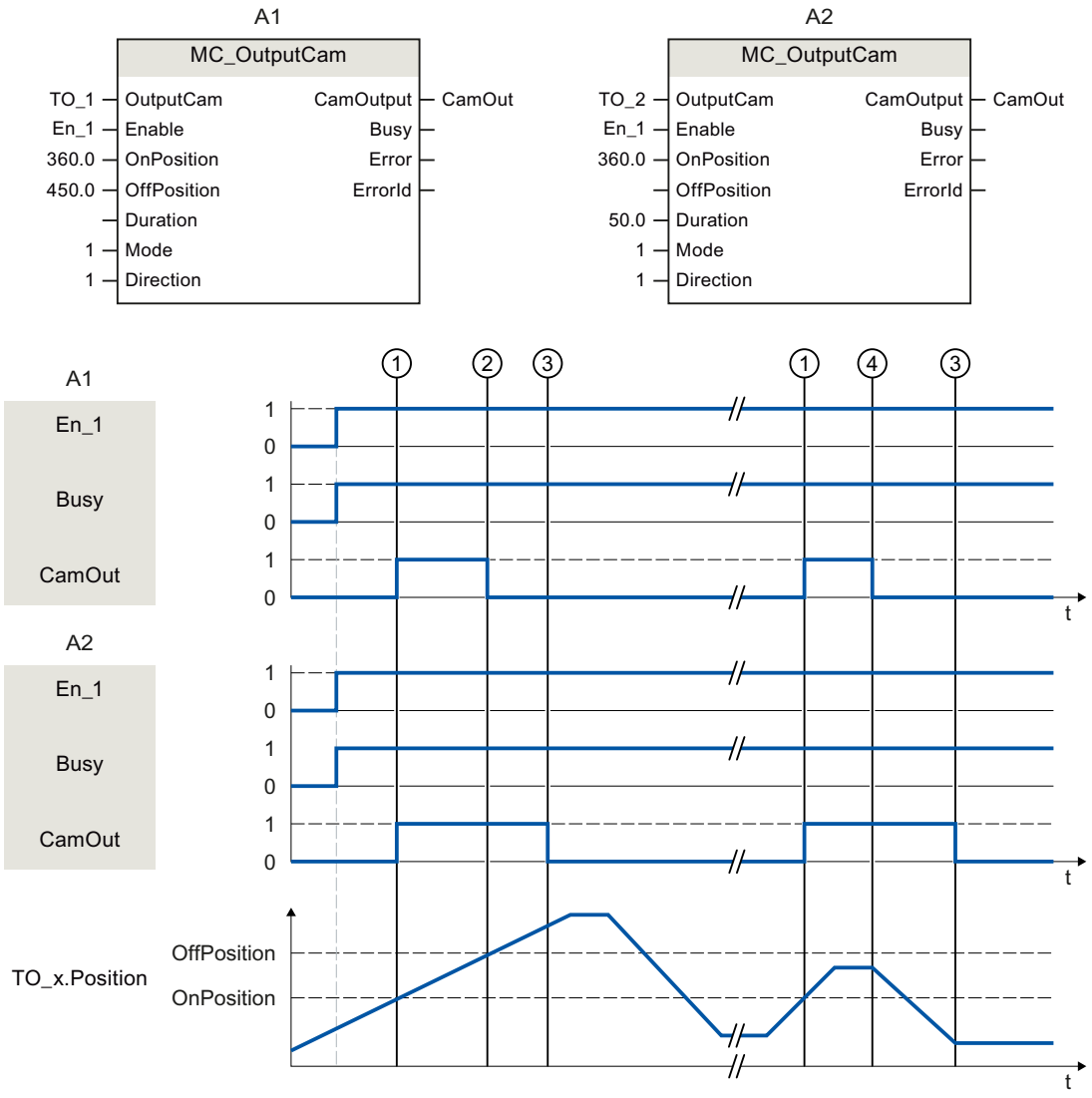

A1 Distance output cam

A2 Time-based output cam

An output cam is activated with "Enable" = TRUE. The output cam is output corresponding to the parameters.

The processing of the output cam is indicated with "Busy" = TRUE.

When activation direction "Direction" = 1 (positive) is set, the output cam behaves as follows, regardless of the output cam type setting:

- The distance output cam switches on at "OnPosition" ① and switches off again at "OffPosition" ②. At a direction reversal the output cam is switched off ④.
- The time-based output cam switches on at "OnPosition" ① and switches off again after expiration of the specified time "Duration" ③. A switched on time-based output cam remains active for the specified switch-on duration "Duration", even if overtravel of the start position occurs again in the reverse direction.

The switching state of the output cam is output using "CamOutput".

With "Enable" = FALSE, the specified cam type is deactivated. "Busy" = FALSE shows that the output cam is not being processed.

MC_CamTrack V4 (S7-1500, S7-1500T)

MC_CamTrack: Activate/deactivate cam track V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_CamTrack", the processing of a cam track is enabled.

Applies to

- Cam track

Requirement

- The technology object has been configured correctly.
- The higher-level technology object must have a valid position.
- Setpoint output cams are not output for an axis in non-position-controlled operation.

Override response

- The "MC_CamTrack" is aborted by the disabling of the cam track with "MC_CamTrack.Enable" = FALSE.
- Only one instance of the "MC_CamTrack" function block can be active on a cam track in the user program. A second instance of the "MC_CamTrack" function block on a currently active cam track is rejected with an error.
- The cam track is recalculated at a change of the parameters of the Motion Control instruction "MC_CamTrack" or in the technology data block. The cam track is processed in accordance with all parameter settings.

Parameters

The following table shows the parameters of Motion Control instruction "MC_CamTrack":

Parameters	Declaration	Data type	Default value	Description	
CamTrack	INPUT	TO_CamTrack	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The cam track is processed.
				FALSE	The cam track is disabled.
Mode	INPUT	INT	1	0	When "Enable" = TRUE: The cam track processing is activated immediately. Cam track data takes effect immediately Previously activated distance output cams are aborted if their track signal is not still set due to the changed cam track data. Previously activated time-based output cams are always aborted.
					When "Enable" = FALSE: The cam track processing is ended immediately. Distance output cams / time-based output cams are aborted immediately.
				1	When "Enable" = TRUE: The cam track processing is activated immediately/with the next track cycle: • When a cam track is activated for the first time, the cam track processing begins immediately. • If the cam track processing is already active, the current cam track is output up to the end of the track cycle. The new cam track data then takes effect.
					When "Enable" = FALSE: The cam track processing is ended at the end of the cam track
				2	When "Enable" = TRUE: The cam track output is switched on immediately and remains switched on
					When "Enable" = FALSE: The cam track output is switched off immediately.
InvertOutput	INPUT	BOOL	FALSE	Inverted output	
				TRUE	The track output is output inverted
				FALSE	The track output is output non-inverted
TrackOutput	OUTPUT	BOOL	-	Indicates the switching state of the cam track.	
Busy	OUTPUT	BOOL	FALSE	TRUE	The processing of the cam track is active.

Parameters	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

MC_CamTrack: Function chart V4 (S7-1500, S7-1500T)

Function chart: Activate cam track

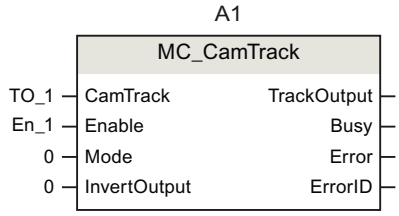

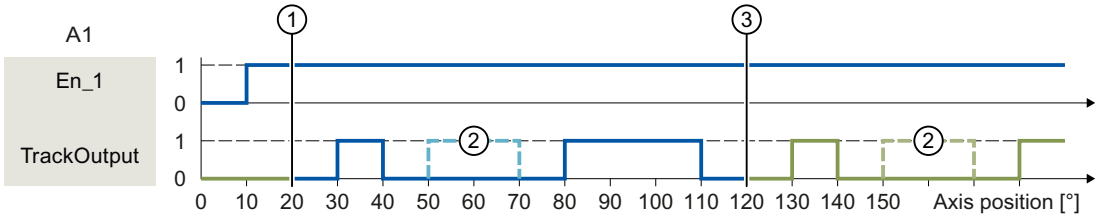

① Axis reference position
② Output cam configured as invalid
③ Start of cyclically continued cam track

A cam track is activated with "Enable" = TRUE. The cam track is output in accordance with the parameters set in the technology data block.

Tag		Value	Description
<TO>Parameter.			
	CamTrackType	0	Specified output cam type for the cam track is distance output cam
	ReferencePosition	20.0	Specified axis reference position for the cam track
	CamTrackLength	100.0	Specified length of the cam track
	CamTrack[1].Existent	TRUE	The first distance output cam of the cam track is specified as valid
	CamTrack[1].OnPosition	10.0	Start position for the first output cam of the cam track
	CamTrack[1].OffPosition	20.0	End position for the first output cam of the cam track
	CamTrack[2].Existent	FALSE	The second distance output cam of the cam track is specified as invalid
	CamTrack[2].OnPosition	30.0	Start position for the second output cam of the cam track
	CamTrack[2].OffPosition	50.0	End position for the second output cam of the cam track
	CamTrack[3].Existent	TRUE	The third distance output cam of the cam track is specified as valid
	CamTrack[3].OnPosition	60.0	Start position for the third output cam of the cam track
	CamTrack[3].OffPosition	90.0	End position for the third output cam of the cam track

Synchronous motion (S7-1500, S7-1500T)

MC_GearIn V4 (S7-1500, S7-1500T)

MC_GearIn: Start gear synchronization V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_GearIn", you start a gearing operation between a leading axis and a following axis.

You define the dynamic behavior of the following axis for synchronization with parameters "Jerk", "Acceleration" and "Deceleration".

The synchronization duration and distance are dependent on the following parameters:

- Start time of the "MC_GearIn" job
- Dynamics of the following axis at the start time
- Dynamic settings for synchronization
- Dynamics of the leading axis

You specify the gear ratio as the relationship between two integers (numerator/denominator) with the parameters "RatioNumerator" and "RatioDenominator".

The numerator of the gear ratio is specified as positive or negative. This yields the following behavior:

- Positive gear ratio:
  The leading and following axes move in the same direction.
- Negative gear ratio:
  The following axis moves in the opposite direction of the leading axis.

You can start synchronous operation when the leading axis is at a standstill or when it is in motion.

Applies to

- Synchronous axis

Requirement

- The technology objects of the leading axis and the following axis have been configured correctly.
- The leading axis is a positioning axis, synchronous axis, or external encoder.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".
- The following axis is enabled.

Override response

The override response for "MC_GearIn" jobs is described in section Override response V4: Synchronous operation jobs (Page 3252).

Disabling the following axis with "MC_Power.Enable" = FALSE aborts the synchronous operation in every status.

Disabling the leading axis with "MC_Power", in contrast, does not abort synchronous operation. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_GearIn":

Parameters	Declaration	Data type	Default value	Description	
Master	INPUT	TO_Axis	-	Leading axis technology object	
Slave	INPUT	TO_Synchronous-Axis	-	Following axis technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
RatioNumerator	INPUT	DINT	1	Gear ratio numerator Permitted integer values: -2147483648 to 2147483647 (value 0 not permitted)	
RatioDenominator	INPUT	DINT	1	Gear ratio denominator Permitted integer values: 1 to 2147483647	
Acceleration	INPUT	LREAL	-1.0	Acceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)

Parameters	Declaration	Data type	Default value	Description	
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	Constant acceleration velocity profile The specified value is used.
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
InGear	OUTPUT	BOOL	FALSE	TRUE	Synchronous operation reached The following axis is synchronized and moves synchronously to the leading axis.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Starting synchronous operation

To start synchronous operation with the Motion Control instruction "MC_GearIn", follow these steps:

1. Check the requirements indicated above.
2. Specify the leading axis, the following axis and the gear ratio at the corresponding parameters.
3. Start the "MC_GearIn" job with a positive edge at parameter "Execute".
   The following axis is synchronized to the leading value of the leading axis. If the "InGear" parameter shows the value "TRUE", the following axis is synchronized and moves synchronously to the leading axis. The parameters "InGear" and "Busy" show the value "TRUE" until the "MC_GearIn" job is overridden by another Motion Control job.

See also

Override response V4: Synchronous operation jobs (Page 3252)

MC_GearIn: Function chart V4 (S7-1500, S7-1500T)

Function chart: Synchronizing and switching the leading value

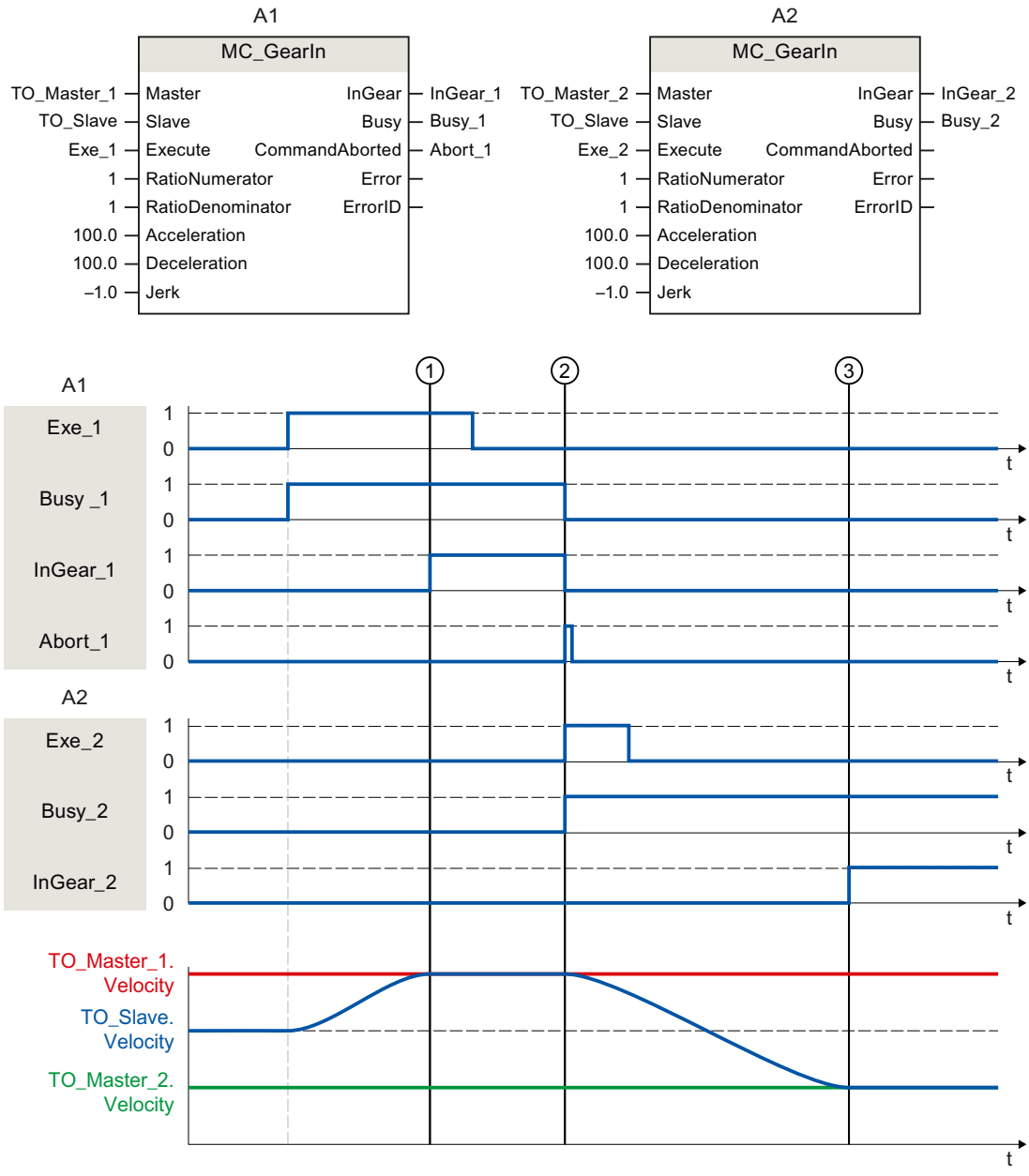

Using "Exe_1", an "MC_GearIn" job (A1) is initiated. The following axis (TO_Slave) is synchronized to the leading axis (TO_Master_1). "InGear_1" signals at time ① that the following axis is synchronized and moves synchronously to the leading axis.

At time ②, synchronous operation is overridden by another "MC_GearIn" job (A2). The abort is signaled via "Abort_1". The following axis is synchronized to the new leading axis (TO_Master_2). "InGear_2" signals at time ③ that the following axis is synchronized and moves synchronously to the leading axis.

MC_GearInPos V4 (S7-1500T)

MC_GearInPos: Start gearing with specified synchronous positions V4 (S7-1500T)

Description

With the Motion Control instruction "MC_GearInPos", you start a gearing operation between a leading axis and a following axis. The synchronous operation is synchronized in advance of the specified synchronous position for the leading and following axis.

The following types of synchronization are possible:

- Synchronization using dynamic parameters ("SyncProfileReference" = 0)
- Synchronization using leading value distance ("SyncProfileReference" = 1)

You specify the gear ratio as the relationship between two integers (numerator/denominator) with the parameters "RatioNumerator" and "RatioDenominator".

The numerator of the gear ratio is specified as positive or negative. This yields the following behavior:

- Positive gear ratio:
  The leading and following axes move in the same direction.
- Negative gear ratio:
  The following axis moves in the opposite direction of the leading axis.

You can start synchronous operation when the leading axis is at a standstill or when it is in motion.

Applies to

- Synchronous axis

Requirement

- The technology objects of the leading axis and the following axis have been configured correctly.
- The leading axis is a positioning axis, synchronous axis, or external encoder.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".
- The following axis is enabled.
- When using the "MasterStartDistance", the leading axis must be at least the specified distance from the "MasterSyncPosition" when starting the job.

Override response

The override response for "MC_GearInPos" jobs is described in section Override response V4: Synchronous operation jobs (Page 3252).

Disabling the following axis with "MC_Power.Enable" = FALSE aborts the synchronous operation in every status.

Disabling the leading axis with "MC_Power", in contrast, does not abort synchronous operation. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_GearInPos":

Parameters	Declaration	Data type	Default value	Description	
Master	INPUT	TO_Axis	-	Leading axis technology object	
Slave	INPUT	TO_Synchronous-Axis	-	Following axis technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
RatioNumerator	INPUT	DINT	1	Gear ratio numerator Permitted integer values: -2147483648 to 2147483647 (value 0 not permitted)	
RatioDenominator	INPUT	DINT	1	Gear ratio denominator Permitted integer values: 1 to 2147483647	
MasterSyncPosition	INPUT	LREAL	0.0	Synchronous position of leading axis Position of the leading axis starting from which the leading and following axes move synchronously	
SlaveSyncPosition	INPUT	LREAL	0.0	Synchronous position of following axis Position of the following axis starting from which the leading and following axes move synchronously	
SyncProfileReference	INPUT	DINT	1	Type of synchronization	
				0	Synchronization using dynamic parameters
				1	Synchronization using leading value distance
MasterStartDistance	INPUT	LREAL	1.0	Leading value distance (with "SyncProfileReference" = 1)	
Velocity	INPUT	LREAL	-1.0	Velocity (with "SyncProfileReference" = 0)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)

Parameters	Declaration	Data type	Default value	Description	
Acceleration	INPUT	LREAL	-1.0	Acceleration (with "SyncProfileReference" = 0)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration (with "SyncProfileReference" = 0)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk (with "SyncProfileReference" = 0)	
				> 0.0	Constant acceleration velocity profile The specified value is used.
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
SyncDirection	INPUT	DINT	3	Direction of synchronization (in effect for axes with activated Modulo setting)	
				1	Positive direction The following axis may only travel in positive direction during synchronization.
				2	Negative direction The following axis may only travel in negative direction during synchronization.
				3	Shortest distance Changes in direction are permitted for the following axis during synchronization.
StartSync	OUTPUT	BOOL	FALSE	TRUE	The following axis is synchronized to the leading axis.
InSync	OUTPUT	BOOL	FALSE	TRUE	Synchronous operation reached The following axis is synchronized and moves synchronously to the leading axis.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.

Parameters	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Starting synchronous operation

To start synchronous operation with the Motion Control instruction "MC_GearInPos", follow these steps:

1. Check the requirements indicated above.
2. Specify the leading axis, the following axis, the gear ratio and the synchronous position with the corresponding parameters.
3. Start the "MC_GearInPos" job with a positive edge at parameter "Execute".
   The following axis is synchronized to the leading value of the leading axis. If the "InSync" parameter shows the value "TRUE", the following axis is synchronized and moves synchronously to the leading axis. The parameters "InSync" and "Busy" show the value "TRUE" until the "MC_GearInPos" job is overridden by another Motion Control job.

See also

Override response V4: Synchronous operation jobs (Page 3252)

MC_GearInPos: Function chart V4 (S7-1500T)

Function chart: Synchronization using dynamic parameters/leading value distance

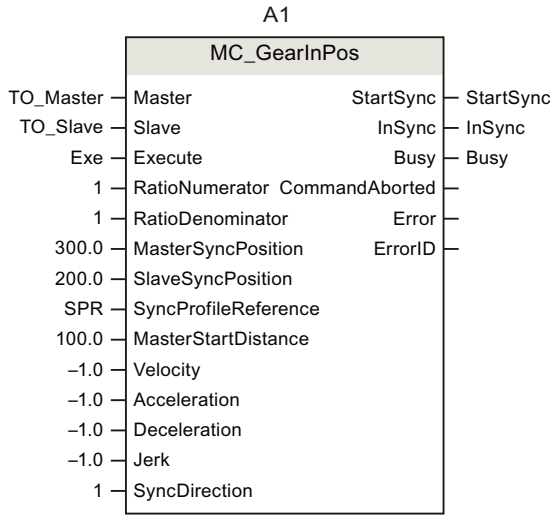

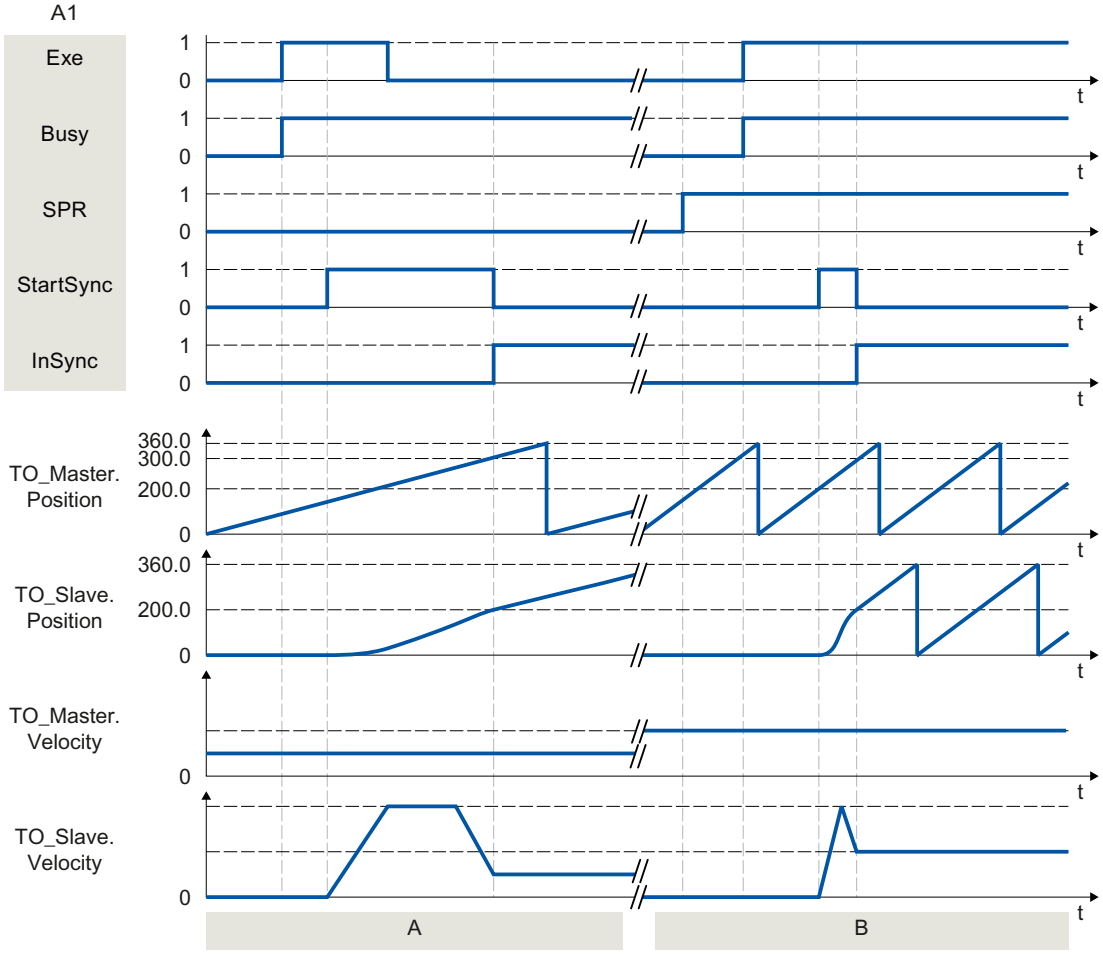

Section A	Using "Exe", an "MC_GearInPos" job (A1) is initiated. The following axis (TO_Slave) is synchronized to the leading axis (TO_Master) by means of the specified dynamic parameters. The distance required for synchronization is calculated by the system. The start of the synchronization is displayed with "StartSync". When the specified synchronous position is reached, "InSync" signals that the following axis is synchronized and moving synchronously to the leading axis.
Section B	Using "Exe", an "MC_GearInPos" job (A1) is initiated. The following axis (TO_Slave) is synchronized to the leading axis (TO_Master) by means of the specified leading value distance. The dynamics required for synchronization is calculated by the system. The start of the synchronization is displayed with "StartSync". "InSync" signals that the following axis is synchronized and moving synchronously to the leading axis.

MC_PhasingAbsolute V4 (S7-1500T)

MC_PhasingAbsolute: Absolute shift of leading value on the following axis V4 (S7-1500T)

Description

With the Motion Control instruction "MC_PhasingAbsolute", you shift the leading value on a following axis during gearing with "MC_GearIn" and "MC_GearInPos" as an absolute shift. The position of the leading axis is not affected by this.

You define the dynamic response of the motion of the following axis with the parameters "Velocity", "Jerk", "Acceleration", and "Deceleration". The dynamic values are added to the values of the synchronous operation motion.

Applies to

- Synchronous axis

Requirement

- The technology objects of the leading axis and the following axis have been configured correctly.
- The leading axis is a positioning axis, synchronous axis, or external encoder.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".
- By means of the Motion Control instruction "MC_GearIn" or "MC_GearInPos", the following axis is synchronized to the leading axis ("MC_GearIn.InGear" = TRUE or "MC_GearInPos.InSync" = TRUE).
- The following axis is enabled.

Override response

The override response for "MC_PhasingAbsolute" jobs is described in section Override response V4: Synchronous operation jobs (Page 3252).

Disabling the leading axis with "MC_Power.Enable" = FALSE does not abort the leading value shift. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_PhasingAbsolute":

Parameters	Declaration	Data type	Default value	Description	
Master	INPUT	TO_Axis	-	Leading axis technology object	
Slave	INPUT	TO_Synchronous-Axis	-	Following axis technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
PhaseShift	INPUT	LREAL	0.0	Absolute leading value shift	
Velocity	INPUT	LREAL	-1.0	Velocity of the following axis for leading value shift (added to synchronous operation motion)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration of the following axis for leading value shift (added to synchronous operation motion)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration of the following axis for leading value shift (added to synchronous operation motion)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk of the following axis for leading value shift (added to synchronous operation motion)	
				> 0.0	Constant acceleration velocity profile The specified value is used.
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)

Parameters	Declaration	Data type	Default value	Description	
Done	OUTPUT	BOOL	FALSE	TRUE	Leading value shift is finished.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	
AbsolutePhaseShift	OUTPUT	LREAL	0.0	As long as "Busy" = TRUE: Display of leading value shift completed up to now	

Start absolute leading value shift

To start an absolute leading value shift with the Motion Control instruction "MC_PhasingAbsolute", follow these steps:

1. Check the requirements indicated above.
2. Specify the absolute leading value shift in the "PhaseShift" parameter.
3. Start the "MC_PhasingAbsolute" job with a positive edge at parameter "Execute".
   The current motion state is indicated in parameters "Busy", "AbsolutePhaseShift", "Done" and "Error".

See also

Override response V4: Synchronous operation jobs (Page 3252)

MC_GearIn V4 (Page 3156)

MC_GearInPos V4 (Page 3160)

MC_PhasingAbsolute: Function chart V4 (S7-1500T)

Function chart: Absolute shift of leading value

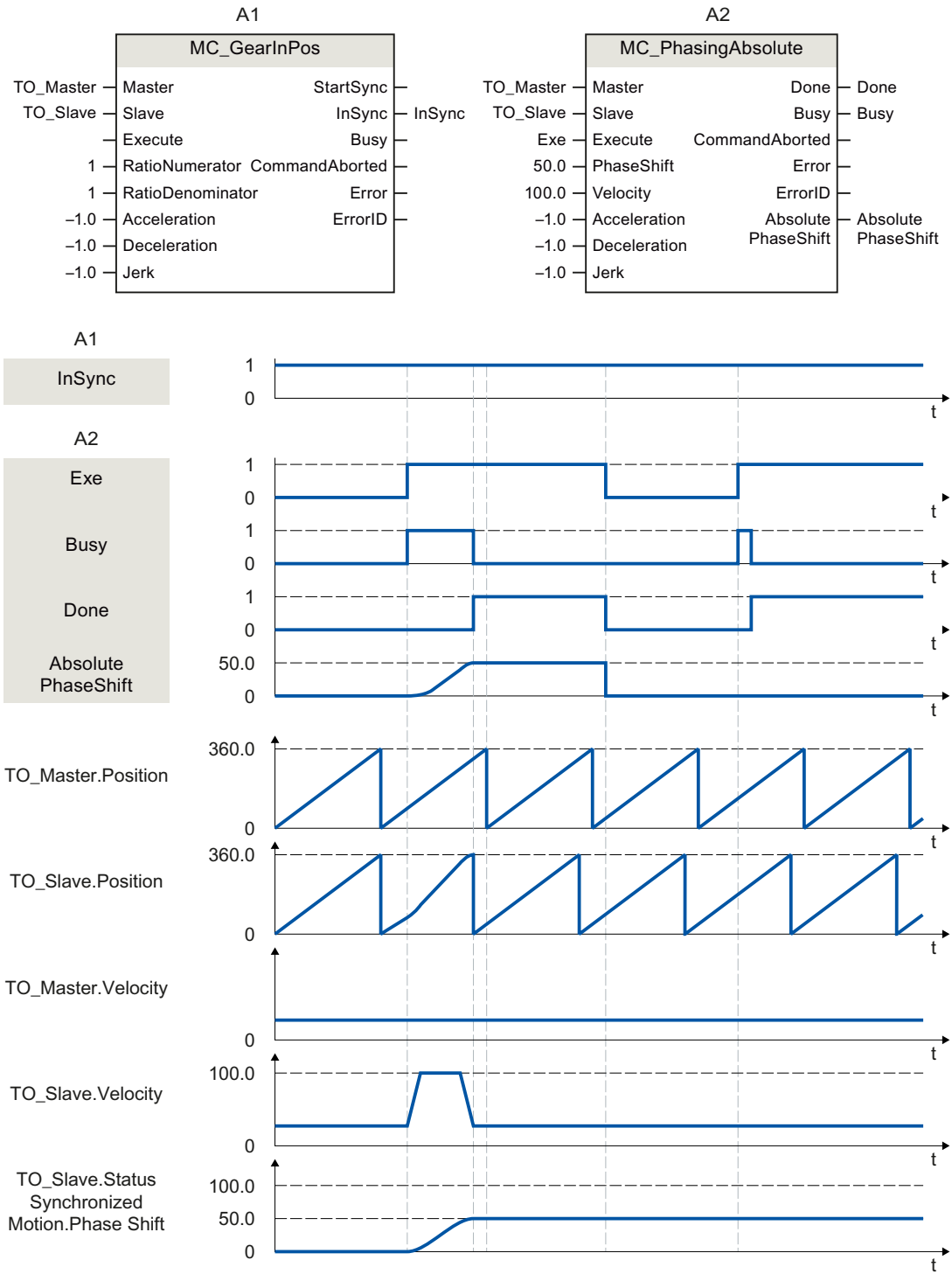

During an active gearing operation with "MC_GearInPos" (A1), a "MC_PhasingAbsolute" job (A2) is initiated using "Exe". The leading value shift is performed with the dynamics specified additively to the synchronous operation motion. "Done" indicates that the leading value was successfully shifted. The leading value shift 50.0 resulting from the job is indicated in "AbsolutePhaseShift". The absolute leading value shift 50.0 is indicated in the tag of the technology object <TO>.StatusSynchronizedMotion.PhaseShift. The motion of the leading axis is not affected.

After the leading value shift, the "MC_PhasingAbsolute" job (A2) is initiated again using "Exe". Because the leading value shift (<TO>.StatusSynchronizedMotion.PhaseShift) is already 50.0, the leading value is not shifted.

MC_PhasingRelative V4 (S7-1500T)

MC_PhasingRelative: Relative shift of leading value on the following axis V4 (S7-1500T)

Description

With the Motion Control instruction "MC_PhasingRelative", you shift the leading value on a following axis during gearing with "MC_GearIn" and "MC_GearInPos" relative to the existing leading value shift. The position of the leading axis is not affected by this.

You define the dynamic response of the motion of the following axis with the parameters "Velocity", "Jerk", "Acceleration", and "Deceleration". The dynamic values are added to the values of the synchronous operation motion.

Applies to

- Synchronous axis

Requirement

- The technology objects of the leading axis and the following axis have been configured correctly.
- The leading axis is a positioning axis, synchronous axis, or external encoder.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".
- By means of the Motion Control instruction "MC_GearIn" or "MC_GearInPos", the following axis is synchronized to the leading axis ("MC_GearIn.InGear" = TRUE or "MC_GearInPos.InSync" = TRUE).
- The following axis is enabled.

Override response

The override response for "MC_PhasingRelative" jobs is described in section Override response V4: Synchronous operation jobs (Page 3252).

Disabling the leading axis with "MC_Power.Enable" = FALSE does not abort the leading value shift. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_PhasingRelative":

Parameters	Declaration	Data type	Default value	Description	
Master	INPUT	TO_Axis	-	Leading axis technology object	
Slave	INPUT	TO_Synchronous-Axis	-	Following axis technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
PhaseShift	INPUT	LREAL	0.0	Relative leading value shift	
Velocity	INPUT	LREAL	-1.0	Velocity of the following axis for leading value shift (added to synchronous operation motion)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration of the following axis for leading value shift (added to synchronous operation motion)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration of the following axis for leading value shift (added to synchronous operation motion)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk of the following axis for leading value shift (added to synchronous operation motion)	
				> 0.0	Constant acceleration velocity profile The specified value is used.
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)

Parameters	Declaration	Data type	Default value	Description	
Done	OUTPUT	BOOL	FALSE	TRUE	Leading value shift is finished.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	
CoveredPhaseShift	OUTPUT	LREAL	0.0	As long as "Busy" = TRUE: Display of leading value shift completed up to now	

Start relative leading value shift

To start a relative leading value shift with the Motion Control instruction "MC_PhasingRelative", follow these steps:

1. Check the requirements indicated above.
2. Specify the relative leading value shift in the "PhaseShift" parameter.
3. Start the "MC_PhasingRelative" job with a positive edge at parameter "Execute".
   The current motion state is indicated in parameters "Busy", "CoveredPhaseShift", "Done" and "Error".

See also

Override response V4: Synchronous operation jobs (Page 3252)

MC_GearIn V4 (Page 3156)

MC_GearInPos V4 (Page 3160)

MC_PhasingRelative: Function chart V4 (S7-1500T)

Function chart: Relative shift of leading value

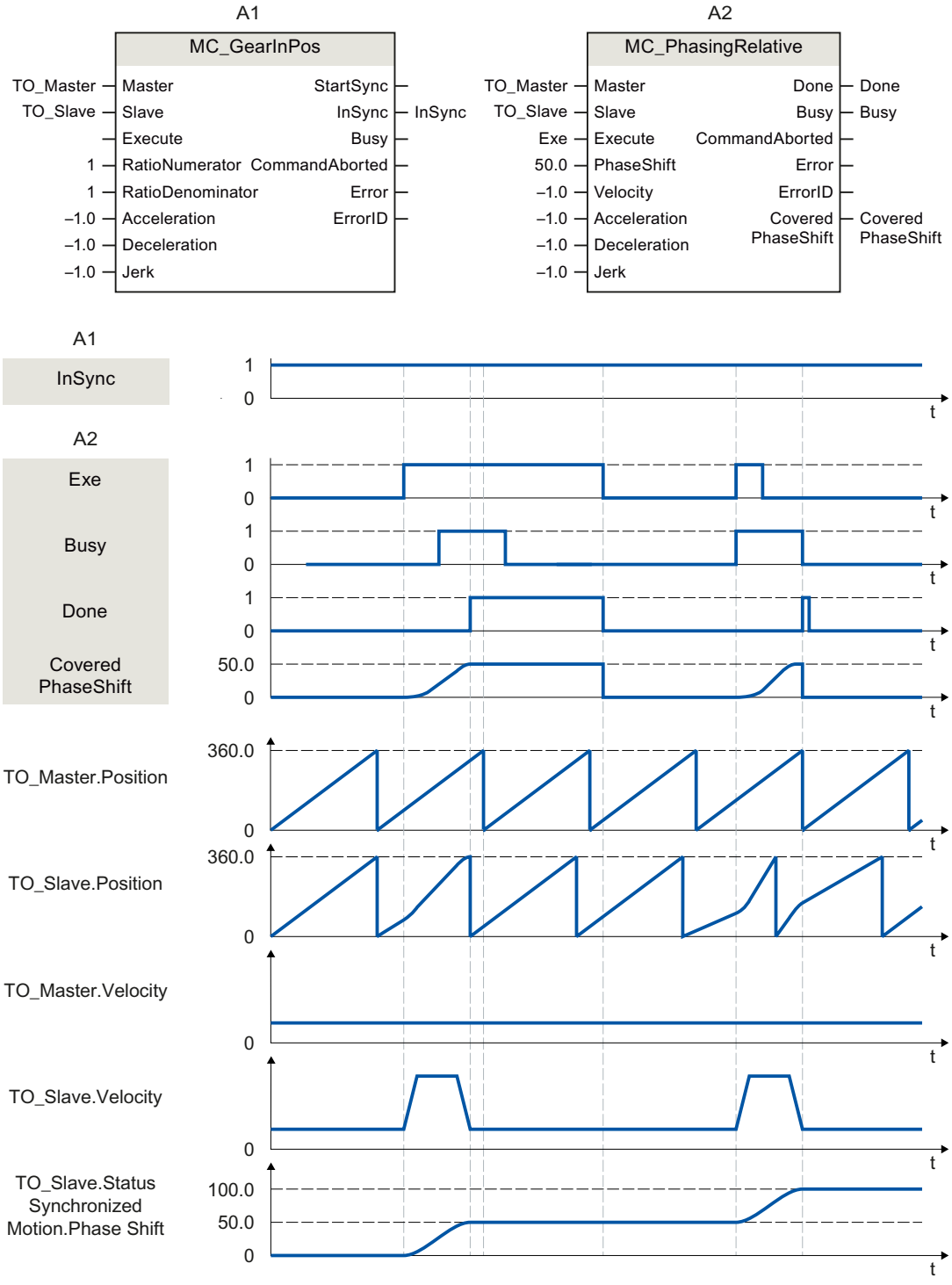

During an active gearing operation with "MC_GearInPos" (A1), a "MC_PhasingRelative" job (A2) is initiated using "Exe". The leading value shift is performed with the dynamics specified additively to the synchronous operation motion. "Done" indicates that the leading value was successfully shifted. The leading value shift 50.0 resulting from the job is indicated in "CoveredPhaseShift". The absolute leading value shift 50.0 is indicated in the tag of the technology object <TO>.StatusSynchronizedMotion.PhaseShift. The motion of the leading axis is not affected.

After the leading value shift, the "MC_PhasingRelative" job (A2) is initiated again using "Exe". The leading value shift is performed again with the dynamics specified additively to the synchronous operation motion. "Done" indicates that the leading value was successfully shifted. The leading value shift 50.0 resulting from the job is indicated in "CoveredPhaseShift". The absolute leading value shift 100.0 is indicated in the tag of the technology object <TO>.StatusSynchronizedMotion.PhaseShift.

MC_CamIn V4 (S7-1500T)

MC_CamIn: Start camming V4 (S7-1500T)

Description

With the Motion Control instruction "MC_CamIn", you start a camming operation between a leading axis and a following axis.

With the "SyncProfileReference" parameter, you specify the synchronization profile.

- **"SyncProfileReference" = 0 or 1**
  The synchronous operation is synchronized in advance of the specified synchronous position of the leading axis.
  The synchronous position of the leading axis starting from which the leading and following axes move synchronously results from the starting position of the cam and the "MasterSyncPosition" and "MasterOffset" parameters.
  With the "MasterOffset" parameter, you offset the leading values of the cam (with "SyncProfileReference" = 0 or 1). This determines the position of the cam in relation to the leading value of the synchronous operation function. This is how you move the cam to the required position range.
- **"SyncProfileReference" = 2**
  Synchronous operation is immediately set to synchronous. You use the "MasterSyncPosition" parameter to set the exact synchronous position in the cam. This setting is used mainly for synchronizing at a standstill.

A cam is defined between the start position (<TO>.StatusCam.StartLeadingValue) and end position (<TO>.StatusCam.EndLeadingValue) after interpolation. The specification for leading and following value range in the configuration of the technology object only effect the display in the graphical editor.

With the "MasterSyncPosition" parameter, you specify the synchronization position in the cam relative to the starting position of the cam. Synchronization is completed upon reaching the synchronous position. To run through the entire cam, specify the value 0.0 (default value) in "MasterSyncPosition". With "MasterSyncPosition" ≠ 0.0, you move the synchronous position within the cam without changing the position of the cam.

The figure below shows the basic effect of the leading value and following value offset as well as the definition of the cam with the following parameter values:

- Start position of the cam > 0
- "MasterSyncPosition" > 0

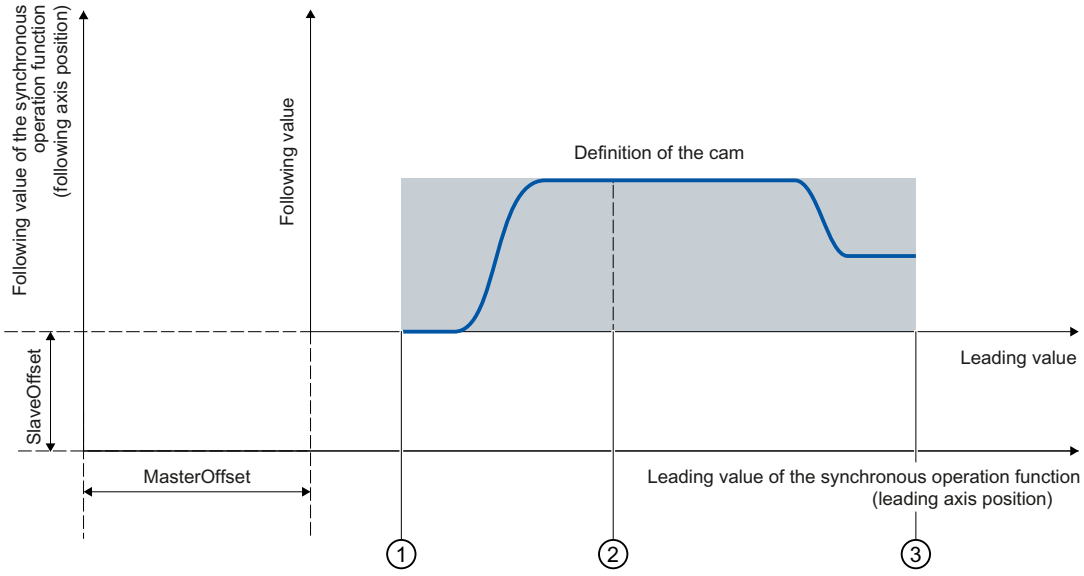

① Start position of the cam

First defined interpolation point/start of the first segment of the cam (<TO>.StatusCam.StartLeadingValue)

② Completion of synchronization ("MasterSyncPosition")

③ End position of the cam

Last defined interpolation point/end of the last segment of the cam (<TO>.StatusCam.EndLeadingValue)

The figure below shows the basic effect of scaling the cam with the parameters "MasterScaling" and "SlaveScaling":

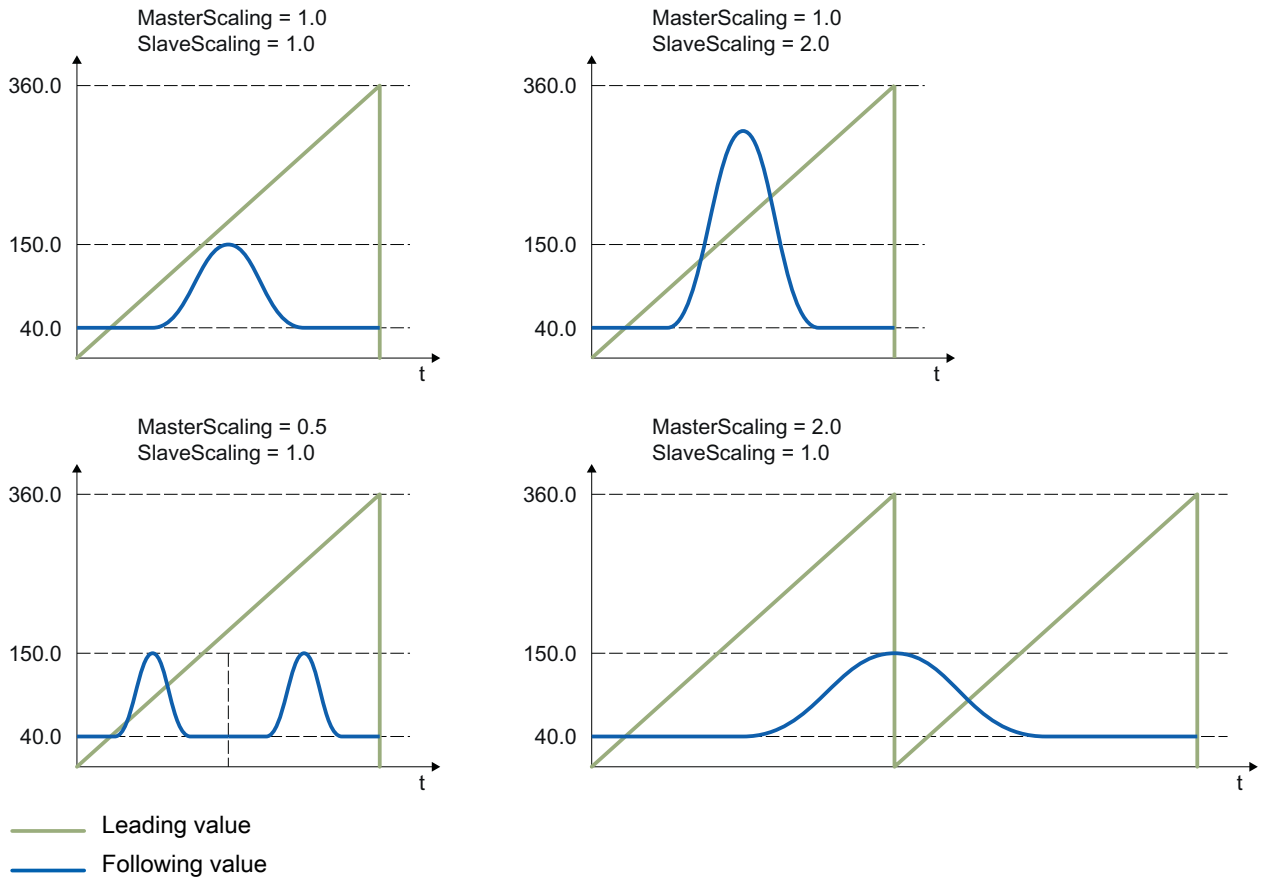

The following types of synchronization are possible:

- Synchronization using dynamic parameters ("SyncProfileReference" = 0)
- Synchronization using leading value distance ("SyncProfileReference" = 1)
- Set direct synchronous ("SyncProfileReference" = 2)

You can start synchronous operation when the leading axis or following axis is at a standstill or when it is in motion.

Applies to

- Synchronous axis

Requirement

- The technology objects of the leading axis, following axis, and cam have been configured correctly.
- The leading axis is a positioning axis, synchronous axis, or external encoder.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".

- The following axis is enabled.
- The cam is interpolated with "MC_InterpolateCam".
- When using the "MasterStartDistance", the leading axis must be at least the specified distance from the "MasterSyncPosition" when starting the job.

Override response

The override response for "MC_CamIn" jobs is described in section Override response V4: Synchronous operation jobs (Page 3252).

Disabling the following axis with "MC_Power.Enable" = FALSE aborts the synchronous operation in every status.

Disabling the leading axis with "MC_Power", in contrast, does not abort synchronous operation. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_CamIn":

Parameter	Declaration	Data type	Default value	Description	
Master	INPUT	TO_Axis	-	Leading axis technology object	
Slave	INPUT	TO_Synchronous-Axis	-	Following axis technology object	
Cam	INPUT	TO_Cam	-	Cam technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
MasterOffset	INPUT	LREAL	0.0	Offset of the leading values of cam (with "SyncProfileReference" =0 or 1) The cam technology object is not changed.	
SlaveOffset	INPUT	LREAL	0.0	Offset of the following values of cam (with "SyncProfileReference" =0 or 1) The cam technology object is not changed.	
MasterScaling	INPUT	LREAL	1.0	Scaling the leading values of the cam The cam technology object is not changed.	
SlaveScaling	INPUT	LREAL	1.0	Scaling the following values of the cam The cam technology object is not changed.	
MasterSyncPosition	INPUT	LREAL	0.0	Synchronous position End position of the synchronization on the leading value side in relation to the start position of the cam (<TO>.StatusCam.StartLeadingValue) The value must be within the definition of the cam.	

Parameter	Declaration	Data type	Default value	Description	
SyncProfileReference	INPUT	DINT	1	Synchronization profile	
				0	Synchronization using dynamic parameters
				1	Synchronization using leading value distance
				2	Direct synchronous setting
MasterStartDistance	INPUT	LREAL	0.0	Leading value distance (distance of the leading axis during the synchronization) (with "SyncProfileReference" = 1)	
Velocity	INPUT	LREAL	-1.0	Velocity (with "SyncProfileReference" = 0)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration (with "SyncProfileReference" = 0)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration (with "SyncProfileReference" = 0)	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk (with "SyncProfileReference" = 0)	
				> 0.0	Constant acceleration velocity profile The specified value is used.
				= 0.0	Trapezoid velocity profile
				< 0.0	The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)

Parameter	Declaration	Data type	Default value	Description	
ApplicationMode	INPUT	DINT	0	Application of the cam	
				0	Once/not cyclic
				1	Cyclic (absolute application on the following value side)
				2	Cyclic appending (continuously appending on the following value side)
SyncDirection	INPUT	DINT	3	Direction of synchronization (in effect for axes with activated Modulo setting)	
				1	Positive direction The following axis may only travel in positive direction during synchronization.
				2	Negative direction The following axis may only travel in negative direction during synchronization.
				3	Shortest distance Changes in direction are permitted for the following axis during synchronization.
StartSync	OUTPUT	BOOL	FALSE	TRUE	The following axis is synchronized to the leading axis.
InSync	OUTPUT	BOOL	FALSE	TRUE	Synchronous operation reached The following axis is synchronized and moves synchronously to the leading axis.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	
EndOfProfile	OUTPUT	BOOL	FALSE	TRUE	The end of the cam has been reached. Displayed for at least one call of "MC_CamIn" in the user program when the cam is used cyclically.

Starting synchronous operation

To start synchronous operation with the Motion Control instruction "MC_CamIn", follow these steps:

1. Check the requirements indicated above.
2. Specify the leading axis, the following axis, the utilized cam and the synchronous position in the corresponding parameters.
3. Start the "MC_CamIn" job with a positive edge at parameter "Execute".
   The following axis is synchronized to the leading value of the leading axis. If the "InSync" parameter shows the value "TRUE", the following axis is synchronized and moves synchronously to the leading axis. With cyclic application of the cam, "InSync" and "Busy" show the value "TRUE" until the "MC_CamIn" job is overridden by another Motion Control job. With non-cyclic application of the cam, "InSync" and "Busy" are set to the value "FALSE" when the parameter "EndOfProfile" is set to the value "TRUE".

See also

Override response V4: Synchronous operation jobs (Page 3252)

MC_CamIn: Function chart V4 (S7-1500T)

Function chart: Synchronizing and switching the cam

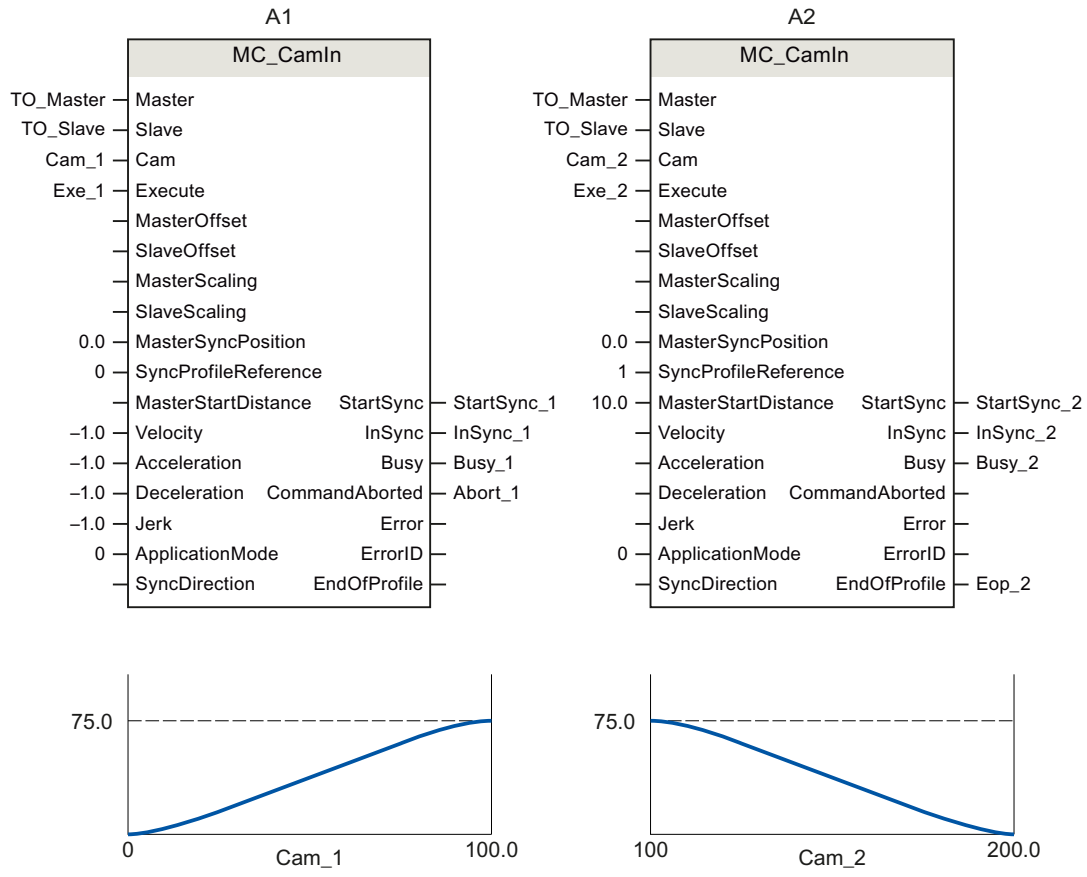

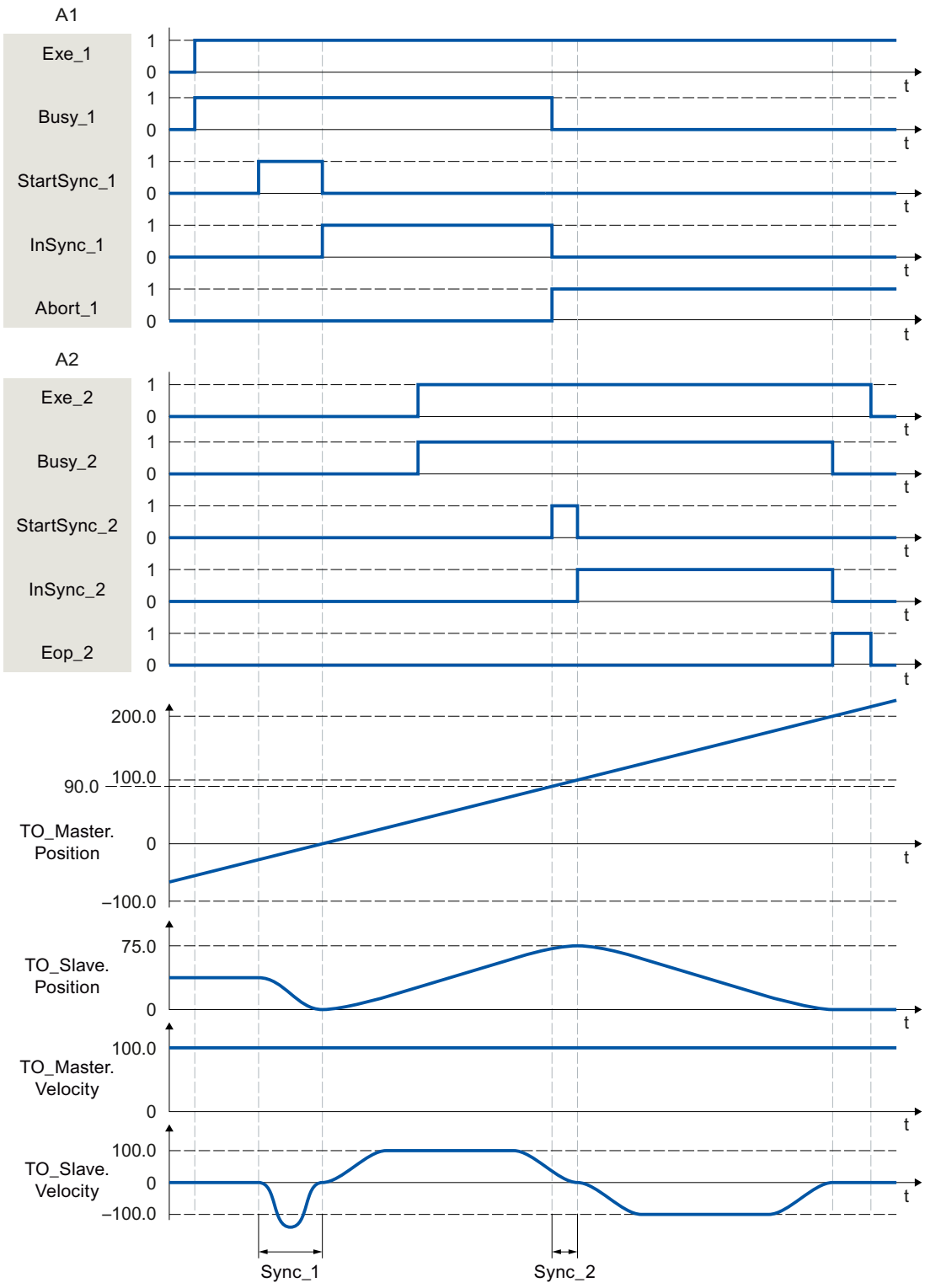

Using "Exe_1", an "MC_CamIn" job (A1) is initiated. The start of the synchronization is displayed with "StartSync_1". The following axis (TO_Slave) is synchronized to the cam (Cam_1) within the range "Sync_1" by means of the specified dynamic parameters. "InSync_1" signals that the following axis is synchronized and moving synchronously to the leading axis.

The synchronous operation is overridden by another "MC_CamIn" job (A2). The abort is signaled via "Abort_1". The start of the synchronization is displayed with "StartSync_2". The following axis is synchronized to the new cam (Cam_2) within the range "Sync_2" by means of the specified leading value distance. Within the range "Sync_2", the axis does not follow the cam "Cam_1" The axis is moved with the dynamics calculated by the system to the synchronous position of A2. "InSync_2" signals that the following axis is synchronized and moving synchronously to the leading axis.

MC_SynchronizedMotionSimulation V4 (S7-1500T)

MC_SynchronizedMotionSimulation: Simulate synchronous operation V4 (S7-1500T)

Description

With the Motion Control instruction "MC_SynchronizedMotionSimulation", you simulate an active synchronous operation on a following axis. As a result, a synchronous operation remains active when the following axis is disabled with a "MC_Power" job. The following axis does not have to be synchronized again after being enabled again.

With the start of a "MC_SynchronizedMotionSimulation" job, the velocity setpoint from the synchronous operation is set to zero. If an overlaid movement is active on the following axis at the start of the simulation, the setpoints of this overlaid movement will continue to be output.

Setpoints of motion jobs that are started during synchronous operation simulation are output to the drive.

If the position of the following axis at the end of the simulation differs from the position at the start of the simulation, this triggers a setpoint step-change.

Applies to

- Synchronous axis

Requirement

- The technology object has been configured correctly.
- The following axis is a synchronous axis.
- Synchronous operation is active on the technology object in status "Synchronous" (<TO>.StatusWord.X22 = TRUE).

Override response

An "MC_SynchronizedMotionSimulation" job is not aborted by any other Motion Control job. The simulated synchronous operation remains active even when the following axis is disabled with "MC_Power.Enable" = FALSE.

A restart of the technology object stops the simulation and aborts the synchronous operation.

A new "MC_SynchronizedMotionSimulation" job does not abort any other Motion Control jobs. With "MC_SynchronizedMotionSimulation.Enable" = TRUE, synchronous operation jobs are rejected.

Parameters

The following table shows the parameters of Motion Control instruction "MC_SynchronizedMotionSimulation":

Parameters	Declaration	Data type	Default value	Description	
Slave	INPUT	TO_Synchronous-Axis	-	Following axis technology object	
Enable	INPUT	BOOL	FALSE	TRUE	Simulation of synchronous operation is started.
				FALSE	Simulation of the synchronous operation is stopped.
InSimulation	OUTPUT	BOOL	FALSE	TRUE	Synchronous operation is being simulated
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Continuing synchronous operation when the following axis is disabled

In order not to stop synchronous operation with "MC_SynchronizedMotionSimulation" Motion Control instruction when the following axis is disabled, follow these steps:

1. Check the requirements indicated above.
2. Specify the following axis in the "Slave" parameter.
3. Bring the leading axis to a standstill (e.g. with "MC_Halt").
4. Start the simulation of the synchronous operation on the following axis with "MC_SynchronizedMotionSimulation.Enable" = TRUE.
5. When the "InSimulation" parameter shows the value TRUE, disable the following axis. The synchronous operation remains active in the simulation.
6. To reset synchronous operation again after the following axis is enabled, stop synchronous operation simulation with "MC_SynchronizedMotionSimulation.Enable" = FALSE.
   The synchronous operation simulation is stopped. The following axis follows the leading axis without re-synchronization.

Cam (S7-1500T)

MC_InterpolateCam V4 (S7-1500T)

MC_InterpolateCam: Interpolate cam disc V4 (S7-1500T)

Description

With the Motion Control instruction "MC_InterpolateCam", you interpolate a cam.

The interpolation closes the gaps between the defined interpolation points and segments of the cam. The cam is interpolated between the following values in the definition range:

- First defined interpolation point/start of the first segment of the cam (<TO>.StatusCam.StartLeadingValue)
- Last defined interpolation point/end of the last segment of the cam (<TO>.StatusCam.EndLeadingValue)

After interpolation, an explicit value in the value range is assigned to each value in the definition range.

The interpolation type defines how missing ranges are interpolated. You specify the interpolation type in the configuration of the technology object. The following interpolation methods are possible:

- Linear interpolation
- Interpolation with cubic splines
- Interpolation with Bézier splines

Applies to

- Cam

Requirement

- The technology object has been configured correctly.
- The cam is not currently being used, e.g. for camming.

Override response

- An "MC_InterpolateCam" job is not aborted by any other Motion Control job.
- A new "MC_InterpolateCam" job does not abort any active motion control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_InterpolateCam":

Parameters	Declaration	Data type	Default value	Description	
Cam	INPUT	TO_Cam	-	Cam technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Done	OUTPUT	BOOL	FALSE	TRUE	The cam is interpolated.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Interpolating a cam disc

To interpolate a cam with the "MC_InterpolateCam" Motion Control instruction, follow these steps:

1. Check the requirements indicated above.
2. Specify the cam to be interpolated in the "Cam" parameter.
3. Start the "MC_InterpolateCam" job with a positive edge at parameter "Execute".
   The cam is interpolated. When the "Done" parameter shows the value "TRUE", the interpolation is finished.

MC_GetCamFollowingValue V4 (S7-1500T)

MC_GetCamFollowingValue: Read out following value of a cam disc V4 (S7-1500T)

Description

With the Motion Control instruction "MC_GetCamFollowingValue", you read the following value and the first and second derivative of the following value for a leading value from a cam.

Applies to

- Cam

Requirement

- The technology object has been configured correctly.
- The cam is interpolated.

Override response

- An "MC_GetCamFollowingValue" job is not aborted by any other Motion Control job.
- A new "MC_GetCamFollowingValue" job does not abort any active motion control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_GetCamFollowingValue":

Parameters	Declaration	Data type	Default value	Description	
Cam	INPUT	TO_Cam	-	Cam technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
LeadingValue	INPUT	LREAL	0.0	Leading value for which the following value is read	
Done	OUTPUT	BOOL	FALSE	TRUE	The following value was read.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	
Value	OUTPUT	LREAL	-	Read following value (position) (valid when "Done" = TRUE)	
FirstDerivative	OUTPUT	LREAL	-	First derivative of read following value (valid when "Done" = TRUE)	
SecondDerivative	OUTPUT	LREAL	-	Second derivative of read following value (valid when "Done" = TRUE)	

Reading a following value

To read a following value from a cam with the "MC_GetCamFollowingValue" Motion Control instruction, follow these steps:

1. Check the requirements indicated above.
2. Specify the cam and the leading value in the corresponding parameters.
3. Start the "MC_GetCamFollowingValue" job with a positive edge at parameter "Execute". When the "Done" parameter shows the value "TRUE", the following value has been read. The following value and the derivatives are output in the "Value", "FirstDerivative" and "SecondDerivative" parameters.

MC_GetCamLeadingValue V4 (S7-1500T)

MC_GetCamLeadingValue: Read out leading value of a cam disc V4 (S7-1500T) (S7-1500T)

Description

With the Motion Control instruction "MC_GetCamLeadingValue", you read the leading value that is defined for a following value from a cam.

Because the same following values can be defined for different leading values, an approximation of the leading value can specified in the "ApproachValue" parameter.

Applies to

- Cam

Requirement

- The technology object has been configured correctly.
- The cam is interpolated.

Override response

- An "MC_GetCamLeadingValue" job is not aborted by any other Motion Control job.
- A new "MC_GetCamLeadingValue" job does not abort any active motion control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_GetCamLeadingValue":

Parameter	Declaration	Data type	Default value	Description	
Cam	INPUT	TO_Cam	-	Cam technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
FollowingValue	INPUT	LREAL	0.0	Following value for which the leading value is read	
ApproachLeadingValue	INPUT	LREAL	0.0	Approximation value for the searched for leading value If the following value is used multiple times in the cam, it can be used to limit the searched leading value.	
Done	OUTPUT	BOOL	FALSE	TRUE	The leading value was read.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.

Parameter	Declaration	Data type	Default value	Description
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"
Value	OUTPUT	LREAL	-	Read leading value (position) (valid when "Done" = TRUE)

Reading a leading value

To read a leading value from a cam with the "MC_GetCamLeadingValue" Motion Control instruction, follow these steps:

1. Check the requirements indicated above.
2. Specify the cam, the following value, and the approximation value for the searched-for leading value in the corresponding parameters.
3. Start the "MC_GetCamLeadingValue" job with a positive edge at parameter "Execute". When the "Done" parameter shows the value "TRUE", the leading value has been determined. The calculation of the leading value can take several cycles. The leading value is output in the "Value" parameter.

MotionIn (S7-1500T)

MC_MotionInVelocity V4 (S7-1500T)

MC_MotionInVelocity: Specify motion setpoints V4 (S7-1500T)

Description

With the Motion Control instruction "MC_MotionInVelocity", you specify cyclically applicable calculated motion setpoints for velocity and acceleration as a basic motion for the axis. No velocity profile is calculated for this, the values are directly active at the technology object. The dynamic limits are not in effect. The motion setting using MotionIn can be superimposed with an "MC_MoveSuperimposed" job (Page 3137).

With the "Velocity" parameter, you specify the desired velocity and with "Acceleration" the desired acceleration. The setpoint acceleration only has an effect when the "MC_MotionInVelocity" job is overridden by another Motion Control job. The acceleration is set to the acceleration setpoint of the "MC_MotionInVelocity" job. If the acceleration setpoint is zero, zero is set when the acceleration is substituted.

Applies to

- Positioning axis
- Speed axis
- Synchronous axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The override response for "MC_MotionInVelocity" jobs is described in section Override response V4: Homing and motion jobs (Page 3251).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MotionInVelocity":

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge The specified setpoints are used as long as the parameter is set to "TRUE".
				FALSE	End of the job with negative edge If the parameter is set from TRUE to FALSE, the setpoints are set to 0.0.
Velocity	INPUT	LREAL	0.0	Velocity setpoint Observe the dynamic limits.	
Acceleration	INPUT	LREAL	0.0	Setpoint acceleration Observe the dynamic limits.	
PositionControlled	INPUT	BOOL	TRUE	TRUE	Position-controlled mode
				FALSE	Controlled running
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

Override response V4: Homing and motion jobs (Page 3251)

MC_MotionInVelocity: Function chart V4 (S7-1500T)

Function chart: Specify motion setpoints

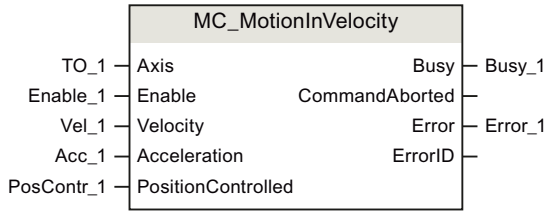

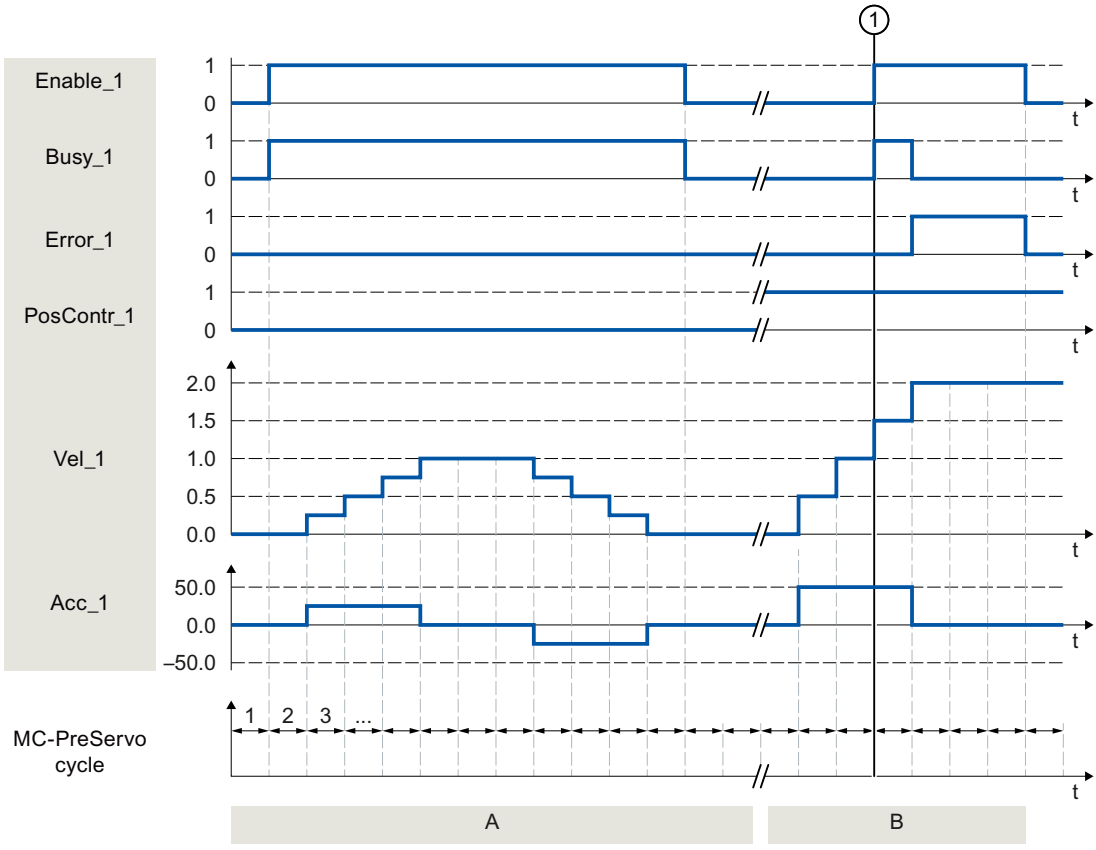

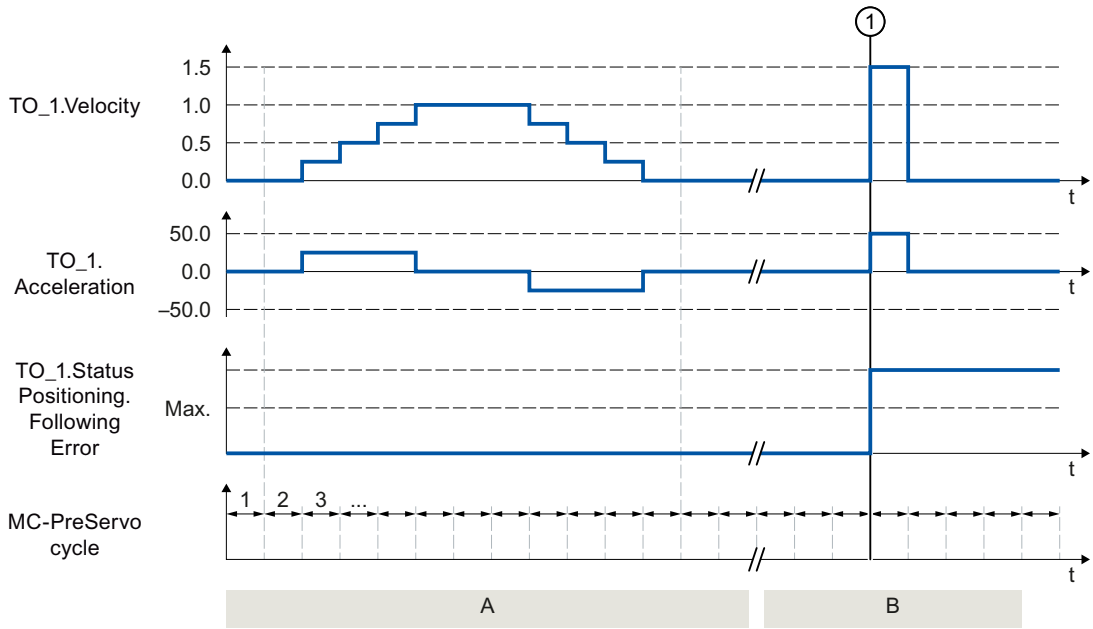

Section A	With "Enable_1 = TRUE" the technology object is specified cyclic in the MC-PreServo-clock velocity "Vel_1" and acceleration "Acc_1". These specifications are applied directly as setpoint velocity "TO_1.Velocity" and setpoint acceleration "TO_1.Acceleration", without hereby calculating a velocity profile. As the position monitoring "PosContr_1" is set to FALSE, no following error "TO_1.StatusPositioning.FollowingError" is determined.
Section B	The velocity and acceleration specifications are not active as long as "Enable_1" is set to FALSE. At time ①, "Enable_1" is set to TRUE. As the position monitoring "PosContr_1" is set to TRUE, a following error "TO_1.StatusPositioning.FollowingError" is determined. The velocity specification "Vel_1" and the acceleration default "Acc_1" cause a setpoint jump which exceeds the maximum permissible following error. With active position lag monitoring, the technology alarm 521 is output, and the technology object is disabled. With deactivated following error monitoring, the setpoint jump is executed with maximum dynamic.

MC_MotionInPosition V4 (S7-1500T)

MC_MotionInPosition: Specify motion setpoints V4 (S7-1500T)

Description

With the Motion Control instruction "MC_MotionInPosition", you specify the cyclic, applicable motion setpoints for position, velocity and acceleration as basic motion for the axis. No velocity profile is calculated for this, the values are directly active at the technology object. The dynamic limits are not in effect. The motion setting using "MotionIn" can be superimposed with an "MC_MoveSuperimposed" job (Page 3137).

With the "Position" parameter, you specify the set position.

With the "Velocity" parameter, you specify the setpoint velocity. The setpoint velocity is used as a feedforward control value when velocity feedforward control is activated. If the

"MotionInPosition" job is overridden by another Motion Control job, the setpoint velocity of the "MC_MotionInPosition" job is used for the velocity.

With the "Acceleration" parameter, you specify the setpoint acceleration. The setpoint acceleration only has an effect when the "MC_MotionInPosition" job is overridden by another Motion Control job. The acceleration is set to the acceleration setpoint of the "MC_MotionInPosition" job. If the acceleration setpoint is zero, zero is set when the acceleration is substituted.

Applies to

- Positioning axis
- Synchronous axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The override response for "MC_MotionInPosition" jobs is described in section Override response V4: Homing and motion jobs (Page 3251).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MotionInPosition":

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning-Axis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge The specified setpoints are used as long as the parameter is set to "TRUE".
				FALSE	End of the job with negative edge If the parameter is set from TRUE to FALSE, the setpoints are set to 0.0. The most recently specified value remains active for the position setpoint.
Position	INPUT	LREAL	0.0	Position setpoint	
Velocity	INPUT	LREAL	0.0	Velocity setpoint Observe the dynamic limits.	
Acceleration	INPUT	LREAL	0.0	Setpoint acceleration Observe the dynamic limits.	
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.

Parameter	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for the parameter "ErrorID"	

See also

Override response V4: Homing and motion jobs (Page 3251)

MC_MotionInPosition: Function chart V4 (S7-1500T)

Function chart: Specify motion setpoints

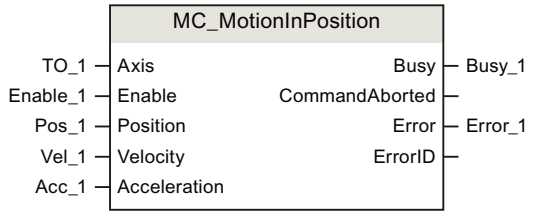

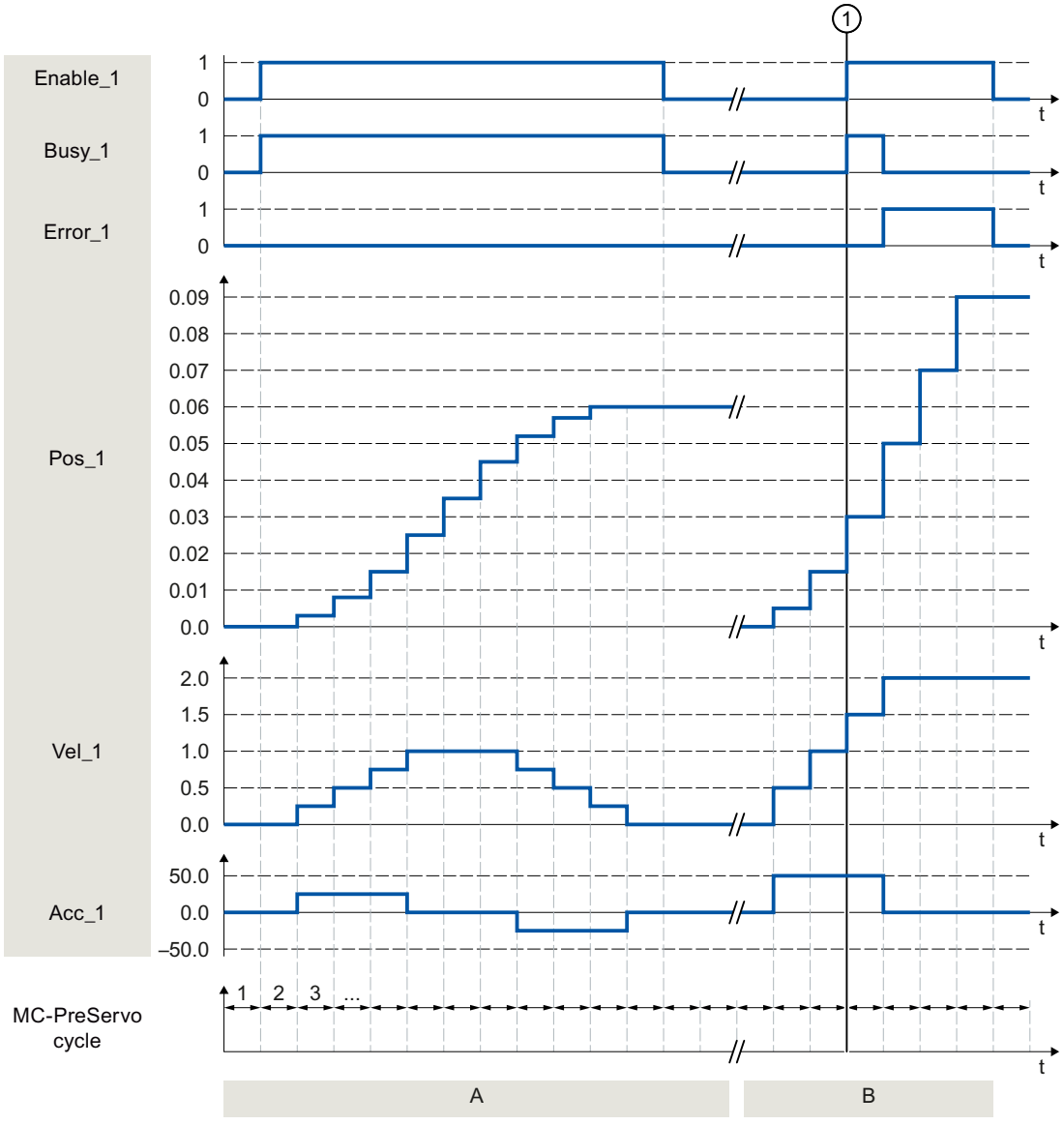

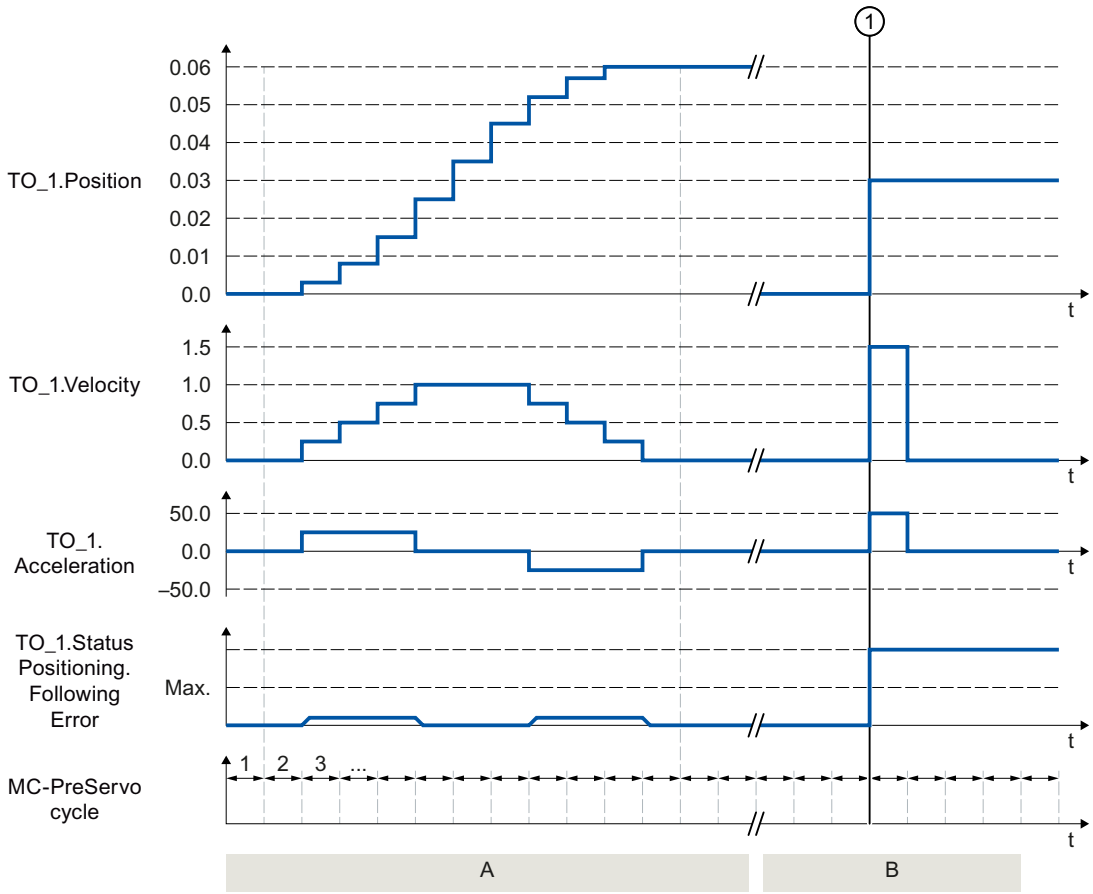

Section A	With "Enable_1 = TRUE" the technology object is specified cyclic in the MC-PreServo-clock position "Pos_1", velocity "Vel_1" and acceleration "Acc_1". These specifications are applied directly as set position "TO_1.Position", setpoint velocity "TO_1.Velocity" and setpoint acceleration "TO_1.Acceleration", without hereby calculating a velocity profile.
Section B	The position, velocity and acceleration specifications are not active as long as "Enable_1" is set to FALSE. At time ①, "Enable_1" is set to TRUE. The default position setting "Pos_1" causes a setpoint jump which exceeds the maximum permissible following error. With active position lag monitoring, the technology alarm 521 is output, and the technology object is disabled. With deactivated following error monitoring, the setpoint jump is executed with maximum dynamic.

Torque data (S7-1500, S7-1500T)

MC_TorqueLimiting V4 (S7-1500, S7-1500T)

MC_TorqueLimiting: Activate and deactivate force/torque limit / fixed stop detection V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_TorqueLimiting", you activate and assign parameters for force/torque limiting and fixed stop detection. Together with a position-controlled motion job, a "Travel to fixed stop" can be realized with the fixed stop detection. In the axis configuration, you can configure whether the force/torque limiting is to relate to the drive side or the load side.

The functions of the Motion Control instruction "MC_TorqueLimiting" can be activated and deactivated before and during a motion job.

Force/torque limiting applies to

- Speed axis
- Positioning axis
- Synchronous axis

Requirements for force/torque limiting

- The technology object and the reference torque of the drive have been configured correctly.
- No errors that prevent enabling are pending for the technology object (the technology object must not be enabled).
- The drive must support force/torque reduction. Only PROFIdrive drives with SIEMENS telegram 10x support force/torque limiting.
- Interconnection in the SINAMICS drive:
  - P1522 to a fixed value of 100 %
  - P1523 to a fixed value of -100 % (e.g. through interconnection to fixed value parameter P2902[i]).

Fixed stop detection applies to

- Positioning axis
- Synchronous axis

Requirement for fixed stop detection

- The fixed stop detection can only be applied to position-controlled axes. For fixed stop detection, the axis must be enabled as position-controlled. Motion jobs must be executed as position-controlled.
- The technology object has been configured correctly.
- When a drive and telegram that support force/torque limiting is used, the reference torque of the drive must be correctly configured for the technology object.
- No errors that prevent enabling are pending for the technology object (the technology object must be enabled).

Override response

- An "MC_TorqueLimiting" job cannot be aborted by any other Motion Control job.
- A new "MC_TorqueLimiting" job does not abort any active Motion Control jobs.
- If the high and low torque limiting is active via the "MC_TorqueRange" job, the "MC_TorqueLimiting" job is rejected with an error message and vice versa. The functions do not override one another.

Parameters

The following table shows the parameters of Motion Control instruction "MC_TorqueLimiting":

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	Activate function corresponding to input parameter "Mode"
Limit	INPUT	LREAL	-1.0	Value of force/torque limiting (in the configured unit of measurement) If drive and telegram do not support force/torque limiting, the specified value is irrelevant.	
				≥ 0.0	Use the value specified in the parameter
				< 0.0	Use the value configured in the "Torque limiting" configuration window. Tag Torque Limit: <TO>.TorqueLimiting.LimitDefaults.Torque Tag Force Limit: <TO>.TorqueLimiting.LimitDefaults.Force
Mode	INPUT	DINT	0	0	Force/torque limiting
				1	Fixed stop detection If drive and telegram support force/torque limiting, this is applied.

Parameter	Declaration	Data type	Default value	Description	
InClamping	OUTPUT	BOOL	FALSE	TRUE	"Mode" = 1: The drive is kept at the fixed stop (clamping), the axis position is within the positioning tolerance.
InLimitation	OUTPUT	BOOL	FALSE	TRUE	"Mode" = 0 and 1: Drive is operating at the force/torque limit.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Changes to input parameters "Limit" and "Mode" are also transferred at the cyclic call of the Motion Control instruction when "Enable" = TRUE.

MC_TorqueLimiting: Function chart V4 (S7-1500, S7-1500T)

Function chart: Stopping an axis when the torque limit is reached

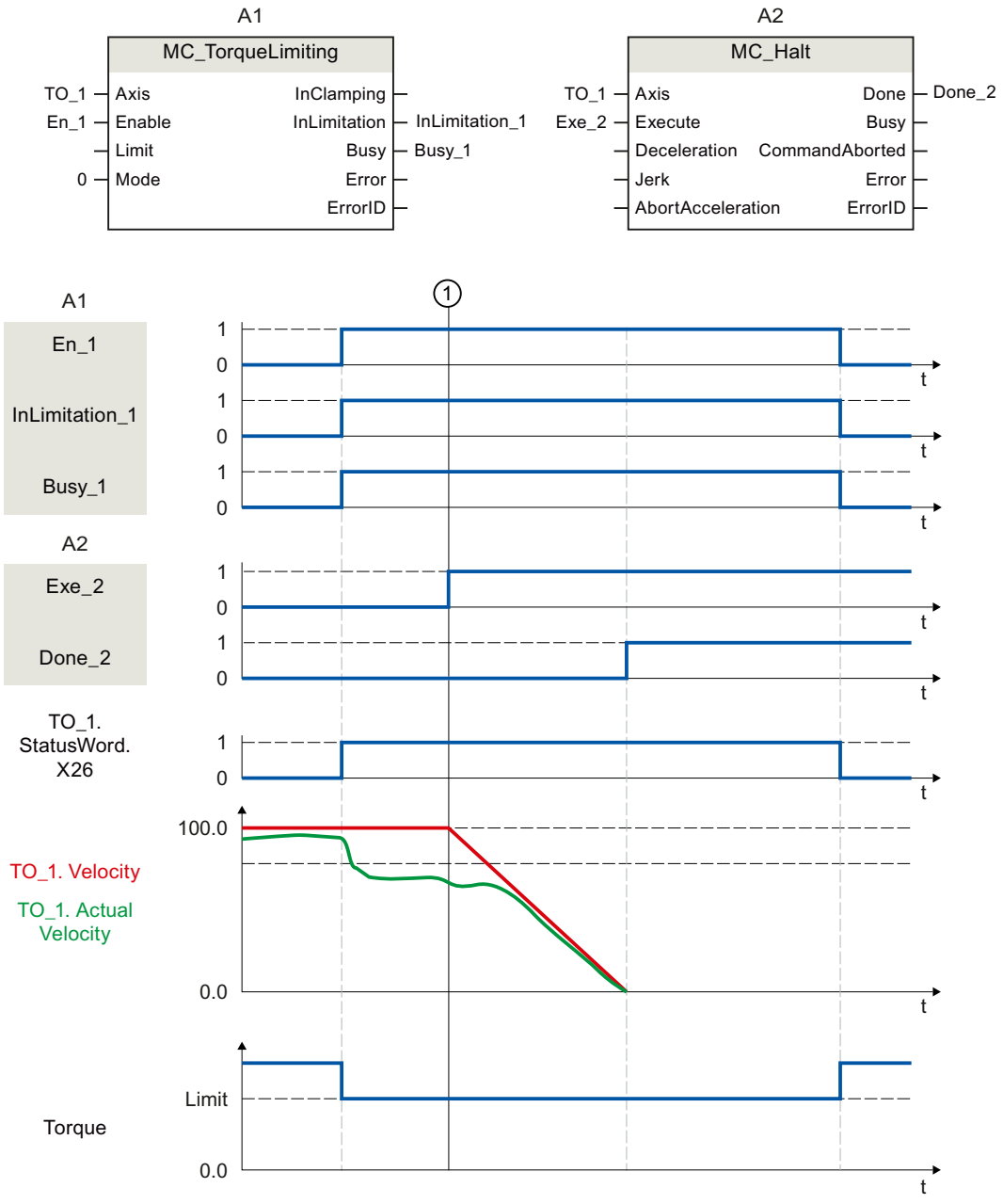

At time ①, an "MC_Halt" job (A2) is executed on an axis with active torque limiting "MC_TorqueLimiting" (A1). Torque limiting is still active ("MC_TorqueLimiting.Enable" = TRUE) and any accumulated following error is retained and will be reduced with the time. When the actual velocity is "0.0" and the minimum dwell time in the standstill window has elapsed, the tag "MC_Halt.Done" shows = TRUE. When position monitoring is activated, reaching of the target position is also monitored.

Function chart: Torque limiting with fixed stop detection (mode = 1)

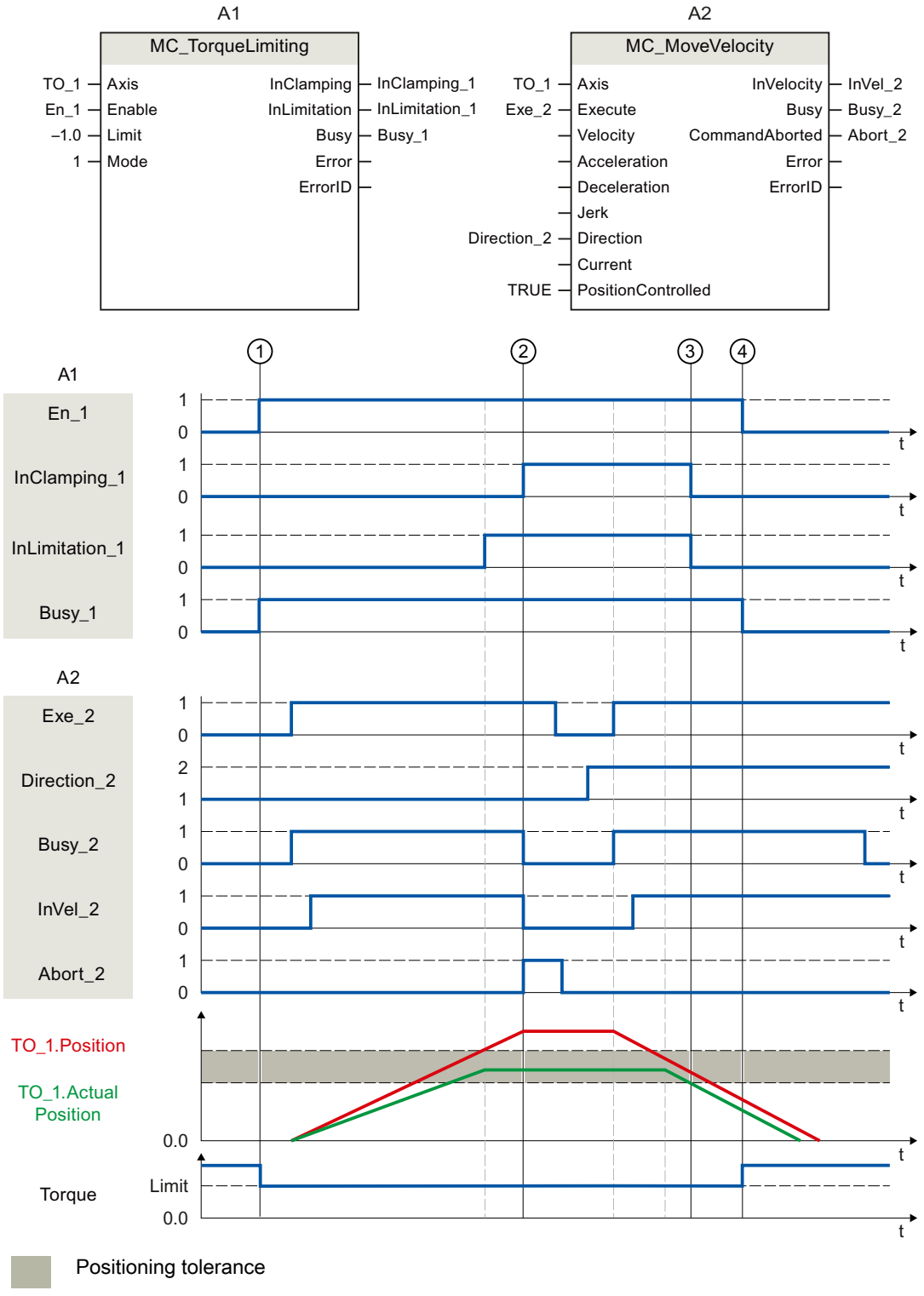

At time ①, a "MC_TorqueLimiting" job (A1) is initiated via "En_1". An "MC_MoveVelocity" job (A2) is executed on the axis with active torque limiting. The torque limiting is still active

"MC_TorqueLimiting.Enable" = TRUE. When the following error limit ② is reached, the "MC_MoveVelocity" job is aborted with "Abort" = TRUE. The drive is kept at the fixed stop (clamping). The actual position of the axis is within the positioning tolerance. An "MC_MoveVelocity" job is once again called via the two tags "Execute" = TRUE and "Direction_2" = TRUE and the axis moves with constant velocity in the opposite direction. The clamping is hereby reduced when the positioning tolerance ③ is exited. At time ④, the torque limiting is cancelled.

MC_TorqueAdditive V4 (S7-1500, S7-1500T)

MC_TorqueAdditive: Specify additive torque V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_TorqueAdditive", you specify an additive torque for the drive to which the technology object is assigned. The torque data are transferred via the telegram 750.

With the "Value" parameter, you specify the additive setpoint torque. The specification of the additive torque setpoint is overriding. An additional torque may be positive or negative. If you invert the setpoint for the technology object, the value for the additive torque is also inverted and transferred inverted to the drive.

Applies to

- Positioning axis
- Speed axis
- Synchronous axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.
- The drive is connected via PROFIdrive telegram.
- Telegram 750 is configured.
  Telegram 750 is available for SINAMICS drives V4.9 and higher.

Override response

- An "MC_TorqueAdditive" job is not aborted by any other Motion Control job.
- A new "MC_TorqueAdditive" job does not abort any active Motion Control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_TorqueAdditive":

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The specified setpoint is used as long as the parameter is set to TRUE.
				FALSE	The additive torque transferred to the drive is zero.
Value	INPUT	LREAL	0.0	Additive setpoint torque Permissible values: -1.0E12 to 1.0E12	
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

MC_TorqueAdditive: Function chart V4 (S7-1500, S7-1500T)

Function chart: Activate/deactivate additive setpoint torque

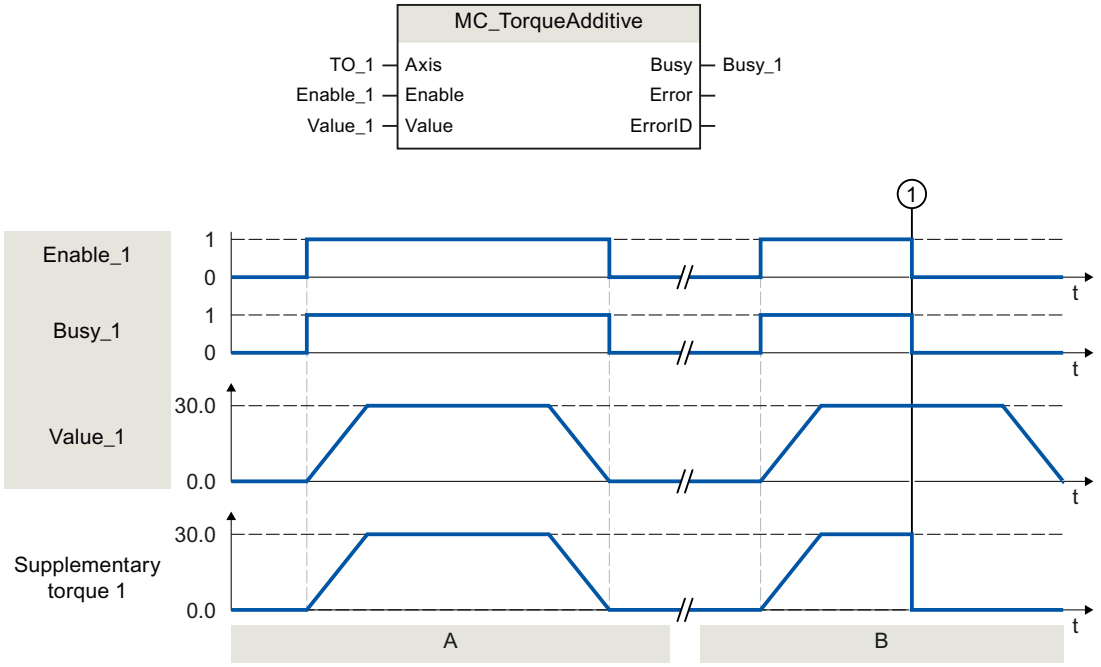

Section A	With "Enable_1 = TRUE", an additive setpoint torque "Value_1" is specified for the drive assigned to the technology object. This setting is transferred to the drive parameter "p1511 - Supplementary torque 1" using telegram 750.
Section B	With "Enable_1 = TRUE", an additive setpoint torque "Value_1" is specified for the drive assigned to the technology object. This setting is transferred to the drive parameter "p1511 - Supplementary torque 1" using telegram 750. The additive setpoint torque is first built up. At time ①, "Enable_1" is already set to FALSE, before the additive setpoint torque is reduced again. The reduction of the torque setpoint is transmitted directly to the drive.

MC_TorqueRange V4 (S7-1500, S7-1500T)

MC_TorqueRange: Set high and low torque limits V4 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_TorqueRange", you can specify an upper and lower torque limit to the drive assigned to the technology object. The torque data are transferred via the telegram 750.

With the "UpperLimit" parameter you specify the upper torque limit and with "LowerLimit" the lower torque limit. The specification of the torque limits has smoothing effect on the movements. If you invert the setpoints for the technology object, the values for the high and low torque limit are also inverted and transferred inverted to the drive.

If the high and low torque limits are active, the following monitors and limits are disabled by default:

- Following error monitoring
- Time limitations for positioning and standstill monitoring

Monitoring remains in effect if you have selected the option "Leave position-related monitoring enabled" under "Technology object > Configuration > Extended parameters > Limits > Torque limit".

Applies to

- Positioning axis
- Speed axis
- Synchronous axis

Requirement

- The technology object has been configured correctly.
- The drive is connected via PROFIdrive telegram.
- Telegram 750 is configured.
  Telegram 750 is available for SINAMICS drives V4.9 and higher.

Override response

- An "MC_TorqueRange" job is not aborted by any other Motion Control job.
- A new "MC_TorqueRange" job does not abort any active Motion Control jobs.
- If the torque limiting is active via the "MC_TorqueLimiting" job, the "MC_TorqueRange" job is rejected with an error message and vice versa. The functions do not override one another.

Parameters

The following table shows the parameters of Motion Control instruction "MC_TorqueRange":

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The specified values are used as long as the TRUE parameter is set.
				FALSE	No values for the high and low torque limit are transferred to the drive.
UpperLimit	INPUT	LREAL	1.0 E12	Upper torque limit (in the configured unit)	
				Permitted range of values: -1.0 E12 to 1.0 E12 The value of the parameter "UpperLimit" must not be greater than the value of the parameter "LowerLimit".	
LowerLimit	INPUT	LREAL	-1.0 E12	Lower torque limit (in the configured unit)	
				Permitted range of values: -1.0 E12 to 1.0 E12 The value of the parameter "LowerLimit" must not be less than the value of the parameter "UpperLimit".	
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

MC_TorqueRange: Function chart V4 (S7-1500, S7-1500T)

Function chart: Set high and low torque limits

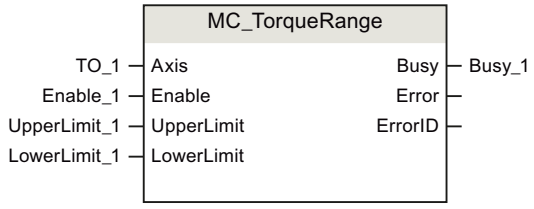

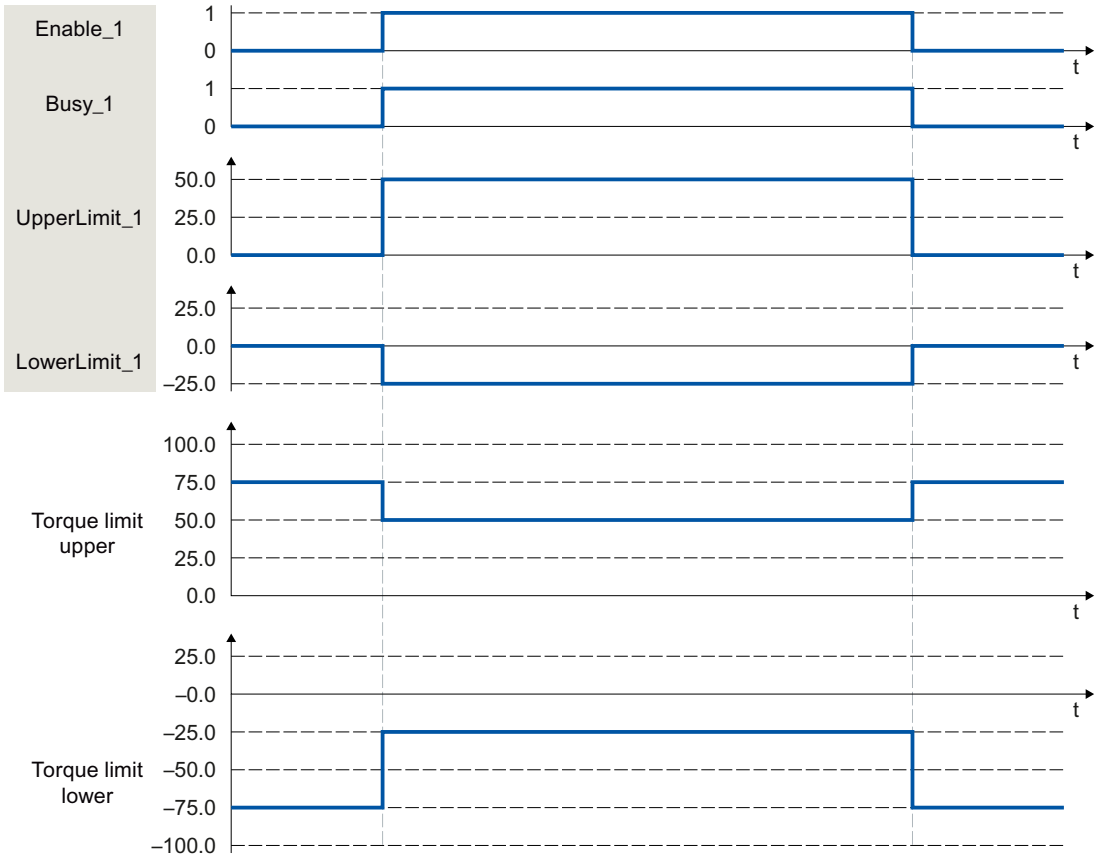

With "Enable_1 = TRUE" an upper torque limit "UpperLimit_1" and a lower torque limit "LowerLimit_1" are specified to the drive assigned to the technology object. These settings are transferred directly to the drive parameters "p1522 - Torque limit upper" and "p1523 - Torque limit lower" using telegram 750. If "Enable_1" is reset to FALSE, the upper and lower torque limits are no longer active.

Kinematics motions (S7-1500T)

MC_GroupInterrupt V4 (S7-1500T)

MC_GroupInterrupt: Interrupt motion execution V4 (S7-1500T)

Description

With the Motion Control instruction "MC_GroupInterrupt", you interrupt the execution of the motion on a kinematics technology object. The interrupted kinematics motion can be resumed with a "MC_GroupContinue" job.

If the kinematics is already at a standstill, the execution of the motion is also interrupted for subsequent motion jobs. New motion jobs are then arranged as pending jobs in the job sequence.

With the "Mode" parameter, you specify the dynamic behavior of the holding.

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.

Override response

The override response for "MC_GroupInterrupt" jobs is described in section Override response V4: Kinematics motion commands (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_GroupInterrupt":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Mode	INPUT	DINT	0	Mode for dynamic behavior	
				0	Stop with the dynamics of the motion job to be interrupted
				1	Stop with maximum dynamics of kinematics motion
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.

Parameter	Declaration	Data type	Default value	Description	
Active	OUTPUT	BOOL	FALSE	TRUE	The setpoints are calculated.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

Override response V4: Kinematics motion commands (Page 3253)

MC_GroupContinue V4 (S7-1500T)

MC_GroupContinue: Continue motion execution V4 (S7-1500T)

Description

With the Motion Control instruction "MC_GroupContinue", you continue a kinematics motion that was previously interrupted with a "MC_GroupInterrupt" job. The kinematics motion can also be continued if the kinematics has not yet come to a standstill following the "MC_GroupInterrupt" job.

The "MC_GroupContinue" job has only an effect if the technology object is in "Interrupted" state.

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.

Override response

- An "MC_GroupContinue" job is not aborted by any other Motion Control job.
- A new "MC_GroupContinue" job aborts a current "MC_GroupInterrupt" job.
- The override response for "MC_GroupContinue" jobs is described in section Override response V4: Kinematics motion commands (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_GroupContinue":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

Override response V4: Kinematics motion commands (Page 3253)

MC_GroupContinue: Function chart V4 (S7-1500T)

Function chart: Continue motion execution

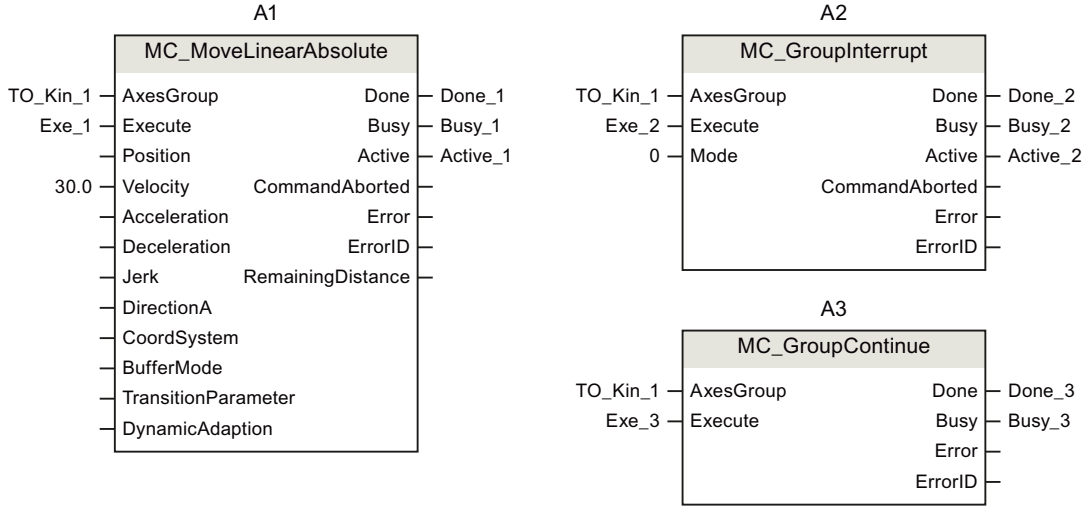

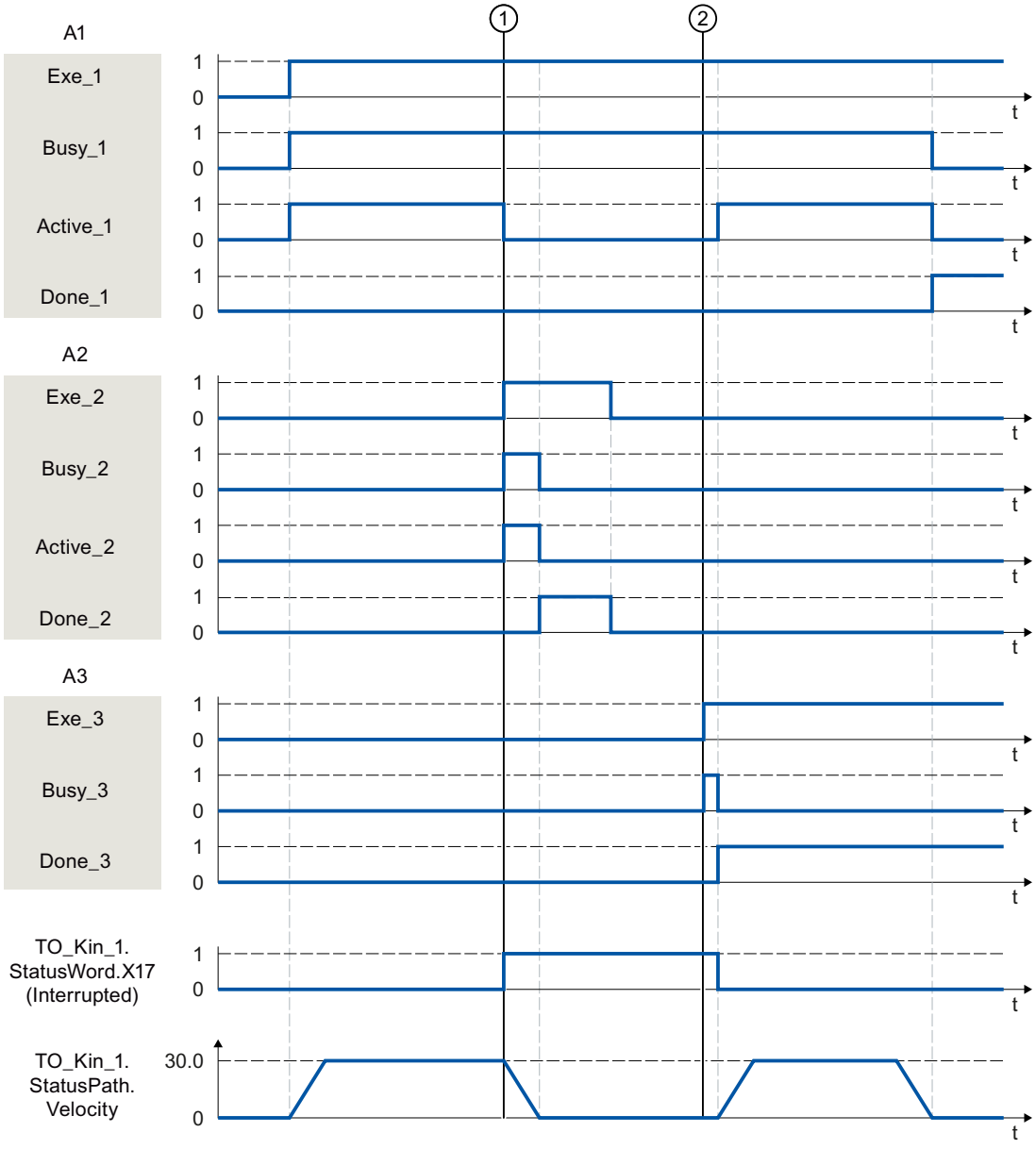

The kinematics is moved with an "MC_MoveLinearAbsolute" job (A1).

At time ①, the "MC_MoveLinearAbsolute" job is interrupted by an "MC_GroupInterrupt" job (A2). The kinematics is in "Interrupted" state. With "Mode = 0" the motion is stopped with the dynamics of the "MC_MoveLinearAbsolute" job. The completion of the "MC_GroupInterrupt" job is reported via "Done_2".

At time ②, the "MC_MoveLinearAbsolute" job is continued by an "MC_GroupContinue" job (A3).

MC_GroupStop V4 (S7-1500T)

MC_GroupStop: Stop motion V4 (S7-1500T)

Description

With the Motion Control instruction "MC_GroupStop", you stop and abort an active motion on the kinematics technology object. If the motion was already interrupted with an "MC_GroupInterrupt", this is aborted. All still pending jobs in the job sequence are also aborted by a "MC_GroupStop" job. As long as the "Execute" parameter is set to TRUE, kinematics jobs are rejected ("ErrorID" = 16#80CD).

With the "Mode" parameter, you specify the dynamic behavior of the stop motion.

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.

Override response

The override response for "MC_GroupStop" jobs is described in section Override response V4: Kinematics motion commands (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_GroupStop":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Mode	INPUT	DINT	0	Mode for dynamic behavior	
				0	Stop with the dynamics of the motion job to be interrupted
				1	Stop with maximum dynamics of kinematics motion When dynamic adaptation is activated, the dynamics of the motion can also be reduced so that the dynamic limits of the axes are not exceeded.
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.

Parameter	Declaration	Data type	Default value	Description	
Active	OUTPUT	BOOL	FALSE	TRUE	The setpoints are calculated.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

Override response V4: Kinematics motion commands (Page 3253)

MC_GroupStop: Function chart V4 (S7-1500T)

Function chart: Stop motion

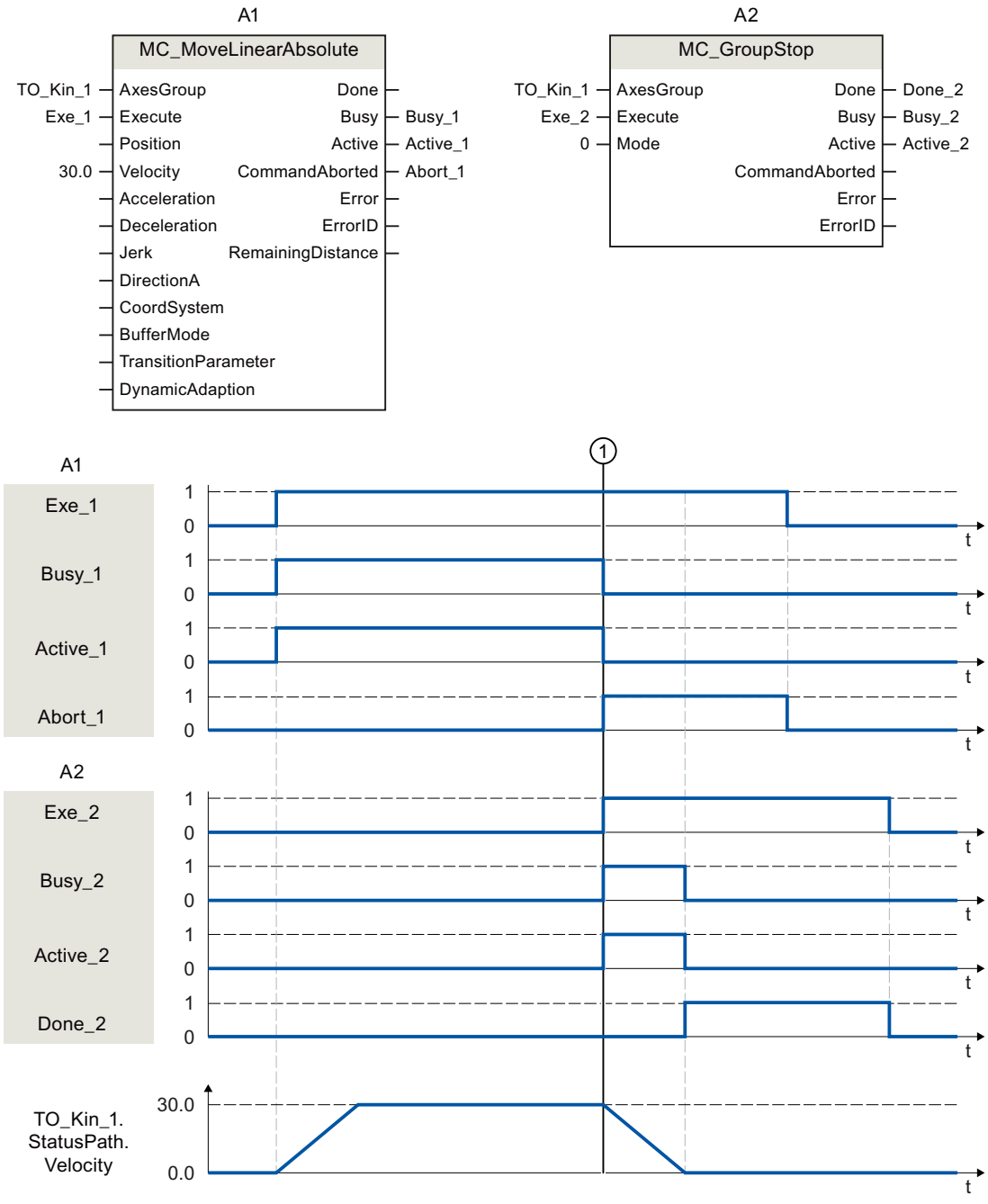

The kinematics is moved with an "MC_MoveLinearAbsolute" job (A1).

At time ①, the "MC_MoveLinearAbsolute" job is aborted by an "MC_GroupStop" job (A2). The job abort is signaled via "Abort_1". With "Mode = 0" the motion is stopped with the dynamics

of the "MC_MoveLinearAbsolute" job. The completion of the "MC_GroupStop" job is reported via "Done_2".

MC_MoveLinearAbsolute V4 (S7-1500T)

MC_MoveLinearAbsolute: Position kinematics with linear motion V4 (S7-1500T)

Description

With the Motion Control instruction "MC_MoveLinearAbsolute", you move a kinematics with a circular motion to an absolute position. Cartesian orientation is also used absolute.

Dynamic behavior during movement is defined with the parameters "Velocity", "Acceleration", "Deceleration" and "Jerk". For the dynamics of the orientation motion, the default values are configured in "Technology object > Configuration > Extended parameters > Dynamic defaults".

- <TO>.DynamicDefaults.Orientation.Velocity
- <TO>.DynamicDefaults.Orientation.Acceleration
- <TO>.DynamicDefaults.Orientation.Deceleration
- <TO>.DynamicDefaults.Orientation.Jerk

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- The interconnected axes are homed.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.

Override response

With the "TransitionParameter[1]" parameter, you determine the distance from the target point of the previous motion job at which a new motion command is superimposed.

The override response for "MC_MoveLinearAbsolute" jobs is described in section Override response V4: Kinematics motion commands (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveLinearAbsolute":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Position	INPUT	ARRAY [1..4] OF LREAL	-	Absolute target coordinates in the specified reference system	
Position[1]	INPUT	LREAL	0.0	x coordinate	
Position[2]	INPUT	LREAL	0.0	y coordinate	
Position[3]	INPUT	LREAL	0.0	z coordinate	
Position[4]	INPUT	LREAL	0.0	A coordinate	
Velocity	INPUT	LREAL	-1.0	Velocity	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	The specified value is used.
				= 0.0	No jerk limitation
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Jerk)

Parameter	Declaration	Data type	Default value	Description	
DirectionA	INPUT	DINT	3	Motion direction of the Cartesian orientation	
				1	Positive direction
				2	Negative direction
				3	Shortest distance
CoordSystem	INPUT	DINT	0	Reference system of the specified target position and target orientation	
				0	World coordinate system (WCS)
				1	Object coordinate system 1 (OCS[1])
				2	Object coordinate system 2 (OCS[2])
				3	Object coordinate system 3 (OCS[3])
BufferMode	INPUT	DINT	1	Motion transition	
				1	Append motion The active motion sequence is completed and the kinematics come to a standstill. The movement of this job is then executed.
				2	Smooth with the lower velocity When the blending distance is reached, the current motion is blended with the motion of this job. The lower velocity of both jobs is then used.
				5	Smooth with the higher velocity When the blending distance is reached, the current motion is blended with the motion of this job. The higher velocity of both jobs is then used.
TransitionParameter	INPUT	ARRAY [1..5] OF LREAL		Transition parameter	
TransitionParameter[1]	INPUT	LREAL	-1.0	Rounding clearance	
				≥ 0.0	The specified value is used.
				< 0.0	The maximum possible rounding clearance is used.
TransitionParameter[2]	INPUT	LREAL	-	Reserved	
TransitionParameter[3]	INPUT	LREAL	-	Reserved	
TransitionParameter[4]	INPUT	LREAL	-	Reserved	
TransitionParameter[5]	INPUT	LREAL	-	Reserved	

Parameter	Declaration	Data type	Default value	Description	
DynamicAdaption	INPUT	DINT	-1	Dynamic adaptation	
				< 0	The configured setting in "Technology object > Configuration > Extended parameters > Dynamic" is used. (<TO>.DynamicDefaults.DynamicAdaption)
				0	Without dynamic adaption
				1	Dynamic adaptation with segmentation of the path
				2	Dynamic adaptation without segmentation of the path
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
Active	OUTPUT	BOOL	FALSE	TRUE	The setpoints are calculated.
				FALSE	When "Busy" = TRUE: Job is waiting. (Typical: The preceding job is still active.)
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	
RemainingDistance	OUTPUT	LREAL	-1.0	Distance-to-go of the current job	

See also

Override response V4: Kinematics motion commands (Page 3253)

MC_MoveLinearAbsolute: Function chart V4 (S7-1500T)

Function chart: Position kinematics with linear path motion

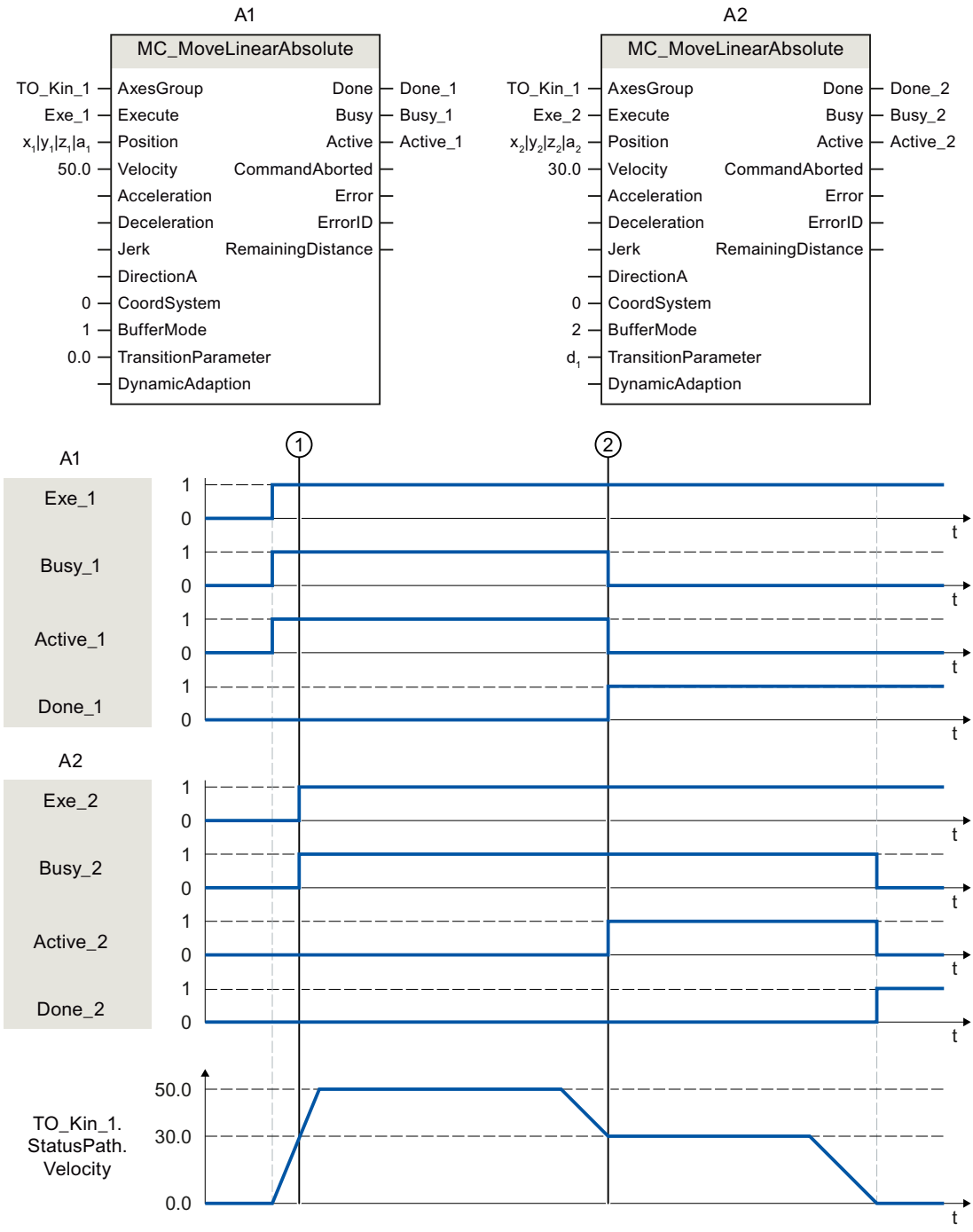

The kinematics is moved with an "MC_MoveLinearAbsolute" job (A1).

At time ① an additional "MC_MoveLinearAbsolute" job (A2) is initiated. As the "MC_MoveLinearAbsolute" jobs have no override effect on one another, the job A2 is arranged in the job sequence.

At the point in time ②, completion of the A1 job is signaled via "Done_1" and the job A2 is started. As "BufferMode = 2" is set for job A2, the motion transition is smoothed with the lower velocity of both jobs. As soon as the target position is reached, the completion of the A2 job is signaled via "Done_2".

MC_MoveLinearRelative V4 (S7-1500T)

MC_MoveLinearRelative: Relative positioning of kinematics with linear motion V4 (S7-1500T)

Description

With the Motion Control instruction "MC_MoveLinearRelative", you can move a kinematics with a linear path relative to the position which was present at the start of the job processing. Cartesian orientation is also used relative.

Dynamic behavior during movement is defined with the parameters "Velocity", "Acceleration", "Deceleration" and "Jerk". For the dynamics of the orientation motion, the default values are configured in "Technology object > Configuration > Extended parameters > Dynamic defaults".

- <TO>.DynamicDefaults.Orientation.Velocity
- <TO>.DynamicDefaults.Orientation.Acceleration
- <TO>.DynamicDefaults.Orientation.Deceleration
- <TO>.DynamicDefaults.Orientation.Jerk

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.

Override response

With the "TransitionParameter[1]" parameter, you determine the distance from the target point of the previous motion job at which a new motion command is superimposed.

The override response for "MC_MoveLinearRelative" jobs is described in section Override response V4: Kinematics motion commands (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveLinearRelative":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Distance	INPUT	ARRAY [1..4] OF LREAL	-	Relative target coordinates in the specified reference system	
Distance[1]	INPUT	LREAL	0.0	x coordinate	
Distance[2]	INPUT	LREAL	0.0	y coordinate	
Distance[3]	INPUT	LREAL	0.0	z coordinate	
Distance[4]	INPUT	LREAL	0.0	A coordinate	
Velocity	INPUT	LREAL	-1.0	Velocity	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	The specified value is used.
				= 0.0	No jerk limitation
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Jerk)

Parameter	Declaration	Data type	Default value	Description	
CoordSystem	INPUT	DINT	0	Reference system of the specified target position and target orientation	
				0	World coordinate system (WCS)
				1	Object coordinate system 1 (OCS[1])
				2	Object coordinate system 2 (OCS[2])
				3	Object coordinate system 3 (OCS[3])
BufferMode	INPUT	DINT	1	Motion transition	
				1	Append motion The active motion sequence is completed and the kinematics come to a standstill. The movement of this job is then executed.
				2	Smooth with the lower velocity When the blending distance is reached, the current motion is blended with the motion of this job. The lower velocity of both jobs is then used.
				5	Smooth with the higher velocity When the blending distance is reached, the current motion is blended with the motion of this job. The higher velocity of both jobs is then used.
TransitionParameter	INPUT	ARRAY [1..5] OF LREAL	-	Transition parameter	
TransitionParameter[1]	INPUT	LREAL	-1.0	Rounding clearance	
				≥ 0.0	The specified value is used.
				< 0.0	The maximum possible rounding clearance is used.
TransitionParameter[2]	INPUT	LREAL	-	Reserved	
TransitionParameter[3]	INPUT	LREAL	-	Reserved	
TransitionParameter[4]	INPUT	LREAL	-	Reserved	
TransitionParameter[5]	INPUT	LREAL	-	Reserved	
DynamicAdaption	INPUT	DINT	-1.0	Dynamic adaptation	
				< 0	The configured setting in "Technology object > Configuration > Extended parameters > Dynamic" is used. (<TO>.DynamicDefaults.DynamicAdaption)
				0	Without dynamic adaption
				1	Dynamic adaptation with segmentation of the path
				2	Dynamic adaptation without segmentation of the path
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.

Parameter	Declaration	Data type	Default value	Description	
Active	OUTPUT	BOOL	FALSE	TRUE	The setpoints are calculated.
				FALSE	When "Busy" = TRUE: Job is waiting. (Typical: The preceding job is still active.)
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	
RemainingDistance	OUTPUT	LREAL	-1.0	Distance-to-go of the current job	

See also

Override response V4: Kinematics motion commands (Page 3253)

MC_MoveLinearRelative: Function chart V4 (S7-1500T)

Function chart: Relative positioning of kinematics with linear path motion

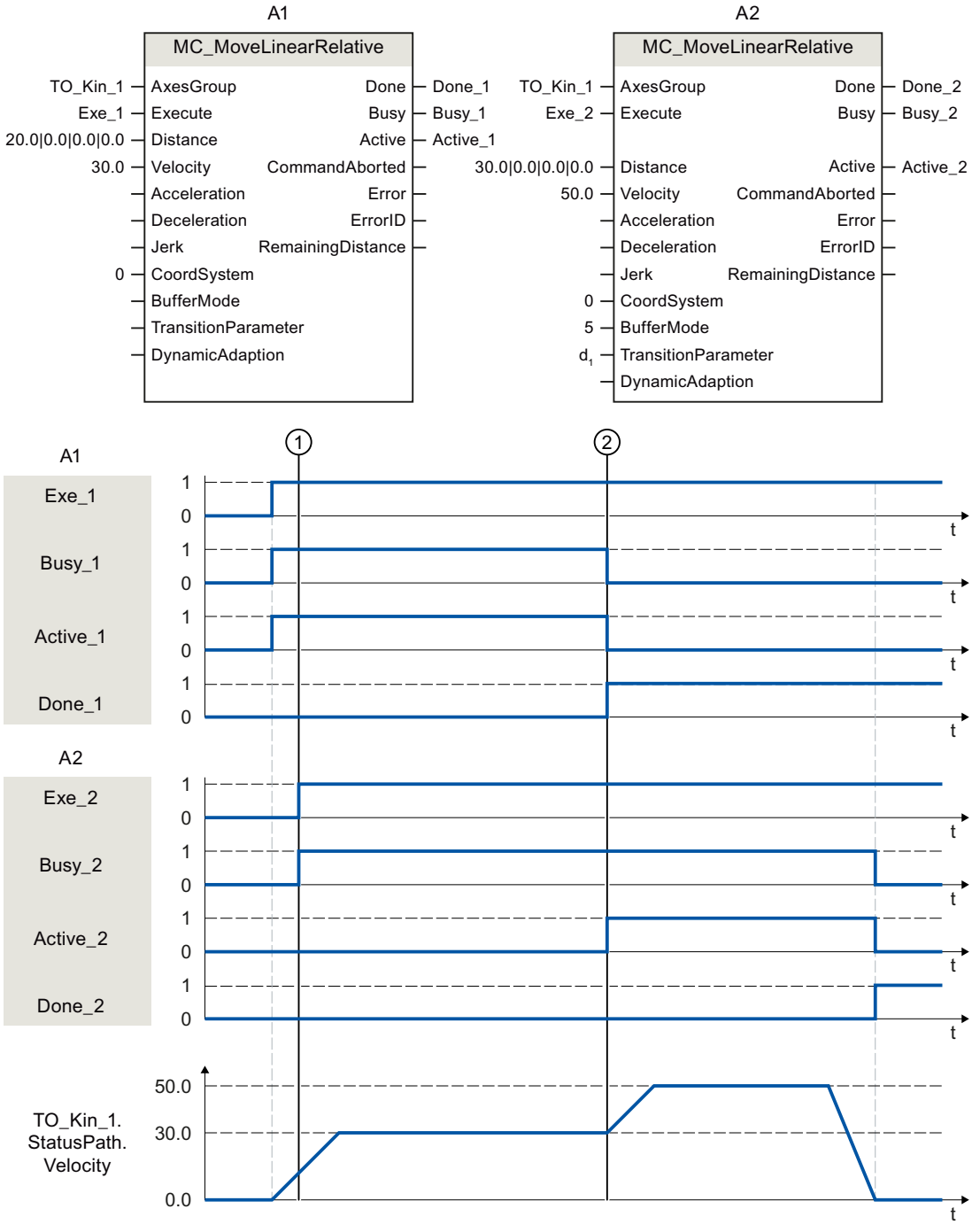

The kinematics is moved with an "MC_MoveLinearRelative" job (A1).

At time ① an additional "MC_MoveLinearRelative" job (A2) is initiated. As the "MC_MoveLinearRelative" jobs have no override effect on one another, the job A2 is arranged in the job sequence.

At the point in time ②, completion of the A1 job is signaled via "Done_1" and the job A2 is started. As "BufferMode = 5" is set for job A2, the motion transition is smoothed with the higher velocity of both jobs. As soon as the target position is reached, the completion of the A2 job is signaled via "Done_2".

MC_MoveCircularAbsolute V4 (S7-1500T)

MC_MoveCircularAbsolute: Position kinematics with circular path motion V4 (S7-1500T)

Description

With the Motion Control instruction "MC_MoveCircularAbsolute", you can move a kinematics with a circular motion to an absolute position. Cartesian orientation is also used absolute.

With the "CircMode" parameter, you specify the definition of the circular path:

- Via an intermediate point and the end point
  With the "AuxPoint" parameter, you specify an intermediate point on the circular path, via which the end point specified in the "EndPoint" parameter is to be approached. The circular path is calculated via the starting, intermediate and end points. Only circular paths less than 360° can be traversed here.
- Via the circle center and angle in a main plane
  With the "AuxPoint" parameter, you define the center point of the circle. The end point of the circular path is calculated via the angle specified in the parameter "Arc". With the "PathChoice" parameter, you specify whether the circular path is to be traversed in positive or negative direction of rotation. With the "CirclePlane" parameter, you specify the main plane in which the circular path is to be traversed.
- Via the circle radius and the end point in a main plane
  With the "EndPoint" parameter, you specify the end point and with "Radius" the radius of the circular path. Depending on the radius, up to four possible circular paths can occur in the plane defined with the "CirclePlane" parameter. With the "PathChoice" parameter, you specify which of these circular paths is to be moved.

Dynamic behavior during Kinematics motion is defined with the parameters "Velocity", "Acceleration", "Deceleration" and "Jerk". For the dynamics of the orientation motion, the default values are configured in "Technology object > Configuration > Extended parameters > Dynamic defaults".

- <TO>.DynamicDefaults.Orientation.Velocity
- <TO>.DynamicDefaults.Orientation.Acceleration
- <TO>.DynamicDefaults.Orientation.Deceleration
- <TO>.DynamicDefaults.Orientation.Jerk

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- The interconnected axes are homed.
- There is no single axis job (e.g. "MC_MoveVelocity") on the interconnected axes.

Override response

With the "TransitionParameter[1]" parameter, you determine the distance from the target point of the previous motion job at which a new motion command is superimposed.

The override response for "MC_MoveCircularAbsolute" jobs is described in section Override response V4: Kinematics motion commands (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveCircularAbsolute":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
CircMode	INPUT	DINT	0	Definition of circular path	
				0	The position vector specified at the "AuxPoint" parameter defines a point on the circular path.
				1	The position specified at the parameter "AuxPoint" defines the circle center.
				2	The "Radius" and "EndPoint" parameters define the circle segment.
AuxPoint	INPUT	ARRAY [1..3] OF LREAL	-	Circular path auxiliary point • When "CircMode" = 0: Point on the circular path • When "CircMode" = 1: Center point of the circular path • When "CircMode" = 2: Not relevant	
AuxPoint[1]	INPUT	LREAL	0.0	x coordinate	
AuxPoint[2]	INPUT	LREAL	0.0	y coordinate	
AuxPoint[3]	INPUT	LREAL	0.0	z coordinate	
EndPoint	INPUT	ARRAY [1..4] OF LREAL	-	Target position in the specified reference system	
				When "CircMode" = 1: Only EndPoint[4] relevant (orientation axis)	

Parameter	Declaration	Data type	Default value	Description	
EndPoint[1]	INPUT	LREAL	0.0	x coordinate	
EndPoint[2]	INPUT	LREAL	0.0	y coordinate	
EndPoint[3]	INPUT	LREAL	0.0	z coordinate	
EndPoint[4]	INPUT	LREAL	0.0	A coordinate	
PathChoice	INPUT	DINT	0	Orientation of the circular path	
				When "CircMode" = 0: Not relevant	
				When "CircMode" = 1:	
				0	Positive direction of rotation
				1	Negative direction of rotation
				When "CircMode" = 2:	
				0	Shorter positive circle segment
				1	Shorter negative circle segment
				2	Longer positive circle segment
				3	Longer negative circle segment
CirclePlane	INPUT	DINT	0	Main plane of the circular path	
				When "CircMode" = 0: Not relevant	
				When "CircMode" = 1 and 2:	
				0	x-z plane
				1	y-z plane
				2	x-y plane
Radius	INPUT	LREAL	0.0	When "CircMode" = 2: Radius of the circular movement	
Arc	INPUT	LREAL	0.0	When "CircMode" = 1: Angle of the circular movement	
Velocity	INPUT	LREAL	-1.0	Velocity	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Acceleration)

Parameter	Declaration	Data type	Default value	Description	
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	The specified value is used.
				= 0.0	No jerk limitation
				< 0.0	The value configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Jerk)
DirectionA	INPUT	DINT	3	Motion direction of the Cartesian orientation	
				1	Positive direction
				2	Negative direction
				3	Shortest distance
CoordSystem	INPUT	DINT	0	Reference system	
				0	World coordinate system (WCS)
				1	Object coordinate system 1 (OCS[1])
				2	Object coordinate system 2 (OCS[2])
				3	Object coordinate system 3 (OCS[3])
BufferMode	INPUT	DINT	1	Motion transition	
				1	Append motion The active motion sequence is completed and the kinematics come to a standstill. The movement of this job is then executed.
				2	Smooth with the lower velocity When the blending distance is reached, the current motion is blended with the motion of this job. The lower velocity of both jobs is then used.
				5	Smooth with the higher velocity When the blending distance is reached, the current motion is blended with the motion of this job. The higher velocity of both jobs is then used.
TransitionParameter	INPUT	ARRAY [1..5] OF LREAL		Transition parameter	

Parameter	Declaration	Data type	Default value	Description	
TransitionParameter[1]	INPUT	LREAL	-1.0	Rounding clearance	
				≥ 0.0	The specified value is used.
				< 0.0	The maximum possible rounding clearance is used.
TransitionParameter[2]	INPUT	LREAL	-	Reserved	
TransitionParameter[3]	INPUT	LREAL	-	Reserved	
TransitionParameter[4]	INPUT	LREAL	-	Reserved	
TransitionParameter[5]	INPUT	LREAL	-	Reserved	
DynamicAdaption	INPUT	DINT	-1.0	Dynamic adaptation	
				< 0	The configured setting in "Technology object > Configuration > Extended parameters > Dynamic" is used. (<TO>.DynamicDefaults.DynamicAdaption)
				0	Without dynamic adaption
				1	Dynamic adaptation with segmentation of the path
				2	Dynamic adaptation without segmentation of the path
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
Active	OUTPUT	BOOL	FALSE	TRUE	The setpoints are calculated.
				FALSE	When "Busy" = TRUE: Job is waiting. (Typical: The preceding job is still active.)
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	
RemainingDistance	OUTPUT	LREAL	-1.0	Distance-to-go of the current job	

See also

Override response V4: Kinematics motion commands (Page 3253)

MC_MoveCircularAbsolute: Function chart V4 (S7-1500T)

Function chart: Position kinematics with circular path motion

In the function chart, sections A1 and A2 of the following kinematics motion are viewed:

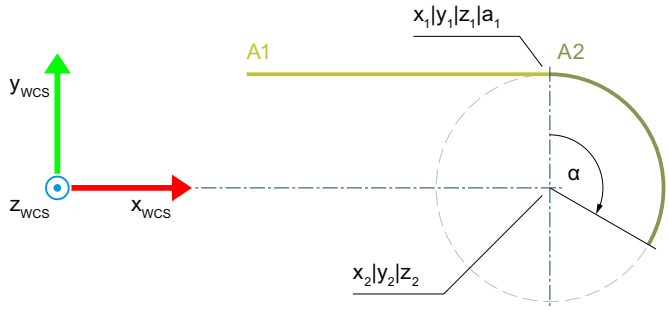

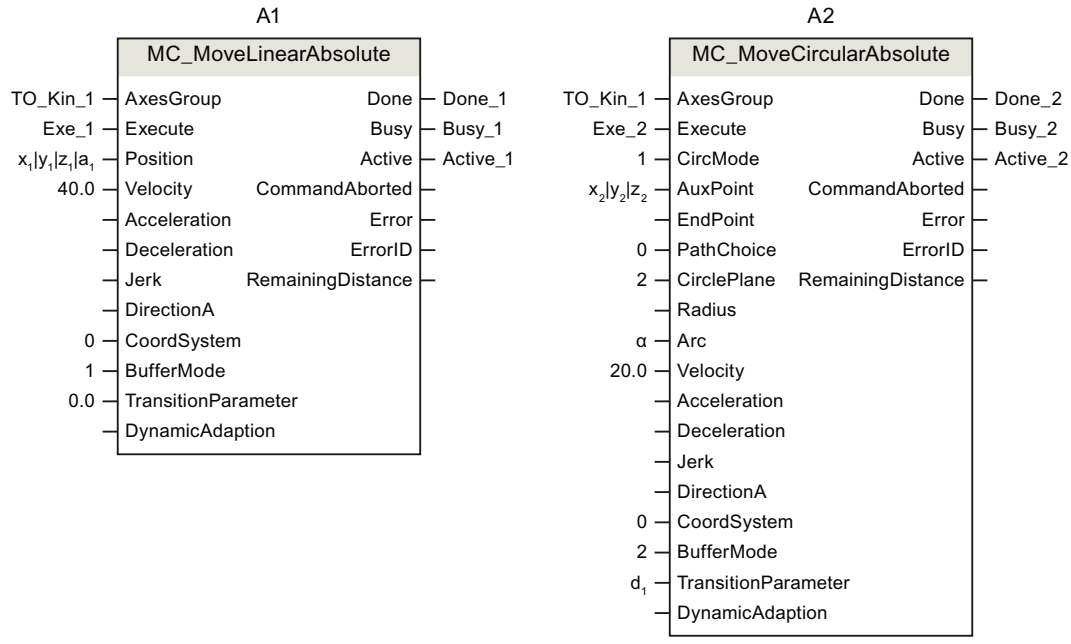

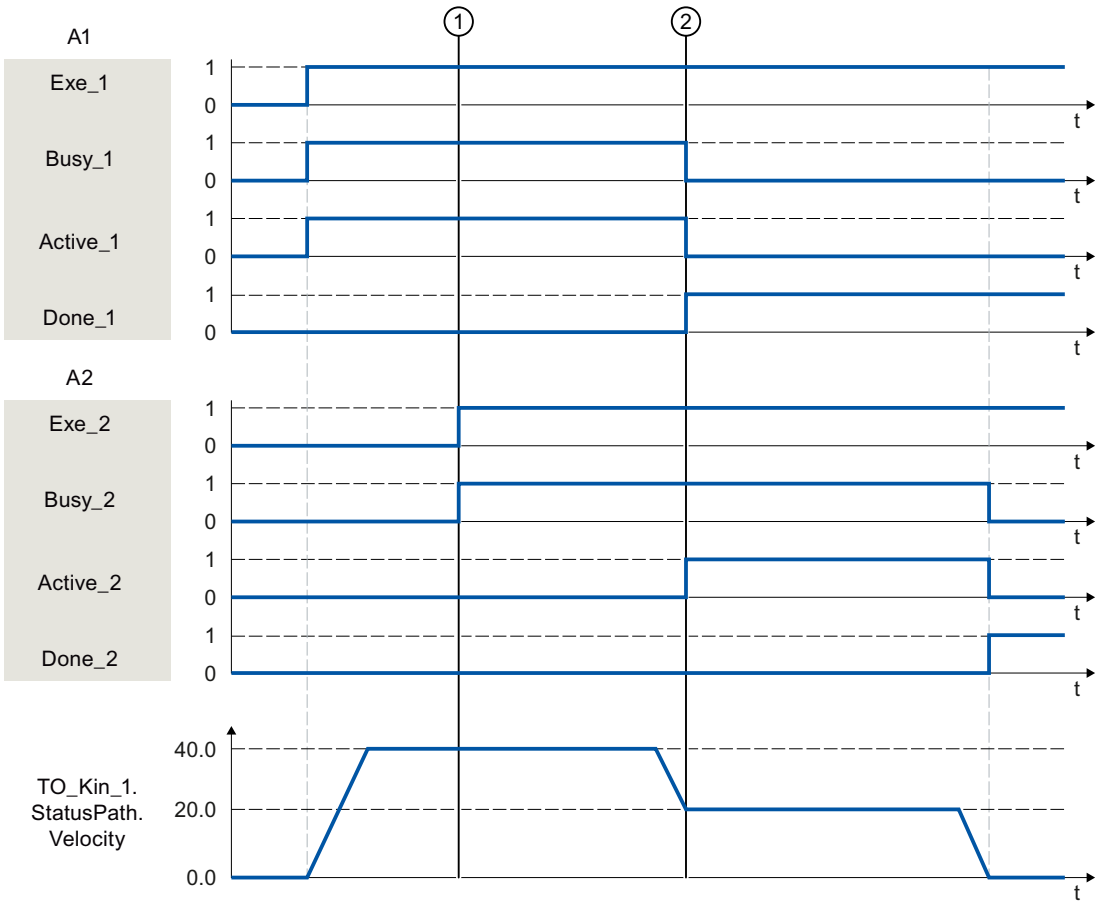

The kinematics is moved with an "MC_MoveLinearAbsolute" job (A1).

At time ① an "MC_MoveCircularAbsolute" job (A2) is initiated. As the "MC_MoveCircularAbsolute" job has no overdrive effect, the job A2 is arranged in the job sequence.

At the point in time ②, completion of the A1 job is signaled via "Done_1" and the job A2 is started. As "BufferMode = 2" is set for job A2, the motion transition is smoothed with the lower velocity of both jobs and the rounding clearance d_1. The circular motion is determined with "CircMode = 1" via the radius r and the end point ($x_2|y_2|z_2$). As soon as the target position is reached, the completion of the A2 job is signaled via "Done_2".

MC_MoveCircularRelative V4 (S7-1500T)

MC_MoveCircularRelative: Relative positioning of kinematics with circular path motion V4 (S7-1500T)

Description

With the Motion Control instruction "MC_MoveCircularRelative", you can move a kinematics with a circular motion to a relative position. Cartesian orientation is also used relative.

With the "CircMode" parameter, you specify the definition of the circular path:

- Via an intermediate point and the end point
  With the "AuxPoint" parameter, you specify an intermediate point on the circular path, via which the end point specified in the "EndPoint" parameter is to be approached. The circular path is calculated via the starting, intermediate and end points. Only circular paths less than 360° can be traversed here.
- Via the circle center and angle in a main plane
  With the "AuxPoint" parameter, you define the center point of the circle. The end point of the circular path is calculated via the angle specified in the parameter "Arc". With the "PathChoice" parameter, you specify whether the circular path is to be traversed in positive or negative direction of rotation. With the "CirclePlane" parameter, you specify the main plane in which the circular path is to be traversed.
- Via the circle radius and the end point in a main plane
  With the "EndPoint" parameter, you specify the end point and with "Radius" the radius of the circular path. Depending on the radius, up to four possible circular paths can occur in the plane defined with the "CirclePlane" parameter. With the "PathChoice" parameter, you specify which of these circular paths is to be moved.

Dynamic behavior during Kinematics motion is defined with the parameters "Velocity", "Acceleration", "Deceleration" and "Jerk". For the dynamics of the orientation motion, the default values are configured in "Technology object > Configuration > Extended parameters > Dynamic defaults".

- <TO>.DynamicDefaults.Orientation.Velocity
- <TO>.DynamicDefaults.Orientation.Acceleration
- <TO>.DynamicDefaults.Orientation.Deceleration
- <TO>.DynamicDefaults.Orientation.Jerk

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- There is no single axis job (e.g. "MC_MoveVelocity") on the interconnected axes.

Override response

With the "TransitionParameter[1]" parameter, you determine the distance from the target point of the previous motion job at which a new motion command is superimposed.

The override response for "MC_MoveCircularAbsolute" jobs is described in section Override response V4: Kinematics motion commands (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveCircularRelative":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
CircMode	INPUT	DINT	0	Definition of circular path	
				0	The position vector specified at the "AuxPoint" parameter defines a point on the circular path.
				1	The position specified at the parameter "AuxPoint" defines the circle center.
				2	The "Radius" and "EndPoint" parameters define the circle segment.
AuxPoint	INPUT	ARRAY [1..3] OF LREAL	-	Circular path auxiliary point • When "CircMode" = 0: Point on the circular path • When "CircMode" = 1: Center point of the circular path • When "CircMode" = 2: Not relevant Coordinate relative to the starting point	
AuxPoint[1]	INPUT	LREAL	0.0	x coordinate	
AuxPoint[2]	INPUT	LREAL	0.0	y coordinate	
AuxPoint[3]	INPUT	LREAL	0.0	z coordinate	
EndPoint	INPUT	ARRAY [1..4] OF LREAL	-	Target position relative to the starting point in the specified reference system	
				When "CircMode" = 1: Only EndPoint[4] relevant (orientation axis)	
EndPoint[1]	INPUT	LREAL	0.0	x coordinate	
EndPoint[2]	INPUT	LREAL	0.0	y coordinate	
EndPoint[3]	INPUT	LREAL	0.0	z coordinate	
EndPoint[4]	INPUT	LREAL	0.0	A coordinate	
PathChoice	INPUT	DINT	0	Orientation of the circular path	
				When "CircMode" = 0: Not relevant	
				When "CircMode" = 1:	
				0	Positive direction of rotation
				1	Negative direction of rotation
				When "CircMode" = 2:	
				0	Shorter positive circular path
				1	Shorter negative circular path
				2	Longer positive circular path
				3	Longer negative circular path

Parameter	Declaration	Data type	Default value	Description	
CirclePlane	INPUT	DINT	0	Main plane of the circular path	
				When "CircMode" = 0: Not relevant	
				When "CircMode" = 1 and 2:	
				0	x-z plane
				1	y-z plane
				2	x-y plane
Radius	INPUT	LREAL	0.0	When "CircMode" = 2: Radius of the circular movement	
Arc	INPUT	LREAL	0.0	When "CircMode" = 1: Angle of the circular movement	
Velocity	INPUT	LREAL	-1.0	Velocity	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration	
				> 0.0	The specified value is used.
				= 0.0	Not permitted
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk	
				> 0.0	The specified value is used.
				= 0.0	No jerk limitation
				< 0.0	The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Path.Jerk)

Parameter	Declaration	Data type	Default value	Description	
CoordSystem	INPUT	DINT	0	Reference system	
				0	World coordinate system (WCS)
				1	Object coordinate system 1 (OCS[1])
				2	Object coordinate system 2 (OCS[2])
				3	Object coordinate system 3 (OCS[3])
BufferMode	INPUT	DINT	1	Motion transition	
				1	Append motion The active motion sequence is completed and the kinematics come to a standstill. The movement of this job is then executed.
				2	Smooth with the lower velocity When the blending distance is reached, the current motion is blended with the motion of this job. The lower velocity of both jobs is then used.
				5	Smooth with the higher velocity When the blending distance is reached, the current motion is blended with the motion of this job. The higher velocity of both jobs is then used.
TransitionParameter	INPUT	ARRAY [1..5] OF LREAL		Transition parameter	
TransitionParameter[1]	INPUT	LREAL	-1.0	Rounding clearance	
				≥ 0.0	The specified value is used.
				< 0.0	The maximum possible rounding clearance is used.
TransitionParameter[2]	INPUT	LREAL	-	Reserved	
TransitionParameter[3]	INPUT	LREAL	-	Reserved	
TransitionParameter[4]	INPUT	LREAL	-	Reserved	
TransitionParameter[5]	INPUT	LREAL	-	Reserved	
DynamicAdaption	INPUT	DINT	-1	Dynamic adaptation	
				< 0	The configured setting in "Technology object > Configuration > Extended parameters > Dynamic" is used. (<TO>.DynamicDefaults.DynamicAdaption)
				0	Without dynamic adaption
				1	Dynamic adaptation with segmentation of the path
				2	Dynamic adaptation without segmentation of the path
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.

Parameter	Declaration	Data type	Default value	Description	
Active	OUTPUT	BOOL	FALSE	TRUE	The setpoints are calculated.
				FALSE	When "Busy" = TRUE: Job is waiting. (Typical: The preceding job is still active.)
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	
RemainingDistance	OUTPUT	LREAL	-1.0	Distance-to-go of the current job	

See also

Override response V4: Kinematics motion commands (Page 3253)

MC_MoveCircularRelative: Function chart V4 (S7-1500T)

Function chart: Relative positioning of kinematics with circular path motion

In the function chart, sections A1 and A2 of the following kinematics motion are viewed:

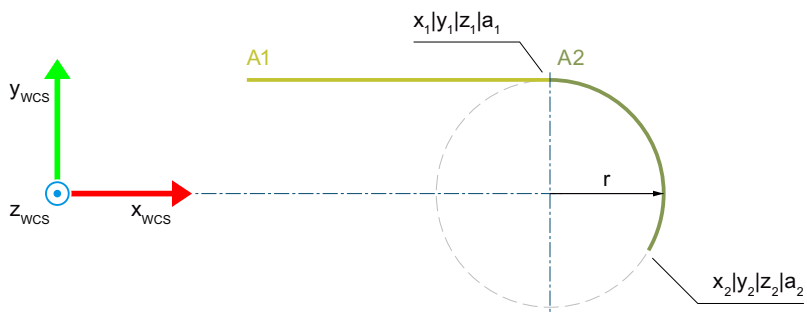

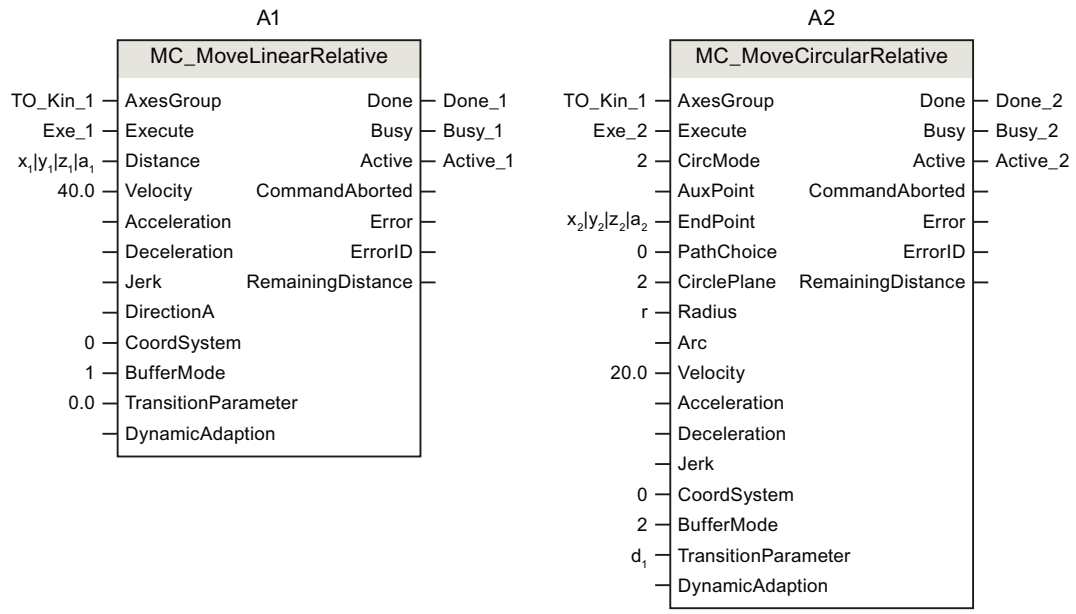

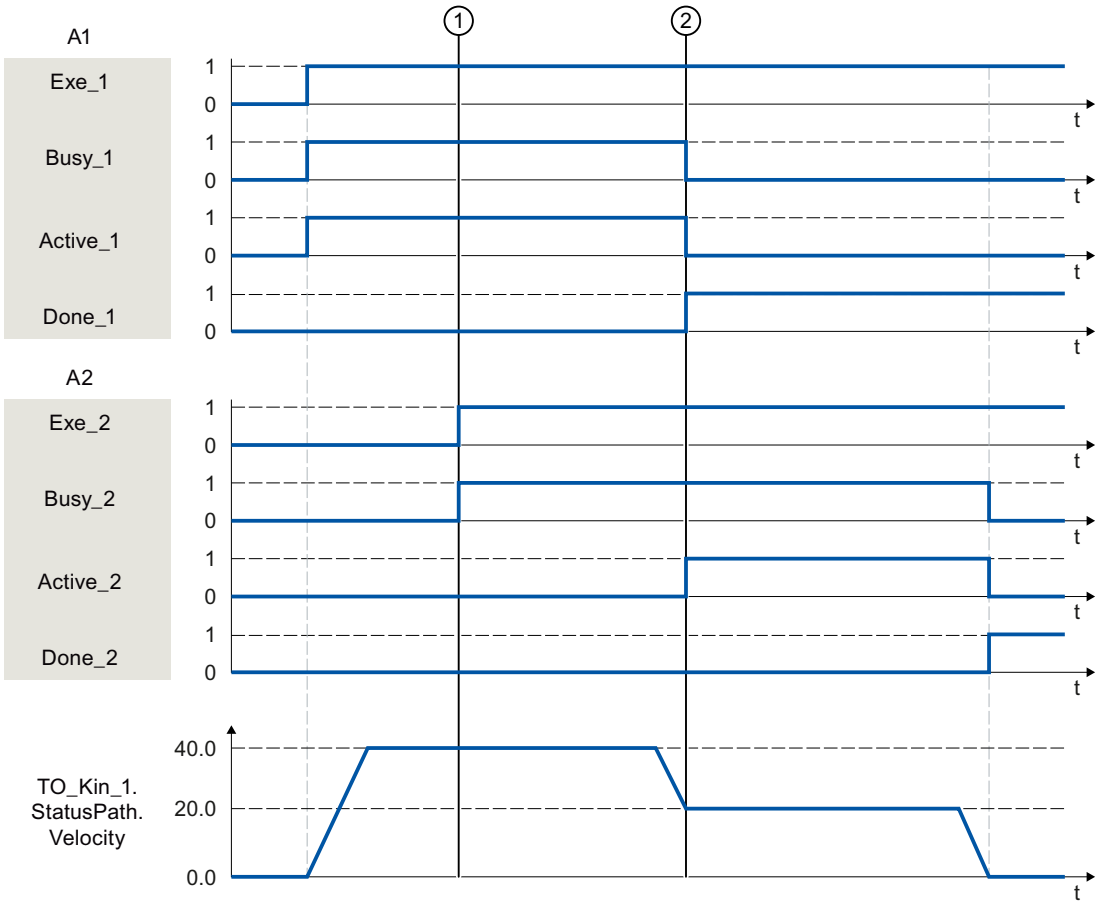

The kinematics is moved with an "MC_MoveLinearRelative" job (A1).

At time ① an "MC_MoveCircularRelative" job (A2) is initiated. As the "MC_MoveCircularRelative" job has no overdrive effect, the job A2 is arranged in the job sequence.

At the point in time ②, completion of the A1 job is signaled via "Done_1" and the job A2 is started. As "BufferMode = 2" is set for job A2, the motion transition is smoothed with the lower velocity of both jobs and the rounding clearance d_1. The circular motion is determined with "CircMode = 2" via the radius r and the end point (x_2|y_2|z_2). As soon as the target position is reached, the completion of the A2 job is signaled via "Done_2".

Zones (S7-1500T)

MC_DefineWorkspaceZone V4 (S7-1500T)

MC_DefineWorkspaceZone: Defining workspace zone V4 (S7-1500T)

Description

With the Motion Control instruction "MC_DefineWorkspaceZone", you define a workspace zone in relation to the world coordinates system or an object coordinates system. The zones (<TO>.WorkspaceZone[1..10]) defined under "Technology object > Configuration > Extended parameters > Zones" are not hereby changed are available again after a restart of the technology object. The "<TO>.StatusWorkspaceZone" tag of the technology object data block contains the currently effective workspace zones.

The "MC_DefineWorkspaceZone" job interjects itself in the job sequence on the Kinematics technology object and therefore effective for the following motion jobs.

With the "GeometryType" and "GeometryParameter" parameters, you specify the zone geometry type and zone size. You use the "ZoneType" parameter to define a workspace zone as work zone, blocked zone or signal zone. You can define up to ten workspace zones. While several of the defined blocked zones and signal zones can be active simultaneously, only one area of the defined work zones can be active.

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.

Override response

- A "MC_DefineWorkspaceZone" job is aborted by an "MC_GroupStop" job.
- A new "MC_DefineWorkspaceZone" job does not abort any active motion control jobs.

Parameter

The following table shows the parameters of Motion Control instruction "MC_DefineWorkspaceZone":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	The job is accepted in the job sequence.
ZoneType	INPUT	DINT	0	Zone type	
				0	Blocked zone
				1	Work zone
				2	Signal zone
ZoneNumber	INPUT	DINT	1	Zone number	
				1	Zone 1
				2	Zone 2
				3	Zone 3
				4	Zone 4
				5	Zone 5
				6	Zone 6
				7	Zone 7
				8	Zone 8
				9	Zone 9
				10	Zone 10
ReferenceSystem	INPUT	DINT	0	Reference system	
				0	World coordinate system (WCS)
				1	Object coordinate system 1 (OCS[1])
				2	Object coordinate system 2 (OCS[2])
				3	Object coordinate system 3 (OCS[3])
Frame	INPUT	TO_Struct_Kinematics_Frame	-	Shifting of the zone zero point in relation to the reference system	
GeometryType	INPUT	DINT	0	Zone geometry type	
				0	Cuboid
				1	Sphere
				2	Cylinder
GeometryParameter	INPUT	ARRAY [1..3] OF LREAL	-	Geometric parameters	

Parameter	Declaration	Data type	Default value	Description	
GeometryParameter[1]	INPUT	LREAL	0.0	Length x	
				When "GeometryType" = 1 or 2: Radius	
GeometryParameter[2]	INPUT	LREAL	0.0	Length y	
				When "GeometryType" = 2: Height	
GeometryParameter[3]	INPUT	LREAL	0.0	Length z	
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

MC_SetWorkspaceZoneActive: Activating workspace zone V4 (Page 3240)

MC_SetWorkspaceZoneInactive: Deactivating workspace zone V4 (Page 3242)

MC_DefineKinematicsZone V4 (S7-1500T)

MC_DefineKinematicsZone: Defining kinematics zone V4 (S7-1500T)

Description

With the Motion Control instruction "MC_DefineKinematicsZone", you define a kinematics zone in relation to the tool and flange coordinate system. The zones (<TO>.KinematicsZone[2..10]) defined under "Technology object > Configuration > Extended parameters > Zones" are not hereby changed are available again after a restart of the technology object. The "<TO>.StatusKinematicsZone" tag of the technology object data block contains the currently effective kinematics zones.

The "MC_DefineKinematicsZone" job is added to the queue of the job sequence and is therefore effective for subsequent motion jobs.

With the "GeometryType" and "GeometryParameter" parameters, you specify the zone geometry type and size. You can define up to nine kinematics zones. Kinematics zone 1 is the tool center point (TCP) and cannot be changed.

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.

Override response

- A "MC_DefineKinematicsZone" job is aborted by an "MC_GroupStop" job.
- A new "MC_DefineKinematicsZone" job does not abort any active motion control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_DefineKinematicsZone":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	The job is accepted in the job sequence.
ZoneNumber	INPUT	DINT	2	Zone number	
				2	Zone 2
				3	Zone 3
				4	Zone 4
				5	Zone 5
				6	Zone 6
				7	Zone 7
				8	Zone 8
				9	Zone 9
				10	Zone 10
ReferenceSystem	INPUT	DINT	0	Reference system	
				0	Flange coordinate system (FCS)
				1	Tool coordinate system (TCS)
Frame	INPUT	TO_Struct_Kinematics_Frame	-	Shifting of the zone zero point in relation to the reference system	
GeometryType	INPUT	DINT	0	Zone geometry type	
				0	Cuboid
				1	Sphere
				2	Cylinder
GeometryParameter	INPUT	ARRAY [1..3] OF LREAL	-	Geometric parameters	

Parameter	Declaration	Data type	Default value	Description	
GeometryParameter[1]	INPUT	LREAL	0.0	Length x	
				When "GeometryType" = 1 or 2: Radius	
GeometryParameter[2]	INPUT	LREAL	0.0	Length y	
				When "GeometryType" = 2: Height	
GeometryParameter[3]	INPUT	LREAL	0.0	Length z	
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

MC_SetKinematicsZoneActive: Activating kinematics zone V4 (Page 3243)

MC_SetKinematicsZoneInactive: Deactivating kinematics zone V4 (Page 3245)

MC_SetWorkspaceZoneActive V4 (S7-1500T)

MC_SetWorkspaceZoneActive: Activating workspace zone V4 (S7-1500T)

Description

The "MC_SetWorkspaceZoneActive" Motion Control instruction is used to activate the workspace zone which you have defined under "Technology object > Configuration > Extended parameters > Zones" of via an "MC_DefineWorkspaceZone" job With the "ZoneNumber" parameter, you enter the number of the zone to be activated.

The "MC_SetWorkspaceZoneActive" job is added to the queue of the job sequence and is therefore effective for subsequent motion jobs.

The "<TO>.StatusWorkspaceZone[1..10].Active" tags of the technology object data block contain the current activation status of the zones. While several of the defined blocked zones and signal zones can be active simultaneously, only one area of the defined work zones can be active.

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.
- The zone to be activated is defined.

Override response

- A "MC_SetWorkspaceZoneActive" job is aborted by an "MC_GroupStop" job.
- A new "MC_SetWorkspaceZoneActive" job does not abort any active motion control jobs.

Parameter

The following table shows the parameters of Motion Control instruction "MC_SetWorkspaceZoneActive":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
ZoneNumber	INPUT	DINT	1	Zone number	
				1	Zone 1
				2	Zone 2
				3	Zone 3
				4	Zone 4
				5	Zone 5
				6	Zone 6
				7	Zone 7
				8	Zone 8
				9	Zone 9
				10	Zone 10
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

MC_DefineWorkspaceZone: Defining workspace zone V4 (Page 3236)

MC_SetWorkspaceZoneInactive: Deactivating workspace zone V4 (Page 3242)

MC_SetWorkspaceZoneInactive V4 (S7-1500T)

MC_SetWorkspaceZoneInactive: Deactivating workspace zone V4 (S7-1500T)

Description

You use Motion Control instruction "MC_SetWorkspaceZoneInactive" to deactivate an active workspace zone. With the "Mode" parameter, you can deactivate a specific zone, all zones of a type or all zones.

The "MC_SetWorkspaceZoneInactive" job is added to the queue of the job sequence and is therefore effective for subsequent motion jobs.

The "<TO>.StatusWorkspaceZone[1..10].Active" tags of the technology object data block contain the current activation status of the zones.

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.
- The zone to be deactivated is defined.

Override response

- A "MC_SetWorkspaceZoneInactive" job is aborted by an "MC_GroupStop" job.
- A new "MC_SetWorkspaceZoneInactive" job does not abort any active motion control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_SetWorkspaceZoneInactive":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge

Parameter	Declaration	Data type	Default value	Description	
ZoneNumber	INPUT	DINT	1	Zone number	
				1	Zone 1
				2	Zone 2
				3	Zone 3
				4	Zone 4
				5	Zone 5
				6	Zone 6
				7	Zone 7
				8	Zone 8
				9	Zone 9
				10	Zone 10
Mode	INPUT	DINT	0	Set zones of a zone type to inactive	
				0	Set a specific zone to inactive
				1	Set all workspace zones to inactive
				2	Set all blocked zones to inactive
				3	Set all signal zones to inactive
				4	Set workspace to inactive.
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

MC_DefineWorkspaceZone: Defining workspace zone V4 (Page 3236)

MC_SetWorkspaceZoneActive: Activating workspace zone V4 (Page 3240)

MC_SetKinematicsZoneActive V4 (S7-1500T)

MC_SetKinematicsZoneActive: Activating kinematics zone V4 (S7-1500T)

Description

The "MC_SetKinematicsZoneActive" Motion Control instruction is used to activate the zone monitoring for a defined kinematics zone which you have defined under "Technology object > Configuration > Extended parameters > Zones" of via an "MC_DefineKinematicsZone" job With the "ZoneNumber" parameter, you enter the number of the kinematics zone to be activated.

The "MC_SetKinematicsZoneActive" job is added to the queue of the job sequence and is therefore effective for subsequent motion jobs.

The "<TO>.StatusKinematicsZone[2..10].Active" tags of the technology object data block contain the current activation status of the kinematics zones.

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.
- The kinematics zone to be activated is defined.

Override response

- An "MC_SetKinematicsZoneActive" job is not aborted by any other Motion Control job.
- A new "MC_SetKinematicsZoneActive" job does not abort any active motion control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_SetKinematicsZoneActive":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
ZoneNumber	INPUT	DINT	2	Zone number	
				2	Zone 2
				3	Zone 3
				4	Zone 4
				5	Zone 5
				6	Zone 6
				7	Zone 7
				8	Zone 8
				9	Zone 9
				10	Zone 10
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.

Parameter	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

MC_DefineKinematicsZone: Defining kinematics zone V4 (Page 3238)

MC_SetKinematicsZoneInactive: Deactivating kinematics zone V4 (Page 3245)

MC_SetKinematicsZoneInactive V4 (S7-1500T)

MC_SetKinematicsZoneInactive: Deactivating kinematics zone V4 (S7-1500T)

Description

You use Motion Control instruction "MC_SetKinematicsZoneInactive" to deactivate an active kinematics zone. With the "Mode" parameter, you can deactivate a specific kinematics zone or all kinematics zones.

The "MC_SetKinematicsZoneInactive" job is added to the queue of the job sequence and is therefore effective for subsequent motion jobs.

The "<TO>.StatusKinematicsZone[2..10].Active" tags of the technology object data block contain the current activation status of the kinematics zones.

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.
- The kinematics zone to be deactivated is defined.

Override response

- An "MC_SetKinematicsZoneInactive" job is not aborted by any Motion Control job.
- A new "MC_SetKinematicsZoneInactive" job does not abort any active motion control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_SetKinematicsZoneInactive":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
ZoneNumber	INPUT	DINT	2	Zone number	
				2	Zone 2
				3	Zone 3
				4	Zone 4
				5	Zone 5
				6	Zone 6
				7	Zone 7
				8	Zone 8
				9	Zone 9
				10	Zone 10
Mode	INPUT	DINT	0	Set zones to inactive	
				0	Set a specific zone to inactive
				1	Set all zones to inactive
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

MC_DefineKinematicsZone: Defining kinematics zone V4 (Page 3238)

MC_SetKinematicsZoneActive: Activating kinematics zone V4 (Page 3243)

Tools (S7-1500T)

MC_DefineTool V4 (S7-1500T)

MC_DefineTool: Redefine tool V4 (S7-1500T)

Description

You redefine the tool frame of the tool 1 job with the "MC_DefineTool" Motion Control instruction. The start values stored in the system are not overwritten. Tool 1 is active by default.

The "MC_DefineTool" job is not added to the queue of the job sequence of the kinematics technology object and therefore takes effect immediately. The "MC_DefineTool" job can be executed only if the kinematics is in standstill.

The configurable coordinates depend on the kinematics type used:

Kinematics type		Configurable coordinates
2D	without orientation	x, z[1)]
	with orientation	z, a[2)]
3D	without orientation	x, y, z[3)]
	with orientation	x, y, z, a

1) The parameters "y" and "a" are pre-defined with the value "0.0".

2) The parameters "x" and "y" are pre-defined with the value "0.0".

3) The parameter "a" can only be defined with the value "0.0".

The following tags of the technology object data block contain the current coordinates of the tool frame 1:

- <TO>.StatusTool.Frame[1].x
- <TO>.StatusTool.Frame[1].y
- <TO>.StatusTool.Frame[1].z
- <TO>.StatusTool.Frame[1].a

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.
- The kinematics is at a standstill.

- The kinematics is not in "Interrupted" state.
- No kinematics motion is active.

Override response

- An "MC_DefineTool" job cannot be aborted by any other Motion Control job.
- A new "MC_DefineTool" job does not abort any active motion control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_DefineTool":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
Frame	INPUT	TO_Struct_Kinematics_KinematicsFrame	-	Coordinates in relation to the FCS	
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

MC_SetTool V4 (S7-1500T)

MC_SetTool: Change active tool V4 (S7-1500T)

Description

With the "MC_SetTool" Motion Control instruction, you activate a tool. With the "ToolNumber" parameter, you specify the tool number. The "MC_SetTool" job can be executed only if the kinematics is in standstill. Tool 1 is active by default.

The "<TO>.StatusTool.ActiveTool" tag of the technology object data block contains the tool number of the currently active tool.

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.
- The kinematics is at a standstill.
- The kinematics is not in "Interrupted" state.
- No kinematics motion is active.

Override response

- An "MC_SetTool" job cannot be aborted by any other Motion Control job.
- A new "MC_SetTool" job does not abort any active motion control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_SetTool":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	Start job with a positive edge
ToolNumber	INPUT	DINT	1	Number of the tool to be activated.	
				1	Tool 1
				2	Tool 2
				3	Tool 3
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Coordinate systems (S7-1500T)

MC_SetOcsFrame V4 (S7-1500T)

MC_SetOcsFrame: Redefining object coordinate systems V4 (S7-1500T)

Description

With the "MC_SetOcsFrame" Motion Control instruction, you define the position of an object coordinate system (OCS) in relation to the world coordinate system (WCS). The initial values stored in the technology object data block are not overwritten in the process.

The "MC_SetOcsFrame" job is added to the queue of the job sequence and is therefore effective only for subsequent motion jobs.

The following tags of the technology object data block contain the current coordinates of the object coordinate systems:

- <TO>.StatusOcsFrame[1..3].x
- <TO>.StatusOcsFrame[1..3].y
- <TO>.StatusOcsFrame[1..3].z
- <TO>.StatusOcsFrame[1..3].a
- <TO>.StatusOcsFrame[1..3].b
- <TO>.StatusOcsFrame[1..3].c

Applies to

- Kinematics

Requirement

- The technology object has been configured correctly.
- The interconnected axes are enabled.
- A single axis job (e.g. "MC_MoveVelocity") is not active on any of the interconnected axes.

Override response

The override response for "MC_SetOcsFrame" jobs is described in section Override response V4: Kinematics motion commands (Page 3253).

Parameters

The following table shows the parameters of Motion Control instruction "MC_SetOcsFrame":

Parameter	Declaration	Data type	Default value	Description	
AxesGroup	INPUT	TO_Kinematics	-	Technology object	
Execute	INPUT	BOOL	FALSE	TRUE	The job is accepted in the job sequence.
Frame	INPUT	TO_Struct_Kinematics_Frame	-	Coordinates in relation to the WCS	
OcsNumber	INPUT	DINT	1	Object coordinate system	
				1	Object coordinate system 1 (OCS[1])
				2	Object coordinate system 2 (OCS[2])
				3	Object coordinate system 3 (OCS[3])
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed. The new values are active for the display and the motion.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being processed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

Override response V4: Kinematics motion commands (Page 3253)

Override response of Motion Control jobs V4 (S7-1500, S7-1500T)

Override response V4: Homing and motion jobs (S7-1500, S7-1500T)

The following table shows how a new Motion Control job affects active homing and motion jobs:

⇒ Active job ⇓ New job	MC_Home "Mode" = 2, 8, 10	MC_Home ("Mode" = 3, 5)	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog	MC_MoveSuperimposed	MC_MotionInVelocity MC_MotionInPosition
MC_Home "Mode" = 3, 5	A	A	A	A	A
MC_Home "Mode" = 9	A	-	-	-	-

⇒ Active job ⇓ New job	MC_Home "Mode" = 2, 8, 10	MC_Home ("Mode" = 3, 5)	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog	MC_MoveSuper-imposed	MC_MotionIn-Velocity MC_MotionIn-Position
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_MotionInVelocity MC_MotionInPosition	-	A	A	A	A
MC_MoveSuperimposed	-	-	-	A	-
MC_GearIn	-	A	A	A	-
MC_GearInPos pending[1)]	-	-	-	-	-
MC_GearInPos active[2)]	-	A	A	A	-
MC_CamIn pending[1)]	-	-	-	-	-
MC_CamIn active[2)]	-	A	A	A	-

A The running job is aborted with "CommandAborted" = TRUE.

- No effect. Running job continues to be executed.

1) "Busy" = TRUE, "StartSync" = FALSE, "InSync" = FALSE

2) "Busy" = TRUE, "StartSync" or "InSync" = TRUE

Override response V4: Synchronous operation jobs (S7-1500, S7-1500T)

The following table shows how a new Motion Control job affects the motion of the axis on active synchronous operation jobs:

⇒ Active job ⇓ New job	MC_GearIn	MC_GearIn-Pos pending[1)]	MC_GearIn-Pos active[2)]	MC_Phasing-Absolute MC_Phasing-Relative	MC_CamIn pending[1)]	MC_CamIn active[2)]
MC_Home "Mode" = 3, 5	A	-	-	-	-	-
MC_Halt	A	-	A	A	-	A
MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog	A	-	A	A	-	A
MC_MotionInVelocity MC_MotionInPosition	A	A	A	-	A	A
MC_MoveSuperimposed	-	-	-	-	-	-
MC_GearIn	A	A	A	A	A	A
MC_GearInPos pending[1)]	-	A	-	-	A	-

⇒ Active job ⇓ New job	MC_GearIn	MC_GearIn-Pos pending[1)]	MC_GearIn-Pos active[2)]	MC_Phasing-Absolute MC_Phasing-Relative	MC_CamIn pending[1)]	MC_CamIn active[2)]
MC_GearInPos active[2)]	A	A	A	A	A	A
MC_PhasingAbsolute MC_PhasingRelative	-	-	-	A	-	-
MC_CamIn pending[1)]	-	A	-	-	A	-
MC_CamIn active[2)]	A	A	A	A	A	A

A The running job is aborted with "CommandAborted" = TRUE.

- No effect. Running job continues to be executed.

1) A pending synchronous operation ("Busy" = TRUE, "StartSync" = FALSE, "InSync" = FALSE) does not cancel active commands. A pending synchronous operation job can only be overridden by a further synchronous operation job on the same following axis. Cancellation is possible with "MC_Power".

2) The status "Busy" = TRUE, "StartSync" or "InSync" = TRUE corresponds to active synchronous operation.

Override response V4: Measuring input jobs (S7-1500, S7-1500T)

The following table shows which new Motion Control jobs will override active measuring input jobs:

⇒ Active job ⇓ New job	MC_MeasuringInput	MC_MeasuringInputCyclic
MC_Home "Mode" = 2, 3, 5, 8, 9, 10	A	A
MC_Home "Mode" = 0, 1, 6, 7	-	-
MC_MeasuringInput MC_MeasuringInputCyclic MC_AbortMeasuringInput	A	A

A The running job is aborted with "CommandAborted" = TRUE.

- No effect. Running job continues to be executed.

Override response V4: Kinematics motion commands (S7-1500T)

Single axis jobs are not overridden by kinematics jobs.

The following table shows how a new Motion Control job affects active kinematics motion jobs:

⇒ Active job ⇓ New job	MC_MoveLinearAbsolute MC_MoveLinearRelative MC_MoveCircularAbsolute MC_MoveCircularRelative MC_DefineWorkspaceZone MC_DefineKinematikZone MC_SetWorkspaceZoneActive MC_SetWorkspaceZoneInactive MC_SetKinematicsZoneActive MC_SetKinematicsZoneInactive MC_SetOcsFrame	MC_GroupInterrupt	MC_GroupStop
MC_Home MC_MoveSuperimposed	N	N	N
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_GearIn MC_GearInPos MC_CamIn MC_MotionInVelocity MC_MotionInPosition MC_GroupStop	A	A	A
MC_GroupInterrupt MC_GroupContinue	B	A	N
MC_MoveLinearAbsolute MC_MoveLinearRelative MC_MoveCircularAbsolute MC_MoveCircularRelative MC_DefineWorkspaceZone MC_DefineKinematikZone MC_SetWorkspaceZoneActive MC_SetWorkspaceZoneInactive MC_SetKinematicsZoneActive MC_SetKinematicsZoneInactive MC_SetOcsFrame	-	-	N

A The running job is aborted with "CommandAborted" = TRUE.

B Running job is interrupted or resumed.

N Not permitted. Running job continues to be executed. The new job is rejected.

- No effect. Running job continues to be executed. The new job is added to the queue of the job sequence.

S7-1500 Motion Control V3 (S7-1500, S7-1500T)

MC_Power V3 (S7-1500, S7-1500T)

MC_Power: Enable, disable technology object V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_Power", a technology object is enabled or disabled.

Applies to

- Synchronous axis
- Positioning axis
- Speed axis
- External encoder

Requirements

- The technology object has been configured correctly.

Override response

- An "MC_Power" job cannot be aborted by any other Motion Control job.
- An "MC_Power" job with parameter "Enable" = TRUE enables a technology object but does not thereby abort any other Motion Control instructions.
- Disabling the technology object (parameter "Enable" = FALSE) aborts all Motion Control jobs on the corresponding technology object in accordance with the selected "StopMode". This process cannot be canceled by the user.

Parameters

The following table shows the parameters of the "MC_Power" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The technology object is enabled.
				FALSE	The technology object is disabled. All current jobs at the technology object are aborted in accordance with the configured "StopMode".

Parameters	Declaration	Data type	Default value	Description	
StartMode	INPUT	DINT	1	0	Enable positioning axis/synchronous axis not position-controlled
				1	Enable positioning axis/synchronous axis position-controlled
				The parameter acts initially when the positioning axis is enabled (Enable changes from FALSE to TRUE) and when the axis is enabled after successful acknowledgment of an alarm that caused the axis to be disabled. This parameter is ignored when a speed axis or external encoder is used.	
StopMode	INPUT	INT	0	Not applicable to the technology object external encoder If you disable a technology object with a negative edge at parameter "Enable", the axis decelerates in accordance with the selected "StopMode".	
				0	Emergency stop When the technology object is disabled, the axis brakes to a standstill without jerk limit, using the emergency stop deceleration configured in "Technology object > Configuration > Extended parameters > Emergency stop ramp". The enable is then canceled. (<TO>.DynamicDefaults.EmergencyDeceleration)
				1	Immediate stop When a technology object is disabled, the setpoint zero is output, and the enable is canceled. The axis is braked to a standstill according to the configuration in the drive.
				2	Stop with maximum dynamic values When the technology object is disabled, the axis is braked to a standstill using the maximum deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic limits". The configured maximum jerk is hereby taken into account. The enable is then canceled. (<TO>.DynamicLimits.MaxDeceleration; <TO>.DynamicLimits.MaxJerk)

Parameters	Declaration	Data type	Default value	Description	
Status	OUTPUT	BOOL	FALSE	Technology object enable status	
				FALSE	Disabled - A positioning axis, synchronous axis or speed axis does not accept any Motion Control jobs. - Speed control and position control are not active. - The actual values of the technology object are not checked for validity.
				TRUE	Enabled - An enabled positioning axis, synchronous axis or speed axis accepts Motion Control jobs. - Speed control and position control are active. - The actual values of the technology object are valid.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction MC_Power. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Enabling technology objects

To enable a technology object, set the "Enable" parameter to TRUE.

When the Status parameter shows the value TRUE , the technology object is enabled.

If an axis is in motion when the technology object is enabled (actual velocity is present), the axis brakes to the setpoint zero, using the maximum deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic limits" (<TO>.DynamicLimits.MaxDeceleration). This braking ramp can be overridden by Motion Control jobs.

Note

Automatic enable after acknowledgment of a technology alarm

If the technology object is disabled due to a technology alarm, it will automatically be enabled again after the cause has been eliminated and the alarm has been acknowledged. This requires the "Enable" parameter to have retained the value TRUE during this process.

Disabling technology objects

To disable a technology object, set the "Enable" parameter to FALSE.

If an axis is in motion, it is braked to a standstill according to the selected "StopMode".

When the "Busy" and "Status" parameters show the value FALSE , the disabling of the technology object is complete.

Drive connection by means of PROFIdrive

When a drive is connected using PROFIdrive, the setpoint, enable and drive status are transmitted via the PROFIdrive telegram.

- **Enable technology object and activate drive**
  With parameter "Enable" = TRUE, the technology object is enabled. The drive is enabled according to the PROFIdrive standard.
  When the tag <TO>.StatusDrive.InOperation shows the value TRUE , the drive is ready to execute setpoints. The "Status" parameter is set to the value TRUE .
- **Disable technology object and deactivate drive**
  With parameter "Enable" = FALSE, the "Status" parameter is set to the value FALSE, and the axis is braked according to the selected "StopMode". The drive is disabled according to the PROFIdrive standard.

Analog drive connection

The setpoint is output via an analog output. Optionally, you can configure an enabling signal via digital output (<TO>.Actor.Interface.EnableDriveOutput), and a readiness signal via digital input (<TO>.Actor.Interface.DriveReadyInput).

- **Enable technology object and activate drive**
  With parameter "Enable" = TRUE, the enable output ("Enable drive output") is set.
  When the drive returns the readiness signal via the ready input ("Drive ready input"), the "Status" parameter and the technology object's <TO>.StatusDrive.InOperation tag are set to TRUE, and the setpoint is switched to the analog output.
- **Disable technology object and deactivate drive**
  With parameter "Enable" = FALSE, the "Status" parameter is set to the value FALSE, and the axis is braked according to the selected "StopMode". When the setpoint zero is reached, the enable output is set to FALSE.

Additional information

Additional information on enabling and disabling technology objects and drives can be found in the appendix, MC_Power Function Charts.

MC_Power: Function chart V3 (S7-1500, S7-1500T)

Function chart: Enabling a technology object and example of alarm response

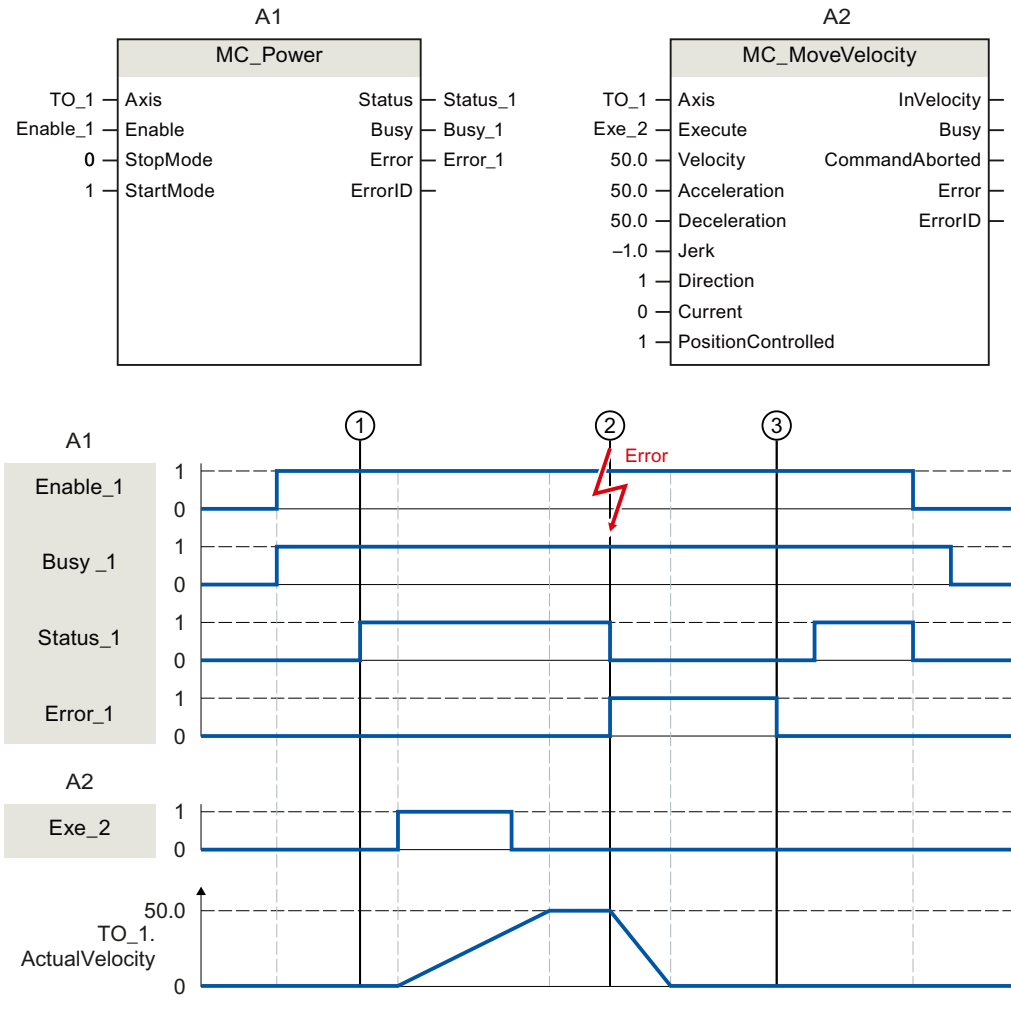

A technology object is enabled with "Enable_1= TRUE". The successful enable can be read from "Status_1" at time ①. The axis will then move with an "MC_MoveVelocity" job (A2). The velocity profile of the axis can be read from "Velocity Axis_1".

At time ② an error occurs in the technology object, which results in the disabling of the technology object (alarm response: remove enable). The axis is braked to a standstill according to the configuration in the drive. When the technology object is disabled, "Status_1" is reset. Since the axis was not disabled using "Enable_1" = FALSE, the selected "StopMode" does not apply. The cause of the error is corrected and the alarm is acknowledged at time ③.

Since "Enable_1" is still set, the technology object is enabled again. The successful enable can be read from "Status_1". Finally, the technology object is disabled with "Enable_1" = FALSE.

MC_Reset V3 (S7-1500, S7-1500T)

MC_Reset: Acknowledge alarms, restart technology object V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_Reset", you acknowledge all technology alarms that can be acknowledged in the user program. Acknowledgment also resets the "Error" and "Warning" bits in the technology data block.

With the Motion Control instruction "MC_Reset" with "Restart" = TRUE, you start reinitialization (restart) of technology objects. Upon restart of the technology object, the new configuration data are applied in the technology data block.

Applies to

- All technology objects

Requirement

- Positioning/synchronous axis, speed axis and external encoder technology objects:
  For a restart, the technology object must be disabled.
  ("MC_Power.Status" = FALSE and "MC_Power.Busy" = FALSE)

Override response

- Parameter "Restart" = FALSE:
  The MC_Reset job does not abort any active motion control jobs.
- Parameter "Restart" = TRUE:
  Processing of the "MC_Reset" instruction with parameter "Restart" = TRUE cannot be aborted by any other Motion Control job.

Parameters

The following table shows the parameters of the "MC_Reset" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Object	-	Technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
Restart	INPUT	BOOL	FALSE	TRUE	"Restart" Reinitialization of the technology object and acknowledgment of pending technology alarms. The technology object is reinitialized with the configured start values.
				FALSE	Acknowledgment of queued technology alarms.
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged. Restart has been executed.

Parameters	Declaration	Data type	Default value	Description	
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Acknowledging technology alarms

To acknowledge technology alarms, follow these steps:

1. Check the requirements indicated above.
2. Set parameter "Restart" = FALSE.
3. Start the acknowledgment of the error with a positive edge at parameter "Execute".

When the Done parameter shows the value TRUE, the error has been acknowledged.

Note

Acknowledging with "Restart" = FALSE

If only the technology alarms are to be acknowledged, set "Restart" = FALSE. The technology object cannot be used during a restart.

Restarting a technology object

To restart a technology object, follow these steps:

1. Check the requirements indicated above.
2. Set parameter "Restart" = TRUE.
3. Perform the restart with a positive edge at parameter "Execute".

When the "Done" parameter shows the value TRUE, the restart of the technology object is complete.

Additional information regarding the restart can be found in the Restarting technology objects section.

MC_Home V3 (S7-1500, S7-1500T)

MC_Home: Home technology object, set home position V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_Home", you create the relationship between the position in the technology object and the mechanical position. The actual position value in the technology object is assigned to a homing mark at the same time. This homing mark represents a known mechanical position.

Homing is performed according to the mode selected with the "Mode" parameter and the configuration under "Technology object > Configuration > Extended parameters > Homing".

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and ≥ V2.0 is available in section Version overview.

The preset values under "Technology object > Configuration > Extended parameters > Dynamic default values" are used for the dynamic values Acceleration, Deceleration and Jerk.

Applies to

- Synchronous axis
- Positioning axis
- External encoder

The following table shows which modes are possible with each of the technology objects:

Operating mode	Positioning axis/ synchronous axis with incremental encoder	Positioning axis/ synchronous axis with absolute encoder	External incremental encoder	External absolute encoder
Active homing ("Mode" = 3, 5)	X	-	-	-
Passive homing ("Mode" = 2, 8, 10)	X	-	X	-
Set actual position ("Mode" = 0)	X	X	X	X
Relative shift of the actual position ("Mode" = 1)	X	X	X	X
Absolute encoder adjustment ("Mode" = 6, 7)	-	X	-	X

Requirement

- The technology object has been configured correctly.
- "Mode" = 2, 3, 5, 8, 10
  The technology object is enabled.
- "Mode" = 0, 1, 6, 7, 8
  The actual encoder values are valid (<TO>.StatusSensor[n].State = 2).
- "Mode" = 6, 7
  The axis is in position-controlled mode.

Override response

The override response for "MC_Home" jobs is described in section Override response of Motion Control commands V3 (Page 3345).

Parameters

The following table shows the parameters of the "MC_Home" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description
Axis	INPUT	TO_Axis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Position	INPUT	LREAL	0.0	The specified value is used according to the selected "Mode".

Parameter	Declaration	Data type	Default value	Description	
Mode	INPUT	INT	0	Operating mode	
				0	Direct homing (absolute) The current position of the technology object is set to the value of parameter "Position".
				1	Direct homing (relative) The current position of the technology object is shifted by the value of parameter "Position".
				2	Passive homing (without reset) Functions like "Mode" 8, with the difference that the "homed" status is **not** reset when the function is enabled.
				3	Active homing The TO positioning axis/synchronous axis performs a homing movement according to the configuration. After the completion of the motion, the axis is positioned at the value of the "Position" parameter.
				4	Reserved
				5	Active homing ("Position" parameter has no effect) The TO positioning axis/synchronous axis performs a homing movement according to the configuration. After completion of the motion, the axis is positioned at the home position configured under "Technology object > Configuration > Extended parameters > Homing > Active homing". (<TO>.Homing.HomePosition)
				6	Absolute encoder adjustment (relative) The current position is shifted by the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<TO>.StatusSensor[n].AbsEncoderOffset)
				7	Absolute encoder adjustment (absolute) The current position is set to the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<TO>.StatusSensor[n].AbsEncoderOffset)
				8	Passive homing

Parameter	Declaration	Data type	Default value	Description	
					When the homing mark is detected, the actual value is set to the value of the "Position" parameter.
				9	Abort passive homing An active job for passive homing is aborted.
				10	Passive homing ("Position" parameter has no effect) When the homing mark is detected, the actual value is set to the home position configured under "Technology object > Configuration > Extended parameters > Homing > Passive homing". (<TO>.Homing.HomePosition)
ReferenceMarkPosition	OUTPUT	LREAL	0.0	Display of the position at which the technology object was homed (valid when "Done" = TRUE)	
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Resetting the "Homed" status

The "Homed" status of a technology object is reset under the following conditions (<TO>.StatusWord.X5 (HomingDone)):

- **Technology objects with incremental actual values:**
  - Starting an "MC_Home" job with "Mode" = 3, 5, 8, 10
    (After successful completion of the homing operation, the "Homed" status is reset.)
  - Error in the encoder system, or encoder failure
  - Restart of the technology object
  - After POWER OFF -> POWER ON of the CPU
  - Memory reset
  - Modification of the encoder configuration
- **Technology objects with absolute actual values:**
  - Error in sensor system/encoder failure
  - Replacement of the CPU
  - Modification of the encoder configuration
  - Restoration of the CPU factory settings
  - Transfer of a different project into the controller

Homing a technology object with "Mode" = 1..8, 10

To home a technology object, follow these steps:

1. Check the requirements indicated above.
2. Specify the desired homing function in the "Mode" parameter.
3. Initialize the necessary parameters with values, and start the homing operation with a positive edge at the "Execute" parameter.

When the "Done" parameter shows the value TRUE, the "MC_Home" job has been completed according to the selected "Mode". The "Homed" status of the technology object is indicated under "Technology object > Diagnostics > Status and error bits > Motion status > Homed" (<TO>.StatusWord.X5 (HomingDone)).

Abort of a passive homing process with "Mode" = 9

With "Mode" = 9, the technology object is not homed. When an active "MC_Home" job for passive homing ("Mode" = 2, 8, 10) is overridden by another "MC_Home" job with "Mode" = 9, the running job is aborted with parameter "CommandAborted" = TRUE. The overriding job with "Mode" = 9 signals successful execution with parameter "Done" = TRUE.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

See also

Override response of Motion Control commands V3 (Page 3345)

MC_Halt V3 (S7-1500, S7-1500T)

MC_Halt: Stop axis V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_Halt", you brake an axis until it comes to a standstill.

You define the dynamic behavior of the braking operation with parameters "Jerk" and "Deceleration".

Applies to

- Synchronous axis
- Positioning axis
- Speed axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The override characteristics for "MC_Halt" jobs are described in section Override response of Motion Control commands V3 (Page 3345).

Parameters

The following table shows the parameters of the "MC_Halt" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	INPUT	TO_SpeedAxis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)

Parameters	Declaration	Data type	Default value	Description	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
AbortAcceleration	INPUT	BOOL	FALSE	FALSE	The current acceleration at the start of the job is reduced using the configured jerk. Afterwards, the deceleration builds up
				TRUE	The acceleration is set to 0.0 at the start of the job, and the deceleration immediately builds up.
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Braking an axis with "MC_Halt"

Proceed as follows to decelerate an axis to a standstill:

1. Check the requirements indicated above.
2. Supply the parameters "Deceleration", "Jerk" and "AbortAcceleration" with the desired values.
3. Start the "MC_Halt" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error". The standstill of the axis is indicated under "Technology object > Diagnostics > Status and error bits > Motion status > Standstill" (<TO>.StatusWord.X7 (Standstill)).

Braking an axis with active force/torque limit

With active force/torque limiting the permitted torque may not be sufficient to move the calculated braking ramp. The "MC_Halt" command does not signal "Done" = TRUE.

To brake an axis with active force/torque limiting, use the Motion Control instruction "MC_MoveVelocity" with "MC_MoveVelocity.Velocity" = 0.0 and "MC_MoveVelocity.PositionControlled" = FALSE.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

See also

Override response of Motion Control commands V3 (Page 3345)

MC_MoveVelocity: Move axis at velocity/speed setpoint V3 (Page 3279)

MC_Halt: Function chart V3 (S7-1500, S7-1500T)

Function chart: Stopping an axis and the overriding job characteristics

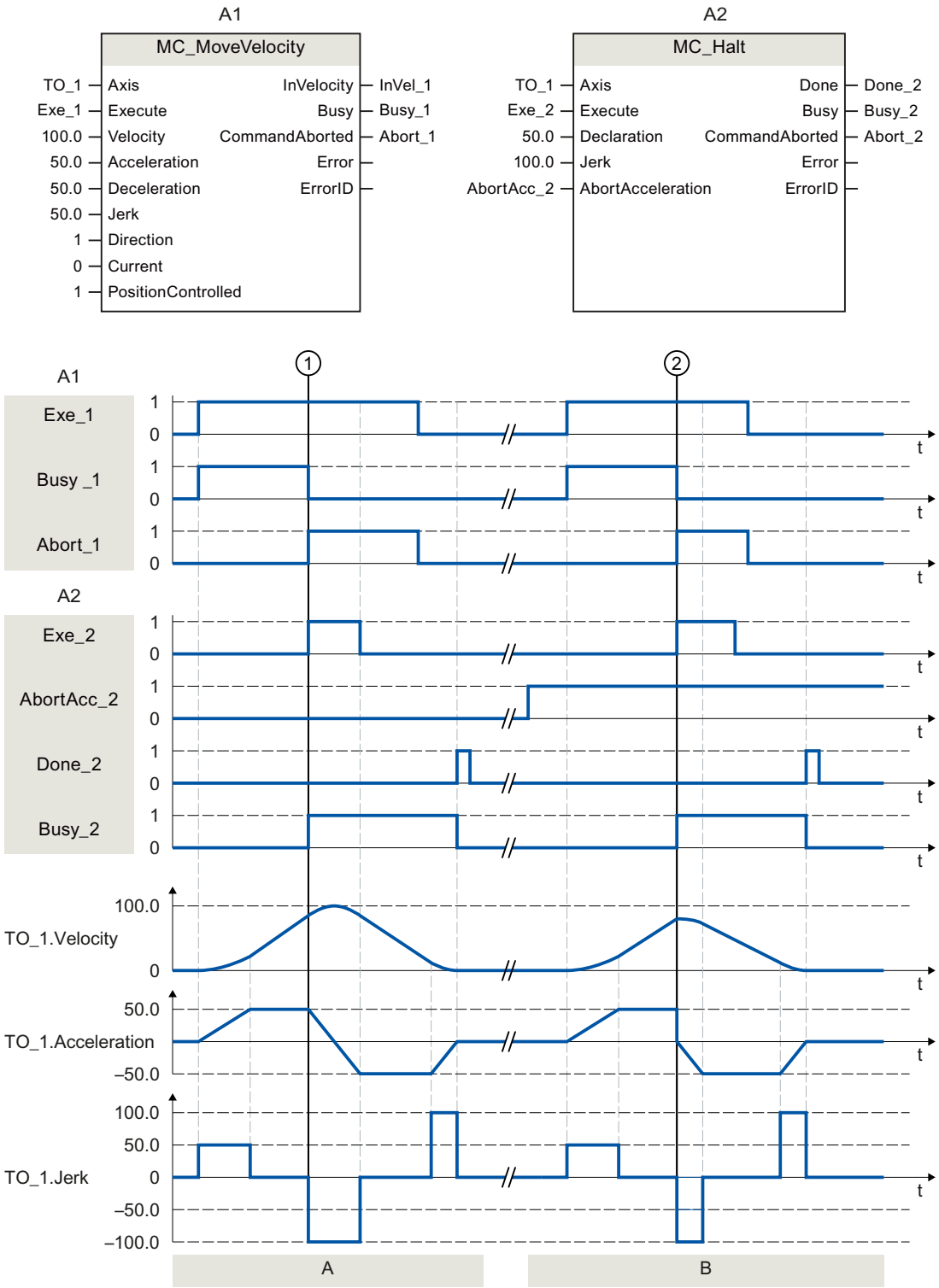

Section A	An axis is moved with an "MC_MoveVelocity" job (A1). At time ①, the "MC_MoveVelocity" job is overridden by an "MC_Halt" job (A2). The job abort is signaled via "Abort_1". With "AbortAcc_2" = FALSE, the current acceleration is reduced with the specified jerk. Afterward, the deceleration builds up and the axis is braked to a standstill. Successful completion of the "MC_Halt" job is signaled via "Done_2".
Section B	The axis is moved with an "MC_MoveVelocity" job (A1). At time ②, the "MC_MoveVelocity" job is overridden by an "MC_Halt" job (A2). The job abort is signaled via "Abort_1". With "AbortAcc_2" = TRUE, the current acceleration is set to zero immediately and the deceleration builds up. The axis is braked to a standstill. Successful completion of the "MC_Halt" job is signaled via "Done_2".

MC_MoveAbsolute V3 (S7-1500, S7-1500T)

MC_MoveAbsolute: Position axis absolutely V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MoveAbsolute", you can move an axis to an absolute position.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

Applies to

- Synchronous axis
- Positioning axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.
- The technology object is homed.

Override response

The override response for "MC_MoveAbsolute" jobs is described in section Override response of Motion Control commands V3 (Page 3345).

Parameters

The following table shows the parameters of the "MC_MoveAbsolute" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	INPUT	TO_Positioning-gAxis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Position	INPUT	LREAL	0.0	Absolute target position
Velocity	INPUT	LREAL	-1.0	Velocity setpoint for the positioning Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
Direction	INPUT	INT	1	Motion direction of the axis Is only evaluated with "modulo" enabled. "Technology object > Configuration > Basic parameters > Enable modulo 1 Positive direction 2 Negative direction 3 Shortest distance
Done	OUTPUT	BOOL	FALSE	TRUE Target position reached.
Busy	OUTPUT	BOOL	FALSE	TRUE Job is running.

Parameters	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Moving an axis to an absolute position

Proceed as follows to move an axis to an absolute position:

1. Check the requirements indicated above.
2. Specify the desired target position in the "Position" parameter.
3. Start the "MC_MoveAbsolute" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error".

See also

Override response of Motion Control commands V3 (Page 3345)

MC_MoveAbsolute: Function chart V3 (S7-1500, S7-1500T)

Function chart: Absolute positioning of an axis, and the response to an overriding job

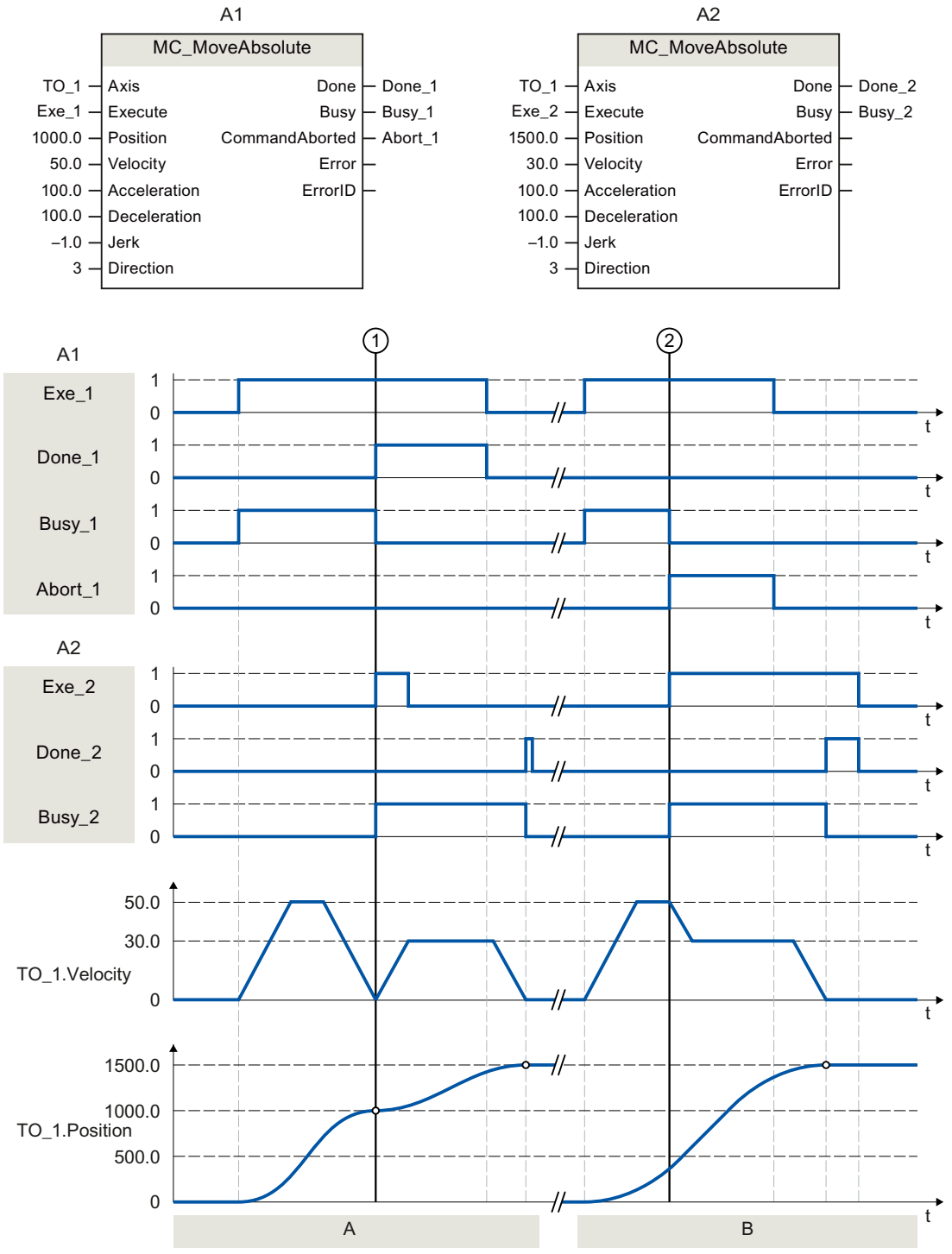

Section A	An axis is moved to absolute position 1000.0 with an "MC_MoveAbsolute" job (A1). When the axis reaches the target position, this is signaled at time ① via "Done_1". At this time ①, another "MC_MoveAbsolute" job (A2) with target position 1500.0 is started. When the axis reaches the target position 1500.0, this is signaled via "Done_2". Since "Exe_2" was previously reset, "Done_2" is applied only to one cycle.
Section B	An active "MC_MoveAbsolute" job (A1) is overridden at time ② by another "MC_MoveAbsolute" job (A2). The abort is signaled via "Abort_1". The axis is braked to the changed velocity and moved to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveRelative V3 (S7-1500, S7-1500T)

MC_MoveRelative: Relative positioning of axis V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MoveRelative", you move an axis relative to its position when execution of the job began.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

Applies to

- Synchronous axis
- Positioning axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The override characteristics for "MC_MoveRelative" jobs are described in section Override response of Motion Control commands V3 (Page 3345).

Parameters

The following table shows the parameters of the "MC_MoveRelative" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	INPUT	TO_Positionin-gAxis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Distance	INPUT	LREAL	0.0	Distance for the positioning process (negative or positive)

Parameters	Declaration	Data type	Default value	Description	
Velocity	INPUT	LREAL	-1.0	Velocity setpoint for the positioning Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)	
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Moving an axis relative to the starting position

Proceed as follows to move an axis relative to the starting position:

1. Check the requirements indicated above.
2. Specify distance to be moved in the "Distance" parameter.
3. Start the "MC_MoveRelative" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error".

See also

Override response of Motion Control commands V3 (Page 3345)

MC_MoveRelative: Function chart V3 (S7-1500, S7-1500T)

Function chart: Relative positioning of an axis, and the overriding job characteristics

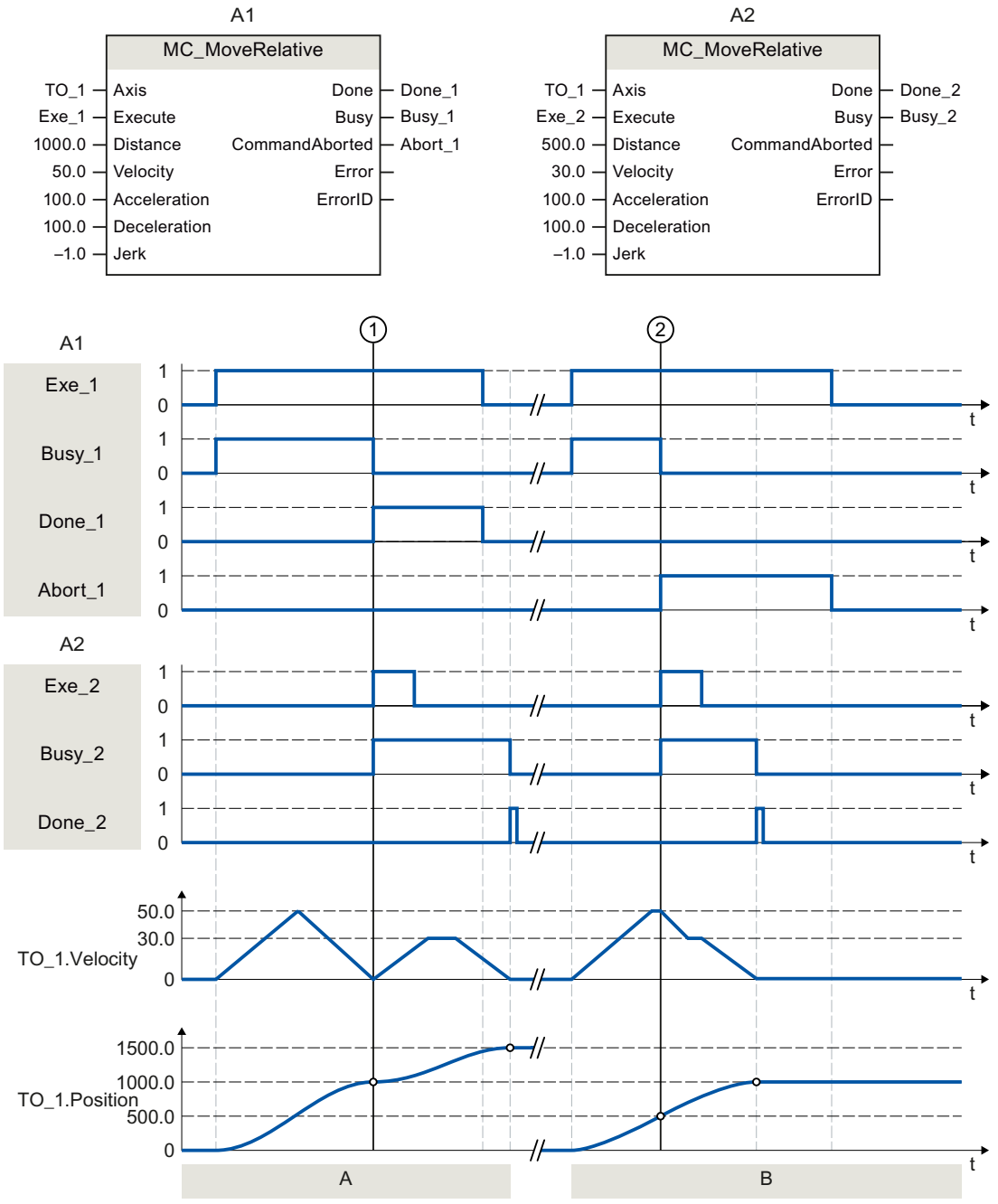

Section A	The axis is moved by an "MC_MoveRelative" job (A1) by the distance ("Distance") 1000.0 (the starting position here is 0.0). When the axis reaches the target position, this is signaled at time ① via "Done_1". At this time ①, another "MC_MoveRelative" job (A2) with distance 500.0 is started. When the axis reaches the new target position, this is signaled via "Done_2". Since "Exe_2" was previously reset, "Done_2" is applied only to one cycle.
Section B	An active "MC_MoveRelative" job (A1) is overridden by another "MC_MoveRelative" job (A2). The abort is signaled at time ② via "Abort_1". The axis is then moved at the new velocity by the distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveVelocity V3 (S7-1500, S7-1500T)

MC_MoveVelocity: Move axis at velocity/speed setpoint V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MoveVelocity", you move an axis at constant velocity/speed.

You define the dynamic behavior of the motion with parameters"Velocity", "Jerk", "Acceleration" and "Deceleration".

- Positioning axis/synchronous axis:
  A velocity is specified in the "Velocity" parameter.
- Speed axis:
  A speed is specified in the "Velocity" parameter.

Applies to

- Synchronous axis
- Positioning axis
- Speed axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The override response for "MC_MoveVelocity" jobs is described in section Override response of Motion Control commands V3 (Page 3345).

Parameters

The following table shows the parameters of the "MC_MoveVelocity" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
Velocity	INPUT	LREAL	100.0	Velocity/speed setpoint for the motion process ("Velocity" = 0.0 is permitted)	
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used. Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Direction	INPUT	INT	0	Direction of rotation of the axis	
				0	The sign of the velocity specified at the "Velocity" parameter defines the direction of rotation.
				1	Positive direction of rotation Value of "Velocity" is used.
				2	Negative direction of rotation Value of "Velocity" is used.

Parameters	Declaration	Data type	Default value	Description	
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	Deactivated The values of parameters "Velocity" and "Direction" are taken into account.
				TRUE	activated The values at the parameters "Velocity" and "Direction" are **not** taken into account. The current velocity and direction at function start are retained. When the axis resumes motion at the velocity that was current at function start, the "InVelocity" parameter returns the value TRUE.
PositionControlled	INPUT	BOOL	TRUE	FALSE	Non position-controlled operation
				TRUE	Position-controlled mode
				The parameter applies as long as the "MC_MoveVelocity" job is being executed. After this, the setting of the following job applies. This parameter is ignored when a speed axis is used.	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity setpoint / speed setpoint was reached and will be maintained.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Behavior with velocity setpoint / speed setpoint zero ("Velocity" = 0.0)

An "MC_MoveVelocity" job with "Velocity" = 0.0 stops the axis with the configured deceleration. When the velocity setpoint / speed setpoint zero is reached, the parameter "InVelocity" will indicate the value TRUE.

Under "Technology object > Diagnostics > Status and error bits > Motion status", "constant velocity" and "standstill" will be displayed (<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill)).

The parameters "InVelocity" and "Busy" show the value TRUE until the "MC_MoveVelocity" job is overridden by another Motion Control job.

Moving an axis with constant velocity / speed

Proceed as follows to move an axis with constant velocity / speed:

1. Check the requirements indicated above.
2. At the parameter "Velocity", specify the velocity / speed, with which the axis should be moved.
3. Start the "MC_MoveVelocity" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "InVelocity" and "Error".

If the "InVelocity" parameter shows the value TRUE, then the velocity setpoint / speed setpoint was reached. The axis continues moving at this constant velocity. The parameters "InVelocity" and "Busy" show the value TRUE until the "MC_MoveVelocity" job is overridden by another Motion Control job.

Note

Response to a change in the override

If the velocity / speed is influenced during constant motion by a change in the override (<TO>.Override.Velocity), the "InVelocity" parameter is reset during the acceleration or deceleration. When the newly calculated velocity / speed is reached ("Velocity" × "Override" %), then "InVelocity" is reset.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

See also

Override response of Motion Control commands V3 (Page 3345)

MC_MoveVelocity: Function chart V3 (S7-1500, S7-1500T)

Function chart: Moving an axis with velocity specification, and the response to an overriding job

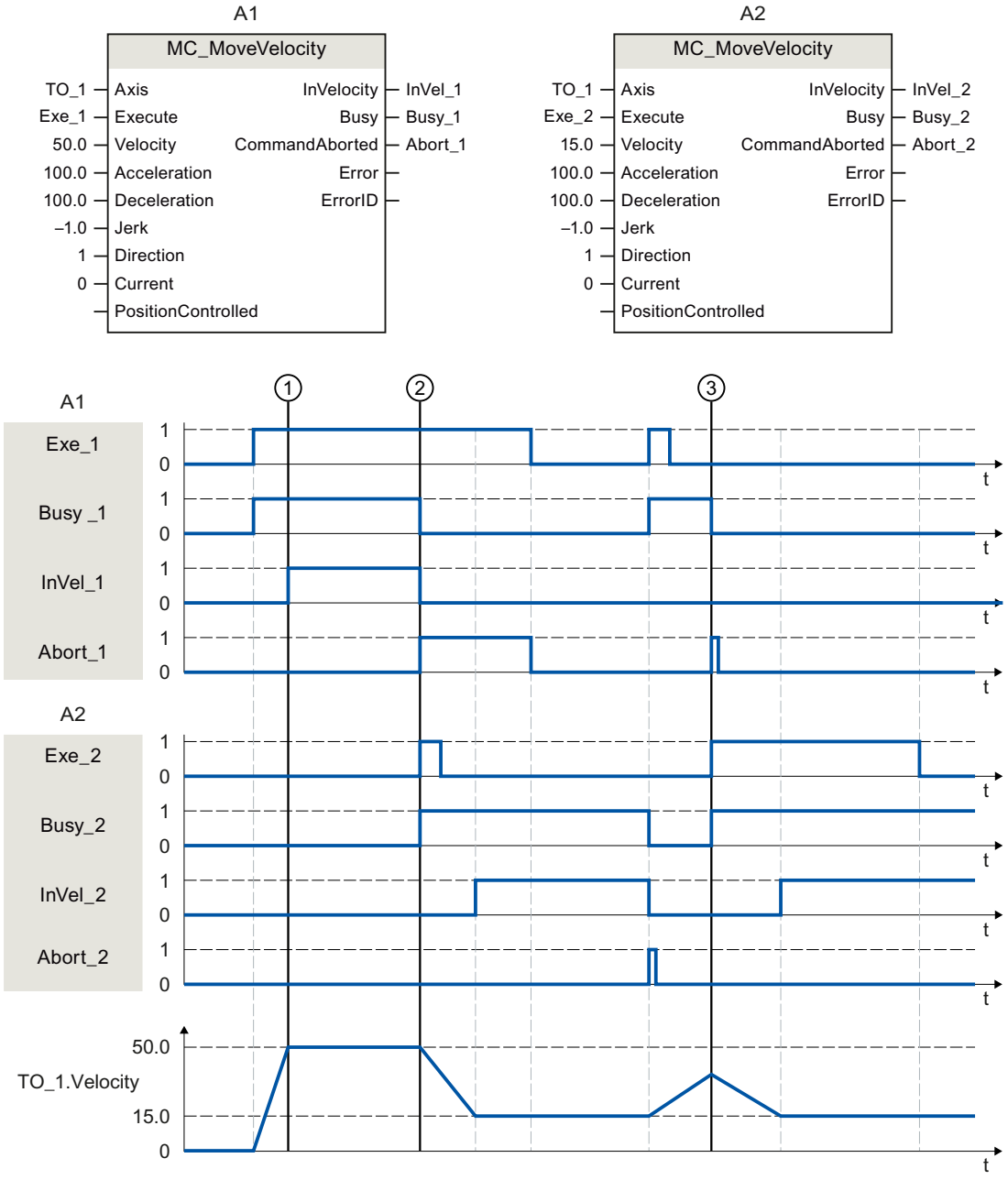

An "MC_MoveVelocity" job (A1) initiated via "Exe_1" accelerates the axis and signals at time ① via "InVel_1" that the velocity setpoint 50.0 has been reached.

At time ②, the job is overridden by another "MC_MoveVelocity" job (A2). The abort is signaled via "Abort_1". When the new velocity setpoint 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the constant velocity 15.0.

The running "MC_MoveVelocity" job (A2) is overridden by another "MC_MoveVelocity" job (A1). The abort is signaled via "Abort_2". The axis is accelerated to the new velocity setpoint 50.0. Before the velocity setpoint is reached, the current "MC_MoveVelocity" job A1) is overridden at time ③ by another "MC_MoveVelocity" job A2). The abort is signaled via "Abort_1". When the new velocity setpoint 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the constant velocity 15.0.

MC_MoveJog V3 (S7-1500, S7-1500T)

MC_MoveJog: Move axis in jog mode V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MoveJog", you move an axis in jog mode.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

- Positioning axis/synchronous axis:
  A velocity is specified in the "Velocity" parameter.
- Speed axis:
  A speed is specified in the "Velocity" parameter.

Applies to

- Synchronous axis
- Positioning axis
- Speed axis

Requirement

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The override response for "MC_MoveJog" jobs is described in section Override response of Motion Control commands V3 (Page 3345).

Parameters

The following table shows the parameters of the Motion Control instruction "MC_MoveJog":

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Technology object	
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".	
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".	
Velocity	INPUT	LREAL	100.0	Velocity setpoint / speed setpoint for the motion Value > 0.0: The specified value is used. Value < 0.0: The absolute value of the specified value is used. ("Velocity" = 0.0 is permitted)	
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used. Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
PositionControlled	INPUT	BOOL	TRUE	FALSE	Non position-controlled operation
				TRUE	Position-controlled mode
				The parameter applies as long as the "MC_MoveJog" job is being executed. After this, the setting of the following job applies. This parameter is ignored when a speed axis is used.	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity setpoint / speed setpoint was reached and will be maintained.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.

Parameters	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Behavior with velocity setpoint / speed setpoint zero ("Velocity" = 0.0)

An "MC_MoveJog" job with "Velocity" = 0.0 stops the axis with the configured deceleration. When the velocity setpoint / speed setpoint zero is reached, the parameter "InVelocity" will indicate the value TRUE.

Under "Technology object > Diagnostics > Status and error bits > Motion status", "constant velocity" and "standstill" will be displayed (<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill)).

Moving an axis in jog mode

Proceed as follows to move an axis in jog mode:

1. Check the requirements indicated above.
2. Move the axis in the positive direction with "JogForward", or in the negative direction with "JogBackward".

The current motion state is indicated in "Busy", "InVelocity" and "Error".

Note

Response to a change in the override

If the velocity / speed is influenced during constant motion by a change in the override (<TO>.Override.Velocity), the "InVelocity" parameter is reset during the acceleration or deceleration. When the newly calculated velocity is reached ("Velocity" × "Override" %), then "InVelocity" is set again.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

See also

Override response of Motion Control commands V3 (Page 3345)

MC_MoveJog: Function chart V3 (S7-1500, S7-1500T)

Function chart: Moving an axis in jog mode

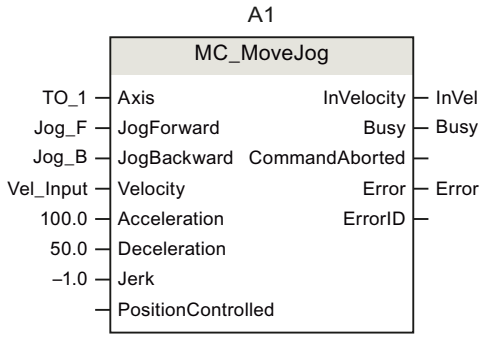

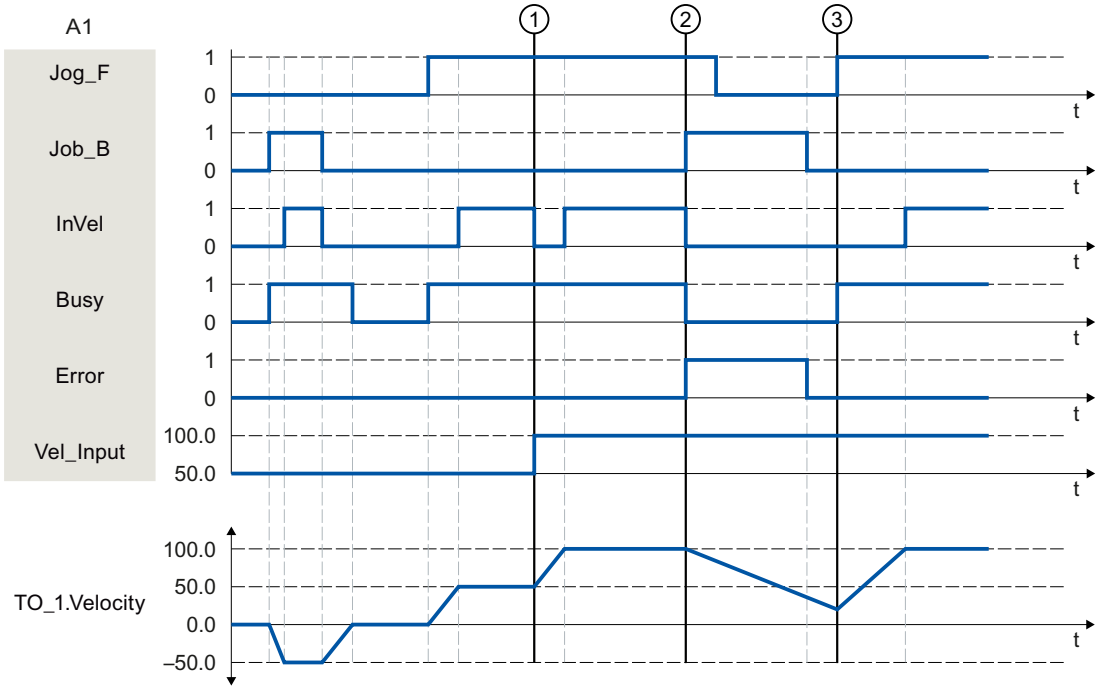

The axis is moved in the negative direction in jog mode via "Jog_B". When the velocity setpoint -50.0 is reached, this is signaled via "InVel" = TRUE. After "Jog_B" is reset, the axis is braked and brought to a standstill. Then the axis is moved in the positive direction via "Jog_F". When the velocity setpoint 50.0 is reached, this is signaled via "InVel" = TRUE.

At time ①, if "Jog_F" is set, the velocity setpoint is changed to 100.0 by means of "Vel_Input". Alternatively, you can also change the velocity setpoint using the velocity override. "InVel" is reset. Axis is being accelerated. When the new velocity setpoint 100.0 is reached, this is signaled via "InVel" = TRUE.

If "Jog_F" is set, "Jog_B" is likewise set at time ②. If both "Jog_F" and "Jog_B" are set, then the axis is braked with the last applicable deceleration. An error is indicated via "Error", and the "ErrorID" of the error 16#8007 (incorrect direction specification) is output.

This error is resolved by resetting the two inputs "Jog_F" and "Jog_B".

During the braking ramp, "Jog_F" is set at time ③. The axis is accelerated to the last configured velocity. When the velocity setpoint 100.0 is reached, this is signaled via "InVel" = TRUE.

MC_MoveSuperimposed V3 (S7-1500, S7-1500T)

MC_MoveSuperimposed: Superimposed positioning of axis V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_MoveSuperimposed", you start a relative positioning motion that is superimposed on a active basic motion.

You define the dynamic behavior of the motion with parameters "VelocityDiff", "Jerk", "Acceleration" and "Deceleration" . The dynamic values are added to the values of the basic motion. The duration of the basic motion is not extended by a superimposed motion.

The dynamics of the total axis motion is the sum of the dynamic values of the basic motion and the superimposed motion.

The behavior of the total motion depends on the type of basic motion:

- The basic motion is a single-axis motion:
  - The maximum dynamics of the superimposed motion is the difference between the current dynamic values of the basic motion and the dynamic limits.
  - The entire motion is limited to the configured dynamic limits.
- The basic motion is a synchronous motion:
  - The maximum dynamics of the superimposed motion is the difference between the current dynamic values of the basic motion and the dynamic limits.
  - The synchronous motion of the following axis is not limited to the dynamic limits of the following axis.
  - An "MC_MoveSuperimposed" job on a leading axis in synchronous operation affects the leading axis and the following axis.
  - An "MC_MoveSuperimposed" job on a following axis in synchronous operation only affects the following axis.

It is always the dynamics of the total motion that is displayed in the technology data block and in the TIA Portal.

Applies to

- Synchronous axis
- Positioning axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The override characteristics for "MC_MoveSuperimposed" jobs are described in section Override response of Motion Control commands V3 (Page 3345).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveSuperimposed":

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_PositioningAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
Distance	INPUT	LREAL	0.0	Additional travel distance for the superimposing positioning operation (negative or positive)	
VelocityDiff	INPUT	LREAL	-1.0	Maximum velocity deviation compared to the active motion Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)	
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	Superimposed positioning complete

Parameters	Declaration	Data type	Default value	Description	
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Starting superimposed positioning motion

To start a superimposed positioning motion with Motion Control instruction "MC_MoveSuperimposed", follow these steps:

1. Check the requirements indicated above.
2. Specify the additional distance to be moved in the "Distance" parameter.
3. Start the "MC_MoveSuperimposed" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error".

See also

Override response of Motion Control commands V3 (Page 3345)

MC_MoveSuperimposed: Function chart V3 (S7-1500, S7-1500T)

Function chart: Positioning axes overlapping

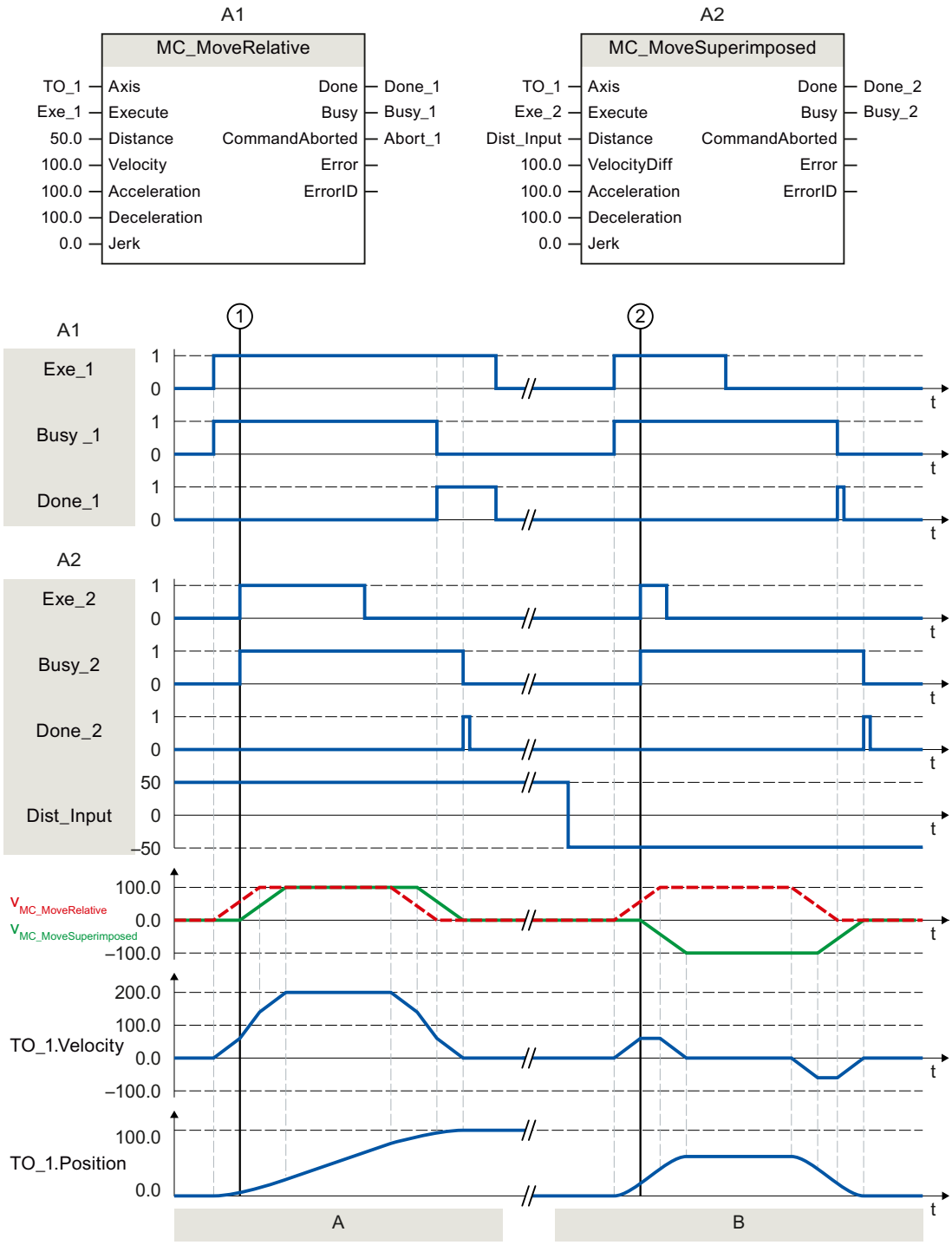

Section A	Using "Exe_1", an "MC_MoveRelative" job with a distance of 50.0 is initiated. At time ①, using "Exe_2", an MC_MoveSuperimposed job with a distance of 50.0 is initiated. The axis is moved with the added dynamic values of both jobs by the distance 50 + 50 = 100.0. When the axis reaches the target position, this is signaled via "Done_2".
Section B	Using "Exe_1", an "MC_MoveRelative" job with a distance of 50.0 is initiated. At time ②, using "Exe_2", a MC_MoveSuperimposed job with a distance of -50.0 is initiated. The axis reverses and is moved with the added dynamic values of both jobs by the distance 50.0 - 50.0 = 0.0. When the axis reaches the target position, this is signaled via "Done_2".

MC_SetSensor V3 (S7-1500T)

MC_SetSensor: Set alternative encoder as operationally active encoder V3 (S7-1500T)

Description

With the Motion Control instruction "MC_SetSensor", you switch over the encoder for closed loop position control of the axis.

The actual value of the addressed encoder can be adapted without switchover using parameter "Mode" 2 and 3.

Applies to

- Synchronous axis
- Positioning axis

Requirements

- The technology object and the alternative encoder have been configured correctly.
- No restart command and no MC-Home job running.

Override response

- An "MC_SetSensor" job is not aborted by any other Motion Control job.
- A new MC_SetSensor job does not abort any active Motion Control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_SetSensor":

Parameters	Declaration	Data type	Default value	Description
Axis	INPUT	TO_PositioningAxis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge

Parameters	Declaration	Data type	Default value	Description	
Sensor	INPUT	INT	1	Number of the new encoder (1 to 4).	
Mode	INPUT	DINT	0	The mode determines the position alignment between the old encoder and new encoder.	
				0	Switch over encoder and transfer current position to the new encoder With this encoder switchover, step changes in the positioning control are prevented. Bumpless switchover of the encoders is possible.
				1	Switch over encoder without aligning the actual position **Note**: When closed loop position control is active, an additional difference of the two encoders acts as additional control deviation and can trigger a compensating motion.
				2	Transfer actual value The current actual value is transferred to the encoder specified in the "Sensor" parameter.
				3	Transfer actual value The actual position of the "Reference encoder" ("ReferenceSensor" parameter) is transferred to the encoder specified in the "Sensor" parameter.
ReferenceSensor	INPUT	INT	1	Number of the reference encoder (see parameter "Mode" 3)	
Done	OUTPUT	BOOL	FALSE	TRUE	Encoder for closed loop position control of the axis was switched over.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job has been aborted.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Changing to absolute encoder

When you switch the encoder to an absolute encoder and transfer the actual value (Mode = 2, 3), the actual value is calculated with the value of the absolute encoder and the absolute value offset. When switching to a different encoder, calculation of the actual value is canceled. The absolute encoder once again returns the absolute value + absolute value offset (<TO>.StatusSensor[n].AbsEncoderOffset) without calculation by the MC_SetSensor.

Measuring input, output cam, cam track (S7-1500, S7-1500T)

MC_MeasuringInput V3 (S7-1500, S7-1500T)

MC_MeasuringInput: Start measuring once V3 (S7-1500, S7-1500T)

With the Motion Control instruction "MC_MeasuringInput", a one-time measuring is started.

With the one-time measuring, one or two edges can be detected with one measuring job. The position of the respective technology object axis or external encoder is assigned to the measuring event. The measurement result is indicated in the function block and in the technology data block and can be further processed in the user program. This completes the measuring job.

Another measuring job must be started again using "MC_MeasuringInput.Execute" = TRUE.

Applies to

- Measuring input

Requirements

- The technology object has been configured correctly.
- The encoder of the axis must have "valid" status (StatusSensor[1..4].State = valid). Otherwise, the measuring job is rejected in the function block with an error indication.
- Measuring using PROFIdrive is not possible during active or passive homing.

Override response

The override characteristics for "MC_MeasuringInput" jobs are described in section Override response of Motion Control commands V3 (Page 3345).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MeasuringInput":

Parameters	Declaration	Data type	Default value	Description
MeasuringInput	INPUT	TO_Measuring-Input	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge

Parameters	Declaration	Data type	Default value	Description	
Mode	INPUT	DINT	0	Type of measurement	
				0	Measurement of next positive edge
				1	Measurement of next negative edge
				2	Measurement of next two edges
				3	Measurement of two edges, beginning with the positive edge • Positive edge = "MeasuredValue1" (measured value 1) • Negative edge = "MeasuredValue2" (measured value 2)
				4	Measurement of two edges, beginning with the negative edge • Negative edge = "MeasuredValue1" • Positive edge = "MeasuredValue2"
MeasuringRange	INPUT	BOOL	FALSE	Acquisition of measured values Observe the time-related boundary conditions.	
				FALSE	Always acquire measured values
				TRUE	Acquire measured values only within the measuring range
StartPosition	INPUT	LREAL	0.0	Start position of the measuring range For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
EndPosition	INPUT	LREAL	0.0	End position of the measuring range For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
Done	OUTPUT	BOOL	FALSE	TRUE	The block has been completely processed. Measured values are valid.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The measuring job has been aborted.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	
MeasuredValue1	OUTPUT	LREAL	0.0	First measured value	
MeasuredValue2	OUTPUT	LREAL	0.0	Second measured value (for measurement of two edges)	

See also

Override response of Motion Control commands V3 (Page 3345)

MC_MeasuringInput: Function chart V3 (S7-1500, S7-1500T)

Function chart: Start measuring job once

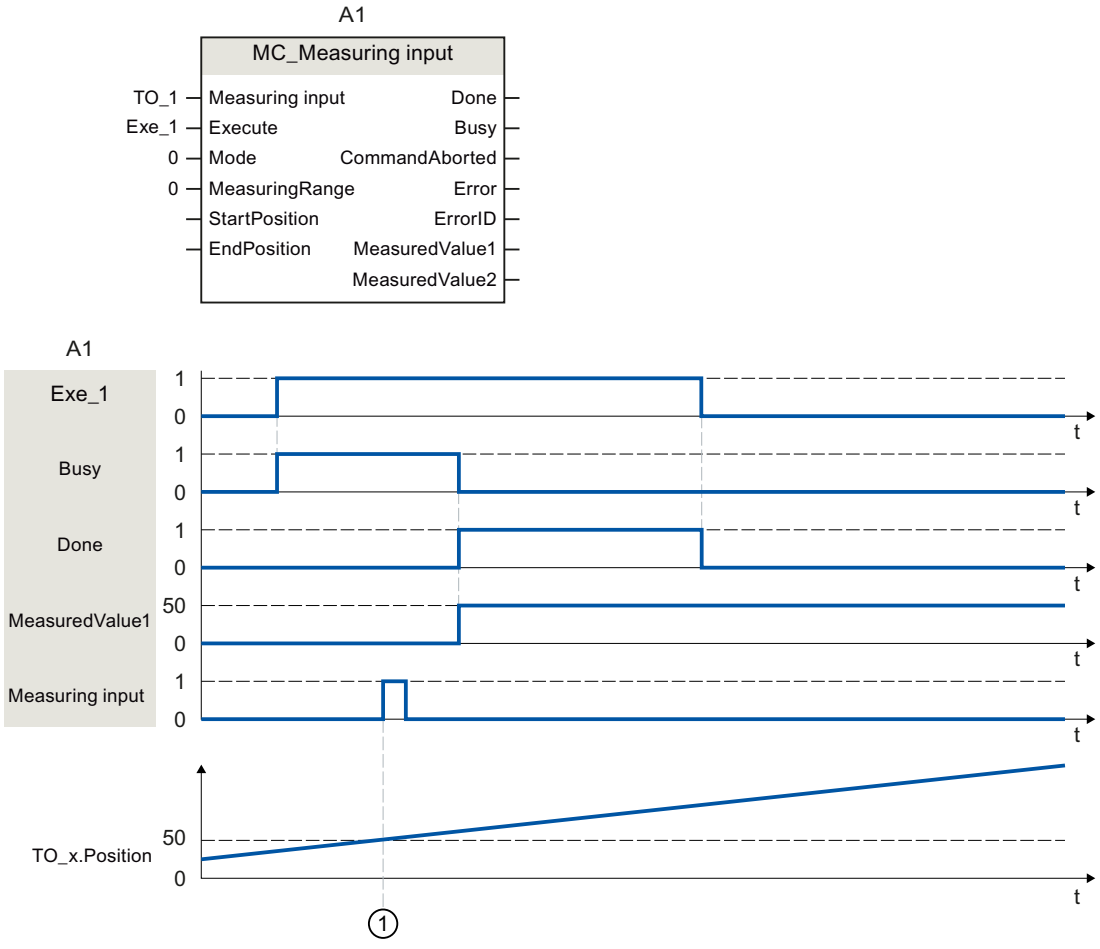

A "MC_MeasuringInput" job with "Mode" = 0 initiated using "Execute" performs a measurement of the next positive edge. A successfully completed measurement is signaled with "Done" = TRUE. The determined measured value ① (in example: 50) is output using "MeasuredValue1".

MC_MeasuringInputCyclic V3 (S7-1500, S7-1500T)

MC_MeasuringInputCyclic: Start cyclic measuring V3 (S7-1500, S7-1500T)

With the Motion Control instruction "MC_MeasuringInputCyclic", a cyclic measuring is started.

With cyclic measuring, up to two measuring events are detected by the system and the associated measuring positions displayed. The measurements are continued cyclically until they are ended per command.

Applies to

- Measuring input

Requirements

- The technology object has been configured correctly.
- Cyclic measuring is only possible when measuring using TM Timer DIDQ.
- The operationally active encoder of the axis must have "valid" status (StatusSensor[1..4].State = valid). Otherwise, the measuring job is rejected in the function block with an error indication.

Override response

The override characteristics for "MC_MeasuringInputCyclic" jobs are described in section Override response of Motion Control commands V3 (Page 3345).

Parameters

The following table shows the parameters of Motion Control instruction "MC_MeasuringInputCyclic":

Parameters	Declaration	Data type	Default value	Description	
MeasuringInput	INPUT	TO_Measuring-Input	-	Technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
Mode	INPUT	DINT	0	Type of measurement	
				0	Measurement of positive edges
				1	Measurement of negative edges
				2	Measurement of both edges
MeasuringRange	INPUT	BOOL	FALSE	Acquisition of measured values Observe the time-related boundary conditions.	
				FALSE	Always acquire measured values
				TRUE	Acquire measured values only within the measuring range
StartPosition	INPUT	LREAL	0.0	Start position of the measuring range For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
EndPosition	INPUT	LREAL	0.0	End position of the measuring range For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The measuring job has been aborted.

Parameters	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	
MeasuredValue1	OUTPUT	LREAL		First measured value	
MeasuredValue2	OUTPUT	LREAL		Second measured value (when measuring multiple edges in one position control cycle)	
MeasuredVal-ue1Counter	OUTPUT	UDINT	0	Count value for the first measured value	
MeasuredVal-ue2Counter	OUTPUT	UDINT	0	Count value for the second measured value	
LostEdgeCounter1	OUTPUT	UDINT	0	Count value for the lost edges in the cycle clock of the first measured value acquisition	
LostEdgeCounter2	OUTPUT	UDINT	0	Count value for the lost edges in the cycle clock of the second measured value acquisition	

See also

Override response of Motion Control commands V3 (Page 3345)

MC_MeasuringInputCyclic: Function chart V3 (S7-1500, S7-1500T)

Function chart: Start cyclic measuring job

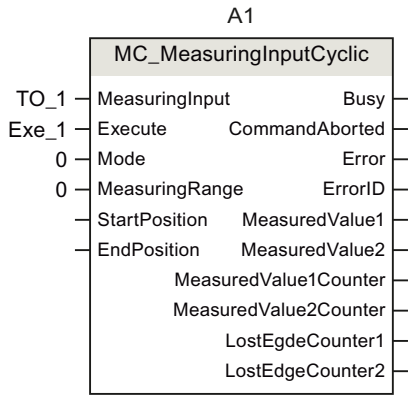

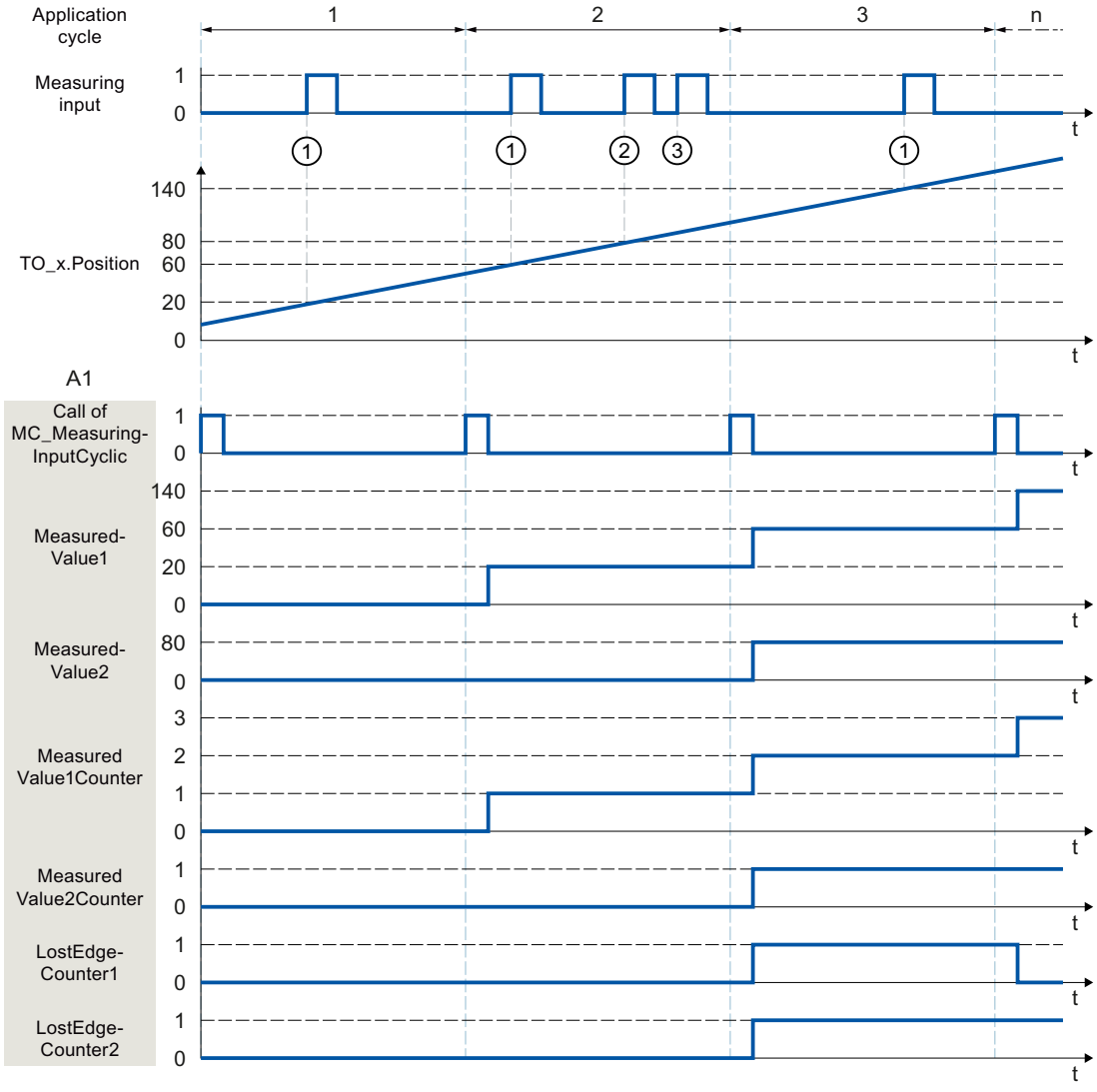

A "MC_MeasuringInputCyclic" command initiated using "Execute" (signal and start point not shown) (e.g. call in MC-PreServo [OB67]) with "Mode" = 0 and without specified measuring range "MeasuringRange" = FALSE performs a measurement of the positive edges. By calling

the Motion Control instruction "MC_MeasuringInputCyclic" in MC-PreServo [OB67] or MC-PreServo [OB67], the measuring command as well as the measurement itself is called in the same application cycle.

The actual position determined at the time of the first positive edge ① in the position control cycle is output using "MeasuredValue1", and count tag "MeasuredValue1Counter" is incremented by "1".

The actual position determined at the time of the second positive edge ② in the position control cycle is output using "MeasuredValue2", and count tag "MeasuredValue2Counter" is incremented by "1".

If additional positive edges occur in the same position control cycle ③, these are recorded in LostEdgeCounter1 and LostEdgeCounter2.

MC_AbortMeasuringInput V3 (S7-1500, S7-1500T)

MC_AbortMeasuringInput: Abort active measuring V3 (S7-1500, S7-1500T)

With the Motion Control instruction "MC_AbortMeasuringInput", an active one-time or cyclic measuring job is aborted.

Applies to

- Measuring input

Requirements

- The technology object has been configured correctly.

Override response

The override characteristics for "MC_AbortMeasuringInput" jobs are described in section Override response of Motion Control commands V3 (Page 3345).

Parameters

The following table shows the parameters of Motion Control instruction "MC_AbortMeasuringInput":

Parameters	Declaration	Data type	Default value	Description	
MeasuringInput	INPUT	TO_Measuring-Input	-	Technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
Done	OUTPUT	BOOL	0	The function block has been processed. The measuring job has been deactivated.	
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job has been aborted.

Parameters	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

See also

Override response of Motion Control commands V3 (Page 3345)

MC_OutputCam V3 (S7-1500, S7-1500T)

MC_OutputCam: Activate/deactivate cam V3 (S7-1500, S7-1500T)

Description

You use Motion Control instruction "MC_OutputCam" to activate the specified cam.

Depending on the cam type, the following input parameters are in effect:

- Distance output cam
  - "OnPosition"
  - "OffPosition"
- Time-based output cam
  - "OnPosition"
  - "Duration"

You define the operating mode and the effective direction of the cam with parameters "Mode" and "Direction".

When "MC_OutputCam.Enable" = TRUE, the input parameters are always read and take effect at the next position control cycle.

When a technological alarm occurs, the cam is processed again after error acknowledgment.

Applies to

- Output cam

Requirements

- The technology object has been configured correctly.
- The higher-level technology object must have a valid position.
- Setpoint output cams are not output for an axis in non-position-controlled operation.
- A setpoint output cam with the setting "MC_OutputCam.OnPosition" = 0 switches at position setpoint = 0.

Override response

The "MC_OutputCam" is canceled by:

- Disabling of the cam with "MC_OutputCam.Enable" = FALSE
- Only one instance of the "MC_OutputCam" function block can be active on an output cam in the user program. A second instance of the "MC_OutputCam" function block on a currently active output cam is rejected with an error.

Parameters

The following table shows the parameters of Motion Control instruction "MC_OutputCam":

Parameters	Declaration	Data type	Default value	Description	
OutputCam	INPUT	TO_OutputCam	-	Technology object	
Enable	INPUT	BOOL	FALSE	FALSE	Cam is disabled
				TRUE	Cam is being processed
OnPosition	INPUT	LREAL	0.0	Start position of the cam [unit of measurement of the interconnected technology object] For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
OffPosition	INPUT	LREAL	0.0	End position of the distance output cam [unit of measurement of the interconnected technology object] For an axis or external encoder with active modulo function, position data outside the modulo range is mirrored in the modulo range.	
Duration	INPUT	LREAL	0.0	Switch-on duration of the time-based output cam [unit of measurement: ms] The value for the switch-on duration of a time-based output cam must be greater than 0.0.	
Mode	INPUT	DINT	1	Operating mode	
				1	Standard Cam functionality (output not inverted)
				2	Cam functionality with inverted output
				3	Cam always ON (while "Enable" = TRUE)
Direction	INPUT	DINT	1	Activation direction of the cam	
				1	Positive direction
				2	Negative direction
				3	Both directions
CamOutput	OUTPUT	BOOL	-	Status display based on last call of Motion Control instruction "MC_OutputCam"	
Busy	OUTPUT	BOOL	FALSE	TRUE	Processing of the cam is active.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

MC_OutputCam: Function chart V3 (S7-1500, S7-1500T)

Function chart: Activate the specified cam type

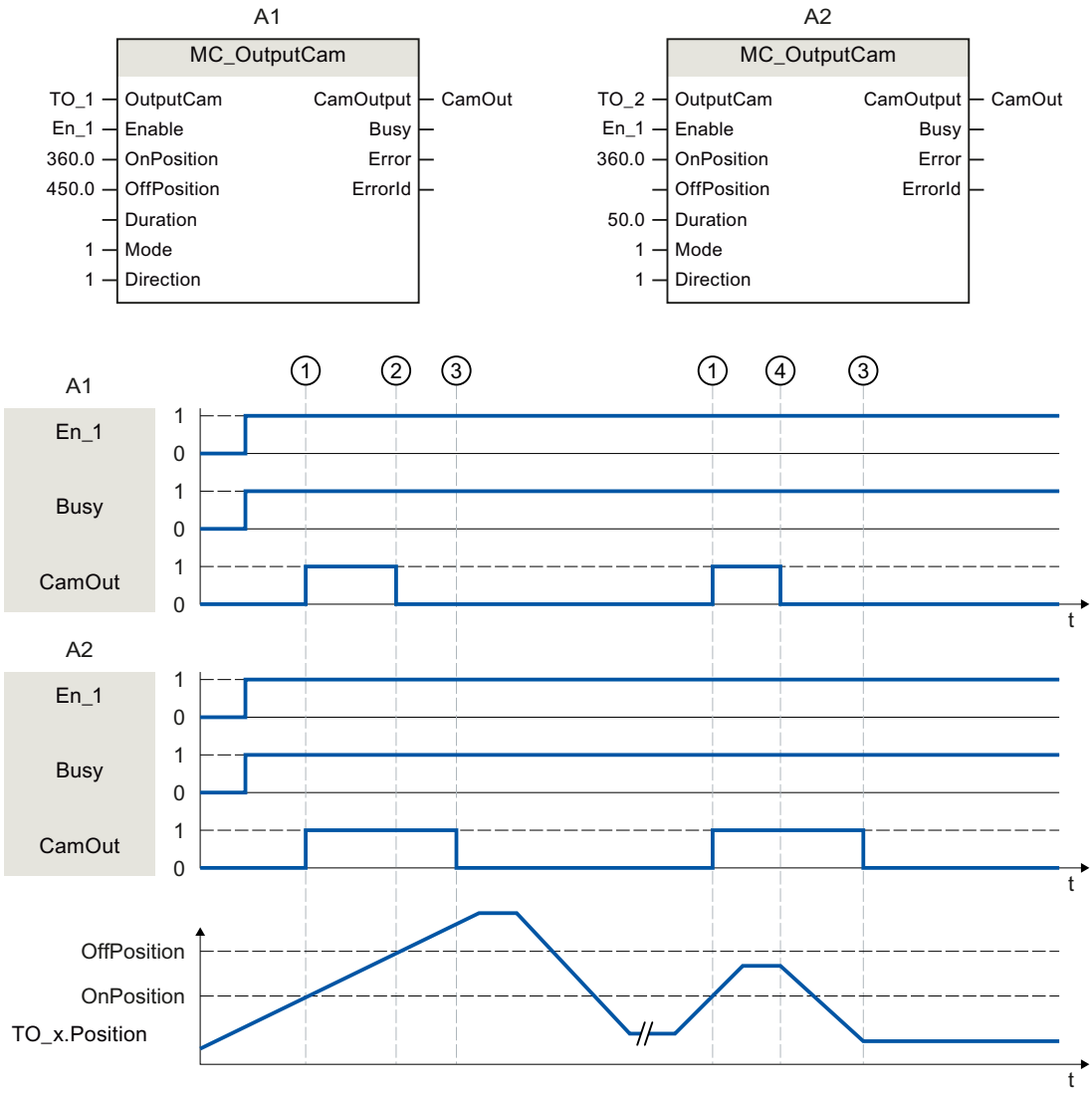

A1 Distance output cam

A2 Time-based output cam

A cam is activated with "Enable" = TRUE. The cam is output corresponding to the parameters. The processing of the cam is indicated with "Busy" = TRUE.

When activation direction "Direction" = 1 (positive) is set, the cam behaves as follows, regardless of the cam type setting:

- The distance output cam switches on at "OnPosition" ① and switches off again at "OffPosition" ②. At a direction reversal the cam is switched off ④.
- The time-based output cam switches on at "OnPosition" ① and switches off again after expiration of the specified time "Duration" ③. A switched on time-based output cam remains active for the specified switch-on duration "Duration", even if overtravel of the start position occurs again in the reverse direction.

The switching state of the cam is output using "CamOutput".

With "Enable" = FALSE, the specified cam type is deactivated. "Busy" = FALSE shows that the cam is not being processed.

MC_CamTrack V3 (S7-1500, S7-1500T)

MC_CamTrack: Activate/deactivate cam track V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_CamTrack", the processing of a cam track is enabled.

Applies to

- Cam track

Requirements

- The technology object has been configured correctly.
- The higher-level technology object must have a valid position.
- Setpoint output cams are not output for an axis in non-position-controlled operation.

Override response

- The "MC_CamTrack" is aborted by the disabling of the cam track with "MC_CamTrack.Enable" = FALSE.
- Only one instance of the "MC_CamTrack" function block can be active on a cam track in the user program. A second instance of the "MC_CamTrack" function block on a currently active cam track is rejected with an error.
- The cam track is recalculated at a change of the parameters of the Motion Control instruction "MC_CamTrack" or in the technology data block. The cam track is processed in accordance with all parameter settings.

Parameters

The following table shows the parameters of Motion Control instruction "MC_OutputCam":

Parameters	Declaration	Data type	Default value	Description		
CamTrack	INPUT	TO_CamTrack	-	Technology object		
Enable	INPUT	BOOL	FALSE	FALSE	Cam track is disabled	
				TRUE	Cam track is processed	
Mode	INPUT	INT	1	0	"Enable" = TRUE	Cam track processing is activated immediately Cam track data takes effect immediately Previously activated distance output cams are aborted if their track signal is not still set due to the changed cam track data. Previously activated time-based output cams are always aborted.
					"Enable" = FALSE	Cam track processing is stopped immediately, distance output/time-based output cams are aborted immediately.
				1	"Enable" = TRUE	Cam track processing is activated immediately/ with the next track cycle: • When a cam track is activated for the first time, the cam track processing begins immediately. • If the cam track processing is already active, the current cam track is output up to the end of the track cycle. The new cam track data then takes effect.
					"Enable" = FALSE	Cam track processing is ended at the end of the cam track
				2	"Enable" = TRUE	Cam track output is switched on immediately and remains switched on
					"Enable" = FALSE	Cam track output is switched off immediately

Parameters	Declaration	Data type	Default value	Description	
InvertOutput	INPUT	BOOL	FALSE	Inverted output	
				TRUE	Track output is output inverted
				FALSE	Track output is output not inverted
TrackOutput	OUTPUT	BOOL	-	Indicates the switching state of the cam track.	
Busy	OUTPUT	BOOL	FALSE	TRUE	Processing of the cam track is active.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

MC_CamTrack: Function chart V3 (S7-1500, S7-1500T)

Function chart: Activate cam track

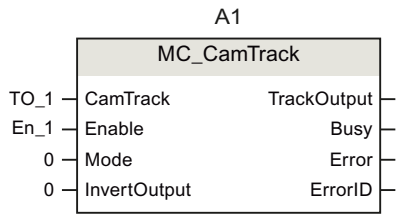

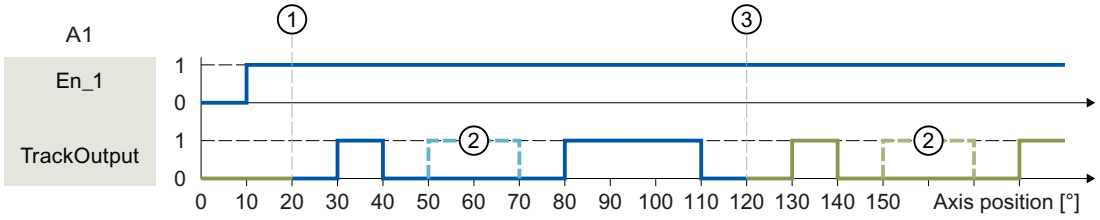

① Axis reference position
② Cam configured as invalid
③ Start of cyclically continued cam track

A cam track is activated with "Enable" = TRUE. The cam track is output in accordance with the parameters set in the technology data block.

Tag		Value	Description
<TO>Parameter.			
	CamTrackType	0	Specified cam type for the cam track is distance output cam
	ReferencePosition	20.0	Specified axis reference position for the cam track
	CamTrackLength	100.0	Specified length of the cam track
	CamTrack[1].Existent	TRUE	The first distance output cam of the cam track is specified as valid
	CamTrack[1].OnPosition	10.0	Start position for the first cam of the cam track
	CamTrack[1].OffPosition	20.0	End position for the first cam of the cam track
	CamTrack[2].Existent	FALSE	The second distance output cam of the cam track is specified as invalid
	CamTrack[2].OnPosition	30.0	Start position for the second cam of the cam track

Tag		Value	Description
	CamTrack[2].OffPosition	50.0	End position for the second cam of the cam track
	CamTrack[3].Existent	TRUE	The third distance output cam of the cam track is specified as valid
	CamTrack[3].OnPosition	60.0	Start position for the third cam of the cam track
	CamTrack[3].OffPosition	90.0	End position for the third cam of the cam track

Synchronous motion (S7-1500, S7-1500T)

MC_GearIn V3 (S7-1500, S7-1500T)

MC_GearIn: Start gear synchronization V3 (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_GearIn", you start a gearing operation between a leading axis and a following axis.

You define the dynamic behavior of the following axis for synchronization with parameters "Jerk", "Acceleration" and "Deceleration".

The synchronization duration and distance are dependent on the following parameters:

- Start time of the "MC_GearIn" job
- Dynamics of the following axis at the start time
- Dynamic settings for synchronization
- Dynamics of the leading axis

You specify the gear ratio as the relationship between two integers (numerator/denominator) with the parameters "RatioNumerator" and "RatioDenominator".

The numerator of the gear ratio is specified as positive or negative. This yields the following behavior:

- **Positive gear ratio:**
  The leading and following axes move in the same direction.
- **Negative gear ratio**:
  The following axis moves in the opposite direction of the leading axis.

You can start synchronous operation when the leading axis is at a standstill or when it is in motion.

Applies to

- Synchronous axis

Requirement

- The technology objects of the leading axis and the following axis have been configured correctly.
- The leading axis is a positioning axis, synchronous axis, or external encoder.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".
- The following axis is enabled.

Override response

The override response for "MC_GearIn" jobs are described in section Override response of Motion Control commands V3 (Page 3345).

Disabling the leading axis with "MC_Power.Enable" = FALSE does not abort synchronous operation. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_GearIn":

Parameters	Declaration	Data type	Default value	Description
Master	INPUT	TO_Axis	-	Leading axis technology object
Slave	INPUT	TO_SynchronousAxis	-	Following axis technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
RatioNumerator	INPUT	DINT	1	Gear ratio numerator Permitted integer values: -2147483648 to 2147483647 (value 0 not permitted)
RatioDenominator	INPUT	DINT	1	Gear ratio denominator Permitted integer values: 1 to 2147483647
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)

Parameters	Declaration	Data type	Default value	Description	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
InGear	OUTPUT	BOOL	FALSE	TRUE	Synchronous operation reached The following axis is synchronized and moves synchronously to the leading axis.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Starting synchronous operation

To start synchronous operation with the Motion Control instruction "MC_GearIn", follow these steps:

1. Check the requirements indicated above.
2. Specify the leading axis, the following axis and the gear ratio at the corresponding parameters.
3. Start the "MC_GearIn" job with a positive edge at parameter "Execute".

The following axis is synchronized to the leading value of the leading axis. If the "InGear" parameter shows the value TRUE, the following axis is synchronized and moves synchronously to the leading axis. The parameters "InGear" and "Busy" show the value TRUE until the "MC_GearIn" job is overridden by another Motion Control job.

See also

Override response of Motion Control commands V3 (Page 3345)

MC_GearIn: Function chart V3 (S7-1500, S7-1500T)

Function chart: Synchronizing and switching the leading value

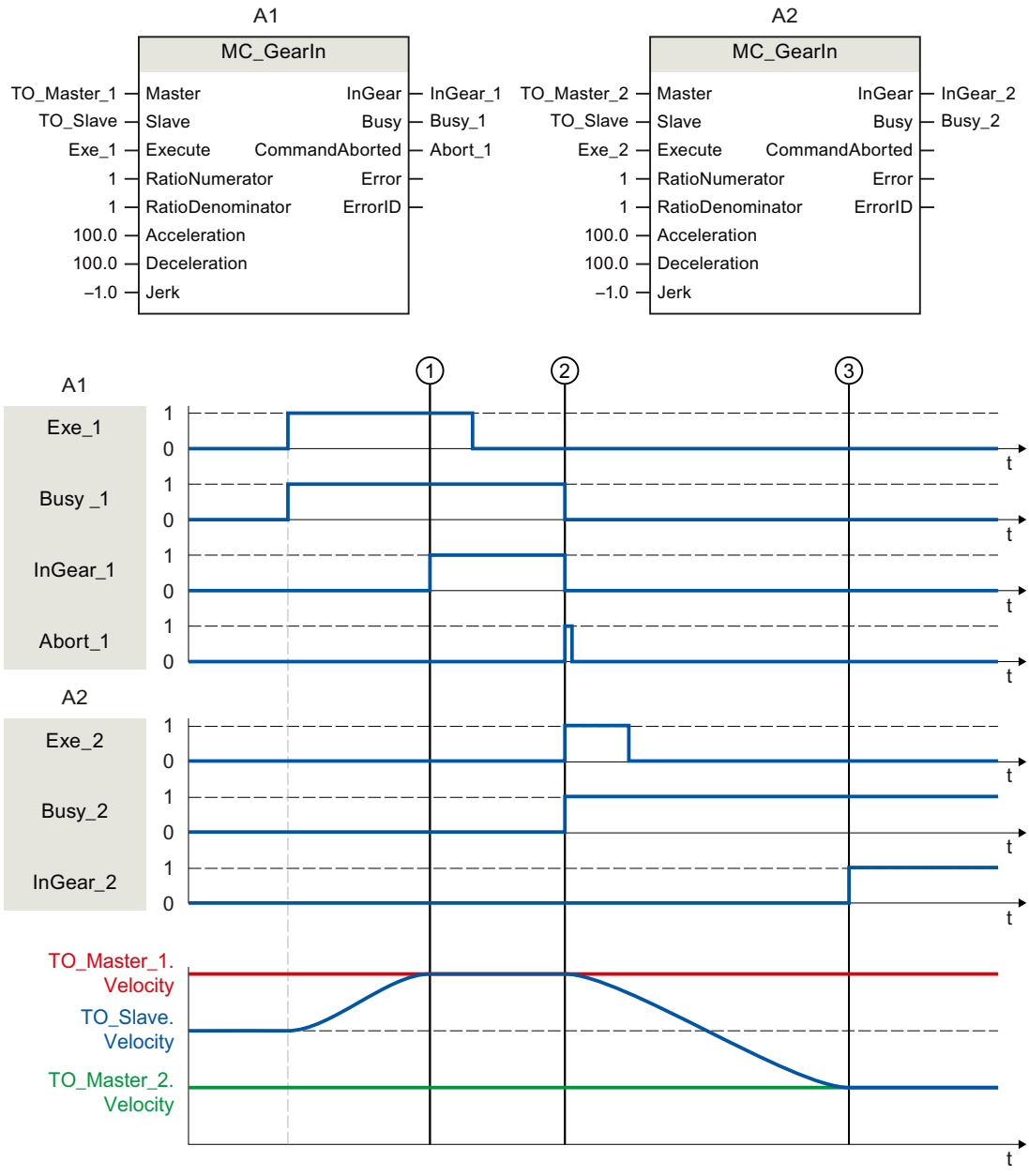

Using "Exe_1", an "MC_GearIn" job (A1) is initiated. The following axis (TO_Slave) is synchronized to the leading axis (TO_Master_1). "InGear_1" signals at time ① that the following axis is synchronized and moves synchronously to the leading axis.

At time ②, synchronous operation is overridden by another "MC_GearIn" job (A2). The abort is signaled via "Abort_1". The following axis is synchronized to the new leading axis (TO_Master_2). "InGear_2" signals at time ③ that the following axis is synchronized and moves synchronously to the leading axis.

MC_GearInPos V3 (S7-1500T)

MC_GearInPos: Start gearing with specified synchronous positions V3 (S7-1500T)

Description

With the Motion Control instruction "MC_GearInPos", you start a gearing operation between a leading axis and a following axis.

The synchronous operation is synchronized in advance of the specified synchronous position for the leading and following axis. The following types of synchronization are possible:

- **Synchronization using dynamic parameters**
  (SyncProfileReference = 0)
- **Synchronization using leading value distance**
  (SyncProfileReference = 1)

You specify the gear ratio as the relationship between two integers (numerator/denominator) with the parameters "RatioNumerator" and "RatioDenominator".

The numerator of the gear ratio is specified as positive or negative. This yields the following behavior:

- **Positive gear ratio:**
  The leading and following axes move in the same direction.
- **Negative gear ratio**:
  The following axis moves in the opposite direction of the leading axis.

You can start synchronous operation when the leading axis is at a standstill or when it is in motion.

Applies to

- Synchronous axis

Requirement

- The technology objects of the leading axis and the following axis have been configured correctly.
- The leading axis is a positioning axis, synchronous axis, or external encoder.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".
- The following axis is enabled.

Override response

The override response for "MC_GearInPos" jobs are described in section Override response of Motion Control commands V3 (Page 3345).

Disabling the leading axis with "MC_Power.Enable" = FALSE does not abort synchronous operation. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_GearInPos":

Parameters	Declaration	Data type	Default value	Description
Master	INPUT	TO_Axis	-	Leading axis technology object
Slave	INPUT	TO_SynchronousAxis	-	Following axis technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
RatioNumerator	INPUT	DINT	1	Gear ratio numerator Permitted integer values: -2147483648 to 2147483647 (value 0 not permitted)
RatioDenominator	INPUT	DINT	1	Gear ratio denominator Permitted integer values: 1 to 2147483647
MasterSyncPosition	INPUT	LREAL	0.0	Synchronous position of leading axis Position of the leading axis starting from which the leading and following axes move synchronously
SlaveSyncPosition	INPUT	LREAL	0.0	Synchronous position of following axis Position of the following axis starting from which the leading and following axes move synchronously
SyncProfileReference	INPUT	DINT	1	Type of synchronization 0: Synchronization using dynamic parameters 1: Synchronization using leading value distance
MasterStartDistance	INPUT	LREAL	1.0	Leading value distance (when "SyncProfileReference" = 1)
Velocity	INPUT	LREAL	-1.0	Velocity (when "SyncProfileReference" = 0) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)

Parameters	Declaration	Data type	Default value	Description	
Acceleration	INPUT	LREAL	-1.0	Acceleration (when "SyncProfileReference" = 0) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	Deceleration (when "SyncProfileReference" = 0) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk (when "SyncProfileReference" = 0) Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
SyncDirection	INPUT	DINT	3	Direction of synchronization (in effect for axes with activated Modulo setting)	
				1	Positive direction The following axis may only travel in positive direction during synchronization.
				2	Negative direction The following axis may only travel in negative direction during synchronization.
				3	Shortest distance Changes in direction are permitted for the following axis during synchronization.
StartSync	OUTPUT	BOOL	FALSE	TRUE	The following axis is synchronized to the leading axis.
InSync	OUTPUT	BOOL	FALSE	TRUE	Synchronous operation reached The following axis is synchronized and moves synchronously to the leading axis.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.

Parameters	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Starting synchronous operation

To start synchronous operation with the Motion Control instruction "MC_GearInPos", follow these steps:

1. Check the requirements indicated above.
2. Specify the leading axis, the following axis, the gear ratio and the synchronous position with the corresponding parameters.
3. Start the "MC_GearInPos" job with a positive edge at parameter "Execute".

The following axis is synchronized to the leading value of the leading axis. If the "InSync" parameter shows the value TRUE, the following axis is synchronized and moves synchronously to the leading axis. The parameters "InSync" and "Busy" show the value TRUE until the "MC_GearInPos" job is overridden by another Motion Control job.

See also

Override response of Motion Control commands V3 (Page 3345)

MC_GearInPos: Function chart V3 (S7-1500T)

Function chart: Synchronization using dynamic parameters/leading value distance

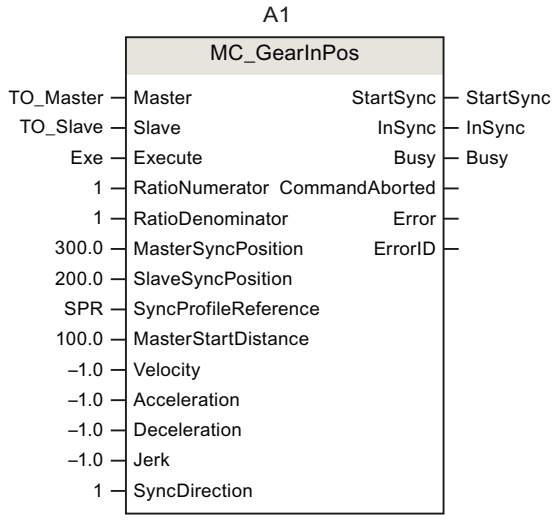

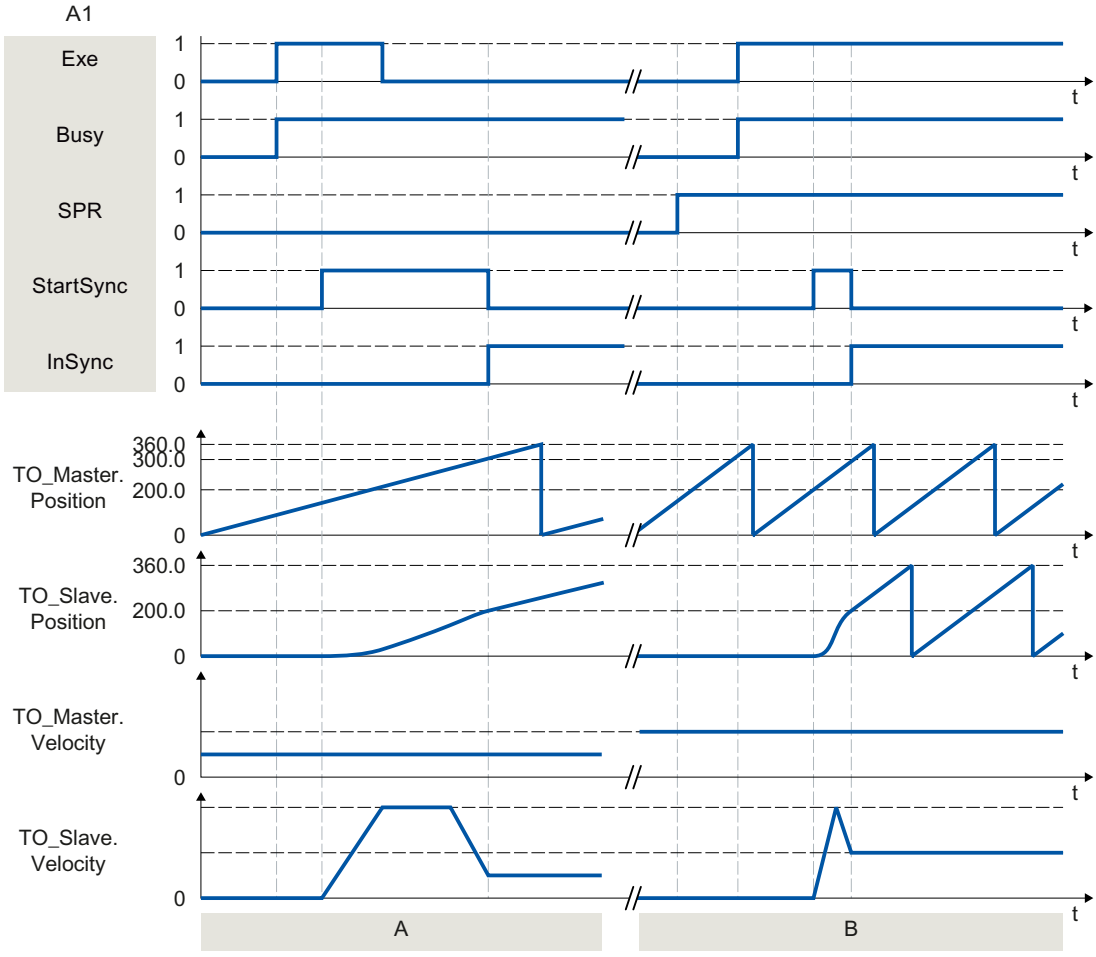

Section A	Using "Exe", an "MC_GearInPos" job (A1) is initiated. The following axis (TO_Slave) is synchronized to the leading axis (TO_Master) by means of the specified dynamic parameters. The distance required for synchronization is calculated by the system. The start of the synchronization is displayed with StartSync. When the specified synchronous position is reached, "InSync" signals that the following axis is synchronized and moving synchronously to the leading axis.
Section B	Using "Exe", an "MC_GearInPos" job (A1) is initiated. The following axis (TO_Slave) is synchronized to the leading axis (TO_Master) by means of the specified leading value distance. The dynamics required for synchronization is calculated by the system. The start of the synchronization is displayed with StartSync. "InSync" signals that the following axis is synchronized and moving synchronously to the leading axis.

MC_PhasingAbsolute V3 (S7-1500T)

MC_PhasingAbsolute: Absolute shift of leading value on the following axis V3 (S7-1500T)

Description

With the Motion Control instruction "MC_PhasingAbsolute", you shift the leading value on a following axis during gearing with "MC_GearIn" and "MC_GearInPos" as an absolute shift. The position of the leading axis is not affected by this.

You define the dynamic response of the motion of the following axis with the parameters "Velocity", "Jerk", "Acceleration", and "Deceleration". The dynamic values are added to the values of the synchronous operation motion.

Applies to

- Synchronous axis

Requirements

- The technology objects of the leading axis and the following axis have been configured correctly.
- The leading axis is a positioning axis, synchronous axis, or external encoder.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".
- The following axis is synchronized to the leading axis ("MC_GearIn.InGear" = TRUE or "MC_GearInPos.InSync" = TRUE) by means of the Motion Control instruction "MC_GearIn" or "MC_GearInPos".
- The following axis is enabled.

Override response

The override characteristics for "MC_PhasingAbsolute" jobs are described in section Override response of Motion Control commands V3 (Page 3345).

Disabling the leading axis with "MC_Power.Enable" = FALSE does not abort the leading value shift. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_PhasingAbsolute":

Parameters	Declaration	Data type	Default value	Description
Master	INPUT	TO_Axis	-	Leading axis technology object
Slave	INPUT	TO_SynchronousAxis	-	Following axis technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
PhaseShift	INPUT	LREAL	0.0	Absolute leading value shift
Velocity	INPUT	LREAL	-1.0	Velocity of the following axis for leading value shift (added to synchronous operation motion) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration of the following axis for leading value shift (added to synchronous operation motion) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration of the following axis for leading value shift (added to synchronous operation motion) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk of the following axis for leading value shift (added to synchronous operation motion) Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
Done	OUTPUT	BOOL	FALSE	TRUE: Leading value shift is finished.

Parameters	Declaration	Data type	Default value	Description	
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	
AbsolutePhaseShift	OUTPUT	LREAL	0.0	As long as "Busy" = TRUE: Display of leading value shift completed up to now	

Start absolute leading value shift

To start an absolute leading value shift with the Motion Control instruction "MC_PhasingAbsolute", follow these steps:

1. Check the requirements indicated above.
2. Specify the absolute leading value shift in the "PhaseShift" parameter.
3. Start the "MC_PhasingAbsolute" job with a positive edge at parameter "Execute".

The current motion state is indicated in parameters "Busy", "AbsolutePhaseShift", "Done" and "Error".

See also

Override response of Motion Control commands V3 (Page 3345)

MC_GearInPos V3 (Page 3312)

MC_GearIn V3 (Page 3308)

MC_PhasingAbsolute: Function chart V3 (S7-1500T)

Function chart: Absolute shift of leading value

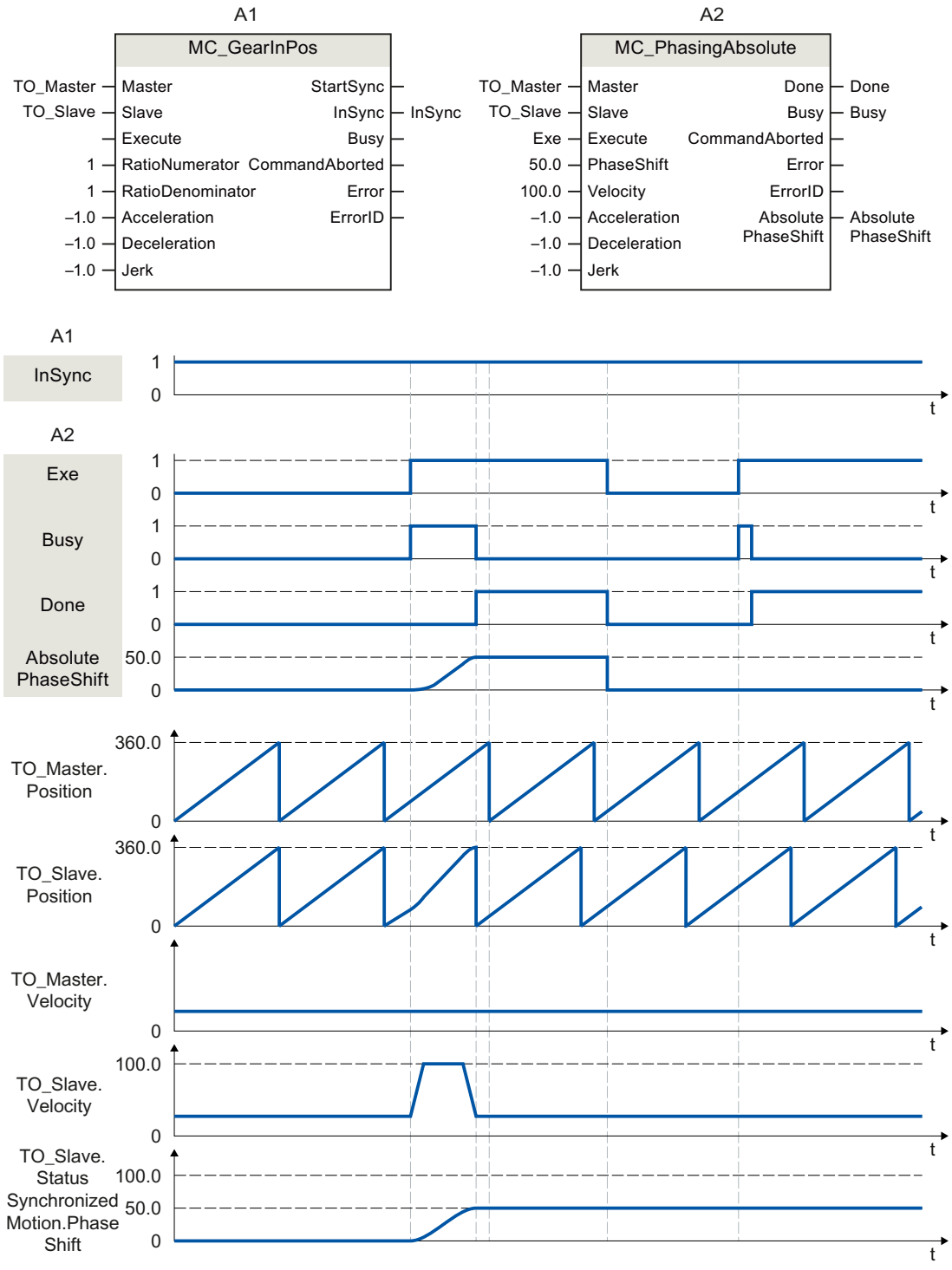

During an active gearing operation with "MC_GearInPos" (A1), a "MC_PhasingAbsolute" job (A2) is initiated using "Exe". The leading value shift is performed with the dynamics specified additively to the synchronous operation motion. "Done" indicates that the leading value was successfully shifted. The leading value shift 50.0 resulting from the job is indicated in "AbsolutePhaseShift". The absolute leading value shift 50.0 is indicated in the tag of the technology object <TO>.StatusSynchronizedMotion.PhaseShift. The motion of the leading axis is not affected.

After the leading value shift, the "MC_PhasingAbsolute" job (A2) is initiated again using "Exe". Because the leading value shift (<TO>.StatusSynchronizedMotion.PhaseShift) is already 50.0, the leading value is not shifted.

MC_PhasingRelative V3 (S7-1500T)

MC_PhasingRelative: Relative shift of leading value on the following axis V3 (S7-1500T)

Description

With the Motion Control instruction "MC_PhasingRelative", you shift the leading value on a following axis during gearing with "MC_GearIn" and "MC_GearInPos" relative to the existing leading value shift. The position of the leading axis is not affected by this.

You define the dynamic response of the motion of the following axis with the parameters "Velocity", "Jerk", "Acceleration", and "Deceleration" . The dynamic values are added to the values of the synchronous operation motion.

Applies to

- Synchronous axis

Requirements

- The technology objects of the leading axis and the following axis have been configured correctly.
- The leading axis is a positioning axis, synchronous axis, or external encoder.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".
- The following axis is synchronized to the leading axis ("MC_GearIn.InGear" = TRUE or "MC_GearInPos.InSync" = TRUE) by means of the Motion Control instruction "MC_GearIn" or "MC_GearInPos".
- The following axis is enabled.

Override response

The override characteristics for "MC_PhasingRelative" jobs are described in section Override response of Motion Control commands V3 (Page 3345).

Disabling the leading axis with "MC_Power.Enable" = FALSE does not abort the leading value shift. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_PhasingRelative":

Parameters	Declaration	Data type	Default value	Description
Master	INPUT	TO_Axis	-	Leading axis technology object
Slave	INPUT	TO_SynchronousAxis	-	Following axis technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
PhaseShift	INPUT	LREAL	0.0	Relative leading value shift
Velocity	INPUT	LREAL	-1.0	Velocity of the following axis for leading value shift (added to synchronous operation motion) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration of the following axis for leading value shift (added to synchronous operation motion) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration of the following axis for leading value shift (added to synchronous operation motion) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk of the following axis for leading value shift (added to synchronous operation motion) Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
Done	OUTPUT	BOOL	FALSE	TRUE: Leading value shift is finished.

Parameters	Declaration	Data type	Default value	Description	
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	
CoveredPhaseShift	OUTPUT	LREAL	0.0	As long as "Busy" = TRUE: Display of leading value shift completed up to now	

Start relative leading value shift

To start a relative leading value shift with the Motion Control instruction "MC_PhasingRelative", follow these steps:

1. Check the requirements indicated above.
2. Specify the relative leading value shift in the "PhaseShift" parameter.
3. Start the "MC_PhasingRelative" job with a positive edge at parameter "Execute".

The current motion state is indicated in parameters "Busy", "CoveredPhaseShift", "Done" and "Error".

See also

Override response of Motion Control commands V3 (Page 3345)

MC_GearInPos V3 (Page 3312)

MC_GearIn V3 (Page 3308)

MC_PhasingRelative: Function chart V3 (S7-1500T)

Function chart: Relative shift of leading value

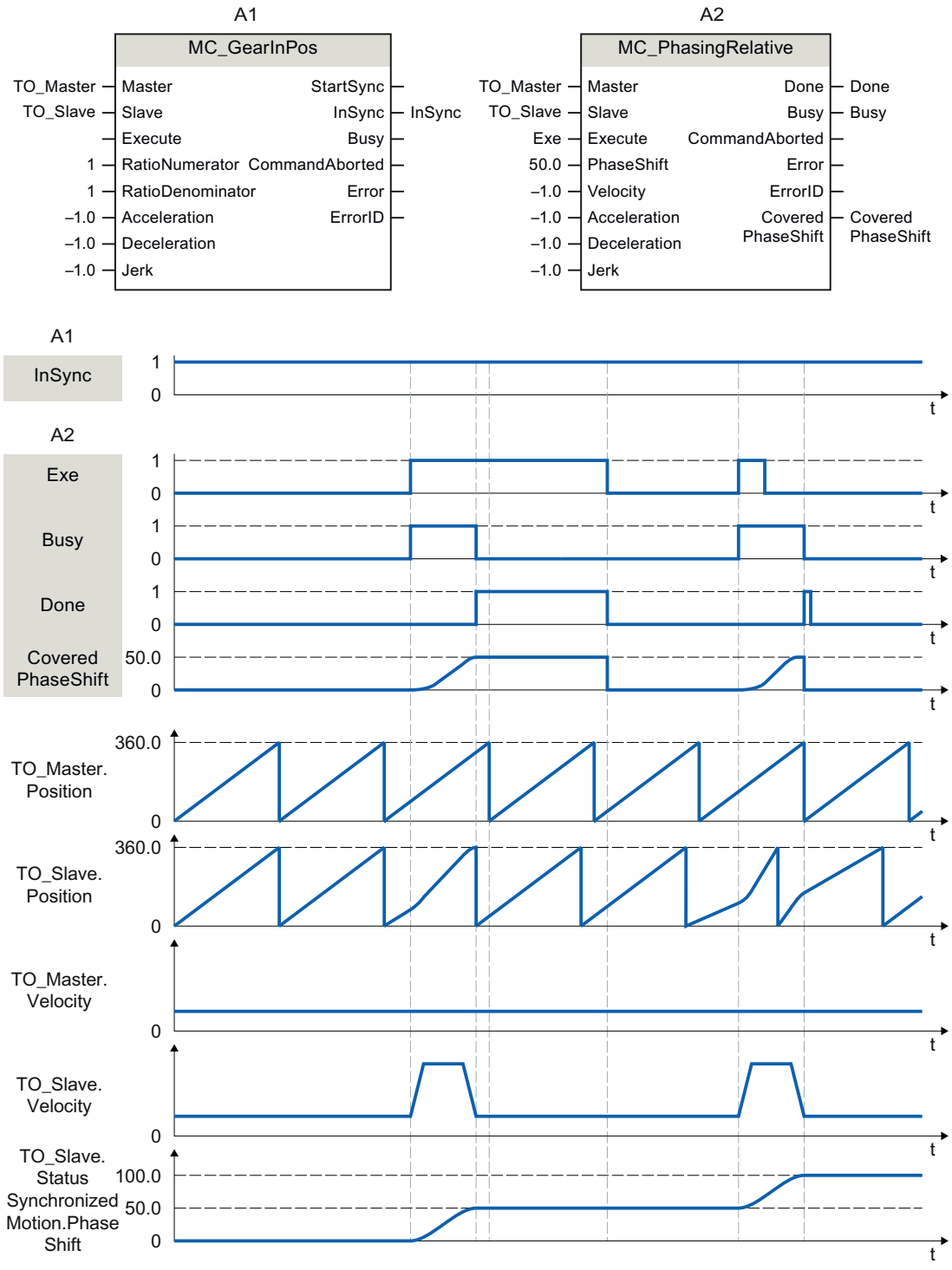

During an active gearing operation with "MC_GearInPos" (A1), a "MC_PhasingRelative" job (A2) is initiated using "Exe". The leading value shift is performed with the dynamics specified additively to the synchronous operation motion. "Done" indicates that the leading value was successfully shifted. The leading value shift 50.0 resulting from the job is indicated in "CoveredPhaseShift". The absolute leading value shift 50.0 is indicated in the tag of the technology object <TO>.StatusSynchronizedMotion.PhaseShift. The motion of the leading axis is not affected.

After the leading value shift, the "MC_PhasingRelative" job (A2) is initiated again using "Exe". The leading value shift is performed again with the dynamics specified additively to the synchronous operation motion. "Done" indicates that the leading value was successfully shifted. The leading value shift 50.0 resulting from the job is indicated in "CoveredPhaseShift". The absolute leading value shift 100.0 is indicated in the tag of the technology object <TO>.StatusSynchronizedMotion.PhaseShift.

MC_CamIn V3 (S7-1500T)

MC_CamIn: Start camming V3. (S7-1500T)

Description

With the Motion Control instruction "MC_CamIn", you start a camming operation between a leading axis and a following axis.

The synchronous operation is synchronized in advance of the specified synchronous position of the leading axis. The synchronous position of the leading axis starting from which the leading and following axes move synchronously results from the starting position of the cam and the "MasterSyncPosition" and "MasterOffset" parameters.

With the "MasterSyncPosition" parameter, you specify the position in the cam relative to the start of the cam starting from which the cam is to be run. If the entire cam is to be run, specify the value 0.0 (default value) in "MasterSyncPosition".

With the "MasterOffset" parameter, you offset the leading values of the cam. As a result, you offset the curve defined as an absolute curve to the position range needed for the application.

The figure below shows the basic effect of the leading value and following value offset as well as of "MasterSyncPosition":

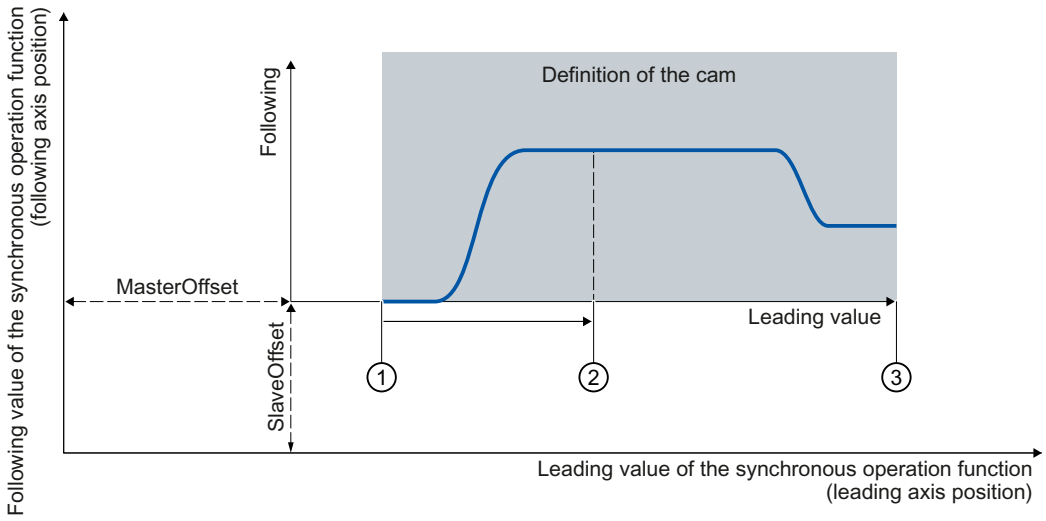

The figure below shows the basic effect of scaling the cam with the parameters "MasterScaling" and "SlaveScaling":

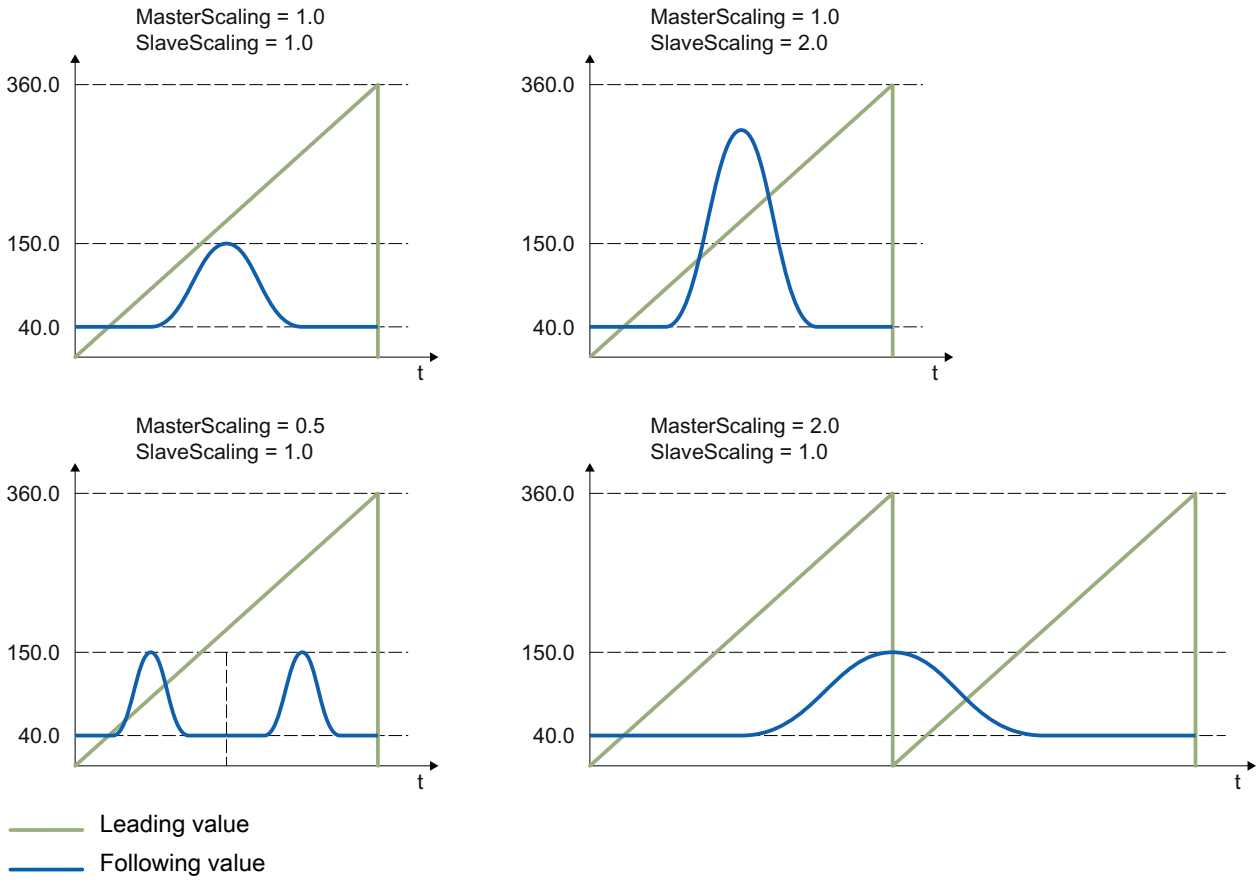

The following types of synchronization are possible:

- **Synchronization using dynamic parameters**
  (SyncProfileReference = 0)
- **Synchronization using leading value distance**
  (SyncProfileReference = 1)

You can start synchronous operation when the leading axis or following axis is at a standstill or when it is in motion.

Applies to

- Synchronous axis

Requirement

- The technology objects of the leading axis, following axis, and cam have been configured correctly.
- The leading axis is a positioning axis, synchronous axis, or external encoder.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".
- The following axis is enabled.
- The cam is interpolated with "MC_InterpolateCam".

Override response

The override response for "MC_CamIn" jobs is described in section Override response of Motion Control commands V3 (Page 3345).

Disabling the leading axis with "MC_Power.Enable" = FALSE does not abort synchronous operation. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_CamIn":

Parameter	Declaration	Data type	Default value	Description
Master	INPUT	TO_Axis	-	Leading axis technology object
Slave	INPUT	TO_SynchronousAxis	-	Following axis technology object
Cam	INPUT	TO_Cam	-	Cam technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
MasterOffset	INPUT	LREAL	0.0	Offset of leading values of cam (The cam technology object is not changed.)

Parameter	Declaration	Data type	Default value	Description
SlaveOffset	INPUT	LREAL	0.0	Offset of following values of cam (The cam technology object is not changed.)
MasterScaling	INPUT	LREAL	1.0	Scaling the leading values of the cam (The cam technology object is not changed.)
SlaveScaling	INPUT	LREAL	1.0	Scaling the following values of the cam (The cam technology object is not changed.)
MasterSyncPosition	INPUT	LREAL	0.0	Starting point in the cam/distance from the cam start with reference to the scaled cam. The value must be within the leading value range of the cam.
SyncProfileReference	INPUT	DINT	1	Synchronization profile 0: Synchronization using dynamic parameters 1: Synchronization using leading value distance
MasterStartDistance	INPUT	LREAL	0.0	Leading value distance (when "SyncProfileReference" = 1)
Velocity	INPUT	LREAL	-1.0	Velocity (when "SyncProfileReference" = 0) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration (when "SyncProfileReference" = 0) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration (when "SyncProfileReference" = 0) Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)

Parameter	Declaration	Data type	Default value	Description	
Jerk	INPUT	LREAL	-1.0	Jerk (when "SyncProfileReference" = 0) Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
ApplicationMode	INPUT	DINT	0	Application of the cam	
				0	Once/not cyclic
				1	Cyclic (absolute application on the following value side)
				2	Cyclic appending (continuously appending on the following value side)
SyncDirection	INPUT	DINT	3	Direction of synchronization (in effect for axes with activated Modulo setting)	
				1	Positive direction The following axis may only travel in positive direction during synchronization.
				2	Negative direction The following axis may only travel in negative direction during synchronization.
				3	Shortest distance Changes in direction are permitted for the following axis during synchronization.
StartSync	OUTPUT	BOOL	FALSE	TRUE	The following axis is synchronized to the leading axis.
InSync	OUTPUT	BOOL	FALSE	TRUE	Synchronous operation reached The following axis is synchronized and moves synchronously to the leading axis.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	
EndOfProfile	OUTPUT	BOOL	FALSE	TRUE	The end of the cam has been reached. Displayed for at least one call of "MC_CamIn" in the user program when the cam is used cyclically.

Starting synchronous operation

To start synchronous operation with the Motion Control instruction "MC_CamIn", follow these steps:

1. Check the requirements indicated above.
2. Specify the leading axis, the following axis, the utilized cam and the synchronous position in the corresponding parameters.
3. Start the "MC_CamIn" job with a positive edge at parameter "Execute".

The following axis is synchronized to the leading value of the leading axis. If the "InSync" parameter shows the value TRUE, the following axis is synchronized and moves synchronously to the leading axis. With cyclic application of the cam, "InSync" and "Busy" show the value TRUE until the "MC_CamIn" job is overridden by another Motion Control job. With non-cyclic application of the cam, "InSync" and "Busy" are set to the value FALSE when the parameter "EndOfProfile" is set to the value TRUE.

See also

Override response of Motion Control commands V3 (Page 3345)

MC_CamIn: Function chart V3 (S7-1500T)

Function chart: Synchronizing and switching the cam

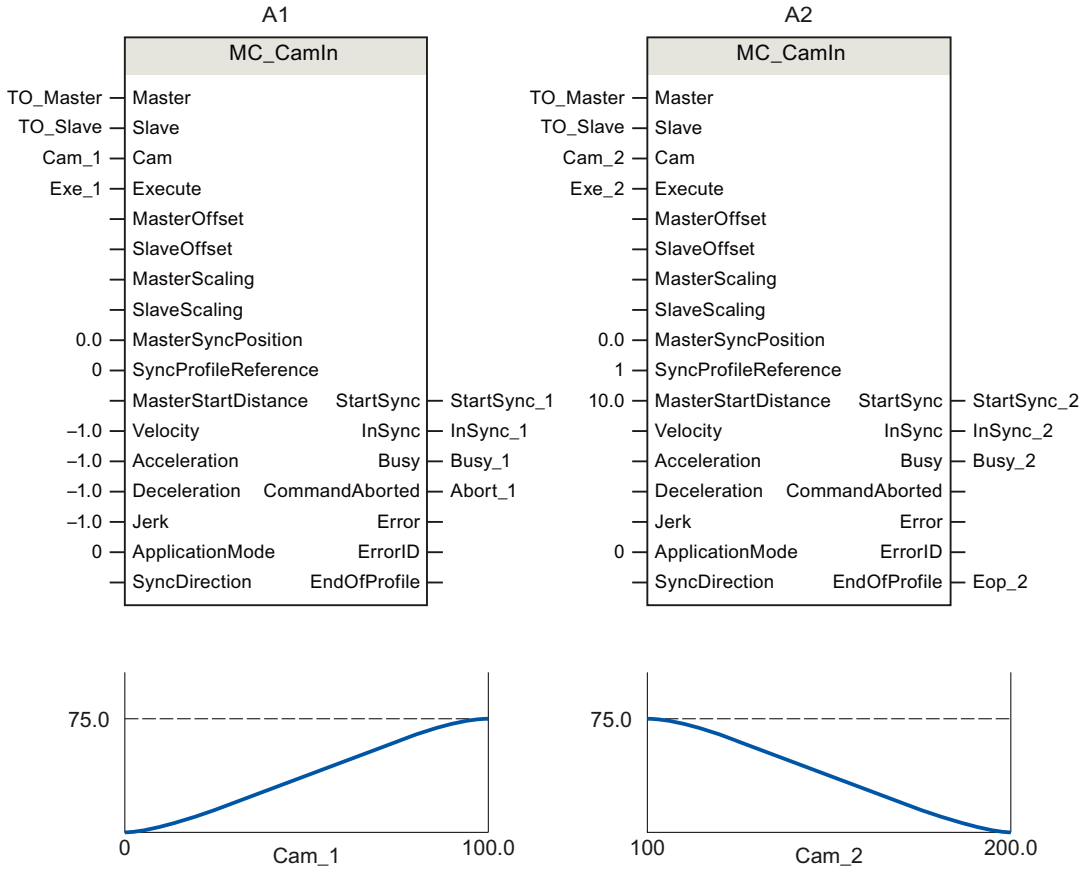

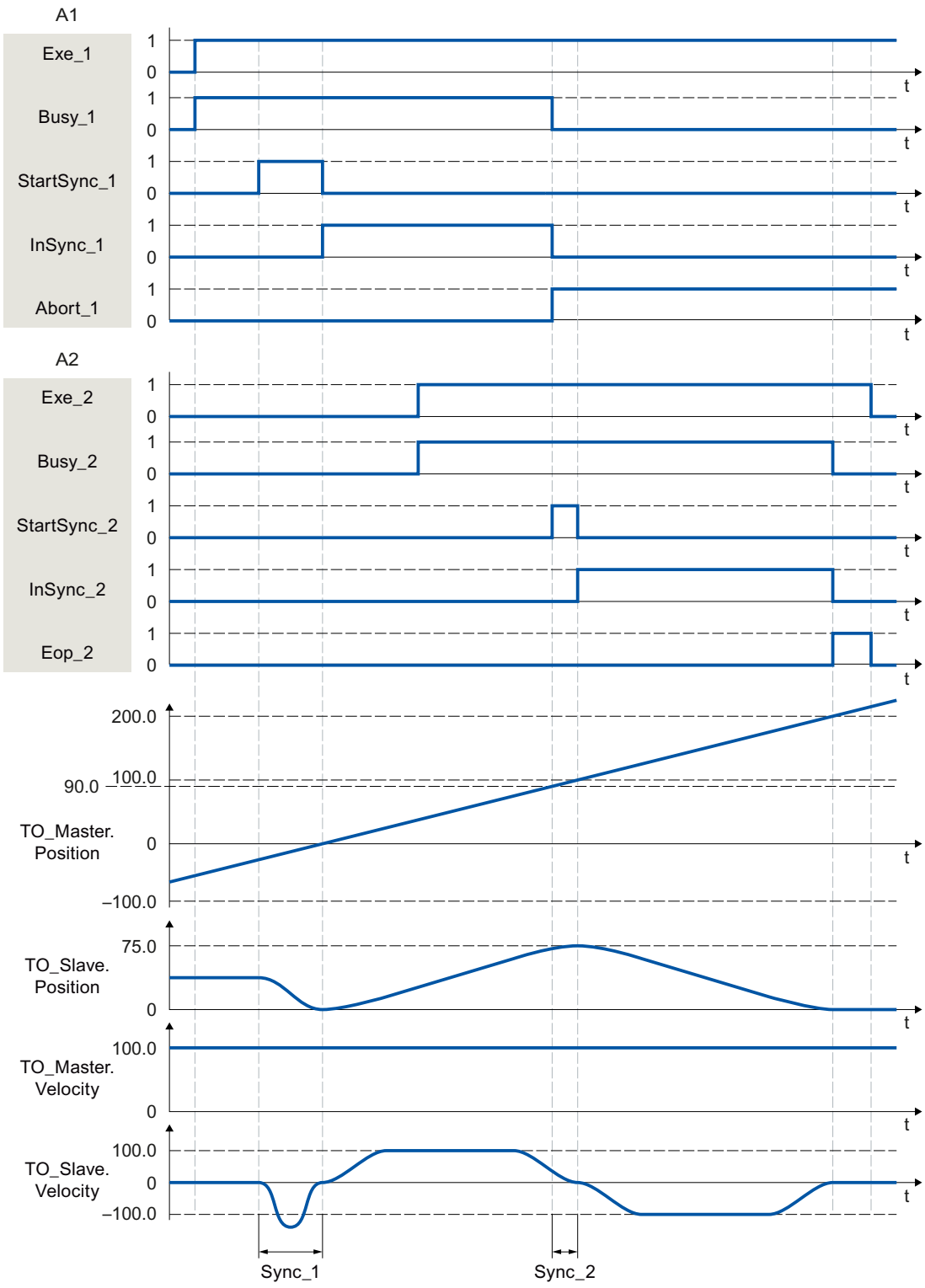

Using "Exe_1", an "MC_CamIn" job (A1) is initiated. The start of the synchronization is displayed with StartSync_1. The following axis (TO_Slave) is synchronized to the cam (Cam_1) within the range "Sync_1" by means of the specified dynamic parameters. "InSync_1" signals that the following axis is synchronized and moving synchronously to the leading axis.

The synchronous operation is overridden by another "MC_CamIn" job (A2). The abort is signaled via "Abort_1". The start of the synchronization is displayed with StartSync_2. The following axis is synchronized to the new cam (Cam_2) within the range "Sync_2" by means of the specified leading value distance. Within the range "Sync_2", the axis does not following the cam "Cam_1" The axis is moved with the dynamics calculated by the system to the synchronous position of A2. "InSync_2" signals that the following axis is synchronized and moving synchronously to the leading axis.

MC_SynchronizedMotionSimulation V3 (S7-1500T)

MC_SynchronizedMotionSimulation: Simulate synchronous operation V3 (S7-1500T)

Description

With the Motion Control instruction "MC_SynchronizedMotionSimulation", you simulate an active synchronous operation on a following axis. As a result, a synchronous operation remains active when the following axis is disabled with "MC_Power". The following axis does not have to be synchronized again after being enabled again.

With the start of a "MC_SynchronizedMotionSimulation" job, the velocity setpoint from the synchronous operation is set to zero. If an overlaid movement is active on the following axis at the start of the simulation, the setpoints of this overlaid movement will continue to be output.

Setpoints of motion jobs that are started during synchronous operation simulation are output to the drive.

If the position of the following axis at the end of the simulation differs from the position at the start of the simulation, this triggers a setpoint step-change.

Applies to

- Synchronous axis

Requirement

- The technology object has been configured correctly.
- The following axis is a synchronous axis.
- A synchronous operation is active on the technology object in status "Synchronous" (<TO>.StatusWord.X22 = TRUE).

Override response

The "MC_SynchronizedMotionSimulation" job is not aborted by any other Motion Control job. The simulated synchronous operation remains active even when the following axis is disabled with "MC_Power.Enable" = FALSE.

A restart of the technology object stops the simulation and aborts the synchronous operation.

The start of an "MC_SynchronizedMotionSimulation" job does not abort any other Motion Control jobs. With "MC_SynchronizedMotionSimulation.Enable" = TRUE, synchronous operation jobs are rejected.

Parameters

The following table shows the parameters of Motion Control instruction "MC_SynchronizedMotionSimulation":

Parameters	Declaration	Data type	Default value	Description	
Slave	INPUT	TO_SynchronousAxis	-	Following axis technology object	
Enable	INPUT	BOOL	FALSE	TRUE	Simulation of synchronous operation is started.
				FALSE	Simulation of the synchronous operation is stopped.
InSimulation	OUTPUT	BOOL	FALSE	TRUE	Synchronous operation is being simulated
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Continuing synchronous operation when the following axis is disabled

In order not to stop synchronous operation with "MC_SynchronizedMotionSimulation" Motion Control instruction when the following axis is disabled, follow these steps:

1. Check the requirements indicated above.
2. Specify the following axis in the "Slave" parameter.
3. Bring the leading axis to a standstill (e.g. with "MC_Halt").
4. Start the simulation of the synchronous operation on the following axis with "MC_SynchronizedMotionSimulation.Enable" = TRUE.
5. When the "InSimulation" parameter shows the value TRUE, disable the following axis. The synchronous operation remains active in the simulation.
6. To stop the synchronous operation after the following axis is enabled, stop the synchronous operation simulation with "MC_SynchronizedMotionSimulation.Enable" = FALSE.
   The synchronous operation simulation is stopped. The following axis follows the leading axis without re-synchronization.

Cam (S7-1500T)

MC_InterpolateCam V3 (S7-1500T)

MC_InterpolateCam: Interpolate cam V3 (S7-1500T)

Description

With the Motion Control instruction "MC_InterpolateCam", you interpolate a cam.

Following the interpolation, the gaps between the defined interpolation points and segments of the cam are closed. The cam is interpolated from the minimum value of the definition range to the maximum value. After interpolation, an explicit value in the value range is assigned to each value in the definition range.

The interpolation method defines how missing ranges are interpolated. You specify the interpolation method in the configuration of the technology object. The following interpolation methods are possible:

- Linear interpolation
- Interpolation with cubic splines
- Interpolation with Bézier splines

Applies to

- Cam

Requirements

- The technology object has been configured correctly.
- The cam is not currently being used, e.g. for camming.

Override response

The "MC_InterpolateCam" job is not aborted by any other Motion Control job.

The start of an "MC_InterpolateCam" job does not abort any active Motion Control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_InterpolateCam":

Parameters	Declara-tion	Data type	Default value	Description	
Cam	INPUT	TO_Cam	-	Cam technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	The cam is interpolated.

Parameters	Declaration	Data type	Default value	Description	
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Interpolating a cam

To interpolate a cam with the "MC_InterpolateCam" Motion Control instruction, follow these steps:

1. Check the requirements indicated above.
2. Specify the cam to be interpolated in the "Cam" parameter.
3. Start the "MC_InterpolateCam" job with a positive edge at parameter "Execute".

The cam is interpolated. When the "Done" parameter shows the value TRUE, the interpolation is finished.

MC_GetCamFollowingValue V3 (S7-1500T)

MC_GetCamFollowingValue: Read following value of a cam V3 (S7-1500T)

Description

With the Motion Control instruction "MC_GetCamFollowingValue", you read the following value and the first and second derivative of the following value for a leading value from a cam.

Applies to

- Cam

Requirements

- The technology object has been configured correctly.
- The cam is interpolated.

Override response

The "MC_GetCamFollowingValue" job is not aborted by any other Motion Control job.

The start of an "MC_GetCamFollowingValue" job does not abort any active Motion Control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_GetCamFollowingValue":

Parameters	Declaration	Data type	Default value	Description	
Cam	INPUT	TO_Cam	-	Cam technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
LeadingValue	INPUT	LREAL	0.0	Leading value for which the following value is read	
Done	OUTPUT	BOOL	FALSE	TRUE	The following value was read.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	
Value	OUTPUT	LREAL	-	Read following value (position) (valid when "Done" = TRUE)	
FirstDerivative	OUTPUT	LREAL	-	First derivative of read following value (valid when "Done" = TRUE)	
SecondDerivative	OUTPUT	LREAL	-	Second derivative of read following value (valid when "Done" = TRUE)	

Reading a following value

To read a following value from a cam with the "MC_GetCamFollowingValue" Motion Control instruction, follow these steps:

1. Check the requirements indicated above.
2. Specify the cam and the leading value in the corresponding parameters.
3. Start the "MC_GetCamFollowingValue" job with a positive edge at parameter "Execute".

When the "Done" parameter shows the value TRUE, the following value has been read. The following value and the derivatives are output in the "Value", "FirstDerivative" and "SecondDerivative" parameters.

MC_GetCamLeadingValue V3 (S7-1500T)

MC_GetCamLeadingValue: Read leading value of a cam V3 (S7-1500T) (S7-1500T)

Description

With the Motion Control instruction "MC_GetCamLeadingValue", you read the leading value that is defined for a following value from a cam.

Because the same following values can be defined for different leading values, an approximation of the leading value can specified in the "ApproachValue" parameter.

Applies to

- Cam

Requirements

- The technology object has been configured correctly.
- The cam is interpolated.

Override response

The "MC_GetCamLeadingValue" job is not aborted by any other Motion Control job.

The start of an "MC_GetCamLeadingValue" job does not abort any active Motion Control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_GetCamLeadingValue":

Parameters	Declaration	Data type	Default value	Description	
Cam	INPUT	TO_Cam	-	Cam technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
FollowingValue	INPUT	LREAL	0.0	Following value for which the leading value is read	
ApproachLeading-Value	INPUT	LREAL	0.0	Approximation value for the searched for leading value If the following value is used multiple times in the cam, it can be used to limit the searched leading value.	
Done	OUTPUT	BOOL	FALSE	TRUE	The leading value was read.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	
Value	OUTPUT	LREAL	-	Read leading value (position) (valid when "Done" = TRUE)	

Reading a leading value

To read a leading value from a cam with the "MC_GetCamLeadingValue" Motion Control instruction, follow these steps:

1. Check the requirements indicated above.
2. Specify the cam, the leading value, and the approximation value for the searched-for leading value in the corresponding parameters.
3. Start the "MC_GetCamLeadingValue" job with a positive edge at parameter "Execute".

When the "Done" parameter shows the value TRUE, the leading value has been read. The leading value is output in the Value parameter.

Torque data (S7-1500, S7-1500T)

MC_TorqueLimiting V3 (S7-1500, S7-1500T)

MC_TorqueLimiting: Activate and deactivate force/torque limit / fixed stop detection (S7-1500, S7-1500T)

Description

With the Motion Control instruction "MC_TorqueLimiting", you activate and assign parameters for force/torque limiting and fixed stop detection.

Together with a position-controlled motion job, a "Travel to fixed stop" can be realized with the fixed stop detection.

In the axis configuration, you can configure whether the force/torque limiting is to relate to the drive side or the load side.

The functions of the Motion Control instruction "MC-TorqueLimiting" can be activated and deactivated before and during a motion job.

Force/torque limiting applies to

- Speed axis
- Positioning axis
- Synchronous axis

Requirements for force/torque limiting

- The technology object and the reference torque of the drive have been configured correctly.
- The drive must support force/torque reduction. Only PROFIdrive drives with SIEMENS telegram 10x support force/torque limiting.
- No enable-preventing errors are pending for the technology object (the technology object does not have to be enabled).

Fixed stop detection applies to

- Positioning axis
- Synchronous axis

Requirement for fixed stop detection

- The fixed stop detection can only be applied to position-controlled axes. For fixed stop detection, the axis must be enabled as position-controlled axis; motion commands must be executed as position-controlled motion commands.
- The technology object has been configured correctly.
- When a drive and telegram that support force/torque limiting is used, the reference torque of the drive must be correctly configured for the technology object.
- No enable-preventing errors are pending for the technology object (the technology object does not have to be enabled).

Override response

- An MC_TorqueLimiting job cannot be aborted by another Motion Control job.
- A new MC_TorqueLimiting job does not abort active Motion Control jobs.

Parameters

The following table shows the parameters of Motion Control instruction "MC_TorqueLimiting":

Parameters	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	Activate function corresponding to input parameter Mode
Limit	INPUT	LREAL	-1.0	Value of force/torque limiting (in the configured unit of measurement) If drive and telegram do not support force/torque limiting, the specified value is irrelevant.	
				≥ 0	Use the value specified in the parameter ("0" value is permissible)
				< 0	Use the value configured in the "Torque limiting" configuration window". Tag Torque Limit: <TO>.TorqueLimiting.LimitDefaults.Torque Tag Force Limit: <TO>.TorqueLimiting.LimitDefaults.Force
Mode	INPUT	DINT	0	0	Force/torque limiting
				1	Fixed stop detection If drive and telegram support force/torque limiting, this is applied.
InClamping	OUTPUT	BOOL	FALSE	TRUE	Mode = 1: The drive is kept at the fixed stop (clamping), the axis position is within the positioning tolerance.
InLimitation	OUTPUT	BOOL	FALSE	TRUE	Mode = 0 and 1: Drive is operating at the force/torque limit.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.

Parameters	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Changes to input parameters "Limit" and "Mode" are also transferred at the cyclic call of the Motion Control instruction when "Enable = TRUE".

MC_TorqueLimiting: Function chart V3 (S7-1500, S7-1500T)

Function chart: Stopping an axis when the torque limit is reached

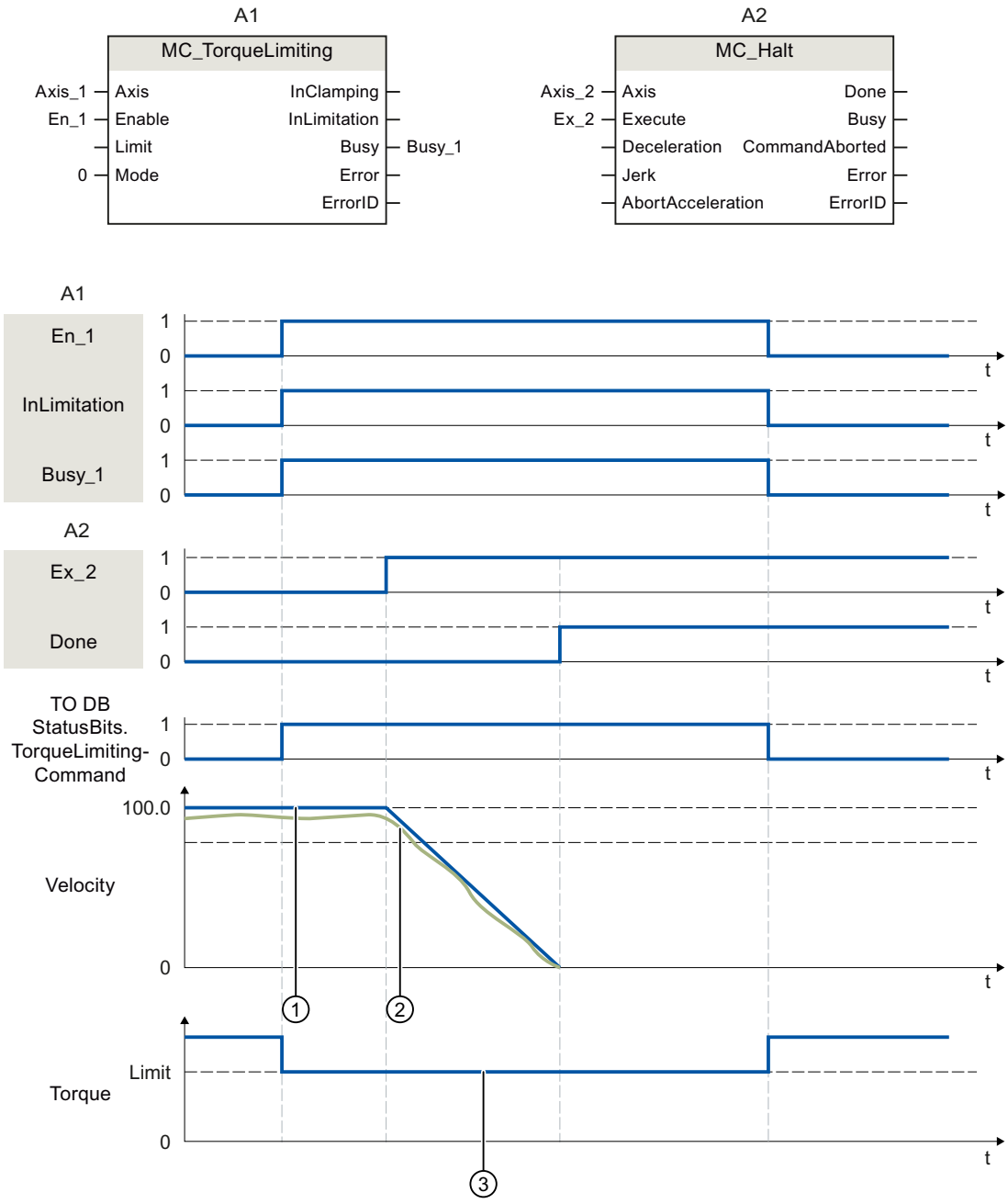

① Velocity setpoint
② Actual velocity
③ Torque

A "MC_Halt" Motion Control instruction is executed for an axis with active torque limiting "MC_TorqueLimiting". Torque limiting still "MC_TorqueLimiting.Enable" remains TRUE

(active) and any accumulated following error is retained and will be reduced. When the actual velocity is "0.0" and the minimum dwell time in the standstill window has expired, "MC_Halt.Done" = TRUE". When position monitoring is activated, reaching of the target position is also monitored.

Function chart: Torque limiting with fixed stop detection (mode = 1)

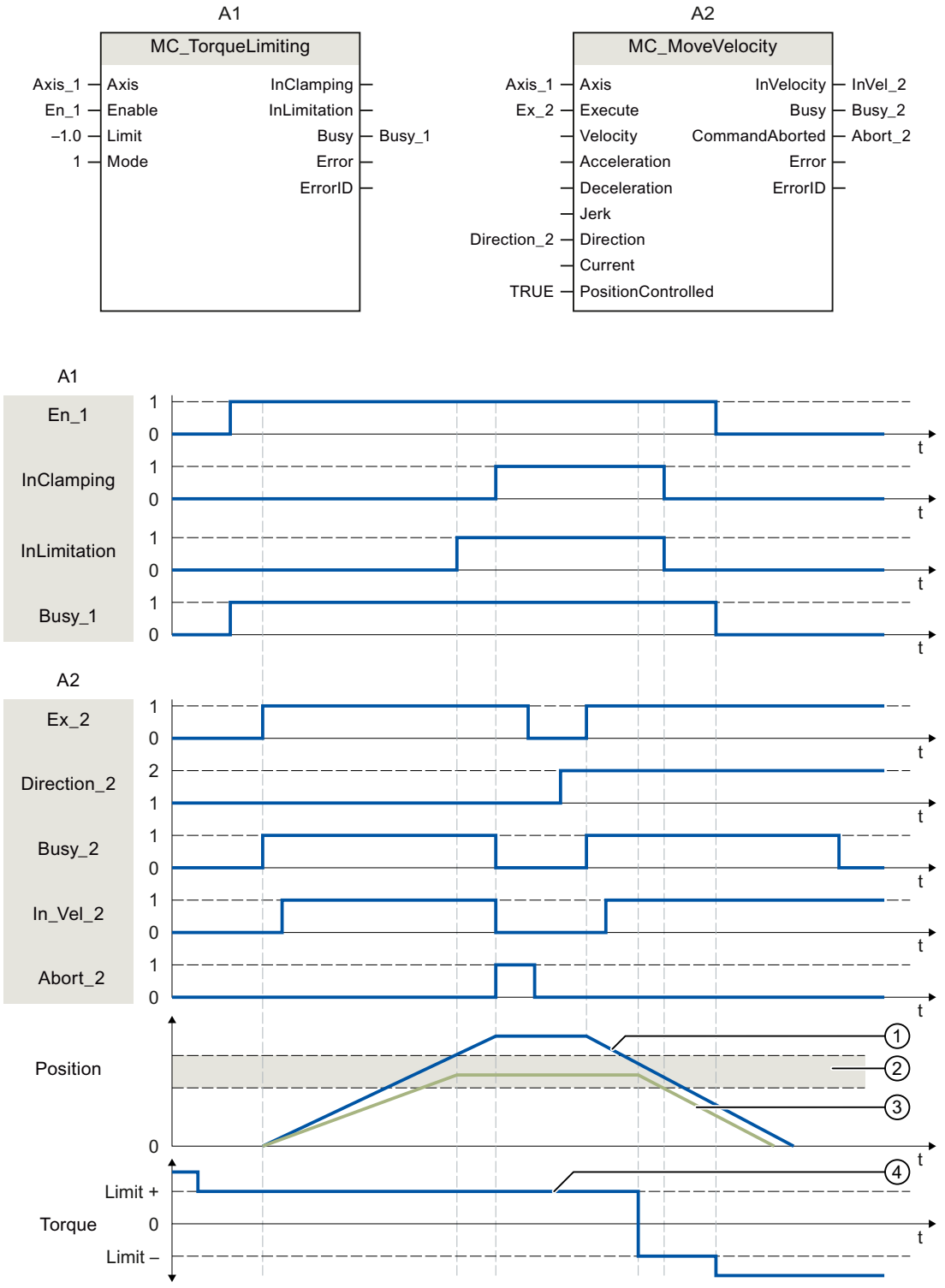

① Actual position remains

② Target position reached (position setpoint)

③ Positioning tolerance
④ Torque

A "MC_MoveVelocity" Motion Control instruction is executed for an axis with active torque limiting "MC_TorqueLimiting". The torque limiting "MC_TorqueLimiting.Enable" is still TRUE (active) and the configured following error is reduced. When reaching the following error limit, the Motion Control instruction "MC_MoveVelocity" is canceled with "Abort" = TRUE. The drive is kept at the fixed stop (clamping), the axis position is within the positioning tolerance. The Motion Control instruction "MC_MoveVelocity" is called again with "Execute" = TRUE, and the axis is moved with constant velocity.

Override response of Motion Control commands V3 (S7-1500, S7-1500T)

Homing and motion jobs

The following table shows how a new Motion Control job affects active homing and motion jobs:

⇒ Active job ⇓ New job	MC_Home Mode = 2, 8, 10	MC_Home Mode = 3, 5	MC_Halt	MC_Move Absolute	MC_Move Relative	MC_Move Velocity	MC_Move Jog	MC_Move Super imposed
MC_Home Mode = 3, 5	A	A	A	A	A	A	A	A
MC_Home Mode = 9	A	-	-	-	-	-	-	-
MC_Halt	-	A	A	A	A	A	A	A
MC_Move Absolute	-	A	A	A	A	A	A	A
MC_Move Relative	-	A	A	A	A	A	A	A
MC_Move Velocity	-	A	A	A	A	A	A	A
MC_MoveJog	-	A	A	A	A	A	A	A
MC_Move Superimposed	-	-	-	-	-	-	-	A
MC_GearIn	-	A	A	A	A	A	A	A
MC_GearInPos pending (Busy = TRUE, StartSync = FALSE, InSync = FALSE)	-	-	-	-	-	-	-	-
MC_GearInPos active (Busy = TRUE, StartSync or InSync = TRUE)	-	A	A	A	A	A	A	A

⇒ Active job ⇓ New job	MC_Home Mode = 2, 8, 10	MC_Home Mode = 3, 5	MC_Halt	MC_Move Absolute	MC_Move Relative	MC_Move Velocity	MC_Move Jog	MC_Move Super imposed
MC_CamIn pending (Busy = TRUE, StartSync = FALSE, InSync = FALSE)	-	-	-	-	-	-	-	-
MC_CamIn active (Busy = TRUE, StartSync or InSync = TRUE)	-	A	A	A	A	A	A	A

A The running job is aborted with "JobAborted" = TRUE. The current dynamics is smoothly transitioned to the dynamic values of the new job.

- No effect, running job continues to be executed

Synchronous operation jobs

The following table shows how a new Motion Control job affects the motion of the axis on active synchronous operation jobs:

⇒ Active job ⇓ New job	MC_GearIn	MC_GearInPos pending (Busy = TRUE, InSync = FALSE, StartSync = FALSE)	MC_GearInPos active (Busy = TRUE, StartSync or InSync = TRUE)	MC_Phasing Absolute	MC_Phasing Relative	MC_CamIn pending (Busy = TRUE, InSync = FALSE, StartSync = FALSE)	MC_CamIn active (Busy = TRUE, StartSync or InSync = TRUE)
MC_Home Mode = 3, 5	A	-	-	-	-	-	-
MC_Halt	A	-	A	A	A	-	A
MC_MoveAbsolute	A	-	A	A	A	-	A
MC_MoveRelative	A	-	A	A	A	-	A
MC_MoveVelocity	A	-	A	A	A	-	A
MC_MoveJog	A	-	A	A	A	-	A
MC_MoveSuperimposed	-	-	-	-	-	-	-
MC_GearIn	A	A	A	A	A	A	A
MC_GearInPos pending (Busy = TRUE, StartSync = FALSE, InSync = FALSE)	-	A	-	-	-	A	-
MC_GearInPos active (Busy = TRUE, StartSync or InSync = TRUE)	A	A	A	A	A	A	A
MC_PhasingAbsolute	-	-	-	A	A	-	-
MC_PhasingRelative	-	-	-	A	A	-	-

⇒ Active job ⇓ New job	MC_GearIn	MC_GearInPos pending (Busy = TRUE, InSync = FALSE, StartSync = FALSE)	MC_GearInPos active (Busy = TRUE, StartSync or InSync = TRUE)	MC_Phasing Absolute	MC_Phasing Relative	MC_CamIn pending (Busy = TRUE, InSync = FALSE, StartSync = FALSE)	MC_CamIn active (Busy = TRUE, StartSync or InSync = TRUE)
MC_CamIn pending (Busy = TRUE, StartSync = FALSE, InSync = FALSE)	-	A	-	-	-	A	-
MC_CamIn active (Busy = TRUE, StartSync or InSync = TRUE)	A	A	A	A	A	A	A

A The running job is aborted with "JobAborted" = TRUE. The current dynamics is smoothly transitioned to the dynamic values of the new job.

- No effect, running job continues to be executed

A pending synchronous operation (Busy = TRUE, StartSync = FALSE, InSync = FALSE) does not cancel active commands. Only a pending synchronous operation command is permitted. A second pending synchronous operation command cancels the first pending synchronous operation command.

When one synchronous operation command is active and one synchronous operation command is pending, a new motion command cancels the active as well as the pending synchronous operation command.

Measuring input

The following table shows which new Motion Control jobs will override active measuring input jobs:

⇒ Active job ⇓ New job	MC_MeasuringInput	MC_MeasuringInputCyclic
MC_Home Mode = 2, 3, 5, 8, 9, 10	A	A
MC_Home Mode = 0, 1, 6, 7	-	-
MC_MeasuringInput	A	A
MC_MeasuringInputCyclic	A	A
MC_AbortMeasuringInput	A	A

A The running job is aborted with "JobAborted" = TRUE. The new job will be executed.

- No effect, running job continues to be executed

S7-1500 Motion Control V2 (S7-1500)

MC_Power V2 (S7-1500)

MC_Power: Enable, disable technology object V2 (S7-1500)

Description

With the Motion Control instruction "MC_Power", a technology object is enabled or disabled.

Applies to

- Synchronous axis
- Positioning axis
- Speed axis
- External encoder

Requirements

- The technology object has been configured correctly.

Override response

- An "MC_Power" job cannot be aborted by any other Motion Control job.
- An "MC_Power" job with parameter "Enable" = TRUE enables a technology object but does not thereby abort any other Motion Control instructions.
- Disabling the technology object (parameter "Enable" = FALSE) aborts all Motion Control jobs on the corresponding technology object in accordance with the selected "StopMode". This process cannot be canceled by the user.

Parameters

The following table shows the parameters of the "MC_Power" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description	
Axis	InOut	TO_Axis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The technology object is enabled.
				FALSE	The technology object is disabled. All current jobs at the technology object are aborted in accordance with the configured "StopMode".

Parameters	Declaration	Data type	Default value	Description	
StopMode	INPUT	INT	0	Not applicable to the technology object external encoder If you disable a technology object with a negative edge at parameter "Enable", the axis decelerates in accordance with the selected "StopMode".	
				0	Emergency stop When the technology object is disabled, the axis brakes to a standstill without jerk limit, using the emergency stop deceleration configured in "Technology object > Configuration > Extended parameters > Emergency stop ramp". The enable is then canceled. (<TO>.DynamicDefaults.EmergencyDeceleration)
				1	Immediate stop When a technology object is disabled, the setpoint zero is output, and the enable is canceled. The axis is braked to a standstill according to the configuration in the drive.
				2	Stop with maximum dynamic values When the technology object is disabled, the axis is braked to a standstill using the maximum deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic limits". The configured maximum jerk is hereby taken into account. The enable is then canceled. (<TO>.DynamicLimits.MaxDeceleration; <TO>.DynamicLimits.MaxJerk)
Status	OUTPUT	BOOL	FALSE	Technology object enable status	
				FALSE	Disabled - A positioning axis or speed axis does not accept any Motion Control jobs. - Speed control and position control are not active. - The actual values of the technology object are not checked for validity.
				TRUE	Enabled - An enabled positioning axis or speed axis accepts Motion Control jobs. - Speed control and position control are active. - The actual values of the technology object are valid.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction MC_Power. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Enabling technology objects

To enable a technology object, set the "Enable" parameter to TRUE.

When the Status parameter shows the value TRUE , the technology object is enabled.

If an axis is in motion when the technology object is enabled (actual velocity is present), the axis brakes to the setpoint zero, using the maximum deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic limits" (<TO>.DynamicLimits.MaxDeceleration). This braking ramp can be overridden by Motion Control jobs.

Note

Automatic enable after acknowledgment of a technology alarm

If the technology object is disabled due to a technology alarm, it will automatically be enabled again after the cause has been eliminated and the alarm has been acknowledged. This requires the "Enable" parameter to have retained the value TRUE during this process.

Disabling technology objects

To disable a technology object, set the "Enable" parameter to FALSE.

If an axis is in motion, it is braked to a standstill according to the selected "StopMode".

When the "Busy" and "Status" parameters show the value FALSE , the disabling of the technology object is complete.

Drive connection by means of PROFIdrive

When a drive is connected using PROFIdrive, the setpoint, enable and drive status are transmitted via the PROFIdrive telegram.

- **Enable technology object and activate drive**
  With parameter "Enable" = TRUE, the technology object is enabled. The drive is enabled according to the PROFIdrive standard.
  When the tag <TO>.StatusDrive.InOperation shows the value TRUE , the drive is ready to execute setpoints. The "Status" parameter is set to the value TRUE .
- **Disable technology object and deactivate drive**
  With parameter "Enable" = FALSE, the "Status" parameter is set to the value FALSE, and the axis is braked according to the selected "StopMode". The drive is disabled according to the PROFIdrive standard.

Analog drive connection

The setpoint is output via an analog output. Optionally, you can configure an enabling signal via digital output (<TO>.Actor.Interface.EnableDriveOutput), and a readiness signal via digital input (<TO>.Actor.Interface.DriveReadyInput).

- **Enable technology object and activate drive**
  With parameter "Enable" = TRUE, the enable output ("Enable drive output") is set.
  When the drive returns the readiness signal via the ready input ("Drive ready input"), the "Status" parameter and the technology object's <TO>.StatusDrive.InOperation tag are set to TRUE, and the setpoint is switched to the analog output.
- **Disable technology object and deactivate drive**
  With parameter "Enable" = FALSE, the "Status" parameter is set to the value FALSE, and the axis is braked according to the selected "StopMode". When the setpoint zero is reached, the enable output is set to FALSE.

Additional information

Additional information on enabling and disabling technology objects and drives can be found in the appendix, MC_Power Function Charts.

MC_Power: Function chart V2 (S7-1500)

Function chart: Enabling a technology object and example of alarm response

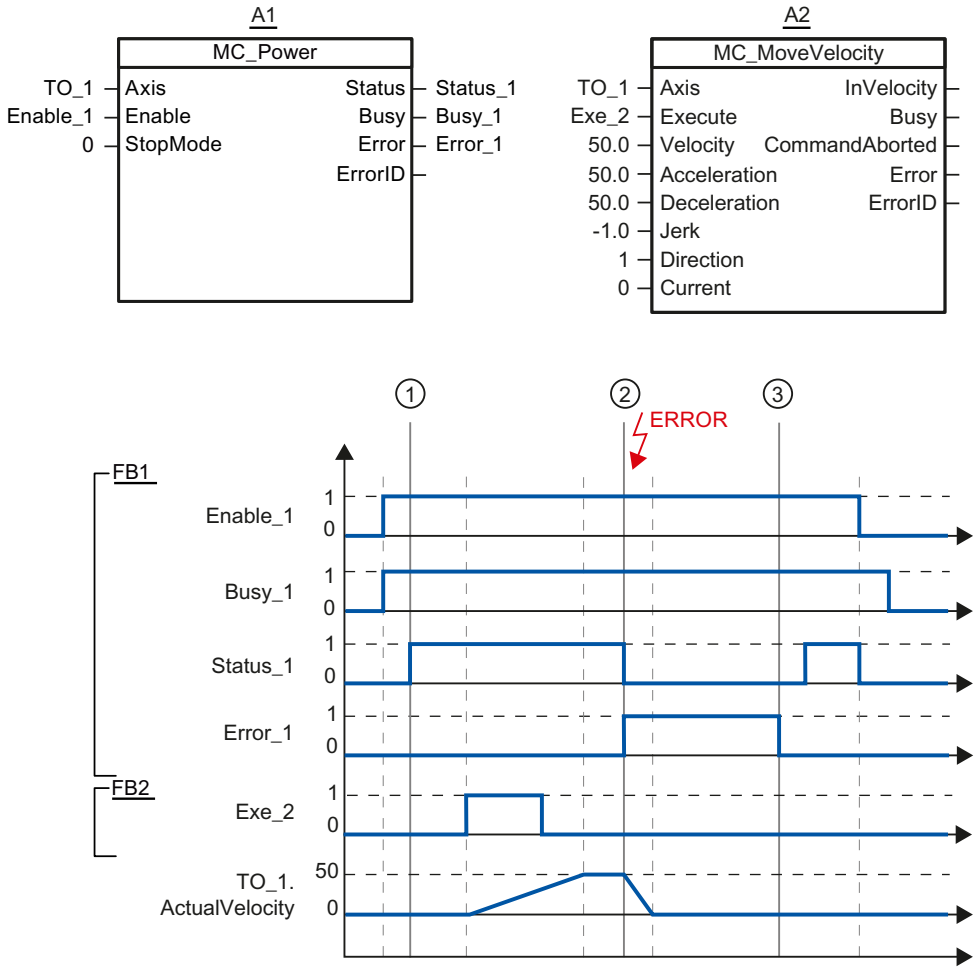

A technology object is enabled with "Enable_1= TRUE". The successful enable can be read from "Status_1" at time ①. The axis will then move with an "MC_MoveVelocity" job (A2). The velocity profile of the axis can be read from "Velocity Axis_1".

At time ② an error occurs in the technology object, which results in the disabling of the technology object (alarm response: remove enable). The axis is braked to a standstill according to the configuration in the drive. When the technology object is disabled, "Status_1" is reset. Since the axis was not disabled using "Enable_1" = FALSE, the selected "StopMode" does not apply. The cause of the error is corrected and the alarm is acknowledged at time ③.

Since "Enable_1" is still set, the technology object is enabled again. The successful enable can be read from "Status_1". Finally, the technology object is disabled with "Enable_1" = FALSE.

MC_Reset V2 (S7-1500)

MC_Reset: Acknowledge alarms, restart technology object V2 (S7-1500)

Description

With the Motion Control instruction "MC_Reset", you acknowledge all technology alarms that can be acknowledged in the user program. Acknowledgment also resets the "Error" and "Warning" bits in the technology data block.

With the Motion Control instruction "MC_Reset" with "Restart" = TRUE, you start reinitialization (restart) of technology objects. Upon restart of the technology object, the new configuration data are applied in the technology data block.

Applies to

- Synchronous axis
- Positioning axis
- Speed axis
- External encoder

Requirements

- The cause of the error has been resolved for a pending technology alarm.
- For a restart, the technology object must be disabled.
  ("MC_Power.Status" = FALSE and "MC_Power.Busy" = FALSE)

Override response

- Parameter "Restart" = FALSE:
  Processing of the "MC_Reset" instruction can be aborted by other Motion Control jobs. The MC_Reset job does not abort any running Motion Control jobs.
- Parameter "Restart" = TRUE:
  Processing of the "MC_Reset" instruction with parameter "Restart" = TRUE cannot be aborted by any other Motion Control job.

Parameters

The following table shows the parameters of the "MC_Reset" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	InOut	TO_Axis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge

Parameters	Declaration	Data type	Default value	Description	
Restart	INPUT	BOOL	FALSE	TRUE	"Restart" Reinitialization of the technology object and acknowledgment of pending technology alarms. The technology object is reinitialized with the configured start values.
				FALSE	Acknowledgment of queued technology alarms.
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged. Restart has been executed.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Acknowledging technology alarms

To acknowledge technology alarms, follow these steps:

1. Check the requirements indicated above.
2. Set parameter "Restart" = FALSE.
3. Start the acknowledgment of the error with a positive edge at parameter "Execute".

When the Done parameter shows the value TRUE, the error has been acknowledged.

Note

Acknowledging with "Restart" = FALSE

If only the technology alarms are to be acknowledged, set "Restart" = FALSE. The technology object cannot be used during a restart.

Restarting a technology object

To restart a technology object, follow these steps:

1. Check the requirements indicated above.
2. Set parameter "Restart" = TRUE.
3. Perform the restart with a positive edge at parameter "Execute".

When the "Done" parameter shows the value TRUE, the restart of the technology object is complete.

Additional information regarding the restart can be found in the Restarting technology objects section.

MC_Home V2 (S7-1500)

MC_Home: Home technology objects, set home position V2 (S7-1500)

Description

With the Motion Control instruction "MC_Home", you create the relationship between the position in the technology object and the mechanical position. The actual position value in the technology object is assigned to a homing mark at the same time. This homing mark represents a known mechanical position.

Homing is performed according to the mode selected with the "Mode" parameter and the configuration under "Technology object > Configuration > Extended parameters > Homing".

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview.

The preset values under "Technology object > Configuration > Extended parameters > Dynamic default values" are used for the dynamic values Acceleration, Deceleration and Jerk.

Applies to

- Synchronous axis
- Positioning axis
- External encoder

The following table shows which modes are possible with each of the technology objects:

Operating mode	Positioning axis/ synchronous axis with incremental encoder	Positioning axis/ synchronous axis with absolute encoder	External incremental encoder	External absolute encoder
Active homing ("Mode" = 3, 5)	X	-	-	-
Passive homing ("Mode" = 2, 8, 10)	X	-	X	-
Set actual position ("Mode" = 0)	X	X	X	X
Relative shift of the actual position ("Mode" = 1)	X	X	X	X
Homing absolute encoder ("Mode" = 6, 7)	-	X	-	X

Requirements

- The technology object has been configured correctly.
- "Mode" = 2, 3, 5, 8, 10
  The technology object must be enabled.
- Mode" = 0, 1, 6, 7, 8
  The encoder values must be valid. (<TO>.StatusSensor[n].State = 2)

Override response

- An MC_Home job for passive homing is aborted by:
  - Disabling the technology object with "MC_Power.Enable" = FALSE
  - "MC_Home" job with parameter "Mode" = 3, 5, 9
- An "MC_Home" job for passive homing does not abort any other Motion Control jobs.
- An "MC_Home" job for active homing is aborted by:
  - Disabling the technology object with "MC_Power.Enable" = FALSE
  - "MC_Home" command "Mode" = 3, 5
  - "MC_Halt" command
  - "MC_MoveAbsolute" command
  - "MC_MoveRelative" command
  - "MC_MoveVelocity" command
  - "MC_MoveJog" command
  - "MC_GearIn" command
- An "MC_Home" job for active homing aborts the following active Motion Control jobs.
  - "MC_Home" job with parameter "Mode" = 3, 5, 8, 10
  - "MC_Halt" command
  - "MC_MoveAbsolute" command
  - "MC_MoveRelative" command
  - "MC_MoveVelocity" command
  - "MC_MoveJog" command
  - "MC_GearIn" command
  - "MC_MoveSuperimposed" command

Parameters

The following table shows the parameters of the "MC_Home" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	InOut	TO_Axis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge

Parameters	Declaration	Data type	Default value	Description
Position	INPUT	LREAL	0.0	The specified value is used according to the selected "Mode".

Parameters	Declaration	Data type	Default value	Description	
Mode	INPUT	INT	0	Operating mode	
				0	Direct homing (absolute) The current position of the technology object is set to the value of parameter "Position".
				1	Direct homing (relative) The current position of the technology object is shifted by the value of parameter "Position".
				2	Passive homing (without reset) Functions like "Mode" 8, with the difference that the "homed" status is **not** reset when the function is enabled.
				3	Active homing The TO positioning axis/synchronous axis performs a homing movement according to the configuration. After the completion of the movement, the axis is positioned at the value of the "Position" parameter.
				4	Reserved
				5	Active homing ("Position" parameter has no effect) The TO positioning axis/synchronous axis performs a homing movement according to the configuration. After completion of the movement, the axis is positioned at the home position configured under "Technology object > Configuration > Extended parameters > Homing > Active homing". (<TO>.Homing.HomePosition)
				6	Absolute encoder adjustment (relative) The current position is shifted by the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<TO>.StatusSensor[n].AbsEncoderOffset)
				7	Absolute encoder adjustment (absolute) The current position is set to the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<TO>.StatusSensor[n].AbsEncoderOffset)
				8	Passive homing

Parameters	Declaration	Data type	Default value	Description	
					When the homing mark is detected, the actual value is set to the value of the "Position" parameter.
				9	Abort passive homing An active job for passive homing is aborted.
				10	Passive homing ("Position" parameter has no effect) When the homing mark is detected, the actual value is set to the home position configured under "Technology object > Configuration > Extended parameters > Homing > Passive homing". (<TO>.Homing.HomePosition)
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Resetting the "Homed" status

The "Homed" status of a technology object is reset under the following conditions (<TO>.StatusWord.X5 (HomingDone)):

- **Technology objects with incremental actual values:**
  - Starting an "MC_Home" job with "Mode" = 3, 5, 8, 10
    (After successful completion of the homing operation, the "Homed" status is reset.)
  - Error in the encoder system, or encoder failure
  - Restart of the technology object
  - After POWER OFF -> POWER ON of the CPU
  - Memory reset
  - Modification of the encoder configuration
- **Technology objects with absolute actual values:**
  - Error in sensor system/encoder failure
  - Replacement of the CPU
  - Modification of the encoder configuration
  - Restoration of the CPU factory settings
  - Transfer of a different project into the controller

Homing a technology object with "Mode" = 1..8, 10

To home a technology object, follow these steps:

1. Check the requirements indicated above.
2. Specify the desired homing function in the "Mode" parameter.
3. Initialize the necessary parameters with values, and start the homing operation with a positive edge at the "Execute" parameter.

When the "Done" parameter shows the value TRUE, the "MC_Home" job has been completed according to the selected "Mode". The "Homed" status of the technology object is indicated under "Technology object > Diagnostics > Status and error bits > Motion status > Homed" (<TO>.StatusWord.X5 (HomingDone)).

Abort of a passive homing process with "Mode" = 9

With "Mode" = 9, the technology object is not homed. When an active "MC_Home" job for passive homing ("Mode" = 2, 8, 10) is overridden by another "MC_Home" job with "Mode" = 9, the running job is aborted with parameter "CommandAborted" = TRUE. The overriding job with "Mode" = 9 signals successful execution with parameter "Done" = TRUE.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

MC_Halt V2 (S7-1500)

MC_Halt: Halt axis V2 (S7-1500)

Description

With the Motion Control instruction "MC_Halt", you brake an axis until it comes to a standstill.

You define the dynamic behavior of the braking operation with parameters "Jerk" and "Deceleration".

Applies to

- Synchronous axis
- Positioning axis
- Speed axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The "MC_Halt" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command

The start of an "MC_Halt" job aborts the following active Motion Control jobs:

- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command
- "MC_MoveSuperimposed" command

Parameters

The following table shows the parameters of the "MC_Halt" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	InOut	TO_SpeedAxis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)

Parameters	Declaration	Data type	Default value	Description	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Braking an axis with "MC_Halt"

Proceed as follows to decelerate an axis to a standstill:

1. Check the requirements indicated above.
2. Supply the parameters "Deceleration" and "Jerk" with the desired values.
3. Start the "MC_Halt" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error". The standstill of the axis is indicated under "Technology object > Diagnostics > Status and error bits > Motion status > Standstill" (<TO>.StatusWord.X7 (Standstill)).

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

MC_Halt: Function chart V2 (S7-1500)

Function chart: Stopping an axis and the overriding job characteristics

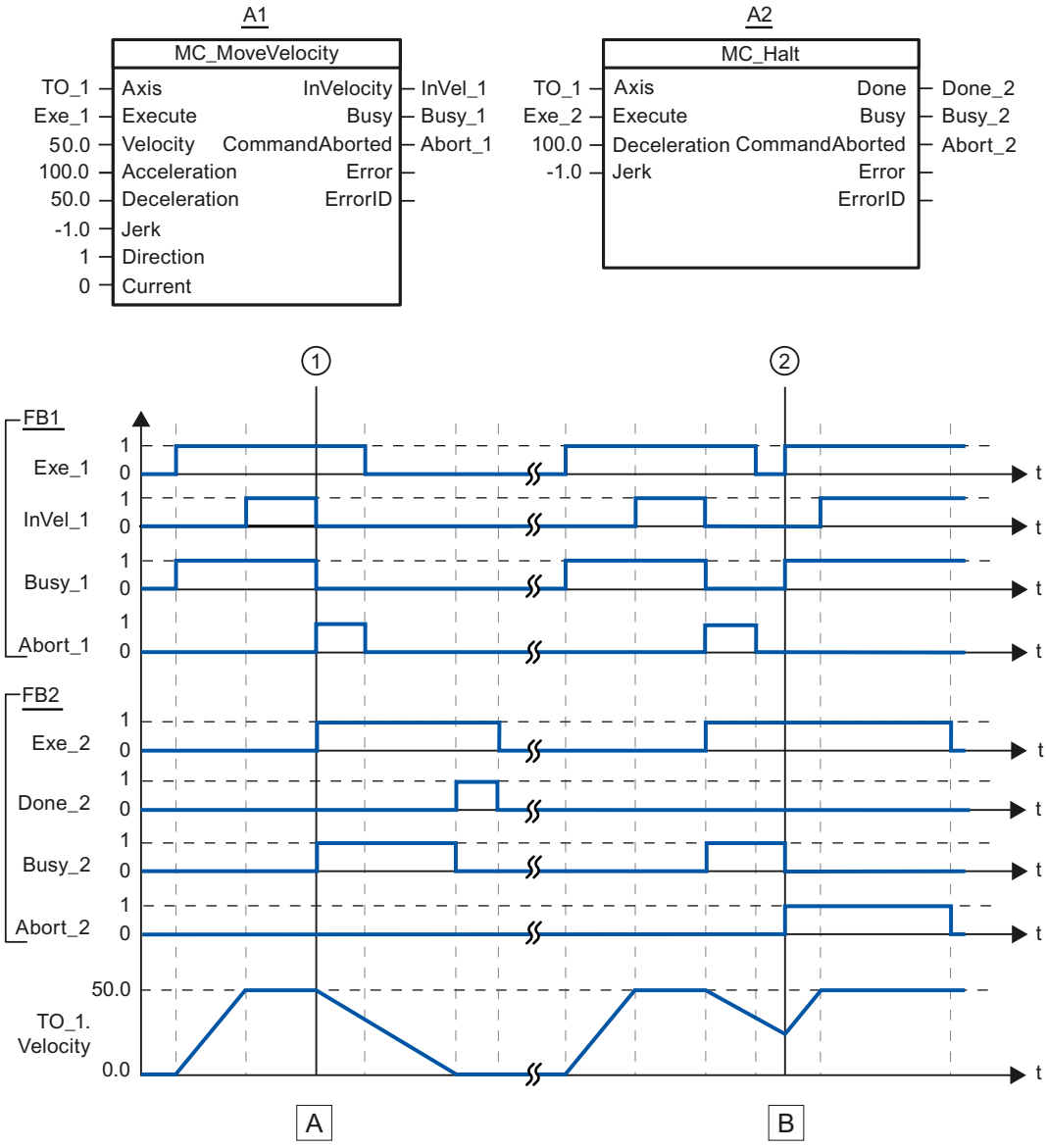

Section A	An axis is moved with an "MC_MoveVelocity" job (A1). When the velocity setpoint 50.0 is reached, this is signaled via "InVel_1". At time ①, the "MC_MoveVelocity" job is overridden by an "MC_Halt" job (A2). The job abort is signaled via "Abort_1". The axis is braked to a standstill. Successful completion of the "MC_Halt" job is signaled via "Done_2".
Section B	The axis is moved with an "MC_MoveVelocity" job (A1). When the velocity setpoint 50.0 is reached, this is signaled via "InVel_1". Next the "MC_MoveVelocity" job is overridden by an "MC_Halt" job (A2). The job abort is signaled via "Abort_1". During the braking process the "MC_Halt" job is overridden at time ② by an "MC_MoveVelocity" job (A1). The job abort is signaled via "Abort_2". When the velocity setpoint 50.0 is reached, this is signaled via "InVel_1". The axis is then moved with constant velocity.

MC_MoveAbsolute V2 (S7-1500)

MC_MoveAbsolute: Absolute positioning of axis V2 (S7-1500)

Description

With the Motion Control instruction "MC_MoveAbsolute", you can move an axis to an absolute position.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

Applies to

- Synchronous axis
- Positioning axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.
- The technology object is homed.

Override response

The "MC_MoveAbsolute" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command

The start of an "MC_MoveAbsolute" job aborts the following active Motion Control jobs:

- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command

- "MC_GearIn" command
- "MC_MoveSuperimposed" command

Parameters

The following table shows the parameters of the "MC_MoveAbsolute" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	InOut	TO_Positioning-Axis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Position	INPUT	REAL	0.0	Absolute target position
Velocity	INPUT	LREAL	-1.0	Velocity setpoint for the positioning Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)

Parameters	Declaration	Data type	Default value	Description	
Direction	INPUT	INT	1	Direction of rotation of the axis Is only evaluated with "modulo" enabled. "Technology object > Configuration > Basic parameters > Enable modulo	
				1	Positive direction of rotation
				2	Negative direction of rotation
				3	Shortest path
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Moving an axis to an absolute position

Proceed as follows to move an axis to an absolute position:

1. Check the requirements indicated above.
2. Specify the desired target position in the "Position" parameter.
3. Start the "MC_MoveAbsolute" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error".

MC_MoveAbsolute: Function chart V2 (S7-1500)

Function chart: Absolute positioning of an axis, and the response to an overriding job

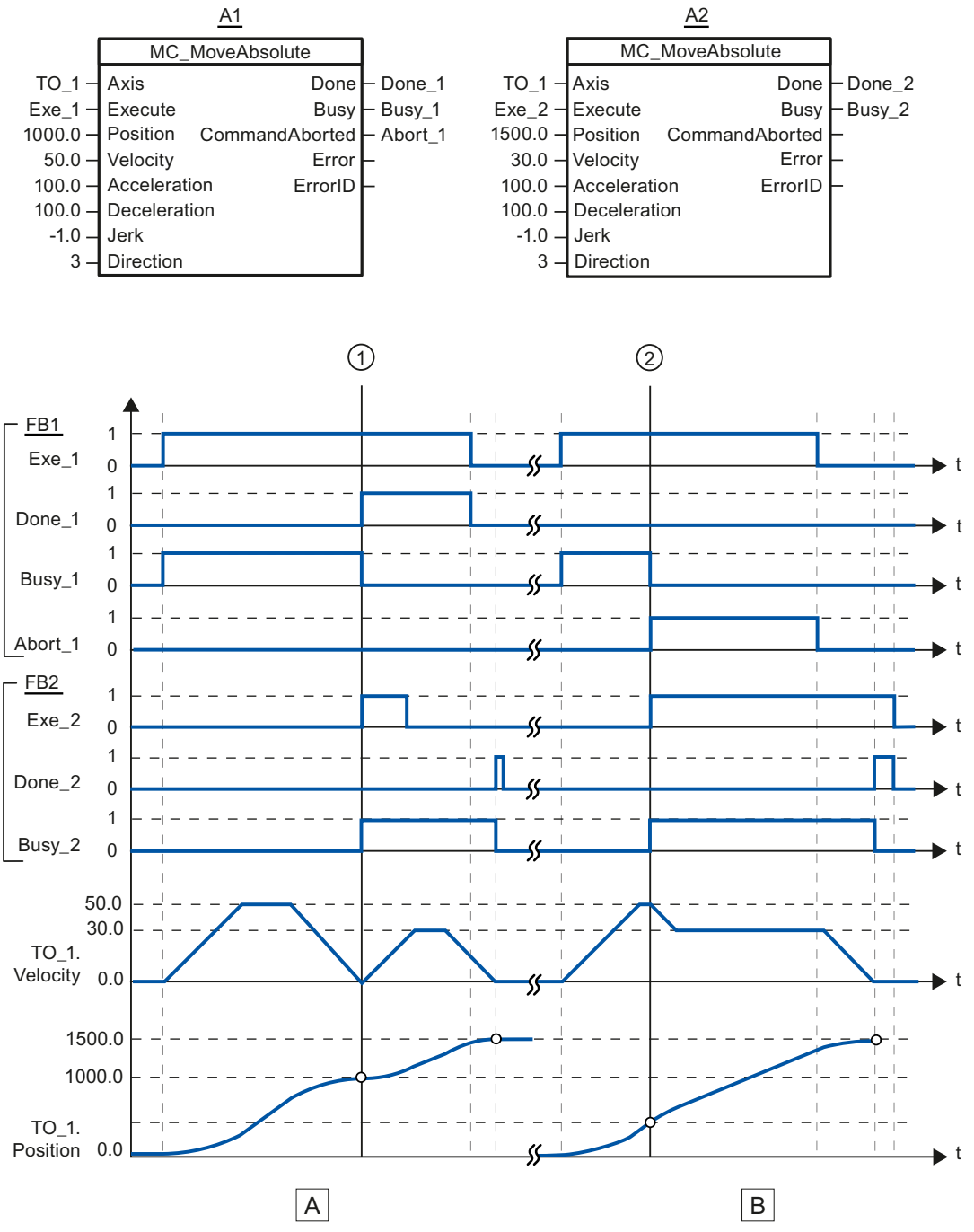

Section A	An axis is moved to absolute position 1000.0 with an "MC_MoveAbsolute" job (A1). When the axis reaches the target position, this is signaled at time ① via "Done_1". At this time ①, another "MC_MoveAbsolute" job (A2) with target position 1500.0 is started. When the axis reaches the target position 1500.0, this is signaled via "Done_2". Since "Exe_2" was previously reset, "Done_2" is applied only to one cycle.
Section B	An active "MC_MoveAbsolute" job (A1) is overridden at time ② by another "MC_MoveAbsolute" job (A2). The abort is signaled via "Abort_1". The axis is braked to the changed velocity and moved to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveRelative V2 (S7-1500)

MC_MoveRelative: Relative positioning of axis V2 (S7-1500)

Description

With the Motion Control instruction "MC_MoveRelative", you move an axis relative to its position when execution of the job began.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

Applies to

- Synchronous axis
- Positioning axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The "MC_MoveRelative" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command

The start of an "MC_MoveRelative" job aborts the following active Motion Control jobs:

- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command
- "MC_MoveSuperimposed" command

Parameters

The following table shows the parameters of the "MC_MoveRelative" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	InOut	TO_Positioning-Axis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Distance	INPUT	LREAL	0.0	Distance for the positioning process (negative or positive)
Velocity	INPUT	LREAL	-1.0	Velocity setpoint for the positioning Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)

Parameters	Declaration	Data type	Default value	Description	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Moving an axis relative to the starting position

Proceed as follows to move an axis relative to the starting position:

1. Check the requirements indicated above.
2. Specify distance to be moved in the "Distance" parameter.
3. Start the "MC_MoveRelative" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error".

MC_MoveRelative: Function chart V2 (S7-1500)

Function chart: Relative positioning of an axis, and the overriding job characteristics

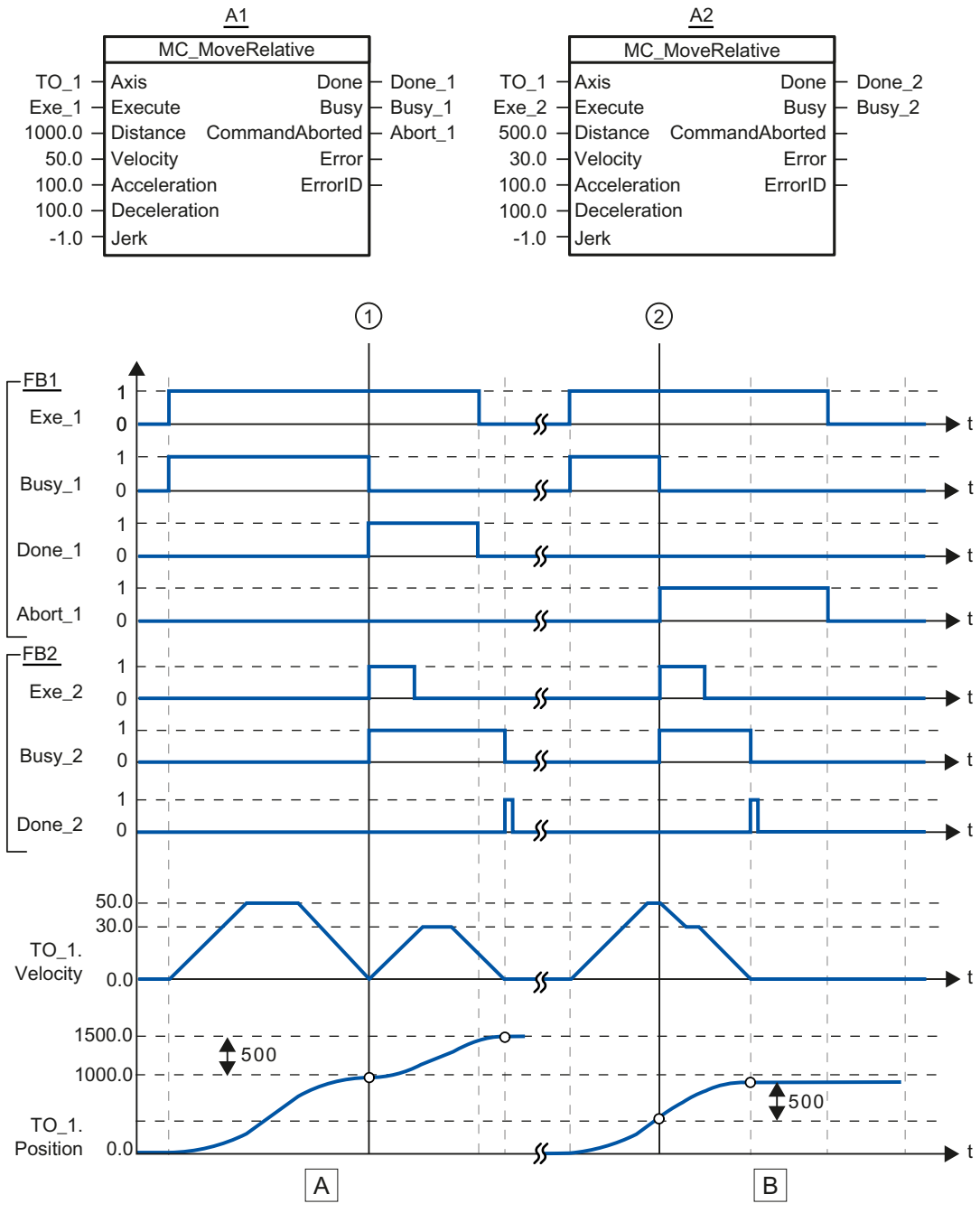

Section A	The axis is moved by an "MC_MoveRelative" job (A1) by the distance ("Distance") 1000.0 (the starting position here is 0.0). When the axis reaches the target position, this is signaled at time ① via "Done_1". At this time ①, another "MC_MoveRelative" job (A2) with distance 500.0 is started. When the axis reaches the new target position, this is signaled via "Done_2". Since "Exe_2" was previously reset, "Done_2" is applied only to one cycle.
Section B	An active "MC_MoveRelative" job (A1) is overridden by another "MC_MoveRelative" job (A2). The abort is signaled at time ② via "Abort_1". The axis is then moved at the new velocity by the distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveVelocity V2 (S7-1500)

MC_MoveVelocity: Move axis at set velocity/speed V2 (S7-1500)

Description

With the Motion Control instruction "MC_MoveVelocity", you move an axis at constant velocity/speed.

You define the dynamic behavior of the motion with parameters"Velocity", "Jerk", "Acceleration" and "Deceleration".

- Positioning axis/synchronous axis:
  A velocity is specified in the "Velocity" parameter.
- Speed axis:
  A speed is specified in the "Velocity" parameter.

Applies to

- Synchronous axis
- Positioning axis
- Speed axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The "MC_MoveVelocity" is canceled by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command

- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command

The start of an "MC_MoveVelocity" job aborts the following active Motion Control jobs:

- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command
- "MC_MoveSuperimposed" command

Parameters

The following table shows the parameters of the "MC_MoveVelocity" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	InOut	TO_SpeedAxis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Velocity	INPUT	LREAL	100.0	Velocity/speed setpoint for the motion process ("Velocity" = 0.0 is permitted)
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)

Parameters	Declaration	Data type	Default value	Description	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used. Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Direction	INPUT	INT	0	Direction of rotation of the axis	
				0	The sign of the velocity specified at the "Velocity" parameter defines the direction of rotation.
				1	Positive direction of rotation Value of "Velocity" is used.
				2	Negative direction of rotation Value of "Velocity" is used.
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	Deactivated The values of parameters "Velocity" and "Direction" are taken into account.
				TRUE	activated The values at the parameters "Velocity" and "Direction" are **not** taken into account. The current velocity and direction at function start are retained. When the axis resumes motion at the velocity that was current at function start, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity setpoint / speed setpoint was reached and will be maintained.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Behavior with velocity setpoint / speed setpoint zero ("Velocity" = 0.0)

An "MC_MoveVelocity" job with "Velocity" = 0.0 stops the axis with the configured deceleration. When the velocity setpoint / speed setpoint zero is reached, the parameter "InVelocity" will indicate the value TRUE.

Under "Technology object > Diagnostics > Status and error bits > Motion status", "constant velocity" and "standstill" will be displayed (<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill)).

The parameters "InVelocity" and "Busy" show the value TRUE until the "MC_MoveVelocity" job is overridden by another Motion Control job.

Moving an axis with constant velocity / speed

Proceed as follows to move an axis with constant velocity / speed:

1. Check the requirements indicated above.
2. At the parameter "Velocity", specify the velocity / speed, with which the axis should be moved.
3. Start the "MC_MoveVelocity" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "InVelocity" and "Error".

If the "InVelocity" parameter shows the value TRUE, then the velocity setpoint / speed setpoint was reached. The axis continues moving at this constant velocity. The parameters "InVelocity" and "Busy" show the value TRUE until the "MC_MoveVelocity" job is overridden by another Motion Control job.

Note

Response to a change in the override

If the velocity / speed is influenced during constant motion by a change in the override (<TO>.Override.Velocity), the "InVelocity" parameter is reset during the acceleration or deceleration. When the newly calculated velocity / speed is reached ("Velocity" × "Override" %), then "InVelocity" is reset.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

MC_MoveVelocity: Function chart V2 (S7-1500)

Function chart: Moving an axis with velocity specification, and the response to an overriding job

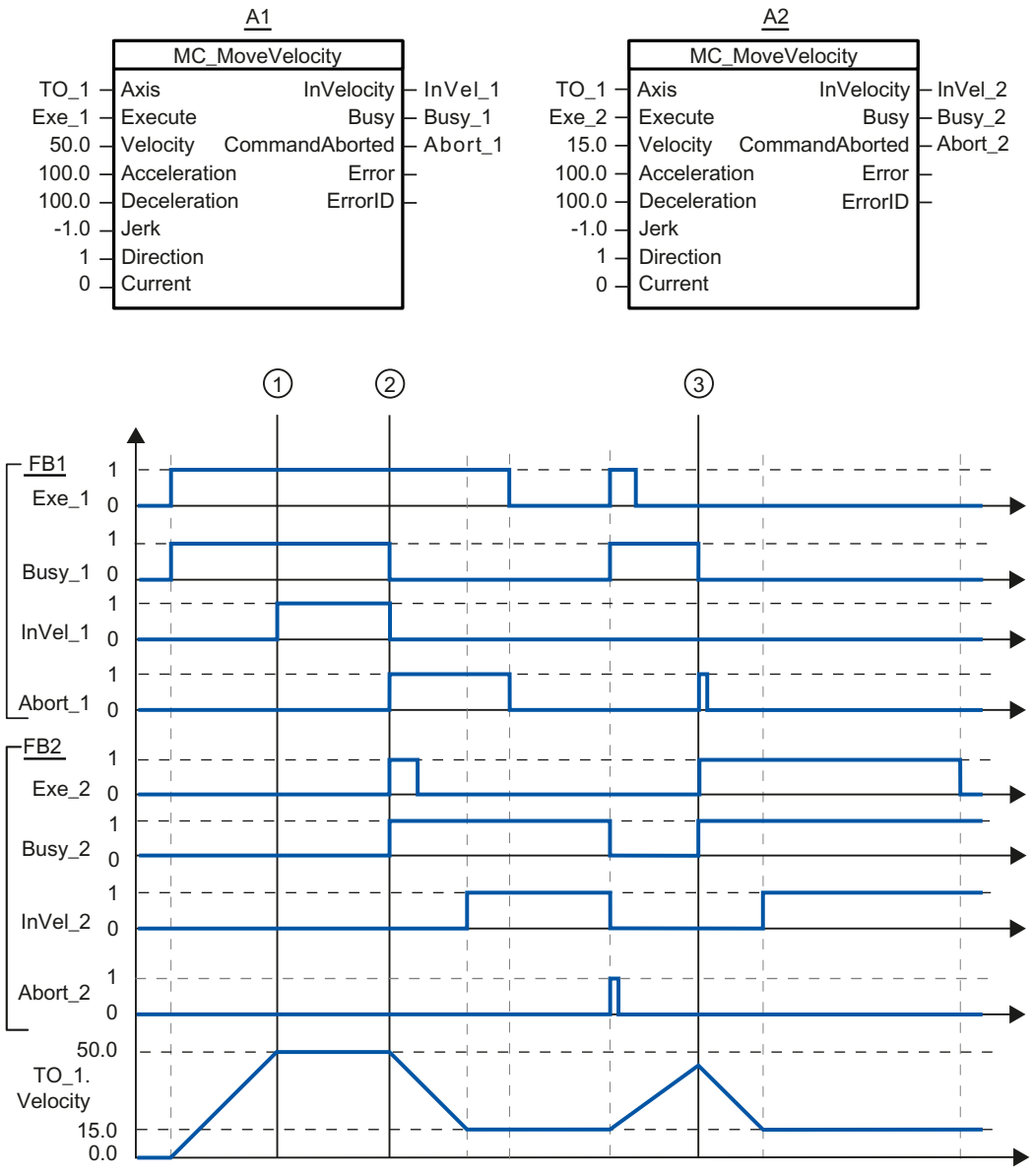

An "MC_MoveVelocity" job (A1) initiated via "Exe_1" accelerates the axis and signals at time ① via "InVel_1" that the velocity setpoint 50.0 has been reached.

At time ②, the job is overridden by another "MC_MoveVelocity" job (A2). The abort is signaled via "Abort_1". When the new velocity setpoint 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the constant velocity 15.0.

The running "MC_MoveVelocity" job (A2) is overridden by another "MC_MoveVelocity" job (A1). The abort is signaled via "Abort_2". The axis is accelerated to the new velocity setpoint 50.0. Before the velocity setpoint is reached, the current "MC_MoveVelocity" job A1) is

overridden at time ③ by another "MC_MoveVelocity" job A2). The abort is signaled via "Abort_1". When the new velocity setpoint 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the constant velocity 15.0.

MC_MoveJog V2 (S7-1500)

MC_MoveJog: Move axis in jog mode V2 (S7-1500)

Description

With the Motion Control instruction "MC_MoveJog", you move an axis in jog mode.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

- Positioning axis/synchronous axis:
  A velocity is specified in the "Velocity" parameter.
- Speed axis:
  A speed is specified in the "Velocity" parameter.

Applies to

- Synchronous axis
- Positioning axis
- Speed axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The "MC_MoveJog" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command

Starting an "MC_MoveJog" job aborts the following active Motion Control jobs:

- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command
- "MC_MoveSuperimposed" command

Parameters

The following table shows the parameters of the Motion Control instruction "MC_MoveJog":

Parameters	Declaration	Data type	Default value	Description
Axis	InOut	TO_SpeedAxis	-	Technology object
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".
Velocity	INPUT	LREAL	100.0	Velocity setpoint / speed setpoint for the motion Value > 0.0: The specified value is used. Value < 0.0: The absolute value of the specified value is used. ("Velocity" = 0.0 is permitted)
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)

Parameters	Declaration	Data type	Default value	Description	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used. Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity setpoint / speed setpoint was reached and will be maintained.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Behavior with velocity setpoint / speed setpoint zero ("Velocity" = 0.0)

An "MC_MoveJog" job with "Velocity" = 0.0 stops the axis with the configured deceleration. When the velocity setpoint / speed setpoint zero is reached, the parameter "InVelocity" will indicate the value TRUE.

Under "Technology object > Diagnostics > Status and error bits > Motion status", "constant velocity" and "standstill" will be displayed (<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill)).

Moving an axis in jog mode

Proceed as follows to move an axis in jog mode:

1. Check the requirements indicated above.
2. Move the axis in the positive direction with "JogForward", or in the negative direction with "JogBackward".

The current motion state is indicated in "Busy", "InVelocity" and "Error".

Note

Response to a change in the override

If the velocity / speed is influenced during constant motion by a change in the override (<TO>.Override.Velocity), the "InVelocity" parameter is reset during the acceleration or deceleration. When the newly calculated velocity is reached ("Velocity" × "Override" %), then "InVelocity" is set again.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

MC_MoveJog: Function chart V2 (S7-1500)

Function chart: Moving an axis in jog mode

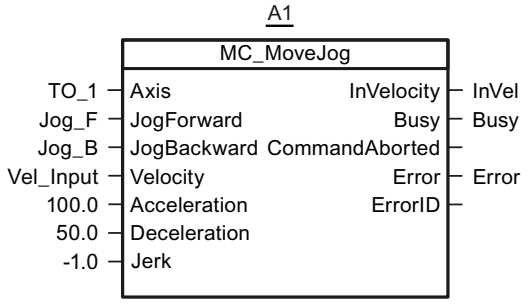

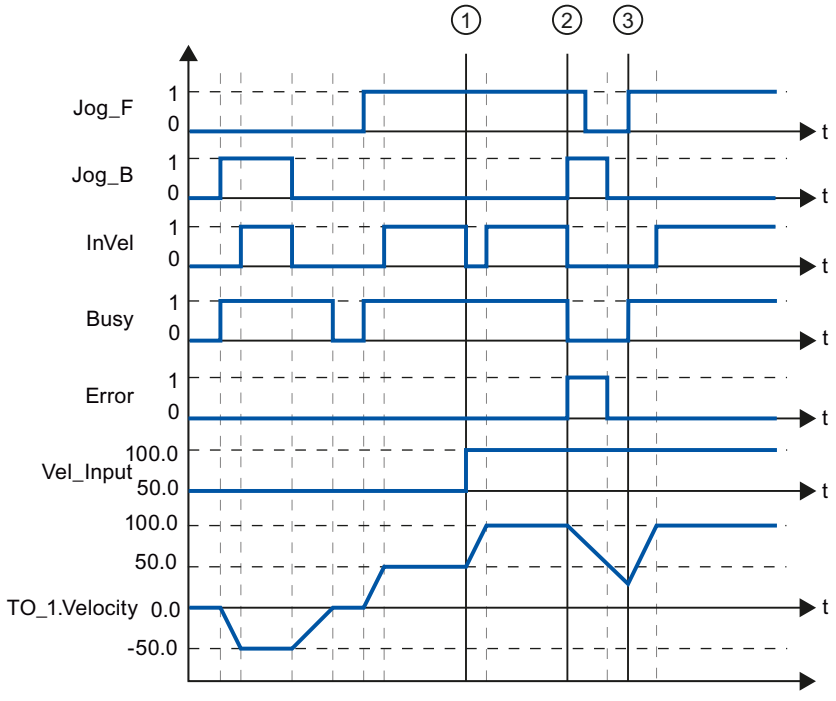

The axis is moved in the negative direction in jog mode via "Jog_B". When the velocity setpoint -50.0 is reached, this is signaled via "InVel" = TRUE. After "Jog_B" is reset, the axis is braked and brought to a standstill. Then the axis is moved in the positive direction via "Jog_F". When the velocity setpoint 50.0 is reached, this is signaled via "InVel" = TRUE.

At time ①, if "Jog_F" is set, the velocity setpoint is changed to 100.0 by means of "Vel_Input". Alternatively, you can also change the velocity setpoint using the velocity override. "InVel" is

reset. Axis is being accelerated. When the new velocity setpoint 100.0 is reached, this is signaled via "InVel" = TRUE.

If "Jog_F" is set, "Jog_B" is likewise set at time ②. If both "Jog_F" and "Jog_B" are set, then the axis is braked with the last applicable deceleration. An error is indicated via "Error", and the "ErrorID" of the error 16#8007 (incorrect direction specification) is output.

This error is resolved by resetting the two inputs "Jog_F" and "Jog_B".

During the braking ramp, "Jog_F" is set at time ③. The axis is accelerated to the last configured velocity. When the velocity setpoint 100.0 is reached, this is signaled via "InVel" = TRUE.

MC_MoveSuperimposed V2 (S7-1500)

MC_MoveSuperimposed: Superimposed positioning of axes V2 (S7-1500)

Description

With the Motion Control instruction "MC_MoveSuperimposed", you start a relative positioning motion that is superimposed on a active basic motion.

You define the dynamic behavior of the motion with parameters "VelocityDiff", "Jerk", "Acceleration" and "Deceleration" . The dynamic values are added to the values of the basic motion. The duration of the basic motion is not extended by a superimposed motion.

The dynamics of the total axis motion is the sum of the dynamic values of the basic motion and the superimposed motion.

The behavior of the total motion depends on the type of basic motion:

- The basic motion is a single-axis motion:
  - The maximum dynamics of the superimposed motion is the difference between the current dynamic values of the basic motion and the dynamic limits.
  - The entire motion is limited to the configured dynamic limits.
- The basic motion is a synchronous motion:
  - The maximum dynamics of the superimposed motion is the difference between the current dynamic values of the basic motion and the dynamic limits.
  - The synchronous motion of the following axis is not limited to the dynamic limits of the following axis.
  - An "MC_MoveSuperimposed" job on a leading axis in synchronous operation affects the leading axis and the following axis.
  - An "MC_MoveSuperimposed" job on a following axis in synchronous operation only affects the following axis.

It is always the dynamics of the total motion that is displayed in the technology data block and in the TIA Portal.

Applies to

- Synchronous axis
- Positioning axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The "MC_MoveSuperimposed" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command
- "MC_MoveSuperimposed" command

The start of an "MC_MoveSuperimposed" job aborts an active "MC_MoveSuperimposed" job.

Parameters

The following table shows the parameters of Motion Control instruction "MC_MoveSuperimposed":

Parameters	Declaration	Data type	Default value	Description
Axis	InOut	TO_PositioningAxis	-	Axis technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Distance	INPUT	LREAL	0.0	Additional travel distance for the superimposing positioning operation (negative or positive)
VelocityDiff	INPUT	LREAL	-1.0	Maximum velocity deviation compared to the active motion Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)

Parameters	Declaration	Data type	Default value	Description	
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	Superimposed positioning complete
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Starting superimposed positioning motion

To start a superimposed positioning motion with Motion Control instruction "MC_MoveSuperimposed", follow these steps:

1. Check the requirements indicated above.
2. Specify the additional distance to be moved in the "Distance" parameter.
3. Start the "MC_MoveSuperimposed" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error".

MC_MoveSuperimposed: Function chart V2 (S7-1500)

Function chart: Positioning axes overlapping

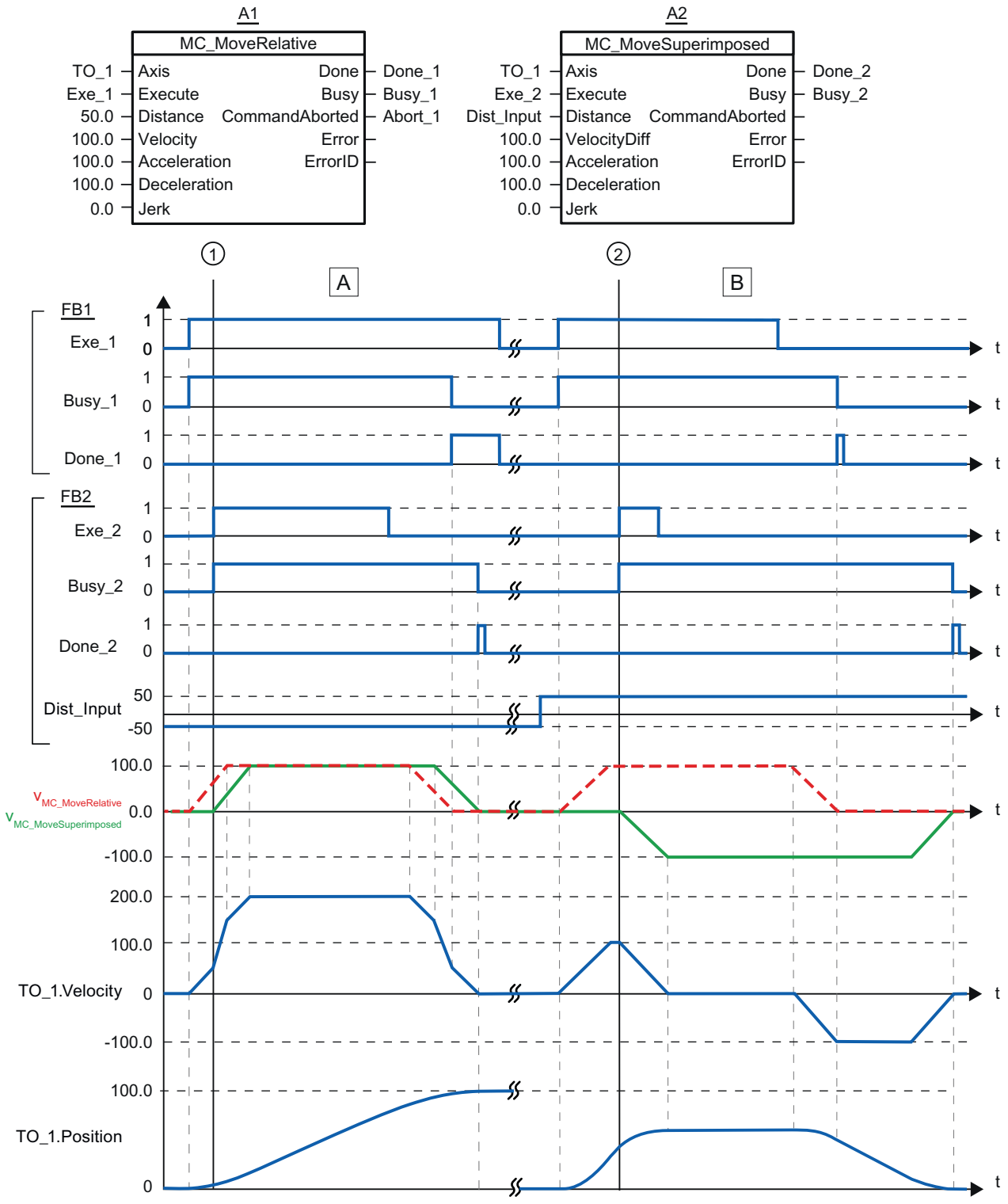

Section A	Using "Exe_1", an "MC_MoveRelative" job with a distance of 50.0 is initiated. At time ①, using "Exe_2", an MC_MoveSuperimposed job with a distance of 50.0 is initiated. The axis is moved with the added dynamic values of both jobs by the distance 50 + 50 = 100.0. When the axis reaches the target position, this is signaled via "Done_2".
Section B	Using "Exe_1", an "MC_MoveRelative" job with a distance of 50.0 is initiated. At time ②, using "Exe_2", a MC_MoveSuperimposed job with a distance of -50.0 is initiated. The axis reverses and is moved with the added dynamic values of both jobs by the distance 50.0 - 50.0 = 0.0. When the axis reaches the target position, this is signaled via "Done_2".

Synchronous motion (S7-1500)

MC_GearIn V2 (S7-1500)

MC_GearIn: Start relative gearing V2 (S7-1500)

Description

With the Motion Control instruction "MC_GearIn", you start relative gearing between a leading axis and a following axis.

You define the dynamic behavior of the following axis during synchronization with parameters "Jerk", "Acceleration" and "Deceleration" .

You specify the gear ratio as the relationship between two integers (numerator/denominator) with the parameters "RatioNumerator" and "RatioDenominator". You can change the gear ratio during operation by issuing a new "MC_GearIn" job. Transition processes are executed with the specified dynamic values.

The numerator of the gear ratio is specified as positive or negative. This yields the following behavior:

- **Positive gear ratio:**
  The leading and following axes move in the same direction.
- **Negative gear ratio**:
  The following axis rotates in the opposite direction of the leading axis.

You can start synchronous operation when the leading axis is at a standstill or when it is in motion.

A position offset can occur between leading axis and following axis during synchronization. This position offset depends on the start time of the "MC_GearIn" and the duration of the synchronization process. The position offset is not compensated.

Applies to

- Synchronous axis

Requirements

- The technology object has been configured correctly.
- The leading axis is a positioning axis or synchronous axis.
- The following axis is a synchronous axis.
- The leading axis is specified as possible leading axis in the configuration of the following axis in "Technology object > Configuration > Leading value interconnections".
- The following axis is enabled.

Override response

The "MC_GearIn" job is aborted by:

- Disabling the following axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command

The start of an "MC_GearIn" job aborts the following active Motion Control jobs:

- "MC_Home" command "Mode" = 3, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command
- "MC_GearIn" command
- "MC_MoveSuperimposed" command

Disabling the leading axis with "MC_Power.Enable" = FALSE does not abort synchronous operation. The following axis follows the leading axis even during the braking ramp and after the leading axis is enabled again.

Parameters

The following table shows the parameters of Motion Control instruction "MC_GearIn":

Parameters	Declaration	Data type	Default value	Description	
Master	InOut	TO_PositioningAxis	-	Leading axis technology object	
Slave	InOut	TO_SynchronousAxis	-	Following axis technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
RatioNumerator	INPUT	DINT	1	Gear ratio numerator Permitted integer values: -2147483648 to 2147483648 (value 0 not permitted)	
RatioDenominator	INPUT	DINT	1	Gear ratio denominator Permitted integer values: 1 to 2147483648	
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
InGear	OUTPUT	BOOL	FALSE	TRUE	Synchronous operation reached The following axis is synchronized and moves synchronously to the leading axis.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.

Parameters	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred while processing the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	0	Error ID for parameter "ErrorID"	

Starting synchronous operation

To start synchronous operation with the Motion Control instruction "MC_GearIn", follow these steps:

1. Check the requirements indicated above.
2. Specify the leading axis, the following axis and the gear ratio at the corresponding parameters.
3. Start the "MC_GearIn" job with a positive edge at parameter "Execute".

The following axis is synchronized to the leading value of the leading axis. If the "InGear" parameter shows the value TRUE, the following axis is synchronized and moves synchronously to the leading axis. The parameters "InGear" and "Busy" show the value TRUE until the "MC_GearIn" job is overridden by another Motion Control job.

MC_GearIn: Function chart V2 (S7-1500)

Function chart: Synchronizing and switching the leading value

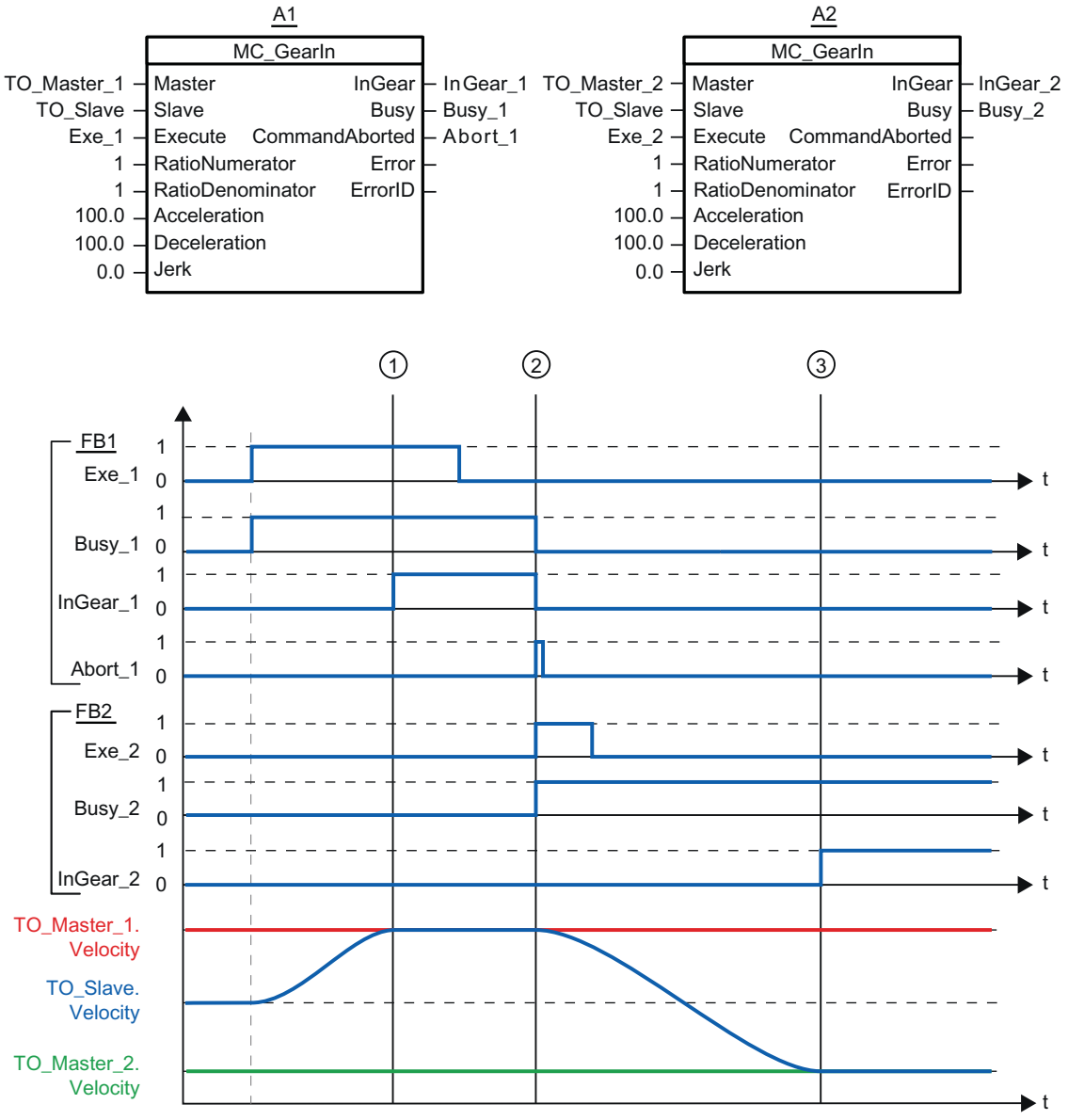

Using "Exe_1", an "MC_GearIn" job (A1) is initiated. The following axis (TO_Slave) is synchronized to the leading axis (TO_Master_1). "InGear_1" signals at time ① that the following axis is synchronized and moves synchronously to the leading axis.

At time ②, synchronous operation is overridden by another "MC_GearIn" job (A2). The abort is signaled via "Abort_1". The following axis is synchronized to the new leading axis (TO_Master_2). "InGear_2" signals at time ③ that the following axis is synchronized and moves synchronously to the leading axis.

S7-1500 Motion Control V1 (S7-1500)

MC_Power V1 (S7-1500)

MC_Power: Enable, disable technology object V1 (S7-1500)

Description

With the Motion Control instruction "MC_Power", a technology object is enabled or disabled.

Applies to

- Positioning axis
- Speed axis
- External encoder

Requirements

- The technology object has been configured correctly.

Override response

- An "MC_Power" job cannot be aborted by any other Motion Control job.
- An "MC_Power" job with parameter "Enable" = TRUE enables a technology object but does not thereby abort any other Motion Control instructions.
- Disabling the technology object (parameter "Enable" = FALSE) aborts all Motion Control jobs on the corresponding technology object in accordance with the selected "StopMode". This process cannot be canceled by the user.

Parameters

The following table shows the parameters of the "MC_POWER" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description	
Axis	InOut	TO_Axis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The technology object is enabled.
				FALSE	The technology object is disabled. All current jobs at the technology object are aborted in accordance with the configured "StopMode".

Parameter	Declaration	Data type	Default value	Description	
StopMode	INPUT	INT	0	Not applicable to the technology object external encoder If you disable a technology object with a negative edge at parameter "Enable", the axis decelerates in accordance with the selected "StopMode".	
				0	Emergency stop When the technology object is disabled, the axis brakes to a standstill without jerk limit, using the emergency stop deceleration configured in "Technology object > Configuration > Extended parameters > Emergency stop ramp". The enable is then canceled. (<TO>.DynamicDefaults.EmergencyDeceleration)
				1	Immediate stop When a technology object is disabled, the setpoint zero is output, and the enable is canceled. The axis is braked to a standstill according to the configuration in the drive.
				2	Stop with maximum dynamic values When the technology object is disabled, the axis is braked to a standstill using the maximum deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic limits". The configured maximum jerk is hereby taken into account. The enable is then canceled. (<TO>.DynamicLimits.MaxDeceleration; <TO>.DynamicLimits.MaxJerk)
Status	OUTPUT	BOOL	FALSE	Technology object enable status	
				FALSE	Disabled - A positioning axis or speed axis does not accept any Motion Control jobs. - Speed control and position control are not active. - The actual values of the technology object are not checked for validity.
				TRUE	Enabled - An enabled positioning axis or speed axis accepts Motion Control jobs. - Speed control and position control are active. - The actual values of the technology object are valid.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction MC_Power. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Enabling technology objects

To enable a technology object, set the "Enable" parameter to TRUE.

When the Status parameter shows the value TRUE , the technology object is enabled.

If an axis is in motion when the technology object is enabled (actual velocity is present), the axis brakes to the setpoint zero, using the maximum deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic limits" (<TO>.DynamicLimits.MaxDeceleration). This braking ramp can be overridden by Motion Control jobs.

Note

Automatic enable after acknowledgment of a technology alarm

If the technology object is disabled due to a technology alarm, it will automatically be enabled again after the cause has been eliminated and the alarm has been acknowledged. This requires the "Enable" parameter to have retained the value TRUE during this process.

Disabling technology objects

To disable a technology object, set the "Enable" parameter to FALSE.

If an axis is in motion, it is braked to a standstill according to the selected "StopMode".

When the "Busy" and "Status" parameters show the value FALSE , the disabling of the technology object is complete.

Drive connection by means of PROFIdrive

When a drive is connected using PROFIdrive, the setpoint, enable and drive status are transmitted via the PROFIdrive telegram.

- **Enable technology object and activate drive**
  With parameter "Enable" = TRUE, the technology object is enabled. The drive is enabled according to the PROFIdrive standard.
  When the tag <TO>.StatusDrive.InOperation shows the value TRUE , the drive is ready to execute setpoints. The "Status" parameter is set to the value TRUE .
- **Disable technology object and deactivate drive**
  With parameter "Enable" = FALSE, the "Status" parameter is set to the value FALSE, and the axis is braked according to the selected "StopMode". The drive is disabled according to the PROFIdrive standard.

Analog drive connection

The setpoint is output via an analog output. Optionally, you can configure an enabling signal via digital output (<TO>.Actor.Interface.EnableDriveOutput), and a readiness signal via digital input (<TO>.Actor.Interface.DriveReadyInput).

- **Enable technology object and activate drive**
  With parameter "Enable" = TRUE, the enable output ("Enable drive output") is set.
  When the drive returns the readiness signal via the ready input ("Drive ready input"), the "Status" parameter and the technology object's <TO>.StatusDrive.InOperation tag are set to TRUE, and the setpoint is switched to the analog output.
- **Disable technology object and deactivate drive**
  With parameter "Enable" = FALSE, the "Status" parameter is set to the value FALSE, and the axis is braked according to the selected "StopMode". When the setpoint zero is reached, the enable output is set to FALSE.

Additional information

Additional information on enabling and disabling technology objects and drives can be found in the appendix, MC_Power Function Charts.

MC_Power: Function chart V1 (S7-1500)

Function chart: Enabling a technology object, and example for alarm response

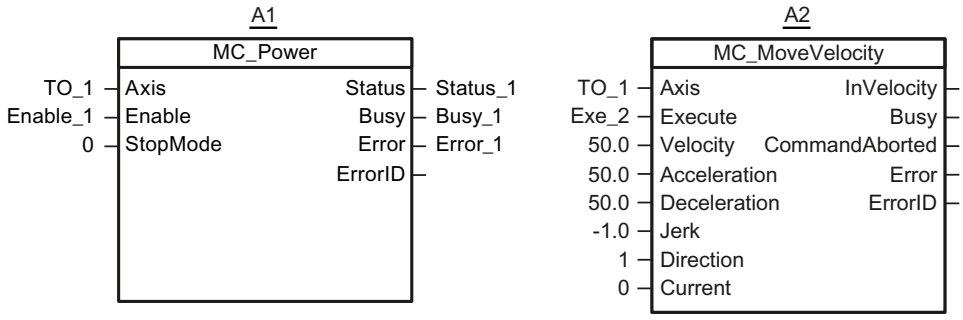

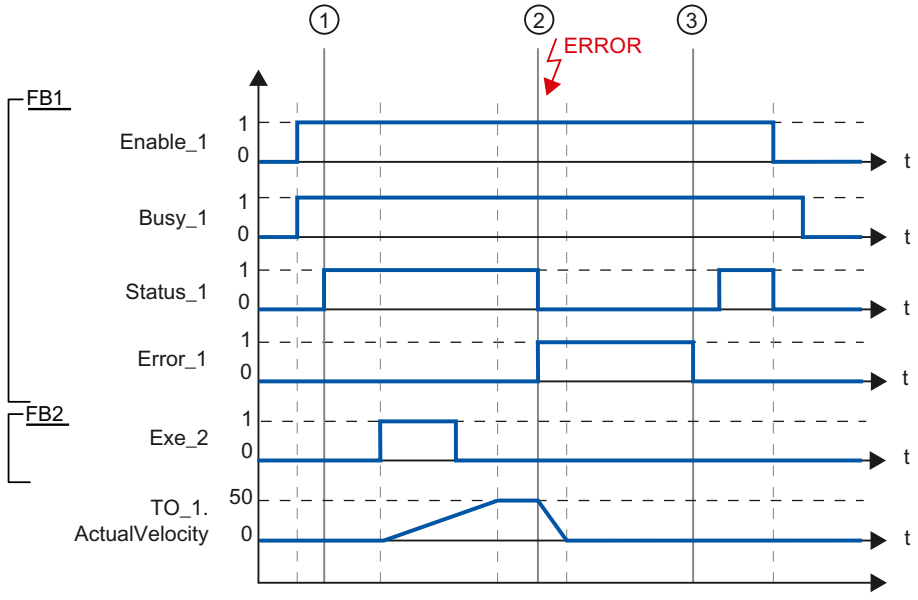

A technology object is enabled with "Enable_1= TRUE". Successful enabling can be read from "Status_1" at time ①. The axis will then move with an "MC_MoveVelocity" job (A2). The velocity profile of the axis can be read from "Velocity Axis_1".

At time ② an error occurs in the technology object, which results in the disabling of the technology object (alarm response: Remove enable). The axis is braked depending on the configuration in the drive, and is brought to a standstill. When the technology object is disabled, Status_1 is reset. Since the axis was not disabled using "Enable_1" = FALSE, the selected "StopMode" does not apply. The cause of the error is corrected and the alarm is acknowledged at time ③.

Since "Enable_1" is still set, the technology object is enabled again. Successful enabling can be read from "Status_1". Next the technology object is disabled with "Enable_1" = FALSE.

MC_Reset V1 (S7-1500)

MC_Reset: Acknowledge alarms, restart technology object V1 (S7-1500)

Description

With the Motion Control instruction "MC_Reset", you acknowledge all technology alarms that can be acknowledged in the user program. Acknowledgment also resets the "Error" and "Warning" bits in the technology data block.

With the Motion Control instruction "MC_Reset" with "Restart" = TRUE, you start reinitialization (restart) of technology objects. Upon restart of the technology object, the new configuration data are applied in the technology data block.

Applies to

- Positioning axis
- Speed axis
- External encoder

Requirements

- The cause of the error has been resolved for a pending technology alarm.
- For a restart, the technology object must be disabled.
  ("MC_Power.Status" = FALSE and "MC_Power.Busy" = FALSE)

Override response

- Parameter "Restart" = FALSE:
  Processing of the "MC_Reset" instruction can be aborted by other Motion Control jobs. The MC_Reset job does not abort any running Motion Control jobs.
- Parameter "Restart" = TRUE:
  Processing of the "MC_Reset" instruction with parameter "Restart" = TRUE cannot be aborted by any other Motion Control job.

Parameters

The following table shows the parameters of the "MC_Reset" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description
Axis	InOut	TO_Axis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge

Parameter	Declaration	Data type	Default value	Description	
Restart	INPUT	BOOL	FALSE	TRUE	"Restart" Reinitialization of the technology object and acknowledgment of pending technology alarms. The technology object is reinitialized with the configured start values.
				FALSE	Acknowledgment of queued technology alarms.
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged. Restart has been executed.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Acknowledging technology alarms

To acknowledge technology alarms, follow these steps:

1. Check the requirements indicated above.
2. Set parameter "Restart" = FALSE.
3. Start the acknowledgment of the error with a positive edge at parameter "Execute".

When the Done parameter shows the value TRUE, the error has been acknowledged.

Note

Acknowledging with "Restart" = FALSE

If only the technology alarms are to be acknowledged, set "Restart" = FALSE. The technology object cannot be used during a restart.

Restarting a technology object

To restart a technology object, follow these steps:

1. Check the requirements indicated above.
2. Set parameter "Restart" = TRUE.
3. Perform the restart with a positive edge at parameter "Execute".

When the "Done" parameter shows the value TRUE, the restart of the technology object is complete.

Additional information regarding the restart can be found in the Restarting technology objects section.

MC_Home V1 (S7-1500)

MC_Home: Home technology objects, set home position V1 (S7-1500)

Description

With the Motion Control instruction "MC_Home", you create the relationship between the position in the technology object and the mechanical position. The actual position value in the technology object is assigned to a homing mark at the same time. This homing mark represents a known mechanical position.

Homing is performed according to the mode selected with the "HomingMode" parameter and the configuration under "Technology object > Configuration > Extended parameters > Homing".

The preset values under "Technology object > Configuration > Extended parameters > Dynamic default values" are used for the dynamic values Acceleration, Deceleration and Jerk.

Applies to

- Positioning axis
- External encoder

The following table shows which modes are possible with each of the technology objects:

Operating mode	Positioning axis with incremental encoder	Positioning axis with absolute encoder	External incremental encoder	External absolute encoder
Active homing (HomingMode = 4, 5)	X	-	-	-
Passive homing (HomingMode = 2, 3, 8)	X	-	X	-
Setting a positioning value ("HomingMode" = 0)	X	X	X	X
Relative shift of the actual value ("HomingMode" = 1)	X	X	X	X
Homing absolute encoder ("HomingMode" = 6, 7)	-	X	-	X

Requirements

- The technology object has been configured correctly.
- HomingMode" = 2, 3, 4, 5, 8
  The technology object must be enabled.
- HomingMode" = 0, 1, 2, 6, 7
  The encoder values must be valid. (<TO>.StatusSensor[n].State = 2)

Override response

- An MC_Home job for passive homing is aborted by:
  - Disabling the technology object with "MC_Power.Enable" = FALSE
  - "MC_Home" command with parameter "HomingMode" = 4, 5, 9
- An "MC_Home" job for passive homing does not abort any other Motion Control jobs.
- An "MC_Home" job for active homing is aborted by:
  - Disabling the technology object with "MC_Power.Enable" = FALSE
  - "MC_Home" command "HomingMode" = 4, 5
  - "MC_Halt" command
  - "MC_MoveAbsolute" command
  - "MC_MoveRelative" command
  - "MC_MoveVelocity" command
  - "MC_MoveJog" command
- An "MC_Home" job for active homing aborts the following active Motion Control jobs.
  - MC_Home" command with parameter "HomingMode" = 2, 3, 4, 5
  - "MC_Halt" command
  - "MC_MoveAbsolute" command
  - "MC_MoveRelative" command
  - "MC_MoveVelocity" command
  - "MC_MoveJog" command

Parameters

The following table shows the parameters of the "MC_Home" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description
Axis	InOut	TO_Axis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Position	INPUT	LREAL	0.0	The specified value is used according to the selected "HomingMode".

Parameter	Declaration	Data type	Default value	Description	
HomingMode	INPUT	INT	4	Operating mode	
				0	Direct homing (absolute) The current position of the technology object is set to the value of parameter "Position".
				1	Direct homing (relative) The current position of the technology object is shifted by the value of parameter "Position".
				2	Passive homing When the homing mark is detected, the actual value is set to the value of the "Position" parameter.
				3	Passive homing ("Position" parameter has no effect) When the homing mark is detected, the actual value is set to the home position configured under "Technology object > Configuration > Extended parameters > Homing > Passive homing". (<TO>.Homing.HomePosition)
				4	Active homing The TO positioning axis performs a homing movement according to the configuration. After the completion of the movement, the axis is positioned at the value of the "Position" parameter.
				5	Active homing ("Position" parameter has no effect) The TO positioning axis performs a homing movement according to the configuration. After completion of the movement, the axis is positioned at the home position configured under "Technology object > Configuration > Extended parameters > Homing > Active homing". (<TO>.Homing.HomePosition)
				6	Absolute encoder adjustment (relative) The current position is shifted by the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<TO>.StatusSensor[n].AbsEncoderOffset)
				7	Absolute encoder adjustment (absolute) The current position is set to the value of parameter "Position". The calculated absolute value offset is stored retentively in the CPU.

Parameter	Declaration	Data type	Default value	Description	
					(<TO>.StatusSensor[n].AbsEncoderOffset)
				8	Passive homing (without reset) Functions like "HomingMode" 2, with the difference that the "homed" status is **not** reset when the function is enabled.
				9	Abort passive homing An active job for passive homing is aborted.
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Resetting the "Homed" status

The "Homed" status of a technology object is reset under the following conditions (<TO>.StatusWord.X5 (HomingDone)):

- **Technology objects with incremental actual values:**
  - Starting an "MC_Home" job with "HomingMode" = 2, 3, 4, 5
    (After successful completion of the homing operation, the "Homed" status is reset.)
  - Error in the encoder system, or encoder failure
  - Restart of the technology object
  - After POWER OFF -> POWER ON of the CPU
  - Memory reset
  - Modification of the encoder configuration
- **Technology objects with absolute actual values:**
  - Restoration of the CPU factory settings
  - Modification of the encoder configuration
  - Replacement of the CPU

Homing a technology object with "HomingMode" = 1..8

To home a technology object, follow these steps:

1. Check the requirements indicated above.
2. Specify the desired homing function in the "HomingMode" parameter.
3. Initialize the necessary parameters with values, and start the homing operation with a positive edge at the "Execute" parameter.

When the "Done" parameter shows the value TRUE, the "MC_Home" job has been completed according to the selected "HomingMode". The "Homed" status of the technology object is indicated under "Technology object > Diagnostics > Status and error bits > Motion status > Homed" (<TO>.StatusWord.X5 (HomingDone)).

Termination of a passive homing process with "HomingMode" = 9

With "HomingMode" = 9, the technology object is not homed. If a running "MC_Home" job for passive homing ("HomingMode" = 2, 3, 8) is overridden by another "MC_Home" job with "HomingMode" = 9, then the running job is terminated with the parameter "CommandAborted" = TRUE. The overriding job with "HomingMode" = 9 signals successful execution with parameter "Done" = TRUE.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

MC_Halt V1 (S7-1500)

MC_Halt: Halt axis V1 (S7-1500)

Description

With the Motion Control instruction "MC_Halt", you brake an axis until it comes to a standstill.

You define the dynamic behavior of the braking operation with parameters "Jerk" and "Deceleration".

Applies to

- Positioning axis
- Speed axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The "MC_Halt" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "HomingMode" = 4, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command

Starting a "MC_Halt" job aborts the following active Motion Control jobs:

- "MC_Home" command "HomingMode" = 4, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command

Parameters

The following table shows the parameters of the "MC_Halt" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description	
Axis	InOut	TO_SpeedAxis	-	Technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.

Parameter	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Braking an axis with "MC_Halt"

Proceed as follows to decelerate an axis to a standstill:

1. Check the requirements indicated above.
2. Supply the parameters "Deceleration" and "Jerk" with the desired values.
3. Start the "MC_Halt" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error". The standstill of the axis is indicated under "Technology object > Diagnostics > Status and error bits > Motion status > Standstill" (<TO>.StatusWord.X7 (Standstill)).

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

MC_Halt: Function chart V1 (S7-1500)

Function chart: Stopping an axis, and the response to an overriding job

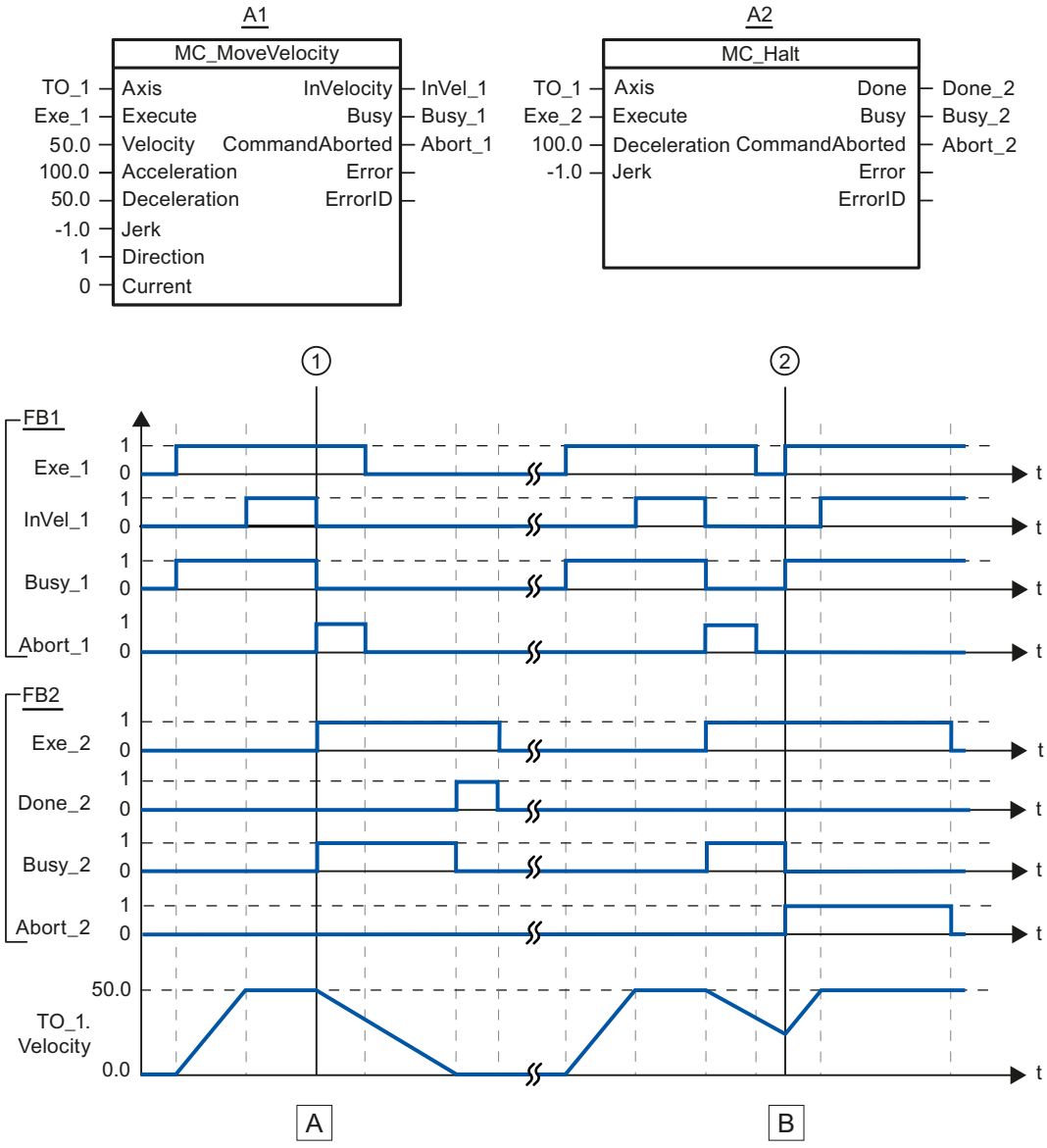

Section A	An axis is moved with an "MC_MoveVelocity" job (A1). When the setpoint velocity 50.0 is reached, this is signaled via "InVel_1". At time ①, the "MC_MoveVelocity" job is overridden by an "MC_Halt" job (A2). The job termination is signaled via "Abort_1". The axis is braked to a standstill. Successful completion of the "MC_Halt" job is signaled via "Done_2".
Section B	The axis is moved with an "MC_MoveVelocity" job (A1). When the setpoint velocity 50.0 is reached, this is signaled via "InVel_1". Next the "MC_MoveVelocity" job is overridden by an "MC_Halt" job (A2). The job termination is signaled via "Abort_1". During the braking process the "MC_Halt" job is overridden at time ② by an "MC_Move-Velocity" job (A1). The job termination is signaled via "Abort_2". When the setpoint velocity 50.0 is reached, this is signaled via "InVel_1". The axis is then moved with constant velocity.

MC_MoveAbsolute V1 (S7-1500)

MC_MoveAbsolute: Absolute positioning of axis V1 (S7-1500)

Description

With the Motion Control instruction "MC_MoveAbsolute", you can move an axis to an absolute position.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

Applies to

- Positioning axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.
- The technology object is homed.

Override response

The "MC_MoveAbsolute" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "HomingMode" = 4, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command

The start of an "MC_MoveAbsolute" job aborts the following active Motion Control jobs:

- "MC_Home" command "HomingMode" = 4, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command

Parameters

The following table shows the parameters of the "MC_MoveAbsolute" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	InOut	TO_Positioning-Axis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Position	INPUT	REAL	0.0	Absolute target position
Velocity	INPUT	LREAL	-1.0	Velocity setpoint for the positioning Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
Direction	INPUT	INT	1	Direction of rotation of the axis Is only evaluated with "modulo" enabled. "Technology object > Configuration > Basic parameters > Enable modulo 1 Positive direction of rotation 2 Negative direction of rotation 3 Shortest path
Done	OUTPUT	BOOL	FALSE	TRUE Target position reached.
Busy	OUTPUT	BOOL	FALSE	TRUE Job is running.

Parameters	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Moving an axis to an absolute position

Proceed as follows to move an axis to an absolute position:

1. Check the requirements indicated above.
2. Specify the desired target position in the "Position" parameter.
3. Start the "MC_MoveAbsolute" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error".

MC_MoveAbsolute: Function chart V1 (S7-1500)

Function chart: Absolute positioning of an axis, and the response to an overriding job

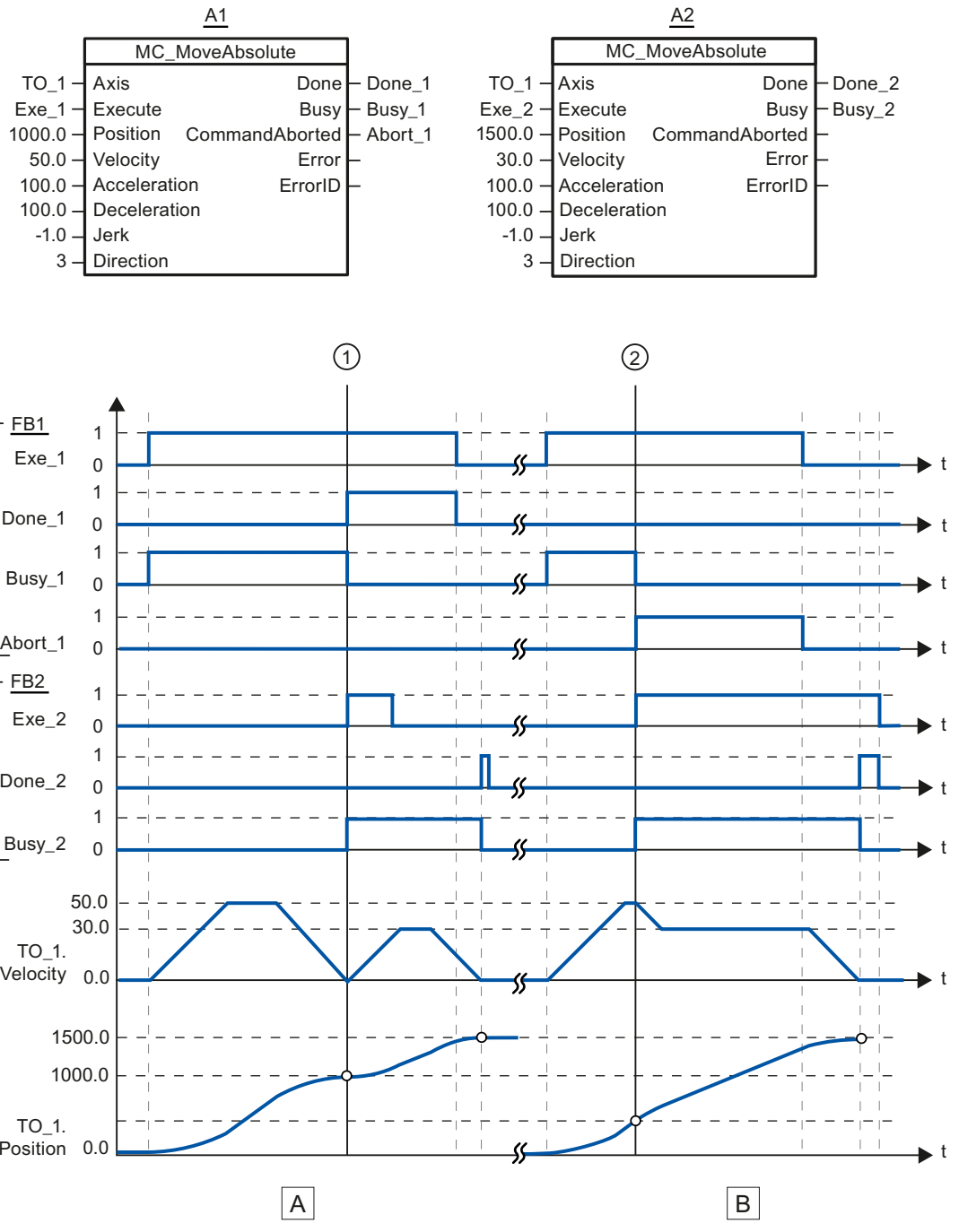

Section A	An axis is moved to absolute position 1000.0 with an "MC_MoveAbsolute" job (A1). When the axis reaches the target position, this is signaled at time ① via "Done_1". At this time ①, another "MC_MoveAbsolute" job (A2) with target position 1500.0 is started. When the axis reaches the target position 1500.0, this is signaled via "Done_2". Since "Exe_2" was previously reset, "Done_2" is applied only to one cycle.
Section B	A running "MC_MoveAbsolute" job (A1) is overridden at time ② by another "MC_MoveAbsolute" job (A2). The abort is signaled via "Abort_1". The axis is braked to the changed velocity and moved to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveRelative V1 (S7-1500)

MC_MoveRelative: Relative positioning of axis V1 (S7-1500)

Description

With the Motion Control instruction "MC_MoveRelative", you move an axis relative to its position when execution of the job began.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

Applies to

- Positioning axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The "MC_MoveRelative" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "HomingMode" = 4, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command

Starting a "MC_MoveRelative" job aborts the following active Motion Control jobs:

- "MC_Home" command "HomingMode" = 4, 5
- "MC_Halt" command

- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command

Parameters

The following table shows the parameters of the "MC_MoveRelative" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description	
Axis	InOut	TO_Positioning-Axis	-	Technology object	
Execute	INPUT	BOOL	FALSE	Start job with a positive edge	
Distance	INPUT	LREAL	0.0	Distance for the positioning process (negative or positive)	
Velocity	INPUT	LREAL	-1.0	Velocity setpoint for the positioning Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Velocity)	
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.

Parameter	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Moving an axis relative to the starting position

Proceed as follows to move an axis relative to the starting position:

1. Check the requirements indicated above.
2. Specify distance to be moved in the "Distance" parameter.
3. Start the "MC_MoveRelative" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error".

MC_MoveRelative: Function chart V1 (S7-1500)

Function chart: Relative positioning of an axis, and the response to an overriding job

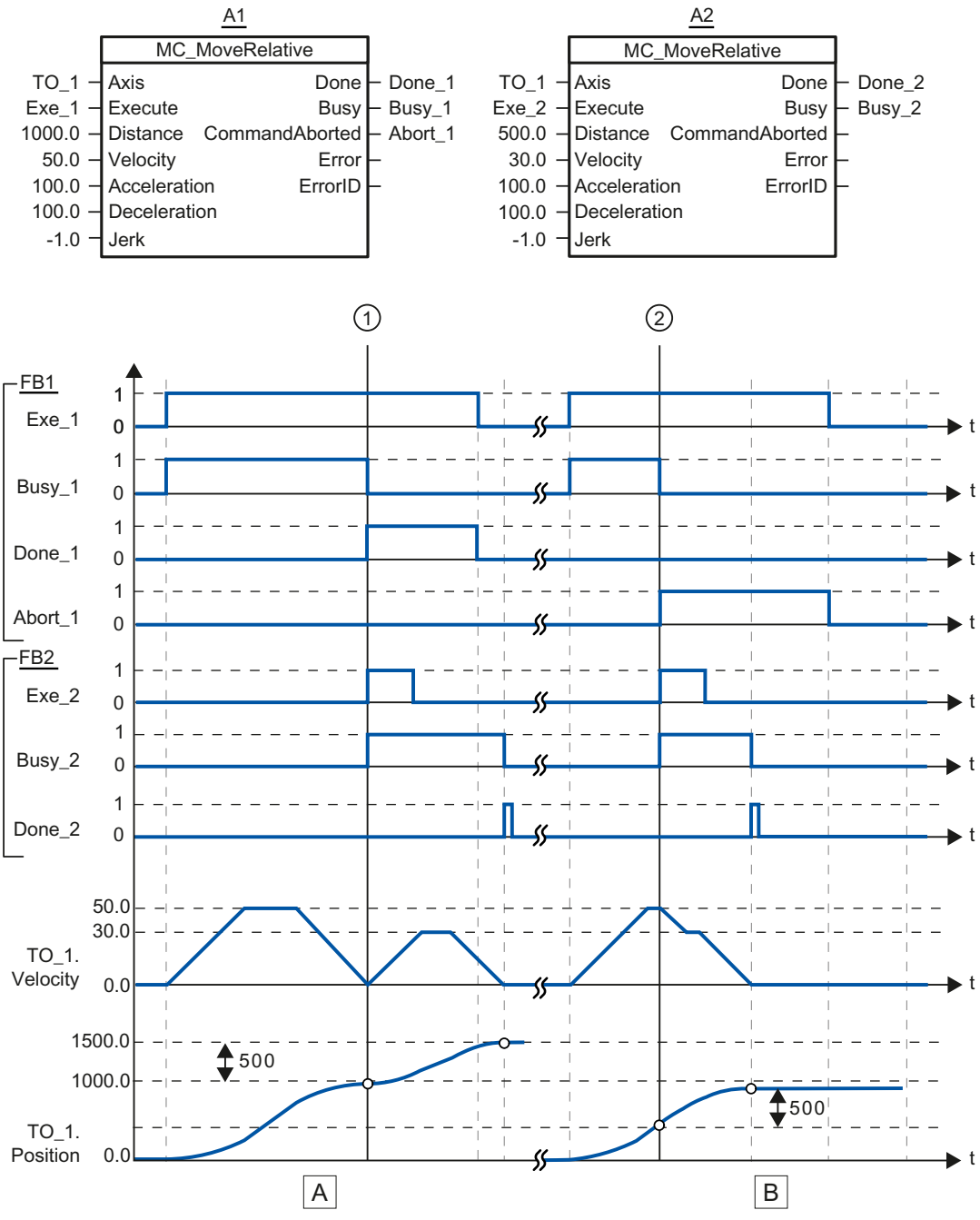

Section A	The axis is moved by an "MC_MoveRelative" job (A1) by the distance ("Distance") 1000.0 (the starting position here is 0.0). When the axis reaches the target position, this is signaled at time ① via "Done_1". At this time ①, another "MC_MoveRelative" job (A2) with distance 500.0 is started. When the axis reaches the new target position, this is signaled via "Done_2". Since "Exe_2" was previously reset, "Done_2" is applied only to one cycle.
Section B	A running "MC_MoveRelative" job (A1) is overridden by another "MC_MoveRelative" job (A2). The abort is signaled at time ② via "Abort_1". The axis is then moved at the new velocity by the distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveVelocity V1 (S7-1500)

MC_MoveVelocity: Move axis at set velocity/speed V1 (S7-1500)

Description

With the Motion Control instruction "MC_MoveVelocity", you move an axis at constant velocity/speed.

You define the dynamic behavior of the motion with parameters"Velocity", "Jerk", "Acceleration" and "Deceleration".

- Positioning axis:
  A velocity is specified in the "Velocity" parameter.
- Speed axis:
  A speed is specified in the "Velocity" parameter.

Applies to

- Positioning axis
- Speed axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The "MC_MoveVelocity" is canceled by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "HomingMode" = 4, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command

- "MC_MoveVelocity" command
- "MC_MoveJog" command

The start of an "MC_MoveVelocity" job aborts the following active Motion Control jobs:

- "MC_Home" command "HomingMode" = 4, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command

Parameters

The following table shows the parameters of the "MC_MoveVelocity" Motion Control instruction:

Parameters	Declaration	Data type	Default value	Description
Axis	InOut	TO_SpeedAxis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start job with a positive edge
Velocity	INPUT	LREAL	100.0	Velocity/speed setpoint for the motion process ("Velocity" = 0.0 is permitted)
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used. Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)

Parameters	Declaration	Data type	Default value	Description	
Direction	INPUT	INT	0	Direction of rotation of the axis	
				0	The sign of the velocity specified at the "Velocity" parameter defines the direction of rotation.
				1	Positive direction of rotation Value of "Velocity" is used.
				2	Negative direction of rotation Value of "Velocity" is used.
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	Deactivated The values of parameters "Velocity" and "Direction" are taken into account.
				TRUE	activated The values at the parameters "Velocity" and "Direction" are **not** taken into account. The current velocity and direction at function start are retained. When the axis resumes motion at the velocity that was current at function start, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity setpoint / speed setpoint was reached and will be maintained.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Behavior with velocity setpoint / speed setpoint zero ("Velocity" = 0.0)

An "MC_MoveVelocity" job with "Velocity" = 0.0 stops the axis with the configured deceleration. When the velocity setpoint / speed setpoint zero is reached, the parameter "InVelocity" will indicate the value TRUE.

Under "Technology object > Diagnostics > Status and error bits > Motion status", "constant velocity" and "standstill" will be displayed (<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill)).

The parameters "InVelocity" and "Busy" show the value TRUE until the "MC_MoveVelocity" job is overridden by another Motion Control job.

Moving an axis with constant velocity / speed

Proceed as follows to move an axis with constant velocity / speed:

1. Check the requirements indicated above.
2. At the parameter "Velocity", specify the velocity / speed, with which the axis should be moved.
3. Start the "MC_MoveVelocity" job with a positive edge at parameter "Execute".

The current motion state is indicated in "Busy", "InVelocity" and "Error".

If the "InVelocity" parameter shows the value TRUE, then the velocity setpoint / speed setpoint was reached. The axis continues moving at this constant velocity. The parameters "InVelocity" and "Busy" show the value TRUE until the "MC_MoveVelocity" job is overridden by another Motion Control job.

Note

Response to a change in the override

If the velocity / speed is influenced during constant motion by a change in the override (<TO>.Override.Velocity), the "InVelocity" parameter is reset during the acceleration or deceleration. When the newly calculated velocity / speed is reached ("Velocity" × "Override" %), then "InVelocity" is reset.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

MC_MoveVelocity: Function chart V1 (S7-1500)

Function chart: Moving an axis with specification of velocity, and the response to an overriding job

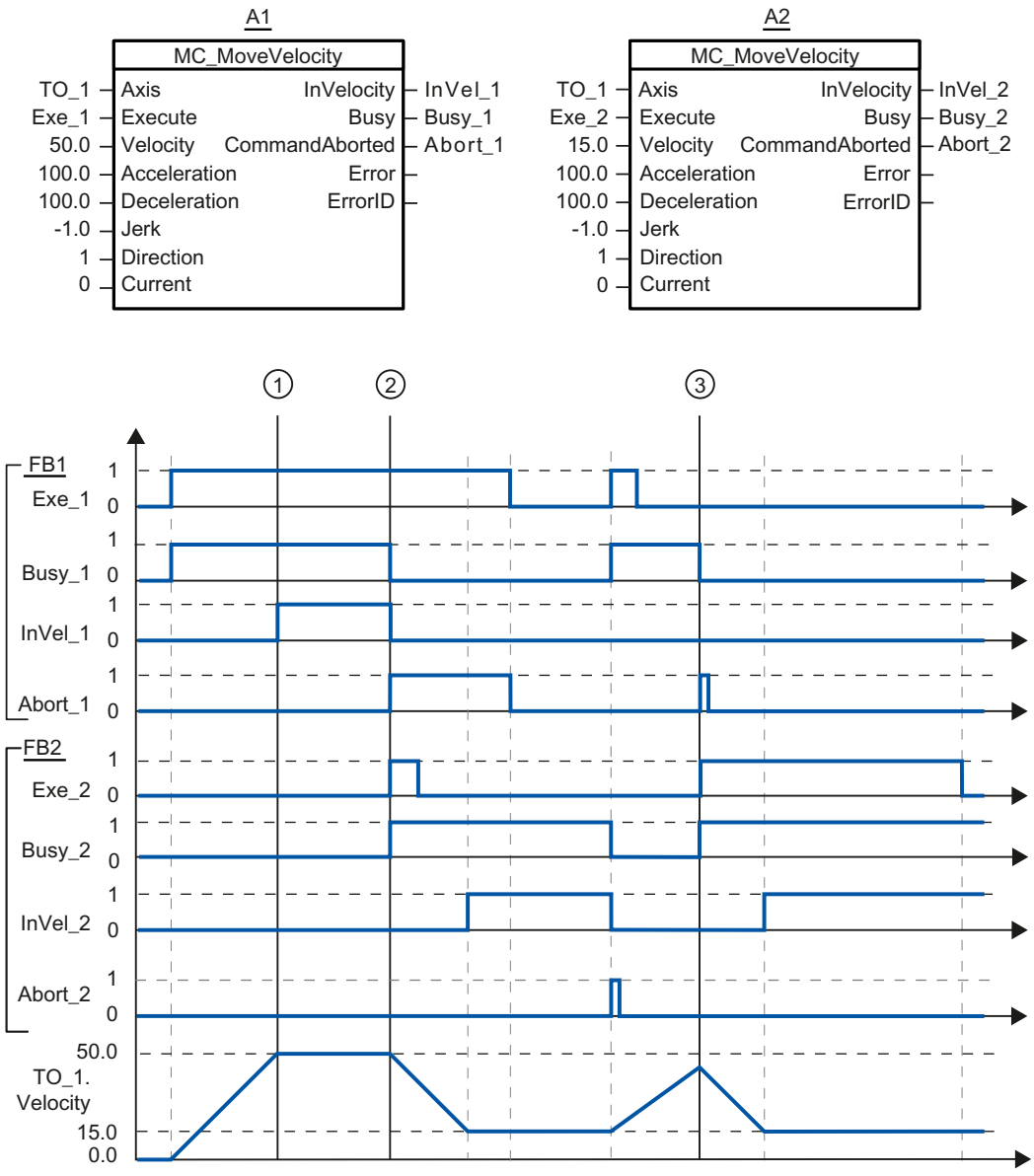

A "MC_MoveVelocity" job (A1) initiated via "Exe_1" accelerates the axis and signals at time ① via "InVel_1" that the setpoint velocity 50.0 has been reached.

At time ②, the task is overridden via another "MC_MoveVelocity" job (A2). The abort is signaled via "Abort_1". When the new setpoint velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the constant velocity 15.0.

The running "MC_MoveVelocity" job (A2) is overridden by another "MC_MoveVelocity" job (A1). The abort is signaled via "Abort_2". The axis is accelerated to the new setpoint velocity 50.0. Before the setpoint velocity is reached, the current "MC_MoveVelocity" job A1) is

overridden at time ③ by another "MC_MoveVelocity" job A2). The abort is signaled via "Abort_1". When the new setpoint velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the constant velocity 15.0.

MC_MoveJog V1 (S7-1500)

MC_MoveJog: Move axis in jog mode V1 (S7-1500)

Description

With the Motion Control instruction "MC_MoveJog", you move an axis in jog mode.

You define the dynamic behavior of the motion with parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

- Positioning axis:
  A velocity is specified in the "Velocity" parameter.
- Speed axis:
  A speed is specified in the "Velocity" parameter.

Applies to

- Positioning axis
- Speed axis

Requirements

- The technology object has been configured correctly.
- The technology object has been enabled.

Override response

The "MC_MoveJog" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" command "HomingMode" = 4, 5
- "MC_Halt" command
- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command

Starting an "MC_MoveJog" job aborts the following active Motion Control jobs:

- "MC_Home" command "HomingMode" = 4, 5
- "MC_Halt" command

- "MC_MoveAbsolute" command
- "MC_MoveRelative" command
- "MC_MoveVelocity" command
- "MC_MoveJog" command

Parameters

The following table shows the parameters of the Motion Control instruction "MC_MoveJog":

Parameters	Declaration	Data type	Default value	Description	
Axis	InOut	TO_SpeedAxis	-	Technology object	
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".	
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".	
Velocity	INPUT	LREAL	100.0	Velocity setpoint / speed setpoint for the motion Value > 0.0: The specified value is used. Value < 0.0: The absolute value of the specified value is used. ("Velocity" = 0.0 is permitted)	
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: Not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used. Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity setpoint / speed setpoint was reached and will be maintained.
Busy	OUTPUT	BOOL	FALSE	TRUE	Job is running.

Parameters	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The job was aborted by another job during execution.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "ErrorID"	

Behavior with velocity setpoint / speed setpoint zero ("Velocity" = 0.0)

An "MC_MoveJog" job with "Velocity" = 0.0 stops the axis with the configured deceleration. When the velocity setpoint / speed setpoint zero is reached, the parameter "InVelocity" will indicate the value TRUE.

Under "Technology object > Diagnostics > Status and error bits > Motion status", "constant velocity" and "standstill" will be displayed (<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill)).

Moving an axis in jog mode

Proceed as follows to move an axis in jog mode:

1. Check the requirements indicated above.
2. Move the axis in the positive direction with "JogForward", or in the negative direction with "JogBackward".

The current motion state is indicated in "Busy", "InVelocity" and "Error".

Note

Response to a change in the override

If the velocity / speed is influenced during constant motion by a change in the override (<TO>.Override.Velocity), the "InVelocity" parameter is reset during the acceleration or deceleration. When the newly calculated velocity is reached ("Velocity" × "Override" %), then "InVelocity" is set again.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord section.

MC_MoveJog: Function chart V1 (S7-1500)

Function chart: Moving an axis in jog mode

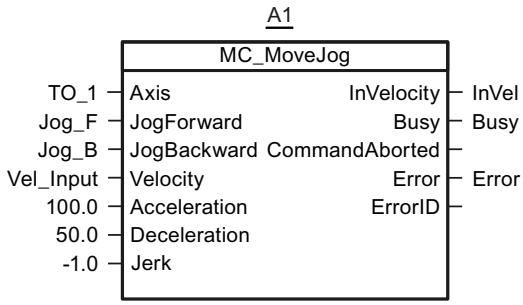

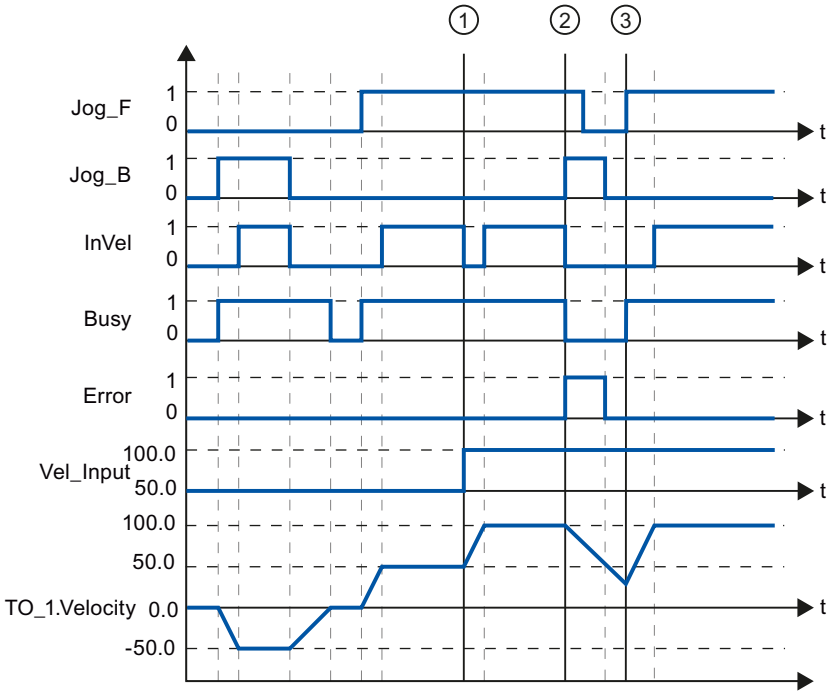

The axis is moved in the negative direction in jog mode via "Jog_B". When the setpoint velocity -50.0 is reached, this is signaled via "InVel" = TRUE. After "Jog_B" is reset, the axis is braked and brought to a standstill. Then the axis is moved in the positive direction via "Jog_F". When the setpoint velocity 50.0 is reached, this is signaled via "InVel" = TRUE.

At time ①, if "Jog_F" is set, the setpoint velocity is changed to 100.0 by means of "Vel_Input". Alternatively, you can also change the setpoint velocity via the velocity override. "InVel" is reset. The axis is being accelerated. When the new setpoint velocity 100.0 is reached, this is signaled via "InVel" = TRUE.

If "Jog_F" is set, "Jog_B" is likewise set at time ②. If both "Jog_F" and "Jog_B" are set, then the axis is braked with the last applicable deceleration. An error is indicated via "Error", and the "ErrorID" of the error 16#8007 (incorrect direction specification) is output.

This error is resolved by resetting the two inputs "Jog_F" and "Jog_B".

During the braking ramp, "Jog_F" is set at time ③. The axis is accelerated to the last configured velocity. When the setpoint velocity 100.0 is reached, this is signaled via "InVel" = TRUE.

4.1.5.5 Time-based IO (S7-1500)

Time-based IO (S7-1500)

TIO_SYNC: Synchronizing TIO modules (S7-1500)

Description

You can use Time-based IO with the TIO_SYNC instruction. TIO_SYNC synchronizes TIO modules to a shared time basis TIO_Time.

You can synchronize up to 8 TIO modules with TIO_SYNC. All TIO modules must be assigned to the same process image partition (PIP). If you select "0" for the input parameter PIP_Mode, you assign the number of the process image partition at the PIP_PART input parameter.

Startup characteristics

At the startup of the CPU, the TIO_SYNC instruction receives and checks the input parameters once and initializes the TIO_Time. If startup takes place without errors, the instruction changes to normal operation. In the event of an error, the instruction does not change to normal operation and generates an error message.

Operating principle

In normal operation, the instruction ensures synchronization of all TIO modules configured at the HWID input.

The calculated TIO_Time for the instructions of the TIO modules is provided at the TIO_SYNC_Data output.

Reaction to error

The Error output indicates whether the instruction parameters are assigned correctly and whether an error has occurred. If the instruction was not processed successfully, the reasons for the errors are displayed at the Status output.

Parameter

The table below shows the parameters of the TIO_SYNC instruction.

Parameter	Declaration	Data type S7-1500	Default	Description
HWID_1 ... HWID_8*	Input	HW_IO	-1	Hardware identifier for TIO module from HWCN
SendClock*	Input	LTime	LT#0ns	Send clock of the sync domain. Apply the send clock from the PROFINET configuration.
PIP_Mode*	Input	USInt	0	Mode for the data update**: • 0: OIP model with internal update of the process image by SYNC_PO and SYNC_PI. • 1: OIP model without internal update of the process image • 2: IPO model without internal update of the process image
PIP_PART*	Input	USInt	1	Only relevant if PIP_Mode = 0 Number of the process image partition which is to be updated isochronously.
TIO_SYNC_Data	Output	"TIO_SYNC_Data"		Calculated system time for the TIO instructions of the TIO modules and internal data used for synchronization. See UDT "TIO_SYNC_Data" (Page 3442).
Status	Output	DWord	16#0	Status of the instruction: see description of Status parameter
Error	Output	Bool	False	If True, this output indicates that an error has occurred. For detailed information see Status parameter. Error is reset as soon as the error is remedied.

* Checked once at startup of the CPU

** The IPO model (PIP_Mode = 2) provides the shortest response times, but it also places the highest demands on system performance. Processing of all TIO instructions and other program sections must be completed within one send clock.

Status parameter

Error codes or status information is output as double word at the Status output.

The double word is divided as follows:

Segment of the error code	Meaning
z0yywwww	Error of a system function: • during startup (z = 0) • during normal operation (z = 1) System functions with subordinate use are coded in yy: See table with error codes. wwww specifies the RET_VAL of the system function. The error information is available in the help for the system function.
z0yy0000	An error that does not originate in a system function. This error receives a consecutive error number yy. The error can occur: • during startup (z = 0) • during normal operation (z = 1)

Table with error codes

Error code (DW#16#...)	Meaning	Solution
00000000	No error.	—
1001xxxx	An error has occurred with system function RD_SINFO. The low word xxxx displays the error information of the RET_VAL return value from RD_SINFO.	• Read the description of the RD_SINFO in the TIA Portal information system. • Make sure that TIO_SYNC is called in a "Synchronous Cycle" OB.
10020000	The read cycle time of the isochronous OB is LT#0ms or has as negative value and is therefore invalid. The instruction can only be used without errors in an isochronous OB.	• Correct the value of the cycle time. • Make sure that TIO_SYNC is called in a "Synchronous Cycle" OB.
10030000	The TIO_SYNC instruction is not called in an isochronous OB. The instruction can only be used without errors in an isochronous OB. Possible cause: The TIO_SYNC instruction was called in an "MC-Servo" or "MC-Interpolator" OB.	Make sure that TIO_SYNC is called in a "Synchronous Cycle" OB.
10040000	The assigned value at the PIP_Mode input parameter is outside the valid range of 0 to 2.	Correct the value at the PIP_Mode input parameter.
10050000	The assigned send clock is outside the permitted range of 0 < SendClock <= 4 ms and is therefore invalid.	Correct the value of the send clock.
0006xxxx*	An error has occurred during execution of the SYNC_PI system function. The low word xxxx displays the error information of the RET_VAL return value from SYNC_PI.	Read the description of the SYNC_PI in the TIA Portal information system.
0007xxxx*	An error has occurred during execution of the SYNC_PO system function. The low word xxxx displays the error information of the return value RET_VAL from SYNC_PO.	Read the description of the SYNC_PO in the TIA Portal information system.
10080000	Isochronous mode missing. Time-based IO cannot be used. Possible cause: Internal error during address calculation	Check the values of input parameters HWID_1 to HWID_8. Specify the hardware identifier of the TIO module from its module properties in the hardware configuration.

Error code (DW#16#...)	Meaning	Solution
10090000	HWID_1 is not a valid hardware identifier. Possible cause: No module with this hardware identifier available.	Check the value of the respective input parameter. In each case, specify the hardware identifier of the TIO module from its module properties in the hardware configuration. An internal system constant of data type Hw_SubModule is available for the symbolic addressing.
100A0000	HWID_2 is not a valid hardware identifier. Possible cause: No module with this hardware identifier available.	
100B0000	HWID_3 is not a valid hardware identifier. Possible cause: No module with this hardware identifier available.	
100C0000	HWID_4 is not a valid hardware identifier. Possible cause: No module with this hardware identifier available.	
100D0000	HWID_5 is not a valid hardware identifier. Possible cause: No module with this hardware identifier available.	
100E0000	HWID_6 is not a valid hardware identifier. Possible cause: No module with this hardware identifier available.	
100F0000	HWID_7 is not a valid hardware identifier. Possible cause: No module with this hardware identifier available.	
10100000	HWID_8 is not a valid hardware identifier. Possible cause: No module with this hardware identifier available.	
10110000	The configured process image partition at input parameter PIP_PART is not within the valid range of 1 to 31.	Correct the value at the PIP_PART input parameter.
x0FF0000	General internal error.	—

* Available only when "0" is selected for the input parameter PIP_Mode "0".

TIO_IOLink_IN: Read process input signals with time stamps (S7-1500)

Description

You can use the Time-based IO with the TIO_IOLink_IN instruction. TIO_IOLink_IN detects an event at the IO-Link Device and returns the process value including the associated time stamp.

The IO-Link Device must be equipped with the time stamp function and the Port must be in "IO-Link, Time based IN" mode.

Startup characteristics

During startup of the CPU, the instruction TIO_IOLink_IN applies the input parameters once and checks the following:

- Checking HWID
- Checking whether the Port number is in the range (1 to 4)
- Checking TIO_SYNC_Data.Error: Is an error present at TIO_SYNC?
- Checking T_o for a positive value
- Checking the setting to type IO-Link
- Checking Port mode regarding configuration for IO-Link Time based IN
- Checking whether the OB is a "Synchronous Cycle" OB
- Checking PortQualifier

If startup takes place without errors, the instruction changes to normal operation. In the event of an error, the instruction does not change to normal operation and generates an error message.

Operating principle

In normal operation, the instruction detects the process data (SA_Data) of an IO-Link Device and the associated time stamp (time = TIO_Time) of the last valid change. Each valid change of the SA_Data with a correctly functioning port is accompanied by a valid time stamp.

You connect the input TIO_SYNC_Data with the output of the same name of the TIO_SYNC instruction. This ensures a shared time basis.

Reaction to error

The Error output indicates if the instruction was processed correctly. The reasons for the errors are displayed at the Status output.

Parameter

The table below shows the parameters of the TIO_IOLink_IN instruction.

Parameter	Declaration	Data type S7-1500	Default	Description
HWID*	Input	HW_IO	0	Hardware identifier for TIO module from HWCN
Port*	Input	USInt	0	Port number (1 to 4) of the connected IO-Link Device
TIO_SYNC_Data*	Input	"TIO_SYNC_Data"		System time provided for the TIO instructions of the TIO modules by the TIO_SYNC instruction. See UDT "TIO_SYNC_Data" (Page 3442). Connect this input parameter with the "TIO_SYNC_Data" output parameter of the TIO_SYNC instruction.
TO*	Input	LTime	LT#0ns	T_o: Time for output of isochronous output data. Apply T_o from the ET 200 station (properties of the PROFINET interface).

Parameter	Declaration	Data type S7-1500	Default	Description
SA_Bit0	Output	Bool	False	Display of a process data change (bit 0 of SA_Data). SA_Bit0 is event-oriented. As long as no valid time stamp has been acquired (EventCount = 0), the process data are invalid.
SA_Bit1	Output	Bool	False	Display of a process data change (bit 1 of SA_Data). SA_Bit1 is event-oriented. As long as no valid time stamp has been acquired (EventCount = 0), the process data are invalid.
SA_Data	Output	DWord	16#0	Process data SA_Data (Sensor Application Data)
TimeStamp	Output	LTime	LT#0ns	Time stamp: Time when a change of the process signal SA-Data took place at the IO-Link Device.
EventCount	Output	UInt	0	Counter: Is incremented with each new, valid time stamp. The counter is reset with each CPU startup.
Status	Output	DWord	16#0	Status of the instruction: see description of Status parameter
Error	Output	Bool	False	If True, this output indicates that an error has occurred. For detailed information see Status parameter. Error is reset as soon as the error is remedied.

* Checked once at startup of the CPU

Status parameter

Error codes or status information is output as double word at the Status output.

The double word is divided as follows:

Segment of the error code	Meaning
z0yywwww	Error of a system function: • during startup (z = 0) • during normal operation (z = 1) System functions with subordinate use are coded in yy: See table with error codes. wwww specifies the RET_VAL of the system function. The system function is available in the help for the system function.
z0yy0000	An error that does not originate in a system function. This error receives a consecutive error number yy. The error can occur: • during startup (z = 0) • during normal operation (z = 1)

Table with error codes

Error code (DW#16#...)	Meaning	Remedy
00000000	No error.	—
10010000	The assigned number at the Port input parameter is outside the valid range of 1 to 4.	Correct the value at the Port input parameter.
10020000	The TIO_IOLink_IN instruction is not called in an isochronous OB. The instruction can only be used without errors in an isochronous OB.	Make sure that TIO_IOLink_IN is called in a "Synchronous Cycle" OB.

Error code (DW#16#...)	Meaning	Remedy
10030000	An error occurred when reading the HWID input parameter.	Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in the hardware configuration. An internal system constant of data type Hw_SubModule is available for the symbolic addressing.
x0040000	The data in TIO_SYNC_Data is invalid/incorrect.	Check the TIO_SYNC instruction and the interconnection of its TIO_SYNC_Data output.
1005xxxx	An error has occurred during execution of the RD_SINFO system function. The low word xxxx displays the error information of the RET_VAL return value from RD_SINFO.	• Read the description of the RD_SINFO in the TIA Portal information system. • Make sure that TIO_IOLink_IN is called in a "Synchronous Cycle" OB.
10060000	Unable to find IO-Link Device. Possible cause: The module configured using the hardware identifier is not an IO-Link Master for Time-based IO.	• Make sure that the configured module is an IO-Link Master for Time-based IO. • Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in the hardware configuration.
10070000	An internal error occurred during the address calculation.	Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in the hardware configuration. An internal system constant of data type Hw_SubModule is available for the symbolic addressing.
00080000	TIO module is not synchronized using the TIO_SYNC instruction.	Check the instruction TIO_SYNC.
10090000	The read cycle time of the isochronous OB is outside the permitted range of $0 < T_{APP} <= 16$ ms and is therefore invalid. The instruction can only be used without errors in an isochronous OB.	• Correct the cycle time. • Make sure that TIO_IOLink_IN is called in a "Synchronous Cycle" OB.
100A0000	The assigned time at the TO input parameter is outside the permitted range of $0 < T_o <= 4$ ms.	Correct the value at the TO input parameter.
100Bxxxx	An error has occurred during execution of the RD_ADDR system function. The low word xxxx displays the error information of the RET_VAL return value from RD_ADDR.	Read the description of the RD_ADDR in the TIA Portal information system.
000C0000	The converted time stamp is invalid.	Check the connected sensor and the interaction between IO-Link Master and sensor (e.g., configuration).
000D0000	The value status PortQualifier of the IO-Link indicates that the process data is invalid.	Check the connected sensor and its configuration.
100E0000	The configured port mode of the IO-Link is incorrect.	Check the configuration of the connected sensor with S7-PCT.
x0FF0000	General internal error.	—

TIO_DI: Reading in edges at digital input and associated time stamps (S7-1500)

Description

You can use the Time-based IO with the TIO_DI instruction. TIO_DI continuously detects the edges at a digital input of a TM Timer DIDQ and returns the associated time stamps.

Startup characteristics

During startup of the CPU, the instruction TIO_DI applies the input parameters once and checks the following:

- Checking HWID
- Checking to see if the number of the digital input (Channel) is in the permitted range (depending on addressed module and channel configuration)
- Checking TIO_SYNC_Data.Error: Is an error present at TIO_SYNC?
- Checking T_o for validity (0 ms to 4 ms)
- Checking whether the OB is a "Synchronous Cycle" OB

If startup takes place without errors, the instruction changes to normal operation. The input parameter EdgeSel can be changed during normal operation. In the event of an error, the instruction does not change to normal operation and generates an error message.

Operating principle

In normal operation, the instruction detects the edges at a digital input and the associated time stamps (time = TIO_Time) of the last valid, defined edge pair. Use the input parameter EdgeSel to determine the edges for which time stamps are detected.

You connect the input TIO_SYNC_Data with the output of the same name of the TIO_SYNC instruction. This ensures a shared time basis.

Reaction to error

The Error output indicates if the instruction was processed correctly. The reasons for the errors are displayed at the Status output.

Parameter

The table below shows the parameters of the TIO_DI instruction.

Parameter	Declaration	Data type	Default	Description
		S7-1500		
HWID*	Input	HW_IO	0	Hardware identifier for TIO module from HWCN
Channel*	Input	UInt	0	Number (0 ... m) of the digital input of the connected TM Timer DIDQ

Parameter	Declaration	Data type S7-1500	Default	Description
TIO_SYNC_Data*	Input	"TIO_SYNC_Data"		System time provided for the TIO instructions of the TIO modules by the TIO_SYNC instruction. See UDT "TIO_SYNC_Data" (Page 3442). Connect this input parameter with the "TIO_SYNC_Data" output parameter of the TIO_SYNC instruction.
EdgeSel	Input	UInt	3	Specify the edges for which time stamps are detected: 0_D: Rearm / repetition time stamp detection 1_D: Two rising edges 2_D: Two falling edges 3_D: Rising and falling edge (order depending on occurrence) 4_D: First rising, then falling edge 5_D: First falling, then rising edge 6 to 255_D: Invalid
TO*	Input	LTime	LT#0ns	T_o: Time for output of isochronous output data. Apply T_o from the ET 200 station (properties of the PROFINET interface).
DI	Output	Bool	False	Status of digital input. If an inversion of the digital input is configured, this parameter is inverted as well.
Time-StampRE	Output	LTime	LT#0ns	Time stamp: Time at which a positive edge was detected. Exception: EdgeSel = 2_D: Time at which a falling edge has been detected (if *multiple* falling edges have occurred during the application cycle).
Time-StampFE	Output	LTime	LT#0ns	Time stamp: Time at which a negative edge was detected. Exception: EdgeSel = 1_D: Time at which a rising edge has been detected (if *multiple* rising edges have occurred during the application cycle).
EventCoun-tRE	Output	UInt	0	Counter: Is incremented with each new, valid time stamp at a positive edge. The counter is reset with each CPU startup.
Even-tCountFE	Output	UInt	0	Counter: Is incremented with each new, valid time stamp at a negative edge. The counter is reset with each CPU startup.
LEC	Output	UInt	0	Counter: Number of edges for which no time stamp could be saved. The module can count a maximum of seven edges per application cycle. The counter is reset with each new application cycle.
Status	Output	DWord	16#0	Status of the instruction: see description of Status parameter
Error	Output	Bool	False	If True, this output indicates that an error has occurred. For detailed information see Status parameter. Error is reset as soon as the error is remedied.

* Checked once at startup of the CPU

Status parameter

Error codes or status information is output as double word at the Status output.

The double word is divided as follows:

Segment of the error code	Meaning
z0yywwww	Error of a system function: • during startup (z = 0) • during normal operation (z = 1) System functions with subordinate use are coded in yy: See table with error codes. wwww specifies the RET_VAL of the system function. The system function is available in the help for the system function.
z0yy0000	An error that does not originate in a system function. This error receives a consecutive error number yy. The error can occur: • during startup (z = 0) • during normal operation (z = 1)

Table with error codes

Error code (DW#16#...)	Meaning	Solution
00000000	No error.	—
10010000	The assigned number of the digital input at the Channel input parameter is outside the permitted range (depending on addressed module and channel configuration).	Correct the value at the Channel input parameter.
10020000	The TIO_DI instruction is not called in an isochronous OB. The instruction can only be used without errors in an isochronous OB.	Make sure that TIO_DI is called in a "Synchronous Cycle" OB.
10030000	An error occurred when reading the HWID input parameter.	Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in the hardware configuration. An internal system constant of data type Hw_SubModule is available for the symbolic addressing.
x0040000	The data in TIO_SYNC_Data is invalid/incorrect.	Check the TIO_SYNC instruction and the interconnection of its TIO_SYNC_Data output.
1005xxxx	An error has occurred during execution of the RD_SINFO system function. The low word xxxx displays the error information of the RET_VAL return value from RD_SINFO.	• Read the description of the RD_SINFO in the TIA Portal information system. • Make sure that TIO_DI is called in a "Synchronous Cycle" OB.
10060000	Unable to find TM Timer DIDQ. Possible cause: The module configured using the hardware identifier is not a TM Timer DIDQ.	• Make sure that the configured module is a TM Timer DIDQ. • Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in the hardware configuration.

Error code (DW#16#...)	Meaning	Solution
10070000	An internal error occurred during the address calculation.	Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in the hardware configuration. An internal system constant of data type Hw_SubModule is available for the symbolic addressing.
00080000	TIO module is not synchronized using the TIO_SYNC instruction.	Check the instruction TIO_SYNC.
10090000	The read cycle time of the isochronous OB is outside the permitted range of 0 < T_{APP} <= 16 ms and is therefore invalid. The instruction can only be used without errors in an isochronous OB.	• Correct the cycle time. • Make sure that TIO_DI is called in a "Synchronous Cycle" OB.
100A0000	The assigned time at the TO input parameter is outside the permitted range of 0 < T_o <= 4 ms.	Correct the value at the TO input parameter.
100Bxxxx	An error has occurred during execution of the RD_ADDR system function. The low word xxxx displays the error information of the RET_VAL return value from RD_ADDR.	Read the description of the RD_ADDR in the TIA Portal information system.
000C0000	The converted time stamp is invalid. Possible cause: Communication error	Check the PROFINET communication with the TIO module.
000D0000	The Quality Information of the digital input indicates that an error has occurred at the digital input.	Check the wiring of the digital input.
000E0000	The assigned number at the Channel input parameter is not a digital input configured as Timer DI.	• Check the channel configuration (only for TM Timer DIDQ 16x24V). • Check the operating mode of the digital input.
000F0000	The assigned value at the EdgeSel input parameter is outside the valid range of 1 to 5.	Correct the value at the EdgeSel input parameter.
10100000	The send clock is outside the permitted range of 0 < SendClock <= 4 ms and is therefore invalid. The error code can also indicate that • The data in TIO_SYNC_Data is invalid or does not exist. • The TIO_DI instruction is not called in an isochronous OB.	Correct the send clock.
x0FF0000	General internal error.	—

TIO_IOLink_OUT: Output time-controlled process output signals (S7-1500)

Description

You can use Time-based IO with the TIO_IOLink_OUT instruction. TIO_IOLink_OUT lets you activate the output data of an IO-Link Device at a specified time.

The IO-Link Device must be equipped with the time stamp function and the Port must be in "IO-Link, Time based OUT" mode.

Startup characteristics

During startup of the CPU, the instruction TIO_IOLink_OUT applies the input parameters once and checks the following:

- Checking HWID
- Checking whether the Port number is in the range (1 to 4)
- Checking TIO_SYNC_Data.Error: Is an error present at TIO_SYNC?
- Checking T_o for a positive value
- Checking the setting to type IO-Link
- Checking Port mode regarding configuration for IO-Link Time based OUT
- Checking whether the OB is a "Synchronous Cycle" OB
- Checking PortQualifier

If startup takes place without errors, the instruction changes to normal operation. The input parameters REQ , Out_Mode, TimeStamp and AA_Data can be changed during normal operation. In the event of an error, the instruction does not change to normal operation and generates an error message.

Operating principle

In normal operation, the instruction sends the process data (AA_Data) to an IO-Link Device. The output data AA_Data are activated at the time defined at the input parameter TimeStamp.

You connect the input TIO_SYNC_Data with the output of the same name of the TIO_SYNC instruction. This ensures a shared time basis.

You start an output job with a positive edge at the "REQ" parameter. You can only start a new job when there is no error pending and no job is active. When starting the output job, the AA_Data (bit 0, 1) are activated on the IO-Link Device at a time defined by TimeStamp. The job is done when the last application cycle is executed before the output time is reached (Done). Status and Error are constantly being updated during the job runtime.

Note

Once the job has been started with a positive edge, you can adapt the output time with the input of a new TimeStamp without having to restart the job.

Constraint:

If the adapted time stamp is less than 16 ms before the output time (TimeStamp - TIO_Time < 16), the last valid time stamp is used.

If you specify the value 0 as TimeStamp, the output is written directly with the data specified at the input AA_Data. This gives you the option to implement a direct control from the TIO module without time stamp in manual mode. You can use the direct control to interrupt an ongoing job.

Reaction to error

The Error output indicates if the instruction was processed correctly. The reasons for the errors are displayed at the Status output.

Parameter

The table below shows the parameters of the TIO_IOLink_OUT instruction.

Parameter	Declaration	Data type S7-1500	Default	Description
REQ	Input	Bool	False	Starts the job at a positive edge.
HWID*	Input	HW_IO	0	Hardware identifier for TIO module from HWCN.
TO*	Input	LTime	LT#0ns	T_o: Time for output of isochronous output data. Apply T_o from the ET 200 station (properties of the PROFINET interface).
Port*	Input	USInt	0	Port number (1 to 4) of the specified IO-Link Device
TIO_SYNC_Data*	Input	"TIO_SYNC_Data"		System time provided for the TIO instructions of the TIO modules by the TIO_SYNC instruction. See UDT "TIO_SYNC_Data" (Page 3442). Connect this input parameter with the "TIO_SYNC_Data" output parameter of the TIO_SYNC instruction.
TimeStamp	Input	LTime	LT#0ns	Time stamp: Time when the process data (AAE1, AAE2) are to be output.
AA_Data	Input	Word	16#0	Process output data: Date to be output (Word). Includes AAE1 and AAE2 on bit 0, 1.
Busy	Output	Bool	False	Busy = True: The job has not yet been completed.
Done	Output	Bool	False	Done = True is displayed for one cycle: The job is signaled as "done, no errors".
Error	Output	Bool	False	If True, this output indicates that an error has occurred. In this case, BUSY and DONE are set to False. For detailed information see Status parameter. Error is reset as soon as the error is remedied.
Status	Output	DWord	16#0	Status of the instruction: see description of Status parameter

* Checked once at startup of the CPU

Status parameter

Error codes or status information is output as double word at the Status output.

The double word is divided as follows:

Segment of the error code	Meaning
z0yywwww	Error of a system function: • during startup (z = 0) • during normal operation (z = 1) System functions with subordinate use are coded in yy: See table with error codes. wwww specifies the RET_VAL of the system function. The error information is available in the help for the system function.
z0yy0000	An error that does not originate in a system function. This error receives a consecutive error number yy. The error can occur: • during startup (z = 0) • during normal operation (z = 1)

Table with error codes

Error code (DW#16#...)	Meaning	Remedy
00000000	No error.	—
10010000	The assigned number at the Port input parameter is outside the valid range of 1 to 4.	Correct the value at the Port input parameter.
10020000	The TIO_IOLink_OUT instruction is not called in an isochronous OB. The instruction can only be used without errors in an isochronous OB.	Make sure that TIO_IOLink_OUT is called in a "Synchronous Cycle" OB.
10030000	An error occurred when reading the HWID input parameter.	Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in the hardware configuration. An internal system constant of data type Hw_SubModule is available for the symbolic addressing.
x0040000	The data in TIO_SYNC_Data is invalid/incorrect.	Check the TIO_SYNC instruction and the interconnection of its TIO_SYNC_Data output.
1005xxxx	An error has occurred during execution of the RD_SINFO system function. The low word xxxx displays the error information of the RET_VAL return value from RD_SINFO.	• Read the description of the RD_SINFO in the TIA Portal information system. • Make sure that TIO_IOLink_OUT is called in a "Synchronous Cycle" OB.
10060000	Unable to find IO-Link Device. Possible cause: The module configured using the hardware identifier is not an IO-Link Master for Time-based IO.	• Make sure that the configured module is an IO-Link Master for Time-based IO. • Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in the hardware configuration.
10070000	An internal error occurred during the address calculation.	Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in the hardware configuration. An internal system constant of data type Hw_SubModule is available for the symbolic addressing.

Error code (DW#16#...)	Meaning	Remedy
00080000	TIO module is not synchronized using the TIO_SYNC instruction.	Check the instruction TIO_SYNC.
10090000	The read cycle time of the isochronous OB is outside the permitted range of 0 < T_{APP} <= 16 ms and is therefore invalid. The instruction can only be used without errors in an isochronous OB.	• Correct the cycle time. • Make sure that TIO_IOLink_OUT is called in a "Synchronous Cycle" OB.
100A0000	The assigned time at the TO input parameter is outside the permitted range of 0 < T_o <= 4 ms.	Correct the value at the TO input parameter.
100Bxxxx	An error has occurred during execution of the RD_ADDR system function. The low word xxxx displays the error information of the RET_VAL return value from RD_ADDR.	Read the description of the RD_ADDR in the TIA Portal information system.
000C0000	The time stamp at the TimeStamp input parameter is invalid.	Check the TimeStamp input parameter.
000D0000	The value status PortQualifier of the IO-Link indicates that the process data is invalid.	Check the connected sensor and its configuration.
100E0000	The configured port mode of the IO-Link is incorrect.	Check the configuration of the connected sensor with S7-PCT.
100F0000	The read cycle time of the OB of type "Synchronous Cycle" is too long: T_{APP} > 16 ms.	Configure a smaller multiple of the send clock as the cycle time.
x0FF0000	General internal error.	—

TIO_DQ: Output edges time-controlled at the digital output (S7-1500)

Description

You can use Time-based IO with the TIO_DQ instruction. TIO_DQ switches the digital output of a TM Timer DIDQ at a specified time.

Startup characteristics

During startup of the CPU, the instruction TIO_DQ applies the input parameters once and checks the following:

- Checking HWID
- Checking to see if the number of the digital input (Channel) is in the permitted range (depending on addressed module and channel configuration)
- Checking TIO_SYNC_Data.Error: Is an error present at TIO_SYNC?
- Checking T_o for validity (0 ms to 4 ms)
- Checking whether the OB is a "Synchronous Cycle" OB

If startup takes place without errors, the instruction changes to normal operation. The input parameters REQ , Out_Mode, TimeStampRE and TimeStampFE can be changed during normal operation. In the event of an error, the instruction does not change to normal operation and generates an error message.

Operating principle

The instruction outputs edges time-controlled at a digital output during normal operation.

- At the time defined at the TimeStampRE input parameter, a positive edge is output at the digital output.
- At the time defined at the TimeStampFE input parameter, a negative edge is output at the digital output.

Depending on the program execution model, a time stamp must exceed the following value:

Program execution model	TimeStampRE> ... TimeStampFE > ...
IPO model	TIO_Time + T_{APP} + T_O
OIP model	TIO_Time + T_{APP} + SendClock + T_O

Use the input parameter Out_Mode to determine if only one edge or both edges are output.

You connect the input TIO_SYNC_Data with the output of the same name of the TIO_SYNC instruction. This ensures a shared time basis.

You start an output job with a positive edge at the "REQ" parameter. You can only start a new job when there is no error pending and no job is active. When the output job is started, the digital output is switched at the times defined with TimeStampRE and TimeStampFE.

- If the digital output is already set at time TimeStampRE, the output job is not transferred to the module for the positive edge.
- If the digital output is not set at time TimeStampFE, the output job is not transferred to the module for the negative edge.

This means the digital output is not switched in both cases.

The job is done when the last application cycle is executed before the second output time is reached (DONE). Status and Error are constantly being updated during the job runtime. You can cancel an active output job by adapting TimeStampRE or TimeStampFE with an invalid time stamp.

The figure below shows an example for the reaction of the bits DONE and BUSY at the start of an output job under the following conditions:

- Out_Mode = 2
- The two time stamps are not adapted between the start of the job and the output.

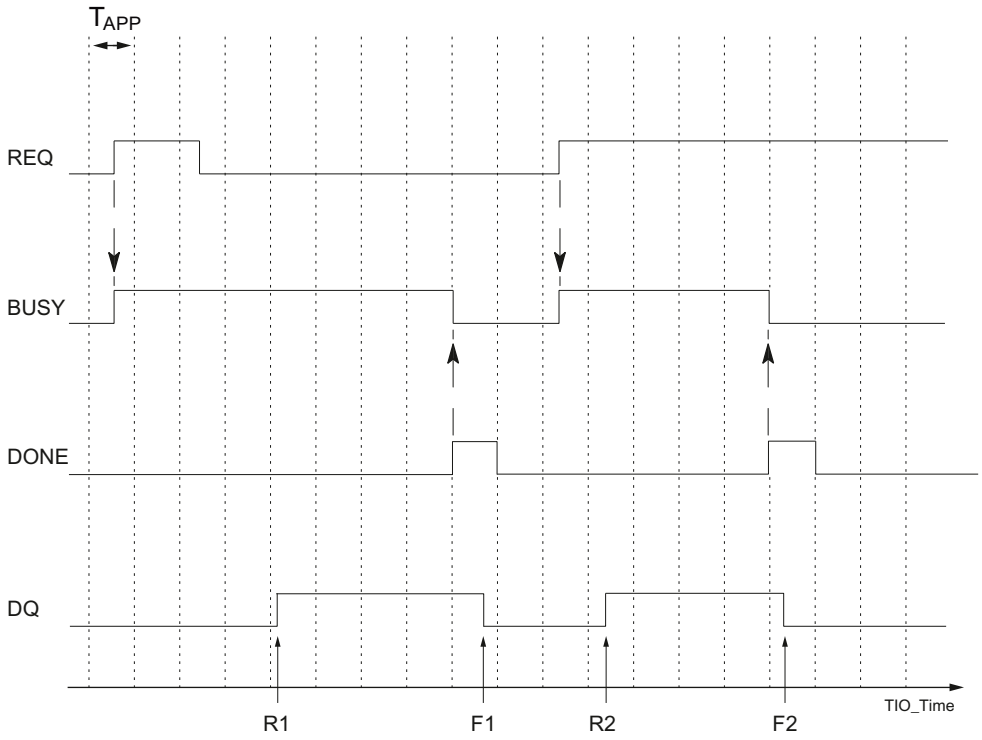

T_{APP}	Application cycle
R1, R2	Specified times of a positive DQ edge
F1, F2	Specified times of a negative DQ edge

Note

Once the job has been started with a positive edge, you can adapt the output times with the input of TimeStampRE and TimeStampFE without having to restart the job.

Constraint:

If an adapted time stamp is less than two application cycles away from the output time (TimeStampRE - TIO_Time < $2*T_{APP}$ or TimeStampFE - TIO_Time < $2*T_{APP}$), the last valid time stamp is used.

If you specify the value "0" as TimeStampRE or TimeStampFE, you can output the respective edge directly at the digital output with the input parameter Out_Mode = 3. This gives you the option to implement a direct control from the TIO module without time stamp in manual mode. You can use the direct control to interrupt an ongoing job.

Changes to the input parameters are not applied until the TIO_DQ instruction detects a startup of the CPU.

Reaction to error

The Error output indicates if the instruction was processed correctly. The reasons for the errors are displayed at the Status output.

Parameter

The table below shows the parameters of the TIO_DQ instruction.

Parameter	Declaration	Data type S7-1500	Default	Description
REQ	Input	Bool	False	Starts the job at a positive edge.
HWID*	Input	HW_IO	0	Hardware identifier for TIO module from HWCN.
Channel*	Input	UInt	0	Number (0 ... m) of the digital output of the connected TM Timer DIDQ
TIO_SYNC_Data*	Input	"TIO_SYNC_Data"		System time provided for the TIO instructions of the TIO modules by the TIO_SYNC instruction. See UDT "TIO_SYNC_Data" (Page 3442). Connect this input parameter with the "TIO_SYNC_Data" output parameter of the TIO_SYNC instruction.
Out_Mode	Input	UInt	2	Specify the output mode for the edges at the digital output: 0_D: Only rising edge is output (TimeStampRE). 1_D: Only falling edge is output (TimeStampFE). 2_D: Both edges are output (TimeStampRE and TimeStampFE). 3_D: Each edge is output directly if TimeStampRE = 0 or TimeStampFE = 0 If both time stamps have the value "0" or if no time stamp has the value "0", no edge is output. 4 to 255_D: Invalid
TO*	Input	LTime	LT#0ns	T_o: Time for output of isochronous output data. Apply T_o from the ET 200 station (properties of the PROFINET interface).
TimeStampRE	Input	LTime	LT#0ns	Time stamp: Time when a positive edge is to be output.
TimeStampFE	Input	LTime	LT#0ns	Time stamp: Time when a negative edge is to be output.
StatusDQ	Output	Bool	False	Actual status of the digital output. If an inversion of the digital output is configured, StatusDQ is inverted as well. TIO_DQ cannot check the parameter assignment of the specific digital output and its actual state. StatusDQ can return an incorrect value if a HW enable is configured, and the digital output is not enabled during an output job.
BUSY	Output	Bool	False	BUSY = True: The job has not yet been completed.
DONE	Output	Bool	False	DONE = True is displayed for one cycle: The job is signaled as "done, no errors".
Status	Output	DWord	16#0	Status of the instruction: see description of Status parameter
Error	Output	Bool	False	If True, this output indicates that an error has occurred. In this case, BUSY and DONE are set to False. For detailed information see Status parameter. Error is reset as soon as the error is remedied.

* Checked once at startup of the CPU

Status parameter

Error codes or status information is output as double word at the Status output.

The double word is divided as follows:

Segment of the error code	Meaning
z0yywwww	Error of a system function: • during startup (z = 0) • during normal operation (z = 1) System functions with subordinate use are coded in yy: See table with error codes. wwww specifies the RET_VAL of the system function. The error information is available in the help for the system function.
z0yy0000	An error that does not originate in a system function. This error receives a consecutive error number yy. The error can occur: • during startup (z = 0) • during normal operation (z = 1)

Table with error codes

Error code (DW#16#...)	Meaning	Solution
00000000	No error.	—
10010000	The assigned number of the digital output at the Channel input parameter is outside the permitted range (depending on addressed module and channel configuration).	Correct the value at the Channel input parameter.
10020000	The TIO_DQ instruction is not called in an isochronous OB. The instruction can only be used without errors in an isochronous OB.	Make sure that TIO_DQ is called in a "Synchronous Cycle" OB.
10030000	An error occurred when reading the HWID input parameter.	Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in HWCN. An internal system constant of data type Hw_SubModule is available for the symbolic addressing.
x0040000	The data in TIO_SYNC_Data is invalid/incorrect.	Check the TIO_SYNC instruction and the interconnection of its TIO_SYNC_Data output.
1005xxxx	An error has occurred during execution of the RD_SINFO system function. The low word xxxx displays the error information of the RET_VAL return value from RD_SINFO.	• Read the description of the RD_SINFO in the TIA Portal information system. • Make sure that TIO_SYNC is called in a "Synchronous Cycle" OB.
10060000	Unable to find TM Timer DIDQ. Possible cause: The module configured using the hardware identifier is not a TM Timer DIDQ.	• Make sure that the configured module is a TM Timer DIDQ. • Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in HWCN.

Error code (DW#16#...)	Meaning	Solution
10070000	An internal error occurred during the address calculation.	Check the value at the HWID input parameter. Specify the hardware identifier of the TIO module from its module properties in HWCN. An internal system constant of data type Hw_SubModule is available for the symbolic addressing.
x0080000	TIO module is not synchronized using the TIO_SYNC instruction. The error code can also indicate that: • A job was already present before the first run of the instruction. • The assigned number at the Channel input parameter is not a digital output.	Check the instruction TIO_SYNC.
10090000	The read cycle time of the isochronous OB is LT#0ms or has as negative value and is therefore invalid. Correct the value. The instruction can only be used without errors in an isochronous OB.	• Correct the cycle time. • Make sure that TIO_DQ is called in a "Synchronous Cycle" OB.
100A0000	The assigned time at the TO input parameter is outside the permitted range of 0 < T_o <= 4 ms.	Correct the value at the TO input parameter.
100Bxxxx	An error has occurred during execution of the RD_ADDR system function. The low word xxxx displays the error information of the RET_VAL return value from RD_ADDR.	Read the description of the RD_ADDR in the TIA Portal information system.
000C0000	One or both time stamps at the TimeStampRE and TimeStampFE input parameters are invalid. The error is only signaled for the duration of one application cycle.	Check the TimeStampRE and TimeStampFE input parameters.
000D0000	The Quality Information of the digital output indicates that an error has occurred at the digital output.	• Check the parameter assignment of the digital output. • Check the wiring of the digital output for short-circuit, overload, and overtemperature.
000E0000	The number configured at the Channel input parameter is not a digital output configured as Timer DQ.	• Check the channel configuration (only for TM Timer DIDQ 16x24V). • Check the operating mode of the digital output.
100F0000	The read cycle time of the OB of type "Synchronous Cycle" is too long: T_{APP} > 16 ms.	Configure a smaller multiple of the send clock as the cycle time.
10100000	The send clock is outside the permitted range of 0 < SEND_CLOCK <= 4 ms and is therefore invalid. The error code can also indicate that • The data in TIO_SYNC_Data is invalid or does not exist. • The TIO_DQ instruction is not called in an isochronous OB.	Correct the send clock.
00110000	The assigned value at the Out_Mode input parameter is outside the valid range of 0 to 3.	Correct the value at the Out_Mode input parameter.
x0FF0000	General internal error.	—

UDT "TIO_SYNC_Data" (S7-1500)

Description

The data type UDT "TIO_SYNC_Data" contains the central structure and data for synchronization of the modules and passing of the TIO_Time.

Parameter

Parameter	Data type	Description
	S7-1500	
TIO_TIME_BASE	LTime	Internal use
TIO_TIME	LTime	Shared time basis (relative time) of the TIO modules.
PIP_MODE	USInt	Mode for data update (is forwarded by the PIP_Mode input parameter of the TIO_SYNC instruction)
APP_CYC	LTime	Application cycle of the "Synchronous Cycle" OB
SEND_CLOCK	LTime	Send clock of sync domain (is forwarded by the Send-Clock input parameter of the TIO_SYNC instruction)
TBase	LTime	Internal use
ERROR	Bool	Internal use

4.1.6 Communication

4.1.6.1 S7 communication

Data consistency

Definition

A data block which cannot be modified by concurrent processes is called consistent data area. A data block which is larger than the consistent data area can thus be falsified as a whole during transmission. This means that a data block which belongs together and which is larger than consistent data area can consist in part of new and of old consistent data at the same time.

Example

An inconsistency can occur when an instruction for communication is interrupted, for example, by a hardware interrupt OB with higher priority. If the user program in this OB now changes the data that has already been partly processed by the instruction, the transferred data originates:

- In part from the time before the hardware interrupt was processed
- And in part from the time after the hardware interrupt was processed

This means that these data are inconsistent (not coherent).

Ensuring data consistency

If the communication process can be interrupted by an interrupt OB, you must ensure that the data is transferred consistently. Make sure that only an image of the data and not the data to be transferred is not changed directly by the interrupt OB. Copy the image of the data to the transfer area of the communication instruction prior to the next data transfer.

- If the user program contains a communication instruction that accesses common data, access to this data area can be coordinated, for example, by means of the DONE parameter. The data consistency of the communication areas which are transferred locally with a communication instruction can therefore be ensured in the user program.
- In the case of S7 communication instructions "PUT (Page 3452)"/"GET (Page 3447)", the size of the consistent data areas must already be taken into consideration during programming or configuration, because a communication block is not available in the user program of the target device (server) to synchronize communication data to the user program.
- For the S7-300 and C7-300 (exception: CPU 318-2 DP) the communication data are copied consistently into the user memory in blocks of 32 bytes in the cycle control point of the operating system. Data consistency is not guaranteed for larger data areas. If a defined data consistency is required, the communication data in the user program may not exceed 32 bytes (maximum of 8 bytes, depending on the version).
- In the S7-400 and S7-1500, on the other hand, the communication data in 462-byte blocks is not processed at the cycle control point, but in fixed time slices during the program cycle. The consistency of a tag is ensured by the system. These communication areas can then be accessed consistently using the "PUT (Page 3452)" / "GET (Page 3447)" instructions or when reading/writing tags, for example by an OP or an OS.

Note

Additional information on data consistency is provided in the description of the specific instructions.

In the S7-1500 module series, you can change parameters during ongoing operation with the instructions of S7 communication. Changed parameters are used immediately, even if a job is still active. Both can lead to data inconsistency! To avoid data inconsistencies, you should never change parameters while a job is running.

Effect on interrupt reaction times

The interrupt reaction times of the CPU are slightly extended by copying the data. The greater the quantity of data which must be transferred with absolute consistency, the longer the interrupt reaction time of a system.

Common parameters of instructions for S7 communication

Classification

The parameters of the instructions for S7 communication can be divided into the following five categories according to their functions:

1. Control parameters are used to activate an instruction.
2. Addressing parameters are used to address the remote communication partner.
3. Send parameters point to the data areas that are to be sent to the remote partner.
4. Receive parameters point to the data areas where the data received from remote partners will be entered.
5. Status parameters are used to monitor whether the instruction has completed its task without error or for the analysis of any errors that have occurred.

Control parameter

Data exchange will only be activated if the appropriate control parameters have a defined value (for example, are set) when the instruction is called or when the value has undergone a specific change since the previous call (for example, a positive edge).

Addressing parameters

Parameter	Description
ID	Reference to the local connection description (specified by the configuration of the connection).
R_ID	With the R_ID parameter, you specify that a send and a receive instruction belong together: The R_ID parameter must match in the instruction at the sending end and the instruction at the receiving end. This allows the communication of several instruction pairs via the same logic connection. • R_ID must be specified in the form DW#16#wxyzWXYZ. • The instruction pairs of a logical connection specified in R_ID must be unique for this connection.

Note

Addressing parameters ID and R_ID

You can reassign the addressing parameters ID and R_ID during runtime. The new parameters are validated with each new job after the previous job has been closed.

You can use the following options to reduce the number of instance DBs and therefore the work memory required:

1. With tag IDs, you can use several connections via one data instance block.
2. With tag R_IDs, you can define several pairs of sending and receiving instructions for one job with a single instance.
3. You can combine cases 1 and 2.

Please note that the new parameters are valid after the last job is executed. When you activate the sending operation, the R_ID parameter in the instruction at the sending end must match its counterpart at the receiving end.

Status parameter

With the status parameters, you monitor whether the instruction has completed its task correctly or whether it is still active. The status parameters also indicate errors.

Note

The status parameters are valid for one cycle only, namely from the first command following the call until the next call. As a result, you must evaluate these parameters after each instruction cycle.

Send and receive parameters

For communication instructions configured at both ends

- The number of the SD_i and RD_i parameters used must match at the send and receive end.
- The data types of the SD_i and RD_i parameters that belong together must match at the send and receive end.
- The amount of data to be sent according to the SD_i parameter must not exceed the range made available by the corresponding RD_i (does not apply to "BSEND (Page 3466)" / "BRCV (Page 3469)"). The RD_i parameters must (with exception of "BSEND"/"BRCV") have the identical data size.

If you do not keep to the rules above, this is indicated by ERROR = 1 and STATUS = 4.

Note

Supplying the send and receive parameters

With the VARIANT data type, send and receive parameters must always be supplied any time a communication instruction is called. It is not possible, for example, to supply the send parameters of the communication instructions at startup and to only trigger the send job in cyclic operation.

User data size

With the "USEND (Page 3458)", "URCV (Page 3460)", "GET (Page 3447)" and "PUT (Page 3452)" instructions, the amount of data to be transferred must not exceed a defined user data size. The maximum user data size depends on:

- The instruction used
- The communication partner.

The guaranteed minimum size of the user data for an instruction with 1-4 tags is listed in the following table:

Instruction	Partner: S7-300	Partner: S7-400	Partner: S7-1200	Partner: S7-1500
PUT / GET	160 bytes	400 bytes	160 bytes	880 bytes
USEND / URCV	160 bytes	440 bytes	-	920 bytes
BSEND / BRCV	32768/65534 bytes	65534 bytes	-	• 65534 bytes with standard access • 65535 bytes with optimized access

Further information on the user data size can be found in the technical data of the respective CPU.

Exact user data size

If the user data size specified above is insufficient you can determine the maximum byte length of the user data as follows:

First read the data block size valid for communication from the following table:

Local CPU	Remote CPU	Data block size in bytes
S7-1200	Any	240
S7-1500	S7-300	240
	S7-400	480
	S7-1200	240
	S7-1500	960

Use this value in the following table to read the maximum possible user data length in bytes as total of the parameters used. It applies for even lengths of the areas SD_i, RD_i, and ADDR_i.

For each range of uneven length the maximum possible user data length is reduced by one byte.

		Number of SD_i, RD_i, ADDR_i parameters used			
Data block size	Instruction	1	2	3	4
240 (S7-300)	PUT/GET/ USEND	160	-	-	-
240 (S7-300 via integrated interface)	PUT	212	-	-	-
	GET	222	-	-	-
	USEND	212	-	-	-
240 (S7-400)	PUT	212	196	180	164
	GET	222	218	214	210
	USEND	212	-	-	-
480 (S7-400)	PUT	452	436	420	404
	GET	462	458	454	450
	USEND	452	448	444	440
240 (S7-1200)	PUT	212	196	180	164
	GET	222	218	214	210
960 (S7-1500)	PUT	932	916	900	884
	GET	942	938	934	930
	USEND	932	928	924	920

GET: Read data from a remote CPU (S7-1200, S7-1500)

Description

With the instruction "GET", you can read data from a remote CPU.

The instruction is started on a positive edge at control input REQ:

- The relevant pointers to the areas to be read out (ADDR_i) are then sent to the partner CPU. The partner CPU can be in RUN or STOP mode.
- The partner CPU returns the data:
  - If the reply exceeds the maximum user data length, this is displayed with error code "2" at the STATUS parameter.
  - The received data is copied to the configured receive areas (RD_i) at the next call.
- Completion of this action is indicated by the status parameter NDR having the value "1".

Reading can only be activated again after the previous reading process has been completed. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being read or if the data type check results in an error.

Changes in the data areas addressed on the partner CPU are not registered by the "GET" instruction.

Requirements for using the instruction

- The "Permit access with PUT/GET communication from remote partner" function was activated in the properties of the partner CPU under "Protection".
- The blocks which you access with the "GET" instruction were created with the access type "standard".
- Make sure that the areas defined with the parameters ADDR_i and SD_i match in terms of number, length and data type.
- The area to be read (ADDR_i parameter) cannot be larger than the area for data storage (RD_i parameter).

Parameters

The following table shows the parameters of the "GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange on a positive edge.
ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameters for specifying the connection to the partner CPU.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: – 0000H: Neither warning nor error – <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
ADDR_1	InOut	REMOTE	I, Q, M, D	Pointers to the areas on the partner CPU that are to be read. When the REMOTE pointer accesses a DB, the DB must always be specified. Example: P#DB10.DBX5.0 Byte 10.
ADDR_2	InOut	REMOTE		
ADDR_3	InOut	REMOTE		
ADDR_4	InOut	REMOTE		
RD_1	InOut	VARIANT	I, Q, M, D, L	Pointers to the areas on the local CPU in which the read data is entered.
RD_2	InOut	VARIANT		
RD_3	InOut	VARIANT		
RD_4	InOut	VARIANT		

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "GET" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	2	• Negative acknowledgment from the partner device. The function cannot be executed. • Response from the remote station exceeds the maximum user data length (see: Common parameters of instructions for S7 communication (Page 3444)). • Access protection is activated on the partner CPU. Deactivate access protection in the CPU settings.
1	4	Error in the pointers to the data storage RD_i: • Data types of the parameters RD_i and ADDR_i are not compatible with each other. • The length of the area RD_i is smaller than the length of the data of the ADDR_i parameter that is to be read.
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).
1	W#16#80C3	(only with S7-1500) • Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).

Note

Data consistency

Data is received consistently if you read the part of the receive area RD_i currently being used completely before initiating another job.

Example

In the following example use an S7 connection and in CPU 1 read out a data record of CPU 2. The data record to be sent is of the data type INT.

Requirement

- Two CPUs of the S7-1500 series are available and connected to each other over PROFINET. The connection is not yet configured.
- A low protection level under “<CPU> > Properties > Protection“ ensures that read and write access is permitted for the CPUs.
- Access with the PUT/GET instruction is permitted.

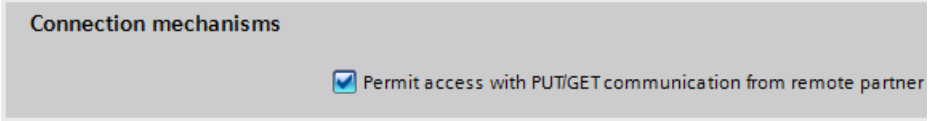

Program of CPU 1

Create six tags in a global data block for storing the data of “GET“.

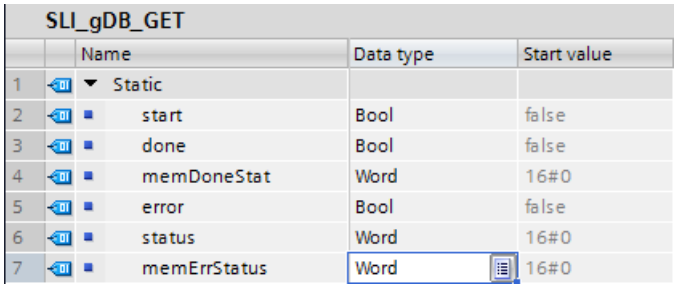

SLI_gDB_GET

	Name	Data type	Start value
1	Static		
2	start	Bool	false
3	done	Bool	false
4	memDoneStat	Word	16#0
5	error	Bool	false
6	status	Word	16#0
7	memErrStatus	Word	16#0

Create one tag in a global data block for storing the data record to be received.

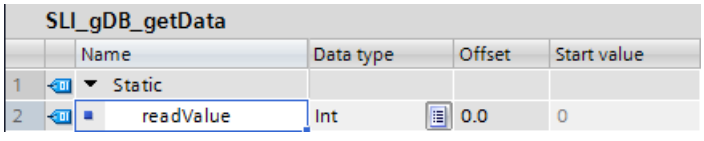

SLI_gDB_getData

	Name	Data type	Offset	Start value
1	Static			
2	readValue	Int	0.0	0

Network 1: Interconnect the parameters of the "GET" instruction as follows.

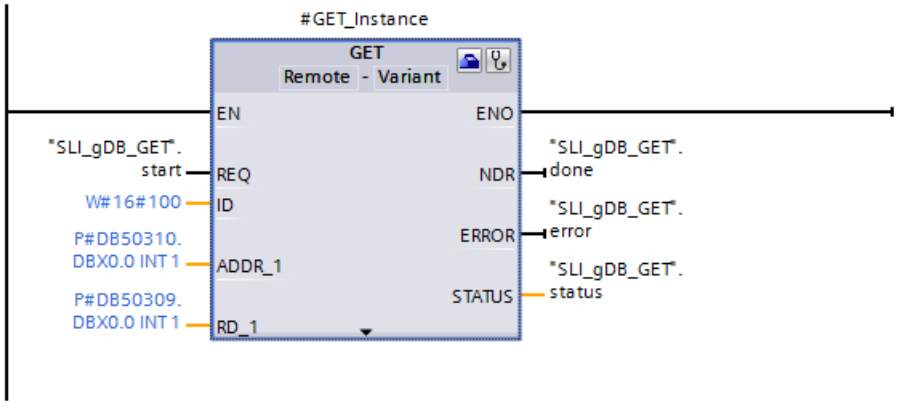

Network 2: Save the status for an error of GET as follows.

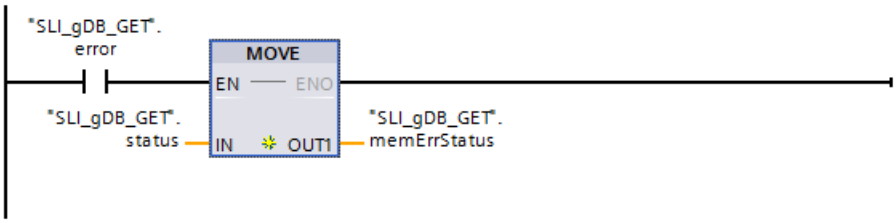

Network 3: Save the success status of GET as follows.

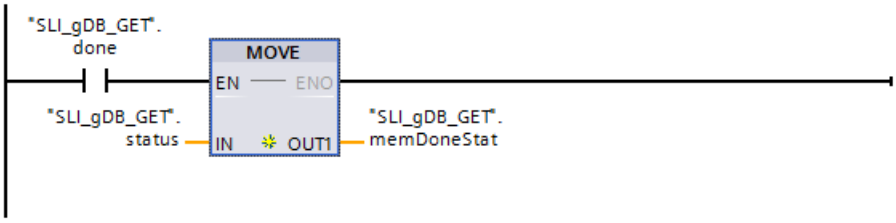

Configuration of GET

To interconnect the input parameter ID, open the wizard of the "GET" instruction with “Properties > Configuration“.

Make the following settings for the connection:

Section	Setting
End point	Select the communication partner. The remaining connection data will be entered automatically. An S7 connection is created automatically and its identifier entered at the input parameter ID.
Active	Make sure that the active connection goes out from CPU 1.

Program of CPU 2

Create one tag in a global data block for storing the data record to be sent.

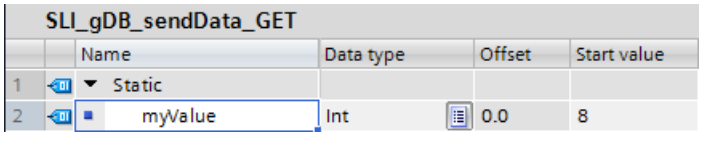

In “Properties > Attributes“ of the data block disable the optimized block access.

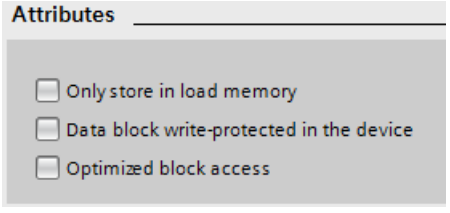

Behavior of GET

When the input parameter REQ ("start") supplies the signal state "TRUE", the "GET" instruction is started. The instruction calls up the connection data and makes contact with the

communications partner of the S7 connection being used. To achieve this, the identifier of the S7 connection is stored at the input parameter ID.

Via the parameter ADDR_1 the storage location of the data record of CPU 2 to be sent (“myValue“) is identified. The data record (“myValue“) is read out and at the parameter RD_1 the identified storage location (“readValue“) is entered.

SLI_gDB_getData

	Name	Data type	Offset	Start value	Monitor value
1	Static				
2	readValue	Int	0.0	0	8

Successful processing is indicated by the output parameter NDR ("done") with the signal state "TRUE" and the output parameter STATUS ("status") with the value "16#0000". Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "memDoneStat". The output parameter ERROR ("error") or the tag "memErrStatus" indicates that processing took place without errors.

SLI_gDB_GET

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	done	Bool	false	FALSE
4	memDoneStat	Word	16#0	16#0000
5	error	Bool	false	FALSE
6	status	Word	16#0	16#0000
7	memErrStatus	Word	16#0	16#0000

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

PUT: Write data to a remote CPU (S7-1200, S7-1500)

Description

You can write data to a remote CPU with the instruction "PUT".

The instruction is started on a positive edge at control input REQ:

- The pointers to the areas to be written (ADDR_i) and the data (SD_i) are then sent to the partner CPU. The partner CPU can be in RUN or STOP mode.
- The data to be sent is copied from the configured send areas ((SD_i). The partner CPU saves the sent data under the addresses supplied with the data and returns an execution acknowledgment.
- If no errors occur, this is indicated at the next instruction call with status parameter DONE = "1". The writing process can only be activated again after the last job is complete.

Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being written or if the execution check results in an error.

Requirements for using the instruction

- The "Permit access with PUT/GET communication from remote partner" function was activated in the properties of the partner CPU under "Protection".
- The blocks which you access with the "PUT" instruction were created with the access type "standard".
- Make sure that the areas defined with the parameters ADDR_i and SD_i match in terms of number, length and data type.
- The area to be written (ADDR_i parameter) must be as large as the send area (SD_i parameter).

Parameters

The following table shows the parameters of the "PUT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange on a positive edge.
ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameters for specifying the connection to the partner CPU.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: – 0000H: Neither warning nor error – <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
ADDR_1	InOut	REMOTE	I, Q, M, D	Pointers to the areas on the partner CPU to which the data will be written. When the REMOTE pointer accesses a DB, the DB must always be specified. Example: P#DB10.DBX5.0 Byte 10. When transferring data structures (e.g., Struct) the data type CHAR must be used at the ADDR_i parameters.
ADDR_2	InOut	REMOTE		
ADDR_3	InOut	REMOTE		
ADDR_4	InOut	REMOTE		

Parameter	Declaration	Data type	Memory area	Description
SD_1	InOut	VARIANT	I, Q, M, D, L	Pointers to the areas on the local CPU which contain the data to be sent. Only the data types BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL are permitted. When transferring data structures (e.g., Struct) the data type CHAR must be used at the SD_i parameters.
SD_2	InOut	VARIANT		
SD_3	InOut	VARIANT		
SD_4	InOut	VARIANT		

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

The following table contains all specific error information for the "PUT" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example, cable, CPU off, CP in STOP mode). • Connection to partner not yet established.
1	2	• Negative acknowledgment of partner CPU. The function cannot be executed. • Access on the partner CPU was not granted. Activate the access in the CPU settings.
1	4	Error in the pointers to the data storage: • Data types of the parameters SD_i and ADDR_i are not compatible with each other. • The length of the area SD_i is greater than the length of the data of the ADDR_i parameter that is to be written. • Not possible to access SD_i. • Maximum user data size exceeded. • Number of the parameters SD_i and ADDR_i does not match.
1	8	Access error with the partner CPU (e.g. DB not loaded or write-protected).
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).
1	W#16#80C3	(only with S7-1500) • Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).

Data consistency

When a send operation is activated (positive edge at REQ), the data to be sent from the send area SD_i is copied from the user program. After the block call, you can write to these areas without corrupting the current send data.

Note

The send operation is only complete when the DONE status parameter has the value "1".

Example

In the following example use an S7 connection and transfer a data record from CPU 1 to CPU 2. The data record to be sent is of the data type INT.

Requirement

- Two CPUs of the S7-1500 series are available and connected to each other over PROFINET. The connection is not yet configured.
- A low protection level under “<CPU> > Properties > Protection“ ensures that read and write access is permitted for the CPUs.
- Access with the PUT/GET instruction is permitted.

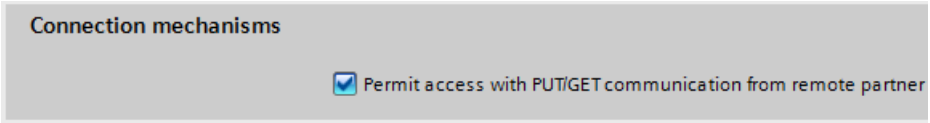

Program of CPU 1

Create six tags in a global data block for storing the data of “PUT“.

SLI_gDB_PUT

	Name	Data type	Start value
1	Static		
2	start	Bool	false
3	done	Bool	false
4	memDoneStat	Word	16#0
5	error	Bool	false
6	status	Word	16#0
7	memErrStatus	Word	16#0

Create one tag in a global data block for storing the data record to be sent.

SLI_gDB_putData

	Name	Data type	Offset	Start value
1	Static			
2	writeValue	Int	0.0	8

Network 1: Interconnect the parameters of the "PUT" instruction as follows.

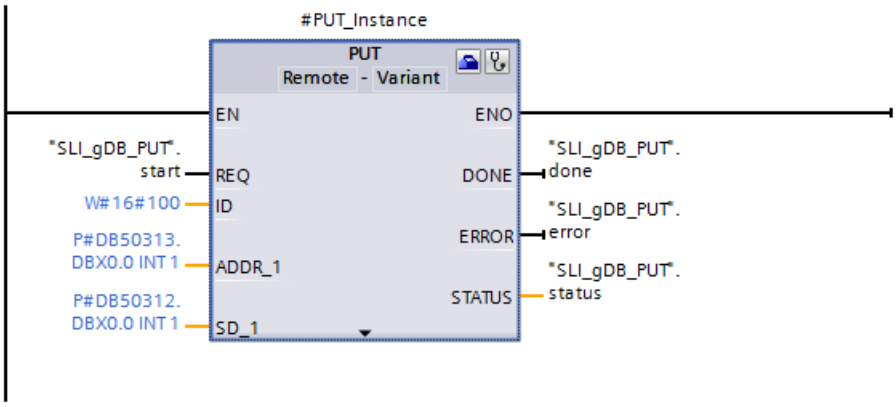

Network 2: Save the status for an error of PUT as follows.

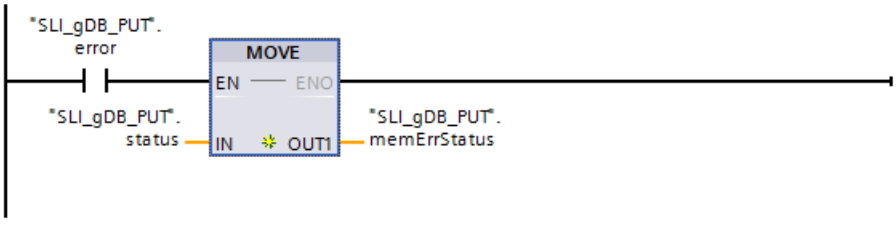

Network 3: Save the success status of PUT as follows.

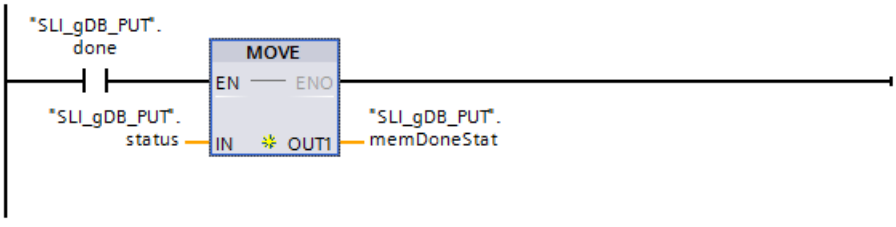

Configuration of PUT

To interconnect the input parameter ID, open the wizard of the "PUT" instruction with “Properties > Configuration“.

Make the following settings for the connection:

Section	Setting
End point	Select the communication partner. The remaining connection data will be entered automatically. An S7 connection is created automatically and its identifier entered at the input parameter ID.
Active	Make sure that the active connection goes out from CPU 1.

Program of CPU 2

Create one tag in a global data block for storing the data record to be received.

SLI_gDB_rcvData_PUT

	Name	Data type	Offset	Start value
1	Static			
2	myValue	Int	0.0	0

In “Properties > Attributes“ of the data block disable the optimized block access.

Attributes

- [ ] Only store in load memory
- [ ] Data block write-protected in the device
- [ ] Optimized block access

Behavior of PUT

When the input parameter REQ ("start") returns the signal state "TRUE", the "PUT" instruction is started. The instruction calls up the connection data and makes contact with the communications partner of the S7 connection being used. To achieve this, the identifier of the S7 connection is stored at the input parameter ID.

At the parameter SD_1 the storage location of the data record of CPU 1 to be sent (“writeValue“) is identified. At the parameter ADDR_1 the storage location (“myValue“) of the data record in CPU 2 is identified. The data record (“writeValue“) is transferred and indicated at the storage location of CPU 2 (“myValue“).

SLI_gDB_rcvData_PUT

	Name	Data type	Offset	Start value	Monitor value
1	Static				
2	myValue	Int	0.0	0	8

Successful processing is indicated by the output parameter DONE ("done") with the signal state "TRUE" and the output parameter STATUS ("status") with the value "16#0000". Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "memDoneStat". The output parameter ERROR ("error") or the tag "memErrStatus" indicates that processing took place without errors.

SLI_gDB_PUT

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	done	Bool	false	FALSE
4	memDoneStat	Word	16#0	16#0000
5	error	Bool	false	FALSE
6	status	Word	16#0	16#0000
7	memErrStatus	Word	16#0	16#0000

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

Other

USEND: Send data uncoordinated (S7-1500)

Description

The "USEND" instruction sends data to a remote partner instruction of type "URCV (Page 3460)". The sending process is carried out without coordination with the partner instruction. This means that the data transfer is carried out without acknowledgment by the partner instruction.

When a send operation is activated (positive edge at REQ), the data to be sent from the send areas SD_i are copied from the user program. After the instruction call, you can write to these send areas again without corrupting the current send data.

Successful completion of the send operation is indicated by the value of the DONE status parameter being set to "1".

Parameters

The following table shows the parameters of the instruction "USEND":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange on a positive edge.
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter R_ID for defining the instruction pairs "USEND" and "URCV". See also: Common parameters of instructions for S7 communication (Page 3444)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred. STATUS supplies detailed information on the type of error.

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.
SD_i (1≤ i ≤4)	InOut	VARIANT	I, Q, M, D, L	Pointer on the i-th send area. Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL or STRUCT. The maximum user data size for SD_i parameters depends on the partner CPU ("URCV" instruction) and on the number of parameters used. Additional information is available in: Common parameters of instructions for S7 communication (Page 3444)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	4	• Errors in the send area pointers SD_i relating to the data length or the data type. • Maximum user data length was exceeded.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	Value at R_ID parameter already exists in the connection specified at the ID parameter (R_ID value must be unique for the connection).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).
1	W#16#80C3	(only with S7-1500) • Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).

Example

You can find the example here: Program example for USEND & URCV (Page 3462).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

URCV: Receive data uncoordinated (S7-1500)

Description

The "URCV" instruction receives data asynchronously from a remote partner instruction of type "USEND (Page 3458)" and copies this data into the configured receive areas.

The instruction is ready to receive if there is a logical 1 at the EN_R input. An active job can be canceled with EN_R=0.

The receive data areas are referenced by the parameters RD_1 to RD_4. Make sure that the areas defined by the parameters RD_i / RD_1 and SD_i / SD_1 (with the associated partner instruction "USEND (Page 3458)") match in terms of number and length.

Successful completion of the copy operation is indicated when the value of the NDR status parameter is set to logical "1". Once the status parameter NDR has changed to "1", there will be new receive data in your receive areas (RD_i). A new block call may cause these data to be overwritten with new receive data. If you want to prevent this, call "URCV" (for example, during cyclic block processing) with the value 0 at EN_R until you have finished processing the received data.

Parameters

The following table shows the parameters of the instruction "URCV":

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter for defining the instruction pairs "USEND" and "URCV". See also: Common parameters of instructions for S7 communication (Page 3444)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error • 1: An error has occurred, STATUS supplies detailed information on the type of error.

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.
RD_i (1≤ i ≤4)	InOut	VARIANT	I, Q, M, D, L	Pointer to the i-th receive area: Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL or STRUCT. Additional information is available in: Common parameters of instructions for S7 communication (Page 3444)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS (decimal)	Explanation
0	9	Warning: older received data were overwritten by newer received data.
0	11	Warning: The received data is already being processed in a priority class with lower priority (error can occur when copying data to the receive area).
0	25	Communication has started. The job is being processed.
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	4	Errors in the receive area pointers RD_i relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	Value at R_ID parameter already exists in the connection specified at the ID parameter (R_ID value must be unique for the connection).
1	19	The associated "USEND (Page 3458)" instruction sends data faster than it can be copied to the receive areas by "URCV".
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).
1	W#16#80C3	(only with S7-1500) • Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).

Example

You can find the example here: Program example for USEND & URCV (Page 3462).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

Program example for USEND & URCV

Introduction

In the following example you create an S7 connection between two CPUs of the S7-1500 series. You send a character string (data type STRING) from CPU 1 to CPU 2. The data transfer is carried out without acknowledgement by the communication partner.

Requirement

- Two CPUs of the S7-1500 series are available and connected to each other over PROFINET. An S7 connection is configured.

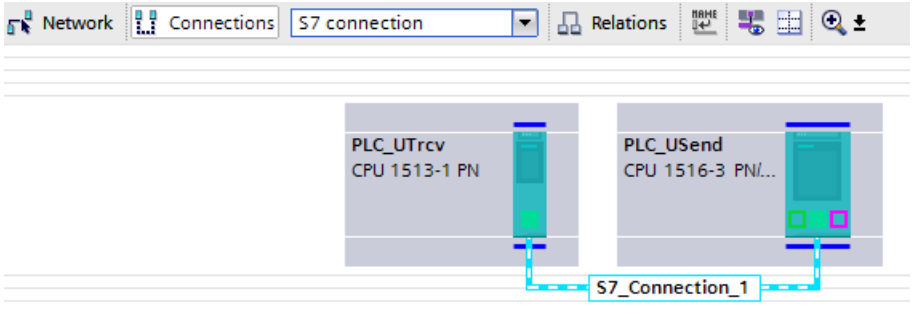

- A low protection level under "<CPU> > Properties > Protection" ensures that read and write access is permitted for the CPUs.

Program of CPU 1

Create the following tags for storing the data of USEND in a global data block.

SLI_gDB_USEND

	Name	Data type	Start value
1	Static		
2	start	Bool	false
3	connectionID	CONN_PRG	16#0100
4	instructionPair	CONN_R_ID	16#01
5	done	Bool	false
6	error	Bool	false
7	status	Word	16#0
8	memErrStatus	Word	16#0
9	sendData	String	'HelloData'

Create an FB "SLI_FB_USEND". Create the following local tags in it.

7	Static			
8	USEND_Instance	USEND		
9	done	Bool	false	Non-ret...

Network 1: Interconnect the parameter of the instruction "USEND" as follows:

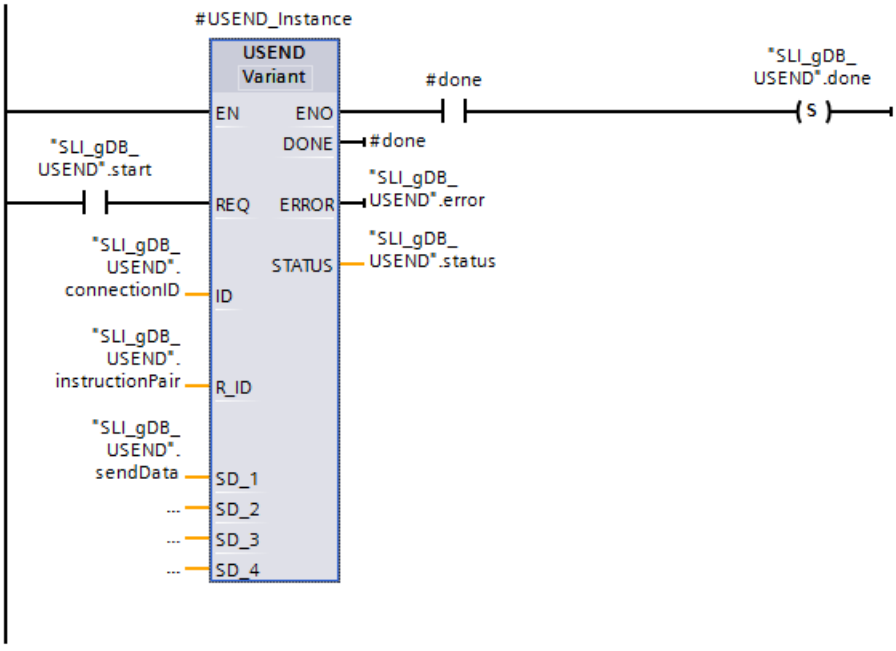

Network 2: In case of error of USEND, save the status as follows.

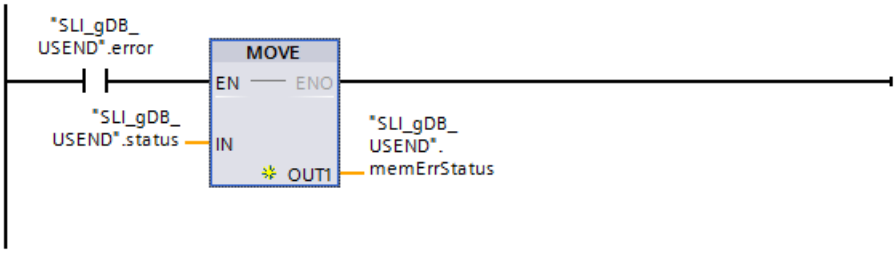

Program of CPU 2

Create the following tags for storing the data of URCV in a global data block.

SLI_gDB_URCV

		Name	Data type	Start value
1		Static		
2		start	Bool	false
3		connectionID	CONN_PRG	16#0100
4		instructionPair	CONN_R_ID	16#01
5		done	Bool	false
6		error	Bool	false
7		status	Word	16#0
8		memErrStatus	Word	16#0
9		rcvData	String	''

Create an FB "SLI_FB_URCV". Create the following local tags in it.

7	Static			
8	URCV_Instance	URCV		
9	done	Bool	false	Non-ret...

Network 1: Interconnect the parameter of the instruction "URCV" as follows:

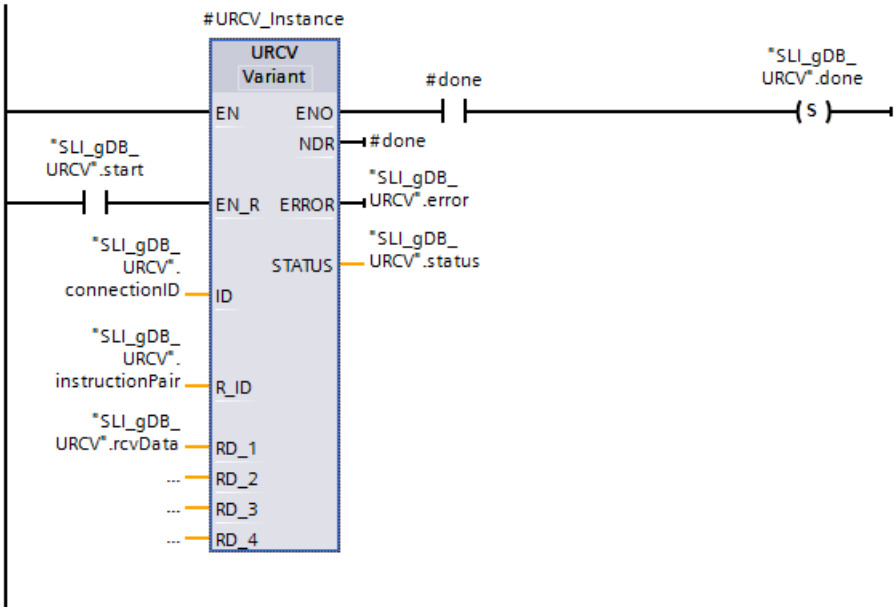

Network 2: In case of error of URCV, save the status as follows.

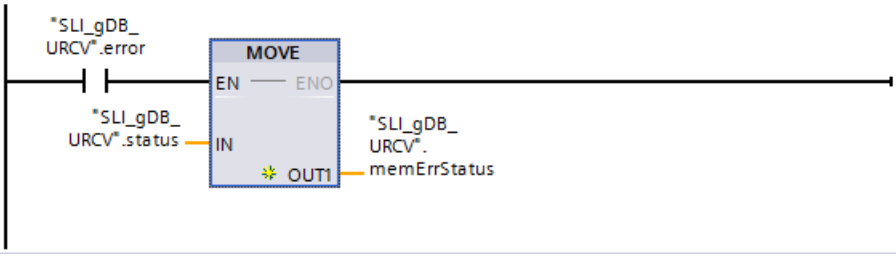

Assigning communication connection

The addressing parameters for connecting the two CPUs and for defining the instruction pair must be uniform.

- In each case store the hexadecimal value of the hardware ID of the configured S7 connection at the input parameter ID ("connectionID").
  You can find the hardware identifier in the "Network view" under "Connections".

Network overview | Connections | Relations | I/O communication | VPN

Local connection n...	Local end point	Local ID (hex)	Partner ID (hex)	Partner	Connection type
S7_Connection_1	PLC_UTrcv	100	100	PLC_USend	S7 connection
S7_Connection_1	PLC_USend	100	100	PLC_UTrcv	S7 connection

- In each case store a self-selected identifier (as hex value, in byte size) for the instruction pair at the input parameter R_ID ("instructionPair"). The identifier must not be already assigned for another instruction pair.

Behavior of CPU 1

When the input parameter REQ ("start") supplies the signal state "TRUE", the "USEND" instruction is started. The "USEND" instruction hereby copies the data record detected at input parameter SD_1 ("sendData") to the work memory of the CPU. Successful copying of the data record is indicated by the output parameter DONE ("#done") with the signal state "TRUE". Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "done".

Corresponding to the input parameter ID ("connectionID") and R_ID ("instructionPair"), the instruction "URCV" of the CPU 2 as receiver of the data record ("sendData" is addressed with the value "HelloData"). Data transfer via S7 connection continues without further participation by USEND ("start" may be set to "FALSE").

The output parameter ERROR ("error") or the tag "memErrStatus" indicates that processing of USEND in the example is taking place without errors.

SLI_gDB_USEND

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	connectionID	CONN_PRG	16#0100	16#0100
4	instructionPair	CONN_R_ID	16#01	16#0000_0001
5	done	Bool	false	TRUE
6	error	Bool	false	FALSE
7	status	Word	16#0	16#0000
8	memErrStatus	Word	16#0	16#0000
9	sendData	String	'HelloData'	'HelloData'

Behavior of CPU 2

CPU 2 receives a data record of CPU 1 via the S7 connection. Corresponding to the input parameter ID ("connectionID") and R_ID ("instructionPair"), the instruction "URCV" knows the receiver of the data record.

When the input parameter EN_R ("start") supplies the signal state "TRUE", the "URCV" instruction is started and is ready to receive. The instruction "URCV" reads out the transferred data and copies the data record into the value area at the input parameter RD_1 ("rcvData").

Successful copying of the data record is indicated by the output parameter NDR ("#done") with the signal state "TRUE". Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "done".

The output parameter ERROR ("error") or the tag "memErrStatus" indicates that processing of URCV in the example is taking place without errors.

SLI_gDB_URCV

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	connectionID	CONN_PRG	16#0100	16#0100
4	instructionPair	CONN_R_ID	16#01	16#0000_0001
5	done	Bool	false	TRUE
6	error	Bool	false	FALSE
7	status	Word	16#0	16#0000
8	memErrStatus	Word	16#0	16#0000
9	rcvData	String	''	'HelloData'

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

USEND: Send data uncoordinated (Page 3458)

URCV: Receive data uncoordinated (Page 3460)

BSEND: Send data in segments (S7-1500)

Description

The "BSEND" instruction sends data to a remote partner instruction of type "BRCV (Page 3469)". With this type of data transfer, more data can be transported between the communication partners than is possible with all other communication instructions for configured S7 connections. In each case, the maximum data volume amounts to 65534 bytes (standard access) or 65535 bytes (optimized access) for the integrated interface and for SIMATIC Net CP.

Functional description

You specify the instruction pair "BSEND" and "BRCV" with the input parameter R_ID. The R_ID parameter must be identical in the related instructions.

The send job is activated after the instruction is called and a positive edge is detected at control input REQ. After the call, "BSEND" is not processed in the background, which means the data can only be read within the user program.

The data area to be transmitted is segmented. Each segment is sent individually to the partner. Each segment is acknowledged by the partner after the application of this segment by "BRCV (Page 3469)". If the data is segmented, "BSEND" must be called multiple times until all segments have been transferred.

The data area of the data to be sent is specified by SD_1. To ensure data consistency, you may only write to the part of the currently used sending area SD_1 after the current send operation is complete. This is the case when the value of the status parameter DONE changes to "1".

The length of the send data is specified by LEN based on the job. With LEN = "0", all data addressed with the SD_1 parameter is sent.

If there is a positive edge at control input R, the current sending process is canceled.

Due to the asynchronous data transfer, a new data transfer can only be initiated if the previous data has been fetched by a partner instruction call. When the data has been collected, the status parameter "NDR" is set in the partner instruction "BRCV".

Note

Migration of S7-400 user programs

An S7-400 CPU interprets the parameter SD_1 as pointer and not as data area.

With S7-1500, LEN may not exceed the area of SD1. This was permitted with S7-400. Recommendation: Use the maximum size for the LEN parameter (65534 bytes for the integrated interface) as size of the data area at the SD_1 parameter.

Parameters

The following table shows the parameters of the instruction "BSEND":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange at a positive edge
R	Input	BOOL	I, Q, M, D, L or constant	Control parameter reset, activates an abort of the current data exchange at a positive edge
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter for defining the instruction pairs "BSEND" and "BRCV (Page 3469)". See also: Common parameters of instructions for S7 communication (Page 3444)
SD_1	InOut	VARIANT	I, Q, M, D, L	Pointer to send area When transferring structures, the structures must be identical at the sending and receiving end.

Parameter	Declaration	Data type	Memory area	Description
LEN	InOut	WORD	I, Q, M, D, L	Length of the block of data to be sent in bytes. With LEN = "0", all data is sent by SD_1.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for "BSEND" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	2	Negative acknowledgment of partner instruction. The instruction cannot be executed.
1	3	R_ID is unknown on the connection specified by the ID or the receive block has not yet been called.
1	4	• Error in the data length or data type in the send area pointer SD_1. • Value for LEN is greater than area SD_1.
1	5	Reset request was executed.
1	6	Partner instruction is in DISABLED status (EN_R has the value "0"). Also check the input parameters of "BRCV (Page 3469)" for consistency with "BSEND".
1	7	"BRCV (Page 3469)" partner instruction not called since the last data transfer.
1	8	Access to remote object in the user memory was rejected: The target range for the associated "BRCV (Page 3469)" is too small. ERROR = 1, STATUS = 4 or ERROR = 1, STATUS = 10 reported at the output parameters of "BRCV (Page 3469)".
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	R_ID already exists in the connection.

ERROR	STATUS (decimal)	Explanation
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).
1	W#16#80C3	(only with S7-1500) • Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).

Example

You can find the example here: Program example for BSEND & BRCV (Page 3471).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

BRCV: Receive data in segments (S7-1500)

Description

The "BRCV" instruction receives data from a remote partner instruction of type "BSEND (Page 3466)". The R_ID parameter must be identical in the related instructions.

The instruction is ready to receive data (STATUS = 25) after it is called with the value "1" at control input EN_R. An active job can be canceled with EN_R=0.

The maximum receive area is specified by RD_1. Data will be received consistently if you fully evaluate the part of the RD_1 receive area currently in use before calling the block again with value 1 at control input EN_R.

After each received data segment, an acknowledgment is sent to the partner instruction. If there are multiple segments it will be necessary to call "BRCV" several times until all of the segments have been received. Asynchronous data receipt is indicated by STATUS = 17. The amount of data currently received is indicated at the parameter LEN. The RD_1 parameter must remain constant during the operation.

Value "1" at the NDR status parameter indicates that all data segments have been received without error. The received data remains unchanged until the next call with EN_R=1.

Parameters

The following table shows the parameters of the instruction "BRCV":

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter for defining the instruction pairs "BSEND (Page 3466)" and "BRCV". See also: Common parameters of instructions for S7 communication (Page 3444)
RD_1	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area. When transferring structures, the structures must be identical at the sending and receiving end.
LEN	InOut	WORD	I, Q, M, D, L	Length of the data already received in bytes.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error • 1: An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for "BRCV" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	17	Warning: Instruction receives data asynchronously. The LEN parameter shows the amount of data already received in bytes.
0	25	Communication has started. The job is being processed.
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	2	Function cannot be executed (protocol error)
1	4	Error in the receive area pointer RD_1 relating to the data length or data type. The block of data sent is longer than the receive area.
1	5	Reset request received, incomplete transfer.

ERROR	STATUS (decimal)	Explanation
1	8	Access error in associated "BSEND (Page 3466)": After the last valid data segment has been sent, ERROR = 1 and STATUS = 4 or ERROR = 1 and STATUS = 10 is reported.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	Value at R_ID parameter already exists in the connection specified at the ID parameter (R_ID value must be unique for the connection).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).
1	W#16#80C3	(only with S7-1500) • Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).

Example

You can find the example here: Program example for BSEND & BRCV (Page 3471).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

Program example for BSEND & BRCV

Introduction

In the following example you create an S7 connection between two CPUs of the S7-1500 series. You send a data record from CPU 1 to CPU 2 using BSEND and BRCV. The data transfer is carried out with acknowledgement by BRCV.

Requirement

- Two CPUs of the S7-1500 series are available and connected to each other over PROFINET. An S7 connection is configured.

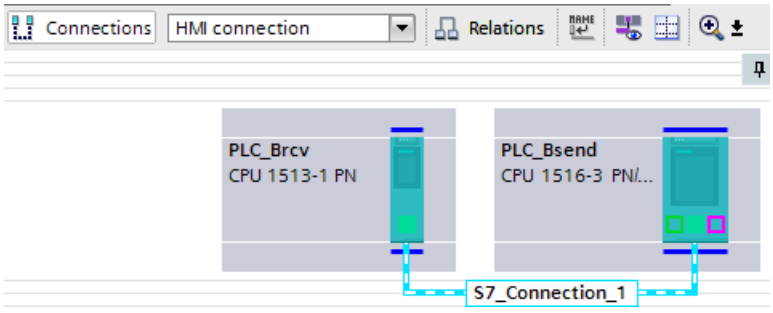

- A low protection level under "<CPU> > Properties > Protection" ensures that read and write access is permitted for the CPUs.

Program of CPU 1

Create the following tags for storing the data of BSEND in a global data block.

SLI_gDB_BSEND

		Name	Data type	Start value
1		Static		
2		start	Bool	false
3		reset	Bool	false
4		connectionID	CONN_PRG	16#0100
5		instructionPair	CONN_R_ID	16#02
6		maxLength	UInt	0
7		done	Bool	false
8		error	Bool	false
9		status	Word	16#0
10		memErrStatus	Word	16#0

For the data record, create the following PLC data type.

BSEND_User

		Name	Data type	Default value
1		prename	String	''
2		lastname	String	''
3		userId	UInt	0

For the data transfer, create the following data block based on the created PLC data type ("BSEND_User").

SLI_plcDB_sendData_BSEND

		Name	Data type	Start value
1		Static		
2		prename	String	'Simon'
3		lastname	String	'Sample'
4		userId	UInt	1

Create an FB "SLI_FB_BSEND". Create the following local tags in it.

7	Static			
8	done	Bool	false	Non-retain
9	BSEND_Instance	BSEND		

Network 1: Interconnect the parameter of the instruction "BSEND" as follows:

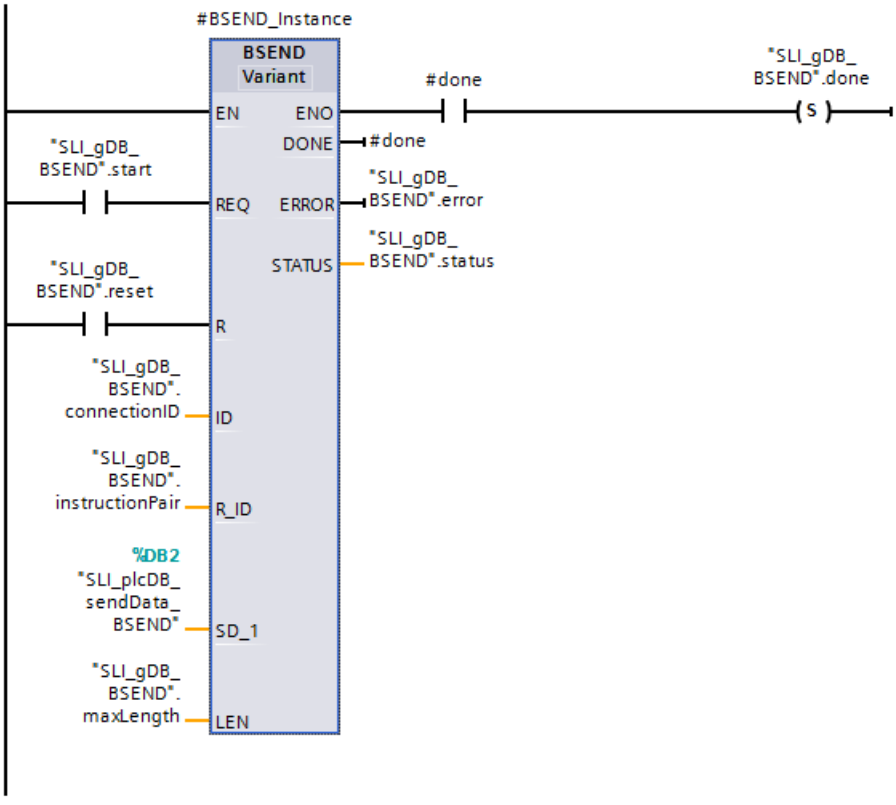

Network 2: In case of error of BSEND, save the status as follows.

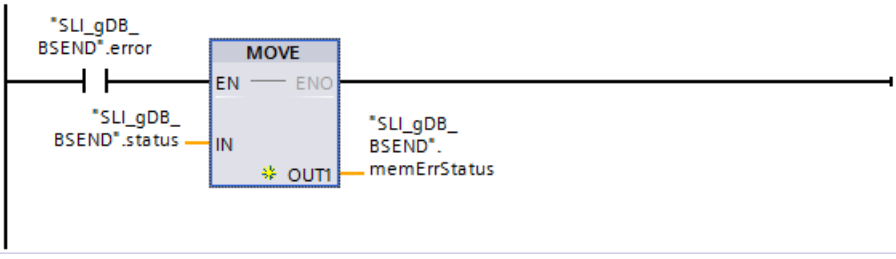

Program of CPU 2

Create the following tags for storing the data of BRCV in a global data block.

SLI_gDB_BRCV			
	Name	Data type	Start value
1	Static		
2	start	Bool	false
3	connectionID	CONN_PRG	16#0100
4	instructionPair	CONN_R_ID	16#02
5	done	Bool	false
6	error	Bool	false
7	status	Word	16#0
8	memErrStatus	Word	16#0
9	readLength	UInt	0

For the data record, create the following PLC data type.

BSEND_User			
	Name	Data type	Default value
1	prename	String	''
2	lastname	String	''
3	userId	UInt	0

For the data transfer, create the following data block based on the created PLC data type ("BSEND_User").

SLI_plcDB_rcvData_BRCV			
	Name	Data type	Start value
1	Static		
2	prename	String	''
3	lastname	String	''
4	userId	UInt	0

Create an FB "SLI_FB_BSEND". Create the following local tags in it.

7	Static			
8	done	Bool	false	Non-ret...
9	length	UInt	0	Non-retain
10	BRCV_Instance	BRCV		

Network 1: Interconnect the parameter of the instruction "BRCV" as follows:

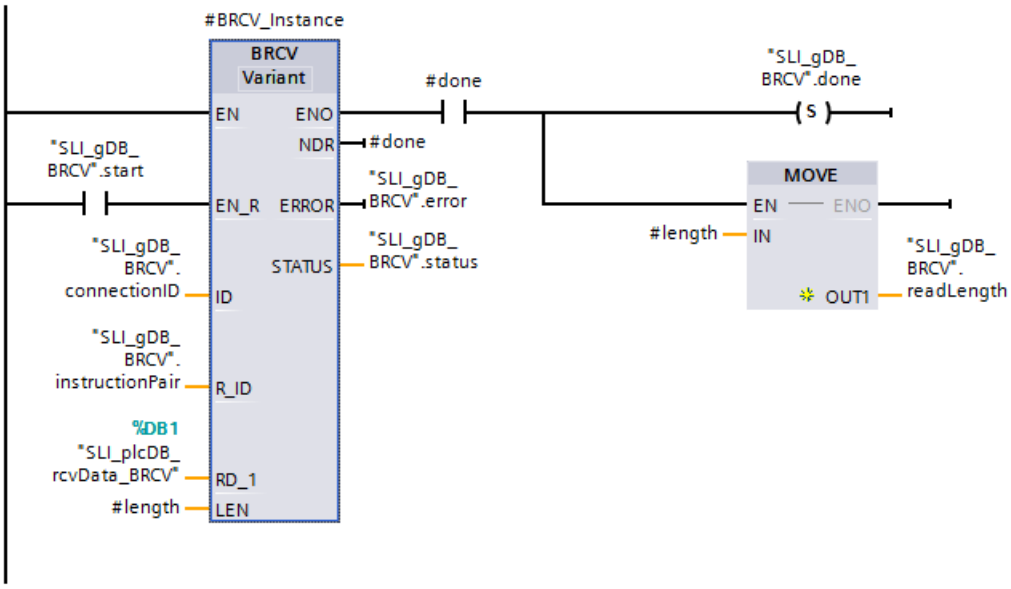

Network 2: In case of error of BRCV, save the status as follows.

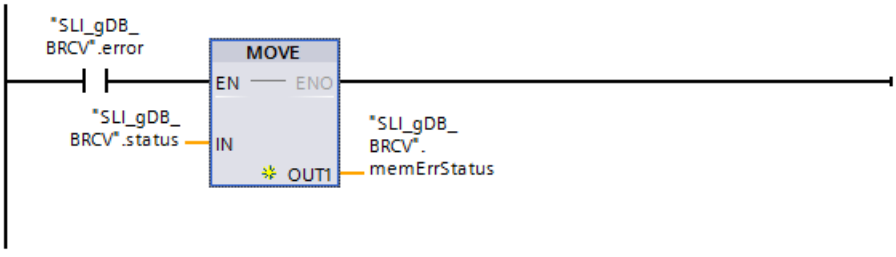

Assigning communication connection

The addressing parameters for connecting the two CPUs and for defining the instruction pair must be uniform.

- In each case store the hexadecimal value of the hardware ID of the configured S7 connection at the input parameter ID ("connectionID").
  You can find the hardware identifier in the "Network view" under "Connections".

Network overview | Connections | Relations | I/O communication

Local connection name	Local end point	Local ID (hex)	Partner ID (hex)
S7_Connection_1	PLC_Bsend	100	100
S7_Connection_1	PLC_Brcv	100	100

- In each case store a self-selected identifier (as hex value, in byte size) for the instruction pair at the input parameter R_ID ("instructionPair"). The identifier must not be already assigned for another instruction pair.

Behavior of CPU 1

Corresponding to the input parameter ID ("connectionID") and R_ID ("instructionPair"), the instruction "BRCV" of the CPU 2 is addressed as receiver of the data record ("SLI_plcDB_sendData_BSEND").

When the input parameter REQ ("start") supplies the signal state "TRUE", the "BSEND" instruction is started. For multiple calls, the "BSEND" instruction transmits the data record detected at input parameter SD_1 ("SLI_plcDB_sendData_BSEND") in segments to the CPU 2. According to the value "0" of the input parameter LEN ("maxLength"), is the length of the data record.

Successful transfer of the data record () is indicated by the output parameter DONE ("#done") with the signal state "TRUE". Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "done".

The output parameter ERROR ("error") or the tag "memErrStatus" indicates that processing in the example is taking place without errors.

SLI_gDB_BSEND

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	reset	Bool	false	FALSE
4	connectionID	CONN_PRG	16#0100	16#0100
5	instructionPair	CONN_R_ID	16#02	16#0000_0002
6	maxLength	UInt	0	0
7	done	Bool	false	TRUE
8	error	Bool	false	FALSE
9	status	Word	16#0	16#0000
10	memErrStatus	Word	16#0	16#0000

A new data transfer can only be initiated if the previous data has been fetched by calling of BRCV.

Behavior of CPU 2

CPU 2 receives a data record (in segments) of CPU 1 via the S7 connection. Corresponding to the input parameter ID ("connectionID") and R_ID ("instructionPair"), the instruction "BRCV" knows the receiver of the data record.

When the input parameter EN_R ("start") supplies the signal state "TRUE", the "BRCV" instruction is started and is ready to receive. Over several calls the "BRCV" instruction reads out the data record in segments and saves the data record at the input parameter RD_1 ("SLI_plcDB_rcvData_BRCV").

SLI_plcDB_rcvData_BRCV

	Name	Data type	Start value	Monitor value
1	Static			
2	prename	String	''	'Simon'
3	lastname	String	''	'Sample'
4	userId	UInt	0	1

Successful receipt of the entire data record is indicated by the output parameter NDR ("#done") with the signal state "TRUE". The length in BYTES of the actually transmitted data record is

detected by the output parameter LEN ("#length"). This value is only displayed during the success status. Then "0" is detected. Because the values of the output parameters are only displayed for the moment of their validity, save the success status of "#done" in the tag "done" and the length in BYTES in the tag "readLength".

The output parameter ERROR ("error") or the tag "memErrStatus" indicates that processing in the example is taking place without errors.

SLI_gDB_BRCV

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	connectionID	CONN_PRG	16#0100	16#0100
4	instructionPair	CONN_R_ID	16#02	16#0000_0002
5	done	Bool	false	TRUE
6	error	Bool	false	FALSE
7	status	Word	16#0	16#0019
8	memErrStatus	Word	16#0	16#0000
9	readLength	UInt	0	514

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

BSEND: Send data in segments (Page 3466)

BRCV: Receive data in segments (Page 3469)

4.1.6.2 Open User Communication

Data consistency

Ensuring data consistency

In the S7-1500 module series, you can change parameters during ongoing operation with the instructions of the Open User Communication. Changed parameters are used immediately, even if a job is still active. Both can lead to data inconsistency! To avoid data inconsistencies, you should never change parameters while a job is running.

TSEND_C: Establishing a connection and sending data

TSEND_C: Send data via Ethernet (S7-1200)

Validity

The following description of the "TSEND_C" instruction applies to S7-1200 CPUs with firmware version < V4.0.

Description

The "TSEND_C" instruction sets up and establishes a TCP or ISO-on-TCP communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU. The connection description specified at the CONNECT parameter is used to set up the communication connection.

The instruction is executed asynchronously and has the following functions:

- A communication connection is set up and established:
  The communication connection is set up and established with CONT=1. Once the connection is successfully established, the DONE parameter is set to "1" for one cycle. An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TSEND_C" again. For information on the number of possible communication connections, refer to the technical specifications for your CPU.
- Sending data via an existing communication connection:
  You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. Do not use a data area with the data type BOOL or Array of BOOL at the DATA parameter. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
- The send job is executed when a rising edge is detected at the REQ parameter. With the LEN parameter, you specify the maximum number of bytes sent with a send job. When sending data (rising edge at the REQ parameter), the CONT parameter must have the value "1" in order to establish or maintain a connection. The data to be sent must not be edited until the send job is completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter is not confirmation that the data sent has already been read by the communication partner.
- Terminating the communication connection:
  The communication connection is terminated when the CONT parameter is set to "0" even if an ongoing data transfer is not complete yet. This does not apply if you are using an already configured connection for "TSEND_C".

When setting the COM_RST parameter to "1", the currently established connection or a current data transmission can be reset at any time. This terminates the existing communication connection and a new connection is established. If data is being transferred when it executes again, this can lead to a loss of data.

To enable "TSEND_C" again after the execution (DONE = 1), call the instruction once with REQ = 0.

Note

Support when programming connections

If you select an instruction for communication TCON, TSEND_C or TRCV_C in a program block and create connections of the type TCP, UDP or ISO-on-TCP and want to assign parameters to them, you can use the support of the connection parameter assignment.

You can find the connection parameter assignment in the Inspector window of the program editor.

Parameters

The following table shows the parameters of the "TSEND_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Starts the send job on a rising edge.
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection • 1: Establish and maintain the communication connection When sending data (rising edge at the REQ parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum number of bytes to be sent with the job. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: AUTOHOTSPOT
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the send area containing the address and the length of the data to be sent (maximum length: 8192 bytes). When transferring structures, the structures must be identical at the sending and receiving end.
COM_RST	InOut	BOOL	I, Q, M, D, L	Restarts the instruction: • 0: Irrelevant • 1: Complete restart of the instruction causing an existing connection to be terminated and a new connection to be established.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job executed without errors

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set if errors occur during execution of "TSEND_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Job completed without error.
0	0001	The connection establishment is complete.
0	0003	The connection termination is complete.
0	7000	No job processing active.
0	7001	• Start execution of the job • Establish connection • Wait for connection partner
0	7002	Data was sent.
0	7003	Connection is being terminated.
0	7004	Connection established and monitored, no job processing active.
1	80A0	Group error for error codes 80A1 and 80A2.

ERROR	STATUS* (W#16#...)	Description
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.
1	80A3	Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80AA	A connection is currently being established with the same connection ID by another block. Repeat the job with a new rising edge at the REQ parameter.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B3	Inconsistent parameter assignment: Group error for error codes 80A0 to 80A2, 80A4, 80B4 to 80B9.
1	80B4	When using the protocol variant ISO on TCP (connection_type = B#16#12) for the passive establishment of a connection (active_est = FALSE) one or both of the following conditions was violated: "local_tsap_id_len >= B#16#02" and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	8085	The LEN parameter is larger than the highest permitted value.
1	8086	The ID parameter within the CONNECT parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	The ID of the local device in the connection description does not correspond to the CPU.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The connection cannot be established because the firewalls on the connection path are not open for the required ports. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST= 1.

ERROR	STATUS* (W#16#...)	Description
1	80C6	Remote network error. Remote partner cannot be reached.
1	8722	Parameter CONNECT: The source area is invalid. The area does not exist in the DB.
1	873A	Parameter CONNECT: Access to the connection description is not possible (for example, DB does not exist).
1	877F	Parameter CONNECT: Internal error.
1	8822	Parameter DATA: Invalid source area, the area does not exist in the DB.
1	8824	Parameter DATA: Area error in the VARIANT pointer.
1	8832	Parameter DATA: The DB number is too high.
1	883A	Parameter DATA: No access to the data area, for example because the data block does not exist
1	887F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	893A	Parameter ADDR: Access to send area not possible (e.g. because the DB does not exist).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Note

Error messages of the instructions "TCON", "TSEND", "T_DIAG" and "TDISCON"

Internally, the "TSEND_C" instruction uses the instructions "TCON (Page 3538)", "TSEND (Page 3553)", "T_DIAG (Page 3603)", "T_RESET (Page 3597)" and "TDISCON (Page 3548)". The error messages of these instructions can also be output at the STATUS parameter. The meaning of the error codes is described in the corresponding instructions. In the event of identical error codes for internally used instructions with different meanings, the instance data block of "TSEND_C" can be used to determine which instruction output the error.

See also

Difference between synchronous and asynchronous instructions (Page 596)

TSEND_C: Establishing a connection and sending data (S7-1200, S7-1500)

Validity

The following description of the "TSEND_C" instruction is valid for:

- Ethernet
  - CPU S7-1200 with firmware version ≥ V4.0 and CPU S7-1500
  - CPU S7-1500 firmware version V2.1 and higher: UDP multicast communication
  - CPU S7-1500 as of firmware version V2.0 and TSEND_C as of instruction version V3.2: Secure communication
- PROFIBUS
  FDL connections of the S7-1500 with CM 1542-5 as of V2.0 with the system data type TCON_FDL

Description

The "TSEND_C" instruction sets up and establishes a communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU.

The instruction is executed asynchronously and has the following functions:

- Setting up and establishing a communication connection
- Sending data via an existing communication connection
- Terminating or resetting the communication connection

Internally, the instruction "TSEND_C" uses the communication instructions "TCON", "TSEND", "T_DIAG", "T_RESET" and "TDISCON".

Note

Support when programming connections

If you select an instruction for communication TCON, TSEND_C or TRCV_C in a program block and wish to create and assign parameters to connections of the type TCP, UDP, ISO-on-TCP or FDL, you can use the support of the connection parameter assignment. UDP multicast connections via the integrated PROFINET interfaces are possible for S7-1500-CPUs as of firmware version V2.1 and higher.

You can find the connection parameter assignment in the Inspector window of the program editor.

Setting up and establishing a communication connection

The communication connection is set up and established with CONT=1. For information on the number of possible communication connections, refer to the technical specifications for your CPU. The connection description specified at the CONNECT parameter is used to set up the communication connection. The following connection types can be used:

- Programmed connections (structure of connection via "TCON"):
  - TCP / UDP: Connection description via the TCON_IP_v4 system data type
  - TCP / UDP with secure communication: Connection description via the system data types TCON_IP_V4_SEC or TCON_QDN_SEC
  - ISO-on-TCP: Connection description via the TCON_IP_RFC system data type
  - ISO: Connection description via the TCON_ISOnative system data type (for CP1543-1 only)
  - Telecontrol connections to SMS clients: Connection description via the TCON_PHONE system data type
    For this connection type, the station requires access to the mobile network via a mobile network CP.
  - FDL connections of the S7-1500 with CM 1542-5 as of V2.0 with the system data type TCON_FDL
- Configured connections
  - Specify an existing connection in the TCON_Configured system data type.

An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TSEND_C" again.

Sending data via an existing communication connection

The send job is executed when a rising edge is detected at the REQ parameter. As described above, the communication connection is established first.

You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. Do not use a data area with the data type BOOL or Array of BOOL at the DATA parameter. With the LEN parameter, you specify the maximum number of bytes sent with a send job. If you use a send area with optimized access at the DATA parameter, the LEN parameter must have the value "0".

The data to be sent must not be edited until the send job is completed.

Terminating and resetting the communication connection

The communication connection is terminated when the CONT parameter is set to "0" even if an ongoing data transfer is not complete yet. This does not apply if you are using a configured connection for "TSEND_C".

The connection can be reset at any time by setting the parameter COM_RST to "1". This terminates the existing communication connection and a new connection is established. If data is being transferred at this time, this can lead to data loss.

Parameters

The following table shows the parameters of the "TSEND_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	Starts the send job on a rising edge.
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection. • 1: Establish and maintain the communication connection.
LEN	Input	UDINT	I, Q, M, D, L or constant	Optional parameter (hidden) Maximum number of bytes to be sent with the job. If you use a send area with optimized access at the DATA parameter, the value "0" must be used at the LEN parameter. For FDL connections of the CM 1542-5, the maximum length is 240 bytes. In this regard, note the maximum lengths that can be processed by the connection partner.

Parameter	Declaration	Data type	Memory area	Description
CONNECT	InOut	VARIANT	D	Pointer to the structure of the connection description: • Programmed connection: – For TCP or UDP, use the TCON_IP_v4 system data type. For a description, refer toAUTOHOTSPOT. – For TCP or UDP with secure communication, use the structure TCON_IP_V4_SEC or TCON_QDN_SEC. For a description, refer to: Connection parameters in accordance with TCON_IP_V4_SEC or connection parameters in accordance with TCON_QDN_SEC – For ISO-on-TCP, use the TCON_IP_RFC system data type. For a description, refer to AUTOHOTSPOT. – For ISO, use the TCON_ISOnative system data type (for CP 1543-1 only). For description, refer to instruction "TCON (Page 3541)". – For connections to SMS clients, use the TCON_PHONE system data type. For a description, refer to AUTOHOTSPOT. – For FDL connections of the CM 1542-5, use the system data type TCON_FDL; see AUTOHOTSPOT. • Configured connection: – For existing connections, use the TCON_Configured system data type. For a description, see "System data type for configured connections" below.
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the send area containing the address and the length of the data to be sent. When transferring structures, the structures must be identical at the sending and receiving end.
ADDR	InOut	VARIANT	D	Hidden parameter that needs to be used, however, with UDP. In this case it contains a pointer to the system data type TADDR_Param. Store the address information of the recipient (IP address and port number) in a data block with the system data type TADDR_Param. See also: Structure of the address information of the remote partner with UDP (Page 3587)

Parameter	Declaration	Data type	Memory area	Description
COM_RST	InOut	BOOL	I, Q, M, D, L	Optional parameter (hidden) Resets the connection: • 0: Irrelevant • 1: The existing connection is reset. The COM_RST parameter is reset after evaluation by the "TSEND_C" instruction and should not, therefore, be interconnected statically.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Send job not yet started or still in progress. • 1: Send job executed without error. This state is only displayed for one cycle. The output parameter DONE is set if an intermediate step was completed successfully during processing (connection establishment, sending, connection termination) and if the execution of "TSEND_C" was completed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Send job not yet started or already completed. • 1: Send job not yet completed. A new send job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred during connection establishment, data transfer or connection termination. The output parameter ERROR can be set due to an error in the "TSEND_C" instruction or the communication instructions used internally.
STATUS	Output	WORD	I, Q, M, D, L	Status of instruction (see the "ERROR and STATUS parameters" description).

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

REQ, CONT and COM_RST parameters

The parameter CONT controls the connection establishment of the "TSEND_C" instruction regardless of the REQ parameter. The behavior of the CONT parameter partially depends on whether a programmed or a configured connection is used:

- With CONT = "0": No data is sent (regardless of whether a programmed or a configured connection is used).
- When changing CONT = "0" to "1":
  - With a programmed connection, it is established with "TCON".
  - With a configured connection, it is checked with "T_DIAG".

- With CONT = "1":
  - As long as no data is sent (REQ="0"), the connection is checked with "T_DIAG".
  - If the internally used communication instructions signal that no connection end point exists, the connection is automatically reestablished with "TCON".
- When changing CONT = "1" to "0":
  - With a programmed connection, it is terminated with "TDISCON".
  - With a configured connection, it is reset with "T_RESET".

The parameter COM_RST resets the connection when changing from "0" to "1":

- If a connection is established, it is reset with "T_RESET" (regardless of whether a programmed or configured connection is used).
- If no connection is established, the setting of the parameter has no effect.

The REQ and COM_RST parameters only have an effect if CONT has been set to "1". The following table shows the relationship between the REQ, CONT and COM_RST parameters:

REQ	CONT	COM_RST	Status of the instruction	Description
Irrelevant	0	Irrelevant	Not yet executed	No job active (STATUS = 7000).
Irrelevant	0	Irrelevant	Initialization	Connection is being terminated. The instruction is being reset.
Irrelevant	0 > 1	Irrelevant	Connection establishment	Connection is being established. Data is not being transferred yet.
0	1	0	Connection established	The connection is established and is monitored with the instruction "T_DIAG".
Irrelevant	1	0 > 1	Connection established	The connection is interrupted by "T_RESET" briefly and reset.
0 > 1	1	0	Connection established	Instruction starts sending.
Irrelevant	1	0 > 1	Data is being sent	Data transfer is interrupted. The connection is being reset.

System data type for configured connections

For configured connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured:

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to the connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	-	Connection type Select 254 (decimal) for a configured connection.

Note

For reasons of compatibility, the parameters InterfaceID and ConnectionType are part of the structure for the connection description to TCON_Configured. These parameters do not have an effect on the connection parameter assignment; only the parameter ID for the connection ID is evaluated in the connection parameter assignment.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a send job executed successfully. The ERROR parameter is set if errors occur during execution of "TSEND_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

DONE	BUSY	ERROR	Description
0	0	0	The instruction has not been executed yet (no rising edge at REQ parameter).
0	1	0	The instruction is being executed and calls the internally used communication instructions.
1	0	0	The send job was completed successfully. "0000" is output at the STATUS parameter. DONE = "1" is only displayed for one cycle.
0	0	1	The execution of the instruction or an intermediate step during processing was terminated with an error. If there is a subsequent error due to an internally used communication instruction, the error that occurred first during processing is displayed. This state is only displayed for one cycle.

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Send job was executed without error.
0	0001	Communication connection established.
0	0003	Communication connection closed.
0	7000	No active send job execution; no communication connection established.
0	7001	Initial call for establishing a connection.
0	7002	Second call for establishing a connection
0	7003	Communication connection is being terminated.
0	7004	Communication connection has been established and is being monitored. No send job execution active.
0	7005	Data transfer in progress.

ERROR	STATUS* (W#16#...)	Description
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A3	The nested "T_DIAG" instruction has reported that the connection has closed.
1	80A4	IP address of the remote endpoint of the connection is invalid or it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80AA	A connection is currently being established with the same connection ID by another block. Repeat the job with a new rising edge at the REQ parameter.
1	80B3	• When using the protocol variant UDP the ADDR parameter does not contain any data. • Error in the connection description • The local port is already being used in a different connection description.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	• For system data type TCON_Param: Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len. • For system data types TCON_IP_V4 and TCON_IP_RFC: IP address of the partner end point was set to 0.0.0.0.
1	8085	The LEN parameter is larger than the highest permitted value.
1	8086	The ID parameter within the CONNECT parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	• The CONNECT parameter does not point to a data block. • The CONNECT parameter does not point to a connection description. • The manually created connection description has an incorrect structure for the selected connection type.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.

ERROR	STATUS* (W#16#...)	Description
1	809B	InterfaceID is invalid: • It does not point to a local CPU interface or a CP. • If you are using the connection parameter assignment, it cannot have the value 0. • It must not have the value 0 in the used TCON_xxx structure. See TCON: Establish communication connection (Page 3541)
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The connection cannot be established because the firewalls on the connection path are not open for the required ports. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON (Page 3548)" instruction. • The connection used is being terminated by a call with COM_RST = 1. • Temporarily no receive resources available at the connection partner. The connection partner is not ready to receive.
1	80C5	• Connection terminated by the communication partner. • LSAP of the remote connection partner is not released
1	80C6	Network error: • Remote partner cannot be reached. • Physical interruption on PROFIBUS
1	8722	Parameter CONNECT: The source area is invalid. The area does not exist in the DB.
1	873A	Parameter CONNECT: Access to the connection description is not possible (for example, because the DB is not available).
1	877F	Parameter CONNECT: Internal error.
1	8822	Parameter DATA: Invalid source area, the area does not exist in the DB.
1	8824	Parameter DATA: Area error in the VARIANT pointer.
1	8832	Parameter DATA: The DB number is too high.
1	883A	Parameter DATA: No access to the data area, for example because the data block does not exist
1	887F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	893A	Parameter ADDR: Access to send area not possible (e.g. because the DB does not exist).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Note

Error messages of the instructions "TCON", "TSEND", "T_DIAG", "T_RESET" and "TDISCON"

Internally, the "TSEND_C" instruction uses the instructions "TCON", "TSEND (Page 3553)", "T_DIAG (Page 3603)", "T_RESET (Page 3597)" and "TDISCON (Page 3548)". The error messages of these instructions can also be output at the STATUS parameter. The meaning of the error codes is described in the corresponding instructions. In the event of identical error codes for internally used instructions with different meanings, the instance data block of "TSEND_C" can be used to determine which instruction output the error.

Example

You can find the example here: Program example for send functions (Page 3506).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

TRCV_C: Establishing a connection and receiving data

TRCV_C: Receive data via Ethernet (S7-1200)

Validity

The following description of the "TRCV_C" instruction applies to S7-1200 CPUs with firmware version < V4.0.

Description

The "TRCV_C" instruction is executed asynchronously and has the following functions:

1. **Setting up and establishing a communication connection:**
   "TRCV_C" sets up and establishes a TCP or ISO-on-TCP communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU.
   The connection description specified at the CONNECT parameter is used to set up the communication connection. To establish a connection, the CONT parameter must be set to the value "1". Once the connection is successfully established, the DONE parameter is set to "1".
   An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TRCV_C" again.
   For information on the number of possible communication connections, refer to the technical specifications for your CPU.
2. **Receiving data via an existing communication connection:**
   If the EN_R parameter is set to the value "1", receipt of data is enabled. When receiving data (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
   The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0) in accordance with the protocol variant used. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
   After data has been received successfully, the signal state at the DONE parameter is "1".
   If errors occur in the data transfer, the DONE parameter is set to "0".
3. **Terminating the communication connection:**
   The communication connection is terminated immediately when the CONT parameter is set to "0".

TRCV_C is executed again when the COM_RST parameter is set. This terminates the existing communication connection and a new connection is established. If data is being received when it executes again, this can lead to a loss of data.

Note

Support when programming connections

If you select an instruction for communication TCON, TSEND_C or TRCV_C in a program block and create connections of the type TCP, UDP or ISO-on-TCP and want to assign parameters to them, you can use the support of the connection parameter assignment.

You can find the connection parameter assignment in the Inspector window of the program editor.

Receive modes of TRCV_C

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type parameter of the connection description	Parameter LEN
TCP (Ad-hoc mode)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	0
TCP (Receipt of data with specified length)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192
ISO on TCP (Message-oriented data transfer)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	• 1 to 1452, if a CP is used. • 1 to 8192, if no CP is used.

TCP (Ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV" instruction.

You set ad-hoc mode by assigning the value "0" to the LEN parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). The data length actually received is output at the RCVD_LEN parameter.

TCP (Receipt of data with specified length)

You use the value of the LEN parameter to specify the length for the data receipt. The data receipt is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

ISO on TCP (Message-oriented data transfer)

Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. When using ISO on TCP, "TRCV_C" signals data receipt as soon as the message block has been completely received. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV_C" signals an error. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

Parameters

The following table shows the parameters of the "TRCV_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Receive enable
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection • 1: Establish and maintain the communication connection When receiving data (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the data to be received (maximum value: 8192 bytes). If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: AUTOHOTSPOT
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area When transferring structures, the structures must be identical at the sending and receiving end.
COM_RST	InOut	BOOL	I, Q, M, D, L	Restarts the instruction: • 0: Irrelevant • 1: Complete restart of the instruction causing an existing connection to be terminated
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set if errors occur during execution of "TRCV_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Job completed without error.
0	0001	The connection establishment is complete.
0	0003	The connection termination is complete.
0	7000	No job processing active.
0	7001	• Start execution of the job • Establish connection • Wait for connection partner
0	7002	Data is being received.
0	7003	Connection is being terminated.
0	7004	• Connection established and monitored • No job processing active
0	7006	Data is currently being received.
1	8085	• The LEN parameter is larger than the highest permitted value. • The value at the LEN or DATA parameter was changed after the first call.
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	The ID of the local device (local_device_id) in the connection description does not correspond to the CPU.
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.

ERROR	STATUS* (W#16#...)	Description
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.
1	80A3	• Attempt being made to re-establish an existing connection. • Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B3	Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9.
1	80B4	When using the protocol variant ISO on TCP (connection_type = B#16#12) for the passive establishment of a connection (active_est = FALSE) one or both of the following conditions was violated: "local_tsap_id_len >= B#16#02" and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The connection cannot be established because the firewalls on the connection path are not open for the required ports. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST= 1.
1	80C6	The remote partner cannot be reached (network error).
1	8722	Error in the CONNECT parameter: Invalid source area (area not declared in data block).
1	873A	Error in the CONNECT parameter: Access to connection description is not possible (no access to data block).
1	877F	Error in the CONNECT parameter: Internal error
1	8922	Parameter DATA: Invalid destination area; the area does not exist in the DB.
1	8924	Parameter DATA: Area error in the VARIANT pointer.
1	8932	Parameter DATA: The DB number is too high.
1	893A	Parameter DATA: No access to the data area, for example because the data block does not exist.

ERROR	STATUS* (W#16#...)	Description
1	897F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	8A3A	Parameter ADDR: No access to the address range, for example because the data block does not exist.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Note

Error messages of the instructions "TCON", "TRCV" and "TDISCON"

Internally, the TRV_C instruction uses the "TCON (Page 3538)", "TRCV (Page 3557)" and "TDISCON (Page 3548)" instructions. The error messages of these instructions are contained in the respective descriptions.

See also

Difference between synchronous and asynchronous instructions (Page 596)

TRCV_C: Establishing a connection and receiving data (S7-1200, S7-1500)

Validity

The following description of the "TRCV_C" instruction is valid for:

- Ethernet:
  - CPU S7-1200 with firmware version V4.0 or higher and CPU S7-1500
  - CPU S7-1500 firmware version V2.1 and higher: UDP multicast communication
  - CPU S7-1500 as of firmware version V2.0 and TRCV_C as of instruction version V3.2: Secure communication
- PROFIBUS: FDL connections of the S7-1500 with CM 1542-5 as of V2.0 with the system data type TCON_FDL

Description

The "TRCV_C" instruction is executed asynchronously and implements the following functions in sequence:

- Setting up and establishing a communication connection
- Receiving data via an existing communication connection
- Terminating or resetting the communication connection

Internally, the instruction "TRCV_C" uses the communication instructions "TCON", "TRCV", "T_DIAG", "T_RESET" and "TDISCON".

Note

Support when programming connections

If you select an instruction for communication TCON, TSEND_C or TRCV_C in a program block and wish to create and assign parameters to connections of the type TCP, UDP, ISO-on-TCP or FDL, you can use the support of the connection parameter assignment. UDP multicast connections via the integrated PROFINET interfaces are possible for S7-1500-CPUs as of firmware version V2.1 and higher.

You can find the connection parameter assignment in the Inspector window of the program editor.

Setting up and establishing a communication connection

The communication connection is set up and established with CONT=1. For information on the number of possible communication connections, refer to the technical specifications for your CPU. The connection description specified at the CONNECT parameter is used to set up the communication connection. The following connection types can be used:

- Programmed connections (structure of connection via "TCON"):
  - TCP / UDP: Connection description via the TCON_IP_v4 system data type
  - TCP / UDP with secure communication: Connection description via the system data types TCON_IP_V4_SEC or TCON_QDN_SEC
  - ISO-on-TCP: Connection description via the TCON_IP_RFC system data type
  - ISO: Connection description via the TCON_ISOnative system data type (for CP1543-1 only)
  - Telecontrol connections to SMS clients: Connection description via the TCON_PHONE system data type
    For this connection type, the station requires access to the mobile network via a mobile network CP.
  - FDL connections of the S7-1500 with CM 1542-5 as of V2.0 with the system data type TCON_FDL
- Configured connections
  - Specify an existing connection in the TCON_Configured system data type.

An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TRCV_C" again.

Receiving data via an existing communication connection

Receipt of data is enabled when the EN_R parameter is set to the value "1". The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0), depending on the protocol variant being used. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".

Receive modes of TRCV_C:

- **TCP (Ad-hoc mode)**
  The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV_C" instruction.
  You set ad-hoc mode by assigning the value "1" to the ADHOC parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). The data length actually received is output at the RCVD_LEN parameter.
- **TCP (Receipt of data with specified length)**
  Assign the value "0" to the ADHOC parameter for receipt of data with specified length. If ad-hoc mode is disabled, the data reception is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.
- **ISO-on-TCP (Message-oriented data transfer)**
  Complete message blocks are sent via a connection with the protocol variant ISO-on-TCP; these are recognized as such by the recipient. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV_C" signals an error. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type parameter of the connection description	Parameter LEN
TCP (Ad-hoc mode)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	1 up to maximum length (depending on the CPU)
TCP (Receipt of data with specified length)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192
ISO-on-TCP (Message-oriented data transfer)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	1 to 8192
FDL	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#15 Decimal: 21	1 to 240

Terminating the communication connection

The communication connection is terminated when the CONT parameter is set to "0" even if an ongoing data transfer is not complete yet. This does not apply if you are using a configured connection, however.

The connection can be reset at any time by setting the parameter COM_RST to "1". This terminates the existing communication connection and a new connection is established. If data is being transferred at this time, this can lead to data loss.

Parameters

The following table shows the parameters of the "TRCV_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L, T, C or constant	Receive enable
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection. • 1: Establish communication connection and maintain after receipt of data.
LEN	Input	UDINT	I, Q, M, D, L or constant	Maximum length of the data to be received. If you use a receive area with optimized access at the DATA parameter, the value "0" must be used at the LEN parameter.
ADHOC	Input	BOOL	I, Q, M, D, L or constant	Optional parameter (hidden) Use ad-hoc mode for the TCP protocol variant. ADHOC must have the value FALSE if no TCP protocol is used.

Parameter	Declaration	Data type	Memory area	Description
CONNECT	InOut	VARIANT	D	Pointer to the connection description • Programmed connection: – For TCP or UDP, use the structure TCON_IP_v4 For a description, refer to:AUTOHOTSPOT – For TCP or UDP with secure communication, use the structure TCON_IP_V4_SEC or TCON_QDN_SEC. For a description, refer to: Connection parameters in accordance with TCON_IP_V4_SEC or connection parameters in accordance with TCON_QDN_SEC – For ISO-on-TCP, use the structure TCON_IP_RFC For a description, refer to: AUTOHOTSPOT – For ISO, use the structure TCON_ISOnative (for CP 1543-1 only) For description, refer to instruction "TCON (Page 3541)": "Structure of the connection description according to TCON_ISOnative" – For connections to SMS clients, use the TCON_PHONE system data type. For a description, refer to AUTOHOTSPOT. – For FDL connections of the CM 1542-5, use the system data type TCON_FDL; see AUTOHOTSPOT. • Configured connection: – For existing connections, use the TCON_Configured system data type. For a description, see "System data type for configured connections" below.
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area. When transferring structures, the structures must be identical at the sending and receiving end.
ADDR	InOut	VARIANT	D	Hidden parameter that needs to be used, however, with UDP. In this case it contains a pointer to the system data type TADDR_Param. Store the address information of the sender (IP address and port number) in a data block with the system data type TADDR_Param. See also: Structure of the address information of the remote partner with UDP (Page 3587)

Parameter	Declaration	Data type	Memory area	Description
COM_RST	InOut	BOOL	I, Q, M, D, L	Optional parameter (hidden) Resets the connection: • 0: Irrelevant • 1: The existing connection is reset. The COM_RST parameter is reset after evaluation by the "TRCV_C" instruction and should not, therefore, be interconnected statically.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Receipt not yet started or still in progress. • 1: Receipt executed without error. This state is only displayed for one cycle. The output parameter DONE is set if an intermediate step was completed successfully during processing (connection establishment, receipt, connection termination) and if the execution of "TRCV_C" was completed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Receipt not yet started or already completed. • 1: Receipt not yet completed. A new send job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred during connection establishment, data reception or connection termination. The output parameter ERROR can be set due to an error in the "TRCV_C" instruction or the communication instructions used internally.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

EN_R, CONT and COM_RST parameters

The parameter CONT controls the connection establishment of the "TRCV_C" instruction regardless of the EN_R parameter. The behavior of the CONT parameter partially depends on whether a programmed or a configured connection is used:

- With CONT = "0": No data is received (regardless of whether a programmed or a configured connection is used).
- When changing CONT = "0" to "1":
  - With a programmed connection, it is established with "TCON".
  - With a configured connection, it is checked with "T_DIAG".

- With CONT = "1":
  - As long as no data is received (EN_R="0"), the connection is checked with "T_DIAG".
  - If the internally used communication instructions signal that no connection end point exists, the connection is automatically reestablished with "TCON".
- When changing CONT = "1" to "0":
  - With a programmed connection, it is terminated with "TDISCON".
  - With a configured connection, it is reset with "T_RESET".

The parameter COM_RST resets the connection when changing from "0" to "1":

- If a connection is established, it is reset with "T_RESET" (regardless of whether a programmed or configured connection is used).
- If no connection is established, the setting of the parameter has no effect.

The EN_R and COM_RST parameters only have an effect if CONT has been set to "1". The following table shows the relationship between the EN_R, CONT and COM_RST parameters:

EN_R	CONT	COM_RST	Status of the instruction	Description
Irrelevant	0	Irrelevant	Not yet executed	No job active (STATUS = 7000).
Irrelevant	0	Irrelevant	Initialization	Connection is being terminated. The instruction is being reset.
Irrelevant	0 > 1	Irrelevant	Connection establishment	Connection is being established. Data is not being transferred yet.
0	1	0	Connection established	The connection is established and is monitored with the instruction "T_DIAG".
Irrelevant	1	0 > 1	Connection established	The connection is interrupted by "T_RESET" briefly and reset.
0 > 1	1	0	Connection established	Instruction starts receiving.
Irrelevant	1	0 > 1	Data is being received	Data transfer is interrupted. The connection is being reset.

System data type for configured connections

For configured connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured:

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to the connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	-	Connection type Select 254 (decimal) for a configured connection.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set if errors occur during execution of "TRCV_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

DONE	BUSY	ERROR	Description
0	0	0	The instruction has not been executed yet (no rising edge at EN_R parameter).
0	1	0	The instruction is being executed and calls the internally used communication instructions.
1	0	0	The receipt has been completed successfully. "0000" is output at the parameter STATUS. DONE = "1" is only displayed for one cycle.
0	0	1	The execution of the instruction or an intermediate step during processing was terminated with an error. If there is a subsequent error due to an internally used communication instruction, the error that occurred first during processing is displayed. This state is only displayed for one cycle.

ERROR and STATUS parameters

ERROR	STATUS (W#16#...)	Description
0	0000	Receive job executed without error.
0	0001	Communication connection established.
0	0003	Communication connection closed.
0	7000	No job processing active.
0	7001	Initial call for establishing a connection.
0	7002	Second call for establishing a connection.
0	7003	Communication connection is being terminated.
0	7004	Communication connection has been established and is being monitored. No receive job processing active.
0	7006	Data is currently being received.
1	8085	• The LEN parameter is larger than the highest permitted value. • The value at the LEN or DATA parameter was changed after the first call.
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	• The CONNECT parameter does not point to a data block. • The CONNECT parameter does not point to a connection description. • The manually created connection description has an incorrect structure for the selected connection type.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.

ERROR	STATUS (W#16#...)	Description
1	809B	InterfaceID is invalid: • It does not point to a local CPU interface or a CP. • If you are using the connection parameter assignment, it cannot have the value 0. • It must not have the value 0 in the used TCON_xxx structure. See TCON: Establish communication connection (Page 3541)
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A3	The nested "T_DIAG" instruction has reported that the connection has closed.
1	80A4	IP address of the remote endpoint of the connection is invalid or it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80AA	A connection is currently being established with the same connection ID by another block. Repeat the job with a new rising edge at the REQ parameter.
1	80B3	• When using the protocol variant UDP the ADDR parameter does not contain any data. • Error in the connection description • The local port is already being used in a different connection description.
1	80B4	When using the protocol variant ISO on TCP (connection_type = B#16#12) for the passive establishment of a connection (active_est = FALSE) one or both of the following conditions was violated: "local_tsap_id_len >= B#16#02" and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	• For system data type TCON_Param: Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len. • For system data types TCON_IP_V4 and TCON_IP_RFC: IP address of the partner end point was set to 0.0.0.0.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The connection cannot be established because the firewalls on the connection path are not open for the required ports. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST = 1.
1	80C6	The remote partner cannot be reached (network error).

ERROR	STATUS (W#16#...)	Description
1	8722	Error in the CONNECT parameter: Invalid source area (area not declared in data block).
1	873A	Error in the CONNECT parameter: Access to connection description is not possible (no access to data block).
1	877F	Error in the CONNECT parameter: Internal error
1	8922	Parameter DATA: Invalid destination area; the area does not exist in the DB.
1	8924	Parameter DATA: Area error in the VARIANT pointer.
1	8932	Parameter DATA: The DB number is too high.
1	893A	Parameter DATA: No access to the data area, for example because the data block does not exist.
1	897F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	8A3A	Parameter ADDR: No access to the address range, for example because the data block does not exist.

Note

Error messages of the instructions "TCON", "TRCV" and "TDISCON"

Internally, the TRV_C instruction uses the "TCON (Page 3541)", "TRCV (Page 3557)" and "TDISCON (Page 3548)" instructions. The error messages of these instructions are contained in the respective descriptions.

Example

You can find the example here: Program example for send functions (Page 3506).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

TSEND_C: Establishing a connection and sending data (Page 3482)

Difference between synchronous and asynchronous instructions (Page 596)

Program example for send functions

Introduction

In the following example, you create a programmed connection between two CPUs of the S7-1500 series and send a character string from CPU 1 to CPU 2. The character string to be sent is of the data type STRING.

Requirement

- Two CPUs of the S7-1500 series are available and connected to each other over PROFINET. Their connection is not yet configured.

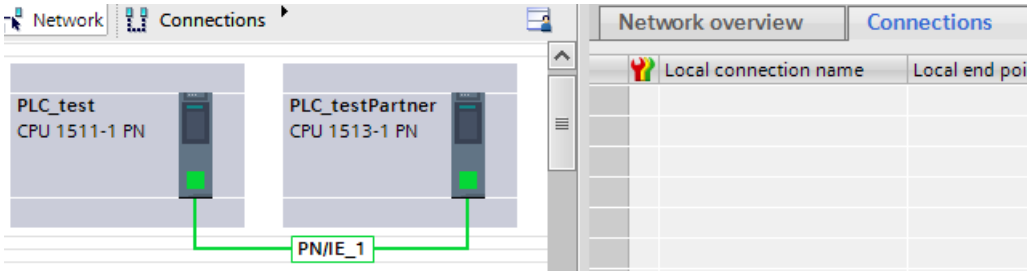

- A low protection level under Properties > Protection ensures that read and write access is permitted for each CPU.

Program of CPU 1

Create nine tags in a global data block for storing the data of "TSEND_C".

SLI_gDB_TSEND_C

		Name	Data type	Start value
1		Static		
2		start	Bool	false
3		comControl	Bool	false
4		done	Bool	false
5		busy	Bool	false
6		error	Bool	false
7		status	Word	16#0
8		memErrStatus	Word	16#0
9		sendData	String	'ProcessXYZ'

Create an FB "SLI_FB_TSEND_C" and create the following local tags in the FB.

7		Static			
8		TSEND_C_Instance	TSEND_C		
9		done	Bool	false	Non-retain

Network 1: Interconnect the parameters of the "TSEND_C" instruction as follows. Do not interconnect the parameter CONNECT yet.

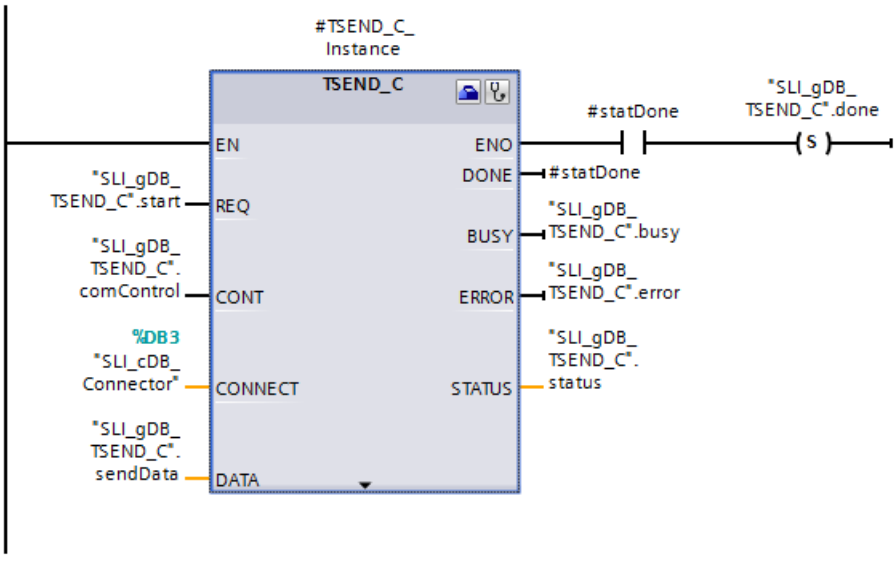

Network 2: Save the status for an error of TSEND_C as follows.

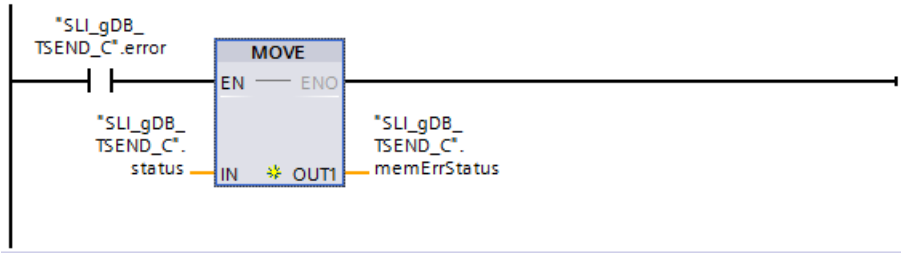

Configuration of TSEND_C

To interconnect the parameter CONNECT, open the wizard of the "TSEND_C" instruction with Properties > Configuration.

Make the following settings for the configuration of TSEND_C:

Input field	Entry
End point	Select a CPU for sender and receiver using the drop-down list. Interface, subnet and address are entered automatically.
Connection data	Use the drop-down list and the selection "New" to create one data block per CPU. This data block is required for storing the connection data. The name of the data block can be freely selected. The active connection establishment is enabled for CPU 1 (local CPU).
Connection type	Select the entry "TCP". This means an Ethernet connection with the "TCP" protocol is used for connection establishment.
Configuration mode	Select the entry "Use program block". This means connection establishment takes place by means of a programmed connection.
Connection ID	You enter a freely selected connection ID for the communication connection. The connection ID must not already be assigned in the project.
Partner port	You enter a value for the partner port. The value must be >=2000.

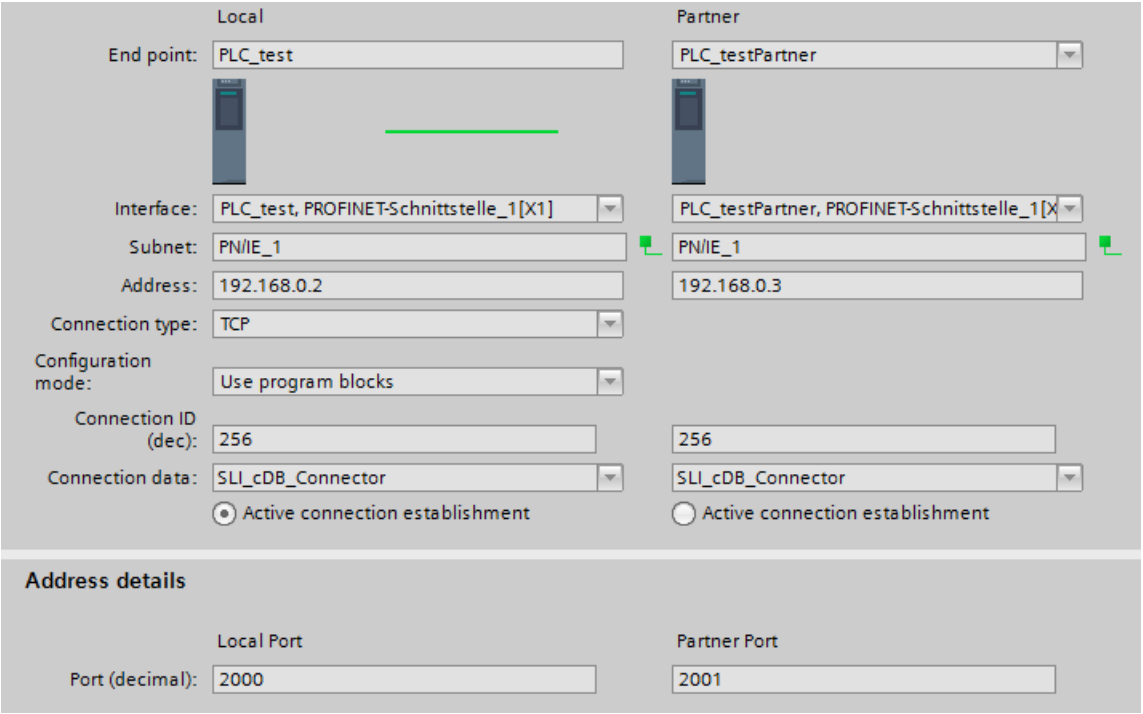

Program of CPU 2

Create ten tags in a global data block for storing the data of "TRCV_C".

SLI_gDB_TRCV_C

	Name	Data type	Start value
1	Static		
2	start	Bool	false
3	comControl	Bool	false
4	done	Bool	false
5	busy	Bool	false
6	error	Bool	false
7	status	Word	16#0
8	memErrStatus	Word	16#0
9	rcvLen	UDInt	0
10	rcvData	String	''

Create an FB "SLI_FB_TRCV_C" and create the following local tags in the FB.

7	Static			
8	TRCV_C_Instance	TRCV_C		
9	statDone	Bool	false	Non-retain
10	statRcvLen	UDInt	0	Non-retain

Network 1: Interconnect the parameters of the "TRCV_C" instruction as follows. Do not interconnect the parameter CONNECT yet.

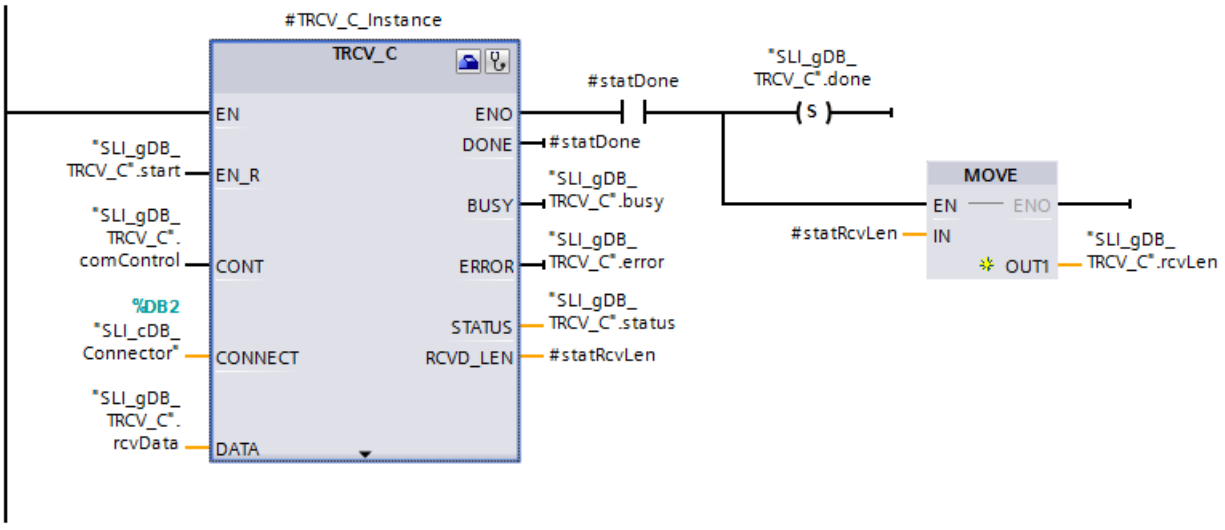

Network 2: Save the status for an error of TRCV_C as follows.

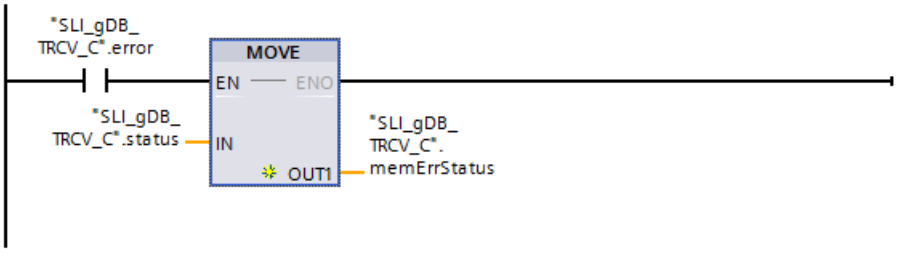

Configuration of TRCV_C

Make the following settings for the configuration of TRCV_C:

Note

Connection data

By configuring TSEND_C, you have already created a data block ("SLI_cDB_Connector") with the stored connection data for each CPU. Instead of creating new data blocks and entering the connection data, you can simply use the created data blocks ("SLI_cDB_Connector").

For the created data blocks to be available for selection, they must still be available in the project tree under Program blocks > System blocks.

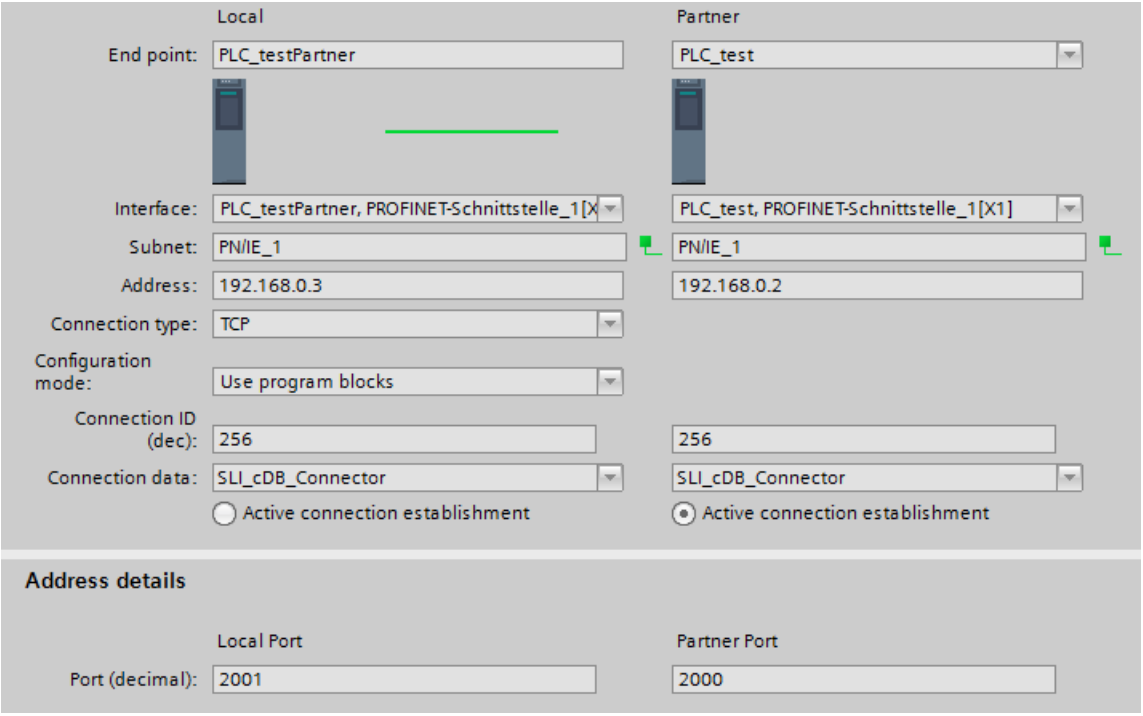

Procedure for connection establishment

To establish the communication connection in RUN mode, follow these steps:

1. Change the EN_R parameter to "1" for TRCV_C.
2. Change the CONT parameter to "1" for TRCV_C.
3. Change the REQ parameter to "1" for TSEND_C.
4. Change the CONT parameter to "1" for TSEND_C.

Procedure for connection termination

To terminate the communication connection in RUN mode, follow these steps:

1. Change the CONT parameter to "0" for TSEND_C.
2. Change the REQ parameter to "0" for TSEND_C.
3. Change the CONT parameter to "0" for TRCV_C.
4. Change the EN_R parameter to "0" for TRCV_C.

Behavior of CPU 1

When the input parameter REQ ("start") returns the signal state "TRUE", the "TSEND_C" instruction is started. When the input parameter CONT ("comControl") supplies the signal state "TRUE", the "TSEND_C" instruction creates a communication connection between CPU 1 and CPU 2. To do this, the connection data is retrieved with the input parameter CONNECT (or the data block "SLI_cDB_Connector").

For multiple calls, the "TSEND_C" instruction transmits the data record detected at input parameter DATA ("sendData"). Successful transfer of the data record is indicated by the output parameter DONE ("#statDone") with the signal state "TRUE" and the output parameter STATUS ("status") with the value "0000". Because the values of the output parameters are only displayed for the moment of their validity, save the success status of DONE ("#statDone") in the tag "done".

After transmitting the data record to CPU 2, the communication connection continues to be monitored (status "7004"). The output parameter ERROR ("error") or the tag "memErrStatus" indicates that processing in the example is taking place without errors.

SLI_gDB_TSEND_C

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	comControl	Bool	false	TRUE
4	done	Bool	false	TRUE
5	busy	Bool	false	FALSE
6	error	Bool	false	FALSE
7	status	Word	16#0	16#7004
8	memErrStatus	Word	16#0	16#0000
9	sendData	String	'ProcessXYZ'	'ProcessXYZ'

Behavior of CPU 2

When the input parameter EN_R ("start") returns the signal state "TRUE", the "TRCV_C" instruction is started. When the input parameter CONT ("comControl") supplies the signal state "TRUE", the "TRCV_C" instruction creates a communication connection between CPU 2 and CPU 1. To do this, the connection data is retrieved with the input parameter CONNECT (or the data block "SLI_cDB_Connector").

For multiple calls, the "TRCV_C" instruction receives the transmitted data record. The data record is recorded at the parameter DATA ("rcvData"). Successful transfer of the data record is indicated by the output parameter DONE ("done") with the signal state "TRUE" and the output parameter STATUS ("status") with the value "0000". The length in BYTE of the actually transmitted data record is detected with the output parameter RCVD_LEN ("#statRcvLen"). This value is only displayed during the success status. Then "0" is detected.

Because the values of the output parameters are only displayed for the moment of their validity, do the following:

- Save the value of "#statRcvLen" in "rcvLen".
- Save the value of "#statDone" in "done".
- Save the value of "#status" in "memErrStatus".

After the data record has been received by CPU 2, the communication connection continues to be monitored (status "7006"). The output parameter ERROR ("error") or the tag "memErrStatus" indicates that processing of the example is taking place without errors.

SLI_gDB_TRCV_C

	Name	Data type	Start value	Monitor value
1	Static			
2	start	Bool	false	TRUE
3	comControl	Bool	false	TRUE
4	done	Bool	false	TRUE
5	busy	Bool	false	TRUE
6	error	Bool	false	FALSE
7	status	Word	16#0	16#7006
8	memErrStatus	Word	16#0	16#0000
9	rcvLen	UDInt	0	256
10	rcvData	String	''	'ProcessXYZ'

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

TMAIL_C: Transfer email

Description of TMAIL_C (S7-1200, S7-1500)

Description

You can use the "TMAIL_C" instruction to send an e-mail via the Ethernet interface of the S7-1500 CPU or S7-1200 > V4.0, a communication module (CM), or a communication processor (CP).

The instruction can only be used once the hardware has been configured and if the network infrastructure allows for a communication connection to the mail server.

You define the content of the e-mail, and the connection data, using the following parameters:

- You define the recipient addresses with the parameters TO_S and CC.
- You define the content of the e-mail with the parameters SUBJECT and TEXT.
- You can define an attachment using VARIANT pointers at the ATTACHMENT and ATTACHMENT_NAME parameters.
- The connection data is defined, and addressing and authentication for the mail server executed, using the system data type TMail_V4, TMail_V6 or TMail_FQDN at the MAIL_ADDR_PARAM parameter.
  - If you are using the interface of the S7-1500 CPU, the system data type TMail_V4 must be used. In this case, the e-mail can only be sent via SMTP.
  - Any of the system data types can be used if you are using the interface of a CM/CP. The e-mail can then also be sent via SMTPS.

- You start the sending of an e-mail with an edge change from "0" to "1" for the REQ parameter.
- The job status is indicated by the output parameters "BUSY", "DONE", "ERROR" and "STATUS".

You cannot send an SMS directly with the "TMAIL_C" instruction. Whether or not the e-mail can be forwarded by the mail server as an SMS depends on your telecommunications provider.

Note

Number of e-mails to be sent

You can send more than one e-mail simultaneously using a PLC. When a CP 1243-8 or CP 1543-1 is used, you can only send one e-mail for each CP. When two CPs are used, parallel sending of two e-mails is possible.

Operation of the instruction

The "TMAIL_C" instruction works asynchronously, which means its execution extends over multiple calls. You must specify an instance when you call the instruction "TMAIL_C".

In the following cases, the connection to the mail server will be lost:

- If the CPU switches to STOP while "TMAIL_C" is active.
- If communication problems occur at the Industrial Ethernet bus.

In this case, the transfer of the e-mail will be interrupted and it will not reach its recipient. The connection is also canceled once the instruction has been successfully executed and the e-mail sent.

NOTICE

Changing user programs

You can change the parts of your user program that directly affect calls of "TMAIL_C" only:

- The CPU is in "STOP" mode.
- No e-mail is being sent (REQ = 0 and BUSY = 0).

This relates, in particular, to deleting and replacing program blocks that contain "TMAIL_C" calls or calls for the instance of "TMAIL_C".

Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status with the TCP/IP communication functions via Industrial Ethernet.

A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The TO_S, CC, SUBJECT, TEXT, ATTACHMENT and MAIL_ADDR_PARAM parameters are applied by the "TMAIL_C" instruction while it is running, which means that they may only be changed after the job has been completed (BUSY = 0).

SMTP authentication

Authentication refers here to a procedure for verifying identity, for example, with a password query.

If you are using the S7-1500 CPU interface, the instruction "TMAIL_C" supports the SMTP authentication procedure AUTH-LOGIN which is required by most mail servers. For information about the authentication procedure of your mail server, please refer to your mail server manual or the website of your Internet service provider.

- Before you can use the AUTH-LOGIN authentication procedure, the "TMAIL_C" instruction requires the user name with which it is to log on to the mail server. This user name corresponds to the user name with which you set up a mail account on your mail server. It is transferred via the UserName parameter to the structure at parameter MAIL_ADDR_PARAM.
  If no user name is specified at the MAIL_ADDR_PARAM parameter, the AUTH-LOGIN authentication procedure is not used. The e-mail is then sent without authentication.
- To log on, the "TMAIL_C" instruction also requires the associated password. This password corresponds to the password you specified when you set up your mail account. It is transferred via the PassWord parameter to the structure at parameter MAIL_ADDR_PARAM.

Parameters

The following table shows the parameters of the "TMAIL_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	Control parameter REQUEST: Activates the sending of an e-mail upon a rising edge.
TO_S (Page 3521)	Input	STRING	D, L or constant	Recipient addresses STRING with a maximum length of 240 characters (bytes). For the e-mail address format, see the example in the parameter description.
CC (Page 3521)	Input	STRING	D, L or constant	CC recipient addresses (optional) STRING with a maximum length of 240 characters (bytes). Same e-mail address format as for the TO_S parameter. If an empty string is assigned here, the e-mail is not sent to a CC recipient.
SUBJECT	Input	STRING	D, L or constant	Subject of the e-mail STRING with a maximum length of 240 characters (bytes).
TEXT	Input	STRING	D, L or constant	Text of the e-mail (optional) STRING with a maximum length of 240 characters (bytes). If an empty string is assigned at this parameter, the e-mail is sent without text.

Parameter	Declaration	Data type	Memory area	Description
ATTACHMENT	Input	VARIANT	D	E-mail attachment (optional) Reference to a byte/word/double word field (ArrayOfByte, ArrayOfWord or ArrayOfDWord) with a maximum length of 64 Kb. If no value is assigned, the e-mail is sent without an attachment.
ATTACHMENT_NAME	Input	STRING	D, L or constant	E-mail attachment name (optional) Reference to a character string with a maximum length of 50 characters (bytes) to define the file name of the attachment. If an empty string is assigned at this parameter, the attachment of the e-mail is received with a file name assigned by the e-mail receiving program. It is therefore recommended to use a defined file name.
MAIL_ADDR_PARAM (Page 3522)	Input	VARIANT	D	Connection parameter and address of the e-mail server To define the connection parameters, use the structure TMail_V4, TMail_V6 or TMail_FQDN (see parameter description).
DONE (Page 3531)	Output	BOOL	I, Q, M, D, L	Status parameter • DONE = 0: Job not yet started or still executing. • DONE = 1: Job executed without errors.
BUSY (Page 3531)	Output	BOOL	I, Q, M, D, L	Status parameter • BUSY = 0: Processing of "TMAIL_C" was terminated • BUSY = 1: Sending of an e-mail is not yet complete.
ERROR (Page 3531)	Output	BOOL	I, Q, M, D, L	Status parameter • ERROR = 0: No error occurred. • ERROR = 1: An error occurred during processing. STATUS supplies detailed information on the type of error.
STATUS (Page 3532)	Output	WORD	I, Q, M, D, L	Status parameter Return value or error information of the "TMAIL_C" instruction (see parameter description).

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

Optional parameters

The optional parameters CC, TEXT, and ATTACHMENT are only sent with the e-mail if the corresponding parameters contain a string of length > 0.

Example

You will find an example of sending e-mails with the TMAIL_CExample: Sending an e-mail with TMAIL_C (Page 3535) instruction with the following link:

See also

Difference between synchronous and asynchronous instructions (Page 596)

Description TMAIL_C as of version V4.0 (Page 3517)

Description of TMAIL_C as of version V5.0 (Page 3519)

Description TMAIL_C as of version V4.0

Introduction

The instruction TMAIL_C as of version V4.0 is described below. This description does not include all details, but instead explains the differences as compared to the instruction versions < V4.0.

The handling of the required certificates is also not described here.

CPs using Secure Communication

The following CPs send secure emails via the instruction TMAIL_C as of version V4.0:

- CP 1543-1 - V2.0
- CP 1542SP-1 IRC - V1.0, CP 1543SP-1 - V1.0
- CP 1243-1 - V2.1
- CP 1242-7 GPRS V2 - V2.1
- CP 1243-7 LTE - V2.1
- CP 1243-8 IRC - V2.1

You set the data to be sent and the TCP port of the email server at the parameter MAIL_ADDR_PARAM of the following system data types:

TMAIL_V4_SEC, TMAIL_V6_SEC, TMAIL_QDN_SEC

Note

Compatibility with the TMAIL_C versions < V4.0

For reasons of compatibility, you can also use the existing system data types TMAIL_V4, TMAIL_V6 and TMAIL_FQDN as follows:

- CPUs:
  TMAIL_V4, TMAIL_FQDN
- All CPs mentioned above and CP 1542SP-1:
  TMAIL_V4, TMAIL_V6, TMAIL_FQDN

Rules for CPs

Observe the following rules:

- You may use only one CP as an email client for each TMAIL_C instance of a CPU.
- The email servers of the CPs (email clients) have to be accessible from an IP network.
- If an email server is to be reached via its fully qualified domain name, the DNS servers, which are configured in the CP, must also be reachable in the IP network.

Parameters

Parameter	Declaration	Data type	Memory area	Description
...				
MAIL_ADDR_PARAM (Page 3525)	Input	VARIANT	D	The data required for the send process (including the connection parameters, the address of the email server and the TCP port used) Select one of the following system data types for this: • TMail_V4_SEC • TMail_V6_SEC • TMail_QDN_SEC
...				

Diagnostic options when an error occurs

You have the following two diagnostic options when an error occurs:

- Evaluation of the STATUS parameter
  The same values can occur as with the instruction-versions < V4.0.
  Only the value W#16#8015 receives an extension. See STATUS parameter (Page 3532)
- Evaluation of the MS_STATUS tag in the instance data of TMAIL_C. This tag has been added in the version V4.0 of TMAIL_C. It contains both the protocol codes and their corresponding textual descriptions. The protocol codes are defined in RFC 5321 "Simple Mail Transfer Protocol". You can also find the associated textual descriptions there, in the corresponding expansions or on other media.
  Examples of diagnostic information in MS_STATUS:
  - 550 SMTP AUTH required before submission
  - 550 StartTLS required
  - 550 Requested action not taken

Description of TMAIL_C as of version V5.0

Introduction

The instruction TMAIL_C as of version V5.0 is described below. This description does not include all details, explaining only the differences as compared to the instruction versions < V5.0.

Description

As of version V5.0, you can use the instruction TMAIL_C to send an e-mail as well with Secure Communication over an integrated Ethernet port of your S7-1500 CPU. This is only possible if your S7-1500 CPU has at least firmware version V2.5. The data required for the sending process is defined with one of the system data types TMAIL_V4_SEC or TMAIL_QDN_SEC at the MAIL_ADDR_PARAM parameter.

If you use the system data type TMAIL_V4, e-mails can only be sent unsecured over an integrated Ethernet port of your S7-1500 CPU.

The system data types TMAIL_FQDN and TMAIL_V6 are not supported for sending e-mails over an integrated S7-1500 CPU Ethernet port.

Note

S7-1500 Software Controller

You cannot use the system data types TMAIL_V4_SEC or TMAIL_QDN_SEC with an S7-1500 Software Controller over ID=59.

STATUS parameter

With version V5.0 of TMAIL_C instructions, the following return value is also output at the STATUS parameter:

STATUS (W#16#...)	Explanation	Notes
8009	Internal function error	An internal function has returned an error. You can find detailed information in the SFB_STATUS parameter of the instance DB. Its possible values are described below.

The following table shows the possible values at the SFB_STATUS parameter of the instance DB:

Parameter SFB_STATUS of the instance DB (W#16#...)	Explanation
8090	WatchDog time must be greater than zero.
8092	The TO_S and CC parameters are empty or the From sub-parameter is empty or incomplete.
8093	The MAIL_ADDR_PARAM parameter requires the connection to be upgraded to a secure connection, but the mail server does not support the STARTTLS command.
8085	The connection ID (ID parameter) is already being used by a configured connection.

Parameter SFB_STATUS of the instance DB (W#16#...)	Explanation
8086	The ID parameter is outside the valid range.
8087	Maximum number of connections reached; no further connections possible
8095	Invalid response from mail server. The mail server may not be RFC-compliant.
809A	The SDT structure at the MAIL_ADDR_PARAM parameter is not supported at an integrated interface.
809B	Invalid interface ID in SDT at MAIL_ADDR_PARAM.
80A1	The specified connection or the remote port is already being used.
80A3	ID is used by a connection created by the user program.
80A4	IP address of the remote endpoint of the connection is invalid or it corresponds to the IP address of the local partner.
80A7	Communication error: You executed TDISCON before TMAIL_C had completed.
80B7	Remote port is 0 or IP address of the partner endpoint was set to 0.0.0.0.
80C3	Too few resources in the CPU
80C4	Temporary communication error: • The connection cannot be established at this time. • The connection cannot be established because the firewalls on the connection path are not open for the required ports. • The interface is currently receiving new parameters. • The configured connection is currently being removed by a TDISCON instruction.
80C5	The mail server refuses to establish the connection, has terminated the connection or is actively ending it.
80C6	The connection partner cannot be reached (network error).
80C7	Execution timeout
80C8	Attempt being made to re-establish an existing connection.
80C9	Validation of the connection partner failed. The mail server does not correspond to the defined partner at Parameter MailServerAddress.
80CE	The IP address of the local interface is 0.0.0.0.
80D0	The MailServerAddress parameter contains an empty string in the attempt to use DNS.
80D1	The MailServerAddress parameter is not a fully qualified domain name. The period at the end may be missing.
80D2	No DNS server address is configured.
80D3	The fully qualified domain name could not be resolved. Possible causes: • The DNS- server cannot be reached, for example because it has been shut down or the remote port cannot be reached. • An error occurred during communication with the DNS-server. • The DNS server returned a valid DNS response, but the response contained no IPv4 address.
80E0	Communication with the mail server failed due to a message with errors. Possible causes: • Invalid message authentication code • Message decoding failed. • Error decompressing a message • Internal capacity overflow

Parameter SFB_STATUS of the instance DB (W#16#...)	Explanation
80E1	Error during the handshake. Possible causes: • Abort by the user • Security not high enough • Renewed negotiation is not supported • SSL/TLS version is not supported. • Decryption error
80E2	Certificate not supported / invalid Possible cause: The time-of-day of the module concerned is not set or the module is not synchronized. Example: The default setting for the date of the module is 1/1/2012 and the date was not set during commissioning. The validity period of the certificate starts on 20 August 2016 and ends on 20 August 2024. In this case, the date of the module is outside the validity period of the certificate; the certificate is invalid for the module.
80E3	Mail server certificate has been discarded.
80E4	No valid certification authority found for the mail server certificate.
80E5	Mail server certificate expired.
80E6	Integrity errors in the Transport-Layer-Security protocol
80E7	Unsupported extension in mail server certificate
80E9	TLS server without server certificate is not supported

Expanded value range of connection parameters InterfaceID and ID

The following applies for the connection parameters InterfaceID and ID in the SDTs TMAIL_V4_SEC and TMAIL_QDN_SEC when you send e-mails over a secure connection via an integrated port in your S7-1500 CPU:

Parameter	Description
InterfaceID	Hardware identifier of the interface Value range: • 0 (new): The operating system selects a suitable integrated port itself. • Hardware identifier of the integrated port to be used.
ID	Reference to the connection: Value range: • 0 (new): The operating system selects a free connection ID from the internal range. • 1 to 4095: the connection ID to be used

TO_S and CC parameters (S7-1200, S7-1500)

Description

The TO_S and CC parameters are strings, for example, with the following content:

- <wenna@mydomain.com>, <ruby@mydomain.com>
- <admin@mydomain.com>, <judy@mydomain.com>

Note the following rules when entering the parameters:

- A space and an opening pointed bracket "<" must be entered before each address.
- A closing pointed bracket ">" must be entered after each address.
- A comma must be entered between the addresses in TO and CC.

For runtime and memory space reasons, the "TMAIL_C" instruction does not perform a syntax check of parameter TO_S or CC.

MAIL_ADDR_PARAM parameter

Description

At the MAIL_ADDR_PARAM parameter, you define the connection for sending the e-mail in the structure TMail_V4, TMail_V6 or Tmail_FQDN, and save the e-mail server address and login details.

The structure you use at the MAIL_ADDR_PARAM parameter depends on the format in which the e-mail server is to be addressed:

- TMail_V4: Addressing by IP address to IPv4.
- TMail_V6: Addressing by IP address to IPv6.
- TMail_FQDN: Addressing by fully qualified domain name (FQDN).

Which structure you can use depends on the interface addressed at the InterfaceId parameter:

- If you want to use the "TMAIL_C" instruction with the internal interface, the TMail_V4 structure must be used at the MAIL_ADDR_PARAM parameter.
- If you are using a communication processor (CP) or communication module (CM), you can use all three addressing options (IPv4, IPv6 and FQDN).

Table 4-70 **TMail_V4: Addressing the mail server by IP address (IPv4)**

Parameter	Data type	Description
TMail_V4	Struct	
InterfaceId	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#20 as the connection type for IPv4.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	• =0: SMTP used (**S**imple **M**ail **T**ransfer **P**rotocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1500 CPU. • ≠0: SMTPS used to secure the connection before it is established (with CPs/CMs).
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	IP_V4	IP address of the mail server. IPv4 in the following format: XXX.XXX.XXX.XXX (decimal). Example: 192.142.131.237.
UserName	STRING[254]	Mail server login name
PassWord	STRING[254]	Mail server password
From	EMAIL_ADDR	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymail-server.com".
LocalPartPlusAt-Sign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@"
FullQualifiedDo-mainName	STRING[254]	Fully Qualified Domain Name (abbreviated to FQDN) of the mail server. Example: "mymailserver.com"

Table 4-71 **TMail_V6: Addressing the mail server by IP address to IPv6**

Parameter	Data type	Description
TMail_V6	Struct	
InterfaceId	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#21 as the connection type for IPv6.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	• =0: SMTP used (**S**imple **M**ail **T**ransfer **P**rotocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1500 CPU. • ≠0: SMTPS used to secure the connection before it is established (with CPs/CMs). You use the CertIndex parameter to specify the certificate to be used (see "Project navigation" > "Global security settings" > "Certificate manager").
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	IP_V6	IP address of the mail server (IPv6) in the following format: XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX (hexadecimal). The address is divided into 8 blocks of 2 bytes each (16 bytes in total). Example: 2001:db8:1f11:08d3:290:27ff:0370:2093
UserName	STRING[254]	Mail server login name
PassWord	STRING[254]	Mail server password
From	EMAIL_ADDR	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@"
FullQualifiedDomainName	STRING[254]	Fully Qualified Domain Name (abbreviated to FQDN) of the mail server. Example: "mymailserver.com"

Table 4-72 **TMail_FQDN: Addressing the mail server via the FQDN**

Parameter	Data type	Description
TMail_FQDN	Struct	
InterfaceId	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#22 as the connection type for FQDN.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	• =0: SMTP used (**S**imple **M**ail **T**ransfer **P**rotocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1500 CPU. • ≠0: SMTPS used to secure the connection before it is established (with CPs/CMs). You use the CertIndex parameter to specify the certificate to be used (see "Project navigation" > "Global security settings" > "Certificate manager").
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	STRING[254]	FQDN (**F**ully **Q**ualified **D**omain **N**ame) of the mail server. The mail server is addressed using the fully qualified domain name. Example: "www.mymailserver.com."
UserName	STRING[254]	Mail server login name
PassWord	STRING[254]	Mail server password
From	Struct	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymail-server.com".
LocalPartPlusAt-Sign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@"
FullQualifiedDo-mainName	STRING[254]	Fully Qualified Domain Name (abbreviated to FQDN) of the mail server. Example: "mymailserver.com"

Parameter MAIL_ADDR_PARAM as of Version 4.0 of TMAIL_C

Description

You use the MAIL_ADDR_PARAM parameter to specify the data required for the send process, for example the connection parameters, the address of the email server and the TCP port used.

Depending on the format in which the e-mail server is to be addressed, you use one of the following structures at the MAIL_ADDR_PARAM parameter:

- TMail_V4_SEC: Addressing via the IP address in IPv4 format
- TMail_V6_SEC: Addressing via the IP address in IPv6 format
- TMail_QDN_SEC: Addressing via the fully qualified host name (FQDN)

TMail_V4_SEC: Addressing of the mail server via the IP address in IPv4 format

Parameter	Data type	Description
TMail_V4_SEC	Struct	
InterfaceId	LADDR	Hardware identifier of the Ethernet interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#20 as the connection type for IPv4.
ActiveEstablishment	BOOL	Active/passive connection establishment. Because the CP is always the SMTP client, this parameter must be set to "1".
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WatchDogTime parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	IP_V4	IP address of the mail server intrinsic in IPv4 format: XXX.XXX.XXX.XXX (decimal place). Example: 192.142.131.237.
UserName	STRING[254]	User name. This is required if users wants to access their email inbox in order to identify themselves as the owner of the email inbox to the email provider.
PassWord	STRING[254]	User password. This is required if users wants to access their email inbox in order to identify themselves as the owner of the email inbox to the email provider.
From	EMAIL_ADDR	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@"

Parameter	Data type	Description
FullQualifiedDomainName	STRING[254]	Fully Qualified Domain Name (abbreviated to FQDN) of the mail server. Example: "my-mailserver.com"
RemotePort	UINT	TCP port of the mail server
ActivateSecureConn	BOOL	• 1: Secure SMTP connection • 0: SMTP connection (non-secure). In this case, the following parameters are irrelevant.
ExtTLSCapabilities	BYTE	Not currently used
TLSServerCertRef	UDINT	Reference to the X.509 V3 (CA) certificate of the mail server, which is used by the TLS client to validate the authentication of the TLS server.

TMail_V6_SEC: Addressing of the e-mail server via the IP address in IPv6 format

Parameter	Data type	Description
TMail_V6_SEC	Struct	
InterfaceId	LADDR	Hardware identifier of the Ethernet interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#21 as the connection type for IPv6.
ActiveEstablishment	BOOL	Active/passive connection establishment. Because the CP is always the SMTP client, this parameter must be set to "1".
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WatchDogTime parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	IP_V6	IP address of the mail server in IPv6 format: XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX (hexadecimal). The address is divided into 8 blocks of 2 bytes each (16 bytes in total). Example: 2001:db8:1f11:08d3:290:27ff:0370:2093
UserName	STRING[254]	User name. This is required if users wants to access their e-mail inbox in order to identify themselves as the owner of the e-mail inbox to the e-mail provider.
PassWord	STRING[254]	User password. This is required if users wants to access their e-mail inbox in order to identify themselves as the owner of the e-mail inbox to the e-mail provider.
From	EMAIL_ADDR	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@"
FullQualifiedDomainName	STRING[254]	Fully Qualified Domain Name (abbreviated to FQDN) of the mail server. Example: "mymailserver.com"

Parameter	Data type	Description
RemotePort	UINT	TCP port of the mail server
ActivateSecureConn	BOOL	• 1: Secure SMTP connection • 0: SMTP connection (non-secure). In this case, the following parameters are irrelevant.
ExtTLSCapabilities	BYTE	Not currently used
TLSServerCertRef	UDINT	Reference to the X.509 V3 (CA) certificate of the mail server, which is used by the TLS client to validate the authentication of the TLS server.

TMail_QDN_SEC: Addressing of the mail server via FQDN

Parameter	Data type	Description
TMail_QDN_SEC	Struct	
InterfaceId	LADDR	Hardware identifier of the Ethernet interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#22 as the connection type for FQDN.
ActiveEstablishment	BOOL	Active/passive connection establishment. Because the S7-CP is always the SMTP client, this parameter must be set to "1".
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WatchDogTime parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerQDN	STRING[254]	FQDN (Fully Qualified Domain Name) of the mail server. The mail server is addressed using the fully qualified domain name, which must be end with "." Example: "www.mymailserver.com"
UserName	STRING[254]	User name This is required if users wants to access their email inbox in order to identify themselves as the owner of the email inbox to the email provider.
PassWord	STRING[254]	User password This is required if users wants to access their email inbox in order to identify themselves as the owner of the email inbox to the email provider.
From	EMAIL_ADDR	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@"
FullQualifiedDomainName	STRING[254]	Fully Qualified Domain Name (abbreviated to FQDN) of the mail server. Example: "mymailserver.com"

Parameter	Data type	Description
RemotePort	UINT	TCP port of the mail server
ActivateSecureConn	BOOL	• 1: Secure SMTP connection • 0: Unsecured SMTP connection In this case, the following parameters are irrelevant.
ExtTLSCapabilities	BYTE	Not currently used
TLSServerCertRef	UDINT	Reference to the X.509 V3 (CA) certificate of the mail server, which is used by the TLS client to validate the authentication of the TLS server.

DONE, BUSY and ERROR parameters

Description

The output parameters DONE, BUSY and ERROR are each displayed for only one cycle if the status of the BUSY output parameter changes from "1" to "0".

The following table shows the relationship between DONE, BUSY, and ERROR. Using this table, you can determine the current status of the instruction "TMAIL_C and when the sending of the e-mail is complete.

DONE	BUSY	ER-ROR	Description
0	1	0	The job is being processed.
1	0	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS (Page 3532) parameter.
0	0	0	The "TMAIL_C" instruction was not assigned a (new) job.

STATUS parameter (S7-1200, S7-1500)

Description

The following table shows the return values of "TMAIL_C" at the STATUS parameter:

Return value STATUS* (W#16#...):	Explanation	Notes
0000	The processing of "TMAIL_C" was completed without errors.	Error-free completion of "TMAIL_C" does not mean that the e-mail sent will necessarily arrive. Incorrectly entering the recipient addresses does not generate a status error of the "TMAIL_C" instruction. In this case, there is no guarantee that the e-mail will be sent to other recipients, even if these were entered correctly.
7001	"TMAIL_C" is active (BUSY = 1).	First call: Job triggered.
7002	"TMAIL_C" is active (BUSY = 1).	Intermediate call: Job already active.
8xxx	The processing of "TMAIL_C" was completed with an error code of the communication instructions called internally.	For detailed information, please refer to the STATUS parameter descriptions for the "TCON (Page 3541)", "TDISCON (Page 3548)", "TSEND (Page 3553)" and "TRCV (Page 3557)" communication instructions.
8010	Error during connection establishment	You can find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the "TCON (Page 3541)" instruction.
8011	Error sending the data	You can find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the "TSEND (Page 3553)" instruction.
8012	Error receiving the data	You can find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the "TRCV (Page 3557)" instruction.
8013	Error during connection establishment	You can find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the TCON (Page 3541)" and "TDISCON (Page 3548)" instructions.

Return value STATUS* (W#16#...):	Explanation	Notes
8014	Establishment of a connection is not possible.	You may have entered an incorrect mail server IP address (MailServerAddress (Page 3522)) or too short a time span (WatchDogTime (Page 3522)) for connection establishment. It is also possible that the CPU has no connection to the network or that the CPU configuration is incorrect.
8015	Incorrect data type for MAIL_ADDR_PARAM	The only valid data types are the system data types (structures) TMail_V4, TMail_V6 and TMail_FQDN. As of instruction version V4.0, the following system data types are also valid: TMail_V4_SEC, TMail_V6_SEC and TMail_QDN_SEC
8016	Incorrect data type for the ATTACHMENT parameter	The only valid data types are ArrayOfByte, ArrayOfWord and ArrayOfDWord.
8017	Incorrect data length for the ATTACHMENT parameter	Data length must be <= 65534 bytes.
8401	No channels available. Possible cause: There is already an e-mail connection via the CP. A second connection cannot be established in parallel.	Specific error for CP 1543
8403	A TCP/IP connection to the mail server was not possible.	Specific error for CP 1543
8405	Server has denied login request.	Specific error for CP 1543
8406	SMTP client has detected an internal SSL error or a problem with the certificate structure.	Specific error for CP 1543
8407	Request to use SSL denied.	Specific error for CP 1543
8408	Client unable to identify socket for establishing a TCP/IP connection to the mail server.	Specific error for CP 1543
8409	Writing via the connection not possible. Possible cause: The communication partner has reset the connection or the connection has been lost.	Specific error for CP 1543
8410	Reading via the connection not possible. Possible cause: The communication partner has reset the connection or the connection has been lost.	Specific error for CP 1543
8411	E-mail could not be sent. Cause: Insufficient memory space to execute the send operation.	Specific error for CP 1543
8412	Configured DNS server was unable to resolve the specified domain name.	Specific error for CP 1543
8413	An internal error in the DNS subsystem prevented domain name resolution.	Specific error for CP 1543

Return value STATUS* (W#16#...):	Explanation	Notes
8414	Empty character string entered as domain name.	Specific error for CP 1543
8415	An internal error has occurred in the cURL module. Execution was stopped.	Specific error for CP 1543
8416	An internal error has occurred in the SMTP module. Execution was stopped.	Specific error for CP 1543
8417	Request to SMTP on a channel already in use or invalid channel ID. Execution was stopped.	Specific error for CP 1543
8418	E-mail send job aborted. Possible causes: Execution time exceeded (WatchDogTime parameter) or CP transition from start to stop.	Specific error for CP 1543
8419	Channel interrupted and cannot be used before the connection is closed.	Specific error for CP 1543
8420	Server certificate string could not be verified with CP root certificate.	Specific error for CP 1543
8421	Internal error occurred. Execution was stopped.	Specific error for CP 1543
82xx, 84xx, or 85xx	The error message originates from the mail server and corresponds, except for the "8", to the error number of the SMTP protocol. The following lines list several error codes that can occur.	You can find more detailed information on the SMTP error code and other error codes in the SMTP protocol on the Internet or in the error documentation of the mail server. You can also view the most recent error message from the mail server in your instance DB in the BUFFER1 parameter. You can find the last data sent by the "TMAIL_C" instruction under DATEN in the instance DB.
8450	Action not executed: Mailbox not available/cannot be reached	Try again later.
8451	Action aborted: Local processing error	Try again later.
8500	Syntax error: Error not recognized. This also includes the error when a command string is too long. This can also occur when the e-mail server does not support the LOGIN authentication procedure.	Check the parameters of "TMAIL_C". Try to send an e-mail without authentication. To do this, replace the content of the UserName parameter with an empty string. If no user name is specified, the LOGIN authentication procedure is not used.
8501	Syntax error: Incorrect input at a parameter	Possible cause: Incorrect address at the parameters TO_S or CC (see also: TO_S and CC parameters (Page 3521)).
8502	Command unknown or not implemented	Check your entries, in particular the FROM parameter. It may be incomplete and you may have forgotten the "@" or "." (see also: TO_S and CC parameters (Page 3521)).
8504	Command parameters not implemented	Check your entries, in particular the FROM parameter. It may be incomplete and you may have forgotten the "@" (see also: TO_S and CC parameters (Page 3521)).

Return value STATUS* (W#16#...):	Explanation	Notes
8535	SMTP authentication incomplete	You have possibly entered an incorrect user name or incorrect password.
8550	Mail server cannot be reached. You have no access rights.	You may have entered an incorrect user name or password, or the mail server may not support your login. Another cause of error could be an incorrect entry of the domain name after the "@" or a missing "." at the TO_S, CC and FROM parameters (see also: TO_S and CC parameters (Page 3521)).
8552	Action aborted: Assigned memory size has been exceeded	Try again later.
8554	Transfer failed	Try again later.
8555	Error due to incorrect entry of the e-mail address.	If multiple addresses were entered, these need to be separated by a comma.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Example: Sending an e-mail with TMAIL_C

Introduction

The following call example shows you how the TMAIL_C instruction works.

Requirement

- Hardware must be configured beforehand for sending e-mails.
- The infrastructure of the network must allow a communication connection to the mail server.

E-mail attachment

The e-mail should be sent with an attachment.

Create an Array of Byte with the name "Data" in a global data block ("SLI_gDB_MailAttach_scl") as a source for the attachment. The array will be interconnected with the ATTACHMENT input parameter at a later time.

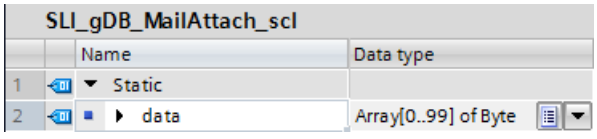

Storage of data

You enter the connection information of the CPU and address data of the mail server in the TMail_V4. system data type

- Create a global data block ("SLI_gDB_TMAIL_C_scl").
- Create a tag "Par1" with the data type "TMail_V4" in the data block.
- In the following structure, enter the parameters of your CPU based on the configuration and the connection data of your mail server. For more information, refer to the description of the MAIL_ADDR_PARAM (Page 3522) input parameter.

The "Par1" tag will be interconnected with the MAIL_ADDR_PARAM input parameter later.

SLI_gDB_TMAIL_C_scl			
	Name	Data type	Start value
1	Static		
2	par1	TMail_V4	
3	InterfaceId	HW_ANY	64
4	ID	CONN_OUC	100
5	ConnectionType	Byte	16#20
6	ActiveEstablished	Bool	false
7	CertIndex	Byte	16#0
8	WatchDogTime	Time	T#5s
9	MailServerAddress	IP_V4	
10	ADDR	Array[1..4] of Byte	
11	ADDR[1]	Byte	192
12	ADDR[2]	Byte	168
13	ADDR[3]	Byte	100
14	ADDR[4]	Byte	10
15	UserName	String[254]	'myusername'
16	PassWord	String[254]	'mypassword'
17	From	EMAIL_ADDR	
18	LocalPartPlusAtSign	String[64]	'station1@'
19	FullQualifiedDomainName	String[254]	'mycpu.com'

Also create the following tags in the global data block:

3	sendMail	Bool	false
4	target	String[240]	' <wenna@mydomain.com>,'
5	subject	String[240]	'Testmail'
6	text	String[240]	'The cpu switched to run'
7	busy	Bool	false
8	done	Bool	false
9	error	Bool	false
10	memErrStatus	Word	16#0

Calling the instruction

Create a new function block in the SCL language with the following interfaces:

7	Static		
8	status	Word	16#0
9	TMAIL_C_Instance	TMAIL_C	
10	Temp		
11	done	Bool	

Copy the following source code to the programming window with a call of TMAIL_C:

Call TMAIL_C

```
//Set "sendMail" to "TRUE" to start TMAIL_C.
#TMAIL_C_Instance(REQ:="SLI_gDB_TMAIL_C_scl".sendMail AND NOT("SLI_gDB_TMAIL_C_scl".busy),
            TO_S:="SLI_gDB_TMAIL_C_scl".target,
            SUBJECT:="SLI_gDB_TMAIL_C_scl".subject,
            TEXT:="SLI_gDB_TMAIL_C_scl".text,
            ATTACHMENT:="SLI_gDB_MailAttach_scl".data,
            MAIL_ADDR_PARAM:="SLI_gDB_TMAIL_C_scl".par1,
            DONE=>#done,
            BUSY=>"SLI_gDB_TMAIL_C_scl".busy,
            ERROR=>"SLI_gDB_TMAIL_C_scl".error,
            STATUS=>#status);
//INFO: Saves the success status of TMAIL_C.
IF #done THEN
   "SLI_gDB_TMAIL_C_scl".done := TRUE;
   "SLI_gDB_TMAIL_C_scl".sendMail := FALSE;
END_IF;
//INFO: In case of an error, saves the error status of TMAIL_C and resets "sendMail".
IF "SLI_gDB_TMAIL_C_scl".error THEN
   "SLI_gDB_TMAIL_C_scl".memErrStatus := #status;
   "SLI_gDB_TMAIL_C_scl".sendMail := FALSE;
END_IF;
```

Result

If the "sendMail" tag returns the signal state "TRUE" (edge change from "0" to "1" at the REQ parameter), the instruction "TMAIL_C" is executed. As soon as the "#done" tag returns the signal state "TRUE", the "sendMail" tag is automatically reset to the signal state "FALSE". Success in sending the e-mail ("#done" is "TRUE") is also saved in the "done" tag.

In the event of an error in the instruction "TMAIL_C", the value of the STATUS parameter ("#status") is saved in the "memErrStatus" tag and the "sendMail" tag is reset.

See also

Description of TMAIL_C (Page 3513)

Other

TCON: Establishing a communication connection

TCON: Establish communication connection (S7-1200)

Validity

The following description of the "TCON" instruction applies to S7-1200 CPUs with firmware version < V4.0.

Description

You use the "TCON" instruction to set up and establish a communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU. "TCON" executes asynchronously.

The connection data specified for the CONNECT and ID parameters are used to set up the communication connection. To establish the connection, a rising edge must be detected at the REQ parameter. Once the connection is successfully established, the DONE parameter is set to "1".

Note

Support when programming connections

If you select an instruction for communication TCON, TSEND_C or TRCV_C in a program block and create connections of the type TCP, UDP or ISO-on-TCP and want to assign parameters to them, you can use the support of the connection parameter assignment.

You can find the connection parameter assignment the Inspector window of the program editor.

Number of possible connections

For information on the number of possible communication connections, refer to the technical specifications for your CPU.

TCP and ISO-on-TCP connections

Both communication partners call the "TCON" instruction to set up and establish the communication connection. During parameter assignment, you specify which partner is the active communication end point and which is the passive one.

If the connection aborts, for example due to a line break or due to the remote communication partner, the active partner attempts to reestablish the configured connection. You do not have to call "TCON" again. This applies only if "TCON" has been executed successfully once (DONE = 1).

An existing connection is terminated and the connection set up is removed when the "TDISCON (Page 3548)" instruction is executed or when the CPU changes to STOP mode. To set up and establish the connection again, you will need to execute "TCON" again.

Parameters

The following table shows the parameters of the "TCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the job to establish the connection specified in the ID upon a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the assigned connection. Value range: W#16#0001 to W#16#0FFF
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: AUTOHOTSPOT
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not a job has been executed successfully. The ERROR parameter is set when errors occurred during execution of "TCON". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Connection successfully established.
0	7000	No job processing active
0	7001	Start job execution, establish connection.
0	7002	Connection is being established (REQ irrelevant).
1	8085	Connection ID (ID parameter) is already being used by a configured connection.
1	8086	The ID parameter is outside the valid range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8089	The CONNECT parameter does not point to a data block.
1	809A	The structure at the CONNECT parameter is not supported on an integrated interface or the length is invalid or an incorrect "InterfaceID" is specified in the connection description (SDT).
1	809B	The element InterfaceId within the TCON_xxx structure does not reference a hardware identifier of a CPU or CM/CP interface or has the value "0".
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.
1	80A1	The specified connection or the port is already being used.
1	80A2	Local or remote port is being used by the system.
1	80A3	Attempt being made to re-establish an existing connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A5	Connection ID is already in use.
1	80A7	Communication error: You executed "TDISCON (Page 3548)" before "TCON" had completed.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B4	You have violated one or more of the following conditions for passive connection establishment with the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0 • With local_tsap_id_len >= B#16#03, local_tsap_id[1] is an ASCII character. • local_tsap_id[1] is an ASCII character and local_tsap_id_len >= B#16#03.
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error at the connection_type parameter of the SDT TCON_Param.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len. Note: If you call TCON in TCP for the passive end, local_tsap_id_len must have the value 2 and rem_tsap_id_len the value 0.
1	80B8	Connection description of the structure element ID and the block parameter ID are not identical.
1	80C3	All connection resources are in use.

ERROR	STATUS* (W#16#...)	Explanation
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The connection cannot be established because the firewalls on the connection path are not open for the required ports. • The interface is currently receiving new parameters. • The configured connection is currently being removed by a "TDISCON (Page 3548)" instruction.
* The error codes in the program editor can be displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".		

See also

Difference between synchronous and asynchronous instructions (Page 596)

Switching display formats in the program status (Page 7698)

TCON: Establish communication connection (S7-1200, S7-1500)

Description

You use the "TCON" instruction to set up and establish a communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU. "TCON" executes asynchronously.

For S7-1500-CPUs as of firmware version V2.0 and higher, you can set up an access point for IPv4 multicast communication with the instruction "TCON" via the integrated PROFINET interfaces.

The connection data specified for the CONNECT and ID parameters are used to set up the communication connection. At the CONNECT parameter, if possible, only use the predefined structures as created by the connection parameter assignment in the Inspector window of the program editor.

As of firmware version V2.0 of the S7-1500 CPUs, you can specify the value 0 for specific TCON_xxx structures at the Interface ID parameter. You thus allow the firmware to select the Industrial Ethernet interface to be used. This specification of the value 0 is possible with the following SDTs: TCON_IP_V4_SEC, TCON_QDN, TCON_QDN_SEC, TMail_QDN_SEC, TMail_V4 and TMAIL_V4_SEC. With all other SDTs, you must assign a fixed value to the Interface ID.

To establish the connection, a rising edge must be detected at the REQ parameter. Once the connection is successfully established, the DONE parameter is set to "1".

Note

Support when programming connections

If you select an instruction for communication TCON, TSEND_C or TRCV_C in a program block and wish to create and assign parameters to connections of the type TCP, UDP, ISO-on-TCP or FDL, you can use the support of the connection parameter assignment.

You can find the connection parameter assignment in the Inspector window of the program editor.

Number of possible connections

For information on the number of possible communication connections, refer to the technical specifications for your CPU.

Connection with TCP and ISO-on-TCP

Both communication partners call the "TCON" instruction to set up and establish the communication connection. During parameter assignment, you specify which partner is the active communication end point and which is the passive one.

If the connection aborts, for example due to a line break or due to the remote communication partner, the active partner attempts to reestablish the configured connection. You do not have to call "TCON" again. This applies only if "TCON" has been executed successfully once (DONE = 1).

An existing connection is terminated and the connection set up is removed when the "TDISCON (Page 3548)" instruction is executed or when the CPU changes to STOP mode. To set up and establish the connection again, you will need to execute "TCON" again.

Telecontrol connections between CP and SMS client

Use the TCON_PHONE system data type for connection description in the CONNECT parameter.

For this connection type, the station requires access to the mobile network via a mobile network CP.

FDL connections

To set up the FDL connections of the CM 1542-5, see AUTOHOTSPOT.

Parameter

The following table shows the parameters of the "TCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the job to establish the connection specified on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the assigned connection. Value range: W#16#0001 to W#16#0FFF
CONNECT	InOut	VARIANT	D	Pointer to the connection description • For TCP or UDP, use the structure TCON_IP_v4 or TCON_QDN For a description, refer to: AUTOHOTSPOT or AUTOHOTSPOT • For TCP or UDP using secure communication, use the structure TCON_IP_V4_SEC or TCON_QDN_SEC For a description, refer to: AUTOHOTSPOT or AUTOHOTSPOT • For ISO-on-TCP, use the structure TCON_IP_RFC For a description, refer to: AUTOHOTSPOT • For ISO connections of the CP 1543-1, use the structure TCON_ISOnative. For a description, refer to: "Structure of the connection description according to TCON_ISOnative" • For connections to SMS clients, use the TCON_PHONE system data type. For a description, refer to AUTOHOTSPOT. • For FDL connections of the CM 1542-5, use the system data type TCON_FDL; see AUTOHOTSPOT.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not a job has been executed successfully. The ERROR parameter is set if errors occur during execution of "TCON". The error information is output at the STATUS parameter.

The instruction "TCON" generates an error message in version 3.0 if an active connection establishment to a remote partner fails. Create a rising edge at the REQ parameter to establish a connection once again.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Structure of the connection description according to TCON_ISOnative

A connection description DB with a structure according to TCON_ISOnative is used to assign the communication connection parameters for ISO. The fixed data structure of the TCON_ISOnative contains all parameters that are required to establish the connection.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceId	HW_ANY	0	Hardware identifier of the CP interface
2 … 3	ID	CONN_OUC	1	Reference to this connection (unique ID in the value range: 1 to 4095).
4	ConnectionType	BYTE	16#16	Connection type: ISO
5	ActiveEstablished	BOOL	TRUE	ID for the manner in which the connection is established: • FALSE: Passive connection establishment • TRUE: Active connection establishment
8 to 13	RemoteMacAddress	ARRAY [1..6] of BYTE	-	MAC address of the partner end point, for example, for 00-74-41-FD-AE-84: • MacAddr[1] = 00 • MacAddr[2] = 74 • MacAddr[3] = 41 • MacAddr[4] = FD • MacAddr[5] = AE • MacAddr[6] = 84

Byte	Parameter	Data type	Start value	Description
14 - 19	LocalMacAddress	ARRAY [1..6] of BYTE	-	MAC address of the local end point, for example, for 00-74-41-FD-AE-84: • MacAddr[1] = 00 • MacAddr[2] = 74 • MacAddr[3] = 41 • MacAddr[4] = FD • MacAddr[5] = AE • MacAddr[6] = 84
20 - 53	RemoteTSelector	Struct	-	TSelector of the remote connection partner: • Bytes 20 to 21 = TSelLength • Bytes 22 to 53 = TSel[1-32]
	TSelLength	UINT	-	Value range from 0 to 32
	TSel	ARRAY [1..32] of BYTE	-	Value range in each case 0 to 255 in bytes
54 - 87	LocalTSelector	Struct	-	TSelector of the remote connection partner: • Bytes 20 to 21 = TSelLength • Bytes 22 to 53 = TSel[1-32]
	TSelLength	UINT		Value range from 0 to 32
	TSel	ARRAY [1..32] of BYTE	-	Value range in each case 0 to 255 in bytes
88 - 89	CrRetransmitionTime	UINT	-	Duration until connection attempt is repeated in seconds.
90 - 91	DataRetransmition-Time	UINT	100 ms	Duration until data transfer is repeated in milliseconds.
92 - 93	MaxRetransmition-Count	UINT	-	Maximum number of repetitions.
94 - 95	InactivityTime	UINT	-	in seconds
96 - 97	WindowTime	UINT		in seconds

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Connection successfully established.
0	7000	No job processing active
0	7001	Start job execution, establish connection.
0	7002	Connection is being established (REQ irrelevant).
1	8085	Connection ID (ID parameter) is already being used by a configured connection.
1	8086	The ID parameter is outside the valid range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8089	The CONNECT parameter does not point to a connection description or the connection description was created manually.

ERROR	STATUS* (W#16#...)	Explanation
1	809A	The structure at the CONNECT parameter is not supported on an integrated interface or the length is invalid.
1	809B	• The InterfaceId element within the TCON_xxx structure does not reference a hardware identifier of a CPU or CM/CP interface. • The InterfaceId element cannot have the value "0" in the TCON_xxx structure used; see section "Description".
1	80A1	The specified connection or the port is already being used.
1	80A2	Local or remote port is being used by the system. The following ports are reserved locally: 20, 21, 80, 102, 135, 161, 162, 443, 34962, 34963, 34964 as well as the area 49152 to 65535.
1	80A3	ID is used by a connection created by the user program, which uses the same connection description at the CONNECT parameter.
1	80A4	IP address of the remote endpoint of the connection is invalid or it corresponds to the IP address of the local partner.
1	80A7	Communication error: You executed "TDISCON (Page 3548)" before "TCON" had completed.
1	80B4	Only with TCON_IP_RFC: The local T selector was not specified or the first byte does not contain the value 0x0E (only with a length of T selector = 2) or the local T selector starts with "SIMATIC-".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP (parameter ActiveEstablished of the structure TCON_IP_v4 / TCON_PARAM has the value TRUE).
1	80B6	Parameter assignment error in the ConnectionType parameter of the data block for connection description. • Only valid with TCON_IP_v4: 0x11, 0x0B and 0x13. • Only valid with TCON_IP_RFC: 0x0C and 0x12
1	80B7	With TCON_IP_v4: • TCP (active connection establishment): Remote port is "0". • TCP (passive connection establishment): Local port is "0". • UDP: Local port is "0". • IP address of the partner end point was set to 0.0.0.0. With TCON_IP_RFC: • Local (LocalTSelector) or remote (RemoteTSelector) T selector was specified with a length of more than 32 bytes. • For TSelLength of the T selector (local or remote), a length greater than 32 was entered. • Error in the length of the IP address of the specific connection partner. • IP address of the partner end point was set to 0.0.0.0.
1	80B8	Parameter ID in the local connection description (structure at CONNECT parameter) and parameter ID of the instruction are different.
1	80C3	All connection resources are assigned, or ports may be dynamically used by other applications or connections.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The connection cannot be established because the firewalls on the connection path are not open for the required ports. • The interface is currently receiving new parameters. • The configured connection is currently being removed by a "TDISCON (Page 3548)" instruction.

ERROR	STATUS* (W#16#...)	Explanation
1	80C5	The connection partner refuses to establish the connection, has terminated the connection or actively ended it.
1	80C6	The connection partner cannot be reached (network error).
1	80C7	Execution timeout.
1	80C8	Value at the ID parameter is already being used by a connection that was created using the user program. The connection uses the identical ID, but different connection settings at the parameter CONNECT.
1	80C9	Validation of the connection partner failed. The connection partner that wants to establish the connection does not match the defined partner of the structure at the CONNECT parameter.
1	80CE	The IP address of the local interface is 0.0.0.0.
1	80D0	In connection with TCP and the active connection end point: The remote_qdn parameter is an empty string. In this case, no connection can be established.
1	80D1	The remote_qdn parameter is not a fully qualified domain name. The period at the end may be missing.
1	80D2	No DNS server address is configured.
1	80D3	The fully qualified domain name could not be resolved. Possible causes: • The DNS server is not reachable, for example, because it has been shut down or the remote port is not reachable. • An error occurred during communication with the DNS server. • The DNS server returned a valid DNS answer, but the answer contained no IPv4 address.
1	80E0	Unsuitable or poor message was received.
1	80E1	Error during the handshake. Possible causes: • Abort by the user • Security not high enough • Renewed negotiation is not supported • SSL/TLS version is not supported • Validation of the host name failed
1	80E2	Certificate not supported / certificate invalid / no certificate Possible cause: The time-of-day of the module concerned is not set or the module is not synchronized. Example: The default setting for the date of the module is 1/1/2012 and it was not set during commissioning. The validity period of the certificate starts on 20 August 2016 and ends on 20 August 2024. In this case, the date of the module is outside the validity period of the certificate; the certificate is invalid for the module.
1	80E3	Certificate was discarded.
1	80E4	No valid certification authority found.
1	80E5	Certificate expired.
1	80E6	Integrity errors in the Transport Layer Security Protocol
1	80E7	Not supported extension in X.509-V3 certificate
1	80E9	TLS server without server certificate is not supported.
1	80EA	DTLS (UDP) protocol is not supported.
1	80EB	A client cannot request a client certificate.
1	80EC	The server cannot perform validation based on the subjectAlternateName (only clients can do this).

ERROR	STATUS* (W#16#...)	Explanation
1	80ED	TLSServerCertRef_m-ID invalid
1	8xyy	For general error information, refer to Evaluating errors with output parameter RET_VAL (Page 162)
* The error codes in the program editor can be displayed as integer or hexadecimal values. For information on switching the display formats, refer to "See also".		

Example

You can find the example here: Program example for TCON, TDISCON, TSEND & TRCV (Page 3566).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

Switching display formats in the program status (Page 7698)

TDISCON: Terminate communication connection (S7-1200, S7-1500)

Description

The "TDISCON" instruction terminates a communication connection from the CPU to a connection partner.

Functional description

"TDISCON" is an instruction that works asynchronously, which means its job processing extends over multiple calls. You start the job for terminating a connection by calling the "TDISCON" instruction with REQ = 1.

After "TDISCON" has been successfully executed, the ID specified for "TCON" is no longer valid and cannot be used for sending or receiving.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions (see also: Difference between synchronous and asynchronous instructions (Page 596)).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can recognize the current status of "TDISCON" or when the establishment of the connection is completed.

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	Job successfully completed.

BUSY	DONE	ERROR	Description
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS parameter.
0	0	0	The instruction was not assigned a (new) job.

Parameters

The following table shows the parameters of the instruction "TDISCON":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
REQ	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Control parameter REQUEST starts the job for terminating the connection specified by ID. The job starts on a rising edge.
ID	Input	CONN_OUC (WORD)	D, L or constant	I, Q, M, D, L or constant	Reference to the connection to be terminated (connection ID) Value range: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status parameter: • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status parameter: • BUSY = 1: The job is not yet complete. • BUSY = 0: The job has been completed or has not yet started.
ERROR	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status parameter: • ERROR = 0: No error. • ERROR=1: An error occurred during processing. STATUS supplies detailed information on the type of error
STATUS	Output	WORD	I, Q, M, D, L	I, Q, M, D, L	Status parameter: Error information (see "Parameters ERROR and STATUS")

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Connection terminated successfully.
0	7000	No job processing active.
0	7001	Start of job processing, connection being terminated.
0	7002	Intermediate call (REQ irrelevant), connection being terminated.
1	8086	The ID parameter is not in the permitted value range.

ERROR	STATUS* (W#16#...)	Explanation
1	80A3	Attempt is being made to terminate a non-existent connection or the connection is already terminated.
1	80C4	Temporary communication error: New parameters are being assigned to the interface or the connection is currently being established.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Example

You can find the example here: Program example for TCON, TDISCON, TSEND & TRCV (Page 3566).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

TSEND: Send data via communication connection

TSEND: Send data via communication connection (S7-1200)

Description

The following description of the "TSEND" instruction is valid for the CPU S7-1200 to version 3.0.

You use the "TSEND" instruction to send data over an existing communication connection. TSEND executes asynchronously.

You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. All data types except BOOL and Array of BOOL can be used for the data to be sent.

The send job is executed when a rising edge is detected at the REQ parameter.

With the LEN parameter, you specify the maximum number of bytes sent with a send job.

- When data is transferred with TCP (streaming protocol), the "TSEND" instruction provides no information about the length of the data sent to "TRCV (Page 3557)".
- When data is transferred with ISO-on-TCP (message-oriented protocol), the length of the data sent is communicated to "TRCV (Page 3557)". The amount of data sent with "TSEND" as packet must also be received again at the receiving end ("TRCV (Page 3557)"):
  - If the receive buffer is too small for the sent data, an error occurs at the receiving end.
  - If the receive buffer is sufficiently large, "TRCV" returns with DONE=1 as soon as the data packet is received.

The data to be sent must not be edited until the send job is completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter is not confirmation that the data sent has already been read by the communication partner.

Parameter

The following table shows the parameters of the "TSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the send job on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON". Value range: W#16#0001 to W#16#0FFF
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum number of bytes to be sent with the job.
DATA	InOut	VARIANT	I, Q, M, D	Pointer to the send area containing the address and the length of the data to be sent. The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block When transferring structures, the structures must be identical at the sending and receiving end.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

LEN and DATA parameters

- With LEN = 0 , all the data specified with the DATA parameter is sent.
- If the number of bytes at the LEN parameter is larger than the length of the data to be sent that was defined with the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS parameter below).
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- With data types STRING and WSTRING, all data is transferred if the parameter LEN = 0. When LEN > 0, the length must be at least the maximum number of bytes plus two additional bytes which contain the length information. You will find more detailed information on the structure of the data types in: "Overview of the valid data types (Page 231)".

- The maximum number of bytes that can be transmitted amounts to 8192.
- When you use structured tags from optimized DBs, the address of the structured tag should be interconnected at the DATA parameter and the parameter LEN = 0. This guarantees type-safe transmission of the entire structure if the same structure is used at the receiving end.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TSEND" is executed asynchronously, you must keep the data in the send area consistent until the DONE parameter or the ERROR parameter changes to the value "1".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Send job completed without error.
0	7000	No job processing active.
0	7001	Start of job execution, data is being sent. When processing this job, the operating system accesses the data in the DATA send area.
0	7002	Job executing (REQ irrelevant). When processing this job, the operating system accesses the data in the DATA send area.
1	8085	• The LEN parameter is greater than the highest permitted value (65536). • DATA and LEN parameters both have the value "0".
1	8086	The ID parameter is outside the permitted address range (1..0xFFF).
1	8088	The LEN parameter is greater than the area specified in DATA.
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Transfer via this connection is not possible. • The interface is being re-initialized.

ERROR	STATUS* (W#16#...)	Description
1	80B3	The protocol variant (ConnectionType parameter in the connection description) is set to UDP. Use the instruction "TUSEND" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.
1	80C5	Connection terminated by the communication partner.
1	80C6	Network error. Communication partner cannot be reached.
1	80C7	Execution timeout.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

See also

Difference between synchronous and asynchronous instructions (Page 596)

TSEND: Send data via communication connection (S7-1200, S7-1500)

Description

The following description of the "TSEND" instruction is valid for:

- Ethernet
  CPU S7-1200 with firmware version ≥ V4.0 and CPU S7-1500
- PROFIBUS
  FDL connections of the S7-1500 with CM 1542-5 as of V2.0 with the system data type TCON_FDL

You use the "TSEND" instruction to send data over an existing communication connection. TSEND executes asynchronously.

You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. All data types except BOOL and Array of BOOL can be used for the data to be sent.

The send job is executed when a rising edge is detected at the REQ parameter.

With the LEN parameter, you specify the maximum number of bytes sent with a send job.

- When data is transferred with TCP (streaming protocol), the "TSEND" instruction provides no information about the length of the data sent to "TRCV (Page 3557)".
- When data is transferred with ISO-on-TCP (message-oriented protocol), the length of the data sent is communicated to "TRCV (Page 3557)". The amount of data sent with "TSEND" as packet must also be received again at the receiving end ("TRCV (Page 3557)"):
  - If the receive buffer is too small for the sent data, an error occurs at the receiving end.
  - If the receive buffer is sufficiently large, "TRCV" returns with DONE=1 as soon as the data packet is received.

The data to be sent must not be edited until the send job is completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter is not confirmation that the data sent has already been read by the communication partner.

Parameter

The following table shows the parameters of the "TSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the send job on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON". Value range: W#16#0001 to W#16#0FFF
LEN	Input	UDINT	I, Q, M, D, L or constant	Maximum number of bytes that are sent with the job (maximum permissible value for S7-1200: 8192, maximum permissible value for S7-1500: 65536). For FDL connections of the CM 1542-5, the maximum length is 240 bytes. In this regard, note the maximum lengths that can be processed by the connection partner.
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the send area containing the address and the length of the data to be sent. The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block • Local data When transferring structures, the structures must be identical at the sending and receiving end.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

LEN and DATA parameters

- With LEN = 0 , all the data specified with the DATA parameter is sent.
- If the number of bytes at the LEN parameter is larger than the length of the data to be sent that was defined with the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS parameter below).
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- With data types STRING and WSTRING, all data is transferred if the parameter LEN = 0. When LEN > 0, the length must be at least the maximum number of bytes plus two additional bytes which contain the length information. You will find more detailed information on the structure of the data types in: "Overview of the valid data types (Page 231)".
- The maximum number of bytes that can be transmitted depends on the device.
- When you use structured tags from optimized DBs, the address of the structured tag should be interconnected at the DATA parameter and the parameter LEN = 0. This guarantees type-safe transmission of the entire structure if the same structure is used at the receiving end.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set if errors occur during execution of "TSEND". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TSEND" is executed asynchronously, you must keep the data in the send area consistent until the DONE parameter or the ERROR parameter changes to the value "1".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Send job completed without error.
0	7000	No job processing active.
0	7001	Start of job execution, data is being sent. When processing this job, the operating system accesses the data in the DATA send area.
0	7002	Job executing (REQ irrelevant). When processing this job, the operating system accesses the data in the DATA send area.
1	8085	• The LEN parameter is larger than the highest permitted value. For S7-1200: 8192; for S7-1500 (TCP): 65536; for S7-1500 (FDL): 240 / 236 • DATA and LEN parameters both have the value "0".
1	8086	The ID parameter is outside the permitted address range (1..0xFFF).
1	8088	The LEN parameter is greater than the area specified in DATA.
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Transfer via this connection is not possible. • The interface is being re-initialized.
1	80B1	You changed the DATA parameter before the current job finished.
1	80B3	The protocol variant (ConnectionType parameter in the connection description) is set to UDP. Use the instruction "TUSEND" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established. • Temporarily no receive resources available at the connection partner. The connection partner is not ready to receive. • The R/H system is in the SYNCUP system state or there is a primary-backup switchover. The connection is closed. Processing of TSEND is stopped.
1	80C5	• Connection terminated by the communication partner. • LSAP of the remote connection partner is not released
1	80C6	Network error: • Remote partner cannot be reached. • Physical interruption on PROFIBUS
1	80C7	Execution timeout.
1	80EE	Handshake not yet completed
1	8xyy	For general error information, refer to Evaluating errors with output parameter RET_VAL (Page 162)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Example

You will find the example here: Program example for TCON, TDISCON, TSEND & TRCV (Page 3566).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

TSEND: Send data via communication connection (Page 3550)

TRCV: Receive data via communication connection

TRCV: Receive data via communication connection (S7-1200)

Description

The following description of the "TRCV" instruction is valid for the CPU S7-1200 up to version 3.0.

You use the "TRCV" instruction to receive data over an existing communication connection. "TRCV" executes asynchronously.

Receipt of data is enabled when the EN_R parameter is set to the value "1". The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0), depending on the protocol variant being used.

While data is being received, you cannot make changes to the DATA parameter or the defined receive area to ensure consistency of the received data.

After successful receipt of data, the NDR parameter is set to the value "1". You can query the amount of data actually received at the RCVD_LEN parameter.

Receive modes of "TRCV"

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type* parameter of the connection description	Parameter LEN
TCP (Ad-hoc mode)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	0
TCP (Receipt of data with specified length)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192

Protocol variant	Availability of data in the receive area	connection_type* parameter of the connection description	Parameter LEN
ISO on TCP (Message-oriented data transfer)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	• 1 to 1452, if a CP is used. • 1 to 8192, if no CP is used.
* See "AUTOHOTSPOT".			

TCP (Ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV" instruction.

You set ad-hoc mode by assigning the value "0" to the LEN parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). When ad-hoc mode is active, receipt of data is already signaled after a received byte at the NDR parameter.

TCP (Receipt of data with specified length)

For receipt of data with a specified length, enter the length of the data at the LEN parameter. The data receipt is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

ISO on TCP (Message-oriented data transfer)

Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. When using ISO on TCP, "TRCV" signals data receipt as soon as the message block has been completely received. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV" signals an error. The successful receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

Parameter

The following table shows the parameters of the "TRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Receive enable
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON (Page 3538)". Value range: W#16#0001 (1) to W#16#0FFF (4095)

Parameter	Declaration	Data type	Memory area	Description
LEN	Input	UINT	I, Q, M, D, L or constant	Length of the receive area in bytes (hidden). If you use a memory area with optimized access at the DATA parameter, the LEN parameter must have the value "0".
DATA	InOut	VARIANT	I, Q, M, D	Pointer to the receive area When transferring structures, the structures must be identical at the sending and receiving end.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter (New Data Received): • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter: Output of status and error information.
RCVD_LEN	Output	UINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

LEN, DATA and RCVD_LEN parameters

- If LEN = 0, the received data is saved in the receive area specified at the DATA parameter. The number of bytes received is indicated at the RCVD_LEN parameter.
  With LEN = 0 (default setting of the LEN parameter), the length of the data to be received is defined by the DATA parameter. It is recommended that the receive area (DATA parameter) has the same size as the data to be transmitted by TSEND.
  If LEN has the value 0 and the sent data was transferred in segments that are smaller than the DATA receive area, the following applies. It is recommended to keep EN_R set until the associated TSEND instruction has sent all data. STATUS shows the value 7002 as long as the size of the data sent by TSEND is not equal to the size of the DATA receive area. EN_R must be set until the amount of the received data is equal to the size of the DATA receive area. If you pulse EN_R, you must do so until BUSY=0 or ERROR <> 0.
  The data in the DATA receive area is only valid when BUSY has the value 0.
- If the length specified at the LEN parameter is greater than the length of the data received at the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS) parameter in the following.
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.

- If the DATA parameter references a data block with optimized access, the LEN parameter must be set to "0".
- If a STRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=2 (LEN = 1 is not permitted).
- If a WSTRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=5.

BUSY, NDR and ERROR parameters

You can check the status of the job with the BUSY, NDR, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the NDR parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of TRCV. The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, NDR and ERROR parameters:

BUSY	NDR	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TRCV" is executed asynchronously, the data in the receive area is only consistent when the NDR parameter is set to the value "1".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Job completed successfully. The current length of the received data is output at the RCVD_LEN parameter.
0	7000	Block not ready to receive.
0	7001	Block is ready to receive, receive job was activated.
0	7002	Interim call, the receive job is executing. Note: While the job is being processed, data is written to the receive area. Access to the receive area during this time may provide inconsistent data.
1	8085	• The LEN parameter is larger than the highest permitted value. • The value of the LEN or DATA parameter was changed after the first call. • Both LEN parameters and the DATA parameter have the value "0" or LEN is longer than the maximum permitted value (65536).
1	8086	The ID parameter is outside the permitted address range (1 .. 0x0FFF).
1	8088	• Receive area is too small. • The value at the LEN parameter is larger than the receive area set at the DATA parameter.

ERROR	STATUS* (W#16#...)	Explanation
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Receive job over this connection is not possible. • The connection is being re-initialized.
1	80B3	The protocol variant (connection_type parameter in the connection description) is set to UDP. Use the instruction "TURCV" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.
1	80C5	The remote partner has terminated the connection.
1	80C6	The remote partner cannot be reached (network error).
1	80C7	Execution timeout.
1	80C9	The length of the receive area is smaller than the length of the sent data.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

See also

Difference between synchronous and asynchronous instructions (Page 596)

TRCV: Receive data via communication connection (S7-1200, S7-1500)

Description

The following description of the "TRCV" instruction is valid for:

- Ethernet
  CPU S7-1200 with firmware version ≥ V4.0 and CPU S7-1500
- PROFIBUS
  FDL connections of the S7-1500 with CM 1542-5 as of V2.0 with the system data type TCON_FDL

You use the "TRCV" instruction to receive data over an existing communication connection. "TRCV" executes asynchronously.

Receipt of data is enabled when the EN_R parameter is set to the value "1". The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0), depending on the protocol variant being used.

While data is being received, you cannot make changes to the DATA parameter or the defined receive area to ensure consistency of the received data.

After successful receipt of data, the NDR parameter is set to the value "1". You can query the amount of data actually received at the RCVD_LEN parameter.

Receive modes of "TRCV"

The following table shows how the received data is entered in the receive area.

Protocol variant	Parameter ADHOC	Availability of data in the receive area	connection_type parameter of the connection description	Parameter LEN
TCP (Ad-hoc mode)	1 (ad-hoc activated)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	As much data will be read as is currently available; not exceeding the data length specified by LEN. LEN = 0 (recommended): Length of the receive area referenced by DATA
TCP (Receipt of data with specified length)	0 (ad-hoc deactivated)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	• S7-1200: 1 to 8192 • S7-1500: 1 to 65536
ISO on TCP (Message-oriented data transfer)	-	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	• S7-1200: 1 to 8192 • S7-1500: 1 to 65536
FDL	-	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#15 Decimal: 21	1 to 240

TCP (Ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV" instruction.

You set ad-hoc mode by assigning the value "1" to the ADHOC parameter. When ad-hoc mode is active, receipt of data is already signaled after a received byte at the NDR parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.).

TCP (Receipt of data with specified length)

Assign the value "0" to the ADHOC parameter for receipt of data with specified length. If ad-hoc mode is disabled, the data reception is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The successful receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

ISO on TCP (Message-oriented data transfer)

Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. When using ISO on TCP, "TRCV" signals data receipt as soon as the message block has been completely received. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV" signals an error. The successful receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

Parameter

The following table shows the parameters of the "TRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Receive enable
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON". Value range: W#16#0001 to W#16#0FFF
LEN	Input	UDINT	I, Q, M, D, L or constant	Length of the receive area in bytes (hidden) Maximum value for S7-1200: 8192; maximum value for S7-1500: 65536; maximum value for CM 1542-5 (FDL): 240. If you use a receive area with optimized access at the DATA parameter, the LEN parameter must have the value "0".
ADHOC	Input	BOOL	I, Q, M, D, L or constant	Use ad-hoc mode for the TCP protocol variant (hidden). ADHOC must have the value FALSE if no TCP protocol is used.
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area When transferring structures, the structures must be identical at the sending and receiving end.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter (New Data Received): • 0: Job not yet started or still in progress • 1: New data received
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

LEN, DATA and RCVD_LEN parameters

- If LEN = 0, the received data is saved in the receive area specified at the DATA parameter. The number of bytes received is indicated at the RCVD_LEN parameter.
  With LEN = 0 (default setting of the LEN parameter), the length of the data to be received is defined by the DATA parameter. It is recommended that the receive area (DATA parameter) has the same size as the data to be transmitted by TSEND.
  If LEN has the value 0 and the sent data was transferred in segments that are smaller than the DATA receive area, the following applies. It is recommended to keep EN_R set until the associated TSEND instruction has sent all data. STATUS shows the value 7002 as long as the size of the data sent by TSEND is not equal to the size of the DATA receive area. EN_R must be set until the amount of the received data is equal to the size of the DATA receive area. If you pulse EN_R, you must do so until BUSY=0 or ERROR <> 0.
  The data in the DATA receive area is only valid when BUSY has the value 0.
- If the length specified at the LEN parameter is greater than the length of the data received at the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS) parameter in the following.
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- If the DATA parameter references a data block with optimized access, the LEN parameter must be set to "0". If the length of the data does not match for elementary data types, the data will not be received and the error code 8088 is output at the STATUS parameter.
- If a STRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=2 (LEN = 1 is not permitted).
- If a WSTRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=5.

BUSY, NDR and ERROR parameters

You can check the status of the job with the BUSY, NDR, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the NDR parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of TRCV. The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, NDR and ERROR parameters:

BUSY	NDR	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TRCV" is executed asynchronously, the data in the receive area is only consistent when the NDR parameter is set to the value "1".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Job completed successfully. The current length of the received data is output at the RCVD_LEN parameter.
0	7000	Block not ready to receive.
0	7001	Block is ready to receive, receive job was activated.
0	7002	Interim call, the receive job is executing. Note: While the job is being processed, data is written to the receive area. Access to the receive area during this time may provide inconsistent data.
1	8085	• The LEN parameter is greater than the maximum permissible value (for S7-1200: 8192 bytes, for S7-1500: 65536 bytes). • The value of the LEN or DATA parameter was changed after the first call. • Both LEN parameters and the DATA parameter have the value "0" or LEN is longer than the maximum permissible value (for S7-1200: 8192 bytes, for S7-1500: 65536 bytes).
1	8086	The ID parameter is outside the permitted value range (1 .. 0x0FFF).
1	8088	• Receive area is too small. • The value at the LEN parameter is larger than the receive area set at the DATA parameter.
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Receive job over this connection is not possible. • The connection is being re-initialized.
1	80B1	You changed the DATA parameter before the current job finished.
1	80B3	The protocol variant (connection_type parameter in the connection description) is set to UDP. Use the instruction "TURCV" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established. • The R/H system is in the SYNCUP system state or there is a primary-backup switchover. The connection is closed. Processing of TRCV is stopped.
1	80C5	The remote partner has terminated the connection.
1	80C6	The remote partner cannot be reached (network error).
1	80C7	Execution timeout.
1	80C9	The length of the receive area is smaller than the length of the sent data.
1	80EE	Handshake not yet completed
1	8xyy	For general error information, refer to Evaluating errors with output parameter RET_VAL (Page 162)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Example

You can find the example here: Program example for TCON, TDISCON, TSEND & TRCV (Page 3566).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

TRCV: Receive data via communication connection (Page 3557)

TCON: Establish communication connection (Page 3541)

Difference between synchronous and asynchronous instructions (Page 596)

Program example for TCON, TDISCON, TSEND & TRCV

Introduction

In the following example, you create a programmed connection between two CPUs of the S7-1500 series and send a data record from CPU 1 to CPU 2.

Requirement

- Two CPUs of the S7-1500 series are available and connected to each other over PROFINET. Their connection is not yet configured.

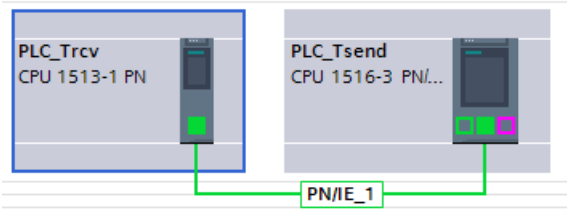

- A low protection level under Properties > Protection ensures that read and write access is permitted for each CPU.

Program of CPU 1: Storage of data

For the data record, create the following PLC data type.

TSEND_User

	Name	Data type	Default value
1	prename	String	''
2	lastname	String	''
3	userId	UInt	0

For the data transfer, create the following data block ("SLI_plcDB_sendData_TSEND") based on the created PLC data type.

SLI_plcDB_sendData_TSEND			
	Name	Data type	Start value
1	Static		
2	prename	String	'Simon'
3	lastname	String	'Sample'
4	userId	UInt	1

To store the data, create a global data block ("SLI_gDB_TSEND") with the following structures and tags.

SLI_gDB_TSEND			
	Name	Data type	Start value
1	Static		
2	connectionID	CONN_OUC	1
3	TCON	Struct	
4	start	Bool	false
5	busy	Bool	false
6	done	Bool	false
7	error	Bool	false
8	status	Word	16#0
9	memErrStatus	Word	16#0
10	TSEND	Struct	
11	start	Bool	false
12	maxLength	UInt	0
13	busy	Bool	false
14	done	Bool	false
15	error	Bool	false
16	status	Word	16#0
17	memErrStatus	Word	16#0
18	TDISCON	Struct	
19	start	Bool	false
20	busy	Bool	false
21	done	Bool	false
22	error	Bool	false
23	status	Word	16#0
24	memErrStatus	Word	16#0

Program of CPU 1: Interconnect parameters

Create the FB "SLI_FB_TSEND" Create the following local tags in it.

7	Static			
8	TCON_Instance	TCON		
9	TSEND_Instance	TSEND		
10	TDISCON_Instance	TDISCON		
11	doneCON	Bool	false	Non-retain
12	doneSEND	Bool	false	Non-retain
13	doneDISC	Bool	false	Non-retain

Network 1: Interconnect the parameter of the instruction "TCON" as follows:

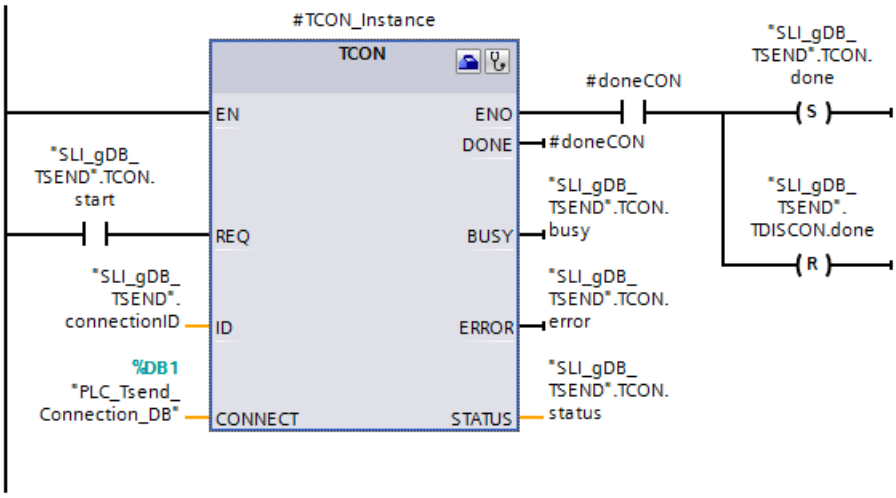

Network 2: In case of error of TCON, save the status as follows.

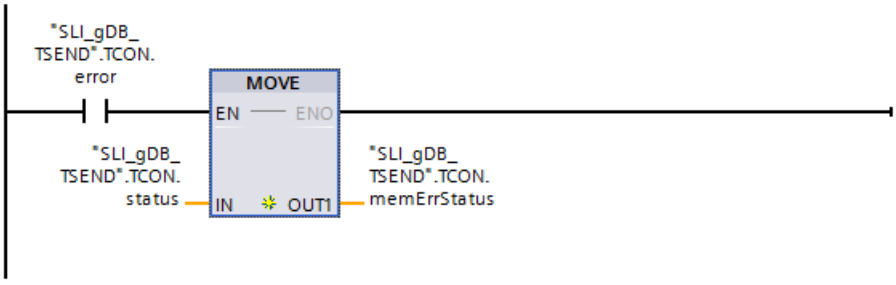

Network 3: Interconnect the parameter of the instruction "TSEND" as follows:

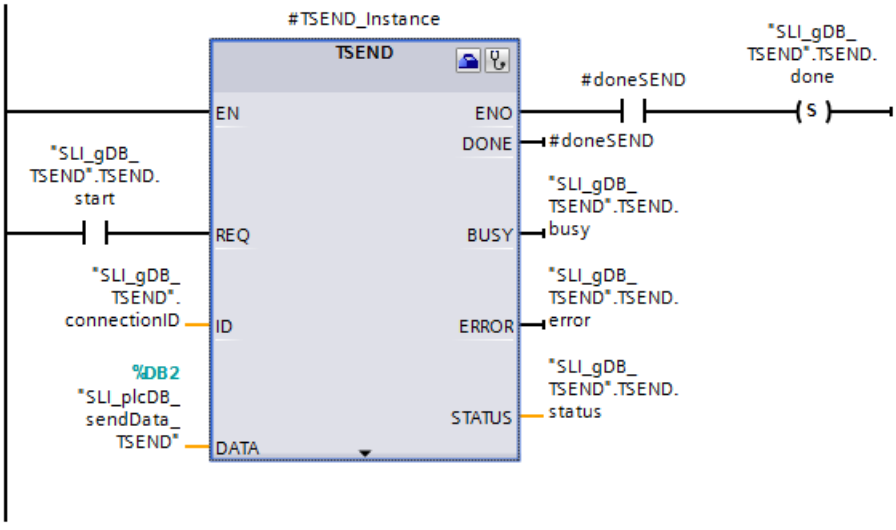

Network 4: In case of error of TSEND, save the status as follows.

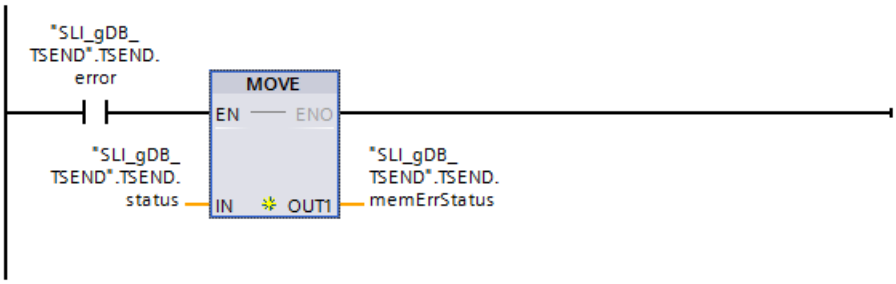

Network 5: Interconnect the parameter of the instruction "TDISCON" as follows:

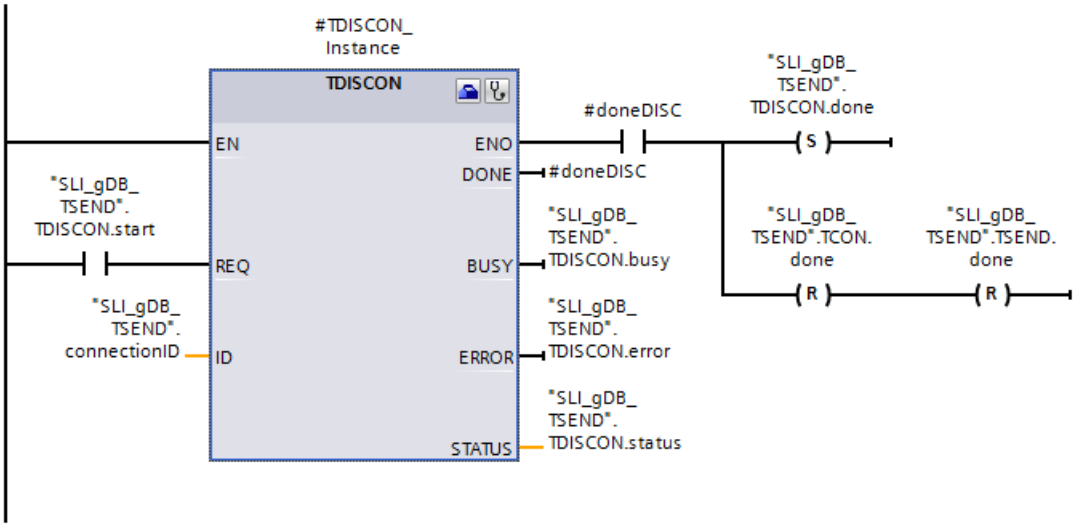

Network 6: In case of error of TDISCON, save the status as follows.

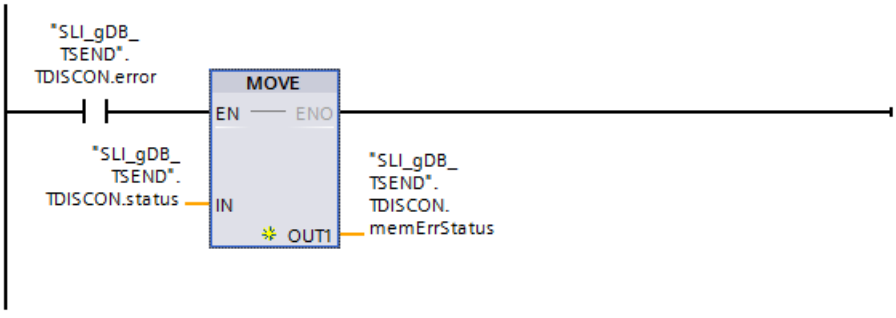

Program of CPU 1: Configuration of TCON

To interconnect the input parameter CONNECT, open the wizard of the "TCON" instruction with its Properties > Configuration.

Make the following settings for the configuration of TCON:

Input field	Entry
End point	Select a CPU for sender and receiver using the drop-down list. Interface, subnet and address are entered automatically.
Connection data	Use the drop-down list and the selection "New" to create one data block per CPU. This data block is required for storing the connection data. The name of the data block can be freely selected. The active connection establishment is enabled for CPU 1 (local CPU).
Connection type	Select the entry "ISO-on-TCP". This means an Ethernet connection with the "ISO-on-TCP" protocol is used for connection setup.
Connection ID	You enter a freely selected connection ID for the communication connection. The connection ID must not already be assigned in the project. Ensure that the value matches the value of the "connectionID" tag that is used.
TSAP ID	The address details are automatically entered during the selection of the connection type "ISO-on-TCP".

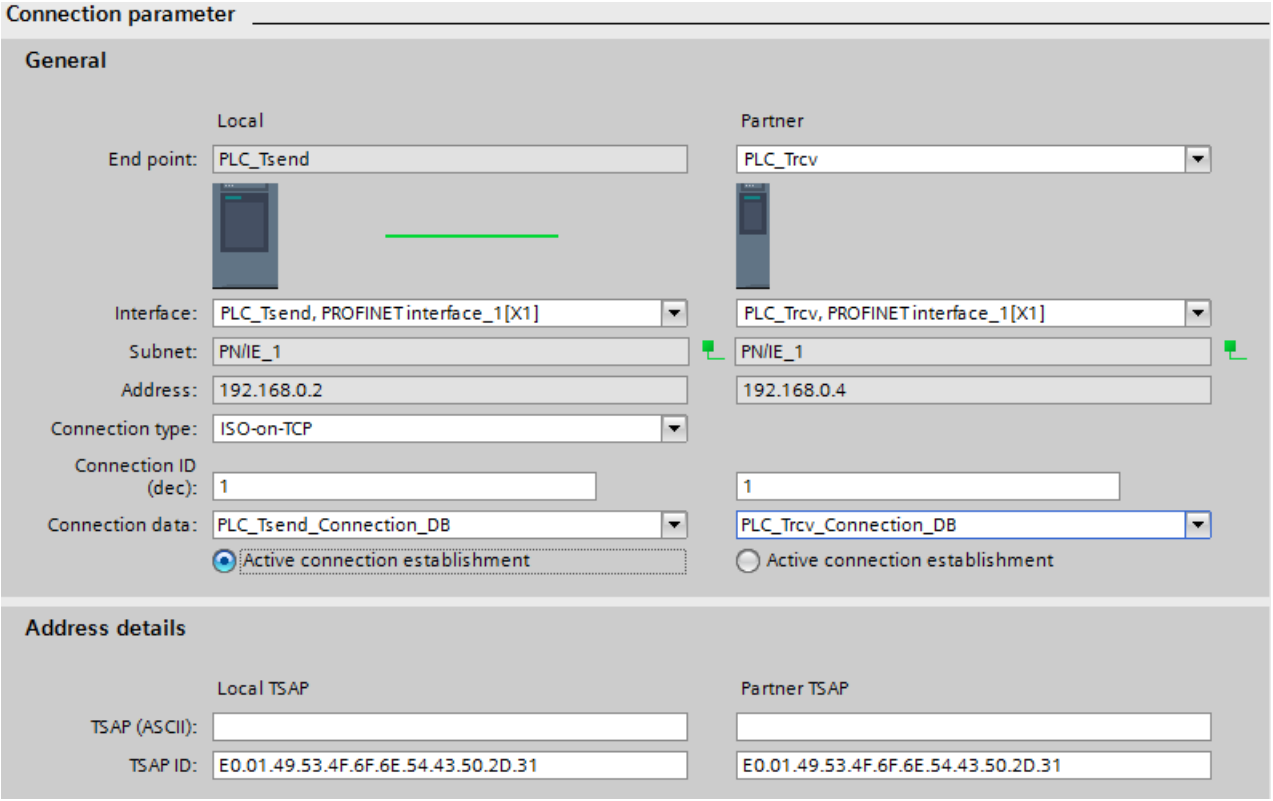

Program of CPU 2: Storage of data

For the data record, create the following PLC data type.

TSEND_User			
	Name	Data type	Default value
1	prename	String	''
2	lastname	String	''
3	userId	UInt	0

For the data transfer, create the following data block ("SLI_plcDB_rcvData_TRCV") based on the created PLC data type.

SLI_plcDB_rcvData_TRCV			
	Name	Data type	Start value
1	Static		
2	prename	String	''
3	lastname	String	''
4	userId	UInt	0

To store the data, create a global data block ("SLI_gDB_TRCV") with the following structures and tags.

SLI_gDB_TRCV			
	Name	Data type	Start value
1	Static		
2	connectionID	CONN_OUC	1
3	TCON	Struct	
4	start	Bool	false
5	busy	Bool	false
6	done	Bool	false
7	error	Bool	false
8	status	Word	16#0
9	memErrStatus	Word	16#0
10	TRCV	Struct	
11	start	Bool	false
12	busy	Bool	false
13	done	Bool	false
14	error	Bool	false
15	status	Word	16#0
16	memErrStatus	Word	16#0
17	readLength	UInt	0
18	TDISCON	Struct	
19	start	Bool	false
20	busy	Bool	false
21	done	Bool	false
22	error	Bool	false
23	status	Word	16#0
24	memErrStatus	Word	16#0

Program of CPU 2: Interconnect parameters

Create the FB "SLI_FB_TRCV" Create the following local tags in it.

7	Static			
8	doneCON	Bool	false	Non-ret...
9	doneRCV	Bool	false	Non-retain
10	doneDISC	Bool	false	Non-retain
11	length	UInt	0	Non-retain
12	TCON_Instance	TCON		
13	TRCV_Instance	TRCV		
14	TDISCON_Instance	TDISCON		

Network 1: Interconnect the parameter of the instruction "TCON" as follows:

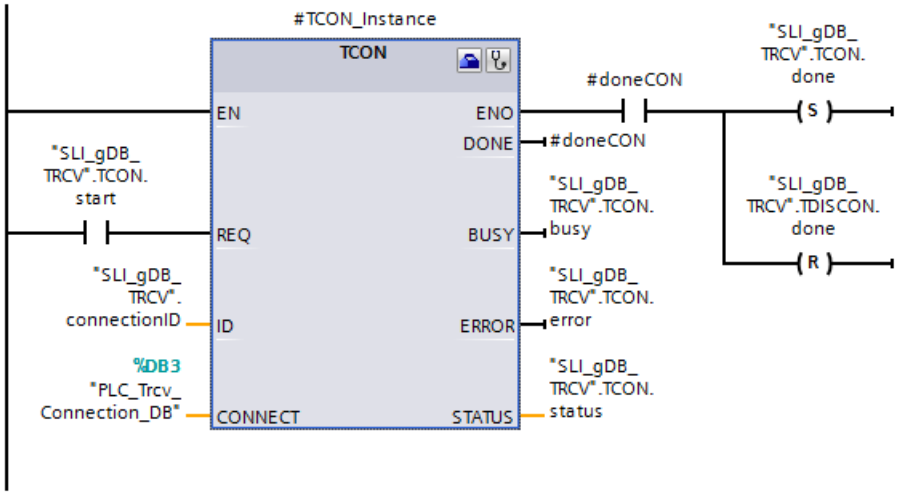

Network 2: In case of error of TCON, save the status as follows.

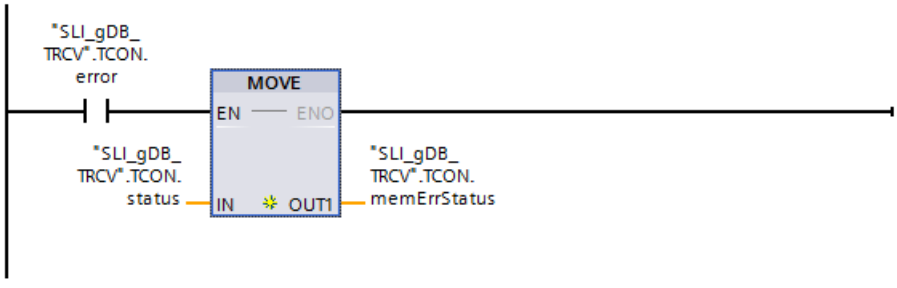

Network 3: Interconnect the parameter of the instruction "TRCV" as follows:

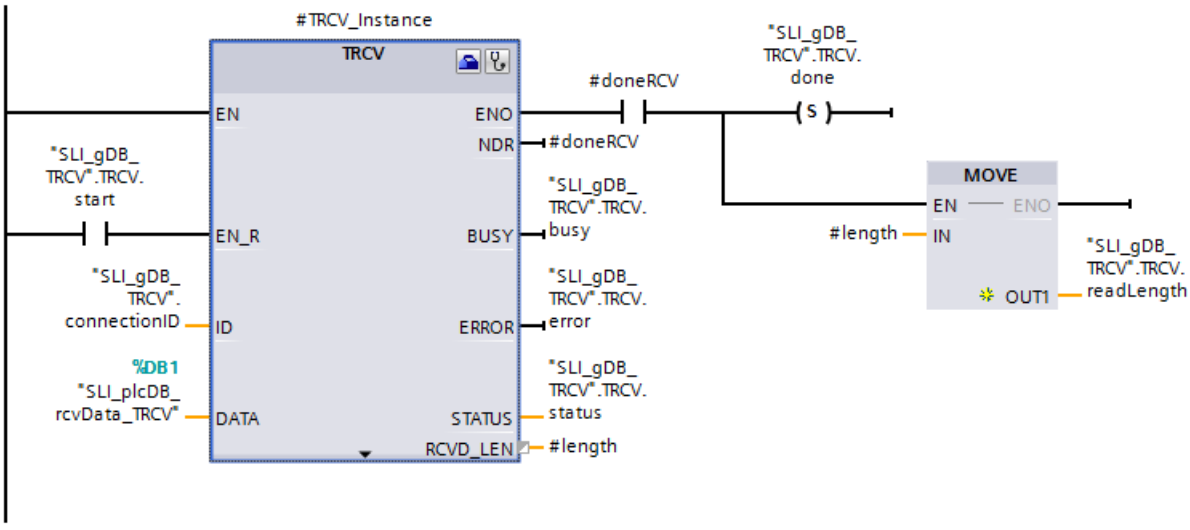

Network 4: In case of error of TRCV, save the status as follows.

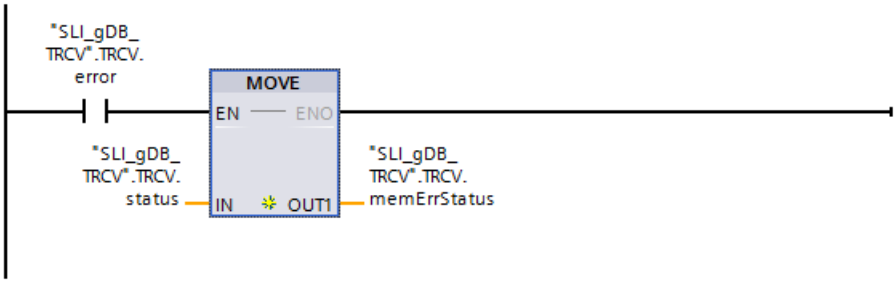

Network 5: Interconnect the parameter of the instruction "TDISCON" as follows:

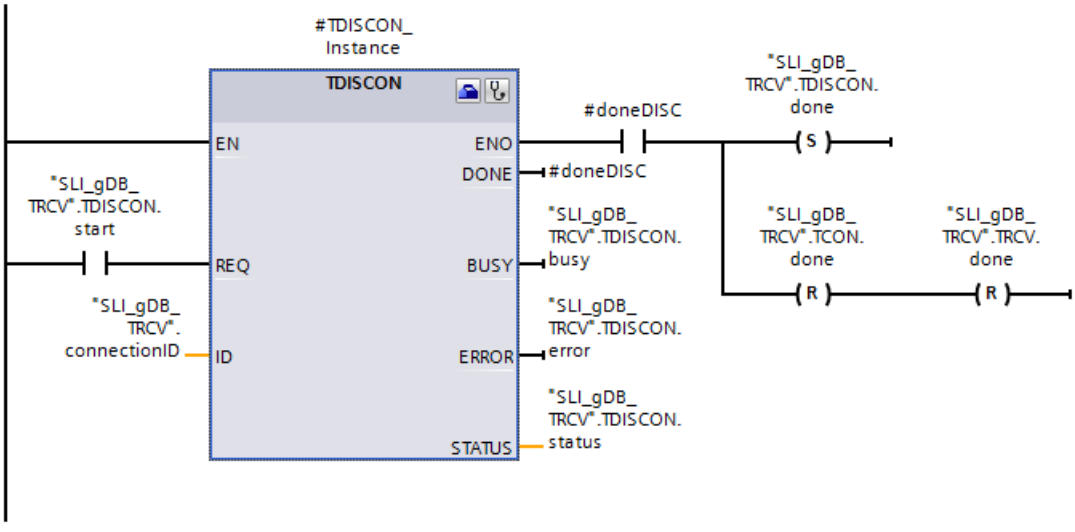

Network 6: In case of error of TDISCON, save the status as follows.

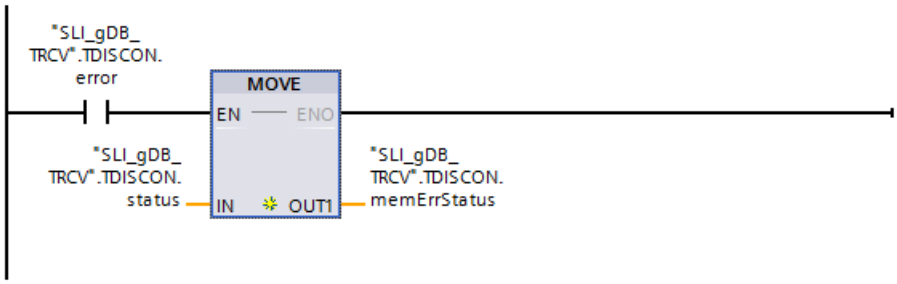

Program of CPU 2: Configuration of TCON

You make the following settings for the connection of CPU 2 / TCON according to the example of CPU 1 / TCON:

Note

Connection data

By configuring CPU 1 / TCON, you have already created a data block ("PLC_Tsend_Connection_DB", "PLC_Trcv_Connection_DB") with the stored connection data for each CPU. Instead of creating new data blocks and entering the connection data, you can simply use the created data blocks ("PLC_Tsend_Connection_DB", "PLC_Trcv_Connection_DB").

For the created data blocks to be available for selection, they must still be available in the project tree under Program blocks > System blocks.

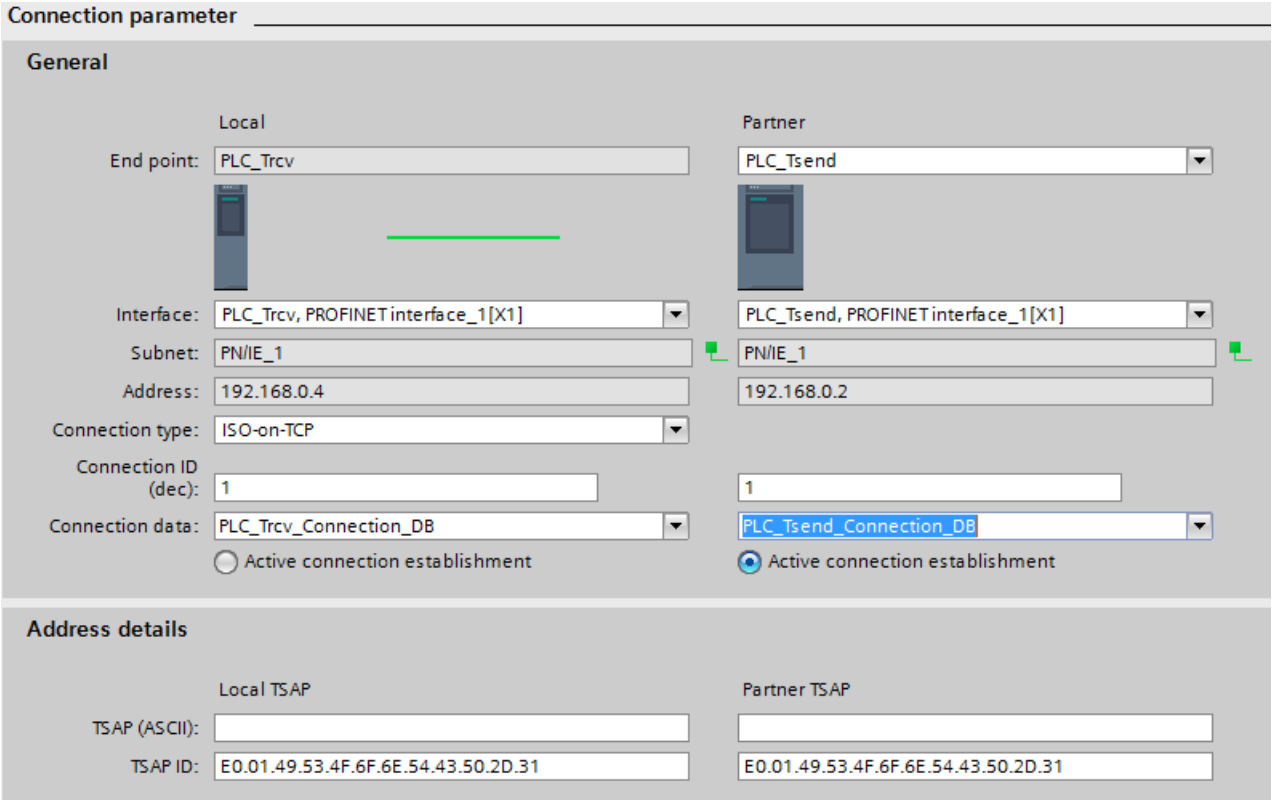

Procedure for establishing connection

Observe the order of the action steps:

1. Change the parameter REQ to "TRUE" for CPU 2 / TCON.
2. Change the parameter REQ to "TRUE" for CPU 1 / TCON.
3. Change the parameter REQ to "FALSE" for CPU 2 / TCON.
4. Change the parameter REQ to "FALSE" for CPU 1 / TCON.

Procedure for terminating connection

Observe the order of the action steps:

1. Change the parameter REQ to "TRUE" for CPU 1 / TDISCON.
2. Change the parameter REQ to "TRUE" for CPU 2 / TDISCON.
3. Change the parameter REQ to "FALSE" for CPU 1 / TDISCON.
4. Change the parameter REQ to "FALSE" for CPU 2 / TDISCON.

Behavior of CPU 1

Network 1 (TCON):

When the input parameter REQ ("TCON.start") supplies the signal state "TRUE", the "TCON" instruction is started. For multiple calls, the "TCON" instruction creates a communication connection to the partner CPU. To do this, the connection data is retrieved with the input parameter CONNECT (data block "PLC_Trcv_Connection_DB").

Successful connection setup is indicated by the output parameter DONE ("#doneCON") with the signal state "TRUE" and the output parameter STATUS ("TCON.status") with the value "0000". Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "TCON.done". Reset the success status of any previous connection termination ("TDISCON.done").

The output parameter ERROR ("TCON.error") or the tag "TCON.memErrStatus" indicates that processing in the example is taking place without errors.

3	TCON	Struct		
4	start	Bool	false	TRUE
5	busy	Bool	false	FALSE
6	done	Bool	false	TRUE
7	error	Bool	false	FALSE
8	status	Word	16#0	16#7000
9	memErrStatus	Word	16#0	16#0000

Network 2 (TCON):

When TCON signals an error ("TCON.error" is "TRUE"), the reported status ("TCON.status") is stored permanently ("TCON.memErrStatus").

Network 3 (TSEND):

Based on the input parameter ID ("connectionID"), the "TSEND" instruction knows the communication connection to be used.

When the input parameter REQ ("TSEND.start") supplies the signal state "TRUE", the "TSEND" instruction is started. For multiple calls, the "TSEND" instruction transmits the data record detected at input parameter DATA ("SLI_plcDB_sendData_TSEND").

Successful transfer of the data record is indicated by the output parameter DONE ("#doneSEND") with the signal state "TRUE" and the output parameter STATUS ("TSEND.status") with the value "0000". Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "TSEND.done".

The output parameter ERROR ("TSEND.error") or the tag "TSEND.memErrStatus" indicates that processing in the example is taking place without errors.

10	TSEND	Struct		
11	start	Bool	false	TRUE
12	maxLength	UInt	0	0
13	busy	Bool	false	FALSE
14	done	Bool	false	TRUE
15	error	Bool	false	FALSE
16	status	Word	16#0	16#7000
17	memErrStatus	Word	16#0	16#0000

Network 4 (TSEND):

When TSEND signals an error ("TSEND.error" is "TRUE"), the reported status ("TSEND.status") is stored permanently ("TSEND.memErrStatus").

Network 5 (TDISCON):

When the input parameter REQ ("TDISCON.start") supplies the signal state "TRUE", the "TDISCON" instruction is started. For multiple calls, the instruction "TDISCON" terminates the communication connection. The communication connection used at the input parameter ID ("connectionID") is retrieved for this purpose.

Successful connection termination is indicated by the output parameter DONE ("#doneDISC") with the signal state "TRUE" and the output parameter STATUS ("TDISCON.status") with the value "0000". Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "TDISCON.done". Reset the success messages for connection setup ("TCON.done") and for data transfer ("TSEND.done").

The output parameter ERROR ("TDISCON.error") or the tag "TDISCON.memErrStatus" indicates that processing in the example is taking place without errors.

18	TDISCON	Struct		
19	start	Bool	false	TRUE
20	busy	Bool	false	FALSE
21	done	Bool	false	TRUE
22	error	Bool	false	FALSE
23	status	Word	16#0	16#7000
24	memErrStatus	Word	16#0	16#0000

Network 6 (TDISCON):

When TDISCON signals an error ("TDISCON.error" is "TRUE"), the reported status ("TDISCON.status") is stored permanently ("TDISCON.memErrStatus").

Behavior of CPU 2

For TCON and TDISCON, the behavior is the same as that for CPU 1.

3	TCON	Struct		
4	start	Bool	false	TRUE
5	busy	Bool	false	FALSE
6	done	Bool	false	TRUE
7	error	Bool	false	FALSE
8	status	Word	16#0	16#7000
9	memErrStatus	Word	16#0	16#0000

18	TDISCON	Struct		
19	start	Bool	false	TRUE
20	busy	Bool	false	FALSE
21	done	Bool	false	TRUE
22	error	Bool	false	FALSE
23	status	Word	16#0	16#7000
24	memErrStatus	Word	16#0	16#0000

Network 3 (TRCV):

Based on the input parameter ID ("connectionID"), the "TRCV" instruction knows the communication connection to be used.

When the input parameter EN_R ("TRCV.start") supplies the signal state "TRUE", the "TRCV" instruction is started. For multiple calls, the "TRCV" instruction receives the transmitted data record. The data record is detected at the input parameter DATA ("SLI_plcDB_rcvData_TRCV").

SLI_plcDB_rcvData_TRCV				
	Name	Data type	Start value	Monitor value
1	Static			
2	prename	String	''	'Simon'
3	lastname	String	''	'Sample'
4	userId	UInt	0	1

The length in BYTE of the actually transmitted data record is detected with the output parameter LEN ("#length"). This value is only displayed during the success status. Then "0" is detected. Successful receipt of the data record is indicated by the output parameter DONE ("#doneRCV") with the signal state "TRUE" and the output parameter STATUS ("TRCV.status") with the value "0000".

Because the values of the output parameters are only displayed for the moment of their validity, do the following:

- Save the success status in the tag "TRCV.done"
- Save the length in BYTE in the tag "TRCV.readLength".

The output parameter ERROR ("TRCV.error") or the tag "TRCV.memErrStatus" indicates that processing in the example is taking place without errors.

10	TRCV	Struct		
11	start	Bool	false	TRUE
12	busy	Bool	false	TRUE
13	done	Bool	false	TRUE
14	error	Bool	false	FALSE
15	status	Word	16#0	16#7002
16	memErrStatus	Word	16#0	16#0000
17	readLength	UInt	0	514

Network 4 (TRCV):

When TRCV signals an error ("TRCV.error" is "TRUE"), the reported status ("TRCV.status") is stored permanently ("TRCV.memErrStatus").

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

TCON: Establish communication connection (Page 3541)

TDISCON: Terminate communication connection (Page 3548)

TSEND: Send data via communication connection (Page 3553)

TRCV: Receive data via communication connection (Page 3561)

Sending and receiving data via Ethernet (UDP) or FDL (S7-1200, S7-1500)

TUSEND: Sending data (S7-1200, S7-1500)

Description

TUSEND supports data transfer via UDP (for S7-1500-CPUs as of firmware version V2.0 and higher via the integrated PROFINET interfaces also as multicast communication) and via FDL connections of the S7-1500 with CM 1542-5 as of V2.0 with the system data type TCON_FDL.

The "TUSEND" instruction sends data to the communication partner addressed by the ADDR parameter.

WARNING

Data transfer via UDP

When transferring data with UDP according to RFC 768, the data is transferred to the partner without acknowledgement and therefore non-securely. This means that data can be lost without this being indicated by the block.

The risk of undetected transmission errors increases significantly when more than 1472 bytes are transferred.

Note

For sequential send operations to different partners, you only need to adjust the ADDR parameter when calling "TUSEND". You do not need to call the "TCON (Page 3538)" and "TDISCON (Page 3548)" instructions again. The UDP port must be set up at the specific partner in order for this partner to receive data.

Sending more than 1472 bytes

As of firmware version V2.5 of the S7-1500 CPUs, you can send up to 2048 bytes of data over UDP rather than just 1472 bytes if you are using Unicast or Multicast.

If you send more than 1472 bytes, you must check that the recipient supports more than 1472 bytes. If this is not the case, the failure of receipt will not be indicated at the sender end.

Functional description

The TUSEND" instruction works asynchronously, which means its job processing extends over multiple calls. Create a rising edge at the REQ parameter to establish a connection once again.

The output parameters BUSY, DONE, ERROR and STATUS indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current status of "TUSEND" or when the send process is concluded.

BUSY	DONE	ERROR	Description
TRUE	FALSE	FALSE	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TUSEND", make sure the data in the send area remains consistent until the DONE parameter or the ERROR parameter has the value TRUE.

Parameter

The following table shows the parameters of the instruction "TUSEND":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter REQUEST starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the associated parameter ID in the local connection description with the .TCON instruction: Value range: W#16#0001 to W#16#0FFF
LEN	Input	UDINT	I, Q, M, D, L or constant	Number of bytes to be sent with the job Value range for Ethernet/UDP: 1 to 1472 (as of firmware version V2.5 of S7-1500 CPUs for Unicast or Multicast: 1 to 2048) For FDL connections of the CM 1542-5, the maximum length is 240 bytes. In this regard, note the maximum lengths that can be processed by the connection partner.

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without errors. This value is only displayed for one cycle.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet complete. A new job cannot be triggered. • BUSY = 0: Job is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR = 1: An error occurred during processing. STATUS supplies detailed information on the type of error
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Error information
DATA	InOut	VARIANT	I, Q, M, D	Send area, contains address and length The address refers to: • The process image of the inputs • The process image of the outputs • A bit memory • A data block When transferring structures, the structures must be identical at the sending and receiving end.
ADDR	InOut	VARIANT	D	Address information of the communication partner See also: Addressing the remote partner via TADDR_Param (Page 3587) and Addressing the remote partner via TADDR_SEND_QDN and TADDR_RCV_IP (Page 3589) and AUTOHOTSPOT

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Send job completed without error
0	7000	No job processing active
0	7001	Start of job processing, data being sent Note: During this processing phase, the operating system accesses the data in the DATA send area.
0	7002	Intermediate call (REQ irrelevant), job is being processed Note: During this processing phase, the operating system accesses the data in the DATA send area.
1	8085	The LEN parameter has the value "0" or is greater than the highest permitted value.
1	8086	The ID parameter is not in the permitted value range.
0	8088	The LEN parameter is greater than the memory area specified in DATA.

ERROR	STATUS* (W#16#...)	Explanation
1	8089	The parameter ADDR does not point to a data block with the structure TADDR_Param or TADDR_SEND_QDN.
1	80A1	Communication error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. Transmission over this connection is not possible. • The interface is being reinitialized.
1	80B1	You changed the DATA parameter before the current job finished.
1	80A4	IP address (at the ADDR parameter) of the remote connection end point is invalid; it may correspond to the local partner's own IP address.
1	80B3	• The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TSEND (Page 3553)". • Parameter ADDR: Invalid information for port no.
1	80B7	The length of the structure referenced by the parameter ADDR is not 8 bytes.
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection between the user program and the communication layer of the operating system cannot be established at this time. • New parameter settings are being assigned to the interface. • Temporarily no receive resources available at the connection partner. The connection partner is not ready to receive. • The R/H system is in the SYNCUP system state or there is a primary-backup switchover. The connection is closed. Processing of TUSEND is stopped.
1	80C5	• Connection terminated by the communication partner. • LSAP of the remote connection partner is not released
1	80C6	Network error: • Remote partner cannot be reached. • Physical interruption on PROFIBUS
1	80C7	Execution timeout.
1	80D3	The fully qualified domain name could not be resolved. Possible causes: • The DNS server is not reachable, for example, because it has been shut down or the remote port is not reachable. • An error occurred during communication with the DNS server. • The DNS server returned a valid DNS answer, but the answer contained no IPv4 address.
1	80EA	DTLS (UDP) protocol is not supported
1	8xyy	For general error information, refer to Evaluating errors with output parameter RET_VAL (Page 162)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Example

You can find the example here: Program example for TUSEND & TURCV (Page 3590).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

GET_ERR_ID: Get error ID locally (Page 883)

TURCV: Receiving data (S7-1200, S7-1500)

Description

The "TURCV" instruction receives data via UDP (for S7-1500-CPUs as of firmware version V2.0 and higher via the integrated PROFINET interfaces also as multicast communication) and via FDL connections of the S7-1500 with CM 1542-5 as of V2.0. After successful completion of "TURCV", the ADDR parameter will show you the address of the remote partner (the sender).

WARNING

Unsecured data transfer

When transferring data with UDP according to RFC 768, the data is transferred to the remote partner without acknowledgement and is therefore unreliable. This means that data can be lost without this being indicated by the block.

Receipt of more than 1472 bytes

As of firmware version V2.5 of the S7-1500 CPUs, you can receive up to 2048 bytes of data via UDP rather than 1472 bytes as before.

However, if your receiving module supports a max. of 1472 bytes and your sending module transfers more than 1472 bytes, TURCV returns a value of W#16#8085 or W#16#80C9 in the STATUS parameter.

Functional description

"TURCV" is an instruction that works asynchronously, which means its job processing extends over multiple calls. You start the receive job by calling "TURCV" with EN_R = 1.

The output parameters BUSY, DONE, ERROR and STATUS indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 596).

The following table shows the relationship between BUSY, NDR and ERROR. Using this table, you can determine the current status of TURCV or when the receive process is concluded.

BUSY	NDR	ERROR	Description
TRUE	FALSE	FALSE	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed. New data was received.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TURCV", the data in the receive area is only consistent when the NDR parameter has the value TRUE.

Parameter

The following table shows the parameters of the instruction "TURCV":

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Control parameter enabled to receive When EN_R = 1, "TURCV" is ready to receive. The receive job is being processed.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the associated ID parameter in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	UDINT	I, Q, M, D, L or constant	Length of the receive area in bytes: 0 (recommended) or 1 to 1472 (as of firmware version V2.5 of S7-1500 CPUs for Unicast or Multicast: 1 to 2048) Maximum value for CM 1542-5 (FDL): 240
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • NDR = 0: Job not yet started or still running. • NDR = 1: Job successfully completed
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: An error occurred during processing. STATUS supplies detailed information on the type of error
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet complete. A new job cannot be triggered. • BUSY = 0: Job is complete.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Error information
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

Parameter	Declaration	Data type	Memory area	Description
DATA	InOut	VARIANT	I, Q, M, D, L	Receive area The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block When transferring structures, the structures must be identical at the sending and receiving end.
ADDR	InOut	VARIANT	D	Address information of the remote partner See also: Addressing the remote partner via TADDR_Param (Page 3587) and Addressing the remote partner via TADDR_SEND_QDN and TADDR_RCV_IP (Page 3589) and AUTOHOTSPOT

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	New data was accepted. The current length of the received data is shown in RCVD_LEN.
0	7000	Block not ready to receive
0	7001	Block is ready to receive, receive job was activated
0	7002	Intermediate call, receive job being processed Note: During this processing phase, "TURCV" writes data to the receive area. For this reason, an error could result in inconsistent data in the receive area.
1	8085	The LEN parameter is greater than the largest permitted value, or you changed the value of the LEN or DATA parameter since the first call
1	8086	The ID parameter is not in the permitted value range
1	8088	• Receive area is too small • Value in LEN is higher than the receive area specified by DATA
1	8089	The parameter ADDR does not point to a data block with the structure TADDR_Param or TADDR_RCV_IP.
1	80A1	Communication error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. A receive job over this connection is not possible. • New parameter settings are being assigned to the interface.
1	80B1	You changed the DATA parameter before the current job finished.
1	80B3	The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TRCV (Page 3557)".
1	80B7	The length at ADDR parameter does not correspond to 8 bytes.

ERROR	STATUS* (W#16#...)	Explanation
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: • New parameter settings are being assigned to the interface. • The R/H system is in the SYNCUP system state or there is a primary-backup switchover. The connection is closed. Processing of TURCV is stopped.
1	80C5	The remote partner has terminated the connection.
1	80C7	Execution timeout.
1	80C9	With RFC1006 / UDP: The received data is longer than expected (size of receive buffer exceeded).
1	80EA	DTLS (UDP) protocol is not supported
1	8xyy	For general error information, refer to Evaluating errors with output parameter RET_VAL (Page 162)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Example

You can find the example here: Program example for TUSEND & TURCV (Page 3590).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

TCON: Establish communication connection (Page 3538)

TDISCON: Terminate communication connection (Page 3548)

GET_ERR_ID: Get error ID locally (Page 883)

Addressing the remote partner via TADDR_Param (S7-1200, S7-1500)

Overview

When using a UDP connection, you can store the address information for the remote partner in the system data type TADDR_Param:

- With the "TUSEND (Page 3579)" instruction, you transfer the address information of the recipient at the parameter ADDR via TADDR_Param.
  The stored address data of the remote partner are read from the system data type by the instruction.

  Note

  Supporting a limited broadcast

  For S7-1500 CPUs as of firmware version V2.0, the following applies to the instruction "TUSEND": When addressing the remote partner via the "TADDR_Param" system data type, a limited broadcast (255.255.255.255 as the IP address of the remote partner) is supported with consistent compliance to IP routing rules. Accordingly, limited broadcasts are only sent over an interface through which a default router is accessible according to configuration.

  It is recommended to use of directed broadcast addresses because they - according to the IP standards - are not subject to this restriction.

- With the "TURCV (Page 3583)" instruction, you receive the address of the sender at the parameter ADDR via TADDR_Param.
  The address data is written by the instruction in the system data type.

Structure of the address information according to TADDR_Param

The system data type TADDR_Param contains the address information of the remote partner and consists of the IP address and the port number.

The system data type TADDR_Param has the following structure:

Byte	Parameter	Data type	Start value	Description
0 to 3	rem_ip_addr	ARRAY [1..4] of USINT	B#16#00 ...	• IP address of the remote partner, e.g. 192.168.002.003: – rem_ip_addr[1] = B#16#C0 (192) – rem_ip_addr[2] = B#16#A8 (168) – rem_ip_addr[3] = B#16#02 (002) – rem_ip_addr[4] = B#16#03 (003) The IP address can be taken from the Devices & networks view in the interface properties of the remote partner. As an alternative these are also displayed in the properties of the UDP connection under Address details. • Multicast address of an IPv4 multicast group (for the "TUSEND" instruction with S7-1500-CPUs as of firmware version V2.0 and higher. Note: For the "TURCV" instruction, the IP address of the sender must also be specified in case of multicast communication.)
4 to 5	rem_port_nr	UINT	B#16#00 ...	Remote port-number (for possible values, see: AUTOHOT-SPOT): • rem_port_nr[1] = high byte of the port no. in hexadecimal notation • rem_port_nr[2] = low byte of port no. in hexadecimal notation The port number can be taken from the Devices & networks view in the UDP connection properties. The port number is indicated under Address details as a decimal value. Example: Port number = 2000 (decimal) / W#16#07D0 (hexadecimal) • rem_port_nr[1] = 07 (high byte) • rem_port_nr[2] = D0 (low byte)
6 to 7	reserved	WORD	B#16#00 ...	Not used. Leave the value "0" at this parameter.

Creating TADDR_Param in a data block

You have the following options for creating TADDR_Param:

- Create a new data block and select TADDR_Param as type in the dialog "Add new data block".
- Open an existing data block create a new tag and enter in the column Data type TADDR_Param.

A data block can contain several system data types TADDR_Param.

Addressing the remote partner via TADDR_SEND_QDN and TADDR_RCV_IP (S7-1500)

Overview

For S7-1500 CPUs as of firmware version V2.0, you can address the recipient with its fully qualified domain name when sending data via UDP. With the instruction TUSEND at the parameter ADDR, you hereby reference a structure of the type TADDR_SEND_QDN.

The receiver can return an IPv4 or an IPv6 address. With the TURCV instruction at the ADDR parameter, you therefore reference a structure of the type TADDR_RCV_IP. Only this can include both IP address types.

Structure of the address information according to TADDR_SEND_QDN

The system data type TADDR_SEND_QDN contains the address information of the remote partner (of the receiver) and consists of the fully qualified domain name and port number.

It has the following structure:

Byte	Parameter	Data type	Start value	Description
0 to 255	RemoteQDN	ARRAY of STRING[1..254]	-	Fully qualified domain name of the partner end point, which must finish with "." Please note that the name including the concluding period must not exceed 254 characters in a SIMATIC environment.
256 to 257	RemotePort	UINT	B#16#00 ...	Remote port-number (for possible values, see: AUTOHOT-SPOT): • RemotePort[1] = high byte of the port no. in hexadecimal notation • RemotePort[2] = low byte of port no. in hexadecimal notation Example: Port number = 2000 (decimal) / W#16#07D0 (hexadecimal) • RemotePort[1] = 07 (high byte) • RemotePort[2] = D0 (low byte)
258 to 259	reserved	WORD	W#16#00	Not used. Leave the value "0" at this parameter.

Note

Runtime of the instruction TUSEND

With each call of TUSEND, the fully qualified domain name is resolved in the IP address. As a result, the execution time of TUSEND is longer than if you address the receiver directly via its IPv4 address.

Structure of the address information according to TADDR_RCV_IP

The system data type TADDR_RCV_IP contains the address information of the remote partner (of the sender) and consists of the IP address, the port number and the information on whether the IP address is an IPv4 or IPv6 address.

It has the following structure:

Byte	Parameter	Data type	Start value	Description
0 to 15	RemoteAd-dress	ARRAY [1..16] of BYTE	B#16#00 ...	IP address of the remote partner, e.g. for the IPv4 address 192.168.002.003: • RemoteAddress[1] = B#16#C0 (192) • RemoteAddress[2] = B#16#A8 (168) • RemoteAddress[3] = B#16#02 (002) • RemoteAddress[4] = B#16#03 (003) The following applies to the IPv6 address.
16 to 17	RemotePort	UINT	B#16#00 ...	Remote port-number (for possible values, see: AUTOHOT-SPOT): • RemotePort[1] = high byte of the port no. in hexadecimal notation • RemotePort[2] = low byte of port no. in hexadecimal notation Example: Port number = 2000 (decimal) / W#16#07D0 (hex-adecimal) • RemotePort[1] = 07 (high byte) • RemotePort[2] = D0 (low byte)
18	RemAddr-Type	BYTE	B#16#00	Type of the remote address: • B#16#03: IPv4 • B#16#04: IPv6
19	reserved	BYTE	B#16#00	Not used. Leave the value "0" at this parameter.

Program example for TUSEND & TURCV (S7-1200, S7-1500)

Introduction

In the following example you create a configured UDP connection between two CPUs of the S7-1500 series. Transmit a data record from CPU1 to CPU2 using the instructions "TUSEND" and "TURCV".

Requirement

- Two CPUs of the S7-1500 series are available and connected to each other over PROFINET. A UDP connection is configured.

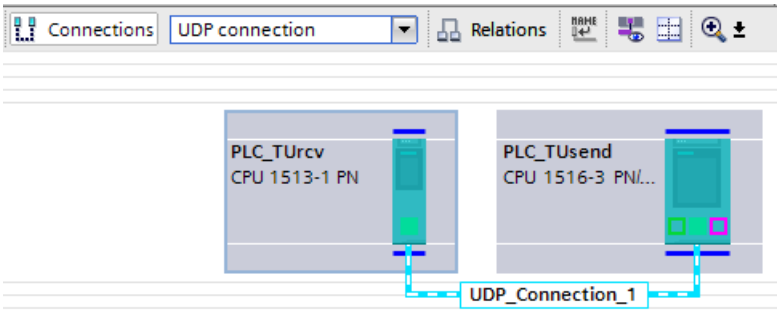

- A low protection level under "<CPU> > Properties > Protection" ensures that read and write access is permitted for the CPUs.

Program of CPU 1

For the data record, create the following PLC data type "TUSEND_User".

TUSEND_User

	Name	Data type	Default value
1	prename	String	''
2	lastname	String	''
3	userId	UInt	0

For the data transfer, create the following data block ("SLI_plcDB_sendData_TUSEND") based on the PLC data type "TUSEND_User".

SLI_plcDB_sendData_TUSEND

	Name	Data type	Start value
1	Static		
2	prename	String	'Simon'
3	lastname	String	'Sample'
4	userId	UInt	1

For addressing the communication partner, create the following data block ("SLI_plcDB_taddr_param_TUSEND") based on the system data type "TADDR_Param.

SLI_plcDB_taddr_param_TUSEND

	Name	Data type	Start value
1	Static		
2	REM_IP_ADDR	Array[1..4] of USInt	
3	REM_IP_ADDR[1]	USInt	192
4	REM_IP_ADDR[2]	USInt	168
5	REM_IP_ADDR[3]	USInt	0
6	REM_IP_ADDR[4]	USInt	3
7	REM_PORT_NR	UInt	2000
8	RESERVED	Word	16#0

You can find the address date in the "Network view" under "Connections > Properties > Address details". The data of the CPU that had been selected previously are specified in the address details under "Local". In the following figure, CPU 1 "TUsend") is selected.

	Local	Partner
IP (dec):	192.168.0.2	192.168.0.3
Port (dec):	2000	2000
	Remote address assignment in block	Remote address assignment in block

To store the data, create a global data block ("SLI_gDB_TUSEND") with the following structures and tags.

SLI_gDB_TUSEND

	Name	Data type	Start value
1	Static		
2	connectionID	CONN_OUC	16#0100
3	TUSEND	Struct	
4	start	Bool	false
5	maxLength	UDInt	0
6	busy	Bool	false
7	done	Bool	false
8	error	Bool	false
9	status	Word	16#0
10	memErrStat	Word	16#0

Create the FB "SLI_FB_TUSEND" Create the following local tags in it.

7	Static			
8	doneSEND	Bool	false	Non-ret...
9	TUSEND_Instance	TUSEND		

Network 1: Interconnect the parameter of the instruction "TUSEND" as follows:

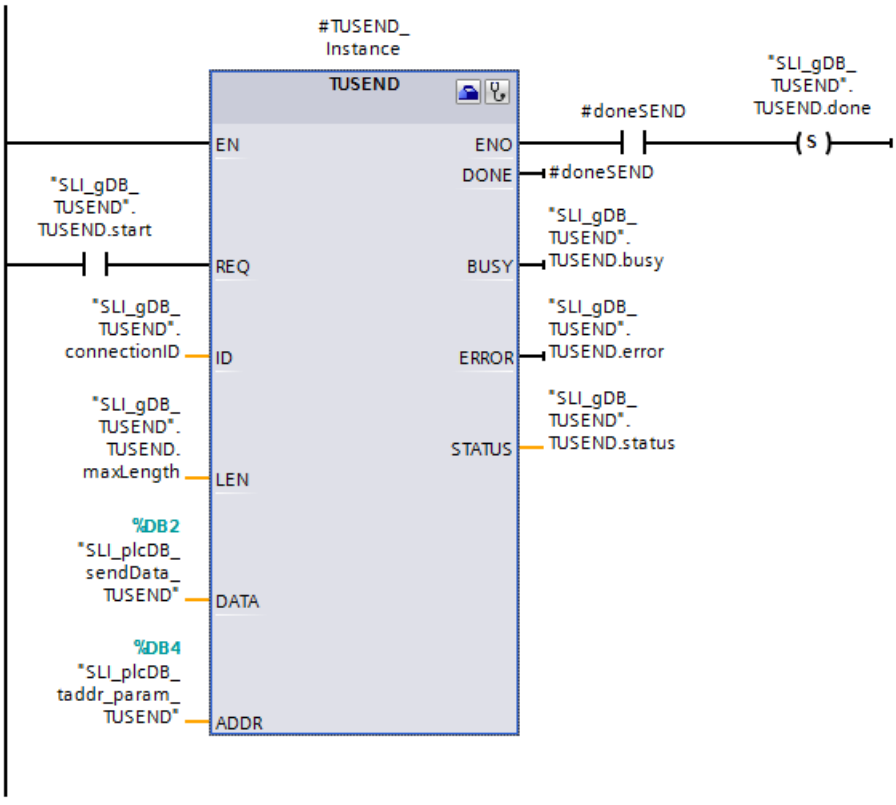

Network 2: In case of error of TUSEND, save the status as follows.

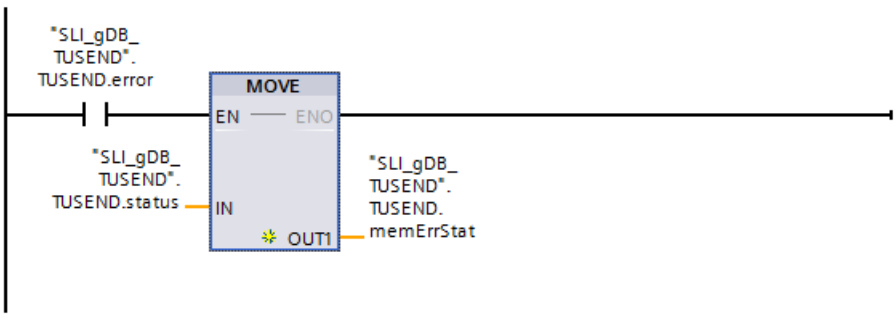

Program of CPU 2

For the data record, create the PLC data type "TUSEND_User" based on the example of CPU 1.

For receiving data, create the following data block ("SLI_plcDB_rcvData_TURCV") based on the created PLC data type ("TUSEND_User").

SLI_plcDB_rcvData_TURCV			
	Name	Data type	Start value
1	Static		
2	prename	String	''
3	lastname	String	''
4	userId	UInt	0

For addressing the communication partner, create the following data block ("SLI_plcDB_taddr_param_TURCV") based on the system data type "TADDR_Param.

SLI_plcDB_taddr_param_TURCV			
	Name	Data type	Start value
1	Static		
2	REM_IP_ADDR	Array[1..4] of USInt	
3	REM_IP_ADDR[1]	USInt	192
4	REM_IP_ADDR[2]	USInt	168
5	REM_IP_ADDR[3]	USInt	0
6	REM_IP_ADDR[4]	USInt	2
7	REM_PORT_NR	UInt	2000
8	RESERVED	Word	16#0

To store the data, create a global data block ("SLI_gDB_TURCV") with the following structures and tags.

SLI_gDB_TURCV			
	Name	Data type	Start value
1	Static		
2	connectionID	CONN_OUC	16#0100
3	TURCV	Struct	
4	start	Bool	false
5	busy	Bool	false
6	done	Bool	false
7	error	Bool	false
8	status	Word	16#0
9	memErrStat	Word	16#0
10	readLength	UDInt	0

Create the FB "SLI_FB_TURCV" Create the following local tags in it.

7	Static			
8	doneRCV	Bool	false	Non-ret...
9	length	UDInt	0	Non-retain
10	TURCV_Instance	TURCV		

Network 1: Interconnect the parameter of the instruction "TURCV" as follows:

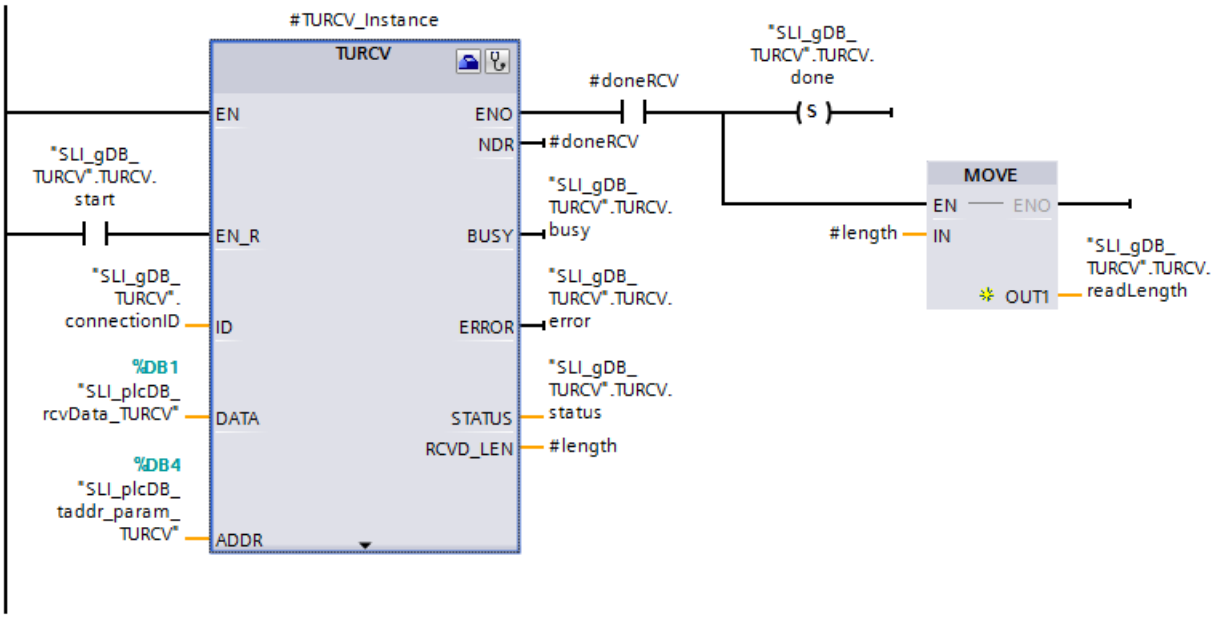

Network 2: In case of error of TURCV, save the status as follows.

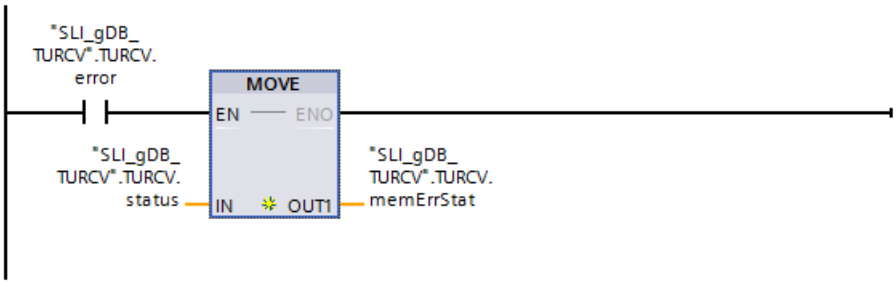

Assigning communication connection

The address parameters for the connection of the two CPUs must be uniform.

- In each case store the hexadecimal value of the hardware identifier of the configured UDP connection at the input parameter ID ("connectionID").
  You can find the hardware identifier in the "Network view" under "Connections".

ork overview | Connections | Relations | I/O communication | VPN

Local connection name	Local end point	Local ID (hex)	Partner ID (hex)	Partner
UDP_Connection_1	PLC_TUrcv	100	100	PLC_TUsend
UDP_Connection_1	PLC_TUsend	100	100	PLC_TUrcv

Result of CPU 1

Network 1 (TUSEND):

Based on the input parameter ID ("connectionID"), the "TUSEND" instruction knows the communication connection to be used. Based on the input parameter ADDR

("SLI_plcDB_taddr_param_TUSEND"), the communication partner (CPU 2) can be addressed.

When the input parameter REQ ("TUSEND.start") supplies the signal state "TRUE", the "TUSEND" instruction is started. For multiple calls, the "TUSEND" instruction transmits the data record detected at input parameter DATA ("SLI_plcDB_sendData_TUSEND"). The size of the data record to be transmitted is restricted by the value of the input parameter LEN ("TUSEND.maxLength").

Successful transfer of the data record is indicated by the output parameter DONE ("#doneSEND") with the signal state "TRUE" and the output parameter STATUS ("TUSEND.status") with the value "0000". Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "TUSEND.done".

The output parameter ERROR ("TUSEND.error") or the tag "TUSEND.memErrStatus" indicates that processing in the example is taking place without errors.

SLI_gDB_TUSEND

	Name	Data type	Start value	Monitor value
1	Static			
2	connectionID	CONN_OUC	16#0100	16#0100
3	TUSEND	Struct		
4	start	Bool	false	TRUE
5	maxLength	UDInt	0	0
6	busy	Bool	false	FALSE
7	done	Bool	false	TRUE
8	error	Bool	false	FALSE
9	status	Word	16#0	16#7000
10	memErrStat	Word	16#0	16#0000

Network 2 (TUSEND):

When TUSEND signals an error ("TUSEND.error" is "TRUE"), the reported status ("TUSEND.status") is stored permanently ("TUSEND.memErrStatus").

Result of CPU 2

Network 1 (TURCV):

Based on the input parameter ID ("connectionID"), the "TURCV" instruction knows the communication connection to be used. The communication partner (CPU 1) can be addressed based on the input parameter ADDR ("SLI_plcDB_taddr_param_TURCV").

When the input parameter EN_R ("TURCV.start") supplies the signal state "TRUE", the "TURCV" instruction is started. For multiple calls, the "TURCV" instruction receives the transmitted data record. The data record is detected at the input parameter DATA ("SLI_plcDB_rcvData_TURCV").

SLI_plcDB_rcvData_TURCV

	Name	Data type	Start value	Monitor value
1	Static			
2	prename	String	''	'Simon'
3	lastname	String	''	'Sample'
4	userId	UInt	0	1

The length in BYTES of the actually transmitted data record is detected with the output parameter LEN ("#length"). This value is only displayed during the success status. Then "0" is detected. Successful receipt of the data record is indicated by the output parameter DONE ("#doneRCV") with the signal state "TRUE" and the output parameter STATUS ("TURCV.status") with the value "0000".

Because the values of the output parameters are only displayed for the moment of their validity, do the following:

- Save the success status in the tag "TURCV.done"
- Save the length in BYTES in the tag "TURCV.readLength".

The output parameter ERROR ("TURCV.error") or the tag "TURCV.memErrStatus" indicates that processing in the example is taking place without errors.

SLI_gDB_TURCV

	Name	Data type	Start value	Monitor value
1	Static			
2	connectionID	CONN_OUC	16#0100	16#0100
3	TURCV	Struct		
4	start	Bool	false	TRUE
5	busy	Bool	false	TRUE
6	done	Bool	false	TRUE
7	error	Bool	false	FALSE
8	status	Word	16#0	16#7002
9	memErrStat	Word	16#0	16#0000
10	readLength	UDInt	0	514

Network 2 (TURCV):

When TURCV signals an error ("TURCV.error" is "TRUE"), the reported status ("TURCV.status") is stored permanently ("TURCV.memErrStatus").

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

TUSEND: Sending data (Page 3579)

TURCV: Receiving data (Page 3583)

T_RESET: Resetting the connection (S7-1200, S7-1500)

Description

The "T_RESET" instruction terminates and then reestablishes an existing connection.

The local end points of the connection are retained. They are generated automatically:

- If a connection has been configured and loaded to the CPU.
- If a connection has been generated by the user program, for example, by calling the instruction "TCON (Page 3538)".

The instruction "T_RESET" can be executed for all connection types (TCP, UDP, ISO-on-TCP, etc.). It does not matter whether the local interface of the CPU or the interface of a CM/CP was used for the connection.

Once the instruction "T_RESET" has been called using the REQ parameter, the connection specified with the ID parameter is terminated and, if necessary, the data send and receive buffer cleared. Canceling the connection also cancels any data transfer in progress. There is therefore a risk of losing data if data transfer is in progress. The CPU defined as the active connection partner will then automatically attempt to restore the interrupted communication connection. You therefore do not need to call the "TCON (Page 3538)" instruction to reestablish the communication connection.

The output parameters DONE, BUSY and STATUS indicate the status of the job.

Parameter

The following table shows the parameters of the instruction "T_RESET":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter REQUEST starts the job for terminating the connection specified by ID. The job starts on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection to the passive partner ID being interrupted must be identical to the corresponding parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter BUSY • 0: Job is complete. • 1: The job is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR • 0: No error occurred. • 1: Error occurred during processing. The STATUS parameter supplies detailed information on the type of error
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS Error information (see "STATUS parameter" table).

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

STATUS* (W#16#...)	Explanation
0000	No error.
0001	Connection has not been established.
7000	No job processing active.
7001	Connection termination launched.
7002	Connection being terminated.
8081	Unknown connection specified at the ID parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Example

You can find the example here: Program example for T_RESET (Page 3599).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Switching display formats in the program status (Page 7698)

Difference between synchronous and asynchronous instructions (Page 596)

Program example for T_RESET

Introduction

In the following example you use a configured connection (by UDP for example) between two CPUs (S7-1500). Reset the connection using instruction "T_RESET".

Requirement

- Two CPUs of the S7-1500 series are available and connected to each other over PROFINET. A UDP connection is configured.

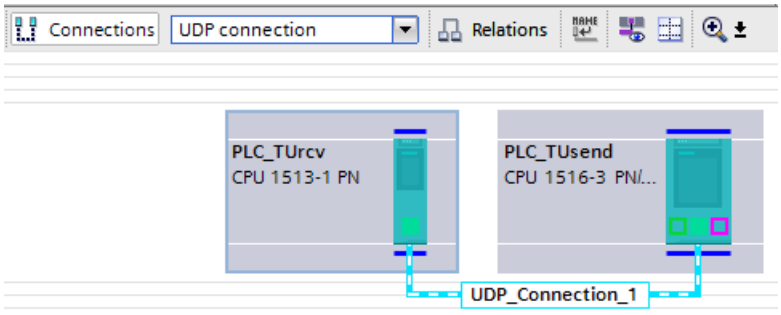

- A low protection level under "<CPU> > Properties > Protection" ensures that read and write access is permitted for the CPUs.

Create tags and interconnect parameters (program of CPU 1)

To store the data, create a global data block ("SLI_gDB_T_RESET") with the following structures and tags.

SLI_gDB_T_RESET

	Name	Data type	Start value
1	Static		
2	connectionID	CONN_OUC	16#0100
3	T_RESET	Struct	
4	start	Bool	false
5	busy	Bool	false
6	done	Bool	false
7	error	Bool	false
8	status	Word	16#0
9	memErrStatus	Word	16#0

Create the FB "SLI_FB_T_RESET" Create the following local tags in it.

7	Static			
8	done	Bool	false	Non-ret...
9	T_RESET_Instance	T_RESET		

Network 1: Interconnect the parameter of the instruction "T_RESET" as follows:

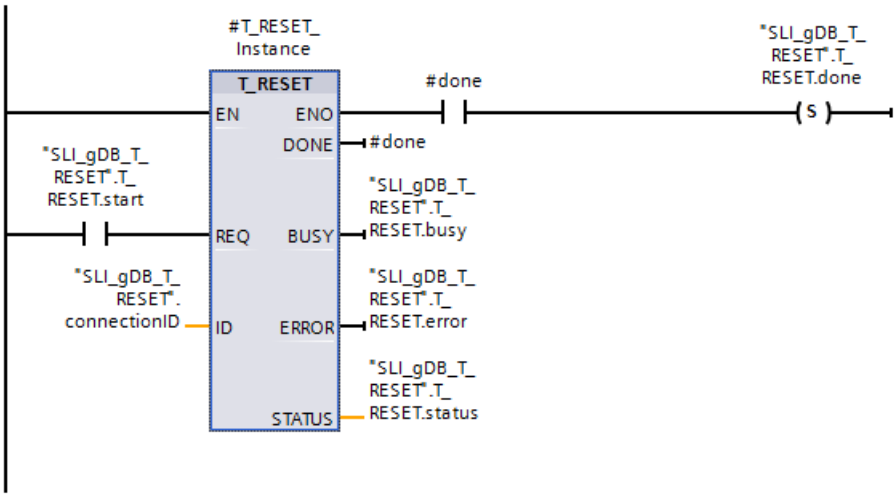

Network 2: In case of error of T_RESET, save the status as follows.

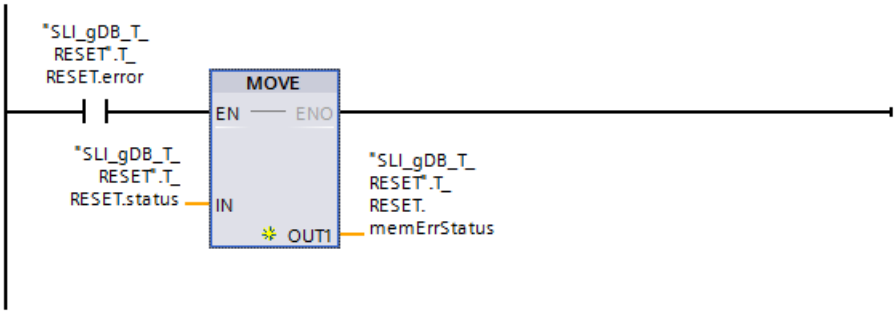

Network 3: To reset the "REQ" parameter, create the following interconnections.

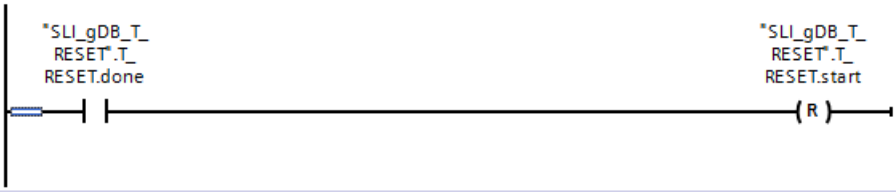

Assigning communication connection

The addressing parameters must be adapted for the connection.

- Store the hexadecimal value of the hardware identifier of the configured UDP connection at the input parameter ID ("connectionID").
  You can find the hardware identifier in the "Network view" under "Connections".

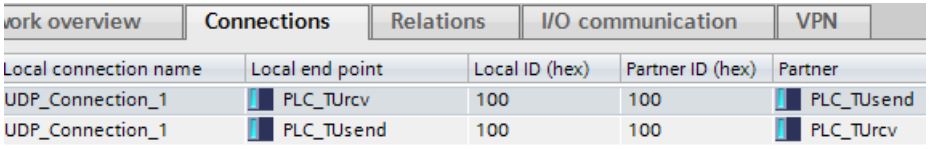

ork overview	Connections	Relations	I/O communication	VPN

Local connection name	Local end point	Local ID (hex)	Partner ID (hex)	Partner
UDP_Connection_1	PLC_TUrcv	100	100	PLC_TUsend
UDP_Connection_1	PLC_TUsend	100	100	PLC_TUrcv

Result

Network 1:

Based on the input parameter ID ("connectionID"), the "T_RESET" instruction knows the communication connection to be used.

When the input parameter REQ ("T_RESET.start") supplies the signal state "TRUE", the "T_RESET" instruction is started. The "T_RESET" instruction terminates the specified communication connection. All jobs that use the connection are terminated. The active connection partner (CPU 1) then re-establishes the connection automatically.

Successful transfer of the data record is indicated by the output parameter DONE ("#done") with the signal state "TRUE" and the output parameter STATUS ("T_RESET.status") with the value "0000".
Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "T_RESET.done".

The output parameter ERROR ("T_RESET.error") or the tag "T_RESET.memErrStatus" indicates that processing in the example is taking place without errors.

SLI_gDB_T_RESET

	Name	Data type	Start value	Monitor value
1	Static			
2	connectionID	CONN_OUC	16#0100	16#0100
3	T_RESET	Struct		
4	start	Bool	false	FALSE
5	busy	Bool	false	FALSE
6	done	Bool	false	TRUE
7	error	Bool	false	FALSE
8	status	Word	16#0	16#7000
9	memErrStatus	Word	16#0	16#0000

Network 2:

When T_RESET signals an error ("T_RESET.error" is "TRUE"), the reported status ("T_RESET.status") is stored permanently ("T_RESET.memErrStatus").

Network 3:

After the successful processing of T_RESET ("T_RESET.done"), T_RESET is stopped.

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

T_RESET: Resetting the connection (Page 3597)

T_DIAG: Checking the connection (S7-1200, S7-1500)

Description

You use the "T_DIAG" instruction to check the status of a connection and read further information on the local end point of this connection.

- The connection is referenced by the ID parameter. You can read both connection end points configured in the connection editor and programmed connection end points (e.g. with the "TCON" instruction).
  Temporary connection end points (for example end points created when you connect to an engineering station) cannot be diagnosed, as no connection ID is generated in this process.
- The connection information read is stored in a structure referenced by the RESULT parameter.
- The output parameter STATUS indicates whether it was possible to read the connection information. The connection information in the structure at the RESULT parameter is only valid if the "T_DIAG" instruction has been completed with STATUS = W#16#0000 and ERROR = FALSE.
  Connection information cannot be evaluated if an error occurs.

Possible connection information

Two different structures can be used to read the connection information at the RESULT parameter:

- The "TDiag_Status" structure only contains the most important information about a connection end point, for example the protocol used, the connection status and the number of data bytes sent or received.
- The structure "TDiag_StatusExt" (with S7-1500 CPUs only) supplies not just the most important information, but also the number of connection attempts, the reason for a connection abort, etc.

The structure and parameters of the two structures are described below (see the "TDIAG_Status and TDIAG_StatusExt structures" table).

Parameters

The following table shows the parameters of the instruction "T_DIAG":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the instruction to check the connection specified in the ID parameter when there is a positive edge.
ID	Input	CONN_OUC (WORD)	I, Q, M, D, L or constant	Reference to the assigned connection. Value range: W#16#0001 to W#16#0FFF

Parameter	Declaration	Data type	Memory area	Description
RESULT	InOut	VARIANT	D	Pointer to the structure in which the connection information is stored. The structures TDiag_Status or TDiag_StatusExt (with S7-1500 CPUs only) can be used at the RESULT parameter (for a description, see the "TDIAG_Status and TDIAG_StatusExt structures" table).
DONE	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Instruction not yet started or still in progress. • 1: Instruction executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Instruction not yet started or already completed. • 1: Instruction not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error. • 1: Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

BUSY, DONE and ERROR parameters

You can check the status of "T_DIAG" instruction execution with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not an instruction has been executed successfully. The ERROR parameter is set if errors occur during execution of "T_DIAG".

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The instruction is being processed.
0	1	0	The instruction has been executed successfully. The data in the structure referenced by RESULT are only valid if this is the case.
0	0	1	Instruction completed with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new instruction has been assigned.

Parameter STATUS

The following table shows the meaning of the values at the STATUS parameter:

STATUS* (W#16#...)	Explanation
0000	The instruction "T_DIAG" has been executed successfully. The data in the structure referenced at the RESULT parameter can be evaluated.
7000	No instruction processing active.
7001	Instruction processing launched.
7002	Connection information is being read (REQ parameter irrelevant).
8086	The value at the ID parameter is outside the valid range (W#16#0001 ... W#16#0FFF).
8089	The RESULT parameter points to an invalid data type (structures TDIAG_Status and TDIAG_StatusExt only).
80A3	The ID parameter references a connection end point which does not exist. With programmed connections, this error can also occur after the "TDISCON" instruction is called.
80B1	The RESULT parameter was changed before the execution of T_DIAG was completed. A change of RESULT is not permitted during execution of T_DIAG.
80C4	Internal error. Access to the connection end point is temporarily unavailable.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Structures TDIAG_Status and TDIAG_StatusExt

Note

TDIAG_StatusExt structure

The TDIAG_StatusExt structure only exists with S7-1500 CPUs.

The table below details the form of the TDIAG_Status and TDIAG_StatusExt structures:

- From the InterfaceID to the ReceivedBytes parameter, the "TDIAG_StatusExt" and "TDIAG_Status" structures are identical.
- The structure TDIAG_StatusExt also includes the parameters ConnTrials to LastDisconnTimeStamp.

The value of each element is only valid if the instruction has been executed without errors. If an error occurs, the content of the parameters will not change.

Name	Data type	Description
The following parameters are in both the TDIAG_Status and the TDIAG_StatusExt structure:		
InterfaceID	HW_ANY	Interface ID (LADDR) of the CPU or the CM/CP.
ID	CONN_OUC	ID of the connection diagnosed. Following a successful call, the value of this element is identical to the parameter ID of the "T_DIAG" instruction.

Name	Data type	Description
ConnectionType	BYTE	Protocol type used for connection: • 0x01: Not used. • ... • 0x0B: TCP protocol (IP_v4) • 0x0C: ISO-on-TCP protocol (RFC1006) • 0x0D: TCP protocol (DNS) • 0x0E: Dial-in protocol • 0x0F: WDC protocol • 0x10: SMTP protocol • 0x11: TCP protocol • 0x12: TCP and ISO-on-TCP protocol (RFC1006) • 0x13: UDP protocol • 0x14: Reserved • 0x15: PROFIBUS bus access protocol (FDL) • 0x16: ISO 8073 transport protocol (ISO native) • ... • 0x20: SMTP or SMTPS protocol - based on IPv4 • 0x21 SMTP or SMTPS protocol - based on IPv6 • 0x22: SMTP or SMTPS protocol based on FQDN (**F**ully **Q**ualified **D**omain **N**ame) • ... • 0x70: S7 connection • Other: Reserved
ActiveEstablished	BOOL	• FALSE: Locally, the passive connection end point • TRUE: Locally, the active connection end point
State	BYTE	Current status of the connection end point • 0x00: Not used. • 0x01: Connection terminated. Temporary status, for example, after the "T_RESET" instruction is called. The system then automatically attempts to reestablish the connection. • 0x02: The active connection end point is attempting to establish a connection to the remote communication partner. • 0x03: The passive connection end point is waiting for establishment of the connection to the remote communication partner. • 0x04: Connection established. • 0x05: The connection is being terminated. This may be because the "T_RESET" or "T_DISCON" instruction has been called. Other possible reasons are protocol errors and line breaks. • 0x06..0xFF: Not used.

Name	Data type	Description
Kind	BYTE	Mode of the connection end point: • 0x00: Not used. • 0x01: Configured, static connection which has been configured and loaded to the CPU. • 0x02: Configured, dynamic connection which has been configured and loaded to the CPU (not currently supported). • 0x03: Programmed connection generated in the user program with the instruction "TCON". A call of the instruction "TDISCON" or a transition to CPU STOP status has destroyed the connection end point. • 0x04: Temporary, dynamic connection established by the engineering station (ES) or operator station (OS) (this connection currently not be diagnosed due to the missing ID). • 0x05..0xFF: Not used.
SentBytes	UDINT	Number of data bytes sent.
ReceivedBytes	UDINT	Number of data bytes received.
The following parameters only occur in the TDiag_StatusExt structure:		
ConnTrials	UDINT	Number of connection attempts. Once a connection has been successfully established, ConnTrials has the value 0. If this element is not equal to 0, this indicates that there are connection problems. Note: With a passive connection end point, this value is never greater than 1.
ConnTrialsSuccess	UDINT	Number of successful connection attempts. This element is never reset during the life cycle of a connection end point, and returns to 0 after reaching 0xFFFF FFFF. Note: This parameter is 1 if there has never been a problem in this connection.
LastConnErrReason	UDINT	Error ID output during the last connection attempt with errors (the error messages are identical to those at the LastDisconnReason parameter): • 0x4F01: Remote connection end point cannot be reached (this error usually occurs during connection establishment). • 0x4F02: Connection terminated locally. • 0x4F03: Connection terminated by remote communication partner. • 0x4F04: Connection terminated by a protocol error. • 0x4F05: Connection terminated by a network problem detected locally. • 0x4F06: Connection terminated by a network problem detected remotely. • 0x4F07: Connected terminated due to timeout in protocol. • 0x4F08: Incorrect parameter assignment: Connection to be established to local partner's own address. • 0x4F09: Connection temporarily reset by a call of the instruction "T_RESET". • 0x4F0A: Insufficient connection resources available (quantity exceeded) • 0x4F0B: Internal error: Incorrect addressing parameters • 0x4F0C: Internal CPU communication error • 0x4F0D: Internal AS communication error between CPU and CM/CP • 0x4F0E: The local TCP/UDP port (or RFC1006-T selector) specified is already in use.
LastConnErrTimeStamp	LDT	Time of the last connection attempt with errors.

Name	Data type	Description
LastDisconnReason	UDINT	Error ID which led to the last connection termination (the error messages are identical to those at the LastConnErrReason parameter): • 0x4F01: Remote connection end point cannot be reached (this error usually occurs during connection establishment). • 0x4F02: Connection terminated locally. • 0x4F03: Connection terminated by remote communication partner. • 0x4F04: Connection terminated by a protocol error. • 0x4F05: Connection terminated by a network problem detected locally. • 0x4F06: Connection terminated by a network problem detected remotely. • 0x4F07: Connected terminated due to timeout in protocol. • 0x4F08: Incorrect parameter assignment: Connection to be established to local partner's own address. • 0x4F09: Connection temporarily reset by a call of the instruction "T_RESET". • 0x4F0A: Insufficient connection resources available (quantity exceeded) • 0x4F0B: Internal error: Incorrect addressing parameters • 0x4F0C: Internal CPU communication error • 0x4F0D: Internal AS communication error between CPU and CM/CP • 0x4F0E: The local TCP/UDP port (or RFC1006-T selector) specified is already in use.
LastDisconnTimeStamp	LDT	Time of the last connection termination.

Example

You can find the example here: Program example for T_DIAG (Page 3608).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Difference between synchronous and asynchronous instructions (Page 596)

Program example for T_DIAG

Introduction

In the following example you use a configured connection (by UDP for example) between two CPUs (S7-1500). The instruction "T_DIAG" is used to diagnose the connection and read information from the local end point of the connection.

Requirement

- Two CPUs of the S7-1500 series are available and connected to each other over PROFINET. A UDP connection is configured.

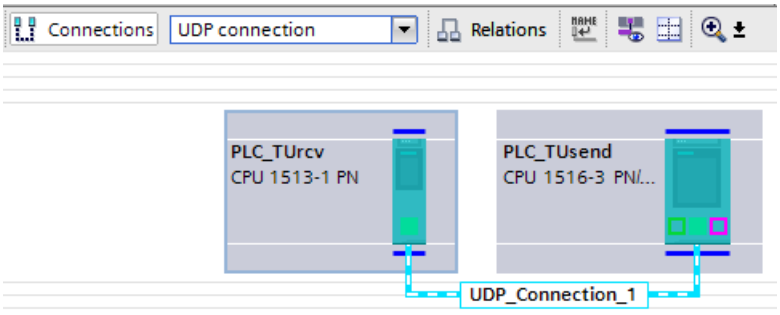

- A low protection level under "<CPU> > Properties > Protection" ensures that read and write access is permitted for the CPUs.

Create tags and interconnect parameters (program of CPU 1)

To store the data, create a global data block ("SLI_gDB_T_DIAG") with the following structures and tags.

Note: "TDiag_StatusExt" is a system data type for recorded the diagnostics data.

SLI_gDB_T_DIAG

	Name	Data type	Start value
1	Static		
2	connectionID	CONN_OUC	16#0100
3	T_DIAG	Struct	
4	start	Bool	false
5	busy	Bool	false
6	done	Bool	false
7	error	Bool	false
8	status	Word	16#0
9	memErrStatus	Word	16#0
10	connectionDiag	TDiag_StatusExt	

Create the FB "SLI_FB_T_DIAG" Create the following local tags in it.

7	Static			
8	T_DIAG_Instance	T_DIAG		
9	done	Bool	false	Non-retain

Network 1: Interconnect the parameter of the instruction "T_DIAG" as follows:

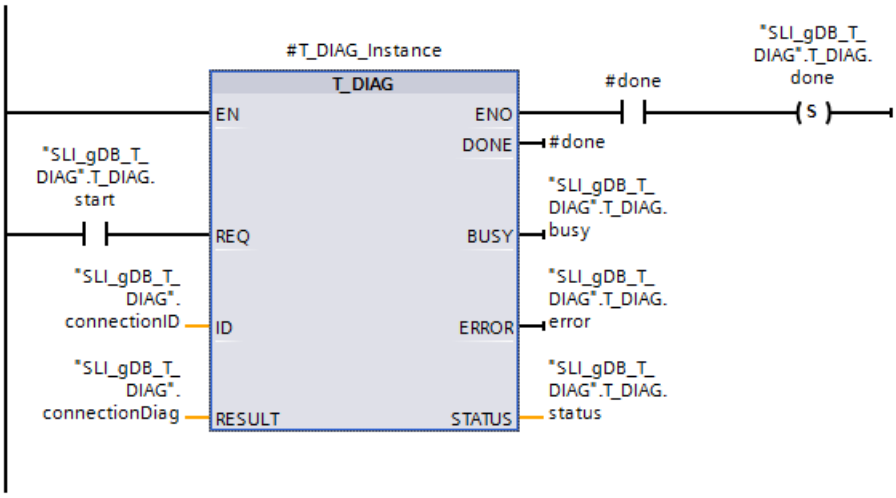

Network 2: In case of error of T_DIAG, save the status as follows.

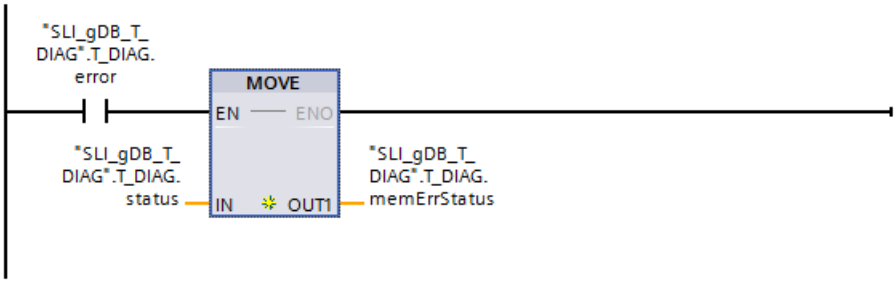

Assigning communication connection

The addressing parameters must be adapted for the connection.

- Store the hexadecimal value of the hardware identifier of the configured UDP connection at the input parameter ID ("connectionID").
  You can find the hardware identifier in the "Network view" under "Connections".

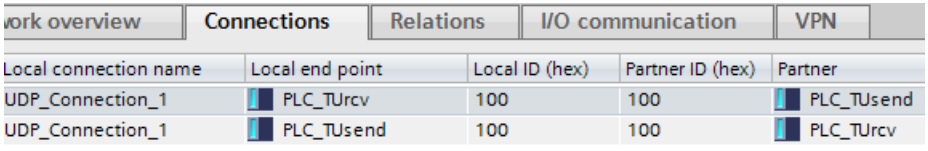

ork overview	Connections	Relations	I/O communication	VPN

Local connection name	Local end point	Local ID (hex)	Partner ID (hex)	Partner
UDP_Connection_1	PLC_TUrcv	100	100	PLC_TUsend
UDP_Connection_1	PLC_TUsend	100	100	PLC_TUrcv

Result

Network 1:

Based on the input parameter ID ("connectionID"), the "T_DIAG" instruction knows the communication connection to be used.

When the input parameter REQ ("T_DIAG.start") supplies the signal state "TRUE", the "T_DIAG" instruction is started. The instruction "T_DIAG" reads the status of the connection and the status of the local end point of the connection.

Successful transfer of the data record is indicated by the output parameter DONE ("#done") with the signal state "TRUE" and the output parameter STATUS ("T_DIAG.status") with the value "0000". Because the values of the output parameters are only displayed for the moment of their validity, save the success status in the tag "T_DIAG.done".

The output parameter ERROR ("T_DIAG.error") or the tag "T_DIAG.memErrStatus" indicates that processing in the example is taking place without errors.

SLI_gDB_T_DIAG				
	Name	Data type	Start value	Monitor value
1	Static			
2	connectionID	CONN_OUC	16#0100	16#0100
3	T_DIAG	Struct		
4	start	Bool	false	TRUE
5	busy	Bool	false	FALSE
6	done	Bool	false	TRUE
7	error	Bool	false	FALSE
8	status	Word	16#0	16#7000
9	memErrStatus	Word	16#0	16#0000

The diagnostics data are recorded at the RESULT parameter ("TDiag_StatusExt").

10	connectionDiag	TDiag_StatusExt		
11	InterfaceId	HW_ANY	16#0	16#0040
12	ID	CONN_OUC	16#0	16#0100
13	ConnectionType	Byte	0	16#13
14	ActiveEstablished	Bool	false	FALSE
15	State	Byte	0	16#04
16	Kind	Byte	0	16#01
17	SentBytes	UDInt	0	0
18	ReceivedBytes	UDInt	0	0
19	ConnTrials	UDInt	0	0
20	ConnTrialsSuccess	UDInt	0	2
21	LastConnErrReason	UDInt	0	0
22	LastConnErrTimeSt...	LDT	LDT#1970-01-01-(	LDT#1970-01-01-0...
23	LastDisconnReason	UDInt	0	20226
24	LastDisconnTimeS...	LDT	LDT#1970-01-01-(	LDT#2012-01-21-0...

Network 2:

When T_DIAG signals an error ("T_DIAG.error" is "TRUE"), the reported status ("T_DIAG.status") is stored permanently ("T_DIAG.memErrStatus").

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

T_DIAG: Checking the connection (Page 3603)

T_CONFIG: Configure interface

Description T_CONFIG (S7-1200, S7-1500)

Description

The "T_CONFIG" instruction is used for the program-controlled configuration of the integrated PROFINET interfaces of the CPU or the interface of a CP/CM.

Configuration of the integrated PROFINET interfaces of the CPU

With the "T_CONFIG" instruction, you can change the Ethernet address, the PROFINET device name or the IP addresses of the NTP servers for time-of-day synchronization from within the user program. The existing configuration data is overwritten.

You can make the following changes:

- Settings for the IP protocol
  - IP address
  - Subnet mask
  - Router address
- Settings for PROFINET
  - Assignment of the PROFINET device name

  Note

  Dependency between device names and IP protocol

  The IP protocol must also be set when a new device name is assigned.

- Settings for time-of-day synchronization
  - Assignment of the IP addresses of the NTP servers for time-of-day synchronization (only with S7-1500 and there only for the PROFINET interface [X1} of a modular CPU)

These settings correspond to the configuration options under "IP protocol" and "PROFINET" in the "Ethernet addresses " dialog or the "NTP mode" in the "Time synchronization" dialog.

This is displayed in the "Devices & Networks" view under the properties of the PROFINET interface.

WARNING

Restart of CPU after execution of the "T_CONFIG" instruction (only S7-1200 CPUs with firmware versions V2.0 to V4.1.2)

The CPU is restarted after you have executed the instruction to change an IP parameter. The CPU goes to STOP mode, a warm restart is carried out and the CPU starts up again (RUN mode).

Make sure that the control process is in a secure operating mode after the CPU has been restarted following execution of the instruction. Uncontrolled operation can result in serious material damage or personal injury due to malfunctions or programming errors, for example. Non-retentive data could be lost.

Configuration of the PROFINET interface of a CP 1543-1

With the "T_CONFIG" instruction you can change the IPv6 address, the MAC address or the IP addresses of up to two DNS servers of the PROFINET interface of a CP 1543-1 from within the user program. In addition, the NTP servers can be configured for the CP. The existing configuration data is overwritten.

Requirement

- To use the instruction, you must set specific properties of the PROFINET interface. For this purpose open the properties of the PROFINET interface in the device view. In the "Ethernet addresses" or "Time synchronization" dialog, enable the following options:
  - In order to change the IP address parameters with "T_CONFIG": Select the setting "IP address is set directly at the device" under "IP protocol".
  - In order to change the PROFINET device name with "T_CONFIG":
    Select the setting "PROFINET device name is set directly at the device" under "PROFINET".
    Also select the setting "IP address is set directly at the device" under "IP protocol".
  - To change the IP addresses of the NTP servers with "T_CONFIG".
    Select the setting "IP address is set directly at the device" under "IP protocol".
    Select "Enabled time synchronization via NTP server" and specify the IP address of at least one NTP server.
- The configuration data must be stored in the following system data types and assigned at the parameter CONF_DATA (Page 3616):
  - Store the IP address, subnet mask, and the router address in the system data type IF_CONF_V4.
  - Store the device name in the system type IF_CONF_NOS. Observe the restrictions that exist for assigning the name of the device (see parameter CONF_DATA (Page 3616)).
  - Use store the IP addresses for the NTP time synchronization in the system data type IF_CONF_NTP.
  - Store the IPv6 address of the PROFINET interface of a CP 1543-1 in the system data type IF_CONF_IPV6.
  - Store the MAC address of the PROFINET interface of a CP 1543-1 in the system data type IF_CONF_MAC.
  - Store the IP addresses of up to two DNS servers for the PROFINET interface of a CP 1543-1 in the system data type IF_CONF_DNS.

Functional description

The "T_CONFIG" instruction works asynchronously, which means its execution extends over multiple calls. The configuration process is started by calling "T_CONFIG" with REQ = 1. Only one job can be active at any time.

The block is edge triggered which means that after BUSY= FALSE the block must be re-called with REQ=FALSE to enable the instance.

Parameter

The following table shows the parameters of the instruction "T_CONFIG":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Calling the instruction with REQ = 1 starts processing of the instruction.
INTERFACE	Input	HW_INTERFACE	I, Q, M, D, L or constant	Hardware identifier of the interface The hardware ID is displayed in the properties of the interface in the device view, and in the system constants of the PLC tags.
CONF_DATA (Page 3616)	Input	VARIANT	D, L	Pointer to the higher-level structure that contains the system data types IF_CONF_HEADER, IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC and IF_CONF_DNS (see description of the parameters CONF_DATA).
DONE (Page 3621)	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Processing not yet complete. • 1: Processing of instruction finished successfully.
BUSY (Page 3621)	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Processing of the instruction has not started, completed or canceled yet. • 1: Processing of instruction is in progress
ERROR (Page 3621)	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error • 1: Error
STATUS (Page 3621)	Output	DWORD	I, Q, M, D, L	Detailed status information: Detailed error and status information in the form of an error code are output at the parameter STATUS.
ERR_LOC (Page 3621)	Output	DWORD	I, Q, M, D, L	Error location: • 0: Error during execution of the instruction or when assigning parameters. • > 0: Errors in the structure or content of the configuration data at the parameter CONF_DATA.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

You will find the example here: Program example for T_CONFIG (Page 3623).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Evaluating errors with output parameter RET_VAL (Page 162)

Difference between synchronous and asynchronous instructions (Page 596)

Parameter CONF_DATA (S7-1200, S7-1500)

Structure of the configuration data

The configuration data at the parameter CONF_DATA can be stored in a global data block or in the section "Static" of the block interface.

The configuration data must be stored according to the following structure:

Name		Data type	Description
ConfData		Struct	Parent structure that is assigned at the parameter CONF_DATA.
	Header	IF_CONF_HEADER	The header is used to define the number of these system data types. The system data type IF_CONF_HEADER must always be included.
	IPData	IF_CONF_V4	The IP address, subnet mask and router address are stored in this system data type.
	NoS	IF_CONF_NOS	The PROFINET device names are stored in this system data type. Only create IF_CONF_NOS if you also want to change the device names with "T_CONFIG".
	NTP	IF_CONF_NTP	In this system data type, use store the IP addresses of the NTP servers for time synchronization.
	IP_V6	IF_CONF_V6	Store the IPv6 address of a CP 1543-1 in this system data type.
	MAC	IF_CONF_MAC	Store the MAC address of a CP 1543-1 in this system data type.
	DNS	IF_CONF_DNS	Store the IP addresses of up to two DNS servers for a CP 1543-1 in this system data type .

Create the system data types IF_CONF_HEADER, IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC and IF_CONF_DNS be entering the name of the system data type in the "Data type " column of the data block or the block interface. The name for the system data types can be freely assigned. The system data types can be combined in any order.

System data type IF_CONF_Header

Use the system data type IF_CONF_Header to specify the number of system data types IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC and IF_CONF_DNS that are used when executing "T_CONFIG".

Byte	Parameter	Data type	Start value	Description
0 ... 1	FieldType	UINT	0	Field type: Must always have the value "0".
2 ... 3	FieldId	UINT	0	Error ID: Must always have the value "0".
4 ... 5	SubfieldCount	UINT	0	Number of system data types IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC and IF_CONF_DNS used

System data type IF_CONF_V4

The IP address, subnet mask and router address are defined with the system data type IF_CONF_V4.

Byte	Parameter	Data type	Start value	Description
0 ... 1	Id	UINT	30	System data type ID The start value of this parameter must not be changed.
2 ... 3	Length	UINT	18	Length of the system data type IF_CONF_V4 The start value must be used for the length specification since the parameters of IF_CONF_V4 have a fixed length and structure.
4 ... 5	Mode	UINT	0	Validity of addressing: • 1: Permanent validity of the configuration data • 2: Temporary validity of the configuration data including the deletion of existing permanent configuration data
6 ... 9	InterfaceAddress	IP_V4 *	0.0.0.0	IP address
10 - 12	SubnetMask	IP_V4 *	0.0.0.0	Subnet mask
14 - 16	DefaultRouter	IP_V4 *	0.0.0.0	Router address
* The data type IP_V4 is a structure of 4 BYTE, which includes the respective address of the respective parameter (e.g. at parameter SubnetMask the four-digit address of the subnet mask of the IP protocol).				

System data type IF_CONF_NOS

With the system data type IF_CONF_NOS you specify the station name to be assigned when the "T_CONFIG" instruction executes.

Byte	Parameter	Data type	Start value	Description
0 ... 1	Id	UINT	40	System data type ID The start value of this parameter must not be changed.
2 ... 3	Length	UINT	246	Length of the system data type IF_CONF_NOS in bytes. • For an absolute length, the value for the Lenght parameter consists of: – 6 bytes for the parameter Id, Length and Mode. – up to 240 bytes for the device name (parameter NOS). Example: An overall length of 10 is the result for the device name "plc1" with a length of 4 characters (=4 bytes). • Use the default start value 246 at the Lenght parameter for a dynamic length. Make sure that you enter the value "0" after the name (see description for NOS parameter).

Byte	Parameter	Data type	Start value	Description
4 ... 5	Mode	UINT	0	Validity of the device name change: • 1: Permanent validity of the device name. • 2: Temporary validity of the device name.
6 ... 244	NOS	ARRAY [1...240] of Byte	0	Device name (Name of Station) • You must occupy the ARRAY from the first byte. The station name is deleted if "0" is assigned as the first byte. • The minimum length for the name is 1 byte. The maximum length for a name is 240 bytes. • If the device name is shorter than specified at the parameter Length then a zero byte (16#0 hex) must be entered after the actual station name (according to IEC 61185-6-10). Otherwise NOS will be rejected and the instruction "T_CONFIG" outputs the error codeDW#16#C0809400 at the parameter STATUS. • If the device name is longer than specified at the parameter Length then the device name will only be written up to to specified length. The following restrictions apply for the device name: • The name must be specified in ASCII code. • Only lowercase letters, numbers, hyphen or dots may be used for the name. – The name may not begin or end with a hyphen. – The name may not have the form n.n.n.n (n = 0, ... 999). – The name may not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9). • A name component between two points may be up to 63 characters long. • No special characters such as umlauts, brackets, underscore, slash, blank etc. The error code C080_9400 is output at the parameter STATUS if an invalid character is used.

System data type IF_CONF_NTP

Using the system data type IF_CONF_NTP you specify the IP addresses of the NTP servers for time synchronization.

Byte	Parameter	Data type	Start value	Description
0 ... 1	Id	UINT	17	System data type ID The start value of this parameter must not be changed.
2 ... 3	Length	UINT	22	Length of the system data type IF_CONF_NTP in bytes The start value must be used for the length specification since the parameters of IF_CONF_NTP have a fixed length and structure.
4 ... 5	Mode	UINT	2	Validity: • 1: Permanent validity (not permitted) • 2: Temporary validity
6 ... 9	NTP_IP[1]	ARRAY [1...4] of IP_V4		IP address of NTP server 1
...	...			IP address of NTP server 2 ... 3
18 ... 21	NTP_IP[4]	ARRAY [1...4] of IP_V4		IP address of NTP server 4

System data type IF_CONF_V6 (only when using a CP 1543-1)

With the system data type IF_CONF_V6 you define an IPv6 address for the interface selected with INTERFACE.

Byte	Parameter	Data type	Start value	Description
0 ... 1	Id	UINT	22	System data type ID The start value of this parameter must not be changed.
2 ... 3	Length	UINT	22	Length of the system data type IF_CONF_V6 in bytes The start value must be used for the length specification since the parameters of IF_CONF_V6 have a fixed length and structure.
4 ... 5	Mode	UINT	0	Validity of the IPv6 address: • 1: Permanent validity • 2: Temporary validity
6 ... 21	InterfaceAddress	IP_V6 *		IPv6 address
* The data type IP_V6 is a structure of 16 bytes.				

System data type IF_CONF_MAC (only when using a CP 1543-1)

With the system data type IF_CONF_MAC you set a MAC address for the interface selected with INTERFACE.

Byte	Parameter	Data type	Start value	Description
0 ... 1	Id	UINT	3	System data type ID The start value of this parameter must not be changed.
2 ... 3	Length	UINT	12	Length of the system data type IF_CONF_MAC in bytes The start value must be used for the length specification since the parameters of IF_CONF_MAC have a fixed length and structure.
4 ... 5	Mode	UINT	0	Validity of MAC address: • 1: Permanent validity • 2: Temporary validity
6 ... 11	Mac	ARRAY [1...6] of Byte		MAC address Note that by changing the MAC address, connections can abort.

System data type IF_CONF_DNS (only when using a CP 1543-1)

With the system data type IF_CONF_DNS you set IP addresses of up to two DNS servers for the interface selected with INTERFACE.

Byte	Parameter	Data type	Start value	Description
0 ... 1	Id	UINT	16	System data type ID The start value of this parameter must not be changed.
2 ... 3	Length	UINT	14	Length of the system data type IF_CONF_DNS in bytes The start value must be used for the length specification since the parameters of IF_CONF_DNS have a fixed length and structure.
4 ... 5	Mode	UINT	0	Validity: • 1: Permanent validity • 2: Temporary validity
6 ... 9	DNS_IP1	IP_V4		IP address of DNS server 1
10 ... 13	DNS_IP2	IP_V4		IP address of DNS server 2 If only one address is to be set, the IP address of DNS server 2 must be set to 0.0.0.0.

DONE, BUSY and ERROR parameters

Description

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current status of the instruction and when the transfer of configuration data is concluded.

BUSY	DONE	ERROR	Description
TRUE	FALSE	FALSE	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the parameter STATUS (Page 3621).
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Parameter STATUS and ERR_LOC

Description

The status and error messages of the instruction "T_CONFIG" are output at the parameters STATUS and ERR_LOC.

- The cause of the error is output at the parameter STATUS.
- The location of the error that occurred is output at the parameter ERR_LOC. The following options are available here:
  - 16#0000_0000: Error when calling the instruction (e.g. errors when assigning parameters to the instruction or in communication with the PROFINET interface).
  - 16#0001_0000: Error with the configuration data in the parameters of the system data type IF_CONF_HEADER.
  - 16#0001_000x: Error in the configuration data in the parameters of system data type IF_CONF_V4 or IF_CONF_NOS or IF_CONF_NTP or IF_CONF_V6 or IF_CONF_MAC or IF_CONF_DNS (x specifies the position of the bad sub block in the T_CONFIG structure. If the T_CONFIG structure contains, for example a sub block for the IP address and a sub block for the station name and the error is located in the sub block for the station name, ERR_LOC has the value 0001_0002.)

The following table shows the possible values for the parameters STATUS and ERR_LOC:

STATUS*	ERR_LOC*	Explanation
0000_0000	0000_0000	Order processing completed without errors.
0070_0000	0000_0000	No job processing active.
0070_0100	0000_0000	Start of the order processing.
0070_0200	0000_0000	Intermediate call (REQ irrelevant).
C08x_yy00	0000_0000	General error information. See also: GET_ERR_ID: Get error ID locally (Page 883)
C080_8000	0000_0000	Error at call of the instruction: The hardware ID at the parameter Interface is invalid.

STATUS*	ERR_LOC*	Explanation
C080_8100	0000_0000	Error at call of the instruction: The hardware ID at the parameter Interface does not address a PROFINET interface.
C080_8700	0000_0000	Error at call of the instruction: Incorrect length of the data block at the parameter CONF_DATA.
C080_8800	0001_0000	Error in the system data type IF_CONF_HEADER: The parameter FieldType has an invalid value. Use the value "0" for FieldType.
C080_8900	0001_0000	Error in the system data type IF_CONF_HEADER: The parameter FieldId has an invalid value. Use the value "0" for FieldId.
C080_8A00	0001_0000	Error in the system data type IF_CONF_HEADER: Incorrect number at the parameter SubfieldCount. Enter the correct number of system data types IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC or IF_CONF_DNS being used.
C080_8B00	0001_000x	Error in the system data type IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC or IF_CONF_DNS: The parameter Id has an invalid value. For IF_CONF_V4 use "30", for IF_CONF_NOS "40", for IF_CONF_NTP "17", for IF_CONF_V6 "22", for IF_CONF_MAC "3", for IF_CONF_DNS "16". Note: IF_CONF_NTP is only permitted if T_CONFIG is used on the first interface [X1] of an S7-1500 modular CPU.
C080_8C00	0001_000x	Error in the system data type IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC or IF_CONF_DNS: Incorrect data type system used, wrong order or multiple use of a system data type.
C080_8D00	0001_000x	Error in the system data type IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC or IF_CONF_DNS: The parameter Length has an incorrect or invalid value.
C080_8E00	0001_000x	Error in the system data type IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC or IF_CONF_DNS: The parameter Mode has an incorrect or invalid value. • With IF_CONF_V4 and IF_CONF_NOS only the values "1" (permanent) or "2" (temporary) are permitted. • With IF_CONF_NTP only the value "2" (temporary) is permitted.
C080_9000	0001_000x	Error in the system data type IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC or IF_CONF_DNS: Configuration data cannot be applied. Possible cause: • With IF_CONF_V4: In the hardware configuration, the setting "Set IP address on the device" was not selected.. • With IF_CONF_NOS: In the hardware configuration, the setting "Set PROFINET device name on the device" was not selected.. • With IF_CONF_NTP: In the hardware configuration, the setting "Enable time synchronization via NTP server " was not selected and no IP address was set for NTP servers..
C080_9400	0001_000x	Error in the system data type IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC or IF_CONF_DNS: A parameter value is undefined or invalid.
C080_9500	0001_000x	Error in the system data type IF_CONF_V4, IF_CONF_NOS, IF_CONF_NTP, IF_CONF_V6, IF_CONF_MAC or IF_CONF_DNS: The values of two parameters are inconsistent.

STATUS*	ERR_LOC*	Explanation
C080_C200	0000_0000	Error at call of the instruction: The configuration data can not be transferred. Possible cause: The PROFINET interface is not accessible.
C080_C300	0000_0000	Error at call of the instruction: Insufficient resources (e.g., multiple calling of "T_CONFIG" with different parameters).
C080_C400	0000_0000	Error at call of the instruction: Temporary communication error. Time indication for change to daylight saving time.
C080_D200	0000_0000	Error at call of the instruction: Call not possible. Instruction is not supported by the selected PROFINET interface.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 7698)

Program example for T_CONFIG

Introduction

In the following example you use a configured connection (by UDP for example) between two CPUs. Configure the IP address and the PROFINET device name of CPU 1 using the instruction "T_CONFIG".

Requirements

Set connection:

- Two CPUs (for example, S7-1513-1 PN) are available and connected to each other via PROFINET.
- A configured connection is not required.

Setting up PROFINET of CPU 1:

1. Open the device view of the CPU 1 "> Properties > PROFINET interface > Ethernet addresses > PROFINET".
2. To set the device names, select the following options:
   - "IP address is set directly at the device".
   - "PROFINET device name is set directly at the device".

Adjust the value of "hwid":

- Change the value of "hwid" according to the hardware identifier of the PROFINET interface of your local device (CPU 1).

Note

Open "PLC tags > Show all tags > System constants". Find the entry "<Local~PROFINET_interface_1>", with the data type "Hw_Interface". The "value" cell contains the hardware identifier.

Create tags and interconnect parameters (program of CPU 1)

To store the data, create a global data block ("SLI_gDB_T_CONFIG") with the following structures and tags.

SLI_gDB_T_CONFIG			
	Name	Data type	Start value
1	Static		
2	T_CONFIG	Struct	
3	start	Bool	false
4	hwid	HW_INTERFACE	64
5	busy	Bool	false
6	done	Bool	false
7	error	Bool	false
8	errorLocation	DWord	16#0
9	status	DWord	16#0
10	memErrStat	DWord	16#0
11	configData	Struct	
12	header	IF_CONF_Header	
13	deviceIP	IF_CONF_v4	
14	deviceName	IF_CONF_NOS	

In the structure "configData": Assign parameters for the system data type "IF_CONF_Header" for specifying the size of the PROFINET data, as follows:

12	header	IF_CONF_Header	
13	FieldType	UInt	0
14	FieldId	UInt	0
15	SubfieldCount	UInt	2

In the structure "configData": Assign parameters for the system data type "IF_CONF_v4" for defining the IP address, as follows:

16	deviceIP	IF_CONF_v4	
17	Id	UInt	30
18	Length	UInt	18
19	Mode	UInt	1
20	InterfaceAddress	IP_V4	
21	ADDR	Array[1..4] of Byte	
22	ADDR[1]	Byte	192
23	ADDR[2]	Byte	168
24	ADDR[3]	Byte	0
25	ADDR[4]	Byte	3
26	SubnetMask	IP_V4	
27	ADDR	Array[1..4] of Byte	
28	ADDR[1]	Byte	255
29	ADDR[2]	Byte	255
30	ADDR[3]	Byte	255
31	ADDR[4]	Byte	0
32	DefaultRouter	IP_V4	
33	ADDR	Array[1..4] of Byte	
34	ADDR[1]	Byte	16#0
35	ADDR[2]	Byte	16#0
36	ADDR[3]	Byte	16#0
37	ADDR[4]	Byte	16#0

In the structure "configData": Assign parameters for the system data type "IF_CONF_NOS" for defining the PROFINET device name, as follows:

38	deviceName	IF_CONF_NOS	
39	Id	UInt	40
40	Length	UInt	11
41	Mode	UInt	1
42	NOS	Array[1..240] of Byte	
43	NOS[1]	Byte	'm'
44	NOS[2]	Byte	'y'
45	NOS[3]	Byte	'p'
46	NOS[4]	Byte	'l'
47	NOS[5]	Byte	'c'

Note

Structure of "configData"

The information of the structure "configData" can be interpreted as follows:

- In the Header (IF_CONF_Header).
  SubfieldCount = 2: The value means: Two additional structures ("deviceIP", "deviceName") (*) are used below.
  *Make sure to maintain the order of both structures.
- For In the structure "deviceName" (system data type IF_CONF_NOS)
  - Lenght = 11 (*). The value corresponds to the overall length of the structure NOS (5 bytes for the device name"myplc" + 6 bytes for the parameters Id, Length and Mode)
    *Instead of an absolute length, you can use the default start value (Lenght = 0) for a dynamic length.
  - Mode = 1. The value causes a permanent change of the device name in "myplc".
  - NOS[1] ... NOS[5]. The NOS array contains the new device name (1 character / byte).

Create the FB "SLI_FB_T_CONFIG" Create the following local tags in it.

7	Static			
8	T_CONFIG_Instance	T_CONFIG		
9	done	Bool	false	Non-retain
10	errorLocation	DWord	16#0	Non-retain

Network 1: Interconnect the parameter of the instruction "T_CONFIG" as follows:

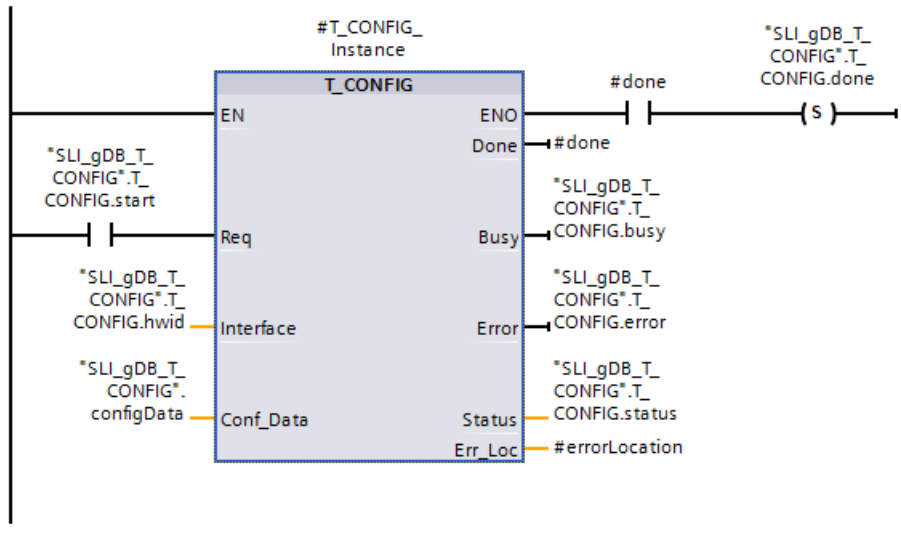

Network 2: In case of error of T_CONFIG, save the status as follows.

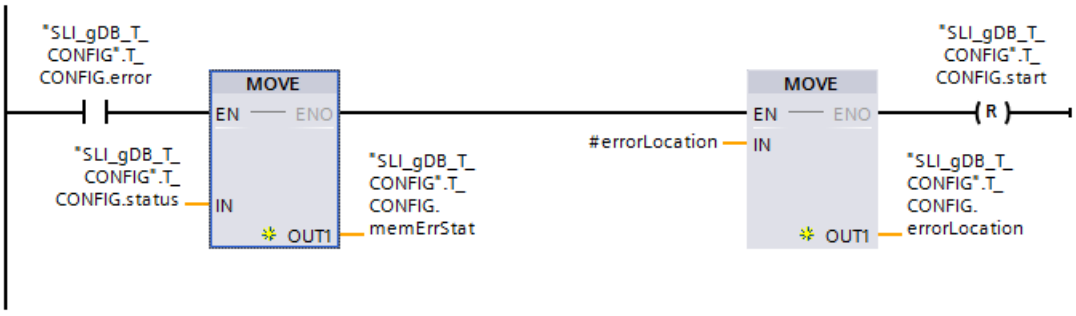

Network 3: To exit T_CONFIG, create the following interconnections.

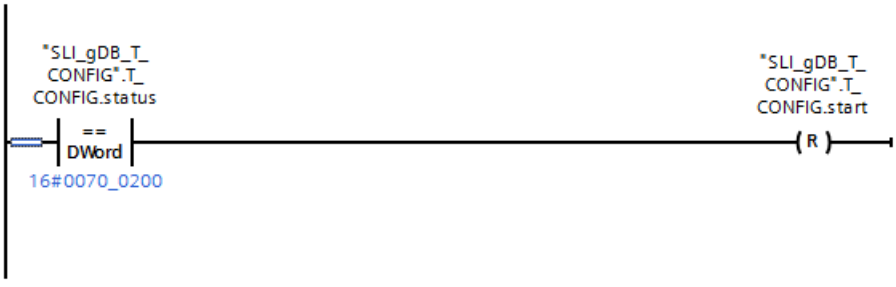

Result

Network 1:

When the input parameter REQ ("T_CONFIG.start") supplies the signal state "TRUE", the "T_CONFIG" instruction is started. The instruction "T_CONFIG" configures the integrated PROFINET interface of the CPU for multiple calls. Based on the input parameter INTERFACE ("T_CONFIG.hwid"), the "T_CONFIG" instruction knows the interface to be used.

Successful transfer of the data record () is indicated by the output parameter DONE ("#done") with the signal state "TRUE". At the same time the value "0000_0000" is displayed at the output parameter STATUS ("T_CONFIG.status"). Because the values of the output parameters are only displayed for the moment of their validity, save the success message in the tag "T_CONFIG.done".

The output parameter ERROR ("T_CONFIG.error") or the tag "T_CONFIG.memErrStat" indicates that processing in the example is taking place without errors.

SLI_gDB_T_CONFIG				
	Name	Data type	Start value	Monitor value
1	Static			
2	T_CONFIG	Struct		
3	start	Bool	false	FALSE
4	hwid	HW_INTERFACE	64	16#0040
5	busy	Bool	false	FALSE
6	done	Bool	false	TRUE
7	error	Bool	false	FALSE
8	errorLocation	DWord	16#0	16#0000_0000
9	status	DWord	16#0	16#0070_0000
10	memErrStat	DWord	16#0	16#0000_0000
11	configData	Struct		

Network 2:

If T_CONFIG signals an error ("T_CONFIG.error", "TRUE"), save the alarm as follows:

- Save the status ("T_CONFIG.status") in the "T_CONFIG.memErrStat" tag.
- Save the error location, output at the output parameter ERR_LOC ("#errorLocation") from T_CONFIG, in the tag "T_CONFIG.errorLocation".

Online & diagnostics

To check whether the PROFINET data were changed, open the path "Online & Diagnostics > Functions > Assign name" in the project tree.

Accessible devices in the network:			
IP address	MAC address	Device	PROFINET device name
192.168.0.2	28-63-36-8D-BB-49	S7-1500	simaticplc
192.168.0.3	00-1B-1B-22-63-D0	S7-1500	myplc

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

See also

Description T_CONFIG (Page 3612)

Changes to the communications instructions

Differences between the versions <= V3.x and >= V4.1 of the OUC library

Introduction

CPU S7-1200 as of firmware version 4.1 supports new instruction versions for Open User Communication (OUC). The new instructions are included in the library with version 4.1.

If you want to use this library, changes must be made to the user program, since some of the new instructions for Open User Communication behave differently.

This section describes the differences in detail, especially in the call behavior of the instructions.

Note

Description only relevant when upgrading from the CPU S7-1200 ≤ V4.0 to S7-1200 ≥ V4.1

If you are using an S7-1500 CPU, changes between the library versions are not relevant. The same applies if you are using an S7-1200 ≥ Version 4.1 and do not convert the library for the Open User Communication to versvon <4.1.

Differences between the versions <= V3.x and >= V4.1 of the Open User Communication library

The following table shows the instruction that have differences between the versions <= V3.x and >= V4.1 of the Open User Communication library Click on the name of the instruction for detailed information.

Instruction	Version in library <= V3.x (CPU FW ≤ V4.0)	Version in library >= V4.1 (CPU FW ≥ V4.1)
TSEND_C (Page 3630)	V2.1	V3.0
TRCV_C (Page 3631)	V2.1	V3.0
TMAIL_C *	V2.1	V3.0
TCON (Page 3633)	V3.0	V4.0
TDISCON	V2.1	V2.1 (identical to library <= V3.x)
TSEND (Page 3635)	V3.0	V4.0
TRCV (Page 3637)	V3.0	V4.0
TUSEND (Page 3635)	V3.0	V4.0
TURCV (Page 3637)	V3.0	V4.0
T_RESET *	V1.1	V1.2
T_DIAG *	V1.1	V1.2
T_CONFIG	V1.0	V1.0 (identical to library <= V3.x)
* There are no differences in the versions that affect the user program.		

Note

Changing the Open User Communication library from a version <= V3.x to a version >= V4.1

When replacing the Open User Communication library from a version <= V3.x to a version >= V4.1 you also need to replace the Modbus TCP library. Afterwards check all the instructions relevant for your program.

See also

Differences between the versions <= V3.x and >= V4.0 of the Modbus TCP library (Page 4139)

Changes with the TSEND_C instruction

Call behavior of TSEND_C (V<3.0)

Up to version 2.1 of the TSEND_C instruction, the DONE output parameter is set twice: Once when a connection is established by the internally used TCON instruction and then after a send operation by the internally used TSEND instruction.

The following diagram shows the connection establishment and the sending of data with TSEND_C V2.1:

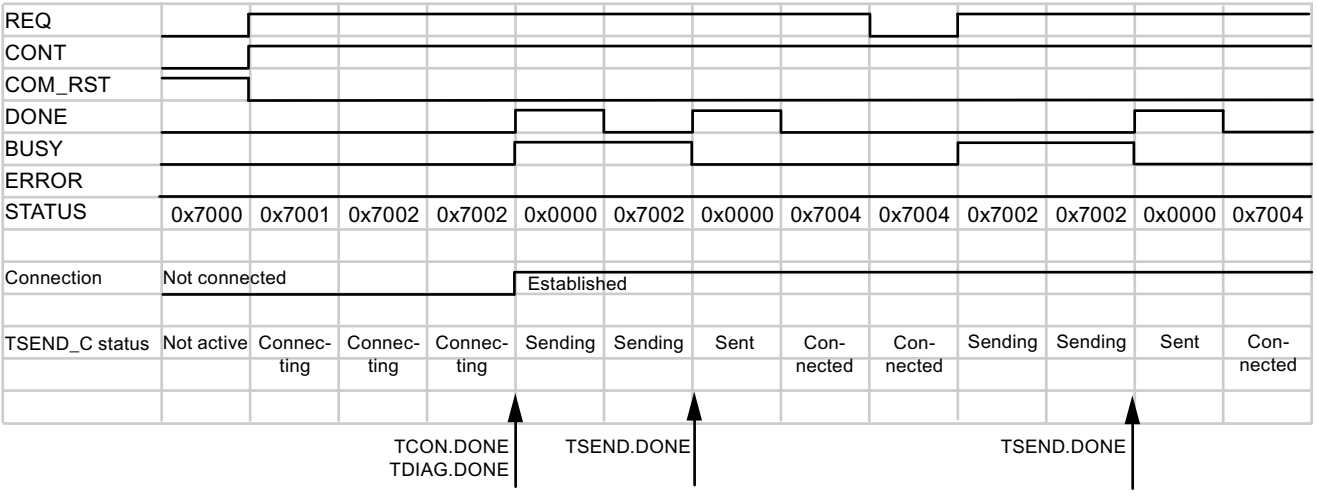

When setting the COM_RST parameter to "1", the current connection establishment or a current data transmission can be reset at any time. This terminates the existing communication connection and a new connection is established.

Call behavior of TSEND_C (V≥3.0)

As of the TSEND_C version, the DONE parameter is only set when data transfer has been completed by the internally used TSEND instruction (STATUS = 0000).

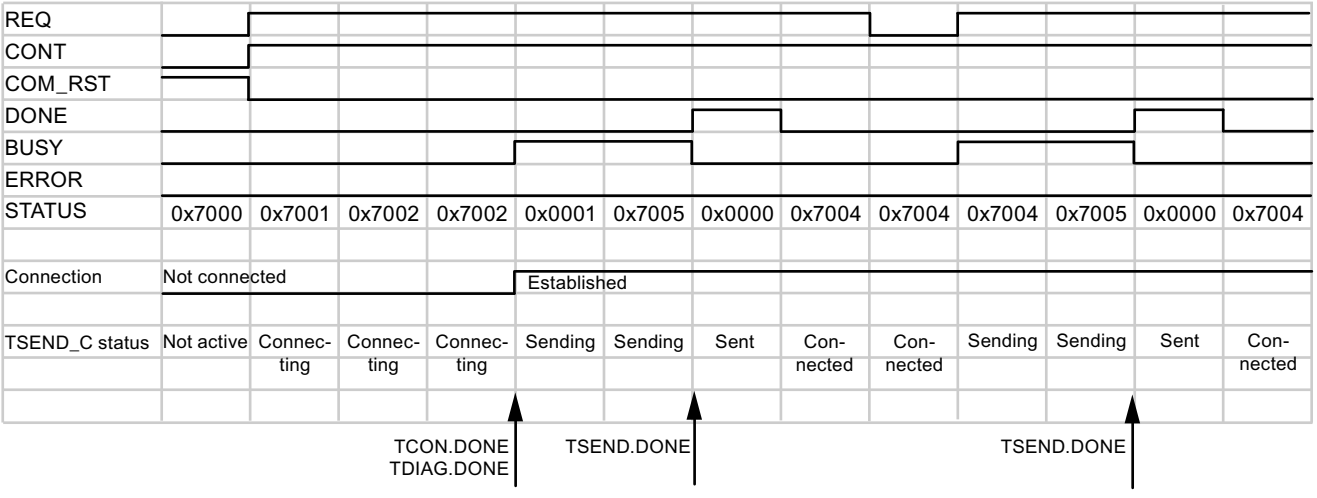

The existing connection is temporarily interrupted and reset when the COM_RST parameter is set to "1". However, unlike TSEND_C V2.1, the connection endpoint is retained.

Note

Additional protocols with TSEND_C as of version 3.0

Version 3.0 of the TSEND_C instruction also supports UDP and UDP Broadcast via the interface of the CPU and via CM/CP.

See also

TSEND_C: Send data via Ethernet (Page 3478)

TSEND_C: Establishing a connection and sending data (Page 3482)

Changes with the TRCV_C instruction

Call behavior of TRCV_C (V<3.0)

Up to version 2.1 of the TRCV_C instruction, the DONE output parameter is set after the connection is established. The STATUS output parameter does not distinguish whether the connection or the data transfer has been completed.

The following diagram shows the connection establishment and the sending of data with TRCV_C V2.1:

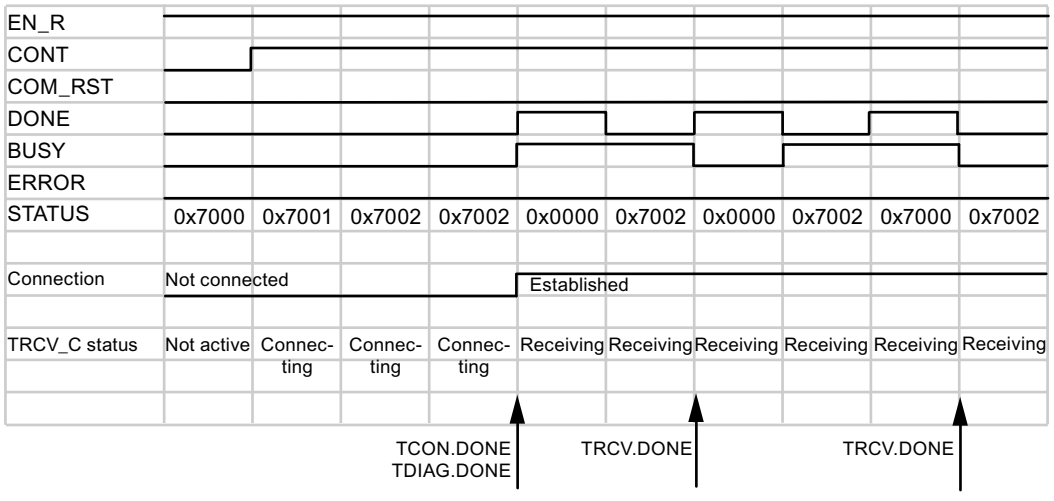

The current connection establishment or a current data transmission can be reset at any time by setting the COM_RST parameter to "1". This terminates the existing communication connection and a new connection is established.

Call behavior of TRCV_C (V≥3.0)

As of the TRCV_C version, the DONE parameter is only set when data transfer has been completed by the internally used TRCV instruction (STATUS = 0000). The completion of the connection establishment by the internally used TCON instruction is displayed by the output value 0x0001 at the STATUS parameter.

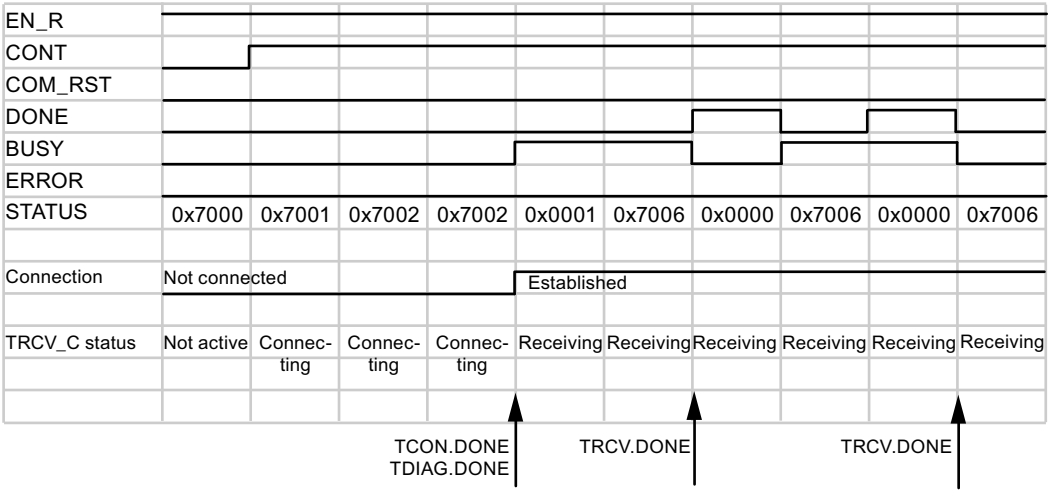

The existing connection is temporarily interrupted and reset when the COM_RST parameter is set to "1". However, unlike TSEND_C V2.1, the connection endpoint is retained.

Note

Additional protocols with TRCV_C as of version 3.0

Version 3.0 of the TRCV_C instruction also supports UDP and UDP Broadcast via the interface of the CPU and via CM/CP.

Note

ADHOC mode for the TCP protocol variant

In accordance with the TRCV instruction, with TRCV_C up to version 2.1, ADHOC mode is activated by assigning the value "0" to the LEN parameter. As of version 3.0 of the instruction, you use the ADHOC parameter for this. You can find detailed information in the descriptions of the instructions.

See also

Changes with the TRCV/TURCV instructions (Page 3637)

TRCV_C: Receive data via Ethernet (Page 3491)

TRCV_C: Establishing a connection and receiving data (Page 3497)

Changes with the TCON instruction

Changed call behavior with connection errors

Previous call behavior of TCON (V<4.0)

- The previous TCON instruction (V<4.0) is activated with a rising edge at the REQ input parameter.
- If the remote communication partner cannot be reached, the instruction sets the BUSY output parameter.
- An error message is not output.

The following diagram shows the behavior of TCON on the part of the active connection partner. If no connection is made, the instruction is not called again.

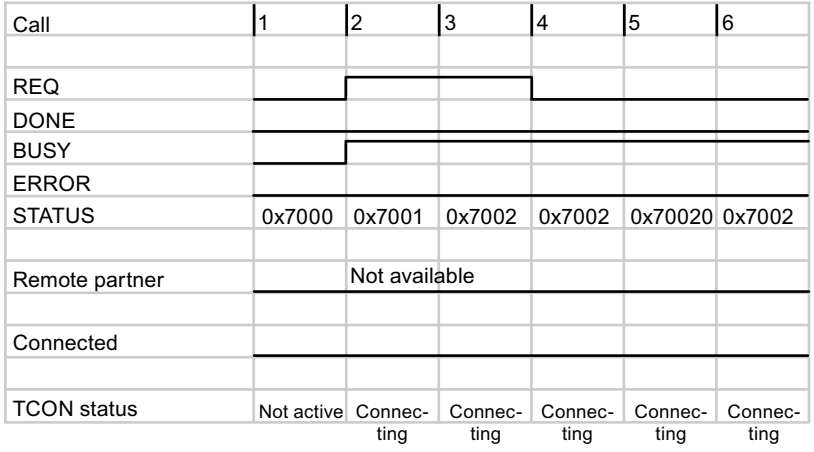

New call behavior of TCON (V≥4.0)

- The new TCON instruction is also activated with a rising edge at the REQ input parameter.
- Unlike before, an error message is output when the remote communication partner cannot be reached. The error can be queried and the call started again by a new edge at the REQ parameter.
- The following changes result for the user program:
  - Additional error messages of TCON that can be evaluated (see description of TCON (Page 3541)).
  - The connection establishment can be re-initialized via a new rising edge. To do this, query the DONE and ERROR parameters to ensure that an error is present.

The following diagram shows the behavior of TCON on the part of the active connection partner. The instruction is called again after a network error (error code 80C6).

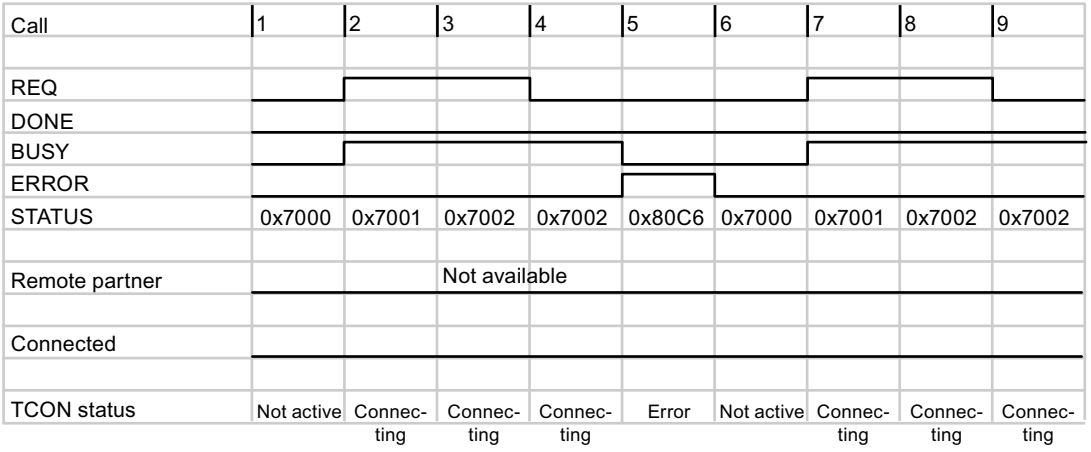

Call behavior without communication errors

If there is no connection error, the TCON versions have the same behavior.

The following diagram shows the behavior of TCON on the part of the active connection partner.

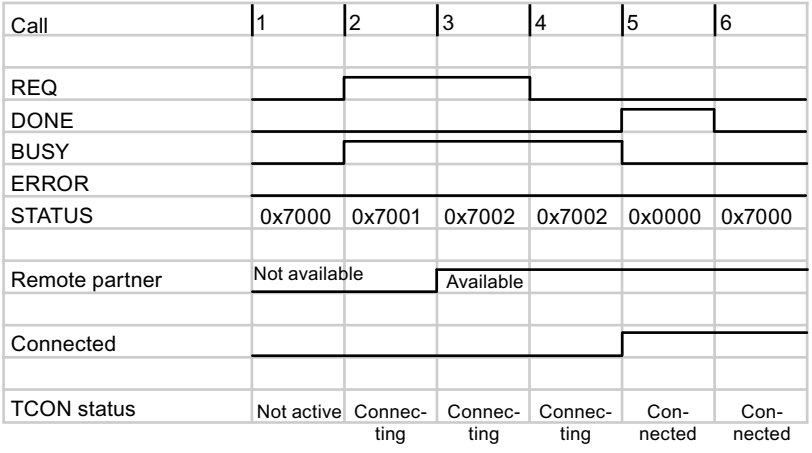

See also

TCON: Establish communication connection (Page 3538)

Changes with the TSEND/TUSEND instructions

Send operation with an established connection

If no delay or interruption occurs during sending, versions 3.0 and 4.0 of the TSEND/TUSEND instructions behave the same way:

- The instruction is started with a positive edge at the "REQ" parameter. The instruction is executed asynchronously, i.e. it must be called until the full execution is indicated by the DONE parameter.
- A maximum of 8 KB of data can be sent with TSEND. A maximum of 1472 bytes with TUSEND (as of firmware version V2.5 of the S7-1500 CPUs for Unicast or Multicast: 2048 bytes). The amounts are based on the full execution of the instruction, i.e. all required calls until the DONE parameter is set.
- Data transmission takes place in three steps:
  - The data is written from the operand area to an internal buffer (indicated by STATUS=7001).
  - The transmission to the remote communication partner follows.
  - If the transmission is successful, DONE is set to "1" and STATUS is reset to "0" (see call 5 in the diagram below).

Call	1	2	3	4	5	6
REQ						
DONE						
BUSY						
ERROR						
STATUS	0x7000	0x7001	0x7002	0x7002	0x0000	0x7000
LEN (max.)	8 KB	8 KB	Internal			
Connected						
TSEND status	Not active	Sending	Sending	Sending	Sent	Not active

Send operation with time-delayed call

If the send operation is not performed via a communication module (CM) or a communication processor (CP), versions 3.0 and 4.0 of the TSEND/TUSEND instructions behave the same way:

- In the following example, TSEND or TUSEND are only called once with a rising edge at the REQ parameter (see call 2 in the diagram below).
- If the time delay is sufficiently long and the data could be transferred, the DONE parameter is set directly to "1" in the next call (call 3 here).

Call	1	2	Time delay → No call		3	4
REQ						
DONE						
BUSY						
ERROR						
STATUS	0x7000	0x7001			0x0000	0x7000
LEN (max.)	8 KB	8 KB	Internal			
Connected						
TSEND status	Not active	Sending	Sending	Sending	Sent	Not active

If the send operation is performed via a CM/CP, version 4.0 of the TSEND/TUSEND instructions behaves differently. In this case, the instruction must be called multiple times until the receipt of data is confirmed by the NDR parameter of the TRCV/TURCV instructions.

Send operation with interrupted connection

If the connection is interrupted during the send operation, the ERROR and STATUS parameters indicate the error and its cause. In the following example, the connection was interrupted during call number 5 and this error is indicated accordingly with the STATUS parameter.

As of version 4.0 of the TSEND/TUSEND instructions, more STATUS alarms are available that can be correspondingly evaluated (see description of TSEND (Page 3553)/TUSEND (Page 3579)) .

Call	1	2	3	4	5	6	7	8	9
REQ									
DONE									
BUSY									
ERROR									
STATUS	0x7000	0x7001	0x7002	0x7002	0x80C4	0x7000	0x7001	0x7002	0x7002
LEN (max.)	8 KB	8 KB	Internal			8 KB	8 KB	Internal	
Connection		Established			Interrupted			Established	
TSEND status	Not active	Sending	Sending	Sending	Error	Not active	Sending	Sending	Sending

See also

TSEND: Send data via communication connection (Page 3550)

Changes with the TRCV/TURCV instructions

Receive operation with an established connection

If no delay or interruption occurs during sending, version 3.0 and 4.0 of the TRCV/TURCV instructions behave the same way:

- A maximum of 8 KB of data can be sent with TRCV. A maximum of 1472 bytes with TURCV (as of firmware version V2.5 of the S7-1500 CPUs for Unicast or Multicast: 2048 bytes). The amounts are based on the full execution of the instruction, i.e. all required calls until the DONE parameter is set.
- The instruction receives data if the EN_R parameter is set to "1".
- The data reception is not complete until the length of data specified at the the LEN parameter has been completely received. The length of the received data is output at the RCVD_LEN output parameter. Only then does the data in the area defined at the DATA parameter become available.

Call	1	2	3	4	5	6
EN_R						
LEN (max.)	8 KB	8 KB	Internal			
NDR						
BUSY						
ERROR						
STATUS	0x7000	0x7001	0x7002	0x7002	0x0000	0x7000
RCVD_LEN	0	0	0	0	max. 8 KB	0
Connection				Established		
TRCV status	Not active	Receiving	Receiving	Receiving	Receiving	Not active

Receive operation with time-delayed call

If the send operation is not performed via a communication module (CM) or a communication processor (CP), versions 3.0 and 4.0 of the TRCV/TURCV instructions behave the same way:

- In the following example, TRCV or TURCV are only called once with a rising edge at the EN_R parameter (see call 2 in the diagram below).
- If the time delay is sufficiently long and the data could be transferred, the NDR parameter is set directly to "1" in the next call (call 3 here).

Call	1	2	Delay No call		3	4
EN_R						
LEN (max.)	8 KB	8 KB	Internal			
NDR						
BUSY						
ERROR						
STATUS	0x7000	0x7001			0x0000	0x7000
RCVD_LEN	0	0	0	0	max. 8 kB	0
Connection				Established		
TRCV status	Not active	Receiving	Receiving	Receiving	Receiving	Not active

If the send operation is performed via a CM/CP, version 4.0 of the TSEND/TUSEND instructions behaves differently. In this case, the instruction must be called multiple times until the receipt of data is confirmed by the NDR parameter.

Receive operation with interrupted connection

If the connection is interrupted during receiving, the ERROR and STATUS parameters indicate the error and its cause. In the following example, call number 5 resulted in a temporary communication error, which is accordingly indicated by the STATUS parameter.

Call	1	2	3	4	5	6	7	8	9
EN_R									
LEN									
NDR									
BUSY									
ERROR	0x7000	0x7001	0x7002	0x7002	0x80C4	0x7000	0x7001	0x7002	0x7002
STATUS	8 KB	8 KB	Internal			8 KB	8 KB	Internal	
RCVD_LEN								Established	
Connection				Established					
TRCV state	Not active	Sending	Sending	Sending	Error	Not active	Sending	Sending	Sending

As of version 4.0 of the TRCV/TURCV instructions, more STATUS alarms are available that can be correspondingly evaluated (see description of TRCV (Page 3561)/TURCV (Page 3583)).

Receive operation using ADHOC mode

ADHOC mode is only available with the TCP protocol variant. You can use ADHOC mode to receive data of variable length with the TRCV/TURCV instruction. If ADHOC mode is active, the receipt of data is confirmed at the NDR parameter when at least one byte has been transferred.

Data reception in ADHOC mode with TRCV < 3.0 (S7-1200 < V4.0)

In the older version of TRCV (Page 3557), ADHOC mode was activated by setting the LEN parameter to "0". In the following example, 10 bytes of data are transferred with call 5.

Call	1	2	3	4	5	6
EN_R						
LEN (max.)	0 = Activation of the ADHOC mode					
NDR						
BUSY						
ERROR						
STATUS	0x7000	0x7001	0x7002	0x7002	0x0000	0x7000
RCVD_LEN	0	0	0	0	10 bytes	0
Connection				Established		
TRCV status	Not active	Receiving	Receiving	Receiving	Receiving	Not active

Data receipt in ADHOC mode with TRCV ≥ 3.0 (S7-1200 ≥ V4.0 or S7-1500)

As of version 3.0 of the TRCV instruction, ADHOC mode is activated with its own parameter (ADHOC).

Call	1	2	3	4	5	6
EN_R						
LEN	Max. 8 KB	Max. 8 KB	Internal			
ADHOC						
NDR						
BUSY						
ERROR						
STATUS	0x7000	0x7001	0x7002	0x7002	0x0000	0x7000
RCVD_LEN	0	0	0	0	10 bytes	0
Connection			Established			
TRCV status	Not active	Receiving	Receiving	Receiving	Receiving	Not active

If an error occurs during execution, as of version 4.0 of the TRCV instructions, there are more STATUS alarms available which can be evaluated accordingly.

If the send operation is performed via a CM/CP, version 4.0 of the TRCV/TURCV instructions behaves differently. In this case, the instruction must be called multiple times until the receipt of data is confirmed by the NDR parameter.

4.1.6.3 OPC UA

Client instructions

OPC UA instructions for client programs

Introduction

Based on program examples, the following sections are showing you how to use the OPC UA instructions for client programs.

The program examples are based on a plant in which an S7-1500 CPU serves as server and another S7-1500 CPU as client:

- The server provides PLC tags for reading and writing as well as a server method.
- The client accesses the values provided and calls the server method.

STEP 7 helps you in supplying the parameters of the instructions when you create a client interface and use the connection parameter assignment for OPC UA. This means you will not have to create many tags manually for supplying the parameters. To find out how to work manually, read the example in the section on the "OPC_UA_TranslatePathList" instruction (Page 3674).

How to use the convenient parameter assignment for OPC UA is described in the following sections for the OPC UA instructions.

Program examples in SCL

The program examples use the Structured Control Language (SCL) programming language.

The complete source code is available for each program example.

Standardized instructions according to PLCopen

The OPC UA instructions for clients have mostly been implemented according to the specification "PLCopen OPC-UA-Client for IEC 61131-3".

Standardization makes it easier to port a user program to a controller of a different manufacturer that also provides the OPC UA instructions according to the PLCopen specification.

If you already know the basic structures and processes of OPC UA clients, you can find your way around more easily.

Many client instructions are working with lists

Many OPC UA client instructions process not only a value but lists.

This means the instruction "OPC_UA_NamespaceGetIndexList", for example, receives a list with namespaces and returns a list with the indexes for these namespaces.

Performance capability of a OPC UA server

OPC UA servers themselves provide information on their performance capability.

The figure below shows the address space of an S7 CPU of the type 1516-3 PN/DP.

In the node "MaxNodesPerRead" you specify, for example, how many PLC tags can be read simultaneously with one call of the "OPC_UA_ReadList" instruction.

The high field limit of the array with which you can supply the "NodeHdls" parameter is limited by the properties of the OPC UA client that is used (see technical specifications of the client).

The following limits apply across all client interfaces (i.e. across all lists):

- 1000 nodes for CPU 1510SP (F), CPU 1511 (C/F/T/TF), CPU 1512C, CPU 1512SP (F), CPU 1513 (F)
- 2000 nodes for CPU 1505 (S/SP/SP F/SP F/SP T/SP TF), CPU 1515 (F/T/TF), CPU 1515 SP PC (F/T/TF), CPU 1516 (F/T/TF)
- 5000 nodes for CPU 1507S (F), CPU 1517 (F/T/TF), CPU 1518 (F)

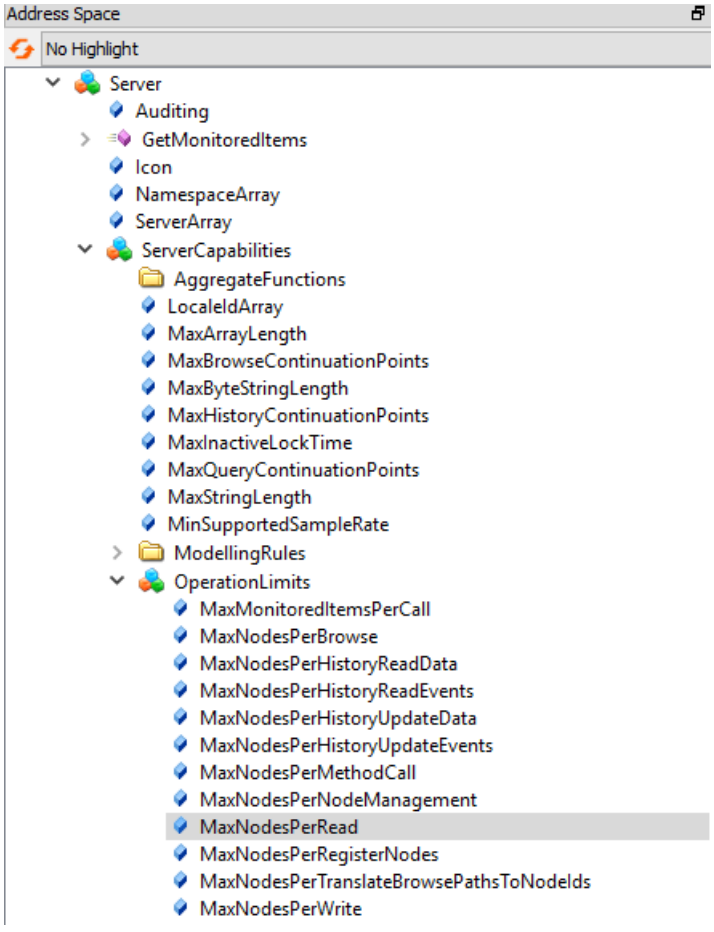

The figure shows a section of the address space of an OPC UA server, shown with the "UaExpert" from Unified Automation.

Error numbers

The error numbers for the instructions are divided up as follows:

- If the error number starts with "16#B0" or "16#C0", it was assigned by Siemens, see Siemens error codes (Page 3768).
  Example: The following error numbers are general numbers that can basically occur for any of the client instructions:

B080_C400	ClientNotEnabled	Client disabled. Enable the client in the TIA Portal.
B080_C500	ClientNotAvailable	Error during initialization of the client
C080_C300	Simatic_OutOfResources	Error at memory allocation or too many instances of the instruction

- If the error number starts with "16#80" it was assigned by the OPC Foundation see Error codes of the OPC Foundation (Page 3770).
- If the error number starts with "16#A0" it was assigned by PLCopen, see PLCopen error codes (Page 3779).
  You can also find a list with error numbers of the PLCopen in section 4 of the specification "PLCopen OPC UA Client for IEC 61131-3". The specification is available from PLCopen.

Preparing data exchange

OPC_UA_Connect: Create connection

Validity

The following description of the "OPC_UA_Connect" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

The "OPC_UA_Connect" instruction establishes a connection to an OPC UA server.

The following figure shows the icon of the instruction in the editor (FBD).

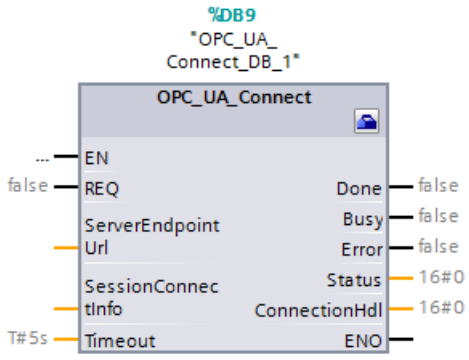

In the figure above, the parameters of the instruction are not supplied yet.

The instruction is used to prepare the data exchange between an OPC UA server and an OPC UA client, see explanation for ① in the figure below:

The instruction returns a connection handle (a numerical reference) which uses the following instructions to refer to the connection.

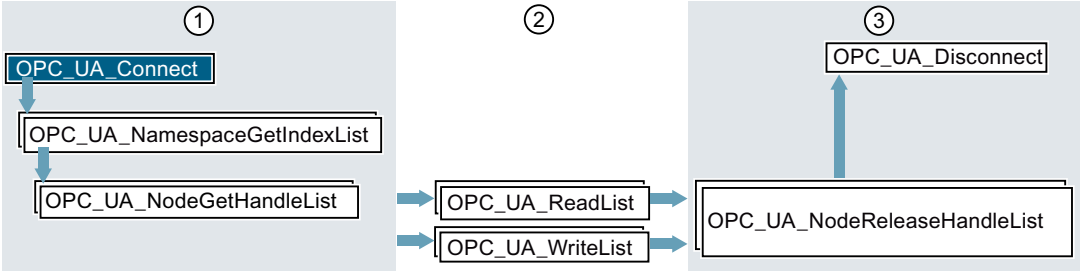

① Instructions for preparation of read and write operations

② Read and write instructions (call the instructions repeatedly depending on the application)

③ Instructions for releasing resources after completed read or write operations

Parameters for "OPC_UA_Connect"

Parameters	Declaration in area	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the instruction.
ServerEndpointUrl	InOut	VARIANT	Pointer to a tag of the type STRING or WSTRING with the address (URL) of the OPC UA server. Only IPv4 addresses are permitted.
SessionConnectInfo	InOut	VARIANT	Pointer to the connection description in a tag with the system data type "SessionConnectInfo (Page 3759)".

Parameters	Declaration in area	Data type	Meaning
Timeout	Input	TIME	Maximum time for the establishment of the connection in milliseconds. The "Timeout" parameter defines the service timeout for lower-level OPC UA service requests that the client is sending to the server. Timeout values smaller than 100 ms or negative values are set to 100 ms, as smaller values are not useful.
Done	Output	BOOL	Execution status parameter: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error numbers for Status" below.
ConnectionHdl	Output	DWORD	Unique identifier for a connection established. This handle is required by other OPC UA instructions as an input parameter.

Error numbers for Status

The "Status" parameter provides information about errors that occurred during execution of the instruction.

The following table explains the error codes:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Instruction finished successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions. • General: Reduce the number of registered OPC UA elements. Use an OPC UA Server diagnostics tool (such as UaExpert from Unified Automation) to ensure that there are no inactive sessions and subscriptions. As countermeasure you can select corresponding timeouts (Subscription Timeout <= Session Timeout) in the client.

Error code (hexadecimal values)	Name of the error	Explanation
8005_0000	OpcUa_BadCommunicationError	Connection timeout. Possible causes: • The server address (ServerEndpointURL) is incorrect or incomplete. • The server does not accept the buffer size sent by the client during connection establishment. • The client's connection resources required to establish the connection are used up. Remedies: • Check and correct the ServerEndpointURL • Check the permitted buffer size. The client of the S7-1500 CPU provides 8192 bytes. The server must accept this value. • If the connection is established and terminated in rapid succession, the frequency must be reduced.

Error code (hexadecimal values)	Name of the error	Explanation
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow. • The network load is too high. • The OPC UA server is not available. • The server may not accept the buffer size sent by the client during connection establishment. Remedies: • Check the URL of the OPC UA server. • Increase the timeout value of the instruction OPC_UA_Connect. • Check the permitted buffer size. The client of the S7-1500 CPU provides 8192 bytes. The server must support this value.
8010_0000	OpcUa_BadTooManyOperations	The request could not be executed because the number of nodes to be read or written was too large. Possible remedy: • Reduce the number of nodes that you want to access with a call.
803C_0000	OpcUa_BadOutOfRange	Range limits violated. Possible remedy for arrays: • Check whether you can access elements of an array that are not defined.
803D_0000	OpcUa_BadNotSupported	Possible causes: • The server address (ServerEndpointURL) is incorrect or incomplete. • Server does not support the requested operation.

Error code (hexadecimal values)	Name of the error	Explanation
8054_0000	OpcUa_BadSecurityModeInsufficient	The server sets higher security requirements (security policy). Possible remedy: • Use a higher security setting for the connection to the server.
8055_0000	OpcUa_BadSecurityPolicyRejected	Server does not support requested Security Policy or Transport Profile
B080_0100	Simatic_BadType_VariantInput1	The data type of the tag to which the first variant parameter of the instruction is pointing is incorrect.
B080_0200	Simatic_BadType_VariantInput2	The data type of the tag to which the second variant parameter of the instruction is pointing is incorrect.
B080_2200	Simatic_BadValue_VariantInput2	The value of the tag to which the second variant parameter of the instruction is pointing is incorrect.
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled. Solution: Activate the OPC UA client in the properties of the CPU (OPC UA > Client area).
B080_C500	Simatic_NothingToDo	Error during initialization of the client.
B080_C600	Simatic_ClientNotAvailable	Error during initialization of the client.
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Error at memory allocation or too many instances of the instruction. Possible remedy: • Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

This is how you use this instruction

The following sections show you how to use the instruction "OPC_UA_Connect" in a program that exchanges data with an OPC UA server.

Requirements

The requirements for the following description are:

- You have created a client interface, see "AUTOHOTSPOT".
- You have configured a connection to an OPC UA server in the properties of this client interface.

Configuring secure connections

You can use the instruction "OPC_UA_Connect" to establish an unsecure connection to an OPC UA server. But you should always use a secure connection.

How to use a client interface

1. In the "Project tree" area, select the CPU that acts as the client.
2. Add a new function block to the "Program blocks" folder.
   In the example, the function block is called "ReadFromProductionline".
   Used language: SCL.
3. Use drag-and-drop to move the instruction "OPC_UA_Connect" from the folder "Instructions > Communication > OPC UA > OPC UA Client" to the editor.
4. Select the call as multi-instance.
   STEP 7 creates an instance of this instruction and calls it "OPC_UA_Connect_Instance".
5. Click on the icon for "Start configuration" at the "OPC_UA_Connect_Instance" instruction.

   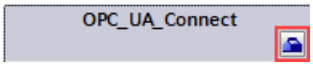

   STEP 7 opens the "Configuration" tab in the Inspector window.
6. Under "Select client interface for OPC UA connection" you select the client interface that you want to use for the instruction.
   In the example, the client interface is called "Productionline".

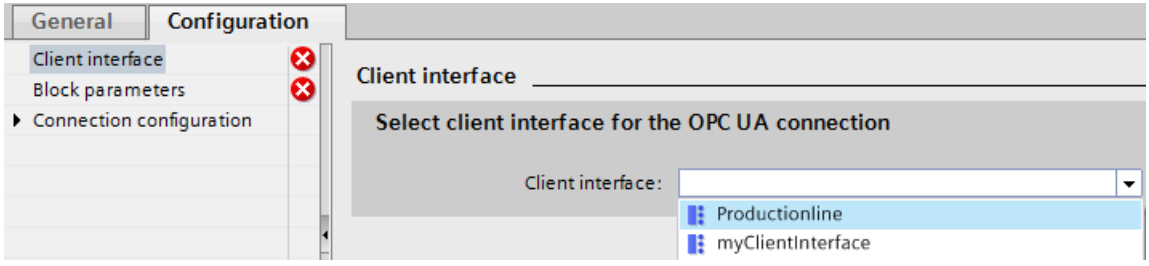

   STEP 7 automatically supplies the parameters of the instruction with the values you have configured for the client interface.
7. Click on "Block parameters" and assign tags manually to the remaining parameters (REQ, Busy, Done, Error, Status).
   STEP 7 adds the selected tags to the function call.

Calling the instruction

The following source code shows you how to use the instruction "OPC_UA_Connect" to establish a connection to an OPC UA server.

You can find the complete program in the section "AUTOHOTSPOT".

The example program is divided into multiple program sections (cases) by a CASE instruction. The #State tag controls which program section is being executed.

In the first program section, a connection is established:

SCL

```
1:  // case 1, connect to server
            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_Connect_Instance(ServerEndpointUrl :=
"Productionline_Configuration".Connection.ServerEndpointUrl,
                                     SessionConnectInfo :=
"Productionline_Configuration".Connection.ConnectInfo,
                                     ConnectionHdl =>
"Productionline_Configuration".Connection.ConnectionHdl,
                                     Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                     REQ := #Req,
                                     Done => #Done,
                                     Busy => #Busy,
                                     Error => #Error,
                                     Status => #Status);


            IF #Done = TRUE THEN
                #State := #State + 1;
            END_IF;

            IF #Error = TRUE THEN
                // Did we get a connection handle?
                IF "Productionline_Configuration".Connection.ConnectionHdl <> 0 THEN
                    // We have to release all resources in the server and disconnect
                    #State := 100;
                    //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                ELSE
                    #State := 99;
                END_IF;

                #Mem_Status := #Status;
                // set parameter REQ of OPC_UA_Connect to FALSE
                #OPC_UA_Connect_Instance(ServerEndpointUrl :=
"Productionline_Configuration".Connection.ServerEndpointUrl,
                                         SessionConnectInfo :=
"Productionline_Configuration".Connection.ConnectInfo,
                                         REQ := FALSE,
                                         ConnectionHdl =>
"Productionline_Configuration".Connection.ConnectionHdl,
                                         Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);
            END_IF;
```

The example program calls the "OPC_UA_Connect" instruction for the following reasons:

- To establish the connection to an OPC UA server.
- If an error occurs, the REQ parameter is set to the value "FALSE".

Asynchronous execution

The instruction "OPC_UA_Connect" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction

This program part establishes a connection to the OPC UA server that you have entered during the configuration of the client interface.

The tag "#State" contains the value "1" at the start of the program.

If the "OPC_UA_Connect" instruction is not yet executed, #Busy is FALSE, so that #Req is set to TRUE. This starts the instruction. In the next cycle, #Req is FALSE.

If the output parameter "Done" is TRUE, a connection could be established. The success status increases the value of the #State tag by one. This means the next program section (2:) is executed in the next cycle, see OPC_UA_NamespaceGetIndexList: Read namespace indexes (Page 3653). The success status also indicates the numerical reference for the established connection at the "ConnectionHdl" output parameter.

If an error has occurred, the "Error" output parameter has the value TRUE.

- If a connection handle is returned in the event of an error, resources must be released in the server. The #State tag is assigned the value 100 for this.
- If no connection handle is returned in the event of an error, the #State tag is assigned the value 99.

The program sections 99 and 100 are reserved for error handling.

See also

Program example for reading PLC tags (Page 3787)

OPC_UA_TranslatePathList: Read node parameter (Page 3674)

Siemens error codes (Page 3768)

OPC UA instructions for client programs (Page 3640)

OPC_UA_NamespaceGetIndexList: Read namespace indexes

Validity

The following description of the "OPC_UA_NamespaceGetIndexList" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

You use the "OPC_UA_NamespaceGetIndexList" instruction to request the current indexes of the namespaces in an OPC UA server.

The following figure shows the icon of the instruction in the editor (FBD).

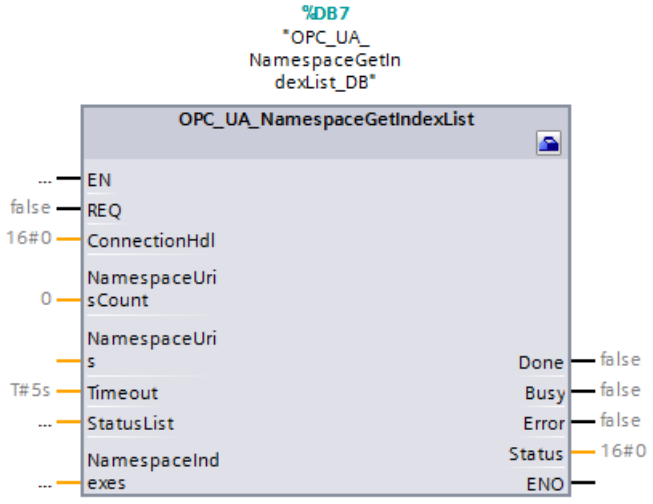

In the figure above, the parameters of the instruction are not supplied yet.

The instruction "OPC_UA_NamespaceGetIndexList" is used to prepare the data exchange with an OPC UA server, see ① in the figure below.

To read or write a PLC tag, or to call a method, you must know the index of the namespace in which the PLC tag is located. The index is a part of the address (NodeID) of this PLC tag in the address space of the OPC UA server.

The instruction returns a list with the indices for the namespaces of the OPC UA server.

This list is necessary to request the NodeIDs that can be used to send the read and write jobs with the "OPC_UA_NodeGetHandleList" instruction.

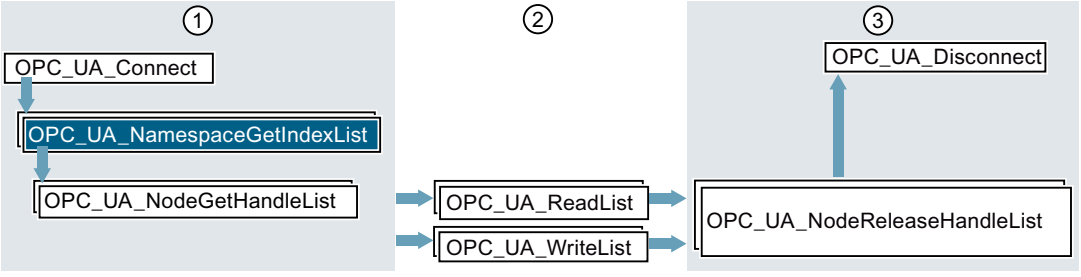

① Instructions for preparation of read and write operations

② Read and write instructions

③ Instructions for releasing resources after completed read or write operations

Parameters for "OPC_UA_NamespaceGetIndexList"

Table 4-73 The parameters of the instruction "OPC_UA_NamespaceGetIndexList"

Parameters	Declaration in area	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the instruction.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established. You get the handle from the instruction OPC_UA_Connect: Create connection (Page 3643).
NamespaceUrisCount	Input	UINT	Number of namespaces in the NamespaceUris parameter
NamespaceUris	InOut	VARIANT	Pointer to an array of the type STRING or WSTRING. The array contains the URIs of the individual namespaces whose indexes are to be requested.
Timeout	Input	TIME	Maximum time for execution of the instruction in milliseconds. See also the explanation for this parameter in OPC_UA_Connect: Create connection (Page 3643).
StatusList	InOut	VARIANT	Pointer to an array of the type DWORD (optional). The array contains the error codes for the individual namespaces. For each namespace, the system specifies whether or not a corresponding index could be found.
NamespaceIndexes	InOut	VARIANT	Pointer to an array of the type UINT. The array contains the indexes of the individual namespaces that have been requested from the OPC UA server.
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error numbers for Status" below

Error numbers for Status

The "Status" parameter provides information about errors that occurred during execution of the instruction.

The following table shows a summary of error codes for this instruction:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Instruction finished successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.
8009_0000	OpcUa_BadUnknownResponse	Server does not respond with the expected number of namespace array elements
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.
800F_0000	OpcUa_BadNothingToDo	NamespaceUrisCount is 0

Error code (hexadecimal values)	Name of the error	Explanation
8010_0000	OpcUa_BadTooManyOperations	The request cannot be executed because too many namespaces are to be read. Possible remedy: • Reduce the number of namespaces that you want to access with a call.
809E_0000	OpcUa_BadDataUnavailable	The indexes of the namespaces are not currently available.
B080_0100	Simatic_BadType_VariantInput1	The data type of the tag to which the first variant parameter of the instruction is pointing is incorrect.
B080_0200	Simatic_BadType_VariantInput2	The data type of the tag to which the second variant parameter of the instruction is pointing is incorrect.
B080_0300	Simatic_BadType_VariantInput3	The data type of the tag to which the third variant parameter of the instruction is pointing is incorrect.
B080_0400	Simatic_BadType_VariantInput4	The data type of the tag to which the fourth variant parameter of the instruction is pointing is incorrect.
B080_0500	Simatic_BadType_VariantInput5	The data type of the tag to which the fifth variant parameter of the instruction is pointing is incorrect.
B080_1100	Simatic_ArrayElements_TooMany	NamespaceUrisCount > MAX_ELEMENTS_NAMESPACES
B080_3100	BadNumElements_VariantInput1	Number of array elements in variant 1 is smaller than the number stored in the count parameter of the instruction (e.g. NodeIDCount for OPC_UA_NodeGetHandleList or NamespaceUrisCount for OPC_UA_NamespaceGetIndexList, etc.).
B080_3200 B080_3300 B080_3400 B080_3500	BadNumElements_VariantInput2 ... BadNumElements_VariantInput5	Number of array elements in variant 2 (... 5) is less than the number stored in the Count parameter of the instruction.
B080_C500	Simatic_NothingToDo	Nothing to do: The instruction is using a list that contains no elements (namespaces).
C080_C300	Simatic_InsufficientResources	Insufficient resources. Either (a) error upon memory allocation or (b) too many SFB instances
For more error codes, see AUTOHOTSPOT.		

Error numbers for "StatusList"

The parameter contains an error code for each individual namespace.

The following table explains the error codes:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	The namespace is found. The index of the namespace is entered in the array to which the "NamespaceIndexes" parameter is pointing.
806F_0000	OpcUA_BadNoMatch	The namespace is not found.
For more error codes, see AUTOHOTSPOT.		

This is how you use this instruction

Based on a program example, this section shows you how to use the instruction "OPC_UA_NamespaceGetIndexList" in a program that exchanges data with an OPC UA server.

Requirements

The requirements for the following description are:

- You have created a client interface, see "AUTOHOTSPOT".
- You have configured a connection to an OPC UA server in the properties of this client interface, see "AUTOHOTSPOT".

In addition, the following requirement must be met for the instruction "OPC_UA_NamespaceGetIndexList":

- A handle for connection to an OPC UA server is available.
  You get the connection handle by calling the instruction OPC_UA_Connect (Page 3643).

Function of the instruction

For the exchange of data, you require the indexes of the namespaces that contain the tags that you want to read or write using your program.

You also need the current indexes of the namespaces that contain the methods for calling methods.

With an OPC UA server, the assignment, according to the OPC UA specification, between the namespace and associated index is not fixed and can change, for example, if new namespaces are added or namespaces are omitted.

You therefore need to request the current indexes from the OPC UA server before you get the node handle using the instruction OPC_UA_NodeGetHandleList. You do this with the instruction "OPC_UA_NamespaceGetIndexList".

What information does the instruction return?

The instruction returns the following information:

- A list of the indexes for the individual namespaces ("NamespaceIndexes" parameter). The order of the indexes corresponds to the order of the namespaces in the "NamespaceUris" parameter.
- A list of error messages ("StatusList" parameter).
  Each error message in this list relates to the corresponding namespace in the "NamespaceUris" parameter.
  For each individual namespace, you can to check whether the index for that namespace can be successfully returned.

Determining namespaces and their indexes with UaExpert

To use the instruction, the namespace URI (Uniform Resource Indentifier) must be known. You can determine namespace URIs and namespace indexes (Namespace URI and Node Namespace Index) with UaExpert, as the following example shows:

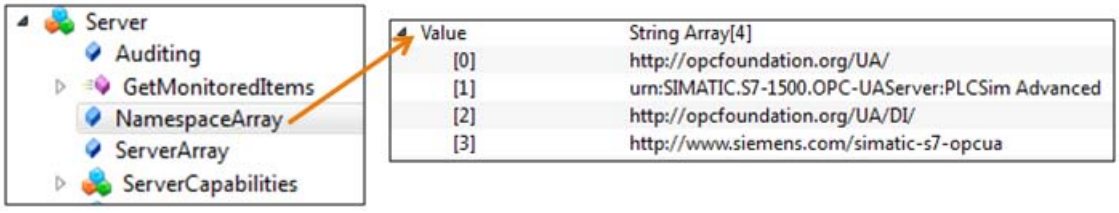

How to use a client interface

1. In the "Project tree" area, select the CPU that acts as the client.
2. Select the function block that is to contain the client instruction in the "Program blocks" folder.
   In the example, the function block is called "ReadFromProductionline".
   Selected language: SCL
3. Use drag-and-drop to move the instruction "OPC_UA_NamespaceGetIndexList" from the folder "Instructions > Communication > OPC UA > OPC UA Client" to the editor.
4. Select the call as multi-instance.
   STEP 7 creates an instance of this instruction and calls it "OPC_UA_NamespaceGetIndexList_Instance".
5. Click on the icon for "Start configuration" at the "OPC_UA_NamespaceGetIndexList_Instance" instruction.
   STEP 7 opens the "Configuration" tab in the Inspector window.

6. Under "Select client interface for OPC UA interface" you select the client interface that you want to use for the instruction.
   In the example, the client interface is called "Productionline". see "AUTOHOTSPOT".
   STEP 7 now automatically supplies the parameters of the instruction with the values you have configured for the client interface.
7. Click on "Block parameters" and assign tags manually to the remaining parameters REQ, Busy, Done, Error, Status.
   STEP 7 adds the selected tag to the function call.

Calling the instruction

The following source code shows you how to use the instruction "OPC_UA_NamespaceGetIndexList" to request the indexes for the namespaces that currently used by the OPC UA server.

You can find the complete program in the section "Program example for reading PLC tags (Page 3787)".

The example program is divided into multiple sections (cases) by a CASE instruction.

In the second program section, the indexes of the namespaces are requested:

SCL

```
2:  // case 2, get index for namespace

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_NamespaceGetIndexList_Instance(NamespaceUris :=
"Productionline_Configuration".Namespaces.NamespaceURIs,
                                                   ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                   NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes,
                                                   StatusList :=
"Productionline_Configuration".Namespaces.NamespaceStatusList,
                                                   NamespaceUrisCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                   Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                   REQ := #Req,
                                                   Done => #Done,
                                                   Busy => #Busy,
                                                   Error => #Error,
                                                   Status => #Status);


            IF #Done = TRUE THEN
               IF "Productionline_Configuration".Namespaces.NamespaceStatusList[0] = 0 THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at NamespaceGetIndexList';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_NamespaceGetIndexList_Instance(NamespaceUris :=
"Productionline_Configuration".Namespaces.NamespaceURIs,
                                                       StatusList :=
"Productionline_Configuration".Namespaces.NamespaceStatusList,
                                                       NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes,
                                                       REQ := FALSE);
            END_IF;
```

The example program calls the "OPC_UA_NamespaceGetIndexList" instruction for the following reasons:

- To request the index for a namespace.
- If an error occurs, the REQ parameter is set to the value FALSE.

Asynchronous execution

The instruction "OPC_UA_NamespaceGetIndexList" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction

The figure above shows case 2. This part of the program requests the indexes for the namespaces.

If the #State tag is 2, then case 2 is executed.

If the "OPC_UA_NamespaceGetIndexList" instruction is not yet executed, #Busy is FALSE an the "#Req" tag is set to TRUE. This starts the instruction. In the next cycle, #Req is FALSE.

If the output parameter "Error" is TRUE, an error occurred during execution of the instruction. This sets the #State tag to the value 100. This case is reserved for troubleshooting.

If the output parameter "Done" is TRUE, the instruction was executed successfully. The success status increases the value of the "#State" tag by one. This means the next program section - case 3 (handles for read access and write access) - is executed in the next cycle. Case 3 is described in OPC_UA_NodeGetHandleList (Page 3663).

Note

Execution of the instruction

It is possible that the instruction is successfully executed ("Error" parameter not set, "Done" parameter set) but no index can be returned for a specific namespace.

Therefore, there is a check in the example to determine whether an index has been found for the namespace to be queried. If an index is found, the first element of the array to which the "StatusList" parameter is pointing contains the value 0.

See also

OPC_UA_TranslatePathList: Read node parameter (Page 3674)

OPC_UA_NodeGetHandleList: Get handles for read and write access

Validity

The following description of the "OPC_UA_NodeGetHandleList" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

You use the "OPC_UA_NodeGetHandleList" instruction to register PLC tags on an OPC UA server.

The following figure shows the icon of the instruction in the editor (FBD).

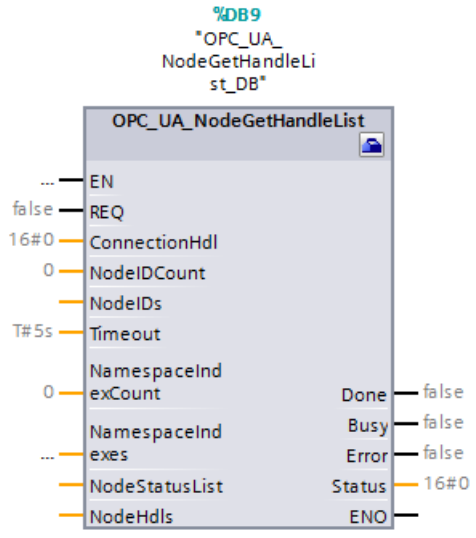

In the figure above, the parameters of the instruction are not supplied yet.

The instruction "OPC_UA_NodeGetHandleList" is used to prepare the data exchange with an OPC UA server, see ① in the figure below:

You use this instruction to register PLC tags on the server that you want to read or write.

This step optimizes later access with the instructions "OPC_UA_ReadList" and "OPC_UA_WriteList".

The instruction returns a list with numerical references (handles) for the registered PLC tags.

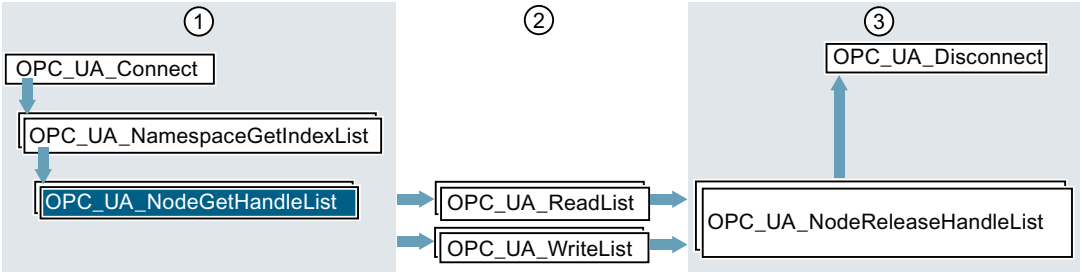

① Instructions for preparation of read and write operations

② Read and write instructions

③ Instructions for releasing resources after completed read or write operations

Parameters for "OPC_UA_NodeGetHandleList"

Table 4-74 The parameters of the instruction "OPC_UA_NodeGetHandleList"

Parameters	Declaration in area	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the instruction.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established. You get the handle from the instruction OPC_UA_Connect.
NodeIDCount	Input	UINT	Number of elements in the NodeIDs parameter
NodeIDs	InOut	VARIANT	Pointer to an array with elements of the type OPC_UA_NodeId. The array contains the NodeIDs of the tags for which the OPC UA server is to provide numeric references (handles) (see "Function of the block" above).
Timeout	Input	TIME	Maximum time for execution of the instruction in milliseconds. See also the explanation for this parameter in AUTOHOTSPOT.
NamespaceIndexCount	Input	UINT	Number of elements in the NamespaceIndexes parameter

Parameters	Declaration in area	Data type	Meaning
NamespaceIndexes	InOut	VARIANT	Pointer to an array of the type DWORD. This parameter is not required if there are no elements in the NamespaceIndexCount parameter (NamespaceIndexCount = 0). The array is used to assign new indexes to the namespace indexes in the NodeIDs parameter. Example: If there is a value of 10 in NamespaceIndexes[4], all NodeIDs in the "NodeIDs" parameter that previously had the namespace index 4 are assigned the namespace index 10.
NodeStatusList	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the error codes for the NodeIDs of the individual tags; see "Error numbers for NodeStatusList" below. For each NodeID, the system specifies whether or not a corresponding handle could be found. It is possible that the function block was successfully executed (Error parameter not set) but no handle could be found for a specific NodeID (tag). A NodeID consists of a namespace index, a tag name (identifier) and an identifier type. See OPC_UA_NodeID.
NodeHdls	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the handles of the individual NodeIDs that have been requested from the OPC UA server (see "Function of the block" above).
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error numbers for Status" below

Error numbers for Status

The "Status" parameter provides information about errors that occurred during execution of the instruction.

The following table shows a summary of error codes for this instruction:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Instruction finished successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.
8009_0000	OpcUa_BadUnknownResponse	The server has sent has response that could not be recognized.
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.
8010_0000	OpcUa_BadTooManyOperations	The Request could not be executed because it contained too many instructions.
8033_0000	OpcUa_BadNodeIdInvalid	The syntax of the node is invalid.
8034_0000	OpcUa_BadNodeIdUnknown	The ID of the node (NodeId) refers to a node that does not exist in the address space of the server.
8035_0000	OpcUa_BadAttributeIdInvalid	The attribute does not exist in this node.
B080_0100	Simatic_BadType_VariantInput1	The data type of the tag to which the first variant parameter of the instruction is pointing is incorrect.
B080_0200	Simatic_BadType_VariantInput2	The data type of the tag to which the second variant parameter of the instruction is pointing is incorrect.

Error code (hexadecimal values)	Name of the error	Explanation
B080_0300	Simatic_BadType_VariantInput3	The data type of the tag to which the third variant parameter of the instruction is pointing is incorrect.
B080_0400	Simatic_BadType_VariantInput4	The data type of the tag to which the fourth variant parameter of the instruction is pointing is incorrect.
B080_0500	Simatic_BadType_VariantInput5	The data type of the tag to which the fifth variant parameter of the instruction is pointing is incorrect.
B080_1100	Simatic_ArrayElements_TooMany	NodeIDCount > MAX_ELEMENTS_NODELIST
B080_3100	BadNumElements_VariantInput1	Number of array elements in variant 1 is smaller than the number stored in the count parameter of the instruction (e.g. NodeIDCount for OPC_UA_NodeGetHandleList or NamespaceUrisCount for OPC_UA_NamespaceGetIndexList, etc.).
B080_3200 B080_3300 B080_3400 B080_3500	BadNumElements_VariantInput2 ... BadNumElements_VariantInput5	Number of array elements in variant 2 (... 5) is less than the number stored in the Count parameter of the instruction.
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled.
B080_C500	Simatic_NothingToDo	Nothing to do: The instruction is using a list that contains no elements.
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Solution: Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

Error numbers for "NodeStatusList"

The "NodeStatusList" parameter contains an error code for each individual NodeID (tag).

The following table explains the error codes:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	The PLC tag (NodeId) was found. The Handle of the tags was entered in the Array to which the "NodeHdls" parameter is pointing.
8033_0000	OpcUa_BadNodeIdInvalid	The syntax of the NodeId is incorrect.
8034_0000	OpcUa_BadNodeIdUnknown	The NodeId refers to a node (tag) that does not exist on the OPC UA server.

Error code (hexadecimal values)	Name of the error	Explanation
A000_0309	PLCopenUA_Bad_OutOfHandles	The NodeId was not registered because the maximum number of nodes (PLCOPEN_MAXREGISTEREDNODES) was reached.
For more error codes, see AUTOHOTSPOT.		

This is how you use this instruction

Based on a program example, this section shows you how to use the instruction "OPC_UA_NodeGetHandleList" in a program that exchanges data with an OPC UA server.

Requirements

The following description assumes that:

- You have created a client interface, see "AUTOHOTSPOT".
- You have configured a connection to an OPC UA server in the properties of this client interface, see "AUTOHOTSPOT".

In addition, the following requirements must be met for the instruction "OPC_UA_NodeGetHandleList":

- A handle for connection to an OPC UA server is available.
  You get the connection handle from the instruction OPC_UA_Connect.
- Indexes of the namespaces in which the tags that you want to read or write with the client are located.
  You get the indexes from the instruction OPC_UA_NamespaceGetIndexList.

Function of the instruction

To optimize access to tags, OPC UA offers the "RegisterNodes" service for repeated accesses.

Servers can use this service to prepare optimized access to tags. The "OPC_UA_NodeGetHandleList" instruction implicitly calls this service to prepare the server for optimized access ("Registered Read/Write" in OPC UA language).

This is how you obtain numerical references (handles) from the OPC UA server through the call of the "OPC_UA_NodeGetHandleList" instruction. You use these handles with the "OPC_UA_ReadList" or "OPC_UA_WriteList" instructions, which correspond to an optimized access with "Registered Read/Write".

What information is required?

The instruction requires the following information to request the numeric references (handles):

- A list of the tags that are to be read or set ("NodeIds" parameter, see system data type "OPC_UA_NodeId").
  This list contains the following information for each tag:
  - Index of the namespace in which the tag is located.
  - Tag identifier (tag name).
  - Type of identifier

What information does the instruction return?

The instruction returns the following information:

- A list of the handles for the tags ("NodeHdls").
  The order of the handles corresponds to the order of the tags in the NodeIDs parameter.
- A list of error messages ("NodeStatusList" parameter).
  Each error message in this list relates to the corresponding tag in the NodeIDs parameter. For each individual tag, you need to check whether the OPC UA server was able to return a handle.

How to use a client interface

1. In the "Project tree" area, select the CPU that acts as the client.
2. Select the function block that is to contain the client instruction in the "Program blocks" folder.
   In the example, the function block is called "ReadFromProductionline".
   Selected language: SCL.
3. Use drag-and-drop to move the instruction "OPC_UA_NodeGetHandleList" from the folder "Instructions > Communication > OPC UA > OPC UA Client" to the editor.
4. Select the call as multi-instance.
   STEP 7 creates an instance of this instruction and calls it "OPC_UA_NodeGetHandleList_Instance".
5. Click on the icon for "Start configuration" at the "OPC_UA_NodeGetHandleList_Instance" instruction.
   STEP 7 opens the "Configuration" tab in the Inspector window.
6. Under "Select client interface for OPC UA interface", you select the client interface that you want to use for the instruction.
   In the example, the client interface "Productionline" is selected, which was created for the example plant, see "AUTOHOTSPOT".
7. Click "File access" and select a read list.
   In the example: Read list "ReadListProduct".
   STEP 7 now automatically supplies the parameters of the instruction with the values you have configured for the client interface.
8. Click on "Block parameters" and assign tags manually to the remaining parameters REQ, Busy, Done, Error, Status.
   STEP 7 adds the selected tag to the function call.

Calling the instruction (initial call)

You can find the complete program in the section "AUTOHOTSPOT".

The program example uses the read list "ReadListProduct", which contains the following PLC tags:

- NewProduct
- ProductNumber

The program receives numerical references (handles) from the server for these tags. The section "AUTOHOTSPOT" shows you how to create a client interface and how to add a read list with PLC tags to it.

The example program is divided into multiple sections (cases) by a CASE instruction.

In the third program section, the handles of the nodes (PLC tags) that are to be read or written are requested:

SCL

```
3:  // case 3, get handles for nodes

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_NodeGetHandleList_Instance(NodeIDs :=
"Productionline_Configuration".ReadLists."ReadListProduct".Nodes,
                                               NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                               NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                               REQ := #Req,
                                               ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                               Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                               NodeIDCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount,
                                               NamespaceIndexCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                               NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes,
                                               Done => #Done,
                                               Error => #Error,
                                               Busy => #Busy,
                                               Status => #Status);
```

Asynchronous execution

The instruction "OPC_UA_NodeGetHandleList" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction (initial call)

The program part in the figure above calls the "OPC_UA_NodeGetHandleList" instruction to request handles for PLC tags.

If the instruction is not yet executed, #Busy is FALSE, so that the "#Req" tag is set to the value TRUE. This starts the instruction. In the next cycle, #Req is FALSE.

Calling the instruction (troubleshooting)

The figure below shows evaluation of the "Done" and "Error" parameters.

SCL

```
            IF #Done = TRUE THEN
                IF "Productionline_Data"."ReadListProduct".NodeStatusList[0] = 0 AND
"Productionline_Data"."ReadListProduct".NodeStatusList[1] = 0
                THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at NodeGetHandleList';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #OPC_UA_NodeGetHandleList_Instance(NodeIDs :=
"Productionline_Configuration".ReadLists."ReadListProduct".Nodes,
                                                   NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                                   NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                                   REQ := FALSE,
                                                   ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                   Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                   NodeIDCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount,
                                                   NamespaceIndexCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                   NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes);
            END_IF;
```

Explanation of the call of the instruction (troubleshooting)

If an error occurs, the value of the "Error" output parameter is set to TRUE. This sets the value of the "#State" tag to 100. This case is reserved for troubleshooting. The example program also calls the instruction "OPC_UA_NodeGetHandleList" to set the REQ parameter to FALSE.

If the output parameter "Done" is TRUE, then the instruction was executed successfully.

Please note: It is possible that the instruction was successfully executed ("Error" parameter not set, "Done" parameter set) but no handle could be returned for a specific PLC tag.

In the example program, therefore, there is a check to determine whether a handle is returned for each PLC tag.

- If a handle returns for the first tag, then the first element of the array to which the "NodeStatusList" parameter is pointing contains the value 0.
- If a handle returns for the second tag, then the second element of the array to which the "NodeStatusList" parameter is pointing contains the value 0.

If both conditions are met, the output parameter "NodeHdls" points to a valid list with the handles for the registered PLC tags.

The value of the "#State" tag is increased by one. This means the next program section - case 4 - is executed in the next cycle.

Case 4 is described in OPC_UA_Readlist (Page 3685).

Note

If you register a large number of tags in your application with OPC_UA_NodeGetHandleList, it can be useful always to increase #State by one when #Done = TRUE. You should then check when using the requested values whether the individual values are valid.

What is the purpose of the parameter "NamespacesIndexes"?

The parameter "NamespaceIndexes" implements the indexes for the namespaces.

Implementation of indexes is necessary, for example, when you read the node identifiers in your user program from an external list, for example, from an exported OPC UA XML file.

In this case you must enter the index that the OPC UA server is currently using for the namespace for all identifiers in the list.

Besides, the indexes of the namespaces may change so that you have to adapt the list each time.

Automatic change of the indexes

You can automatically implement the indexes of the namespaces: To do so, use the parameters "NamespaceIndexes" and "NamespacesIndexCount" of the "OPC_UA_NodeGetHandleList" instruction.

Proceed as follows:

1. Check which namespace is assigned to the NodeIDs that you read in from a list.
   In the example, the node IDs are assigned namespace index 0.
   In the example, the node IDs are in the Siemens namespace "http://www.siemens.com/simatic-s7-opcua".
2. Create an array of the type "WString".
   In the example, the array is called "myNamespaces".
3. Enter the Siemens namespace "http://www.siemens.com/simatic-s7-opcua" under "myNamespaces[0]" (also in array element 0).

4. Supply the "NamespaceUris" parameter of the instruction "OPC_UA_NamespaceGetIndexList" with the new array "myNamespaces". The instruction provides the result (the current index) at the parameter "NamespaceIndexes".
5. Create an array of the type "UInt" for the result. In the example, the array is called "myNamespaceIndexes".
6. Supply the "NamespaceIndexes" parameter of the instruction "OPC_UA_NamespaceGetIndexList" with the new array "myNamespaceIndexes".
7. Also assign this array "myNamespaceIndexes" to the "NamespacesIndexes" parameter of the instruction "OPC_UA_NodeGetIndexList". This instruction then reads the current index of the Siemens namespace from "myNamespaceIndexes[0]" and uses it (instead of the original namespace index 0). This implements the index.
8. Assign the value 1 to the "NamespacesIndexCount" parameter of the instruction "OPC_UA_NodeGetIndexList" (we are using only one namespace in the example).

The following program code shows an implementation of the example:

SCL

```
#OPC_UA_NodeGetHandleList_Instance(NodeIDs := #TargetNodeIds,
                                   NodeStatusList := #NodeStatusList,
                                   NodeHdls := #NodeHdls,
                                   ConnectionHdl := #ConnectionHdl,
                                   Timeout := T#6S,
                                   NodeIDCount := 2,
                                   REQ := #Req,
                                   Busy => #Busy,
                                   Done => #Done,
                                   Error => #Error,
                                   Status => #Status,
                                   NamespaceIndexCount := 1,
                                   NamespaceIndexes := #myNamespaceIndexes);
```

The parameters "NamespaceIndexes" and "NamespaceIndexCount" are optional.

The parameters are optional.

This means they must not be supplied if an index must not be implemented.

An index must not be implemented if you are using a client interface or if you have determined the NodeIDs by browsing (navigating) on the server (with the instruction "OPC_UA_TranslatePathList".)

See also

OPC_UA_TranslatePathList: Read node parameter (Page 3674)

OPC UA instructions for client programs (Page 3640)

OPC_UA_TranslatePathList: Read node parameter

Validity

The following description of the "OPC_UA_TranslatePathList" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

You need this instruction when you only know the symbolic access path and want to determine the corresponding NodeIDs dynamically. To do so, you navigate (browse) in the address space of the OPC UA server with the instruction "OPC_UA_TranslatePathList".

You do not need this instruction when you are working with a client interface.

You also do not need this instruction if you know the NodeIDs of the OPC UA nodes that you want to read, write or call.

The following figure shows the icon of the instruction in the editor (FBD).

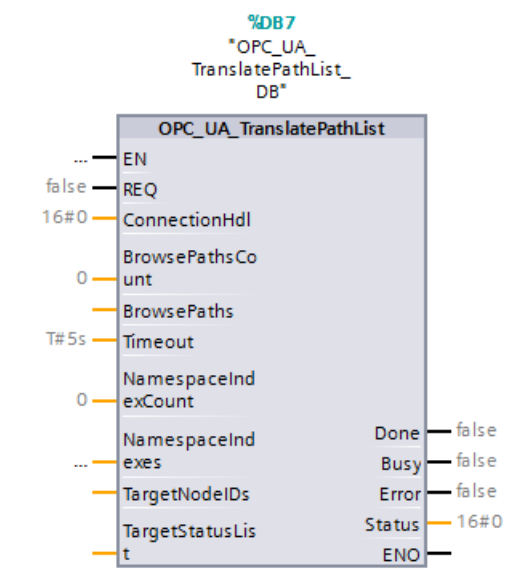

In the figure above, the parameters of the instruction are not supplied yet.

OPC_UA_TranslatePathList is used to prepare the data exchange, see ① in the figure below.

You use the instruction "OPC_UA_TranslatePathList" to request NodeIDs of nodes from the server.

To do so, you define the browse paths.

The instruction returns the NodeIDs of the nodes you are looking for.

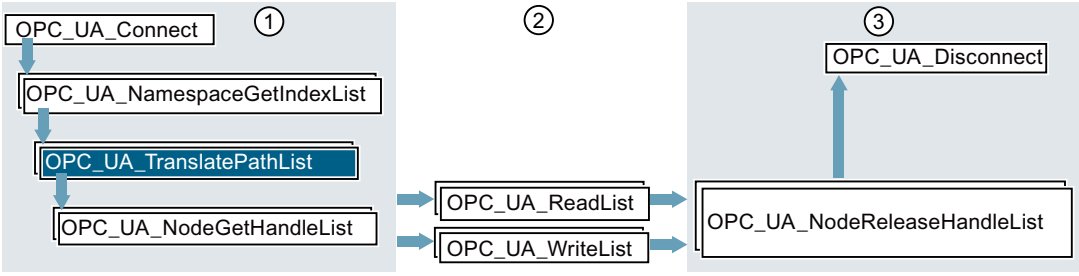

① Instructions for preparation of read and write operations

② Read and write instructions

③ Instructions for releasing resources after completed read or write operations

Parameters for "OPC_UA_TranslatePathList"

Table 4-75 The parameters of the instruction "OPC_UA_TranslatePathList"

Parameters	Declaration in area	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the instruction.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established. You get the handle from the instruction OPC_UA_Connect.
BrowsePathsCount	Input	UINT	Number of elements in BrowsePaths (max. 10).
BrowsePaths	InOut	VARIANT	Pointer to an array of the type OPC_UA_BrowsePath. The array contains the individual browse paths you want to follow.
Timeout	Input	TIME	Maximum time for execution of the instruction in milliseconds. See also the explanation for this parameter in AUTOHOTSPOT.
NamespaceIndexCount	Input	UINT	Number of elements in NamespaceIndexes.
NamespaceIndexes	InOut	VARIANT	Pointer to an array of the type UINT. The array implements the indexes for the namespaces. Example: If there is a value of 10 in NamespaceIndexes[4], all NodeIDs in the "NodeIDs" parameter that previously had the namespace index 4 are assigned the namespace index 10.
TargetNodeIDs	InOut	VARIANT	Pointer to an array of the type OPC_UA_NodeId. The array contains the NodeIds found.

Parameters	Declaration in area	Data type	Meaning
TargetStatusList	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the error numbers for the individual browse paths. The first element in the array refers to the first element in the BrowsePaths parameter, etc.; see "Error numbers for TargetStatusList"
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error numbers for Status" below

"NamespacesIndexes" parameter

The parameter "NamespaceIndexes" is optional.

You only need this parameter if you want to use a NodeId from a namespace other than CORE (which means an index that is not 0) in the BrowsePaths in "OPC_UA_RelativePathElement" in ReferenceTypeId, and you received this NodeID offline.

You can a description of the parameter in the section "OPC_UA_NodeGetHandleList".

The program code below does not use this parameter: The "NamespaceIndexCount" parameter is set to 0. This means the instruction ""OPC_UA_TranslatePathList" does not evaluate the parameter "NamespaceIndexes"".

Error numbers for Status

The "Status" parameter provides information about errors that occurred during execution of the instruction.

The following table shows a summary of error codes for this instruction:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Instruction finished successfully.
8002_0000	OpcUa_BadInternalError	An error occurred as a result of an internal programming error in the function block. Please contact Support.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.
8009_0000	OpcUa_BadUnknownResponse	Server did not respond with the expected number of results
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.
800F_0000	OpcUa_BadNothingToDo	Nothing to do: The OPC UA server received an empty list with no instructions from the OPC UA client.
8010_0000	OpcUa_BadTooManyOperations	The request could not be executed because it contained too many instructions.

Error code (hexadecimal values)	Name of the error	Explanation
B080_0100	Simatic_BadType_VariantInput1	The data type of the tag to which the first variant of the instruction is pointing is incorrect.
B080_0200	Simatic_BadType_VariantInput2	The data type of the tag to which the second variant of the instruction is pointing is incorrect.
B080_0300	Simatic_BadType_VariantInput3	The data type of the tag to which the third variant of the instruction is pointing is incorrect.
B080_0400	Simatic_BadType_VariantInput4	The data type of the tag to which the fourth variant of the instruction is pointing is incorrect.
B080_1100	Simatic_ArrayElements_TooMany	BrowsePathscount > MAX_ELEMENTS_RELATIVEPATH
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled.
B080_C500	Simatic_NothingToDo	Nothing to do: Parameter "BrowsePathsCount" is not supplied or "BrowsePathsCount" = 0.
B080_C600	Simatic_ClientNotAvailable	Error during initialization of the client
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Possible remedy: • Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

Error numbers for "TargetStatusList"

The "TargetStatusList" parameter contains an error code for each individual NodeID (tag).

The following table explains the error codes:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Value successfully read.
8033_0000	OpcUa_BadNodeIdInvalid	The syntax of the NodeID is incorrect.
8034_0000	OpcUa_BadNodeUnknown	The node handle transferred is not known.
8035_0000	OpcUa_BadAttributeInvalid	The attribute required is not supported for the specified node.
8037_0000	OpcUa_BadIndexRangeNoD	There is no data in the index range.
8039_0000	OpcUa_BadDataEncodingUnsupported	The OPC UA server does not support the required data decoding for this node.
803A_0000	OpcUa_BadNotReadable	No rights for reading or subscribing (Subscribing) this node.

Error code (hexadecimal values)	Name of the error	Explanation
803C_0000	OpcUa_BadOutOfRange	The index value specified in the NodeAddInfos parameter is outside the permitted range.
803D_0000	OpcUa_BadNotSupported	The OPC UA server does not support one of the requested functions. Some OPC UA servers do not allow access to the index ranges of an array.
8060_0000	OpcUa_BadBrowseNameInvalid	The NamespaceIndex of the QualifiedName does not exist.
806F_0000	OpcUa_BadNoMatch	The specified path does not lead to a node.
For more error codes, see AUTOHOTSPOT.		

This is how you use this instruction

Based on an example, this section shows you how to use the instruction "OPC_UA_TranslatePathList" in a program.

Example

Let us assume that RFID readers scan the outgoing goods at the gates of a plant.

In this example, the first reader is called "RfidReader_Door_1".

The RFID readers are OPC UA servers that make their data and methods available to OPC UA clients.

Specification AutoID

The RFID readers come with a standardized interface according to the Companion Specification "AutoID".

The names of the properties and methods of the RFID readers are specified by the standardization.

The figure below shows a section of the address space of the reader "FfidReaderDoor_1" in UAExpert, a tool from Unified Automation.

The node "RfidReader_Door_1" is located under ""DeviceSet".

Below it, you can see the released properties and methods of the reader according to the AutoId:

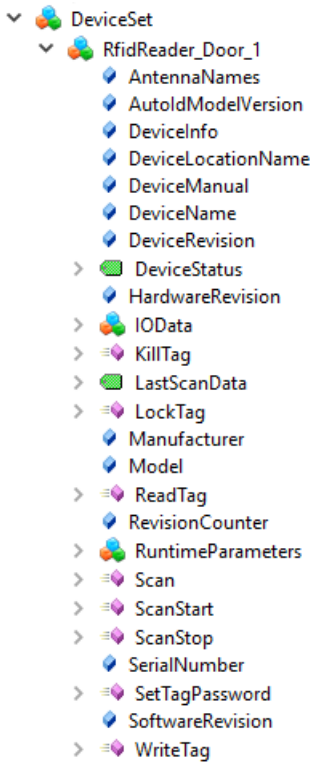

The properties and methods of the reader are known because the reader adhere to the AutoID specification.

Not known are the following elements:

- The name "RfidReader_Door_1". It can be freely selected or is specified in the respective plant.
- The NodeID of the node "RfidReader_Door_1" in the address space of the reader
- The NodeIDs of the properties and methods of the reader, which means all nodes below "RfidReader_Door_1", see figure above.

The following procedures are available to determine the NodeIDs:

- You are using a tool such as UAExpert and read the NodeID of "RfidReader_Door_1" as well as the NodeIDs of all lower-level properties and methods of this reader that you want to use in your user program.
  The next step in your program is the call of "OPC_UA_NodeGetHandleList".
- You are working with the instruction "OPC_UA_TranslatePathList".
  Starting with the node "DeviceSet", whose NodeID you know, you define a path (BrowsePath) in your user program.
  Then you use the "OPC_UA_TranslatePathList" instruction to determine the NodeIDs of all lower-level properties and methods of the reader that you use in your user program.
  The next step is the call of "OPC_UA_NodeGetHandleList".
  This procedure is shown in the following section.

Requirement

The following requirements must be met for the instruction "OPC_UA_TranslatePathList":

- A handle for connection to an OPC UA server is available.
  You get the connection handle from the instruction OPC_UA_Connect.

Calling the instruction (initial call)

This section describes the function block "Read_from_RfidReader_Door_1". The function block shows you how to use the instruction "OPC_UA_TranslatePathList".

You can find the complete example in the section "AUTOHOTSPOT".

Declaration of tags

Declare an instance of the instruction "OPC_UA_TranslatePathList" and the tags with which you are supplying the instruction parameters.

The program example uses the following declaration:

OPC_UA_TranslatePathList_Instance	OPC_UA_TranslatePathList
BrowsePaths	Array[0..1] of OPC_UA_BrowsePath
TagetNodeIds	Array[0..1] of OPC_UA_NodeId
TargetStatusList	Array[0..1] of DWord

User program

The following excerpt from the user program defines the browse path for the PLC tag "DeviceInfo". The definition for "DeviceStatus" is not shown.

First, the start node (StartingNode) is defined. This leads to the browse path of the sought after node. The path to the node (RelativePath) you are looking for is defined below.

SCL

```
IF #FirstCall = TRUE THEN
                //set BrowsePath[0] for DeviceInfo
                //set BrowsePath.StartingNode
                #BrowsePaths[0].StartingNode.Identifier := WString#'85';
                #BrowsePaths[0].StartingNode.IdentifierType := 0;
                #BrowsePaths[0].StartingNode.NamespaceIndex := 0;

                //set BrowsePath.RelativePath
                #BrowsePaths[0].RelativePath.NoOfElements := 4;
                #BrowsePaths[0].RelativePath.Elements[1].ReferenceTypeId.Identifier :=
WString#'33';
               #BrowsePaths[0].RelativePath.Elements[1].ReferenceTypeId.IdentifierType :=
0;
               #BrowsePaths[0].RelativePath.Elements[1].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[0].RelativePath.Elements[1].TargetName.Name :=
WString#'productionline';
                #BrowsePaths[0].RelativePath.Elements[1].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

                #BrowsePaths[0].RelativePath.Elements[2].ReferenceTypeId.Identifier :=
WString#'33';
               #BrowsePaths[0].RelativePath.Elements[2].ReferenceTypeId.IdentifierType :=
0;
               #BrowsePaths[0].RelativePath.Elements[2].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[0].RelativePath.Elements[2].TargetName.Name :=
WString#'DataBlocksGlobal';
                #BrowsePaths[0].RelativePath.Elements[2].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

                #BrowsePaths[0].RelativePath.Elements[3].ReferenceTypeId.Identifier :=
WString#'33';
               #BrowsePaths[0].RelativePath.Elements[3].ReferenceTypeId.IdentifierType :=
0;
               #BrowsePaths[0].RelativePath.Elements[3].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[0].RelativePath.Elements[3].TargetName.Name :=
WString#'RfidReader_Door_1';
                #BrowsePaths[0].RelativePath.Elements[3].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

                #BrowsePaths[0].RelativePath.Elements[4].ReferenceTypeId.Identifier :=
WString#'33';
               #BrowsePaths[0].RelativePath.Elements[4].ReferenceTypeId.IdentifierType :=
0;
               #BrowsePaths[0].RelativePath.Elements[4].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[0].RelativePath.Elements[4].TargetName.Name :=
WString#'DeviceInfo';
                #BrowsePaths[0].RelativePath.Elements[4].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];
```

Program example part 2

The following section of the program example configures the connection to an OPC UA server.

The ServerEndpoint is set for the "#ServerEndpointUrl" tag. In the example, the server endpoint is "opc.tcp://192.168.1.1:4840", because the "opc.tcp" protocol is used for OPC UA and the server has the IP address "192.168.1.1".

The security mode "2" (SecurityMsgMode) is set so that messages can be sent signed. The value "Basic128Rsa15" is set as the procedure for signing (SecurityPolicy). The number of the client certificate (CerificateID) is also specified. This is the number 6 in our example.

SCL

```
CASE #State OF
        1:  // case 1, connect to server


            IF #FirstCall = TRUE THEN
                #FirstCall := FALSE;

                //set ServerEndPointUrl of the server we want to connect to
                #SeverEndpointUrl := WString#'opc.tcp://192.168.1.1:4840';

                //set Security Message Mode
                // 1 = None, 2 = Sign, 3 = Sign & Encrypt
                #SessionConnectInfo.SecurityMsgMode := 1;

                //set Security Policy
                // 1 = None, 2 = Basic128Rsa15,, 3 = Basic256, 4 = Basic256Sha256
                #SessionConnectInfo.SecurityPolicy := 2;

                //set the number of the client certificate you are using
                //in our case, this is 6
                //#SessionConnectInfo.CertificateID := 6;
            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_Connect_Instance(REQ:=#Req,
                                     ServerEndpointUrl:=#SeverEndpointUrl,
                                     SessionConnectInfo:=#SessionConnectInfo,
                                     Timeout:=T#8S,
                                     Done=>#Done,
                                     Busy=>#Busy,
                                     Error=>#Error,
                                     Status=>#Status,
                                     ConnectionHdl=>#ConnectionHdl);
```

Program example part 3

The following excerpt from the example program "Read_From_RfidReader_Door_1" queries the OPC UA server for node IDs of nodes with the BrowseName "DeviceInfo" and "DeviceStatus".

You can find the complete example in the section "AUTOHOTSPOT".

SCL

```
IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_TranslatePathList_Instance(BrowsePaths := #BrowsePaths,
                                               TargetNodeIDs := #TagetNodeIds,
                                               TargetStatusList := #TargetStatusList,
                                               ConnectionHdl := #ConnectionHdl,
                                               NamespaceIndexCount := 0,
                                               BrowsePathsCount := 2,
                                               Timeout := T#6S,
                                               REQ := #Req,
                                               Busy => #Busy,
                                               Error => #Error,
                                               Done => #Done,
                                               Status => #Status);
            IF #Done = TRUE THEN
                IF #TargetStatusList[0] = 0 AND #TargetStatusList[1] = 0 THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at TranslatePathList';
                END_IF;
                #FirstCall := TRUE;
            END_IF;
            IF #Error = TRUE THEN
                //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #State := 100;
                #Mem_Status := #Status;
                #OPC_UA_TranslatePathList_Instance(BrowsePaths := #BrowsePaths,
                                                   TargetNodeIDs := #TagetNodeIds,
                                                   REQ := FALSE,
                                                   TargetStatusList := #TargetStatusList,
                                                   ConnectionHdl := #ConnectionHdl);
                #FirstCall := TRUE;
            END_IF;
```

The example program calls the "OPC_UA_TranslatePathList" instruction for the following reasons:

- To request the NodeID for the nodes with BrowseName "DeviceInfo" and "DeviceStatus".
- To set the REQ parameter to FALSE, if an error has occurred.

Asynchronous execution

The instruction "OPC_UA_TranslatePathList" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the program example part 3

In the figure above, a section of the function block "Read_From_RfidReader_Door_1" is shown.

If the "OPC_UA_TranslatePathList" instruction is not yet executed, #Busy is FALSE, so that the "#Req" tag is set to TRUE. This starts the instruction.

In the next cycle, #Req is FALSE.

If the output parameter "Error" is TRUE, an error occurred during execution of the instruction. This sets the "#State" tag to the value 100. This case is reserved for troubleshooting.

If the output parameter "Done" is TRUE, the instruction was executed successfully.

Please note: It is possible that the instruction was successfully executed ("Error" parameter not set, "Done" parameter set) but no NodeID could be returned for a PLC tag.

Therefore, there is a check in the example to determine whether the program has obtained a node ID for each PLC tag:

- If a NodeID returns for the first tag, then the first element of the array to which the "TargetStatusList" parameter is pointing contains the value 0.
- If a NodeID returns for the second tag, then the second element of the array to which the "TargeteStatusList" parameter is pointing contains the value 0.

If both conditions are met, the output parameter "TargetNodeHdls" points to a valid list with the NodeIDs. This increases the value of the "#State" tag by one. The next program section - Case 4 - is then executed in the next cycle.

Case 4 is described in OPC_UA_NodeGetHandleList.

Data exchange

OPC_UA_ReadList: Reading tags

Validity

The following description of the "OPC_UA_ReadList" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

You use the instruction "OPC_UA_ReadList" to read the values of PLC tags.

The following figure shows the icon of the instruction in the editor (FBD).

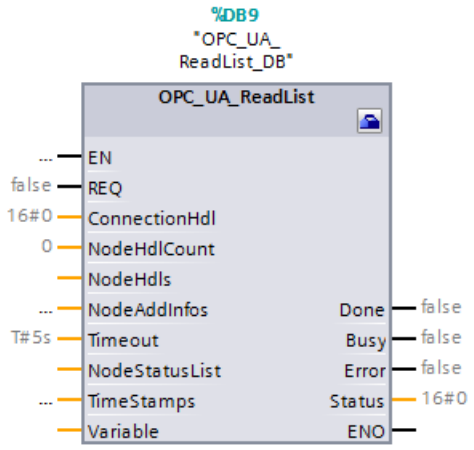

In the figure above, the parameters of the instruction are not supplied yet.

The instruction "OPC_UA_ReadList" is used to read tags from a OPC UA server, see ② in the figure below.

The instruction returns a list with values of tags.

In addition to the tag values, you can also read other tag attributes. This is defined by the AttributeID in NodeAdditionalInfo.

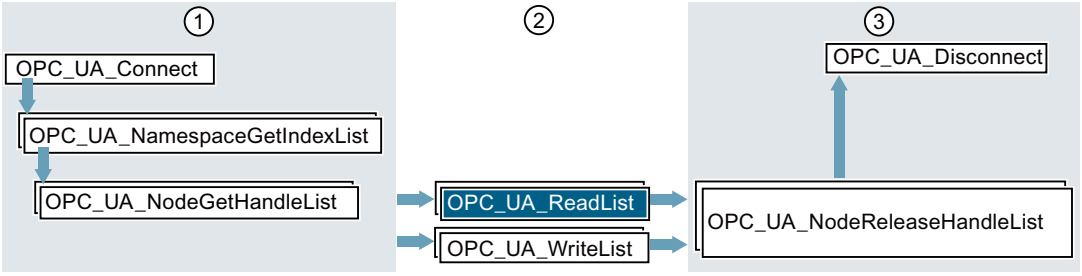

① Instructions for preparation of read and write operations

② Read and write instructions

③ Instructions for releasing resources after completed read or write operations

Parameters for "OPC_UA_ReadList"

Table 4-76 The parameters of the instruction "OPC_UA_ReadList"

Parameters	Declaration	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the instruction.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established. You get the handle from the instruction AUTOHOTSPOT.
NodeHdlCount	Input	UINT	Number of elements in the array that points to the NodeHdls parameter.

Parameters	Declaration	Data type	Meaning
NodeHdls	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the node handles of the tags whose values are to be read.
NodeAddInfos	InOut	VARIANT	Pointer to an array of the type AUTOHOTSPOT or AUTOHOTSPOT. The array defines which attribute is to be read in the node (in a tag). The first element in this array refers to the first element in the array to which the NodeHdls parameter points. The parameter is optional. If it is not set, the values of all nodes (tags) are read. When reading arrays, you can limit which elements of the array are to be read, see Reading array range information with OPC_UA_ReadList (Page 3695).
Timeout	Input	TIME	Maximum time for execution of the instruction in milliseconds. See also the explanation for this parameter in AUTOHOTSPOT.
NodeStatusList	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the error codes for the individual tags (see "Error numbers for NodeStatusList" below). Whether or not the value could be read is indicated for each tag. It is possible that the instruction was successfully executed (Error parameter not set) but no value could be read for a specific NodeID (tag).
TimeStamps	InOut	VARIANT	Pointer to an array of the type LDT. The first element in this array refers to the first element in the array to which the NodeHdls parameter points. The parameter is optional. If it is not set, the OPC UA server does not return a time stamp.
Variable	InOut	VARIANT	Pointer to a tag that saves the values to be read. You must create a PLC data type (UDT) or a structure (STRUCT) for this tag.
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors

Parameters	Declaration	Data type	Meaning
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error numbers for Status" below

Error numbers for Status

The "Status" parameter provides information about errors that occurred during execution of the instruction.

The following table shows a summary of error codes for this instruction:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Function block executed successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.
8009_0000	OpcUa_BadUnknownResponse	Server did not respond with the expected number of results

Error code (hexadecimal values)	Name of the error	Explanation
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.
800F_0000	OpcUa_BadNothingToDo	Nothing to do: The OPC UA server received an empty list with no instructions from the OPC UA client.
8010_0000	OpcUa_BadTooManyOperations	The request could not be executed because it contained too many instructions.
8074_0000	OpcUa_BadTypeMismatch	You are using a data type that does not correspond to the data type in the server.
B080_0100	Simatic_BadType_VariantInput1	The data type of the tag to which the first variant parameter of the instruction is pointing is incorrect.
B080_0200	Simatic_BadType_VariantInput2	The data type of the tag to which the second variant parameter of the instruction is pointing is incorrect.
B080_0300	Simatic_BadType_VariantInput3	The data type of the tag to which the third variant parameter of the instruction is pointing is incorrect.
B080_0400	Simatic_BadType_VariantInput4	The data type of the tag to which the fourth variant parameter of the instruction is pointing is incorrect.
B080_0500	Simatic_BadType_VariantInput5	The data type of the tag to which the fifth variant parameter of the instruction is pointing is incorrect.
B080_1100	Simatic_ArrayElements_TooMany	NodeHdlCount > MAX_ELEMENTS_NODELIST NamespaceIndexCount > MAX_ELEMENTS_NAMESPACES
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled.
B080_C500	Simatic_NothingToDo	Nothing to do: The instruction is using a list that contains no elements.

Error code (hexadecimal values)	Name of the error	Explanation
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Possible remedy: • Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

Error numbers for "NodeStatusList"

The "NodeStatusList" parameter contains an error code for each individual node handle (tag).

The following table explains the error codes:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Value successfully read.
8034_0000	OpcUa_BadNodeIdUnknown	The node handle transferred is not known.
8035_0000	OpcUa_BadAttributeInvalid	The attribute required is not supported for the specified node.
8037_0000	OpcUa_BadIndexRangeNoD	There is no data in the index range.
8039_0000	OpcUa_BadDataEncodingUnsupported	The OPC UA server does not support the required data decoding for this node.
8074_0000	OpcUa_BadTypeMismatch	Double / Float value in ReadResponse is NaN UDT Member tag has incorrect type
803A_0000	OpcUa_BadNotReadable	No rights for reading or subscribing (Subscribing) this node.
803C_0000	OpcUa_BadOutOfRange	The index value specified in the NodeAddInfos parameter is outside the permitted range.
803D_0000	OpcUa_BadNotSupported	The OPC UA server does not support one of the requested functions. Some OPC UA servers do not allow access to the index ranges of an array.
80AB_0000	OpcUa_BadInvalidArgument	StartIndex > EndIndex
For more error codes, see AUTOHOTSPOT.		

This is how you use this instruction

Based on a program example, this section shows you how to use the instruction "OPC_UA_ReadList" in a user program that reads the values of PLC tags.

Requirements

The requirements for the following description are:

- You have created a client interface, see "AUTOHOTSPOT".
- You have created and configured a connection to an OPC UA server, see "Creating and configuring connections".

In addition, the following requirements must be met for the instruction "OPC_UA_ReadList":

- A handle for connection to an OPC UA server is available.
  You get the connection handle from the instruction AUTOHOTSPOT.
- Indexes of the namespaces that contain the tags that you want to read with the client.
  You get the indexes from the instruction AUTOHOTSPOT.
- A list of the handles for the individual tags whose values you want to read from the OPC UA server.
  You get this handle list from the instruction AUTOHOTSPOT.

Function of the instruction

You use the instruction "OPC_UA_ReadList", for example, to read the values of tags from an OPC UA server (see above).

What information is required?

The instruction requires the following information:

- The handles for the tags with the values you want to read (in the "NodeHdls" parameter).

What information does the instruction return?

The instruction returns the following information:

- The current values of the tags (in the "Variable" parameter).
  The order of the values corresponds to the order of the handles in the "NodeHdls" parameter.
  Refer to the following cases for this purpose:
  - If you use a client interface with a read list, STEP 7 automatically creates a system data type with the tags that need to be read.
  - If you are not using a client interface, you must create a PLC data type (UDT) or a structure (STRUCT) for the "Variable" parameter.
    You define the components of this UDT according to the tags that have to be read.
    Use the SIMATIC data type compatible with the respective OPC UA data type, see AUTOHOTSPOT.
- A list of error messages (in the "NodeStatusList" parameter).
  Each error message in this list relates to the corresponding handle in the "NodeHdls" parameter, NodeStatusList[0] refers to NodeHdls[0].
  For each individual tag (each individual "NodeHdl"), you need to check whether the OPC UA server was able to return a valid value.

How to use a configured connection

1. In the "Project tree" area, select the CPU that acts as the client.
2. Select the function block that is to contain the client instruction in the "Program blocks" folder.
   In the example, the function block is called "ReadFromProductionline".
   Selected language: SCL.
3. Use drag-and-drop to move the instruction "OPC_UA_ReadList" from the folder "Instructions > Communication > OPC UA > OPC UA Client" to the editor.
4. Select the call as multi-instance.
   STEP 7 creates an instance of this instruction and calls it "OPC_UA_ReadList_Instance".
5. Click on the icon for "Start configuration" at the "OPC_UA_ReadList_Instance" instruction.
   STEP 7 opens the "Configuration" tab in the Inspector window.
6. Under "Select client interface for OPC UA interface" you select the client interface that you want to use for the instruction.
   In the example, the client interface is called "Productionline".
7. Click "File access" and select a read list.
   In the example, the read list is called "ReadListProduct".
   STEP 7 now automatically supplies the parameters of the instruction with the values you have configured for the client interface.
8. Click on "Block parameters" and assign tags manually to the remaining parameters REQ, Busy, Done, Error, Status.
   STEP 7 adds the selected tag to the function call.

Calling the instruction (initial call)

You can find the complete program in the section "AUTOHOTSPOT".

The program example uses the read list "ReadListProduct", which contains the following PLC tags:

- NewProduct
- ProductNumber

The program reads the values of these tags from the server.

The section "AUTOHOTSPOT" shows you how to create a client interface and how to add a read list with PLC tags to it.

The example program is divided into multiple sections (cases) by a CASE instruction.

In the fourth program section, the tag values are read:

SCL

```
4:  // case 4, read from nodes

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_ReadList_Instance(
                                    NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                    NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                    Variable :=
"Productionline_Data"."ReadListProduct".Variable,
                                    ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                    Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                    NodeHdlCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount,
                                    TimeStamps :=
"Productionline_Data"."ReadListProduct".TimeStamps,
                                    REQ := #Req,
                                    Done => #Done,
                                    Busy => #Busy,
                                    Error => #Error,
                                    Status => #Status);
```

Asynchronous execution

The instruction "OPC_UA_ReadList" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction (initial call)

The figure above shows case 4.

This part of the program calls the "OPC_UA_ReadList" instruction to read the values of PLC tags.

If the instruction is not yet executed, #Busy is FALSE, so that the "#Req" tag is set to the value TRUE. This starts the instruction. In the next cycle, #Req is FALSE.

Calling the instruction (troubleshooting)

The figure below shows evaluation of the "Done" and "Error" parameters.

SCL

```
            IF #Done = TRUE THEN
                FOR #i := 0 TO
UINT_TO_INT("Productionline_Configuration".ReadLists."ReadListProduct".NodeCount) - 1 DO
                   IF NOT ("Productionline_Data"."ReadListProduct".NodeStatusList[#i] = 0)
THEN
                        #Output_Error_Message := CONCAT_WSTRING(IN1 := WSTRING#'Error at
Readlist "ReadListProduct", Index: ',
                                                                    IN2 := INT_TO_WSTRING(#i));
                    END_IF;
                END_FOR;
                #State := #State + 1;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_ReadList_Instance(
                                          NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                          NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                          Variable :=
"Productionline_Data"."ReadListProduct".Variable,
                                          REQ := FALSE,
                                          ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                          Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                          NodeHdlCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount,
                                          TimeStamps :=
"Productionline_Data"."ReadListProduct".TimeStamps);
            END_IF;
```

Explanation of the call of the instruction (troubleshooting)

If an error occurs, the "Error" output parameter is set to the value TRUE. This sets the value of the "#State" tag to 100. This case is reserved for troubleshooting. The example program also calls the instruction "OPC_UA_ReadList" to set the REQ parameter to FALSE.

If the TRUE value is in the "Done" output parameter, the instruction is successfully executed.

Please note: It is possible that the instruction was successfully executed ("Error" parameter not set, "Done" parameter set) but no PLC tag could be read.

Therefore, there is a check in the example for each PLC tag to determine whether the program has obtained a valid value:

- If the server is sending a valid value for the first PLC tag, the first element of the array to which the "NodeStatusList" parameter is pointing contains the value 0.
  In the example, the status code is listed in "Productionline_Data"."ReadListProduct".NodeStatusList[0].
- If the server is sending a valid value for the second PLC tag, the second element of the array to which the "NodeStatusList" parameter is pointing contains the value 0.
  In the example, the status code is listed in "Productionline_Data"."ReadListProduct".NodeStatusList[1].

The program outputs an error message when it recognizes an invalid value.

Even if one or two invalid values are detected, the value of the "#State" tag is increased by one. This means the next program section - case 5 - is executed in the next cycle.

Case 5 is described in AUTOHOTSPOT.

See also

OPC_UA_NodeReleaseHandleList: Release handles for read and write access (Page 3715)

OPC_UA_TranslatePathList: Read node parameter (Page 3674)

OPC UA instructions for client programs (Page 3640)

Reading array range information with OPC_UA_ReadList

Introduction

This section is based on "OPC_UA_ReadList"; see OPC_UA_ReadList (Page 3685).

The section describes how to read parts of an array with the "OPC_UA_ReadList" instruction.

Reading parts of an array from an OPC UA server basically works as follows:

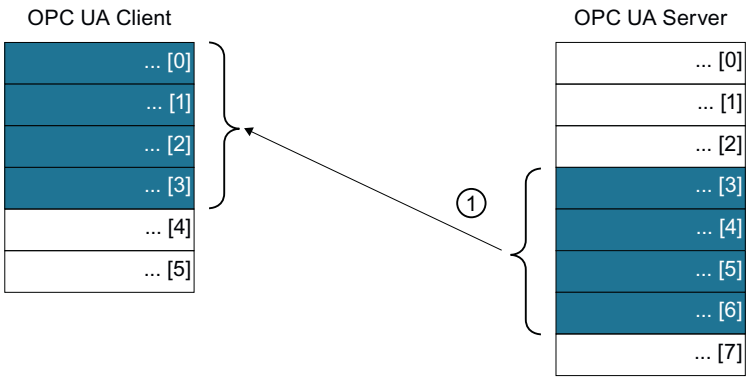

① Reading parts of an array from an OPC UA server

Requirements

The following description assumes that:

- You have configured a CPU as **OPC UA server**.
  The server provides an array that can be read by OPC UA clients.
  In the example, the CPU is called "Productionline", which provides the data block "Data_for_OPC_UA_Clients" for OPC UA clients:

Static	
NewProduct	Bool
ProductNumber	Int
Temperature	Array[0..9] of Real
Temperature[0]	Real
Temperature[1]	Real
Temperature[2]	Real
Temperature[3]	Real
Temperature[4]	Real
Temperature[5]	Real
Temperature[6]	Real
Temperature[7]	Real
Temperature[8]	Real
Temperature[9]	Real

- You have configured a CPU as **OPC UA client**.
  You have created a client interface in the client.
  The client interface contains a read list with an array of which you want to read a part.
  In the example, the client interface is called "Productionline" with the read list "ReadListProduct":

NewProduct	BOOL
ProductNumber	INT
Temperature	ARRAY[0..9] of REAL

The values in "Temperature[5]" to "Temperature[9]" are read from the "Temperature" array of the server.

Providing additional information

"OPC_UA_ReadList" shows how to read out entire scalars (tags with a value) or arrays (tags with multiple values) using the "OPC_UA_ReadList" instruction.

The following section provides additional details to "OPC_UA_ReadList" that are required for using "OPC_UA_ReadList" to read part of an array.

OPC_UA_NodeAdditionalInfo

First create a local tag of the type "OPC_UA_NodeAdditionalInfo (Page 3764)".

This tag is used to tell the "OPC_UA_ReadList" instruction which part of the array is to be read.

Proceed as follows:

1. Create an array tag of the type "OPC_UA_NodeAdditionalInfo".
   In the example, the tag is created in the declaration section of the function block, which calls the "OPC_UA_ReadList" instruction.
   The new local tag is called "NodeAdditionalInformation":

NodeAdditonalInformation	Array[0..2] of OPC_UA_NodeAdditionalInfo
NodeAdditonalInformation[0]	OPC_UA_NodeAdditionalInfo
AttributeID	UDInt
StartIndex	UDInt
EndIndex	UDInt
NodeAdditonalInformation[1]	OPC_UA_NodeAdditionalInfo
AttributeID	UDInt
StartIndex	UDInt
EndIndex	UDInt
NodeAdditonalInformation[2]	OPC_UA_NodeAdditionalInfo
AttributeID	UDInt
StartIndex	UDInt
EndIndex	UDInt

   The new local tag "NodeAdditionalInformation" contains three components of the type OPC_UA_AdditionalInfo, because the read list "ReadListProduct" contains three nodes.
   NodeAdditionalInformation[0] provides additional information on the first tag to be read (in the example, "NewProduct")
   NodeAdditionalInformation[1] provides additional information on the second tag to be read (in the example, "ProductNumber")
   NodeAdditionalInformation[2] provides additional information on the third tag to be read (in the example, "Temperature")

2. Assign values to the new tag (in the "Default values" column of the declaration section).
   In the example, the third tag (Temperature is partially read.
   NodeAdditionalInformation[2].AttributeId is therefore assigned 13 (i.e. the values of the array are read).
   NodeAdditionalInformation[2].StartIndex is assigned a value of 5, as the reading starts at (and includes) index 5.
   NodeAdditionalInformation[2].EndIndex is assigned a value of 9, as the reading starts at (and includes) index 9.
   NodeAdditionalInformation[0] NodeAdditionalInformation[1] are set to default values.
   The constant "#ignore" contains the value 4_294_967_295. The server ignores this value for StartIndex and EndIndex and reads scalars and arrays completely.

NodeAdditonalInformation	Array[0..2] of OPC_UA_NodeAdditionalInfo	
NodeAdditonalInformation[0]	OPC_UA_NodeAdditionalInfo	
AttributeID	UDInt	13
StartIndex	UDInt	#ignore
EndIndex	UDInt	#ignore
NodeAdditonalInformation[1]	OPC_UA_NodeAdditionalInfo	
AttributeID	UDInt	13
StartIndex	UDInt	#ignore
EndIndex	UDInt	#ignore
NodeAdditonalInformation[2]	OPC_UA_NodeAdditionalInfo	
AttributeID	UDInt	13
StartIndex	UDInt	5
EndIndex	UDInt	9

Alternative:
You can also set the values for NodeAdditionalInformation in the user program:

```
                // set additonal information to Array-Index 0
                // we set default values here
                #NodeAdditionalInformation[0].AttributeID := 13;
                #NodeAdditionalInformation[0].StartIndex :=
4_294_967_295;
                #NodeAdditionalInformation[0].EndIndex :=
4_294_967_295;

                // set additonal information to Array-Index 1
                // we set default values here
                #NodeAdditonalInformation[1].AttributeID := 13;
                #NodeAdditonalInformation[1].StartIndex :=
4_294_967_295;
                #NodeAdditonalInformation[1].EndIndex :=
4_294_967_295;

                // set additonal information to Array-Index 2
                // because we want to read only five values from
an array in the OPC UA server
                #NodeAdditonalInformation[2].AttributeID := 13;
                #NodeAdditonalInformation[2].StartIndex := 5;
                #NodeAdditonalInformation[2].EndIndex := 9;
```

Note:
The array NodeAdditionalInformation of the type "OPC_UA_NodeAdditionalInfo" must contain as many components as the read list (3 in the example).

3. Assign NodeAdditionalInformation to the parameter "NodeAddInfos" of the instruction "OPC_UA_ReadList".
   The following figure shows the call of OPC_UA_ReadList with the new supply of the "NodeAddInfos" parameter:

```
    #OPC_UA_ReadList_Instance(
                              NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdl
s,
                              NodeStatusList :=
"myProductionline_Data"."ReadListProduct".NodeStatusList,
                              Variable :=
"myProductionline_Data"."ReadListProduct".Variable,
                              ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                              Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeo
ut,
                              NodeHdlCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCou
nt,
                              NodeAddInfos :=
#NodeAddtionalInformation,
                              TimeStamps :=
"myProductionline_Data".ReadListProduct.TimeStamps,
                              REQ := #Req,
```

```
Done => #Done,
Busy => #Busy,
Error => #Error,
Status => #Status);
```

Result in the client

The "OP_UA_ReadList" instruction now reads only part of the "Temperature" array from the OPC UA server.

The figure below shows the array "Temperature". The client assigns the read values as of Index 0:

Temperature	Array[0..9] of Real		
Temperature[0]	Real	0.0	206.6
Temperature[1]	Real	0.0	207.7
Temperature[2]	Real	0.0	208.8
Temperature[3]	Real	0.0	209.9
Temperature[4]	Real	0.0	210.0
Temperature[5]	Real	0.0	0.0
Temperature[6]	Real	0.0	0.0
Temperature[7]	Real	0.0	0.0
Temperature[8]	Real	0.0	0.0
Temperature[9]	Real	0.0	0.0

Note

To read a section of a multi-dimensional array (up to six dimensions), use the "OPC_UA_NodeAdditionalInfoExt" data type.

OPC_UA_WriteList:Write tags

Validity

The following description of the "OPC_UA_WriteList" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

You use the instruction "OPC_UA_WriteList" to write new values to PLC tags.

The following figure shows the icon of the instruction in the editor (FBD).

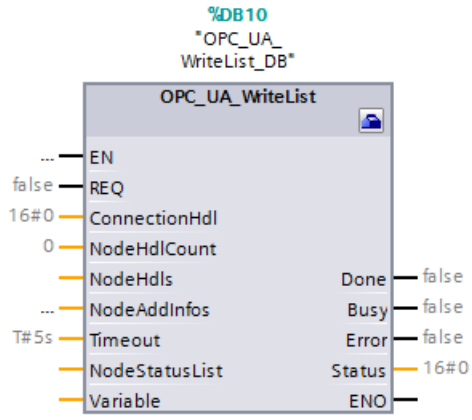

In the figure above, the parameters of the instruction are not supplied yet.

The instruction "OPC_UA_WriteList" is used to assign new values to PLC tags, see ② in the figure below.

The instruction returns a list with status information for each PLC tag.

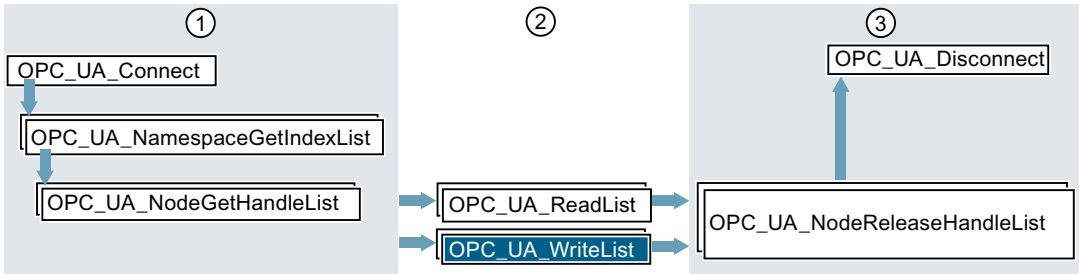

① Instructions for preparation of read and write operations

② Read and write instructions

③ Instructions for releasing resources after completed read or write operations

Parameters for OPC_UA_WriteList

Table 4-77 The parameters of the instruction OPC_UA_WriteList

Parameters	Declaration in area	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the instruction.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established. You get the handle from the instruction AUTOHOTSPOT.
NodeHdlCount	Input	UINT	Number of elements in the array that points to the NodeHdls parameter.
NodeHdls	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the node handles of the tags whose values are to be written.

Parameters	Declaration in area	Data type	Meaning
NodeAddInfos	InOut	VARIANT	Pointer to an array of the type AUTOHOTSPOT or AUTOHOTSPOT. The array defines which attribute is to be set in a node (in a tag). The first element in this array refers to the first element in the array to which the NodeHdls parameter points. The parameter is optional. If it is not set, the value is set in all nodes (tags). When writing arrays, you can limit which elements of the array are to be written, see Writing an array section with OPC_UA_WriteList (Page 3709).
Timeout	Input	TIME	Maximum time for execution of the instruction in milliseconds. See also the explanation for this parameter in AUTOHOTSPOT.
NodeStatusList	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the error codes for the individual tags (see "Error numbers for NodeStatusList" below). Whether or not the value could be set is indicated for each tag. It is possible that the instruction was successfully executed (Error parameter not set) but a specific tag could not be assigned a value. NodeStatusList[0] relates to NodeHdls[0], etc.
Variable	InOut	VARIANT	Pointer to a tag that contains the values to be written. You must create a PLC data type (UDT) for this tag.
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error numbers for Status" below

Error numbers for Status

The "Status" parameter provides information about errors that occurred during execution of the instruction.

The following table shows a summary of error codes for this instruction:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Function block executed successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.
8009_0000	OpcUa_BadUnknownResponse	The server sends a response, that cannot be recognized.
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.
800F_0000	OpcUa_BadNothingToDo	Nothing to do: The OPC UA server received an empty list with no instructions from the OPC UA client.
8010_0000	OpcUa_BadTooManyOperations	The request could not be executed because it contained too many instructions.
B080_0100	Simatic_BadType_VariantInput1	The data type of the tag to which the first variant parameter of the instruction is pointing is incorrect.
B080_0200	Simatic_BadType_VariantInput2	The data type of the tag to which the second variant parameter of the instruction is pointing is incorrect.
B080_0300	Simatic_BadType_VariantInput3	The data type of the tag to which the third variant parameter of the instruction is pointing is incorrect.
B080_0400	Simatic_BadType_VariantInput4	The data type of the tag to which the fourth variant parameter of the instruction is pointing is incorrect.

Error code (hexadecimal values)	Name of the error	Explanation
B080_1100	Simatic_ArrayElements_TooMany	General error code. Occurs when an array has too many elements.
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled.
B080_C500	Simatic_NothingToDo	Nothing to do: The instruction is using a list that contains no elements.
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Possible remedy: • Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

Error numbers for "NodeStatusList"

The "NodeStatusList" parameter contains an error code for each individual node handle (tag).

The following table explains the error codes:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Value successfully read.
8034_0000	OpcUa_BadNodeUnknown	The node handle transferred is not known.
8035_0000	OpcUa_BadAttributeInvalid	The attribute required is not supported for the specified node.
8037_0000	OpcUa_BadIndexRangeNoD	There is no data in the index range.
8039_0000	OpcUa_BadDataEncodingUnsupported	The OPC UA server does not support the required data decoding for this node.
803A_0000	OpcUa_BadNotReadable	No rights for reading or subscribing (Subscribing) this node.
803C_0000	OpcUa_BadOutOfRange	The index value specified in the NodeAddInfos parameter is outside the permitted range.
803D_0000	OpcUa_BadNotSupported	The OPC UA server does not support one of the requested functions. Some OPC UA servers do not allow access to the index ranges of an array.
80AB_0000	OpcUa_BadInvalidArgument	One or more arguments are invalid.
For more error codes, see AUTOHOTSPOT.		

This is how you use this instruction

Based on a program example, this section shows you how to use the instruction "OPC_UA_WriteList" in a user program that sets new values in PLC tags.

Requirements

The following description assumes that:

- You have created a client interface, see "AUTOHOTSPOT".
- You have created and configured a connection to an OPC UA server, see "AUTOHOTSPOT".

In addition, the following requirements must be met for the instruction "OPC_UA_WriteList":

- A handle for connection to an OPC UA server is available.
  You get the connection handle from the instruction AUTOHOTSPOT.
- Indexes of the namespaces that contain the tags that you want to write with the client.
  You get the indexes from the instruction AUTOHOTSPOT.
- A list of the handles for the individual tags whose values you want to set in the OPC UA server.
  You get this handle list from the instruction AUTOHOTSPOT.

In the program example below, the instruction is called once to send an enable signal to the OPC UA server.

Function of the block

You use the instruction "OPC_UA_WriteList" to write the values of tags in an OPC UA server.

What information is required?

The function requires the following information:

- The handles for the tags whose values you want to write ("NodeHdls" parameter).
- The values to be transferred for the individual tags ("Variable" parameter).
  Refer to the following cases for this purpose:
  - If you use a client interface with a write list, STEP 7 automatically creates a system data type with the tags that need to be transferred.
  - If you are not using a client interface, you must create a PLC data type (UDT) or a structure (STRUCT) for the "Variable" parameter.
    You define the components of this UDT according to the tags that have to be transferred.
    Use the SIMATIC data type compatible with the respective OPC UA data type, see AUTOHOTSPOT.

What information does the instruction return?

The instruction returns the following information:

- A list of error messages ("NodeStatusList" parameter).
  Each error message in this list relates to the corresponding handle in the "NodeHdls" parameter.
  For each individual tag ("NodeHdl"), you need to check whether the OPC UA server was able to accept the new value.

How to use a configured connection

1. In the "Project tree" area, select the CPU that acts as the client.
2. Select the function block that is to execute the client instruction in the "Program blocks" folder.
   In the example, the function block is called "WriteToProductionline".
   Selected language: SCL.
3. Use drag-and-drop to move the instruction "OPC_UA_WriteList" from the folder "Instructions > Communication > OPC UA > OPC UA Client" to the editor.
4. Select the call as multi-instance.
   STEP 7 creates an instance of this instruction and calls it "OPC_UA_WriteList_Instance".
5. Click "OPC_UA_WriteList_Instance".
   STEP 7 opens the "Configuration".
6. Under "Select client interface for OPC UA interface" you select the client interface that you want to use for the instruction.
   In the example, the client interface is called "Productionline", which we created for our AUTOHOTSPOT, see "AUTOHOTSPOT".
7. Click "File access" and select a read list.
   In the example, the read list is called "WriteListStatus".
   STEP 7 now automatically supplies the parameters of the instruction with the values you have configured for the client interface.
8. Click on "Block parameters" and assign tags manually to the remaining parameters REQ, Busy, Done, Error, Status.
   STEP 7 adds the selected tag to the function call.

Calling the instruction (initial call)

You can find the complete program in the section "AUTOHOTSPOT".

The program example uses the write list "WriteListStatus" which contains the following PLC tag:

- ProductionEnabled

The program writes the new value of this tag to the server.

The section "AUTOHOTSPOT" shows you how to create a client interface and how to add a write list with PLC tags to it.

The example program is divided into multiple sections (cases) by a CASE instruction.

In the fourth program section, the tag values are written:

SCL

```
4:  // case 4;  write value to PLC variable "ProductionEnabled"

            IF #SetProductionEnabled = TRUE THEN
                #SetProductionEnabled := FALSE;
                //set new value to true
               "Productionline_Data"."WriteListStatus".Variable.ProductionEnabled := TRUE;
            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_WriteList_Instance(
                                        NodeHdls :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeHdls,
                                        NodeStatusList :=
"Productionline_Data"."WriteListStatus".NodeStatusList,
                                        Variable :=
"Productionline_Data"."WriteListStatus".Variable,
                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        NodeHdlCount :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeCount,
                                        REQ := #Req,
                                        Busy => #Busy,
                                        Error => #Error,
                                        Status => #Status,
                                        Done => #Done);
```

Asynchronous execution

The instruction "OPC_UA_WriteList" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction (initial call)

The figure above shows case 4.

In the example program, this case only assigns one PLC tag a new value. In your application, you can assign new values to a large number of PLC tags.

The new value is set to TRUE, if the "#SetProductionEnabled" tag obtains the value "TRUE".

We then set "#SetProductionEnabled" to FALSE, as initialization is only to run once (in the first cycle).

The "OPC_UA_WriteList" instruction writes the new value to the PLC tag "ProductionEnabled" in the OPC UA server.

If the instruction is not yet executed, #Busy is FALSE, so that the "#Req" tag is set to the value TRUE. This starts the instruction. In the next cycle, #Req is FALSE.

Calling the instruction (troubleshooting)

The figure below shows evaluation of the "Done" and "Error" parameters.

SCL

```
             IF #Done = TRUE THEN
                FOR #i := 0 TO
UINT_TO_INT("Productionline_Configuration".WriteLists.WriteListStatus.NodeCount) - 1 DO
                   IF NOT ("Productionline_Data"."WriteListStatus".NodeStatusList[#i] = 0)
THEN
                        #Output_Error_Message := CONCAT_WSTRING(IN1 := WSTRING#'Error at
Writelist "WriteListStauts", Index: ',
                                                                    IN2 := INT_TO_WSTRING(#i));
                    END_IF;
                END_FOR;
                #State := #State + 1;
            END_IF;
            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_WriteList_Instance(
                                            NodeHdls :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeHdls,
                                            NodeStatusList :=
"Productionline_Data"."WriteListStatus".NodeStatusList,
                                            Variable :=
"Productionline_Data"."WriteListStatus".Variable,
                                            REQ := FALSE,
                                            ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                            Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                            NodeHdlCount :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeCount);
            END_IF;
```

Explanation of the call of the instruction (troubleshooting)

If an error occurs, the "Error" output parameter is set to the value TRUE. This sets the "#State" tag to the value 100. This case is reserved for troubleshooting. The example program also calls the instruction "OPC_UA_WriteList" to set the REQ parameter to FALSE.

If the output parameter "Done" is TRUE, the instruction was executed successfully.

Please note: It is possible that the instruction was successfully executed ("Error" parameter not set, "Done" parameter set) but no PLC tag could be set.

Therefore, there is a check in the example for the PLC tag to determine whether the program was able to set a valid value:

- If the server is able to write the new value to the first PLC tag, then the first element of the array to which the "NodeStatusList" parameter is pointing contains the value 0.
  In the example, only one tag is used. Your status code is available in "Productionline_Data"."WriteListStatus".NodeStatusList[0]".

The program outputs an error message when a PLC tag could not be set.

The value of the "#State" tag is increased by one. This means the next program section - case 5 - (Release Handle List) is executed in the next cycle.

See also

OPC UA instructions for client programs (Page 3640)

OPC_UA_NodeReleaseHandleList: Release handles for read and write access (Page 3715)

Writing an array section with OPC_UA_WriteList

Introduction

This section is based on "OPC_UA_WriteList"; see OPC_UA_WriteList (Page 3700).

The section describes how to assign new values to (write) a part of an array with the "OPC_UA_WriteList" instruction.

Writing to a part of an array of an OPC UA server basically works as follows:

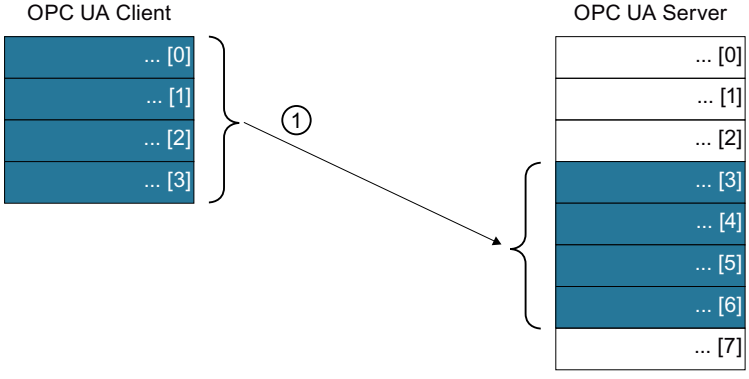

① Writing array elements to a part of an array of an OPC UA server

Requirements

The following description assumes that:

- You have configured a CPU as **OPC UA server**.
  The server provides an array that can be written by OPC UA clients.
  In the example, you use the CPU "Productionline", which provides the data block "Data_from_OPC_UA_Clients" for OPC UA clients:

Static	
ProductionEnabled	Bool
Serial_Number	Array[0..9] of LInt
Serial_Number[0]	LInt
Serial_Number[1]	LInt
Serial_Number[2]	LInt
Serial_Number[3]	LInt
Serial_Number[4]	LInt
Serial_Number[5]	LInt
Serial_Number[6]	LInt
Serial_Number[7]	LInt
Serial_Number[8]	LInt
Serial_Number[9]	LInt

- You have configured a CPU as **OPC UA client**.
  You have created a client interface in the client, see "AUTOHOTSPOT".
  The client interface contains a write list with an array.
  Only a part of this array is to be used for writing the array on the server.
  The example uses the client interface "Productionline" with the write list "WriteListStatus":

ProductionEnabled	BOOL
Serial_Number	ARRAY[0..9] of LINT

  In the "Serial_Number" array of the server, new values are to be written to "Serial_Number[5]" to "Serial_Number[9]".

Adapting write list data type and write list DB

"OPC_UA_WriteList" shows how to write new values to entire scalars (tags with a value) and arrays (tags with multiple values) using the "OPC_UA_WriteList" instruction.

The following section provides additional details to "OPC_UA_WriteList" that are required for using OPC_UA_WriteList to write new values to part of an array.

Step 1: Copying and adjusting write list data type

First, you change the data type of the tag that provides the values that you are writing to the OPC UA server.

This change is required because the data type generated by STEP 7 is too long.

The data type is too long because you only want to assign new values to part of the array (and not to the entire array).

In the example, you access the array "Serial_Number", which contains ten elements.

STEP 7 therefore generates a data type with ten elements.

Proceed as follows to shorten this data type:

1. Open the "PLC data types" folder in the "Project tree".
2. Open the "System data types" folder.
3. Copy the data type "<client interface>.<write list>".
   In this example, the data type is called "Productionline.WriteListStatus".
4. Paste the copied data type to the "PLC data types" folder.
5. Change the name of the data type inserted so that it is meaningful in your project.
   In the example, we are changing the name to "myUDTProductionline.WriteListStatus".
6. Change the data type of the array.
   In the example, we want to write the values from Serial_Number[5] to Serial_Number[9] to the OPC UA server.
   We therefore shorten the data type from "Array[0..9] of Lint" to "Array[0..4] of Lintl".
   In the example:

ProductionEnabled	Bool
Serial_Number	Array[0..4] of LInt
Serial_Number[0]	LInt
Serial_Number[1]	LInt
Serial_Number[2]	LInt
Serial_Number[3]	LInt
Serial_Number[4]	LInt

Step 2: Copy write list DB and apply new write list data type to it

In this step, you use the new data type in a new data block.

This step is required because the data block generated by STEP 7 contains the original data type for the array.

It is not possible to change the data type: Each time the data block is compiled, the original data type is restored. The changes are lost.

Proceed as follows to create a new data block:

1. In the "Project tree" area, select the CPU that acts as the client.
2. Open the "Program blocks" folder
3. Copy the data type <client interface>_Data
   In the example, the data block is called "Productionline_Data"
4. Paste the copied data block to the "Program blocks" folder.
5. Change the name of the data block inserted so that it is meaningful in your project.
   In the example, we change the name to "myProductionline_Data"

6. Now use the new data type that we created in Step 1.
   In the example, we assign the new data type to the tag "Variable" in the write list "WriteListStatus":

WriteListStatus	Struct
Variable	"myUDT_Productionline.WriteListStatus"
ProductionEnabled	Bool
Serial_Number	Array[0..4] of LInt
Serial_Number[0]	LInt
Serial_Number[1]	LInt
Serial_Number[2]	LInt
Serial_Number[3]	LInt
Serial_Number[4]	LInt

7. Supply the "Tag" parameter of the instruction "OPC_UA_WriteList" with the new data block (in the example "myProductionline_Data").
   `Variable := "myProductionline_Data"."WriteListStatus".Variable,`

Step 3: Create tag for OPC_UA_ReadList parameter "NodeAddInfos"

In this step, you create a local tag of the type "OPC_UA_NodeAdditionalInfo".

This tag is used to tell the "OPC_UA_WriteList" instruction which part of the array is to be assigned new values.

Proceed as follows:

1. Create an array tag of the type "OPC_UA_NodeAdditionalInfo".
   In the example, you create the tag in the declaration section of the function block that calls the instruction "OPC_UA_WriteList".
   You name the new local tag "NodeAdditionalInformation":

NodeAdditonalInformation	Array[0..1] of OPC_UA_NodeAdditionalInfo
NodeAdditonalInformation[0]	OPC_UA_NodeAdditionalInfo
AttributeID	UDInt
StartIndex	UDInt
EndIndex	UDInt
NodeAdditonalInformation[1]	OPC_UA_NodeAdditionalInfo
AttributeID	UDInt
StartIndex	UDInt
EndIndex	UDInt

   The new local tag "NodeAdditionalInformation" contains two elements of the type "OPC_UA_AdditonalInfo".
   NodeAdditionalInformation[0] provides additional information on the first tag to be written (in the example, "ProductionEnabled")
   NodeAdditionalInformation[1] provides additional information on the second tag to be written (in the example, "Serial_Number")

2. Assign values to the new tag (in the "Default values" column of the declaration section).
   In the example you do not write all elements of the second tag (Serial_Number).
   You therefore assign NodeAdditionalInformation[1].AttributeId the value of 13 (which means the values of the array are written).
   NodeAdditionalInformation[1].StartIndex is assigned the value of 5, as you are writing from (and including) index 5.
   NodeAdditionalInformation[1].EndIndex is assigned the value of 9, as you are writing from (and including) index 9.
   You set the following values for NodeAdditionalInformation[0] which signal to the server that the first element of the write list is not a part of an array but a scalar or an array that is to be written completely.
   The constant "#ignore" contains the value 4_294_967_295. The server ignores this value for StartIndex and EndIndex and writes scalars and arrays completely.

NodeAdditonalInformation	Array[0..1] of OPC_UA_NodeAdditionalInfo	
NodeAdditonalInformation[0]	OPC_UA_NodeAdditionalInfo	
AttributeID	UDInt	13
StartIndex	UDInt	#ignore
EndIndex	UDInt	#ignore
NodeAdditonalInformation[1]	OPC_UA_NodeAdditionalInfo	
AttributeID	UDInt	13
StartIndex	UDInt	5
EndIndex	UDInt	9

   Alternative:
   You can also set the values for NodeAdditionalInformation in the user program:

```
                // set additonal information to Array-Index 0
                // we set default values here
                #NodeAdditonalInformation[0].AttributeID := 13;
                #NodeAdditonalInformation[0].StartIndex :=
4_294_967_295;
```

```
                #NodeAdditonalInformation[0].EndIndex :=
4_294_967_295;

                // set additonal information to Array-Index 1
               // because we want to write only five values to the
array in the OPC UA server
                #NodeAdditonalInformation[1].AttributeID := 13;
                #NodeAdditonalInformation[1].StartIndex := 5;
                #NodeAdditonalInformation[1].EndIndex := 9;
```

Note:
The array NodeAdditionalInformation of the type "OPC_UA_NodeAdditionalInfo" must contain as many elements as the write list (2 in the example).

3. Assign the local tag NodeAdditionalInformation to the "NodeAddInfos" parameter of the "OPC_UA_WriteList" instruction.
   The following figure shows the call of OPC_UA_WriteList with the new supply of the parameters "Variable" and "NodeAddInfos":

```
#OPC_UA_WriteList_Instance(
                           NodeHdls :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeHd
ls,
                           NodeStatusList :=
"myProductionline_Data"."WriteListStatus".NodeStatusList,
                           Variable :=
"myProductionline_Data"."WriteListStatus".Variable,
                           ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                           Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeo
ut,
                           NodeHdlCount :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeCo
unt,
                        NodeAddInfos := #NodeAdditonalInformation,
                           REQ := #Req,
                           Busy => #Busy,
                           Error => #Error,
                           Status => #Status,
                           Done => #Done);
```

Result in the server

The "OP_UA_WriteList" instruction now writes only part of the "Serial_Number" array to the OPC UA server.

The figure below shows the array "Serial_Number". The server assigns the written values as of the index that you specify with NodeAdditionalInformation (as of index 5 in the example):

Serial_Number	Array[0..9] of LInt		
Serial_Number[0]	LInt	0	0
Serial_Number[1]	LInt	0	0
Serial_Number[2]	LInt	0	0
Serial_Number[3]	LInt	0	0
Serial_Number[4]	LInt	0	0
Serial_Number[5]	LInt	0	100
Serial_Number[6]	LInt	0	101
Serial_Number[7]	LInt	0	102
Serial_Number[8]	LInt	0	103
Serial_Number[9]	LInt	0	104

Note

Use the data type "OPC_UA_NodeAdditionalInfoExt" to write to a section of a multi-dimensional array (up to six dimensions).

Ending data exchange

OPC_UA_NodeReleaseHandleList: Release handles for read and write access

Validity

The following description of the "OPC_UA_WriteList" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

You use the instruction "OPC_UA_WriteList" to release handles.

The following figure shows the icon of the instruction in the editor (FBD).

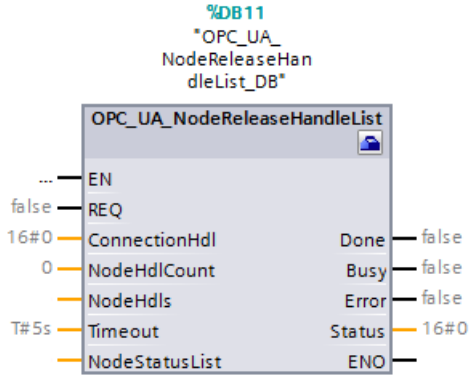

In the figure above, the parameters of the instruction are not supplied yet.

The instruction "OPC_UA_NodeReleaseHandleList" is used to release handles for PLC tags that were previously registered with the instruction "OPC_UA_NodeGetHandleList", see ③ in the figure below.

The instruction "OPC_UA_NodeReleaseHandleList" is not required when you want to disconnect the connection to the OPC UA server. In this case call the instruction "OPC_UA_Disconnect" immediately. This instruction automatically releases all handles.

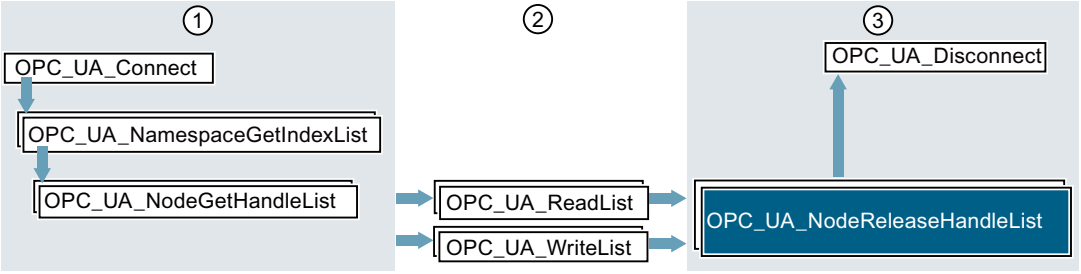

① Instructions for preparation of read and write operations

② Read and write instructions

③ Instructions for releasing resources after completed read or write operations

Parameters for "OPC_UA_NodeReleaseHandleList"

Table 4-78 The parameters of the function block "OPC_UA_NodeReleaseHandleList"

Parameters	Declaration in area	S7 data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the block.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established: Connection handle. You get the connection handle by calling the function block OPC_UA_Connect.
NodeHdlCount	Input	UINT	Number of elements in the NodeHdls parameter
NodeHdls	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the handles for the tags that you no longer want to read or set in your program.
Timeout	Input	TIME	Maximum time for execution of the function block. See also the explanation for this parameter in AUTOHOTSPOT.
NodeStatusList	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the error codes for the individual node handles (tags); see "Error numbers in the NodeStatusList parameter". For each node handle, you can see whether or not it could be released.

Parameters	Declaration in area	S7 data type	Meaning
Done	Output	BOOL	Status parameter for block processing • 0: Block processing aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	Output	BOOL	Status parameter for block processing • 0: Block not being executed • 1: Block currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	see "Error numbers in the Status parameter" below

Error numbers for Status

The following table shows a summary of error codes for this instruction:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Function block executed successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.

Error code (hexadecimal values)	Name of the error	Explanation
800F_0000	OpcUa_BadNothingToDo	Nothing to do: The OPC UA server received an empty list with no instructions from the OPC UA client.
8010_0000	OpcUa_BadTooManyOperations	The request could not be executed because it contained too many instructions.
B080_0100	Simatic_BadType_VariantInput1	The data type of the tag to which the first variant parameter of the instruction is pointing is incorrect.
B080_1100	Simatic_ArrayElements_TooMany	NodeHdlCount > MAX_ELEMENTS_NODELIST
B080_0200	Simatic_BadType_VariantInput2	The data type of the tag to which the second variant parameter of the instruction is pointing is incorrect.
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled.
B080_C500	Simatic_NothingToDo	Nothing to do: The instruction is using a list that contains no elements.
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Possible remedy: • Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

Error numbers for "NodeStatusList"

The "NodeStatusList" parameter contains an error code for each individual node handle (tag).

The following table explains the error codes:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Handle released successfully.
8034_0000	OpcUa_BadNodeUnknown	The node handle transferred is not known.
For more error codes, see AUTOHOTSPOT.		

This is how you use this instruction

Based on a program example, this section shows you how to use the instruction "OPC_UA_NodeReleaseHandleList" in a program that exchanges data with an OPC UA server.

Requirements

The following description assumes that:

- You have created a client interface.
- You have created and configured a connection to an OPC UA server, see "AUTOHOTSPOT".

In addition, the following requirements must be met for the instruction "OPC_UA_NodeReleaseHandleList":

- A handle for connection to an OPC UA server is available.
  You get the connection handle by calling the function block AUTOHOTSPOT.
- A list of the handles for the individual tags whose values you no longer want to read or write in your program.
  You get this handle list by calling the function block AUTOHOTSPOT.

Function of the instruction

You call the instruction "OPC_UA_NodeReleaseHandleList" to release node handles that you no longer require in your program.

Node handles are numeric references to nodes (for example PLC tags) on the OPC UA server.

What information is required for the function block?

The function block requires the following information:

- A list of the handles for the tags (or other nodes) whose values you no longer want to read or write (in the "NodeHdls" parameter).

What information does the function block return?

The function block returns the following information:

- A list of error messages (in the "NodeStatusList" parameter).
  Each error message in this list relates to the corresponding handle in the "NodeHdls" parameter.
  For each individual node handle, you need to check whether the OPC UA server was able to release it.
  If a node handle could not be released, you user program must catch this error: For example, call this instruction once again in a troubleshooting session.

How to use a configured connection

1. In the "Project tree" area, select the CPU you want to use as a client.
2. Select the function block that is to execute the client instruction in the "Program blocks" folder.
   In the example, the function block is called "ReadFromProductionline".
   Selected language: SCL.
3. Use drag-and-drop to move the instruction "OPC_UA_NodeReleaseHandleList" from the folder "Instructions > Communication > OPC UA > OPC UA Client" to the editor.

4. Select the call as multi-instance.
   STEP 7 creates an instance of this instruction and calls it "OPC_UA_NodeReleaseHandleList_Instance".
5. Click on the icon for "Start configuration" at the "OPC_UA_NodeReleaseHandleList_Instance" instruction.
   STEP 7 opens the "Configuration" tab in the Inspector window.
6. Under "Select client interface for OPC UA interface" you select the client interface that you want to use for the instruction.
   In the example, the client interface is called "Productionline", which we created for our AUTOHOTSPOT.
7. Click "File access" and select a read list.
   In the example, the read list is called "ReadListProduct".
   STEP 7 now automatically supplies the parameters of the instruction with the values you have configured for the client interface.
8. Click on "Block parameters" and assign tags manually to the parameters REQ, Busy, Done, Error, Status.
   STEP 7 adds the selected tag to the function call.

Calling the instruction (initial call)

You can find the complete program in the section "Program example for reading PLC tags (Page 3787)".

The program example uses the read list "ReadListProduct", which contains the following PLC tags:

- NewProduct
- ProductNumber

The handles for these PLC tags are released by the following program example.

CASE instruction

The example program is divided into multiple sections (cases) by a CASE instruction.

Case 5 is shown here:

SCL

```
5:  // case 5, release node handles

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_NodeReleaseHandleList_Instance(
                                                    NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                                    NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                                    REQ := #Req,
                                                    ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                    Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                    NodeHdlCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount,
                                                    Done => #Done,
                                                    Error => #Error,
                                                    Busy => #Busy,
                                                    Status => #Status);
```

Asynchronous execution

The instruction "OPC_UA_NodeReleaseHandleList" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction (initial call)

The figure above shows case 5.

This part of the program calls the instruction "OPC_UA_NodeReleaseHandleList". This releases handles for PLC tags. The tags are listed in the "ReadListProduct" read list.

If the instruction is not yet executed, #Busy is FALSE, so that the "#Req" tag is set to the value TRUE. This starts the instruction. In the next cycle, #Req is FALSE.

Calling the instruction (troubleshooting)

The figure below shows evaluation of the "Done" and "Error" parameters.

SCL

```
            IF #Done = TRUE THEN
              IF "Productionline_Data"."ReadListProduct".NodeStatusList[0]
= 0 AND
                 "Productionline_Data"."ReadListProduct".NodeStatusList[1]
= 0
                THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    //In case 100, to set REQ of instruction
"OPC_UA_Disconnect" to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at
NodeReleaseHandleList';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
               //In case 100, to set REG of instruction "OPC_UA_Disconnect"
to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_NodeReleaseHandleList_Instance(
                                                        NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                                        NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                                        REQ := FALSE,
                                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                        NodeHdlCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount);
            END_IF;
```

Explanation of the call of the instruction (troubleshooting)

If an error occurs, the "Error" output parameter is set to the value TRUE. This sets the #State tag to the value 100. This case is reserved for troubleshooting. The example program also calls the instruction "OPC_UA_NodeReleaseHandleList" to set the REQ parameter to FALSE.

If the TRUE value is in the "Done" output parameter, the instruction was successfully executed.

Please note: It is possible that the instruction was successfully executed ("Error" parameter not set, "Done" parameter set) but a handle could not be released for a PLC tag.

Therefore, there is a check in the example to determine whether the handles for both tags have been released:

- If the server releases the handle for the first PLC tag, the first element of the array to which the "NodeStatusList" parameter is pointing contains the value 0.
  In the example, the status code is available in "Productionline_Data"."ReadListProduct".NodeStatusList[0].
- If the server releases the handle for the second PLC tag, the second element of the array to which the "NodeStatusList" parameter is pointing contains the value 0.
  In the example, the status code is available in "Productionline_Data"."ReadListProduct".NodeStatusList[1].

If both conditions are met, the value of tag "#State" is increased by one. This means the next program section - case 6 (close connection) - is executed in the next cycle.

Case 6 is described in AUTOHOTSPOT.

Note

Case 6 is for enabling the connection to the OPC UA server. This also releases all other resources that the client program has occupied in the server. If you want to terminate the connection to the server, your user program can execute case 6 immediately and skip case 5.

See also

OPC UA instructions for client programs (Page 3640)

OPC_UA_Disconnect: Close the connection

Validity

The following description of the "OPC_UA_Disconnect" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

You use the instruction "OPC_UA_Disconnect" to disconnect the connection to a OPC UA server.

The following figure shows the icon of the instruction in the editor (FBD).

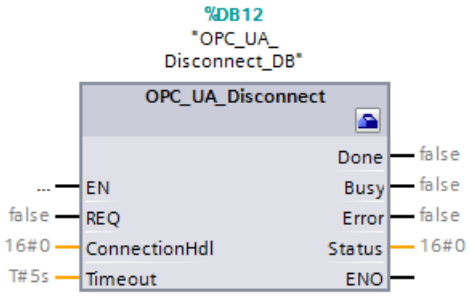

In the figure above, the parameters of the instruction are not supplied yet.

The instruction "OPC_UA_Disconnect" is used to disconnect the connection to an OPC UA server and terminate the session, see ③ in the figure below:

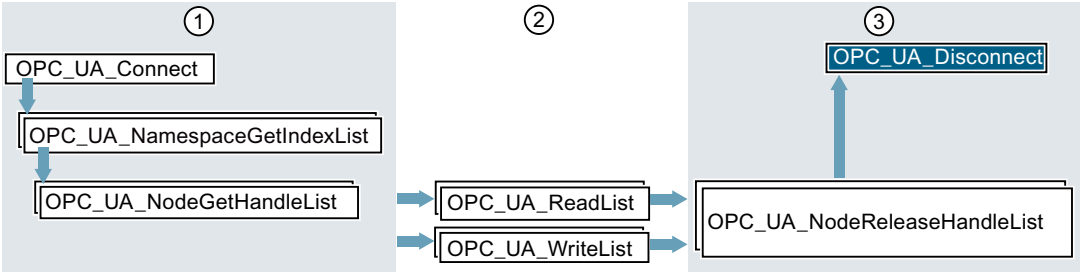

① Instructions for preparation of read and write operations

② Read and write instructions

③ Instructions for releasing resources after completed read or write operations

Parameters for "OPC_UA_Disconnect"

Table 4-79 The parameters of the function block "OPC_UA_Disconnect"

Parameters	Declaration in area	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the block.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established: Connection handle. You get the connection handle by calling the function block OPC_UA_Connect.
Timeout	Input	TIME	Maximum time for execution of the function block. See also the explanation for this parameter in OPC_UA_Connect: Create connection (Page 3643).
Done	Output	BOOL	Status parameter for block processing • **0**: Block processing aborted, not yet completed or not yet started • **1**: Block processing completed without errors

Parameters	Declaration in area	Data type	Meaning
Busy	Output	BOOL	Status parameter for block processing • 0: Block not being executed • 1: Block currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	see "Error numbers in the Status parameter" below

Error numbers in the Status parameter

The "Status" parameter provides information about errors that occurred during execution of the instruction.

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Function block executed successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.

Error code (hexadecimal values)	Name of the error	Explanation
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.
800F_0000	OpcUa_BadNothingToDo	Nothing to do: The OPC UA server received an empty list with no instructions from the OPC UA client.
8010_0000	OpcUa_BadTooManyOperations	The request cannot be executed because it contains too many instructions.
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled.
B080_C600	Simatic_ClientNotAvailable	Error during initialization of the client
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Possible remedy: • Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

This is how you use this instruction

Based on a program example, this section shows you how to use the instruction "OPC_UA_Disconnect" in a program that exchanges data with an OPC UA server.

Requirements

The following description assumes that:

- You have created a client interface, see "AUTOHOTSPOT".
- You have created and configured a connection to an OPC UA server, see "AUTOHOTSPOT".

In addition, the following requirement must be met for the instruction "OPC_UA_Disconnect":

- A handle for connection to an OPC UA server is available.
  You get the connection handle by calling the instruction OPC_UA_Connect: Create connection (Page 3643).

Function of the instruction

You call the instruction "OPC_UA_Disconnect" to disconnect the connection to an OPC UA server.

How to use a configured connection

1. In the "Project tree" area, select the CPU you want to use as a client.
2. Select the function block that is to execute the client instruction in the "Program blocks" folder.
   In the example, the function block is called "ReadFromProductionline".
   Selected language: SCL.
3. Use drag-and-drop to move the instruction "OPC_UA_Connect" from the folder "Instructions > Communication > OPC UA > OPC UA Client" to the editor.
4. Select the call as multi-instance.
   STEP 7 creates an instance of this instruction and calls it "OPC_UA_Disconnect_Instance".
5. Click on the icon for "Start configuration" at the "OPC_UA_Disconnect_Instance" instruction.
   STEP 7 opens the "Configuration" tab in the Inspector window.
6. Under "Select client interface for OPC UA interface" you select the client interface that you want to use for the instruction.
   In the example, the client interface is called "Productionline", which was created for the AUTOHOTSPOT, see "AUTOHOTSPOT".
   STEP 7 now automatically supplies the parameters of the instruction with the values you have configured for the client interface.
7. Click on "Block parameters" and assign tags manually to the remaining parameters REQ, Busy, Done, Error, Status.
   STEP 7 adds the selected tag to the call of the instruction.

Calling the instruction

You can find the complete program in the section "AUTOHOTSPOT".

The example program terminates the connection to an OPC UA server and the session:

SCL

```
6:  // case 6, disconnect from server

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        REQ := #Req,
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        Done => #Done,
                                        Error => #Error,
                                        Busy => #Busy,
                                        Status => #Status);
```

The example program calls the "OPC_UA_Disconnect" instruction for the following reasons:

- To terminate the connection to the OPC UA server.
- To set the REQ parameter to FALSE, if an error has occurred.

Asynchronous execution

The instruction "OPC_UA_Disconnect" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction

The figure above shows case 6. This part of the program terminates the connection to the connected OPC UA server and the session.

If the #State tag is 6, then case 6 is executed.

If the "OPC_UA_Disconnect" instruction is not yet executed, #Busy is FALSE, so that the "#Req" tag is set to the value TRUE. This starts the instruction. In the next cycle, #Req is FALSE.

If the output parameter "Done" is TRUE, the handle for the connection could be released and the connection could be terminated. This increases the value of the "#State" tag by one. This means the next program section - case 7 - is executed in the next cycle. This case sets the status information for the function block (Output_Busy = FALSE, Output_Done = TRUE).

If the instruction cannot be executed without errors, the output parameter "Error" is TRUE and the tag #State gets the value 100. This case is reserved for troubleshooting.

Diagnostics

OPC_UA_ConnectionGetStatus: Read connection status

Validity

The following description of the "OPC_UA_ConnectionGetStatus" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

You use the instruction "OPC_UA_ConnectionGetStatus" to get information about the quality of the connection to an OPC UA server.

OPC_UA_ConnectionGetStatus is used to prepare the data exchange, see ① in the figure below.

The instruction returns information on the connection, the server and the service level.

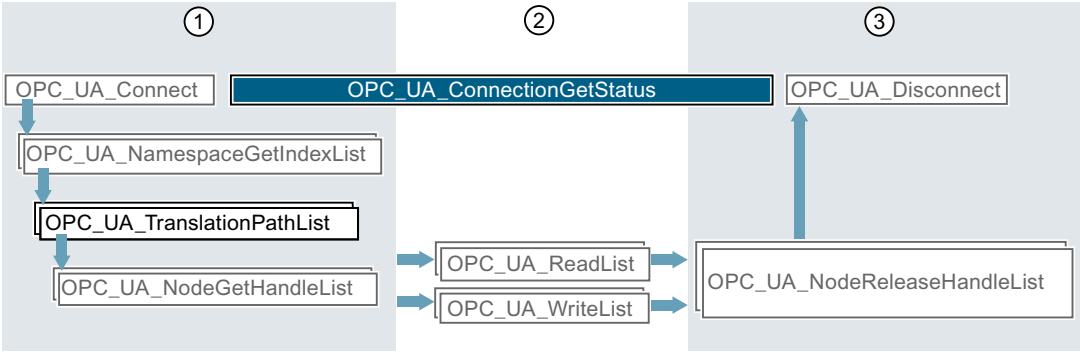

① Instructions for preparation of read and write operations

② Read and write instructions

③ Instructions for "clean-up" after a completed read or write operation

Parameters for "OPC_UA_ConnectionGetStatus"

Table 4-80 The parameters of the function block "OPC_UA_ConnectionGetStatus"

Parameters	Declaration in area	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the instruction.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established. You get the handle from the instruction AUTOHOTSPOT.

Parameters	Declaration in area	Data type	Meaning
Timeout	Input	TIME	Maximum time for execution of the instruction in milliseconds. See also the explanation for this parameter in AUTOHOTSPOT.
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error numbers for Status" below
ConnectionStatus	Output	UDINT	Status of the connection that is specified in the ConnectionHdl parameter. Meaning: • 0: Connection to the OPC UA server has been established. • 1: Connection with errors to the OPC UA server. • 2: The connection to the OPC UA server has been ended by the server.

Parameters	Declaration in area	Data type	Meaning
ServerState	Output	UDINT	information to the OPC UA server Meaning of the values: • 0: The server is running normally. • 1: An error has occurred in the server. The server is no longer operating reliably. • 2: The server is running. However, no configuration has been loaded and data is therefore not being exchanged. • 3: The server is temporarily unavailable. • 4: The server has been or is in the process of being shut down. • 5: The server is running in test mode. Outputs are not interconnected. • 6: The server is running normally. However, there are problems with reading the PLC tags. • 7: Server status is not known. This value is set if there is no connection to the server.
ServiceLevel	Output	Byte	Information on the power of the connected OPC UA server. The server returns a number between 0 and 255 to indicate its power: The higher the number, the more powerful the server.

Error numbers for Status

The "Status" parameter provides information about errors that occurred during execution of the instruction.

The following table shows a summary of error codes for this instruction:

Error code (hexadecimal values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Instruction finished successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.

Error code (hexadecimal values)	Name of the error	Explanation
8009_0000	OpcUa_BadUnknownResponse	The server sends an unknown or invalid response.
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.
8010_0000	OpcUa_BadTooManyOperations	The request could not be executed because it contained too many instructions.
8074_0000	OpcUa_BadTypeMismatch	ServerState / ServiceLevel Value(s) have unexpected type
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled.
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Possible remedy: • Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

Requirement

The following requirement must be met for the instruction "OPC_UA_ConnectionGetStatus":

- A handle for connection to an OPC UA server is available.
  You get the connection handle from the instruction AUTOHOTSPOT.

Calling the instruction

This section describes the function block "Analyze_Connection". The function block shows you how to use the instruction "OPC_UA_ConnectionGetStatus".

You can find the complete example in the section "Example program for OPC_UA_ConnectionGetStatus (Page 3781)".

Declaration of tags

Declare an instance of the instruction "OPC_UA_ConnectionGetStatus" and the tags with which you are supplying the instruction parameters.

The program example uses the following declaration:

ConnectionHdl	DWord
ConnectionStatus	UDInt
ServerState	UDInt
ServiceLevel	Byte
▸ OPC_UA_ConnectionGetStatus_Instance	OPC_UA_ConnectionGetStatus

User program

The figure below shows the section from the example program "Analyze_Connection" that determines the quality of the connection to the server.

The example program is divided into multiple sections (cases) by a CASE instruction.

Case 2 (analysis of the connection) is shown here:

SCL

```
2: // case 2, analyse connection
   IF #Busy = FALSE THEN
       #Req := TRUE;
   ELSE
       #Req := FALSE;
   END_IF;

   #OPC_UA_ConnectionGetStatus_Instance(ConnectionHdl := #ConnectionHdl,
                                        ConnectionStatus => #ConnectionStatus,
                                        ServerState => #ServerState,
                                        ServiceLevel => #ServiceLevel,
                                        Timeout := T#6S,
                                        Done => #Done,
                                        Error => #Error,
                                        Status => #Status,
                                        Busy => #Busy,
                                        REQ := #Req);
   IF #Done = TRUE THEN
       #State := #State + 1;
   END_IF;

   IF #Error = TRUE THEN
       #State := 100;
       #Mem_Status := #Status;
       #OPC_UA_ConnectionGetStatus_Instance(REQ := FALSE, ConnectionHdl := #ConnectionHdl);
   END_IF;
```

If an error occurs, the instruction "OPC_UA_ConnectionGetStatus" (#Error has the value "TRUE"), the parameter REQ is set to the value FALSE.

Asynchronous execution

The instruction "OPC_UA_NodeGetHandleList" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction

The figure above shows case 2 of the example program "Analyze_Connection".

This part of the program requests the current values for "ConnectionStatus", "ServerState" and "ServiceLevel" from the connected OPC UA server.

If the #State tag is 2, then case 2 is executed.

If the "OPC_UA_ConnectionGetStatus" instruction is not yet executed, #Busy is FALSE, so that the "#Req" tag is set to TRUE. This starts the instruction. In the next cycle, #Req is FALSE.

If the output parameter "Error" is TRUE, an error occurred during execution of the instruction. This sets the "#State" tag to the value 100. This case is reserved for troubleshooting.

If the output parameter "Done" is TRUE, the instruction was executed successfully. This increases the value of the "#State" tag by one. This means the next program section - case 3 - is executed in the next cycle. Case 3 is described in OPC_UA_Disconnect: Close the connection (Page 3723).

Methods

OPC_UA_MethodGetHandleList: Get handles for method calls

Validity

The following description of the "OPC_UA_MethodGetHandleList" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

A programming example shows how to use the instruction in SCL.

This section assumes that you are creating and configuring a client interface for OPC UA communication; see "AUTOHOTSPOT". The efficient procedure ensures that most parameters for the client instruction are supplied automatically.

You use the instruction "OPC_UA_MethodGetHandleList" in your user program to receive numerical references (handles) from the server for methods that the server provides for OPC UA clients. OPC_UA_MethodGetHandleList is used to prepare method calls, see ① in the figure below.

The instruction returns the handles for the server methods.

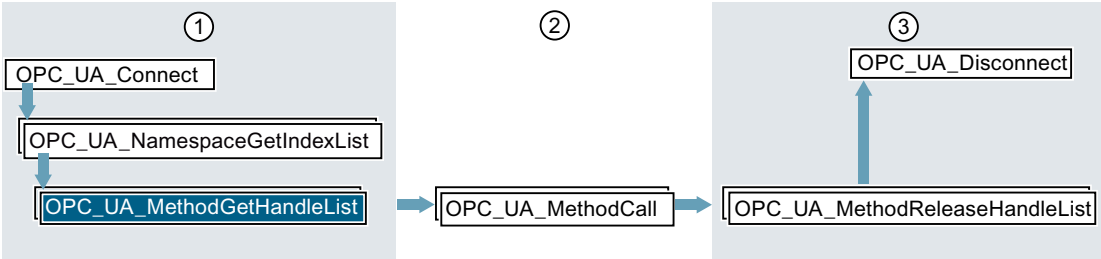

① Instructions for preparation of method calls

② Method calls

③ Instructions for "clean-up" after completed method calls

Parameters for "OPC_UA_MethodGetHandleList"

Table 4-81 The parameters of the instruction "OPC_UA_MethodGetHandleList"

Parameters	Declaration in area	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the instruction.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established. You get the handle from the instruction OPC_UA_Connect.
NodeIDCount	Input	UINT	Number of elements in the ObjectNodeIDs or MethodeNodeIDs parameter
ObjectNodeIDs	InOut	VARIANT	Pointer to an array with elements of the type OPC_UA_NodeId. The array contains the node IDs (NodeIDs) of the objects (folders) in which the methods are located.
MethodNodeIDs	InOut	VARIANT	Pointer to an array with elements of the type OPC_UA_NodeId. The array contains the node IDs (NodeIDs) of the methods for which handles are requested from the server.
Timeout	Input	TIME	Maximum time for execution of the instruction in milliseconds. See also explanation for this parameter in AUTOHOTSPOT
NamespaceIndexCount	Input	UINT	Number of array elements in the parameter NamespaceIndexes.

Parameters	Declaration in area	Data type	Meaning
NamespaceIndexes	InOut	VARIANT	Pointer to an array of the type UINT. This parameter is not required if there are no elements in the NamespaceIndexCount parameter (NamespaceIndexCount = 0). The array is used to assign new indexes to the namespace indexes in the NodeIDs parameter. Example: If there is a value of 10 in NamespaceIndexes[4], all NodeIDs in the "NodeIDs" parameter that previously had the namespace index 4 are assigned the namespace index 10.
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error numbers for Status" below
StatusList	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the error codes for the individual methods; see "Error numbers for NodeStatusList" below. For each method, the system specifies whether or not a corresponding handle could be found.
MethodHdls	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the handles for the individual methods (NodeIDs) returned by the OPC UA server.

Determining the "ObjectNodeIDs" and "MethodNodeIDs" parameters

ObjectNodeIDs

You can determine OPC_UA_NodeId, which must be entered at the "ObjectNodeIDs" parameter, for example with UaExpert:

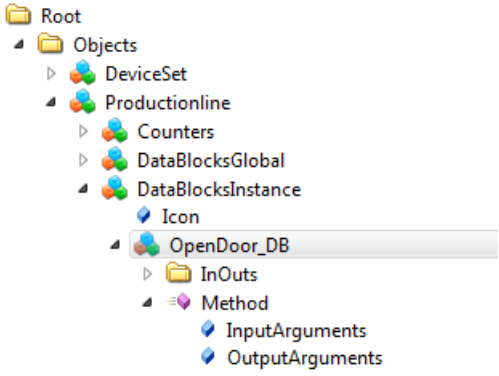

NodeId	NodeId
NamespaceIndex	3
IdentifierType	String
Identifier	"OpenDoor_DB"

MethodNodeIDs

You can find the OPC_UA_NodeId at the "MethodNodeIDs" parameter here:

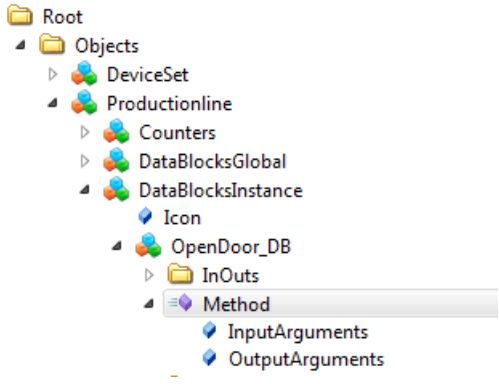

NodeId	NodeId
NamespaceIndex	3
IdentifierType	String
Identifier	"OpenDoor_DB".Method

Error numbers for Status

The following table shows a summary of error codes for this instruction:

Error code (hex. values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Instruction finished successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.
8009_0000	OpcUa_BadUnknownResponse	Server does not respond with the expected number of results
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.
800F_0000	OpcUa_BadNothingToDo	Nothing to do: The OPC UA server received an empty list with no instructions from the OPC UA client.
8010_0000	OpcUa_BadTooManyOperations	The request could not be executed because it contained too many instructions.
8074_0000	OpcUa_BadTypeMismatch	You are using a data type that does not correspond to the data type in the server.
B080_0100	Simatic_BadType_VariantInput1	The data type of the tag to which the first variant parameter of the instruction is pointing is incorrect.
B080_0200	Simatic_BadType_VariantInput2	The data type of the tag to which the second variant parameter of the instruction is pointing is incorrect.
B080_0300	Simatic_BadType_VariantInput3	The data type of the tag to which the third variant parameter of the instruction is pointing is incorrect.

Error code (hex. values)	Name of the error	Explanation
B080_0400	Simatic_BadType_VariantInput4	The data type of the tag to which the fourth variant parameter of the instruction is pointing is incorrect.
B080_0500	Simatic_BadType_VariantInput5	The data type of the tag to which the fifth variant parameter of the instruction is pointing is incorrect.
B080_1100	Simatic_ArrayElements_TooMany	Maximum number allowed for method list (Max_ELEMENTS_METHODLIST) exceeded.
B080_3100 B080_3200 B080_3300 B080_3400 B080_3500 General: B080_3N00(N=Nth VARIANT of the instruction)	Example for B080_3300: Simatic_BadNumElements_VariantInput3 General: Simatic_BadNumElements_VariantInputN (N=nth version of the instruction)	Example for wrong number of array elements at the 3rd VARIANT parameter: • NamespaceIndexCount > Number of array elements from NamespaceIndexes (3rd VARIANT parameter). Example for wrong number of array elements at the first VARIANT parameter: • NodeIDCount > Number of array elements from ObjectNodeIDs (1st VARIANT parameter). General: Wrong number of elements for the Nth VARIANT parameter.
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled.
B080_C500	Simatic_NothingToDo	Nothing to do: The instruction is using a list that contains no elements.
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Possible remedy: • Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

Error numbers for StatusList

The StatusList parameter contains an error code for each individual node ID (method).

The following table explains the error codes:

Error code (hex. values)	Name of the error	Explanation
0000_0000	OpcUa_Good	No errors
8033_0000	OpcUa_BadNodeIdInvalid	The syntax of the NodeID is incorrect.
8034_0000	OpcUa_BadNodeIdUnknown	The NodeID refers to a node (method) that does not exist on the OPC UA server.
8074_0000	OpcUa_BadTypeMismatch	InputArguments / OutputArguments have incorrect type
For more error codes, see AUTOHOTSPOT.		

Requirements

The following description assumes that:

- You have created a client interface, see "AUTOHOTSPOT".
- You have created and configured a connection to an OPC UA server, see "AUTOHOTSPOT".

In addition, the following requirements must also be met for the instruction "OPC_UA_MethodGetHandleList" (cf diagram above):

- A handle for connection to an OPC UA server is available.
  You get the connection handle from the instruction OPC_UA_Connect.
- Indexes of the namespaces in which the methods you want to call are located.
  You get the indices from the instruction AUTOHOTSPOT.

Function of the instruction

To speed up method calls, OPC UA servers use numeric references (handles) for methods.

You get these handles from the OPC UA server with the instruction "OPC_UA_MethodGetHandleList".

You need these handles for the OPC UA instruction for calling methods, "OPC_UA_MethodCall".

How to use a configured connection

1. In the "Project tree" area, select the CPU you want to use as a client.
2. Select the function block that is to execute the client instruction in the "Program blocks" folder.
   In the example, the function block is called "Call_OpenDoor_On_Productionline".
   Selected language: SCL.
3. Use drag-and-drop to move the instruction "OPC_UA_MethodGetHandleList" from the folder "Instructions > Communication > OPC UA > OPC UA Client" to the editor.
4. Select the call as multi-instance.
   STEP 7 creates an instance of this instruction and calls it "OPC_UA_MethodGetHandleList_Instance".
5. Click on the icon for "Start configuration" at the "OPC_UA_MethodGetHandleList_Instance" instruction.
   STEP 7 opens the "Configuration" tab in the Inspector window.
6. Under "Select client interface for OPC UA interface", you select the client interface that you want to use for the instruction.
   In the example, the client interface is called "Productionline". see "AUTOHOTSPOT".
7. Click "File access" and select a read list.
   In the example, the method list is called "MethodListOpenDoor".
   STEP 7 now automatically supplies most parameters of the instruction with the correct tags.
8. Click on "Block parameters" and assign tags manually to the remaining parameters.
   STEP 7 adds the selected tag to the function call.

Calling the instruction (initial call)

The following excerpt from the program "Call_OpenDoor_On_Productionine" shows how to use the instruction "OPC_UA_MethodGetHandleList".

You can find the complete example program in the section AUTOHOTSPOT

The example program is divided into multiple sections (Cases) by a CASE instruction.

Case 3 is shown here:

SCL

```
3:  // case 3, get an handle for server method OpenDoor

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_MethodGetHandleList_Instance(
                                                REQ := #Req,
                                                ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                ObjectNodeIDs :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".ObjectNode
s,
                                                MethodNodeIDs :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodNode
s,
                                                NamespaceIndexCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                Done => #Done,
                                                Busy => #Busy,
                                                Error => #Error,
                                                Status => #Status,
                                                StatusList :=
"Productionline_Data"."MethodListOpenDoor".MethodStatusList,
                                                MethodHdls :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls
,
                                                Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                NodeIDCount :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodCoun
t,
                                                NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes);
```

```
111  3:  // case 3, get an handle for server method OpenDoor
112
113  IF #Busy = FALSE THEN
114      #Req := TRUE;
115  ELSE
116      #Req := FALSE;
117  END_IF;
118  #OPC_UA_MethodGetHandleList_Instance(
119                  REQ := #Req,
120                  ConnectionHdl := "Productionline_Configuration".Connection.ConnectionHdl,
121                  ObjectNodeIDs := "Productionline_Configuration".MethodLists."MethodListOpenDoor".ObjectNodes,
122                  MethodNodeIDs := "Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodNodes,
123                  NamespaceIndexCount := "Productionline_Configuration".Namespaces.NamespaceCount,
124                  Done => #Done,
125                  Busy => #Busy,
126                  Error => #Error,
127                  Status => #Status,
128                  StatusList := "Productionline_Data"."MethodListOpenDoor".MethodStatusList,
129                  MethodHdls := "Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls,
130                  Timeout := "Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
131                  NodeIDCount := "Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodCount,
132                  NamespaceIndexes := "Productionline_Configuration".Namespaces.ServerNamespaceIndexes);
133
134
```

Asynchronous execution

The instruction "OPC_UA_MethodGetHandleList" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction (initial call)

The figure above shows case 3. This part of the program registers the "OpenDoor" method on the OPC UA server. The "OPC_UA_MethodGetHandleList" method is called for this.

If the instruction is not yet executed, #Busy is FALSE, so that the "#Req" tag is set to the value TRUE. This starts the instruction. In the next cycle, #Req is FALSE.

Calling the instruction (troubleshooting)

The figure below shows evaluation of the "Done" and "Error" parameters.

SCL

```
             IF #Done = TRUE THEN
                IF
"Productionline_Data"."MethodListOpenDoor".MethodStatusList[0] = 0
THEN      //one method
                    #State := #State + 1;
                ELSE
                    #State := 100;
                  //In case 100, set REQ of instruction "OPC_UA_Disconnect"
to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at
MethodGetHandleList';
                END_IF;
            END_IF;
            IF #Error = TRUE THEN
                #State := 100;
               //In case 100, to set REG of instruction "OPC_UA_Disconnect"
to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_MethodGetHandleList_Instance(
                                                        ObjectNodeIDs :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".ObjectNode
s,
                                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                        MethodNodeIDs :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodNode
s,
                                                        StatusList :=
"Productionline_Data"."MethodListOpenDoor".MethodStatusList,
                                                        MethodHdls :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls
,
                                                        REQ := FALSE,
                                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                        NodeIDCount :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodCoun
t,
                                                       NamespaceIndexCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                        NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes);
            END_IF;
```

Explanation of the call of the instruction (troubleshooting)

If an error occurs, the value TRUE is set at the "Error" output parameter. This sets "#State" to the value 100. This case is reserved for troubleshooting.

If the output parameter "Done" is TRUE, the instruction was executed successfully.

It is possible that the instruction was successfully executed ("Error" parameter not set, "Done" parameter set) but no handle could be returned for a method.

Therefore, there is a check in the example to determine whether the handle for the "OpenDoor" method has been returned:

- If the condition is met, the value of the "#State" tag is increased by one. This means the next program section - case 4 (Call method) - is executed in the next cycle. Case 4 is described in OPC_UA_MethodCall: Call method (Page 3744).
- If the condition is not met, this is assessed as an error and #State has a value of 100. The error case is run through in the next cycle.

Note

If you register a multiple methods with OPC_UA_MethodGetHandleList in your application, it can be useful always to increase #State by one when #Done = TRUE and only check whether a valid handle was returned for a given method in question before the individual methods are called with "OPC_UA_MethodCall" (in case 4).

OPC_UA_MethodCall: Call method

Validity

The following description of the "OPC_UA_MethodCall" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

A programming example shows how to use the instruction in SCL.

This section assumes that you are creating and configuring a client interface for OPC UA communication; see "AUTOHOTSPOT". The efficient procedure ensures that most parameters for the client instruction are supplied automatically.

You use the instruction "OPC_UA_MethodCall" in your user program to call a method on the server.

The figure below shows the structure of a user program. The instruction "OPC_UA_MethodCall" is located in area 2:

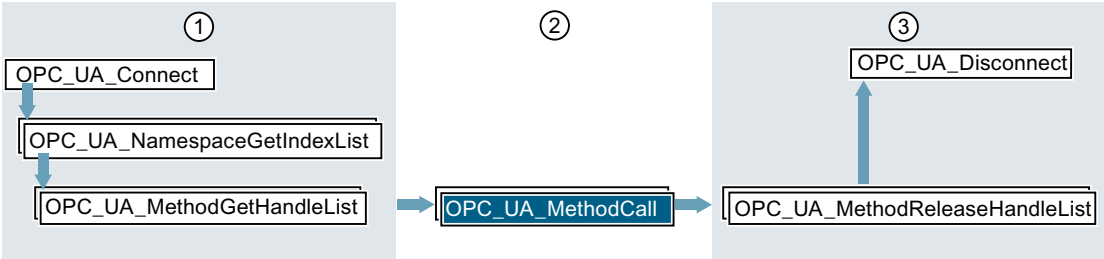

① Instructions for preparation of method calls

② Method calls

③ Instructions for "clean-up" after completed method calls

Parameters for "OPC_UA_MethodCall"

Table 4-82 The parameters of the instruction "OPC_UA_MethodCall"

Parameters	Declaration in area	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the instruction.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established. You get the handle from the instruction AUTOHOTSPOT.
Timeout	Input	TIME	Maximum time for execution of the instruction in milliseconds. See also the explanation for this parameter in AUTOHOTSPOT.
MethodHdl	Input	DWORD	Handle for the method to be called.
InputArguments	InOut	VARIANT	Pointer to a tag with the PLC data type (UDT) or STRUCT, the name, number, order and data type of which correspond to the expected input parameters.
OutputArguments	InOut	VARIANT	Pointer to a tag with the PLC data type (UDT) or STRUCT, the name, number, order and data type of which correspond to the expected output parameters.
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors

Parameters	Declaration in area	Data type	Meaning
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error numbers for Status" below
MethodResult	InOut	VARIANT	Pointer to a tag of the type DWORD that contains the error code (result) of the server method called. This error code is provided by the user program of the OPC UA server. In the case of an S7-1500 CPU as OPC UA server, it is the "UAMethod_Result" parameter of the server instruction "OPC_UA_ServerMethodPost" that you can read out with the client instruction "OPC_UA_MethodCall". If the instruction "OPC_UA_MethodCall" was not called successfully (status & 0x8000_0000 ≠ 0), no value is set for MethodResult.

Error numbers for Status

The Status parameter provides information about errors that occur during execution of the instruction.

The following table explains the error codes:

Error code (hex. values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Instruction finished successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.

Error code (hex. values)	Name of the error	Explanation
8006_0000	OpcUa_BadEncodingError	Error decoding the output argument
8009_0000	OpcUa_BadUnknownResponse	Server does not respond with the expected number of results
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.
800F_0000	OpcUa_BadNothingToDo	Nothing to do: The OPC UA server received an empty list with no instructions from the OPC UA client.
8010_0000	OpcUa_BadTooManyOperations	The request cannot be executed because it contains too many instructions.
8034_0000	BadNodeIdUnknown	The node id refers to a node that does not exist in the server address space. The NamespaceIndex of the specified NodeID may not exist. If the error code is displayed when the instruction "OPC_UA_MethodCall" is called: Another possible cause is the use of invalid ObjectIDs in the preceding call of OPC_UA_MethodGetHandleList. The instruction "OPC_UA_MethodGetHandleList" does not evaluate the ObjectIDs.
8074_0000	OpcUa_BadTypeMismatch	Argument variant has incorrect type.
8075_0000	OpcUa_BadMethodInvalid	No registered method for the specified MethodHdl
80AB_0000	OpcUa_BadInvalidArgument	Input arguments have incorrect type.
B080_0100	Simatic_BadType_VariantInput1	The data type of the tag to which the first variant parameter of the instruction is pointing is incorrect.
B080_0200	Simatic_BadType_VariantInput2	The data type of the tag to which the second variant parameter of the instruction is pointing is incorrect.
B080_1100	Simatic_ArrayElements_TooMany	General error code. Occurs when an array has too many elements.
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled.
B080_C500	Simatic_NothingToDo	Nothing to do: The instruction is using a list that contains no elements.

Error code (hex. values)	Name of the error	Explanation
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Possible remedy: • Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

Requirements

The following description assumes that:

- You have created a client interface, see "AUTOHOTSPOT".
- You have created and configured a connection to an OPC UA server, see "AUTOHOTSPOT".

In addition, the following requirements must also be met for the instruction "OPC_UA_MethodCall" (cf diagram above):

- A handle for connection to an OPC UA server is available.
  You get the connection handle from the instruction AUTOHOTSPOT.
- Indexes of the namespaces that contain the methods that you want to call with the client.
  You get the indexes from the instruction "OPC_UA_NamespaceGetIndexList".
  A handle for the method that you want to call in your user program.
  You get the handle from the instruction AUTOHOTSPOT.

Function of the instruction

You use the instruction "OPC_UA_MethodCall" to call a method that has been released in an OPC UA server.

How to use a configured connection

1. In the "Project tree" area, select the CPU you want to use as a client.
2. Select the function block that is to execute the client instruction in the "Program blocks" folder.
   In the example, the function block is called "Call_OpenDoor_On_Productionline".
   Selected language: SCL.
3. Use drag-and-drop to move the instruction "OPC_UA_MethodCall" from the folder "Instructions > Communication > OPC UA > OPC UA Client" to the editor.
4. Select the call as multi-instance.
   STEP 7 creates an instance of this instruction and calls it "OPC_UA_MethodCall_Instance".
5. Click on the icon for "Start configuration" at the "OPC_UA_MethodCall_Instance" instruction.
   STEP 7 opens the "Configuration" tab in the Inspector window.
6. Under "Select client interface for OPC UA interface", you select the client interface that you want to use for the instruction.
   In the example, the client interface is called "Productionline".

7. Click "File access" and select a method list.
   In the example, the method list is called "MethodListOpenDoor".
   STEP 7 automatically supplies most parameters of the instruction with tags, but not the parameters Busy, Done, Error.
8. Create the local tags for the parameters yet to be supplied.
9. Click on "Block parameters" and assign the local tags to the remaining parameters.
   STEP 7 adds the selected tag to the function call.

Calling the instruction (initial call)

The following excerpt from the program "Call_OpenDoor_On_Productionine" shows how to use the instruction "OPC_UA_MethodGetHandleList".

You can find the complete example program in the section AUTOHOTSPOT

The example program is divided into multiple sections (Cases) by a CASE instruction.

Case 4 is shown here:

SCL

```
        4:  // case 4, call method OpenDoor

            IF #Init_Method_InputParameter = TRUE THEN
                #Init_Method_InputParameter := FALSE;
                // for our server method at Productionline, we set input
parameters to 1

"Productionline_Data".MethodListOpenDoor.Method.Inputs.Number := 1;
            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_MethodCall_Instance(
                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        MethodHdl :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls
[0],
                                        Status =>
"Productionline_Data"."MethodListOpenDoor".MethodStatusList[0],
                                        InputArguments :=
"Productionline_Data"."MethodListOpenDoor"."Method".Inputs,
                                        OutputArguments :=
"Productionline_Data"."MethodListOpenDoor"."Method".Outputs,
                                        REQ := #Req,
                                        Done => #Done,
                                        Busy => #Busy,
                                        Error => #Error,
                                        MethodResult := #MethodResult);

        #Output_MethodResult := #MethodResult;
```

Asynchronous execution

The instruction "OPC_UA_MethodCall" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction (initial call)

The figure above shows case 4. This part of the program calls the server method "OpenDoor" on the OPC UA server.

The "OPC_UA_MethodCall" instruction receives the handle for the server method and the input parameters.

In the example, only the "Number" input parameter is used. The value is set to 1, so that the server method can be fully executed.

Calling the instruction (troubleshooting)

The figure below shows evaluation of the "Done" and "Error" parameters.

SCL

```
            IF #Done = TRUE THEN
                IF #MethodResult < 16#8000_0000 THEN
                    #State := #State + 1;
                ELSE
                    //server method did not run successfully
                    #State := 100;
                  //In case 100, set REQ of instruction "OPC_UA_Disconnect"
to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#' Error at server
method';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
               //In case 100, to set REQ of instruction "OPC_UA_Disconnect"
to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status :=
"Productionline_Data".MethodListOpenDoor.MethodStatusList[0];
                #OPC_UA_MethodCall_Instance(
                                            ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                            MethodHdl :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls
[0],
                                            REQ := FALSE,
                                            Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                            Status =>
"Productionline_Data"."MethodListOpenDoor".MethodStatusList[0],
                                            InputArguments :=
"Productionline_Data"."MethodListOpenDoor"."Method".Inputs,
                                            OutputArguments :=
"Productionline_Data"."MethodListOpenDoor"."Method".Outputs,
MethodResult := #MethodResult);
            END_IF;
```

Explanation of the call of the instruction (troubleshooting)

If the output parameter "Done" is TRUE, the "OPC_UA_MethodCall" instruction was executed successfully.

Also check whether the method called on the OPC UA server, "Productionline", was executed successfully. This is the case if the value of the "MethodResult" parameter is "0".

If executed successfully, the value of the "#State" tag is increased by one. This means the next program section - case 5 - is executed in the next cycle.

Case 5 is described in OPC_UA_MethodReleaseHandleList: Release handles for method calls (Page 3752).

If an error occurs, the "OPC_UA_MethodCall" instruction a the "Error" output parameter is set to the value TRUE. This sets the value of the "#State" tag to 100. This case is reserved for troubleshooting.

OPC_UA_MethodReleaseHandleList: Release handles for method calls

Validity

The following description of the "OPC_UA_MethodReleaseHandleList" instruction applies to S7-1500 CPUs with firmware version V2.6 and higher.

Description

A programming example shows how to use the instruction manually in SCL programming language.

This section assumes that you are creating and configuring a client interface for OPC UA communication; see "AUTOHOTSPOT". The efficient procedure ensures that most parameters for the client instruction are supplied automatically.

You use the instruction "OPC_UA_MethodReleaseHandleList" in your user program to release numerical references (handles) for methods.

OPC_UA_MethodReleaseHandleList is used to release method handles, see ③ in the figure below.

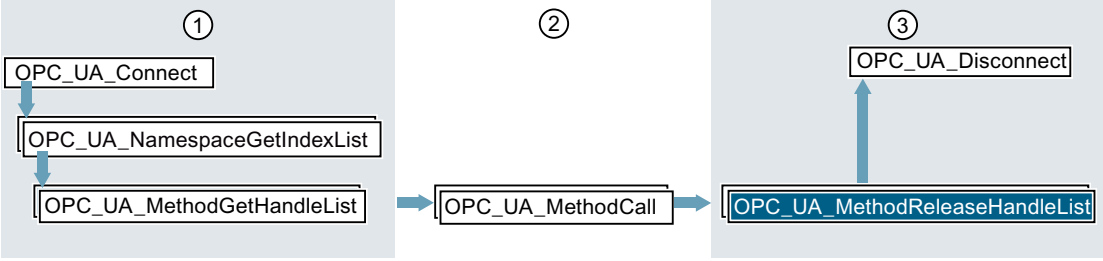

① Instructions for preparation of method calls
② Method calls
③ Instructions for "clean-up" after completed method calls

Parameters for "OPC_UA_MethodReleaseHandleList"

Table 4-83 The parameters of the instruction "OPC_UA_MethodReleaseHandleList"

Parameters	Declaration in area	Data type	Meaning
REQ	Input	BOOL	A rising edge 0 → 1 at the parameter triggers execution of the instruction.
ConnectionHdl	Input	DWORD	Unique identifier for a connection established. You get the handle from the instruction OPC_UA_Connect.
Timeout	Input	TIME	Maximum time for execution of the instruction in milliseconds. See also the explanation for this parameter in AUTOHOTSPOT.
MethodHdlCount	Input	UINT	Number of elements in the MethodHdls parameter
MethodHdls	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the handles for the individual methods (NodeIDs) that are to be released.
StatusList	InOut	VARIANT	Pointer to an array of the type DWORD. The array contains the error codes for the individual methods; see "Error numbers for NodeStatusList" below. For each method, the system specifies whether or not a corresponding handle could be found.
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error numbers for Status" below

Error numbers for Status

The following table shows a summary of error codes for this instruction:

Error code (hex. values)	Name of the error	Explanation
0000_0000	OpcUa_Good	Instruction finished successfully.
8003_0000	OpcUa_BadOutOfMemory	No memory available for the OPC UA client. As the OPC UA client and OPC UA server share a memory area, you should reduce the memory requirement of the server. You have the following options: • Release fewer PLC tags for OPC UA. • Reduce the number of OPC UA clients that are currently connected to the server. • Set up fewer subscriptions.
800A_0000	OpcUa_BadTimeout	A network timeout occurred. Possible causes: • Connection to the OPC UA server is too slow (insufficient capacity) • The network load is too high. • The OPC UA server is not available. Possible remedy: • Check the URL of the OPC UA server • Increase the timeout setting (higher value for the Timeout parameter of the function block OPC_UA_Connect).
800D_0000	OpcUa_BadServerNotConnected	The server is not connected or the connection handle is incorrect or invalid.
800F_0000	OpcUa_BadNothingToDo	Nothing to do: The OPC UA server received an empty list with no instructions from the OPC UA client.
8010_0000	OpcUa_BadTooManyOperations	The request could not be executed because it contained too many instructions.
B080_0100	Simatic_BadType_VariantInput1	The data type of the tag to which the first variant parameter of the instruction is pointing is incorrect.
B080_0200	Simatic_BadType_VariantInput2	The data type of the tag to which the second variant parameter of the instruction is pointing is incorrect.
B080_1100	Simatic_ArrayElements_TooMany	MethodHdlCount > MAX_ELEMENTS_METHOD
B080_C400	Simatic_ClientNotEnabled	The OPC UA client is disabled.
B080_C500	Simatic_NothingToDo	Nothing to do: The instruction is using a list that contains no elements.
C080_C300	Simatic_OutOfResources	The CPU resources have been exhausted. Possible remedy: • Reduce the number of open connections
For more error codes, see AUTOHOTSPOT.		

Error numbers for "StatusList"

The "StatusList" parameter contains an error code for each individual handle.

The following table explains the error codes:

Error code (hex. values)	Name of the error	Explanation
0000_0000	OpcUa_Good	No errors
8034_0000	OpcUa_BadNodeIdUnknown	The method handle transferred is not known.
For more error codes, see AUTOHOTSPOT.		

Requirements

The requirements for the following description are:

- You have created a client interface, see "AUTOHOTSPOT".
- You have created and configured a connection to an OPC UA server, see "AUTOHOTSPOT".

In addition, the following requirements must also be met for the instruction "OPC_UA_MethodReleaseHandleList" (cf diagram above):

- A handle for connection to an OPC UA server is available.
  You get the connection handle from the instruction OPC_UA_Connect.
- A list of method handles that you no longer want to call in your user program.
  You get this handle list from the instruction OPC_UA_MethodGetHandleList.
- Optional (not needed here): Indexes of the namespaces that contain the methods.
  You get the indexes from the instruction OPC_UA_NamespaceGetIndexList.

Function of the instruction

You call the instruction "OPC_UA_MethodReleaseHandleList" to release handles for methods that you no longer require in your program.

How to use a configured connection

1. In the "Project tree" area, select the CPU you want to use as a client.
2. Select the function block that is to execute the client instruction in the "Program blocks" folder.
   In the example, the function block is called "Call_OpenDoor_On_Productionline".
   Selected language: SCL.
3. Use drag-and-drop to move the instruction "OPC_UA_MethodReleaseHandleList" from the folder "Instructions > Communication > OPC UA > OPC UA Client" to the editor.
4. Select the call as multi-instance.
   STEP 7 creates an instance of this instruction and calls it "OPC_UA_MethodReleaseHandleList_Instance".
5. Click "OPC_UA_MethodReleaseHandleList_Instance".
   STEP 7 opens the "Configuration".

6. Under "Select client interface for OPC UA interface" you select the client interface that you want to use for the instruction.
   In the example, the client interface is called "Productionline", which was created for the example plant, see "AUTOHOTSPOT".
7. Click "File access" and select a method list.
   In the example, the method list is called "MethodListOpenDoor".
   STEP 7 automatically supplies most parameters of the instruction with tags, but not the parameters Busy, Done, Error.
8. Create the local tags for the parameters yet to be supplied.
9. Click on "Block parameters" and assign tags manually to the remaining parameters.
   STEP 7 adds the selected tag to the function call.

Calling the instruction (initial call)

The following excerpt from the program "Call_OpenDoor_On_Productionine" shows how to use the instruction "OPC_UA_MethodReleaseHandleList".

You can find the complete example program in the section "Program example for calling server methods".

The example program is divided into multiple sections (Cases) by a CASE instruction.

Case 5 is shown here:

SCL

```
 5:  // case 5, release method handle

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_MethodReleaseHandleList_Instance(
                                                    REQ := #Req,
                                                    ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                    MethodHdlCount :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodCoun
t,
                                                    MethodHdls :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls
,
                                                    Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                    Done => #Done,
                                                    Busy => #Busy,
                                                    Error => #Error,
                                                    Status => #Status,
                                                    StatusList :=
"Productionline_Data"."MethodListOpenDoor".MethodStatusList);
```

```
221  5:  // case 5, release method handle
222
223  IF #Busy = FALSE THEN
224      #Req := TRUE;
225  ELSE
226      #Req := FALSE;
227  END_IF;
228  #OPC_UA_MethodReleaseHandleList_Instance(
229              REQ := #Req,
230              ConnectionHdl := "Productionline_Configuration".Connection.ConnectionHdl,
231              MethodHdlCount := "Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodCount,
232              MethodHdls := "Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls,
233              Timeout := "Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
234              Done => #Done,
235              Busy => #Busy,
236              Error => #Error,
237              Status => #Status,
238              StatusList := "Productionline_Data"."MethodListOpenDoor".MethodStatusList);
239
```

Asynchronous execution

The instruction "OPC_UA_MethodReleaseHandleList" runs asynchronously to the user program and requires multiple program cycles.

You can check the job status with the "Busy", "Done", "Error" and "Status" parameters.

Asynchronous program execution is described in the section "Difference between synchronous and asynchronous instructions (Page 596)".

Explanation of the call of the instruction (initial call)

The figure above shows case 5. This part of the program releases the handle for a server method again.

In the example, the handle for the "OpenDoor" server method is passed to the "MethodHdls" parameter.

If this handle cannot be released, the "StatusList" parameter contains an error code.

Calling the instruction (troubleshooting)

The figure below shows evaluation of the "Done" and "Error" parameters.

SCL

```
            IF #Done = TRUE THEN
                #Mem_Status := #Status;
                IF
"Productionline_Data"."MethodListOpenDoor".MethodStatusList[0] = 0 THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    //In case 100, to set REQ of instruction
"OPC_UA_Disconnect" to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at
MethodGetHandleList';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
               //In case 100, to set REG of instruction "OPC_UA_Disconnect"
to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_MethodReleaseHandleList_Instance(
                                                        MethodHdls :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls
,
                                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                        StatusList :=
"Productionline_Data"."MethodListOpenDoor".MethodStatusList,
                                                       MethodHdlCount :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodCoun
t,
                                                        REQ := FALSE,
                                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);
            END_IF;
```

Explanation of the call of the instruction (troubleshooting)

If an error occurs, the "OPC_UA_MethodReleaseHandleList" instruction at the "Error" output parameter is set to the value TRUE. This sets the value of the "#State" tag to 100. This case is reserved for troubleshooting. The example program also calls the instruction "OPC_UA_MethodReleaseHandleList" to set the REQ parameter to FALSE.

If the output parameter "Done" is TRUE, the "OPC_UA_MethodReleaseHandleList" instruction was executed successfully.

In the example, there is also a check to determine of the handle for the "OpenDoor" server method has been released.

If executed successfully, the value of the "#State" tag is increased by one. This means the next program section - case 6 - is executed in the next cycle.

Case 6 is described in OPC_UA_Disconnect (Page 3723).

Note

Case 6 is for enabling the connection to the OPC UA server. This also releases all other resources that the client program has occupied in the server. If you want to terminate the connection to the server, your user program can execute case 6 immediately and skip case 5.

System data types for OPC UA client instructions

OPC_UA_SessionConnectInfo

OPC_UA_SessionConnectInfo

See the following table for the meaning of the connection information for the "SessionConnectInfo" parameter of the "OPC_UA_Connect" instruction.

Table 4-84 Parameters of system data type "OPC_UA_SessionConnectInfo"

Parameter	S7 data type	Meaning
SessionName	WSTRING[64]	Name of the session, which you can enter here. The parameter can be empty. If you do not enter a name, the OPC UA server assigns one. The session name is used, for example, to identify the connection for diagnostic purposes.
ApplicationName	WSTRING[64]	Name of the OPC UA client. The application name is a component of the client description, which in turn is a component of the session (see SessionName). It is not possible to set a unique ApplicationName for each session. The string **must** be empty, otherwise the error code "Simatic_BadValue_VariantInput2 (0xB080_0200)" is returned. The configured application name is used as the ApplicationName for SIMATIC (CPU parameters in the OPC UA area).
SecurityMsgMode	UDINT	Security process • 0 = Best possible procedure Please note: This setting is not permitted! • 1 = None • 2 = Sign • 3 = Sign&Encrypt

Parameter	S7 data type	Meaning
SecurityPolicy	UDINT	Securityprofile • 0 = Best possible security profile Please note: This setting is not permitted! • 1 = No security profile • 2 = Basic128Rsa15 • 3 = Basic256 • 4 = Basic256Sha256
ServerUri	STRING[254]	Only relevant for connections to an OPC UA server via a gateway server. The current firmware of the CPU does not support gateway servers. This parameter therefore must be supplied with an empty string.
CheckServerCertificate	BOOL	If this bit is set, the client checks the certificate sent by the server against the configured trustworthy certificates. You can access the list of trusted certificates via the properties of the Client CPU > "Protection & Security" area > Certificate Manager. There you can find the "Certificates of the partner devices" table, which you have to fill out. The requirement for filling is that you have imported the required certificates via the global certificate manager (Project tree > Certificate manager > "Trusted Certificates and Root Certification Authorities" tab). If the bit is not set, the client does not check the server certificate.
TransportProfile	UDINT	1 = UATP_UATcp According to the PLCopen specification, only this transport profile is supported.
UserIdentityTokenType	UDINT	For user authentication data, see the explanation below this table.
UserTokenParam1	WSTRING[64]	The meaning depends on UserIdentityTokenType (for example user name).
UserTokenParam2	WSTRING[64]	The meaning depends on UserIdentityTokenType (for example password).

Parameter	S7 data type	Meaning
CertificateID	UDINT	The ID of the certificate that you use for the OPC UA client. You can find the ID in STEP 7 (TIA Portal) under "Global security settings" > "Certificate manager" > "Device certificates" and also in the CPU properties ("Protection & Security" > "Certificate Manager" area). There you can find the "Device certificates" table with the IDs of the certificates. Proceed as follows to generate a certificate for the client: 1. In the TIA Portal, select the properties of the CPU that is used as the client. 2. If a self-signed certificate is sufficient for you, skip this step. If you want to use a certificate signed by a CA, then select the "Use global security settings for certificate manager" option under "Security" > "Certificate manager" > "Global security settings". 3. Navigate to the "Device certificates" table and double-click on an empty field in the "Certificate holder" column. A certificate icon appears at the left edge of the row. 4. Click on the "Certificate holder" field again. A drop-down list opens showing the existing device certificates. 5. Click on the "Add" button. A dialog with setting options for the certificate is displayed. 6. Select "OPC UA Client" or "OPC UA Client & Server" for the use and then "OK".
SessionTimeout	TIME	Maximum time for which a session is maintained after the connection to the OPC UA server has been interrupted (in milliseconds). Default setting: 20000 (20 seconds)
MonitorConnection	TIME	Connection monitoring time (in milliseconds) Time interval for checking a connection over which no data is currently being transferred. Default setting: 5000 (5 seconds)
LocaleIDs	ARRAY[1..5] of STRING[6]	Optional language and regional identifier acc. to RFC 3066. 0 = no or unknown LocaleID.

"UserIdentityTokenType" parameter

The following table shows the relationship between the "UserIdentityTokenType", "UserTokenParam1" and "UserTokenParam2" parameters of the system data type "UASessionConnectInfo":

UserIdentityTokenType	UserTokenParam1	UserTokenParam2	Description
0: Anonymous access	Ignored	Ignored	Guest account, without authentication
1: Access with user name and password	User name	Password	The password is stored in plain text in the corresponding global data block or instance data block and can be read out. Recommendation: Use the SIMATIC User type instead.
42: SIMATIC User	Name of a user configured with STEP 7.	Ignored	For a SIMATIC user, you download the password together with the other user data and the configuration data to the client CPU. This way you protect the password from being accessed from the outside. You must have assigned the function right "User authentication of the OPC UA client" to the user (SIMATIC user) in the project tree (Security settings > Users and roles). When you make the settings for user authentication in the OPC UA client interface, select the option "User (TIA Portal – Security settings)" and enter a previously configured user name.

See also

OPC_UA_Connect: Create connection (Page 3643)

OPC_UA_NodeId

OPC_UA_NodeId

See the following table for the meaning of the "OPC_UA_NodeId" parameters for identifying the destination node in the OPC UA server. "OPC_UA_NodeId" supplies the "NodeIDs" parameter of the "OPC_UA_NodeGetHandleList" instruction.

Table 4-85 Parameters of system data type "OPC_UA_NodeId"

Parameter	S7 data type	Meaning
NamespaceIndex	UINT	Namespace index of the node in the OPC UA server. A node can, for example, be an object or a tag.
Identifier	WSTRING[254]	The designation of the node (object or tag) depends on the identifier type: • Numeric identifier: The node is labeled with a number, for example "12345678" (STRING with converted DWORD). • String identifier: The node is labeled with a name, for example "MyTag". Names are case-sensitive. • For the names of nodes of "GUID" and "Opaque" type, see below.
IdentifierType	UDINT	Type of identifier • 0: Numeric identifier • 1: String identifier • 2: GUID • 3: Opaque

Identifier of IdentifierType "GUID"

Format: {XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}

Example for entering the format of IdentifierType GUID (2) in STEP 7:
`WSTRING#'5ce9dbce-5d79-434c-9ac3-1cfba9a6e92c'`

Identifier of IdentifierType "Opaque"

Opaque identifiers are identifiers with an apparently random sequence of characters without immediately recognizable semantics.

With OPC UA, you can generate an identifier of Opaque type by encoding any byte array of defined length (e.g. ASCII string 'abcd' or the HEX counterpart (0x61626364) Base64. Numerous tools are available for Base64 encoding.

Example for the format of IdentifierType Opaque (3) in STEP 7 after coding the HEX value:
`WSTRING#'YWJjZA=='`

The server displays the uncoded value of the identifier, in this case the HEX value.

See also

OPC_UA_NodeGetHandleList: Get handles for read and write access (Page 3663)

OPC_UA_NodeAdditionalInfo

OPC_UA_NodeAdditionalInfo

Specifies attribute types and index range of nodes for the "NodeAddInfos" parameter of the "OPC_UA_ReadList" and "OPC_UA_WriteList" instructions.

- The data type is used for scalar values and one-dimensional arrays.
  To read a section of a multi-dimensional array (up to six dimensions, (matrix)), use the "OPC_UA_NodeAdditionalInfoExt" data type.
  Read the information provided on Mapping data types.
- The StartIndex must be greater than or equal to "0" and less than or equal to EndIndex.
- Settings for the full reading of reading lists consisting of scalars and arrays:
  If both StartIndex and EndIndex have the value 4_294_967_295 (UDINT_MAX), the server ignores the value for StartIndex and EndIndex and fully reads scalars and arrays, see. Reading array range information with OPC_UA_ReadList (Page 3695).

Table 4-86 Parameters of system data type "OPC_UA_NodeAdditionalInfo"

Parameters	Data type	Meaning
AttributeId	UDINT	Specifies which attribute of a node (tag) is to be read. The IDs of the attributes are defined in the OPC UA enumeration type "UAAttributeId". Examples: • 5: Description (Description) • 13: Value (value) • 14: DataType (Data type) • 15: ValueRank (number of dimensions in the Value attribute) • 16: ArrayDimensions (length of a dimension, number of elements) The default is 13 (Value).
StartIndex	UDINT	When AttributeId = 13: The start index of an array. Values are read starting at this index.
EndIndex	UDINT	When AttributeId = 13: The end index of an array. Values are read to this index.

See also

OPC_UA_NodeAdditionalInfoExt (Page 3765)

OPC_UA_NodeAdditionalInfoExt

OPC_UA_NodeAdditionalInfoExt

Specifies the attribute of the node and the index are for the "NodeAddInfos" parameter ("OPC_UA_ReadList" and "OPC_UA_WriteList" instructions).

Table 4-87 Parameters of system data type "OPC_UA_NodeAdditionalInfoExt"

Parameter	S7 data type	Meaning
AttributeID	UDINT	Specifies which attribute of a node (tag) is to be read or written. The default is 13 (Value).
IndexRangeCount	UDINT	Number of the following index ranges. You do not need to specify all six index ranges.
StartIndex1	UDINT	The index from which values for array dimension 1 are to be read or written.
EndIndex1	UDINT	The index up to which values for array dimension 1 are to be read or written.
StartIndex2	UDINT	The index from which values for array dimension 2 are to be read or written.
EndIndex2	UDINT	The index up to which values for array dimension 2 are to be read or written.
StartIndex3	UDINT	The index from which values for array dimension 3 are to be read or written.
EndIndex3	UDINT	The index up to which values for array dimension 3 are to be read or written.
StartIndex4	UDINT	The index from which values for array dimension 4 are to be read or written.
EndIndex4	UDINT	The index up to which values for array dimension 4 are to be read or written.
StartIndex5	UDINT	The index from which values for array dimension 5 are to be read or written.
EndIndex5	UDINT	The index up to which values for array dimension 5 are to be read or written.
StartIndex6	UDINT	The index from which values for array dimension 6 are to be read or written.
EndIndex6	UDINT	The index up to which values for array dimension 6 are to be read or written.

See also

OPC_UA_NodeAdditionalInfo (Page 3764)

OPC_UA_BrowsePath

OPC_UA_BrowsePath

See the following table for the structure of the system data type "OPC_UA_BrowsePath":

Table 4-88 Parameters of system data type "OPC_UA_BrowsePath"

Name	Data type	Meaning
StartingNode	OPC_UA_NodeId	NodeID of the node (or tag) at which the path starts. Browsing is to be from this node on.
RelativePath	OPC_UA_RelativePath	The relative path for browsing; see OPC_UA_RelativePath.

OPC_UA_RelativePath

OPC_UA_RelativePath

See the following table for the structure of the system data type "OPC_UA_RelativePath":

Table 4-89 Parameters of system data type "OPC_UA_RelativePath"

Name	Data type	Meaning
NoOfElements	UINT	Number of elements in Elements
Elements	Array [4] of OPC_UA_RelativePathElement	Array with up to four elements of the type OPC_UA_RelativePathElement (Page 3766)

OPC_UA_RelativePathElement

OPC_UA_RelativePathElement

See the following table for the structure of the system data type "OPC_UA_RelativePathElement":

Table 4-90 Parameters of system data type "OPC_UA_RelativePathElement"

Name	Data type	Meaning
ReferenceTypeId	OPC_UA_NodeId	The reference type to be followed in this RelativePathElement. ReferenceTypeId is set by default for hierarchical references.
IsInverse	BOOL	If TRUE, the inverse direction is followed. The default is FALSE.

Name	Data type	Meaning
IncludeSubtypes	BOOL	If TRUE, the reference types derived from the type specified under ReferenceTypeId are also followed. The default is TRUE.
TargetName	OPC_UA_QualifiedName	See system data type OPC_UA_QualifiedName

OPC_UA_QualifiedName

OPC_UA_QualifiedName

See the following table for the structure of the system data type "OPC_UA_QualifiedName":

Table 4-91 Parameters of system data type "OPC_UA_QualifiedName"

Name	Data type	Meaning
NamespaceIndex	UINT	The namespace index of the name.
Name	WSTRING[64]	BrowseName of the node (or tag).

Error codes

Siemens error codes

SIMATIC status codes for OPC UA instructions

The table below lists the Siemens error codes for the OPC UA client instructions:

Status code	Meaning
B080_0100	BadType_VariantInput1: The data type of the tag to which the first variant parameter of the instruction is pointing is incorrect. **An example:** For the instruction "OPC_UA_Connect", the parameter "ServerEndpointUrl" is the first variant parameter. The order of VariantInput1 to VariantInput5 corresponds to the order of the variant parameters in the documentation for the respective OPC UA instruction. If the first variant parameter points to a tag that has the incorrect data type, the status code "B080_0100" is returned. The correct data type is listed in the "Parameter" table of the respective instruction. In the example, you can find the data type of the 'Parameter for "OPC_UA_Connect"' table in the section on the instruction "OPC_UA_Connect (Page 3643)".
B080_0200	BadType_VariantInput2. The data type of the tag to which the second variant parameter of the instruction is pointing is incorrect.
B080_0300	BadType_VariantInput3: The data type of the tag to which the third variant parameter of the instruction is pointing is incorrect.
B080_0400	BadType_VariantInput4: The data type of the tag to which the fourth variant parameter of the instruction is pointing is incorrect.
B080_0500	BadType_VariantInput5: The data type of the tag to which the fifth variant parameter of the instruction is pointing is incorrect.
B080_3100	BadNumElements_VariantInput1: Number of array elements in variant 1 is smaller than the number stored in the count parameter of the instruction (e.g. NodeIDCount for OPC_UA_NodeGetHandleList or NamespaceUrisCount for OPC_UA_NamespaceGetIndexList, etc.).
B080_3200	BadNumElements_VariantInput2: Number of array elements in variant 2 is smaller than the number stored in the count parameter of the instruction.
B080_3300	BadNumElements_VariantInput3: Number of array elements in variant 3 is smaller than the number stored in the count parameter of the instruction.

Status code	Meaning
B080_3400	BadNumElements_VariantInput4: Number of array elements in variant 4 is smaller than the number stored in the count parameter of the instruction.
B080_3500	BadNumElements_VariantInput5: Number of array elements in variant 5 is smaller than the number stored in the count parameter of the instruction.
B080_C400	Simatic_ClientNotEnabled: The OPC UA client is disabled.
B080_C500	SimaticNothingToDo: Error during initialization of the client
B080_C600	ClientNotAvailable Error during initialization of the client
B080_0B00	ArrayElements_TooMany: The array has too many elements.
B080_1100	ArrayElements_TooMany: General error code. Occurs when an array has too many elements: NamespaceUrisCount > MAX_ELEMENTS_NAMESPACES NodeIDCount > MAX_ELEMENTS_NODELIST BrowsePathscount > MAX_ELEMENTS_RELATIVEPATH MethodHdlCount > MAX_ELEMENTS_METHODLIST MAX_ELEMENTS_NAMESPACES = 20 MAX_ELEMENTS_NODELIST = 300 MAX_ELEMENTS_METHODLIST = 100 MAX_ELEMENTS_BROWSEPATH = 10
B080_1200	ArrayElements_TooFew
B080_2100	BadValue_VariantInput1: The value of the tag to which the first variant of the instruction is pointing is incorrect.
B080_2200	BadValue_VariantInput2: The value of the tag to which the second variant of the instruction is pointing is incorrect.
B080_2300	BadValue_VariantInput3: The value of the tag to which the third variant of the instruction is pointing is incorrect.
B080_2400	BadValue_VariantInput4: The value of the tag to which the fourth variant of the instruction is pointing is incorrect.
B080_2500	BadValue_VariantInput5: The value of the tag to which the fifth variant of the instruction is pointing is incorrect.

Status code	Meaning
B080_B000	TooManyMethods: Maximum number of server methods or max. number of server method instances exceeded. (Calls of the instructions OPC_UA_ServerMethodPre, OPC_UA_ServerMethodPost): For S7-1510 to S7-1513 ...: Max. 20 For S7-1515, S7-1516...: Max. 50 For S7-1517, S7-1518, S7-1507S...: Max. 100
C080_C300	InsufficientResources: Insufficient resources. Either (a) error upon memory allocation or (b) too many SFB instances

Error codes of the OPC Foundation

Error codes

The following table contains error codes of the OPC Foundation.

The names and explanations of the errors are listed in the original (English) and are selectively supplemented with additional information.

Error codes

Status "Good" (0000_0000 - 3FFF_FFFF)		
Error code (hex)	Name	Meaning
0000_0000	Good	Success, no error.
002D_0000	GoodSubscriptionTransferred	The subscription was transferred to another session.
002E_0000	GoodCompletesAsynchronously	The processing will complete asynchronously.
002F_0000	GoodOverload	Sampling has slowed down due to resource limitations.
0030_0000	GoodClamped	The value written was accepted but was clamped.
0096_0000	GoodLocalOverride	The value has been overridden.
00A2_0000	GoodEntryInserted	The data or event was successfully inserted into the historical database.
00A3_0000	GoodEntryReplaced	The data or event field was successfully replaced in the historical database.
00A5_0000	GoodNoData	No data exists for the requested time range or event filter.
00A6_0000	GoodMoreData	The data or event field was successfully replaced in the historical database.
00A7_0000	GoodCommunicationEvent	The communication layer has raised an event.
00A8_0000	GoodShutdownEvent	The system is shutting down.
00A9_0000	GoodCallAgain	The operation is not finished and needs to be called again.
00AA_0000	GoodNonCriticalTimeout	A non-critical timeout occurred.
00BA_0000	GoodResultsMayBeIncomplete	The server should have followed a reference to a node in a remote server but did not. The result set may be incomplete.

Status "Good" (0000_0000 - 3FFF_FFFF)		
Error code (hex)	Name	Meaning
00D9_0000	GoodDataIgnored	The request pecifies fields which are not valid for the EventType or cannot be saved by the historian.
00DC_0000	GoodEdited	The value does not come from the real source and has been edited by the server.
00DD_0000	GoodPostActionFailed	There was an error in execution of these post-actions.
00E0_0000	GoodDependentValue-Changed	A dependent value has been changed but the change has not been applied to the device.

Status "Uncertain" (4000_0000 - 7FFF_FFFF)		
Error code (hex)	Name	Meaning
406C_0000	UncertainReferenceOutOf-Server	One of the references to follow in the relative path references to a node in the address space in another server.
408F_0000	UncertainNoCommunication-LastUsableValue	Communication to the data source has failed. The variable value is the last value that had a good quality.
4090_0000	UncertainLastUsableValue	Whatever was updating this value has stopped doing so.
4091_0000	UncertainSubstituteValue	The value is an operational value that was manually overwritten.
4092_0000	UncertainInitialValue	The value is an initial value for a variable that normally receives its value from another variable.
4093_0000	UncertainSensorNotAccurate	The value is at one of the sensor limits.
4094_0000	UncertainEngineeringUnit-sExceeded	The value is outside of the range of values defined for this parameter.
4095_0000	UncertainSubNormal	The value is derived from multiple sources and has less than the required number of Good sources.
40A4_0000	UncertainDataSubNormal	The value is derived from multiple values and has less than the required number of Good values.
40BC_0000	UncertainReferenceNotDeleted	The server was not able to delete all target references.
40C0_0000	UncertainNotAllNodesAvailable	The list of references may not be complete because the underlying system is not available.
40DE_0000	UncertainDominantValue-Changed	The related EngineeringUnit has been changed but the Variable Value is still provided based on the previous unit.
40E2_0000	UncertainDependentValue-Changed	A dependent value has been changed but the change has not been applied to the device. The quality of the dominant variable is uncertain.

Status "Bad" (8000_0000 - FFFF_FFFF)		
Error code (hex)	Name	Meaning
8001_0000	BadUnexpectedError	An unexpected error occurred
8002_0000	BadInternalError	An internal error occurred as a result of a programming or configuration error. **Possible cause**: You have write-accessed a single element of a tag of data type DTL. Write access to tags of data type DTL is only possible to the complete structure.

Status "Bad" (8000_0000 - FFFF_FFFF)		
Error code (hex)	**Name**	**Meaning**
8003_0000	BadOutOfMemory	Not enough memory to complete the operation. **Possible cause:** The maximum quantity structure of the OPC UA server has been exceeded. **Solution:** Reduce the number of registered OPC UA elements. Use an OPC UA Server diagnostics tool (such as UaExpert from Unified Automation) to ensure that there are no inactive sessions and subscriptions. As countermeasure you can select corresponding timeouts (Subscription Timeout <= Session Timeout) in the client. See also the description to the error messages OPC_UA_Connect: Create connection (Page 3643).
8004_0000	BadResourceUnavailable	An operating system resource is not available
8005_0000	BadCommunicationError	A low level communication error occurred. **Possible cause:** The server address (ServerEndpointURL) is incorrect or incomplete. See also the explanation for this parameter in OPC_UA_Connect: Create connection (Page 3643).
8006_0000	BadEncodingError	Encoding halted because of invalid data in the objects being serialized.
8007_0000	BadDecodingError	Decoding halted because of invalid data in the stream.
8008_0000	BadEncodingLimitsExceeded	The message encoding/decoding limits imposed by the stack have been exceeded.
8009_0000	BadUnknownResponse	An unrecognized response was received from the server.
80B8_0000	BadRequestTooLarge	The request message size exceeds limits set by the server.
80B9_0000	BadResponseTooLarge	The response message size exceeds limits set by the client.
800A_0000	BadTimeout	The operation timed out. See also error message at OPC_UA_Connect: Create connection (Page 3643).
800B_0000	BadServiceUnsupported	The server does not support the requested service.
800C_0000	BadShutdown	The operation was cancelled because the application is shutting down.
800D_0000	BadServerNotConnected	The operation could not complete because the client is not connected to the server.
800E_0000	BadServerHalted	The server has stopped and cannot process any requests.
800F_0000	BadNothingToDo	There was nothing to do because the client passed a list of operations with no elements.
8010_0000	BadTooManyOperations	The request could not be processed because it specified too many operations. See also error message at OPC_UA_Connect: Create connection (Page 3643).
8011_0000	BadDataTypeIdUnknown	The extension object cannot be (de)serialized because the data type id is not recognized.
8012_0000	BadCertificateInvalid	The certificate provided as a parameter is not valid.
8013_0000	BadSecurityChecksFailed	An error occurred verifying security. The certificate provided as a parameter is not valid.
8014_0000	BadCertificateTimeInvalid	The Certificate has expired or is not yet valid.

Status "Bad" (8000_0000 - FFFF_FFFF)		
Error code (hex)	Name	Meaning
8015_0000	BadCertificateIssuer TimeInvalid	An Issuer Certificate has expired or is not yet valid.
8016_0000	BadCertificateHost NameInvalid	The HostName used to connect to a Server does not match a HostName in the Certificate.
8017_0000	BadCertificateUriInvalid	The URI specified in the Application Description does not match the URI in the Certificate.
8018_0000	BadCertificateUseNotAl-lowed	The Certificate may not be used for the requested operation.
8019_0000	BadCertificateIssuerUse NotAllowed	The Issuer Certificate may not be used for the requested operation.
801A_0000	BadCertificateUntrusted	The Certificate is not trusted.
801B_0000	BadCertificateRevocation Unknown	It was not possible to determine if the Certificate has been revoked.
801C_0000	BadCertificateIssuer RevocationUnknown	It was not possible to determine if the Issuer Certificate has been revoked.
801D_0000	BadCertificateRevoked	The Certificate has been revoked.
801E_0000	BadCertificateIssuerRevoked	The Issuer Certificate has been revoked.
801F_0000	BadUserAccessDenied	User does not have permission to perform the requested operation.
8020_0000	BadIdentityTokenInvalid	The user identity token is not valid.
8021_0000	BadIdentityTokenRejected	The user identity token is valid but the server has rejected it.
8022_0000	BadSecureChannelIdInvalid	The specified secure channel is no longer valid.
8023_0000	BadInvalidTimestamp	The timestamp is outside the range allowed by the server.
8024_0000	BadNonceInvalid	The nonce does appear to be not a random value or it is not the correct length.
8025_0000	BadSessionIdInvalid	The session id is not valid.
8026_0000	BadSessionClosed	The session was closed by the client.
8027_0000	BadSessionNotActivated	The session cannot be used because ActivateSession has not been called.
8028_0000	BadSubscriptionIdInvalid	The subscription id is not valid.
802A_0000	BadRequestHeaderInvalid	The header for the request is missing or invalid.
802B_0000	BadTimestampsTo ReturnInvalid	The timestamps to return parameter is invalid.
802C_0000	BadRequestCancelled ByClient	The request was cancelled by the client.
8031_0000	BadNoCommunication	Communication with the data source is defined, but not established, and there is no last known value available.
8032_0000	BadWaitingForInitialData	Waiting for the server to obtain values from the underlying data source.
8033_0000	BadNodeIdInvalid	The syntax of the node id is not valid.
8034_0000	BadNodeIdUnknown	The node id refers to a node that does not exist in the server address space. The NamespaceIndex of the specified NodeId may not exist. If the error code is displayed when the instruction "OPC_UA_MethodCall" is called: Another possible cause is the use of invalid ObjectIDs in the preceding call of OPC_UA_MethodGetHandleList. The instruction "OPC_UA_MethodGe-tHandleList" does not evaluate the ObjectIds.
8035_0000	BadAttributeIdInvalid	The attribute is not supported for the specified Node.
8036_0000	BadIndexRangeInvalid	The syntax of the index range parameter is invalid.

Status "Bad" (8000_0000 - FFFF_FFFF)		
Error code (hex)	**Name**	**Meaning**
8037_0000	BadIndexRangeNoData	No data exists within the range of indexes specified.
8038_0000	BadDataEncodingInvalid	The data encoding is invalid.
8039_0000	BadDataEncoding Unsupported	The server does not support the requested data encoding for the node.
803A_0000	BadNotReadable	The access level does not allow reading or subscribing to the Node.
803B_0000	BadNotWritable	The access level does not allow writing to the Node.
803C_0000	BadOutOfRange	The value was out of range. Possible remedy for arrays: Check whether you can access elements of an array that are undefined.
803D_0000	BadNotSupported	The requested operation is not supported. **Possible cause:** The server address (ServerEndpointURL) is incorrect or incomplete.
803E_0000	BadNotFound	A requested item was not found or a search operation ended without success.
803F_0000	BadObjectDeleted	The object cannot be used because it has been deleted.
8040_0000	BadNotImplemented	Requested operation is not implemented.
8041_0000	BadMonitoringModeInvalid	The monitoring mode is invalid.
8042_0000	BadMonitoredItemIdInvalid	The monitoring item id does not refer to a valid monitored item.
8043_0000	BadMonitoredItem FilterInvalid	The monitored item filter parameter is not valid.
8044_0000	BadMonitoredItem FilterUnsupported	The server does not support the requested monitored item filter.
8045_0000	BadFilterNotAllowed	A monitoring filter cannot be used in combination with the attribute specified.
8046_0000	BadStructureMissing	A mandatory structured parameter was missing or null.
8047_0000	BadEventFilterInvalid	The event filter is not valid.
8048_0000	BadContentFilterInvalid	The content filter is not valid.
8049_0000	BadFilterOperandInvalid	The operand used in a content filter is not valid.
804A_0000	BadContinuation PointInvalid	The continuation point provide is longer valid.
804B_0000	BadNoContinuationPoints	The operation could not be processed because all continuation points have been allocated.
804C_0000	BadReferenceTypeIdInvalid	The operation could not be processed because all continuation points have been allocated.
804D_0000	BadBrowseDirectionInvalid	The browse direction is not valid.
804E_0000	BadNodeNotInView	The node is not part of the view.
804F_0000	BadServerUriInvalid	The ServerUri is not a valid URI.
8050_0000	BadServerNameMissing	No ServerName was specified
8051_0000	BadDiscoveryUrlMissing	No DiscoveryUrl was specified.
8052_0000	BadSempahoreFileMissing	The semaphore file specified by the client is not valid.
8053_0000	BadRequestTypeInvalid	The security token request type is not valid.
8054_0000	BadSecurityModeRejected	The security mode does not meet the requirements set by the Server. Server is setting higher security requirements. Possible remedy: Use a higher security setting for the connection to the server.
8055_0000	BadSecurityPolicyRejected	The security policy does not meet the requirements set by the Server.

Status "Bad" (8000_0000 - FFFF_FFFF)		
Error code (hex)	Name	Meaning
8056_0000	BadTooManySessions	The server has reached its maximum number of sessions.
8057_0000	BadUserSignatureInvalid	The user token signature is missing or invalid.
8058_0000	BadApplicationSignature Invalid	The signature generated with the client certificate is missing or invalid.
8059_0000	BadNoValidCertificates	The client did not provide at least one software certificate that is valid and meets the profile requirements for the server.
805A_0000	BadRequestCancelled ByRequest	The request was cancelled by the client with the Cancel service.
805B_0000	BadParentNodeIdInvalid	The parent node id does not to refer to a valid node.
805C_0000	BadReferenceNotAllowed	The reference could not be created because it violates constraints imposed by the data model.
805D_0000	BadNodeIdRejected	The requested node id was reject because it was either invalid or server does not allow node ids to be specified by the client.
805E_0000	BadNodeIdExists	The requested node id is already used by another node.
805F_0000	BadNodeClassInvalid	The node class is not valid.
8060_0000	BadBrowseNameInvalid	The browse name is invalid.
8061_0000	BadBrowseNameDuplicated	The browse name is not unique among nodes that share the same relationship with the parent.
8062_0000	BadNodeAttributesInvalid	The node attributes are not valid for the node class.
8063_0000	BadTypeDefinitionInvalid	The type definition node id does not reference an appropriate type node.
8064_0000	BadSourceNodeIdInvalid	The source node id does not reference a valid node.
8065_0000	BadTargetNodeIdInvalid	The target node id does not reference a valid node.
8066_0000	BadDuplicateReference NotAllowed	The reference type between the nodes is already defined.
8067_0000	BadInvalidSelfReference	The server does not allow this type of selfreference on this node.
8068_0000	BadReferenceLocalOnly	The reference type is not valid for a reference to a remote server.
8069_0000	BadNoDeleteRights	The server will not allow the node to be deleted.
806A_0000	BadServerIndexInvalid	The server index is not valid.
806B_0000	BadViewIdUnknown	The view id does not refer to a valid view node.
806D_0000	BadTooManyMatches	The requested operation has too many matches to return.
806E_0000	BadQueryTooComplex	The requested operation requires too many resources in the server.
806F_0000	BadNoMatch	The requested operation has no match to return.
8070_0000	BadMaxAgeInvalid	The max age parameter is invalid.
8071_0000	BadHistoryOperationInvalid	The history details parameter is not valid.
8072_0000	BadHistoryOperation Unsupported	The server does not support the requested operation.
8073_0000	BadWriteNotSupported	The server not does support writing the combination of value, status and time-stamps provided.
8074_0000	BadTypeMismatch	The value supplied for the attribute is not of the same type as the attribute's value.
8075_0000	BadMethodInvalid	The method id does not refer to a method for the specified object.
8076_0000	BadArgumentsMissing	The client did not specify all of the input arguments for the method.

Status "Bad" (8000_0000 - FFFF_FFFF)		
Error code (hex)	**Name**	**Meaning**
8077_0000	BadTooManySubscriptions	The server has reached its maximum number of subscriptions. **Possible cause:** Maximum number of subscriptions is exceeded **Solution:** Create a subscription in a new session.
8078_0000	BadTooManyPublish Requests	The server has reached the maximum number of queued publish requests.
8079_0000	BadNoSubscription	There is no subscription available for this session.
807A_0000	BadSequenceNumber Unknown	The sequence number is unknown to the server.
807B_0000	BadMessageNotAvailable	The requested notification message is no longer available.
807C_0000	BadInsufficientClientProfile	The Client of the current Session does not support one or more Profiles that are necessary for the Subscription.
80BF_0000	BadStateNotActive	The sub-state machine is not currently active.
807D_0000	BadTcpServerTooBusy	The server cannot process the request because it is too busy.
807E_0000	BadTcpMessageTypeInvalid	The type of the message specified in the header invalid.
807F_0000	BadTcpSecureChannel Unknown	The SecureChannelId and/or TokenId are not currently in use.
8080_0000	BadTcpMessageTooLarge	The size of the message specified in the header is too large.
8081_0000	BadTcpNotEnough Resources	There are not enough resources to process the request.
8082_0000	BadTcpInternalError	An internal error occurred.
8083_0000	BadTcpEndpointUrlInvalid	The Server does not recognize the QueryString specified.
8084_0000	BadRequestInterrupted	The request could not be sent because of a network interruption.
8085_0000	BadRequestTimeout	Timeout occurred while processing the request.
8086_0000	BadSecureChannelClosed	The secure channel has been closed.
8087_0000	BadSecureChannelToken Unknown	The token has expired or is not recognized.
8088_0000	BadSequenceNumberInvalid	The sequence number is not valid.
8089_0000	BadConfigurationError	There is a problem with the configuration that affects the usefulness of the value.
808A_0000	BadNotConnected	The variable should receive its value from another variable, but has never been configured to do so.
808B_0000	BadDeviceFailure	There has been a failure in the device/data source that generates the value that has affected the value.
808C_0000	BadSensorFailure	There has been a failure in the sensor from which the value is derived by the device/data source.
808D_0000	BadOutOfService	The source of the data is not operational.
808E_0000	BadDeadbandFilterInvalid	The dead band filter is not valid.
8097_0000	BadRefreshInProgress	This Condition refresh failed, a Condition refresh operation is already in progress.
8098_0000	BadConditionAlreadyDisabled	This condition has already been disabled.
8099_0000	BadConditionDisabled	Property not available, this condition is disabled.

Status "Bad" (8000_0000 - FFFF_FFFF)		
Error code (hex)	Name	Meaning
809A_0000	BadEventIdUnknown	The specified event id is not recognized.
809B_0000	BadNoData	No data exists for the requested time range or event filter.
809D_0000	BadDataLost	Data is missing due to collection started/stopped/lost.
809E_0000	BadDataUnavailable	Expected data is unavailable for the requested time range due to an unmounted volume an off-line archive or tape or similar reason for temporary unavailability.
809F_0000	BadEntryExists	The data or event was not successfully inserted because a matching entry exists.
80A0_0000	BadNoEntryExists	The data or event was not successfully updated because no matching entry exists.
80A1_0000	BadTimestampNotSupported	The client requested history using a timestamp format the server does not support (i. e. requested ServerTimestamp when server only supports SourceTimestamp).
80AB_0000	BadInvalidArgument	One or more arguments are invalid. **Possible cause:** The server address (ServerEndpointURL) is incorrect or incomplete.
80AC_0000	BadConnectionRejected	Could not establish a network connection to remote server.
80AD_0000	BadDisconnect	The server has disconnected from the client.
80AE_0000	BadConnectionClosed	The network connection has been closed.
80AF_0000	BadInvalidState	The operation cannot be completed because the object is closed uninitialized or in some other invalid state.
80B0_0000	BadEndOfStream	Cannot move beyond end of the stream.
80B1_0000	BadNoDataAvailable	No data is currently available for reading from a non-blocking stream.
80B2_0000	BadWaitingForResponse	The asynchronous operation is waiting for a response.
80B3_0000	BadOperationAbandoned	The asynchronous operation was abandoned by the caller.
80B4_0000	BadExpectedStreamToBlock	The stream did not return all data requested (possibly because it is a non-blocking stream).
80B5_0000	BadWouldBlock	Non-blocking behavior is required and the operation would block.
80B6_0000	BadSyntaxError	A value had an invalid syntax.
80B7_0000	BadMaxConnections Reached	The operation could not be finished because all available connections are in use.
80BB_0000	BadEventNot Acknowledgeable	The event cannot be acknowledged.
80BD_0000	BadInvalidTimestamp Argument	The defined timestamp to return was invalid.
80BE_0000	BadProtocolVersion Unsupported	The applications do not have compatible protocol versions.
80C1_0000	BadFilterOperatorInvalid	An unrecognized operator was provided in a filter.
80C2_0000	BadFilterOperator Unsupported	A valid operator was provided, but the server does not provide support for this filter operator.
80C3_0000	BadFilterOperandCount Mismatch	The number of operands provided for the filter operator was less than expected for the operand provided.
80C4_0000	BadFilterElementInvalid	The referenced element is not a valid element in the content filter.
80C5_0000	BadFilterLiteralInvalid	The referenced literal is not a valid value.

Status "Bad" (8000_0000 - FFFF_FFFF)		
Error code (hex)	**Name**	**Meaning**
80C6_0000	BadIdentityChangeNotSup-ported	The Server does not support changing the user identity assigned to the session.
80C8_0000	BadNotTypeDefinition	The provided Nodeid was not a type definition nodeid.
80C9_0000	BadViewTimestampInvalid	The view timestamp is not available or not supported.
80CA_0000	BadViewParameterMismatch	The view parameters are not consistent with each other.
80CB_0000	BadViewVersionInvalid	The view version is not available or not supported.
80CC_0000	BadConditionAlready Enabled	This condition has already been enabled.
80CD_0000	BadDialogNotActive	The dialog condition is not active.
80CE_0000	BadDialogResponseInvalid	The response is not valid for the dialog.
80CF_0000	BadConditionBranch AlreadyAcked	The condition branch has already been acknowledged.
80D0_0000	BadConditionBranch AlreadyConfirmed	The condition branch has already been confirmed.
80D1_0000	BadConditionAlready-Shelved	The condition has already been shelved.
80D2_0000	BadConditionNotShelved	The condition is not currently shelved.
80D3_0000	BadShelvingTimeOutO-fRange	The shelving time not within an acceptable range.
80D4_0000	BadAggregateListMismatch	The requested number of Aggregates does not match the requested number of NodeIds.
80D5_0000	BadAggregateNotSupported	The requested Aggregate is not support by the server.
80D6_0000	BadAggregateInvalidInputs	The aggregate value could not be derived due to invalid data inputs.
80DB_0000	BadTooManyMonitoredItems	The request could not be processed because there are too many monitored items in the subscription. **Possible cause:** The maximum number of supervised elements has been reached in a subscription. **Solution:** To register additional elements, you must create a new subscription. For projects as of TIA Portal V15 and higher, you can also set the maximum number of supervised elements in the CPU configuration. **Additional possible cause:** Request with too many elements **Solution:** Note the maximum values specified on the server in the "ServerCapabilities" object.
80D7_0000	BadBoundNotFound	No data found to provide upper or lower bound value.
80D8_0000	BadBoundNotSupported	The server cannot retrieve a bound for the variable.
80DA_0000	BadAggregateConfiguration Rejected	The aggregate configuration is not valid for specified node.
80E1_0000	BadDominantValueChanged	The related EngineeringUnit has been changed but this change has not been applied to the device. The Variable Value is still dependent on the previous unit but its status is currently bad.

Status "Bad" (8000_0000 - FFFF_FFFF)		
Error code (hex)	Name	Meaning
80E3_0000	BadDependentValue-Changed	A dependent value has been changed but the change has not been applied to the device. The quality of the dominant variable is bad.
80E4_0000	BadRequestNotAllowed	The request was rejected by the server because it did not meet the criteria set by the server.
80E5_0000	BadTooManyArguments	Too many arguments were provided.
80E6_0000	BadSecurityModeInsufficient	The operation is not permitted over the current secure channel.
810D_0000	BadCertificateChainIncom-plete	The certificate chain is incomplete.
810E_0000	BadLicenseExpired	The server requires a license to operate in general or to perform a service or operation, but existing license is expired.
810F_0000	BadLicenseLimitsExceeded	The server has limits on number of allowed operations / objects, based on installed licenses, and these limits where exceeded.
8110_0000	BadLicenseNotAvailable	The server does not have a license which is required to operate in general or to perform a service or operation.
8111_0000	BadNotExecutable	The executable attribute does not allow the execution of the method.
8112_0000	BadNumericOverflow	The number was not accepted because of a numeric overflow.
8113_0000	BadRequestNotComplete	The request has not been processed by the server yet.

PLCopen error codes

Error codes

The following table contains PLCopen error codes.

The names and explanations of the errors are listed in the original. They are only available in English.

PLCopen error codes

ErrorID (hex)	Name	Meaning
A000_0001	PLCopenUA_Bad_FW_ PermanentError	Internal, permanent error.
A000_0002	PLCopenUA_Bad_FW_ TempError	Temp. Error; FB could retry to reach FW.
A000_0003	PLCopenUA_Bad_ ConnectionError	Connection could not be established.
A000_0004	PLCopenUA_Bad_ HostNotFound	The requested hostname could not be found.
A000_0005	PLCopenUA_Bad_ AlreadyConnected	Connection was already established.
A000_0006	PLCopenUA_Bad_ SecurityFailed	Connection failed due to security setup.
A000_0007	PLCopenUA_Bad_ Suspended	Connection is suspended.

ErrorID (hex)	Name	Meaning
A000_0008	PLCopenUA_Bad_ ConnectionInvalidHdl	Provided ConnectionHdl is not known.
A000_0009	PLCopenUA_Bad_ NSNotFound	A namespace with the requested name cannot be found on server.
A000_000A	PLCopenUA_Bad_ ResultTooLong	Target PLC variable is too short for retrieved data.
A000_000B	PLCopenUA_Bad_ InvalidType	Invalid or unsupported Type.
A000_000C	PLCopenUA_Bad_ NodeInvalidHdl	Provided NodeHdl is not known.
A000_000D	PLCopenUA_Bad_ MethodInvalidHdl	Provided MethodHdl is not known.
A000_000E	PLCopenUA_Bad_ ReadFailed	Read failed for unknown reason.
A000_000F	PLCopenUA_Bad_ WriteFailed	Write failed for unknown reason.
A000_0010	PLCopenUA_Bad_ CallFailed	Method Call failed for unknown reason.
A000_0011	PLCopenUA_Bad_ InParamFailed	Method Call Input parameter conversion failed.
A000_0012	PLCopenUA_Bad_ OutParamFailed	Method Call Output parameter conversion failed. ATTENTION: this means the MethodCall was executed successfully but the returned values could not be converted.
A000_0013	PLCopenUA_Bad_ SubscriptionInvalidHdl	Provided SubscriptionHdl is not known.
A000_0014	PLCopenUA_Bad_ MonitoredItemInvalidHdl	Provided MonitoredItemHdl is not known.

Example programs for OPC UA clients

Preliminary remarks

The example function blocks "Call_OpenDoor_On_Productionline", "ReadFromProductionline" and "WriteToProductionline" only illustrate the individual functions and are not designed for common use. They can therefore only run individually.

Each FB establishes its own connection with the OPC_UA_Connect instruction and accesses the same client interface and thus the DB "Productionline_Configuration".

The connection handles of the three connections are stored in the same tag when used together and overwrite each other.

Take this fact into account when using all the functions described in your project.

Example program for OPC_UA_ConnectionGetStatus

Example program for analyzing connections

This section includes the complete program code for the example program "Analyze_Connection".

The example shows you how a user program uses the instruction "OPC_UA_ConnectionGetStatus" to get information about a connection to an OPC UA server.

The description of the instruction "OPC_UA_ConnectionGetStatus" can be found here (Page 3729).

Program structure

The program operates as OPC UA client and comprises the following steps:

1. Establishing a connection to an OPC UA server.
2. Analyzing the connection
3. Terminating the connection to the OPC UA server.

You start the program with a positive edge at "Input_REQ".

Declaration

The following figure shows the declaration of local tags of a function block that uses the instruction "OPC_UA_ConnectionGetStatus":

Analyze_Connection

Name	Data type	Default value
Input		
Input_REQ	Bool	false
Output		
Output_Busy	Bool	false
Output_Error	Bool	false
Output_Done	Bool	false
Output_Status	DWord	16#0
Output_Error_Message	WString	WSTRING#''
ConnectionStatus	UDInt	0
ServerState	UDInt	0
ServiceLevel	Byte	16#0
InOut		
<Hinzufügen>		
Static		
Mem_Input_REQ	Bool	false
Run	Bool	false
Started	Bool	false
State	Int	0
Mem_Status	DWord	16#0
Req	Bool	false
Busy	Bool	false
Error	Bool	false
Done	Bool	false
Status	DWord	16#0
SetServerEndpoint	Bool	TRUE
ServerEndpointUrl	WString	WSTRING#''
OPC_UA_SessionConnectInfo_Instance	OPC_UA_SessionConnectInfo	
ConnectionHdl	DWord	16#0
OPC_UA_ConnectionGetStatus_Instance	OPC_UA_ConnectionGetStatus	
OPC_UA_Disconnect_Instance	OPC_UA_Disconnect	
ResetREQ_For_OPC_UA_Disconnect	Bool	false
OPC_UA_Connect_Instance	OPC_UA_Connect	

Program

SCL

```
IF #Run = FALSE THEN
    #Started := #Input_REQ AND NOT #Mem_Input_REQ;
    #Mem_Input_REQ := #Input_REQ;
    IF #Started THEN
        #Output_Busy := TRUE;
        #Output_Done := FALSE;
        #Output_Error := FALSE;
        #State := 1;
        #Run := TRUE;
        #ResetREQ_For_OPC_UA_Disconnect := TRUE;
        #SetServerEndpoint := TRUE;
    END_IF;
END_IF;

IF #Run = TRUE THEN


     CASE #State OF

        1:  // case 1, connect to server

             IF #SetServerEndpoint = TRUE THEN
                #SetServerEndpoint := FALSE;

                //set ServerSendpoint
                #ServerEndpointUrl := WSTRING#'opc.tcp://192.168.1.1:4840';

                //setting for SecurityMsgMode
                //1 = None
                //2 = Sign
                //3 = Sign & Encrypt
                #OPC_UA_SessionConnectInfo_Instance.SecurityMsgMode := 3;

                //setting for SecurityPolicy
                //1 = None
                //2 = Basic128Rsa15
                //3 = Basic256
                //4 = Basic256Sha256
                #OPC_UA_SessionConnectInfo_Instance.SecurityPolicy := 4;

                //set TransportProfile to 1
                #OPC_UA_SessionConnectInfo_Instance.TransportProfile := 1;

                //we are using our Client-Certificate No. 10
                //because No. 10 is the latest Certificate we created for our client
application
                //please look into Certificate manager for the number of your Client
certificate
                //Certificate manager is located at
                // "Project tree > Security settings > Security features"
                #OPC_UA_SessionConnectInfo_Instance.CertificateID := 10;

            END_IF;
```

SCL

```
            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_Connect_Instance(ServerEndpointUrl := #ServerEndpointUrl,
                                     SessionConnectInfo := #OPC_UA_SessionConnectInfo_Instance,
                                         REQ := #Req,
                                         Error => #Error,
                                         Done => #Done,
                                         Busy => #Busy,
                                         ConnectionHdl => #ConnectionHdl,
                                         Status => #Status);
            IF #Done = TRUE THEN
                #State := #State + 1;
                #ResetREQ_For_OPC_UA_Disconnect := TRUE;
            END_IF;

            IF #Error = TRUE THEN
                // Did we get a connection handle?
                IF #ConnectionHdl <> 0 THEN
                    // We have to release all resources in the server and disconnect
                    #State := 100;
                ELSE
                    #State := 99;
                END_IF;
                #Mem_Status := #Status;
                #OPC_UA_Connect_Instance(ServerEndpointUrl := #ServerEndpointUrl,
                                         SessionConnectInfo :=
#OPC_UA_SessionConnectInfo_Instance,
                                         REQ := FALSE);
                #ResetREQ_For_OPC_UA_Disconnect := TRUE;
            END_IF;


        2:  // case 2, analyze connection

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_ConnectionGetStatus_Instance(ConnectionHdl := #ConnectionHdl,
                                                 ConnectionStatus => #ConnectionStatus,
                                                 ServerState => #ServerState,
                                                 ServiceLevel => #ServiceLevel,
                                                 Timeout := T#6S,
                                                 Done => #Done,
                                                 Error => #Error,
                                                 Status => #Status,
                                                 Busy => #Busy,
                                                 REQ := #Req);
            IF #Done = TRUE THEN
                #State := #State + 1;
```

SCL

```
        END_IF;

        IF #Error = TRUE THEN
            #State := 100;
            #Mem_Status := #Status;
            #OPC_UA_ConnectionGetStatus_Instance(REQ := FALSE,
                                                 ConnectionHdl := #ConnectionHdl);
        END_IF;


    3:  // case 3, disconnect from server
        IF #Busy = FALSE THEN
            #Req := TRUE;
        ELSE
            #Req := FALSE;
        END_IF;

        #OPC_UA_Disconnect_Instance(ConnectionHdl := #ConnectionHdl,
                                    REQ := #Req,
                                    Timeout := T#6S,
                                    Done => #Done,
                                    Error => #Error,
                                    Busy => #Busy,
                                    Status => #Status);

        IF #Done = TRUE THEN
            #State := #State + 1;
        END_IF;

        IF #Error = TRUE THEN
            #State := 100;
            #Mem_Status := #Status;
            #OPC_UA_Disconnect_Instance(ConnectionHdl := #ConnectionHdl,
                                        REQ := FALSE);
        END_IF;


     4: //case = 4, function block has run successfully

        #Output_Done := TRUE;
        #Output_Busy := FALSE;
        #State := 0;
        #Mem_Input_REQ := FALSE;
        #Run := FALSE;
        #Started := FALSE;


    99: // ERROR handling

        #Output_Error := TRUE;
        #Output_Status := #Mem_Status;
        #State := 0;
        #Output_Busy := FALSE;
        #Run := FALSE;
        #Mem_Input_REQ := FALSE;
        #Started := FALSE;
```

SCL

```
    100:    // ERROR handling, disconnect from server to release resources (handles)

            IF #ResetREQ_For_OPC_UA_Disconnect = TRUE THEN
                #ResetREQ_For_OPC_UA_Disconnect := FALSE;

            //set REQ to FALSE
                #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                            REQ := FALSE);
            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        REQ := #Req,
                                        Timeout := T#6S,
                                        Done => #Done,
                                        Busy => #Busy,
                                        Error => #Error,
                                        Status => #Status);

            IF #Done = TRUE THEN
                #Output_Error := TRUE;
                #Output_Status := #Mem_Status;
                #State := 0;
                #Output_Busy := FALSE;
                #Run := FALSE;
                #Mem_Input_REQ := FALSE;
                #Started := FALSE;
            END_IF;

            IF #Error = TRUE THEN
                #Output_Error_Message := WString#'Error handling: Resources cannot be
released!';
                #Output_Error := TRUE;
                #Output_Status := #Mem_Status;
                #State := 0;
                #Output_Busy := FALSE;
                #Run := FALSE;
                #Mem_Input_REQ := FALSE;
                #Started := FALSE;
            END_IF;


    END_CASE;

END_IF;
```

SCL

Program example for reading PLC tags

Example program for reading PLC tags

This section includes the complete program code for the example program "ReadFromProductionline".

The example shows you how a user program reads the values of PLC tags.

Program structure

The program operates as OPC UA client and comprises the following steps:

1. Establishing a connection to the OPC UA server of the CPU from which the values are to be read.
2. Reading the values.
3. Terminating the connection to the OPC UA server.

You start the program with a positive edge at "Input_REQ".

Declaration

The following figure shows the declaration of local tags for the function block "ReadFromProductionline".

Input	
Input_REQ	Bool
Output	
Output_Busy	Bool
Output_Error	Bool
Output_Done	Bool
Output_Status	DWord
Output_Error_Message	WString
InOut	
<Hinzufügen>	
Static	
Mem_Input_REQ	Bool
Run	Bool
Started	Bool
State	Int
Mem_Status	DWord
Busy	Bool
Error	Bool
Done	Bool
Status	DWord
Req	Bool
Set_REQ_To_FALSE	Bool
i	Int
▸ OPC_UA_Connect_Instance	OPC_UA_Connect
▸ OPC_UA_NamespaceGetIndexList_Instance	OPC_UA_NamespaceGetIndexList
▸ OPC_UA_NodeGetHandleList_Instance	OPC_UA_NodeGetHandleList
▸ OPC_UA_ReadList_Instance	OPC_UA_ReadList
▸ OPC_UA_NodeReleaseHandleList_Instance	OPC_UA_NodeReleaseHandleList
▸ OPC_UA_Disconnect_Instance	OPC_UA_Disconnect

The example program also uses the following data blocks:

- Productionline_Configuration
- Productionline_Data

These data blocks are automatically created by STEP 7 (TIA Portal) because the example program uses the parameter assignment for OPC UA.

Program

SCL

```
IF #Run = FALSE THEN
    #Started := #Input_REQ AND NOT #Mem_Input_REQ;
    #Mem_Input_REQ := #Input_REQ;
    IF #Started THEN
        #Output_Busy := TRUE;
        #Output_Done := FALSE;
        #Output_Error := FALSE;
        #Output_Status := 0;
        #State := 1;
        #Run := TRUE;
        #Set_REQ_To_FALSE := TRUE;      //In case 100, to set REQ of instruction
"OPC_UA_Disconnect" to FALSE
    END_IF;
END_IF;

IF #Run = TRUE THEN


    CASE #State OF
        1:  // case 1, connect to server
            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_Connect_Instance(ServerEndpointUrl :=
"Productionline_Configuration".Connection.ServerEndpointUrl,
                                     SessionConnectInfo :=
"Productionline_Configuration".Connection.ConnectInfo,
                                     ConnectionHdl =>
"Productionline_Configuration".Connection.ConnectionHdl,
                                     Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                     REQ := #Req,
                                     Done => #Done,
                                     Busy => #Busy,
                                     Error => #Error,
                                     Status => #Status);


            IF #Done = TRUE THEN

                #State := #State + 1;
            END_IF;

            IF #Error = TRUE THEN
                // Did we get a connection handle?
                IF "Productionline_Configuration".Connection.ConnectionHdl <> 0 THEN
                    // We have to release all resources in the server and disconnect
                    #State := 100;
                    //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                ELSE
                    #State := 99;
```

SCL

```
                END_IF;

                #Mem_Status := #Status;
                // set parameter REQ of OPC_UA_Connect to FALSE
                #OPC_UA_Connect_Instance(ServerEndpointUrl :=
"Productionline_Configuration".Connection.ServerEndpointUrl,
                                         SessionConnectInfo :=
"Productionline_Configuration".Connection.ConnectInfo,
                                         REQ := FALSE,
                                         ConnectionHdl =>
"Productionline_Configuration".Connection.ConnectionHdl,
                                         Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);
            END_IF;

        2:  // case 2, get index for namespace

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_NamespaceGetIndexList_Instance(NamespaceUris :=
"Productionline_Configuration".Namespaces.NamespaceURIs,
                                                   ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                   NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes,
                                                   StatusList :=
"Productionline_Configuration".Namespaces.NamespaceStatusList,
                                                   NamespaceUrisCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                   Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                   REQ := #Req,
                                                   Done => #Done,
                                                   Busy => #Busy,
                                                   Error => #Error,
                                                   Status => #Status);


            IF #Done = TRUE THEN
               IF "Productionline_Configuration".Namespaces.NamespaceStatusList[0] = 0 THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at NamespaceGetIndexList';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
```

SCL

```
                #Mem_Status := #Status;
                #OPC_UA_NamespaceGetIndexList_Instance(NamespaceUris :=
"Productionline_Configuration".Namespaces.NamespaceURIs,
                                  StatusList :=
"Productionline_Configuration".Namespaces.NamespaceStatusList,
                                  NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes,
                                  EQ := FALSE);
            END_IF;

        3:  // case 3, get handles for nodes

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_NodeGetHandleList_Instance(NodeIDs :=
"Productionline_Configuration".ReadLists."ReadListProduct".Nodes,
                                               NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                               NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                               REQ := #Req,
                                               ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                               Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                               NodeIDCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount,
                                               NamespaceIndexCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                               NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes,
                                               Done => #Done,
                                               Error => #Error,
                                               Busy => #Busy,
                                               Status => #Status);

            IF #Done = TRUE THEN
                IF "Productionline_Data"."ReadListProduct".NodeStatusList[0] = 0 AND
"Productionline_Data"."ReadListProduct".NodeStatusList[1] = 0
                THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at NodeGetHandleList';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
```

SCL

```
                #OPC_UA_NodeGetHandleList_Instance(NodeIDs :=
"Productionline_Configuration".ReadLists."ReadListProduct".Nodes,
                                                   NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                                   NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                                   REQ := FALSE,
                                                   ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                   Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                   NodeIDCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount,
                                                   NamespaceIndexCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                   NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes);
            END_IF;


        4:  // case 4, read from nodes

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_ReadList_Instance(
                                      NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                      NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                      Variable :=
"Productionline_Data"."ReadListProduct".Variable,
                                      ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                      Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                      NodeHdlCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount,
                                      TimeStamps :=
"Productionline_Data"."ReadListProduct".TimeStamps,
                                      REQ := #Req,
                                      Done => #Done,
                                      Busy => #Busy,
                                      Error => #Error,
                                      Status => #Status);


            IF #Done = TRUE THEN
                FOR #i := 0 TO
UINT_TO_INT("Productionline_Configuration".ReadLists."ReadListProduct".NodeCount) - 1 DO
                  IF NOT ("Productionline_Data"."ReadListProduct".NodeStatusList[#i] = 0)
THEN
                       #Output_Error_Message := CONCAT_WSTRING(IN1 := WSTRING#'Error at
Readlist "ReadListProduct", Index: ',
```

SCL

```
                                                                    IN2 := INT_TO_WSTRING(#i));
                    END_IF;
                END_FOR;
                #State := #State + 1;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_ReadList_Instance(
                                          NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                          NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                          Variable :=
"Productionline_Data"."ReadListProduct".Variable,
                                          REQ := FALSE,
                                          ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                          Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                          NodeHdlCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount,
                                          TimeStamps :=
"Productionline_Data"."ReadListProduct".TimeStamps);
            END_IF;


        5:  // case 5, release node handles

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_NodeReleaseHandleList_Instance(
                                                   NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                                   NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                                   REQ := #Req,
                                                   ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                   Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                   NodeHdlCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount,
                                                   Done => #Done,
                                                   Error => #Error,
                                                   Busy => #Busy,
                                                   Status => #Status);

            IF #Done = TRUE THEN
```

SCL

```
                IF "Productionline_Data"."ReadListProduct".NodeStatusList[0] = 0 AND
                    "Productionline_Data"."ReadListProduct".NodeStatusList[1] = 0
                THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at NodeReleaseHandleList';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_NodeReleaseHandleList_Instance(
                                                        NodeHdls :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeHdls,
                                                        NodeStatusList :=
"Productionline_Data"."ReadListProduct".NodeStatusList,
                                                        REQ := FALSE,
                                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                        NodeHdlCount :=
"Productionline_Configuration".ReadLists."ReadListProduct".NodeCount);
            END_IF;


        6:  // case 6, disconnect from server

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        REQ := #Req,
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        Done => #Done,
                                        Error => #Error,
                                        Busy => #Busy,
                                        Status => #Status);

            IF #Done = TRUE THEN
                #State := #State + 1;
            END_IF;

            IF #Error = TRUE THEN
```

SCL

```
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
            END_IF;

        7: //case = 7, function block has run successfully

            #Output_Done := TRUE;
            #Output_Busy := FALSE;
            #State := 0;
            #Mem_Input_REQ := FALSE;
            #Run := FALSE;
            #Started := FALSE;


        99: // ERROR handling
            #Output_Error := TRUE;
            #Output_Status := #Mem_Status;
            #State := 0;
            #Output_Busy := FALSE;
            #Run := FALSE;
            #Mem_Input_REQ := FALSE;
            #Started := FALSE;


        100:    // ERROR handling, disconnect from server to release resources (handles)

            IF #Set_REQ_To_FALSE = TRUE THEN
                #Set_REQ_To_FALSE := FALSE;

                //set REQ to FALSE
                #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                            REQ := FALSE,
                                            Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);
            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        REQ := #Req,
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        Done => #Done,
                                        Error => #Error,
                                        Busy => #Busy,
                                        Status => #Status);
```

SCL

```
            IF #Done = TRUE THEN
                #Output_Error := TRUE;
                #Output_Status := #Mem_Status;
                #State := 0;
                #Output_Busy := FALSE;
                #Run := FALSE;
                #Mem_Input_REQ := FALSE;
                #Started := FALSE;
            END_IF;

            IF #Error = TRUE THEN
                #Output_Error_Message := WString#'Error handling: Resources cannot be
released!';
                #Output_Error := TRUE;
                #Output_Status := #Mem_Status;
                #State := 0;
                #Output_Busy := FALSE;
                #Run := FALSE;
                #Mem_Input_REQ := FALSE;
                #Started := FALSE;
            END_IF;
    END_CASE;
END_IF;
```

Example program for writing PLC tags

Example program for writing PLC tags

This section includes the complete program code for the example program "WriteToProductionline".

The example shows you how a user program writes the values of PLC tags.

Program structure

The program operates as OPC UA client and comprises the following steps:

1. Establishing a connection to the OPC UA server of the CPU to which the values are to be written.
2. Writing the values
3. Terminating the connection to the OPC UA server

You start the program with a positive edge at "Input_REQ".

Declaration

The following figure shows the declaration of local tags for the function block "WriteToProductionline".

Input	
Input_REQ	Bool
Output	
Output_Busy	Bool
Output_Error	Bool
Output_Done	Bool
Output_Status	DWord
Output_Error_Message	WString
InOut	
<Hinzufügen>	
Static	
Mem_Input_REQ	Bool
Run	Bool
Started	Bool
State	Int
Mem_Status	DWord
Req	Bool
Busy	Bool
Error	Bool
Done	Bool
Status	DWord
Set_REQ_To_FALSE	Bool
i	Int
SetProductionEnabled	Bool
▸ OPC_UA_Connect_Instance	OPC_UA_Connect
▸ OPC_UA_NamespaceGetIndexList_Instance	OPC_UA_NamespaceGetIndexList
▸ OPC_UA_NodeGetHandleList_Instance	OPC_UA_NodeGetHandleList
▸ OPC_UA_WriteList_Instance	OPC_UA_WriteList
▸ OPC_UA_NodeReleaseHandleList_Instance	OPC_UA_NodeReleaseHandleList
▸ OPC_UA_Disconnect_Instance	OPC_UA_Disconnect

The example program also uses the following data blocks:

- Productionline_Configuration
- Productionline_Data

These data blocks are automatically created by STEP 7 (TIA Portal) because the example program uses the parameter assignment for OPC UA.

Program

SCL

```
IF #Run = FALSE THEN
    #Started := #Input_REQ AND NOT #Mem_Input_REQ;
    #Mem_Input_REQ := #Input_REQ;
    IF #Started THEN
        #Output_Busy := TRUE;
        #Output_Done := FALSE;
        #Output_Error := FALSE;
        #Output_Status := 0;
        #State := 1;
        #Run := TRUE;
        #SetProductionEnabled := TRUE;
        #Set_REQ_To_FALSE := TRUE;      //In case 100, to set REQ of instruction
"OPC_UA_Disvonnect" to FALSE
    END_IF;
END_IF;

IF #Run = TRUE THEN


    CASE #State OF
        1:  // case 1, connect to server

            IF #Busy = FALSE THEN

                #Req := TRUE;
            ELSE
                #Req := FALSE;

            END_IF;
            #OPC_UA_Connect_Instance(ServerEndpointUrl :=
"Productionline_Configuration".Connection.ServerEndpointUrl,
                                     SessionConnectInfo :=
"Productionline_Configuration".Connection.ConnectInfo,
                                     ConnectionHdl =>
"Productionline_Configuration".Connection.ConnectionHdl,
                                     Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                     REQ := #Req,
                                     Done => #Done,
                                     Busy => #Busy,
                                     Error => #Error,
                                     Status => #Status);


            IF #Done = TRUE THEN
                #State := #State + 1;
            END_IF;

            IF #Error = TRUE THEN

                // Did we get a connection handle?
                IF "Productionline_Configuration".Connection.ConnectionHdl <> 0 THEN
                    // We have to release all resources in the server and disconnect
                    #State := 100;
```

SCL

```
                    //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                ELSE
                    #State := 99;
                END_IF;

                #Mem_Status := #Status;
                #OPC_UA_Connect_Instance(ServerEndpointUrl :=
"Productionline_Configuration".Connection.ServerEndpointUrl,
                                         SessionConnectInfo :=
"Productionline_Configuration".Connection.ConnectInfo,
                                         REQ := FALSE,
                                         ConnectionHdl =>
"Productionline_Configuration".Connection.ConnectionHdl,
                                         Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);
            END_IF;

        2:  // case 2, get index for namespace

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_NamespaceGetIndexList_Instance(NamespaceUris :=
"Productionline_Configuration".Namespaces.NamespaceURIs,
                                                   ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                   NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes,
                                                   StatusList :=
"Productionline_Configuration".Namespaces.NamespaceStatusList,
                                                   REQ := #Req,
                                                   NamespaceUrisCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                   Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                   Busy => #Busy,
                                                   Done => #Done,
                                                   Error => #Error);

            IF #Done = TRUE THEN
               IF "Productionline_Configuration".Namespaces.NamespaceStatusList[0] = 0 THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at NamespaceGetIndexList';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
```

SCL

```
                #Mem_Status := #Status;
                #OPC_UA_NamespaceGetIndexList_Instance(NamespaceUris :=
"Productionline_Configuration".Namespaces.NamespaceURIs,
                                                        REQ := FALSE,
                                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                        NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes,
                                                        StatusList :=
"Productionline_Configuration".Namespaces.NamespaceStatusList,
                                                        NamespaceUrisCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);
            END_IF;

        3:  // case 3, get handles for nodes

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_NodeGetHandleList_Instance(NodeIDs :=
"Productionline_Configuration".WriteLists."WriteListStatus".Nodes,
                                               NodeStatusList :=
"Productionline_Data"."WriteListStatus".NodeStatusList,
                                               NodeHdls :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeHdls,
                                               REQ := #Req,
                                               ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                               Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                               NodeIDCount :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeCount,
                                               NamespaceIndexCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                               NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes,
                                               Busy => #Busy,
                                               Done => #Done,
                                               Error => #Error,
                                               Status =>#Status);

            IF #Done = TRUE THEN
                IF "Productionline_Data"."WriteListStatus".NodeStatusList[0] = 0
                THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at NodeGetHandleList';
                END_IF;
            END_IF;
```

SCL

```
            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_NodeGetHandleList_Instance(NodeIDs :=
"Productionline_Configuration".WriteLists."WriteListStatus".Nodes,
                                                   NodeStatusList :=
"Productionline_Data"."WriteListStatus".NodeStatusList,
                                                   NodeHdls :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeHdls,
                                                   REQ := FALSE,
                                                   ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                   Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                   NodeIDCount :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeCount,
                                                   NamespaceIndexCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                   NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes);
            END_IF;

        4:  // case 4;  write value to PLC variable "ProductionEnabled"

            IF #SetProductionEnabled = TRUE THEN
                #SetProductionEnabled := FALSE;
                //set new value to true
               "Productionline_Data"."WriteListStatus".Variable.ProductionEnabled := TRUE;
            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_WriteList_Instance(
                                       NodeHdls :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeHdls,
                                       NodeStatusList :=
"Productionline_Data"."WriteListStatus".NodeStatusList,
                                       Variable :=
"Productionline_Data"."WriteListStatus".Variable,
                                       ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                       Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                       NodeHdlCount :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeCount,
                                       REQ := #Req,
                                       Busy => #Busy,
                                       Error => #Error,
                                       Status => #Status,
```

SCL

```
                                        Done => #Done);

            IF #Done = TRUE THEN
                FOR #i := 0 TO
UINT_TO_INT("Productionline_Configuration".WriteLists.WriteListStatus.NodeCount) - 1 DO
                   IF NOT ("Productionline_Data"."WriteListStatus".NodeStatusList[#i] = 0)
THEN
                        #Output_Error_Message := CONCAT_WSTRING(IN1 := WSTRING#'Error at
Writelist "WriteListStauts", Index: ',
                                                                    IN2 := INT_TO_WSTRING(#i));
                    END_IF;
                END_FOR;
                #State := #State + 1;
            END_IF;
            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_WriteList_Instance(
                                        NodeHdls :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeHdls,
                                        NodeStatusList :=
"Productionline_Data"."WriteListStatus".NodeStatusList,
                                        Variable :=
"Productionline_Data"."WriteListStatus".Variable,
                                        REQ := FALSE,
                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        NodeHdlCount :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeCount);
            END_IF;

        5:  // case 5, release node handles

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_NodeReleaseHandleList_Instance(NodeHdls :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeHdls,
                                                NodeStatusList :=
"Productionline_Data"."WriteListStatus".NodeStatusList,
                                                REQ := #Req,
                                                ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                NodeHdlCount :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeCount,
                                                Busy => #Busy,
                                                Done => #Done,
                                                Error => #Error,
```

SCL

```
                                                Status => #Status);

            IF #Done = TRUE THEN
                IF "Productionline_Data"."WriteListStatus".NodeStatusList[0] = 0
                THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at NodeReleaseHandleList';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;

                #OPC_UA_NodeReleaseHandleList_Instance(NodeHdls :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeHdls,
                                                       NodeStatusList :=
"Productionline_Data"."WriteListStatus".NodeStatusList,
                                                       REQ := FALSE,
                                                       ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                       Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                       NodeHdlCount :=
"Productionline_Configuration".WriteLists."WriteListStatus".NodeCount);

                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
            END_IF;


        6:  // case 6, disconnect from server

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        REQ := #Req,
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        Busy => #Busy,
                                        Done => #Done,
                                        Error => #Error,
                                        Status => #Status);


            IF #Done = TRUE THEN
                #State := #State + 1;
```

SCL

```
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
            END_IF;

        7: //case = 7, function block has run successfully

            #Output_Done := TRUE;
            #Output_Busy := FALSE;
            #State := 0;
            #Mem_Input_REQ := FALSE;
            #Run := FALSE;
            #Started := FALSE;

        99: // ERROR handling
            #Output_Error := TRUE;
            #Output_Status := #Mem_Status;
            #State := 0;
            #Output_Busy := FALSE;
            #Run := FALSE;
            #Mem_Input_REQ := FALSE;
            #Started := FALSE;

        100:    // ERROR handling, disconnect from server to release resources (handles)

            IF #Set_REQ_To_FALSE = TRUE THEN
                #Set_REQ_To_FALSE := FALSE;

                //set REQ to FALSE
                #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                            REQ := FALSE,
                                            Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);
            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        REQ := #Req,
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        Done => #Done,
                                        Busy => #Busy,
                                        Error => #Error,
                                        Status => #Status);
```

SCL

```
            IF #Done = TRUE THEN
                #Output_Error := TRUE;
                #Output_Status := #Mem_Status;
                #State := 0;
                #Output_Busy := FALSE;
                #Run := FALSE;
                #Mem_Input_REQ := FALSE;
                #Started := FALSE;
            END_IF;

            IF #Error = TRUE THEN
                #Output_Error_Message := WString#'Error handling: Resources cannot be
released!';
                #Output_Error := TRUE;
                #Output_Status := #Mem_Status;
                #State := 0;
                #Output_Busy := FALSE;
                #Run := FALSE;
                #Mem_Input_REQ := FALSE;
                #Started := FALSE;
            END_IF;

    END_CASE;


END_IF;
```

Example program for OPC_UA_TranslatePathList

Example program for the instruction "OPC_UA_TranslatePathList"

This section includes the complete program code for the example program "Read_From_RfidReader_Door_1".

The example shows you how a user program uses the instruction "OPC_UA_TranslatePathList" to get the NodeIDs of nodes and then read the values of these nodes.

The description of the instruction "OPC_UA_TranslatePathList" can be found here (Page 3674).

Program structure

The program operates as OPC UA client and comprises the following steps:

1. Establishing a connection to the OPC UA server of the CPU from which the values are to be read.
2. Reading the values.
3. Terminating the connection to the OPC UA server.

You start the program with a positive edge at "Input_REQ".

Declaration

The following figure shows the declaration of local tags of the function block "Read_From_RfidReader_Door_1" that uses the instruction "OPC_UA_TanslatePathList":

Read_From_RfidReader_Door_1		
	Name	Data type
1	Input	
2	Input_REQ	Bool
3	Output	
4	Output_Busy	Bool
5	Output_Error	Bool
6	Output_Done	Bool
7	Output_Status	DWord
8	Output_Error_Message	WString

11	Static	
12	Mem_Input_REQ	Bool
13	Run	Bool
14	Started	Bool
15	State	Int
16	Mem_Status	DWord
17	Req	Bool
18	Busy	Bool
19	Error	Bool
20	Done	Bool
21	Status	DWord
22	i	Int
23	FirstCall	Bool
24	SeverEndpointUrl	WString
25	SessionConnectInfo	OPC_UA_SessionConnectInfo
26	OPC_UA_Connect_Instance	OPC_UA_Connect
27	NamespaceUris	Array[0..0] of WString
28	NamespaceIndexes	Array[0..0] of UInt
29	NamespaceStatusList	Array[0..0] of DWord
30	OPC_UA_NamespaceGetIndexList_Instance	OPC_UA_NamespaceGetIndexList
31	OPC_UA_NodeGetHandleList_Instance	OPC_UA_NodeGetHandleList
32	OPC_UA_ReadList_Instance	OPC_UA_ReadList
33	OPC_UA_NodeReleaseHandleList_Instance	OPC_UA_NodeReleaseHandleList
34	OPC_UA_Disconnect_Instance	OPC_UA_Disconnect
35	OPC_UA_TranslatePathList_Instance	OPC_UA_TranslatePathList
36	BrowsePaths	Array[0..1] of OPC_UA_BrowsePath
37	TagetNodeIds	Array[0..1] of OPC_UA_NodeId
38	TargetStatusList	Array[0..1] of DWord
39	NodeStatusList	Array[0..1] of DWord
40	NodeHdls	Array[0..1] of DWord
41	ConnectionHdl	DWord
42	Set_REQ_To_FALSE	Bool
43	Variable	"UDT_Variable"

Program

The following example shows you how to use the instruction "OPC UA-TranslatePathList":

SCL

```
IF #Run = FALSE THEN
    #Started := #Input_REQ AND NOT #Mem_Input_REQ;
    #Mem_Input_REQ := #Input_REQ;
    IF #Started THEN
        #Output_Busy := TRUE;
        #Output_Done := FALSE;
        #Output_Error := FALSE;
        #Output_Status := 0;
        #State := 1;
        #Run := TRUE;
        #FirstCall := TRUE;
        #Set_REQ_To_FALSE := TRUE;      //In case 100, to set REG of instruction
"OPC_UA_Disconnect" to FALSE
    END_IF;
END_IF;

IF #Run = TRUE THEN


    CASE #State OF
        1:  // case 1, connect to server


            IF #FirstCall = TRUE THEN
                #FirstCall := FALSE;

                //set ServerEndPointUrl of the server we want to connect to
                #SeverEndpointUrl := WString#'opc.tcp://192.168.1.1:4840';

                //set Security Message Mode
                // 1 = None, 2 = Sign, 3 = Sign & Encrypt
                #SessionConnectInfo.SecurityMsgMode := 1;

                //set Security Policy
                // 1 = None, 2 = Basic128Rsa15,, 3 = Basic256, 4 = Basic256Sha256
                #SessionConnectInfo.SecurityPolicy := 1;

                //set the number of the client certificate you are using
                //in our case, this is 6
                //#SessionConnectInfo.CertificateID := 6;
            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_Connect_Instance(REQ:=#Req,
                                     ServerEndpointUrl:=#SeverEndpointUrl,
                                     SessionConnectInfo:=#SessionConnectInfo,
                                     Timeout:=T#8S,
                                     Done=>#Done,
                                     Busy=>#Busy,
                                     Error=>#Error,
```

SCL

```
                                Status=>#Status,
                                ConnectionHdl=>#ConnectionHdl);

        IF #Done = TRUE THEN
            #State := #State + 1;
            #FirstCall := TRUE;
        END_IF;

        IF #Error = TRUE THEN
            // Did we get a connection handle?
            IF #ConnectionHdl <> 0 THEN
                // We got a connection handle
                // We have to release all resources in the server and disconnect
                #State := 100;
                //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
            ELSE
                // We did not get a connection handle
                #State := 99;
            END_IF;

            #Mem_Status := #Status;
            // set parameter REQ of OPC_UA_Connect to FALSE
            #OPC_UA_Connect_Instance(ServerEndpointUrl := #SeverEndpointUrl,
                                     SessionConnectInfo := #SessionConnectInfo,
                                     REQ := FALSE);
            #FirstCall := TRUE;
        END_IF;

    2:  // case 2, get index of Siemens namespace


        IF #FirstCall = TRUE THEN
            #FirstCall := FALSE;

            //set Simatic namespace
            #NamespaceUris[0] := WString#'http://www.siemens.com/simatic-s7-opcua';
        END_IF;

        IF #Busy = FALSE THEN
            #Req := TRUE;
        ELSE
            #Req := FALSE;
        END_IF;

        #OPC_UA_NamespaceGetIndexList_Instance(NamespaceUris := #NamespaceUris,
                                               ConnectionHdl := #ConnectionHdl,
                                               NamespaceIndexes := #NamespaceIndexes,
                                               StatusList := #NamespaceStatusList,
                                               REQ := #Req,
                                               NamespaceUrisCount := 1,
                                               Timeout := T#6S,
                                               Busy => #Busy,
                                               Done => #Done,
                                               Error => #Error,
                                               Status => #Status);
```

SCL

```
            IF #Done = TRUE THEN
                IF #NamespaceStatusList[0] = 0 THEN  //one namespace
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    #Mem_Status := #Status;
                  #Output_Error_Message := WString#' Error at NamespaceGetIndexList, Index
0';
                END_IF;
                #FirstCall := TRUE;
            END_IF;

            IF #Error = TRUE THEN
                //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #State := 100;
                #Mem_Status := #Status;
                #OPC_UA_NamespaceGetIndexList_Instance(NamespaceUris := #NamespaceUris,
                                                       REQ := FALSE,
                                                       StatusList := #NamespaceStatusList);
                #FirstCall := TRUE;
            END_IF;



        3:  // case 3, get NodeIds from server




            IF #FirstCall = TRUE THEN
                //set BrowsePath[0] for DeviceInfo
                //set BrowsePath.StartingNode
                #BrowsePaths[0].StartingNode.Identifier := WString#'85';
                #BrowsePaths[0].StartingNode.IdentifierType := 0;
                #BrowsePaths[0].StartingNode.NamespaceIndex := 0;

                //set BrowsePath.RelativePath
                #BrowsePaths[0].RelativePath.NoOfElements := 4;
                #BrowsePaths[0].RelativePath.Elements[1].ReferenceTypeId.Identifier :=
WString#'33';
                #BrowsePaths[0].RelativePath.Elements[1].ReferenceTypeId.IdentifierType :=
0;
                #BrowsePaths[0].RelativePath.Elements[1].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[0].RelativePath.Elements[1].TargetName.Name :=
WString#'productionline';
                #BrowsePaths[0].RelativePath.Elements[1].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

                #BrowsePaths[0].RelativePath.Elements[2].ReferenceTypeId.Identifier :=
WString#'33';
                #BrowsePaths[0].RelativePath.Elements[2].ReferenceTypeId.IdentifierType :=
0;
```

SCL

```
                #BrowsePaths[0].RelativePath.Elements[2].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[0].RelativePath.Elements[2].TargetName.Name :=
WString#'DataBlocksGlobal';
                #BrowsePaths[0].RelativePath.Elements[2].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

                #BrowsePaths[0].RelativePath.Elements[3].ReferenceTypeId.Identifier :=
WString#'33';
                #BrowsePaths[0].RelativePath.Elements[3].ReferenceTypeId.IdentifierType :=
0;
                #BrowsePaths[0].RelativePath.Elements[3].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[0].RelativePath.Elements[3].TargetName.Name :=
WString#'RfidReader_Door_1';
                #BrowsePaths[0].RelativePath.Elements[3].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

                #BrowsePaths[0].RelativePath.Elements[4].ReferenceTypeId.Identifier :=
WString#'33';
                #BrowsePaths[0].RelativePath.Elements[4].ReferenceTypeId.IdentifierType :=
0;
                #BrowsePaths[0].RelativePath.Elements[4].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[0].RelativePath.Elements[4].TargetName.Name :=
WString#'DeviceInfo';
                #BrowsePaths[0].RelativePath.Elements[4].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

                //set BrowsePath[1] for DeviceStatus
                //set BrowsePath.StartingNode
                #BrowsePaths[1].StartingNode.Identifier := WString#'85';
                #BrowsePaths[1].StartingNode.IdentifierType := 0;
                #BrowsePaths[1].StartingNode.NamespaceIndex := 0;

                //set BrowsePath.RelativePath
                #BrowsePaths[1].RelativePath.NoOfElements := 4;
                #BrowsePaths[1].RelativePath.Elements[1].ReferenceTypeId.Identifier :=
WString#'33';
                #BrowsePaths[1].RelativePath.Elements[1].ReferenceTypeId.IdentifierType :=
0;
                #BrowsePaths[1].RelativePath.Elements[1].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[1].RelativePath.Elements[1].TargetName.Name :=
WString#'productionline';
                #BrowsePaths[1].RelativePath.Elements[1].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

                #BrowsePaths[1].RelativePath.Elements[2].ReferenceTypeId.Identifier :=
WString#'33';
                #BrowsePaths[1].RelativePath.Elements[2].ReferenceTypeId.IdentifierType :=
0;
                #BrowsePaths[1].RelativePath.Elements[2].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[1].RelativePath.Elements[2].TargetName.Name :=
WString#'DataBlocksGlobal';
```

SCL

```
                #BrowsePaths[1].RelativePath.Elements[2].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

                #BrowsePaths[1].RelativePath.Elements[3].ReferenceTypeId.Identifier :=
WString#'33';
                #BrowsePaths[1].RelativePath.Elements[3].ReferenceTypeId.IdentifierType :=
0;
                #BrowsePaths[1].RelativePath.Elements[3].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[1].RelativePath.Elements[3].TargetName.Name :=
WString#'RfidReader_Door_1';
                #BrowsePaths[1].RelativePath.Elements[3].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

                #BrowsePaths[1].RelativePath.Elements[4].ReferenceTypeId.Identifier :=
WString#'33';
                #BrowsePaths[1].RelativePath.Elements[4].ReferenceTypeId.IdentifierType :=
0;
                #BrowsePaths[1].RelativePath.Elements[4].ReferenceTypeId.NamespaceIndex :=
0;
                #BrowsePaths[1].RelativePath.Elements[4].TargetName.Name :=
WString#'DeviceStatus';
                #BrowsePaths[1].RelativePath.Elements[4].TargetName.NamespaceIndex :=
#NamespaceIndexes[0];

            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_TranslatePathList_Instance(BrowsePaths := #BrowsePaths,
                                               TargetNodeIDs := #TagetNodeIds,
                                               TargetStatusList := #TargetStatusList,
                                               ConnectionHdl := #ConnectionHdl,
                                               NamespaceIndexCount := 0,
                                               BrowsePathsCount := 2,
                                               Timeout := T#6S,
                                               REQ := #Req,
                                               Busy => #Busy,
                                               Error => #Error,
                                               Done => #Done,
                                               Status => #Status);
            IF #Done = TRUE THEN
                IF #TargetStatusList[0] = 0 AND #TargetStatusList[1] = 0 THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at TranslatePathList';
                END_IF;
                #FirstCall := TRUE;
            END_IF;
            IF #Error = TRUE THEN
```

SCL

```
            //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
            #Set_REQ_To_FALSE := TRUE;
            #State := 100;
            #Mem_Status := #Status;
            #OPC_UA_TranslatePathList_Instance(BrowsePaths := #BrowsePaths,
                                               TargetNodeIDs := #TagetNodeIds,
                                               REQ := FALSE,
                                               TargetStatusList := #TargetStatusList,
                                               ConnectionHdl := #ConnectionHdl);
            #FirstCall := TRUE;
        END_IF;


    4:  // case 4, get handles for the NodeIds we have got in case 3




        IF #Busy = FALSE THEN
            #Req := TRUE;
        ELSE
            #Req := FALSE;
        END_IF;

        #OPC_UA_NodeGetHandleList_Instance(NodeIDs := #TagetNodeIds,
                                           NodeStatusList := #NodeStatusList,
                                           NodeHdls := #NodeHdls,
                                           ConnectionHdl := #ConnectionHdl,
                                           Timeout := T#6S,
                                           NodeIDCount := 2,
                                           REQ := #Req,
                                           Busy => #Busy,
                                           Done => #Done,
                                           Error => #Error,
                                           Status => #Status);
        IF #Done = TRUE THEN
            IF #NodeStatusList[0] = 0 AND #NodeStatusList[1] = 0
            THEN
                #State := #State + 1;
            ELSE
                //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #State := 100;
                #Mem_Status := #Status;
                #Output_Error_Message := WString#'Error at NodeGetHandleList';
            END_IF;
        END_IF;

        IF #Error = TRUE THEN
            //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
            #Set_REQ_To_FALSE := TRUE;
            #State := 100;
            #Mem_Status := #Status;
            #OPC_UA_NodeGetHandleList_Instance(NodeIDs := #TagetNodeIds,
                                               NodeStatusList := #NodeStatusList,
                                               NodeHdls := #NodeHdls,
```

SCL

```
                                                REQ := FALSE);
            END_IF;

        5:  // case 5, read the values of the NodeIds



            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_ReadList_Instance(NodeHdls := #NodeHdls,
                                      NodeStatusList := #NodeStatusList,
                                      Variable := #Variable,
                                      REQ := #Req,
                                      ConnectionHdl := #ConnectionHdl,
                                      Timeout := T#6S,
                                      NodeHdlCount := 2,
                                      Busy => #Busy,
                                      Done => #Done,
                                      Error => #Error,
                                      Status => #Status);


            IF #Done = TRUE THEN
                FOR #i := 0 TO 1 DO
                    IF NOT (#NodeStatusList[#i] = 0) THEN
                        #Output_Error_Message := CONCAT_WSTRING(IN1 := WSTRING#'Error at
Readlist, Index: ',
                                                                IN2 := INT_TO_WSTRING(#i));
                    END_IF;
                END_FOR;
                #State := #State + 1;
            END_IF;

            IF #Error = TRUE THEN
                //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #State := 100;
                #Mem_Status := #Status;
                #OPC_UA_ReadList_Instance(NodeHdls := #NodeHdls,
                                          NodeStatusList := #NodeStatusList,
                                          Variable := #Variable,
                                          REQ := FALSE);
            END_IF;


        6:  // case 6, release the node handles


            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
```

SCL

```
        END_IF;

        #OPC_UA_NodeReleaseHandleList_Instance(NodeHdls := #NodeHdls,
                                               NodeStatusList := #NodeStatusList,
                                               REQ := #Req,
                                               ConnectionHdl := #ConnectionHdl,
                                               Timeout := T#6S,
                                               NodeHdlCount := 1,
                                               Busy => #Busy,
                                               Done => #Done,
                                               Error => #Error,
                                               Status => #Status);

        IF #NodeStatusList[0] = 0 AND #NodeStatusList[1] = 0
        THEN
            #State := #State + 1;
        ELSE
            //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
            #Set_REQ_To_FALSE := TRUE;
            #State := 100;
            #Mem_Status := #Status;
            #Output_Error_Message := WString#'Error at NodeReleasetHandleList';
        END_IF;

        IF #Error = TRUE THEN
            //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
            #Set_REQ_To_FALSE := TRUE;
            #State := 100;
            #Mem_Status := #Status;
            #OPC_UA_NodeReleaseHandleList_Instance(NodeHdls := #NodeHdls,
                                                   NodeStatusList := #NodeStatusList,
                                                   REQ := FALSE);
        END_IF;


    7:  // case 7; disconnect from server


        IF #Busy = FALSE THEN
            #Req := TRUE;
        ELSE
            #Req := FALSE;
        END_IF;

        #OPC_UA_Disconnect_Instance(ConnectionHdl := #ConnectionHdl,
                                    REQ := #Req,
                                    Timeout := T#6S,
                                    Busy => #Busy,
                                    Done => #Done,
                                    Error => #Error,
                                    Status => #Status);

        IF #Done = TRUE THEN
            #State := #State + 1;
        END_IF;
```

SCL

```
        IF #Error = TRUE THEN
            //In case 100, to set REG of instruction "OPC_UA_Disconnect" to FALSE
            #Set_REQ_To_FALSE := TRUE;
            #State := 100;
            #Mem_Status := #Status;
            #OPC_UA_Disconnect_Instance(ConnectionHdl := #ConnectionHdl,
                                        REQ := FALSE);

        END_IF;


    8: //case = 8, function block has run successfully



        #Output_Done := TRUE;
        #Output_Busy := FALSE;
        #State := 0;
        #Mem_Input_REQ := FALSE;
        #Run := FALSE;
        #Started := FALSE;


    99: // ERROR handling



        #Output_Error := TRUE;
        #Output_Status := #Mem_Status;
        #State := 0;
        #Output_Busy := FALSE;
        #Run := FALSE;
        #Mem_Input_REQ := FALSE;
        #Started := FALSE;


    100:    // ERROR handling, disconnect from server to release resources (handles)



        IF #Set_REQ_To_FALSE = TRUE THEN
            #Set_REQ_To_FALSE := FALSE;

            //set REQ to FALSE
            #OPC_UA_Disconnect_Instance(ConnectionHdl := #ConnectionHdl,
                                        REQ := FALSE);
        END_IF;

        IF #Busy = FALSE THEN
            #Req := TRUE;
        ELSE
            #Req := FALSE;
        END_IF;

        #OPC_UA_Disconnect_Instance(ConnectionHdl := #ConnectionHdl,
```

SCL

```
                                        REQ := #Req,
                                        Timeout := T#6S,
                                        Busy => #Busy,
                                        Done => #Done,
                                        Error => #Error,
                                        Status => #Status);


            IF #Done = TRUE THEN
                #Output_Error := TRUE;
                #Output_Status := #Mem_Status;
                #State := 0;
                #Output_Busy := FALSE;
                #Run := FALSE;
                #Mem_Input_REQ := FALSE;
                #Started := FALSE;
            END_IF;

            IF #Error = TRUE THEN
                #Output_Error_Message := WString#'Error handling: Resources cannot be
released!';
                #Output_Error := TRUE;
                #Output_Status := #Mem_Status;
                #State := 0;
                #Output_Busy := FALSE;
                #Run := FALSE;
                #Mem_Input_REQ := FALSE;
                #Started := FALSE;
            END_IF;

    END_CASE;

END_IF;
```

Example program for calling a method of an OPC UA server

Example program for calling a server method

This section includes the complete program code for the example program "Call_OpenDoor_On_Productionline".

The example shows you how a user program uses the instruction "OPC_UA_MethodCall" to call a server method.

The description of the instruction "OPC_UA_MethodCall" can be found here (Page 3744).

Program structure

The program operates as OPC UA client and comprises the following steps:

1. Establishing a connection to an OPC UA server.
2. Calling the server method.
3. Terminating the connection to the OPC UA server.

You start the program with a positive edge at "Input_REQ".

Declaration

The following figure shows the declaration of local tags for the function block ""Call_OpenDoor_On_Productionline".

Input	
Input_REQ	Bool
Output	
Output_Busy	Bool
Output_Error	Bool
Output_Done	Bool
Output_Status	DWord
Output_Error_Message	WString
Output_MethodResult	DWord
InOut	
<Hinzufügen>	
Static	
Mem_Input_REQ	Bool
Run	Bool
Started	Bool
State	Int
Mem_Status	DWord
MethodResult	DWord
Req	Bool
Busy	Bool
Error	Bool
Done	Bool
Status	DWord
Init_Method_InputParameter	Bool
Set_REQ_To_FALSE	Bool
▶ OPC_UA_Connect_Instance	OPC_UA_Connect
▶ OPC_UA_NamespaceGetIndexList_Instance	OPC_UA_NamespaceGetIndexList
▶ OPC_UA_MethodGetHandleList_Instance	OPC_UA_MethodGetHandleList
▶ OPC_UA_MethodCall_Instance	OPC_UA_MethodCall
▶ OPC_UA_MethodReleaseHandleList_Instance	OPC_UA_MethodReleaseHandleList
▶ OPC_UA_Disconnect_Instance	OPC_UA_Disconnect

Program

SCL

```
IF #Run = FALSE THEN
    #Started := #Input_REQ AND NOT #Mem_Input_REQ;
    #Mem_Input_REQ := #Input_REQ;
    IF #Started THEN
        #Output_Busy := TRUE;
        #Output_Done := FALSE;
        #Output_Error := FALSE;
        #Output_Status := 0;
        #Output_Error_Message := WSTRING#'';
        #State := 1;
        #Run := TRUE;
        #Init_Method_InputParameter := TRUE;
        #Set_REQ_To_FALSE := TRUE;      //In case 100, to set REQ of instruction
"OPC_UA_Disconnect" to FALSE
    END_IF;
END_IF;


IF #Run = TRUE THEN


    CASE #State OF
        1:  // case 1, connect to server

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_Connect_Instance(REQ := #Req,
                                     ServerEndpointUrl :=
"Productionline_Configuration".Connection.ServerEndpointUrl,
                                     SessionConnectInfo :=
"Productionline_Configuration".Connection.ConnectInfo,
                                     Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                     Done => #Done,
                                     Busy => #Busy,
                                     Error => #Error,
                                     Status => #Status,
                                     ConnectionHdl =>
"Productionline_Configuration".Connection.ConnectionHdl);


            IF #Done = TRUE THEN
                #State := #State + 1;
            END_IF;

            IF #Error = TRUE THEN
                // Did we get a connection handle?
                IF "Productionline_Configuration".Connection.ConnectionHdl <> 0 THEN
                    // We have to release all resources in the server and disconnect
                    #State := 100;
                    //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
```

SCL

```
                    #Set_REQ_To_FALSE := TRUE;
                ELSE
                    #State := 99;
                END_IF;
                #Mem_Status := #Status;
                #OPC_UA_Connect_Instance(ServerEndpointUrl :=
"Productionline_Configuration".Connection.ServerEndpointUrl,
                                         SessionConnectInfo :=
"Productionline_Configuration".Connection.ConnectInfo,
                                         REQ := FALSE,
                                         ConnectionHdl =>
"Productionline_Configuration".Connection.ConnectionHdl,
                                         Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);

            END_IF;

        2:  // case 2, get index of Siemens namespace

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_NamespaceGetIndexList_Instance(REQ := #Req,
                                                   ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                   NamespaceUrisCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                   NamespaceUris :=
"Productionline_Configuration".Namespaces.NamespaceURIs,
                                                   Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                   Done => #Done,
                                                   Busy => #Busy,
                                                   Error => #Error,
                                                   Status => #Status,
                                                   StatusList :=
"Productionline_Configuration".Namespaces.NamespaceStatusList,
                                                   NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes);


            IF #Done = TRUE THEN
                IF "Productionline_Configuration".Namespaces.NamespaceStatusList[0] = 0
THEN  //one namespace
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                    #Mem_Status := #Status;
                  #Output_Error_Message := WString#' Error at NamespaceGetIndexList, Index
0';
                END_IF;
```

SCL

```
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_NamespaceGetIndexList_Instance(NamespaceUris :=
"Productionline_Configuration".Namespaces.NamespaceURIs,
                                                       REQ := FALSE,
                                                       ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                       NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes,
                                                       StatusList :=
"Productionline_Configuration".Namespaces.NamespaceStatusList,
                                                       NamespaceUrisCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                       Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);
            END_IF;


        3:  // case 3, get an handle for server method OpenDoor

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_MethodGetHandleList_Instance(
                                                 REQ := #Req,
                                                 ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                 ObjectNodeIDs :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".ObjectNodes,
                                                 MethodNodeIDs :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodNodes,
                                                 NamespaceIndexCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                 Done => #Done,
                                                 Busy => #Busy,
                                                 Error => #Error,
                                                 Status => #Status,
                                                 StatusList :=
"Productionline_Data"."MethodListOpenDoor".MethodStatusList,
                                                 MethodHdls :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls,
                                                 Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                 NodeIDCount :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodCount,
                                                 NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes);
```

SCL

```
            IF #Done = TRUE THEN
                IF "Productionline_Data"."MethodListOpenDoor".MethodStatusList[0] = 0
THEN      //one method
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at MethodGetHandleList';
                END_IF;
            END_IF;
            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_MethodGetHandleList_Instance(
                                                    ObjectNodeIDs :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".ObjectNodes,
                                                    ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                    MethodNodeIDs :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodNodes,
                                                    StatusList :=
"Productionline_Data"."MethodListOpenDoor".MethodStatusList,
                                                    MethodHdls :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls,
                                                    REQ := FALSE,
                                                    Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                    NodeIDCount :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodCount,
                                                    NamespaceIndexCount :=
"Productionline_Configuration".Namespaces.NamespaceCount,
                                                    NamespaceIndexes :=
"Productionline_Configuration".Namespaces.ServerNamespaceIndexes);
            END_IF;

        4:  // case 4, call method OpenDoor

            IF #Init_Method_InputParameter = TRUE THEN
                #Init_Method_InputParameter := FALSE;
                // for our server method at Productionline, we set input parameters to 1
                "Productionline_Data".MethodListOpenDoor.Method.Inputs.Number := 1;
            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_MethodCall_Instance(
                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
```

SCL

```
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        MethodHdl :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls[0],
                                        Status =>
"Productionline_Data"."MethodListOpenDoor".MethodStatusList[0],
                                        InputArguments :=
"Productionline_Data"."MethodListOpenDoor"."Method".Inputs,
                                        OutputArguments :=
"Productionline_Data"."MethodListOpenDoor"."Method".Outputs,
                                        REQ := #Req,
                                        Done => #Done,
                                        Busy => #Busy,
                                        Error => #Error,
                                        MethodResult := #MethodResult);

            #Output_MethodResult := #MethodResult;

            IF #Done = TRUE THEN
                IF #MethodResult < 16#8000_0000 THEN
                    #State := #State + 1;
                ELSE
                    //server method did not run successfully
                    #State := 100;
                    //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#' Error at server method';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
               #Mem_Status := "Productionline_Data".MethodListOpenDoor.MethodStatusList[0];
                #OPC_UA_MethodCall_Instance(
                                            ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                            MethodHdl :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls[0],
                                            REQ := FALSE,
                                            Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                            Status =>
"Productionline_Data"."MethodListOpenDoor".MethodStatusList[0],
                                            InputArguments :=
"Productionline_Data"."MethodListOpenDoor"."Method".Inputs,
                                            OutputArguments :=
"Productionline_Data"."MethodListOpenDoor"."Method".Outputs,
                                            MethodResult := #MethodResult);
            END_IF;


        5:  // case 5, release method handle
```

SCL

```
            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_MethodReleaseHandleList_Instance(
                                                    REQ := #Req,
                                                    ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                    MethodHdlCount :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodCount,
                                                    MethodHdls :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls,
                                                    Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                                    Done => #Done,
                                                    Busy => #Busy,
                                                    Error => #Error,
                                                    Status => #Status,
                                                    StatusList :=
"Productionline_Data"."MethodListOpenDoor".MethodStatusList);


            IF #Done = TRUE THEN
                #Mem_Status := #Status;
                IF "Productionline_Data"."MethodListOpenDoor".MethodStatusList[0] = 0 THEN
                    #State := #State + 1;
                ELSE
                    #State := 100;
                    //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                    #Set_REQ_To_FALSE := TRUE;
                    #Mem_Status := #Status;
                    #Output_Error_Message := WString#'Error at MethodGetHandleList';
                END_IF;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_MethodReleaseHandleList_Instance(
                                                        MethodHdls :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodHdls,
                                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                                        StatusList :=
"Productionline_Data"."MethodListOpenDoor".MethodStatusList,
                                                        MethodHdlCount :=
"Productionline_Configuration".MethodLists."MethodListOpenDoor".MethodCount,
                                                        REQ := FALSE,
                                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);
            END_IF;
```

SCL

```
        6:  // case 6, disconnect from server

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;
            #OPC_UA_Disconnect_Instance(REQ := #Req,
                                        ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        Done => #Done,
                                        Busy => #Busy,
                                        Error => #Error,
                                        Status => #Status);


            IF #Done = TRUE THEN
                #State := #State + 1;
                #Mem_Status := #Status;
            END_IF;

            IF #Error = TRUE THEN
                #State := 100;
                //In case 100, to set REQ of instruction "OPC_UA_Disconnect" to FALSE
                #Set_REQ_To_FALSE := TRUE;
                #Mem_Status := #Status;
                #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                            REQ := FALSE,
                                            Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);

            END_IF;


        7: //case = 7, function block has run successfully

            #Output_Done := TRUE;
            #Output_Busy := FALSE;
            #State := 0;
            #Mem_Input_REQ := FALSE;
            #Run := FALSE;
            #Started := FALSE;


        99: // ERROR handling
            #Output_Error := TRUE;
            #Output_Status := #Mem_Status;
            #State := 0;
            #Output_Busy := FALSE;
            #Run := FALSE;
            #Mem_Input_REQ := FALSE;
            #Started := FALSE;
```

SCL

```
        100:    // ERROR handling, disconnect from server to release resources (handles)
            IF #Set_REQ_To_FALSE = TRUE THEN
                #Set_REQ_To_FALSE := FALSE;

                //set REQ to definitely to FALSE
                #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                            REQ := FALSE,
                                            Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout);
            END_IF;

            IF #Busy = FALSE THEN
                #Req := TRUE;
            ELSE
                #Req := FALSE;
            END_IF;

            #OPC_UA_Disconnect_Instance(ConnectionHdl :=
"Productionline_Configuration".Connection.ConnectionHdl,
                                        REQ := #Req,
                                        Timeout :=
"Productionline_Configuration".Connection.ConnectInfo.SessionTimeout,
                                        Busy => #Busy,
                                        Done => #Done,
                                        Error => #Error,
                                        Status => #Status);


            IF #Done = TRUE THEN
                #Output_Error := TRUE;
                #Output_Status := #Mem_Status;
                #State := 0;
                #Output_Busy := FALSE;
                #Run := FALSE;
                #Mem_Input_REQ := FALSE;
                #Started := FALSE;
            END_IF;

            IF #Error = TRUE THEN
                #Output_Error_Message := WString#'Error handling: Resources cannot be
released!';
                #Output_Error := TRUE;
                #Output_Status := #Mem_Status;
                #State := 0;
                #Output_Busy := FALSE;
                #Run := FALSE;
                #Mem_Input_REQ := FALSE;
                #Started := FALSE;
            END_IF;

    END_CASE;

END_IF;
```

Server instructions

OPC_UA_ServerMethodPre: Preparing the server method call

Validity

The following description of the "OPC_UA_ServerMethodPre" instruction applies to S7-1500 CPUs with firmware version V2.5 and higher.

Description

This section describes the instruction "OPC_UA_ServerMethodPre".

Because the instructions "OPC_UA_ServerMethodPre" and "OPC_UA_ServerMethodPost" always have to be called up in pairs in the user program, please also take into account the section on the instruction "OPC_UA_ServerMethodPost".

Parameters for "OPC_UA_ServerMethodPre"

Table 4-92 The parameters of the instruction "OPC_UA_ServerMethodPre"

Parameter	Declaration	Data type	Meaning
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error codes for the status" below.
UAMethodCalled	Output	BOOL	The method provided has been called by an OPC UA client.
UAMethod_InParameters	InOut	VARIANT	Pointer to a tag that contains the input parameters for the method provided.

Error codes for the status

The "Status" parameter provides information about errors that occurred during execution of the instruction.

The following table describes the different categories of error codes.

Table 4-93 Error codes for the status

Error code (hexadecimal values)	Explanation
0000_0000	Instruction finished successfully.
8xxx_xxxx	OPC UA specific error
Axxx_xxxx	PLCopen specific error
B080_C300	Insufficient resources
B08x_yz00	SIMATIC-specific error
For more error codes, see AUTOHOTSPOT	

Declaration of tags

Declare an instance of the instruction "OPC_UA_ServerMethodPre" and the tags with which you supply the instruction parameters, see also Example program for providing a method for OPC UA clients (Page 3833).

The following points are important for the declaration:

- Create the instruction "OPC_UA_ServerMethodPre" as a multi-instance in the calling function block.

  Note

  Name of the multiple instance

  The multiple instance has to be named "OPC_UA_ServerMethodPre_Instance", otherwise no method is created on the server.

  Use drag-and-drop to move the instruction from the folder "Instructions > Communication > OPC UA > OPC UA Server" to the editor.
  Next, click "Multi-instance".

- When the server method has one or more output parameters, you must declare a tag with the name "**UAMethod_InParameters**".
  First, create a PLC data type (UDT) for the input parameters of the server method.
  Then use this UDT for the tag "UAMethod_InParameters".
  In the example, the data type is called "UDT_OpenDoorInArguments" and includes the element Number.
  Alternative:
  You can also assign the data type "Struct" to the tag "**UAMethod_InParameters**". Then create the components of this data type according to the input parameters of the server method (same names and data types).

OPC_UA_ServerMethodPre_Instance	OPC_UA_ServerMethodPre
Input	
Output	
Done	Bool
Busy	Bool
Error	Bool
Status	DWord
UAMethod_Called	Bool
InOut	
UAMethod_InParameters	Variant
Static	
UAMethod_InParameters	"UDT_OpenDoorInArguments"
Number	Int

Calling the instruction

The instruction "OPC_UA_ServerMethodPre" queries the operating system to determine whether or not the server method was called.

If the server method was called by the client, the instruction "OPC_UA_ServerMethodPre" provides the input parameters for the server method.

Assignment of data types (SIMATIC – OPC UA)

For the input and output parameters of methods, please note the explanations regarding the rules on usable data types in the section (AUTOHOTSPOT).

Note

Supply of structured data types with nested arrays

If a structured data type (Struct/UDT) contains an array, the OPC UA server does not provide information about the length of this array.

If you use such a structure as the input or output parameter of a server method, for example, you must ensure that the nested array is supplied with the correct length when the method is called.

If you do not adhere to this rule, the method fails with the error code "BadInvalidArgument".

OPC_UA_ServerMethodPost: Post preparation of the server method call

Validity

The following description of the "OPC_UA_ServerMethodPost" instruction applies to S7-1500 CPUs with firmware version V2.5 and higher.

Description

This section describes the instruction "OPC_UA_ServerMethodPost".

Because the instructions "OPC_UA_ServerMethodPre" and "OPC_UA_ServerMethodPost" always have to be called up in pairs in the user program, please also take into account the section on the instruction "OPC_UA_ServerMethodPre".

Parameters for "OPC_UA_ServerMethodPost"

Table 4-94 The parameters of the instruction "OPC_UA_ServerMethodPost"

Parameter	Declaration	Data type	Meaning
Done	Output	BOOL	Status of execution: • 0: Execution of the instruction aborted, not yet complete or not yet started • 1: Execution of instruction completed without errors
Busy	Output	BOOL	Execution status parameter: • 0: Instruction not being executed • 1: Instruction currently being executed

Parameter	Declaration	Data type	Meaning
Error	Output	BOOL	Error display • 0: No error • 1: An error has occurred. See "Status" parameter.
Status	Output	DWORD	Cause of the error, see "Error codes for the status" below.
UAMethod_Result	Input	DWORD	Error codes for the OPC UA server, provided by the user program. Recommendation: Use the feedback message of error codes that begin with 0xFF. For OPC UA the following areas are defined: • Good: 0x0000_0000 to 0x3FFF_FFFF • Uncertain: 0x4000_0000 to 0x7FFF_FFFF • Bad: 0x8000_0000 to 0xFFFF_FFFF Depending on the client it is possible that the codes of the areas "Good" and "Uncertain" are not output.
UAMethod_Finished	Input	BOOL	Set the value of the parameter to TRUE if the method provided has been executed.
UAMethod_OutParameters	InOut	VARIANT	Pointer to a tag that contains the output parameters of the method provided.

Error codes for the status

The "Status" parameter provides information about errors that occurred during execution of the instruction.

The following table describes the different categories of error codes.

Table 4-95 Error codes for the status

Error code (hexadecimal values)	Explanation
0000_0000	Instruction finished successfully.
8xxx_xxxx	OPC UA specific error
Axxx_xxxx	PLCopen specific error
B080_C300	Insufficient resources
B08x_yz00	SIMATIC-specific error
For more error codes, see AUTOHOTSPOT	

Declaration of tags

Declare an instance of the instruction "OPC_UA_ServerMethodPost" and the tags with which you supply the instruction parameters, see also Example program for providing a method for OPC UA clients (Page 3833).

The following points are important for the declaration:

- Create the instruction "OPC_UA_ServerMethodPost" as a multi-instance in the calling function block.

  Note

  Name of the multiple instance

  The multiple instance has to be named "OPC_UA_ServerMethodPost_Instance", otherwise no method is created on the server.

  Use drag-and-drop to move the instruction from the folder "Instructions > Communication > OPC UA > OPC UA Server" to the editor. Next, click "Multi-instance".

- When the server method has one or more output parameters, you must declare a tag with the name "**UAMethod_OutParameters**".
  First, create a PLC data type (UDT) for the output parameters of the server method.
  Then use this UDT for the tag "UAMethod_OutParameters".
  In the example, the data type is called "UDT_OpenDoorOutArguments"; the only output parameter is Result.
  Alternative:
  You can also assign the data type "Struct" to the tag "**UAMethod_OutParameters**". Then create the components of this data type according to the output parameters of the server method (same names and data types).

OPC_UA_ServerMethodPost_Instance	OPC_UA_ServerMethodPost
Input	
UAMethod_Result	DWord
UAMethod_Finished	Bool
Output	
Done	Bool
Busy	Bool
Error	Bool
Status	DWord
InOut	
UAMethod_OutParameters	Variant
Static	
UAMethod_OutParameters	"UDT_OpenDoorOutArguments"
Result	Int

Figure 4-2 Declaration of tags

Calling the instruction

You use the "OPC_UA_ServerMethodPost" instruction to inform the operating system that the server method has been executed and that the values of the output parameters are valid.

Assignment of data types (SIMATIC – OPC UA)

For the input and output parameters of methods, please note the explanations regarding the rules on usable data types in the section "Mapping of data types".

Note

Supply of structured data types with nested arrays

If a structured data type (Struct/UDT) contains an array, the OPC UA server does not provide information about the length of this array.

If you use such a structure as the input or output parameter of a server method, for example, you must ensure that the nested array is supplied with the correct length when the method is called.

If you do not adhere to this rule, the method fails with the error code "BadInvalidArgument".

Example program for providing a method for OPC UA clients

Example program for a server method

This section includes the complete program code for the example program "OpenDoor".

The example shows you how a user program uses the instructions "OPC_UA_ServerMethodPre" and "OPC_UA_ServerMethodPost".

The instructions are described in the sections OPC_UA_ServerMethodPre: Preparing the server method call (Page 3827) and OPC_UA_ServerMethodPost: Post preparation of the server method call (Page 3830).

The program provides a server method for OPC UA clients: The program sets the output parameter "Result" to the value 1 when the input parameter "Number" has the value "1".

In order to keep the example easy and simple, a detailed error evaluation ("Status" parameter) is not included.

Program structure

The program is divided into the following sections:

1. Calling the "OPC_UA_ServerMethodPre" instruction to determine if the server method was called by a client.
2. If the server method was called, the server method is executed. It defines the actual functionality when the method is called by an OPC UA client.
3. When the server method has been completed, the instruction "OPC_UA_ServerMethodPost" is called. This section informs the operating system that the server method has been executed.

Declaration

The following figure shows the declaration of local tags for the example program:

OpenDoor

	Name	Datentyp	Offset	Defaultwert
1	Input			
2	<Hinzufügen>			
3	Output			
4	<Hinzufügen>			
5	InOut			
6	doorLocked	Bool	0.0	TRUE
7	Static			
8	UAMethod_OutParameters	"UDT_OpenDoorOutArguments"	2.0	
9	UAMethod_InParameters	"UDT_OpenDoorInArguments"	4.0	
10	Method_Result	DWord	6.0	16#0
11	Method_Finished	Bool	10.0	false
12	Started	Bool	10.1	false
13	Error_Message	WString	12.0	WSTRING#''
14	OPC_UA_ServerMethodPre_Instance	OPC_UA_ServerMethodPre		
15	OPC_UA_ServerMethodPost_Instance	OPC_UA_ServerMethodPost		
16	Temp			
17	Method_Called	Bool	0.0	
18	Pre_Done	Bool	0.1	
19	Pre_Error	Bool	0.2	
20	Post_Done	Bool	0.3	
21	Post_Error	Bool	0.4	

Program

The following program shows how you use OPC UA instructions to provide a method for OPC UA clients that is executed in your user program (server method).

Calling the "OPC_UA_ServerMethodPre" instruction

First the "OPC_UA_ServerMethodPre" instruction is called to query at the operating system as to whether the server method was called by a OPC UA client.

If the server method was called, the tag "#Method_Called" is "TRUE".

If the "OPC_UA_ServerMethodPre" instruction was also executed successfully (#Pre_Done = TRUE), the tag "#Started" is set to the value "TRUE".

Server method

When the tag "#Started" has the value "TRUE", this starts the actual user program.

You have all kinds of programming options in this section: You can read in or output process values, access global data blocks, call functions and function blocks, etc.

The application program may take several cycles.

To signal the end of the user program set the tag "#Method_Finished" to "TRUE".

With "#Method_Result" you transfer a self-defined error code that is used at the block parameter "UAMethod_Result" of the instruction "OPC_UA_ServerMethodPost".

Calling the "OPC_UA_ServerMethodPost" instruction

The tag "#Method_Finished" was set (server method) to save the state that the user program was executed.

This tag is used in the instruction "OPC_UA_ServerMethodPost" to inform the operating system if the server method was executed or not.

The operating system then takes over the acknowledgment to the client that called the method.

SCL

```
REGION PRE

    IF #Started = FALSE AND #Method_Finished = FALSE THEN

        #OPC_UA_ServerMethodPre_Instance(Done => #Pre_Done,
                                         Error => #Pre_Error,
                                         UAMethod_Called => #Method_Called,
                                        UAMethod_InParameters := #UAMethod_InParameters);
      IF #Pre_Error THEN
            #Error_Message := WString#'Error at OPC_UA_ServerMethodPre';
        END_IF;
        IF #Pre_Done AND #Method_Called THEN
            #Started := TRUE;
            #Error_Message := WString#'';
        END_IF;
    END_IF;

END_REGION

REGION FUNCTIONALITY

    // The purpose of this demo method is to demonstrate the basic structure of a server
method
    IF #Started = TRUE THEN

        // This demo server method just checks if input parameter
#UAMethod_InParameters.Number is 1
        // If this is the case, the server method sets the output parameter.
        IF #UAMethod_InParameters.Number = 1 THEN
            //set doorLocked to false, so that the door can be opened
            #doorLocked := FALSE;

            //set values for output parameters
            //this is just an example
            #UAMethod_OutParameters.Result := 1;
            #Method_Result := 16#0000_0000;
            #Method_Finished := TRUE;
            #Started := FALSE;
        ELSE
           #Error_Message := WString#'Input-Parameter Number must be 1 to run this example
server method';
            //we send back an error code to the calling OPC UA client
            //for our example, we define 16#FFFF_FFFF = "Input-Parameter Number must be 1"
            #Method_Result := 16#FFFF_FFFF;
            #Started := FALSE;
            #Method_Finished := TRUE;
        END_IF;
    END_IF;

END_REGION

REGION POST

    IF #Method_Finished = TRUE THEN
        #OPC_UA_ServerMethodPost_Instance(UAMethod_Result := #Method_Result,
```

SCL

```
                                        UAMethod_Finished := #Method_Finished,
                                        Done => #Post_Done,
                                        Error => #Post_Error,
                                    UAMethod_OutParameters := #UAMethod_OutParameters);

        IF #Post_Error = TRUE THEN
            #Error_Message := WString#'Error at OPC_UA_ServerMethodPost';
            #Method_Finished := FALSE;
        END_IF;
        IF #Post_Done = TRUE THEN
            #Method_Finished := FALSE;
        END_IF;
    END_IF;

END_REGION
```

Deleting method

If you no longer make a server method available and want to delete it, you not only have to remove the call of the corresponding FB, but also delete the instance data block.

If the instance data block is not deleted, the method is visible in the address space. The tags are not supplied.

4.1.6.4 Web server

WWW: Synchronizing user pages

WWW: Synchronizing user pages

Description

The instruction WWW initializes the web server of the CPU or synchronizes user-defined web pages, in short: "user pages", with the user program in the CPU.

Together with the Web server, user web pages make it possible for the CPU to access freely designed web pages of the CPU with a web browser.

Use script instructions (such as Javascript) and HTML code in user web pages to transfer data via a web browser for further processing to the CPU and to display data from the operand area of the CPU in the web browser. To synchronize the user program and the Web server, but also to initialize, call the instruction "WWW" in the user program.

Initialization

User web pages are "packaged" in data blocks for processing by the CPU. You will have to generate appropriate data blocks from the source files (HTML files, screens, Javascript files, ...) during configuration. The Web Control DB takes on a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. The data blocks with coded web pages are called fragment DBs.

When the data blocks are downloaded into the CPU, the CPU does not "know" that user web pages are coded inside it. The instruction "WWW" in the startup OB, for example, will inform the CPU which DB is the Web Control DB. After this initialization, the user pages are accessible via a web browser.

Synchronization

If you want the user program to interact with the user pages, then the instruction "WWW" must be used in the cyclical program part.

Examples of interaction between user program and web page:

- Check received data
- Assemble and send back data to the web browser making the request

In this case it must be possible to evaluate the current status information and the Web server must receive control information, such as release of a web page requested by a web browser.

Parameter

The following table shows the parameters of the "WWW" instruction:

Parameter	Declaration	Data type	Memory area	Description
CTRL_DB	Input	DB_WWW	I, Q, M, D, L or constant	Data block that describes the user pages (Web control DB)
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to Overview of the valid data types (Page 231).

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred. There are no web page requests that have to be released by the user program.
00xy	x: indicates whether an error has occurred during initialization of the Web Control DB (CTRL_DB): x=0: No errors occurred. x=1: Error occurred. The error is coded in the byte "CTRL_DB.last_error" of the Web Control DB, see description of Web Control DB. y: Number of the pending request. Several requests are possible (e. g. requests "0" and "1" are pending: y="3". y="1": Request "0" y="2": Request "1" y="4": Request "2" y="8": Request "3"
803A	The specified Web Control DB does not exist on the CPU.
8081	Incorrect version or incorrect format of the Web Control DB.
80C1	There are no resources to initialize the web application, for example, because only two or four web applications may be running.

Example

You will find the example here: Program example of WWW (Page 3840).

You can find additional information and the program code for the example here: Sample Library for Instructions (Page 593).

See also

Overview of the valid data types (Page 231)

Program example of WWW

Introduction

In the following example, you synchronize a user-defined Website with the program example in a CPU S7-1500 and test the program example via a Web server.

Requirement

Create nineteen tags in a global data block for storing the data.

SLI_gDB_www			
	Name	Data type	Start value
1	Static		
2	executeWWW	Bool	false
3	returnValueWWW	Word	16#0
4	executeSetOB	Bool	false
5	setDone	Bool	false
6	start	Bool	false
7	stop	Bool	false
8	engineStarted	Bool	false
9	reset	Bool	false
10	tempReset	Bool	false
11	openValve	Bool	false
12	closeValve	Bool	false
13	valveOutput	Bool	false
14	flowrate	Int	1
15	tankLevel	Int	0
16	alarm	Int	0
17	tankLevelOverflow	Int	160
18	tankLevelMax	Int	150
19	tankLevelMidth	Int	75
20	tankLevelMin	Int	5

Interconnect parameters - in FB “SLI_FB_www“

You create the following interconnections in an FB "SLI_FB_www": You call the FB in an OB1.

Network 1: Interconnect the parameters of the "WWW" instruction as follows.

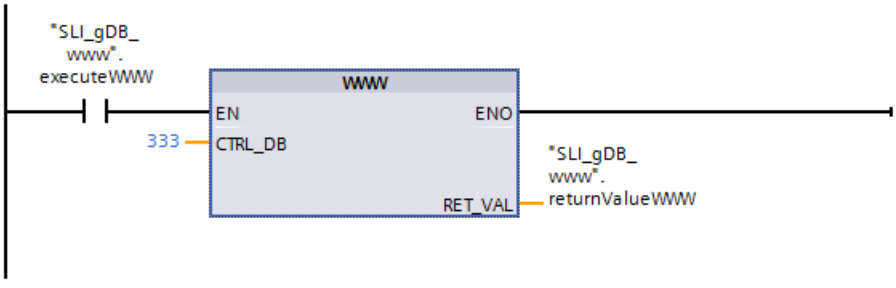

Network 2: You create the program for the Website in the FC “SLI_FC_MainWebProg_www“. You then call the FC “SLI_FC_MainWebProg_www“ in network 2 of the FB.

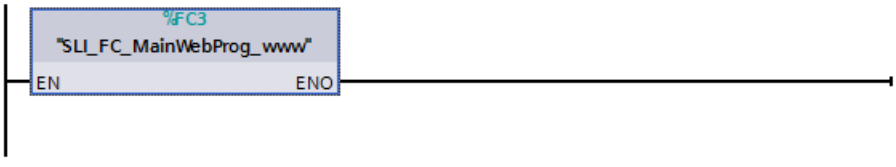

Interconnecting parameters - in FC "SLI_FC_SetTOD_www"

In an FC “SLI_FC_SetTOD_www“ you create a program for using a time-of-day interrupt OB (OB11). You create the following interconnections:

Network 1: In the FC you interconnect the following local tags with the instructions “RD_SYS_T“ and “T_ADD“.

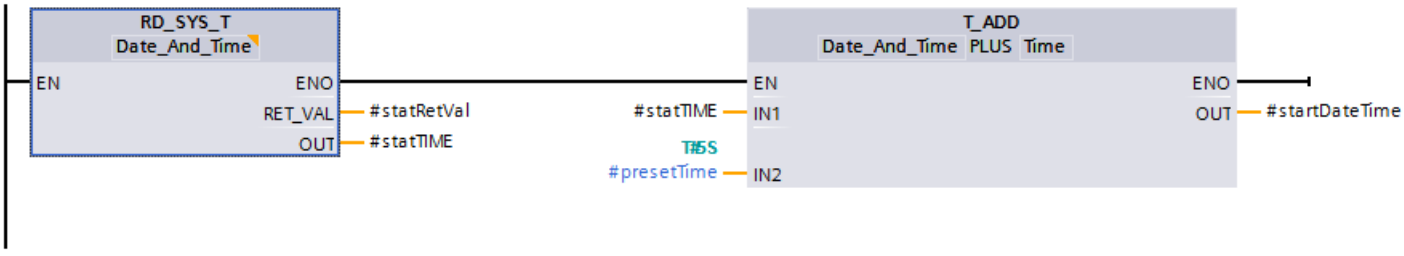

Network 2: This is followed by interconnecting the tags for setting the time-of-day interrupt OB.

The local tag “#startDateTime“ (data type “Date_And_Time”) transfers the start time to the time-of-day interrupt OB.

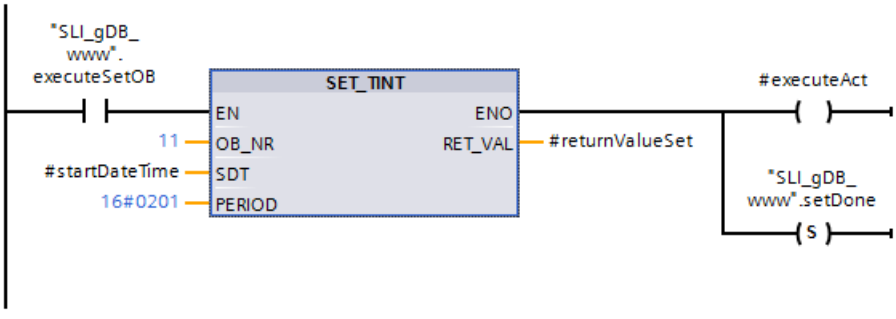

Network 3: This is followed by interconnecting the tags for activating the time-of-day interrupt OB.

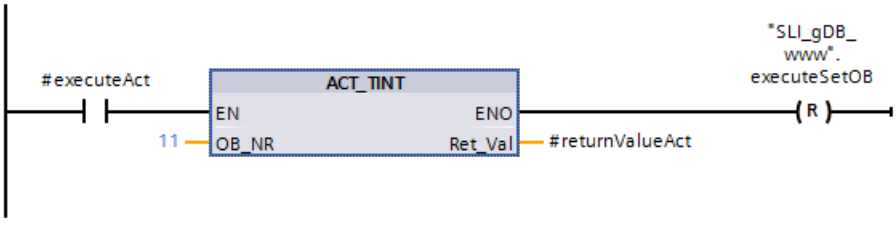

Interconnecting parameters - in the OB “SLI_todOB_www“

In the time-of-day interrupt OB “SLI_todOB_www“ (OB11) you create the following interconnections for cyclic increasing of the tag “tankLevel“.

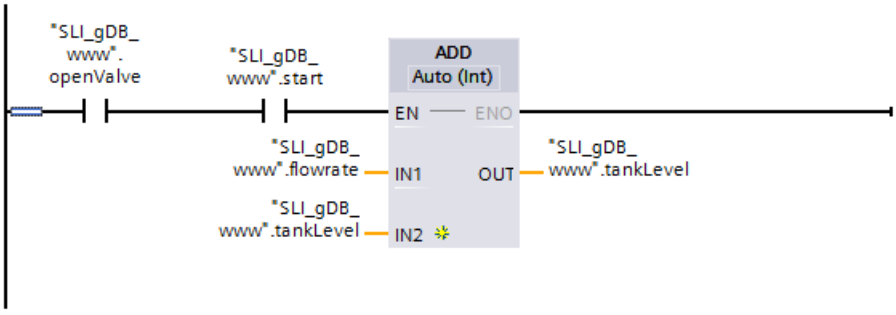

Interconnecting parameters - in FC "SLI_FC_MainWebProg_www"

You create the following interconnections in the FC "SLI_FC_MainWebProg_www“.

The program “SLI_FC_MainWebProg_www“ allows control of a motor, a valve and the tank filling. In addition to this the output of various interrupt statuses is made possible. The program is controlled via the Web server.

Network 1: You call the FC “SLI_FC_SetTOD_www“ in the FC “SLI_FC_MainWebProg_www“.

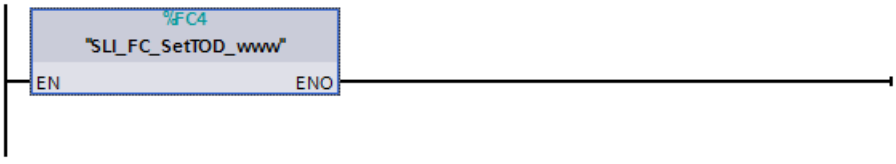

Network 2: This is followed by interconnecting the tags for opening the valve.

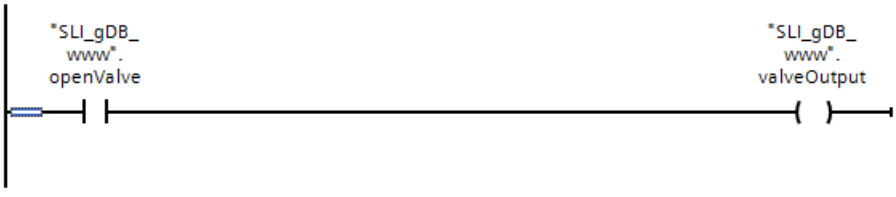

Network 3: This is followed by interconnecting the tags for starting the motor.

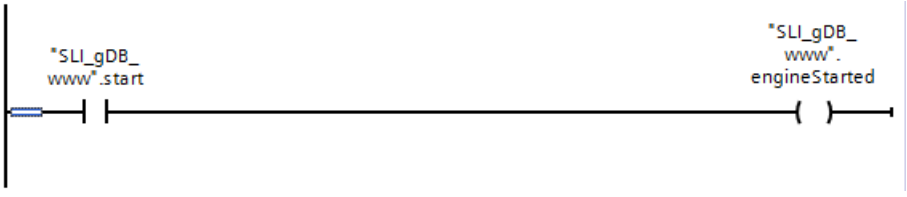

Network 4: This is followed by interconnecting the tags for querying the interrupt status “1“.

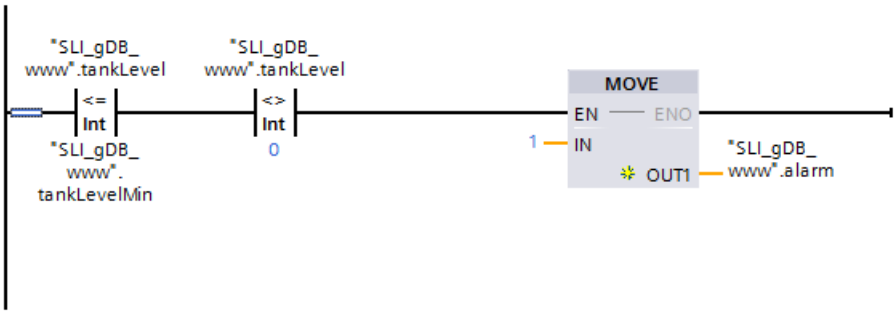

Networks 5 to 8: This is followed by interconnecting the tags for querying the interrupt status “2“.

Interconnect the tags for querying the other interrupt statuses in the same way.

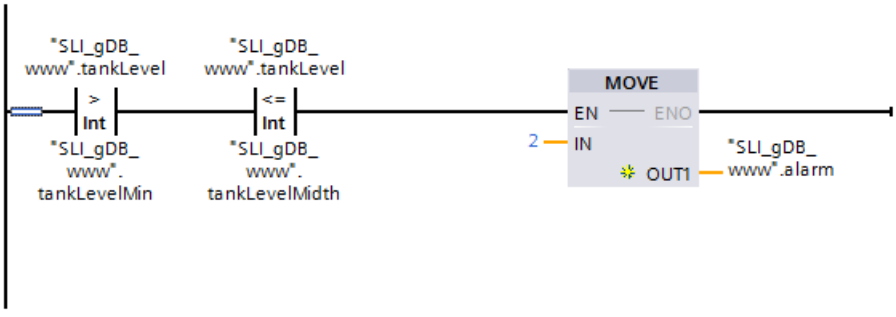

Network 9: This is followed by interconnecting the tags for resetting the the motor and the filling of the tank..

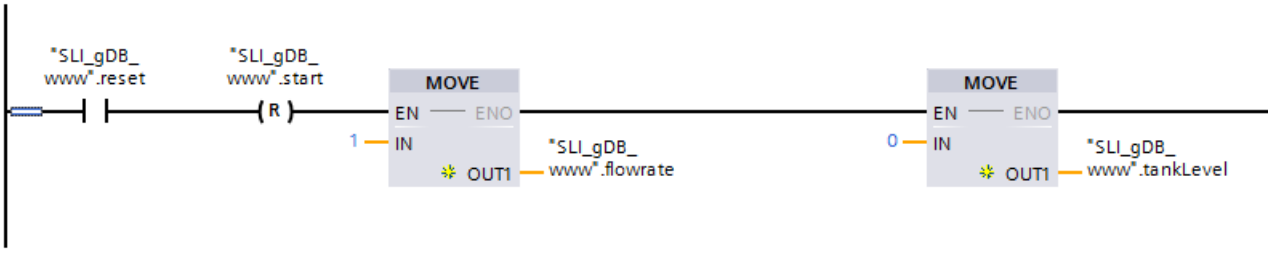

Network 10: This is followed by interconnecting the tags for resetting all the other values (BOOL).

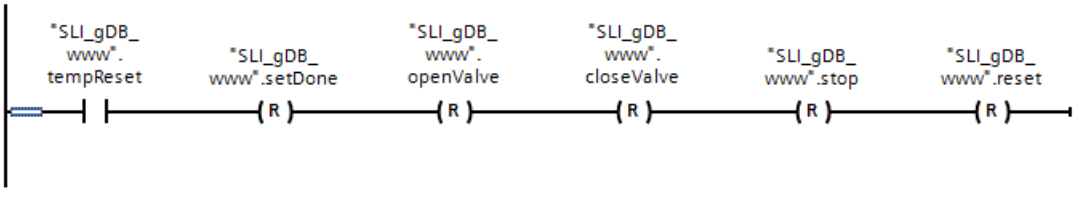

Setting CPU properties

Make the following settings in the CPU properties:

- Activate web server access under "... > PROFINET interface > Access to the web server".
- In “CPU properties > Web server enable the Web server for the module,
- In the section “User administration“ create a new user with read and write rights.
- Make the following settings in the section "User pages":
  - You set the “HTML file path”.

    Note

    The storage path should be the same as the path of the Sample Library for Instructions (SLI). The folder name is “SLI_html“. For example: “C:\TIA_library\SLI_html“.

  - Enter the text "index.html" for the Start HTML page.
  - For the application you enter the name “WWW sample“.
  - Click the "Create block" button. The Web DB (333) and the Fragment DB (334) are then created.

Result for WWW

If the normally open contact ("executeWWW") supplies the signal state "TRUE", the "WWW" instruction is executed. The number of the data block that describes user defined Website is stored via the the input parameter CTRL_DB ("333"). The “WWW“ instruction initializes the user defined Website in the Web server of the CPU and immediately synchronizes the program example.

The output parameter RET_VAL ("returnValueWWW") indicates that the processing is running without errors.

SLI_gDB_www

	Name	Data type	Start value	Monitor value
1	Static			
2	executeWWW	Bool	false	TRUE
3	returnValueWWW	Word	16#0	16#0000
4	executeSetOB	Bool	false	FALSE
5	setDone	Bool	false	TRUE
6	start	Bool	false	TRUE
7	stop	Bool	false	FALSE
8	engineStarted	Bool	false	TRUE
9	reset	Bool	false	FALSE
10	tempReset	Bool	false	FALSE
11	openValve	Bool	false	TRUE
12	closeValve	Bool	false	FALSE
13	valveOutput	Bool	false	TRUE
14	flowrate	Int	1	3
15	tankLevel	Int	0	3
16	alarm	Int	0	1
17	tankLevelOverflow	Int	160	160
18	tankLevelMax	Int	150	150
19	tankLevelMidth	Int	75	75
20	tankLevelMin	Int	5	5

Result for the Web server

You call the Web server via the “Internet browser > IP of the Web server”. You log in with your user name user name and -password.

In the Web server in “User pages” a link to the website defined by the user is displayed.

Customer pages

Homepage of the application WWW sample:

On the Website of the program example it is possible for you to output tags and transfer new values to the tags.

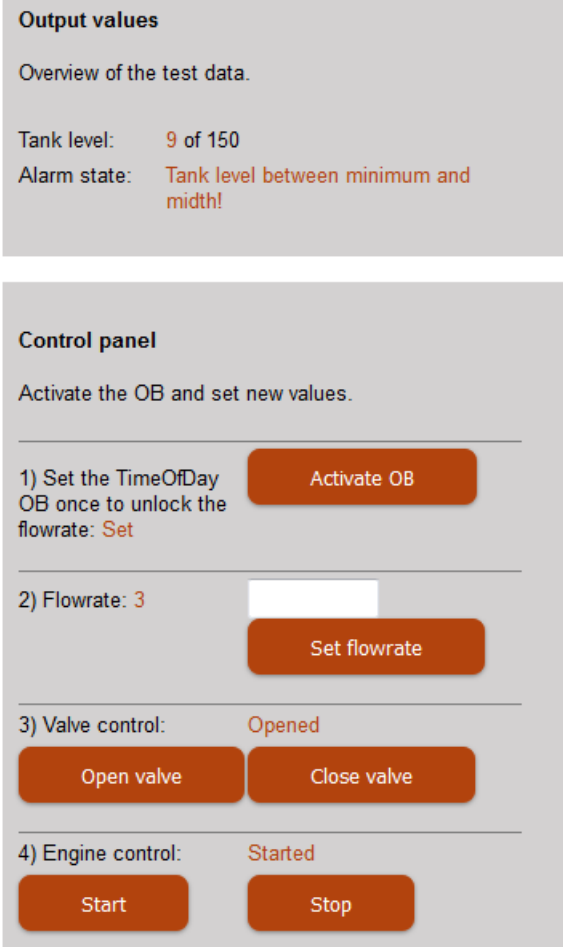

Functions used in the HTML document

Below you will see excerpts of the functions used in the program example with which the tags are transferred to the source code in the HTML document. For each function a tag is implemented via a call in the HTML document and at the relevant location in the HTML document is used with a further call.

Function	To call in the HTML document	To output in the HTML element
Data output:	`<!-- AWP_In_Variable Name='"SLI_gDB_www".tankLevel' -->`	`:="SLI_gDB_www".tankLevel:`
Enumeration (value replacement):	`<!-- AWP_Enum_Def Name="OpValvValue" Values='0:"Closed", 1:"Opened"' -->`	`<!-- AWP_Enum_Ref Name='"SLI_gDB_www".valveOutput' Enum="OpValvValue" -->:="SLI_gDB_www".valveOutput:`

Function	To call in the HTML document	To output in the HTML element
Value change (from INT):	<!-- AWP_In_Variable Name='"SLI_gDB_www".flowrate' -->	<form method="post" action="" onsubmit="return check();"> <input type="text" name='"SLI_gDB_www".flowrate' size="10px"/> <input class="button1" type="submit" value="Set flowrate"/> </form>
Value change (from BOOL):	<!-- AWP_Enum_Def Name="ResetValue" Values='0:"Off", 1:"On"' -->	<form method="post" action=""> <input class="button1" type="submit" value="Reset"/> <input type="hidden" name='"SLI_gDB_www".reset' size="34px" value="1"/> </form>

Program code

You can find additional information and the program code for the above-named example here: Sample Library for Instructions (Page 593).

4.1.6.5 Communications processor

SIMATIC NET CM/CP

S7-1500 CM/CP

Industrial Ethernet

Instructions for FTP services

FTP_CMD for FTP services

Overview of FTP_CMD

Meaning

Using the FTP_CMD instruction, you can establish FTP connections and transfer files from and to an FTP server.

Note

Block versions

You can use the version V2.x of FTP_CMD in a station only in conjunction with a CPU and a CP V2.x V2.x.

As soon as the station obtains a CPU V1.x or CP V1.x, you must use FTP_CMD in the older version V1.x (e.g. V1.4). To do this, change the version of the "SIMATIC NET CP" library to V3.4. You can then select an older version of the block.

Data transfer is possible using FTP or FTPS (secure SSL connections).

Note

FTPS: Comparing certificates

FTPS requires a comparison of the certificates between FTP server and FTP client. If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server. Import the certificate of the FTP server as a trusted certificate in the certificate manager.

How it works

The FTP_CMD instruction references a job block (ARG) in which the FTP command is specified. Depending on the type of FTP command (CMD), this job block uses different data structures for parameter assignment. Suitable data types (UDTs) are available for these various structures.

The following diagram shows the call structure:

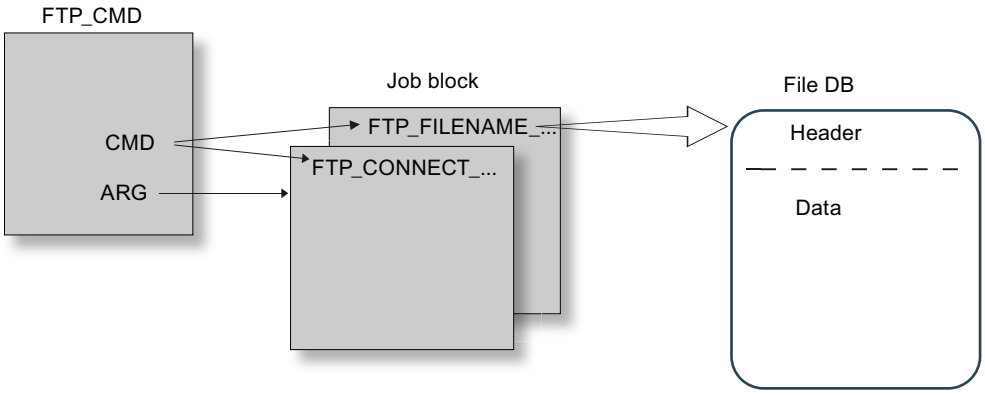

Job blocks

The following data structures are used for the job blocks:

- Connection establishment
  Various data structures are available for the connection establishment using the following types of access:
  - FTP_CONNECT_IPV4: Connection establishment with IP addresses according to IPv4
  - FTP_CONNECT_IPV6: Connection establishment with IP addresses according to IPv6
  - FTP_CONNECT_NAME: Connection establishment with server name (DNS)
- Data transfer
  For the data transfer, two different data structures are available:
  - FTP_FILENAME: Data structure for access to a complete file
  - FTP_FILENAME_PART: Data structure for read access to a data area

Data transfer in the File_DB

The data transfer is achieved using data blocks containing a header for job data and the area for the user data. The data block is specified in the job buffer.

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- For all data blocks being used as file DBs, disable the "Optimized block access" attribute.
- Only when using a CPU V1.x and a CP V1.1.x:
  Disable the "Disable PUT/GET communication" option in the configuration data of the CPU under "Properties > General > Protection" (PUT/GET must be released).

Validity

The FTP_CMD instruction can be used with the following module types:

- CP 1543-1

Call interface

Call interface in FBD representation

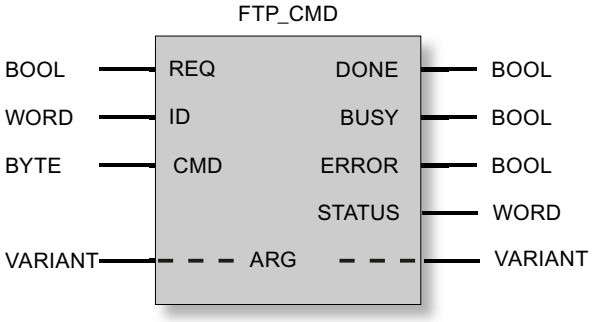

See also

Output parameters and status information FTP_CMD (Page 3858)

Input parameter - FTP_CMD (Page 3852)

Structure of the data blocks (file DBs) for FTP services - FTP client mode (Page 3862)

Input parameter - FTP_CMD

Explanation of the input parameters

You supply the FTP_CMD instruction with the following input parameters:

Table 4-96 Formal parameters of the FTP_CMD instruction - input parameters

Parameter	Declaration	Data type	Memory area	Meaning / remarks
REQ	Input	BOOL	I, Q, M, D, L	Starts the send job on a rising edge.
ID *	INPUT	INT	1, 2 ... 64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
CMD *	INPUT	BYTE	See following table "Commands".	FTP command to be executed when the instruction is called. You will find value ranges for the FTP command types after the table. Note: The FTP command specified here must be specified identically in the job block (ARG parameter). If a command is not supported by the CP firmware, an error message with STATUS = $8F6B_H$ is output.
ARG *	INPUT	VARIANT	See following table "Commands".	Job block References the data area with the execution parameters suitable for the FTP command. Specific data types (UDT) are used depending on the FTP command. The UDTs are shown below. The ANY data type is not permitted for the pointer to be specified here!

* The values of the input parameters "ID" and "CMD" overwrite the value of the input parameter "ARG".

FTP commands in the "CMD" parameter

The following table shows you the significance of the commands of the "CMD" parameter and which UDTs you use to supply the job blocks.

Table 4-97 Command types

CMD (command type)	Relevant job blocks / UDT	Meaning / handling
0 (NOOP)	*	The called FC does not execute any action. The status codes are set as follows when these parameters are supplied: DONE=1; ERROR=0; STATUS=0
1 (CONNECT)	FTP_CONNECT_IPV4 FTP_CONNECT_IPV6 FTP_CONNECT_NAME	FTP connection establishment With this command, the FTP client establishes an FTP connection to an FTP server (port 21). The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user.
2 (STORE)	FTP_FILENAME	This function call transfers a data block (file DB) from the FTP client (S7-CPU) to the FTP server. Caution: If the file (file DB) already exists on the FTP server, it will be overwritten.
3 (RETRIEVE)	FTP_FILENAME	This function call transfers a file from the FTP server to the FTP client (S7-CPU). Caution: If the data block (file DB) on the FTP client already contains a file, it will be overwritten.
4 (DELETE)	FTP_FILENAME	With this function call, you delete a file on the FTP server.
5 (QUIT)	*	With this function call, you establish the FTP connections selected with the ID.
6 (APPEND)	FTP_FILENAME	Similar to "STORE", the "APPEND" command saves a file on the FTP server. With "APPEND", the file on the FTP server is, however, not overwritten. The new content is appended to the existing file. If the file (file DB) does not exist on the FTP server, it will be created.
7 (RETR_PART)	FTP_FILENAME_PART	Using the "RETR_PART" command (retrieve part) , you can request a section of a file from the FTP server. If very large files are involved, this allows you to restrict the read to the part you currently require. To do this, you need to know the structure of the file. Enter the required part of the file using the two parameters "OFFSET" and "LEN" in FB40.

* With the command types 0 (NOOP) and 5 (QUIT) a freely selectable job block (UDT) must be specified. This is not evaluated.

See also

Overview of FTP_CMD (Page 3849)

Job blocks for FTP_CMD

Meaning

You supply the FTP_CMD instruction with a job block using the ARG parameter. The structure depends on the FTP command type. By using the default data types (UDT), the instruction recognizes the type of the job block. Below, you will find the relevant data types (UDTs) for the following job blocks:

- FTP connection establishment with IP address according to IPv4
- FTP connection establishment with IP address according to IPv6
- FTP connection establishment with server name
- Write and read access and other FTP commands
- FTP command RETR_PART

Job block for FTP connection establishment with IP address according to IPv4

For FTP connection establishment with IP address according to IPv4, the following data structure is used.

Table 4-98 FTP_CONNECT_IPV4

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPCmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You can find the value ranges for the command types in the section Input parameter - FTP_CMD (Page 3852). Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	'benutzer'	User name for the login on the FTP server

Parameter	Type	Range of values	Meaning / remarks
Password	STRING[32]	'passwort'	Password for the login on the FTP server
FTPserverIPaddr	IP_V4	ADDR(1) ... ADDR(4)	IP address of the FTP server as Array[1..4] of Byte, where 1 byte specifies one block of the address. Example: ADDR(1) specifies the first address block (the first byte of the address).

Job block for FTP connection establishment with IP address according to IPv6

For FTP connection establishment with IP address according to IPv6, the following data structure is used.

Table 4-99 FTP_CONNECT_IPV6

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPCmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You can find the value ranges for the command types in the section Input parameter - FTP_CMD (Page 3852). Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	'user'	User name for the login on the FTP server
Password	STRING[32]	'password'	Password for the login on the FTP server
FTPserverIPaddr	IP_V6	ADDR(1) ... ADDR(16)	IP address of the FTP server as Array[1..16] of Byte, where 2 bytes specify one block of the address. Example: ADDR(1) + ADDR(2) specify the first address block.

Job block for FTP connection establishment with server name

For FTP connection establishment specifying the server name, the following data structure is used. The server name is assigned to an IP address using DNS.

Table 4-100 FTP_CONNECT_NAME

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPcmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You can find the value ranges for the command types in the section Input parameter - FTP_CMD (Page 3852). Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	’benutzer’	User name for the login on the FTP server
Password	STRING[32]	’passwort’	Password for the login on the FTP server
FTPserverName	STRING[254]		IP address of the FTP server

Job block for write and read access and other FTP commands

The following data structure is used for the FTP commands store, retrieve, delete and append.

Table 4-101 FTP_FILENAME

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	

Parameter	Type	Range of values	Meaning / remarks
FTPcmd	BYTE	2, 3, 4, 6	FTP command "STORE / RETRIEVE / DELETE / APPEND" FTP command that executes when the instruction is called. You can find the value ranges for the command types in the section Input parameter - FTP_CMD (Page 3852). Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
DataBlockNumber	UINT		The data block specified here contains the file DB to be read / written.
LenFilename	UINT	0...1000	The "LenFilename" parameter for specifying the total length of the file name is not evaluated. Instead the length information in the string of the "Filename" parameter is evaluated.
Filename	ARRAY[0..3] OF STRING[220]		File name of the destination or source file.

Job block for the RETR_PART FTP command

The following data structure is used for the RETR_PART FTP command.

Table 4-102 FTP_FILENAME_PART

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPcmd	BYTE	7	FTP command "RETR_PART" FTP command that executes when the instruction is called. You can find the value ranges for the command types in the section Input parameter - FTP_CMD (Page 3852). Note: The FTP command specified here must be specified identically in the CMD input parameter.

Parameter	Type	Range of values	Meaning / remarks
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
Offset	DWORD		Offset in bytes starting at which the file will be read.
Length	DWORD		Sublength in bytes that is read starting at the value specified in "OFFSET". Special features: • If "DW#16#FFFFFFFF" is specified, the available rest of the file will be read. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET > length of the original file: Length of the destination file (ACT_LENGTH in file DB): 0 bytes on the CPU. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET + LEN > length of the original file (and LEN ≠ 0xFFFFFFFF): Length of the destination file (ACT_LENGTH in file DB): Available bytes starting at "OFFSET". Result OK (DONE = 1, STATUS = 0) if no other error occurred.
DataBlockNumber	UINT		The data block specified here contains the file DB to be read / written.
LenFilename	UINT	0...1000	The "LenFilename" parameter for specifying the total length of the file name is not evaluated. Instead the length information in the string of the "Filename" parameter is evaluated.
Filename	ARRAY[0..3] OF STRING[220]		File name of the destination or source file.

See also

Overview of FTP_CMD (Page 3849)

Output parameters and status information FTP_CMD

Parameters BUSY, DONE and ERROR

You control the execution status with the parameters BUSY, DONE, ERROR and STATUS. The BUSY parameter indicates the processing status. With the DONE parameter, you check whether or not a job was correctly executed. The ERROR parameter is set if errors occur during execution of "FTP_CMD". Error information is output in the STATUS parameter.

The following table shows the relationship between the parameters BUSY, DONE and ERROR:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was terminated with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Evaluating status codes

Note

For entries coded with 8FxxH in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 4-103 FTP_CMD: Meaning of the STATUS parameter in conjunction with DONE and ERROR

DONE	ERROR	STATUS	Meaning
0	0	0000_H	No job is being executed.
1	0	0000_H	the job was completed without error.
0	0	7001_H	The job was triggered the first time (BUSY=1)
0	0	7002_H	The job is still being executed (BUSY=1)
0	0	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).
0	0	8183_H	The configuration does not match the job parameters.
0	1	8401_H	Unknown error Possible causes: • A timeout was detected on the connection. • The FTP server has aborted the connection. Remedy: • Send the QUIT and CONNECT commands again to re-establish the connection.
0	1	8402_H	The connection has an error status. The timeout of the connection may have been exceeded or the FTP server may have terminated the connection. Remedy: Send the QUIT and CONNECT commands to re-establish the connection.
0	1	8403_H	Login has failed.
0	1	8404_H	FTP server is not obtainable.
0	1	8405_H	Transfer has failed.

DONE	ERROR	STATUS	Meaning
0	1	8406_H	Timeout for current action
0	1	8407_H	File was not found on the FTP server.
0	1	8408_H	Transfer not possible.
0	1	8409_H	File could not be fetched.
0	1	8410_H	Setting the TCP port for the data connection has failed.
0	1	8411_H	Offset information does not match.
0	1	8412_H	Error changing the specified directory
0	1	8413_H	Error receiving data
0	1	8414_H	Error sending data
0	1	8415_H	Specified CMD (command type) was rejected by the client.
0	1	8416_H	Connection was closed by the FTP server.
0	1	8418_H	Error in the user data. Possible causes: • File name is empty. • Data length is "0" • etc.
0	1	8419_H	There is no socket resource for opening a data connection.
0	1	8420_H	There is no socket resource for opening a control connection.
0	1	8421_H	Error opening the file DB to be read.
0	1	8422_H	Error opening the file DB to be written.
0	1	8423_H	Connection establishment to the FTP server has failed.
0	1	8424_H	Internal error
0	1	8425_H	Format error in the domain name
0	1	8426_H	There are too many DNS queries pending.
0	1	8427_H	The specified DNS server could not assign the specified domain name.
0	1	8428_H	There is no connection resource available.
0	1	8429_H	Unknown channel ID
0	1	8430_H	The file DB is too short.
0	1	8431_H	Error when writing to the file DB.
0	1	8432_H	Error when reading from the file DB.
0	1	8433_H	Error when accessing the file DB.
0	1	8434_H	Action was aborted.
0	1	8435_H	Channel will be reset.
0	1	8436_H	Unexpected server reply
0	1	8437_H	Certificate could not be verified.
0	1	8438_H	Unknown error occurred
0	1	8439_H	The FTP command causes an error. The cause must be looked for on the FTP server (REST command).
0	1	8440_H	The FTP server does not support the requested SSL protocol.
0	1	8446_H	After the FTP password was sent to the FTP server, an unexpected code was returned by the FTP server.
0	1	8451_H	An error was signaled when attempting to change the transmission mode from binary to ASCII.

DONE	ERROR	STATUS	Meaning
0	1	8455_H	A memory request has failed on the CM/CP.
0	1	8460_H	A problem has occurred handling SSL/TLS.
0	1	8469_H	Interface error The specified output interface could not be used. Remedy: Set the interface to be used for outgoing connections.
0	1	8475_H	The SSL certificate or the SSH md5 fingerprint was not considered trusted.
0	1	8476_H	Nothing was received from the FTP server. In the current status, an incorrect response must be assumed.
0	1	8477_H	The specified "Crypto engine" (cryptographic module) was not found.
0	1	8478_H	The attempt to set the selected SSL "Crypto engine" as the default failed.
0	1	8480_H	A problem has occurred with the certificate of the FTP client.
0	1	8481_H	The specified number could not be used.
0	1	8482_H	The FTP server uses a coding that is not supported.
0	1	8484_H	The maximum file size was exceeded.
0	1	8485_H	The file DB was modified while being processed to be sent or the file DB is incorrectly structured.
0	1	8489_H	Data could not be sent. There is not enough memory available for the action on the FTP server.
0	1	8492_H	The file already exists. The file will not be overwritten.
0	1	8496_H	A problem occurred reading the SSL CA certificate.
0	1	8497_H	An unexpected error occurred in the SSH session.
0	1	8498_H	It was not possible to terminate the SSL connection.
0	1	8499_H	The socket is not ready for sending/receiving. Wait until it is ready and try again.
0	1	8501_H	The SSL certificate check by the FTP server has failed.
0	1	8507_H	A timeout has occurred establishing the connection during the active FTP session while waiting for the FTP server.
0	1	$8F55_H$	Header status bit: Locked
0	1	$8F56_H$	The NEW bit in the file DB header was not reset
0	1	$8F6B_H$	Possible causes: • Bad value for the CMD parameter Values from 0 to 15 are permitted. • An FB40 command is not supported. Possible cause: Wrong firmware on the CP Remedy: Firmware update (with older CPs, use the functions FC 40...FC 44 instead of FB 40.)
0	1	$8F7F_H$	Internal error, for example illegal ANY reference

See also

Overview of FTP_CMD (Page 3849)

Structure and use of the file DB - FTP client mode

Structure of the data blocks (file DBs) for FTP services - FTP client mode

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data as "Array [..] of Byte" or "Array [..] of Char" (has a variable length and structure)

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.
- For all data blocks being used as file DBs, disable the "Optimized block access" attribute.

File DB header for FTP client mode

The file DB header described here is identical to the file DB header described for server mode.

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data (file does not exist). • 1: The file DB contains valid user data (file exists).	The DELETE FTP command sets EXIST=0. The STORE FTP command sets EXIST=1.
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The "STORE" and "RETRIEVE" FTP commands set LOCKED=1 when they are executed if the bit was previously at 0. The user program on the S7 CPU can also set or reset LOCKED during write access to achieve data consistency. This results in mutual locking between the user program and FTP handling to ensure consistency. Recommended sequence in the user program: 1. Check LOCKED bit (if = 0) 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit (if = 0) 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program in the S7-CPU must set NEW=0 to allow a new "RETRIEVE" command.
WRITE_ACCESS	BOOL	• 0: The user program has write access rights for the file DBs on the S7 CPU. • 1: The user program has no write access rights for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.

Parameter	Type	Value / meaning	Supply
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	Unsigned integer (16-bit) containing the last reply code from FTP as a binary value. The content of this field is only valid when EXIST = 1.	Updated by the FTP protocol handler with the FTP command processing of the server.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. The reference is the startup time of the CP (initialization value: 01.01.1994 00:00h).

See also

FILE_DB_HEADER data block as template - FTP client mode (Page 3864)

FILE_DB_HEADER data block as template - FTP client mode

Meaning

The file DB header described here is identical for the FTP client mode and FTP server mode.

The data type FILE_DB_HEADER is predefined for creating a file DB header.

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)

Follow the steps outlined below:

1. In the CPU on which you create the user program with the FTP instructions, create a data block of the type "Global DB".
2. Open the properties dialog of the newly created data block (shortcut menu), disable the "Optimized block access" attribute and then close the properties dialog.
3. In the block editor of the DB, select the line you want to use as the start line for the file DB.
4. From the drop-down list in the "Data type" column, select a structure element of the type "FILE_DB_HEADER" or enter the type "FILE_DB_HEADER" using the keyboard.
   Result: A data structure with the header structure required for the file DB is created.
5. Set the "WRITE_ACCESS" parameter to "true" to enable access.
6. Enter a value for the length of the user data at the "MAX_LENGTH" parameter, see next item.
7. Below this, create a data field of the type "Array [..] of Byte" or "Array [..] of Char" for the user data to be sent.
   The size of the field must match the specification of MAX_LENGTH in the header.

Note

"Add new block" function - type selection

When you create new data blocks, the "FILE_DB_HEADER" block type is also available under the "Type" entry in the drop-down list. Do not use this option! The data block created in this way only contains the header structure and cannot be expanded by the area required for storing user data.

FILE_DB_HEADER data block - example and template for the file DB header

In the declaration view, you will see the following structure:

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	bit08	BOOL	FALSE	reserved
+0.1	bit09	BOOL	FALSE	reserved
+0.2	bit10	BOOL	FALSE	reserved
+0.3	bit11	BOOL	FALSE	reserved
+0.4	bit12	BOOL	FALSE	reserved
+0.5	bit13	BOOL	FALSE	reserved
+0.6	bit14	BOOL	FALSE	reserved
+0.7	bit15	BOOL	FALSE	reserved
+1.0	EXIST	BOOL	FALSE	if TRUE: File DB does not contain valid data
+1.1	LOCKED	BOOL	FALSE	if TRUE: The file DB is blocked due to a change to the content.

Address	Name	Type	Initial value	Comment
+1.2	NEW	BOOL	FALSE	if TRUE: The content of the File DB is new and must not be overwritten.
+1.3	WRITE_ACCESS	BOOL	FALSE	if TRUE: The FTP server has write access.
+1.4	bit04	BOOL	FALSE	reserved
+1.5	bit05	BOOL	FALSE	reserved
+1.6	bit06	BOOL	FALSE	reserved
+1.7	bit07	BOOL	FALSE	reserved
+2.0	ACT_LENGTH	DINT	L#0	Current length of the content in bytes (not including 20 bytes for the header)
+6.0	MAX_LENGTH	DINT	L#0	Maximum length of the content in bytes (not including 20 bytes for the header)
+10.0	FTP_REPLY_CODE	INT	0	Updated by the FTP protocol handler with the FTP command processing, last response information from the FTP server.
+12.0	DATE_TIME	DATE_AND_TIME	DT#00-1-1-0:0:0.000	Date and time of the last change to the content of the file DB.
=20.0		END_STRUCT		

S7-1200 CM/CP

Telecontrol

Telecontrol instructions

TC_CON: Establish connection via the GSM network

Meaning

The TC_CON instruction allows an S7-1200 with telecontrol CP (e.g. CP 1242-7 - 6GK7242-7KX30-0XE0) to establish connections of the following types:

- ISO-ON-TCP
  Connection partner is a CP in an S7 station.
  ISO-ON-TCP connection are only used in "GPRS direct" mode with the CP 1242-7.
- UDP
  Any connection partner is possible.

- SMS
  The connection partner is an SMS client.
- Telecontrol connection (protocol: TeleControl Basic)
  The connection partner is either the telecontrol server or another station that can be reached via the telecontrol server.

A TC_CON establishes exactly one connection. Depending on the mode of the CP 1242-7 and the protocol you are using, a maximum of 3 to 5 simultaneous connections with unique IDs (see below) are supported per CP. You will find the maximum number of simultaneous connections in the performance data of the CP.

The CONNECT parameter uses a data block (DB) with the structure of a system data type (SDT) for the connection description.

The required connection type is defined using a connection-specific SDT "TCON_..." (see below). For each of the connection types listed above, one of the following SDTs must be assigned:

- TCON_IP_RFC for ISO-ON-TCP connections
- TCON_IP_V4 for UDP connections
- TCON_PHONE for SMS connections
- TCON_WDC for telecontrol connections

The "ActiveEstablished" parameter of these SDTs also specifies whether or not connection establishment is active or passive.

For parameter settings for these SDTs, see TCON_...: SDTs for the telecontrol connection establishment (Page 3883).

The ID parameter references the connection. The ID is assigned and must be unique within the CPU.

The INTERFACE parameter references the interface of the required local CP. This must be taken from STEP 7.

Call interface in FBD representation

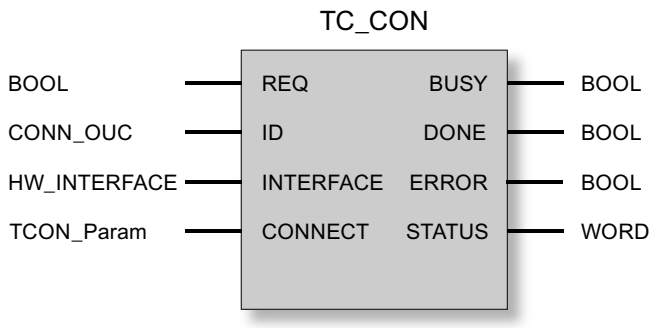

Explanation of the formal parameters

The following table explains all the formal parameters for the TC_CON instruction.

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection. The ID is assigned. The value of ID is also required by the system data type (SDT) of the CONNECT parameter.
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP (Hardware identifier) *
CONNECT	INOUT	TCON_Param	See also "TCON_...: SDTs for telecontrol connection establishment"	Reference to a data block for connection establishment. The SDTs of the type TCON_IP_RFC, TCON_IP_V4, TCON_PHONE or TCON_WDC specify the structure of the data block suitable for the relevant connection. In the SDTs, note the parameter "ActiveEstablished" (active / passive connection establishment).
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	The status parameter indicates whether the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

* Depending on the CP type, the "hardware identifier" of either the telecontrol interface or the Ethernet interface must be referenced. After selecting the CP in the STEP 7 network or device view, you can find the "Hardware identifier" at the following location in the Inspector window:

- Up to STEP 7 V14 SP1
  Tab "Properties" > "General" > "Relevant interface" > Parameter group "Hardware identifier"
- As of STEP 7 V15
  Tab "Properties" > "System constants" > "Hardware identifier" (system constant of data type "Hw_Interface")
  Depending on the CP type, the following system constant is relevant:
  - CP 1242-7 (6GK7242-7KX30-0XE0)
    "Local~<CP name>~<Name of the telecontrol interface>"
  - All other telecontrol CPs
    "Local~<CP name>~<Name of the Ethernet interface>"

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

When called, the instruction remains in the BUSY = 1 state for several seconds. In the following situations, the BUSY state = 1 can last for a longer time:

- On active ISO-on-TCP connections if the partner cannot be reached.
- On passive connections when no frame is received.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	8087_H	Maximum number of connections reached, no further connection possible
0	1	$80E3_H$	The ID is already being used by another connection. This means for TC_SEND, TC_RECV or TC_DISCON, BUSY is currently TRUE. The status code is output when EN_R of TC_RECV is permanently TRUE. This mostly results in TC_RECV being called. Remedy for this situation: Turn EN_R off before TC_CON or TC_DISCON is called. EN_R may only be turned on again if TC_CON executed free of errors.
0	1	$80E6_H$	No query in progress (instruction call not started)

DONE	ERROR	STATUS	Meaning
0	1	$80E8_H$	Remote partner cannot be reached. Check the connection parameters. In the "GPRS direct" mode, the message is output if the partner can be reached but is not accepting a connection request.
0	1	$80EB_H$	Request temporarily denied (TC_CON has already been called with the same destination address.)
0	1	$80EC_H$	Opening the Listener Port failed: Check the connection parameters.
0	1	$80F2_H$	The CP is in the wrong mode: • Telecontrol connections are permitted only in "Telecontrol" mode. • ISO-ON-TCP connections are permitted only in "GPRS direct" mode.
0	1	$80F3_H$	No free connection endpoint for sending data: • Use less connections or • Use less passive connections or • Turn off NTP. Remember the maximum number of simultaneous connections of the CP 1242-7.
0	1	$80F4_H$	Connection endpoint cannot be generated: Repeat the call. If necessary, check the connection parameters.
0	1	$80F5_H$	Invalid connection endpoint: Connection establishment by TC_CON failed. Repeat the block call.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format, invalid value or phone number in TCON_Phone longer than 20 characters) Check the configuration of the "TC_CON..." SDT.

TC_DISCON: Terminate connection via the GSM network

Meaning

The TC_DISCON instruction on an S7-1200 with CP 1242-7 terminates an ISO-ON-TCP, UDP, SMS or telecontrol connection that was established with the TC_CON instruction.

You will find detailed information on the connection types in the description of the TC_CON instruction.

TC_DISCON terminates the connection to the telecontrol server only logically. At the TCP/IP level, the connection is retained.

If you want the connection to the telecontrol server to be terminated physically, configure the connection as a "Temporary connection" in the "telecontrol server" parameter group in STEP 7. Temporary stations terminate the connection automatically after sending the data.

Note

Stopping the execution of further program blocks by TC_DISCON

Calling up TC_DISCON ends the execution of TC_CON, TC_SEND, and TC_RECV blocks, that were called up with the same connection ID (parameter "ID") and interface (parameter "INTERFACE"). These blocks then signal an ERROR.

Do not call up TC_DISCON if TC_CON indicates "Error = 1".

If TC_CON indicates "ERROR", then the connection is not established. In this case TC_DISCON must not be called.

If TC_DISCON is called in this case, the connection ID ("ID") is briefly reserved, and a TC_CON called up immediately afterwards would indicate ERROR and STATUS 80E3.

The ID parameter references the GPRS connection. The ID must be unique within the CPU and the same as the ID used with TC_CON.

The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

Call interface in FBD representation

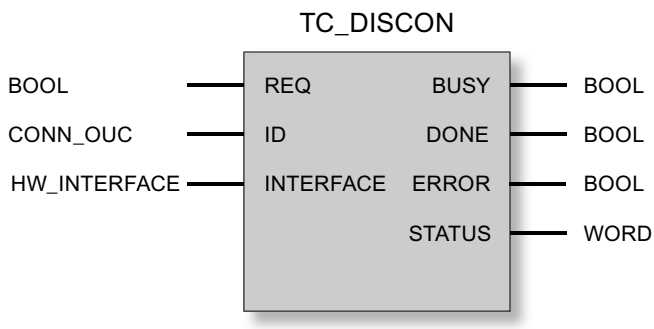

Explanation of the formal parameters

The following table explains all the formal parameters for the TC_DISCON instruction

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection

Parameter	Declaration	Data type	Range of values	Description
INTERFACE	INPUT	HW_INTER-FACE		Reference to the interface of the CP For information on the "Hardware identifier", refer to the section TC_CON: Establish connection via the GSM network (Page 3866).
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	The status parameter indicates whether the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	The instruction has not yet been called.

You will find all other code combinations of DONE and ERROR in the following table.

Note

When called, the instruction remains in the BUSY = 1 state for several seconds.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)

DONE	ERROR	STATUS	Meaning
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	$80E4_H$	Unknown ID: No connection with this ID has been established by TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed or • Connection terminated by remote partner.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the configuration of the "TC_CON..." SDT.

TC_SEND: Send data via the GSM network

Meaning

The TC_SEND instruction allows the sending of data via programmed connections of the following types:

- ISO-ON-TCP connections
- UDP connections
- SMS connections
  The sending of SMS messages is supported only if this was set up in the STEP 7 configuration of the CP.
- Telecontrol connections

Note

Sending SMS messages to multiple recipients

If you want to send an identical SMS message to several recipients, you need to establish a connection to each recipient.

You will find more detailed information on the connection types in the description of the TC_CON instruction.

The ID parameter references the GPRS connection. The value of ID must correspond to the value used for ID by TC_CON.

The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

The amount of data to be sent is specified with the LEN parameter.

The size of the data area specified in DATA must be at least as large as the number of bytes configured for LEN. Permitted data types in the data area specified in DATA are all except BOOL and ARRAY of BOOL.

The destination address (connection partner) for the data to be sent is configured in the TC_CON instruction.

Call interface in FBD representation

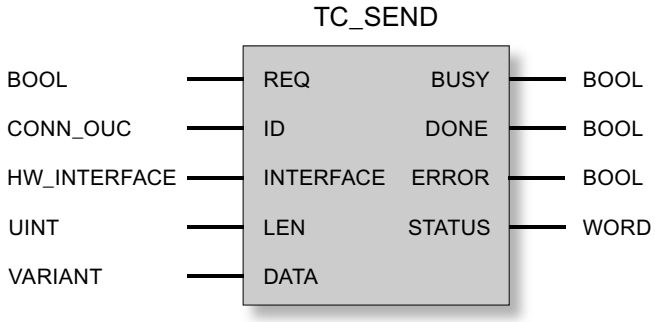

Explanation of the formal parameters

The following table explains all the formal parameters for the TC_SEND instruction.

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the CP For information on the "Hardware identifier", refer to the section TC_CON: Establish connection via the GSM network (Page 3866).
LEN	INPUT	UINT	1...2048	Number of bytes of data to be sent. The value must be ≥ 1 and ≤ 2048. The value should match the size of the range of DATA.
DATA	INOUT	VARIANT		Address reference to the send data area of the CPU *
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	The status parameter indicates whether the job was completed without errors. ** For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.

Parameter	Declaration	Data type	Range of values	Description
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

* For special features of the DATA parameter for SMS texts, refer to the next section.

** After sending a frame, TC_SEND sets DONE = 1. Note the following response:
The loss of an ISO-on-TCP connection is only recognized by the sender after 1 to 2 minutes. The transferred data may be lost although TC_SEND has set DONE = 1 at the sender.
If an ISO-on-TCP connection is aborted after receiving a frame before TC_RECV was started, the transferred data may be lost even if TC_SEND sets DONE = 1 at the sender.

Configuring SMS texts with the DATA parameter

The instruction sends the data referenced by the pointer of the type VARIANT of the DATA parameter as an SMS text.

If an operand of the data type STRING is referenced by DATA for SMS texts, the first two bytes are transferred with length information of the string.

One option for the correct text representation of SMS messages to be sent is to convert the text string into an Array of BYTE or Array of CHAR using the conversion function Strg_TO_Chars. Strg_TO_Chars at the EN parameter is linked to the output parameter ENO by TC_SEND.

For SMS texts, the CP does not support all special characters, for example umlauts (ü, ä etc.). The specification GSM 03.38 applies. There may be additional restrictions imposed by the GSM network provider.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS *	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)

DONE	ERROR	STATUS *	Meaning
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	$80E0_H$	Internal error If you send the frames directly to the telecontrol server (mode "Telecontrol"), make sure that the send cycle time is ≥ 1 second.
0	1	$80E1_H$	Timeout: • Increase the value of the "Connection monitoring time" in the configuration of the CP 1242-7 or • Check the connection partner.
0	1	$80E4_H$	Unknown ID: First call TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80E7_H$	Data to be sent not completely transferred: Repeat the job.
0	1	$80E8_H$	Remote partner cannot be reached. Check the connection parameters. In the "GPRS direct" mode, the message is output if the partner can be reached but is not accepting a connection request.
0	1	$80E9_H$	Connection establishment by remote partner: Check the connection partner. If necessary, terminate the connection with TC_DISCON and establish it again with TC_CON.
0	1	$80EA_H$	Error message from remote partner: • Check the connection partner. Enable the "TC_RECV" instruction on the communications partner. • If necessary, terminate the connection with TC_DISCON and establish it again with TC_CON.
0	1	$80EF_H$	SMS could not be sent: • Check whether the destination address (telephone number of the destination subscriber) exists. • Check whether the inserted SIM card allows sending of SMS. • Check the length of the SMS text that was sent. SMS texts > 160 characters will not be sent. • Make sure that when the data block TCON_PHONE was created, the "Standard" option was selected for block access.
0	1	$80F1_H$	Sending of SMS messages is not enabled in the STEP 7 configuration of the CP: Enalble the "Allow SMS" option in the configuration of the CP.
0	1	$80F4_H$	Connection endpoint cannot be generated: Check the connection partner.
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed. or • Connection terminated by remote partner: First call TC_DISCON.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value): Check the configuration of the "TC_CON..." SDT.

* Further statuses that are not listed here can be found in the status display is of the "RDREC" or "WRREC" instructions in the two middle status bytes (STATUS[2], STATUS[3]).

TC_RECV: Receive data via the GSM network

Meaning

The TC_RECV instruction allows the reception of data via programmed connections of the following types:

- ISO-ON-TCP connections
- SMS connections
  To receive SMS messages, the phone number of the sender must be configured in the STEP 7 configuration of the receiving CP (authorized phone numbers). The sender must support the CLIP function.
  The phone number of the connection partner must be entered in the "TCON_PHONE" SDT.
  Wake-up SMS messages are filtered out.
- Telecontrol connections

Note

Receiving SMS messages from different senders

If you want to receive SMS messages from different senders, you have two alternatives:

- You configure several connections (TC_CON, TC_RECV, TC_DISCON).
  or
- You may only enter no telephone number for only one configured connection in the required data block "TCON_PHONE" in the "PhoneNumber" parameter. When receiving messages, this is then interpreted as a placeholder for all authorized connection partners.

You will find more detailed information on the connection types in the description of the TC_CON instruction.

The ID parameter references the GPRS connection. The value of ID must correspond to the value used for ID by TC_CON.

The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

The maximum amount of data to be received is specified with the LEN parameter.

The size of the data area specified in DATA must be at least as large as the number of bytes configured for LEN. Permitted data types in the data area specified in DATA are all except BOOL and ARRAY of BOOL. The received data is interpreted as if the sending partner had used the same data types.

The DB (system data type) used for the connection description of TC_RECV must differ from a DB used for TC_SEND.

Storing SMS messages

Received SMS messages are stored retentively in the CP 1242-7 (25 storage spaces) and on the SIM card (varying number of storage spaces).

- After an SMS message has been read by TC_RECV the SMS message will be deleted from its storage space.
- If all of the storage spaces have been allocated and a new SMS message is received, the oldest SMS message is deleted.

Call interface in FBD representation

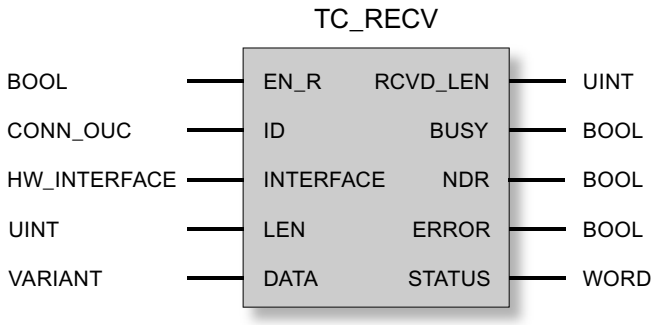

Explanation of the formal parameters

The following table explains all the formal parameters for the TC_RECV instruction

Parameter	Declaration	Data type	Range of values	Description
EN_R	INPUT	BOOL	0: Data reception locked 1: Data reception enabled	Enables / locks the reception of data. • Block version 1.1: After setting 1 to 0, the block is inactive. • Block version 1.0: After setting 1 to 0, the program block receives data again (until DONE = 0 and ERROR = 0). Note the information on the status code 80E3 of TC_CON.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTER-FACE		Reference to the interface of the CP 1242-7 For information on the "Hardware identifier", refer to the section TC_CON: Establish connection via the GSM network (Page 3866).
LEN	INPUT	UINT	1...2048	(minimum) number of bytes of data to be received, maximum 2048
DATA	INOUT	VARIANT		Address reference to the receive data area of the CPU *

Parameter	Declaration	Data type	Range of values	Description
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
RCVD_LEN	OUTPUT	UINT		Number of bytes of received data
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	The status parameter indicates whether the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

* For special features of the DATA parameter for SMS texts, refer to the next section.

Configuring SMS texts with the DATA parameter

The instruction references the received SMS text with the pointer of the type VARIANT of the DATA parameter to the data area of the CPU.

If DATA references an operand of the data type STRING for the SMS text, the first two bytes of the SMS text will be interpreted as length information of the data type STRING and not as SMS text.

One option for the correct text representation of SMS messages to be received is to convert an Array of BYTE or Array of CHAR to a text string using the conversion function Chars_TO_Strg. Chars_TO_Strg at the EN parameter is linked to the output parameter ENO of TC_RECV.

For SMS texts, the CP does not support all special characters, for example umlauts (ü, ä etc.). The specification GSM 03.38 applies. There may be additional restrictions imposed by the GSM network provider.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS *	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	$80A3_H$	• An attempt is made to re-establish an existing connection. • An attempt is made to terminate a non-existent connection.
0	1	$80E0_H$	Internal error
0	1	8086_H	Illegal value for ID
0	1	$80E4_H$	Unknown ID: First call TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed. or • Connection terminated by remote partner: First call TC_DISCON.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the configuration of the "TC_CON..." SDT.

* Further statuses that are not listed here can be found in the status display is of the "RDREC" or "WRREC" instructions in the two middle status bytes (STATUS[2], STATUS[3]).

TC_CONFIG: Transferring configuration data to a CP

Meaning

With the instruction TC_CONFIG the parameters of an S7-1200 mobile wireless CP configured in STEP 7 can be changed. The configured values are not overwritten retentively. The overwritten values remain valid until TC_CONFIG is called again or until the station starts up again (cold restart after cycling power).

To be able to use the function, there must already be values present in the STEP 7 basic configuration of the CP.

If the STEP 7 configuration data of the CP needs to be changed permanently, the instruction needs to be called again each time the station restarts (cold restart) or a modified project must be downloaded to the station.

The CONFIG parameter points to the memory area with the configuration data. The configuration data is stored in a data block (DB). The structure of the DB is specified by the system data type (SDT) IF_CONF.

The configuration data to be modified on the CP is put together as necessary in blocks in IF_CONF "IF_CONF_..." for the individual parameters.

Parameters that are not intended to change as a result of the instruction are not entered in IF_CONF. They retain the value configured in STEP 7.

For detailed information on settings for IF_CONF, refer to the section IF_CONF: SDT for telecontrol configuration data (Page 3887).

The INTERFACE parameter references the GPRS interface of the required local CP.

Call interface in FBD representation

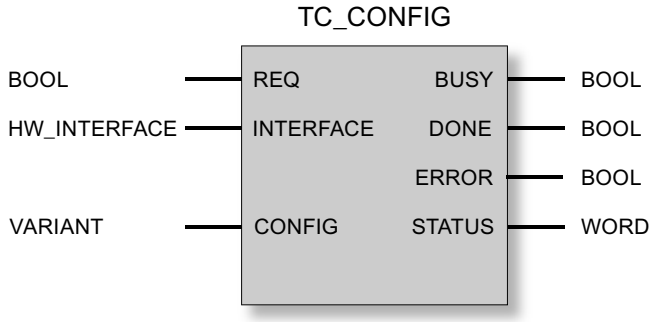

Explanation of the formal parameters

The following table explains all the formal parameters for the TC_CONFIG instruction

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
INTERFACE	INPUT	HW_INTERFACE (WORD)		Reference to the interface of the CP 1242-7 For information on the "Hardware identifier", refer to the section TC_CON: Establish connection via the GSM network (Page 3866).
CONFIG	INOUT	VARIANT	See also "IF_CONF: SDT for telecontrol configuration data	Reference to the memory area with the collected configuration data to be modified
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction

Parameter	Declaration	Data type	Range of values	Description
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	The status parameter indicates whether the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	$80E0_H$	Internal error (for example no value in the basic configuration of a parameter)
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80EB_H$	Query temporarily rejected (the CP is currently being configured by STEP 7).
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the "IF_CONF" SDT.
0	1	$80F7_H$	Wrong ID in the parameter fields of the configuration data: Check the "IF_CONF" SDT.

TCON_...: SDTs for the telecontrol connection establishment

System data types TCON_... for telecontrol connections

The four system data types TCON_... described below are used, for example, for the connection description of telecontrol connections of a telecontrol CP.

TCON_PHONE is also used with the mobile wireless CPs of the S7-1200 for the description of a connection of Open User Communication to transfer SMS messages (TCON / TDISCON / TSEND or TSEND_C).

To configure a telecontrol connection for the telecontrol CP using the TC_CON instruction, the CONNECT parameter of the instruction is used for the connection description.

The connection description is specified by the structure of the system data type (SDT). The structure of the relevant SDT contains the parameters necessary to establish the connection with the remote communications partner.

For different connection types that depend on the remote communications partner, the following SDTs are used:

- TCON_IP_RFC for ISO-on-TCP connections to IPv4 stations with telecontrol CP
- TCON_IP_V4 for UDP connections to IPv4 stations (sending only)
- TCON_PHONE for connections to SMS clients
- TCON_WDC for connections to telecontrol servers or stations that can be reached via the telecontrol server.

The parameter assignment of the connection description is made in a data block of the same type as the SDT.

Creating a DB of the type TCON_...

You will need to type in the data types of the relevant DBs with the keyboard. They are not displayed in the selection list. The data types are not case-sensitive.

To create a TCON_... DB, follow the steps outlined below:

1. Create a data block of the type "Global DB" with standard block access (not optimized).
2. Create an SDT in the table of the parameter configuration of the DB by assigning the name and typing in the required type (for example "TCON_IP_RFC") in the cell of the data type. The SDT and its parameters are created (see below).
3. Configure the parameters that are described below for each SDT type.

Reserved bits are not displayed.

System data type TCON_IP_RFC for connections to IPv4 stations

This connection type is supported only on ISO-on-TCP connections to communications partners with a fixed IP address. The CP must be configured for the "GPRS direct" mode.

Table 4-104 Parameters of TCON_IP_RFC

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the CP For information on the "Hardware identifier", refer to the section TC_CON: Establish connection via the GSM network (Page 3866).
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0C	Protocol variant 12 (C_h): ISO-on-TCP connection
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment • 1: Active connection establishment
6 ... 7	-	-	-	- reserved -
8 ... 11	RemoteAddress	IP_V4		IP address of the connection partner
	ADDR	Array [1...4] of Byte		IP address of the relevant connection partner
12 ... 13	RemoteTSelector	TSelector		Remote T selector
	TSelLen	UINT		Length of the remote T selector "RemoteTSelector"
14 ... 45	TSel	Array [1...32] of Byte	any	Remote transport selector of the connection • When "ActiveEstablished" = 1: With active connection establishment, the T selector of the local partner must be the same as the T selector of the connection partner (passive connection establishment on the remote partner). • When "ActiveEstablished" = 0 correspondingly (passive connection establishment local, active connection establishment remote)
46 ... 47	LocalTSelector	TSelector		Local T selector
	TSelLen	UINT		Length of the local T selector "LOCAL_TSel"
48 ... 79	TSel	Array [1...32] of Byte	any	Local transport selector of the connection • When "ActiveEstablished" = 1: With active connection establishment, the T selector of the local partner must be the same as the T selector of the connection partner (passive connection establishment on the remote partner). • When "ActiveEstablished" = 0 correspondingly (passive connection establishment local, active connection establishment remote)

System data type TCON_IP_V4 for connections to IPv4 stations

This connection type is supported only for sending on UDP connections to communications partners with a fixed IP address.

To receive, ActiveEstablished = 0 must be set.

Table 4-105 Parameters of TCON_IP_V4

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the CP For information on the "Hardware identifier", refer to the section TC_CON: Establish connection via the GSM network (Page 3866).
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0B	Protocol variant 11 (B_h): UDP connection
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment Setting for sending and receiving data. • 1: Active connection establishment Setting for sending data only.
6 ... 9	RemoteAddress	IP_V4		IP address of the connection partner
	ADDR	Array [1...4] of Byte		The four bytes (ADDR[1] ... ADDR[4]) specify the four blocks of the IP address.
10 ... 11	RemotePort	UINT	1...65535	IP port of the connection partner Not relevant if ActiveEstablished = 0.
12 ... 13	LocalPort	UINT	1...65535	Local IP port ("0" is not permitted) Not relevant if ActiveEstablished = 1.

System data type TCON_PHONE for SMS connections

Note

Authorized phone numbers

The CP only accepts an SMS if the sending communication partner is authorized based on its phone number. These numbers are in configured for the CP in STEP 7 in the "authorized phone numbers" list.

SMS text

- Programmed SMS texts for SMS messages to be sent are accessed using the DATA parameter of the TC_SEND instruction.
- The text of a received SMS message is assigned to the address area of the CPU by the DATA parameter of the TC_RECV instruction.

Table 4-106 Parameters of TCON_PHONE

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the CP For information on the "Hardware identifier", refer to the section TC_CON: Establish connection via the GSM network (Page 3866).
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the local mobile wireless connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0E	Protocol variant 14 (E$_h$): SMS connection
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment (not relevant for the CP 1242-7): • 0: Passive connection establishment (not relevant here) • 1: Active connection establishment
6...7	-	-	-	- reserviert -
8 ... 31	PhoneNumber	STRING[22]		Call number of the connection partner Permitted values: Plus character (+) and numbers Note the exact notation of the international dialing code of the relevant phone number assigned by the network provider ("+" character or zeros). Without an entry for the PhoneNumber parameter, no connection partner is specified and SMS messages can be received reception from all authorized connection partners. Note the following during startup: Without an entry, TC_RECV first delivers the oldest received SMS message.

System data type TCON_WDC for connections to telecontrol servers or remote stations

You can configure the connection to the telecontrol server assigned to the S7-1200 or to a remote station that can be reached via the telecontrol server with TCON_WDC. The address data of the telecontrol server assigned to the CP can be found in STEP 7 in the "Telecontrol interface > Mode" tab of the CP. The telecontrol server or a remote station is addressed using the host name or the IP address.

The "RemoteWdcAddress" parameter of TCON_WDC specifies the Access ID of the connection partner.

Table 4-107 Parameters of TCON_WDC

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the CP For information on the "Hardware identifier", refer to the section TC_CON: Establish connection via the GSM network (Page 3866).
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0F	Protocol variant 15 (F$_h$): Telecontrol connection using an IP address
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment • 1: Active connection establishment
6 ... 7	-	-	-	- reserved -
8 ... 11	RemoteWdcAddress	DWORD		Specifies the Access ID (hex). The access ID depends on the connection partner. • Connection to a remote CP: The access ID is made up of the following: – STEP 7 project number – Station number – Slot If the remote station has more than one GPRS-CP and you do not want to specify the path, the last byte for the slot must be set to 0. You will find the access ID in the STEP 7 project in the "CP authentication of the CP" parameter group. • Connection to the telecontrol server: Access ID = 0 • To only write to the process image of the CP: Access ID = DW#16#FEEDDADA

IF_CONF: SDT for telecontrol configuration data

Structure of the system data type IF_CONF for the TC_CONFIG instruction

The TC_CONFIG parameter of the CONFIG instruction references the memory area containing the CP configuration data to be modified. The configuration data stored in a data block is described as a structure of the IF_CONF system data type (SDT).

To be able to use the function, there must already be values present in the STEP 7 basic configuration of the CP.

IF_CONF is made up of a header followed by fields that correspond to the parameters or parameter areas of the CP in the device properties of the STEP 7 project.

The CP configuration data to be modified is collected together as IF_CONF fields. Parameters that will not be modified are ignored in the IF_CONF structure and remain as they were configured in the STEP 7 project.

Creating the DB and the IF_CONF structures

You can create the parameters of the CP within the IF_CONF DB in one or more structures each with one or more fields.

You will need to type in the data types of the fields using the keyboard. They are not displayed in the selection list. The data types are not case-sensitive.

Follow the steps below to create IF_CONF:

1. Create a data block of the type "global DB" with block access "Standard".
2. Create a structure (data type "Struct") in the table of the parameter configuration of the DB. You can specify any name.
3. Under this structure add a header by assigning the name of the header and typing it in the cell of the data type "IF_CONF_Header".
   The header of the structure and its three parameters (see below) is created.
4. Create a field for the first parameter to be changed by typing in the required data type (for example "IF_CONF_APN") in the cell of the data type.
5. Repeat the last step for all parameters you want to change on the CP using the TC_CONFIG instruction.
6. Finally, update the number of fields in the header in the "subfieldCnt" parameter.

Header of IF_CONF

Table 4-108 IF_CONF_Header

Byte	Parameter	Data type	Initial value	Description
0 ... 1	fieldType	UINT		Field type: Must always be 0.
2 ... 3	fieldId	UINT		Field ID: Must always be 0.
4 ... 5	subfieldCnt	UINT		Total number of fields contained in the structure

General parameters of the parameter fields

Each field has the following general parameters:

- Id
  This parameter identifies the field and must not be modified.
- Length
  This parameter indicates the length of the field. The value serves as information.
  Fields with strings and / or arrays have a variable length. Due to hidden bytes, the actual length of fields can be greater than the sum of the displayed parameters.
- Mode
  The following values are permitted to these parameters:

Table 4-109 Values of "Mode"

Value	Meaning
1	Permanent validity of the configuration data Not relevant for the CP
2	Temporary validity of the configuration data, including deleting of existing permanent configuration data The permanent configuration data is replaced by the parameter fields of IF_CONF.

Field for the parameter area "GPRS access"

Table 4-110 IF_CONF_APN

Parameter	Data type	Initial value	Description
Id	UINT	4	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 174
Mode	UINT		Validity (1: permanent, 2: temporary)
AccesspointGPRS	STRING [98]		APN: Name of the access point of the GSM network provider to the Internet
AccesspointUser	STRING [42]		APN user name
AccesspointPassword	STRING [22]		APN password

Field for the parameter area "CP identification"

Table 4-111 IF_CONF_Login

Parameter	Data type	Initial value	Description
Id	UINT	5	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 54
Mode	UINT		Validity (1: permanent, 2: temporary)

Parameter	Data type	Initial value	Description
ModemName	STRING [22]		Access ID The value cannot be set.
ModemPassword	STRING [22]		Telecontrol password (max. 20 characters) The password with the CP 1242-7 (6GK7 242-7KX30-0XE0) cannot be changed via the SDT.

Field for the parameter area "Telecontrol server access"

This field is only used when the telecontrol server is addressed with a name that can be resolved by DNS. If the telecontrol server is addressed with its IP address, the "IF_CONF_TCS_IP_V4" field is used.

In STEP 7, the corresponding data is located in the "Mode" parameter area.

If there is more than one telecontrol server, use the field once per server.

Table 4-112 IF_CONF_TCS_Name

Parameter	Data type	Initial value	Description
Id	UINT	6	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 266
Mode	UINT		Validity (1: permanent, 2: temporary)
TcsName	-	-	- reserved -
	STRING [254]		Name of the telecontrol server that can be resolved by DNS
RemotePort	UINT		Port of the telecontrol server
Rank	UINT		Priority of the server [1, 2] 1 = first telecontrol server, 2 = second telecontrol server

Field for the parameter area "Telecontrol server access"

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

This field is only used when the telecontrol server is addressed by its IP address. If the telecontrol server is addressed by its DNS name, the "IF_CONF_TCS_Name" field is used.

In STEP 7, the corresponding data is located in the "Mode" parameter area.

If there is more than one telecontrol server, use the field once per server.

Table 4-113 IF_CONF_TCS_IP_v4

Parameter	Data type	Initial value	Description
Id	UINT	7	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
RemoteAddress	IP_V4		IP address of the telecontrol server

Parameter	Data type	Initial value	Description
RemotePort	UINT		Port of the telecontrol server
Rank	UINT		Priority of the server [1, 2] 1 = first telecontrol server, 2 = second telecontrol server

Field for the "Mode" parameter area

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

In STEP 7, the corresponding data is located in the parameter areas "Mode" and Modem settings".

Table 4-114 IF_CONF_GPRS_Mode

Parameter	Data type	Initial value	Description
Id	UINT	8	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 10
Mode	UINT		Validity (1: permanent, 2: temporary)
GPRSmode	UINT		Mode of the CP: • 0 = Telecontrol • 1 = GPRS direct
TemporaryStation	BOOL		Bit 0: Temporary connection If this option is selected, the CP only establishes a temporary connection to send data. Once the messages have been transferred, the CP terminates the connection again. • 1: activated (temporary connection) • 0: deactivated (permanent connection)
SMS_Enabled	BOOL		Bit 1: Allow SMS Selecting the option allows the S7 station to send SMS messages. • 1: activated (SMS allowed) • 0: deactivated (no SMS)

Field for the "SMSC" parameter

In STEP 7, the corresponding data is located in the parameter area "Modem settings".

Table 4-115 IF_CONF_SMS_Provider

Parameter	Data type	Initial value	Description
Id	UINT	10	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 28

Parameter	Data type	Initial value	Description
Mode	UINT		Validity (1: permanent, 2: temporary)
SMSProvider	STRING [20]		Node number of the SMS center (SMSC) of the GSM network provider with which the contract was signed for this station.

Field for the "PIN" parameter

In STEP 7, the corresponding data is located in the parameter area "Modem settings".

Table 4-116 IF_CONF_PIN

Parameter	Data type	Initial value	Description
Id	UINT	11	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 16
Mode	UINT		Validity (1: permanent, 2: temporary)
Pin	STRING [8]		PIN of the SIM card inserted in the SIM card The parameter is not relevant if the PIN was correctly configured. If the PIN was incorrectly configured, the correct PIN can be entered.

Field for monitoring times

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

In STEP 7, the corresponding data is located in the parameter areas "Keepalive timeout" and Operating mode".

Table 4-117 IF_CONF_TC_Timeouts

Parameter	Data type	Initial value	Description
Id	UINT	12	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 12
Mode	UINT		Validity (1: permanent, 2: temporary)
KeepAliveTimeout	-	-	- Reserved - (cannot be set)

Parameter	Data type	Initial value	Description
SendTimeout	UINT		Connection monitoring time: Monitoring time of the connection to the communications partner (seconds) Relevant in the modes "Telecontrol" and "GPRS direct"
RedialTimeout	UINT		Reconnection delay: Basic value for the wait time until the next attempt to establish a connection following an unsuccessful connection establishment. After every 3 attempts, the basic value is doubled up to a maximum of 900 s. Range of values: 10 to 600 s. Example: Basic value 20 results in the following dialing intervals: three times 20 s, three times 40 s, three times 80 s etc. up to a maximum of 900 s. If a second telecontrol server / router is configured, the CP attempts to connect to the second partner at the 4th attempt. If the second partner is also unreachable, the 7th time the CP attempts to connect to the first partner again. Not relevant for SMS connections

Field for the "Wake up right" parameter area

Table 4-118 IF_CONF_WakeupList

Parameter	Data type	Initial value	Description
Id	UINT	13	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 246
Mode	UINT		Validity (1: permanent, 2: temporary)
WakeupPhone [1...10]	ARRAY [1...10] of STRING [22]		Phone number subscriber authorized to wake up The asterisk (*) at the end of a call number is used a placeholder for direct dialing numbers.

Field for the "Preferred GSM networks" parameter area

Table 4-119 IF_CONF_PrefProvider

Parameter	Data type	Initial value	Description
Id	UINT	14	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 46
Mode	UINT		Validity (1: permanent, 2: temporary)
Provider [1...5]	ARRAY [1...5] of STRING [6]		Alternative GSM networks with priority 1 to 5 into which the CP dials. Up to 5 networks can be configured. No. 1 with highest priority, no. 5 with lowest priority. Entry of the Public Land Mobile Network (PLMN) of the network provider consisting of Mobile Country Code (MCC) and Mobile Network Code (MNC). Example (test network of Siemens AG): 26276

Field for the "DNS configuration" parameter area

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

Table 4-120 IF_CONF_DNS

Parameter	Data type	Initial value	Description
Id	UINT	16	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
DNS_IP [1]	IP_V4		IP address of the 1st Domain Name System server
DNS_IP [2]	IP_V4		IP address of the 2nd Domain Name System server

Field for the "Time-of-day synchronization" parameter area

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

Table 4-121 IF_CONF_NTP

Parameter	Data type	Initial value	Description
Id	UINT	17	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 24
Mode	UINT		Validity (1: permanent, 2: temporary)
NTP_IP [1]	ARRAY [1...4] of IP_V4		IP address of NTP server 1
...	...		(IP address of NTP server 2...3)
NTP_IP [4]	ARRAY [1...4] of IP_V4		IP address of NTP server 4

Block for activating / deactivating TeleService users

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

SDT for activating or deactivating TeleService users already configured in the STEP 7 project of the CP. In STEP 7, the corresponding data can be found in the parameter area "TeleService settings" > "TeleService user management".

Table 4-122 IF_CONF_GPRS_UserList

Parameter	Data type	Initial value	Description
Id	UINT	19	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 506
Mode	UINT		Validity (1: permanent, 2: temporary)
GPRS_User [1...10]	ARRAY [1...10] of GPRS_User		TeleService user no. 1 to max. no. 10

The array is formed from the parameter records for the TeleService users ("GPRS_User" [1...n]).

Table 4-123 GPRS_User [n] (parameter for TeleService user)

Parameter	Data type	Initial value	Description
UserName [n]	STRING [22]		TeleService user name
Password [n]	STRING [22]		- The string must be empty!
Diag_Allowed [n]	BOOL		- Reserved - (cannot be set)
Teleserv_Allowed [n]	BOOL		Activation of the TeleService user • 0 = user is deactivated • 1 = user is activated
FW_Load_Allowed [n]	BOOL		- Reserved - (cannot be set)

Field for setting the parameters for TeleService access (DNS name of the server)

Access data of the TeleService server (switching station).

In STEP 7, the corresponding data is located in the parameter area "TeleService settings".

If there is more than one TeleService server, use the field once per server.

Table 4-124 IF_CONF_TS_Name

Parameter	Data type	Initial value	Description
Id	UINT	20	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 266
Mode	UINT		Validity (1: permanent, 2: temporary)
ts_name	String [254]		Name of the TeleService server that can be resolved by DNS
RemotePort	UINT		Port of the engineering station
Rank	UINT		Priority of the server [1] or [2] 1 = server 1, 2 = server 2

Field for setting the parameters for TeleService access (IP address of the server)

Access data of the TeleService server (switching station).

In STEP 7, the corresponding data is located in the parameter area "TeleService settings".

If there is more than one TeleService server, use the field once per server.

Table 4-125 IF_CONF_TS_IF_V4

Parameter	Data type	Initial value	Description
Id	UINT	21	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
RemoteAddress	IP_V4		IP address of the TeleService server

Parameter	Data type	Initial value	Description
RemotePort	UINT		Port of the TeleService server
Rank	UINT		Priority of the server [1] or [2] 1 = server 1, 2 = server 2

S7-300/400-CP

CP 443-1 OPC UA

Program blocks for the OPC UA client

Overview of the program blocks for the OPC UA client function

The function blocks listed below are available in the library "SIMATIC NET CP" for the OPC UA client function of the CP. Use the blocks on the CPU.

Some blocks require special user-defined PLC data types (UDTs) that you will find in the following below the relevant program block. The UDTs are available as pre-assembled data types. To do this under the entry "PLC data types" of the station create a new data type and in the "data type" box, type in the name of the UDT. for example "UA_SessionConnectInfo". The required parameters will then be created automatically in the declaration table of the UDT.

- UA_Connect
  Establishes a connection for a session with an OPC UA server.
  - UA_SessionConnectInfo
    Contains connection information for the block parameter "SessionConnectInfo".
  - UA_UserIdentityToken
    Contains the data of the user authentication for the block parameter "UserIdentityToken".
- UA_NamespaceGetIndex
  Fetches the name space index of a name space URI.
- UA_NodeGetHandleList
  Registers node IDs on the connected server and fetches the node handles in the form of a list.
  - UA_NodeID
    Contains the parameters for identifying the target node for the block parameter "NodeID".
- UA_NodeReleaseHandleList
  Releases the node handles of the used list on the server.
- UA_Disconnect
  Terminates the connection of a current session with an OPC UA server.

- UA_ReadList
  Using the list of node handles, reads the data from nodes of the connected server.
  - UA_NodeAdditionalInfo
    Specifies the item attribute and the index range for the block parameter "NodeAddInfos".
  - UA_IndexRange
    Specifies the start and end index for the block parameter "IndexRange”.
  - UA_AnyPointer
    References the memory area of the CPU in which the process data for the block parameter "Variables" is stored.
  - UA_TimeStamps
    Contains the time stamps of the elements of the data area of the block parameter "Variables".
- UA_WriteList
  Using the list of node handles, writes data to nodes of the connected server.
  - UA_NodeAdditionalInfo
    Specifies the item attribute and the index range for the block parameter "NodeAddInfos".
  - UA_IndexRange
    Specifies the start and end index for the block parameter "IndexRange”.
  - UA_AnyPointer
    References the memory area of the CPU in which the process data for the block parameter "Variables" is stored.

Additionally required system functions (SFC)

The following system functions are required for the full functionality of the program blocks for the OPC UA client:

- BLKMOV (SFC20)
- TIME_TCK (SFC64)

For its function, UA_Connect also requires:

- LGC_GADR (SFC49)
- RDSYSST (SFC51)

Creating SFCs

Create the system functions by opening an organization block in the block directory of the CPU, for example OB1.

In the block catalog that is displayed by opening OB1, expand the corresponding block group.

Find the SFCs in the following block order:

- BLKMOV (SFC20): "Simple instructions" group > "Move" folder
- TIME_TCK (SFC64): "Extended instructions" group > "Day and time" folder
- LGC_GADR (SFC 49): "Extended instructions" > "Addressing" folder
- RDSYSST (SFC 51): "Extended instructions" group > "Diagnostics" folder

Drag the SFCs to the network of the organization block and supply the parameters of the SFC.

Calling the program blocks and their interaction

Note

General notes on the program blocks and instance DBs

Create the function blocks with "Default access method". In the assigned instance data blocks leave the option "Retentivity" disabled.

Each program block must be called cyclically until the status parameter "Done" or "Error" changes to 1.

Within a session only a single block can be called at any one time. Different blocks can only be processed one after another in a session.

With UA_Connect you establish a connection from the CP as OPC UA client to a UA server and open a session. The block must be called for every connection to a UA server. When necessary for more than one connection to a UA server, the block can be called more than once with different instance data blocks (IDBs). You can use this, for example, to read via one connection and to write via another connection or to read or write different data via multiple connections.

For connections to more than one UA server, the UA_Connect block must be called several times with different input parameters. As a UA client, the CP can establish connections to up to five UA servers.

To query the name space index of a name space URI of the UA server, call a UA_NamespaceGetIndex within a session. To query multiple name space URIs you can call the block more than once with different input parameters.

To create a handle list as preparation for the read and write services call up UA_NodeGetHandleList. The block is called separately with suitable information at the "NodeIDs" input parameter for each target node.

Following this you can use the blocks UA_ReadList und UA_WriteList to read and write the data of the items. You can call these two blocks as often as necessary within a session.

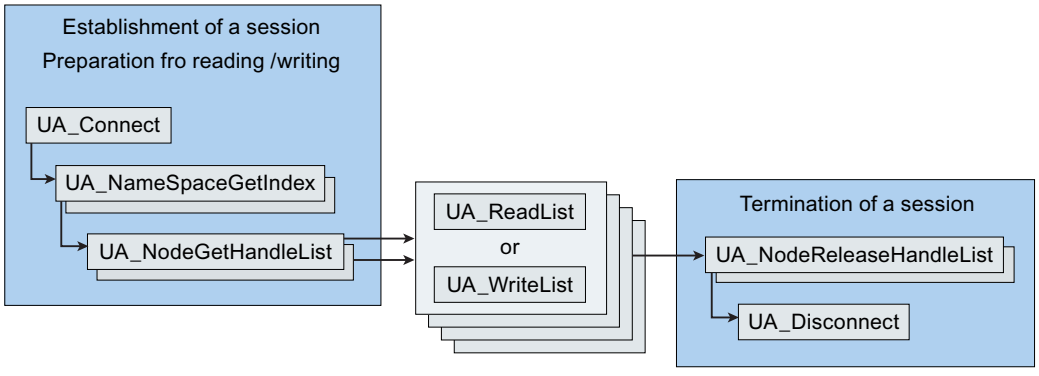

Figure 4-3 Calling the client program blocks and their interaction

If the connection is no longer required, release the handles on the server again using the block UA_NodeReleaseHandleList. This deletes the handles on the server.

The connection is terminated with UA_Disconnect and the session is ended.

See also

Time monitoring of the blocks (Page 3899)

Time monitoring of the blocks

Parameters for the time response of the blocks

To control and monitor the time response of the program blocks, the the following three parameters are available:

- Timeout
  Input parameter in all blocks for the OPC UA client
- SessionTimeout
  Input parameter in the block UA_Connect
- MonitorConnection
  Input parameter in the block UA_Connect

Since the parameters influence the running of the OPC UA communication as a whole, below notes on these parameters precede the block description.

Timeout

With this input parameter you monitor every block call. If a block call cannot be completed within the configured maximum permissible time, the status parameter "Error" is set to 1 and processing aborted.

SessionTimeout

With this input parameter you monitor the duration of a session without data traffic without the block "UA_Disconnect" being called to terminate a connection.

The minimum value of SessionTimeout" is 30 seconds. If the value is exceeded, the connection is terminated.

The parameter allows a server to reduce bound resources if the client does not use the session for a longer period of time.

An adequate time, however, ensures that the session is not terminated immediately if, for example, data cannot be transferred immediately due to network disruptions.

The value of "SessionTimeout" should generally be higher than the values of "Timeout" for the "UA_Connect" blocks used.

MonitorConnection

With this input parameter the status of a connection without data traffic is checked. When this connection monitoring time elapses, a frame is sent to check the status of the connection to the server.

To ensure that the session is not terminated immediately using the "SessionTimeout" parameter when there is no data traffic and the connection monitoring time is addressed, it is recommended that you select a value for "MonitorConnection" that is lower than half of "SessionTimeout".

See also

Program blocks for the OPC UA client (Page 3896)

UA_Connect

Function

Function of the block

With the UA_Connect block you establish a connection from the CP as OPC UA client to a UA server and open a session.

As the target address, you specify the URL of the UA server at the "ServerEndpointUrl" parameter.

You store the connection information at the "SessionConnectInfo" parameter in a data block.

UA_Connect also requires the system functions LGC_GADR (SFC 49) and RDSYSST (SFC 51).

See also

Parameter (Page 3900)

Program blocks for the OPC UA client (Page 3896)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-126 Parameters of the block UA_Connect

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ServerEndpointUrl	IN	STRING	Max. 254 characters	Address (URL) of the connection partner (server). Only IPv4 addresses are permitted.
SessionConnectInfo	IN	UDT	See "Meaning"	Connection information, see UASessionConnectInfo (Page 3913).

Parameter	Declaration	S7 data type	Range of values	Meaning
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
ConnectionHdl	OUT	DWORD	1 .. 5	Unique identifier of an established connection It is required by other blocks as an input parameter.
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 3919). Note: If the error code B000F002 is output, the OPC UA session is not yet established. Call the block again.

UA_NamespaceGetIndex

Function

Function of the block

With the UA_NamespaceGetIndex block you fetch the name space index of the name space URI of the connected UA server.

See also

Parameter (Page 3902)

Program blocks for the OPC UA client (Page 3896)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-127 Parameters of the block UA_NamespaceGetIndex

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block. As long as Execute = 1, the other input parameters cannot be changed.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
NamespaceUri	IN	STRING	Max. 254 characters	Specifies the address (name space URI) of the target server Example of the name space of the CPU symbols in the S7-400 with CP 443-1 OPC UA: http://www.siemens.com/simatic-classic-s7-opcua
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
NamespaceIndex	OUT	WORD	0 .. 65535	Output of the name space index of the name space specified by the "NamespaceUri" parameter. The namespace indexes 0 and 1 are reserved and specified by the OPC Foundation.
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 3919).

UA_NodeGetHandleList

Function

Function of the block

With the UA_NodeGetHandleList block, you register nodeIDs on the connected UA server.

The UA server returns the handles. These are output by the block with the "NodeHdls" parameter. You access the items of the node handles with the blocks "UA_ReadList" and "UA_WriteList" at their input parameter "NodeHdls".

The nodeIDs of the target server that you specify in the "NodeIDs" input parameter must be known to you. You store these in a UDT "UANodeIDs".

The number of target nodes at the input parameter "NodeIDCount" is always 1 so you need to call the block separately for each nodeID

See also

Parameter (Page 3903)

Program blocks for the OPC UA client (Page 3896)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-128 Parameters of the block UA_NodeGetHandleList

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
NodeIDCount	IN	WORD	1	Number of elements in the data area von "NodeIDs". The number of must be 1.
NodeIDs	IN	ARRAY of UDT	See "Meaning"	Structure of the parameters of the target node of the OPC server. The number of nodeIDs (1) must be identical to the number at the output parameter "NodeHdls". For information on the structure of the UDT, see UANodeID (Page 3915).
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.

Parameter	Declaration	S7 data type	Range of values	Meaning
NodeHdls	OUT	ARRAY of DWORD	0 .. 4 294 967 295	Array of the handles that the OPC UA server assigns as the reply to the client's query. The handles serve to uniquely identify the item on the server. The handles remain valid until they are released by calling the UA_NodeReleaseHandleList block, in other words become invalid. When a session is terminated, the CP cancels all registered node handles of this session. The parameter is used for the input parameter "NodeHdls" of the "UA_ReadList" and "UA_WriteList" blocks.
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 3919).
NodeErrorIDs	OUT	ARRAY of DWORD	See "Meaning"	Contains the error numbers for all elements of the data area of "NodeIDs". For the significance of the numbers, refer to the section Error numbers (Page 3919).

UA_NodeReleaseHandleList

Function

Function of the block

With the UA_NodeReleaseHandleList block you release the node handles of the current session on the connected UA server. This deletes the list.

The input parameter "NodeHdls" references the data block to which the handles of the output parameter "NodeHdls" of the "UA_NodeGetHandleList" block are written.

The number of handles at the input parameter "NodeHdlCount" is always 1 so you need to call the block separately for each handle.

See also

Parameter (Page 3905)
Program blocks for the OPC UA client (Page 3896)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-129 Parameters of the block UA_NodeReleaseHandleList

Parameter	Declaration	S7 data type	Range of values	Meaning / remarks
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
NodeHdlCount	IN	WORD	1	Number of elements in the data area von "NodeHdls". The number of must be 1.
NodeHdls	IN	ARRAY of DWORD	0 .. 4 294 967 295	Array of the IDs of the handles to be canceled. If execution of the block is successful, the handles are canceled on the server and therefore become invalid.
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 3919).
NodeErrorIDs	OUT	ARRAY of DWORD	See "Meaning"	Contains error codes for all elements of the data area of "NodeHdls".

UA_Disconnect

Function

Function of the block

With the UA_Disconnect block you terminate an existing connection between the CP as OPC UA client and a UA server. This ends the existing session.

See also

Program blocks for the OPC UA client (Page 3896)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-130 Parameters of the block UA_Disconnect

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 3919).

UA_ReadList

Function

Function of the block

With the UA_ReadList block you read data from the connected UA server.

You program the attributes to be read in a UDT "UANodeAdditionalInfo" that is referenced by the input parameter "NodeAddInfos".

The handles required at the input parameter "NodeHdls" are read from array of the output parameter "NodeHdls" of the "UA_NodeGetHandleList" block.

The number of nodeIDs at the input parameter "NodeIDCount" is always 1 so you need to call the block separately for each target node.

See also

Parameter (Page 3907)

Program blocks for the OPC UA client (Page 3896)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-131 Parameters of the block UA_ReadList

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
NodeHdlCount	IN	WORD	1	Number of elements in the target area to be read. The number of must be 1.
NodeHdls	IN	ARRAY of DWORD	1	Array of the handles of the target nodes (items) of the OPC UA server. The handles are read from array of the output parameter "NodeHdls" of the "UA_NodeGetHandleList" block.
NodeAddInfos	IN	ARRAY of UDT	See "Meaning"	Specifies the attribute of the items to be read and the index range. For information on the structure of the UDT, see UANodeAdditionalInfo (Page 3916).

Parameter	Declaration	S7 data type	Range of values	Meaning
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
Variables	IN_OUT	UDT	See "Meaning"	Reference to the data to be read. For information on the structure of the UDT, see UAAnyPointer (Page 3917). For the supported data types of the process data, refer to the section Data types (Page 3910).
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "ErrorID" parameter.
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 3919).
NodeErrorIDs	OUT	ARRAY of DWORD	See "Meaning"	Contains error numbers for all elements of the data area of "Variables". For the significance of the numbers, refer to the section Error numbers (Page 3919).
TimeStamps	OUT	UDT	See "Meaning"	Contains time stamps for all elements of the data area of "Variables". For information on the structure of the UDT, see UATimeStamp (Page 3917).

UA_WriteList

Function

Function of the block

With the UA_WriteList block you write data to the connected UA server.

You program the attributes to be written in a UDT "UANodeAdditionalInfo" that is referenced by the input parameter "NodeAddInfos".

The handles required at the input parameter "NodeHdls" are read from array of the output parameter "NodeHdls" of the "UA_NodeGetHandleList" block.

The number of nodeIDs at the input parameter "NodeIDCount" is always 1 so you need to call the block separately for each target node.

See also

Parameter (Page 3909)

Program blocks for the OPC UA client (Page 3896)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-132 Parameters of the block UA_WriteList

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
NodeHdlCount	IN	WORD	1	Number of elements in the target area to be written to. The number of must be 1.
NodeHdls	IN	ARRAY of DWORD	1	Array of the handles of the target nodes (items) of the OPC UA server. The handles are read from array of the output parameter "NodeHdls" of the "UA_NodeGetHandleList" block.
NodeAddInfos	IN	ARRAY of UDT	See "Meaning"	Specifies the attribute of the items to be written and the index range. For information on the structure of the UDT, see UANodeAdditionalInfo (Page 3916).
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
Variables	IN_OUT	UDT	See "Meaning"	Reference to the data to be written. For information on the structure of the UDT, see UAAnyPointer (Page 3917). For the supported data types of the process data, refer to the section Data types (Page 3910).
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing

Parameter	Declaration	S7 data type	Range of values	Meaning
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 3919).
NodeErrorIDs	OUT	ARRAY of DWORD	See "Meaning"	Contains error numbers for all elements of the data area of "Variables". For the significance of the numbers, refer to the section Error numbers (Page 3919).

Data types

Data types and memory areas

The CP 443-1 OPC UA supports the data types listed below for process data.

The corresponding symbols on the CPU are permitted for the operand areas listed. The memory areas have the following abbreviations. The short forms in brackets are abbreviations depending on the mnemonics set (German/English).

- I = Process input image (E/I)
- Q = Process output image (A/Q)
- PI = I/O area inputs (PE/PI)
- PQ = I/O area outputs (PA/PQ)
- M = Memory bit
- DB = Data block / system data block / instance data block

I/O areas are only supported by the OPC UA server.

Table 4-133 Data types

Size [bit], type	Area	Data type			Remark	Operand area of the CPU
		SIMATIC S7	IEC 61131-3	OPC UA		
1	0 .. 255	BOOL	BOOL	Boolean	0 = false 1 .. 255 = true	I, Q, PI, PQ, M, DB
Numbers						
8, UInt8	0 .. 255	BYTE	BYTE, USINT	Byte		I, Q, PI, PQ, M, DB
8, UInt8	0 .. 255	CHAR	CHAR	Byte	ASCII characters 32 .. 255	I, Q, PI, PQ, M, DB
16, UInt16	0 .. 65535	WORD	WORD, UINT	UInt16		I, Q, PI, PQ, M, DB
16, Int16	-32768 .. 32767	INT	INT	Int16		I, Q, PI, PQ, M, DB

Size [bit], type	Area	Data type			Remark	Operand area of the CPU
		SIMATIC S7	IEC 61131-3	OPC UA		
32, UInt32	0 .. 4294967295	DWORD	DWORD, UDINT	UInt32		I, Q, PI, PQ, M, DB
32, Int32	2147483648 .. 2147483647	DINT	DINT	Int32		I, Q, PI, PQ, M, DB
32, Float	±1.175 495e-38 .. 3.402823e+38	REAL	REAL	Float		DB
Time and date						
16, UInt16	0 .. 999	S5TIME	TIME	UInt16	SIMATIC time For the coding see online help of STEP 7.	DB
16, UInt16	D#1990-1-1 .. D#2168-12-31 (0 .. 65378)	DATE	DATE	UInt16	IEC date [d]	DB
32, UInt32	TOD#0:0:0.0 .. TOD#23:59:59.999 (0 .. 86399999)	TIME_OF_DAY (TOD)	TIME_OF_DAY	UInt32	Time of day [ms]	DB
32, UInt32	-24D_20H_31M_23S_648MS .. T#24D_20H_31M_23S_647MS (-2147483648 .. 2147483647)	TIME	TIME	UInt32	IEC time [ms]	DB
Complex data types						
64	1990-1-1-0:0:0.0 .. 2089-12-31-23:59:59.999	DATE_AND_TIME	DT DATE_AND_TIME	UInt32+ UInt32	Date and time of day For the coding see online help of STEP 7.	DB
254 * 8		STRING *	STRING	String	Max. 254 bytes user data	DB
		ARRAY *			Multidimensional field of elementary or complex data types	DB
		UDT *			User-defined data type for specific parameter groups	DB

* For the coding see below.

You will find a general description of the coding of the data types in the STEP 7 online help.

Special features of complex data types

With the complex data types used by the OPC UA function of the CP. the special coding rules apply that are described below.

STRING

A STRING can contain a maximum of 254 characters.

ASCII characters from the range 32 (0x20) .. 255 (0xFF) are permitted.

Of these, the following characters are not permitted: 127 (0x7F), 129 (0x81), 141 (0x8D), 143 (0x8F), 144 (0x90), 157 (0x9D)

Special features of the OPC UA server

The characters are coded in the format UTF-8 in the OPC UA server of the CP.

Special features of the OPC UA client

With process data with the data type STRING that are referenced by the "UAAnyPointer", the data type "2" (BYTE) is specified.

As the repetition factor in the UAAnyPointer always select the maximum value of 256 (0x100); 254 for the user data + 2 for the header.

ARRAY

Special features of the OPC UA client

The client program blocks support only the transfer of one dimensional arrays from the memory area "DB".

The transfer of the data type "ARRAY of BOOL" is not supported by the client program blocks.

With a ARRAY of STRING, with the limits of the dimension of every string element the maximum length (256) must be specified "n" times, where "n" is the number of strings in the array.

Special features of the OPC UA server

The data type "ARRAY of BOOL" is readable in all cases.

The data type "ARRAY of BOOL" can only be written when the length is a multiple of 8.

UDT

You can find the structure of the UDTs required by the program blocks for the OPC UA client function in the section AUTOHOTSPOT.

Parameter types

The parameter types "TIMER" and "COUNTER" are only supported for the OPC UA server of the CP.

UDTs for the CP 443-1 OPC UA

UASessionConnectInfo

UASessionConnectInfo

The following table shows the meaning of the connection information for the parameter "SessionConnectInfo" of the UA_Connect block.

Table 4-134 Parameters of UDT UASessionConnectInfo

Parameter	S7 data type	Range of values	Meaning
SessionName	STRING	0 .. 254 characters	Name of the session If the string remains empty, the following is entered by the system as the session name: <Connection>+<decimal value of "ConnectionHdl">
ApplicationName	STRING	0 .. 254 characters	The parameter is given the application name of the CP (OPC UA client) configured In HW Config by STEP 7.
SecurityMsgMode	WORD	0 .. 3	Security process • 0 = Best possible procedure • 1 = No security process • 2 = Authenticate • 3 = Authenticate and encrypt
SecurityPolicy	WORD	0 .. 4	Securityprofile • 0 = Best possible security profile • 1 = No security profile • 2 = Basic128Rsa15 • 3 = Basic256 • 4 = Basic256Sha256
CertificateStore	STRING	0 .. 254 characters	Certificate store of the CP (OPC UA client) The parameter is inserted by STEP 7.
ClientCertificateName	STRING	0 .. 254 characters	Name of the client certificate The parameter is inserted by STEP 7.
ServerUri	STRING	0 .. 254 characters	Server address (URI) that is also stored in the server certificate. The parameter my only be left empty for sessions without a security process. Example: urn:<ApplicationName>:GUID

Parameter	S7 data type	Range of values	Meaning
CheckServerCertificate	BOOL	0, 1	Comparison (check) of the certificate sent by the server with the server certificate stored in the certificate store of the client CP: • 0 = No check • 1 = Check **Note** The settings of this parameter are overwritten by the security settings in the configuration. If you enable the check (1) but the security functions are disabled in the configuration of the CP, connection establishment is aborted with an error message. Recommendation: Leave the parameter at the default setting (0) since the security functions are enabled by the configuration of the CP.
TransportProfile	WORD	1	1 = UATP_UATcp According to the PLCopen specification, only this transport profile is supported.
UserIdentityToken	UDT	See "Meaning"	Data of the user authentication, see UAUserIdentityToken (Page 3915).
VendorSpecificParameter	WORD	See "Meaning"	Entry of the logical address of the CP 443-1 OPC UA. You will find this in STEP 7 properties dialog of the CP as an input address in the "Addresses" tab.
SessionTimeout	TIME	30000 .. 86 400 000	Maximum time that a session remains established when there is no data traffic (milliseconds). If the value is exceeded, the session (connection) is aborted. In this case, you will need to re-establish the connection by calling UA_Connect. If you program values outside the permitted range of values (30 seconds ... 1 day), the value 1200000 (20 minutes) is used.
MonitorConnection	TIME	5000 .. 86 400 000	Connection monitoring time (milliseconds) Time without data traffic after which the client checks the connection to the server by sending a frame. Default setting: 15000 (15 seconds)
LocaleIDs	ARRAY[1..5] of String[6]	Examples: en-US, de-DE, zh-CHS ...	Optional language and regional identifier acc. to RFC 3066. 0 = no or unknown LocaleID.

UAUserIdentityToken

UAUserIdentityToken

The following table shows the meaning of the user authentication data that supplies the UserIdentityToken parameter of the UDT "UASessionConnectInfo".

Table 4-135 Parameters of UDT UAUserIdentityToken

Parameter	S7 data type	Range of values	Meaning
UserIdentity TokenType	WORD	0, 1	0 = No authentication 1 = Authentication When the communications partner (server) demands authentication with a user name and password, set this parameter to 1 and the two following according to the requirements of the server.
TokenParam1	STRING	1 .. 254 characters	User name
TokenParam2	STRING	1 .. 254 characters	Password

UANodeID

UANodeID

The following table shows the meaning of the parameters of UANodeID to identify the target node on the OPC UA server. UANodeID supplies the parameter "NodeIDs" of the UA_NodeGetHandleList" block.

Table 4-136 Parameters of UDT UANodeID

Parameter	S7 data type	Range of values	Meaning
NamespaceIndex	WORD	0 .. 65535	Index of the name space of the server
Identifier	STRING	Max. 254	Specifies the nodeID in the name space index.
IdentifierType	WORD	1 .. 2	Specifies the format and the area of application (generally the server) of the node ID. Supported types: • 1: UAIdentifierType_String String identifiers distinguish upper and lower case. • 2: UAIdentifierType_Numeric Numeric identifier

UANodeAdditionalInfo

UANodeAdditionalInfo

Specifies the attribute of the item and the index range for the block parameter "NodeAddInfos".

Table 4-137 Parameters of UDT UANodeAdditionalInfo

Parameter	S7 data type	Range of values	Meaning
AttributeID	WORD	13	Attribute of the items Only attribute 13 (UAAI_Value) for the value of the item is supported.
IndexRangeCount	WORD	0 .. 1	Number of index ranges The value is unimportant with elementary data types. For items data type ARRAY the following applies: • 0: A single index The entire array is read/written • 1: A subsection of an array defined by "IndexRange".
IndexRange	UDT	See “Meaning”	Area of the array for the parameter IndexRangeCount. For information on the structure of the UDT, see UAIndexRange (Page 3916).

UAIndexRange

UAIndexRange

Specifies the start and end index for the parameter "IndexRange” of the UDT "UANodeAdditionalInfo".

Table 4-138 Parameters of UDT UAIndexRange

Parameter	Data type	Range of values	Meaning
StartIndex	WORD	0 .. 65535	Index as of which reading will take place.
EndIndex	WORD	0 .. 65535	Index up to which reading will take place.

Rules:

If IndexRangeCount = 1 is used, the following rules apply to programming the index range:

- StartinIndex and EndIndex must be assigned.
- For access to more than one element the StartIndex must be lower than the EndIndex. If this is not the case, this causes an error with the number 80360000.

- If you want to read a single element of an array, enter the same value for the StartIndex and EndIndex.
- To be able to access all elements of the array, the StartIndex and EndIndex must be assigned based on the total number of elements of the array.
  If values are specified outside the size of the array, this causes an error with the number 80370000.

Example of assigning "StartIndex" and EndIndex" with arrays.

From an array with a size of 10 elements (no. 1 .. 10) the elements 3 to 5 need to be read. The two indexes are programmed as follows:

- StartIndex = 2 (element no. 3)
- EndIndex = 4 (element no. 5)

UATimeStamp

UATimeStamp

The UDT contains the time stamps of the elements of the data area in milliseconds. UATimeStamp supplies the parameter "TimeStamps" of the block "UA_ReadList".

Table 4-139 Parameters of UDT UATimeStamp

Parameter	S7 data type	Range of values	Meaning
TimestampL	WORD	see below	Least significant word (bits 0..15)
TimestampH	WORD	see below	Most significant word (bits 16..31)

Range of values of the entire double word:

TOD#0:0:0.0 .. TOD#23:59:59.999 (0 .. 86 399 999 milliseconds)

UAAnyPointer

UAAnyPointer

The UDT "UAAnyPointer" is referenced by the "Variables" parameter of the program block.

The UDT references the memory area of the process data in the CPU. To do this process data in the operand areas "inputs" or "outputs" must be copied to a data block.

Store the UDT in a data block.

Table 4-140 Parameters of UDT UAAnyPointer

Parameter	S7 data type	Range of values	Meaning
SyntaxID	WORD	10	The value for the syntax ID is always 10.
DataType	WORD	See "Meaning"	Data type of the target node For the range of values see below, table "Coding of the data type".

Parameter	S7 data type	Range of values	Meaning
RepetitionFactor	WORD	0 .. 65535	Repetition factor For further information, refer to the STEP 7 online help.
DB_Number	WORD	0 .. 65535	Number of the data block (DB) Enter either the number of a DB or a memory area. If you specify a DB, enter a zero in "memArea".
MemArea	WORD	See "Meaning"	Memory area Enter either the number of a DB or a memory area. If you specify a memory area, enter a zero in "DB_Number". For the range of values of the memory area see below, table "Coding of the memory area".
ByteOffset	WORD	Depends on the memory area, see "Meaning"	Byte offset in the specified memory area, as of which the data is accessed.
BitOffset	WORD	0 .. 7	Bit offset in the specified memory area

Data type

The following table explains the coding of the data type in the "DataType" parameter of the UDT UAAnyPointer.

Table 4-141 Coding of the data type

Hexadecimal code	S7 data type	Description
b#16#01	BOOL	Bit
b#16#02	BYTE	Byte (8 bits)
b#16#03	CHAR	Character (8 bits)
b#16#04	WORD	Word (16 bits)
b#16#05	INT	Integer (16 bits)
b#16#06	DWORD	Word (32 bits)
b#16#07	DINT	Integer (32 bits)
b#16#08	REAL	Floating-point number (32 bits)
b#16#09	DATE	Date
b#16#0A	TIME_OF_DAY (TOD)	Time of day
b#16#0B	TIME	Time
b#16#0C	S5TIME	Data type S5TIME
b#16#0E	DATE_AND_TIME (DT)	Date and time (64 bits)
b#16#02 *	STRING	Character string

* The string uses the lower-level data type BYTE. Refer to the special features in the section Data types (Page 3910).

Memory area

The following table explains the coding of the memory area in the "MemArea" parameter of the UDT UAAnyPointer.

Table 4-142 Coding of the memory area

Hexadecimal code	Area	Description
b#16#80	P	Memory area of the I/O
b#16#81	E	Memory area of the inputs
b#16#82	A	Memory area of the outputs
b#16#83	M	Memory area of the bit memory
b#16#84	DB	Data block

Error numbers

Evaluation of the output parameters

The output parameters Done (BOOL), Error (BOOL) and ErrorID (DWORD) of the OPC UA client program blocks are updated each time the block is called. Evaluate the states of the output parameters in the user program.

Example:

After it is called, the program block UA_Connect returns the following values at the three output parameters.

- Done = 1
- Error = 0
- ErrorID = 00000000

Meaning: The job was completed successfully.

Meaning of the error numbers (hexadecimal) of ErrorID

Table 4-143 Meaning of ErrorID in conjunction with Done and Error

Done	Error	ErrorID (hex)	Name	Meaning
1	0	00000000		The block execution was completed successfully.
CP-specific error numbers				
0	1	B0006001		Missing positive edge for the program block call
0	1	B0007001		Timeout
0	1	B0008004		Invalid NodeIDCount
0	1	B0008086		Invalid node handle
0	1	B0008087		The maximum number of connections (5) has been reached.
0	1	B0008090		Wrong logical address. Connection establishment aborted.
0	1	B0008093		The logical address cannot be processed. An impermissible module is being used.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	B00080A1		The node handle was not set up for this connection.
0	1	B00080C3		The connection resources of the CPU are temporarily exhausted.
0	1	B0009085		The maximum memory space for items (64000 bytes) of the OPC UA client has been exceeded.
0	1	B000800A		The OPC client was not enabled in the configuration.
0	1	B0008723		The read/write job references a non-existent data area in a DB.
0	1	B000F001		Two function blocks were called in one cycle.
0	1	B000F002		OPC UA session not yet established. Call the UA_Connect block again.
0	1	B000F003		Communication error
0	1	B000F004		Invalid value for the parameter "SyntaxID" in the UDT "UAAnyPointer"
0	1	B000F005		Invalid data type in UAAnyPointer
0	1	B000F006		Invalid value for the parameter "RepetitionFactor" in the UDT "UAAnyPointer"
0	1	B000F007		Invalid operand area of UAAnyPointer
0	1	B000F010		Invalid operand area in a read or write job: Arrays of inputs, arrays of outputs, arrays of memory bits, arrays of local data and arrays of local data of the predecessor block are not permitted.
0	1	B000F011		The bit offset must be zero.
0	1	B0010001		Wrong data packet in UA_Connect
0	1	B0010002		Invalid URL of a server end point
0	1	B0010003		Invalid value for Timeout
0	1	B0010004		Invalid value for SessionTimeout
0	1	B0010005		Invalid value for MonitorConnection
0	1	B0010006		Invalid transport profile
0	1	B0010007		Invalid server URL
0	1	B0010008		Invalid Security procedure
0	1	B0010009		Invalid Security profile
0	1	B001000A		Invalid value for UserIdentityToken
0	1	B001000B		Invalid application name
0	1	B0010101		Invalid format of the server end point URL. It must start with "opc.tcp://".
0	1	B0010102		The path of the server end point URL is not supported.
0	1	B0010103		The client must not connect itself to the server of its own CP.
0	1	B0010104		Invalid port
0	1	B0010105		Invalid address. DNS is not supported.
0	1	B0020001		Wrong data packet in UA_NameSpaceGetIndex
0	1	B0030001		Wrong data packet in UA_NodeGetHandleList
0	1	B0030002		Invalid NodeID Identifier
0	1	B0030003		Max. number of items (64000) exceeded
0	1	B0030004		Unsupported attribute "ValueRank" of a tag. The ValueRank values 1 (one dimensional array) and -1 (scalar) are supported.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	B0040001		Wrong data packet in UA_NodeReleaseHandleList
0	1	B0050001		Wrong data packet in UA_Disconnect
0	1	B0050002		UA_Disconnect cannot be called, because there is no connection.
0	1	B0060001		Wrong data packet in UA_ReadList
0	1	B0060002		The length of the data to be read exceeds the maximum of 64000.
0	1	B0060003		Invalid range of IndexRangeCount
0	1	B0060004		Invalid IndexRange. StartIndex must not be higher than EndIndex and EndIndex must not be higher than the array length.
0	1	B0060005		Invalid AttributeID
0	1	B0060006		The read data area is too small. Correct the data in the UDT "UAIndexRange".
0	1	B0060007		Invalid string length. The value (user data) must be 254.
0	1	B0060008		Invalid value of the node. The value cannot be read into the S7 tag.
0	1	B0060010		Invalid operand area in a read or write job: Arrays of inputs, arrays of outputs, arrays of memory bits, arrays of local data and arrays of local data of the predecessor block are not permitted.
0	1	B0070001		Wrong data packet in UA_WriteList
0	1	B0070002		The length of the data to be written exceeds the maximum.
0	1	B0070003		Invalid range of IndexRangeCount
0	1	B0070004		Invalid IndexRange. StartIndex must not be higher than EndIndex and EndIndex must not be higher than the array length.
0	1	B0070005		Invalid AttributeID
0	1	B0070006		The data area is outside the permitted area.
0	1	B0070009		Invalid character in the string
Error numbers according to IEC 61131-3 for OPC UA				
0	1	002D0000	GoodSubscriptionTransferred	The subscription was transferred to another session.
0	1	002E0000	GoodCompletesAsynchronously	The processing will complete asynchronously.
0	1	002F0000	GoodOverload	Sampling has slowed down due to resource limitations.
0	1	00300000	GoodClamped	The value written was accepted but was clamped.
0	1	00960000	GoodLocalOverride	The value has been overridden.
0	1	00A20000	GoodEntryInserted	The data or event was successfully inserted into the historical database.
0	1	00A30000	GoodEntryReplaced	The data or event field was successfully replaced in the historical database.
0	1	00A50000	GoodNoData	No data exists for the requested time range or event filter.
0	1	00A60000	GoodMoreData	The data or event field was successfully replaced in the historical database.
0	1	00A70000	GoodCommunicationEvent	The communication layer has raised an event.
0	1	00A80000	GoodShutdownEvent	The system is shutting down.
0	1	00A90000	GoodCallAgain	The operation is not finished and needs to be called again.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	00AA0000	GoodNonCriticalTimeout	A non-critical timeout occurred.
0	1	00BA0000	GoodResultsMayBe Incomplete	The server should have followed a reference to a node in a remote server but did not. The result set may be incomplete.
0	1	406C0000	UncertainReferenceOutOf Server	One of the references to follow in the relative path references to a node in the address space in another server.
0	1	408F0000	UncertainNoCommunication LastUsableValue	Communication to the data source has failed. The variable value is the last value that had a good quality.
0	1	40900000	UncertainLastUsableValue	Whatever was updating this value has stopped doing so.
0	1	40910000	UncertainSubstituteValue	The value is an operational value that was manually overwritten.
0	1	40920000	UncertainInitialValue	The value is an initial value for a variable that normally receives its value from another variable.
0	1	40930000	UncertainSensorNotAccurate	The value is at one of the sensor limits.
0	1	40940000	UncertainEngineeringUnits Exceeded	The value is outside of the range of values defined for this parameter.
0	1	40950000	UncertainSubNormal	The value is derived from multiple sources and has less than the required number of Good sources.
0	1	40A40000	UncertainDataSubNormal	The value is derived from multiple values and has less than the required number of Good values.
0	1	40BC0000	UncertainReferenceNot Deleted	The server was not able to delete all target references.
0	1	40C00000	UncertainNotAllNodes Available	The list of references may not be complete because the underlying system is not available.
0	1	80010000	BadUnexpectedError	An unexpected error occurred
0	1	80020000	BadInternalError	An internal error occurred as a result of a programming or configuration error.
0	1	80030000	BadOutOfMemory	Not enough memory to complete the operation.
0	1	80040000	BadResourceUnavailable	An operating system resource is not available
0	1	80050000	BadCommunicationError	A low level communication error occurred.
0	1	80060000	BadEncodingError	Encoding halted because of invalid data in the objects being serialized.
0	1	80070000	BadDecodingError	Decoding halted because of invalid data in the stream.
0	1	80080000	BadEncodingLimitsExceeded	The message encoding/decoding limits imposed by the stack have been exceeded.
0	1	80B80000	BadRequestTooLarge	The resources (memory) of the CPU are temporarily occupied
0	1	80B90000	BadResponseTooLarge	The response message size exceeds limits set by the client.
0	1	80B90000	BadUnknownResponse	An unrecognized response was received from the server.
0	1	800A0000	BadTimeout	The operation timed out.
0	1	800B0000	BadServiceUnsupported	The server does not support the requested service.
0	1	800C0000	BadShutdown	The operation was cancelled because the application is shutting down.
0	1	800D0000	BadServerNotConnected	The operation could not complete because the client is not connected to the server.
0	1	800E0000	BadServerHalted	The server has stopped and cannot process any requests.
0	1	800F0000	BadNothingToDo	There was nothing to do because the client passed a list of operations with no elements.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	80100000	BadTooManyOperations	The request could not be processed because it specified too many operations.
0	1	80110000	BadDataTypeIdUnknown	The extension object cannot be (de)serialized because the data type id is not recognized.
0	1	80120000	BadCertificateInvalid	The certificate provided as a parameter is not valid.
0	1	80130000	BadSecurityChecksFailed	The certificate provided as a parameter is not valid.
0	1	80140000	BadCertificateTimeInvalid	The Certificate has expired or is not yet valid.
0	1	80150000	BadCertificateIssuer TimeInvalid	An Issuer Certificate has expired or is not yet valid.
0	1	80160000	BadCertificateHost NameInvalid	The HostName used to connect to a Server does not match a HostName in the Certificate.
0	1	80170000	BadCertificateUriInvalid	The URI specified in the Application Description does not match the URI in the Certificate.
0	1	80180000	BadCertificateUseNotAllowed	The Certificate may not be used for the requested operation.
0	1	80190000	BadCertificateIssuerUse NotAllowed	The Issuer Certificate may not be used for the requested operation.
0	1	801A0000	BadCertificateUntrusted	The Certificate is not trusted.
0	1	801B0000	BadCertificateRevocation Unknown	It was not possible to determine if the Certificate has been revoked.
0	1	801C0000	BadCertificateIssuer RevocationUnknown	It was not possible to determine if the Issuer Certificate has been revoked.
0	1	801D0000	BadCertificateRevoked	The Certificate has been revoked.
0	1	801E0000	BadCertificateIssuerRevoked	The Issuer Certificate has been revoked.
0	1	801F0000	BadUserAccessDenied	User does not have permission to perform the requested operation.
0	1	80200000	BadIdentityTokenInvalid	The user identity token is not valid.
0	1	80210000	BadIdentityTokenRejected	The user identity token is valid but the server has rejected it.
0	1	80220000	BadSecureChannelIdInvalid	The specified secure channel is no longer valid.
0	1	80230000	BadInvalidTimestamp	The timestamp is outside the range allowed by the server.
0	1	80240000	BadNonceInvalid	The nonce does appear to be not a random value or it is not the correct length.
0	1	80250000	BadSessionIdInvalid	The session id is not valid.
0	1	80260000	BadSessionClosed	The session was closed by the client.
0	1	80270000	BadSessionNotActivated	The session cannot be used because ActivateSession has not been called.
0	1	80280000	BadSubscriptionIdInvalid	The subscription id is not valid.
0	1	802A0000	BadRequestHeaderInvalid	The header for the request is missing or invalid.
0	1	802B0000	BadTimestampsTo ReturnInvalid	The timestamps to return parameter is invalid.
0	1	802C0000	BadRequestCancelled ByClient	The request was cancelled by the client.
0	1	80310000	BadNoCommunication	Communication with the data source is defined, but not established, and there is no last known value available.
0	1	80320000	BadWaitingForInitialData	Waiting for the server to obtain values from the underlying data source.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	80330000	BadNodeIdInvalid	The syntax of the node id is not valid.
0	1	80340000	BadNodeIdUnknown	The node id refers to a node that does not exist in the server address space.
0	1	80350000	BadAttributeIdInvalid	The attribute is not supported for the specified Node.
0	1	80360000	BadIndexRangeInvalid	The syntax of the index range parameter is invalid.
0	1	80370000	BadIndexRangeNoData	No data exists within the range of indexes specified.
0	1	80380000	BadDataEncodingInvalid	The data encoding is invalid.
0	1	80390000	BadDataEncoding Unsupported	The server does not support the requested data encoding for the node.
0	1	803A0000	BadNotReadable	The access level does not allow reading or subscribing to the Node.
0	1	803B0000	BadNotWritable	The access level does not allow writing to the Node.
0	1	803C0000	BadOutOfRange	The value was out of range.
0	1	803D0000	BadNotSupported	The requested operation is not supported.
0	1	803E0000	BadNotFound	A requested item was not found or a search operation ended without success.
0	1	803F0000	BadObjectDeleted	The object cannot be used because it has been deleted.
0	1	80400000	BadNotImplemented	Requested operation is not implemented.
0	1	80410000	BadMonitoringModeInvalid	The monitoring mode is invalid.
0	1	80420000	BadMonitoredItemIdInvalid	The monitoring item id does not refer to a valid monitored item.
0	1	80430000	BadMonitoredItem FilterInvalid	The monitored item filter parameter is not valid.
0	1	80440000	BadMonitoredItem FilterUnsupported	The server does not support the requested monitored item filter.
0	1	80450000	BadFilterNotAllowed	A monitoring filter cannot be used in combination with the attribute specified.
0	1	80460000	BadStructureMissing	A mandatory structured parameter was missing or null.
0	1	80470000	BadEventFilterInvalid	The event filter is not valid.
0	1	80480000	BadContentFilterInvalid	The content filter is not valid.
0	1	80490000	BadFilterOperandInvalid	The operand used in a content filter is not valid.
0	1	804A0000	BadContinuation PointInvalid	The continuation point provide is longer valid.
0	1	804B0000	BadNoContinuationPoints	The operation could not be processed because all continuation points have been allocated.
0	1	804C0000	BadReferenceTypeIdInvalid	The operation could not be processed because all continuation points have been allocated.
0	1	804D0000	BadBrowseDirectionInvalid	The browse direction is not valid.
0	1	804E0000	BadNodeNotInView	The node is not part of the view.
0	1	804F0000	BadServerUriInvalid	The ServerUri is not a valid URI.
0	1	80500000	BadServerNameMissing	No ServerName was specified
0	1	80510000	BadDiscoveryUrlMissing	No DiscoveryUrl was specified.
0	1	80520000	BadSempahoreFileMissing	The semaphore file specified by the client is not valid.
0	1	80530000	BadRequestTypeInvalid	The security token request type is not valid.
0	1	80540000	BadSecurityModeRejected	The security mode does not meet the requirements set by the Server.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	80550000	BadSecurityPolicyRejected	The security policy does not meet the requirements set by the Server.
0	1	80560000	BadTooManySessions	The server has reached its maximum number of sessions.
0	1	80570000	BadUserSignatureInvalid	The user token signature is missing or invalid.
0	1	80580000	BadApplicationSignature Invalid	The signature generated with the client certificate is missing or invalid.
0	1	80590000	BadNoValidCertificates	The client did not provide at least one software certificate that is valid and meets the profile requirements for the server.
0	1	805A0000	BadRequestCancelled ByRequest	The request was cancelled by the client with the Cancel service.
0	1	805B0000	BadParentNodeIdInvalid	The parent node id does not to refer to a valid node.
0	1	805C0000	BadReferenceNotAllowed	The reference could not be created because it violates constraints imposed by the data model.
0	1	805D0000	BadNodeIdRejected	The requested node id was reject because it was either invalid or server does not allow node ids to be specified by the client.
0	1	805F0000	BadNodeClassInvalid	The node class is not valid.
0	1	80600000	BadBrowseNameInvalid	The browse name is invalid.
0	1	80610000	BadBrowseNameDuplicated	The browse name is not unique among nodes that share the same relationship with the parent.
0	1	80620000	BadNodeAttributesInvalid	The node attributes are not valid for the node class.
0	1	80630000	BadTypeDefinitionInvalid	The type definition node id does not reference an appropriate type node.
0	1	80640000	BadSourceNodeIdInvalid	The source node id does not reference a valid node.
0	1	80650000	BadTargetNodeIdInvalid	The target node id does not reference a valid node.
0	1	80660000	BadDuplicateReference NotAllowed	The reference type between the nodes is already defined.
0	1	80670000	BadInvalidSelfReference	The server does not allow this type of selfreference on this node.
0	1	80680000	BadReferenceLocalOnly	The reference type is not valid for a reference to a remote server.
0	1	80690000	BadNoDeleteRights	The server will not allow the node to be deleted.
0	1	806A0000	BadServerIndexInvalid	The server index is not valid.
0	1	806B0000	BadViewIdUnknown	The view id does not refer to a valid view node.
0	1	806D0000	BadTooManyMatches	The requested operation has too many matches to return.
0	1	806E0000	BadQueryTooComplex	The requested operation requires too many resources in the server.
0	1	806F0000	BadNoMatch	The requested operation has no match to return.
0	1	80700000	BadMaxAgeInvalid	The max age parameter is invalid.
0	1	80710000	BadHistoryOperationInvalid	The history details parameter is not valid.
0	1	80720000	BadHistoryOperation Unsupported	The server does not support the requested operation.
0	1	80730000	BadWriteNotSupported	The server not does support writing the combination of value, status and timestamps provided.
0	1	80740000	BadTypeMismatch	The value supplied for the attribute is not of the same type as the attribute's value.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	80750000	BadMethodInvalid	The method id does not refer to a method for the specified object.
0	1	80760000	BadArgumentsMissing	The client did not specify all of the input arguments for the method.
0	1	80770000	BadTooManySubscriptions	The server has reached its maximum number of subscriptions.
0	1	80780000	BadTooManyPublish Requests	The server has reached the maximum number of queued publish requests.
0	1	80790000	BadNoSubscription	There is no subscription available for this session.
0	1	807A0000	BadSequenceNumber Unknown	The sequence number is unknown to the server.
0	1	807B0000	BadMessageNotAvailable	The requested notification message is no longer available.
0	1	807C0000	BadInsufficientClientProfile	The Client of the current Session does not support one or more Profiles that are necessary for the Subscription.
0	1	80BF0000	BadStateNotActive	The sub-state machine is not currently active.
0	1	807D0000	BadTcpServerTooBusy	The server cannot process the request because it is too busy.
0	1	807E0000	BadTcpMessageTypeInvalid	The type of the message specified in the header invalid.
0	1	807F0000	BadTcpSecureChannel Unknown	The SecureChannelId and/or TokenId are not currently in use.
0	1	80800000	BadTcpMessageTooLarge	The size of the message specified in the header is too large.
0	1	80810000	BadTcpNotEnough Resources	There are not enough resources to process the request.
0	1	80820000	BadTcpInternalError	An internal error occurred.
0	1	80830000	BadTcpEndpointUrlInvalid	The Server does not recognize the QueryString specified.
0	1	80840000	BadRequestInterrupted	The request could not be sent because of a network interruption.
0	1	80850000	BadRequestTimeout	Timeout occurred while processing the request.
0	1	80860000	BadSecureChannelClosed	The secure channel has been closed.
0	1	80870000	BadSecureChannelToken Unknown	The token has expired or is not recognized.
0	1	80880000	BadSequenceNumberInvalid	The sequence number is not valid.
0	1	80890000	BadConfigurationError	There is a problem with the configuration that affects the usefulness of the value.
0	1	808A0000	BadNotConnected	The variable should receive its value from another variable, but has never been configured to do so.
0	1	808B0000	BadDeviceFailure	There has been a failure in the device/data source that generates the value that has affected the value.
0	1	808C0000	BadSensorFailure	There has been a failure in the sensor from which the value is derived by the device/data source.
0	1	808D0000	BadOutOfService	The source of the data is not operational.
0	1	808E0000	BadDeadbandFilterInvalid	The dead band filter is not valid.
0	1	80970000	BadRefreshInProgress	This Condition refresh failed, a Condition refresh operation is already in progress.
0	1	80980000	BadConditionAlreadyDisabled	This condition has already been disabled.
0	1	80990000	BadConditionDisabled	Property not available, this condition is disabled.
0	1	809A0000	BadEventIdUnknown	The specified event id is not recognized.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	809B0000	BadNoData	No data exists for the requested time range or event filter.
0	1	809D0000	BadDataLost	Data is missing due to collection started/stopped/lost.
0	1	809E0000	BadDataUnavailable	Expected data is unavailable for the requested time range due to an unmounted volume an off-line archive or tape or similar reason for temporary unavailability.
0	1	809F0000	BadEntryExists	The data or event was not successfully inserted because a matching entry exists.
0	1	80A00000	BadNoEntryExists	The data or event was not successfully updated because no matching entry exists.
0	1	80A10000	BadTimestampNotSupported	The client requested history using a timestamp format the server does not support (i. e. requested ServerTimestamp when server only supports SourceTimestamp).
0	1	80AB0000	BadInvalidArgument	One or more arguments are invalid.
0	1	80AC0000	BadConnectionRejected	Could not establish a network connection to remote server.
0	1	80AD0000	BadDisconnect	The server has disconnected from the client.
0	1	80AE0000	BadConnectionClosed	The network connection has been closed.
0	1	80AF0000	BadInvalidState	The operation cannot be completed because the object is closed uninitialized or in some other invalid state.
0	1	80B00000	BadEndOfStream	Cannot move beyond end of the stream.
0	1	80B10000	BadNoDataAvailable	No data is currently available for reading from a non-blocking stream.
0	1	80B20000	BadWaitingForResponse	The asynchronous operation is waiting for a response.
0	1	80B30000	BadOperationAbandoned	The asynchronous operation was abandoned by the caller.
0	1	80B40000	BadExpectedStreamToBlock	The stream did not return all data requested (possibly because it is a non-blocking stream).
0	1	80B50000	BadWouldBlock	Non-blocking behavior is required and the operation would block.
0	1	80B60000	BadSyntaxError	A value had an invalid syntax.
0	1	80B70000	BadMaxConnections Reached	The operation could not be finished because all available connections are in use.
0	1	80BB0000	BadEventNot Acknowledgeable	The event cannot be acknowledged.
0	1	80BD0000	BadInvalidTimestamp Argument	The defined timestamp to return was invalid.
0	1	80BE0000	BadProtocolVersion Unsupported	The applications do not have compatible protocol versions.
0	1	80C10000	BadFilterOperatorInvalid	An unrecognized operator was provided in a filter.
0	1	80C20000	BadFilterOperator Unsupported	A valid operator was provided, but the server does not provide support for this filter operator.
0	1	80C30000	BadFilterOperandCount Mismatch	The number of operands provided for the filter operator was less than expected for the operand provided.
0	1	80C40000	BadFilterElementInvalid	The referenced element is not a valid element in the content filter.
0	1	80C50000	BadFilterLiteralInvalid	The referenced literal is not a valid value.
0	1	80C90000	BadViewTimestampInvalid	The view timestamp is not available or not supported.
0	1	80CA0000	BadViewParameterMismatch	The view parameters are not consistent with each other.
0	1	80CB0000	BadViewVersionInvalid	The view version is not available or not supported.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	80CC0000	BadConditionAlready Enabled	This condition has already been enabled.
0	1	80CD0000	BadDialogNotActive	The dialog condition is not active.
0	1	80CF0000	BadConditionBranch AlreadyAcked	The condition branch has already been acknowledged.
0	1	80D00000	BadConditionBranch AlreadyConfirmed	The condition branch has already been confirmed.
0	1	80D10000	BadConditionAlready-Shelved	The condition has already been shelved.
0	1	80D20000	BadConditionNotShelved	The condition is not currently shelved.
0	1	80D30000	BadShelvingTimeOutO-fRange	The shelving time not within an acceptable range.
0	1	80D40000	BadAggregateListMismatch	The requested number of Aggregates does not match the requested number of NodeIds.
0	1	80D50000	BadAggregateNotSupported	The requested Aggregate is not support by the server.
0	1	80D60000	BadAggregateInvalidInputs	The aggregate value could not be derived due to invalid data inputs.
0	1	80DB0000	BadTooManyMonitoredItems	The request could not be processed because there are too many monitored items in the subscription.
0	1	80D70000	BadBoundNotFound	No data found to provide upper or lower bound value.
0	1	80D80000	BadBoundNotSupported	The server cannot retrieve a bound for the variable.
0	1	00D90000	GoodDataIgnored	The request specifies fields which are not valid for the EventType or cannot be saved by the historian.
0	1	80DA0000	BadAggregateConfiguration Rejected	The aggregate configuration is not valid for specified node.
Error numbers of PLCOpen for OPC UA				
0	1	A0000001	PLCopenUA_Bad_FW_ PermanentError	Internal, permanent error.
0	1	A0000002	PLCopenUA_Bad_FW_ TempError	Temp. Error; FB could retry to reach FW.
0	1	A0000003	PLCopenUA_Bad_ ConnectionError	Connection could not be established.
0	1	A0000004	PLCopenUA_Bad_ HostNotFound	The requested hostname could not be found.
0	1	A0000005	PLCopenUA_Bad_ AlreadyConnected	Connection was already established.
0	1	A0000006	PLCopenUA_Bad_ SecurityFailed	Connection failed due to security setup.
0	1	A0000007	PLCopenUA_Bad_ Suspended	Connection is suspended.
0	1	A0000008	PLCopenUA_Bad_ ConnectionInvalidHdl	Provided ConnectionHdl is not known.
0	1	A0000009	PLCopenUA_Bad_ NSNotFound	A namespace with the requested name cannot be found on server.
0	1	A000000A	PLCopenUA_Bad_ ResultTooLong	Target PLC variable is too short for retrieved data.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	A000000B	PLCopenUA_Bad_ InvalidType	Invalid or unsupported Type.
0	1	A000000C	PLCopenUA_Bad_ NodeInvalidHdl	Provided NodeHdl is not known.
0	1	A000000D	PLCopenUA_Bad_ MethodInvalidHdl	Provided MethodHdl is not known.
0	1	A000000E	PLCopenUA_Bad_ ReadFailed	Read failed for unknown reason.
0	1	A000000F	PLCopenUA_Bad_ WriteFailed	Write failed for unknown reason.
0	1	A0000010	PLCopenUA_Bad_ CallFailed	Method Call failed for unknown reason.
0	1	A0000011	PLCopenUA_Bad_ InParamFailed	Method Call Input parameter conversion failed.
0	1	A0000012	PLCopenUA_Bad_ OutParamFailed	Method Call Output parameter conversion failed. ATTEN-TION: this means the MethodCall was executed successfully but the returned values could not be converted.
0	1	A0000013	PLCopenUA_Bad_ SubscriptionInvalidHdl	Provided SubscriptionHdl is not known.
0	1	A0000014	PLCopenUA_Bad_ MonitoredItemInvalidHdl	Provided MonitoredItemHdl is not known.

Point-to-point (S7-1200, S7-1500)

Overview of Freeport communication (S7-1200, S7-1500)

STEP 7 offers extended instructions that can be used for Freeport communication with a protocol specified in the user program. These instructions can be divided into two categories:

- Configuration instructions
- Communication instructions

Note

CPU capacity

The Freeport instructions communicate with the communication modules by reading or writing data records.
When using the instructions, you should therefore check the capacity of the CPU for reading and writing data records.
If several instructions need to read or write data records simultaneously on a CPU, the user program may need to create a time offset between the calls of each instruction.

Configuration instructions

Before the user program can start the Freeport communication, the communication interface and the parameters for sending and receiving of data must be configured.

The interface configuration and the data configuration can be set for each CM in the device configuration or with the following instructions of your user program:

- Port_Config (Page 3934)
- Send_Config (Page 3937)
- Receive_Config (Page 3939)
- P3964_Config (Page 3944)

NOTICE
Device configuration <-> Configuration instructions The device configuration parameters are transferred to the CM upon each Power On of the CPU (return of voltage). The parameters of the configuration instructions are transferred to the CM as defined in your user program. The parameters of the device configuration are not synchronized with the parameters of the configuration instructions, which means the parameters of the configuration instructions are not applied to the CPU device configuration. With your user program, you determine the parameters that apply in the CM and when they apply.

Communication instructions

The user program uses the instructions for Freeport communication to send data to and receive data from the communication interfaces. The CMs send data to and receive data from the communication stations.

- Send_P2P (Page 3946)
- Receive_P2P (Page 3949)

Note

Data consistency

- If the data to be sent is transmitted consistently, it cannot be changed after the positive edge at the REQ parameter until DONE has been set by the Send_P2P instruction.
- If the receive data is to be read consistently, it may only be evaluated in the cycle in which NDR = TRUE.

The receive buffer can be reset with additional instructions and special RS232 signals can be queried and set.

- Receive_Reset (Page 3951)
- Signal_Get (Page 3952)
- Signal_Set (Page 3953)

The following instructions let you read or write extended functions, as long as these are supported by the module.

- Get_Features (Page 3955)
- Set_Features (Page 3956)

All Freeport instructions work asynchronously. The instructions must therefore be called until the DONE output parameter indicates that execution is complete.

The user program can determine the send and receive status with the help of the query architecture. Send_P2P and Receive_P2P can be run at the same time. The communication modules buffer the send and receive data as required until a module-specific maximum buffer size has been reached.

Note

Resolution of bit times

The number of bit times is specified with the configured data transmission rate for different parameters. Specifying the parameter in bit times makes it independent of the data transmission rate. All parameters with unit of bit times can be specified with a maximum number of 65535.

Using the instructions (S7-1200, S7-1500)

The Freeport instructions must be called cyclically to query received data or the end of transmission for a send process.

Depending on the data volume, data transmission may take place over several calls (program cycles).
If a command is completed with DONE = TRUE or NDR = TRUE, it has been executed without errors.

Note

Backing up STATUS

The DONE, NDR, ERROR and STATUS parameters are only available for one block cycle. To display the STATUS, you should therefore copy it to a free data area.

Master

Typical sequence for a master:

1. The Send_P2P instruction triggers transmission to the CM.
   Data transmission is initiated by a positive edge at the REQ input.
2. The Send_P2P instruction is executed in subsequent cycles to query the status of the transmission process.
3. When the Send_P2P instruction signals that transmission is complete at the DONE output, the user code can prepare the receipt of the answer.
4. The Receive_P2P instruction is run repeatedly to query an answer. If the CM has acquired response data, the Receive_P2P instruction copies the response to the CPU and signals that new data has been received at the NDR output.
5. The user program can process the response.
6. Back to step 1 and repetition of the sequence.

Slave

Typical sequence for a slave:

1. The user program runs the Receive_P2P instruction in each cycle.
2. If the CM has received a request, the Receive_P2P signals that new data is available at the NDR output and the request is copied to the CPU.
3. The user program processes the request and creates a response.
4. The response is returned to the master with the Send_P2P instruction.
5. The Send_P2P instruction must be run repeatedly to ensure that the send process is actually taking place.
6. Back to step 1 and repetition of the sequence.

The slave must ensure that Receive_P2P is called up often enough so that a transmission can be received by the master before it cancels the process due to a timeout while waiting for the response. To do so, the user program Receive_P2P can be called from within a cycle OB whose cycle time is sufficiently short so that the master can receive a transmission before the timeout setting expires. If the OB cycle time is set so that two runs can take place during the timeout setting of the master, the user program can receive all transmissions without any losses.

General parameters for Freeport operations (S7-1200, S7-1500)

Table 4-144 General input parameters of the Freeport instructions

Parameter	Description
REQ	Data transmission is initiated by a positive edge at the REQ input. Another edge at REQ may only be generated after the command has been completed (DONE or ERROR). Data transmission can take several calls (program cycles), depending on the data volume. When you add a Freeport instruction to your program, STEP 7 prompts you to specify the instance DB (or to have STEP 7 create a corresponding instance DB). Use a unique DB for each PtP instruction call.
PORT	A port address is assigned during configuration of the communication module. The PORT parameter communicates assignment to a specific communication module to the instruction. You can select a symbolic name for the standard port after the configuration. The assigned CM port value is the "Hardware ID" property of the device configuration with S7-1200/1500 and the "Input address" with S7-300/400. The symbolic port name is assigned in the symbol table.

The output parameters DONE, NDR, ERROR and STATUS of the Freeport instructions indicate the execution status of the Freeport functions.

Table 4-145 Output parameters DONE, NDR, ERROR and STATUS

Parameter	Data type	Standard	Description
DONE	Bool	FALSE	Set to TRUE for one cycle to indicate that the last request was completed with errors; otherwise FALSE.
NDR	Bool	FALSE	Set to TRUE for one cycle to indicate that new data has been received; otherwise FALSE.
ERROR	Bool	FALSE	Set to TRUE for one cycle to indicate that the last request was completed with errors; the corresponding error code can be found in STATUS; otherwise FALSE.
STATUS	Word	16#0000 or 16#7000	Result status: • If the DONE or NDR bit is set, STATUS is set to 0/16#7000 or to a specific error code. • If the ERROR bit is set, STATUS displays an error code. • If none of the bits listed above is set, the instruction can return status results that describe the current status of the function. The value in STATUS is valid until you call this instruction again (with one and the same port address).

Table 4-146 In/out parameter COM_RST

Parameter	Data type	Standard	Description
COM_RST	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. COM_RST is then set back to FALSE. Note: You must set COM_RST to TRUE during startup and should not subsequently change the parameter (that is, do not assign a value when you call the instruction). COM_RST is reset by the instruction following initialization of the instance DB.

Note

Please note that the parameters DONE, NDR, ERROR and STATUS are only set for one cycle.

Table 4-147 Shared error codes

Error code	Description
16#0000	No error
16#7000	Function not active
16#7001	Initial call after request started.
16#7002	Subsequent call after request started.
16#8x3A	Invalid pointer in parameter x

Table 4-148 Shared error classes of the STATUS parameter

Description of the class	Error classes	Description
Port configuration	16#81Ax	For the description of frequent errors in the interface configuration
Send configuration	16#81Bx	For the description of errors in the send configuration
Receive configuration	16#81Cx	For the description of errors in the receive configuration
Sending	16#81Dx	For the description of runtime errors during sending
Receiving	16#81Ex	For the description of runtime errors during receiving
RS232 accompanying signals	16#81Fx	For the description of errors in connection with signal processing

Port_Config: Configure PtP communication port (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Port_Config instruction (port configuration) allows you to change parameters such as the data transmission rate in runtime using your program. The data pending in the CM is deleted with the execution of Port_Config.

Configuration changes of Port_Config are saved on the CM and not in the CPU. When the voltage returns, the CM is configured with the data saved in the device configuration.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
PROTO-COL	IN	UInt	Word	0	Protocol • 0 = Freeport protocol • 1 = Protocol 3964(R)
BAUD	IN	UInt	Word	6	Data transmission rate of the port: • 1 = 300 bps • 2 = 600 bps • 3 = 1200 bps • 4 = 2400 bps • 5 = 4800 bps • 6 = 9600 bps • 7 = 19200 bps • 8 = 38400 bps • 9 = 57600 bps • 10 = 76800 bps • 11 = 115200 bps
PARITY	IN	UInt	Word	1	Parity of the port: • 1 = no parity • 2 = even parity • 3 = odd parity • 4 = mark parity • 5 = space parity • 6 = any
DATABITS	IN	UInt	Word	1	Bits per character: • 1 = 8 data bits • 2 = 7 data bits

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
STOPBITS	IN	UInt	Word	1	Stop bits: • 1 = 1 stop bit • 2 = 2 stop bits
FLOWCTRL	IN	UInt	Word	1	Flow control: • 1 = no flow control • 2 = XON/XOFF • 3 = Hardware RTS always ON • 4 = Hardware RTS switched • 5 = Hardware RTS always ON, ignore DTR/DSR
XON-CHAR	IN	Char		16#0011	Specifies the character that serves as XON character. It is typically a DC1 character (11H). This parameter is only evaluated when software flow control is active.
XOFF-CHAR	IN	Char		16#0013	Specifies the character that serves as XOFF character. It is typically a DC3 character (13H). This parameter is only evaluated when software flow control is active.
WAITIME	IN	UInt	Word	2000	Specifies how long to wait for an XON character after receipt of an XOFF character or how long to wait for a CTS = ON signal after CTS = OFF (0 to 65535 ms). This parameter is only evaluated when flow control is active.
MODE	IN	USInt	Byte	0	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master; CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave; CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode [1)]
LINE_PRE	IN	USInt	Byte	0	Receive line initial state Valid initial states are: • 0 = "No" initial state [1)] • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire mode (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
BRK_DET	IN	USInt	Byte	0	Break detection The following settings are permitted: • 0 = break detection deactivated • 1 = break detection activated
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))

[1] Required setting for the use of PROFIBUS cables with CM 1241 for RS485

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

Send_Config: Configure PtP sender (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Send_Config instruction (send configuration) allows you to change send parameters in runtime using your program (conditions that identify the start and the end of the data to be sent). Any data pending in a CM is deleted when Send_Config is executed.

Configuration changes of Send_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Activates the configuration change upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
RTSOND-LY	IN	UInt	Word	0	Number of milliseconds to wait after activation of RTS before a transmission of send data is started. This parameter is only valid if the hardware flow control is active. The valid range is 0 to 65535 ms. The value 0 deactivates the function.
RTSOFFDLY	IN	UInt	Word	0	Number of milliseconds to wait after transmission of send data before RTS is deactivated: This parameter is only valid if the hardware flow control is active. The valid range is 0 to 65535 ms. The value 0 deactivates the function.
BREAK	IN	UInt	Word	0	This parameter specifies that a BREAK is to be sent at the start of each frame for the specified number of bit times. The maximum is 65535 bit times. The value 0 deactivates the function.
IDLELINE	IN	UInt	Word	0	This parameter specifies that the line is to remain in idle for the specified number of bit times prior to the start of each frame. The maximum is 65535 bit times. The value 0 deactivates the function.
USR_END	IN	STRING[2]		0	Input of end delimiters. No more than 2 end delimiters can be configured. All data including the end delimiter(s) is sent, independent of the configured frame length.
APP_END	IN	STRING[5]		0	Input of characters to be appended. You can append up to 5 characters.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

Receive_Config: Configure PtP recipient (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Receive_Config instruction (receive configuration) allows you to change receive parameters in runtime using your program. This instruction configures the conditions that mark the start and the end of received data. Any data pending in a CM is deleted when Receive_Config is executed.

Configuration changes of Receive_Config are saved **non-retentive** on the CM. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module. The Receive_Config instruction therefore must be called again from the user program when the voltage returns to the CPU or to the communication module in order to overwrite the parameters stored in the device configuration.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Activates the configuration change upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
RECEIVE_CONDITIONS	IN	Variant	Any	-	The data structure of Receive_Conditions specifies the start and end conditions used to identify the start and end of a frame.

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

Start conditions for the Receive_P2P instruction

The Receive_P2P instruction uses the configuration specified in the device configuration or by the Receive_Config instruction to determine the start and end of Freeport communication frames. The start of the frame is defined by the start conditions. The start of the frame can be determined with one or several start conditions.
If Break as well as Idle Line is activated, Break must be met first and then Idle Line as well. After that, one of the other conditions (start character or start sequence) is sufficient to start data transmission.
The start condition "Any character" cannot be combined with other start conditions.

Data type structure of the Receive_Conditions parameter, part 1 (start conditions)

Table 4-149 Structure of Receive_Conditions for start conditions

Parameter	Declaration	Data type	Default	Description
START.START-COND	IN	Word	16#0002	Specifying the start condition • 01H - detection of the start character • 02H - Any character • 04H - detection of a line break • 08H - detection of an idle line • 10H - detection of start sequence 1 • 20H - detection of start sequence 2 • 40H - detection of start sequence 3 • 80H - detection of start sequence 4 The start conditions can be combined by adding the values together.
START.IDLE-TIME	IN	Word	16#0028	The number of bit times required in the idle state for a new frame start to be detected (default value: W#16#28). Only in connection with the condition "Detection of an idle line". 0 to FFFF

Parameter	Declaration	Data type	Default	Description
START.START CHAR	IN	Byte	16#0002	The start character for the condition "Start character". (default value: B#16#2)
START.SEQ[1]. CTL	IN	Byte	0	Start sequence 1, deactivate/activate comparison for each character: (default value: B#16#0) These are the activation bits for each character of the start character string. • 01H - character 1 • 02H - character 2 • 04H - character 3 • 08H - character 4 • 10H - character 5 When a bit is deactivated for a specific character, this means that each character in this position in the character string represents a valid start character string (e.g. 1FH = all 5 characters interpreted).
START.SEQ[1]. STR[1] .. START.SEQ[1]. STR.[5]	IN	Char[5]	0	Start sequence 1, start character (5 characters).
START.SEQ[2]. CTL	IN	Byte	0	Start sequence 2, deactivate/activate comparison for each character. Default value: B#16#0)
START.SEQ[2]. STR[1] .. START.SEQ[2]. STR.[5]	IN	Char[5]	0	Start sequence 2, start character (5 characters).
START.SEQ[3]. CTL	IN	Byte	0	Start sequence 3, deactivate/activate comparison for each character. Default value: B#16#0
START.SEQ[3]. STR[1] .. START.SEQ[3]. STR.[5]	IN	Char[5]	0	Start sequence 3, start character (5 characters).
START.SEQ[4]. CTL	IN	Byte	0	Start sequence 4, deactivate/activate comparison for each character. Default value: B#16#0
START.SEQ[4]. STR[1] .. START.SEQ[4]. STR.[5]	IN	Char[5]	0	Start sequence 4, start character (5 characters),

Example

Have a look at the following received data in hexadecimal coding: "**68** 10 aa **68** bb 10 aa 16". The configured start character strings are available in the following table. Start character strings are evaluated once the first character 68H has been successfully received. After the fourth character has been successfully received (the second 68H), start condition 1 has been met. Once the start conditions have been met, the evaluation of the end conditions starts.

Processing of the start character string can be canceled due to different errors in parity, framing or time intervals between characters. These errors prevent receipt of the data because the start condition has not been met (an error message is output).

Table 4-150 Start conditions

Start condition	First character	First character +1	First character +2	First character +3	First character +4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the Receive_P2P instruction

The end of a frame is defined by the first occurrence of one or more configured end conditions.

You can configure the end conditions either in the properties of the communication interface in the device configuration, or with the Receive_Config instruction. The receive parameters (start and end conditions) are reset to the settings in the device configuration each time the voltage returns to the CPU or the communication module. When the STEP 7 user program executes Receive_Config, the settings are set to the parameters of Receive_Config .

Data type structure of the Receive_Conditions parameter, part 2 (end conditions)

Table 4-151 Structure of Receive_Conditions for end conditions

Parameter	Declaration	Data type	Default	Description
END.END-COND	IN	Word	0	This parameter specifies the condition for the frame end: • 01H - response timeout • 02H - message timeout • 04H - character delay time • 08H - maximum frame length • 10H - read message length from message (N+LEN+M) • 20H - end sequence • 40H - fixed frame length
END.FIXLEN	IN	Word	1	Fixed frame length: Only used if the end condition "Fixed frame length" has been selected. 1 to 4000 bytes (up to 4 KB depending on the module)
END.MAXLEN	IN	Word	1	Maximum frame length: Only used if the end condition "Maximum frame length" has been selected. 1 to 4000 bytes (up to 4 KB depending on the module)
END.N	IN	Word	0	Byte position of the length field in the frame. Only used with end condition N+LEN+M. 1 to 4000 bytes (up to 4 KB depending on the module)
END.LENGTH-SIZE	IN	Word	0	Size of the length field (1, 2, or 4 bytes). Only used with end condition N+LEN+M.

Parameter	Declaration	Data type	Default	Description
END.LENGTHM	IN	Word	0	Number of characters after the length field that are not included in the value of the length field. This entry is only used with end condition N+LEN+M. 0 to 255 bytes
END.RCVTIME	IN	Word	200	Specify the wait time for the first received character after a frame has been sent. The receive instruction is terminated with an error message if a character is not received within the specified time. This information is used only with the condition "Response timeout". (0 to 65535 ms). Note: This parameter cannot be used as sole end criterion but only in connection with at least one other end condition.
END.MSGTIME	IN	Word	200	Specify how long to wait for receipt of the complete frame after receipt of the first character. This parameter is used only if the condition "Message timeout" is selected. (0 to 65535 ms)
END.CHARGAP	IN	Word	12	Enter the maximum number of bit times between characters. If the number of bit times between characters exceeds the specified value, the end condition has been met. This information is used only with the condition "Character delay time". (0 to 65535 bit times)
END.SEQ.CTL	IN	Byte	0	Character sequence 1, deactivate/activate comparison for each character: These are the activation bits for each character of the end character string. Character 1 is bit 0, character 2 is bit 1, ..., character 5 is bit 4. If a bit is deactivated for a specific character, this means that each character represents a congruence at this position of the character string.
END.SEQ.STR[1] .. END.SEQ.STR[5]	IN	Char[5]	0	Character string 1, start character (5 characters)

Table 4-152 General parameters of the Receive_P2P instruction

Parameter	Declaration	Data type	Default	Description
GENER-AL.MBUF_SIZE	IN	Byte	255	Input number of frames that are to be buffered in the receive buffer of the CM. If no other conditions are active that influence the reaction of the receive buffer (prevent timeout, data flow control), additional frames are discarded once the limit has been reached. (1 to 255 frames)
GENER-AL.OW_PROT	IN	Byte	0	Activates the no overwriting function of the buffered frame if the CM receives a new frame and the receive buffer of the CM was not yet read. This step prevents already buffered received frames from being lost. • 0 - not activated • 1 - activated
GENER-AL.CLR_MBUF	IN	Byte	0	Activates deletion of the receive buffer during CPU startup. The receive buffer is automatically deleted when the CPU switches from STOP to RUN. The receive buffer only contains frames received after CPU startup. • 0 - not activated • 1 - activated

P3964_Config: Configuring the 3964(R) protocol (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The P3964_Config instruction (protocol configuration) allows you to change protocol parameters for 3964(R), such as character delay time, priority and block check, in runtime using your program.

Configuration changes of P3964_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
BCC	IN	USInt	Byte	1	Activates/deactivates the use of the block check • 0 = without block check • 1 = with block check
Priority	IN	USInt	Byte	1	Selection of the priority • 0 = low priority • 1 = high priority
CharacterDelayTime	IN	UInt	Word	16#00DC	Setting the character delay time (depending on the set data transmission rate) (default value: 220 ms) 1 ms to 65535 ms
AcknDelayTime	IN	UInt	Word	16#07D0	Setting the acknowledgment delay time (depending on the set data transmission rate) (default value: 2000 ms) 1 ms to 65535 ms
BuildupAttempts	IN	USInt	Byte	16#0006	Setting the number of connection attempts (default value: 6 connection attempts) 1 to 255
RepetitionAttempts	IN	USInt	Byte	16#0006	Setting the number of transmission attempts (default value: 6 connection attempts) 1 to 255
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

Send_P2P: Sending data (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Send_P2P instruction (send point-to-point data) starts the transmission of data and transmits the contents of the assigned buffer to the communication module. The CPU program is still being executed while the CM sends the data with the data transmission rate. Only one send instruction per communication module may be pending at any time. The CM signals an error if a second Send_P2P instruction is executed while the CM is already sending a frame.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
BUFFER	IN	Variant	Any	0	This parameter points to the memory area of the send buffer. **Notes:** • Boolean data and Boolean fields are not supported. • If the send buffer is in the optimized memory area, the maximum permitted length of the sent data is 1024 bytes. Exception: Arrays of Byte, Word or DWord are supported up to a length of 4096 bytes. • If the send buffer is a String or WString, the content of the string is transferred without the current and maximum length. Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 3948)"
LENGTH	IN	UInt	Word	0	Length in bytes of the data to be transferred. The memory area addressed in the BUFFER parameter is completely transmitted with LENGTH = 0. Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 3948)"
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Send_P2P instruction The instruction is initialized with 1. The instruction then resets COM_RST to 0. Note: The parameter is only available for S7-300/400 instructions.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

Parameters

The DONE and ERROR outputs are in FALSE status when a send instruction is being processed. At the end of the send instruction, one of the DONE or ERROR outputs is set to TRUE for one cycle to signal the status of the send instruction. The error code at the STATUS output can be evaluated when the status of ERROR is TRUE.

The instruction outputs the status 16#7001 when the communication interface accepts the send data. Subsequent executions of Send_P2P output the value 16#7002 if the CM is still sending. At the end of the send instruction, the CM outputs the status 16#0000 for the send

instruction (if no error has occurred). Subsequent executions of Send_P2P with REQ = 0 output the status 16#7000 (free).

The diagram below shows the relationship between the output values and REQ. It is based on the assumption that the instruction is called cyclically to check the status of the send process (indicated by the STATUS values).

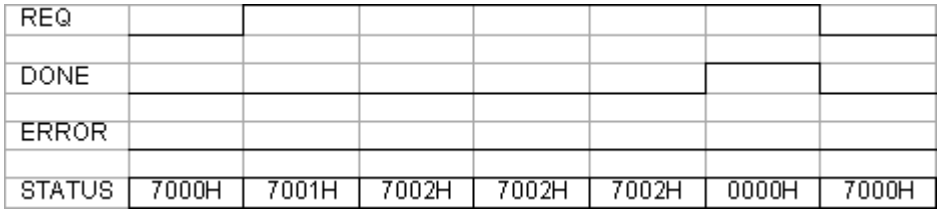

The figure below shows how the DONE and STATUS parameters are only valid for one cycle if a pulse is pending at the REQ line (for one cycle) to trigger the send instruction.

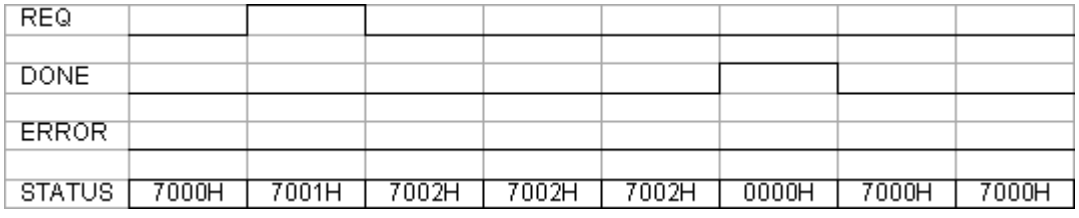

The figure below shows the relationship of the DONE, ERROR and STATUS parameters in case of an error.

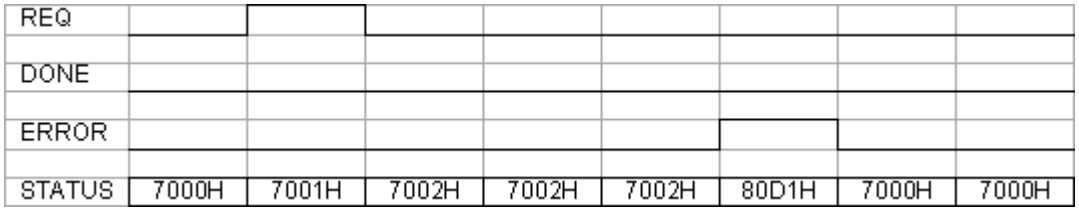

The DONE, ERROR and STATUS values are only valid until Send_P2P is executed again with the same instance DB.

Using the BUFFER and LENGTH parameters for communication operations (S7-1200, S7-1500)

Interaction of BUFFER and LENGTH parameters for Send_P2P

The minimum data size sent by the Send_P2P instruction is one byte.

The BUFFER parameter specifies the size of the data to be sent if a "0" is passed at the LENGTH parameter during call. The specification of a tag is sufficient for this.

You cannot use the data type Bool or arrays of the Bool type for the BUFFER parameter. If large amounts of data are being transferred we recommend the mapping to the array or structure data types.

Table 4-153 BUFFER parameter

BUFFER	Description
Elementary data type	When sending: The LENGTH value must include the byte size of this data type. Example: For a Word value, the LENGTH must be two. For a DWord value or Real value, the LENGTH must be four.
Structure	For optimized memory: The maximum permitted length of the BUFFER is 1024 Byte; otherwise 4 KB are permitted depending on the module. For sending: The LENGTH value can include a byte size smaller than the complete byte length of the structure; in this case, only the first LENGTH bytes of the structure from BUFFER are sent.
Array	For optimized memory: If the array data type is not equal to Byte, Word or DWord, the maximum permitted buffer length is 1024 Byte. Depending on the module, up to 4 KB can be transmitted if the memory is not optimized, independent of the data structure. For sending: The LENGTH value can include a byte size smaller than the complete byte length of the array, whereby, this byte size is a multiple of the byte size of the data element. Example: The LENGTH parameter of an array of the Word type must be a multiple of two and a multiple of four for an array of the Real type. If LENGTH is specified, the number of array elements is transmitted. If BUFFER includes an array with 15 DWord elements (a total of 60 Bytes), for example, and you specify LENGTH = 20, the first five DWord elements from the array are transmitted.
String	The LENGTH parameter includes the number or characters to be sent. Only the characters of the String are transmitted. The bytes with the maximum and actual length of the String are not sent.

Table 4-154 LENGTH parameter

LENGTH	Description
= 0	The complete content of the memory area specified by BUFFER is transferred. If BUFFER points to a string, the entire content of the string is transferred, without the bytes with the maximum and actual length.
> 0	The content up to the configured length of the memory area specified by BUFFER is transferred.

Receive_P2P: Receiving data (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Receive_P2P instruction (receive data using point-to-point communication) checks the frames received in the CM. If a frame is available, it is transmitted from the CM to the CPU. A receive error is indicated at the STATUS parameter.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
BUFFER	IN	Variant	Any	0	This parameter points to the start address of the receive buffer. This buffer must be large enough to receive the maximum frame length. Note: • Boolean data or Boolean fields are not supported. • If the receive buffer is in the optimized memory area, the maximum permitted length of the received data is 1024 bytes. Exception: Arrays of Byte, Word or DWord are supported up to a length of 4096 bytes. • If the receive buffer is a String or WString, the received data is written to the content of the string and the current length of the string is set accordingly. Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 3948)"
LENGTH	OUT	UInt	Word	0	Length of the frame received in bytes Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 3948)"
NDR	OUT	Bool		FALSE	TRUE for one cycle if new data is available and the instruction has been completed without errors.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

The error code at the STATUS output can be evaluated if the status of ERROR is TRUE. The STATUS value provides the reason for terminating the receive operation in the CM.
This is usually a positive value which indicates that the receive operation has been successful and the frame criterion that has been detected.
If the STATUS value is negative (the most significant bit of the hexadecimal value is set), the receive operation was terminated due to an error condition, such as a parity, framing or overflow error.

Each communication module can buffer a module-specific number for frames. If several frames are available in the CM, the Receive_P2P instruction outputs the oldest available frame (FIFO).

Receive_Reset: Clear receive buffer (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Receive_Reset instruction (reset receiver) clears the receive buffer in the CM.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle means that the last request was completed without errors.

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
ERROR	OUT	Bool		FALSE	TRUE means that the last request was completed with errors. If this output is TRUE, the STATUS output contains the corresponding error codes.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

Signal_Get: Read status (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Signal_Get instruction (get RS232 signals) reads the current states of the RS232 accompanying signals and displays them at the corresponding instruction outputs.

Note

Restriction

- This instruction can only be used with CMs RS232 BA and RS232 HF.
- If RS232C is set for the operating mode, this instruction can also be used with CM PtP (ET200SP).

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
NDR	OUT	Bool		FALSE	TRUE for one cycle if the RS232 accompanying signals have been read and the instruction has been completed without errors.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))
DTR	OUT	Bool		FALSE	Data device ready, module ready (output)
DSR	OUT	Bool		FALSE	Data device ready, communication station ready (input)
RTS	OUT	Bool		FALSE	Send request, module ready to send (output)
CTS	OUT	Bool		FALSE	Ready to send, communication station can receive data (input)
DCD	OUT	Bool		FALSE	Data carrier signal detected, signal level received
RING	OUT	Bool		FALSE	Call display, signaling incoming call

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

Signal_Set: Set accompanying signals (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Signal_Set instruction (set RS232 signals) allows you to set the RS232 communication signals.

Note

Restrictions

- This instruction can only be used with CMs RS232 BA and RS232 HF.
- If RS232C is set for the operating mode, this instruction can also be used with CM PtP (ET200SP).

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
SIGNAL	IN	Byte		0	Selection of the signal to be set (more than one possible): • 01H = RTS • 02H = DTR • 04H = DSR (for interface type DCE only)
RTS	IN	Bool		FALSE	Send request, module ready to send Set this value at the output (TRUE or FALSE), default value: FALSE
DTR	IN	Bool		FALSE	Data terminal ready, module ready Set this value at the output (TRUE or FALSE), default value: FALSE
DSR	IN	Bool		FALSE	Data terminal ready (for DCE interface type only), not used.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

Get_Features: Get extended functions (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

If supported by the module, you can use the Get_Features instruction (get extended functions) to get information on the ability of the module to support CRC and to generate diagnostic messages.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	PORT	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
NDR	OUT	Bool		FALSE	TRUE for one cycle if new data is available and the instruction has been completed without errors

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
MODBUS_CRC	OUT	Bool		FALSE	Modbus CRC support
DIAG_ALARM	OUT	Bool		FALSE	Generation of diagnostic messages
SUPPLY_VOLT	OUT	Bool		FALSE	Diagnostics for missing supply voltage L+ is available
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

Set_Features: Set extended functions (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

If supported by the module, you can use the Set_Features instruction (select extended functions) to activate CRC support and the generation of diagnostic messages.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	The instruction to set extended functions is started upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
EN_MODBUS_CRC	IN	Bool		FALSE	Activate Modbus CRC support
EN_DIAG_ALARM	IN	Bool		FALSE	Activate generation of diagnostic messages
EN_SUPPLY_VOLT	IN	Bool		FALSE	Enable diagnostics for missing supply voltage L+
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one execution once the last request is completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 3957))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 3933)".

Error messages (S7-1200, S7-1500)

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		

Error code	Description	Remedy
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#7002	Interim call: Data transmission running	-
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.

Error code	Description	Remedy
16#8193	The module does not support supply voltage diagnostics L+.	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire operation (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire operation. (4)

Error code	Description	Remedy
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see AUTOHOTSPOT).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)

Error code	Description	Remedy
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (bytes) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer on startup is deactivated (0) • Clear receive buffer on startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)

Error code	Description	Remedy
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end delimiters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
16#81DF	The module has reset the interface to the FB for one of the following reasons: • Module was restarted • Module parameters were reassigned • CPU STOP	—
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Break in receive line to partner. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (bytes) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		

Error code	Description	Remedy
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
16#81F2	The DSR signal cannot be set because the module has the type DTE.	Check the configured type of the module. The module type must be DCE (data communication equipment).
16#81F3	The DTR signal cannot be set because the module has the type DCE.	Check the configured type of the module. The module type must be DTE (data terminal equipment).
16#81F4	Block header error (e.g. incorrect block type or incorrect block length)	Check the instance DB and the block header.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditions points to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 KB • Non-optimized memory area: 4 KB Note: If the pointer points to an optimized memory area, do not send more than 1 KB.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in the RDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter before the 1st call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.

Error code	Description	Remedy
16#8301 [1]	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 [1]	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 [1]	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 [1]	Receive_Conditions does not point to a structure of the typeReceive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.

Error code	Description	Remedy
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the frame header and data string or the connection is terminated with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

1) Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.

Error code	Description	Remedy
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.

Error code	Description	Remedy
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 4009) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 4009) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 4009) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#80C9	The slave does not respond within the time set by Blocked_Proc_Timeout.	Check the setting for Blocked_Proc_Timeout. Check if the module has been configured with the Modbus_Comm_Load instruction. The module may possibly need to be reconfigured using Modbus_Comm_Load after a pull/plug or after voltage recovery.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.

Error code	Description	Remedy
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1)]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1)]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1)]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1)]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1)]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		
16#8428 [1)]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during call of SFC51	Call the Modbus_Slave instruction again
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200% to +200%
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request. Check if the instructions USS_Read_Param, USS_Write_Param or USS_Port_Scan have reported an error. If they have, check the value of the static tag USS_DB. w_USSExtendedError of the USS_Drive_Control instruction.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.

Error code	Description	Remedy
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

USS (S7-1200, S7-1500)

Overview of USS communication (S7-1200, S7-1500)

USS communication

The USS instructions control the operation of drives which support the protocol of the universal serial interface (USS). You can communicate with several drives by means of RS485 connections and the USS instructions with the PtP communication modules. Each RS485 port can operate up to 16 drives.

The USS protocol uses a master-slave network for communication via a serial bus. The master uses an address parameter to send data to a selected slave. A slave cannot send without having first received a send request. Communication between individual slaves is not possible. USS communication takes place in half-duplex mode. The figure below shows a network diagram for a sample application with 16 drives.

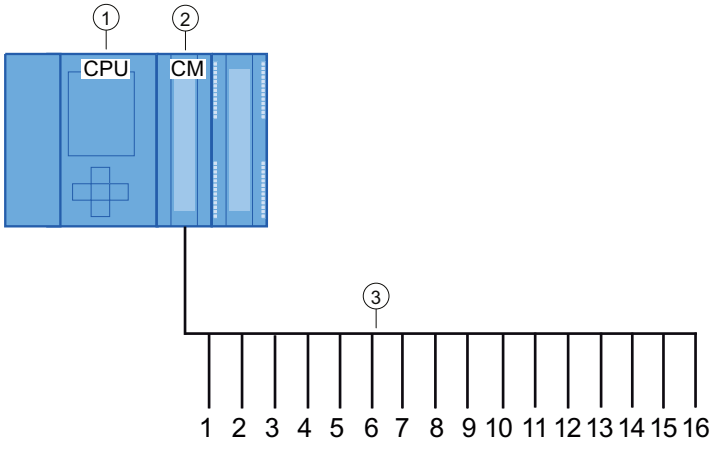

① CPU
② CM
③ USS drives in a USS network

Figure 4-4 Wiring example with S7-1500 communication module

Note

Communicating with a drive via RS232

You can basically also use CM PtP RS232 BA and CM PtP RS232 HF for communication with a drive. However, only **one** drive can be connected to an RS232 port.

Communicating with a drive via RS422

You can basically also use the RS422 interface of the CM PtP RS422/485 BA and CM PtP RS422/485 HF for communication with a drive. However, only **one** drive can be connected to an RS422 port.

USS instructions in your program

- USS_Port_Scan: The instruction USS_Port_Scan allows you to communicate via a communication module with up to 16 drives using a USS network (must be called cyclically). There is only one USS_Port_Scan instruction per PtP communication port in the program, and it controls transmission to all drives.
- USS_Drive_Control: The instruction USS_Drive_Control allows you to prepare the send data from USS_Port_Scan for a drive and display its receive data. USS_Drive_Control configures the data to be sent and evaluates the data received in a previous request from USS_Port_Scan.
- USS_Read_Param: The instruction USS_Read_Param allows you to read parameters from a drive.
- USS_Write_Param: The instruction USS_Write_Param allows you to change parameters in a drive.

Requirements for using the USS protocol (S7-1200, S7-1500)

The four USS instructions use 2 FBs and 2 FCs to support the USS protocol. For each USS network, one instance data block (DB) is used for USS_Port_Scan and one instance data block for all calls of USS_Drive_Conrol.

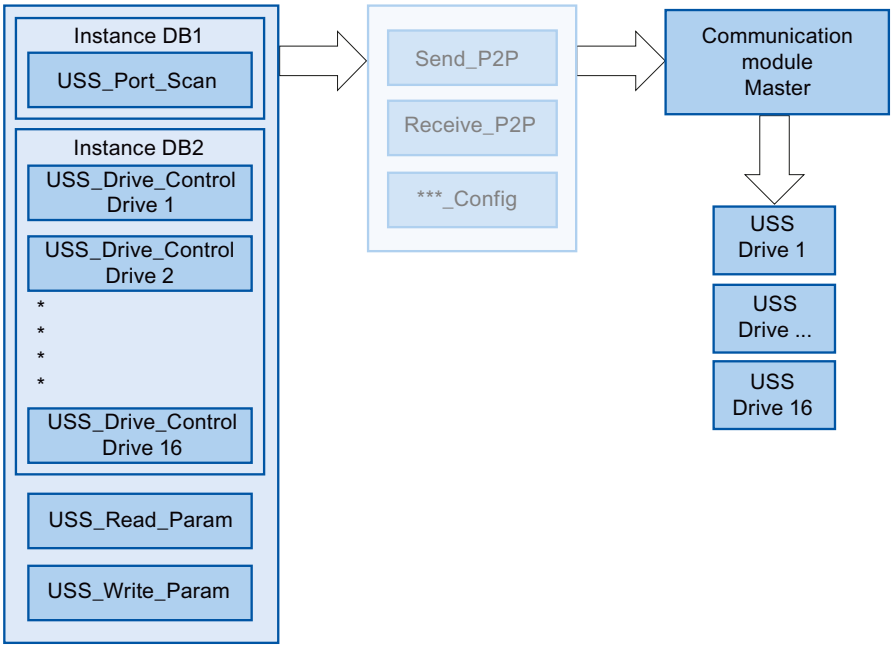

Figure 4-5 USS program sequence

All drives (max. 16) that are connected to one RS485 port are part of the same USS network. All drives that are connected to another RS485 port are part of a different USS network. Each USS network is managed using a unique instance data block for all USS_Drive_Control instructions and one further instance data block for the USS_Port_Scan instruction. All instructions that are part of a USS network must share this instance data block for USS_Drive_Conrol. The USS_Port_Scan, USS_Read_Param and USS_Write_Param instructions have the USS_DB parameter for this function. This parameter must be connected to the (static) USS_DB parameter of the instance DB of the USS_Drive_Control instruction.

- The instructions USS_Drive_Control and USS_Port_Scan are function blocks (FB). If you add the USS_Drive_Control or USS_Port_Scan instruction to the programming editor, you are prompted to assign a DB for this FB in the "Call options" dialog. If it is the first USS_Drive_Control instruction in this program for this USS network, you can apply the DB standard assignment (or change the name, if necessary) and the new DB is created for you. If, however, it is not the first USS_Drive_Control instruction for this drive, you need to select the DB which has already been assigned to this USS network from the drop-down list in the "Call options" dialog.
- The USS_Write_Param and USS_Read_Param instructions are functions (FCs). No DB is assigned when you add these FCs in the editor. If you add these FCs or the USS_Port_Scan instruction in the editor, you need to assign the USS_DB parameter of the corresponding USS_Drive_Control instance DB to the USS_DB input of these instructions. Double-click the parameter field and click the symbol to display the available DBs. Enter a period "." and select the USS_DB parameter from the drop-down list.

- The USS_Port_Scan function controls communication between the CPU and the drives via the point-to-point RS485 communication port. A communication with the drive is processed every time you call this function. Your program must call this function fast enough so that the drives do not signal a timeout. To ensure a constant time response for frame communication, you should call this instruction in a cyclic interrupt OB.
- The USS_Drive_Control instruction gives your program access to a specified drive in the USS network. Its inputs and outputs correspond to the states and operating functions of the drive. If there are 16 drives in your network, you must call USS_Drive_Control at least 16 times in your program, i.e. once for each drive.
  You should only call the USS_Drive_Control instruction from a cyclic OB.
- The operating parameters of the drive are read and written with the USS_Read_Param and USS_Write_Param functions. These parameters control the internal operation of the drive. See the drive manual for a definition of these parameters. Your program may include any number of these functions, but only one read or write request may be active for a drive at any particular time. You may only call the USS_Read_Param and USS_Write_Param functions from the cycle OB of a main program.

NOTICE
USS instruction calls Only ever call USS_Drive_Control, USS_Read_Param or USS_Write_Param from a cycle OB of the main program. The USS_Port_Scan instruction function can be called from any OB, but it is usually called from a cyclic interrupt OB. Do not use the USS_Drive_Control, USS_Read_Param or USS_Write_Param instruction in an OB with a higher priority than the corresponding USS_Port_Scan instruction. For example, do not add USS_Port_Scan to the main program or USS_Read_Param to a cyclic interrupt OB. You may encounter unexpected errors if the execution of USS_Port_Scan is interrupted by another instruction.

Note

Parameter ID value

You need to configure the use of 4 PIV words (ParameterIDValue) for the drives.

Calculating time for communication with the drive

Communication with the drive takes place asynchronously to the cycle of the CPU. The CPU usually runs through several cycles before communication with a drive is complete.

To ensure that the watchdog set for the drive is not triggered, the send frames must be sent to the drive within the watchdog time. You must allow for the number of any retries which may be needed to complete the transaction if communication errors occur. By default, up to 2 retries are made for each transaction with the USS protocol.
The maximum time interval between two send frames is calculated as follows:

N * (5 * cycle time + frame runtime + timeout of the receive frame max.) * (number of transmission attempts)

N	Number of drives in this network
Factor 5	5 cycles are typically needed for sending and receiving frames.
Cycle time	Max. cycle time of the cyclic interrupt OB in which the USS_Port_Scan instruction is called.
Frame runtime	Frame time = (number of characters per frame) * (11 Bit per character) / (data transmission rate in Bit/s)
Number of transmission attempts	Number of retries + 1
Timeout of the receive frame	RCVTIME (if no response is received from the drive)
Timeout of the receive frame max.	RCVTIME + MSGTIME (if an incomplete answer is received shortly before the expiration of RCVTIME and monitoring of MSGTIME has expired **or** if a response is still being processed at the expiration of RCVTIME, the timeout is extended by the MSGTIME)

The following times apply for "Received frame timeout" (ms):

Bit/s	115200	57600	38400	19200	9600	4800	2400	1200
Receive_Conditions.END.RCVTIME	25	29	33	56	72	100	100	100
Receive_Conditions.END.MSGTIME	25	29	33	56	72	124	240	460

Timeout of the receive frame max. = (Receive_Conditions.END.RCVTIME (0.072 s) + Receive_Conditions.END.MSGTIME (0.072 s))

Example:

5 drives
Data transmission rate = 9600 bps
28 characters per frame
Cycle time = 0.020 s
Number of retries = 2

Time interval = 5 * (5 * 0.02 s + 1 * 28 * 11 / 9600 s + 0.072 s + 0.072 s) * 3 = 3 s

The time monitoring of the drive must be set to about 3 seconds in this case.

USS_Port_Scan / USS_Port_Scan_31: Communication by means of a USS network (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Port_Scan_31 instruction

It is only possible to use the USS_Port_Scan_31 instruction on an S7-1500 CPU.

Description

The USS_Port_Scan instruction processes the communication by means of a USS network for max. 16 drives.

The USS_Port_Scan_31 instruction processes the communication by means of a USS network for max. 31 drives.

STEP 7 automatically creates the instance DB when you add the instruction.

Parameters

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
PORT	IN	Port	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
BAUD	IN	DInt		9600	Data transmission rate for USS communication The following are valid: • 1200 bps • 2400 bps • 4800 bps • 9600 bps • 19200 bps • 38400 bps • 57600 bps • 115200 bps
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control / USS_Drive_Control_31 instruction to your program.

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
COM_RST	INOUT	---	Bool	FALSE	Initialization of the USS_Port_Scan / USS_Port_Scan_31 instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.
ERROR	OUT	Bool		FALSE	If TRUE, this output indicates that an error has occurred and that the STATUS output is valid. You may need to check the value of the static tag USS_DB. w_USSExtendedError in the instance DB of the USS_Drive_Control / USS_Drive_Control_31 instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 3990)).

There is only one USS_Port_Scan / USS_Port_Scan_31 instruction per PtP communication port in the program and each call of this instruction controls a transmission to or from all drives in this network. All USS functions assigned to one USS network and one PtP communication port must use the same instance DB.

Your program must execute the USS_Port_Scan / USS_Port_Scan_31 instruction often enough to prevent a timeout in the drive (see Requirements for using the USS protocol (Page 3973) "Calculating time for communication with the drive").

The USS_Port_Scan / USS_Port_Scan_31 instruction is usually called from a cyclic interrupt OB to prevent drive timeouts and to make the last USS data updates available for calls of USS_Drive_Control / USS_Drive_Control_31 .

USS_Port_Scan / USS_Port_Scan_31 data block tags

The table below shows the public static tags in the instance DB of USS_Port_Scan / USS_Port_Scan_31 that you can use in your program.

Table 4-155 Static tags in the instance DB

Tag	Data type	Standard	Description
MODE	USInt	4	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master; CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave; CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode [1]
LINE_PRE	USInt	2	Receive line initial state Valid initial states are: • 0 = "No" initial state [1] • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire mode (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
BRK_DET	USInt	0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
RETRIES_MAX	SInt/Byte	2	Number of retries when communication errors occur. You can use this parameter to set the number of send retries for a request frame if the response frame is not received within the set time.
EN_DI-AG_ALARM	Bool	0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
EN_SUP-PLY_VOLT	Bool	0	Enable diagnostics for missing supply voltage L+ • 0 - not activated • 1 - activated

[1] Required setting for the use of PROFIBUS cables with CM 1241 for RS485

Version 2.5 is functionally identical to version 2.4 and its version number was only incremented due to internal measures.

Instruction versions

USS_Port_Scan:

Version 2.5 is functionally identical to version 2.4 and its version number was only incremented due to internal measures.

USS_Port_Scan_31:

Version 1.2 is functionally identical to version 1.1 and its version number was only incremented due to internal measures.

USS_Drive_Control / USS_Drive_Control_31: Preparing and displaying data for the drive (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Drive_Control_31 instruction

It is only possible to use the USS_Drive_Control_31 instruction on an S7-1500 CPU.

Description

The USS_Drive_Control instruction prepares send data for max. 16 drives and evaluates the response data of the drives.

The USS_Drive_Control_31 instruction prepares send data for max. 31 drives and evaluates the response data of the drives.

You need to use a separate instance of the instruction for each drive, and all USS functions assigned to one USS network and one PtP communication port must use the same instance DB. You must enter the DB name when you add the first USS_Drive_Control / USS_Drive_Control_31 instruction. You then refer to this DB that was created when the first instruction was added.

STEP 7 automatically creates the DB when you add the instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/ WinAC		
RUN	IN	Bool		FALSE	Start bit of the drive: If this parameter is TRUE, this input allows you to operate the drive with the preset speed. If RUN changes to FALSE during operation of the drive, the motor coasts to a standstill. This behavior differs from disconnection of the power supply (OFF2) and braking of the motor (OFF3).
OFF2	IN	Bool		FALSE	"Coast to standstill" bit: If this parameter is FALSE, this bit causes the drive to coast to a standstill without braking.
OFF3	IN	Bool		FALSE	Fast stop bit: If this parameter is FALSE, this bit causes a fast stop by braking the drive.
F_ACK	IN	Bool		FALSE	Error acknowledgment bit: This bit resets the error bit of a drive. The bit is set after clearing the error and the drive detects this way that the previous error no longer has to be reported.
DIR	IN	Bool		FALSE	Direction control of the drive: This bit is set if the drive is to run forward (if SPEED_SP is positive; see table "Interaction of SPEED_SP and DIR parameters").
DRIVE	IN	USInt	Byte	1	Address of the drive: This input is the address of the USS drive. The valid range is between drive 1 and drive 16.
PZD_LEN	IN	USInt	Byte	2	Word length: This is the number of PZD data words. Valid values are 2, 4, 6 or 8 words.
SPEED_SP	IN	Real		0.0	Speed setpoint: This is the speed of the drive as a percentage of the configured frequency. A positive value means that the drive runs forward (if DIR is true). The valid range is 200.00 to -200.00.
CTRL3	IN	Word		0	Control word 3: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL4	IN	Word		0	Control word 4: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL5	IN	Word		0	Control word 5: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL6	IN	Word		0	Control word 6: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL7	IN	Word		0	Control word 7: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL8	IN	Word		0	Control word 8: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/ WinAC		
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the USS_Drive_Control / USS_Drive_Control_31 instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.
NDR	OUT	Bool		FALSE	New data available: If this parameter is TRUE, the bit signals that data of a new communication request is available at the output.
ERROR	OUT	Bool		FALSE	Error occurred: If TRUE, this indicates that an error has occurred and that the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan / USS_Port_Scan_31 instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 3990)).
RUN_EN	OUT	Bool		FALSE	Operation enabled: This bit signals if the drive is running.
D_DIR	OUT	Bool		FALSE	Drive direction: This bit signals if the drive is running forward. • FALSE – forward • TRUE – backward
INHIBIT	OUT	Bool		FALSE	Drive blocked: This bit signals the status of the block bit for the drive. • FALSE – not blocked • TRUE – blocked
FAULT	OUT	Bool		FALSE	Drive error: This bit signals that an error occurred in the drive. You must remedy the fault and set the F_ACK bit to clear this bit.
SPEED	OUT	Real		0.0	Actual value drive speed (scaled value of STATUS 2 of the drive): This is the speed of the drive as a percentage of the configured speed.
STATUS1	OUT	Word		0	STATUS 1 of the drive This value includes fixed status bits of a drive.
STATUS3	OUT	Word		0	STATUS 3 of the drive This value includes a user-definable status word of the drive.
STATUS4	OUT	Word		0	STATUS 4 of the drive This value includes a user-definable status word of the drive.
STATUS5	OUT	Word		0	STATUS 5 of the drive This value includes a user-definable status word of the drive.
STATUS6	OUT	Word		0	STATUS 6 of the drive This value includes a user-definable status word of the drive.
STATUS7	OUT	Word		0	STATUS 7 of the drive This value includes a user-definable status word of the drive.
STATUS8	OUT	Word		0	STATUS 8 of the drive This value includes a user-definable status word of the drive.

When the initial execution of USS_Drive_Control / USS_Drive_Control_31 takes place, the drive specified by the USS address (DRIVE parameter) is initialized in the instance DB. After initialization, subsequent USS_Port_Scan / USS_Port_Scan_31 instructions can start communication with the drive from this drive number.

If you change the drive number, you must first place the CPU in STOP and then back in RUN to initialize the instance DB. The input parameters are configured in the USS send buffer and any outputs are read from a "previous" valid response buffer. USS_Drive_Control / USS_Drive_Control_31 only configures the data to be sent and evaluates data received in a previous request.

You can control the drive's direction of rotation by using the D_IR input (Bool) or the sign (positive or negative) at the SPEED_SP input (Real). The table below explains how these inputs work together to determine the direction of rotation of the drive, provided the motor rotates forward.

Table 4-156 Interaction of SPEED_SP and DIR parameters

SPEED_SP	DIR	Direction of rotation of the drive
Value > 0	0	Backward
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Backward

USS_Drive_Control / USS_Drive_Control_31 data block tags

The table below shows the public static tags in the instance DB of USS_Drive_Control / USS_Drive_Control_31 that you can use in your program.

Table 4-157 Static tags in the instance DB

Tag	Data type	Standard	Description
USS_DB. w_USSExtendedError	Word	16#0	USS Drive Extended Error Code - drive-specific value The meaning of the error message depends on which instruction reported an error **first** (ERROR = TRUE). The following cases are distinguished: • USS_Write_Param / USS_Write_Param_31: You can find the meaning of the error code in the description of the drive. • USS_Read_Param / USS_Read_Param_31: You can find the meaning of the error code in the description of the drive. • USS_Port_Scan / USS_Port_Scan_31: Number of the drive affected by the error message.

Instruction versions

USS_Drive_Control:

Version 2.0 is functionally identical to version 1.2 and its version number was only incremented due to internal measures.

USS_Drive_Control_31:

Version 2.0 is functionally identical to version 1.0 and its version number was only incremented due to internal measures.

USS_Read_Param / USS_Read_Param_31: Read data from drive (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Read_Param_31 instruction

It is only possible to use the USS_Read_Param_31 instruction on an S7-1500 CPU.

Description

The USS_Read_Param instruction reads a parameter from one of max. 16 drives.

The USS_Read_Param_31 instruction reads a parameter from one of max. 31 drives.

All USS functions assigned to one USS network and one PtP communication port must use the instance data block of the USS_Drive_Control / USS_Drive_Control_31 instruction. USS_Read_Param / USS_Read_Param_31 must be called from a cycle OB of the main program.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		–	A positive edge at REQ creates a new read request.
DRIVE	IN	USInt	Byte	–	Address of the drive: DRIVE is the address of the USS drive. The valid range is between drive 1 and drive 16.
PARAM	IN	UInt		–	Parameter number: PARAM specifies the drive parameter to write. The range for this parameter is between 0 and 2047. With some drives, the most significant byte of the INDEX parameter can be used to access parameter values greater than 2047. Additional information on access to an extended range is available in your drive manual.
INDEX	IN	UInt		–	Parameter index: INDEX specifies the drive parameter index to which to write. It is a 16-bit value in which the least significant bit is the actual index value with a range of (0 to 255). The drive can also use the most significant byte, which is drive-specific. Additional information is available in your drive manual.

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control / USS_Drive_Control_31 instruction to your program.
DONE[1]	OUT	Bool		FALSE	If this parameter is TRUE, the previously requested value of the read parameter is available at the VALUE output. This bit is set when the USS_Drive_Control / USS_Drive_Control_31 instruction recognizes the read response of the drive. This bit is reset the next time USS_Read_Param / USS_Read_Param_31 is called.
ERROR	OUT	Bool		FALSE	ERROR = TRUE: An error has occurred and the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan / USS_Port_Scan_31 instruction. You may need to check the value of the static tag USS_DB. w_USSExtendedError in the instance DB of the USS_Drive_Control / USS_Drive_Control_31 instruction.
STATUS	OUT	UInt		0	Error code (see Error messages (Page 3990)).
VALUE	OUT	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, UInt, DWord, DInt, UDInt, Real)	–	This is the value of the parameter that was read and is only valid if the DONE bit is true.

[1] The DONE bit indicates that valid data was read out of the referenced motor drive and transmitted to the CPU. It does not indicate that the instruction is cabable of immediately read an additional parameter. An empty read request must be sent to the motor drive and acknowledged by the instruction before the parameter channel is freed up for use by the respective drive. The immediate call of USS_Read_Param / USS_Read_Param_31 or USS_Write_Param / USS_Write_Param_31 for the specific motor drive results in error 16#818A.

Instruction versions

USS_Read_Param:

Version 1.5 is functionally identical to version 1.4 and its version number was only incremented due to internal measures.

USS_Read_Param_31:

Version 1.1 is functionally identical to version 1.0 and its version number was only incremented due to internal measures.

USS_Write_Param / USS_Write_Param_31: Change data in drive (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Write_Param_31 instruction

It is only possible to use the USS_Write_Param_31 instruction on an S7-1500 CPU.

Note

For EEPROM write instructions (EEPROM in a USS drive):

Keep the number of EEPROM write operations to a minimum in order to maximize the EEPROM service life.

Description

The USS_Write_Param instruction changes a parameter in one of max. 16 drives.

The USS_Write_Param_31 instruction changes a parameter in one of max. 31 drives.

All USS functions assigned to one USS network and one PtP communication port must use the instance data block of the USS_Drive_Control / USS_Drive_Control_31.

USS_Write_Param / USS_Write_Param_31 must be called from the cycle OB of a main program.

Parameters

Table 4-158 Data types for the parameters

Parame-ters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/ WinAC		
REQ	IN	Bool		–	A positive edge at REQ creates a new write request.
DRIVE	IN	USInt	Byte	–	Address of the drive: DRIVE is the address of the USS drive. The valid range is between drive 1 and drive 16.
PARAM	IN	UInt		–	Parameter number: PARAM specifies the drive parameter to write. The range for this parameter is between 0 and 2047. With some drives, the most significant byte of the INDEX parameter can be used to access parameter values greater than 2047. Additional information on access to an extended range is available in your drive manual.

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/ WinAC		
INDEX	IN	UInt		–	Parameter index: INDEX specifies the drive parameter index to which to write. It is a 16-bit value in which the least significant bit is the actual index value with a range of (0 to 255). The drive can also use the most significant byte, which is drive-specific. Additional information is available in your drive manual.
EEPROM	IN	Bool		–	Save in EEPROM of the drive: If TRUE, the transaction of a parameter for writing to the drive is saved in the EEPROM of the drive. If FALSE, the written value is saved only temporarily and is lost the next time you switch on the drive.
VALUE	IN	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, UInt, DWord, DInt, UDInt, Real)	–	Value of the parameter in which you want to write. It must be valid with a positive edge of REQ.
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control / USS_Drive_Control_31 instruction to your program.
DONE[1]	OUT	Bool		FALSE	If TRUE, the VALUE input is written to the drive. This bit is set when the USS_Drive_Control / USS_Drive_Control_31 instruction recognizes the write response of the drive. This bit is reset the next time USS_Write_Param / USS_Write_Param_31 is called.
ERROR	OUT	Bool		FALSE	If TRUE, an error has occurred and the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan / USS_Port_Scan_31 instruction. You may need to check the value of the static tag USS_DB.w_USSExtendedError in the instance DB of the USS_Drive_Control / USS_Drive_Control_31 instruction.
STATUS	OUT	UInt		0	Error code (see Error messages (Page 3990)).

[1] The DONE bit indicates that valid data was read out of the referenced motor drive and transmitted to the CPU. It does not indicate that the USS library is able to read out an additional parameter immediately. An empty write request must be sent to the motor drive and acknowledged by the instruction before the parameter channel is freed up for use by the respective drive. The immediate call of USS_Read_Param / USS_Read_Param_31 or USS_Write_Param / USS_Write_Param_31 FC for the specific motor drive results in error 0x818A.

Instruction versions

USS_Write_Param:

Version 1.6 is functionally identical to version 1.5 and its version number was only incremented due to internal measures.

USS_Write_Param_31:

Version 1.1 is functionally identical to version 1.0 and its version number was only incremented due to internal measures.

General information on drive setup (S7-1200, S7-1500)

Requirements for the drive setup

- You need to configure the use of 4 PIV words (ParameterIDValue) for the drives.
- The drives can be configured for 2, 4, 6 or 8 PZD words (Process data area).
- The number of PZD words in the drive must correspond to the PZD_LEN input of the USS_Drive_Control instruction of the drive.
- Make sure that the data transmission rate of all drives corresponds to the BAUD input of the USS_Port_Scan instruction.
- Make sure that the drive is set up for USS communication.
- Make sure that it is specified in the drive that the frequency setpoint is provided by the USS interface.
- Make sure that the drive address is specified (areas: 1-16).
  This address must correspond to the DRIVE input at the USS_Drive_Control block of the drive.
- Make sure that the RS485 network is terminated correctly.

Connection and setup of SINAMICS V20 drive

An application example for operation of a SINAMICS V20 at an S7-1200 is available on the Internet. (http://support.automation.siemens.com/WW/view/en/63696870)

Connecting the SINAMICS V20 drive

Connection example of a SIEMENS G120(C) drive to a USS network. Connection examples for other drives are available in the manual of the respective drive.

The connection of a SINAMICS G120(C) drive to the USS network takes place via plug-in connection. The connection is short-circuit proof and isolated.

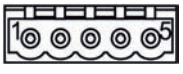

1 0 V reference potential
2 RS485N, receiving and sending (-)
3 RS485N, receiving and sending (+)
4 Cable shield
5 Not used

Figure 4-6 USS connection

NOTICE
Different reference voltages If you connect devices that do not have the same reference voltage, you may create unwanted currents in the connection cable. These unwanted currents may cause communication errors or damages in the devices. Make sure that all devices you connect with a communication cable either have the same reference conductor in the circuit or are electrically disconnected to prevent the occurrence of unwanted currents. Make sure that the shield is connected to the ground or pin 1 of the bus connector on the drive. Make sure that wiring terminal 2 (GND) of the G120(C) is connected to the ground.

If the RS485 master (e.g., S7-1200 CPU with CM1241 communication module) is connected by means of a PROFIBUS connector, wire the bus cables as follows:

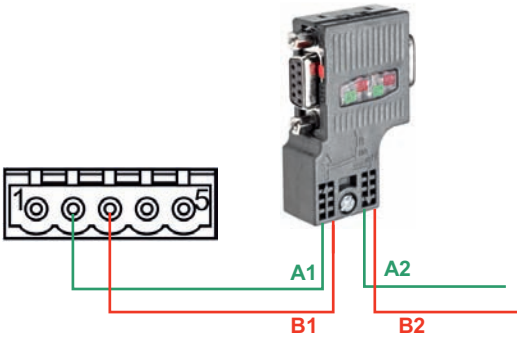

Figure 4-7 Connection of communication module

If the RS485 master is a terminating station in the network or a point-to-point connection, you must use terminals A1 and B1 (not A2 and B2) of the PROFIBUS connector, as these terminals provide termination settings (for example, with the DP plug connector 6ES7972-0BB52-0XA0).

If the G120(C) is configured as terminating station in the network, you must set the switch for the bus terminating resistor to "ON".

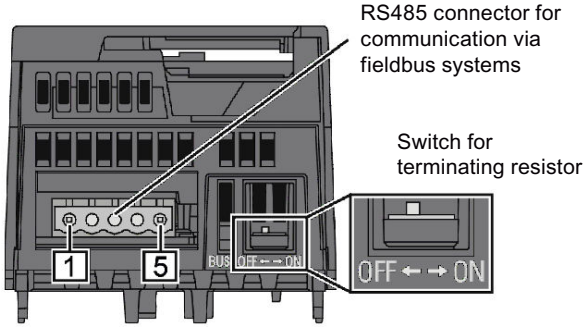

Figure 4-8 Connection of terminating stations

Setup of a G120(C) drive

Before you connect a drive to the S7-1500 or ET 200SP, make sure that the drive has the following system parameters.

Step	Instruction	Operating Instructions	
		G120 [1]	G120C [2]
1	Perform a basic commissioning of the drive with the Operator Panel BOP-2. The converter offers different defaults (macros) for its inputs and outputs and the fieldbus interface. In the ninth step of the basic commissioning (MAC PAR p15), select macro 21 for USS communication. This determines the default for the following parameters: • Data transmission rate (p2020): 38400 bps • Number of PZD (p2022): 2 • Number of PIV (p2023): variable Note: You can also conduct basic commissioning with the STARTER commissioning software or with SINAMICS Startdrive.	Chapter 4.4.3	Chapter 6.4.1
2	Specify the USS address of the converter with the address switch on the control unit of the G120 or on the G120(C). • Valid address range: 1 … 30 Note: Yon can also set the USS address with the parameter p2021 or with STARTER or SINAMICS Startdrive.	Chapter 6.2.2.1	Chapter 8.4.2.1
	With the following steps you are accessing the parameters directly with the BOP-2 by entering their numbers and modifying their values.	Chapter 4.4.2	Chapter 6.4.2
3	Adapt the following communication-related converter parameters of your application: • Data transmission rate (p2020), of ≠ 38400 bps (Make sure that setting is identical with the BAUD parameter of the USS_Port_Scan communication instruction.) • Number of PZD (p2022), if ≠ 2 (Make sure that the setting is identical with the PZD_LEN parameter of the USS_Drive_Control communication instruction.) • Number of PIV (p2023) = 4 (Change the value set to "variable" (127) with macro 21 by default to "4" (required by instructions USS_Read_Param and USS_Write_Param.) • Fieldbus SS monitoring time [ms] (p2040)	Chapter 6.2.2.2	Chapter 8.4.2.1
4	Specify the source for the speed setpoint. • n_set Eval (p1000[0]) = 6 (The speed setpoint is provided by the USS bus.)		

Step	Instruction	Operating Instructions	
		G120 [1]	G120C [2]
5	Set the reference value for speed and frequency. • n_reference f_reference (p2000) = (6,00 min^{-1} to 210000,00 min^{-1}) (all relative speeds or frequencies refer to this reference value. The reference value corresponds to 100% or 4000_{hex} (word) or $4000\ 0000_{hex}$ (double word). The following applies: $f_{reference_value}$ (in Hz) = $n_{reference_value}$ (in ((min^{-1}) / 60) x number of pole pairs))		
6	Transfer the parameters to the non-volatile memory. • Save par (p0971) = 1		

1) G120 (http://support.automation.siemens.com/WW/view/en/62089662)

2) G120(C) (http://support.automation.siemens.com/WW/view/en/61462568)

Error messages (S7-1200, S7-1500)

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)

Error code	Description	Remedy
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#7002	Interim call: Data transmission running	-
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
16#8193	The module does not support supply voltage diagnostics L+.	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)

Error code	Description	Remedy
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire operation (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire operation. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		

Error code	Description	Remedy
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see AUTOHOTSPOT).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (bytes) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)

Error code	Description	Remedy
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer on startup is deactivated (0) • Clear receive buffer on startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end delimiters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
16#81DF	The module has reset the interface to the FB for one of the following reasons: • Module was restarted • Module parameters were reassigned • CPU STOP	—
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.

Error code	Description	Remedy
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Break in receive line to partner. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (bytes) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
16#81F2	The DSR signal cannot be set because the module has the type DTE.	Check the configured type of the module. The module type must be DCE (data communication equipment).
16#81F3	The DTR signal cannot be set because the module has the type DCE.	Check the configured type of the module. The module type must be DTE (data terminal equipment).
16#81F4	Block header error (e.g. incorrect block type or incorrect block length)	Check the instance DB and the block header.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING

Error code	Description	Remedy
16#8225	Receive_Conditions points to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 KB • Non-optimized memory area: 4 KB Note: If the pointer points to an optimized memory area, do not send more than 1 KB.
16#8229 [1]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in the RDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter before the 1st call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 [1]	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 [1]	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 [1]	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 [1]	Receive_Conditions does not point to a structure of the typeReceive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		

Error code	Description	Remedy
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the frame header and data string or the connection is terminated with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.

Error code	Description	Remedy
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.

Error code	Description	Remedy
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.

Error code	Description	Remedy
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 4009) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 4009) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 4009) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).

Error code	Description	Remedy
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#80C9	The slave does not respond within the time set by Blocked_Proc_Timeout.	Check the setting for Blocked_Proc_Timeout. Check if the module has been configured with the Modbus_Comm_Load instruction. The module may possibly need to be reconfigured using Modbus_Comm_Load after a pull/plug or after voltage recovery.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.

Error code	Description	Remedy
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200% to +200%
16#8187	Incorrect drive number returned	Check the response frame of the drive.

Error code	Description	Remedy
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request. Check if the instructions USS_Read_Param, USS_Write_Param or USS_Port_Scan have reported an error. If they have, check the value of the static tag USS_DB. w_USSExtendedError of the USS_Drive_Control instruction.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

MODBUS (RTU) (S7-1200, S7-1500)

Overview of the Modbus RTU communication (S7-1200, S7-1500)

Modbus RTU communication

Modbus RTU (Remote Terminal Unit) is a standard protocol for communication in the network and uses the RS232 or RS422/485 connection for serial data transmission between Modbus devices in the network.

Modbus RTU uses a master/slave network in which all communication is triggered by a single master device while the slaves can only respond to the request of the master. The master sends a request to a slave address and only the slave with this slave address responds to the command.

Exception: Modbus slave address 0 sends a broadcast frame to all slaves (without slave response).

Modbus function codes

- A CPU that is operated as a Modbus RTU master can read and write data and I/O states in a Modbus RTU slave connected by means of a communication connection.
- A CPU operated as a Modbus RTU slave allows a Modbus RTU master connected over a communication connection to read and write data and I/O states in its own CPU.

Table 4-159 Functions for reading data: Reading distributed I/O and program data

Modbus function code	Functions for reading data from the slave (server) - standard addressing
01	Read output bits: 1 to 2000/1992[1)] bits per request
02	Read input bits: 1 to 2000/1992[1)] bits per request
03	Read hold register: 1 to 125/124[1)] words per request
04	Read input words: 1 to 125/124[1)] words per request

1) for extended addressing

Table 4-160 Functions for writing data: Changing distributed I/O and program data

Modbus function code	Functions for writing of data in the slave (server) - standard addressing
05	Write one output bit: 1 bit per request
06	Write one hold register: 1 word per request
15	Write one or several output bits: 1 to 1960 bits per request
16	Write one or several hold registers: 1 to 122 words per request

- The Modbus function codes 08 and 11 offer diagnostic options for communication with the slave device.
- Modbus slave address 0 sends a broadcast frame to all slaves (without slave response; for function codes 5, 6, 15, 16).

Table 4-161 Station addresses in the Modbus network

Station		Address
RTU station	Standard station address	1 to 247 and 0 for broadcast
	Extended station address	1 to 65535 and 0 for broadcast

Modbus memory addresses

The number of Modbus memory addresses (input/output addresses) that is actually available depends on the CPU version and the available work memory.

Modbus RTU instructions in your program

- Modbus_Comm_Load: You need to run Modbus_Comm_Load to set up PtP parameters such as data transmission rate, parity and data flow control. Once you have configured the communication module for the Modbus RTU protocol, it can only be used by the Modbus_Master instruction or the Modbus_Slave instruction.
- Modbus_Master: The CPU can be used as Modbus RTU master device with the Modbus master instruction for communication with one or more Modbus slave devices.
- Modbus_Slave: The CPU can be used as Modbus RTU slave device with the Modbus slave instruction for communication with one Modbus master device.

Modbus_Comm_Load: Configure communication module for Modbus (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Modbus_Comm_Load instruction configures a communication module for communication by means of the Modbus RTU protocol. An instance data block is automatically assigned when you add the Modbus_Comm_Load instruction in your program.

Configuration changes of Modbus_Comm_Load are saved on the CM and not in the CPU. With voltage recovery and pulling/plugging, the CM is configured with the data saved in the device configuration. The Modbus_Comm_Load instruction must be called in these scenarios.

Parameters

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	Port	Laddr	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
BAUD	IN	UDInt	DWord	9600	Selection of the data transmission rate Valid values are: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 bit/s.
PARITY	IN	UInt	Word	0	Selection of parity: • 0 – None • 1 – Odd • 2 – Even
FLOW_CTRL	IN	UInt	Word	0	Selection of flow control: • 0 – (default) no flow control • 1 – Hardware flow control with RTS always ON (not with RS422/485 CMs) • 2 – Hardware flow control with RTS switched (not with RS422/485 CMs)
RTS_ON_DLY	IN	UInt	Word	0	Selection RTS ON delay: • 0 – No delay from "RTS active" until the first character of the frame is sent. • 1 to 65535 – Delay in milliseconds from "RTS active" until the first character of the frame is sent (not with RS422/485 CMs). RTS delays must be used independent of the selection FLOW_CTRL.
RTS_OFF_DLY	IN	UInt	Word	0	Selection RTS OFF delay: • 0 – No delay after transmission of last character until "RTS inactive" • 1 to 65535 – Delay in milliseconds after transmission of last character until "RTS inactive" (not with RS422/485 ports). RTS delays must be used independent of the selection FLOW_CTRL.
RESP_TO	IN	UInt	Word	1000	Response timeout: 5 ms to 65535 ms - Time in milliseconds that Modbus_Master waits for a response from the slave. If the slave does not respond within this period, Modbus_Master repeats the request or terminates the request with an error if the specified number of repetitions (see below, RETRIES parameter) has been sent.
MB_DB	IN/OUT	MB_BASE		-	A reference to the instance data block of the Modbus_Master or Modbus_Slave instructions. The MB_DB parameter must be connected with the (static and therefore not visible in the instruction) MB_DB parameter of the Modbus_Master or Modbus_Slave instruction.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Comm_Load instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
DONE	OUT	Bool		FALSE	The DONE bit is TRUE for one cycle after the last request has been completed without errors.
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. The error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 4029))

Modbus_Comm_Load is executed to configure a port for the Modbus RTU protocol. Once you have configured the port for the Modbus RTU protocol, it can only be used by the Modbus_Master or Modbus_Slave instructions.

You have to run Modbus_Comm_Load for the configuration of each communication port that is to be used for Modbus communication. You must assign a unique Modbus_Comm_Load instance DB to each port that you use. Only run Modbus_Comm_Load again if you need to change communication parameters, such as data transmission rate or parity, or in case the network has returned.

For example, an instance data block is assigned to the instruction if you add Modbus_Master or Modbus_Slave to your program. You need to connect the MB_DB parameter of the Modbus_Comm_Load instruction to the MB_DB parameter of the Modbus_Master or Modbus_Slave instruction.

Modbus_Comm_Load data block tags

The table below shows the public static tags in the instance DB of Modbus_Comm_Load that you can use in your program.

Table 4-162 Static tags in the instance DB

Tag	Data type		Standard	Description
	S7-1200/1500	S7-300/400/WinAC		
ICHAR_GAP	Word		0	Maximum character delay time between characters. This parameter is specified in milliseconds and increases the anticipated period between the received characters. The corresponding number of bit times for this parameter is added to the Modbus default value of 35 bit times (3.5 character times).
RETRIES	Word		2	Number of retries that the master executes before the error code 0x80C8 for "No response" is returned.
EN_SUPPLY_VOLT	Bool		0	Enable diagnostics for missing supply voltage L+

Tag	Data type		Stand-ard	Description
	S7-1200/1500	S7-300/400/WinAC		
MODE	USInt	Byte	0	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master, CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave, CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode [1]
LINE_PRE	USInt	Byte	0	Receive line initial state Valid initial states are: • 0 = "No" initial state [1] • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire mode (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
BRK_DET	USInt	Byte	0	Break detection The following are valid: • 0 = break detection deactivated • 1 = break detection activated
EN_DI-AG_ALARM	Bool		0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
STOP_BITS	USINT	Byte	1	Number of stop bits; • 1 = 1 stop bit, • 2 = 2 stop bits, • 0, 3 to 255 = reserved

[1] Required setting for the use of PROFIBUS cables with CM 1241 for RS485

Instruction versions

Version 3.1 is functionally identical to version 3.0 and its version number was only incremented due to internal measures.

Modbus_Master: Communicate as Modbus master (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Modbus_Master instruction communicates as Modbus master via a port configured by the Modbus_Comm_Load instruction. An instance data block is automatically assigned when you add the Modbus_Master instruction in your program. The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the (static) MB_DB parameter of the Modbus_Master instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	FALSE = no request TRUE = request to send data to the Modbus slave
MB_ADDR	IN	UInt	Word	-	Modbus RTU station address: Standard addressing range (1 to 247 as well as 0 for Broadcast) Extended addressing range (1 to 65535 as well as 0 for Broadcast) The value 0 is reserved for the broadcast of a frame to all Modbus slaves. Only the Modbus function codes 05, 06, 15 and 16 are supported for the broadcast.
MODE	IN	USInt	Byte	0	Mode selection: Specifies the type of request (read, write or diagnostics). Additional information is available in the table of Modbus functions below.
DATA_ADDR	IN	UDInt	DWord	0	Start address in the slave: Specifies the start address of the data that is accessed in the Modbus slave. The valid addresses are listed in the table of Modbus functions below.
DATA_LEN	IN	UInt	Word	0	Data length: Specifies the number of bits or words this instruction is to access. The valid lengths are listed in the table of Modbus functions below.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Master instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
DATA_PTR	IN/OUT	Variant	Any	-	Data pointer: Points to the flag or DB address for the data to be written or read. As of instruction version V3.0: The parameter may point to an optimized memory area. In the optimized memory area, a single element or an array is permitted with the following data types: Bool, Byte, Char, Word, Int, DWord, DInt, Real, USInt, UInt, UDInt, SInt, WChar. Every other data type results in error message 16#818C.
DONE	OUT	Bool		FALSE	The DONE bit is TRUE for one cycle after the last request has been completed without errors.
BUSY	OUT	Bool		-	• FALSE – no command active for Modbus_Master • TRUE – command for Modbus_Master in progress
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. The error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		0	Error code (see Error messages (Page 4029))

Tags in the data block of the Modbus master

The table below shows the public static tags in the instance DB of Modbus_Master that you can use in your program.

Table 4-163 Static tags in the instance DB

Tag	Data type	Standard	Description
Blocked_Proc_Timeout	Real	3.0	Duration (in seconds) for which to wait for a blocked Modbus master instance before this instance is removed as ACTIVE. This may happen, for example, if a master request was output and the program then stops to call the master function before it has completely finished the request. The time value must be greater than 0 and less than 55 seconds to avoid an error to occur. See also "Rules for communication by the Modbus-Master" and "Calling theModbus_Master instruction with different parameter settings".
Extended_Addressing	Bool	FALSE	Configures the slave station address as single or double byte. • FALSE = One-byte address; 0 to 247 • TRUE = Two-byte address (corresponds to extended addressing); 0 to 65535

Tag	Data type	Standard	Description
Compatibility_Mode [1]	Bool	FALSE	Compatibility mode with CP 341 and CP 441-2 and ET 200SP 1SI with driver for Modbus RTU and with ET 200SP 1SI for Modbus. The default value is 0. • FALSE = as per Modbus specification, not compatible • TRUE = compatible – For FC1 and FC2: The data read from the received frame is written word for word to the addressed CPU memory and exchanged byte by byte. If the number of bits to be transmitted is not a multiple of 16, the bits which are not relevant are set to null in the last word. – For FC15: The words to be transmitted are read word by word from the addressed memory and written byte by byte to the send frame. If the number of bits to be transmitted is not a multiple of 8, the bits in the last byte which are not relevant are read unchanged from the addressed memory and entered in the send frame.
MB_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to this MB_DB parameter of the Modbus_Master instruction.

[1] The PtP communication modules respond as defined in the Modbus specification. To retain a response as with CP 341, CP 441-2 and ET 200SP 1SI for Modbus, use the "Compatibility_Mode" parameter.

You program can write values to the Blocked_Proc_Timeout and Extended_Addressing tags to control the Modbus master operations.

Rules for communication by the Modbus-Master

- Modbus_Comm_Load must be run to configure a port so that the Modbus_Master instruction can communicate with this port.
- A port which is to be used as Modbus master must not be used by Modbus_Slave . You can use one or several instances of Modbus_Master [1] with this port. But all versions of the Modbus_Master must use the same instance DB for the port.
- The Modbus instructions do not use communication alarm events to control the communication process. Your program must query the Modbus_Master instruction for completed commands (DONE, ERROR).
- We recommend to call all executions of Modbus_Master for a specific port from a program cycle OB. Modbus master instructions can only be executed in one program cycle or in one cyclical/time-controlled processing level. They may not be processed in different processing levels. The priority interruption of a Modbus master instruction by another Modbus master instruction in a processing level with higher priority results in improper operation. Modbus master instructions may mot be processed in startup, diagnostic or time error levels.

[1] "Instance of Modbus master" here means a call of the Modbus_Master instruction with the same interconnection to a Modbus_Comm_Load instruction and the same setting for the MB_ADDR, MODE, DATA_ADDR and DATA_LEN parameters.

Example

Modbus_Master is called with MODE=0 and DATA_ADDR=10

This job is now active until it is completed with DONE=1 or ERROR=1 or until the time monitoring configured at the Blocked_Proc_Timeout parameter has expired. If a new command is started after the watchdog time expires and before the previous command has been completed, the previous command is aborted without an error message.

If, while this command is running, the instruction is now called a second time with the same instance data but different MODE and DATA_ADDR parameter settings, this second call is terminated with ERROR=1 and STATUS=8200.

Calling the Modbus_Master instruction with different parameter settings

If multiple calls of the Modbus_Master instruction with different settings for MB_ADDR, MODE, DATA_ADDR or DATA_LEN are placed in your program, you must ensure that only one of these calls is active at any given time. Otherwise, the error message 16#8200 is output (interface is busy with an ongoing request).

If a call cannot be processed in full, the watchdog is activated by the Blocked_Proc_Timeout parameter and terminates the ongoing command.

REQ parameter

FALSE = no request; TRUE = request to send data to the Modbus slave

Enable the requested transmission. This transmits the contents of the buffer to the point-to-point communication interface.

You use the DATA_ADDR and MODE parameters to select the Modbus function code.

DATA_ADDR (Modbus start address in the slave): Specifies the start address of the data that is accessed in the Modbus slave.

The Modbus_Master instruction uses the MODE input instead of a function code input. The combination of MODE and DATA_ADDR specifies the function code that is used in the actual Modbus frame. The table below shows how the MODE parameter, the Modbus function code and the Modbus address range in DATA_ADDR are related.

Table 4-164 Modbus functions

MODE	DATA_ADDR (Modbus address)			DATA_LEN (data length)			Modbus function code	Operation and data		
0				Bits per request			01	Read output bits:		
	1	to	9999	1	to	2000/1992 [1]		0	to	9998
0				Bits per request			02	Read input bits:		
	10001	to	19999	1	to	2000/1992 [1]		0	to	9998
0				Words per request			03	Read hold register:		
	40001	to	49999	1	to	125/124 [1]		0	to	9998
	400001	to	465535	1	to	125/124 [1]		0	to	65534
0				Words per request			04	Read input words:		
	30001	to	39999	1	to	125/124 [1]		0	to	9998

MODE	DATA_ADDR (Modbus address)			DATA_LEN (data length)			Modbus function code	Operation and data		
1				Bits per request			05	Write one output bit:		
	1	to	9999	1				0	to	9998
1				1 word per request			06	Write one hold register:		
	40001	to	49999	1				0	to	9998
	400001	to	465535	1				0	to	65524
1				Bits per request			15	Write multiple output bits:		
	1	to	9999	2	to	1968/1960 [1]		0	to	9998
1				Words per request			16	Write multiple hold registers:		
	40001	to	49999	2	to	123/122		0	to	9998
	400001	to	465534	2	to	123/122 [1]		0	to	65534
2 [2]				Bits per request			15	Write one or several output bits:		
	1	to	9999	1	to	1968/1960 [1]		0	to	9998
2 [2]				Words per request			16	Write one or several hold registers:		
	40001	to	49999	1	to	123		0	to	9998
	400001	to	465535	1	to	122 [1]		0	to	65534
11	Both DATA_ADDR and DATA_LEN operands of the Modbus_Master are ignored with this function.						11	Read status word and event counter of the slave communication. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful processing of a frame.		
80				1 word per request			08	Check slave status with data diagnostic code 0x0000 (loopback test – slave returns an echo of the request)		
	-			1				-		
81				1 word per request			08	Reset slave event counter using data diagnostic code 0x000A		
	-			1				-		
104 [3]				Words per request			04	Read input words		
	0	to	65535	1	to	125/124 [1]		0	to	65535
3 to 10, 12 to 79, 82 to 103, 105 to 255	-			-				Reserved		

[1] In extended addressing, see the Extended_Adressing parameter, the maximum data length is shorter by 1 byte or 1 word depending on the data type of the function.

[2] MODE 2 allows you to write one or more output bits and one or more holding registers using the Modbus functions 15 and 16. MODE 1 uses the Modbus functions 5 and 6 to write 1 output bit and 1 holding register, and Modbus functions 15 and 16 to write multiple output bits and multiple holding registers.

[3] The following applies to S7-300/400/WinAC: Reserved

DATA_PTR parameter

The DATA_PTR parameter points to the DB or bit memory address in which reading or writing is performed. If you use a data block, you must create a global data block that provides the data memory for read and write processes on Modbus slaves.

Note

S7-1200/1500 - The data block addressed using DATA_PTR must support direct addressing

The data block must permit direct (absolute) and symbolic addressing.

Note

Using function code 5

Function code 5 is used to set or delete individual bits.

When a bit is set, the value "16#FF00" must be specified in the first word of the addressed DB or bit memory area via DATA_PTR.

- With S7-1200, the value "16#0100" can also be specified to set a bit.
- To reset a bit, the value "16#0000" must be specified in the first word of the DB or bit memory area addressed via DATA_PTR.

All other values are rejected with ERROR = TRUE and STATUS = 16#8384.

Data block structures for the DATA_PTR parameter

- These data types are valid for reading words of the Modbus address range (DATA_PTR) 30001 to 39999, 40001 to 49999 and 400001 to 465535 as well as for writing words to the Modbus address range (DATA_PTR parameter) 40001 to 49999 and 400001 to 465535.
  - Standard array of data types WORD, UINT or INT
  - Named structure of the WORD, UINT or INT type in which each element has a unique name and a 16-bit data type.
  - Named complex structure in which each element has a unique name and a 16-bit or 32-bit data type.
- For reading and writing bits for the Modbus address range (DATA_PTR parameter) 00001 to 09999 and for reading bits from 10001 to 19999.
  - Standard field from Boolean data types.
  - Named Boolean structure from clearly named Boolean tags.
- It is not essential, but nevertheless advisable, to allocate each Modbus_Master instruction its own separate memory area. The reason for this is that data destruction is far more likely if multiple Modbus_Master instructions are reading and writing in the same memory area.
- It is not necessary for the data areas for DATA_PTR to be located in the same global data block. You can create a data block with several areas for Modbus read processes, a data block for Modbus write processes or a data block for each slave station.

Instruction versions

Version 3.0 is functionally identical to version 2.4 and its version number was only incremented due to internal measures.

See also

Modbus_Slave: Communicate as Modbus slave (Page 4015)

Modbus_Slave (S7-1200, S7-1500)

Modbus_Slave: Communicate as Modbus slave (S7-1200, S7-1500)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

Your program can use the Modbus_Slave instruction to communicate as a Modbus slave by using a CM (RS422/485 or RS232). STEP 7 automatically creates an instance DB when you add the instruction. The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the (static) MB_DB parameter of the Modbus_Slave instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
MB_ADDR	IN	UInt	Word	-	Standard address of the Modbus slave: Standard addressing range (1 to 247) Extended addressing range (0 to 65535) Note: 0 is the broadcast address
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Slave instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
MB_HOLD_REG	IN/OUT	Variant	Any	-	Pointer to the Modbus hold register DB: The Modbus hold register may be the memory area of the flags or a data block. As of instruction version V4.0: The parameter must point to a memory area that has a length of at least 16 bits. A shorter length results in error message 16#8187. This applies to single elements, arrays, STRUCTs and UDTs. For example, a Single Bool or an array consisting of less than 16 Boolean elements results in the error message. If the length is not a multiple of 16 bits, the remaining bits at the end of the memory area cannot be read or written by the Modbus_Slave instruction. The parameter may point to an optimized memory area. In the optimized memory area, a single element or an array is permitted with the following data types: Bool, Byte, Char, Word, Int, DWord, DInt, Real, USInt, UInt, UDInt, SInt, WChar. Every other data type results in error message 16#818C.
NDR	OUT	Bool		FALSE	New data available: • FALSE – No new data • TRUE – Indicates that new data was written by the Modbus master The NDR bit is TRUE for one cycle after the last request has been completed without errors.
DR	OUT	Bool		FALSE	Read data: • FALSE – No data read • TRUE – Indicates that the instruction has stored the data received by the Modbus master in the target area. The DR bit is TRUE for one cycle after the last request has been completed without errors.
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. If the execution was terminated with an error, the error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		0	Error code (see Error messages (Page 4029))

The function codes of the Modbus communication (1, 2, 4, 5 and 15) can read and write bits and words directly in the process image input and in the process image output of the CPU. The MB_HOLD_REG parameter must be defined as data type greater than one byte for these

function codes. The table below shows the sample assignment of Modbus addresses to the process image in the CPU.

Table 4-165 Assignment of Modbus addresses to the process image

Modbus functions						S7-1200			
Code	Function	Data area	Address area			Data area	CPU address		
01	Read bits	Output	0	to	8191	Process image output	O0.0	to	O1023.7
02	Read bits	Input	0	to	8191	Process image input	I0.0	to	I1023.7
04	Read words	Input	0	to	511	Process image input	IW0	to	IW1022
05	Write bit	Output	0	to	8191	Process image output	O0.0	to	O1023.7
15	Write bits	Output	0	to	8191	Process image output	O0.0	to	O1023.7

Table 4-166 Assignment of Modbus addresses to the process image

Modbus functions						S7-1500 / S7-300 / S7-400			
Function code	Function	Data area	Address area			Data area	CPU address		
01	Read bits	Output	0	to	9998	Process image output	O0.0	to	A1249.6
02	Read bits	Input	0	to	9998	Process image input	I0.0	to	I1249.6
04	Read words	Input	0	to	9998	Process image input	IW0	to	IW19996
05	Write bit	Output	0	to	9998	Process image output	O0.0	to	A1249.6
15	Write bits	Output	0	to	9998	Process image output	O0.0	to	A1249.6

Note

The available address area may be smaller, depending on the memory configuration of the CPU.

The function codes of the Modbus communication (3, 6, 16) use a Modbus hold register which is an address area in the memory area of the flags or a data block. The type of holding register is specified by the MB_HOLD_REG parameter of the Modbus_Slave instruction.

Note

S7-1200/1500 - type of the MB_HOLD_REG data block

The data block with Modbus hold register must permit direct (absolute) and symbolic addressing.

Table 4-167 Diagnostics functions

Modbus diagnostic functions of the S7-1200 Modbus_Slave		
Function codes	**Subfunction**	**Description**
08	0000H	Output request data of echo test: The Modbus_Slave instruction returns the echo of a received data word to the Modbus master.
08	000AH	Clear communication event counter: The Modbus_Slave instruction clears the communication event counter used for Modbus function 11.
11		Call communication event counter: The Modbus_Slave instruction uses an internal communication event counter to detect the number of successful Modbus read and Modbus write requests that are sent to the Modbus slave. The counter is not incremented for function 8, function 11 and broadcast requests. It is also not incremented for requests that result in communication errors (for example, parity or CRC errors).

The Modbus_Slave instruction supports broadcast write requests from Modbus masters as long as the requests include access to valid addresses. Modbus_Slave generates error code 16#8188 for function codes that are not supported by the broadcast function.

Variables of the Modbus slave in instruction version V3.0

This table below shows the public static tags in the instance data block of Modbus_Slave that you can use in your program.

Table 4-168 Variables of the Modbus slave

Tag	Data type	Standard	Description
HR_Start_Offset	Word	0	Specifies the start address of the Modbus hold register (default = 0)
QB_Start	Word	0	Start address of the valid writable addressing range of the outputs (byte 0 to 65535) Note: The variable is not available for S7-300, S7-400 and WinAC.
QB_Count	Word	0xFFFF	Number of output bytes that can be written by the Modbus master. Note: The variable is not available for S7-300, S7-400 and WinAC.
Extended_Addressing	Bool	FALSE	Extended addressing, configures slave addressing as single or double byte (FALSE = single byte address, TRUE = double byte address)
Request_Count	Word	0	The number of all requests received by this slave
Slave_Message_Count	Word	0	The number of requests received for this specific slave
Bad_CRC_Count	Word	0	The number of received requests that have a CRC error
Broadcast_Count	Word	0	The number of received broadcast requests
Exception_Count	Word	0	Modbus-specific errors that are acknowledged with an exception to the master
Success_Count	Word	0	The number of received requests without protocol errors for this specific slave
MB_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to this MB_DB parameter of the Modbus_Master instruction.

You program can write values to the HR_Start_Offset and Extended_Addressing tags and control the Modbus slave operations. The other tags can be read to monitor the Modbus status.

Rules for Modbus slave communication

- Modbus_Comm_Load must be run to configure a port so that the Modbus_Slave instruction can communicate by means of this port.
- If a port is to respond as slave to a Modbus master, this port may not be programmed with the Modbus_Master instruction.
- Only one instance of Modbus_Slave can be used with a specific port; otherwise you may encounter unexpected behavior.
- The Modbus instructions do not use communication alarm events to control the communication process. Your program must control the communication process by querying the Modbus_Slave instruction for completed send and receive processes.
- The Modbus_Slave instruction must be executed regularly with a frequency that allows a timely response to incoming requests of a Modbus master. We recommend executing Modbus_Slave in each cycle from a program cycle OB. Modbus_Slave can be executed from a cyclic interrupt OB but we do not recommend it, because excessive time delays in the interrupt program can temporarily block the execution of other interrupt programs.

Time control of the Modbus signal

Modbus_Slave must be executed regularly to receive each request of the Modbus master and respond accordingly. The frequency with which Modbus_Slave is executed depends on the timeout value specified for the response by the Modbus master. This can be seen in the figure below.

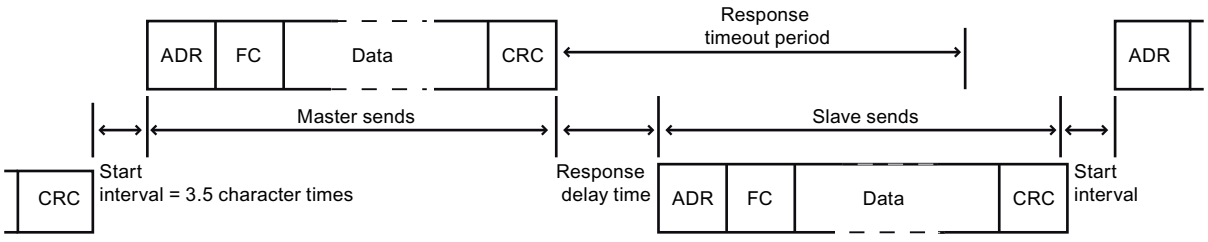

The timeout period of the (RESP_TO) response is the duration that a Modbus master waits for the beginning of an answer from a Modbus slave. This period is not defined by the Modbus protocol, but by a parameter of the Modbus_Comm_Load instruction. As both receiving and sending a frame requires multiple calls of the Modbus_Slave instruction (at least three), you should execute Modbus_Slave at least twelve times during the timeout period for the response of the Modbus master so that the Modbus slave receives and sends data twice as many times as specified by the timeout period.

HR_Start_Offset

The Modbus holding register addresses start at 40001 or 400001. These addresses correspond to the start address of the holding register in the target system memory. But you can configure the HR_Start_Offset tag to configure a start address different than 40001 or 400001 for the Modbus hold register.

The address 0 in the received frame correspond to the start address of the hold register in the target system memory. Use the tag HR_Start_Offset to configure a start address other than 0 for the Modbus hold register.

You can, for example, configure a hold register with start at MW100 and a length of 100 words. With HR_Start_Offset = 20, the address 20 in the received frame corresponds to the start address of the holding register in the target memory (MW100). Each address in the received frame below 20 and above 119 results in an addressing error.

Table 4-169 Example for addressing the Modbus hold register when DATA_PTR is a pointer to MW100 with a length of 100 words

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (word)	0	99
	S7-1500 address	MW100	MW298
20	Modbus address (word)	20	119
	S7-1500 address	MW100	MW298

HR_Start_Offset is a word value which specifies the start address of the Modbus hold register and is saved in the Modbus_Slave instance data block. You select this public static tag by means of the parameter drop-down list once you have added Modbus_Slave in your program.

If you have added Modbus_Slave to an LAD network, for example, you can go to a previous network and assign the value HR_Start_Offset using the move command. The value must be assigned prior to execution of Modbus_Slave.

Enter Modbus slave tag using the standard DB name:

1. Position the cursor in the OUT1 parameter field and enter the character m.
2. Select the required instance DB of the Modbus_Slave instruction from the drop-down list.
3. Position the cursor to the right of the DB name (after the quotation mark) and enter a point.
4. Select "Modbus_Slave_DB.HR_Start_Offset" in the drop-down list.

Instruction versions

Version 4.0 is functionally identical to version 3.0 and its version number was only incremented due to internal measures.

Access to data areas in DBs instead of direct access to MODBUS addresses as of version V5.0 (S7-1200, S7-1500)

Access to data areas in DBs instead of direct access to MODBUS addresses as of version V4.0

As of instruction version V4.0 of Modbus_Slave and as of firmware versions V2.5 (S7-1500 CPUs) or V4.2 (S7-1200 CPUs), you can access data areas in DBs instead of directly accessing process images and holding registers. In doing so, the attribute "Optimized block access" must be disabled for the DB and it must not be located solely in the load memory.

If a MODBUS request arrives and you have not defined a data area for the MODBUS data type of the corresponding function code, the request is treated as in the previous instruction versions, i.e. process images and holding registers are accessed directly.

If you have defined a data area for the MODBUS data type of the function code, however, the Modbus_Slave instruction reads from this data area or writes to it. Whether it reads or writes depends on the job type.

One individual MODBUS request can only ever be read from or written to one data area. If, for example, you want to read holding registers that extend over multiple data areas, you therefore require multiple MODBUS requests.

Rules for defining data areas

You can define up to eight data areas in different DBs; each DB must only contain one data area. An individual MODBUS request can only ever read from precisely one data area or write to precisely one data area. Each data area corresponds to one MODBUS address area. The data areas are defined in the static tag Data_Area_Array of the instance DB; Data_Area_Array is a field consisting of eight elements.

If you want to use less than eight data areas, the required data areas must be located one behind the other without any gaps. The first blank entry in the data areas ends the data area search during processing. If, for example, you have defined the field elements 1, 2, 4 and 5, only field elements 1 and 2 will be recognized as field element 3 is empty.

The Data_Area_Array field consists of 8 elements: Data_Area_Array[1] to Data_Area_Array[8]

Each field element Data_Area_Array[x], 1 <= x <= 8, is a UDT of the type MB_DataArea and is structured as follows:

Parameter	Data type	Meaning
data_type	UInt	Identifier for the MODBUS data type that is mapped to this data area: • 0: Identifier for an empty field element or an unused data area. In this case, the values of db, start and length are irrelevant. • 1: Process image output (used with function codes 1, 5 and 15) • 2: Process image input (used with function code 2) • 3: Holding register (used with function codes 3, 6 and 16) • 4: Input register (used with function code 4) Note: If you have defined a data area for a MODBUS data type, the instruction MB_SERVER can no longer access this MODBUS data type directly. If the address of a MODBUS request for such a data type does not correspond to a defined data area, a value of W#16#8383 is returned in STATUS.
db	UInt	Number of the data block to which the MODBUS register or bits subsequently defined are mapped. The DB number must be unique in the data areas. The same DB number must not be defined in multiple data areas. The DB must have standard access and must not be located solely in the load memory. Data areas also start with the byte address 0 of the DB. Permitted values: 1 to 60999
start	UInt	First MODBUS address that is mapped to the data block starting from address 0.0. Permitted values: 0 to 65535
length	UInt	Number of bits (for the values 1 and 2 of data_type) or number of registers (for the values 3 and 4 of data_type). The MODBUS address areas of one and the same MODBUS data type must not overlap. Permitted values: 1 to 65535

Examples of the definition of data areas

- First example: data_type = 3, db = 1, start = 10, length = 6
  The holding registers (data_type = 3) are mapped in data block 1 (db = 1). The Modbus address 10 (start = 10) is located at data word 0. The last valid Modbus address 15 (length = 6) is located at data word 5.
- Second example: data_type = 2, db = 15, start = 1700, length = 112
  The inputs (data_type = 2) are mapped in data block 15 (db = 15). The Modbus address 1700 (start = 1700) is located at data word 0. The last valid Modbus address 1811 (length = 112) is located at data word 111.

Restriction of read access to process images as of version V5.0 (S7-1200, S7-1500)

Restriction of read access to process images

As of instruction version V4.0 of Modbus_Slave, you can define one area each in the process image of the inputs and in the process image of the outputs to which remote MODBUS devices have read access. Read access by remote MODBUS devices to addresses outside these process image areas is then no longer possible.

Note

Restriction of write access to process images

The option for restricting write access to the process image of the outputs to a specific area is available as of instruction version V3.0.

Definition of read areas in the process images

Read areas in the process images are defined in the following static tags of the instance DB:

- QB_Read_Start: Address of the first byte in the process image output that can be read by a remote MODBUS device (applies to function code 1)
- QB_Read_Count: Number of bytes in the process image output that can be read by a remote MODBUS device (applies to function code 1)
- IB_Read_Start: Address of the first byte in the process image input that can be read by a remote MODBUS device (applies to function codes 2 and 4)
- IB_Read_Count: Number of bytes in the process image input that can be read by a remote MODBUS device (applies to function codes 2 and 4)

Static tags in the instance DB for defining write and read areas in the process images

The following table describes the static variables listed above in the instance DB of the Modbus_Slave instruction that you use to define the read areas in the process images.

For the sake of completeness, the static variables with which you define the write areas in the process images (QB_Start and QB_Count) as of version V3.0 are also specified.

Tag	Data type	Start value
QB_Start	UInt	0
QB_Count	UInt	65535
QB_Read_Start	UInt	0
QB_Read_Count	UInt	65535
IB_Read_Start	UInt	0
IB_Read_Count	UInt	65535

Frame structure (S7-1200, S7-1500)

Extended_Addressing

You access the Extended_Addressing tag as described for the HR_Start_Offset reference, except that the Extended_Addressing tag is a Boolean value.

You can configure a single byte (Modbus standard) or two bytes (Extended_Adressing = TRUE) with Extended_Adressing = FALSE for addressing the Modbus slave. Extended addressing is used to address more than 247 devices in a single network. With Extended_Adressing = TRUE you can address up to 65535 addresses. The following example shows a Modbus frame.

Table 4-170 Slave address with one byte (Byte 0)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	
Request	Slave address	Function code	Start address		Data		
Valid response	Slave address	Function code	Length	Data...			
Error message	Slave address	0xxx	Exception code				

Table 4-171 Slave address with two bytes (Byte 0 and Byte 1)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Request	Slave address		Function code	Start address		Data	
Valid response	Slave address		Function code	Length	Data...		
Error message	Slave address		0xxx	Exception code			

Frame description

Data traffic between master and slave / slave and master starts with the slave address, following by the function code. The data is then transferred. The structure of the data field depends on the function code used. The checksum (CRC) is transmitted at the end of the frame.

Function code 1 - This function allows individual output bits to be read

Table 4-172 FC 1 - Read output bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 1	Start address		Number of outputs	
Valid response	Slave address	Function code 1	Length [1]	Output data [3)]		
Error message	Slave address	0x81	Exception code [2]	---		

[1] Length: If there is a remainder when the number of outputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

[3] The output data can comprise multiple bytes

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 1	Start address		Number of outputs	
Valid response	Slave address		Function code 1	Length [1]	Output data		
Error message	Slave address		0x81	Exception code [2]	---		

[1] Length: If there is a remainder when the number of outputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

[3] The output data can comprise multiple bytes

Function code 2 - This function allows individual input bits to be read

Table 4-173 FC 2 - Read input bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 2	Start address		Number of inputs	
Valid response	Slave address	Function code 2	Length [1]	Input data		
Error message	Slave address	0x82	Exception code [2]	---		

[1] Length: If there is a remainder when the number of inputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 2	Start address		Number of inputs	
Valid response	Slave address		Function code 2	Length [1]	Input data		
Error message	Slave address		0x82	Exception code [2]	---		

[1] Length: If there is a remainder when the number of inputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

Function code 3 - This function allows individual registers to be read

Table 4-174 FC 3 - Read hold register

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 3	Start address		Number of registers	
Valid response	Slave address	Function code 3	Length [1]	Register data		
Error message	Slave address	0x83	Exception code [2]	---		

[1] Length: Number of bytes

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 3	Start address		Number of registers	
Valid response	Slave address		Function code 3	Length [1]	Register data		
Error message	Slave address		0x83	Exception code [2]	---		

[1] Length: Number of bytes

[2] E code: 01 or 02 or 03 or 04

Function code 4 - This function allows individual registers to be read

Table 4-175 FC 4 - Read input words

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 4	Start address		Number of input words	
Valid response	Slave address	Function code 4	Length [1]	Input data		
Error message	Slave address	0x84	Exception code [2]	---		

[1] Length: 2 * number of input words

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 4	Start address		Number of input words	
Valid response	Slave address		Function code 4	Length [1]	Input data		
Error message	Slave address		0x84	Exception code [2]	---		

[1] Length: 2 * number of input words

[2] E code: 01 or 02 or 03 or 04

Function code 5 - This function can set or delete individual bits

Table 4-176 FC 5 - Write an output bit

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 5	Start address		Value	
Valid response	Slave address	Function code 5	Length	Value		
Error message	Slave address	0x85	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 5	Start address		Value	
Valid response	Slave address		Function code 5	Length	Value		
Error message	Slave address		0x85	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

Function code 6 - This function allows individual registers to be written

Table 4-177 FC 6 - Write hold register

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 6	Address		Register	
Valid response	Slave address	Function code 6	Address		Register	
Error message	Slave address	0x86	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 6	Address		Register	
Valid response	Slave address		Function code 6	Address		Register	
Error message	Slave address		0x86	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

Function code 8 - This function is used to check the communication connection

Table 4-178 FC 8 - Slave status

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 8	Diagnostic code		Test value	
Valid response	Slave address	Function code 8	Diagnostic code		Test value	
Error message	Slave address	0x88	Exception code [1]	---		

[1] E code: 01 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 8	Diagnostic code		Test value	
Valid response	Slave address		Function code 8	Diagnostic code		Test value	
Error message	Slave address		0x88	Exception code [1]	---		

[1] E code: 01 or 03 or 04

Function code 11 - This function can read 2 bytes of "Status word" and 2 bytes of "Event counter"

Table 4-179 FC 11 - Event counter for slave communication

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 11	---			
Valid response	Slave address	Function code 11	Status		Event counter	
Error message	Slave address	0x8B	Exception code [1]	---		

[1] E code: 01 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 11	---			
Valid response	Slave address		Function code 11	Status		Event counter	
Error message	Slave address		0x8B	Exception code [1]	---		

[1] E code: 01 or 04

Function code 15 - This function allows multiple bits to be written

Table 4-180 FC 15 - Write one/multiple output bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte n
Query	Slave address	Function code 15	Start address		Number of output words		Byte counter [1]	Value	
Valid response	Slave address	Function code 15	Start address		Number of output words		---		
Error message	Slave address	0x8F	Exception code [2]	---					

[1] Byte counter: If there is a remainder when the number of bytes is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte n
Query	Slave address		Function code 15	Start address		Number of output words		Byte counter [1]	Value	
Valid response	Slave address		Function code 15	Start address		Number of output words		---		
Error message	Slave address		0x8F	Exception code [2]		---				

[1] Byte counter: If there is a remainder when the number of bytes is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

Function code 16 - This function allows individual or multiple registers to be written

Table 4-181 FC 16 - Write one/multiple hold registers

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte n
Query	Slave address	Function code 16	Start address		Number of registers		Byte counter [1]	Value	
Valid response	Slave address	Function code 16	Start address		Number of registers		---		
Error message	Slave address	0x90	Exception code [2]	---					

[1] Byte counter: Number of registers * 2

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte n
Query	Slave address		Function code 16	Start address		Number of registers		Byte counter [1]	Value	
Valid response	Slave address		Function code 16	Start address		Number of registers		---		
Error message	Slave address		0x90	Exception code [2]		---				

[1] Byte counter: Number of registers * 2

[2] E code: 01 or 02 or 03 or 04

Error messages (S7-1200, S7-1500)

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-

Error code	Description	Remedy
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#7002	Interim call: Data transmission running	-
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
16#8193	The module does not support supply voltage diagnostics L+.	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).

Error code	Description	Remedy
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire operation (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire operation. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).

Error code	Description	Remedy
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see AUTOHOTSPOT).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)

Error code	Description	Remedy
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (bytes) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer on startup is deactivated (0) • Clear receive buffer on startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end delimiters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.

Error code	Description	Remedy
16#81DF	The module has reset the interface to the FB for one of the following reasons: • Module was restarted • Module parameters were reassigned • CPU STOP	—
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Break in receive line to partner. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (bytes) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
16#81F2	The DSR signal cannot be set because the module has the type DTE.	Check the configured type of the module. The module type must be DCE (data communication equipment).

Error code	Description	Remedy
16#81F3	The DTR signal cannot be set because the module has the type DCE.	Check the configured type of the module. The module type must be DTE (data terminal equipment).
16#81F4	Block header error (e.g. incorrect block type or incorrect block length)	Check the instance DB and the block header.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditions points to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 KB • Non-optimized memory area: 4 KB Note: If the pointer points to an optimized memory area, do not send more than 1 KB.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in the RDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter before the 1st call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 [1)]	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 [1)]	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		

Error code	Description	Remedy
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 [1)]	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 [1)]	Receive_Conditions does not point to a structure of the typeReceive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.

Error code	Description	Remedy
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the frame header and data string or the connection is terminated with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.

Error code	Description	Remedy
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.

Error code	Description	Remedy
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 4009) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 4009) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 4009) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#80C9	The slave does not respond within the time set by Blocked_Proc_Timeout.	Check the setting for Blocked_Proc_Timeout. Check if the module has been configured with the Modbus_Comm_Load instruction. The module may possibly need to be reconfigured using Modbus_Comm_Load after a pull/plug or after voltage recovery.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.

Error code	Description	Remedy
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1)]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1)]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1)]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1)]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1)]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		
16#8428 [1)]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during call of SFC51	Call the Modbus_Slave instruction again
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200% to +200%
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request. Check if the instructions USS_Read_Param, USS_Write_Param or USS_Port_Scan have reported an error. If they have, check the value of the static tag USS_DB. w_USSExtendedError of the USS_Drive_Control instruction.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.

Error code	Description	Remedy
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

ET 200S serial interface (S7-1200, S7-1500)

Notes on use of an S7-1500 (S7-1200, S7-1500)

S_SEND / S_RCV / S_USSI / S_MODB on S7-1500

Note the following when using the S_SEND, S_RCV, S_USSI or S_MODB instructions of the ET 200S 1SI distributed I/O on an S7-1500:

- S_SEND
  Data type of the parameter DB_NO: UInt.
  Only data from non-optimized memory areas can be sent. Keep this in mind during assignment of the DB_NO, DBB_NO and LEN parameters.
- S_RCV
  Data type of the parameter DB_NO: UInt.
  Only data from non-optimized memory areas can be received. Keep this in mind during assignment of the DB_NO, DBB_NO and LEN parameters.
- S_USSI
  - Data type of the parameters DBND, DBPA and DBCP: UInt
    Invalid DB numbers for the DBND, DBPA and DBCP parameters: 60000 to 60999
  - Reaction of CPU to error
    The CPU is not switched to STOP in response to error.
    If the CPU is to switch to STOP in response to error, the ANZ parameter must be evaluated in the user program
    (if ANZ is not equal to zero, the STP instruction must be started):
- S_MODB
  Data type of the parameter DB_NO: DB_ANY.
  Only data from non-optimized memory areas can be exchanged. Keep in mind during parameter assignment of the DB_NO parameter and the data block with the Modbus data conversion table.

Instructions of the "Communication processor > ET200S serial interface" library

Note the following when using the instructions of the ET 200S 1SI distributed I/O on an S7-1500 (library: "Communication processor > ET200S serial interface"):

- The hardware identifier of the communication module must be configured at the LADDR parameter.
- The send buffer and the receive buffer must both be located outside the optimized memory area.

Instructions of ET 200S 1SI (S7-1200, S7-1500)

You can find descriptions of the instructions for ET 200S 1SI in the following sections:

- ET 200S 1SI for 3964(R) and ASCII (Page 6450)
- ET 200S 1SI for Modbus (Page 6468)
- ET 200S 1SI for USS (Page 6485)

Point-to-point (S7-1200)

PORT_CFG: Configure communication parameters dynamically (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

The instruction "PORT_CFG" allows dynamic configuration of communications parameters for a point-to-point communications port.

You set up the original static configuration of the port in the hardware configuration. You can change this configuration by executing the "PORT_CFG" instruction. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it.

With "PORT_CFG" you can influence the following communications parameter settings:

- Parity
- Baud rate
- Number of bits per character
- Number of stop bits
- Type and properties of flow control

The changes made by the "PORT_CFG" instruction are not stored permanently on the target system.

You can transfer serial data via the electrical connections RS-232 (half and full duplex) and RS-485 (half duplex).

Parameters

The following table shows the parameters of the "PORT_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the configuration change on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
PROTOCOL	Input	UINT	I, Q, M, D, L or constant	Transmission protocol: • 0: Point-to-point communication protocol • 1..n: Future definition for specific transmission protocols
BAUD	Input	UINT	I, Q, M, D, L or constant	Baud rate of the port: • 1: 300 baud • 2: 600 baud • 3: 1200 baud • 4: 2400 baud • 5: 4800 baud • 6: 9600 baud (default) • 7: 19200 baud • 8: 38400 baud • 9: 57600 baud • 10: 76800 baud • 11: 115200 baud
PARITY	Input	UINT	I, Q, M, D, L or constant	Parity of the port: • 1: No parity (default) • 2: Even parity • 3: Odd parity • 4: Mark parity • 5: Space parity
DATABITS	Input	UINT	I, Q, M, D, L or constant	Bits per character: • 1: 8 bits per character (default) • 2: 7 bits per character
STOPBITS	Input	UINT	I, Q, M, D, L or constant	Number of stop bits: • 1: 1 stop bit (default) • 2: 2 stop bits

Parameter	Declaration	Data type	Memory area	Description
FLOWCTRL	Input	UINT	I, Q, M, D, L or constant	Data flow control: • 1: None (default) • 2: XON/XOFF • 3: Hardware flow control (RTS always activated) • 4: Hardware flow control (RTS can be deactivated during transmission)
XONCHAR	Input	CHAR	I, Q, M, D, L or constant	Indicates the character used as XON character. The character DC1 (11H) is set as default.
XOFFCHAR	Input	CHAR	I, Q, M, D, L or constant	Indicates the character used as XOFF character. The character DC3 (13H) is set as default.
WAITTIME	Input	UINT	I, Q, M, D, L or constant	Specifies the wait time for XON or CTS after the start of the transmission. The specified value must be greater than 0. 2000 milliseconds are set as default.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Error code* (W#16#...)	Description
80A0	The specified protocol is invalid.
80A1	The specified baud rate is invalid.
80A2	The specified parity rate is invalid.
80A3	The specified number of bits per character is invalid.
80A4	The specified number of stop bits is invalid.
80A5	The specified type of flow control is invalid.
80A6	Incorrect value at the WAITTIME parameter When the data flow control is enabled, the value at the WAITTIME parameter must be greater than zero.
80A7	Invalid values at XONCHAR and XOFFCHAR parameters.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in “General status information of the communications blocks (Page 4064)“.

See also

Switching display formats in the program status (Page 7698)

SEND_CFG: Configure serial transmission parameters dynamically (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

The instruction "SEND_CFG" allows dynamic configuration of serial transmission parameters for a point-to-point communications port. All the messages waiting for transfer are discarded after execution of SEND_CFG.

You set up the original static configuration of the port in the hardware configuration. You can change this configuration by executing the "SEND_CFG" instruction. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it. With "SEND_CFG" you can influence the following transmission parameter settings:

- Time between the activation of RTS (Request to Send) and the start of the transmission
- Time between the end of transmission and the deactivation of RTS
- Define bit times for breaks

The changes made by the "SEND_CFG" instruction are not stored permanently on the target system.

You can transfer serial data via the electrical connections RS-232 (half and full duplex) and RS-485 (half duplex).

Parameters

The following table shows the parameters of the "SEND_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the configuration change on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
RTSONDLY	Input	UINT	I, Q, M, D, L or constant	The time that should elapse after activating RTS until the start of transmission. Valid values for this parameter are as follows: • 0 (default) • 0 to 65535 ms in steps of 1 ms This parameter does not apply to RS-485 modules.

Parameter	Declaration	Data type	Memory area	Description
RTSOFFDLY	Input	UINT	I, Q, M, D, L or constant	Time that should elapse after the end of transmission until deactivation of RTS. Valid values for this parameter are as follows: • 0 (default) • 0 to 65535 ms in steps of 1 ms This parameter does not apply to RS-485 modules.
BREAK	Input	UINT	I, Q, M, D, L or constant	Specifies the bit times for a break, which are sent at the start of the message. 12 bit times are set as default. A maximum of 25000 bit times can be specified.
IDLELINE	Input	UINT	I, Q, M, D, L or constant	Specifies the bit times for idle line after the break at the start of the message. 12 bit times are set as default. A maximum of 25000 bit times can be specified.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Error code* (W#16#...)	Description
80B0	The configuration of a transmission interruption is not permitted.
80B1	The specified break time exceeds the permitted maximum of 25000 bit times.
80B2	The specified time for idle line exceeds the permitted maximum of 25000 bit times.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in “General status information of the communications blocks (Page 4064)“.

See also

Switching display formats in the program status (Page 7698)

RCV_CFG: Configure serial receive parameters dynamically (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

The instruction "RCV_CFG" allows dynamic configuration of serial receive parameters for a point-to-point communications port. You can use this instruction to configure the conditions that specify the start and end of the message to be transmitted. The receipt of messages that correspond to these conditions can be enabled by the "RCV_PTP (Page 4059)" instruction.

You set up the original static configuration of the port in the properties of the hardware configuration. Execute the "RCV_CFG" instruction in your program to change the configuration. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it. The changes made by the "RCV_CFG" instruction are not stored permanently on the target system.

All the messages waiting for transfer are discarded after execution of the "RCV_CFG" instruction.

Parameters

The following table shows the parameters of the "RCV_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the configuration change on a rising edge.
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
CONDITIONS	Input	CONDITIONS	D, L	Data structure defining the conditions for start and end of data transmission.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Data type CONDITIONS

You can use the CONDITIONS structure to define the start and end conditions for the message transmission. The structure CONDITIONS is included in the instance DB of the "RCV_CFG" instruction. Use the structure CONDITIONS to define the start and end conditions when the transmission of a message is complete and when the next message transfer is to start:

- You define the start conditions for the data transfer in the START structure.
- You define the end conditions for the data transfer in the END structure.

You can define one or more start and end conditions for this. If you specify multiple start or end conditions these are linked by an OR logic instruction.

The following table shows the "CONDITIONS" structure:

Parameters		Data type	Description
START		STRUCT	Start conditions
	STARTCOND	UINT	Specifies the start condition (details, see below). The start condition can be specified as a 16-bit hexadecimal value. Possible values for the start condition are: • 1: Start character • 2: Any character (default) • 4: Line break • 8: Idle line • 16: Character string 1 • 32: Character string 2 • 64: Character string 3 • 128: Character string 4 Multiple start conditions can also be defined at the STARTCOND parameter. The total from the values of the individual conditions is specified for this. If, for example, you want to define "Idle line" OR "Character string 1" OR "Character string 4" as start condition, the value "152" must be specified.
	IDLETIME	UINT	Specifies the maximum idle time of the line before receipt is started. Valid values for this parameter are as follows: • 40 bit times (default) • 0 to 2500 bit times
	STARTCHAR	BYTE	Specifies the start character. This setting is only enabled when the configured start condition is "Start character". Valid values for this parameter are as follows: • 02 (STX): Default setting • B#16#00 to B#16#FF
	SEQ[1].CTL	BYTE	Character string 1: Control sequence for each character You can use the bit position of the character to define which characters of the character string will be considered or ignored. To evaluate the characters, the corresponding bits have to be set. • Bit 0: 1 character • Bit 1: 2 characters • Bit 2: 3 characters • Bit 3: 4 characters • Bit 4: 5 characters A character is ignored when the corresponding bit is reset.
	SEQ[1].STR	CHAR[5]	Character string 1: Start character (5 characters)
	SEQ[2].CTL	BYTE	Character string 2: Ignore/compare control sequence for each character
	SEQ[2].STR	CHAR[5]	Character string 2: Start character (5 characters)
	SEQ[3].CTL	BYTE	Character string 3: Ignore/compare control sequence for each character
	SEQ[3].STR	CHAR[5]	Character string 3: Start character (5 characters)
	SEQ[4].CTL	BYTE	Character string 4: Ignore/compare control sequence for each character
	SEQ[4].STR	CHAR[5]	Character string 4: Start character (5 characters)

Parameters	Data type	Description
END	STRUCT	End conditions

Parameters		Data type	Description
	ENDCOND	UINT	Specifies the end condition (details, see below). The end condition can be specified as a 16-bit hexadecimal value. Possible values for the end condition are: • 1: Reply timeout • 2: Message timeout • 4: Timeout within the character string • 8: Maximum length • 16: N+LEN+M; the information on the message length in integrated in the message and will be evaluated. • 32: Character string 1 Multiple end conditions can also be defined at the ENDCOND parameter. The total from the values of the individual end conditions is specified for this. If, for example, you want to define the end condition "Maximum length" OR "Sequence 1", the value "40" must be specified.
	MAXLEN	UINT	Specifies the maximum number of characters in a message. Valid values* for this parameter are as follows: • 1 character (default) • 0 to 1024 characters This setting is only enabled if the "Maximum length" end condition is set at the ENDCOND parameter.
	N	UINT	Offset of the length field in the message Valid values for this parameter are as follows: • 0 characters (default) • 0 to 1024 characters This setting is only enabled if the "N+LEN+M" end condition is set at the END-COND parameter.
	LENGTHSIZE	UINT	Size of the length field in bytes Valid values* for this parameter are as follows: • 0 bytes (default) • 1 byte • 2 bytes • 4 bytes This setting is only enabled if the "N+LEN+M" end condition is set at the END-COND parameter.
	LENGTHM	UINT	Specifies the number of end characters that follow the length field but are not contained in the length of the message. Valid values for this parameter are as follows: • 0 characters (default) • 0 to 255 characters This setting is only enabled if the "N+LEN+M" end condition is set at the END-COND parameter.
	RCVTIME	UINT	Specifies the maximum duration for the receipt of the first character of a message. Valid values for this parameter are as follows: • 200 ms (default)

Parameters		Data type	Description
			• 0 to 65535 ms in steps of 1 ms This setting is only enabled if the "Reply timeout" end condition is set at the ENDCOND parameter.
	MSGTIME	UINT	Specifies the maximum duration of the receipt of a message. Valid values for this parameter are as follows: • 200 ms (default) • 0 to 65535 ms in steps of 1 ms This setting is only enabled if the "Message timeout" end condition is set at the ENDCOND parameter.
	CHARGAP	UINT	Specifies the time interval between received consecutive characters. Valid values for this parameter are as follows: • 12 bit times (default) • 0 to 2500 bit times This setting is only enabled if the "Timeout within the character string" end condition is set at the ENDCOND parameter.
	SEQ.CTL	BYTE	Character string: Control sequence for each character You can use the bit position of the character to define which characters of the character string will be considered or ignored. To evaluate the characters, the corresponding bits have to be set. • Bit 0: 1 character • Bit 1: 2 characters • Bit 2: 3 characters • Bit 3: 4 characters • Bit 4: 5 characters A character is ignored when the corresponding bit is reset.
	SEQ.STR	CHAR[5]	Character string: Start character (5 characters)
	* These value ranges also apply to the corresponding hardware settings for specifying the end of message.		

Start conditions for the message receipt (STARTCOND parameter)

The start of the message is recognized by the receiver if a configured start condition applies. The following conditions can be defined as start conditions for message receipt:

- Start character: The start of a message is recognized when a certain character occurs. This character is stored as first character of the message. All characters received before the start character are rejected.
- Any character: Any character can define the start of a message. This character is stored as first character of the message.
- Line break: The start of a message is recognized if the received data stream is interrupted for longer than one character length.

- Idle line: The start of a message is recognized when the send transmission line is in the idle state for a certain time (specified in bit times) followed by renewed transmission of characters.
- Character string (sequence): The start of a message is recognized when a specified character sequence occurs in the data stream. You can specify up to four character sequences with up to five characters each.
  Example: A received hexadecimal message includes the following characters: "68 10 aa 68 bb 10 aa 16". The configured start character sequences are listed in the following table. Start character sequences will be evaluated once the first character 68H has been received successfully. After the fourth character has been received successfully (the second 68H), the start condition "1" has been met. Once the start conditions have been met, evaluation of the end conditions will start.
  Processing of the start character sequence can end due to different errors in parity, framing or time intervals between characters. These errors will prevent reception of the message, because the start condition has not been met.

Start condition	First character	First character +1	First character +2	First character +3	First character +4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the message receipt (ENDCOND parameter)

The start of a message is recognized by the receiver if a configured end condition applies. The following conditions can be defined as end conditions for message receipt:

- Reply timeout: The receipt of messages will end when the specified maximum duration for the receipt of a character is exceeded. The maximum duration is defined at the RCVTIME parameter. The defined time starts to run down as soon at the last transmission is completed and the RCV_PTP instruction enables the receipt of the message. If no character was received within the defined time (RCVTIME), the RCV_PTP instruction reports an error.
- Message timeout: The receipt of messages will end when the specified maximum duration for the receipt of a message is exceeded. The maximum duration is defined at the MSGTIME parameter. The defined time starts to run down as soon as the first character of the message is received.
- Timeout within the character string: The receipt of messages will end when the time interval between the receipt of two consecutive characters is longer than the value at the CHARGAP parameter.
- Maximum length: The receipt of messages will end when the length of the message defined at the MAXLEN parameter is exceeded.

- Reading message length (N+LEN+M): The receipt of messages will end when a certain message length is reached. This length is calculated by the values of the following parameters:
  - N: Position of the character in the message, as of which the length field begins.
  - LENGTHSIZE: Size of the length field in bytes
  - LENGTHM: Number of end characters that follow the length field. These characters are not taken into account in the evaluation of the message length.
- Character string: The receipt of messages will end when a defined character sequence is received. The character string can contain a maximum of five characters. For each character of the character string, you can use the bit position to define if this will be considered or ignored in the evaluation.

STATUS parameter

Error code* (W#16#...)	Description
80C0	Error in start condition
80C1	• Error in end condition • No end condition defined
80C2	Receive interrupt enabled
80C3	A value that is equal to 0 or greater than 4132 was entered at the MAXLEN parameter while the "Maximum length" end condition was set.
80C4	A value that is greater than 4131 was entered at the N parameter while the "N+LEN+M" end condition was set.
80C5	A value that is equal to 0 or invalid was entered at the LENGTHSIZE parameter while the "N+LEN+M" end condition was set.
80C6	A value that is greater than 255 was entered at the LENGTHM parameter while the "N+LEN+M" end condition was set.
80C7	A message length greater than 4132 was calculated while the "N+LEN+M" end condition was set.
80C8	A value that is equal to 0 was entered at the RCVTIME parameter while the "Reply timeout" end condition was set.
80C9	A value that is equal to 0 or greater than 2500 was entered at the CHARGAP parameter while the "Timeout within a character string" end condition was set.
80CA	A value that is equal to 0 or greater than 2500 was entered at the IDLETIME parameter while the "Idle line" start condition was set.
80CB	All characters of the character string are marked as "Don't care" even though "Character string" is set as the end condition.
80CC	All characters of the character string are marked as "Don't care" even though "Character string" is set as the start condition.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in “General status information of the communications blocks (Page 4064)“.

See also

Switching display formats in the program status (Page 7698)

SEND_PTP: Transmit send buffer data (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

You use the "SEND_PTP" instruction to start the transmission of data. The "SEND_PTP" instruction does not execute the actual transmission of the data. The data of the send buffer is transmitted to the relevant point-to-point communication module (CM). The CM executes the actual transmission.

Parameters

The following table shows the parameters of the "SEND_PTP" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Enables the requested transmission on a rising edge of this enable input. The content of buffer is transmitted to the point-to-point communication module (CM).
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
BUFFER	Input	VARIANT	I, Q, M, D, L or constant	Pointer to the start address of the send buffer. Boolean values or Array of BOOL are not supported.
LENGTH	Input	UINT	I, Q, M, D, L or constant	Length of the send buffer
PTRCL	Input	BOOL	I, Q, M, D, L or constant	This parameter selects the buffer for normal point-to-point communication or for specific Siemens protocols implemented in the connected CM. FALSE = point-to-point operations controlled by the user program (only valid option)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Error code* (W#16#...)	Description
7000	The send operation is not active.
7001	The send operation is processing the first call.
7002	The send operation is processing subsequent calls (queries following the first call).
8080	The identifier entered for the communications port number is invalid.
8088	The length of the LENGHT parameter does not correspond to the length of data to be sent. See also: Parameters LENGHT and BUFFER.
80D0	A new send request was received while a transmission was taking place.
80D1	The transmission was interrupted because the CTS signal was not confirmed within the specified wait time.
80D2	The send request was interrupted because the communications partner (DCE) signaled that it was not willing to receive (DSR).
80D3	The send request was interrupted because the maximum size of the waiting loop was exceeded (more than 1024 Byte).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in “General status information of the communications blocks (Page 4064)“.

LENGTH and BUFFER parameters

The minimum data size that can be sent by the "PTP_SEND" instruction is one byte. The parameter BUFFER defines the size of the data to be sent. You can use neither the BOOL nor the Array of BOOL data type for the BUFFER parameter.

LENGTH parameter	BUFFER parameter	Description
LENGTH = 0	Not used	The complete data is sent as defined by the BUFFER parameter. If LENGTH = 0, you do not need to specify the number of bytes transferred.
LENGTH > 0	Elementary data type	The LENGTH value must contain the byte count of this data type. Otherwise, data is not transferred and error 8088 is output.
	STRUCT	The LENGTH value can contain a byte count that is smaller than the complete byte length of the structure. In this case, only the first LENGTH bytes are transferred.
	ARRAY	The LENGTH value can contain a byte count that is smaller than the complete byte length of the field. In this case, only the field elements that fit completely in the LENGTH bytes are transferred. The LENGTH value must be a multiple of the byte count of the data elements. Otherwise, STATUS = 8088, ERROR = 1, and no data is transferred.
	STRING	The complete memory arrangement of the character sequence format will be transmitted as well as the information about maximum length of the character string and the actual length of the character string. The LENGTH value must contain bytes for maximum length, actual length, and the characters of the character string. With the data type STRING, all lengths and characters have the size of one byte. If a character string is used for the BUFFER parameter, the LENGTH value must also contain two bytes for the two length fields.

See also

Switching display formats in the program status (Page 7698)

RCV_PTP: Enable receive messages (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

With the RCV_PTP instruction you enable receipt of a sent message. Each message must be enabled individually. The sent data is only available in the receive area when the message has been acknowledged by the relevant communications partner.

Parameters

The following table shows the parameters of the "RCV_PTP" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Enables receipt on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
BUFFER	Input	VARIANT	I, Q, M, D, L or constant	Points to the start address of the receive buffer. Do not use a tag of the type STRING in the receive buffer.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
LENGTH	Output	UINT	I, Q, M, D, L	Length of the message in the receive buffer

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Error code* (W#16#....)	Description
80E0	Receipt of messages was terminated because the receive buffer is full.
80E1	Receipt of messages was terminated as a result of a parity error.
80E2	Receipt of messages was terminated as a result of a framing error.
80E3	Receipt of messages was terminated as a result of an overflow error.
80E4	Receipt of messages was terminated because the calculated message length (N+LEN+M) exceeds the size of the receive buffer.
8080	The identifier entered for the communications port number is invalid.
8088	A data type STRING is referenced via the BUFFER parameter.
0094	Receipt of messages was terminated because the maximum character length was received.
0095	Receipt of messages was terminated as a result of a timeout.
0096	Receipt of messages was terminated because of a timeout within the character string.
0097	Receipt of messages was terminated as a result of a reply timeout.
0098	Receipt of messages was terminated because the "N+LEN+M" length condition has been satisfied.
0099	Receipt of messages was terminated because the character string defined as the end condition was received.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in “General status information of the communications blocks (Page 4064)“.

See also

Switching display formats in the program status (Page 7698)

RCV_RST: Delete receive buffer (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

With the "RCV_RST" instruction, you delete the receive buffer of a communications partner.

Parameters

The following table shows the parameters of the "RCV_RST" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Enables deleting of the receive buffer on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction You will find more detailed information on general error codes of the communication instructions in “General status information of the communications blocks (Page 4064)“.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

SGN_GET: Query RS-232 signals (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

With the "SGN_GET" instruction, you query the current state of several signals of an RS-232 communications module.

Parameters

The following table shows the parameters of the "SGN_GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Enables the query on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
NDR	Output	BOOL	I, Q, M, D, L	Is set for one cycle if new data are ready for sending and the instruction was executed error-free.
DTR	Output	BOOL	I, Q, M, D, L	Data terminal ready, module ready
DSR	Output	BOOL	I, Q, M, D, L	Data set ready, communications partner ready
RTS	Output	BOOL	I, Q, M, D, L	Send request, module ready to send
CTS	Output	BOOL	I, Q, M, D, L	Clear to send, communications partner can receive data (reaction to RTS = ON of the module).
DCD	Output	BOOL	I, Q, M, D, L	Data carrier detect, received signal level
RING	Output	BOOL	I, Q, M, D, L	Ring display, display of an incoming call
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Error code* (W#16#....)	Description
80F0	The communication module is an RS-485 module and no signals are available.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in “General status information of the communications blocks (Page 4064)“.

See also

Switching display formats in the program status (Page 7698)

SGN_SET: Set RS-232 signals (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

With the "SGN_SET" instruction, you set the status of the output signals of an RS-232 communications module.

Parameters

The following table shows the parameters of the "SGN_SET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the action on a rising edge Initial value: FALSE
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID) Initial value: FALSE
SIGNAL	Input	BYTE	I, Q, M, D, L or constant	Specifies the signals to be set: • Set 01H = RTS • Set 02H = DTR • Set 04H = DSR Initial value: FALSE
RTS	Input	BOOL	I, Q, M, D, L or constant	Send request, module ready to send Initial value: FALSE
DTR	Input	BOOL	I, Q, M, D, L or constant	Data terminal ready, module ready Initial value: FALSE
DSR	Input	BOOL	I, Q, M, D, L or constant	Data set ready (applies only to interfaces of the DCE type) Initial value: FALSE
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors Initial value: FALSE

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred Initial value: FALSE
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction Initial value: 0

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Error code* (W#16#...)	Description
80F0	The communication module is an RS-485 module and no signals are available.
80F1	No signals are settable because H/W flow control is enabled.
80F2	The DSR signal cannot be set because the module is a DTE device.
80F3	The DTR signal cannot be set because the module is a DCE device.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in “General status information of the communications blocks (Page 4064)“.

See also

Switching display formats in the program status (Page 7698)

General status information of the communications blocks (S7-1200)

General information on execution status of the communications blocks

The following table shows which general information can be output at the STATUS parameter of the communications blocks:

Error code* (W#16#....)	Description
0000	No error
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0.
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8x3A	Invalid pointer at parameter x.
8070	All internal instance memories are in use.
8080	The identifier entered for the communications port is invalid

Error code* (W#16#....)	Description
8081	Timeout, module error, internal error
8082	Parameter assignment failed because parameter assignment is currently being performed in the background.
8083	Buffer overflow: The CM or CB has returned a receipt message with a length that is greater than permitted by the length parameter.
8085	Error specifying the length at the LENGHT parameter. The specified length is "0" or greater than the maximum permitted value.
8090	Message length invalid, module invalid, message invalid
8091	Incorrect version in parameterization message
8092	Invalid record length in parameterization message
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 7698)

USS (S7-1200)

Overview of USS instructions (S7-1200)

Introduction

The USS instructions control the operation of drives that support the universal serial interface (USS). With the USS instructions, you can communicate with more than one drive via an RS-485 connection.

To do this, you require a CM 1241 RS-485 communications module or a CB 1241 RS-485 communications board. Up to three CM 1241 RS-485 modules and one CB 1241 RS-485 board can be installed in an S7-1200 CPU.

Each RS-485 port can operate up to sixteen drives.

The USS protocol uses a master/slave network for communication via a serial bus. The master uses an address parameter to send a message to a selected slave. A slave itself can never send without previously receiving a request. Direct exchange of messages between slaves is not possible. USS communication works in half duplex mode.

The following figure shows an example of a USS network diagram:

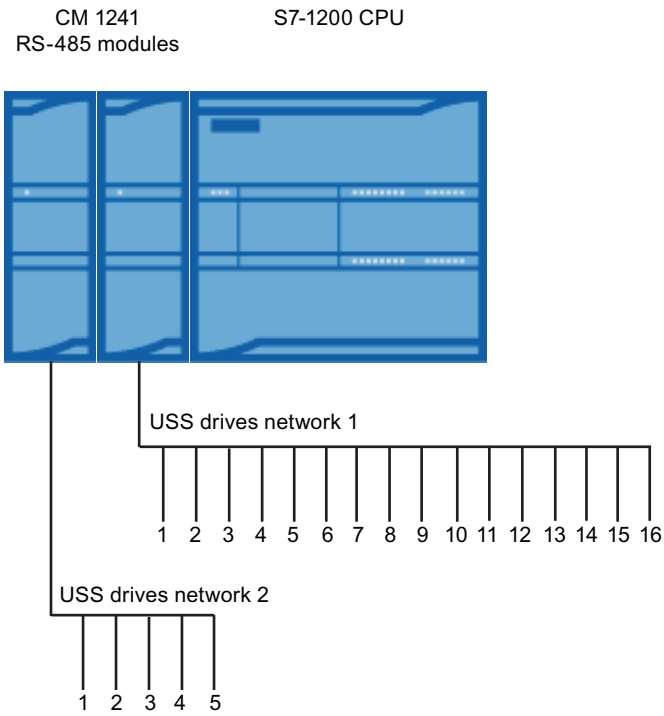

Requirements for using the USS protocol (S7-1200)

General requirements for setting up a drive

- The use of 4 PKW words must be set up for the drives.
- The drives can be configured for 2, 4, 6 or 8 PZD words.
- The number of PZD words in the drive must correspond to the PZD_LEN input of the "USS_DRIVE (Page 4070)" instruction of the drive.
- The baud rate of all drives must match the baud rate of the BAUD input parameter of the "USS_PORT (Page 4069)" instruction.
- The drive must be set up for remote control.
- USS must be specified for the desired frequency value on the COM connection of the drive.
- 1 to 16 must be set as the drive address. This address must correspond to the address of the DRIVE input parameter of the "USS_DRIVE (Page 4070)" instruction.
- To control the direction of the drive, the polarity of the drive desired value must be set up.
- The RS-485 network must be connected correctly.

Definition: PKW / PZD area

- The PKW area relates to the handling of the parameter-identifier-value interface.
  The PKW interface is not a physical interface but describes a mechanism that controls the exchange of parameters between two communications partners, in other words, reading and writing parameter values, parameter descriptions and corresponding texts and the handling of parameter changes due to spontaneous messages. All the tasks handled via the PKW interface are essentially tasks for operator control and monitoring, service and diagnostics.
- The PZD area includes the signals required for automation:
  - Control word(s), and setpoint(s) from the master to the slave
  - Status word(s), and actual value(s) from the slave to the master.

Both areas together result in the user data field. This is transferred by the master to the slave as a job frame or by the slave to the master as a reply frame.

Description

Each communication module CM 1241 RS485 supports a maximum of 16 drives. A single instance data block contains temporary memory and buffer functions for all drives in the USS network that are connected to a PtP communications module you installed. The USS instructions for these drives have shared access to the information in this data block.

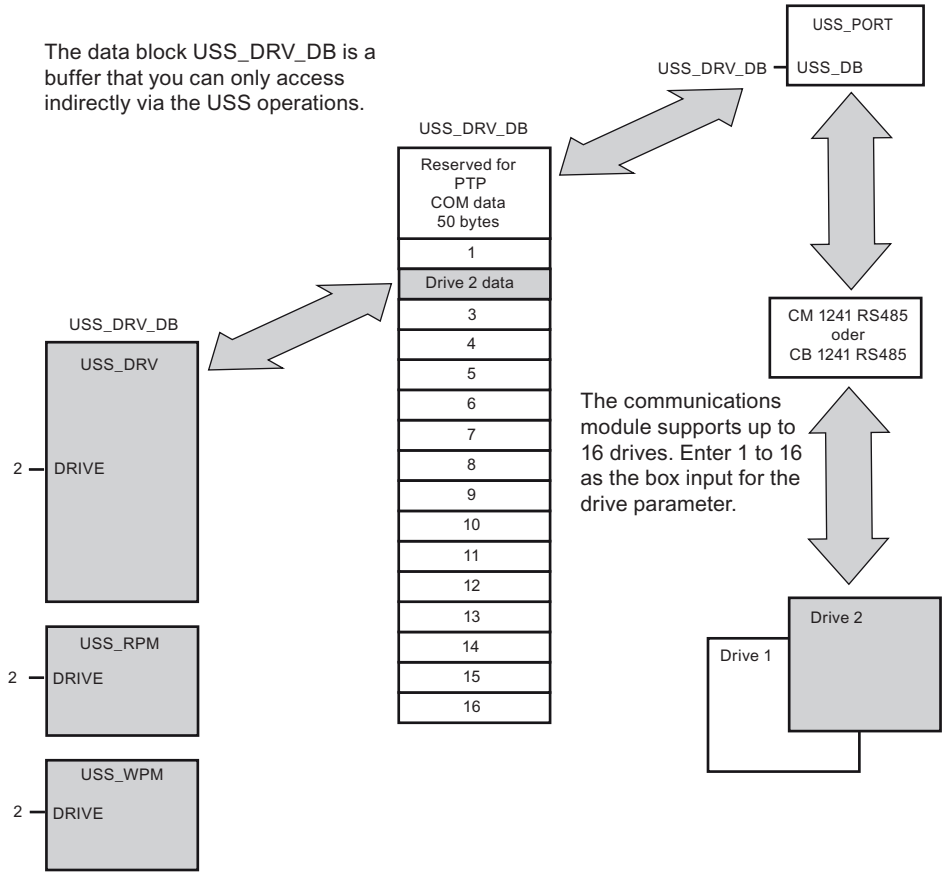

- All drives (max. 16) that are connected to an RS485 port are part of the same USS network. All drives that are connected to a different RS485 port are part of a different USS network. As the S7-1200 supports up to three CM 1241 RS485 modules, you can set up up to three USS networks, each having a maximum of 16 drives, thus a total of 48 USS drives are supported.
- Each USS network is managed via a unique data block (three data blocks are required for three USS networks with three CM 1241 RS485 modules). All instructions that belong to a USS network must share this data block. This includes all "USS_DRIVE (Page 4070)", "USS_PORT (Page 4069)", "USS_RPM (Page 4073)", and "USS_WPM (Page 4074)" instructions for controlling all drives in a USS network.
- The instruction "USS_DRIVE (Page 4070)" is a function block (FB). When you insert the instruction "USS_DRIVE" in the editor, the "Call options" dialog will ask you to assign a DB to the instruction.
  - If this is the first "USS_DRIVE" instruction in this program for this USS network, you can accept the default DB assignment (or, if necessary, change the name) and the new DB will be created.
  - If, however, this is not the first "USS_DRIVE" instruction for this network, you must select the DB that was previously assigned to this USS network in the drop-down list in the "Call options" dialog.
- All "USS_PORT (Page 4069), USS_RPM (Page 4073)", and "USS_WPM (Page 4074)" instructions are functions (FCs). When you insert these FCs in the editor, no DB is assigned. Instead, you have to assign the DB in question to the USS_DB input of these instructions (double-click on the parameter field and then on the icon to display the available DBs).
- The instruction "USS_PORT (Page 4069)" controls communication between the CPU and the drives via the PtP communications module. Whenever this instruction is called, communication with a drive is processed. Your program must call this function quickly enough so that the drives do not report a timeout. The instruction can be called from the main program or any interrupt OB.
- The function block "USS_DRIVE (Page 4070)" gives your program access to a specified drive in the USS network. Its inputs and outputs correspond to the states and operating functions of the drive. If there are 16 drives in the network, "USS_DRIVE" must be called at least 16 times in your program, i.e., once for each drive. How quickly these blocks are called depends on the required speed for controlling drive functions.
  You can only call the instruction "USS_DRIVE" from the main program OB.

  CAUTION

  Call "USS_DRIVE", "USS_RPM", "USS_WPM" only from the main program OB. The instruction "USS_PORT" can be called from any OB, it is usually called from a time-delay interrupt OB. If the instruction "USS_PORT" is interrupted during execution, this may result in unexpected errors.

The "USS_RPM" and "USS_WPM" instructions are used to read and write the operating parameters of the drive. These parameters control the internal mode of operation of the drive. A definition of these parameters can be found in the drive manual.

Your program may also contain any number of these instructions; however only one read or write request can be active for a drive. You may only call the instructions "USS_RPM" and "USS_WPM" from the main program OB.

Calculating the time for communication with the drive

Communication with the drive is runs asynchronous to the cycle of the S7-1200. The S7-1200 runs through several cycles before communication with a drive is completed.

The interval of "USS_PORT" is the time required for a drive transaction. The following table shows the minimum intervals for "USS_PORT" for each baud rate. Calling the "USS_PORT" more frequently than the "USS_PORT" interval will not increase the number of transactions. The timeout interval of the drive is the period of time available to a transaction if 3 attempts are required to complete the transaction due to communications errors. By default, up to 2 further attempts are made for each transaction with the USS protocol.

Baud rate	Calculated minimum interval for calling USS_PORT (ms)	Drive message interval timeout per drive (ms)
1200	790	2370
2400	405	1215
4800	212.5	638
9600	116.3	349
19200	68.2	205
38400	44.1	133
57600	36.1	109
115200	28.1	85

USS_PORT: Edit communication via USS network (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

The "USS_PORT" instruction handles communication over the USS network. In the program, use one "USS_PORT" instruction per PtP communications port to control the transmission to or from one drive.

All USS instructions that are assigned to one USS network and one PtP communications port must use the same instance data block.

Call

Your program must execute the "USS_PORT" instruction often enough to prevent timeouts in the drive. You should therefore call the "USS_PORT" instruction from a cyclic interrupt OB to prevent drive timeouts and keep the most recent USS data updates available for "USS_DRIVE (Page 4070)" calls.

Parameters

The following table shows the parameters of the "USS_PORT" instruction:

Parameter	Declaration	Data type	Memory area	Description
PORT	Input	PORT	D, L or constant	PtP communications port identifier Constant that can be referenced within the "Constants" tab of the default tag table.
BAUD	Input	DINT	I, Q, M, D, L or constant	Baud rate for USS communication.
USS_DB	InOut	USS_BASE	D	Reference to the instance DB of the "USS_DRIVE (Page 4070)" instruction.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR is set to TRUE if an error occurs. A corresponding error code will be output at the STATUS output.
STATUS (Page 4076)	Output	WORD	I, Q, M, D, L	Status value of the request. It indicates the result of the cycle or initialization. Additional information is available in the "USS_Extended_Error (Page 4076)" tag for some status codes.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

USS_DRIVE: Swap data with drive (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

The "USS_DRIVE" instruction exchanges data with the drive by creating request messages and interpreting the drive response messages. A separate instruction must be used for each drive, but all USS instructions assigned to one USS network and one PtP communications module must use the same instance data block. You must create the DB name when you place the first "USS_DRIVE" instruction. Then reuse this DB that was created when the initial instruction was inserted.

When the "USS_DRIVE" instruction is executed the first time, the drive indicated by the USS address (parameter DRIVE) is initialized in the instance DB. After this initialization, subsequent

"USS_PORT (Page 4069)" instructions can start communication with the drive at this drive number.

Changing the drive number requires a PLC STOP to RUN mode transition that initializes the instance DB. Input parameters are configured in the USS send buffer, and outputs are read from a "previous" valid response buffer if any exists. There is no data transmission during execution of the "USS_DRIVE" instruction. After executing “USS_PORT (Page 4069) there is communication with the drives, “USS_DRIVE" only configures the messages to be sent and evaluates data received in a previous request.

You can control the drive direction of rotation using either the DIR (BOOL) input or using the sign (positive or negative) at the SPEED_SP (REAL) input. The following table explains how these inputs work together to determine the drive direction, assuming the motor is wired for forward rotation.

SPEED_SP	DIR	Direction of rotation of drive
Value > 0	0	Reverse
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Reverse

Parameters

Expand the box to display all the parameters by clicking the bottom of the box. The parameter connections that are grayed are optional and do not need to be assigned.

The following table shows the parameters of the "USS_DRIVE" instruction:

Parameter	Declaration	Data type	Memory area	Description
RUN	Input	BOOL	I, Q, M, D, L or constant	Drive start bit: If this parameter has the value TRUE, this input enables the drive to run at the preset speed.
OFF2	Input	BOOL	I, Q, M, D, L or constant	"Electrical stop" bit: If this parameter has the value FALSE, this bit cause the drive to coast to a stop without braking.
OFF3	Input	BOOL	I, Q, M, D, L or constant	Fast stop bit - If this parameter has the value FALSE, this bit causes a fast stop by braking the drive.
F_ACK	Input	BOOL	I, Q, M, D, L or constant	Fault acknowledge bit - This bit resets the fault bit on a drive. This bit is set after the fault is cleared to indicate to the drive that it no longer needs to indicate the previous fault.
DIR	Input	BOOL	I, Q, M, D, L or constant	Drive direction control - This bit is set to indicate that the direction is forward (when SPEED_SP is positive).
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PZD_LEN	Input	USINT	I, Q, M, D, L or constant	Word length - This is the number of words of PZD data. Valid values are 2, 4, 6, or 8 words. The default is 2.
SPEED_SP	Input	REAL	I, Q, M, D, L or constant	Speed setpoint - This is the speed of the drive as a percentage of configured frequency. A positive value specifies forward direction (when DIR has the value TRUE).

Parameter	Declaration	Data type	Memory area	Description
CTRL3	Input	WORD	I, Q, M, D, L or constant	Control word 3 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL4	Input	WORD	I, Q, M, D, L or constant	Control word 4 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL5	Input	WORD	I, Q, M, D, L or constant	Control word 5 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL6	Input	WORD	I, Q, M, D, L or constant	Control word 6 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive.
CTRL7	Input	WORD	I, Q, M, D, L or constant	Control word 7 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL8	Input	WORD	I, Q, M, D, L or constant	Control word 8 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
NDR	Output	BOOL	I, Q, M, D, L	New data ready - If this parameter has the value TRUE, the bit indicates that the output contains data from a new communication request.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred - If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT" instruction.
STATUS (Page 4076)	Output	WORD	I, Q, M, D, L	Status value of the request. It indicates the result of the cycle. This is not a status word returned from the drive.
RUN_EN	Output	BOOL	I, Q, M, D, L	Run enabled - This bit indicates whether the drive is running.
D_DIR	Output	BOOL	I, Q, M, D, L	Drive direction - This bit indicates whether the drive is running forward.
INHIBIT	Output	BOOL	I, Q, M, D, L	Drive inhibited - This bit indicates the state of the inhibit bit on the drive.
FAULT	Output	BOOL	I, Q, M, D, L	Drive fault - This bit indicates that the drive has registered a fault. The user must eliminate the fault and then set the F_ACK bit to clear this bit.
SPEED	Output	REAL	I, Q, M, D, L	Drive current speed (scaled value of drive status word 2) - The value of the speed of the drive as a percentage of configured speed.
STATUS1	Output	WORD	I, Q, M, D, L	Drive status word 1 - This value contains fixed status bits of a drive.
STATUS3	Output	WORD	I, Q, M, D, L	Drive status word 3 - This value contains a user-configurable status word on the drive.
STATUS4	Output	WORD	I, Q, M, D, L	Drive status word 4 - This value contains a user-configurable status word on the drive.
STATUS5	Output	WORD	I, Q, M, D, L	Drive status word 5 - This value contains a user-configurable status word on the drive.

Parameter	Declaration	Data type	Memory area	Description
STATUS6	Output	WORD	I, Q, M, D, L	Drive status word 6 - This value contains a user-configurable status word on the drive.
STATUS7	Output	WORD	I, Q, M, D, L	Drive status word 7 - This value contains a user-configurable status word on the drive.
STATUS8	Output	WORD	I, Q, M, D, L	Drive status word 8 - This value contains a user-configurable status word on the drive.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

USS_RPM: Readout parameters from the drive (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

The "USS_RPM" instruction reads a parameter from the drive. All USS functions that are assigned to one USS network and one PtP communications module must use the same data block. "USS_RPM" must be called from the main program OB.

Parameter

The following table shows the parameters of the "USS_RPM" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Send request: If this parameter has the value TRUE, it indicates that a new read request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	Input	UINT	I, Q, M, D, L or constant	Parameter number: This input specifies which drive parameter is written. The range of this parameter is 0 to 2047. See your drive manual for details on how to access any parameters above this range.
INDEX	Input	UINT	I, Q, M, D, L or constant	Parameter index: This input specifies which drive parameter index is to be written. This is a 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for additional information.
USS_DB	InOut	USS_BASE	D	Reference to the instance DB that is created and initialized when a "USS_DRIVE" instruction is inserted in your program.

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	If this parameter has the value TRUE, it indicates that the VALUE output holds the previously requested read parameter value. This bit is set when the "USS_DRIVE" instruction recognizes the read response from the drive. This bit is reset when either: • you request the response data via another "USS_RPM" poll or • the second of the next two calls of "USS_DRIVE (Page 4070)" is executed
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred - If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT (Page 4069)" instruction.
STATUS (Page 4076)	Output	WORD	I, Q, M, D, L	This is the status value of the request. It indicates the result of the read request. Additional information is available in the "USS_Extended_Error (Page 4076)" tag for some status codes.
VALUE	Output	VARIANT	I, Q, M, D, L	This is the value of the parameter that was read and is valid only when the DONE bit has the value TRUE.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

USS_WPM: Change parameters in the drive (S7-1200)

Note

Use with CM1241

The use of this instruction with a CM1241 is possible only as of firmware version V2.1 of the module.

Description

The "USS_WPM" instruction modifies a parameter in the drive. All USS functions that are assigned to one USS network and one PtP communications module must use the same data block. "USS_WPM" must be called from the main program OB.

Note

EEPROM write operations

Beware of overusing the EEPROM write operation. Minimize the number of EEPROM write operations to extend the EEPROM life.

Parameter

The following table shows the parameters of the "USS_WPM" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Send request: If this parameter has the value TRUE, it indicates that a new write request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	Input	UINT	I, Q, M, D, L or constant	Parameter number: This input specifies which drive parameter is written. The range of this parameter is 0 to 2047. See your drive manual for details on how to access any parameters above this range.
INDEX	Input	UINT	I, Q, M, D, L or constant	Parameter index: This input specifies which drive parameter index is to be written. This is a 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for additional information.
EEPROM	Input	BOOL	I, Q, M, D, L or constant	Store to drive EEPROM: If this parameter has the value TRUE, values written to the drive parameter will be stored in the drive EEPROM. If this parameter has the value FALSE, the value written is only temporarily saved and will be lost the next time the drive is switched on.
VALUE	Input	VARIANT	I, Q, M, D, L or constant	The value of the parameter that is to be written. It must be valid on the transition of REQ.
USS_DB	InOut	USS_BASE	D	This is a reference to the instance DB that is created and initialized when a "USS_DRIVE (Page 4070)" instruction is inserted in your program.
DONE	Output	BOOL	I, Q, M, D, L	If this parameter has the value TRUE, the VALUE input was written to the drive. This bit is set when the "USS_DRIVE (Page 4070)" instruction recognizes the write response from the drive. This bit is reset when either: You request the drive's confirmation that the write operation has been completed via another "USS_WPM" query or when the second of the next two calls of "USS_DRIVE (Page 4070)" is executed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred: If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT (Page 4069)" instruction.
STATUS (Page 4076)	Output	WORD	I, Q, M, D, L	This is the status value of the request. It indicates the result of the write request. Additional information is available in the "USS_Extended_Error (Page 4076)" tag for some status codes.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS of USS instructions (S7-1200)

STATUS parameter

The following table contains the status codes of the USS operation that are output at the STATUS output of the USS instructions:

STATUS* (W#16#....)	Description
0000	No error
8180	The length of the drive response did not match the characters received from the drive. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8181	The parameter VALUE is not of the data type WORD, REAL, or DWORD
8182	User supplied a parameter value of the type word and received a DWORD or REAL from the drive in the response
8183	User supplied a parameter value of the type DWORD or REAL and received a word from the drive in the response
8184	Response telegram from drive had a bad checksum. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8185	Illegal drive address (valid drive address range: 1-16)
8186	Speed set point out of valid range (valid speed SP range: -200% to 200%)
8187	Wrong drive number responded to the request sent. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8188	Illegal PZD word length specified (valid range = 2, 4, 6 or 8 words)
8189	Illegal baud rate was specified
818A	Parameter request channel is in use by another request for this drive
818B	Drive has not responded to requests and retries. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
818C	Drive returned an extended error on a parameter request operation. See the extended error description below this table.
818D	Drive returned an invalid access error on a parameter request operation. See your drive manual for information of why parameter access may be limited
818E	Drive has not been initialized: This error code is output at "USS_RPM (Page 4073)" or "USS_WPM (Page 4074)" if the "USS_DRIVE (Page 4070)" instruction has not been called at least once for this drive. This prevents the initialization of the first cycle of "USS_DRIVE (Page 4070)" from overwriting a pending parameter read or write request since it initializes the drive as a new entry. To eliminate this error, call the "USS_DRIVE (Page 4070)" instruction for this drive.
80Ax-80Fx	Specific errors returned from PtP (Point-to-Point) communication instructions called by the USS library: These error code values are not modified by the USS library and are defined in the PtP instruction descriptions.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

USS_Extended_Error - USS drive extended error codes

USS drives support read and write access to a drive's internal parameters. This feature allows distributed control and configuration of the drive. Drive parameter access operations can fail due to errors such as values out of range or invalid requests in a drive's current mode. The drive generates an error code that is output in the "USS_Extended_Error" variable in the instance DB of the "USS_DRIVE (Page 4070)" instruction. This error code value is only valid for the last execution of the "USS_RPM (Page 4073)" or "USS_WPM (Page 4074)" instruction. The drive error code is put into the "USS_Extended_Error" tag when the value of STATUS is hexadecimal 818C. The error code of "USS_Extended_Error" depends on the drive variant. See the drive's manual for a description of the extended error codes for read and write parameter operations.

See also

Switching display formats in the program status (Page 7698)

MODBUS (RTU) (S7-1200)

MB_COMM_LOAD: Configure port on the PtP module for Modbus RTU (S7-1200)

Description

The "MB_COMM_LOAD" instruction configures a port for communication using the Modbus RTU protocol. The following hardware can be used for this:

- Up to three point-to-point modules (PtP) CM 1241 RS485 or CM 1241 RS232
- A communications board CB 1241 RS485 in addition to this

After configuration of the port, you communicate over Modbus by executing the "MB_SLAVE" or "MB_MASTER" instruction.

Call

"MB_COMM_LOAD" must be called once to configure the port for the Modbus RTU protocol. On completion of the configuration, the port can be used by the "MB_MASTER (Page 4080)" and "MB_SLAVE (Page 4088)" instructions.

"MB_COMM_LOAD" only needs to be called again if one of the communication parameters has to be modified. Each "MB_COMM_LOAD" call deletes the communications buffer. To avoid data loss during communication, you should not call the instruction unnecessarily.

One instance of "MB_COMM_LOAD" must be used to configure the port of each communication module that is used for Modbus communication. You assign a unique "MB_COMM_LOAD" instance data block for each port that you use. The S7-1200 CPU is limited to three communication modules.

An instance data block is assigned when you insert the "MB_MASTER (Page 4080)" or "MB_SLAVE (Page 4088)" instruction. This instance data block is referenced when you specify the MB_DB parameter on the "MB_COMM_LOAD" instruction.

Parameters

The following table shows the parameters of the "MB_COMM_LOAD" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Execution of the instruction upon a rising edge.
PORT	Input	PORT	I, Q, M, D, L or constant	ID of the communications port: After you have inserted the communications module in the device configuration, the port ID appears in the drop-down list at the PORT box connection. This constant can also be referenced within the "Constants" tab of the tag table.
BAUD	Input	UDINT	I, Q, M, D, L or constant	Baud rate selection: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 All other values are invalid.
PARITY	Input	UINT	I, Q, M, D, L or constant	Parity selection: • 0 – None • 1 – Odd • 2 – Even
FLOW_CTRL	Input	UINT	I, Q, M, D, L or constant	Flow control selection: • 0 – (default) No flow control • 1 – Hardware flow control with RTS always ON (does not apply to RS485 ports) • 2 - Hardware flow control with RTS switched
RTS_ON_DLY	Input	UINT	I, Q, M, D, L or constant	RTS on-delay selection: • 0 – (default) No delay of RTS active until the first character of the message is transmitted. • 1 to 65535 – Delay in milliseconds of "RTS active" until the first character of the message is transmitted (does not apply to RS-485 ports). RTS delays must be used depending on the selection of FLOW_CTRL.
RTS_OFF_DLY	Input	UINT	I, Q, M, D, L or constant	RTS off-delay selection: • 0 – (default) No delay after the last character transmitted until "RTS inactive" • 1 to 65535 – Delay in milliseconds after the last character transmitted until "RTS inactive" (does not apply to RS-485 ports). RTS delays must be used regardless of the selection of FLOW_CTRL.
RESP_TO	Input	UINT	I, Q, M, D, L or constant	Response timeout: Time in milliseconds allowed by "MB_MASTER (Page 4080)" for the slave to respond. If the slave does not respond in this time, "MB_MASTER (Page 4080)" repeats the request or terminates the request with an error if the specified number of retries has been sent. 5 ms to 65535 ms (default = 1000 ms).

Parameter	Declaration	Data type	Memory area	Description
MB_DB	Input	MB_BASE	D	A reference to the instance data block of the "MB_MASTER (Page 4080)" or "MB_SLAVE (Page 4088)" instructions. After you insert "MB_SLAVE (Page 4088)" or "MB_MASTER (Page 4080)" in your program, the DB identifier appears in the drop-down list at the MB_DB box connection.
DONE	Output	BOOL	I, Q, M, D, L	Execution of instruction completed without error.
ERROR	Output	BOOL	I, Q, M, D, L	Error: • 0 – No error detected • 1 – Indicates that an error was detected. An error code is output in the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Port configuration error code

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

Error code* (W#16#...)	Description
0000	No error
8180	Invalid value for the port ID (wrong address for the communications module).
8181	Invalid baud rate value.
8182	Invalid parity value.
8183	Invalid flow control value.
8184	Invalid value for the timeout of the response (the time before which a timeout is reported must be at least 25 ms).
8185	Incorrect pointer in the MB_DB parameter to the instance DB of the "MB_MASTER (Page 4080)" or "MB_SLAVE (Page 4088)" instruction.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

MB_COMM_LOAD data block tags

The table below shows the public static tags in the instance DB of MB_COMM_LOAD that you can use in your program.

Table 4-182 Static tags in the instance DB

Tag	Data type	Default	Description
ICHAR_GAP	WORD	0	Delay for the character spacing between characters. This parameter is specified in milliseconds and is used to increase the anticipated period between the received characters. The corresponding number of bit times for this parameter is added to the Modbus standard value of 35 bit times (3.5 character times).
RETRIES	WORD	2	The number of repeated attempts by the master before the error code 0x80C8 is returned for "No response".
STOP_BITS	USINT	1	Number of stop bits per character Valid values 1 and 2

See also

Switching display formats in the program status (Page 7698)

MB_MASTER: Communicate via the PtP port as Modbus master (S7-1200)

Description of MB_MASTER (S7-1200)

Description

The "MB_MASTER" instruction allows your program to communicate as a Modbus master using a port on a point-to-point module (CM) or a communications board (CB). You can access data in one or more Modbus slave devices.

Before the "MB_MASTER" instruction can communicate with a port, "MB_COMM_LOAD (Page 4077)" must first execute.

An instance DB is created when you insert the "MB_MASTER" instruction in your program. You specify this instance DB in the MB_DB input parameter of the "MB_COMM_LOAD (Page 4077)" instruction.

Rules for Modbus master communication

- A port used for Modbus master requests cannot be used for "MB_SLAVE".
- A port can be used for one or more "MB_MASTER" calls if the same instance DB is used.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must poll the "MB_MASTER" instruction for completed send and receive operations.

- Calling the instruction:
  - Call the "MB_MASTER" instruction if possible in a cyclic program OB. The instruction can only be called in a time delay or cyclic interrupt OB.
  - Do not call more than one "MB_MASTER" instruction in organization blocks with different priority classes. If a "MB_MASTER" instruction executes "preemptively" from a higher priority class, the instruction may execute incorrectly.
  - Do not call the "MB_MASTER" instruction in a startup, diagnostics or time error OB.
- After a transfer has started, the EN parameter (LAD/FBD) must remain set to the value "1" until the DONE or ERROR output parameter is set to "1" by the instruction. A renewed call by the REQ parameter while the instruction is executing causes an error. After the instruction executes, the bit in the REQ parameter remains set for the time specified by the BLOCKED_PROC_TIMEOUT parameter in the instance DB.
- If "MB_MASTER" sends a request to a slave, make sure that "MB_MASTER" continues to execute until the response from the slave arrives.

Parameters

The following table shows the parameters of the "MB_MASTER" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ (Page 4083)	Input	BOOL	I, Q, M, D, L	Request input: • 0 – No request • 1 – Request to transmit data to Modbus slave(s)
MB_ADDR	Input	UINT	I, Q, M, D, L or constant	Modbus RTU station address: • Default address range: 0 to 247 • Extended address range: 0 to 65535 The value "0" is reserved for broadcasting a message to all Modbus slaves. Modbus function codes 05, 06, 15, and 16 are the only function codes supported for broadcast.
MODE (Page 4083)	Input	USINT	I, Q, M, D, L or constant	Mode selection: Specifies the type of request: Read, write, or diagnostics: Refer to the Modbus functions table for details.
DATA_ADDR (Page 4083)	Input	UDINT	I, Q, M, D, L or constant	Starting address in the slave: Specifies the starting address of the data to be accessed in the Modbus slave. You will find the valid addresses in the Modbus functions table.
DATA_LEN	Input	UINT	I, Q, M, D, L or constant	Data length: Specifies the number of bits or words to be accessed in this request. You will find the valid lengths in the Modbus functions table.
DATA_PTR (Page 4085)	In_Out	VARIANT	M, D	Points to the DB or bit memory address of the CPU for the data to be written or read. For a DB, this must be created with the "Standard - compatible with S7-300/400" access type.
DONE	Output	BOOL	I, Q, M, D, L	• 0: Transaction not completed • 1: Transaction completed without error

Parameters	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	• 0: No "MB_MASTER" transaction in progress • 1: "MB_MASTER" transaction in progress
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error • 1: Error, the error code is indicated by the STATUS parameter
STATUS	Output	WORD	I, Q, M, D, L	Execution condition code

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

Table 4-183 Communications and configuration error messages of the instruction

Error code* (W#16#....)	Description
0000	No error
80C8	Slave timeout. Check the baud rate, parity and the connectors on the slave.
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission within the wait time. This error is also generated during hardware flow control if the recipient does not detect CTS within the wait time.
80D2	The send request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full.
80E1	The message was terminated as a result of a parity error.
80E2	The message was terminated as a result of a framing error.
80E3	The message was terminated as a result of an overrun error.
80E4	The message was terminated as a result of the specified length exceeding the total buffer size.
8180	Invalid value for the port ID.
8186	Invalid Modbus station address
8188	The MODE parameter has an invalid value for a broadcast call.
8189	Invalid data address value.
818A	Invalid data length value.
818B	Invalid pointer to the local data source/destination: Size not correct
818C	The DATA_PTR parameter has an invalid pointer. Use a pointer to a bit memory area or a DB with the "Standard - compatible with S7-300/400" access type.
8200	Port is busy processing a send request
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Table 4-184 Error messages of the Modbus protocol

Error code* (W#16#....)	Response code of slave	Description
8380	-	CRC error
8381	01	Function code not supported
8382	03	Data length error
8383	02	Error in the data address or address outside the valid range of DATA_PTR
8384	> 03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)
8386	-	Function code of the response does not match the function code of the query.
8387	-	Response from wrong slave
8388	-	The response of the slave to a write call is not correct. The data sent by the slave does not match the query from the master.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 7698)

Parameter REQ (S7-1200)

Description

- REQ = FALSE: No request
- REQ = TRUE: Request to transmit data to Modbus slave(s)

You can control this input by means of a level-controlled or edge-controlled contact.

A state machine is started every time this input is activated to ensure that another instruction "MB_MASTER" that uses the same instance DB can only issue a request when the current request has been processed. All other input states are recorded and saved internally for the current request until the reply has been received or an error has been detected.

If the same instance of "MB_MASTER" is executed again with REQ input = 1 before the current request is completely processed, no subsequent transfers are made. However, if the request has been processed, a new request is issued at the time when "MB_MASTER" is executed again with REQ input = 1.

DATA_ADDR and MODE parameters (S7-1200)

Description

You specify the start address for data access to the Modbus slave using the DATA_ADDR parameter.

With the MODE parameter and the Modbus address, you specify the function code to be transferred to the Modbus slave. The following table shows the relationship between the MODE parameter, the function code and Modbus address range.

MODE	Modbus function	Data length	Operation and data	Modbus address
0	01	1 to 2000 1 to 1992 (1)	Read output bits: 1 to (1992 or 2000) bits per query	1 to 9999
0	02	1 to 2000 1 to 1992 (1)	Read input bits: 1 to (1992 or 2000) bits per query	10001 to 19999
0	03	1 to 125 1 to 124 (1)	Read holding register: 1 to (124 or 125) WORD per query	40001 to 49999 or 400001 to 465535
0	04	1 to 125 1 to 124 (1)	Read input WORD: 1 to (124 or 125) WORD per query	30001 to 39999
1	05	1	Writing an output bit: One bit per query	1 to 9999
1	06	1	Writing a holding register: 1 WORD per query	40001 to 49999 or 400001 to 465535
1	15	2 to 1968 2 to 1960 (1)	Writing multiple output bits: 2 to (1960 or 1968) bits per query	1 to 9999
1	16	2 to 123 2 to 122 (1)	Writing multiple holding registers: 2 to (122 or 123) WORD per query	40001 to 49999 or 400001 to 465535
2	15	1 to 1968 2 to 1960 (1)	Writing one or more output bits: 1 to (1960 or 1968) bits per query	1 to 9999
2	16	1 to 123 2 to 122 (1)	Writing one or more holding registers: 1 to (122 or 123) WORD per query	40001 to 49999 or 400001 to 465535
11	11	0	Reading out the communications status word of the slaves and the event counter: The status word indicates execution of the instruction (0: is not executing; 0xFFFF: is executing). The event counter is incremented each time a message is transferred successfully. The DATA_ADDR and DATA_LEN parameters of the "MB_MASTER" instruction are ignored with this function.	-
80	08	1	Checking the slave status by reading the error code (0x0000): 1 WORD per query	-
81	08	1	Resetting the event counter of the slave with the diagnostics code 0x000A: 1 WORD per query	-
3 to 10, 12 to 79, 82 to 2555			Reserved	-

(1) For the "Extended address range", the maximum data length is reduced by one byte or one WORD depending on which data type is used for the function.

Parameter DATA_PTR (S7-1200)

Description

The DATA_PTR parameter is a pointer to a data block or bit memory from which the data should be written or read. If you use a data block, create a global data block with the "Standard - compatible with S7-300/400" access type.

Data block structures for the DATA_PTR parameter

- These data types are valid for **reading of words** of Modbus addresses 30001 to 39999, 40001 to 49999, and 400001 to 465536 and also for **writing of words** to Modbus addresses 40001 to 49999 and 400001 to 465536.
  - Standard array of WORD, UINT, or INT data types (see below).
  - Named WORD, UINT, or INT structure where each element has a unique name and 16 bit data type.
  - Named complex structure where each element has a unique name and 16 bit or 32 bit data type.
- For reading and writing of bits of Modbus addresses 00001 to 09999 and 10001 to 19999.
  - Standard array of Boolean data types.
  - Named Boolean structure of uniquely named Boolean variables.
- Although not required, it is recommended that each "MB_MASTER" instruction has its own separate memory area in a global data block. The reason for this recommendation is that there is a greater possibility of data corruption if multiple "MB_MASTER" instructions are reading and writing the same area of a global data block.
- The memory areas for DATA_PTR do not need to be in the same global data block. You can create one data block with multiple areas for Modbus read operations, one data block for Modbus write operations, or one data block for each slave station.

Instance DB of the "MB_MASTER" instruction (S7-1200)

Static variables of the instance DB

The following table describes the static variables of the instance DB of the instruction that you can use in the user program.

Variable	Data type	Description
MB_STATE	UINT	Internal status of the Modbus instruction
BLOCKED_ PROC_TIMEOUT	REAL	Time between completion of the instruction call and resetting the ACTIVE bit in the instance DB. The time buffer is used to avoid execution of the instruction being terminated before a job has been sent completely. The default time is 500 ms.
EXTENDED_ AD-DRESSING	BOOL	Configuring addressing: • 0: Default address area (1 byte) • 1: Extended address area (2 bytes) For additional information, refer to the section EXTENDED_AD-DRESSING: Instance DB of the "MB_SLAVE" instruction (Page 4091)

Sample program for a Modbus master (S7-1200)

Networks (LAD)

Network 1: Initialize parameters of the RS-485 module only once during the first cycle.

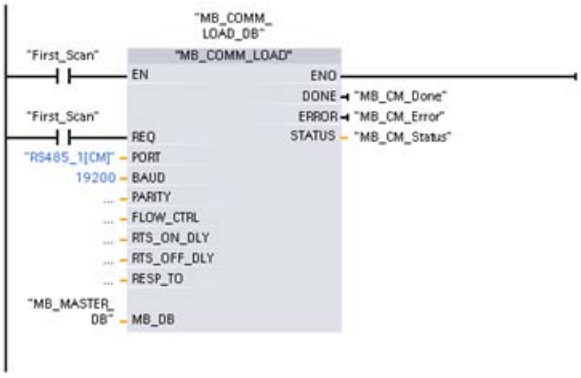

Network 2: Read 100 words from the holding register of the slave.

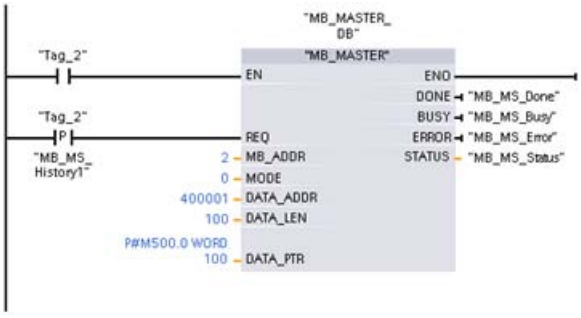

Network 3: This is an optional network that displays the values of the first 3 words as soon as the read operation has executed.

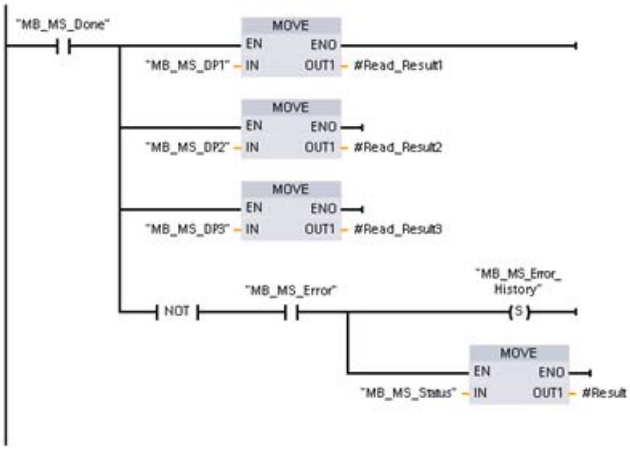

Network 4: Write 64 bits to the process image output, starting at slave address Q2.0.

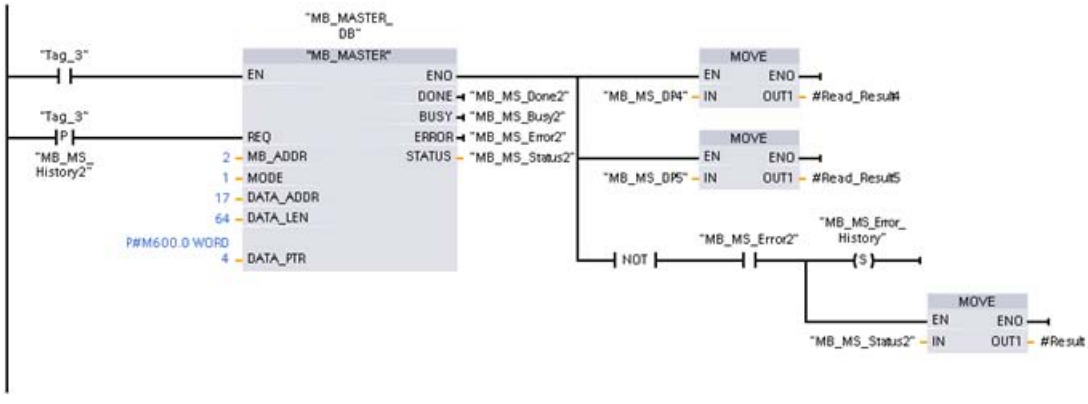

MB_SLAVE: Communicate via the PtP port as Modbus slave (S7-1200)

Description of MB_SLAVE (S7-1200)

Description

The "MB_SLAVE" instruction allows your program to communicate as a Modbus slave using a port on a point-to-point module (PtP) or a communications board (CB). A Modbus RTU master can issue a request and then your program responds via "MB_SLAVE" execution.

You must assign a unique instance data block when you insert the "MB_SLAVE" instruction in your program. This instance data block is used when you specify it at the MB_DB parameter of the "MB_COMM_LOAD (Page 4077)" instruction.

Modbus communication function codes (1, 2, 4, 5, and 15) can read and write bits and words directly in the process image input and process image output in the target system. The following table shows the mapping of Modbus addresses to the process image in the CPU.

Modbus functions of "MB_SLAVE"						S7-1200	
Codes	Function	Data area	Address range			Data area	CPU address
01	Read bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
02	Read bits	Input	10001	to	18192	Process image input	I0.0 to I1023.7
04	Read words	Input	30001	to	30512	Process image input	IW0 to IW1022
05	Write bit	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
15	Write bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7

Modbus communication function codes (function codes 3, 6, 16) use a separate holding register. To do this, you can use bit memory or a data block with the "Standard - compatible with S7-300/400" access type.

You specify the type of the holding register using the MB_HOLD_REG parameter of the MB_SLAVE instruction. The following table shows the mapping of the Modbus holding register to the DB address of MB_HOLD_REG in the target system.

Modbus functions of "MB_SLAVE"				S7-1200	
Codes	Function	Data area	Address range (WORD number)	Address in the DB (BYTE number)	Bit memory address (BYTE number)
03	Read words	Holding register	40001 to 49999 or	DW0 to DW19998 or	MW0 to CPU limit
			400001 to 465535	DW0 to DW131068	
06	Write word	Holding register	40001 to 49999 or	DW0 to DW19998 or	
			400001 to 465535	DW0 to DW131068	
16	Write words	Holding register	40001 to 49999 or	DW0 to DW19998 or	
			400001 to 465535	DW0 to DW131068	

The table below shows the supported Modbus diagnostic functions.

S7-1200 "MB_SLAVE" Modbus diagnostic functions		
Codes	Subfunction	Description
08	0000H	Return query data echo test: The "MB_SLAVE" instruction returns the echo of a received data word to a Modbus master.
08	000AH	Clear communication event counter: The "MB_SLAVE" instruction clears the communication event counter that is used for Modbus function 11.
11	-	Get communication event counter: The "MB_SLAVE" instruction uses an internal communication event counter for recording the number of successful Modbus read and write requests that are sent to the Modbus slave. The counter is not incremented on any Function 8, Function 11, or broadcast requests. It is also not incremented on any requests that result in a communication error (for example, parity or CRC errors).

The "MB_SLAVE" instruction supports broadcast write requests from Modbus masters as long as the requests include access to valid addresses.

Regardless of the validity of a request, "MB_SLAVE" gives no response to a Modbus master as the result of a broadcast request.

Rules of Modbus slave communication

- "MB_COMM_LOAD" must be executed to configure a port, before the "MB_SLAVE" instruction can communicate with this port.
- If a port is to respond as a slave to a Modbus master, then that port cannot be used by "MB_MASTER (Page 4080)". Only one instance of "MB_SLAVE" can be used with a given port.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must control the communication process by polling the "MB_SLAVE" instruction for completed send and receive operations.
- The "MB_SLAVE" instruction must be executed periodically at a rate that allows it to make a timely response to incoming requests from a Modbus master. It is therefore advisable to call the instruction in a cyclic program OB. Calling the "MB_SLAVE" instruction in an interrupt OB is possible but not advisable since it can lead to long delays in execution.

Frequency of execution of "MB_SLAVE"

The "MB_SLAVE" instruction must be executed periodically to receive each request from the Modbus master and to respond as required. The frequency of execution of "MB_SLAVE" is dependent upon the specified response timeout period of the Modbus master. This is illustrated in the following diagram.

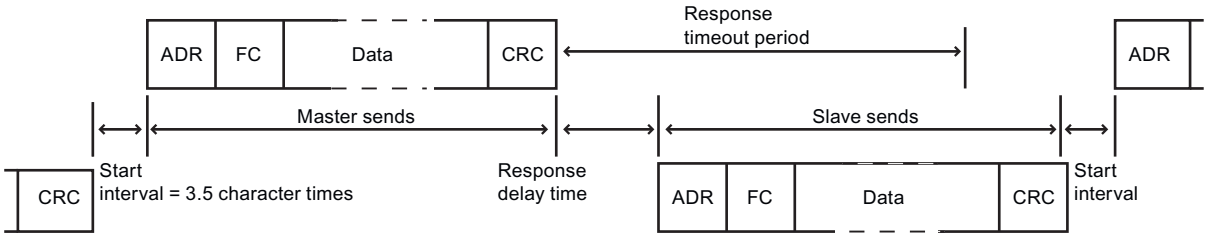

The response timeout period is the amount of time a Modbus master waits for the start of a response from a Modbus slave. This time period is not defined by the Modbus protocol, but rather by a parameter of each Modbus master. The frequency of execution (time between one execution and the next execution) of "MB_SLAVE" must be based on the particular parameters of your Modbus master. As a minimum, you should execute "MB_SLAVE" twice within the response timeout period of the Modbus master.

Parameter

The following table shows the parameters of the "MB_SLAVE" instruction:

Parameter	Declaration	Data type	Memory area	Description
MB_ADDR	Input	V1.0: USINT V2.0: UINT	I, Q, M, D, L or constant	Station address of the Modbus slave (address range: 0 to 255)
MB_HOLD_REG	In_Out	VARIANT	D	Pointer to the Modbus holding register DB. The DB must be created with the "Standard - compatible with S7-300/400" access type.
NDR	Output	BOOL	I, Q, M, D, L	New data ready: • 0: No new data • 1: Indicates that new data has been written by the Modbus master
DR	Output	BOOL	I, Q, M, D, L	Read data: • 0: No data read • 1: Indicates that data has been read by the Modbus master
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error detected • 1: Error, a corresponding error code is output in the STATUS
STATUS	Output	WORD	I, Q, M, D, L	Error code

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter STATUS

STATUS* (W#16#....)	Description
80C8	The specified response timeout (refer to RCVTIME or MSGTIME) is "0".
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission within the wait time. This error is also generated during hardware flow control if the recipient does not detect CTS within the wait time.
80D2	The send request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full
80E1	The message was terminated as a result of a parity error

STATUS* (W#16#....)	Description	
80E2	The message was terminated as a result of a message frame error	
80E3	The message was terminated as a result of a overrun error	
80E4	The message was terminated as a result of the specified length exceeding the total buffer size	
8180	Invalid value for the port ID.	
8186	Invalid Modbus station address	
8187	Invalid pointer to MB_HOLD_REG-DB	
818C	Pointer to a type safe DB type MB_HOLD_REG (must be a Classic DB type)	
Response code sent to Modbus master (B#16#...)		
8380	No response	CRC error
8381	01	Function code not supported or not supported in a broadcast
8382	03	Data length error
8383	02	Error in the data address or address outside the valid range of MB_HOLD_REG
8384	03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 7698)

Instance DB of the "MB_SLAVE" instruction (S7-1200)

Static variables of the instance DB

The following table describes the static variables of the instance DB of the instruction that you can use in the user program. Your program can write values to the HR_Start_Offset and Extended_Addressing variables and control the Modbus slave operations.

The other variables can be read to monitor the Modbus status.

Variable	Data type	Description
HR_Start_Offset	WORD	Start address of the Modbus holding register (default = " 0")
Extended_ Addressing	BOOL	Configuring addressing: • 0: Default address area (1 byte) • 1: Extended address area (2 bytes)
Request_Count	WORD	Total number of queries received by the slave
Slave_Message_Count	WORD	Number of queries sent to this specific slave
Bad_CRC_Count	WORD	Number of received queries with CRC errors
Broadcast_Count	WORD	Number of received broadcast queries

Variable	Data type	Description
Exception_Count	WORD	Number of Modbus-specific errors that require the return of an exception
Success_Count	WORD	The number of requests received for this specific slave without protocol errors

HR_Start_Offset

The addresses of the Modbus holding register start at 40001 or 400001. These addresses correspond to the start address of the holding register in the target system memory. Using the HR_Start_Offset variable, you can set the offset to a different start address.

Example: A holding register starts at MW 100 and has a length of 100 WORD. With an offset of 20 in the HR_Start_Offset parameter, the holding register begins at address 40021 instead of 40001. Each address below 40021 and above 400119 causes an addressing error.

	HR_Start_Offset = 0			HR_Start_Offset = 20	
	Modbus word address	S7-1200 byte address		Modbus word address	S7-1200 byte address
Minimum	40001	MW100		40021	MW100
Maximum	40099	MW198		40119	MW198

Extended_Addressing

To address the Modbus slave, a single byte (default address range) or a double byte (extended address range) can be configured. Extended addressing is used to address more than 247 devices in a single network. If you decide on extended addressing, you can address a maximum of 64,000 addresses. Below, you will see a frame of Modbus function 1 as an example.

Table 4-185 Slave address with one byte (byte 0)

Function 1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	
Request	Slave address	F code	Start address		Length of the coils		
Valid response	Slave address	F code	Length	Coil data			
Error response	Slave address	0x81	E code				

Table 4-186 Slave address with two bytes (byte 0 and byte 1)

Function 1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Request	Slave address		F code	Start address		Length of the coils	
Valid response	Slave address		F code	Length	Coil data		
Error response	Slave address		0x81	F code			

Sample program for a Modbus slave (S7-1200)

Networks (LAD)

Network 1: Initialize parameters of the RS-485 module only once during the first cycle.

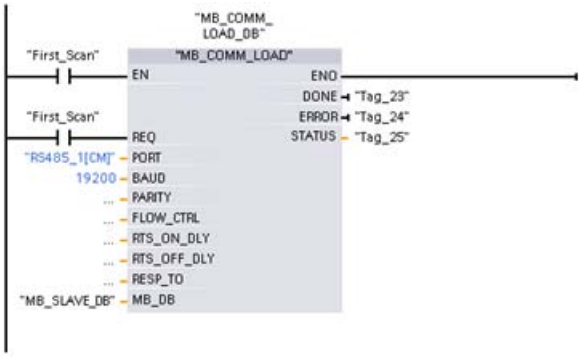

Network 2: Check the requests of the Modbus master in every cycle. 100 words starting at MW1000 are configured for the Modbus holding register.

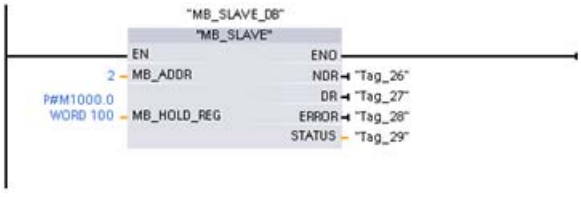

MODBUS (TCP)

MODBUS (TCP) for the library versions V2.1 and V3.x of the S7-1200 CPUs (S7-1200)

MB_CLIENT: Communicating via PROFINET as a Modbus TCP client (S7-1200)

Description of MB_CLIENT (S7-1200)

Description

The "MB_CLIENT" instruction communicates as a Modbus TCP client via the PROFINET connection of the S7-1200 CPU. To use the instruction, you do not require any additional hardware module. With the "MB_CLIENT" instruction, you establish a connection between the client and the server, send requests and receive responses and control connection termination of the Modbus TCP server.

Parameters

The following table shows the parameters of the "MB_CLIENT" instruction:

Parameter	Declaration	Data type	Description
REQ (Page 4097)	Input	BOOL	Communication request with the Modbus TCP server The REQ parameter is level-controlled. This means that as long as the input is set (REQ=true), the instruction sends communication requests. • The instance DB for other clients is blocked with the communications request. • Changes to the input parameters will not become effective until the server has responded or an error message has been output. • If the parameter REQ is set again during an ongoing Modbus request, no additional transmission takes place afterwards.
DISCONNECT (Page 4097)	Input	BOOL	With this parameter, you control the establishment and termination of the connection to the Modbus server: • 0: Establish a communications connection to the specified IP address and port number. • 1: Disconnect the communication connection. No other function is executed during connection termination. The value 7003 is output at the STATUS parameter after successful connection termination. The request is sent immediately if the REQ parameter is set during connection establishment.
CONNECT_ID	Input	UINT	Unique ID to identify the connection. Each instance of the instructions "MB_CLIENT" and "MB_SERVER (Page 4103)" must be assigned a unique connection ID.
IP_OCTET_1	Input	USINT	1st octet of the IP address* of the Modbus TCP server.
IP_OCTET_2	Input	USINT	2nd octet of the IP address* of the Modbus TCP server.

Parameter	Declaration	Data type	Description
IP_OCTET_3	Input	USINT	3rd octet of the IP address* of the Modbus TCP server.
IP_OCTET_4	Input	USINT	4th octet of the IP address* of the Modbus TCP server.
IP_PORT	Input	UINT	IP port number of the server to which the client establishes the connection and communicates using the TCP/IP protocol (default value: 502).
MB_MODE (Page 4097)	Input	USINT	Selects the mode of the request (read, write or diagnostics).
MB_DATA_ADDR (Page 4097)	Input	UDINT	Start address of the data accessed by the "MB_CLIENT" instruction.
DATA_LEN	Input	UINT	Data length: Number of bits or words for the data access (see "MB_MODE and MB_DATA_ADDR parameters" - data length).
MB_DATA_PTR (Page 4099)	InOut	VARIANT	Pointer to the Modbus data register: The register is a buffer for the data received from the Modbus server or to be sent to the Modbus server. The pointer must reference a global data block with standard access. The number of bits addressed must be divisible by 8.
DONE	Out	BOOL	The bit at output parameter DONE is set to "1" as soon as the last job is completed without errors.
BUSY	Out	BOOL	• 0: No "MB_CLIENT " job in progress • 1: "MB_ CLIENT " job in progress
ERROR	Out	BOOL	• 0: No error • 1: Error occurred. The cause of error is indicated by the STATUS parameter.
STATUS (Page 4100)	Out	WORD	Error code of the instruction.
* 8-bit long component of the 32-bit IPv4 IP address of the Modbus TCP server.			

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

Consistent input data during an "MB_CLIENT" call

When a Modbus client calls a Modbus instruction, the status of the input parameters is stored internally and then compared at the next call. The comparison is used to determine whether this particular call initialized the current request. Several "MB_CLIENT" calls can be executed if you use a common instance DB. The values of the input parameters must not be changed while an "MB_CLIENT" instance is being executed. If the input parameters are changed during execution, "MB_CLIENT" cannot be used to check whether or not the instance is currently being executed.

Multiple client connections

A Modbus TCP client can support several TCP connections and the maximum number of connections depends on the CPU being used. The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections. Modbus TCP connections can also be shared by client and/or server connections.

With individual client connections, remember the following rules:

- Each "MB_CLIENT" connection must use a unique instance DB.
- For each "MB_CLIENT" connection, a unique server IP address must be specified.
- Each "MB_CLIENT" connection requires a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- Unique numbers for the IP port may or may not be required depending on the server configuration.

Static tags of the instruction

The following table describes the editable static tags of the instance data block of the "MB_CLIENT" instruction.

Tag	Data type	Start value	Description
Blocked_Proc_Time-out	REAL	3.0	Wait time in seconds before the static tag ACTIVE is reset if there is a blocked Modbus instance. This can, for example, occur if a client request is output and the execution of the client function aborts before the request was fully executed. The maximum wait time is 55 seconds.
MB_Transaction_ID	WORD	1	Transaction ID of the Modbus TCP protocol. The start value of "1" should only be changed if the Modbus TCP server requires a different value.
MB_Unit_ID	BYTE	255	Unit Identifier Modbus device detection: A Modbus TCP server is addressed using its IP address. For this reason, the MB_UNIT_ID parameter is not used in the case of Modbus TCP addressing. The MB_UNIT_ID parameter corresponds to the field of the slave address in the Modbus RTU protocol. If a Modbus TCP server is used as a gateway for a Modbus RTU protocol, the slave device in the serial network can be identified using MB_UNIT_ID. The MB_UNIT_ID parameter would in this case forward the request to the correct Modbus RTU slave address. Please note that some Modbus TCP devices may require the MB_UNIT_ID parameter for the initialization within a limited value range.
RCV_TIMEOUT	REAL	2.0	Time interval in seconds in which the "MB_CLIENT" instruction waits for a response from the server.
Connected	BOOL	0	Indicates whether the connection to the assigned client has been established or not: 1 = connected, 0 = not connected.

See also

MB_CLIENT example 1: Send several requests via a TCP connection (Page 4109)

MB_CLIENT example 2: Send multiple requests via several TCP connections (Page 4110)

MB_CLIENT example 3: Coordinate several requests (Page 4111)

Difference between synchronous and asynchronous instructions (Page 596)

REQ and DISCONNECT parameters (S7-1200)

Description

If no instance of the "MB_CLIENT" instruction is executing and if the value of the DISCONNECT parameter is "0", a new job executes if REQ=1. If there is not yet a connection, this is established during execution.

If the same instance of the "MB_CLIENT" instruction executes again (DISCONNECT = 0 and REQ = 1), before the active job was executed, this is not executed on completion of the active job. A new job can only be started on completion of the active job (REQ = 1).

You can monitor the status of execution with the DONE parameter. You can use this to monitor the status of execution if the "MB_CLIENT" instruction is executed sequentially.

See also

Description of MB_CLIENT (Page 4094)

MB_MODE, MB_DATA_ADDR and DATA_LEN parameters (S7-1200)

Description

The combination of the MB_MODE, MB_DATA_ADDR and DATA_LEN parameters defines the Modbus function code used in the current Modbus message for the values 0, 1 and 2 of MB_MODE:

- MB_MODE contains the information on whether to read or to write.
  MB_MODE=0: Read, MB_MODE=1 and 2: Writing
- MB_DATA_ADDR contains the information on what is to be read or written, as well as address information from which the "MB_CLIENT" instruction calculates the remote address.
- DATA_LEN contains the number of values to be read/written.

Examples:

- The combination MB_MODE=1, MB_DATA_ADDR=1, DATA_LEN=1 specifies the function code 05. 1 output bit is written starting from the remote address 0.
- The combination MB_MODE=1, MB_DATA_ADDR=1, DATA_LEN=2 specifies the function code 15. 2 output bits are written starting from the remote address 0.

The following table shows the relationship between the input parameters MB_MODE, MB_DATA_ADDR, DATA_LEN of the "MB_CLIENT" instruction and the associated Modbus function.

MB_MODE	MB_DATA_ADDR	DATA_LEN	Modbus function	Function and data type
0	1 to 9,999	1 to 2,000	01	Read 1 to 2,000 output bits on the remote address 0 to 9,998
0	10,001 to 19,999	1 to 2,000	02	Read 1 to 2,000 input bits on the remote address 0 to 9,998
0	40,001 to 49,999	1 to 125	03	Read 1 to 125 holding registers on the remote address 0 to 9,998
0	30,001 to 39,999	1 to 125	04	Read 1 to 125 input words on the remote address 0 to 9,998
1	1 to 9,999	1	05	Write 1 output bit on the remote address 0 to 9,998
1	40,001 to 49,999	1	06	Write 1 holding register on the remote address 0 to 9,998
1	1 to 9,999	2 to 1,968	15	Write 2 to 1,968 output bits on the remote address 0 to 9,998
1	40,001 to 49,999	2 to 123	16	Write 2 to 123 holding registers on the remote address 0 to 9,998
2	1 to 9,999	1 to 1,968	15	Write 1 to 1,968 output bits on the remote address 0 to 9,998
2	40,001 to 49,999	1 to 123	16	Write 1 to 123 holding registers on the remote address 0 to 9,998
11	The MB_DATA_ADDR and DATA_LEN parameters are not evaluated when this function is executed.		11	Read status word and event counter of the server: • The status word reflects the the processing status (0 - not processing, 0xFFFF - processing). • The event counter is incremented when the Modbus request was executed successfully. If an error occurred during execution of a Modbus function, a message is sent by the server but the event counter is not incremented.
80	-	1	08	Check the server status with the diagnostic code 0x0000 (return loop test - the server sends the request back): • 1 WORD per call
81	-	1	08	Reset the event counter of the server with the diagnostic code 0x000A: • 1 WORD per call
3 to 10, 12 to 79, 82 to 255				Reserved

Note

Unit Identifier for RTU devices over MB TCP

You can find the Unit Identifier for RTU devices over Modbus TCP as "MB_Unit_ID" tag in the editable static tags of the instance data block of the instruction "MB_CLIENT". The Unit Identifier is used for unique assignment of the coupling partners.

See also

Description of MB_CLIENT (Page 4094)

MB_DATA_PTR parameter (S7-1200)

Description

The MB_DATA_PTR parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

For a buffer in the memory area (M), use a pointer in the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

The MB_DATA_PTR parameter uses a communications buffer:

- For the communication functions of the "MB_CLIENT" instruction:
  - Reading and writing of 1 bit of data of the Modbus server addresses 00001 to 09999 and 10001 to 19999.
  - Reading of 16-bit WORD data of the Modbus server addresses 30001 to 39999 and 40001 to 49999.
  - Writing 16-bit WORD data of the Modbus server addresses 40001 to 49999.
- During data transmission (length: bit or WORD) from or to the global DB or the memory area (M) that you assigned with the MB_DATA_PTR parameter.

If you use a data block for the buffer in the MB_DATA_PTR parameter, you will need to assign data types to the DB elements.

- Use the 1-bit data type BOOL for a Modbus bit address
- Use a 16-bit data type such as WORD, UINT, INT or REAL for a Modbus WORD address.
- Use a 32-bit data type (double word) such as DWORD, DINT or REAL for two Modbus WORD addresses.
- With MB_DATA_PTR, you can also access complex DB elements such as:
  - Standard arrays
  - Structures with unique element names
  - Complex structures with unique naming of the elements and data type lengths of 16 or 32 bits.
- The data areas for the MB_DATA_PTR parameter can also be in different global data blocks (or in different memory areas). You can, for example, use a data block for the the read jobs and another one for the write jobs or a separate data block for each "MB_CLIENT" station.

See also

Description of MB_CLIENT (Page 4094)

Parameter STATUS (S7-1200)

Parameter STATUS (general status information) (library version V2.1)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
7000	No call active (REQ=0).
7001	First call with REQ=1: Processing started; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). Processing already active; BUSY has the value 1.
7003	Connection is being terminated.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Parameter STATUS (general status information) (library version V3.x)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active (REQ=0).
7001	First call with REQ=1: Processing started; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). Processing already active; BUSY has the value 1.
7003	Terminating connection.
7004	Connection established and monitored. No job processing active.
7005	Data was sent.
7006	Data was received.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Parameter STATUS (protocol error)

STATUS* (W#16#)	Local and/ or remote errors	Error code in the answer from MB_SERVER (B#16#)	Description
80C8	Local	-	No response of the server in the defined period. Check the connection to the Modbus server. This error is only reported on completion of the configured repeated attempts. If the "MB_CLIENT" instruction does not receive an answer with the originally transferred transaction ID (MB_TRANSACTION_ID tag) within the defined period, this error code is output.
8380	Local	-	Received Modbus frame has incorrect format or too few bytes were received.

STATUS* (W#16#)	Local and/ or remote errors	Error code in the answer from MB_SERVER (B#16#)	Description
8381	Remote	01	Function code is not supported.
8382	Local	-	• The length of the Modbus frame in the frame header does not match the number of received bytes. • The number of bytes does not match the number of actually transmitted bytes (only functions 1-4). For example, this is the case when "MB_CLIENT" requests an odd number of words, but "MB_SERVER" always sends an even number of words. • The start address in the received frame does not match the saved start address (functions 5, 6, 15, 16). • The number of words does not match the number of actually transmitted words (functions 15 and 16).
	Remote	03	Invalid length specification in received Modbus frame. Check the server side.
8383	Local	-	Error reading or writing data or access outside the address area of MB_DATA_PTR (Page 4099).
	Remote	02	Error reading or writing data or access outside the address area of the server
8384	Local	-	• Invalid exception code received. • A different data value was received than was originally sent by the client (functions 5, 6 and 8). • Invalid status value received (function 11)
	Remote	03	Error in data value for function 5
8385	Local	-	• Diagnostics code not supported. • A different subfunction code was received than was originally sent by the client (function 8).
	Remote	03	Diagnostics code not supported
8386	Local	-	Received function code does not match the one sent originally.
8387	Local	-	• The assigned connection ID is different from that used for previous requests. Only one connection ID can be used for each instance DB of the "MB_CLIENT" instruction. • The error code is also output when the ID of the Modbus TCP protocol received by the server is not "0".
8388	Local	-	The Modbus server sent a different data length than was requested. This error occurs only when using the Modbus functions 15 or 16.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".			

Parameter STATUS (parameter error)

In addition to the errors listed in the following table, errors are also possible with the "MB_CLIENT" instruction caused by the communications instructions ("TCON", "TDISCON", "TSEND" and "TRCV").

STATUS* (W#16#)	Description
80BB	Invalid value at ACTIVE_EST parameter (identifier for the type of connection establishment, see T_CON_PARAM): • Only passive connection establishment permitted for server (ACTIVE_EST = FALSE). • Only active connection establishment permitted for client (ACTIVE_EST = TRUE).
8188	The MB_MODE parameter has an invalid value.
8189	Invalid addressing of data at the MB_DATA_ADDR parameter.
818A	Invalid data length at the MB_DATA_LEN parameter.
818B	The MB_DATA_PTR parameter has an invalid pointer. You should also check the values of the MB_DATA_ADDR (Page 4097) and MB_DATA_LEN parameters.
818C	• The MB_DATA_PTR (Page 4094) pointer references an an optimized data block. Either use a data block with standard access or a memory area. • Timeout at parameter BLOCKED_PROC_TIMEOUT (see static tags of instruction). The limit of 55 seconds has been exceeded.
8200	• A further Modbus request is currently being processed via the port. • Another instance of MB_CLIENT with the same connection parameters is processing an existing Modbus request.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Note

Error codes of internally used communications instructions.

With the "MB_CLIENT" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND" and "TRCV") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_CLIENT" instruction. The error codes are displayed for the respective instruction under STATUS in the Static section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

See also

MB_HOLD_REG parameter (Page 4106)

Switching display formats in the program status (Page 7698)

MB_SERVER: Communicating via PROFINET as a Modbus TCP server (S7-1200)

Description of MB_SERVER (S7-1200)

Description

The "MB_SERVER" instruction communicates as a Modbus TCP server via the PROFINET connection of the S7-1200 CPU. To use the instruction, you do not require any additional hardware module. The "MB_SERVER" instruction processes connection requests of a Modbus TCP client, receives requests from Modbus functions and sends responses.

NOTICE
Security information Note that each client of the network is given read and write access to the process image inputs and outputs and to the data block or bit memory area defined by the Modbus holding register. The option is available to restrict access to an IP address to prevent unauthorized read and write operations. Note, however, that the shared address can also be used for unauthorized access.

Parameters

The following table shows the parameters of the "MB_SERVER" instruction:

Parameter	Declaration	Data type	Description
DISCONNECT	Input	BOOL	The instruction "MB_SERVER" enters into a passive connection with a partner module, which means the server responds to a TCP connection request from each requesting IP address. You can use this parameter to control when a connection request is accepted: • 0: A passive connection is established when there is no communications connection. • 1: Initialization of the connection termination. If the input is set, no other operations are executed. The value 7003 is output at the STATUS parameter after successful connection termination.
CONNECT_ID	Input	UINT	The parameter uniquely identifies a connection within the CPU. Each individual instance of the instructions "MB_CLIENT (Page 4094)" and "MB_SERVER" must have a unique CONNECT_ID parameter.
IP_PORT	Input	UINT	Start value=502. The number of the IP port defines which IP port is monitored for connection requests of the Modbus client. The following TCP port numbers must not be used for the passive connection of the "MB_SERVER" instruction: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.
MB_HOLD_REG (Page 4106)	InOut	VARIANT	Pointer to the Modbus holding register of the "MB_SERVER" instruction: Use a global data block with standard access as holding register. The holding register contains the values that may be accessed by a Modbus client using the Modbus functions 3 (read), 6 (write) and 16 (read).

Parameter	Declaration	Data type	Description
NDR	Output	BOOL	"New Data Ready": • 0: No new data • 1: New data written by the Modbus client written
DR	Output	BOOL	"Data Read": • 0: No data read • 1: Data read by the Modbus client
ERROR	Output	BOOL	If an error occurs during the call of the "MB_SERVER" instruction, the output of the ERROR parameter is set to TRUE. Detailed information about the cause of the problem is indicated by the STATUS parameter.
STATUS (Page 4107)	Output	WORD	Error code of the instruction.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Mapping of Modbus addresses to the process image

The "MB_SERVER" instruction allows incoming Modbus functions (1, 2, 4, 5 and 15) direct read and write access to the process image input and output of the S7-1200 CPU (use of the data types BOOL and WORD).

For the data transfer of the function codes 3, 6 and 16, the holding register (MB_HOLD_REG parameter) must be defined longer than one byte. The following table shows the mapping of the Modbus addresses to the process image of the CPU.

Modbus function						S7-1200	
Code	Function	Data area	Address space			Data area	CPU address
01	Read: Bits	Output	0	to	8191	Process image output	Q0.0 to Q1023.7
02	Read: Bits	Input	0	to	8191	Process image input	I0.0 to I1023.7
04	Read: WORD	Input	0	to	1021	Process image input	IW0 to IW1022
05	Write: Bits	Output	0	to	8191	Process image output	Q0.0 to Q1023.7
15	Write: Bits	Output	0	to	8191	Process image output	Q0.0 to Q1023.7

Incoming Modbus alarms with the function codes 3, 6 and 16 write to or read from the Modbus holding registers (you specify the holding registers with the MB_HOLD_REG parameter).

Multiple server connections

You can create multiple Server connections. This allows a single CPU to establish connections to more than one Modbus TCP client at the same time.

A Modbus TCP server can support several TCP connections and the maximum number of connections depends on the CPU being used.

The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections.

Modbus TCP connections can also be shared by client and/or server connections.

In the case of Server connections, remember the following rules:

- Each "MB_SERVER" connection must use a unique instance DB.
- Each "MB_SERVER" connection must be created with a unique IP port number. Only one connection is supported for each port.
- Each "MB_SERVER" connection must use a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- For each connection, the "MB_SERVER" instruction must be called individually.

Modbus diagnostics functions

The table below contains a description of the Modbus diagnostics functions.

Code	Subfunction	Description
08	0x0000	Echo test: The "MB_SERVER" instruction receives a data word and returns this unchanged to the Modbus master.
08	0x000A	Reset event counter: The "MB_SERVER" instruction resets the event counter for communication that is used for Modbus function 11.
11	-	Fetch event counter of the communication: The "MB_SERVER" instruction uses an internal event counter for communication to record the number of successfully executed read and write requests sent to the Modbus server. The event counter is not incremented by the functions 8, 11 or broadcast requests. The same applies to requests that result in a communications error (for example parity errors or CRC errors). The broadcast function is not available for Modbus TCP because only one client/server connection can exist at any one time.

Static tags of the instruction

The following table describes the static tags of the instance data block of the "MB_SERVER" instruction used in the program. You can write the HR_Start_Offset tag. You can read the other tags to monitor the Modbus status.

Tag	Data type	Start value	Description
HR_Start_Offset	WORD	0	Assign the start address of the Modbus holding register.
Request_Count	WORD	0	Total number of requests received by the server.
Server_Message_Count	WORD	0	Total number of received alarms for the relevant server.
Xmt_Rcv_Count	WORD	0	Counter for detecting the number of transfers during which an error occurred. The counter is also incremented if an invalid Modbus alarm is received.
Exception_Count	WORD	0	Counter for detecting the number of errors specifically for Modbus cause an exception.
Success_Count	WORD	0	Counter for detecting the number of requests that contain no error in the transferred protocol.
Connected	BOOL	0	Indicates whether the connection to the assigned client has been established or not: 1 = connected, 0 = not connected.

Example: Addressing via static tag HR_Start_Offset

The addresses of the Modbus holding register start at 0 (from the perspective of the MB_CLIENT at 40.001). These addresses correspond to the address area of the CPU memory area for the holding register. You can also define the HR_Start_Offset tag so that the Modbus holding register has a start address other than 0.

Example: The holding register begins at MW100, and has a length of 100 WORD. An offset value in the HR_Start_Offset parameter means that the start address of the holding register is moved from 0 to 20. This causes an error whenever the holding register is addressed below the address 20 and above the address 119.

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (WORD)	0	99
	CPU address	MW100	MW298
20	Modbus address (WORD)	20	119
	CPU address	MW100	MW298

See also

MB_SERVER example: Multiple TCP connections (Page 4112)

Difference between synchronous and asynchronous instructions (Page 596)

MB_HOLD_REG parameter (S7-1200)

Description

The MB_HOLD_REG parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

- The high limit for the number of addresses in the data block (D) is determined by the maximum work memory of the CPU.
- The high limit for the number of bit memories (M) is determined by the size of the CPU memory area.

The following table shows examples of mapping Modbus addresses to the holding register for the Modbus functions 3 (read WORD), 6 (write WORD), 16 (write several WORD) and 23 (write and read several words).

Modbus addresses	MB_HOLD_REG parameter - examples		
0	MW100	DB10.DBW0	"Recipe".ingredient[1]
1	MW102	DB10.DBW2	"Recipe".ingredient[2]
2	MW104	DB10.DBW4	"Recipe".ingredient[3]
3	MW106	DB10.DBW6	"Recipe".ingredient[4]
4	MW108	DB10.DBW8	"Recipe".ingredient[5]

See also

Description of MB_SERVER (Page 4103)

Parameter STATUS (S7-1200)

Parameter STATUS (general status information) (library version V2.1)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
7000	No call active (REQ=0).
7001	First call with REQ=1: Processing started; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). Processing already active; BUSY has the value 1.
7003	Connection is being terminated.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Parameter STATUS (general status information) (library version V3.x)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active (REQ=0).
7001	First call. Connection establishment triggered.
7002	Intermediate call. Establishing connection.
7003	Terminating connection.
7005	Data is being sent.
7006	Data is being received.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Parameter STATUS (protocol error)

STATUS* (W#16#)	Error code in the answer from MB_SERVER (B#16#)	Description
8380	-	Received Modbus frame has incorrect format or too few bytes were received.
8381	01	Function code is not supported.
8382	03	Error in data length

STATUS* (W#16#)	Error code in the answer from MB_SERVER (B#16#)	Description
8383	02	Error in the data address or access outside the address area of MB_HOLD_REG (Page 4106).
8384	03	Error in the data value (function 5)
8385	03	Diagnostics code not supported
* The status codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Parameter STATUS (parameter error)

STATUS* (W#16#)	Description
80BB	Invalid value at ACTIVE_EST parameter (identifier for the type of connection establishment, see T_CON_PARAM): • Only passive connection establishment permitted for server (ACTIVE_EST = FALSE). • Only active connection establishment permitted for client (ACTIVE_EST = TRUE).
8187	The MB_HOLD_REG parameter has an invalid pointer. Data area is too small.
818C	• The pointer at parameter MB_HOLD_REG references an optimized data block. Either use a data block with standard access or a memory area. • Error due to timeout of execution (more than 55 seconds).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Note

Error codes of internally used communications instructions.

With the "MB_SERVER" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND" and "TRCV") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_SERVER" instruction. The error codes are displayed for the respective instruction under STATUS in the Static section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

See also

Description of MB_SERVER (Page 4103)

Switching display formats in the program status (Page 7698)

Examples (S7-1200)

MB_CLIENT example 1: Send several requests via a TCP connection (S7-1200)

Description

Several requests of the Modbus Client can be sent via a TCP connection. To do this, use the same instance DB, the same connection ID and the same port number.

Only one client can be active at any one time. After processing of a client has been completed, the next client is processed. The order of execution must be defined in the program.

In the following sample program, the value of the STATUS output parameter is also copied.

Network 1: Modbus function 1 - 16 read output bits

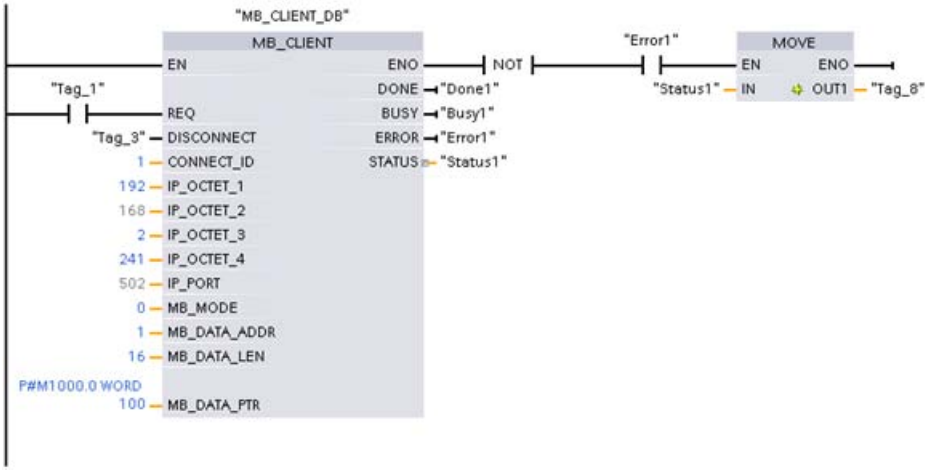

Network 2: Modbus function 2 - 32 read input bits

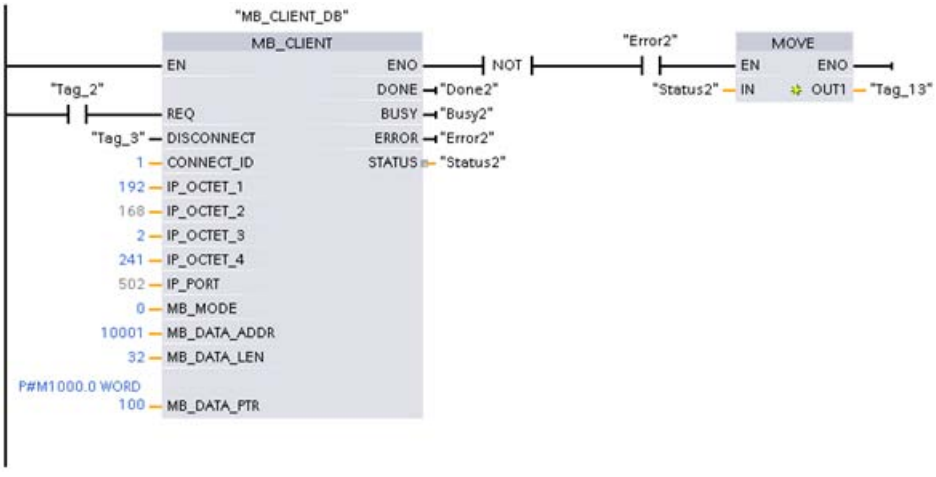

MB_CLIENT example 2: Send multiple requests via several TCP connections (S7-1200)

Description

Requests from the Modbus client can be sent via different TCP connections. If you require this, use a different instance DB and a different connection ID.

Use a different port number, if the connections are to the same Modbus server. If the connections are to different Modbus servers, you can freely assign the port number.

Network 1: Modbus function 4 - read input (WORD)

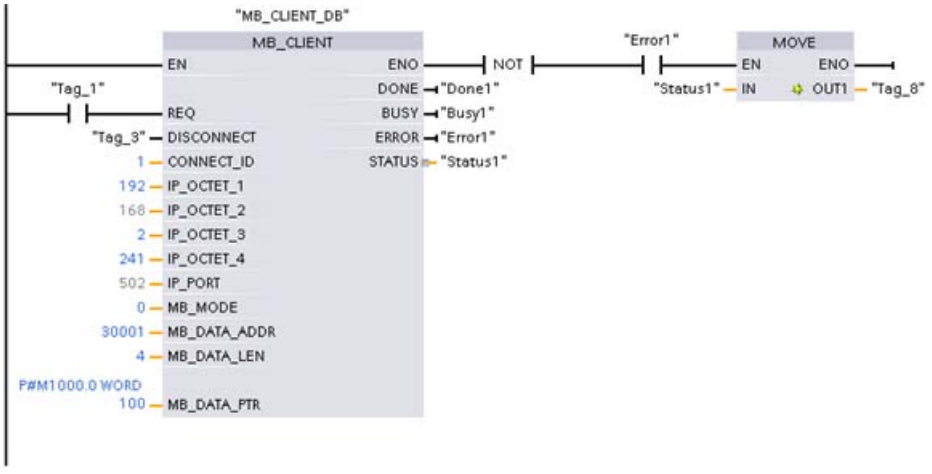

Network 2: Modbus function 3 - read holding register (WORD)

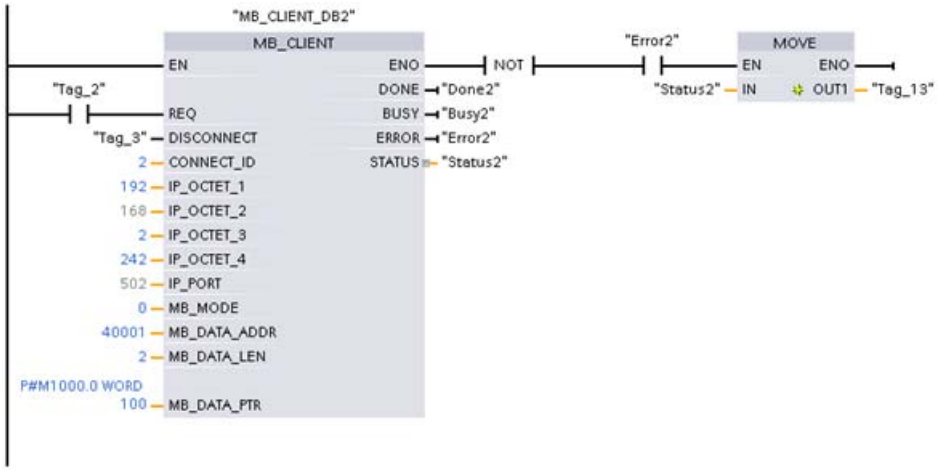

MB_CLIENT example 3: Coordinate several requests (S7-1200)

Description

Make sure that the individual Modbus requests are executed. You control coordination of requests with the program. The following example demonstrates how the output parameters of the first and second client request can be used to coordinate execution of the instructions.

Network 1: Modbus function 3 - read holding register (WORD)

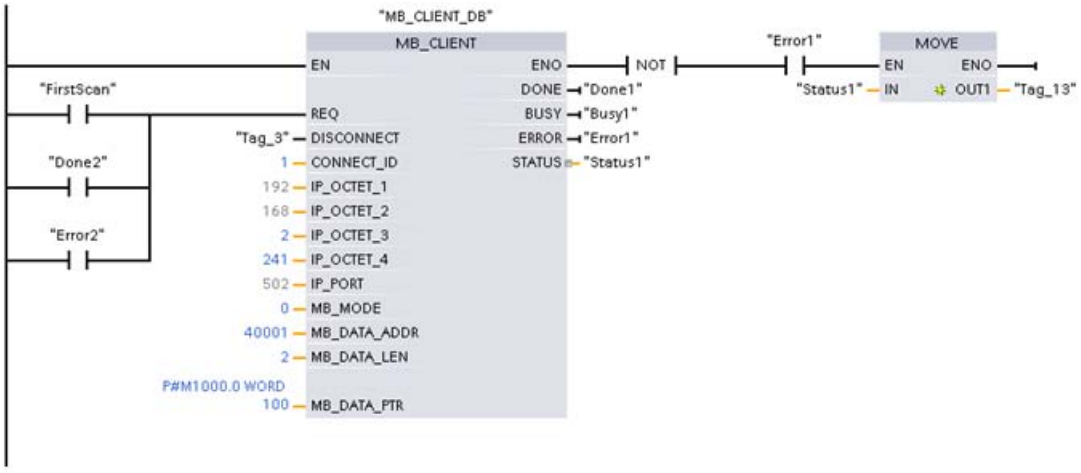

Network 2: Modbus function 3 - read holding register (WORD)

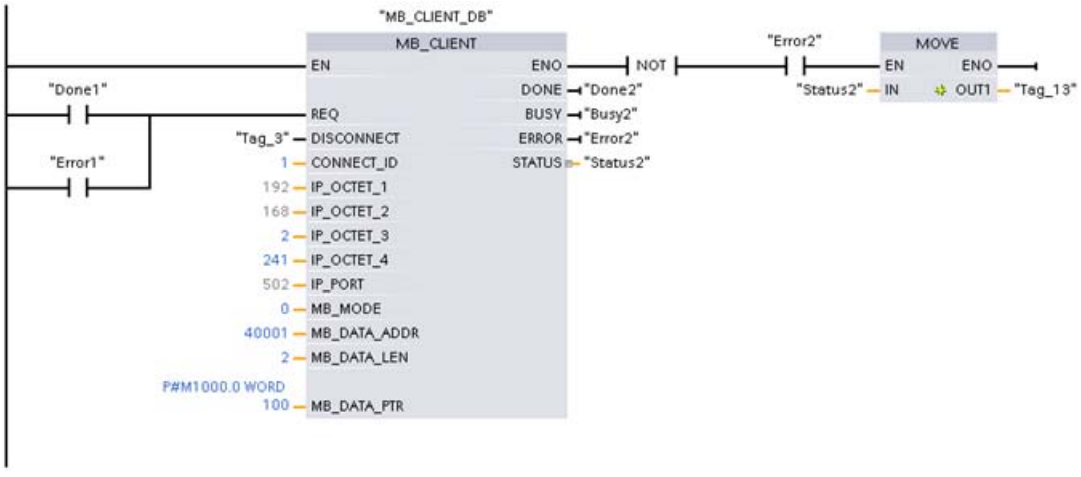

MB_SERVER example: Multiple TCP connections (S7-1200)

Description

You can use several Modbus TCP server connections. To do this, the "MB_SERVER" instruction must be called independently for each connection.

Every connection requires the following:

- An independent instance data block of the instruction
- A unique connection ID
- A separate IP port (on the S7-1200, only one connection is permitted per IP port)

To optimize the performance "MB_SERVER" should be executed once per program cycle for each connection.

Network 1: Connection #1 with associated IP port connection ID and instance DB

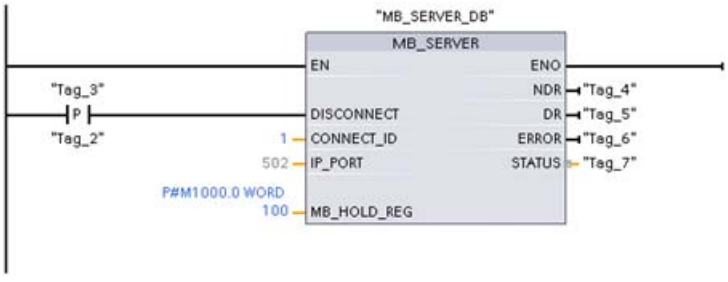

Network 2: Connection #1 with associated IP port connection ID and instance DB

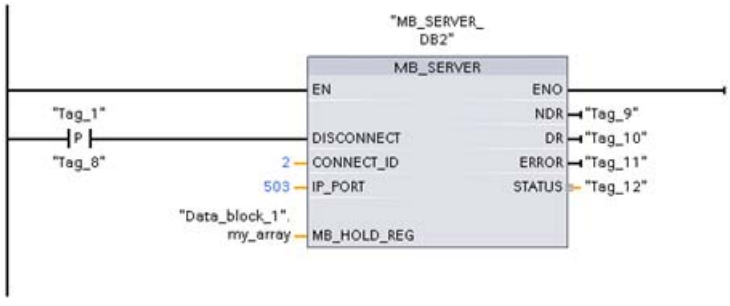

MODBUS (TCP) for the library versions V4.0 and later of the S7-1200 CPUs and as of V3.x of the S7-1500 CPUs (S7-1200, S7-1500)

MB_CLIENT: Communicating via PROFINET as a Modbus TCP client (S7-1200, S7-1500)

Description of MB_CLIENT (S7-1200, S7-1500)

Description

The "MB_CLIENT" instruction communicates as a Modbus TCP client via the PROFINET connection. With the "MB_CLIENT" instruction, you establish a connection between the client and the server, send Modbus requests and receive responses and control connection termination of the Modbus TCP client.

For the S7-1200 with firmware version V4.0 you can use the "MB_CLIENT" instruction up to and including library version V3.1. With the S7-1200 as of firmware version V4.1 and the S7-1500, you can use the "MB_CLIENT" instruction of all library versions.

The connection can take place via the local interface of the CPU or CM/CP.

To use the instruction, you do not require an additional hardware module.

Multiple client connections

A Modbus TCP client can support several TCP connections and the maximum number of connections depends on the CPU being used. The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections. Modbus TCP connections can also be shared by "MB_CLIENT" and/or "MB_SERVER" instances.

With individual client connections, remember the following rules:

- Each "MB_CLIENT" connection must use a unique instance DB.
- For each "MB_CLIENT" connection, a unique server IP address must be specified.
- Each "MB_CLIENT" connection requires a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- Unique numbers for the IP port may or may not be required depending on the server configuration.

Parameters

The following table shows the parameters of the "MB_CLIENT" instruction:

Parameter	Declaration	Data type	Description
REQ (Page 4117)	Input	BOOL	Modbus query to the Modbus TCP server The REQ parameter is level-controlled. This means that as long as the input is set (REQ=true), the instruction sends communication requests. • After the Modbus query has started, the instance DB is locked for other clients. • Changes to the input parameters will not become effective until the server has responded or an error message has been output. • If the parameter REQ is set again during an ongoing Modbus request, no additional transmission takes place afterwards.
DISCONNECT (Page 4117)	Input	BOOL	With this parameter, you control the establishment and termination of the connection to the Modbus server: • 0: Establish communications connection to the connection partner configured at the CONNECT parameter (see CONNECT parameter). • 1: Disconnect the communication connection. No other function is executed during connection termination. The value 0003 is output at the STATUS parameter after successful connection termination. The Modbus request is sent immediately if the REQ parameter is set during connection establishment.
MB_MODE (Page 4118)	Input	USINT	Selects the mode of the Modbus request (read, write or diagnostics) or direct selection of a Modbus function
MB_DATA_ADDR (Page 4118)	Input	UDINT	depends on MB_MODE
MB_DATA_LEN	Input	UINT	Data length: Number of bits or words for the data access (see MB_MODE, MB_DATA_ADDR and MB_DATA_LEN parameters (Page 4118)).
MB_DATA_PTR (Page 4120)	InOut	VARIANT	Pointer to a data buffer for the data to be received from the Modbus server or to be sent to the Modbus server.
CONNECT (Page 4121)	InOut	VARIANT	Pointer to the structure of the connection description The following structures (system data types) can be used: • TCON_IP_v4: Includes all address parameters that are required for establishing a programmed connection. When using TCON_IP_v4, the connection is established when calling the "MB_CLIENT" instruction. • TCON_Configured: Includes the address parameters of a configured connection. When using TCON_Configured, an existing connection is used that was created by the CPU after download of the hardware configuration.
DONE	Out	BOOL	The bit at output parameter DONE is set to "1" as soon as the last Modbus job is completed without errors.

Parameter	Declaration	Data type	Description
BUSY	Out	BOOL	• 0: No Modbus request in progress • 1: Modbus request being processed The output parameter BUSY is not set during connection establishment and termination.
ERROR	Out	BOOL	• 0: No error • 1: Error occurred. The cause of error is indicated by the STATUS parameter.
STATUS (Page 4123)	Out	WORD	Detailed status information of the instruction.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

Consistent input data during an "MB_CLIENT" call

When a Modbus client instruction is called, the values of the input parameters are stored internally and then compared at the next call. The comparison is used to determine whether this particular call initialized the current request. Several "MB_CLIENT" calls can be executed if you use a common instance DB. The values of the input parameters must not be changed while an "MB_CLIENT" instance is being executed. If the input parameters are changed during execution, "MB_CLIENT" cannot be used to check whether or not the instance is currently being executed.

Static tags of the instruction

The following table describes the editable static tags of the instance data block of the "MB_CLIENT" instruction.

Tag	Data type	Start value	Description
Blocked_Proc_Timeout	REAL	3.0	Wait time in seconds before the static tag ACTIVE is reset if there is a blocked Modbus instance. This can, for example, occur if a client request is output and the execution of the client function aborts before the request was fully executed. The wait time must be between 0.5 s and 55 s.
MB_Transaction_ID	WORD	1	Transaction ID of the Modbus TCP protocol. The start value of "1" should only be changed if the Modbus TCP server requires a different value.

Tag	Data type	Start value	Description
MB_Unit_ID	BYTE	255	Unit Identifier Modbus device detection: A Modbus TCP server is addressed using its IP address. For this reason, the MB_UNIT_ID parameter is not used in the case of Modbus TCP addressing. The MB_UNIT_ID parameter corresponds to the field of the slave address in the Modbus RTU protocol. If a Modbus TCP server is used as a gateway for a Modbus RTU protocol, the slave device in the serial network can be identified using MB_UNIT_ID. The MB_UNIT_ID parameter would in this case forward the request to the correct Modbus RTU slave address. Please note that some Modbus TCP devices may require the MB_UNIT_ID parameter for the initialization within a limited value range. Use the following links to get more information on the MB_Unit_ID parameter: https://support.industry.siemens.com/cs/ww/en/view/102420337 (https://support.industry.siemens.com/cs/document/102420337/what-is-the-function-of-the-mb_unit_id-in-the-instance-data-block-of-the-modbus-block-mb_client?dti=0&dl=en&lc=de-WW) http://support.automation.siemens.com/WW/view/en/109736516 (https://support.industry.siemens.com/cs/document/109736516/how-do-you-clear-the-status-168382-in-the-case-of-a-modbus-tcp-connection-between-simatic-s7-1500-s7-1200-and-sentron-pac-devices?dti=0&dl=en&lc=de-WW)
RCV_TIMEOUT	REAL	2.0	Time interval in seconds in which the "MB_CLIENT" instruction waits for a response from the server. It must be between 0.5 s and 55 s.
Connected	BOOL	0	Indicates whether the connection to the assigned server has been established or not: 1 = connected, 0 = not connected.
RETRIES	WORD	0	Number of send attempts made by the "MB_CLIENT" instruction before it returns the error W#16#80C8.

Note

Tag MB_Transaction_ID

If the transaction ID in the answer of the Modbus TCP server does not match the transaction ID of the job from "MB_CLIENT", the "MB_CLIENT" instruction waits for the time period RCV_TIMEOUT * RETRIES for the answer of the Modbus TCP server with the correct transaction ID; once this time has expired, it returns the error W#16#80C8.

Example

An example project for the Modbus TCP communication between two S7-1500 CPUs can be found in the Service and Support Portal under Entry ID 94766380 (http://support.automation.siemens.com/WW/view/en/94766380).

Two Modbus functions are used in this example. For each Modbus function, a Modbus TCP connection is established using a Modbus block pair (MB_CLIENT and MB_SERVER).

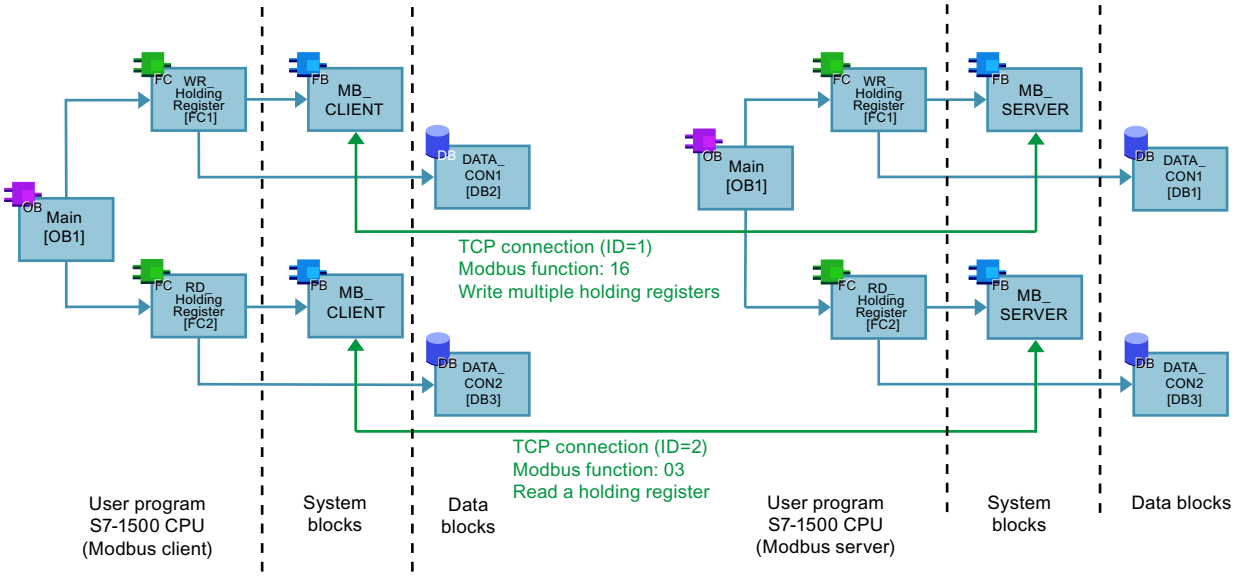

See also

Difference between synchronous and asynchronous instructions (Page 596)

REQ and DISCONNECT parameters (S7-1200, S7-1500)

Description

If no job is active and if the value of the DISCONNECT parameter is "0", a new job executes if REQ=1. If there is not yet a connection, this is established during execution.

If the same instance of the "MB_CLIENT" instruction executes again (DISCONNECT = 0 and REQ = 1), before the active job was executed, this is not executed on completion of the active job. A new job can only be started on completion of the active job (REQ = 1).

The status of the execution is output by the output parameters. You can use it to monitor the execution status when the "MB_CLIENT" instruction is executed sequentially.

MB_MODE, MB_DATA_ADDR and MB_DATA_LEN parameters (S7-1200, S7-1500)

Description

The combination of the MB_MODE, MB_DATA_ADDR and MB_DATA_LEN parameters defines the Modbus function code used in the current Modbus message for the values 0, 1 and 2 of MB_MODE:

- MB_MODE contains the information on whether to read or to write.
  MB_MODE=0: Read, MB_MODE=1 and 2: Write (Note: With MB_MODE=2, there is not distinction between Modbus functions 15 and 05 or between Modbus functions 16 and 06.)
- MB_DATA_ADDR contains the information on what is to be read or written, as well as address information from which the "MB_CLIENT" instruction calculates the remote address.
- MB_DATA_LEN contains the number of values to be read/written.

Examples:

- The combination MB_MODE=1, MB_DATA_ADDR=1, MB_DATA_LEN=1 specifies the function code 05. 1 output bit is written starting from the remote address 0.
- The combination MB_MODE=1, MB_DATA_ADDR=1, MB_DATA_LEN=2 specifies the function code 15. 2 output bits are written starting from the remote address 0.

The following applies to the values 101 to 106 and 115 to 116 from MB_MODE:

- MB_MODE defines the Modbus function code.
- MB_DATA_ADDR contains the remote address.
- MB_DATA_LEN contains the number of values to be read/written.

Example:

- MB_MODE=104, MB_DATA_ADDR=17,834, MB_DATA_LEN=125
  - MB_MODE=104 defines the function code 04 (read input words).
  - MB_DATA_ADDR=17,834 defines the remote address 17,834.
  - MB_DATA_LEN=125 defines that 125 values are read.

The following table shows the relationship between the input parameters MB_MODE, MB_DATA_ADDR, MB_DATA_LEN of the "MB_CLIENT" instruction and the associated Modbus function.

MB_MODE	MB_DATA_ADDR	MB_DATA_LEN	Modbus function	Function and data type
0	1 to 9,999	1 to 2,000	01	Read 1 to 2,000 output bits on the remote address 0 to 9,998
0	10,001 to 19,999	1 to 2,000	02	Read 1 to 2,000 input bits on the remote address 0 to 9,998
0	• 40,001 to 49,999 • 400,001 to 465,535	1 to 125	03	• Read 1 to 125 holding registers on the remote address 0 to 9,998 • Read 1 to 125 holding registers on the remote address 0 to 65,534
0	30,001 to 39,999	1 to 125	04	Read 1 to 125 input words on the remote address 0 to 9,998
1	1 to 9,999	1	05	Write 1 output bit on the remote address 0 to 9,998

MB_MODE	MB_DATA_ADDR	MB_DATA_LEN	Modbus function	Function and data type
1	• 40,001 to 49,999 • 400,001 to 465,535	1	06	• Write 1 holding register on the remote address 0 to 9,998 • Write 1 holding register on the remote address 0 to 65,534
1	1 to 9,999	2 to 1,968	15	Write 2 to 1,968 output bits on the remote address 0 to 9,998
1	• 40,001 to 49,999 • 400,001 to 465,535	2 to 123	16	• Write 2 to 123 holding registers on the remote address 0 to 9,998 • Write 2 to 123 holding registers on the remote address 0 to 65,534
2	1 to 9,999	1 to 1,968	15	Write 1 to 1,968 output bits on the remote address 0 to 9,998
2	• 40,001 to 49,999 • 400,001 to 465,535	1 to 123	16	• Write 1 to 123 holding registers on the remote address 0 to 9,998 • Write 1 to 123 holding registers on the remote address 0 to 65,534
11	The MB_DATA_ADDR and MB_DATA_LEN parameters are not evaluated when this function is executed.		11	Read status word and event counter of the server: • The status word reflects the the processing status (0 - not processing, 0xFFFF - processing). • The event counter is incremented when the Modbus request was executed successfully. If an error occurred during execution of a Modbus function, a message is sent by the server but the event counter is not incremented.
80	-	1	08	Check the server status with the diagnostic code 0x0000 (return loop test - the server sends the request back): • 1 WORD per call
81	-	1	08	Reset the event counter of the server with the diagnostic code 0x000A: • 1 WORD per call
101	0 to 65,535	1 to 2,000	01	Read 1 to 2,000 output bits on the remote address 0 to 65,535
102	0 to 65,535	1 to 2,000	02	Read 1 to 2,000 input bits on the remote address 0 to 65,535
103	0 to 65,535	1 to 125	03	Read 1 to 125 holding registers on the remote address 0 to 65,535
104	0 to 65,535	1 to 125	04	Read 1 to 125 input words on the remote address 0 to 65,535
105	0 to 65,535	1	05	Write 1 output bit on the remote address 0 to 65,535
106	0 to 65,535	1	06	Write 1 holding register on the remote address 0 to 65,535
115	0 to 65,535	1 to 1,968	15	Write 1 to 1,968 output bits on the remote address 0 to 65,535
116	0 to 65,535	1 to 123	16	Write 1 to 123 holding registers on the remote address 0 to 65,535
3 to 10, 12 to 79, 82 to 100, 107 to 114, 117 to 255				Reserved

Note

Unit Identifier for RTU devices over MB TCP

You can find the Unit Identifier for RTU devices over Modbus TCP as "MB_Unit_ID" tag in the editable static tags of the instance data block of the instruction "MB_CLIENT". The Unit Identifier is used for unique assignment of the coupling partners.

See also

Description of MB_CLIENT (Page 4113)

MB_DATA_PTR parameter (S7-1200, S7-1500)

Description

The MB_DATA_PTR parameter is a pointer to a data buffer for the data to be received from the Modbus server or to be sent to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

For a buffer in the memory area (M), use a pointer in the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

Depending on the memory area in which the data buffer is located, MB_DATA_PTR can reference different data structures:

- When you use a global DB with optimized access, MB_DATA_PTR can reference a tag with elementary data type or an array of elementary data types. The following data types are supported:

Data type	Length in bits
Bool	1
Byte, SInt, USInt, Char	8
Word, Int, WChar, UInt	16
DWord, DInt, UDInt, Real	32

  In such cases, all supported data types can be used for all Modbus functions. For example, MB_CLIENT can also write a received bit in a tag of the byte type to a specified address without changing other bits in this byte. Therefore, it is not necessary to have an array of bits in order to execute bit-oriented functions.

- If you use a bit memory address area or a global DB with standard access as memory area, there is no longer any restriction to the elementary data types for MB_DATA_PTR; MB_DATA_PTR can then also reference complex data structures such as PLC data types (UDTs) and system data types (SDTs).

Note

Using a bit memory address area as data buffer

If you use a bit memory address area as data buffer for MB_DATA_PTR, you need to observe this variable. With S7-1500-CPUs it is 16 KB, with S7-1200-CPUs it is 8 KB.

CONNECT parameter (S7-1200, S7-1500)

Connection descriptions at the CONNECT parameter up to and including instruction version V4.1 of MB_CLIENT

Two different connection descriptions can be used for the "MB_CLIENT" instruction:

- Programmed connections with the structure TCON_IP_v4
  The connection parameters are stored in the TCON_IP_v4 structure and the connection is set up with the call of the instruction "MB_CLIENT".
- Configured connections with the structure TCON_Configured
  The configured connection has already been established by the CPU. Use the structure TCON_Configured to specify which existing connection is to be used for the instruction.

Each instance of the instruction "MB_CLIENT" requires a unique connection. Create a separate structure TCON_IP_v4 or TCON_Configured for each instance of the instruction to describe the connection.

Connection description for programmed connections

Use the following structure for connection description to TCON_IP_v4 for programmed connections at the CONNECT parameter:

- Make sure that only connections of the type TCP are specified in the TCON_IP_v4 structure.
- The connection may not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). The parameter uniquely identifies a connection within the CPU. Each individual instance of the instruction "MB_CLIENT" must use a unique ID.
4	ConnectionType	BYTE	11	Connection type Select 11 (decimal) for TCP. Other connection types are not permitted. If another connection type (e.g. UDP) is used, a corresponding error message is output at the STATUS parameter of the instruction.
5	ActiveEstablished	BOOL	TRUE	ID for the manner in which the connection is established Select TRUE for active connection establishment.

Byte	Parameter	Data type	Start value	Description
6 … 9	RemoteAddress	ARRAY [1..4] of BYTE	-	IP address of the connection partner (Modbus server), for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1
10 … 11	RemotePort	UINT	502	Port number of the remote connection partner (value range: 1 to 49151). Use the IP port number of the server to which the client establishes the connection and communicates using the TCP/IP protocol (default value: 502).
12 … 13	LocalPort	UINT	0	Port number of the local connection partner: • Port numbers: 1 to 49151 • Any port: "0"

Note

Migration of the instruction "MB_CLIENT" version 2.1

The parameters CONNECT_ID, IP_PORT and IP_OCTET_x are mapped in version 3.0 of the "MB_CLIENT" instruction in the TCON_IP_v4 structure:

- The parameter CONNECT_ID of the "MB_CLIENT" V2.1 instruction corresponds to the ID parameter of TCON_IP_v4.
- The parameter IP_PORT of the "MB_CLIENT" V2.1 instruction corresponds to the RemotePort parameter of TCON_IP_v4.
- The four parameters IP_OCTET_x of the "MB_CLIENT" V2.1 instruction correspond to the array of the RemoteAddress parameter of TCON_IP_v4.

Connection description for configured connections

For programmed connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured.

- Make sure that only connections of the type TCP are specified in the TCON_Configured structure.
- The connection may not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceId	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	0	Connection type Select 254 (decimal) for a configured connection.

Connection descriptions at the CONNECT parameter for instruction versions > V4.1 and for firmware versions >= V2.5 of S7-1500 CPUs

Alongside the connections with the structures TCON_IP_v4 and TCON_Configured, the following connection descriptions for TCP are also possible for instruction versions > V4.1 of MB_CLIENT and as of firmware version V2.5 of S7-1500 CPUs:

- TCON_IP_V4_SEC
  For a description, see AUTOHOTSPOT
- TCON_QDN
  For a description, see AUTOHOTSPOT
- TCON_QDN_SEC
  For a description, see AUTOHOTSPOT

For more information on secure connections, see AUTOHOTSPOT

Parameter STATUS (S7-1200, S7-1500)

Parameter STATUS (general status information)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No job active and no connection established (REQ=0, DISCONNECT=1).
7001	Connection establishment triggered.
7002	Intermediate call. Connection is being established.
7003	Connection is being terminated.
7004	Connection established and monitored. No job processing active.
7005	Data is being sent.
7006	Data is being received.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Parameter STATUS (protocol error)

STATUS* (W#16#)	Local and/ or remote errors	Error code in the answer from MB_SERVER (B#16#)	Description
80C8	Local	-	No response of the server in the defined period. Check the connection to the Modbus server. This error is only reported on completion of the configured repeated attempts. If the "MB_CLIENT" instruction does not receive an answer with the originally transferred transaction ID (see static tag MB_TRANSACTION_ID) within the defined period, this error code is output.
8380	Local	-	Received Modbus frame has incorrect format or too few bytes were received.
8381	Remote	01	Function code is not supported.
8382	Local	-	• The length of the Modbus frame in the frame header does not match the number of received bytes. • The number of bytes does not match the number of actually transmitted bytes (only functions 1-4). For example, this is the case when "MB_CLIENT" requests an odd number of words, but "MB_SERVER" always sends an even number of words. • The start address in the received frame does not match the saved start address (functions 5, 6, 15, 16). • The number of words does not match the number of actually transmitted words (functions 15 and 16).
	Remote	03	Invalid length specification in received Modbus frame. Check the server side.
8383	Local	-	Error reading or writing data or access outside the address area of MB_DATA_PTR (Page 4120).
	Remote	02	Error reading or writing data or access outside the address area of the server
8384	Local	-	• Invalid exception code received. • A different data value was received than was originally sent by the client (functions 5, 6 and 8). • Invalid status value received (function 11)
	Remote	03	Error in data value for function 5
8385	Local	-	• Diagnostics code not supported. • A different subfunction code was received than was originally sent by the client (function 8).
	Remote	03	Diagnostics code not supported
8386	Local	-	Received function code does not match the one sent originally.
8387	Local	-	The protocol ID of the Modbus TCP frame received by the server is not "0".
8388	Local	-	The Modbus server sent a different data length than was requested. This error occurs only when using the Modbus functions 5, 6, 15 or 16.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".			

Parameter STATUS (parameter error)

STATUS* (W#16#)	Description
80B6	Invalid connection type, only TCP connections are supported.
80BB	Invalid value at ActiveEstablished parameter (identifier for the type of connection establishment, see CONNECT parameter (Page 4121)): • Only passive connection establishment permitted for server (ActiveEstablished = FALSE). • Only active connection establishment permitted for client (ActiveEstablished = TRUE).
8188	The MB_MODE parameter has an invalid value.
8189	Invalid addressing of data at the MB_DATA_ADDR parameter.
818A	Invalid data length at the MB_DATA_LEN parameter.
818B	The MB_DATA_PTR parameter has an invalid pointer. You should also check the values of the MB_DATA_ADDR (Page 4120) and MB_DATA_LEN parameters.
818C	Timeout at parameter BLOCKED_PROC_TIMEOUT or RCV_TIMEOUT (see static tags of instruction). BLOCKED_PROC_TIMEOUT and RCV_TIMEOUT must be between 0.5 s and 55 s.
8200	• A different Modbus request is currently being processed via the port. • Another instance of MB_CLIENT with the same connection parameters is processing an existing Modbus request.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Note

Error codes of internally used communications instructions

With the "MB_CLIENT" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND", "TRCV", "T_DIAG" and "TRESET") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_CLIENT" instruction. The error codes are displayed for the respective instruction under STATUS in the "Static" section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

Note

Communication error when sending or receiving data

If a communication error occurs when sending or receiving data (80C4 (Temporary communications error. The specified connection is temporarily down.), 80C5 (Remote partner closed connection actively.), 80A1 (The specified connection is disconnected or is not yet established.)), the existing connection is terminated.

This also means that you can see all STATUS values that are returned when the connection is terminated and that the STATUS code that caused the connection to be terminated is only output when the connection is terminated.

Example: If a temporary communication error occurs when data is received, the STATUS 7003 (ERROR=false) is output initially and then 80C4 (ERROR=true).

See also

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 883)

MB_SERVER: Communicating via PROFINET as a Modbus TCP server (S7-1200, S7-1500)

Description of MB_SERVER (S7-1200, S7-1500)

Description

The "MB_SERVER" instruction communicates as Modbus TCP server via a PROFINET connection. The instruction "MB_SERVER" processes connection requests of a Modbus TCP client, receives and processes Modbus requests and sends responses.

For the S7-1200 with firmware version V4.0 you can use the "MB_SERVER" instruction up to and including library version V3.1. With the S7-1200 as of firmware version V4.1 and the S7-1500, you can use the "MB_SERVER" instruction of all library versions.

The connection can take place via the local interface of the CPU or CM/CP.

To use the instruction, you do not require an additional hardware module.

NOTICE
Security information Note that each client of the network is given read and write access to the process image inputs and outputs and to the data block or bit memory area defined by the Modbus holding register. The option is available to restrict access to an IP address to prevent unauthorized read and write operations. Note, however, that the shared address can also be used for unauthorized access.

Multiple server connections

You can create multiple Server connections. This allows a single CPU to accept connections from multiple Modbus TCP clients at the same time.

A Modbus TCP server can support several TCP connections and the maximum number of connections depends on the CPU being used.

The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections.

Modbus TCP connections can also be shared by "MB_CLIENT" and/or "MB_SERVER" instances.

In the case of Server connections, remember the following rules:

- Each "MB_SERVER" connection must use a unique instance DB.
- Each "MB_SERVER" connection must use a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- For each connection, the "MB_SERVER" instruction must be called individually.

Parameters

The following table shows the parameters of the "MB_SERVER" instruction:

Parameter	Declaration	Data type	Description
DISCONNECT	Input	BOOL	The "MB_SERVER" instruction is used to enter into a passive connection with a partner module. The server responds to a connection request from the IP address which is entered in the SDT "TCON_IP_v4" in the CONNECT parameter. You can use this parameter to control when a connection request is accepted: • 0: A passive connection is established when there is no communications connection. • 1: Initialization of the connection termination. If the input is set, no other operations are executed. The value 0003 is output at the STATUS parameter after successful connection termination.
MB_HOLD_REG (Page 4131)	InOut	VARIANT	Pointer to the Modbus holding register of the "MB_SERVER" instruction MB_HOLD_REG must always reference a memory area that is larger than two bytes. The holding register contains the values that may be accessed by a Modbus client using the Modbus functions 3 (read), 6 (write), 16 (multiple write) and 23 (reading and writing in one job). As the holding register, use either a global data block with optimized access or the memory area of the bit memories.
CONNECT (Page 4131)	InOut	VARIANT	Pointer to the structure of the connection description The following structures (SDTs) can be used: • TCON_IP_v4: Includes all address parameters that are required for establishing a programmed connection. The default address is 0.0.0.0 (any IP address), but you can enter a specific IP address so that the server only responds to requests from this address. When using TCON_IP_v4, the connection is established when calling the "MB_SERVER" instruction. • TCON_Configured (only with S7-1500): Includes the address parameters of a configured connection. When using TCON_Configured, the connection is established by the CPU after download of the hardware configuration.
NDR	Output	BOOL	"New Data Ready": • 0: No new data • 1: New data written by the Modbus client written

Parameter	Declaration	Data type	Description
DR	Output	BOOL	"Data Read": • 0: No data read • 1: Data read by the Modbus client
ERROR	Output	BOOL	If an error occurs during the call of the "MB_SERVER" instruction, the output of the ERROR parameter is set to "1". Detailed information about the cause of the problem is indicated by the STATUS parameter.
STATUS (Page 4134)	Output	WORD	Detailed status information of the instruction.

Note

Use of Modbus function 23 in the instruction "MB_SERVER"

The instruction "MB_SERVER" supports the use of Modbus function 23 with which you can write to a holding register and read from a holding register in a job; however, the instruction "MB_CLIENT" does NOT support this function and returns an error code.

Also note that for a job with read access and write access, write access is executed before the read access.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Static tags of MB_SERVER in instruction version V4.2

The following table describes the static tags of the instance data block of the "MB_SERVER" instruction used in the program. You can write the HR_Start_Offset tag. You can read the other tags to monitor the Modbus status.

Tag	Data type	Start value	Description
HR_Start_Offset	WORD	0	Assign the start address of the Modbus holding register.
QB_Start	WORD	0	Start address of the permitted addressing range of the outputs that can be written (bytes 0 to 65535)
QB_Count	WORD	0xFFFF	Number of output bytes that can be written by the Modbus master. Example: QB_Start=0 and QB_Count=10: Output bytes 0 to 9 can be written. QB_Count=0: No output byte can be written.
Request_Count	WORD	0	Total number of requests received by the server.
Server_Message_Count	WORD	0	Total number of received alarms for the relevant server.
Xmt_Rcv_Count	WORD	0	Counter for detecting the number of transfers during which an error occurred. The counter is only incremented when an invalid Modbus request is received.
Exception_Count	WORD	0	Counters for detecting the number of errors specifically for Modbus which cause an error message to "MB_CLIENT".
Success_Count	WORD	0	Event counter for detecting the number of requests that were successfully executed by the server.
Connected	BOOL	FALSE	Indicates whether the connection to the assigned client has been established or not: TRUE = connected, FALSE = not connected.

Mapping of Modbus addresses to the process image

The "MB_SERVER" instruction allows incoming Modbus functions (1, 2, 4, 5 and 15) direct read and write access to the process image inputs and outputs of the CPU (use of the data types BOOL and WORD).

For S7-1200-CPUs, the address space for the process image of the inputs and the process image of the outputs is 1 KB; it is 32 KB each for S7-1500-CPUs.

The following table shows the address space of the Modbus functions listed above.

Modbus function					
Function code	Function	Data area	Address space		
01	Read: Bits	Output	0	to	65.535
02	Read: Bits	Input	0	to	65.535
04	Read: WORD	Input	0	to	65.535
05	Write: Bit	Output	0	to	65.535
15	Write: Bits	Output	0	to	65.535

Incoming Modbus requests with the function codes 3, 6, 16 and 23 write or read the Modbus holding registers (you specify the holding register with the MB_HOLD_REG parameter).

Example: Addressing via static tag HR_Start_Offset

The addresses of the Modbus holding register start at 0 (from the perspective of the MB_CLIENT at 40.001). These addresses correspond to the address area of the CPU memory area for the holding register. You can also define the HR_Start_Offset tag so that the Modbus holding register has a start address other than 0.

Example: The holding register begins at MW100, and has a length of 100 WORD. An offset value in the HR_Start_Offset parameter means that the start address of the holding register is moved from 0 to 20. This causes an error whenever the holding register is addressed below the address 20 and above the address 119.

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (WORD)	0	99
	CPU address	MW100	MW298
20	Modbus address (WORD)	20	119
	CPU address	MW100	MW298

Modbus functions

The following table lists all the Modbus functions that are supported by the "MB_SERVER" instruction.

Function code	Description
01	Read output bits
02	Read input bits
03	Read a holding register
04	Read input words

Function code	Description
05	Write an output bit
06	Write a holding register
08	Diagnostics function: • Echo test (subfunction 0x0000): The "MB_SERVER" instruction receives a data word and returns this unchanged to the Modbus client. • Reset event counter (subfunction 0x000A): The "MB_SERVER" instruction resets the following event counters: "Success_Count", "Xmt_Rcv_Count", "Exception_Count", "Server_Message_Count" and "Request_Count".
11	Diagnostics function: Fetch event counter of the communication The "MB_SERVER" instruction uses an internal event counter for communication to record the number of successfully executed read and write requests sent to the Modbus server. The event counter is not incremented with the functions 8 or 11. The same holds true for requests that cause a communications error, for example, if a protocol error has occurred (e.g., the function code in the received Modbus request is not supported).
15	Write output bits
16	Write a holding register
23	Write a holding register and read holding register with a request

Example

An example project for the Modbus TCP communication between two S7-1500 CPUs can be found in the Service and Support Portal under Entry ID 94766380 (http://support.automation.siemens.com/WW/view/en/94766380).

Two Modbus functions are used in this example. For each Modbus function, a Modbus TCP connection is established using a Modbus block pair (MB_CLIENT and MB_SERVER).

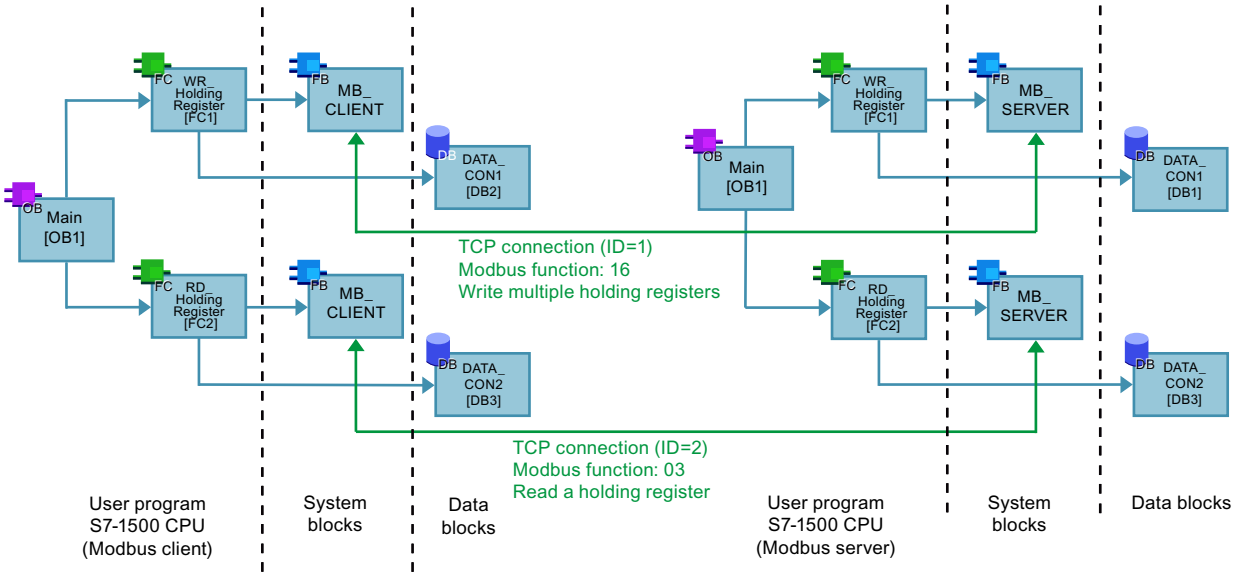

See also

Difference between synchronous and asynchronous instructions (Page 596)

MB_HOLD_REG parameter (S7-1200, S7-1500)

Description

The MB_HOLD_REG parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. You can use a global data block or a bit memory (M) as memory area.

- The high limit for the number of addresses in the data block (D) is determined by the maximum size of a DB of your CPU.
- The high limit for the number of bit memories (M) is determined by the maximum bit memory area of your CPU.

The following table shows examples of mapping Modbus addresses to the holding register for the Modbus functions 3 (read multiple WORD), 6 (write one WORD), 16 (write multiple WORD) and 23 (write and read multiple words).

Modbus addresses	MB_HOLD_REG parameter - examples		
0	MW100	DB10.DBW0	"Recipe".ingredient[1]
1	MW102	DB10.DBW2	"Recipe".ingredient[2]
2	MW104	DB10.DBW4	"Recipe".ingredient[3]
3	MW106	DB10.DBW6	"Recipe".ingredient[4]
4	MW108	DB10.DBW8	"Recipe".ingredient[5]

CONNECT parameter (S7-1200, S7-1500)

Connection descriptions at the CONNECT parameter up to and including instruction version V4.2 of MB_SERVER

Two different connection descriptions can be used for the "MB_SERVER" instruction:

- Programmed connections with the structure TCON_IP_v4
  The connection parameters are stored in the TCON_IP_v4 structure and the connection is set up with the call of the instruction "MB_SERVER".
- Configured connections with the structure TCON_Configured (only with S7-1500)
  The configured connection has already been established by the CPU. Use the structure TCON_Configured to specify which existing connection is to be used for the instruction.

Each instance of the instruction "MB_SERVER" requires a unique connection. Create a separate structure TCON_IP_v4 or TCON_Configured for each instance of the instruction to describe the connection.

Connection description for programmed connections

For programmed connections at the CONNECT parameter, use the following structure for connection description to TCON_IP_v4.

Make sure that only connections of the type TCP are specified in the TCON_IP_v4 structure.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). The parameter uniquely identifies a connection within the CPU. Each individual instance of the instruction "MB_SERVER" must use a unique ID. The ID must also not be used simultaneously by another instruction of a different type of communication.
4	ConnectionType	BYTE	11	Connection type Select 11 (decimal) for TCP. Other connection types are not permitted. If another connection type (e.g. UDP) is used, a corresponding error message is output at the STATUS parameter of the instruction.
5	ActiveEstablished	BOOL	FALSE	ID for the manner in which the connection is established Select FALSE for passive connection establishment.
6 … 9	RemoteAddress	ARRAY [1..4] of BYTE	0.0.0.0	IP address of the connection partner, for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1 If the instruction "MB_SERVER" is to accept connection requests from any connection partner, use "0.0.0.0" as IP address.
10 … 11	RemotePort	UINT	0	Port number of the remote connection partner (value range: 1 to 49151). If the instruction "MB_SERVER" is to accept connection requests from any port of the remote partner, use "0" as port number.
12 … 13	LocalPort	UINT	502	Port number of the local connection partner (value range: 1 to 49151). The number of the IP port defines which IP port is monitored for connection requests of the Modbus client. The default value is 502.

Note

Migration of the instruction "MB_SERVER" version 2.1

The parameters CONNECT_ID and IP_PORT are mapped in version 3.0 of the "MB_SERVER" instruction in the TCON_IP_v4 structure:

- The parameter CONNECT_ID of the "MB_SERVER" V2.1 instruction corresponds to the ID parameter of TCON_IP_v4.
- The parameter IP_PORT of the "MB_SERVER" V2.1 instruction corresponds to the LocalPort parameter of TCON_IP_v4.

Connection description for configured connections

For configured connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured.

Make sure that only connections of the type TCP are specified in the TCON_Configured structure.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). ID must be unique CPU-wide. ID must also not be used simultaneously by another instruction of a different type of communication. Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	-	Connection type Select 254 (decimal) for a configured connection.

Connection descriptions at the CONNECT parameter for instruction versions > V4.2 and for firmware versions >= V2.5 of S7-1500 CPUs

Alongside the connections with the structures TCON_IP_v4 and TCON_Configured, the following connection descriptions for TCP are also possible for instruction versions > V4.2 of MB_SERVER and as of firmware version V2.5 of S7-1500 CPUs:

- TCON_IP_V4_SEC
  For a description, see AUTOHOTSPOT
- TCON_QDN
  For a description, see AUTOHOTSPOT
- TCON_QDN_SEC
  For a description, see AUTOHOTSPOT

For more information on secure connections, see AUTOHOTSPOT

Parameter STATUS (S7-1200, S7-1500)

Parameter STATUS (general status information)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active and no connection established (REQ=0, DISCONNECT=1).
7001	First call. Connection establishment triggered.
7002	Intermediate call. Connection is being established.
7003	Connection is being terminated.
7005	Data is being sent.
7006	Data is being received.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Parameter STATUS (protocol error)

STATUS* (W#16#)	Error code in the error message from "MB_SERVER" (B#16#)	Description
8380	-	Received Modbus frame has incorrect format or too few bytes were received.
8381	01	Function code is not supported.
8382	03	Error in data length: • Invalid length specification in received Modbus frame. • The frame length entered in the header of the Modbus frame does not match the number of actually received bytes. • The number of bytes entered in the header of the Modbus frame does not match the number of actually received bytes (functions 15 and 16).
8383	02	Error in data address or access outside the address area of the holding register (MB_HOLD_REG (Page 4131) parameter).
8384	03	Error in the data value (function 05)
8385	03	Diagnostic code is not supported (only with function 08).
* The status codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".		

Parameter STATUS (parameter error)

STATUS* (W#16#)	Description
80BB	Invalid value at ActiveEstablished parameter (identifier for the type of connection establishment, see CONNECT parameter (Page 4131)): Only passive connection establishment permitted for server (active_established = FALSE).
8187	The MB_HOLD_REG parameter has an invalid pointer. Data area is too small.
8389	Invalid data area definition: • Invalid data_type value • DB number invalid or does not exist: – Invalid db value – DB number does not exist – DB number is already used by another data area – DB with optimized access – DB is not located in the work memory • Invalid length value • Overlapping of MODBUS address ranges that belong to the same MODBUS data type
* The status codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Note

Error codes of internally used communications instructions

With the "MB_SERVER" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND", "TRCV", "T_DIAG" and "T_RESET") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_SERVER" instruction. The error codes are displayed for the respective instruction under STATUS in the "Static" section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

Note

Communication error when sending or receiving data

If a communication error occurs when sending or receiving data (80C4 (Temporary communication error. The specified connection is temporarily terminated.), 80C5 (The remote partner has actively terminated the connection.), 80A1 (The specified connection is interrupted or has not been established yet.)), the existing connection is terminated.

This also means that you can see all STATUS values that are returned when the connection is terminated and that the STATUS code that caused the connection to be terminated is only output when the connection is terminated.

Example: If a temporary communication error occurs when data is received, the STATUS 7003 (ERROR=false) is output initially and then 80C4 (ERROR=true).

See also

Switching display formats in the program status (Page 7698)

GET_ERR_ID: Get error ID locally (Page 883)

Access to data areas in DBs instead of direct access to MODBUS addresses as of version V5.0 (S7-1200, S7-1500)

Access to data areas in DBs instead of direct access to MODBUS addresses as of version V5.0

With version V5.0 of MB_SERVER instructions and firmware versions V2.5 (S7-1500 CPUs) and V4.2 (S7-1200 CPUs), you can access data areas in DBs instead of directly accessing process images and holding registers. In doing so, the attribute "Optimized block access" must be disabled for the DB and it must not be located solely in the load memory.

If a MODBUS request arrives and you have not defined a data area for the MODBUS data type of the corresponding function code, the request is treated as in the previous instruction versions, i.e. process images and holding registers are accessed directly.

If, on the other hand, you have defined a data area for the MODBUS data type of the function code, the instruction MB_SERVER reads from or writes to that data area. Whether it reads or writes depends on the job type.

One individual MODBUS request can only ever be read from or written to one data area. If, for example, you want to read holding registers that extend over multiple data areas, you therefore require multiple MODBUS requests.

Rules for defining data areas

You can define up to eight data areas in different DBs; each DB must only contain one data area. An individual MODBUS request can only ever read from precisely one data area or write to precisely one data area. Each data area corresponds to one MODBUS address area. The data areas are defined in the static tag Data_Area_Array of the instance DB; Data_Area_Array is a field consisting of eight elements.

If you want to use less than eight data areas, the required data areas must be located one behind the other without any gaps. The first blank entry in the data areas ends the data area search during processing. If, for example, you have defined the field elements 1, 2, 4 and 5, only field elements 1 and 2 will be recognized as field element 3 is empty.

The Data_Area_Array field consists of 8 elements: Data_Area_Array[1] to Data_Area_Array[8]

Each field element Data_Area_Array[x], 1 <= x <= 8, is a UDT of the type MB_DataArea and is structured as follows:

Parameter	Data type	Meaning
data_type	UInt	Identifier for the MODBUS data type that is mapped to this data area: • 0: Identifier for an empty field element or an unused data area. In this case, the values of db, start and length are irrelevant. • 1: Process image output (used with function codes 1, 5 and 15) • 2: Process image input (used with function code 2) • 3: Holding register (used with function codes 3, 6 and 16) • 4: Input register (used with function code 4) Note: If you have defined a data area for a MODBUS data type, the instruction MB_SERVER can no longer access this MODBUS data type directly. If the address of a MODBUS request for such a data type does not correspond to a defined data area, a value of W#16#8383 is returned in STATUS.
db	UInt	Number of the data block to which the MODBUS register or bits subsequently defined are mapped. The DB number must be unique in the data areas. The same DB number must not be defined in multiple data areas. The DB must have standard access and must not be located solely in the load memory. Data areas also start with the byte address 0 of the DB. Permitted values: 1 to 60999
start	UInt	First MODBUS address that is mapped to the data block starting from address 0.0. Permitted values: 0 to 65535
length	UInt	Number of bits (for the values 1 and 2 of data_type) or number of registers (for the values 3 and 4 of data_type). The MODBUS address areas of one and the same MODBUS data type must not overlap. Permitted values: 1 to 65535

Examples of the definition of data areas

- First example: data_type = 3, db = 1, start = 10, length = 6
  The holding registers (data_type = 3) are mapped in data block 1 (db = 1). The Modbus address 10 (start = 10) is located at data word 0. The last valid Modbus address 15 (length = 6) is located at data word 5.
- Second example: data_type = 2, db = 15, start = 1700, length = 112
  The inputs (data_type = 2) are mapped in data block 15 (db = 15). The Modbus address 1700 (start = 1700) is located at data word 0. The last valid Modbus address 1811 (length = 112) is located at data word 111.

Restriction of read access to process images as of version V5.0 (S7-1200, S7-1500)

Restriction of read access to process images

With version V5.0 of MB_SERVER instructions, you can define one area in the process image input and one area in the process image output to which remote MODBUS devices can have read access. Read access by remote MODBUS devices to addresses outside these process image areas is then no longer possible.

Note

Restriction of write access to process images

The option of restricting write access to the process image output of a specific area is possible as of instruction version V4.2.

Definition of read areas in the process images

Read areas in the process images are defined in the following static tags of the instance DB:

- QB_Read_Start: Address of the first byte in the process image output that can be read by a remote MODBUS device (applies to function code 1)
- QB_Read_Count: Number of bytes in the process image output that can be read by a remote MODBUS device (applies to function code 1)
- IB_Read_Start: Address of the first byte in the process image input that can be read by a remote MODBUS device (applies to function codes 2 and 4)
- IB_Read_Count: Number of bytes in the process image input that can be read by a remote MODBUS device (applies to function codes 2 and 4)

Static tags in the instance DB for defining write and read areas in the process images

The following table describes the static tags in the instance DB of instruction MB_SERVER that are listed above and which you use to define the read areas in the process images.

For the sake of completeness, the table also lists the static tags that you use to define the write areas in the process images as of version V4.2 (QB_Start and QB_Count).

Tag	Data type	Start value
QB_Start	UInt	0
QB_Count	UInt	65535
QB_Read_Start	UInt	0
QB_Read_Count	UInt	65535
IB_Read_Start	UInt	0
IB_Read_Count	UInt	65535

The statistical tags NDR_immediate and DR_immediate (S7-1200, S7-1500)

NDR_immediate and DR_immediate

The statistical tags NDR_immediate and DR_immediate are contained in the instance data block of the instruction MB_SERVER as of instruction version V5.0.

Tag	Data type	Start value	Meaning
NDR_immediate	BOOL	FALSE	Identical meaning as the parameter NDR (New Data Ready). NDR_immediate is updated in the same call by MB_SERVER in which a MODBUS-TCP write request is processed.
DR_immediate	BOOL	FALSE	Identical meaning as the parameter DR (Data Read). DR_immediate is updated in the same call by MB_SERVER in which a MODBUS-TCP write request is processed.

Differences between the versions <= V3.x and >= V4.0 of the Modbus TCP library (S7-1200, S7-1500)

Differences between the versions <= V3.x and >= V4.0 of the Modbus instructions

The following differences between versions exist with the MB_SERVER/MB_CLIENT MODBUS instructions:

- Address parameter
  - In versions <= 3.x, the address data for the Modbus TCP server is specified via the input parameters "IP_x".
  - Versions >= 4.0 use TCON_IP_V4 and TCON_Configured system data types at the CONNECT input parameter for this purpose.
- If an error occurs during execution, as of version 4.0 of the Modbus instructions, there are more STATUS alarms available which can be evaluated accordingly.

You can find detailed information in the instruction descriptions for MB_CLIENT (Page 4113) and MB_SERVER (Page 4126).

Note

Changing the Modbus TCP library from a version <= V3.x to a version >= V4.0

When you replace the Modbus TCP library from a version <= V3.x to a version >= V4.0 you also need to replace the Open User Communication library. Afterwards check all the instructions relevant for your program.

Redundant MODBUS (TCP) communication for library versions V5.0 or higher

MB_RED_CLIENT: Redundant communication over PROFINET as MODBUS-TCP client (S7-1200, S7-1500)

MB_RED_CLIENT description (S7-1200, S7-1500)

Description

You can use this instruction to establish a connection between a 1200/1500 CPU and a device that supports the Modbus/TCP protocol.

The instruction "MB_RED_CLIENT" communicates as a Modbus/TCP client over the PROFINET connection. You use the instruction "MB_RED_CLIENT" to establish a redundant connection between the client and the server, send Modbus requests, receive responses and control connection termination by the Modbus/TCP client.

The instruction "MB_RED_CLIENT" can be used in the following CPUs:

	"MB_RED_CLIENT" V1.0	"MB_RED_CLIENT" V1.1
S7-1200	FW V4.2 and higher	FW V4.2 and higher
S7-1500	FW V2.5 and higher	FW V2.5 and higher
S7-1500R	--	FW V2.6 and higher
S7-1500H	--	FW V2.6 and higher

To use the instruction, you do not require an additional hardware module.

Multiple client connections

The CPUs can process multiple Modbus/TCP client connections. The maximum number of connections depends on the CPU being used and can be found in the technical specifications of the CPU. The total number of connections of one CPU, including those of the Modbus/TCP clients and server must not exceed the maximum number of supported connections.

With individual client connections, remember the following rules:

- Each "MB_RED_CLIENT" connection must use a unique instance DB.
- For each "MB_RED_CLIENT" connection, a unique server IP address must be specified.
- Each "MB_RED_CLIENT" connection requires a unique connection ID. The connection IDs must be unique throughout the CPU.

For information concerning the redundancy setup, refer to: Operation and redundancy (Page 4142).

Parameters

The following table shows the parameters of the instruction "MB_RED_CLIENT":

Parameters	Declaration	Data type	Description
REG_KEY (Page 4150)	Input	STRING[17]	Registration code for licensing The "MB_RED_CLIENT" instruction must be licensed on each CPU individually.
USE_ALL_CONN	Input	BOOL	Specify the number of configured connections over which the frame is to be sent • 0: Send frame over one connection, switch to next connection only in case of error • 1: Send frame over all configured connections
REQ	Input	BOOL	Modbus query to the Modbus/TCP server The REQ parameter is level-controlled. This means that as long as the input is set (REQ = TRUE), the instruction sends communication requests. If the connection has not been established yet, it is established now and the Modbus frame is sent immediately thereafter. Changes to the input parameters will not become effective until the server has responded or an error message has been output. If the parameter REQ is set again during an ongoing Modbus request, no additional transmission takes place afterwards.
DISCONNECT	Input	BOOL	With this parameter, you control the establishment and termination of the connection to the Modbus server: • 0: Establish communication connection to the connection partner configured at the CONNECT parameter (see CONNECT parameter). • 1: Disconnect the communication connection. No other function is executed during connection termination. The value 0003 is output at the STATUS_x parameter after successful connection termination.
MB_MODE (Page 4152)	Input	USINT	Selects the mode of the Modbus request (read, write or diagnostics) or direct selection of a Modbus function
MB_DATA_ADDR (Page 4152)	Input	UDINT	Modbus address depending on MB_MODE
MB_DATA_LEN	Input	UINT	Data length: Number of bits or registers for the data access.
MB_DATA_PTR (Page 4152)	InOut	VARIANT	Pointer to a data buffer for the data to be received from the Modbus server or to be sent to the Modbus server.
LICENSED (Page 4150)	Output	BOOL	• 0: Instruction is not licensed • 1: Instruction is licensed
IDENT_CODE (Page 4150)	Output	STRING[18]	Identification for licensing. Use this string to request the registration code REG_KEY.
DONE	Output	BOOL	The bit at output parameter DONE is set to "1" as soon as the activated Modbus job is completed without errors on at least one connection.
BUSY	Output	BOOL	• 0: No Modbus request in progress • 1: Modbus request being processed The output parameter BUSY is not set during connection establishment and termination.

Parameters	Decla-ration	Data type	Description
ERROR	Output	BOOL	• 0: No error • 1: The activated Modbus job could not be transmitted successfully on any of the configured connections. The cause of error is indicated by the STATUS_x parameter.
STATUS_0A (Page 4155)	Output	WORD	Detailed status information of the instruction on connection 0A.
STATUS_1A (Page 4155)	Output	WORD	Detailed status information of the instruction on connection 1A.
STATUS_0B (Page 4155)	Output	WORD	Detailed status information of the instruction on connection 0B.
STATUS_1B (Page 4155)	Output	WORD	Detailed status information of the instruction on connection 1B.
RED_ERR_S7 (Page 4155)	Output	BOOL	• 0: No redundancy error in SIMATIC • 1: Redundancy error in SIMATIC
RED_ERR_DEV (Page 4155)	Output	BOOL	• 0: No redundancy error on side of link partner • 1: Redundancy error on side of link partner
TOT_COM_ERR (Page 4155)	Output	BOOL	• 0: At least 1 configured connection is established • 1: Complete loss of communication, all configured connections are terminated

Note

Consistent input data during an "MB_RED_CLIENT" call

When a Modbus client instruction is called, the values of the input parameters are stored internally. They must not be changed while the frame is being processed.

Operation and redundancy (S7-1200, S7-1500)

Description

The communication nodes can be designed as standalone or redundant. If one of the partners is designed as standalone, we refer to it as single-sided redundancy. If both partners are designed redundantly, we refer to it as double-sided redundancy.

Port numbers for client and server

The Modbus client uses a port number starting at 2000. The Modbus server is usually addressed over the port number 502.

Single-sided redundancy

Description

One connection each must be configured for each connection between the communication partners. The connection points of the **SIMATIC S7** are referred to as **0** and **1**; the connection points of the **communication partner** are referred to as **A** and **B**.

The R-CPU or H-CPU 1 refers to the connection point 0, the R-CPU or H-CPU 2 refers to the connection point 1.

Configuration

If the S7 is designed redundantly, one connection is created from the S7 connection point 0 to junction A of the link partner and one connection from the S7 connection point 1 to junction A of the link partner.

- Connection from the S7 connection point **0** to the partner/node **A** => connection **0A**
- Connection from the S7 connection point **1** to the partner/node **A** => connection **1A**

The figure illustrates the connection designations.

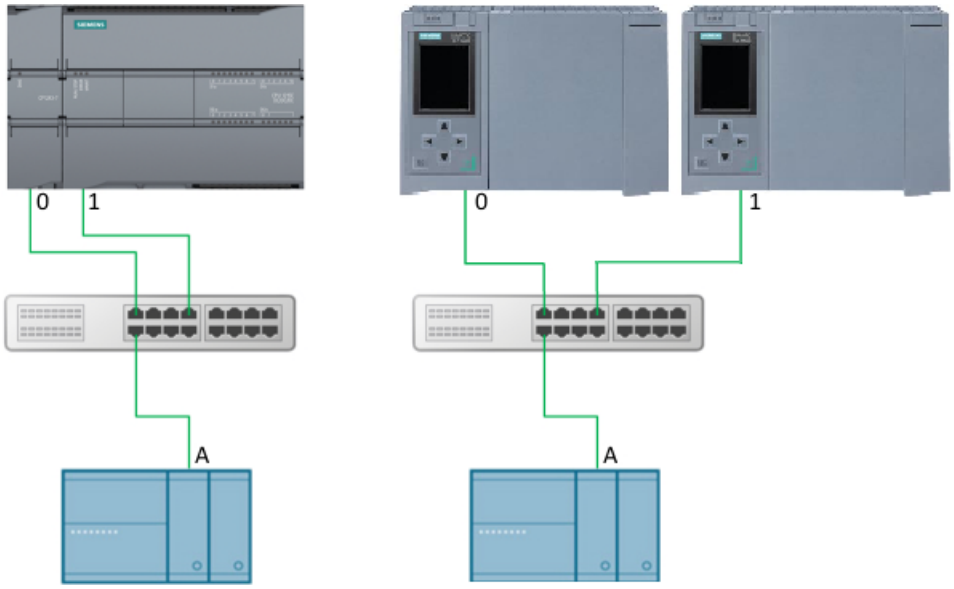

Figure 4-9 Modbus/TCP: Single-sided redundancy S7

If the S7 is designed as standalone and the link partner is designed redundantly, one connection is created from the S7 connection point 0 to junction A of the link partner and one connection from the S7 connection point 0 to junction B of the link partner.

- Connection from the S7 connection point **0** to the partner/node **A** => connection **0A**
- Connection from the S7 connection point **0** to the partner/node **B** => connection **0B**

The figure illustrates the connection designations.

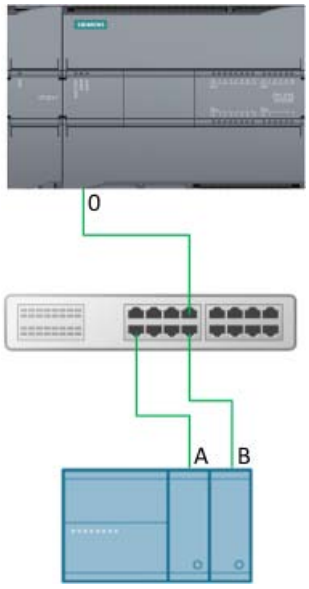

Figure 4-10 Modbus/TCP: Single-sided redundancy partner

Double-sided redundancy

Description

One connection each must be configured for each connection between the communication partners. The connection points of the **SIMATIC S7** are referred to as **0** and **1**; the connection points of the **communication partner** are referred to as **A** and **B**.

The R-CPU or H-CPU 1 refers to the connection point 0, the R-CPU or H-CPU 2 refers to the connection point 1.

Configuration

In the case of double-sided redundancy, two connections are created from connection point 0 and two connections from connection point 1 of the S7 to the junctions A and B of the link partner.

- Connection from the S7 connection point **0** to the partner/node **A** => connection **0A**
- Connection from the S7 connection point **1** to the partner/node **A** => connection **1A**
- Connection from the S7 connection point **0** to the partner/node **B** => connection **0B**
- Connection from the S7 connection point **1** to the partner/node **B** => connection **1B**

The figure illustrates the connection designations.

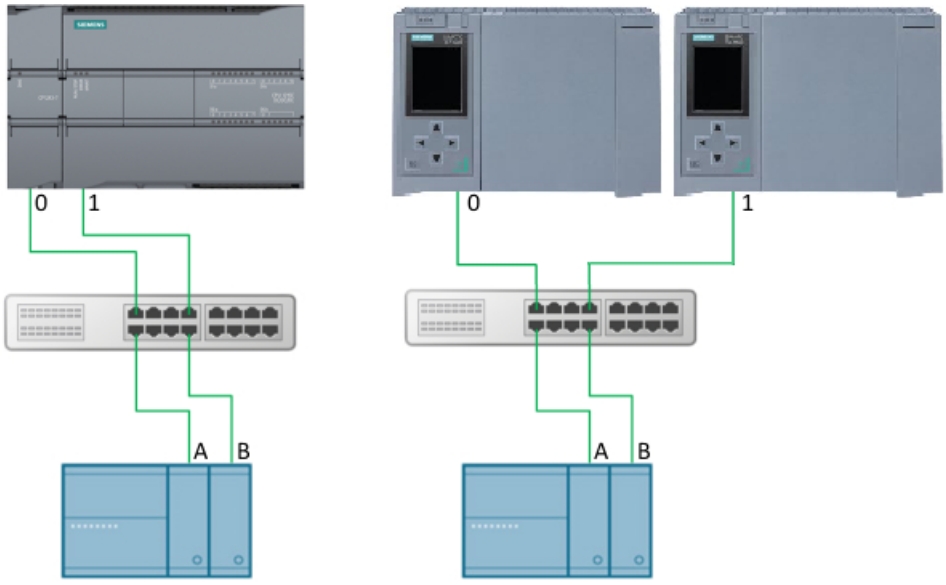

Figure 4-11 Modbus/TCP: Double-sided redundancy

Frame processing

General

The frames can be sent via one or via all configured connections.

Send frames via one connection

The MODBUS frame is sent via one - the currently active - connection with the setting USE_ALL_CONN = FALSE. In case of a timeout (no response from the server) or a connection fault an attempt is made to send the frame via the other (maximum 4) configured connections. The sequence is then 0A, 1A, 0B and 1B. If a frame was transmitted successfully via a connection, this connection is marked as "active" and the further frame traffic is executed via this connection. In the case of a connection fault of the active connection it is again attempted to send the frame via all configured connections. If all send attempts fail, ERROR and STATUS_x are set accordingly.

If a response frame was received, a plausibility check is executed. If this check is successful, the required actions are performed and the job is executed without errors; the output DONE is set. If errors are detected during the check, the job is ended without errors, the bit ERROR is set and an error number is displayed at STATUS_x. In this case no new attempt is made to send the frame on the next configured connection. A switchover to the other configured connections only takes place if a connection fault was detected or no response was received.

Sending frames via all connections

The MODBUS frame is sent via all configured, established connections with the setting USE_ALL_CONN = TRUE. A validity check is performed after the response frame has been received on one of the connections. If this check is successful, the required actions are performed. If a valid response frame was received on at least one connection, the output DONE is set.

Redundancy outputs RED_ERR_S7, RED_ERR_DEV and TOT_COM_ERR

Description

The redundancy bits RED_ERR_S7, RED_ERR_DEV and TOT_COM_ERR are set based on the states of the status outputs.

Display of the interrupt bits for redundancy setup on both sides

Number of faulty connections	STATUS_0A	STATUS_0B	STATUS_1A	STATUS_1B	RED_ERR_S7	RED_ERR_DEV	TOT_COM_ERR
0	okay	okay	okay	okay	FALSE	FALSE	FALSE
1	okay	okay	okay	Error	FALSE	FALSE	FALSE
	okay	okay	Error	okay	FALSE	FALSE	FALSE
	okay	Error	okay	okay	FALSE	FALSE	FALSE
	Error	okay	okay	okay	FALSE	FALSE	FALSE
2	okay	okay	Error	Error	TRUE	FALSE	FALSE
	okay	Error	okay	Error	FALSE	TRUE	FALSE
	Error	okay	okay	Error	FALSE	FALSE	FALSE
	okay	Error	Error	okay	FALSE	FALSE	FALSE
	Error	okay	Error	okay	FALSE	TRUE	FALSE
	Error	Error	okay	okay	TRUE	FALSE	FALSE
3	Error	Error	Error	okay	TRUE	TRUE	FALSE
	Error	Error	okay	Error	TRUE	TRUE	FALSE
	Error	okay	Error	Error	TRUE	TRUE	FALSE
	okay	Error	Error	Error	TRUE	TRUE	FALSE
4	Error	Error	Error	Error	TRUE	TRUE	TRUE

Display of the interrupt bits for redundancy setup on one side

Number of faulty connections	STATUS_0A	STATUS_0B	STATUS_1A	STATUS_1B	RED_ERR_S7	RED_ERR_DEV	TOT_COM_ERR
0	okay	0AFF	okay	0AFF	FALSE	FALSE	FALSE
1	okay	0AFF	Error	0AFF	TRUE	TRUE	FALSE
	Error	0AFF	okay	0AFF	TRUE	TRUE	FALSE
2	Error	0AFF	Error	0AFF	TRUE	TRUE	TRUE

Parameter assignment (S7-1200, S7-1500)

Description

The instruction "MB_RED_CLIENT" **V1.0** and **V1.1** can be used for S7-1200 and for S7-1500.

S7-1200

The connections are implemented over the local interface of the CPU or CM/CP. The connections are configured and established using the structure TCON_IP_V4.

S7-1500

The connections are implemented over the local interface of the CPU or CM/CP. Configured connections over the structure TCON_IP_V4 as well as configured connections over the structure TCON_Configured are possible.

The instruction "MB_RED_CLIENT" **V1.1** can be used for S7-1200R and for S7-1500H.

S7-1500R and S7-1500H

The connections are implemented over the local interface of the CPU or CM/CP. The connections are configured and established using the structure TCON_IP_V4.

Configuration of "MB_RED_CLIENT"

The following settings are made with the help of the configuration dialog of the instruction "MB_RED_CLIENT":

- Connection parameters for the connections 0A, 1A, 0B and 1B (Redundancy configuration see Operation and Redundancy (Page 4142))
- Internal parameter (optional)

The configuration dialog can be opened with the instruction or via the technology objects.

Connection parameters

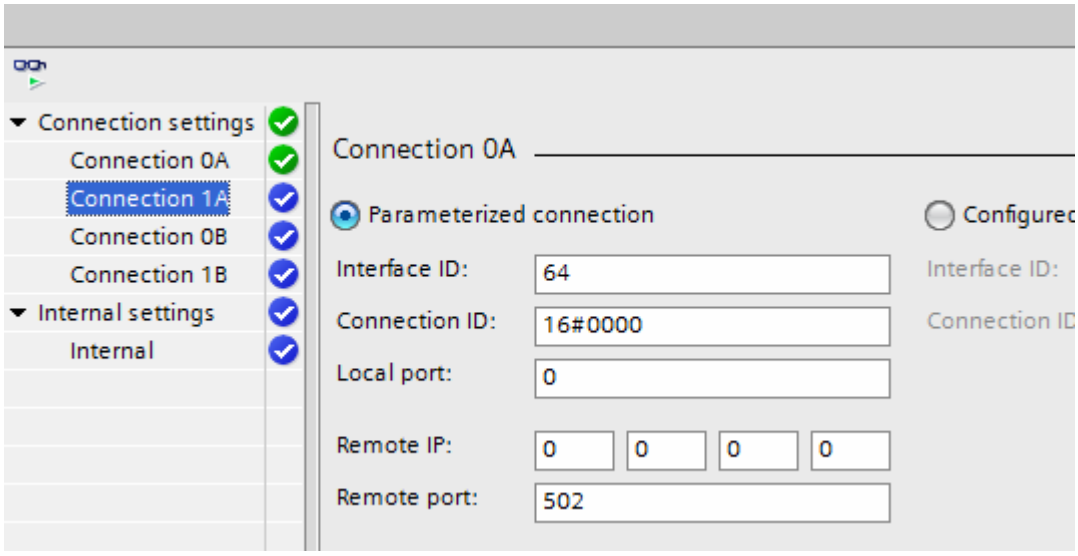

Figure 4-12 Modbus/TCP: configured client connection

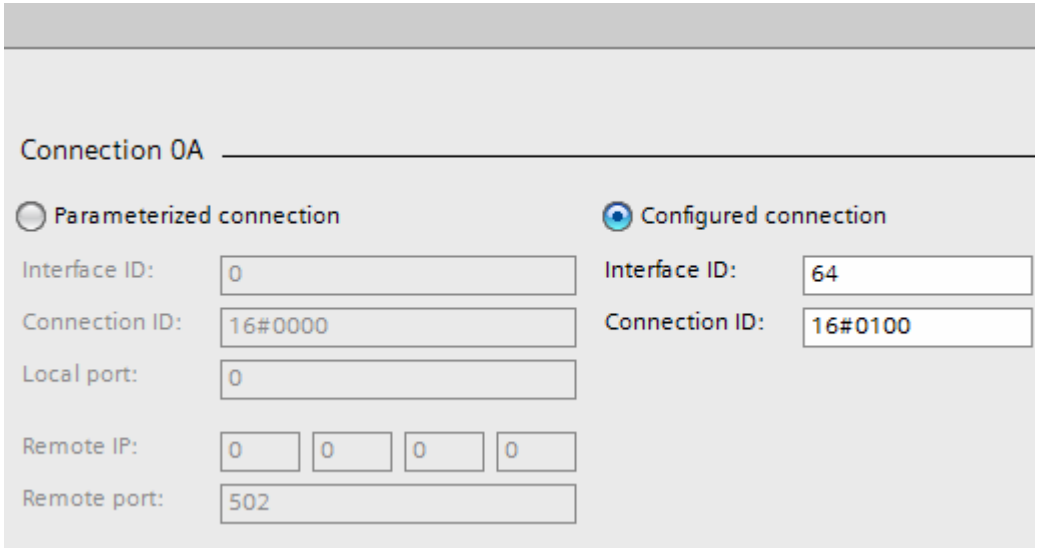

Figure 4-13 Modbus/TCP: configured client connection

Tag	Start value	Description
Configured connections		
Interface ID	64	HW identifier of the PN interface used
Connection ID	16#0000	Connection IDs for the connections used These connection IDs must be unique throughout the CPU.
Local port	0	Local port number of the client. By default no port number is entered for the client.
Remote IP	0.0.0.0	Remote IP address of the server

Tag	Start value	Description
Remote port	502	Remote port number of the server The default port for Modbus/TCP server is 502.
Configured connections		
Interface ID	64	HW identifier of the PN interface used
Connection ID	16#0000	Connection IDs for the connections used These connections are configured in the network view.

Internal parameter (optional)

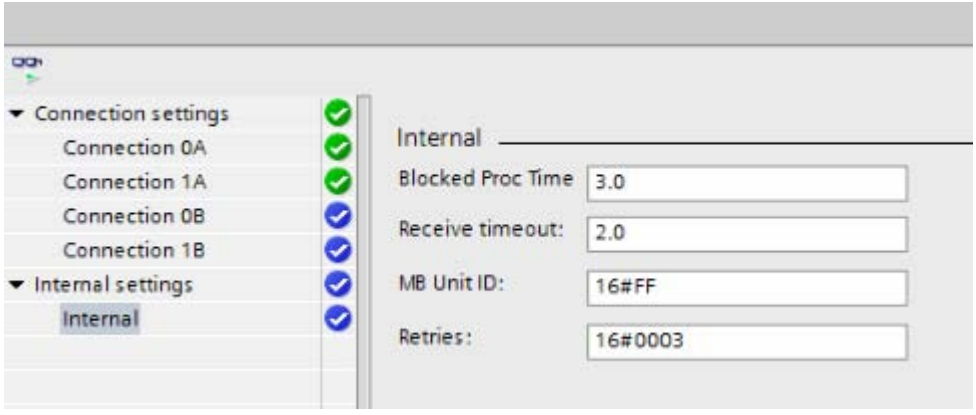

Tag	Data type	Start value	Description
Blocked Proc Time	REAL	3.0	Wait time in seconds before the static tag ACTIVE is reset if there is a blocked Modbus instance. This can, for example, occur if a client request is output and the execution of the client function aborts before the request was fully executed. The wait time must be between 0.5 s and 55 s.
Receive timeout	REAL	2.0	Time interval in seconds in which the "MB_RED_CLIENT" instruction waits for a response from the server. It must be between 0.5 s and 55 s.
MB_Unit_ID	BYTE	255	Modbus device detection: A Modbus TCP server is addressed using its IP address. For this reason, the MB_UNIT_ID parameter is not used in the case of Modbus TCP addressing. The MB_UNIT_ID parameter corresponds to the field of the slave address in the Modbus RTU protocol. If a Modbus/TCP server is used as a gateway for a Modbus RTU protocol, the slave device in the serial network can be identified using MB_UNIT_ID. The MB_UNIT_ID parameter would in this case forward the request to the correct Modbus RTU slave address. Please note that some Modbus/TCP devices may require the MB_UNIT_ID parameter for the initialization within a limited value range.
Retries	WORD	3	Number of send attempts made by the "MB_RED_CLIENT" instruction before it returns the error W#16#80C8.

Note

Tag MB_Transaction_ID

If the transaction ID in the answer of the Modbus TCP server does not match the transaction ID of the job from "MB_RED_CLIENT", the "MB_RED_CLIENT" instruction waits for the time period RCV_TIMEOUT * RETRIES for the answer of the Modbus TCP server with the correct transaction ID; once this time has expired, it returns the error W#16#80C8.

Licensing (S7-1200, S7-1500)

The "MB_RED_CLIENT" instruction is subject to a fee and must be licensed on each CPU individually. Licensing takes place in two steps:

- Reading out the IDENT_CODE and
- Entering the registration key REG_KEY.

Note

S7-1500R and S7-1500H

For an S7-R/H station only CPU1 is licensed. A replacement of CPU1 after licensing is therefore no longer possible.

Procedure

Reading out the IDENT_CODE

Proceed as follows to read out the IDENT_CODE:

1. Assign parameters to the "MB_RED_CLIENT" instruction in line with your requirements in a cyclic OB. Download the program to the CPU and set it to RUN.
2. Open the instance DB of the Modbus instruction and click the "Monitor all" button.

3. An 18-digit character string is displayed at the IDENT_CODE output.

	Name	Data type	Start value	Monitor value
1	Input			
2	REG_KEY	String[17]	''	''
3	USE_ALL_CONN	Bool	false	FALSE
4	REQ	Bool	false	FALSE
5	DISCONNECT	Bool	false	FALSE
6	MB_MODE	USInt	0	0
7	MB_DATA_ADDR	UDInt	0	0
8	MB_DATA_LEN	UInt	0	0
9	Output			
10	LICENSED	Bool	false	FALSE
11	IDENT_CODE	String[18]	''	'RTPCFIGDCDIIHJHAH4'
12	DONE	Bool	false	FALSE
13	BUSY	Bool	false	FALSE
14	ERROR	Bool	false	FALSE

4. Copy this string using copy/paste from the data block and paste it in the form that was sent to you by email after you ordered the product or is included on the CD.
5. Send the form to Customer support (https://support.industry.siemens.com/my/ww/en/requests/#createRequest) using a service request. You will then receive the registration key for your CPU.

Entering the registration key REG_KEY

The registration key REG_KEY must be specified at each "MB_RED_CLIENT" instruction. You should save the REG_KEY in a global data block from which all "MB_RED_CLIENT" instructions receive the necessary registration key.

Proceed as follows to enter the registration key REG_KEY:

- Insert a new global data block with a unique symbolic name, for example "License_DB", using "Add new block…".
- Create a REG_KEY parameter in this block with the data type STRING[17].

LICENSE_DB

	Name	Data type	Offset	Start value
1	Static			
2	REG_KEY	String[17]	0.0	''

Figure 4-14 REG_KEY in DB

- Copy the transmitted 17-digit registration key using copy/paste to the "Start value" column.
- In the cyclic OB, enter the name of the license DB and the name of the string, e.g. License_DB.REG_KEY, at the parameter REG_KEY of the "MB_RED_CLIENT" instruction.
- Download the modified blocks to the CPU. The registration key can be entered during runtime; a change from STOP -> RUN is not necessary.

The Modbus/TCP communication using the "MB_RED_CLIENT" instruction is now licensed for this CPU, the output bit LICENSED is TRUE.

Missing or incorrect licensing

If you enter an incorrect registration key or none at all, the ERROR-LED of the CPU flashes. In addition for S7-1200 and S7-1500, a cyclic entry is made in the diagnostics buffer regarding the missing license. Only the area length error is displayed for R and H stations.

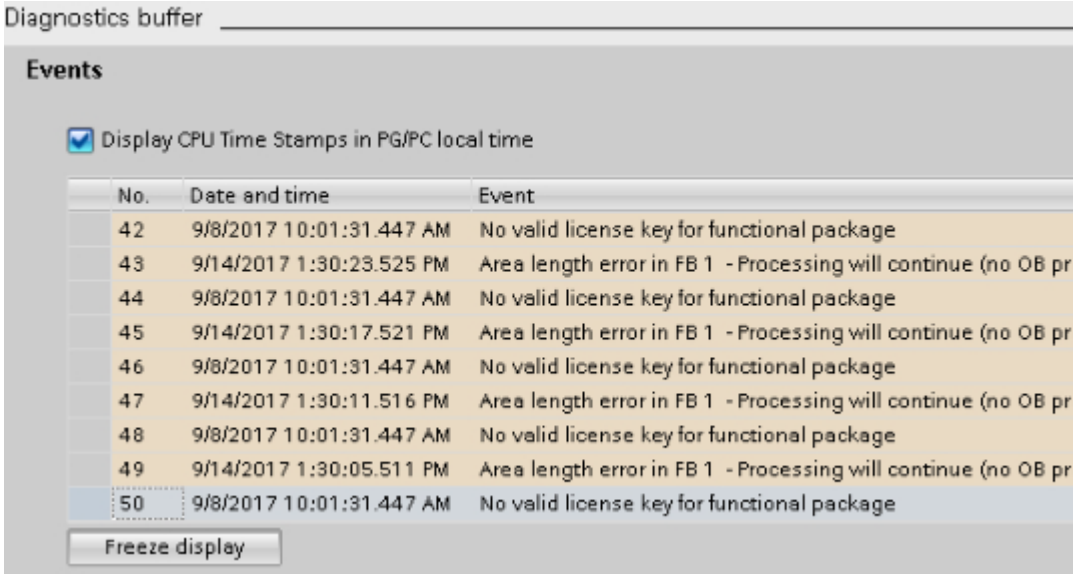

No.	Date and time	Event
42	9/8/2017 10:01:31.447 AM	No valid license key for functional package
43	9/14/2017 1:30:23.525 PM	Area length error in FB 1 - Processing will continue (no OB pr
44	9/8/2017 10:01:31.447 AM	No valid license key for functional package
45	9/14/2017 1:30:17.521 PM	Area length error in FB 1 - Processing will continue (no OB pr
46	9/8/2017 10:01:31.447 AM	No valid license key for functional package
47	9/14/2017 1:30:11.516 PM	Area length error in FB 1 - Processing will continue (no OB pr
48	9/8/2017 10:01:31.447 AM	No valid license key for functional package
49	9/14/2017 1:30:05.511 PM	Area length error in FB 1 - Processing will continue (no OB pr
50	9/8/2017 10:01:31.447 AM	No valid license key for functional package

In the case of a missing or incorrect registration key, the Modbus/TCP communication is processed but W#16#0A90 "No valid license available" is always displayed at the STATUS_x output. The output bit LICENSED is FALSE.

⚠ WARNING
If OB121 is missing in the controller in the S7-1500 (R/H), the CPU is set to STOP.

Input parameters MB_MODE, MB_DATA_ADDR, MB_DATA_LEN and MB_DATA_PTR (S7-1200, S7-1500)

Description

The combination of the parameters MB_MODE, MB_DATA_ADDR and MB_DATA_LEN defines the function code used in the current Modbus message:

- **MB_MODE** contains the information on whether to read or to write.
  Read: MB_MODE = 0, 101, 102, 103 and 104
  Write: MB_MODE = 1, 2, 105, 106, 115 and 116 (Note: With MB_MODE = 2, there is not distinction between Modbus functions 15 and 05 or between Modbus functions 16 and 06.)
- **MB_DATA_ADDR** contains the information on what is to be read or written, as well as address information from which the "MB_RED_CLIENT" instruction calculates the remote address.
- **MB_DATA_LEN** contains the number of values to be read/written.

The following table shows the relationship between the input parameters MB_MODE, MB_DATA_ADDR, MB_DATA_LEN of the "MB_RED_CLIENT" instruction and the Modbus function.

MB_MODE	MB_DATA_ADDR	MB_DATA_LEN	Modbus function	Function and data type
0	1 to 9,999	1 to 2,000	01	Read 1 to 2,000 output bits on the remote address 0 to 9,998
0	10,001 to 19,999	1 to 2,000	02	Read 1 to 2,000 input bits on the remote address 0 to 9,998
0	• 40,001 to 49,999 • 400,001 to 465,535	1 to 125	03	• Read 1 to 125 holding registers on the remote address 0 to 9,998 • Read 1 to 125 holding registers on the remote address 0 to 65,534
0	30,001 to 39,999	1 to 125	04	Read 1 to 125 input words on the remote address 0 to 9,998
1	1 to 9,999	1	05	Write 1 output bit on the remote address 0 to 9,998
1	• 40,001 to 49,999 • 400,001 to 465,535	1	06	• Write 1 holding register on the remote address 0 to 9,998 • Write 1 holding register on the remote address 0 to 65,534
1	1 to 9,999	2 to 1,968	15	Write 2 to 1,968 output bits on the remote address 0 to 9,998
1	• 40,001 to 49,999 • 400,001 to 465,535	2 to 123	16	• Write 2 to 123 holding registers on the remote address 0 to 9,998 • Write 2 to 123 holding registers on the remote address 0 to 65,534
2	1 to 9,999	1 to 1,968	15	Write 1 to 1,968 output bits on the remote address 0 to 9,998
2	• 40,001 to 49,999 • 400,001 to 465,535	1 to 123	16	• Write 1 to 123 holding registers on the remote address 0 to 9,998 • Write 1 to 123 holding registers on the remote address 0 to 65,534
11	The MB_DATA_ADDR and MB_DATA_LEN parameters are not evaluated when this function is executed.		11	Read status word and event counter of the server: • The status word reflects the processing status (0 - not processing, 0xFFFF - processing). • The event counter is incremented when the Modbus request was executed successfully. If an error occurred during execution of a Modbus function, a message is sent by the server but the event counter is not incremented.
80	-	1	08	Check the server status with the diagnostic code 0x0000 (return loop test - the server sends the request back): • 1 WORD per call
81	-	1	08	Reset the event counter of the server with the diagnostic code 0x000A: • 1 WORD per call

MB_MODE	MB_DATA_ADDR	MB_DATA_LEN	Modbus function	Function and data type
101	0 to 65,535	1 to 2,000	01	Read 1 to 2,000 output bits on the remote address 0 to 65,535
102	0 to 65,535	1 to 2,000	02	Read 1 to 2,000 input bits on the remote address 0 to 65,535
103	0 to 65,535	1 to 125	03	Read 1 to 125 holding registers on the remote address 0 to 65,535
104	0 to 65,535	1 to 125	04	Read 1 to 125 input words on the remote address 0 to 65,535
105	0 to 65,535	1	05	Write 1 output bit on the remote address 0 to 65,535
106	0 to 65,535	1	06	Write 1 holding register on the remote address 0 to 65,535
115	0 to 65,535	1 to 1,968	15	Write 1 to 1,968 output bits on the remote address 0 to 65,535
116	0 to 65,535	1 to 123	16	Write 1 to 123 holding registers on the remote address 0 to 65,535
3 to 10, 12 to 79, 82 to 100, 107 to 114, 117 to 255				Reserved

Example

Tag	Meaning
MB_MODE = 1 MB_DATA_ADDR = 1 MB_DATA_LEN = 1	1 output bit with function code 5 is written starting from the remote address 0.
MB_MODE = 1 MB_DATA_ADDR = 1 MB_DATA_LEN = 2	2 output bits with function code 15 are written starting from the remote address 0.
MB_MODE = 104 MB_DATA_ADDR = 17834 MB_DATA_LEN = 125	125 input words with function code 4 is read starting from the remote address 17.834.

MB_DATA_PTR

The MB_DATA_PTR parameter is a pointer to a data buffer for the data to be received from the Modbus server or to be sent to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

For a buffer in the memory area (M), use a pointer in the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

Depending on the memory area in which the data buffer is located, MB_DATA_PTR can reference different data structures:

- When you use a global DB with optimized access, MB_DATA_PTR can reference a tag with elementary data type or an array of elementary data types. The following data types are supported:

Data type	Length in bits
Bool	1
Byte, SInt, USInt, Char	8
Word, Int, WChar, UInt	16
DWord, DInt, UDInt, Real	32

  You can hereby use all supported data types for all Modbus functions. For example, MB_RED_CLIENT can also write a received bit in a tag of the byte type to a specified address without changing other bits in this byte. Therefore, it is not necessary to have an array of bits in order to execute bit-oriented functions.

- If you use a bit memory address area or a global DB with standard access as memory area, there is no longer any restriction to the elementary data types for MB_DATA_PTR; MB_DATA_PTR can then also reference complex data structures such as PLC data types (UDTs) and system data types (SDTs).

Note

Using a bit memory address area as data buffer

If you use a bit memory address area as data buffer for MB_DATA_PTR, you need to observe this variable. With S7-1500(R/H) CPUs it is 16 KB, with S7-1200 CPUs it is 8 KB.

Output parameters STATUS_x, RED_ERR_S7, RED_ERR_DEV and TOT_COM_ERR (S7-1200, S7-1500)

Parameter STATUS_x (general status information)

The error messages are displayed at the status outputs of the instruction "MB_RED_CLIENT".

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
0A90	The instruction "MB_RED_CLIENT" is not licensed. Additional information is available here: Licensing (Page 4150).
0AFF	The connection is not configured and is not used. The connection 0A must be configured.
7000	No job active and no connection established (REQ=0, DISCONNECT=1).
7001	Connection establishment triggered.
7002	Intermediate call. Connection is being established.
7003	Connection is being terminated.

STATUS* (W#16#)	Description
7004	Connection established and monitored. No job processing active.
7005	Data is being sent.
7006	Data is being received.
* The status codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Parameter STATUS_x (protocol error)

STATUS* (W#16#)	Description
80C8	No response of the server in the defined period. Check the connection to the Modbus server. This error is only reported on completion of the configured repeated attempts. If the "MB_RED_CLIENT" instruction does not receive an answer with the originally transferred transaction ID (see static tag MB_TRANSACTION_ID) within the defined period, this error code is output.
8380	Received Modbus frame has incorrect format or too few bytes were received.
8382	• The length of the Modbus frame in the frame header does not match the number of received bytes. • The number of bytes does not match the number of actually transmitted bytes (only functions 1-4). • The start address in the received frame does not match the saved start address (functions 5, 6, 15, 16). • The number of words does not match the number of actually transmitted words (functions 15 and 16).
8383	Error reading or writing data or access outside the address area of MB_DATA_PTR (Page 4152).
8384	• Invalid exception code received. • A different data value was received than was originally sent by the client (functions 5, 6 and 8). • Invalid status value received (function 11)
8385	• Diagnostics code not supported. • A different subfunction code was received than was originally sent by the client (function 8).
8386	Received function code does not match the one sent originally.
8387	The protocol ID of the Modbus TCP frame received by the server is not "0".
8388	The Modbus server sent a different data length than was processed. This error occurs only when using the Modbus functions 5, 6, 15 or 16.
* The status codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Parameter STATUS_x (parameter error)

STA-TUS* (W#16#)	Description	
80B6	Invalid connection type, only TCP connections are supported.	
80BB	The ActiveEstablished parameter has an invalid value. Only active connection establishment permitted for client (ActiveEstablished = TRUE).	
8188	The MB_MODE parameter has an invalid value.	
8189	Invalid addressing of data at the MB_DATA_ADDR parameter.	
818A	Invalid data length at the MB_DATA_LEN parameter.	

STA-TUS* (W#16#)	Description	
818B	The MB_DATA_PTR parameter has an invalid pointer. You should also check the values of the MB_DATA_ADDR (Page 4152) and MB_DATA_LEN parameters.	
818C	Timeout at parameter BLOCKED_PROC_TIMEOUT or RCV_TIMEOUT (see static tags of instruction). BLOCKED_PROC_TIMEOUT and RCV_TIMEOUT must be between 0.5 s and 55 s.	
8200	• A different Modbus request is currently being processed via the port. • Another instance of MB_RED_CLIENT with the same connection parameters is processing an existing Modbus request.	
* The status codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

Note

Error codes of internally used communication instructions

With the "MB_RED_CLIENT" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND", "TRCV", "T_DIAG" and "TRESET") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_RED_CLIENT" instruction. The error codes are displayed for the respective instruction under STATUS in the "Static" section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

Note

Communication error when sending or receiving data

If a communication error occurs when sending or receiving data (80C4 (Temporary communications error. The specified connection is temporarily down.), 80C5 (Remote partner closed connection actively.), 80A1 (The specified connection is disconnected or is not yet established.)), the existing connection is terminated.

This also means that you can see all STATUS values that are returned when the connection is terminated and that the STATUS code that caused the connection to be terminated is only output when the connection is terminated.

Example: If a temporary communication error occurs when data is received, the STATUS 7003 (ERROR=false) is output initially and then 80C4 (ERROR=true).

MB_RED_SERVER: Communicating via PROFINET as a MODBUS TCP server (S7-1200, S7-1500)

MB_RED_SERVER description (S7-1200, S7-1500)

Description

You can use this instruction to establish a connection between a 1200/1500 CPU and a device that supports the Modbus/TCP protocol.

The instruction "MB_RED_SERVER" communicates as a Modbus/TCP server over the PROFINET connection. The instruction ""MB_RED_SERVER"" processes connection requests of a Modbus/TCP client, receives and processes Modbus requests and sends responses.

The instruction "MB_RED_SERVER" can be used in the following CPUs:

	"MB_RED_SERVER" V1.0	"MB_RED_SERVER" V1.1
S7-1200	FW V4.2 and higher	FW V4.2 and higher
S7-1500	FW V2.5 and higher	FW V2.5 and higher
S7-1500R	--	FW V2.6 and higher
S7-1500H	--	FW V2.6 and higher

To use the instruction, you do not require an additional hardware module.

NOTICE
Security information Note that each client of the network is given read and write access to the process image inputs and outputs and to the data block or bit memory area defined by the Modbus holding register. The option is available to restrict access to an IP address to prevent unauthorized read and write operations. Note, however, that the shared address can also be used for unauthorized access.

Multiple server connections

The CPUs can:

- processing multiple server connections and
- accepting multiple connections from different clients simultaneously at one server port.

The maximum number of connections depends on the CPU being used and can be found in the technical specifications of the CPU. The total number of connections of one CPU, including those of the Modbus/TCP clients and server must not exceed the maximum number of supported connections.

In the case of server connections, remember the following rules:

- Each ""MB_RED_SERVER"" connection must use a unique instance DB.
- One unique connection/connection ID is required for each and every client that wants to connect to the server port.
- The connection IDs must be unique throughout the CPU.

For information concerning the redundancy setup, refer to: Operation and redundancy (Page 4162).

Parameters

The following table shows the parameters of the instruction "MB_RED_SERVER":

Parameters	Declaration	Data type	Description
REG_KEY (Page 4173)	Input	STRING[17]	Registration code for licensing The "MB_RED_SERVER" instruction must be licensed on each CPU individually.
DISCONNECT	Input	BOOL	The "MB_RED_SERVER" instruction is used to enter into a passive connection with a partner module. The server responds to a connection request from the IP addresses which are given in the connection descriptions as specified or unspecified. You can use this parameter to control when a connection request is accepted: • 0: A passive connection is established when there is no communications connection. • 1: Initialization of the connection termination. If the input is set, no additional client requests are being processed and termination of the connection is initiated. The value 0003 is output at the STATUS_x parameter after successful connection termination.
MB_HOLD_REG (Page 4175)	InOut	VARIANT	Pointer to the Modbus holding register of the "MB_RED_SERVER" instruction MB_HOLD_REG must always reference a memory area that is larger than two bytes. The holding register contains the values that may be accessed by a Modbus client using the Modbus functions 3 (read), 6 (write), 16 (multiple write) and 23 (reading and writing in one job).
LICENSED (Page 4173)	Output	BOOL	• 0: Instruction is not licensed • 1: Instruction is licensed
IDENT_CODE (Page 4173)	Output	STRING[18]	Identification for licensing. Use this string to request the registration code REG_KEY.
DR_NDR_0A	Output	BOOL	"Data Read" or "New Data Ready" to connection 0A: • 0: No new data • 1: New data read or written by the Modbus client
ERROR_0A	Output	BOOL	If an error occurs during a call of the "MB_RED_SERVER" instruction to connection 0A, the output of the ERROR_0A parameter is set to "1". Detailed information about the cause of the problem is indicated by the STATUS_0A parameter.

Parameters	Decla-ration	Data type	Description
STATUS_0A (Page 4176)	Output	WORD	Detailed status information of the instruction on connection 0A.
DR_NDR_1A	Output	BOOL	"Data Read" or "New Data Ready" to connection 1A: • 0: No new data • 1: New data read or written by the Modbus client
ERROR_1A	Output	BOOL	If an error occurs during a call of the "MB_RED_SERVER" instruction to connection 1A, the output of the ERROR_1A parameter is set to "1". Detailed information about the cause of the problem is indicated by the STATUS_1A parameter.
STATUS_1A (Page 4176)	Output	WORD	Detailed status information of the instruction on connection 1A.
DR_NDR_0B	Output	BOOL	"Data Read" or "New Data Ready" to connection 0B: • 0: No new data • 1: New data read or written by the Modbus client
ERROR_0B	Output	BOOL	If an error occurs during a call of the "MB_RED_SERVER" instruction to connection 0B, the output of the ERROR_0B parameter is set to "1". Detailed information about the cause of the problem is indicated by the STATUS_0B parameter.
STATUS_0B (Page 4176)	Output	WORD	Detailed status information of the instruction on connection 0B.
DR_NDR_1B	Output	BOOL	"Data Read" or "New Data Ready" to connection 1B: • 0: No new data • 1: New data read or written by the Modbus client
ERROR_1B	Output	BOOL	If an error occurs during a call of the "MB_RED_SERVER" instruction to connection 1B, the output of the ERROR_1B parameter is set to "1". Detailed information about the cause of the problem is indicated by the STATUS_1B parameter.
STATUS_1B (Page 4176)	Output	WORD	Detailed status information of the instruction on connection 1B.
RED_ERR_S7 (Page 4162)	Output	BOOL	• 0: No redundancy error in SIMATIC • 1: Redundancy error in SIMATIC
RED_ERR_DEV (Page 4162)	Output	BOOL	• 0: No redundancy error on side of link partner • 1: Redundancy error on side of link partner
TOT_COM_ERR (Page 4162)	Output	BOOL	• 0: At least one configured connection is established • 1: Complete loss of communication, all configured connections are terminated

Mapping of Modbus addresses to the process image

The "MB_RED_SERVER" instruction allows incoming Modbus functions (1, 2, 4, 5 and 15) direct read and write access to the process image inputs and outputs of the CPU (use of the data types BOOL and WORD).

For S7-1200-CPUs, the address space for the process image of the inputs and the process image of the outputs is 1 KB; it is 32 KB each for S7-1500-CPUs.

The following table shows the address space of the Modbus functions listed above.

Modbus function					
Function code	Function	Data area	Address space		
01	Read: Bits	Output	0	to	65.535
02	Read: Bits	Input	0	to	65.535
04	Read: WORD	Input	0	to	65.535
05	Write: Bit	Output	0	to	65.535
15	Write: Bits	Output	0	to	65.535

Incoming Modbus requests with the function codes 3, 6, 16 and 23 write or read the Modbus holding registers (you specify the holding register with the MB_HOLD_REG parameter or via Data_Area_Array).

Modbus functions

The following table lists all the Modbus functions that are supported by the "MB_RED_SERVER" instruction.

Function code	Description
01	Read output bits
02	Read input bits
03	Read a holding register
04	Read input words
05	Write an output bit
06	Write a holding register
08	Diagnostics function: • Echo test (subfunction 0x0000): The "MB_RED_SERVER" instruction receives a data word and returns this unchanged to the Modbus client. • Reset event counter (subfunction 0x000A): The ""MB_RED_SERVER"" instruction resets the following event counters: "Success_Count", "Xmt_Rcv_Count", "Exception_Count", "Server_Message_Count" and "Request_Count".
11	Diagnostics function: Fetch event counter of the communication The ""MB_RED_SERVER"" instruction uses an internal event counter for communication to record the number of successfully executed read and write requests sent to the Modbus server. The event counter is not incremented with the functions 8 or 11. The same holds true for requests that cause a communications error, for example, if a protocol error has occurred (e.g., the function code in the received Modbus request is not supported).
15	Write output bits
16	Write a holding register
23	Write a holding register and read a holding register with a request

Operation and redundancy (S7-1200, S7-1500)

Description

The communication nodes can be designed as standalone or redundant. If one of the partners is designed as standalone, we refer to it as single-sided redundancy. If both partners are designed redundantly, we refer to it as double-sided redundancy.

Port numbers for client and server

The Modbus client uses a port number starting at 2000. The Modbus server is usually addressed over the port number 502. Depending on the CPU, it is possible to configure port 502 for multiple connections (Multiport). If the local port 502 was configured for two or more connections, the requesting clients are randomly distributed to the existing server connections in the case of unspecified connections. The first client that wants to connect to the "MB_RED_SERVER" instruction is not automatically assigned the connection 0A. Once the assignment of the client requests to the server connections has taken place, the assignment remains intact during the frame exchange until the connection is terminated.

Single-sided redundancy

Description

One connection each must be configured for each connection between the communication partners. The connection points of the **SIMATIC S7** are referred to as **0** and **1**; the connection points of the **communication partner** are referred to as **A** and **B**.

The R-CPU or H-CPU 1 refers to the connection point 0, the R-CPU or H-CPU 2 refers to the connection point 1.

Configuration

If the S7 is designed redundantly, one connection is created from the S7 connection point 0 to junction A of the link partner and one connection from the S7 connection point 1 to junction A of the link partner.

- Connection from the S7 connection point **0** to the partner/node **A** => connection **0A**
- Connection from the S7 connection point **1** to the partner/node **A** => connection **1A**

The figure illustrates the connection designations.

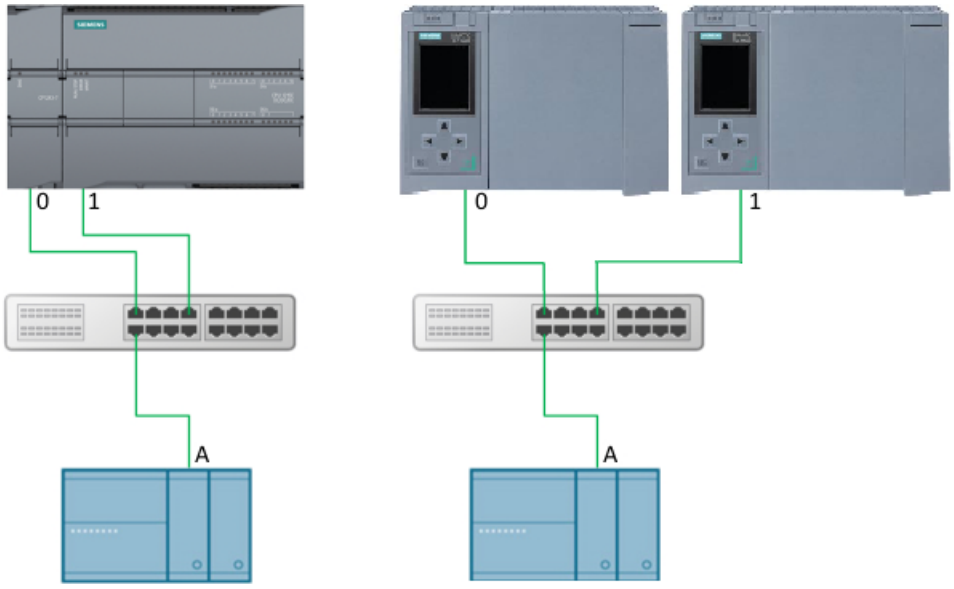

Figure 4-15 Modbus/TCP: Single-sided redundancy S7

If the S7 is designed as standalone and the link partner is designed redundantly, one connection is created from the S7 connection point 0 to junction A of the link partner and one connection from the S7 connection point 0 to junction B of the link partner.

- Connection from the S7 connection point **0** to the partner/node **A** => connection **0A**
- Connection from the S7 connection point **0** to the partner/node **B** => connection **0B**

The figure illustrates the connection designations.

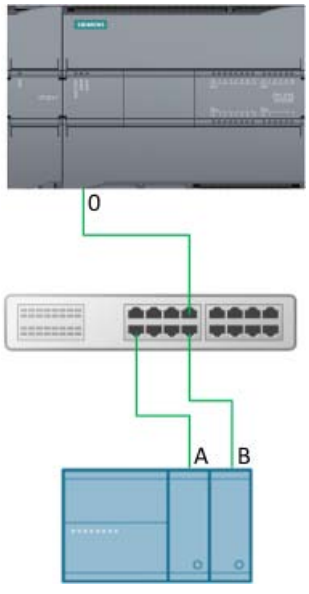

Figure 4-16 Modbus/TCP: Single-sided redundancy partner

Double-sided redundancy

Description

One connection each must be configured for each connection between the communication partners. The connection points of the **SIMATIC S7** are referred to as **0** and **1**; the connection points of the **communication partner** are referred to as **A** and **B**.

The R-CPU or H-CPU 1 refers to the connection point 0, the R-CPU or H-CPU 2 refers to the connection point 1.

Configuration

In the case of double-sided redundancy, two connections are created from connection point 0 and two connections from connection point 1 of the S7 to the junctions A and B of the link partner.

- Connection from the S7 connection point **0** to the partner/node **A** => connection **0A**
- Connection from the S7 connection point **1** to the partner/node **A** => connection **1A**
- Connection from the S7 connection point **0** to the partner/node **B** => connection **0B**
- Connection from the S7 connection point **1** to the partner/node **B** => connection **1B**

The figure illustrates the connection designations.

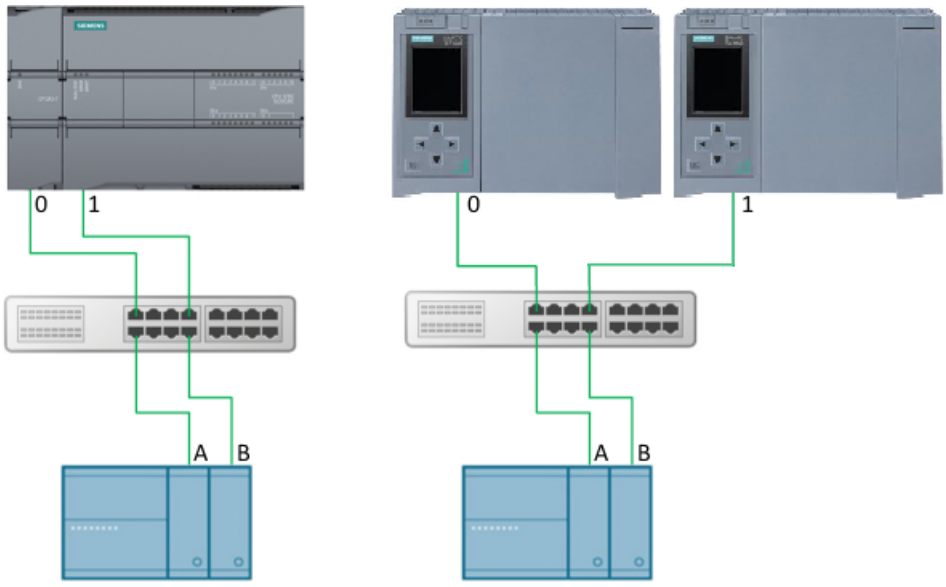

Figure 4-17 Modbus/TCP: Double-sided redundancy

Frame processing

Frames can be received via all configured connections. The client can send frames either via one connection or via all connections. If a frame was received on a connection, the status is displayed at the corresponding output DR_NDR_x or ERROR_x. Each connection runs independently and has no influence on the display of the other connections.

Redundancy outputs RED_ERR_S7, RED_ERR_DEV and TOT_COM_ERR

Description

The redundancy bits RED_ERR_S7, RED_ERR_DEV and TOT_COM_ERR are set based on the states of the status outputs.

Display of the interrupt bits for redundancy setup on both sides

Number of faulty connections	STATUS_0A	STATUS_0B	STATUS_1A	STATUS_1B	RED_ERR_S7	RED_ERR_DEV	TOT_COM_ERR
0	okay	okay	okay	okay	FALSE	FALSE	FALSE
1	okay	okay	okay	Error	FALSE	FALSE	FALSE
	okay	okay	Error	okay	FALSE	FALSE	FALSE
	okay	Error	okay	okay	FALSE	FALSE	FALSE
	Error	okay	okay	okay	FALSE	FALSE	FALSE
2	okay	okay	Error	Error	TRUE	FALSE	FALSE
	okay	Error	okay	Error	FALSE	TRUE	FALSE
	Error	okay	okay	Error	FALSE	FALSE	FALSE
	okay	Error	Error	okay	FALSE	FALSE	FALSE
	Error	okay	Error	okay	FALSE	TRUE	FALSE
	Error	Error	okay	okay	TRUE	FALSE	FALSE
3	Error	Error	Error	okay	TRUE	TRUE	FALSE
	Error	Error	okay	Error	TRUE	TRUE	FALSE
	Error	okay	Error	Error	TRUE	TRUE	FALSE
	okay	Error	Error	Error	TRUE	TRUE	FALSE
4	Error	Error	Error	Error	TRUE	TRUE	TRUE

Display of the interrupt bits for redundancy setup on one side

Number of faulty connections	STATUS_0A	STATUS_0B	STATUS_1A	STATUS_1B	RED_ERR_S7	RED_ERR_DEV	TOT_COM_ERR
0	okay	0AFF	okay	0AFF	FALSE	FALSE	FALSE
1	okay	0AFF	Error	0AFF	TRUE	TRUE	FALSE
	Error	0AFF	okay	0AFF	TRUE	TRUE	FALSE
2	Error	0AFF	Error	0AFF	TRUE	TRUE	TRUE

Parameter assignment (S7-1200, S7-1500)

Description

The instruction "MB_RED_SERVER" **V1.0** and **V1.1** can be used for S7-1200 and for S7-1500.

S7-1200

The connections are implemented over the local interface of the CPU or CM/CP. The connections are configured and established using the structure TCON_IP_V4.

S7-1500

The connections are implemented over the local interface of the CPU or CM/CP. Configured connections over the structure TCON_IP_V4 as well as configured connections over the structure TCON_Configured are possible.

The instruction "MB_RED_SERVER" **V1.1** can be used for S7-1200R and for S7-1500H.

S7-1500R and S7-1500H

The connections are implemented over the local interface of the CPU or CM/CP. The connections are configured and established using the structure TCON_IP_V4.

Configuration of "MB_RED_SERVER"

The following settings are made with the help of the configuration dialog of the instruction "MB_RED_SERVER":

- Connection parameters for the connections 0A, 1A, 0B and 1B (Redundancy configuration see Operation and Redundancy (Page 4162))
- Internal parameter (optional)

The configuration dialog can be opened with the instruction or via the technology objects.

Connection parameters

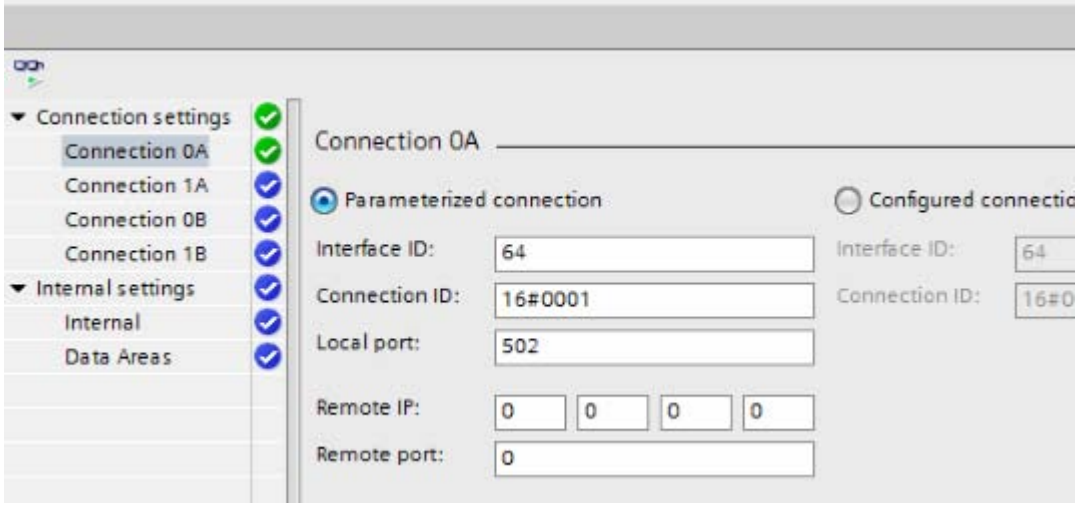

Figure 4-18 Modbus/TCP: configured server connection

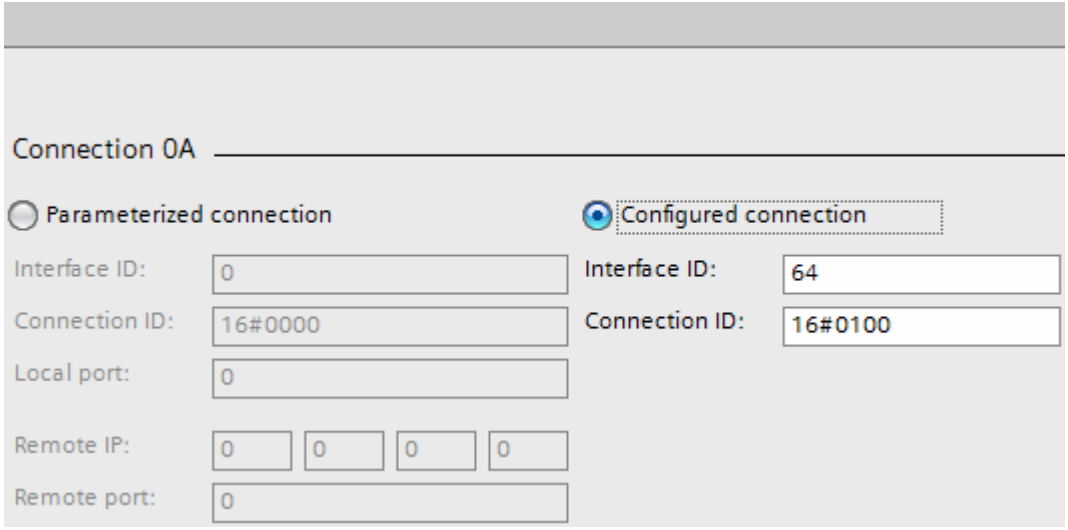

Figure 4-19 Modbus/TCP: configured server connection

Tag	Start value	Description
Configured connections		
Interface ID	64	HW identifier of the PN interface used
Connection ID	16#0000	Connection IDs for the connections used. These connection IDs must be unique throughout the CPU.
Local port	502	Local port number of the server block. The default port for Modbus/TCP server is 502.
Remote IP	0.0.0.0	Remote IP address of the client. By default no IP address is entered for the client.
Remote port	0	Remote port number of the client. By default no port number is entered for the client.
Configured connections		
Interface ID	64	HW identifier of the PN interface used
Connection ID	16#0000	Connection IDs for the connections used. These connections are configured in the network view.

Internal parameter (optional)

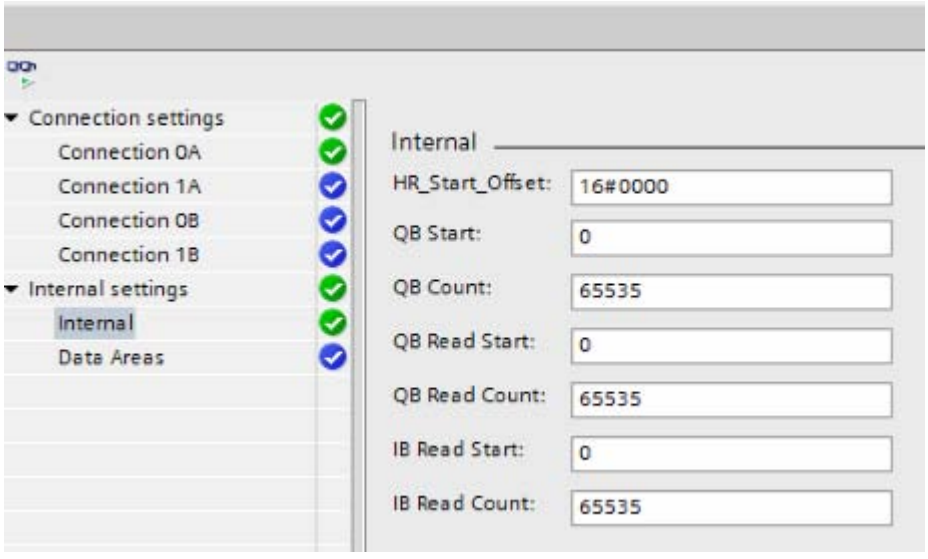

Tag	Data type	Start value	Description
HR_Start_Offset	WORD	0	Assign the start address of the Modbus holding register.
QB_Start	UINT	0	Start address of the permitted addressing range of the outputs that can be written (bytes 0 to 65535)
QB_Count	UINT	0	Number of output bytes that can be written by the Modbus master. *Example:* QB_Start=0 and QB_Count=10: Output bytes 0 to 9 can be written. QB_Count=0: No output byte can be written.
QB_Read_Start	UINT	0	Start address of the permitted addressing range of the outputs that can be read (bytes 0 to 65535)
QB_Read_Count	UINT	0	Number of output bytes that can be read by the Modbus master. *Example:* QB_Read_Start=0 and QB_Read_Count=10: Output bytes 0 to 9 can be read. QB_Read_Count=0: No output byte can be read.
IB_Read_Start	UINT	0	Start address of the permitted addressing range of the inputs that can be read (bytes 0 to 65535)
IB_Read_Count	UINT	0	Number of input bytes that can be read by the Modbus master. *Example:* IB_Read_Start=0 and IB_Read_Count=10: Input bytes 0 to 9 can be read. IB_Read_Count=0: No input byte can be read.
Data_Area_Array	ARRAY [1..8]		
data_type	UINT	0	Data Type: 0 to 4

Tag	Data type	Start value	Description
db	UINT	0	Data block number
start	UINT	0	First Modbus address in the data block
length	UINT	0	Number of Modbus values in the data block

Addressing via static tag HR_Start_Offset

The addresses of the Modbus holding register start at 0.

Example: The holding register begins at MW100, and has a length of 100 WORD.

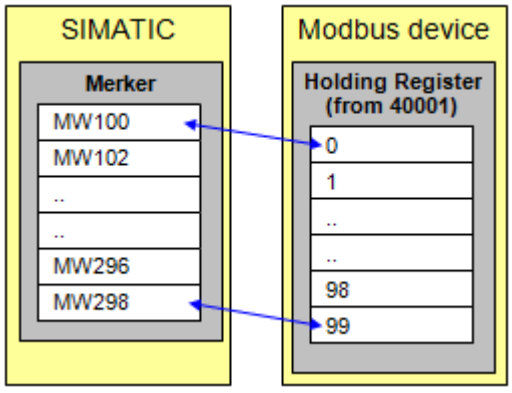

Figure 4-20 Modbus/TCP: HR_Start_Offset = 0

You can define the tag HR_Start_Offset so that the Modbus holding register has a start address other than 0.

Example: An offset value 20 in the parameter HR_Start_Offset means that the start address of the holding register is moved from 0 to 20. This causes an error whenever the holding register is addressed below the address 20 and above the address 119.

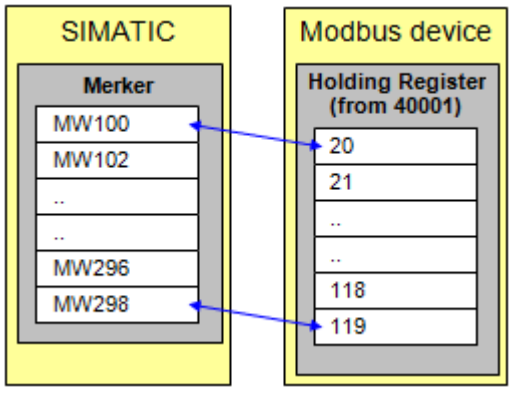

Figure 4-21 Modbus/TCP: HR_Start_Offset = 20

Data_Area_Array [1..8]

Description

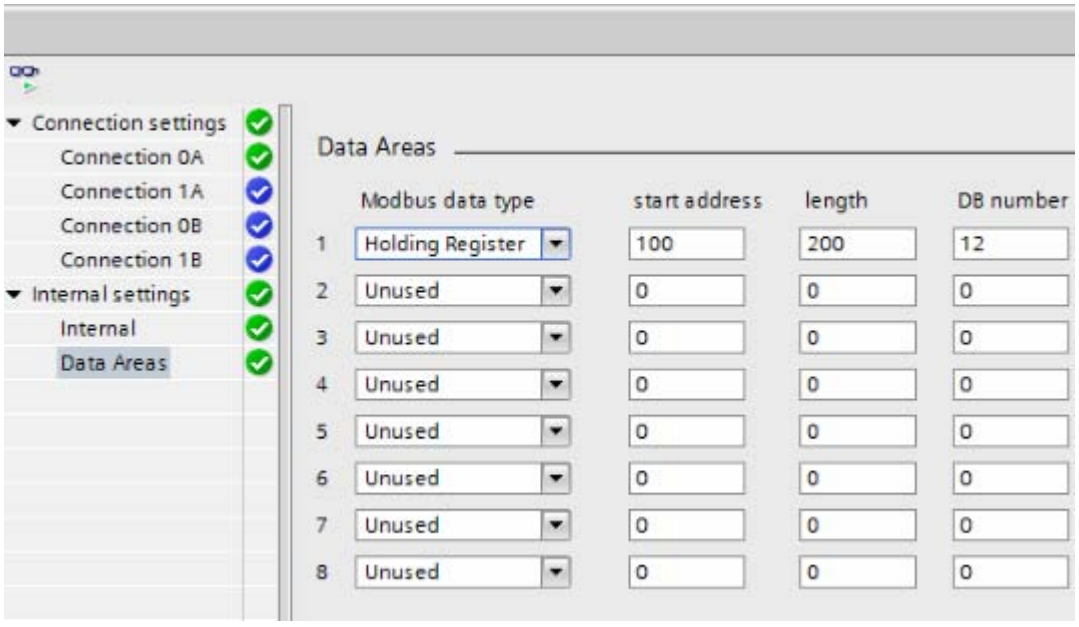

There are eight data areas available for mapping the MODBUS addresses in the SIMATIC S7 memory. If the first data area is defined, the parameter MB_HOLD_REG is not evaluated. Instead, the Modbus register and bits are written in the data blocks or read from them depending on the job type. These values can be further processed in the subsequent execution of the program.

You can only read from one DB or write to one DB with any job. Access to registers or bit values that are located in several DBs, even if the numbers are in a sequence without gaps, are to be divided into two jobs. Keep this in mind during configuration. It is possible to map more Modbus areas (registers or bit values) in one data block than can be processed with one frame.

data_type

The data_type parameter specifies which MODBUS data types are mapped in this data block. If the value 0 is entered in data_type, the corresponding data area is not used. If multiple Data_Area are to be used, these must be defined one after the other. All entries after a data_type = 0 are not processed.

Identifi-er	Data type	Description
0	Area not used	
1	Output bits (Coils)	Bit
2	Input bits (Inputs)	Bit
3	Holding register	Word
4	Input words (Input Register)	Word

db

The db parameter specifies the data block which maps the MODBUS registers or bit values defined below. The DB number 0 is not permitted because it is reserved for the system.

start, length

start specifies the first Modbus address which is mapped in data word 0 of the DB. The parameter length defines the length of how many MODBUS addresses are mapped in the data block. The defined data areas must not overlap. The length parameter must not be 0.

Example: Address mapping with Data_Area_Array

Data area 1	data_type	3: Holding register
	db	11
	start	0
	length	500
Data area 2	data_type	3: Holding register
	db	12
	start	720
	length	181
Data area 3	data_type	4: Input words
	db	13
	start	720
	length	281
Data area 4	data_type	1: Output bits
	db	14
	start	640
	length	611
Data area 5	data_type	2: Input bit
	db	15
	start	1700
	length	601
Data area 6	data_type	1: Output bits
	db	16
	start	1700
	length	601
Data area 7	data_type	Not used
	db	0
	start	0
	length	0
Data area 8	data_type	Not used
	db	0
	start	0
	length	0

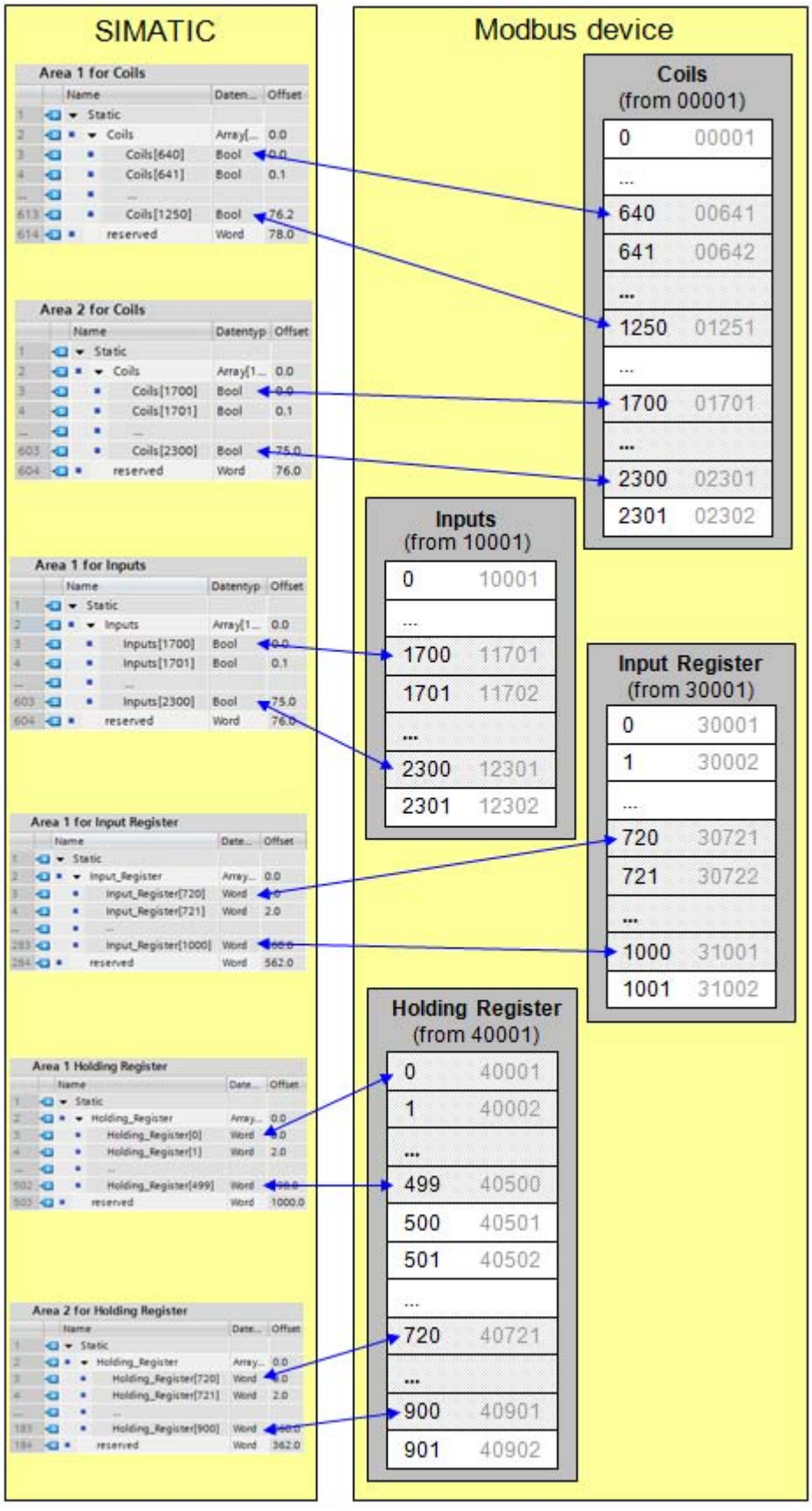
SIMATIC
Area 1 for Coils
Name Daten... Offset
1 Static
2 Coils Array[... 0.0
3 Coils[640] Bool 0.0
4 Coils[641] Bool 0.1
... ...
613 Coils[1250] Bool 76.2
614 reserved Word 78.0
Area 2 for Coils
Name Datentyp Offset
1 Static
2 Coils Array[1... 0.0
3 Coils[1700] Bool 0.0
4 Coils[1701] Bool 0.1
... ...
603 Coils[2300] Bool 75.0
604 reserved Word 76.0
Area 1 for Inputs
Name Datentyp Offset
1 Static
2 Inputs Array[1... 0.0
3 Inputs[1700] Bool 0.0
4 Inputs[1701] Bool 0.1
... ...
603 Inputs[2300] Bool 75.0
604 reserved Word 76.0
Area 1 for Input Register
Name Date... Offset
1 Static
2 Input_Register Array... 0.0
3 Input_Register[720] Word 0.0
4 Input_Register[721] Word 2.0
... ...
283 Input_Register[1000] Word 560.0
284 reserved Word 562.0
Area 1 Holding Register
Name Date... Offset
1 Static
2 Holding_Register Array... 0.0
3 Holding_Register[0] Word 0.0
4 Holding_Register[1] Word 2.0
... ...
502 Holding_Register[499] Word 998.0
503 reserved Word 1000.0
Area 2 for Holding Register
Name Date... Offset
1 Static
2 Holding_Register Array... 0.0
3 Holding_Register[720] Word 0.0
4 Holding_Register[721] Word 2.0
... ...
183 Holding_Register[900] Word 360.0
184 reserved Word 362.0
Modbus device
Coils
(from 00001)
0 00001
...
640 00641
641 00642
...
1250 01251
...
1700 01701
...
2300 02301
2301 02302
Inputs
(from 10001)
0 10001
...
1700 11701
1701 11702
...
2300 12301
2301 12302
Input Register
(from 30001)
0 30001
1 30002
...
720 30721
721 30722
...
1000 31001
1001 31002
Holding Register
(from 40001)
0 40001
1 40002
...
499 40500
500 40501
501 40502
...
720 40721
...
900 40901
901 40902

Licensing (S7-1200, S7-1500)

The instruction "MB_RED_SERVER" is subject to a fee and must be licensed on each CPU individually. Licensing takes place in two steps:

- Reading out the IDENT_CODE and
- Entering the registration key REG_KEY.

Note

S7-1500R and S7-1500H

For an S7-R/H station only CPU1 is licensed. A replacement of CPU1 after licensing is therefore no longer possible.

Procedure

Reading out the IDENT_CODE

Proceed as follows to read out the IDENT_CODE:

1. Assign parameters to the "MB_RED_SERVER" instruction in line with your requirements in a cyclic OB. Download the program to the CPU and set it to RUN.
2. Open the instance DB of the Modbus instruction and click the "Monitor all" button.
3. An 18-digit character string is displayed at the IDENT_CODE output.

		Name	Data type	Start value	Monitor value
1		Input			
2		REG_KEY	String[17]	''	''
3		USE_ALL_CONN	Bool	false	FALSE
4		REQ	Bool	false	FALSE
5		DISCONNECT	Bool	false	FALSE
6		MB_MODE	USInt	0	0
7		MB_DATA_ADDR	UDInt	0	0
8		MB_DATA_LEN	UInt	0	0
9		Output			
10		LICENSED	Bool	false	FALSE
11		IDENT_CODE	String[18]	''	'RTPCFIGDCDIIHJHAH4'
12		DONE	Bool	false	FALSE
13		BUSY	Bool	false	FALSE
14		ERROR	Bool	false	FALSE

4. Copy this string using copy/paste from the data block and paste it in the form that was sent to you by email after you ordered the product or is included on the CD.
5. Send the form to Customer support (https://support.industry.siemens.com/my/ww/en/requests/#createRequest) using a service request. You will then receive the registration key for your CPU.

Entering the registration key REG_KEY

The registration key REG_KEY must be specified at each "MB_RED_SERVER" instruction. You should save the REG_KEY in a global data block from which all "MB_RED_SERVER" instructions receive the necessary registration key.

Proceed as follows to enter the registration key REG_KEY:

- Insert a new global data block with a unique symbolic name, for example "License_DB", using "Add new block…".
- Create a REG_KEY parameter in this block with the data type STRING[17].

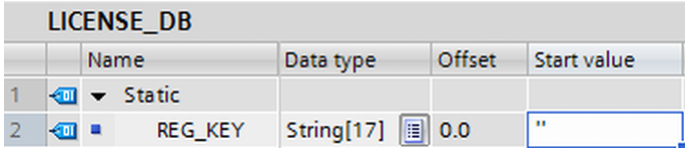

LICENSE_DB

		Name	Data type	Offset	Start value
1		Static			
2		REG_KEY	String[17]	0.0	''

Figure 4-22 REG_KEY in DB

- Copy the transmitted 17-digit registration key using copy/paste to the "Start value" column.
- In the cyclic OB, enter the name of the license DB and the name of the string, e.g. License_DB.REG_KEY, at the parameter REG_KEY of the ""MB_RED_SERVER"" instruction.
- Download the modified blocks to the CPU. The registration key can be entered during runtime; a change from STOP -> RUN is not necessary.

The Modbus/TCP communication using the "MB_RED_SERVER" instruction is now licensed for this CPU, the output bit LICENSED is TRUE.

Missing or incorrect licensing

If you enter an incorrect registration key or none at all, the ERROR-LED of the CPU flashes. In addition for S7-1200 and S7-1500, a cyclic entry is made in the diagnostics buffer regarding the missing license. Only the area length error is displayed for R and H stations.

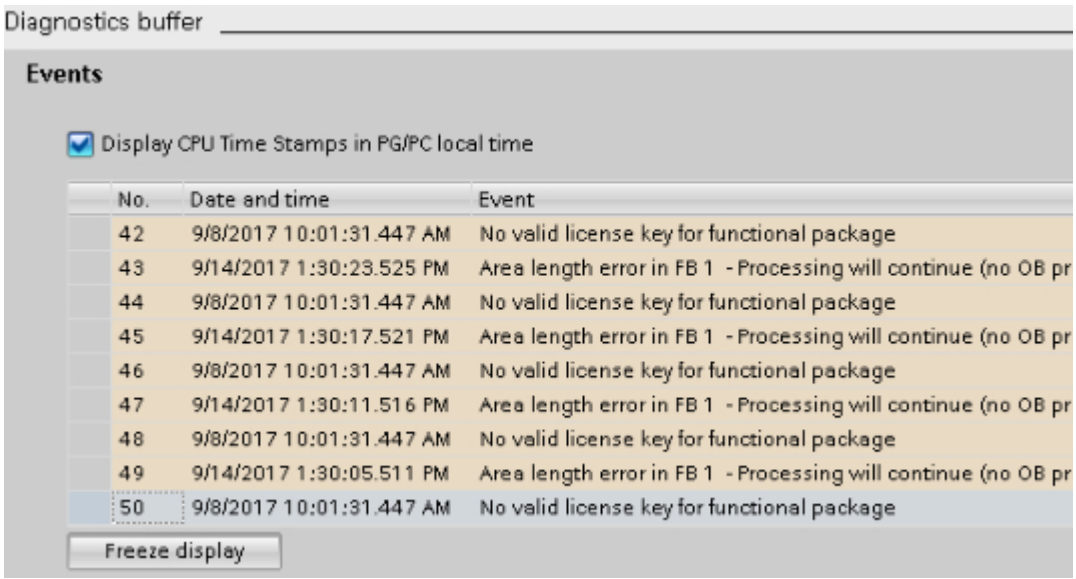

Diagnostics buffer

Events

Display CPU Time Stamps in PG/PC local time

No.	Date and time	Event
42	9/8/2017 10:01:31.447 AM	No valid license key for functional package
43	9/14/2017 1:30:23.525 PM	Area length error in FB 1 - Processing will continue (no OB pr
44	9/8/2017 10:01:31.447 AM	No valid license key for functional package
45	9/14/2017 1:30:17.521 PM	Area length error in FB 1 - Processing will continue (no OB pr
46	9/8/2017 10:01:31.447 AM	No valid license key for functional package
47	9/14/2017 1:30:11.516 PM	Area length error in FB 1 - Processing will continue (no OB pr
48	9/8/2017 10:01:31.447 AM	No valid license key for functional package
49	9/14/2017 1:30:05.511 PM	Area length error in FB 1 - Processing will continue (no OB pr
50	9/8/2017 10:01:31.447 AM	No valid license key for functional package

Freeze display

In the case of a missing or incorrect registration key, the Modbus/TCP communication is processed but W#16#0A90 "No valid license available" is always displayed at the STATUS_x output. The output bit LICENSED is FALSE.

⚠ WARNING
If OB121 is missing in the controller in the S7-1500 (R/H), the CPU is set to STOP.

MB_HOLD_REG input parameter (S7-1200, S7-1500)

Description

The MB_HOLD_REG parameter is a pointer to a data buffer for storing the data to which a Modbus client has read or write access. You can use a global data block (D) or bit memory (M) as memory area.

- The high limit for the number of addresses in the data block (D) is determined by the maximum size of a DB of your CPU.
- The high limit for the number of bit memories (M) is determined by the maximum bit memory area of your CPU.

The following figures show some examples of mapping Modbus addresses to the holding register for the Modbus functions 3 (read multiple WORD), 6 (write one WORD), 16 (write multiple WORD) and 23 (write and read multiple WORD).

MB_HOLD_REG: Data block with offset 0

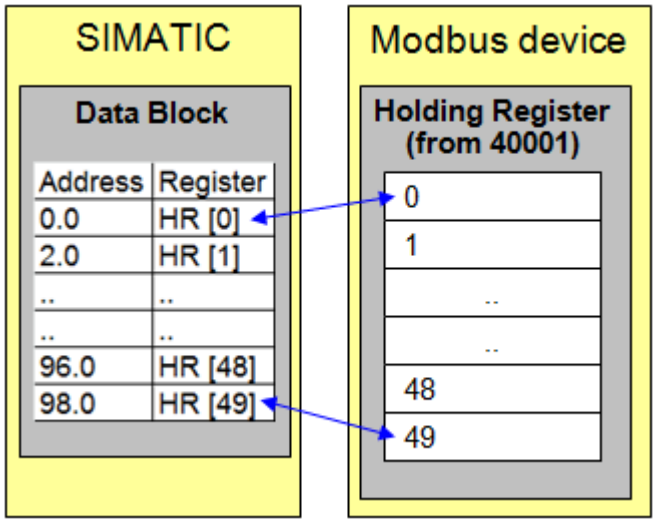

MB_HOLD_REG: Data block with offset 60

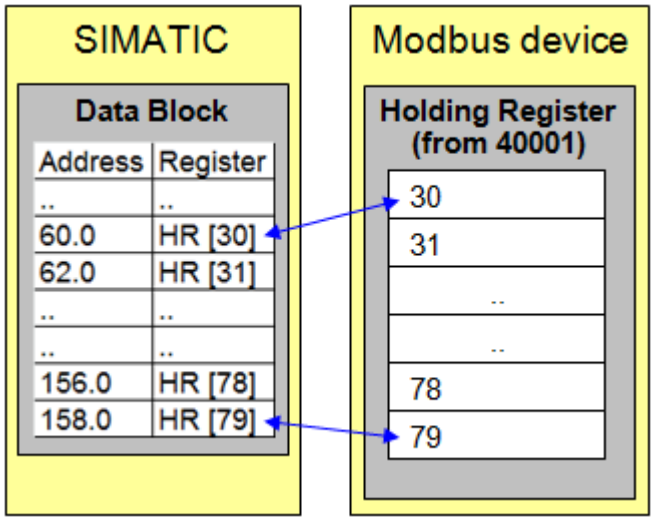

Data_Area_Array [1..8]

To use the optional parameters Data_Area_Array [1..8] see Parameter assignment (Page 4166).

Output parameters ERROR_x, STATUS_x, RED_ERR_S7, RED_ERR_DEV and TOT_COM_ERR (S7-1200, S7-1500)

Parameter STATUS_x (general status information)

The error messages are displayed at the status outputs of the instruction "MB_RED_SERVER".

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
0A90	The instruction "MB_RED_SERVER" is not licensed. Additional information is available here: Licensing (Page 4173).
0AFF	The connection is not configured and is not used. The connection 0A must be configured.
7000	No call active and no connection established (REQ=0, DISCONNECT=1).
7001	First call. Connection establishment triggered.
7002	Intermediate call. Connection is being established.
7003	Connection is being terminated.
7005	Data is being sent.
7006	Data is being received.
* The status codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Parameter STATUS_x (protocol error)

STATUS* (W#16#)	Error code in the error message from "MB_RED_SERVER" (B#16#)	Description
8380	-	Received Modbus frame has incorrect format or too few bytes were received.
8381	01	Function code is not supported.
8382	03	Error in data length: • Invalid length specification in received Modbus frame • The frame length entered in the header of the Modbus frame does not match the number of actually received bytes. • The number of bytes entered in the header of the Modbus frame does not match the number of actually received bytes (functions 15 and 16).
8383	02	Error in data address or access outside the address area of the holding register (MB_HOLD_REG (Page 4175) parameter).
8384	03	Error in the data value (function 05)
8385	03	Diagnostic code is not supported (only with function 08).
* The status codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

Parameter STATUS_x (parameter error)

STATUS* (W#16#)	Description
80BB	The ActiveEstablished parameter has an invalid value Only passive connection establishment permitted for server (active_established = FALSE).
8187	The MB_HOLD_REG parameter has an invalid pointer. Data area is too small.
8389	Invalid data area definition: • Invalid data_type value • DB number invalid or does not exist: – Invalid db value – DB number does not exist – DB number is already used by another data area – DB with optimized access – DB is not located in the work memory • Invalid length value • Overlapping of MODBUS address ranges that belong to the same MODBUS data type
* The status codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Note

Error codes of internally used communication instructions

With the "MB_RED_SERVER"" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND", "TRCV", "T_DIAG" and "T_RESET") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_RED_SERVER" instruction. The error codes are displayed for the respective instruction under STATUS in the "Static" section of the individual instances.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

Note

Communication error when sending or receiving data

If a communication error occurs when sending or receiving data (80C4 (Temporary communication error. The specified connection is temporarily terminated.), 80C5 (The remote partner has actively terminated the connection.), 80A1 (The specified connection is interrupted or has not been established yet.)), the existing connection is terminated.

This also means that you can see all STATUS values that are returned when the connection is terminated and that the STATUS code that caused the connection to be terminated is only output when the connection is terminated.

Example: If a temporary communication error occurs when data is received, the STATUS 7003 (ERROR=false) is output initially and then 80C4 (ERROR=true).

4.1.6.6 TeleService (S7-1200)

TM_MAIL: Transfer email (S7-1200)

Description of TM_MAIL (S7-1200)

Description

The "TM_MAIL" instruction works asynchronously, in other words, its execution extends over multiple calls. You must specify an instance when you call the instruction "TM_MAIL". The attribute "retentive" may not be set in the instance. This attribute ensures that the instance is initialized when the CPU switches from STOP to RUN and that a new e-mail send job can be triggered afterwards.

You start the sending of an e-mail with an edge change from "0" to "1" for the REQ parameter. The job status is indicated by the output parameters BUSY, "DONE", "ERROR", "STATUS" and "SFC_STATUS". "SFC_STATUS" corresponds in this case to the "STATUS" output parameter of the called communication blocks.

The output parameters DONE, ERROR, STATUS, and SFC_STATUS are each displayed for only one cycle if the status of the BUSY output parameter changes from "1" to "0". The following table shows the relationship between BUSY, DONE, and ERROR. Using this table, you can determine the current status of the instruction "TM_MAILand when the sending of the e-mail is complete.

DONE	BUSY	ERROR	Description
0	1	0	The job is being processed.
1	0	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS and SFC_STATUS parameters.
0	0	0	The "TM_MAIL" instruction was not assigned a (new) job.

If the CPU changes to STOP mode while "TM_MAIL" is active, the communication connection to the mail server aborts. The communication connection to the mail server will also be lost if communication problems occur on the Industrial Ethernet bus. In this case, the transfer of the e-mail will be interrupted and it will not reach its recipient.

NOTICE

Changing user programs

You can change the parts of your user program that directly affect calls of "TM_MAIL" only:

- when the CPU is in "STOP" mode or
- when no mail is being sent (REQ = 0 and BUSY = 0).

This relates, in particular, to deleting and replacing program blocks that contain "TM_MAIL" calls or calls for the instance of "TM_MAIL".

Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status for the TPC/IP communication functions via Industrial Ethernet.

A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The ADDR_MAIL_SERVER input parameter of the instruction is taken from the "TM_MAIL" instruction again each time the sending of an e-mail is triggered. If a change is made during operation, the "new" value only becomes effective once an e-mail is triggered again.

In contrast, the WATCH_DOG_TIME, TO_S, CC, FROM, SUBJECT, TEXT, ATTACHMENT, and, if applicable, USERNAME and PASSWORD parameters are taken from the "TM_MAIL" instruction while it is running, which means that they may only be changed after the job is complete (BUSY = 0)

Setting the parameters of the TS Adapter IE

On the TS Adapter IE, you need to specify parameters for outgoing calls in such a way as to enable the TS Adapter IE to establish a connection to the dial-up server of your service provider.

If you set "When required" for connection establishment, the connection will only be established when an e-mail needs to be sent.

Connection establishment can take longer (approx. one minute) for an analog modem connection. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection.

Parameter

The following table shows the parameters of the "TM_MAIL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter REQUEST: Activates the sending of an e-mail on a rising edge.
ID	Input	CONN_OUC (Word)	D, L or constant	Reference to the connection to be established. See ID parameter of the TCON (Page 3538), TDISCON (Page 3548), TSEND (Page 3553), and TRCV (Page 3557) instructions. A number that is not used for any additional instances of this instruction in the user program must be entered here.
TO_S (Page 4182)	Input	STRING	D	Input parameter for receiver addresses: STRING with a maximum length of 240 characters (see example call).
CC (Page 4182)	Input	STRING	D	Input parameter for CC recipient addresses (optional): STRING with a maximum length of 240 characters (see example call). If an empty string is assigned here, the e-mail is not sent to a CC recipient.
SUBJECT	Input	STRING	D	Input parameter for subject of the e-mail: STRING with a maximum length of 240 characters.
TEXT	Input	STRING	D	Input parameter for text of the e-mail (optional): Reference to a data string with a maximum length of 240 characters. If an empty string is assigned at this parameter, the e-mail is sent without text.
ATTACHMENT	Input	VARIANT	I, Q, M, D, L	Input parameter for e-mail attachment (optional): Reference to a byte/word/double word field with a maximum length of 65534 bytes. The specified attachment must be in the file format BIN. If no value is assigned, the e-mail is sent without an attachment.
DONE	Output	BOOL	I, Q, M, D, L	• DONE = 0: Job not yet started or still executing. • DONE = 1: Job executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: Sending of an e-mail is not yet complete. • BUSY = 0: Processing of "TM_MAIL" was terminated
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during processing. STATUS and SFC_STATUS supply detailed information about the type of error.

Parameter	Declaration	Data type	Memory area	Description
STATUS (Page 4183)	Output	WORD	I, Q, M, D, L	Output/status parameter STATUS: Return value or error information of the "TM_MAIL" instruction.
ADDR_MAIL_SERVER	Static*	DWORD	I, Q, M, D, L	IP address input parameter of the mail server: Specify as a data word in HEX format, for example: IP address = 192.168.0.200. ADDR_MAIL_SERVER = DW#16#C0A800C8, where: • 192 = 16#C0, • 168 =16#A8 • 0 = 16#00 and • 200 = 16#C8.
WATCH_DOG_TIME	Static*	TIME	I, Q, M, D, L	Input parameter for max. period of time: The "TM_MAIL" instruction should establish a connection within the period specified by WATCH_DOG_TIME. The block is exited with an error if this period is exceeded. The time before the block is exited and the error is output can exceed the WATCH_DOG_TIME because connection termination also takes a certain amount of time. To begin with, you should set a period of 2 minutes. If you have an ISDN telephone connection, a significantly shorter period can be selected.
USERNAME	Static*	STRING	D	Input parameter for user name: STRING with a maximum length of 180 characters. A user name is an absolute requirement for authentication.
PASSWORD	Static*	STRING	D	Input parameter for password: STRING with a maximum length of 180 characters. A password is an absolute requirement for authentication.
FROM (Page 4182)	Static*	STRING	D	Input parameter for sender address: STRING with a maximum length of 240 characters (see example call).
SFC_STATUS (Page 4183)	Static*	WORD	I, Q, M, D, L	Output/status parameter "SFC_STATUS": Error information of the called communication blocks.
*The values of the parameter are not modified at every call of the instruction "TM_MAIL". The values are in the static parameters of the instance and are only written once at the first call of the instruction.				

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

Optional parameters

The optional parameters CC, TEXT, and ATTACHMENT are only sent with the e-mail if the corresponding parameters contain a string of length > 0.

Note

Parameter ATTACHMENT

The specified attachment must be in the file format BIN. If text is to be transferred, you have the following options:

- In Windows, you link the file format BIN with a text editor.
- You use the Windows functions "Send to" or "Open with".

SMTP authentication

Authentication refers here to a procedure for ensuring an identity, for example, by a password query.

The "TM_MAIL" instruction supports the SMTP authentication procedure AUTH-LOGIN that is requested by most mail servers. For information about the authentication procedure of your mail server, please refer to your mail server manual or the website of your Internet service provider.

To use the AUTH-LOGIN authentication procedure, the "TM_MAIL" instruction requires the user name with which it can log onto the mail server. This user name corresponds to the user name with which you set up a mail account on your mail server. It is made available to the "TM_MAIL" instruction in the USERNAME parameter.

To log on, the "TM_MAIL" instruction also requires the associated password. This password corresponds to the password you specified when you set up your mail account. It is made available to the "TM_MAIL" instruction in the PASSWORD parameter.

The user name and the password are each transferred to the mail server unencrypted (BASE64-coded).

If no user name is specified in the DB, the AUTH-LOGIN authentication procedure is not used. The e-mail is then sent without authentication.

Example

You can find a detailed application example in the Siemens Industry Online Support under the following FAQ ID: 67262019 (https://support.industry.siemens.com/cs/ww/en/view/67262019).

Parameters TO_S, CC, and FROM (S7-1200)

Description

The TO_S, CC, and FROM parameters are strings with, for example, the following content:

- TO: <wenna@mydomain.com>, <ruby@mydomain.com>,
- CC: <admin@mydomain.com>, <judy@mydomain.com>,
- FROM: <admin@mydomain.com>

Note the following rules when entering the parameters:

- The "TO:", "CC:", and "FROM:" characters must be entered.
- A space and an opening pointed bracket "<" must be entered before each address.
- A closing pointed bracket ">" must be entered after each address.
- A comma must be entered after each address in TO and CC.
- Only one e-mail may be entered in FROM, and there must not be a comma at the end of this address

Because of runtime and memory space, the "TM_MAIL" instruction does not perform any syntax check of the parameters TO_S, CC and FROM.

STATUS and SFB_STATUS parameters (S7-1200)

Description

The return values of the "TM_MAIL" instruction can be classified as follows:

- W#16#0000: "TM_MAIL" was completed successfully
- W#16#7xxx: Status of "TM_MAIL"
- W#16#8xxx: An error was reported during the internal call of a communication block or from the mail server.

The following table shows the return values of "TM_MAIL" except for error codes of the internally called communication blocks.

Return value STATUS* (W#16#...):	Return value SFB_STATUS (W#16#...):	Explanation	Notes
0000	-	The processing of "TM_MAIL" was completed without error.	A without error completion of "TM_MAIL" does not mean that the sent e-mail will necessarily arrive (see below - note on point 1)
7001		"TM_MAIL" is active (BUSY = 1).	Initial call; job has started
7002	7002	"TM_MAIL" is active (BUSY = 1).	Intermediate call; job already active
8xxx	xxxx	The processing of "TM_MAIL" was completed with an error code of the internally called communication instructions.	For detailed information on the evaluation of the SFB_STATUS parameter, refer to the descriptions of the STATUS parameter of the communication instructions.
8010	xxxx	Error during connection establishment.	For further information on the evaluation of the SFB_STATUS parameter, refer to the descriptions of the STATUS parameter of the "TCON (Page 3538)" instruction.
8011	xxxx	Error sending the data.	For further information on the evaluation of SFB_STATUS, refer to the descriptions of the STATUS parameter of the "TSEND (Page 3553)" instruction.

Return value STATUS* (W#16#...):	Return value SFB_STATUS (W#16#...):	Explanation	Notes
8012	xxxx	Error receiving the data.	For further information on the evaluation of SFB_STATUS, refer to the descriptions of the STATUS parameter of the "TRCV (Page 3557)" instruction.
8013	xxxx	Error during connection establishment.	For further information on the evaluation of SFB_STATUS, refer to the descriptions of the STATUS parameter of the "TCON (Page 3538)" and "TDISCON (Page 3548)" instructions.
8014	-	Establishment of a connection is not possible.	You have possibly entered an incorrect mail server IP address (ADDR_MAIL_SERVER) or a time span that is too short (WATCH_DOG_TIME) to establish the connection. It is also possible that the CPU has no connection to the network or that the CPU configuration is incorrect.
8016	-	Error copying the attachment	-
82xx, 84xx, or 85xx	-	The error message originates from the mail server and the last three digits correspond to the error number of the SMTP protocol. The following columns list several error codes that can occur:	See point 2 in the note.
8450	-	Action not executed: Mailbox not available/ not reachable.	Try again later.
8451	-	Action aborted: Local processing error	Try again later.
8500	-	Syntax error: Error not recognized. This also includes the error when a command string is too long. This can also be occur when the e-mail server does not support the LOGIN authentication procedure.	Check the parameters of "TM_MAIL". Try to send an e-mail without authentication. To do this, replace the USERNAME parameter with an empty string.
8501	-	Syntax error: Parameter or argument incorrect	You have possibly entered an incorrect address in TO_S or CC.
8502	-	Command unknown or not implemented.	Check your entries, in particular the FROM parameter. This is possibly incomplete and you have forgotten "@" or ".".
8535	-	SMTP authentication incomplete.	You have possibly entered an incorrect user name or incorrect password.
8550	-	Mail server cannot be reached, you have no access rights.	You have possibly entered an incorrect user name or password, or the mail server does not support your LOGIN. Another cause of error could be an incorrect entry of the domain name after the "@" in TO_S or CC.
8552	-	Action aborted: Assigned memory size has been exceeded	Try again later.

Return value STATUS* (W#16#...):	Return value SFB_STATUS (W#16#...):	Explanation	Notes
8554	-	Transmission failed.	Try again later.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".			

Note

Status error

1. An incorrect entry of the recipient addresses does not generate a status error of the "TM_MAIL" instruction. In this case, there is no guarantee that the e-mail will be sent to other recipients, even if these were entered correctly.
2. You can find more detailed information on the SMTP error code and other error codes in SMTP protocol on the Internet or in the error documentation of the mail server. You can also view the most recent message from the mail server in your instance DB in the BUFFER1 parameter. If you look under "Data", you will find the data most recently sent by the "TM_MAIL" instruction.

See also

Switching display formats in the program status (Page 7698)

4.2 Instructions

4.2.1 Asynchronous instructions

4.2.1.1 Difference between synchronous and asynchronous instructions

Asynchronous instructions

In program processing, a differentiation is made between synchronous and asynchronous instructions.

The "synchronous" and "asynchronous" properties relate to the temporal relationship between the call and execution of the instruction.

- The following applies to synchronous instructions: When the call of a synchronous instruction is ended, the execution is also ended.
- This is different in the case of asynchronous instructions: The execution of an asynchronous instruction can extend over multiple calls. The CPU processes asynchronous instructions in parallel with the cyclic user program. Asynchronous instructions occupy resources in the CPU while they are being processed.

Asynchronous instructions are usually instructions for transferring data (data records for modules, communication data, diagnostics data).

Additional information about asynchronous instructions can be found here:SIMATIC S7-1500 / ET 200MP Manual Collection (https://support.industry.siemens.com/cs/ww/en/view/86140384) In the system manual under Basics of program processing > Asynchronous instructions

Identification of the job

To execute an instruction over multiple calls, the CPU must be able to uniquely assign a subsequent call to a running job of the instruction. To assign a call to a job, the CPU uses one of the following two mechanisms, depending on the type of the instruction:

- The instance of the instruction (with type "SFB" system function block)
- The input parameter of the instruction identifying the job

If you use asynchronous instructions to trigger a process interrupt, output control commands to DP slaves, start a data transfer, or abort a non-configured connection and then call the same instruction again before the current job is completed, the reaction of the instruction will depend on whether the second call involves the same job.

Passing parameters to program blocks that are to be executed asynchronously

Code blocks (FBs/FCs) and data blocks (DBs) can be created with different access types ("standard" and "optimized"). In code blocks, you can call any instructions. Some instructions (e.g. "WRIT_DBL" and "READ_DBL") are executed asynchronously. These blocks cannot be supplied with tags from TEMP because the data must not be changed during execution.

Ensure that you do not use these instructions in programs in which code blocks of different access types are called reciprocally. This could cause the following to occur:

- A structure from a standard data block is directly or indirectly passed to an optimized code block, which forwards this structure directly or indirectly to one of the blocks mentioned above.
- The reverse scenario, whereby a structure from an optimized code block is directly or indirectly passed to a standard data block, which forwards this structure directly or indirectly to one of the program blocks mentioned above.

Otherwise, a hidden copy of the passed data is created in TEMP, which leads to the asynchronous program blocks returning a negative acknowledgment.

Parameter REQ

The input parameter REQ (request) is used solely to start the job:

- Trigger the job by setting the input parameter REQ to "1" (case 1).
- If a particular job has been started and not yet completed and you call the instruction again to perform the same job (for example, in a cyclic interrupt OB), then REQ is not evaluated by the instruction (case 2).
- If the job is completed, but the input parameter REQ is still set to "1", the job is immediately started again.

Note

REQ bit = "1"

Please note that, when the REQ bit is set, the asynchronous instruction will be started again after the previous call is completed. This may not be desired. To make the project clearer and easier to maintain, it is therefore advisable to reset the REQ bit to "0" as early as possible.

Parameter RET_VAL and BUSY

The output parameters RET_VAL and BUSY indicate the status of the job.

Pay attention to the note in section: Evaluating errors with output parameter RET_VAL (Page 162)

- In case 1 (first call with REQ = 1), code W#16#7001 will be entered in RET_VAL and BUSY will be set to "1" if system resources are available and the supply of the input parameters is correct.
  If the required system resources are currently being used or the input parameters contain an error, the corresponding error code is entered in RET_VAL and BUSY receives the value "0".
- In case 2 (intermediate call), the code W#16#7002 is entered in RET_VAL (this corresponds to the message that the call is currently still being processed) and BUSY is set to "1".
- The following applies to the last call for a job:
  - For instructions "DPNRM_DG (Page 5287)", "X_GET (Page 6799)" and "I_GET (Page 6784)" in case of fault-free data transmission, the number of supplied data will be entered in RET_VAL as a positive number in bytes. BUSY has the value "0" in this case.
    If there is an error, then the error information will be entered in RET_VAL and you should not evaluate BUSY in this case.
  - With "RD_REC (Page 5256), the size of the data record in bytes is entered in RET_VAL or the value 0 if no error has occurred during data transmission. BUSY receives the value "0" in this case. If there is an error, the error code will be entered in RET_VAL and BUSY will receive the value "0".
  - For all other instructions, "0" will be entered in RET_VAL if the job was executed without errors and BUSY will receive the value "0". If there is an error, the error code will be entered in RET_VAL and BUSY will receive the value "0".

Note

If the first and last call coincide, the reaction for RET_VAL and BUSY is the same as for the last call.

Resources occupied by the job

Every job occupies CPU resources. The instruction "RDREC", for example, occupies memory space for the destination area. These resources are assigned exclusively to this job and are occupied as long as the instruction is active, i.e. as long as BUSY has the value "1".

You cannot change occupied resources or use them for other purposes. For example, you cannot change either the length or the content of the destination area for the instruction "RDREC".

Overview

The following table provides you with an overview of the relationships explained above. In particular, it shows the possible values of the output parameters if the execution of the job is not completed after an instruction call has been completed.

Note

Following every call, you must evaluate the relevant output parameters in your program.

Relationship between call, REQ, RET_VAL and BUSY during a "running" job.

Number of the call	Type of call	REQ	RET_VAL	BUSY	DONE	ERROR
1	First call	1	W#16#7001	1	0	0
			Error code	0	0	1
2 to (n - 1)	Intermediate call	irrelevant	W#16#7002	1	0	0
n	Last call	irrelevant	W#16#0000 (exceptions: "RD_REC (Page 5256)" if the destination area is larger than the data record transferred; "DPNRM_DG (Page 5287)", "X_GET (Page 6799)" and "I_GET (Page 6784)"), if no errors have occurred.	0	1	0
			Error code if errors occurred	0	0	1

4.2.2 Basic instructions

4.2.2.1 LAD

Bit logic operations

---| |---: Normally open contact

Description

The activation of the normally open contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally open contact closes and the signal state at the output is set to the signl state of the input.

When the operand has signal state "0", the normally open contact is not activated and the signal state at the output of the instruction is reset to "0".

Two or more normally open contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally open contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

Create a global data block with the following content for this purpose:

Name of the block: SLI_gDB_NOContact	
Name	Data type
start1	BOOL
start2	BOOL
start3	BOOL
startOut	BOOL

Write the following program code:

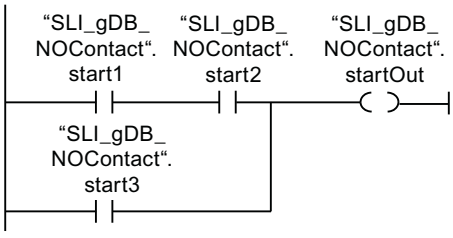

The "SLI_gDB_NOContact".startOut operand is set when one of the following conditions is fulfilled:

- The "SLI_gDB_NOContact".start1 and "SLI_gDB_NOContact".start2 operands have the signal state "1".
- The "SLI_gDB_NOContact".start3 operand has the signal state "1".

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

---| / |---: Normally closed contact

Description

The activation of the normally closed contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally closed contact opens and the signal state at the output of the instruction is reset to "0".

When the operand has signal state "0", the normally closed contact is not enabled and the signal state of the input is transferred to the output.

Two or more normally closed contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally closed contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

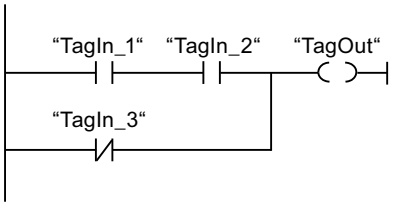

The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

--|NOT|--: Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). If the signal state is "1" at the input of the instruction, the output of the instruction has signal state "0". If the signal state is "0" at the input of the instruction, the output has the signal state "1".

Example

The following example shows how the instruction works:

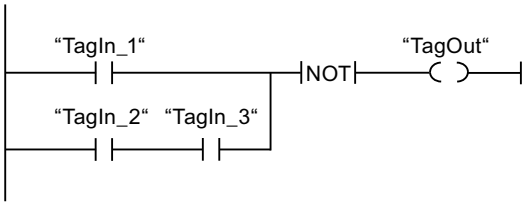

Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The signal state of the operands "TagIn_2" and "TagIn_3" is "1".

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

---( )---: Assignment

Description

You can use the "Assignment" instruction to set the bit of a specified operand. If the result of logic operation (RLO) at the input of the coil has signal state "1", the specified operand is set to signal state "1". If the signal state is "0" at the input of the coil, the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output.

Parameters

The following table shows the parameters of the "Assignment" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:

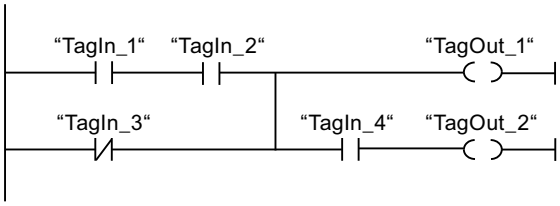

The "TagOut_1" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

The "TagOut_2" operand is set when one of the following conditions is fulfilled:

- Operands "TagIn_1", "TagIn_2", and "TagIn_4" have signal state "1".
- The signal state of the "TagIn_3" operand is "0" and the signal state of the "TagIn_4" operand is "1".

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

---(R)---: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is reset to "0". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

Parameters

The following table shows the parameters of the "Reset output" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L, T, C	Operand that is reset when RLO = "1".

Example

The following example shows how the instruction works:

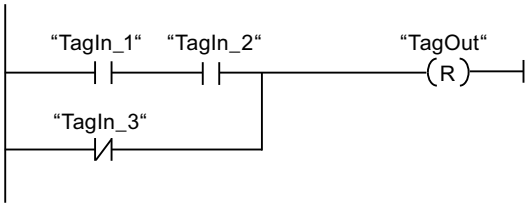

Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

---(S)---: Set output

Description

You can use the "Set output" instruction to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is set to "1". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

Parameters

The following table shows the parameters of the "Set output" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:

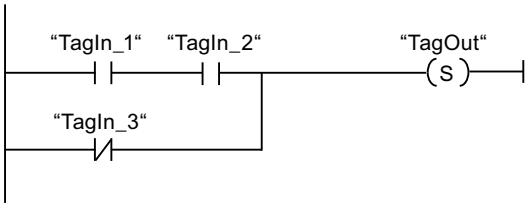

The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

SR: Set/reset flip-flop

Description

Use the instruction "Set/reset flip-flop" to set or reset the bit of the specified operand, depending on the signal state of the inputs S and R1. If the signal state at input S is "1" and the input R1 is "0", the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Parameters

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameters	Declaration	Data type	Memory area	Description
S	Input	BOOL	I, Q, M, D, L	Enable setting
R1	Input	BOOL	I, Q, M, D, L, T, C	Enable resetting
<Operand>	InOut	BOOL	I, Q, M, D, L	Operand that is set or reset.
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:

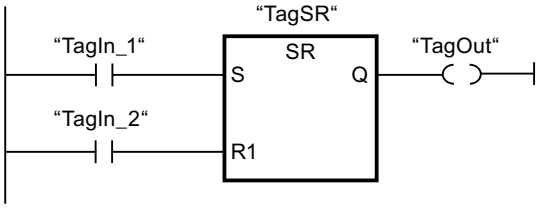

The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 231)

Master control relay (Page 191)

---(MCR<): Open MCR ranges (Page 4339)

Basic information on LAD (Page 7121)

RS: Reset/set flip-flop

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Parameters

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameters	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, D, L	Enable resetting
S1	Input	BOOL	I, Q, M, D, L, T, C	Enable setting
<Operand>	InOut	BOOL	I, Q, M, D, L	Operand that is reset or set.
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:

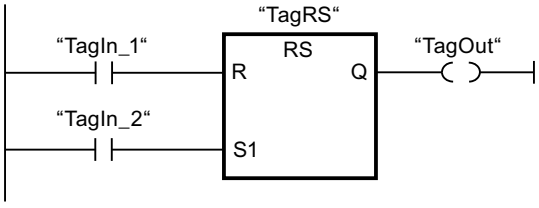

The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 231)

Master control relay (Page 191)

---(MCR<): Open MCR ranges (Page 4339)

Basic information on LAD (Page 7121)

--|P|--: Scan operand for positive signal edge

Description

You can use the "Scan operand for positive signal edge" instruction to determine if there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

The following figure shows the change in the signal state in case of a negative and a positive signal edge:

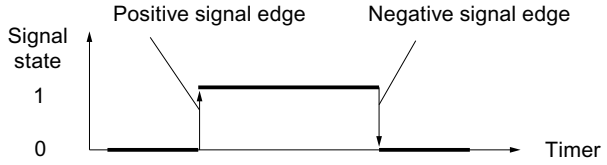

The positive signal edge is queried each time the instruction executes. When a positive signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:

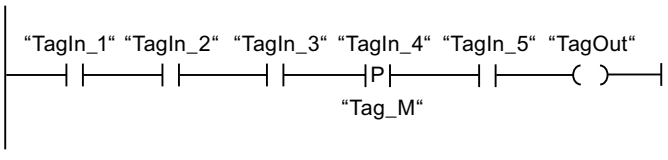

Operand "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a rising edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

--|N|--: Scan operand for negative signal edge

Description

You can use the "Scan operand for negative signal edge" instruction to determine if there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan that is saved in an edge memory bit <Operand2>. If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

The following figure shows the change in the signal state in case of a negative and a positive signal edge:

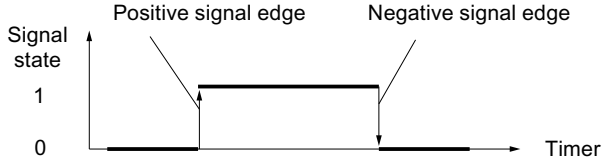

The negative signal edge is queried each time the instruction executes. When a negative signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:

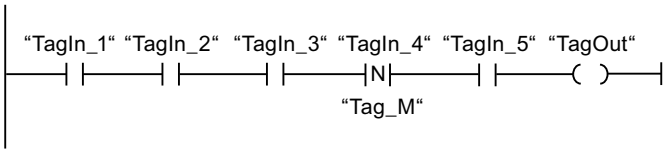

Operand "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a negative signal edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

P_TRIG: Scan RLO for positive signal edge

Description

Use the "Scan RLO for positive signal edge" instruction to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

The positive signal edge is queried each time the instruction executes. As soon as a positive single edge is detected, the output Q of the instruction returns the signal state "1" for the length of a program cycle. In all other cases, the output returns the signal state "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the instruction "Scan RLO for positive signal edge":

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

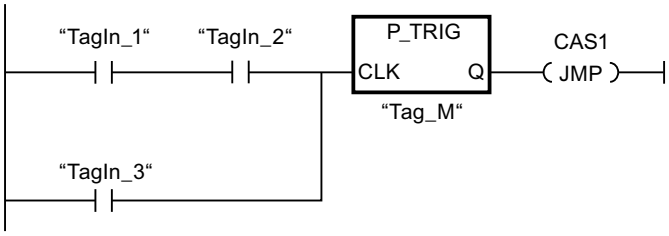

The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

N_TRIG: Scan RLO for negative signal edge

Description

Use the "Scan RLO for negative signal edge" instruction to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative signal edge.

The negative signal edge is queried each time the instruction executes. As soon as a negative single edge is detected, the output Q of the instruction returns the signal state "1" for the length of a program cycle. In all other cases, the signal state at the output of the instruction is "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

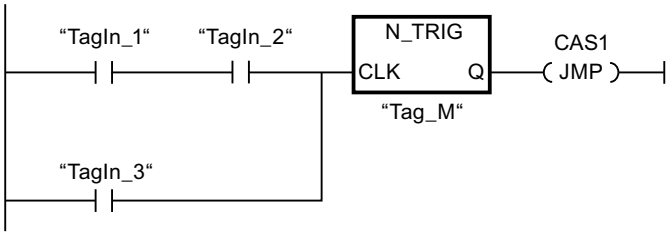

The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

Timer operations

IEC Timers

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the output Q for a programmed duration. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The execution of the instruction requires a preceding logic operation. It can be placed within or at the end of the network. The programmed time PT begins when the instruction starts. Output Q is set for the duration PT, regardless of the subsequent course of the input signal. Even if a new positive signal edge is detected, the signal state at the output Q is not affected as long as the PT time duration is running.

You can scan the current time value at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. When the duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TP (for example, "TP_DB")
- Declaration as a local tag of the type TP in the "Static" section of a block (for example, #MyTP_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the

instruction "Generate pulse" are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the "Generate pulse" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P, or constant	Duration of the pulse The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current time value

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:

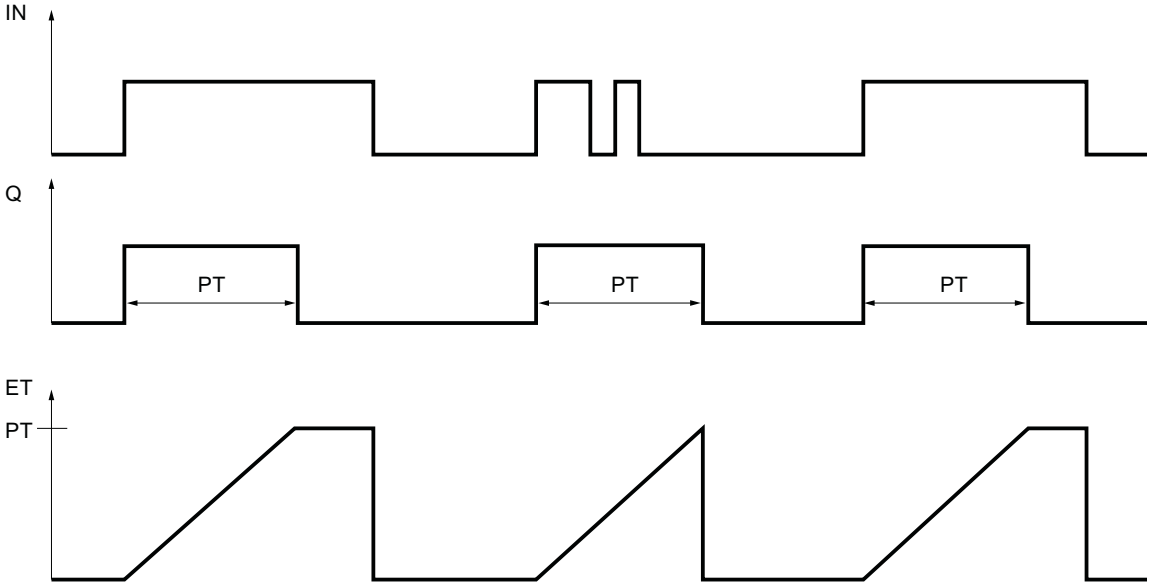

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the programmed duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The execution of the instruction requires a preceding logic operation. It can be placed within or at the end of the network. The programmed time PT begins when the instruction starts. When the duration PT expires, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can scan the current time value at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TON (for example, "TON_DB")
- Declaration as a local tag of the type TON in the "Static" section of a block (for example, #MyTON_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P, or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current time value

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:

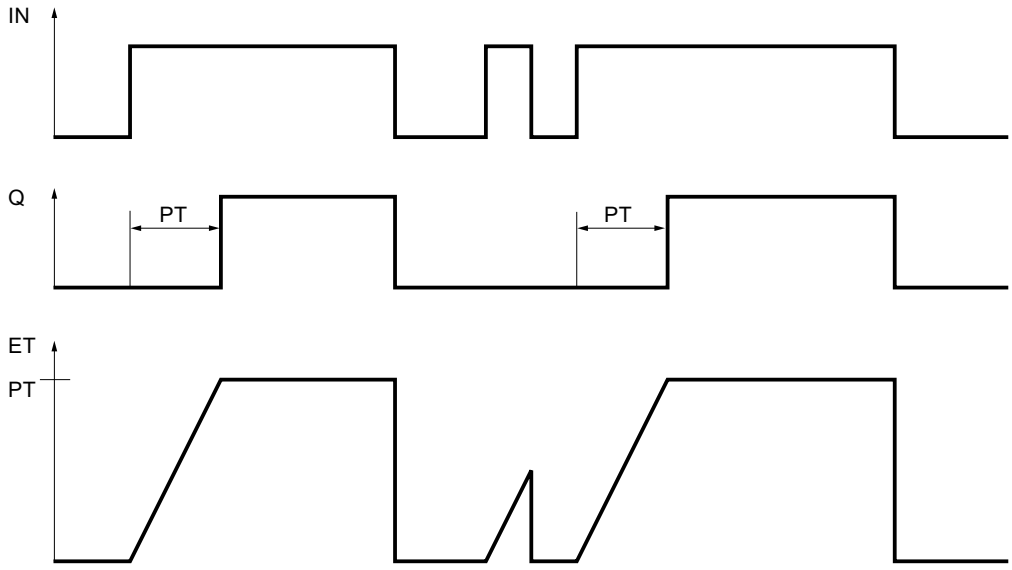

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed duration PT. The Q output is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The execution of the instruction requires a preceding logic operation. It can be placed within or at the end of the network. When the signal state at input IN changes back to "0", the programmed time PT starts. Output Q remains set as long as the duration PT is running. When duration PT expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q continues to be "1".

You can scan the current time value at the ET output. The current timer value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET output remains set to the current value until the IN input changes back to "1". If input IN switches to "1" before the duration PT has expired, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TOF (for example, "TOF_DB")
- Declaration as a local tag of the type TOF in the "Static" section of a block (for example, #MyTOF_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the "Generate off-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P, or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is reset when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current time value

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate off-delay" instruction:

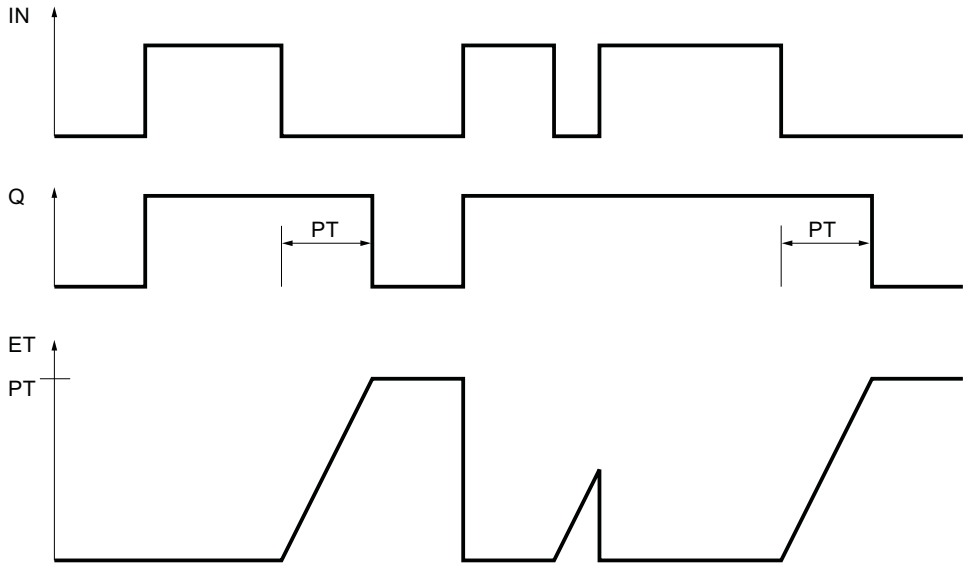

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

SIMATIC Timers

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as soon as the signal state at input S is "1". If the signal state at input S changes to "0" before the programmed duration expires, the timer is stopped. In this case, the signal state at output Q is "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign pulse timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction "Assign pulse timer parameters and start":

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction "Assign pulse timer parameters and start":

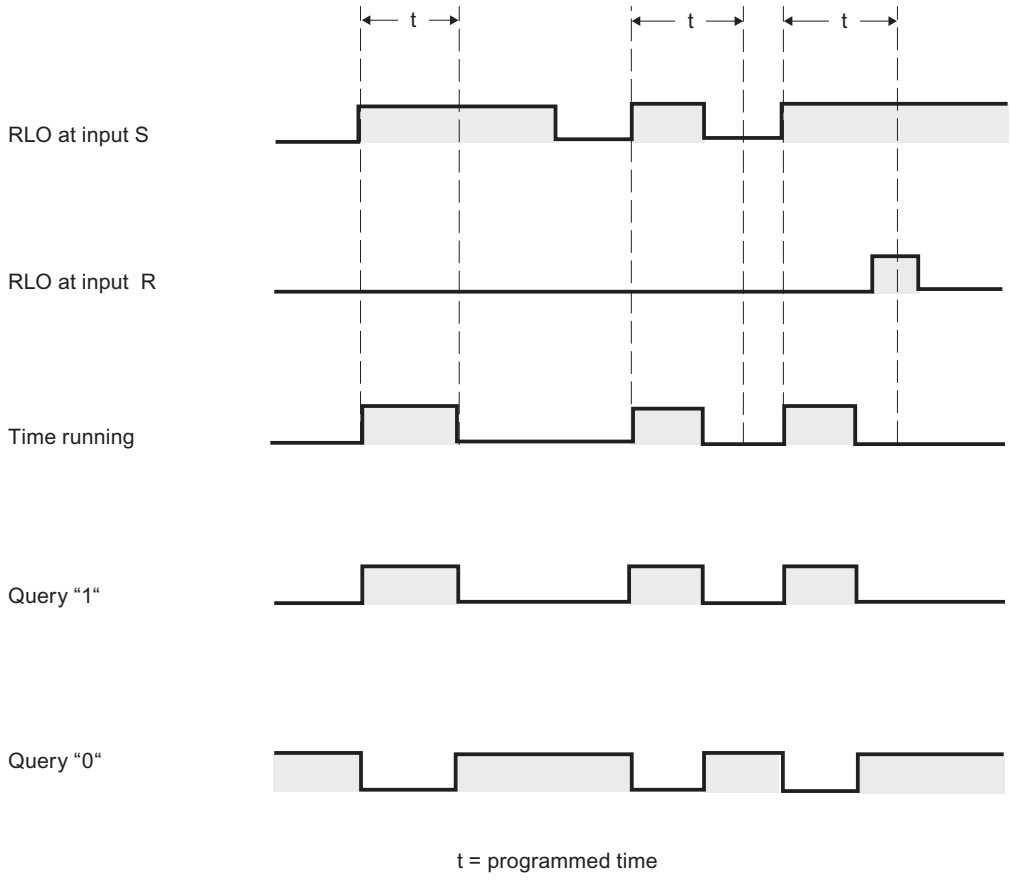

Example

The following example shows how the instruction works:

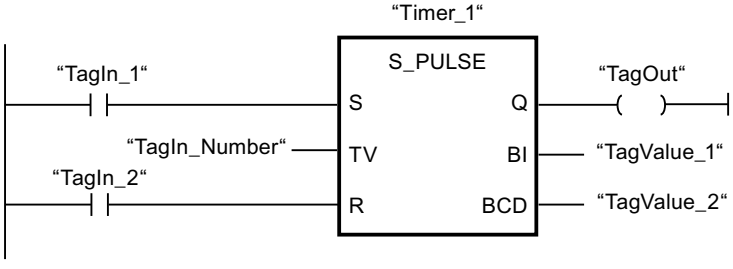

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" as long as the operand "TagIn_1" has the signal state "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer "Timer_1" is stopped. The operand "TagOut" is reset to "0" in this case.

The operand "TagOut" has the signal state "1" as long as the timer is running and the operand "TagIn_1" has the signal state "1". When the timer expires or is reset, the operand "TagOut" is reset to "0".

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on LAD (Page 7121)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". As long as the timer is running, the output Q has the signal state "1". When the timer expires, the output Q is reset to "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input TV.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign extended pulse timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction "Assign extended pulse timer parameters and start":

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:

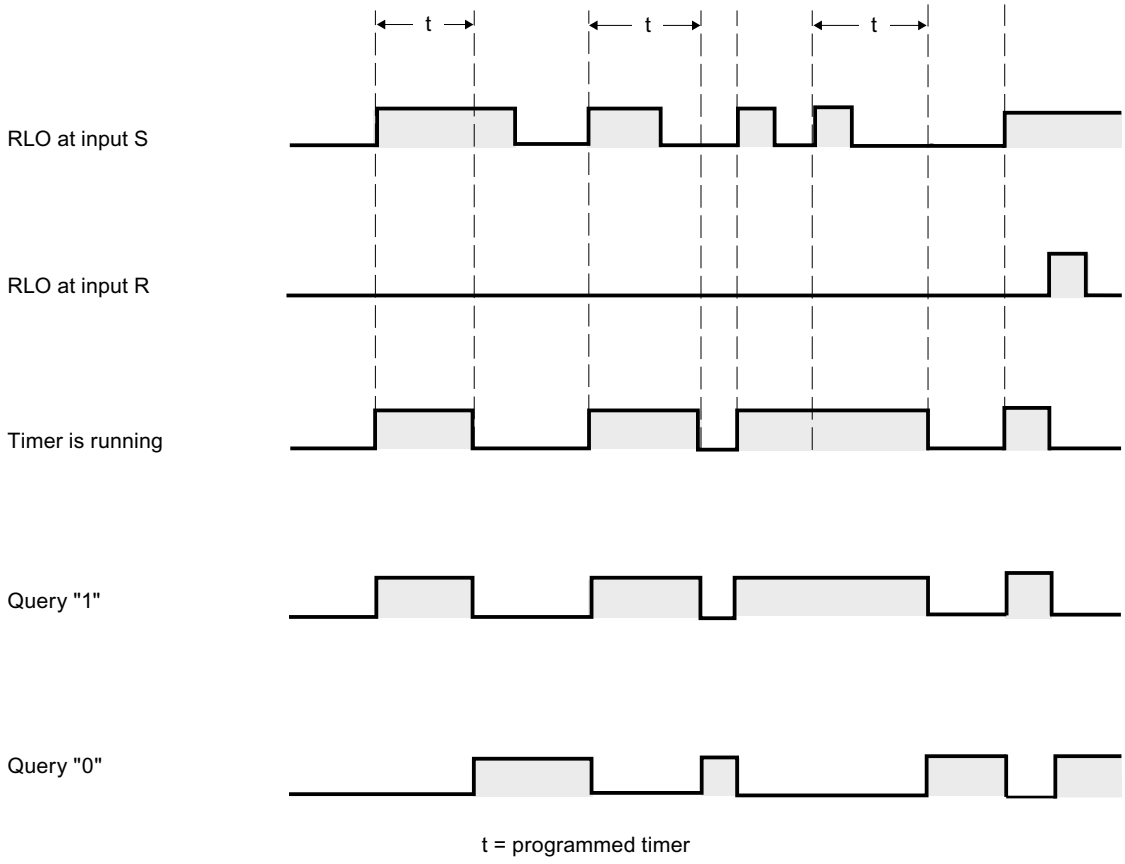

Example

The following example shows how the instruction works:

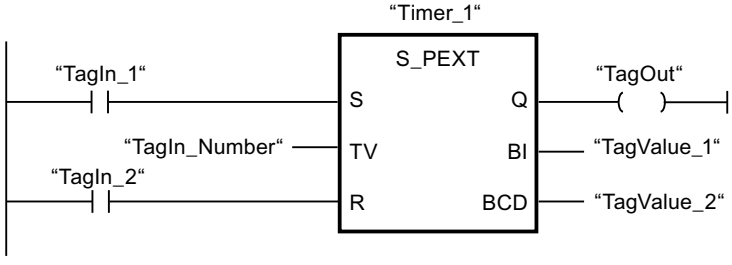

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative edge at input S. If the signal state of the operand "TagIn_1" changes from "0" to "1" before the timer expires, the timer is restarted.

The operand "TagOut" has the signal state "1" as long as the timer is running. When the timer expires or is reset, the operand "TagOut" is reset to "0".

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on LAD (Page 7121)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as soon as the signal state at input S is "1". If the timer expires correctly and input S still has signal state "1" then output Q returns signal state "1". If the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the time is running and the signal state at input R changes from "0" to "1" then the current time value and the time base are also set to zero. In this case, the signal state at output Q is "0". The timer is reset if the signal state is "1" at the R input even if the timer is not running and the RLO at input S is "1".

Specify the timer of the instruction in the placeholder above the box. The timer must be declared with the data type TIMER.

The "Assign on-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign on-delay timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:

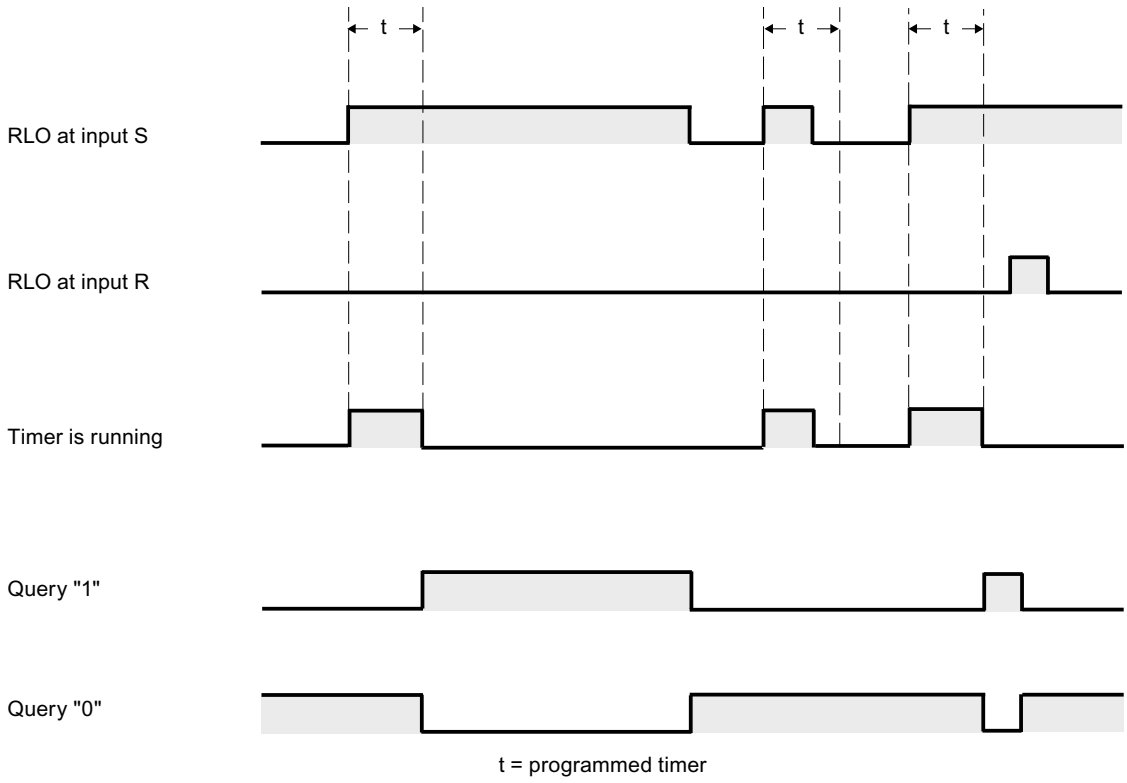

Example

The following example shows how the instruction works:

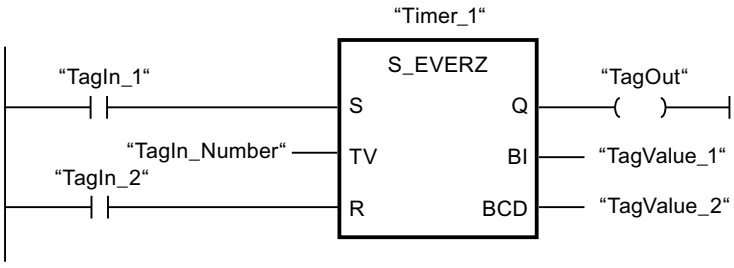

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". If the timer expires and the operand has the signal state "1", the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. The operand "TagOut" has the signal state "0" in this case.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on LAD (Page 7121)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". If the timer expires, the "Q" output returns signal state "1" regardless of the signal state at input "S". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input (TV).

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0" regardless of the signal state at start input S. In this case, the signal state at output Q is "0".

The "Assign retentive on-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign retentive on-delay timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:

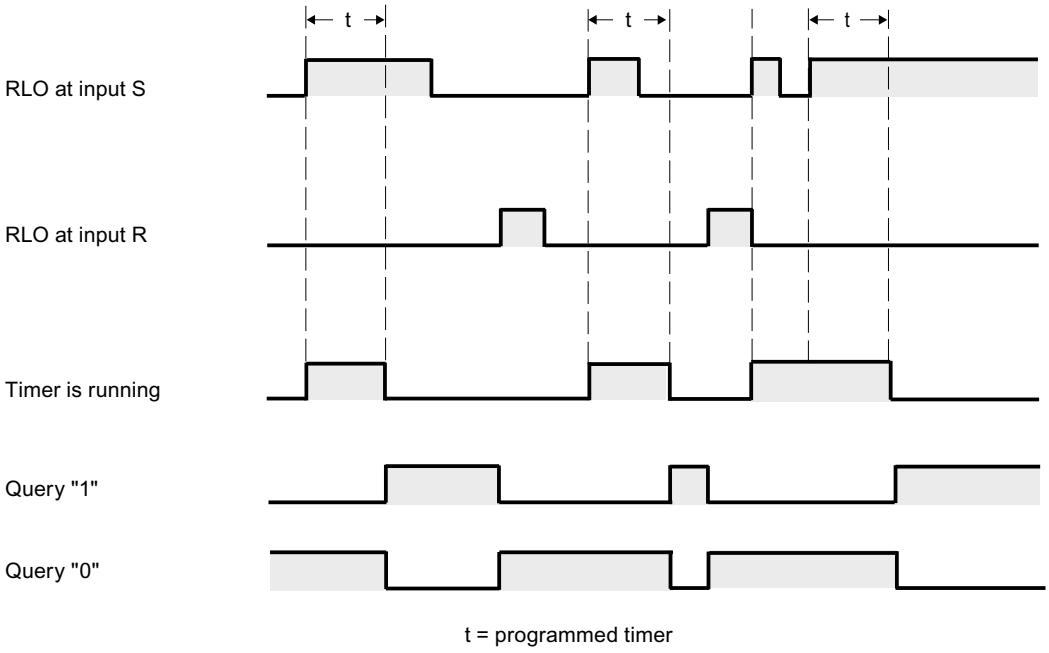

Example

The following example shows how the instruction works:

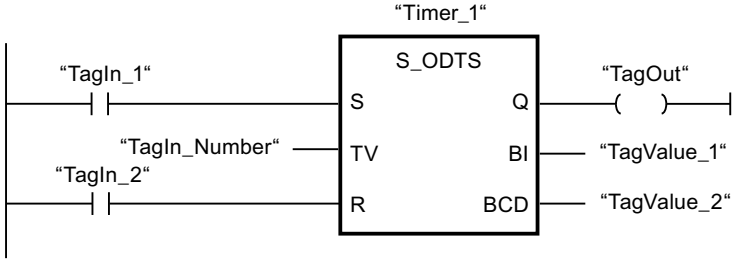

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number", even if the signal state of the operand "TagIn_1" changes to "0". When the timer expires, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is restarted.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on LAD (Page 7121)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV). As long as the timer is running or input S returns signal state "1", then output Q has signal state "1". When the timer expires and the signal state at input S is "0", output Q is reset to the signal state "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at input S.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0". In this case, the signal state at output Q is "0".

The "Assign off-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign off-delay timer parameters and start" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)

Parameters	Declaration	Data type	Memory area	Description
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:

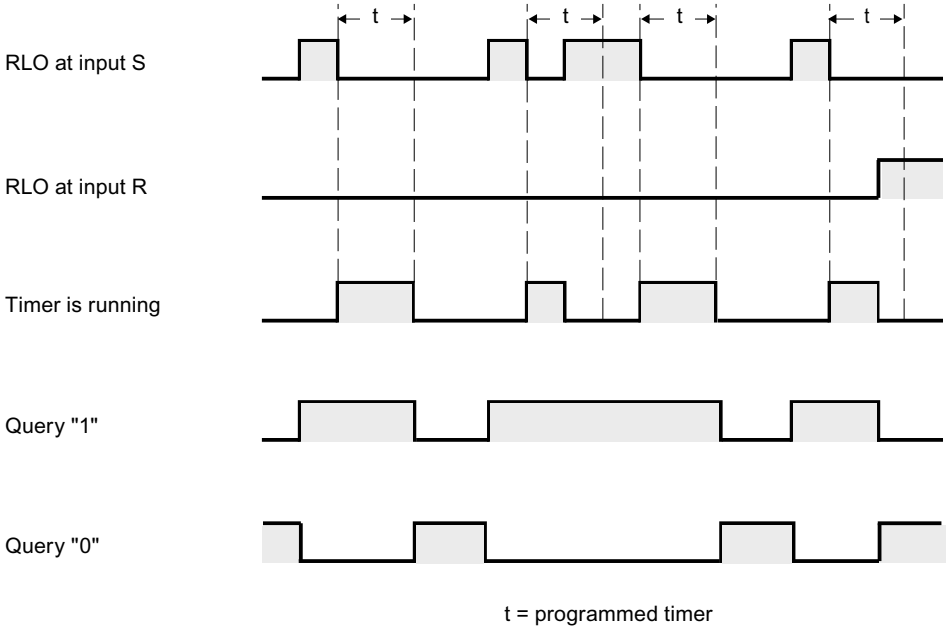

Example

The following example shows how the instruction works:

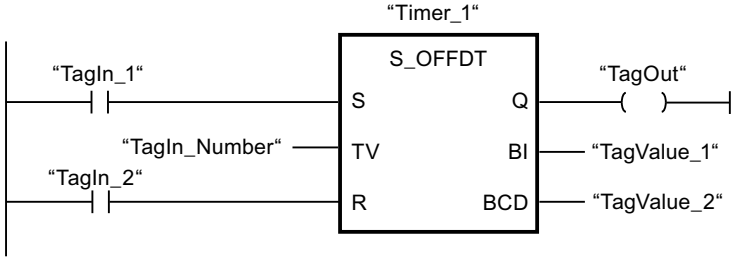

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "1" to "0". The timer expires with the time value of the operand "TagIn_Number". The operand "TagOut" is set to "1" when the timer is running and when the operand "TagIn_1" has the signal

state "0". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is reset.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on LAD (Page 7121)

---(SP): Start pulse timer

Description

The "Start pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer runs with the specified duration as long as the RLO has the signal state "1". As long as the timer is running, the query of timer status "1" returns the signal state "1". If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer stops. In this case, the querying of the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down to zero. The time base indicates the time period with which the time value is changed.

The "Start pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction "Start pulse timer":

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

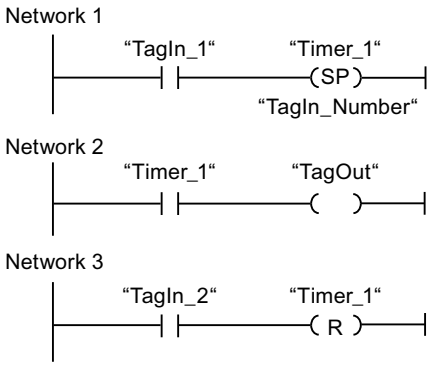

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand as long as the signal state of the "TagIn_1" operand is "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. As long as the timer is running, the "TagOut" operand returns signal state "1". A signal state change of the operand "TagIn_2" from "0" to "1" resets the timer, which stops the timer and sets the current time value to "0".

The figure below shows the timer diagram of the example:

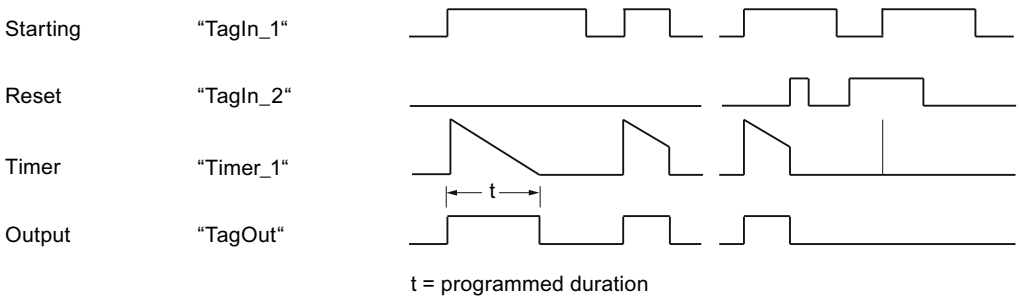

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on LAD (Page 7121)

---(SE): Start extended pulse timer

Description

The "Start extended pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer runs for the specified time period even when the RLO changes to signal state "0". As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is restarted with the programmed time period. The querying of the timer status for "1" returns the signal state "0" when the timer expires.

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down to zero. The time base indicates the time period with which the time value is changed.

The "Start extended pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameters

The following table shows the parameters of the "Start extended pulse timer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

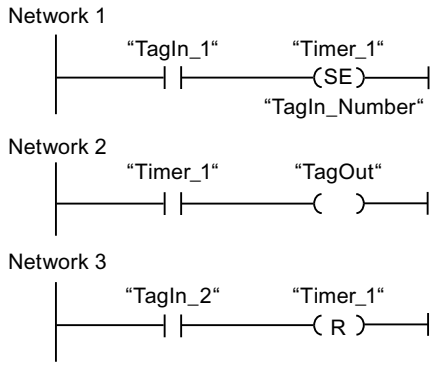

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative signal edge in the RLO. As long as the timer is running, the "TagOut" operand returns signal state "1". If the signal state of the operand "TagIn_1" changes again from "0" to "1" before the timer expires, the timer is restarted.

The figure below shows the timer diagram of the example:

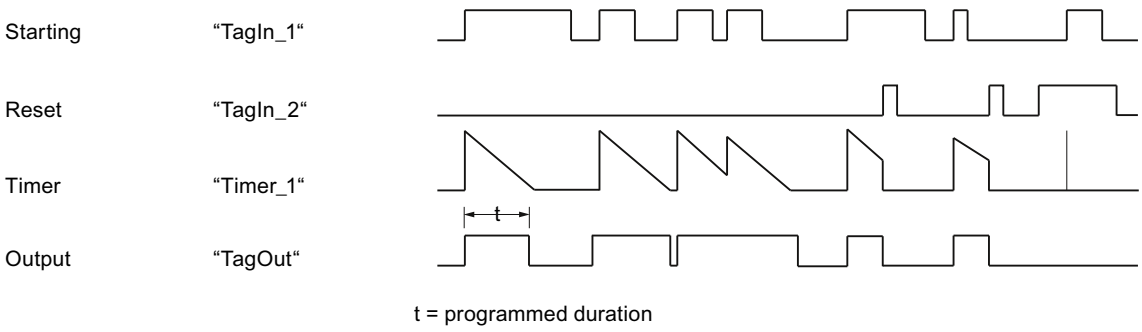

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(SD): Start on-delay timer

Description

The "Start on-delay timer" instruction starts a programmed timer when the signal state "1" is detected at the start input. As long as this signal state remains "1", the timer expires with the specified duration. When the timer expires and the signal state at the start input is still "1", the query for timer status also returns "1". If the signal at the start input is "0", the timer is reset. In this case, querying the timer status returns the signal state "0". As soon as the signal at the start input changes to "1" again, the timer starts running again.

The signal state of the timer output is identical to the signal state of the start input. The start input is directly interconnected to the output and is not linked to the timer.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameters

The following table shows the parameters of the "Start on-delay timer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

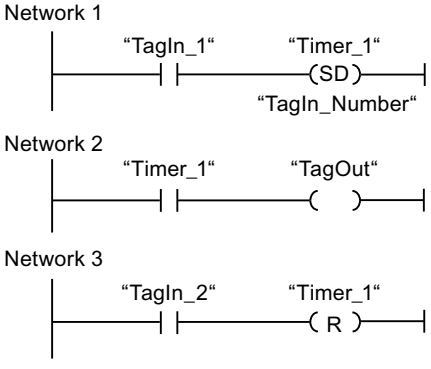

Network 1:

"#Timer_1" starts when the signal state of the "#TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "#TagIn_Number". If the signal state of the "#TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is reset.

Network 2:

When the timer has expired, the #TagIn_1 operand at the start input has signal state "1" and the timer was not reset, the #TagOut operand is "1".

Network 3:

When the signal state of the #TagIn_2 operand has signal state "1", the timer #Timer_1 and the output #TagOut are reset.

If you want to restart #Timer_1, the signal state at the #TagIn_2 operand must be "0" and the signal state at the start input #TagIn_1 must change from "0" to "1".

The figure below shows the timer diagram of the example:

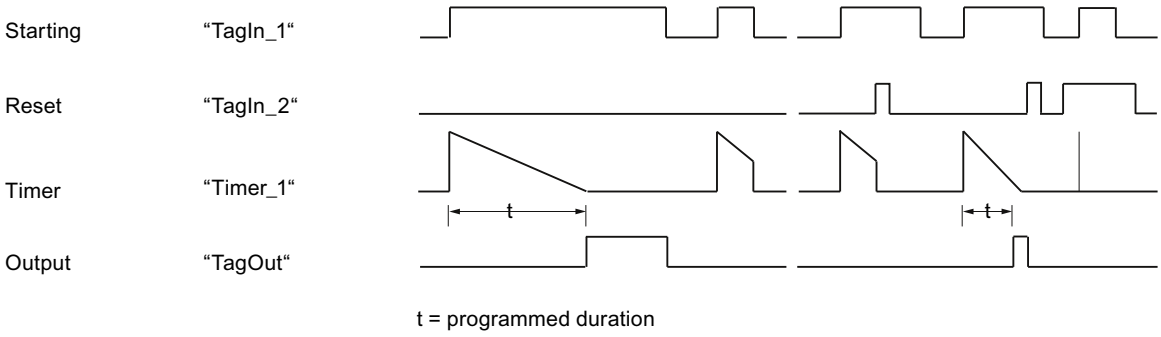

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(SS): Start retentive on-delay timer

Description

The "Start retentive on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration, even if the RLO changes to the signal state "0". When the timer expires, the querying of the timer status for "1" returns the signal state "1". After expiry of the timer, the timer can only be restarted if it is explicitly reset.

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start retentive on-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameter

The following table shows the parameters of the "Start retentive on-delay timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

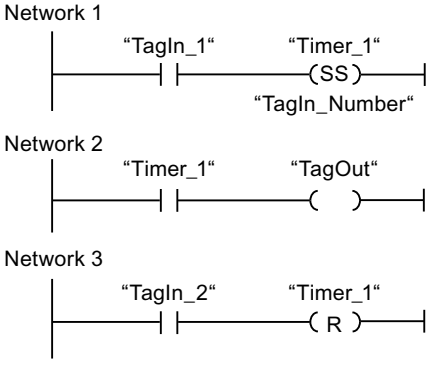

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". When the timer expires, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is restarted. If the signal state of the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

The figure below shows the timer diagram of the example:

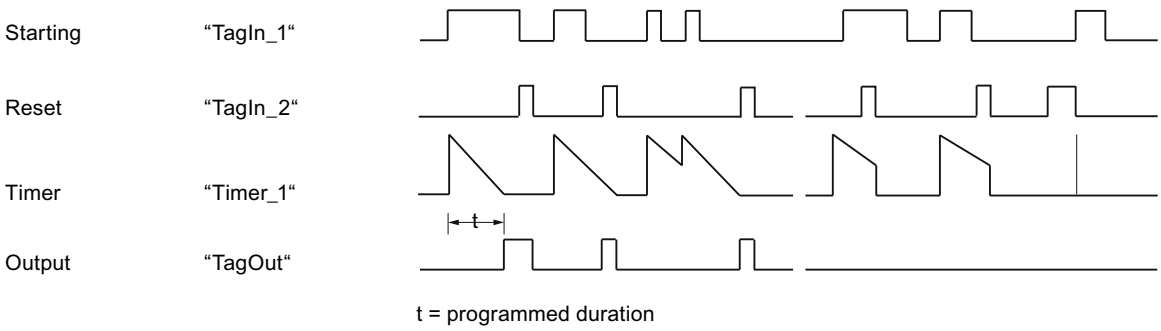

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(SF): Start off-delay timer

Description

The "Start off-delay timer" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration. As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is reset. The timer is always restarted when the RLO changes from "1" to "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed timer value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The queries of the timer status for "1" return the query result "1" if the signal state of the result of logic operation is "1" at the execution of the instruction. If the RLO is "0", queries of the timer status for "1" return the query result "0".

The "Start off-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameters

The following table shows the parameters of the instruction "Start off-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

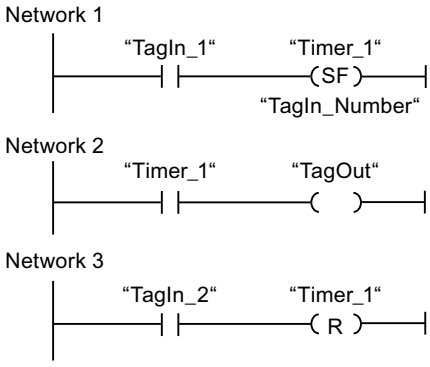

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "1" to "0". The timer expires with the value of operand "TagIn_Number". As long as the timer is running, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" while the timer is running, the timer is restarted. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

The figure below shows the timer diagram of the example:

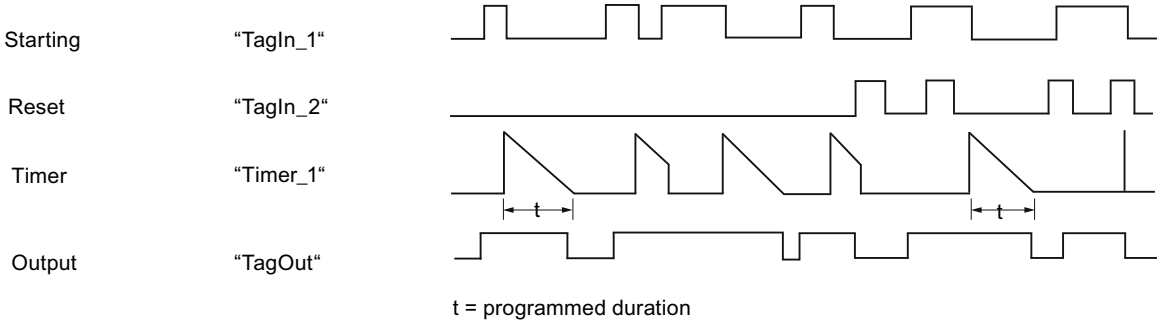

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

Counter operations

IEC Counters

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type INT specified at the CV output. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can query the counter status in the Q output. The signal state at the Q output is determined by the PV parameter. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is reset to zero when the signal state at input R changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")
- Declaration as a local tag of the type CTU in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the R parameter of the instruction in a startup OB. If

instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

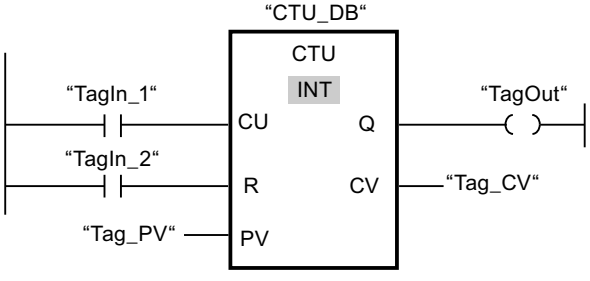

When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the operand "Tag_CV" is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on LAD (Page 7121)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type (INT). When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can query the counter status in the Q output. If the current counter value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the parameter PV.

The value at the CV output is set to the value of the PV parameter when the signal state at the LD input changes to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTD (for example, "CTD_DB")
- Declaration as a local tag of the type CTD in the "Static" section of a block (for example, #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case, the desired initial value for the CV parameter is specified in the PV parameter.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

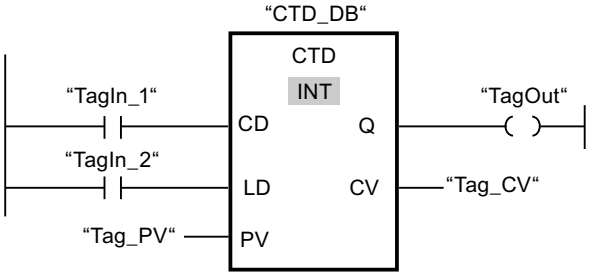

When the signal state of the "TagIn_1" operand changes from "0" to "1", the instruction is executed and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the counter value is decremented until it reaches the low limit of the data type (INT = -32768).

The "TagOut" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on LAD (Page 7121)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type INT specified at the CV output. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of the specified data type (INT) is reached, the counter value is not decremented any further.

When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter. As long as the LD input has signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0".

You can query the status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")
- Declaration as a local tag of the type CTUD in the "Static" section of a block (for example, #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a startup OB:

- When used as up counter, the value at the R parameter must be set to "1".
- When used as down counter, the value at the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P, or constant	Value at which the QU output is set / value to which the CV output is set when LD = 1.
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	INT	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

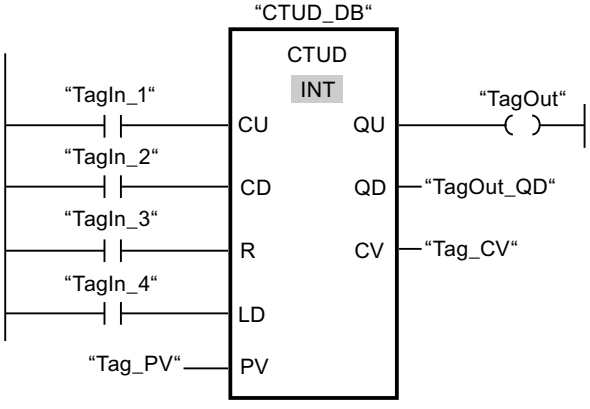

If the signal state at the "TagIn_1" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input, the current counter value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_2" input, the counter value

is decremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the CU input, the counter value is incremented until it reaches the high limit of 32767. If input CD has a positive signal edge, the counter value is decremented until it reaches the low limit of INT = -32768.

The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut" output has signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut_QD" output has signal state "0".

See also

Overview of the valid data types (Page 231)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

Instances (Page 62)

SIMATIC Counters

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter . If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CU is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

The "Assign parameters and count up" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

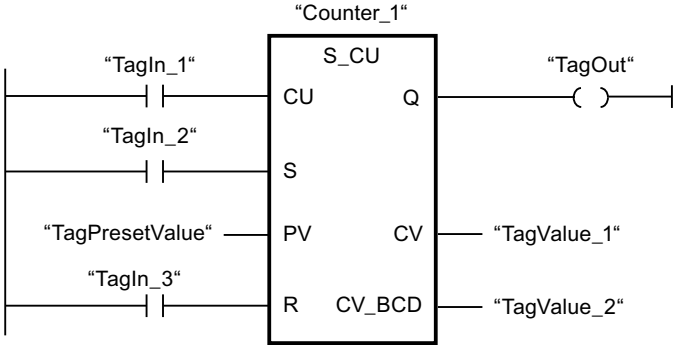

When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. When the signal state at input "TagIn_2" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on LAD (Page 7121)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter . If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be decremented until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CD is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

The "Assign parameters and count down" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input

Parameters	Declaration	Data type	Memory area	Description
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexa-decimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

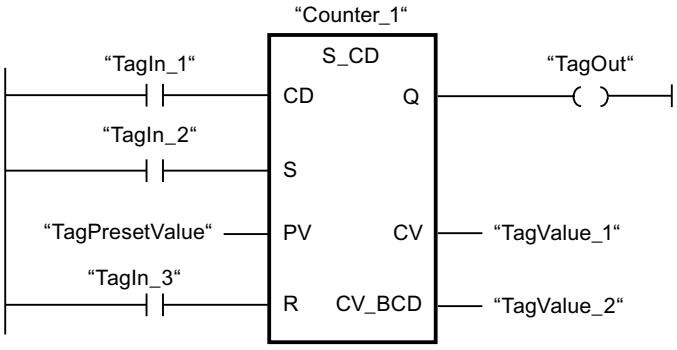

When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decremented by one. When the signal state at input "TagIn_2" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on LAD (Page 7121)

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment or decrement the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. If there is a positive signal edge at the CU and CD inputs in one program cycle, the counter value remains unchanged.

The counter value can be incremented until the high limit of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit "0" is reached, the counter value is not decremented any further.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO is "1" at the inputs CU and CD, the counter counts accordingly in the next scan cycle, even if no change in the signal edge was detected.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU, CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

The "Assign parameters and count up / down" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count up / down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L, T, C	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexadecimal)

Parameters	Declaration	Data type	Memory area	Description
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

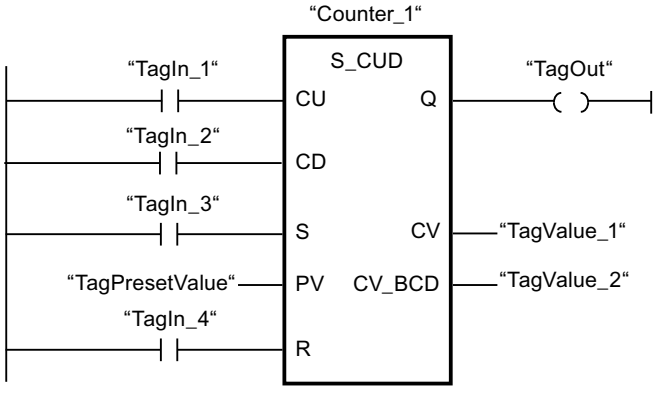

If the signal state at the input "TagIn_1" or at the input "TagIn_2" changes from "0" to "1" (positive edge), the instruction "Assign parameters and count up / down is executed. When there is a positive signal edge at the "TagIn_1" input and the current counter value is less than "999", the counter value is incremented by one. When there is a positive signal edge at the "TagIn_2" input and the current counter value is greater than "0", the counter value is decremented by one.

When the signal state at input "TagIn_3" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_4" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on LAD (Page 7121)

---(SC): Set counter value

Description

You can use the "Set counter value" instruction to set the value of a counter. The instruction is executed when the result of logic operation (RLO) at the input changes from "0" to "1". When the instruction is executed, the counter is set to the specified counter value.

The "Set counter value" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameters

The following table shows the parameters of the "Set counter value" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Count value>	Input	WORD	I, Q, M, D, L or constant	Value with which the counter is preset in the BCD format. (C#0 to C#999)
<Counter>	InOut/Input	COUNTER	C	Counter that is preset

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

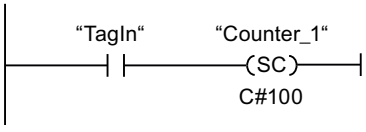

The counter "Counter_1" is preset with the value "100" when the signal state of the operand "TagIn" changes from "0" to "1".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on LAD (Page 7121)

---(CU): Count up

Description

With the "Count up" instruction you can increment the value of the specified counter by one if there is a positive signal edge in the result of logic operation (RLO). The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

The "Count up" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter whose value is incremented.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

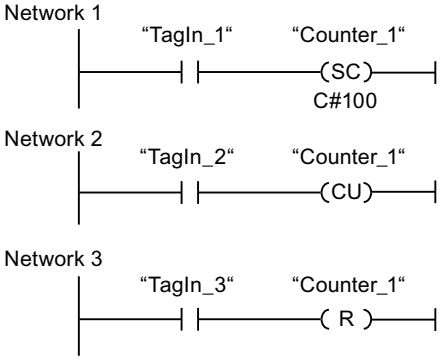

When the signal state of the "TagIn_1" operand changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of the counter "Counter_1" is incremented by one when the signal state of the operand "TagIn_2" changes from "0" to "1".

When the operand "TagIn_3" has the signal state "1", the value of the counter "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on LAD (Page 7121)

---(CD): Count down

Description

With the "Count down" instruction you can increment the value of the specified counter by one if there is a positive signal edge in the result of logic operation (RLO). The counter value can be decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer changed on a positive signal edge.

The "Count down" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter whose value is decremented.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

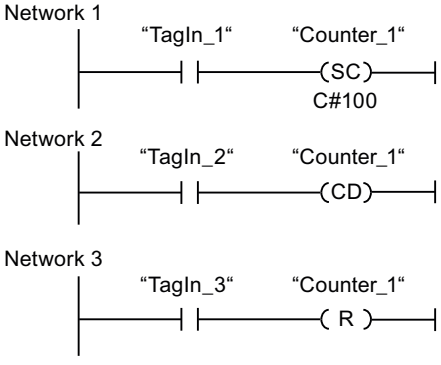

When the signal state of the "TagIn_1" operand changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of the counter "Counter_1" is decremented by one when the signal state of the operand "TagIn_2" changes from "0" to "1".

When the operand "TagIn_3" has the signal state "1", the value of the counter "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on LAD (Page 7121)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to determine if a first comparison value (<Operand1>) is equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Note

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, instead of the instruction "CMP ==: Equal", use the instruction "IN_RANGE: Value within range".

Parameters

The following table shows the parameters of the "Equal" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:

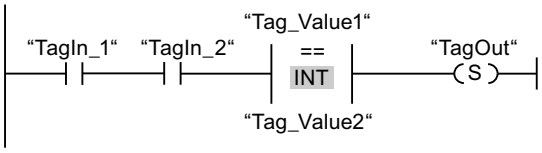

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" = "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to determine if a first comparison value (<Operand1>) is not equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Parameters

The following table shows the parameters of the "Not equal" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

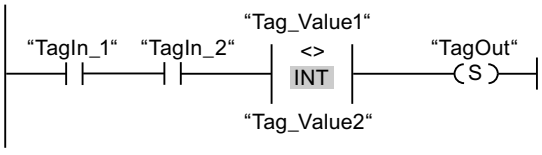

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if ("Tag_Value1" <> "Tag_Value2").

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

Basic information on the status word (Page 173)

Basic information on LAD (Page 7121)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to determine if a first comparison value (<Operand1>) is greater than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) in the operand placeholder below the instruction.

Parameters

The following table shows the parameters of the "Greater or equal" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

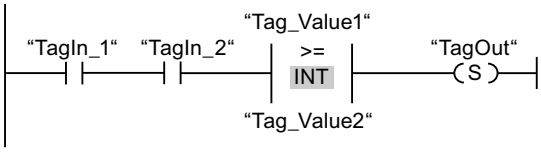

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if ("Tag_Value1" >= "Tag_Value2").

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to determine if a first comparison value (<Operand1>) is less than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Parameters

The following table shows the parameters of the "Less or equal" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

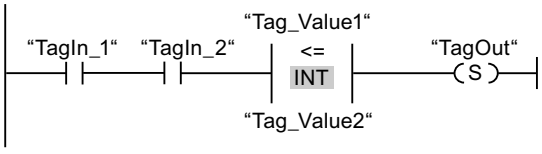

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if ("Tag_Value1" <= "Tag_Value2").

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

CMP >: Greater than

Description

You can use the "Greater than" instruction to determine if a first comparison value (<Operand1>) is greater than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Parameters

The following table shows the parameters of the "Greater than" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

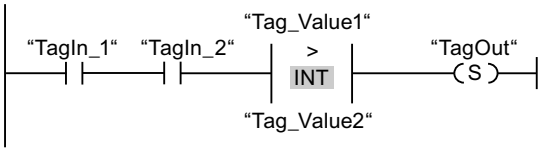

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if ("Tag_Value1" > "Tag_Value2").

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

CMP <: Less than

Description

You can use the "Less than" instruction to determine if a first comparison value (<Operand1>) is less than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Parameters

The following table shows the parameters of the "Less than" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

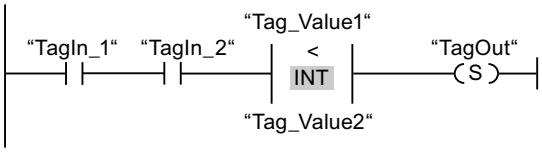

The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if ("Tag_Value1" < "Tag_Value2").

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

Math functions

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 and the value at input IN2 and query the sum at output OUT (OUT := IN1+IN2). A maximum of 2 inputs can be specified.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Add" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	First number to be added

Parameters	Declaration	Data type	Memory area	Description
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Second number to be added
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Sum

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:

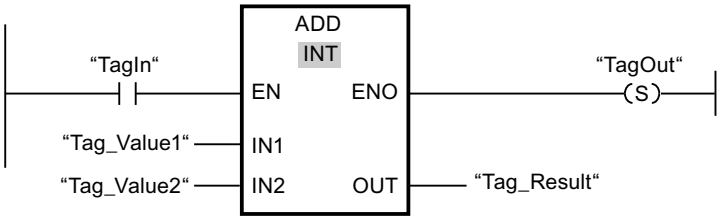

If operand "TagIn" has the signal state "1", the "Add" instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at output OUT (OUT := IN1-IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Subtract" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Subtracting
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Difference

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

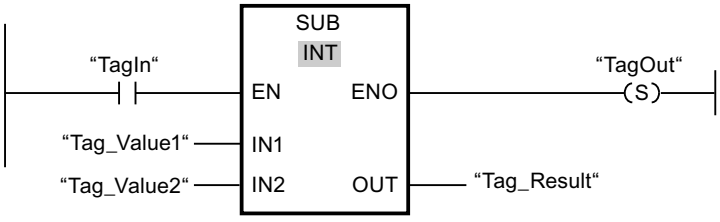

If operand "TagIn" has the signal state "1", the "Subtract" instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at input IN1 with the value at input IN2 and query the product at output OUT (OUT := IN1*IN2). A maximum of 2 inputs can be specified.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Multiply" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Multiplier
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Number being multiplied
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Product

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

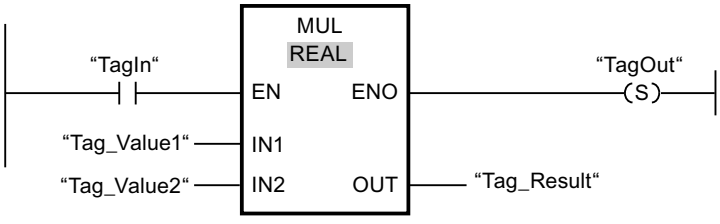

If operand "TagIn" has the signal state "1", the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The result of the multiplication is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT (OUT := IN1/IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Divide" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Quotient value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

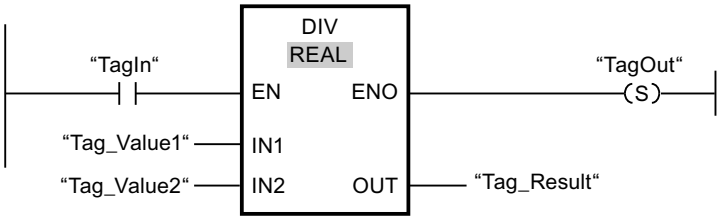

If operand "TagIn" has the signal state "1", the "Divide" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The division result is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

LAD programming examples (Page 7172)

Basic information on LAD (Page 7121)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at input IN1 by the value at input IN2 and query the remainder of division at output OUT.

Parameters

The following table shows the parameters of the "Return remainder of division" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	DINT	I, Q, M, D, L, P, or constant	Dividend
IN2	Input	DINT	I, Q, M, D, L, P, or constant	Divisor
OUT	Output	DINT	I, Q, M, D, L, P	Remainder of division

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

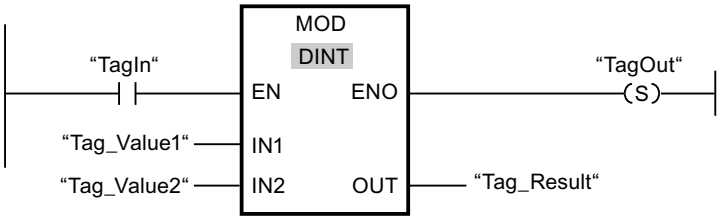

If operand "TagIn" has the signal state "1", the "Return remainder of division" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder of division is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at the IN input and query the result at the OUT output. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output (only valid for integers).

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers, floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

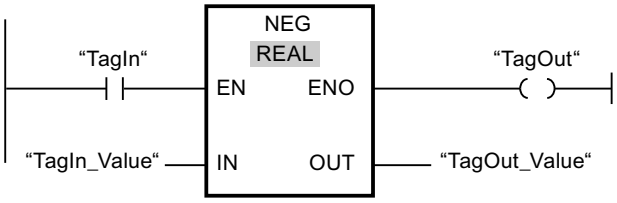

If operand "TagIn" has the signal state "1", the "Create twos complement" instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is provided at output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

ABS: Form absolute value

Description

You can use the "Form absolute value" instruction to calculate the absolute value of the value specified at input IN. The result of the instruction is sent to the OUT output and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form absolute value":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

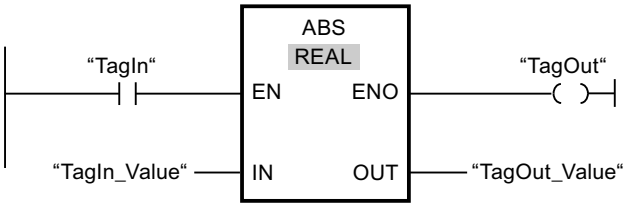

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

If operand "TagIn" has the signal state "1", the "Form absolute value" instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the lowest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters even with deactivated IEC check. If you use the data types WORD or DWORD at the input parameters, the instruction returns an undefined result.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Second input value
IN3	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Third input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

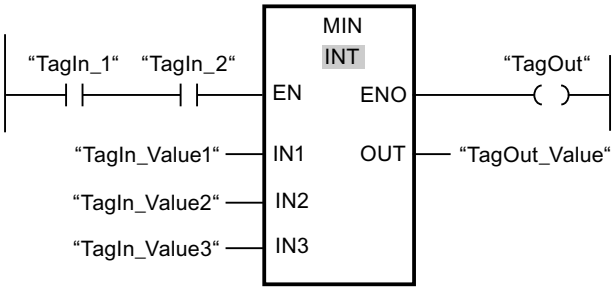

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

If the "TagIn_1" and "TagIn_2" operands have signal state "1", the "Get minimum" instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the highest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters even with deactivated IEC check. If you use the data types WORD or DWORD at the input parameters, the instruction returns an undefined result.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Second input value
IN3	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Third input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

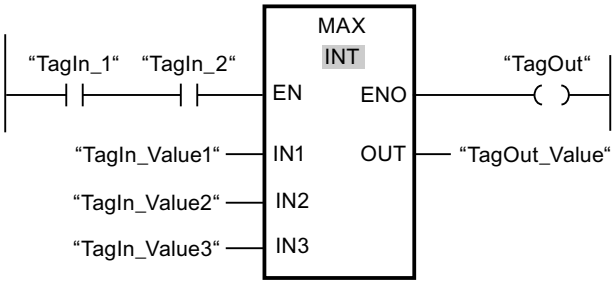

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12666
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

If the "TagIn_1" and "TagIn_2" operands have signal state "1", the "Get maximum" instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

LIMIT: Set limit value

Description

You use the "Set limit value" instruction to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN, the output OUT is set to the value of the MN input. If the high limit MX is exceeded, output OUT is set to the value of the input MX.

If the value at the MN input is greater than at the MX input, the result is undefined and the enable output ENO is "0".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at input MN is greater than the value at input MX.

Note

Only the INT, DINT, and REAL data types are permitted for the input parameters, even when the IEC check is disabled. If you use the data types WORD or DWORD at the input parameters, the instruction returns an undefined result.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MN	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Low limit
IN	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Input value
MX	Input	Integers, floating-point numbers	I, Q, M, D, L, P	High limit
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

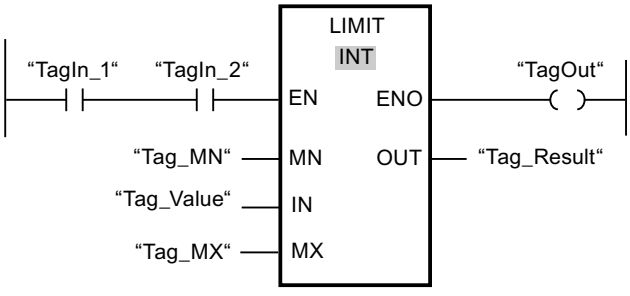

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Set limit value" instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value of operand "Tag_Value" is less than the low limit, the value of operand "Tag_MN" is copied to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

SQR: Form square

Description

The instruction "Form square" is used to square the value of a floating-point number at input IN and to write the result to output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form square":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

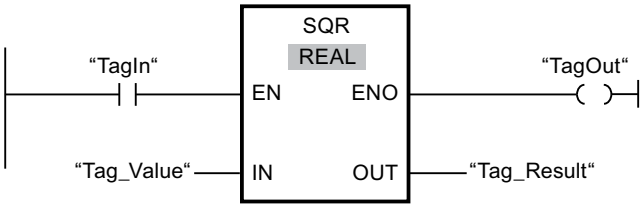

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If the operand "TagIn" has the signal state "1", the "Form square" instruction is executed. The instruction squares the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

SQRT: Form square root

Description

The instruction "Form square root" is used form the square root from the value of the floating-point number at input IN and to write the result to output OUT. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number. If the value at input IN is "0", the result is also "0".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form square root":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

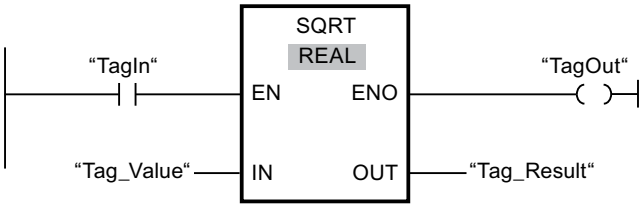

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If the operand "TagIn" has the signal state "1", the "Form square root" instruction is executed. The instruction calculates the square root of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) of the value at input IN. The result is sent to output OUT and can be queried there. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form natural logarithm":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

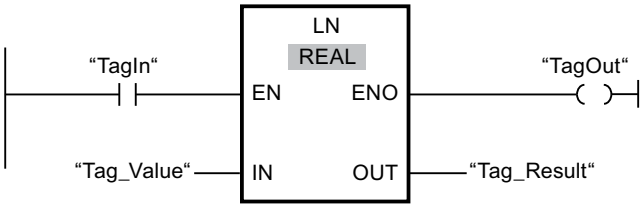

If the operand "TagIn" has the signal state "1", the "Form natural logarithm" instruction is executed. The instruction forms the natural logarithm of the value at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

EXP: Form exponential value

Description

You can use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value specified at input IN. The result is provided at output OUT and can be queried there (OUT = e^{IN}).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form exponential value":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Exponential value of input value IN

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

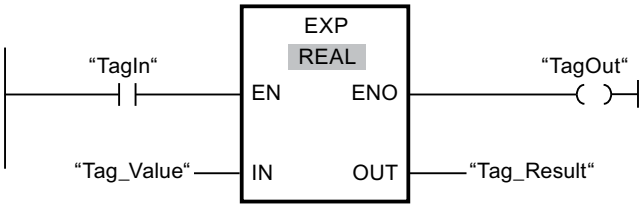

If the operand "TagIn" has the signal state "1", the "Form exponential value" instruction is executed. The instruction forms the exponent from base e and the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form sine value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Sine of the specified angle

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

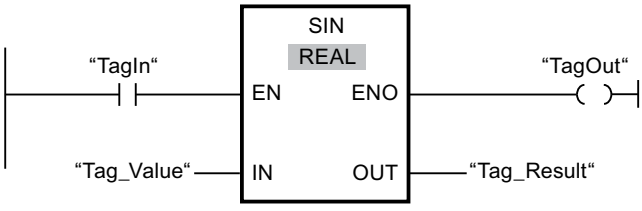

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

If operand "TagIn" has the signal state "1", the "Form sine value" instruction is executed. The instruction calculates the sine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form cosine value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Cosine of the specified angle

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

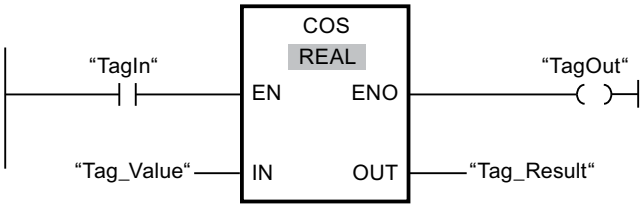

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

If operand "TagIn" has the signal state "1", the "Form cosine value" instruction is executed. The instruction calculates the cosine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the tangent of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameter

The following table shows the parameters of the "Form tangent value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Tangent of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

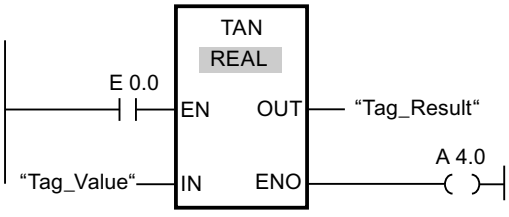

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the tangent of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from the arcsine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the "Form arcsine value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

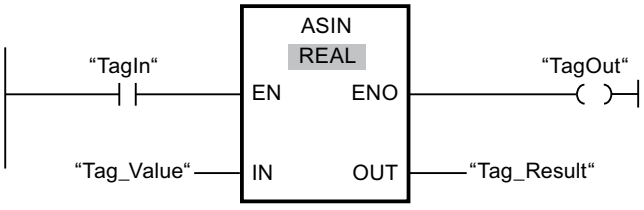

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

If operand "TagIn" has the signal state "1", the "Form arcsine value" instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at the

"Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from the cosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from 0 to +π.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the "Form arccosine value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

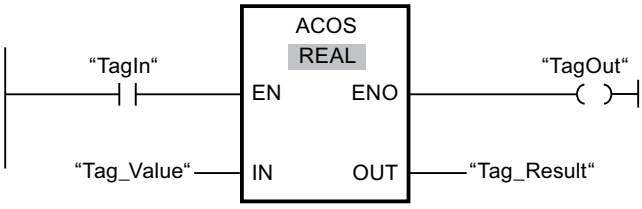

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

If operand "TagIn" has the signal state "1", the "Form arccosine value" instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from the tangent value specified at input IN, which corresponds to this value. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) at input IN. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form arctangent value":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Tangent value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

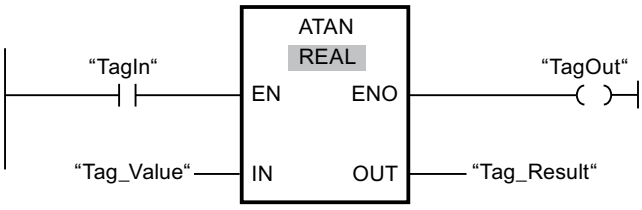

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398 (π/4)

If the operand "TagIn" has the signal state "1", the "Form arctangent value" instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Invalid floating-point numbers (Page 253)

Basic information on LAD (Page 7121)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the contents of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of the ascending address.

The following table shows the possible transfers:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD, CHAR
WORD	WORD; DWORD	BYTE, WORD, DWORD, INT, DINT, TIME, S5TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, INT, DINT, REAL, TIME, DATE, TOD, CHAR
INT	INT	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
TIME	TIME	BYTE, WORD, DWORD, INT, DINT, TIME
S5TIME	S5TIME	WORD, S5TIME
DATE	DATE	BYTE, WORD, DWORD, INT, DINT, DATE
TOD	TOD	BYTE, WORD, DWORD, INT, DINT, TOD
CHAR	CHAR	BYTE, WORD, DWORD, CHAR
COUNTER	INT, WORD, COUNTER	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	INT, WORD, TIMER	WORD, DWORD, INT, UINT, DINT, UDINT

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at the IN input is less than the bit length of the data type at the OUT1 output, then the more significant bits of the destination value will be filled with zeros.

You can also use the "Move block" (BLKMOV) and "Move block uninterruptible" (UBLKMOV) instructions to move fields and structures.

Parameters

The following table shows the parameters of the "Move value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings, integers, floating-point numbers, timers, DATE, TOD, CHAR, TIMER, COUNTER	I, Q, M, D, L or constant	Source value
OUT1	Output	Bit strings, integers, floating-point numbers, timers, DATE, TOD, CHAR, TIMER, COUNTER	I, Q, M, D, L	Destination address

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

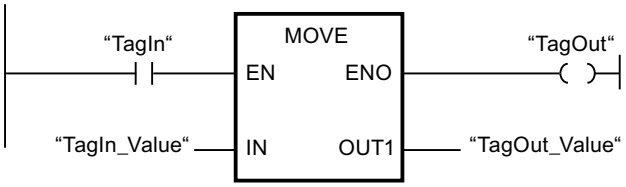

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If operand "TagIn" has the signal state "1", the "Move value" instruction is executed. The instruction copies the contents of operand "TagIn_Value" to operand "TagOut_Value" and sets output "TagOut" to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

BLKMOV: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:

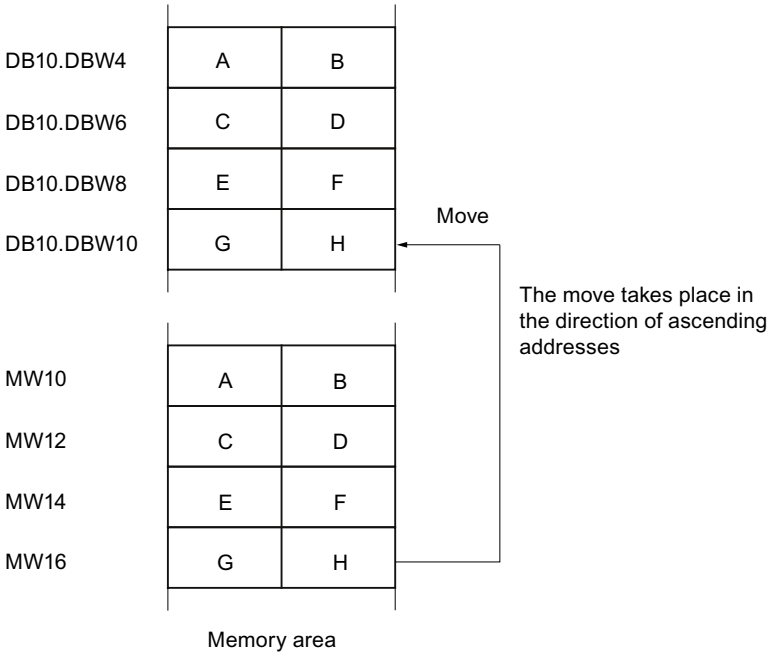

Consistency of the source data and the target data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, BLKMOV is interrupted while copying from a DB that is not runtime-relevant, the execution of BLKMOV can no longer be nested.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks not runtime-relevant

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the specified length of the area must be dividable by 8, otherwise it will not be possible to execute the instruction.

Note

If an actual source or destination area is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes to the SRCBLK and DSTBLK parameters.

Rules for moving data blocks that are not runtime-relevant

The source area is in a data block in load memory that is not runtime-relevant. Data blocks that are not runtime-relevant are indicated by the keyword UNLINKED.

If a non-runtime-relevant data block is copied to the work memory with the "Move block" instruction and loaded at the same time, for example, by the programming device, the execution of the instruction can be delayed for several milliseconds. This results in a longer OB cycle and may trip the cycle monitoring.

If a non-runtime-relevant data block is moved with the "Move block" instruction and the move operation is interrupted, the execution of the instruction can no longer be continued.

If the instruction Read from data block in the load memory is available on your CPU, you must use this instruction to read data blocks that are not runtime-relevant from load memory. If you use the "Move block" instruction, error W#16#8092 is output.

Parameters

The following table lists the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1]	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

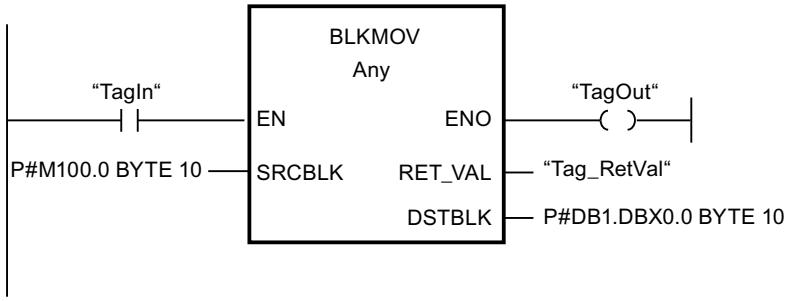

If the "TagIn" operand returns signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

The move operation cannot be interrupted by other operating system activities. As a result the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the specified length of the area must be dividable by 8, otherwise it will not be possible to execute the instruction.

You can use the "Move block uninterruptible" instruction to move a maximum of 512 bytes. Note the CPU-specific restrictions for this.

Note

If an actual source or destination area is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of actually moved characters. If areas of the data type STRING are moved, you have to specify "1" as area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1)]	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source or destination area is located in the load memory only.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

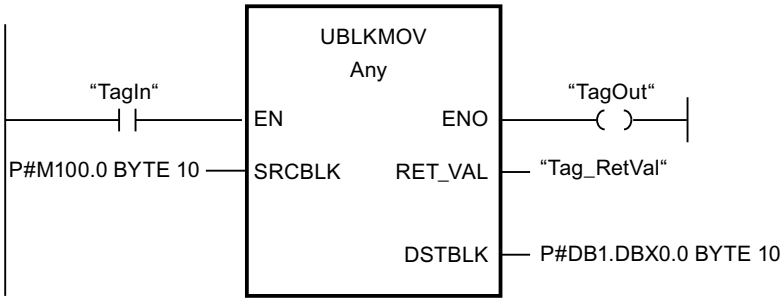

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

FILL: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:

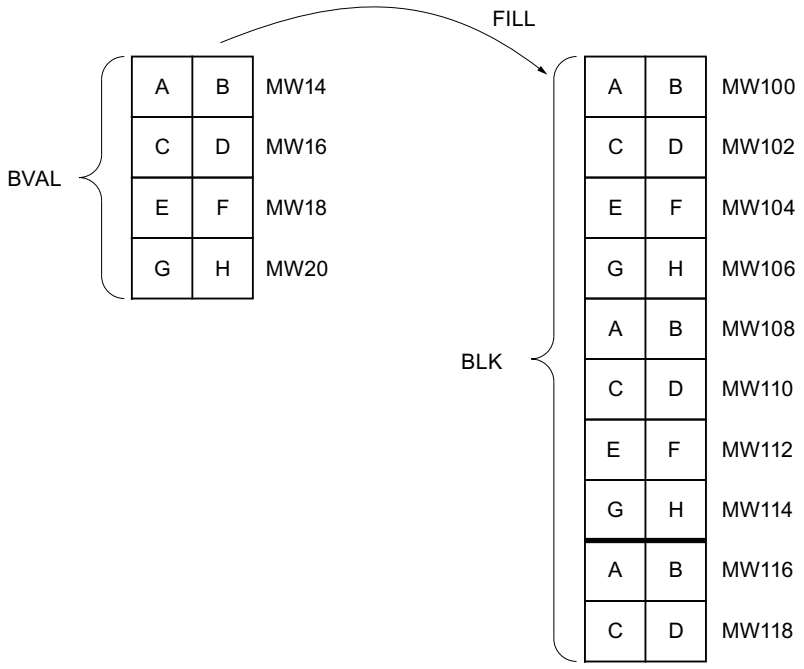

Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the input parameter BVAL, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is data type STRING, the instruction writes the entire string including the administration information.

Note

If an actual source or destination area is smaller than a destination or source area specified on the BVAL or BLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving structures

When you transfer a structure as an input parameter, you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
BVAL	Input	ANY	I, Q, M, D, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
BLK	Output [1]	ANY	I, Q, M, D, L, P	Specification of the memory area that will be filled with the content of the source area.
1) The BLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

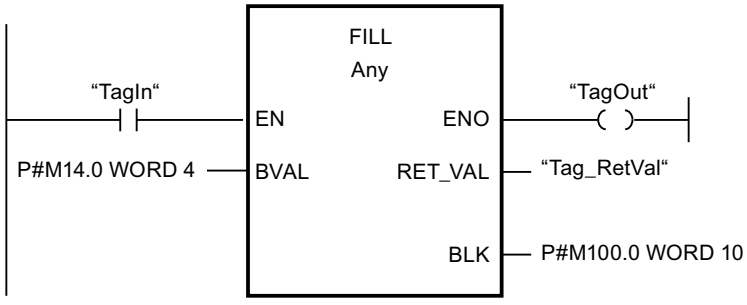

If the "TagIn" operand returns signal state "1", the instruction is executed. The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types selected in the instruction box. The converted value is provided at output OUT.

For information on possible conversions, refer to the "Explicit conversion" section at "See also".

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Convert value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P, or constant	Value to be converted.
OUT	Output	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

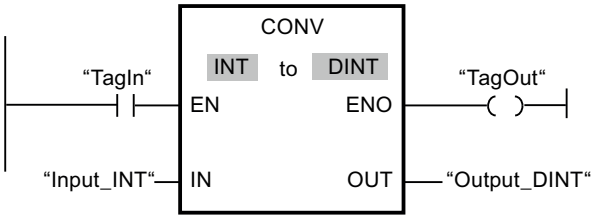

If operand "TagIn" has signal state "1", the content of the operand "TagIn_Value" is read and converted to an integer (32 bits). The result is stored in the operand "TagOut_Value". The output "TagOut" is set to "1" if the instruction was executed without errors.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Explicit conversions (Page 569)

Basic information on LAD (Page 7121)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to an integer of data type DINT. If the input value is exactly between an even and odd number, the even number is selected. The result of the instruction is sent to the OUT output and can be queried there.

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P, or constant	Input value to be rounded.
OUT	Output	DINT	I, Q, M, D, L, P	Result of rounding

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

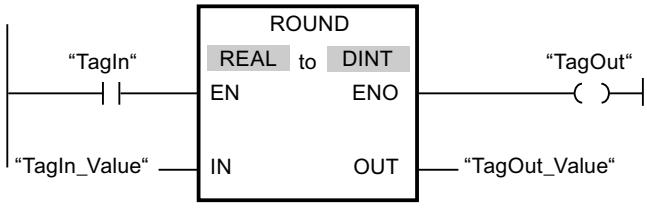

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

CEIL: Generate next higher integer from floating-point number

Description

You can use the "Generate next higher integer from floating-point number" instruction to round the value at input IN to the next higher integer. The instruction interprets the value at the IN input as a floating-point number and converts this number to the next higher integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be greater than or equal to the input value.

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Generate next higher integer from floating-point number" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next higher integer

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

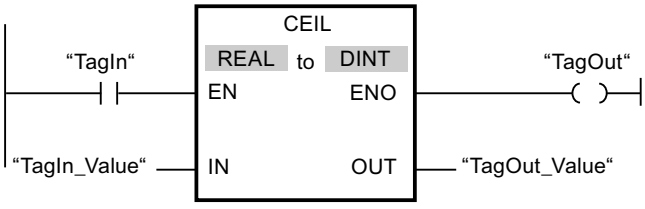

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at the "TagIn_Value" input is rounded to the next higher integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

FLOOR: Generate next lower integer from floating-point number

Description

You can use the "Generate next lower integer from floating-point number" instruction to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be less than or equal to the input value.

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Generate next lower integer from floating-point number" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next lower integer

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

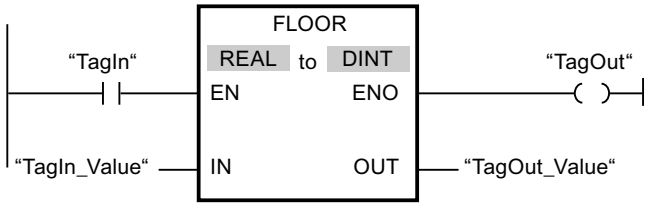

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at the IN input. The value at the IN input is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to the OUT output without decimal places.

Enable output ENO has signal state "0" if one of the following conditions applies:

- Enable input EN has signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Truncate numerical value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Integers	I, Q, M, D, L	Integer part of the input value

You can select the data type for the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

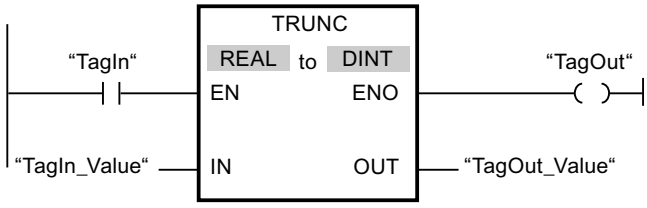

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The integer part of the floating-point number at the "TagIn_Value" input is converted to an integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on LAD (Page 7121)

SCALE: Scale

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number, which can be scaled in physical units between a low limit and a high limit. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit (LO_LIM) and an error is output.

When the indicated low limit is greater than the high limit (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	INT	I, Q, M, D, L, P, or constant	Input value to be scaled
HI_LIM	Input	REAL	I, Q, M, D, L, P, or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P, or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"

Error code* (W#16#...)	Explanation
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on switching display formats, refer to "See also".	

Example

The following example shows how the instruction works:

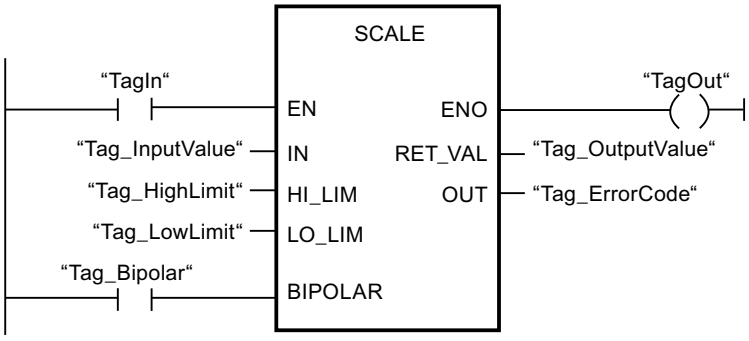

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

UNSCALE: Unscale

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at parameter IN is not within the limits defined by HI_LIM and LO_LIM, an error is output and the result is set to the nearest limit.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Input	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

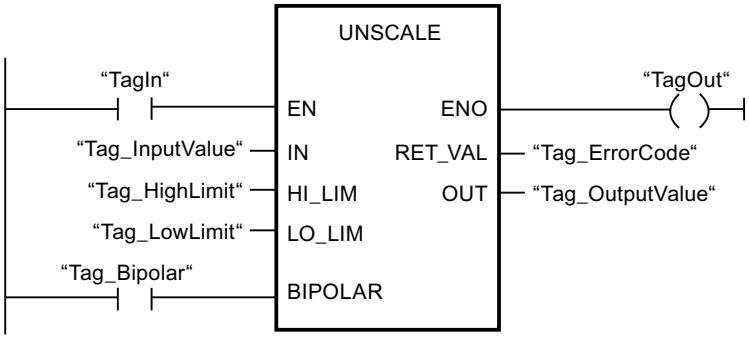

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

Program control operations

---(JMP): Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil is permitted within a network.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:

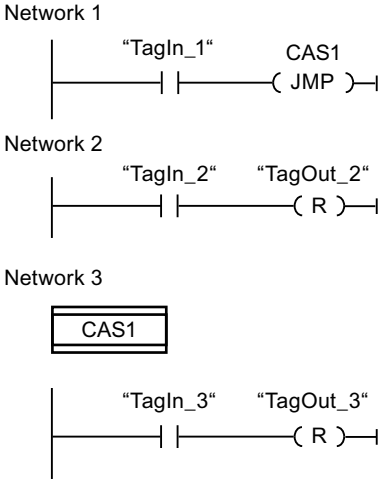

If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has signal state "1", the "TagOut_3" output is set.

See also

Basic information on LAD (Page 7121)

---(JMPN): Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil is permitted within a network.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of logic operation at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:

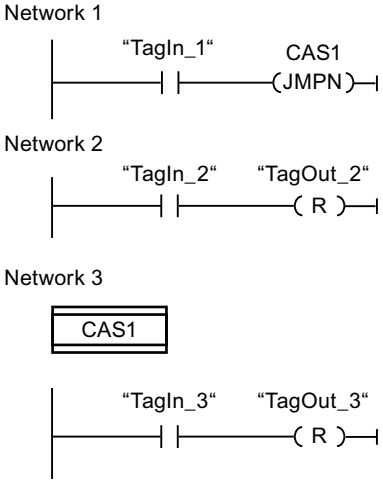

If operand "TagIn_1" has the signal state "0", the "Jump if RLO = 0" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has signal state "1", the "TagOut_3" output is set.

See also

Basic information on LAD (Page 7121)

LABEL: Jump label

Description

You can use a jump label to identify a destination network, in which the program execution should resume when a jump is executed.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

The following grammar rules must be followed for the jump label:

- Letters (a - z, A - Z)
- A combination of letters and numbers. Check that the order is correct, i.e. first the letters, then the numbers (a - z, A - Z, 0 - 9).
- Special characters or a combination of letters and numbers in reverse order, i.e. first the numbers and then the letters (0-9, a - z, A - Z), must not be used.

Example

The following example shows how the instruction works:

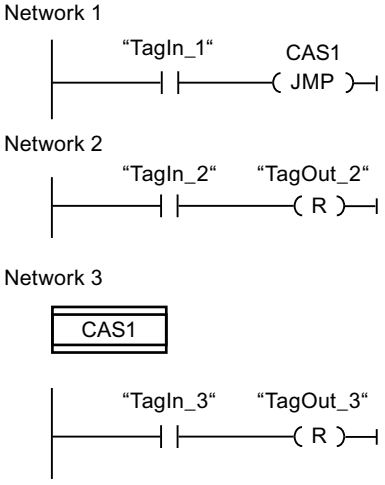

If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has signal state "1", the "TagOut_3" output is set.

See also

Basic information on LAD (Page 7121)

--(RET): Return

Description

You use the "Return" instruction to end the execution of a program block.

- If the result of logic operation (RLO) at the input of the "Return" instruction is "1", the program execution is ended in the currently called program block and continued after the call function in the calling program block (for example, in the calling organization block). The signal state of the enable output ENO of the call function is determined by the RLO. The RLO is displayed as a fixed parameter value above the instruction.
- If the result of logic operation (RLO) at the input of the instruction is "0", then the instruction will not be executed. The program execution is continued in the next network of the called program block.

If the instruction is programmed in the last network of the called block, the enable output ENO of the call function is set to the current signal state of the RLO.

If an organization block is ended with the instruction, the CPU goes on working in the system program.

Example

The following example shows how the instruction works:

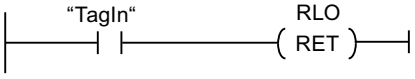

If the "TagIn" operand returns signal state "1", the instruction is executed. Program execution is ended in the called program block and continues in the calling program block. The ENO enable output of the call function is set to signal state "1".

See also

Basic information on LAD (Page 7121)

---(OPN): Open global data block

Description

You can use the "Open global data block" instruction to open a global data block (DB). The number of the data block is transferred to the DB register. Subsequent DB commands access the relevant blocks depending on the content of the register.

Parameters

The following table shows the parameters of the "Open global data block" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Data block>	Input	DB_ANY	DB	Data block to be opened.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

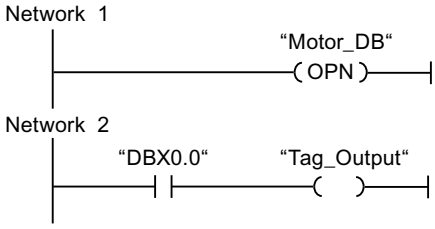

The "Motor_DB" data block is called in network 1. The number of the data block is transferred to the DB register. The "DBX0.0" operand is queried in network 2. The signal state of the DBX0.0 operand is assigned to the Tag_Output operand.

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

---(OPNI): Open instance data block

Description

You can use the "Open instance data block" instruction to open a data block (DB). The number of the data block is transferred to the DI register. Subsequent DI commands access the relevant blocks depending on the content of the register.

Parameters

The following table shows the parameters of the "Open instance data block" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Data block>	Input	DB_ANY	DB	Data block to be opened

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

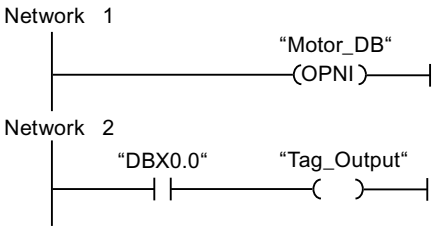

The "Motor_DB" data block is called in network 1. The number of the data block is transferred to the DI register. The "DBX0.0" operand is queried in network 2. The signal state of the DBX0.0 operand is assigned to the Tag_Output operand.

See also

Overview of the valid data types (Page 231)

Basic information on LAD (Page 7121)

Runtime control

COMPRESS: Compress CPU memory (S7-400)

Description

The "Compress CPU memory" instruction is used to compress the working memory of the CPU or the RAM work memory. During this compression the gaps caused by deleting or loading blocks are filled by pushing together the existing blocks.

The compression operation is started by calling the "Compress CPU memory" instruction. The compression operation is divided into several program cycles. The compression operation is monitored by the values of the individual parameters. If value "1" is on the BUSY parameter of the instruction, the compression operation is still being executed. The value "1" on the DONE parameter reports that the compression operation has been completed. Error information can be queried on the RET_VAL parameter.

If an externally started compression operation has already been executed, the calling of the "Compress CPU memory" instruction causes an error to be displayed.

Parameter

The following table shows the parameters of the "Compress CPU memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output on the RET_VAL parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Information whether compression operation is active.
DONE	Output	BOOL	I, Q, M, D, L	Information whether compression operation was completed.

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error The status of the compression operation can be queried on the BUSY and DONE parameters.
8091	An externally started compression operation is already being executed.
8092	The operation cannot be executed, because one of the following conditions applies: • The "Delete block" function is active. • A test and commissioning function affects a block (for example, status). • The "copy blocks" function was triggered externally and is active. • The H-CPU is running a link-up or update function.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

RE_TRIGR: Restart cycle monitoring time

Description

The "Restart cycle monitoring time" instruction is used to restart the cycle monitoring time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called in all blocks regardless of the priority .

If the instruction is called in a block with a higher priority, for example a hardware interrupt or a diagnostics interrupt, the instruction is not executed and the enable output ENO is set to signal state "0".

The "Restart cycle monitoring time" instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Parameters

The instruction "Restart cycle monitoring time" has no parameters.

See also

Basic information on LAD (Page 7121)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and thus end the program execution. The effects at a transition from RUN to STOP depend on the CPU configuration.

When the result of logic operation (RLO) at the input of the instruction is "1", the CPU changes to STOP mode and the program execution is ended. The signal state at the output of the instruction is not evaluated.

When the RLO is "0" at the input of the instruction, then the instruction will not be executed.

Parameters

The "Exit program" instruction has no parameters.

See also

Basic information on LAD (Page 7121)

WAIT: Configure time delay

Description

The "Configure time delay" instruction pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to 32767 microseconds (µs). The smallest possible delay time depends on the CPU and corresponds to the execution time of the instruction.

The execution of the instruction can be interrupted by higher priority events and does not return any error information.

Applies to S7-300-CPUs:

For the S7-300-CPUs (except S7-318), the delay time programmed by the instruction is a minimum time. It is extended by the execution time of the nested priority classes and by system utilizations.

Note

Negative delay time

If you specify a negative delay time at parameter WT, the enable output ENO, or the RLO and the BR bit, will return the signal state FALSE. A negative delay time does not affect the CPU. The following instructions linked to enable output ENO are not executed in LAD or FBD.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
WT	Input	INT	I, Q, M, D, L, P, or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

PROTECT: Change protection level

Description

The instruction "Change protection level" is used to change the configured protection level "1" of the CPU.

The configured protection level "1" with selected option "Removable with password", means that you have to configure a password. This password enables you to write-access during operation the CPU which was read-only protected with the instruction "Change Protection Level" (e. g. Load).

You specify the protection level on the MODE parameter of the instruction. The current protection level of your CPU is displayed in the inspector window under "Properties > Protection".

The following table shows the meaning of the protection levels.

Protection level	Meaning
1	All programming device functions are permitted. Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
2	• The program and the configuration of the CPU cannot be changed. The program in the CPU can be read. • The functions for process control, process monitoring, and process communication are permitted. • All informational functions are permitted. • If you know the valid password you can cancel the set write protection. • Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
3	• You cannot cancel the set read and write protection, even if you know the valid password. • If a valid connection exists at the time when the instruction "Change protection level" is called, the call of the instruction does not affect this connection.

If you have configured protection level "2" or "3" for your CPU, calling the "Change protection level" instruction has no effect.

You can use the SSL parts list with the SSL ID W#16#0232 and index W#16#0004 to check whether protection level "2" or "3" is still in effect after the execution of the "Change protection level" instruction.

Note

You can only use the "Change protection level" instruction to set a protection level that is equal to or higher than the protection level you have configured with STEP 7 "Configure hardware. You have, for example, configured protection level 3 and then called the instruction "Change protection level" with the MODE parameter = 12. This blocks the password authentication. If the instruction is called again with the parameter MODE = 0 or 1, protection level 3 is retained and the read and write protection canceled by entry of the valid password.

Effects of the operating state transitions on the protection level

The following table shows what effects the operating state transitions have on the protection level set with the "Change protection level" instruction.

Action	Effects on the protection level
• Operating mode switch to "STOP" • Unbuffered POWER ON • Memory reset with operating mode switch • Configuration and/or programming changes in the "STOP" mode of the S7-400 CPU	The protection level is set to "1". All programming device functions are permitted.
• Cold restart • Warm restart	The protection level is set by the operating system to "1". All programming device functions are permitted. If required, you can set the protection level in your program to "2" or "3" by calling the "Change protection level" instruction.
Buffered POWER ON	Remains unchanged
Operating mode change RUN/STARTUP/HOLD -> STOP (by calling the instruction "Exit program" (STP), due to a program error without corresponding error OB or by an operator action).	Remains unchanged
S7-400: Restart by user action or automatically following POWER ON	Remains unchanged

Parameter

The following table shows the parameters of the "Change protection level" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MODE	Input	WORD	I, Q, M, D, L or constant	Protection level Possible values: • W#16#0000: Setting protection level "1" • W#16#0001: Setting protection level "2" with password authentication • W#16#000C: Setting protection level "3" with password authentication Constants can also be specified on the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs during processing, information on the error is output.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error
8090	Invalid value at parameter MODE.
80C3	The required operating resources are currently occupied.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

CiR: Control CiR process (S7-400)

Description

The "Control CiR process" instruction is used to influence the CiR process in the user program as follows:

- Disable CiR process. In this case, the download of a modified configuration from the programming device to the CPU is always rejected. The disable is maintained until you cancel it again with the "Control CiR process" instruction.
- You can disable the CiR process conditionally by specifying a high limit for the CiR synchronization time on the FRZ_TIME parameter. In this case, the download of a modified configuration from the programming device to the CPU is only accepted if the CiR synchronization time calculated in the CPU does not exceed the specified high limit.
- Disable CiR process. The high limit of the CiR synchronization time is set to the default value 1000 ms.
- Determine whether the CiR process is enabled or not. If the CiR process is enabled or conditionally enabled, you will get the current high limit for the CiR synchronization time in the A_FT parameter of the instruction.

Note

During the CiR synchronization period the outputs are frozen and the inputs are not evaluated.

Parameters

The following table shows the parameters of the "Control CiR process" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Valid values on the MODE parameter are: • 0: Information function • 1: Enable CiR process • 2: Disable CiR process • 3: Disable CiR conditionally. You specify the high limit of the CiR synchronization time in FRZ_TIME.
FRZ_TIME	Input	TIME	I, Q, M, D, L or constant	High limit of the CiR synchronization time in milliseconds. Permitted range of values: 60 ... 2500 ms (default: 1000 ms) The FRZ_TIME parameter is only relevant when the value 3 is set on the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	Error information When the MODE parameter is "0", the RET_VAL parameter supplies information on the enabling of the CiR process.
A_FT	Output	TIME	I, Q, M, D, L	Currently valid high limit of the CiR synchronization time

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error This error code can be output only if the value "1", "2" or "3" is set on the MODE parameter.
0001	CiR is enabled. This error code can be output only if the value "0" is set on the MODE parameter.
0002	The CiR process is disabled. This error code can be output only if the value "0" is set on the MODE parameter.
0003	CiR is disabled conditionally. This error code can be output only if the value "0" is set on the MODE parameter.

Error code* (W#16#...)	Explanation
8001	The CPU is not ready for a CiR operation. You are using an H CPU in an H-system (stand-alone), or you are working with a standard CPU in multicomputing mode.
8002	The value at the MODE parameter is invalid.
8003	The value at the FRZ_TIME parameter is invalid.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The "Control CiR process" instruction can be used, for example, to ensure that no CiR process is initiated during periods in which maximum CPU performance is required for the execution of the process.

To use the "Control CiR process" for this, the instruction must be called before the start of the period with increased process activity and executed with the value "2" (disable CiR process) on the MODE parameter. The CiR process is then disabled.

To enable the CiR process again after the completion of the period with increased process activity, you have to again call the "Control CiR process" instruction and execute it with the value "1" on the MODE parameter. Alternatively, you can also set the value 3 on the MODE parameter to disable the CiR process conditionally.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

Word logic operations

AND: AND logic operation

Description

You can use the "AND logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by AND logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are logically ANDed. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

The result bit has signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "AND logic operation" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P, or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P, or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

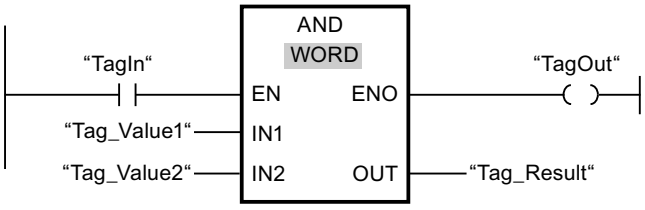

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0000 0000 0000 0101

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ANDed. The result is mapped bit-for-bit and output in operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bit for word logic operations (Page 182)

Basic information on LAD (Page 7121)

OR: OR logic operation

Description

You can use the "OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are combined by OR logic operation. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "OR logic operation" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P, or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P, or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

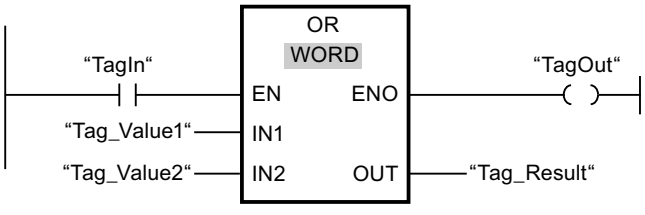

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1111

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ORed. The result is mapped bit-for-bit and output in operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bit for word logic operations (Page 182)

Basic information on LAD (Page 7121)

XOR: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by EXCLUSIVE OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are combined by EXCLUSIVE OR logic operation. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

The result bit has signal state "1" when one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P, or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P, or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

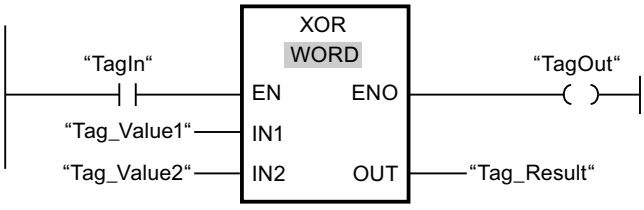

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1010

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are combined by EXCLUSIVE OR logic. The result is mapped bit-for-bit and output in operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bit for word logic operations (Page 182)

Basic information on the status word (Page 173)

Inserting LAD elements (Page 7134)

Selecting the data type of a LAD element (Page 7138)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

Basic information on LAD (Page 7121)

INVERT: Create ones complement

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits at input IN. When the instruction is processed, the value at the IN input and a hexadecimal template (W#16#FFFF for 16-bit numbers or DW#16#FFFFFFFF for 32-bit numbers) are combined by EXCLUSIVE OR logic. This inverts the signal state of the individual bits that are then stored at output OUT.

Parameters

The following table shows the parameters of the instruction "Create ones complement":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers	I, Q, M, D, L, P, or constant	Input value
OUT	Output	Integers	I, Q, M, D, L, P	Ones complement of the value at input IN

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

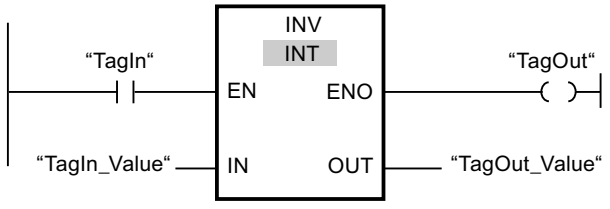

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	W#16#000F
OUT	TagOut_Value	W#16#FFF0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction inverts the signal state of the individual bits at input "TagIn_Value" and writes the result to output "TagOut_Value". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bit for word logic operations (Page 182)

Basic information on LAD (Page 7121)

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value that is specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. When the value at input IN is greater than 31, a modulo 32 instruction is executed.

Parameters

The following table shows the parameters of the "Decode" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P, or constant	Position of the bit in the output value which is set.
OUT	Output	DWORD	I, Q, M, D, L, P	Output value

Example

The following example shows how the instruction works:

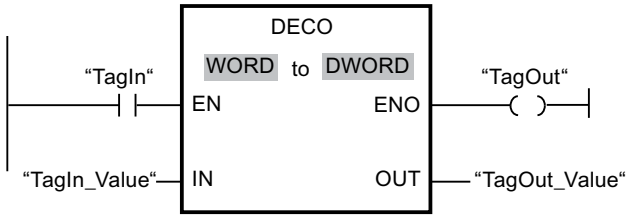

The following figure shows how the instruction works using specific operand values:

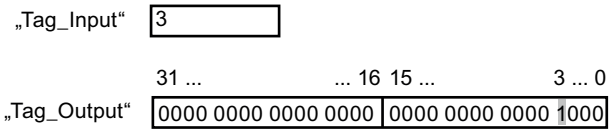

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction reads bit number "3" from the value of the operand "TagIn_Value" at the input and sets the third bit to the value of the operand "TagOut_Value" at the output.

If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bit for word logic operations (Page 182)

Basic information on LAD (Page 7121)

ENCO: Encode

Description

Use the "Encode" instruction to read the bit number of the least significant bit in the input value and to send it to the output OUT.

The "Encode" instruction selects the least significant bit of the value at the IN input and writes its bit number to the tag in the OUT output.

Parameters

The following table shows the parameters of the "Encode" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P, or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

Example

The following example shows how the instruction works:

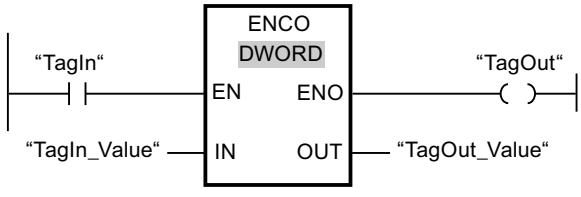

The following figure shows how the instruction works using specific operand values:

	31 16	15 ... 3 ... 0
“Tag_Input“	0000 1111 0000 0101	0000 1001 0000 1000
“Tag_Output“	3	

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction selects the least significant bit at the "TagIn_Value" input and writes bit position "3" to the tag in the "TagOut_Value" output.

If the instruction is executed without errors, the ENO enable output has signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bit for word logic operations (Page 182)

Basic information on LAD (Page 7121)

SEL: Select

Description

Depending on a switch (G input), the "Select" instruction selects one of the inputs, IN0 or IN1, and copies its content to the OUT output. When the G input has signal state "0", the value at the IN0 input is moved. When the G input has signal state "1", the value at the IN1 input is moved to the OUT output.

The instruction is only executed if the tags of all parameters are of the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
G	Input	BOOL	I, Q, M, D, L, T, C	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	First input value

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

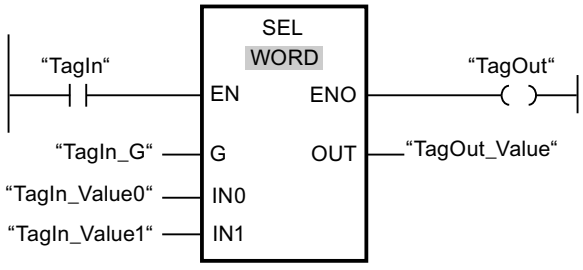

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If the "TagIn" operand has signal state "1", the instruction is executed. Based on the signal state at the "TagIn_G" input, the value at the "TagIn_Value0" or "TagIn_Value1" input is selected and moved to the "TagOut_Value" output. If the instruction is executed without errors, enable output ENO has signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bit for word logic operations (Page 182)

Basic information on LAD (Page 7121)

Shift and rotate

SHR: Shift right

Description

You can use the "Shift right" instruction to shift the content of the operand at the IN input bit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure shows how the content of an operand of the INT data type is shifted by four bit positions to the right:

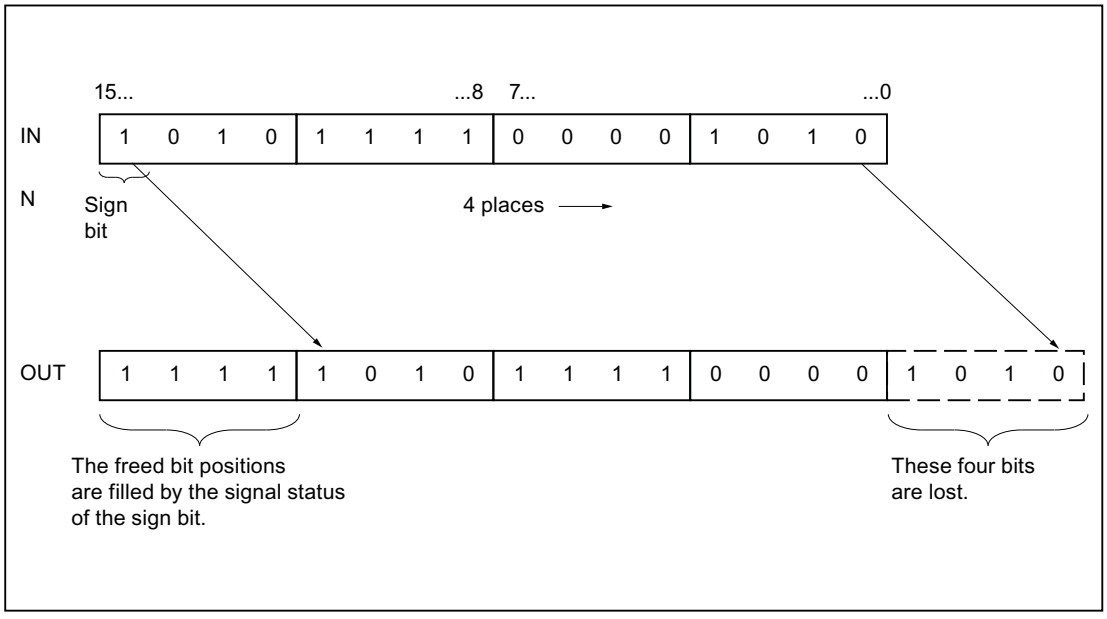

Parameters

The following table shows the parameters of the "Shift right" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L, P, or constant	Value to be shifted.
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P, or constant	Number of bit positions by which the value is shifted.
OUT	Output	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

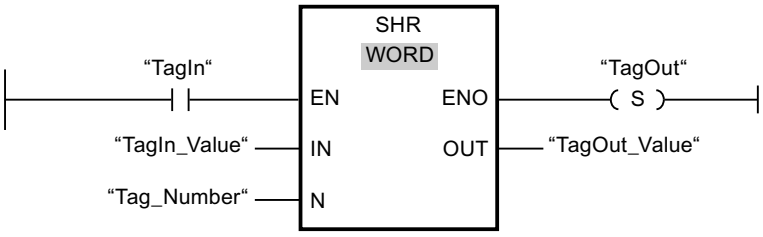

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If operand "TagIn" has the signal state "1", the "Shift right" instruction is executed. The content of the operand "TagIn_Value" is shifted three bit positions to the right. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on LAD (Page 7121)

SHL: Shift left

Description

You can use the "Shift left" instruction to shift the content of the operand at the IN input bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure shows how the content of an operand of the WORD data type is shifted by six bit positions to the left:

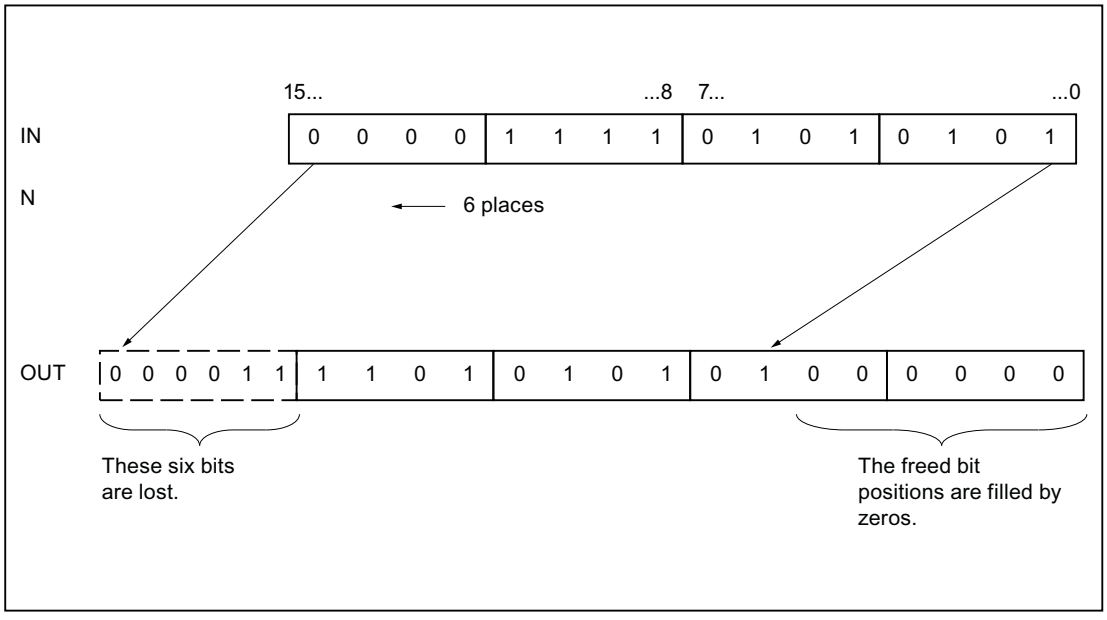

Parameters

The following table shows the parameters of the "Shift left" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L, P, or constant	Value to be shifted

Parameters	Declaration	Data type	Memory area	Description
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P, or constant	Number of bit positions by which the value is shifted
OUT	Output	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

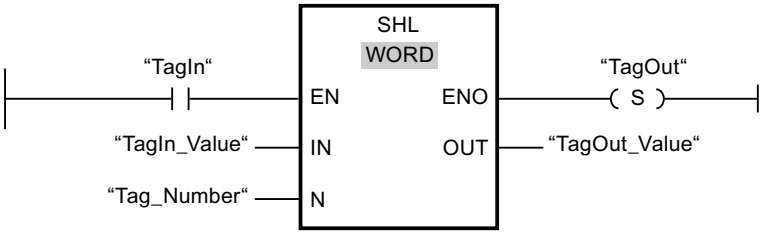

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If operand "TagIn" has the signal state "1", the "Shift left" instruction is executed. The content of the operand "TagIn_Value" is shifted four bit positions to the left. The result is sent at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on LAD (Page 7121)

ROR: Rotate right

Description

You can use the "Rotate right" instruction to rotate the content of the operand at the IN input bit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:

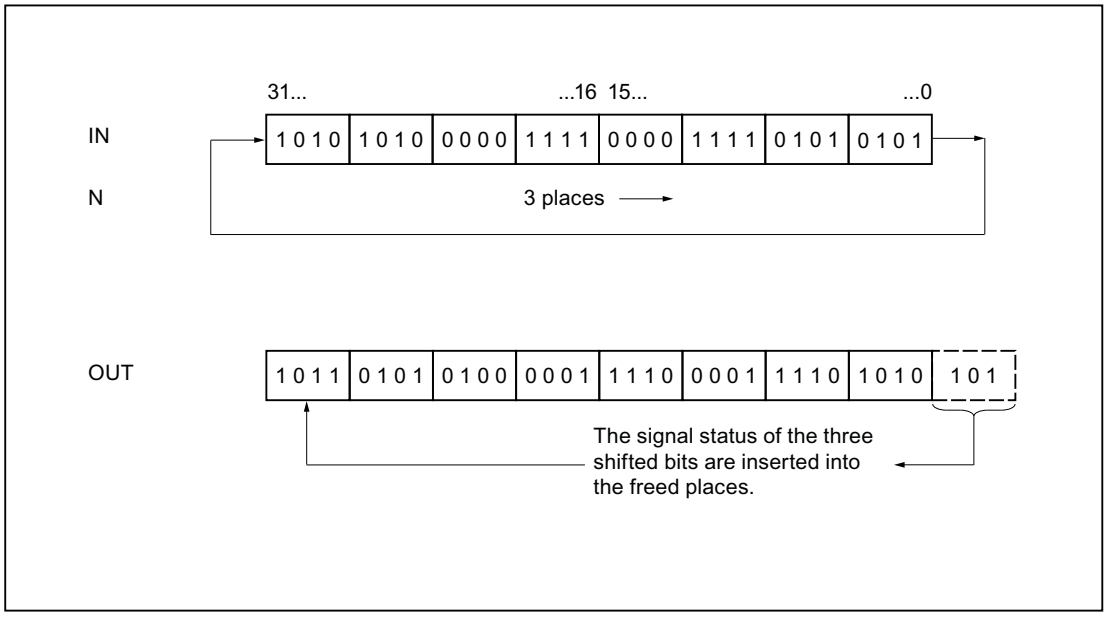

Parameters

The following table shows the parameters of the "Rotate right" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P, or constant	Value to be rotated
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P, or constant	Number of bit positions by which the value is rotated
OUT	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

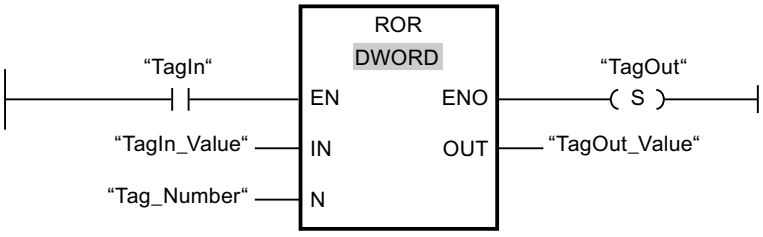

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	1010 1010 0000 1111 0000 1111 0101 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1101 0101 0000 0111 1000 0111 1010

If operand "TagIn" has the signal state "1", the "Rotate right" instruction is executed. The content of operand "TagIn_Value" is rotated five bit positions to the right. The result is sent at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on LAD (Page 7121)

ROL: Rotate left

Description

You can use the "Rotate left" instruction to rotate the content of the operand at the IN input bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:

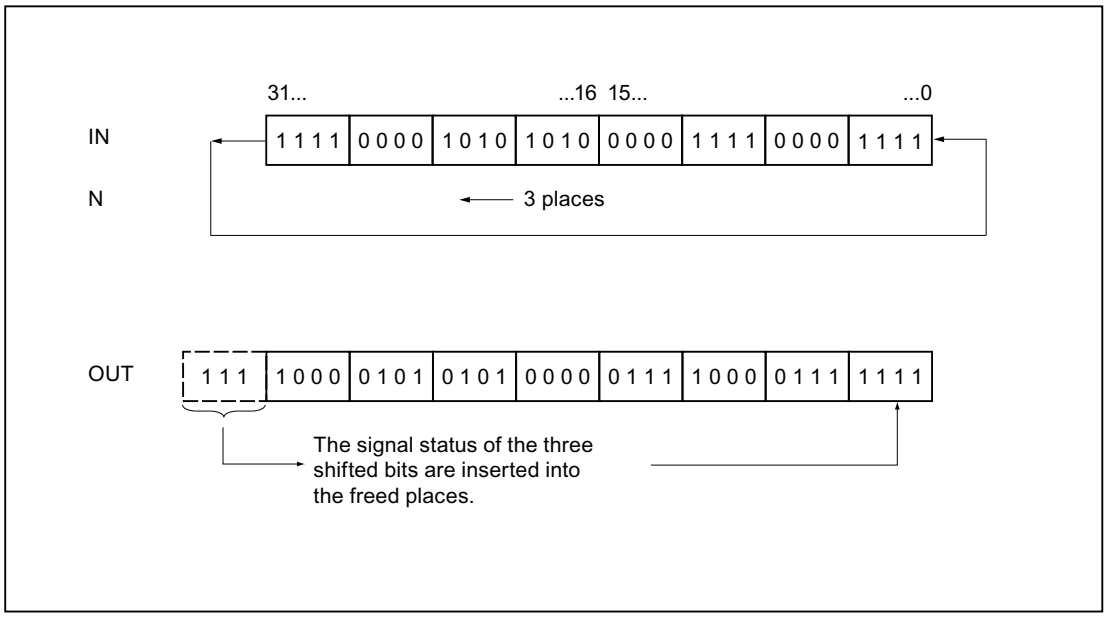

Parameters

The following table shows the parameters of the "Rotate left" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P, or constant	Value to be rotated
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P, or constant	Number of bit positions by which the value is rotated
OUT	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

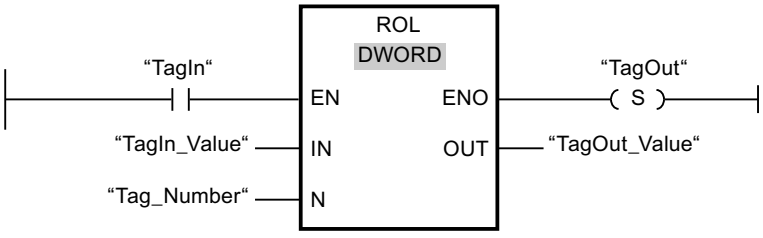

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	1111 0000 1010 1010 0000 1111 0000 1111
N	Tag_Number	5
OUT	TagOut_Value	0001 0101 0100 0001 1110 0001 1111 1110

If the "TagIn" input has the signal state "1", the "Rotate left" instruction is executed. The content of operand "TagIn_Value" is rotated five bit positions to the left. The result is sent at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on LAD (Page 7121)

Additional instructions

---| |---: Get status bit

Description

With the "Get status bit" instruction you can query the signal state of the following status bits:

- BR: Binary result
- OV: Overflow
- OS: Latching overflow
- CC 0 and CC 1: Conditional codes

You define which status bits are queried by selecting the corresponding symbol from the drop-down list of the instruction. The following table shows the possible ways of querying status bits:

Symbol	Query	Explanation
==0	CC 0 = 0 and CC 1 = 0	The instruction asks whether the result of a mathematical function is equal to 0.
<>0	CC 0 = 0 and CC 1 = 1 or CC 0 = 0 and CC 1 = 0	The instruction asks whether the result of a mathematical function is not equal to 0.

Symbol	Query	Explanation
>=0	CC 0 = 0	The instruction asks whether the result of a mathematical function is greater than or equal to 0.
<=0	CC 1 = 0	The instruction asks whether the result of a mathematical function is less than or equal to 0.
>0	CC 0 = 0 and CC 1 = 1	The instruction asks whether the result of a mathematical function is greater than 0.
<0	CC 0 = 1 and CC 1 = 0	The instruction asks whether the result of a mathematical function is less than 0.
UO	CC 0 = 1 and CC 1 = 1	The instruction asks whether the result of a mathematical function is invalid. The cause of an invalid result can be a floating-point number or division by zero.
OV	OV = 1	The instruction queries whether a number range overflow has occurred.
OS	OS = 1	The instruction queries whether a saving number range overflow has occurred.
BR	BR= 1	The instruction asks the signal state of the status bit BR.

The "Get status bit" instruction functions like a normally open contact. When the query is satisfied, the instruction has the signal state "1". If the query is not satisfied, the instruction has the signal state "0".

Example

The following example shows how the instruction works:

Network 1

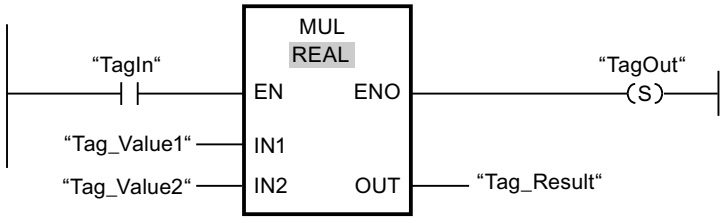

Network 2

If operand "TagIn" has the signal state "1", the "Multiply" (MUL) instruction is executed. In the event of an overflow during the execution of the "Multiply" instruction, the status bit OV is set. Following the query of the status bit OV, the "Set output" instruction (S) is executed in network 2 and the operand "TagOV" is set.

See also

Basic information on the status word (Page 173)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Setting status bits (Page 175)

---| / |---: Get negated status bit

Description

With the "Get negated status bit" instruction you can query the signal state of the following status bits and negate the query result:

- BR: Binary result
- OV: Overflow
- OS: Latching overflow
- CC 0 and CC 1: Conditional codes

You define which status bits are queried by selecting the corresponding symbol from the drop-down list of the instruction. The following table shows the possible ways of querying status bits:

Symbol	Query	Explanation
==0	CC 0 = 0 and CC 1 = 0	The instruction asks whether the result of a mathematical function is equal to 0.
<>0	CC 0 = 0 and CC 1 = 1 or CC 0 = 0 and CC 1 = 0	The instruction asks whether the result of a mathematical function is not equal to 0.
>=0	CC 0 = 0	The instruction asks whether the result of a mathematical function is greater than or equal to 0.
<=0	CC 1 = 0	The instruction asks whether the result of a mathematical function is less than or equal to 0.
>0	CC 0 = 0 and CC 1 = 1	The instruction asks whether the result of a mathematical function is greater than 0.
<0	CC 0 = 1 and CC 1 = 0	The instruction asks whether the result of a mathematical function is less than 0.

Symbol	Query	Explanation
UO	CC 0 = 1 and CC 1 = 1	The instruction asks whether the result of a mathematical function is invalid. The cause of an invalid result can be a floating-point number or division by zero.
OV	OV = 1	The instruction asks whether a number range overflow has occurred.
OS	OS = 1	The instruction asks whether a stored number range overflow has occurred in an instruction.
BR	BR= 1	The instruction asks the signal state of the status bit BR.

The "Get negated status bit" instruction functions like a normally closed contact. When the query is satisfied, the instruction has the signal state "0". If the query is not satisfied, the instruction has the signal state "1".

Example

The following example shows how the instruction works:

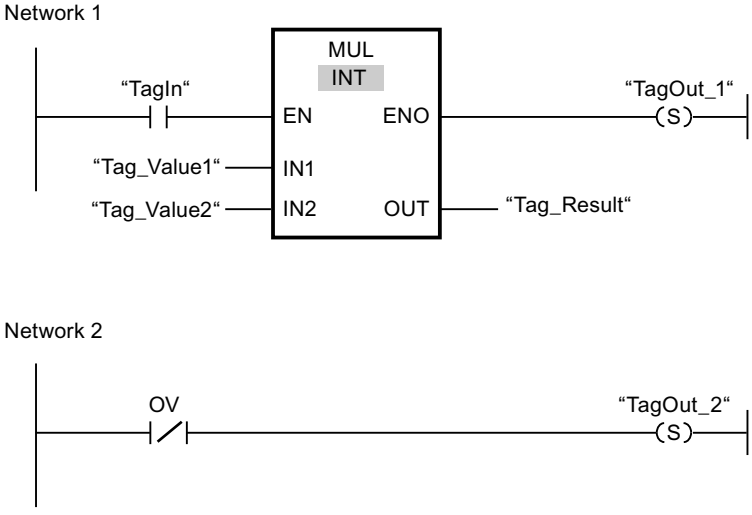

If operand "TagIn" has the signal state "1", the "Multiply" (MUL) instruction is executed. If an overflow does not occur during the execution of the instruction, the status bit OV is reset to "0". Following the query of the status bit OV, the instruction "Set output" (S) is executed in network 2 and the operand "TagOut_2" is set.

See also

Basic information on the status word (Page 173)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Setting status bits (Page 175)

---(CALL): Call block

Description

The "Call block" instruction calls a function (FC) without parameter. The call is conditional or unconditional depending on the preceding logic operation.

In the statement part of a function (FC), you cannot specify any parameter of the BLOCK_FC data type as the operand for a conditional call. You can, however, specify a parameter of the type BLOCK_FC as the operand in a function block (FB).

A conditional call is then only executed if the result of logic operation (RLO) is "1". If a conditional call is not executed the RLO is "0" after the call operation. If the operation is executed, it works as follows:

- It stores the return address of the calling block.
- It stores the two data block registers (data block and instance data block).
- It updates the local data for the called function.
- It moves the MA bit (active MCR bit) to the block stack (B stack).

Parameters

The following table shows the parameters of the "Call block" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Block>	Output	BLOCK_FC	-	Called function

Example

The following example shows how the instruction works:

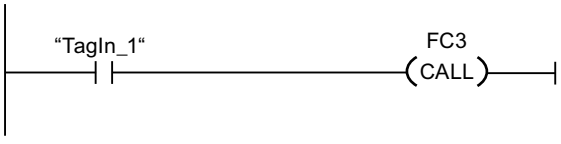

The function FC3 is called if the operand "TagIn_1" has the signal state "1".

See also

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

---(SAVE): Save RLO in BR bit

Description

Use the instruction "Save RLO in BR bit" to save the result of logic operation (RLO) in the binary result (BR). During processing, the instruction transfers the signal state of the current RLO to the status bit BR. The instruction operates independent of the conditions and does not affect any other status bits.

Note

After application of the instruction "Save RLO in BR bit", the BR bit can be modified again by the following instructions in the same or lower-level blocks.

Use the instruction "Save RLO in BR bit" to check the execution status of a block. If you program the instruction "Save RLO in BR bit" at the end of a block, for example, the BR bit is set to the signal state of the current result of logic operation of the block. If the block is executed without errors, the BR bit and the enable output ENO in the calling block are set to signal state "1". Signal state "0" is assigned to the BR bit if an error occurs during processing of the block. You can program a troubleshooting routine by querying the BR bit.

Example

The following example shows how the instruction works:

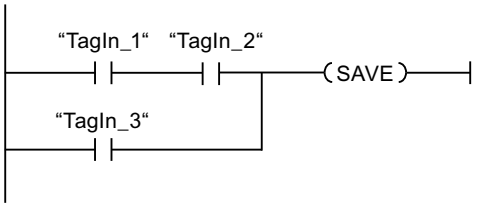

The BR bit is set to the signal state "1" if one of the following conditions is met:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".

See also

Basic information on the status word (Page 173)

Example of the use of the EN/ENO mechanism in LAD (Page 146)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Setting status bits (Page 175)

---(MCR<): Open MCR ranges

Description

The "Open MCR ranges" instruction saves the result of logic operation (RLO) in the MCR stack and enables MCR dependency within an MCR range (MCR = Master Control Relay).

The MCR stack can contain up to a maximum of 8 entries and functions like a LIFO buffer (Last In, First Out). When the stack is full, the instruction generates an error message (MCRF).

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Use the instructions "Enable MCR range" (MCRA) and "Disable MCR range" (MCRD) to specify the start and end of an MCR range. Within the defined MCR range you can enable and disable MCR dependency for one or more program sections.

MCR dependency is enabled when the RLO is "1" before processing of the instruction. If the RLO is "0" before the instruction, MCR dependency is disabled. The end of the MCR dependent program section is programmed using the instruction "Close MCR ranges". The program section between the two instructions is affected by the MCR. Only the following MCR-dependent instructions are affected by this:

- Assignment
- Set output (S)
- Reset output (R)
- Reset/set flip-flop (RS)
- Set/reset flip-flop (SR)
- Move value (MOVE)

You can also nest the enabling and disabling of MCR dependency by programming the instruction "Open MCR ranges" in an MCR dependent program section. You can nest the instruction up to 8 levels deep. The MCR dependency of the higher-level program section hereby controls the MCR dependencies in all lower-level program sections. If MCR dependency is disabled in the higher-level program section, this MCR dependency cannot be switched on in any of the lower-level MCR dependent program sections.

If you call a block within an MCR range, MCR dependency is disabled in the called block. When the block is exited, MCR dependency is set so that it is the same as before the block call.

The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Example

The following example shows how the instruction works:

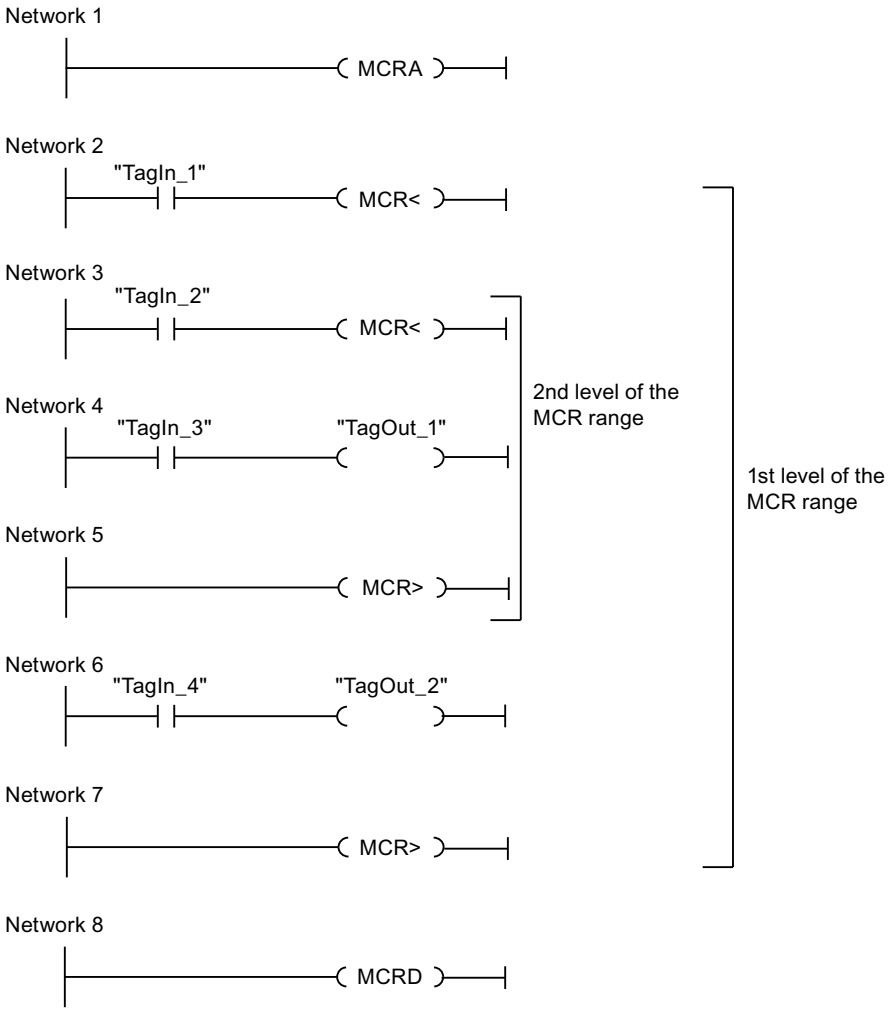

The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns signal state "1", MCR dependency is enabled in the first level of the MCR range. In this case, the signal state of the "TagIn_4" operand is assigned to the "TagOut_2" operand (network 6). If the "TagIn_1" operand returns signal state "0", MCR dependency is disabled and the "TagOut_2" operand is reset to "0". MCR dependency of the MCR range is disabled in network 7.

In network 3, MCR dependency is enabled in the second level of the MCR range, when the "TagIn_2" operand has signal state "1" and the first level of the MCR range is enabled. In this case, the signal state of the "TagIn_3" operand is assigned to the "TagOut_1" operand. If the "TagIn_2" operand returns signal state "0", MCR dependency is disabled and the "TagOut_1" operand is reset to "0". MCR dependency in the second level of the MCR range is disabled in network 5. The end of the MCR range is programmed in network 8.

See also

Important notes on using MCR functions (Page 192)

Master control relay (Page 191)

---(MCR>): Close MCR ranges (Page 4341)

---(MCRA): Enable MCR range (Page 4343)

---(MCRD): Disable MCR range (Page 4344)

---(MCR>): Close MCR ranges

Description

The instruction "Close MCR ranges" completes an MCR dependent (MCR = Master Control Relay) program section. In this case the instruction deletes the result of logic operation (RLO) from the last entry in the MCR stack.

The MCR stack can contain up to a maximum of 8 entries and functions like a LIFO buffer (Last In, First Out). If the stack is already empty, the instruction generates an error message (MCRF).

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Use the instruction "Open MCR ranges" to program MCR dependency. The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Example

The following example shows how the instruction works:

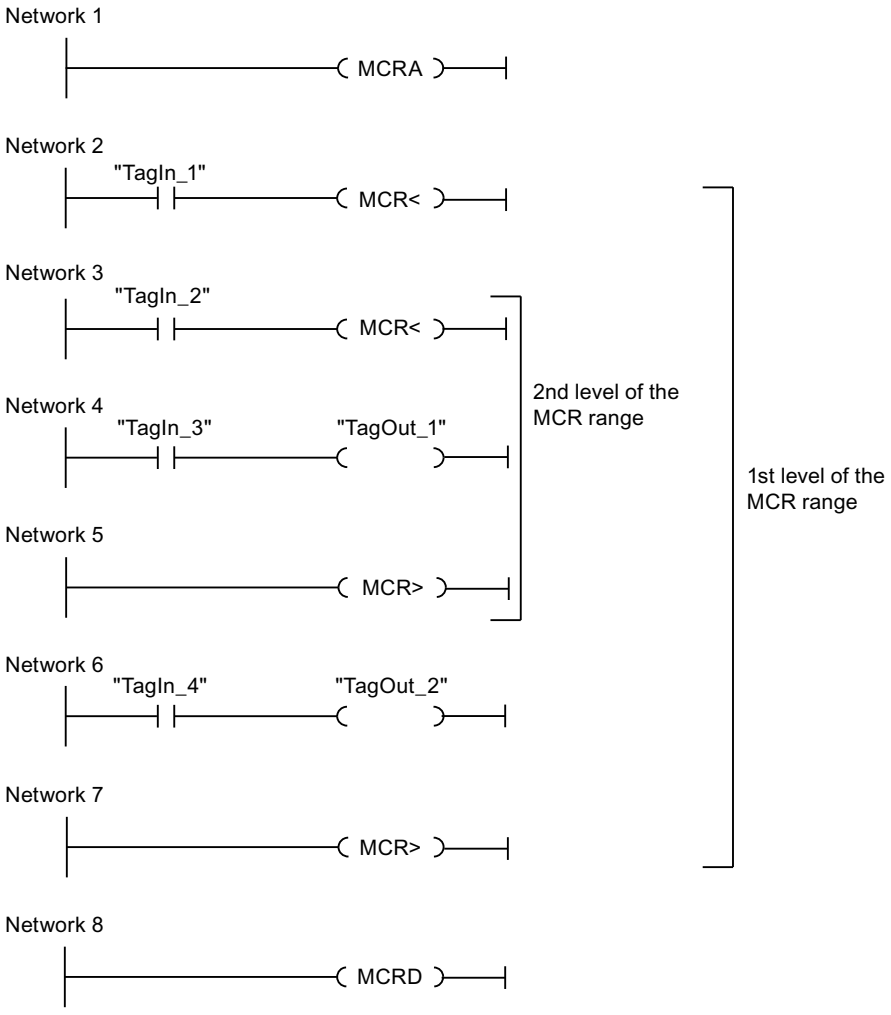

The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns signal state "1", MCR dependency is enabled in the first level of the MCR range. In this case, the signal state of the "TagIn_4" operand is assigned to the "TagOut_2" operand (network 6). If the "TagIn_1" operand returns signal state "0", MCR dependency is disabled and the "TagOut_2" operand is reset to "0". MCR dependency of the MCR range is disabled in network 7.

In network 3, the MCR-dependency is enabled in the second level of the MCR range, when the "TagIn_2" operand has signal state "1". In this case, the signal state of the "TagIn_3" operand is assigned to the "TagOut_1" operand. If the "TagIn_2" operand returns signal state "0", the MCR dependency is disabled and the "TagOut_1" operand is reset to "0" regardless of the signal state of the "TagIn_3" operand. MCR dependency in the second level of the MCR range is disabled in network 5. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

---(MCR<): Open MCR ranges (Page 4339)

---(MCRA): Enable MCR range (Page 4343)

---(MCRD): Disable MCR range (Page 4344)

---(MCRA): Enable MCR range

Description

Use the instruction "Enable MCR range" to specify the start of an MCR range. Within this range, you can switch MCR dependency on and off using the following instructions:

- ---(MCR<): Open MCR ranges
- ---(MCR>): Close MCR ranges

You specify the end of the MCR range with the instruction "Disable MCR range" (MCRD). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

If you call a block within an MCR range, MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Example

The following example shows how the instruction works:

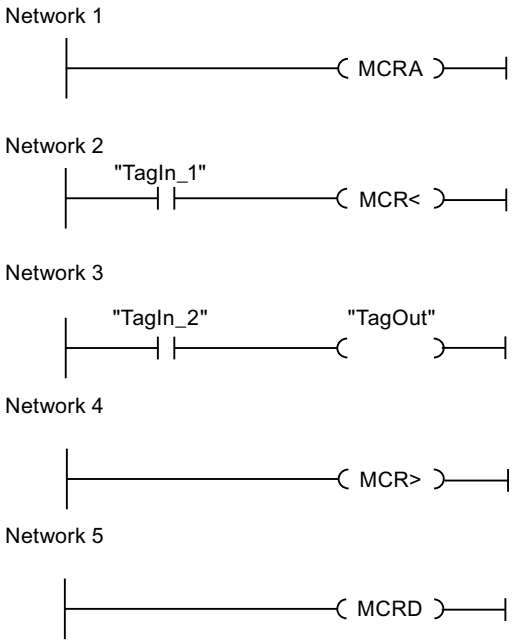

The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns the signal state "1", the MCR dependency is enabled. In this case, the signal state of the "TagIn_2" operand is assigned to the "TagOut" operand (network 3). If the "TagIn_1" operand returns signal state "0", the MCR dependency is disabled and the "TagOut" operand is reset to "0" regardless of the signal state of the "TagIn_2" operand. MCR dependency of the MCR range is disabled in network 4. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

---(MCR<): Open MCR ranges (Page 4339)

---(MCR>): Close MCR ranges (Page 4341)

---(MCRD): Disable MCR range (Page 4344)

---(MCRD): Disable MCR range

Description

Use the instruction "Disable MCR range" to specify the end of an MCR range.

You specify the start of the MCR range with the instruction "Enable MCR range" (MCRA). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Within an MCR range you can program the MCR dependency with the help of the following instructions:

- ---(MCR<): Open MCR ranges
- ---(MCR>): Close MCR ranges

If you call a block within an MCR range, MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Example

The following example shows how the instruction works:

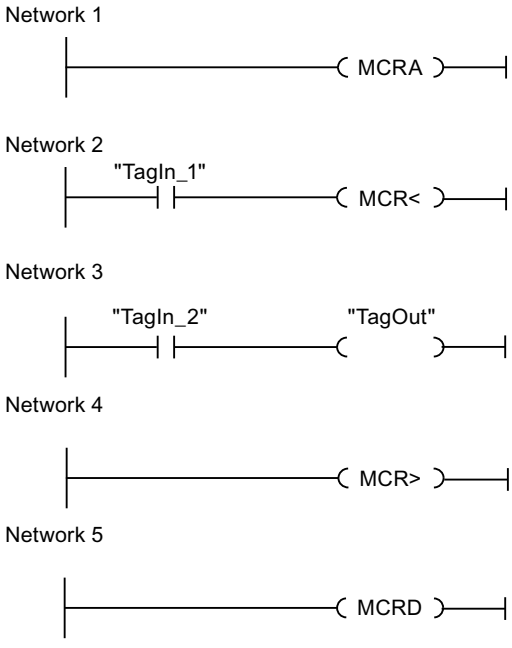

The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns the signal state "1", the MCR dependency is enabled. In this case, the signal state of the "TagIn_2" operand is assigned to the "TagOut" operand (network 3). If the "TagIn_1" operand returns signal state "0", the MCR dependency is disabled and the "TagOut" operand is reset to "0" regardless of the signal state of the "TagIn_2" operand. MCR dependency of the MCR range is disabled in network 4. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

---(MCR<): Open MCR ranges (Page 4339)

---(MCR>): Close MCR ranges (Page 4341)

---(MCRA): Enable MCR range (Page 4343)

SET: Set bit array

Description

The "Set bit array" instruction sets the signal state of the bit in a specified area to "1". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. At the N parameter, you specify the number of bits to be set in the specified area. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Note

Parameter N

You can also interconnect an element of a data block at parameter N.

Parameters

The following table shows the parameters of the "Set bit array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P, or constant	Number of bits to be set

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

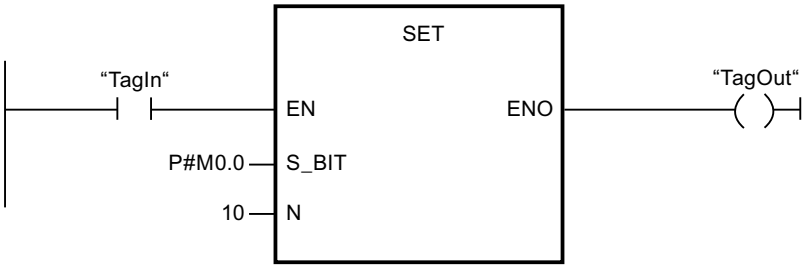

If the TagIn operand has the signal state "1", the instruction sets 10 bits in the area from M0.0 to M1.2.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

SETP: Set a bit array in the I/O area (S7-400)

Description

The instruction "Set a bit array in the I/O area" sets the bits of an I/O area to the signal state "1". At the same time the corresponding bits in the process image of the outputs are also set parallel to the I/O bits.

You use the N and SA parameters of the instruction to define the bit array that will be set. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output on the RET_VAL parameter.

The I/O area is set byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first or last byte to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

  WARNING

  Unintended behavior

  The setting of the bits to the value of the associated bits in the process image output can lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are set to the signal state "1".

The instruction "Set a bit array in the I/O area" must be called in an MCR area.

Calling the "Set a bit array in the I/O area" instruction has no effect and the enable output ENO is set to "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The N parameter has the value "0"
- The MCR bit has the signal state "1".

Parameters

The following table shows the parameters of the "Set a bit array in the I/O area" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be set
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be set

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

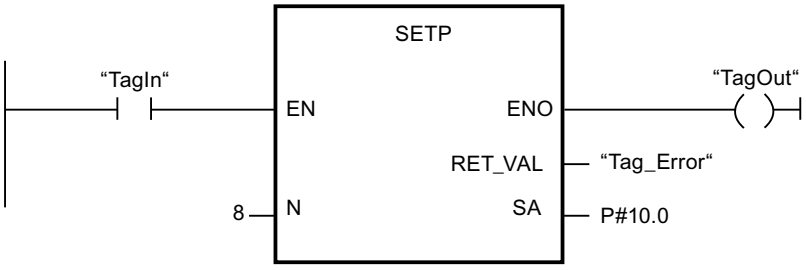

If the operand "TagIn" has the signal state "1", the instruction sets 8 bits in the I/O range corresponding to the outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

SETI: Set byte array

Description

The "Set byte array" instruction sets the signal state of the bit in a specified area of bytes to "1". You define the start point of the array on the S_BYTE. You define the size of the array by specifying the number of bits to be set on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "I/0" memory area is accessed as bytes, words, or double words, the pointer on the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output is reset to "0".

Calling the "Set byte array" instruction has no effect and the ENO enable output is set to "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The pointer on the S_BYTE parameter does not reference the external I/O memory area.
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Parameters

The following table shows the parameters of the "Set byte array" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BYTE	Input	POINTER	P or constant	Pointer to the first byte of the array whose bits are to be set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array to be set The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

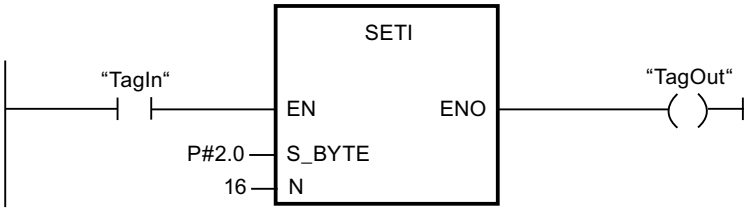

If the operand "TagIn" has the signal state "1", the instruction sets 2 bytes (16 bits) in the range from P2.0 to P3.7.

See also

Overview of the valid data types (Page 231)
Basics of the EN/ENO mechanism (Page 141)
Master control relay (Page 191)
Important notes on using MCR functions (Page 192)
Inserting LAD elements (Page 7134)
Editing LAD elements (Page 7152)
Inserting operands into LAD instructions (Page 7159)

RESET: Reset bit array

Description

You use"Reset bit array" instruction to reset the signal state of the bits in a specified area to "0". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. You specify the number of bits to be reset in the specified range at the N parameter of the instruction. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Parameters

The following table shows the parameters of the "Reset bit array" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are to be reset For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be reset.

Example

The following example shows how the instruction works:

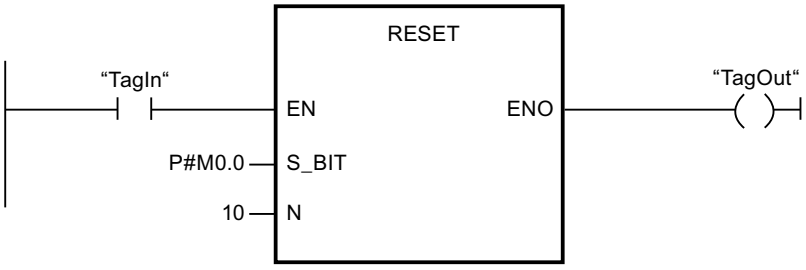

If the operand "TagIn" has the signal state "1", the instruction resets 10 bits in the range from M0.0 to M1.2.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

RESETP: Reset a bit array in the I/O area (S7-400)

Description

The instruction "Reset a bit array in the I/O area" sets the bits of an I/O area to the signal state "0". At the same time the corresponding bits in the process image of the outputs are also reset parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be reset. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output on the RET_VAL parameter.

The I/O area is reset byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output table.

  WARNING

  Unintended behavior

  The setting of the bits at the value of the relevant bit in the process image output table can lead, for example, to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are reset to the signal state "0".

The "Reset a bit array in the I/O area" instruction must be called in an MCR range.

Calling the "Reset a bit array in the I/O area" instruction has no effect and the enable output ENO is set to "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The N parameter has the value "0"
- The MCR bit has the signal state "1".

Parameters

The following table shows the parameters of the "Reset a bit array in the I/O area" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be reset.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

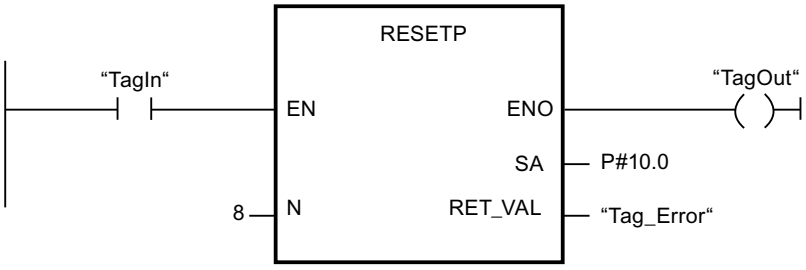

If the operand "TagIn" has the signal state "1", the instruction sets 8 bits in the I/O range corresponding to the outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

RESETI: Reset byte array

Description

The "Reset byte array" instruction is used to reset the signal state of the bits in a specified range of bytes to "0". You use the pointer on the S_BYTE parameter to define the starting point of the range. You define the size of the range by specifying the number of bits to be reset on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area. Since the "I/0" memory area is accessed as bytes, words, or double words, the pointer on the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output is reset to "0".

Calling the "Reset byte array" instruction has no effect and the ENO enable output is set to "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The pointer on the S_BYTE parameter does not reference the external I/O memory area.
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Parameters

The following table shows the parameters of the "Reset byte array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BYTE	Input	POINTER	P or constant	Pointer to the first byte of the range whose bits are to be reset. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array whose bits are to be reset. The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

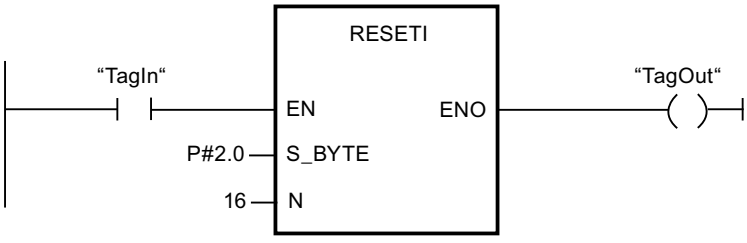

If the operand "TagIn" has the signal state "1", the instruction resets 2 bytes (16 bits) in the range from P2.0 to P3.7.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

REPL_VAL: Enter substitute value

Description

The "Enter substitute value" instruction is used to enter a substitute value in the accumulator 1 (ACCU 1) from a synchronous error organization block.

Note

You can only call the "Enter substitute value" instruction in a synchronous error organization block (OB 121, OB 122).

You use the "Enter substitute value" instruction when you can no longer read any value from a module. The OB 122 is started after each access to this module. By calling the "Enter substitute value" instruction you can cause the substitute value to be loaded in accumulator 1. In this case the program is executed with this substitute value. You can take the required information for the selection of the substitute value from the local tags of the OB 122.

Parameters

The following table shows the parameters of the "Enter substitute value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
VAL	Input	DWORD	I, Q, M, D, L or constant	Substitute value
RET_VAL	Output	INT	I, Q, M, D, L	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Description
0000	No error
8080	The instruction was not called from a synchronous error OB.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on LAD (Page 7121)

DRUM: Implement sequencer

Description

The "Implement sequencer" instruction is used to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word OUT_WORD. The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK), you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask has signal state "1", the OUT_VAL value sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit at the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit at the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" indicates the reset condition.
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L	The signal state "1" causes the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.

Parameter	Declaration	Data type	Memory area	Description
DSP	Static	BYTE	I, Q, M, D, L, P	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P	Current numerical value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L	Preset count for each step [1 to 16]; where 1 clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

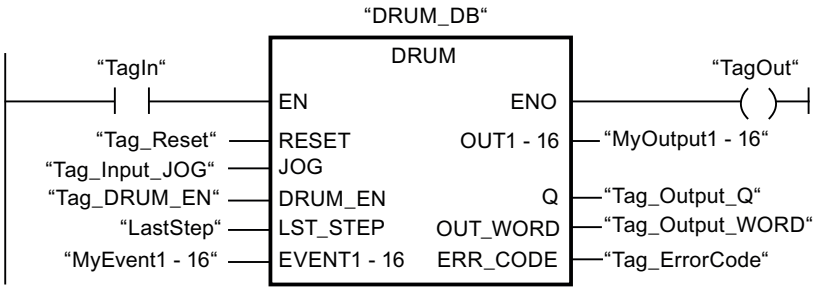

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameter	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE

Parameter	Address	Value
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE

Parameter	Operand	Address	Value
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

DRUM_X: Implement sequencer

Description

The "Implement sequencer" instruction is used to implement a sequencer with maximum 16 steps. You specify the number of the first step on the DSP parameter. You specify the number of the last step on the LST_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT_WORD (in which the output bits are collected together) are written. An output bit is assigned either the corresponding bit of the OUT_VAL array that you specified or the output bit of the preceding step. The value assigned depends on the value of the mask bit on the S_MASK parameter.

Note

The default setting for the mask bit is 0. If you want to change the current setting of one or more mask bits, make this change in the instance data block.

The "Implement sequencer" instruction advances to the next step when there is a rising edge on the JOG parameter. If the instruction is already in the last step, the tags Q and EOD are set when the JOG parameter has a rising edge. In this case the DCC parameter is set to the value "0". The instruction remains in the last step until the RESET parameter is set to the value "1".

You can also assign parameters so that switching to the next step is time dependent. To do this, the DRUM_EN parameter has to be set to the value "1". The transition to the next step is

made when the event bit (EVENTi) belonging to the current step is set and the time specified for the current step has expired. The time for a step is the product of the (DTBP) time base and the time factor valid for the current step (S_PRESET).

Note

The execution time (DCC) remaining in the current step is reduced only when the corresponding event bit (EVENTi) is set.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

If the value "1" is set on the RESET parameter when the instruction is called, the sequencer goes to the step whose number you assigned on the DSP parameter.

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1".

You can also move the next step at any time via the value at the JOG parameter.

When the instruction is called for the first time, you must set the value "1" on the RESET parameter.

When the sequencer is in the last step and the execution time specified for this step has expired, the parameters Q and EOD are set. The instruction remains in the last step until the RESET parameter changes to the value "1".

The instruction is executed only in the STARTUP and RUN modes.

The operating system resets the "Implement sequencer" instruction during a cold restart. During a warm restart, the instruction is not reset. If you want the "Implement sequencer" instruction to be initialized after a warm restart, you have to call the OB 100 with the value "1" on the RESET parameter.

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" resets the sequencer. When the instruction is called for the first time, you have to set the value "1" on the RESET parameter.

Parameter	Declaration	Data type	Memory area	Description
JOG	Input	BOOL	I, Q, M, D, L	A positive signal edge (compared to the last call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you assigned to the DRUM_EN parameter.
DRUM_EN	Input	BOOL	I, Q, M, D, L	Control parameter that specifies whether time-dependent switching to the next step is possible. If the value is "1" a time-dependent switching to the next step is possible.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last step Values from 1 to 16 are permitted.
EVENTi, 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit number i (belongs to step i)
OUTj, 0 ≤ j ≤ 15	Output	BOOL	I, Q, M, D, L	Output bit number j (identical to the bit number j j of OUT_WORD)
Q	Output	BOOL	I, Q, M, D, L	Status parameter that indicates whether the specified execution time of the last step has expired.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Output bits collected together in a tag
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	Not relevant to user (input parameter JOG of the preceding call)
EOD	Static	BOOL	I, Q, M, D, L	Identical to the output parameter Q
DSP	Static	BYTE	I, Q, M, D, L, P, or constant	Number of the first step Values from 1 to 16 are permitted.
DSC	Static	BYTE	I, Q, M, D, L, P, or constant	Number of the current step
DCC	Static	DWORD	I, Q, M, D, L, P	The execution time still remaining in the current step in milliseconds (ms) (only relevant if DRUM_EN = 1 and the corresponding event bit = 1)
DTBP	Static	WORD	I, Q, M, D, L, P, or constant	The time base valid for all steps in milliseconds (ms)
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	(Not relevant for the user: system time of the previous call)

Parameter	Declaration	Data type	Memory area	Description
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	One-dimensional array with the time factor for each step. Recommended selection of the indices: [1 to 16]. In this case, S_PRESET [x] has the time factor of step x.
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the values to be output in each step if they have not been masked out using S_MASK. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, OUT_VAL [x, y] has the value assigned to the output bit OUTy in step x.
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the mask bits for each step. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, S_MASK [x, y] contains the mask bit for the "y"-th value output in step x. Meaning of the mask bits: • 0: The corresponding value of the previous step is assigned to the relevant output bit. • 1: The corresponding value from OUT_VAL is assigned to the relevant output bit.

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE* (W#16#...)	Explanation
0000	No error
8081	Invalid value at parameter LST_STEP.
8082	Invalid value on parameter DSC.
8083	Invalid value on parameter DSP.

ERR_CODE* (W#16#...)	Explanation
8084	The product DCC = DTBP * S_PRESET[DSC] exceeds the value 2**31-1 (approximately 24.86 days).
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT0 to OUT15) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

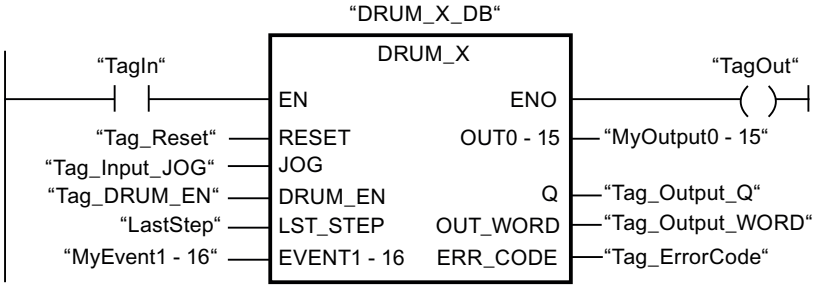

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameter	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_X_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE

Parameter	Address	Value
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT0	MyTag_Output_0	M4.0	TRUE
OUT1	MyTag_Output_1	M4.1	TRUE
OUT2	MyTag_Output_2	M4.2	TRUE
OUT3	MyTag_Output_3	M4.3	TRUE
OUT4	MyTag_Output_4	M4.4	TRUE
OUT5	MyTag_Output_5	M4.5	TRUE
OUT6	MyTag_Output_6	M4.6	TRUE
OUT7	MyTag_Output_7	M4.7	TRUE
OUT8	MyTag_Output_8	M5.0	TRUE
OUT9	MyTag_Output_9	M5.1	TRUE
OUT10	MyTag_Output_10	M5.2	TRUE
OUT11	MyTag_Output_11	M5.3	TRUE
OUT12	MyTag_Output_12	M5.4	TRUE
OUT13	MyTag_Output_13	M5.5	TRUE
OUT14	MyTag_Output_14	M5.6	TRUE
OUT15	MyTag_Output_15	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT0	MyTag_Output_0	M4.0	TRUE
OUT1	MyTag_Output_1	M4.1	FALSE
OUT2	MyTag_Output_2	M4.2	FALSE
OUT3	MyTag_Output_3	M4.3	FALSE
OUT4	MyTag_Output_4	M4.4	FALSE
OUT5	MyTag_Output_5	M4.5	TRUE
OUT6	MyTag_Output_6	M4.6	FALSE
OUT7	MyTag_Output_7	M4.7	FALSE
OUT8	MyTag_Output_8	M5.0	TRUE
OUT9	MyTag_Output_9	M5.1	TRUE
OUT10	MyTag_Output_10	M5.2	FALSE
OUT11	MyTag_Output_11	M5.3	FALSE
OUT12	MyTag_Output_12	M5.4	FALSE
OUT13	MyTag_Output_13	M5.5	FALSE
OUT14	MyTag_Output_14	M5.6	TRUE
OUT15	MyTag_Output_15	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are saved in the "DRUM_X_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

DCAT: Discrete control-timer alarm

Description

You can use the "Discrete control-timer alarm" instruction to accumulate the time from the point at which theCMD parameter issued the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state on the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET, OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET, OA, CA and CMD_HIS are reset to "0".
- When the signal state of the parameters CMD and CMD_HIS is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state on the parameter OA is set to "1". If the value of the parameter ET does not exceed the value of the parameter PT, the signal state on the parameter OA is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state of the parameter CA is set to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the parameter CMD_HIS is set to the value of the parameter CMD.

- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the parameter C_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CMD	Input	BOOL	I, Q, M, D, L	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
CMD_HIS	Static	BOOL	I, Q, M, D, L	CMD history bit

You can find additional information on valid data types under "See also".

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.

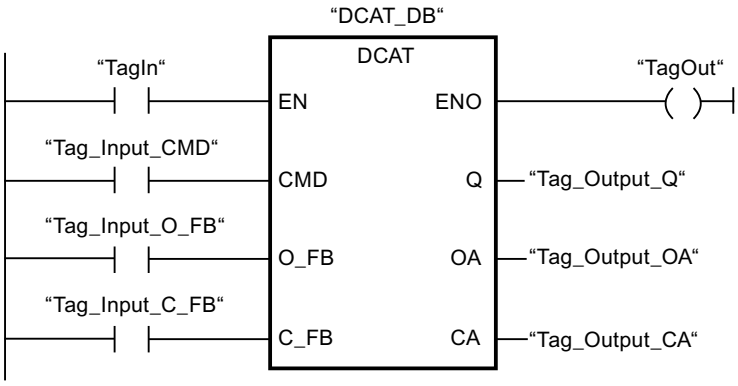

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

MCAT: Motor control-timer alarm

Description

The instruction "Motor control-timer alarm" is used to accumulate the time from the point of time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the instruction "Motor control-timer alarm" to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped
Legend:																
INC	Add the time difference (ms) since the last processing of the FB to ET															
PT	PT is set to the same value as ET															

Input parameters	Output parameters
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT
If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case: OO = FALSE CO = FALSE OA = FALSE CA = FALSE ET = PT Q = TRUE	

The "Motor control-timer alarm" instruction returns no error information.

Parameters

The following table shows the reactions of the instruction "Motor control-timer alarm":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when open-ing
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition.
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
O_HIS	Static	BOOL	I, Q, M, D, L	"Open" history bit
C_HIS	Static	BOOL	I, Q, M, D, L	"Close" history bit

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

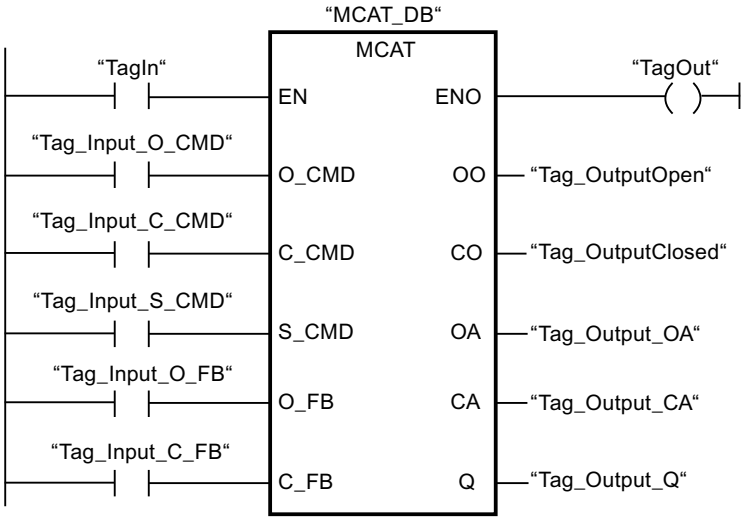

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

IMC: Compare input bits with the bits of a mask

Description

The instruction "Compare input bits with the bits of a mask" is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. At the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate

data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 to be compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 to be compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 to be compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 to be compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 to be compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 to be compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 to be compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 to be compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 to be compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 to be compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 to be compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 to be compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 to be compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 to be compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 to be compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 to be compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.

Parameter	Declaration	Data type	Memory area	Description
OUT	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that a match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

You can find additional information on valid data types under "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

SMC: Compare scan matrix

Description

The "Compare scan matrix" instruction is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE. If several steps have a corresponding mask, only the the first match is specified at the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 to be compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 to be compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 to be compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 to be compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 to be compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 to be compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 to be compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 to be compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 to be compared with bit 8 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 to be compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 to be compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 to be compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 to be compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 to be compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 to be compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 to be compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY of WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

You can find additional information on valid data types under "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

LEAD_LAG: Lead and lag algorithm

Description

The "Lead and lag algorithm" instruction is used to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the instruction "Lead and lag algorithm" is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two operations together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed. Constants can also be specified on the IN parameter.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time Constants can also be specified on the SAMPLE_T parameter.
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in in the same unit as sample time.
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

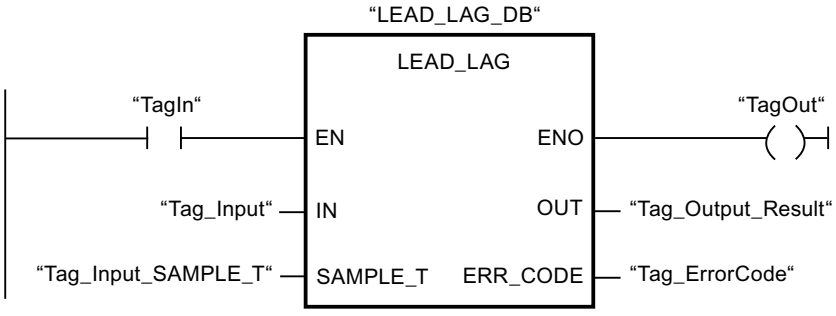

The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

TONR_X: Time accumulator

Description

You use the "Time accumulator" instruction to accumulate time values within a period which you set using the PV input. The instruction is executed when the signal state on the TMR_EN start input changes from "0" to "1". Simultaneously, the timer runs for the duration set at the PV input. While the instruction is being executed, it adds the time value of the DELTA_T input to the time value of the InOut ET.

The following conditions must be met:

- The TMR_EN start input has the signal state "1".
- The time value at InOut ET is less than the time value at the PV parameter.
- The RESET input has the signal state "0".

The instruction is interrupted when the signal state on the TMR_EN start input changes from "1" to "0". As soon as the signal state changes back to "1", the instruction resumes execution.

The instruction is ended when the time value at InOut ET has reached the time value at the PV input. The Q output then has the signal state "1".

As long as the Q output has the signal state "1", the time value at InOut ET remains unchanged. The time value and the Q output are not reset to "0" until the RESET input is set to signal state "1".

Since the "Time accumulator" instruction uses the cycle time (DELTA_T) of the last cycle of the organization block (OB), the instruction may only be called in cyclic organization blocks.

Note

You have to move the cycle time of the organization block from the startup information to the global tag of the DELTA_T parameters.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L, T, C	Enable output
TMR_EN (Timer Enable)	Input	BOOL	I, Q, M, D, L, T, C	Start input Starts the instruction and the expiration of the preset time duration at the PV input
RESET	Input	BOOL	I, Q, M, D, L, T, C	Reset input
PV (Preset Value)	Input	DINT	I, Q, M, D, L, P, or constant	Maximum duration of time recording
DELTA_T	Input	INT	I, Q, M, D, L or constant	Cycle time of organization block of the previous cycle
Q	Output	BOOL	I, Q, M, D, L	Output which is set when the preset time value is reached
ET (Elapsed Time)	InOut	DINT	I, Q, M, D, L	Accumulated time value (Currently elapsed time value + time value of the DELTA_T input)

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:

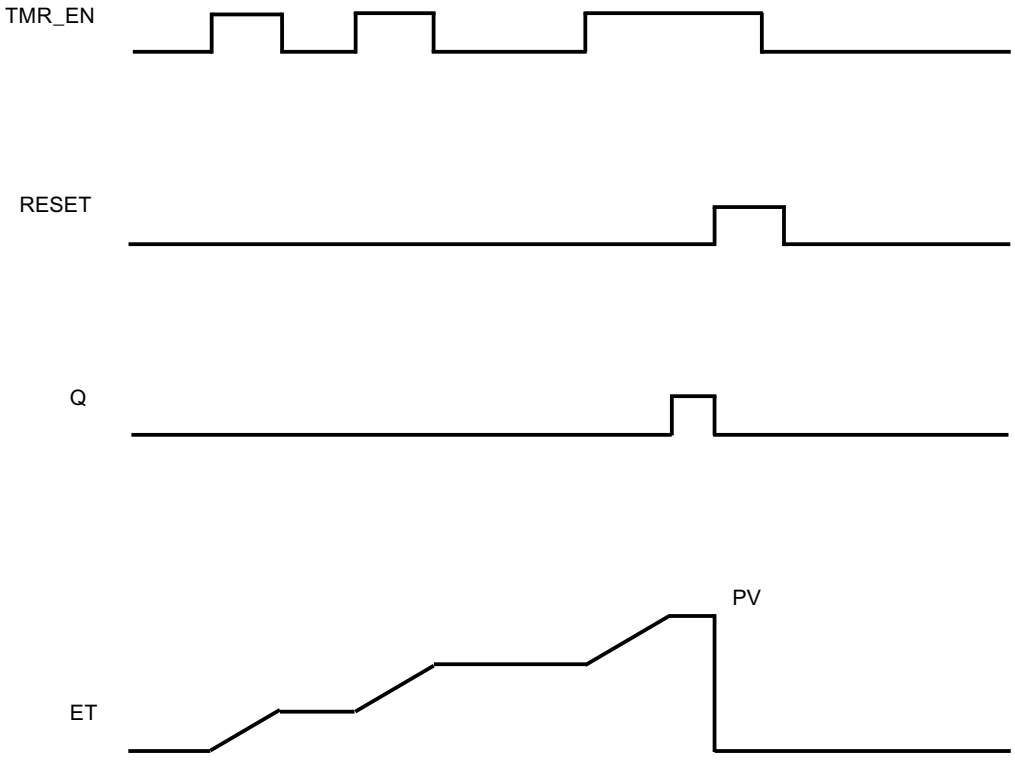

Example

The following example shows how the instruction works:

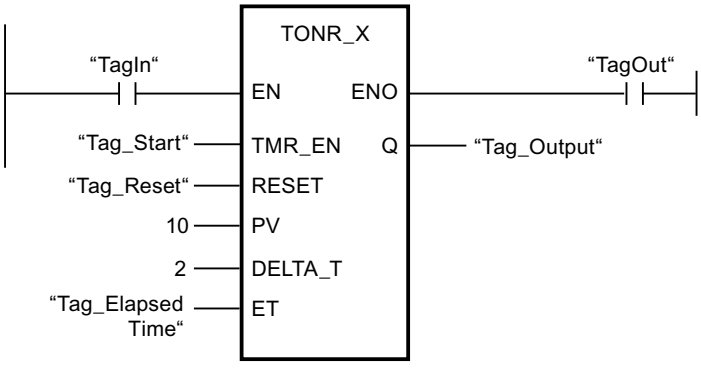

If the operands "TagIn" and "Tag_Start" have the signal state "1", the instruction is started. Each time the signal state changes from "0" to "1" at the TMR_EN input, the cycle time "2" at the DELTA_T input is added to the current time value of InOut ET until the maximum time value "10" of the PV input is reached. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

WSR: Save data to shift register

Description

The "Save data to shift register" instruction is used to write a bit from a specified source to a shift register. Each element of the register is shifted into the next address. The data contained in the last address of the shift register is lost after the shift.

Each time the instruction is executed with the value "0" on the RESET parameter, new data are read from the source (S_DATA) and shifted to the start address (START) of the shift register. The LENGTH parameter specifies the number of elements to be shifted. You specify the data type of the elements on the E_TYPE parameter. The instruction is not executed if an invalid data type is specified on the E_TYPE parameter.

The register is deleted if the RESET parameter is set to "1" when the instruction is executed. The Q parameter is set to the signal state "1" when the shift register is filled with zeros or deleted.

Parameters

The following table shows the parameters of the "Save data to shift register" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_DATA	Input	POINTER	I, Q, M, D	Points to the source data element to be shifted to the start address (START). For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
START	Input	POINTER	I, Q, M, D	Points to the start address of the register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
LEN	Input	WORD	I, Q, M, D, L, P	Number of elements to be shifted.

Parameter	Declaration	Data type	Memory area	Description
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the elements to be shifted. The following data types are permitted: • W#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
Q	Output	BOOL	Q, M, D, L	Is set to "0" if RESET parameter is active (1) or all elements to be shifted have the value "0".

Example

In the following example four elements of the data type WORD are shifted starting from the address specified on the START parameter. When the first element of the register has been shifted to the next address, the first address is filled with the data that the pointer references on the S_DATA parameter. The value of the last element in the register is lost in the process.

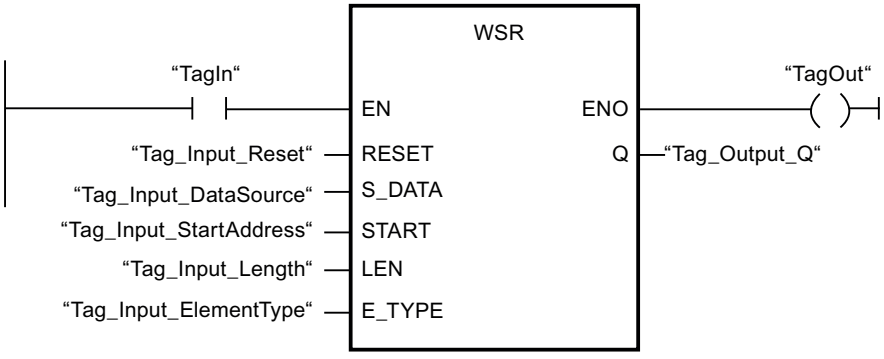

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1111 DW2 = W#16#2222 DW4 = W#16#3333 DW6 = W#16#4444
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04

Parameter	Operand	Value
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1234 DW2 = W#16#1111 DW4 = W#16#2222 DW6 = W#16#3333
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

SHRB: Shift bit to shift register

Description

The "Shift bit to shift register" instruction is used to write a bit from a specified source (DATA) to a shift register. Each time the instruction is executed with a value "0" on the RESET parameter, new data are read from the source and shifted to the start address (S_BIT) of the shift register. All following bits are shifted by one. The bit contained at the last address (S_BIT + N) is lost after the shift. If the RESET parameter is set to "1" during the processing of the instruction, the addresses in the shift register are reset to "0" and not shifted.

Parameters

The following table shows the parameters of the "Shift bit to shift register" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DATA	Input	BOOL	I, Q, M, D, L	Source bit
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_BIT	Input	POINTER	I, Q, M, D	Points to the start bit of the shift register For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	WORD	I, Q, M, D, L, P	Number of bits to be shifted.

Example

In the following example the N parameter is set to the value 14 ("E" in hexadecimal). As a result 14 bits starting from the address specified on the S_BIT parameter are shifted. After the bits have been shifted, the first address is filled with the data from the source on the DATA parameter. The last bit value is lost.

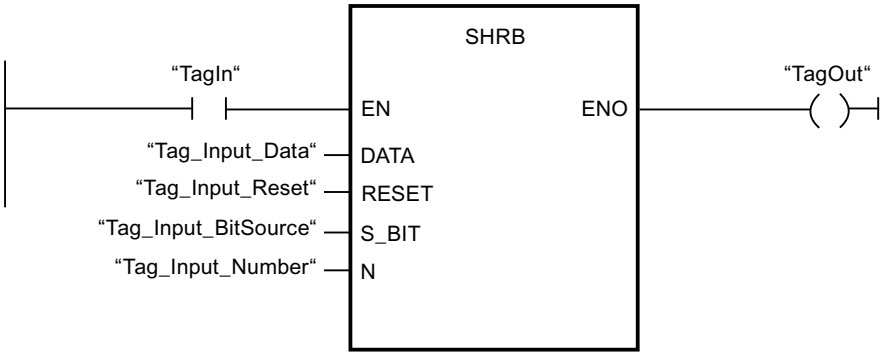

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
DATA	Tag_Input_Data	0

Parameter	Operand	Value
S_BIT	Tag_Input_BitSource	P#DB1.DBX1.4
N	Tag_Input_Number	B#16#0E

Before processing the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	0	1	1	1	0	1	0

2.7	...					...	2.0
0	0	1	1	1	0	1	0

3.7	...					...	3.0
0	0	1	1	1	0	1	0

After processing

After the processing of the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	1	1	0	1	0	1	0

2.7	...					...	2.0
0	1	1	1	0	1	0	0

3.7	...					...	3.0
0	0	1	1	1	0	0	0

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

SEG: Create bit pattern for seven-segment display

Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)
0000	00111111	0
0001	00000110	1
0010	01011011	2
0011	01001111	3
0100	01100110	4
0101	01101101	5
0110	01111101	6
0111	00000111	7
1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	B
1100	00111001	C
1101	01011110	D
1110	01111001	E
1111	01110001	F

Seven-segment display

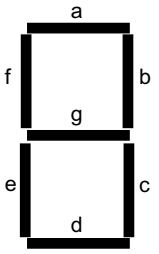

Parameter

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:

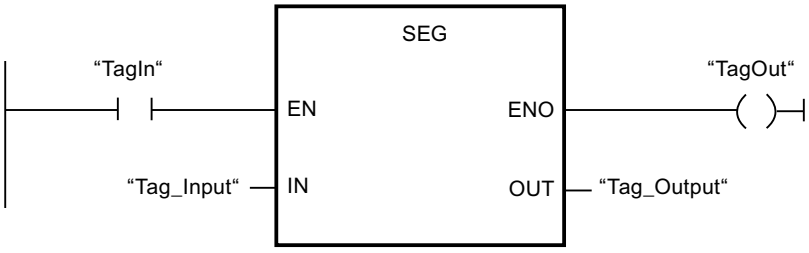

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 Display: 1234

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

BCDCPL: Create tens complement

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:

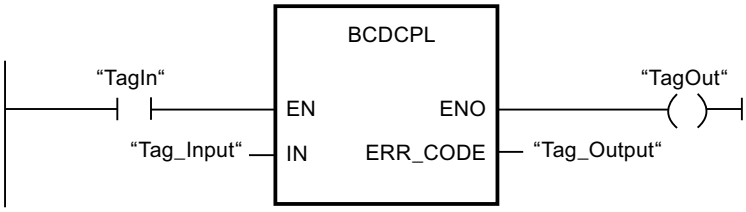

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output on the RET_VAL parameter.

Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

Example

The following example shows how the instruction works:

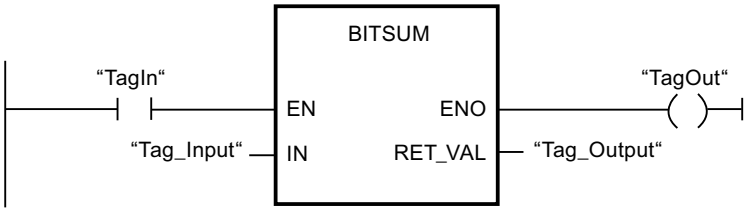

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on LAD (Page 7121)

4.2.2.2 FBD

Bit logic operations

&: AND logic operation

Description

You can use the instruction "AND logic operation" to query the signal states of two or more specified operands and evaluate them according to the AND truth table.

If the signal state of all the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of one of the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

Parameters

The following table shows the parameters of the instruction "AND logic operation":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:

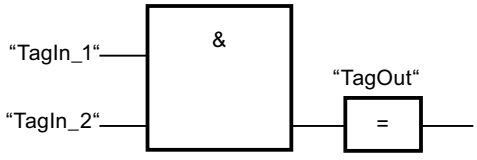

Output "TagOut" is set, when the signal state of the operands "TagIn_1" and "TagIn_2" is "1" and reset when the state of the operands "TagIn_1" and "TagIn_2" is "0".

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

AND truth table (Page 4398)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

AND truth table

Results of the logic operation

The following table shows the results that arise from the AND logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	1	1
0	1	0
1	0	0
0	0	0

See also

&: AND logic operation (Page 4397)

>=1: OR logic operation

Description

You can use the instruction "OR logic operation" to query the signal states of two or more specified operands and evaluate them according to the OR truth table.

If the signal state of one of the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of all the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

Parameters

The following table shows the parameters of the instruction "OR logic operation":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:

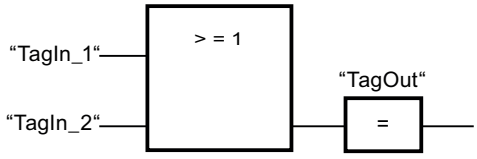

Output "TagOut" is set, when the signal state of the operands "TagIn_1" or "TagIn_2" is "1".

See also

Overview of the valid data types (Page 231)
Setting the status bit for bit logic operations (Page 175)
OR truth table (Page 4399)
FBD programming examples (Page 7232)
Basic information on FBD (Page 7185)

OR truth table

Results of the logic operation

The following table shows the results that arise from the OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	1
0	0	0

See also

>=1: OR logic operation (Page 4398)

X: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" instruction to query the result of a signal state query according to the EXCLUSIVE OR truth table.

With an "EXCLUSIVE OR logic operation" instruction, the signal state is "1" when the signal state of one of the two specified operands is "1". When more than two operands are queried, the common RLO is "1" if an odd number of the queried operands returns the result "1".

Parameters

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:

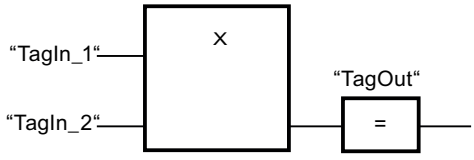

Output "TagOut" is set when the signal state of the operands "TagIn_1" and "TagIn_2" is "1". When both operands return the signal state "1" or "0", the output "TagOut" is reset.

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

EXCLUSIVE OR truth table (Page 4400)

Basic information on FBD (Page 7185)

EXCLUSIVE OR truth table

Results of the logic operation

The following table shows the results that arise from the EXCLUSIVE OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	0
0	0	0

The following table shows the results that arise from the EXCLUSIVE OR logic operation of three operands:

Signal state of the first operand	Signal state of the second operand	Signal state of the third operand	Result of the logic operation
1	0	0	1
0	1	1	0
0	1	0	1
1	0	1	0
0	0	1	1
1	1	0	0
1	1	1	1
0	0	0	0

See also

X: EXCLUSIVE OR logic operation (Page 4399)

Insert input

Description

The "Insert input" instruction is used to add an input to the box of one of the following instructions:

- "AND logic operation"
- "OR logic operation"
- "EXCLUSIVE OR logic operation"

You can query the signal state of several operands by the extension of an instruction box.

Parameters

The following table shows the parameters of the "Insert input" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:

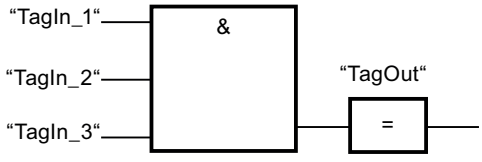

The box of the instruction was extended by an additional input at which the signal state of the "TagIn_3" operand is queried. The "TagOut" output is set, when the operands "TagIn_1" and "TagIn_2" and "TagIn_3" return signal state "1".

See also

Setting the status bit for bit logic operations (Page 175)

Basic information on the status word (Page 173)

&: AND logic operation (Page 4397)

>=1: OR logic operation (Page 4398)

X: EXCLUSIVE OR logic operation (Page 4399)

Basic information on FBD (Page 7185)

Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO).

Example

The following example shows how the instruction works:

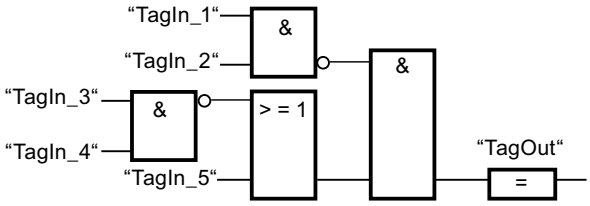

The "TagOut" output is set when the following conditions are fulfilled:

- The input "TagIn_1" and/or "TagIn_2" has signal state "0".
- The input "TagIn_3" and/or "TagIn_4" has signal state "0" or the input "TagIn_5" has signal state "1".

See also

Setting the status bit for bit logic operations (Page 175)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

=: Assignment

Description

You can use the instruction "Assignment" to set the bit of a specified operand. If the result of logic operation (RLO) at the box input has the signal state "1", the specified operand is set to signal state "1". If the signal state at the box input is "0", the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the box input is assigned directly to the operand above the assignment box.

Parameters

The following table shows the parameters of the instruction "Assignment":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:

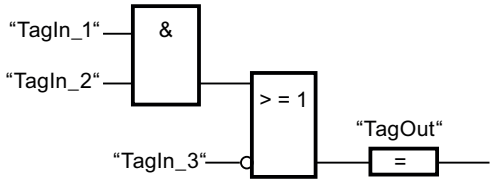

The operand "TagOut" is set at the output of the "Assignment" instruction when one of the following conditions is fulfilled:

- The inputs "TagIn_1" and "TagIn_2" have the signal state "1".
- The signal state at the input "TagIn_3" is "0".

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

R: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is reset to "0". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

Parameters

The following table shows the parameters of the "Reset output" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, T, C, D, L	Operand that is reset if RLO = "1".

Example

The following example shows how the instruction works:

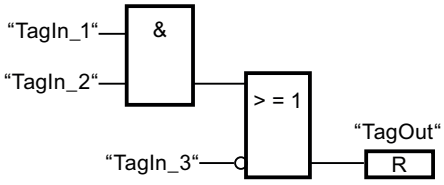

Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

S: Set output

Description

You can use the instruction "Set output" to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is set to "1". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

Parameters

The following table shows the parameters of the instruction "Set output":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:

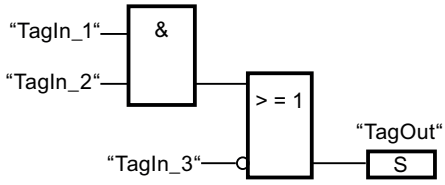

The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

SR: Set/reset flip-flop

Description

You can use the "Set/reset flip-flop" instruction to set or reset the bit of a specified operand based on the signal state of the inputs S and R1. If the signal state is "1" at input S and "0" at input R1, the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Parameters

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameters	Declaration	Data type	Memory area	Description
S	Input	BOOL	I, Q, M, D, L, T, C	Enable setting
R1	Input	BOOL	I, Q, M, D, L, T, C	Enable resetting
<Operand>	InOut	BOOL	I, Q, M, D, L	Operand that is set or reset
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:

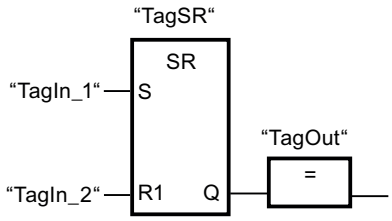

The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

Basic information on FBD (Page 7185)

RS: Reset/set flip-flop

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Parameters

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameters	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, T, C, D, L	Enable resetting
S1	Input	BOOL	I, Q, M, T, C, D, L	Enable setting
<Operand>	InOut	BOOL	I, Q, M, D, L	Operand that is reset or set
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:

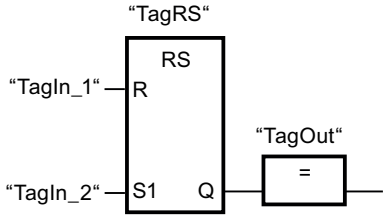

The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

Basic information on FBD (Page 7185)

P: Scan operand for positive signal edge

Description

You can use the "Scan operand for positive signal edge" instruction to determine whether there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

The following figure shows the change in the signal state in case of a negative and a positive signal edge:

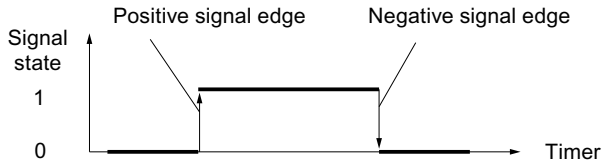

The positive signal edge is queried each time the instruction executes. When a positive signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, T, C, D, L	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved.

Example

The following example shows how the instruction works:

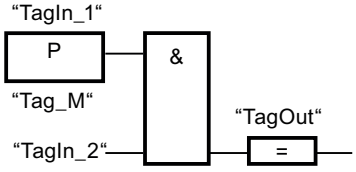

The "TagOut" output is set when the following conditions are fulfilled:

- There is a rising edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

N: Scan operand for negative signal edge

Description

You can use the "Scan operand for negative signal edge" instruction to determine whether there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

The following figure shows the change in the signal state in case of a negative and a positive signal edge:

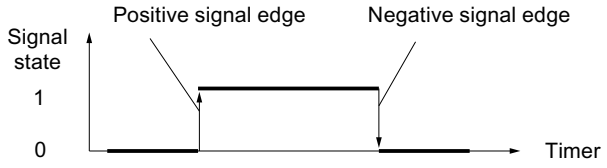

The negative signal edge is queried each time the instruction executes. When a negative signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, T, C, D, L	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved.

Example

The following example shows how the instruction works:

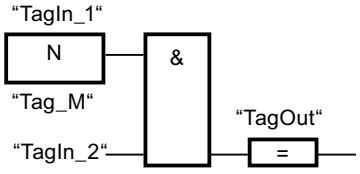

The "TagOut" output is set when the following conditions are fulfilled:

- There is a falling edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

Basic information on FBD (Page 7185)

P_TRIG: Scan RLO for positive signal edge

Description

Use the instruction "Scan RLO for positive signal edge" to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

The positive signal edge is queried each time the instruction executes. As soon as a positive single edge is detected, the output Q of the instruction returns the signal state "1" for the length of a program cycle. In all other cases, the output returns the signal state "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the instruction "Scan RLO for positive signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

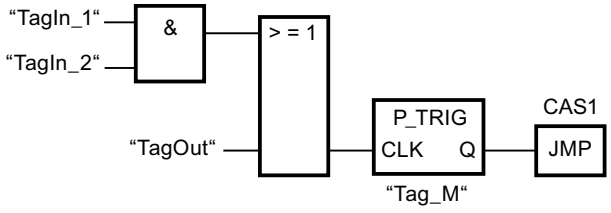

The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

Basic information on FBD (Page 7185)

N_TRIG: Scan RLO for negative signal edge

Description

Use the instruction "Scan RLO for negative signal edge" to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query saved in the edge memory bit (<Operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative signal edge.

The negative signal edge is queried each time the instruction executes. As soon as a negative single edge is detected, the output Q of the instruction returns the signal state "1" for the length of a program cycle. In all other cases, the signal state at the output of the instruction is "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

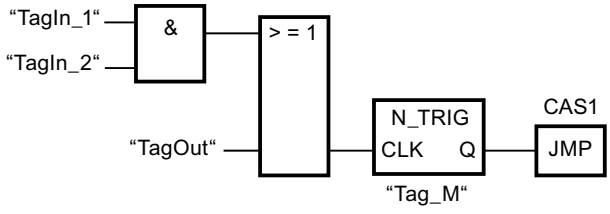

The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

Basic information on FBD (Page 7185)

Timer operations

IEC Timers

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set output Q for a programmed duration. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The execution of the instruction requires a preceding logic operation. It can be placed within or at the end of the network. The programmed time PT begins when the instruction starts. Output Q is set for the duration PT, regardless of the subsequent course of the input signal. Even when a new positive signal edge is detected, the signal state of the Q output is not affected as long as the PT duration is running.

You can query the current time value at the ET output. The time value starts at T#0s and ends when the value of the time duration PT is reached. If the time duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Each call of the "Generate pulse" instruction must be assigned an IEC Timer in which the instruction data is stored. You can declare an IEC Timer as follows:

- Declaration of a data block of the type TP (for example, "TP_DB")
- Declaration as a local tag of the type TP in the "Static" section of a block (for example, #MyTP_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the

instruction "Generate pulse" are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the "Generate pulse" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, T, C	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current timer value

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:

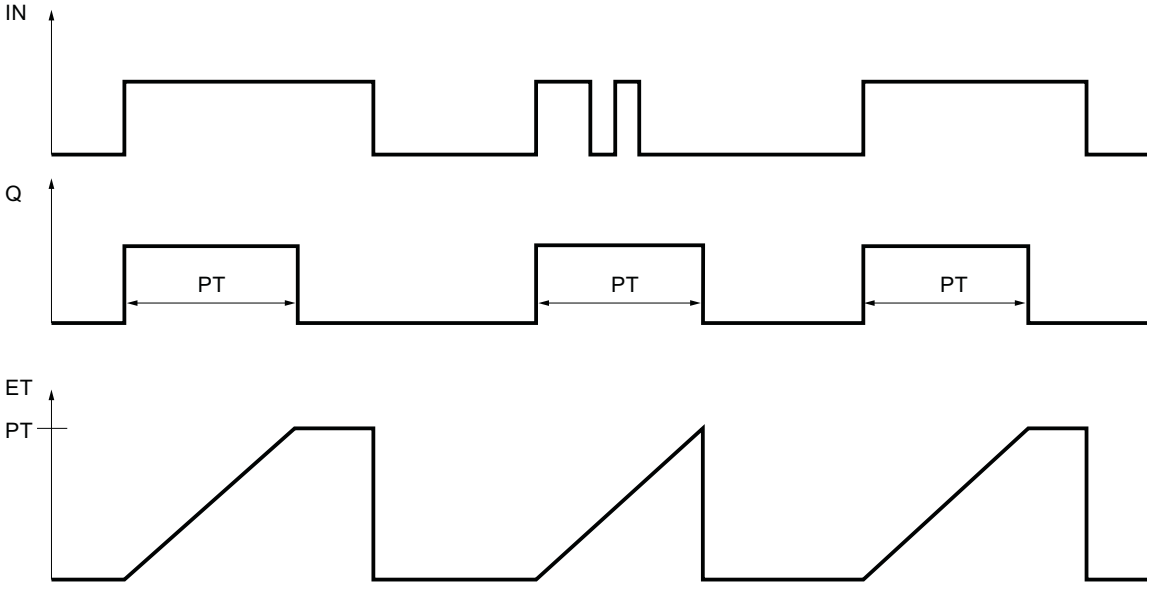

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the time configured with the PT time. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The execution of the instruction requires a preceding logic operation. It can be placed within or at the end of the network. The programmed time PT begins when the instruction starts. When the duration PT expires, output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of the time duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned an IEC Timer in which the instruction data is stored. You can declare an IEC Timer as follows:

- Declaration of a data block of the type TON (for example, "TON_DB")
- Declaration as a local tag of the type TON in the "Static" section of a block (for example, #MyTON_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, T, C	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:

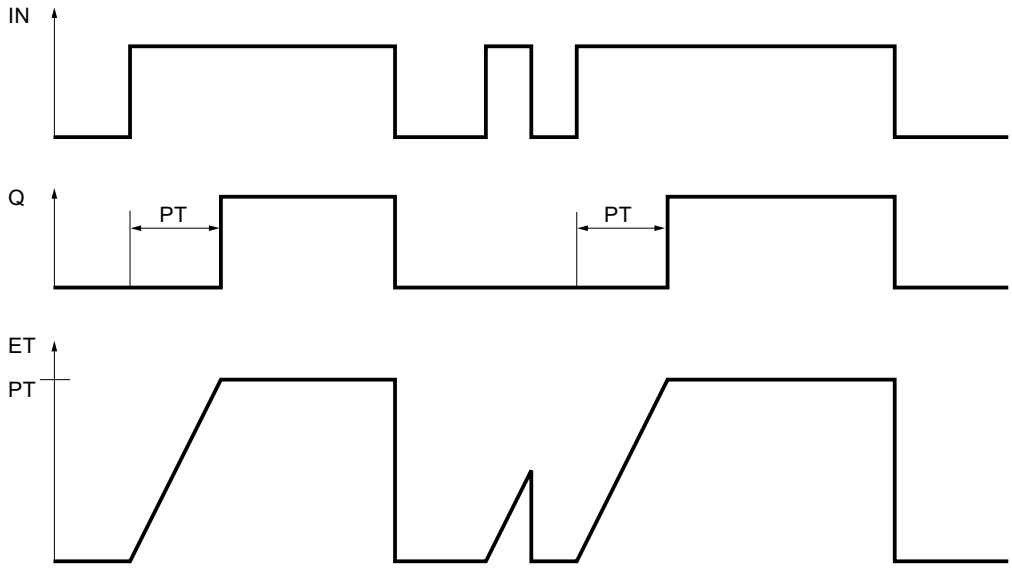

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay resetting of the Q output by the time configured with the PT time. The output Q is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The execution of the instruction requires a preceding logic operation. It can be placed within or at the end of the network. When the signal state at input IN changes back to "0", the programmed time PT starts. Output Q remains set as long as the time duration PT is running. When the PT time duration expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q will continue to be "1".

You can query the current time value at the ET output. The current timer value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET output remains set to the current value until input IN changes back to "1". If input IN changes to "1" before the time duration PT has expired, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC Timer in which the instruction data is stored. You can declare an IEC Timer as follows:

- Declaration of a data block of the type TOF (for example, "TOF_DB")
- Declaration as a local tag of the type TOF in the "Static" section of a block (for example, #MyTOF_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the "Generate off-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, T, C	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is reset when the timer PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate off-delay" instruction:

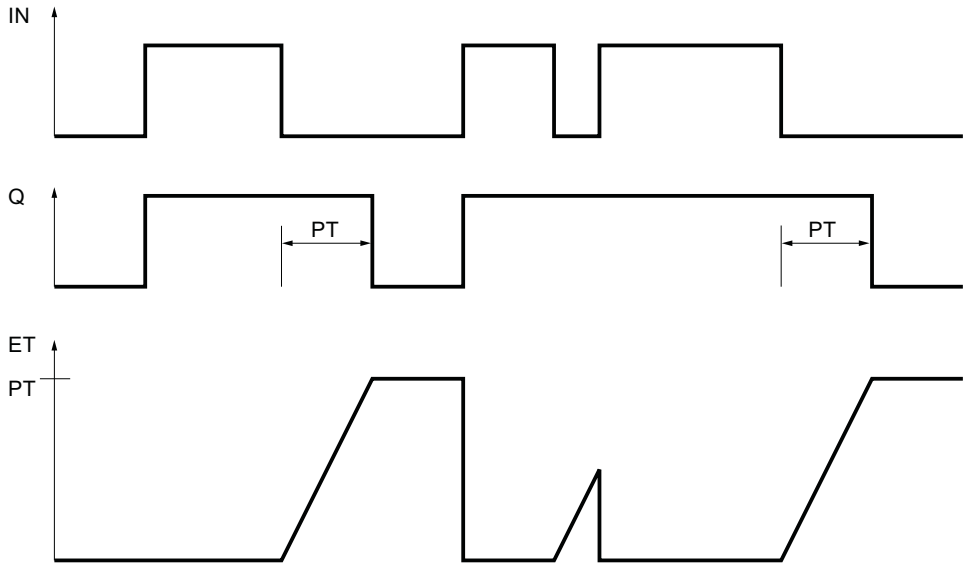

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

SIMATIC Timers

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as long as the signal state at input S is "1". If the signal state at input S changes to "0" before the programmed duration expires, the timer is stopped. In this case, the signal state at output Q is "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1", the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign pulse timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign pulse timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign pulse timer parameters and start" instruction:

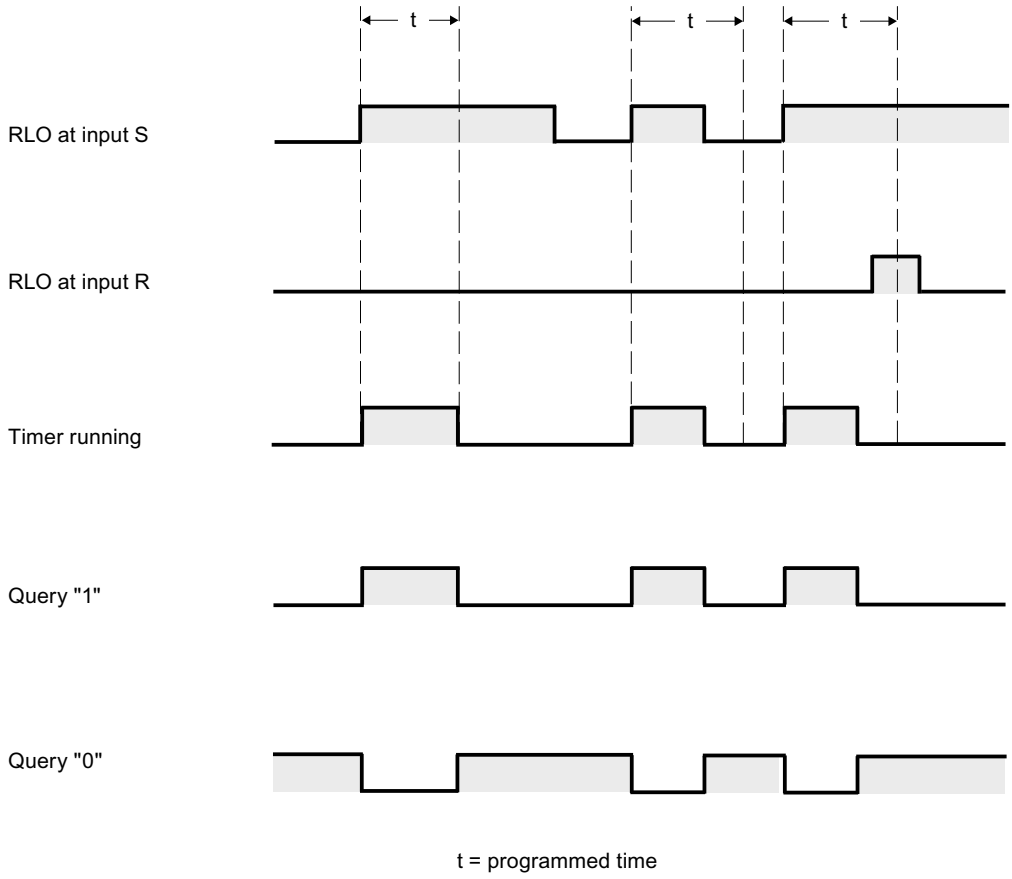

Example

The following example shows how the instruction works:

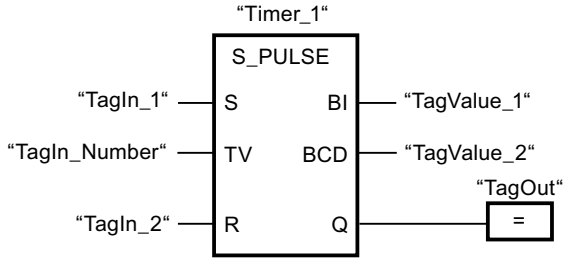

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer runs for the time value of the "TagIn_Number" operand as long as the "TagIn_1" operand has the signal state "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer "Timer_1" is stopped. The "TagOut" operand is reset to signal state "0".

The "TagOut" operand has the signal state "1" as long as the timer is running and the "TagIn_1" operand has the signal state "1". When the time has expired or is reset, the "TagOut" operand is reset to "0".

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on FBD (Page 7185)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". As long as the timer is running, output Q has the signal state "1". When the timer has expired, output Q is reset to "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input TV.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1", the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign extended pulse timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign extended pulse timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:

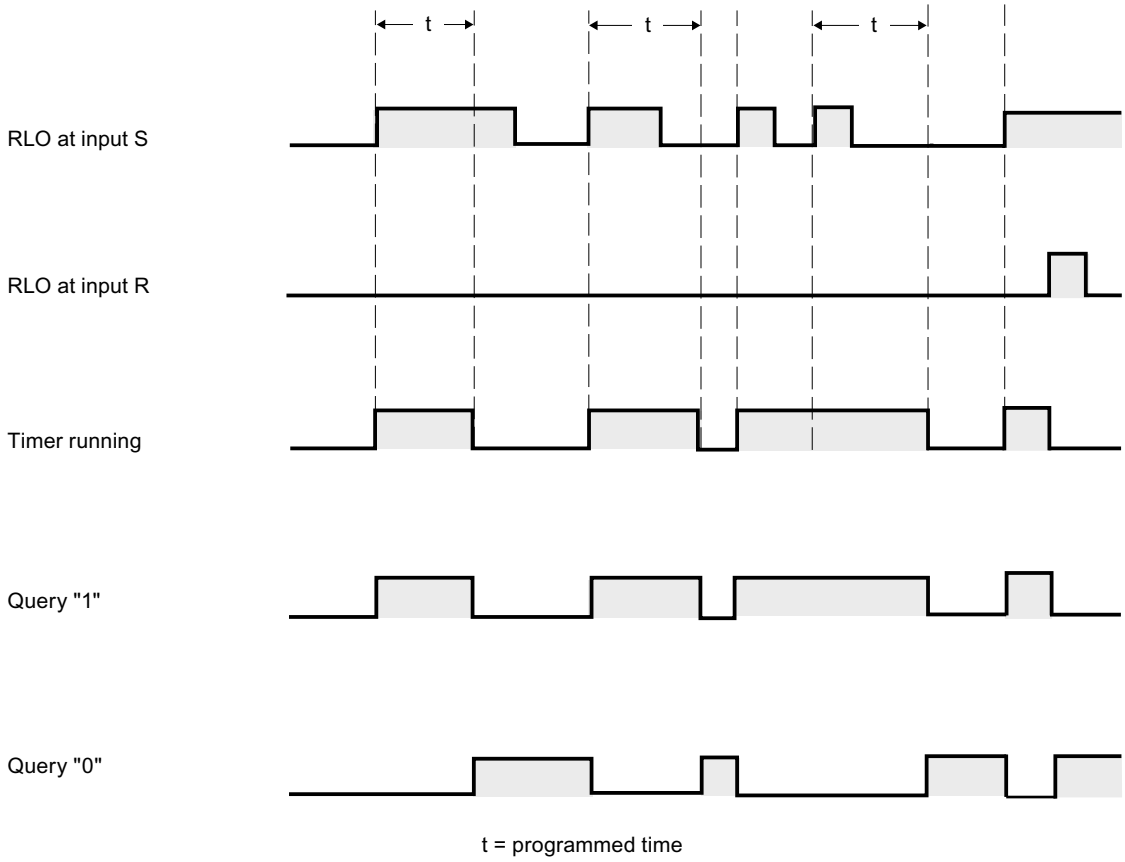

Example

The following example shows how the instruction works:

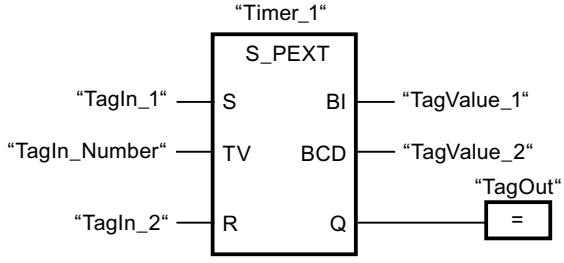

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer runs for the time value of the "TagIn_Number" operand without being affected by a negative edge at the S input. If the signal state at the "TagIn_1" operand changes from "0" to "1" before the timer expires, the timer is restarted.

The "TagOut" operand has the signal state "1" as long as the timer is running. When the time has expired or is reset, the "TagOut" operand is reset to "0".

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on FBD (Page 7185)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as long as the signal state at input S is "1". If the timer expires correctly and input S still has signal state "1", output Q returns signal state "1". If the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the time is running and the signal state at input R changes from "0" to "1", the current time value and the time base are also set to zero. In this case, the signal state at output Q is "0". The timer is reset if the signal state is "1" at the R input even if the timer is not running and the RLO at input S is "1".

The "Assign on-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:

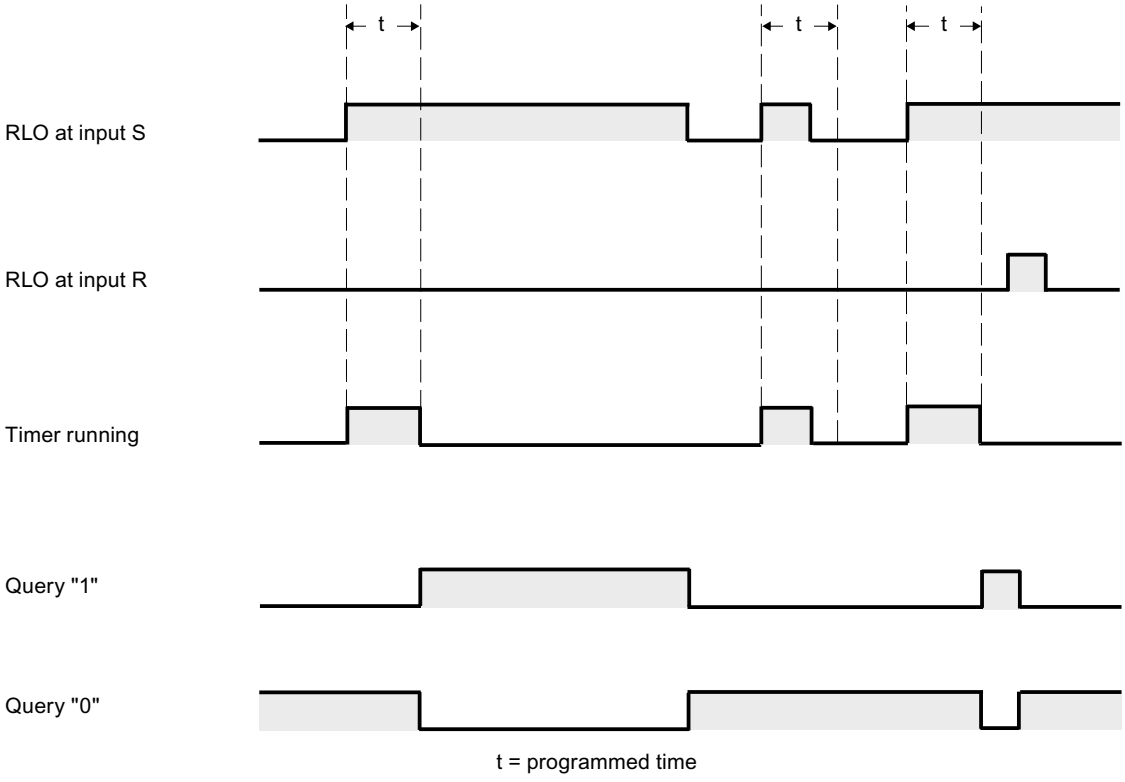

Example

The following example shows how the instruction works:

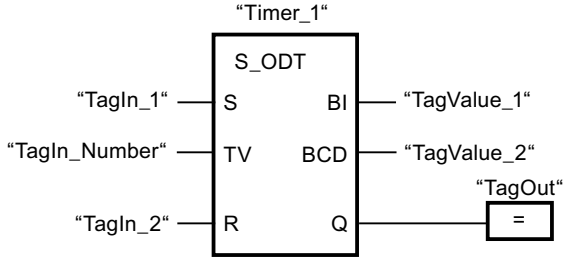

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". If the timer expires and the signal state of the operand is "1", the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. The "TagOut" operand has the signal state "0".

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on FBD (Page 7185)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". If the timer expires, the "Q" output returns signal state "1" regardless of the signal state at input "S". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input (TV).

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0" regardless of the signal state at start input S. In this case, the signal state at output Q is "0".

The "Assign retentive on-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign retentive on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:

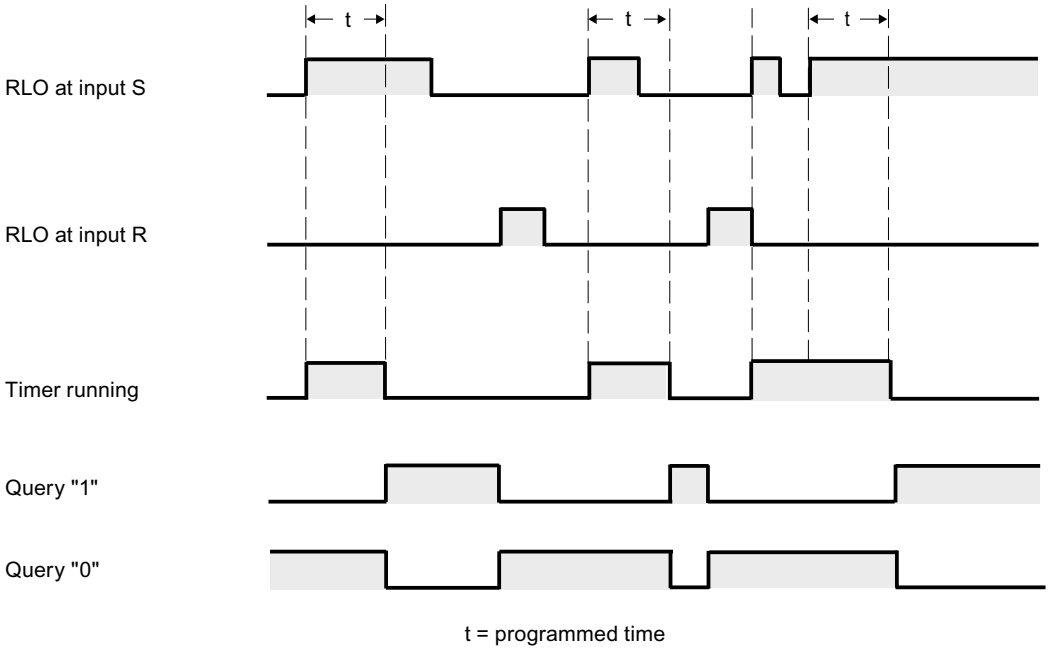

Example

The following example shows how the instruction works:

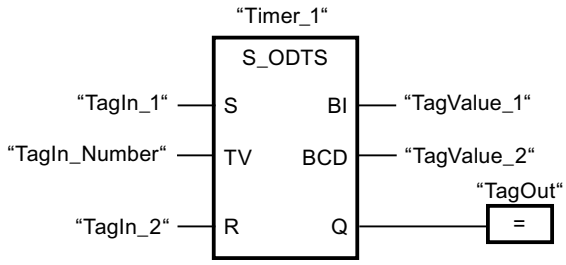

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand, even when the signal state of the "TagIn_1" operand changes to "0". When the time expires, the "TagOut" operand is reset to "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is restarted.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on FBD (Page 7185)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a transition from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV). As long as the timer is running or input S returns signal state "1", output Q has signal state "1". If the timer expires and the signal state is "0", output Q is reset to signal state "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is reset. The timer is only restarted after a falling signal edge is detected at input S.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0". In this case, the signal state at output Q is "0".

The "Assign off-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign off-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:

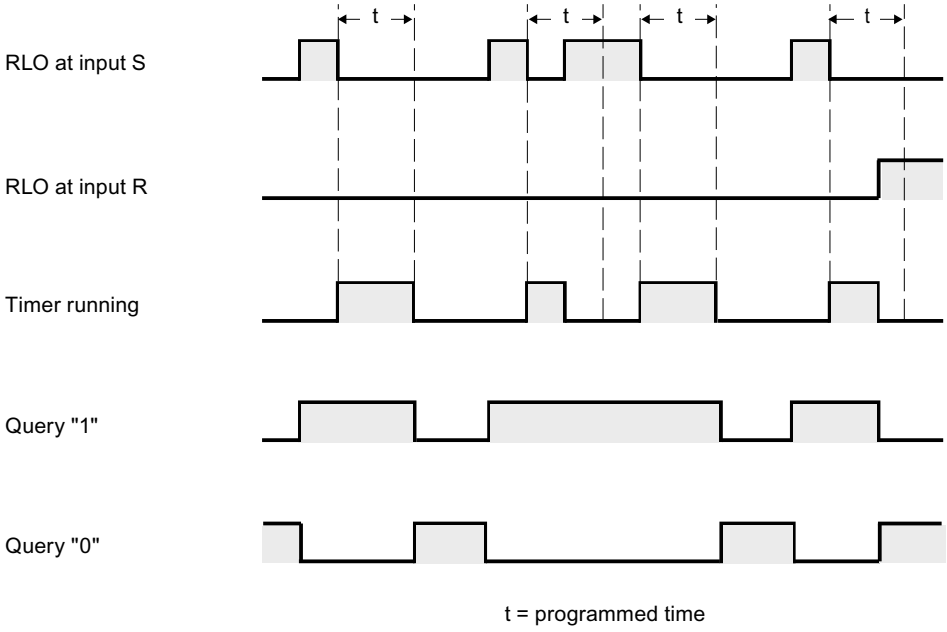

Example

The following example shows how the instruction works:

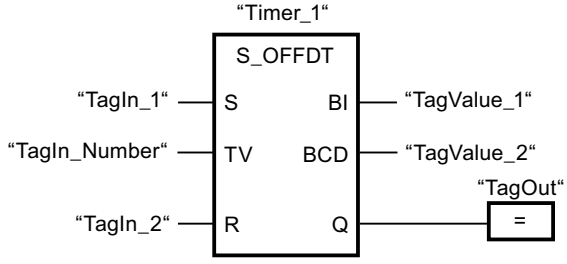

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "1" to "0". The timer expires with the value of operand "TagIn_Number". The "TagOut" operand is set to "1" if the timer is running or the "TagIn_1" operand has the signal state "0". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is reset.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on FBD (Page 7185)

SP: Start pulse timer

Description

The instruction "Start pulse timer" starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs with the specified duration as long as the RLO has the signal state "1". As long as the timer is running, the query of timer status "1" returns the signal state "1". If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer stops. In this case, the query for timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed timer value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction "Start pulse timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

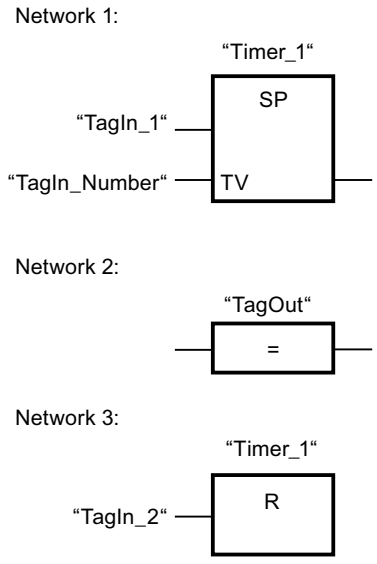

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand as long as the signal state of the "TagIn_1" operand is "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. As long as the timer is running, the "TagOut" operand returns signal state "1". A signal state change of the operand "TagIn_2" from "0" to "1" resets the timer, which stops the timer and sets the current time value to "0".

The figure below shows the timer diagram of the example:

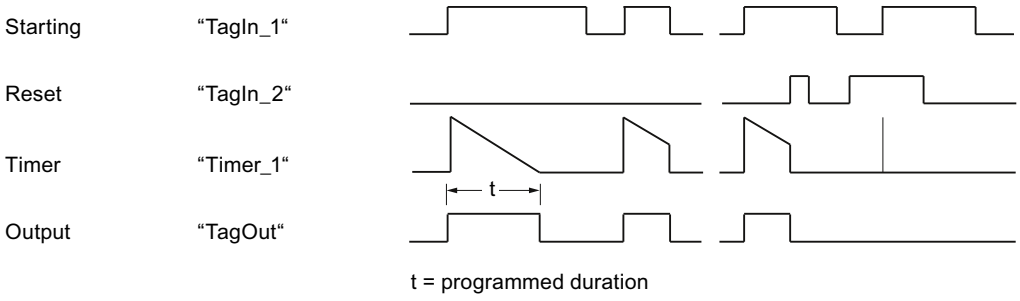

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basic information on FBD (Page 7185)

SE: Start extended pulse timer

Description

The "Start extended pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period even when the RLO changes to signal state "0". As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is restarted with the programmed time period. The querying of the timer status for "1" returns the signal state "0" when the timer expires.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed timer value is counted down towards zero. The time base indicates the time period with which the time value is changed.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameters

The following table shows the parameters of the "Start extended pulse timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

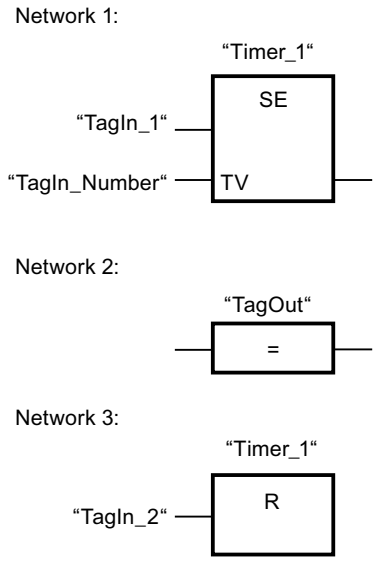

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative signal edge in the RLO. As long as the timer is running, the "TagOut" operand returns signal state "1". If the signal state of the operand "TagIn_1" changes again from "0" to "1" before the timer expires, the timer is restarted.

The figure below shows the timer diagram of the example:

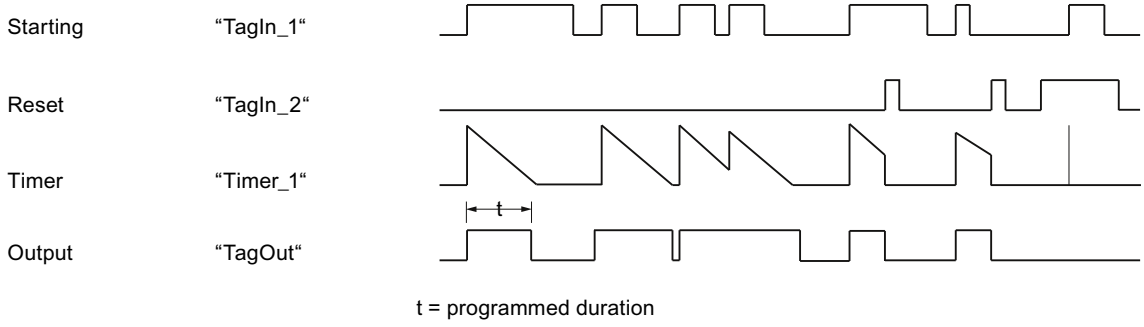

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

SD: Start on-delay timer

Description

The "Start on-delay timer" instruction starts a programmed timer when the signal state "1" is detected at the start input. As long as this signal state remains "1", the timer expires with the specified duration. When the timer expires and the signal state at the start input is still "1", the query for timer status also returns "1". If the signal at the start input is "0", the timer is reset. In this case, querying the timer status returns the signal state "0". As soon as the signal at the start input changes to "1" again, the timer starts running again.

The signal state of the timer output is identical to the signal state of the start input. The start input is directly interconnected to the output and is not linked to the timer.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also L: Load timer value (Page 4734).

Parameters

The following table shows the parameters of the instruction "Start on-delay timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

Network 1:

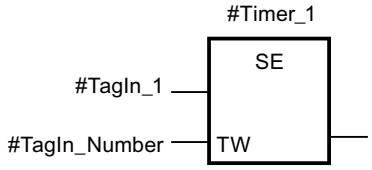

"#Timer_1" starts when the signal state of the "#TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "#TagIn_Number". If the signal state of the "#TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is reset.

Network 2:

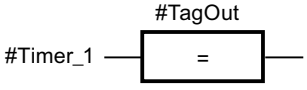

When the timer has expired, the #TagIn_1 operand at the start input has signal state "1" and the timer was not reset, the #TagOut operand is "1".

Network 3:

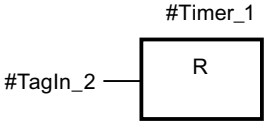

When the signal state of the #TagIn_2 operand has signal state "1", the timer #Timer_1 and the output #TagOut are reset.

If you want to restart #Timer_1, the signal state at the #TagIn_2 operand must be "0" and the signal state at the start input #TagIn_1 must change from "0" to "1".

The figure below shows the timer diagram of the example:

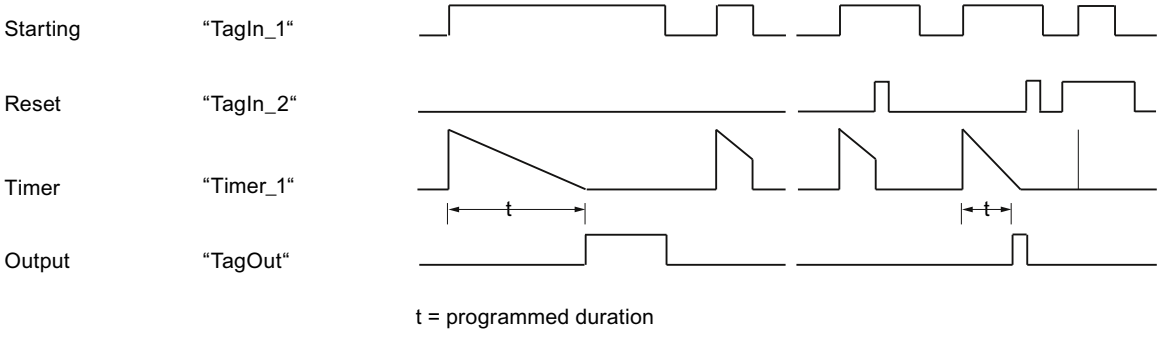

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

SS: Start retentive on-delay timer

Description

The "Start retentive on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period even when the RLO changes to signal state "0". When the timer expires, the query for timer status for "1" returns the signal state "1". When the timer expires, the timer can only be restarted if it is explicitly reset.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed timer value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start retentive on-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameters

The following table shows the parameters of the instruction "Start retentive on-delay timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

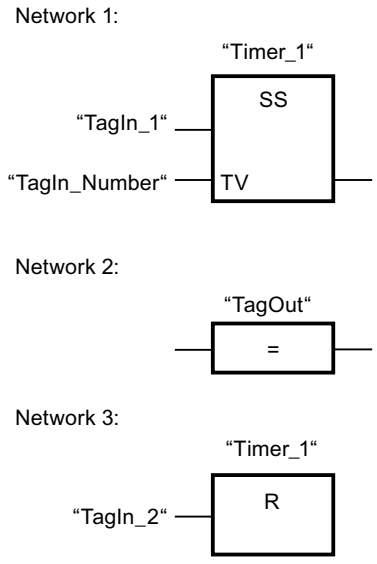

"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". When the time expires, the "TagOut" operand is reset to "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is restarted. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

The figure below shows the timer diagram of the example:

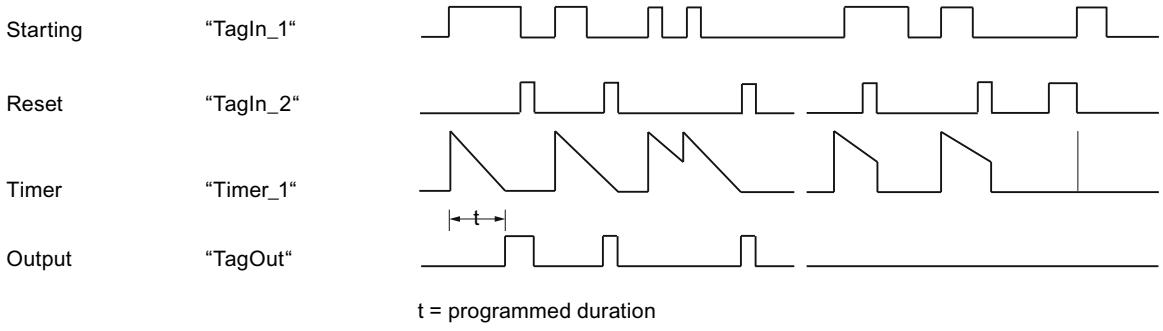

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

SF: Start off-delay timer

Description

The "Start off-delay timer" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at the start input. The timer expires with the specified duration. As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is reset. The timer is always restarted when the RLO changes from "1" to "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed timer value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The queries of the timer status for "1" return the query result "1" if the signal state of the result of logic operation is "1" at the execution of the instruction. If the RLO is "0", queries of the timer status for "1" return the query result "0".

The "Start off-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameters

The following table shows the parameters of the instruction "Start off-delay timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

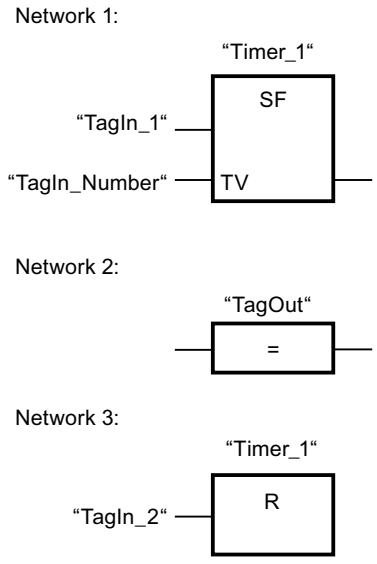

The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "1" to "0". The timer expires with the value of operand "TagIn_Number". As long as the timer is running, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" while the timer is running, the timer is restarted. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

The figure below shows the timer diagram of the example:

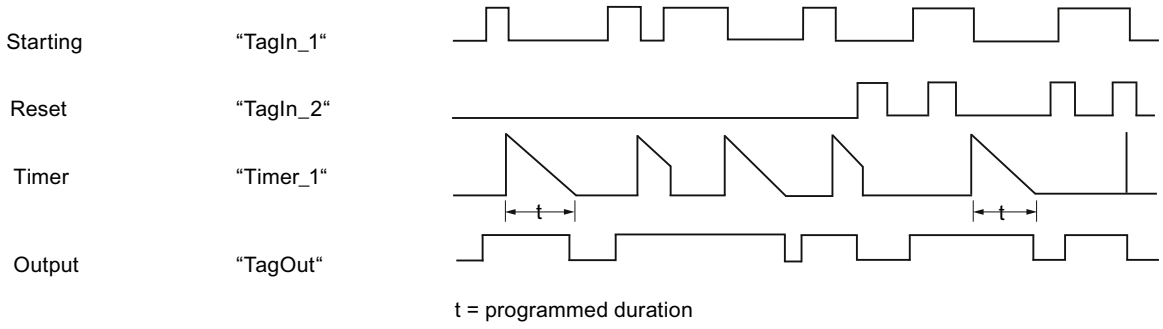

See also

Overview of the valid data types (Page 231)

Basic information on FBD (Page 7185)

Counter operations

IEC Counters

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is incremented by one. The counter value is increased each time a positive signal edge is detected, until it reaches the high limit of the data type (INT). When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can query the counter status in the Q output. The signal state at the Q output is determined by the parameter PV. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value at the CV output is reset to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC Counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")
- Declaration as a local tag of the type CTU in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the R parameter of the instruction in a startup OB. If instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
PV	Input	INT	I, Q, M, D, L, P, or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

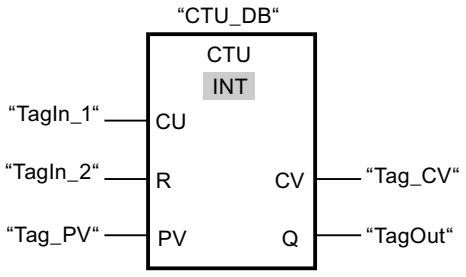

When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CV" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until it reaches the high limit of the data type (INT = 32767).

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on FBD (Page 7185)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type (INT). When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can query the counter status in the Q output. If the current counter value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value at the CV output is set to the value of the PV parameter when the signal state at the LD input changes to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC Counter as follows:

- Declaration of a data block of the type CTD (for example, "CTD_DB")
- Declaration as a local tag of the type CTD in the "Static" section of a block (for example, #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case the desired initial value for the CV parameter is specified in the PV parameter.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, T, C, P	Load input

Parameters	Declaration	Data type	Memory area	Description
PV	Input	INT	I, Q, M, D, L, P, or constant	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

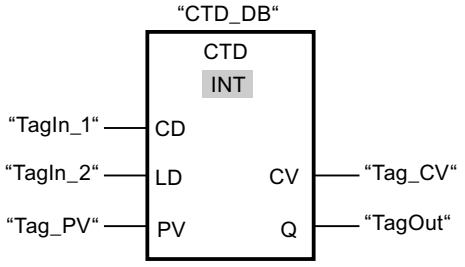

When the signal state of the "TagIn_1" operand changes from "0" to "1", the instruction is executed and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the counter value is decremented until it reaches the low limit of the data type (INT = -32768).

The "TagOut" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basic information on FBD (Page 7185)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit of the specified data type (INT). When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of the specified data type (INT) is reached, the counter value is not decremented any further.

When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter. As long as the LD input has signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0". You can also specify a constant for the PV parameter.

You can query the status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Each call of the "Count up and down" instruction must be assigned an IEC Counter in which the instruction data is stored. You can declare an IEC Counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")
- Declaration as a local tag of the type CTUD in the "Static" section of a block (for example, #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a startup OB:

- When used as up counter, the value at the R parameter must be set to "1".
- When used as down counter, the value at the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, T, C, P	Load input
PV	Input	INT	I, Q, M, D, L, P, or constant	Value at which the output QU is set. / Value to which the CV output is set with LD = 1.
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	INT	I, Q, M, D, L, P	Current counter value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

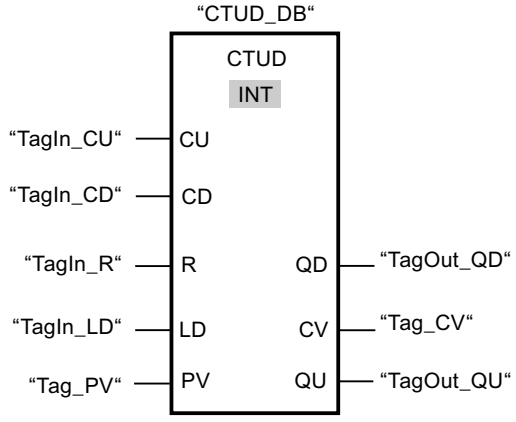

If the signal state at the "TagIn_CU" or "TagIn_CD" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_CU" input, the current counter value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_CD" input, the counter value is decremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the CU input, the counter value is incremented until it reaches the high limit of INT = 32767. If input CD has a positive signal edge, the counter value is decremented until it reaches the low limit of INT = -32768.

The "TagOut_QU" output has signal state "1" as long as the current counter value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut_QU" output has signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut_QD" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

SIMATIC Counters

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CU is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Avoid using a counter at more than one point in the program (risk of counting errors).

The "Assign parameters and count up" instruction requires a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L, T, C	Count up input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter

Parameter	Declaration	Data type	Memory area	Description
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexa-decimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

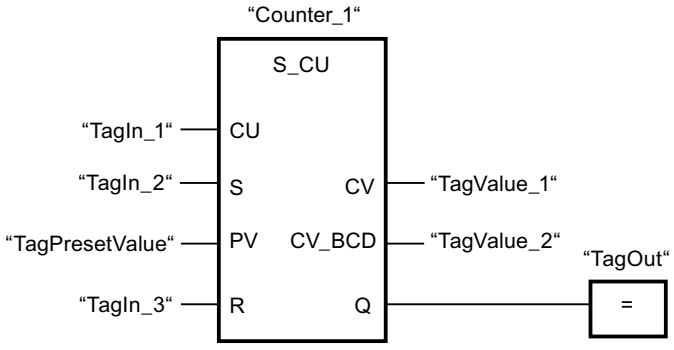

When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. When the signal state at the "TagIn_2" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be decremented until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CD is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Avoid using a counter at more than one point in the program (risk of counting errors).

The "Assign parameters and count down" instruction requires a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

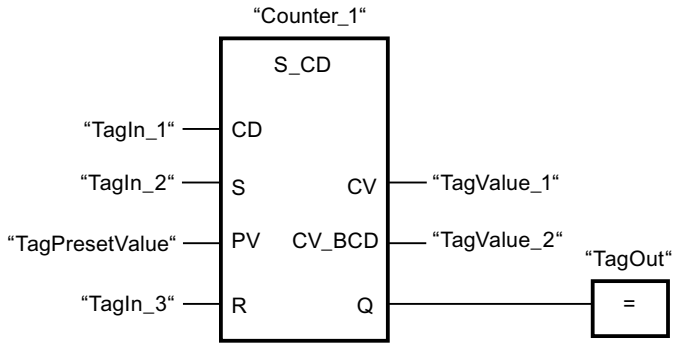

When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decremented by one. When the signal state at the "TagIn_2" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment or decrement the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. If there is a positive signal edge at the CU and CD inputs in one program cycle, the counter value remains unchanged.

The counter value can be incremented until the high limit of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of "0" is reached, the counter value is no longer decremented.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO is "1" at the inputs CU and CD, the counter counts accordingly in the next scan cycle, even if no change in the signal edge is detected.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU, CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Avoid using a counter at more than one point in the program (risk of counting errors).

The "Assign parameters and count up / down" instruction requires a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Assign parameters and count up / down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L, T, C	Count up input
CD	Input	BOOL	I, Q, M, D, L, T, C	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

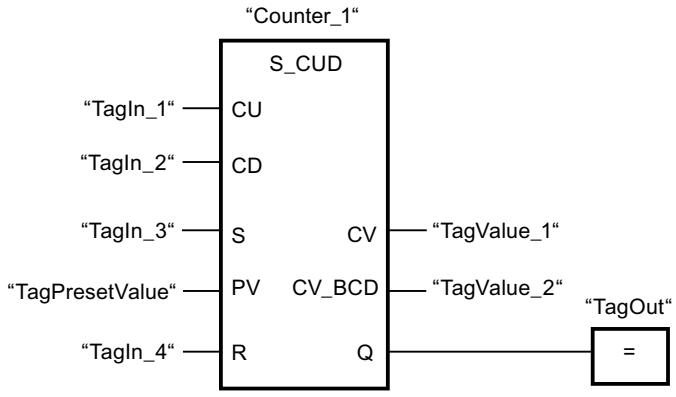

If the signal state at the "TagIn_1" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Assign parameters and count up / down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input and the current counter value is less than "999", the counter value is incremented by one. When there is a positive signal edge at the "TagIn_2" input and the current counter value is greater than "0", the counter value is decremented by one.

When the signal state at the "TagIn_3" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_4" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

SC: Set counter value

Description

You can use the "Set counter value" instruction to increment the value of a counter. The instruction is executed when the result of logic operation (RLO) at the start input of the instruction changes from "0" to "1". If the instruction is executed, the counter is set to the specified counter value.

The "Set counter value" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Set counter value" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
PV	Input	WORD	I, Q, M, D, L or constant	Value with which the counter is preset in the BCD format. (C#0 to C#999)
<Counter>	InOut/Input	COUNTER	C	Counter which is preset.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

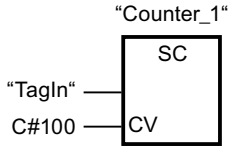

"Counter_1" starts with the value "100" when the signal state of the "TagIn" operand changes from "0" to "1".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

CU: Count up

Description

You use the "Count up" instruction to increment the value of the specified counter by the count of one on a rising edge at the start input. The count is incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

The "Count up" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
<Counter>	InOut/Input	COUNTER	C	Counter whose value is incremented.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

Network 1

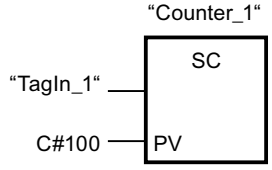

Network 2

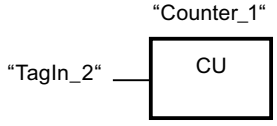

Network 3

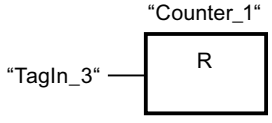

When the signal state of the operand "TagIn_1" changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of "Counter_1" is incremented by the count of one when the signal state of the "TagIn_2" operand changes from "0" to "1".

If the "TagIn_3" operand returns signal state "1", the value of "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

CD: Count down

Description

You use the "Count down" instruction to decrement the value of the specified counter by the count of one on a rising edge at the start input. The count is decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer changed on a positive signal edge.

The "Count down" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
<Counter>	InOut/Input	COUNTER	C	Counter whose value is decremented.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

Network 1

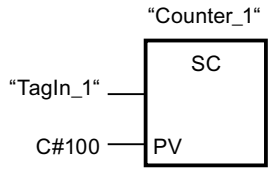

Network 2

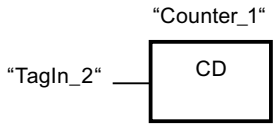

Network 3

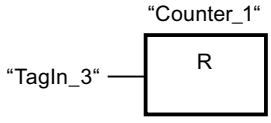

When the signal state of the operand "TagIn_1" changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of "Counter_1" is incremented by the count of one when the signal state of the "TagIn_2" operand changes from "0" to "1".

If the "TagIn_3" operand returns signal state "1", the value of "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 231)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to query whether the value at input IN1 is equal to the value at input IN2.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Note

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, instead of the instruction "CMP ==: Equal", use the instruction "IN_RANGE: Value within range".

Parameters

The following table shows the parameters of the instruction "Equal":

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:

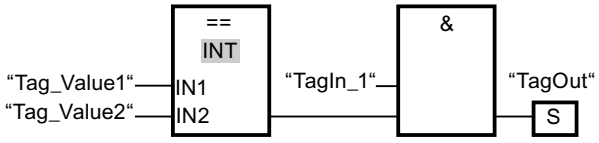

Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" = "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to query whether the value at input IN1 is not equal to the value at input IN2.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction "Not equal":

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

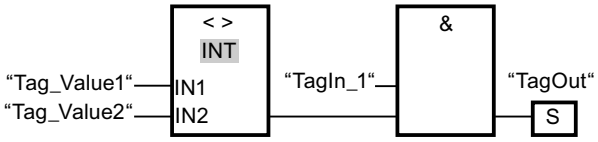

Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <> "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to query whether the value at input IN1 is greater or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction "Greater or equal":

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

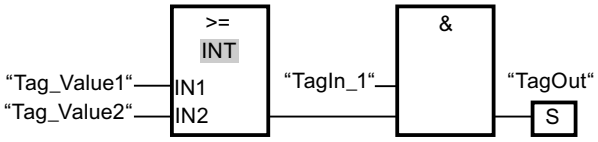

Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" >= "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

FBD programming examples (Page 7232)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to query whether the value at input IN1 is less or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction "Less or equal":

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

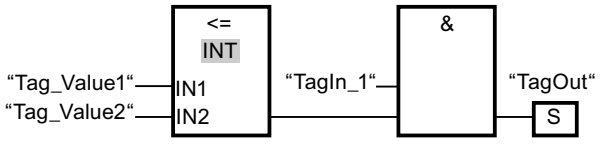

Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <= "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

CMP >: Greater than

Description

You can use the "Greater than" instruction to query whether the value at input IN1 is greater than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction "Greater than":

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

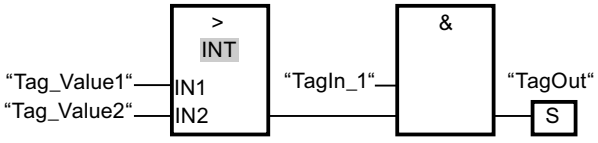

Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" > "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

CMP <: Less than

Description

You can use the "Less than" instruction to query whether the value at input IN1 is less than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction "Less than":

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

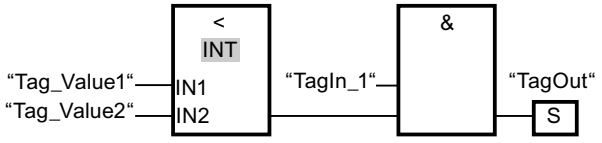

Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" < "Tag_Value2".

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

FBD programming examples (Page 7232)

Math functions

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 to the value at input IN2 and query the sum at output OUT (OUT := IN1+IN2).

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Add":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second number to be added
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Sum

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:

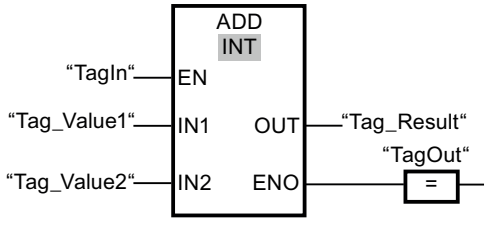

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is

stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at output OUT (OUT := IN1-IN2).

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Subtract":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Subtrahend
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Difference

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

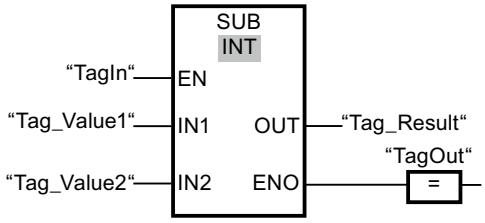

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on FBD (Page 7185)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at input IN1 with the value at input IN2 and query the total at output OUT (OUT := IN1*IN2).

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Multiply":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Multiplier

Parameters	Declaration	Data type	Memory area	Description
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Multiplicand
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Product

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

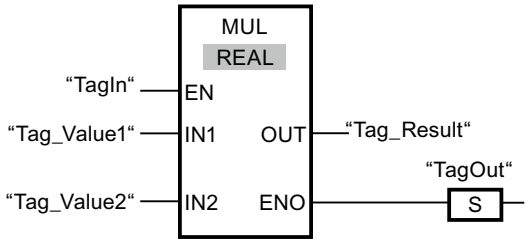

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is multiplied with the value of operand "Tag_Value2". The result of the multiplication is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Overview of the valid data types (Page 231)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT (OUT := IN1/IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Divide":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Quotient value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

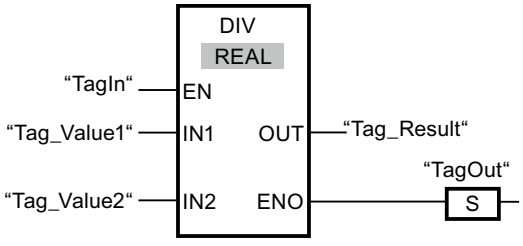

If the operand "TagIn" has the signal state "1", the "Divide" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The result of the division is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

FBD programming examples (Page 7232)

Basic information on FBD (Page 7185)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at input IN1 by the value at input IN2 and query the remainder of division at output OUT.

Parameters

The following table shows the parameters of the "Return remainder of division" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	DINT	I, Q, M, D, L, P or constant	Dividend
IN2	Input	DINT	I, Q, M, D, L, P or constant	Divisor
OUT	Output	DINT	I, Q, M, D, L, P	Remainder of division

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

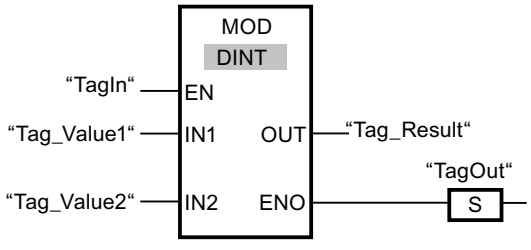

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder of division is stored in operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on FBD (Page 7185)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at input IN and query the result at output OUT. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the valid range for the data type specified at the OUT output (only applies to integers).

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

Parameters	Declaration	Data type		Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

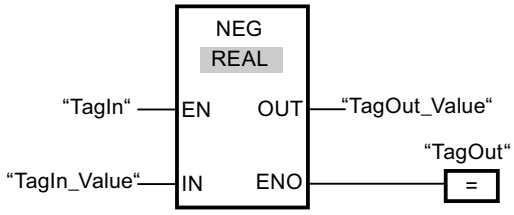

If the operand "TagIn" has the signal state "1", the "Create twos complement" instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is provided at output "TagOut_Value". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on FBD (Page 7185)

ABS: Form absolute value

Description

You can use the "Form absolute value" instruction to calculate the absolute value of the value specified at input IN. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form absolute value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

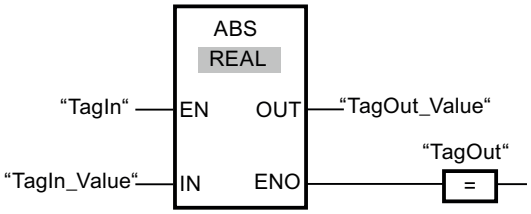

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the lowest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters, even when the IEC check is disabled. If you use the data types WORD or DWORD at the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Second input value
IN3	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Third input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

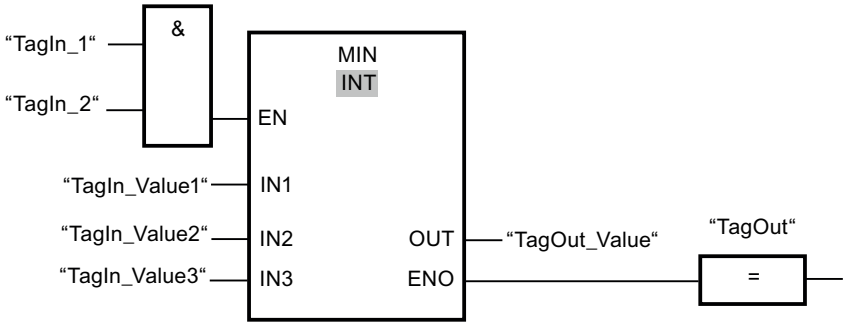

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Removing inputs and outputs (Page 7156)

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the highest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters, even when the IEC check is disabled. If you use the data types WORD or DWORD at the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Second input value
IN3	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Third input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

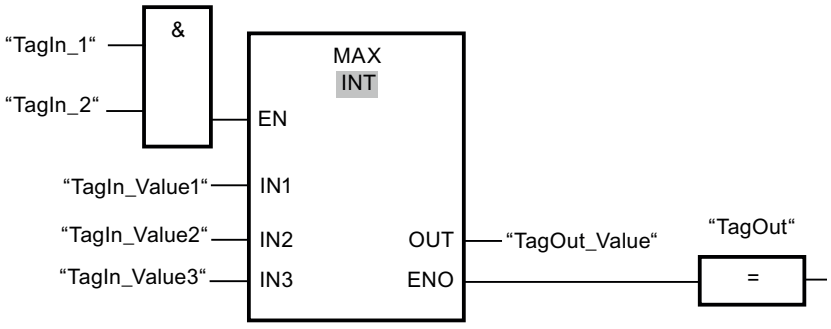

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN, output OUT is set to the value of the input MN. If the high limit MX is exceeded, output OUT is set to the value of the input MX.

If the value at input MN is greater than that at input MX, the result is undefined and the enable output ENO is "0".

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN input is greater than the value at the MX input.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters, even when the IEC check is disabled. If you use the data types WORD or DWORD at the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MN	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Low limit
IN	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Input value
MX	Input	Integers, floating-point numbers	I, Q, M, D, L, P	High limit
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

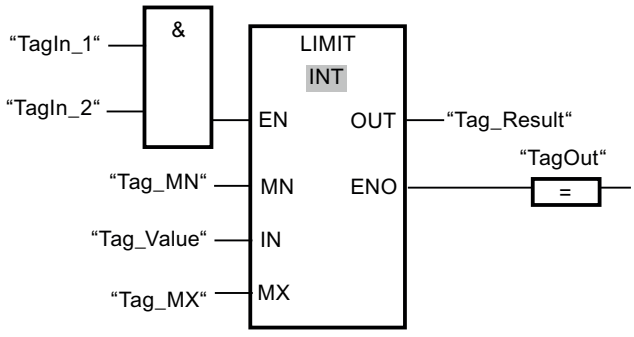

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value of the operand "Tag_Value" is less than the low limit, the value of the operand "Tag_MN" is copied to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)
Basics of the EN/ENO mechanism (Page 141)
Basic information on the status word (Page 173)
Setting the status bits in instructions with integers (Page 178)
Setting the status bits in instructions with floating-point numbers (Page 179)
Inserting FBD elements (Page 7198)
Selecting the data type of an FBD element (Page 7201)
Editing FBD elements (Page 7215)
Inserting operands in FBD instructions (Page 7222)

SQR: Form square

Description

The instruction "Form square" is used to square the value of a floating-point number at input IN and to write the result to output OUT.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form square":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

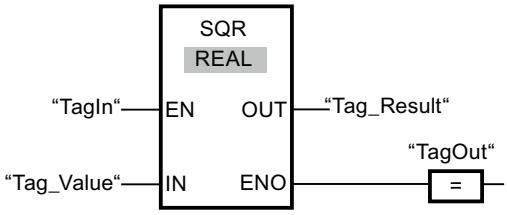

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction squares the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

SQRT: Form square root

Description

The instruction "Form square root" is used form the square root from the value of the floating-point number at input IN and to write the result to output OUT. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number. If the value at input IN is "0", the result is also "0".

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form square root":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

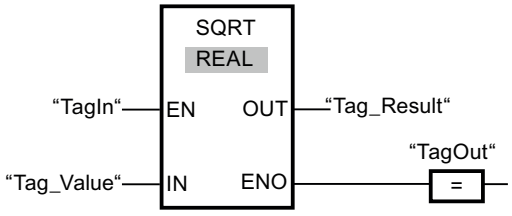

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the square root of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2718282) of the value at input IN. The result is sent to output OUT and can be queried there. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form natural logarithm":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

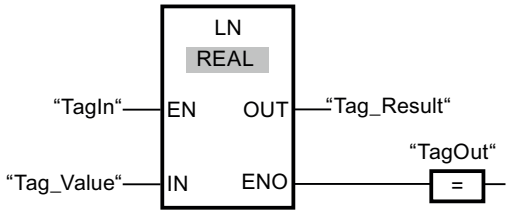

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction forms the natural logarithm of the value at input "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)
Basics of the EN/ENO mechanism (Page 141)
Basic information on the status word (Page 173)
Setting the status bits in instructions with integers (Page 178)
Setting the status bits in instructions with floating-point numbers (Page 179)
Inserting FBD elements (Page 7198)
Selecting the data type of an FBD element (Page 7201)
Editing FBD elements (Page 7215)
Inserting operands in FBD instructions (Page 7222)

EXP: Form exponential value

Description

You can use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value specified at input IN. The result is provided at output OUT and can be queried there (OUT = e^{IN}).

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form exponential value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Exponential value of input value IN

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

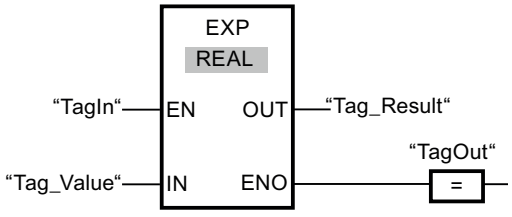

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction forms the exponent from base e and the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

SIN: Form sine value

Description

You can use the "Form sine value" instruction to calculate the sine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form sine value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of the angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Sine of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

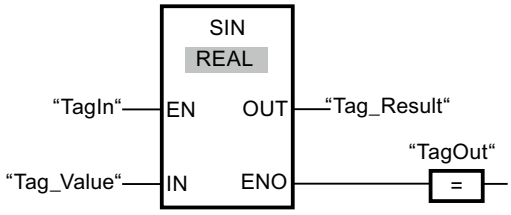

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the sine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

COS: Form cosine value

Description

You can use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form cosine value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of the angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Cosine of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

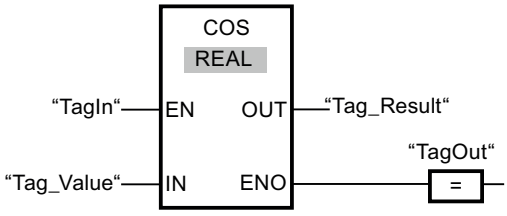

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the cosine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)
Basics of the EN/ENO mechanism (Page 141)
Basic information on the status word (Page 173)
Setting the status bits in instructions with integers (Page 178)
Setting the status bits in instructions with floating-point numbers (Page 179)
Inserting FBD elements (Page 7198)
Selecting the data type of an FBD element (Page 7201)
Editing FBD elements (Page 7215)
Inserting operands in FBD instructions (Page 7222)

TAN: Form tangent value

Description

You can use the "Form tangent value" instruction to calculate the tangent of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form tangent value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of the angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Tangent of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

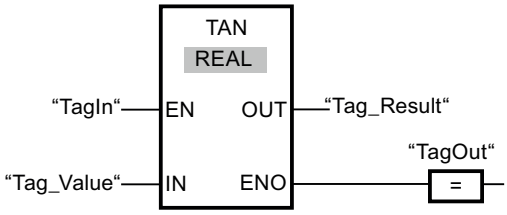

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the tangent of the angle specified at input "Tag_Value" and stores the result at output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from the sine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

The following table shows the parameters of the instruction "Form arcsine value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of the angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

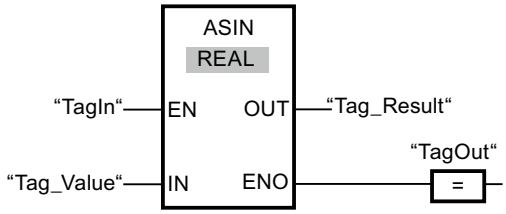

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from the cosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is output in radians at output OUT and can range in value from 0 to +π.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

The following table shows the parameters of the instruction "Form arccosine value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of the angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

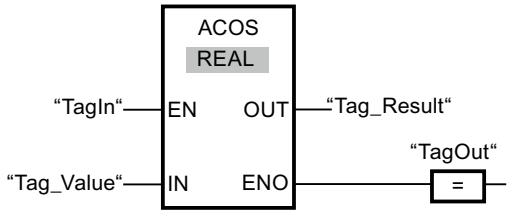

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from the tangent value specified at input IN, which corresponds to this value. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) at input IN. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form arctangent value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Tangent value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of the angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

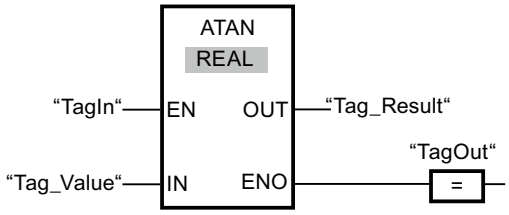

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_Value	1,0
OUT	Tag_Result	+0.785398 (π/4)

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Invalid floating-point numbers (Page 253)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of the higher address.

The following table shows the possible transfers:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD, CHAR
WORD	WORD; DWORD	BYTE, WORD, DWORD, INT, DINT, TIME, S5TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, INT, DINT, REAL, TIME, DATE, TOD, CHAR
INT	INT	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
TIME	TIME	BYTE, WORD, DWORD, INT, DINT, TIME
S5TIME	S5TIME	WORD, S5TIME
DATE	DATE	BYTE, WORD, DWORD, INT, DINT, DATE
TOD	TOD	BYTE, WORD, DWORD, INT, DINT, TOD
CHAR	CHAR	BYTE, WORD, DWORD, CHAR
COUNTER	INT, WORD, COUNTER	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	INT, WORD, TIMER	WORD, DWORD, INT, UINT, DINT, UDINT

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at input IN is less than the bit length of the data type at output OUT1, the higher-order bits of the destination value will be overwritten with zeros.

You can also use the "Move block" (BLKMOV) and "Move block uninterruptible" (UBLKMOV) instructions to copy fields and structures.

Parameters

The following table shows the parameters of the "Move value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings, integers, floating-point numbers, timers, DATE, TOD, CHAR, TIMER, COUNTER	I, Q, M, D, L or constant	Source value
OUT1	Output	Bit strings, integers, floating-point numbers, timers, DATE, TOD, CHAR, TIMER, COUNTER	I, Q, M, D, L	Destination address

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

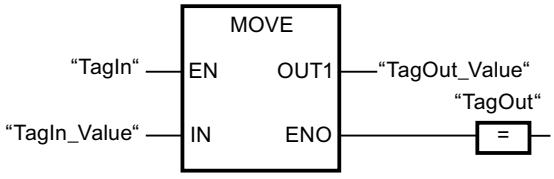

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies the content of the operand "TagIn_Value" to the operand "TagOut_Value" and set output "TagOut" to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

BLKMOV: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:

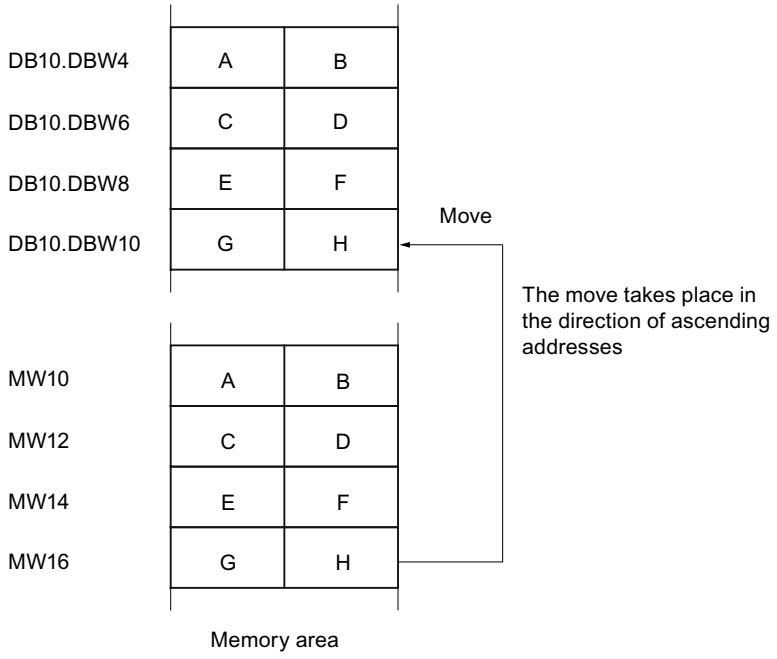

Consistency of source and destination data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, BLKMOV is interrupted while copying from a DB that is not runtime-relevant, the execution of BLKMOV can no longer be nested.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks not runtime-relevant

General rules for moving

The source and destination areas must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the specified length of the area must be divisible by 8, otherwise it will not be possible to execute the instruction.

Note

If an actual source or destination area is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes to the SRCBLK and DSTBLK parameters.

Rules for moving data blocks that are not runtime-relevant

The source area is in a data block in load memory that is not runtime-relevant. Data blocks that are not runtime-relevant are indicated by the keyword UNLINKED.

If a non-runtime-relevant data block is copied to the work memory with the "Move block" instruction and loaded at the same time, for example, by the programming device, the execution of the instruction can be delayed for several milliseconds. This results in a longer OB cycle and may trip the cycle monitoring.

If a non-runtime-relevant data block is moved with the "Move block" instruction and the move operation is interrupted, the execution of the instruction can no longer be continued.

If the instruction Read from data block in the load memory is available on your CPU, you must use this instruction to read data blocks that are not runtime-relevant from load memory. If you use the "Move block" instruction, error W#16#8092 is output.

Parameters

The following table shows the parameters of the instruction "Move block":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
DSTBLK	Output [1)]	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

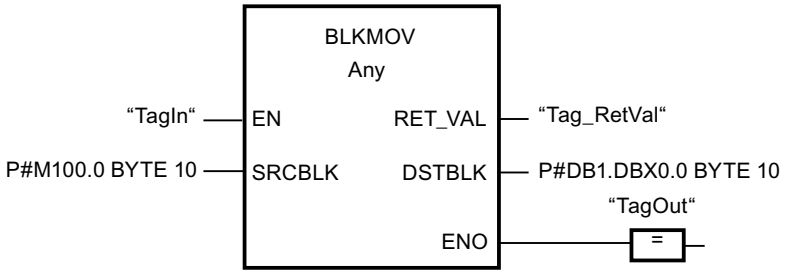

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

The move operation cannot be interrupted by other operating system activities. As a result the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the specified length of the area must be divisible by 8, otherwise it will not be possible to execute the instruction.

You can use the "Move block uninterruptible" instruction to move a maximum of 512 bytes. Note the CPU-specific restrictions for this.

Note

If an actual source or destination area is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
DSTBLK	Output [1)]	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source or destination area is located in the load memory only.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

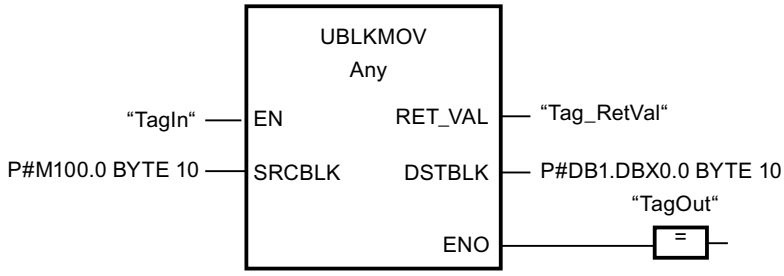

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

FILL: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:

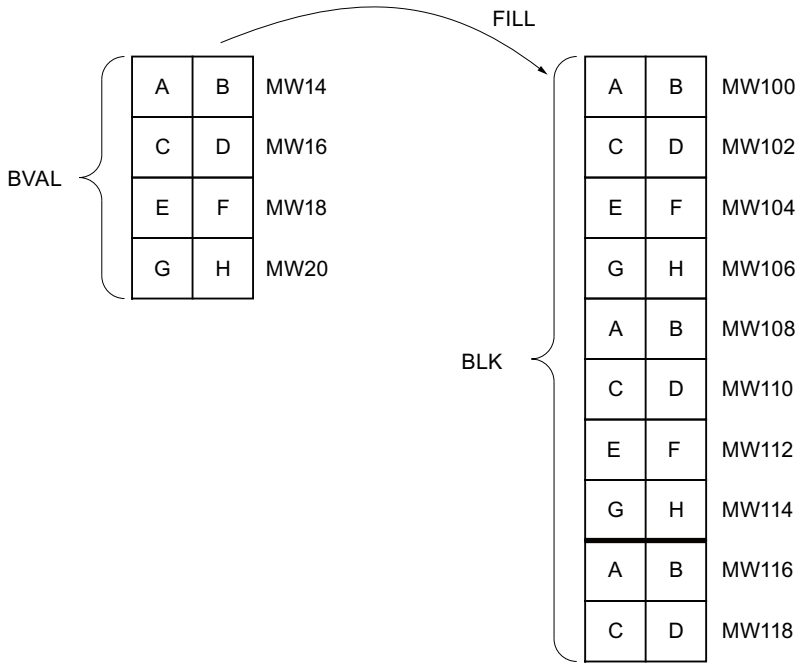

Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is data type STRING, the instruction writes the entire string including the administration information.

Note

If an actual source or destination area is smaller than a destination or source area specified on the BVAL or BLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
BVAL	Input	ANY	I, Q, M, D, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
BLK	Output [1)]	ANY	I, Q, M, D, L, P	Specification of the memory area that will be filled with the content of the source area.
1) The BLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is located in the load memory only.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

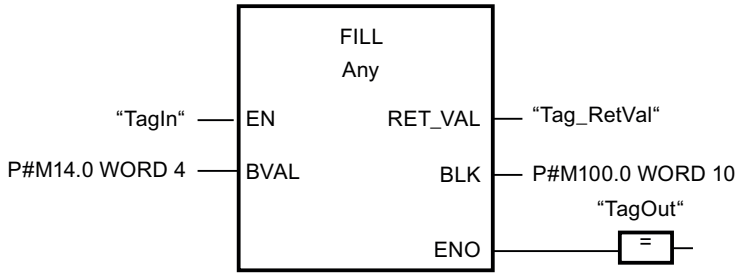

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types selected in the instruction box. The converted value is output at the OUT output.

For information on possible conversions, refer to the "Explicit conversion" section at "See also".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Convert value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted.
OUT	Output	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

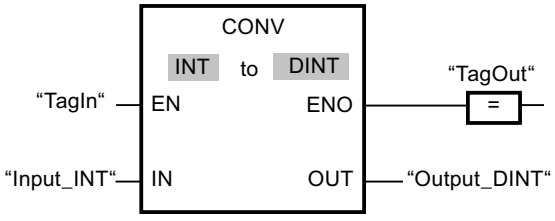

If the operand "TagIn" has the signal state "1", the content of the operand "TagIn_Value" is read and converted to an integer (16 bits). The result is stored in the operand "TagOut_Value". The output "TagOut" is set to "1" if the instruction was executed without errors.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with floating-point numbers (Page 179)

Setting the status bits in instructions with integers (Page 178)

Explicit conversions (Page 569)

Basic information on FBD (Page 7185)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next higher integer of the data type DINT. If the input value is exactly between an even and odd number, the even number is selected. The result of the instruction is output at the OUT output and can be queried there.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be rounded.
OUT	Output	DINT	I, Q, M, D, L, P	Result of the rounding

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

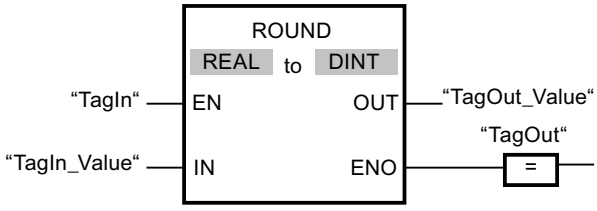

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on FBD (Page 7185)

CEIL: Generate next higher integer from floating-point number

Description

You can use the "Generate next higher integer from floating-point number" instruction to round the value at input IN to the next higher integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next higher integer. The result of the instruction is output at the OUT output and can be queried there. The output value can be greater than or equal to the input value.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next higher integer from floating-point number":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with the next higher integer

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

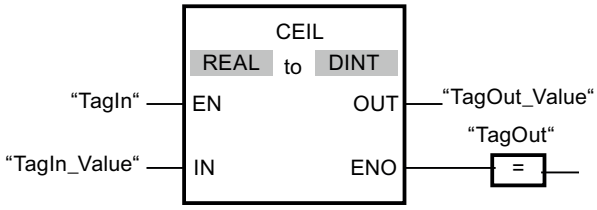

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next higher integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on FBD (Page 7185)

FLOOR: Generate next lower integer from floating-point number

Description

You can use the "Generate next lower integer from floating-point number" instruction to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is provided at output OUT and can be queried there. The output value can be less than or equal to the input value.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next lower integer from floating-point number":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with the next lower integer

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

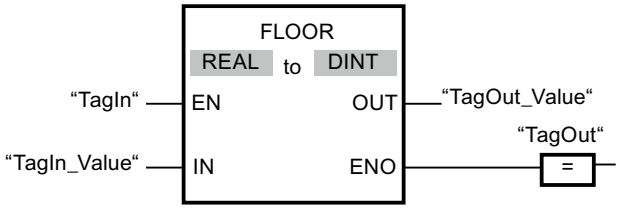

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and displayed at output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on FBD (Page 7185)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at input IN. The value at input IN is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to output OUT without decimal places.

The enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Truncate numerical value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L or constant	Input value
OUT	Output	Integers	I, Q, M, D, L	Integer component of the input value

You can select the data types of the instruction from the "???" drop-down lists of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

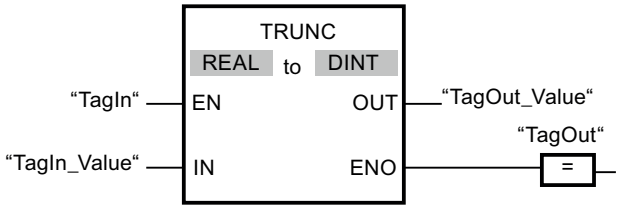

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

If the "TagIn" operand has signal state "1", the instruction is executed. The integer part of the floating-point number at input "TagIn_Value" is converted to an integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

SCALE: Scale

Description

Use the "Scale" instruction to convert the integer in the IN parameter into a floating-point number, which can be scaled in physical units between a low limit value and a high limit value. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible in the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit

Parameters	Declaration	Data type	Memory area	Description
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1".
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

The following example shows how the instruction works:

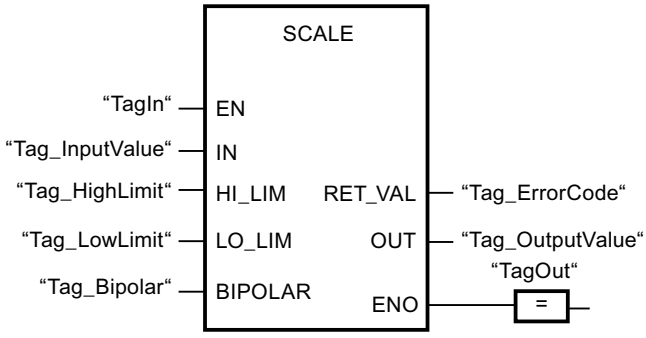

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

UNSCALE: Unscale

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at parameter IN is not within the limits defined by HI_LIM and LO_LIM, an error is output and the result is set to the nearest limit.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Input	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit

Parameters	Declaration	Data type	Memory area	Description
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

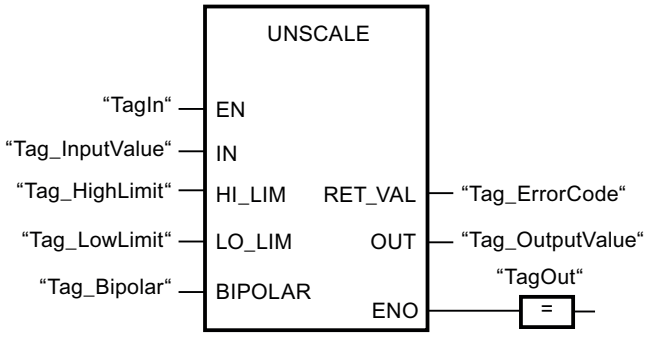

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1

Parameters	Operand	Value
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

Program control operations

JMP: Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil can occur in a network.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:

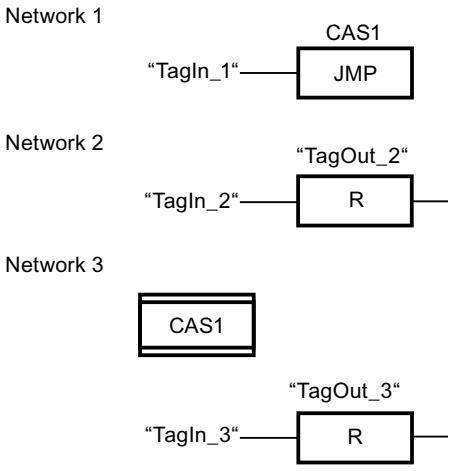

If the "TagIn_1" operand has signal state "1", the instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

JMPN: Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil can occur in a network.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of the logic operation RLO at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:

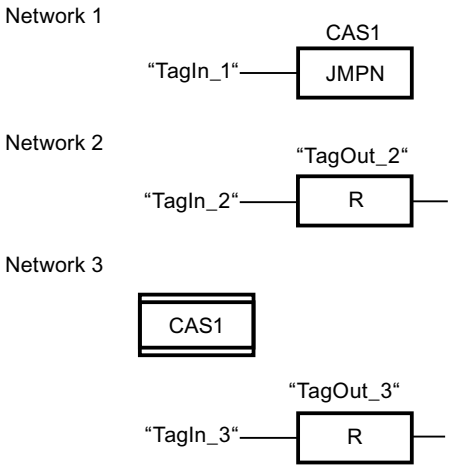

If the "TagIn_1" operand has signal state "0", the instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

LABEL: Jump label

Description

The jump label identifies a destination network in which the execution of the program can be resumed after the execution of a jump instruction.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

The following grammar rules must be followed for the jump label:

- Letters (a - z, A - Z)
- A combination of letters and numbers. Check that the order is correct, i.e. first the letters, then the numbers (a - z, A - Z, 0 - 9).
- Special characters or a combination of letters and numbers in reverse order, i.e. first the numbers and then the letters (0-9, a - z, A - Z), must not be used.

Example

The following example shows how the instruction works:

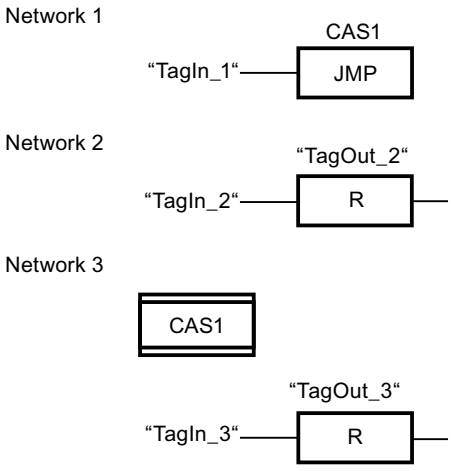

If the "TagIn_1" operand has signal state "1", the instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has signal state "1", the "TagOut_3" output is set.

See also

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

RET: Return

Description

You use the "Return" instruction to end the execution of a program block.

- If the result of logic operation (RLO) at the input of the "Return" instruction is "1", the program execution is ended in the currently called program block and continued after the call function in the calling program block (for example, in the calling organization block). The signal state of the enable output ENO of the call function is determined by the RLO. The RLO is shown via the instruction as a fixed parameter value.
- If the result of logic operation (RLO) at the input of the instruction is "0", then the instruction will not be executed. The program execution is continued in the next network of the called program block.

If the instruction is programmed in the last network of the called block, the enable output ENO of the call function is set to the current signal state of the RLO.

If an organization block is ended with the instruction, the CPU goes on working in the system program.

Example

The following example shows how the instruction works:

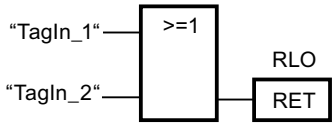

If one of the operands "TagIn_1" or "TagIn_2" have signal state "1", the instruction is executed. Program execution is ended in the called program block and continues in the calling program block. The enable output ENO of the call function is reset to signal state "1".

See also

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

OPN: Open global data block

Description

You can use the "Open global data block" instruction to open a global data block (DB). The number of the data block is transferred to the DB register. Subsequent DB commands access the relevant blocks depending on the content of the registers.

Parameters

The following table shows the parameters of the "Open global data block" instruction:

Parameters	Declaration	Data type	Memory area	Description
<data block>	Input	DB_ANY	-	Data block to be opened

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Network 1

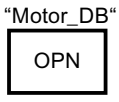

Network 2

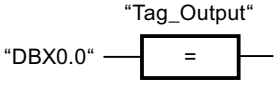

The "Motor_DB" data block is called in network 1. The number of the data block is transferred to the DB register. The "DBX0.0" operand is queried in network 2. The signal state of the "DBX0.0" operand is assigned to the "Tag_Output" operand.

See also

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

OPNI: Open instance data block

Description

You can use the "Open instance data block" instruction to open a data block (DB). The number of the data block is transferred to the DI register. Subsequent DI commands access the relevant blocks depending on the content of the registers.

Parameters

The following table shows the parameters of the "Open instance data block" instruction:

Parameters	Declaration	Data type	Memory area	Description
<data block>	Input	DB_ANY	-	Data block to be opened

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Network 1

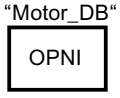

Network 2

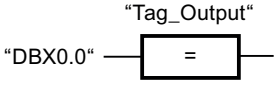

The "Motor_DB" data block is called in network 1. The number of the data block is transferred to the DI register. The "DBX0.0" operand is queried in network 2. The signal state of the "DBX0.0" operand is assigned to the "Tag_Output" operand.

See also

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Runtime control

COMPRESS: Compress CPU memory (S7-400)

Description

The "Compress CPU memory" instruction is used to compress the working memory of the CPU or the RAM load memory. During this compression the gaps caused by deleting or loading blocks are filled by pushing together the existing blocks.

The compression operation is started by calling the "Compress CPU memory" instruction. The compression operation is divided into several program cycles. The compression operation is monitored by the values of the individual parameters. If value "1" is on the BUSY parameter of the instruction, the compression operation is still being executed. The value "1" on the DONE parameter reports that the compression operation has been completed. Error information can be queried on the RET_VAL parameter.

If an externally started compression operation has already been executed, the calling of the "Compress CPU memory" instruction will cause an error to be displayed.

Parameters

The following table shows the parameters of the "Compress CPU memory" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Information whether compression operation is active.
DONE	Output	BOOL	I, Q, M, D, L	Information whether compression operation was completed.

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error The status of the compression operation can be queried on the BUSY and DONE parameters.
8091	An externally started compression operation is already being executed.
8092	The instruction cannot be executed, because one of the following conditions applies: • The "Delete block" function is active. • A test and commissioning function affects a block (for example, status). • The "Copy blocks" function is enabled by an external trigger. • The H-CPU is running a linking or update routine.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

RE_TRIGR: Restart cycle monitoring time

Description

The "Restart cycle monitoring time" instruction is used to restart the cycle monitoring time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called in all blocks regardless of the priority .

If the instruction is called in a block with a higher priority, for example a hardware interrupt or a diagnostics interrupt, the instruction is not executed and the enable output ENO is set to signal state "0".

The "Restart cycle monitoring time" instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Parameters

The instruction "Restart cycle monitoring time" has no parameters.

See also

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

STP: Exit program

Description

The instruction "Exit program" instruction is used to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

When the (RLO) at the input of the instruction is "1", the CPU changes to STOP mode and the execution of the program is terminated. The signal state at the output of the instruction is not evaluated.

If the RLO at the input of the instruction is "0", then the instruction will not be executed.

Parameters

The instruction "Exit program" has no parameters.

See also

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

WAIT: Configure time delay

Description

The "Configure time delay" instruction pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to 32767 microseconds (µs). The smallest possible delay time depends on the CPU and corresponds to the execution time of the instruction.

The execution of the instruction can be interrupted by higher priority events and does not return any error information.

Applies to S7-300-CPUs:

For the S7-300-CPUs (except S7-318), the delay time programmed by the instruction is a minimum time. It is extended by the execution time of the nested priority classes and by system utilizations.

Note

Negative delay time

If you specify a negative delay time at parameter WT, the enable output ENO, or the RLO and the BR bit, will return the signal state FALSE. A negative delay time does not affect the CPU. The following instructions linked to enable output ENO are not executed in LAD or FBD.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

PROTECT: Change protection level

Description

The instruction "Change protection level" is used to change the configured protection level "1" of the CPU.

The configured protection level "1" with selected option "Removable with password", means that you have to configure a password. This password enables you to write-access during operation the CPU which was read-only protected with the instruction "Change Protection Level" (e. g. Load).

You specify the protection level on the MODE parameter of the instruction. The current protection level of your CPU is displayed in the inspector window under "Properties > Protection".

The following table shows the meaning of the protection levels.

Protection level	Meaning
1	All programming device functions are permitted. Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
2	• The program and the configuration of the CPU cannot be changed. The program in the CPU can be read. • The functions for process control, process monitoring, and process communication are permitted. • All informational functions are permitted. • If you know the valid password you can cancel the set write protection. • Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
3	• You cannot cancel the set read and write protection, even if you know the valid password. • If a valid connection exists at the time when the instruction "Change protection level" is called, the call of the instruction does not affect this connection.

If you have configured protection level "2" or "3" for your CPU, calling the "Change protection level" instruction has no effect.

You can use the SSL parts list with the SSL ID W#16#0232 and index W#16#0004 to check whether protection level "2" or "3" is still in effect after the execution of the "Change protection level" instruction.

Note

You can only use the "Change protection level" instruction to set a protection level that is equal to or higher than the protection level you have configured with STEP 7 "Configure hardware. You have, for example, configured protection level 3 and then called the instruction "Change protection level" with the MODE parameter = 12. This blocks the password authentication. If the instruction is called again with the parameter MODE = 0 or 1, protection level 3 is retained and the read and write protection canceled by entry of the valid password.

Effects of the operating state transitions on the protection level

The following table shows what effects the operating state transitions have on the protection level set with the "Change protection level" instruction.

Action	Effects on the protection level
• Operating mode switch to "STOP" • Unbuffered POWER ON • Memory reset with operating mode switch • Configuration and/or programming changes in "STOP" mode of the S7-400 CPU	The protection level is set to "1". All programming device functions are permitted.
• Cold restart • Warm restart	The operating system sets the protection level to "1". All programming device functions are permitted. If required, you can set the protection level in your program to "2" or "3" by calling the "Change protection level" instruction.
Buffered POWER ON	Remains unchanged
Operating mode transition RUN/STARTUP/HOLD -> STOP (by calling the instruction "Exit program" (STP), due to a program error without corresponding error OB or by an operator action).	Remains unchanged
S7-400: Restart by user action or automatically following POWER ON	Remains unchanged

Parameters

The following table shows the parameters of the "Change protection level" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MODE	Input	WORD	I, Q, M, D, L or constant	Protection level Possible values: • W#16#0000: Setting protection level "1" • W#16#0001: Setting protection level "2" with password authentication • W#16#000C: Setting protection level "3" with password authentication Constants can also be specified in the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs during processing, information on the error is output.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error
8090	Invalid value at parameter MODE.
80C3	The required operating resources are currently occupied.
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

CiR: Control CiR process (S7-400)

Description

The "Control CiR process" instruction is used to influence the CiR process in the user program as follows:

- Disable CiR process In this case, the download of a modified configuration from the programming device to the CPU is always rejected. The disable is maintained until you cancel it again with the "Control CiR process" instruction.
- Disable the CiR process conditionally by specifying a high limit for the CiR synchronization time on the FRZ_TIME parameter. In this case, the download of a modified configuration from the programming device to the CPU is only accepted if the CiR synchronization time calculated in the CPU does not exceed the specified high limit.
- Enable CiR process The high limit of the CiR synchronization time is set to the default value 1 000 ms.
- Determine whether the CiR process is enabled. If the CiR process is enabled or conditionally enabled, you will get the current high limit for the CiR synchronization time in the A_FT parameter of the instruction.

Note

During the CiR synchronization period the outputs are frozen and the inputs are not evaluated.

Parameters

The following table shows the parameters of the "Control CiR process" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Valid values on the MODE parameter are: • 0: Information function • 1: Enable CiR process • 2: Disable CiR process • 3: Disable CiR conditionally. You specify the high limit of the CiR synchronization time in FRZ_TIME.
FRZ_TIME	Input	TIME	I, Q, M, D, L or constant	High limit of the CiR synchronization time in milliseconds. Permitted range of values: 60 to 2500 ms (default: 1000 ms) The FRZ_TIME parameter is only relevant when the value "3" is set on the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	Error information When the MODE parameter is "0", the RET_VAL parameter supplies information on the enabling of the CiR process.
A_FT	Output	TIME	I, Q, M, D, L	Currently valid high limit of the CiR synchronization time

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error This error code can be output only if the value "1", "2", or "3" is set at the MODE parameter.
0001	CiR is enabled. This error code can be output only if the value "0" is set on the MODE parameter.
0002	The CiR process is disabled. This error code can be output only if the value "0" is set on the MODE parameter.
0003	CiR is disabled conditionally. This error code can be output only if the value "0" is set on the MODE parameter.

Error code* (W#16#...)	Explanation
8001	The CPU is not ready for a CiR operation. You are using an H CPU in an H system (stand-alone), or you are working with a standard CPU in multicomputing mode.
8002	The value at the MODE parameter is invalid.
8003	The value at the FRZ_TIME parameter is invalid.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The "Control CiR process" instruction can be used, for example, to ensure that no CiR process is initiated during periods in which maximum CPU performance is required for the execution of the process.

To use the "Control CiR process" for this, the instruction must be called before the start of the period with increased process activity and executed with the value "2" (disable CiR process) on the MODE parameter. The CiR process is then disabled.

To enable the CiR process again after the completion of the period with increased process activity, you have to again call the "Control CiR process" instruction and execute it with the value "1" on the MODE parameter. Alternatively, you can also set the value "3" on the MODE parameter to disable the CiR process conditionally.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

Word logic operations

AND: AND logic operation

Description

You can use the instruction "AND logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by AND logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by AND logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

The result bit has the signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "AND logic operation":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

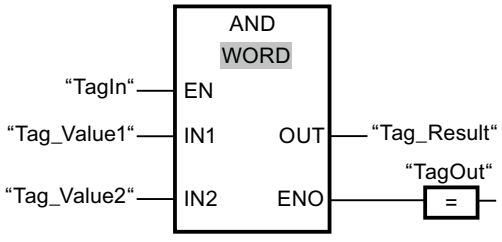

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	00000000 00000101

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is linked by AND to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bit for word logic operations (Page 182)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

OR: OR logic operation

Description

You can use the instruction "OR logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

The result bit has the signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "OR logic operation":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

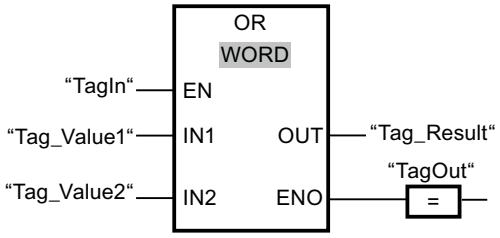

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	01010101 01011111

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is linked by OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bit for word logic operations (Page 182)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

XOR: EXCLUSIVE OR logic operation

Description

You can use the instruction "EXCLUSIVE OR logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by EXCLUSIVE OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by EXCLUSIVE OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

The result bit has the signal state "1" when one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

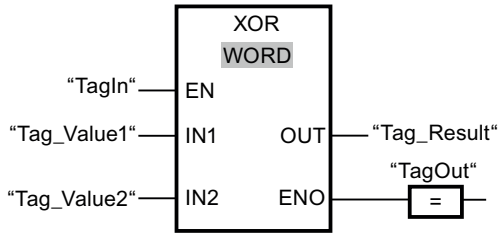

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	01010101 01011010

If the "TagIn" operand has signal state "1", the instruction is executed. The value of operand "Tag_Value1" is linked by EXCLUSIVE OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bit for word logic operations (Page 182)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

INVERT: Create ones complement

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits at input IN. When the instruction is processed, the value at input IN is linked to EXCLUSIVE OR by a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit number). This inverts the signal state of the individual bits that are then stored at output OUT.

Parameters

The following table shows the parameters of the instruction "Create ones complement":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers	I, Q, M, D, L, P	Ones complement of the value at input IN

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

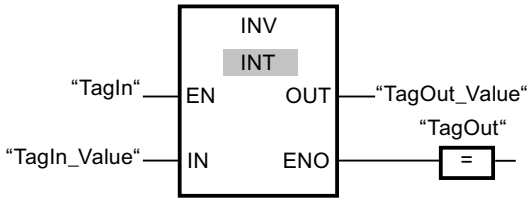

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	W#16#000F
OUT	TagOut_Value	W#16#FFF0

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction inverts the signal state of the individual bits at input "TagIn_Value" and writes the result to output "TagOut_Value". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. If the value at input IN is greater than 31, a modulo 32 instruction is executed.

Parameters

The following table shows the parameters of the "Decode" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set.
OUT	Output	DWORD	I, Q, M, D, L, P	Output value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

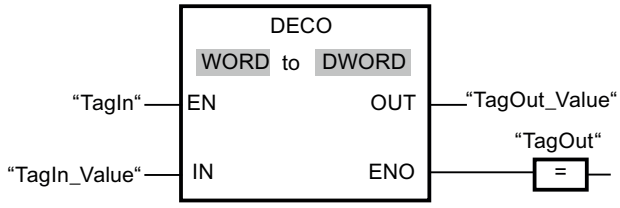

The following figure shows how the instruction works using concrete operand values:

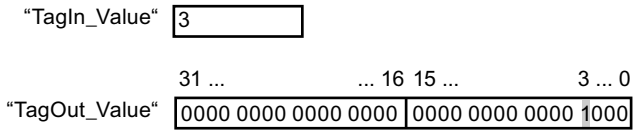

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction reads bit number "3" from the value of the operand "TagIn_Value" at the input and sets the third bit to the value of the operand "TagOut_Value" at the output.

If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

ENCO: Encode

Description

The instruction "Encode" is used to read the bit number of the lowest bit in the input value and output it to the output OUT.

The "Encode" instruction selects the least significant bit of the value in the IN input and writes its bit number to the tag in the OUT output.

Parameters

The following table shows the parameters of the instruction "Encode":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

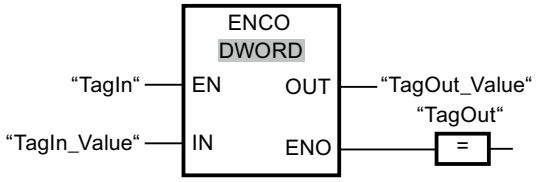

The following figure shows how the instruction works using concrete operand values:

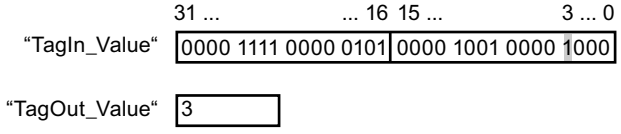

If the "TagIn" operand has signal state "1", the instruction is executed. The instruction selects the least significant bit at the "TagIn_Value" input and writes bit position "3" to the tag in the "TagOut_Value" output.

If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

SEL: Select

Description

Depending on a switch (input G), the "Select" instruction selects one of the inputs IN0 or IN1 and moves its content to the output OUT. When the input G has the signal state "0", the value at the input IN0 is moved. When the input G has the signal state "1", the value at the input IN1 is moved to the output OUT.

The instruction can only be executed if the enable input EN has the signal state "1" and the tags of all parameters are of the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
G	Input	BOOL	I, Q, M, D, L, T, C	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

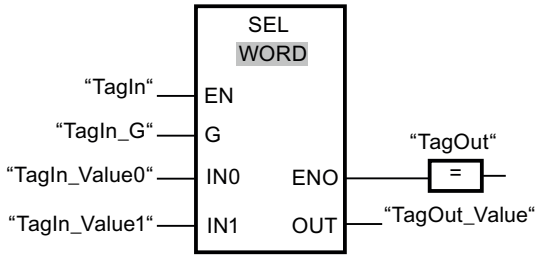

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If the "TagIn" operand has signal state "1", the instruction is executed. Based on the signal state at the input "TagIn_G", the value at input "TagIn_Value0" or "TagIn_Value1" is selected and copied to output "TagOut_Value". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

Shift and Rotate

SHR: Shift right

Description

You can use the "Shift right" instruction to shift the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure shows how the content of an operand of the INT data type is shifted by four bit positions to the right:

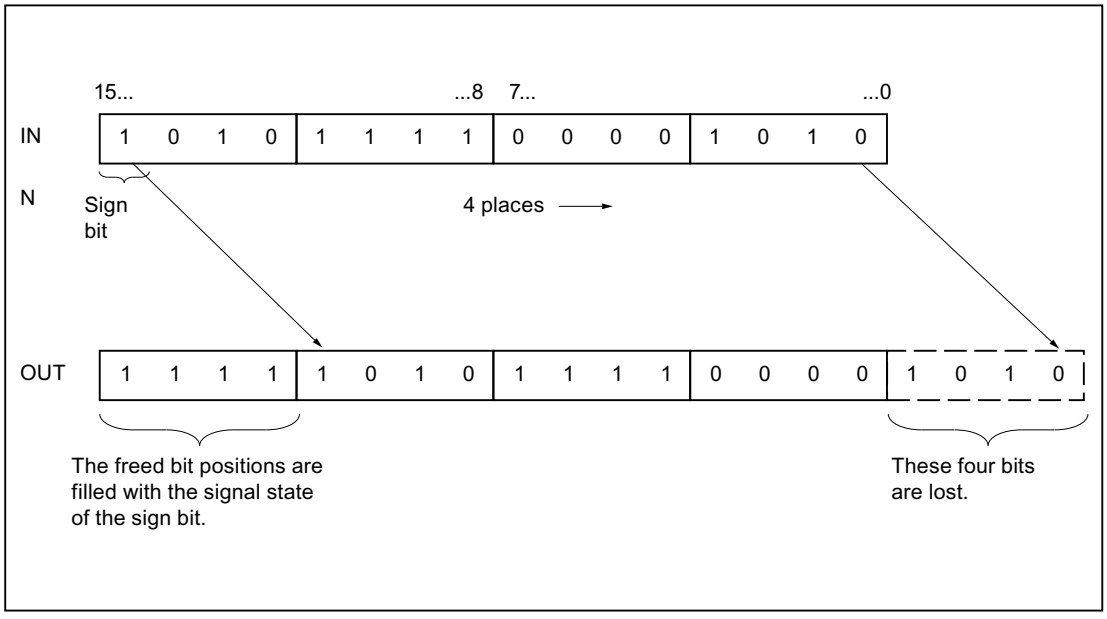

Parameters

The following table shows the parameters of the instruction "Shift right":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L, P or constant	Value to be shifted
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is shifted
OUT	Output	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

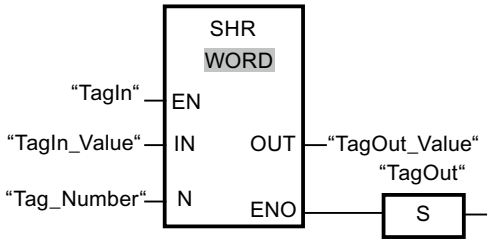

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If the "TagIn" operand has signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is shifted three bit positions to the right. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

SHL: Shift left

Description

You can use the "Shift left" instruction to shift the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of bit positions, the operand value at input IN is shifted to the left by the available number of bit positions.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure shows how the content of an operand of the WORD data type is shifted by six bit positions to the left:

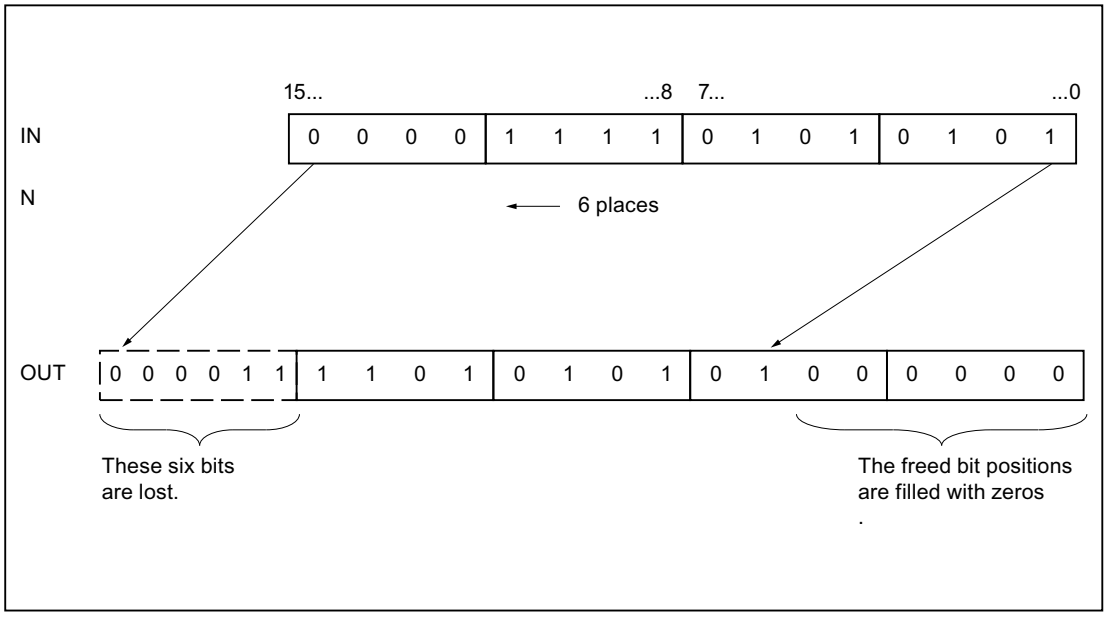

Parameters

The following table shows the parameters of the instruction "Shift left":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L, P or constant	Value to be shifted
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is shifted
OUT	Output	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

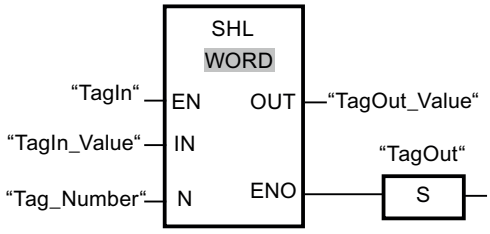

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If the "TagIn" operand has signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is shifted four bit positions to the left. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

ROR: Rotate right

Description

You can use the "Rotate right" instruction to rotate the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. The parameter N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value of the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:

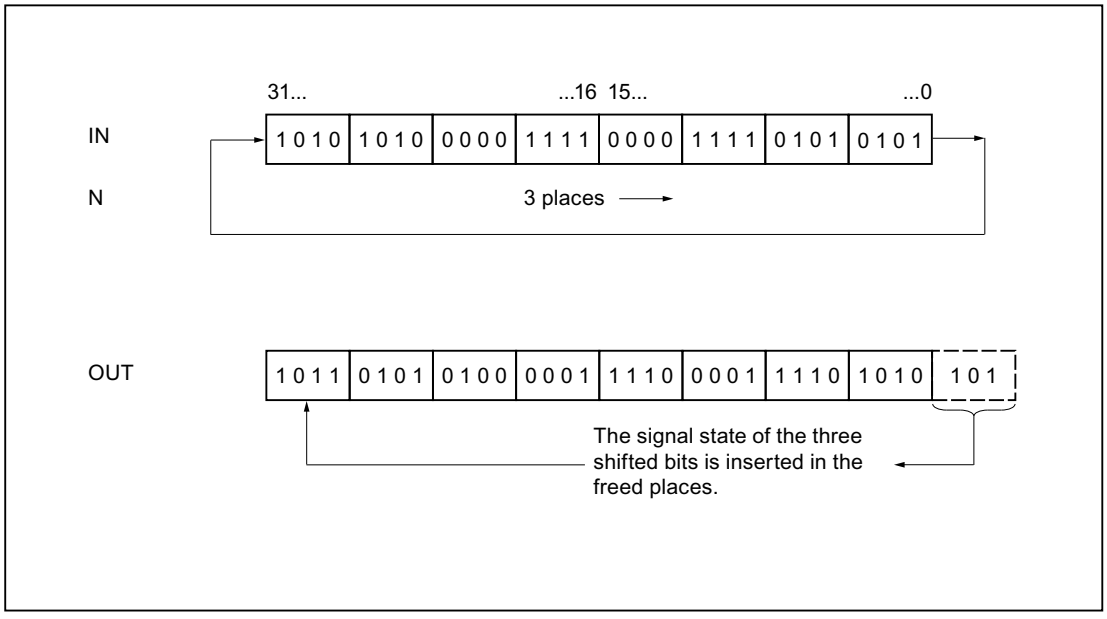

Parameters

The following table shows the parameters of the instruction "Rotate right":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P or constant	Value to be rotated
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is rotated
OUT	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

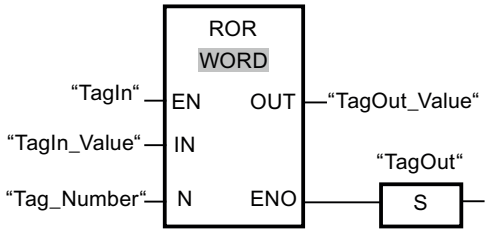

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	1010 1010 0000 1111 0000 1111 0101 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1101 0101 0000 0111 1000 0111 1010

If the "TagIn" operand has signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the right. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

ROL: Rotate left

Description

You can use the "Rotate left" instruction to rotate the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. The parameter N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value of the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:

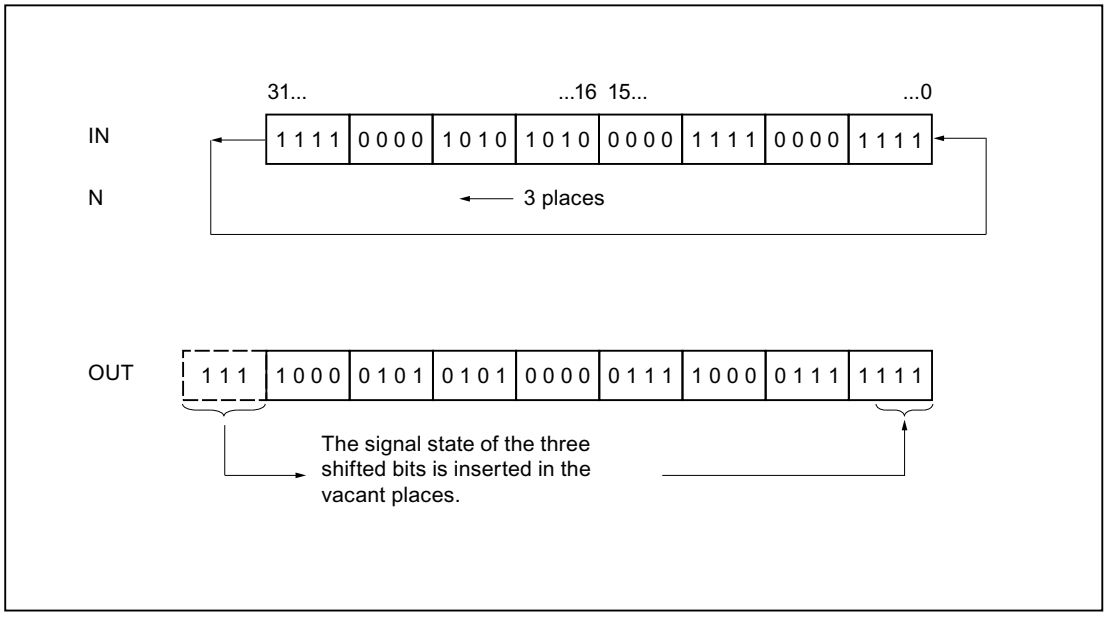

Parameters

The following table shows the parameters of the instruction "Rotate left":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P or constant	Value to be rotated.
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is rotated.
OUT	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

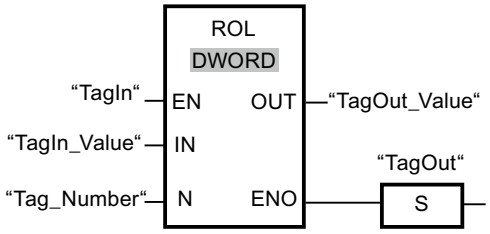

The following table shows how the instruction works using concrete operand values:

Parameters	Operand	Value
IN	TagIn_Value	1111 0000 1010 1010 0000 1111 0000 1111
N	Tag_Number	5
OUT	TagOut_Value	0001 0101 0100 0001 1110 0001 1111 1110

If the "TagIn" has the signal state "1", the instruction is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the left. The result is output at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Selecting the data type of an FBD element (Page 7201)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

Other instructions

---| |---: Get status bit

Description

With the "Get status bit" instruction you can query the signal state of the following status bits:

- BR: Binary result
- OV: Overflow
- OS: Latching overflow
- CC 0 and CC 1: Display bits

You determine the bits to be queried by selecting the appropriate symbols from the drop-down list of the instruction. The following table shows the options for querying the status bits:

Symbol	Query	Explanation
==0	CC 0 = 0 and CC 1 = 0	The instruction queries whether the result of a mathematical function is equal to 0.
<>0	CC 0 = 0 and CC 1 = 1 or CC 0 = 0 and CC 1 = 0	The instruction queries whether the result of a mathematical function is not equal to 0.
>=0	CC 0 = 0	The instruction queries whether the result of a mathematical function is greater than or equal to 0.
<=0	CC 1 = 0	The instruction queries whether the result of a mathematical function is less than or equal to 0.
>0	CC 0 = 0 and CC 1 = 1	The instruction queries whether the result of a mathematical function is greater than 0.
<0	CC 0 = 1 and CC 1 = 0	The instruction queries whether the result of a mathematical function is less than 0.
OV	CC 0 = 1 and CC 1 = 1	The instruction queries whether the result of a mathematical function is invalid. An invalid result may be caused by an invalid floating-point number or a division by zero.
OV	OV = 1	The instruction queries whether a number range overflow has occurred in an instruction.
OS	OS = 1	The instruction queries whether a saving number range overflow has occurred in an instruction.
BR	BR = 1	The instruction queries the signal state of the BR status bits.

If the query is met, the instruction returns signal state "1". If the query is not met, the instruction returns signal state "0".

You can program a query for status bits "0" using the "Invert RLO" instruction.

Example

The following example shows how the instruction works:

Network 1

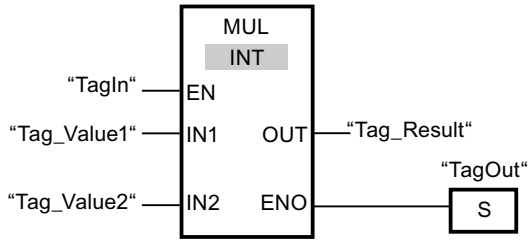

Network 2

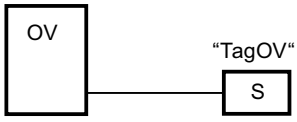

If the operand "TagIn" has the signal state "1", the instruction "Multiply" (MUL) is executed. If an overflow occurs during the execution of the "Multiply" instruction, the OV status bit is set. After the query of the OV status bit, the "Set output" (S) instruction is executed in network 2 and the "TagOV" operand is set.

See also

Basic information on the status word (Page 173)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Setting status bits (Page 175)

CALL: Call block

Description

The "Call block" instruction calls a function (FC) without parameter. The call is conditional or unconditional, depending on the preceding logic operation.

In the code section of a function (FC), you cannot specify any parameter of the data type BLOCK_FC as the operand for a conditional call. You can, however, specify a parameter of the type BLOCK_FC as the operand in a function block (FB).

A conditional call is then only executed if the result of logic operation (RLO) is "1". If a conditional call is not executed the RLO is "0" after the call instruction. If the instruction is executed, it works as follows:

- It stores the return address of the calling block.
- It stores the two data block registers (data block and instance data block).
- It updates the local data for the called function.
- It shifts the MA bit (active MCR bit) to the block stack (B stack).

Parameters

The following table shows the parameters of the "Call block" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Block>	Output	BLOCK_FC	-	Called function

Example

The following example shows how the instruction works:

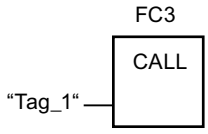

The function FC3 is called if the operand "Tag_1" has the signal state "1".

See also

Master control relay (Page 191)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

SAVE: Save RLO in BR bit

Description

Use the instruction "Save RLO in BR bit" to save the result of logic operation (RLO) in the binary result (BR). During processing, the instruction transfers the signal state of the current result of logic operation to the status bit BR. The instruction operates independent of the conditions and does not affect any other status bits.

Note

After application of the instruction "Save RLO in BR bit", the BR bit can be modified again by the following instructions in the same or lower-level blocks.

Use the instruction "Save RLO in BR bit" to check the execution status of a block. If you program the "Save RLO in BR bit" instruction at the end of a block, for example, the BR bit is set to the signal state of the current result of logic operation of the block. If the block is executed without errors, the BR bit and the ENO enable output in the calling block are set to signal state "1". Signal state "0" is assigned to the BR bit if an error occurs during processing of the block. You can program a troubleshooting routine by querying the BR bit.

Example

The following example shows how the instruction works:

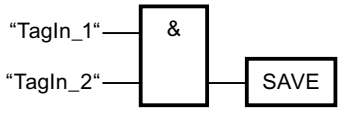

The BR is set to signal state "1" when the "TagIn_1" and "TagIn_2" operands return signal state "1".

See also

Basic information on the status word (Page 173)

Example of the use of the EN/ENO mechanism in FBD (Page 150)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

MCR<: Open MCR ranges

Description

The "Open MCR ranges" instruction saves the result of logic operation (RLO) in the MCR stack and enables MCR dependency within an MCR range (MCR = Master Control Relay).

The MCR stack can contain up to a maximum of 8 entries and functions like a LIFO buffer (Last In, First Out). When the stack is full, the instruction generates an error message (MCRF).

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Use the instructions "Enable MCR range" (MCRA) and "Disable MCR range" (MCRD) to specify the start and end of an MCR range. Within the defined MCR range you can enable and disable MCR dependency for one or more program sections.

MCR dependency is enabled when the RLO is "1" before processing of the instruction. If the RLO is "0" before the instruction, MCR dependency is disabled. The end of the MCR dependent program section is programmed using the instruction "Close MCR ranges". The program

section between the two instructions is affected by the MCR. Only the following MCR-dependent instructions are affected by this:

- Assignment
- Set output (S)
- Reset output (R)
- Reset/set flip-flop (RS)
- Set/reset flip-flop (SR)
- Move value (MOVE)

You can also nest the enabling and disabling of MCR dependency by programming the instruction "Open MCR ranges" in an MCR dependent program section. You can nest the instruction up to 8 levels deep. The MCR dependency of the higher-level program section hereby controls the MCR dependencies in all lower-level program sections. If MCR dependency is disabled in the higher-level program section, this MCR dependency cannot be switched on in any of the lower-level MCR dependent program sections.

If you call a block within an MCR range, MCR dependency is disabled in the called block. When the block is exited, MCR dependency is set so that it is the same as before the block call.

The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Example

The following example shows how the instruction works:

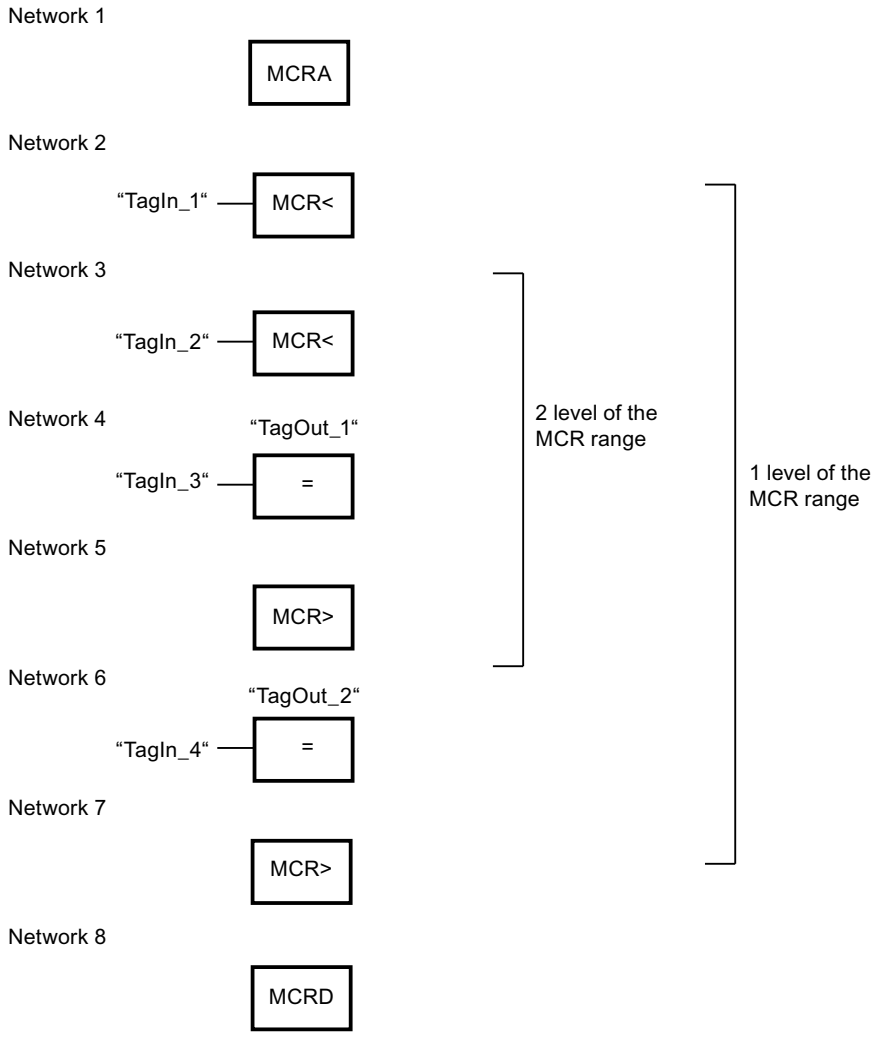

The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns signal state "1", MCR dependency is enabled in the first level of the MCR range. In this case, the signal state of the "TagIn_4" operand is assigned to the "TagOut_2" operand (network 6). If the "TagIn_1" operand returns signal state "0", MCR dependency is disabled and the "TagOut_2" operand is reset to "0". MCR dependency of the MCR range is disabled in network 7.

In network 3, MCR dependency is enabled in the second level of the MCR range, when the "TagIn_2" operand has signal state "1" and the first level of the MCR range is enabled. In this case, the signal state of the "TagIn_3" operand is assigned to the "TagOut_1" operand. If the "TagIn_2" operand returns signal state "0", MCR dependency is disabled and the "TagOut_1" operand is reset to "0". MCR dependency in the second level of the MCR range is disabled in network 5. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

MCR>: Close MCR ranges (Page 4554)

MCRA: Enable MCR range (Page 4556)

MCRD: Disable MCR range (Page 4557)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

MCR>: Close MCR ranges

Description

The instruction "Close MCR ranges" closes an MCR dependent program section (MCR = Master Control Relay). In this case the instruction deletes the result of logic operation (RLO) from the last entry in the MCR stack.

The MCR stack can contain up to a maximum of 8 entries and functions like a LIFO buffer (Last In, First Out). If the stack is already empty, the instruction generates an error message (MCRF).

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Use the instruction "Open MCR ranges" to program MCR dependency. The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Example

The following example shows how the instruction works:

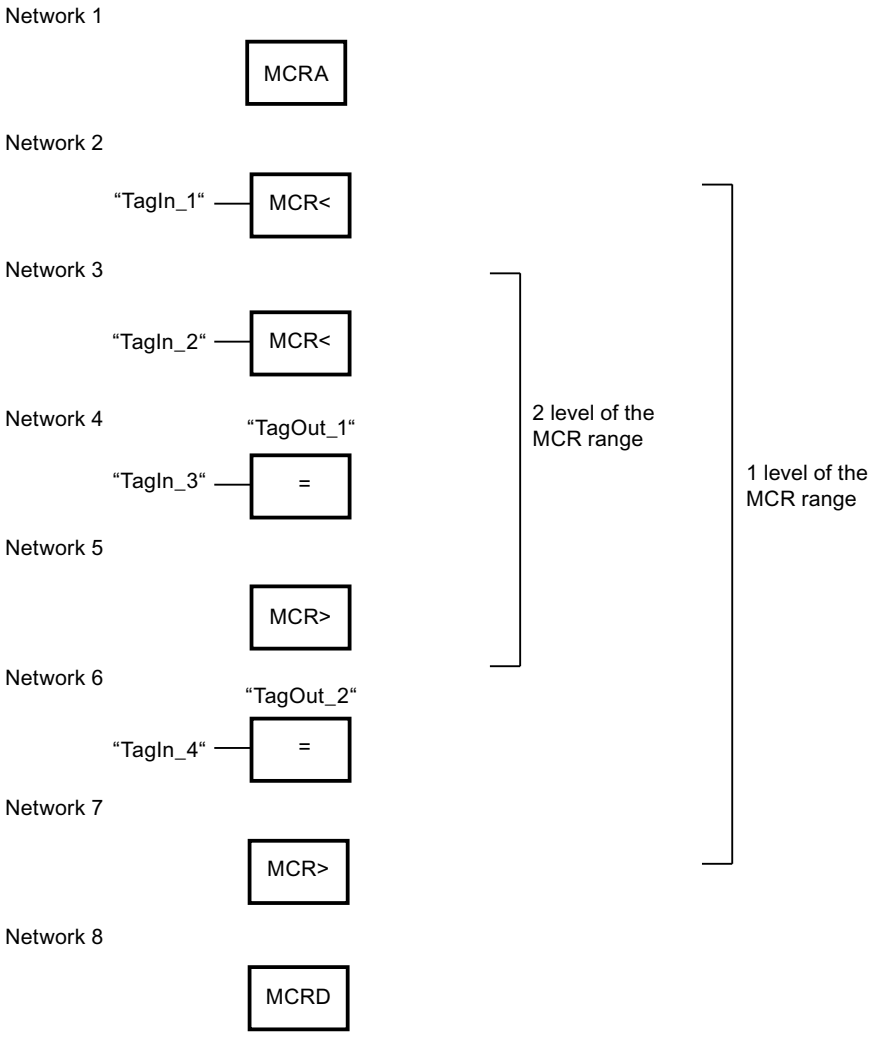

The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns signal state "1", MCR dependency is enabled in the first level of the MCR range. In this case, the signal state of the "TagIn_4" operand is assigned to the "TagOut_2" operand (network 6). If the "TagIn_1" operand returns signal state "0", MCR dependency is disabled and the "TagOut_2" operand is reset to "0". MCR dependency of the MCR range is disabled in network 7.

In network 3, the MCR-dependency is enabled in the second level of the MCR range, when the "TagIn_2" operand has signal state "1". In this case, the signal state of the "TagIn_3" operand is assigned to the "TagOut_1" operand. If the "TagIn_2" operand returns signal state "0", the MCR dependency is disabled and the "TagOut_1" operand is reset to "0" regardless of the signal state of the "TagIn_3" operand. MCR dependency in the second level of the MCR range is disabled in network 5. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

MCR<: Open MCR ranges (Page 4551)

MCRA: Enable MCR range (Page 4556)

MCRD: Disable MCR range (Page 4557)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

MCRA: Enable MCR range

Description

Use the instruction "Enable MCR range" to specify the start of an MCR range (MCR = Master Control Relay). Within this range, you can switch MCR dependency on and off using the following instructions:

- MCR<: Open MCR ranges
- MCR>: Close MCR ranges

You specify the end of the MCR range with the instruction "Disable MCR range" (MCRD). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

If you call a block within an MCR range, MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Example

The following example shows how the instruction works:

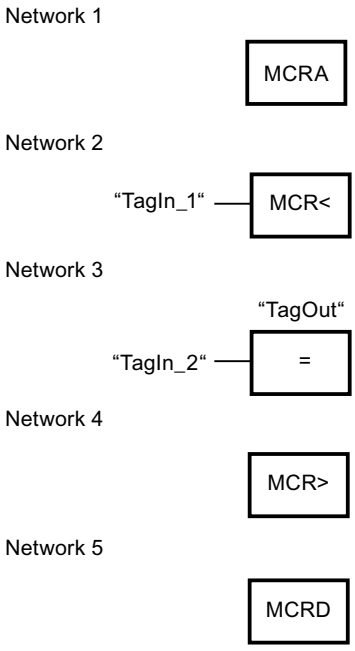

The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns the signal state "1", the MCR dependency is enabled. In this case, the signal state of the "TagIn_2" operand is assigned to the "TagOut" operand (network 3). If the "TagIn_1" operand returns signal state "0", the MCR dependency is disabled and the "TagOut" operand is reset to "0" regardless of the signal state of the "TagIn_2" operand. MCR dependency of the MCR range is disabled in network 4. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

MCR<: Open MCR ranges (Page 4551)

MCR>: Close MCR ranges (Page 4554)

MCRD: Disable MCR range (Page 4557)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

MCRD: Disable MCR range

Description

Use the instruction "Disable MCR range" to specify the start of an MCR range (MCR = Master Control Relay).

You specify the start of the MCR range with the instruction "Enable MCR range" (MCRA). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Within an MCR range you can program the MCR dependency with the help of the following instructions:

- MCR<: Open MCR ranges
- MCR>: Close MCR ranges

If you call a block within an MCR range, MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Example

The following example shows how the instruction works:

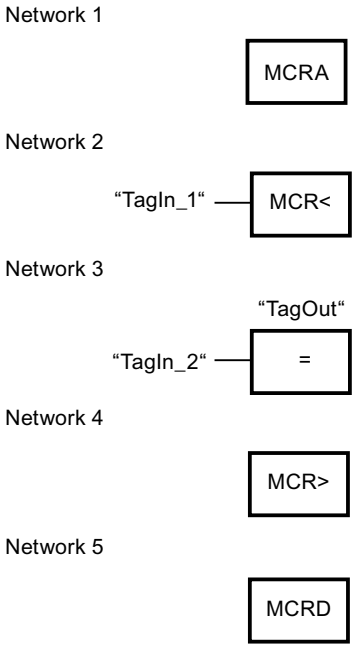

The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns the signal state "1", the MCR dependency is enabled. In this case, the signal state of the "TagIn_2" operand is assigned to the "TagOut" operand (network 3). If the "TagIn_1" operand returns signal state "0", MCR dependency is disabled and the "TagOut" operand is reset to "0". MCR dependency of the MCR range is disabled in network 4. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

MCR<: Open MCR ranges (Page 4551)

MCR>: Close MCR ranges (Page 4554)

MCRA: Enable MCR range (Page 4556)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

SET: Set bit array

Description

The "Set bit array" instruction sets the signal state of the bit in a specified area to "1". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. At the N parameter, you specify the number of bits to be set in the specified area. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Note

Parameter N

You can also interconnect an element of a data block at parameter N.

Parameters

The following table shows the parameters of the "Set bit array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are set For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be set

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

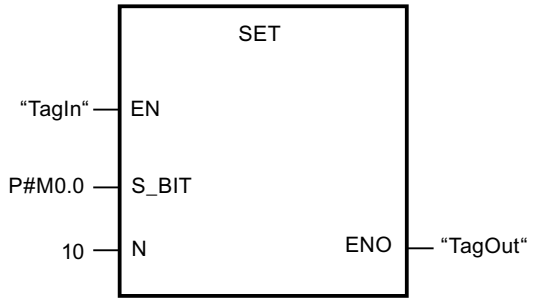

If the "TagIn" operand has the signal state "1", the instruction sets 10 bits in the area from M0.0 to M1.2.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Master control relay (Page 191)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

SETP: Set a bit array in the I/O area (S7-400)

Description

The instruction "Set a bit array in the I/O area" sets the bits of an I/O area to the signal state "1". At the same time the corresponding bits in the process image of the outputs are also set parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be set. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output on the RET_VAL parameter.

The I/O area is set byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output table.

  WARNING

  Unintended behavior

  The setting of the bits at the value of the relevant bit in the process image output table can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are set to the signal state "1".

The instruction "Set a bit array in the I/O area" must be called in an MCR area.

Calling the "Set a bit array in the I/O area" instruction has no effect and the ENO enable output is set to "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The N parameter has the value "0".
- The MCR bit has the signal state "1".

Parameters

The following table shows the parameters of the "Set a bit array in the I/O area" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be set
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be set

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

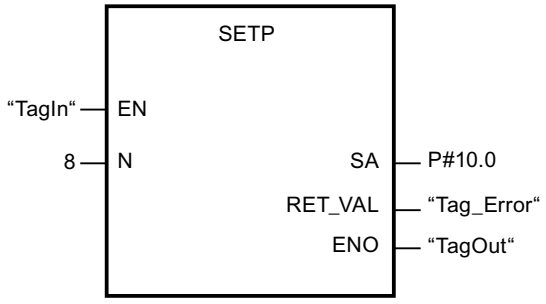

If the "TagIn" operand returns the signal state "1", the instruction sets 8 bits in the I/O area based on outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

SETI: Set byte array

Description

The "Set byte array" instruction sets the signal state of the bits in a specified area to "1". You define the start point of the array on the S_BYTE. You define the size of the array by specifying the number of bits to be set on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number 8 (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer on the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output table (Q) is reset to "0".

Calling the "Set byte array" instruction has no effect and the ENO enable output is set to "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Parameters

The following table shows the parameters of the "Set byte array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BYTE	Input	POINTER	P or constant	Pointer to the first byte of the array whose bits are to be set For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array to be set The array is specified by the number of bits as a multiple of 8 (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

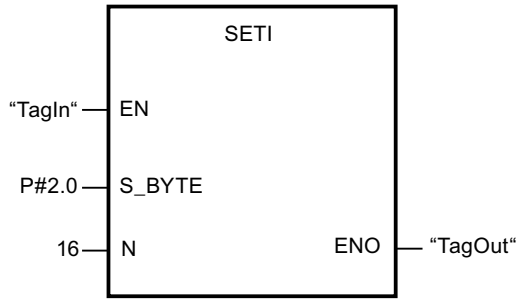

If the "TagIn" operand returns the signal state "1", the instruction sets 2 bytes (16 bits) in the range from P2.0 to P3.7

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Master control relay (Page 191)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

RESET: Reset bit array

Description

The "Reset bit array" instruction sets the signal state of the bit in a specified area to "0". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. At the N parameter, you specify the number of bits to be reset in the specified range. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Parameters

The following table shows the parameters of the "Reset bit array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are to be reset For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be reset

Example

The following example shows how the instruction works:

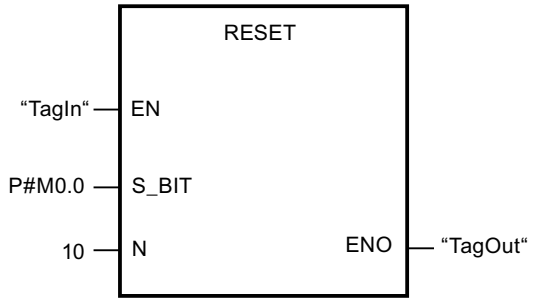

If the "TagIn" operand has the signal state "1", the instruction resets 10 bits in the area from M0.0 to M1.2.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Master control relay (Page 191)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

RESETP: Reset a bit array in the I/O area (S7-400)

Description

The instruction "Reset a bit array in the I/O area" resets the bits of an I/O area to the signal state "0". At the same time the corresponding bits in the process image of the outputs are also reset parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be reset. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output on the RET_VAL parameter.

The I/O area is reset byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output table.

  WARNING

  Unintended behavior

  The setting of the bits at the value of the relevant bit in the process image output table can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are reset to the signal state "0".

The instruction "Reset a bit array in the I/O area" must be called in an MCR range.

Calling the instruction "Reset a bit array in the I/O area" has no effect and enable output ENO is set to "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The N parameter has the value "0".
- The MCR bit has the signal state "1".

Parameters

The following table shows the parameters of the instruction "Reset a bit array in the I/O area":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be reset.

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

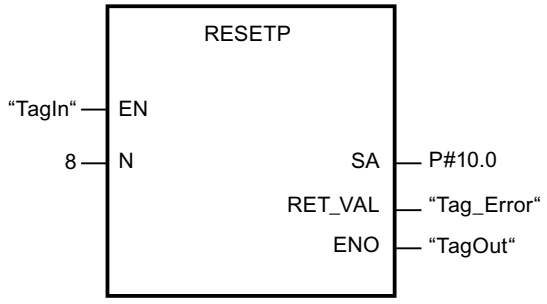

If the "TagIn" operand returns the signal state "1", the instruction sets 8 bits in the I/O area based on outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

RESETI: Reset byte array

Description

The "Reset byte array" instruction sets the signal state of the bits in a specified area to "0". You use the pointer on the S_BYTE parameter to define the starting point of the range. You define the size of the range by specifying the number of bits to be reset on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number 8 (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer on the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output table (Q) is reset to "0".

Calling the "Reset byte array" instruction has no effect and the ENO enable output is set to "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Parameters

The following table shows the parameters of the "Reset byte array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BYTE	Input	POINTER	P or constant	Pointer to the first byte of the range whose bits are to be reset For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array whose bits are to be reset. The array is specified by the number of bits as a multiple of 8 (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

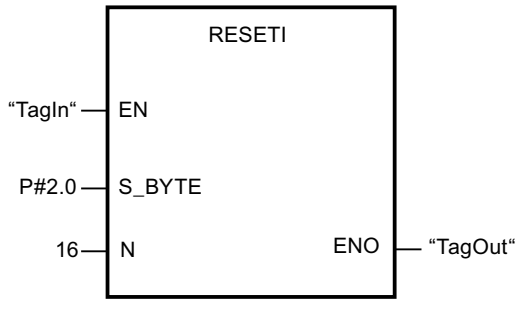

If the "TagIn" operand returns the signal state "1", the instruction sets 2 bytes (16 bits) in the range from P2.0 to P3.7

See also

Overview of the valid data types (Page 231)
Basics of the EN/ENO mechanism (Page 141)
Master control relay (Page 191)
Inserting FBD elements (Page 7198)
Editing FBD elements (Page 7215)
Inserting operands in FBD instructions (Page 7222)

REPL_VAL: Enter substitute value

Description

The "Enter substitute value" instruction is used to enter a substitute value in the accumulator 1 (ACCU 1) from a synchronous error organization block.

Note

You can only call the "Enter substitute value" instruction in a synchronous error OB (OB 121, OB 122).

You use the "Enter substitute value" instruction when you can no longer read any value from a module. The OB 122 is started after each access to this module. By calling the "Enter substitute value" instruction you can cause the substitute value to be loaded in accumulator 1. In this case the program will then be processed with this substitute value. You can take the required information for the selection of the substitute value from the local tags of the OB 122.

Parameters

The following table shows the parameters of the "Enter substitute value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
VAL	Input	DWORD	I, Q, M, D, L or constant	Substitute value
RET_VAL	Output	INT	I, Q, M, D, L	Error information

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Description
0000	No error
8080	The instruction was not called from a synchronous error OB.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on FBD (Page 7185)

DRUM: Implement sequencer

Description

You can use the "Implement sequencer" instruction to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK), you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask has signal state "1", the OUT_VAL value sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit at the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit at the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" indicates the reset condition.
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L	The signal state "1" causes the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.

Parameter	Declaration	Data type	Memory area	Description
DSP	Static	BYTE	I, Q, M, D, L, P	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P	Current numerical value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L	Preset count for each step [1 to 16]; where 1 clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Result

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

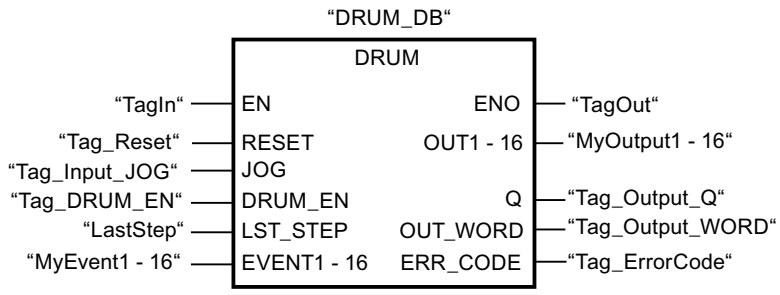

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameter	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE

Parameter	Address	Value
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE

Parameter	Operand	Address	Value
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

DRUM_X: Implement sequencer

Description

The "Implement sequencer" instruction is used to implement a sequencer with maximum 16 steps. You specify the number of the first step on the DSP parameter. You specify the number of the last step on the LST_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT_WORD (in which the output bits are collected together) are written. An output bit is assigned either the corresponding bit of the OUT_VAL array that you specified or the output bit of the preceding step. The value assigned depends on the value of the mask bit on the S_MASK parameter.

Note

The default setting for the mask bit is 0. If you want to change the current setting of one or more mask bits, make this change in the instance data block.

The "Implement sequencer" instruction advances to the next step when there is a rising edge on the JOG parameter. If the instruction is already in the last step, the tags Q and EOD are set when the JOG parameter has a rising edge. In this case the DCC parameter is set to the value "0". The instruction remains in the last step until the RESET parameter is set to the value "1".

You can also assign parameters so that switching to the next step is time dependent. To do this, the DRUM_EN parameter has to be set to the value "1". The transition to the next step is

made when the event bit (EVENTi) belonging to the current step is set and the time specified for the current step has expired. The time for a step is the product of the (DTBP) time base and the time factor valid for the current step (S_PRESET).

Note

The execution time (DCC) remaining in the current step is reduced only when the corresponding event bit (EVENTi) is set.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

If the value "1" is set on the RESET parameter when the instruction is called, the sequencer goes to the step whose number you assigned on the DSP parameter.

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1".

You can also move the next step at any time via the value at the JOG parameter.

When the instruction is called for the first time, you must set the value "1" on the RESET parameter.

When the sequencer is in the last step and the execution time specified for this step has expired, the parameters Q and EOD are set. The instruction remains in the last step until the RESET parameter changes to the value "1".

The instruction is executed only in the STARTUP and RUN modes.

The operating system resets the "Implement sequencer" instruction during a cold restart. During a warm restart, the instruction is not reset. If you want the "Implement sequencer" instruction to be initialized after a warm restart, you have to call the OB 100 with the value "1" on the RESET parameter.

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" resets the sequencer. When the instruction is called for the first time, you have to set the value "1" on the RESET parameter. Constants can also be specified in the RESET parameter.
JOG	Input	BOOL	I, Q, M, D, L	A rising edge (compared to the last call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you assigned to the DRUM_EN parameter.
DRUM_EN	Input	BOOL	I, Q, M, D, L	Control parameter that specifies whether time-dependent switching to the next step is possible. If the value is "1" a time-dependent switching to the next step is possible.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last step Values from 1 to 16 are permitted.
EVENTi, 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit number i (belongs to step i)
OUTj, 0 ≤ j ≤ 15	Output	BOOL	I, Q, M, D, L	Output bit j
Q	Output	BOOL	I, Q, M, D, L	Status parameter that indicates whether the specified execution time of the last step has expired.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Output bits collected together in a tag
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	Not relevant to user (input parameter JOG of the preceding call)
EOD	Static	BOOL	I, Q, M, D, L	Identical to the output parameter Q
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Number of the first step Values from 1 to 16 are permitted.
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Number of the current step

Parameters	Declaration	Data type	Memory area	Description
DCC	Static	DWORD	I, Q, M, D, L, P	The execution time still remaining in the current step in milliseconds (ms) (only relevant if DRUM_EN = 1 and the corresponding event bit = 1)
DTBP	Static	WORD	I, Q, M, D, L, P or constant	The time base valid for all steps in milliseconds (ms)
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	(Not relevant for the user: system time of the previous call)
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	One-dimensional array with the time factor for each step. Recommended selection of the indices: [1 to 16]. In this case, S_PRESET [x] has the time factor of step x.
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the values to be output in each step if they have not been masked out using S_MASK. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, OUT_VAL [x, y] has the value assigned to the output bit OUTy in step x.
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the mask bits for each step. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, S_MASK [x, y] contains the mask bit for the "y"the value to be output in step x. Meaning of the mask bits: • 0: The corresponding value of the previous step is assigned to the relevant output bit. • 1: The corresponding value from OUT_VAL is assigned to the relevant output bit.

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE* (W#16#...)	Explanation
0000	No error
8081	Invalid value at parameter LST_STEP.
8082	Invalid value on parameter DSC.
8083	Invalid value on parameter DSP.
8084	The product DCC = DTBP * S_PRESET[DSC] exceeds the value 2**31-1 (approximately 24.86 days).
*The error codes can be displayed as integer or hexadecimal values in the program editor. You will find information on changing display formats in "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT0 to OUT15) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

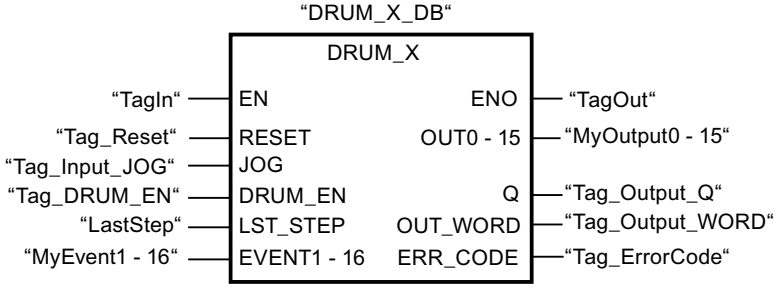

The following tables show how the instruction works based on concrete values:

Before processing

In this example, the following values are used for initializing the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE

Parameters	Operand	Address	Value
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_X_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE

Parameters	Address	Value
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameters	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT0	MyTag_Output_0	M4.0	TRUE
OUT1	MyTag_Output_1	M4.1	TRUE
OUT2	MyTag_Output_2	M4.2	TRUE
OUT3	MyTag_Output_3	M4.3	TRUE
OUT4	MyTag_Output_4	M4.4	TRUE
OUT5	MyTag_Output_5	M4.5	TRUE
OUT6	MyTag_Output_6	M4.6	TRUE
OUT7	MyTag_Output_7	M4.7	TRUE
OUT8	MyTag_Output_8	M5.0	TRUE
OUT9	MyTag_Output_9	M5.1	TRUE
OUT10	MyTag_Output_10	M5.2	TRUE
OUT11	MyTag_Output_11	M5.3	TRUE
OUT12	MyTag_Output_12	M5.4	TRUE
OUT13	MyTag_Output_13	M5.5	TRUE

Parameters	Operand	Address	Value
OUT14	MyTag_Output_14	M5.6	TRUE
OUT15	MyTag_Output_15	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Address	Value
OUT0	MyTag_Output_0	M4.0	TRUE
OUT1	MyTag_Output_1	M4.1	FALSE
OUT2	MyTag_Output_2	M4.2	FALSE
OUT3	MyTag_Output_3	M4.3	FALSE
OUT4	MyTag_Output_4	M4.4	FALSE
OUT5	MyTag_Output_5	M4.5	TRUE
OUT6	MyTag_Output_6	M4.6	FALSE
OUT7	MyTag_Output_7	M4.7	FALSE
OUT8	MyTag_Output_8	M5.0	TRUE
OUT9	MyTag_Output_9	M5.1	TRUE
OUT10	MyTag_Output_10	M5.2	FALSE
OUT11	MyTag_Output_11	M5.3	FALSE
OUT12	MyTag_Output_12	M5.4	FALSE
OUT13	MyTag_Output_13	M5.5	FALSE
OUT14	MyTag_Output_14	M5.6	TRUE
OUT15	MyTag_Output_15	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are saved in the "DRUM_X_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

DCAT: Discrete control-timer alarm

Description

The instruction "Discrete control-timer alarm" is used to accumulate the time from the point at which the parameter CMD issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state on the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET, OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET, OA, CA and CMD_HIS are reset to "0".
- When the signal state of the CMD and CMD_HIS parameters is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state on the parameter OA is set to "1". If the value of the parameter ET does not exceed the value of the parameter PT, the signal state on the parameter OA is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the CMD, CMD_HIS and C_FB parameters return signal state "0", the time difference (ms) since the last execution of the instruction is added to the value of the ET parameter. If the value of the parameter ET exceeds the value of the parameter PT, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the parameter CMD_HIS is set to the value of the parameter CMD.

- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the parameter C_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CMD	Input	BOOL	I, Q, M, D, L	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where one count = 1 ms.
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
CMD_HIS	Static	BOOL	I, Q, M, D, L	CMD history bit

You can find additional information on valid data types under "See also".

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.

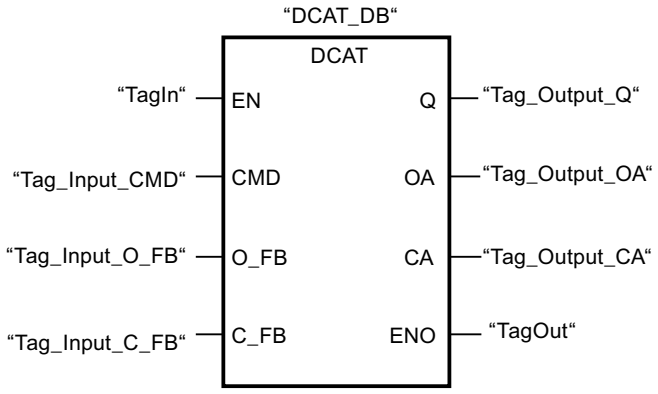

The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

MCAT: Motor control-timer alarm

Description

You can use the "Motor control-timer alarm" instruction to accumulate the time from the point in time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction "Motor control-timer alarm" returns no error information.

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the instruction "Motor control-timer alarm" to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed

Input parameters								Output parameters								
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row of the above-named table (X) is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	A signal state of "0" indicates an error condition.
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset time value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
O_HIS	Static	BOOL	I, Q, M, D, L	"Open" history bit
C_HIS	Static	BOOL	I, Q, M, D, L	"Close" history bit

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

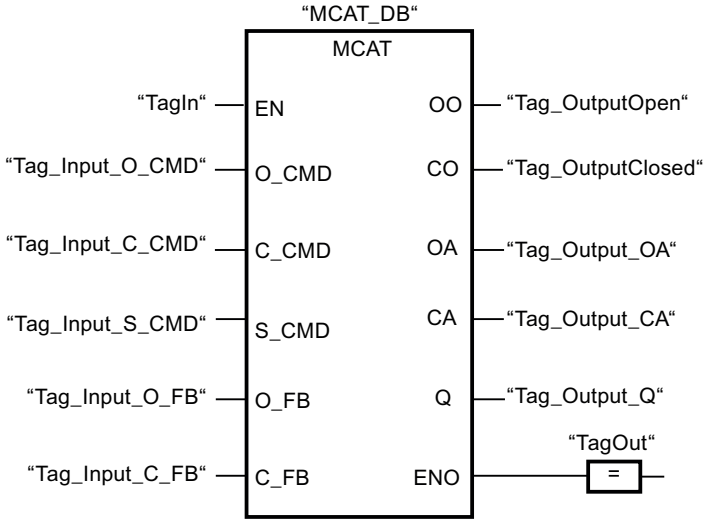

The following tables show how the instruction works based on concrete values:

Before processing

In this example the following values are used for the input and output parameters:

Parameters	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameters	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22

Parameters	Address	Value
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

IMC: Compare input bits with the bits of a mask

Description

The instruction "Compare input bits with the bits of a mask" is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. At the CMP_STEP parameter, specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.

Parameter	Declaration	Data type	Memory area	Description
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

You can find additional information on valid data types under "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

SMC: Compare scan matrix

Description

The "Compare scan matrix" instruction is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found, the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written to the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value of the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY of WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

You can find additional information on valid data types under "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

LEAD_LAG: Lead and lag algorithm

Description

The "Lead and lag algorithm" instruction is used to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the instruction "Lead and lag algorithm" is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the instruction "Lead and lag algorithm" in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

You can find additional information on valid data types under "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

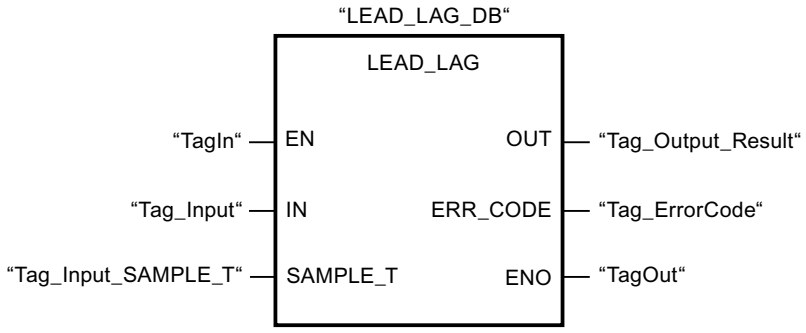

The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

TONR_X: Time accumulator

Description

You use the "Time accumulator" instruction to accumulate time values within a period which you set using the PV input. The instruction is executed when the signal state on the TMR_EN start input changes from "0" to "1". Simultaneously, the timer runs for the duration set at the PV input. While the instruction is being executed, it adds the time value of the DELTA_T input to the time value of the InOut ET.

The following conditions must be met:

- The TMR_EN start input has the signal state "1".
- The time value at InOut ET is less than the time value at the PV parameter.
- The RESET input has the signal state "0".

The instruction is interrupted when the signal state on the TMR_EN start input changes from "1" to "0". As soon as the signal state changes back to "1", the instruction resumes execution.

The instruction is ended when the time value at InOut ET has reached the time value at the PV input. The Q output then has the signal state "1".

As long as the Q output has the signal state "1", the time value at InOut ET remains unchanged. The time value and the Q output are not reset to "0" until the RESET input is set to signal state "1".

Since the "Time accumulator" instruction uses the cycle time (DELTA_T) of the last cycle of the organization block (OB), the instruction may only be called in cyclic organization blocks.

Note

You have to move the cycle time of the organization block from the startup information to the global tag of the DELTA_T parameters.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L, T, C	Enable output
TMR_EN (Timer Enable)	Input	BOOL	I, Q, M, D, L, T, C	Start input Starts the instruction and the expiration of the preset time duration at the PV input
RESET	Input	BOOL	I, Q, M, D, L, T, C	Reset input

Parameters	Declaration	Data type	Memory area	Description
PV (Preset Value)	Input	DINT	I, Q, M, D, L, P or constant	Maximum duration of time recording
DELTA_T	Input	INT	I, Q, M, D, L or constant	Cycle time of organization block of the previous cycle
Q	Output	BOOL	I, Q, M, D, L	Output which is set when the preset time value is reached
ET (Elapsed Time)	InOut	DINT	I, Q, M, D, L	Accumulated time value (Currently elapsed time value + time value of the DELTA_T input)

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:

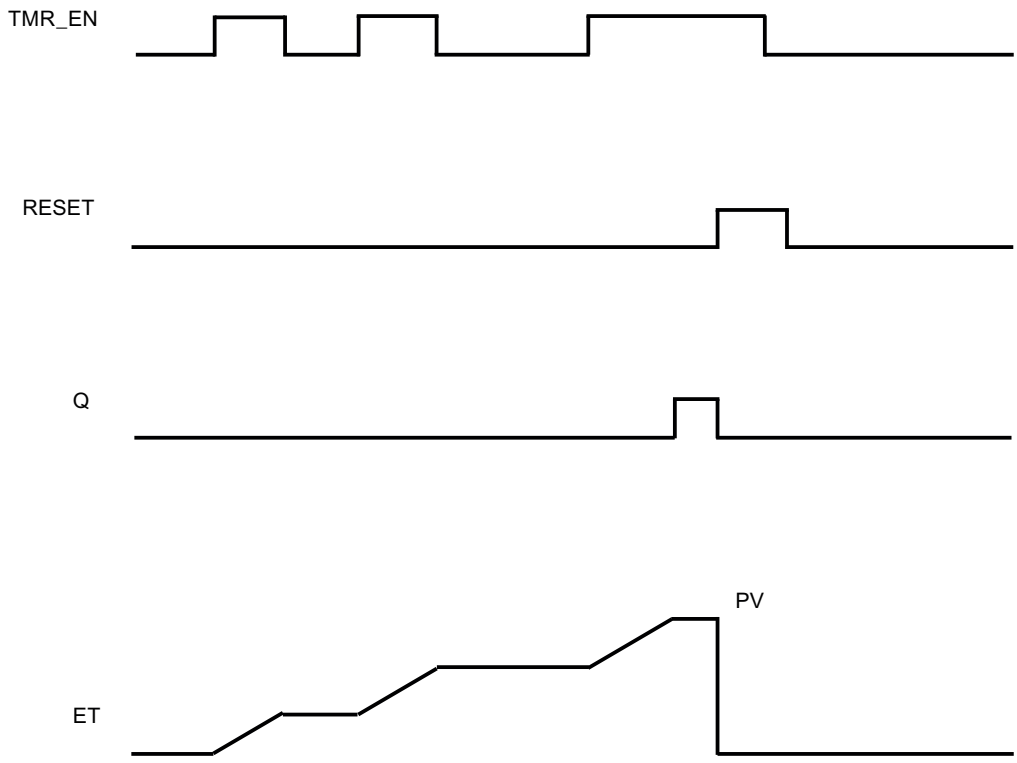

Example

The following example shows how the instruction works:

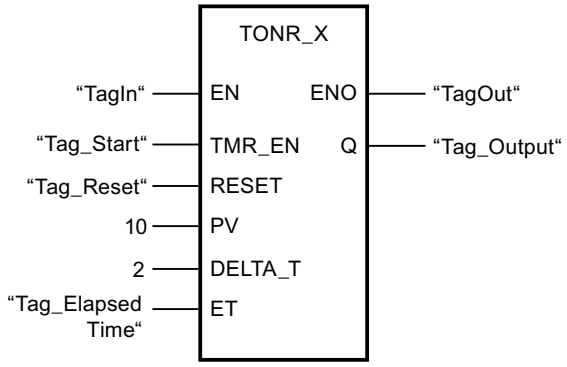

If the operands "TagIn" and "Tag_Start" have the signal state "1", the instruction is started. Each time the signal state changes from "0" to "1" at the TMR_EN input, the cycle time "2" at the DELTA_T input is added to the current time value of InOut ET until the maximum time value "10" of the PV input is reached. If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

WSR: Save data to shift register

Description

The "Save data to shift register" instruction is used to write a bit from a specified source to a shift register. Each element of the register is shifted into the next address. The data contained in the last address of the shift register is lost after the shift.

Each time the instruction is executed with the value "0" on the RESET parameter, new data are read from the source (S_DATA) and shifted to the start address (START) of the shift register. The LENGTH parameter specifies the number of elements to be shifted. You specify the data type of the elements on the E_TYPE parameter. The instruction is not executed if an invalid data type is specified on the E_TYPE parameter.

The register is deleted if the RESET parameter is set to "1" when the instruction is executed. The Q parameter is set to the signal state "1" when the shift register is filled with zeros or deleted.

Parameters

The following table shows the parameters of the "Save data to shift register" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_DATA	Input	POINTER	I, Q, M, D	Points to the source data element to be shifted to the start address (START). For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
START	Input	POINTER	I, Q, M, D	Points to the start address of the register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
LEN	Input	WORD	I, Q, M, D, L, P	Number of elements to be shifted.
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the elements to be shifted. The following data types are permitted: • W#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
Q	Output	BOOL	Q, M, D, L	Is set to "0" if RESET parameter is active (1) or all elements to be shifted have the value "0".

Example

In the following example four elements of the data type WORD are shifted starting from the address specified on the START parameter. When the first element of the register has been shifted to the next address, the first address is filled with the data that the pointer references on the S_DATA parameter. The value of the last element in the register is lost in the process.

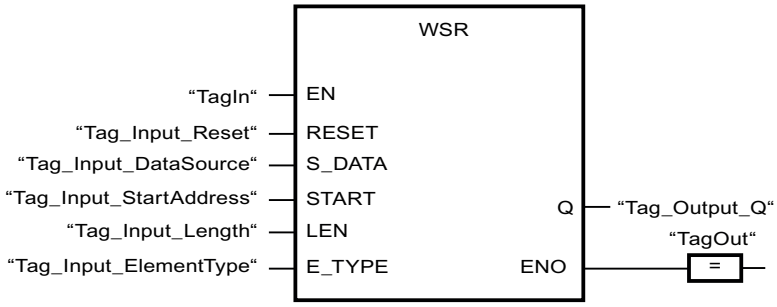

The following tables show how the instruction works based on concrete values:

Before processing

In this example the following values are used for the input and output parameters:

Parameters	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1111 DW2 = W#16#2222 DW4 = W#16#3333 DW6 = W#16#4444
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1234 DW2 = W#16#1111 DW4 = W#16#2222 DW6 = W#16#3333
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04

Parameters	Operand	Value
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

SHRB: Shift bit to shift register

Description

The "Shift bit to shift register" instruction is used to write a bit from a specified source (DATA) to a shift register. Each time the instruction is executed with a value "0" on the RESET parameter, new data are read from the source and shifted to the start address (S_BIT) of the shift register. All following bits are shifted by one. The bit contained at the last address (S_BIT + N) is lost after the shift. If the RESET parameter is set to "1" during the processing of the instruction, the addresses in the shift register are reset to "0" and not shifted.

Parameters

The following table shows the parameters of the "Shift bit to shift register" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DATA	Input	BOOL	I, Q, M, D, L	Source bit
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_BIT	Input	POINTER	I, Q, M, D	Points to the start bit of the shift register For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	WORD	I, Q, M, D, L, P	Number of bits to be shifted.

Example

In the following example the N parameter is set to the value 14 ("E" in hexadecimal). As a result 14 bits starting from the address specified on the S_BIT parameter are shifted. After the bits have been shifted, the first address is filled with the data from the source on the DATA parameter. The last bit value is lost.

The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input and output parameters:

Parameters	Operand	Value
RESET	Tag_Input_Reset	0
DATA	Tag_Input_Data	0
S_BIT	Tag_Input_BitSource	P#DB1.DBX1.4
N	Tag_Input_Number	B#16#0E

Before processing the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	0	1	1	1	0	1	0

2.7	...					...	2.0
0	0	1	1	1	0	1	0

3.7	...					...	3.0
0	0	1	1	1	0	1	0

After processing

After the processing of the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	1	1	0	1	0	1	0

2.7	...					...	2.0
0	1	1	1	0	1	0	0

3.7	...					...	3.0
0	0	1	1	1	0	0	0

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

SEG: Create bit pattern for seven-segment display

Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)
0000	00111111	0
0001	00000110	1
0010	01011011	2
0011	01001111	3
0100	01100110	4
0101	01101101	5
0110	01111101	6
0111	00000111	7
1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	B
1100	00111001	C
1101	01011110	D
1110	01111001	E
1111	01110001	F

Seven-segment display

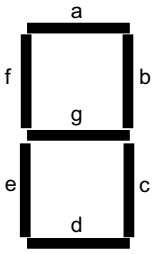

Parameters

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:

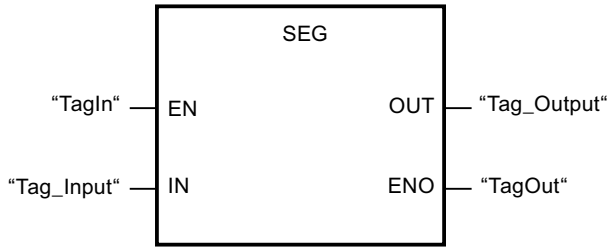

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

BCDCPL: Create tens complement

Description

You can use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:

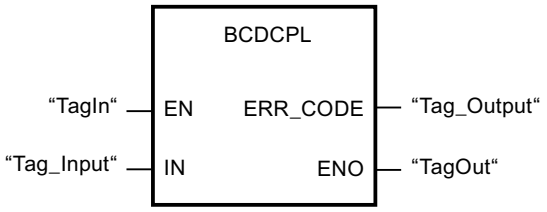

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Switching display formats in the program status (Page 7698)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output at the RET_VAL parameter.

Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

Example

The following example shows how the instruction works:

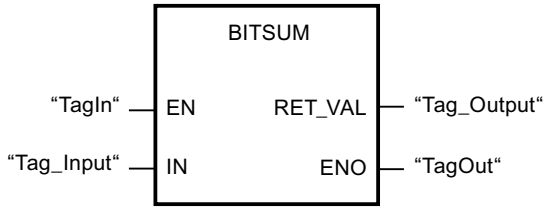

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Basics of the EN/ENO mechanism (Page 141)

Basic information on FBD (Page 7185)

4.2.2.3 STL

IEC timers

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set an operation for a programmed duration. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The time programmed in the (PT) parameter begins when the instruction starts. The Q parameter is set for the period of time PT, regardless of the subsequent changes in the signal of the IN parameter. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the time duration PT is active.

You can query the current time value at the ET output. The time value starts at T#0s and ends when the value of the time PT is reached. When the time PT has elapsed and the signal state at input IN is "0", the ET output is reset.

Use the "Call block" (CALL) instruction in the program code to call the "Generate pulse" instruction.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored. You can declare an IEC timer as follows:

- Declaration of a data block of the type TP (for example, "TP_DB")
- Declaration as a local tag of the type TP in the "Static" section of a block (for example, #MyTP_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate pulse" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the "Generate pulse" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value specified at the PT parameter must be positive
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the behavior of the "Generate pulse" instruction after the start:

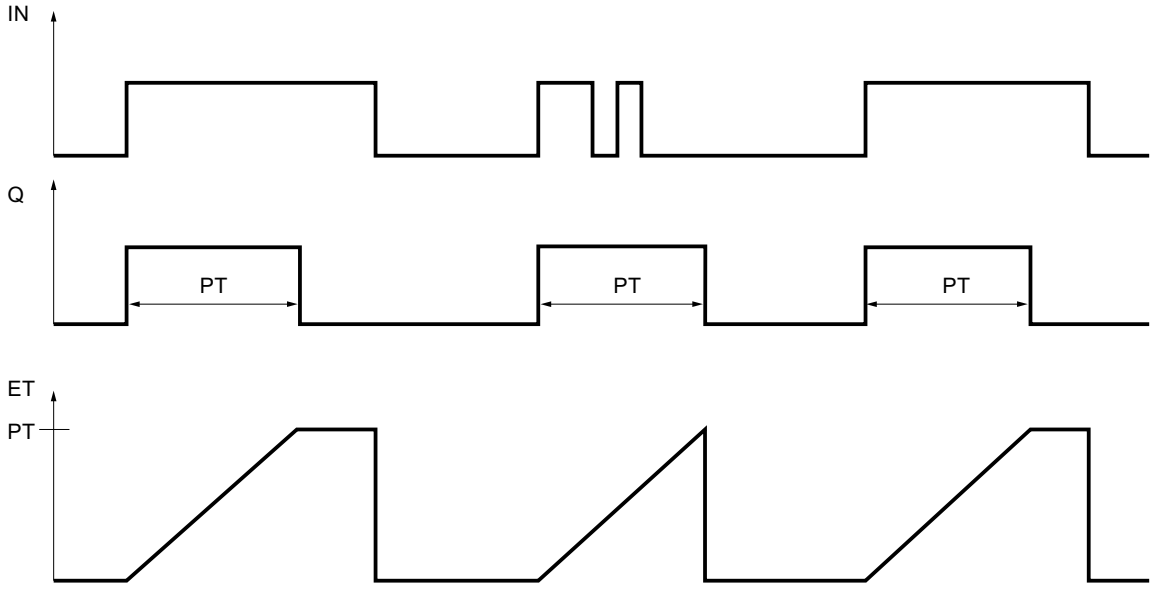

Example

The following example shows how the instruction works:

STL	Explanation
CALL TP, "TP_DB"	// The instruction is called. The "TP_DB" data block is assigned to the instruction.
IN := "Tag_Start"	// The instruction is executed on a rising signal edge of the operand.
PT := "Tag_Preset-TIME"	// Duration of the pulse
Q := "Tag_Output"	// The operand is set for the duration that was specified by the "Tag_PresetTIME" operand.
ET := "Tag_Elapsed"	// Current timer value

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay a positive signal edge by the time set for PT. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the time PT has expired, the Q parameter has signal state "1". The operand of the Q parameter remains set as long as the IN start input is still "1". When the signal state at the start input changes from "1" to "0", the Q parameter is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The time value starts at T#0s and ends when the value of the time PT is reached. When the time PT has elapsed and the signal state at input IN is "0", the ET output is reset.

In the program code, you call the "Generate on-delay" instruction with the "Call block" (CALL) instruction.

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. You can declare an IEC timer as follows:

- Declaration of a data block of the type TON (for example, "TON_DB")
- Declaration as a local tag of the type TON in the "Static" section of a block (for example, #MyTON_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value specified at the PT parameter must be positive
Q	Output	BOOL	I, Q, M, D, L	Signal state that is delayed by the time PT
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the behavior of the "Generate on-delay" instruction after the start:

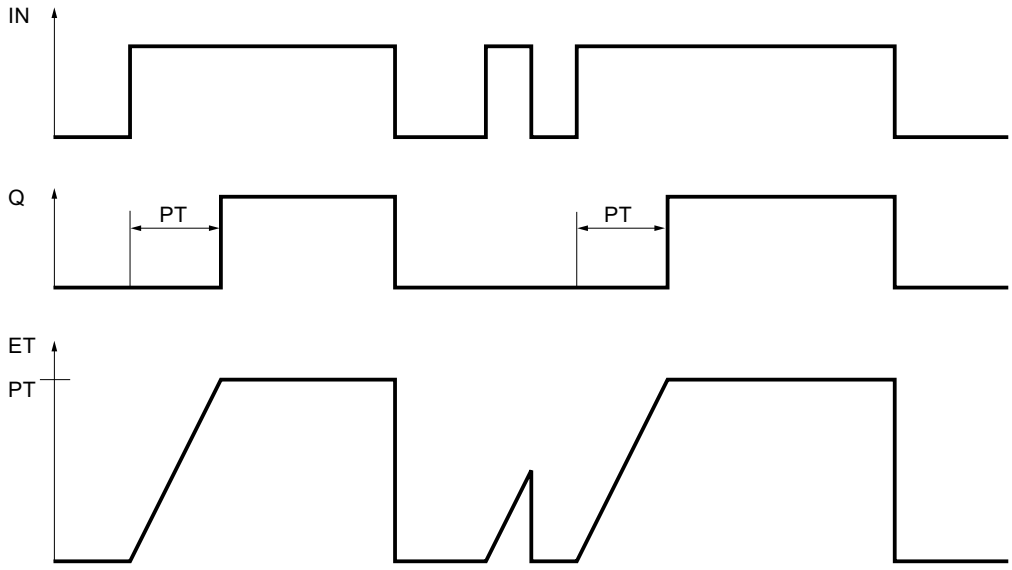

Example

The following example shows how the instruction works:

STL	Explanation
CALL TON, "TON_DB"	// The instruction is called. The "TON_DB" data block is assigned to the instruction.
IN := "Tag_Start"	// The instruction is executed on a rising signal edge of the operand.
PT := "Tag_PresetTIME"	// Specifies the time by which the rising signal edge of the IN parameter is delayed.
Q := "Tag_Output"	// The operand is set if the duration PT that was specified by the "Tag_PresetTIME" tag has expired.
	// The Q parameter remains set as long as the "Tag_Start" variable is "1".
	// When the signal state at the start input changes from "1" to "0", the operand of the Q parameter is reset.
ET := "Tag_ElapsedTIME"	// Current timer value

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay a negative signal edge by the time set for PT. The operand of the Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state of the IN parameter changes back to "0", the programmed time PT starts. The operand of the Q parameter remains set as long the time duration PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before time PT expires, the timer is reset. The signal state of the Q parameter remains set to "1".

You can query the current time value set in the ET parameter. The current timer value starts at T#0s and ends when the value of the time duration PT is reached. When the time PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1".

In the program code, you call the "Generate off-delay" instruction with the "Call block" (CALL) instruction.

Each call of the "Generate off-delay" instruction must be assigned to an IEC Timer in which the instruction data is stored. You can declare an IEC timer as follows:

- Declaration of a data block of the type TOF (for example, "TOF_DB")
- Declaration as a local tag of the type TOF in the "Static" section of a block (for example, #MyTOF_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the "Generate off-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the off delay The value specified at the PT parameter must be positive
Q	Output	BOOL	I, Q, M, D, L	Signal state that is delayed by the time PT
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the behavior of the "Generate off-delay" instruction after the start:

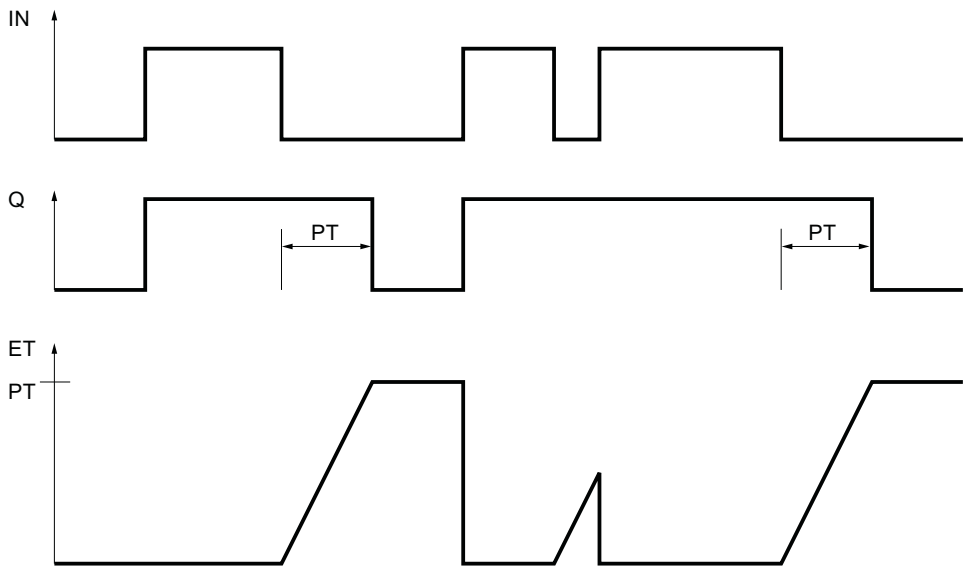

Example

The following example shows how the instruction works:

STL	Explanation
`CALL TOF, "TOF_DB"`	`// The instruction is called. The "TOF_DB" data block is assigned to the instruction.`
`IN := "Tag_Start"`	`// The instruction is executed on a rising signal edge of the operand.`
`PT := "Tag_PresetTIME"`	`// Specifies the time by which the falling signal edge of the IN parameter is delayed.`

STL	Explanation

Q := "Tag_Output"	// The operand is set when the instruction is started by a rising signal edge of the IN parameter.
	// When the value at the IN parameter changes from "1" to "0", the "Tag_Output" operand remains set as long as the time specified by the "Tag_PresetTIME" tag is running.
	// When the time of the PT parameter expires, the operand is reset.

ET := "Tag_ElapsedTIME"	// Current timer value

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

IEC Counters

CTU: Count up

Description

You can use the "Count up" instruction to increment the counter value at the CV parameter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the parameter CV is incremented by one. The counter value is increased each time a positive signal edge is detected, until it reaches the high limit of the data type INT. When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current counter value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the R parameter has signal state "1", the signal state of the CU parameter has no effect on the instruction.

In the program code, you call the "Count up" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC Counter in which the instruction data is stored. You can declare an IEC Counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")
- Declaration as a local tag of the type CTU in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "1" set for the R parameter of the instruction. If instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, P	Reset input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the Q output is set
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

Example

The following example shows how the instruction works:

STL	Explanation
`CALL CTU, "CTU_DB"`	`// The instruction is called. The "CTU_DB" data block is assigned to the instruction.`
`CU := "Tag_StartCTU"`	`// When the signal state of the "Tag_StartCTU" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one.`
	`// The counter value is incremented until the high limit of INT = 32767 is reached.`
`R := "Tag_ResetCounter"`	`// When the signal state of the "Tag_ResetCounter" operand changes to "1", the "Tag_CounterValue" operand is reset to "0".`
`PV := "Tag_PresetValue"`	`// The operand determines the value at which the operand of the Q parameter is set.`

STL	Explanation
`Q := "Tag_CounterStatus"`	`// The operand is set as long as the current counter value is greater than or equal to the value of the PV parameter.`
`CV := "Tag_CounterValue"`	`// Current counter value`

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Editing STL instructions (Page 7277)

Inserting STL instructions (Page 7259)

Instances (Page 62)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the counter value of the CV parameter. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type INT. When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the LD parameter has signal state "1", the signal state of the CD parameter has no effect on the instruction.

In the program code, you call the "Count down" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC Counter in which the instruction data is stored. You can declare an IEC Counter as follows:

- Declaration of a data block of the type CTD (for example, "CTD_DB")
- Declaration as a local tag of the type CTD in the "Static" section of a block (for example, #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will

find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case the desired initial value for the CV parameter is specified in the PV parameter. If instances of the "Count down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

Example

The following example shows how the instruction works:

STL	Explanation
`CALL CTD, "CTD_DB"`	`// The instruction is called. The "CTD_DB" data block is assigned to the instruction.`
`CD := "Tag_StartCTD"`	`// When the signal state of the "Tag_StartCTD" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterValue" operand is decremented by one.`
	`// The counter value of the CV parameter is decremented until the low limit of INT = -32768 is reached.`
`LD := "Tag_LoadPV"`	`// When the signal state of the "Tag_LoadPV" operand changes to "1", the "Tag_CounterValue" operand is set to the value of the "Tag_PresetValue" operand.`
`PV := "Tag_PresetValue"`	`// Specifies the value to which the counter is set when the signal state is "1" in the LD parameter.`
`Q := "Tag_CounterStatus"`	`// The operand is set when the current counter value is less than or equal to zero.`
`CV := "Tag_CounterValue"`	`// Current counter value`

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

CTUD: Count up and down

Description

Use the "Count up and down" instruction to increment or decrement the counter value at the CV parameter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the counter value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in a program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the INT parameter. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of the specified INT data type is reached, the counter value is not decremented any further.

When the signal state of the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has signal state "1", the signal state of the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0". You can also specify a constant for the PV parameter.

You can query the status of the down counter at the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

In the program code, you call the "Count up and down" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC Counter in which the instruction data is stored. You can declare an IEC Counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")
- Declaration as a local tag of the type CTUD in the "Static" section of a block (for example, #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Counter is stored in its own data block (single instance) or as a

local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a startup OB:

- When used as up counter, the value at the R parameter must be set to "1".
- When used as down counter, the value at the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the QU output is set. / Value to which the CV output is set when LD = 1. Value at which the QU output is set. / Value to which the CV output is set when LD = 1.
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	INT	I, Q, M, D, L, P	Current counter value

Example

The following example shows how the instruction works:

STL	Explanation
`CALL CTUD, "CTUD_DB"`	`// The instruction is called. The "CTUD_DB" data block is assigned to the instruction.`
`CU := "Tag_StartCTU"`	`// When the signal state of the "Tag_StartCTU" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterVal-ue" operand is incremented by one.`
	`// The counter value is incremented on a rising edge of the CU parameter, until it reaches the high limit of INT = 32767.`

STL	Explanation
CD := "Tag_StartCTD"	// When the signal state of the "Tag_StartCTD" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterValue" operand is decremented by one. // The counter value of the CV parameter is decremented until the low limit of -32768 is reached.
R := "Tag_ResetCounter"	// When the signal state of the "Tag_ResetCounter" operand changes to "1", the "Tag_CounterValue" operand is reset to "0".
LD := "Tag_LoadPV"	// When the signal state of the "Tag_LoadPV" operand changes to "1", the "Tag_CounterValue" operand is set to the value of the "Tag_PresetValue" operand.
PV := "Tag_PresetValue"	// Specifies the value to which the counter is set when the signal state is "1" in the LD parameter.
QU := "Tag_CounterStatus"	// The operand is set as long as the current counter value is greater than or equal to the value of the PV parameter.
QD := "Tag_CounterStatus"	// The operand is set when the current counter value is less than or equal to zero.
CV := "Tag_CounterValue"	// Current counter value

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Instances (Page 62)

Math functions

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the lowest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT parameter is invalid if one of the following conditions is met:

- The specified operands are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters even with deactivated IEC check. If you use the data types WORD or DWORD at the input parameters, the instruction returns an undefined result.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Second input value
IN3	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Third input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the parameters INn and OUT from the "???" drop-down list.

Example

The following example shows how the instruction works:

STL	Explanation
CALL MIN	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
IN1 := "TagIn_Value1"	// First input value is compared.
IN2 := "TagIn_Value2"	// Second input value to be compared.
IN3 := "TagIn_Value3"	// Third input value to be compared.
OUT := "Tag_Minimum"	The "Tag_Minimum" operand is written with the value of the "TagIn_Value1" operand because this has the lowest value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Minimum	12222

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the highest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is fulfilled:

- The specified operands are not of the same data type.
- An operand has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters even with deactivated IEC check. If you use the data types WORD or DWORD at the input parameters, the instruction returns an undefined result.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Second input value
IN3	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Third input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the parameters INn and OUT from the "???" drop-down list.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL MAX                     // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
value_type := Int            // Data type of the parameters IN and OUT
IN1 := "TagIn_Value1"        // First input value is compared.
IN2 := "TagIn_Value2"        // Second input value to be compared.
IN3 := "TagIn_Value3"        // Third input value to be compared.
OUT := "Tag_Maximum"         // The "Tag_Maximum" operand is written with the value
                             of the "TagIn_Value2" operand because this has the
                             highest value.
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Maximum	14444

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value at the IN input to the values at the inputs MN and MX. If the value at the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the IN input value is below the low limit MN, the OUT output is set to the value of the MN input. If the high limit MX is exceeded, the OUT output is set to the value of the MX input.

The instruction is only executed if all the tags at the inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is fulfilled:

- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN parameter is greater than the value of the MX parameter.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters even with deactivated IEC check. If you use the data types WORD or DWORD at the input parameters, the instruction returns an undefined result.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameters	Declaration	Data type	Memory area	Description
MN	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Low limit
IN	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Input value
MX	Input	Integers, floating-point numbers	I, Q, M, D, L, P	High limit
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the parameters MN, IN, MX and OUT from the "???" drop-down list.

Example

The following example shows how the instruction works:

STL	Explanation
CALL LIMIT	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
MN := "Tag_LowLimit"	// Low limit
IN := "Tag_InputValue"	// Input value
MX := "Tag_HighLimit"	// High limit
OUT := "Tag_Result"	// The "Tag_Result" operand is written with the value of the "Tag_LowLimit" operand because the value of the "Tag_InputValue" operand is outside the defined limit range.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_LowLimit	12000
IN	Tag_InputValue	8000
MX	Tag_HighLimit	16000
OUT	Tag_Result	12000

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Move operations

BLKMOV: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:

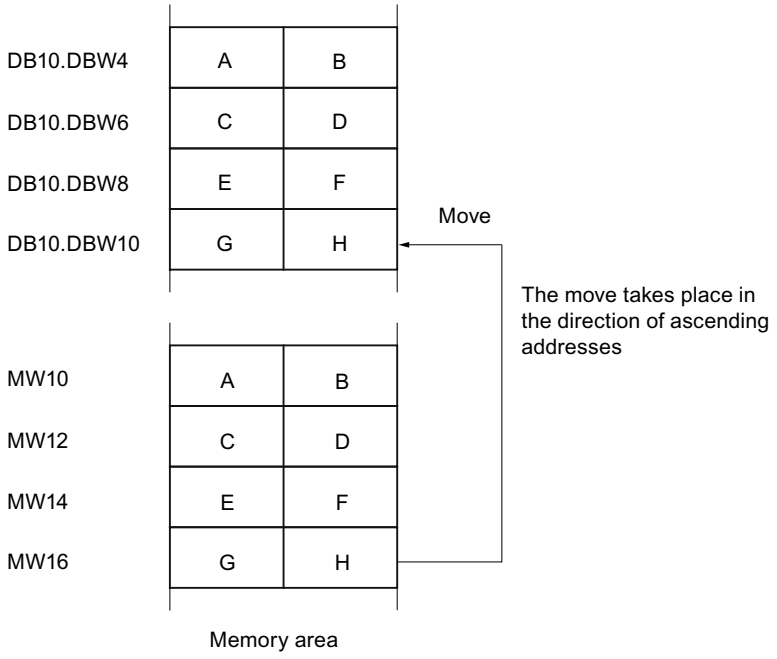

Consistency of source and destination data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, BLKMOV is interrupted while copying from a DB that is not runtime-relevant, the execution of BLKMOV can no longer be nested.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks not runtime-relevant

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If an area of the BOOL data type is copied, the specified length of the area must be divisible by 8, otherwise the instruction will not execute.

Note

If the actual source or destination area is smaller than a destination or source area specified by the parameter SRCBLK or DSTBLK, the response depends on the CPU type:

- The following applies to S7-300 CPUs:
  No data is transferred. Error code W#16#837F is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data is transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data is transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Rules for moving data blocks that are not runtime-relevant

The source area can also be in a data block in load memory that is not runtime-relevant. Data blocks that are not runtime-relevant are indicated by the keyword UNLINKED.

If a data block that is not runtime-relevant is copied to work memory with the "Move block" instruction and loaded at the same time, for example, by the programming device, the execution of the instruction can be delayed for several milliseconds. This results in a longer OB cycle and may trip the cycle monitoring.

If a data block that is not runtime-relevant is moved with the "Move block" instruction and the move operation is interrupted, the execution of the instruction can no longer be continued.

If the "Read from data block in the load memory" instruction is available on your CPU, you must use this instruction to read data blocks that are not runtime-relevant from load memory. If you use the "Move block" instruction, error W#16#8092 is output.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1]	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8xyy	You can find additional information here: Evaluating errors with output parameter RET_VAL (Page 162)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

STL	Explanation
`CALL BLKMOV`	`// The instruction is called.`
`SRCBLK := P#M100.0 BYTE 10`	`// 10 Bytes starting from MB100 are moved.`
`RET_VAL := "Tag_ErrorCode"`	`// If an error occurs during the move operation, its error code is output in the variable "Tag_ErrorCode".`
`DSTBLK := P#M200.0 BYTE 10`	`// The moved bytes are written to the 10 bytes starting from MB200`

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with integers (Page 178)

STL Basics (Page 7245)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If an area of the BOOL data type is copied, the specified length of the area must be divisible by 8, otherwise the instruction will not execute.

You can use the "Move block uninterruptible" instruction to move a maximum of 512 bytes. Note the CPU-specific restrictions for this.

Note

If the actual source or destination area is smaller than a destination or source area specified by the parameter SRCBLK or DSTBLK, the response depends on the CPU type:

- The following applies to S7-300 CPUs:
  No data is transferred. Error code W#16#837F is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data is transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data is transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as the area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1]	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
1) The DSTBLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source or target area is only available in the load memory.
8xyy	You can find additional information here: Evaluating errors with output parameter RET_VAL (Page 162)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

```
STL                              Explanation
CALL UBLKMOV                     // The instruction is called.
SRCBLK := P#M100.0 BYTE 10       // 10 Bytes starting from MB100 are moved.
RET_VAL := "Tag_ErrorCode"       // If an error occurs during the move operation, its
                                 error code is output in the variable "Tag_ErrorCode".
DSTBLK := P#M200.0 BYTE 10       // The moved bytes are written to the 10 bytes start-
                                 ing from MB200
```

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with integers (Page 178)

STL Basics (Page 7245)

FILL: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:

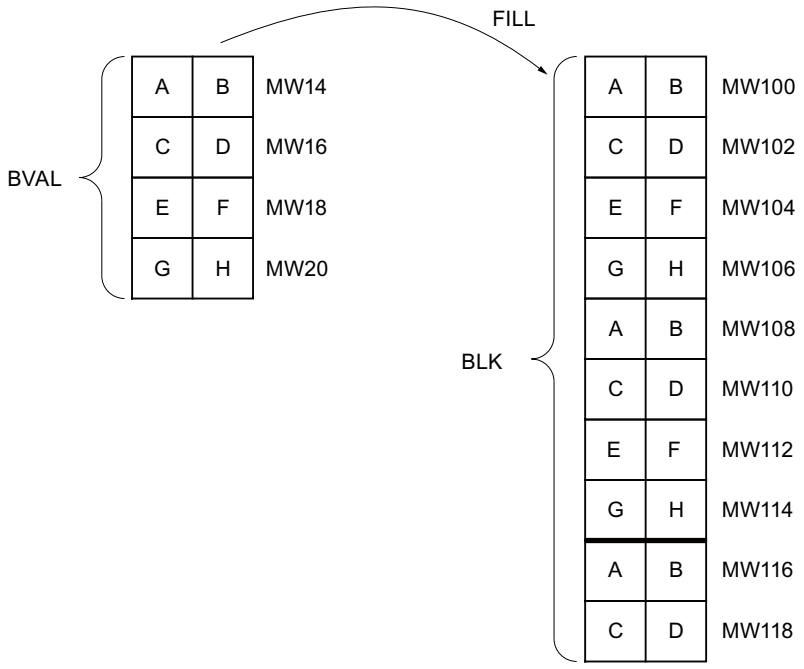

Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, the specified length must be divisible by 8; otherwise the instruction will not execute.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Note

If the actual source or destination area is smaller than a destination or source area specified by the parameter BVAL or BLK, the response depends on the CPU type:

- The following applies to S7-300 CPUs:
  No data is transferred. Error code W#16#837F is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data is transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data is transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
BVAL	Input	ANY	I, Q, M, D, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
BLK	Output [1)]	ANY	I, Q, M, D, L, P	Specification of the memory area that will be filled with the content of the source area.
1) The BLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8xyy	You can find additional information here: Evaluating errors with output parameter RET_VAL (Page 162)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL FILL                    // The "Fill block" instruction is called.
BVAL := P#M14.0 WORD 4       // The area from MW14 to MW20 is moved
RET_VAL := "Tag_ErrorCode"   // Error information
BLK := P#M100.0 WORD 10      // The memory area from MW100 to MW118 is filled with
                             the content of 4 words that are contained in the memo-
                             ry area on the BVAL parameter
```

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with integers (Page 178)

STL Basics (Page 7245)

Conversion operations

SCALE: Scale

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number that can be scaled in physical units between a low limit value and a high limit value. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled inversely proportionate to the input value.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"

Error code* (W#16#...)	Explanation
8xyy	You can find additional information here: Evaluating errors with output parameter RET_VAL (Page 162)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SCALE                   // The instruction is called.
IN := "Tag_InputValue"       // Specifies the value to be converted and scaled
HI_LIM := "Tag_HighLimit"    // High limit
LO_LIM := "Tag_LowLimit"     // Low limit
BIPOLAR := "Tag_Bipolar"     // Indicates whether the value at the IN parameter is
                             to be interpreted as bipolar or unipolar
RET_VAL := "Tag_ErrorCode"   // Error information
OUT := "Tag_OutputValue"     // Result of the instruction
```

The following table shows the values of the various operands before execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	50.03978588

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Setting the status bits in instructions with integers (Page 178)

STL Basics (Page 7245)

UNSCALE: Unscale

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at parameter IN is not within the limits defined by HI_LIM and LO_LIM, an error is output and the result is set to the nearest limit.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
8xyy	You can find additional information here: Evaluating errors with output parameter RET_VAL (Page 162)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL UNSCALE                 // The instruction is called.
IN := "Tag_InputValue"       // Specifies the value to be unscaled
HI_LIM := "Tag_HighLimit"    // High limit
LO_LIM := "Tag_LowLimit"     // Low limit
BIPOLAR := "Tag_Bipolar"     // Indicates whether the value at the IN parameter is
                             to be interpreted as bipolar or unipolar
RET_VAL := "Tag_ErrorCode"   // Error information
OUT := "Tag_OutputValue"     // Result of the instruction
```

The following table shows the values of the various operands before execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	22

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

STL Basics (Page 7245)

Program control operations

Runtime control

COMPRESS: Compress CPU memory (S7-400)

Description

The "Compress CPU memory" instruction is used to compress the working memory of the CPU or the RAM load memory. During this compression the gaps caused by deleting or loading blocks are filled by pushing together the existing blocks.

The compression operation is started by calling the "Compress CPU memory" instruction. The compression operation is divided into several program cycles. The compression operation is monitored by the values of the individual parameters. If value "1" is on the BUSY parameter of the instruction, the compression operation is still being executed. The value "1" on the DONE parameter reports that the compression operation has been completed. Error information can be queried on the RET_VAL parameter.

If an externally started compression operation has already been executed, calling the "Compress CPU memory" instruction will cause an error display.

Checking the compression function

Calling the instruction "Compress CPU memory" once starts the compression function. However, you have no control over whether the compression is performed successfully.

If you want to check the compression function, follow the steps outlined below:

The "Compress CPU memory" instruction must be called in the cycle. First evaluate the RET_VAL parameter after every call. If it has value "0", the parameters BUSY and DONE has to be evaluated. If BUSY = "1" and DONE = "0", this indicates that the compression function is still active. The compression has not been completed successfully until BUSY changes to value "0" and DONE to the value "1". Compression is repeated if the "Compress CPU memory" instruction is called again afterwards.

Parameters

The following table shows the parameters of the "Compress CPU memory" instruction:

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Information on whether compression operation is active
DONE	Output	BOOL	I, Q, M, D, L	Information on whether compression operation is complete

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error The status of the compression operation can be queried on the BUSY and DONE parameters.
8091	An externally started compression operation is already being executed.
8092	The instruction cannot be executed, because one of the following conditions applies: • The "Delete block" function is active. • A programming device function is accessing one of the blocks to be moved. • An externally started compression operation is already being executed. • The H-CPU is running a link-up or update function.
8xyy	You can find additional information here: Evaluating errors with output parameter RET_VAL (Page 162)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

See also

Overview of the valid data types (Page 231)

Setting status bits in program control operation instructions (Page 180)

STL Basics (Page 7245)

RE_TRIGR: Restart cycle monitoring time

Description

The "Restart cycle monitoring time" instruction is used to restart the cycle monitoring time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called in all blocks regardless of the priority .

If the instruction is called in a block with a higher priority, for example, a hardware interrupt or a diagnostic interrupt, the instruction will not be executed.

The "Restart cycle monitoring time" instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Parameters

The "Restart cycle monitoring time" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 231)

Setting status bits in program control operation instructions (Page 180)

STL Basics (Page 7245)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Parameters

The "Exit program" instruction has no parameters and returns no error information.

See also

Overview of the valid data types (Page 231)

Setting status bits in program control operation instructions (Page 180)

STL Basics (Page 7245)

WAIT: Configure time delay

Description

You can use the "Configure time delay" instruction to pause the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to 32767 microseconds (µs). The smallest possible delay time depends on the CPU and corresponds to the execution time of the instruction.

The execution of the instruction can be interrupted by higher priority events and does not return any error information.

Applies to S7-300-CPUs:

For the S7-300-CPUs (except S7-318), the delay time programmed by the instruction is a minimum time. It is extended by the execution time of the nested priority classes and by system utilizations.

Note

Negative delay time

If you specify a negative delay time at parameter WT, the enable output ENO, or the RLO and the BR bit, will return the signal state FALSE. A negative delay time does not affect the CPU. The following instructions linked to enable output ENO are not executed in LAD or FBD.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 231)

Setting status bits in program control operation instructions (Page 180)

STL Basics (Page 7245)

PROTECT: Change protection level

Description

The instruction "Change protection level" is used to change the configured protection level "1" of the CPU.

The configured protection level "1" with selected option "Removable with password", means that you have to configure a password. This password enables you to write-access during operation the CPU which was read-only protected with the instruction "Change Protection Level" (e. g. Load).

You specify the protection level on the MODE parameter of the instruction. The current protection level of your CPU is displayed in the inspector window under "Properties > Protection".

The following table shows the meaning of the protection levels.

Protection level	Meaning
1	All programming device functions are permitted. Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
2	• The program and the configuration of the CPU cannot be changed. The program in the CPU can be read. • The functions for process control, process monitoring, and process communication are permitted. • All informational functions are permitted. • If you know the valid password you can cancel the set write protection. • Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
3	• You cannot cancel the set read and write protection, even if you know the valid password. • If a valid connection exists at the time when the instruction "Change protection level" is called, the call of the instruction does not affect this connection.

If you have configured protection level "2" or "3" for your CPU, using the instruction "Change protection level" has no effect.

You can use the SSL parts list with the SSL ID W#16#0232 and index W#16#0004 to check whether protection level "2" or "3" is still in effect after the execution of the "Change protection level" instruction.

Note

You can only use the "Change protection level" instruction to set a protection level that is equal to or higher than the protection level you have configured with STEP 7 "Configure hardware. You have, for example, configured protection level 3 and then called the instruction "Change protection level" with the MODE parameter = 12. This blocks the password authentication. If the instruction is called again with the parameter MODE = 0 or 1, protection level 3 is retained and the read and write protection canceled by entry of the valid password.

Effects of the operating state transitions on the protection level

The following table shows what effects the operating state transitions have on the protection level set with the "Change protection level" instruction.

Action	Effects on the protection level
• Operating mode switch to STOP • Unbuffered POWER ON • Memory reset with operating mode switch • Configuration and/or programming changes in STOP mode of the S7-400 CPU	The protection level is set to "1". All programming device functions are permitted.
• Cold restart • Warm restart	The protection level is set by the operating system to "1". All programming device functions are permitted. If required, you can set the protection level in your program to "2" or "3" by calling the "Change protection level" instruction.
Buffered POWER ON	Remains unchanged
Operating mode transition RUN/STARTUP/HOLD -> STOP (by calling the "Exit program" instruction (STP) due to program error without corresponding error OB, or by operator action)	Remains unchanged
S7-400: Restart by user action or automatically following POWER ON	Remains unchanged

Parameters

The following table shows the parameters of the "Change protection level" instruction:

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	WORD	I, Q, M, D, L or constant	Protection level Possible values: • W#16#0000: Setting protection level "1" • W#16#0001: Setting protection level "2" with password authentication • W#16#000C: Setting protection level "3" without password authentication Constants can also be specified at the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs during processing, information on the error is output.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error
8090	Invalid value at MODE parameter.
80C3	The required operating resources are currently occupied.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The CPU should always be protected against access in RUN mode. The protection should only be rescinded when the operating mode switch is at STOP. You can reach this functionality by calling the "Change protection level" instruction in a startup OB and executing the instruction with the value W#16#0001 (protection level "2") on the MODE parameter. This way, protection level "2" remains set even after a warm restart.

```
STL                            Explanation
CALL PROTECT                   // The instruction is called.
MODE := "Tag_ProtectionLevel"  // Set protection level
RET_VAL := "Tag_ErrorCode"     // Error information
```

See also

Overview of the valid data types (Page 231)

Setting status bits in program control operation instructions (Page 180)

Evaluating errors with output parameter RET_VAL (Page 162)

STL Basics (Page 7245)

CiR: Control CiR process (S7-400)

Description

The "Control CiR process" instruction is used to influence the CiR process in the user program as follows:

- Disable CiR process. In this case, the download of a modified configuration from the programming device to the CPU is always rejected. The disable is maintained until you cancel it again with the "Control CiR process" instruction.
- You can disable the CiR process conditionally by specifying a high limit for the CiR synchronization time on the FRZ_TIME parameter. In this case, the download of a modified configuration from the programming device to the CPU is only accepted if the CiR synchronization time calculated in the CPU does not exceed the specified high limit.

- Enable CiR process. The high limit of the CiR synchronization time is set to the default value 1000 ms.
- Determine whether the CiR process is enabled or not. If the CiR process is enabled or conditionally enabled, you will get the current high limit for the CiR synchronization time in the A_FT parameter of the instruction.

Note

During the CiR synchronization period the outputs are frozen and the inputs are not evaluated.

Parameters

The following table shows the parameters of the "Control CiR process" instruction:

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Valid values on the MODE parameter are: • 0: Information function • 1: Enable CiR process • 2: Disable CiR process • 3: Disable CiR conditionally. You specify the high limit of the CiR synchronization time in FRZ_TIME.
FRZ_TIME	Input	TIME	I, Q, M, D, L or constant	High limit of the CiR synchronization time in milliseconds. Permitted range of values: 60 ... 2500 ms (default: 1000 ms) The FRZ_TIME parameter is only relevant when the value 3 is set on the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	Error information When the MODE parameter is "0", the RET_VAL parameter supplies information on the enabling of the CiR process.
A_FT	Output	TIME	I, Q, M, D, L	Currently valid high limit of the CiR synchronization time

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error This error code can be output only if the value "1", "2" or "3" is set on the MODE parameter.
0001	CiR is enabled. This error code can be output only if the value "0" is set on the MODE parameter.
0002	The CiR process is disabled. This error code can be output only if the value "0" is set on the MODE parameter.
0003	CiR is disabled conditionally. This error code can be output only if the value "0" is set on the MODE parameter.
8001	The CPU is not ready for a CiR operation. You are using an H CPU in an H-system (stand-alone), or you are working with a standard CPU in multicomputing mode.
8002	The value at the MODE parameter is invalid.
8003	The value at the FRZ_TIME parameter is invalid.
8xyy	You can find additional information here: Evaluating errors with output parameter RET_VAL (Page 162)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The "Control CiR process" instruction can be used, for example, to ensure that no CiR process is initiated during periods in which maximum CPU performance is required to control the process.

To use the "Control CiR process" for this, the instruction must be called in the program before the start of the period with increased process activity and executed with the value "2" (disable CiR process) for the MODE parameter. The CiR process is then disabled.

To enable the CiR process again after the completion of the period with increased process activity, call the "Control CiR process" instruction again and execute it with the value "1" for the MODE parameter. Alternatively, you can also set the value 3 on the MODE parameter to disable the CiR process conditionally.

See also

Overview of the valid data types (Page 231)

Setting status bits in program control operation instructions (Page 180)

STL Basics (Page 7245)

Word logic operations

DECO: Decode

Description

The instruction "Decode" sets a bit specified by the input value in the output value.

The "Decode" instruction reads the value of the IN parameter and sets the bit in the OUT parameter whose bit position corresponds to the value read. The other bits in the output value are filled with zeroes. When the value at the IN parameter is greater than 31, a modulo-32 operation is executed.

Parameters

The following table shows the parameters of the "Decode" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	WORD, INT	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set
OUT	Output	DWORD	I, Q, M, D, L, P	Output value

You can select the data type of the IN parameter from the "???" drop-down list.

Example

The following example shows how the instruction works:

STL	Explanation
CALL DECO	// The instruction is called.
	// Select the required data type from the "???" drop-down list.
IN := "Tag_Input"	// Input value that specifies the bit to be set in the output value
OUT := "Tag_Output"	// Result

The following table shows how the instruction functions using specific values:

Tag_Input	3	
	31 16	15 ... 3 ... 0
Tag_Output	0000 0000 0000 0000	0000 0000 0000 1000

The instruction reads bit number "3" from the value of the operand "Tag_Input" and sets the third bit to the value of the operand "Tag_Output".

See also

Overview of the valid data types (Page 231)

Setting the status bit for word logic operations (Page 182)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

ENCO: Encode

Description

The instruction "Encode" is used to read the bit number of the least significant set bit in the input value and to output it at the parameter OUT.

The "Encode" instruction selects the least significant bit of the value at the IN parameter and writes this bit number to the operand at the OUT parameter. If the IN parameter has the value DW#16#00000001 or DW#16#00000000, the value "0" is output at the OUT parameter.

Parameters

The following table shows the parameters of the "Encode" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL ENCO                    // The instruction is called.
IN := "Tag_Input"            // Input value the least significant bit of which is
                             read.
OUT := "Tag_Output"          // Bit number of least significant bit set in the in-
                             put value
```

The following table shows how the instruction functions using specific values:

	31 16	15 ... 3 ... 0
“Tag_Input“	0000 1111 0000 0101	0000 1001 0000 1000
“Tag_Output“	3	

The instruction selects the least significant set bit of the variable "Tag_Input" and writes its bit position "3" to the variable "Tag_Output".

See also

Overview of the valid data types (Page 231)

Setting the status bit for word logic operations (Page 182)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SEL: Select

Description

Depending on a switch (G input), the "Select" instruction selects one of the inputs, IN0 or IN1 and copies its content to the OUT output. When the G input has signal state "0", the value at the IN0 input is moved. If input G has signal state "1", the value at input IN1 is copied to output OUT.

The instruction is only executed if the tags of all parameters are of the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

Parameters	Declaration	Data type	Memory area	Description
G	Input	BOOL	I, Q, M, D, L	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Result

You can select the data type of the parameters IN0, IN1 and OUT from the "???" drop-down list.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SEL                     // The instruction is called.
                             // Select the required data type from the "???" drop-
                             down list.
G := "Tag_Input_G"           // Signal state of the switch
IN0 := "Tag_Input0"          // First input value
IN1 := "Tag_Input1"          // Second input value
OUT := "Tag_Output"          // Value of the selected input
```

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value
G	Tag_Input_G	1
IN0	Tag_Input0	W#16#0000
IN1	Tag_Input1	W#16#FFFF
OUT	Tag_Output	W#16#FFFF

See also

Overview of the valid data types (Page 231)

Setting the status bit for word logic operations (Page 182)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Additional instructions

SET: Set bit array

Description

You use the "Set bit array" instruction to set the signal state of the bits in a specified area to "1". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. You specify the number of bits to be set in the specified area in the N parameter. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Note

Parameter N

You can also interconnect an element of a data block at parameter N.

Parameters

The following table shows the parameters of the "Set bit array" instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be set

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SET                     // The "Set bit array" instruction is called
S_BIT := P#M0.0              // Pointer to the first bit to be set (M0.0)
N := 10                      // Number of bits to be set
```

In the example the instruction sets 10 bits in the range from M0.0 to M1.2.

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

SETP: Set a bit array in the I/O area

Description

The instruction "Set a bit array in the I/O area" sets the bits of an I/O area to the signal state "1". At the same time the corresponding bits in the process image outputs are also set parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be set. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output at the RET_VAL parameter.

The I/O area is set byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

  WARNING

  Unintended behavior

  The setting of the bits at the value of the relevant bit in the process image output can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are set to the signal state "1".

The instruction "Set a bit array in the I/O area" must be called in an MCR area.

Calling the "Set a bit array in the I/O area" instruction has no effect if one of the following conditions applies:

- The N parameter has the value "0".
- The MCR bit has the signal state "0".

Parameters

The following table shows the parameters of the "Set a bit array in the I/O area" instruction:

Parameters	Declaration	Data type	Memory area	Description
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be set
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be set

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	You can find additional information here: Evaluating errors with output parameter RET_VAL (Page 162)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SETP                    // The instruction is called.
N := 8                       // Number of bits to be set
RET_VAL := "Tag_ErrorCode"   // Error information
SA := P#10.0                 // Pointer to the first bit to be set (Q10.0)
```

In the example the instruction sets 8 bits in the I/O range corresponding to the outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

SETI: Set byte array

Description

The "Set byte array" instruction sets the signal state of the bits in a specified area of bytes to "1". You define the start point of the array on the S_BYTE. You define the size of the array by specifying the number of bits to be set on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer of

the S_BYTE parameter must point to an operand that corresponds to a byte limit value, i.e., the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output (Q) is reset to "0".

Calling the "Set byte array" instruction has no effect if one of the following conditions applies:

- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Parameters

The following table shows the parameters of the "Set byte array" instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P or constant	Points to the first byte of the array whose bits will be set For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array to be set The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SETI                    // The instruction is called.
S_BYTE := P#2.0              // Pointer to the first byte of the range (P2.0)
N := 16                      // Number of bits to be set
```

In the example the instruction sets 2 bytes (16 bits) in the range from P2.0 to P3.7 (I/O outputs).

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

RESET: Reset bit array

Description

The "Reset bit array" instruction is used to reset the signal state of the bits in a specified range to "0". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. You specify the number of bits to be reset in the specified range at the N parameter of the instruction. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Parameters

The following table shows the parameters of the "Reset bit array" instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Points to the first bit of the range whose bits will be reset. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be reset.

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL RESET                   // The instruction is called.
S_BIT := P#M0.0              // Pointer to the first bit (M0.0) to be reset
N := 10                      // Number of bits which are reset
```

In the example the instruction resets 10 bits in the range from M0.0 to M1.2.

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

RESETP: Reset a bit array in the I/O area

Description

The "Reset a bit array in the I/O area" instruction is used to reset the bits of an I/O area to signal state "0". At the same time the corresponding bits in the process image outputs are also reset parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be reset. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output at the RET_VAL parameter.

The I/O area is reset byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

  WARNING

  Unintended behavior

  The setting of the bits at the value of the relevant bit in the process image output can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are reset to the signal state "0".

The instruction "Reset a bit array in the I/O area" must be called in an MCR range.

Calling the "Reset a bit array in the I/O area" instruction has no effect if one of the following conditions applies:

- The N parameter has the value "0".
- The MCR bit has the signal state "0".

Parameters

The following table shows the parameters of the "Reset a bit array in the I/O area" instruction:

Parameters	Declaration	Data type	Memory area	Description
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be reset.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	You can find additional information here: Evaluating errors with output parameter RET_VAL (Page 162)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL RESETP                  // The instruction is called.
N := 8                       // Number of bits to be reset
RET_VAL := "Tag_ErrorCode"   // Error information
SA := P#10.0                 // Pointer to the first bit (Q10.0) to be reset
```

In the example the instruction sets 8 bits in the I/O area corresponding to the outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

RESETI: Reset byte array

Description

The "Reset byte array" instruction is used to reset the signal state of the bits in a specified range of bytes to "0". You use the pointer on the S_BYTE parameter to define the starting point of the range. You define the size of the range by specifying the number of bits to be reset on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer of the S_BYTE parameter must point to an operand that corresponds to a byte limit value, i.e., the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output (Q) is reset to "0".

Calling the "Reset byte array" instruction has no effect if one of the following conditions applies:

- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Parameters

The following table shows the parameters of the "Reset byte array" instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P or constant	Points to the first byte of the range whose bits will be reset. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array whose bits are to be reset. The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL RESETI                  // The instruction is called.
S_BYTE := P#2.0              // Pointer to the first byte of the range (P2.0)
N := 16                      // Number of bits to be reset
```

In the example the instruction resets 2 Bytes (16 bits) in the range from P2.0 to P3.7 (I/O outputs).

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

REPL_VAL: Enter substitute value

Description

Use the "Enter substitute value" instruction to enter a substitute value in accumulator 1 (ACCU 1) from a synchronous error organization block.

Note

You can only call the "Enter substitute value" instruction in a synchronous error OB (OB 121, OB 122).

Use the "Enter substitute value" instruction when you can no longer read any value from a module. The OB 122 is started after each access to this module. By calling the "Enter substitute value" instruction you can have the substitute value loaded in accumulator 1. In this case the program will then be processed with this substitute value. You can take the required information for the selection of the substitute value from the local variable of the OB 122.

Parameters

The following table shows the parameters of the "Enter substitute value" instruction:

Parameters	Declaration	Data type	Memory area	Description
VAL	Input	DWORD	I, Q, M, D, L or constant	Substitute value
RET_VAL	Output	INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Description
0000	No error
8080	The instruction was not called from a synchronous error OB.

Error code* (W#16#...)	Description
8xyy	You can find additional information here: Evaluating errors with output parameter RET_VAL (Page 162)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

DRUM: Implement sequencer

Description

You use the "Implement sequencer" instruction to assign the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD) with the programmed values of the OUT_VAL parameter of the relevant step. The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). When a bit of the configurable mask has a signal state of "1", the OUT_VAL value sets/resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the

corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	A signal state of "1" indicates a reset condition
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L	The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step.
EVENT(i), 1<= i <=16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P	Preset first step of the sequencer (1 to 16)
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P	Remaining processing time for the current step

Parameters	Declaration	Data type	Memory area	Description
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L or constant	System time of the previous call
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L	Preset counter value for each step [1 to 16] where one clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

```
STL                          Explanation
CALL DRUM, "DRUM_DB"         // The instruction is called and the instance data
                             block "DRUM_DB" is created.
RESET := "Tag_Reset"         // Reset input
JOG := "Tag_Input_Jog"       // On a rising edge, the instruction advances to the
                             next step
```

STL	Explanation
DRUM_EN := "Tag_Input_DrumEN"	// The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP := "Tag_Number_Last-Step"	// Number of the last programmed step
EVENT1 := "MyTag_Event_1"	// Event bit 1
EVENT2 := "MyTag_Event_2"	// Event bit 2
EVENT3 := "MyTag_Event_3"	// Event bit 3
EVENT4 := "MyTag_Event_4"	// Event bit 4
EVENT5 := "MyTag_Event_5"	// Event bit 5
EVENT6 := "MyTag_Event_6"	// Event bit 6
EVENT7 := "MyTag_Event_7"	// Event bit 7
EVENT8 := "MyTag_Event_8"	// Event bit 8
EVENT9 := "MyTag_Event_9"	// Event bit 9
EVENT10 := "MyTag_Event_10"	// Event bit 10
EVENT11 := "MyTag_Event_11"	// Event bit 11
EVENT12 := "MyTag_Event_12"	// Event bit 12
EVENT13 := "MyTag_Event_13"	// Event bit 13
EVENT14 := "MyTag_Event_14"	// Event bit 14
EVENT15 := "MyTag_Event_15"	// Event bit 15
EVENT16 := "MyTag_Event_16"	// Event bit 16
OUT1 := "MyTag_Output_1"	// Output bit 1
OUT2 := "MyTag_Output_2"	// Output bit 2
OUT3 := "MyTag_Output_3"	// Output bit 3
OUT4 := "MyTag_Output_4"	// Output bit 4
OUT5 := "MyTag_Output_5"	// Output bit 5
OUT6 := "MyTag_Output_6"	// Output bit 6
OUT7 := "MyTag_Output_7"	// Output bit 7
OUT8 := "MyTag_Output_8"	// Output bit 8
OUT9 := "MyTag_Output_9"	// Output bit 9
OUT10 := "MyTag_Output_10"	// Output bit 10
OUT11 := "MyTag_Output_11"	// Output bit 11
OUT12 := "MyTag_Output_12"	// Output bit 12
OUT13 := "MyTag_Output_13"	// Output bit 13
OUT14 := "MyTag_Output_14"	// Output bit 14
OUT15 := "MyTag_Output_15"	// Output bit 15
OUT16 := "MyTag_Output_16"	// Output bit 16
Q := "Tag_Output_Q"	// The signal state "1" indicates that the time for the last step has elapsed
OUT_WORD := "Tag_OutputWord"	// Word address to which the sequencer writes the output values
ERR_CODE := "Tag_ErrorCode"	// Error information

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE

Parameters	Address	Value
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameters	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE

Parameters	Operand	Address	Value
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE

Parameters	Address	Value
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

DRUM_X: Implement sequencer

Description

You use the "Implement sequencer" instruction to implement a sequencer with a maximum of 16 steps. You specify the number of the first step on the DSP parameter. You specify the number of the last step on the LST_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT_WORD (in which the output bits are collected together) are written. An output bit is assigned either the corresponding bit of the OUT_VAL array that you specified or the output bit of the preceding step. The value assigned depends on the value of the mask bit on the S_MASK parameter.

Note

The default of the mask bits is 0. If you want to change the current setting of one or more mask bits, make this change in the instance data block.

The "Implement sequencer" instruction advances to the next step when there is a rising signal edge on the JOG parameter. If the instruction is already in the last step, the Q and EOD tags are set when the JOG parameter has a positive signal edge. In this case the DCC parameter is set to the value "0". The instruction remains in the last step until the RESET parameter is set to the value "1".

You can also assign parameters so that switching to the next step is time dependent. To do this, the DRUM_EN parameter has to be set to the value "1". The transition to the next step is made when the event bit (EVENTi) belonging to the current step is set and the time specified for the current step has expired. The time for a step is the product of the (DTBP) time base and the time factor valid for the current step (S_PRESET).

Note

The execution time (DCC) remaining in the current step is reduced only when the corresponding event bit (EVENTi) is set.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single

instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

If the value "1" is set in the RESET parameter when the instruction is called, the sequencer goes to the step whose number is specified in the DSP parameter.

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1".

You can also move the next step at any time via the value at the JOG parameter.

When the instruction is called for the first time, you must set the value "1" on the RESET parameter.

When the sequencer is in the last step and the execution time specified for this step has expired, the parameters Q and EOD are set. The instruction remains in the last step until the RESET parameter changes to the value "1".

The instruction is executed only in the STARTUP and RUN modes.

The operating system resets the "Implement sequencer" instruction during a cold restart. During a warm restart, the instruction is not reset. If you want the "Implement sequencer" instruction to be initialized after a warm restart, you must call the instruction in OB 100 with the value "1" in the RESET parameter.

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" resets the sequencer. When the instruction is called for the first time, the RESET parameter must be set to "1".
JOG	Input	BOOL	I, Q, M, D, L	A rising edge (compared to the last call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you assigned to the DRUM_EN parameter.
DRUM_EN	Input	BOOL	I, Q, M, D, L	Control parameter that specifies whether time-dependent switching to the next step is possible. If the value is "1" a time-dependent switching to the next step is possible.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last step Values from 1 to 16 are permitted.

Parameters	Declaration	Data type	Memory area	Description
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i) (belongs to step i)
OUTj, 0 ≤ j ≤ 15	Output	BOOL	I, Q, M, D, L	Output bit (j) (identical with bit no. j of OUT_WORD)
Q	Output	BOOL	I, Q, M, D, L	Status parameter that indicates whether the specified execution time of the last step has expired.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Output bits collected together in a tag
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	Not relevant to user (input parameter JOG of the preceding call)
EOD	Static	BOOL	I, Q, M, D, L	Identical with output parameter Q
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Number of the first step Values from 1 to 16 are permitted.
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Number of the current step
DCC	Static	DWORD	I, Q, M, D, L, P	The execution time still remaining in the current step in milliseconds (ms) (only relevant if DRUM_EN = 1 and the corresponding event bit = 1)
DTBP	Static	WORD	I, Q, M, D, L, P or constant	The time base valid for all steps in milliseconds (ms)
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	(Not relevant for the user: system time of the previous call)
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	One-dimensional array with the time factors for each step. Recommended selection of the indices: [1 to 16] In this case, S_PRESET [x] has the time factor of step x.

Parameters	Declaration	Data type	Memory area	Description
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the values to be output in each step if they have not been masked out using S_MASK. Recommended selection of the indices: [1 to 16, 0 to 15] In this case, OUT_VAL [x, y] has the value assigned to the output bit OUTy in step x.
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the mask bits for each step. Recommended selection of the indices: [1 to 16, 0 to 15] In this case, S_MASK [x, y] contains the mask bit for the "y"the value to be output in step x. Meaning of the mask bits: • 0: The corresponding value of the previous step is assigned to the relevant output bit. • 1: The corresponding value from OUT_VAL is assigned to the relevant output bit.

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE* (W#16#...)	Explanation
0000	No error
8081	Invalid value at parameter LST_STEP
8082	Invalid value on parameter DSC
8083	Invalid value on parameter DSP
8084	The product DCC = DTBP * S_PRESET[DSC] exceeds the value 2**31-1 (approximately 24.86 days).
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT0 to OUT15) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

STL	Explanation
CALL DRUM_X, "DRUM_X_DB"	// The instruction is called and the instance data block "DRUM_DB" is created.
RESET := "Tag_Reset"	// Reset input
JOG := "Tag_Input_Jog"	// On a rising edge, the instruction advances to the next step
DRUM_EN := "Tag_Input_DrumEN"	// The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP := "Tag_Number_Last-Step"	// Number of the last programmed step
EVENT1 := "MyTag_Event_1"	// Event bit 1
EVENT2 := "MyTag_Event_2"	// Event bit 2
EVENT3 := "MyTag_Event_3"	// Event bit 3
EVENT4 := "MyTag_Event_4"	// Event bit 4
EVENT5 := "MyTag_Event_5"	// Event bit 5
EVENT6 := "MyTag_Event_6"	// Event bit 6
EVENT7 := "MyTag_Event_7"	// Event bit 7
EVENT8 := "MyTag_Event_8"	// Event bit 8
EVENT9 := "MyTag_Event_9"	// Event bit 9
EVENT10 := "MyTag_Event_10"	// Event bit 10
EVENT11 := "MyTag_Event_11"	// Event bit 11
EVENT12 := "MyTag_Event_12"	// Event bit 12
EVENT13 := "MyTag_Event_13"	// Event bit 13
EVENT14 := "MyTag_Event_14"	// Event bit 14
EVENT15 := "MyTag_Event_15"	// Event bit 15
EVENT16 := "MyTag_Event_16"	// Event bit 16
OUT0 := "MyTag_Output_0"	// Output bit 0
OUT1 := "MyTag_Output_1"	// Output bit 1
OUT2 := "MyTag_Output_2"	// Output bit 2
OUT3 := "MyTag_Output_3"	// Output bit 3
OUT4 := "MyTag_Output_4"	// Output bit 4
OUT5 := "MyTag_Output_5"	// Output bit 5
OUT6 := "MyTag_Output_6"	// Output bit 6
OUT7 := "MyTag_Output_7"	// Output bit 7
OUT8 := "MyTag_Output_8"	// Output bit 8
OUT9 := "MyTag_Output_9"	// Output bit 9

```
STL                              Explanation
OUT10 := "MyTag_Output_10"       // Output bit 10
OUT11 := "MyTag_Output_11"       // Output bit 11
OUT12 := "MyTag_Output_12"       // Output bit 12
OUT13 := "MyTag_Output_13"       // Output bit 13
OUT14 := "MyTag_Output_14"       // Output bit 14
OUT15 := "MyTag_Output_15"       // Output bit 15
Q := "Tag_Output_Q"              // The signal state "1" indicates that the time for
                                 the last step has elapsed
OUT_WORD := "Tag_OutputWord"     // Word address to which the sequencer writes the out-
                                 put values
ERR_CODE := "Tag_ErrorCode"      // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_X_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE

Parameters	Address	Value
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE

Parameters	Address	Value
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameters	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT0	MyTag_Output_0	M4.0	TRUE
OUT1	MyTag_Output_1	M4.1	TRUE
OUT2	MyTag_Output_2	M4.2	TRUE
OUT3	MyTag_Output_3	M4.3	TRUE
OUT4	MyTag_Output_4	M4.4	TRUE
OUT5	MyTag_Output_5	M4.5	TRUE
OUT6	MyTag_Output_6	M4.6	TRUE
OUT7	MyTag_Output_7	M4.7	TRUE
OUT8	MyTag_Output_8	M5.0	TRUE
OUT9	MyTag_Output_9	M5.1	TRUE
OUT10	MyTag_Output_10	M5.2	TRUE
OUT11	MyTag_Output_11	M5.3	TRUE
OUT12	MyTag_Output_12	M5.4	TRUE
OUT13	MyTag_Output_13	M5.5	TRUE
OUT14	MyTag_Output_14	M5.6	TRUE
OUT15	MyTag_Output_15	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Address	Value
OUT0	MyTag_Output_0	M4.0	TRUE
OUT1	MyTag_Output_1	M4.1	FALSE
OUT2	MyTag_Output_2	M4.2	FALSE
OUT3	MyTag_Output_3	M4.3	FALSE
OUT4	MyTag_Output_4	M4.4	FALSE
OUT5	MyTag_Output_5	M4.5	TRUE
OUT6	MyTag_Output_6	M4.6	FALSE
OUT7	MyTag_Output_7	M4.7	FALSE
OUT8	MyTag_Output_8	M5.0	TRUE
OUT9	MyTag_Output_9	M5.1	TRUE
OUT10	MyTag_Output_10	M5.2	FALSE
OUT11	MyTag_Output_11	M5.3	FALSE

Parameters	Operand	Address	Value
OUT12	MyTag_Output_12	M5.4	FALSE
OUT13	MyTag_Output_13	M5.5	FALSE
OUT14	MyTag_Output_14	M5.6	TRUE
OUT15	MyTag_Output_15	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are saved in the "DRUM_X_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

DCAT: Discrete control-timer alarm

Description

You can use the "Discrete control-timer alarm" to count the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following effects on the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET, OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET, OA, CA and CMD_HIS are reset to "0".
- When the signal state of the CMD and CMD_HIS parameters is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state at the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- When the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameters	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing

Parameters	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where one clock pulse = 1 ms.
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
CMD_HIS	Static	BOOL	I, Q, M, D, L	CMD history bit

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The CMD_HIS parameter of the instance data block is set to the signal state "1" and the ET parameter is reset to "0".

Note

You can initialize static parameters in the data block.

```
STL                          Explanation
CALL DCAT, "DCAT_DB"         // The instruction is called and the instance data
                             block "DCAT_DB" is created.
CMD := "Tag_Input_CMD"       // Opening or closing of device
O_FB := "Tag_Input_O_FB"     // Feedback when opening
C_FB := "Tag_Input_C_FB"     // Feedback when closing
Q := "Tag_Output_Q"          // Shows the status of the parameter CMD
OA := "Tag_Output_OA"        // Alarm output when opening
CA := "Tag_Output_CA"        // Alarm output when closing
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

MCAT: Motor control-timer alarm

Description

The "Motor control-timer alarm" instruction is used to accumulate the time from the point at which one of the command inputs (opening or closing) is switched on. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

When the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to the signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore not possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameters	Declaration	Data type	Memory area	Description
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input

Parameters	Declaration	Data type	Memory area	Description
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
O_HIS	Static	BOOL	I, Q, M, D, L	"Open" history bit
C_HIS	Static	BOOL	I, Q, M, D, L	"Close" history bit

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL MCAT, "MCAT_DB"             // The instruction is called and the instance data
                                 block "MCAT_DB" is created.
O_CMD := "Tag_Input_O_CMD"       // "Open" command input
C_CMD := "Tag_Input_C_CMD"       // "Close" command input
S_CMD := "Tag_Input_S_CMD"       // "Stop" command input
O_FB := "Tag_Input_O_FB"         // Feedback when opening
C_FB := "Tag_Input_C_FB"         // Feedback when closing
OO := "Tag_OutputOpen"           // "Open" output
CO := "Tag_OutputClosed"         // "Close" output
OA := "Tag_Output_OA"            // Alarm output when opening
CA := "Tag_Output_CA"            // Alarm output when closing
Q := "Tag_Output_Q"              // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

IMC: Compare input bits with the bits of a mask

Description

You can use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.

Parameters	Declaration	Data type	Memory area	Description
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

In the following example all 16 input bits are compared with the mask for step 2. The signal state of the OUT parameter is set to TRUE, since the input bits match the mask for step 2.

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL IMC, "IMC_DB"               // The instruction is called and the instance data
                                 block "IMC_DB" is created.
IN_BIT0 := "Tag_Input_BIT0"      // Input bit 0
IN_BIT1 := "Tag_Input_BIT1"      // Input bit 1
IN_BIT2 := "Tag_Input_BIT2"      // Input bit 2
IN_BIT3 := "Tag_Input_BIT3"      // Input bit 3
IN_BIT4 := "Tag_Input_BIT4"      // Input bit 4
IN_BIT5 := "Tag_Input_BIT5"      // Input bit 5
IN_BIT6 := "Tag_Input_BIT6"      // Input bit 6
IN_BIT7 := "Tag_Input_BIT7"      // Input bit 7
IN_BIT8 := "Tag_Input_BIT8"      // Input bit 8
IN_BIT9 := "Tag_Input_BIT9"      // Input bit 9
IN_BIT10 := "Tag_Input_BIT10"    // Input bit 10
IN_BIT11 := "Tag_Input_BIT11"    // Input bit 11
IN_BIT12 := "Tag_Input_BIT12"    // Input bit 12
IN_BIT13 := "Tag_Input_BIT13"    // Input bit 13
IN_BIT14 := "Tag_Input_BIT14"    // Input bit 14
IN_BIT15 := "Tag_Input_BIT15"    // Input bit 15
CMP_STEP := "Tag_CMP_STEP"       // Step number of the mask
OUT := "Tag_Output"              // Indicates whether or not a match was found
ERR_CODE := "Tag_ErrorCode"      // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE

Parameters	Operand	Value
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
CMP_STEP	Tag_CMP_STEP	B#16#02
OUT	Tag_Output	FALSE
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the instance data block "IMC_DB" of the instruction:

Parameters	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output	TRUE
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

SMC: Compare scan matrix

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.

Parameters	Declaration	Data type	Memory area	Description
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY of WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

In this example, all 16 input bits are compared with the mask for steps 0 through 5 until a match is found. Only the masks for steps 0 through 2 are scanned, since the mask for step 2 matches the input bit.

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL SMC, "SMC_DB"               // The instruction is called and the instance data
                                 block "SMC_DB" is created.
IN_BIT0 := "Tag_Input_BIT0"      // Input bit 0
IN_BIT1 := "Tag_Input_BIT1"      // Input bit 1
IN_BIT2 := "Tag_Input_BIT2"      // Input bit 2
IN_BIT3 := "Tag_Input_BIT3"      // Input bit 3
IN_BIT4 := "Tag_Input_BIT4"      // Input bit 4
IN_BIT5 := "Tag_Input_BIT5"      // Input bit 5
IN_BIT6 := "Tag_Input_BIT6"      // Input bit 6
IN_BIT7 := "Tag_Input_BIT7"      // Input bit 7
IN_BIT8 := "Tag_Input_BIT8"      // Input bit 8
IN_BIT9 := "Tag_Input_BIT9"      // Input bit 9
IN_BIT10 := "Tag_Input_BIT10"    // Input bit 10
IN_BIT11 := "Tag_Input_BIT11"    // Input bit 11
IN_BIT12 := "Tag_Input_BIT12"    // Input bit 12
IN_BIT13 := "Tag_Input_BIT13"    // Input bit 13
IN_BIT14 := "Tag_Input_BIT14"    // Input bit 14
IN_BIT15 := "Tag_Input_BIT15"    // Input bit 15
OUT := "Tag_Output"              // Indicates whether or not a match was found.
OUT_STEP := "Tag_Output_STEP"    // Contains the step number with the relevant mask.
ERR_CODE := "Tag_ErrorCode"      // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
OUT	Tag_Output	FALSE
OUT_STEP	Tag_Output_STEP	B#16#00
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the instance data block "SMC_DB" of the instruction:

Parameters	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE
LAST	DB84	B#16#05

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output	TRUE
OUT_STEP	Tag_Output_STEP	B#16#02
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

LEAD_LAG: Lead and lag algorithm

Description

You can use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the instruction "Lead and lag algorithm" in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single

instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

```
STL                                Explanation
CALL LEAD_LAG, "LEAD_LAG_DB"       // The instruction is called and the instance data
                                   block "LEAD_LAG_DB" is created.
IN := "Tag_Input"                  // The input value of the current sample time (cycle
                                   time) to be processed.
SAMPLE_T := "Tag_In-               // Sample time
put_SAMPE_T"
OUT := "Tag_Output_Result"         // Result of the instruction
ERR_CODE := "Tag_ErrorCode"        // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameters	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameters	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameters	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

Master Control Relay (Page 191)

TONR_X: Time accumulator

Description

You use the "Time accumulator" instruction to accumulate time values within a period which you set using the PV input. The instruction is executed when the signal state on the TMR_EN start input changes from "0" to "1". Simultaneously, the timer runs for the duration set at the PV input. While the instruction is being executed, it adds the time value of the DELTA_T input to the time value of the InOut ET.

The following conditions must be met:

- The TMR_EN start input has the signal state "1".
- The time value at InOut ET is less than the time value at the PV parameter.
- The RESET input has the signal state "0".

The instruction is interrupted when the signal state on the TMR_EN start input changes from "1" to "0". As soon as the signal state changes back to "1", the instruction resumes execution.

The instruction is ended when the time value at InOut ET has reached the time value at the PV input. The Q output then has the signal state "1".

As long as the Q output has the signal state "1", the time value at InOut ET remains unchanged. The time value and the Q output are not reset to "0" until the RESET input is set to signal state "1".

Since the "Time accumulator" instruction uses the cycle time (DELTA_T) of the last cycle of the organization block (OB), the instruction may only be called in cyclic organization blocks.

Note

You have to move the cycle time of the organization block from the startup information to the global tag of the DELTA_T parameter.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type	Memory area	Description
TMR_EN (Timer Enable)	Input	BOOL	I, Q, M, D, L	Start input Starts the instruction and the expiration of the preset time duration at the PV input
RESET	Input	BOOL	I, Q, M, D, L	Reset input
PV (Preset Value)	Input	DINT	I, Q, M, D, L, P or constant	Maximum duration of time recording

Parameters	Declaration	Data type	Memory area	Description
DELTA_T	Input	INT	I, Q, M, D, L or constant	Cycle time of organization block of the previous cycle
Q	Output	BOOL	I, Q, M, D, L	Output which is set when the preset time value is reached
ET (Elapsed Time)	InOut	DINT	I, Q, M, D, L	Accumulated time value (Currently elapsed time value + time value of the DELTA_T input)

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:

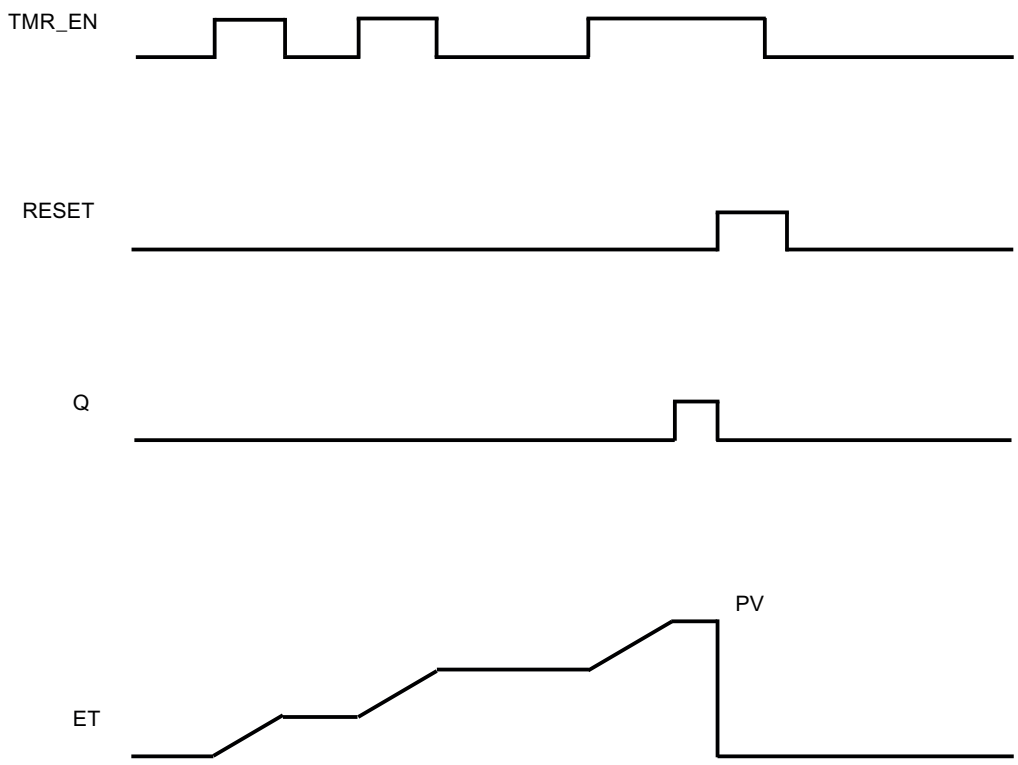

Example

The following example shows how the instruction works:

STL	Explanation
`CALL TONR_X`	`// The instruction is called.`
`TMR_EN := "Tag_Start"`	`// Start input`
`RESET := "Tag_Reset"`	`// Reset input`
`PV := 10`	`// Maximum duration of time recording`

STL	Explanation
DELTA_T := 2	// Cycle time of organization block of the previous cycle
ET := "Tag_Elapsed_Time"	// Accumulated time
Q := "Tag_Output"	// Indicates whether the time set for the PV parameter has been reached.

See also

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

WSR: Save data to shift register

Description

The "Save data to shift register" instruction is used to write a bit from a specified source to a shift register. Each element of the register is shifted into the next address. The data contained in the last address of the shift register is lost after the shift.

Each time the instruction is executed with the value "0" in the RESET parameter, new data are read from the source (S_DATA) and shifted to the start address (START) of the shift register. The LEN parameter specifies the number of elements to be shifted. You specify the data type of the elements on the E_TYPE parameter. The instruction is not executed if an invalid data type is specified on the E_TYPE parameter.

The register is deleted if the RESET parameter is set to "1" when the instruction is executed. The Q parameter is set to the signal state "1" when the shift register is filled with zeros or deleted.

Parameters

The following table shows the parameters of the "Save data to shift register" instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_DATA	Input	POINTER	I, Q, M, D	Points to the source data element to be shifted to the start address (START). For area-crossing, indirect register addressing, the pointer must be specified in double-word format.

Parameters	Declaration	Data type	Memory area	Description
START	Input	POINTER	I, Q, M, D	Points to the start address of the register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
LEN	Input	WORD	I, Q, M, D, L, P	Number of elements to be shifted.
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the elements to be shifted. The following data types are permitted: • W#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
Q	Output	BOOL	Q, M, D, L	Is set to "1" if RESET parameter is active (1) or all elements to be shifted have the value "0".

Example

In the following example four elements of the data type WORD are shifted starting from the address specified on the START parameter. When the first element of the register has been shifted to the next address, the first address is filled with the data that the pointer references on the S_DATA parameter. The value of the last element in the register is lost in the process.

```
STL                                Explanation
CALL WSR                           // The instruction is called.
RESET := "Tag_Input_Reset"         // Reset input
S_DATA := "Tag_Input_Data-         // Source element that is shifted to the start address.
Source"
START := "Tag_Input_StartAd-       // Start address
dress"
LEN := "Tag_Input_LEN"             // Number of elements to be shifted
E_TYPE := "Tag_Input_Element-      // Data type of the elements
Type"
Q := "Tag_Output_Q"                // Status of the register
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_SourceData	P#DB2.DBX6.0 DW6 = W#16#1234
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW0 = W#16#1111 DW2 = W#16#2222 DW4 = W#16#3333 DW6 = W#16#4444
LEN	Tag_Input_LEN	W#16#04
E_TYPE	Tag_Input_ElementType	W#16#04
Q	Tag_Output_Q	1

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_SourceData	P#DB2.DBX6.0 DW6 = W#16#1234
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW0 = W#16#1234 DW2 = W#16#1111 DW4 = W#16#2222 DW6 = W#16#3333
LEN	Tag_Input_LEN	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

See also

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

SHRB: Shift bit to shift register

Description

The "Shift bit to shift register" instruction is used to write a bit from a specified source (DATA) to a shift register. Each time the instruction is executed with a value "0" on the RESET parameter, new data are read from the source and shifted to the start address (S_BIT) of the shift register. All following bits are shifted by one. The bit contained at the last address (S_BIT + N) is lost after the shift. If the RESET parameter is set to "1" during the processing of the instruction, the addresses in the shift register are reset to "0" and not shifted.

Parameters

The following table shows the parameters of the "Shift bit to shift register" instruction:

Parameters	Declaration	Data type	Memory area	Description
DATA	Input	BOOL	I, Q, M, D, L	Source bit
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_BIT	Input	POINTER	I, Q, M, D	Points to the start bit of the shift register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	WORD	I, Q, M, D, L, P	Number of bits to be shifted.

Example

In the following example the N parameter is set to the value 14 ("E" in hexadecimal). As a result 14 bits starting from the address specified on the S_BIT parameter are shifted. After the bits have been shifted, the first address is filled with the data from the source on the DATA parameter. The last bit value is lost.

```
STL                          Explanation
CALL SHRB                    // The instruction is called.
DATA := "Tag_Input_Data"     // Source bit that is shifted to start address
RESET := "Tag_Input_Reset"   // Reset input
S_BIT := "Tag_Input_Bit-     // Start address
Source"
N := "Tag_Input_Number"      // Number of bits to be shifted
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
RESET	Tag_Input_Reset	0
DATA	Tag_Input_Data	0
S_BIT	Tag_Input_BitSource	P#DB1.DBX1.4
N	Tag_Input_Number	B#16#0E

Before processing the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	0	1	1	1	0	1	0

2.7	...					...	2.0
0	0	1	1	1	0	1	0

3.7	...					...	3.0
0	0	1	1	1	0	1	0

After processing

After the processing of the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	1	1	0	1	0	1	0

2.7	...					...	2.0
0	1	1	1	0	1	0	0

3.7	...					...	3.0
0	0	1	1	1	0	0	0

See also

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

SEG: Create bit pattern for seven-segment display

Description

The "Create bit pattern for seven-segment display" instruction is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a 7-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)
0000	00111111	0
0001	00000110	1
0010	01011011	2
0011	01001111	3
0100	01100110	4
0101	01101101	5
0110	01111101	6
0111	00000111	7
1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	B
1100	00111001	C
1101	01011110	D
1110	01111001	E
1111	01110001	F

Seven-segment display

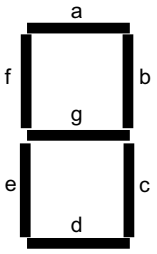

Parameters

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SEG                     // The instruction is called.
IN := "Tag_Input"            // Source word
OUT := "Tag_Output"          // Bit pattern for the seven-segment display
```

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16#065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

BCDCPL: Create tens complement

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL BCDCPL                  // The instruction is called.
IN := "Tag_Input"            // 7-digit BCD number
ERR_CODE := "Tag_Output"     // Tens complement of the specified BCD number
```

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)		

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output at the RET_VAL parameter.

Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL BITSUM                  // The instruction is called.
IN := "Tag_Input"            // Operand whose set bits are counted
```

STL	Explanation
RET_VAL := "Tag_Output"	// Number of set bits of the operand "Tag_Input"

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)		

See also

Overview of the valid data types (Page 231)

Setting status bits (Page 175)

STL Basics (Page 7245)

STL Mnemonic

Bit logic operations

A: AND logic operation"

Description

Use the instruction "AND logic operation" to query whether the signal state of a binary operand is "1" and to AND the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand.

After execution of the instruction the RLO will be "1" if both linked signal states are also "1". If one of the linked signal states is "0", the RLO is "0" after executing the instruction.

If you use the instruction multiple times consecutively, all query results must be "1" for the common result of the logic operation to also be "1".

You can also use the instruction "AND logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "AND logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program		Relay circuit diagram	
		Power rail	
A	“Tag_Input_1“	Signal state 1:	Normally open contact
A	“Tag_Input_2“	Signal state 1:	Normally open contact
=	“Tag_Output“	Signal state 1:	Coil
Shows closed switch			

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

AN: Negated AND logic operation

Description

Use the instruction "Negated AND logic operation" to query whether the signal state of a binary operand is "0" and to AND the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. If the operand has signal state "0," the result of the query is "1". If the operand has signal state "1," the result of the query is "0".

After the instruction executes, the RLO is "1" if the result of the query and the RLO before the execution of the instruction have signal state "1".

After the instruction executes, the RLO is "0" if the result of the query or the RLO before the execution of the instruction have signal state "0".

You can also use the instruction "Negated AND logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "Negated AND logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program		Relay circuit diagram	
		Power rail	
A	“Tag_Input_1“	Signal state 0:	Normally open contact
AN	“Tag_Input_2“	Signal state 1:	Normally closed contact
=	“Tag_Output“	Signal state 0:	Coil

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

O: OR logic operation

Description

Use the instruction "OR logic operation" to query whether the signal state of a binary operand is "1" and to OR the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand.

After the instruction executes, the RLO is "1" if one of the linked signal states is "1". If both linked signal states are "0", the RLO is "0" after the instruction executes.

If you use the instruction multiple times consecutively, it is sufficient that one of the results of the query is "1" for the common result of the logic operation to also be "1".

You can also use the instruction "OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
O “Tag_Input_1“	Signal state 1 normally open contact (“Tag_Input_1“) / Signal state 0 normally open contact (“Tag_Input_2“)
O “Tag_Input_2“	
= “Tag_Output“	Signal state 1: Coil
	Shows closed switch

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

ON: Negated OR logic operation

Description

Use the instruction "Negated OR logic operation" to query whether the signal state of a binary operand is "0" and to OR the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. If the operand has signal state "0," the result of the query is "1". If the operand has signal state "1," the result of the query is "0".

After the instruction executes, the RLO is "1" if the result of the query or the RLO before the execution of the instruction have signal state "1". If both linked signal states are "0", the RLO is also "0" after the instruction executes.

If you use the instruction multiple times consecutively, it is sufficient that one of the results of the query is "1" for the common result of the logic operation to also be "1".

You can also use the instruction "Negated OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "Negated OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is checked

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
O “Tag_Input_1“	Signal state 0
ON “Tag_Input_2“	Signal state 1
= “Tag_Output“	Signal state 1

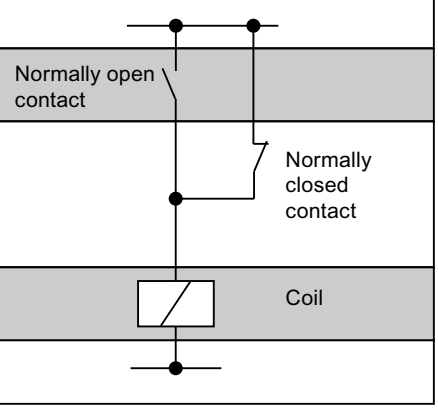

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

X: EXCLUSIVE OR logic operation

Description

Use the instruction "EXCLUSIVE OR logic operation" to query whether the signal state of a binary operand is "1" and to XOR the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand. After the instruction EXECUTES, the RLO is "1" if the linked signal states are not equal. The RLO is "0" if both signal states are the same.

If, for example, the RLO before execution of the instruction is "1" and the queried operand is "1", the RLO is reset to "0". If the RLO is "1" and the operand has signal state "0", the RLO is set to "1" after the instruction executes.

If you use the instruction several times consecutively, it is sufficient that one of the query results is "1" for the common RLO to be "1" as well. If all query results are "1" or "0" the RLO is reset to "0".

You can also use the instruction "EXCLUSIVE OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
X “Tag_Input_1“	Contact
X “Tag_Input_2“	Contact
= “Tag_Output“	Coil

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

XN: Negated EXCLUSIVE OR logic operation

Description

Use the "Negated EXCLUSIVE OR logic operation" instruction to query whether the signal state of a binary operand is "0" and to OR the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. After the instruction EXECUTES, the RLO is "1" if the linked signal states are not equal. The RLO is "0" if both signal states are the same.

If, for example, the queried operand has signal state "0," the result of the query is "1". XORing the query result with an RLO "1" results in signal state "0". The logic operation with an RLO "0" results in signal state "1".

You can also use the "Negated EXCLUSIVE OR logic operation" instruction to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Parameters

The following table shows the parameters of the "Negated EXCLUSIVE OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program		Relay circuit diagram	
		Power rail	
X	“Tag_Input_1“	Contact	
XN	“Tag_Input_2“	Contact	
=	“Tag_Output“	Coil	

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

O: OR logic operation of AND functions

Description

Use the instruction "OR logic operation of AND functions" to execute several AND logic operations before an OR logic operation. The overall result of the AND logic operations is stored and ORed with the current result of logic operation (RLO) and to any other signal queries.

Example

The following example shows how the instruction works:

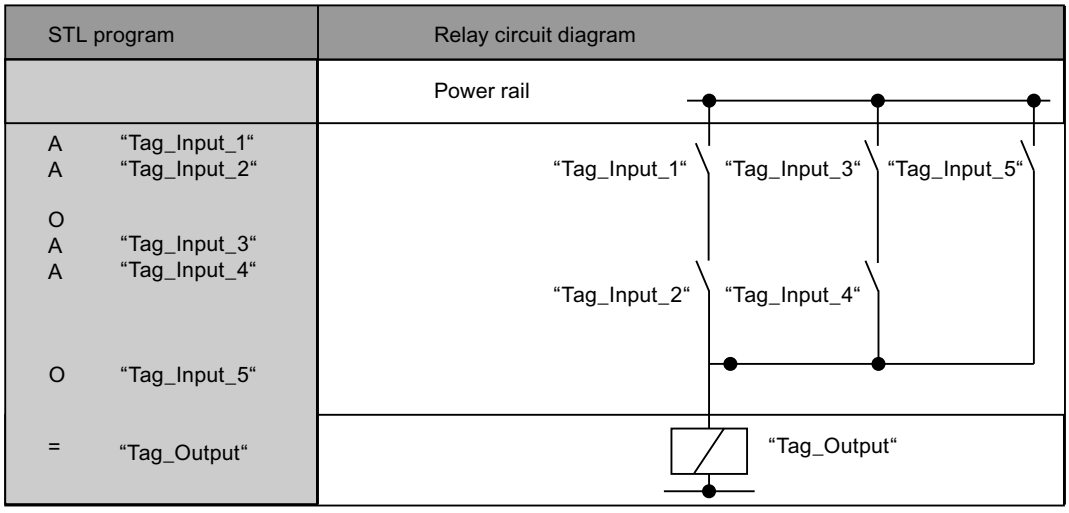

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

A(: AND logic operation with branch

Description

Use the instruction "AND logic operation with branch" to execute the instructions of a parenthesized expression before an AND logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. The saved RLO is ANDed with the result of logic operation of the complete nested expression. If additional signal queries follow after the nested expression, these are also ANDed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

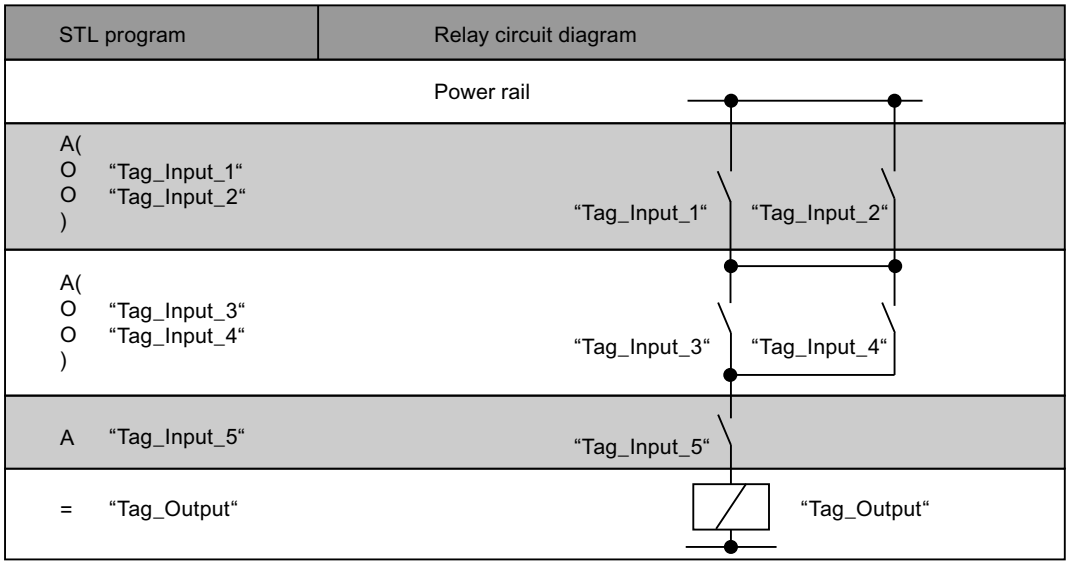

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

AN(: Negated AND logic operation with branch

Description

Use the "Negated AND logic operation with branch" instruction to execute the instructions of a parenthesized expression before an AND logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. After the instructions in the parenthesized expression have executed, the RLO of the complete parenthesized expression is negated and ANDed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also ANDed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

STL	Explanation
AN(	// Start of the nested expression
	// AND the RLO of the parenthesized expression and negate the complete result
O "Tag_Input_1"	// Query the signal state of the operand for "1" and OR with the current RLO.
O "Tag_Input_2"	// Query the signal state of the operand for "1" and OR with the current RLO.
)	// End of the nested expression
AN "Tag_Input_3"	// Query the signal state of the operand for "0" and AND with the current RLO.
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

O(: OR logic operation with branch

Description

Use the instruction "OR logic operation with branch" to execute the instructions of a parenthesized expression before an OR logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. After the instructions contained in the parenthesized expression have executed, the saved RLO is ORed with the RLO of the complete parenthesized expression. If additional signal queries follow after the parenthesized expression, these are also ORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
O(	// Start of the nested expression
	// OR the RLO of the nested expression
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

ON(: Negated OR logic operation with branch

Description

Use the "Negated OR logic operation with branch" instruction to execute the instructions of a parenthesized expression before an OR logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. After the instructions in the parenthesized expression have executed, the RLO of the complete parenthesized expression is negated and ORed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also ORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
ON(	// Start of the nested expression
	// Link RLO of the nested expression by OR logic and negate the complete result.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

X(: EXCLUSIVE OR logic operation with branch

Description

Use the instruction "EXCLUSIVE OR logic operation with branch" to execute the instructions of a parenthesized expression before an EXCLUSIVE OR logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. After the instructions in the parenthesized expression have executed, the saved RLO is XORed with the RLO of the complete parenthesized expression. If additional signal queries follow after the parenthesized expression, these are also XORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

STL	Explanation
X(	// Start of the nested expression
	// Link RLO of the nested expression by EXCLUSIVE OR logic
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
X "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

XN(: Negated EXCLUSIVE OR logic operation with branch

Description

Use the "Negated EXCLUSIVE OR logic operation with branch" instruction to execute the instructions of a parenthesized expression before an EXCLUSIVE OR logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. After the instructions contained in the parenthesized expression have been executed, the RLO of the complete parenthesized expression will be negated and EXCLUSIVE ORed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also XORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Example

The following example shows how the instruction works:

STL	Explanation
XN(	// Start of the nested expression
	// Link RLO of the nested expression by EXCLUSIVE OR logic and negate the complete result
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
XN "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Querying status bits with bit logic operation instructions (Page 183)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

): Close branch

Description

Use the "Close branch" instruction to specify the end of a parenthesized expression. After processing of the instruction the complete result of the nested expression is linked according to the instruction that was used to open the nested expression.

Use the following instructions to open a nested expression:

- A(: AND logic operation with branch
- AN(: Negated AND logic operation with branch
- O(: OR logic operation with branch
- ON(: Negated OR logic operation with branch
- X(: EXCLUSIVE OR logic operation with branch
- XN(: Negated EXCLUSIVE OR logic operation with branch

After the "Close branch" instruction, the CPU sets the binary result BR to the signal state that it had before the parenthesized expression.

Example

The following example shows how the instruction works:

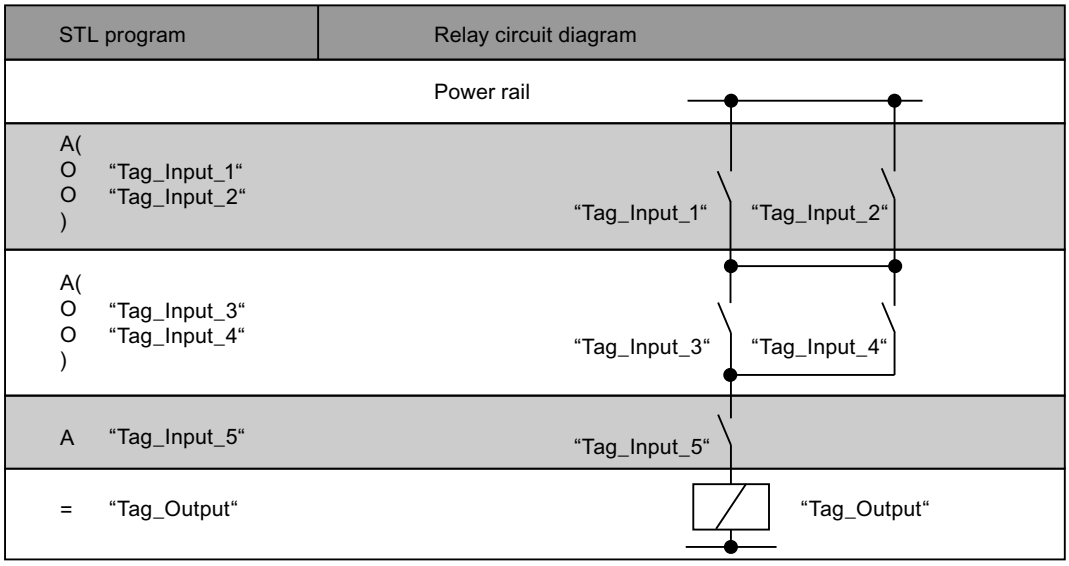

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Querying status bits in STL (Page 183)

=: Assignment

Description

Use the "Assignment" instruction to assign the signal state of a result of logic operation (RLO) saved in the CPU to a specified operand. If the RLO has signal state "1", the operand is set. If the signal is "0", the operand is reset to "0".

The execution of the "Assignment" instruction is influenced by the Master Control Relay (MCR). When the MCR bit is set to "0", the specified operand is set to signal state "0".

Parameters

The following table shows the parameters of the "Assignment" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the current RLO is assigned.

Example

The following example shows how the instruction works:

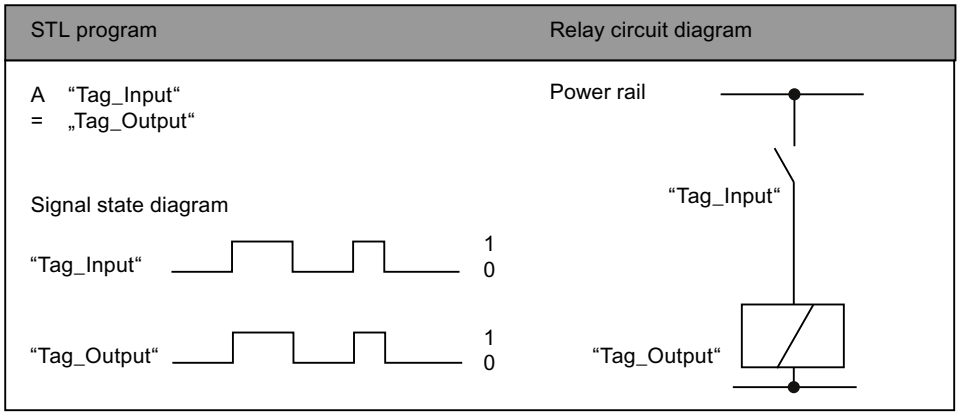

See also

Overview of the valid data types (Page 231)

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

R: Reset

Description

Use the "Reset" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed when the current result of logic operation (RLO) is "1". After execution of the instruction the specified operand is reset to "0". If there is a current RLO of "0", the signal state of the specified operand remains unchanged.

The execution of the "Reset" instruction is influenced by the Master Control Relay (MCR). If the MCR bit is set to "0", the signal state of the specified operand remains unchanged.

Parameters

The following table shows the parameters of the "Reset" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is reset when RLO = "1".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input_1"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
A "Tag_Input_2"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
R "Tag_Output"               // Reset the operand to "0" if RLO is "1"
```

See also

Overview of the valid data types (Page 231)

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

S: Set

Description

Use the "Set" instruction to set the signal state of a specified operand to "1".

The instruction is only executed when the current result of logic operation (RLO) is "1". After the instruction executes, the specified operand is set to "1". If there is a current RLO of "0", the signal state of the specified operand remains unchanged.

The execution of the "Set" instruction is influenced by the Master Control Relay (MCR). If the MCR bit is set to "0", the signal state of the specified operand remains unchanged.

Parameters

The following table shows the parameters of the "Set" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is set when RLO = "1".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input_1"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
A "Tag_Input_2"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
S "Tag_Output"               // Sets the operand to "1" if RLO = "1".
```

See also

Overview of the valid data types (Page 231)

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

NOT: Invert RLO

Description

Use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). You can use the "Invert RLO" instruction anywhere, even within a logic operation.

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input_1"              // Query the signal state of the "Tag_Input_1" operand
                             for "1" and AND with the current RLO.
```

STL	Explanation
A "Tag_Input_2"	// Query the signal state of the "Tag_Input_2" operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the "Tag_Input_3" operand for "1" and AND with the current RLO.
NOT	// Invert signal state of the RLO
= "Tag_Output"	// Assign signal state of the RLO to the "Tag_Output" operand.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Input_1	1	1	1	0	0	0	1	0
Tag_Input_2	0	1	1	0	1	1	0	1
Tag_Input_3	0	0	1	0	0	1	1	1
Tag_Output	1	1	0	1	1	1	1	1

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SET: Set RLO to 1

Description

Use the "Set RLO to 1" instruction to set the current result of logic operation (RLO) to signal state "1".

Example

The following example shows how the instruction works:

STL program	Signal state	Result of logic operation (RLO)
SET		1
= “Tag_Output_1“	1	
= “Tag_Output_2“	1	
= “Tag_Output_3“	1	
CLR		0
= “Tag_Output_4“	0	
= “Tag_Output_5“	0	

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

CLR: Reset RLO to 0

Description

Use the "Reset RLO to 0" instruction to set the current result of logic operation (RLO) to signal state "0".

Example

The following example shows how the instruction works:

STL program	Signal state	Result of logic operation (RLO)
SET		1
= “Tag_Output_1“	1	
= “Tag_Output_2“	1	
= “Tag_Output_3“	1	
CLR		0
= “Tag_Output_4“	0	
= “Tag_Output_5“	0	

See also

Overview of the valid data types (Page 231)

Setting the status bit for bit logic operations (Page 175)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SAVE: Save RLO in BR bit

Description

Use the "Save RLO in BR bit" instruction to save the result of logic operation (RLO) in the binary result (BR). When it executes, the instruction transfers the signal state of the current result of logic operation to the status bit BR. The instruction operates independent of the conditions and does not affect any other status bits.

Note

After executing the "Save RLO in BR bit" instruction, the BR bit can be modified again by the following instructions in the same or lower-level blocks.

Use the "Save RLO in BR bit" instruction to check the execution status of a block. If, for example, you use the instruction "Save RLO in BR bit" at the end of a block, the BR bit is set to the signal state of the current result of logic operation of the block.

The BR bit helps to implement the EN/ENO mechanism for block calls. For more information on this topic, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state of the Tag_Input operand for "1".
SAVE	// Transfer signal state of the RLO to the BR bit.
BEC	// End of block if RLO = "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Example of the simulation of the EN/ENO mechanism in STL (Page 153)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

FN: Scan RLO for negative signal edge

Description

Use the "Scan RLO for negative signal edge" instruction to query if a "1" to "0" change exists in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query saved in the edge memory bit (<Operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative signal edge.

The following figure shows the change in the RLO in case of a negative or positive signal edge:

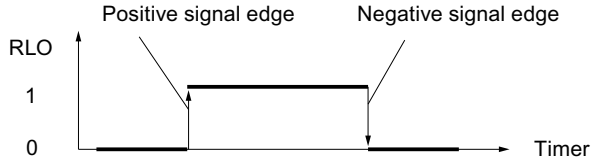

The negative signal edge is queried each time the instruction executes. When a negative signal edge is detected, the RLO is set to signal state "1" for the length of one program cycle after the edge evaluation. In all other cases, the signal state of the RLO is "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the previous query is saved.

Example

The following example shows how the instruction works:

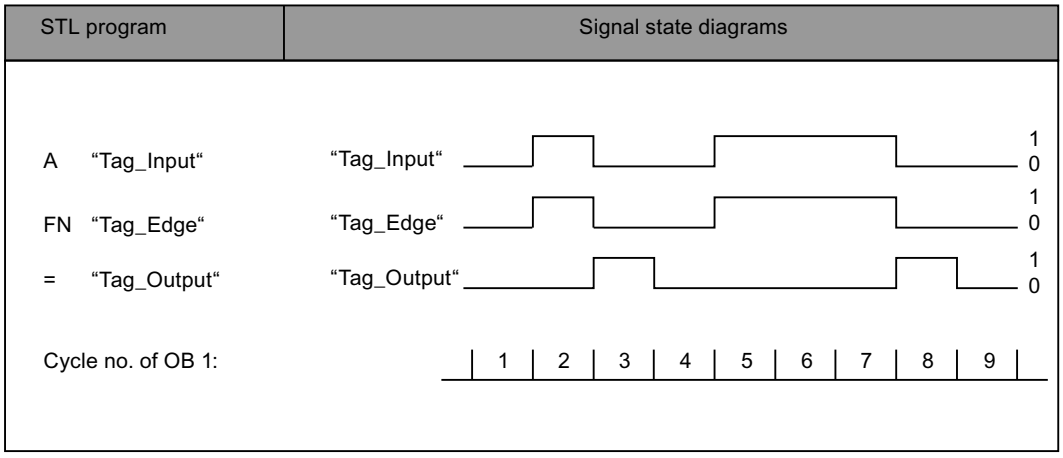

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

FP: Scan RLO for positive signal edge

Description

Use the "Scan RLO for positive signal edge" instruction to query if a "0" to "1" change exists in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query saved in the edge memory bit (<Operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

The following figure shows the change in the RLO in case of a negative or positive signal edge:

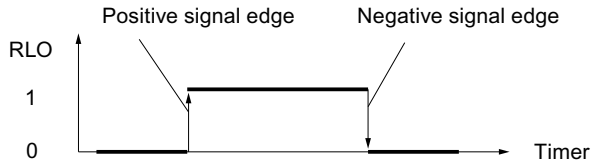

The positive signal edge is queried each time the instruction executes. When a positive signal edge is detected, the RLO is set to signal state "1" for the length of one program cycle after the edge evaluation. In all other cases, the signal state of the RLO is "0".

Note

Modification of the address of the edge memory bit

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

Parameters

The following table shows the parameters of the instruction "Scan RLO for positive signal edge":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the previous query is saved.

Example

The following example shows how the instruction works:

STL program	Signal state diagrams

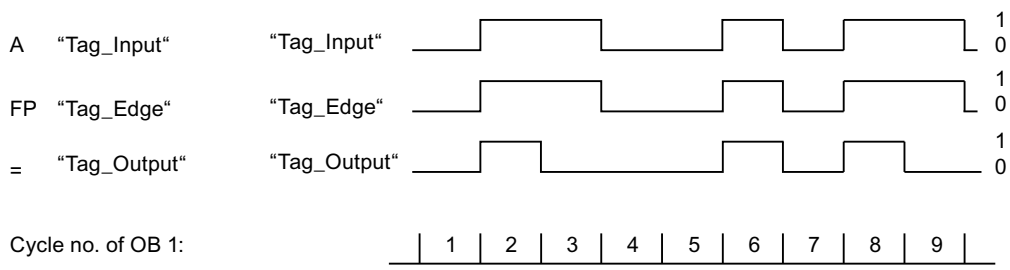

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bit for bit logic operations (Page 175)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

Timer operations

FR: Enable timer

Description

Use the instruction "Enable timer" to restart a timer. The instruction is executed with a positive edge and resets the internal edge trigger flag for the starting of a timer. During the processing of an instruction for starting the timer with a result of logic operation "1", the timer restarts even if no positive signal edge is present before the start instruction. If the result of logic operation is "0" before a start instruction, the resetting of the internal edge trigger flag does not affect the timer.

A change of the result of logic operation to "0" before the "Enable timer" instruction has no affect after the restart of the timer.

A valid time value in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of a start function.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Enable timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Enabled timer

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_EnableInput"	// Query operand for "1" and link with the current RLO according to AND.
FR "MyTimer"	// If the signal edge is positive, enable timer.
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

L: Load timer value (Page 4734)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

L: Load timer value

Description

With the "Load timer value" instruction you load the timer value of a specified timer in accumulator 1. The dual-coded timer value is loaded without the timebase. Zeros are written to accumulator 1 instead of the timebase.

After the loading, the value in accumulator 1 corresponds to a positive number of the data type INT.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

The following figure is an example of how the time value is loaded in the accumulator:

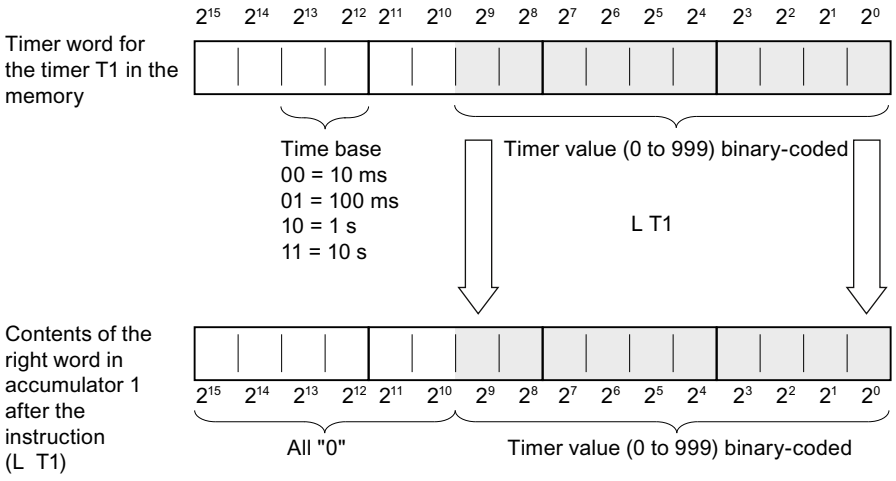

Parameters

The following table shows the parameters of the "Load timer value" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer whose current time value is loaded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_EnableInput"`	`// Query operand for "1" and link with the current RLO according to AND.`
`FR "MyTimer"`	`// If the signal edge is positive, enable timer.`
`A "Tag_StartInput"`	`// Query operand for "1" and link with the current RLO according to AND.`
`L S5T#10s`	`// Load period of time (10s)`

STL	Explanation
SP "MyTimer"	// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Setting status bits in timer and counter instructions (Page 177)

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

LC: Load BCD-coded timer value

Description

With the "Load BCD-coded timer value" instruction you load a BCD-coded timer value of a specified timer that exists dual coded in accumulator 1. During loading the timebase is also transferred to Accumulator 1.

After loading, the value in accumulator 1 corresponds to a time duration in S5TIME format. The left word in accumulator 1 is padded with zeros.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

The following figure shows how the time value is loaded in the accumulator:

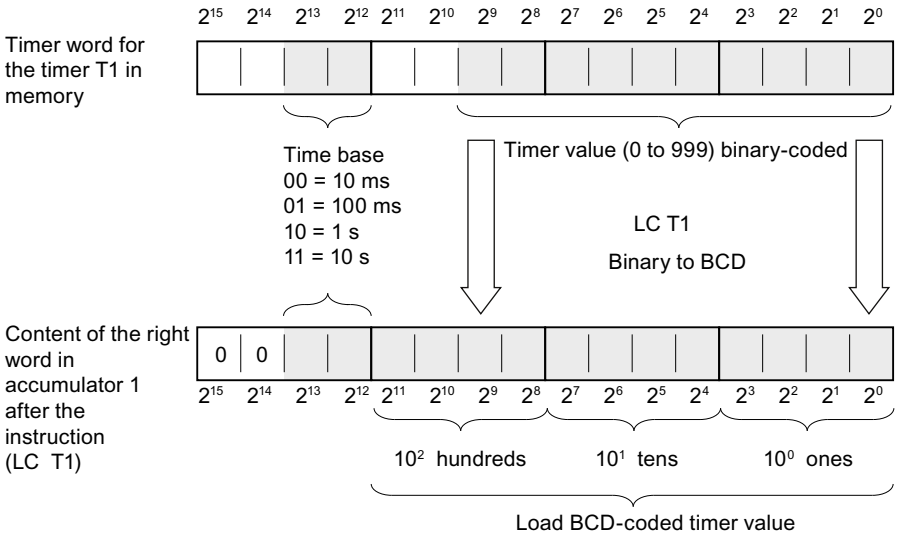

Parameters

The following table shows how the parameters of the "Load BCD-coded timer value" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer whose current BCD-coded timer value is loaded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_EnableInput"`	`// Query operand for "1" and link with the current RLO according to AND.`
`FR "MyTimer"`	`// If the signal edge is positive, enable timer.`
`A "Tag_StartInput"`	`// Query operand for "1" and link with the current RLO according to AND.`
`L S5T#10s`	`// Load period of time (10s)`
`SP "MyTimer"`	`// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".`
`A "Tag_Reset"`	`// Query operand for "1" and link with the current RLO according to AND.`
`R "MyTimer"`	`// If RLO = "1", reset timer.`
`A "MyTimer"`	`// Query status of timer for "1".`
`= "Tag_TimerStatus"`	`// Assign timer status to the operand.`
`LC "MyTimer"`	`// Load current BCD-coded timer value of timer in accumulator 1.`

STL	Explanation
`T "Tag_TimerValue"`	`// Transfer current time value to operand "Tag_TimerValue".`

See also

Basic information on the status word (Page 173)

Setting status bits in timer and counter instructions (Page 177)

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

R: Reset timer

Description

Use the "Reset timer" instruction to reset a specified timer to "0". The instruction is executed when the current result of logic operation (RLO) is "1". As long as the RLO is "1" before the instruction, the specified timer returns the value "0". The instruction also resets the time value and the time basis of the programmed period of time to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

Note

The "Reset timer" instruction does not reset the internal edge memory bit. To reset the internal edge memory bit, program the "Enable timer" instruction or execute an instruction to start the time with the signal state "0".

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Reset timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is reset

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SP: Start pulse timer

Description

Use the "Start pulse timer" instruction to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1, as long as the result of logic operation (RLO) before the instruction remains "1". If the RLO changes to "0" before the end of the period of time, the timer is stopped. The queries of the timer status for "1" return the query result "1", as long as the period of time is running.

The timer can be restarted with a positive signal edge before the "Start pulse timer" instruction or using the "Enable timer" instruction. This restart can be effected provided the timer is not reset.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start pulse timer" instruction, the time value is counted down according to the timebase. The timer runs until the count value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If accumulator 1 does not contain a valid BCD

value, the synchronous error handling (STOP or OB122) is performed. The status bit OV is not influenced.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Start pulse timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer that is started as pulse

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:

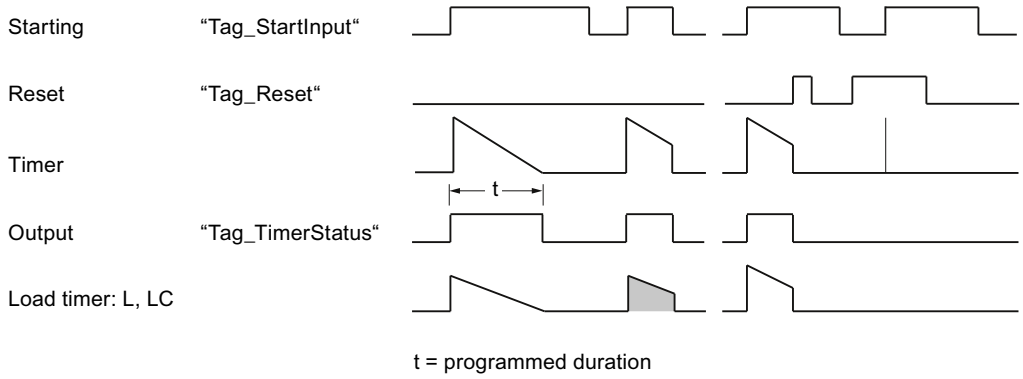

See also

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SE: Start extended pulse timer

Description

Use the "Start extended pulse timer" to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1, as long as the result of logic operation changes to "0" before the instruction. The queries of the timer status for "1" return the query result "1", as long as the period of time is running.

With each positive signal edge, the instruction restarts the timer with the programmed time duration, even if the timer has still not run down.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start extended pulse timer" instruction, the time value is counted down according to the timebase. The timer runs until the count value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If accumulator 1 does not contain a valid BCD

value, the synchronous error handling (STOP or OB 122) is performed. The status bit OV is not influenced.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameters

The following table shows the parameters of the "Start extended pulse timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as extended pulse

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_StartInput"           // Query operand for "1" and link with the current RLO
                             according to AND.
L S5T#10s                    // Load period of time (10s)
SE "MyTimer"                 // Start timer as extended pulse if there is a posi-
                             tive signal edge at the "Tag_StartInput" operand or at
                             the "Tag_EnableInput" operand and the "Tag_StartIn-
                             put" operand has signal state "1".
A "Tag_Reset"                // Query operand for "1" and link with the current RLO
                             according to AND.
R "MyTimer"                  // If RLO = "1", reset timer and stop timing
A "MyTimer"                  // Query status of timer for "1".
= "Tag_TimerStatus"          // Assign timer status to the operand.
L "MyTimer"                  // Load current time value of timer in accumulator 1.
T "Tag_TimerValue"           // Transfer current time value to operand "Tag_Time-
                             rValue".
LC "MyTimer"                 // Load current time value in BCD format in accumula-
                             tor 1.
T "Tag_TimerValue_BCD"       // Transfer time value in BCD format to operand
                             "Tag_TimerValue_BCD"
```

The figure below shows the timer diagram of the example:

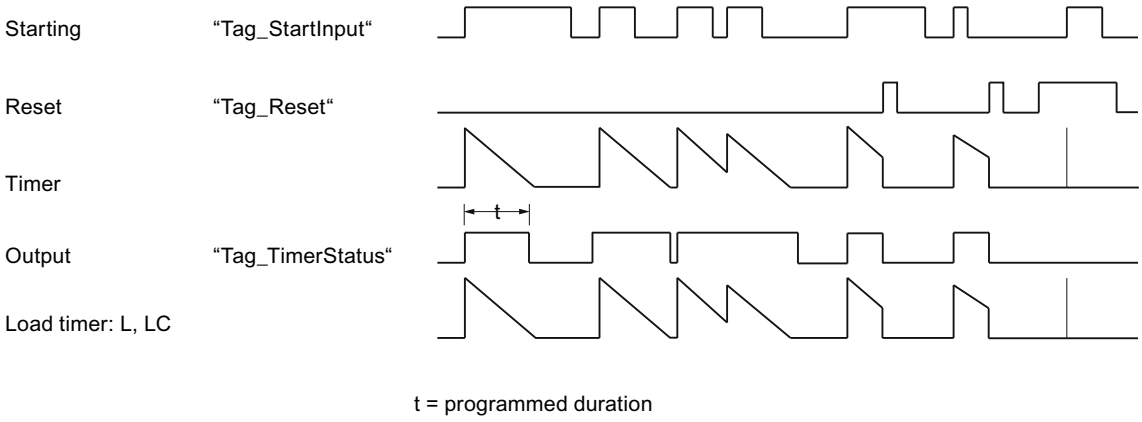

See also

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

SD: Start on-delay timer

Description

The "Start on-delay timer" instruction starts a programmed timer when the signal state "1" is detected at the start input. As long as this signal state remains "1", the timer expires with the duration specified in accumulator 1. When the timer expires and the signal state at the start input is still "1", the query for timer status also returns "1". If the signal at the start input is "0", the timer is reset. In this case, querying the timer status returns the signal state "0". As soon as the signal at the start input changes to "1" again, the timer starts running again.

The signal state of the timer output is identical to the signal state of the start input. The start input is directly interconnected to the output and is not linked to the timer.

The duration in accumulator 1 is made up internally of a time value and a time base. When the instruction starts, the programmed time value counts down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit

OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameters

The following table shows the parameters of the "Start on-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as on delay

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_StartInput"           // Query operand for "1" and link with the current RLO
                             according to AND.
L S5T#10s                    // Load period of time (10s)
SD "MyTimer"                 // Start on-delay timer if there is a positive signal
                             edge at the "Tag_StartInput" operand or at the "Tag_En-
                             ableInput" operand and the "Tag_StartInput" operand
                             has signal state "1".
A "Tag_Reset"                // Query operand for "1" and link with the current RLO
                             according to AND.
R "MyTimer"                  // If RLO = "1", reset timer and stop timing
A "MyTimer"                  // Query status of timer for "1".
= "Tag_TimerStatus"          // Assign timer status to the operand.
L "MyTimer"                  // Load current time value of timer in accumulator 1.
T "Tag_TimerValue"           // Transfer current time value to operand "Tag_Time-
                             rValue".
LC "MyTimer"                 // Load current time value in BCD format in accumula-
                             tor 1.
T "Tag_TimerValue_BCD"       // Transfer time value in BCD format to the operand
                             "Tag_TimerValue_BCD"
```

The figure below shows the timer diagram of the example:

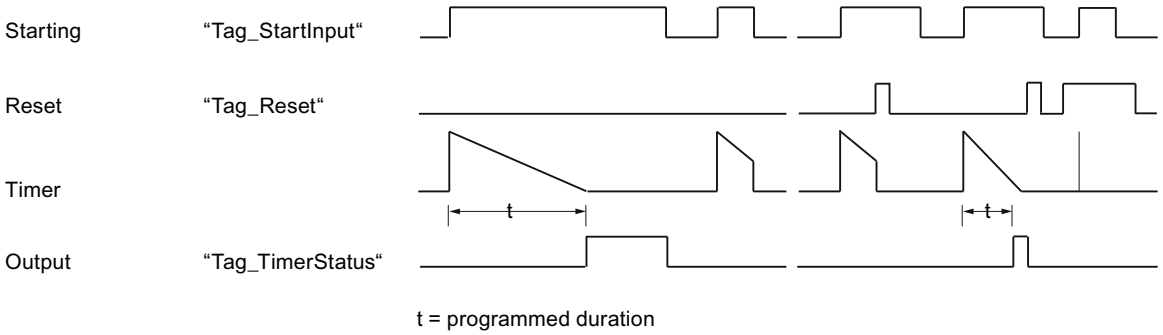

See also

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SS: Start retentive on-delay timer

Description

Use the "Start retentive on-delay timer" instruction to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs with the period of time specified in accumulator 1, even if the result of logic operation (RLO) changes to "0" before the instruction. When the time has run out, queries of the timer status for "1" return the query result "1". The query result is not hereby affected by the signal state of the current result of logic operation before the instruction.

With each positive signal edge, the instruction restarts the timer with the programmed time duration, even if the timer has still not run down.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start retentive on-delay timer" instruction, the time value is counted down according to the time base. The timer runs until the count value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If accumulator 1 does not contain a valid BCD

value, the synchronous error handling (STOP or OB 122) is performed. The status bit OV is not influenced.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameters

The following table shows the parameters of the "Start retentive on-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer that is started as retentive on-delay

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SS "MyTimer"	// Start retentive on-delay timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to the operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:

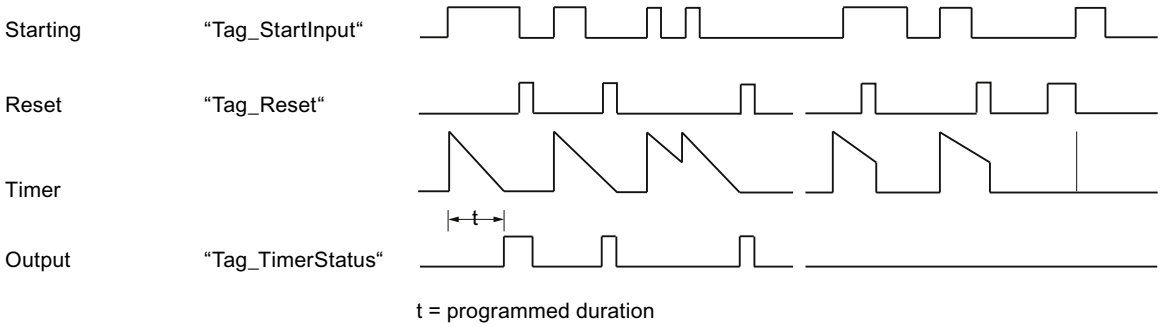

See also

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SF: Start off-delay timer

Description

Use the "Start off-delay timer" instruction to start a specified timer on a positive signal edge. If a negative signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1. If the result of logic operation (RLO) changes to "1" before the end of the period of time, the timer is reset. The timer is only restarted if a negative signal edge is detected during the processing of the instruction.

The queries of the timer status for "1" return the query result "1" if the signal state of the result of logic operation is "1" at the execution of the instruction or if the programmed time is running. If the time does not elapse or the RLO is "0", queries of the timer status for "1" return the query result "0".

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start off-delay timer" instruction, the time value is counted down according to the timebase. The timer runs until the count value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If accumulator 1 does not contain a valid BCD

value, the synchronous error handling (STOP or OB 122) is performed. The status bit OV is not influenced.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed here: L: Load timer value (Page 4734)

Parameters

The following table shows the parameters of the "Start off-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as off delay

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_StartInput"           // Query operand for "1" and link with the current RLO
                             according to AND.
L S5T#10s                    // Load period of time (10s)
SF "MyTimer"                 // Start off-delay timer if there is a negative signal
                             edge at the "Tag_StartInput" operand or at the "Tag_En-
                             ableInput" operand and the "Tag_StartInput" operand
                             has signal state "1".
A "Tag_Reset"                // Query operand for "1" and link with the current RLO
                             according to AND.
R "MyTimer"                  // If RLO = "1", reset timer and stop timing
A "MyTimer"                  // Query status of timer for "1".
= "Tag_TimerStatus"          // Assign timer status to the operand.
L "MyTimer"                  // Load current time value of timer in accumulator 1.
T "Tag_TimerValue"           // Transfer current time value to operand "Tag_Time-
                             rValue".
LC "MyTimer"                 // Load current time value in BCD format in accumula-
                             tor 1.
T "Tag_TimerValue_BCD"       // Transfer time value in BCD format to operand
                             "Tag_TimerValue_BCD"
```

The figure below shows the timer diagram of the example:

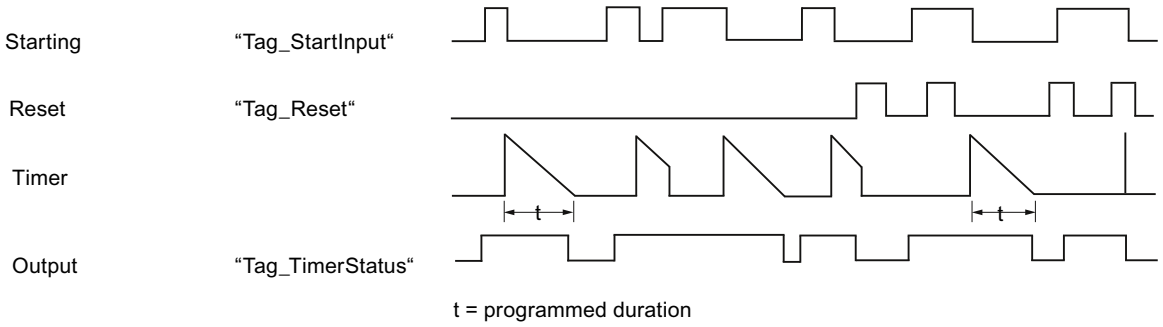

See also

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Counter operations

FR: Enable counter

Description

Use the "Enable counter" instruction to execute the instructions "Set counter", "Count up" and "Count down" without the presence of a positive signal edge. During processing, the instruction resets the internal edge flag to the signal state "0". The query before the corresponding count instruction must continue to have signal state "1".

The "Enable counter" instruction is executed when the result of logic operation (RLO) before the instruction changes from "0" to "1".

A counter does not have to be enabled for the count instruction to be executed.

Parameters

The following table shows the parameters of the "Enable counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter that is enabled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query signal state for "1" and AND with the current "RLO.
FR "MyCounter"	// Enable counter
A "Tag_Input_2"	// Query signal state for "1" and AND with the current "RLO.
CD "MyCounter"	// Count down
A "MyCounter"	// Query counter status
= "Tag_Output"	// Assign query result to the operand.

In the example the counter ("MyCounter") is enabled when the signal state of the operand "Tag_Input_1" changes from "0" to "1". When the operand "Tag_Input_2" has signal state "1", the count value is decremented by one when the counter is enabled. When the signal state of the address "Tag_Input_2" is "0", the count value remains unchanged. The count value is also decremented by one regardless of the counter enabling, when the signal state of the address "Tag_Input_2" changes from "0" to "1".

The query of the counter status has the result "1" when the counter reading is greater than zero. If the counter reading is equal to zero, the result of the query is "0".

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

L: Load counter

Description

With the "Load counter" instruction you transfer the current value of a specified counter in binary coding to the right word of accumulator 1. The loaded counter is then available as a 16-bit integer for additional processing.

The following figure shows how the instruction works:

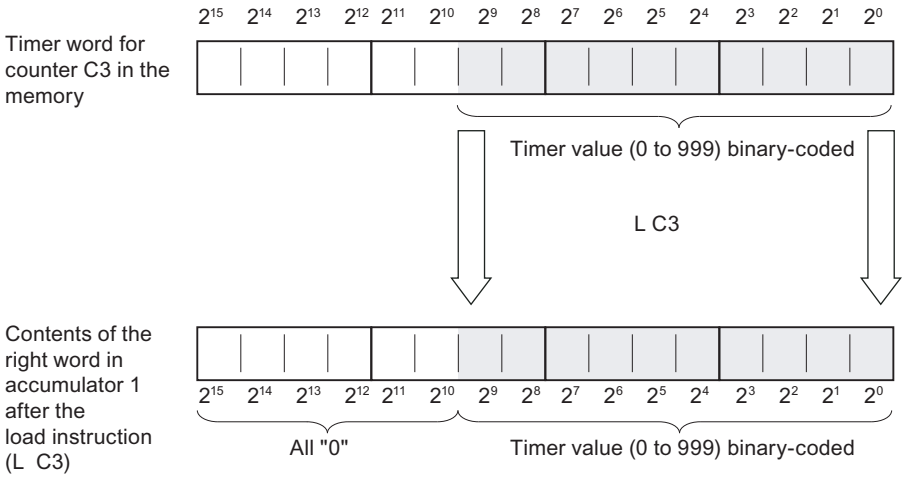

Parameters

The following table shows the parameters of the "Load counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose current value is loaded.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "MyCounter"                // Load current count value binary-coded in accumula-
                             tor 1
L "Tag_Value"                // Move the counter value to accumulator 2.
                             // Load value of the operand "Tag_Value" in accumula-
                             tor 1
>=I                          // Query whether the count value is greater than or
                             equal to the value of the address "Tag_Value"
= "Tag_Output"               // Assign query result to the operand.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting status bits in timer and counter instructions (Page 177)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

LC: Load BCD-coded counter value

Description

With the "Load BCD-coded counter value" instruction you transfer the current value of a specified BCD-coded counter to the right word of accumulator 1. The loaded counter value is then available in BCD form for additional processing.

The following figure shows how the "Load BCD-coded counter value" instruction works:

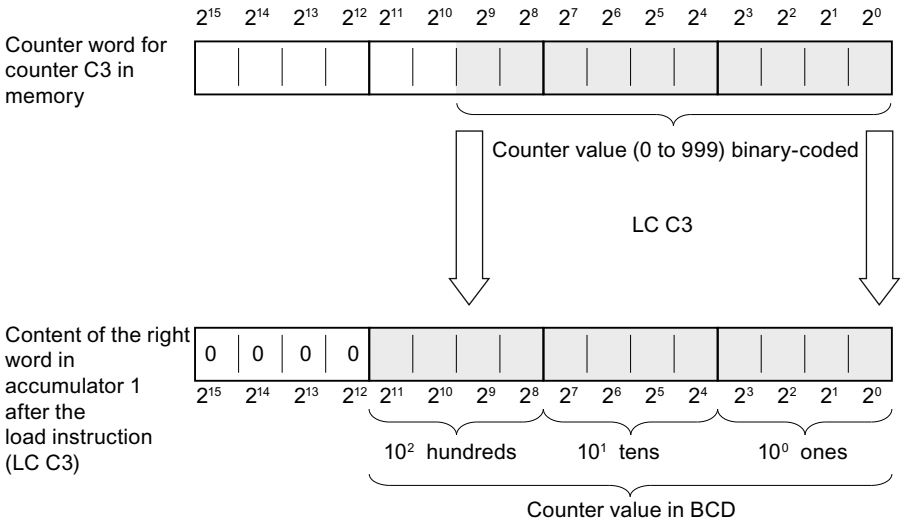

Parameters

The following table shows the parameters of the "Load BCD-coded counter value" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose current value is loaded.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`LC "MyCounter"`	`// Load current BCD-coded counter value in accumulator 1.`
`T "Tag_Output"`	`// Transfer current counter value to the address "Tag_Output" and save.`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting status bits in timer and counter instructions (Page 177)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

R: Reset counter

Description

Use the "Reset counter" instruction to reset the current count value of a specified counter to "0".

The instruction is only executed when the result of logic operation (RLO) before the instruction has signal state "1". As long as the RLO before the "Reset counter" instruction is "1", the queries of the counter status for "1" return the result "0".

When a counter is reset, the internal edge memory bit for the instructions "Set counter", "Count up" and "Count down" is not reset.

Parameters

The following table shows the parameters of the "Reset counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose count value is reset to "0".

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input"                // Query signal state for "1" and AND with the current
                             "RLO.
R "MyCounter"                // Reset counter
```

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

S: Set counter

Description

Use the "Set counter" instruction to set a specified counter to the value contained in accumulator 1 at the time of the query.

The "Set counter" instruction is executed when the result of logic operation before the instruction changes from "0" to "1". The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

To set a counter, accumulator 1 must contain a valid count value. Valid count values are values in BCD format with three decades. Only positive count values are permitted; the count instruction cannot process negative count values. If accumulator 1 does not contain a valid BCD value, the synchronous error handling (STOP or OB122) is performed. The status bit OV is not influenced.

Parameters

The following table shows the parameters of the "Set counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter that is set.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input"                // Query signal state for "1" and AND with the current
                             "RLO.
L "Tag_CountValue"           // Load count value in accumulator 1.
S "MyCounter"                // Set counter to the value that is in accumulator 1
```

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

CU: Count up

Description

Use the "Count up" instruction to increment the current count value by one when the result of logic operation (RLO) before the instruction changes from "0" to "1". The counter value continues to be incremented for each positive signal edge until the high limit of "999" is reached. When the limit is reached the RLO before the instruction no longer has any affect on the count value.

A positive signal edge is always required to execute the "Count up" instruction. The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

Parameters

The following table shows the parameters of the "Count up" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter, whose count value is incremented by one on a positive edge.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_Input"`	`// Query signal state for "1" and AND with the current "RLO.`
`CU "MyCounter"`	`// Increment count value by one on a positive edge.`

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

CD: Count down

Description

Use the "Count down" instruction to decrement the current count value by one when the result of logic operation (RLO) before the instruction changes from "0" to "1". The count value continues to be decremented for each positive signal edge until the low limit of "0" is reached. When the limit is reached the RLO before the instruction no longer has any affect on the count value. Counting does not occur when count value is negative.

A positive signal edge is always required to execute the "Count down" instruction. The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

Parameters

The following table shows the parameters of the "Count down" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter, whose count value is decremented by one on a positive edge.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input"                // Query signal state for "1" and AND with the current
                             "RLO.
CD "MyCounter"               // Decrement counter value by one on a positive edge.
```

See also

Overview of the valid data types (Page 231)

Setting status bits in timer and counter instructions (Page 177)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

Comparator operations

? I: Compare 16-bit integers

Description

With the "Compare 16-bit integers" instruction you compare the content of the right word in accumulator 2 with the content of the right word in accumulator 1. The contents of accumulators 1 and 2 are interpreted as 16-bit integers. The instruction performs the comparison depending on the specified comparison function.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==I	Compare 16-bit integers for "equivalence"
<>I	Compare 16-bit integers for "non-equivalence"
>I	Compare 16-bit integers for "greater than"
<I	Compare 16-bit integers for "less than"
>=I	Compare 16-bit integers for "greater than or equal to"
<=I	Compare 16-bit integers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". When the condition of the comparison is not met, the RLO is set to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==I	0	1	0
<>I	1	0	1
>I	1	0	0
<I	0	0	1
>=I	1	1	0
<=I	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function you can evaluate the result of the logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input_1"`	`// Load value of the operand "Tag_Input_1" in accumulator 1`

STL	Explanation
L "Tag_Input_2"	// Shift the contents of accumulator 1 (Tag_Input_1) to accumulator 2.
	// Load value of the operand "Tag_Input_2" in accumulator 1
>I	// Compare whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".
= "Tag_Output"	// Assign result of the comparison to the operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	10	10	5
Tag_Input_2	5	10	10
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

? D: Compare 32-bit integers

Description

Use the instruction "Compare 32-bit integers" to compare the contents of accumulator 2 (ACCU 2) with the contents of accumulator 1 (ACCU 1). The contents of accumulators 1 and 2 are interpreted as 32-bit integers. The instruction performs the comparison depending on the specified comparison function.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==D	Compare 32-bit integers for "equivalence"
<>D	Compare 32-bit integers for "non-equivalence"
>D	Compare 32-bit integers for "greater than"
<D	Compare 32-bit integers for "less than"
>=D	Compare 32-bit integers for "greater than or equal to"
<=D	Compare 32-bit integers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". If the condition of the comparison is not met, the RLO is reset to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==D	0	1	0
<>D	1	0	1
>D	1	0	0
<D	0	0	1
>=D	1	1	0
<=D	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function, you can evaluate the result of the logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input_1"`	`// Load value of the operand "Tag_Input_1" in accumulator 1`
`L "Tag_Input_2"`	`// Contents of accumulator 1 (Tag_Input_1) shifted to accumulator 2.`
	`// Load value of the operand "Tag_Input_2" in accumulator 1`
`>D`	`// Compare whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".`
`= "Tag_Output"`	`// Assign result of the comparison to the operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	150 000	150 000	100 000
Tag_Input_2	100 000	150 000	150 000
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

? R: Compare floating-point numbers

Description

Use the instruction "Compare floating-point numbers" to compare the contents of accumulator 2 (ACCU 2) with the contents of accumulator 1 (ACCU 1). The contents of accumulators 1 and 2 are interpreted as floating-point numbers. For the comparison, accumulators 1 and 2 must contain valid floating-point numbers. The instruction performs the comparison depending on the specified comparison function.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==R	Compare floating-point numbers for "equivalence"
<>R	Compare floating-point numbers for "non-equivalence"
>R	Compare floating-point numbers for "greater than"
<R	Compare floating-point numbers for "less than"
>=R	Compare floating-point numbers for "greater than or equal to"
<=R	Compare floating-point numbers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". If the condition of the comparison is not met, the RLO is reset to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==R	0	1	0
<>R	1	0	1
>R	1	0	0
<R	0	0	1
>=R	1	1	0
<=R	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

Note

During a comparison of invalid floating-point numbers, the CPU sets the result of logic operation to "0" and the status bits CC 0, CC 1, OVand OS to "1".

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function you can evaluate the result of the logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input_1"`	`// Load value of the operand "Tag_Input_1" in accumulator 1`
`L "Tag_Input_2"`	`// Shift the contents of accumulator 1 (Tag_Input_1) to accumulator 2.`
	`// Load value of the operand "Tag_Input_2" in accumulator 1`
`>R`	`// Compare whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".`
`= "Tag_Output"`	`// Assign result of the comparison to the operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	10 515	10 515	5 232
Tag_Input_2	5 232	10 515	10 515
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 231)

Setting the status bit for comparison instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Math functions

Integers

+I: Add integers (16 bit)

Description

Use the instruction "Add integers (16 bit)" to add the values in the right words of accumulators 1 and 2. The instruction interprets these values as 16-bit integers.

The instruction saves the sum in the right word of accumulator 1. The left word of accumulator 1 remains unchanged.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for addition.
L "Tag_Value_2"	// Load second value for addition.
+I	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

-I: Subtract integers (16 bit)

Description

With the "Subtract integers (16 bit)" instruction you subtract the value of the right word of accumulator 1 from the value of the right word of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as 16-bit integers.

The instruction saves the difference in the right word of accumulator 1. The left word of accumulator 1 remains unchanged.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load first value of subtraction into accumulator 1.`
`L "Tag_Value_2"`	`// Shift the contents of accumulator 1 to accumulator 2.`
	`// Load second value for addition in accumulator 1.`
`-I`	`// Subtract value of accumulator 1 from value of accumulator 2.`
`T "Tag_Result"`	`// Transfer difference to the operand "Tag_Result".`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

*I: Multiply integers (16 bit)

Description

Use the instruction "Multiply integers (16 bit)" to multiply the values in the right words of accumulators 1 and 2. The instruction interprets these values as 16-bit integers.

The instruction saves the product in accumulator 1 as 32-bit integer (DINT).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load first value for multiplication.`
`L "Tag_Value_2"`	`// Load second value for multiplication.`
`*I`	`// Multiply values`
`T "Tag_Result"`	`// Transfer product to the operand "Tag_Result".`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

/I: Divide integers (16 bit)

Description

With the "Divide integers (16 bit)" instruction you divide the contents of the right word in accumulator 2 by the contents of the right word in accumulator 1. The instruction interprets the values of the accumulators as 16-bit integers.

The instruction provides two results: the quotients and the remainder of the division. The quotient is the whole number result of the division. The instruction saves this in the right word

of accumulator 1. The instruction saves the remainder of the division in the left word of accumulator 1. If the dividend is negative, the remainder of the division is also negative.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value as quotient and in each case the value zero as remainder. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load dividend in accumulator 1.`
`L "Tag_Value_2"`	`// Move dividend to accumulator 2.`
	`// Load divisor in accumulator 1.`
`/I`	`// Divide the value of accumulator 2 by the value of accumulator 1.`
`T "Tag_Result"`	`// Transfer the result to operand "Tag_Result".`

Before processing

The following table shows the contents of the right words in accumulators 1 and 2 before executing the "Divide integers (16 bit)" instruction:

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the "Divide integers (16 bit)" instruction:

Accumulator 1	Value
Bit 0 to 15 (right word)	3
Bit 16 to 31 (left word)	1

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

+D: Add double integers (32 bit)

Description

Use the "Add double integers (32 bit)" instruction to add the values in accumulators 1 and 2. The instruction interprets these values as 32-bit integers. The instruction saves the sum in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load first value for addition.`
`L "Tag_Value_2"`	`// Load second value for addition.`
`+D`	`// Add values`
`T "Tag_Result"`	`// Transfer sum to the operand "Tag_Result".`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

-D: Subtract double integers (32 bit)

Description

With the "Subtract double integers (32 bit)" instruction you subtract the value of accumulator 1 from the value of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as 32-bit double integers. The instruction saves the result of the subtraction in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load the second value of subtraction into accumulator 1.
-D	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

*D: Multiply double integers (32 bit)

Description

Use the "Multiply double integers (32 bit)" instruction to multiply the values in accumulators 1 and 2. The instruction interprets these values as 32-bit integers. The instruction saves the product in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*D	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

/D: Divide double integers (32 bit)

Description

With the "Divide double integers (32 bit)" instruction you divide the value of accumulator 2 by the value of accumulator 1. The instruction interprets the values of accumulators 1 and 2 as 32-bit double integers. The instruction saves the result of the division (quotient) in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value zero as quotient. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load dividend in accumulator 1.`
`L "Tag_Value_2"`	`// Move dividend to accumulator 2.`
	`// Load divisor in accumulator 1.`
`/D`	`// Divide the value of accumulator 2 by the value of accumulator 1.`
`T "Tag_Result"`	`// Transfer quotients to the operand "Tag_Result".`

Before processing

The following table shows the contents of accumulators 1 and 2 before executing the "Divide double integers (32 bit)" instruction:

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the "Divide double integers (32 bit)" instruction:

Accumulator	Value
Accumulator 1	3

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

+: Add constants

Description

With the "Add constants" instruction, you add the value of a specified constant to the contents of accumulator 1. You can specify the constant value as 16-bit integer or 32-bit double integer. The instruction stores the result in accumulator 1. When adding 16-bit integers, the instruction only influences the right word of accumulator 1.

To execute the addition of a 16-bit integer as DINT math function, enter "L#" before the constant. The "Add constants" instruction permits the use of negative numbers.

The instruction does not affect any status bits.

Parameters

The following table shows the parameters of the "Add constants" instruction:

Parameters	Data type	Description
<constant>	INT, DINT	Constants to be added

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
+ 25                         // Increment the value of accumulator 1 by 25.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand
                             "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2" in accumu-
                             lator 1
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in accumu-
                             lator 1
+ -100                       // Decrement the value of accumulator 1 by 100.
<I                           // Check whether the value of accumulator 2 is less
                             than the value of accumulator 1
JC NEXT                      // If the result of the query is "1", jump to jump la-
                             bel NEXT .
                             // If the result of the query is "0", execute the next
                             instruction.
T "Tag_Result_2"             Transfer contents of accumulator 1 to the operand
                             "Tag_Result_2".
L "Tag_Value_4"              // Load value of the operand "Tag_Value_4".
+ L#20                       // Increment content of accumulator 1 by 20.
                             // The addition is done as a DINT calculation
T "Tag_Result_3"             // Transfer contents of accumulator 1 to the operand
                             "Tag_Result_3".
```

```
STL                          Explanation
NEXT: L "Tag_Value_5"        // Jump label "NEXT"
                             // Load value of the operand "Tag_Value_5".
+D                           // Add values of accumulators 1 and 2.
T "Tag_Result_4"             // Transfer sum to the operand "Tag_Result_4".
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Setting the status bits in instructions with floating-point numbers (Page 179)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

INC: Increment

Description

Use the "Increment" instruction to increment the value in accumulator 1 by the value specified as parameter of the instruction. The specified value can be in a value range from 0 to 255. The value is incremented only in the right byte of accumulator 1. There is no transfer to the byte located on the left.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of accumulator 2, 3 and 4 remain unchanged.

If the parameter value is outside the permitted range of value, the result will be corrupted without generating an error message.

Parameters

The following table shows the parameters of the "Increment" instruction:

Parameters	Format	Description
<constant>	8-bit integer	Value by which the value is incremented.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load value of the operand in accumulator 1.
INC 5                        // Increment value of accumulator 1 by 5.
```

STL	Explanation
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

DEC: Decrement

Description

Use the "Decrement" instruction to decrement the value in accumulator 1 by the value specified as parameter of the instruction. The specified value can be in a value range from 0 to 255. The value is decremented only in the right byte of accumulator 1. There is no transfer to the byte located on the left.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of accumulator 2, 3 and 4 remain unchanged.

If the parameter value is outside the permitted range of value, the result will be corrupted without generating an error message.

Parameters

The following table shows the parameters of the "Decrement" instruction:

Parameters	Format	Description
<constant>	8-bit integer	Value by which the value is decremented.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load value of the operand in accumulator 1.
DEC 5	// Decrement the value of accumulator 1 by 5.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

MOD: Return remainder of division

Description

Use the "Return remainder of division" instruction to divide the value of accumulator 2 by the value of accumulator 1 and save the remainder of the division.

The instruction interprets the values of accumulators 1 and 2 as 32-bit integers. The instruction stores the remainder of the division in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the remainder of division is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction returns the value zero as result. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
MOD	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer remainder of division to the operand "Tag_Result".

Before processing

The following table shows the contents of accumulators 1 and 2 before executing the instruction "Return remainder of division":

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the instruction "Return remainder of division":

Accumulator	Value
Accumulator 1	1

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Floating-point numbers

+R: Add floating-point numbers

Description

Use the "Add floating-point numbers" instruction to add the values contained in accumulators 1 and 2. The instruction interprets these values as floating-point numbers. The instruction saves the sum in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for addition.
L "Tag_Value_2"	// Load second value for addition.
+R	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

-R: Subtract floating-point numbers

Description

With the "Subtract floating-point numbers" instruction you subtract the value of accumulator 1 from the value of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as floating-point numbers. The instruction stores the difference in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.

STL	Explanation
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load the second value of subtraction into accumulator 1.
-R	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

*R: Multiply floating-point numbers

Description

Use the "Multiply floating-point numbers" instruction to multiply the values contained in accumulators 1 and 2. The instruction interprets these values as floating-point numbers. The instruction saves the product in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*R	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

/R: Divide floating-point numbers

Description

With the "Divide floating-point numbers" instruction you divide the value of accumulator 2 by the value of accumulator 1. The instruction interprets the values of accumulators 1 and 2 as floating-point numbers. The instruction saves the result of the division (quotient) in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value zero as quotient. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load dividend in accumulator 1.`
`L "Tag_Value_2"`	`// Move dividend to accumulator 2.`
	`// Load divisor in accumulator 1.`
`/R`	`// Divide the value of accumulator 2 by the value of accumulator 1.`
`T "Tag_Result"`	`// Transfer quotients to the operand "Tag_Result".`

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

ABS: Form absolute value

Description

Use the "Form absolute value" instruction to form the absolute value of the value in accumulator 1. The instruction interprets the contents of accumulator 1 as a floating-point number and sets the sign of the mantissa to "0". The result is stored in accumulator 1.

The "Form absolute value" instruction does not affect any status bits.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load floating-point number into accumulator 1
ABS                          // Form absolute value
T "Tag_Result"               // Transfer the result to operand "Tag_Result".
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	-1.5E+02
Tag_Result	1.5E+02

See also

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Advanced floating-point numbers

SQR: Form square

Description

Use the "Form square" instruction to calculate the square of a floating-point number located in accumulator 1. The result is stored in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SQR	// Form square
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SQRT: Form square root

Description

Use the "Form square root" instruction to form the square root of the floating-point number located in accumulator 1. The result is stored in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value less than zero or an invalid floating-point number, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

If accumulator 1 has the value "0", the instruction returns the value "0".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
SQRT	// Get square root
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

EXP: Form exponential value

Description

Use the "Form exponential value" to form the exponent from the base e (e = 2.718282) and the floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
EXP	// Form exponential value
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

LN: Form natural logarithm

Description

Use the "Form natural logarithm" instruction to calculate the natural logarithm of the value in accumulator 1 to base e (e = 2.718282).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value less than or equal to zero, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
LN	// Form natural logarithm
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle in accumulator 1. The angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SIN	// Form sine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

ASIN: Form arcsine value

Description

Use the "Form arcsine value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number in a value range from "-1" to "1".

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from $-\pi/2$ to $+\pi/2$ (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ASIN	// Form arcsine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the angle in accumulator 1. The size of the angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
COS	// Form cosine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

ACOS: Form arccosine value

Description

Use the "Form arccosine value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number in a value range from "-1" to "1".

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from "0" to "π" (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ACOS	// Form arccosine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the tangent of the angle in accumulator 1. The size of the angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
TAN	// Form tangent value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

ATAN: Form arctangent value

Description

Use the "Form arctangent value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number.

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from $-\pi/2$ to $+\pi/2$ (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ATAN	// Form arctangent value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Load and transfer

Load

L: Load

Description

With the "Load" instruction you load the content of a specified operand in accumulator 1. The operand that is to be loaded can be addressed as a byte, word or double word in one of the following memory areas:

- Process image of the inputs and outputs (I, Q)
- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)
- Pointer
- I/O (PI)

- Timer (T)
- Counter (C)

The memory area of accumulator 1 is organized in bytes and is 32 bits wide.

The instruction writes the content of an operand that is to be loaded in byte format right aligned in accumulator 1. The remaining Bytes in accumulator 1 are filled with "0".

The instruction writes the content of the operand to be loaded in word format to the right word of accumulator 1. The higher addressed byte is hereby transferred into the right byte (bits 0 to 7) of accumulator 1. The lower addressed byte is written to the left of it. The remaining bytes in the left word of accumulator 1 are filled with "0".

The instruction writes the content of the operand to be loaded in double word format to the 32 bits of accumulator 1. The highest addressed byte is hereby transferred into the right byte (bits 0 to 7) of accumulator 1. The lowest addressed byte is written in bits 24 to 31.

The following table shows, as an example, how the contents of accumulator 1 are changed by the loading of operands in byte, word and double word format:

Instruction	Accumulator 1							
	31 . . .	. . . 24	23 . . .	. . . 16	15 . . .	. . . 8	7 . . .	. . . 0
Loading of a byte: L MB10	0000	0000	0000	0000	0000	0000	<MB10>	
Loading of a word: L MW10	0000	0000	0000	0000	<MB10>		<MB11>	
Load a double word: L MD10	<MB10>		<MB11>		<MB12>		<MB13>	

During the loading of the specified operand the previous contents of accumulator 1 are moved to accumulator 2. The "Load" instruction transfers the complete contents of accumulator 1 to accumulator 2. The previous content of accumulator 2 is lost during this.

In case of controllers with four accumulators, the contents of accumulators 3 and 4 are not influenced by the instruction.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Parameters

The following table shows the parameters of the "Load" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Bit strings, integers, timers, REAL, DATE, TOD, CHAR	I, Q, PI, M, L, DB, DI, T, C, pointer, parameter	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

STL programming examples (Page 7280)

L STW: Load status word in accumulator 1

Description

Use the "Load status word in accumulator 1" instruction to load the status word in accumulator 1. When it executes, the instruction writes the individual status bits in the right word of accumulator 1 The remaining bits of accumulator 1 are filled with "0".

The following table shows the contents of the accumulator after executing the "Load status word in accumulator 1" instruction:

	Accumulator 1									
Bit number	31-9	8	7	6	5	4	3	2	1	0
Content	0	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC

The instruction is always executed independently of the signal state of the status bit.

Note

The controllers of the S7-300 series (except S7-318) do not load the status bits /FC, STA and OR into accumulator 1. These locations are written with "0".

Example

The following example shows how the instruction works:

STL	Explanation
L STW	// Load status word in accumulator 1.
T "Tag_STW"	// Transfer the contents of accumulator 1 (status bits) to the operand "Tag_STW".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

LAR1: Load AR1 with contents of accumulator 1

Description

With the "Load AR1 with contents of accumulator 1" instruction you load address register 1 (AR1) with the contents of accumulator 1. The value of accumulator 1 must correspond to the format of a POINTER.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
L P#10.0	// Load pointer (P#10.0) in accumulator 1.
LAR1	// Load address register 1 with contents of accumulator 1.
L MW [AR1,P#4.0]	// Load MW14 in accumulator 1.

STL	Explanation
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR1,P#2.1]	// Check whether signal state of bit I12.1 is "1" and link AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

LAR1 <D>: Load AR1 with double word or area pointer

Description

Use the "Load AR1 with double word or area pointer" instruction to load address register 1 (AR1) with the contents of a double word or an area internal or area crossing pointer. The contents of the double word must match the format of an area pointer.

The pointer and the double word can address one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Parameters

The following table shows the parameters of the "Load AR1 with double word or area pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Input	DWORD, POINTER	D, M, L	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.0	// Load address register 1 with contents of the area internal pointer P#10.0.
L MW [AR1,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR1,P#2.1]	// Check whether signal state of bit I12.1 is "1" and link AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".
LAR1 MD20	// Load address register 1 with contents of the MD20 (MD20 = P#30.0)
A I [AR1,P#2.1]	// Check whether signal state of bit I32.1 is "1" and link AND with current RLO.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

LAR1 AR2: Load AR1 with contents of AR2

Description

Use the "Load AR1 with contents of AR2" instruction to load the contents of address register 2 (AR2) into address register 1 (AR1).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
LAR2 P#20.0	// Load address register 2 with contents of the area internal pointer P#20.0.
LAR1 AR2	// Copy contents of address register 2 into address register 1.
L MW [AR1,P#4.0]	// Load MW24 in accumulator 1.
T "Tag_2"	// Transfer accumulator 1 to "Tag_2".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

LAR2: Load AR2 with contents of accumulator 1

Description

With the "Load AR2 with contents of accumulator 1" instruction you load address register 2 (AR2) with the contents of accumulator 1. The value of accumulator 1 must correspond to the format of a POINTER.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
L P#10.0	// Load pointer (P#10.0) in accumulator 1.
LAR2	// Load address register 2 with contents of accumulator 1.
L MW [AR2,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.

STL	Explanation
A I [AR2,P#2.1]	// Check whether signal state of bit I12.1 is "1" and link AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

LAR2 <D>: Load AR2 with double word or area pointer

Description

Use the "Load AR2 with double word or area pointer" instruction to load address register 2 (AR2) with the contents of a double word or an area-internal or cross-area pointer. The contents of the double word must match the format of an area pointer (POINTER).

The pointer and the double word can address one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Load AR2 with double word or area pointer" instruction:

```
LAR2 <D>
```

Parameters

The following table shows the parameters of the "Load AR2 with double word or area pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Input	DWORD, POINTER	D, M, L	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

```
STL                          Explanation
LAR2 P#10.0                  // Load address register 2 with contents of the area
                             internal pointer P#10.0.
L MW [AR2,P#4.0]             // Load MW14 in accumulator 1.
L "Tag_Value"                // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load content of the operand "Tag_Value" in accumu-
                             lator 1.
>I                           // Compare whether the value of accumulator 2 is great-
                             er than the value of accumulator 1.
A I [AR2,P#2.1]              // Check whether signal state of bit I12.1 is "1" and
                             link AND with current RLO.
= "Tag_Output"               // If the condition is satisfied (RLO = "1"), set the
                             operand "Tag_Output" to "1".
LAR2 MD20                    // Load address register 2 with contents of the MD20
                             (MD20 = P#30.0)
A I [AR2,P#2.1]              // Check whether signal state of bit I32.1 is "1" and
                             link AND with current RLO.
```

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

Transfer

T: Transfer

Description

Use the "Transfer" instruction to transfer the contents of accumulator 1 in byte, word or double word format to a specified operand. The contents of accumulator 1 are not changed by this instruction. The specified operand must be addressed in one of the following memory areas:

- Process image of the inputs and outputs (I, Q)
- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)
- I/O (PQ)

The number of bytes to be transferred depends on the format of the specified operand. If the operand exists in byte format, the instruction transfers the contents of the bits 0 to 7 of accumulator 1.

If the operand exists in word format, the instruction transfers the contents of the word that is right-aligned in accumulator 1. The right byte of accumulator 1 is transferred to the higher addressed byte of the operand. The bits 8 to 15 of the accumulator 1 are transferred to the lower byte of the operand.

If the specified operand exists in double word format, the instruction transfers the entire content of accumulator 1. The right byte of accumulator 1 is transferred to the highest addressed byte of the operand. The bits 24 to 31 of accumulator 1 are transferred to the lowest addressed byte of the operand.

The following table shows an example of how the contents of accumulator 1 are transferred in byte, word and double word format:

Instruction	Accumulator 1			
Bits	31 24	23 16	15 8	7 0
Contents of accumulator 1	byte (n)	byte (n+1)	byte (n+2)	byte (n+3)
Transfer of a byte				byte (n+3)
Transfer of a word			byte (n+2)	byte (n+3)
Transferring a double word	byte (n)	byte (n+1)	byte (n+2)	byte (n+3)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The execution of the "Transfer" instruction is influenced by the Master Control Relay (MCR). When the MCR bit is set to "0", the value "0" is transferred to the specified operand.

Parameters

The following table shows the parameters of the "Transfer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	Bit strings, integers, timers, REAL, DATE, TOD, CHAR	I, Q, PQ, M, L, DB, DI	Operand to which the contents of accumulator 1 are transferred.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result_1"	// Transfer product (contents of Accumulator 1) to the operand "Tag_Result_1".
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
T "Tag_Result_2"	// Transfer sum (contents of Accumulator 1) to the operand "Tag_Result_2"

See also

Overview of the valid data types (Page 231)

Master control relay (Page 191)

Important notes on using MCR functions (Page 192)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

STL programming examples (Page 7280)

T STW: Transfer accumulator 1 to status word

Description

Use the "Transfer accumulator 1 to status word" instruction to transfer bits 0 to 8 of accumulator 1 to the status word.

The following table shows the bits in accumulator 1 that will overwrite status bits with processing of the instruction:

	Accumulator 1								
Bit number	8	7	6	5	4	3	2	1	0
Status bits	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC

The instruction is always executed independently of the signal state of the status bit.

Note

The bits of the status word /FC, STA and OR are not loaded by the "Transfer accumulator 1 to status word" instruction. Only bits 1, 4, 5, 6, 7 and 8 are loaded to the corresponding bit positions of the lower value word of accumulator 1.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_STW"`	`// Load value of the operand "Tag_STW" in accumulator 1.`
`T STW`	`// Transfer bits 0 to 8 of accumulator 1 to status word`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

CAR: Switch AR1 and AR2

Description

Use the instruction "Switch AR1 and AR2" to switch the contents of address registers 1 (AR1) and 2 (AR2).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.0	// Load address register 1 with contents of the area internal pointer P#10.0.
LAR2 P#20.0	// Load address register 2 with contents of the area internal pointer P#20.0.
CAR	// Switch contents of the address registers.
L MD [AR1,P#2.0]	// Load MD22 in accumulator 1.
T MD [AR2,P#2.0]	// Transfer MD12 too accumulator 2.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

TAR1: Transfer AR1 to accumulator 1

Description

With the "Transfer AR1 to accumulator 1" instruction you transfer the contents of address register 1 (AR1) to accumulator 1. The content of accumulator 1 is moved to accumulator 2 during this.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#15.0	// Load contents of the area-internal pointer P#15.0 into address register 1.
TAR1	// Transfer contents of address register 1 (P#15.0) to accumulator 1.

STL	Explanation
T "Tag_Pointer"	// Transfer contents of accumulator 1 to the operand "Tag_Pointer".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

TAR1 <D>: Transfer AR1 to double word

Description

Use the "Transfer AR1 to double word" instruction to transfer the contents of address register 1 (AR1) to a double word. The double word must hereby be addressed in one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Parameters

The following table shows the parameters of the "Transfer AR1 to double word" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Output	DWORD	D, M, L	Operand to which the contents of accumulator 1 are transferred.

Example

The following example shows how the instruction works:

STL	Explanation
TAR1 %DBD20	// Transfer contents of address register 1 to data double word DBD20.
TAR1 %DID30	// Transfer contents of address register 1 to instance double word DID30.

STL	Explanation
TAR1 %LD18	// Transfer contents of address register 1 to local data double word LD18.
TAR1 %MD24	// Transfer contents of address register 1 to memory double word MD24.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

TAR1 AR2: Transfer AR1 to AR2

Description

Use the "Transfer AR1 to AR2" instruction to copy the contents of address register 1 (AR1) to address register 2 (AR2).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.4	// Load contents of the area-internal pointer P#10.4 into address register 1.
TAR1 AR2	// Transfer contents of address register 1 to address register 2.
A I [AR2,P#3.2]	// Check whether signal state of bit I13.6 is "1" and AND with current RLO.
= "Tag1"	// Set operand "Tag_1" to "1"

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

TAR2: Transfer AR2 to accumulator 1

Description

With the "Transfer AR2 to accumulator 1" instruction you transfer the contents of address register 2 (AR2) to accumulator 1. The content of accumulator 1 is moved to accumulator 2 during this.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
`LAR2 P#20.0`	`// Load address register 2 with contents of the area internal pointer P#20.0.`
`TAR2`	`// Transfer contents of address register 2 (P#20.0) to accumulator 1.`
`T "Tag_Pointer"`	`// Transfer contents of accumulator 1 to the operand "Tag_Pointer".`

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

TAR2 <D>: Transfer AR2 to double word

Description

Use the "Transfer AR2 to double word" instruction to transfer the contents of address register 2 (AR2) to a double word. The double word must hereby be addressed in one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Parameters

The following table shows the parameters of the "Transfer AR2 to double word" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Output	DWORD	D, M, L	Operand to which the contents of accumulator 2 are transferred.

Example

The following example shows how the instruction works:

STL	Explanation
TAR2 %DBD20	// Transfer contents of address register 2 to data double word DBD20.
TAR2 %DID30	// Transfer contents of address register 2 to instance double word DID30.
TAR2 %LD18	// Transfer contents of address register 2 to local data double word LD18.
TAR2 %MD24	// Transfer contents of address register 2 to memory double word MD24.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

Conversion operations

BTI: Convert BCD to integer (16 bit)

Description

Use the "Convert BCD to integer (16 bit)" instruction to convert the value in the right word of accumulator 1 to a 16-bit integer. The instruction interprets the word that is to be converted as a three-digit binary coded decimal (BCD).

The bits 1 to 11 of accumulator 1 indicate the value that will be converted. Values in the range from "-999" to "+999" are permitted.

The sign of the result value is read from bit 15 of accumulator 1. If the signal state of the bit is "0", the sign is positive. If the signal state of the bit is "1," the sign is negative. Bits 12 through 14 are not used in the conversion.

If the BCD code contains a pseudo tetrad, the CPU reports a programming error and opens the organization block "OB121". If the organization block OB121 is not present, the CPU changes to STOP mode.

The result of the conversion is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load BCD in accumulator 1.`
`BTI`	`// Convert BCD to 16-bit integer.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:

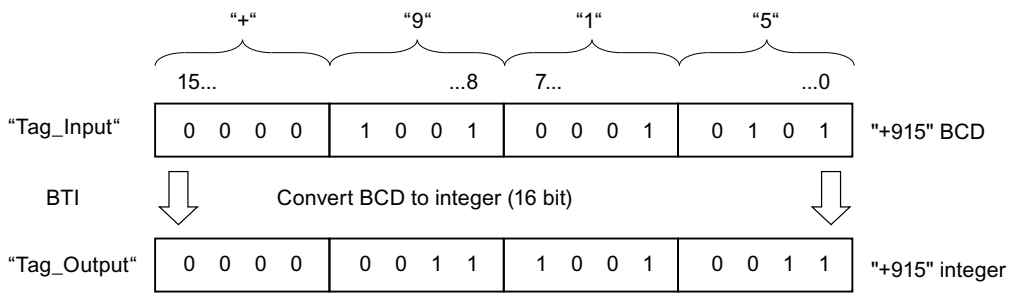

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

ITB: Convert integer (16 bit) to BCD

Description

Use the "Convert integer (16 bit) to BCD" instruction to convert the value in the right word of accumulator 1 to a three-digit binary-coded decimal (BCD). The instruction interprets the value that is to be converted as a 16-bit integer.

The bits 0 to 11 of accumulator 1 indicate the value that will be converted. Values in the range from "-999" to "+999" are permitted. If the value to be converted is outside this range, the bits OV and OS are set to signal state "1". In this case the conversion is not executed.

The sign of the result value is read from bits 12 through 15 of accumulator 1. If the signal state of the bits is "0", the sign is positive. If the signal state of all four bits is "1", the sign is negative.

The result of the conversion is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load integer (16 bit) in accumulator 1.`
`ITB`	`// Convert integer to BCD.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:

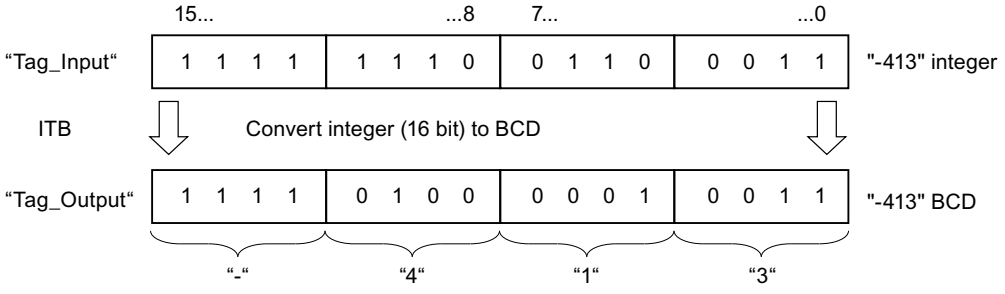

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with integers (Page 178)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

BTD: Convert BCD to integer (32 bit)

Description

Use the "Convert BCD to integer (32 bit)" to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a seven-digit binary coded binary coded decimal (BCD).

The bits 0 to 27 of accumulator 1 indicate the value that will be converted. Values in the range from "-9999999" through "+9999999" are permitted.

The sign of the result value is read from bit 31 of accumulator 1. If the signal state of the bit is "0", the sign is positive. If the signal state of the bit is "1," the sign is negative. Bits 28 through 30 are not used in the conversion.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

If the BCD code contains a pseudo tetrad, the CPU reports a programming error and opens the organization block "OB121". If the organization block OB121 is not present, the CPU changes to stop mode.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load BCD in accumulator 1.`
`BTD`	`// BCD is converted to 32-bit integer.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:

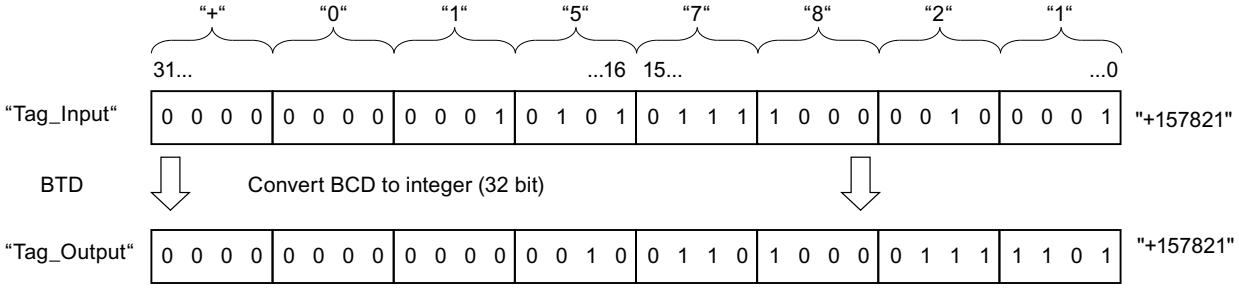

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

ITD: Convert integer (16 bit) to integer (32 bit)

Description

Use the "Convert integer (16 bit) to integer (32 bit)" instruction to convert the value in the right word of accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a 16-bit integer.

Bits 0 through 15 are not used in the conversion. The bits 16 through 31 of the result value are filled with the signal state of the bit 15.

The result value is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load integer (16 bit) in accumulator 1.
ITD                          // Convert integer (16 bit) to integer (32 bit).
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Accumulator 1							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Before processing	XXXX	XXXX	XXXX	XXXX	1111	1111	1111	0110
After processing	1111	1111	1111	1111	1111	1111	1111	0110
	X: The signal state of the bits is irrelevant for the conversion							

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

DTB: Convert integer (32 bit) to BCD

Description

Use the "Convert integer (32 bit) to BCD" instruction to convert the value in accumulator 1 to a seven-digit binary coded decimal (BCD). The instruction interprets the value that is to be converted as a 32-bit integer.

The bits 0 to 27 of accumulator 1 indicate the value that will be converted. Values in the range from "-9999999" through "+9999999" are permitted. If the value to be converted is outside this range, the bits OV and OS are set to signal state "1". In this case the conversion is not executed.

The sign of the result value is read from bits 28 through 31 of accumulator 1. If the signal state of the bits is "0", the sign is positive. If the signal state of all four bits is "1", the sign is negative.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load integer (32 bit) to accumulator 1`
`DTB`	`// Convert integer (32 bit) to BCD`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:

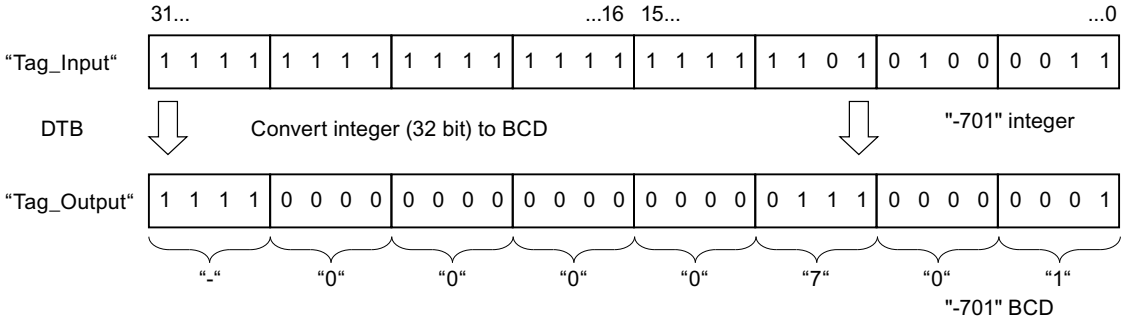

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with integers (Page 178)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

DTR: Convert integer (32 bit) to floating-point number

Description

Use the "Convert integer (32 bit) to floating-point number" instruction to convert the value in accumulator 1 to a floating-point number. The instruction interprets the value that is to be converted as a 32-bit integer.

Because a 32-bit integer has a higher accuracy than a floating-point number, the result of the conversion is rounded off to the next integer.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load integer (32 bit) to accumulator 1`
`DTR`	`// Convert integer (32 bit) to floating-point number.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:

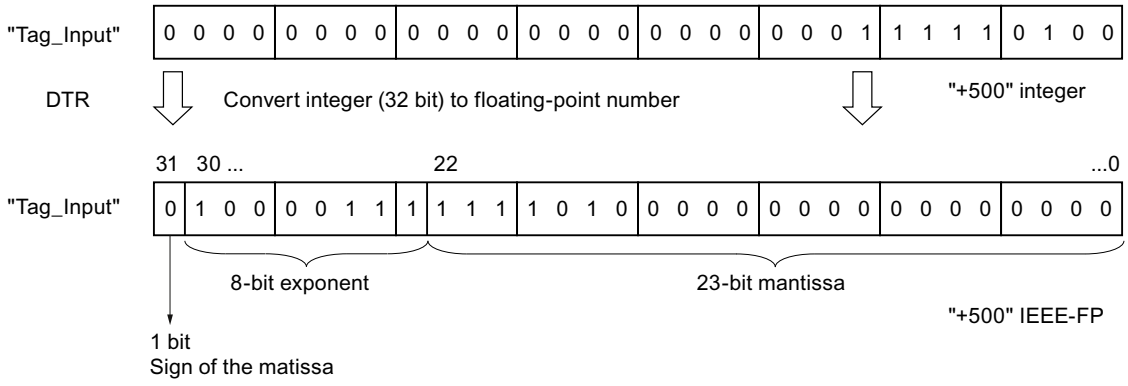

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

INVI: Create ones complement integer (16 bit)

Description

Use the "Create ones complement integer (16 bit)" instruction to negate the value in the right word of accumulator 1 bit for bit.

When executed, the instruction inverts the signal state of each individual bit in the right word of accumulator 1. The ones are replaced with zeros and the zeros with ones.

The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value in accumulator 1
INVI                         // Create ones complement
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0110	0011	1010	1110
Tag_Output	1001	1100	0101	0001

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

INVD: Create ones complement double integer (32 bit)

Description

Use the "Create ones complement double integer (32 bit)" to negate the value in accumulator 1 bit for bit.

During execution, the instruction inverts the signal state of each individual bit in accumulator 1. The ones are replaced with zeros and the zeros with ones.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value in accumulator 1
INVD                         // Create ones complement
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0110	1111	1000	1100	0110	0011	1010	1110
Tag_Output	1001	0000	0111	0011	1001	1100	0101	0001

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

NEGI: Negate integer (16 bit)

Description

Use the "Negate integer (16 bit)" instruction to change the sign of the value in the right word of accumulator 1 by forming the twos complement. The instruction interprets the value of accumulator 1 as a 16-bit integer.

The processing of the instruction is equal to a multiplication with "-1". The result of the instruction is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

The "Negate integer (16 bit)" instruction affects the status bits CC 0, CC 1, OV and OS. The following table shows how the instruction influences the status bit depending on the result:

Result:	CC 1	CC 0	OV	OS
+1 to +32767	1	0	0	-
0	0	0	0	-
-1 to -32767	0	1	0	-
(-) 32768	0	1	1	1

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value in accumulator 1
NEGI                         // Negate integer (16 bit)
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0101	1101	0011	1000
Tag_Output	1010	0010	1100	1000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

NEGD: Negate integer (32 bit)

Description

Use the "Negate integer (32 bit)" instruction to change the sign of the value in accumulator 1 by forming the twos complement. The instruction interprets the value of accumulator 1 as a 32-bit integer.

The processing of the instruction is equal to a multiplication with "-1". The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The "Negate integer (32 bit)" instruction affects the status bits CC 0, CC 1, OV and OS. The following table shows how the instruction influences the status bit depending on the result:

Result	CC 1	CC 0	OV	OS
+1 to +2147483647	1	0	0	-
0	0	0	0	-
-1 to -2147483647	0	1	0	-
(-) 2147483648	0	1	1	1

Example

The following example shows how the instruction works:

```
STL                         Explanation
L "Tag_Input"               // Load value in accumulator 1
NEGD                        // Negate integer (32 bit).
T "Tag_Output"              // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Status	Value							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0101	1111	0110	0100	0101	1101	0011	1000
Tag_Output	1010	0000	1001	1011	1010	0010	1100	1000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Setting the status bits in instructions with integers (Page 178)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

NEGR: Negate floating-point number

Description

With the "Negate floating-point number" instruction you invert the signal state of bit 31 in accumulator 1. The instruction interprets the value of accumulator 1 as a floating-point number. The sign of the mantissa is in bit 31 of the floating-point number.

The processing of the instruction is equal to a multiplication with "-1". After the processing of the instruction, the sign of the mantissa is inverted.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The instruction does not affect any status bits.

Example

The following example shows how the instruction works:

```
STL                         Explanation
L "Tag_Input"               // Load floating-point number into accumulator 1
NEGR                        // Negate floating-point number.
T "Tag_Output"              // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Input	1.5E+02
Tag_Output	-1.5E+02

See also

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

CAW: Switch bytes in the right word of accumulator 1

Description

Use the "Switch bytes in the right word of accumulator 1" instruction to switch the sequence of the two right bytes in the right word of accumulator 1.

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Bytes in accumulator 1			
Before processing	Value A	Value B	Value C	Value D
After processing	Value A	Value B	Value D	Value C

The result of the instruction is saved in the right word of accumulator 1. The bytes in the left word of accumulator 1 are not influenced by the instruction and remain unchanged.

Example

The following example shows how the instruction works:

```
STL                     Explanation
L "Tag_Input"           // Load value in accumulator.
CAW                     // Switch bytes in the right word of accumulator 1.
T "Tag_Output"          // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Input	0000	1111	0000	1111
Tag_Output	0000	1111	1111	0000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

CAD: Switch all bytes in accumulator 1

Description

Use the "Switch all bytes in accumulator 1" instruction to switch the sequence of the bytes in accumulator 1.

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Bytes in accumulator 1			
Before processing	Value A	Value B	Value C	Value D
After processing	Value D	Value C	Value B	Value A

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value of the operand in accumulator 1.
CAD                          // Switch bytes of accumulator 1.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Input	1111	0000	0000	1111	0000	0000	1111	1111
Tag_Output	1111	1111	0000	0000	0000	1111	1111	0000

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

RND: Round numerical value

Description

Use the "Round numerical value" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and round this to the next integer.

If the floating-point number is exactly between an even and odd number, the even number is selected as result.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load floating-point number into accumulator 1
RND                          // Round floating-point number to an integer.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	101.5	-101.5
Tag_Output	102	-102

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

TRUNC: Truncate numerical value

Description

Use the "Truncate numerical value" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value to be converted as a floating-point number and returns its whole number as result.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load floating-point number into accumulator 1`
`TRUNC`	`// Convert floating-point number to an integer of data type DINT.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	101.5	-101.5
Tag_Output	101	-101

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

RND+: Generate next higher integer from floating-point number

Description

Use the "Generate next higher integer from floating-point number" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and rounds this number to the next higher integer. The result of the instruction is a number of the data type DINT that is greater than or equal to the floating-point number to be converted.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load floating-point number into accumulator 1
RND+                         // Round floating-point number to next higher integer.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	100.5	-100.5
Tag_Output	101	-100

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

RND-: Generate next lower integer from floating-point number

Description

Use the "Generate next lower integer from floating-point number" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and rounds this number to the next lower integer. The result of the instruction is a number of the data type DINT that is less than or equal to the floating-point number to be converted.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load floating-point number into accumulator 1
RND-                         // Round floating-point number to next lower integer.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	100.5	-100.5
Tag_Output	100	-101

See also

Overview of the valid data types (Page 231)

Setting the status bits in instructions with floating-point numbers (Page 179)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Program control operations

Jumps

Jump label

Description

Use a jump label to specify the place in the program at which the program execution should resume after a jump. The name of the jump label can consist of up to 128 letters, numbers or underscores.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block. Each jump label can jump to several locations.

Both forward and backward jumps are possible. The maximum jump distance is -32768 or +32767 words of program code. The actual maximum number of instructions you can jump over depends on the mix of the statements used in your program (one, two, or three-word instructions).

If you use the MCR functions, the jump label must be located in the same MCR range as the jump instruction.

Adhere to the following grammatical rules for jump labels:

- Letters (a to z, A to Z)
- A combination of letters and numbers. Check that the order is correct, i.e. first the letters, then the numbers (a - z, A - Z, 0 - 9).
- Special characters or a combination of letters and numbers must not be used in reverse order, i.e. first the numbers and then the letters (0-9, a - z, A - Z).

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input_1"`	`// Load first comparison value.`
`L "Tag_Input_2"`	`// Load second comparison value.`
`>I`	`// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2"`
`JC MyLABEL`	`// With a scan result "1" jump to jump label "MyLABEL" and continue with program processing`
	`// If the result of the query is "0", execute the next instruction.`
`L "Tag_Input_3"`	`// Load content of the operand "Tag_Input_3" in accumulator 1.`
`T "Tag_Output"`	`// Load contents of accumulator 1 in operand "Tag_Output".`

STL	Explanation
MyLABEL:A "Tag_Input_4"	// If there was a jump, continue program execution at this point. // Query whether operand "Tag_Input_4" is "1"

See also

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JU: Unconditional jump

Description

Use the "Unconditional jump" instruction to interrupt the linear program execution and continue at the point that is marked by the specified jump label. The instruction is always executed independently of conditions.

The "Unconditional jump" instruction does not affect the status bits.

Parameters

The following table shows the parameters of the "Unconditional jump" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2"
JC MyLABEL_1	// With a scan result "1" jump to jump label "MyLABEL_1" and continue with program processing
	// If the result of the query is "0", continue linear program execution.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.

STL	Explanation
T "Tag_Output_1"	// Load contents of accumulator 1 in operand "Tag_Output_1".
JU MyLABEL_2	// Jump to jump label "MyLABEL_2" and resume execution of the program there.
MyLABEL_1: L "Tag_Input_4"	// Jump label
	// Load content of the operand "Tag_Input_4" in accumulator 1.
T "Tag_Output_2"	// Load contents of accumulator 1 in operand "Tag_Output_2".
MyLABEL_2: A "Tag_Input_5"	// Jump label "MyLABEL_2"
	// Query whether operand "Tag_Input_5" is "1"

See also

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

JC: Jump if RLO = 1

Description

Use the "Jump if RLO = 1" instruction to interrupt linear program execution depending on the result of logic operation and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed when the current RLO is "1". If the current RLO before the instruction is "0", the jump is not executed and the program is resumed with the next instruction.

The "Jump if RLO = 1" instruction sets the RLO and the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Parameters

The following table shows the parameters of the "Jump if RLO = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2"
JC MyLABEL	// If RLO = "1", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "0", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output_1"	// Load contents of accumulator 1 in operand "Tag_Output_1".
MyLABEL: L "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Load content of the operand "Tag_Input_4" in accumulator 1.
T "Tag_Output_2"	// Transfer contents of accumulator 1 to the operand "Tag_Output_2".

See also

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

JCN: Jump if RLO = 0

Description

Use the "Jump if RLO = 0" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed when the current RLO is "0". If the current RLO before the instruction is "1", the jump is not executed and the program is resumed with the next instruction.

The "Jump if RLO = 0" instruction sets the RLO and the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Parameters

The following table shows the parameters of the "Jump if RLO = 0" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query whether the value of operand "Tag_Input_1" is "1" and AND.
A "Tag_Input_2"	// Query whether the value of operand "Tag_Input_2" is "1" and AND.
JCN MyLABEL	// If RLO = "0", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "1", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output"	// Load contents of accumulator 1 in operand "Tag_Output".
MyLABEL: A "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Query whether the value of operand "Tag_Input_4" is "1" and AND with the RLO

See also

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

JCB: Jump if RLO = 1 and save RLO

Description

Use the "Jump if RLO = 1 and save RLO" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label. At the same time, the signal state of the current RLO is copied to the binary result (BR).

The jump to the specified jump label is only executed when the current RLO is "1". In this case the instruction set the binary result to "1".

If the current RLO before the instruction is "0", the jump is not executed and the program is resumed with the next instruction. In this case the instruction assigns the signal state "0" to the binary result.

The "Jump if RLO = 1 and save RLO" instruction sets the RLO and the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Parameters

The following table shows the parameters of the "Jump if RLO = 1 and save RLO" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input_1"              // Load first comparison value.
L "Tag_Input_2"              // Load second comparison value.
>I                           // Check whether the value of the operand "Tag_In-
                             put_1" is greater than the value of the operand
                             "Tag_Input_2"
JCB MyLABEL                  // Copy current RLO to BR.
                             // If RLO = "1", jump to jump label "MyLABEL" and con-
                             tinue program execution there.
                             // If the RLO = "0", execute the next instruction.
L "Tag_Input_3"              // Load content of the operand "Tag_Input_3" in accu-
                             mulator 1.
T "Tag_Output_1"             // Load contents of accumulator 1 in operand "Tag_Out-
                             put_1".
MyLABEL: L "Tag_Input_4"     // If there was a jump, continue program execution at
                             this point.
                             // Load content of the operand "Tag_Input_4" in accu-
                             mulator 1.
T "Tag_Output_2"             // Load contents of accumulator 1 in operand "Tag_Out-
                             put_2".
```

See also

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

JNB:: Jump if RLO = 0 and save RLO

Description

Use the "Jump if RLO = 0 and save RLO" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label. At the same time, the signal state of the current result of logic operation is copied to the binary result (BR).

The jump to the specified jump label is only executed when the current RLO is "0". In this case the instruction sets the binary result to "0".

If the current RLO before the instruction is "1", the jump is not executed and the program is resumed with the next instruction. In this case the instruction assigns the signal state "1" to the binary result.

The "Jump if RLO = 0 and save RLO" instruction sets the RLO and the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Parameters

The following table shows the parameters of the "Jump if RLO = 0 and save RLO" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query whether the value of operand "Tag_Input_1" is "1" and AND.
A "Tag_Input_2"	// Query whether the value of operand "Tag_Input_2" is "1" and AND.
JNB MyLABEL	// Copy current RLO to BR.
	// If RLO = "0", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "1", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output"	// Transfer contents of accumulator 1 to the operand "Tag_Output".
MyLABEL: A "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Query whether the value of operand "Tag_Input_4" is "1" and AND with the RLO

See also

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

JBI: Jump if BR = 1

Description

Use the "Jump if BR = 1" instruction to interrupt linear program execution depending on the binary result and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit BR is "1". If the status bit BR is "0", the jump is not executed and the program resumes with the next instruction.

The "Jump if BR = 1" instruction sets the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Parameters

The following table shows the parameters of the "Jump if BR = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
SET                          // Set RLO at the start of block to "1"
SAVE                         // Transfer signal state of the RLO to the BR bit.
....                         // Any program
AN OV                        // Query OV bit at end of block.
SAVE                         // If there is an overflow, set BR bit to "0".
                             // Set BR to "1" if there is no overflow.
JBI END                      // If BR = "1", jump to jump label END.
                             // If BR = "0", execute the next instruction.
R "Tag_Output_1"             // Reset operand "Tag_Output_1" to "0".
END: S "Tag_Output_2"        // Set operand "Tag_Output_2" to "1"
```

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JNBI: Jump if BR = 0

Description

Use the "Jump if BR = 0" instruction to interrupt the linear program execution depending on the binary result (BR) and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit BR is "0". If the status bit BR before the instruction is "1", the jump is not executed and the program resumes with the next instruction.

The "Jump if BR = 0" instruction sets the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Parameters

The following table shows the parameters of the "Jump if BR = 0" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
SET                          // Set RLO at the start of block to "1"
SAVE                         // Transfer signal state of the RLO to the BR bit.
....                         // Any program
AN OV                        // Query OV bit at end of block.
SAVE                         // If there is an overflow, set BR bit to "0".
                             // Set BR to "1" if there is no overflow.
JNBI END                     // If BR = "0", jump to jump label END.
                             // If BR = "1", execute the next instruction.
S "Tag_Output"               // Set operand "Tag_Output" to "1"
END: BE                      // End block
```

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JO: Jump if OV = 1

Description

Use the "Jump if OV = 1" instruction to interrupt linear program execution depending on the status bit OV and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit OV is "1". The status bit OV is set if, for example, the result of a calculation is outside the permitted range or an invalid value is returned during the comparison of floating-point numbers.

If the status bit OV is "0", the jump is not executed and the program continues with the next instruction.

Parameters

The following table shows the parameters of the "Jump if OV = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value
L "Tag_Value_2"              // Load value
*I                           // Multiply values
JO OVER                      // If the status bit OV has signal state "1", jump to
                             jump label "OVER" and continue program execution
                             there.
                             // If the signal state of the status bit OV is "0",
                             execute the next instruction.
T "Tag_Result"               // Transfer the result of the multiplication to the
                             operand "Tag_Result".
JU NEXT                      // Jump to jump label "NEXT" and resume execution of
                             the program there.
```

STL	Explanation
OVER: SET	// Continue program execution at this point if the condition (OV = "1") is satisfied.
	// Set RLO to signal state "1"
R "Tag_Output"	// Reset operand "Tag_Output" to "0".
NEXT: A "MyTag_1"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_1" is "1" and AND.
A "MyTag_2"	// Query whether the value of operand "MyTag_2" is "1" and AND.
S "Tag_Output_2"	// If RLO = 1, set the "Tag_Output_2" operand to "1".

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JOS: Jump if OS = 1

Description

Use the "Jump if OS = 1" instruction to interrupt linear program execution depending on the status bit OS and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit OS is "1". The status bit OS is always set when a number range overflow sets the status bit OV to "1". In contrast to status bit OV, the status bit OS remains set even if a result is subsequently in the permitted number range.

If the status bit OS is "0", the jump is not executed and the program continues with the next instruction.

Parameters

The following table shows the parameters of the "Jump if OS = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value
L "Tag_Value_2"	// Load value
*I	// Multiply loaded values.
	// Store product in accumulator 1.
L "Tag_Value_3"	// Load value
+I	// Add loaded value to the product.
	// Store product in accumulator 1.
L "Tag_Value_4"	// Load value
-I	// Subtract loaded value from the calculated sum.
	// Store product in accumulator 1.
JOS OVER	// If there is an overflow in one of the three preceding instructions, the status bit OS is set to "1"
	// If the signal state of the status bit OS is "1", the program resumes at the jump label "OVER".
	// If the signal state of the status bit OS is "0", the next instruction is executed
T "Tag_Result"	// Transfer the result of the entire calculation to the operand "Tag_Result"
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
OVER: SET	// Continue program execution at this point if the condition (OV = "1") is satisfied.
	// Set RLO to signal state "1"
R "Tag_Output"	// Reset operand "Tag_Output" to "0".
NEXT: A "MyTag_1"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_1" is "1" and AND.
A "MyTag_2"	// Query whether the value of operand "MyTag_2" is "1" and AND.
S "Tag_Output_2"	// If RLO = 1, set the "Tag_Output_2" operand to "1".

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JZ: Jump if the result is zero

Description

Use the "Jump if the result is zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bits CC 0 and CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, accumulator 1 has the value "0".
- A negative overflow occurs when executing the "Add integers (16 bit)" (+I) or "Add double integers (32 bit)" (+D) instruction.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- The content of accumulator 2 is equal to the content of accumulator 1 after the execution of a comparison instruction.
- The content of accumulator 1 is zero after the execution of a word logic operation.
- The value of the last shifted out bit is "0" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load value of the operand "Tag_Value".
SRW 1                        // Shift the contents of the right word in accumulator
                             1 one place to the right.
JZ ZERO                      // If the signal state of the last bit shifted is "0",
                             jump to jump label "OVER" and continue with program
                             processing from there
                             // If the signal state of the last bit shifted is "1",
                             execute the next instruction.
L "MyTag_1"                  // Load value of the operand "MyTag_1" in accumulator
                             1
INC 1                        // Add right-hand byte of accumulator 1 with the value
                             "1".
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
```

STL	Explanation
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
ZERO: L "MyTag_2"	// Jump label "ZERO"
	// Load value of the operand "MyTag_2" in accumulator 1
INC 1	// Add right-hand byte of accumulator 1 with the value "1".
T "Tag_Result_2"	// Transfer the result to operand "Tag_Result_2".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JN: Jump if the result is not zero

Description

Use the "Jump if the result is not zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal states of the status bits CC 0 and CC 1 are different. The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is not equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when executing a math instruction with integers (+I, -I, *I, /I, +D, -D, *D, /D, NEGI, NEGD).
- A positive or negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The content of accumulator 2 is not equal to the content of accumulator 1 after the execution of a comparison instruction.

- The content of accumulator 1 is not zero after the execution of a word logic operation.
- The value of the last shifted out bit is "1" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is not zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Valu_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
XOW                          // Apply EXCLUSIVE OR logic operation
JN NOZERO                    // If the value in accumulator 1 is not equal to zero,
                             the jump to the "NOZERO" jump label is performed and
                             program processing continues from there.
                             // If the value in accumulator 1 is equal to zero, the
                             next instruction will be processed.
AN "MyTag_1"                 // Query whether the value of operand "MyTag_1" is "0"
                             and AND.
S "Tag_Output_1"             // If RLO is "1", set the operand "Tag_Output_1" to
                             "1".
JU NEXT                      // Jump to jump label "NEXT" and resume execution of
                             the program there.
NOZERO: AN "MyTag_2"         // Jump label "NOZERO"
                             // Query whether the value of operand "MyTag_2" is "0"
                             and AND.
S "Tag_Output_2"             // If RLO is "1", set the operand "Tag_Output_2" to "1"
NEXT: A "MyTag_3"            // Jump label "NEXT"
                             // Query whether the value of operand "MyTag_3" is "1"
                             and AND.
A "MyTag_4"                  // Query whether the value of operand "MyTag_4" is "1"
                             and AND.
```

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JP: Jump if the result is greater than zero

Description

Use the "Jump if the result is greater than zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is executed only if the signal state of status bit CC 0 is "0" and CC 1 is "1". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is greater than zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (*I, /I, *D, /D).
- A positive overflow occurs when processing an arithmetic instruction with floating-point numbers.
- After the execution of a comparison instruction, the content of accumulator 2 is greater than the content of accumulator 1.
- The contents of accumulator 1 are not zero after the execution of a word logic operation.
- The value of the last shifted out bit is "1" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is greater than zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
```

STL	Explanation
-I	// Subtract value of the operand "Tag_Value_2" from the value of the operand "Tag_Value_1".
JP POSITIVE	// At a positive result value in accumulator "1" jump to jump label "POSITIVE" and continue program processing from there
	// If the result in accumulator 1 is negative, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO is "1", set the operand "Tag_Output_1" to "1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
POSITIVE: AN "MyTag_2"	// Jump label "POSITIVE"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO is "1", set the operand "Tag_Output_2" to "1".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JM: Jump if the result is less than zero

Description

Use the "Jump if the result is less than zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is executed only if the signal state of status bit CC 0 is "1" and CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is less than zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (*I, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (+I, -I, +D, -D, NEGI, NEGD).
- A negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- After the execution of a comparison instruction, the content of accumulator 2 is less than the content of accumulator 1.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is less than zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
-I                           // Subtract value of the operand "Tag_Value_2" from
                             the value of the operand "Tag_Value_1".
JM NEGATIVE                  // At a negative result value in accumulator "1" jump
                             to jump label "NEGATIVE" and continue program process-
                             ing from there
                             // If the result in accumulator 1 is positive, execute
                             the next instruction.
AN "MyTag_1"                 // Query whether the value of operand "MyTag_1" is "0"
                             and AND.
S "Tag_Output_1"             // If RLO is "1", set the operand "Tag_Output_1" to
                             "1".
JU NEXT                      // Jump to jump label "NEXT" and resume execution of
                             the program there.
NEGATIVE: AN "MyTag_2"       // Jump label "NEGATIVE"
                             // Query whether the value of operand "MyTag_2" is "0"
                             and AND.
S "Tag_Output_2"             // If RLO is "1", set the operand "Tag_Output_2" to
                             "1".
```

STL	Explanation
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JPZ: Jump if the result is greater than or equal to zero

Description

Use the "Jump if the result is greater than or equal to zero" instruction to interrupt linear program execution depending on the status bit CC 0 and continue it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bit CC 0 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is greater than or equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (*I, /I, *D, /D).
- A positive overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- After the execution of a comparison instruction, the content of accumulator 2 is greater than or equal to the content of accumulator 1.
- An instruction for word logic operation was executed.
- A shift instruction was executed.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is greater than or equal to zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
-I                           // Subtract value of the operand "Tag_Value_2" from
                             the value of the operand "Tag_Value_1".
JPZ REGULAR                  // At a positive result value or null in accumulator
                             "1" jump to jump label "REGULAR" and continue program
                             processing from there
                             // If the result in accumulator 1 is negative, execute
                             the next instruction.
AN "MyTag_1"                 // Query whether the value of operand "MyTag_1" is "0"
                             and AND.
S "Tag_Output_1"             // If RLO is "1", set the operand "Tag_Output_1" to
                             "1".
JU NEXT                      // Jump to jump label "NEXT" and resume execution of
                             the program there.
REGULAR: AN "MyTag_2"        // Jump label "REGULAR"
                             // Query whether the value of operand "MyTag_2" is "0"
                             and AND.
S "Tag_Output_2"             // If RLO is "1", set the operand "Tag_Output_2" to
                             "1".
NEXT: A "MyTag_3"            // Jump label "NEXT"
                             // Query whether the value of operand "MyTag_3" is "1"
                             and AND.
A "MyTag_4"                  // Query whether the value of operand "MyTag_4" is "1"
                             and AND.
```

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JMZ: Jump if the result is less than or equal to zero

Description

Use the "Jump if the result is less than or equal to zero" instruction to interrupt linear program execution depending on the status bit CC 1 and continue it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bit CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is less than or equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, *I, +D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (+I, -I, +D, -D, NEGI, NEGD).
- A negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- After the execution of a comparison instruction, the content of accumulator 2 is less than or equal to the content of accumulator 1.
- The value of accumulator 1 is equal to zero after processing a word logic.
- The value of the last shifted bit is equal to zero after processing a write instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is less than or equal to zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
-I                           // Subtract value of the operand "Tag_Value_2" from
                             the value of the operand "Tag_Value_1".
```

STL	Explanation
JMZ MyLABEL	// At a negative result value or null in accumulator "1" jump to jump label "MyLABEL" and continue program processing from there
	// If the result in accumulator 1 is positive, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO = "1", set the operand "Tag_Output_1" to "1"
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL: AN "MyTag_2"	// Jump label "MyLABEL"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO="1" set the operand "Tag_Output_2" to "1"
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JUO: Jump if the result is invalid

Description

Use the "Jump if the result is invalid" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bits CC 0 and CC 1 is "1". The situation occurs when one of the following conditions applies:

- It is divided by zero with a mathematical instruction (/I, /D, MOD).
- When processing an arithmetic instruction with floating-point numbers an overflow occurs and the result value is an invalid floating-point number.
- When processing a comparative instruction with floating-point numbers an invalid floating-point number was used or occurred as a result value.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Parameters

The following table shows the parameters of the "Jump if the result is invalid" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
/I                           // Divide value of the operand "Tag_Value_1" by value
                             of the operand "Tag_Value_2".
JUO ERROR                    // At a division by zero, jump to jump label "ERROR"
                             and continue with program processing from there
                             // Otherwise execute next instruction.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand
                             "Tag_Result".
A "MyTag_1"                  // Query whether the value of operand "MyTag_1" is "1"
                             and AND.
R "MyTag_1"                  // If RLO = "1", reset the operand "MyTag_1" to "0"
JU NEXT                      // Jump to jump label "NEXT" and resume execution of
                             the program there.
ERROR: AN "MyTag_1"          // Jump label "ERROR"
                             // Query whether the value of operand "MyTag_1" is "0"
                             and AND.
S "MyTag_1"                  // If RLO is "1", set the operand "MyTag_1" to "1".
NEXT: A "MyTag_3"            // Jump label "NEXT"
                             // Query whether the value of operand "MyTag_3" is "1"
                             and AND.
A "MyTag_4"                  // Query whether the value of operand "MyTag_4" is "1"
                             and AND.
```

See also

Basic information on the status word (Page 173)

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Setting status bits (Page 175)

JL: Define jump list

Description

Use the instruction "Define jump list" to program a list that consists of several entries of the instruction "Unconditional jump" (JU). The list begins directly after the instruction "Define jump list" and may contain up to 255 entries. The numbering of the jump functions in the list begins at zero. The jump list must be programmed without any gaps. The end of the jump list is marked by a jump label that is specified for the "Define jump list" instruction.

Which jump function of the list executes, depends on the value in the right byte of accumulator 1. If, for example, the accumulator 1 has the value "0", the first jump function executes. If the value in accumulator 1 is "1", the second jump function executes. If the value in accumulator 1 is greater than the number of list entries, the "Define jump list" instruction references the end of the list.

The "Jump distributor" instruction is processed independently of conditions and does not affect the status bits.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load jump number in accumulator 1.
JL END	// Start of the jump list
JU MyLABEL_1	// If the value is "0", execute in accumulator 1 and jump to jump label "MyLABEL_1"
JU MyLABEL_2	// If the value is "1", execute in accumulator 1 and jump to jump label "MyLABEL_2"
JU MyLABEL_3	// If the value is "2", execute in accumulator 1 and jump to jump label "MyLABEL_3"
END: L "MyTag_1"	// End of the jump list
	// Load content of the operand "MyTag_1" in accumulator 1.
L "MyTag_2"	// Load content of the operand "MyTag_2" in accumulator 1.
+I	// Add values
T "Tag_Output_2"	// Load contents of accumulator 1 in operand "Tag_Output_2".
MyLABEL_1: ...	// Jump label "MyLABEL_1"
....	// Any program
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL_2: ...	// Jump label "MyLABEL_2"
....	// Any program

STL	Explanation
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL_3: ...	// Jump label "MyLABEL_3"
....	// Any program
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
NEXT: ...	// Jump label "NEXT"

See also

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

LOOP: Loop

Description

Use the "Loop" instruction to program loops in the program.

The instruction interprets the right word of accumulator 1 as an unsigned 16-bit integer in the range from 0 to 65535. During execution the instruction first decrements the content of accumulator 1 by one. If the value in accumulator 1 after the decrement is not zero, the jump is executed at the specified jump label. If the value is zero, the jump is not executed and the processing of the program is resumed with the next instruction. The value in accumulator 1 therefore specifies the number of program loops that have to be executed. This number has to be saved in a loop counter.

The operand <number> contains the number of the loop cycles. The operand <Loop counter> contains the number of the loop cycles still to be executed. At the end of the program loop, the content of the loop counter is loaded in accumulator 1 and decremented by one by the instruction. The jump to the jump label at the start of the program loop is executed if the value in the accumulator is not equal to zero after the decrement.

The "Loop" instruction does not affect any status bits.

Note

If the loop counter is not preset or is preset with the value "0", the loop counter is decremented further and further into the negative, resulting in an endless loop that can change the CPU to STOP mode.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load number of time loop will execute into the right word of accumulator 1.
START: T "Tag_Counter"	// Start of the jump list
	// Transfer contents of accumulator 1 to the loop counter.
L "MyTag_1"	// Load value of the operand "MyTag_1".
L "MyTag_2"	// Load value of the operand "MyTag_2".
*D	// Multiply values
T "MyTag_1"	// Transfer the result of the multiplication to the operand "MyTag_1".
L "Tag_Counter"	// Load value of the loop counter into accumulator 1.
LOOP START	// Decrement value in accumulator 1 by one.
	// Jump to the start of the program loop if the value is not equal to zero.
	// If the value is zero, execute the next instruction.
L "MyTag_2"	// Load value of the operand "MyTag_2".
L 100	// Load the value 100.
>I	// Compare whether the value of the operand "MyTag_2" is greater than 200.
= "MyTag_3"	// Write comparison result to the operand "MyTag_3".

See also

Setting status bits in program control operation instructions (Page 180)

Querying status bits with jump instructions (Page 184)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Data blocks

OPN: Open global data block

Description

You can open a global data block (DB) with the "Open global data block" instruction. The number of the data block is transferred to the DB register. Subsequent DB commands access the relevant blocks depending on the content of the register.

The "Open global data block" instruction is processed without any conditions and does not affect either the result of logic operation or the accumulator contents.

Parameters

The following table shows the parameters of the "Open global data block" instruction:

Operand	Declaration	Data block type	Description
<data block>	-	DB	Data block that will be opened.

Example

The following example shows how the instruction works:

```
STL                                  Explanation
OPN "GlobalDataBlock"                // Open global data block.
L %DBW0                              // Load data word DBW0 of the opened global
                                     data block into accumulator 1.
T "MyTag"                            // Transfer contents of accumulator 1 to the
                                     operand "MyTag_1".
```

See also

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

OPNDI: Open instance data block

Description

With the "Open instance data block" instruction, you open an instance data block (DI). The number of the data block is transferred to the DI register. Subsequent DI commands access the relevant blocks depending on the content of the register.

The "Open instance data block" instruction is processed without any conditions and does not affect either the result of logic operation or the accumulator contents.

Parameters

The following table shows the parameters of the "Open instance data block" instruction:

Operand	Declaration	Data block type	Description
<data block>	-	DI	Data block that will be opened.

Example

The following example shows how the instruction works:

STL	Explanation
OPNDI "InstanceDataBlock"	// Open instance data block
L %DIW0	// Load data word DIW0 of the opened instance data block into accumulator 1.
T "MyTag"	// Transfer contents of accumulator 1 to the operand "MyTag".

See also

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

CDB: Swap data block register

Description

Use the "Swap data block register" instruction to swap the contents of the data block register. The instruction executes regardless of conditions and does not affect the status bits.

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block
OPNDI "InstanceDataBlock"	// Open instance data block
CDB	// Swap data block register
	// The DB register points to "InstanceDataBlock" and the DI register points to "GlobalDataBlock".
L %DIW0	// Load DW0 from "GlobalDataBlock" into accumulator 1.

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

L DBLG: Load the length of a global data block into accumulator 1

Description

With the "Load the length of a global data block into accumulator 1" instruction you load the length of a global data block opened by means of the data block register into accumulator 1. The previous content of accumulator 1 is moved to accumulator 2.

If no global data block was opened via the data block register prior to the processing of the instruction, the value "0" is loaded into accumulator 1. The length of the data block is equal to the number of the data bytes.

The status bits are not influenced by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block
L DBLG	// Load the length of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
<D	// Compare whether the length of the data block is less than the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

L DBNO: Load the number of a global data block into accumulator 1

Description

With the "Load the number of a global data block into accumulator 1" instruction you load the number of a global data block opened by means of the data block register into accumulator 1. The previous content of accumulator 1 is moved to accumulator 2.

If no global data block was opened via the data block register prior to the processing of the instruction, the value "0" is loaded into accumulator 1.

The status bits are not influenced by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block
L DBNO	// Load the number of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
==I	// Compare whether the number of the data block is equal to the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

L DILG: Load the length of an instance data block into accumulator 1

Description

With the "Load the length of an instance data block into accumulator 1" instruction you load the length of an instance data block opened by means of the data block register into accumulator 1. The previous content of accumulator 1 is moved to accumulator 2.

If no instance data block was opened via the data block register prior to the instruction executing, the value "0" is loaded into accumulator 1. The length of the data block is equal to the number of the data bytes.

The status bits are not influenced by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
OPNDI "InstanceDataBlock"	// Open instance data block
L DILG	// Load the length of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
<I	// Compare whether the length of the data block is less than the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

L DINO: Load the number of an instance data block into accumulator 1

Description

With the "Load the number of an instance data block into accumulator 1" instruction you load the number of an instance data block opened by means of the data block register into accumulator 1. The previous content of accumulator 1 is moved to accumulator 2.

If no instance data block was opened via the data block register prior to the instruction executing, the value "0" is loaded into accumulator 1.

The status bits are not influenced by the instruction.

Example

The following example shows how the instruction works:

STL	Explanation
OPNDI "InstanceDataBlock"	// Open instance data block
L DINO	// Load the number of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
==I	// Compare whether the number of the data block is equal to the value of the operand "MyTag_1".
= "MyTag_2"	// Assign query result to the "MyTag_2" operand.

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Code blocks

BE: Block end

Description

Use the "Block end" instruction to end the block currently being processed and go to the point in the program where the block is called. The program processing is then resumed with the instruction that is located directly after the block call.

When the block ends the current result of logic operation is transferred to the calling block.

The "Block end" instruction is executed independently of conditions. If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SIN	// Form sine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand "Tag_Result".
BE	// End block

See also

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

BEC: Conditional block end

Description

Use the "Conditional block end" instruction to end the block currently being processed depending on the result of logic operation (RLO) and to go to point in the program where the block is called.

If the RLO is "1", the instruction is executed. The block currently being processed is terminated and the program processing is resumed in the calling block. The program processing is then resumed with the instruction that is located directly after the block call.

When the block ends the current result of logic operation is transferred to the calling block. The MCR dependency establishes itself in the calling block in the same way as it was before the block call. The data blocks that were open when the block was called are re-opened.

If the RLO is "0" when the instruction is processed, the instruction is not executed. In this case the CPU sets the RLO to "1" and executes the next instruction.

If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A "Tag_Input"	// Query the value of operand "Tag_Input" for "1" and AND with the RLO.
BEC	// End block if condition satisfied (RLO = "1").
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand "Tag_Result".

See also

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

BEU: Unconditional block end

Description

Use the "Unconditional block end" instruction to end the block currently being processed and go to the point in the program where the block is called. The program processing is then resumed with the instruction that is located directly after the block call.

When the block ends the current result of logic operation is transferred to the calling block. The MCR dependency establishes itself in the calling block in the same way as it was before the block call. The data blocks that were open when the block was called are re-opened.

The "Unconditional block end" instruction is executed independently of conditions. In contrast to the "Block end" instruction, you can program the "Unconditional block end" instruction more than once in a block. If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A "Tag_Input"	// Query the value of operand "Tag_Input" for "1" and AND with the RLO.
JC NEXT	// If the condition is satisfied (RLO = "1"), continue program execution at jump label"NEXT".
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
BEU	// End block
NEXT: T "Tag_Result"	// Jump label "NEXT"
	// Transfer contents of accumulator 1 to the operand "Tag_Result".

See also

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

CALL: Call block

Description

Use the "Call block" instruction to call the following block types in the program:

- Functions
- Function blocks

The "Call block" instruction executes without any conditions. After processing of the instruction, program execution is resumed in the called block.

You can feed a called block with data. The data is transferred via the block parameters. The parameters of the called block are listed according to the call instruction in the called block. You can assign the required current parameters to these parameters. During execution of the "Call block" instruction, the data is transferred to the called block. By transferring the data, the content of the status word and the address and data block registers are changed.

Those parameters of a function block that are not fed with parameters, retain their current value. All parameters must be fed when functions are called. If a called block has no parameters, no parameter list is displayed.

If the called block requires an instance data block, give it to the block, separated by a comma when called. The specified data block must be created before the call.

When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

The MCR dependency establishes itself in the calling block in the same way as it was before the block call. The data blocks that were open when the block was called are re-opened.

Example

The following example shows how the instruction works:

STL	Explanation
CALL "MyFunction"	// Call function "MyFunction"
Input_1 := "Tag_Input_1",	// Assign actual parameters
Input_2 := "Tag_Input_2"	
Output_1 := "Tag_Output_1"	
Output_2 := "Tag_Output_1"	
CALL "MyFunctionBlock", "MyFB_DB"	// Call function block "MyFunctionBlock"
Value_1 := "Tag_Value_1"	// Assign actual parameters
Value_2 := "Tag_Value_2"	
Output := "Tag_Output"	
CALL "LIMIT"	// Call "Set limit value" instruction
MN := "Tag_LowLimit"	// Assign actual parameters
IN := "Tag_InputValue"	
MX := "Tag_HighLimit"	
OUT := "Tag_Output"	

STL	Explanation
CALL "CTU", "CTU_DB"	// Call "Count up" counter
CU := "Tag_StartCTU"	// Assign actual parameters
R := "Tag_ResetCounter"	
PV := "Tag_PresetValue"	
Q := "Tag_CounterStatus"	
CV := "Tag_CounterValue"	

See also

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

CC: Conditional block call

Description

Use the "Conditional block call" instruction to call functions (FCs) and function blocks (FBs) depending on the result of logic operation (RLO).

The instruction is only executed if the current result of logic operation (RLO) before execution of the instruction is "1". After processing of the instruction, program execution is resumed in the called block. When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

The status bits OS and /FC are reset to "0" when there is a block change. The status bits CC 0, CC 1 and OV are not affected.

The MCR dependency establishes itself in the calling block in the same way as it was before the block call. The data blocks that were open when the block was called are re-opened.

The contents of the accumulators and the address registers are not changed by the "Conditional block call" instruction.

If the current RLO is "0", the instruction as well as the block call are not executed and the RLO is set to "1".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the value of operand "Tag_Input_1" for "1" and AND with the current RLO.

STL	Explanation
CC "MyFunction"	// If condition is satisfied (RLO = "1"), call the function MyFunction.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
A "Tag_Input_2"	// Query the value of operand "Tag_Input_2" for "1" and AND with the current RLO.
CC "MyFunctionBlock"	// If condition is satisfied (RLO = "1"), call the function block MyFunctionBlock.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.

See also

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

UC: Unconditional block call

Description

You can use the "Unconditional block call" instruction to call functions (FCs) and function blocks (FBs) that have no parameters and no instance data blocks.

The instruction is executed regardless of conditions. After processing of the instruction, program execution is resumed in the called block. When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

The status bits OS and /FC are reset to "0" when there is a block change. The status bits CC 0, CC 1 and OV are not affected.

The MCR dependency establishes itself in the calling block in the same way as it was before the block call. The data blocks that were open when the block was called are re-opened.

The contents of the accumulators and the address registers are not changed by the "Unconditional block call" instruction.

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the value of operand "Tag_Input_1" for "1" and AND with the current RLO.
UC "MyFunction"	// Call function "MyFunction".

STL	Explanation
A "Tag_Input_2"	// Query the value of operand "Tag_Input_2" for "1" and AND with the current RLO.
UC "MyFunctionBlock"	// Call function block "MyFunctionBlock".

See also

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Word logic operations

AW: AND logic operation word by word

Description

Use the instruction "AND logic operation word by word" to AND the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" if both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

The following table shows how the result is formed by an AND word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	0	0	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the "AND logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is ANDed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into the right word of
                             accumulator 1.
AW W#16#F6B5                 // AND the value of the right word in accumulator 1
                             with the constant (W#16#F6B5).
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into the right word of
                             accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" into the
                             right word of accumulator 1.
AW                           // AND the value of the right word in accumulator 2
                             ("Tag_Value_2") with the value of the right word in
                             accumulator 1 ("Tag_Value_3")
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	0101	0000	0011	0001
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	0100	1000	0000	0010

See also

Overview of the valid data types (Page 231)

Setting the status bit for word logic operations (Page 182)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

OW: OR logic operation word by word

Description

Use the instruction "OR logic operation word by word" to OR the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both bits of the logic operation have signal state "0", the corresponding result bit is reset to "0".

The following table shows how the result is formed by an OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the "OR logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is ORed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into the right word of
                             accumulator 1.
```

STL	Explanation
OW W#16#F6B5	// OR the value of the right word in accumulator 1 with the constant (W#16#F6B5).
	// Save the result in the right word of accumulator 1.
T "Tag_Result_1"	// Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"	// Load value of the operand into the right word of accumulator 1.
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" into the right word of accumulator 1.
OW	// OR the value of the right word in accumulator 2 ("Tag_Value_2") with the value of the right word in accumulator 1 ("Tag_Value_3")
	// Save the result in the right word of accumulator 1.
T "Tag_Result_2"	// Transfer the result to operand "Tag_Result_2".

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	1111	1111	1011	1111
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	1111	1110	1011	1011

See also

Overview of the valid data types (Page 231)

Setting the status bit for word logic operations (Page 182)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

STL programming examples (Page 7280)

XOW: EXCLUSIVE OR logic operation word by word

Description

Use the instruction "EXCLUSIVE OR logic operation word by word" to EXCLUSIVE OR the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" if the signal state of the two bits to be linked is different. If both of the bits of the logic operation have the same signal state, the corresponding result bit is reset to "0".

The following table shows how the result is formed by an EXCLUSIVE OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	0

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is XORed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into the right word of
                             accumulator 1.
XOW W#16#F6B5                // XOR the value of the right word in accumulator 1
                             with the constant (W#16#F6B5).
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into the right word of
                             accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" into the
                             right word of accumulator 1.
XOW                          // XOR the value of the right word in accumulator 2
                             ("Tag_Value_2") with the value of the right word in
                             accumulator 1 ("Tag_Value_3")
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	1010	1111	1000	1110
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	1011	0110	1011	1001

See also

Overview of the valid data types (Page 231)

Setting the status bit for word logic operations (Page 182)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

AD: AND logic operation double word by double word

Description

Use the instruction "AND logic operation double word by double word" to AND the contents of accumulator 1 with the contents of accumulator 2 bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" if both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

The following table shows how the result is formed by an AND word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	0	0	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the instruction "AND logic operation double word by double word":

Parameters	Data type	Description
<constant>	DWORD	Value that is ANDed with the value in accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
AD DW#16#39C657AC            // Link the value in accumulator 1 with the constant
                             (DW#16#39C657AC) according to AND.
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in accumu-
                             lator 1.
AD                           // AND the value of accumulator 2 Tag_Value_2 with the
                             value of accumulator 1 ("Tag_Value_3")
                             // Store result in accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0001	1001	0100	0100	0001	0101	0010	1000
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0010	0001	0100	0010	0101	0101	0010	1010

See also

Overview of the valid data types (Page 231)

Setting the status bit for word logic operations (Page 182)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

OD: OR logic operation double word by double word

Description

Use the instruction "OR logic operation double word by double word" to OR the contents of accumulator 1 with the contents of accumulator 2 bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both bits of the logic operation have signal state "0", the corresponding result bit is reset to "0".

The following table shows how the result is formed by an OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the "OR logic operation double word by double word" instruction:

Parameters	Data type	Description
<constant>	DWORD	Value that is ORed with the value in accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
OD DW#16#39C657AC            // Link the value in accumulator 1 with the constant
                             (DW#16#39C657AC) according to OR.
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
```

STL	Explanation
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1.
OD	// OR the value of accumulator 2 Tag_Value_2 with the value of accumulator 1 ("Tag_Value_3")
	// Store result in accumulator 1
T "Tag_Result_2"	// Transfer the result to operand "Tag_Result_2".

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39 C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0111	1111	1110	0110	1101	1111	1011	1111
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0111	1101	1100	0111	0101	1111	1110	1111

See also

Overview of the valid data types (Page 231)

Setting the status bit for word logic operations (Page 182)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

XOD: EXCLUSIVE OR logic operation double word by double word

Description

Use the instruction "EXCLUSIVE OR logic operation double word by double word" to XOR the contents of accumulator 1 with the contents of accumulator 2 bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" if the signal state of the two bits to be linked is different. If both of the bits of the logic operation have the same signal state, the corresponding result bit is reset to "0".

The following table shows how the result is formed by an EXCLUSIVE OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	0

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation double word by double word" instruction:

Parameters	Data type	Description
<constant>	DWORD	Value that is XORed with the value in accumulator 1

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
XOD DW#16#39C657AC           // Link the value in accumulator 1 with the constant
                             (DW#16#39C657AC) according to XOR.
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in accumu-
                             lator 1.
XOD                          // XOR the value of accumulator 2 Tag_Value_2 with the
                             value of accumulator 1 ("Tag_Value_3")
                             // Store result in accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39 C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Re-sult_1	0110	0110	1010	0010	1100	1010	1001	0111

Operand	Value							
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Re-sult_2	0101	1100	1000	0101	0000	1010	1000	0111

See also

Overview of the valid data types (Page 231)

Setting the status bit for word logic operations (Page 182)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Shift and rotate

Shift

SSI: Shift word by word with sign

Description

Use the instruction "Shift word by word with sign" to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the right. During the shifting, positions that become free are padded with the signal state of bit 15 (sign bit for INT numbers). The bits 16 to 31 of accumulator 1 remain unchanged. The instruction is executed regardless of the RLO. The status bit CC 1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SSI <number>: The number of shifts is specified by the operand <number>. Values between 0 and 15 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SSI: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with the signal state of bit 15.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC 1 is set to "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SSI 6                        // Shift bits 0 to 15 of accumulator 1 six places to
                             the right.
                             // Fill the now empty bit places with the state of bit
                             15.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 3                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SSI                          // Shift bits 0 to 15 of accumulator 1 three places to
                             the right.
                             // Fill the now empty bit places with the state of bit
                             15.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Tag_Re-sult_1	0101	1111	0110	0100	1111	1110	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Re-sult_2	0101	1111	0110	0100	0000	1011	1010	0101

See also

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SSD: Shift double word by double word with sign

Description

Use the instruction "Shift double word by double word with sign" to shift the complete contents of accumulator 1 bit by bit to the right. During the shifting, positions that become free are padded with the signal state of bit 31 (sign bit for DINT numbers). The instruction is executed regardless of the RLO. The status bit CC 1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SSD <number>: The number of shifts is specified by the operand <number>. Values between 0 and 31 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SSD: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  For values greater than 31, all bits of accumulator 1 are padded with the signal state of bit 31.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC 1 is set to "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
```

STL	Explanation
SSD 7	// Shift bits 0 to 31 of accumulator 1 seven places to the right.
	// Fill the now empty bit places with the state of bit 31.
T "Tag_Result_1"	// Transfer contents of accumulator 1 to the operand.
L 4	// Load number of places shifted into accumulator 1
L "Tag_Value_2"	// Move number of places shifted to the right byte of accumulator 2.
	// Load value of the operand in accumulator 1.
SSD	// Shift bits 0 to 31 of accumulator 1 four places to the right.
	// Fill the now empty bit places with the state of bit 31.
T "Tag_Result_2"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	1000	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1111	1111	0001	1110	1100	1000	1011	1010
Tag_Value_2	0010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	0010	1000	1010	0010	1001	1011	1100

See also

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SLW: Shift left word by word

Description

Use the "Shift left word by word" instruction to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the left. The bit places that become free are padded with zeros. The bits 16 to 31 of accumulator 1 remain unchanged. The instruction is executed regardless of the RLO. The status bit CC 1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SLW <Number>: The number of shifts is specified by the operand <number>. Values between 0 and 15 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SLW: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.

If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC 1 is set to "0".

Parameters

The following table shows the parameters of the "Shift left word by word" instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand in accumulator 1.
SLW 5	// Shift bits 0 to 15 of accumulator 1 five places to the left.
	// Fill the now empty bit places with zeros.
T "Tag_Result_1"	// Transfer contents of accumulator 1 to the operand.
L 4	// Load number of places shifted into accumulator 1
L "Tag_Value_2"	// Move number of places shifted to the right byte of accumulator 2.
	// Load value of the operand in accumulator 1.
SLW	// Shift bits 0 to 15 of accumulator 1 four places to the left.
	// Fill the now empty bit places with zeros.
T "Tag_Result_2"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1010	0111	0110	0000
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	1101	0010	1011	0000

See also

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SRW: Shift right word by word

Description

Use the "Shift right word by word" instruction to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the right. The bit places that become free are padded with zeros. The bits 16 to 31 of accumulator 1 remain unchanged. The instruction is executed regardless of the RLO. The status bit CC 1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SRW <Number>: The number of shifts is specified by the operand <number>. Values between 0 and 15 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SRW: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC 1 is set to "0".

Parameters

The following table shows the parameters of the "Shift right word by word" instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SRW 6                        // Shift bits 0 to 15 of accumulator 1 six places to
                             the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SRW                          // Shift bits 0 to 15 of accumulator 1 four places to
                             the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Re-sult_1	0101	1111	0110	0100	0000	0001	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Re-sult_2	0101	1111	0110	0100	0000	0101	1101	0010

See also

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SLD: Shift left double word by double word

Description

Use the "Shift left double word by double word" instruction to shift the complete contents of accumulator 1 bit by bit to the left. The bit places that become free are padded with zeros. The instruction is executed regardless of the RLO. The status bit CC 1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SLD <number>: The number of shifts is specified by the operand <number>. Values between 0 and 31 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SLD: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  For values greater than 31, all bits of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Parameters

The following table shows the parameters of the "Shift left double word by double word" instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SLD 5                        // Shift bits 0 to 31 of accumulator 1 five places to
                             the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
```

STL	Explanation
L "Tag_Value_2"	// Move number of places shifted to the right byte of accumulator 2.
	// Load value of the operand in accumulator 1.
SLD	// Shift bits 0 to 31 of accumulator 1 four places to the left.
	// Fill the now empty bit places with zeros.
T "Tag_Result_2"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1110	1100	1000	1011	1010	0111	0110	0000
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	1000	1010	0010	1001	1011	1100	1101	0000

See also

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

SRD: Shift right double word by double word

Description

Use the instruction "Shift right double word by double word" to shift the complete contents of accumulator 1 bit by bit to the right. The bit places that become free are padded with zeros. The instruction is executed regardless of the RLO. The status bit CC 1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- SRD <number>: The number of shifts is specified by the operand <number>. Values between 0 and 31 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- SRD: The number of shifts is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  For values greater than 31, all bits of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC 1 is set to "0".

Parameters

The following table shows the parameters of the "Shift right double word by double word" instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SRD 7                        // Shift bits 0 to 31 of accumulator 1 seven places to
                             the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SRD                          // Shift bits 0 to 31 of accumulator 1 four places to
                             the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Re-sult_1	0000	0000	1011	1110	1100	1000	1011	1010

Operand	Value							
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	1010	1000	1010	0010	1001	1011	1100

See also

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Rotate

RLD: Rotate left double word by double word

Description

Use the "Rotate left double word by double word" instruction to rotate the complete contents of accumulator 1 bit by bit to the left. During processing the instruction shifts bit 0 to 31 of accumulator 1 bit by bit to the left and pads the places that become free during writing with the pushed out bit places. The instruction is executed regardless of the RLO. The status bit CC 1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be rotated:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- RLD <number>: The number of rotations is specified by the operand <number>. Values between 0 and 32 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- RLD: The number of rotations is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  For values greater than 32, the rotation number is calculated with a modulo division by 32. If, for example, the rotate number is 34, the bits of accumulator 1 are rotated by 2 bit places.

If the specified rotate number is 32, the contents of accumulator 1 remain unchanged.

If the specified rotate number is equal to zero, the instruction is still executed. The status bit CC 1 is set to "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places to be rotated.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
RLD 4                        // Rotate bits 0 to 31 of accumulator 1 four places to
                             the left
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 6                          // Load number of places to be rotated in accumulator
                             1.
L "Tag_Value_2"              // Move number of places to be rotated to right byte
                             of accumulator 2
                             // Load value of the operand in accumulator 1.
RLD                          // Rotate bits 0 to 31 of accumulator 1 six places to
                             the left
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	**0101**	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1111	0110	0100	0101	1101	0011	1011	**0101**
Tag_Value_2	**1010**	**10**00	1010	0010	1001	1011	1100	1101
Tag_Result_2	0010	1000	1010	0110	1111	0011	01**10**	**1010**

See also

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

RRD: Rotate right double word by double word

Description

Use the "Rotate right double word by double word" instruction to rotate the complete contents of accumulator 1 bit by bit to the right. During processing the instruction shifts bits 0 to 31 of accumulator 1 bit by bit to the right and pads the places that become free during writing with the pushed out bit places. The instruction is executed regardless of the RLO. The status bit CC 1 is set to the signal state of the last bit that was shifted out.

The following options are available for specifying the number of bit places by which the bits are to be rotated:

- Specification of a positive integer as parameter of the instruction. (<number>).
- Specification via the value of the right byte in accumulator 2: The byte is interpreted as a positive integer.

You can use the following formats:

- RRD <number>: The number of rotations is specified by the operand <number>. Values between 0 and 32 are permitted. The status bits CC 0 and OV are reset to "0" when the <number> is greater than zero.
- RRD: The number of rotations is specified by the value in accumulator 2. Values between 0 and 255 are permitted. The status bits CC 0 and OV are reset to "0" when the content of accumulator 2 is greater than zero.
  For values greater than 32, the rotation number is calculated with a modulo division by 32. If, for example, the rotate number is 34, the bits of accumulator 1 are rotated by 2 bit places.

If the specified rotate number is 32, the contents of accumulator 1 remain unchanged.

If the specified rotate number is equal to zero, the instruction is still executed. The status bit CC 1 is set to "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Format	Description
<Number>	Positive integers: SINT, INT, UINT, USINT	Number of bit places to be rotated.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
RRD 4                        // Rotate bits 0 to 31 of accumulator 1 four places to
                             the right.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 6                          // Load number of places to be rotated in accumulator
                             1.
```

STL	Explanation
L "Tag_Value_2"	// Move number of places to be rotated to right byte of accumulator 2
	// Load value of the operand in accumulator 1.
RRD	// Rotate bits 0 to 31 of accumulator 1 six places to the right.
T "Tag_Result_2"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	**1011**
Tag_Re-sult_1	**1011**	0101	1111	0110	0100	0101	1101	0011
Tag_Value_2	1010	1000	1010	0010	1001	1011	11**00**	**1101**
Tag_Re-sult_2	**0011**	**01**10	1010	0010	1000	1010	0110	1111

See also

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Overview of the valid data types (Page 231)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

RLDA: Rotate left by status bit CC 1

Description

You can use the "Rotate left by status bit CC 1" instruction to shift the contents of accumulator 1 one bit to the left. The bit places (0) that become free during shifting are padded with the signal state of status bit CC 1. Status bit CC 1 receives the signal state of the shifted out bit (31).

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0".

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load value of the operand in accumulator 1.

```
STL                          Explanation
RLDA                         // Rotate bits 0 to 31 of accumulator 1 one place to
                             the left
                             // Fill bit 1 with the signal state of status bit CC
                             1.
                             // Write the shifted-out bit to the status bit CC 1.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	CC 1	Value							
Tag_Value	**1**	**0**101	1111	0110	0100	0101	1101	0011	1011
Tag_Result	**0**	1011	1110	1100	1000	1011	1010	0111	011**1**

See also

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

RRDA: Rotate right by status bit CC 1

Description

Use the "Rotate right by status bit CC 1" instruction to shift the contents of accumulator 1 one bit to the right. The bit places (31) that become free during shifting are padded with the signal state of status bit CC 1. Status bit CC 1 receives the signal state of the shifted out bit (0).

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0".

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load value of the operand in accumulator 1.
RRDA                         // Rotate bits 0 to 31 of accumulator 1 one place to
                             the right
                             // Fill bit 31 with the signal state of status bit CC
                             1.
                             // Write the shifted-out bit to the status bit CC 1.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	CC 1	Value							
Tag_Value	1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result	0	1010	1111	1011	0010	0010	1110	1001	1101

Status bit CC 1 has the value "1" because the bit 0 of the "Tag_Value" operand also has the value "1".

See also

Setting the status bits for shift and rotate instructions (Page 182)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Additional instructions

Accumulators

TAK: Swap content of accumulators 1 and 2

Description

Use the instruction "Swap content of accumulators 1 and 2" to switch the contents accumulator 1 with the contents of accumulator 2. In case of controllers with four accumulators, the contents of accumulators 3 and 4 are not influenced by the instruction.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load value of the operand "Tag_Value_1" in accumulator 1.`
`L "Tag_Value_2"`	`// Shift the contents of accumulator 1 to accumulator 2.`
	`// Load value of the operand "Tag_Value_2" in accumulator 1`
`>I`	`// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.`

STL	Explanation
JC NEXT	// If condition is satisfied (RLO = "1"), jump to jump label NEXT.
	// If the condition is not satisfied (RLO = "0"), execute the next instruction.
TAK	// Swap the contents of accumulators 1 and 2.
NEXT: -I	// Jump label "NEXT"
	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Output"	// Transfer contents of accumulator 1 to the operand "Tag_Output".

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

PUSH: Shift content to the next higher accumulator

Description

Use the "Shift content to the next higher accumulator" instruction to move the contents of the accumulators 1 to 3 to the next higher accumulator.

In case of controllers with 2 accumulators, only the contents of accumulator 1 are copied to accumulator 2.

In controls with 4 accumulators, the content of accumulator 1 is moved to accumulator 2; the content of accumulator 2 is moved to accumulator 3 and the content of accumulator 3 to accumulator 4. The content of accumulator 4 will be lost.

The content of accumulator 1 is not influenced by the instruction and remains unchanged after processing the instruction.

The following table shows the contents of accumulators 1 to 4 before and after the processing of the instruction:

Status	Accumulator			
	1	2	3	4
Before processing	Value A	Value B	Value C	Value D
After processing	Value A	Value A	Value B	Value C

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load value of the operand in accumulator 1.`
`PUSH`	`// Shift contents of accumulators 1 to 3 to the next higher accumulator.`
`L "Tag_Value_2"`	`// Load value of the operand in accumulator 1.`

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

POP: Shift content to the next lower accumulator

Description

Use the "Shift content to the next lower accumulator" instruction to move the contents of the accumulators 4 to 2 to the next lower accumulator.

In controls with 2 accumulators the content of accumulator 2 will be moved to accumulator 1.

In controls with 4 accumulators, the content of accumulator 4 is moved to accumulator 3; the content of accumulator 3 is moved to accumulator 2 and the content of accumulator 2 to accumulator 1. The content of accumulator 1 will be lost.

The content of accumulator 4 is not influenced by the instruction and remains unchanged after processing the instruction.

The following table shows the contents of accumulators 1 to 4 before and after the processing of the instruction:

Status	Accumulator			
	1	2	3	4
Before processing	Value A	Value B	Value C	Value D
After processing	Value B	Value C	Value D	Value D

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
`T "Tag_Value_1"`	`// Transfer contents of accumulator 1 to the operand "Tag_Value_1".`
`POP`	`// Shift content of accumulators 4 to 2 to the next lower accumulator`

STL	Explanation
T "Tag_Value_2"	// Transfer contents of accumulator 1 to the operand "Tag_Value_2".

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

ENT: Shift accumulators 2 and 3 to the next higher accumulator

Description

With the "Shift accumulators 2 and 3 to the next higher accumulator" you shift the contents of accumulator 3 to accumulator 4 and the contents of accumulator 2 to accumulator 3. The content of accumulators 1 and 2 remains unchanged.

The following table shows the contents of accumulators 1 to 4 before and after the processing of the instruction:

Status	Accumulator			
	1	2	3	4
Before processing	Value A	Value B	Value C	Value D
After processing	Value A	Value B	Value B	Value C

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product (intermediate result) in accumulator 1.
L "Tag_Value_3"	// Shift the content of accumulator 1 (intermediate result) to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1

STL	Explanation
ENT	// Copy the content of accumulator 2 (intermediate result) to accumulator 3.
	// Copy the content of accumulator 3 to accumulator 4.
L "Tag_Value_4"	// Move the contents of accumulator 1 ("Tag_Value_3") to accumulator 2.
	// Load value of the operand "Tag_Value_4" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
	// Transfer the value of accumulator 3 (intermediate result) to accumulator 2.
LEAVE	// Shift the content of accumulator 3 (intermediate result) to accumulator 2.
	// Shift the contents of accumulator 4 to accumulator 3.
/R	// Divide the values of accumulator 1 and 2.
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

LEAVE: Shift accumulators 3 and 4 to the next lower accumulator

Description

With the "Shift accumulators 3 and 4 to the next lower accumulator" you shift the content of accumulator 3 to accumulator 2 and the content of accumulator 4 to accumulator 3. The content of accumulators 1 and 4 remains unchanged.

Instructions for arithmetic calculations include the functionality of the "Shift accumulators 3 and 4 to the next lower accumulator" instruction.

The following table shows the contents of accumulators 1 to 4 before and after the processing of the instruction:

Status	Accumulator			
	1	2	3	4
Before processing	Value A	Value B	Value C	Value D
After processing	Value A	Value C	Value D	Value D

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
AD	// AND the value of accumulator 2 Tag_Value_1 with the value of accumulator 1 ("Tag_Value_2")
	// Store result in accumulator 1.
L "Tag_Value_3"	// Shift the contents of accumulator 1 (intermediate result) to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
ENT	// Copy the content of accumulator 2 (intermediate result) to accumulator 3.
	// Copy the content of accumulator 3 to accumulator 4.
L "Tag_Value_4"	// Move the contents of accumulator 1 ("Tag_Value_3") to accumulator 2.
	// Load value of the operand "Tag_Value_4" in accumulator 1
AD	// AND the value of accumulator 2 Tag_Value_3 with the value of accumulator 1 ("Tag_Value_4")
	// Store result in accumulator 1.
LEAVE	// Shift the contents of accumulator 3 (intermediate result) to accumulator 2.
	// Shift the contents of accumulator 4 to accumulator 3.
OD	// OR the value of accumulator 2 with the value of accumulator 1
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer the result to operand "Tag_Result".

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Address register

+AR1: Add accumulator 1 to AR1

Description

With the "Add accumulator 1 to AR1" instruction you add a value to the content of address register 1. The type of pointer located in address register 1 and the operand range remain as they are.

The following options are available for specifying the value to be added:

- Specification via a constant: The instruction adds the value of the constant to address register 1. The value of the constant must correspond to the format of a range-internal pointer (POINTER).
- Specification via the value of the right word in the accumulator 1: The instruction interprets the value of the right word in accumulator 1 as a 16-bit integer and extends it with the correct sign to 24 bits. The instruction then adds value of accumulator 1 to address register 1. Values from -32768 to +32767 are permitted.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Parameters

The following table shows the parameters of the "Add accumulator 1 to AR1" instruction:

Parameters	Data type	Description
<constant>	POINTER	Value that is added to address register 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
+AR1 P#10.0                  // Add pointer to address register 1.
L "Tag_Value"                // Load value of the operand "Tag_Value" into the
                             right word of accumulator 1.
+AR1                         // Add value of accumulator 1 to address register 1.
TAR1 %MD24                   // Transfer contents of address register 1 to double
                             word MD24.
```

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

+AR2: Add accumulator 1 to AR2

Description

With the "Add accumulator 1 to AR2" instruction you add a value to the content of address register 2. The type of pointer located in address register 2 and the operand range remain as they are.

The following options are available for specifying the value to be added:

- Specification via a constant: The instruction adds the value of the constant to address register 2. The value of the constant must correspond to the format of a range-internal pointer (POINTER).
- Specification via the value of the right word in the accumulator 1: The instruction interprets the value of the right word in accumulator 1 as a 16-bit integer and extends it with the correct sign to 24 bits. The instruction then adds value of accumulator 1 to address register 2. Values from -32768 to +32767 are permitted.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Note

Address register AR2 is used when processing multi-instances. If you program the "Add accumulator 1 to AR2" instruction, you must first "save" the contents of address register 2 and reload it later.

Parameters

The following table shows the parameters of the "Add accumulator 1 to AR2" instruction:

Parameters	Data type	Description
<constant>	POINTER	Value that is added to address register 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
+AR2 P#10.0                  // Add pointer to address register 2.
L "Tag_Value"                // Load value of the operand "Tag_Value" into the
                             right word of accumulator 1.
+AR2                         // Add value of accumulator 1 to address register 2.
TAR2 %MD24                   // Transfer contents of address register 1 to double
                             word MD24.
```

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Master Control Relay

MCR(: Open MCR ranges

Description

Use the "Open MCR ranges" instruction to switch the MCR dependency of an MCR range on and off depending on the result of logic operation (RLO).

Use the instructions "Enable MCR range" (MCRA) and "Disable MCR range" (MCRD) to specify the start and end of an MCR range. Within the defined MCR range you can switch the MCR dependency on and off for one or more program sections.

Note

Read the information in the section "Important notes on using the MCR functionality" in "See also".

The MCR dependency is switched on when the RLO is "1" before processing of the instruction. If the RLO is "0" before the instruction, the MCR dependency is switched off. The end of the MCR dependent program section is programmed using the instruction "Close MCR range". The program section between the two instructions is affected by the MCR. Only the MCR-dependent instructions (for example, the "Transfer" instruction (T)) are affected.

You can also nest the switching on and off of MCR dependency by programming the "Open MCR ranges" instruction in an MCR-dependent program section. You can nest the instruction up to 8 levels deep. The MCR dependency of the higher-level program section hereby controls the MCR dependencies in all lower-level program sections. If the MCR dependency is switched off in the higher-level program section, this MCR dependency cannot be switched on in any of the lower-level MCR dependent program sections.

If you call a block within an MCR range, the MCR dependency is disabled in the called block. When the block is exited, the MCR dependency is set so that it is the same as before the block call.

The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Example

The following example shows how the instruction works:

STL	Explanation
MCRA	// Start of the MCR range
A "Tag_Input_1"	// Query operand for "1" and link with the current RLO according to AND.

STL	Explanation
MCR(	// Start of the MCR dependency
	// If signal state of RLO is "1", switch on MCR dependency.
	// If signal state of RLO is "0", switch off MCR dependency.
A "Tag_Input_2"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_1"	// If MCR dependency is switched off, reset operand to "0".
	// If MCR dependency is switched on, set or reset operand depending on RLO
A "Tag_Input_3"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Nesting of MCR dependency
A "Tag_Input_4"	// Query operand for "1" and link with the current RLO according to AND.
A "Tag_Input_5"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_2"	// If MCR dependency is switched off, reset operand to "0".
	// If MCR dependency is switched on, set or reset operand depending on RLO
)MCR	// End of the nesting
L "Tag_InputValue_1"	// Load value of the operand in accumulator 1.
T "Tag_OutputValue_1"	// If MCR dependency is switched off, write the value "0" to the operand.
	// If MCR dependency is switched on, transfer the contents of accumulator 1 to the operand.
)MCR	// End of the MCR dependency
A "Tag_Input_6"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Start of the MCR dependency
	// If signal state of RLO is "1", switch on MCR dependency.
	// If signal state of RLO is "0", switch off MCR dependency.
L "Tag_InputValue_2"	// Load value of the operand in accumulator 1.
T "Tag_OutputValue_2"	// If MCR dependency is switched off, write the value "0" to the operand.
	// If MCR dependency is switched on, transfer the contents of accumulator 1 to the operand.
)MCR	// End of the MCR dependency
MCRD	// End of the MCR range
A "Tag_Input_7"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_3"	// Set or reset operand depending on RLO

See also

Important notes on using MCR functions (Page 192)

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

)MCR: Close MCR ranges

Description

The "Close MCR ranges" instruction completes an MCR dependent program section. In this case the instruction deletes the RLO from the last entry in the MCR stack.

Note

Read the information in the section "Important notes on using the MCR functionality" in "See also".

Use the instruction "Open MCR ranges" to program the MCR dependency. The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Example

The following example shows how the instruction works:

STL	Explanation
MCRA	// Start of the MCR range
A "Tag_Input_1"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Start of the MCR dependency
	// If signal state of RLO is "1", switch on MCR dependency.
	// If signal state of RLO is "0", switch off MCR dependency.
A "Tag_Input_2"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_1"	// If MCR dependency is switched off, reset operand to "0".
	// If MCR dependency is switched on, set or reset operand depending on RLO
L "Tag_InputValue"	// Load value of the operand in accumulator 1.

STL	Explanation
T "Tag_OutputValue"	// If MCR dependency is switched off, write the value "0" to the operand.
	// If MCR dependency is switched on, transfer the contents of accumulator 1 to the operand.
)MCR	// End of the MCR dependency
MCRD	// End of the MCR range
A "Tag_Input_3"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_2"	// Set or reset operand depending on RLO

See also

Important notes on using MCR functions (Page 192)

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

MCRA: Enable MCR range

Description

Use the instruction "Enable MCR range" to specify the start of an MCR range. Within this range you can switch MCR dependency on and off using the following instructions:

- MCR(: Open MCR ranges
-)MCR: Close MCR ranges

You specify the end of the MCR range with the instruction "Disable MCR range" (MCRD). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

Read the information in the section "Important notes on using the MCR functionality" in "See also".

If you call a block within an MCR range, the MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Example

The following example shows how the instruction works:

STL	Explanation
MCRA	// Start of the MCR range
A "Tag_Input_1"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Start of the MCR dependency
	// If signal state of RLO is "1", switch on MCR dependency.
	// If signal state of RLO is "0", switch off MCR dependency.
A "Tag_Input_2"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_1"	// If MCR dependency is switched off, reset operand to "0".
	// If MCR dependency is switched on, set or reset operand depending on RLO
L "Tag_InputValue"	// Load value of the operand in accumulator 1.
T "Tag_OutputValue"	// If MCR dependency is switched off, write the value "0" to the operand.
	// If MCR dependency is switched on, transfer the contents of accumulator 1 to the operand.
)MCR	// End of the MCR dependency
MCRD	// End of the MCR range
A "Tag_Input_3"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_2"	// Set or reset operand depending on RLO

See also

Important notes on using MCR functions (Page 192)

Setting status bits in program control operation instructions (Page 180)

Basic information on the status word (Page 173)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

MCRD: Disable MCR range

Description

Use the instruction "Disable MCR range" to specify the end of an MCR range.

You specify the start of the MCR range with the instruction "Enable MCR range" (MCRA). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

Read the information in the section "Important notes on using the MCR functionality" in "See also".

Within an MCR range you can program the MCR dependency with the help of the following instructions:

- MCR(: Open MCR ranges
-)MCR: Close MCR ranges

If you call a block within an MCR range, the MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Example

The following example shows how the instruction works:

STL	Explanation
MCRA	// Start of the MCR range
A "Tag_Input_1"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Start of the MCR dependency
	// If signal state of RLO is "1", switch on MCR dependency.
	// If signal state of RLO is "0", switch off MCR dependency.
A "Tag_Input_2"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_1"	// If MCR dependency is switched off, reset operand to "0".
	// If MCR dependency is switched on, set or reset operand depending on RLO
L "Tag_InputValue"	// Load value of the operand in accumulator 1.
T "Tag_OutputValue"	// If MCR dependency is switched off, write the value "0" to the operand.
	// If MCR dependency is switched on, transfer the contents of accumulator 1 to the operand.
)MCR	// End of the MCR dependency
MCRD	// End of the MCR range
A "Tag_Input_3"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_2"	// Set or reset operand depending on RLO.

See also

Important notes on using MCR functions (Page 192)
Setting status bits in program control operation instructions (Page 180)
Basic information on the status word (Page 173)
Inserting STL instructions (Page 7259)
Editing STL instructions (Page 7277)

Null instructions

BLD: Program display (null instruction)

Description

The instruction "Program display (null instruction)" does not execute any function and does not affect the status bits. The instruction is used to recognize code sequences during a parameter transfer or with LAD/FBD networks. It is created automatically when a LAD or FBD program is displayed in STL. The parameter value is the number of the instruction and is generated by the programming device.

Parameters

The following table shows the parameters of the "Program display (null instruction)" instruction:

Parameters	Data type	Description
<number>	WORD	Number of the instruction

See also

Inserting STL instructions (Page 7259)
Editing STL instructions (Page 7277)

NOP 0: Null instruction

Description

If "Null instruction" has a parameter 0, it does not execute any function and does not affect the status bits. The instruction code contains a bit pattern with 16 zeros. The instruction is only relevant for the programming device when a program is displayed.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A	// Query the signal state of the operand for "1" and AND with the current RLO.
JC NEXT	// If condition is satisfied (RLO = "1"), jump to jump label NEXT.
	// If the condition is not satisfied (RLO = "0"), execute the next instruction.
TAK	// Swap the contents of accumulators 1 and 2.
NEXT: NOP 0	// Jump label "NEXT"

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

NOP 1: Null instruction

Description

If "Null instruction" has a parameter 1, it does not execute any function and does not affect the status bits. The instruction code contains a bit pattern with 16 ones. The instruction is only relevant for the programming device when a program is displayed.

See also

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

4.2.2.4 SCL

Timer operations

IEC Timers

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the Q parameter for the duration PT. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The time programmed for PT begins when the instruction starts. The Q parameter is set for the period of time PT, regardless of the subsequent changes in the signal of the IN parameter. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the time duration PT is active.

You can query the current time value set in the ET parameter. The time value starts at T#0s and ends when the value of the time PT is reached. When the time duration PT is reached and the signal state at the IN parameter is "0", the ET parameter is reset.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored. You can declare an IEC timer as follows:

- Declaration of a data block of the type TP (for example, "TP_DB")
- Declaration as a local tag of the type TP in the "Static" section of a block (for example, #MyTP_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate pulse" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

IEC timer as instance data block of system data type IEC_<Timer> (Shared DB)

You can declare the IEC timer as a data block as follows:

<IEC_Timer_DB>.TP();

IEC timer as local tag from the block interface (multi-instance)

You can declare the IEC timer as a local tag as follows:

#myLocal_Timer();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Operand that is set for the PT duration.
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:

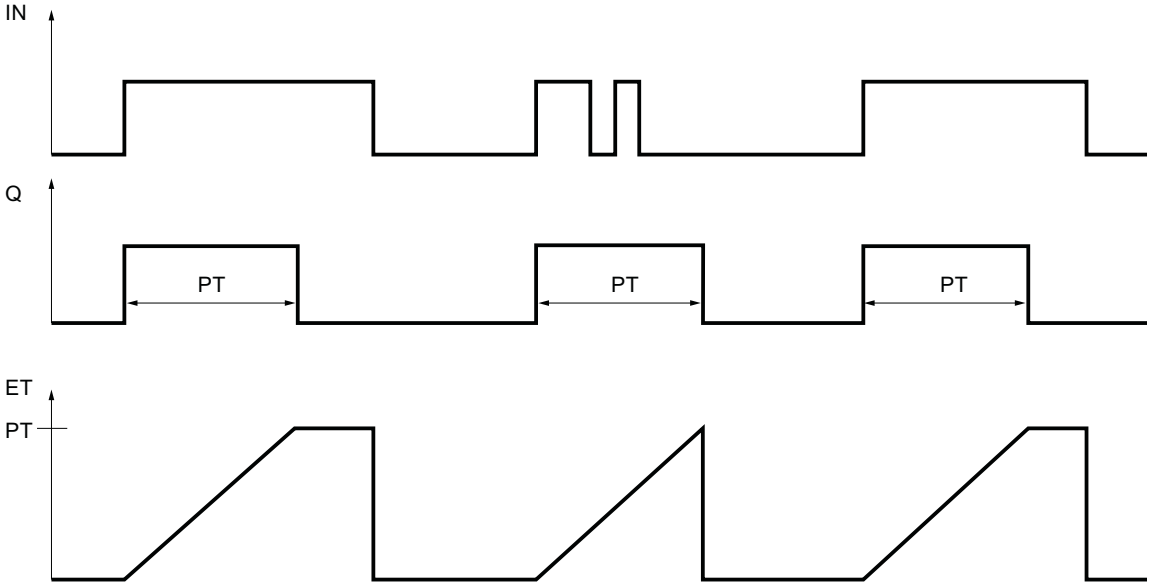

Example

The following example shows how the instruction works:

SCL

```
"TP_DB".TP(IN := "Tag_Start",
           PT := "Tag_PresetTime",
           Q => "Tag_Status",
           ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basics of SCL (Page 7293)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay the setting of the Q parameter for the programmed duration PT. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the PT duration has expired, the Q parameter returns signal state "1". The Q parameter remains set as long as the start input is still "1". If the signal state of the IN parameter changes from "1" to "0", the parameter Q will be reset. The timer function is started again when a new rising edge is detected at the parameter IN.

You can query the current time value set in the ET parameter. The time value starts at T#0s and ends when the value of the time PT is reached. The ET parameter is reset as soon as the signal state of the IN parameter changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. You can declare an IEC timer as follows:

- Declaration of a data block of the type TON (for example, "TON_DB")
- Declaration as a local tag of the type TON in the "Static" section of a block (for example, #MyTON_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

IEC timer as instance data block of system data type IEC_<Timer> (Shared DB)

You can declare the IEC timer as a data block as follows:

<IEC_Timer_DB>.TON();

IEC timer as local tag from the block interface (multi-instance)

You can declare the IEC timer as a local tag as follows:

#myLocal_Timer();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Operand that is set when the timer PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:

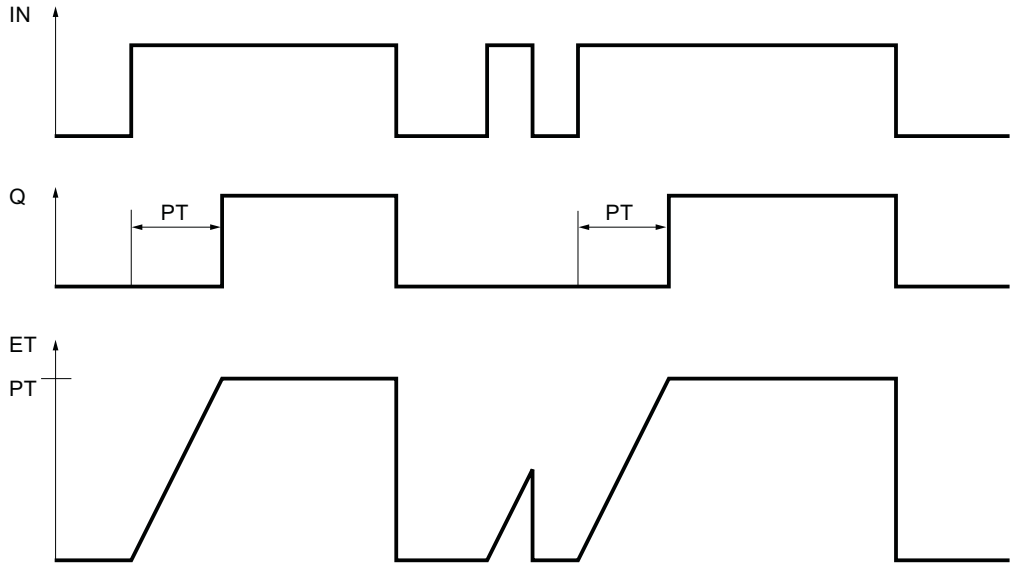

Example

The following example shows how the instruction works:

SCL

```
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. At the end of the time, the "Tag_Start" operand is set to signal state "1" if the "Tag_Status" operand has signal state "1". At the end of the time the "Tag_Status" operand is set to signal state "1" if the "Tag_Start" operand has signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basics of SCL (Page 7293)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay the resetting of the Q parameter for the programmed duration PT. The Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state at the IN parameter changes back to "0", the programmed time PT starts. The Q parameter remains set as long as time PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before time PT expires, the timer is reset. The signal state of the Q parameter remains set to "1".

You can query the current time value set in the ET parameter. The current timer value starts at T#0s and ends when the value of the time duration PT is reached. When the time PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1". If the IN parameter changes to "1" before the time duration PT has expired, the ET parameter is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC Timer in which the instruction data is stored. You can declare an IEC timer as follows:

- Declaration of a data block of the type TOF (for example, "TOF_DB")
- Declaration as a local tag of the type TOF in the "Static" section of a block (for example, #MyTOF_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

IEC timer as instance data block of system data type IEC_<Timer> (Shared DB)

You can declare the IEC timer as a data block as follows:

<IEC_Timer_DB>.TOF();

IEC timer as local tag from the block interface (multi-instance)

You can declare the IEC timer as a local tag as follows:

#myLocal_Timer();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Operand that is reset when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse diagram of the "Generate off-delay" instruction:

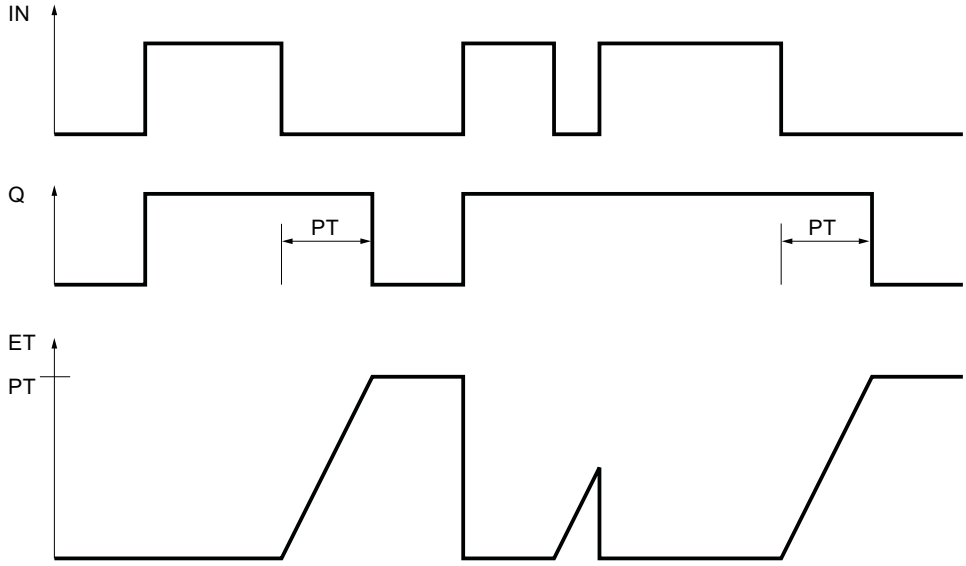

Example

The following example shows how the instruction works:

SCL

```
"TOF_DB".TOF(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

With a change in the signal state of the "Tag_Start" operand from "0" to "1", the "Tag_Status" operand is set. When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set. When the time has expired, the "Tag_Status" operand is reset. The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basics of SCL (Page 7293)

SIMATIC Timers

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts the time programmed in the T_NO parameter when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) of the S parameter. The timer runs for the programmed time (TV) as long as the signal state of the S parameter is "1".

When the signal state of the S parameter changes to "0" before the programmed time has expired, the timer is stopped and the Q parameter is reset to "0".

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

Parameter Q returns signal state "1" as long as the timer is running and the signal state at parameter S is "1". When the signal state of the S parameter changes to "0" before the programmed time has expired, the Q parameter returns signal state "0". If the timer is reset by parameter R or if the timer has expired then parameter Q also returns signal state "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value

Parameters	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign pulse timer parameters and start" instruction:

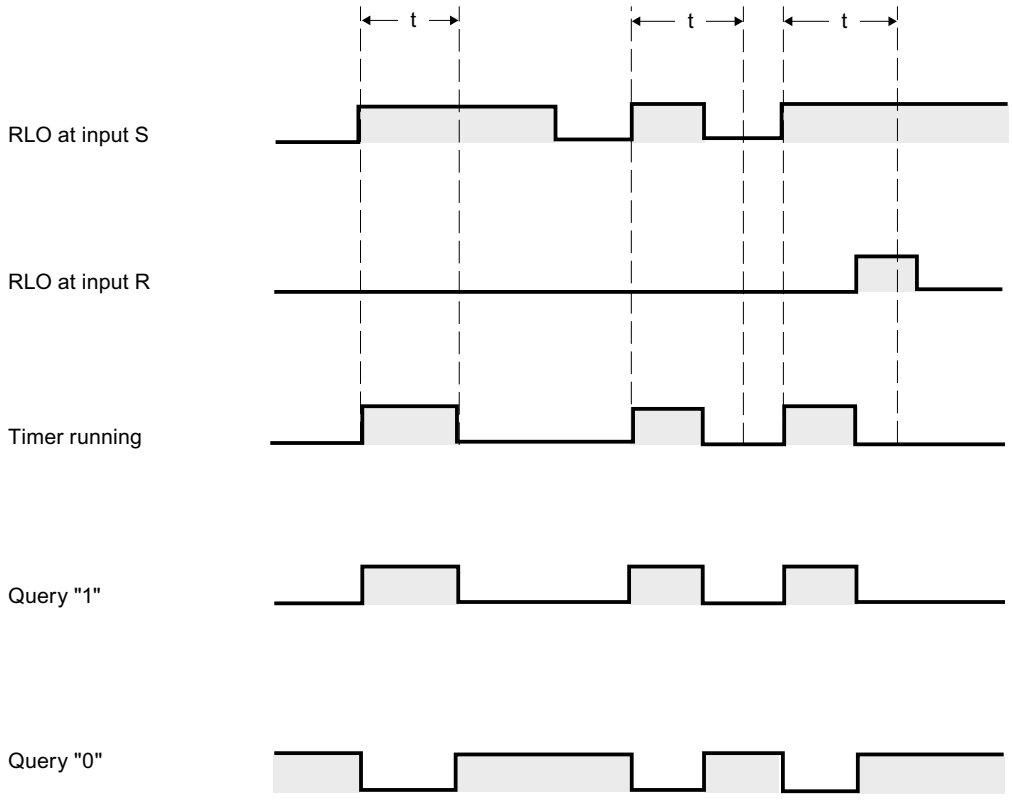

t = programmed time

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := S_PULSE(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q := "Tag_Status",
                        BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer counts down with the time value of the operand "Tag_Number" until operand "Tag_1" returns signal state "1".

If the signal state at the S parameter changes to "0" before the programmed time has elapsed, the "Tag_Status" operand is reset to "0". If the timer is reset by the R parameter or if the timer has expired, the "Tag_Status" operand also returns signal state "0".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basics of SCL (Page 7293)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) even if the signal state at the S parameter changes to "0". As long as the timer runs, parameter Q returns the signal state "1".

When the timer has expired, parameter Q is reset to "0". If the signal state at the S parameter changes from "0" to "1" while the timer is running, the timer is restarted with the time programmed in the TV parameter.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the timer is running and the signal state at parameter R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running then signal state "1" at parameter R has no effect.

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:

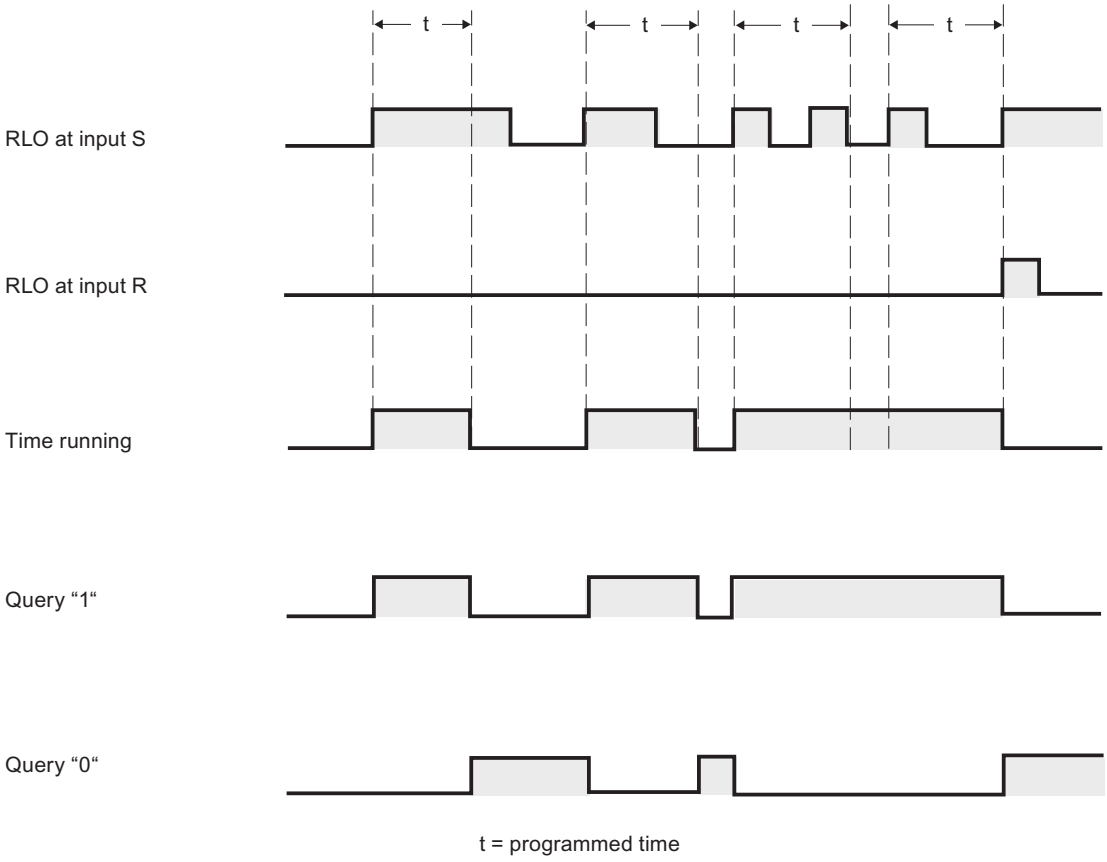

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_PEXT(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q := "Tag_Status",
                       BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". As long as the timer runs, operand "Tag_Status" returns the signal state "1". When the timer has expired, operand "Tag_Status" is reset to "0". If the signal state at the S input changes from "0" to "1" while the timer is running, the timer is restarted with the time "Tag_Number".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basics of SCL (Page 7293)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer as on-delay when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) as long as the signal state of the S parameter is "1".

If the timer expires correctly and parameter S still has signal state "1" then parameter Q returns signal state "1". If the signal state at the S parameter changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the time is running and the signal state at input R changes from "0" to "1" then the current time value and the time base are also set to zero. In this case, the signal state at parameter Q is "0". The timer is reset if the signal state at the R parameter is "1", even if the timer is not running and the result of logic operation (RLO) at the S parameter is "1".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value

Parameters	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:

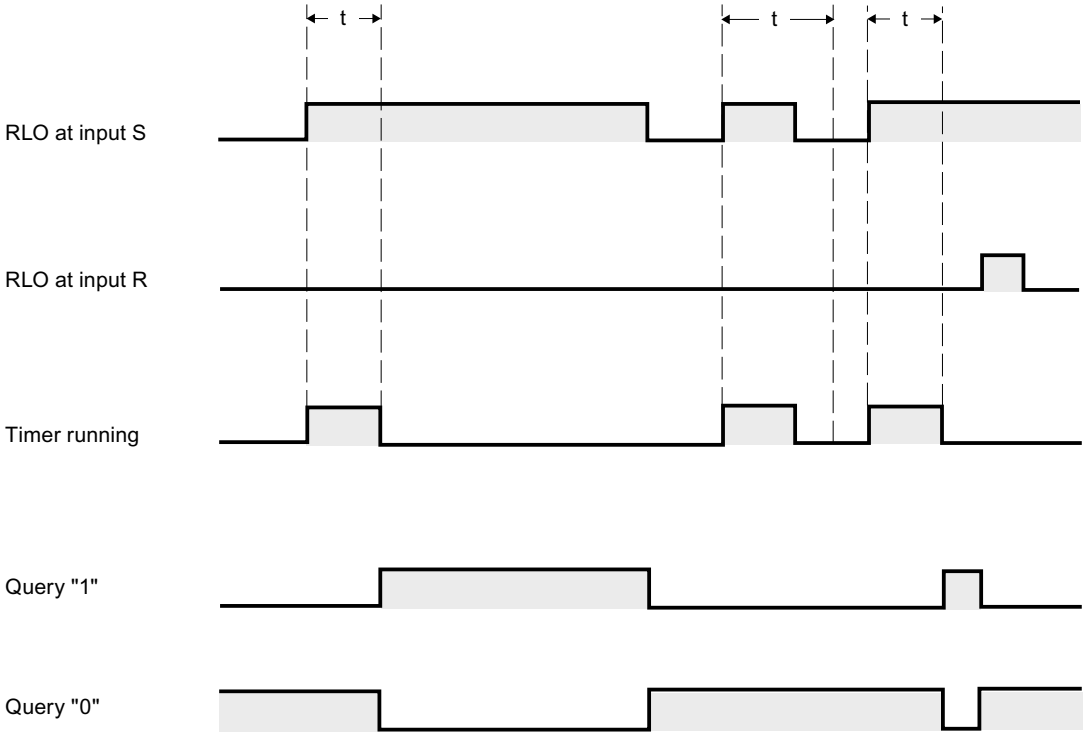

t = programmed time

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_ODT(T_NO := "Timer_1",
                      S := "Tag_1",
                      TV := "Tag_Number",
                      R := "Tag_Reset",
                      Q := "Tag_Status",
                      BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number" as long as the signal state of operand "Tag_1" is "1".

If the timer expires correctly and operand "Tag_Status" has signal state "1" then operand "Tag_Status" is reset to "1". If the signal state of the "Tag_1" operand changes from "1" to "0" while the timer is running, the timer is stopped. In this case, operand "Tag_Status" returns signal state "0".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basics of SCL (Page 7293)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) even if the signal state at the S parameter changes to "0".

When the timer expires, the "Q" parameter returns signal state "1" regardless of the signal state of the "S" parameter. If the signal state at the S parameter changes from "0" to "1" while the timer is running, the timer is restarted with the programmed time TV.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

Signal state "1" at parameter R resets the current time value and time base to "0", independent of the signal state at parameter S. In this case, the signal state at parameter Q is "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:

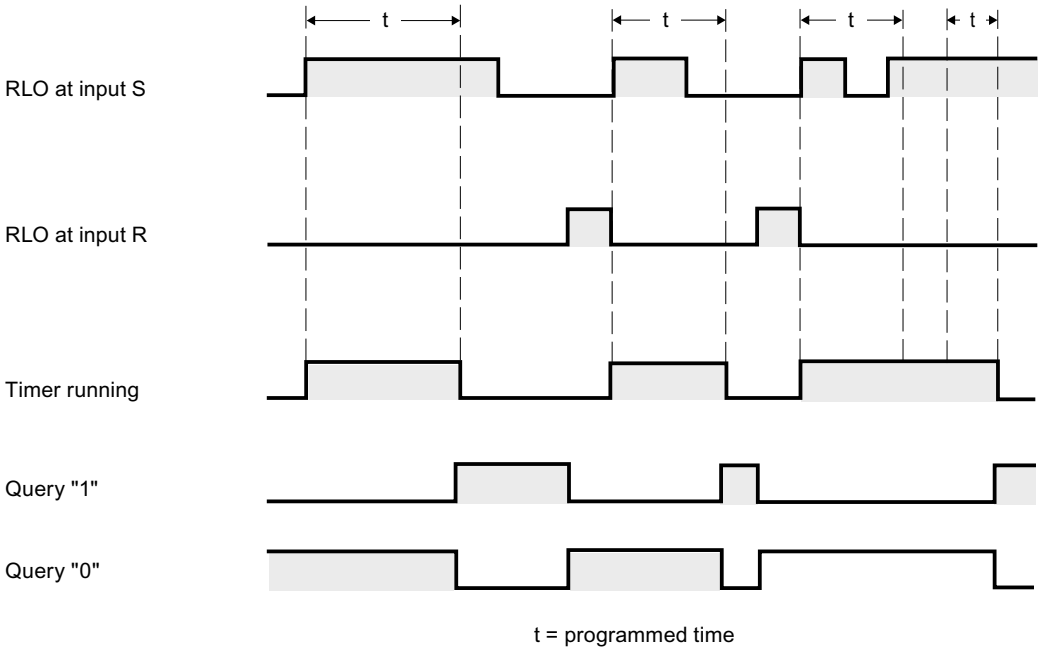

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := S_ODTS(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q := "Tag_Status",
                       BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number".

If the timer expires, operand "Tag_Status" returns signal state "1" independent of the signal state of operand "Tag_1". If the signal state of the "Tag_1" operand changes from "0" to "1" while the timer is running, the timer is restarted with the time "Tag_Number".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basics of SCL (Page 7293)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a negative signal edge is detected at the S parameter. The timer runs for the programmed time (TV). As long as the timer is running or parameter S returns signal state "1", then parameter Q has signal state "1".

If the timer expires and the signal state is "0" then parameter Q is reset to signal state "0". If the signal state at parameter S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at parameter S.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

Signal state "1" at parameter R resets the current time value and time base to "0". In this case, the signal state at parameter Q is "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
T_NO	Input	TIMER, INT	T	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L	Preset time value
R	Input	BOOL	I, Q, M, D, L, P	Reset input
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer
BI	Output	WORD	I, Q, M, D, L, P	Current dual-coded timer value
Function value		S5TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:

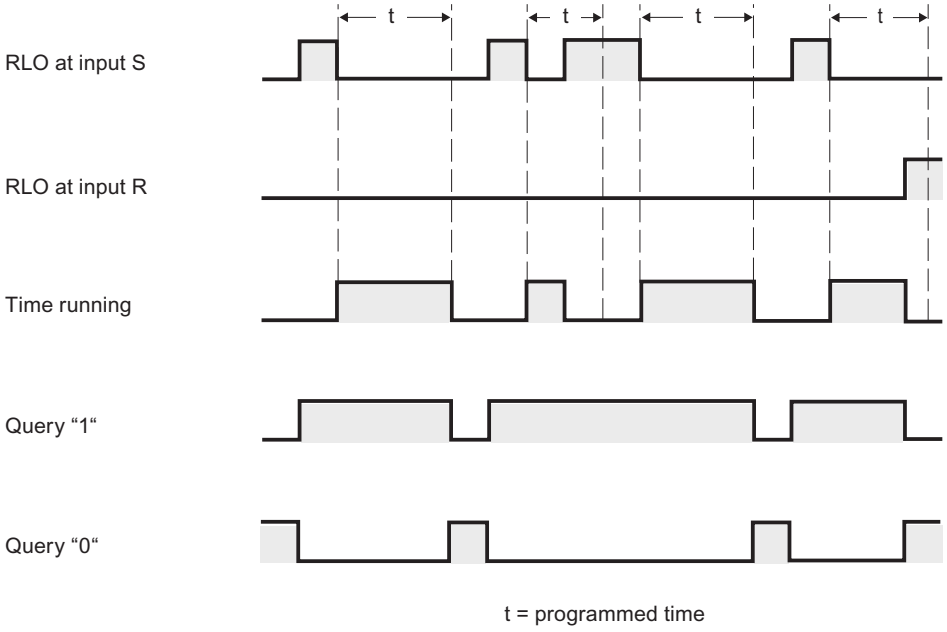

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_OFFDT(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q := "Tag_Status",
                        BI := "Tag_Value");
```

The timer "Timer_1" is started when the signal state of the operand "Tag_1" changes from "1" to "0". The timer runs for the duration "Tag_Number". As long as the timer is running or operand "Tag_1" returns signal state "1", then operand "Tag_Status" has signal state "1".

If the timer expires and the signal state of operand "Tag_1" is "0" then operand "Tag_Status" is reset to signal state "0". If the signal state of the "Tag_1" operand changes from "0" to "1" while the timer is running, the timer is reset. The timer is only restarted after a falling edge is detected at parameter S.

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 231)

L: Load timer value (Page 4734)

Basics of SCL (Page 7293)

Counter operations

IEC Counters

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at the CV parameter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the parameter CV is incremented by one. The counter value is increased each time a positive signal edge is detected, until it reaches the high limit of the data type (INT). When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current counter value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the signal state of the R parameter is "1", the signal state of the CU parameter has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")
- Declaration as a local tag of the type CTU in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "1" set for the R parameter of the instruction. If

instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

IEC counter as data block of system data type IEC_<Counter> (Shared DB)

You can declare the IEC counter as a data block as follows:

<IEC_Counter_DB>.CTU();

IEC counter as local tag from the block interface (multi-instance)

You can declare the IEC counter as a local tag as follows:

#myLocal_Counter();

Syntax

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, P	Reset input
PV	Input	INT	I, Q, M, D, L, P	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"CTU_DB".CTU(CU := "Tag_Start",
             R := "Tag_Reset",
             PV := "Tag_PresetValue",
             Q => "Tag_Status",
             CV => "Tag_CounterValue");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the data type (INT = 32767) is reached.

The "Tag_Status" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" operand. In all other cases, the "Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basics of SCL (Page 7293)

CTD: Count down

Description

The "Count down" instruction is used to decrement the value at the parameter CV. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type (INT). When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the signal state of the LD parameter is "1", the signal state of the CD parameter has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTD (for example, "CTD_DB")
- Declaration as a local tag of the type CTD in the "Static" section of a block (for example, #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case the desired initial value for the CV parameter is specified in the PV parameter. If instances of the "Count down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

IEC counter as data block of system data type IEC_<Counter> (Shared DB)

You can declare the IEC counter as a data block as follows:

<IEC_Counter_DB>.CTD();

IEC counter as local tag from the block interface (multi-instance)

You can declare the IEC counter as a local tag as follows:

#myLocal_Counter();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"CTD_DB".CTD(CD := "Tag_Start",
             LD := "Tag_Load",
             PV := "Tag_PresetValue",
             Q => "Tag_Status",
             CV => "Tag_CounterValue");
```

When the signal state of the "Tag_Start" changes from "0" to "1", the "Count down" instruction is executed and the value of the "Tag_CounterValue" operand is decremented by one. With each additional positive signal edge, the counter value is decremented until it reaches the low limit of the (INT = -32768) data type.

The operand "Tag_Status" has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basics of SCL (Page 7293)

CTUD: Count up and down

Description

Use the "Count up and down" instruction to increment or decrement the counter value at the CV parameter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the counter value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in a program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit of the specified data type (INT). When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of the specified data type (INT) is reached, the counter value is not decremented any further.

When the signal state of the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has signal state "1", the signal state of the CU and CD parameters has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0". You can also specify a constant for the PV parameter.

You can query the status of the down counter at the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")
- Declaration as a local tag of the type CTUD in the "Static" section of a block (for example, #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a startup OB:

- When used as up counter, the value at the R parameter must be set to "1".
- When used as down counter, the value at the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

IEC counter as data block of system data type IEC_<Counter> (Shared DB)

You can declare the IEC counter as a data block as follows:

<IEC_Counter_DB>.CTUD();

IEC counter as local tag from the block interface (multi-instance)

You can declare the IEC counter as a local tag as follows:

#myLocal_Counter();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P	Value at which the QU output is set / value to which the CV output is set when LD = 1.
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	INT	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"CTUD_DB".CTUD(CU := "Tag_Start1",
               CD := "Tag_Start2",
               LD := "Tag_Load",
               R := "Tag_Reset",
               PV := "Tag_PresetValue",
               QU => "Tag_CU_Status",
               QD => "Tag_CD_Status",
               CV => "Tag_CounterValue");
```

If the "Tag_Start1" operand has a positive signal edge in the signal state, the current counter value is incremented by one and stored in the "Tag_CounterValue" operand. If the "Tag_Start2" operand has a positive signal edge in the signal state, the counter value is decremented by one and is also stored in the "Tag_CounterValue" operand. The counter value is incremented on a positive signal edge of the CU parameter until it reaches the high limit of 32 767. The counter value is decremented on a positive signal edge of the CD parameter until it reaches the low limit of INT = -32 768.

The operand "Tag_CU_Status" has signal state "1" as long as the current counter value is greater than or equal to the value of the operand "Tag_PresetValue". In all other cases, the "Tag_CU_Status" output has signal state "0".

The operand "Tag_CD_Status" has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_CD_Status" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Instances (Page 62)

Basics of SCL (Page 7293)

SIMATIC Counters

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is provided at the parameter CV. The counter value is incremented until the limit of "999" is reached. When the limit value is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state of the S parameter changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if the result of logic operation (RLO) at the CU input is "1", the counter counts once in the next cycle, even when no signal edge change is detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has the signal state "1", a change in the signal state of the parameters CU and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
C_NO	Input	COUNTER, INT	C	Counters The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
S	Input	BOOL	I, Q, M, D, L	Input for presetting the counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	I, Q, M, D, L	Reset input
Q	Output	BOOL	I, Q, M, D, L	Status of the counter
CV	Output	WORD	I, Q, M, D, L, P	Current counter value
Function value		WORD	I, Q, M, D, L, P	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CU(C_NO := "Counter_1",
                     CU := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

If the signal state of the "Tag_Start" parameter changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. If the signal state of the "Tag_1" input changes from "0" to "1", the counter value in BCD format

is set to the value of the "Tag_PresetValue" operand. The counter value is reset to "0" when the "Tag_Reset" operand has signal state "1".

The current counter value is stored in hexadecimal form in the operand "Tag_Value".

The "Tag_Status" output has the signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the current counter value is decreased by one. The current counter value is provided at the parameter CV. The counter value is decreased until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decreased on a positive signal edge.

When the signal state of the S parameter changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if the result of logic operation (RLO) at the CD parameter is "1", the counter counts once in the next cycle, even when no signal edge change is detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has the signal state "1", a change in the signal state of the parameters CD and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
C_NO	Input	COUNTER, INT	C	Counters The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L	Count down input

Parameters	Declaration	Data type	Memory area	Description
S	Input	BOOL	I, Q, M, D, L	Input for presetting the counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	I, Q, M, D, L	Reset input
Q	Output	BOOL	I, Q, M, D, L	Status of the counter
CV	Output	WORD	I, Q, M, D, L, P	Current counter value
Function value		WORD	I, Q, M, D, L, P	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CD(C_NO := "Counter_1",
                     CD := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decreased by one. If the signal state of the "Tag_1" operand changes from "0" to "1", the counter value in BCD format is set to the value of the "Tag_PresetValue" operand. The counter value is reset to "0" when the "Tag_Reset" operand has signal state "1".

The current counter value is stored in the "Tag_Value" operand.

The "Tag_Status" operand returns signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment and decrement the value of a counter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the counter value is decreased by one. The current counter value is provided at the parameter CV. If there is a positive signal edge at the parameters CU and CD in one program cycle, the counter value remains unchanged.

The counter value is incremented until the high limit of "999" is reached. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit "0" is reached, the counter value is not decremented any further.

When the signal state of the S parameter changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if the result of logic operation (RLO) of the CU and CD parameters is "1", the counter will count once in the next cycle, even if no signal edge change was detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the parameter R has the signal state "1", processing of the signal state at the parameters CU,CD and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
C_NO	Input	COUNTER, INT	C	Counters The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
S	Input	BOOL	I, Q, M, D, L	Input for presetting the counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	I, Q, M, D, L	Reset input
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

Parameters	Declaration	Data type	Memory area	Description
CV	Output	WORD	I, Q, M, D, L, P	Current counter value (hexa-decimal)
Function value		WORD	I, Q, M, D, L, P	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CUD(C_NO := "Counter_1",
                      CU := "Tag_CU",
                      CD := "Tag_CD",
                      S := "Tag_1",
                      PV := "Tag_PresetValue",
                      R := "Tag_Reset",
                      Q => "Tag_Status",
                      CV => "Tag_Value");
```

When a positive signal edge is detected in the signal state of the "Tag_CU" operand and the current counter value is less than "999", the counter value is incremented by one. When a positive signal edge is detected in the signal state of the "Tag_CD" operand and the current counter value is greater than "0", the counter value is decremented by one.

If the signal state of the "Tag_1" operand changes from "0" to "1", the counter value in BCD format is set to the value of the "Tag_PresetValue" operand. The counter value is reset to "0" when the "Tag_Reset" operand has signal state "1".

The current counter value is stored in the "Tag_Value" operand.

The "Tag_Status" operand returns signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Math functions

ABS: Form absolute value

Description

Use the "Form absolute value" instruction to calculate the absolute value of an input value and to save the result in the specified operands.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Integers, floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := ABS("Tag_Value");
"Tag_Result2" := ABS("Tag_Value1"*"Tag_Value2");
```

The absolute value of the input value is returned in the format of the input value as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	-2
Tag_Result1	2
Tag_Value1	4
Tag_Value2	-1
Tag_Result2	4

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

MIN: Get minimum

Description

The instruction "Get minimum" compares the values of the specified inputs and returns the lowest value as the result. The instruction is only executed if the tags of all inputs are of the same data type.

A minimum of two and a maximum of 32 inputs can be specified for execution of the instruction.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	Second input value
INn	Input	Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	Additionally inserted inputs whose values are to be compared
Function value		Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MIN(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	12222
IN2	Tag_Value2	14444
IN3	Tag_Value3	13333
Function value	Tag_Result	12222

The instruction compares the values of the specified operands and copies the lowest value ("Tag_Value1") to the "Tag_Result" operand.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

MAX: Get maximum

Description

The instruction "Get maximum" compares the values of the specified inputs and returns the greatest value as the result. The instruction is only executed if the tags of all inputs are of the same data type.

A minimum of two and a maximum of 32 inputs can be specified for execution of the instruction.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	Second input value
INn	Input	Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	Additionally inserted inputs whose values are to be compared
Function value		Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MAX(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	12222
IN2	Tag_Value2	14444

Parameters	Operand	Value
IN3	Tag_Value3	13333
Function value	Tag_Result	14444

The instruction compares the values of the specified operands and copies the greatest value ("Tag_Value2") to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

LIMIT: Set limit value

Description

The "Set limit value" instruction limits the value of the parameter IN to the values of the parameters MN and MX. The value of the parameter MN may not be greater than the value of the parameter MX.

If the value of the IN parameter fulfills the condition MN <= IN <= MX, it is returned as the result of the instruction. If the condition is not fulfilled and the IN input value is less than the MN low limit, the value of the MN parameter is returned as the result. If the high limit MX is exceeded, the value of the MX parameter is returned as the result.

If the value of the MN input is greater than the value of the MX input, the result is undefined.

The instruction is only executed if the operands of all parameters are of the same data type.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
MN	Input	Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	Low limit
IN	Input	Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	Input value
MX	Input	Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	High limit
Function value		Integers, floating-point numbers, TIME, DATE, DT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := LIMIT(MN := "Tag_Minimum",
                      IN := "Tag_Value",
                      MX := "Tag_Maximum");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_Minimum	12000
IN	Tag_Value	8000
MX	Tag_Maximum	16000
Function value	Tag_Result	12000

The value of operand "Tag_Value" is compared with the values of operands "Tag_Minimum" and "Tag_Maximum". Because the value of the operand "Tag_Value" is less than the low limit value, the value of operand "Tag_Minimum" will be copied to operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SQR: Form square

Description

Use the "Form square" instruction to square the input value and save the result in the specified operand.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := SQR("Tag_Value");
"Tag_Result2" := SQR((SQR("Tag_Value1"))*"Tag_Value2");
```

The square of the input value is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	6.25
Tag_Value1	6.0
Tag_Value2	2.0
Tag_Result2	5184.0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SQRT: Form square root

Description

Use the "Form square root" instruction to calculate the square root of the input value and save the result in the specified operand. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number. If the input value is "0", the result is also "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := SQRT("Tag_Value");
"Tag_Result2" := SQRT((SQR("Tag_Value1"))+"Tag_Value2");
```

The square root of the input value is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	4.0
Tag_Result1	2.0
Tag_Value1	3.0
Tag_Value2	16.0
Tag_Result2	5.0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e=2.718282) from the input value. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL
```
"Tag_Result1" := LN("Tag_Value");
"Tag_Result2" := LN("Tag_Value1"+"Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	0.916
Tag_Value1	1.5
Tag_Value2	3.2
Tag_Result2	1.548

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

EXP: Form exponential value

Description

The "Form exponential value" instruction calculates the exponent from the base e (e = 2.718282) and the input value and saves the result in the specified operand.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value
Function value		Floating-point numbers	I, Q, M, D, L, P	Exponential value of the in-put value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL
```
"Tag_Result1" := EXP("Tag_Value");
```

SCL

```
"Tag_Result2" := EXP("Tag_Value1"/"Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Resultxy" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	20.5
Tag_Result1	799 902 200
Tag_Value1	15.5
Tag_Value2	30.2
Tag_Result2	1.671

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the input value. The input value must be given in radians.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value (size of an angle in radians)
Function value		Floating-point numbers	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SIN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796 (π/2)
Tag_Result	1.0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the input value. The input value must be given in radians.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value (size of an angle in radians)
Function value		Floating-point numbers	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := COS("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796 (π/2)
Tag_Result	0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the sine of the input value. The input value must be given in radians.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Input value (size of an angle in radians)
Function value		Floating-point numbers	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := TAN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+3.141593 (π)
Tag_Result	0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from a sine value, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in radians and can range in value from -π/2 to +π/2.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Sine value
Function value		Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ASIN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+1.570796 (π/2)

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from a cosine value, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in radians and can range in value from 0 to +π.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Cosine value
Function value		Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ACOS("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	0
Tag_Result	+1.570796 (π/2)

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from a tangent value, which corresponds to this value. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) as input value. The calculated angle size is given in radians and can range in value from -π/2 to +π/2.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	Floating-point numbers	I, Q, M, D, L, P	Tangent value
Function value		Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ATAN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+0.785398 (π/4)

See also

Overview of the valid data types (Page 231)

Invalid floating-point numbers (Page 253)

Basics of SCL (Page 7293)

Move operations

BLKMOV: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:

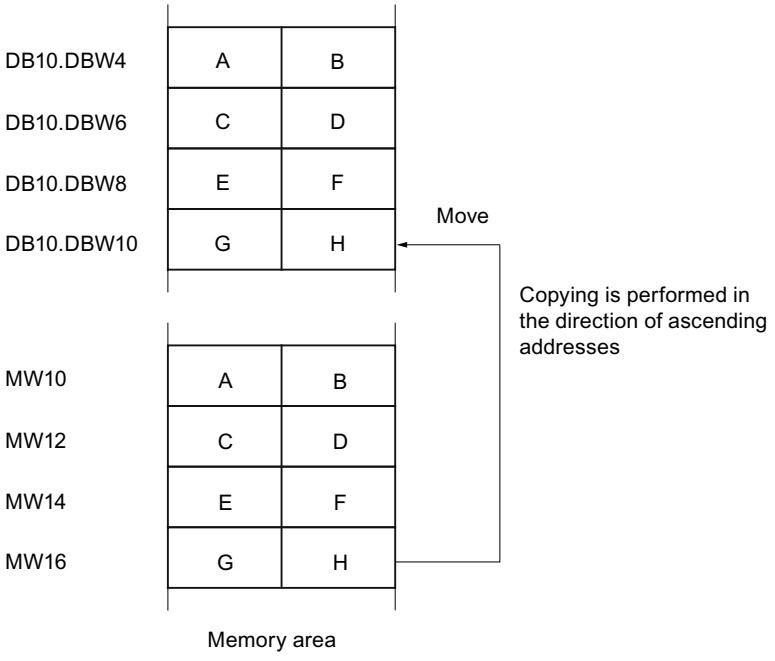

Consistency of source and destination data

Note that while the instruction "Move block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Interruptibility

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, BLKMOV is interrupted while copying from a DB that is not runtime-relevant, the execution of BLKMOV can no longer be nested.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks not runtime-relevant

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If an area of the BOOL data type is copied, the specified length of the area must be divisible by 8, otherwise the instruction will not execute.

Note

If the actual source or destination area is smaller than a destination or source area specified by the parameter SRCBLK or DSTBLK, the response depends on the CPU type:

- The following applies to S7-300 CPUs:
  No data is transferred. Error code W#16#837F is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data is transferred. Error code W#16#8122 or W#16#8323 is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data is transferred. Error code W#16#8122 or W#16#8323 is output at the parameter RET_VAL. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Rules for moving data blocks that are not runtime-relevant

The source area can also be in a data block in load memory that is not runtime-relevant. Data blocks that are not runtime-relevant are indicated by the keyword UNLINKED.

If a data block that is not runtime-relevant is copied to work memory with the "Move block" instruction and loaded at the same time, for example, by the programming device, the execution of the instruction can be delayed for several milliseconds. This results in a longer OB cycle and may trip the cycle monitoring.

If a data block that is not runtime-relevant is moved with the "Move block" instruction and the move operation is interrupted, the execution of the instruction can no longer be continued.

If the "Read from data block in the load memory" instruction is available on your CPU, you must use this instruction to read data blocks that are not runtime-relevant from load memory. If you use the "Move block" instruction, error W#16#8092 is output.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
DSTBLK	Output [1]	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
Function value (RET_VAL)		INT	I, Q, M, D, L, P	Error information
1) The DSTBLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8xyy	For additional information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_RetVal" := BLKMOV(SRCBLK := P#M100.0 BYTE 10,
                       DSTBLK => P#DB1.DBX0.0 BYTE 10);
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If an area of the BOOL data type is copied, the specified length of the area must be divisible by 8, otherwise the instruction will not execute.

You can use the "Move block uninterruptible" instruction to move a maximum of 512 bytes. Note the CPU-specific restrictions for this.

Note

If the actual source or destination area is smaller than a destination or source area specified by the parameter BVAL or BLK, the response depends on the CPU type:

- The following applies to S7-300 CPUs:
  No data is transferred. Error code W#16#837F is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data is transferred. Error code W#16#8122 or W#16#8323 is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data is transferred. Error code W#16#8122 or W#16#8323 is output at the parameter RET_VAL. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both of the data type STRING, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the data type STRING are moved, you must specify "1" as the area length.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
DSTBLK	Output [1)]	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).
Function value (RET_VAL)		INT	I, Q, M, D, L, P	Error information
1) The DSTBLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source or target area is only available in the load memory.
8xyy	For additional information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_RetVal" := UBLKMOV(SRCBLK := P#M100.0 BYTE 10,
                        DSTBLK => P#DB1.DBX0.0 BYTE 10);
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_RetVal" tag.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

FILL: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:

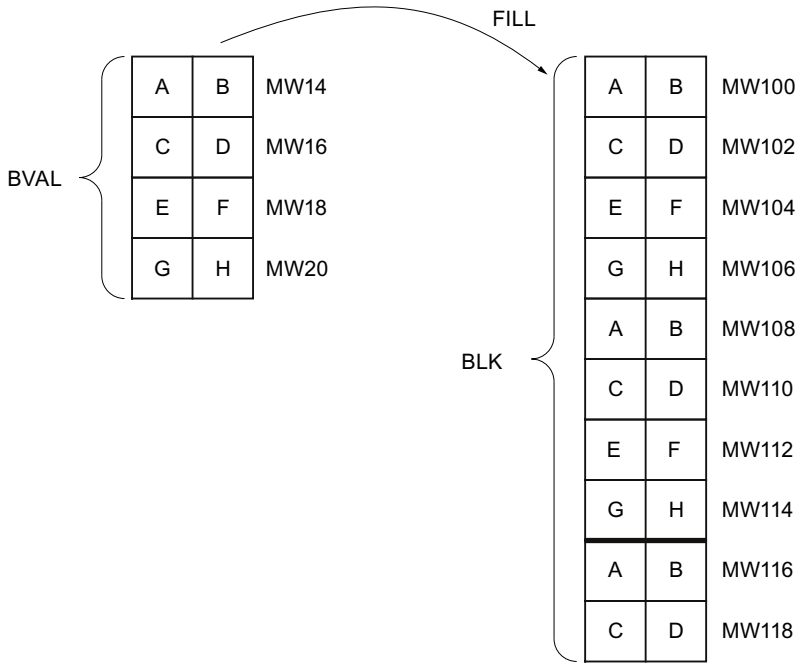

Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, the specified length must be divisible by 8; otherwise the instruction will not execute.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Note

If the actual source or destination area is smaller than a destination or source area specified by the parameter BVAL or BLK, the response depends on the CPU type:

- The following applies to S7-300 CPUs:
  No data is transferred. Error code W#16#837F is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4 CPUs (V4, V4 H-CPU and V4.5 H CPU):
  No data is transferred. Error code W#16#8122 or W#16#8323 is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data is transferred. Error code W#16#8122 or W#16#8323 is output at the parameter RET_VAL. This error code has to be evaluated in your user program.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
BVAL	Input	ANY	I, Q, M, D, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
BLK	Output [1)]	ANY	I, Q, M, D, L, P	Specification of the memory area that will be filled with the content of the source area.
Function value (RET_VAL)		INT	I, Q, M, D, L, P	Error information
1) The BLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

For additional information on valid data types, refer to "See also".

BVAL parameter

When you transfer a structure as an input parameter, remember that the length of a structure is always based on an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8xyy	For additional information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_RetVal" := FILL(BVAL := P#M14.0 WORD 4,
                     BLK => P#M100.0 WORD 10);
```

The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

Conversion operations

CONVERT: Convert value

Description

Use the "Convert value" instruction to program explicit conversions. You determine the data types to be converted in a dialog box with opens automatically when you insert the instruction. When executed, the instruction reads the source value and converts it into the specified target value.

For information on possible conversions, refer to the "Explicit conversion" section at "See also".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Source type>	Input	Bit strings, integers, floating-point numbers, times, date and time, character strings	I, Q, M, D, L, P or constant	Value to be converted.
<Destination type>	Output	Bit strings, integers, floating-point numbers, times, date and time, character strings	I, Q, M, D, L, P	Result of the conversion

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_INT" := REAL_TO_INT("Tag_REAL");
```

The following table shows how the instruction works using specific operand values:

Operand	Data type	Value
Tag_REAL	REAL	20.56
Tag_INT	INT	21

During the conversion, the value of the "Tag_REAL" operand will be rounded to the nearest integer and saved in the "Tag_INT" operand.

See also

Overview of the valid data types (Page 231)

Explicit conversions (Page 569)

Basics of SCL (Page 7293)

ROUND: Round numerical value

Description

The "Round numerical value" instruction is used to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts it into an integer or floating-point number. If the input value is exactly between an even and odd number, the even number is selected.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	REAL	I, Q, M, D, L, P	Input value to be rounded.
Function value		DINT	I, Q, M, D, L, P	Result of the rounding

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ROUND("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	1.50000000	-1.50000000
Tag_Result	2	-2

The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

CEIL: Generate next higher integer from floating-point number

Description

Use the "Generate next higher integer from floating-point number" instruction to round the value to the nearest integer. The instruction interprets the input value as floating-point number and converts it to the next higher integer. The function value can be greater than or equal to the input value.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	REAL	I, Q, M, D, L, P	Input value
Function value		DINT	I, Q, M, D, L, P	Input value rounded up

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := CEIL("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.5	-0.5
Tag_Result	1	0

The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

FLOOR: Generate next lower integer from floating-point number

Description

Use the "Generate next lower integer from floating-point number" instruction to round the value of a floating point number to the next lower integer. The instruction interprets the input value as floating-point number and converts it to the next lower integer. The function value can be equal or less than the input value.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expres-sion>	Input	REAL	I, Q, M, D, L, P	Input value
Function value		DINT	I, Q, M, D, L, P	Input value rounded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := FLOOR("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.5	-0.5
Tag_Result	0	-1

The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

TRUNC: Truncate numerical value

Description

The "Truncate numerical value" instruction is used to generate an integer from the input value without rounding. The instruction selects only the integer part of the input value and returns this part without decimal places as the function value.

Use the following syntax to change the data type of the instruction:

TRUNC_<data type>();

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Expression>	Input	REAL	I, Q, M, D, L, P	Input value
_<Data type>		Integers default: DINT		Data type of the function value: • You do not need to specify the data type if using the default. • Any other valid data type you may use must be declared explicitly.
Function value		Integers	I, Q, M, D, L	Integer component of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := TRUNC("Tag_Value1");
"Tag_Result2" := TRUNC("Tag_Value2"+"Tag_Value3");
"Tag_Result3" := TRUNC_REAL("Tag_Value4");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value1	-1.5
Tag_Result1	-1
Tag_Value2	2.1
Tag_Value3	3.2
Tag_Result2	5
Tag_Result3	2.0
Tag_Value4	2.4

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SCALE: Scale

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number that can be scaled in physical units between a low limit value and a high limit value. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the "K1" and "K2" constants are determined by the signal state on the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled inversely proportionate to the input value.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
Function value (RET_VAL)		WORD	I, Q, M, D, L, P	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the parameter IN is greater than 27 648 or is less than 0 (unipolar) or -27 648 (bipolar).
8xyy	For additional information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
"Tag_ErrorCode" := SCALE(IN := "Tag_InputValue",
                         HI_LIM := "Tag_HighLimit"
                         LO_LIM := "Tag_LowLimit"
                         BIPOLAR := "Tag_Bipolar",
                         OUT => "Tag_Result");
```

The error information of the instruction is returned in the operand "Tag_ErrorCode" as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_Result	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

UNSCALE: Unscale

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at parameter IN is not within the limits defined by HI_LIM and LO_LIM, an error is output and the result is set to the nearest limit.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
Function value (RET_VAL)		WORD	I, Q, M, D, L, P	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
8xyy	For additional information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

```
SCL
"Tag_ErrorCode" := UNSCALE(IN := "Tag_InputValue",
                           HI_LIM := "Tag_HighLimit"
                           LO_LIM := "Tag_LowLimit"
                           BIPOLAR := "Tag_Bipolar",
                           OUT => "Tag_Result");
```

The error information of the instruction is returned in the operand "Tag_ErrorCode" as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_Result	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

Program control operations

IF: Run conditionally

Description

The instruction "Run conditionally" branches the program flow depending on a condition. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Parameters

Depending on the type of branch, you can program the following forms of the instruction:

- Branch through IF:
  IF <condition> THEN <instructions>
  END_IF;

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the execution of the program continues with the next instruction after the END_IF.

- Branch through IF and ELSE:
  IF <condition> THEN <instructions1>
  ELSE <Instructions0>
  END_IF;

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the instructions programmed after the ELSE are executed. Then the execution of the program continues with the next instruction after the END_IF.

- Branch through IF, ELSIF and ELSE:
  IF <condition1> THEN <instructions1>
  ELSIF <condition2> THEN <instruction2>
  ELSE <Instructions0>
  END_IF;

If the first condition (<Condition1>) is satisfied, the instructions (<Instructions1>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If the first condition is not satisfied, the second condition (<Condition2>) is checked. If the second condition (<Condition2>) is fulfilled, the instructions (<Instructions2>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If none of the conditions are fulfilled, the instructions (<Instructions0>) after ELSE are executed followed by the execution of the program after END_IF.

You can nest as many combinations of ELSIF and THEN as you like within the IF instruction. The programming of an ELSE branch is optional.

The syntax of the IF instruction consists of the following parts:

Parameters	Data type	Memory area	Description
<Condition>	BOOL	I, Q, M, D, L	Expression to be evaluated
<Instructions>	-		Instructions to be executed with satisfied condition. An exception are instructions programmed after the ELSE. These are executed if no condition within the program loop is satisfied.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF "Tag_1" = 1
THEN "Tag_Value" := 10;
ELSIF "Tag_2" = 1
THEN "Tag_Value" := 20;
ELSIF "Tag_3" = 1
THEN "Tag_Value" := 30;
ELSE "Tag_Value" := 0;
END_IF;
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_1	1	0	0	0
Tag_2	0	1	0	0
Tag_3	0	0	1	0
Tag_Value	10	20	30	0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

CASE: Create multiway branch

Description

The instruction "Create multiway branch" executes one of several instruction sequences depending on the value of a numerical expression.

The value of the expression must be an integer. When the instruction is executed, the value of the expression is compared with the values of several constants. If the value of the expression agrees with the value of a constant, the instructions programmed directly after this constant are executed. The constants can assume the following values:

- An integer (for example, 5)
- A range of integers (for example 15...20)
- An enumeration consisting of integers and ranges (for example, 10, 11, 15...20)

You can declare the instruction as follows:

CASE <Tag> OF

<Constant1>: <Instructions1>;

<Constant2>: <Instructions2>;

<ConstantX>: <InstructionsX>; // X >= 3

ELSE <Instructions0>;

END_CASE;

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
<Tag>	Integers	I, Q, M, D, L	Value which is compared to the programmed constant values.
<Constant>	Integers	-	Constant values which form the condition for the execution of an instruction sequence. The constants can assume the following values: • An integer (for example, 5) • A range of integers (for example 15...20) • An enumeration consisting of integers and ranges (for example, 10, 11, 15...20)
<Instruction>	-	-	Any instructions which are executed if the value of the expression agrees with the value of a constant. An exception are instructions programmed after the ELSE. These instructions are executed if the values do not agree.

For additional information on valid data types, refer to "See also".

If the value of the expression agrees with the value of the first constant (<Constant1>), the instructions (<Instructions1>) which are programmed directly after the first constant are executed. Program execution subsequently resumes after the END_CASE.

If the value of the expression does not agree with the value of the first constant (<Constant1>), this value is compared to the value of the constant which is programmed next. In this way, the CASE instruction is executed until the values agree. If the value of the expression does not correspond to any of the programmed constant values, the instructions (<Instructions0>) which are programmed after the ELSE are executed. ELSE is an optional part of the syntax and can be omitted.

The CASE instruction can also be nested by replacing an instruction block with CASE. END_CASE represents the end of the CASE instruction.

Example

The following example shows how the instruction works:

SCL

```
CASE "Tag_Value" OF
     0 :
          "Tag_1" := 1;
     1,3,5 :
          "Tag_2" := 1;
     6...10 :
          "Tag_3" := 1;
     16,17,20...25 :
          "Tag_4" := 1;
     ELSE
          "Tag_5" := 1;
END_CASE;
```

The following table shows how the instruction works using specific operand values:

Operand	Values				
Tag_Value	0	1, 3 , 5	6, 7, 8, 9, 10	16, 17, 20, 21, 22, 23, 24, 25	2
Tag_1	1	-	-	-	-
Tag_2	-	1	-	-	-
Tag_3	-	-	1	-	-
Tag_4	-	-	-	1	-
Tag_5	-	-	-	-	1
1: The operand is set to the signal state "1". -: The signal state of the operand remains unaltered.					

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

FOR: Run in counting loop

Description

The instruction "Run in counting loop" causes repeated execution of a program loop until a loop variable lies within a specified value range.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Note

Information on the number of runs and the run tag

The number of runs cannot be changed while the program is running.

For performance reasons, the run tag should be declared in the block interface in the "Temp" section. In addition, the run tag should not be changed while the program is running. Otherwise the instruction may return an undefined result depending on the implementation.

Limits for FOR statements

To program "safe" FOR statements that do not run endlessly, observe the following rules and limits:

FOR <Run_tag> := <Start_value> TO <End_value> BY <Increment> DO <Instructions>;

END_FOR;

If ...	... then	Note
Start value < End value	End value < (PMAX increment)	Run tag runs in positive direction
Start value > End value AND increment < O	End value > (NMAX increment)	Run tag runs in negative direction

Limits

Different limits apply to the possible data types:

Data type	PMAX	NMAX
Run tag of type INT	32767	-32768
Run tag of type DINT	2147483647	-2147483648

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
<Run tag>	INT, DINT	I, Q, M, D, L	Operand whose value is evaluated with the loop execution. The data type of the loop variable determines the data type of the other parameters.
<Start value>	INT, DINT	I, Q, M, D, L	Expression whose value is allocated at the start of the loop execution of the loop variables.

Parameters	Data type	Memory area	Description
<End value>	INT, DINT	I, Q, M, D, L	Expression whose value defines the last run of the program loop. The value of the loop variable is checked before each loop: • End value not reached: The instructions according to DO are executed • End value is reached: The FOR loop is executed one last time • End value exceeded: The FOR loop is completed An alteration to the end value is not permitted during execution of the instruction.
<Increment>	INT, DINT	I, Q, M, D, L	Expression by whose value the loop variable is increased (positive increment) or decreased (negative increment) after each loop. Specification of the increment is optional. If no increment is given, the value of the loop variable is increased by 1 after each loop. An alteration of the increment is not permitted during execution of the instruction.
<Instructions>	-		Instructions which are carried out with each loop, as long as the value of the loop variable lies within the value range. The value range is defined by the start and end values.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
FOR i := 2 TO 8 BY 2
DO "a_array[i] := "Tag_Value"*"b_array[i]";
END_FOR;
```

The operand "Tag_Value" is multiplied with the elements (2, 4, 6, 8) of the ARRAY tag "b_array". The result is read in to the elements (2, 4, 6, 8) of the ARRAY tag "a_array".

See also

CONTINUE: Recheck loop condition (Page 1841)

EXIT: Exit loop immediately (Page 1842)

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

WHILE: Run if condition is met

Description

The instruction "Run if condition is met" causes a program loop to be repeatedly executed until the implementation condition is satisfied. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

You can declare the instruction as follows:

WHILE <Condition> DO <Instructions>;

END_WHILE;

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
<Condition>	BOOL	I, Q, M, D, L	Expression which is evaluated before each loop.
<Instructions>	-		Instructions to be executed with satisfied condition. If the condition has not been satisfied, program execution continues after END_WHILE.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
WHILE
      "Tag_Value1" <> "Tag_Value2"
      DO "Tag_Result"
      := "Tag_Input";
END_WHILE;
```

As long as the values of the operands "Tag_Value1" and "Tag_Value2" do not match, the value of the operand "Tag_Input" is allocated to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

CONTINUE: Recheck loop condition (Page 4972)

EXIT: Exit loop immediately (Page 4973)

Basics of SCL (Page 7293)

REPEAT: Run if condition is not met

Description

The instruction "Run if condition is not met" causes a program loop to be repeatedly executed until a termination condition is met. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

The instructions are executed once, even if the termination condition is fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

You can declare the instruction as follows:

REPEAT <instructions>;

UNTIL <condition> END_REPEAT;

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
<Instructions>	-		Instructions that are executed as long as the programmed condition has the value FALSE. The instructions are executed once, even if the termination condition is fulfilled.
<Condition>	BOOL	I, Q, M, D, L	Expression which is evaluated after each loop. If the expression has the value FALSE, the program loop is executed once again. If the expression has the value TRUE, the program loop continues after END_REPEAT.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
REPEAT "Tag_Result"
        := "Tag_Value";
UNTIL "Tag_Error"
END_REPEAT;
```

As long as the value of the operand "Tag_Error" has the signal state "0", the value of the operand "Tag_Value" is allocated to the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

CONTINUE: Recheck loop condition

Description

The "Recheck loop condition" instruction ends the current program run of a FOR, WHILE or REPEAT loop.

After execution of the instruction, the conditions for the continuation of the program loop are evaluated again. The instruction affects the program loop which directly contains the instruction.

Example

The following example shows how the instruction works:

```
SCL
FOR i
     := 1 TO 15 BY 2 DO
     IF (i < 5) THEN
     CONTINUE;
END_IF;
      "DB10".Test[i] := 1;
END_FOR;
```

For additional information on valid data types, refer to "See also".

If the condition i < 5 is satisfied, the subsequent value allocation ("DB10".Test[i] := 1) will not be executed. The run variable (i) is increased by the increment of "2" and checked to see whether its current value lies in the programmed value range. If the run variable lies in the value range, the IF condition is evaluated again.

If the condition i < 5 is not satisfied, the subsequent value allocation ("DB10".Test[i] := 1) will be executed and a new loop started. In this case, the run variable is also increased by the increment "2" and checked.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

EXIT: Exit loop immediately

Description

The instruction "Exit loop immediately" cancels the execution of a FOR, WHILE or REPEAT loop at any point regardless of conditions. The execution of the program is continued after the end of the loop (END_FOR, END_WHILE, END_REPEAT).

The instruction affects the program loop which directly contains the instruction.

Example

The following example shows how the instruction works:

SCL

```
FOR i := 15 TO 1 BY -2 DO
IF (i < 5)
THEN EXIT;
END_IF;
"DB10".Test[i] := 1;
END_FOR;
```

For additional information on valid data types, refer to "See also".

If the condition i < 5 is satisfied, then the execution of the loop will be canceled. Program execution resumes after the END_FOR.

If the condition i < 5 is not satisfied, the subsequent value allocation ("DB10".Test[i] :=1) will be executed and a new loop started. The run tag (i) is decreased by the increment of "-2" and it is checked whether its current value lies in the programmed value range. If the (i) run variable lies within the value range, the IF condition is evaluated again.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

GOTO: Jump

Description

Use the instruction "Jump" to resume the execution of a program at a given point marked with a jump label.

The jump labels and the instruction "Jump" must be in the same block. The name of a jump label can only be assigned once within a block. Each jump label can be the target of several jump instructions.

A jump from the "outside" into a program loop is not permitted, but a jump from a loop to the "outside" is possible.

Adhere to the following grammatical rules for jump labels:

- Letters (a to z, A to Z)
- A combination of letters and numbers. Check that the order is correct, i.e. first the letters, then the numbers (a - z, A - Z, 0 - 9).
- Special characters or a combination of letters and numbers must not be used in reverse order, i.e. first the numbers and then the letters (0-9, a - z, A - Z).

You can declare the instruction as follows:

GOTO <jump label>;

...

<Jump label>: <Instructions>

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Description
<jump label>	-	Jump label to be jumped to
<instructions>	-	Instructions which are executed after the jump.

Example

The following example shows how the instruction works:

SCL

```
CASE "Tag_Value" OF
1 : GOTO MyLABEL1;
2 : GOTO MyLABEL2;
3 : GOTO MyLABEL3;
ELSE GOTO MyLABEL4;
END_CASE;
MyLABEL1: "Tag_1" := 1;
MyLABEL2: "Tag_2" := 1;
MyLABEL3: "Tag_3" := 1;
MyLABEL4: "Tag_4" := 1;
```

Depending on the value of the "Tag_Value" operand, the execution of the program will resume at the point identified by the corresponding jump label. If the operand "Tag_Value" has the value 2, for example, program execution will resume at the jump label "MyLABEL2". The program line identified by the jump label "MyLABEL1" will be skipped in this case.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

RETURN: Exit block

Description

The instruction "Exit block" exits the program execution in the currently edited block and continues in the calling block.

The instruction can be omitted at the end of the block.

Example

The following example shows how the instruction works:

SCL

```
IF "Tag_Error" <>0 THEN RETURN;
END_IF;
```

If the signal state of the "Tag_Error" operand is not zero, execution of the program ends in the block currently being processed.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

(*...*): Insert a comment section

Description

You can use the "Insert a comment section" instruction to add a comment section. The text within the parenthesis "(*...*)" is treated as a comment.

Example

The following example shows how the instruction works:

SCL

```
(*This is a comment section.*)
```

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

REGION: Structure program code

Description

You can use the instruction "Structure program code" to structure the program code in an SCL block by dividing it into different areas.

For additional information on working with areas, please refer to:Working with regions (Page 7331)

You can declare the instruction as follows:

REGION <Name>

<Instructions>

END_REGION

Parameters

The following table shows the parameters of the instruction:

Parameters	Data type	Memory area	Description
REGION	-	-	Keyword with which the area starts
<Name>	-	-	Text designating the REGION
<Instructions>	-	-	Program code that is preceded and succeeded by the REGION
END_REGION	-	-	Keyword with which the area ends

Example

The following example shows how the instruction works:

SCL

```
REGION Feeder System
     // Structures the source code for the feeder system
   IF "Variable_1" = 0 THEN
      "Variable_2" := 10;
   END_IF;
END_REGION
```

The Feeder System area comprises the complete program code.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Runtime control

COMPRESS: Compress CPU memory (S7-400)

Description

The "Compress CPU memory" instruction is used to compress the working memory of the CPU or the RAM load memory. During this compression the gaps caused by deleting or loading blocks are filled by pushing together the existing blocks.

The compression operation is started by calling the "Compress CPU memory" instruction. The compression operation is divided into several program cycles. The compression operation is monitored by the values of the individual parameters. If value "1" is on the BUSY parameter of the instruction, the compression operation is still being executed. The value "1" on the DONE parameter reports that the compression operation has been completed. Error information can be obtained from the RET_VAL parameter.

If an externally started compression operation has already been executed, calling the "Compress CPU memory" instruction will cause an error display.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	Information whether compression operation is active.
DONE	Output	BOOL	I, Q, M, D, L	Information whether compression operation was completed.
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error The status of the compression operation can be queried on the BUSY and DONE parameters.
8091	An externally started compression operation is already being executed.

Error code* (W#16#...)	Explanation
8092	The instruction cannot be executed, because one of the following conditions applies: • The "Delete block" function is active. • A programming device function is accessing one of the blocks to be moved. • An externally started compression operation is already being executed. • The H-CPU is running a link-up or update function.
8xyy	For additional information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

RE_TRIGR: Restart cycle monitoring time

Description

The "Restart cycle monitoring time" instruction is used to restart the cycle monitoring time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called in all blocks regardless of the priority .

If the instruction is called in a block with a higher priority, for example, a hardware interrupt or a diagnostic interrupt, the instruction will not be executed.

The "Restart cycle monitoring time" instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Parameters

The "Restart cycle monitoring time" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

WAIT: Configure time delay

Description

The "Configure time delay" instruction pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays from -32768 up to 32767 microseconds (µs). The smallest possible delay time depends on the CPU and corresponds to the execution time of the instruction.

The execution of the instruction can be interrupted by higher priority events and does not return any error information.

Applies to S7-300-CPUs:

For the S7-300-CPUs (except S7-318), the delay time programmed by the instruction is a minimum time. It is extended by the execution time of the nested priority classes and by system utilizations.

Note

Negative delay time

If you specify a negative delay time at parameter WT, the enable output ENO, or the RLO and the BR bit, will return the signal state FALSE. A negative delay time does not affect the CPU. The following instructions linked to enable output ENO are not executed in LAD or FBD.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

PROTECT: Change protection level

Description

The instruction "Change protection level" is used to change the configured protection level "1" of the CPU.

The configured protection level "1" with selected option "Removable with password", means that you have to configure a password. This password enables you to write-access during operation the CPU which was read-only protected with the instruction "Change Protection Level" (e. g. Load).

You specify the protection level on the MODE parameter of the instruction. The current protection level of your CPU is displayed in the inspector window under "Properties > Protection".

The following table shows the meaning of the protection levels.

Protection level	Meaning
1	All programming device functions are permitted. Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
2	• The program and the configuration of the CPU cannot be changed. The program in the CPU can be read. • The functions for process control, process monitoring, and process communication are permitted. • All informational functions are permitted. • If you know the valid password you can cancel the set write protection. • Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
3	• You cannot cancel the set read and write protection, even if you know the valid password. • If a valid connection exists at the time when the instruction "Change protection level" is called, the call of the instruction does not affect this connection.

If you have configured protection level "2" or "3" for your CPU, using the instruction "Change protection level" has no effect.

You can use the SSL parts list with the SSL ID W#16#0232 and index W#16#0004 to check whether protection level "2" or "3" is still in effect after the execution of the "Change protection level" instruction.

Note

You can only use the "Change protection level" instruction to set a protection level that is equal to or higher than the protection level you have configured with STEP 7 "Configure hardware. You have, for example, configured protection level 3 and then called the instruction "Change protection level" with the MODE parameter = 12. This blocks the password authentication. If the instruction is called again with the parameter MODE = 0 or 1, protection level 3 is retained and the read and write protection canceled by entry of the valid password.

Effects of the operating state transitions on the protection level

The following table shows what effects the operating state transitions have on the protection level set with the "Change protection level" instruction.

Action	Effects on the protection level
• Operating mode switch to "STOP" • Unbuffered POWER ON • Memory reset with operating mode switch • Configuration and/or programming changes in the "STOP" mode of the S7-400 CPU	The protection level is set to "1". All programming device functions are permitted.
• Cold restart • Warm restart	The protection level is set by the operating system to "1". All programming device functions are permitted. If required, you can set the protection level in your program to "2" or "3" by calling the "Change protection level" instruction.
Buffered POWER ON	Remains unchanged
Operating mode transition RUN/STARTUP/HOLD -> STOP (by calling the "Exit program" instruction (STP), by program error without corresponding error OB, or by operator action).	Remains unchanged
S7-400: Restart by user action or automatically following POWER ON	Remains unchanged

Syntax

The following syntax is used for the instruction "Change protection level":

SCL

```
PROTECT(MODE:= <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	WORD	I, Q, M, D, L	Protection level Possible values: • W#16#0000: Setting protection level "1" • W#16#0001: Setting protection level "2" with password authentication • W#16#000C: Setting protection level "3" without password authentication
Function value (RET_VAL)		INT	I, Q, M, D, L	If an error occurs during processing, information on the error is output.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error
8090	Invalid value at MODE parameter.
80C3	The required operating resources are currently occupied.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

CiR: Control CiR process (S7-400)

Description

The "Control CiR process" instruction is used to influence the CiR process in the user program as follows:

- Disable CiR process. In this case, the download of a modified configuration from the programming device to the CPU is always rejected. The disable is maintained until you cancel it again with the "Control CiR process" instruction.
- You can disable the CiR process conditionally by specifying a high limit for the CiR synchronization time on the FRZ_TIME parameter. In this case, the download of a modified configuration from the programming device to the CPU is only accepted if the CiR synchronization time calculated in the CPU does not exceed the specified high limit.
- Enable CiR process. The high limit of the CiR synchronization time is set to the default value 1000 ms.
- Determine whether the CiR process is enabled or not. If the CiR process is enabled or conditionally enabled, you will get the current high limit for the CiR synchronization time in the A_FT parameter of the instruction.

Note

During the CiR synchronization period the outputs are frozen and the inputs are not evaluated.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L	Job identifier Valid values on the MODE parameter are: • 0: Information function • 1: Enable CiR process • 2: Disable CiR process • 3: Disable CiR conditionally. You specify the high limit of the CiR synchronization time in FRZ_TIME.
FRZ_TIME	Input	TIME	I, Q, M, D, L	High limit of the CiR synchronization time in milliseconds. Permitted range of values: 60 ... 2500 ms (default: 1000 ms) The FRZ_TIME parameter is only relevant when the value 3 is set on the MODE parameter.
A_FT	Output	TIME	I, Q, M, D, L	Currently valid high limit of the CiR synchronization time
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information When the MODE parameter is "0", the RET_VAL parameter supplies information on the enabling of the CiR process.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error This error code can be output only if the value "1", "2" or "3" is set on the MODE parameter.
0001	CiR is enabled. This error code can be output only if the value "0" is set on the MODE parameter.
0002	The CiR process is disabled. This error code can be output only if the value "0" is set on the MODE parameter.
0003	CiR is disabled conditionally. This error code can be output only if the value "0" is set on the MODE parameter.
8001	The CPU is not ready for a CiR operation.

Error code* (W#16#...)	Explanation
8002	The value at the MODE parameter is invalid.
8003	The value at the FRZ_TIME parameter is invalid.
8xyy	For additional information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The "Control CiR process" instruction can be used, for example, to ensure that no CiR process is initiated during periods in which maximum CPU performance is required to control the process.

To do this the "Control CiR process" instruction must be called before the start of the period with increased process activity in the program and executed with the value "2" (Disable CiR process) on the MODE parameter . The CiR process is then disabled.

To enable the CiR process again after the completion of the period with increased process activity, call the "Control CiR process" instruction again and execute it with the value "1" for the MODE parameter. Alternatively, you can also set the value 3 on the MODE parameter to disable the CiR process conditionally.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

Word logic operations

DECO: Decode

Description

The instruction "Decode" sets a bit specified by the input value in the output value.

The instruction "Decode" reads the value of the parameter IN and sets the bit in the output value, whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. If the value of the IN parameter is greater than 31, a modulo 32 instruction is executed.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P	Position of the bit in the output value which is set.
Function value		DWORD	I, Q, M, D, L, P	Current output value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := DECO(IN := "Tag_Value");
```

The following figure shows how the instruction works using specific operand values:

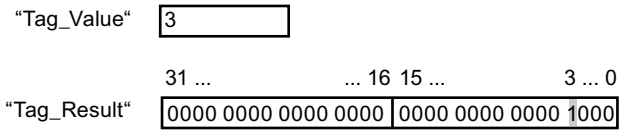

The instruction reads the number "3" from the value of the operand "Tag_Value" and sets the third bit to the value of the operand "Tag_Result".

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ENCO: Encode

Description

The instruction "Encode" reads the bit number of the lowest-value bit set in the input value and issues this as a result.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P	Input value
Function value		INT	I, Q, M, D, L, P	Bit number of the bit in the input value that is read out.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ENCO(IN := "Tag_Value");
```

The following figure shows how the instruction works using specific operand values:

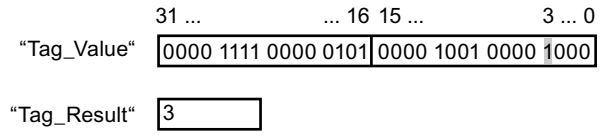

The instruction reads the lowest-value set bit of the operand "Tag_Value" and writes the bit position "3" in the operand "Tag_Result". The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SEL: Select

Description

The instruction "Select" selects one of the parameters IN0 or IN1 depending on a switch (G parameter) and issues its content as a result. When the parameter G has the signal status "0", the value at parameter IN0 is moved. When the parameter G has the signal status "1", the value at parameter IN1 is moved and returned as a function value.

The instruction is only executed if the tags of all parameters have the same data type class.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
G	Input	BOOL	I, Q, M, D, L	Switch
IN0	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	I, Q, M, D, L, P	First input value

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	I, Q, M, D, L, P	Second input value
Function value		Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SEL(G := "Tag_Value",
                    IN0 := "Tag_0",
                    IN1 := "Tag_1");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0	1
Tag_0	W#16#0000	W#16#4C
Tag_1	W#16#FFFF	D#16#5E
Tag_Result	W#16#0000	D#16#5E

The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

MUX: Multiplex

Description

The "Multiplex" instruction copies the value of a selected input parameter and issues it. You can use the parameter K to determine the number of the input parameter whose value will be moved. Numbering starts at IN0 and is incremented continuously with each new input. You can declare a maximum of 32 input parameters.

If the value of the K parameter is greater than the number of inputs and the INELSE parameter is not set, the instruction function value is invalid and the ENO enabling output is set to 0.

All tags with assigned parameters must be of the same data type. The function value assumes the data type having the highest value.

Note

K parameter has a negative integer

If a tag with a valid data type is specified at the input parameters and the K parameter has a negative integer, the value of the tag is changed.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
K	Input	Integers	I, Q, M, D, L, P	Specifies the parameter whose content is to be transferred. • If K = 0 => parameter IN0 • If K = 1 => parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	I, Q, M, D, L, P	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	I, Q, M, D, L, P	Second input value
INn	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	I, Q, M, D, L, P	Optional input values
INELSE	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	I, Q, M, D, L, P	Specifies the value to be copied when K <> n.
Function value		Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MUX(K := "Tag_Number",
                    IN0 := "Tag_1",
                    IN1 := "Tag_2",
                    INELSE := "Tag_3");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Number	1	4
Tag_1	DW#16#00000000	DW#16#00000000
Tag_2	DW#16#003E4A7D	DW#16#003E4A7D
Tag_3	DW#16#FFFF0000	DW#16#FFFF0000
Tag_Result	DW#16#003E4A7D	DW#16#FFFF0000

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

Shift and Rotate

SHR: Shift right

Description

The "Shift right" instruction shifts the contents of the IN parameter bit-by-bit to the right and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of available bit places, the value of the IN parameter is shifted to the right by the available number of bit places.

The bit positions that are freed by shifting in the left operand area are filled with zeros.

The following figure shows how the content of an operand of the INT data type is shifted four bit positions to the right:

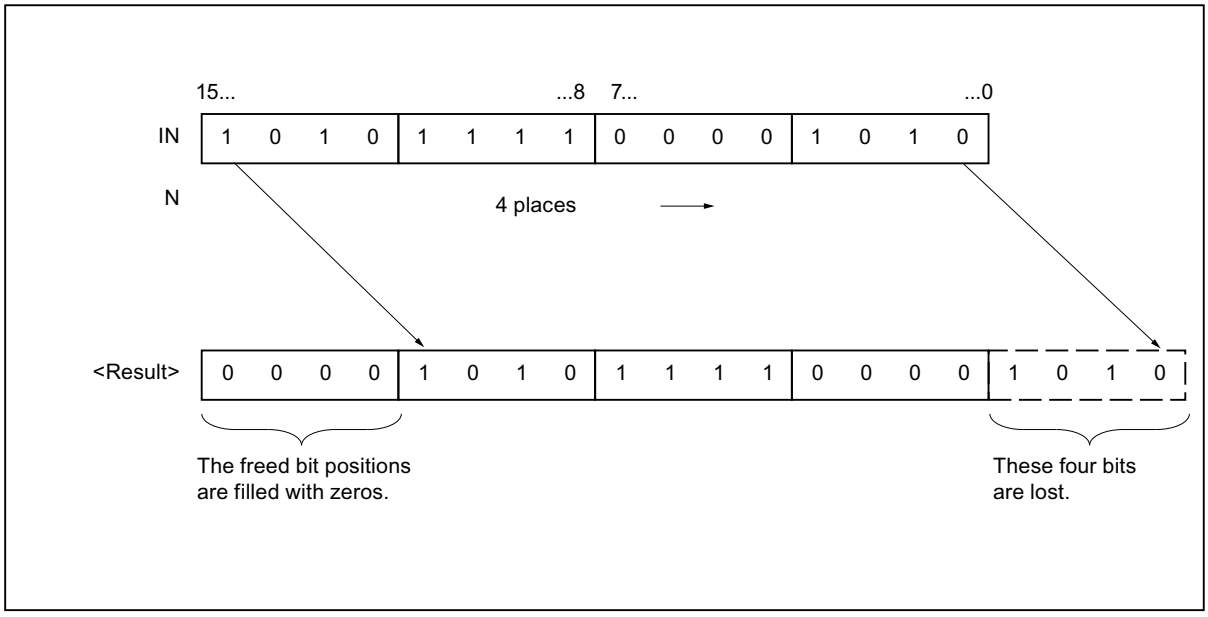

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P	Value to be shifted.
N	Input	Integers	I, Q, M, D, L, P	Number of bits by which the value (IN) is shifted. The value of the N parameter must be positive.
Function value		Bit strings	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SHR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	3
Function value	Tag_Result	0000 0111 1111 0101

The content of the "Tag_Value" operand is shifted by three bit positions to the right. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SHL: Shift left

Description

The "Shift left" instruction shifts the contents of the IN parameter bit-by-bit to the left and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of bit places, then the value of the IN parameter will be shifted to the left by the available number of bit places.

The bit positions freed by the shift are filled with zeros in the result value.

The following figure shows how the content of an operand of the WORD data type is shifted six bit positions to the left:

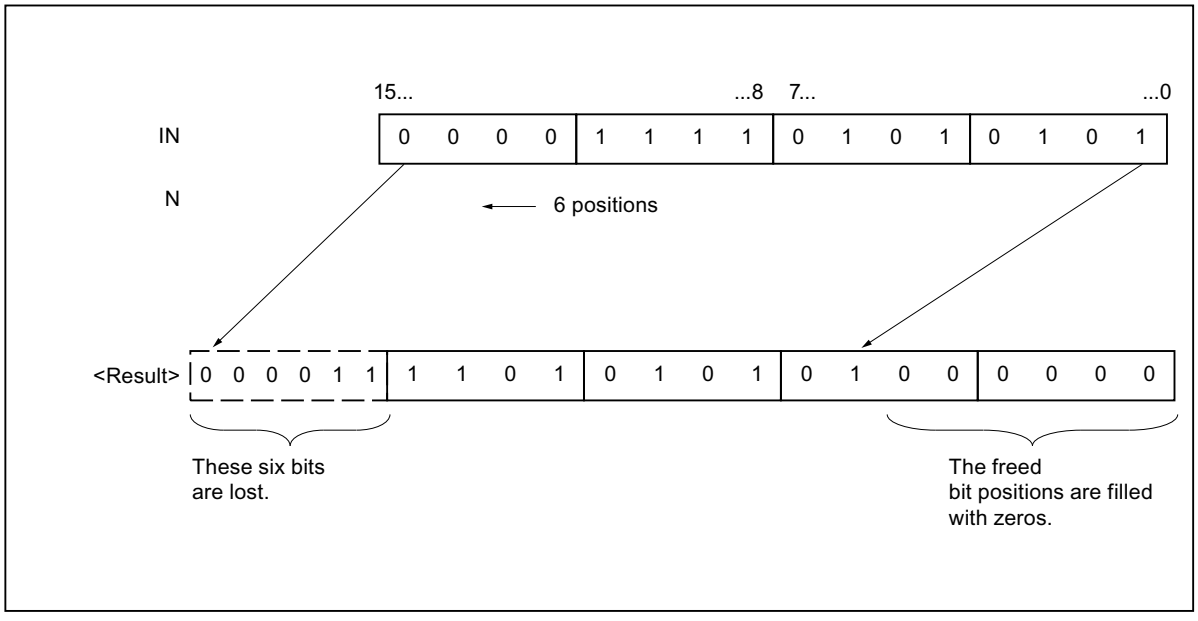

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P	Value to be shifted
N	Input	Integers	I, Q, M, D, L, P	Number of bits by which the value IN is shifted. The value of the N parameter must be positive.
Function value		Bit strings	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SHL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	4
Function value	Tag_Result	1111 1010 1111 0000

The value of the "Tag_Value" operand will be shifted by four bit places to the left. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ROR: Rotate right

Description

The "Rotate right" instruction rotates the content of the parameter IN bit-by-bit and returns it as a function value. The N parameter is used to specify the number of bit places by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:

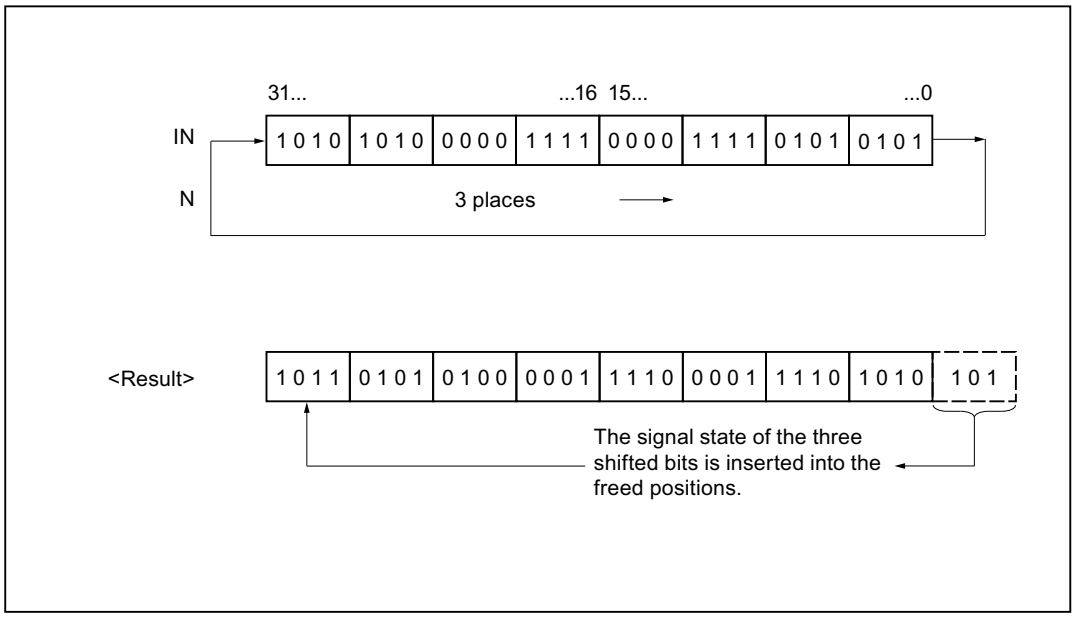

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P	Value to be rotated
N	Input	Integers	I, Q, M, D, L, P	Number of bit positions by which the value IN is rotated. The value of the N parameter must be positive.
Function value		Bit strings	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ROR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1010 1010 0000 1111 0000 1111 0101 0101
N	Tag_Number	5
Function value	Tag_Result	1010 1101 0101 0000 0111 1000 0111 1010

The content of the "Tag_Value" operand is rotated by five bit positions to the right. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

ROL: Rotate left

Description

The "Rotate left" instruction rotates the contents of the IN parameter bit-by-bit to the left and returns it as a function value. The N parameter is used to specify the number of bit places by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:

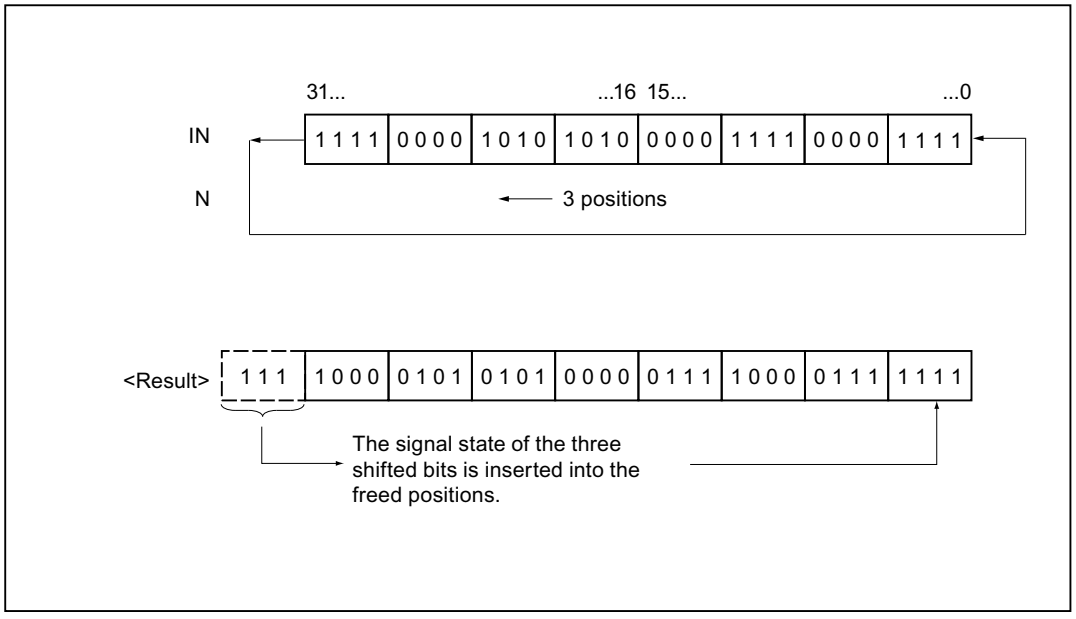

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P	Value to be rotated
N	Input	Integers	I, Q, M, D, L, P	Number of bit positions by which the value IN is rotated. The value of the N parameter must be positive.
Function value		Bit strings	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ROL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1111 0000 1010 1010 0000 1111 0000 1111
N	Tag_Number	5
Function value	Tag_Result	0001 0101 0100 0001 1110 0001 1111 1110

The content of operand "Tag_Value" is rotated five bit places to the left. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 231)

Entering SCL instructions (Page 7328)

Editing SCL instructions (Page 7358)

Basics of SCL (Page 7293)

Additional instructions

SET: Set bit array

Description

You use the "Set bit array" instruction to set the signal state of the bits in a specified area to "1". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. At the N parameter of the instruction, you specify the number of bits to be set in the specified area. When the N parameter has the value "0", calling the instruction has no effect.

Note

Parameter N

You can also interconnect an element of a data block at parameter N.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be set

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
SET(S_BIT := P#M0.0,
    N := 10);
```

In the example, the instruction sets 10 bits in the range from M0.0 to M1.1.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SETP: Set a bit array in the I/O area (S7-400)

Description

The instruction "Set a bit array in the I/O area" sets the bits of an I/O area to the signal state "1". At the same time the corresponding bits in the process image outputs are also set parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be set. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output at the RET_VAL parameter.

The I/O area is set byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output. This can lead to unintended reactions such as starting a motor or turning off a cooling system.

  WARNING

  Unintended behavior

  The setting of the bits at the value of the relevant bit in the process image output can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are set to the signal state "1".

Calling the "Set a bit array in the I/O area instruction has no effect if the N parameter returns the value "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
N	Input	INT	I, Q, M, D, L	Number of bits to be set
SA	Output	POINTER	P	Pointer to the first bit to be set
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For additional information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Error" := SETP(N := 8,
                    SA => P#10.0);
```

In the example, the instruction sets 8 bits in the I/O range corresponding to the outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

SETI: Set byte array

Description

The "Set byte array" instruction sets the signal state of the bits in a specified area of bytes to "1". You define the start point of the array on the S_BYTE. You define the size of the array by specifying the number of bits to be set on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer on the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output (Q) is reset to "0".

Calling the "Set byte array" instruction has no effect if one of the following conditions applies:

- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P	Points to the first byte of the array whose bits will be set For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P	Size of the array to be set The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

SCL

```
SETI(S_BYTE := byte_array[100],
     N := 16);
```

In the example, the instruction sets 2 bytes (16 bits) in the range from 100.0 to 101.7 of the byte_array.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

RESET: Reset bit array

Description

The "Reset bit array" instruction is used to reset the signal state of the bits in a specified range to "0". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. At the N parameter of the instruction, you specify the number of bits to be reset in the specified range. When the N parameter has the value "0", calling the instruction has no effect.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Points to the first bit of the range whose bits will be reset. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P	Number of bits to be reset.

Example

The following example shows how the instruction works:

SCL

```
RESET(S_BIT := %M0.0,
      N := 10);
```

In the example the instruction resets 10 bits in the range from M0.0 to M1.1.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

RESETP: Reset a bit array in the I/O area (S7-400)

Description

The "Reset a bit array in the I/O area" instruction is used to reset the bits of an I/O area to signal state "0". At the same time the corresponding bits in the process image outputs are also reset parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be reset. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output at the RET_VAL parameter.

The I/O area is reset byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

  WARNING

  Unintended behavior

  The setting of the bits at the value of the relevant bit in the process image output can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are reset to the signal state "0".

Calling the "Reset a bit array in the I/O area" instruction has no effect if the N parameter returns the value "0".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
N	Input	INT	I, Q, M, D, L	Number of bits to be reset.
SA	Output	POINTER	P	Pointer to the first bit to be reset.
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For additional information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Error" := RESETP(N := 8,
                      SA => P#10.0);
```

In the example, the instruction resets 8 bits in the I/O range corresponding to the outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

RESETI: Reset byte array

Description

The "Reset byte array" instruction is used to reset the signal state of the bits in a specified range of bytes to "0". You use the pointer on the S_BYTE parameter to define the starting point of the range. You define the size of the range by specifying the number of bits to be reset on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer on

the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output (Q) is reset to "0".

Calling the "Reset byte array" instruction has no effect if one of the following conditions applies:

- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P	Points to the first byte of the range whose bits will be reset. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P	Size of the array whose bits are to be reset. The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

SCL

```
RESETI(S_BYTE := byte_array[100],
       N := 16);
```

In the example, the instruction resets 2 bytes (16 bits) in the range from 100.0 to 101.7 of the byte_array.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

REPL_VAL: Enter substitute value

Description

Use the "Enter substitute value" instruction to enter a substitute value in accumulator 1 (ACCU 1) from a synchronous error organization block.

Note

You can only call the "Enter substitute value" instruction in a synchronous error OB (OB 121, OB 122).

Use the "Enter substitute value" instruction when you can no longer read any value from a module. The OB 122 is started after each access to this module. By calling the "Enter substitute value" instruction you can have the substitute value loaded in accumulator 1. In this case the program will then be processed with this substitute value. You can take the required information for the selection of the substitute value from the local variable of the OB 122.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
VAL	Input	DWORD	I, Q, M, D, L	Substitute value
Function value (RET_VAL)		INT	I, Q, M, D, L	Error information

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Description
0000	No error
8080	The instruction was not called from a synchronous error OB.
8xyy	For additional information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Evaluating errors with output parameter RET_VAL (Page 162)

Basics of SCL (Page 7293)

DRUM: Implement sequencer

Description

You can use the "Implement sequencer" instruction to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). When a bit of the configurable mask has a signal state of "1", the OUT_VAL value sets/resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L	The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L	Number of the last programmed step.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P	Current counter value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer
PrevTime	Static	TIME	I, Q, M, D, L	Previous system time
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L	Preset counter value for each step [1 to 16] where one clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

SCL

```
"DRUM_DB"(RESET := "Tag_Reset"
          JOG := "Tag_Input_Jog"
          DRUM_EN := "Tag_Input_DrumEN"
          LST_STEP := "Tag_Number_LastStep"
          EVENT1 := "MyTag_Event_1"
          EVENT2 := "MyTag_Event_2"
          EVENT3 := "MyTag_Event_3"
          EVENT4 := "MyTag_Event_4"
          EVENT5 := "MyTag_Event_5"
          EVENT6 := "MyTag_Event_6"
          EVENT7 := "MyTag_Event_7"
          EVENT8 := "MyTag_Event_8"
          EVENT9 := "MyTag_Event_9"
          EVENT10 := "MyTag_Event_10"
          EVENT11 := "MyTag_Event_11"
          EVENT12 := "MyTag_Event_12"
          EVENT13 := "MyTag_Event_13"
          EVENT14 := "MyTag_Event_14"
          EVENT15 := "MyTag_Event_15"
```

SCL

```
EVENT16 := "MyTag_Event_16"
OUT1 => "MyTag_Output_1"
OUT2 => "MyTag_Output_2"
OUT3 => "MyTag_Output_3"
OUT4 => "MyTag_Output_4"
OUT5 => "MyTag_Output_5"
OUT6 => "MyTag_Output_6"
OUT7 => "MyTag_Output_7"
OUT8 => "MyTag_Output_8"
OUT9 => "MyTag_Output_9"
OUT10 => "MyTag_Output_10"
OUT11 => "MyTag_Output_11"
OUT12 => "MyTag_Output_12"
OUT13 => "MyTag_Output_13"
OUT14 => "MyTag_Output_14"
OUT15 => "MyTag_Output_15"
OUT16 => "MyTag_Output_16"
Q => "Tag_Output_Q"
OUT_WORD => "Tag_OutputWord"
ERR_CODE => "Tag_ErrorCode");
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE

Parameters	Address	Value
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameters	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

DRUM_X: Implement sequencer

Description

You use the "Implement sequencer" instruction to implement a sequencer with a maximum of 16 steps. You specify the number of the first step on the DSP parameter. You specify the number of the last step on the LST_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT_WORD (in which the output bits are collected together) are written. An output bit is assigned either the corresponding bit of the OUT_VAL array that you specified or the output bit of the preceding step. The value assigned depends on the value of the mask bit on the S_MASK parameter.

Note

The default of the mask bits is 0. If you want to change the current setting of one or more mask bits, make this change in the instance data block.

The "Implement sequencer" instruction advances to the next step when there is a rising signal edge on the JOG parameter. If the instruction is already in the last step, the tags Q and EOD are set when the JOG parameter has a rising edge. In this case the DCC parameter is set to the value "0". The instruction remains in the last step until the RESET parameter is set to the value "1".

You can also assign parameters so that switching to the next step is time dependent. To do this, the DRUM_EN parameter has to be set to the value "1". The transition to the next step is made when the event bit (EVENTi) belonging to the current step is set and the time specified for the current step has expired. The time for a step is the product of the (DTBP) time base and the time factor valid for the current step (S_PRESET).

Note

The execution time (DCC) remaining in the current step is reduced only when the corresponding event bit (EVENTi) is set.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

If the value "1" is set in the RESET parameter when the instruction is called, the sequencer goes to the step whose number is specified in the DSP parameter.

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1".

You can also move the next step at any time via the value at the JOG parameter.

When the instruction is called for the first time, you must set the value "1" on the RESET parameter.

When the sequencer is in the last step and the execution time specified for this step has expired, the parameters Q and EOD are set. The instruction remains in the last step until the RESET parameter changes to the value "1".

The instruction is executed only in the STARTUP and RUN modes.

The operating system resets the "Implement sequencer" instruction during a cold restart. During a warm restart, the instruction is not reset. If you want the "Implement sequencer" instruction to be initialized after a warm restart, you must call the instruction in OB 100 with the value "1" in the RESET parameter.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" resets the sequencer. When the instruction is called for the first time, the RESET parameter must be set to "1".
JOG	Input	BOOL	I, Q, M, D, L	A positive signal edge (compared to the last call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you assigned to the DRUM_EN parameter.
DRUM_EN	Input	BOOL	I, Q, M, D, L	Control parameter that specifies whether time-dependent switching to the next step is possible. If the value is "1" a time-dependent switching to the next step is possible.
LST_STEP	Input	BYTE	I, Q, M, D, L	Number of the last step Values from 1 to 16 are permitted.
EVENTi, 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit number i (belongs to step i)
OUTj, 0 ≤ j ≤ 15	Output	BOOL	I, Q, M, D, L	Output bit number j (identical to the bit number j of OUT_WORD)
Q	Output	BOOL	I, Q, M, D, L	Status parameter that indicates whether the specified execution time of the last step has expired.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Output bits collected together in a tag
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information

Parameters	Declaration	Data type	Memory area	Description
JOG_HIS	Static	BOOL	I, Q, M, D, L	Not relevant to user (input parameter JOG of the preceding call)
EOD	Static	BOOL	I, Q, M, D, L	Identical with output parameter Q
DSP	Static	BYTE	I, Q, M, D, L, P	Number of the first step Values from 1 to 16 are permitted.
DSC	Static	BYTE	I, Q, M, D, L, P	Number of the current step
DCC	Static	DWORD	I, Q, M, D, L, P	The execution time still remaining in the current step in milliseconds (ms) (only relevant if DRUM_EN = 1 and the corresponding event bit = 1)
DTBP	Static	WORD	I, Q, M, D, L, P	The time base valid for all steps in milliseconds (ms)
PREV_TIME	Static	DWORD	I, Q, M, D, L	(Not relevant for the user: system time of the previous call)
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L	One-dimensional array with the time factors for each step. Recommended selection of the indices: [1 to 16] In this case, S_PRESET [x] has the time factor of step x.

Parameters	Declaration	Data type	Memory area	Description
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the values to be output in each step if they have not been masked out using S_MASK. Recommended selection of the indices: [1 to 16, 0 to 15] In this case, OUT_VAL [x, y] has the value assigned to output bit OUTy in step x.
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the mask bits for each step. Recommended selection of the indices: [1 to 16, 0 to 15] In this case, S_MASK [x, y] contains the mask bit for the "y"the value to be output in step x. Meaning of the mask bits: • 0: The corresponding value of the previous step is assigned to the relevant output bit. • 1: The corresponding value from OUT_VAL is assigned to the relevant output bit.

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE* (W#16#...)	Explanation
0000	No error
8081	Invalid value at parameter LST_STEP
8082	Invalid value on parameter DSC
8083	Invalid value on parameter DSP
8084	The product DCC = DTBP * S_PRESET[DSC] exceeds the value 2**31-1 (approximately 24.86 days).
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT0 to OUT15) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

SCL

```
"DRUM_DB"(RESET := "Tag_Reset"
          JOG := "Tag_Input_Jog"
          DRUM_EN := "Tag_Input_DrumEN"
          LST_STEP := "Tag_Number_LastStep"
          EVENT1 := "MyTag_Event_1"
          EVENT2 := "MyTag_Event_2"
          EVENT3 := "MyTag_Event_3"
          EVENT4 := "MyTag_Event_4"
          EVENT5 := "MyTag_Event_5"
          EVENT6 := "MyTag_Event_6"
          EVENT7 := "MyTag_Event_7"
          EVENT8 := "MyTag_Event_8"
          EVENT9 := "MyTag_Event_9"
          EVENT10 := "MyTag_Event_10"
          EVENT11 := "MyTag_Event_11"
          EVENT12 := "MyTag_Event_12"
          EVENT13 := "MyTag_Event_13"
          EVENT14 := "MyTag_Event_14"
          EVENT15 := "MyTag_Event_15"
          EVENT16 := "MyTag_Event_16"
          OUT0 => "MyTag_Output_0"
          OUT1 => "MyTag_Output_1"
          OUT2 => "MyTag_Output_2"
          OUT3 => "MyTag_Output_3"
          OUT4 => "MyTag_Output_4"
          OUT5 => "MyTag_Output_5"
          OUT6 => "MyTag_Output_6"
          OUT7 => "MyTag_Output_7"
          OUT8 => "MyTag_Output_8"
          OUT9 => "MyTag_Output_9"
          OUT10 => "MyTag_Output_10"
          OUT11 => "MyTag_Output_11"
          OUT12 => "MyTag_Output_12"
          OUT13 => "MyTag_Output_13"
          OUT14 => "MyTag_Output_14"
```

SCL

```
OUT15 => "MyTag_Output_15"
Q => "Tag_Output_Q"
OUT_WORD => "Tag_OutputWord"
ERR_CODE => "Tag_ErrorCode");
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_X_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE

Parameters	Address	Value
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameters	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT0	MyTag_Output_0	M4.0	TRUE
OUT1	MyTag_Output_1	M4.1	TRUE
OUT2	MyTag_Output_2	M4.2	TRUE
OUT3	MyTag_Output_3	M4.3	TRUE
OUT4	MyTag_Output_4	M4.4	TRUE
OUT5	MyTag_Output_5	M4.5	TRUE
OUT6	MyTag_Output_6	M4.6	TRUE
OUT7	MyTag_Output_7	M4.7	TRUE
OUT8	MyTag_Output_8	M5.0	TRUE
OUT9	MyTag_Output_9	M5.1	TRUE
OUT10	MyTag_Output_10	M5.2	TRUE
OUT11	MyTag_Output_11	M5.3	TRUE
OUT12	MyTag_Output_12	M5.4	TRUE
OUT13	MyTag_Output_13	M5.5	TRUE
OUT14	MyTag_Output_14	M5.6	TRUE
OUT15	MyTag_Output_15	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Address	Value
OUT0	MyTag_Output_0	M4.0	TRUE
OUT1	MyTag_Output_1	M4.1	FALSE
OUT2	MyTag_Output_2	M4.2	FALSE
OUT3	MyTag_Output_3	M4.3	FALSE
OUT4	MyTag_Output_4	M4.4	FALSE
OUT5	MyTag_Output_5	M4.5	TRUE
OUT6	MyTag_Output_6	M4.6	FALSE
OUT7	MyTag_Output_7	M4.7	FALSE
OUT8	MyTag_Output_8	M5.0	TRUE
OUT9	MyTag_Output_9	M5.1	TRUE
OUT10	MyTag_Output_10	M5.2	FALSE
OUT11	MyTag_Output_11	M5.3	FALSE
OUT12	MyTag_Output_12	M5.4	FALSE
OUT13	MyTag_Output_13	M5.5	FALSE
OUT14	MyTag_Output_14	M5.6	TRUE
OUT15	MyTag_Output_15	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE

Parameters	Operand	Address	Value
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are saved in the "DRUM_X_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

DCAT: Discrete control-timer alarm

Description

You can use the "Discrete control-timer alarm" to count the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET, OA and CA are influenced as follows:
  - The Q and CMD_HIS parameters are set to "1".
  - The ET, OA, and CA parameters are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET, OA, CA and CMD_HIS are reset to "0".

- When the signal state of the parameters CMD and CMD_HIS is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state at the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- When the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where one clock pulse = 1 ms.

Parameters	Declaration	Data type	Memory area	Description
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
CMD_HIS	Static	BOOL	I, Q, M, D, L	CMD history bit

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The CMD_HIS parameter of the instance data block is set to the signal state "1" and the ET parameter is reset to "0".

Note

You can initialize static parameters in the data block.

SCL

```
"DCAT_DB"(CMD := "Tag_Input_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          Q => "Tag_Output_Q",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA");
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

MCAT: Motor control-timer alarm

Description

The "Motor control-timer alarm" instruction is used to accumulate the time from the point at which one of the command inputs (opening or closing) is switched on. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop

Input parameters								Output parameters								
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped
Legend:																
INC	Add the time difference (ms) since the last processing of the FB to ET															
PT	PT is set to the same value as ET															
X	Cannot be used															
<PT	ET < PT															
>=PT	ET >= PT															
When the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to the signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore not possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case: OO = FALSE CO = FALSE OA = FALSE CA = FALSE ET = PT Q = TRUE																

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	A signal state of "0" indicates an error condition.

Parameters	Declaration	Data type	Memory area	Description
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
O_HIS	Static	BOOL	I, Q, M, D, L	"Open" history bit
C_HIS	Static	BOOL	I, Q, M, D, L	"Close" history bit

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

SCL

```
"MCAT_DB"(O_CMD := "Tag_Iput_O_CMD",
          C_CMD := "Tag_Input_C_CMD",
          S_CMD := "Tag_Input_S_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          OO => "Tag_OutputOpen",
          CO => "Tag_OutputClosed",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA",
          Q => "Tag_Output_Q");
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE

Parameters	Operand	Value
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameters	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

IMC: Compare input bits with the bits of a mask

Description

You can use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P	The step number of the mask used for the comparison.

Parameters	Declaration	Data type	Memory area	Description
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF BOOL	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

SMC: Compare scan matrix

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.

Parameters	Declaration	Data type	Memory area	Description
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

LEAD_LAG: Lead and lag algorithm

Description

You can use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the instruction "Lead and lag algorithm" in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of output OUT so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P	The input value of the current sample time (cycle time) to be processed. Constants can also be specified at the IN parameter.
SAMPLE_T	Input	INT	I, Q, M, D, L, P	Sample time Constants can also be specified at the SAMPLE_T parameter.
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P	Previous output

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

SCL

```
"LEAD_LAG_DB"(IN := "Tag_Input",
              SAMPLE_T := "Tag_Input_SAMPLE_T",
              OUT => "Tag_Output_Result",
              ERR_CODE => "Tag_ErrorCode");
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameters	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameters	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameters	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

TONR_X: Time accumulator

Description

You use the "Time accumulator" instruction to accumulate time values within a period which you set using the PV input. The instruction is executed when the signal state on the TMR_EN start input changes from "0" to "1". Simultaneously, the timer runs for the duration set at the PV input. While the instruction is being executed, it adds the time value of the DELTA_T input to the time value of the InOut ET.

The following conditions must be met:

- The TMR_EN start input has the signal state "1".
- The time value at InOut ET is less than the time value at the PV parameter.
- The RESET input has the signal state "0".

The instruction is interrupted when the signal state on the TMR_EN start input changes from "1" to "0". As soon as the signal state changes back to "1", the instruction resumes execution.

The instruction is ended when the time value at InOut ET has reached the time value at the PV input. The Q output then has the signal state "1".

As long as the Q output has the signal state "1", the time value at InOut ET remains unchanged. The time value and the Q output are not reset to "0" until the RESET input is set to signal state "1".

Since the "Time accumulator" instruction uses the cycle time (DELTA_T) of the last cycle of the organization block (OB), the instruction may only be called in cyclic organization blocks.

Note

You have to move the cycle time of the organization block from the startup information to the global tag of the DELTA_T parameter.

Syntax

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
TMR_EN (Timer Enable)	Input	BOOL	I, Q, M, D, L	Start input Starts the instruction and the expiration of the preset time duration at the PV input
RESET	Input	BOOL	I, Q, M, D, L	Reset input
PV (Preset Value)	Input	DINT	I, Q, M, D, L, P or constant	Maximum duration of time recording
DELTA_T	Input	INT	I, Q, M, D, L or constant	Cycle time of organization block of the previous cycle
Q	Output	BOOL	I, Q, M, D, L	Output which is set when the preset time value is reached

Parameters	Declaration	Data type	Memory area	Description
ET (Elapsed Time)	InOut	DINT	I, Q, M, D, L	Accumulated time value (Currently elapsed time value + time value of the DELTA_T input)
Function value		VOID		Empty function value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:

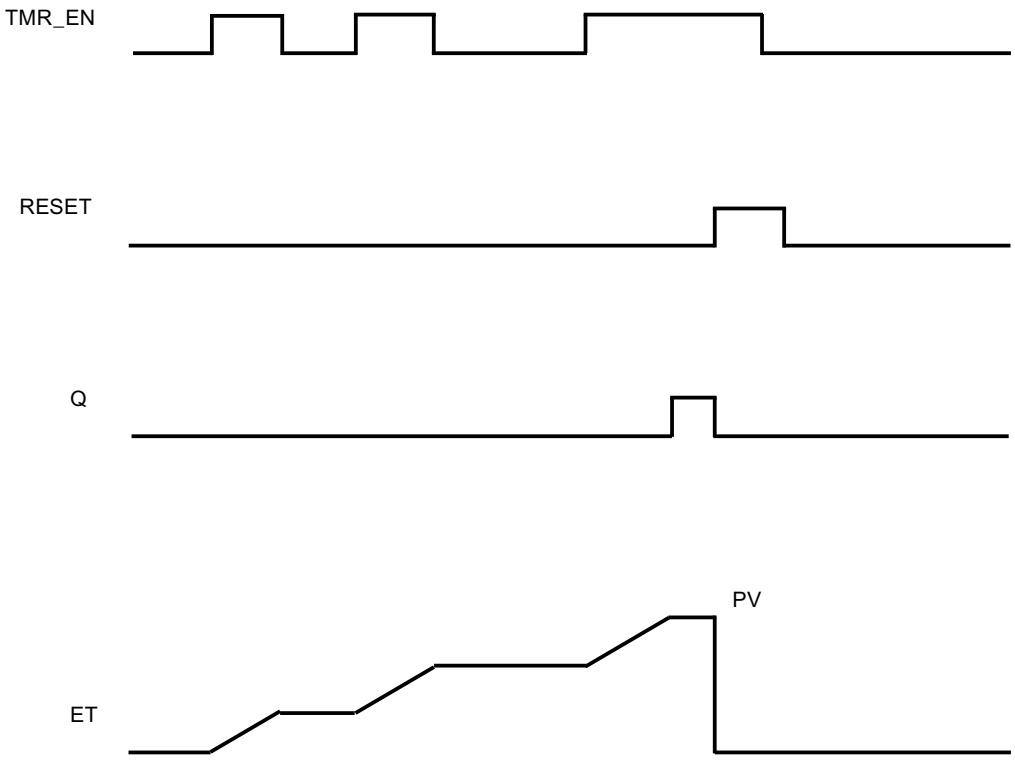

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := TONR_X(TMR_EN := "Tag_Start",
                       RESET := "Tag_Reset",
                       PV := 10,
                       DELTA_T := 2,
                       Q => "Tag_Output",
                       ET := "Tag_Elapsed_Time");
```

Each time the signal state changes from "0" to "1" at the TMR_EN input, the cycle time "2" at the DELTA_T input is added to the current time value of InOut ET until the maximum time value "10" of the PV input is reached.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

WSR: Save data to shift register

Description

The "Save data to shift register" instruction is used to write a bit from a specified source to a shift register. Each element of the register is shifted into the next address. The data contained in the last address of the shift register is lost after the shift.

Each time the instruction is executed with the value "0" on the RESET parameter, new data are read from the source (S_DATA) and shifted to the start address (START) of the shift register. The LEN parameter specifies the number of elements to be shifted. You specify the data type of the elements on the E_TYPE parameter. The instruction is not executed if an invalid data type is specified on the E_TYPE parameter.

The register is deleted if the RESET parameter is set to "1" when the instruction is executed. The Q parameter is set to the signal state "1" when the shift register is filled with zeros or deleted.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_DATA	Input	POINTER	I, Q, M, D	Points to the source data element to be shifted to the start address (START). For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
START	Input	POINTER	I, Q, M, D	Points to the start address of the register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
LEN	Input	WORD	I, Q, M, D, L, P	Number of elements to be shifted.

Parameters	Declaration	Data type	Memory area	Description
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the elements to be shifted. The following data types are permitted: • W#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
Q	Output	BOOL	I, Q, M, D, L	Is set to "0" if RESET parameter is active (1) or all elements to be shifted have the value "0".
Function value		VOID	I, Q, M, D, L	Empty function value

Example

In the following example four elements of the data type WORD are shifted starting from the address specified on the START parameter. When the first element of the register has been shifted to the next address, the first address is filled with the data that the pointer references on the S_DATA parameter. The value of the last element in the register is lost in the process.

SCL

```
WSR(RESET := "Tag_Input_Reset",
    S_DATA := "Tag_Input_DataSource",
    START := "Tag_Input_StartAddress",
    LEN := "Tag_Input_Length",
    E_TYPE := "Tag_Input_ElementType",
    Q => "Tag_Output_Q");
```

No function value is transferred.

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	DB2.DBX6.0 DW0 = W#16#1111 DW2 = W#16#2222 DW4 = W#16#3333 DW6 = W#16#4444
START	Tag_Input_StartAddress	DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04

Parameters	Operand	Value
E_TYPE	Tag_Input_ElementType	W#16#04
Q	Tag_Output_Q	1

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	DB2.DBX6.0 DW0 = W#16#1234 DW2 = W#16#1111 DW4 = W#16#2222 DW6 = W#16#3333
START	Tag_Input_StartAddress	DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SHRB: Shift bit to shift register

Description

The "Shift bit to shift register" instruction is used to write a bit from a specified source (DATA) to a shift register. Each time the instruction is executed with a value "0" on the RESET parameter, new data are read from the source and shifted to the start address (S_BIT) of the shift register. All following bits are shifted by one. The bit contained at the last address (S_BIT + N) is lost after the shift. If the RESET parameter is set to "1" during the processing of the instruction, the addresses in the shift register are reset to "0" and not shifted.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
DATA	Input	BOOL	I, Q, M, D, L	Source bit
RESET	Input	BOOL	I, Q, M, D, L	Reset input

Parameters	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Points to the start bit of the shift register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	WORD	I, Q, M, D, L, P	Number of bits to be shifted.
Function value		VOID	I, Q, M, D, L	Empty function value

Example

In the following example the N parameter is set to the value 14 ("E" in hexadecimal). As a result 14 bits starting from the address specified on the S_BIT parameter are shifted. After the bits have been shifted, the first address is filled with the data from the source on the DATA parameter. The last bit value is lost.

SCL

```
SHRB(DATA := "Tag_Input_Data",
     RESET := "Tag_Input_Reset",
     S_BIT := "Tag_Input_BitSource",
     N := "Tag_Input_Number");
```

No function value is transferred.

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
RESET	Tag_Input_Reset	0
DATA	Tag_Input_Data	0
S_BIT	Tag_Input_BitSource	DB1.DBX1.4
N	Tag_Input_Number	B#16#0E

Before processing the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	0	1	1	1	0	1	0

2.7	...					...	2.0
0	0	1	1	1	0	1	0

3.7	...					...	3.0
0	0	1	1	1	0	1	0

After processing

After the processing of the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	1	1	0	1	0	1	0

2.7	...					...	2.0
0	1	1	1	0	1	0	0

3.7	...					...	3.0
0	0	1	1	1	0	0	0

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

SEG: Create bit pattern for seven-segment display

Description

The "Create bit pattern for seven-segment display" instruction is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a 7-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	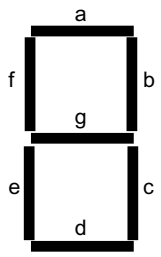
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display
Function value		VOID	I, Q, M, D, L	Empty function value

Example

The following example shows how the instruction works:

SCL

```
SEG(IN := "Tag_Input",
    OUT => "Tag_Output");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW16#065B4F66	00000110 01011011 01001111 01100110 Display: 1234

No function value is transferred.

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

BCDCPL: Create tens complement

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the operand. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	DWORD	I, Q, M, D, L, P	7-digit BCD number
Function value		DWORD	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := BCDCPL("Tag_Input");
```

The following table shows how the instruction functions using specific values:

Operand	Value*
Tag_Input	DW#16#01234567
Tag_Result	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Switching display formats in the program status (Page 7698)

Basics of SCL (Page 7293)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1".

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	DWORD	I, Q, M, D, L, P	Operand whose set bits are counted
Function value		INT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := BITSUM("Tag_Input");
```

The following table shows how the instruction functions using specific values:

Operand	Value*
Tag_Input	DW#16#12345678
Tag_Result	W#16#000D (13 bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 231)

Basics of SCL (Page 7293)

4.2.2.5 GRAPH

GRAPH sequencer

Step and transition

Description

You can use the "Step and transition" structure element to insert a step and transition simultaneously in your sequencer.

See also:

Step (Page 5045)

Transition (Page 5046)

See also

Step (Page 5045)

Transition (Page 5046)

Sequence end (Page 5047)

Jump to step (Page 5047)

Alternative branch (Page 5048)

Simultaneous branch (Page 5048)

Close branch (Page 5049)

Steps and transitions (Page 7375)

Step

Description

Steps are used to subdivide complex automation tasks into straightforward subtasks that can then be accomplished using actions. The individual steps are organized in sequencers so that every step can be executed in the specified order in the program execution. Each step must be assigned a unique name and number.

For the execution to actually be performed, the step must be activated by one of the following conditions:

- The step has been defined as an initial step.
- The transition of the previous step has been fulfilled.
- The step has been called by an event-dependent action.

Once all actions have been executed, the step becomes inactive again.

Steps in which no actions are programmed are called empty steps. Such an empty step reacts like an active step and the subsequent transition is always fulfilled.

See also

Basics on steps (Page 7375)

Step and transition (Page 5045)

Transition (Page 5046)

Sequence end (Page 5047)

Jump to step (Page 5047)

Alternative branch (Page 5048)

Simultaneous branch (Page 5048)

Close branch (Page 5049)

Transition

Description

Transitions are located between the steps and contain the conditions for advancing from one step to the next (step-enabling). When the step-enabling conditions of a transition are fulfilled, the next step becomes active and its actions are executed. You can program the conditions of a transition either in LAD or FBD.

See also

Basics on transitions (Page 7377)

Step and transition (Page 5045)

Step (Page 5045)

Sequence end (Page 5047)

Jump to step (Page 5047)

Alternative branch (Page 5048)

Simultaneous branch (Page 5048)

Close branch (Page 5049)

Sequence end

Description

You can use the "Sequence end" structure element to stop a sequencer or a branch. In a simultaneous branch, a transition must precede the sequence end.

Note

If all branches of a sequencer are completed with a sequence end, the sequencer can be restarted by the "INIT_SQ" parameter or using the "Initialize" button in the "Sequence control" pane of the "Testing" task card.

See also

Elements of a sequencer (Page 7372)

Step and transition (Page 5045)

Step (Page 5045)

Transition (Page 5046)

Jump to step (Page 5047)

Alternative branch (Page 5048)

Simultaneous branch (Page 5048)

Close branch (Page 5049)

Jump to step

Description

You can use a jump to continue the program execution at any step within the GRAPH function block. Jumps can be inserted at the end of the main branch or an alternative branch to enable cyclic processing of the sequencer. The jump and jump destination are represented in the sequencer as arrows, in which case the return transition is specified for the jump destination.

See also

Elements of a sequencer (Page 7372)

Step and transition (Page 5045)

Step (Page 5045)

Transition (Page 5046)

Sequence end (Page 5047)

Alternative branch (Page 5048)

Simultaneous branch (Page 5048)

Close branch (Page 5049)

Alternative branch

Description

You can use alternative branches to program OR branches. This means that you insert branches that start with a transition after a step. The corresponding branch is executed depending on which transition is satisfied first. If several transitions are satisfied at the same time, the transition farthest to the left has the highest priority and the corresponding branch is executed. Alternative branches end once again with a transition.

You can program up to 125 alternative branches in a sequencer.

See also

Elements of a sequencer (Page 7372)

Step and transition (Page 5045)

Step (Page 5045)

Transition (Page 5046)

Sequence end (Page 5047)

Jump to step (Page 5047)

Simultaneous branch (Page 5048)

Close branch (Page 5049)

Simultaneous branch

Description

You can use simultaneous branches to program AND branches. This means that you use one transition to activate multiple steps, the actions of which are then executed. Simultaneous branches always start with a step.

The subsequent transitions of the simultaneous branches are located on the main branch, in which case the various simultaneous branches can be connected to various points of the main branch. Note that branches that you join together in a transition only advance to the next step when all branches are completely executed.

You can program up to 249 simultaneous branches in a sequencer.

See also

Elements of a sequencer (Page 7372)

Step and transition (Page 5045)

Step (Page 5045)
Transition (Page 5046)
Sequence end (Page 5047)
Jump to step (Page 5047)
Alternative branch (Page 5048)
Close branch (Page 5049)

Close branch

Description

You can use the "Close branch" element to close simultaneous and alternative branches to their parent branch. This is necessary if you do not want to complete the branch with a jump or a sequence end. In a simultaneous branch, you can insert "Close branch" only after a step.

See also

Elements of a sequencer (Page 7372)
Step and transition (Page 5045)
Step (Page 5045)
Transition (Page 5046)
Sequence end (Page 5047)
Jump to step (Page 5047)
Alternative branch (Page 5048)
Simultaneous branch (Page 5048)

GRAPH actions

Timer operations

TP: Generate pulse

Description

You use the "Generate pulse" instruction to set the Q output for a programmed duration. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The execution of the instruction requires a preceding logic operation. It can be placed within or at the end of the network. The programmed time PT begins when the instruction starts. The Q parameter is set for the time PT, regardless of the subsequent changes in the input signal. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the time duration PT is active.

You can query the current time value set in the ET parameter. The timer value starts at T#0s and ends when the value of time PT is reached. When time PT is reached and the signal state of the IN parameter is "0", the ET parameter is reset.

Each call of the "Generate pulse" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TP (for example, "TP_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate pulse" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET parameter returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current timer value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse diagram of the "Generate pulse" instruction:

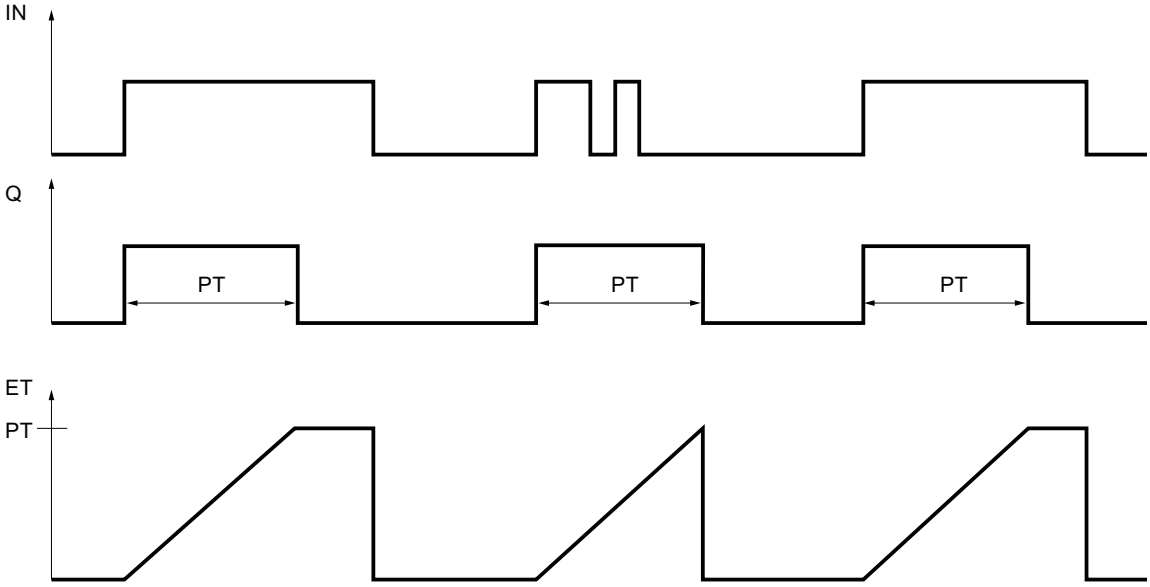

Example

The following example shows how the instruction works:

GRAPH

```
CALL TP TIME, "IEC_TP_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTIME" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

TON: Generate on-delay

Description

You use the "Generate on-delay" instruction to delay setting of the Q output by the programmed time PT. The instruction is started when the result of logic operation (RLO) in the IN input changes from "0" to "1" (positive signal edge). The execution of the instruction requires a preceding logic operation. It can be placed within or at the end of the network. The programmed time PT begins when the instruction starts. When time PT expires, the Q output has the signal state "1". The Q output remains set as long as the start input IN is still "1". When the signal state of the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of time PT is reached. The ET output is reset as soon as the signal state in the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TON (for example, "TON_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse diagram of the "Generate on-delay" instruction:

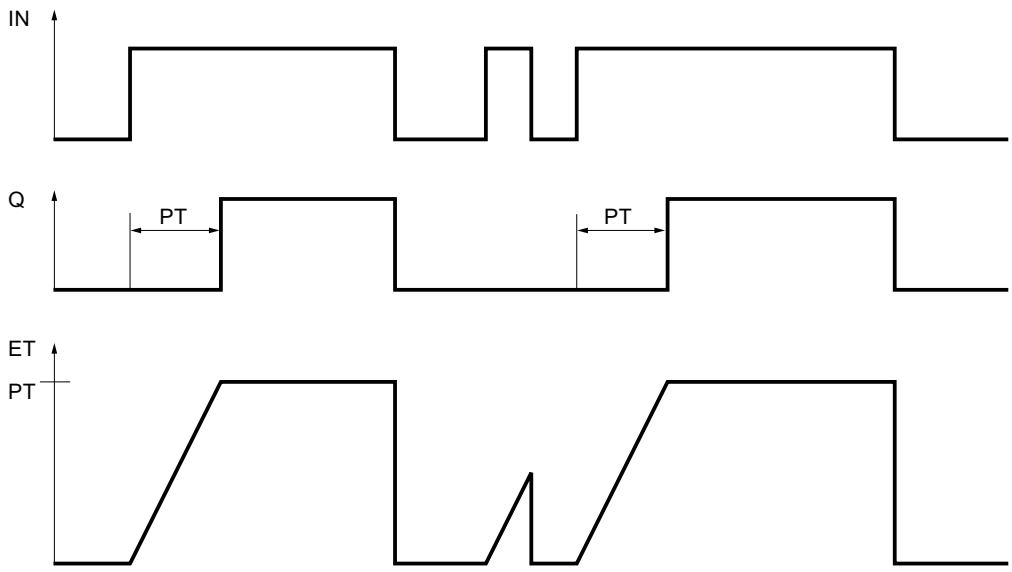

Example

The following example shows how the instruction works:

GRAPH

```
CALL TON TIME, "IEC_TON_DB"
      (IN := "Tag_Start"
       PT := "Tag_PresetTIME"
       Q => "Tag_Status"
       ET => "Tag_ElapsedTIME"
       )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. When the time expires, the "Tag_Status" operand is set to the signal state "1". The "Tag_Status" operand remains set to "1" as long as the "Tag_Start" operand has signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand. When the signal state of the "Tag_Start" operand changes from "1" to "0", the "Tag_Status" operand is reset.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

TOF: Generate off-delay

Description

You use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed time PT. The Q output is set when the result of logic operation (RLO) in the IN input changes from "0" to "1" (positive signal edge). The execution of the instruction requires a preceding logic operation. It can be placed within or at the end of the network. When the signal state of the IN input changes back to "0", the programmed time PT starts. The Q output remains set as long as time PT is running. When time PT expires, the Q output is reset. If the signal state of the IN input changes to "1" before time PT expires, the timer is reset. The signal state of the Q output continues to be "1".

The current time value can be queried in the ET output. The timer value starts at T#0s and ends when the value of time PT is reached. When time PT expires, the ET output remains set to the current value until the IN input changes back to "1". If the IN input changes to "1" before time PT has expired, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TOF (for example, "TOF_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Skipping the instruction

If the instruction is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Note

Update of the instruction data

The instruction data is updated only when the instruction is called. The data is not updated when the Q or ET outputs are accessed.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is reset when time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse diagram of the "Generate off-delay" instruction:

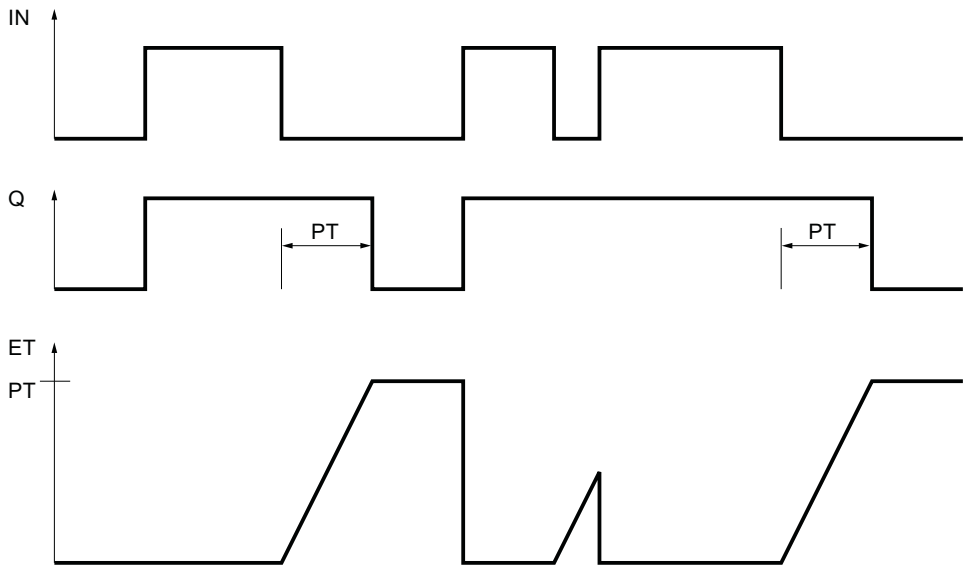

Example

The following example shows how the instruction works:

GRAPH

```
CALL TOF TIME, "IEC_TOF_DB"
      (IN := "Tag_Start"
       PT := "Tag_PresetTIME"
       Q => "Tag_Status"
       ET => "Tag_ElapsedTIME"
       )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Tag_Status" operand is set. When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set. When the time has expired, the "Tag_Status" operand is reset. The current time value is stored in the "Tag_ElapsedTIME" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

Counter operations

CTU: Count up

Description

You can use the "Count up" instruction to increment the value in the CV output. When the signal state of the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value in the CV output is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the CV output. When the high limit is reached, the signal state of the CU input no longer has an effect on the instruction.

You can query the counter status in the Q output. The signal state of the Q output is determined by the PV parameter. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value of the CV output is reset to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", the signal state of the CU input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the R parameter of the instruction in a startup OB. If instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTU INT,"IEC_CTU_DB"
      (CU := "Tag_Start"
       R := "Tag_ResetCOUNTER"
       PV := "Tag_PresetValue"
```

GRAPH

```
Q => "Tag_Status"
CV => "Tag_CounterValue"
)
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "Tag_Status" output. The "Tag_Status" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" operand. In all other cases, the "Tag_Status" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value of the CV output. When the signal state of the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV output is decremented by one. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state of the CD input no longer has an effect on the instruction.

You can query the counter status in the Q output. If the current counter value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value of the CV output is set to the value of the PV parameter when the signal state of the LD input changes to "1". As long as the LD input has signal state "1", the signal state of the CD input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTD (for example, "CTD_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case, the desired initial value for the CV parameter is specified in the PV parameter.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value to which the CV output is set with LD = 1.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTD INT, "IEC_CTD_DB"
     (CD := "Tag_Start"
      LD := "Tag_LoadPV"
      PV := "Tag_PresetValue"
      Q => "Tag_Status"
      CV => "Tag_CounterValue"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the instruction is executed and the value at the "Tag_CounterValue" output is decremented by one. With each additional positive signal edge, the counter value is decremented until the low limit of the specified data type (INT = -32768) is reached.

The "Tag_Status" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_Status" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the value of the CV output. When the signal state of the CU input changes from "0" to "1" (positive signal edge), the current counter value of the CV output is incremented by one. When the signal state of the CD input changes from "0" to "1" (positive signal edge), the current counter value of the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value of the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV output. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state of the LD input changes to "1", the counter value of the CV output is set to the value of the PV parameter. As long as the LD input has signal state "1", the signal state of the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state of the R input changes to "1". As long as the R input has signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can query the status of the up counter in the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0".

You can query the status of the down counter in the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a startup OB:

- When used as up counter, the value of the R parameter must be set to "1".
- When used as down counter, the value of the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the QU output is set / value to which the CV output is set when LD = 1.
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	INT	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTUD INT, "IEC_CTUD_DB"
     (CU := "Tag_StartCTU"
      CD := "Tag_StartCTD"
      R := "Tag_ResetCOUNTER"
      LD := "Tag_LoadPV"
      PV := "Tag_PresetValue"
      QU => "Tag_CounterStatusUP"
      QD => "Tag_CounterStatusDOWN"
      CV => "Tag_CounterValue"
      )
```

When the signal state of the "Tag_StartCTU" or "Tag_StartCTD" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "Tag_StartCTU" input, the current counter value is incremented by one and stored in the "Tag_CounterValue" output. When there is a positive signal edge at the "Tag_StartCTD" input, the counter value is decremented by one and stored in the "Tag_CounterValue" output. When there is a positive signal edge at the CU input, the counter value is incremented until it reaches the high limit of 32767. When there is a positive signal edge at the CD input, the counter value is decremented until it reaches the low limit of INT = -32768.

The "Tag_CounterStatusUP" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" input. In all other cases, the "Tag_CounterStatusUP" output has signal state "0".

The "Tag_CounterStatusDOWN" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_CounterStatusDOWN" output has signal state "0".

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

Math functions

NEG: Create twos complement

Description

You use the "Create twos complement" instruction to change the sign of the operand value. For example, if the value is positive, the negative equivalent of this value is output.

The value of the OUT output is invalid if the result of the instruction is outside the range permitted for the data type specified at the OUT output (only valid for integers).

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := NEG_REAL("Tag_InValue")
```

The instruction changes the sign of the value of the "TagIn_Value" operand and outputs the result in the "TagOut_Value" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

ABS: Form absolute value

Description

You use the "Form absolute value" instruction to calculate the absolute value of the operand value.

The value of the result is invalid if a floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ABS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
TagIn_Value	-6.234
TagOut_Value	6.234

The instruction calculates the absolute value of the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

MIN: Get minimum

Description

You use the "Get minimum" instruction to compare the values of the IN1, IN2, and IN3 inputs and write the lowest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is met:

- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the INT, DINT, and REAL data types are permitted for the input parameters, even when the IEC check is disabled. If you use the WORD or DWORD data type for the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Second input value
IN3	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Third input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MIN INT
      (IN1 := "TagIn_Value1"
       IN2 := "TagIn_Value2"
       IN3 := "TagIn_Value3"
       OUT => "Tag_Minimum"
       )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Minimum	12222

The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to the "Tag_Minimum" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

MAX: Get maximum

Description

You use the "Get maximum" instruction to compare the values of the IN1, IN2, and IN3 inputs and write the highest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is met:

- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the INT, DINT, and REAL data types are permitted for the input parameters, even when the IEC check is disabled. If you use the WORD or DWORD data type for the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Second input value
IN3	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Third input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MAX INT
     (IN1 := "TagIn_Value1"
      IN2 := "TagIn_Value2"
      IN3 := "TagIn_Value3"
      OUT => "Tag_Maximum"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12666
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to the "Tag_Maximum" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

LIMIT: Set limit value

Description

You use the "Set limit value" instruction to limit the value of the IN input to the values of the MN and MX inputs. If the value of the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not met and the IN input value is below the low limit MN, the OUT output is set to the value of the MN input. If the high limit MX is exceeded, the OUT output is set to the value of the MX input.

If the value of the MN input is greater than the value of the MX input, the result is undefined. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is met:

- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value of the MN input is greater than the value of the MX input.

Note

Only the INT, DINT, and REAL data types are permitted for the input parameters, even when the IEC check is disabled. If you use the WORD or DWORD data type for the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
MN	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Low limit
IN	Input	Integers, floating-point numbers	I, Q, M, D, L, P	Input value
MX	Input	Integers, floating-point numbers	I, Q, M, D, L, P	High limit
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL LIMIT INT
      (MN := "Tag_LowLimit"
```

GRAPH

```
IN := "Tag_InputValue"
MX := "Tag_HighLimit"
OUT => "Tag_Result"
)
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_LowLimit	12000
IN	Tag_InputValue	8000
MX	Tag_HighLimit	16000
OUT	Tag_Result	12000

The value of the "Tag_InputValue" operand is compared with the values of the "Tag_LowLimit" and "Tag_HighLimit" operands. Since the value of the "Tag_InputValue" operand is less than the low limit, the value of the "Tag_LowLimit" operand is copied to the "Tag_Result" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

SQR: Form square

Description

The "Form square" instruction is used to square the value of the floating-point number of the operand and write the result to the output.

If the operand contains an invalid floating-point number, the value of the result is invalid.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Square of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SQR("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	5.0
Tag_OutValue	25.0

The instruction squares the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

SQRT: Form square root

Description

Use the "Form square root" instruction to calculate the square root of the input value and save the result in the specified operand. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number. If the value of the operand is "0", then the result is also "0".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L	Square root of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SQRT("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	25.0
Tag_OutValue	5.0

The instruction calculates the square root of the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

LN: Form natural logarithm

Description

You use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e=2.718282) from the input value. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := LN("Tag_InValue")
```

The instruction calculates the natural logarithm of the value in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

EXP: Form exponential value

Description

You use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2,718282) and the value of the operand.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Exponential value of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := EXP("Tag_InValue")
```

The instruction calculates the exponent from base e and the value of the "Tag_InValue" operand and outputs the result (e^{IN}) in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the value of the operand is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P or constant	Size of angle in radians
<Result>	Output	REAL	I, Q, M, D, L, P	Sine of the specified angle

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SIN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+1.570796 (π/2)
Tag_OutValue	1.0

The instruction calculates the sine of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

COS: Form cosine value

Description

You use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the value of the operand is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P or constant	Size of angle in radians
<Result>	Output	REAL	I, Q, M, D, L, P	Cosine of the specified angle

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := COS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+1.570796 (π/2)
Tag_OutValue	0

The instruction calculates the cosine of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

TAN: Form tangent value

Description

You use the "Form tangent value" instruction to calculate the tangent of an angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the value of the operand is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P or constant	Size of angle in radians
<Result>	Output	REAL	I, Q, M, D, L, P	Tangent of the specified angle

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := TAN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+3.141593 (π)
Tag_OutValue	0

The instruction calculates the tangent of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

ASIN: Form arcsine value

Description

You use the "Form arcsine value" instruction to calculate the size of the angle that corresponds to the sine value specified in the operand. Only valid floating-point numbers within the range -1 to +1 can be specified for the operand. The calculated angle size can range in value from -π/2 to +π/2.

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P or constant	Sine value
<Result>	Output	REAL	I, Q, M, D, L, P	Size of angle in radians

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ASIN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1.0
Tag_OutValue	+1.570796 (π/2)

The instruction calculates the size of the angle corresponding to the sine value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

ACOS: Form arccosine value

Description

You use the "Form arccosine value" instruction to calculate the size of the angle that corresponds to the cosine value specified in the operand. Only valid floating-point numbers within the range -1 to +1 can be specified for the operand. The calculated angle size can range in value from 0 to +π.

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P or constant	Cosine value
<Result>	Output	REAL	I, Q, M, D, L, P	Size of angle in radians

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ACOS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0
Tag_OutValue	+1.570796 (π/2)

The instruction calculates the size of the angle corresponding to the cosine value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

ATAN: Form arctangent value

Description

You use the "Form arctangent value" instruction to calculate the size of the angle that corresponds to the tangent value specified in the operand. It is only permitted to specify valid floating-point numbers (or -NaN/+NaN) at the operand. The calculated angle size can range in value from -π/2 to +π/2.

The value at the OUT output is invalid if the value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Tangent value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ATAN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1.0
Tag_OutValue	+0.785398 (π/4)

The instruction calculates the size of the angle corresponding to the tangent value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Invalid floating-point numbers (Page 253)

Basic information on GRAPH (Page 7369)

Move operations

SWAP: Swap

Description

You use the "Swap" instruction to change the order of the bytes of the operand.

Depending on the data type used, you can swap all bytes in accumulator 1 or only the bytes in the right word of accumulator 1.

The following figure shows how the bytes of the operand of data type DWORD are swapped:

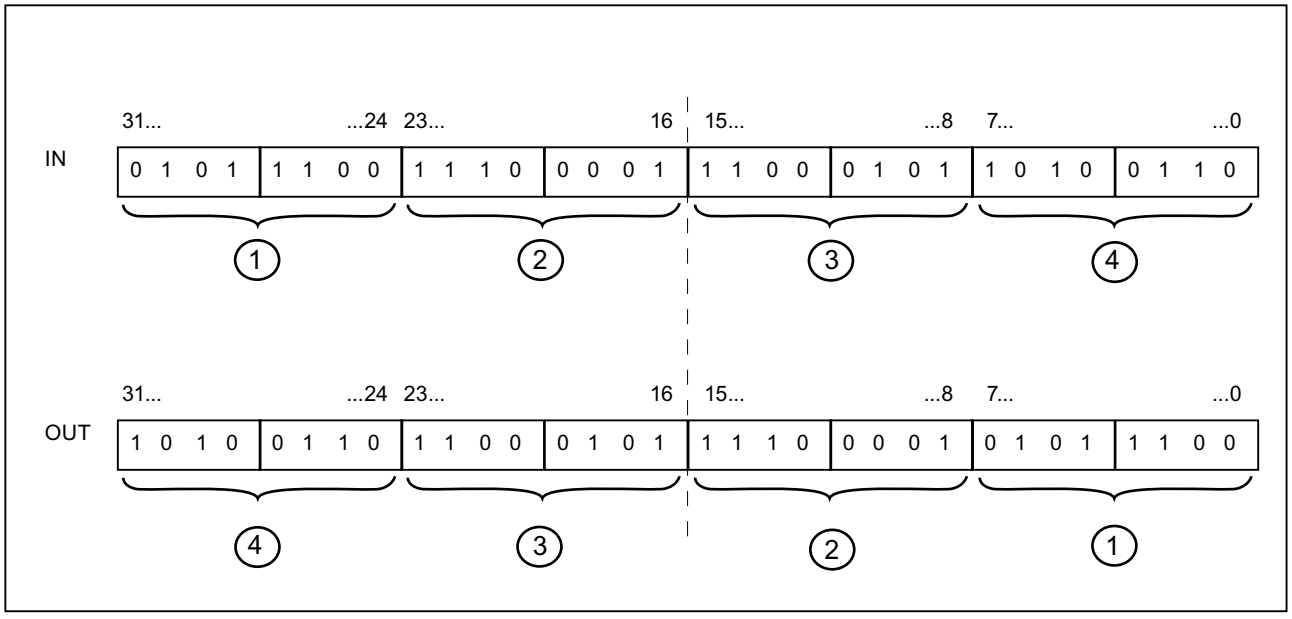

Order of bytes in the right word of accumulator 1

For data type WORD, you swap the bytes in the right word of accumulator 1.

The following table shows the contents of accumulator 1 before and after execution of the instruction:

Status	Bytes in accumulator 1			
Before execution	Value A	Value B	Value C	Value D
After execution	Value A	Value B	Value D	Value C

The result of the instruction is saved in the right word of accumulator 1. The bytes in the left word of accumulator 1 are not influenced by the instruction and remain unchanged.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	WORD, DWORD	I, Q, M, D, L or constant	Operand whose bytes are swapped
<Result>	Output	WORD, DWORD	I, Q, M, D, L	Result

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SWAP("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_InValue	0000	1111	0000	1111
Tag_OutValue	0000	1111	1111	0000

Swapping all bytes in accumulator 1:

For data type DWORD, you swap all bytes in accumulator 1.

The following table shows the contents of accumulator 1 before and after execution of the instruction:

Status	Bytes in accumulator 1			
Before execution	Value A	Value B	Value C	Value D
After execution	Value D	Value C	Value B	Value A

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_InValue	1111	0000	0000	1111	0000	0000	1111	1111
Tag_OutValue	1111	1111	0000	0000	0000	1111	1111	0000

The bytes in the "Tag_InValue" operand are swapped and stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

BLKMOV: Move block

Description

You use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:

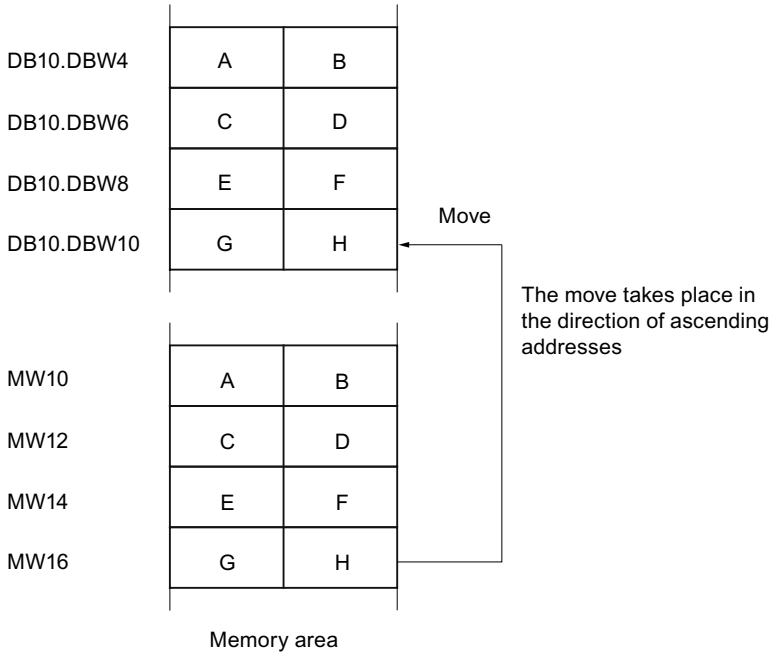

Consistency of source and destination data

Make sure that the source data remains unchanged during execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, BLKMOV is interrupted while copying from a DB that is not runtime-relevant, the execution of BLKMOV can no longer be nested.

Memory areas

You use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks that are not runtime-relevant

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the specified length of the block must be divisible by 8, otherwise it will not be possible to execute the instruction.

Note

If an actual source or destination area is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. The error code W#16#837F is output in the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H CPU):
  No data are transferred. The error code W#16#8122 or W#16#8323 is output in the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. The error code W#16#8122 or W#16#8323 is output in the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is of data type STRING, the characters that are actually contained in the character string will be moved. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Rules for moving data blocks that are not runtime-relevant

The source area can also be in a non-runtime-relevant data block in the load memory. Data blocks that are not runtime-relevant are indicated by the keyword UNLINKED.

If a non-runtime-relevant data block is copied to the work memory with the "Move block" instruction and loaded at the same time, for example, by the programming device, the execution of the instruction can be delayed for several milliseconds. This results in a longer OB cycle and may trip the cycle monitoring.

If a non-runtime-relevant data block is moved with the "Move block" instruction and the move operation is interrupted, the execution of the instruction can no longer be continued.

If the "Read from data block in the load memory" instruction is available on your CPU, you must use this instruction to read non-runtime-relevant data blocks in the load memory. If you use the "Move block" instruction, error W#16#8092 is output.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1]	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is moved (destination area).
1) The DSTBLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is only available in the load memory.
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL BLKMOV ANY
     (SRCBLK := P#M100.0 BYTE 10
      RET_VAL => "Tag_ErrorCode"
      DSTBLK => P#DB1.DBX0.0 BYTE 10
      )
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_ErrorCode" tag.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

UBLKMOV: Move block uninterruptible

Description

You use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during execution of the "Move block uninterruptible" instruction.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

Memory areas

You use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the specified length of the block must be divisible by 8, otherwise it will not be possible to execute the instruction.

You use the "Move block uninterruptible" instruction to move a maximum of 512 bytes. Note the CPU-specific restrictions for this.

Note

If an actual source or destination area is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. The error code W#16#837F is output in the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H CPU):
  No data are transferred. The error code W#16#8122 or W#16#8323 is output in the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. The error code W#16#8122 or W#16#8323 is output in the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of data type STRING. If only the source area is of data type STRING, the characters that are actually contained in the character string will be moved. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as the area length.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output [1]	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is moved (destination area).
1) The DSTBLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source or destination area is only available in the load memory.
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL UBLKMOV ANY
     (SRCBLK := P#M100.0 BYTE 10
      RET_VAL => "Tag_ErrorCode"
      DSTBLK => P#DB1.DBX0.0 BYTE 10
      )
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_ErrorCode" tag.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

FILL: Fill block

Description

You use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:

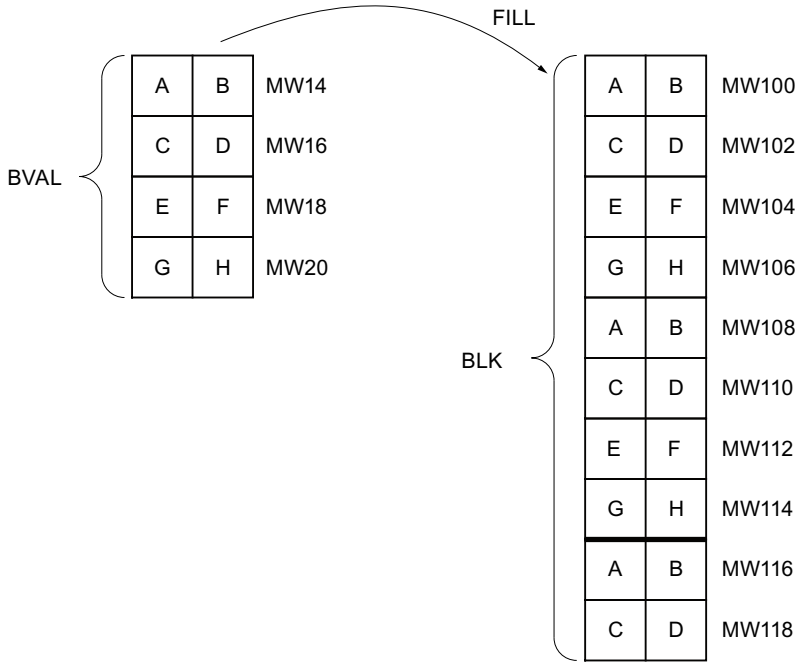

Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Make sure that the source data remain unchanged during execution of the "Fill block" instruction, since otherwise consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of data type BOOL, the specified length of the area must be divisible by 8; otherwise the instruction will not be executed.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Note

If an actual source or destination area is smaller than a destination or source area specified in the BVAL or BLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. The error code W#16#837F is output in the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H CPU):
  No data are transferred. The error code W#16#8122 or W#16#8323 is output in the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. The error code W#16#8122 or W#16#8323 is output in the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
BVAL	Input	ANY	I, Q, M, D, L, P	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
BLK	Output [1]	ANY	I, Q, M, D, L, P	Specification of the memory area that will be filled with the content of the source area.
1) The BLK parameter is declared as Output, but it has to be declared as InOut in the block interface.				

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or destination area is only available in the load memory.
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL FILL ANY
      (BVAL := P#M14.0 WORD 4
       RET_VAL => "Tag_ErrorCode"
       BLK => P#M100.0 WORD 10
       )
```

The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

Conversion operations

CONVERT: Convert value

Description

You use the "Convert value" instruction to read the content of the operand and convert it according to the assigned data types.

For information on possible conversions, refer to section "Explicit conversion" under "See also".

The value of the result is invalid if errors, such as an overflow, occur during processing.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted
<Result>	Output	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data types of the instruction from the "???" drop-down lists.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := INT_TO_DINT("Tag_InValue")
```

The content of the "Tag_InValue" operand is read and converted to an integer (32-bit). The result is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

ROUND: Round numerical value

Description

You use the "Round numerical value" instruction to round the value of the operand to the nearest integer. The instruction interprets the value as a floating-point number and converts this to an integer of data type DINT. If the input value is exactly between an even and odd number, the even number is selected.

The value of the result is invalid if errors, such as an overflow, occur during processing.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value to be rounded
<Result>	Output	DINT	I, Q, M, D, L, P	Result of rounding

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ROUND("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	1.50000000	-1.50000000
Tag_OutValue	2	-2

The floating-point number in the "Tag_InValue" operand is rounded to the nearest even integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

CEIL: Generate next higher integer from floating-point number

Description

You use the "Generate next higher integer from floating-point number" instruction to round the value of the operand to the next higher integer. The instruction interprets the value as a floating-point number and converts this number to the next higher integer. The result of the instruction can be greater than or equal to the input value.

The value of the result is invalid if errors, such as an overflow, occur during processing.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next higher integer

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := CEIL("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	0.50000000	-0.50000000
Tag_OutValue	1	0

The floating-point number in the "Tag_InValue" operand is rounded to the next higher integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

FLOOR: Generate next lower integer from floating-point number

Description

You use the "Generate next lower integer from floating-point number" instruction to round the value of the operand to the next lower integer. The instruction interprets the value as a floating-point number and converts this number to the next lower integer. The result of the instruction can be less than or equal to the input value.

The value of the result is invalid if errors, such as an overflow, occur during processing.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next lower integer

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := FLOOR("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	0.50000000	-0.50000000
Tag_OutValue	0	-1

The floating-point number in the "Tag_InValue" operand is rounded to the next lower integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

TRUNC: Truncate numerical value

Description

You use the "Truncate numerical value" instruction to form an integer from the value of the operand. The value is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and outputs it without decimal places as the result.

The value of the result is invalid if errors, such as an overflow, occur during processing.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
<Result>	Output	Integers	I, Q, M, D, L	Integer part of the input value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := TRUNC("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	1.50000000	-1.50000000
Tag_OutValue	1	-1

The integer part of the floating-point number in the "Tag_InValue" operand is converted to an integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

SCALE: Scale

Description

You use the "Scale" instruction to convert the integer of the IN parameter to a floating-point number that can be scaled in physical units between a low limit and a high limit. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit (LO_LIM) and an error is output.

When the specified low limit is greater than the high limit (LO_LIM > HI_LIM), the result is scaled inversely proportional to the input value.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL SCALE
     (IN := "Tag_InputValue"
      HI_LIM := "Tag_HighLimit"
      LO_LIM := "Tag_LowLimit"
      BIPOLAR := "Tag_Bipolar"
      RET_VAL => "Tag_ErrorCode"
      OUT => "Tag_OutputValue"
      )
```

The following table shows the values of the various operands before execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	50.03978588

The "Tag_InputValue" operand indicates the value that is to be converted and scaled. The high and low limits are defined by the "Tag_HighLimit" and "Tag_LowLimit" operands. You use operand "Tag_Bipolar" = TRUE to specify that the value of the IN parameter is to be interpreted as bipolar. The result of the instruction is output at the "Tag_OutputValue" operand.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

UNSCALE: Unscale

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state at the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at parameter IN is not within the limits defined by HI_LIM and LO_LIM, an error is output and the result is set to the nearest limit.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit

Parameters	Declaration	Data type	Memory area	Description
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats here: Switching display formats in the program status (Page 7698)	

Example

The following example shows how the instruction works:

GRAPH

```
CALL UNSCALE
      (IN := "Tag_InputValue"
       HI_LIM := "Tag_HighLimit"
       LO_LIM := "Tag_LowLimit"
       BIPOLAR := "Tag_Bipolar"
       RET_VAL => "Tag_ErrorCode"
       OUT => "Tag_OutputValue"
       )
```

The following table shows the values of the various operands before execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0

Parameters	Operand	Value
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	22

The "Tag_InputValue" operand indicates the value that is to be converted and unscaled. The high and low limits are defined by the "Tag_HighLimit" and "Tag_LowLimit" operands. You use operand "Tag_Bipolar" = TRUE to specify that the value of the IN parameter is to be interpreted as bipolar. The result of the instruction is output at the "Tag_OutputValue" operand.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Basic information on GRAPH (Page 7369)

Program control operations

Runtime control

COMPRESS: Compress CPU memory (S7-400)

Description

You use the "Compress CPU memory" instruction to compress the work memory of the CPU or the RAM load memory. During this compression the gaps caused by deleting or loading blocks are filled by pushing together the existing blocks.

The compression operation is started by calling the "Compress CPU memory" instruction. The compression operation is divided into several program cycles. The compression operation is monitored using the values of the individual parameters. If the BUSY parameter has the value "1", the compression operation is still being executed. The value "1" in the DONE parameter indicates that the compression operation is complete. Error information can be queried in the RET_VAL parameter.

If an externally started compression operation is already being executed, the calling of the "Compress CPU memory" instruction causes an error to be displayed.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Information on whether compression operation is active
DONE	Output	BOOL	I, Q, M, D, L	Information on whether compression operation is complete

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error The status of the compression operation can be queried in the BUSY and DONE parameters.
8091	An externally started compression operation is already being executed.
8092	The instruction cannot be executed, because one of the following conditions applies: • The "Delete block" function is active. • A test and commissioning function is acting on a block (for example, status). • The "copy blocks" function was triggered externally and is active. • The H-CPU is running a link-up or update function.
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

RE_TRIGR: Restart cycle monitoring time

Description

The "Restart cycle monitoring time" instruction is used to restart the cycle monitoring time of the CPU. The cycle monitoring time then restarts with the time you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called in all blocks regardless of the priority .

If the instruction is called in a block with a higher priority, for example, a hardware interrupt or a diagnostic interrupt, the instruction will not be executed.

The "Restart cycle monitoring time" instruction executes completely within a time span (10 times the maximum program cycle), regardless of the number of calls. Once this time has expired, the program cycle can no longer be prolonged.

Parameters

The "Restart cycle monitoring time" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Parameters

The "Exit program" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

WAIT: Configure time delay

Description

You can use the "Configure time delay" instruction to pause the program execution for a specific period of time. You indicate the period of time in microseconds in the WT parameter.

You can configure time delays from -32768 up to 32767 microseconds (µs). The smallest possible delay time depends on the respective CPU and corresponds to the execution time of the instruction.

The execution of the instruction can be interrupted by higher priority events and does not return any error information.

Valid for S7-300 CPUs:

For the S7-300 CPUs (except S7-318), the delay time programmed by the instruction is a minimum time. It is extended by the execution time of the nested priority classes and by system utilizations.

Note

Negative delay time

If you specify a negative delay time at the WT parameter, the enable output ENO, the RLO and the BR bit return the signal state FALSE. A negative delay time has no effect on the CPU. The following instructions connected to the ENO enable output are not executed in LAD and FBD.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

PROTECT: Change protection level

Description

You use the "Change protection level" instruction to change the configured protection level "1" of the CPU.

The assigned protection level "1" with selected option "Removable with password" means that you have to assign a password. This password enables you to write-access the CPU that was write-protected with the "Change Protection Level" instruction during operation (e.g., for loading).

You specify the protection level in the MODE parameter of the instruction. The current protection level of your CPU is displayed in the inspector window under "Properties > Protection".

The following table shows the meaning of the protection levels.

Protection level	Meaning
1	All programming device functions are permitted. Any existing blocking of the password authentication is canceled by calling the "Change protection level" instruction.
2	• The program and the configuration of the CPU cannot be changed. The program in the CPU can be read. • The functions for process control, process monitoring, and process communication are permitted. • All informational functions are permitted. • If you know the valid password you can cancel the set write protection. • Any existing blocking of the password authentication is canceled by calling the "Change protection level" instruction.
3	• You also cannot cancel the set read and write protection, even if you know the valid password. • If a valid connection exists at the time when the "Change protection level" instruction is called, the call of the instruction does not affect this connection.

If you have configured protection level "2" or "3" for your CPU, calling the "Change protection level" instruction has no effect.

You can use the SSL parts list with the SSL ID W#16#0232 and index W#16#0004 to check whether protection level "2" or "3" is still in effect after the execution of the "Change protection level" instruction.

Note

You can only use the "Change protection level" instruction to set a protection level that is equal to or higher than the protection level you have configured with STEP 7 "Configure hardware". You have, for example, configured protection level 3 and then called the "Change protection level" instruction with parameter MODE = 12. This blocks the password authentication. If the instruction is called again with parameter MODE = 0 or 1, protection level 3 is retained and the read and write protection is canceled by entry of the valid password.

Effects of the operating state transitions on the protection level

The following table shows what effects the operating state transitions have on the protection level set with the "Change protection level" instruction.

Action	Effects on the protection level
• Operating mode switch to STOP • Unbuffered POWER ON • Memory reset with operating mode switch • Configuration and/or programming changes in STOP mode of the S7-400 CPU	The protection level is set to "1". All programming device functions are permitted.
• Cold restart • Warm restart	The protection level is set by the operating system to "1". All programming device functions are permitted. If required, you can set the protection level in your program to "2" or "3" by calling the "Change protection level" instruction.
Buffered POWER ON	Remains unchanged
Operating mode transition RUN/STARTUP/HOLD -> STOP (by calling the "Exit program" instruction (STP) due to program error without corresponding error OB, or by operator action)	Remains unchanged
S7-400: Restart by user action or automatically following POWER ON	Remains unchanged

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	WORD	I, Q, M, D, L or constant	Protection level Possible values: • W#16#0000: Setting protection level "1" • W#16#0001: Setting protection level "2" with password authentication • W#16#000C: Setting protection level "3" without password authentication Constants can also be specified for the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs during processing, information on the error is output.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error
8090	Invalid value at parameter MODE.
80C3	The required operating resources are currently occupied.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

CiR: Control CiR process (S7-400)

Description

You use the "Control CiR process" instruction to influence the CiR process in the user program as follows:

- Disable CiR process. In this case, the download of a modified configuration from the programming device to the CPU is always rejected. The disable is maintained until you cancel it again with the "Control CiR process" instruction.
- You can disable the CiR process conditionally by specifying a high limit for the CiR synchronization time in the FRZ_TIME parameter. In this case, the download of a modified configuration from the programming device to the CPU is only accepted if the CiR synchronization time calculated in the CPU does not exceed the specified high limit.
- Enable CiR process. The high limit of the CiR synchronization time is set to the default value 1000 ms.
- Determine whether the CiR process is enabled or not. If the CiR process is enabled or conditionally enabled, you will get the current high limit for the CiR synchronization time in the A_FT parameter of the instruction.

Note

During the CiR synchronization period the outputs are frozen and the inputs are not evaluated.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Valid values for the MODE parameter are: • 0: Information function • 1: Enable CiR process • 2: Disable CiR process • 3: Disable CiR conditionally. You specify the high limit of the CiR synchronization time in FRZ_TIME.
FRZ_TIME	Input	TIME	I, Q, M, D, L or constant	High limit of the CiR synchronization time in milliseconds. Permitted range of values: 60 to 2500 ms (default: 1000 ms) The FRZ_TIME parameter is only relevant when the value "3" is set in the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	Error information When the MODE parameter has the value "0", the RET_VAL parameter supplies information on the enabling of the CiR process.
A_FT	Output	TIME	I, Q, M, D, L	Currently valid high limit of the CiR synchronization time

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error This error code can be output only if the value "1", "2" or "3" is set in the MODE parameter.
0001	CiR is enabled. This error code can be output only if the value "0" is set in the MODE parameter.
0002	The CiR process is disabled. This error code can be output only if the value "0" is set in the MODE parameter.
0003	CiR is disabled conditionally. This error code can be output only if the value "0" is set in the MODE parameter.

Error code* (W#16#...)	Explanation
8001	The CPU is not ready for a CiR operation. You are using an H CPU in an H-system (stand-alone), or you are working with a standard CPU in multicomputing mode.
8002	The value of the MODE parameter is invalid.
8003	The value of the FRZ_TIME parameter is invalid.
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

You use the "Control CiR process" instruction, for example, to ensure that no CiR process is initiated during periods in which maximum CPU performance is required for the execution of the process.

To use the "Control CiR process" instruction for this, the instruction must be called in the program before the start of the period with increased process activity and executed with the value "2" (disable CiR process) in the MODE parameter. The CiR process is then disabled.

To enable the CiR process again after the completion of the period with increased process activity, you must call the "Control CiR process" instruction again and execute it with the value "1" in the MODE parameter. Alternatively, you can also set the value "3" in the MODE parameter to disable the CiR process conditionally.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

Word logic operations

NOT: Inverting

Description

You use the instruction "Inverting" to invert the signal status of the bits of the operand. When the instruction is executed, the value of the operand is linked by EXCLUSIVE OR to a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit numbers). This inverts the signal state of the individual bits that are then output as the result.

Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Bit strings, integers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Bit strings, integers	I, Q, M, D, L, P	Ones complement of the operand value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"TagOut_Value" := NOT("TagIn_Value")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
TagIn_Value	W#16#000F
TagOut_Value	W#16#FFF0

The instruction inverts the signal state of the individual bits at "TagIn_Value" operand and writes the result to the "TagOut_Value" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

DECO: Decode

Description

You use the "Decode" instruction to set a bit in the output value that is specified by the input value .

The "Decode" instruction reads the value of the IN input and sets the bit in the OUT parameter whose bit position corresponds to the value read. The other bits in the output value are filled with zeroes. If the value of the IN input is greater than 31, a modulo 32 instruction is executed.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	WORD, INT	I, Q, M, D, L, P or constant	Position of the bit in the output value that is set.
OUT	Output	DWORD	I, Q, M, D, L, P	Output value

You can find additional information on valid data types under "See also".

You can select the data type of the instruction from the "???" drop-down list.

Example

The following example shows how the instruction works:

GRAPH

```
CALL DECO INT_DWORD
     (IN := "Tag_Input"
      OUT => "Tag_Output"
      )
```

The following figure shows how the instruction works using specific operand values:

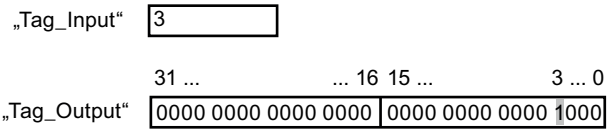

The instruction reads bit number "3" from the value of the "Tag_Input" operand of the input and sets the third bit in the value of the "Tag_Output" operand of the output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

ENCO: Encode

Description

You use the "Encode" instruction to read the bit number of the least significant set bit in the input value and output it in the OUT output.

The "Encode" instruction selects the least significant bit of the value in the IN input and writes its bit number to the tag in the OUT output. If the IN input has the value DW#16#00000001 or DW#16#00000000, the value "0" is output at the OUT parameter.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL ENCO DWORD
     (IN := "Tag_Input"
      OUT => "Tag_Output"
      )
```

The following figure shows how the instruction works using specific operand values:

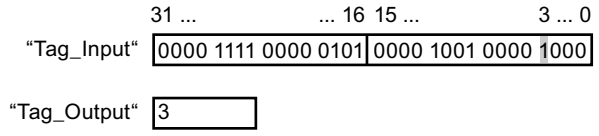

The instruction selects the least significant set bit in the "TagInput" input and writes bit position "3" to the tag in the "Tag_Output" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

SEL: Select

Description

You use the "Select" instruction to select, dependent on a switch (G input) one of two inputs, IN0 or IN1, and move its content to the OUT output. If the G input has the signal state "0", the value at the IN0 input is moved. If the G input has the signal state "1", the value of the IN1 input is moved to the OUT output.

The instruction is only executed if the tags of all parameters have the same data type.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
G	Input	BOOL	I, Q, M, D, L, T, C	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SEL INT
      (G := "Tag_Input_G"
       IN0 := "Tag_Input0"
       IN1 := "Tag_Input1"
       OUT => "Tag_Output"
       )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
G	Tag_Input_G	0	1
IN0	Tag_Input0	W#16#0000	W#16#4C
IN1	Tag_Input1	W#16#FFFF	W#16#5E
OUT	Tag_Output	W#16#0000	W#16#5E

Based on the signal state of the "Tag_Input_G" input, the value of the "Tag_Input0" or "Tag_Input1" input is selected and moved to the "Tag_Output" output.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

Shift and Rotate

SHR: Shift right

Description

You use the "Shift right" instruction to shift the content of Operand1 to the right bitwise. You use Operand2 to specify the number of bit positions the specified value will be shifted.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is shifted to the right by the available number of bit positions.

When unsigned values are shifted, the free bit positions in the left area of Operand1 are filled by zeroes. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure shows how the content of Operand1 of data type INT is shifted by four bit positions to the right:

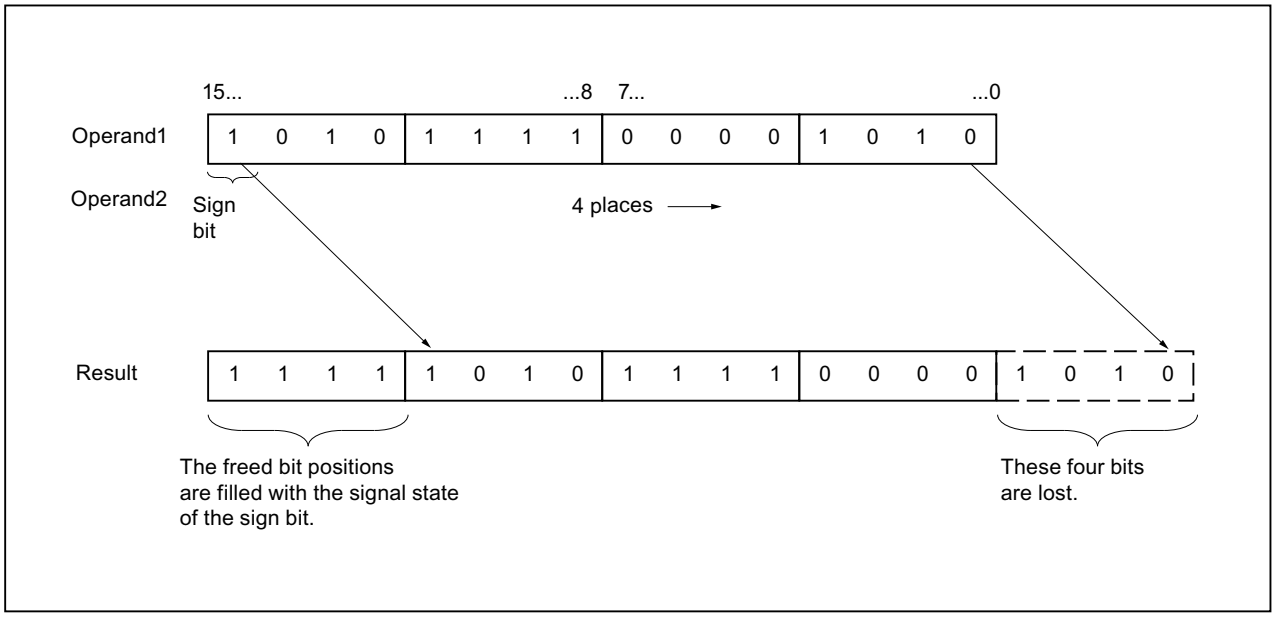

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L, P or constant	Value to be shifted
<Operand2>	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is shifted
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SHR_INT("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0011 1111 1010 1111
Tag_Number	3
Tag_OutValue	0000 0111 1111 0101

The content of the "Tag_InValue" operand is shifted three bit positions to the right and output as the result in the Tag_OutValue operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

SHL: Shift left

Description

You use the "Shift left" instruction to shift the content of Operand1 to the left bitwise. You use Operand2 to specify the number of bit positions the specified value will be shifted.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is shifted to the left by the available number of bit positions.

The bit positions in the right part of Operand1 freed by shifting are filled with zeros.

The following figure shows how the content of Operand1 of data type WORD is shifted by six bit positions to the left:

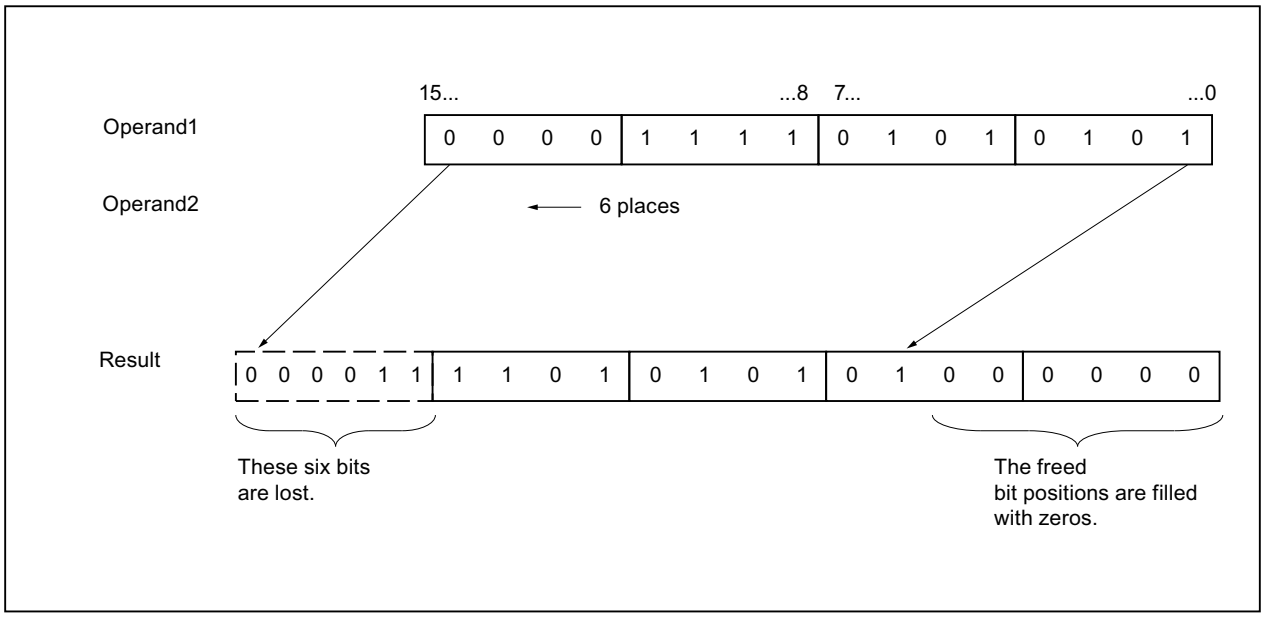

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L, P or constant	Value to be shifted
<Operand2>	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is shifted
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "???" drop-down list.

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SHL_WORD("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0011 1111 1010 1111
Tag_Number	4
Tag_OutValue	1111 1010 1111 0000

The content of the "Tag_InValue" operand is shifted four bit positions to the left and output as the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

ROR: Rotate right

Description

You use the "Rotate right" instruction to rotate the content of Operand1 to the right bitwise. You use Operand2 to specify the number of bit positions the specified value will be rotated.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is rotated by the specified number of bit positions.

The bit positions freed by rotating are filled with the bit positions that are pushed out.

The following figure shows how the content of an operand of data type DWORD is rotated three bit positions to the right:

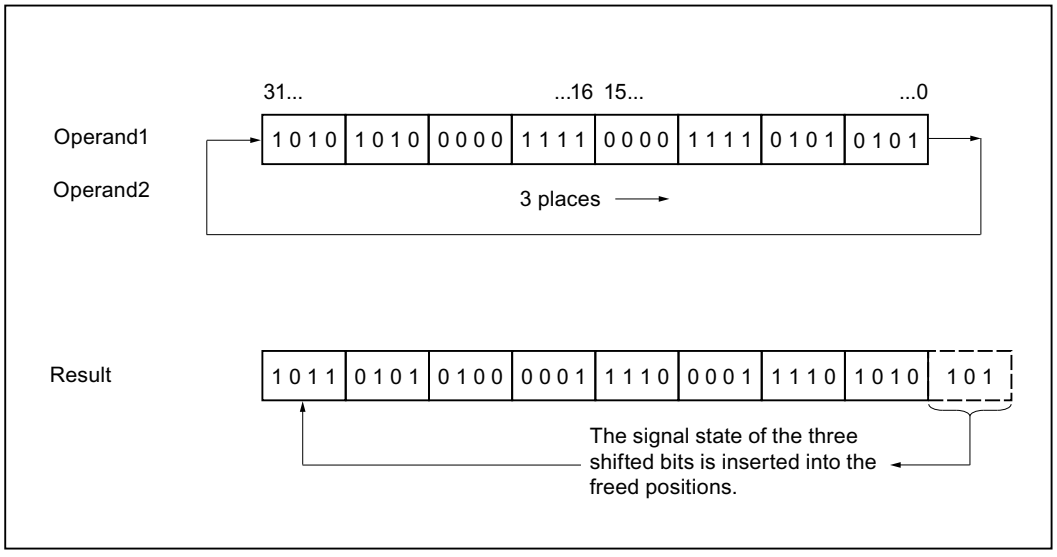

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P or constant	Value to be rotated
<Operand2>	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is rotated
<Result>	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ROR("Tag_InValue", "Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1010 1010 0000 1111 0000 1111 0101 0101
Tag_Number	5
Tag_OutValue	1010 1101 0101 0000 0111 1000 0111 1010

The content of the "Tag_InValue" operand is rotated five bit positions to the right and output as the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

ROL: Rotate left

Description

You use the "Rotate left" instruction to rotate the content of Operand1 to the left bitwise. You use Operand2 to specify the number of bit positions the specified value will be rotated.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is rotated by the specified number of bit positions.

The bit positions freed by rotating are filled with the bit positions that are pushed out.

The following figure shows how the content of an operand of data type DWORD is rotated three bit positions to the left:

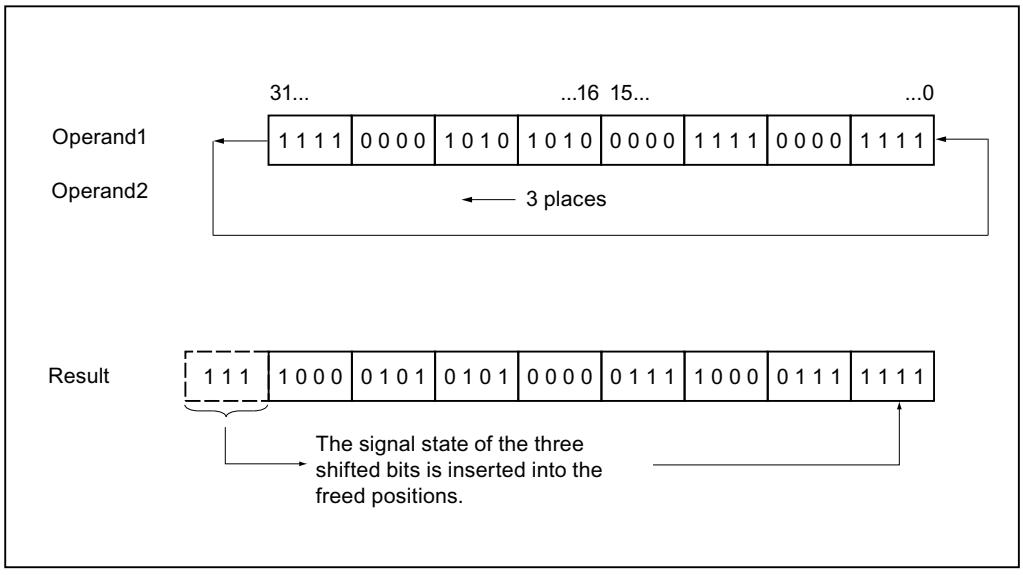

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P or constant	Value to be rotated
<Operand2>	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is rotated
<Result>	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ROL("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1111 0000 1010 1010 0000 1111 0000 1111
Tag_Number	5
Tag_OutValue	0001 0101 0100 0001 1110 0001 1111 1110

The content of the "Tag_InValue" operand is rotated five bit positions to the left and output as the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

Additional instructions

SET: Set bit array

Description

You use the "Set bit array" instruction to set the signal state of the bits in a specified area to "1". You use the pointer of the S_BIT parameter to define the starting point of the area. If the pointer points to the memory area of the external I/O, the instruction is not executed. You specify the number of bits to be set in the specified area in the N parameter. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified area are not changed.

Note

Parameter N

You can also interconnect an element of a data block at parameter N.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Points to the first bit of the area whose bits are set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be set

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SET
      (S_BIT := P#M0.0
       N := 10
       )
```

The instruction sets 10 bits in the area from M0.0 to M1.2.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

SETP: Set a bit array in the I/O area (S7-400)

Description

You use the "Set a bit array in the I/O area" instruction to set the bits of an I/O area to signal state "1". In parallel with the I/O bits, the corresponding bits in the process image output are set simultaneously.

You use the N and SA parameters to define the bit array that will be set. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case error information is output in the RET_VAL parameter.

The I/O area is set byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first or last byte to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

  WARNING

  Unintended behavior

  The setting of the bits to the value of the associated bits in the process image output can lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are set to the signal state "1".

The "Set a bit array in the I/O area" instruction must be called in an MCR area.

Calling the "Set a bit array in the I/O area" instruction has no effect if one of the following conditions applies:

- The N parameter has the value "0"
- The MCR bit has the signal state "1".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be set
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be set

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL SETP
     (N := 8
      RET_VAL => "Tag_Error"
      SA => P#10.0
      )
```

The instruction sets 8 bits in the I/O area corresponding to outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

SETI: Set byte array

Description

You use the "Set byte array" instruction to set the signal state of the bit in a specified area of bytes to "1". You define the start point of the area in the S_BYTE. You define the size of the area by specifying the number of bits to be set in the N parameter.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified in the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer of the S_BYTE parameter must point to an operand that corresponds to a byte limit value, i.e., the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process image output is reset to "0".

Calling the "Set byte array" instruction has no effect if one of the following conditions applies:

- The pointer of the S_BYTE parameter does not point to the external I/O memory area.
- The pointer of the S_BYTE parameter points to an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P or constant	Points to the first byte of the area whose bits are to be set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the area to be set The area is specified by the number of bits as a multiple of eight (for example, 8, 16, 24, etc.).

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SETI
     (S_BYTE := P#2.0
      N := 16
      )
```

The instruction sets 2 bytes (16 bits) in the area from P2.0 to P3.7.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

RESET: Reset bit array

Description

You use "Reset bit array" instruction to reset the signal state of the bits in a specified area to "0". You use the pointer of the S_BIT parameter to define the starting point of the area. If the pointer points to the memory area of the external I/O, the instruction is not executed. You specify the number of bits to be reset in the specified area in the N parameter. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified area are not changed.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Points to the first bit of the area whose bits are to be reset For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be reset

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL RESET
      (S_BIT := P#M0.0
       N := 10
       )
```

The instruction resets 10 bits in the area from M0.0 to M1.2.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

RESETP: Reset a bit array in the I/O area (S7-400)

Description

You use the "Reset a bit array in the I/O area" instruction to reset the bits of an I/O area to the signal state "0". In parallel with the I/O bits, the corresponding bits in the process image output are also reset simultaneously.

You use the N and SA parameters to define the bit array that will be reset. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to reset the entire bit array. In this case error information is output in the RET_VAL parameter.

The I/O area is reset byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first or last byte to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

  WARNING

  Unintended behavior

  The setting of the bits to the value of the associated bits in the process image output can lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are reset to the signal state "0".

The "Reset a bit array in the I/O area" instruction must be called in an MCR range.

Calling the "Reset a bit array in the I/O area" instruction has no effect if one of the following conditions applies:

- The N parameter has the value "0"
- The MCR bit has the signal state "1".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be reset
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be reset

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL RESETP
      (N := 8
       RET_VAL => "Tag_Error"
       SA => P#10.0
       )
```

The instruction resets 8 bits in the I/O area corresponding to outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 231)

Evaluating errors with output parameter RET_VAL (Page 162)

Switching display formats in the program status (Page 7698)

Basic information on GRAPH (Page 7369)

RESETI: Reset byte array

Description

You use the "Reset byte array" instruction to reset the signal state of the bits in a specified area of bytes to "0". You use the pointer of the S_BYTE parameter to define the starting point of the area. You define the size of the area by specifying the number of bits to be reset in the N parameter.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified in the S_BYTE parameter must point to the external I/O memory area. Since the "P" memory area is accessed as bytes, words, or double words, the pointer of the S_BYTE parameter must point to an operand that corresponds to a byte limit value, i.e., the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process image output is reset to "0".

Calling the "Reset byte array" instruction has no effect if one of the following conditions applies:

- The pointer of the S_BYTE parameter does not point to the external I/O memory area.
- The pointer of the S_BYTE parameter points to an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P or constant	Points to the first byte of the area whose bits are to be reset. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the area whose bits are to be reset. The area is specified by the number of bits as a multiple of eight (for example, 8, 16, 24, etc.).

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL RESETI
     (S_BYTE := P#2.0
      N := 16
      )
```

The instruction resets 2 bytes (16 bits) in the area from P2.0 to P3.7.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

REPL_VAL: Enter substitute value

Description

You use the "Enter substitute value" instruction to enter a substitute value in accumulator 1 (ACCU 1) from a synchronous error organization block.

Note

You can only call the "Enter substitute value" instruction in a synchronous error organization block (OB 121, OB 122).

You use the "Enter substitute value" instruction when you can no longer read any value from a module. The OB 122 is started after each access to this module. By calling the "Enter substitute value" instruction you can cause the substitute value to be loaded in accumulator 1. The program execution is then continued with this substitute value. You can take the required information for the selection of the substitute value from the local tags of OB 122.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
VAL	Input	DWORD	I, Q, M, D, L or constant	Substitute value
RET_VAL	Output	INT	I, Q, M, D, L	Error information

You can find additional information on valid data types under "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Description
0000	No error
8080	The instruction was not called from a synchronous error OB.
8xyy	For additional information on errors, refer to "See also".
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

DRUM: Implement sequencer

Description

You use the "Implement sequencer" instruction to assign the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD) with the programmed values of the OUT_VAL parameter of the relevant step. The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). When a bit of the configurable mask has a signal state of "1", the OUT_VAL value

sets/resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L	The signal state "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P	Current counter value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer

Parameter	Declaration	Data type	Memory area	Description
PrevTime	Static	TIME	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY[1..16] of WORD	I, Q, M, D, L	Preset counter value for each step [1 to 16] where one clock pulse = 1 ms.
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

You can find additional information on valid data types under "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value at the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value at the LST_STEP parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL DRUM, "DRUM_DB"
      (RESET := "Tag_Reset"
       JOG := "Tag_Input_Jog"
       DRUM_EN := "Tag_Input_Drum_EN"
       LST_STEP := "Tag_Number_LastStep"
       EVENTn := "MyTag_Event_n"
```

GRAPH

```
OUTn => "MyTag_Output_n"
Q => "Tag_Output_Q"
OUT_WORD => "Tag_OutputWord"
ERR_CODE => "Tag_ErrorCode"
)
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameter	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE

Parameter	Address	Value
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE

Parameter	Operand	Address	Value
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

DRUM_X: Implement sequencer

Description

You use the "Implement sequencer" instruction to implement a sequencer with a maximum of 16 steps. You specify the number of the first step in the DSP parameter. You specify the number of the last step on the LST_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT_WORD (in which the output bits are collected together) are written. An output bit is assigned either the corresponding bit of the OUT_VAL array that you specified or the output bit of the preceding step. The value assigned depends on the value of the mask bit on the S_MASK parameter.

Note

The default setting for the mask bit is 0. If you want to change the current setting of one or more mask bits, make this change in the instance data block.

The "Implement sequencer" instruction advances to the next step when there is a rising edge in the JOG parameter. If the instruction is already in the last step, a positive edge in the JOG parameter sets the Q and EOD tags. In this case the DCC parameter is set to the value "0". The instruction remains in the last step until the RESET parameter is set to the value "1".

You can also assign parameters so that switching to the next step is time dependent. To do this, the DRUM_EN parameter has to be set to the value "1". The transition to the next step is made when the event bit (EVENTi) belonging to the current step is set and the time specified

for the current step has expired. The time for a step is the product of the (DTBP) time base and the time factor valid for the current step (S_PRESET).

Note

The execution time (DCC) remaining in the current step is reduced only when the corresponding event bit (EVENTi) is set.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

If the value "1" is set in the RESET parameter when the instruction is called, the sequencer goes to the step whose number is specified in the DSP parameter.

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1".

You can also move the next step at any time via the value at the JOG parameter.

When the instruction is called for the first time, you must set the value "1" on the RESET parameter.

When the sequencer is in the last step and the execution time specified for this step has expired, the parameters Q and EOD are set. The instruction remains in the last step until the RESET parameter changes to the value "1".

The instruction is executed only in the STARTUP and RUN modes.

The operating system resets the "Implement sequencer" instruction during a cold restart. During a warm restart, the instruction is not reset. If you want the "Implement sequencer" instruction to be initialized after a warm restart, you must call the instruction in OB 100 with the value "1" in the RESET parameter.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" resets the sequencer. When the instruction is called for the first time, the RESET parameter must be set to "1".
JOG	Input	BOOL	I, Q, M, D, L	A positive signal edge (compared to the last call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you assigned to the DRUM_EN parameter.

Parameter	Declaration	Data type	Memory area	Description
DRUM_EN	Input	BOOL	I, Q, M, D, L	Control parameter that specifies whether time-dependent switching to the next step is possible. If the value is "1" a time-dependent switching to the next step is possible.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last step Values from 1 to 16 are permitted.
EVENTi, 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit number i (belongs to step i)
OUTj, 0 ≤ j ≤ 15	Output	BOOL	I, Q, M, D, L	Output bit number j (identical to the bit number j of OUT_WORD)
Q	Output	BOOL	I, Q, M, D, L	Status parameter that indicates whether the specified execution time of the last step has expired.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Output bits collected together in a tag
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	Not relevant to user (input parameter JOG of the preceding call)
EOD	Static	BOOL	I, Q, M, D, L	Identical to the output parameter Q
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Number of the first step Values from 1 to 16 are permitted.
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Number of the current step
DCC	Static	DWORD	I, Q, M, D, L, P	The execution time still remaining in the current step in milliseconds (ms) (only relevant if DRUM_EN = 1 and the corresponding event bit = 1)
DTBP	Static	WORD	I, Q, M, D, L, P or constant	The time base valid for all steps in milliseconds (ms)
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	(Not relevant for the user: system time of the previous call)
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	One-dimensional array with the time factors for each step. Recommended selection of the indices: [1 to 16]. In this case, S_PRESET [x] has the time factor of step x.

Parameter	Declaration	Data type	Memory area	Description
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the values to be output in each step if they have not been masked out using S_MASK. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, OUT_VAL [x, y] has the value assigned to the output bit OUTy in step x.
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the mask bits for each step. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, S_MASK [x, y] contains the mask bit for the "y"-th value output in step x. Meaning of the mask bits: • 0: The corresponding value of the previous step is assigned to the relevant output bit. • 1: The corresponding value from OUT_VAL is assigned to the relevant output bit.

You can find additional information on valid data types under "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE* (W#16#...)	Explanation
0000	No error
8081	Invalid value on parameter LST_STEP.
8082	Invalid value on parameter DSC.
8083	Invalid value on parameter DSP.
8084	The product DCC = DTBP * S_PRESET[DSC] exceeds the value 2**31-1 (approximately 24.86 days).
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT0 to OUT15) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL DRUM_X, "DRUM_X_DB"
      (RESET := "Tag_Reset"
       JOG := "Tag_Input_Jog"
       DRUM_EN := "Tag_Input_Drum_EN"
       LST_STEP := "Tag_Number_LastStep"
       EVENTn := "MyTag_Event_n"
       OUTn => "MyTag_Output_n"
       Q => "Tag_Output_Q"
       OUT_WORD => "Tag_OutputWord"
       ERR_CODE => "Tag_ErrorCode"
       )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameter	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_X_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE

Parameter	Address	Value
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT0	MyTag_Output_0	M4.0	TRUE
OUT1	MyTag_Output_1	M4.1	TRUE
OUT2	MyTag_Output_2	M4.2	TRUE
OUT3	MyTag_Output_3	M4.3	TRUE
OUT4	MyTag_Output_4	M4.4	TRUE
OUT5	MyTag_Output_5	M4.5	TRUE
OUT6	MyTag_Output_6	M4.6	TRUE
OUT7	MyTag_Output_7	M4.7	TRUE
OUT8	MyTag_Output_8	M5.0	TRUE
OUT9	MyTag_Output_9	M5.1	TRUE
OUT10	MyTag_Output_10	M5.2	TRUE
OUT11	MyTag_Output_11	M5.3	TRUE
OUT12	MyTag_Output_12	M5.4	TRUE
OUT13	MyTag_Output_13	M5.5	TRUE
OUT14	MyTag_Output_14	M5.6	TRUE
OUT15	MyTag_Output_15	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT0	MyTag_Output_0	M4.0	TRUE
OUT1	MyTag_Output_1	M4.1	FALSE
OUT2	MyTag_Output_2	M4.2	FALSE
OUT3	MyTag_Output_3	M4.3	FALSE
OUT4	MyTag_Output_4	M4.4	FALSE
OUT5	MyTag_Output_5	M4.5	TRUE
OUT6	MyTag_Output_6	M4.6	FALSE
OUT7	MyTag_Output_7	M4.7	FALSE
OUT8	MyTag_Output_8	M5.0	TRUE
OUT9	MyTag_Output_9	M5.1	TRUE
OUT10	MyTag_Output_10	M5.2	FALSE
OUT11	MyTag_Output_11	M5.3	FALSE
OUT12	MyTag_Output_12	M5.4	FALSE
OUT13	MyTag_Output_13	M5.5	FALSE
OUT14	MyTag_Output_14	M5.6	TRUE
OUT15	MyTag_Output_15	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are saved in the "DRUM_X_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

DCAT: Discrete control-timer alarm

Description

You can use the "Discrete control-timer alarm" instruction to accumulate the time from the point at which theCMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the Q, CMD_HIS, ET, OA, and CA parameters are influenced as follows:
  - The Q and CMD_HIS parameters are set to "1".
  - The ET, OA and CA parameters are reset to "0".
- When the signal state of the CMD parameter changes from "1" to "0", the Q, ET, OA, CA, and CMD_HIS parameters are reset to "0".
- When the signal state at the CMD and CMD_HIS parameters is "1" and the O_FB parameter is set to "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state at the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value at the CMD_HIS parameter is reset to the value of the CMD parameter.
- If the signal state of the CMD, CMD_HIS, and O_FB parameters is set to "1" and the C_FB parameter has the value "0", the signal state of the OA parameter is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the CMD, CMD_HIS, and C_FB parameters have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the ET parameter. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the CA parameter is reset to "1". If the value at the PT parameter is not exceeded, the CA parameter has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the CMD, CMD_HIS, and O_FB parameters have the signal state "0" and the C_FB parameter is set to "1", the CA parameter is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the O_FB and C_FB parameters simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction supplies no error information.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L	The signal state "0" indicates the "close" command. The signal state "1" indicates the "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the CMD parameter
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
CMD_HIS	Static	BOOL	I, Q, M, D, L	CMD history bit

You can find additional information on valid data types under "See also".

Example

In the following example, the CMD parameter changes from "0" to "1". After the execution of the instruction, the Q parameter is set to "1" and the two alarm outputs OA and CA have the signal state "0". The CMD_HIS parameter of the instance data block is set to the signal state "1" and the ET parameter is reset to "0".

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL DCAT, "DCAT_DB"
     (CMD := "Tag_Input_CMD"
      O_FB := "Tag_Input_O_FB"
      C_FB := "Tag_Input_C_FB"
      Q => "Tag_Output_Q"
      OA => "Tag_Output_OA"
      CA => "Tag_Output_CA"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the "DCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the "DCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

MCAT: Motor control-timer alarm

Description

You use the "Motor control-timer alarm" instruction to accumulate the time from the point at which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT

Input parameters		Output parameters
>=PT	ET >= PT	
If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to the signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore not possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case: OO = FALSE CO = FALSE OA = FALSE CA = FALSE ET = PT Q = TRUE		

The "Motor control-timer alarm" instruction supplies no error information.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when open-ing
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition.
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
O_HIS	Static	BOOL	I, Q, M, D, L	"Open" history bit
C_HIS	Static	BOOL	I, Q, M, D, L	"Close" history bit

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL MCAT, "MCAT_DB"
     (O_CMD := "Tag_Input_O_CMD"
      C_CMD := "Tag_Input_C_CMD"
      S_CMD := "Tag_Input_S_CMD"
      O_FB := "Tag_Input_O_FB"
      C_FB := "Tag_Input_C_FB"
      OO => "Tag_OutputOpen"
      CO => "Tag_OutputClosed"
      OA => "Tag_Output_OA"
      CA => "Tag_Output_CA"
      Q => "Tag_Output_Q"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

IMC: Compare input bits with the bits of a mask

Description

You use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the CMP_VAL[x,0] mask, where "x" indicates the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise, the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF BOOL	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

You can find additional information on valid data types under "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

SMC: Compare scan matrix

Description

You use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the CMP_VAL[x,0] mask, where "x" indicates the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY of WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

You can find additional information on valid data types under "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

LEAD_LAG: Lead and lag algorithm

Description

You use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and error information is output in the ERR_CODE parameter.

You use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two operations together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

You can find additional information on valid data types under "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL LEAD_LAG, "LEAD_LAG_DB"
     (IN := "Tag_Input"
```

GRAPH

```
SAMPLE_T := "Tag_Input_SAMPLE_T"
OUT => "Tag_Output_Result"
ERR_CODE => "Tag_ErrorCode"
)
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the "LEAD_LAD_DB" instance data block of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

TONR_X: Time accumulator

Description

You use the "Time accumulator" instruction to accumulate time values within a period which you set using the PV input. The instruction is executed when the signal state on the TMR_EN start input changes from "0" to "1". Simultaneously, the timer runs for the duration set at the PV input. While the instruction is being executed, it adds the time value of the DELTA_T input to the time value of the InOut ET.

The following conditions must be met:

- The TMR_EN start input has the signal state "1".
- The time value at InOut ET is less than the time value at the PV parameter.
- The RESET input has the signal state "0".

The instruction is interrupted when the signal state on the TMR_EN start input changes from "1" to "0". As soon as the signal state changes back to "1", the instruction resumes execution.

The instruction is ended when the time value at InOut ET has reached the time value at the PV input. The Q output then has the signal state "1".

As long as the Q output has the signal state "1", the time value at InOut ET remains unchanged. The time value and the Q output are not reset to "0" until the RESET input is set to signal state "1".

Since the "Time accumulator" instruction uses the cycle time (DELTA_T) of the last cycle of the organization block (OB), the instruction may only be called in cyclic organization blocks.

Note

You must move the cycle time of the organization block from the startup information to the global tag of the DELTA_T parameter.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
TMR_EN (Timer Enable)	Input	BOOL	I, Q, M, D, L	Start input Starts the instruction and the expiration of the preset time duration at the PV input
RESET	Input	BOOL	I, Q, M, D, L	Reset input
PV (Preset Value)	Input	DINT	I, Q, M, D, L, P or constant	Maximum duration of time recording
DELTA_T	Input	INT	I, Q, M, D, L or constant	Cycle time of organization block of the previous cycle

Parameter	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L	Output which is set when the preset time value is reached
ET (Elapsed Time)	InOut	DINT	I, Q, M, D, L	Accumulated time value (Currently elapsed time value + time value of the DELTA_T input)

You can find additional information on valid data types under "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:

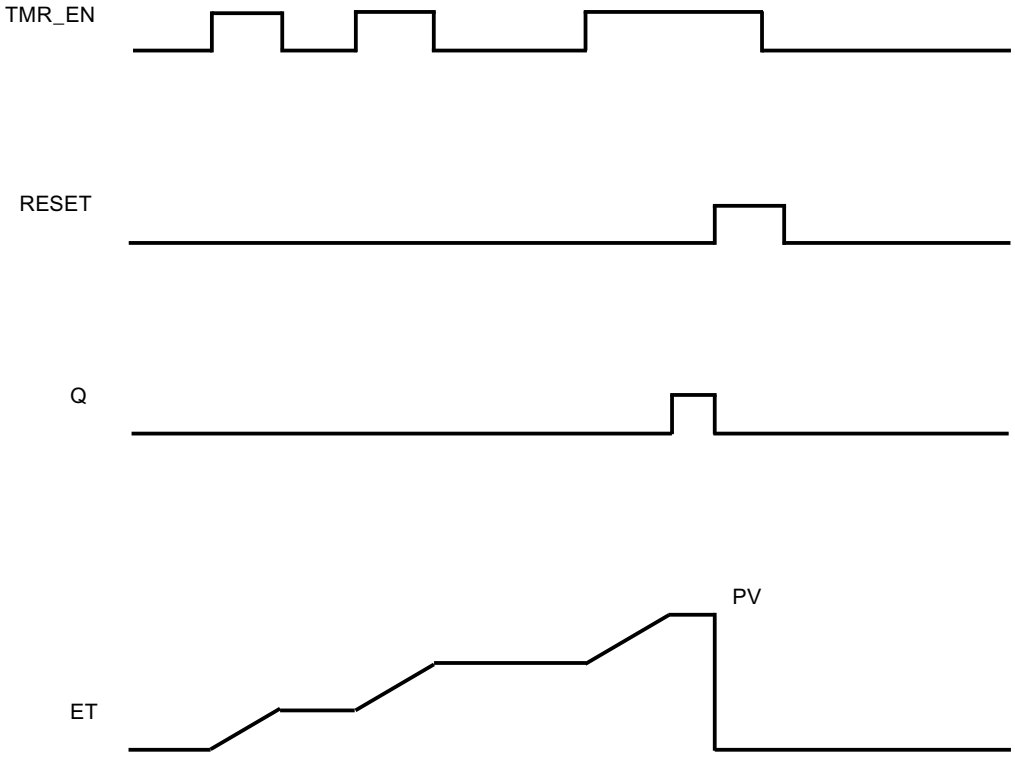

Example

The following example shows how the instruction works:

GRAPH

```
CALL TONR_X
      (TMR_EN := "Tag_Start"
       RESET := "Tag_Reset"
       PV := 10
       DELTA_T := 2
       Q => "Tag_Output"
```

GRAPH

```
      ET := "Tag_Elapsed_Time"
      )
```

Each time the signal state changes from "0" to "1" at the TMR_EN input, the cycle time "2" at the DELTA_T input is added to the current time value of InOut ET until the maximum time value "10" of the PV input is reached.

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

WSR: Save data to shift register

Description

You use the "Save data to shift register" instruction to write data from a specified source to a shift register. Each element of the register is shifted into the next address. The data contained in the last address of the shift register is lost after the shift.

Each time the instruction is executed with the value "0" in the RESET parameter, new data are read from the source (S_DATA) and shifted to the start address (START) of the shift register. The LENGTH parameter specifies the number of elements to be shifted. You specify the data type of the elements in the E_TYPE parameter. The instruction is not executed if an invalid data type is specified for the E_TYPE parameter.

The register is deleted if the RESET parameter is set to "1" when the instruction is executed. The Q parameter is set to the signal state "1" when the shift register is filled with zeros or deleted.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_DATA	Input	POINTER	I, Q, M, D	Points to the source data element to be shifted to the start address (START). For area-crossing, indirect register addressing, the pointer must be specified in double-word format.

Parameter	Declaration	Data type	Memory area	Description
START	Input	POINTER	I, Q, M, D	Points to the start address of the register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
LEN	Input	WORD	I, Q, M, D, L, P	Number of elements to be shifted.
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the elements to be shifted. The following data types are permitted: • W#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
Q	Output	BOOL	Q, M, D, L	Is set to "0" if the RESET parameter is active (1) or all elements to be shifted have the value "0".

You can find additional information on valid data types under "See also".

Example

In the following example four elements of the data type WORD are shifted starting from the address specified in the START parameter. When the first element of the register has been shifted to the next address, the first address is filled with the data that the pointer in the S_DATA parameter points to. The value of the last element in the register is lost in the process.

GRAPH

```
CALL WSR
      (RESET := "Tag_Input_Reset"
       S_DATA := "Tag_Input_DataSource"
       START := "Tag_Input_StartAddress"
       LEN := "Tag_Input_Length"
       E_TYPE := "Tag_Input_ElementType"
       Q => "Tag_Output_Q"
       )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1111 DW2 = W#16#2222 DW4 = W#16#3333 DW6 = W#16#4444
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1234 DW2 = W#16#1111 DW4 = W#16#2222 DW6 = W#16#3333
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

SHRB: Shift bit to shift register

Description

You use the "Shift bit to shift register" instruction to write a bit from a specified source (DATA) to a shift register. Each time the instruction is executed with a value "0" on the RESET parameter, new data are read from the source and shifted to the start address (S_BIT) of the shift register. All following bits are shifted by one. The bit contained at the last address (S_BIT + N) is lost after the shift. If the RESET parameter is set to "1" during the processing of the instruction, the addresses in the shift register are reset to "0" and not shifted.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
DATA	Input	BOOL	I, Q, M, D, L	Source bit
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_BIT	Input	POINTER	I, Q, M, D	Points to the start bit of the shift register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	WORD	I, Q, M, D, L, P	Number of bits to be shifted.

Example

In the following example the N parameter is set to the value 14 ("E" in hexadecimal). As a result 14 bits starting from the address specified in the S_BIT parameter are shifted. After the bits have been shifted, the first address is filled with the data from the source in the DATA parameter. The last bit value is lost.

GRAPH

```
CALL SHRB
      (DATA := "Tag_Input_Data"
       RESET := "Tag_Input_Reset"
       S_BIT := "Tag_Input_BitSource"
       N := "Tag_Input_Number"
       )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
DATA	Tag_Input_Data	0
S_BIT	Tag_Input_BitSource	P#DB1.DBX1.4
N	Tag_Input_Number	B#16#0E

Before processing the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	0	1	1	1	0	1	0

2.7	...					...	2.0
0	0	1	1	1	0	1	0

3.7	...					...	3.0
0	0	1	1	1	0	1	0

After processing

After the processing of the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	1	1	0	1	0	1	0

2.7	...					...	2.0
0	1	1	1	0	1	0	0

3.7	...					...	3.0
0	0	1	1	1	0	0	0

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

SEG: Create bit pattern for seven-segment display

Description

You use the "Create bit pattern for seven-segment display" instruction to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word of the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	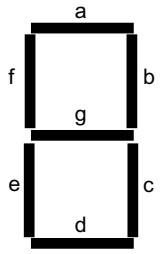
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SEG
      (IN := "Tag_Input"
       OUT => "Tag_Output"
       )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 Display: 1234

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

BCDCPL: Create tens complement

Description

You use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

You can find additional information on valid data types under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL BCDCPL
      (IN := "Tag_Input"
       ERR_CODE => "Tag_Output"
       )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Basic information on the status word (Page 173)

Basic information on GRAPH (Page 7369)

BITSUM: Count number of set bits

Description

You use the "Count number of set bits" instruction to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified in the IN parameter. The result of the instruction is output at the RET_VAL parameter.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

Example

The following example shows how the instruction works:

GRAPH

```
CALL BITSUM
      (IN := "Tag_Input"
       RET_VAL => "Tag_Output"
       )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 bits)
* The error codes can be displayed as integer or hexadecimal values in the program editor. You can find information on switching the display formats under "See also".		

See also

Overview of the valid data types (Page 231)

Basic information on GRAPH (Page 7369)

GRAPH LAD instructions

General LAD instructions in GRAPH

Description

Various LAD instructions are available to you when programming in GRAPH.

The following table shows an overview of the instructions and their availability:

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
General		
Insert network	x	
Empty box	x	x
Open branch	x	x
Close branch	x	x
Bit logic operations		
---| |---: Normally open contact (Page 4189)	x	x
---| / |---: Normally closed contact (Page 4191)	x	x
--| NOT |--: Invert RLO (Page 4192)	x	x
---( )---: Assignment (Page 4192)	x	
---(R)---: Reset output (Page 4193)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
---(S)---: Set output (Page 4194)	x	
SR: Set/reset flip-flop (Page 4195)	x	
RS: Reset/set flip-flop (Page 4196)	x	
--|P|--: Scan operand for positive signal edge (Page 4197)	x	
--|N|--: Scan operand for negative signal edge (Page 4199)	x	
P_TRIG: Scan RLO for positive signal edge (Page 4200)	x	
N_TRIG: Scan RLO for negative signal edge (Page 4202)	x	
Timers		
IEC timers		
TP: Generate pulse (Page 4203)	x	
TON: Generate on-delay (Page 4205)	x	
TOF: Generate off-delay (Page 4207)	x	
SIMATIC Timers		
S_PULSE: Assign pulse timer parameters and start (Page 4209)	x	
S_PEXT: Assign extended pulse timer parameters and start (Page 4212)	x	
S_ODT: Assign on-delay timer parameters and start (Page 4215)	x	
S_ODTS: Assign retentive on-delay timer parameters and start (Page 4217)	x	
S_OFFDT: Assign off-delay timer parameters and start (Page 4220)	x	
Counters		
IEC counters		
CTU: Count up (Page 4231)	x	
CTD: Count down (Page 4233)	x	
CTUD: Count up and down (Page 4235)	x	
SIMATIC Counters		
S_CU: Assign parameters and count up (Page 4237)	x	
S_CD: Assign parameters and count down (Page 4239)	x	
S_CUD: Assign parameters and count up / down (Page 4241)	x	
Comparator operations		
CMP ==: Equal (Page 4246)	x	x
CMP <>: Not equal (Page 4247)	x	x
CMP >=: Greater or equal (Page 4248)	x	x
CMP <=: Less or equal (Page 4250)	x	x
CMP >: Greater than (Page 4251)	x	x
CMP <: Less than (Page 4252)	x	x
CMP>T: Greater than step activation time (Page 5166)		x
CMP>U: Greater than uninterrupted step activation time (Page 5167)		x
Math functions		
ADD: Add (Page 4253)	x	
SUB: Subtract (Page 4254)	x	
MUL: Multiply (Page 4256)	x	
DIV: Divide (Page 4257)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
MOD: Return remainder of division (Page 4258)	x	
NEG: Create twos complement (Page 4259)	x	
ABS: Form absolute value (Page 4261)	x	
MIN: Get minimum (Page 4262)	x	
MAX: Get maximum (Page 4263)	x	
LIMIT: Set limit value (Page 4265)	x	
SQR: Form square (Page 4267)	x	
SQRT: Form square root (Page 4268)	x	
LN: Form natural logarithm (Page 4270)	x	
EXP: Form exponential value (Page 4271)	x	
SIN: Form sine value (Page 4272)	x	
COS: Form cosine value (Page 4273)	x	
TAN: Form tangent value (Page 4274)	x	
ASIN: Form arcsine value (Page 4276)	x	
ACOS: Form arccosine value (Page 4277)	x	
ATAN: Form arctangent value (Page 4944)	x	
Move operations		
MOVE: Move value (Page 4280)	x	
BLKMOV: Move block (Page 4282)	x	
UBLKMOV: Move block uninterruptible (Page 4285)	x	
FILL: Fill block (Page 4288)	x	
Conversion operations		
CONVERT: Convert value (Page 4292)	x	
ROUND: Round numerical value (Page 4293)	x	
CEIL: Generate next higher integer from floating-point number (Page 4294)	x	
FLOOR: Generate next lower integer from floating-point number (Page 4296)	x	
TRUNC: Truncate numerical value (Page 4297)	x	
SCALE: Scale (Page 4298)	x	
UNSCALE: Unscale (Page 4301)	x	
Program control operations		
Runtime control		
RE_TRIGR: Restart cycle monitoring time (Page 4310)	x	
STP: Exit program (Page 4310)	x	
WAIT: Configure time delay (Page 4311)	x	
PROTECT: Change protection level (Page 4312)	x	
COMPRESS: Compress CPU memory (S7-400) (Page 4308)	x	
CiR: Control CiR process (S7-400) (Page 4314)	x	
Word logic operations		
AND: AND logic operation (Page 4316)	x	
OR: OR logic operation (Page 4318)	x	
XOR: EXCLUSIVE OR logic operation (Page 4319)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
INVERT: Create ones complement (Page 4321)	x	
DECO: Decode (Page 4322)	x	
ENCO: Encode (Page 4323)	x	
SEL: Select (Page 4324)	x	
Shift and rotate		
SHR: Shift right (Page 4326)	x	
SHL: Shift left (Page 4328)	x	
ROR: Rotate right (Page 4330)	x	
ROL: Rotate left (Page 4331)	x	
Other instructions		
SET: Set bit array (Page 4346)	x	
SETP: Set a bit array in the I/O area (S7-400) (Page 4347)	x	
SETI: Set byte array (Page 4349)	x	
RESET: Reset bit array (Page 4351)	x	
RESETP: Reset a bit array in the I/O area (S7-400) (Page 4352)	x	
RESETI: Reset byte array (Page 4354)	x	
REPL_VAL: Enter substitute value (Page 4356)	x	
DRUM: Implement sequencer (Page 4357)	x	
DRUM_X: Implement sequencer (Page 4363)	x	
DCAT: Discrete control-timer alarm (Page 4371)	x	
MCAT: Motor control-timer alarm (Page 4374)	x	
IMC: Compare input bits with the bits of the mask (Page 4377)	x	
SMC: Compare scan matrix (Page 4380)	x	
LEAD_LAG: Lead and lag algorithm (Page 4382)	x	
TONR_X: Time accumulator (Page 4385)	x	
WSR: Save data to shift register (Page 4388)	x	
SHRB: Shift bit to shift register (Page 4390)	x	
SEG: Create bit pattern for seven-segment display (Page 4393)	x	
BCDCPL: Create tens complement (Page 4394)	x	
BITSUM: Count number of set bits (Page 4396)	x	

CMP>T: Greater than step activation time

Description

You use the "Greater than step activation time" instruction to monitor the total duration of a step in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the

comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the "Greater than step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers	I, Q, M, D, L or constant	Current or last activation time of the step
<Operand2>	Input	DINT, TIME, S5TIME	I, Q, M, D, L or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

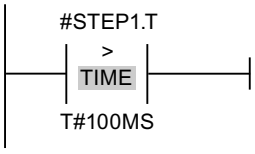

As long as the activation time of #STEP1.T is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

CMP>U: Greater than uninterrupted step activation time

Description

You use the "Greater than uninterrupted step activation time" instruction to monitor the duration of a step, minus any faults, in supervision conditions. The times of any events or faults are not recorded and only the pure step duration is monitored.

In the programmed condition, the total activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the "Greater than uninterrupted step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers	I, Q, M, D, L, or constant	Current or last activation time of the step minus the faults
<Operand2>	Input	DINT, TIME, S5TIME	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

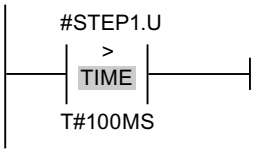

As long as the activation time of #STEP1.U, minus any faults, is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

GRAPH FBD instructions

General FBD instructions in GRAPH

Description

Different FBD instructions are available to you when programming in GRAPH.

The following table shows an overview of the instructions and their availability:

FBD instruction	Permanent operation	Transition/Supervision/Interlock
General		
New network: Insert network	x	
Empty box	x	x
Branch: Open branch	x	x
-|: Insert input (Page 4401)	x	x
-o|: Invert RLO (Page 4402)	x	x
Bit logic operations		
&: AND logic operation (Page 4397)	x	x
>=1: OR logic operation (Page 4398)	x	x

FBD instruction	Permanent operation	Transition/Supervision/Interlock
x: EXCLUSIVE OR logic operation (Page 4399)	x	
=: Assignment (Page 4402)	x	
-(R): Reset output (Page 4403)	x	
-(S): Set output (Page 4404)	x	
SR: Set/reset flip-flop (Page 4405)	x	
RS: Reset/set flip-flop (Page 4406)	x	
-|P|-: Scan operand for positive signal edge (Page 4407)	x	
-|N|-: Scan operand for negative signal edge (Page 4409)	x	
P_TRIG: Scan RLO for positive signal edge (Page 4410)	x	
N_TRIG: Scan RLO for negative signal edge (Page 4412)	x	
Timer operations		
IEC timers		
TP: Generate pulse (Page 4413)	x	
TON: Generate on-delay (Page 4415)	x	
TOF: Generate off-delay (Page 4417)	x	
SIMATIC timers		
S_PULSE: Assign pulse timer parameters and start (Page 4419)	x	
S_PEXT: Assign extended pulse timer parameters and start (Page 4422)	x	
S_ODT: Assign on-delay timer parameters and start (Page 4425)	x	
S_ODTS: Assign retentive on-delay timer parameters and start (Page 4428)	x	
S_OFFDT: Assign off-delay timer parameters and start (Page 4430)	x	
Counter operations		
IEC counters		
CTU: Count up (Page 4443)	x	
CTD: Count down (Page 4445)	x	
CTUD: Count up and down (Page 4446)	x	
SIMATIC counters		
S_CU: Assign parameters and count up (Page 4449)	x	
S_CD: Assign parameters and count down (Page 4451)	x	
S_CUD: Assign parameters and count up / down (Page 4452)	x	
Comparator operations		
CMP==: Equal (Page 4458)	x	x
CMP<>: Not equal (Page 4459)	x	x
CMP>=: Greater or equal (Page 4460)	x	x
CMP<=: Less or equal (Page 4461)	x	x
CMP>: Greater than (Page 4462)	x	x
CMP<: Less than (Page 4463)	x	x
CMP>T: Greater than step activation time (Page 5172)		x
CMP>U: Greater than uninterrupted step activation time (Page 5172)		x
Math functions		
ADD: Add (Page 4465)	x	

FBD instruction	Permanent operation	Transition/Supervision/Interlock
SUB: Subtract (Page 4466)	x	
MUL: Multiply (Page 4467)	x	
DIV: Divide (Page 4468)	x	
MOD: Return remainder of division (Page 4470)	x	
NEG: Create twos complement (Page 4471)	x	
ABS: Form absolute value (Page 4472)	x	
MIN: Get minimum (Page 4473)	x	
MAX: Get maximum (Page 4475)	x	
LIMIT: Set limit value (Page 4477)	x	
SQR: Form square (Page 4479)	x	
SQRT: Form square root (Page 4480)	x	
LN: Form natural logarithm (Page 4482)	x	
EXP: Form exponential value (Page 4483)	x	
SIN: Form sine value (Page 4484)	x	
COS: Form cosine value (Page 4486)	x	
TAN: Form tangent value (Page 4487)	x	
ASIN: Form arcsine value (Page 4488)	x	
ACOS: Form arccosine value (Page 4490)	x	
ATAN: Form arctangent value (Page 4491)	x	
Move operations		
MOVE: Move value (Page 4493)	x	
BLKMOV: Move block (Page 4495)	x	
UBLKMOV: Move block uninterruptible (Page 4498)	x	
FILL: Fill block (Page 4501)	x	
Conversion operations		
CONVERT: Convert value (Page 4504)	x	
ROUND: Round numerical value (Page 4506)	x	
CEIL: Generate next higher integer from floating-point number (Page 4507)	x	
FLOOR: Generate next lower integer from floating-point number (Page 4508)	x	
TRUNC: Truncate numerical value (Page 4510)	x	
SCALE: Scale (Page 4511)	x	
UNSCALE: Unscale (Page 4513)	x	
Program control operations		
Runtime control		
RE_TRIGR: Restart cycle monitoring time (Page 4522)	x	
STP: Exit program (Page 4523)	x	
PROTECT: Change protection level (Page 4524)	x	
WAIT: Configure time delay (Page 4523)	x	
COMPRESS: Compress CPU memory (S7-400) (Page 4521)	x	
CiR: Control CiR process (S7-400) (Page 4527)	x	
Word logic operations		

FBD instruction	Permanent operation	Transition/Supervision/Interlock
AND: AND logic operation (Page 4529)	x	
OR: OR logic operation (Page 4531)	x	
XOR: EXCLUSIVE OR logic operation (Page 4532)	x	
INVERT: Create ones complement (Page 4534)	x	
DECO: Decode (Page 4535)	x	
ENCO: Encode (Page 4536)	x	
SEL: Select (Page 4538)	x	
Shift and rotate		
SHR: Shift right (Page 4539)	x	
SHL: Shift left (Page 4541)	x	
ROR: Rotate right (Page 4543)	x	
ROL: Rotate left (Page 4545)	x	
Additional instructions		
SET: Set bit array (Page 4559)	x	
SETP: Set a bit array in the I/O area (S7-400) (Page 4560)	x	
SETI: Set byte array (Page 4562)	x	
RESET: Reset bit array (Page 4564)	x	
RESETP: Reset a bit array in the I/O area (S7-400) (Page 4565)	x	
RESETI: Reset byte array (Page 4567)	x	
REPL_VAL: Enter substitute value (Page 4569)	x	
DRUM: Implement sequencer (Page 4570)	x	
DRUM_X: Implement sequencer (Page 4576)	x	
DCAT: Discrete control-timer alarm (Page 4584)	x	
MCAT: Motor control-timer alarm (Page 4587)	x	
IMC: Compare input bits with the bits of the mask (Page 4590)	x	
SMC: Compare scan matrix (Page 4593)	x	
LEAD_LAG: Lead and lag algorithm (Page 4595)	x	
TONR_X: Time accumulator (Page 4598)	x	
WSR: Save data to shift register (Page 4600)	x	
SHRB: Shift bit to shift register (Page 4603)	x	
SEG: Create bit pattern for seven-segment display (Page 4607)	x	
BCDCPL: Create tens complement (Page 4608)	x	
BITSUM: Count number of set bits (Page 4609)	x	

See also

JMP: Jump if RLO = 1 (Page 4515)

JMPN: Jump if RLO = 0 (Page 4516)

LABEL: Jump label (Page 4517)

CMP>T: Greater than step activation time

Description

You use the "Greater than step activation time" instruction to monitor the total duration of a step in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the "Greater than step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers	I, Q, M, D, L, or constant	Current or last activation time of the step
<Operand2>	Input	DINT, TIME, S5TIME	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

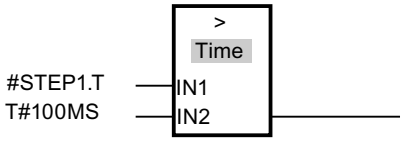

As long as the activation time of #STEP1.T is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

CMP>U: Greater than uninterrupted step activation time

Description

You use the "Greater than uninterrupted step activation time" instruction to monitor the duration of a step, minus any faults, in supervision conditions. The times of any events or faults are not recorded and only the pure step duration is monitored.

In the programmed condition, the total activation time of a step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the "Greater than uninterrupted step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers	I, Q, M, D, L, or constant	Current or last activation time of the step minus the faults
<Operand2>	Input	DINT, TIME, S5TIME	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:

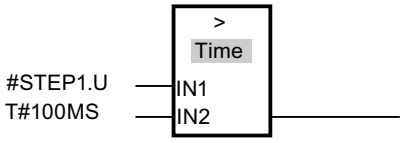

As long as the activation time of #STEP1.U, minus any faults, is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

4.2.3 Extended instructions

4.2.3.1 Date and time

T_COMP: Compare time tags

Description

You use the instruction to compare the contents of two tags with the data type DT, and output the result of the comparison as return value at parameter OUT. The parameter OUT is set to "1", when the applied comparison condition is satisfied.

The following comparison options can be used:

Symbol	Description
EQ	The return value has the signal state “1" if the time is the same at the parameter IN1 and IN2.
NE	The return value has the signal state “1" if the time at parameters IN1 and IN2 is not identical.
GE	The return value has the signal state "1" if the time at parameter IN1 is greater (more recent) than or equal to the time at parameter IN2 .
LE	The return value has the signal state "1" if the time at parameter IN1 is less (less recent) than or equal to the time at parameter IN2.

Symbol	Description
GT	The return value has the signal state “1" if the time at parameter IN1 is greater (more recent) than the time at parameter IN2.
LT	The return value has the signal state "1" if the time at parameter IN1 is less (less recent) than the time at parameter IN2 .

Parameter

The following table shows the parameters of the "T_COMP" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DT	D, L	Value to be compared
IN2	Input	DT	D, L	Value to be compared
OUT	Output	BOOL	I, Q, M, D, L	Return value

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

T_CONV: Convert times and extract

Description

The "T_CONV" instruction is used to convert a DT (DATE_AND_TIME), S5TIME and TIME data types to a DATE, INT, TOD (TIME_OF_DAY), TIME or S5TIME data type.

Parameter

You decide the type of conversion by selecting the data types in the instruction boxes of the input and output of the instruction. You can query the result of the conversion at the OUT parameter.

The following table shows the parameters of the "T_CONV" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DT, S5TIME, TIME	I, Q, M, D, L	Value to be converted. The data format of the tag for parameter IN must match the data type selected using the instruction box.
OUT	Return	DATE, INT, TOD, TIME, S5TIME	I, Q, M, D, L	Result of the conversion. The data format of the tag for parameter OUT must match the data type selected using the instruction box.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Examples

Table 4-187 Converting data format TIME to data format S5TIME

Parameter	Declaration	Data type	Description
IN	Output	TIME	Input variable in the TIME data format.
OUT	Input	S5TIME	The function converts the TIME data format to the S5TIME format. The value is rounded down during conversion. If the input parameter is greater than the S5TIME format permits (greater than TIME#02:46:30.000), the result S5TIME#999.3 will be output and the binary result (BR) is set to "0".

Table 4-188 Extraction of the weekday from the DATE_AND_TIME format

Parameter	Declaration	Data type	Description
IN	Output	DT	Input variable in the DATE_AND_TIME data format.
OUT	Input	INT	The function extracts the weekday from the DATE_AND_TIME (DT) format. The weekday is output in INTEGER format. 1: Sunday 2: Monday 3: Tuesday 4: Wednesday 5: Thursday 6: Friday 7: Saturday

T_ADD: Add times

Description

You use the instruction "T_ADD" to add a duration (format TIME) to a new time (format DT) and output the result as a new time (format DT).

The time DT for the parameter IN1 must be in the range between DT#1990-01-01-00:00:00.000 and DT#2089-12-31-23:59:59.999. The instruction does not run an input check. If the result of the addition is not within the valid range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to “0".

Parameters

The following table shows the parameters of the "T_ADD" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	DT	D, L	Time in the DT (DATE_AND_TIME) format.
IN2	Input	TIME	I, Q, M, D, L or constant	Duration to be added in the TIME format.
OUT	Return	DT	D, L	Result of the addition in the DT (DATE_AND_TIME) format.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

T_SUB: Subtract times

Description

You use the instruction "T_SUB" to subtract a duration (format TIME) from a time (format DT)and output the result as a new time (format DT).

The time DT for the parameter IN1 must be in the range between DT#1990-01-01-00:00:00.000 and DT#2089-12-31-23:59:59.999. The instruction does not run an input check. If the result of the subtraction is not within the valid range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to “0".

Parameters

The following table shows the parameters of the "T_SUB" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	DT	D, L	Time in the DT (DATE_AND_TIME) format.
IN2	Input	TIME	I, Q, M, D, L or constant	Duration to be subtracted in the TIME format.
OUT	Return	DT	D, L	Result of the subtraction in the DT (DATE_AND_TIME) format.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

T_DIFF: Time difference

Description

You use the instruction "T_DIFF" to determine the difference between two times by subtracting two times of the DT format. The time at the IN2 input is subtracted form the time at the IN1 input. The result of the subtraction is output at the OUT output in the TIME format.

- If the first time at the IN1 input is greater (younger) than the second time at the IN2 input, the result at the OUT output is positive.
- If the first time at the IN1 input is less (older) than the second time at the IN2 input, the result at the OUT output is negative.

If the result of the subtraction is outside the TIME range, the result is limited to the corresponding value and the binary result (BR) is set to "0".

Parameter

The following table shows the parameters of the "T_DIFF" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	DT	D, L	Minuend
IN2	Input	DT	D, L	Subtrahend
OUT	Return	TIME	I, Q, M, D, L	Difference in the TIME format.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

T_COMBINE: Combine times

Description

The "T_COMBINE" instruction combines the DATE and TIME_OF_DAY (TOD) data formats and converts these formats into the DATE_AND_TIME (DT) data format. The instruction does not report any errors.

- The DATE input value must be between DATE#1990-01-01 and DATE#2089-12-31 (will not be checked).
- At the IN2 input value, the TIME_OF_DAY data type is used.
- At the OUT output value, the DATE_AND_TIME data type is output.

Parameter

The following table shows the parameters of the "T_COMBINE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DATE	D, L or constant	Input tag in the DATE format.
IN2	Input	TOD	D, L or constant	Input tag in the TOD format.
OUT	Return	DT	D, L	Return value in the DT format.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Time-of-day functions

WR_SYS_T: Set time-of-day

Description

You set the time and the date of the CPU clock with the "WR_SYS_T" instruction. The clock then runs starting from the set time and date. You specify the date and time in the DT data format at the IN input of the instruction. You can use the "T_COMBINE (Page 5177)" instruction to convert the DATE and TOD formats to the required DT data format.

If the clock is a master clock, the CPU also starts the time-of-day synchronization when the instruction is called. You set the synchronization intervals per configuration. The "WR_SYS_T" instruction cannot be used to pass information about the local time zone or daylight saving time.

At the RET_VAL output, you can query whether errors have occurred during execution of the instruction.

Parameter

The following table shows the parameters of the "WR_SYS_T" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DT	D, L	Date and time
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
8080	Error in date
8081	Year invalid
8082	Month invalid
8083	Day invalid
8084	Hour information invalid
8085	Minute information invalid
8086	Second information invalid
8087	Nanosecond information invalid
80B0	The real-time clock has failed

Example

You enter the date and time as DT data type. As an example: for January 15, 2010, 10:30 a.m. and 30 seconds you would enter: DT#2010-01-15-10:30:30 The time-of-day can only be input down to seconds. The day of the week is calculated by "WR_SYS_T" based on the date.

See also

Evaluating errors with output parameter RET_VAL (Page 162)

RD_SYS_T: Read time-of-day

Description

You use the "RD_SYS_T" instruction to read the current date and current time-of-day of the CPU clock. The read data is provided in DT format at the OUT output of the instruction. The provided value does not include information about the local time zone or daylight saving time. At the RET_VAL output, you can query whether errors have occurred during execution of the instruction.

Parameter

The following table shows the parameters of the instruction "RD_SYS_T":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
OUT	Output	DT	D,L	Date and time of CPU

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
8222	The result is outside the permitted range of values.
8223	The result cannot be saved in the specified data type.

See also

Evaluating errors with output parameter RET_VAL (Page 162)

TIME_TCK: Read system time

Description

With the "TIME_TCK" instruction, you read the system time of the CPU. The system time is a time counter that counts from 0 to a maximum of 2147483647 ms. In case of an overflow, the system time is counted again starting with "0". The time scale and the accuracy of the system time is 1 ms. The system time is only influenced by the operating modes of the CPU. You can use the system time, for example, to measure the duration of processes by comparing the results of two "TIME_TCK" calls. The instruction does not provide any error information.

The following table provides an overview of how the system time changes depending on the operating modes of the CPU.

Mode	System time ...
Startup	... is constantly updated
RUN	
STOP	... is stopped and retains the current value
Restart (not with the S7-300)	... continues with the value saved at the change to STOP mode
Warm restart	... is deleted and restarts with "0"
Cold restart	

Parameter

The following table shows the parameters of the instruction "TIME_TCK":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	TIME	I, Q, M, D, L	The RET_VAL parameter contains the read system time in the range from 0 to 2^{31} -1 ms.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

SET_CLKS: Set time-of-day and time-of-day status

Description

You use the "SET_CLKS" instruction to set the time-of-day of your CPU and the time-of-day status.

Note

Use "SET_CLKS" only if the time-of-day for your CPU is not going to be synchronized. Otherwise, with every synchronization the master clock time-of-day status would be applied. This step would overwrite the value specified with "SET_CLKS".

Note

You can determine the current time-of-day status of your CPU by reading SZL-ID W#16#0132 Index W#16#0008 (Page 5475) using the "RDSYSST (Page 5444)" instruction.

Parameters

The following table shows the parameters of the "SET_CLKS" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Operating mode: • B#16#01: Set time-of-day • B#16#02: Set time-of-day status • B#16#03: Set time-of-day and time-of-day status
PDT	Input	DT	D, L	Specified time-of-day
CORR	Input	INT	I, Q, M, D, L or constant	Correction value (in 0.5 h pattern) Possible values: –24 to +26
SUMMER	Input	BOOL	I, Q, M, D, L	Daylight saving time/standard time ID: • 0 = Standard time • 1 = Daylight saving time
ANN_1	Input	BOOL	I, Q, M, D, L	Announcement hour 1: At the next hourly change daylight saving time is switched over to standard time or vice versa.
RET_VAL	Return	INT	I, Q, M, D, L	Error code

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter MODE

Use the MODE parameter to specify whether to change only the time-of-day, only the time-of-day status, or both. This is explained in the table below:

MODE	Meaning
B#16#01	Set time-of-day The call corresponds to the call of the "SET_CLK" instruction. The input parameters CORR, SUMMER and ANN_1 are not evaluated.
B#16#02	Set time-of-day status The input parameter PDT is not evaluated. The remaining input parameters form the following time-of-day status elements: • Correction value including the sign • Announcement hour • Daylight saving time/standard time indicator The time-of-day resolution is matched to that of your CPU. The bit for synchronization failure of the time-of-day status is written with FALSE. The time-of-day remains unchanged.
B#16#03	Set time-of-day and time-of-day status

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	MODE outside the permitted value range
8081	CORR outside the permitted value range (only for MODE = B#16#02 or for MODE = B#16#03)
8082	PDT outside the permitted value range Illegal date and/or time-of-day
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

SNC_RTCB: Synchronize slave clocks

Definition: Synchronization of slave clocks

Synchronize slave clocks refers to the transmission of the date and time from the master clock of a bus segment (for example, the S7-400 communication bus, MPI, or S7 backplane bus) to all clock slaves of the bus segment.

Description

To ensure that the time-of-day is consistent in all modules on the network, the time-of-day slaves are updated by the system program at regular assignable intervals.

A synchronization can be performed independently from the configured synchronization interval by calling the "SNC_RTCB" instruction. This involves transmitting the date and time from the master clock to the slave clock for all time slaves on an existing bus segment.

Successful synchronization is only possible if "SNC_RTCB" is called on a CPU whose real-time clock was assigned the master clock function for at least one bus segment.

Parameter

The following table shows the parameters of the instruction "SNC_RTCB":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred during synchronization.
0001	The existing clock was not assigned the master clock function for any of the bus segments.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Additional standard libraries

Handling of the runtime meter

Runtime meters

Introduction

The CPUs have a specific quantity of runtime meters (refer to the technical specifications of your CPUs).

- If your CPU is equipped with 16-bit runtime meters, you can set, start, stop, and read them with the "SET_RTM (Page 5186)", "CTRL_RTM (Page 5187)" and "READ_RTM (Page 5188)" instructions.
- If your CPU is equipped with 32-bit runtime meters, you can set, start, stop or read them with the "RTM (Page 5185)" instruction.

Note

You can also use the "SET_RTM (Page 5186)", "CTRL_RTM (Page 5187)" and "READ_RTM (Page 5188)" instructions for the 32-bit runtime meters. In this case, however, the runtime meters operate in the same way as 16-bit runtime meters (range of values: 0 to 32767 hours).

See also: Data record of the partial list extract with SZL-ID W#16#0132 index W#16#000B (Page 5477).

Application

You can use a runtime meter for a variety of applications:

- For measuring the runtime of the CPU
- For measuring the runtime of controlled equipment or connected devices.

Characteristics of the runtime meter

When it is started, the runtime meter begins to count starting at the last recorded value. If you want it to start at a different initial value, you must explicitly specify this value ("SET_RTM (Page 5186)" or "RTM (Page 5185)" with MODE=4). If the CPU changes to STOP mode, or you stop the runtime meter, the CPU records the current value of the runtime meter. When a warm restart or a cold restart of the CPU is executed, the runtime meter must be restarted ("CTRL_RTM (Page 5187)" or "RTM (Page 5185)" with MODE=1).

The runtime meters are reset to "0" following an update of the operating system or following a CPU reset to the factory state.

Range of values

- CPU with 16-bit runtime meters: 0 to 32 767 hours
- CPU with 32-bit runtime meters: 0 to (2E31) -1 hours = 2,147,483,647 hours

RTM: Runtime meters

Description

With the "RTM" instruction, you can set, start, stop, and read a 32-bit operating hours counter of your CPU.

If you want to read all 32-bit operating hours counters of your CPU, use the "RDSYSST (Page 5444)" instruction with the SZL_ID=W#16#0132 parameter and:

- INDEX=W#16#000B (for the operating hours counters 0 to 7) or
- INDEX=W#16#000C (for the operating hours counters 8 to 15).

See also: Data record of the partial list extract with SZL-ID W#16#0132 index W#16#000B (Page 5477)

Parameter

The following table shows the parameters of the "RTM" instruction:

Parameters	Declaration	Data type	Memory area	Description
NR	Input	BYTE	I, Q, M, D, L or constant	Number of the operating hours counter Numbering starts with 0. For information on the number of operating hours counters of your CPU, refer to the technical data.
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID: • 0: Read out (the status is then written to CQ and the current value to CV). After the operating hours counter has reached (2^31) - 1 hours, it stops at the highest value that can be displayed and outputs an "Overflow" error message. • 1: start (at the last counter value) • 2: stop • 4: set (to the value specified in PV) • 5: set (to the value specified in PV) and then start • 6: set (to the value specified in PV) and then stop
PV	Input	DINT	I, Q, M, D, L or constant	New value for the operating hours counter
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

Parameters	Declaration	Data type	Memory area	Description
CQ	Output	BOOL	I, Q, M, D, L	Status of the operating hours counter (1: running)
CV	Output	DINT	I, Q, M, D, L	Current value of the operating hours counter

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Compatibility to programs developed for a CPU with 16-bit operating hours counters

You can also use the 32-bit operating hours counters with the "SET_RTM (Page 5186)", "CTRL_RTM (Page 5187)" and "READ_RTM (Page 5188)" instructions. In this case, however, the 32-bit operating hours counters operate in the same way as 16-bit meters (range of values: 0 to 32767 hours).

The partial list extract with SZL-ID W#16#0132 and the index W#16#0008 displays the 32-bit operating hours counters 0 to 7 in 16-bit mode. This means you can continue to use programs developed for a CPU with 16-bit operating hours counters that use the partial list extract with SZL-ID W#16#0132 and the index W#16#0008 .

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the operating hours counter
8081	A negative value was passed to the PV parameter.
8082	Overflow of the operating hours counter
8091	The MODE input parameter contains an invalid value.
8xyy	General error information See also:Evaluating errors with output parameter RET_VAL (Page 162)

SET_RTM: Set runtime meters

Description

You use the instruction to set a CPU runtime meter to a specified value. The number of runtime meters you can set depends on the particular CPU you are using.

Parameters

The following table shows the parameters of the instruction "SET_RTM":

Parameters	Declaration	Data type	Memory area	Description
NR	Input	BYTE	I, Q, M, D, L or constant	Input NR contains the number of the runtime meter you want to set. The numbering starts with "0". For information on the number of runtime meters of your CPU, refer to the technical data.
PV	Input	INT	I, Q, M, D, L or constant	Input PV contains the setting for the runtime meter.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the runtime meter
8081	A negative value was transferred to the PV parameter.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

CTRL_RTM: Start and stop runtime meters

Description

You use the instruction to start or stop a runtime meter.

Parameters

The following table shows the parameters of the "CTRL_RTM" instruction:

Parameter	Declaration	Data type	Memory area	Description
NR	Input	BYTE	I, Q, M, D, L or constant	Input NR contains the number of the runtime meter you want to start or stop. The numbering starts with "0". For information on the number of runtime meters of your CPU, refer to the technical data.
S	Input	BOOL	I, Q, M, D, L	Input S starts or stops the runtime meter. • Set the signal state to "0" when you want to stop the runtime meter. • Set the signal state to "1" when you want to start the runtime meter.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the runtime meter
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

READ_RTM: Read runtime meters

Description

You use the instruction to read a runtime meter. The instruction provides the current number of operating hours as output data and the status of the counter, that is, "stopped" or "counting".

If the runtime meter runs for longer than 32767 hours, it stops at the count 32767 and outputs the "Overflow" error message.

Parameter

The following table shows the parameters of the "READ_RTM" instruction:

Parameters	Declaration	Data type	Memory area	Description
NR	Input	BYTE	I, Q, M, D, L, P or constant	Input NR contains the number of the runtime meter you want to start or stop. The numbering starts with "0". For information on the number of runtime meters of your CPU, refer to the technical data.
RET_VAL	Return	INT	I, Q, M, D, L, P	If an error occurs while the instruction is being executed, the return value contains an error code.
CQ	Output	BOOL	I, Q, M, D, L	Output CQ indicates whether the runtime meter is running or stopped. The signal state "0" shows that the runtime meter is stopped. Signal state "1" shows that the runtime meter is running.
CV	Output	INT	I, Q, M, D, L, P	Output CV indicates the current value of the runtime meter.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the runtime meter
8081	Overflow of the runtime meter
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Local time

LOC_TIME: Calculate local time

Description

The instruction reads the time-of-day status and time-of-day of the CPU and calculates the local time from this information. It only works with S7-400 CPUs. The instruction can be called in OBs of every priority class.

Functional description

Internally, "LOC_TIME" uses the " RDSYSST (Page 5444)" instruction to read out the module time, the correction value with sign and the daylight-saving time ID from the current time-of-day status of the CPU. "RDSYSST (Page 5444)" errors terminate the block. The error codes are passed to RET_VAL .

The correction value is the number of half-hours between the basic time and local time. It is also identified as daylight-saving or standard time.

If the CPU does not have a valid time status, "LOC_TIME" is terminated with an error.

After successfully reading out the time-of-day status, the time difference based on the correction value is added to the module time and applied to output LT .

The SUMMER output adopts the value of the daylight saving/standard time bit from the time-of-day status.

Parameters

The following table shows the parameters of the "LOC_TIME" instruction:

Parameters	Declaration	Data type	Description
RET_VAL	Return	INT	Error code
LT	Output	DATE_AND_TIME	Local time (Local Time)
SUMMER	Output	BOOL	Daylight saving time identification • 0: Winter • 1: Summer

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameters RET_VAL, LT and SUMMER

The following table shows the output values and the errors of "LOC_TIME":

RET_VAL	LT	SUMMER	Description
0	Local time	TRUE/FALSE	Instruction executed without errors
1	Local time	TRUE/FALSE	No error, but jump in date
8xyy	DT#90-01-01-0:0:0	FALSE	General error codes of "RDSYSST (Page 5444)"
8091	DT#90-01-01-0:0:0	FALSE	No time status

BT_LT: Calculate local time from base time

Description

The instruction calculates the local time for the base time set at the input.

The base time entered at the BT input is converted to the local time based on data stored in a data block (DB) and output at the LT output.

The data block (DB) contains the number of 30 minute units by which the base and local time differ and the difference between daylight saving and standard time also in units of 30 minutes.

If there is a date overflow in the conversion, this is indicated by a special return value.

The "BT_LT" instruction can be called in OBs of every priority class.

Parameter

The following table shows the parameters of the instruction "BT_LT":

Parameter	Declaration	Data type	Description
BT	Input	DATE_AND_TIME	Base time
WS_DAT	Input	BLOCK_DB	Information on the time zone and daylight saving/standard time adjustment (rule DB) At parameter WS_DAT, use a pointer to a data block of type "WS_RULES (Page 5204)".
RET_VAL	Return	INT	Error code
LT	Output	DATE_AND_TIME	Local time

For more information about the valid data types, refer to "Overview of the valid data types (Page 231)".

Note

Parameters of data type BLOCK_DB can only be transferred to FB calls, but not to FC calls.

Parameters RET_VAL and LT

The following table shows the output values of "BT_LT":

RET_VAL	LT	Description
0	Local time	Instruction executed without errors
1	Local time	No error, but jump in date
8082	DT#90-01-01-0:0:0	Invalid data in the rule data block

LT_BT: Calculate base time from local time

Description

The instruction calculates the base time for the local time set at the input.

The local time entered at input LT is converted to the base time with the help of data stored in a data block (DB) and the result returned at output BT.

The data block (DB) contains the number of 30 minute units by which the base and local time differ and the difference between daylight saving and standard time also in units of 30 minutes. If there is a date overflow in the conversion, this is indicated by a special return value.

Special considerations when switching between daylight saving and standard time:

- At the changeover from standard to daylight saving time, the local time is put forward one hour. This means that the hour in between does not exist.
  At an instant LT within this hour, "LT_BT" "thinks" in daylight saving time. The result is indicated by the return value 4 or 5.
- At the changeover from daylight saving to standard time, the local time is put back one hour. This means that this one hour is passed through twice ("duplicated hour"). (For CE(S)T, the identifiers 2A and 2B are used).
  This prevents unambiguous mapping to a base time at an instant LT of this hour.
  "LT_BT" receives an LT as input parameter and must determine whether the value is a daylight saving or standard time before it converts it to BT. If LT is within the duplicated hour, the LT is interpreted as standard time. This result is indicated by the return value 2 or 3.

The "LT_BT" instruction can be called in OBs of every priority class.

Parameter

The following table shows the parameters of the instruction "LT_BT":

Parameter	Declaration	Data type	Description
LT	Input	DATE_AND_TIME	Local time
WS_DAT	Input	BLOCK_DB	Information on the time zone and daylight saving/standard time adjustment (rule DB) At parameter WS_DAT, use a pointer to a data block of type "WS_RULES (Page 5204)".
RET_VAL	Return	INT	Error code
BT	Output	DATE_AND_TIME	Base time

For more information about the valid data types, refer to "Overview of the valid data types (Page 231)".

Note

Parameters of data type BLOCK_DB can only be transferred to FB calls, but not to FC calls.

Parameters RET_VAL and BT

The following table shows the output values of "LT_BT":

RET_VAL	BT	Description
0	Base time	Instruction executed without errors
1	Base time	No error, but jump in date
2	Base time	The LT at the input is within the "duplicated" hour.
3	Base time	As for 2, but with additional date jump
4	Base time	The LT at the input is within the "prohibited" hour.
5	Base time	As for 4, but with additional date jump
8082	DT#90-01-01-0:0:0	Invalid data in the rule data block

S_LTINT: Set time-of-day interrupt using local time

Description

The instruction generates the required time-of-day interrupt at the set instant. This time is set in local time.

The local time entered at input LT is converted to the base time using the rule that is stored in a data block (DB).

The data block (DB) contains the number of 30 minute units by which the base and local time differ and the difference between daylight saving and standard time also in units of 30 minutes. The parameters of the specified time-of-day interrupt OB are set and the OB activated with the calculated base time. If there is a date overflow in the conversion, this is indicated by a special return value.

Special considerations when switching between daylight saving and standard time:

- At the changeover from standard to daylight saving time, the local time is put forward one hour. This means that the hour in between does not exist.
  At an instant LT within this hour, "S_LTINT" calculates the time in daylight saving format. This process is indicated by the return value (RET_VAL) 4 or 5.
- At the changeover from daylight saving to standard time, the local time is put back one hour. This means that this one hour is passed through twice ("duplicated hour"). (For CE(S)T, the identifiers 2A and 2B are used).
  This prevents unambiguous mapping to a base time at an instant LT of this hour. "S_LTINT" receives an LT as input parameter and must determine whether the value is a daylight saving or standard time before it converts it to BT. If LT is within the duplicated hour, the LT is interpreted as standard time. This process is indicated by the return value (RET_VAL) 2 or 3.

The "S_LTINT" instruction can be called in OBs of every priority class.

Parameter

The following table shows the parameters of the instruction "S_LTINT":

Parameter	Declaration	Data type	Description
OB_NR	Input	INT	Number of the OB to be started (permitted 10 – 17)
SDT	Input	DATE_AND_TIME	Start date and time-of-day in local time See also: "SET_TINT (Page 5369)"
PERIOD	Input	WORD	Periods from start point SDT onwards: • W#16#0000 = once • W#16#0201 = once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end

Parameter	Declaration	Data type	Description
WS_DAT	Input	BLOCK_DB	Information on the time zone and daylight saving/standard time adjustment (rule DB) At parameter WS_DAT, use a pointer to a data block of type "WS_RULES (Page 5204)".
RET_VAL	Return	INT	Error code

For more information about the valid data types, refer to "Overview of the valid data types (Page 231)".

Note

Parameters of data type BLOCK_DB can only be transferred to FB calls, but not to FC calls.

Parameters RET_VAL

RET_VAL	Description
0	Instruction executed without errors
1	No error, but jump in date
2	The LT at the input was within the "duplicated" hour.
3	As for 2, but with additional date jump
4	The LT at the input is within the "prohibited" hour.
5	As for 4, but with additional date jump
8082	Invalid data in the rule data block
8090	Incorrect parameter OB_NR
8091	Incorrect parameter SDT
8092	Incorrect parameter PERIOD
80A1	The set start time is in the past.
80A2	OB is not loaded
80A3	OB cannot be started

SET_SW: Set daylight saving time/standard time without time-of-day status

Description

The instruction supports the switch from daylight-saving time to standard time in CPUs that are not equipped with time-of-day status. The instruction "SET_SW" adjusts the CPU clock according to the current time and the adjustment rules in the rule DB.

"SET_SW" is called in the organization block OB1 and in a time-of-day interrupt OB once with the same instance. In the time-of-day interrupt OB, the parameters of the "SET_SW" instruction are not connected. The data block (DB) that contains the rule for the start and end of daylight-saving time is specified at input WS_DAT.

The following figure shows the time chart on which the function of "SET_SW" is based:

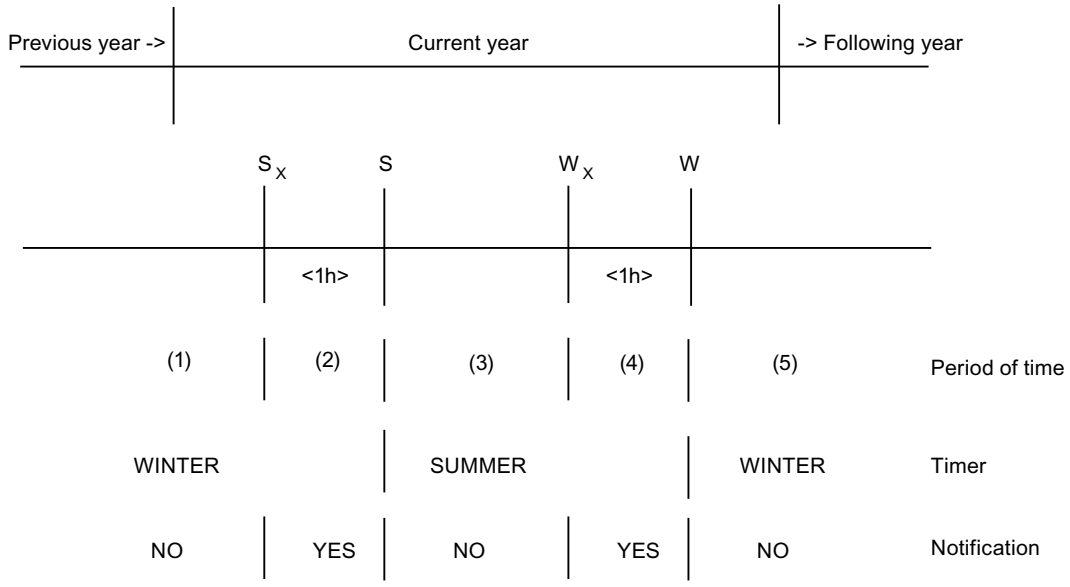

There are four adjustment times (S_A, S, W_A, W) defined. Their precise position is derived from the rules in the rule DB. The notification hours are one hour before the adjustment times and signal the pending adjustment.

A positive edge at the input REQ triggers initialization and activation of the daylight-saving/standard time adjustment. This setting remains valid as long as REQ = 1.

For the time adjustment, "SET_SW" sets the parameters and activates the specified time-of-day interrupt OB with the next changeover time in the mode "once". The time-of-day interrupt OB then takes over control. When it becomes due, it changes the time on the CPU according to the current call time and sets parameters and activates itself for the next changeover time.

If the current time is in segment (3), the time-of-day interrupt OB is set to the start time W_A ("standard time notification").

On a falling edge at REQ, the daylight-saving/standard time adjustment is disabled.

Parameters

The following table shows the parameters of the instruction "SET_SW":

Parameters	Declaration	Data type	Description
REQ	Input	BOOL	Control of the changeover
WS_DAT	Input	BLOCK_DB	Information on the time zone and daylight-saving/standard time adjustment (rule DB) At parameter WS_DAT, use a pointer to a data block of type "WS_RULES (Page 5204)".
OB_NR	Input	INT	Number of the time-of-day interrupt to be used
STATUS	Output	INT	Error code
ISSUMMER	Output	BOOL	1 = Daylight saving time
ANN_1	Output	BOOL	1 = notification of a changeover

For more information about the valid data types, refer to "Overview of the valid data types (Page 231)".

Note

Parameters of data type BLOCK_DB can only be transferred to FB calls, but not to FC calls.

Parameters STATUS

STATUS	Meaning
0	No error
8xyy	General error codes of "RDSYSST (Page 5444)"
8082	Invalid data in WS_DAT
8083	The month number for the change to standard time is higher than the month number for the change to daylight-saving time and notification hour for standard time is earlier than the hour for daylight-saving or the month number for the change to standard time is lower than the month number for the change to daylight-saving time and the notification hour for daylight-saving is earlier than the hour for standard time.
8090	Illegal OB_NR. (Only time-of-day interrupts permitted: OB10 – OB17, depending on the CPU)
80A2	Illegal OB_NR: The OB is not loaded on the CPU
80A3	The OB cannot be started. See also: "QRY_TINT (Page 5371)"

Application

"SET_SW" can be used in CPUs that are not equipped with time-of-day status. CPUs with time-of-day status use "SET_SW_S (Page 5197)".

Calling OBs

"SET_SW" must be called with the same instance in the following organization blocks (OBs):

- **OB1 (cyclic program):**
  "SET_SW" evaluates the REQ input. Based on the current time, the OB it sets the start of the specified time-of-day interrupt to the next of four possible changeover times and activates the time-of-day interrupt with the 'once' identifier at the positive edge at REQ. The time-of-day interrupt is triggered when this time is reached.
- **OB1x (time-of-day interrupt):**
  OB1x became active because a changeover time was reached. "SET_SW" sets the time on the CPU according to the current time (daylight-saving time!), sets its parameters to the next changeover time and activates it. In segment (5), it sets itself to the daylight-saving time notification (S_A) of the next year.

"Multiple instances of SET_SW" are not permitted.

You can use the following instructions to query, set and activate the time-of-day interrupt: "QRY_TINT (Page 5371)", "SET_TINT (Page 5369)", "ACT_TINT (Page 5371)", "CAN_TINT (Page 5370)"

Note

Adjusting the CPU time, for example from a PG, when "SET_SW" (REQ = 1) is activated is only permitted without special measures if this does not result in the time being adjusted past one of the changeover points.

If you missed out on one of the changeover points while adjusting the clock, you have to disable daylight-saving/standard time adjustment with REQ = 0 and then restart it with REQ = 1 after having completed the adjustment.

When you set the CPU clock, you must, of course, make sure that you set the daylight-saving time hour in summer.

SET_SW_S: Set daylight saving time/standard time with time-of-day status

Description

The instruction supports the switch from daylight-saving time to standard time in CPUs that are equipped with time-of-day status. The instruction "SET_SW_S" adjusts the CPU clock according to the current time and the adjustment rules in the rule DB.

"SET_SW_S" is called in the organization block OB1 and in a time-of-day interrupt OB 1x with the same instance. In the time-of-day interrupt OB, the parameters of the "SET_SW_S" instruction are not connected. The data block (DB) that contains the rule for the start and end of daylight-saving time is specified at input WS_DAT.

The following figure shows the time chart on which the function of "SET_SW_S" is based:

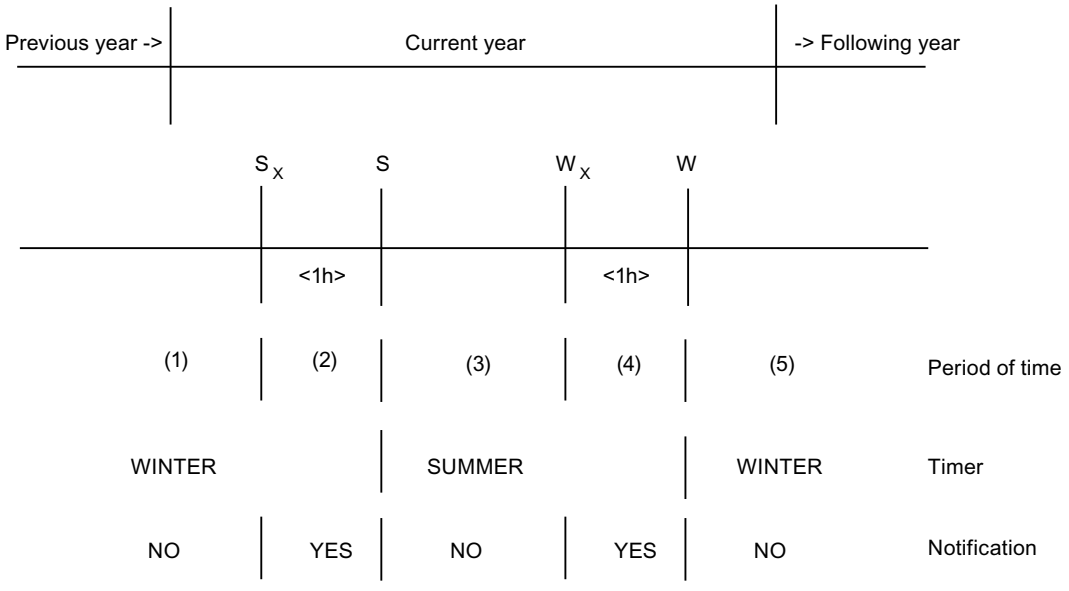

There are four adjustment times (S_A, S, W_A, W) defined. Their precise position is derived from the rules in the rule DB. The notification hours are 1 hour before the adjustment times and signal the pending adjustment.

A positive edge at the input REQ triggers initialization and activation of the daylight-saving/ standard time adjustment. This setting remains valid as long as REQ = 1.

For the time adjustment, "SET_SW_S" sets the parameters and activates the specified time-of-day interrupt OB with the next changeover time in the mode "once". The time-of-day interrupt OB then takes over control. When it becomes due, it changes the time and the time status on the CPU according to the current call time and sets its parameters and activates itself for the next changeover time.

Example: If the current time is in segment (3), the time-of-day interrupt OB is set to the start time WA ("standard time notification").

On a falling edge at REQ, the daylight-saving/standard time adjustment is disabled.

Parameters

The following table shows the parameters of the instruction "SET_SW_S":

Parameters	Declaration	Data type	Description
REQ	Input	BOOL	Control of the changeover
WS_DAT	Input	BLOCK_DB	Information on the time zone and daylight-saving/standard time adjustment (rule DB) At parameter WS_DAT, use a pointer to a data block of type "WS_RULES (Page 5204)".
OB_NR	Input	INT	Number of the time-of-day interrupt to be used
STATUS	Output	INT	Error code
ISSUMMER	Output	BOOL	It is summer (DST)
ANN_1	Output	BOOL	Notification of the adjustment

For more information about the valid data types, refer to "Overview of the valid data types (Page 231)".

Note

Parameters of data type BLOCK_DB can only be transferred to FB calls, but not to FC calls.

Parameter STATUS

STATUS	Meaning
0	No error
8xyy	General error codes of "RDSYSST (Page 5444)"
8082	Invalid data in WS_DAT
8083	The month number for the change to standard time is higher than the month number for the change to daylight-saving time and notification hour for standard time is earlier than the hour for daylight-saving or the month number for the change to standard time is lower than the month number for the change to daylight-saving time and the notification hour for daylight-saving is earlier than the hour for standard time.
8090	Illegal OB_NR. (Only time-of-day interrupts permitted: OB10 – OB17, depending on the CPU)
8091	Time status invalid ('status_valid' in time status set to FALSE)
80A2	Illegal OB_NR: The OB is not loaded on the CPU
80A3	The OB cannot be started. (see "QRY_TINT (Page 5371)")

Application

"SET_SW_S" can be used in CPUs that are equipped with time-of-day status.

Calling OBs

"SET_SW_S" must be called with the same instance in the following organization blocks (OBs):

- **OB1 (cyclic program):**
  "SET_SW_S" evaluates the REQ input. Based on the current time, the OB it sets the start of the specified time-of-day interrupt to the next of four possible changeover times and activates the time-of-day interrupt with the 'once' identifier at the positive edge at REQ. The time-of-day interrupt is therefore triggered when this time is reached.
- **OB1x (time-of-day interrupt):**
  OB1x became active because a changeover time was reached. "SET_SW_S" sets the time-of-day and the time-of-day status on the CPU according to the current time, sets its parameters to the next changeover time and activates it. In segment (5), it sets itself to the daylight-saving notification of the next year.

"Multiple instances of SET_SW_S" are not permitted.

You can use the following instructions to query, set and activate the time-of-day interrupt: "QRY_TINT (Page 5371)", "SET_TINT (Page 5369)", "ACT_TINT (Page 5371)", "CAN_TINT (Page 5370)"

Note

Adjusting the CPU time, for example from a PG, when "SET_SW_S" (REQ = 1) is activated is only permitted without special measures if this does not result in the time being adjusted past one of the changeover points.

If you missed out on one of the changeover points while adjusting the clock, you have to disable daylight-saving/standard time adjustment with REQ = 0 and then restart it with REQ = 1 after having completed the adjustment.

When setting the CPU clock and the time status, make sure that you set the daylight-saving time identifier or the notification hour identifier depending on whether you are setting daylight-saving or standard time.

TIMESTMP: Transfer time-stamped alarms

Description

The instruction transmits messages with time stamp of an IM153-2 to its instance DB. The data is available there for further processing by the user program.

- IM153-2
  The IM153-2 detects changes in binary signals, adds a time stamp to them and stores this information in data records with a maximum of 20 messages. With certain events, it outputs special messages. When a data record is ready to be fetched, the IM153-2 triggers a process interrupt.
- Call in the hardware interrupt OB
  "TIMESTMP" reads the start information of the process interrupt OB and stores the information relating to the time stamping in a circulating buffer for further processing. "TIMESTMP" can save the data of up to 17 process interrupts.
- Call in the cyclic OB:
  During cyclic processing, "TIMESTMP" reads the data record to be fetched using the instruction "RD_REC (Page 5256)" (read data record) and stores it in the message buffer MESSAGE. If there are several data records to be fetched, the oldest is read first. If the TIMECONV bit is set at the input, "TIMESTMP" converts all time stamps from the ISP format to the DATE_AND_TIME (DT) format.
  If the parameter is BUFRDY = TRUE, the messages in the message buffer can be further processed by the user program and, for example, transferred to a recipient (HMI device or printer). The number of existing messages is indicated by the MSG_QTY parameter. After processing, the user program must reset BUFRDY to release the message buffer for further messages.
  The BUFNOTREAD parameter indicates the number of received process interrupts for which "TIMESTMP" has not yet read data records from the IM153-2 . With BUFNOTREAD = 15 all data records of the IM153-2 are filled. The IM153-2 will then recognize no additional signal changes. Additional data records that are generated by the IM153-2 at a change of binary signals are lost. The MSGLOST parameter (message loss) is set to "1".

Parameter

The following table shows the parameters of the instruction "TIMESTMP":

Parameter	Declaration	Data type	Description
LADDR	Input	INT	Logical address (diagnostic address) of the IM153-2
LADDR2	Input	INT	Logical address (diagnostic address) of slot 2 of the IM153-2
TIMECONV	Input	BOOL	1 = convert ISP time stamp to S7-DT format
MSG_QTY	Output	INT	Number of valid message in the message buffer
BUFNOTREAD	Output	INT	Number of data records of the IM153-2 containing interrupts but not yet read
READERR	Output	BOOL	1 = error reading from the IM153-2
READSTATUS	Output	INT	Return value of instruction "RD_REC (Page 5256)"
BUFRDY	InOut	BOOL	1 = message buffer ready
MSGLOST	InOut	BOOL	1 = messages lost

Parameter	Declaration	Data type	Description
MESSAGE	Static	ARRAY[1..20] OF STRUCT	Message buffer for up to 20 messages of the IM153-2
SLOT_NO	./.	BYTE	Slot number / 2 = special message
CH_NO	./.	BYTE	Channel number / ID of the special message
SIGNAL	./.	BYTE	Signal state / Characteristic of the special message
TIME1	./.	DWORD	Seconds starting from 1900.0 (ISP)/Year, Month, Day, Hour ((DT)
TIME2	./.	DWORD	Fractions of seconds (ISP) / Minute, Second, Millisecond, Day of week (DT)

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Hardware configuration

You can only use "TIMESTMP" in configurations that support forwarding of the time-of-day and time stamping of binary signals. These are:

- CPUs
  - CPU 4xx with CP443-5 for forwarding time frames (6GK7443-5DX02-0XE0 and higher), or
  - CPU 4xx with forwarding of time frames (for example, CPU416-2 DP as of firmware version 3.0)
- Distributed I/O
  - Interface module IM153-2 with redundancy capability with time stamping (6ES7153-2AA02 and higher, firmware version V1.2.3 (version 10))
    Or:
    IM153-2 FO with redundancy capability with integrated FO interface (6ES7153-2AB01 and higher, firmware version V1.2.3 and higher (version 9))
  - Digital input SM321 (6ES7321-7BH00-0AB0) isolated 16 DI, 24 V DC, hardware interrupt, diagnostics
    Or:
    Digital input SM321 (6ES7321-7TH00-0AB0) isolated 16 DI, 24 V DC, NAMUR, diagnostic capability, with process control functions
    You should only use time stamping for selected binary signals that are important in your application. It is advisable to distribute these signals over several IM153-2 or over several stations due to the load on PROFIBUS-DP and the IM153-2.

Software configuration

- Calling OBs
  "TIMESTMP" must be called with the same instance in the following organization blocks (OBs) and the block parameters must be written and read only in the cyclic program:
  - OB1 (cyclic program) or alternatively a cyclic interrupt OB (OB30 to OB38)
  - OB86 (rack failure)
  - OB100 (startup)
    If you use other startup OBs (OB101, OB102), "TIMESTMP" must be called there as well.
  - OB40 or, if selectable in the hardware configuration, also in other process interrupt OBs (OB41 to OB47)
- Addressing
  - Use a separate instance of "TIMESTMP" for each IM153-2. At the LADDR block input, enter the logical address (diagnostic address) set in the hardware configuration of the IM153-2.
  - If you use the DP master in the "S7-compatible" mode, enter the same value at input LADDR2 as for LADDR. If you use the "DPV1" mode, enter the diagnostic address of slot 2 of the IM153-2 for LADDR2 .
  - If you change addresses during operation, "TIMESTMP" deletes the previously stored process interrupt data.

Error handling

If "TIMESTMP" detects unrecoverable errors when reading a data record, "TIMESTMP" sets the READERR parameter to TRUE. The return value of the WR_USMSG instruction is available for further analysis in the READSTATUS parameter. Because "TIMESTMP" possibly reads another data record in the next cycle, READERR and READSTATUS are valid for only one cycle. You should therefore include suitable evaluation in the user program.

If data records are read while the DP master is out of service, "TIMESTMP" will output READERR = TRUE and READSTATUS =80B2h ("The configured slot is not in use").

Startup characteristics IM153-2

During startup/warm restart/cold restart, the IM153-2 once again outputs process interrupts for the data records; these interrupts had entries before the startup but these have not been fetched yet.

The IM153-2 enters the following messages in the first free data record:

- Special message "Begin startup data"
- Signal changes that occurred immediately prior to a CPU STOP ("incoming" or "outgoing" according to the parameter setting for the edge)
- Current signal status for all binary signals to be time stamped ("incoming" or "outgoing" according to the parameter setting for the edge)
- Special message "End startup data"

Startup characteristics of "TIMESTMP"

If "TIMESTMP" is called in a startup OB (OB100, OB101, OB102), "TIMESTMP" deletes all stored process interrupt data and resets BUFRDY . If data records with interrupts are known but not yet read, "TIMESTMP" sets MSGLOST (message loss) to TRUE. The user program must reset MSGLOST .

Response of "TIMESTMP" in OB86 (module failure)

If "TIMESTMP" is called in OB86 and if the event comes from the corresponding IM153-2, "TIMESTMP" reacts as during startup. This is, for example, the case when the IM153-2 fails and returns to operation and when the DP master returns to operation.

Redundancy

In H systems with two IM153-2 modules, time stamping is redundant in the following situations:

- There is communication between the two IM153-2 modules via the K bus.
- The updating of the active and passive IM153-2 modules has been completed without error.

Time stamping is interrupted during a failover between active and passive IM153-2 modules. The time for which time stamping is interrupted is indicated with the special message "Redundancy failover start/end".

Normally, the active IM153-2 module informs the passive one of the current I/O status. If this communication is disrupted, the special message "Loss of redundancy information incoming" is output. As soon as communication is possible again between the active and passive IM153-2 modules, the special message "Loss of redundancy information outgoing" is output. Later resynchronization is not possible, in other words, a failover in the case of the IM153-2 causes a loss of messages.

Structure of the message buffer

The message buffer contains a data record of the IM153-2. The interpretation of the time stamp depends on the setting of the TIMECONV parameter.

Time formats of "TIMESTMP"

If the input is TIMECONV = TRUE , all time stamps are stored in the data type DATE_AND_TIME (BCD format):

The following table shows the structure of the data type DATE_AND_TIME:

Bytes	Contents	Range
0	Year	90 … 89 Corresponding to 1990 ... 2089
1	Month	01 ... 12
2	Day	01 ... 31
3	Hour	00 ... 23
4	Minute	00 ... 59
5	Second	00 ... 59

Bytes	Contents	Range
6	2 MSD (most significant decade) of ms	00 ... 99
7 (4 MSB)	LSD (least significant decade) of ms	0 ... 9
7 (4 LSB)	Day of week	1 ... 7 (1 = Sunday)

If the input is TIMECONV = FALSE , all time stamps are stored in ISP format (2 double words).

The following table shows the distribution of the bytes:

Bytes	Contents	Range
0 … 3	Seconds starting from 1.1.1900 0:00:00.000	Corresponding to 1.1.1900 ... 6.2.2036
4 … 7	Fractions of seconds in multiples of $1/2^{32}$ s	0 … <1

The time conversion is valid in the date range covered by both time formats 1.1.1990 up to and including 6.2.2036.

WS_RULES: Rule DB for converting the base time to local time

Data block of "WS_RULES" type

A data block of the "WS_RULES" type has the following structure.

Name		Data type	Start value	Description
// Conversion base time<->local time and "Set interrupt according to local time"				
B2L		Struct		
	S	INT	2	// Offset base->local time [30 min] in winter, permitted -24 ... +24.
	T	INT	2	// Difference standard and daylight-saving time [30 min], permitted: 2
// Rule for: Standard time -> daylight-saving time. Default: Last Sunday in March; 2:00 a.m.				
W2S		Struct		// in STANDARD TIME
	M	BYTE	B#16#3	// Month of conversion
	W	BYTE	B#16#9	// n-th occurrence of the day of the week (1=first, 2=second,..,9=last)
	D	BYTE	B#16#1	// Day of the week (Sunday = 1)
	H	BYTE	B#16#2	// Hour
// Rule for: Daylight-saving time -> standard time. Default: Last Sunday in October; 3:00 a.m.				
S2W				// in DAYLIGHT-SAVING TIME
	M	BYTE	B#16#10	// Month of conversion
	W	BYTE	B#16#9	// n-th occurrence of the day of the week (9 = last)
	D	BYTE	B#16#1	// Day of the week (Sunday = 1)
	H	BYTE	B#16#3	// Hour

Note

Create data block of "WS_RULES" type

You can create a data block of type "WS_RULES", by selecting type "WS_RULES" during the creation of a new DB.

Note

All parameters that have the BYTE format are interpreted as BCD values (binary-coded decimal value). W2S is output in standard time; S2W in daylight-saving time. The defining of the daylight-saving/standard time changeover times by a rule has been prescribed in the EU since 2002.

4.2.3.2 String + Char

S_COMP: Compare character strings

Description

The instruction compares the contents of two tags in the STRING format and outputs the result of the comparison as a return value. The tags that are to be compared will be interconnected at the IN1 and IN2 inputs. You can only assign a symbolically defined tag for the input parameters.

Use the instruction box to select the comparison condition. If the comparison condition (for example, greater or equal) is satisfied, the signal state is set to "1" on the output parameter OUT .

The following comparison options can be used:

Symbol	Description
EQ	The return value has the signal state "1" if the string at the IN1 parameter is the same as the string at the IN2 parameter.
NE	The return value has the signal state "1" if the string at the IN1 parameter is not equal to the string at the IN2 parameter.
GT (1)	The return value has the signal state “1" if the string at the IN1 parameter is greater than the string at the IN2 parameter.
LT (1)	The return value has the signal state "1" if the string at the IN1 parameter is less than the string at the IN2 parameter.
GE (1)	The return value has the signal state "1" if the string at the IN1 parameter is greater than or equal to the string at the IN2 parameter.
LE (1)	The return value has the signal state "1" if the string at the IN1 parameter is less than or equal to the string at the IN2 parameter.
(1) The characters are compared by their ASCII code (for example, 'a' is greater than 'A'), starting from the left. The first character to be different decides the result of the comparison. If the first characters are the same, the longer string is greater.	

Parameters

The following table shows the parameters of the instruction "S_COMP":

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L	Input tag in the STRING format.
IN2	Input	STRING*	D, L	Input tag in the STRING format.
OUT	Output	BOOL	I, Q, M, D, L	Result of comparison
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

S_CONV: Convert character string

Description

You use this instruction to convert the value at the IN input to the data format you have specified in the OUT output. You can use the instruction to convert character strings to integers and floating-point numbers or to convert integers and floating-point numbers to character strings.

Note

Explicit conversions in SCL

You can find additional information and SCL-specific features for explicit conversions here:

- Explicit conversions (Page 569)

Converting a character string to integers and floating-point numbers

Following conversions of a character string to integers and floating-point numbers are possible:

- Conversion of a character string to a tag in INT format. The first character in the string may be a sign or a number, the characters which then follow must be numbers. If the length of the string is equal to zero or greater than 6, or if invalid characters are found in the string, no conversion takes place and the binary result (BR) bit of the status word is set to "0". If the result of the conversion is outside the INT range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".
- Conversion of a character string to a tag in DINT format. The first character in the string may be a sign or a number, the characters which then follow must be numbers. If the length of the string is equal to zero or greater than 11, or if invalid characters are found in the string, no conversion takes place and the binary result (BR) bit of the status word is set to "0". If the result of the conversion is outside the DINT range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".
- Conversion of a character string to a tag in REAL format. The string must have the following format: ±v.nnnnnnnE±xx
  - ± = sign
  - v = 1 digit before the decimal point
  - n = 7 digits after the decimal point
  - x = 2 exponential digits

  If the length of the string is less than 14, or if it is not structured as shown above, no conversion takes place and the binary result (BR) bit of the status word is set to "0". If the result of the conversion is outside the REAL range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

Converting an integer and floating-point number to a character string

The following conversions of integers and floating-point numbers to a character string are possible:

- Conversion of a tag in INT format to a character string. The character string is shown preceded by a sign. If the tag specified at the return parameter is too short, no conversion will take place and the binary result (BR) bit of the status word is set to "0". You can only assign a symbolically defined tag for the output parameter.
- Conversion of a tag in DINT format to a character string. The character string is shown preceded by a sign. If the tag specified at the return parameter is too short, no conversion will take place and the binary result (BR) bit of the status word is set to "0". You can only assign a symbolically defined tag for the output parameter.
- Conversion of a tag in REAL format to a character string. The string is shown with 14 digits: ±v.nnnnnnnE±xx
  - ± = sign
  - v = 1 digit before the decimal point
  - n = 7 digits after the decimal point
  - x = 2 exponential digits

  If the tag specified at the return parameter is too short or if no valid floating-point number is specified at the INparameter, no conversion will take place and the binary result (BR) bit of the status word is set to "0". You can only assign a symbolically defined tag for the output parameter.

Parameters

The following table shows the parameters of the "S_CONV" instruction:

Table 4-189 Parameters for converting a character string to integers and floating-point numbers

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L	Value to be converted
OUT	Output	INT, DINT, REAL	I, Q, M, D, L	Result of the conversion.
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

Table 4-190 Parameters for converting an integer and floating-point number to a character string

Parameters	Declaration	Data type	Memory area	Description
IN	Input	INT, DINT, REAL	I, Q, M, D, L or constant	Value to be converted
OUT	Output	STRING*	D, L	Result of the conversion
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ATH: Convert ASCII string to hexadecimal number

Description

You use the instruction to convert the ASCII string specified at the IN input to a hexadecimal number. The result of the conversion is stored at the address specified by the OUT parameter.

- You specify the number of ASCII characters to be converted with the N parameter.
- A maximum of 65 535 valid ASCII characters can be converted.
- Only the numbers 0 to 9 and upper case letters A to F can be interpreted. All other characters are converted to zeros.

Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output word length is only half of the input word length. The ASCII characters are converted and positioned in the output in the same order as they are read in. If there is an odd number of ASCII characters, the hexadecimal number is padded with zeros in the nibble to the right of the last converted hexadecimal number.

If an invalid ASCII character is detected, it is converted as "0" and an error message is output to the RET_VAL parameter.

Parameter

The following table shows the parameters of the "ATH" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	POINTER	D, L	Pointer to the start address of an ASCII string
N	Input	WORD	I, Q, M, D, L, P or constant	Number of ASCII characters to be converted
RET_VAL	Output	WORD	I, Q, M, D, L, P	Status of the instruction
OUT	Output	POINTER	I, Q, M, D, L	Pointer to the address of the result
* Pointer in double-word format for area-crossing, indirect register addressing				

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#....)	Description
0000	No error
0007	Invalid character

Example

The instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, the N input parameter is 5 indicating that five ASCII characters are to be converted. The ASCII characters are stored in data block 1 beginning at the address specified by the IN pointer: DB1.DBX10.0. The character string is output at the address specified by the OUT pointer: Start at DB2.DBX0.0 (data block 2). Since there is an odd number of ASCII characters, the last hexadecimal digit contains only zeros in the right nibble, resulting in the hexadecimal value 0xC0.

If the instruction was executed without errors, RET_VAL will be set to the value W#16#0000.

The following table shows examples of the conversion of ASCII character strings to hexadecimal numbers.

IN	N	OUT	BR status
'0123'	4	16#0123	1
'123AFx1a23'	10	16#123AF01023	0

Table 4-191 The following table shows the ASCII characters and the corresponding hexadecimal values.

ASCII character	ASCII-coded hexadecimal value	Hexadecimal digit
"0"	30	0
"1"	31	1
"2"	32	2
"3"	33	3
"4"	34	4
"5"	35	5
"6"	36	6
"7"	37	7
"8"	38	8
"9"	39	9
"A"	41	A
"B"	42	B
"C"	43	C
D	44	D
E	45	E
F	46	F

HTA: Convert hexadecimal number to ASCII string

Description

You use the instruction to convert the hexadecimal number specified at the IN input to an ASCII string. The result of the conversion is stored at the address specified by the OUT parameter.

You specify the number of hexadecimal bytes to be converted with the N parameter. Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output value is twice as long as the input value. Each nibble of the hexadecimal number is converted to a character while maintaining the original order.

A maximum of 65 635 characters can be written to the ASCII character string. The result of the conversion is represented by the digits 0 to 9 and upper-case letters A to F.

The instruction does not detect any error conditions.

Parameter

The following table shows the parameters of the "HTA" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	POINTER*	I, Q, M, D	Start address of the hexadecimal digits
N	Input	WORD	I, Q, M, D, L, P or constant	Number of hexadecimal bytes to be converted
OUT	Output	POINTER*	D, L	Address at which the result is stored.
* Pointer in double-word format for area-crossing, indirect register addressing				

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Example

The instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, the N input parameter is set to 3 indicating that three hexadecimal characters are to be converted. The hexadecimal bytes are stored in data block 1 beginning at the address specified by the IN pointer: DB1.DBX10.0. The character string is output at the address specified by the OUT pointer: Start at DB2.DBX0.0 (data block 2).

The following table shows examples of the conversion of hexadecimal numbers to ASCII character strings:

IN	N	OUT	BR status
W#16#0123	2	'0123'	1
16#123AF01023	4	'123AF010'	0

The following table shows the ASCII characters and the corresponding hexadecimal values:

Hexadecimal digit	ASCII-coded hexadecimal value	ASCII character
0	30	"0"
1	31	"1"
2	32	"2"
3	33	"3"
4	34	"4"
5	35	"5"
6	36	"6"
7	37	"7"

Hexadecimal digit	ASCII-coded hexadecimal value	ASCII character
8	38	"8"
9	39	"9"
A	41	"A"
B	42	"B"
C	43	"C"
D	44	"D"
E	45	"E"
F	46	"F"

Other instructions

LEN: Determine the length of a character string

Description

The instruction outputs the current length of a character string (number of valid characters) as return value.

A STRING tag contains two lengths:

- The maximum length (which is specified in square brackets when the tag is defined)
- The current length (this is the number of currently valid characters)

The current length must be smaller than or equal to the maximum length. The number of bytes occupied by a string is 2 greater than the maximum length. A blank string (' ') has the length zero. The maximum length is 254. The instruction does not report any errors.

Parameters

The following table shows the parameters of the instruction "LEN":

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L* *	Input tag in the format STRING; you can only assign a symbolically defined tag for the input parameter.
OUT	Return	INT	I, Q, M, D, L	Number of valid characters
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type). ** With SCL: Constants may also be used.				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

CONCAT: Combine character strings

Description

The instruction combines two STRING tags into one character string. If the result string is longer than the tag given at the output parameter, the result string is limited to the maximum set length and the binary result (BR) bit of the status word set to "0".

Parameters

The following table shows the parameters of the instruction "CONCAT": You can assign only a symbolically defined tag for the parameters.

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L	Input tag in the STRING format.
IN2	Input	STRING*	D, L	Input tag in the STRING format.
OUT	Return	STRING*	D, L	Combined character string
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

LEFT: Read the left character of a character string

Description

The instruction provides the first L characters of a string.

- If L is greater than the current length of the STRING tag, the input value is returned.
- With L = 0 and with a blank string as the input value, a blank string is returned.
- If L is negative, a blank string is output and the binary result (BR) bit of the status word is set to "0".

Parameters

The following table shows the parameters of the instruction "LEFT": You can only assign a symbolically defined tag for the parameter IN and the return value RET_VAL.

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L	Input tag in the STRING format.
L	Input	INT	I, Q, M, D, L or constant	Length of the left character string
OUT	Return	STRING*	D, L	Output tag in format STRING
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

RIGHT: Read the right characters of a character string

Description

The instruction provides the last L characters of a string.

- If L is greater than the current length of the STRING tag, the input value is returned.
- If L is negative, a blank string is returned and the binary result (BR) bit of the status word is set to "0".
- With L = 0 and with a blank string as the input value, a blank string is returned.

Parameters

The following table shows the parameters of the instruction "RIGHT": You can only assign a symbolically defined tag for the parameter IN and the return value RET_VAL.

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L	Input tag in the STRING format.
L	Input	INT	I, Q, M, D, L or constant	Length of the right character string
OUT	Return	STRING*	D, L	Output tag in format STRING
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

MID: Read middle characters of a character string

Description

The instruction provides the middle part of a character string (L characters from and including the P character).

- If the total of L and (P-1) exceeds the current length of the STRING tag, a string from the P character to the end of the input value is provided.
- In all other cases (P is outside the current length, P and/or L are equal to zero or negative), a blank string is returned and the binary result (BR) bit of the status word is set to "0".

Parameters

The following table shows the parameters of the instruction "MID": You can only assign a symbolically defined tag for the parameter IN and the return value RET_VAL.

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L	Input tag in the STRING format.
L	Input	INT	I, Q, M, D, L or constant	Length of the middle character string
P	Input	INT	I, Q, M, D, L or constant	Position of first character
OUT	Return	STRING*	D, L	Output tag in format STRING
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

DELETE: Delete characters in a character string

Description

The instruction deletes (L characters in a character string from and including the P character).

- If L and/or P are equal to zero or if P is greater than the current length of the input string, the input string is returned.
- If the sum of L and P is greater than the input string, the string is deleted up to the end.
- If L and/or P is negative, a blank string is returned and the binary result (BR) bit of the status word is set to "0".

Parameters

The following table shows the parameters of the instruction "DELETE": You can only assign a symbolically defined tag for the parameter IN and the output parameter RET_VAL.

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L	STRING tag in which the delete is made
L	Input	INT	I, Q, M, D, L or constant	Number of characters to be deleted
P	Input	INT	I, Q, M, D, L or constant	Position of first character to be deleted
OUT	Return	STRING*	D, L	Result string
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

INSERT: Insert characters in a character string

Description

The instruction adds the character string at the IN2 parameter to the character string at the IN1 parameter after the P character.

- If P equals zero, then the second character string will be inserted before the first character string.
- If P is greater than the actual length of the first character string, then the second character string is appended to the first character string.
- If P is negative, then a blank string will be output and the binary result (BR) of the status word is set to "0".

The binary result (BR) bit is also set to "0" if the result string is longer than the tag specified at the output parameter; in this case the result string is limited to the maximum set length.

Parameters

The following table shows the parameters of the instruction "INSERT": You can only assign a symbolically defined tag for the input parameters IN1 and IN2 and the output parameter.

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L	STRING tag in which the insertion is made
IN2	Input	STRING*	D, L	STRING tag that is to be inserted
P	Input	INT	I, Q, M, D, L or constant	Insert position
OUT	Return	STRING*	D, L	Result string
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

REPLACE: Replace characters in a character string

Description

The instruction replaces the number of L characters of the first character string (IN1) starting from and including the P character by the complete second character string (IN2).

- If L equals zero and P does not equal zero, then the first string is returned.
- If L equals zero and P equals zero, then the second string is positioned in front of the first string.
- If L does not equal zero and P equals zero or one, it is replaced starting from and including the first character.

- If P is outside the first string, then the second string is appended to the first string.
- If L and/or P are/is negative, then a blank string will be returned and the binary result (BR) bit of the status word is set to "0". The binary result (BR) bit is also set to "0" if the result string is longer than the tag specified at the output parameter; in this case the result string is limited to the maximum set length.

Parameters

The following table shows the parameters of the instruction "REPLACE": You can only assign a symbolically defined tag for the input parameters IN1 and IN2 and the output parameter.

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L	STRING tag in which the insertion is made
IN2	Input	STRING*	D, L	STRING tag that is to be inserted
L	Input	INT	I, Q, M, D, L or constant	Number of characters to be replaced
P	Input	INT	I, Q, M, D, L or constant	Position of first replaced character
OUT	Return	STRING*	D, L	Result string
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

FIND: Find characters in a character string

IND description

The instruction provides the position of the second string (IN2) within the first string (IN1). The search begins left. The first occurrence of the string is reported.

If the second string is not found in the first, zero is returned. The instruction does not report any errors.

Parameters

The following table shows the parameters of the instruction "FIND": You can only assign a symbolically defined tag for the input parameters IN1 and IN2 .

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L	STRING tag in which the search is made
IN2	Input	STRING*	D, L	STRING tag that is searched for

Parameters	Declaration	Data type	Memory area	Description
OUT	Return	INT	I, Q, M, D, L	Position of the string found
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

4.2.3.3 Process image

UPDAT_PI: Update the process image inputs

Description

With the instruction, you update the OB 1 process image (=process image partition 0) of the inputs or a process image partition input of the inputs defined by configuration.

If you have configured repeated signaling of I/O access errors for the system-side process image update, then the selected process image will be updated constantly.

Otherwise, this update will only be performed if the selected process image partition is not updated by the system, in other words:

- If you have not assigned this process image partition to an interrupt OB,

  or

- If you selected process image partition 0 and have disabled updating of the OB 1 process image partition during configuration.

Note

Each logical address that you have assigned to a process image partition input table per configuration no longer belongs to the OB 1 process image input table.

When you update a process image partition with "UPDAT_PI", you must not perform a simultaneous update with the "SYNC_PI (Page 5221)" instruction.

System-side updating of the OB 1 process image input table and the process image partitions of inputs that you have assigned to an interrupt OB takes place independently of "UPDAT_PI" calls.

Parameter

The following table shows the parameters of the instruction "UPDAT_PI":

Parameters	Declaration	Data type	Memory area	Description
PART	Input	BYTE	I, Q, M, D, L or constant	Number of the process image partition input table to be updated. Maximum value range (depending on the CPU): 0 to 15 (0 means OB 1 process image, n where 1 < n < 15 means process image partition n)
RET_VAL	Return	INT	I, Q, M, D, L	Error information
FLADDR	Output	WORD	I, Q, M, D, L	Address of the first byte to cause an error if an access error has occurred.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for PART parameter
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU.
8092	The process image partition is being updated by the system with an OB and you have not configured repeated signaling of all I/O access errors for this purpose. An update with "UPDAT_PI" was not performed.
80A0	An access error was detected when accessing the I/O.
8xyy	For general error information, refer to Evaluating errors with output parameter RET_VAL (Page 162)

Note

If you use the instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPRD_DAT (Page 5262)" instruction are also possible.

UPDAT_PO: Update the process image outputs

Description

You use the instruction to transfer the signal states of the OB 1 process image (= process image partition 0) of the outputs or a process image partition of the outputs defined per configuration to the output modules.

If you have specified a consistency range for the selected process image partition, corresponding data is transferred as consistent data to the respective I/O module.

Note

Each logical address you have assigned to a process image partition output table during configuration no longer belongs to the OB 1 process image output table.

Outputs that you update with "UPDAT_PO" must not be updated simultaneously with the "SYNC_PO (Page 5222)" instruction.

The OB 1 process image output table and the process image partitions of the outputs that you have assigned to an interrupt OB are transferred by the system to the output modules independently of "UPDAT_PO" calls.

Parameter

The following table shows the parameters of the instruction "UPDAT_PO":

Parameters	Declaration	Data type	Memory area	Description
PART	Input	BYTE	I, Q, M, D, L or constant	Number of the process image partition output table to be transferred. Maximum value range (depending on the CPU): 0 to 15. (0 means OB 1 process image, n where 1 < n < 15 means process image partition n)
RET_VAL	Return	INT	I, Q, M, D, L	Error information
FLADDR	Output	WORD	I, Q, M, D, L	Address of the first byte to cause an error if an access error has occurred.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for PART parameter
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU.
80A0	An access error was detected when accessing the I/O.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Note

If you use the instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPWR_DAT (Page 5265)" instruction are also possible.

SYNC_PI: Synchronize the process image inputs

Description

The "SYNC_PI" is used to update the process image partition of inputs in isochronous mode. A user program linked to a DP cycle or PN send cyle can use this instruction to update input data acquired in a process image partition input table isochronously and consistently.

Call

"SYNC_PI" is interruptible and can only be called in OBs 61, 62, 63, and 64.

Note

A call of the "SYNC_PI" instruction in OBs 61 to 64 not allowed unless you have assigned the process image partition concerned to the associated OB in the hardware configuration.

Process image partitions updated with "SYNC_PI" may not be updated by means of "UPDAT_PI (Page 5218)" as well.

Parameter

The following table shows the parameters of the instruction "SYNC_PI":

Parameter	Declaration	Data type	Memory area	Range of values	Description
PART	Input	BYTE	I, Q, M, D, L or constant	1 to 30	Number of the process image partition input table to be updated isochronously.
RET_VAL	Return	INT	I, Q, M, D, L	-	Error information
FLADDR	Output	WORD	I, Q, M, D, L	-	Address of the first byte to cause an error if an access error has occurred.

For more information about the valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Event class error code	Explanation
W#16#0001	Consistency warning. The update of the process image partition was distributed over two DP and PN cycles. However, the data in one slave or IO device were consistently transferred.
W#16#8090	Illegal value on PART parameter or updating the specified process image partition input table is not permitted in this OB. The process image partition input table was not updated.
W#16#8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU. The process image partition input table was not updated.
W#16#80A0	During updating an access error was detected. The affected inputs were set to "0".
W#16#80A1	The update time is after the permitted access window. The process image partition input table was not updated. The DP or PN cycle is too short to ensure enough time for processing the instruction. You must therefore increase the TDP (also known as T_DC), Ti, and To timers.
W#16#80A2	Access error with consistency warning In the case of update of the specified process image partition an access error with simultaneous consistency warning was detected. • The data of the incorrect inputs was not read by the I/O. In the process image partition of the inputs the involved inputs are set to zero. • The update of the process image partition of the not involved by an access error input data was distributed over two DP and PN cycles.
W#16#80C1	The update time is before the permitted access window. The process image partition input table was not updated.
W#16#8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Note

If you use the "SYNC_PI" instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPRD_DAT (Page 5262)" instruction are also possible.

SYNC_PO: Synchronize the process image outputs

Description

Instruction "SYNC_PO" is used to update a process image partition of outputs in isochronous mode A user program linked to a DP cycle can use this instruction to transfer the calculated output data of a process image partition outputs to the I/O isochronously and consistently.

Call

"SYNC_PO" is interruptible and can only be called in OBs 61, 62, 63, and 64.

Note

A call of the "SYNC_PO" instruction in OBs 61 to 64 is only permitted if you have assigned the affected process image partition to the associated OB in the HW configuration.

A process image partition that you update with "SYNC_PO" must not be updated simultaneously with the "UPDAT_PO (Page 5219)" instruction.

Parameter

The following table shows the parameters of the instruction "SYNC_PO":

Parameter	Declaration	Data type	Memory area	Range of values	Description
PART	Input	BYTE	I, Q, M, D, L or constant	1 to 30	Number of the process image partition output table to be updated isochronously.
RET_VAL	Return	INT	I, Q, M, D, L	-	If an error occurs while the instruction is being executed, the return value contains an error code.
FLADDR	Output	WORD	I, Q, M, D, L	-	Address of the first byte that has caused the error.

For more information about the valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Event class error code	Explanation
W#16#0001	Consistency warning. The update of the process image partition was distributed over two DP and PN cycles. However, the data in one slave or IO device were consistently transferred.
W#16#8090	Illegal value in PART parameter, or updating the specified process image partition output table is not permitted in this OB. Outputs were not transferred to the I/O. The process image partition output table remains unchanged.
W#16#8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU. Outputs were not transferred to the I/O. The process image partition output table remains unchanged.
W#16#80A0	In the case of update of the specified process image partition of the outputs an access error was detected. Incorrect outputs were not transferred to the I/O. During process image partition of the outputs, these outputs remain unchanged.
W#16#80A1	Access error with consistency warning In the case of update of the specified process image partition of the outputs an access error with simultaneous consistency warning was detected. • The data of the incorrect outputs were not transferred to the I/O. During process image partition of the outputs, the involved outputs remain unchanged. • The update of the process image partition of the not involved by an access error input data was distributed over two DP and PN cycles.

Event class error code	Explanation
W#16#80A2	The update time is after the permitted access window. Outputs were not transferred to the I/O. The process image partition output table remains unchanged.
W#16#80C1	The update time is before the permitted access window. Outputs were not transferred to the I/O. The process image partition output table remains unchanged.
W#16#8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Note

If you use the "SYNC_PO" instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPWR_DAT (Page 5265)" instruction are also possible.

4.2.3.4 Distributed I/O

RDREC: Read data record

Description

You use the instruction to read the data record with the number INDEX from the component addressed using the ID . This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

With MLEN , you specify the maximum number of bytes you want to read. The selected length of the destination area RECORD should have at least the length of MLEN bytes.

The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the destination area RECORD . In this case, the LEN output parameter contains the length of the read data in bytes.

If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR . In this case, the output parameter STATUS contains the error information.

Note

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not read any data records from the I/O modules in the user program with "RDREC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: set the interface of the DP master to "DPV1".

Note

The interface of the "RDREC" instruction is identical to that in the "RDREC" FB defined in the standard "PROFIBUS and PROFINET Guideline Communication Function Blocks on PROFIBUS DP and PROFINET IO".

Functional description

"RDREC" is an instruction that works asynchronously, which means its execution extends over multiple calls. You start the data record transfer by calling RDREC with REQ = 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS . The two central bytes of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Difference between synchronous and asynchronous instructions (Page 4186)

The transfer of the data record is complete when the output parameter BUSY has the value FALSE .

As long as the data record transfer is taking place, the destination area must not be changed, neither its length nor its content.

Parameter

The following table shows the parameters of the "RDREC" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Transfer data record
ID	Input	DWORD	I, Q, M, D, L or constant	Logical address of the DP slave/PROFINET IO component (module or sub-module) • For an output module, bit 15 has to be set (for example, for address 5: ID:=DW#16#8005). • For a mixed module, the lower of the two addresses must be specified.
INDEX	Input	INT	I, Q, M, D, L or constant	Data record number
MLEN	Input	INT	I, Q, M, D, L or constant	maximum length in bytes of the data record information to be read
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information For interpretation of the parameter STATUS , refer to parameter STATUS (Page 5235).

Parameter	Declaration	Data type	Memory area	Description
LEN	Output	INT	I, Q, M, D, L	Length of the read data record information
RECORD	InOut	ANY	I, Q, M, D, L	Destination area for the read data record Note: Note that for S7-300 CPUs, the parameter RECORD always requires the full specification of the DB parameters (for example: P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

If you use the instruction to read a data record with PROFINET IO, negative values in the INDEX, MLEN and LEN parameters are interpreted as an unsigned 16-bit integer.

WRREC: Write data record

Description

You use the instruction to transfer the data record RECORD to the component addressed using ID. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

With LEN (hidden), you specify in bytes the length of the data record to be transmitted. The selected length of the source area RECORD should have at least the length of LEN bytes.

The value TRUE at output parameter DONE indicates that the data record has been successfully transferred.

If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR . In this case, the output parameter STATUS contains the error information.

Note

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not write any data records from the I/O modules to the user program with "WRREC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: set the interface of the DP master to "DPV1".

Note

The interface of the "WRREC" instruction is identical to that in the "WRREC" instruction defined in the standard "PROFIBUS and PROFINET Guideline Communication Function Blocks on PROFIBUS DP and PROFINET IO".

Functional description

"WRREC" is an instruction that works asynchronously, which means its execution extends over multiple calls. You start the data record transfer by calling WRREC with REQ = 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS . The two central bytes of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Difference between synchronous and asynchronous instructions (Page 4186)

Note that you must assign the same value to the actual parameter of RECORD for all "WRREC"calls that belong to one and the same job. The same applies to the actual parameters of LEN.

The transfer of the data record is complete when the output parameter BUSY has the value FALSE .

Parameter

The following table shows the parameters of the "WRREC" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Transfer data record
ID	Input	DWORD	I, Q, M, D, L or constant	Logical address of the DP slave/PROFINET IO component (module or sub-module) Bit 15 must be set for an output module (e.g. for address 5: ID:=DW#16#8005). The smaller of the two addresses must be specified for a mixed module.
INDEX	Input	INT	I, Q, M, D, L or constant	Data record number
LEN	Input	INT	I, Q, M, D, L or constant	(hidden) Maximum length of the data record to be transferred in bytes
DONE	Output	BOOL	I, Q, M, D, L	Data record was transferred
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the writing process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information For interpretation of the parameter STATUS , refer to parameter STATUS (Page 5235).
RECORD	InOut	ANY	I, Q, M, D, L	Data record Note: Note that for S7-300 CPUs, the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

If you use the instruction to read a data record with PROFINET IO, negative values in the INDEX and LEN parameters are interpreted as an unsigned 16-bit integer.

GETIO: Read process image

Description

You use the instruction to read out all inputs of a DP standard slave/PROFINET IO device. The instruction calls the "DPRD_DAT (Page 5262)" instruction. If there was no error during the data transmission, the data that have been read are entered in the destination area indicated by INPUTS .

The destination area must have the same length that you configured for the selected component.

If you read from a DP standard slave with a modular configuration or with several DP identifiers, you can only access the data of one component/DP identifier at the configured start address per "GETIO" call.

Parameter

The following table shows the parameters of the instruction "GETIO":

Parameters	Declaration	Data type	Memory area	Description
ID	Input	DWORD	I, Q, M, D, L or constant	• Low word: logical address of the DP slave/PROFINET IO component (module or sub-module) • High word: irrelevant
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPRD_DAT (Page 5262)" in the form DW#16#40xxxx00
LEN	Output	INT	I, Q, M, D, L	Amount of data read in bytes
INPUTS	InOut	ANY	I, Q, M, D	Destination area for the read data. It must have the same length as the area that you configured for the selected DP slave/PROFINET IO component. Only the BYTE data type is permitted.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter STATUS

See also: DPRD_DAT: Read consistent data of a DP standard slave (Page 5262).

SETIO: Transfer process image

Description

You use the "SETIO" instruction to consistently transfer the data from the source area spanned by OUTPUTS to the addressed DP standard slave/PROFINET IO device, and, if necessary, to the process image (if you have configured the relevant address area of the DP standard slave as a consistent range in a process image). "SETIO" calls the instruction "DPWR_DAT (Page 5265)".

The source area must have the same length that you configured for the selected component.

In the case of a DP standard slave with a modular configuration or with several DP identifiers, you can only access one component/DP identifier of the DP slave per "SETIO" call.

Parameter

The following table shows the parameters of the instruction "SETIO":

Parameters	Declaration	Data type	Memory area	Description
ID	Input	DWORD	I, Q, M, D, L or constant	• Low word: logical address of the DP slave/PROFINET IO component (module or sub-module) • High word: irrelevant
LEN	Input	INT	I, Q, M, D, L	irrelevant
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPWR_DAT (Page 5265)" in the form DW#16#40xxxx00
OUTPUTS	InOut	ANY	I, Q, M, D	Source area for the data to be written. It must have the same length as the area that you configured for the selected DP slave/PROFINET IO component. Only the BYTE data type is permitted.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter STATUS

See also: DPWR_DAT: Write consistent data of a DP standard slave (Page 5265).

GETIO_PART: Read process image area

Description

You use the "GETIO_PART" instruction to consistently read part of the process image area belonging to a DP standard slave/PROFINET IO device. "GETIO_PART" calls the instruction "UBLKMOV.

Note

You must assign a process image partition input table to the OB in which "GETIO_PART" is called. Before calling "GETIO_PART" you must add the associated DP standard slave or the associated PROFINET IO device to this process image partition input table. If your CPU does not recognize any process image partitions or you want to call "GETIO_PART" in OB 1, you must add the associated DP standard slave or the associated PROFINET IO device to this process image input table before calling "GETIO_PART".

You use the OFFSET and LEN parameters to specify the portion of the process image area to be read for the components addressed by means of their ID.

- If there was no error during the data transmission, ERROR receives the value FALSE, and the data that have been read are entered in the destination area indicated by INPUTS.
- If there was an error during the data transmission, ERROR receives the value TRUE, and STATUS receives the error information of "UBLKMOV".
- If the destination area (parameter INPUTS) is less than LEN, then as many bytes as INPUTS can accept will be transferred. ERROR receives the value FALSE. If the destination area is greater than LEN, then the first LEN bytes of the destination area will be written. ERROR receives the value FALSE.

Note

"GETIO_PART" does not check the process image input table for delimiters between data belonging to different PROFIBUS DP or PROFINET IO components. This means you have to make sure that the process image area specified by means of OFFSET and LEN belongs to one component. Reading of data for more than one component cannot be guaranteed for future systems and compromises the transferability to systems from other manufacturers.

Parameter

The following table shows the parameters of the instruction "GETIO_PART":

Parameters	Declaration	Data type	Memory area	Description
ID	Input	DWORD	I, Q, M, D, L or constant	• Low word: logical address of the DP slave/PROFINET IO component (module or sub-module) • High word: irrelevant
OFFSET	Input	INT	I, Q, M, D, L or constant	Number of the first byte to be read in the process image for the component (smallest possible value: 0)

Parameters	Declaration	Data type	Memory area	Description
LEN	Input	INT	I, Q, M, D or constant	Amount of bytes to be read
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "UBLK-MOV" in the form DW#16#40xxxx00, if ERROR = TRUE
ERROR	Output	BOOL	I, Q, M, D, L	Error display: ERROR = TRUE if an error occurs when "UBLKMOV" is called.
INPUTS	InOut	ANY	I, Q, M, D	Destination area for read data: • If the destination area is less than LEN, then as many bytes as INPUTS can accept will be transferred. ERROR receives the value FALSE. • If the destination area is greater than LEN, then the first LEN bytes of the destination area will be written. ERROR receives the value FALSE.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters STATUS and ERROR

See instruction "UBLKMOV".

SETIO_PART: Transfer process image area

Description

You use the "SETIO_PART" instruction to consistently transfer data from the source area spanned by OUTPUTS into a part of the process image area belonging to a DP standard slave/ PROFINET IO device. "SETIO_PART" calls the instruction "UBLKMOV".

Note

You must assign a process image partition input table to the OB in which "SETIO_PART" is called. Before calling "SETIO_PART", you must add the associated DP standard slave or the associated PROFINET IO device to this process image partition output table. If your CPU does not recognize any process image partitions or you want to call "SETIO_PART" in OB 1, you must add the associated DP standard slave or the associated PROFINET IO device to this process image output table before calling "SETIO_PART".

You use the OFFSET and LEN parameters to specify the portion of the process image area to be written for the components addressed by means of their ID.

- If there was no error during the data transmission, ERROR receives the value FALSE.
- If there was an error during the data transmission, ERROR receives the value TRUE, and STATUS receives the error information of "UBLKMOV".
- If the source area (OUTPUTS parameter) is less than LEN, then as many bytes as OUTPUTS can accept will be transferred. ERROR receives the value FALSE. If the source area is greater than LEN, then the first LEN bytes from OUTPUTS will be transferred. ERROR receives the value FALSE.

Note

"SETIO_PART" does not check the process image input table for delimiters between data belonging to different PROFIBUS DP or PROFINET IO components. This means you have to make sure that the process image area specified by means of OFFSET and LEN belongs to one component. Writing of data for more than one component cannot be guaranteed for future systems and compromises the transferability to systems from other manufacturers.

Parameter

The following table shows the parameters of the instruction "SETIO_PART":

Parameters	Declaration	Data type	Memory area	Description
ID	Input	DWORD	I, Q, M, D, L or constant	• Low word: logical address of the DP slave/PROFINET IO component (module or sub-module) • High word: irrelevant
OFFSET	Input	INT	I, Q, M, D, L or constant	Number of the first byte to be written in the process image for the component (smallest possible value: 0)
LEN	Input	INT	I, Q, M, D, L or constant	Amount of bytes to be written
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "UBLK-MOV" in the form DW#16#40xxxx00, if ER-ROR = TRUE
ERROR	Output	BOOL	I, Q, M, D, L	Error display: ERROR = TRUE if an error occurs when "UBLKMOV" is called.
OUTPUTS	InOut	ANY	I, Q, M, D	Source area for the data to be written: • If the source area is less than LEN, then as many bytes as OUTPUTS can accept will be transferred. ERROR receives the value FALSE. • If the source area is greater than LEN, then the first LEN bytes from OUTPUTS will be transferred. ERROR receives the value FALSE.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters STATUS and ERROR

See instruction "UBLKMOV".

RALRM: Receive interrupt

Description RALRM

Description

The instruction receives an interrupt with all corresponding information from an I/O module (centralized structure) or from a DP slave or PROFINET IO device component; it supplies this information to its output parameters.

The information in the output parameters contains the start information of the called OB as well as information of the interrupt source.

Call the instruction only within the interrupt OB started by the CPU operating system as a result of the I/O interrupt that is to be examined.

Note

If you call the instruction in an OB whose start event is not an I/O interrupt, the instruction will provide less information to your outputs.

Note that you use different instance DBs when you call the instruction in different OBs. If you evaluate data resulting from an "RALRM" call outside of the associated interrupt OB, you should moreover use a separate instance DB per OB start event.

Note

The interface of the "RALRM" instruction is identical to that in the "RALRM" FB defined in the standard "PROFIBUS and PROFINET Guideline Communication Function Blocks on PROFIBUS DP and PROFINET IO".

Calling "RALRM"

You can call the instruction in three operating modes (MODE). These modes are explained in the following table.

MODE	RALRM ...
0	... shows the component that triggered the interrupt in the output parameter ID and sets the output parameter NEW to TRUE.
1	... writes all output parameters, regardless of the component triggering the interrupt.
2	... checks whether the component specified in the F_ID input parameter has triggered the interrupt. • If not, NEW = FALSE • If yes, NEW = TRUE and all other output parameters are written.

Parameters

The following table shows the parameters of the "RALRM" instruction:

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	DWORD	I, Q, M, D, L or constant	Logical start address of the component (module) from which interrupts will be received
MLEN	Input	INT	I, Q, M, D, L or constant	Maximum length of the interrupt information to be received, in bytes
NEW	Output	BOOL	I, Q, M, D, L	A new interrupt was received.
STATUS (Page 5235)	Output	DWORD	I, Q, M, D, L	Error code of the instruction or DP master
ID	Output	DWORD	I, Q, M, D, L	Logical start address of the component (module) from which an interrupt was received. Bit 15 contains the I/O ID: 0 = input address, 1 = output address.
LEN	Output	INT	I, Q, M, D, L	Length of the received interrupt information

Parameters	Declaration	Data type	Memory area	Description
TINFO (Page 5240)	InOut	ANY	I, Q, M, D, L	(task information) Destination area for OB start and management information Note: Note that for S7-300 CPUs, the parameter TINFO always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
AINFO (Page 5243)	InOut	ANY	I, Q, M, D, L	(alarm information) Destination area for header information and additional interrupt information For AINFO you should provide a length of at least MLEN bytes. Note: Note that for S7-300 CPUs, the parameter AINFO always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

If you select a destination area TINFO or AINFO that is too short, the instruction cannot enter the full information.

See also: Destination area TINFO and AINFO (Page 5250)

Parameter STATUS

Parameter STATUS structure

The STATUS output parameter contains error information. If it is interpreted as ARRAY[1...4] OF BYTE , the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	• B#16#00, if no error • Function ID from DPV1-PDU: If an error occurs, B#16#80 will be output (for read data record: B#16#DE, for write data record: B#16#DF). If no DPV1 protocol element is used: B#16#C0.
STATUS[2]	Error_Decode	Location of the error ID

Field element	Name	Meaning
STATUS[3]	Error_Code_1	Error ID
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Field element STATUS[2]

STATUS[2] can have the following values:

Error_Decode (B#16#...)	Source	Meaning
00 to 7F	CPU	No error or no warning
80	DPV1	Error according to IEC 61158-6
81 to 8F	CPU	B#16#8x shows an error in the xth call parameter of the instruction.
FE, FF	DP profile	Profile-specific error

Field element STATUS[3]

STATUS[3] can have the following values:

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
00	00		No error, no warning
70	00	reserved, reject	Initial call; no active data record transfer
	01	reserved, reject	Initial call; data record transfer has started
	02	reserved, reject	Intermediate call; data record transfer already active
80	90	reserved, pass	Invalid logical start address
	92	reserved, pass	Illegal type for ANY pointer
	93	reserved, pass	The DP component addressed via ID or F_ID is not configured.
	95		Error when reading additional interrupt information (when reading out additional interrupt information for central or distributed I/O via an external DP interface, this error will be output as a "group error".) Note: During link-up and update, the additional interrupt information may not be available temporarily.

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	96		The master CPU is in STOP mode. At that time, an OB was being processed. The instruction "RALRM (Page 5233)" cannot supply the OB start information, the management information, the header information, or the additional interrupt information. The OB start information can be read with the "RD_SINFO (Page 5442)" instruction. In addition, you can use the "DPNRM_DG (Page 5287)" instruction to asynchronously read the current diagnostics frame of the affected DP slave for OBs 4x, 55, 56, 57, 82, and 83 (address information from the OB start information).
	A0	read error	Negative acknowledgment while reading the module.
	A1	write error	Negative acknowledgement when writing to the module
	A2	module failure	DP protocol error at layer 2 (e.g., slave failure or bus problems)
	A3	reserved, pass	General communication error or IO device / DP slave is unreachable
	A4	reserved, pass	Communication on the communication bus disrupted
	A5	reserved, pass	–
	A7	reserved, pass	DP slave or module is occupied (temporary error
	A8	version conflict	DP slave or module reports non-compatible versions
	A9	feature not supported	Function is not supported by DP slave or module
	AA to AF	user specific	DP slave or module reports a manufacturer-specific error in its application. Please check the documentation from the manufacturer of the DP slave or module.
	B0	invalid index	Data record not known in module Illegal data record number ≥ 256
	B1	write length error	The length specified in the RECORD parameter is incorrect; • With "RALRM (Page 5233)": length error in AINFO, • With "RDREC (Page 5224)" and "WRREC (Page 5226)": length error in MLEN
	B2	invalid slot	• The configured slot is not occupied. • For PROFINET IO and PROFIBUS DP: IO device / DP slave is unreachable

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	B3	type conflict	Actual module type does not match specified module type
	B4	invalid area	DP slave or module reports access to an invalid area
	B5	state conflict	DP slave or module not ready
	B6	access denied	DP slave or module denies access
	B7	invalid range	DP slave or module reports an invalid range for a parameter or value
	B8	invalid parameter	DP slave or module reports an invalid parameter
	B9	invalid type	DP slave or module reports an invalid type • With "RDREC (Page 5224)": buffer too small (subsets cannot be read) • With "WRREC (Page 5226)": buffer too small (subsets cannot be written)
	BA to BF	user specific	DP slave or module reports a manufacturer-specific error when accessing. Please check the documentation from the manufacturer of the DP slave or module. Note on value B#16#BA: The following applies for PROFINET in the H system: If a data record job with the return value W#16#80BA is rejected, the job must be repeated.
	C0	read constrain conflict	With "WRREC (Page 5226)": the data can only be written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You can only write the data online with PG/PC. With "RDREC (Page 5224)": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. In this case, you can only read the data online with PG/PC.
	C1	write constrain conflict	The data of the previous write job on the module for the same data record have not yet been processed by the module.
	C2	resource busy	The module is currently processing the maximum possible number of jobs for a CPU.
	C3	resource unavailable	The required operating resources are currently occupied.

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	C4		Internal temporary error. Job could not be carried out. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.
	C5		DP slave or module not available.
	C6		Data record transfer was canceled due to priority class cancellation
	C7		Job aborted due to warm or cold restart on the DP master
	C8 bis CF		DP slave or module reports a manufacturer-specific resource error. Please check the documentation from the manufacturer of the DP slave or module.
	Dx	user specific	DP slave specific. Refer to the description of the DP slave.
8x (x = 1, ... 9, A, B, C, D, E, F)	00 to FF		Error in y-th call parameter (y = 1, ...15) • Error_Code_1 = 00: Illegal operating mode • For all other values of Error_Code_1 refer also to: Evaluating errors with output parameter RET_VAL (Page 162)
FE, FF	00 bis FF		Profile-specific error

Field element STATUS[4]

With DPV1 errors, the DP master passes on STATUS[4] to the CPU and the instruction. Without DPV1 error, this value is set to 0, with the following exceptions for the "RDREC (Page 5224)" and "WRREC (Page 5226)" instructions:

- STATUS[4] contains the target range length from RECORD, if LEN > the target range length from RECORD
- STATUS[4]=LEN, if the actual data record length < LEN < the target range length from RECORD
- STATUS[4]=0, if STATUS[4] > 255 has to be set.

In PROFINET IO, STATUS[4] has the value 0.

Parameter TINFO

Data structure of the destination area TINFO

Byte	Meaning
0 to 19	Start information of the OB in which "RALRM" was currently called
20 and 21	Address, for exact description, see below
22 to 31	Management information, for exact description, see below

Structure of the address (bytes 20 and 21)

The address contains:

- In a central configuration, the rack number (0-31).

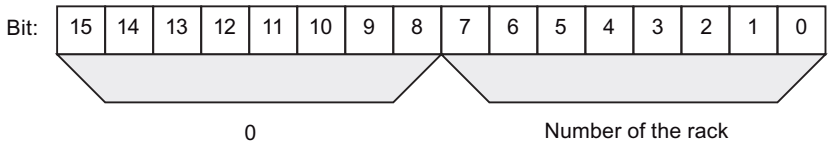

- In a distributed configuration with PROFIBUS DP
  - The DP master system ID (1-32)
  - The station number (0-127).

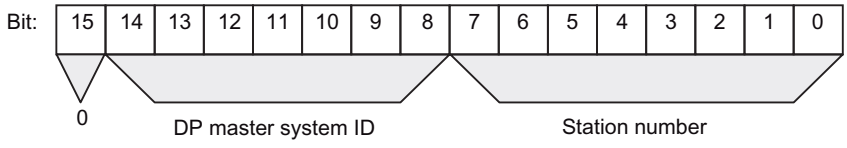

- In a distributed configuration with PROFINET IO:
  - The last two positions in the PROFINET IO system ID (0-15). To obtain the complete PROFINET IO system ID, you must add 100 (decimal) to it.
  - The station number (0-2047).

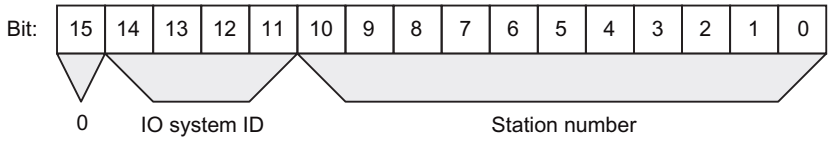

Structure of the management information in bytes 20 to 25

Byte no. for TINFO	Data type	Meaning				
20	BYTE	central:	0			
		distrib-uted:	PROFIBUS DP: DP master ID: 1 to 32)			
			PROFINET IO: see above			
21	BYTE	central:	Module rack number (possible values: 0 to 31)			
		distrib-uted:	Number of the DP station (possible values: 0 to 127)			
			PROFINET IO: see above			
22	BYTE	central:	• 0: Data record 0 or data record 1			
		distrib-uted:	Bit 0 to 3:	Slave type	0000: 0001: 0010: 0011: 0100 – 0111: 1000: 1001 and high-er:	DP (data record 0 structure) DPS7 (data record 0 or data re-cord 1 structure) DPS7 V1 (data record 0 or data record 1 structure) DPV1 (structure acc. to PROFI-BUS DP standard) reserved PROFINET IO (structure acc. to PROFINET IO Standard) reserved
			• Bit 4 to 7:	Profile type		Reserved
23	BYTE	central:	• 0			
		distrib-uted:	• Bit 0 to 3:	Interrupt info type	0000:	Transparent, which is always the case for PROFINET IO (interrupt originates from a con-figured distributed module)
					0001:	Representative (interrupt originates from a non-DPV1 slave/non IO device or a slot that is not configured)
					0010:	Generated (interrupt generated in the CPU)
					0011 and higher:	Reserved

Byte no. for TINFO	Data type	Meaning				
			• Bit 4 to 7:	Structure version	0000: 0001 and higher:	Initial Reserved
24	BYTE	central:	• 0			
		distrib-uted:	**Flags of the PROFIBUS DP master interface module/PROFINET IO controller master interface module**			
			Bit 0 = 0:	Interrupt originating from an integrated interface module (PROFINET IO or PROFIBUS DP)		
			Bit 0 = 1:	Interrupt originating from an external interface module (PROFI-NET IO or PROFIBUS DP)		
			Bit 1 to 7:	Reserved		
25	BYTE	central:	• 0			
		distrib-uted:	**Flags of the PROFIBUS DP slave interface module**			
			• Bit 0:	EXT_DIAG_FLAG from the diagnostics message frame, or 0 if this bit does not exist in the interrupt The bit is 1 if the DP slave is faulty.		
			• Bit 1 to 7:	Reserved		
			Flags of the PROFINET IO controller interface module			
			• Bit 0:	ARDiagnosisState or 0 if there is no information in the interrupt. The bit is 1 if the IO device is faulty.		
			• Bit 1 to 7:	Reserved		

Structure of the management information in bytes 26 to 27 with PROFIBUS and a central configuration

Byte no. for TIN-FO	Data type	Meaning	
26 and 27	WORD	central:	0
	WORD	distributed:	PROFIBUS ID number as unique identifier of the PROFIBUS DP slave
28 and 29	WORD	0	(Bytes 28 and 29 can be omitted)
30 and 31	WORD	0	(Bytes 30 and 31 can be omitted)

Structure of the management information in bytes 26 to 31 with PROFINET IO

Byte no. for TIN-FO	Data type	Meaning	
26 and 27	WORD	distributed:	PROFINET IO device ID number as unique identifier of the PROFINET IO device
28 and 29	WORD	distributed:	Manufacturer ID
30 and 31	WORD	distributed:	ID number of the instance

Parameter AINFO

Data structure of the destination area AINFO with interrupts from PROFIBUS DP or central I/O devices

The information for PROFINET IO is provided below.

Byte	Meaning	
0 to 3	Header information, for exact description, see below	
4 to 199	Additional interrupt information: data for the respective interrupt:	
	central:	ARRAY[0] to ARRAY[195]
	distributed:	ARRAY[0] to ARRAY[59]

Structure of the header information with interrupts from PROFIBUS DP or central IO devices

Byte	Data type	Meaning		
0	BYTE	Length of the received interrupt information in bytes		
		central: distributed:	4 to 224 4 to 63	
1	BYTE	central:	Reserved	
		distributed:	ID for the interrupt type	
			1: 2: 3: 4: 5: 6: 31	Diagnostics interrupt Process interrupt Removal interrupt Insertion interrupt Status interrupt Update Interrupt Failure of an expansion device, DP master system, or DP station
			32 to 126:	Manufacturer-specific interrupt
2	BYTE	Slot number of the component that triggered the interrupt		
3	BYTE	central:	Reserved	
		distributed:	Specifier	
			Bits 0 and 1	0: no further information; 1: Incoming event, faulty slot 2: Outgoing event, slot not faulty anymore 3: Outgoing event, slot still faulty
			Bit 2:	Add_Ack
			Bits 3 to 7:	Sequence number

Data structure of the destination area AINFO with interrupts from PROFINET IO

Byte	Meaning
0 to 25	Header information, for exact description, see below
26 to 1431	Additional interrupt information: Standardized diagnostics data for the respective interrupt: ARRAY[0] to ARRAY[1405] Note: The additional interrupt information may also be omitted.

Structure of the header information with interrupts from PROFINET IO

Byte	Data type	Meaning
0 and 1	WORD	• Bits 0 to 7: Block type • Bits 8 to 15: Reserved
2 and 3	WORD	Block length
4 and 5	WORD	Version: • Bits 0 to 7: low byte • Bits 8 to 15: high byte
6 and 7	WORD	ID for interrupt type: • 1: Diagnostics interrupt (incoming) • 2: Process interrupt • 3: Remove module interrupt • 4: Insert module interrupt • 5: Status interrupt • 6: Update interrupt • 7: Redundancy interrupt • 8: Controlled by supervisor • 9: Released by supervisor • 10: Configured module not inserted • 11: Return of the sub-module • 12: Diagnostics interrupt (outgoing) • 13: Slave-to-slave connection alarm • 14: Neighborhood change alarm • 15: Clock synchronization message (bus end) • 16: Clock synchronization alarm (device end) • 17: Network component alarm • 18: Time synchronization alarm (bus end) • 19 to 31: Reserved • 32 to 127: Manufacturer-specific interrupt • 128 to 65535: Reserved
8 to 11	DWORD	API (Application Process Identifier)
12 to 13	WORD	Slot number of the component triggering the interrupt (range of values 0 to 65535)

Byte	Data type	Meaning
14 to 15	WORD	Submodule slot number of the component triggering the interrupt (range of values 0 to 65535)
16 to 19	DWORD	Module identification; specific information on the source of the interrupt
20 to 23	DWORD	Submodule identification; specific information on the source of the interrupt
24 to 25	WORD	Interrupt specifier: • Bits 0 to 10: Sequence number (range of values 0 to 2047) • Bit 11: Channel diagnostics: 0: No channel diagnostics available 1: Channel diagnostics information exists • Bit 12: Status of manufacturer-specific diagnostics: 0: No manufacturer-specific status information available 1: Manufacturer-specific status information available • Bit 13: Status of diagnostics for sub-module: 0: No status information available, all errors have been corrected 1: At least one item of channel diagnostics and/or status information is available • Bit 14: Reserved • Bit 15: Application relationship diagnosis state: – 0: None of the modules configured within this application relationship reports diagnostics information – 1: At least one of the modules configured in this AR is reporting diagnostics information

Structure of additional interrupt information with interrupts from PROFINET IO

The additional interrupt information for PROFINET IO depends on the format identifier. It can comprise multiple data blocks with the same or different format identifier. The following format identifiers are available:

- W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics
2 to n	BYTE	See manufacturer's manual.

- W#16#8000: Channel diagnostics
  Channel diagnostics is output in blocks of 6 bytes each. The additional interrupt information (without format identifier) is only output for disrupted channels.

Byte	Data type	Meaning	
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8000: Channel diagnostics	
2 to 3	WORD	Channel number of the component triggering the interrupt (range of values: 0 to 65535): • W#16#0000 to W#16#7FFF: Channel number of the interface module/sub-module • W#16#8000: The generic substitute for the entire sub-module • W#16#8001 to W#16#FFFF: Reserved	
4	BYTE	Bits 0 to 2:	Reserved
		Bits 3 to 4:	Type of error: • 0: Reserved • 1: Incoming error • 2: Outgoing error • 3: Outgoing error, other errors present
		Bits 5 to 7:	Type of channel: • 0: Reserved • 1: Input channel • 2: Output channel • 3: Input/output channel

Byte	Data type	Meaning
5	BYTE	Data format: • B#16#00: Free data format • B#16#01: Bit • B#16#02: 2 bits • B#16#03: 4 bits • B#16#04: Byte • B#16#05: Word • B#16#06: Double word • B#16#07: 2 double words • B#16#08 to B#16#FF: Reserved
6 to 7	WORD	Type of error: • W#16#0000: Reserved • W#16#0001: Short circuit • W#16#0002: Undervoltage • W#16#0003: Overvoltage • W#16#0004: Overload • W#16#0005: Overtemperature • W#16#0006: Wire break • W#16#0007: High limit exceeded • W#16#0008: Low limit exceeded • W#16#0009: Error • W#16#000A to W#16#000F: Reserved • W#16#0010 to W#16#001F: Manufacturer-specific • W#16#0020 to W#16#00FF: Reserved • W#16#0100 to W#16#7FFF: Manufacturer-specific • W#16#8000: Device diagnostics available • W#16#8001 to W#16#FFFF: Reserved Not all channels support every error type. For detailed information, refer to the description of the diagnostics data for the specific device.

Note

The section from "channel number" to "type of error" can occur from 0 to n times.

W#16#8001

W#16#8001: MULTIPLE (different types of diagnostics information are transmitted).

In this case, the additional interrupt information is transmitted as blocks of variable length.

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8001: Manufacturer-specific diagnostics and/or channel diagnostics
2 to 3	WORD	Block type
4 to 5	WORD	Block length
6	BYTE	Version: high byte
7	BYTE	Version: low byte
8 to 11	DWORD	API (only if low byte of version = 1)
12 to 13	WORD	Slot number
14 to 15	WORD	Subslot number
16 to 17	WORD	Channel number:
18 to 19	WORD	Channel properties
20 to 21	WORD	Format identifier: • W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics • W#16#8000: Channel diagnostics • W#16#8002: Extended channel diagnostics • W#16#8003: Stepped extended channel diagnostics • W#16#8004 to W#16#80FF: Reserved
22 to n	BYTE	Data depend on the format identifier

Note

The section starting from "block type" can occur from 1 to n times.

W#16#8002

W#16#8002: Extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8002
2 to 3	Channel number:
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information

W#16#8003

W#16#8003: Stepped extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8003
2 to 3	Channel number:
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information
14 to 17	Qualified channel qualifier

W#16#8100

W#16#8100: Maintenance information

Byte	Meaning
0 to 1	Format identifier W#16#8100
2 to 3	Block type
4 to 5	Block length
6 to 7	Block version
8 to 9	Reserved
10 to 13	Maintenance status

Note

You can find more detailed information about the structure of the additional alarm information in the *Programming Manual SIMATIC PROFINET IO from PROFIBUS DP to PROFINET IO* and the current version of IEC 61158-6-10-1.

Destination area TINFO and AINFO

Destination area TINFO and AINFO

Depending on the respective OB in which the instruction "RALRM" is called, the destination areas TINFO and AINFO are only partially written. Refer to the table below to find out which information is entered respectively.

Interrupt type	OB	TINFO OB status information	TINFO Management information	AINFO Header information	AINFO Additional interrupt information	
Process interrupt	4x	Yes	Yes	Yes	central:	No
					distributed:	as supplied by PROFIBUS DP slave/PROFINET IO device
Status interrupt	55	Yes	Yes	Yes	Yes	Yes
Update interrupt	56	Yes	Yes	Yes	Yes	Yes
Manufacturer-specific interrupt	57	Yes	Yes	Yes	Yes	Yes
I/O redundancy error	70	Yes	Yes	No	No	No
Diagnostics interrupt	82	Yes	Yes	Yes	central:	Data record 1
					distributed:	as supplied by PROFIBUS DP slave/PROFINET IO device
Insert/remove interrupt	83	Yes	Yes	Yes	central:	No
					distributed:	as supplied by PROFIBUS DP slave/PROFINET IO device
Special form of the remove module interrupt: Controlled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Special form of the insert module interrupt: Enabled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Unconfigured module inserted	83	Yes	Yes	Yes	PROFINET IO only	
Rack failure/ station failure	86	Yes	Yes	No	No	
... all other OBs		Yes	No	No	No	

D_ACT_DP: Enable/disable DP slaves

Description

Use the "D_ACT_DP" instruction to specifically deactivate and reactivate configured DP slaves/ PROFINET IO devices. In addition, you can determine whether each assigned DP slave or PROFINET IO device is currently activated or deactivated.

If you use the instruction to deactivate an IE/PB Link PN IO type of gateway, then all connected PROFIBUS DP slaves will also cease to function. These failures will be reported.

The instruction cannot be used on PROFIBUS PA field devices that are connected by a DP/ PA link to a DP master system.

Note

As long as one or more "D_ACT_DP" jobs are active, you cannot load a changed configuration from the programming device to the CPU (within the scope of CiR).
When a changed configuration is loaded from the programming device to the CPU during ongoing operation (CiR), the CPU rejects activation of a "D_ACT_DP" job.

Several runs through the cycle control point are required to perform the disabling or enabling job. Therefore, you cannot wait for the end of such a job in a programmed loop.

Parameter

The following table shows the parameters of the "D_ACT_DP" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Level-triggered control parameter REQ=1: Run activation or deactivation
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Possible values: • 0: Request information on whether the addressed component is activated or deactivated • 1: Activate the DP slave/PROFINET IO device • 2: Deactivate the DP slave/PROFINET IO device • 3: Activate DP slave/PROFINET IO device and call OB 86 after the change of the activation state has been completed • 4: Deactivate DP slave/PROFINET IO device and call OB 86 after the change of the activation state has been completed

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	WORD	I, Q, M, D, L or constant	Any logical address of the DP slave/PROFINET IO device. For an output address, the most significant bit must be set.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	Active code: • BUSY=1: The job is still active. • BUSY=0: The job was terminated.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	Job completed without errors.
0001	The DP slave/PROFINET IO device is active (this error code is possible only with MODE = 0.)
0002	The DP slave/PROFINET IO device is deactivated (this error code is possible only with MODE = 0.)
7000	First call with REQ= 0: The job specified with LADDR is not active; BUSY has the value "0".
7001	First call with REQ= 1. The job specified with LADDRwas triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant). The activated job is still active; BUSY has the value "1".
8090	• You have not configured a module with the address specified in LADDR. • You operate your CPU as I-Slave and you have specified in LADDR an address of this I-Slave.
8092	The deactivation of the currently addressed DP slave/PROFINET IO device (MODE=2) cannot be canceled by being activated (MODE=1). Activate the component at a later time.
8093	No DP slave/PROFINET IO device is assigned to the address specified in LADDR (no configuration available) or the MODE parameter is not known.
8094	You have attempted to activate a device which is potential partner for a tool change port. However, another device is already activated on this tool change port at this time. The activated device will remain activated.
80A1	The addressed component could not have parameters assigned. (This error code is only possible when MODE = 1.) Note: "D_ACT_DP" returns this error information only if the activated slave/device fails again during parameter assignment. If the parameter assignment of a single module was unsuccessful, "D_ACT_DP" returns the error information W#16#0000.
80A2	The addressed DP slave does not return an acknowledgment. (This error information is not available with PROFINET IO devices. The process is not time-monitored by PROFINET.)
80A3	The DP master/PROFINET IO controller concerned does not support this function.
80A4	The CPU does not support this function for external DP masters/PROFINET IO controller.

Error code (W#16#...)	Explanation
80A6	Slot error in the DP slave/PROFINET IO device; not all user data can be accessed (this error code is only available when MODE=1). Note: "D_ACT_DP" returns this error information only if the activated component fails again after parameter assignment and before the end of "D_ACT_DP". If only a single module is unavailable, "D_ACT_DP" returns the error information W#16#0000.
80C1	D_ACT_DP" was started and will continue with another logical address (this error code is only possible when MODE=1 and MODE=2).
80C3	• Temporary resource error: The CPU is currently processing the maximum possible number of activation/deactivation jobs. (This error code is only possible when MODE = 1 and MODE = 2.) • The CPU is busy receiving a modified configuration. Currently you can not enable/disable DP slaves/ PROFINET IO devices.
80C5	DP: Jobs not collected by the user are discarded at restart.
80C6	PROFINET: Jobs not collected by the user are discarded at restart.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Application

If you configure DP slaves/PROFINET IO devices in a CPU which are not actually present or not currently required, the CPU will nevertheless continue to access these DP slaves/ PROFINET IO devices at regular intervals. After the slaves are deactivated, further CPU accessing will stop. With PROFIBUS DP, the fastest possible DP bus cycle can be achieved in this manner and the corresponding error events no longer occur.

Examples

From a machine OEM's point of view, there are numerous device options possible in series production of machines. However, each delivered machine includes only one combination of selected options.

Every one of these possible machine options is configured as a DP slave/PROFINET IO device by the manufacturer in order to create and maintain a common user program having all possible options. Use "D_ACT_DP" to deactivate all DP slaves/PROFINET IO devices not present at machine startup.

A similar situation exists for machine tools having numerous tooling options available but actually using only a few of them at any given time. These tools are implemented as DP slaves/ PROFINET IO devices. With "D_ACT_DP", the user program activates the tools currently needed and deactivates those required later.

Functional description

"D_ACT_DP" is an instruction that works asynchronously, which means its execution extends over multiple calls. You start the job by calling D_ACT_DP with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186)

Identification of a job

If you have started a deactivation or activation job and you call "D_ACT_DP" again before the job is complete, the behavior of the instruction depends on whether or not the new call involves the same job: If the input parameter LADDR matches, then the call will be interpreted as a follow-on call.

Deactivating DP slaves/PROFINET IO devices

When you deactivate a DP slave/PROFINET IO device with "D_ACT_DP", its process outputs are set to the configured substitute values or to 0 (safe state). The assigned DP master/PROFINET IO controller does not continue to address this component. Deactivated DP slaves/PROFINET IO devices are not identified as faulty or missing by the error LEDs on the DP master/PROFINET IO controller or CPU.

The process image of the inputs of deactivated DP slaves/PROFINET IO devices is updated with 0, that is, it is handled just as it is for failed DP slaves/PROFINET IO devices.

If you are using your program to directly access the user data of a previously deactivated DP slave/PROFINET IO device, the I/O access error OB (OB 122) is called, and the corresponding start event is entered in the diagnostics buffer. If you attempt to access a deactivated DP slave/PROFINET IO device via an instruction (such as "RD_REC (Page 5256)"), you will receive the same error information in RET_VAL as for an unavailable DP slave/PROFINET IO device.

Deactivating a DP slave/PROFINET IO device does not start the program error OB (OB 85), even if its inputs or outputs belong to the system-side process image to be updated. Also there is no entry in the diagnostics buffer.

Whether the deactivation of a DP slave/PROFINET IO device leads to a start of the rack failure OB (OB 86) and an entry in the diagnostics buffer, depends on the MODE parameter:

- MODE = 2: No OB 86 start and no diagnostics buffer entry
- MODE = 4: OB 86 start and diagnostics buffer entry

If a DP station/PNIO station fails after you have deactivated it with "D_ACT_DP", the operating system does not detect the failure. As a result, there is no subsequent start of OB 86 or diagnostics buffer entry.

Applies to PROFIBUS DP: If you wish to deactivate DP slaves functioning as transmitters in slave-to-slave communication, we recommend that you first deactivate the receivers (listeners) that detect which input data the transmitter is transferring to its DP master. Deactivate the transmitter only after you have performed this step.

Activating DP slaves/PROFINET IO devices

When you reactivate a DP slave/PROFINET IO device with "D_ACT_DP", this component is configured and assigned parameters by the associated DP master/PROFINET IO controller (as with the return of a failed DP station/PROFINET IO station). This activation is complete when the component is able to transfer user data.

Activating a DP slave/PROFINET IO device does not start the program execution error OB (OB 85), even if its inputs or outputs belong to the system-side process image to be updated. Also there is no entry in the diagnostics buffer.

Whether the deactivation of a DP slave/PROFINET IO device leads to a start of the rack failure OB (OB 86) and an entry in the diagnostics buffer, depends on the MODE parameter:

- MODE = 1: No OB 86 start and no diagnostics buffer entry
- MODE = 3: OB 86 start and diagnostics buffer entry

If you attempt to use "D_ACT_DP" to activate a DP slave that has been deactivated and is physically separated from the DP bus, the instruction will return the error code W#16#80A2 after approximately one minute and the DP slave will remain deactivated. If the slave is reconnected to the DP bus at a later time, it must be reactivated with "D_ACT_DP".

If you attempt to activate a PROFINET IO device that is physically separated from the PROFINET bus, "D_ACT_DP" will remain active. There is no automatic cancellation after a specific period as with DP slaves. You need to manually cancel the running job.

Note

Activating a DP slave/PROFINET IO device may be time-consuming. If you want to cancel a currently running activation job, start "D_ACT_DP" with the same value for LADDR and MODE =2. You repeat the call for "D_ACT_DP" with MODE = 2 until the successful cancellation of the activation job is displayed with RET_VAL =0.

If you wish to activate DP slaves which take part in the slave-to-slave communication, we recommend that you first activate the transmitters and then the receivers (listeners).

CPU startup

Depending on the startup mode, the CPU operating system behaves as follows regarding DP slaves/PROFINET IO devices:

- During cold and warm restarts, slaves/devices are activated automatically.
  - With S7-400: Activation of the DP slaves/PROFINET IO devices may take some time. In this case, the CPU continues to run and I/O access errors occur until the activation is complete. The reaction of the CPU depends on the configuration (OB 85 call for I/O access errors) and the program in OB 85. You cannot suppress these I/O access errors.
  - For S7-300: Activation of the DP slaves/PROFINET IO devices may take some time. In this case, the CPU waits until the DP slaves / PROFINET IO devices are activated. The maximum waiting period for activation of the DP slaves / PROFINET IO devices is approx. 1 minute. No I/O access errors occur during this time.
- During a hot restart, the activation status of slaves/devices remains unchanged: Activated slaves/devices remain activated, deactivated slaves/devices remain deactivated.

After the CPU start-up, the CPU cyclically attempts to contact all configured and not deactivated slaves/devices that are either not present or not responding.

Note

A call of "D_ACT_DP" in startup OBs is not supported.

Additional instructions

RD_REC: Read data record from I/O

Description

Use the instruction to read the data record with the number RECNUM from the addressed module. You start the read process by assigning the value "1" to the input parameter REQ during the call. If the read process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY . If BUSY has the value "1", reading is not yet complete.

See also: Difference between synchronous and asynchronous instructions (Page 4186). The data record read is entered in the destination area spanned by the RECORD parameter, providing the data transfer was free of errors.

Note

When you read out a data record with a number greater than one from an FM or a CP you have purchased prior to February 1997 (below referred to as "old modules"), "RD_REC" responds differently than it does in new modules. This special situation is covered in the section "Using old S7-300 FMs and CPs with data record numbers >1" (see below).

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not read any data records from the I/O modules in the user program with "RD_REC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: Set the interface of the DP master to "DPV1".

Parameter

The following table shows the parameters of the instruction "RD_REC":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Read request
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, you will have to specify the area identifier of the lower address. If the addresses are identical, you will have to specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical address of the module. For a mixed module, the lower of the two addresses must be specified.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. In addition: the length of the data record actually transferred in bytes (possible values: +1 to +240), if the destination area is greater than the transferred data record and if no error occurred during the transfer.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
RECORD	Output	ANY	I, Q, M, D, L	Destination area for the data record read. With asynchronous execution of "RD_REC", make sure that the actual parameters of RECORD have the same length information for all calls. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RECORD

Note

If you want to ensure that the entire data record is always read, select a destination area with a length of 241 bytes. If the data transfer is error-free, RET_VAL contains the actual data record length.

Using old S7-300 FMs and CPs with data record numbers > 1

If you want to use the instruction "RD_REC" to read out a data record with a number greater than one from an old S7-300 FM or old S7-300 CP, remember the following points:

- If the destination area is greater than the actual length of the required data record, no data are entered in RECORD . RET_VAL is written with W#16#80B1.
- If the destination area is smaller than the actual length of the required data record, the CPU reads as many bytes beginning at the start of the record as are specified in the length information of RECORD and enters this number of bytes inRECORD . RET_VAL has the value "0".
- If the length specified in RECORD is the same as the actual length of the required data record, the CPU reads the data record and enters it in RECORD . RET_VAL is written with "0".

Parameter RET_VAL

- If an error has occurred while the function was being executed, the return value contains an error code.
- If no error occurred during the transfer, RET_VAL contains the following:
  - 0, if the entire destination area was filled with data from the selected data record (the data record can also be incomplete).
  - The length of the data record actually transferred in bytes (possible values: +1 to +240) if the destination area is greater than the transferred data record.

  Note

  If the general error W#16#8745 occurs, this only indicates that access to at least one byte of the process image was blocked. The data record was read by the module correctly and written to the I/O memory area.

When looking at the "real" error information (error codes W#16#8xyz) in the following table, we distinguish between two cases:

- Temporary errors (error codes W#16#80A2 to 80A4, 80Cx):
  This type of error can possibly be eliminated without user action, meaning it is helpful to call the instruction again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#809x, 80A0, 80A1, 80Bx):
  This type of error does not correct itself. A new call of the instruction will not be successful until you have eliminated the error. Example of a permanent error: Wrong length specification in RECORD (W#16#80B1).

Note

If you transfer data records to a DPV1 slave with "WR_REC (Page 5260)" or if you read data records from a DPV1 slave with RD_REC and if this DPV1 slave operates in DPV1 mode, the DP master evaluates the error information it received from the slave as follows:

If the error information lies within the range from W#16#8000 to W#16#80FF or W#16#F000 to W#16#FFFF, the DP master passes the error information to the instruction. If the error information is outside this range, the DP master passes the value W#16#80A2 to the instruction and suspends the slave.

For a description of the error information originating from DPV1 slaves, refer to STATUS[3] Parameter STATUS (Page 5235).

Parameter RET_VAL for WR_REC and RD_REC

Error code (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer started; BUSY has the value 1.	Distributed I/O

Error code (W#16#...)	Explanation	Restriction
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value 1.	Distributed I/O
8090	Specified logical base address invalid: There is no assignment in SDB1/SDB2x or there is no base address.	-
8092	A type other than BYTE is specified in the ANY reference.	For S7-400 only
8093	This instruction is not permitted for the module specified by LADDR and IOID . (permitted are S7-300 modules for an S7-300, S7-400 modules for an S7-400, S7-DP modules for an S7-300 and S7-400).	-
80A0	Negative acknowledgement when reading from the module: Module was removed during the reading process or is defective	With "RD_REC" only
80A1	Negative acknowledgement when writing to the module: Module was removed during the writing process or is defective	With "WR_REC (Page 5260)" only
80A2	• DP protocol error at layer 2 (for example, slave failure or bus problems) • For ET200S, data record cannot be read in DPV0 mode.	Distributed I/O
80A3	DP protocol error with user interface/user	Distributed I/O
80A4	Communication on the communication bus disrupted	Error occurs between the CPU and the external DP interface module.
80B0	• Instruction not possible for module type. • The module does not recognize the data record. • Data record number 241 not permitted. • With "WR_REC (Page 5260)", data records 0 and 1 are not permitted.	-
80B1	The length specified in parameter RECORD is incorrect.	• With "WR_REC (Page 5260)": Length incorrect • With "RD_REC" (only when using old S7-300 FMs and S7-300 CPs): specified length > data record length • With DPNRM_DG: specified length > record length
80B2	The configured slot is not occupied.	-
80B3	Actual module type does not match the specified module type in SDB1.	-
80B7	DP slave or module reports an invalid range for a parameter or value.	With "RD_REC" only

Error code (W#16#...)	Explanation	Restriction
80C0	With "WR_REC (Page 5260)": the data can only be written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You can only write the data online with PG/PC. With "RD_REC": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. In this case, you can only read the data online with PG/PC. With "DPNRM_DG (Page 5287)": There are no diagnostics data available.	With "WR_REC (Page 5260)" or "RD_REC" or "DPNRM_DG (Page 5287)"
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	-
80C4	Internal temporary error. Job could not be carried out. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available.	Distributed I/O
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)	-

WR_REC: Write data record to I/O

Description

Use the instruction "WR_REC" to transfer the data record RECORD to the addressed module.

You start the writing process by assigning the value "1" to the input parameter REQ during the call. If the writing process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY. If BUSY has the value "1", writing is not yet complete.

Note

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not write any data records from the I/O modules to the user program with "WR_REC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: set the interface of the DP master to "DPV1".

Parameters

The following table shows the parameters of the instruction "WR_REC":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Write request
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, you will have to specify the area identifier of the lower address. If the addresses are identical, you will have to specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical address of the module. For a mixed module, the lower of the two addresses must be specified.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 2 to 240)
RECORD	Input	ANY	I, Q, M, D, L	Data record. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
RET_VAL	Output	Return	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY= 1: The writing process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RECORD

The data to be transferred are read from the RECORD parameter during the first call. If the transfer of the data record takes longer than the duration of one call, the contents of the RECORD parameter are no longer relevant for the subsequent instruction calls (for the same job).

Parameter RET_VAL

See also: RD_REC: Read data record from I/O (Page 5256)

Note

If the general error W#16#8544 occurs, it only indicates that access to at least one byte of the I/O memory area containing the data record was denied. The data transfer was continued.

See also

Difference between synchronous and asynchronous instructions (Page 4186)

DPRD_DAT: Read consistent data of a DP standard slave

Description

You use the instruction "DPRD_DAT" to consistently read data of a DP standard slave/ PROFINET IO device; the maximum length is as follows:

- See the documentation supplied with your CPU for the maximum length.
- If there was no error during the data transmission, the data that have been read are entered in the destination area indicated by RECORD.

The destination area must have the same length that you configured for the selected module. If you read from a DP standard slave with a modular configuration or with several DP identifiers, you can only access the data of one module/DP identifier at the configured start address per "DPRD_DAT" call.

Note

CPUs of the S7-300/400 product range support up to 64 bytes of consistent data. Use instruction "DPRD_DAT" to handle consistent data ranges that exceed 4 bytes. However, you may also use this instruction for a data range of 1 byte. Access errors trigger the output of error code W#16#8090.

Note

S7-400 CPUs as of firmware version V6.0

For S7-400 CPUs as of firmware version V6.0, the call of DPRD_DAT is generally accepted for all addresses. This means you can use DPRD_DAT to access addresses for which you have not configured overall consistency.

Parameter

The following table shows the parameters of the instruction "DPRD_DAT":

Parameters	Declaration	Data type	Memory area	Description
LADDR	Input	WORD	I, Q, M, D, L or constant	Configured start address from the I-area of the module from which the data will be read. Note: The address must be specified in hexadecimal form. For example, start address 100 means: LADDR:=W#16#64.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	ANY	I, Q, M, D, L	Destination area for the user data that were read. It must be precisely as long as you have configured for the selected module. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For more information about the valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. For a description of this error information, refer to STATUS[3] Parameter STATUS (Page 5235).

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	• You have not configured a module for the specified logical base address. • You have ignored the restriction concerning the length of consistent data. (For S7-400 CPUs only up to and including firmware version V5.3; this cause becomes obsolete as of firmware version V6.0.) • You have not entered the start address in the LADDR parameter in hexadecimal format.
8092	A type other than BYTE is specified in the ANY reference.
8093	No DP module/PROFINET IO device from which you can read consistent data exists at the logical address specified in LADDR.

Error code (W#16#...)	Explanation
80A0	Access error detected while I/O devices were being accessed.
80B0	• For S7-300 CPUs (all firmware versions) and S7-400 CPUs up to and including firmware version V5.3: Slave failure on external DP interface module. • For S7-400 CPUs as of firmware version V6.0: Slave failure or access error to non-consistent data at external DP connection (CP also signals to CPU that an error has occurred.)
80B1	The length of the specified destination area is not identical to the configured user data length.
80B2	System error with external DP interface module.
80B3	System error with external DP interface module.
80C0	The data haven't yet been read by the module.
80C2	System error with external DP interface module.
80Fx	System error with external DP interface module.
87xy	System error with external DP interface module.
808x	System error with external DP interface module.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Application

You require "DPRD_DAT" because you can only read out a maximum of four continuous bytes using the load commands that access the I/O or the process image input table.

Note

If required, you can also read consistent data via the process image of the inputs.

- See the associated documentation to determine whether your S7-300 CPU handles this functionality.
- All S7-400 CPUs handle this functionality.

CAUTION

I/O access

When using "DPRD_DAT", avoid accessing I/O areas that have process image partitions with OB6x connections (isochronous mode interrupts) assigned to them.

Data consistency

See also: Section Data consistency (Page 6134).

DPWR_DAT: Write consistent data of a DP standard slave

Description

Use the instruction to consistently transfer the data in RECORD to the addressed DP standard slave/PROFINET IO device, and, if necessary, to the process image (if you have configured the relevant address area of the DP standard slave as a consistent range in a process image).

The following applies for the maximum length of the data that will be transferred: See the documentation supplied with your CPU for the maximum length. The data is transferred synchronously, that is, the write process is completed when the instruction is completed.

The source area must have the same length that you have configured for the selected module. If the DP standard slave has a modular design, you can only access one module of the DP slave.

Note

CPUs of the S7-1200/1500 product range support up to 64 bytes of consistent data. Use instruction "DPWR_DAT" to write consistent data ranges that exceed 4 bytes to a submodule or device. However, you may also use this instruction for a data range of 1 byte. Access errors trigger the output of error code W#16#8090.

Note

S7-400 CPUs as of firmware version V6.0

For S7-400 CPUs as of firmware version V6.0, the call of DPWR_DAT is generally accepted for all addresses. This means you can use DPWR_DAT to access addresses for which you have not configured overall consistency.

Parameter

The following table shows the parameters of the instruction "DPWR_DAT":

Parameters	Declaration	Data type	Memory area	Description
LADDR	Input	WORD	I, Q, M, D, L or constant	Configured start address from the PIQ area of the module to which the data is written. Note: The address must be specified in hexadecimal form. For example, start address 100 means: LADDR:=W#16#64.
RECORD	Input	ANY	I, Q, M, D, L	Source area for the user data to be written. It must be precisely as long as you have configured for the selected module. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For more information about the valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. A description of this error information is available in STATUS[3] Parameter STATUS (Page 5235).

Error code (W#16#...)	Explanation
0000	No error occurred.
808x	System error with external DP interface module.
8090	• You have not configured a module for the specified logical base address. • You have ignored the restriction concerning the length of consistent data. (For S7-400 CPUs only up to and including firmware version V5.3; this cause becomes obsolete as of firmware version V6.0.) • You have not entered the start address in the LADDR parameter in hexadecimal format.
8092	A type other than BYTE is specified in the ANY reference.
8093	No DP module/PROFINET IO device to which you can write consistent data exists at the logical address specified in LADDR.
80A1	Access error detected while I/O devices were being accessed.

Error code (W#16#...)	Explanation
80B0	• For S7-300 CPUs (all firmware versions) and S7-400 CPUs up to and including firmware version V5.3: Slave failure on external DP interface module. • For S7-400 CPUs as of firmware version V6.0: Slave failure or access error to non-consistent data at external DP connection (CP also signals to CPU that an error has occurred.)
80B1	The length of the specified destination area is not identical to the configured user data length.
80B2	System error with external DP interface module.
80B3	System error with external DP interface module.
80C1	The data of the previous write job on the module have not yet been processed by the module.
80C2	System error with external DP interface module.
80Fx	System error with external DP interface module.
85xy	System error with external DP interface module.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Application

You require the instruction "DPWR_DAT" because you can only write a maximum of four continuous bytes using the transfer commands that access the I/O or the process image output table.

Note

If required, you can also read consistent data via the process image of the inputs.
See the associated documentation to determine whether your S7-300 CPU handles this functionality,

- All S7-400 CPUs handle this functionality.
- Do not use both possibilities concurrently when writing consistent data: Either use "DPWR_DAT" or write via the process image output table.

CAUTION

I/O access

When using "DPWR_DAT", avoid accessing I/O areas that have process image partitions with OB6x connections (isochronous mode interrupts) assigned to them.

Data consistency

See also: Data consistency (Page 6134).

I-device / I-slave

RCVREC: Receive data record

Description

An I-device can receive a data record from a superordinate controller. The receipt takes place in the user program with the instruction "RCVREC" (receive record).

The instruction has the following operating modes:

- Check whether the I-device has a request for a data record receipt.
- Make the data record available to the output parameters.
- Send an answer to the superordinate controller.

You can determine the operating mode executed by the instruction using the input parameter MODE (see below).

The I-device must be in the operating mode RUN or STARTUP.

With MLEN, you specify the maximum number of bytes you want to receive. The selected length of the destination area RECORD should have at least the length of MLEN bytes.

If a data record was received (MODE=1 or MODE=2), the output parameter NEW will indicate that the data record was stored in RECORD. Note that RECORD has a sufficient length. The output parameter LEN contains the actual length of the data record received in bytes.

Set CODE1 and CODE2 to zero for the positive answer to the superordinate controller. If the received data record is to be rejected, enter the negative answer to the superordinate controller in Error Code 1 of the CODE1 and in Error Code 2 of the CODE2.

Note

If the I-device has received a request for a data record receipt, you must recognize the delivery of this request within a certain duration. After recognition, you must send an answer to the superordinate controller within this time period. Otherwise the I-device will experience a timeout error which causes the operating system of the I-device to send a negative answer to the superordinate controller. For information on the value for the time period, please refer to the specifications of your CPU.

The output parameter STATUS receives the error information after the occurrence of an error.

Operating modes

You can determine the operating mode of the instruction "RCVREC" with the input parameter MODE. This step will be explained in the following table.

MODE	Meaning
0	Check whether a request for a data record receipt exists If a data record from a superordinate controller exists on the I-device, the instruction will only write the output parameters NEW, SLOT, INDEX and LEN. If you call the instruction several times with MODE=0, then the output parameter will only refer to one and the same request.
1	Receiving a data record for any subslot of the I-device If a data record from a superordinate controller exists on the I-device for any subslot of the I-device, the instruction will write the output parameter and transfer the data record to the parameter RECORD.
2	Receiving a data record for a specific subslot of the I-device If a data record from a superordinate controller exists on the I-device for a specific subslot of the I-device, the instruction will write the output parameter and transfer the data record to the parameter RECORD.
3	Sending a positive answer to the superordinate controller The instruction checks the request of the superordinate controller to receive a data record, accepts the existing data record and sends a positive acknowledgement to the superordinate controller.
4	Sending a negative answer to the superordinate controller The instruction checks the request of the superordinate controller to receive a data record, rejects the existing data record and sends a negative acknowledgement to the superordinate controller. Enter the reason for the rejection in the input parameters CODE1 and CODE2.

Note

After the receipt of a data record (NEW=1) you must call "RCVREC" twice to ensure complete processing. You must do this in the following order:

- First call with MODE=1 or MODE=2
- Second call with MODE=3 or MODE=4

Parameter

The following table shows the parameters of the instruction "RCVREC":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	DWORD	I, Q, M, D, L or constant	Subslot in the transfer area of the I-device for the data record to be received (only relevant for MODE=2). The high word is always set to zero.
MLEN	Input	INT	I, Q, M, D, L or constant	Maximum length of the data record to be received in bytes
CODE1	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 1 (for MODE=4)
CODE2	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 2 (for-MODE=4)

Parameters	Declaration	Data type	Memory area	Description
NEW	Output	BOOL	I, Q, M, D, L	• MODE=0: New data record was received • MODE=1 or 2: Data record was transferred to RECORD
STATUS	Output	DWORD	I, Q, M, D, L	Error information
SLOT	Output	INT	I, Q, M, D, L	Identical to F_ID
SUBSLOT	Output	INT	I, Q, M, D, L	Identical to F_ID
INDEX	Output	INT	I, Q, M, D, L	Number of the data record received
LEN	Output	INT	I, Q, M, D, L	Length of the data record received
RECORD	InOut	ANY	I, Q, M, D, L	Destination area for the data record received. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 Byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and results in an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter STATUS

For information about the interpretation of parameter STATUS, refer to section: Parameter STATUS (Page 5235)

PRVREC: Make data record available

Description

An I-device can receive a request from a superordinate controller to make a data record available. The I-device makes the data record available in the user program with the instruction "PRVREC" (provide record).

The instruction has the following operating modes:

- Check whether the I-device has a request for making a data record available.
- Transfer the requested data record to the superordinate controller.
- Sending an answer to the superordinate controller.

You can determine the operating mode executed by the instruction using the input parameter MODE(see below).

The I-device must be in the operating mode RUN or STARTUP.

Enter the maximum number of bytes the data record to be sent should have with LEN. The selected length of the destination area RECORD should have at least the length of LEN bytes.

If a request to make a data record available exists, (MODE=0), the output parameter NEW will be set to TRUE.

If the request for making a data record available is accepted, write RECORD for the positive answer to the superordinate controller with the requested data record and write zero for CODE1 and CODE2. If the request for making a data record available is to be rejected, enter the negative answer to the superordinate controller in Error Code 1 of the CODE1 and in Error Code 2 of theCODE2.

Note

If the I-device has received a request for making a data record available, you must recognize the delivery of this request within a certain time period. After recognition, you must send an answer to the superordinate controller within this time period. Otherwise the I-device will experience a timeout error which causes the operating system of the I-device to send a negative answer to the superordinate controller. For information on the value for the time period, please refer to the specifications of your CPU.

The output parameter STATUS receives the error information after the occurrence of an error.

Operating modes

You can determine the operating mode of the instruction "PRVREC" with the input parameter MODE. This step will be explained in the following table.

MODE	Meaning
0	Check whether a request for making a data record available exists If a request from a superordinate controller for making a data record available exists on the I-device, the instruction will only write the output parameters NEW, SLOT, INDEX and RLEN. If you call the instruction several times with MODE=0, then the output parameter will only refer to one and the same request.
1	Receiving a request for making a data record available for any subslot of the I-device If such a request from a superordinate controller for any subslot of the I-device exists on the I-device, the instruction will write the output parameter.
2	Receiving a request for making a data record available for a specific subslot of the I-device If such a request from a superordinate controller for a specific subslot of the I-device exists on the I-device, the instruction will write the output parameter.
3	Make the data record available and send a positive answer to the superordinate controller The instruction checks the request of the superordinate controller to make a data record available, makes the request data record available to RECORD and sends a positive acknowledgement to the superordinate controller.
4	Sending a negative answer to the superordinate controller The instruction checks the request of the superordinate controller to make a data record available, rejects this request and sends a negative acknowledgement to the superordinate controller. Enter the reason for the rejection in the input parameters CODE1 and CODE2.

Note

After the receipt of a request (NEW=1) you must call the instruction twice to ensure complete processing. You must do this in the following order:

- First call with MODE=1 or MODE=2
- Second call with MODE=3 or MODE=4

Parameter

The following table shows the parameters of the instruction "PRVREC":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	DWORD	I, Q, M, D, L or constant	Subslot in the transfer area of the I-device for the data record to be sent (only relevant for MODE=2). The high word is always set to zero.
CODE1	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 1 (for-MODE=4)
CODE2	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 2 (for-MODE=4)
LEN	Input	INT	I, Q, M, D, L or constant	Maximum length of the data record to be sent in bytes
NEW	Output	BOOL	I, Q, M, D, L	The new data record was requested by the superordinate controller.
STATUS	Output	DWORD	I, Q, M, D, L	Error information
SLOT	Output	INT	I, Q, M, D, L	Identical to F_ID
SUBSLOT	Output	INT	I, Q, M, D, L	Identical to F_ID
INDEX	Output	INT	I, Q, M, D, L	Number of the data record to be sent
RLEN	Output	INT	I, Q, M, D, L	Length of the data record to be sent
RECORD	InOut	ANY	I, Q, M, D, L	Data record made available Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 Byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and results in an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter STATUS

For information about the interpretation of parameter STATUS, refer to section: Parameter STATUS (Page 5235)

SALRM: Send interrupt

Description SALRM

Description

You use the instruction to send an interrupt of a slot in the transition area (virtual slot) from the user program of an intelligent slave to the associated DP master. This starts the associated OB in the DP master.

You can send additional interrupt-specific information along with the interrupt. You can use the instruction to read all additional information in the DP master.

The instruction can only be used in S7-compatible mode.

- DP: The master uses a GSD file to integrate the I-slave.
- S7-compatible: The I-slave is connected to a master during configuration.

Note

The interface of the "SALRM" instruction is identical to that in the "SALRM" FB defined in the standard "PROFIBUS and PROFINET Guideline Communication Function Blocks on PROFIBUS DP and PROFINET IO".

Functional description

The "SALRM" instruction works asynchronously, that is, its execution extends over multiple calls. You start the interrupt transfer by calling the instruction with REQ = 1.

The sending process remains active until interrupt processing is either acknowledged or canceled by the DP master.

The output parameter BUSY and bytes 2 and 3 of the output parameter STATUS show the status of the job. Bytes 2 and 3 of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Difference between synchronous and asynchronous instructions (Page 4186)

The transfer of the interrupt is complete when the output parameter BUSY has the value FALSE .

Identification of the job

When you initiate the transfer of an interrupt to the DP master with the instruction "SALRM" and then call this instruction once again before the current job has been completed, the further response of this instruction will depend on whether or not the new call involves the same job.

If the ID and ATYPE parameters match a still unfinished job, the new call is regarded as a follow-on call.

Parameters

The following table shows the parameters of the "SALRM" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Transfer the interrupt
ID	Input	DWORD	I, Q, M, D, L or constant	Any logical address of the transition area to the DP master (virtual slots), as viewed from the DP slave, except the diagnostics address of the station and the logical address of slot 2. The relevant information is found in the low word. Enter zero in the high word. Bit 15 contains the I/O ID: 0 = input address, 1 = output address.
ATYPE (Page 5276)	Input	INT	I, Q, M, D, L or constant	Interrupt type ID for the interrupt type. Possible values: • 1: Diagnostics interrupt • 2: Process interrupt
ASPEC (Page 5276)	Input	INT	I, Q, M, D, L or constant	Interrupt specifier: • 0: No further information • 1: Incoming event, faulty slot • 2: Outgoing event, slot no longer faulty • 3: Outgoing event, slot still faulty
LEN (Page 5277)	Input	INT	I, Q, M, D, L or constant	Length of additional interrupt information to be sent, in bytes Highest value: 16
DONE	Output	BOOL	I, Q, M, D, L	DONE=1: Interrupt was transferred
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: Interrupt transfer is not yet complete
ERROR (Page 5277)	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error has occurred.
STATUS (Page 5277)	Output	DWORD	I, Q, M, D, L	Error information
AINFO (Page 5278)	InOut	ANY	I, Q, M, D, L	Interrupt info Source area for additional interrupt information Note: Note that for S7-300 CPUs, the parameter AINFO always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Effect of the "SALRM" call on the module status information and the group error LED (SF)

Like every CPU, an intelligent slave has stored the properties of its slots in the module status information.

See also: SZL-ID W#16#xy91 - module status information (Page 5482)

When you use "SALRM" to send a diagnostic interrupt, the operating system at the intelligent slave influences the slave-local module status information and the SF LED based on bit 0 in byte 0 in AINFO (this bit is added to the status information as "module faulty"). However, there is no diagnostics buffer entry made in the I-slave, and no diagnostics interrupt OB is started.

Consistency of module status information between DP master and I-slave

The following section presents different scenarios and discusses their effects on module status information:

- Station return (results in start of OB 86 in the DP master and in the I-slave)
  The module status information will be affected with I-slave and with S7 master ("module fault" will be reset). If, after a station return, there are faults at the I-slave from the point of view of the user, these must be reported to the DP master by means of an "SALRM" call.
- STOP-RUN transition of the DP master (results in start of OB 82 in the I-slave)
  The module status information on the I-slave remains unchanged. The DP master resets the "module fault" bit in the relevant module status information.
  To ensure the consistency of module status information between the DP master and I-slave in S7-compatible mode, you must respond to the I-slave as follows:
  - For each error-free virtual slot, you use "SALRM" to send an outgoing diagnostics interrupt to the DP master.
  - For each faulty virtual slot, you use "SALRM" to send an incoming diagnostics interrupt to the DP master.
- STOP-RUN transition of the I-slave (results in start of OB 82 in the DP master)
  The module status information on the DP master remains unchanged and is reset on the I-slave ("module fault" will be reset).
  To ensure the consistency of module status information between the DP master and I-slave, you must react to the I-slave as follows:
  - For each error-free virtual slot, you use "SALRM" to send an outgoing diagnostics interrupt to the DP master.
  - For each faulty virtual slot, you use "SALRM" to send an incoming diagnostics interrupt to the DP master.

Note

Because "SALRM" functions asynchronously, the "SALRM" calls cannot be completed in the startup OBs. In other words, they must be run in the cyclical program until they are finished.

All the above-mentioned differences between the module status information in the master and the I-slave can only occur at those slots that receive diagnostics interrupts by means of "SALRM". This means that the remedies discussed above only apply to such slots.

Parameter ATYPE

Parameter ATYPE

For all permitted values of ATYPE , the following table shows which OB is started in the associated DP master and in which DP mode the given interrupt type is permitted.

ATYPE	Meaning in DPV1 standard	Associated OB in the S7 DP master	DP mode	
			DP	S7-compatible
1	Diagnostics interrupt	Diagnostics interrupt OB (OB82)	–	Yes
2	Process interrupt	Process interrupt OBs (OBs 40 to 47)	–	Yes

Note

In addition to the information given in the table above, the extent to which interrupt types can be used can also be restricted by the DP master.

Dependency of the interrupt type on the operating mode of the S7 master

For a slave in S7-compatible mode (operated on an S7 master), hardware and diagnostics interrupts can only be sent when the associated DP master is in RUN mode (DP: Operate). If the DP master is in STOP mode (DP: Clear), the interrupt is not sent and "SALRM" provides the error information W#16#80C8. In this case, the user is responsible for sending the interrupt at a later time.

Parameter ASPEC

Parameter ASPEC

According to the applicable standard, this parameter shows the diagnostics status of the virtual slot. For this reason, you can assign a value other than zero to ASPEC only when sending a diagnostics interrupt.

Because the additional S7 interrupt information for a diagnostics interrupt (data record 0) contains incoming/outgoing information (see Diagnostics data>Main byte 0 bit 0), you must write bit 0 (module fault) in byte 0 of the additional interrupt information as follows:

ASPEC	"Module fault" bit in AINFO (Page 5278)
0	0
1	1
2	0
3	1

Parameter LEN

Parameter LEN

In LEN, you specify in bytes the length of the additional interrupt information to be sent. The maximum permitted range of values is 0 to 16.

The following table shows the values LEN can have in the individual modes of an intelligent slave for all possible interrupt types.

Interrupt type	DP	S7-compatible
Diagnostics interrupt	–	4 to 16
Process interrupt	–	4

The following table shows the response of "SALRM" when you assign LEN a value other than the length of AINFO (Page 5278) in BYTE.

Value of LEN	Characteristics of "SALRM"
<= length specification of AINFO	"SALRM" sends an interrupt to the DP master. The number of bytes of additional interrupt information transmitted is as specified in LEN.
Outside the permitted range of values (< 0 or > 16)	"SALRM" does not send an interrupt. Error information: W#16#80B1, STATUS[4]=B#16#FF
> Length specification of AINFO	"SALRM" sends an interrupt to the DP master. The amount of bytes of additional interrupt information transmitted is as specified in the length information of AINFO. Error information: W#16#00B1, STATUS[4]= length specification of AINFO

Parameters STATUS and ERROR

Parameters STATUS and ERROR

The STATUS output parameter contains error information. If it is interpreted as ARRAY[1 ... 4] OF BYTE, the error information has the following structure:

Array element	Meaning
STATUS[1]	• B#16#00: No error • B#16#C0: Error detected by I-slave
STATUS[2], STATUS[3]	Corresponds to the output parameter RET_VAL of instructions
STATUS[4]	B#16#00 with the exception of some length conflicts between LEN and the length of AINFO (Page 5278). These exceptions are shown in the next table.

The following table indicates all specific error information of "SALRM".

ERROR	STATUS[2,3] (W#16#...)	Explanation
0	0000	Job was executed without errors. If LEN is < length of AINFO, only LEN bytes of additional interrupt information were transmitted.
0	00B1	LEN > length of AINFO. The job was completed. The additional interrupt information specified in AINFO was transmitted. STATUS[4] contains the length of AINFO.
0	7000	Initial call with REQ=0 (empty cycle). No interrupt was sent. BUSY has the value "0".
0	7001	Initial call with REQ=1. The job was initiated. BUSY has the value "1".
0	7002	Intermediate call (REQ irrelevant). The interrupt sent was not yet acknowledged by the DP master. BUSY has the value "1".
1	8090	The address specified in the ID is outside the permitted address range or was not configured.
1	8091	• You have disabled the interrupt in the configuration. • The interrupt is not permitted for this type of slave.
1	8092	Illegal data type in AINFO (BYTE and BLOCK-DB are permitted)
1	8093	ID belongs to a virtual slot from which an interrupt cannot be requested.
1	80B0	ASPEC • Does not match bit 0 in byte 0 of AINFO • Must have a value of 0 for the interrupt type used • Is outside the permitted range of values
1	80B1	LEN Is outside the permitted range of values. STATUS[4] contains B#16#FF.
1	80B5	Call of "SALRM" in the DP master is not permitted.
1	80C3	The required resources (memory, etc.) are occupied at this time.
1	80C5	Distributed I/O device is not available at this time (e.g., station failure)
1	80C8	The function is not permitted in the current DP master operating mode (the DP master is an S7 master and is in STOP mode).

Parameter AINFO

Parameter AINFO

The AINFO parameter is the source area for additional interrupt information. As far as the intelligent slave is concerned, you can write any values to this area. However, if you are using a DP master of the S7 family, the additional information sent along with the interrupt must conform to S7 conventions.

If you send a **diagnostics interrupt** (ATYPE (Page 5276) =1), then you are responsible for appropriate assignment of data record 0 and, if necessary, data record 1.

The following table shows you a recommendation for an S7-compatible assignment. In this case, the "module fault" bit (see above) was already set. Except for the indicated bit, this

recommendation corresponds to the default assignment (the one existing after a POWER UP, after a STOP-RUN transition of the intelligent slave, or a station return).

Data record no.	Assignment
0	B#16#01, 0B, 00, 00
1	For S7-compatible mode: data record 0 + 12 bytes with zero

Additional information

See also: Structure of diagnostic data (Page 5394).

PROFIBUS

DP_PRAL: Trigger hardware interrupt from DP standard slave

Description

Use the instruction to trigger a process interrupt on the associated DP master from the user program of an intelligent slave. This interrupt starts OB 40 on the DP master.

Using the input parameter AL_INFO , you can identify the cause of the process interrupt. This interrupt identifier is transferred to the DP master and you can evaluate the identifier in OB 40 (tag OB40_POINT_ADDR).

The requested process interrupt is uniquely specified by the input parameters IOID and LADDR . For each configured address area in the transfer memory, you can trigger exactly one process interrupt at any time.

Functional description

"DP_PRAL" works asynchronously, that is, its execution extends over multiple calls. You start the process interrupt request by calling "DP_PRAL" with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

The job is complete when execution of OB 40 is complete on the DP master.

Note

If you operate the DP slave as a standard slave, the job is complete as soon as the diagnostics frame is obtained by the DP master.

Identification of a job

The input parameters IOID and LADDR uniquely specify the job.

If you have called the "DP_PRAL" instruction on a DP slave and you call this instruction again before the DP master has acknowledged the requested process interrupt, the way in which

the instruction reacts depends on whether the new call involves the same job: If the parameters IOID and LADDR match a job that is not yet complete, the call is interpreted as a follow-on call regardless of the value of the parameter AL_INFO and the value W#16#7002 is entered in RET_VAL .

Parameter

The following table shows the parameters of the instruction "DP_PRAL":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ=1: Trigger process interrupt on the associated DP master
IOID	Input	BYTE	I, Q, M, D, L or constant	Identifier of the address range in the transfer memory (from the point of view of the DP slave): • B#16#00: Bit15 of LADDR specifies whether an input (Bit15=0) or output address (Bit15=1) will be present. • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, you will have to specify the area identifier of the lower address. If the addresses are identical, you will have to specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Start address of the address range in the transfer memory (from the point of view of the DP slave). If this is a range belonging to a mixed module, the lower of the two addresses must be specified.
AL_INFO	Input	DWORD	I, Q, M, D, L or constant	Interrupt ID This is transferred to OB 40 that is to be started on the associated DP master (tag OB40_POINT_ADDR). If you operate the intelligent slave with a remote master, you must evaluate the diagnostics frame on the master.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The triggered process interrupt has not yet been acknowledged by the DP master.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	Job was executed without errors.
7000	First call with REQ=0. No process interrupt request is active; BUSY has the value "0".
7001	First call with REQ=1. A process interrupt request has already been sent to the DP master; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): The triggered process interrupt has not yet been acknowledged by the DP master; BUSY has the value "1".
8090	Start address of the address range in the transfer memory is incorrect.
8091	Interrupt is blocked (block configured by user)
8093	The parameters IOID and LADDR address a module that is not capable of a process interrupt request.
80B5	Call in the DP master not permitted.
80C3	The required resources (memory, etc.) are occupied at this time.
80C5	Distributed I/O device is not available at this time (i.e. station failure).
80C8	The function is not permitted in the current DP master operating mode.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

DPSYC_FR: Synchronize DP slaves / Freeze inputs

Description

You use the instruction to synchronize one or more groups of DP slaves.

The function involves sending one of the control commands below, or a combination of them, to the relevant groups:

- SYNC (simultaneous output and freezing of output states on the DP slaves)
- UNSYNC (cancels the SYNC control command)
- FREEZE (freeze the input states on the DP slaves and read in the frozen inputs)
- UNFREEZE (cancels the FREEZE control command)

Before you send the control commands listed above, you must assign the DP slaves to groups per configuration. You must know which DP slave is assigned to which group, with which number, and know the reactions of the various groups to SYNC/FREEZE.

Note

Note that the control commands SYNC and FREEZE also remain valid when you perform a warm/cold restart.

Please note also that you may initiate only one SYNC/UNSYNC job or only one FREEZE/UNFREEZE job at a time.

Functional description

"DPSYC_FR" works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "DPSYC_FR" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

Identification of a job

If you have triggered a SYNC/FREEZE job and called "DPSYC_FR" again before the first job was completed, the response of the instruction depends on whether the new call is for the same job. If the input parameters LADDR, GROUP and MODE match, the call is interpreted as a follow-on call.

Writing outputs of DP modules

The writing of outputs of DP modules is triggered as follows:

- By transfer commands to the DP I/O
- By writing the process image output table to the modules (by the operating system at the end of OB 1 or by calling the instruction "UPDAT_PO (Page 5219)")
- Calling the "DPWR_DAT (Page 5265)" instruction.

In normal operation, the DP master transfers the output bytes cyclically (within the cycle of the PROFIBUS DP bus) to the outputs of the DP slaves.

If you want to have certain output data (possibly distributed on several slaves) applied to the outputs to the process at exactly the same time, you can send the SYNC control command to the relevant DP master using the "DPSYC_FR" instruction.

What are the effects of SYNC?

With the SYNC control command, the DP slaves of the selected groups are switched to Sync mode. In other words, the DP master transfers the current output data and instructs the DP slaves involved to freeze their outputs. With the following output message frames, the DP slaves enter the output data in an internal buffer and the state of the outputs remains unchanged.

Following each SYNC control command, the DP slaves of the selected groups apply the output data of their internal buffer to the outputs on the process.

The outputs are only updated cyclically again when you send the UNSYNC control command using the "DPSYC_FR" instruction.

Note

If the DP slaves of the selected group(s) are not currently connected to the network or have failed when the control command was sent, they will not be switched to SYNC mode. This information will not be communicated in the return value of the instruction.

Reading input data of DP modules

The input data of the DP modules are read as follows:

- Using load commands to the DP I/O
- When the process image input table is updated (by the operating system at the start of OB 1 or by calling the "UPDAT_PI (Page 5218)" instruction)
- By calling the "DPRD_DAT (Page 5262)" instruction.

In normal operation, the DP master receives this input data cyclically (within the cycle of the PROFIBUS DP bus) from its DP slaves and makes them available to the CPU.

If you want to have certain input data (possibly distributed on several slaves) to be read from the process at exactly the same time, send the FREEZE control command to the relevant DP master using the "DPSYC_FR" instruction.

What are the effects of FREEZE?

With the FREEZE control command, the DP slaves involved are switched to Freeze mode, in other words the DP master instructs the DP slaves to freeze the current state of the inputs. It then transfers the frozen data to the input area of the CPU.

Following each FREEZE control command, the DP slaves freeze the state of their inputs again.

The DP master only receives the current state of the inputs cyclically again after you have sent the UNFREEZE control command with the "DPSYC_FR" instruction.

Note

If the DP slaves of the selected group(s) are not currently connected to the network or have failed when the control command has been sent, they will not be switched to FREEZE mode. This information will not be communicated in the return value of the instruction.

Data consistency

Because the DPSYC_FR functions are asynchronous and can be interrupted by higher priority classes, you should make sure that the process images are consistent with the actual inputs and outputs of the I/O when using the "DPSYC_FR" instruction.

This is guaranteed if you comply with the following consistency rules:

- Define suitable process image partitions for the "SYNC outputs" and the "FREEZE inputs" (only possible on the S7-400). Call the "UPDAT_PO (Page 5219)" instruction immediately before each first call of a SYNC job. Call the "UPDAT_PI (Page 5218)" instruction immediately after the respective last call of a FREEZE job.
- As an alternative: Use only direct I/O access for outputs involved in a SYNC job and for inputs involved in a FREEZE job. Do not write to these outputs when a SYNC job is active, and do not read in these inputs when a FREEZE job is active.

Use of DPWR_DAT and DPRD_DAT

If you use the "DPWR_DAT (Page 5265)" instruction, it must be complete before you send a SYNC job for the outputs involved.

If you use the "DPRD_DAT (Page 5262)" instruction, it must be complete before you send a FREEZE job for the inputs involved.

Startup and "DPSYC_FR"

The user alone must take responsibility for sending the SYNC and FREEZE control commands in the startup OBs.

If you want the outputs of one or more groups to be in SYNC mode when the user program starts, you must initialize these outputs during startup and completely execute the "DPSYC_FR" instruction with the SYNC control command.

If you want the inputs of one or more groups to be in FREEZE mode when the user program starts, you must execute the "DPSYC_FR" instruction with the FREEZE control command completely for these inputs during startup.

Parameter

The following table shows the parameters of the instruction "DPSYC_FR":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Level-triggered control parameter REQ=1: Triggering of the SYNC/FREEZE job
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical address of the DP master
GROUP	Input	BYTE	I, Q, M, D, L or constant	Group selection Bit 0 = 1: group 1 selected Bit 1 = 1: group 2 selected : Bit 7 = 1: group 8 selected You can select several groups per job. The value B#16#0 is invalid.

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID (coding complies with EN 50 170 Volume 2, PROFIBUS) Bit 0: reserved (value 0) Bit 1: reserved (value 0) Bit 2: • = 1: UNFREEZE will be executed • = 0: no meaning Bit 3: • = 1: FREEZE will be executed • = 0: no meaning Bit 4: • = 1: UNSYNC will be executed • = 0: no meaning Bit 5: • = 1: SYNC will be executed • = 0: no meaning Bit 6: reserved (value 0) Bit 7: reserved (value 0) Possible values: • with exactly one ID per job: – B#16#04 (UNFREEZE) – B#16#08 (FREEZE) – B#16#10 (UNSYNC) – B#16#20 (SYNC) • with more than one ID per job: – B#16#14 (UNSYNC, UNFREEZE) – B#16#18 (UNSYNC, FREEZE) – B#16#24 (SYNC, UNFREEZE) – B#16#28 (SYNC, FREEZE)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. Make sure that you evaluate RET_VAL each time the block has been executed.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The SYNC/FREEZE job is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. For a description of this error information, refer to STATUS[3], STATUS (Page 5235) parameter.

Error code (W#16#...)	Explanation
0000	Job was executed without errors.
7000	First call with REQ=0. The job specified with LADDR, GROUP and MODE is not active; BUSY has the value 0.
7001	First call with REQ=1. The job specified with LADDR, GROUP and MODE was triggered; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). The activated SYNC/FREEZE job is still running; BUSY has the value 1.
8090	The module selected with LADDR is not a DP master.
8093	This instruction is not permitted for the module selected with LADDR (configuration or version of the DP master).
8094	GROUP parameter is incorrect
8095	MODE parameter is incorrect
80B0	The group selected with GROUP is not configured.
80B1	The group selected with GROUP is not assigned to this CPU.
80B2	The SYNC job specified with MODE is not permitted on the group selected with GROUP.
80B3	The FREEZE job specified with MODE is not permitted on the group selected with GROUP .
80C2	Temporary shortage of resources on the DP master: The DP master is currently processing the maximum number of jobs for a CPU.
80C3	This SYNC/UNSYNC job cannot be activated at present because only one SYNC/UNSYNC job can be triggered at a time. Check your user program.
80C4	This FREEZE/UNFREEZE job cannot be activated at present because only one FREEZE/UNFREEZE job can be triggered at a time. Check your user program.
80C5	Short circuit directly at DP interface
80C6	Job aborted due to I/O disconnection by CPU
80C7	Job aborted due to warm or cold restart on the DP master
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

DPNRM_DG: Read diagnostics data from a DP slave

Description

You use the "DPNRM_DG" instruction to read the current diagnostics data of a DP slave in the form specified in EN 50 170 Volume 2, PROFIBUS. The data that has been read is entered in the target range indicated by RECORD following error-free data transfer.

Refer to the following table for the basic structure of the slave diagnostics data and to the manuals of the DP slaves for further information.

Byte	Meaning
0	Station status 1
1	Station status 2
2	Station status 3
3	Master station number
4	Vendor ID (high byte)
5	Vendor ID (low byte)
6 ...	Additional slave-specific diagnostic information

Functional description

The reading process is executed asynchronously, in other words, it can extend over several calls. The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

Call

You start the read process by assigning the value "1" to the input parameter REQ when the instruction is called.

Parameter

The following table shows the parameters of the instruction "DPNRM_DG":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Read request
LADDR	Input	WORD	I, Q, M, D, L or constant	Configured diagnostic address of the DP slave Note: Address must be specified in hexadecimal form, for example, diagnostic address 1022 means: LADDR:=W#16#3FE.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error has occurred, the length of the data actually transferred is entered in RET_VAL .

Parameter	Declaration	Data type	Memory area	Description
RECORD	Output	ANY	I, Q, M, D, L	Destination area for the diagnostics data that were read. Only the BYTE data type is permitted. The minimum length of the data record to be read or the target range is 6. The maximum length of the data record to be sent is 240. Standard slaves can provide more than 240 bytes of diagnostics data up to a maximum of 244 bytes. In this case, the first 240 bytes are transferred to the target range and the overflow bit is set in the data. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The reading process is not yet complete.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

You will find information on data type conversion in the different programming languages under "Overview of data type conversion (Page 354)".

Parameter RECORD

The CPU evaluates the actual length of the read diagnostics data:

If the length specified for RECORD

- is less than the number of data bytes supplied, the data are discarded and a corresponding error code is entered in RET_VAL.
- is greater than or equal to the number of data bytes supplied, the data are accepted in the destination area and the actual length is entered in RET_VAL as a positive value.

Note

You must ensure that the actual parameters of RECORD correspond in all calls belonging to a job.

A job is uniquely identified by the LADDR input parameter.

RET_VAL parameter

- If an error occurs while the function is being executed, the return value contains an error code.
- If no errors have occurred during data transfer, RET_VAL contains the length of the data read in bytes as a positive number.

Note

The amount of data read in a DP slave depends on its diagnostics status.

For the evaluation of the error information of the RET_VAL parameter, refer to Evaluating errors with output parameter RET_VAL (Page 162). This chapter also contains the general error information for the instructions. The specific error information of "DPNRM_DG" is a subset of the error information of "RD_REC (Page 5256)".

Standard slaves with more than 240 bytes of diagnostics data

With standard slaves on which the number of standard diagnostics data is between 241 and 244 bytes, note the following points:

If the length specified for RECORD

- is less than 240 bytes, the data are discarded and a corresponding error code is entered in RET_VAL.
- If the length specified for RECORD is greater than or equal to 240 bytes, the first 240 bytes of the standard diagnostics data are transferred to the target range and the overflow bit is set in the data.

System resources for S7-400

When "DPNRM_DG" is called for a job that is not currently processed, CPU resources (memory space) will be occupied for S7-400. You can call "DPNRM_DG" for multiple DP slaves briefly one after the other if you do not exceed the maximum number of "concurrently" active "DPNRM_DG" jobs for your CPU.

If you activate several jobs "simultaneously," all the jobs will be executed without interfering with each other.

If you reach the limits of the system resources, this is indicated in RET_VAL. In this case, repeat the job.

See also

Overview of data type conversion (Page 354)

DP_TOPOL: Determine topology for DP master system

Description

You use the instruction to trigger the topology determination for a selected DP master system. Calling the instruction will address all diagnostics repeaters on a DP master system.

Note

The topology can only be determined for one DP master system at a time.

Topology determination is the prerequisite for the detailed display of the error location if line errors occur. After configuration and after every change to the physical configuration of a DP master system, you must repeat the topology determination with "DP_TOPOL".

Changes to the physical configuration are:

- Changes in line lengths
- Adding or removing stations or components with repeater function
- Changing station addresses

If an error is reported by a diagnostics repeater, "DP_TOPOL" writes the DPR and DPRI outputs for the duration of one "DP_TOPOL pass. If errors are reported by multiple diagnostics repeaters of the selected DP master system, "DP_TOPOL" writes DPR and DPRI with information regarding the first diagnostics repeater reporting the error. You can read out the entire diagnostic information on the programming device or using the "DPNRM_DG (Page 5287)" instruction. If no diagnostics repeater reports an error, the DPR and DPRI outputs have the value NULL.

If you want to repeat a topology determination after an error occurs, you must first reset "DP_TOPOL". This is done by calling "DP_TOPOL" with REQ=0 and R=1.

Functional description

"DP_TOPOL" works asynchronously, that is, its execution extends over multiple calls. You start determining the bus topology by calling "DP_TOPOL" with REQ=1 . If you want to cancel the process, call "DP_TOPOL" with R=1 .

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

Note

Determining the topology may take several minutes.

Identification of a job

The input parameter DP_ID uniquely specifies a job.

If you have called "DP_TOPOL" and you call this instruction again before the topology is re-determined, the manner in which the instruction reacts depends on whether the new call

involves the same job: If the DP_ID parameter matches a job that has not yet been completed, then the call is interpreted as a follow-up call and the value W#16#7002 will be entered in RET_VAL. On the other hand, if another job is involved, the CPU rejects it.

Parameters

The following table shows the parameters of the "DP_TOPOL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ=1: Trigger topology determination
R	Input	BOOL	I, Q, M, D, L	R=1: Cancel topology determination
DP_ID	Input	INT	I, Q, M, D, L or constant	DP master system ID of those master systems for which the topology will be determined
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: Topology determination is not yet complete.
DPR	Output	BYTE	I, Q, M, D, L	PROFIBUS address of the diagnostics repeater reporting the error.
DPRI	Output	BYTE	I, Q, M, D, L	Measuring segment diagnostics repeater reporting the error: • Bit 0 = 1: Temporary faults on segment DP2 • Bit 1 = 1: Permanent faults on segment DP2 • Bit 4 = 1: Temporary faults on segment DP3 • Bit 5 = 1: Permanent faults on segment DP3

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

When looking at the "real" error information (error codes W#16#8xyz) in the following table, we distinguish between two cases:

- Temporary errors (error codes W#16#80A2 to 80A4, 80C3, 80C5):
  It is possible to eliminate this type of error without user action; in other words, it is advisable to call "DP_TOPOL" again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#8082, 80B0, 80B2):
  This type of error does not correct itself. A new call of "DP_TOPOL" is only advisable after you have eliminated the error. Example of a permanent error: The DP master/CPU does not support this service. (W#16#80B0).

Error code (W#16#...)	Explanation
0000	Job completed without error.
7000	First call with REQ=0. Topology determination is not triggered. BUSY has the value "0".

Error code (W#16#...)	Explanation
7001	First call with REQ=1. The request to execute topology determination was sent. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Topology determination is not yet complete. BUSY has the value "1".
7010	You have attempted to cancel topology identification. But there is no running job with the specified DP_ID. BUSY has the value "0".
7011	First call with R=1. The cancellation of the topology determination was triggered. BUSY has the value "1".
7012	Intermediate call: The cancellation of the topology determination is not yet complete. BUSY has the value "1".
7013	Final call: The topology determination was cancelled. BUSY has the value "0".
8082	No DP master system is configured with the specified DP_ID.
80A2	Error during topology determination; for more detailed information, refer to output parameters DPR and DPRI.
80A3	Error during topology determination: Monitoring time has elapsed (timeout).
80A4	Communication on the communication bus disrupted
80B0	The DP master/CPU does not support this service.
80B2	Error during topology determination: No diagnostics repeater was found at the selected DP master system.
80C3	Resources required are currently in use. Possible cause: You have initiated a second topology determination (only one topology determination is permitted at one time).
80C5	The DP master system is currently not available.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

ASi

ASi_3422: Control AS-i master behavior

Description

The instruction "ASi_3422" is necessary to use the operating mode "Extended Operation" with CP 342-2. The extended operation allows complete control of the master behavior using the user program. The access to the inputs and outputs takes place as before in the standard operation of the CP 342-2.

Command calls to the CP342-2 occur from the user program via the instruction. You will specify a command call in a send buffer and start the job.

The instruction "ASi_3422" transfers the command call to the CP342-2. The job status will be transferred with the conclusion of the job and possible answer data will be provided in a receive buffer.

Call

The instruction must be cyclically called for each CP342-2. Only one job can be processed at a time per CP342-2. A current job cannot be interrupted and the instruction will not monitor the time. The instruction "ASi_3422" is executed uninterruptedly. Calls can thus not be programmed in program priority classes which interrupt each other (for example, with calls in OB 1 and in OB 35).

Configure the command processing in the user program as follows:

1. When restarting the S7 user program, call "ASi_3422" once with the parameter value STARTUP = TRUE.
2. Specify the command call in a send buffer of the user program. Transfer this send buffer with the call parameter SEND.
   You can find more information about the command interfaces and the ASi slave command in the hardware documentation of the communication processor.
3. You may need an answer buffer depending on the command type. Transfer this answer buffer with the call parameter RECV. The answer buffer is not necessary for status information with this interface.
4. Activate the job using parameter ACT=1.
5. Then query the parameters DONE, ERROR and STATUS.

Parameters

The following table shows the parameters of the "ASi_3422" instruction:

Parameter	Declaration	Data type	Memory area	Description
ACT	Input	BOOL	I, Q, M, D, L	The command processing by the function occurs level-triggered, meaning that a command processing will be started if a call is not already being processed as long as ACT = 1 is active.
STARTUP	Input	BOOL	I, Q, M, D, L	The instruction will indicate a CPU startup with STARTUP = 1. After the first run, the function STARTUP must be reset by the user.
LADDR	Input	WORD	I, Q, M, D, L or constant	Module start address The module start address is to be determined using the information regarding the slot-oriented issuing of addresses to signal modules.
SEND	Input	ANY	I, Q, M, D, L	Send buffer The parameter refers to a memory area in which the command is to be specified by the user, for example: P#DB20.DBX 20.0 Byte 10.
RECV	Input	ANY	I, Q, M, D, L	Receive buffer This buffer is only relevant for commands which deliver answer data. The parameter refers to a memory area in which a command answer is stored. The length of the data area parameterized here is irrelevant, for example: P#DB30.DBX 20.0 Byte 1
DONE	Output	BOOL	Q, M, D, L	DONE = 1 signals ‘job complete without error'.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	Q, M, D, L	ERROR = 1 signals ‘job complete with errors'.
STATUS	InOut	DWORD	M, D	1. Word: Job status / error code (see following table); an error code will be generated for 'job complete with error' for more specific error description. 2. Word: Is required by the instruction for internal purposes and may not be changed. For instruction calls to different logical addresses (LADDR) different double words must be assigned for the parameter STATUS.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter LADDR

The start address of the address area is determined by the slot of CP 342-2.

The CP 342-2 occupies 16 input bytes and 16 output bytes in I/O address area of the S7 automation device (analogous area of the AS). 31 x 4 bits of the 16 byte address area of the CP 342-2 are occupied by the AS-i slave data. The remaining 4 bits are reserved for later applications.

Parameters DONE, ERROR and STATUS

If an error occurs during the processing of the function, a "0" will be in the BR bit in addition to the information in ERROR and STATUS mentioned above. The querying of the BR bit is different for LAD and STL user programs:

- LAD: Query using output parameter ENO
- STL: Direct query of the BR bit

The following table contains the possible display in the first word of STATUS dependent upon DONE and ERROR.

DONE	ERROR	STATUS	Meaning
0	0	8181	Job running.
1	0	0000	Job complete with errors.
0	1	8090	Address at parameter LADDR invalid.
0	1	80A0	Negative acknowledgement when reading from the module.
0	1	80A1	Negative acknowledgement when writing to the module.
0	1	80B0	The module does not recognize the data record.
0	1	80B1	Data record length entered is incorrect.
0	1	80C0	Cannot read data record.
0	1	80C1	The requested data record is still being processed.
0	1	80C2	Jobs are backed up.
0	1	80C3	Resources (memory) occupied.
0	1	80C4	Communication error

DONE	ERROR	STATUS	Meaning
0	1	8182	Meaning: Identification after restart with STARTUP=TRUE (no errors).
0	1	8184	Data type of formal operand RECV illegal.
0	1	8381	Slave address incorrect
0	1	8382	Slave is not activated (not in LAS).
0	1	8383	Error in AS interface
0	1	8384	Command (in status of CP) illegal.
0	1	8385	Slave 0 exists.
0	1	83A1	Slave with address to be changed not found at AS interface.
0	1	83A2	Slave 0 present.
0	1	83A3	Slave with new address already present at AS interface.
0	1	83A4	Slave address cannot be deleted.
0	1	83A5	Slave address cannot be set.
0	1	83A6	Slave address cannot be permanently set.
0	1	83F8	Job number unknown.
0	1	83F9	EEPROM error
0	1	8F22	Range length error when reading a parameter.
		8F23	Range length error when writing a parameter. This error code indicates that a parameter is located either entirely or partly outside the range of an address, or that the length of a bit range is not a multiple of 8 with an ANY parameter.
0	1	8F24	Range error when reading a parameter.
		8F25	Range error when writing a parameter. This error code indicates that a parameter is located in a range that is illegal for a system function.
0	1	8F28	Alignment error when reading a parameter.
		8F29	Alignment error when writing a parameter. This error code indicates that the reference to parameter is an operand with bit address that is not equal to "0".
0	1	8F30	Parameter is in the read-only global DB.
		8F31	Parameter is in the read-only instance DB. This error code indicates that a parameter is located in a read-only data block.
0	1	8F32	Parameter contains a DB number which is too large
0	1	8F3A	The parameter contains the number of a DB that is not loaded.
0	1	8F42	An access error occurred while the system was attempting to read a parameter from the peripheral input area.
		8F43	An access error occurred while the system was attempting to write a parameter to the peripheral output area.
0	1	8F44	This error code indicates that the read access to the parameter has been denied.
		8F45	This error code indicates that the write access to the parameter has been denied.
0	1	8F7F	Internal error

ASI_CTRL: Control AS-i master behavior

Description of ASI_CTRL

Description

Using the instruction "ASI_CTRL", you can control the behavior of the AS-i master from the user program of the PLC. The instruction processes the command protocol automatically. It also enables parameter assignment on SIMATIC AS-i masters and reading out information data. The functions and operation of the command interface are detailed in the manual for the AS-i master.

Both centrally inserted AS-i masters and distributed AS-i masters via PROFIBUS DP are supported. Combinations with PROFINET IO (e.g. IE/PB Link PN IO) are also possible.

The schematic diagram below shows the functions of the "ASI_CTRL" instruction:

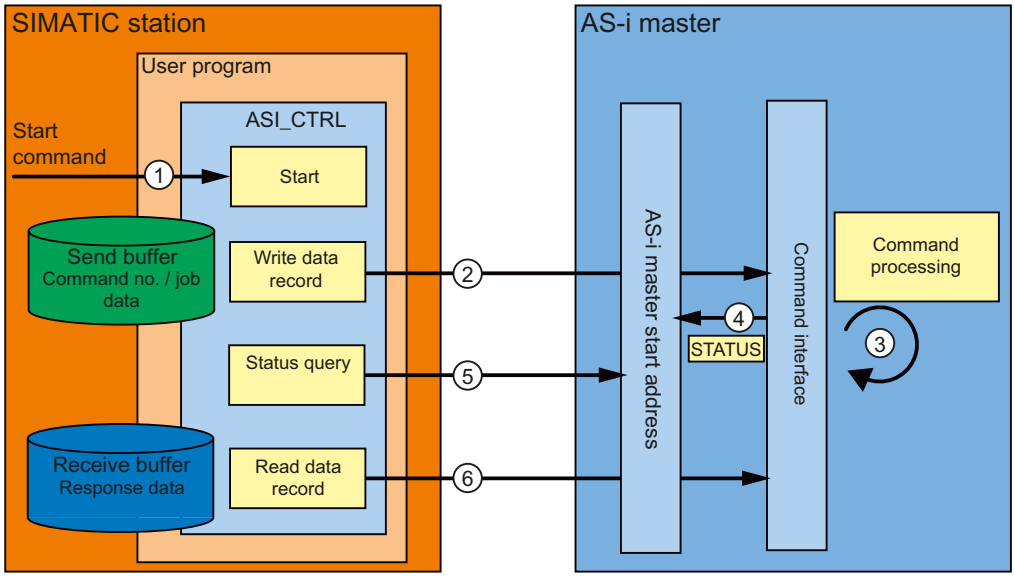

① Start of processing at the REQ parameter.

② The program sends the required command to the AS-i master via the instruction "RDREC".

③ The AS-i master processes the command.

④ The current status of the AS-i master is stored in the input area of the binary data (logical start address).

⑤ The instruction "ASI_CTRL" cyclically queries and evaluates the 4 status bits.

⑥ Once command processing is complete, the command job is completed with "RDREC". Depending on the command, the data field of "RDREC" may contain the response data for the command or additional status information.

Dependencies between instruction versions

Version V1.3 of instruction "ASI_CTRL" requires version V1.1 of instruction "WRREC".

Version V1.2 of instruction "ASI_CTRL" requires version V1.1 of instruction "WRREC".

Differences in command calls between IE/ AS-i LINK and CPs / DP/AS-i LINK

There are significant differences in how a controller exchanges commands with an AS-i master.

- **IE/AS-i Link** (PROFINET) used the data record interface. The different commands are called with either "Write data record" ("WRREC" instruction) or "Read data record" ("RDREC" instruction) by various data record numbers.
- **S7-300 CPs** and **DP/AS-i Links** (PROFIBUS) use the command interface. All commands are called by data record number 2 with "Write data record" ("WRREC" instruction) plus "Read data record" ("RDREC" instruction). The type of command is defined by the data content in the write job.

Changes compared to the instruction "ASi_3422".

The instruction "ASI_CTRL" is a revised version of the instruction "ASi_3422 (Page 5292)" and provides improved functionality and compatibility. The specific changes are as follows:

- For writing and reading diagnostic data records, the instructions "WR_REC (Page 5260)" and "RD_REC (Page 5256)" have been replaced by the instructions "RDREC (Page 5224)" and "WRREC (Page 5226)". Their function is identical; however, they support data transfer via PROFINET IO.
- The block type of the instruction has been changed from a function (FC) to a function block (FB). "ASI_CTRL" includes an instance block and supports multiple instances.
- The designation of the formal parameters of "ASI_CTRL" complies with the SIMATIC system blocks. There is no STARTUP input parameter. The definition of the STATUS parameter is based on the instructions "RDREC (Page 5224)" and "WRREC (Page 5226)". The status identifiers for the parameter DONE and the new parameter BUSY have also been adjusted.

Operation of the "ASI_CTRL" instruction

The instruction "ASI_CTRL" is an asynchronous function block; in other words, its processing extends over multiple calls.

- A job is started with REQ = TRUE.
- The job status is displayed via the BUSY output parameters and the two central bytes of the output parameter STATUS.
- The BUSY parameter is set during job processing. Upon the first call, STATUS is assigned the value 00700100_H. Upon all subsequent calls for this job, it is assigned the value 00700200_H. When the job is complete, the result is output at the parameters DONE or ERROR.
  - If no errors have occurred, DONE is set. For jobs with response data from the AS-i master, this data is provided in the specified receive buffer. In such cases, the STATUS parameter also displays the volume of data supplied in bytes. For jobs without response data, the value 00000000_H is entered in STATUS.
  - If an error occurs during job processing, ERROR is set. The content of the receive buffer is invalid in such a case. An error code is entered in the STATUS parameter for a more detailed description of the error which has occurred.

Number of command calls

If you use the instruction "ASI_CTRL" to send commands, you may not simultaneously send other commands to the same AS-i master with "RDREC (Page 5224)" or "WRREC (Page 5226)". This also applies to multiple calls of the instruction for the same AS-i master.

The instruction "ASI_CTRL" cannot be run with interruptions. Calls therefore cannot be programmed in program priority classes which interrupt each other (for example with calls in OB 1 and in OB 35).

Parameters

The following table shows the parameters of the "ASI_CTRL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = TRUE starts a new job unless a job is already in progress. No edge evaluation takes place.
LADDR	Input	WORD	I, Q, M, D, L or constant	Start address of the AS-i master in the S7 address space (logical base address). The start address is defined in the hardware configuration when the master is configured.
SD	Input	ANY	I, Q, M, D, L	Send buffer The parameter refers to a memory area in which the command is to be specified (see "ASi commands (Page 5300)"). Example: P#DB101.DBX 0.0 BYTE 223

Parameter	Declaration	Data type	Memory area	Description
RD	Input	ANY	I, Q, M, D, L	Receive buffer This buffer is only relevant for commands which deliver answer data. The parameter refers to a memory area in which a command response is stored (see "ASi commands (Page 5300)"). Example: P#DB102.DBX 224.0 BYTE 221
DONE	Output	BOOL	Q, M, D, L	DONE = TRUE: Job completed without errors.
BUSY	Output	BOOL	Q, M, D, L	BUSY = TRUE: Job in process.
ERROR	Output	BOOL	Q, M, D, L	ERROR = TRUE: Job aborted with error.
STATUS	Output	DWORD	M, D	Job status / error code See the "STATUS parameter" description.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Note

LADDR, SD and RD parameters

The parameters LADDR, SD and RD must not be changed in any block cycle when a job is in progress: they must remain constant.

Parameter STATUS

The following table shows the possible STATUS displays dependent upon DONE and ERROR.

DONE	BUSY	ERROR	STATUS	Meaning
0	0	0	00700000_H	First call with REQ = FALSE; no active job.
0	1	0	00700100_H	First call with REQ = TRUE; job has started.
0	1	0	00700200_H	Subsequent call (REQ irrelevant); job is still in progress.
1	0	0	00000000_H	Job completed without errors. No response data.
1	0	0	$0000xx00_H$	Job completed without errors. Number of xx bytes of response data.
0	0	1	$C0818400_H$	Data type of formal operand RD invalid.
0	0	1	$C0818500_H$	Error in communication with AS-i master (incorrect address configured at the LADDR parameter).
0	0	1	$C0838100_H$	Incorrect AS-i slave address.
0	0	1	$C0838200_H$	AS-i slave is not enabled (not in LAS).
0	0	1	$C0838300_H$	Error at the AS interface (the SD parameter may have been set too small).
0	0	1	$C0838400_H$	The command is not valid with the current AS-i master status.
0	0	1	$C0838500_H$	There is an AS-i slave with the address "0".

DONE	BUSY	ERROR	STATUS	Meaning
0	0	1	C0838600 H	The AS-i slave has invalid configuration data (I/O or ID codes).
0	0	1	C083A100 H	The AS-i slave addressed has not been found at the AS interface.
0	0	1	C083A200 H	There is an AS-i slave with the address "0".
0	0	1	C083A300 H	There is already an AS-i slave with the new address at the AS interface.
0	0	1	C083A400 H	The AS-i slave address cannot be deleted.
0	0	1	C083A500 H	The AS-i slave address cannot be set.
0	0	1	C083A600 H	The AS-i slave address cannot be permanently saved.
0	0	1	C083A700 H	Error during reading of the extended ID1 code.
0	0	1	C083A800 H	The target address is not plausible (e.g. a B slave address has been used for a standard slave).
0	0	1	C083B100 H	A length error has occurred during string transfer.
0	0	1	C083B200 H	A protocol error has occurred during string transfer.
0	0	1	C083F800 H	Unknown job number or job parameter.
0	0	1	C083F900 H	The AS-i master has detected an EEPROM error.

ASi commands

Description

The command interface allows the controller and AS-i master to exchange parameter assignment and information data.

These commands

- provide the complete functionality of the M4 master profile of the AS-i master specifications.
- enable the AS-i master to be completely configured from the controller.

Note

AS-i commands supported

Please see the manual of the relevant AS-i master for the AS-i commands supported and a detailed description.

General structure of the send buffer

The general structure of the send buffer for commands and job data is set out in the table below. The area for the command number must always be filled. The number of bytes for the job data can be found in the description of the command (see AS-i master documentation). In this case, "q" refers to the start address of the send buffer.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Meaning							
q + 0	Command number							
q + 1	Job data							
q + 2	Job data							
q + ...	Job data							

General structure of the receive buffer

The general structure of the receive buffer for the command response data is set out in the table below. The number of bytes for the response data depends on the command. Some commands do not return response data, and therefore only require the definition of a virtual receive buffer which is not filled with data. In this case, "n" refers to the start address of the receive buffer.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Meaning							
n + 0	Response data							
n + 1	Response data							
n + 2	Response data							
n + ...	Response data							

NOTICE

Memory areas can be overwritten

If the receive buffer of the "ASI_CTRL" instruction is too short, neighboring memory areas may be overwritten. The length specified in the ANY pointer of the RD parameter when the instruction "ASI_CTRL" is called is irrelevant. The required length for the receive buffer can be found in the description of the command.

The following applies for command numbers 39_H, 41_H, 42_H, 43_H and 44_H:

The receive buffer must be 221 bytes long (bytes 0 to 220), even if the command returns less data. Depending on the command, the highest bytes in the receive buffer may be overwritten with zero values by the AS-i master.

AS-i commands

A selection of possible AS-i commands is set out in the table below.

Name	Parameter	Return	Coding
Set_permanent_parameter (Set_Permanent_Parameter)	Slave address, parameter		00_H
Get_permanent_parameter (Get_Permanent_Parameter)	Slave address	Parameter	01_H
Write_parameter (Write_Parameter)	Slave address, parameter	Parameter echo	02_H
Read_parameter (Read_Parameter)	Slave address	Parameter value	03_H
Store_actual_parameters (Store_Actual_Parameters)			04_H
Set_configuration_data	Slave address, configuration		25_H
Get_configuration_data	Slave address	set configuration data	26_H
Store_actual_configuration (Store_Actual_Configuration)			07_H
Get_actual_configuration	Slave address	Actual configuration data	28_H
Configure_LPS	LPS		29_H
Set_offline_mode	Mode		$0A_H$
Select_auto-program	Mode		$0B_H$
Set_mode	Mode		$0C_H$
Change_AS–iSlave_address (Change_AS–iSlave_Address)	Address1, Address2		$0D_H$
Get_AS-iSlave_status	Slave address	Error record of the AS-i slave	$0F_H$
Read_lists_and_flags		LDS, LAS, LPS, flags	30_H
Get_overall_configuration		Actual configuration data, current parameters, LAS, flags	39_H
Set_overall_configuration	Overall configuration		$3A_H$
Write_parameter_list	Parameter list		$3C_H$
Read_parameter_echo_list		Parameter echo list	33_H
Write_CTT2_request	Slave address CTT2 string	CTT2 string	44_H
Read_version_identifier		Version string	14_H
Read_AS-i_slave	Slave address	ID code	17_H
Read_AS-i_slave_extended_ID1	Slave address	Extended ID1 code	37_H
Write_AS-iSlave_extended_ID1	Extended ID1 code		$3F_H$
Read_AS-iSlave_extended_ID2	Slave address	Extended ID2 code	38_H
Read_AS-iSlave_IO	Slave address	I/O configuration	18_H
Read_I/O_error_list		LPF	$3E_H$
Write_AS–i-slave_parameter_string	Slave address, parameter string		40_H
Read_AS–iSlave_parameter_string	Slave address	Parameter string	41_H

Name	Parameter	Return	Coding
Read_AS-iSlave_ID_string	Slave address	ID string	42_H
Read_AS-iSlave_diagnostic_string	Slave address	Diagnostic string	43_H
Read_AS-i_line_error_counter			$4A_H$
Read_and_clear_AS-i_line_error_counter			$4B_H$
Read_AS-iSlave_error_counter	Slave address		$4C_H$
Read_and_clear_AS-iSlave_error_counter	Slave address		$4D_H$
Additional command for DP/ AS-i F-Link:			
AS–i_status/diag_of_F_slaves		Status/diagnostics of all AS-i safe slaves	51_H

Note

Re-initializing the AS-i master command interface

Another command which is not mentioned in the table is command 77_H. This call serves to re-initialize the command interface of the AS-i master. Any command which the AS-i master specified is currently processing will be terminated.

As of version V2.1.20 of DP/AS-i LINK Advanced, command $0E_H$ is also available. This call enables you to release and block the ground fault monitoring function for a line.

4.2.3.5 PROFIenergy (S7-300, S7-400)

Description of PROFIenergy (S7-300, S7-400)

PROFIenergy

PROFIenergy is a manufacturer- and device-neutral profile for energy management with PROFINET. PROFIenergy enables central, coordinated device shutdown to reduce electricity consumption during breaks in production and unplanned interruptions.

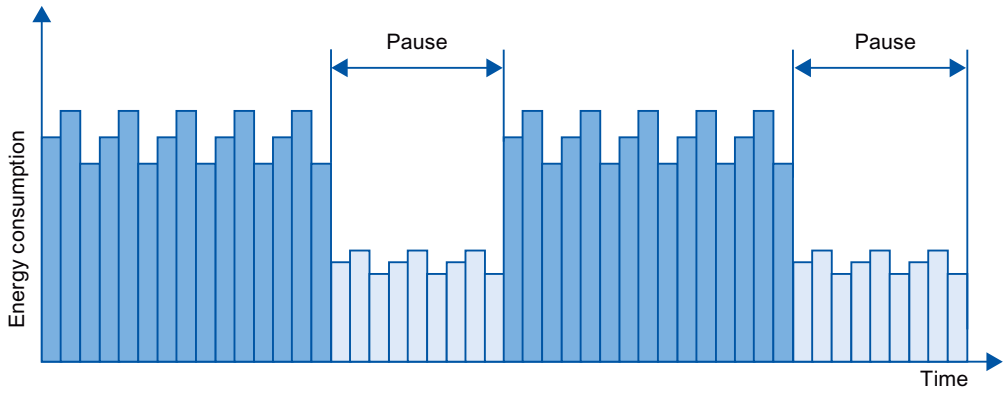

The PROFINET devices/power modules are switched off via special commands in the user program of the PROFINET IO controller. You do not need additional hardware. The PROFIenergy commands are interpreted directly by the PROFINET devices.

PROFIenergy controller (PE controller)

The PE controller is a PLC that activates or deactivates the idle state of slave devices. Specific production components, or complete production lines are deactivated and reactivated via the user program. Commands (e.g. "Start_Pause" and "End_Pause") are sent to the lower-level device using corresponding instructions (function blocks). The commands are sent via the PROFINET communication protocol.

PROFIenergy entity (PE entity)

The PE entity receives the PROFIenergy commands of the PE controller and converts these accordingly (e.g. by returning a measured value, or by activating the energy-saving mode). Integration of the PE entity in a device that supports PROFIenergy is based on specific device and manufacturer data.

The PE entity can be implemented, for example:

- Within the proxy of a submodule: The PE commands are valid for the addressed submodule and any existing lower-level submodules.
- Within the proxy of a module: The PE commands are valid for different submodules within the module.

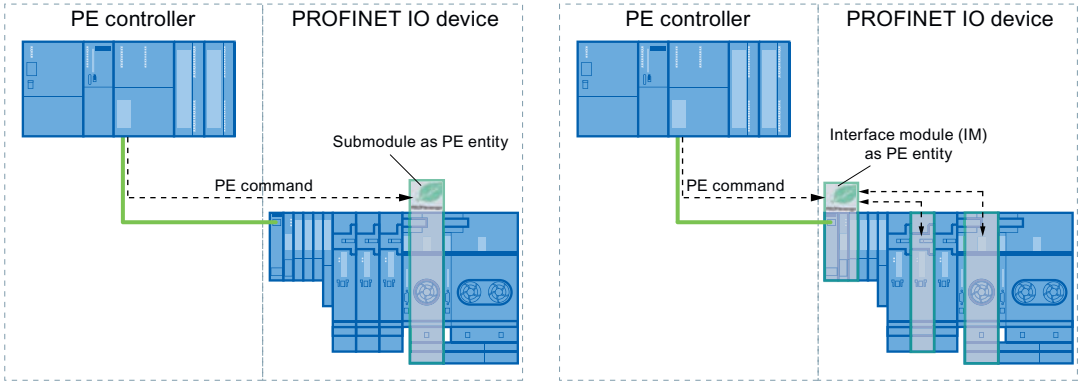

- For networked submodules without a proxy function: The PE commands in this case are only valid for the respective submodule.

PROFIenergy instructions

- Instructions for IO controller
  - The "PE_START_END (Page 5306)" instruction represents the simplest way to activate or deactivate the idle state of the PROFINET devices (PROFIenergy commands "Start_Pause" and "End_Pause"). This is done using a positive and negative signal edge in the instruction.
  - The "PE_CMD (Page 5310)" instruction allows you to transmit all PROFIenergy commands, including "Start_Pause" and "End_Pause". With other commands, you can query the current status of the PROFINET devices or the behavior during breaks, for example.
  - The instruction "PE_DS3_Write_ET200S (Page 5314)" is used to define the switching characteristic of up to 8 slots of ET 200S. The instruction is not a PROFIenergy instruction; however, it supplements the PROFIenergy functions for an ET 200S.
- Instruction for iDevices
  The "PE_I_DEV (Page 5338)" instruction allows you to implement PROFIenergy on iDevices as well. The instruction receives PROFIenergy commands on the iDevice and forwards these to the user program for processing. After processing the command, the user program calls the "PE_I_DEV (Page 5338)" instruction again in order to send the acknowledgement to the IO controller. For these replies, each command offers you a corresponding auxiliary block that supplies the response data to the instruction "PE_I_DEV (Page 5338)".

PROFIenergy commands (PE commands)

The PE controller transmits PE commands to the PE entity. The PE command can be either a control command to switch a PE entity to the energy-saving mode, or a command to read a status or measured value:

- PI commands for control
  PROFIenergy supports two control commands that can be executed using either the "PE_Start_End (Page 5306)" instruction or the "PE_CMD (Page 5310)" instruction:
  - Start_Pause: Starting a suitable energy-saving mode (PE Energy-saving mode)
  - End_Pause: Exiting the energy-saving mode (switch to PE_ready_to_operate mode)
- PI commands for reading a status or measured value
  Certain condition information can be read by the control system using the following status commands with the instruction "PE_CMD (Page 5310)":
  - PE_Identify: Reading the list of PE commands supported by the PE entity.
  - PEM_Status: Reading the currently active mode of a PE entity (e.g. PE_ready_to_operate).
  - Query_Modes: Output an overview of all supported energy-saving modes, including the time and energy information
  - Query_Measurement: Output the measured values of a PE entity

Example applications

Examples for use of PROFIenergy instructions can be found in the Industry Online Support in the entry "PROFIenergy - Saving Energy with SIMATIC S7 (http://support.automation.siemens.com/WW/view/en/41986454)".

See also

Service and Support (http://support.automation.siemens.com/)

IO controller (S7-300, S7-400)

PE_START_END: Start and exit energy-saving mode (S7-300, S7-400)

Description

The instruction "PE_START_END" is used to start and exit the energy-saving mode of a specified PE entity (e.g. ET 200S).

The instruction "PE_START_END" is used in the PE controller, preferably when only field devices from which energy data does not need to be read out are connected to the corresponding PE devices. Alternatively, the instruction "PE_CMD (Page 5310)" can be used to read energy data.

The energy-saving modes are configured in the user program of the PE controller. On completion of the "PE_START_END" instruction, the PE entity reports its current energy-saving mode and outputs this data at parameter PE_MODE_ID.

Write and read jobs of the "PE_START_END" instruction.

The instruction "PE_START_END" uses "WRREC (Page 5226)" internally to transmit a PROFIenergy command as write job to the PE entity. "PE_START_END" then waits for the acknowledgment frame from the PE entity. The acknowledgement data record is read with the instruction "RDREC (Page 5224)" every 100 milliseconds. Until acknowledgment is received from the PE entity, the function repeats the read job for 10 seconds at intervals of 100 milliseconds. The response data returned by the PE entity is also read with the instruction "RDREC (Page 5224)".

Below is a flowchart of the write and read jobs:

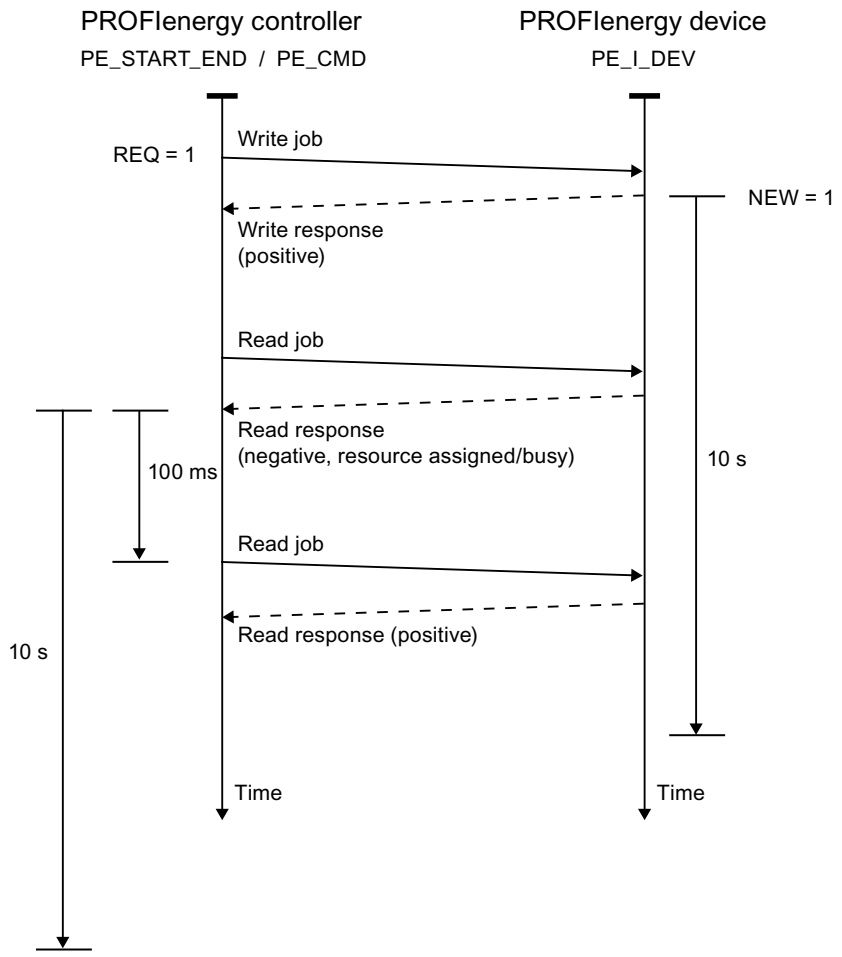

Parameters

The following table shows the parameters of the "PE_START_END" instruction:

Parameter	Declaration	Data type	Memory area	Description
START	Input	BOOL	I, Q, M, D, L	Transmitting the PE command "Start_Pause" to the PE entity with the address set at parameter ID.
END	Input	BOOL	I, Q, M, D, L	Transmitting the PE command "End_Pause" to the PE entity with the address set at parameter ID.
ID	Input	DWORD	I, Q, M, D, L or constant	Address of the PE entity (e.g. ET 200S). The address may be taken from the hardware configuration.
PAUSE_TIME	Input	TIME	I, Q, M, D, L or constant	Planned pause duration. • Range: T#1MS to T#24D20H31M23S647MS • Start value: T#0MS
VALID	Output	BOOL	I, Q, M, D, L	PE command successfully sent.

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	PE command still being processed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred during processing. The error message is output at the STATUS parameter.
STATUS	Output	DWORD	I, Q, M, D, L	Block status / error number (see "STATUS parameter")
PE_MODE_ID	Output	BYTE	I, Q, M, D, L	Identification number of energy-saving mode (energy-saving level for the duration of the pause).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

PAUSE_TIME parameter

The parameter PAUSE_TIME is used to set the default duration of the energy-saving period at the PE entity. The PE entity checks whether the pause period is of sufficient length and whether it can be implemented. The minimum pause duration (Time_min_Pause) must be greater than the sum of the times the device needs to switch to energy-saving mode (Time_to_Pause) and to switch back to operating mode (Time_to_Operate).

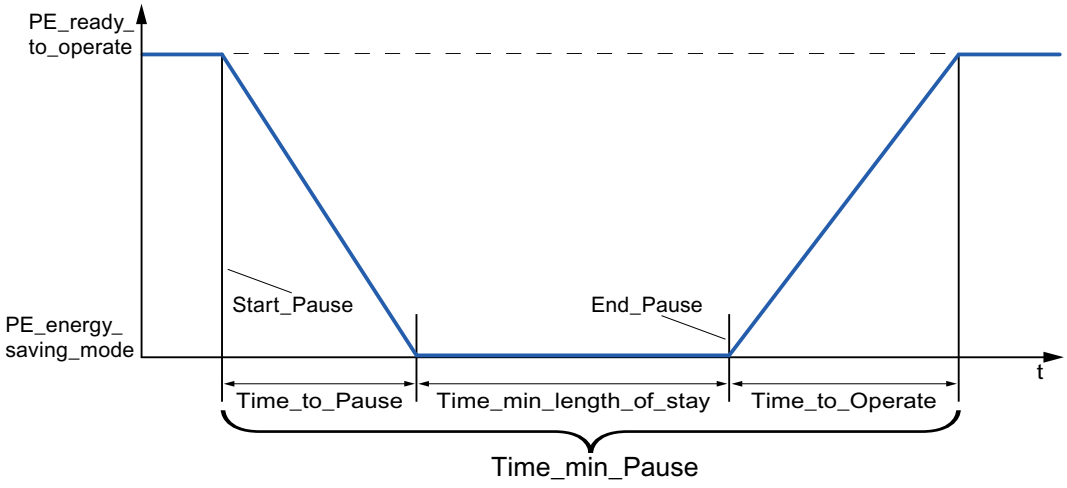

ET 200S checks whether the scheduled pause duration is greater than or equal to the minimum pause duration (PM-E_Pause_Min) saved on the ET 200S. This is fixed at 10 seconds. If a shorter pause is used, the power modules (PM-E) of the ET 200S will remain on.

The module does not switch back on automatically at the end of the pause; it remains in OFF mode until the "END" command is sent. This prevents uncoordinated switch-on, which can cause undesired peak loads.

Parameter STATUS

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Array element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error during reading of data record • B#16#**DF**: Error during writing of data record • B#16#**C0**: Error message from the instruction or from the communication instructions "RDREC (Page 5224)" and "WRREC (Page 5226)" used internally.
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific • **FE**: DP/PNIO profiles - PROFIenergy-specific error
STATUS[3]	Error_Code_1	Error ID • When Error_Decode = **80**: – **80**: Simultaneous rising edge at the input parameters START and END. – **81**: Length conflict at the CMD_PARAM and CMD_PARAM_LEN parameters. – **82-8F**: Other error messages (reserved) • When Error_Decode = **FE**: – **01**: "Service Request ID" invalid – **02**: Incorrect "Request_Reference" – **03**: "Modifier" invalid – **04**: "Data Structure Identifier RQ" invalid – **05**: "Data Structure Identifier RS" invalid – **06**: "PE energy-saving modes" are not supported – **07**: "Response" is too long (maximum transmissible length exceeded) – **08**: "Count" invalid – **50**: No suitable "energy mode" is available. – **51**: Time value specified is not supported. – **52**: "PE_Mode_ID" invalid
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Note

Error messages of the instructions RDREC and WRREC

The instruction "PE_START_END" uses the instructions "WRREC (Page 5226)" and "RDREC (Page 5224)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 5235) parameter for an explanation of the error codes of the "WRREC (Page 5226)" and "RDREC (Page 5224)" instructions.

PE_CMD: Start and exit energy-saving mode / Read out status information (S7-300, S7-400)

Description

The instruction "PE_CMD" is used in the PE controller to start or terminate a pause in the energy-saving mode in the PE entity. "PE_CMD" can be used to read additional information and energy measurement values from a PE entity.

The instruction is best used with PE controllers to which assigned PE devices are connected from which energy measurements are to be read out. If this is not the case, the instruction "PE_START_END (Page 5306)" can also be used to start and end the pauses.

Transfer of the PROFIenergy commands (PE commands)

The instruction "PE_CMD" transmits a PROFIenergy command to a PE entity.

The instruction can also be used if additional commands are to be added to the PROFIenergy profile in future. The commands which can be used with the current PROFIenergy profile are listed in the description of the CMD and CMD_MODIFIER parameters (see the "CMD and CMD_MODIFIER parameters" table).

- The individual PE commands which are transferred to the PE entity using the instruction are assigned defined "Service_Request_IDs". The Service_Request_IDs 01 to 05 and 16 are assigned at the CMD parameter.
- The two PE commands 04 (Query_Modes) and 16 (Query_Measurement) are defined in more detail with the CMD_MODIFIER parameter.
- Additional values are transferred at the CMD_PARA parameter for individual PE commands (see the description of the individual PE commands). The parameter CMD_PARA_LEN defines the length of the data at the parameter CMD_PARA.

Commands are transmitted without a plausibility check. The response data of the PE entity is saved to the RESPONSE_DATA data area that is addressed by ANY pointer (for information about the response frame contents, refer to the description of the respective PE command).

Write and read jobs of the "PE_CMD" instruction.

The instruction "PE_CMD" uses "WRREC (Page 5226)" internally to transmit a PROFIenergy command as a write job to the PE entity. "PE_CMD" then waits for the acknowledgment from the PE entity. The acknowledgement data record is read with the instruction "RDREC (Page 5224)" every 100 milliseconds. Until acknowledgment is received from the PE entity, the function repeats the read job for 10 seconds at intervals of 100 milliseconds. The response data returned by the PE entity is also read with the instruction "RDREC (Page 5224)".

Below is a flowchart of the write and read jobs:

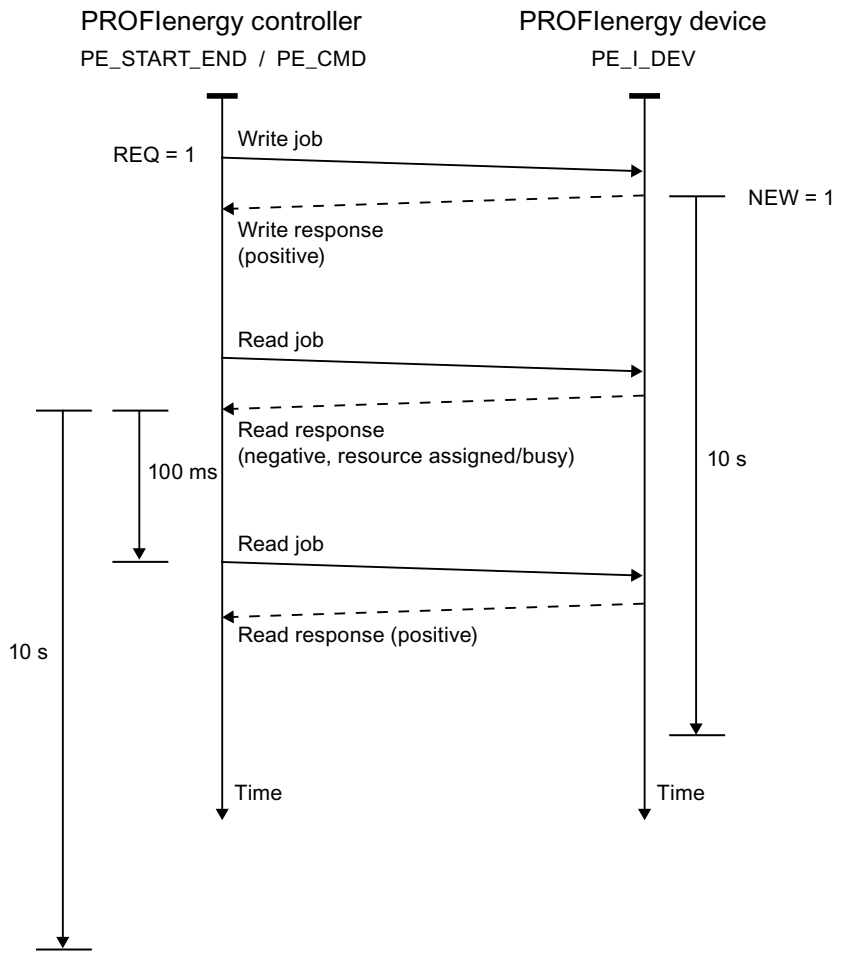

Parameters

The following table shows the parameters of the "PE_CMD" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Starts transferring the PE command upon a rising edge.
ID	Input	DWORD	I, Q, M, D, L or constant	Address of the PE entity For PROFINET IO devices, you can transfer the address from the hardware configuration.
CMD	Input	BYTE	I, Q, M, D, L, P or constant	Service-Request-ID of the PROFIenergy command in accordance with the PROFIenergy profile (see "CMD and CMD_MODIFIER parameters"). Further service request IDs are possible following extensions of the PROFIenergy profile.

Parameter	Declaration	Data type	Memory area	Description
CMD_MODIFIER	Input	BYTE	I, Q, M, D, L, P or constant	PROFIenergy sub-command (only when CMD=3 or CMD=16, see "CMD and CMD_MODIFIER parameters") Further sub-commands are possible following extensions of the PROFIenergy profile.
CMD_PARA	Input	ANY	I, Q, M, D, L	Parameters for the PE commands: • Get mode: PE_mode_ID • Get measurement values: List of Measurement_Ids The complete Service Data Request is entered.
CMD_PARA_LEN	Input	INT	I, Q, M, D, L, P or constant	The actual length of the command parameters (<= length defined in CMD_PARA; is verified by the instruction).
RESPONSE_DATA	InOut	ANY	I, Q, M, D, L	PROFIenergy information May be complete response frame including block header, depending on the command. Note: If the buffer is too small, only the number of bytes which is specified in the ANY pointer will be entered.
VALID	Output	BOOL	I, Q, M, D, L	Command successfully sent.
BUSY	Output	BOOL	I, Q, M, D, L	Command still being processed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred during processing.
STATUS	Output	DWORD	I, Q, M, D, L	Block status / error number (see "STATUS parameter"):

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters CMD and CMD_MODIFIER

CMD	CMD_ MODIFIER	PROFIenergy command	Description
01	0	Start_Pause (Page 5328)	Start energy-saving mode or switch to a different energy-saving mode.
02	0	End_Pause (Page 5329)	End energy-saving mode.
03	1	Query_Modes - List of energy-saving modes (Page 5329)	Output the energy-saving modes supported.
	2	Query_Modes - Get mode (Page 5330)	Output attributes of the energy-saving mode currently enabled.
04	0	PEM_Status (Page 5332)	Query status of energy-saving mode.
05	0	PE_Identify (Page 5334)	Read out the number and description of PE commands supported.

CMD	CMD_ MODIFIER	PROFIenergy command	Description
16	1	Query_Measurement - Get_Measurement_List (Page 5334)	List of measured values supported by the PE entity.
	2	Query_Measurement - Get_Measurement_Values (Page 5337)	Output the measured values of the PE entity

Parameter STATUS

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Array element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error during reading of data record • B#16#**DF**: Error during writing of data record • B#16#**C0**: Error message from the internal communication instructions "RDREC (Page 5224)" and "WRREC (Page 5226)".
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific • **FE**: DP/PNIO profiles - PROFIenergy-specific error
STATUS[3]	Error_Code_1	Error ID • When Error_Decode = **80**: – **81**: Length conflict at the CMD_PARA and CMD_PARA_LEN parameters, or maximum data record length (4095 bytes) exceeded. – **82-8F**: Other error messages (reserved) • When Error_Decode = **FE**: – **01**: "Service Request ID" invalid – **02**: Incorrect "Request_Reference" – **03**: "Modifier" invalid – **04**: "Data Structure Identifier RQ" invalid – **05**: "Data Structure Identifier RS" invalid – **06**: "PE energy-saving modes" are not supported – **07**: "Response" is too long (maximum transmissible length exceeded) – **08**: "Count" invalid – **50**: No suitable energy-saving mode (energy mode) is available. – **51**: Time value specified is not supported. – **52**: "PE_Mode_ID" invalid
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Note

Error messages of the instructions RDREC and WRREC

The "PE_CMD" instructions use the "WRREC (Page 5226)" and "RDREC (Page 5224)" instructions for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 5235) parameter for an explanation of the error codes of the "WRREC (Page 5226)" and "RDREC (Page 5224)" instructions.

PE_DS3_Write_ET200S: Set power module switching behavior (S7-300, S7-400)

Description

The instruction "PE_DS3_Write_ET200S" sends basic settings for power module switching behavior to the ET 200S. The switching behavior of up to 8 slots in the ET 200S (e.g. for power modules) can be defined with the instruction "PE_DS3_Write_ET200S".

Note

This instruction is not part of the PROIenergy profile but instead supplements SIMATIC-specific functions.

Parameters

The following table shows the parameters of the "PE_DS3_Write_ET200S" instruction:

Parameter	Declaration	Data type	Memory area	Description
ENABLE	Input	BOOL	I, Q, M, D, L	A positive edge triggers the transfer of the data record. The data record must be transferred again after voltage OFF/ON.
ID	Input	DWORD	I, Q, M, D, L or constant	Address of the ET 200S The address may be taken from the hardware configuration.
SLOT_NO_X	Input	INT	I, Q, M, D, L, P or constant	Slot number of switchable power module X.

Parameter	Declaration	Data type	Memory area	Description
FUNC_X	Input	INT	I, Q, M, D, L, P or constant	Function of the module in this slot. Use the parameter FUNC_X to define the switching behavior of the PM-E (power module of the ET 200S): • FALSE: – With "PAUSE_START": -No effect on PM-E -PM-E remains on – With "PAUSE_STOP": -Switches PM_E back on • TRUE: – With "PAUSE_START": -Switches PM_E off – With "PAUSE_STOP": -Switches PM-E back on
BUSY	Output	BOOL	I, Q, M, D, L	Transfer not yet complete.
DONE	Output	BOOL	I, Q, M, D, L	Transfer completed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Transfer completed with error.
STATUS	Output	DWORD	I, Q, M, D, L, P	Error number (see STATUS parameter of the instruction "PE_Start_End (Page 5306)")

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

PE_WOL: Starting and stopping energy-saving mode via WakeOnLan (S7-300, S7-400)

Description PE_WOL (S7-300, S7-400)

Description

The "PE_WOL" instruction sends the PROFIenergy commands "Start_Pause" and "End_Pause" to multiple PROFIenergy-enabled devices in the PROFINET I/O systems.

Multiple PE devices can be coordinated via the instruction provided that the PE devices support the "Wake on LAN" function over a UDP connection.

The "PE_WOL" instruction can only be executed on a CPU with and integrated Ethernet interface. This CPU should be able to load blocks with a size of about 40 KB. You cannot use the block with PROFINET I/O systems that are connected via an Ethernet CP.

The instruction "PE_WOL" is executed asynchronously.

Definition: Wake on LAN

The Wake on LAN function allows data processing equipment to resume working from a state near total shutdown when a special Ethernet packet is received.

For this procedure to work, the data processing equipment must have a network controller that is capable of receiving such a packet.

This packet (Magic Packet™) has a special format. It contains the MAC address of the network adapter 15 times.

Selection of devices

The devices are selected using the user data block at the parameter PENERGY. The user DB represents the database for processing multiple devices.

Prior to initialization of "PE_WOL", you must store at least the following information in the user DB:

- ID of the PROFINET I/O system
- Data of the connection which is used for "Wake On LAN".
- Port number which is used for "Wake On LAN".
- For each device
  - Pause time (PauseTime)
  - Switching of the device to the PE_SLEEP_MODE (EnableSleep)

You initialize the "PE_WOL" instruction with the COM_RST parameter. The jobs stored in the user DB are processed one after the other after initialization.

The graphic below illustrates how the PE command "Start_Pause" is transmitted to several devices:

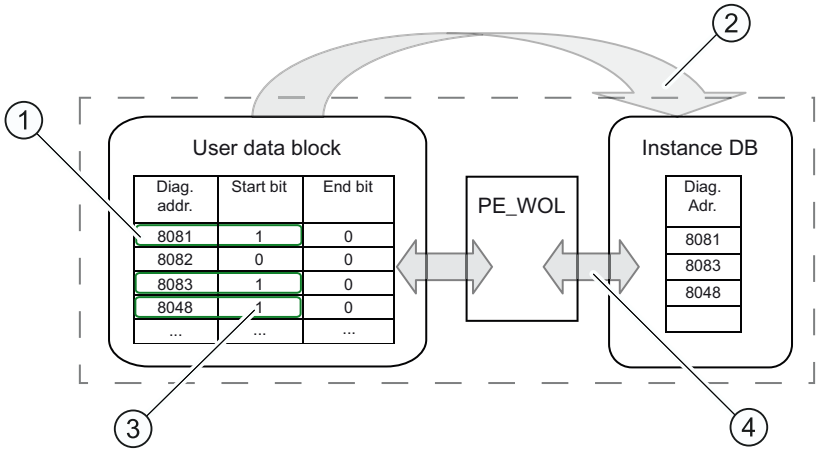

(1) Step 1: The bit "CmdStartPause" of the devices to be shut down is set to "1" by the user.

(2) Step 2: The diagnostic addresses of the devices to be shut down (CmdStartPause = "1") are linked to the queue.

(3) Step 3: The bit "CmdStartPause" is reset automatically once the jobs have been linked.

(4) Step 4: The instruction "PE_WOL" processes the jobs as soon as they are linked.

A PROFIenergy command "CmdStartPause" or "CmdEndPause" can be sent for all recognized devices in the PROFINET IO system via the START and END parameters.

The status of the job processing as well as possible errors during processing are output by the STATUS parameter.

Operation of the instruction using the user DB

The operation of the instruction "PE_WOL" can only take place via the user DB. A basic procedure applies in this case:

1. Selection of the command to be executed for a device:
   - START_PAUSE ("CmdStartPause" in user DB)
   - ENDE_PAUSE ("CmdEndPause" in user DB)
   - UPDATE_STATUS ("CmdUpdateStatus" in user DB)
2. Setting the bit for updating ("Update" in the header of the user DB)
   At least one CPU cycle should pass with an "Update" = False between two updates; otherwise, edge detection cannot be guaranteed.

Prioritizing the PE commands

The graphic below shows the chronological sequence of three possible commands.

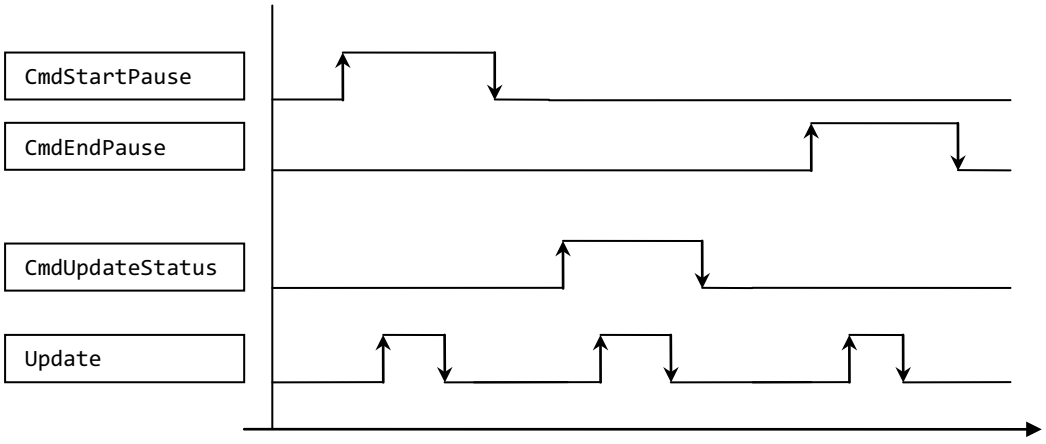

These are processed one after the other, regardless of whether the previous command call was successful or completed with an error.

Only one command is executed if you set two commands, such as "CmdEndPause" and "CmdUpdateStatus" simultaneously. A prioritization is present within the block:

- The command "CmdStartPause" has the highest priority and is therefore always executed if selected.
- The command "CmdEndPause" has the second highest priority.
- The command "CmdUpdateStatus" has the lowest priority.

If all three commands are set at the same time, the commands that are not executed remain preselected. In this case, the next rising edge executes the next command.

Parameter

The following table shows the parameters of the "PE_WOL" instruction:

Parameter	Declaration	Data type	Memory area	Description
COM_RST (Page 5318)	Input	BOOL	I, Q, M, D, L	Resets the block and performs a re-initialization. As long as True is set here, the initialization is started but not yet fully completed. Only the falling edge continues the initialization and switches to the normal operating mode after initialization.
START (Page 5320)	Input	BOOL	I, Q, M, D, L	A rising edge performs a "CmdStartPause" PROFIenergy command for all detected devices that support this function.
END (Page 5321)	Input	BOOL	I, Q, M, D, L	A rising edge performs a "CmdEndPause" PROFIenergy command for all detected devices that support this function.
PENERGY (Page 5321)	InOut	ANY	D	Pointer to the user DB that contains the database for processing multiple devices.
STATUS (Page 5326)	Output	DWORD	I, Q, M, D, L	Status/error number for the current status of the instruction (see "STATUS parameter").

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Parameter COM_RST (S7-300, S7-400)

Sequence of the initialization routine

You start the initialization of the "PE_WOL" instruction with the COM_RST parameter.

The following flow diagram shows the initialization routine.

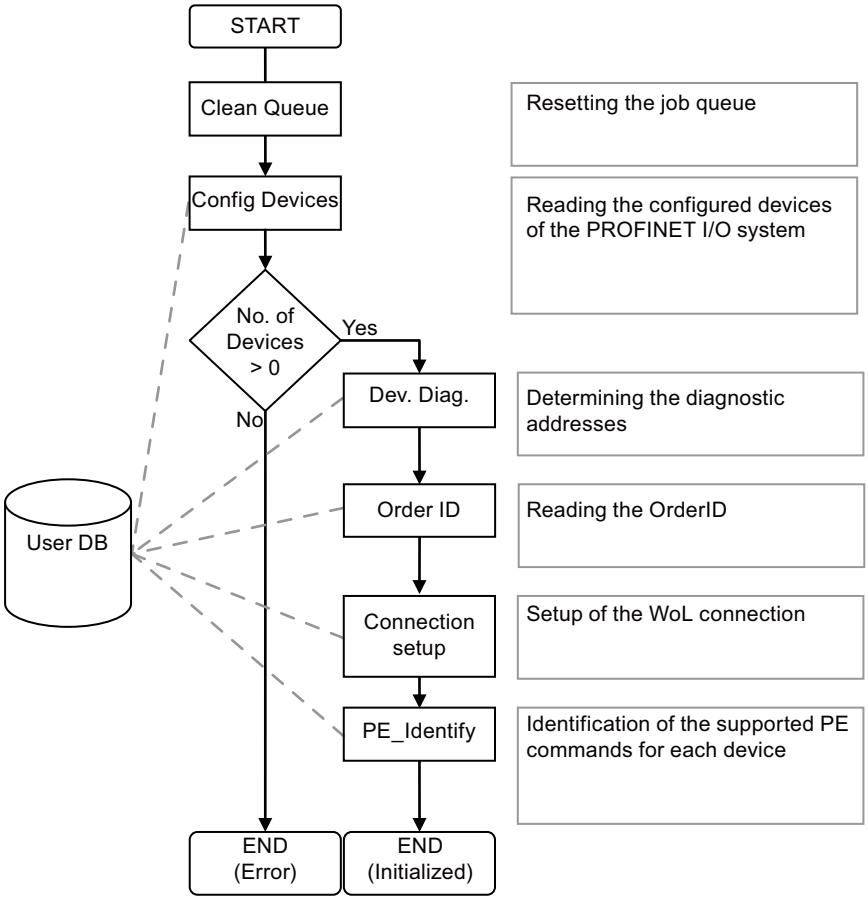
START
Clean Queue
Resetting the job queue
Config Devices
Reading the configured devices of the PROFINET I/O system
No. of Devices > 0
Yes
No
Dev. Diag.
Determining the diagnostic addresses
Order ID
Reading the OrderID
User DB
Connection setup
Setup of the WoL connection
PE_Identify
Identification of the supported PE commands for each device
END (Error)
END (Initialized)

Parameter START (S7-300, S7-400)

Sequence of the CmdStartPause command

The flow diagram shows the internally used functions and interaction with a device when executing the CmdStartPause command.

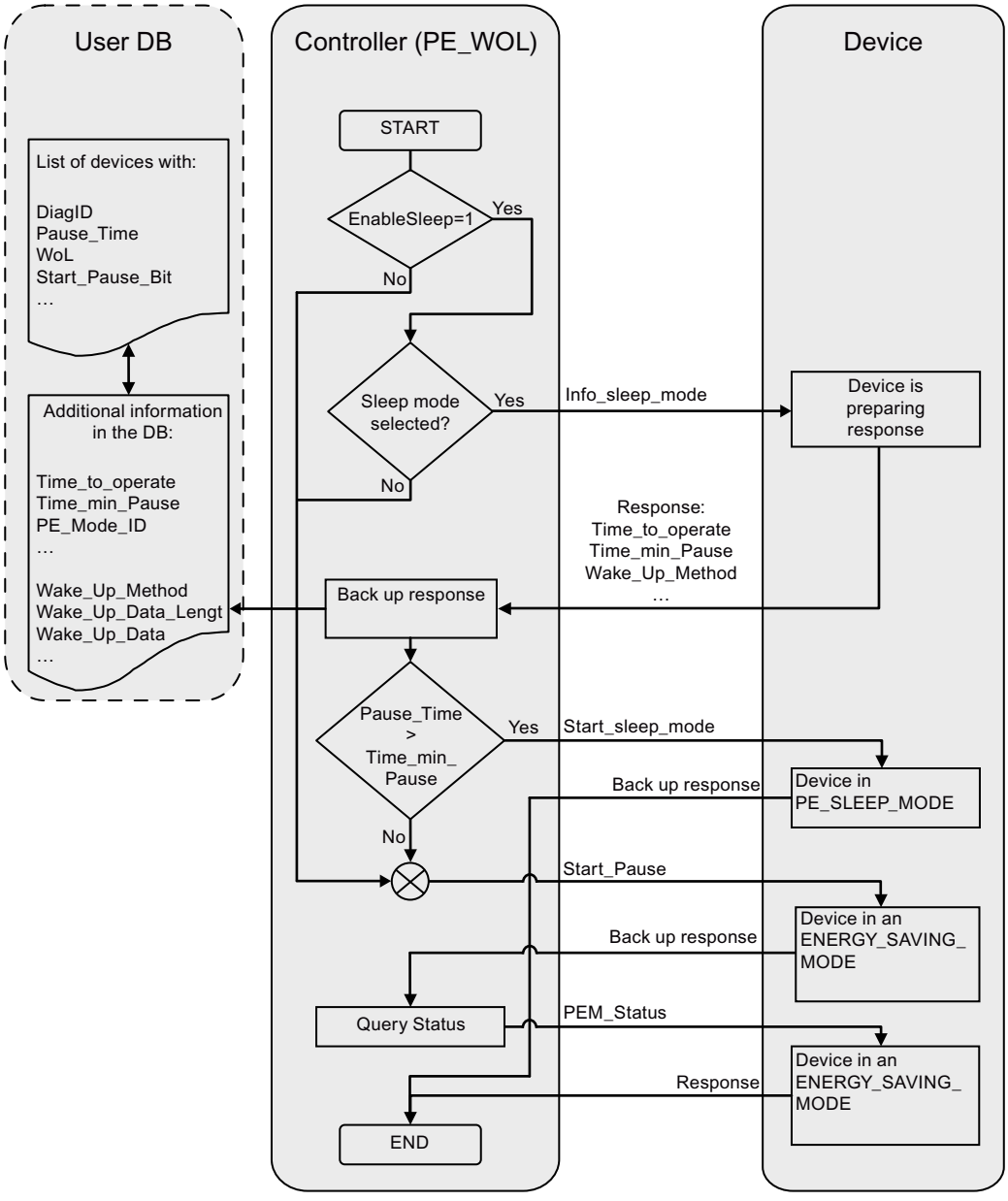

Parameter END (S7-300, S7-400)

Sequence of the CmdEndPause command

The flow diagram shows the internally used functions and interaction with a device when executing the CmdEndPause command.

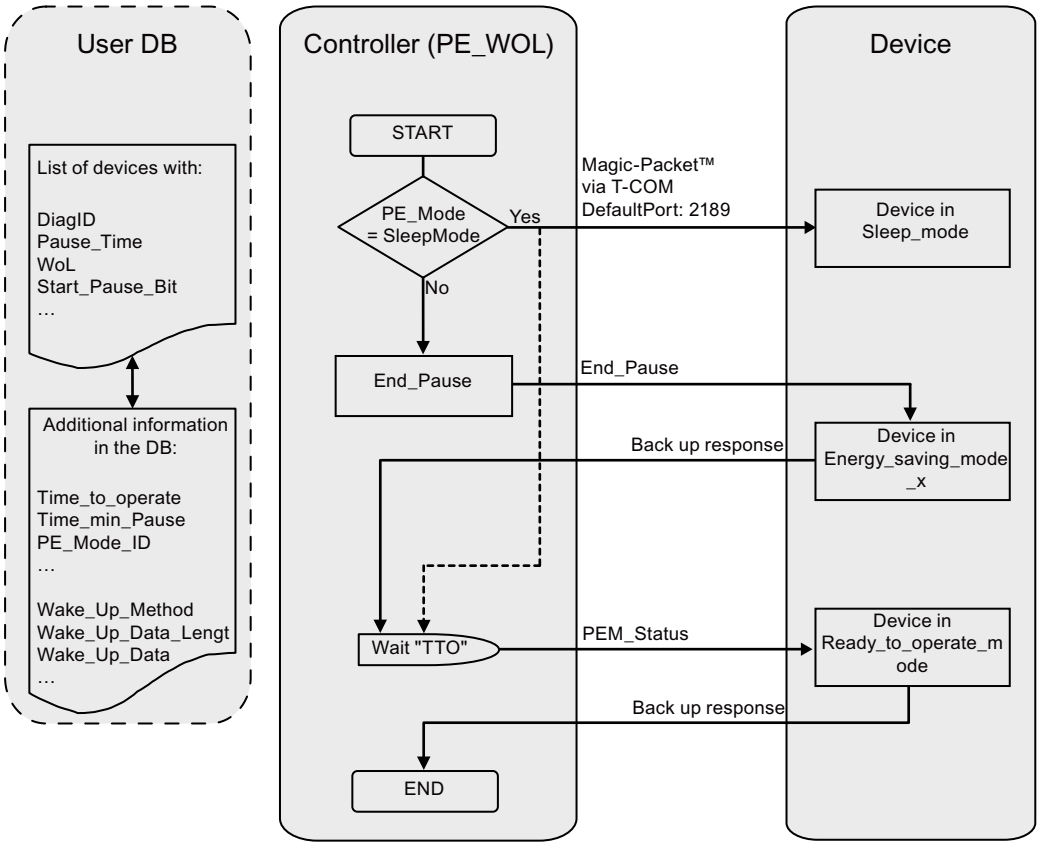

PENERGY parameter (S7-300, S7-400)

Data block at the PENERGY parameter

The user DB for the "PE_WOL" PROFIenergy instruction represents a data base for processing multiple devices.

The data block is usually divided into two parts. These are:

- Header 110 bytes
- Device section for a maximum of 256 devices each with 100 bytes (Device). These include:
  - Device-specific data (Device)
  - PROFIenergy-specific data (PE)
  - Data for the job processing (Task)
  - User data (UserData)

The individual sections have a certain size, which is constant. Overall, these result in a constant size of 25,746 bytes for the data block.

Connection parameter "Connection"

The "PE_WOL" instruction reserves a connection resource in the area of "Open User Communication". This is used as a UDP connection. The following parameters must be defined in the data block for this:

- Connection ID ("Connection.id" parameter)
  The connection ID is an integer within the range of 1 to 32. The default setting is 31. It is used to identify the communication resources allocated by the firmware, for example the send and receive buffers.
  The connection ID must be unique throughout the CPU.
- Port number that is used for the "Wake on LAN" function ("Header.PortNo" parameter)
  Number of the UDP port via which a "Wake On LAN" packet is sent. These port numbers are part of the communication resources, which are identified and provided by the firmware through the connection ID. The default setting for port 2189 used here is currently not assigned by IANA. The port number is transferred to the connection configuration. This affects the "Connection.local_tsap_id[1]" and "Connection.rem_tsap_id[1]" parameters.
- Interface ID ("Connection.local_device_id" parameter)
  The interface ID is also part of the connection description. This ID identifies the CPU interface to use for this connection. There are several possible valid values.
  However, these need to be adapted to the employed CPU and interface:
  - B#16#01 for ET200S CPU or WinAC RTX with Ethernet interface in subslot IF1
  - B#16#02 for CPU 315(F)-2PN/DP or CPU 317(F)-2PN/DP
  - B#16#03 for CPU 319(F)-2PN/DP
  - B#16#05 for CPU 41x(F)-3PN/DP
  - B#16#06 for WinAC RTX with Ethernet interface in subslot IF2
  - B#16#0B for WinAC RTX with Ethernet interface in subslot IF3
  - B#16#0F for WinAC RTX with Ethernet interface in subslot IF4

Structure of the data block

The data block has the following structure:

Name	Data type	Offset	Comment
Header	STRUCT	-	Header information
Update (1)	BOOL	0.0	Signal for indicating a change in the data area. • True = indicates a change by the user. • False = indicates the application of changes.
Initialized	BOOL	0.1	Signal indicating the completed initialization. • True = initialization completed. • False = indicates that the block is not initialized.
LinkUp	BOOL	0.2	Indicates successful configuration of the Ethernet interface. • True = interface ready-to-use. • False = interface not yet configured.
LinkDown	BOOL	0.3	Indicates an unconfigured interface. • True = interface is not configured. • False = interface is currently being configured or is now configured.
PROFINET_ID (1)	INT	2.0	ID of the PROFINET I/O system
Reserved	ARRAY [1..37] OF BYTE	4.0	Reserved
LastDeviceID	INT	42.0	Contains the highest device ID in the PROFINET I/O system.
PortNo (1)	INT	44.0	Port number that is used for the "Wake on LAN" function (default= 2189).
Connection	STRUCT	-	Contains the connection configuration of the "Wake on LAN" connection.

Name	Data type	Offset	Comment
block_length	WORD	46.0	Length of the structure (always B#16#40).
id (1)	WORD	48.0	Connection ID
connection_type (1)	BYTE	50.0	Connection type (UDP = B#16#13)
active_est (1)	BOOL	51.0	Active connection establishment (always passive for UDP)
local_device_id (1)	BYTE	52.0	Contains the interface ID (CPU dependent)
local_tsap_id_len (1)	BYTE	53.0	Contains the length in bytes of the own/local UDP port
rem_subnet_id_len (1)	BYTE	54.0	Unused (always B#16#00)
rem_staddr_len (1)	BYTE	55.0	Contains the length of the remote IP address or B#16#00.
rem_tsap_id_len (1)	BYTE	56.0	Contains the length in bytes of the remote UDP port.
next_staddr_len (1)	BYTE	57.0	Contains the length of the default router address (not relevant).
local_tsap_id (1)	ARRAY[1..8] OF INT	58.0	Contains your own local port number.
rem_subnet_id (1)	ARRAY[1..6] OF BYTE	74.0	Unused (always B#16#00)
rem_staddr (1)	ARRAY[1..6] OF BYTE	80.0	Contains the remote IP address.
rem_tsap_id (1)	ARRAY[1..8] OF INT	86.0	Contains the remote UDP port number.
next_staddr (1)	ARRAY[1..6] OF BYTE	102.0	Irrelevant
spare (1)	WORD	108.0	
Device	ARRAY[1..256] OF STRUCT	110.0	Array of devices
Device	STRUCT	110.0	Contains data for each device
DiagID	WORD	110.0	Diagnostic address of the device. Assigned by the hardware configuration.
InterfaceID	WORD	112.0	Diagnostic address of the device interface. Assigned by the hardware configuration.
MACAdr	ARRAY[1..6] OF BYTE	114.0	Contains the MAC address of the device.
IPAdr	ARRAY[1..4] OF BYTE	120.0	Contains the IP address of the device.
OrderID	STRUCT	124.0	Contains the OrderID of the device.
MxLen	BYTE		Contains the maximum length of the array.
ActLen	BYTE		Contains the current number of data in the array.
Data	ARRAY[1..20] CHAR		Contains the data.
PE	STRUCT	146.0	PROFIenergy-specific data.

Name	Data type	Offset	Comment
ModeID	BYTE	146.0	PE_MODE_ID according to Profienergy specification.
Result	BYTE	147.0	PEErrorCode according to Profienergy specification.
PauseTime (1)	TIME	148.0	Contains the idle time in ms.
TimeToPause	TIME	152.0	Contains the time it takes for the device to go into Pause mode.
TimeToOperate	TIME	156.0	Contains the time it takes for the device to go into operating mode.
MinSleepTime	TIME	160.0	Contains the minimum time of the device in PE_SLEEP_MODE.
SleepToOperate	TIME	164.0	Contains the time it takes for the device to go from PE_SLEEP_MODE into operating mode.
StatusOperate	BOOL	168.0	Indicates the operating mode of the device.
StatusPause	BOOL	168.1	Indicates the device is in Pause mode.
StatusSleep	BOOL	168.2	Indicates the PE_SLEEP_MODE of the device.
StatusTransitOK	BOOL	168.3	Indicates the transition from one energy state to another.
StatusInTransit	BOOL	168.4	Indicates a current state transition.
StatusTransitNOK	BOOL	168.5	Indicates that the state change was not successful.
StatusError	BOOL	168.6	Indicates an error with the device.
StatusRetryEx	BOOL	168.7	Indicates an unsuccessful execution of a command. No more attempts will be made to execute this command.
CmdStartPause (1)	BOOL	169.0	Places a START_PAUSE command for this device in the queue.
CmdEndPause (1)	BOOL	169.1	Places a END_PAUSE command for this device in the queue.
CmdUpdateStatus (1)	BOOL	169.2	Places a PEM_STATUS command for this device in the queue.
EnableSleep (1)	BOOL	169.3	Allows PE_SLEEP_MODE for this device. • True = device should go to PE_SLEEP_MODE when idle time is too long • False = device may not go into PE_SLEEP_MODE.
Services	WORD	170.0	Shows all supported PROFIenergy services.
UserData (2)	ARRAY[1..24] OF BYTE	172.0	User-defined data
Task	STRUCT	196.0	Job processing

Name	Data type	Offset	Comment
Cmd	BYTE	196.0	Internal bits for job processing
CmdJ	BYTE	197.0	Internal bits for job processing
TimeStart	BOOL	198.0	Starts a delay time.
TimeStarted	BOOL	198.1	The delay time has just started.
TimeDone	BOOL	198.2	Shows the expiration of the delay time.
Done	BOOL	198.3	Indicates that this job has been completed.
DelayedCmd	BOOL	198.4	Indicates that a delayed command is still pending.
IsV1_0	BOOL	198.5	Indicates that this device is a Spec. V1.0 device.
IsWakeOnLAN	BOOL	198.6	Indicates that this device is woken up by "Wake On LAN".
RetryCount	BYTE	199.0	Repetition counter for PE_COMMANDS
Duration	TIME	200.0	Contains the value for the delay in ms.
StartTime	TIME	204.0	Contains the start time for the delay time.
MachineState	INT	208.0	Contains the internal status of the job.

(1) To be filled in by the user.
(2) Available for use by the user.

Parameter STATUS (S7-300, S7-400)

Parameter STATUS

The output value at the STATUS parameter is divided into three areas:

- Bits 31 to 24: MESSAGE
- Bit 23 to 16: LOCATION
- Bit 15 to 0: INFORMATION

You can learn the meaning of each error code for the three areas in the following tables:

Table 4-192 Possible values for MESSAGE

Error code (W#16#...)	Description
00	No error.
50	Instruction initialized.
51	Determining configuration of the PROFINET I/O system.
52	The instruction could not determine any configured devices in the PROFINET I/O system.
53	Determining the logical addresses of configured devices.
54	Reading interface information out of the devices.
55	Determining I&M data (data record 0 only) of the configured devices.
56	Configuring PROFINET interface for sending the "Wake on LAN" MagicPaket™ via UDP.
57	Determining PROFIenergy capabilities of the connected devices.
62	Invalid PROFINET I/O system ID detected. The causative number is displayed in the INFORMATION field.

Error code (W#16#...)	Description
70	The instruction is initiated and processing jobs. The value in the INFORMATION field contains the number of currently active jobs.
80	The instruction has lost the initialization during processing of jobs. This generally occurs when either the instance DB or the user DB has been reloaded.
FF	Unknown error has occurred.

Table 4-193 Possible values for LOCATION

Error code (W#16#...)	Description
00	The instruction is either not initialized or it is idle.
70	The instruction is waiting for jobs.
71	The instruction has appended a job to the job list.
72	The instruction is preparing to send a job.
73	The instruction is sending a job to a device.
74	The instruction is waiting for a response from the device.
75	The instruction is evaluating the response of the device.
76	The instruction is removing the job from the job list.
FF	Unknown error has occurred.

Table 4-194 Possible values for INFORMATION

Error code (W#16#...)	Description
0000	No additional information available and no jobs active.
0001 -00FF	1 – 255 jobs are currently being processed.
8001	Error at parameter 1
8002	Error at parameter 2
8003	Error at parameter 3
8004	Error at parameter 4
8005	Error at parameter 5. This error is reported if there is no interconnection or an invalid interconnection to the user DB. Possible causes: • The user DB is too small. • The user DB is write-protected. • There is no user DB in the RAM. • The user DB is invalid for the employed CPU.
8100	An attempt has been made to place more than the maximum of 256 possible jobs. This is a temporary error, which is resolved by completing a few jobs. The placed jobs are not accepted and must be placed again.
8200	An attempt has been made to send an invalid or an unsupported PROFIenergy command (PE_COMMAND).

Error code (W#16#...)	Description
84xx	A communication error has occurred. The number of the device that generated the error is output to "xx".
85xx	An error has been returned from device xx. The number of the device that generated the error is output to "xx".
8600	The requested wake-up method is currently not supported.
FFFF	Unknown error has occurred.

PROFIenergy commands (S7-300, S7-400)

PI Command "Start_Pause" (S7-300, S7-400)

Description

Use the PE command "Start_Pause" to start the energy-saving mode. The command Start_Pause can be used to:

- Switch the PE from "ready" state (PE_ready_to_operate) to an energy-saving mode (PE_energy_saving_mode).
- The PE entity can automatically switch between the energy-saving modes.
  Energy consumption can increase or decrease when you switch energy-saving mode.

Calling the PI command "Start_Pause"

The command "Start_Pause" is called with the instruction "PE_CMD (Page 5310)" with the following parameters:

Parameter	Value	Description
CMD	1	Call of PE command "Start_Pause".
CMD_MODIFIER	0	There are no additional specifications of the command call for command "Start_Pause".
CMD_PARA_LEN	4	Length of the CMD_PARA parameter of 4 bytes.
CMD_PARA	ANY	ANY pointer to the value for "Pause_Time" (TIME).

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at at parameter RESPONSE_DATA (see instruction "PE_CMD (Page 5310)"):

Attribute	Value	Data type	Description
PE_Mode_ID	1 to 255	BYTE	Identification number of the energy-saving mode
Reserved	0	BYTE	-

PI Command "End_Pause" (S7-300, S7-400)

Description

Use the PE command "End_Pause" to terminate the energy-saving mode of the PE entity.

Calling the PI command "End_Pause"

The command "End_Pause" is called with the instruction "PE_CMD (Page 5310)" with the following parameters:

Parameter	Value	Description
CMD	2	Call of PE command "End_Pause".
CMD_MODIFIER	0	There are no additional specifications of the command call for command "End_Pause".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 5310)"):

Attribute	Value	Data type	Description
Time_to_operate	-	DWORD	Expected time it takes to switch to the "PE_ready_to_operate" mode.

PI command "Query_modes" - "List_Energy_Saving_Modes" (S7-300, S7-400)

Description

Use PE command "Query_modes" and sub-command (modifier) "List_Energy_Saving_Modes" to output all energy-saving modes (PE_Mode_ID) supported by the PE entity.

The query result is written in the form of a response frame to the data block referenced by theRESPONSE_DATA parameter.

Calling the PI command "Query_modes" - "List_Energy_Saving_Modes"

The command "List_Energy_Saving_Modes" is called with the instruction "PE_CMD (Page 5310)" with the following parameters:

Parameter	Value	Description
CMD	3	Call of PE command "Query_modes".
CMD_MODIFIER	1	Specification of the command call: Select the sub-command "List_Energy_Saving_Modes" to output the number and types of energy-saving modes supported.

Parameter	Value	Description
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 5310)"):

Attribute	Value	Data type	Description
Number_of_PE_ Mode_IDs	1	BYTE	Number of energy-saving modes.
PE_Mode_IDs	-	Array [...] of BYTE	Array with the IDs of the supported energy-saving modes. The significance of the IDs depends on the PE entity.

PI command "Query_modes" - "Get_Mode" (S7-300, S7-400)

Description

You use the PE command "Query_modes" and the sub-command (modifier) "Get_Mode" to output the attributes of the energy-saving mode which is currently enabled.

Calling the PI command "Query_modes" - "Get_Mode"

The call of the command with the instruction "PE_CMD" occurs with the following parameters:

Parameter	Value	Description
CMD	3	Call of the PE command "Query_modes"
CMD_MODIFIER	2	Specification of the command call: Select the sub-command "Get_Mode" to output the status of the mode which is currently enabled.
CMD_PARA_LEN	1	Length of the CMD_PARA parameter of 1 byte.
CMD_PARA	ANY	ANY pointer to the value for PE_MODE_ID.

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 5310)"):

Attribute	Value	Data type	Description
PE_Mode_ID	• 0 "PE_power_off" Mode • 1...254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" Mode	BYTE	ID of the energy-saving mode currently enabled.
PE_Mode_Attributes	Bit 0: • = 0: Only static energy consumer and time values are available. • = 1: Dynamic energy consumer and time values are available. Bit 1 to 7: • Reserved	BYTE	
Time_min_Pause [1]	Time difference without date	DWORD	Minimum pause time for the PI mode. The minimum pause time is the sum of the values of the following attributes: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay See the description of the "PAUSE_TIME parameter" of the instruction "PE_START_END: Start and exit energy-saving mode (Page 5306)".
Time_to_Pause [1]	Time difference without date	DWORD	Switch off time: Duration from call of energy-saving mode to start of energy-saving mode (transition duration of PE_ready_to_operate to PE_energy_saving_mode). The switch off time depends on the PE entity.
Time_to_operate [1]	Time difference without date	DWORD	Switch on time: Duration of the transition from energy-saving mode (PE_energy_saving_mode) to ready to operate mode (PE_ready_to_operate). The PE entity calculates the duration dynamically in the output operation.
Time_min_length_of_stay [1]		DWORD	Minimum active period of the energy-saving mode on the PE entity.
Time_max_length_of_stay [1]		DWORD	Maximum active period of the energy-saving mode on the PE entity.
Mode_Power_Consumption [2]		REAL	Power consumption of the PE entity in active energy-saving mode. Unit: kW

Attribute	Value	Data type	Description
Energy_ Consumption_ to_pause [2]		REAL	Energy consumption of the PE entity at the transition from the ready to operate mode (PE_ready_to_operate) to the energy-saving mode (PE_energy_saving_mode) Unit: kWh
Energy_ Consumption_ to_operate [2]		REAL	Energy consumption of the PE entity at the transition from the energy-saving mode (PE_energy_saving_mode) to the ready to operate mode (PE_ready_to_operate) Unit: kWh
[1] If the duration is infinite, 0xFFFFFFFF will be output. If the duration is zero, "0" will be output. [2] If the PE entity has no defined energy and power consumption, the "0.0" is output.			

PI command "PEM_Status" (S7-300, S7-400)

Description

Using PE command "PEM_Status", you can query the currently active energy-saving mode of a PE entity.

Calling the PI command "PEM_Status"

The command "PEM_Status" is called with the instruction "PE_CMD" with the following parameters:

Parameter	Value	Description
CMD	4	Call of PE command "PEM_Status".
CMD_MODIFIER	0	There are no other specifications of the command call for command "PEM_Status".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 5310)"):

Attribute	Value	Data type	Description
PE_Mode_ID_ Source	• 0 "PE_power_off" mode • 1 to 254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	Mode of the PE entity before the PE command is sent.
PE_Mode_ID_ Destination	• 0 "PE_power_off" mode • 1 to 254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	Mode of the PE entity after a PE command is executed.
Time_to_operate	Time difference without date	DWORD	Switch on time: Duration of the transition from energy-saving mode (PE_energy_saving_mode) to ready to operate mode (PE_ready_to_operate). The PE entity calculates the duration dynamically in the output operation.
Remaining_time_to_ destination	Time difference without date	DWORD	Remaining time to switch to another mode.
Mode_Power_ Consumption		REAL	Power consumption of the PE entity in active energy-saving mode. Unit: kW
Energy_ Consumption_ to_Destination		REAL	Energy consumption for the current PI transition Unit: kWh
Energy_ Consumption_ to_operate		REAL	Energy consumption of the PE entity at the transition from the energy-saving mode (PE_energy-saving mode) to the ready to operate mode (PE_ready_to_operate) Unit: kWh

PI command "PE_Identify" (S7-300, S7-400)

Description

Use PE command "PE_Identify" to read the number and description of PE commands supported by the PE entity. The number of specific commands supported depends on the PE entity. As PE_Identify is itself a PE command, at least three supported PE commands will be output upon a positive response: Start_Pause, End_Pause and PE_Identify.

Calling the PI command "PE_Identify"

The command "PE_Identify" is called with the instruction "PE_CMD (Page 5310)" with the following parameters:

Parameter	Value	Description
CMD	5	Call of the command "PE_Identify".
CMD_MODIFIER	0	There are no other specifications of the command call for command "PE_Identify".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 5310)"):

Attribute	Value	Data type	Description
Count [1]	6	BYTE	Number of supported PROFIenergy commands
Start_Pause	1	BYTE	First PE command supported (Service_Request_ID)
End_Pause	2	BYTE	...
Query_Modes	3	BYTE	...
PEM_Status	4	BYTE	...
PE_Identify	5	BYTE	...
Query_Measurement	16	BYTE	Last PE command supported (Service_Request_ID)
[1] The number of commands supported is manufacturer-specific and depends on the PE entity used. The values given are an example of a response frame where all 6 PE commands are supported.			

PI Command "Query_Measurement" - "Get_Measurement_list" (S7-300, S7-400)

Description

Use PE command "Query_Measurement" and sub-command (modifier) "Get_measurement_list" to query the specific measured values supported by the PE entity. The supported measured values are output as a list in the data block referenced by the RESPONSE_DATA parameter.

Calling the PI command "Query_Measurement" - "Get_Measurement_list"

The command is called with the instruction "PE_CMD (Page 5310)" with the following parameters:

Parameter	Value	Description
CMD	16	Call of the command "Query_Measurement".
CMD_MODIFIER	1	Specification of the command call: Select the sub-command "Get_Measurement_List" to output a list of supported measured values.
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at at parameter RESPONSE_DATA (see "PE_CMD (Page 5310)"):

Attribute	Value	Data type	Description
Count	-	BYTE	Number of measurement IDs
reserved	-	BYTE	
...			
Measurement_ID	-	WORD	First supported Measurement_ID The Measurement_ID is manufacturer-specific. For more information, refer to the manual of the respective PE entity.
Accuracy_Domain	-	BYTE	See"accuracy domain" table.
Accuracy_Class	-	BYTE	See "accuracy classes" table.
Range	-	REAL	Specifies the scale end value of the measured value (only for accuracy domain 1). The attribute range uses the same unit as defined using attribute Measurement_ID (only one unit is used for each Measurement_ID).
...			
Measurement_ID	-	WORD	Last supported Measurement_ID
Accuracy_Domain	-	BYTE	See"accuracy domain" table.
Accuracy_Class	-	BYTE	See "accuracy classes" table.
Range	-	REAL	Specifies the scale end value of the measurement value (only for accuracy domain 1). The attribute range uses the same unit as defined using attribute Measurement_ID (only one unit is used for each Measurement_ID).

Accuracy domains

Accuracy domain	Description
0	Reserved
1	The accuracy deviation is output in percentages of the scale end value. The percentage of the possible deviation is divided into accuracy classes (see table: accuracy classes of the accuracy domains 1 and 2).

Accuracy domain	Description
2	The accuracy deviation is output in percentages of the current measurement value. The percentage of the possible deviation is divided into accuracy classes (see table: accuracy classes of the accuracy domains 1 and 2).
3	The measuring accuracy adheres to the standard IEC 61557-12. The function performance classes for performance measurement and monitoring devices (PMD) without external sensors, and the system performance classes for PMD with external sensors, are coded as set out in the "Accuracy classes of accuracy domain 3" table.
4	The entry of the accuracy adheres to the standard EN 50470-3, chapter 8 (see also table: accuracy classes of accuracy domain 4).

Accuracy classes

Table 4-195 Accuracy classes of the accuracy domains 1 and 2

Accuracy class	0	1	2	3	4	5	6	7	8
Meaning	Reserved	0.01%	0.02%	0.05%	0.1%	0.2%	0.5%	1%	1.5%

Accuracy class	9	10	11	12	13	14	15	>15
Meaning	2%	2.5%	3%	5%	10%	20%	>20%	Not defined

Table 4-196 Accuracy classes of accuracy domain 3

Accuracy class	0	1	2	3	4	5	6	7	8
Meaning	Reserved	0.02	0.05	0.1	0.2	0.5	1	1.5	2

Accuracy class	9	10	11	12	13	14	>13
Meaning	2.5	3	5	10	20	20%	Not defined

Table 4-197 Accuracy classes of accuracy domain 4

Accuracy class	0	1	2	3	4	5	6	>7
Meaning	Reserved	0.5	1.0	1.5	2.0	2.5	3.0	Not defined

PI Command "Query_Measurement" - "Get_Measurement_values" (S7-300, S7-400)

Description

Use PE command "Query_Measurement" and sub-command (modifier) "Get_measurement_values" to output a list of measured values supported by the PE entity. The measured values are output as a list in the data block referenced by the RESPONSE_DATA parameter.

Calling the PI command "Query_Measurement" - "Get_Measurement_values"

The command is called with the instruction "PE_CMD (Page 5310)" with the following parameters:

Parameter	Value	Description
CMD	16	Call of the command "Query_Measurement".
CMD_MODIFIER	2	Specification of the command call: Select the command "Get_Measurement_Values" to output a list of supported measurement values.
CMD_PARA_LEN	0	Depends on the number of measurement values. The length of the parameters results from the attribute count and the sum of the lengths of the attributes for the transferred measurement values.
CMD_PARA	ANY	ANY pointer to data structure with list of measured values to be queried (see "CMD_PARA parameter").

Parameter CMD_PARA

The structure specified with the ANY pointer at the CMD_PARA parameter must have the following setup:

Attribute	Value	Data type	Description
Count	-	BYTE	Number of measured values (Measurement-IDs)
reserved	0	BYTE	Not used
Measurement_ID	-	WORD	First measured value queried
...			
Measurement_ID	-	WORD	Last measured value queried

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 5310)"):

Attribute	Value	Data type	Description
Count [1]	-	BYTE	Number of measured values (Measurement-IDs)
reserved	0	BYTE	Not used
Length_of_Structure	2 to 65535	WORD	Length of the structure in bytes

Attribute	Value	Data type	Description
Measurement_Data_Structure_ID	1 = simple value	BYTE	Defines the following structure.
Measurement_ID	0 to 65535	WORD	ID of the supported measurement value.
Status_of_ Measurement_Value	1 to 3	BYTE	Status of the measurement value: • 1: Valid • 2: Not supported • 3: Invalid
Transmission_Data_Type	-	REAL	
End_of_demand	-	TOD	Optional time stamping with data type TimeOfDay.
...			
Length_of_Structure	-	WORD	Length of the structure in bytes
Measurement_Data_Structure_ID	-	BYTE	Defines the following structure.
Measurement_ID	-	WORD	ID of the supported measurement value.
Status_of_Measurement_Value	-	BYTE	Status of the measurement value: • 1: Valid • 2: Not supported • 3: Invalid
Transmission_Data_Type	-	REAL	
End_of_demand	-	TOD	Optional time stamping with data type TimeOfDay.
[1] If the data length of the measurement values queried exceeds the size of the PDU (Protocol Data Unit) of the protocol layer, the data will not be completely transferred and only the supported measurement values will be output.			

iDevice / iSlave (S7-300)

PE_I_DEV: Control PROFIenergy commands in the I-Device (S7-300)

Description

The instruction "PE_I_DEV" is used to process the PROFIenergy profile in the intelligent IO device (iDevice). The functions which are executed by the firmware in standard PROFIenergy-compatible IO devices, such as the ET 200S, are implemented in the iDevice by the instruction "PE_I_DEV" and the corresponding auxiliary blocks:

- The instruction "PE_I_DEV" is called cyclically by the user program of the iDevice and receives all PROFIenergy commands.
- The PROFIenergy response is generated by configuring an auxiliary block. The response in the pause is fully programmable. The response data must be provided within 10 seconds; otherwise, "State conflict 0x80B5" will occur at the STATUS parameter of the instruction in the IO controller.

No specific knowledge of the PROFIenergy standard is required to use the instruction.

PROFIenergy auxiliary blocks (PE auxiliary blocks)

The PE auxiliary blocks are used to generate the response frame. Enter the response data (in plain text) at the input parameters of the relevant block.

- For each PROFIenergy command, there is a corresponding auxiliary block for a positive response:
  - PE command "Start_Pause": PE_Start_RSP (Page 5343)
  - PE command "End_Pause": PE_End_RSP (Page 5344)
  - PE command "Query_modes" - "List_Energy_Saving_Modes": PE_List_Modes_RSP (Page 5345)
  - PE command "Query_modes" - "Get_Mode": PE_Get_Mode_RSP (Page 5347)
  - PE command "PEM_Status": PE_PEM_Status_RSP (Page 5349)
  - PE command "PE_Identify": PE_Identify_RSP (Page 5350)
  - PE command "Query_Measurement" - "Get_Measurement_list": PE_Measurement_List_RSP (Page 5352)
  - PE command "Query_Measurement" - "Get_Measurement_values": PE_Measurement_Value_RSP (Page 5353)
- Irrespective of the PROFIenergy command, there is an additional shared auxiliary block for a negative response (see PE_Error_RSP (Page 5342)).

Interconnection of the auxiliary blocks

The instruction "PE_I_DEV" and the auxiliary blocks are coordinated. Some of the parameters are simply interconnected. The diagram below shows which of the parameters need to be interconnected.

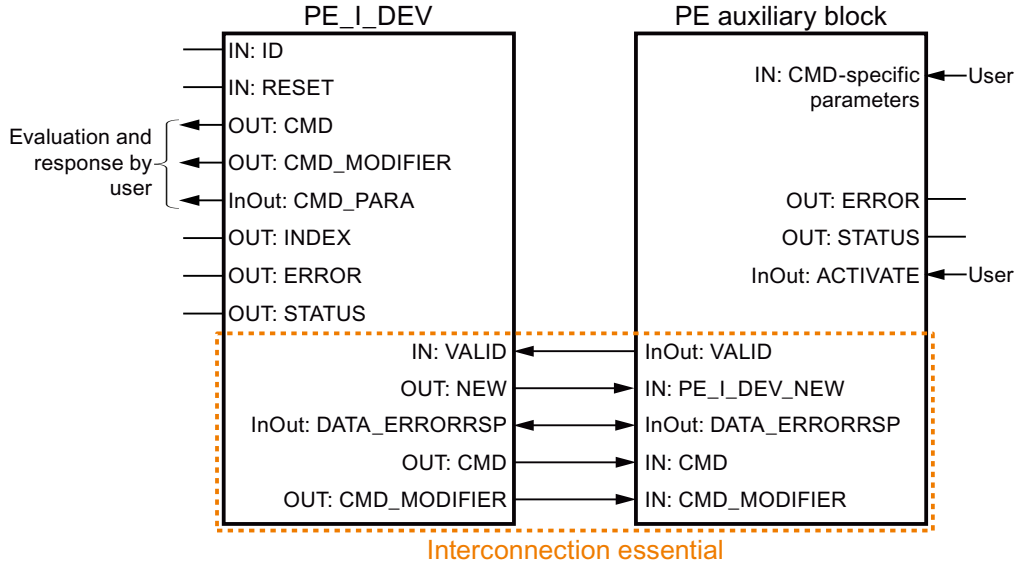

Parameters

The following table shows the parameters of the instruction "PE_I_DEV":

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	Resets the instruction.
ID	Input	DWORD	I, Q, M, D, L or constant	Address of the iDevice The address may be taken from the hardware configuration.
VALID	Input	BOOL	I, Q, M, D, L	The response data for the PROFIenergy controller is ready and can be sent.
CMD_PARA	Output	ANY	I, Q, M, D, L	Parameters for: • Get mode: PE_mode_ID • Get measurement values: List of Measurement_IDs (list of IDs of the tags to be read; either one tag or multiple tags may be read at any given time). Maximum length: 234 bytes
DATA_ERRORRSP	InOut	ANY	I, Q, M, D, L	Pointer to the data area containing the acknowledgment data for the PROFIenergy controller. This must correspond to the pointer which is also used with the auxiliary blocks.
INDEX	Output	INT	I, Q, M, D, L	Data record number of the PROFIenergy record (set at 0x80A0)
CMD	Output	INT	I, Q, M, D, L	Service-Request-ID of the PROFIenergy command in accordance with the PROFIenergy profile (see "CMD and CMD_MODIFIER parameters"). Further PE commands (Service-Request-IDs) are possible following extensions of the PROFIenergy profile.
CMD_MODIFIER	Output	INT	I, Q, M, D, L	PROFIenergy sub-command: • Only when CMD=3 or CMD=16; see "CMD and CMD_MODIFIER parameters". • For all other commands: "0". Further sub-commands are possible following extensions of the PROFIenergy profile.
NEW	Output	BOOL	I, Q, M, D, L	New data available from the PROFIenergy controller.
ERROR	Output	BOOL	I, Q, M, D, L	Command completed with error.
STATUS	Output	DWORD	I, Q, M, D, L	Error information (see "STATUS parameter").

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters CMD and CMD_MODIFIER

CMD	CMD_ MODIFIER	PROFIenergy command	Description
01	0	Start_Pause	Start energy-saving mode or switch to a different energy-saving mode.
02	0	End_Pause	End energy-saving mode.
03	1	Query_Modes - List energy saving Modes	Output the energy-saving modes supported.
	2	Query_Modes - Get Mode	Output attributes of the energy-saving mode currently enabled.
04	0	PEM_Status	Query status of energy-saving mode.
05	0	PE_Identify	Read out the number and description of PE commands supported.
16	1	Query_Measurement - Get_Measurement_List	List of measured values supported by the PE entity.
	2	Query_Measurement - Get_Measurement_Values	Output of the measured values of the PE entity.

STATUS parameter

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error during reading of data record • B#16#**DF**: Error during writing of data record • B#16#**C0**: Error message from the instruction "PE_I_DEV" or from the internal communication instructions ""PRVREC" and "RCVREC".
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific
STATUS[3]	Error_Code_1	Error ID (when Error_Decode = **80**): • **B1**: Write length error (error in write length or insufficient length information with the data type ANY).
STATUS[4]	Error_Code_2	For PROFINET errors: Output of IO controller error message If no PROFINET error has occurred, the value in STATUS[4] = "0".

Note

Error messages of the "PRVREC" and "RCVREC" instructions

The instruction "PE_I_DEV" uses the instructions "PRVREC (Page 5270)" and "RCVREC (Page 5268)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 5235) parameter for an explanation of the error codes of the "PRVREC" and "RCVREC" instructions.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

Auxiliary blocks of the instruction PE_I_DEV (S7-300)

PE_Error_RSP: Generate negative answer to command (S7-300)

Description

The auxiliary block "PE_Error_RSP" (Response with failure) generates a negative response if the command requested is not supported (either in general or temporarily). Response generation does not depend on the command requested.

Parameter

The following table shows the parameters of the "PE_Error_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
ERROR_CODE	Input	BYTE	I, Q, M, D, L or constant	Error number
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 5338)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 5338)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_Start_RSP: Generate answer to command at start of pause (S7-300)

Description

The auxiliary block "PE_Start_RSP" (Start Pause) generates the response to the PE command Start_Pause (Page 5328). The instruction returns the energy-saving mode to which the device has switched (PE_MODE_ID parameter).

If the response differs depending on the length of the pause, you can return this fact in the feedback on the energy-saving mode (PE_Mode_ID = 1 for a brief pause, PE_Mode_ID = 2 for a longer pause, etc.).

Parameter

The following table shows the parameters of the "PE_Start_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 5338)" instruction.
PE_MODE_ID	Input	BYTE	I, Q, M, D, L or constant	PE mode that the process assumes
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 5338)" instruction.

Parameter	Declaration	Data type	Memory area	Description
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 5338)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_End_RSP: Generate answer to command at end of pause (S7-300)

Description

The auxiliary block "PE_End_RSP" generates the response to the PE command End_Pause (Page 5329). The response returned is the time required to switch from the current mode to "Ready_To_Operate" mode.

Parameter

The following table shows the parameters of the "PE_End_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 5338)" instruction.
Time_to_Operate	Input	DWORD	I, Q, M, D, L or constant	Time required to switch from the current mode to "Ready_To_Operate".

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 5338)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 5338)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_List_Modes_RSP: Generate queried energy savings modes as answer (S7-300)

Description

The auxiliary block "PE_List_Modes_RSP" generates the response to the PE command List_Energy_Saving_Modes (Page 5329). The response generated includes the number and the IDs of energy-saving modes supported.

Parameter

The following table shows the parameters of the "PE_List_Modes_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 5338)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Number_of_PE_Mode_IDs	Input	BYTE	I, Q, M, D, L or constant	Number of energy-saving modes supported. Permitted values: 1 to 254
PE_MODE_ID	Input	ANY	I, Q, M, D, L	Points to the area in which the IDs of the energy-saving modes supported (PE_Mode_ID) are stored. Valid range: 1 to 254.
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 5338)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 5338)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_Get_Mode_RSP: Generate queried energy data as answer (S7-300)

Description

The auxiliary block "PE_Get_Mode_RSP" generates the response to the command Get_Mode (Page 5330). The response frame contains the time, performance, or energy data of an energy-saving mode.

Parameter

The following table shows the parameters of the "PE_Get_Mode_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 5338)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
PE_Mode_ID	Input	BYTE	I, Q, M, D, L or constant	ID of the energy-saving mode currently enabled.
PE_Mode_Attributes	Input	BYTE	I, Q, M, D, L or constant	Additional information regarding energy saving mode
Time_min_Pause	Input	DWORD	I, Q, M, D, L or constant	Minimum pause duration for this PE energy-saving mode. This is the sum of the following three parameters: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay
Time_to_Pause	Input	DWORD	I, Q, M, D, L or constant	Time between the edge at the START parameter (see "PE_I_DEV (Page 5338)") and the requested PE energy-saving mode being reached.

Parameter	Declaration	Data type	Memory area	Description
Time_to_Operate	Input	DWORD	I, Q, M, D, L or constant	Maximum switch-on time to "PE_ready_to_operate". The "Time_to_operate" parameter can be used directly for the relevant calculations. The value may be a static maximum, or be calculated dynamically by the PE entity.
Time_min_Lenght_of_stay	Input	DWORD	I, Q, M, D, L or constant	Minimum dwell time of the PE entity in this PE mode.
Time_max_Lenght_of_stay	Input	DWORD	I, Q, M, D, L or constant	Maximum dwell time of the PE entity in this PE mode.
Mode_Power_Consumption	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the current PE mode in [kW].
Energy_Consum_to_Pause	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from "PE_ready_to_operate" to the current PE mode in [kWh].
Energy_Consum_to_operate	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from the current PE mode to "PE_ready_to_operate" in [kWh].
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 5338)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 5338)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_PEM_Status_RSP: Generate PEM status as answer (S7-300)

Description

The auxiliary block "PE_PEM_Status_RSP" generates the response to the command PEM_Status (Page 5332).

Parameter

The following table shows the parameters of the "PE_PEM_Status_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 5338)" instruction.
PE_MODE_ID_ Source	Input	BYTE	I, Q, M, D, L or constant	Source and Destination for PEM_STATUS. Values: • 0x00: PE_POWER_OFF • 0x01 – 0xFE: manufacturer-specific • 0xFF: PE_READY_TO_OPERATE
PE_MODE_ID_ Destination	Input	BYTE	I, Q, M, D, L or constant	
Time_to_Operate[1]	Input	DWORD	I, Q, M, D, L or constant	Maximum switch-on time to "PE_ready_to_operate". "Time_to_operate" can be used directly for the relevant calculations. The value may be a static maximum, or be calculated dynamically by the PE entity.
Remaining_time_ to_destination[1]	Input	DWORD	I, Q, M, D, L or constant	Optional: Time remaining until the requested PE mode. Dynamic value or static maximum value
Mode_Power_ Consumption[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the current PE mode in [kW].
Energy_ Consumption_ to_Destination[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the time until the requested PE mode in [kW].
Energy_ Consumption_ to_operate[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from the current PE mode to "PE_ready_to_operate" in [kWh].
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 5338)" instruction.

Parameter	Declaration	Data type	Memory area	Description
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 5338)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range
[1] If the time period is unlimited, the maximum value "0xFFFFFFFF" can be specified. If the time period is "Zero", "0x00" can be used. [2] If no energy consumption value has been defined, "0.0" can be specified.				

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_Identify_RSP: Generate supported PROFIenergy commands as answer (S7-300)

Description

The auxiliary block "PE_Identify_RSP" generates the response to the command PE_Identify (Page 5334). In the response to the command, specify which PROFIenergy commands are supported. Please note that PE_IDENTIFY is itself a PE command and has to be included in the response.

Parameter

The following table shows the parameters of the "PE_Identify_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 5338)" instruction.
Start_Pause	Input	BOOL	I, Q, M, D, L or constant	One parameter for each of the relevant PROFIenergy commands: • 0: PE command is not supported • 1: PE command is supported
End_Pause	Input	BOOL	I, Q, M, D, L or constant	
Query_Modes	Input	BOOL	I, Q, M, D, L or constant	
PEM_Status	Input	BOOL	I, Q, M, D, L or constant	
PE_Identify	Input	BOOL	I, Q, M, D, L or constant	
Query_Measurement	Input	BOOL	I, Q, M, D, L or constant	
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 5338)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 5338)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_Measurement_List_RSP: Generate list of supported measured values as answer (S7-300)

Description

The auxiliary block "PE_Measurement_List_RSP" generates the response to the command Get_measurement_list (Page 5334). In the response, specify which measured values (Measurement-IDs) are supported.

Parameter

The following table shows the parameters of the "PE_Measurement_List_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 5338)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Count	Input	BYTE	I, Q, M, D, L or constant	Number of measured values supported (measurement IDs)
Measurement_List	Input	ANY	D	Pointer to the array with the Measurement_IDs supported. For information on the structure of the array in accordance with the PROFIenergy profile, see: PI Command "Query_Measurement" - "Get_Measurement_list" (Page 5334)

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 5338)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 5338)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

PE_Measurement_Value_RSP: Generate queried measured values as answer (S7-300)

Description

The auxiliary block "PE_Measurement_Value_RSP" generates the response to the command Get_measurement_values (Page 5337). In the response, return the values of the requested measurements.

Parameter

The following table shows the parameters of the "PE_Measurement_Value_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 5338)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Count	Input	BYTE	I, Q, M, D, L or constant	Number of measured values (Measurement_Values).
Measurement_Values	Input	ANY	D	Pointer to the array with the measured values (Measurement_IDs). For information on the structure of the array in accordance with the PROFIenergy profile, see PI Command "Query_Measurement" - "Get_Measurement_values" (Page 5337)
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 5338)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 5338)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 5338)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

4.2.3.6 Module parameter assignment

Writing and reading data records

Principle

Some modules have a write-only system data area to which your program can transfer data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see following table).

Modules can also have a read-only system data area in which your program can read data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see following table).

Note

There are modules that have both system data areas. These are physically separate areas and the only thing they have in common is their logical structure.

Write-only system data area

The following table shows the structure of the write-only system data area. This table also shows how the permitted size of the individual data records and which instructions can be used to write them.

Data record number	Contents	Size	Restriction	Can be written with instruction
0	Parameter	For S7-300: 2 to 14 Bytes	Can only be written by an S7-400	WR_DPARM (Page 5365) PARM_MOD (Page 5360)
1	Parameter	For S7-300: 2 to 14 Bytes (Data records 0 and 1 together have a total of exactly 16 bytes.)	-	WR_PARM (Page 5364) WR_DPARM (Page 5365) PARM_MOD (Page 5360)
2 to 127	User data	Each ≤ 240 bytes	-	WR_PARM (Page 5364) WR_DPARM (Page 5365) PARM_MOD (Page 5360) WR_REC (Page 5260)
128 to 240	Parameter	Each ≤ 240 bytes	-	WR_PARM (Page 5364) WR_DPARM (Page 5365) PARM_MOD (Page 5360) WR_REC (Page 5260)

Read-only system data area

The following table shows the structure of the read-only system data area. This table also shows the permitted size of the individual data records and which instructions can be used to read them.

Data record number	Contents	Size	Can be read with instruction
0	Module-specific diagnostic data (set as standard for the entire system)	4 bytes	RDSYSST (Page 5444) (SZL_ID00B1H) RD_REC (Page 5256)
1	Channel-specific diagnostic data (incl. data record 0)	• For S7-300: 16 bytes • with S7-400: 4 to 220 Bytes	RDSYSST (Page 5444) (SZL_ID00B2H and 00B3H) RD_REC (Page 5256)
2 to 127	User data	Each ≤ 240 bytes	RD_REC (Page 5256)
128 to 240	Diagnostic data	Each ≤ 240 bytes	RD_REC (Page 5256)

System resources

If you start several asynchronous data record transfers one after the other with only short intervals between them, the allocation of system resources by the operating system ensures that all the jobs are executed and that they do not interfere with each other.

If the limits of the system resources are reached, this is indicated in RET_VAL. You can remedy this temporary error situation by simply repeating the job.

The maximum number of "simultaneously" active jobs of a single instruction type depends on the CPU.

RD_DPAR: Read module data record

Description

You use the instruction to read the data record with the number INDEX of the addressed component from the configured system data. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the destination area RECORD. In this case, the LEN output parameter contains the length of the read data in bytes.

If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR . In this case, the output parameter STATUS contains the error information.

Functional description

The "RD_DPAR" instruction works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling "RD_DPAR" with REQ = 1.

The output parameter BUSY and bytes 2 and 3 of the output parameter STATUS show the status of the job. Bytes 2 and 3 of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

The transfer of the data record is complete when the output parameter BUSY has the value FALSE.

Parameter

The following table shows the parameters of the instruction "RD_DPAR":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Read request
LADDR	Input	WORD	I, Q, M, D, L or constant	Any logical address of the module In bit 15 you indicate whether an input address (bit 15 = 0) or an output address (bit 15 = 1) is present.
INDEX	Input	INT	I, Q, M, D, L or constant	Data record number
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The job is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.
STATUS	Output	DWORD	I, Q, M, D, L	• Call ID (bytes 2 and 3) or error code • Byte 1: B#16#00, if no error. Otherwise function ID from DPV1-PDU: In the case of error for data record reading B#16#DE, in the case of error for data record writing B#16#DF. If no DPV1 protocol element is used: B#16#C0. • Byte 4: Manufacturer-specific error ID expansion
LEN	Output	INT	I, Q, M, D, L	Length of the read data record information
RECORD	InOut	ANY	I, Q, M, D, L	Destination area for the read data record Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter ERROR and STATUS

See also: PARM_MOD: Transfer module data records (Page 5360).

RD_DPARA: Read module data record asynchronously (S7-300)

Description

You use the instruction to read the data record with the number RECNUM of a selected module from the configured system data. The read data record is entered into the destination area spanned by the RECORD parameter.

Functional description

The "RD_DPARA" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling the instruction with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

Parameter

The following table shows the parameters of the instruction "RD_DPARA":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Read request
LADDR	Input	WORD	I, Q, M, D, L or constant	Any address of the module. For an output address, the most significant bit must be set.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error occurred during the transmission, the following two cases are distinguished: • RET_VAL contains the length of the actually read data record in bytes if the destination area is larger than the read data record. • RET_VAL contains "0" if the length of the read data record is equal to the length of the destination area.

Parameters	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The job is not yet complete.
RECORD	Output	ANY	I, Q, M, D, L	Destination area for the data record read. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

See also: PARM_MOD: Transfer module data records (Page 5360).

PARM_MOD: Transfer module data records

Description

You use the instruction "PARM_MOD" to transfer all configured data records of a module to the module. With this instruction, it is irrelevant whether the data records are static or dynamic.

Parameter

The following table shows the parameters of the "PARM_MOD" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ= 1: Write request
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical base address of the module. For a mixed module, the lower of the two addresses must be specified.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Error information

The "real" error information (error codes W#16#8xyz) can be divided into two classes:

- Temporary errors (error codes W#16#80A2 to 80A4, 80Cx):
  This type of error can possibly be eliminated without user action, meaning it is helpful to call the instruction again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#809x, 80A1, 80Bx, 80Dx):
  This type of error does not correct itself. Retries to call the instruction will only be successful after the error has been eliminated.
  Example of a permanent error: Incorrect length of the data record to be transferred (W#16#80B1).

Note

If you transfer data records to a DPV1 slave with the WR_PARM, WR_DPARM or PARM_MOD instruction and if this slave operates in DPV1 mode, the DP master evaluates the error information it has received from this slave as follows:

If the error information lies within the range from W#16#8000 to W#16#80FF or W#16#F000 to W#16#FFFF, the DP master passes the error information to the instruction. If the error information is outside this range, the DP master passes the value W#16#80A2 to the instruction and suspends the slave.

Displays for the "RD_DPARM (Page 5363)", "WR_PARM (Page 5364)", "WR_DPARM (Page 5365)" and "PARM_MOD" instructions

Parameter RET_VAL

Error code (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Specified logical base address invalid: There is no assignment in SDB1/SDB2x or there is no base address.	-
8092	A type other than BYTE is specified in the ANY reference.	Only with S7-400 for RD_DPARM (Page 5363) and WR_PARM (Page 5364)
8093	This instruction is not permitted for the modules selected withLADDR and IOID. (Permitted are S7-300 modules for an S7-300, S7-400 modules for an S7-400, S7-DP modules for an S7-300 and S7-400.)	-

Error code (W#16#...)	Explanation	Restriction
80A1	Negative acknowledgment when sending the data record to the module (the module was removed or became defective during transfer).	1)
80A2	DP protocol error at layer 2, possibly hardware/interface fault in DP slave.	Distributed I/O 1)
80A3	DP protocol error with user interface/user.	Distributed I/O 1)
80A4	Communication on the communication bus disrupted	Error occurred between the CPU and external DP interface module 1)
80B0	Instruction for module type not possible, or module does not recognize the data record.	1)
80B1	The length of the data record to be sent is incorrect. For RD_DPARM (Page 5363): The length of the destination area spanned by RECORD is too short.	-
80B2	The configured slot is not occupied.	1)
80B3	Actual module type does not match the specified module type in SDB1.	1)
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	1)
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	1)
80C3	The required resources (memory, etc.) are currently occupied.	
80C4	Internal temporary error. Job could not be carried out. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	1)
80C5	Distributed I/O not available or deactivated.	Distributed I/O 1)
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O 1)
80D0	No entry for the module in the corresponding SDB.	-
80D1	The data record number is not configured in the corresponding SDB for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-
80D3	The SDB cannot be accessed since it does not exist.	-
80D4	SDB structure error: The SDB internal pointer points to a value outside the SDB.	For S7-300 only
80D5	The data record is static.	With "WR_PARM (Page 5364)" only

Error code (W#16#...)	Explanation	Restriction
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)	-
[1] does not occur in "RD_DPARM"		

See also

Difference between synchronous and asynchronous instructions (Page 4186)

RD_DPARM: Read data record from configured system data (S7-400)

Description

You use instruction to read the data record with the number RECNUM of the addressed module from the configured system data. The read data record is entered into the destination area spanned by the RECORD parameter.

Parameter

The following table shows the parameters of the instruction "RD_DPARM":

Parameters	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical base address of the module. For a mixed module, the lower of the two addresses must be specified.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	Length of the data record read in bytes if the read data record fits in the destination area and no error occurred in the transfer. If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	ANY	I, Q, M, D, L	Destination area for the data record read. Only the BYTE data type is permitted.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

See also: PARM_MOD: Transfer module data records (Page 5360).

WR_PARM: Write module data record

Description

You use the instruction to transfer the data record RECORD to the addressed module. Parameters transferred to the module do not overwrite parameters created during configuration.

Requirements

The data record to be transferred must not be static. The information concerning which data records of a module are static is provided in the module description.

Parameters

The following table shows the parameters of the instruction "WR_PARM":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Write request
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical base address of the module. For a mixed module, the lower of the two addresses must be specified.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number
RECORD	Input	ANY	I, Q, M, D, L	Data record Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RECORD

The data to be transferred are read from the RECORD parameter during the first call. If the transfer of the data record takes longer than the duration of one call, the contents of the RECORD parameter are no longer relevant for the subsequent instruction calls (for the same job).

Parameter RET_VAL

See also: PARM_MOD: Transfer module data records (Page 5360).

Note

For S7-400 only

If the general error W#16#8544 occurs, it only indicates that access to at least one byte of the I/O memory area containing the data record was denied. The data transfer was continued.

See also

Difference between synchronous and asynchronous instructions (Page 4186)

WR_DPARM: Transfer data record

Description

You use the instruction to transfer the data record with the number RECNUM from the configuration data to the addressed module. With this instruction, it is irrelevant whether the data record is static or dynamic.

Parameter

The following table shows the parameters of the instruction "WR_DPARM":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Write request
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, you will have to specify the area identifier of the lower address. If the addresses are identical, you will have to specify B#16#54.

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical base address of the module. For a mixed module, the lower of the two addresses must be specified.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

See also: PARM_MOD: Transfer module data records (Page 5360).

See also

Difference between synchronous and asynchronous instructions (Page 4186)

4.2.3.7 Interrupts

Time-of-day interrupt

Using the instructions SET_TINT, CAN_TINT, ACT_TINT, QRY_TINT

Definition

A time-of-day interrupt causes the time-triggered call of a time-of-day interrupt OB.

Prerequisites for the call

Before a time-of-day interrupt OB can be called by the operating system, the following conditions must be met:

- The time-of-day interrupt OB parameters must be assigned (start date and time, execution)
  - Per configuration, or
  - With the "SET_TINT (Page 5369)" instruction in the user program.
- The time-of-day interrupt OB must be activated
  - Per configuration, or
  - with the "ACT_TINT (Page 5371)" instruction in the user program.

- The time-of-day interrupt OB must not be deselected during configuration.
- The time-of-day interrupt OB must exist in the CPU.
- If you use the "ACT_TINT (Page 5371)" instruction for setting, and if you have specified the execution of the OB as **once only**, the start date and time must not yet have elapsed; for **periodic execution**, the time-of-day interrupt OB will be started at the next expired period (start time + multiple of the period duration).

Note

You can assign parameters to the time-of-day interrupt using the configuration tool and then activate the interrupt in your user program ("ACT_TINT (Page 5371)").

Purpose of the instructions "SET_TINT", "CAN_TINT", "ACT_TINT" and "QRY_TINT"

You use the instructions to

- set time-of-day interrupts ("SET_TINT (Page 5369)")
- delete time-of-day interrupts ("CAN_TINT (Page 5370)")
- enable time-of-day interrupts ("ACT_TINT (Page 5371)")
- query time-of-day interrupts ("QRY_TINT (Page 5371)")

Effects on the time-of-day interrupt

The following table lists a number of different situations and explains the effect they have on a time-of-day interrupt.

If ...	Then ...
a new time-of-day is set (Calling of "SET_TINT (Page 5369)"),	The current time-of-day interrupt is canceled automatically.
The time-of-day interrupt will be cancelled (Calling of "CAN_TINT (Page 5370)"),	The start date and time are cleared. The time-of-day interrupt must then be set again before it can be activated.
The time-of-day interrupt OB does not exist when it is called,	The priority class error is generated automatically, which means that the operating system calls OB 85. If OB 85 does not exist, the CPU changes to STOP.
The clock is synchronized or the time of day set forward,	If the start date/time is skipped because the clock is moved forward: • The operating system then calls OB 80.[1] • Following OB 80, every skipped time-of-day interrupt OB is called (once, regardless of the number of periods that were skipped) provided that it was not manipulated in OB 80.[2] If OB 80 does not exist, the CPU changes to STOP.

If ...	Then ...
The real-time clock is synchronized or the clock is set back	S7-400 CPUs and CPU 318: If the time-of-day interrupt OBs had already been called during the time by which the clock has been set back, they are not called again the second time around. S7-300 CPUs: The affected time-of-day interrupt OBs are all executed.
[1] OB 80 contains encoded start event information, indicating which time-of-day interrupt OBs could not be called due to moving the clock forward. The time of day in the start event information corresponds to the time adjusted forward. [2] The time of day in the start event information of the time-of-day interrupt OB activated later after being skipped corresponds to the start time of the first skipped time-of-day interrupt.	

Response to warm restart or cold restart

During a warm restart or a cold restart, all the time-of-day interrupt settings made in the user program by means of instructions are cleared. In this case the parameters set per configuration are effective.

Execution of the time-of-day interrupt OB

The following table shows the different effects of the "execution" parameter. This parameter must be set per configuration or with the "SET_TINT (Page 5369)" instruction (input parameter PERIOD).

Execution of the time-of-day interrupt OB	Reaction
None (can only be set per configuration)	The time-of-day interrupt OB is not executed even when it exists in the CPU. Parameter reassignment, which means you can set the time-of-day interrupt ("SET_TINT (Page 5369)") in the user program.
Once	The time-of-day interrupt is canceled after the time-of-day interrupt OB has been called. It can then be set and activated again.
Periodic (every minute, hour, day, week, month, year)	If the start date and time have already passed when the interrupt is activated, the time-of-day interrupt OB interrupts the cyclic program at the next possible point "start date/time + multiple of the selected period". In extremely rare situations, processing of the time-of-day interrupt OB may not yet be completed when it is called again. Result: • Time error, (the operating system calls OB 80; if OB 80 does not exist, the CPU changes to STOP). • The time-of-day interrupt OB is executed later.

SET_TINT: Set time-of-day interrupt

Description

With the instruction, you can set the start date and time of the time-of-day interrupt organization blocks. The seconds and milliseconds of the specified start time are ignored and set to "0".

Parameter

The following table shows the parameters of the instruction "SET_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB started at the time SDT + multiple of PERIOD (OB 10 to OB 17).
SDT	Input	DT	D, L	Start date and time: the seconds and milliseconds of the specified start time are ignored and set to 0. If you want to set a monthly start of a time-of-day interrupt OB, you can only use the days 1, 2, ... 28 as a start date.
PERIOD	Input	WORD	I, Q, M, D, L or constant	Periods from start point SDT onwards: • W#16#0000 = once • W#16#0201 = once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
8091	Incorrect parameter SDT
8092	Incorrect parameter PERIOD

Error code (W#16#...)	Description
80A1	The set start time is in the past. (This error code occurs only when PERIOD = W#16#0000.)
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

CAN_TINT: Cancel time-of-day interrupt

Description

You use the instruction to set the start date and time of the time-of-day interrupt organization block. This disables the time-of-day interrupt and the organization block is no longer called.

If you would you like to use the time-of-day interrupt again, you must first set the start time ("SET_TINT (Page 5369)" instruction) and then enable the time-of-day interrupt ("ACT_TINT (Page 5371)" instruction).

Parameter

The following table shows the parameters of the instruction "CAN_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB, in which the start date and time will be canceled (OB 10 to OB 17).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
80A0	No start date/time specified for the time-of-day interrupt OB
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

ACT_TINT: Enable time-of-day interrupt

Description

You use the instruction to activate a time-of-day interrupt organization block.

Parameter

The following table shows the parameters of the instruction "ACT_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB to be activated (OB 10 to OB 17).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
80A0	Start date/time-of day not set for the respective time-of-day interrupt OB.
80A1	The activated time is in the past. This error only occurs if execution = once is selected.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

QRY_TINT: Query status of time-of-day interrupt

Description

You use the instruction to display the status of a time-of-day interrupt organization block at the output parameter STATUS .

Parameter

The following table shows the parameters of the instruction "QRY_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB that will be queried for status (OB 10 to OB 17).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-of-day interrupt; see following table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter STATUS

Note

If RET_VAL contains a value different than zero, you may not evaluate STATUS.

Bit	Value	Meaning
0	0	In Run.
	1	During startup.
1	0	The time-of-day interrupt is enabled.
	1	The time-of-time interrupt is disabled by "DIS_IRT (Page 5388)".
2	0	Time-of-day interrupt is not activated or has elapsed.
	1	The time-of-day interrupt is activated.
4	0	An OB with an OB number as specified at OB_NR parameter does not exist.
	1	An OB with an OB number as specified at OB_NR parameter does exist.
Other		Always "0"

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Time-delay interrupt

Using time-delay interrupts

Definition

After you have called the "SRT_DINT (Page 5374)" instruction, the operating system generates an interrupt after the specified delay time has elapsed, in other words, the assigned time-delay interrupt OB is called.

Prerequisites for the call

Before a time-delay interrupt can be called by the operating system, the following conditions must be met:

- The time-delay interrupt OB must be started by the "SRT_DINT (Page 5374)" instruction.
- The time-delay interrupt OB must not be deselected during configuration.
- The time-delay interrupt OB must exist in the CPU.

Purpose of the instructions "SRT_DINT", "CAN_DINT" and "QRY_DINT"

You use the instructions to

- Start time-delay interrupts ("SRT_DINT (Page 5374)")
- Cancel time-delay interrupts ("CAN_DINT (Page 5375)")
- Query time-delay interrupts ("QRY_DINT (Page 5376)")

Effects on the time-delay interrupt

The following table lists a number of different situations and explains the effect they have on a time-delay interrupt.

If ...	and ...	Then ...
A time-delay interrupt is started (by calling "SRT_DINT (Page 5374)")	The time-delay interrupt has already started,	The delay time is overwritten; the time-delay interrupt is started again.
	The time-delay interrupt OB does not exist at the time of the call,	The operating system generates a priority class error (calls OB 85). If OB 85 does not exist, the CPU changes to STOP.

If ...	and ...	Then ...
	The interrupt is started in a startup OB and the delay time elapses before the CPU changes to RUN,	The call of the time-delay interrupt OB is delayed until the CPU is in RUN mode.
The delay time has elapsed,	A previously started time-delay interrupt OB is still being executed,	The operating system generates a time error (calls OB 80). If OB 80 does not exist, the CPU changes to STOP.

Response to warm restart and cold restart

During a warm restart or a cold restart, all the time-delay interrupt settings made in the user program by means of instructions are cleared.

Starting in a startup OB

A time-delay interrupt can be started in a startup OB. Two conditions must be satisfied to call the time-delay OB:

- The delay time must have elapsed.
- The CPU must be in the RUN mode.

If the delay time has elapsed and the CPU is not yet in the RUN mode, the time-delay interrupt OB call is delayed until the CPU is in RUN mode. The time-delay interrupt OB is then called before the first instruction in OB Main [OB 1] is executed.

SRT_DINT: Start time-delay interrupt

Description

You use the instruction to start a time-delay interrupt that calls a time-delay interrupt OB once the assigned delay time has elapsed (DTIME parameter).

With the SIGN parameter, you can enter an identifier that identifies the start of the time-delay interrupt. The values of DTIME and SIGN appear again in the start event information of the specified OB when it is executed.

The maximum time between the "SRT_DINT" instruction call and the start of the time-delay interrupt OB is **one millisecond** less than the assigned time, provided that no interruption events delay the call.

Parameter

The following table shows the parameters of the instruction "SRT_DINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB to be started after a time delay (OB 20 to OB 23).
DTIME	Input	TIME	I, Q, M, D, L or constant	Time delay value (1 to 60000 ms) You can realize longer times, for example, by using a counter in a time-delay interrupt OB.
SIGN	Input	WORD	I, Q, M, D, L or constant	Identifier that appears in the start event information of the OB when the time-delay interrupt OB is called.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
8091	Incorrect parameter DTIME
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

CAN_DINT: Cancel time-delay interrupt

Description

You use the instruction to cancel a started time-delay interrupt (see "SRT_DINT (Page 5374)"). The time-delay interrupt OB is not called in this case.

Parameters

The following table shows the parameters of the instruction "CAN_DINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB to be canceled (OB 20 to OB 23).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
80A0	Time-delay interrupt has not started.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

QRY_DINT: Query time-delay interrupt status

Description

You use the instruction to query the status of a time-delay interrupt. Time-delay interrupts are managed by organization blocks OB 20 to OB 23.

Parameter

The following table shows the parameters of the instruction "QRY_DINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB that will be queried for status (OB 20 to OB 23).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-delay interrupt, see following table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter STATUS

Note

If RET_VAL contains a value different than zero, you may not evaluate STATUS.

Bit	Value	Meaning
0	0	In Run.
	1	During startup.
1	0	The time-delay interrupt is enabled.
	1	The time-delay interrupt is disabled by "DIS_IRT (Page 5388)".
2	0	Time-delay interrupt is not activated or has elapsed.
	1	The time-delay interrupt is activated.
4	0	An OB with an OB number as specified at OB_NR parameter does not exist.
	1	An OB with an OB number as specified at OB_NR parameter is loaded.
Other		Always "0"

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Synchronous error events

Mask synchronous error events

Introduction

Synchronous errors are programming and access errors. Such errors occur as a result of programming with incorrect operand areas or numbers, or incorrect addresses. **Masking** these synchronous errors means the following:

- Masked synchronous errors do not trigger an error OB call and do not lead to a programmed alternative reaction.
- The CPU "records" the masked errors that have occurred in an error status register.

Masking is carried out by calling the "MSK_FLT (Page 5384)" instruction.

Unmasking errors means canceling a previously set mask and clearing the corresponding bit in the event status register of the current priority class. Masking is canceled as follows:

- By calling the "DMSK_FLT (Page 5385)" instruction.
- When the current priority class has been completed (for S7-400 only).

If an error event occurs after it has been unmasked, then the operating system will start the associated error OB. You program OB 121 for the reaction to programming errors and OB 122 for the reaction to access errors.

You can use the "READ_ERR (Page 5385)" instruction to read the masked errors that have occurred.

Note

With the S7-300 (except CPU 318), regardless of whether an error is masked or unmasked, the error is entered in the diagnostic buffer and the group error LED of the CPU is illuminated.

Handling errors in general

If programming and access errors occur in a user program, you can react to them in different ways:

- You can program an error OB that is called by the operating system when the corresponding error occurs.
- You can disable the error OB call individually for each priority class. In this case, the CPU does not change to STOP when an error of this type occurs in the particular priority class. The CPU enters the error in an error register. From this entry, however, you cannot recognize when or how often the error occurred.

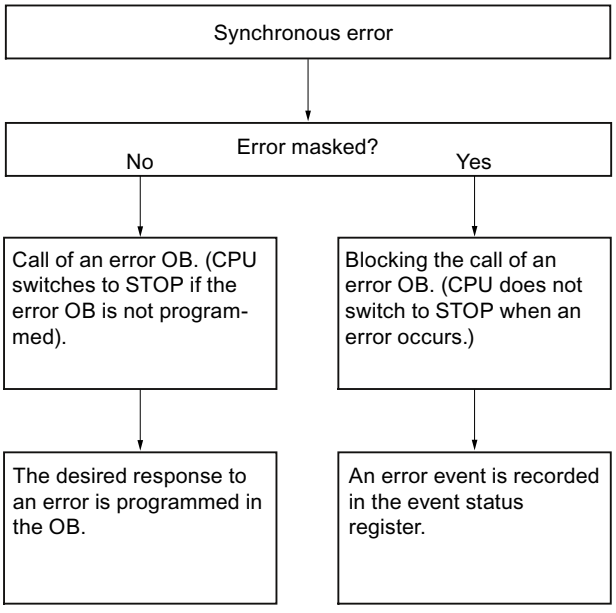

Error filter

Synchronous errors are assigned to a particular bit pattern known as the **error filter**. You can find this error filter in the input and output parameters of the "MSK_FLT (Page 5384)", "DMSK_FLT (Page 5385)", and "READ_ERR (Page 5385)" instructions.

Synchronous errors are divided into programming and access errors that you can mask using two error filters. The error filters are illustrated in the following figures.

Programming error filter

The following figure shows the bit pattern of the error filter for programming errors. The error filter for the programming errors is available in the "PRGFLT_..." parameters (see below "Programming errors, Low-Word" or "Programming errors, High-Word").

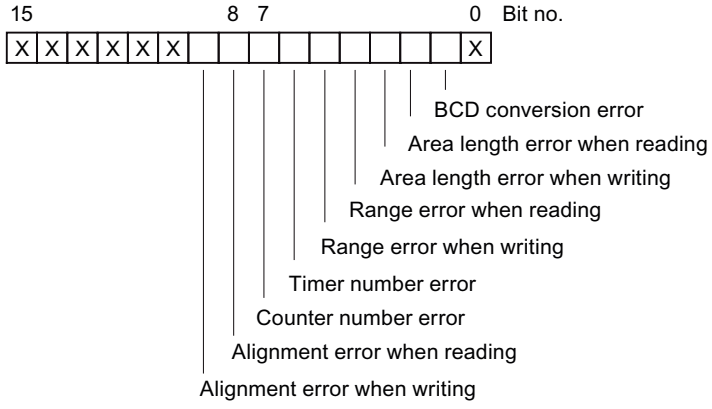

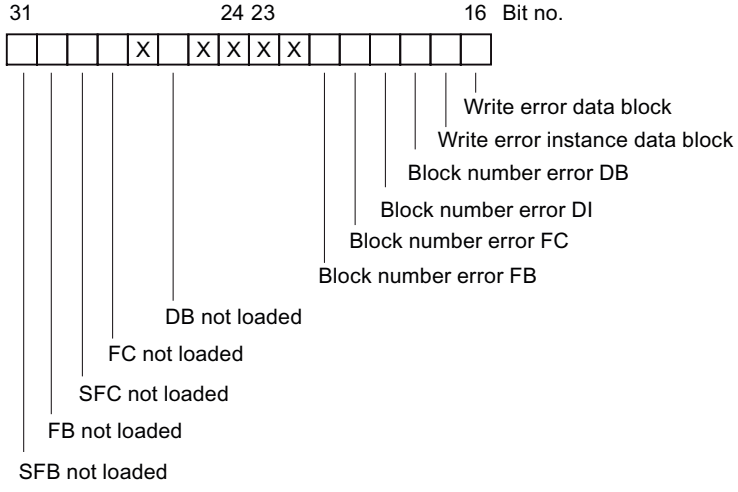

Legend: X Not relevant

Non-relevant bits

In the figure above, **x** means ...

• ... Input parameters	For "MSK_FLT (Page 5384)", "DMSK_FLT (Page 5385)", "READ_ERR (Page 5385)"	= **"0"**
• ... Output parameters	For "MSK_FLT (Page 5384)", "DMSK_FLT (Page 5385)"	= **"1"** for S7-300 = **"0"** for S7-400
	For "READ_ERR (Page 5385)"	= **"0"**

Access error filter for all CPUs

The following figure shows you the bit pattern of the error filter for access errors. The error mask for access errors is located in the parameters ACCFLT_...

An explanation of the access errors can be found in the following tables (see below).

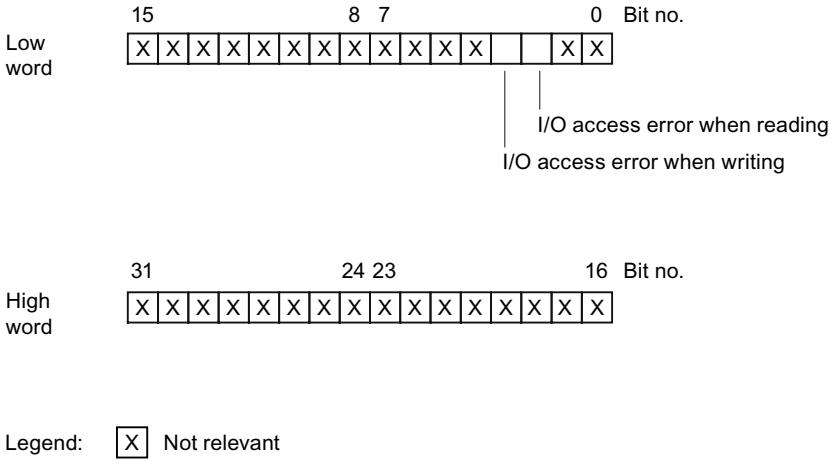

Example

The following figure shows the low word of the error filter for access errors with all masked errors for all CPUs.

- As input parameters for "MSK_FLT (Page 5384)"
- As output parameters for "MSK_FLT (Page 5384)"

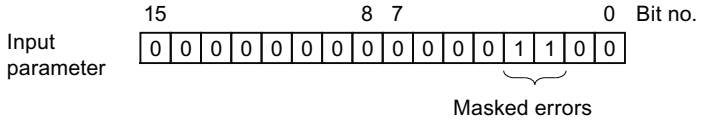

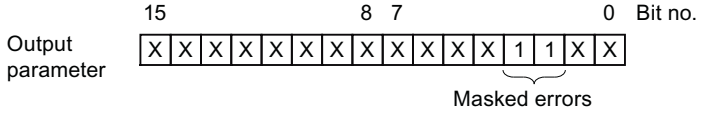

Legend:
- X Not relevant
- 0 Not masked
- 1 Masked

Programming error low word

The following table lists the errors assigned to the low word of the error filter for programming errors. The possible causes of the errors are also listed.

Possible causes of programming errors, low word

Error	Event ID (W#16#...)	Error occurs ...
BCD conversion error	2521	... when the value to be converted is not a BCD number (for example, 5E8).
Area length error when reading	2522	... when an addressed operand is being used that is not completely within the possible operand area. Example: MW 320 should be read although the memory area is only 256 bytes long.
Area length error when writing	2523	... when an addressed operand is being used that is not completely within the possible operand area. Example: A value should be written to MW 320 although the memory area is only 256 bytes long.

Error	Event ID (W#16#...)	Error occurs ...
Range error when reading	2524	... when an incorrect area identifier is specified for the operand when using indirect, area-overlapping addressing. Example: Correct: LAR1 P#E 12.0 L W[AR1, P#0.0] Incorrect: LAR1 P#12.0 L W[AR1, P#0.0] The range error is reported during this operation.
Range error when writing	2525	... when an incorrect area identifier is specified for the operand when using indirect, area-overlapping addressing. Example: Correct: LAR1 P#E 12.0 T W[AR1, P#0.0] Incorrect: LAR1 P#12.0 T W[AR1, P#0.0] The range error is reported during this operation.
Timer number error	2526	... when a non-existent timer is accessed. Example: SP T [MW 0] where MW 0 = 129; timer 129 should be started although there are only 128 timers available.
Counter number error	2527	... when a non-existent counter is accessed. Example: CU C [MW 0] where MW 0 = 600; counter 600 must be accessed although there are only 512 counters available (CPU 416-1).
Alignment error when reading	2528	... when a byte, word, or double word operand is addressed with a bit address ≠ 0. Example: Correct: LAR1 P#M12.0 L B[AR1, P#0.0] Incorrect: LAR1 P#M12.4 L B[AR1, P#0.0]
Alignment error when writing	2529	... when a byte, word, or double word operand is addressed with a bit address ≠ 0. Example: Correct: LAR1 P#M12.0 T B[AR1, P#0.0] Incorrect: LAR1 P#M12.4 T B[AR1, P#0.0]

Programming error high word

The following table lists the errors assigned to the high word of the error filter for programming errors. The possible causes of the errors are also listed.

Possible causes of programming errors, high word

Error	Event ID (W#16#...)	Error occurs ...
Write error data block	2530	... when the data block to be written to is read-only.
Write error instance data block	2531	... when the instance data block to be written to is read-only.
Block number error DB	2532	... when a data block must be opened whose number is higher than the highest permitted number.
Block number error DI	2533	... when an instance data block must be opened whose number is higher than the highest permitted number.
Block number error FC	2534	... when a function is called whose number is higher than the highest permitted number.
Block number error FB	2535	... when a function block is called whose number is higher than the highest permitted number.
DB not loaded	253A	... when the data block to be opened is not loaded.
Instruction not loaded	253C to 253F	... when the called instruction is not loaded.

Access error

The following table lists the errors assigned to the error filter for access errors for all CPUs. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Error occurs ...
I/O access error when reading	2942	... when a signal module is not assigned to the address in the I/O area. Or ... when access to this I/O area is not acknowledged within the specified module monitoring time (timeout).
I/O access error when writing	2943	... when a signal module is not assigned to the address in the I/O area. Or ... when access to this I/O area is not acknowledged within the specified module monitoring time (timeout).

MSK_FLT: Mask synchronous error events

Description

You use the instruction to control the reaction of the CPU to synchronous errors. To do this, you mask the respective synchronous errors (for error filters see Mask synchronous error events (Page 5377)). When you call "MSK_FLT", you mask the synchronous errors in the current priority class.

If you set individual bits of the synchronous error filter to "1" in the input parameters, other bits that were set previously retain their value of "1", You obtain new error filters that you can read out using the output parameters. The synchronous errors you have masked do not call an OB but are simply entered in an error register. You can read the error register with the "READ_ERR (Page 5385)" instruction.

Parameter

The following table shows the parameters of the instruction "MSK_FLT":

Parameters	Declaration	Data type	Memory area	Description
PRGFLT_SET_MASK	Input	DWORD	I, Q, M, D, L or constant	Programming error to be masked
ACCFLT_SET_MASK	Input	DWORD	I, Q, M, D, L or constant	Access error to be masked
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_MASKED	Output	DWORD	I, Q, M, D, L	Masked programming errors
ACCFLT_MASKED	Output	DWORD	I, Q, M, D, L	Masked access errors

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	None of the errors were already masked.
0001	At least one of the errors was already masked. Nevertheless the other errors will be masked.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

DMSK_FLT: Unmask synchronous error events

Description

You use the instruction to unmask the errors masked with "MSK_FLT (Page 5384)". To do this, you must set the corresponding bits of the error filter to "1" in the input parameters. With the "DMSK_FLT" call, you unmask the corresponding synchronous errors of the current priority class. At the same time, the queried entries are cleared in the error register. You can read out the new error filters using the output parameters.

Parameters

The following table shows the parameters of the instruction "DMSK_FLT":

Parameters	Declaration	Data type	Memory area	Description
PRGFLT_ RESET_MASK	Input	DWORD	I, Q, M, D, L or constant	Programming errors to be unmasked
ACCFLT_ RESET_MASK	Input	DWORD	I, Q, M, D, L or constant	Access errors to be unmasked
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_ MASKED	Output	DWORD	I, Q, M, D, L	Still masked programming errors
ACCFLT_ MASKED	Output	DWORD	I, Q, M, D, L	Still masked access errors

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	All specified errors were unmasked.
0001	At least one of the errors was not masked. Nevertheless the other errors will be unmasked.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

READ_ERR: Read out event status register

Description

You use the instruction to read the error register. The structure of the error register corresponds to that of the programming and access error filters, which you can program as input parameters with "MSK_FLT (Page 5384)" and "DMSK_FLT (Page 5385)".

In the input parameters, you enter the synchronous errors you want to read from the error register. When you call "READ_ERR", you read the required entries from the error register and at the same time clear these entries.

The error register contains information that tells you which of the masked synchronous errors in the current priority class occurred at least once. If a bit is set, this means that the corresponding masked synchronous error occurred at least once.

Parameter

The following table shows the parameters of the instruction "READ_ERR":

Parameters	Declaration	Data type	Memory area	Description
PRGFLT_QUERY	Input	DWORD	I, Q, M, D, L or constant	Query programming error
ACCFLT_QUERY	Input	DWORD	I, Q, M, D, L or constant	Query access error
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_CLR	Output	DWORD	I, Q, M, D, L	Programming errors that occurred
ACCFLT_CLR	Output	DWORD	I, Q, M, D, L	Access errors that occurred

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	All queried errors are masked.
0001	At least one of the queried errors is not masked.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Asynchronous error event

Delaying and disabling interrupt and asynchronous error events

Purpose of the instructions "DIS_IRT", "EN_IRT", "DIS_AIRT", "EN_AIRT"

You use these instructions to

- Disable processing of interrupts and asynchronous error events with "DIS_IRT (Page 5388)" for all subsequent CPU cycles.
- Delay higher priority classes with "DIS_AIRT (Page 5388)" until the end of the OB.
- To re-enable processing of interrupts and asynchronous error events with "EN_IRT (Page 5390)" or "EN_AIRT (Page 5392)".

You program the processing of interrupts and asynchronous error events in the user program. You must also program the corresponding OBs.

Purpose of the instructions "DIS_AIRT" and "EN_AIRT"

Delaying higher priority interrupts and asynchronous error events with the "DIS_AIRT (Page 5391)" instruction and then enabling them again with the "EN_AIRT (Page 5392)" instruction has the following advantage:

The number of interrupt delays is counted by the CPU. If you have programmed a delay of interrupts and asynchronous error events, the delay cannot be canceled by standard FC calls if the interrupts and asynchronous error events are also disabled and then enabled again in the standard FCs themselves.

Interrupt classes

The interrupts are divided into various classes. The following table lists all the interrupt classes and the corresponding OBs.

Interrupt class	OB
Time-of-day interrupts	OB 10 to OB 17
Time-delay interrupts	OB 20 to OB 23
Cyclic interrupts	OB 30 to OB 38
Process interrupts	OB 40 to OB 47
Interrupts for DPV1	OB 55 to OB 57
Multicomputing interrupt	OB 60
Redundancy error interrupts	OB 70, OB 72
Asynchronous error interrupts	OB 80 to OB 87 (se below)
Synchronous error interrupts	OB 121, OB 122 (You mask or unmask the execution of synchronous error interrupts with the "MSK_FLT (Page 5384)", "DMSK_FLT (Page 5385)" and "READ_ERR (Page 5385)" instructions)

Asynchronous error events

The following table lists all the asynchronous error events to which you can react with an OB call in the user program.

Asynchronous error events	OB
Time error (for example, cycle time exceeded)	OB 80
Power supply error (for example, battery fault)	OB 81
Diagnostics interrupt (for example, defective fuse on a signal module)	OB 82
Insert/remove module interrupt	OB 83
CPU hardware fault (for example, interface error)	OB 84
Program execution error	OB 85
Rack failure	OB 86
Communication error	OB 87

DIS_IRT: Disable interrupt event

Description

With the instruction you disable the processing of new interrupts and asynchronous error evetns. Disabling means that if an interrupting event occurs, the operating system of the CPU reacts as follows:

- It **neither** calls an interrupt OB nor an asynchronous error OB,
- **nor** does it trigger the normal reaction if an interrupt OB or asynchronous error OB is not programmed.

If you disable interrupts and asynchronous error events, this remains in effect for all priority classes. The disable can only be canceled with the "EN_IRT (Page 5390)" instruction or by a warm or a cold restart.

Whether the operating system writes interrupts and asynchronous error events to the diagnostics buffer when they occur depends on the input parameter setting you select for MODE.

Note

Remember that when you program the "DIS_IRT" instruction, all interrupts that occur will be discarded.

Parameter

The following table shows the parameters of the instruction "DIS_IRT":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Specifies which interrupts and asynchro-nous errors are disabled.
OB_NR	Input	INT	I, Q, M, D, L or constant	OB number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter MODE

MODE (B#16#...)	Meaning
00	All newly occurring interrupts and asynchronous error events are disabled. (Synchronous errors are not disabled.) Assign the value "0" to the OB_NR parameter. Entries continue to be made in the diagnostics buffer.
01	All newly occurring events belonging to a specified interrupt class are disabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Process interrupts: 40 • Interrupts for DPV1: 50 • Multicomputing interrupt: 60 • Redundancy error interrupts: 70 • Asynchronous error interrupts: 80 Entries continue to be made in the diagnostics buffer.
02	All new occurrences of a specified interrupt are disabled. You designate the interrupt using the OB number. Entries continue to be made in the diagnostics buffer.
80	All newly occurring interrupts and asynchronous error events are disabled and are no longer entered in the diagnostics buffer. Assign the value "0" to the OB_NR parameter. The operating system enters event W#16#5380 in the diagnostics buffer.
81	All newly occurring events belonging to a specified interrupt class are disabled and are no longer entered in the diagnostics buffer. The operating system enters event W#16#5380 in the diagnostics buffer.
82	All newly occurring events belonging to a specified interrupt are disabled and are no longer entered in the diagnostics buffer. The operating system enters event W#16#5380 in the diagnostics buffer.

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	The OB_NR input parameter contains an illegal value.
8091	The MODE input parameter contains an illegal value.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

EN_IRT: Enable interrupt event

Description

You use the instruction to enable the processing of new interrupts and asynchronous error events that you have previously disabled with the "DIS_IRT (Page 5388)" instruction. This means that if an interrupting event occurs, the operating system of the CPU reacts in one of the following ways:

- It calls an interrupt OB or asynchronous error OB.
  Or
- It triggers the standard reaction if the interrupt OB or asynchronous error OB is not programmed.

Parameters

The following table shows the parameters of the instruction "EN_IRT":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Specifies which interrupts and asynchronous error events will be enabled.
OB_NR	Input	INT	I, Q, M, D, L or constant	OB number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter MODE

MODE	Meaning
0	All newly occurring interrupts and asynchronous error events are enabled.
1	All newly occurring events belonging to a specified interrupt class are enabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Process interrupts: 40 • Interrupts for DPV1: 50 • Multicomputing interrupt: 60 • Redundancy error interrupts: 70 • Asynchronous error interrupts: 80
2	All newly occurring events of a specified interrupt are enabled. You designate the interrupt using the OB number.

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	The OB_NR input parameter contains an illegal value.
8091	The MODE input parameter contains an illegal value.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

DIS_AIRT: Delay execution of higher priority interrupts and asynchronous error events

Description

With the instruction you delay the processing of interrupt OBs and asynchronous OBs with priority higher than that of the current OB. You can call the "DIS_AIRT" instruction more than once in an OB. The instruction calls are counted by the operating system. The processing delay is in force until you use the "EN_AIRT (Page 5392)" instruction to cancel each delayed processing of interrupt OBs and asynchronous error OBs with "DIS_AIRT", or the current OB has been executed.

The pending interrupts or asynchronous error events are processed as soon as the processing delay has been canceled with the "EN_AIRT (Page 5392)" instruction or the current OB processing is complete.

Parameter

The following table shows the parameters of the instruction "DIS_AIRT":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of delays (= number of DIS_AIRT calls)

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

The following table shows the return value of "DIS_AIRT" that is output with the RET_VAL parameter.

Return value	Description
n	"n" shows the number of processing delays, which is the number of "DIS_AIRT calls, after the instruction is complete (interrupt processing is only enabled again when n = 0; see "EN_AIRT (Page 5392)").

EN_AIRT: Enable execution of higher priority interrupts and asynchronous error events

Description

With the instruction you re-release the execution of high-priority interrupt events or asynchronous error events delayed with "DIS_AIRT (Page 5391)". Each individual processing delay must be canceled with the "EN_AIRT" instruction.

Example

If, for example, you have delayed interrupts five times with five "DIS_AIRT (Page 5391)" calls, you must cancel each of the interrupt delays using five "EN_AIRT" calls.

Parameter

The following table shows the parameters of the instruction "EN_AIRT":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of delays still programmed on completion of "EN_AIRT" or error message.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

How you evaluate the error information of the RET_VAL parameter is explained in the chapter Evaluating errors with output parameter RET_VAL (Page 162). This chapter also contains the general error information for the instructions. The following table contains the return value or error information specific to "EN_AIRT" that can be output with the RET_VAL parameter.

Return value and error information	Description
n	"n" shows the number of processing delays, which is the number of "EN_AIRT calls, after the instruction is complete (interrupt processing is only enabled again when n = 0).
W#16#8080	"EN_AIRT" was called even though interrupt processing has already been enabled.

Multicomputing interrupt

MP_ALM: Multicomputing interrupt (S7-400)

Description

Calling the instruction during multiprocessing will trigger the multicomputing interrupt. This leads to a synchronized start of OB 60 on all CPUs involved. In single processor mode and when operating with a segmented rack, OB 60 is only started on the CPU on which you called MP_ALM.

Use the input parameter JOB to identify the cause of the multicomputing interrupt. This job identifier is transferred to all associated CPUs, where you can evaluate it in the multicomputing interrupt OB (OB 60).

You can call the instruction at any point in your program. Since the call would be pointless in any mode other than RUN, if it is called in STARTUP mode, the multicomputing interrupt is suppressed. The function value informs you of this.

Parameters

The following table shows the parameters of the instruction "MP_ALM":

Parameters	Declaration	Data type	Memory area	Description
JOB	Input	BYTE	I, Q, M, D, L or constant	Job identifier Possible values: 1 to 15
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	The JOB input parameter contains an illegal value.
80A0	Execution of OB 60 following the last multicomputing interrupt is not completed either on the local CPU or on another CPU.
80A1	Incorrect mode (STARTUP instead of RUN).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

4.2.3.8 Alarms

Structure of diagnostic data

Data record 0 and 1 of the system data

The diagnostic data of a module are located in the data records 0 and 1 of the system data area, see Writing and reading data records (Page 5355):

- Data record 0 contains 4 bytes of diagnostic data that describe the current status of a signal module.
- Data record 1 contains
  - The 4 bytes of diagnostic data, also located in data record 0 and
  - The diagnostic data specific to the module.

Structure and contents of the diagnostic data:

This section below describes the structure and contents of the individual bytes of the diagnostic data. The general principle: Whenever an error occurs, the corresponding bit is set to "1."

Byte	Bit	Meaning	Remark	Data record
0	0	Module is defective		0 and 1
	1	Internal error		
	2	External error		
	3	Channel fault		
	4	External voltage failed		
	5	Front panel connector not plugged in		
	6	No parameter assignment		
	7	Wrong parameters in the module		
1	0 to 3	Module class	0101: Analog module 0000: CPU 1000: Function module 1100: CP 1111: Digital module 0011: DP standard slave 1011: I-slave 0100: IM	0 and 1
	4	Channel information exists		
	5	User information exists		
	6	Diagnostic interrupt from substitute		
	7	Maintenance requirement (PROFINET IO only)		

Byte	Bit	Meaning	Remark	Data record
2	0	Memory card missing or wrong		0 and 1
	1	Communication problem		
	2	Operating mode	0: RUN 1: STOP	
	3	Cycle monitoring tripped		
	4	Internal power supply failed		
	5	Battery exhausted		
	6	Entire backup failed		
	7	Maintenance request (PROFINET IO only)		
3	0	Expansion rack failure		0 and 1
	1	Processor failure		
	2	EPROM fault		
	3	RAM fault		
	4	ADC/DAC error		
	5	Fuse tripped		
	6	Hardware interrupt lost		
	7	Reserved		
4	0 to 6	Channel type	B#16#70: Digital input B#16#72: Digital output B#16#71: Analog input B#16#73: Analog output B#16#74: FM-POS B#16#75: FM-REG B#16#76: FM-COUNT B#16#77: FM-TECHNO B#16#78: FM-NCU B#16#79: To B#16#7D: Reserved B#16#7E: US300 B#16#7F: Reserved	1
	7	Additional channel type exists?	0: No 1: Yes	
5	0 to 7	Number of diagnostic bits output per channel by a module.	The number of diagnostic bits per channel is rounded up to byte boundaries	1
6	0 to 7	Number of same type channels on a module	If different channel types exist on a module, the structure is repeated in data record 1 from byte 4 onwards for each channel type.	1

Byte	Bit	Meaning	Remark	Data record
7	0	Channel error channel 0/ channel group 0	First byte of the channel error vector (the length of the channel error vector depends on the number of channels and is rounded up to a byte boundary).	1
	1	Channel error channel 1/ channel group 1		
	2	Channel error channel 2/ channel group 2		
	3	Channel error channel 3/ channel group 3		
	4	Channel error channel 4/ channel group 4		
	5	Channel error channel 5/ channel group 5		
	6	Channel error channel 6/ channel group 6		
	7	Channel error channel 7/ channel group 7		
...	-	Channel-specific errors (see Structure of channel-specific diagnostic data (Page 5396))		1

Structure of channel-specific diagnostic data

Channel-specific errors

Starting at the byte immediately following the channel error vector, the channel-specific errors are indicated for each channel of the module. The tables below show the structure of channel-specific diagnostics data for the different channel types. The bits have the following meaning:

- 1 = Error
- 0 = No error

Analog input channel

Diagnostics byte for an analog input channel:

Bit	Meaning	Remark
0	Configuration/parameter assignment error	Can be signaled with WR_USMSG (Page 5428) and EVENTN= W#16#8x50
1	Common mode error	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x51
2	P short circuit	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x52
3	M short circuit	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x53
4	Wire break	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x54
5	Reference channel error	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x55
6	Measuring range undershoot	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x56
7	Measuring range exceeded	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x57

Analog output channel

Diagnostics byte for an analog output channel

Bit	Meaning	Remark
0	Configuration/parameter assignment error	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x60
1	Common mode error	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x61
2	P short circuit	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x62
3	M short circuit	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x63
4	Wire break	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x64
5	0	Reserved
6	No load voltage	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x66
7	0	Reserved

Digital input channel

diagnostics byte for a digital input channel:

Bit	Meaning	Remark
0	Configuration/parameter assignment error	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x70
1	Ground error	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x71
2	I/O short circuit (sensor)	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x72
3	M short circuit	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x73
4	Wire break	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x74
5	No sensor power supply	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x75
6	0	Reserved
7	0	Reserved

Digital output channel

Diagnostics byte for a digital output channel:

Bit	Meaning	Remark
0	Configuration/parameter assignment error	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x80
1	Ground error	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x81

Bit	Meaning	Remark
2	P short circuit	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x82
3	M short circuit	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x83
4	Wire break	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x84
5	Fuse tripped	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x85
6	No load voltage	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x86
7	Overtemperature	Can be signaled with WR_USMSG (Page 5428) and EVENTN = W#16#8x87

Instructions for generating PLC alarms with instance DB (S7-400)

Introduction to generating PLC alarms with instructions (S7-400)

Instructions for generating block-related alarms

You generate a PLC alarm by calling one of the following instructions in your program:

- "NOTIFY (Page 5409)"
- "NOTIFY_8P (Page 5402)"
- "ALARM (Page 5412)"
- "ALARM_8 (Page 5404)"
- "ALARM_8P (Page 5406)"

These instructions have the following properties:

- With "NOTIFY (Page 5409)" and "NOTIFY_8P (Page 5402)", all 0 -> 1 or 1 -> 0 signal changes detected during the call will cause an alarm to be sent.
- Also for "ALARM (Page 5412)", "ALARM_8 (Page 5404)" and "ALARM_8P (Page 5406)", every signal change detected for a call causes an alarm to be sent, provided acknowledgment-triggered reporting is inactive.
  If, on the other hand, you have activated the acknowledgment-triggered reporting, then not every detected signal change will cause an alarm to be sent (see below for more information). In this case, a sent alarm is not displayed in the alarm display.

- After the block processing is complete, all of the associated values (SD_i inputs) are detected and the alarm is assigned (refer to "Send and receive parameters" in Common parameters of instructions for S7 communication (Page 6135).)
  The following applies with regard to the consistency of the associated values compared to higher priority classes: Every associated value SD_i is inherently consistent.
- With status parameters DONE, ERROR and STATUS, you monitor the processing status of the block (refer to "Status parameters" in Common parameters of instructions for S7 communication (Page 6135)).

Note

The parameters ID and EV_ID are only evaluated at the initial call of the block (the actual parameters or the defined values of the instance).

Logging on display devices

Before the instructions for generating PLC alarms can send an alarm when a signal change is detected, at least one display device must be logged on for PLC alarms. STATUS parameter = "1" if no login exists.

Signal change detection

One alarm memory with two memory locations is available for each instance of an alarm instruction.
This alarm memory is initially empty. As soon as the instruction detects a signal change at the SIG input or at one of the inputs SIG_1, ... SIG_8, it is entered in the first memory location. It remains occupied until the associated alarm is sent.
The next detected signal at the SIG or at one of the SIG_1, ... SIG_8 inputs is then entered in the second memory block. If the first memory block is still occupied and additional signals follow, then the second memory block in the alarm memory is always overwritten.
The loss of this alarm is indicated via the output parameters ERROR and STATUS (ERROR=0, STATUS=11). In addition the logged on display devices get an alarm in this regard with the next alarm that can be sent.
If he first memory block is free then the second memory block will be transferred into the first. Thus, the second memory block is cleared again.

Acknowledgment-triggered reporting

You can reduce alarm traffic ín your system by using acknowledgment-triggered alarm generation with the "ALARM (Page 5412)", "ALARM_8 (Page 5404)" and "ALARM_8P (Page 5406)" instructions.

That is, after an incoming alarm has been generated (Signal transition 0 > 1) initially for a signal, subsequent alarms will not be generated until you have acknowledged the first alarm on a display device. The next alarm displayed on the display unit after your acknowledgment is an outgoing alarm (signal transition 1 to 0). Subsequently the alarm cycle restarts with an incoming alarm (signal change from 0 to 1) that must be acknowledged. In this manner you can control the reports of signal change (until the outgoing alarm) via the display device.

You specify the alarm generation method (enable or disable acknowledgment-triggered alarm generation) for the "ALARM (Page 5412)", "ALARM_8 (Page 5404)" and "ALARM_8P (Page 5406)" instructions globally for the CPU during configuration.

To ensure consistent alarm evaluation within your system, you should verify that all display devices can handle acknowledgment-triggered reporting.

Note

(Note on display devices that cannot handle acknowledgment-triggered reporting.)

A CPU with enabled acknowledgment-triggered reporting will distribute the alarms only to display devices capable of handling this reporting method. The CPU will not send any alarms if none of the display devices can handle acknowledgment-triggered reporting. This situation is indicated once with ERROR=1 and STATUS=1.

Alarm acknowledgment for the instructions "ALARM", "ALARM_8" and "ALARM_8P"

A centralized acknowledgment concept is used. This means that, when you have acknowledged the alarm at a display device, the acknowledgment information is first sent to the CPU that generated the alarm. This then distributes the acknowledgment information to all stations logged on for this purpose.

You always acknowledge a signal and not an individual alarm. If, for example, several rising edges of a signal were indicated and you acknowledge the event entering the state, all previous events with the same alarm number count as having been acknowledged.

Acknowledgment display

The "NOTIFY (Page 5409)" and "NOTIFY_8P (Page 5402)" instructions have no acknowledgment display. With "ALARM (Page 5412)", you can obtain the acknowledgment status from the ACK_UP and ACK_DN output parameters. With "ALARM_8P (Page 5406)" and "ALARM_8 (Page 5404)", the acknowledgment status is available in the ACK_STATE output parameter. These outputs are updated when the block is called, providing the control parameter EN_R has the value "1".

Disabling and enabling alarms with the instruction or display device

In some situations, it may be useful to suppress alarms (e.g. when you reconfigure your system). You can therefore disable and enable alarms at the display device or in your program. Disabling/enabling applies to all stations that are logged on for the particular alarm. A disabled alarm remains disabled until it is enabled again.

You are informed of disabled alarms with the ERROR and STATUS output parameters (ERROR = 1, STATUS = 21).

Work memory required by the instructions for generating block-related alarms

For their proper operation, the instructions for generating PLC alarms require a communications data buffer in the work memory of the CPU (code area) that is generally dependent on the associated value. Refer to the table below for information on the size of used memory.

Instruction	Required space (in bytes) in the work memory of the CPU
NOTIFY	200 + 2 * length of the associated values specified at SD_1,...SD_10 at the first call
NOTIFY_8P	200 + 2 * length of the associated values specified at SD_1,...SD_10 at the first call
ALARM	200 + 2 * length of the associated values specified at SD_1,...SD_10 at the first call
ALARM_8	100
ALARM_8P	200 + 2 * length of the associated values specified at SD_1,...SD_10 at the first call
AR_SEND	54

Amount of transferable data

The amount of data that can be transferred using the associated values SD_i of the "NOTIFY (Page 5409)", "NOTIFY_8P (Page 5402)", "ALARM (Page 5412)" and "ALARM_8P (Page 5406)" instructions must not exceed a maximum length. This maximum data length is calculated as follows:

maxleng =

min (pdu_local, pdu_remote) - diff - 4 * number of SD_i parameters used

Where:

- min (pdu_local, pdu_remote) is the lowest value of the numbers pdu_lokal and pdu_remote
- pdu_local is the maximum length of the fields of the local CPU (see the technical data of your CPU)
- pdu_remote: the maximum length for display device data blocks
- diff = 48, if acknowledgment-triggered reporting is enabled, and diff = 44 if disabled.

Example

A CPU 414-2 is sending alarms via Industrial Ethernet to HMI. Acknowledgment-triggered reporting is disabled.

The associated values SD_1, SD_2 and SD_3 are used.

pdu_local = 480 bytes, pdu_remote = 480 bytes,

Number of SD_i parameters used: 3

Result:

maxleng = min (480, 480) - 44 - 4 * 3 = 480 - 44 - 12 = 424

The maximum length of data that can be transferred per instruction is 424 bytes.

NOTIFY_8P: Report up to eight signal changes (S7-400)

Description

The "NOTIFY_8P" instruction extends the "NOTIFY (Page 5409)" instruction to include eight signals.

An alarm is generated if at least one signal transition has been detected. An alarm is always generated at the first call of "NOTIFY_8P". All eight signals have a common alarm ID that is split into eight individual alarms on the display device.

One alarm memory with two memory locations is available for each instance of "NOTIFY_8P". For more information on the caching of signal changes, refer to Introduction to generating PLC alarms with instructions (Page 5398).

Note

The display device shows the last two signal transitions, irrespective of alarm loss.

WARNING

Calling "NOTIFY_8P"

Before you call "NOTIFY_8P" in an automation system, you must ensure that all connected display devices are familiar with this block. If you fall to heed this instruction the communication between the automation system and the connected display devices will be aborted. In this case you can no longer access your system with its connected display devices.

Parameters

The following table shows the parameters of the instruction "NOTIFY_8P":

Parameters	Declaration	Data type	Memory area	Description
SIG_i, 1 ≤ i ≤ 8	Input	BOOL	I, Q, M, D, L	i-(nth) signal to be monitored
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.
EV_ID	Input	C_NOTIFY_8P	I, Q, M, D, L	Alarm number (not allowed: 0) EV_ID is only evaluated at the first call. Subsequently, the alarm number used for the first call applies to every call of "NOTIFY_8P" with the corresponding instance DB. The alarm number is assigned automatically. This ensures the consistency of the alarm numbers. The alarm number must be unique within your user program.

Parameters	Declaration	Data type	Memory area	Description
SEVERITY	Input	WORD	I, Q, M, D, L or constant	Weighting of the event Possible values: 0 to 127 (value 0 means highest weighting); default value: 64 This parameter is irrelevant for processing the alarm.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Generation of alarm is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information
SD_i, 1 ≤ i ≤ 10	InOut	ANY	I, Q, M, D, T, C	i-th associated value Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME **Note:** When the ANY pointer accesses a DB, the DB must always be specified. (e.g.: P# DB10.DBX5.0 Byte 10)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

The following table contains all specific error information for "NOTIFY_8P" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Alarm loss: The previous signal change or the previous alarm could not be sent and is replaced by the current alarm.
0	22	• Error in the pointer to the associated values SD_i: – relating to the data length or the data type – No access to associated values in user memory, for example, due to deleted DB or area length error – The activated alarm is transferred without or, if required, with the maximum possible number of associated values. • The actual parameter you have selected for SEVERITY is higher than the permitted range. The activated alarm is sent with SEVERITY=127.
0	25	Communication has started. The alarm is being processed.
1	1	Communication problems: connection termination or no logon

ERROR	STATUS (decimal)	Explanation
1	4	At the first call: • The specified EV_ID is outside the permitted range or • the ANY pointer SD_i has a formal error. • The maximum memory area that can be sent for the CPU per "NOTIFY_8P" has been exceeded.
1	10	Access to local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "NOTIFY_8P" • A global DB was specified instead of an instance DB.
1	18	EV_ID was already used by the "NOTIFY (Page 5409)", "NOTIFY_8P", "ALARM (Page 5412)", "ALARM_8 (Page 5404)" or "ALARM_8P (Page 5406)" instructions.
1	20	Not enough work memory.
1	21	The alarm with the specified EV_ID is disabled.

See also

Creating program alarms (Page 7503)

ALARM_8: Create PLC alarms without associated values for eight signals (S7-400)

Description

Except for the non-existing, associated values SD_1, ..., SD_10, the instruction is identical to the "NOTIFY_8P (Page 5402)" instruction.

Parameters

The following table shows the parameters of the "ALARM_8" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive determines whether the output ACK_STATE is updated (EN_R=1) or not (EN_R=0) when the block is called.
SIG_i, 1≤i ≤8	Input	BOOL	I, Q, M, D, L	i-(nth) signal to be monitored
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.

Parameter	Declaration	Data type	Memory area	Description
EV_ID	Input	C_ALARM_8	I, Q, M, D, L	Alarm number (not allowed: 0) EV_ID is only evaluated at the first call. Subsequently, the alarm number used for the first call applies to every call of "ALARM_8" with the corresponding instance DB. The alarm number is assigned automatically. This ensures the consistency of the alarm numbers. The alarm number must be unique within your user program.
SEVERITY	Input	WORD	I, Q, M, D, L or constant	Weighting of the event Possible values: 0 to 127 (value 0 means highest weighting) This parameter is irrelevant for processing this alarm.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Generation of alarm is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information
ACK_STATE	Output	WORD	I, Q, M, D, L	Bit array with the current acknowledgement status of all eight alarms (1: Event acknowledged, 0: Event not acknowledged): • Bits 0 to 7 are mapped to the incoming event of SIG_1 to SIG_8 • Bits 8 to 15 are mapped to the outgoing event of SIG_1 to SIG_8 • Initialization status: W#16#FFFF, that is, all incoming and outgoing events have been acknowledged

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

The following table contains all specific error information for the instruction "ALARM_8" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Alarm loss: The previous signal change or the previous alarm could not be sent and is replaced by the current alarm.
0	22	The actual parameter you have selected for SEVERITY is higher than the permitted range. The activated alarm is sent with SEVERITY=127.
0	25	Communication has started. The alarm is being processed.

ERROR	STATUS (decimal)	Explanation
1	1	Communication problems: connection termination or no logon With acknowledgement-triggered reporting active: Temporary display, if display devices do not support acknowledgement-triggered reporting
1	4	At the first call, the specified EV_ID is outside the permitted range.
1	10	Access to local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "ALARM_8" • A global DB was specified instead of an instance DB.
1	18	EV_ID was already used by the "NOTIFY (Page 5409)", "NOTIFY_8P (Page 5402)", "ALARM (Page 5412)", "ALARM_8" or "ALARM_8P (Page 5406)" instructions.
1	20	Not enough work memory.
1	21	The alarm with the specified EV_ID is disabled.

Note

After the first call, all the bits of the ACK_STATE output are set and it is assumed that the previous values of SIG_i, 1 ≤ i ≤ 8 inputs were 0.

See also

Creating program alarms (Page 7503)

ALARM_8P: Create PLC alarms without associated values for eight signals (S7-400)

Description

The instruction is the extension of "ALARM (Page 5412)" to include eight signals.

As long as you have not enabled acknowledgement-triggered reporting, an alarm will always be generated when a signal transition is detected at one or more signals (exception: a alarm is always sent at the first block call). All eight signals have a common alarm ID that is split into eight separate alarms on the display device. You can acknowledge each individual alarm separately or a group of alarms at once.

Use the output parameter ACK_STATE to further process the acknowledgement status of the specific alarms in your program. If you enable or release an alarm of an "ALARM_8P" instruction, this action always affects the entire "ALARM_8P" instruction. Disabling and enabling of individual signals is not possible.

One alarm memory with two memory locations is available for each instance of "ALARM_8P". For more information on the caching of signal changes, refer to Introduction to generating PLC alarms with instructions (Page 5398).

Parameters

The following table shows the parameters of the "ALARM_8P" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive determines whether the output ACK_STATE is updated (EN_R=1) or not (EN_R=0) when the block is called.
SIG_i, 1≤i ≤8	Input	BOOL	I, Q, M, D, L	i-(nth) signal to be monitored
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.
EV_ID	Input	C_ALARM_8P	I, Q, M, D, L	Alarm number (not allowed: 0); EV_ID is only evaluated at the first call. Subsequently, the alarm number used for the first call applies to every call of "ALARM_8P" with the corresponding instance DB. The alarm number is assigned automatically. This ensures the consistency of the alarm numbers. The alarm number must be unique within your user program.
SEVERITY	Input	WORD	I, Q, M, D, L or constant	Weighting of the event Possible values: 0 to 127 (value 0 means highest weighting) This parameter is irrelevant for processing this alarm.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Generation of alarm is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information

Parameter	Declaration	Data type	Memory area	Description
ACK_STATE	Output	WORD	I, Q, M, D, L	Bit array with the current acknowledgement status of all eight alarms (1: Event acknowledged, 0: Event not acknowledged): • Bits 0 to 7 are mapped to the incoming event of SIG_1 to SIG_7 • Bits 8 to 15 are mapped to the outgoing event of SIG_1 to SIG_7 • Initialization status: W#16#FFFF, that is, all incoming and outgoing events have been acknowledged
SD_j, 1≤ j ≤10	InOut	ANY	I, Q, M, D, T, C	j-th associated value The associated values apply for all alarms. Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME. **Note:** When the ANY pointer accesses a DB, the DB must always be specified (for example: P# DB10.DBX5.0 byte 10).

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

The following table contains all specific error information for the "ALARM_8P" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Alarm loss: The previous signal change or the previous alarm could not be sent and is replaced by the current alarm.
0	22	• Error in the pointer to the associated values SD_i: – relating to the data length or the data type – No access to associated values in user memory, for example, due to deleted DB or area length error – The activated alarm is sent without associated values • The actual parameter you have selected for SEVERITY is higher than the permitted range. The activated alarm is sent with SEVERITY=127 .
0	25	Communication has started. The alarm is being processed.
1	1	Communication problems: Connection termination or no logon. With acknowledgement-triggered reporting active: Temporary display, if display devices do not support acknowledgement-triggered reporting.

ERROR	STATUS (decimal)	Explanation
1	4	At the first call: • The specified EV_ID is outside the permitted range or • the ANY pointer SD_i has a formal error. • The maximum memory area that can be sent for the CPU per "ALARM_8P" has been exceeded.
1	10	Access to local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called: • An instance DB was specified that does not belong to "ALARM_8P" • A global DB was specified instead of an instance DB.
1	18	EV_ID was already used by the "NOTIFY (Page 5409)", "NOTIFY_8P (Page 5402)", "ALARM (Page 5412)", "ALARM_8 (Page 5404)" or "ALARM_8P" instructions.
1	20	Not enough work memory.
1	21	The alarm with the specified EV_ID is disabled.

Note

After the first call, all the bits of the ACK_STATE output are set and it is assumed that the previous values of SIG_i, 1< i < 8 inputs were "0".

See also

Creating program alarms (Page 7503)

NOTIFY: Report a signal change (S7-400)

Description

The instruction monitors a signal. It generates an alarm both on a rising edge (incoming event) and on a falling edge (outgoing event). You can have up to ten associated values sent with the alarm. The alarm is sent to all stations logged on for this purpose. When the SFB is first called, an alarm with the current signal state is sent.

The associated values are queried when the edge is detected and assigned to the alarm.

One alarm memory with two memory locations is available for each instance of the instruction. For more information on the caching of signal changes, refer to Introduction to generating PLC alarms with instructions (Page 5398).

The "NOTIFY" instruction conforms to IEC 1131-5.

Parameter

The following table shows the parameters of the instruction "NOTIFY":

Parameters	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L	The signal to be monitored
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.
EV_ID	Input	C_NOTIFY	I, Q, M, D, L	Alarm number (not allowed: 0) EV_ID is only evaluated at the first call. Subsequently, the alarm number used for the first call applies to every call of "NOTIFY" with the corresponding instance DB. The alarm number is assigned automatically. This ensures the consistency of the alarm numbers. The alarm number must be unique within your user program.
SEVERITY	Input	WORD	I, Q, M, D, L or constant	Weighting of the event Possible values: 0 to 127 (value 0 means highest weighting) This parameter is irrelevant for processing this alarm.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Generation of alarm is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information
SD_i, 1≤i ≤10	InOut	ANY	I, Q, M, D, T, C	i-th associated value Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME. **Note:** When the ANY pointer accesses a DB, the DB must always be specified. (e.g.: P# DB10.DBX5.0 Byte 10)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

The following table contains all specific error information that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Alarm loss: The previous signal change or the previous alarm could not be sent and is replaced by the current alarm.
0	22	• Error in the pointer to the associated values SD_i: – relating to the data length or the data type – No access to associated values in user memory, for example, due to deleted DB or area length error – The activated alarm is transferred without or, if required, with the maximum possible number of associated values. • The actual parameter you have selected for SEVERITY is higher than the permitted range. The activated alarm is sent with SEVERITY=127 .
0	25	Communication has started. The alarm is being processed.
1	1	Communication problems: connection termination or no logon
1	4	At the first call: • The specified EV_ID is outside the permitted range or • the ANY pointer SD_i has a formal error. • The maximum memory area that can be sent for the CPU per "NOTIFY" has been exceeded.
1	10	Access to local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called: • An instance DB was specified that does not belong to "NOTIFY" • A global DB was specified instead of an instance DB.
1	18	EV_ID was already used by the "NOTIFY", "NOTIFY_8P (Page 5402)", "ALARM (Page 5412)", "ALARM_8 (Page 5404)" or "ALARM_8P (Page 5406)" instructions.
1	20	Not enough work memory.
1	21	The alarm with the specified EV_ID is disabled.

See also

Creating program alarms (Page 7503)

ALARM: Create PLC alarms with acknowledgement display (S7-400)

Description

The instruction monitors a signal.

- Default mode (that is, acknowledgement triggered reporting is disabled): The block generates an alarm both on a rising edge (incoming event) and on a falling edge (outgoing event). You can have up to ten associated values sent with the alarm.
- Acknowledgement-triggered reporting is enabled: After an incoming alarm is generated, the block will no longer create alarms for the signal until you have acknowledged this incoming alarm on a display device.
  See also: Introduction to generating PLC alarms with instructions (Page 5398).

The alarm is sent to all stations logged on for this purpose.

When the SFB is first called, an alarm with the current signal state is sent.

The ACK_UP output is reset at the rising edge. It is set when your acknowledgement of the incoming event has arrived from a logged on display device.

The situation for the ACK_DN output is analogous: this is reset at the falling edge. It is set when your acknowledgement of the outgoing event is received from a logged on display device. Once your acknowledgement has been received from a logged on display device, the acknowledgement information is passed on to all other stations logged on for this purpose.

One alarm memory with two memory locations is available for each instance of the "ALARM" instruction. For more detailed information on buffering of signal changes, refer to Section "Signal change detection" in Introduction to generating PLC alarms with instructions (Page 5398).

The "ALARM" instruction conforms to IEC 1131-5.

Parameters

The following table shows the parameters of the "ALARM" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive decides whether the outputs ACK_UP and ACK_DN are updated at the first block call (EN_R=1) or not (EN_R=0). If EN_R=0, then the output parameters ACK_UP and ACK_DN remain unchanged.
SIG	Input	BOOL	I, Q, M, D, L	The signal to be monitored
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.

Parameter	Declaration	Data type	Memory area	Description
EV_ID	Input	C_ALARM	I, Q, M, D, L	Alarm number (not allowed: 0) EV_ID is only evaluated at the first call. Subsequently, the alarm number used for the first call applies to every call of "ALARM" with the corresponding instance DB. The alarm number will be assigned automatically. This ensures the consistency of the alarm numbers. The alarm number must be unique within your user program.
SEVERITY	Input	WORD	I, Q, M, D, L or constant	Weighting of the event Possible values: 0 to 127 (value 0 means highest weighting) This parameter is irrelevant for processing this alarm.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Generation of alarm is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information
ACK_DN	Output	BOOL	I, Q, M, D, L	Event leaving state was acknowledged on a display device Initialization status: 1
ACK_UP	Output	BOOL	I, Q, M, D, L	Incoming event was acknowledged on a display device Initialization status: 1
SD_i, 1≤i ≤10	InOut	ANY	I, Q, M, D, T, C	i-th associated value Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME. **Note:** When the ANY pointer accesses a DB, the DB must always be specified. (e.g.: P# DB10.DBX5.0 Byte 10)

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

The following table contains all specific error information for the instruction "ALARM" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Alarm loss: The previous signal change or the previous alarm could not be sent and is replaced by the current alarm.
0	22	• Error in the pointer to the associated values SD_i: – Involving the data length or the data type – No access to associated values in user memory, for example, due to deleted DB or area length error – The activated alarm is sent without associated values • The actual parameter you have selected for SEVERITY is higher than the permitted range. The activated alarm is sent with SEVERITY=127.
0	25	Communication has started. The alarm is being processed.
1	1	Communication problems: connection termination or no logon With acknowledgement-triggered reporting active: Temporary display, if display devices do not support acknowledgement-triggered reporting
1	4	At the first call: • The specified EV_ID is outside the permitted range or • the ANY pointer SD_i has a formal error. • The maximum memory area that can be sent for the CPU per "ALARM " has been exceeded.
1	10	Access to local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to ALARM • A global DB was specified instead of an instance DB.
1	18	EV_ID was already used by the "NOTIFY (Page 5409)", "NOTIFY_8P (Page 5402)", "ALARM", "ALARM_8 (Page 5404)" or "ALARM_8P (Page 5406)" instructions.
1	20	Not enough work memory.
1	21	The alarm with the specified EV_ID is disabled.

Note

After the first call, the ACK_UP and ACK_DN outputs have the value "1" and it is assumed that the previous value of the SIG input was "0".

See also

Creating program alarms (Page 7503)

Instructions for generating PLC alarms without instance DB

Startup characteristics of the instructions for generating PLC alarms

Warm restart behavior

During a warm restart, the instance DBs of the instructions for generating PLC alarms are set to the not initialized status. The actual parameters stored in the instance DBs are unchanged. The parameters ID and EV_ID will be re-evaluated at the next block call.

Cold restart behavior

During a cold restart, the contents of the instance DBs of the instructions for generating PLC alarms are set to their initial values.

Hot restart

During a hot restart, the instructions for generating PLC alarms behave like user function blocks that are capable of resuming execution. They continue from the point of interruption.

Memory reset

A memory reset always causes the termination of all connections so that no station is logged on for alarms. The user program is deleted. If you have inserted a FLASH card, the program sections relevant to execution are loaded on the CPU again from the card and the CPU executes a warm or cold restart (implicitly this is always a cold restart, since all user data are initialized after clearing memory).

Error behavior of instructions for generating PLC alarms

Connection termination

The connections assigned to the instances are monitored for termination. If a connection is terminated, the stations involved are removed from the internal CPU list of stations logged on for PLC alarms. Any alarms pending for these stations are deleted.

If other stations are still logged on following a connection termination, they continue to receive alarms. The instructions only stop sending alarms when there are no more connections to any logged on stations. The ERROR and STATUS output parameters indicate this situation (ERROR = 1, STATUS = 1).

Error interface to the user program

If an error occurs during the execution of an instruction for generating PLC alarms, the ERROR output parameter is set to "1" and the STATUS output parameter has the corresponding error identifier. You can evaluate this error information in your program.

Examples of possible errors:

- Sending not possible due to lack of resources
- Error accessing one of the signals to be monitored.

Introduction to generating PLC alarms with instructions

Instructions for generating PLC alarms

You can generate a PLC alarm with the following instructions:

- "ALARM_SQ (Page 5421)"
- "ALARM_S (Page 5418)"
- "ALARM_DQ (Page 5425)"
- "ALARM_D (Page 5423)"

These instructions have the following properties:

- The PLC alarms of "ALARM_SQ (Page 5421)" and "ALARM_DQ (Page 5425)" sent with signal state "1" can be acknowledged from a logged-on display device. The PLC alarms of "ALARM_S" (Page 5418) and "ALARM_D (Page 5423)" are always acknowledged implicitly.
- It is not a detected edge change, but rather every call that generates a PLC alarm.
- After the block processing the associated value SD is completely entered and allocated to the PLC alarm.
  The following applies for the associated value regarding the consistency relative to higher priority classes: The following are consistent:
  - The simple data types (bit, byte, word, and double word)
  - an array of the data type byte up to a maximum length specific for a CPU

ALARM_SC

With the "ALARM_SC (Page 5427)" instruction, you can:

- Determine the acknowledgement status of the last "incoming alarm" and the signal status at the last call of "ALARM_SQ (Page 5421)" / "ALARM_DQ (Page 5425)", or
- Determine the signal status at the last call of "ALARM_S (Page 5418)" / "ALARM_D (Page 5423)".

Logging on display devices

The instructions for generating PLC alarms only send a PLC alarm when they are called if at least one display device has logged on for PLC alarms.

Alarm storage

To avoid alarms being lost when there is a lot of traffic on the communications system, the "ALARM_SQ (Page 5421)", "ALARM_S (Page 5418)", "ALARM_DQ (Page 5425)" and "ALARM_D (Page 5423)" instructions can each buffer two alarms.

If PLC alarms are still lost, you will be informed in RET_VAL. The logged on display devices are informed of this the next time a PLC alarm can be sent.

Alarm acknowledgement for the instructions "ALARM_SQ" and "ALARM_DQ"

If you have acknowledged an "incoming alarm" at a display device, this acknowledgement information is first sent to the CPU where the alarm originated. This then distributes the acknowledgement information to all stations logged on for this purpose.

Disabling and enabling alarms

You cannot disable and then re-enable PLC alarms that you have generated with "ALARM_SQ (Page 5421)" or "ALARM_S (Page 5418)" or "ALARM_DQ (Page 5425)" or "ALARM_D (Page 5423)".

Changes in your program containing "ALARM_SQ" / "ALARM_S" calls

Note

When you download a block with "ALARM_SQ (Page 5421)" / "ALARM_S (Page 5418)" calls that is already on the CPU, it is possible that the previous block has sent an incoming PLC alarm but that the new block has not sent a corresponding outgoing PLC alarm. This means that the alarm remains in the internal alarm memory of the CPU. This status can also occur if you delete blocks with "ALARM_SQ (Page 5421)" / "ALARM_S (Page 5418)" calls.

You can remove such PLC alarms from the internal alarm memory of the CPU by changing the CPU to STOP mode and then executing a warm restart or cold restart.

Changes in your program containing "ALARM_DQ" / "ALARM_D" calls

Even though your program might contain "ALARM_DQ (Page 5425)" and/or "ALARM_D (Page 5423)" calls, the program modifications described above may cause the PLC alarms to become resident in the internal alarm memory and thus permanently occupy system resources.

In contrast to system resources that were allocated by "ALARM_SQ (Page 5421)" / "ALARM_S (Page 5418)" calls, you can again release system resources allocated by "ALARM_DQ (Page 5425)" / "ALARM_D (Page 5423)" calls without having to switch your CPU to STOP mode. The "DEL_SI (Page 5440)" instruction is used for this purpose. Before you enable dynamically-allocated system resources by calling "DEL_SI", it is advisable to use the "READ_SI" instruction to output information regarding the system resources of your CPU that are currently dynamically allocated (see instruction "READ_SI (Page 5437)").

Amount of transferable data

The amount of data that can be transferred using the associated values SD of the "ALARM_S (Page 5418)", "ALARM_SQ (Page 5421)", "ALARM_D (Page 5423)" and "ALARM_DQ (Page 5425)" instructions must not exceed a maximum length. This maximum data length is calculated as follows:

maxleng = min (pdu_local, pdu_remote) - 48

Where:

- pdu_local: the maximum length of CPU data blocks (SZL_ID W#16#0131, INDEX 1, tag pdu)
- pdu_remote: the maximum length for display device data blocks

Example

A CPU 414-2 sends an alarm to a programming device PG 760 (via MPI).

pdu_local = 480 bytes, pdu_remote = 480 bytes,

Result:

maxleng = min (480, 480) - 48 = 480 - 48 = 432

The maximum length of data that can be transferred per instruction is 432 bytes.

ALARM_S: Generate alarm message

Description

With every call, the instruction generates an alarm to which you can append an associated value. The alarm is sent to all stations logged on for this purpose. The instruction provides you with a simple alarm mechanism. You must make sure that you call the instruction only when the value of the alarm-triggering signal SIG is inverted compared to the last call. If this is not the case, then this situation will be indicated in RET_VAL and no alarm will be sent. With the very first call of the instruction you must ensure that the value "1" is applied at the SIG input. Otherwise you will get error information via RET_VAL and no alarm will be sent.

Note

In newly created programs you should use only the "ALARM_D (Page 5423)" instruction (provided your CPU supports this instruction) because they provide improved options for managing system resources.

Use of system resources

When alarms are generated the operating system allocates a system resource for the duration of one signal cycle.

The signal cycle lasts from the call with SIG=1 until the next call with SIG=0. If, during the signal cycle, the alarm-generating block is overloaded or deleted, the associated system resource remains assigned until the next restart (warm restart).

Acknowledging alarms

Alarms you have sent with "ALARM_S" are always implicitly acknowledged. You can determine the signal state at the last "ALARM_S" call using "ALARM_SC (Page 5427)".

Caching signal states

The ALARM_S" instruction allocates system resources Here among other things the last two signal states, including time stamp and associated value are placed in the buffer. If "ALARM_S" is called at a time when the signal states of the two last "valid" calls have not yet been sent (signal overflow), the current signal state and the last signal state are discarded and an overflow ID is set in the buffer. At the next possible opportunity, the next to last signal and the overflow identifier are sent.

Example:

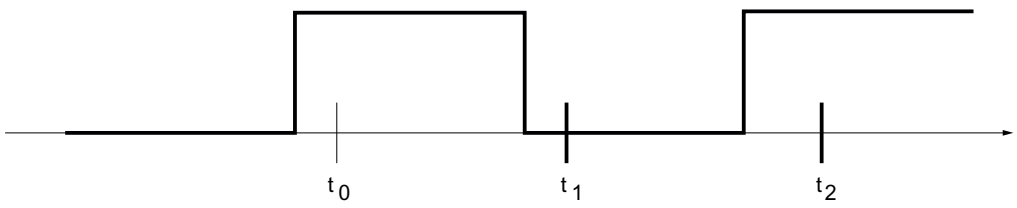

If t_0, t_1 and t_2 are the call times of "ALARM_S". If the signal states of t_0 and t_1 have not yet been sent at the time t_2, the signal states of t_1 and t_2 will be discarded and the overflow identifier is set at the signal state of t_0.

Instance overflow

If the number of "ALARM_S" calls is greater than the maximum number of CPU system resources, this may result in a lack of resources (instance overflow). This situation is indicated both by the information in RET_VAL as well as by indications at the logged on display devices.

The maximum number of "ALARM_S" calls is dependent on the CPU.

Parameter

The following table shows the parameters of the instruction "ALARM_S":

Parameters	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L	The alarm triggering signal
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE
EV_ID	Input	C_ALARM_S	I, Q, M, D, L	Alarm number (not allowed: "0")
SD	Input	ANY	I, Q, M, D, T, C	Associated value Maximum length: 12 bytes Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME.
RET_VAL	Return	INT	I, Q, M, D, L	Error information

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
0001	• The length of the associated value exceeds the maximum permitted length. • Access to user memory not possible (for example, access to deleted DB). The alarm is sent. • The associated value points to a value in the local data area. The alarm is sent. (S7-400 only)
0002	Warning: The last free alarm acknowledge memory was assigned. (S7-400 only)
8081	The specified EV_ID is outside the permitted range or
8082	Alarm loss because your CPU has no more resources for generating PLC alarms with instructions.
8083	Alarm loss, because the same signal transition is already present but could not be sent yet (signal overflow).
8084	With the current and the previous "ALARM_S", the alarm-triggering signal SIG has the same value.
8085	There is no logon for the specified EV_ID.
8086	A call for the specified EV_ID is already being processed in a lower priority class.
8087	During the first call of "ALARM_S", the alarm-triggering signal had the value "0".
8088	The specified EV_ID is already in use by another system resource ("ALARM_S", ALARM_DQ (Page 5425), ALARM_D (Page 5423)).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

See also

Creating program alarms (Page 7503)

ALARM_SQ: Generate alarm message with acknowledgement

Description

With every call, the instruction generates an alarm to which you can append an associated value. The alarm is sent to all stations logged on for this purpose. "ALARM_SQ" also provides you with a simple alarm mechanism. You must make sure that you call "ALARM_SQ" only when the value of the alarm-triggering signal SIG is inverted to the last call. If this is not the case, then this situation will be indicated in RET_VAL and no alarm will be sent. Thus with the very first call of "ALARM_SQ", you must ensure that the value "1" is applied at the SIG input. Otherwise you will get error information via RET_VAL and no alarm will be sent.

Note

In newly created programs you should use only the "ALARM_DQ (Page 5425)" instruction (provided your CPU supports this instruction) because they provide improved options for managing system resources.

Use of system resources

When alarms are generated with "ALARM_SQ", the operating system allocates a system resource for the duration of one signal cycle.

With "ALARM_SQ", the signal cycle lasts from the call with SIG=1 until the next call with SIG=0. Added to this time interval is possibly also the time until the acknowledgement of the incoming signal by a logged on display device.

If, during the signal cycle, the alarm-generating block is overloaded or deleted, the associated system resource remains assigned until the next restart (warm restart).

Acknowledging alarms

On a logged-on display device, you can acknowledge the alarms with signal status "1" sent by "ALARM_SQ". You can determine the acknowledgement status of the last "incoming alarm" and the signal status at the last "ALARM_SQ" call using "ALARM_SC (Page 5427)".

Caching signal states

"ALARM_SQ" allocates system resources Here among other things the last two signal states, including time stamp and associated value are placed in the buffer. If "ALARM_SQ" is called at a time when the signal states of the two last "valid" calls have not yet been sent (signal overflow), the current signal state and the last signal state are discarded and an overflow ID is set in the buffer. At the next possible opportunity, the next to last signal and the overflow identifier are sent.

Example:

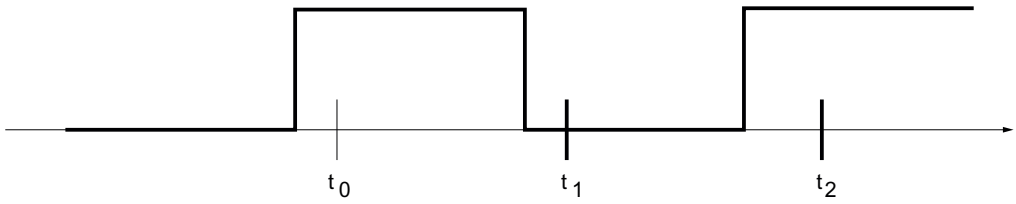

If t_0, t_1 and t_2 are the call times of "ALARM_SQ". If the signal states of t_0 and t_1 have not yet been sent at the time t_2, the signal states of t_1 and t_2 will be discarded and the overflow identifier is set at the signal state of t_0.

Instance overflow

If the number of "ALARM_SQ" calls is greater than the maximum number of CPU system resources, this may result in a lack of resources (instance overflow). This situation is indicated both by the information in RET_VAL as well as by indications at the logged on display devices.

The maximum number of "ALARM_SQ" calls is dependent on the CPU.

Parameter

The following table shows the parameters of the instruction "ALARM_SQ":

Parameter	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L	The alarm triggering signal
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE
EV_ID	Input	C_ALARM_S	I, Q, M, D, L	Alarm number (not allowed: 0)
SD	Input	ANY	I, Q, M, D, T, C	Associated value Maximum length: 12 bytes Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME.
RET_VAL	Return	INT	I, Q, M, D, L	Error information

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
0001	• The length of the associated value exceeds the maximum permitted length. • Access to user memory not possible (for example, access to deleted DB). The alarm is sent. • The associated value points to a value in the local data area. The alarm is sent. (S7-400 only)
0002	Warning: The last free alarm acknowledge memory was assigned. (S7-400 only)

Error code (W#16#...)	Explanation
8081	The specified EV_ID is outside the permitted range or
8082	Alarm loss because your CPU has no more resources for generating PLC alarms with instructions.
8083	Alarm loss, because the same signal transition is already present but could not be sent yet (signal overflow).
8084	With the current and the previous "ALARM_SQ", the alarm-triggering signal SIG has the same value.
8085	There is no logon for the specified EV_ID.
8086	A call for the specified EV_ID is already being processed in a lower priority class.
8087	During the first call of "ALARM_SQ", the alarm-triggering signal had the value "0".
8088	The specified EV_ID is already in use by another system resource (ALARM_S (Page 5418), ALARM_DQ (Page 5425), ALARM_D (Page 5423)).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

See also

Creating program alarms (Page 7503)

ALARM_D: Create permanently acknowledged PLC alarms

Description

With every call, the instruction generates a PLC alarm to which you can append an associated value. It conforms to the "ALARM_S (Page 5418)" instruction in this regard.

When alarms are generated with "ALARM_D", the operating system allocates a system resource for the duration of one signal cycle.

With "ALARM_D", the signal cycle lasts from the call with SIG=1 until the next call with SIG=0. If, during the signal cycle, the alarm-generating block is overloaded or deleted, the associated system resource remains assigned until the next restart (warm restart).

Compared to the "ALARM_S (Page 5418)" instruction, "ALARM_D" has an additional functionality that allows it to manage allocated system resources.

- With the help of READ_SI, you can read information regarding allocated system resources.
- With DEL_SI, you can re-enable allocated system resources. This is of special significance for permanently assigned system resources. A currently allocated system resource, for example, stays allocated until the next restart (warm restart) if you delete an FB call that contains "ALARM_D" calls in the course of a program change. When you change the program and reload an FB with "ALARM_D" calls, it may be the case that "ALARM_D" does not generate any more PLC alarms.

The instruction "ALARM_D" contains one parameter more than the "ALARM_S (Page 5418)" instruction, namely the input CMP_ID. You use this input to assign the PLC alarms generated by the "ALARM_D" instruction to logical areas, for example, to parts of the system. If you call "ALARM_D" in an FB, the obvious thing to do is to assign the number of the corresponding instance DB to "CMP_ID".

Parameter

The following table shows the parameters of the instruction "ALARM_D":

Parameter	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L	The alarm triggering signal
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE
EV_ID	Input	C_ALARM_S	I, Q, M, D, L	Alarm number (not allowed: 0)
CMP_ID	Input	DWORD	I, Q, M, D, L or constant	component identifier (not permitted: 0) ID for the subsystem to which the corresponding alarm is assigned Recommended values: • Low word: 1 to 65535 • High word: 0 You will not have any problems with the SIEMENS program package if you comply with these recommendations.
SD	Input	ANY	I, Q, M, D, T, C	Associated value Maximum length: 12 bytes Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME
RET_VAL	Return	INT	I, Q, M, D, L	Error information

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
0001	• The length of the associated value exceeds the maximum permitted length, or • Access to user memory not possible (for example, access to deleted DB). The alarm is sent. • The associated value points to a value in the local data area. The alarm is sent. (S7-400 only)
0002	Warning: The last free alarm acknowledge memory was assigned. (S7-400 only)
8081	The specified EV_ID is outside the permitted range or
8082	Alarm loss because your CPU has no more resources for generating PLC alarms with instructions.
8083	Alarm loss, because the same signal transition is already present but could not be sent yet (signal overflow).
8084	With the current and the previous "ALARM_D", the alarm-triggering signal SIG has the same value.
8085	There is no logon for the specified EV_ID.
8086	A call for the specified EV_ID is already being processed in a lower priority class.
8087	During the first call of "ALARM_D", the alarm-triggering signal had the value "0".

Error code (W#16#...)	Explanation
8088	The specified EV_ID is already in use by another system resource ("ALARM_SQ (Page 5421)", "ALARM_S (Page 5418)", "ALARM_DQ" (Page 5425)).
8089	You have assigned the value "0" to CMP_ID.
808A	CMP_ID does not match EV_ID
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

See also

Creating program alarms (Page 7503)

ALARM_DQ: Create acknowledgeable PLC alarms

Description

With every call, the instruction "ALARM_DQ" generates an alarm to which you can append an associated value. They conform to the "ALARM_SQ (Page 5421)" instruction in this regard.

When alarms are generated with "ALARM_DQ", the operating system allocates a system resource for the duration of one signal cycle.

With "ALARM_DQ", the signal cycle lasts from the call with SIG=1 until the next call with SIG=0. Added to this time interval is possibly also the time until the acknowledgement of the incoming signal by a logged on display device.

If, during the signal cycle, the alarm-generating block is overloaded or deleted, the associated system resource remains assigned until the next restart (warm restart).

The additional functionality of "ALARM_DQ" compared to the "ALARM_SQ (Page 5421)" instruction allows "ALARM_DQ" to manage occupied system resources .

- With the help of "READ_SI (Page 5437)", you can read information regarding occupied system resources.
- With "DEL_SI (Page 5440)", you can release occupied system resources again. This is of special significance for permanently assigned system resources. A currently allocated system resource, for example, stays allocated until the next restart (warm restart) if you delete an FB call that contains "ALARM_DQ" calls in the course of a program change. When you change the program and reload an FB with "ALARM_DQ" calls, it may be the case that "ALARM_DQ" does not generate any more alarms.

The "ALARM_DQ" instruction contains one parameter more than the "ALARM_SQ (Page 5421)" instruction, namely the input CMP_ID. You use this input to assign the alarms generated by the "ALARM_DQ" instruction to logical areas, for example, to parts of the system. If you call "ALARM_DQ" in an FB, the obvious thing to do is to assign the number of the corresponding instance DB to CMP_ID.

Parameter

The following table shows the parameters of the instruction "ALARM_DQ":

Parameter	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L	The alarm triggering signal
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE
EV_ID	Input	C_ALARM_S	I, Q, M, D, L	Alarm number (not allowed: 0)
CMP_ID	Input	DWORD	I, Q, M, D, L or constant	component identifier (not permitted: 0); ID for the subsystem to which the corresponding alarm is assigned Recommended values: • Low word: 1 to 65535 • High word: 0 You will not have any problems with the SIEMENS program package if you comply with these recommendations.
SD	Input	ANY	I, Q, M, D, T, C	Associated value Maximum length: 12 bytes Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME
RET_VAL	Return	INT	I, Q, M, D, L	Display

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
0001	• The length of the associated value exceeds the maximum permitted length, or • Access to user memory not possible (for example, access to deleted DB). The alarm is sent. • The associated value points to a value in the local data area. The alarm is sent. (S7-400 only)
0002	Warning: The last free alarm acknowledge memory was assigned. (S7-400 only)
8081	The specified EV_ID is outside the permitted range or
8082	Alarm loss because your CPU has no more resources for generating PLC alarms with instructions.
8083	Alarm loss, because the same signal transition is already present but could not be sent yet (signal overflow).
8084	With the current and the previous "ALARM_DQ", the alarm-triggering signal SIG has the same value.
8085	There is no logon for the specified EV_ID.
8086	A call for the specified EV_ID is already being processed in a lower priority class.
8087	During the first call of "ALARM_DQ", the alarm-triggering signal had the value "0".

Error code (W#16#...)	Explanation
8088	The specified EV_ID is already in use by another system resource (ALARM_SQ (Page 5421) , ALARM_S (Page 5418) , ALARM_D (Page 5423)).
8089	You have assigned the value "0" to CMP_ID .
808A	CMP_ID does not match EV_ID
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

See also

Creating program alarms (Page 7503)

ALARM_SC: Determine acknowledgement status of the last ALARM_SQ incoming alarm

Description

You use the instruction to do the following:

- Determine the acknowledgement status of the last "ALARM_SQ (Page 5421)" / "ALARM_DQ (Page 5425)" incoming alarm and the status of the alarm-triggering signal when the last "ALARM_SQ (Page 5421)" / "ALARM_DQ (Page 5425)" was called, or
- Determine the status of the alarm-triggering signal at the last "ALARM_S (Page 5418)" / "ALARM_D (Page 5423)" call.

The alarm or the signal is uniquely referenced via the alarm number that you assign here, if you have assigned alarms numbers via the alarm configuration.

The instruction accesses the temporarily allocated memory of the "ALARM_SQ (Page 5421)", "ALARM_S (Page 5418)", "ALARM_DQ (Page 5425)" and "ALARM_D (Page 5423)" instructions.

Parameter

The following table shows the parameters of the instruction "ALARM_SC":

Parameter	Declaration	Data type	Memory area	Description
EV_ID	Input	C_ALARM_S	I, Q, M, D, L	Alarm number for which you want to determine the signal state at the last call or the acknowledgement status of the last incoming alarm (only with ALARM_SQ and with ALARM_DQ).
RET_VAL	Return	INT	I, Q, M, D, L	Error information

Parameter	Declaration	Data type	Memory area	Description
STATE	Output	BOOL	I, Q, M, D, L	State of the alarm-triggering signal at the last call
Q_STATE	Output	BOOL	I, Q, M, D, L	• If the specified EV_ID parameter belongs to an "ALARM_S" / "ALARM_D" call: "1" • If the specified EV_ID parameter belongs to an "ALARM_SQ" / "ALARM_DQ" call: Acknowledgement status of the last incoming alarm: – 0: Not acknowledged – 1: Acknowledged

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
8081	The specified EV_ID is outside the permitted range or
8082	No memory location is currently assigned to this EV_ID (possible cause: The corresponding signal state was never "1" or its signal state has already returned to "0").
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

See also

Creating program alarms (Page 7503)

WR_USMSG: Write a user diagnostics event to the diagnostic buffer

Description

You use the instruction to write a user diagnostics result in the diagnostics buffer. You can also send the corresponding user diagnostics alarm to all stations logged on for this purpose (by setting the input parameter SEND = TRUE). If an error occurs, the output parameter RET_VAL provides the error information.

Sending user diagnostics alarm

The instruction writes a user diagnostics event to the diagnostics buffer. You can also send the corresponding user diagnostics alarm to all stations logged on for this purpose (by setting the input parameter SEND = TRUE). The user diagnostics alarm is then written to the send buffer and automatically sent to the stations logged on for this purpose.

You can check whether the sending of user diagnostics alarms is currently possible. To do this, call the "RDSYSST (Page 5444)" instruction with the SZL_ID = W#16#0132 and INDEX = W#16#0005 parameters. The fourth word of the data record supplied in this manner indicates whether sending is currently possible (1) or not (0).

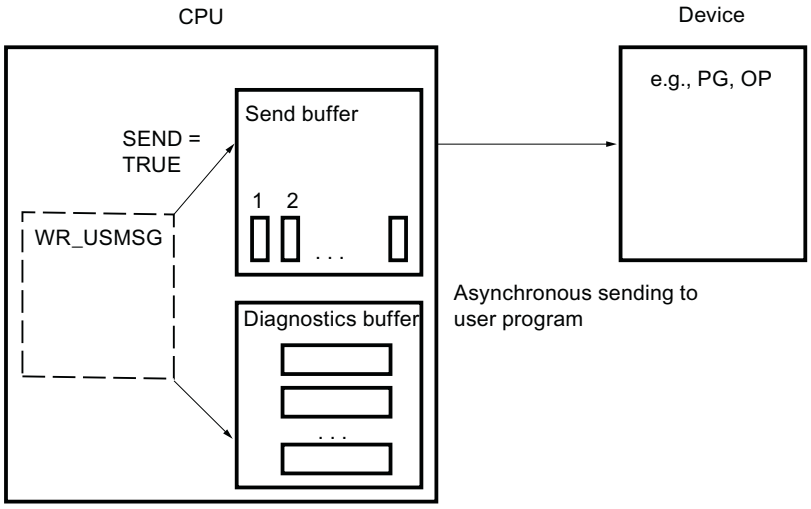

Send buffer full

The user diagnostics alarm can only be entered in the send buffer if the send buffer is not full. The number of entries that can be made in the send buffer depends on the type of CPU you are using.

If the send buffer is full, then:

- The diagnostics event is nevertheless entered in the diagnostics buffer.
- The RET_VAL parameter indicates that the send buffer is full. (RET_VAL = W#16#8092).

Station not logged on

If a user diagnostics alarm is to be sent (SEND = TRUE) and no station is logged on,

- the user diagnostics event will still be entered in the diagnostics buffer.
- the RET_VAL parameter indicates that no station is logged on (W#16#0091 or W#16#8091. The value W#16#8091 appears only with older versions of the CPU.)

Structure of an entry

The internal structure of an entry in the diagnostics buffer is as follows:

Byte	Contents
1 and 2	Event ID
3	Priority class
4	OB number
5 and 6	Reserved

Byte	Contents
7 and 8	Additional information 1
9, 10, 11 and 12	Additional information 2
13 to 20	Time stamp

Event ID

An event ID is assigned to every event.

Additional information

This is additional information about the event. The additional information can differ for each event. When you create a user diagnostics event, you can decide on the content of these entries yourself.

When you send a user diagnostics alarm, you can integrate the additional information as associated values in the (event ID-specific) alarm text.

Time stamp

The time stamp is of the type Date_and_Time.

Parameters

The following table shows the parameters of the "WR_USMSG" instruction:

Parameter	Declaration	Data type	Memory area	Description
SEND	Input	BOOL	I, Q, M, D, L	Enable the sending of the user diagnostics alarm to all logged-on stations
EVENTN	Input	WORD	I, Q, M, D, L or constant	Event ID. You assign the event ID. The assignment is not executed by the alarm server.
INFO1	Input	ANY	I, Q, M, D, L	Additional information: 1 word long
INFO2	Input	ANY	I, Q, M, D, L	Additional information: 2 words long
RET_VAL	Return	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter SEND

If SEND = TRUE , the user diagnostics alarm will be sent to all logged-on stations. The alarm is only sent if the station is logged on and if the send buffer is not full. The sending of the element is asynchronous to the user program.

Parameter EVENTN

The EVENTN parameter contains the event ID of the user event. You can enter event IDs of the form W#16#8xyz , W#16#9xyz, W#16#Axyz, W#16#Bxyz.

IDs in the form W#16#8xyz and W#16#9xyz belong to predefined events, IDs in the form W#16#Axyz and W#16#Bxyz belong to freely defined events.

An incoming event is indicated by x = 1, an outgoing event by x = 0. With events in class A and B: yz is the number assigned in hexadecimal presentation in the alarm configuration for the corresponding alarm.

Parameter INFO1

The INFO1 parameter contains information that is one word long. The following data types are permitted for INFO1 :

- WORD
- INT
- ARRAY [0...1] OF CHAR

You can integrate the INFO1 parameter as an associated value in the alarm text and use it to add up-to-date information to the alarm.

Parameter INFO2

The INFO2 parameter contains information that is two words long. The following data types are permitted for INFO2 :

- DWORD
- DINT
- REAL
- TIME
- ARRAY [0...3] OF CHAR

You can integrate the INFO2 parameter as an associated value in the alarm text and use it to add up-to-date information to the alarm.

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
0091	No station logged on (diagnostics event entered in the diagnostics buffer).
8083	INFO1 data type is not permitted
8084	INFO2 data type is not permitted
8085	EVENTN data type is not permitted
8086	Length of INFO1 is not permitted
8087	Length of INFO2 is not permitted

Error code (W#16#...)	Explanation
8091	(This error code appears only with older versions of the CPU.) No station logged on (diagnostics event entered in the diagnostics buffer).
8092	Sending not possible at present, send buffer full (diagnostics event entered in the diagnostics buffer).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

AR_SEND: Send archive data (S7-400)

Description

The instruction sends archive data to operator control and operating systems logged on for this purpose. These systems inform the CPU of the relevant archive number in the logon message frame. Depending on the memory available on the CPU and the operand area used, the archive data can be up to 65 534 bytes long. The defaults of the operator interface system you are using must be taken into consideration in the structure of the archive data.

The sending process is activated after the block is called and a rising edge is detected at the control input REQ. The start address of the archive data that is to be sent is specified by SD_1, the length of the data field by LEN. Data transfer is asynchronous to the execution of the user program. Successful completion of the send operation is indicated by the value of the DONE status parameter being set to "1". If there is a rising edge at control input R, the current sending process is canceled.

Parameters

The following table shows the parameters of the "AR_SEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request
R	Input	BOOL	I, Q, M, D, L	Control parameter reset: Current job aborted
ID	Input	WORD	I, Q, M, D or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.
AR_ID	Input	C_AR_SEND	I, Q, M, D, L or constant	Archive number (not permitted: 0); AR_ID is only evaluated at the first call. Subsequently, the archive number used for the first call applies to every call of AR_SEND with the corresponding instance DB. The archive number is assigned automatically. This ensures the consistency of the archive numbers.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Sending complete

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information
SD_1	InOut	ANY	I, Q, M, D, T, C	Pointer to archive data. The length specification is not evaluated. Only the BOOL data types are permitted (not permitted: Bit array, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME. The archive data must have a destination-system specific structure. **Note:** When the ANY pointer accesses a DB, the DB must always be specified (for example: P# DB10.DBX5.0 byte 10).
LEN	InOut	WORD	I, Q, M, D, L	Length of the data field to be sent in bytes

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problems
1	2	Negative acknowledgement, function cannot be executed
1	3	There is no logon for the specified AR_ID.
1	4	• Error in the archive data pointer SD_1 involving data length or data type. • At the first call, the specified AR_ID is outside the permitted range.
1	5	Requested reset was executed.
1	7	RESET job irrelevant because the current function was completed or not activated (block in incorrect status).
1	10	Access to local user memory not possible (e.g. access to deleted DB)
1	12	When the instruction was called • An instance DB was specified that does not belong to AR_SEND • A global DB was specified instead of an instance DB.
1	18	AR_ID has already been used by an AR_SEND.
1	20	Not enough work memory.

Data consistency

To ensure data consistency, you may only write to the part of the currently used sending area SD_1 after the current send operation is complete. This is the case when the DONE status parameter assumes the value "1".

Others

EN_MSG: Enable PLC alarms (S7-400)

Description

You use this instruction to enable PLC alarms you have disabled. You may have disabled the alarms either on a display device or using "DIS_MSG (Page 5436)".

You specify the alarms to be enabled using the MODE and MESGN input parameters. Successful enabling of alarms with the instruction is only possible if the instruction is not already actively enabling alarms.

You start the enable process by assigning the value "1" to the input parameter REQ during the call.

Functional description

The enable process is executed asynchronously, in other words, it can extend over several calls from the instruction (see also Difference between synchronous and asynchronous instructions (Page 4186)):

- When the instruction is first called (REQ =1), it checks the input parameters and attempts to assign the required system resources. In the good case, RET_VAL W#16#7001 is entered, BUSY is set, and enabling is triggered.
  Otherwise the associated error information is entered in RET_VAL and the job is concluded. In this case, BUSY should not be evaluated.
- If there are further calls in the meantime, the value RET_VAL W#16#7002 is entered (job still being executed by the CPU) and BUSY is set. An intermediate call does not affect the current job.
- The last time the instruction is called, the value RET_VALW#16#0000 is entered if no error has occurred. In this case, BUSY is written with 0. If there is an error then the error information is entered in RET_VAL and BUSY should not be evaluated.

Parameter

The following table shows the parameters of the instruction "EN_MSG":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Trigger enable
MODE	Input	BYTE	I, Q, M, D, L or constant	See the table below for parameters for selecting the alarms to be enabled;

Parameter	Declaration	Data type	Memory area	Description
MESGN	Input	DWORD	I, Q, M, D, L or constant	Alarm number Only relevant when MODE = 6. This allows a single alarm to be enabled.
RET_VAL	Return	INT	I, Q, M, D, L	Error information (see below);
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The enabling process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter MODE

The following table shows the permitted values for the MODE input parameter.

Value	Meaning
0	All PLC alarms of the CPU generated with instructions
1	All CPU PLC alarms generated with instructions, which means all alarms of "NOTIFY (Page 5409)", "NOTIFY_8P (Page 5402)", "ALARM (Page 5412)", "ALARM_8P (Page 5406)" and "ALARM_8 (Page 5404)".
6	Single alarm of the "PLC alarms generated with instructions" class

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	Enabling was terminated without an error.
7000	REQ = 0 at first call: enabling was not activated.
7001	REQ = 1 at first call: enabling was triggered.
7002	Intermediate call: enabling is already active.
8081	Error accessing a parameter
8082	MODE has an illegal value.
8083	The alarm number is outside the permitted range of values.
8084	There is no logon for the alarm(s) specified with MODE and possibly MESGN .
80C3	Enabling of alarm(s) specified via MODE and possibly MESGN , cannot be initiated because another enabling process of "EN_MSG " is currently active.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

DIS_MSG: Disable PLC alarms (S7-400)

Description

You use this instruction to disable PLC alarms you have created with instructions. You select alarms to be disabled using the input parameters MODE and MESGN. Calling the "DIS_MSG" instruction and successfully disabling an alarm is only possible if disabling of an alarm is not already active with "DIS_MSG".

Alarms that are ready to be sent when "DIS_MSG" is called but that are still in an internal buffer can no longer be disabled and are sent.

A disabled alarm transmission is indicated at the ERROR and STATUS outputs of the "NOTIFY (Page 5409)", "NOTIFY_8P (Page 5406)", "ALARM (Page 5412)", "ALARM_8P (Page 5406)" and "ALARM_8 (Page 5404)" instructions.

You start the process by assigning the value "1" to the input parameter REQ when "DIS_MSG" is called.

Functional description

The reading process is executed asynchronously, in other words, it can extend over several calls (see also Difference between synchronous and asynchronous instructions (Page 4186)):

- When it is first called (REQ =1), "DIS_MSG" checks the input parameters and attempts to occupy the required system resources. In the good case, RET_VALW#16#7001 is entered, BUSY is set and disabling is triggered.
  Otherwise the associated error information is entered in RET_VAL and the job is concluded. In this case, BUSYmust not be evaluated.
- If there are further calls in the meantime, the value RET_VALW#16#7002 is entered (job still being executed by the CPU) and BUSY is set. An intermediate call does not affect the current job.
- The last time the instruction is called, the value RET_VALW#16#0000 is entered if no error has occurred. In this case, BUSY is written with "0". If there is an error then the error information is entered in RET_VAL and BUSY should not be evaluated.

Parameter

The following table shows the parameters of the instruction "DIS_MSG":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Trigger disable
MODE	Input	BYTE	I, Q, M, D, L or constant	Parameter for selecting the alarms to be disabled, see following table
MESGN	Input	DWORD	I, Q, M, D, L or constant	Alarm number Alarm number only relevant when MODE = 5, 6, 7. This allows a single alarm to be disabled.

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Error code (see below)
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The disabling process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter MODE

The following table shows the permitted values for the MODE input parameter.

Value	Meaning
0	All PLC alarms of the CPU generated with instructions
1	All CPU PLC alarms generated with instructions, which means all alarms of "NOTIFY (Page 5409)", "NOTIFY_8P (Page 5402)", "ALARM (Page 5412)", "ALARM_8P (Page 5406)" and "ALARM_8 (Page 5404)".
6	Single alarm of the "PLC alarms generated with instructions" class

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	Disabling was terminated without an error.
7000	REQ = 0 at first call: disabling was not activated.
7001	REQ = 1 at first call: disabling was triggered.
7002	Intermediate call: disabling is already active.
8081	Error accessing a parameter
8082	MODE has an illegal value.
8083	The alarm number is outside the permitted range of values.
8084	There is no logon for the alarm(s) specified with MODE and possibly MESGN .
80C3	Disabling of alarm(s) specified via MODE and possibly MESGN , cannot be initiated because another disabling process of "DIS_MSG" is currently active.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

READ_SI: Read out dynamically assigned system resources

Origin of dynamically allocated system resources when generating alarms with the "ALARM_DQ" and "ALARM_D" instructions

For alarm generation with "ALARM_DQ (Page 5425)" and "ALARM_D (Page 5423)", the operating system temporarily allocates memory space in the system memory.

For example, if you delete an FB with "ALARM_DQ (Page 5425)" or "ALARM_D (Page 5423)" calls that was present in the CPU, it may be the case that corresponding system resources

remain permanently allocated. If you reload the FB with "ALARM_DQ (Page 5425)" or "ALARM_D (Page 5423)" calls, it may be that the "ALARM_DQ (Page 5425)" and "ALARM_D (Page 5423)" instructions are no longer processed properly.

Description

Use the "READ_SI" instruction to read out the currently used system resources that were allocated for alarm generation with "ALARM_DQ (Page 5425)" and "ALARM_D (Page 5423)". This is done via the values of EV_ID and CMP_ID used there. These values are passed on to the "READ_SI" instruction in the SI_ID parameter.

The "READ_SI" instruction has four possible operating modes, which are explained in the following table. You set the requested operating mode via the MODE parameter.

MODE	Which of the system resources allocated to "ALARM_DQ" / "ALARM_D" are read out?
1	All (calling of "READ_SI" using SI_ID:=0)
2	The system resources that were allocated to the "ALARM_DQ" / "ALARM_D" call with EV_ID:=ev_id calling of READ_SI using SI_ID:=ev_id)
3	All system resources that were allocated to the "ALARM_DQ" / "ALARM_D" call with CMP_ID:=cmp_id (calling of using READ_SI using SI_ID:=cmp_id)
0	Additional system resources that could not be read with the previous call in MODE=1 or MODE=3 because you have selected a destination field SYS_INST that is too small

Functional description

If you have selected a sufficiently large SYS_INST destination area when you called READ_SI with MODE=1 or MODE=3, it will contain the content of all currently occupied system resources selected via the MODE parameter after the call.

If the amount of system resources currently occupied is very large, the runtime will be correspondingly high. That is, a high load on CPU performance may cause the the maximum configurable cycle monitoring time to be exceeded.

You can work around this runtime problem as follows: Select a relatively small SYS_INST destination area. If the instruction cannot enter all of the system resources that will be read out in SYS_INST, this fact will be communicated to you via RET_VAL=W#16#0001. Then call READ_SI with MODE=0 and the same SI_ID as for the previous call until RET_VAL takes on the value W#16#0000.

Note

Because the operating system does not coordinate the READ_SI calls that belong to a read job, you should execute all READ_SI calls in the same priority class.

Parameter

The following table shows the parameters of the instruction "READ_SI":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Job identifier Permitted values: • 1: Read all system resources • 2: Read the system resources to which "ALARM_DQ"-/ "ALARM_D" = ev_id was allocated when EV_ID = ev_id was called. • 3: Read the system resources to which "ALARM_DQ"-/ "ALARM_D" = ev_id was allocated when CMP_ID = cmp_id was called. • 0: subsequent call
SI_ID	Input	DWORD	I, Q, M, D, L or constant	ID for the system resource(s) to be read Permitted values: • 0, if MODE=1 • Alarm number ev_id, if MODE=2 • ID cmp_id for identification of the subsystem, if MODE=3
RET_VAL	Return	INT	I, Q, M, D, L	Return value (error information or job status)
N_SI	Output	INT	I, Q, M, D, L	Number of system resources output in SYS_INT
SYS_INST	Output	ANY	D	Destination area for the read system resources.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
0001	Not all system resources could be read because the SYS_INST destination range you have selected is too short.
8081	(only with MODE=2 or 3). You have assigned the value 0 to SI_ID .
8082	(only with MODE=1). You have assigned one of 0 different values to SI_ID .
8083	(only with MODE=0). You have assigned SI_ID a value other than the value assigned for the preceding call with MODE=1 or 3.
8084	You have assigned an illegal value to MODE .
8085	READ_SI is already being processed in another OB.
8086	The destination area SYS_INST is too small for a system resource.

Error code (W#16#...)	Explanation
8087 Or 8092	The destination area SYS_INST does not exist in a DB or error in the ANY pointer.
8xyy	General error information See also: ALARM_D: Create permanently acknowledged PLC alarms (Page 5423)

Destination area parameter SYS_INST - Structure

The destination area for the read assigned system resources must be within a DB. You should appropriately define the destination area as a field of structures; a structure is comprised of the following:

Structure element	Data type	Description
SFC_NO	WORD	Number of the instruction allocated by the system resource
LEN	BYTE	Length of the structure in bytes, including SFC_NO and LEN: B#16#0C
SIG_STAT	BOOL	Signal state
ACK_STAT	BOOL	Acknowledgement status of the incoming event (positive edge)
EV_ID	DWORD	Alarm number
CMP_ID	DWORD	Subsystem ID

DEL_SI: Delete dynamically assigned system resources

Origin of dynamically allocated system resources when generating alarms with the "ALARM_DQ" and "ALARM_D" instructions

For alarm generation with "ALARM_DQ (Page 5425)" and "ALARM_D (Page 5423)", the operating system temporarily allocates memory space in the system memory.

For example, if you delete an FB with "ALARM_DQ (Page 5425)" or "ALARM_D (Page 5423)" calls that was present in the CPU, it may be the case that corresponding system resources remain permanently allocated. If you reload the FB with "ALARM_DQ (Page 5425)" / "ALARM_D (Page 5423)" calls, it may be that the "ALARM_DQ (Page 5425)" and "ALARM_D (Page 5423)" instructions will no longer be processed properly.

Description

You use the "DEL_SI" instruction to delete system resources that are currently in use.

"DEL_SI" has three possible operating modes, which are explained in the following table. You set the requested operating mode via the MODE parameter.

MODE	Which of the system resources allocated to "ALARM_DQ"/ "ALARM_D" will be deleted?
1	All (calling of "DEL_SI" using SI_ID:=0)
2	The system resources that were allocated to the EV_ID:=ev_id when "ALARM_DQ" (Page 5425) / ALARM_D (Page 5423) was called (DEL_SI called using SI_ID:=ev_id)
3	All system resources to which CMP_ID:=cmp_id was allocated when "ALARM_DQ" (Page 5425) / ALARM_D (Page 5423) was called (DEL_SI called with SI_ID:=cmp_id)

Parameter

The following table shows the parameters of the instruction "DEL_SI":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Job identifier Permitted values: • 1: delete all system resources • 2: Delete the system resources that were allocated to EV_ID = ev_id when "ALARM_DQ" (Page 5425) / "ALARM_D (Page 5423)" was called • 3: Delete the system resources that were allocated to CMP_ID = cmp_id when "ALARM_DQ" (Page 5425) / "ALARM_D (Page 5423)" was called
SI_ID	Input	DWORD	I, Q, M, D, L or constant	ID of the system resource(s) to be deleted Permitted values: • 0, if MODE=1 • Alarm number ev_id, if MODE=2 • ID cmp_id for identification of the subsystem, if MODE=3
RET_VAL	Return	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8081	(only with MODE=2 or 3). You have assigned the value "0" to SI_ID .
8082	(only with MODE=1). You have assigned one of "0" different values to SI_ID.
8084	You have assigned an illegal value to MODE .
8085	"DEL_SI" is currently being processed.
8086	Not all selected system resources could be deleted because at least one of them was being processed when "DEL_SI" was called.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

4.2.3.9 Diagnostics

RD_SINFO: Read current OB start information

Description

You read the start information with the instruction:

- of the OB last called that has not yet been completely executed, and
- of the startup OB started last.

There is no time stamp in either case. If the call is in OB 100 or OB 101 or OB 102, two identical start information messages will be returned.

Parameters

The following table shows the parameters of the instruction "RD_SINFO":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Error information
TOP_SI	Output	STRUCT	D, L	Start information of the current OB
START_UP_SI	Output	STRUCT	D, L	Start information of the startup OB last started

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

The instruction provides only general and not specific error information. The general error information and its evaluation are written to the "RET_VAL" output parameter.

See also: Evaluating errors with output parameter RET_VAL (Page 162).

Parameter TOP_SI and START_UP_SI

The output parameters TOP_SI and START_UP_SI are two identically set up structures. Their structure is provided in the following table.

Structure element	Data type	Description
EV_CLASS	BYTE	• Bits 0 to 3: Event ID • Bits 4 to 7: Event class
EV_NUM	BYTE	Event number
PRIORITY	BYTE	Number of the priority class (meaning of B#16#FE: OB not available or disabled or cannot be started in current operating mode)
NUM	BYTE	OB number
TYP2_3	BYTE	Data ID 2_3: Identifies the information entered in ZI2_3

Structure element	Data type	Description
TYP1	BYTE	Data ID 1: Identifies the information entered in ZI1
ZI1	WORD	Additional information 1
ZI2_3	DWORD	Additional information 2_3

Note

The structure elements listed in the above table are identical in content with the temporary tags of an OB.

Please note, however, that temporary tags of the individual OBs can have different names and different data types. Also note that the call interface of each OB includes additional information regarding the date and the time of the OB request.

Bits 4 to 7 of the EV_CLASS structure element contain the event class. The following values are possible here:

- 1: Start events from standard OBs
- 2: Start events from synchronous error OBs
- 3: Start events from asynchronous error OBs

The PRIORITY structure element supplies the priority class belonging to the current OB.

Apart from these two elements, NUM is also relevant. NUM contains the number of the current OB or the startup OB that was started last.

Example

OB 80 is the OB that was called last and that has not yet been completely processed and OB 100 is the start-up OB that was started last.

The following table shows the assignment between the structure elements of the TOP_SI parameter of the "RD_SINFO" instruction and the associated local tags of OB 80.

TOP_SI Structure element	Data type	OB 80 - Associated local tag	Data type
EV_CLASS	BYTE	OB80_EV_CLASS	BYTE
EV_NUM	BYTE	OB80_FLT_ID	BYTE
PRIORITY	BYTE	OB80_PRIORITY	BYTE
NUM	BYTE	OB80_OB_NUMBR	BYTE
TYP2_3	BYTE	OB80_RESERVED_1	BYTE
TYP1	BYTE	OB80_RESERVED_2	BYTE
ZI1	WORD	OB80_ERROR_INFO	WORD
ZI2_3	DWORD	OB80_ERR_EV_CLASS	BYTE
		OB80_ERR_EV_NUM	BYTE
		OB80_OB_PRIORITY	BYTE
		OB80_OB_NUM	BYTE

The following table shows the assignment between the structure elements of the START_UP_SI parameter of the "RD_SINFO" instruction and the associated local tags of OB 100.

START_UP_SI Structure element	Data type	OB 100 - Local tag	Data type
EV_CLASS	BYTE	OB100_EV_CLASS	BYTE
EV_NUM	BYTE	OB100_STRTUP	BYTE
PRIORITY	BYTE	OB100_PRIORITY	BYTE
NUM	BYTE	OB100_OB_NUMBR	BYTE
TYP2_3	BYTE	OB100_RESERVED_1	BYTE
TYP1	BYTE	OB100_RESERVED_2	BYTE
ZI1	WORD	OB100_STOP	WORD
ZI2_3	DWORD	OB100_STRT_INFO	DWORD

RDSYSST: Read system status list

Description

With the instruction you read a partial list of the system status lists (SZL) or a SZL partial list extract.

You start the read process by assigning the value "1" to the input parameter REQ when "RDSYSST" is called. If the read process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY. If BUSY has the value "1", reading is not yet complete.

Note

If you call the "RDSYSST" instruction in the diagnostic interrupt OB with the SZL-IDW#16#00B1 or W#16#00B2 or W#16#00B3 and access the module that initiated the diagnostic interrupt, the system status will be read immediately.

With "RDSYSST", only complete data records will be transferred.

System resources

If you start several asynchronous reading processes (the jobs with parameter SZL_IDW#16#00B4 and W#16#4C91 and W#16#4092 and W#16#4292 and W#16#4692 and possibly W#16#00B1 and W#16#00B3) one after the other at brief intervals, the operating system ensures that all the read jobs will be executed and that they do not interfere with each other. If the limits of the system resources are reached, this is indicated in RET_VAL. You can remedy this temporary error situation by repeating the job.

The maximum number of "simultaneously" active jobs of the instruction depends on the CPU.

Note

With S7-400 CPUs, the "RDSYSST" can be used to transfer a maximum of 432 bytes to the destination area.

Parameters

The following table shows the parameters of the "RDSYSST" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Trigger processing
SZL_ID	Input	WORD	I, Q, M, D, L or constant	SZL-ID of the partial list or partial list extract
INDEX	Input	WORD	I, Q, M, D, L or constant	Type or number of an object in a partial list.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the RET_VAL parameter contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	TRUE: Reading not yet complete.
SZL_HEADER	Output	STRUCT	D, L	See below.
DR	Output	ANY	I, Q, M, L, D	Destination area of the read SZL partial list or the read SZL partial list extract: • If you have only read out the header information of an SZL partial list, you should not evaluate DR but only SZL_HEADER. • Otherwise, the product of LENTHDR and N_DR indicates how many bytes were entered in DR.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters SZL_HEADER

The SZL_HEADER parameter is a structure defined as follows:

```
SZL_HEADER:   STRUCT
    LENTHDR:  WORD
    N_DR:     WORD
END_STRUCT
```

LENTHDR is the length of a data record of the SZL partial list or the SZL partial list extract.

- If you have only read out the header information of an SZL partial list, N_DR contains the number of data records belonging to it.
- Otherwise, N_DR contains the number of data records transferred to the destination area.

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
0081	Result field too short. (Nevertheless as many data records as possible are supplied. The SZL-Header indicates this number.)
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	Result field too short (not enough space for one data record).
8082	SZL_ID is incorrect or is unknown in the CPU or instruction.
8083	INDEX incorrect or not permitted
8085	Due to a problem in the system, information is not currently available (for example, due to a lack of resources).
8086	The data record cannot be read due to a system error (bus, modules, operating system).
8087	Data record cannot be read because the module does not exist or does not acknowledge.
8088	Data record cannot be read because the actual module identifier is different from the expected module identifier.
8089	Data record cannot be read because the module is not capable of diagnostics or the data record is not supported.
80A2	DP protocol error (layer 2 error) (temporary error)
80A3	DP protocol error with user interface/user (temporary error)
80A4	Communication problem on communication bus (error occurs between the CPU and the external DP interface module)
80C5	Distributed I/O not available (temporary error).
80C6	Data record transfer stopped due to priority class abort (restart or background)
80D2	Data record cannot be read because the module is not capable of diagnostics.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

Parameters SZL_ID

Note

Partial lists of S7-CPUs that can be read out

- The partial lists for S7-300 that can be read out via "RDSYSST" are provided in the corresponding operation list.
- The partial lists for S7-400 that can be read out via "RDSYSST" are provided in the following table.

SZL_ID (W#16#...)	Partial list	INDEX (W#16#...)
Module ID		
0111 (Page 5455)	An identification data record	
	ID of the module	0001
	Identification of basic hardware	0006
	Identification of the basic firmware	0007
CPU characteristics		
0012 (Page 5456)	All characteristics	irrelevant
0112 (Page 5456)	Characteristics of one group	
	MC7 processing unit	0000
	Time system	0100
	System behavior	0200
	MC7 language description	0300
	Availability of the "C_DIAG (Page 5510)" instruction	0400
0F12 (Page 5456)	Only SZL partial header information	irrelevant
User memory areas		
0113 (Page 5458)	A data record for the memory area specified	
	Work memory	0001
System areas		
0014 (Page 5459)	Data records of all system areas	irrelevant
0F14 (Page 5459)	Only SZL partial header information	irrelevant
Block types		
0015 (Page 5460)	Data records of all block types	irrelevant
Identification of one component		
001C (Page 5461)	Identification of all components	irrelevant

SZL_ID (W#16#...)	Partial list	INDEX (W#16#...)
011C (Page 5461)	Identification of one component	
	Name of the automation system	0001
	Name of the module	0002
	System ID of the module	0003
	Copyright entry	0004
	Serial number of the module	0005
	Module type name	0007
	Serial number of the memory card	0008
	Manufacturer and profile of a CPU module	0009
	Location designation of a module	000B
0F1C (Page 5461)	Only SZL partial header information	irrelevant
Interrupt status		
0222 (Page 5468)	Data record for indicated interrupt	OB number
Assignment of process images to CPUs		
0025 (Page 5471)	Assignment of all process images to OBs	irrelevant
0125 (Page 5471)	Assignment of a process image partition to the corre-sponding OB	Process image partition no.
0225 (Page 5471)	Assignment of an OB to the corresponding process image partitions	OB no.
0F25 (Page 5471)	Only SZL partial header information	irrelevant
Communication status data		
0132 (Page 5473)	Status data for one communication unit	
	Diagnostics	0005
	Time system	0008
0232 (Page 5473)	Status data for one communication unit	
	CPU protection level and operator control settings	0004
Status of the module LEDs (cannot be read out for all CPUs).		
0174 (Page 5479)	Status of an LED	LED ID
DP master system information		
0090 (Page 5481)	Information regarding all DP master systems known to the CPU	0000
0190 (Page 5481)	Information regarding one DP master system	DP master system ID
0F90 (Page 5481)	Only SZL partial header information	0000
Module status information (a maximum of 27 data records is supplied)		

SZL_ID (W#16#...)	Partial list	INDEX (W#16#...)
0091 (Page 5482)	Status information of all modules/sub-modules inserted	irrelevant
0191 (Page 5482)	Status information of all non-deactivated modules/racks with incorrect module ID	irrelevant
0291 (Page 5482)	Module status information of all faulty and non-deactivated modules	irrelevant
0391 (Page 5482)	Module status information of all unavailable modules	irrelevant
0591 (Page 5482)	Module status information of all sub-modules of the host module	irrelevant
0991 (Page 5482)	Module status information of one DP master system	DP master system ID
0C91 (Page 5482)	Module status information of a module in a central configuration or connected to an integrated DP interface module or a PROFINET interface module (integrated or external)	Logical base address
4C91 (Page 5482)	Module status information of a module connected to an external DP interface module	Logical base address
0D91 (Page 5482)	Module status information of all modules in the rack/station specified (DP or PROFINET)	Rack or DP master system ID and station number or station number and the last two digits of the PNIO subsystem ID
0E91 (Page 5482)	Module status information of all assigned modules	irrelevant
0F91 (Page 5482)	Only SZL partial header information	irrelevant
Rack/station status information		
0092 (Page 5488)	Expected status of the rack in the central configuration/stations of a DP master system.	0 / DP master system ID
4092 (Page 5488)	Expected status of the stations of a DP master system connected via an external DP interface module.	DP master system ID
0192 (Page 5488)	Activation status of the stations of a DP master system that is connected via an integrated DP interface module	DP master system ID
0292 (Page 5488)	Current status of the rack in the central configuration / stations of a DP master system	0 / DP master system ID
4292 (Page 5488)	Current status of the stations of a DP master system that is connected via an external DP interface module	DP master system ID
0392 (Page 5488)	Status of the backup batteries in a rack/module rack of a CPU after at least one battery has failed.	0
0492 (Page 5488)	Status of the overall battery backup of all racks/module racks of a CPU.	0
0592 (Page 5488)	Status of the 24 V power supply to all racks/module racks of a CPU	0

SZL_ID (W#16#...)	Partial list	INDEX (W#16#...)
0692 (Page 5488)	Diagnostics status of the expansion racks in the central configuration/ of the stations of a DP master system connected via an integrated DP interface module	0 / DP master system ID
4692 (Page 5488)	Diagnostics status of the stations of a DP master system connected via an external DP interface module	DP master system ID
Rack/station status information		
0094 (Page 5491)	Expected status of the rack in the central configuration / stations of an IO controller system.	0 / PNIO subsystem ID
0194 (Page 5491)	Activation status of a station of an IO controller system that is configured and deactivated	PNIO subsystem ID
0294 (Page 5491)	Actual status of the rack in the central configuration / stations of an IO controller system	0 / PNIO subsystem ID
0694 (Page 5491)	Diagnostic status of the expansion units in the central configuration / stations of an IO controller system.	0 / PNIO subsystem ID
0794 (Page 5491)	Maintenance status of the central rack / stations of an IO controller system.	0 / PNIO subsystem ID
0F94 (Page 5491)	Only header information	-
Extended DP master system information		
0195 (Page 5493)	Extended information about a DP master system	DP master system ID
0F95 (Page 5493)	Only SZL partial header information	0000
Module status information PROFINET IO and PROFIBUS DP		
0696 (Page 5494)	Module status information of all sub-modules of a specified module (only with PROFINET IO on an integrated interface module).	Address with I/O ID
0C96 (Page 5494)	Module status information of a module/sub-module in a central configuration or on an integrated PROFIBUS DP or PROFINET interface module (integrated or external).	Start address with I/O ID
Diagnostics buffer (a maximum of 21 data records is supplied)		
00A0 (Page 5501)	All entries that can be supplied in the currently active operating mode	irrelevant
01A0 (Page 5501)	The most recent entries, the number is specified in the index	Number
0FA0 (Page 5501)	Only SZL partial header information	irrelevant
Diagnostic data on modules		
00B1 (Page 5502)	The first four diagnostics bytes of one module (data record 0)	Logical base address
00B2 (Page 5504)	All diagnostic data of one module (220 bytes, data record 1) (no DP module)	Rack, slot

SZL_ID (W#16#...)	Partial list	INDEX (W#16#...)
00B3 (Page 5505)	All diagnostic data of one module (220 bytes, data record 1)	Logical base address
00B4 (Page 5505)	Diagnostic data of a DP slave	Configured diagnostics address

See also

Difference between synchronous and asynchronous instructions (Page 4186)

System status list

Overview of the system status list (SZL)

Definition: System status list

The system status list (SZL) describes the current status of an automation system. The contents of the SZL (system status list) can only be read using information functions, contents cannot be modified. The partial lists are virtual lists, in other words, they are only created by the operating system of the CPUs when specifically requested.

You can only read out one partial list using "RDSYSST (Page 5444)". This chapter describes the partial lists of the system status list that can be read with "RDSYSST (Page 5444)" in the user program, and that have information relating to

- CPUs or to
- modules whose partial lists are not module-specific (for example, SZL IDs W#16#00B1, W#16#00B2, W#16#00B3).

Module-specific partial lists, for example, for CPs and FMs are included in the descriptions of the particular modules.

Contents

The system status lists contain information about the following:

- System data
- Module status data in the CPU
- Diagnostics data on modules
- Diagnostics buffer

System data

System data are fixed or parameter- assigned characteristic data of a CPU. They provide information about the following:

- The configuration of the CPU
- The status of the priority classes
- Communication

Module status data

Module status information describes the current status of the components monitored by the system diagnostics.

Diagnostics data on modules

The modules with diagnostics capabilities assigned to a CPU have diagnostics data that are stored directly on the module.

Diagnostics buffer

The diagnostics buffer contains diagnostics entries in the order in which they occur.

Structure of a partial SZL list

Basics

You can read a partial list or a partial list extract using "RDSYSST (Page 5444)". You specify what you want to read using the parameters SZL_ID (System status list ID) and INDEX.

Structure

A partial list consists of the following:

- A header and
- the data records.

Header

The header of a partial list consists of the following:

- SZL-ID
- INDEX
- Length of a data record of the partial list in bytes
- Number of data records contained in the partial list.

Index

With certain partial lists or partial list extracts an object type ID or an object number must be specified. The index is used for this purpose. If it is not required for the information, its contents are irrelevant.

Data records

A data record in a partial list has a specific length. This depends on the information in the partial list. How the data words in a data record are used also depends on the particular partial list.

SZL-ID

SZL-ID (System status list ID)

Every partial list within the system status list has a number. You can output a complete partial list or an extract from it. The possible partial list extracts are predefined and are identified by a number. The SZL-ID consists of the number of the partial list, the number of the partial list extract and the module class.

Structure

The SZL-ID is one word long. The meaning of the bits in the SZL ID is as follows:

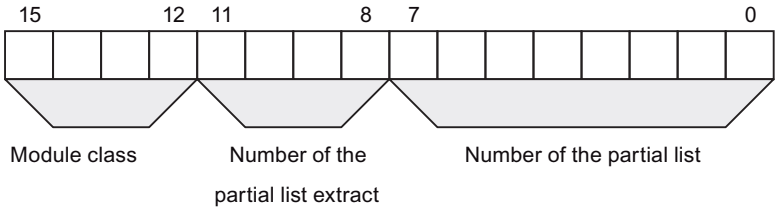

Figure 4-23 Structure of SZL-ID

Module class

Examples of module classes:

Module class	Coding (binary)
CPU	0000
IM	0100
FM	1000
CP	1100

Number of the partial list extract

The numbers of the partial list extracts and their meaning depend on the particular system status list to which they belong. With the number of the partial list extract, you specify which subset of a partial list you want to read.

Number of the partial List

Using the number of the partial list, you specify which partial list of the system status list you want to read out.

Possible SZL partial lists

Subset

Any one module only has a subset of all the possible partial lists. Which partial lists are available depends on the particular module.

Possible SZL partial lists

The following table lists all possible partial lists with the number contained in the SZL-ID (system status list ID).

Partial list	SZL-ID
Module ID	W#16#xy11 (Page 5455)
CPU characteristics	W#16#xy12 (Page 5456)
User memory areas	W#16#xy13 (Page 5458)
System areas	W#16#xy14 (Page 5459)
Block types	W#16#xy15 (Page 5460)
Identification of one component	W#16#xy1C (Page 5461)
Interrupt status	W#16#xy22 (Page 5468)
Assignment of partial process images and OBs	W#16#xy25 (Page 5471)
Communication: Status data	W#16#xy32 (Page 5473)
Status of the module LEDs	W#16#xy74 (Page 5479)
DP master system information	W#16#xy90 (Page 5481)
Module status information	W#16#xy91 (Page 5482)
Rack /station status information	W#16#xy92 (Page 5488)
Rack /station status information	W#16#xy94 (Page 5491)
Extended DP master system information	W#16#xy95 (Page 5493)

Partial list	SZL-ID
Module status information PROFINET IO and PROFIBUS DP	W#16#xy96 (Page 5494)
Diagnostics buffer	W#16#xyA0 (Page 5501)
Module diagnostic information (data record 0)	W#16#00B1 (Page 5502)
Module diagnostic information (data record 1), physical address	W#16#00B2 (Page 5504)
Module diagnostic information (data record 1), logical address	W#16#00B3 (Page 5505)
Diagnostic data of a DP slave	W#16#00B4 (Page 5505)

SZL-ID W#16#xy11 - module identification

Purpose

You obtain the module identification of this module via the partial list with the SZL-ID (system status list ID) W#16#xy11.

Header

The header of the SZL with the SZL-IDW#16#xy11 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0111: A single identification data record
INDEX	Number of a particular data record W#16#0001: ID of the module W#16#0006: Identification of the basic hardware W#16#0007: Identification of the basic firmware
LENTHDR	W#16#001C: One data record is 14 words long (28 bytes)
N_DR	Number of data records

Data record

A data record of the SZL with the SZL-IDW#16#xy11 has the following structure:

Name	Length	Meaning
Index	1 word	Number of an identification data record
MRPC	20 bytes	Machine-Readable Product Code With INDEX W#16#0007: Reserved With INDEXW#16#0001 and W#16#0006: Machine readable product code of the module; string consists of 19 characters and a blank (20H); such as for CPU 314: "6ES7 314-0AE01-0AB0"
ModuleType	1 word	Reserved
OutMod1	1 word	With INDEXW#16#0001: Output status of the module With INDEXW#16#0006 and W#16#0007: "V" and first number of the version ID
OutMod2	1 word	With INDEXW#16#0001: Reserved With INDEXW#16#0006 and W#16#0007: remaining numbers of the version ID

SZL-ID W#16#xy12 - CPU characteristics

Purpose

CPU modules have different characteristics depending on the hardware being used. An ID is assigned to each of these characteristics. You obtain the characteristics of this module via the partial list with the SZL-ID (system status list ID) W#16#xy12.

Header

The header of the partial list with the SZL-IDW#16#xy12 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0012: All characteristics W#16#0112: Characteristics of a group; you specify the group in the INDEX parameter. W#16#0F12: Only SZL partial header information
INDEX	Group W#16#0000: MC7 processing unit W#16#0100: Time system W#16#0200: System behavior W#16#0300: MC7 language description of the CPU W#16#0400: Availability of the "C_DIAG (Page 5510)" instruction
LENTHDR	W#16#0002: One data record is one word long (2 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xy12 is one word long. An identifier is entered for each characteristic. A characteristic identifier is one word long.

Note

All data records relevant to your CPU will be output. They follow each other in sequence with no gaps.

Characteristic identifier

The following table lists all the characteristic identifiers.

Identifier	Meaning
W#16#0000 - 00FF	MC7 processing unit (group with index 0000)
W#16#0001	MC7 processing generating code
W#16#0002	MC7 interpreter
W#16#0100 - 01FF	Time system (group with index 0100)
W#16#0101	1 ms resolution
W#16#0102	10 ms resolution
W#16#0103	No real time clock
W#16#0104	BCD time-of-day format
W#16#0105	All time-of-day functions (set time-of-day, set and read time-of-day, time-of-day synchronization: time-of-day slave and time-of-day master)
W#16#0106	OB_RT is available
W#16#0200 - 02FF	System response (group with index 0200)
W#16#0201	Capable of multiprocessor mode
W#16#0202	Cold restart, warm restart and hot restart possible
W#16#0203	Cold restart and hot restart possible
W#16#0204	Warm restart and hot restart possible
W#16#0205	Only warm restart possible
W#16#0206	New distributed I/O configuration is possible during RUN by using predefined resources
W#16#0208	For taking motion control functionality into account
W#16#0300 - 03FF	MC7 language description of the CPU (group with index 0300)
W#16#0301	Reserved
W#16#0302	All 32 bit fixed-point instructions
W#16#0303	All floating-point instructions
W#16#0304	sin, asin, cos, acos, tan, atan, sqr, sqrt, ln, exp
W#16#0305	Accumulator 3/accumulator 4 with associated commands (ENT, PUSH, POP, LEAVE)
W#16#0306	Master control relay instructions
W#16#0307	Address register 1 exists with corresponding instructions
W#16#0308	Address register 2 exists with corresponding instructions
W#16#0309	Operations for area-overlapping addressing

Identifier	Meaning
W#16#030A	Operations for area-internal addressing
W#16#030B	All memory-indirect addressing instructions for bit memory (M)
W#16#030C	All memory-indirect addressing instructions for data blocks (DB)
W#16#030D	All memory-indirect addressing instructions for DI
W#16#030E	All memory-indirect addressing instructions for local data
W#16#030F	All instructions for parameter transfer in instructions
W#16#0310	Memory bit edge instructions for I
W#16#0311	Memory bit edge instructions for Q
W#16#0312	Memory bit edge instructions for bit memory
W#16#0313	Memory bit edge instructions for data blocks (DB)
W#16#0314	Memory bit edge instructions for DI
W#16#0315	Memory bit edge instructions for local data (LD)
W#16#0316	Dynamic evaluation of the ERAB bit
W#16#0317	Dynamic local data area with the corresponding instructions
W#16#0318	Reserved
W#16#0319	Reserved
W#16#0401	Reserved
W#16#0402	Reserved
W#16#0403	"C_DIAG (Page 5510)" is available

SZL-ID W#16#xy13 - user memory areas

Purpose

You obtain information on the memory areas of the module via the partial list with the SZL-ID (system status list ID) W#16#xy13.

Header

The header of the partial list with the SZL-IDW#16#xy13 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0113: Data record for a memory area You use the INDEX parameter to specify the memory area.
INDEX	Specification of a memory area (only with SZL-IDW#16#0113) W#16#0001: Work memory
LENTHDR	W#16#0024: One data record is 18 words long (36 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xy13 has the following structure:

Name	Length	Meaning
Index	1 word	Index of a memory area W#16#0001: Work memory
Code	1 word	Memory type: • W#16#0001: Volatile memory (RAM) • W#16#0002: Non-volatile memory (FEPROM) • W#16#0003: Mixed memory (RAM and FEPROM)
Size	2 words	Total size of the selected memory (total of Area1 and Area2)
Mode	1 word	Logical mode of the memory • Bit 0: volatile memory area • Bit 1: non-volatile memory area • Bit 2: mixed memory area for work memory • Bit 3: code and data separated • Bit 4: code and data together
Granularity	1 word	Always has the value "0"
Area1	2 words	Size of the volatile memory area in bytes.
Reserved1	2 words	Size of the volatile memory area being used
Block1	2 words	Largest free block in the volatile memory area If "0": no information available or cannot be ascertained.
Area2	2 words	Size of the non-volatile memory area in bytes
Reserved2	2 words	Size of the non-volatile memory area being used
Block2	2 words	Largest free block in the volatile memory area If "0": no information available or cannot be ascertained.

SZL-ID W#16#xy14 - system areas

Purpose

You obtain information on the system areas of the module via the partial list with the SZL-ID (system status list ID) W#16#xy14.

Header

The header of the partial list with the SZL-IDW#16#xy14 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0014: All system areas of a module W#16#0F14: Only SZL partial header information
INDEX	irrelevant

Contents	Meaning
LENTHDR	W#16#0008: One data record is 4 words long (8 bytes)
N_DR	Number of data records You must at least assign a number of 9 data records. If you select a destination area that is too small, "RDSYSST (Page 5444)" does not provide a data record.

Data record

A data record of the partial list with the SZL-IDW#16#xy14 has the following structure:

Name	Length	Meaning
Index	1 word	Index of the system area • W#16#0001: Process image input (number in bytes) • W#16#0002: Process output image (number in bytes) • W#16#0003: Bit memory (number in bits) **Note:** This index is only provided by the CPU, where the number of flags can be shown in one word. If your CPU does not provide this value, you must evaluate the index W#16#0008 . • W#16#0004: Times (number) • W#16#0005: Counters (number) • W#16#0006: Number of bytes in the logical address area • W#16#0007: Local data (total local data area of the CPU in bytes) **Note:** This index is only provided by CPUs, where the length of the entire local data area can be shown in one word. If your CPU does not provide this value, you must evaluate the index W#16#0009. • W#16#0008: Bit memory (number in bytes) • W#16#0009: Local data (total local data area of the CPU in kilobytes)
code	1 word	Memory type • W#16#0001: Volatile memory (RAM) • W#16#0002: Non-volatile memory (FEPROM) • W#16#0003: Mixed memory (RAM and FEPROM)
number	1 word	Number of elements of the system area
retentive	1 word	Number of retentive elements

SZL-ID W#16#xy15 - block types

Purpose

If you read the partial list with the SZL-ID (system status list ID) W#16#xy15, you obtain the block types that exist on the module.

Header

The header of the partial list with the SZL-IDW#16#xy15 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0015: Data records of all block types of a module
INDEX	irrelevant
LENTHDR	W#16#000A: One data record is 5 words long (10 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xy15 has the following structure:

Name	Length	Meaning
Index	1 word	Block type number • W#16#0800: OB • W#16#0A00: DB • W#16#0B00: SDB • W#16#0C00: FC • W#16#0E00: FB
MaxNumber	1 word	Maximum number of blocks of the type • Foi OBs: max. possible number of OBs for a CPU • For DBs: max. possible number of DBs including DB0 • For SDBs: max. possible number of SDBs including SDB2 • For FCs and FBs: max. possible number of loadable blocks
MaxLenght	1 word	Maximum total size of the object to be loaded in kilobytes
MaxWorkMem	2 words	Maximum length of the work memory part of a block in bytes

SZL-ID W#16#xy1C - component identification

Purpose

If you read the partial list with the SZL-ID (system status list ID) W#16#xy1C, you can identify the CPU or the automation system.

Header

The header of the partial list with the SZL-IDW#16#xy1C is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract
	• W#16#001C: Identification of all components • W#16#011C: Identification of one component • W#16#0F1C: Only SZL partial header information
INDEX	• Identification of components for a partial list extract with the SZL-IDW#16#011C and W#16#031C – W#16#0001: Name of the automation system – W#16#0002: Name of the module – W#16#0003: System ID of the module – W#16#0004: Copyright entry – W#16#0005: Serial number of the module – W#16#0007: Module type name – W#16#0008: Serial number of the memory card No data record is supplied for modules in which a memory card cannot be inserted. – W#16#0009: Manufacturer and profile of a CPU module – W#16#000A: OEM ID of a module (S7-300 only) – W#16#000B: Location designation of a module – W#16#000C: Serial number of Sync module 1 – W#16#000D: Serial number of Sync module 2 • Rack no. for the partial list extract with the SZL-IDW#16#021C (Byte0: Rack no., byte1: B#16#00)
LENTHDR	W#16#0022: One data record is 17 words long (34 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xy1C has the following structure:

INDEX = W#16#0001

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0001 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#01
name	12 words	Name of the automation system (max. 24 characters; when using shorter names, the gaps are filled with B#16#00)
res	4 words	Reserved

INDEX = W#16#0002

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0002 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#02
name	12 words	Name of the module (max. 24 characters; when using shorter names, the gaps are filled with B#16#00)
res	4 words	Reserved

INDEX = W#16#0003

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0003 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#03
tag	16 words	Plant identification of the module (max. 32 characters; when using shorter system IDs the gaps will be filled with B#16#00)

INDEX = W#16#0004

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0004 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#04
copyright	13 words	Constant character sequence "Original Siemens Equipment"
res	3 words	Reserved

INDEX = W#16#0005

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0005 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#05
serialn	12 words	Serial number of the module; character string with max. length of 24 characters. Shorter names are filled with B#16#00. Note: The serial numbers for SIMATIC components are unique worldwide. It is permanently affixed to the CPU hardware, in other words it remains unchanged when the firmware is updated.
res	4 words	Reserved

INDEX = W#16#0007

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0007 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#07
cputypname	16 words	Module type name; character string with a max. length of 32 characters. Shorter names are filled with B#16#00.

INDEX = W#16#0008

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0008 • For the partial list extracts W#16#011C, W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#08
sn_mc/mmc	16 words	Serial number of the Memory Card/Micro Memory Card; character string with a max. length of 32 characters. Shorter names are filled with B#16#00. • Siemens serial number: only serial number, no index • Product serial number (PSN) of an S7 Micro Memory Card: "MMC" plus serial number (PSN) • Serial number of an S7 Memory Card: "MC" plus serial number The character string ends immediately after "MMC" or "MC" if no Memory Card is installed.

INDEX = W#16#0009

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0009 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#09
manufacturer_id	1 word	See PROFIBUS Profile Guidelines Part 1, Identification & Maintenance Functions
profile_id	1 word	See PROFIBUS Profile Guidelines Part 1, Identification & Maintenance Functions
profile_specific_typ	1 word	See PROFIBUS Profile Guidelines Part 1, Identification & Maintenance Functions
res	13 words	Reserved

INDEX = W#16#000A

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#000A • For the partial list extracts: W#16#021C and W#16#031C: Byte0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#0A
oem_copy-right_string	13 words	OEM copyright ID; character string with a max. length of 20 characters. Shorter names are filled with B#16#00.
oem_id	1 word	OEM-ID. This is assigned by Siemens.
oem_add_id	2 words	OEM-ID supplemental ID. This can be assigned by the user.

INDEX = W#16#000B

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#000B • For the partial list extracts: W#16#021C and W#16#031C: Byte0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#0B
loc_id	16 words	Location designation; character string with a max. length of 32 characters. Shorter names are filled with B#16#00.

INDEX = W#16#000C

Name	Length	Meaning
index	1 word	• Byte0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 • Byte 1: Component ID: B#16#0C
mlfb	5 words	10th to 19th character of the order number of Sync module 1
fill	1 word	Four spaces (ASCII)
es	1 word	Product version (ASCII), e.g. "01"
fill	1 byte	Two spaces (ASCII)
vendor_sn	17 bytes	Serial number

Note

If Sync module 1 is not available or cannot be identified, the data record will be filled in accordance with the "index" tag with B#16#00.

INDEX = W#16#000D

Name	Length	Meaning
index	1 word	• Byte0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 • Byte 1: Component ID: B#16#0D
mlfb	5 words	10th to 19th character of the order number of Sync module 2
fill	1 word	Four spaces (ASCII)
es	1 word	Product version (ASCII), e.g. "01"
fill	1 byte	Two spaces (ASCII)
vendor_sn	17 bytes	Serial number

Note

If Sync module 2 is not available or cannot be identified, the data record will be filled in accordance with the "index" tag with B#16#00.

SZL-ID W#16#xy22 - interrupt status

Purpose

The partial list with the SZL-ID (system status list ID) W#16#xy22 contains information about the current status of interrupt processing and interrupt generation in the module.

Header

The header of the partial list with the SZL-IDW#16#xy22 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0222 Data record for specified interrupt The interrupt (OB no.) is specified in the INDEX parameter.
INDEX	OB no. or interrupt class (for SZL-IDW#16#0222) W#16#0000: Free cycle W#16#000A: Time-of-day interrupt W#16#0014: Time-delay interrupt W#16#001E: Cyclic interrupt W#16#0028: Hardware interrupt W#16#0032: DP interrupt W#16#003C: Synchronous cycle interrupt W#16#0050: Asynchronous error interrupt W#16#005A: Background W#16#0064: Startup W#16#0078: Synchronous error interrupt
LENTHDR	W#16#001C: One data record is 14 words long (28 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xy22 has the following structure:

Name	Length	Meaning
Info	10 words	Start info for the given OB, with following exceptions: • OB 1 provides the current minimum (in bytes 8 and 9) and maximum cycle time (in bytes 10 and 11) (time base: ms, byte count begins at 0). • When a job is active for a time-delay interrupt, bytes 8 and 11 (byte count begins at 0) get the remaining time in ms left of the delay time set as a parameter. • OB 80 contains the configured minimum (in bytes 8 and 9) and maximum cycle time (in bytes 10 and 11) (time base: ms, byte count begins at 0). • Error interrupts without the current information • Interrupts contain the status info from the current parameter settings of the interrupt source. • In the case of synchronous errors, the priority class B#16#7F is entered if the OBs have not been processed yet; otherwise, the priority class of the last call will be entered. • If an OB has several start events and these have not yet occurred at the information time, then event no. W#16#xyzz returned with: – x: Event class – zz: Smallest defined number of the group – y: undefined Otherwise, the number of the last start event that occurred is used.
AlarmStatus1	1 word	Processing identifiers: • Bit 0: Interrupt event is caused by parameters – = 0: Enabled – = 1: disabled • Bit 1: Interrupt event as per "DIS_IRT (Page 5388)" – = 0: Not disabled – = 1: disabled • Bit 2 = 1: Interrupt source is active (generate job is present for time interrupts (time-of-day interrupt OB started, time-delay interrupt OB started, cyclic interrupt OB was configured) • Bit 4: Interrupt OB – = 0: Is not loaded – = 1: Is loaded • Bit 5: Interrupt OB is by TIS – = 1: disabled • Bit 6: Entry in diagnostic buffer – = 1: disabled

Name	Length	Meaning
AlarmStatus2	1 word	Reaction with non-loaded/disabled OB • Bit 0 = 1: Disable interrupt source • Bit 1 = 1: Generate interrupt event error • Bit 2 = 1: CPU goes into STOP mode • Bit 3 = 1: Only discard interrupt
AlarmStatus3	2 words	Discarded by TIS functions: Bit no. x set means: the event number that is greater by x than the smallest event number of the affected OB is discarded by the TIS function.

SZL ID W#16#xy25 - assignment of process image partitions to OBs

Purpose

The partial list with the SZL-ID (system status list ID) W#16#xy25 shows you how process image partitions are assigned to the OBs.

This list provides information about

- Process image partitions you have assigned to specific OBs for update by the system
- Process image partitions you have assigned to specific clock synchronization interrupt OBs (OBs 61 to 64). The partial process image is updated here by calling the "SYNC_PI (Page 5221)" and "SYNC_PO (Page 5222)".
  The assignment of DP master systems to the synchronous cycle interrupt OBs is obtained via SZL W#16#xy95 (Page 5493).

Header

The header of the partial list with the SZL-IDW#16#xy25 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0025: Assignment of all process image partitions to the OBs in the CPU • W#16#0125: Assignment of a process image partition to the corresponding OB Specify the process image partition ID in the INDEX parameter. • W#16#0225: Assignment of the OB to the corresponding process image partitions Specify the OB number in the INDEX parameter. Note: The clock synchronization interrupt OBs (OBs 61 to 64) are the only ones you can assign to multiple process image partitions. • W#16#0F25: Only SZL partial header information
INDEX	• For SZL-ID W#16#0025: irrelevant • For SZL-ID W#16#0125: Process image partition no. • For SZL-ID W#16#0225: OB no. • For SZL-ID W#16#0F25: irrelevant

Contents	Meaning
LENTHDR	W#16#0004: One data record is 2 words long (4 bytes)
N_DR	Number of data records

Data record

A data record of the partial list extract with the SZL-IDW#16#xy25 has the following structure:

Name	Length	Meaning
PPI*_nr	1 byte	Process image partition no.
PPI*_use	1 byte	Type of assignment between process image partitions and the OB: • Bit 0 = 1: The process image partition of the inputs is assigned to the specified OB for update by the system. • Bit 1 = 1: The process image partition of the outputs is assigned to the specified OB for update by the system. • Bit 2 = 1: The process image partition input table is assigned to the specified synchronous cycle interrupt OB. It can be updated in this OB by calling "SYNC_PI (Page 5221)". • Bit 3 = 1: The process image partition output table is assigned to the specified synchronous cycle interrupt OB. It can be updated in this OB by calling "SYNC_PO (Page 5222)". • Bits 4 to 7: 0
ob_nr	1 byte	OB no.
res	1 byte	Reserved
*PPI: Parial Process Image		

Partial list extracts

- Partial list extract with SZL-ID = W#16#0025:
  The data records of all partial process images you have assigned to an OB in your configuration are returned in ascending order. For partial process images without OB assignment, ob_nr has the value zero. No data record is returned for partial process images.
- Partial list extract with SZL-ID = W#16#0125:
  A data record is returned if you have assigned the addressed process image partition to an OB in your configuration. No data record is returned if you have not assigned an OB.

Note

OB 1 is permanently assigned to partial process image 0. You always get a data record for the information about partial process image 0.

- Partial list extract with SZL-ID = W#16#0225:
  A data record is returned for each partial process image that is assigned to the addressed OB. No data record is returned if you have not assigned the addressed process image partition to an OB in your configuration.

  Note

  Clock synchronization interrupt OBs can be assigned multiple process image partitions. If this is the case, several data records will be returned.

- Partial list extract with SZL-ID = W#16#0F25:
  The maximum possible number of data records is returned as number.

Example of the meaning of data records

Call parameters of "RDSYSST (Page 5444)"	Returned variables	Explanation
SZL_ID = W#16#0125, INDEX = W#16#0008	PPI*_nr = B#16#08, PPI*_use = B#16#03, ob_nr = B#16#15	A data record is returned. The input and output process image 8 are assigned to OB 21 for system-side process image update.
SZL_ID = W#16#0125, INDEX = W#16#0009	–	No data record is returned. Thus: process image partition 9 is not assigned to any OB.
SZL_ID = W#16#0225, INDEX = W#16#003D	PPI*_nr = B#16#0A, PPI*_use = B#16#C0, ob_nr = B#16#3D PPI*_nr = B#16#10, PPI*_use = B#16#C0, ob_nr = B#16#3D	Two data records are returned. The input and output process images 10 and 16 are assigned to OB 61. These can be updated in OB 61 by calling "SYNC_PI (Page 5221)" and "SYNC_PO (Page 5222)".
SZL_ID = W#16#0225, INDEX = W#16#0001	PPI*_nr = B#16#00, PPI*_use = B#16#03, ob_nr = B#16#01	A data record is returned. The input and output process image 0 are assigned to OB 1. They are updated by the system.
*PPI: Parial Process Image		

SSL ID W#16#xy32 - Communication status data

SZL-ID W#16#xy32 - communication status data

Purpose

If you read the partial list with SZL-IDW#16#xy32 you obtain the communication status data of the module.

Header

The header of the partial list with the SZL-IDW#16#xy32 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0132: Status data for one communication section (always only one data record). Specify the communication section with the INDEX parameter. • W#16#0232: status data for one communication unit. Specify the communication section with the INDEX parameter.
INDEX	Communication section • For SZL-IDW#16#0132: – W#16#0005 Diagnostics – W#16#0008 Time system – W#16#000B Time system – W#16#000C Time system • For SZL-IDW#16#0232: W#16#0004 CPU protection level, operator switch settings and version IDs / checksums
LENTHDR	W#16#0028: One data record is 20 words long (40 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#0132 is always 20 words long. The data records have different contents. The contents depend on the INDEX parameter, in other words, on the communication section to which the data record belongs.

Data record of the partial list extract with SZL-ID W#16#0132 index W#16#0005

Contents

The partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0005 contains information about the diagnostics status of the module.

Data record

A data record of the partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0005 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0005: Diagnostics
ext	1 word	Extended functions • 0: No • 1: Yes

Name	Length	Meaning
send	1 word	Automatic send • 0: No • 1: Yes
possible	1 word	Sending user-defined diagnostic messages currently possible • 0: No • 1: Yes
res	16 words	Reserved

Data record of the partial list extract with SZL-ID W#16#0132 index W#16#0008

Contents

The partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0008 contains information about the status of the module time system.

Data record

A data record of the partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0008 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0008: Time system status
cycle	1 word	Reserved
corr	1 word	Correction factor for the time
clock 0	1 word	Runtime meter 0: time in hours
clock 1	1 word	Runtime meter 1: time in hours
clock 2	1 word	Runtime meter 2: time in hours
clock 3	1 word	Runtime meter 3: time in hours
clock 4	1 word	Runtime meter 4: time in hours
clock 5	1 word	Runtime meter 5: time in hours
clock 6	1 word	Runtime meter 6: time in hours
clock 7	1 word	Runtime meter 7: time in hours
time	4 words	Current date and time (format: date_and_time)
rtm _0	1 byte	Runtime meter Bit x: Overflow of runtime meter x, 0 ≤ x ≤ 7 (Bit = 1: run-time meter is busy)
res	1 byte	Reserved
rtmOV _0	1 byte	Runtime meter Overflow Bit x: Overflow of runtime meter x, 0 ≤ x ≤ 7 (Bit = 1: Overflow)
res	1 byte	Reserved
status	1 word	Time status (for bit assignment, see below)
res	3 bytes	Reserved
status_valid	1 byte	Validity of variable status: B#16#01: status valid

Time-of-day status (status)

Bit	Default value	Description
15	0	Sign for the correction value (0: positive, 1: negative)
14 to 10	00000	Correction value This parameter allows the basic time in the frame to be corrected to local time: `Local time = basic time ± correction value * 0.5 h` This correction takes into account the time zone and the time difference in summer time (daylight savings time) and winter time (standard time).
9	0	Reserved
8	0	Reserved
7	0	Notification hour This parameter indicates whether the next time adjustment also includes a switchover from summer (daylight savings time) to winter time (standard time) or vice versa. (0: no adjustment made, 1: adjustment made).
6	0	Summer time (daylight savings time)/winter time (standard time) indicator This parameter indicates whether the local time calculated using the correction value is summer or winter time.)0Ö winter time, 1: summer time)
5	0	Parameter not used by S7.
4 to 3	00	Time resolution This parameter indicates the resolution of the transmitted clock time. (00: 0.001 s, 01: 0.01 s, 10: 0.1 s, 11: 1 s)
2	0	Parameter not used by S7.
1	0	Parameter not used by S7.
0	0	Synchronization failure This parameter indicates whether the time transmitted in the message frame from an external time master (e.g. SICLOCK) is synchronized. (0: synchronization failed, 1: synchronization occurred) **Note:** Evaluation of this bit in a CPU is only meaningful if there is continuous external time synchronization.

Data record of the partial list extract with SZL-ID W#16#0132 index W#16#000B

Contents

The partial list extract with the SZL-IDW#16#0132 and the IndexW#16#000B contains information about the status of 32-bit runtime meter 0 to 7 of the module.

Note

In the partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0008 this runtime meter is displayed as 16-bit runtime meter.
With it you can continue to implement programs that were developed for a CPU with 16-bit runtime meters, and which continue to use the partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0008.

Data record

A data record of the partial list extract with the SZL-IDW#16#0132 and the index W#16#000B has the following structure:

Name	Length	Meaning
Index	1 word	W#16#000B: Time system status
rtm _0	1 byte	Runtime meter Bit x: Overflow of runtime meter x, 0 ≤ x ≤ 7 (Bit = 1: run-time meter is busy)
res	1 byte	Reserved
rtmOV _0	1 byte	Runtime meter overflow Bit x: Overflow of runtime meter x, 0 ≤ x ≤ 7 (Bit = 1: Overflow)
res	1 byte	Reserved
clock 0	2 words	Runtime meter 0: time in hours
clock 1	2 words	Runtime meter 1: time in hours
clock 2	2 words	Runtime meter 2: time in hours
clock 3	2 words	Runtime meter 3: time in hours
clock 4	2 words	Runtime meter 4: time in hours
clock 5	2 words	Runtime meter 5: time in hours
clock 6	2 words	Runtime meter 6: time in hours
clock 7	2 words	Runtime meter 7: time in hours
res	1 word	Reserved

Data record of the partial list extract with SZL-ID W#16#0132 index W#16#000C

Contents

The partial list extract with the SZL-IDW#16#0132 and the INDEXW#16#000C contains information about the status of 32-bit runtime meter 8 to 15 of the module.

Data record

A data record of the partial list extract with the SZL-IDW#16#0132 and the INDEXW#16#000C has the following structure:

Name	Length	Meaning
Index	1 word	W#16#000C: Time system status
rtm_0	1 byte	Runtime meter Bit x: Status of run-time meter (8+x) , 0 ≤ x ≤ 7 (Bit = 1: run-time meter is busy)
res	1 byte	Reserved
rtmOV _0	1 byte	Runtime meter overflow Bit x: Overflow of run-time meter (8+x) , 0 ≤ x ≤ 7 (Bit = 1: Overflow)
res	1 byte	Reserved
clock 8	2 words	Runtime meter 8: time in hours
clock 9	2 words	Runtime meter 9: time in hours
clock 10	2 words	Runtime meter 10: time in hours
clock 11	2 words	Runtime meter 11: time in hours
clock 12	2 words	Runtime meter 12: time in hours
clock 13	2 words	Runtime meter 13: time in hours
clock 14	2 words	Runtime meter 14: time in hours
clock 15	2 words	Runtime meter 15: time in hours
res	1 word	Reserved

Data record of the partial list extract with SZL-ID W#16#0232 index W#16#0004

Contents

The partial list extract with the SZL-IDW#16#0232 and the Index W#16#0004 contains information about the CPU protection level and the settings of the operator mode switch and checksums of the hardware configuration and the user program.

Data record

A data record of the partial list extract with the SZL-IDW#16#0232 and the Index W#16#0004 has the following structure:

Name	Length	Meaning
Index	1 word	• Byte 1: B#16#04: CPU protection level, operator control settings and checksums • Byte 0: Standard CPU: B#16#00 Bit 3: 0 = Reserve CPU, 1 = Master CPU Bits 4 to 7: 1111
ProtLevel_ModeSwitch	1 word	Protection level set with the mode selector (1, 2, 3)
ProtLevel_Par	1 word	Protection level set in parameters (0, 1, 2, 3; 0: no password, protection level invalid).

Name	Length	Meaning
ProtLevel_Valid	1 word	Valid protection level of the CPU
ModeSwitch	1 word	Mode selector setting (1:RUN, 2:RUN-P, 3:STOP, 4:MRES, 0:undefined or cannot be determined)
Startup_ModeSwitch	1 word	Startup switch setting (1:CRST, 2:WRST, 0:undefined, does not exist or cannot be determined)
res	1 word	Reserved
ID_Val_CheckSum	1 word	ID for the validity of the four following checksums (0: invalid)
ID_CheckSum1_HWConf	1 word	Checksum 1 of the hardware configuration (Intel format): Exclusive OR link over the lengths of all system data blocks.
ID_CheckSum2_HWConf	1 word	Checksum 2 of the hardware configuration (Intel format): Exclusive OR link over the checksums of all system data blocks.
ID_CheckSum1_UserProg	1 word	Checksum 1 of the user program (Intel format): Exclusive-or link over the lengths of the following blocks: OBs, DBs, FBs, FCs
ID_CheckSum2_UserProg	1 word	Checksum 2 of the user program (Intel format): Exclusive-or link over the checksums of the following blocks: OBs, DBs, FBs, FCs
res	2 words	Reserved
sfc_req	1 word	Protection level 2 or 3 request by the instruction "PROTECT" (1: request has been made)
sfc_act	1 word	Protection level 2 or 3 activation by the instruction "PROTECT" (1: activated)
res	4 words	Reserved

SZL-ID W#16#xy74 - status of the module LEDs

Purpose

If you read the partial list with SZL-ID W#16#xy74 with standard CPUs (if present) you obtain the status of the module LEDs.

Header

The header of the partial list with the SZL-ID W#16#xy74 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0174 Status of an LED. You use the INDEX parameter to select the LED.
INDEX	LED ID (only relevant for SZL-IDW#16#0174) • W#16#0001: SF (group error) • W#16#0002: INTF (internal fault) • W#16#0003: EXTF (external fault) • W#16#0004: RUN • W#16#0005: STOP • W#16#0006: FRCE (Force) • W#16#0007: CRST (cold restart) • W#16#0008: BAF (battery fault/overload, short circuit of battery voltage on bus) • W#16#0009: USR (user-defined) • W#16#000A: USR1 (user-defined) • W#16#000B: BUS1F (bus fault interface 1) • W#16#000C: BUS2F (Bus error interface 2) and BUS5F (Bus error interface 5) for the CPUs 414-3 PN/DP,416-3 PN/DP and 416F-3 PN/DP • W#16#0012: IFM1F (interface fault interface module 1) • W#16#0013: IFM2F (interface fault interface module 2) • W#16#0014: BUS3F (bus fault interface 3) • W#16#0015: MAINT (maintenance request) • W#16#0016: DC24V • W#16#0017: BUS5F (bus fault interface 5) • W#16#0080: IF (init failure) • W#16#0081: UF (user failure) • W#16#0082: MF (monitoring failure) • W#16#0083: CF (communication failure) • W#16#0084: TF (task failure) • W#16#00EC: APPL_STATE_RED • W#16#00ED: APPL_STATE_GREEN
LENTHDR	W#16#0004: One data record is 2 words long (4 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-ID W#16#xy74 has the following structure:

Name	Length	Meaning
CPU_LED_ID	1 word	• Byte 0: – Standard CPU: B#16#00 – Bit 3: 0=standby CPU, 1=master CPU – Bits 4 to 7: 1111 • Byte 1: LED ID
LED_on	1 byte	Status of the LED: • 0: off • 1: on
LED_flash	1 byte	Flashing status of the LED: • 0: not flashing • 1: flashing normally (2 Hz) • 2: flashing slowly (0.5 Hz))

SZL-ID W#16#xy90 - DP master system information

Purpose

If you read the partial list SZL-IDW#16#xy90, you obtain the status information of all DP master systems known to the CPU.

Header

The header of the partial list with the SZL-IDW#16#xy90 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0090: Information regarding all DP master systems known to the CPU W#16#0190: Information regarding one DP master system W#16#0F90: Only SZL partial header information
INDEX	• For the partial list extract with the SZL-IDW#16#0190: – Low byte: B#16#00 – High byte: DP master system ID • For the partial list extract with the SZL-IDW#16#0090 and W#16#0F90: – W#16#0000
LENTHDR	W#16#000E: One data record is 7 words long (14 bytes)
N_DR	Number of data records For the partial list extract with the SZL-IDW#16#0190: 0 to 1 For the partial list extract with the SZL-IDW#16#0090: • For a standard CPU: 0 to 14

Data record

A data record of the partial list with the IDW#16#xy90 has the following structure:

Name	Length	Meaning	
dp_m_id	1 byte	DP master system ID	
dp_m_rack	1 byte	Rack number of the DP master • For a standard CPU: 0	
dp_m_slot	1 byte	Slot of the DP master or slot of the CPU (with integrated DP interface)	
dp_m_SubModule	1 byte	• with integrated DP interface: interface number of the DP master: – 1: X2 – 2: X1 – 3: IF1 – 4: IF2 • with external DP interface: 0	
logadr	1 word	logic start address of the DP master	
res	1 word	Reserved	
res	1 word	Reserved	
dp_m_state	1 byte	additional properties of the DP master system	
		Bit 0:	DP mode • 0: S7-compatible • 1: DPV1
		Bit 1:	DP cycle • 0: not constant bus cycle • 1: constant bus cycle
		Bit 2 to 6:	• Reserved
		Bit 7:	DP master type • 0: integrated DP master • 1: external DP master
res	3 bytes	Reserved	

SZL-ID W#16#xy91 - module status information

Purpose

If you read the partial list with SZL-ID W#16#xy91, you obtain the status information of modules assigned to the CPU.

Note

An evaluation of the SZL-ID W#16#xy91 is not possible for a module with packed addresses (ET 200S).

Header

The header of the partial list with the SZL-ID W#16#xy91 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0091 Module status information of all inserted and configured modules/sub-modules of the CPU (S7-400 only) • W#16#0191 Status information of all modules / racks with incorrect module ID that are not disabled (S7-400 only) • W#16#0291 Module status information of all modules that are disrupted and not disabled (S7-400 only) • W#16#0391 Module status information of all modules that are not available (S7-400 only) • W#16#0591 Module status information of all sub-modules of the host module • W#16#0991 Module status information of one DP master system • W#16#0A91 Status information of all DP sub and master systems (S7-300 only without CPU 318-2 DP) or PROFINET IO systems • W#16#0C91 Module status information of a module in the central rack or on an integrated DP interface module or on a PROFINET interface module over the logical base address • W#16#4C91 Module status information of a module on an external DP interface module using the logical base address If you use more than four external DP interface modules,RET_VAL W#16#80A4 may occur incorrectly. • W#16#0D91 Module status information of all modules in the specified rack/station (DP or PROFINET) • W#16#0E91 Module status information of all assigned modules • W#16#0F91 Only SZL partial header information

Contents	Meaning
INDEX	• For the partial list extract with the SZL-ID W#16#0C91: – S7-400: Bits 0 to 14: Logical base address of the module, Bit 15: 0 = input, 1 = output – S7-300: Bits 0 to 14: Any logical address of the module, Bit 15: 0 = input, 1 = output • For the partial list extract with the SZL-ID W#16#4C91 (S7-400 only): – Bits 0 to 14: Logical base address of the module, – Bit 15: 0 = input / 1 = output • For the partial list extract with the SZL IDs W#16#0091, W#16#0191,W#16#0291, W#16#0391, W#16#0491, W#16#0591, W#16#0A91, W#16#0E91, W#16#0F91: INDEX is irrelevant, all modules (in rack and in distributed I/O devices) • For the partial list extract with the SZL-ID W#16#0991 (S7-400 only): W#16#xx00: All modules of a DP master system (xx contains the DP master system ID) • For the partial list extract with the SZL-ID W#16#0D91 – W#16#00xx: All modules and interface modules of a rack (xx contains the number of the rack) – W#16#xxyy: All modules of a DP station or all IO devices of a PROFINET IO station (PROFIBUS DP: xx contains the DP master system ID, yy the station number; PROFINET IO: Bit 0 to 10: station number, bits 11 to 14: the last two places in the PNIO subsystem ID, bit 15: 1 (see third illustration below for adr1)
LENTHDR	W#16#0010: One data record is 8 words long (16 bytes)
N_DR	Number of data records; depending on the product the number of records transferred in the "RDSYSST (Page 5444)" can be lower

In the case of W#16#0091, W#16#0191 and W#16#0F91 two additional data records are supplied per rack:

- A record for the power supply in as far as it exists and has been planned and
- a record for the rack.

The sequence of the records in case of a centralized structure is: PS, Slot 1, Slot 2, ..., Slot 18, rack.

A data record of the partial list with the ID W#16#xy91 has the following structure:

Name	Length	Meaning
adr1	1 word	• For a central configuration: number of the rack • For a distributed configuration with PROFIBUS DP: DP master system ID, station number • For a distributed configuration with PROFINET IO: bit 15 = 1 (PROFINET IO identifier) the last two places in the PROFINET IO system ID, station number Note: A PROFINET interface is always handled as a "sub-module in the central configuration", regardless of the use for PROFINET IO.
adr2	1 word	• For a central configuration and a distributed configuration with PROFIBUS DP: slot number and sub-module slot number • For a distributed configuration with PROFINET IO: slot number Note: A PROFINET interface is always handled as a "sub-module in the central configuration", regardless of the use for PROFINET IO.
LogicalAddr	1 word	First assigned logical I/O address (base address)
SetType	1 word	PROFINET IO: expected (configured) type (see below) otherwise reserved
ActualType	1 word	PROFINET IO: actual type (see below) otherwise reserved
res	1 word	00xx=CPU no.1-4 (only S7-400) for PROFINET IO: • SZL-ID=W#16#0C91: Number of actually existing sub-modules (without sub-module 0) • SZL-ID=W#16#0D91: Number of sub-modules (without sub-module 0) • SZL-ID=W#16#4C91: Number of actually existing sub-modules (without sub-module 0) • SZL-ID=W#16#4D91: Number of actually existing sub-modules (without sub-module 0)

Name	Length	Meaning
IOStat	1 word	I/O status • Bit 0 = 1: Module faulty (detected by diagnostic interrupt) • Bit 1 = 1: module exists • Bit 2 = 1: module does not exist • Bit 3 = 1: module disabled • Bit 4 = 1: station error (only representative slot) • Bit 5 = 1: A CiR event at this module / station is busy or not yet completed • Bit 6 = 1: reserved for S7-400 • Bit 7 = 1: module in local bus segment • Bit 8 to 15: Data ID for logical address (input: B#16#B4, output: B#16#B5, integrated DP interface: B#16#FE, external DP interface: B#16#FF)
AreaID_ModuleWidth	1 word	Area identification/module width • Bit 0 to 2: Module width • Bit 3: Reserved • Bit 4 to 6: Area ID – 0 = S7-400 – 1 = S7-300 – 2 = ET area – 3 = P-area – 4 = Q-area – 5 = IM3-area – 6 = IM4-area • Bit 7: Reserved

At certain modules the following values are indicated in the record:

Name	PS (S7-400 only)	CPU	IFM-CPU (S7-300)	Rack (only S7-400)
adr1	Number of the rack	Standard information as described above	Standard information as described above	Number of the rack
adr2	W#16#01FF	W#16#0200 or W#16#0200 to W#16#1800	W#16#0200	W#16#00FF
LogicalAddr	W#16#0000	W#16#7FFF	W#16#007C	W#16#0000
SetType	Standard information as described above	W#16#00C0 or W#16#0081 or W#16#0082	W#16#00C0	Standard information as described above
IOStat	W#16#0000	Standard information as described above	Standard information as described above	W#16#0000
AreaID_Module-Width	W#16#0000	W#16#0011 or W#16#0001 or W#16#0002	W#16#0011	W#16#0000

Parameter adr1

The adr1 parameter contains:

- In a central configuration, the rack number (0-31).

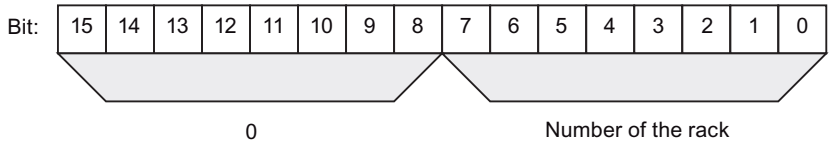

- In a distributed configuration with PROFIBUS DP
  - The DP master system ID (1-32)
  - The station number (0-127).

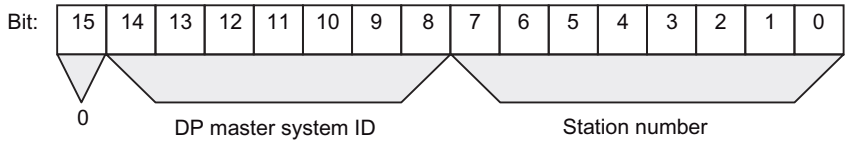

- In a distributed configuration with PROFINET IO:
  - Identifier bit for PROFINET IO (bit 15)
  - The last two positions in the PROFINET IO system ID (0-15). To obtain the complete PROFINET IO system ID, you must add 100 (decimal) to it.
  - The station number (0-2047).

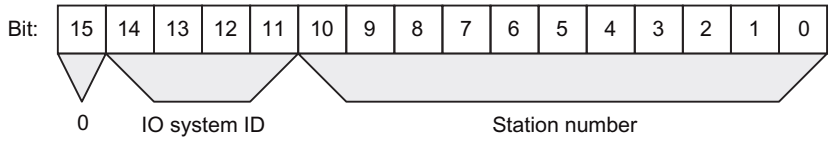

Parameter adr2

The adr2 parameter contains:

- For a central configuration and for distributed configuration with PROFIBUS DP, the slot number and the sub-module slot number.

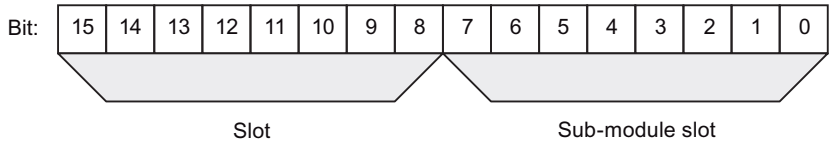

- For a distributed configuration with PROFINET IO, the slot number.

Set type parameter (SetType) and actual type parameter (ActualType) for PROFINET IO

Type identifier (W#16#...)	Meaning
8100	Entered for expected (configured) type and actual type if type check is not possible
8101	Entered for expected (configured) type if a type check is possible
8101	Entered as actual type if expected = actual
8102	Entered as actual type if expected <> actual

SZL-ID W#16#xy92 - rack / station status information

Purpose

If you read the partial list with the SZL-IDW#16#xy92, you obtain information about the expected and the actual hardware configuration of centrally installed racks and stations of a DP master system.

Reading out the SZL using "RDSYSST" for an S7-400 CPU

If you read out the partial list with "RDSYSST (Page 5444)" you must strictly ensure that the parameters SZL_ID and INDEX of "RDSYSST (Page 5444)" match each other.

SZL_ID	INDEX
W#16#0092 or W#16#0192 or W#16#0292 or W#16#0392 or W#16#0492 or W#16#0592 or W#16#0692	DP master system ID of a DP master system which is connected via an **integrated** DP connection.
:	:
W#16#4092 or W#16#4292 or W#16#4692	DP master system ID of a DP master system which is connected via an **external** DP connection.

Header

The header of the partial list with the SZL-IDW#16#xy92 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0092: Set status of the central racks / stations of a DP master system connected via an integrated DP interface module. • W#16#4092: Set status of the stations of a DP master system connected via an external DP interface module • W#16#0192: Activation status of the stations of a DP master system that is connected via an integrated DP interface module • W#16#0292: Set status of the central racks / stations of a DP master system connected via an integrated DP interface module. • W#16#0392: Status of the backup batteries in a rack / module rack of a CPU after at least one battery has failed • W#16#0492: Status of the overall battery backup of all racks / module racks of a CPU • W#16#0592: Status of the 24 V power supply to all racks / module racks of a CPU • W#16#4292: Actual status of the stations of a DP master system that is connected via an external DP interface module • W#16#0692: Diagnostics status of the expansion racks in the central configuration / stations of a DP master system connected via an integrated DP interface • W#16#4692: Diagnositics status of the stations of a DP master system connected via an external DP interface
INDEX	0 / DP master system ID
LENTHDR	W#16#0010: Data record is eight words long (16 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the IDW#16#xy92 has the following structure:

Contents	Length	Meaning
status_0 to status_15	16 bytes	Rack status/ station status or backup status. (The backup status is only relevant for DP modules) • W#16#0092: – Bit=0: rack/station not configured – Bit=1: rack/station not configured • W#16#4092: – Bit=0: station not configured – Bit=1: Station configured • W#16#0192: – Bit=0: Station is not configured or configured and activated – Bit=1: station is configured and activated • W#16#0292: – Bit=0: Rack/station failure, deactivated or not configured – Bit=1: Rack/station exists, activated and has not failed • W#16#4292: – Bit=0: Station failure, deactivated or not configured – Bit=1: Station exists, activated and has not failed • W#16#0692: – Bit=0: All modules of the expansion rack / station exist, are available with no problems and the station is activated – Bit=1: At least one module of the expansion rack / station is not OK or the station is deactivated • W#16#4692: – Bit=0: All modules of a station exist, are available with no problems and the station is activated – Bit=1: At least one module of a station is not OK or the station is deactivated
status_0	1 byte	Bit 0: Central rack (INDEX = 0) or station 1 (INDEX <> 0) Bit 1: 1. Expansion rack or station 2 : : Bit 7: 7. Expansion rack or station 8
status_1	1 byte	Bit 0: 8. Expansion rack or station 9 : : Bit 7: 15. Expansion rack or station 16

Contents	Length	Meaning
status_2	1 byte	Bit 0: 16. Expansion rack or station 17 : : Bit 5: 21. Expansion rack or station 22 Bit 6: 0 or station 23 Bit 7: 0 or station 24
status_3	1 byte	Bit 0: 0 or station 25 : : Bit 5: 0 or station 30 Bit 6: Expansion rack SIMATIC S5 area or station 31 Bit 7: 0 or station 32
status_4	1 byte	Bit 0: 0 or station 33 : : Bit 7: 0 or station 40
: :		
status_15	1 byte	Bit 0: 0 or station 121 : : Bit 7: 0 or station 128

SZL-ID W#16#0x94 - status information for rack/station

Purpose

The partial list with the SZL-IDW#16#0x94 contains information about the set and actual configuration of module racks in central configurations and stations of a PROFIBUS DP master system/PROFINET IO controller system.

Header

The header of the partial list with the SZL-IDW#16#0y94 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0094: Set status of the rack in the central rack / stations of an IO controller system (Status bit = 1: rack/station configured) • W#16#0194: Activation status of a station of an IO controller system that is configured and deactivated (Status bit =1) • W#16#0294: Set status of the rack in the central rack / stations of an IO controller system (Status bit = 1: rack/station exists, activated and not failed) • W#16#0694: Diagnostic status of the expansion units in the central rack / stations of an IO controller system (Status bit = 1: At least one module of the rack/station is disrupted or deactivated) • W#16#0794: Diagnostic / maintenance status of the central rack / stations of an IO controller system (Status bit = 0: No fault and no maintenance necessary, Status bit = 1: Rack / station has malfunction or / and maintenance required or / and maintenance demanded) • W#16#0F94: Only header information
INDEX	• 0: Central module • 1-32: Distributed module on PROFIBUS DP • 100-115: distributed module on PROFINET IO
LENTHDR	Length of the following data record
N_DR	Number of data records

Data record

A data record of the partial list with the IDW#16#0y94 has the following structure:

Contents	Length	Meaning
index	1 word	• 0: Central module • 1-32: Distributed module on PROFIBUS DP • 100-115: distributed module on PROFINET IO
status_0	BOOL	Group information • 1: At least one of the following status bits has the value 1 • 0: all the following status bits have the value 0
status_1	BOOL	Status, station 1
status_2	BOOL	Status, station 2
..		
status_2047	BOOL	Status, station 2047

A status bit of non-configured racks/stations/devices has the value "0".

Note

Important difference to the previous SZL-IDW#16#xy92

Compared to the previous SZL-IDW#16#xy92, the data have been shifted by one bit since bit status_0 is used for group information.

SZL-ID W#16#xy95 - extended DP master system / PROFINET IO system information

Purpose

The partial list with the SZL-IDW#16#xy95 supplies you with extended status information on all DP master systems/PROFINET IO systems known to the CPU. Compared to the partial list with the SZL-IDW#16#xy90, this list contains information about PROFINET IO systems and additional information about the isochronous mode of a DP master system.

Header

The header of the partial list with the SZL-IDW#16#xy95 is structured as follows:

Contents	Meaning	
SZL-ID	The SZL-ID of the partial list extract	
	W#16#0195:	Extended information on a DP master system/PROFINET IO system
	W#16#0F95:	Only SZL partial header information
INDEX	• For the partial list extract with the SZL-IDW#16#0195: – Low byte: B#16#00 – High byte: DP master system ID/PROFINET IO system ID • For the partial list extract with the SZL-IDW#16#0F95: – W#16#0000	
LENTHDR	W#16#0028:	one data record is 20 words long (40 bytes)
N_DR	Number of data records: For the partial list extract with the SZL-IDW#16#0195: 0 to 1	

Data record

A data record of the partial list with the IDW#16#xy95 has the following structure:

Name	Length	Meaning
dp_m_id	1 byte	DP master system ID/PROFINET IO system ID
dp_m_rack	1 byte	Rack number of the DP master / IO controller • For a standard CPU: 0
dp_m_slot	1 byte	Slot of the DP master / IO controller or slot of the CPU (with integrated DP interface)

Name	Length	Meaning	
dp_m_SubModule	1 byte	• with integrated DP interface: Interface ID of the DP master / IO controller: – 1: X2 – 2: X1 – 3: IF1 – 4: IF2 • with external interface: 0	
LogicalAddr	2 bytes	Logical start address of the DP master / IO controller	
res	2 bytes	Reserved	
res	2 bytes	Reserved	
dp_m_state	1 byte	• Additional properties of the DP master system / PROFINET IO system	
		Bit 0:	DP mode (PROFIBUS DP only) • 0: S7-compatible • 1: DPV1
		Bit 1:	DP and PN cycle • 0: not constant bus cycle • 1: constant bus cycle
		Bit 2 to 6:	• Reserved
		Bit 7:	Type of DP master / IO controller • 0: integrated DP master / IO controller • 1: external DP master / IO controller
dp_address	1 byte	DP node number (PROFIBUS address)	
res	2 bytes	Reserved	
SyncCyclInter_OB	1 byte	Assigned synchronous cycle interrupt OB (only relevant if the DP cycle is equidistant)	
res	1 byte	Reserved	
baudrate	4 bytes	Baud rate of the DP master system and PNIO system as hex value	
IsoDPCycle	4 bytes	Period of the equidistant DP and PN cycle in µs	
res	16 bytes	Reserved	

SZL-ID W#16#xy96 - PROFINET IO and PROFIBUS DP module status information

Purpose

If you read the partial list with SZL-IDW#16#xy96, you obtain the status information of modules assigned to the CPU.

The information in the partial list with the SZL-IDW#16#xy96 supplements SZL-IDW#16#xy91 (Page 5482) and provides additional status data on modules and sub-modules.

It provides information specific to PROFINET IO as well as information on PROFIBUS DP modules and central modules.

Header

The header of the partial list with the SZL-IDW#16#xy96 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0696 Module status information of all sub-modules of a specified module (with PROFIBUS DP and central modules, the sub-module level does not exist). • W#16#0C96 Module status information of a module/sub-module located centrally or on a PROFIBUS DP/ PROFINET interface via the start address.
INDEX	• Bits 0 to 14: Address of the module • Bit 15: 0 = input, 1 = output
LENTHDR	Length of the following data record
N_DR	Number of data records

Data record

A data record of the partial list with the IDW#16#xy96 has the following structure:

Contents	Length	Meaning
LogicalAddr	1 word	• Bits 0 to 14: Address of the module • Bit 15: 0 = input, 1 = output
System	1 word	Identifier for the central module/DP master system ID /PROFINET IO system ID: • 0: Central module • 1-32: Distributed module on PROFIBUS DP • 100-115: distributed module on PROFINET IO
API*	2 words	Configured user profile for a distributed PROFINET device. Profiles are sector-specific or technology-specific specifications that go beyond the PROFINET standard. Profile 0 means that the data correspond to the specification in the PROFINET standard.
Station	1 word	Rack no./station number/device number
Slot	1 word	Slot no.
Subslot	1 word	Sub-module slot (if no sub-module can be inserted, "0" must be entered here)
Offset	1 word	Offset in user data address area of the associated module

<table>
<tr><th>Contents</th><th>Length</th><th colspan="3">Meaning</th></tr>
<tr><td rowspan="9">SetType</td><td rowspan="9">7 words</td><td colspan="3">Expected type
The expected type is has a hierarchical structure in PROFINET IO</td></tr>
<tr><td>Word</td><td>PROFINET IO</td><td>PROFIBUS DP</td></tr>
<tr><td>1:</td><td>Manufacturer number or profile ID (e.g.: W#16#FF00 for PROFIBUS)</td><td>0000</td></tr>
<tr><td>2:</td><td>Device</td><td>0000</td></tr>
<tr><td>3:</td><td>Sequential number or profile index</td><td>0000</td></tr>
<tr><td>4:</td><td>1. word of the double word for sub-module identification</td><td>Type identifier</td></tr>
<tr><td>5:</td><td>2. word of the double word for sub-module identification</td><td>0000</td></tr>
<tr><td>6:</td><td>1. word of the double word for sub-module identification</td><td>0000</td></tr>
<tr><td>7:</td><td>2. word of the double word for sub-module identification</td><td>0000</td></tr>
<tr><td>SetType_UnEqual_ActualType</td><td>1 word</td><td colspan="3">Expected/actual identifier
• Bit 0 = 0: Set matches actual
• Bit 0 = 1: Set unequal to actual
• Bit 1 to 15: Reserved</td></tr>
<tr><td>res</td><td>1 word</td><td colspan="3">Reserved</td></tr>
<tr><td>IOStat</td><td>1 word</td><td colspan="3">I/O status
• Bit 0 = 1: Module faulty (detected by diagnostic interrupt)
• Bit 1 = 1: Module exists

Bit 2 = 1: Module does not exist
• Bit 3 = 1: Module disabled

Bit 4 = 1: station error (only representative slot)
• Bit 5 = 1: M7: Module can be host module for sub-modules
:
S7: A CiR event at this module /station is busy or not yet completed
• Bit 6 = 1: reserved for S7-400
• Bit 7 = 1: Module in local bus segment (S7-300 only)
• Bit 8 = 1: Module maintenance demanded ("green")
• Bit 9 = 1: Module maintenance request ("yellow")
• Bit 10 to 15: Reserved</td></tr>
</table>

Contents	Length	Meaning
AreaID_ModuleWidth	1 word	Area identification/module width • Bit 0 to 2: Module width • Bit 3: Reserved • Bit 4 to 6: Area ID – 0 = S7-400 – 1 = S7-300 – 2 = PROFINET IO (distributed) – 3 = P-area – 4 = Q-area – 5 = IM3-area – 6 = IM4-area • Bit 7: Reserved • Bit 7: Reserved
res	5 words	Reserved
* Application Process Instance		

Partial list with the SZL-IDW#16#0696 for modules on PROFIBUS DP

This results in the error message "sub-module level not present".

SZL-ID W#16#xy9C - Tool changeover information (PROFINET IO)

Purpose

You obtain information on the configured tool changeover and their tools via the partial lists with the SZL-ID W#16#xy9C.

Tool changeover contacts are IO devices that manage tools. A tool can consist of one or several IO devices. Each tool is uniquely assigned to a tool changeover contact port.

In the case of a tool changeover all IO devices that belong to the active tool at the time are deactivated and the IO devices that belong to the new tool are activated. Deactivating and activating take place via the "D_ACT_DP (Page 5251)" instruction.

Header

The header of the partial list with the SZL-ID W#16#xy9C is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#009C Information about all tool changeover contacts and their tools on a PROFINET IO system • W#16#019C Information about all tool changeover contacts on a PROFINET IO system • W#16#029C Information about a tool changeover contact and its tools • W#16#039C Information about a tool changeover contact and its IO devices • W#16#0F9C Only SZL partial header information
INDEX	For the partial list extract with the SZL-ID • W#16#009C: PROFINET IO system ID • W#16#019C: PROFINET IO system ID • W#16#029C: logic address of a tool changeover contact • W#16#039C: logic address of any IO device of a tool • W#16#0F9C: PROFINET IO system ID
LENTHDR	W#16#000C (Length of the following data record in byte: 12)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-ID W#16#xy9C has the following structure:

Contents	Length	Meaning
StationW	1 WORD	Station number (of a data record, could be a tool changeover contact or a tool station)
LogAdrW	1 WORD	• Bits 0 to 14: Address of the module • Bit 15: 0 = input, 1 = output
StationWZK	1 WORD	Station number of a tool head (The tool head is the tool IO device directly connected to the tool changeover contact port.)
StationWZW	1 WORD	Station number of a tool changeover contact
SlotWZW	1 WORD	Slot of a tool changeover contact
SubslotWZW	1 WORD	Submodule slot of a tool changeover contact

Note

If a tool changeover contact has several ports that manage tools, a data record is supplied for each port.

Example

The usage of the individual partial lists is shown once again in the following example.

On the phase of a PROFINET IO system, there are two tool changeover contacts (IOD 3 and IOD 10), whose tools have the following structure:

- Tool changeover contact IOD 3 (IO-Device 3) with the following three tools on the tool changeover contact port 2:
  - Tool 1 (IOD 4, IOD 5 and IOD 6)
  - Tool 2 (IOD 7)
  - Tool 3 (IOD 8 and IOD 9)
- Tool changeover contact IOD 10 with two tool changeover contact ports of which each has two tools
  - Tool changeover contact port 3: Tool 1 (IOD 11 and IOD 12) tool 2 (IOD 13)
  - Tool changeover contact port 4: Tool 1 (IOD 14, IOD 15 and IOD 16), tool 2 (IOD 17 and IOD 18)

The result is the following configuration:

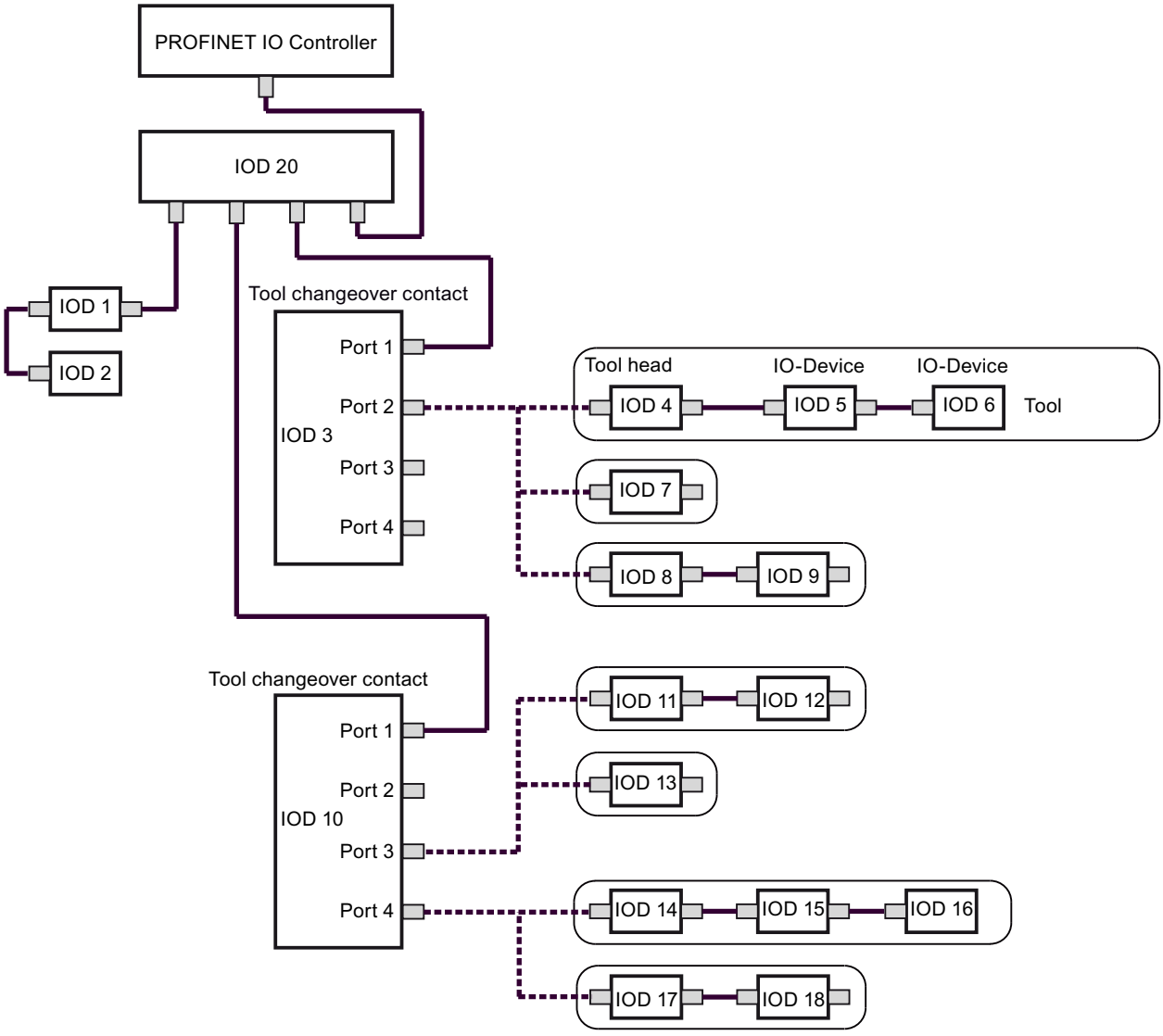

The SZL partial lists return the data records for the following IO-Devices:

- Partial list extract with SZL-ID W#16#009C (Index: PROFINET IO system ID): Returns 17 data records for the following IO-Devices:
  - Tool changeover contacts: IOD 3, IOD 10 (for port 3) and IOD 10 (for port 4)
  - Tools: IOD 4, IOD 5, IOD 6, IOD 7, IOD 8, IOD 9, IOD 11, IOD 12, IOD 13, IOD 14, IOD 15, IOD 16, IOD 17 and IOD 18
- Partial list extract with SZL-ID W#16#019C (Index: PROFINET IO system ID): Returns 3 data records for the following IO-Devices:
  - Tool changeover contacts: IOD 3, IOD 10 (for port 3) and IOD 10 (for port 4)
  - Tools: None

- Partial list extract with SZL-ID W#16#029C (Index: Address of IOD 3): Returns 3 data records for the following IO-Devices:
  - Tool changeover contacts: IOD 3
  - Tools: IOD 4, IOD 5, IOD 6, IOD 7, IOD 8 and IOD 9
- Partial list extract with SZL-ID W#16#029C (Index: Address of IOD 10): Returns 10 data records for the following IO-Devices:
  - Tool changeover contacts: IOD 10 (for port 3) and IOD 10 (for port 4)
  - Tools: IOD 11, IOD 12, IOD 13, IOD 14, IOD 15, IOD 16, IOD 17 and IOD 18
- Partial list extract with SZL-ID W#16#039C (Index: Logic address of IOD 4): Returns 3 data records for the following IO-Devices: (This is analog for the logic address of IOD 5 and IOD 6)
  - Tool changeover contacts: None
  - Tools: IOD 4, IOD 5 and IOD 6
- Partial list extract with SZL-ID W#16#039C (Index: Logic address of IOD 13): Returns data record for the following IO-Devices:
  - Tool changeover contacts: None
  - Tools: IOD 13

SZL-ID W#16#xyA0 - diagnostic buffer

Purpose

If you read the partial list with SZL-IDW#16#xyA0 you obtain the entries in the diagnostics buffer of the module.

Note

The S7-300-CPUs return maximum as many data records as the displayed diagnostic buffer entries number in operating mode (default value: 10). The S7-400-CPUs return maximum 21 data records.

Header

The header of the partial list with the SZL-IDW#16#xyA0 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#00A0: All entries that can be supplied in the currently active operating mode • W#16#01A0: The most recent entries; you specify the number of most recent entries with the INDEX parameter. If the number of alarms in the diagnostics buffer is less than the configured maximum number of alarms, the "RDSYSST (Page 5444)" instruction may provide invalid values in this partial list extract. You therefore should avoid a non-buffered POWER OFF! • W#16#0FA0: Only SZL partial header information
INDEX	Only for SZL-IDW#16#01A0: Number of most recent entries
LENTHDR	W#16#0014: One data record is 10 words long (20 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xyA0 has the following structure:

Name	Length in words	Meaning
ID	1 word	Event ID
info	5 words	Information about the event and its consequences
time	4 words	Time stamp of the event

Diagnostics buffer

Additional information on events in the diagnostics buffer is available via context-sensitive help for the displayed event.

SZL-ID W#16#00B1 - module diagnostic information

Purpose

If you read the partial list SZL-IDW#16#00B1, you obtain the first 4 diagnostic bytes of a module with diagnostic capability.

Header

The header of the partial list with the SZL-IDW#16#00B1 is structured as follows:

Contents	Meaning
SZL-ID	W#16#00B1
INDEX	• Bit 0 to 14: Logical base address • Bit 15: 0 = input, 1 = output
LENTHDR	W#16#0004: One data record is 2 words long (4 bytes)
N_DR	1

Data record

A data record of the partial list with the SZL-IDW#16#00B1 has the following structure:

Name	Length	Meaning
byte0	1 byte	• Bit 0: Module fault/OK (group fault ID) • Bit 1: Internal error • Bit 2: External error • Bit 3: Channel fault • Bit 4: External voltage failed • Bit 5: Front panel connector not plugged in • Bit 6: Module is not configured • Bit 7: Incorrect parameters on module
byte1	1 byte	• Bit 0 to bit 3: Module class (CPU, FM, CP, IM, SM, ...) • Bit 4: Channel information exists • Bit 5: User information exists • Bit 6: Diagnostic interrupt from substitute • Bit 7: Maintenance requirement (PROFINET IO only)

Name	Length	Meaning
byte2	1 byte	• Bit 0: User module is missing or has an error • Bit 1: Communication problem • Bit 2: Operating mode RUN/STOP (0 = RUN, 1 = STOP) • Bit 3: Time monitoring responded (watchdog) • Bit 4: Module internal supply voltage failure • Bit 5: Battery exhausted (BFS) • Bit 6: Entire backup failed • Bit 7: Maintenance request (PROFINET IO only)
byte3	1 byte	• Bit 0: Expansion unit failure (detected by IM) • Bit 1: Processor failure • Bit 2: EPROM fault • Bit 3: RAM fault • Bit 4: ADC/DAC error • Bit 5: Fuse tripped • Bit 6: Hardware interrupt lost • Bit 7: Reserve (initialized with 0)

SZL-ID W#16#00B2 - diagnostic data record 1 with physical address

Purpose

If you read the partial list SZL-IDW#16#00B2, you obtain diagnostic data record 1 of a module in a central rack (in other words, not for DP or sub-modules). You specify the module by using the rack and slot number.

Header

The header of the partial list with the SZL-IDW#16#00B2 is structured as follows:

Contents	Meaning
SZL-ID	W#16#00B2
INDEX	W#16#xxyy: • xx contains the number of the rack • yy contains the slot number
LENTHDR	The length of the data record depends on the module.
N_DR	1

Data record

The size of a data record of the partial list SZL-IDW#16#00B2 and its contents depend on the particular module. More detailed information in this regard is provided by the respective manual for the module.

SZL-ID W#16#00B3 - module diagnostic data with logical base address

Purpose

If you read the partial list SZL-IDW#16#00B3, you will obtain all diagnostic data of a module. You can also obtain this information for DP and sub-modules. You select the module using its logical base address.

Header

The header of the partial list with the SZL-IDW#16#00B3 is structured as follows:

Contents	Meaning
SZL-ID	W#16#00B3
INDEX	• Bit 0 to 14: Logical base address • Bit 15: 0 = input, 1 = output
LENTHDR	The length of the data record depends on the module.
N_DR	1

Data record

The size of a data record of a partial list SZL-IDW#16#00B3 and its contents depends on the particular module. More detailed information in this regard is provided by the respective manual for the module.

SZL-ID W#16#00B4 - diagnostic data of a DP slave

Purpose

If you read the partial list SZL-IDW#16#00B4, you will obtain the diagnostic data of a DP slave. This diagnostic data is structured in compliance with EN 50 170 Volume 2, PROFIBUS. You select the module using the diagnostic address you configured.

Header

The header of the partial list with the SZL-IDW#16#00B4 is structured as follows:

Contents	Meaning
SZL-ID	W#16#00B4
INDEX	Configured diagnostic address of the DP slave.
LENTHDR	Length of a data record. The maximum length is 240 bytes; for standard slaves, which have a diagnostic data length of more than 240 bytes up to a maximum of 244 bytes, the first 240 bytes are read and the overflow bit is set in the data.
N_DR	1

Data record

A data record of the partial list with the SZL-IDW#16#00B4 has the following structure:

Name	Length	Meaning
status1	1 byte	Station status1
status2	1 byte	Station status2
status3	1 byte	Station status3
Station_No	1 byte	Master station number
Vendor_ID_highByte	1 byte	Vendor ID (high byte)
Vendor_ID_lowByte	1 byte	Vendor ID (low byte)
....	....	Additional slave-specific diagnostic information

OB_RT: Determine OB program runtime (S7-400)

Description

With the instruction, you can determine the runtime of individual OBs over different time periods.

Note

The instruction returns the last recorded time values for the required OB, regardless of whether or not this OB is currently being loaded. The data are not reset by deleting or overwriting them but rather only after a warm restart.

Parameter

The following table shows the parameters of the instruction "OB_RT":

Parameters	Declaration	Data type	Memory area	Meaning
OB_NR	Input	INT	I, Q, M, D, L	OB whose last evaluated times are to be queried. Valid OB numbers are all those in the OB configuration of your CPU, except for OB 121 and OB 122. Synchronous error processing time is included in the time required for processing the OB that has caused the error. Specification of OBs 121 and 122 or OBs that are not implemented in the CPU causes an error message. For OB_NR=0, the data of the OB in whose context the instruction was called will be transferred. When "OB_RT" is called in OB 121 or OB 122 with OB_NR=0, all time data of the interrupt triggering OB are output, including the time data in OB 12x.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. Otherwise, RET_VAL contains the number of the OB for which these data were called.
PRIO	Output	INT	I, Q, M, D, L	The priority class of the queried OB is output in PRIO .
LAST_RT	Output	DINT	I, Q, M, D, L	Runtime (in microseconds) of the last completed execution of the specified OB. If the OB for which you request the runtimes is currently being processed, the following applies: For the first call of the "OB_RT" instruction during the current execution of the required OB, LAST_RT indicates the run time of the last completed OB execution. With each subsequent call of the "OB_RT" instruction during the current execution of the required OB, LAST_RT • DW#16#FFFF FFFF indicates, if the required OB has already called an "OB_RT" with OB_NR=0. • The runtime of the last completed OB, if an "OB_RT" call with OB_NR=0 has not occurred in the required OB. Note: Interruptions caused by OBs with higher priority are not included in LAST_RT . The OB-specific operating system services (such as generation and provision of the OB start information, updating of the process image, updating of the process image partition) are contained in LAST_RT .

Parameters	Declaration	Data type	Memory area	Meaning
LAST_ET	Output	DINT	I, Q, M, D, L	Time interval (in microseconds) between the OB call and the end of execution of the specified OB. If the OB for which you want to determine runtimes is currently in process the following applies: For the first call of the "OB_RT" instruction during the current processing of the required OB, the time interval between the last completely executed OB request and the end of processing of the specified OB is specified in LAST_ET . For each subsequent call of the "OB_RT" instruction during current processing of the required OB in LAST_ET • DW#16#FFFF FFFF indicates, if the required OB has already called an "OB_RT" with OB_NR=0. • The time interval between the last completely executed OB request and the end of execution of the required OB, if an "OB_RT" call with OB_NR=0 has not occurred in the required OB. Note: Interruptions caused by OBs with higher priority are included in LAST_ET .
CUR_T	Output	DINT	I, Q, M, D, L	Timing of the OB request of the specified OB that is currently being processed, as a relative time in microseconds. If the specified OB is currently not being processed, CUR_T will contain the value "0". Note: The system time is a counter that counts from "0" to "2 147 483 647" microseconds. If an overflow occurs, the counter restarts at "0".
CUR_RT	Output	DINT	I, Q, M, D, L	Previous runtime of the current execution of the specified OB in microseconds.CUR_RT is "0", if the OB is not being processed or has not yet been processed. After processing, the runtime is transferred to LAST_RT and CUR_RT is set to zero. Note: Interruptions caused by OBs with higher priority are not included in CUR_RT . The OB-specific operating system services performed until "OB_RT" is called are contained in CUR_RT .

Parameters	Declaration	Data type	Memory area	Meaning
CUR_ET	Output	DINT	I, Q, M, D, L	Time elapsed since the request of the specified OB that is currently being processed, in microseconds. CUR_ET is "0", if the specified OB is currently not being processed. After processing, the runtime is transferred to LAST_ET and CUR_ET is set to "0". Note: Interruptions caused by OBs with higher priority are included in CUR_ET .
NEXT_ET	Output	DINT	I, Q, M, D, L	If additional calls of the specified OB are queued before the current request has been completed, NEXT_ET indicates the time-to-go between the current time and the time of execution of the next request in microseconds.. NEXT_ET is "0", if no other start event exists besides the start events for this OB that are currently pending or undergoing processing. WinLC RTX and the S7-400 CPUs do not use this parameter. In these cases, NEXT_ET has the value"FFFF FFFF". Note: Interruptions caused by OBs with higher priority are included in NEXT_ET .

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

The times also include the runtimes for any nested processing of synchronous error interrupts (OB 121, OB 122).

Note

When you declare an OB number in OB_NR that exists in the dynamic project data on your CPU without the OS having called the corresponding OB, or you have not yet downloaded it to the CPU, RET_VAL contains the specified OB number and PRIO contains the configured (default, if required) priority of the specified OB. The runtime parameter (CUR_RT, CUR_ET, LAST_RT, LAST_ET, NEXT_ET) returns the initial value DW#16#FFFF FFFF. For the following restart methods or operating state changes, the runtime parameters are set to their initial value: Warm restart, cold restart, hot restart.

Parameter RET_VAL

Event class error code	Explanation
1 to 102	Number of the OB to which information is being transferred.
W#16#8080	OB_NR parameter contains an illegal value.
W#16#8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

C_DIAG: Determine current connection status (S7-400)

Description

You use the instruction to determine the current status of all S7 connections and of all fault-tolerant S7 connections (or their partial connections).

Suitable evaluation of these connection data lets you recognize failures of S7 connections as well as of fault-tolerant S7 connections and report these, if necessary, to an operator control and monitoring system. Monitored connections can be a connection between automation systems as well as the connection of an automation system to an operator control and monitoring system.

Note

A change in the operating state of the CPU: RUN -> STOP ->RUN, does not affect the state of the configured connections. Exception: When an H-station changes from the redundant system state to the Stop system state, the partial connections of all fault-tolerant connections to the standby CPU will be disconnected.

After a power failure, on the other hand, all configured connections will be reestablished and this changes the connection status.

When the instruction is called for the first time or after startup, the connection information will differ, depending on whether the last operating mode of the CPU was STOP or POWER OFF.

Functional description

The "C_DIAG" instruction functions asynchronously, that is, its execution extends over multiple calls.

You start the job by calling "C_DIAG" with REQ=1.

If the job could be executed immediately, the instruction will return the value "0" at the output parameter BUSY . If BUSY = "1", the job is still active.

Call

To recognize the failure of S7 connections and fault-tolerant S7 connections, you call "C_DIAG" in a cyclic interrupt OB that is started, for example, every 10 seconds by the operating system.

Because the status of a connection normally does not change, it is appropriate to copy the connection data to the user program with these cyclic calls only if they have changed since their last call (call with MODE=B#16#02, see below).

The "C_DIAG" instruction has four possible operating modes, which are explained in the following table.

MODE (B#16#...)	"C_DIAG" copies connection data to the user program	"C_DIAG" transfers acknowledgement information to the operating system
00	No	Yes
01	Yes	Yes

MODE (B#16#...)	"C_DIAG" copies connection data to the user program	"C_DIAG" transfers acknowledgement information to the operating system
02	• Yes, if connection data have changed • No, if connection data have not changed	Yes
03	Yes	No

The status changes of the connection data since the last call of "C_DIAG" (with MODE=B#16#00, 01 or 02) are confirmed by transferring the acknowledgement information to the operating system.

Note

If you operate "C_DIAG" in a cyclic interrupt OB in "Conditional copying" mode (MODE=B#16#02), you must ensure that no initial values are contained in the target range after a cold restart of the CPU. You can achieve this in OB 102 with a single call of "C_DIAG" with "Unconditional copying with acknowledgment" mode (MODE=B#16#01).

Parameter

The following table shows the parameters of the instruction "C_DIAG":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request to activate REQ=1: Initialize the job, if not already started
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Possible values: • B#16#00: "C_DIAG" does not copy connection data, but merely transfers acknowledgement information to the operating system. • B#16#01: Regardless of the status change, "C_DIAG" copies all connection data to the user program and transfers an acknowledgement information to the operating system. • B#16#02: If connection data have changed, "C_DIAG" will copy them to the user program. If not changed, they are not copied. "In both cases, C_DIAG" transfers an acknowledgement information to the operating system. • B#16#03: "C_DIAG" copies the connection data to the user program, independent of its change status. It does not transfer acknowledgement information to the operating system.
RET_VAL	Return	INT	I, Q, M, D, L	Return value (error code or job status)
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The job is not yet complete.

Parameter	Declaration	Data type	Memory area	Description
N_CON	Output	INT	I, Q, M, D, L	Index of the last structure in CON_ARR, with .DIS_PCON or .DIS_CON = TRUE . This means only the first N_CON elements of CON_ARR need to be checked in the user program. Note: The first structure in the CON_ARR array has the index "1".
CON_ARR	Output	ANY	I, Q, M, D, L	Destination area for the read connection data. Only the BYTE data type is permitted. A structure is assigned to each connection. Choose a target range size that can receive all structures even at the maximum number of possible connections for your CPU. This is particularly important after a CPU replacement when the new CPU which has greater connection resources than the previous one.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter CON_ARR

The target range for the read connection data is an array of structures. A structure is assigned to each connection.

Initially, the array does not need to be occupied with valid entries and it may contain invalid entries between two valid entries.

The connections are not sorted by connection reference.

Note

Data consistency of a connection is ensured if you copy connection data from the operating system to the selected target range.

Structure organization

Parameters	Data type	Description
CON_ID	WORD	Connection reference that you have assigned to this connection during configuration W#16#FFFF: Invalid ID, which means the connection has not been configured. If CON_ARR[i].DIS_PCON or CON_ARR[i].DIS_CON (see below) is also set, this connection has been reconfigured or deleted since the last call of "C_DIAG".
STAT_CON	BYTE	The current status of the S7 connection or fault-tolerant S7 connection Possible values: • B#16#00: S7 connection not established • B#16#10: Fault-tolerant S7 connection not established • B#16#01: S7 connection is currently being established • B#16#11: Fault-tolerant S7 connection is currently being established • B#16#02: S7 connection is established • B#16#12: Fault-tolerant S7 connection is established, but is not fault tolerant • B#16#13: Fault-tolerant S7 connection has been established and is fault tolerant
PROD_CON	BYTE	Partial connection number of the productive connection Possible values: 0, 1, 2, 3
STBY_CON	BYTE	Partial connection number of the standby connection (B#16#FF: no standby connection) Possible values: 0, 1, 2, 3 Note: Only a fault-tolerant S7 connection can have a standby connection.
DIS_PCON	BOOL	The transitions W#16#12 -> W#16#13 and W#16#13 -> W#16#12 from CON_ARR[i].STAT_CON since the last instruction call set CON_ARR[i].DIS_PCON to "1". All other status changes of the i connection leave CON_ARR[i].DIS_PCON unchanged. Note: • With MODE=B#16#01 and 02 , the operating system bit that corresponds to DIS_PCON is reset when connection data are copied to the target range. • With MODE=B#16#03 , the operating system bit that corresponds to DIS_PCON remains unchanged.
DIS_CON	BOOL	Each modification of CON_ARR[i].STAT_CON since the last "C_DIAG" call, with the exception of the transitions W#16#12 -> W#16#13 and W#16#13 -> W#16#12 sets CON_ARR[i].DIS_CON to 1. Note: • With MODE=B#16#01 and 02 , the operating system bit that corresponds to "DIS_CON" is reset when connection data are copied to the target range. • With MODE=B#16#03 , the operating system bit that corresponds to DIS_CON remains unchanged.
RES0	BYTE	Reserved (B#16#00)
RES1	BYTE	Reserved (B#16#00)

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	• MODE=B#16#00, 01 or 02: No connection status change (structure element STAT_CON) since the last call. Job was executed without errors. • MODE=B#16#03: The copy procedure was carried out without errors.
0001	• MODE=B#16#00, 01 or 02: Connection status change (structure element STAT_CON) with at least one connection since the last call. Job was executed without errors. • MODE=B#16#03: RET_VAL W#16#0001 is not possible:
7000	First call with REQ=0. The job specified with MODE is not active. BUSY has the value "0".
7001	First call with REQ=1. The job specified with MODE is not triggered. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant). The activated job is still running. BUSY has the value "1".
8080	The MODE parameter contains an invalid value.
8081	The CON_ARR parameter contains an illegal data type.
8082	The length specification in the CON_ARR parameter is too small. The instruction copies no data to the target range.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

4.2.3.10 Data block functions

READ_DBL: Read from data block in the load memory

Description

With the instruction you copy a DB or an area of a DB in load memory (Micro Memory Card) to the data area of a destination DB. The destination DB must be relevant for execution; that is, it must not be created with the attribute UNLINKED. The content of the load memory is not changed during the copy process.

To ensure data consistency, you must not change the destination area while "READ_DBL" is being executed (which means as long as the BUSY parameter has the value TRUE).

The following restrictions apply to the parameters SRCBLK and DSTBLK:

- For an ANY pointer of the type BOOL, the length must be divisible by 8.
- For an ANY pointer of the type STRING, the length must be equal to 1.

If required, you can determine the length of the source DB with the "TEST_DB (Page 5526)" instruction.

Note

READ_DBL is processed asynchronously. Therefore, it is not suitable for frequent (or cyclical) reading of tags in the load memory.

Once started, a job is always completed. If the maximum number of simultaneously active "READ_DBL" instructions is reached and you call "READ_DBL" once again at this time in a priority class having higher priority, error code W#16#80C3 will be returned. This means it does not make sense to restart the high-priority job right away.

Functional description

The "READ_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "READ_DBL" with REQ =1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186)

Parameter

The following table shows the parameters of the instruction "READ_DBL":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Read request
SRCBLK	Input	ANY	D	Pointer to data block in the load memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
DSTBLK	Output	ANY	D	Pointer to the data block in the work memory that is to be written to

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
0081	The destination area is larger than the source area. The source area is written completely to the destination area; the remaining bytes of the destination area will not be changed.
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".

Error code (W#16#...)	Description
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source area is larger than the destination area. **The destination area is fully written. The remaining bytes of the source area are ignored.**
8093	No data block or a data block that is not in the work memory is specified for the DSTBLK parameter.
80B1	No data block is specified for the SRCBLK parameter, or the data block specified there is not an object in the load memory (for example, a DB created using "CREAT_DB (Page 5520)").
80B4	DB with F-attribute must not be read.
80C0	The destination DB is currently being processed by another instruction or a communication function.
80C3	The maximum number of simultaneously active "READ_DBL" instructions has already been reached.
8xyy	General error codes See also: Evaluating errors with output parameter RET_VAL (Page 162)

WRIT_DBL: Write to data block in the load memory

Description

You use the instruction to transfer the contents of a DB or a DB area from the work memory to a DB or a DB area in the load memory (Micro Memory Card). The source DB must be relevant for execution, which means it must not be created with the attribute UNLINKED. It can, however, have been created with the "CREAT_DB" instruction.

To ensure data consistency, you must not change the source area while "WRIT_DBL" is being executed (which means as long as the BUSY parameter has the value TRUE).

The following restrictions apply to the parameters SRCBLK and DSTBLK:

- For an ANY pointer of the type BOOL , the length must be divisible by 8.
- For an ANY pointer of the type STRING , the length must be equal to 1.

If required, you can determine the length of the destination DB with the "TEST_DB (Page 5526)" instruction.

"WRIT_DBL" does not change the checksum of the user program if you write a DB that was created using an instruction. However, when a loaded DB is written, the first entry in this DB changes the checksum of the user program.

Note

"WRIT_DBL" is not suitable for frequent (or cyclical) writing of tags in the load memory. This is because the Micro Memory Card technology limits the number of write accesses that can be made to a Micro Memory Card.

Functional description

The "WRIT_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "WRIT_DBL" with REQ =1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186)

Parameter

The following table shows the parameters of the instruction "WRIT_DBL":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Write request
SRCBLK	Input	ANY	D	Pointer to the DB in the work memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.
DSTBLK	Output	ANY	D	Pointer to the data block in the load memory that is to be written to.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
0081	The destination area is larger than the source area. The source area is written completely to the destination area. The remaining bytes of the destination area will not be changed.
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source area is larger than the destination area. **The destination area is fully written. The remaining bytes of the source area are ignored.**
8092	Incorrect operating mode: While "WRIT_DBL" was active, the CPU went into STOP mode. This error code is supplied at the next transition to RUN. Call "WRIT_DBL" again.
8093	No data block or a data block that is not in the work memory is specified for the SRCBLK parameter.
80B1	No data block is specified for the DSTBLK parameter, or the data block specified there is not an object in the load memory (for example, a DB created using "CREAT_DB (Page 5520)").
80B4	DB with F-attribute must not be changed
80C0	The destination is currently being processed by another instruction or a communication function. Example: You upload a DB from the CPU to the programming device (PG). You want to change the contents of this DB using "WRIT_DBL".

Error code (W#16#...)	Description
80C3	The maximum number of simultaneously active "WRIT_DBL" instructions has already been reached.
8xyy	General error codes See also: Evaluating errors with output parameter RET_VAL (Page 162)

DEL_DB: Delete data block

Description

You use the instruction to delete a data block located in the work memory and, if present, in the load memory of the CPU. The DB to be deleted must not be open in the current or in any lower priority class, in other words, it must not be entered in either of the two DB registers or in the B stack.

Otherwise, the CPU starts OB 121 when the instruction is called. If OB 121 is not present, the CPU switches to STOP mode. For S7-300 (exception: CPU 318), the DB is deleted without calling OB 121.

Note

It is not advisable to delete instance DBs with "DEL_DB" as this always leads to program errors. Therefore, avoid deleting instance DBs with "DEL_DB".

The following table explains when a DB can be deleted with "DEL_DB".

If ...	then use "DEL_DB" ...
the DB was created by calling the "CREAT_DB" instruction,	it can be deleted.
was transferred to the CPU and was not created with the attribute "Only store in load memory"	it can be deleted.
was created with the attribute "Only store in load memory"	• For S7-300 can be deleted: • For S7-400 cannot be deleted:
was created by calling the "CREA_DBL" instruction	can be deleted
The DB was stored on the flash card,	it cannot be deleted.

Interrupt ability

The instruction can be interrupted by priority classes of a higher priority. If the instruction is called again there, then this second call will be aborted and W#16#8091 will be entered in RET_VAL .

Parameter

The following table shows the parameters of the instruction "DEL_DB":

Parameters	Declaration	Data type	Memory area	Description
DB_NUMBER	Input	WORD	I, Q, M, D, L or constant	Number of the DB to be deleted
RET_VAL	Return	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8091	"DEL_DB" calls were nested and the maximum nesting level of the CPU used was exceeded.
8092	The "Delete a DB" function cannot be executed currently because • The "Compress user memory" function is currently active. • You are copying the DB to be deleted from the CPU to an offline project. • The WinAC software CPU has detected an error in the OS of the computer on which WinAC is installed.
80A1	Error in input parameter DB_NUMBER: the actual parameter selected • Is "0" • Is greater than the maximum permitted DB number for the CPU used.
80B1	The DB with the specified number does not exist in the work memory of the CPU.
80B3	The DB is on a flash card.
80B4	The DB could not be deleted. Possible causes: • It belongs to an F-program. • It is an instance DB of a block for S7 communication (for S7-400 only). • It is a technology DB.
80C1	The "Delete a DB" function cannot be executed at this time due to a temporary resource bottleneck.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

CREAT_DB: Create data block

Description

You use the instruction to create a data block with no predefined values in the user program. Instead, the DB contains random data. The instruction creates a data block with a number from a specified area and with a predefined size. The instruction assigns the smallest possible number from the specified area to the DB. To create a DB with a specific number, enter the same number for the upper and lower limits of the specified area. You cannot assign the numbers of the DBs already contained in the user program. The length of the DB must be specified with an even number.

Interrupt ability

The instruction can be interrupted by higher-priority OBs. If a "CREAT_DB" instruction is called in a higher-priority OB, this call will be rejected with the error code W#16#8091.

Parameter

The following table shows the parameters of the instruction "CREAT_DB":

Parameters	Declaration	Data type	Memory area	Description
LOW_LIMIT	Input	WORD	I, Q, M, D, L or constant	The lower limit value is the lowest number in the range of numbers that you can assign to your data block.
UP_LIMIT	Input	WORD	I, Q, M, D, L or constant	The upper limit value is the highest number in the range of numbers that you can assign to your data block.
COUNT	Input	WORD	I, Q, M, D, L or constant	The count value indicates the number of data bytes that you want to reserve for your data block. You must specify an even number of bytes (maximum 65534).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
DB_NUMBER	Output	WORD	I, Q, M, D, L	The data block number is the number of the created data block. In case of an error (bit 15 of RET_VAL was set), the value "0" will be entered in DB_NUMBER .

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8091	You have called CREAT_DB as a nested instruction.
8092	The "Create a DB" function is currently unavailable because • The "Compress user memory" function is currently active. • The number of DBs in the CPU has already reached the maximum possible number. • The H-CPU is running link-up or update functions. • The WinAC software CPU has detected an error in the OS of the computer on which WinAC is installed. • The previous Delete is not completed yet
80A1	Error in the DB number: • The number is "0" • The number exceeds the CPU-specific DB quantity • Lower limit > upper limit
80A2	Error in length of DB: • The length is "0" • The length was specified with an uneven number • The length is greater than the CPU allows
80B1	There is no DB number free.
80B2	There is not enough free memory available.
80B3	There is not enough continuous memory space available (remedy: compress memory!)
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

CREA_DB: Create data block / retentive

Description

You use the instruction to create a data block with no predefined values in the user program. Instead, the DB contains random data. The instruction creates a data block with a number from a specified area and with a predefined size. The instruction assigns the smallest possible number from the specified area to the DB. To create a DB with a specific number, enter the same number for the upper and lower limits of the specified area. You cannot assign the numbers of the DBs already contained in the user program. The length of the DB must be specified with an even number.

Depending on the selection you have made for the ATTRIB parameter, the DB created has the property RETAIN or NON_RETAIN:

- RETAIN (=retentive) means that the DB is created in the retentive part of the work memory. This means the current values of the DB are retained after each power OFF/power ON transition and every restart (warm restart).
- NON_RETAIN (=non-retentive) means that the DB is created in the non-retentive part of the work memory. This means the current values of the DB are undefined after each power OFF/power ON transition and every restart (warm restart).

If no distinction has been made between retentive and non-retentive work memory, the ATTRIB parameter will be ignored. This means the values of the DB will be retained after each power OFF/power ON transition and every restart (warm restart).

Interrupt ability

The "CREA_DB" instruction can be interrupted by higher-priority OBs. If a "CREA_DB" instruction is called in a higher-priority OB, this call will be rejected with the error code W#16#8091 .

Parameter

The following table shows the parameters of the instruction "CREA_DB":

Parameter	Declaration	Data type	Memory area	Description
LOW_LIMIT	Input	WORD	I, Q, M, D, L or constant	The lower limit value is the lowest number in the range of numbers that you can assign to your data block.
UP_LIMIT	Input	WORD	I, Q, M, D, L or constant	The upper limit value is the highest number in the range of numbers that you can assign to your data block.
COUNT	Input	WORD	I, Q, M, D, L or constant	The count value indicates the number of data bytes that you want to reserve for your data block. You must specify an even number of bytes (maximum 65534).
ATTRIB	Input	BYTE	I, Q, M, D, L or constant	DB attributes: • B#16#00: RETAIN • B#16#04: NON_RETAIN
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
DB_NUMBER	Output	WORD	I, Q, M, D, L	The data block number is the number of the created data block. In case of an error (bit 15 of RET_VAL was set), the value "0" will be entered in DB_NUMBER .

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8091	You have called CREA_DB as a nested instruction.
8092	The "Create a DB" function is currently unavailable because • The "Compress user memory" function is currently active. • The WinAC software CPU has detected an error in the OS of the computer on which WinAC is installed.
8094	Invalid value in parameter ATTRIB
80A1	Error in the DB number: • The number is "0" • The number exceeds the CPU-specific DB quantity • Lower limit > upper limit
80A2	Error in length of DB: • The length is "0" • The length was specified with an uneven number • The length is greater than the CPU allows
80B1	There is no DB number free.
80B2	There is not enough free memory available.
80B3	There is not enough continuous memory space available (remedy: compress memory!)
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

CREA_DBL: Create data block in the load memory

Description

You use the instruction to create a new data block in the load memory (Micro Memory Card). The instruction creates a data block with a number from a specified area and with a predefined size. The instruction assigns the smallest possible number from the specified area to the DB. To create a DB with a specific number, enter the same number for the upper and lower limits of the range to be specified. You cannot assign the numbers of the DBs already contained in the user program. If a DB with the same number already exists in work memory and/or load memory, or if the DB exists as a copied version, the instruction is terminated and error information is generated.

Note

With the "TEST_DB (Page 5526)" instruction, you can determine whether a DB with the same number already exists.

The contents of the data area to which the parameter SRCBLK(source block) points are written to the DB. This data area must be a DB or an area from a DB. To ensure data consistency,

you must not change this data area while "CREA_DBL" is being executed (which means as long as the BUSY parameter has the value TRUE).

A DB with the READ_ONLY attribute can only be created and initialized using "CREA_DBL".

"CREA_DBL" does not change the checksum of the user program.

Functional description

The "CREA_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "CREA_DBL" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186)

Parameter

The following table shows the parameters of the instruction "CREA_DBL":

Parameters	Declaration	Data type	Memory area	Description	
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Request to create the DB	
LOW_LIMIT	Input	WORD	I, Q, M, D, L	Low limit of the range used by "CREA_DBL" to assign a number to your DB	
UP_LIMIT	Input	WORD	I, Q, M, D, L	Upper limit of the range used by "CREA_DBL" to assign a number to your DB	
COUNT	Input	WORD	I, Q, M, D, L	The count value specifies the quantity of data bytes you want to reserve for your DB. Here you must specify an even number of bytes.	
ATTRIB	Input	BYTE	I, Q, M, D, L	DB properties:	
				Bit 0 = 1:	UNLINKED: The DB exists only in load memory.
				Bit 1 = 1:	READ_ONLY: The DB is read-only.
				Bit 2 = 1:	NON_RETAIN: The DB is not retentive.
				Bit 3 to 7:	Reserved
SRCBLK	Input	ANY	D	Pointer to the data block whose values will be used to initialize the created DB	
RET_VAL	Return	INT	I, Q, M, D, L	Error information	
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The process is not yet complete.	
DB_NUM	Output	WORD	I, Q, M, D, L	Number of the created DB	

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
0081	The target range is larger than the source range. The complete source range is written to the target range. The remaining bytes of the target range are filled with 0.
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source range is larger than the target range. The complete target range is written. The remaining bytes of the source range are ignored.
8091	You have called "CREA_DBL" as a nested instruction.
8092	The "Create data block" function is currently unavailable because • The "Compress user memory" function is currently active. • The maximum number of blocks on your CPU has already been reached.
8093	No data block or a data block that is not in the work memory is specified for the SRCBLK parameter.
8094	A not yet supported attribute was specified for the ATTRIB parameter.
80A1	DB number error: • The number is 0 or is not within the valid range for the CPU (CPU15xx: 60001 to 60999) • The number exceeds the CPU-specific DB quantity • Low limit > high limit
80A2	DB length error: • The length is "0" • The length is an odd number • The length is greater than permitted by the CPU
80B1	No free DB number
80B2	Insufficient work memory
80BB	Insufficient load memory
80C0	The destination is currently being processed by another instruction or a communication function.
80C3	The maximum number of simultaneously active "CREA_DBL" instructions has already been reached.
8xyy	General error codes, for example: • Source DB does not exist or it is only available as copied version • Source range in DB does not exist See also: Evaluating errors with output parameter RET_VAL (Page 162)

TEST_DB: Test data block

Description

With the instruction you receive on S7-300 information about a data block that is in the CPU work or load memory, on S7-400 information about a data block that is in the CPU work memory. The instruction determines the number of data bytes in the selected DB and checks whether or not the DB is read only.

Parameters

The following table shows the parameters of the instruction "TEST_DB":

Parameters	Declaration	Data type	Memory area	Description
DB_NUMBER	Input	WORD	I, Q, M, D, L or constant	Number of the DB to be tested
RET_VAL	Return	INT	I, Q, M, D, L	Error information
DB_LENGTH	Output	WORD	I, Q, M, D, L	Number of data bytes the selected DB contains.
WRITE_PROT	Output	BOOL	I, Q, M, D, L	Information about the write-protect identifier of the DB ("1" means read only).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
80A1	Error in input parameter DB_NUMBER: the actual parameter selected • Is "0" • Is greater than the maximum permitted DB number for the CPU used.
80B1	The DB with the specified number does not exist on the CPU.
80B2	The DB was created using the keyword UNLINKED.
8xyy	General error information, See also: Evaluating errors with output parameter RET_VAL (Page 162)

4.2.3.11 Table functions

ATT: Add value to table

Description

The instruction supplements the DATA parameter as the next entry in a table and increments the number of entries by one. The table consists of words. You use the instruction to create table entries that are used by "FIFO (Page 5528)" and "LIFO (Page 5532)".

- The first entry in a "FIFO (Page 5528)" or "LIFO (Page 5532)" table specifies the maximum length of the table.
- The second entry in the table specifies the number of existing entries.
- The third entry in the table contains the first word of data.

Note

When you create the table, initialize the first two entries.

Parameter

The following table shows the parameters of the instruction "ATT":

Parameters	Declaration	Data type	Memory area	Description
DATA	Input	WORD	I, Q, M, D, L, P or constant	Data to be entered in the table.
TABLE	Input	*Pointer	I, Q, M, D	Points to the start address of the "FIFO" or "LIFO" table.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

If the number of entries is equal to or greater than the table length, the data will not be added to the table and the signal state of "BR" is set to "0".

Example (LAD)

The "ATT" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, DATA is added as the fifth entry in the table and the number of entries is increased from 4 to 5.

If the "ATT" instruction is executed without errors, the signal states of "ENO" and Q 0.0 will be set to "1".

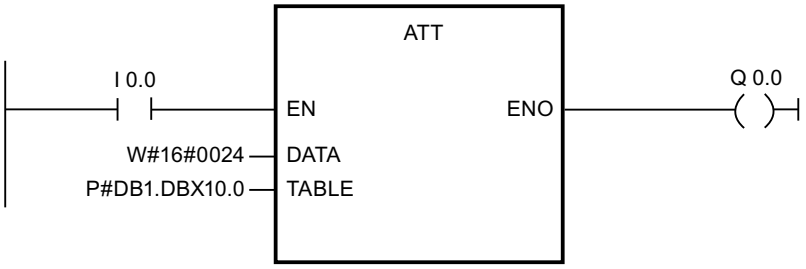

Before processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0004 DBW14 = W#16#0012 DBW16 = W#16#0029 DBW18 = W#16#0090 DBW20 = W#16#0002 DBW22 = W#16#0000 DBW24 = W#16#0000

After processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0005 DBW14 = W#16#0012 DBW16 = W#16#0029 DBW18 = W#16#0090 DBW20 = W#16#0002 DBW22 = W#16#0024 DBW24 = W#16#0000

FIFO: Output first value of the table

Description

The instruction outputs the oldest entry of the "FIFO" table as function value. The number of entries is decremented by one. If there are still entries in the table, these are shifted down. The "FIFO" table consists of words. Use ATT (Page 5527) to enter values in the "FIFO" table.

- The first entry in the table specifies the maximum length of the table.
- The second entry in the table specifies the number of existing entries.
- The third entry in the table contains the first word of data.

Parameter

The following table shows the parameters of the instruction "FIFO":

Parameters	Declaration	Data type	Memory area	Description
TABLE	Input	*Pointer	I, Q, M, D	Points to the start address of the "FIFO" table.
RET_VAL	Return	WORD	I, Q, M, D, L, P	The oldest entry from the "FIFO" table.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

If the "FIFO" table is empty (number of entries = 0), then RET_VAL will remain the same and the signal state of "BR / ENO" will be set to "0".

Example (LAD)

"FIFO" is executed if the signal state at input I 0.0 is 1 (activated). In this example, the oldest entry in the table is returned as the function value (MW 2.0). The number of entries is decremented from 5 to 4, and the remaining entries are shifted down in the table.

If "FIFO" is executed without errors, then the signal states of "BR / ENO" and Q 0.0 will be set to "1".

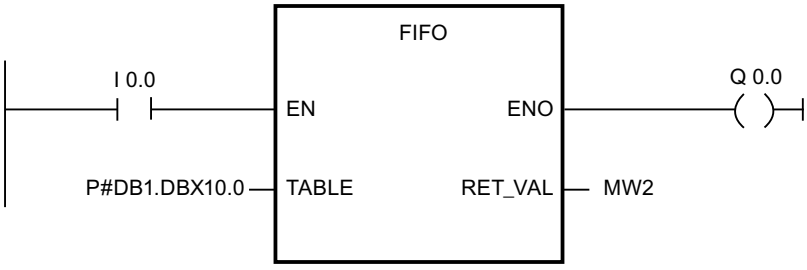

Before processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0005 DBW14 = W#16#0012 DBW16 = W#16#0029 DBW18 = W#16#0090 DBW20 = W#16#0002 DBW22 = W#16#0024 DBW24 = W#16#0000
RET_VAL	MW2 = W#16#0000

After processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0004 DBW14 = W#16#0029 DBW16 = W#16#0090 DBW18 = W#16#0002 DBW20 = W#16#0024 DBW22 = W#16#0024 DBW24 = W#16#0000
RET_VAL	MW2 = W#16#0012

TBL_FIND: Find value in table

Description

You use the instruction to search the memory for patterns. The "TBL_FIND" instruction performs a comparison (CMD parameter) between the source pattern (PATRN parameter) and the entries in the source table (SRC parameter). Starting at the entry specified by the INDX parameter, the instruction searches for the next entry in the table that matches the comparison criteria. The number of the entry is stored in the INDX parameter. If no match is found, the "INDX parameter will point beyond the end of the table and the output of the "TBL_FIND" instruction will be deactivated.

- If CMD = 1, then the "TBL_FIND" instruction searches for the first value that corresponds to the value at the PATRN parameter.
- If CMD = 2, then the "TBL_FIND" instruction searches for the first value that does not correspond to the value at the PATRN parameter.
- The first entry in the table contains the maximum length of the table.
- The second entry in the table contains the first table value.

Note

Make sure that you initialize the first entry (table length) of the table.

Parameter

The following table shows the parameters of the instruction "TBL_FIND":

Parameters	Declaration	Data type	Memory area	Description
SRC	Input	*Pointer	I, Q, M, D	Points to the start address of the table.
PATRN	Input	*Pointer	I, Q, M, D	Points to the pattern to be searched for.
CMD	Input	BYTE	I, Q, M, D, L, P	Specifies the command: • B#16#01 = equal to • B#16#02 = unequal to

Parameters	Declaration	Data type	Memory area	Description
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries: B#16#02 = BYTE B#16#04= WORD B#16#05=INT B#16#06=DWORD B#16#07=DINT B#16#08=REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
INDX	InOut	WORD	I, Q, M, D, L	Table index that provides the following information: • Input: Number of the entry at which the search starts. • Output: Number of the entry that matches the specified pattern.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

If any of the following conditions occur, the table values are not changed. The signal state of "BR / ENO" will be set to "0" and the return value will be set accordingly:

RET_VAL	Explanation
W#16#0007	The value of the INDX parameter is higher than the value of the table length.
W#16#0008	No match was found.
W#16#0009	The E_TYPE and/or CMD parameter(s) is/are invalid.

Example (LAD)

The "TBL_FIND" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, because the E_TYPE parameter = "4", data in the table is stored in words, starting at the address pointed to by the SRC parameter. These words are compared with the pattern "5555", which is stored at the address specified by the PATRN parameter. Because CMD = 1, the instruction searches for the first value in the SRC parameter that matches the pattern. The INDX parameter points to the address where the search should begin. After the instruction executes, the INDX parameter specifies the entry number in the table where a match for the pattern was found.

If the "TBL_FIND" instruction is executed without errors, then the signal states of ENO and Q 0.0 will be set to "1", and RET_VAL is set to the value W#16#0000.

The "MOVE" instruction is simply used to reset the value of "MW2" in each cycle.

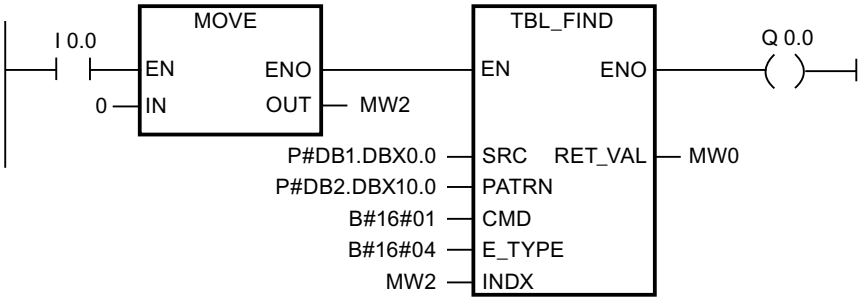

Before processing:

SRC (table length)	DBW0 = W#16#0004 DBW2 = W#16#1111 DBW4 = W#16#3333 DBW6 = W#16#5555 DBW8 = W#16#7777	The first table entry specifies the length of the find instruction. The value 0004 specifies that the next four entries will be searched.
INDX	MW2 = W#16#0000	INDX is transferred from DW0 (before execution) to DW6 (after execution) because the pattern of DW6 matches the pattern 5555 of PATRN .
PATRN	DBW10 = W#16#5555	

After processing:

INDX	MW2 = W#16#0003	

LIFO: Output last value in the table

Description

The instruction outputs the most recent entry of the "LIFO" table as function value. The number of entries is decremented by one. The "LIFO" table consists of words. You can use the ATT (Page 5527) instruction to enter values in the "LIFO" table.

- The first entry in the table specifies the maximum number of entries in the table (table length).
- The second entry in the table specifies the number of existing entries.
- The third entry in the table contains the first word of data.

Parameter

The following table shows the parameters of the instruction "LIFO":

Parameters	Declaration	Data type	Memory area	Description
TABLE	Input	*Pointer	I, Q, M, D	Points to the start address of the "LIFO" table.
RET_VAL	Return	WORD	I, Q, M, D, L, P	The most recent entry from the "LIFO" table.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

If the "LIFO" table is empty (number of entries = 0), then RET_VAL will remain the same and the signal state of "BR / ENO" will be set to "0".

Example (LAD)

The "LIFO" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, the most recent entry in the "LIFO" table is returned as the function value (MW 2.0). The number of entries is decremented from 5 to 4.

If "LIFO" is executed without errors, then the signal states of "ENO" and Q 0.0 will be set to "1".

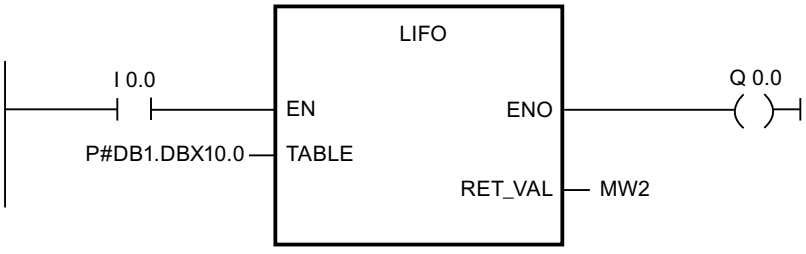

Before processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0005 DBW14 = W#16#0012 DBW16 = W#16#0029 DBW18 = W#16#0090 DBW20 = W#16#0002 DBW22 = W#16#0024 DBW24 = W#16#0000
RET_VAL	MW2 = W#16#0000

After processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0004 DBW14 = W#16#0012 DBW16 = W#16#0029 DBW18 = W#16#0090 DBW20 = W#16#0002 DBW22 = W#16#0024 DBW24 = W#16#0000
RET_VAL	MW2 = W#16#0024

TBL: Execute table instruction

Description

The instruction executes the specified instruction (CMD parameter) with the source table and writes the result to the same entry in the table.

- The first entry in the table specifies the maximum number of entries in the table (table length).
- The second entry in the table contains the first table value.
- If the E_TYPE parameter is set to REAL , the CMD value for ones complement is invalid.

Note

Initialize the first entry when you create the table.

Parameter

The following table shows the parameters of the instruction "TBL":

Parameters	Declaration	Data type	Memory area	Description
SRC	Input	*Pointer	I, Q, M, D	Points to the start address of the table.
CMD	Input	BYTE	I, Q, M, D, L, P	Specifies the instruction to be executed. The following instructions are valid: B#16#03 = Ones complement B#16#04 = Delete B#16#05 = Negate B#16#06 = Square root

Parameters	Declaration	Data type	Memory area	Description
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

If the CMD parameter or the E_TYPE parameter is invalid, and/or if the CMD parameter and the E_TYPE parameter do not match, then the values in the table will remain the same. The signal state of BR / ENO is set to "0" and RET_VAL will be set to the value W#16#0008 .

Example (LAD)

The "TBL" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, the SRC parameter points to the address in the data block that will be processed by the instruction. Because the E_TYPE parameter is 4, data in the table will be stored in words starting at the address specified by the SRC parameter. Because CMD is 4 (clear), all the words in the table will be cleared (set to "0") when the "TBL" instruction executes. The table length value of "5" in the first table entry causes the next five table entries to be cleared.

If the "TBL" instruction is executed without error, the signal states of ENO and Q 0.0 will be set to "1", and RET_VAL will be set to the value W#16#0000.

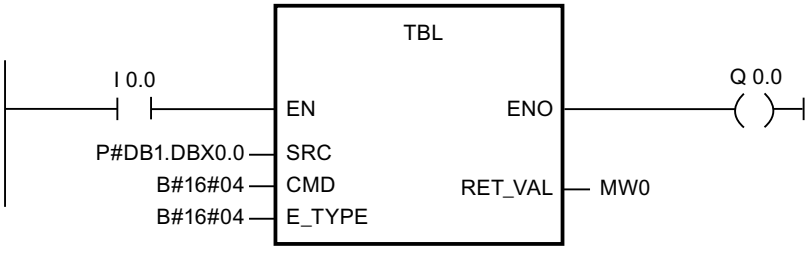

Before processing:

SRC (table length)	DBW0 = W#16#0005 DBW2 = W#16#2000 DBW4 = W#16#3000 DBW6 = W#16#4000 DBW8 = W#16#5000 DBW10 = W#16#6000

After processing:

SRC (table length)	DBW0 = W#16#0005 DBW2 = W#16#0000 DBW4 = W#16#0000 DBW6 = W#16#0000 DBW8 = W#16#0000 DBW10 = W#16#0000

TBL_WRD: Copy value from table

Description

The instruction "TBL_WRD" (copy value from table) copies the entry specified by the INDX parameter from the SRC table to the DEST output. The value of INDX will be incremented if the value is less than the maximum length specified in the first word of the (SRC[0]) table. If the INDX parameter specifies the last table entry, then the output bit will be set to "0" after execution.

- The first entry in the table specifies the maximum number of entries in the table (table length).
- The second entry in the table contains the first table value.

Note

Initialize the first entry when you create the table.

Parameter

The following table shows the parameters of the instruction "TBL_WRD":

Parameter	Declaration	Data type	Memory area	Description
SRC	Input	*Pointer	I, Q, M, D	Points to the start address of the table.
DEST	Input	*Pointer	I, Q, M, D	Points to the start address of the destination.

Parameter	Declaration	Data type	Memory area	Description
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: • B#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
Q	Output	BOOL	Q, M, D, L	Returns the value "0" if the INDX variable contains the last value of the table while calling "TBL_WRD".
INDX	InOut	WORD	I, Q, M, L	Number of the entry to be copied.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

If any of the following conditions occur, "TBL_WRD" will not be executed. The signal state of ENO is set to "0" and the return value will be set accordingly:

RET_VAL	Explanation
W#16#0007	The index is "0".
W#16#0008	E_TYPE is invalid.
W#16#0009	The index is outside the range of the table.

Example

"TBL_WRD" is executed if the signal state at input I 0.0 is 1 (activated). In this example, because E_TYPE = 4 the data word stored in the table entry referenced by SRC is copied to the entry referenced by DEST. The value of INDX points to the table entry to be copied. After "TBL_WRD" executes, the value of INDX is automatically incremented by one entry. If the content of INDX is not the last table entry when "TBL_WRD" is called, output ENO is set to "1" after execution.

If "TBL_WRD" is executed without errors, then the signal states of ENO will be set to "1" and RET_VAL will be set to W#16#0000.

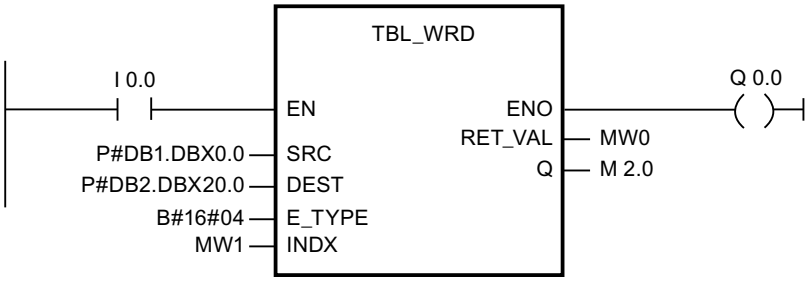

Before processing:

SRC(table length)	DBW0 = W#16#0004 DBW2 = W#16#2000 DBW4 = W#16#3000 DBW6 = W#16#4000 DBW8 = W#16#5000
INDX	MW1 = W#16#0001
DEST	DBW20 = W#16#0000

Before processing:

INDX	MW1 = W#16#0002
DEST	DBW20 = W#16#2000

WRD_TBL: Link value logically with table element and save

Description

The instruction executes the specified instruction (CMD parameter) with the source item and the entry in the table referenced by the INDX parameter. The instruction executes only as long as the INDX parameter is less than the length of the table stored in the first word of the table.

- The first entry in the table specifies the maximum number of entries in the table (table length).
- The second entry in the table contains the first table value.
- If the data type specified by the E_TYPE parameter is REAL, then the CMD parameter can only be "Transfer".

Note

Initialize the first entry when you create the table.

Parameter

The following table shows the parameters of the instruction "WRD_TBL":

Parameters	Declaration	Data type	Memory area	Description
SRC	Input	*Pointer	I, Q, M, D	Points to the source data
TABLE	Input	*Pointer	I, Q, M, D	Points to the start of the table.
CMD	Input	BYTE	I, Q, M, D, L, P	Indicates the instruction to be executed. The following instructions are valid: B#16#0E = Transfer B#16#07 = AND logic operation B#16#08 = OR logic operation B#16#09 = EXCLUSIVE OR logic operation
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns a value W#16#0000 if the instruction executes without errors.
Q	Output	BOOL	Q, M, D, L	Is set to "0" if the tag at the INDX parameter contains the number of the last entry in the table.
INDX	InOut	WORD	I, Q, M, D, L	Number of the entry on which the instruction will be executed.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

If any of the following conditions occur, then the "WRD_TBL" instruction will not be executed. The signal state of ENO is set to "0" and the return value will be set accordingly:

RET_VAL	Explanation
W#16#0007	The index is 0.
W#16#0008	CMD parameter or E_TYPE parameter is invalid or CMD parameter and E_TYPE parameter do not match.
W#16#0009	The index is outside the range of the table.

Example (LAD)

The "WRD_TBL" instruction will be executed if the signal state at input I 0.0 is 1 (activated). Because the E_TYPE parameter in this example is "6", the double word of the data in the table is saved at the address specified by TABLE . The first word in the table indicates that the table contains three double words. The INDX value points to the table entry that is to be modified. Because CMD is 8, the instruction performs an OR operation with the value to which the INDX parameter points. Because INDX is 2, the second double word (66665544) is ORed with the value to which the SRC parameter (11111111) points. After the instruction executes, the result of the OR operation (77775555) is written back to the table, and the value of INDX will be automatically incremented by one entry. If the INDX parameter points to the last table entry, the "Q" output bit is set to "0" after execution of the instruction. In this example, the value of INDX is not set to the last entry in this table and the "Q" output is therefore set to "1" after execution.

If the "WRD_TBL" instruction is executed without errors, then the signal states of ENO and Q 0.0 will be set to "1", and RET_VAL is set to the value W#16#0000.

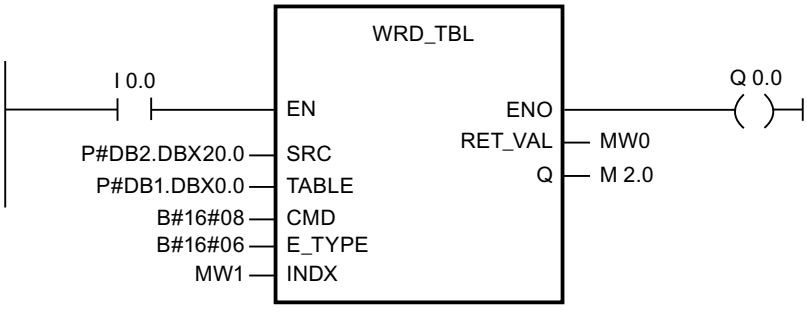

Before processing:

SRC	DBD20 = DW#16#11111111
TABLE (table length)	DBW0 = W#16#0003 DBD2 = DW#16#99998877 DBD6 = DW#16#66665544 DBD10 = DW#16#33332222
INDX	MW1 = W#16#0002

After processing:

TABLE (table length)	DBW0 = W#16#0003 DBD2 = DW#16#99998877 DBD6 = DW#16#77775555 DBD10 = DW#16#33332222
INDX	MW1 = W#16#0003

DEV: Calculate standard deviation

Description

The instruction calculates the standard deviation of a group of values stored in a table ("TBL (Page 5534)" instruction). The result will be stored in OUT . The standard deviation is calculated according to the following formula:

$$\text{Standard deviation} = \sqrt{\frac{(N * Sqsum) - Sum^2}{N * (N - 1)}}$$

Where:

- Sum = the sum of the values in the "TBL (Page 5534)" instruction
- N = the number of the values in the "TBL (Page 5534)" instruction
- Sqsum = the sum of all values in the "TBL (Page 5534)" instruction squared

All calculations use IEEE floating point values, with any necessary type conversions being done automatically by the "DEV" instruction.

- The first entry in the table contains the number of entries in the table (table length).
- The second entry in the table contains the first table value.
- The size of the table entries and the calculated value (OUT parameter) are specified by the E_TYPE parameter.

Parameter

The following table shows the parameters of the instruction "DEV":

Parameters	Declaration	Data type	Memory area	Description
TBL	Input	*Pointer	I, Q, M, D	**Points to the start address of a table of values.
OUT	Input	*Pointer	I, Q, M, D	**Points to the address of the computed standard deviation value.
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: B#16#05 = INT B#16#07 = DINT B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
* Double word pointer format for area-crossing register indirect addressing ** Source and destination block must be identical				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

If any of the following conditions occur, then the "DEV" instruction will not be executed. The signal state of BR / ENO will be set to "0" and the return value will be set accordingly:

RET_VAL	Explanation
W#16#0001	Invalid memory type for a parameter of "DEV".
W#16#0002	The E_TYPE parameter is invalid.
W#16#0004	The table length is zero.

Example (LAD)

The "DEV" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, there are five table values. This specifies the first word in the table. The table values are of the REAL data type. This fact is indicated by the E_TYPE parameter.

If "DEV" is executed without errors, then the signal states of ENO and Q 0.0 will be set to "1", and RET_VAL will be set to "W#16#0000".

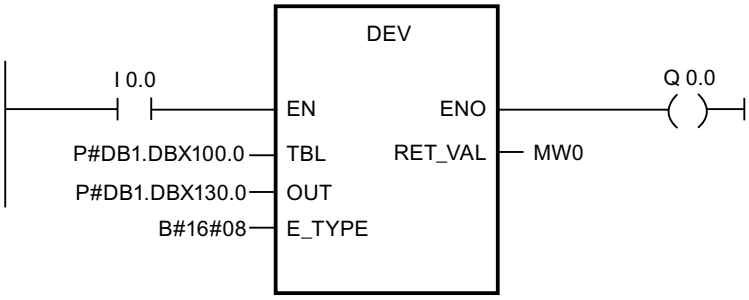

Before processing:

TBL (table length)	DBW100 = W#16#0005 DBD102 = 2.0 DBD106 = 4.0 DBD110 = 8.0 DBD114 = 16.0 DBD118 = 32.0
OUT	DBD130 = 0.0

After processing:

OUT	DBD130 = 12.19836055

CDT: Correlated data tables

Description

The instruction compares an input value (IN parameter) with an existing table of input values (IN_TBL parameter) and locates the first value that is greater than or equal to the input value. Based on its index, the located value is then used to copy the corresponding output value (OUT parameter) to the table of output values (OUT_TBL parameter).

- The values in the input table must be sorted in ascending order. This means that the lowest value is in the first table entry, the highest value is in the last table entry.
- The size of the input value, the table values and output values are defined by the E_TYPE parameter.
- The first entry in the table contains the number of entries of the table (table length).
- The second entry in the table contains the first table value.
- The number of elements in both tables must be identical and be greater than zero.

Note

Initialize the first entry when you create each table.

Parameter

The following table shows the parameters of the instruction "CDT":

Parameters	Declaration	Data type	Memory area	Description
IN_TBL	Input	*Pointer	I, Q, M, D	Points to the start of the input table.
OUT_TBL	Input	*Pointer	I, Q, M, D	Points to the start of the output table.
IN	Input	*Pointer	I, Q, M, D	Points to the input table.
OUT	Input	*Pointer	I, Q, M, D	Points to the output table.
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: B#16#05 = INT B#16#07 = DINT B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

If any of the following conditions occur, CDT will not be executed. The signal state of BR / ENO will be set to "0" and the return value will be set accordingly:

RET_VAL	Explanation
W#16#0001	Invalid memory type specified for a parameter of CDT .
W#16#0002	The E_TYPE parameter is invalid.
W#16#0003	The input and output table lengths do not match.
W#16#0004	The table length is zero.
W#16#0007	No value in the IN_TBL parameter is greater than or equal to the input value.

Example (LAD)

The "CDT" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, the IN_TBL parameter and the OUT_TBL parameter each have five table entries. This is specified by the first word of the respective table. The data type of the table values is INTEGER. This fact is indicated by the E_TYPE parameter. The value of the IN parameter is 22. The value of the IN_TBL parameter that is greater than or equal to "22" is "64" which has an index of "5". The corresponding value in OUT_TBL is "25". The value "25" is therefore written in the OUT parameter.

If the "CDT" instruction is executed without errors, then the signal states of ENO and Q 0.0 will be set to "1" and RET_VAL will be set to W#16#0000 .

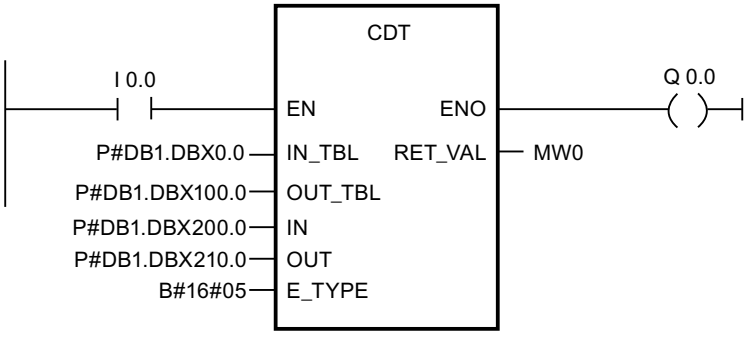

Before processing:

IN_TBL (table length)	DBW0 = W#16#0005 DBW2 = 2 DBW4 = 4 DBW6 = 8 DBW8 = 16 DBW10 = 64
OUT_TBL (table length)	DBW100 = W#16#0005 DBW102 = 5 DBW104 = 10 DBW106 = 15 DBW108 = 20 DBW110 = 25
IN	DBW200 = 22
OUT	DBW210 = 0

After processing:

OUT DBW210 = 25

TBL_TBL: Link tables

Description

The instruction executes the command specified at the CMD parameter with the corresponding entries of the two source tables (parameters TBL1 and TBL2) and writes the result in the corresponding entries of the destination table (DEST_TBL).

- The data types INT, DINT and REAL are only valid for math instructions.
- The first entry in the table contains the number of entries of the table (table length).
- The number of entries in all of the tables must be the same and be greater than zero.

Note

Initialize the first entry when you create each table.

Parameter

The following table shows the parameters of the instruction "TBL_TBL":

Parameters	Declaration	Data type	Memory area	Description
TBL1	Input	*Pointer	I, Q, M, D	Points to the start of the first source table.
TBL2	Input	*Pointer	I, Q, M, D	Points to the start of the second source table.
DEST_TBL	Input	*Pointer	I, Q, M, D	Points to the start of the destination table.

Parameters	Declaration	Data type	Memory area	Description
CMD	Input	BYTE	I, Q, M, D, L, P	Specifies the command to be performed. The following commands are valid: B#16#07 = AND logic operation B#16#08= OR logic operation B#16#09 = EXCLUSIVE OR logic operation B#16#0a = Add B#16#0b = Subtract B#16#0c = Multiply B#16#0d = Divide
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

If any of the following conditions occur, then the "TBL_TBL" instruction will not be executed. The signal state of BR / ENO will be set to "0" and the return value will be set accordingly:

RET_VAL	Explanation
W#16#0001	An invalid memory area was specified for a parameter of the "TBL_TBL" instruction.
W#16#0002	The E_TYPE parameter is invalid.
W#16#0003	The input and output table lengths do not match.
W#16#0004	The table length is zero.
W#16#0005	The parameters E_TYPE and CMD do not match.
W#16#0006	The CMD parameter is invalid.

Example (LAD)

The "TBL_TBL" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, each of the tables has three table entries. This is specified by the first word of the respective table. The data type of the table values is WORD. E_TYPE specifies this. CMD specifies the command with which TBL1 and TBL2 are to be linked.

If the "TBL_TBL" instruction is executed without errors, then the signal states of ENO and Q 0.0 are set to "1", and RET_VAL is set to the value W#16#0000.

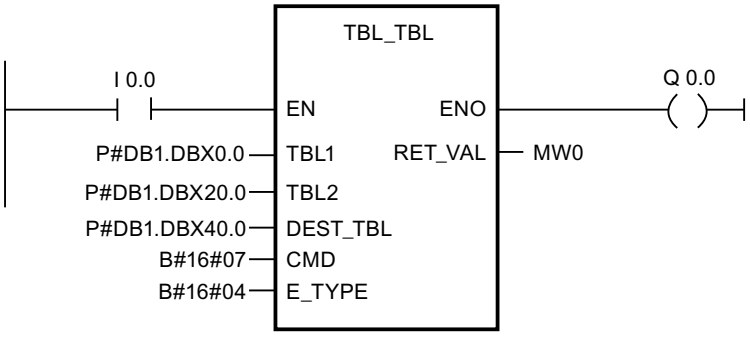

Before processing:

TBL1 (table length)	DBW0 = W#16#0003 DBW2 = W#16#00FF DBW4 = W#16#FF00 DBW6 = W#16#FFFF
TBL2 (table length)	DBW20 = W#16#0003 DBW22 = W#16#1111 DBW24 = W#16#2222 DBW26 = W#16#3333
DEST_TBL (table length)	DBW40 = W#16#0003 DBW42 = W#16#0000 DBW44 = W#16#0000 DBW46 = W#16#0000

After processing:

DST_TBL (table length)	DBW40 = W#16#0003 DBW42 = W#16#0011 DBW44 = W#16#2200 DBW46 = W#16#3333

PACK: Collect/distribute table data

Description

The instruction transfers data located between any addresses and a table. The direction of transfer is specified by the DIR parameter. Each operation of the PACK instruction processes up to five blocks with data (P_DATA1 to P_DATA5). If the DIR parameter has the value "to", the PACK instruction transfers data from the addresses to the specified table. If the DIR parameter has the value "from", data is distributed from the table to the specified addresses.

The rules for transferring data "to" a table are as follows:

- Single bits (BOOL) are transferred to the next available bit in the table.
- 8-bit data types are transferred to the next available byte in the table. When a byte is written to the table, unused bits in the previous byte are filled with zeros.
- 16- and 32-bit data types are transferred to the next available word in the table. When a word is written to the table, unused bits in the previous word are filled with zeros.

The rules for transferring data "from" a table are as follows:

- Sections of a table cannot be skipped.
- All specified data of the BOOL type will be transferred from the table.
- 8-bit data types are transferred to the first available byte in the table. Unused bits in the previous byte of the table are not included in the byte that is transferred from the table.
- 16- and 32-bit data types are transferred from the first available word in the table. Unused bits in the previous word of the table are not included in the a word that is transferred from the table.

The following data types are valid for the ANY pointer and are supported by "PACK": BOOL, WORD, INT, BYTE, DINT, REAL, CHAR, DWORD.

Parameter

The following table shows the parameters of the instruction "PACK":

Parameters	Declaration	Data type	Memory area	Description
TABLE	Input	*Pointer	I, Q, M, D	Points to the start of the table.
P_DATA1	Input	ANY	I, Q, M, D	Points to the start of the block whose data will be transferred.
P_DATA2	Input	ANY	I, Q, M, D	Points to the start of the block whose data will be transferred.
P_DATA3	Input	ANY	I, Q, M, D	Points to the start of the block whose data will be transferred.
P_DATA4	Input	ANY	I, Q, M, D	Points to the start of the block whose data will be transferred.
P_DATA5	Input	ANY	I, Q, M, D	Points to the start of the block whose data will be transferred.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
DIR	Static	BOOL	I, Q, M, D, L	Direction of transfer. The following signal states are possible: 0=to, 1=from.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

If any of the following error conditions occurs, then the "PACK" instruction will not be executed. The signal state of BR / ENO will be set to "0" and ERR_CODE will be set accordingly:

ERR_CODE	Explanation
W#16#0001	An invalid memory area was specified for a parameter of the "PACK" instruction.
W#16#0002	The "E_TYPE" instruction is invalid.

Example (LAD)

"PACK" is executed if the signal state at input I 0.0 is 1 (activated). In this example, there are four blocks with data being transferred "to" the table.

If "PACK" is executed without errors, then the signal states of ENO and Q 0.0 will be set to "1", and ERR_CODE will be set to "W#16#0000".

Note

You can initialize static parameters with the Data Block Editor.

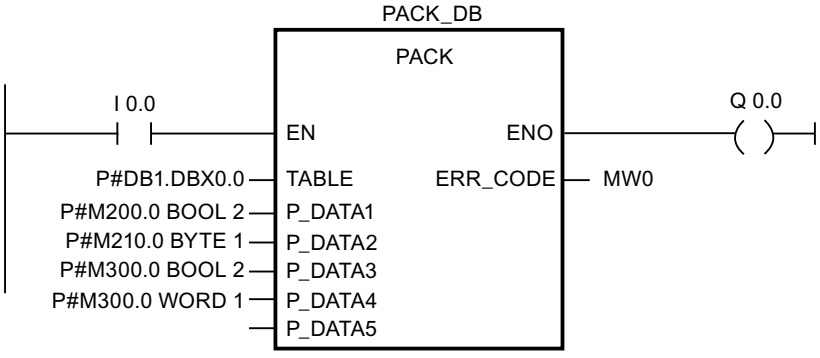

Before processing:

TABLE	DBB0 = B#16#00 DBB1 = B#16#00 DBB2 = B#16#00 DBB3 = B#16#00 DBB4 = B#16#00 DBB5 = B#16#00
P_DATA1	M 200.0 = TRUE M 200.1 = TRUE
P_DATA2	MB210 = B#16#FF
P_DATA3	M 300.0 = TRUE M 300.1 = TRUE
P_DATA4	MW330 = W#16#FFFF

Instance DB PACK_DB	
DIR	DBX58.0 = FALSE

After processing:

TABLE	DBB0 = B#16#03 DBB1 = B#16#FF DBB2 = B#16#03 DBB3 = B#16#00 DBB4 = B#16#FF DBB5 = B#16#FF

4.2.3.12 Addressing

GEO_LOG: Determine start address of a module

Description

The associated module slot of the module is known from the channel of a signal module. With the "GEO_LOG" you determine the associated start address of the module, which means the smallest I or O address.

If you use the "GEO_LOG" instruction on power modules or modules with packed addresses (ET 200S), then the diagnostics address will be returned.

Parameter

The following table shows the parameters of the instruction "GEO_LOG":

Parameters	Declaration	Data type	Memory area	Description
MASTER	Input	INT	I, Q, M, D, L or constant	Area ID: • 0, if the slot in a centralized configuration is located in: Rack 0 to 3 (for S7-300) or 0 to 21 (for S7-400) • 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	Input	INT	I, Q, M, D, L or constant	• Number of the rack if the area identifier = 0. • Device number of field device if area ID> 0
SLOT	Input	INT	I, Q, M, D, L or constant	Slot no.

Parameters	Declaration	Data type	Memory area	Description
SUBSLOT	Input	INT	I, Q, M, D, L or constant	Sub-module slot (if no sub-module can be inserted, 0 must be entered here)
RET_VAL	Return	INT	I, Q, M, D, L	Error information
LADDR	Output	WORD	I, Q, M, D, L	Start address of the module Bit 15 of LADDR indicates whether an input address (bit 15 = 0) or an output address (bit 15 = 1) will be present.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8094	No subnet was configured with the specified SUBNETID .
8095	Illegal value for STATION parameter
8096	Illegal value for SLOT parameter
8097	Illegal value for SUBSLOT parameter
8099	The slot is not configured.
809A	The interface module address is not configured for the selected slot.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

LOG_GEO: Determine the slot belonging to a logical address

Description

With the instruction you determine the module slot as well as the offset in the user data address space belonging to a logical address.

Parameter

The following table shows the parameters of the instruction "LOG_GEO":

Parameters	Declaration	Data type	Memory area	Description
LADDR	Input	WORD	I, Q, M, D, L or constant	Any logical address of the module In bit 15 you indicate whether an input address (bit 15 = 0) or an output address (bit 15 = 1) is present.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
AREA	Output	INT	I, Q, M, D, L	Area ID: indicates how the remaining output parameters are to be interpreted.

Parameters	Declaration	Data type	Memory area	Description
MASTER	Output	INT	I, Q, M, D, L	Area ID: • 0, if the slot is located in a centralized configuration: Rack 0 to 3 (for S7-300) or 0 to 21 (for S7-400) • 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	Output	INT	I, Q, M, D, L	No. of the rack when: • Area ID = 0 • Device number of field device if area ID> 0
SLOT	Output	INT	I, Q, M, D, L	Slot no.
SUBSLOT	Output	INT	I, Q, M, D, L	Interface module number
OFFSET	Output	INT	I, Q, M, D, L	Offset in user data address area of the associated module

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter AREA

Value of AREA	System	Meaning of MASTER, STATION, SLOT, SUBSLOT and OFFSET
0	S7-400	MASTER: 0 STATION: Rack no. SLOT: Slot no. SUBSLOT: 0 OFFSET: Difference between the logical address and the logical base address.
1	S7-300	MASTER: 0 STATION: Rack no. SLOT: Slot no. SUBSLOT: 0 OFFSET: Difference between the logical address and the logical base address.
2	PROFIBUS DP	MASTER: DP master ID STATION: Device number SLOT: Slot no. in the station SUBSLOT: 0 OFFSET: Offset in user data address area of the associated module
	PROFINET IO	MASTER: PROFINET IO system ID STATION: Device number SLOT: Slot no. in the station SUBSLOT: Sub-module number OFFSET: Offset in user data address area of the associated module

Value of AREA	System	Meaning of MASTER, STATION, SLOT, SUBSLOT and OFFSET
3	S5-P area	MASTER: 0 STATION: Rack no. SLOT: Slot number of the adaptation capsule SUBSLOT: 0 OFFSET: Address in the S5 x area
4	S5-Q area	MASTER: 0 STATION: Rack no. SLOT: Slot number of the adaptation capsule SUBSLOT: 0 OFFSET: Address in the S5 x area
5	S5-IM3 area	MASTER: 0 STATION: Rack no. SLOT: Slot number of the adaptation capsule OFFSET: Address in the S5 x area
6	S5-IM4 area	MASTER: 0 STATION: Rack no. SLOT: Slot number of the adaptation capsule SUBSLOT: 0 OFFSET: Address in the S5 x area

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

RD_LGADR: Determine all logical addresses of a module

Description

You use a logic address of a data block, central submodule or a submodule for PNIO. You use the instruction to determine all the declared logical addresses of this module and the submodule. You have previously configured the assignment of logical addresses to modules and submodules. The "RD_LGADR" instruction enters the determined logical addresses in the PEADDR or PAADDR parameter in ascending order.

Parameter

The following table shows the parameters of the instruction "RD_LGADR":

Parameters	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#00: Bit15 of LADDR specifies whether an input (Bit 15=0) or output address (Bit 15=1) is present. • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ)
LADDR	Input	WORD	I, Q, M, D, L or constant	One logical address
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PEADDR	Output	ANY	I, Q, M, D, L	Array for the PI addresses, array elements must be of the WORD data type.
PECOUNT	Output	INT	I, Q, M, D, L	Number of returned PI addresses
PAADDR	Output	ANY	I, Q, M, D, L	Array for the PQ addresses, array elements must be of the WORD data type.
PACOUNT	Output	INT	I, Q, M, D, L	Number of returned PQ addresses

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid or illegal value for the IOID parameter.
80A0	Error in the output parameter PEADDR: The data type of the array elements is not WORD. (This error code only exists for S7-400 and CPU 318.)
80A1	Error in the output parameter PAADDR: The data type of the array elements is not WORD. (This error code only exists for S7-400 and CPU 318.)
80A2	Error in the output parameter PEADDR: The specified array could not accommodate all the logical addresses.
80A3	Error in the output parameter PAADDR: The specified array could not accommodate all the logical addresses.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

GADR_LGC: Query logical start address of a module

Description

Based on the channel of a signal module, the corresponding module slot and the offset in the user data address area of the module are known. With the "GADR_LGC" instruction you determine the associated logical base address of the module, which means the smallest I or Q address.

If you use the "GADR_LGC" instruction on power modules or modules with packed addresses (ET 200S), then the diagnostics address will be returned.

Parameter

The following table shows the parameters of the instruction "GADR_LGC":

Parameters	Declaration	Data type	Memory area	Description
SUBNETID	Input	BYTE	I, Q, M, D, L or constant	Area ID: • 0, if the slot is in one of the racks 0 (central rack) or 1 to 21 (expansion racks). • DP master system ID of the corresponding distributed I/O system if the slot is in a distributed I/O device.
RACK	Input	WORD	I, Q, M, D, L or constant	• Number of the rack if the area identifier is 0. • Device number of the distributed I/O device if the area identifier > 0.
SLOT	Input	WORD	I, Q, M, D, L or constant	Slot no.
SUBSLOT	Input	BYTE	I, Q, M, D, L or constant	Sub-module slot (if no sub-module can be inserted, 0 must be entered here)
SUBADDR	Input	WORD	I, Q, M, D, L or constant	Offset in the user data address area of the module
RET_VAL	Return	INT	I, Q, M, D, L	Error information
IOID	Output	BYTE	I, Q, M, D, L	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) In case of a mixed module, the instruction supplies the area identifier of the lower address. If the addresses are equal, then the instruction will supply the identifier B#16#54.
LADDR	Output	WORD	I, Q, M, D, L	Logical base address of the module

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8093	Invalid value at parameter SUBNETID (the "GADR_LGC" instruction is not valid for PROFINET IO).
8094	No subnet was configured with the specified SUBNETID .
8095	Illegal value for RACK parameter
8096	Illegal value for SLOT parameter
8097	Illegal value for SUBSLOT parameter
8098	Illegal value for SUBADDR parameter
8099	The slot is not configured.
809A	The sub-address of the selected slot is not configured (only possible with central IO devices for CPU and IM).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

LGC_GADR: Determine the slot belonging to a logical address

Description

With the instruction you determine the module slot as well as the offset in the user data address space belonging to a logical address.

Note

The "LGC_GADR" instruction cannot be used on a module with packed addresses (ET 200S).

Parameters

The following table shows the parameters of the instruction "LGC_GADR":

Parameters	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#00: Bit15 of LADDR specifies whether an input (Bit15=0) or output address (Bit15=1) will be present. • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, you will have to specify the area identifier of the lower address. If the addresses are identical, you will have to specify B#16#54 .
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical address For a mixed module, the lower of the two addresses must be specified.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
AREA	Output	BYTE	I, Q, M, D, L	Area ID: indicates how the remaining output parameters are to be interpreted.
RACK	Output	WORD	I, Q, M, D, L	Rack no.
SLOT	Output	WORD	I, Q, M, D, L	Slot no.
SUBADDR	Output	WORD	I, Q, M, D, L	Offset in user data address area of the associated module.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Note

If you are using an S7-400H automation system in redundant mode and specify the logical address of a module in a connected DP slave when calling the "LGC_GADR" instruction in the LADDR parameter, then the DP master system ID of the active channel will be supplied in the high byte of the RACK parameter. If no active channel exists, the DP master system ID for the associated DP master system is output to the master CPU.

Parameter AREA

The output parameter AREA specifies how the output parameters RACK, SLOT and SUBADDR are to be interpreted. The following table explains this dependency.

Value of AREA	System	Meaning of RACK, SLOT and SUBADDR
0	S7-400	RACK : Rack no. SLOT : Slot no. SUBADDR : Difference between the logical address and the logical base address.
1	S7-300	RACK : Rack no. SLOT : Slot no. SUBADDR : Difference between the logical address and the logical base address.
2	DP	RACK : (low byte): Device number RACK : (high byte): DP master ID SLOT : Slot no. in the station SUBADDR : Offset in user data address area of the associated module
3	S5-P area	RACK : Rack no. SLOT : Slot number of the adaptation capsule SUBADDR : Address in the S5 x area
4	S5-Q area	RACK : Rack no. SLOT : Slot number of the adaptation capsule SUBADDR : Address in the S5 x area
5	S5-IM3 area	RACK : Rack no. SLOT : Slot number of the adaptation capsule SUBADDR : Address in the S5 x area
6	S5-IM4 area	RACK : Rack no. SLOT : Slot number of the adaptation capsule SUBADDR : Address in the S5 x area

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid or illegal value for the IOID parameter.
8093	This instruction is not permitted for the module selected by the parameters IOID and LADDR :
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

4.2.3.13 Additional functions (S7-300)

iSlave (S7-300)

SET_ADDR: Setting network address as own iSlave (S7-300)

Description

You use this instruction to modify the MAC address of an iSlave DP interface. The addressing is performed via the diagnostics address of the DP interface. For the DP master, the DP slave with the oldest address is discarded from the bus and a DP slave with the new address is included in the bus.

- If the address is already assigned by a station in the DP network, both DP slaves will be discarded from the network.
- If the address is also used as MPI address for the programming device connection, access to the device is possibly disabled. The information about whether the devices are at all accessible is displayed in the "Accessible Stations" dialog.

The network address is not written in the SDB (system data block) and will not be loaded in the work memory. The network address is maintained when the CPU goes to STOP and a warm restart is effected. After a reset or a cold restart, the originally configured address will be used.

The instruction cannot be used for PROFINET IO, because the Ethernet address can be defined as disjoint. If the instruction is used with PROFINET IO, then the error code 16#809B will be output.

Functional description

The "SET_ADDR" instruction works asynchronously, which means its execution extends over multiple calls. You start the job by calling "READ_DBL" with REQ =1.

The output parameters RET_VAL and BUSY indicate the status of the job.

Parameter

The following table shows the parameters of the "SET_ADDR" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ=1: Function call
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical base address
ADDR	Input	BYTE	I, Q, M, D, L or constant	New device address
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The job is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
700x	The job is not yet complete. Parameter BUSY=1
8091	New device address (ADDR) is invalid.
8092	Station is not an individual slave address of the I-slave
8093	LADDR is invalid or no interface.
809B	Instruction cannot be executed (for example, the interface is not a DP slave or is not active).
80C3	Insufficient resources (for example, multiple calling of instructions with different parameters).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

4.2.4 Technology

4.2.4.1 PID Control (S7-300, S7-400)

PID basic functions (S7-300, S7-400)

CONT_C (S7-300, S7-400)

Description CONT_C (S7-300, S7-400)

The CONT_C instruction is used on SIMATIC S7 automation systems to control technical processes with continuous input and output variables. You can assign parameters to enable or disable sub-functions of the PID controller and adapt it to the process. In addition to the functions in the setpoint and process value branches, the instruction implements a complete PID controller with continuous output value output and the option of manually influencing the value of the output value.

Application

You can use the controller as a PID fixed setpoint controller, or in multi-loop control systems, also as a cascade, blending or ratio controller. The functions of the controller are based on the PID control algorithm of the sampling controller with an analog signal, if necessary extended by including a pulse shaper stage to generate pulse-width modulated output signals for two or three step controllers with proportional actuators.

Call

The CONT_C instruction has an initialization routine that is run through when input parameter COM_RST = TRUE is set. During initialization, the integral action is set to the initialization value I_ITVAL. All the signal outputs are set to zero. COM_RST = FALSE has to be set after the initialization routine has been completed.

The calculation of the values in the control blocks is only correct if the block is called at regular intervals. You should therefore call the control blocks in a cyclic interrupt OB (OB 30 to OB 38). Enter the sampling time in the CYCLE parameter.

If you call the instruction CONT_C as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for CONT_C directly in the multiple instance DB and commission it via a watch table.

Error information

The error message word RET_VAL is not evaluated by the block.

See also

Differences compared to CONT_C S7-300/400 (Page 2968)

How CONT_C works (S7-300, S7-400)

Setpoint branch

The setpoint is entered in floating-point format at the SP_INT input.

Process value branch

The process value can be input in I/O or floating-point format. The function CRP_IN converts the I/O value PV_PER to a floating-point format -100 to +100 % in accordance with the following rule:

Output of CRP_IN = PV_PER * 100 / 27648

The PV_NORM function scales the output of CRP_IN according to the following rule:

Output of PV_NORM = (output of CRP_IN) *PV_FAC + PV_OFF

PV_FAC has a default of 1 and PV_OFF a default of 0.

Forming the error signal

The difference between the setpoint and process value is the error signal. To suppress a minor sustained oscillation due to manipulated variable quantization (e.g. with a pulse width modulation with PULSEGEN), the error signal is applied to a dead band (DEADBAND). With DEADB_W = 0, the dead band is switched off.

PID Algorithm

The PID algorithm operates as a position algorithm. The proportional, integral (INT), and differential (DIF) actions are connected in parallel and can be activated or deactivated individually. This allows P, PI, PD, and PID controllers to be configured. Pure I controllers are also possible.

Manual value processing

It is possible to switch over between manual and automatic mode. In manual mode, the manipulated variable is corrected to a manually selected value.

The integral action (INT) is set internally to LMN - LMN_P - DISV and the derivative action (DIF) is set to 0 and synchronized internally. Changeover to automatic mode is therefore bumpless.

Manipulated value processing

You can use the LMNLIMIT function to limit the manipulated value to selected values. Alarm bits indicate when a limit is exceeded by the input variable.

The LMN_NORM function normalizes the output of LMNLIMIT according to the following rule:

LMN = (output of LMNLIMIT) * LMN_FAC + LMN_OFF

LMN_FAC has a default of 1 and LMN_OFF a default of 0.

The manipulated value is also available in I/O format. The CRP_OUT function converts the LMN floating-point value to a I/O value according to the following rule:

LMN_PER = LMN * 27648 / 100

Feedforward control

A disturbance variable can be added at the DISV input.

See also

Differences compared to CONT_C S7-300/400 (Page 2968)

CONT_C block diagram (S7-300, S7-400)

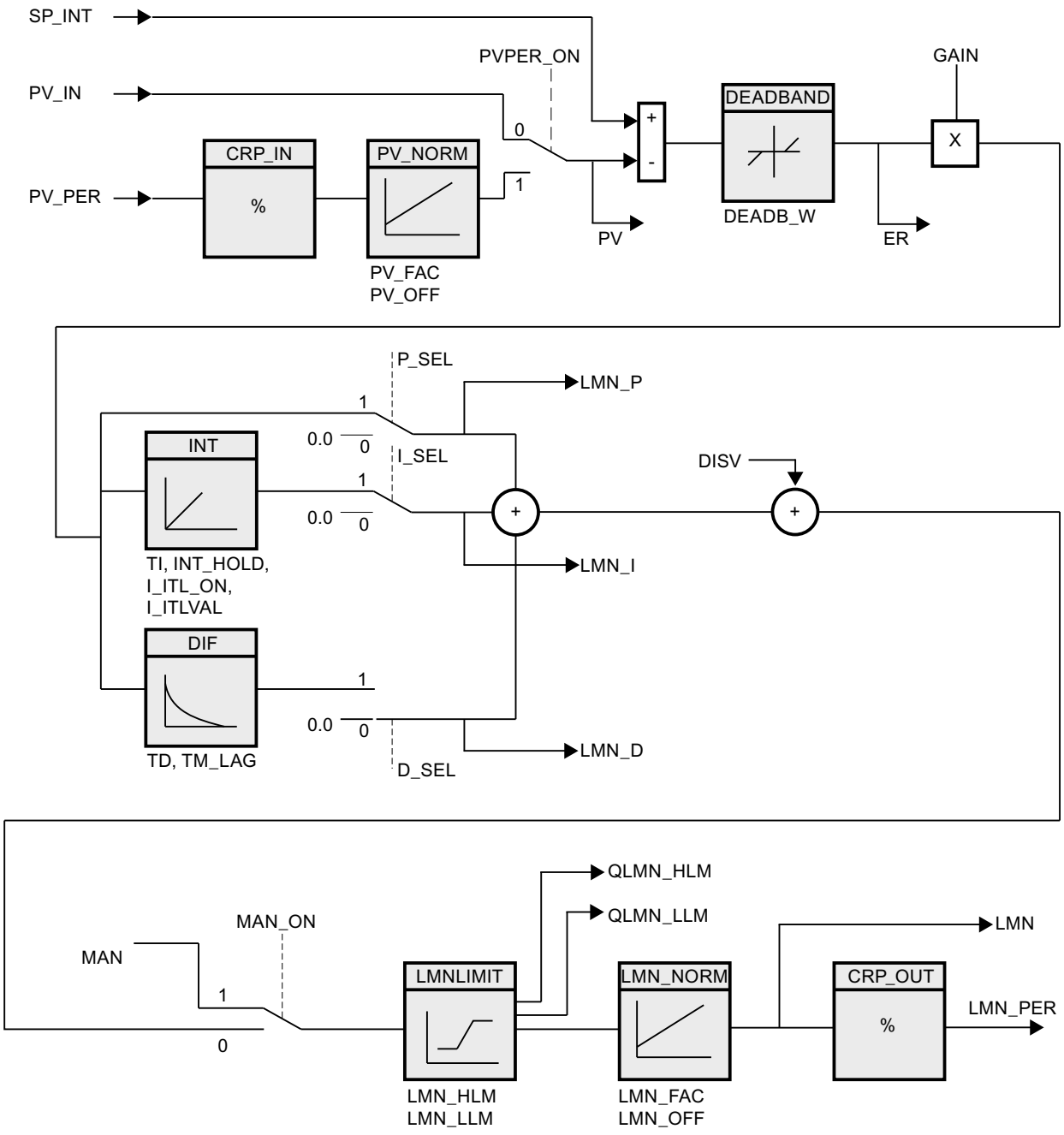

See also

Differences compared to CONT_C S7-300/400 (Page 2968)

Input parameter CONT_C (S7-300, S7-400)

Table 4-198

Parameters	Data type	Default	Description
COM_RST	BOOL	FALSE	The instruction has an initialization routine that is processed when the "Restart" input is set.
MAN_ON	BOOL	TRUE	If the input "Enable manual mode" is set then the control loop is interrupted. A manual value is set as the manipulated value.
PVPER_ON	BOOL	FALSE	If the process value is to be read in from the I/Os, the PV_PER input must be interconnected with the I/Os and the "Enable process value I/Os" input must be set.
P_SEL	BOOL	TRUE	The PID actions can be switched on and off individually in the PID algorithm. P-action is on when the "Enable P-action" input is set.
I_SEL	BOOL	TRUE	The PID actions can be switched on and off individually in the PID algorithm. I action is on when the input "I-action on" is set.
INT_HOLD	BOOL	FALSE	The output of the integral action can be frozen. For this the input "I-action hold" must be set.
I_ITL_ON	BOOL	FALSE	The output of the integral action can be set at the I_ITLVAL input. For this the input "Set I-action" must be set.
D_SEL	BOOL	FALSE	The PID actions can be switched on and off individually in the PID algorithm. D-action is on when the input "Enable D-action" is set.
CYCLE	TIME	T#1s	The time between block calls must be constant. The "Sampling time" input specifies the time between block calls. CYCLE >= 1ms
SP_INT	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. Permissible are values from -100 to 100 % or a physical variable 1).
PV_IN	REAL	0.0	At the "Process value input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. Permissible are values from -100 to 100 % or a physical variable 1).
PV_PER	WORD	W#16#0000	The process value in I/O format is interconnected with the controller at the "Process value I/0" input.
MAN	REAL	0.0	The "Manual value" input is used to set a manual value using the operator interface functions. Permissible are values from -100 to 100 % or a physical variable 2).
GAIN	REAL	2.0	The "Proportional gain" input specifies controller amplification.
TI	TIME	T#20s	The "Integration time" input determines the time response of the integral action. TI >= CYCLE
TD	TIME	T#10s	The "Derivative action time" input determines the time response of the derivative action. TD >= CYCLE

Parameters	Data type	Default	Description
TM_LAG	TIME	T#2s	Time lag of the D-action The algorithm of the D-action contains a delay for which parameters can be assigned at the input "Time lag of the D-action". TM_LAG >= CYCLE/2
DEADB_W	REAL	0.0	A dead band is applied to the system deviation. The "Dead band width" input determines the size of the dead band. DEADB_W >= 0.0 (%) or a physical variable 1)
LMN_HLM	REAL	100.0	The manipulated value is always restricted to a high limit and low limit. The "High limit of manipulated value" input specifies the high limit. Permissible are real values starting at LMN_LLM (%) or a physical variable 2).
LMN_LLM	REAL	0.0	The manipulated value is always restricted to a high limit and low limit. The "Low limit of manipulated value" input specifies the low limit. Permissible are real values up to LMN_HLM (%) or a physical variable 2).
PV_FAC	REAL	1.0	The "Process value factor" input is multiplied by the process value. The input is used to scale the process value range.
PV_OFF	REAL	0.0	The input "Process value offset" is added to the process value. The input is used to scale the process value range.
LMN_FAC	REAL	1.0	The "Manipulated value factor" input is multiplied with the manipulated value. The input is used to scale the manipulated value range.
LMN_OFF	REAL	0.0	The input "Manipulated value offset" is added to the process value. The input is used to scale the manipulated value range.
I_ITLVAL	REAL	0.0	The output of the integral action can be set at the I_ITL_ON input. The initialization value is applied to the input "Initialization value of the I-action." Permissible are values of -100.0 to 100.0 (%) or a physical variable 2).
DISV	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input. Permissible are values of -100.0 to 100.0 (%) or a physical variable 2).

1) Parameters in the setpoint and process value branches with the same unit

2) Parameters in the manipulated value branch with the same unit

See also

Differences compared to CONT_C S7-300/400 (Page 2968)

Output parameters CONT_C (S7-300, S7-400)

Table 4-199

Parameter	Data type	Default	Description
LMN	REAL	0.0	The effective "Manipulated value" is output in floating point format at the "Manipulated value" output.
LMN_PER	WORD	W#16#0000	The manipulated value in I/O format is interconnected on the input "Manipulated value I/O" with the controller.
QLMN_HLM	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output "High limit of manipulated value reached" indicates that the high limit has been reached.
QLMN_LLM	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output "Low limit of manipulated value reached" indicates that the low limit has been reached.
LMN_P	REAL	0.0	The "P-action" output contains the proportional action of the manipulated variable.
LMN_I	REAL	0.0	The "I-action" output contains the integral action of the manipulated variable.
LMN_D	REAL	0.0	The "D-action" output contains the derivative action of the manipulated variable.
PV	REAL	0.0	The effective process value is output at the "Process value" output.
ER	REAL	0.0	The effective system deviation is output at the "Error signal" output.

See also

Differences compared to CONT_C S7-300/400 (Page 2968)

CONT_S (S7-300, S7-400)

Description CONT_S (S7-300, S7-400)

The CONT_S instruction is used on SIMATIC S7 automation systems to control technical processes with binary output value output signals for actuators with integrating behavior. During parameter assignment, you can activate or deactivate sub-functions of the PI step controller to adapt the controller to the controlled system. In addition to the functions in the process value branch, the instruction implements a complete proportional-plus-integral-action controller with binary output value output and the option of manually influencing the value of the output value. The step controller operates without a position feedback signal.

Application

You can use the controller as a PI fixed setpoint controller or in secondary control loops in cascade, blending or ratio controllers, however you cannot use it as the primary controller. The functions of the controller are based on the PI control algorithm of the sampling controller supplemented by the functions for generating the binary output signal from the analog actuating signal.

Call

The CONT_S instruction has an initialization routine that is run through when input parameter COM_RST = TRUE is set. All the signal outputs are set to zero. COM_RST = FALSE has to be set after the initialization routine has been completed.

The calculation of the values in the control blocks is only correct if the block is called at regular intervals. You should therefore call the control blocks in a cyclic interrupt OB (OB 30 to OB 38). Enter the sampling time in the CYCLE parameter.

If you call the instruction CONT_S as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for CONT_S directly in the multiple instance DB and commission it via a watch table.

Error information

The error message word RET_VAL is not evaluated by the block.

See also

Differences compared to CONT_S S7-300/400 (Page 2968)

Mode of operation CONT_S (S7-300, S7-400)

Setpoint branch

The setpoint is entered in floating-point format at the SP_INT input.

Process value branch

The process value can be input in I/O or floating-point format. The function CRP_IN converts the I/O value PV_PER to a floating-point format -100 to +100 % in accordance with the following rule:

Output of CRP_IN = PV_PER * 100 / 27648

The PV_NORM function normalizes the output of CRP_IN according to the following rule:

Output of PV_NORM = (output of CRP_IN) * PV_FAC + PV_OFF

PV_FAC has a default of 1 and PV_OFF a default of 0.

Forming the error signal

The difference between the setpoint and process value is the error signal. To suppress a small constant oscillation due to the manipulated variable quantization (for example, due to a limited resolution of the manipulated value by the control valve), a dead band is applied to the error signal (DEADBAND). With DEADB_W = 0, the dead band is switched off.

PI step algorithm

The instruction operates without position feedback. The I-action of the PI algorithm and the assumed position feedback signal are calculated in **one** integral action (INT) and compared with the remaining P-action as a feedback value. The difference is applied to a three-step element (THREE_ST) and a pulse shaper (PULSEOUT) that generates the pulses for the control valve. The switching frequency of the controller can be reduced by adapting the response threshold of the three-step element.

Feedforward control

A disturbance variable can be added at the DISV input.

See also

Differences compared to CONT_S S7-300/400 (Page 2968)

Block diagram CONT_S (S7-300, S7-400)

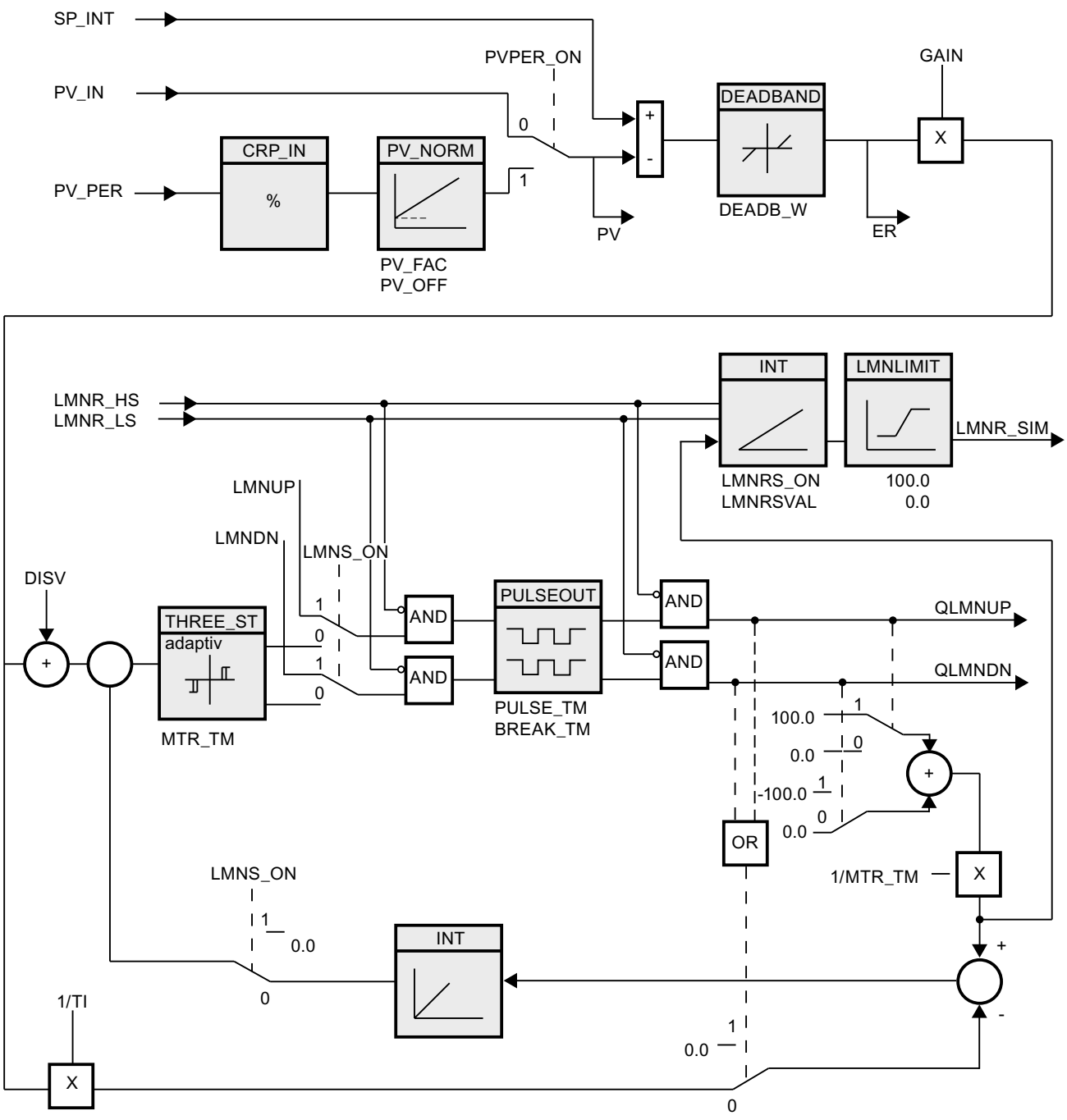

See also

Differences compared to CONT_S S7-300/400 (Page 2968)

Input parameters CONT_S (S7-300, S7-400)

Table 4-200

Parameters	Data type	Default	Description
COM_RST	BOOL	FALSE	The block has an initialization routine that is processed when the "Restart" input is set.
LMNR_HS	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected at the input "High endstop signal of position feedback". LMNR_HS=TRUE means: The control valve is at high endstop.
LMNR_LS	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected on the input "Low endstop signal of position feedback". LMNR_LS=TRUE means The control valve is at low endstop.
LMNS_ON	BOOL	FALSE	Manipulated value signal processing is switched to manual mode at the "Enable manual mode of manipulated signal".
LMNUP	BOOL	FALSE	The output signal QLMNUP is operated in manual mode of the manipulated value signals at the input "Manipulated value signal up".
LMNDN	BOOL	FALSE	The output signal QLMNDN is operated in manual mode of the manipulated value signals at the input "Manipulated value signal down"
PVPER_ON	BOOL	FALSE	If the process value is to be read from the I/O then the input PV_PER must be interconnected with the I/O and the input "Enable process value I/O" must be set.
CYCLE	TIME	T#1s	The time between block calls must be constant. The "Sampling time" input specifies the time between block calls. CYCLE >= 1ms
SP_INT	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. Permissible are values from -100 to 100 % or a physical variable [1].
PV_IN	REAL	0.0	At the "Process value input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. Permissible are values from -100 to 100 % or a physical variable [1].
PV_PER	WORD	W#16#0000	The process value in I/O format is interconnected with the controller at the "Process value I/O" input.
GAIN	REAL	2.0	The "Proportional gain" input specifies controller amplification.
TI	TIME	T#20s	The "Integration time" input determines the time response of the integral action. TI >= CYCLE
DEADB_W	REAL	1.0	A dead band is applied to the system deviation. The "Dead band width" input determines the size of the dead band. Permissible are values from 0 to 100 % or a physical variable [1].
PV_FAC	REAL	1.0	The "Process value factor" input is multiplied by the process value. The input is used to scale the process value range.
PV_OFF	REAL	0.0	The input "Process value offset" is added to the process value. The input is used to scale the process value range.
PULSE_TM	TIME	T#3s	You can assign a minimum pulse time at the parameter "Minimum pulse time". PULSE_TM >= CYCLE
BREAK_TM	TIME	T#3s	You can assign a minimum break time at the parameter "Minimum break time". BREAK_TM >= CYCLE

Parameters	Data type	Default	Description
MTR_TM	TIME	T#30s	The time required by the actuator to move from limit stop to limit stop is entered at the "Motor actuating time" parameter. MTR_TM >= CYCLE
DISV	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input. Permissible are values from -100 to 100 % or a physical variable [2].

[1] Parameters in setpoint and process value branches with identical unit

[2] Parameters in the manipulated value branch with same unit

See also

Differences compared to CONT_S S7-300/400 (Page 2968)

Output parameters CONT_S (S7-300, S7-400)

Table 4-201

Parameters	Data type	Default	Description
QLMNUP	BOOL	FALSE	If the output "Manipulated value signal up" is set then the control valve should be open.
QLMNDN	BOOL	FALSE	If the output "Manipulated value signal down" is set then the control valve should be closed.
PV	REAL	0.0	The effective process value is output at the "Process value" output.
ER	REAL	0.0	The effective system deviation is output at the "Error signal" output.

See also

Differences compared to CONT_S S7-300/400 (Page 2968)

PULSEGEN (S7-300, S7-400)

Description PULSEGEN (S7-300, S7-400)

The instruction PULSEGEN serves as the structure of a PID controller with impulse output for proportional actuators. PULSEGEN transforms the input value INV (= LMN of the PID controller) through modulation of the impulse width in an impulse sequence with a constant period duration, which corresponds with the cycle time with which the input value is updated.

Application

You can use the PULSEGEN instruction to configure two- or three-step PID controllers with pulse width modulation. The function is normally used in conjunction with the continuous controller CONT_C.

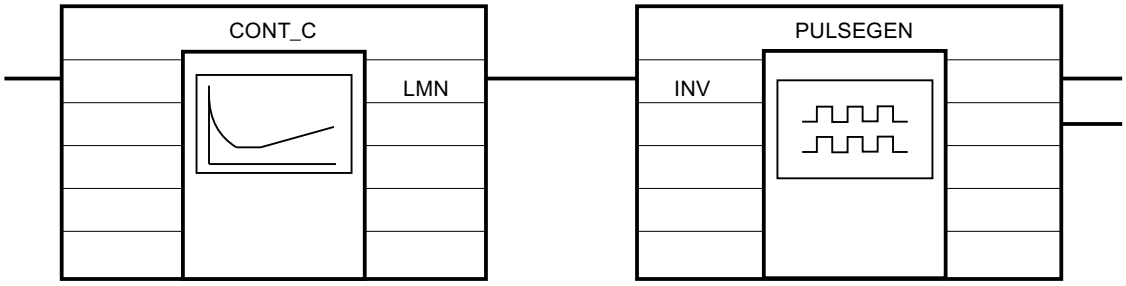

Call

The PULSEGEN instruction has an initialization routine that is run through when input parameter COM_RST = TRUE is set. All the signal outputs are set to zero. COM_RST = FALSE has to be set after the initialization routine has been completed.

The calculation of the values in the control blocks is only correct if the block is called at regular intervals. You should therefore call the control blocks in a cyclic interrupt OB (OB 30 to OB 38). Enter the sampling time in the CYCLE parameter.

Responses in the event of an error

The error message word RET_VAL is not evaluated by the block.

See also

Differences compared to PULSEGEN S7-300/400 (Page 2968)

Mode of operation PULSEGEN (S7-300, S7-400)

Impulse width modulation

The duration of a pulse per period duration is proportional to the input variable. The cycle assigned via PER_TM is not identical to the processing cycle of the PULSEGEN instruction. Rather, a PER_TM cycle is made up of several processing cycles of the PULSEGEN instruction, whereby the number of PULSEGEN calls per PER_TM cycle determines the accuracy of the pulse width.

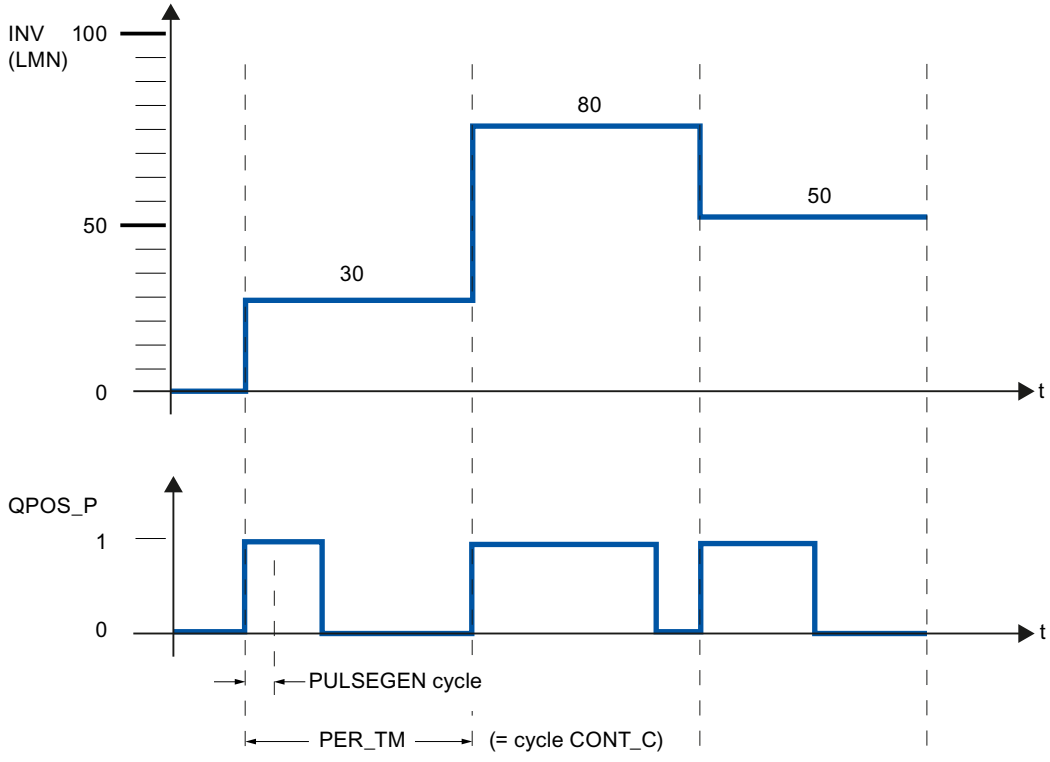

An input variable of 30% and 10 PULSEGEN calls per PER_TM mean the following:

- "One" at the QPOS_P output for the first three calls of PULSEGEN (30% of 10 calls)
- "Zero" at the QPOS_P output for seven further calls of PULSEGEN (70% of 10 calls)

Block diagram

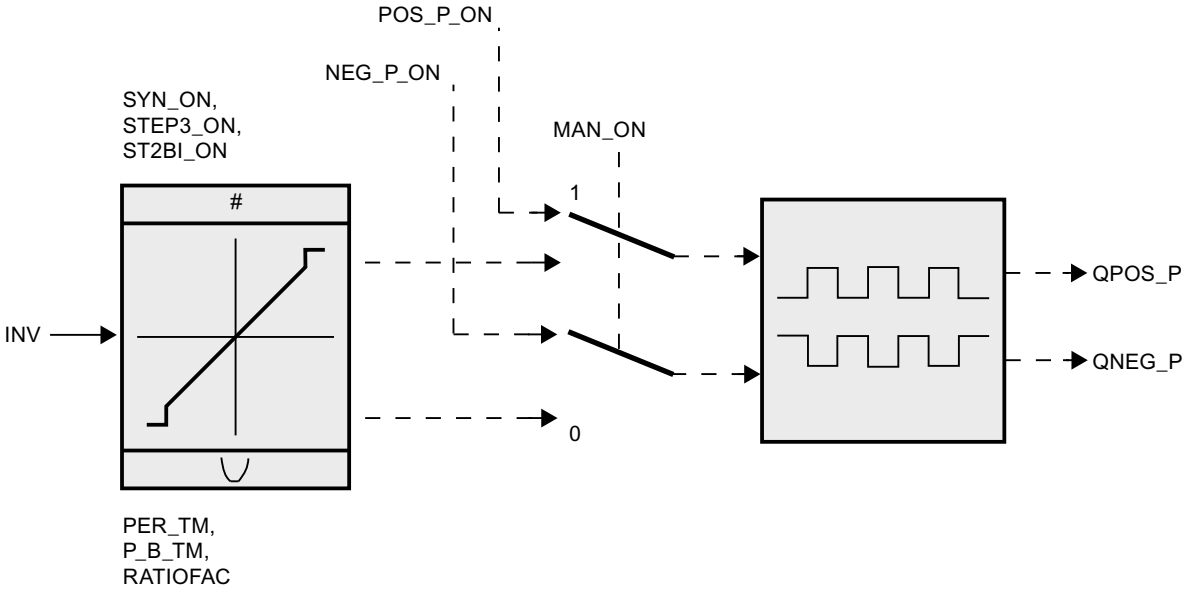

Accuracy of the manipulated value

With a "Sampling ratio" of 1:10 (CONT_C calls to PULSEGEN calls) the accuracy of the manipulated value in this example is restricted to 10%, in other words, set input values INV can only be simulated by a pulse duration at the QPOS_P output in steps of 10 %.

The accuracy is increased as the number of PULSEGEN calls per CONT_C call is increased.

If PULSEGEN is called, for example, 100 times more often than CONT_C, a resolution of 1 % of the manipulated value range is achieved.

Note

The reduction ratio of the call frequency must be programmed by the user.

Automatic synchronization

It is possible to automatically synchronize the pulse output with the instruction that updates the input variable INV (e.g. CONT_C). This ensures that a change in the input variable is output as quickly as possible as a pulse.

The pulse shaper evaluates the input value INV at intervals corresponding to the period duration PER_TM and converts the value into a pulse signal of corresponding length.

Since, however, INV is usually calculated in a slower cyclic interrupt class, the pulse shaper should start the conversion of the discrete value into a pulse signal as soon as possible after the updating of INV.

To allow this, the block can synchronize the start of the period using the following procedure:

If INV changes and if the block call is not in the first or last two call cycles of a period, a synchronization is performed. The pulse duration is recalculated and in the next cycle is output with a new period.

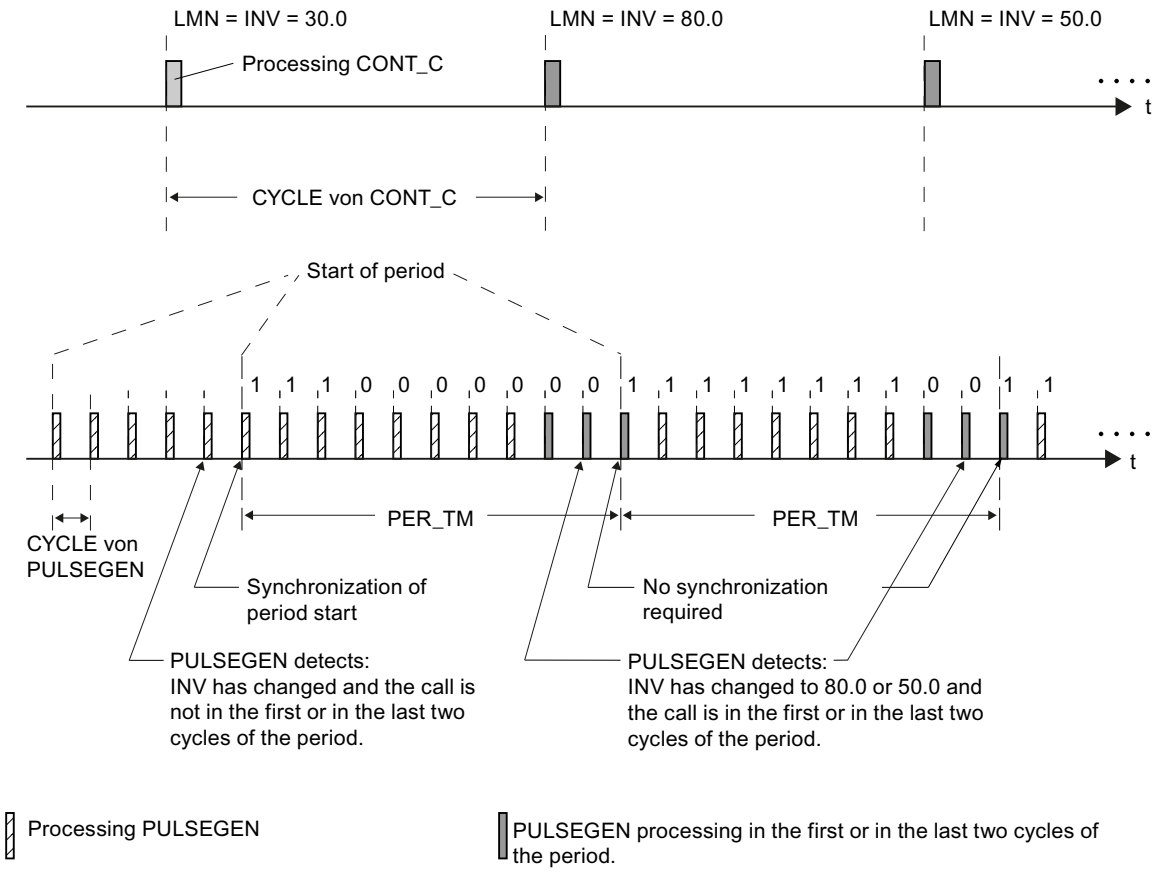

The automatic synchronization is switched off, if SYN_ON = FALSE.

Note

The start of a new period and subsequent synchronization usually leads to a certain imprecision when the old value of INV (i.e. of LMN) is mapped to the pulse signal.

See also

Differences compared to PULSEGEN S7-300/400 (Page 2968)

Mode of operation PULSEGEN (S7-300, S7-400)

Modes

Depending on the parameters assigned to the pulse shaper, PID controllers with a three-step output or with a bipolar or unipolar two-step output can be configured. The following table illustrates the setting of the switch combinations for the possible modes.

Mode	MAN_ON	STEP3_ON	ST2BI_ON
Three-step control	FALSE	TRUE	Any
Two-step control with bi-polar Manipulating range (-100 % to 100 %)	FALSE	FALSE	TRUE
Two-step control with unipolar Manipulating range (0 % to 100 %)	FALSE	FALSE	FALSE
Manual mode	TRUE	Any	Any

Manual mode in two/three-step control

In the manual mode (MAN_ON = TRUE), the binary outputs of the three-step or two-step controller can be set using the signals POS_P_ON and NEG_P_ON regardless of INV.

Control	POS_P_ON	NEG_P_ON	QPOS_P	QNEG_P
Three-step control	FALSE	FALSE	FALSE	FALSE
	TRUE	FALSE	TRUE	FALSE
	FALSE	TRUE	FALSE	TRUE
	TRUE	TRUE	FALSE	FALSE
Two-step control	FALSE	Any	FALSE	TRUE
	TRUE	Any	TRUE	FALSE

See also

Differences compared to PULSEGEN S7-300/400 (Page 2968)

Three-step control (S7-300, S7-400)

Three-step control

In "Three-step control" mode, it is possible to generate three statuses of the actuating signal. For this, the status values of the binary output signals QPOS_P and QNEG_P are assigned to the respective operating statuses of the actuator. The table shows the example of a temperature control:

Output signals	Heat	Off	Cool
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

The pulse duration is calculated from the input variable via a characteristic curve. The form of the characteristic curve is defined by the minimum pulse duration or minimum interval and the ratio factor. The normal value for the ratio factor is 1.

The "doglegs" in the curves are caused by the minimum pulse duration or minimum interval.

Minimum pulse duration or minimum interval

A correctly assigned minimum pulse duration or minimum interval P_B_TM can prevent short on/off times, which reduce the working life of switching elements and actuators. Small absolute values of input variable LMN that would otherwise generate a pulse duration shorter than P_B_TM are suppressed. Large input values that would generate a pulse duration longer than PER_TM - P_B_TM are set to 100% or -100%.

The duration of positive or negative pulses is calculated by multiplying the input variable (in %) by the period duration:

Pulse duration = INV / 100 * PER_TM

The following figure shows a symmetrical characteristic curve of the three-step controller (ratio factor = 1).

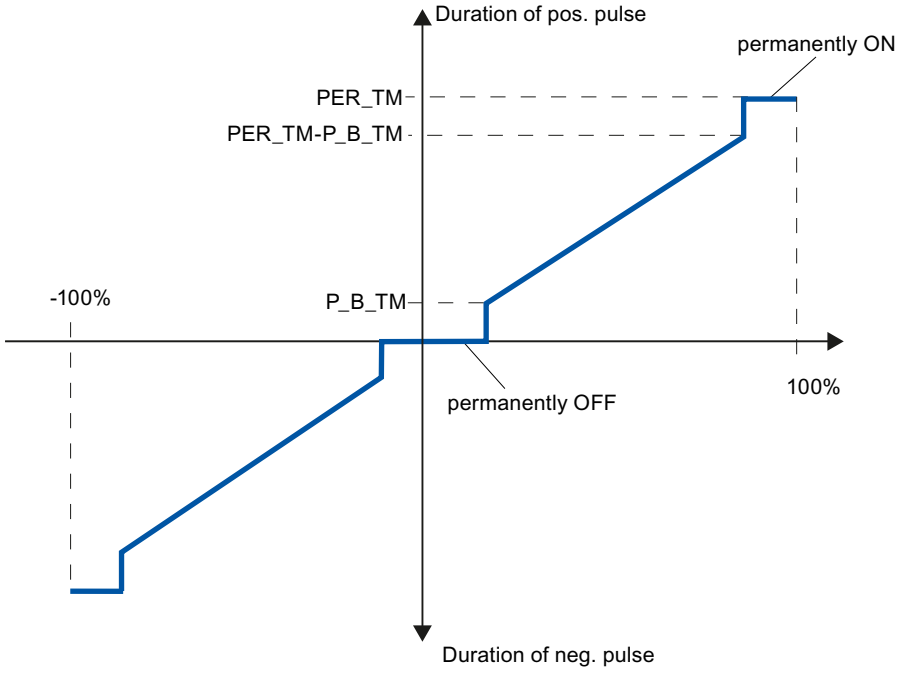

Asymmetrical three-step control

Using the ratio factor RATIOFAC, the ratio of the duration of positive to negative pulses can be changed. In a thermal process, for example, this would allow different system time constants for heating and cooling.

Ratio factor < 1

The pulse duration at the negative pulse output, calculated by multiplying the input variable by the period duration, is multiplied by the ratio factor.

Positive pulse duration = INV /100 * PER_TM

Negative pulse duration = INV / 100 * PER_TM * RATIOFAC

The following figure shows the asymmetrical characteristic curve of the three-step controller (ratio factor = 0.5):

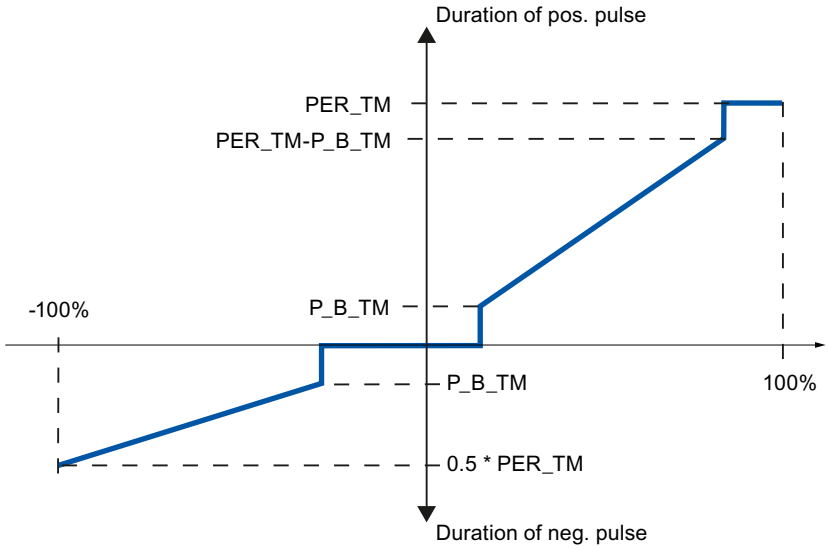

Ratio factor > 1

The pulse duration at the positive pulse output, calculated by multiplying the input variable by the period duration, is divided by the ratio factor.

Positive pulse duration = INV / 100 * PER_TM / RATIOFAC

Negative pulse duration = INV / 100 * PER_TM

See also

Differences compared to PULSEGEN S7-300/400 (Page 2968)

Two-step control (S7-300, S7-400)

In two-step control, only the positive pulse output QPOS_P of PULSEGEN is connected to the on/off actuator. Depending on the manipulated value range used, the two-step controller has a bipolar or a unipolar manipulated value range.

Two-step control with bipolar manipulated variable range (-100% to 100%)

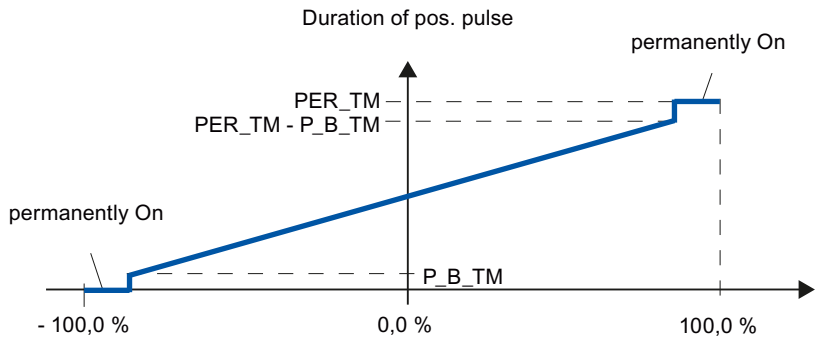

Two-step control with unipolar manipulated variable range (0% to 100%)

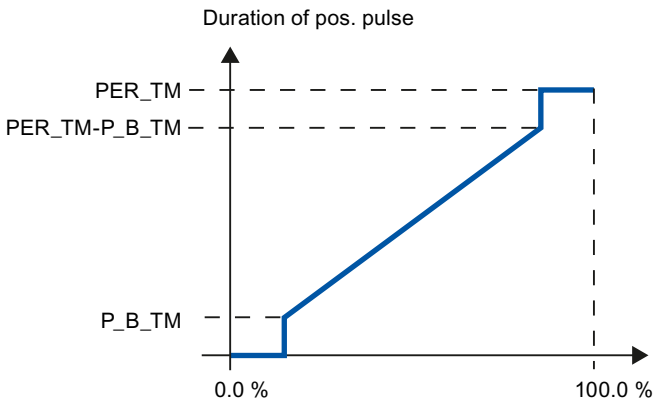

The negated output signal is available at QNEG_P if the connection of the two-step controller in the control loop requires a logically inverted binary signal for the actuating pulses.

Pulse	Actuator On	Actuator Off
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

See also

Differences compared to PULSEGEN S7-300/400 (Page 2968)

Input parameters PULSEGEN (S7-300, S7-400)

The values of the input parameters are not limited in the block. There is no parameter check.

Table 4-202

Parameters	Data type	Default	Description
INV	REAL	0.0	At the input parameter "Input variable" an analog manipulated variable is connected. Values from -100 to 100 % are permitted.
PER_TM	TIME	T#1s	At the parameter "Period duration" the constant period duration of the pulse width modulation is entered. This corresponds to the sampling time of the controller. The ratio between the sampling time of the pulse shaper and the sampling time of the controller determines the accuracy of the pulse width modulation. PER_TM >=20*CYCLE
P_B_TM	TIME	T#50 ms	You can assign a minimum pulse/break time at the parameter "Minimum pulse/break time". P_B_TM >= CYCLE
RATIOFAC	REAL	1.0	Using the "Ratio factor" input parameter the ratio of the duration of positive to negative pulses can be changed. In a thermal process, this would, for example, allow different time constants for heating and cooling to be compensated (for example, in a process with electrical heating and water cooling). Values from 0.1 to 10.0 are permitted.
STEP3_ON	BOOL	TRUE	At the input parameter "Enable three-step control" the appropriate mode is activated. In three-step control both output signals are active.
ST2BI_ON	BOOL	FALSE	At the input parameter "Enable two-step control for bipolar manipulated value range" you can select from the modes "Two-step control for bipolar manipulated value range" and "Two-step control for unipolar manipulated value range". STEP3_ON = FALSE is required.
MAN_ON	BOOL	FALSE	Setting the input parameter "Enable manual mode" allows the output signals to be set manually.
POS_P_ON	BOOL	FALSE	For manual mode three-step control, the output signal QPOS_P can be operated on the input parameter "Positive pulse on". In manual mode with two-step control, QNEG_P is always set inversely to QPOS_P.
NEG_P_ON	BOOL	FALSE	For manual mode three-step control, the output signal QNEG_P can be operated on the input parameter "Negative pulse on". In manual mode with two-step control, QNEG_P is always set inversely to QPOS_P.
SYN_ON	BOOL	TRUE	By setting the input parameter "Enable synchronization", it is possible to synchronize the pulse output automatically with the block that updates the input variable INV. This ensures that a change in the input variable is output as quickly as possible as a pulse.

Parameters	Data type	Default	Description
COM_RST	BOOL	FALSE	The block has an initialization routine that is processed when the input "Restart" is set.
CYCLE	TIME	T#10ms	The time between block calls must be constant. The "Sampling time" input specifies the time between block calls. CYCLE >= 1ms

See also

Differences compared to PULSEGEN S7-300/400 (Page 2968)

Output parameter PULSEGEN (S7-300, S7-400)

Table 4-203

Parameters	Data type	Default	Description
QPOS_P	BOOL	FALSE	The output parameter "Output signal positive pulse" is set if a pulse will be output. In three-step control, this is always the positive pulse. In two-step control, the QNEG_P is always set inversely to QPOS_P.
QNEG_P	BOOL	FALSE	The output parameter "Output signal negative pulse" is set if a pulse will be output. In three-step control, this is always the negative pulse. In two-step control, QNEG_P is always set inversely to QPOS_P.

See also

Differences compared to PULSEGEN S7-300/400 (Page 2968)

TCONT_CP (S7-300, S7-400)

Description TCONT_CP (S7-300, S7-400)

The instruction TCONT_CP is used to control temperature processes with continuous or pulsed control signals. The controller functionality is based on the PID control algorithm with additional functions for temperature processes. To improve the control response with temperature processes, the block includes a control zone and reduction of the proportional component if there is a setpoint step change.

The instruction can set the PI/PID parameters itself using the controller optimization function.

Application

The controller controls one actuator; in other words, with one controller you can either heat or cool but not both. If you use the block for cooling, GAIN must be assigned a negative value. This inversion of the controller means that if the temperature rises, for example, the manipulated variable LMN and with it the cooling action is increased.

Call

The instruction TCONT_CP must be called equidistant. To achieve this, use a cyclic interrupt priority class (for example, OB35 for an S7-300).

The TCONT_CP instruction has an initialization routine that is run through when input parameter COM_RST = TRUE is set. During initialization, the integral action is set to the initialization value I_ITVAL. All the signal outputs are set to zero. Following execution of the initialization routine, the block sets COM_RST back to FALSE. If you require initialization when the CPU restarts, call the block in OB100 with COM_RST = TRUE.

If you call the instruction TCONT_CP as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for TCONT_CP directly in the multiple instance DB and commission it via a watch table.

See also

Operating principle of the pulse generator (Page 5591)

Block diagram TCONT_CP (Page 5594)

Differences compared to TCONT_CP S7-300/400 (Page 2969)

Mode of operation TCONT_CP (S7-300, S7-400)

Setpoint branch

The setpoint is entered at input SP_INT in floating-point format as a physical value or percentage. The setpoint and process value used to form the control deviation must have the same unit.

Process value options (PVPER_ON)

Depending on PVPER_ON, the process value can be read in, in the I/O or floating-point format.

PVPER_ON	Process Value Input
TRUE	The process value is read in via the analog I/Os (PIW xxx) at input PV_PER.
FALSE	The process value is acquired in floating-point format at input PV_IN.

Process value format conversion CRP_IN (PER_MODE)

The CRP_IN function converts the I/O value PV_PER to floating-point format depending on the PER_MODE switch according to the following rules:

PER_MODE	Output of CRP_IN	Analog Input Type	Unit
0	PV_PER * 0.1	Thermoelements; PT100/NI100; standard	°C;°F
1	PV_PER * 0.01	PT100/NI100; climate;	°C;°F
2	PV_PER * 100/27648	Voltage/current	%

Process value scaling PV_NORM (PF_FAC, PV_OFFS)

The PV_NORM function calculates the output of CRP_IN according to the following rule:

"Output of PV_NORM" = "Output of CRP_IN)" * PV_FAC + PV_OFFS

It can be used for the following purposes:

- Process value adjustment with PV_FAC as process value factor and PV_OFFS as process value offset.
- Scaling of temperature to percentage
  You want to enter the setpoint as a percentage and must now convert the measured temperature value to a percentage.
- Scaling of percentage to temperature
  You want to enter the setpoint in the physical temperature unit and must now convert the measured voltage/current value to a temperature.

Calculation of the parameters:

- PV_FAC = range of PV_NORM/range of CRP_IN;
- PV_OFFS = LL (PV_NORM) - PV_FAC * LL(CRP_IN);
  where LL: Low limit

The scaling is switched off with the default values (PV_FAC = 1.0 and PV_OFFS = 0.0). The effective process value is output at the PV output.

Note

With pulse control, the process value must be transferred to the block in the fast pulse call (reason: mean value filtering). Otherwise, the control quality can deteriorate.

Example of Process Value Scaling

If you want to enter the setpoint as a percentage, and you have a temperature range of -20 to 85 °C applied to , CRP_IN you must normalize the temperature range as a percentage.

The diagram below shows an example of adapting the temperature range -20 to 85 °C to an internal scale of 0 to 100 %:

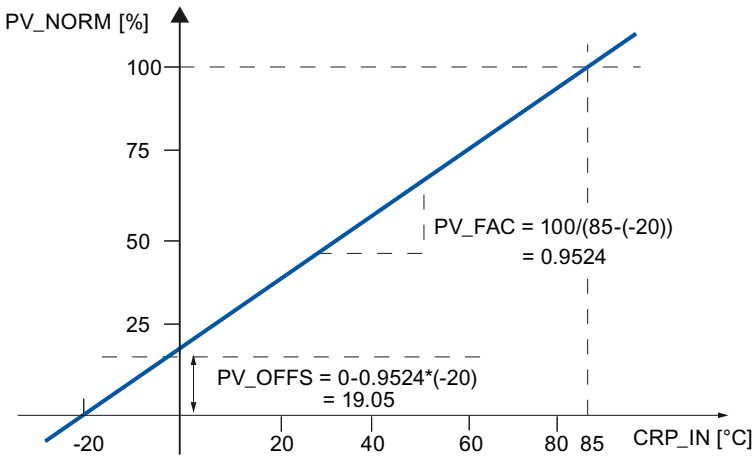

Forming the control deviation

The difference between the setpoint and process value is the control deviation before the dead band.

The setpoint and process value must exist in the same unit.

Dead band (DEADB_W)

To suppress a minor sustained oscillation due to the manipulated variable quantization (for example, in pulse width modulation with PULSEGEN) a dead band is applied to the (DEADBAND) control deviation. With DEADB_W = 0.0, the dead band is disabled. The effective control deviation is indicated by the ER parameter.

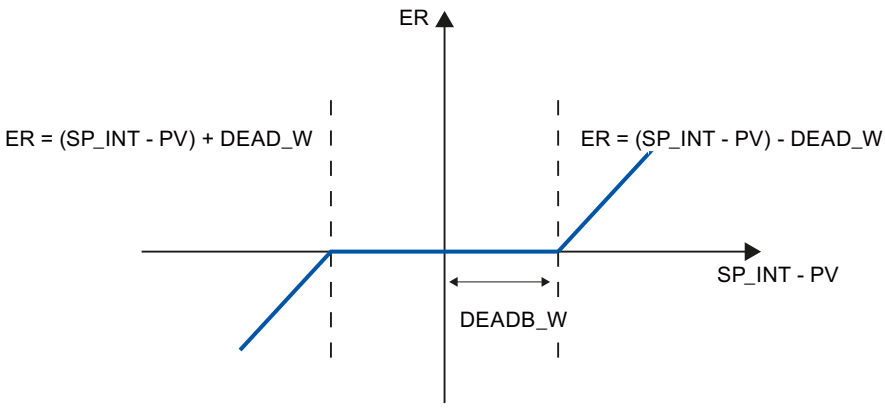

PID Algorithm

The following figure shows the block diagram of the PID algorithm.

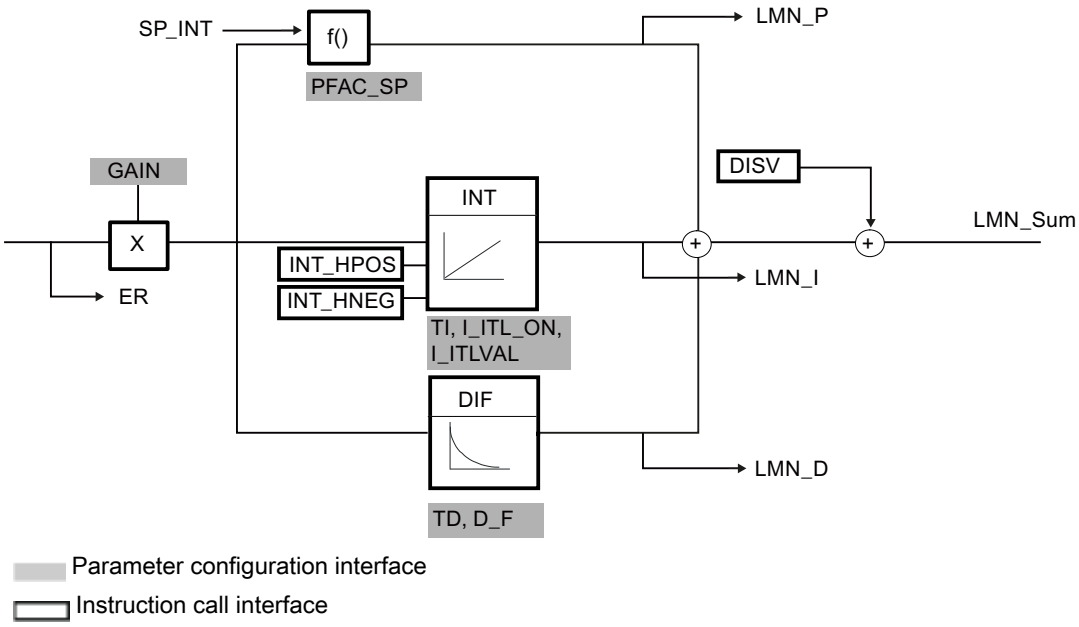

PID Algorithm (GAIN, TI, TD, D_F)

The PID algorithm operates as a position algorithm. The proportional, integral (INT), and derivative (DIF) actions are connected in parallel and can be activated or deactivated individually. This allows P, PI, PD, and PID controllers to be configured.

Controller tuning supports PI and PID controllers. Controller inversion is implemented using a negative GAIN (cooling controller).

If you set TI and TD to 0.0, you obtain a pure P controller at the operating point.

The step response in the time range is:

$$LMN_Sum(t) = GAIN \cdot ER(0) \cdot \left(1 + \frac{1}{TI} \cdot t + D_F \cdot e^{\frac{-t}{TD\ /\ D_F}} \right)$$

Where:

LMN_Sum(t) the manipulated variable in the controller's automatic mode

ER (0) is the step height of the normalized control deviation

GAIN is the controller gain

TI is the integration time

TD is the derivative action time

D_F is the derivative factor

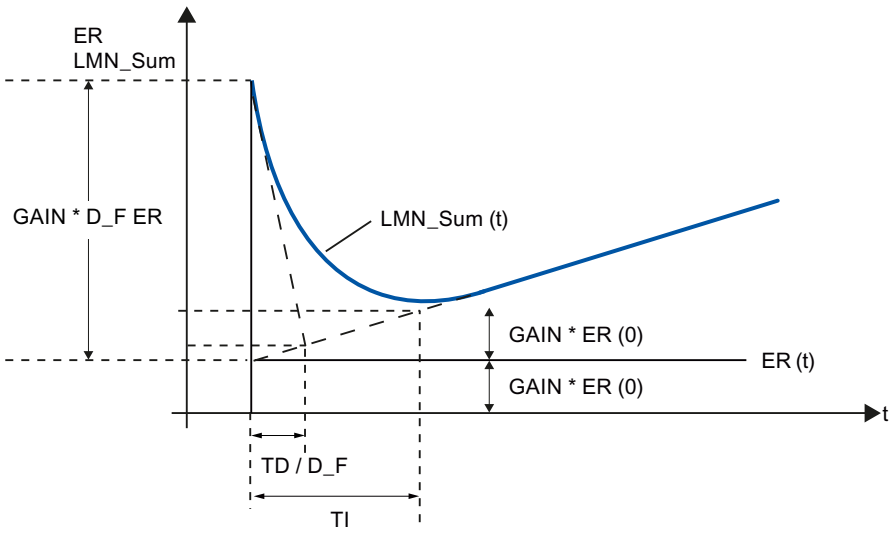

Integral action (TI, I_ITL_ON, I_ITLVAL)

In manual mode, it is corrected as follows: LMN_I = LMN - LMN_P - DISV.

If the output value is limited, the integral action is halted. If the control deviation moves the integral action back in the direction of the output range, the integral action is enabled again.

The integral action is also modified by the following measures:

- The integral action of the controller is deactivated by TI = 0.0
- Weakening of the proportional action when setpoint changes occur
- Control zone
- The output value limits can be modified online

Weakening of the proportional action when setpoint changes occur (PFAC_SP)

To prevent overshoot, you can weaken the proportional action using the parameter "Proportional factor for setpoint changes" (PFAC_SP). Using PFAC_SP, you can select continuously between 0.0 and 1.0 to decide the effect of the proportional action when the setpoint changes:

- PFAC_SP = 1.0: Proportional action has full effect if the setpoint changes
- PFAC_SP = 0.0: Proportional action has no effect if the setpoint changes

The weakening of the proportional action is achieved by compensating the integral action.

Derivative action (TD, D_F)

- The derivative action of the controller is deactivated by TD = 0.0
- If the derivative action is active, the following relationship should apply:
  TD = 0.5 * CYCLE * D_F

Parameter Settings of a P or PD Controller with Operating Point

In the user interface, deactivate the integral action (TI = 0.0) and possibly also the derivative action (TD = 0.0). Then make the following parameter settings:

- I_ITL_ON = TRUE
- I_ITLVAL = operating point;

Feedforward control (DISV)

A disturbance variable can be added at the DISV input.

Calculating the output value

The diagram below is the block diagram of the output value calculation:

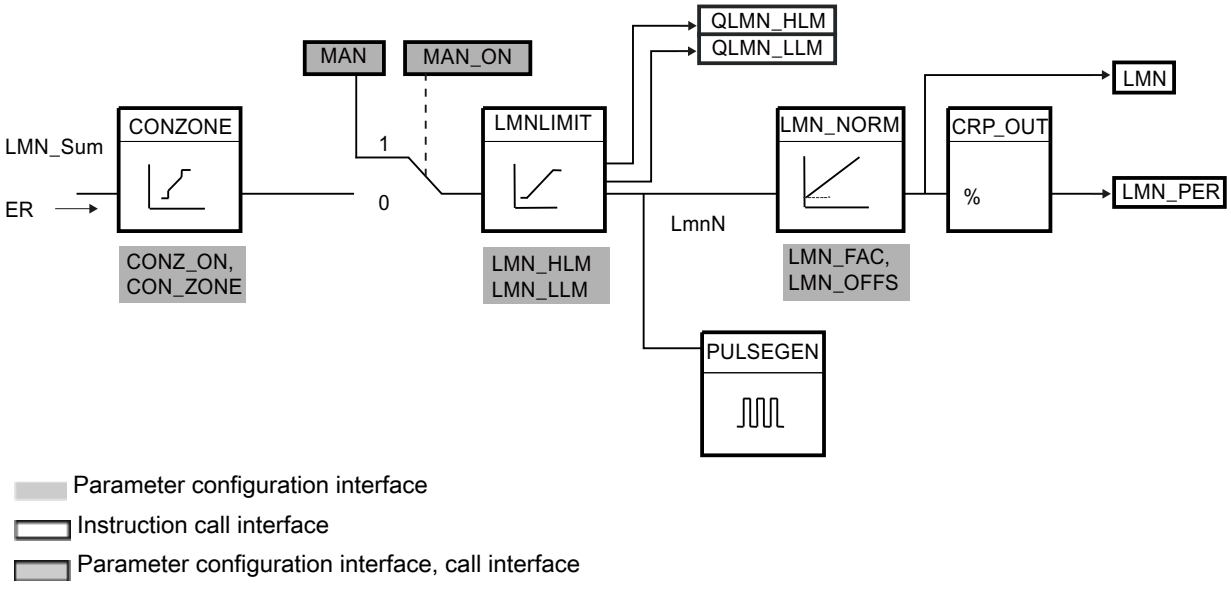

Control zone (CONZ_ON, CON_ZONE)

If CONZ_ON = TRUE, the controller operates with a control zone. This means that the controller operates according to the following algorithm:

- If process value PV exceeds the setpoint SP_INT by more than CON_ZONE, the value LMN_LLM is output as the manipulated variable.
- If the process value PV falls below setpoint SP_INT by more than CON_ZONE, LMN_HLM is output.
- If the process value PV is within the control zone (CON_ZONE), the output value takes its value from the PID algorithm LMN_Sum.

Note

Changing the manipulated variable from LMN_LLM or LMN_HLM to LMN_Sum occurs under compliance of a hysteresis of 20% of the control zone.

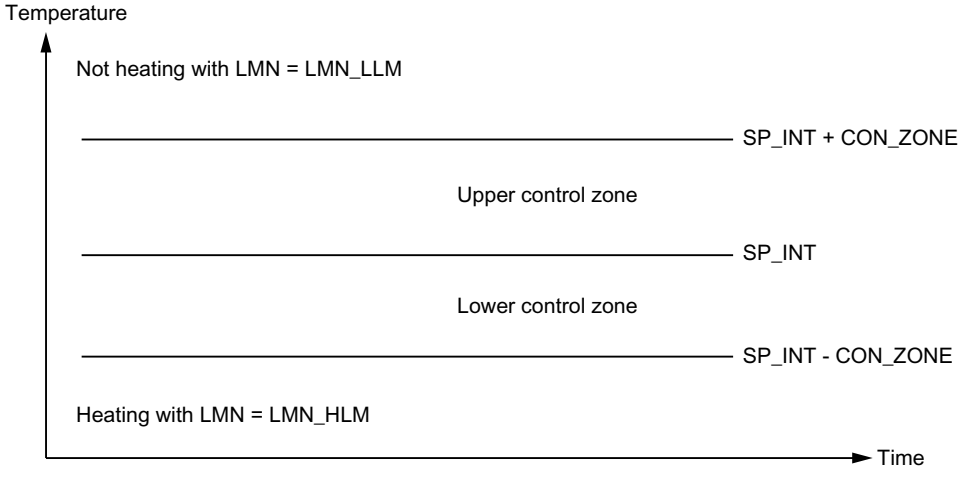

Note

Before enabling the control zone manually, make sure that the control zone band is not too narrow. If the control zone band is too small, oscillations will occur in the manipulated variable and process value.

Advantage of the Control Zone

When the process value enters the control zone, the D-action causes an extremely fast reduction of the manipulated variable. This means that the control zone is only useful when the D-action is activated. Without a control zone, only the reducing P-action would essentially reduce the manipulated variable. The control zone leads to faster settling without overshoot or undershoot if the output minimum or maximum manipulated variable is a long way from the manipulated variable required for the new operating point.

Manual value processing (MAN_ON, MAN)

You can change over between manual and automatic mode. In manual mode, the manipulated variable is corrected to a manually selected value.

The integral action (INT) is set internally to LMN - LMN_P - DISV and the derivative action (DIF) is set to 0 and synchronized internally. Changeover to automatic mode is therefore bumpless.

Note

The MAN_ON parameter has no effect during tuning.

Output value limit LMNLIMIT (LMN_HLM, LMN_LLM)

The LMNLIMIT function is used to limit the output value to the limits LMN_HLM and LMN_LLM. If these limits are reached, this is indicated by the message bits QLMN_HLM and QLMN_LLM.

If the output value is limited, the integral action is halted. If the control deviation moves the integral action back in the direction of the output range, the integral action is enabled again.

Changing the Manipulated Value Limits Online

If the range of the output value is reduced and the new unlimited value of the output value is outside the limits, the integral action and therefore the output value shifts.

The output value is reduced by the same amount as the output value limit changed. If the output value was unlimited prior to the change, it is set exactly to the new limit (described here for the high output value limit).

Scaling of output value LMN_NORM (LMN_FAC, LMN_OFFS)

The LMN_NORM function normalizes the output value according to the following rule:

LMN = LmnN * LMN_FAC + LMN_OFFS

It can be used for the following purposes:

- Output value scaling with LMN_FAC as output value factor and LMN_OFFS as output value offset.

The output value is also available in I/O format. The CRP_OUT function converts the LMN floating-point value to an I/O value according to the following rule:

LMN_PER = LMN * 27648/100

The scaling is switched off with the default values (LMN_FAC = 1.0 and LMN_OFFS = 0.0). The effective output value is sent to output LMN.

Save controller parameters SAVE_PAR

If you classify the current controller parameters as utilizable, you can save these before a manual change in structure parameters provided specifically for this in the instance DB of the instruction TCONT_CP. If you optimize the controller, the saved parameters are overwritten by the values that were valid prior to tuning.

PFAC_SP, GAIN, TI, TD, D_F, CONZ_ON and CONZONE are written to the structure PAR_SAVE.

Reloading Saved Controller Parameters UNDO_PAR

The last controller parameter settings you saved can be activated for the controller again using this function (in manual mode only).

Change between PI and PID parameters LOAD_PID (PID_ON)

Following tuning, the PI and PID parameters are stored in the PI_CON and PID_CON structures. Depending on PID_ON, you can use LOAD_PID in manual mode to write the PI or PID parameters to the effective controller parameters.

PID parameters PID_ON = TRUE	PI parameters PID_ON = FALSE
• GAIN = PID_CON.GAIN • TI = PID_CON.TI • TD = PID_CON.TD	• GAIN = PI_CON.GAIN • TI = PI_CON.TI

Note

The controller parameters are only written back to the controller with UNDO_PAR or LOAD_PID, if the controller gain is not equal to 0:

With LOAD_PID, the parameters are only copied if the corresponding GAIN <> 0 is (either the PI or PID parameters). This strategy takes into account the situation that no tuning has yet been made or that PID parameters are missing. If PID_ON = TRUE and PID.GAIN = FALSE, PID_ON is set to FALSE and the PI parameter is copied.

- D_F, PFAC_SP are preset by the the tuning. These can then be modified by the user. LOAD_PID does not change these parameters.
- With LOAD_PID, the control zone is always recalculated (CON_ZONE = 250/GAIN), even if CONZ_ON = FALSE.

See also

Operating principle of the pulse generator (Page 5591)

Block diagram TCONT_CP (Page 5594)

Differences compared to TCONT_CP S7-300/400 (Page 2969)

Operating principle of the pulse generator (S7-300, S7-400)

The function PULSEGEN transforms the analog manipulated value LmnN through pulse width module into an impulse sequence with the period duration PER_TM. PULSEGEN is switched on with PULSE_ON = TRUE and is processed in the cycle CYCLE_P.

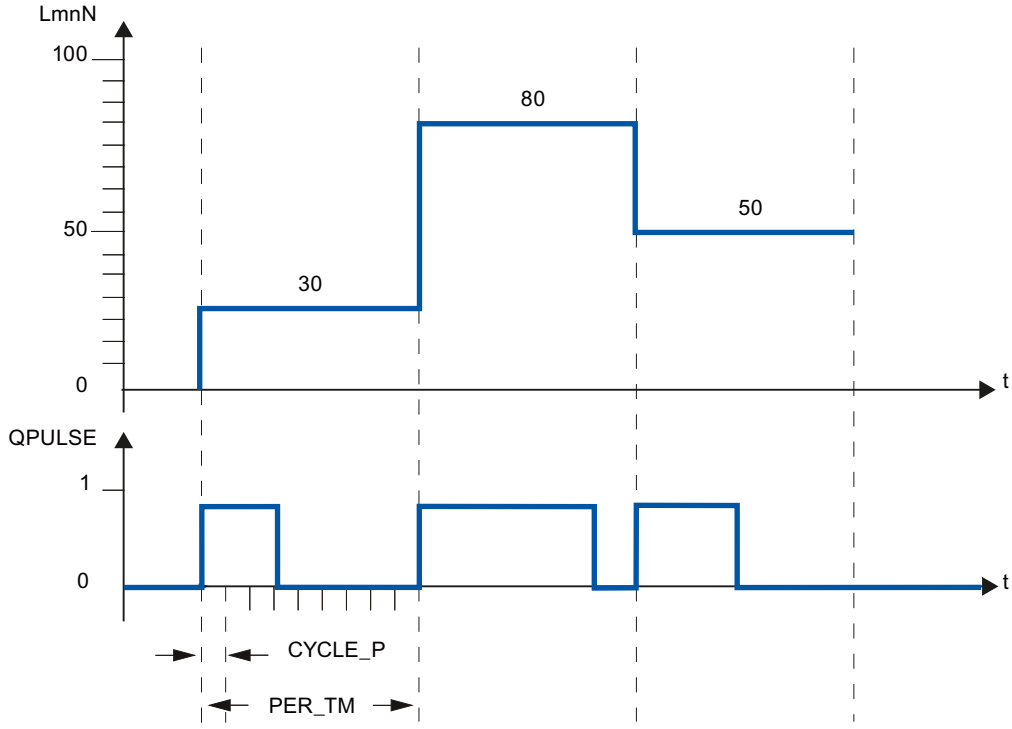

A manipulated value of LmnN = 30% and 10 PULSEGEN calls per PER_TM therefore means:

- TRUE at output QPULSE for the first three PULSEGEN calls (30% of 10 calls)
- FALSE at output QPULSE for seven further PULSEGEN calls (70% of 10 calls)

The duration of a pulse per pulse repetition period is proportional to the manipulated variable and is calculated as follows:

Pulse duration = PER_TM * LmnN /100

By suppressing the minimum pulse or break time, the characteristic curve of the conversion develops "knees" in the start and end regions.

The following diagram illustrates two-step control with a unipolar manipulated variable range (0% to 100%):

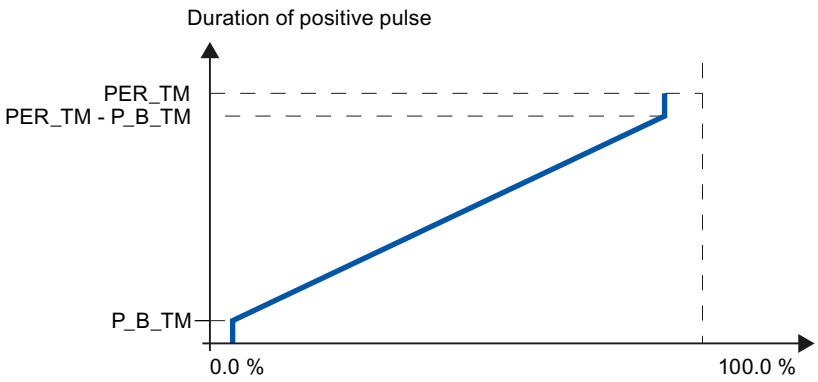

Minimum pulse or minimum break time (P_B_TM)

Short on or off times hinder the lifespan of actuators and fine controlling units. These can be avoided by setting a minimum pulse duration or minimum break time P_B_TM.

Small absolute values at the input variable LmnN that could otherwise generate a pulse duration shorter than P_B_TM are suppressed.

Large input values that would generate a pulse duration greater than PER_TM - P_B_TM are set to 100%. This reduces the dynamics of pulse generation.

Set values of P_B_TM ≤ 0,1 * PER_TM are recommended for the minimum pulse duration and the minimum break duration.

The "knees" in the curves in the diagram above are caused by the minimum pulse or minimum break times.

The following schematic illustrates the switching response of the pulse output:

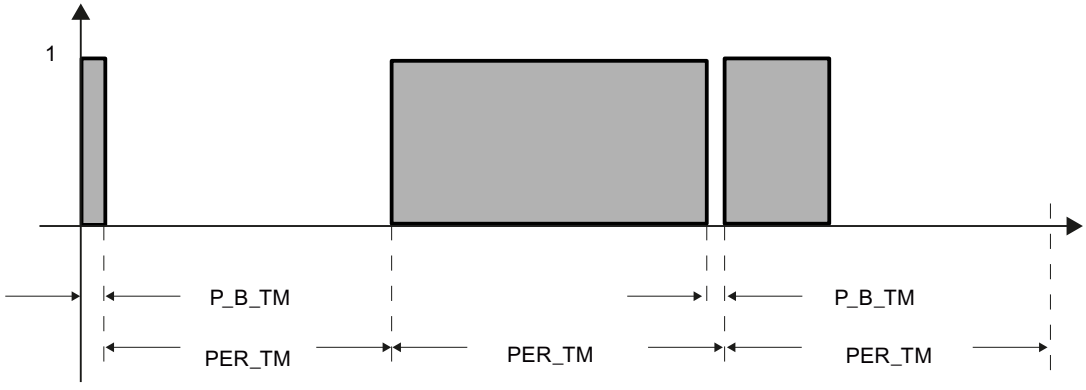

Accuracy of pulse generation

The smaller the pulse generator CYCLE_P is compared to the period duration PER_TM, the more precise the pulse width modulation is. To achieve sufficiently accurate control, the following relationship should apply:

CYCLE_P ≤ PER_TM/50

The manipulated value is transformed with a resolution of ≤ 2 % into an impulse.

Note

When calling the controller in the pulse shaper cycle, you must note the following:

Calling the controller in the pulse shaper cycle will cause the process value to be averaged. As a result, at output PV, different values may be at input PV_IN and PV_PER. If you want to track the setpoint value, you must save the process value at input parameter PV_IN at the call times for complete controller processing (QC_ACT = TRUE). For pulse shaper calls occurring between these times, you must supply the input parameters PV_IN and SP_INT with the saved process value.

See also

Description TCONT_CP (Page 5581)

Mode of operation TCONT_CP (Page 5582)

Block diagram TCONT_CP (Page 5594)

Input parameters TCONT_CP (Page 5595)

Output parameters TCONT_CP (Page 5596)

In/out parameters TCONT_CP (Page 5597)

Static variables TCONT_CP (Page 5597)

Parameter STATUS_H (Page 5602)

Parameters STATUS_D (Page 5603)

Differences compared to TCONT_CP S7-300/400 (Page 2969)

Block diagram TCONT_CP (S7-300, S7-400)

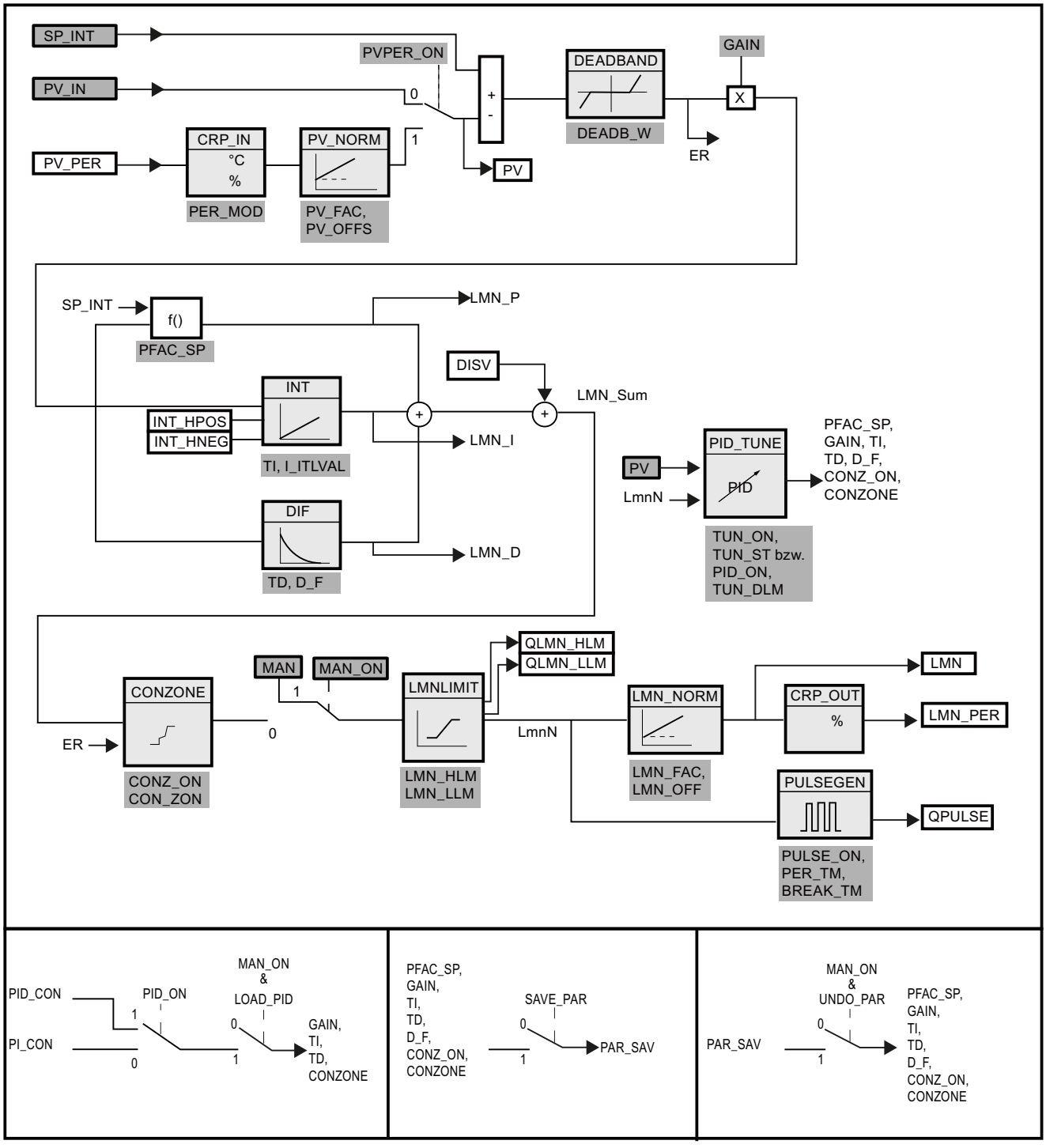

See also

Description TCONT_CP (Page 5581)
Mode of operation TCONT_CP (Page 5582)
Operating principle of the pulse generator (Page 5591)
Input parameters TCONT_CP (Page 5595)
Output parameters TCONT_CP (Page 5596)
In/out parameters TCONT_CP (Page 5597)
Static variables TCONT_CP (Page 5597)
Parameter STATUS_H (Page 5602)
Parameters STATUS_D (Page 5603)
Differences compared to TCONT_CP S7-300/400 (Page 2969)

Input parameters TCONT_CP (S7-300, S7-400)

Table 4-204

Parameters	Address	Data type	Default	Description
PV_IN	0.0	REAL	0.0	At the "Process value input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. The valid values depend on the sensors used.
PV_PER	4.0	INT	0	The process value in I/O format is interconnected with the controller at the "Process value I/O" input.
DISV	6.0	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input.
INT_HPOS	10.0	BOOL	FALSE	The output of the integral action can be held in the positive direction. For this, the input INT_HPOS must be set to TRUE. In a cascade control, INT_HPOS of the primary controller is connected to QLMN_HLM of the secondary controller.
INT_HNEG	10.1	BOOL	FALSE	The output of the integral action can be held in the negative direction. For this, the input INT_HNEG must be set to TRUE. In a cascade control, INT_HNEG of the primary controller is connected to QLMN_LLM of the secondary controller.
SELECT	12.0	INT	0	If the pulse shaper is on, there are several ways of calling the PID algorithm and pulse shaper: • SELECT = 0: The controller is called in a fast cyclic interrupt priority class and the PID algorithm and pulse shaper are processed. • SELECT = 1: The controller is called in OB1 and only the PID algorithm is processed. • SELECT = 2: The controller is called in a fast cyclic interrupt priority class and only the pulse shaper is processed. • SELECT = 3: The controller is called a slow cyclic interrupt priority class and only the PID algorithm is processed.

See also

Operating principle of the pulse generator (Page 5591)
Block diagram TCONT_CP (Page 5594)
Differences compared to TCONT_CP S7-300/400 (Page 2969)

Output parameters TCONT_CP (S7-300, S7-400)

Table 4-205

Parameter	Address	Data type	Default	Description
PV	14.0	REAL	0.0	The effective process value is output at the "Process value" output. The valid values depend on the sensors used.
LMN	18.0	REAL	0.0	The effective "Manipulated value" is output in floating point format at the "Manipulated value" output.
LMN_PER	22.0	INT	0	The manipulated value in I/O format is interconnected with the controller on the output "Manipulated value I/O".
QPULSE	24.0	BOOL	FALSE	The manipulated value is pulse-width-modulated at the QPULSE output.
QLMN_HLM	24.1	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output QLMN_HLM signals that the high limit has been reached.
QLMN_LLM	24.2	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output QLMN_LLM signals that the low limit has been reached.
QC_ACT	24.3	BOOL	TRUE	This parameter indicates whether continuous control component will be processed the next time the block is called (relevant only when SELECT has the value 0 or 1).

See also

Operating principle of the pulse generator (Page 5591)
Block diagram TCONT_CP (Page 5594)
Parameter STATUS_H (Page 5602)
Parameters STATUS_D (Page 5603)
Differences compared to TCONT_CP S7-300/400 (Page 2969)

In/out parameters TCONT_CP (S7-300, S7-400)

Table 4-206

Parameters	Address	Data type	Default	Description
CYCLE	26.0	REAL	0.1 s	Sets the sampling time for the PID algorithm. In phase 1, the tuner calculates the sampling time and enters it in CYCLE. CYCLE > 0.001 s
CYCLE_P	30.0	REAL	0.02 s	At this input, you set the sampling time for the pulse shaper action. In phase 1, the TCONT_CP instruction calculates the sampling time and enters it in CYCLE_P. CYCLE_P > 0.001 s
SP_INT	34.0	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. The valid values depend on the sensors used.
MAN	38.0	REAL	0.0	The "Manual value" input is used to set a manual value. In automatic mode, it tracks the manipulated value.
COM_RST	42.0	BOOL	FALSE	The block has an initialization routine that is processed when the COM_RST input is set.
MAN_ON	42.1	BOOL	TRUE	If the input "Enable manual mode" is set then the control loop is interrupted. The manual value MAN is set as manipulated value.

See also

Operating principle of the pulse generator (Page 5591)
Block diagram TCONT_CP (Page 5594)
Differences compared to TCONT_CP S7-300/400 (Page 2969)

Static variables TCONT_CP (S7-300, S7-400)

Table 4-207

Parameters	Address	Data type	Default	Description
DEADB_W	44.0	REAL	0.0	A deadband is applied to the control deviation. The "Deadband width" input determines the size of the deadband. The valid values depend on the sensors used.
I_ITLVAL	48.0	REAL	0.0	The output of the integrator can be set at the I_ITL_ON input. The initialization value is applied to the "Initialization value of the I-action" input. During a restart COM_RST = TRUE, the I-action is set to the initialization value. Values from -100 to 100 % are permitted.
LMN_HLM	52.0	REAL	100.0	The output value is always restricted to a high limit and low limit. The "Manipulated value high limit" input specifies the high limit. LMN_HLM > LMN_LLM
LMN_LLM	56.0	REAL	0.0	The output value is always restricted to a high limit and low limit. The "Manipulated value low limit" input specifies the low limit. LMN_LLM < LMN_HLM

Parameters	Address	Data type	Default	Description
PV_FAC	60.0	REAL	1.0	The "Process value factor" input is multiplied by the "Process value I/O". The input is used to scale the process value range.
PV_OFFS	64.0	REAL	0.0	The "Process value offset" input is added to the "Process value I/O". The input is used to scale the process value range.
LMN_FAC	68.0	REAL	1.0	The "Output value factor" input is multiplied with the output value. The input is used to scale the output value range.
LMN_OFFS	72.0	REAL	0.0	The "Output value offset" input is added to the output value. The input is used to scale the output value range.
PER_TM	76.0	REAL	1.0 s	The period duration of the pulse width modulation is entered at the PER_TM parameter. The relationship of the period duration to the sampling time of the pulse shaper determines the accuracy of the pulse width modulation. PER_TM ≥ CYCLE
P_B_TM	80.0	REAL	0.02 s	You can assign a minimum pulse or break time at the parameter "Minimum pulse/ break time". P_B_TM is internally limited to > CYCLE_P.
TUN_DLMN	84.0	REAL	20.0	Process excitation for controller tuning results from a output value step change at TUN_DLMN. Values from -100 to 100 % are permitted.
PER_MODE	88.0	INT	0	You can use this switch to enter the type of I/O module. The process value at input PV_PER is then scaled as follows at the PV output. • PER_MODE = 0: Thermoelements; PT100/NI100; standard PV_PER * 0.1 Unit: °C, °F • PER_MODE = 1: PT100/NI100; climate PV_PER * 0.01 Unit: °C, °F • PER_MODE = 2: Current/voltage PV_PER * 100/27648 Unit: %
PVPER_ON	90.0	BOOL	FALSE	If the process value is to be read in from the I/Os, the PV_PER input must be interconnected with the I/Os and the "Enable process value I/Os" input must be set.
I_ITL_ON	90.1	BOOL	FALSE	The output of the integrator can be set at the I_ITLVAL input. The "Set I-action" input must be set for this.
PULSE_ON	90.2	BOOL	FALSE	If PULSE_ON = TRUE is set, the pulse shaper is activated.
TUN_KEEP	90.3	BOOL	FALSE	The mode changes to automatic only when TUN_KEEP changes to FALSE.
ER	92.0	REAL	0.0	The effective control deviation is output at the "Control deviation" output. The valid values depend on the sensors used.
LMN_P	96.0	REAL	0.0	The "P-action" output contains the proportional action of the manipulated tag.
LMN_I	100.0	REAL	0.0	The "integral action" output contains the integral action of the manipulated tag.
LMN_D	104.0	REAL	0.0	The "D-action" output contains the derivative action of the manipulated tag.

Parameters	Address	Data type	Default	Description
PHASE	108.0	INT	0	The current phase of controller tuning is indicated at the PHASE output. • PHASE = 0: No tuning mode; automatic or manual mode • PHASE = 1: Ready to start tuning; check parameters, wait for excitation, measure the sampling times • PHASE = 2: Actual tuning: Searching for point of inflection with constant output value. Entering the sampling time in instance DB. • PHASE = 3: Calculating process parameters. Saving valid controller parameters prior to tuning. • PHASE = 4: Controller design • PHASE = 5: Following up the controller to the new manipulated tag • PHASE = 7: Validating the process type
STATUS_H	110.0	INT	0	STATUS_H indicates the diagnostic value via the search for the point of inflection during the heating process.
STATUS_D	112.0	INT	0	STATUS_D indicates the diagnostic value via the controller design during the heating process.
QTUN_RUN	114.0	BOOL	0	The tuning manipulated tag has been applied, tuning has started and is still in phase 2 (searching for point of inflection).
PI_CON	116.0	STRUCT		PI controller parameters
GAIN	+0.0	REAL	0.0	PI controller gain %/phys. unit
TI	+4.0	REAL	0.0 s	PI integration time [s]
PID_CON	124.0	STRUCT		PID controller parameters
GAIN	+0.0	REAL	0.0	PID controller gain
TI	+4.0	REAL	0.0s	PID integration time [s]
TD	+8.0	REAL	0.0s	PID derivative action time [s]
PAR_SAVE	136.0	STRUCT		The PID parameters are saved in this structure.
PFAC_SP	+0.0	REAL	1.0	Proportional factor for setpoint changes Values from 0.0 to 1.0 are permitted.
GAIN	+4.0	REAL	0.0	Controller gain %/phys. unit
TI	+8.0	REAL	40.0 s	Integration time [s]
TD	+12.0	REAL	10.0 s	Derivative action time (s)
D_F	+16.0	REAL	5.0	Derivative factor Values from 5.0 to 10.0 are permitted.
CON_ZONE	+20.0	REAL	100.0	Control zone band If the control deviation is greater than the control zone band, the high output value limit is output as output value. If the control deviation is less than the negative control zone band, the low output value limit is output as the output value. CON_ZONE ≥ 0.0
CONZ_ON	+24.0	BOOL	FALSE	Enable control zone

Parameters	Address	Data type	Default	Description
PFAC_SP	162.0	REAL	1.0	PFAC_SP specifies the effective P-action when there is a setpoint change. This is set between 0 and 1. • 1: P-action has full effect if the setpoint changes. • 0: P-action has no effect if the setpoint changes. Values from 0.0 to 1.0 are permitted.
GAIN	166.0	REAL	2.0	The "Proportional gain" input specifies controller amplification. The direction of control can be reversed by giving GAIN a negative sign. %/phys. unit
TI	170.0	REAL	40.0 s	The "Integration time" (integral-action time) input defines the integrator's time response.
TD	174.0	REAL	10.0 s	The "Derivative-action time" (rate time) input decides the time response of the differentiator.
D_F	178.0	REAL	5.0	The derivative factor decides the lag of the D-action. D_F = derivative-action time/"Lag of the D-action" Values from 5.0 to 10.0 are permitted.
CON_ZONE	182.0	REAL	100.0	If the control deviation is greater than the control zone band, the high output value limit is output as output value. If the control deviation is less than the negative control zone band, the low output value limit is output as the output value. The valid values depend on the sensors used.
CONZ_ON	186.0	BOOL	FALSE	You can use CONZ_ON =TRUE to enable the control zone.
TUN_ON	186.1	BOOL	FALSE	If TUN_ON=TRUE, the output value is averaged until the output value excitation TUN_DLMN is enabled either by a setpoint step-change or by TUN_ST=TRUE.
TUN_ST	186.2	BOOL	FALSE	If the setpoint is to remain constant during controller tuning at the operating point, a output value step-change by the amount of TUN_DLMN is activated by TUN_ST=1.
UNDO_PAR	186.3	BOOL	FALSE	Loads the controller parametersPFAC_SP, GAIN, TI, TD, D_FCONZ_ON and CON_ZONE from the data structure PAR_SAVE (only in manual mode).
SAVE_PAR	186.4	BOOL	FALSE	Saves the controller parameters PFAC_SP, GAIN, TI, TD, D_F, CONZ_ON and CON_ZONE in the data structure PAR_SAVE.
LOAD_PID	186.5	BOOL	FALSE	Loads the controller parametersGAIN, TI,TD depending on PID_ON from the data structure PI_CON or PID_CON (only in manual mode)
PID_ON	186.6	BOOL	TRUE	At the PID_ON input, you can specify whether or not the tuned controller will operate as a PI or PID controller. • PID controller: PID_ON = TRUE • PI controller: PID_ON = FALSE With certain process types it is nevertheless possible that only a PI controller will be designed despite PID_ON = TRUE.
GAIN_P	188.0	REAL	0.0	Identified process gain. In the case of process type I, GAIN_P tends to be estimated too low.
TU	192.0	REAL	0.0	Identified time lag of the process. TU ≥ 3*CYCLE
TA	196.0	REAL	0.0	Identified recovery time of the process. In the case of process type I, TA tends to be estimated too low.

Parameters	Address	Data type	Default	Description
KIG	200.0	REAL	0.0	Maximum process value rise at manipulated tag excitation from 0 to 100 % [1/s] GAIN_P = 0.01 * KIG * TA
N_PTN	204.0	REAL	0.0	The parameter specifies the order of the process. "Non-integer values" are also possible. Values from 1.01 to 10.0 are permitted.
TM_LAG_P	208.0	REAL	0.0	Time constants of a PTN model (practical values only for N_PTN >= 2).
T_P_INF	212.0	REAL	0.0	Time from process excitation until the point of inflection.
P_INF	216.0	REAL	0.0	Process value change from process excitation until the point of inflection. The valid values depend on the sensors used.
LMN0	220.0	REAL	0.0	Output value at the start of tuning Detected in phase 1 (mean value). Values from 0 to 100 % are permitted.
PV0	224.0	REAL	0.0	Process value at the start of tuning
PVDT0	228.0	REAL	0.0	Process value slew rate at start of tuning [1/s] Sign adapted.
PVDT	232.0	REAL	0.0	Current process value slew rate [1/s] Sign adapted.
PVDT_MAX	236.0	REAL	0.0	Max. change in the process value per second [1/s] Maximum derivative of the process value at the point of inflection (sign adapted, always > 0); is used to calculate TU and KIG.
NOI_PVDT	240.0	REAL	0.0	Noise action in PVDT_MAX in % The higher the noise action, the less accurate (less aggressive) the control parameters.
NOISE_PV	244.0	REAL	0.0	Absolute noise in process value Difference between maximum and minimum process value in phase 1.
FIL_CYC	248.0	INT	1	Number of cycles of the mean value filter The process value is determined through FIL_CYC cycles. FIL_CYC is increased from 1 to a max. of 1024 if needed.
POI_CMAX	250.0	INT	2	Maximum number of cycles after point of inflection This time is used to find another (i.e. better) inflection point for measuring noise. The tuning is completed only after this time.
POI_CYCL	252.0	INT	0	Number of cycles after inflection point

See also

Operating principle of the pulse generator (Page 5591)

Block diagram TCONT_CP (Page 5594)

Differences compared to TCONT_CP S7-300/400 (Page 2969)

Parameter STATUS_H (S7-300, S7-400)

STATUS_H	Description	Remedy
0	Default, or no/no new controller parameters	
10000	Tuning completed + suitable controller parameters found	
2xxxx	Tuning completed + controller parameters uncertain	
2xx2x	Point of inflection not reached (only if excited via setpoint step-change)	If the controller oscillates, weaken the controller parameters, or repeat the test with a smaller manipulated value difference TUN_DLMN.
2x1xx	Estimation error (TU < 3*CYCLE)	Reduce CYCLE and repeat attempt. Special case for PT1-only process: Do not repeat test, if necessary reduce controller parameters.
2x3xx	Estimation error TU too high	Repeat test under better conditions.
21xxx	Estimation error N_PTN < 1	Repeat test under better conditions.
22xxx	Estimation error N_PTN > 10	Repeat test under better conditions.
3xxxx	Tuning canceled in phase 1 owing to faulty parameter assignment:	
30002	Effective manipulated value differential < 5%	Correct manipulated value differential TUN_DLMN.
30005	The sampling times CYCLE and CYCLE_P differ by more than 5% of the measured values.	Compare CYCLE and CYCLE_P with the cycle time of the cyclic interrupt priority class and note any loop scheduler. Check CPU load. An excessively loaded CPU can result in prolonged sampling times that are inconsistent with CYCLE or CYCLE_P.

Note

If you cancel tuning in phase 1 or 2, STATUS_H = 0 is set. However, STATUS_D still displays the status of the last controller calculation.

The higher the value of STATUS_D, the higher the order of the control process, the greater the TU/TA ratio and the gentler the controller parameters will be.

See also

Operating principle of the pulse generator (Page 5591)

Block diagram TCONT_CP (Page 5594)

Differences compared to TCONT_CP S7-300/400 (Page 2969)

Parameters STATUS_D (S7-300, S7-400)

STATUS_D	Description
0	No controller parameters were calculated.
110	N_PTN <= 1.5 Process type I fast
121	N_PTN > 1.5 Process type I
200	N_PTN > 1.9 Process type II (transition range)
310	N_PTN >= 2.1 Process type III fast
320	N_PTN > 2.6 Process type III
111, 122, 201, 311, 321	Parameters have been corrected from phase 7.

Note

The higher the value of STATUS_D, the higher the order of the control process, the greater the TU/TA ratio and the gentler the controller parameters will be.

See also

Operating principle of the pulse generator (Page 5591)

Block diagram TCONT_CP (Page 5594)

Differences compared to TCONT_CP S7-300/400 (Page 2969)

TCONT_S (S7-300, S7-400)

Description TCONT_S (S7-300, S7-400)

The TCONT_S instruction is used on SIMATIC S7 automation systems to control technical temperature processes with binary manipulated value output signals for actuators with integrating behavior. The functionality is based on the PI control algorithm of the sampling controller. The step controller operates without a position feedback signal.

Application

You can also use the controller in a cascade control as a secondary position controller. You specify the actuator position via the setpoint input SP_INT. In this case, you must set the process value input and the parameter TI (integration time) to zero. An application might be, for example, temperature control with heating power control using pulse-break activation and cooling control using a butterfly valve. To close the valve completely, the manipulated variable (ER*GAIN) should be negative.

Call

The instruction TCONT_S must be called equidistant. To achieve this, use a cyclic interrupt priority class (for example, OB35 for an S7-300). The sampling time is specified at the CYCLE parameter.

If you call the instruction TCONT_S as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for TCONT_S directly in the multiple instance DB and commission it via a watch table.

CYCLE sampling time

The CYCLE sampling time match the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios).

The controller sampling time should not exceed 10% of the calculated integration time of the controller (TI). Generally, you must set the sampling time to a much lower value to achieve the required accuracy of the step controller.

Required accuracy G	MTR_TM	CYCLE = MTR_TM*G	Comment
0.5 %	10 s	0.05 s	The sampling time is determined by the required accuracy of the step controller.

Start-up

The TCONT_S instruction has an initialization routine that is run through when input parameter COM_RST = TRUE is set. Following execution of the initialization routine, the block sets COM_RST back to FALSE. All outputs are set to their initial values. If you require initialization when the CPU restarts, call the block in OB100 with COM_RST = TRUE.

See also

Block diagram TCONT_S (Page 5608)

Differences compared to TCONT_S S7-300/400 (Page 2969)

Mode of operation TCONT_S (S7-300, S7-400)

Setpoint branch

The setpoint is entered at input SP_INT in floating-point format as a physical value or percentage. The setpoint and process value used to form the control deviation must have the same unit.

Process value options (PVPER_ON)

Depending on PVPER_ON, the process value can be read in, in the I/O or floating-point format.

PVPER_ON	Process Value Input
TRUE	The process value is read in via the analog I/Os (PIW xxx) at input PV_PER.
FALSE	The process value is acquired in floating-point format at input PV_IN.

Process value format conversion CRP_IN (PER_MODE)

The CRP_IN function converts the I/O value PV_PER to floating-point format depending on the PER_MODE switch according to the following rules:

PER_MODE	Output of CRP_IN	Analog Input Type	Unit
0	PV_PER * 0.1	Thermoelements; PT100/NI100; standard	°C;°F
1	PV_PER * 0.01	PT100/NI100; climate;	°C;°F
2	PV_PER * 100/27648	Voltage/current	%

Process value scaling PV_NORM (PF_FAC, PV_OFFS)

The PV_NORM function calculates the output of CRP_IN according to the following rule:

"Output of PV_NORM" = "Output of CRP_IN)" * PV_FAC + PV_OFFS

It can be used for the following purposes:

- Process value adjustment with PV_FAC as process value factor and PV_OFFS as process value offset.
- Normalization of temperature to percentage
  You want to enter the setpoint as a percentage and must now convert the measured temperature value to a percentage.
- Normalization of percentage to temperature
  You want to enter the setpoint in the physical temperature unit and must now convert the measured voltage/current value to a temperature.

Calculation of the parameters:

- PV_FAC = range of PV_NORM/range of CRP_IN;
- PV_OFFS = LL (PV_NORM) - PV_FAC * LL(CRP_IN);
  where LL: low limit

The normalization is switched off with the default values (PV_FAC = 1.0 and PV_OFFS = 0.0). The effective process value is output at the PV output.

Example of Process Value Normalization

If you want to enter the setpoint as a percentage, and you have a temperature range of -20 to 85 °C applied to , CRP_IN you must normalize the temperature range as a percentage.

The following diagram shows an example of adapting the temperature range -20 to 85 °C to an internal scale of 0 to 100 %:

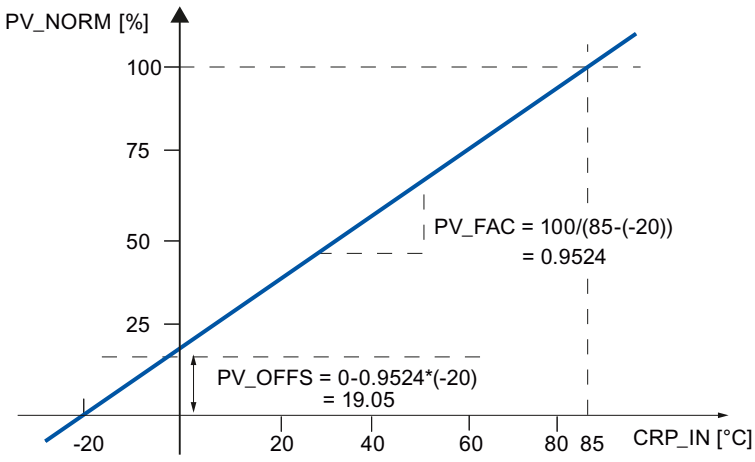

Forming the control deviation

The difference between the setpoint and process value is the control deviation before the dead band.

The setpoint and process value must exist in the same unit.

Dead band (DEADB_W)

To suppress a minor sustained oscillation due to the manipulated variable quantization (for example, in pulse width modulation with PULSEGEN) a dead band is applied to the (DEADBAND) control deviation. With DEADB_W = 0.0, the dead band is switched off.

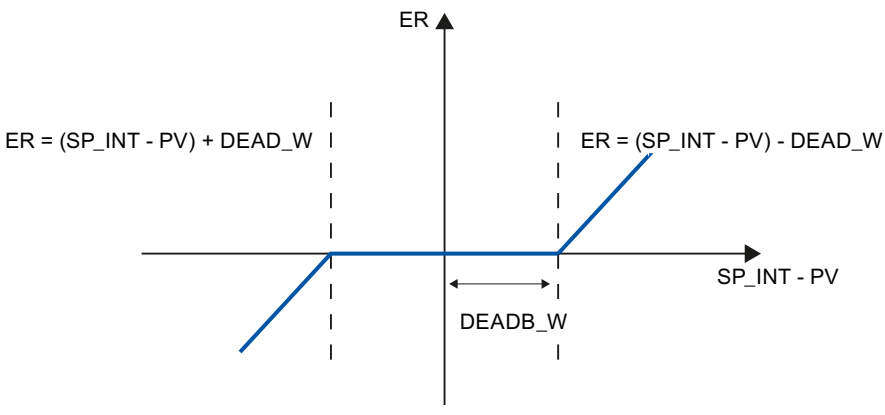

PI step controller algorithm

The instruction TCONT_S operates without position feedback. The I-action of the PI algorithm and the assumed position feedback signal are calculated in an integrator (INT) and compared as a feedback value with the remaining P-action. The difference is applied to a three-step element (THREE_ST) and a pulse shaper (PULSEOUT) that generates the pulses for the control valve. Adapting the response threshold of the three-step element reduces the switching frequency of the controller.

Weakening of the P-action when setpoint changes occur

To prevent overshoot, you can weaken the P-action using the "Proportional factor for setpoint changes" parameter (PFAC_SP). Using PFAC_SP, you can now select continuously between 0.0 and 1.0 to decide the effect of the P-action when the setpoint changes:

- PFAC_SP = 1.0: Proportional action for setpoint change is fully effective
- PFAC_SP = 0.0: Proportional action has no effect in the setpoint change

As in the case of the continuous controller, a value of PFAC_SP < 1.0 can reduce the overshoot if the motor run time MTR_TM is small compared with the recovery time TA and the ratio is TU/TA < 0.2. If MTR_TM reaches 20% of TA, only a slight improvement can still be achieved.

Feedforward control

A disturbance variable can be added at the DISV input.

Manual value processing (LMNS_ON, LMNUP, LMNDN)

With LMNS_ON, you can change between manual and automatic mode. In manual mode, the actuator is halted and the integral action (INT) is set to 0 internally. Using LMNUP and LMNDN, the actuator can be adjusted to OPEN and CLOSED. Switching over to automatic mode therefore involves a bump. As a result of the GAIN, the existing control deviation leads to a step change in the internal manipulated variable. The integral component of the actuator, however, results in a ramp-shaped excitation of the process.

See also

Block diagram TCONT_S (Page 5608)

Differences compared to TCONT_S S7-300/400 (Page 2969)

Block diagram TCONT_S (S7-300, S7-400)

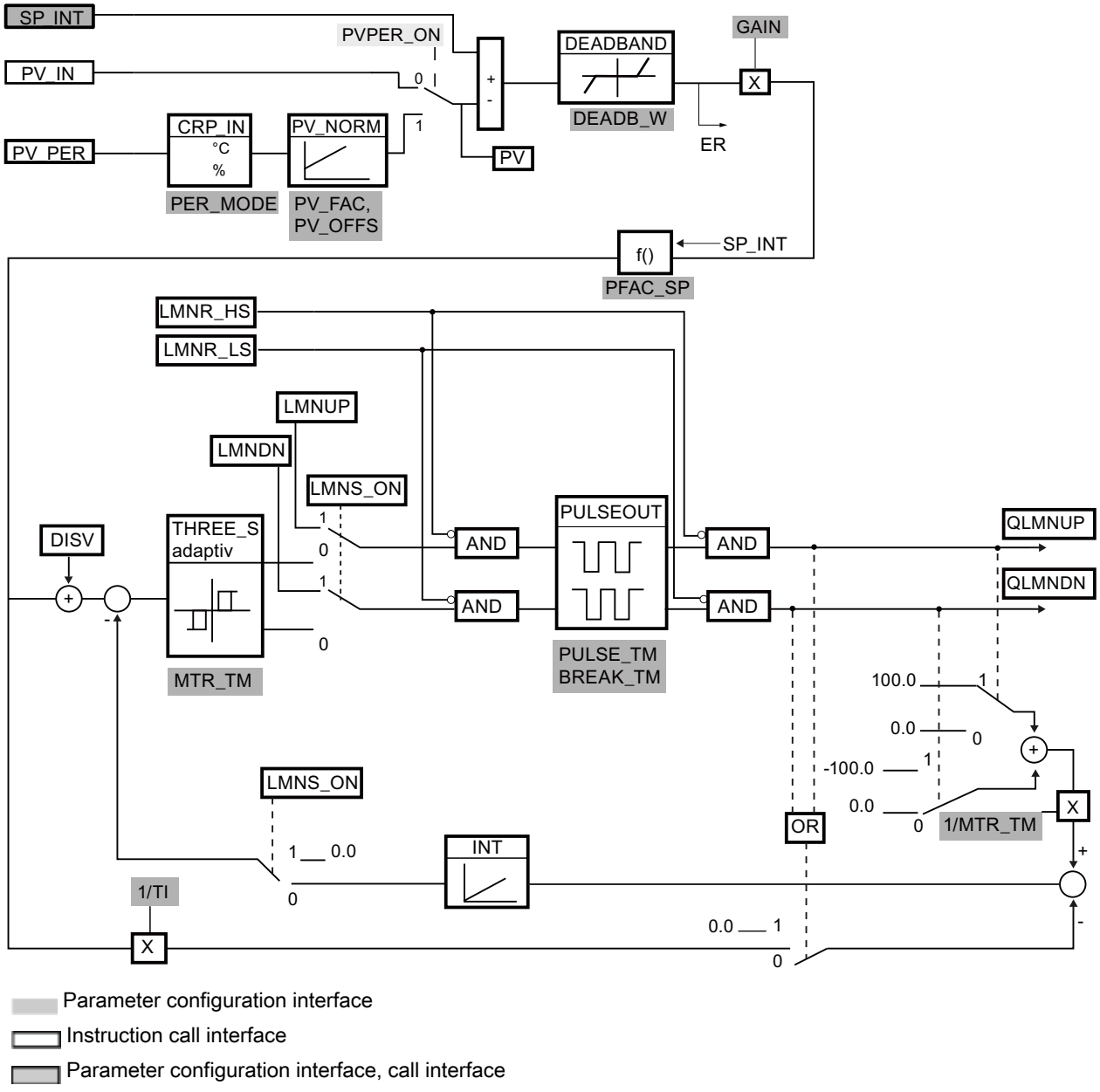

Parameter configuration interface

Instruction call interface

Parameter configuration interface, call interface

See also

Differences compared to TCONT_S S7-300/400 (Page 2969)

Description TCONT_S (Page 5603)

Mode of operation TCONT_S (Page 5604)

Input paramters TCONT_S (Page 5609)

Output parameters TCONT_S (Page 5610)
In/out parameters TCONT_S (Page 5610)
Static variables TCONT_S (Page 5610)

Input paramters TCONT_S (S7-300, S7-400)

Table 4-208

Parame-ters	Ad-dres s	Data type	De-fault	Description
CYCLE	0.0	REAL	0.1 s	At this input, you enter the sampling time for the controller. CYCLE ≥ 0.001
SP_INT	4.0	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. The valid values depend on the sensors used.
PV_IN	8.0	REAL	0.0	At the "Process variable input" you can assign parameters to a com-missioning value or you can interconnect an external process value in floating-point format. The valid values depend on the sensors used.
PV_PER	12.0	INT	0	The process value in I/O format is interconnected with the controller at the "Process value I/O" input.
DISV	14.0	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input.
LMNR_H S	18.0	BOO L	FAL SE	The signal "Control valve at high endstop" is interconnected on the input "High endstop signal of position feedback". • LMNR_HS=TRUE: The control valve is at high endstop.
LMNR_L S	18.1	BOO L	FAL SE	The signal "Control valve at low endstop" is interconnected on the input "Low endstop signal of position feedback". • LMNR_LS=TRUE: The control valve is at low endstop.
LMNS_O N	18.2	BOO L	TRU E	Manipulated value signal processing is switched to manual mode at the "Enable manual mode of manipulated signal".
LMNUP	18.3	BOO L	FAL SE	In manual mode of manipulated signals, the output parameter QLMNUP is operated at the input parameter "Manipulated signal up".
LMNDN	18.4	BOO L	FAL SE	In manual mode of the manipulated signals, the output parameter QLMNDN is operated at the input parameter "Manipulated signal down".

See also

Block diagram TCONT_S (Page 5608)
Differences compared to TCONT_S S7-300/400 (Page 2969)

Output parameters TCONT_S (S7-300, S7-400)

Table 4-209

Parameters	Address	Data type	Default	Description
QLMNUP	20.0	BOOL	FALSE	If the output "Manipulated value signal up" is set then the control valve should be open.
QLMNDN	20.1	BOOL	FALSE	If the output "Manipulated value signal down" is set then the control valve should be closed.
PV	22.0	REAL	0.0	The effective process value is output at the "Process value" output.
ER	26.0	REAL	0.0	The effective system deviation is output at the "Error signal" output.

See also

Block diagram TCONT_S (Page 5608)

Differences compared to TCONT_S S7-300/400 (Page 2969)

In/out parameters TCONT_S (S7-300, S7-400)

Table 4-210

Parameters	Address	Data type	Default	Description
COM_RST	30.0	BOOL	FALSE	The block has an initialization routine that is processed when the COM_RST input is set.

See also

Block diagram TCONT_S (Page 5608)

Differences compared to TCONT_S S7-300/400 (Page 2969)

Static variables TCONT_S (S7-300, S7-400)

Table 4-211

Parameters	Address	Data type	Default	Description
PV_FAC	32.0	REAL	1.0	The "Process value factor" input is multiplied by the process value. The input is used to scale the process value range.
PV_OFFS	36.0	REAL	0.0	The input "Process value offset" is added to the process value. The input is used to scale the process value range. The valid values depend on the sensors used.

Parameters	Address	Data type	Default	Description
DEADB_W	40.0	REAL	0.0	A deadband is applied to the control deviation. The "Deadband width" input determines the size of the deadband. DEADB_W ≥ 0.0
PFAC_SP	44.4	REAL	1.0	PFAC_SP specifies the effective P-action when there is a setpoint change. • 1: P-action has full effect if the setpoint changes. • 0: P-action has no effect if the setpoint changes. Values from 0.0 to 1.0 are permitted.
GAIN	48.0	REAL	2.0	The "Proportional gain" input specifies controller amplification. The direction of control can be reversed by giving GAIN a negative sign. %/phys. unit
TI	52.0	REAL	40.0 s	The "Integration time" (integral-action time) input defines the integrator's time response.
MTR_TM	56.0	REAL	30 s	The runtime from endstop to endstop of the control valve is entered at the "Motor actuating time" parameter. MTR_TM ≥ CYCLE
PULSE_TM	60.0	REAL	0.0 s	A minimum pulse time can be configured at the "Minimum pulse time" parameter.
BREAK_TM	64.0	REAL	0.0 s	You can assign a minimum break time at the parameter "Minimum break time".
PER_MODE	68.0	INT	0	You can use this switch to enter the type of I/O module. The process value at input PV_PER is then scaled as follows at the PV output. • PER_MODE = 0: Thermoelements; PT100/NI100; standard PV_PER * 0.1 Unit: °C, °F • PER_MODE = 1: PT100/NI100; climate PV_PER * 0.01 Unit: °C, °F • PER_MODE = 2: Current/voltage PV_PER * 100/27648 Unit: %
PVPER_ON	70.0	BOOL	FALSE	If the process value is to be read in from the I/Os, the PV_PER input must be interconnected with the I/Os and the "Enable process value I/Os" input must be set.

See also

Block diagram TCONT_S (Page 5608)

Differences compared to TCONT_S S7-300/400 (Page 2969)

Integrated system functions (S7-300, S7-400)

CONT_C_SF (S7-300, S7-400)

CONT_C_SF

The instruction CONT_C_SF is integrated in the S7-300 compact CPUs. The instruction must not be transmitted to the S7-300 CPU during loading. The scope of function corresponds with the instruction CONT_C.

See also

Description CONT_C (Page 5560)

How CONT_C works (Page 5561)

CONT_C block diagram (Page 5563)

Input parameter CONT_C (Page 5564)

Output parameters CONT_C (Page 5566)

CONT_S_SF (S7-300, S7-400)

CONT_S_SF

The instruction CONT_S_SF is integrated in the S7-300 compact CPUs. The instruction must not be transmitted to the S7-300 CPU during loading. The scope of function corresponds with the instruction CONT_S.

See also

Description CONT_S (Page 5566)

Mode of operation CONT_S (Page 5567)

Block diagram CONT_S (Page 5569)

Input parameters CONT_S (Page 5570)

Output parameters CONT_S (Page 5571)

PULSEGEN_SF (S7-300, S7-400)

PULSEGEN_SF

The instruction PULSEGEN_SF is integrated in the S7-300 compact CPUs. The instruction must not be transmitted to the S7-300 CPU during loading. The scope of function corresponds with the instruction PULSEGEN.

See also

Description PULSEGEN (Page 5571)
Mode of operation PULSEGEN (Page 5573)
Mode of operation PULSEGEN (Page 5576)
Three-step control (Page 5576)
Two-step control (Page 5578)
Input parameters PULSEGEN (Page 5580)
Output parameter PULSEGEN (Page 5581)

PID Self Tuner (S7-300, S7-400)

TUN_EC (S7-300, S7-400)

Description TUN_EC (S7-300, S7-400)

The TUN_EC instruction is used to tune a continuous PID controller. Depending on the controller, you interconnect TUN_EC with one of the following instructions:

- FM 355 C, FM 355 S pulse controller, FM 455 C or FM 455 S pulse controller
  - PID_FM
- PID Basic Functions
  - CONT_C with or without PULSEGEN
- Standard PID Control (PID Professional optional package)
  - PID_CP
- Modular PID Control (PID Professional optional package)
  - PID
  - LMNGEN_C

Call

TUN_EC is called in constant intervals of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

When ADAPT1ST = TRUE , the initialization routine of the TUN_EC instruction is run. The following pre-assignments are made:

- The system switches to PI controller. The parameters of PI_CON are transferred to the controller for this purpose. If the PI_CON.GAIN and PI_CON.TI parameters both have the value 0.0, they are pre-assigned with PI_CON.GAIN = 1.5 and PI_CON.TI = 3600.0 s.
- PROCESS.GAIN = 999.0

- RATIOFAC = 1.0
- STATUS_H = 0; STATUS_D = 0; STATUS_C = 0

Note

Unless explicitly required, do **not** call TUN_ES during a warm restart of the CPU in its initialization routine as is usual for other instructions. If you do, all process parameters will be lost.

If you call the instruction TUN_EC as a multiple instance DB, no technology object is created. No commissioning interface is available. You must commission TUN_EC using a watch table.

CYCLE sampling time

You enter the sampling time in the CYCLE parameter. The sampling time should not exceed 10% of the determined integral action time of the controller. It must agree with the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios).

In order for the process parameters to be identified reliably, the smallest time constant of the process (delay time) should be sampled at least 10 to 20 times.

In connection with the FM355/455 controller modules, the sampling time of TUN_EC must be approximately equal to the resulting cycle time of the controllers on the FM (Parameter setting > Module parameters).

Reaction to error

The error message word RET_VAL is not evaluated by the block.

See also

Mode of operation TUN_EC (Page 5617)

Input parameters TUN_EC (Page 5622)

Output parameters TUN_EC (Page 5624)

In/out parameters TUN_EC (Page 5626)

Parameter TUN_EC.PHASE (Page 5627)

Parameter TUN_EC.STATUS_H (Page 5630)

Parameter TUN_EC.STATUS_H (Page 5631)

Parameter TUN_EC.STATUS_D (Page 5631)

Scope of TUN_EC (S7-300, S7-400)

The Self-Tuner can be used particularly well for:

- Temperature control systems, but also for
- Level control systems and
- Flow control systems

In the case of flow control systems a distinction must be made as to whether only the control valve itself is to be controlled or whether a process with associated delay follows the control valve. In the case of control valve control only, the Self-Tuner cannot be used.

Process requirements

The process must meet the following requirements:

- **Stable asymptotic transient response with associated delay**
  The process value must settle to steady state after a manipulated variable jump. This therefore excludes processes that already exhibit an oscillating response in the absence of closed-loop control, as well as controlled systems without inherent regulation (integrator in the controlled system).
- **Delay times not too large**
  The range of application can be specified based on the ratio of the delay time t_u and response time t_a. The delay time also encompasses any existing dead time.
  The pretuning or adaptation is designed for the range $t_u < 1/10\ t_a$, within which most controlled temperature systems fall.
  For the range of especially long delay times $1/10\ t_a < t_u < 1/3\ t_a$, the design of a utilizable PI controller is generally still also possible.
- **Adequate linear response across a sufficiently large operating range**
  This means that non-linear effects within this operating range can be ignored both during identification and during normal closed-loop control operation. However, it is possible to re-identify the process when there is a change from one operating point to another if the adaptation process is carried out again close to the new operating point and provided that the non-linearity is not passed through during adaptation.
  If certain static non-linearities (e.g., valve characteristic curves) are known, it is always advisable to compensate for them in advance with a polyline in order to linearize the process response.
- **Minimal disturbances in temperature processes**
  Disturbances such as thermal transfer to neighboring zones or heat gains or losses due to changes from one state of matter to another must not influence the overall temperature process to any great extent. If necessary, adaptation at the operating point may be necessary. For example, when zones of an extruder are tuned, all zones must be heated up simultaneously.

- **Quality of the measuring signals**
  The quality of the measured signals has to be sufficient; in other words, the signal-to-disturbance ratio has to be sufficiently high.
- **Process gains not too high**
  The process gains must not be too high. Normalization of the process values is not required. The process gain K can thus include physical units, such as:
  K =ΔPV/ΔLMN, [K]= °C/%
  The final controller design is based on a calculation of the process gain K and can thus in principle compensate for any values of K. However, K is initially unknown during the learning phase and overshoots cannot be avoided when there are extreme combinations of gain and learning step changes. The overshoot is also reduced by reducing the parameter LHLM_TUN in the heating process or by increasing LLLM_TUN in the cooling process.

Requirements for processes with a control valve with integral action

In processes with control valves with an integral action, further requirements must be met in addition to those specified above:

The motor run time of the control valve must be less than the time required to find a point of inflection after a manipulated variable jump (also refer to the process response figure). If this is not the case, the process involved is often a flow rate control system in which only the control valve is effective as the dominating process action. It is not advisable to use the Self-Tuner in this case. You can then set the PI step controller manually in accordance with the following rule of thumb:

GAIN = 1, TI = Motor transition time

See also

Mode of operation TUN_EC (Page 5617)

Input parameters TUN_EC (Page 5622)

Output parameters TUN_EC (Page 5624)

In/out parameters TUN_EC (Page 5626)

Parameter TUN_EC.PHASE (Page 5627)

Parameter TUN_EC.STATUS_H (Page 5630)

Parameter TUN_EC.STATUS_H (Page 5631)

Parameter TUN_EC.STATUS_D (Page 5631)

Mode of operation TUN_EC (S7-300, S7-400)

TUN_EC has the following functions for the control and optimization of the controller:

- **Manual mode**

  The Self-Tuner forces the controller into manual mode and transfers its manual parameters to it.

- **Manual/automatic changeover with predictive manipulated-value specification**

  After the controller has been optimized successfully, the Self-Tuner can change over the controller from manual mode to automatic mode by feedforwarding a predictive manipulated value in such a manner that the process settles rapidly into the setpoint. This type of changeover involves a bump. You can also set bumpless changeover at the Self-Tuner.

- **Controller tuning**

  The process is first identified during the optimization of the controller parameters. A controller is designed based on the process characteristics. The new system parameters are calculated on this basis and these are activated for the controller.

  – **First adaption**

    Pretuning can only be carried out for a positive setpoint jump. It is usually used during initial commissioning or in case of a serious change in the controlled system characteristics.

  – **Fine tuning**

    The controller can be fine tuned online during a positive setpoint jump when there is a change from one operating point to another or when there are changes in the process response.

  – **Cooling tuning**
    For control systems that operate with two opposing actuators (for example, in case of temperature control: heating and cooling actuators) the Self-Tuner determines the ratio of the process gains under the influence of a second actuator in the case of a negative manipulated value step (using the same example of a temperature control: under influence of the cooling actuator).

  Tuning the response to setpoint changes
  The controller is designed for optimum disturbance response, in principle. The resultant "Stringent" parameters would result in overshoots of 10% to 20 % of the step height at setpoint steps.
  The **control zone** represents an initial remedy. To tune the start-up control response for large setpoint steps, the Self-Tuner calculates a control zone band during optimization. The controller is active within the band. Outside the band the Self-Tuner controls the controller with the maximum or minimum manipulated variable of the controller.
  The **structure segmentation** represents an additional remedy. In this case, the proportional or derivative action is located in the feedback loop. A change in the setpoint thus only acts on the integral action of the controller.
  If structure segmentation is not possible for the controller or if the control zone fails, the following function can be activated to tune the command characteristics:

- **Structure switchover:**
  Temporary disabling of the I-action or predictive manipulated value specification for large positive setpoint steps. This function is not effective when the control zone is activated.
- **Saving the controller parameters**

  If you assess the current controller parameters to be utilizable, you can save these in specially provided parameters in the instance DB of the Self-Tuner, before a manual change in parameters. If you tune the controller, the saved parameters are overwritten by the values that were valid prior to tuning.
- **Reloading saved controller parameters**
  The last controller parameter settings you saved can be activated for the controller again using this function.
- **Changing the controller parameters**

  If you want to set the controller parameters yourself for the controller, you can assign these through special parameters in the instance DB of the Self-Tuner and transfer them to the controller by means of this function.

Enabling the functions

Function	Start bit	Enable bit	Start condition: PHASE	Start condition: Setpoint step-change at SP
Manual mode	—	MAN_ON	0 to 7	—
Manual/automatic changeover with predictive manipulated-value specification	—	PRED_ON	Transition 7 -> 4	—
First adaption	ADAPT1ST	—	0, 4	Pos.: > MIN_STEP
First adaption or fine tuning	ADAPT_ON	—	4	Pos.: > MIN_STEP
Cooling tuning	COOLID_ON	—	4	—
Control zone	—	CONZ_ON	4	
Structure switchover	—	STRUC_ON	4	Pos.: > MIN_STEP, Neg: > MIN_STEP × RATIO_FAC
Save controller parameters	SAVE_PAR	—	2 to 7	—
Reloading controller parameters	UNDO_PAR	—	4	—
Change controller parameters	LOAD_PAR	—	4	—

Note

Start bits are reset after the function has been started successfully. Enable bits remain set.

See also

Description TUN_EC (Page 5613)
Input parameters TUN_EC (Page 5622)
Output parameters TUN_EC (Page 5624)
In/out parameters TUN_EC (Page 5626)
Parameter TUN_EC.PHASE (Page 5627)
Parameter TUN_EC.STATUS_H (Page 5630)
Parameter TUN_EC.STATUS_H (Page 5631)
Parameter TUN_EC.STATUS_D (Page 5631)

Control zone and structure switchover (S7-300, S7-400)

Control zone

The Self-Tuner operates with a control zone. This means that the controller operates based on the following algorithm:

- If the control deviation is greater than CON_ZONE, the system is heated with maximum power.
- If the control deviation is less than (-CON_ZONE) × RATIOFAC, the system is cooled with maximum power.

During pretuning or fine tuning, a CON_ZONE control zone is determined by the Self-Tuner and activated if the process type is suitable: CONZ_ON = TRUE. In closed-loop control, you can change the control zone or deactivate it completely (with CONZ_ON = FALSE).

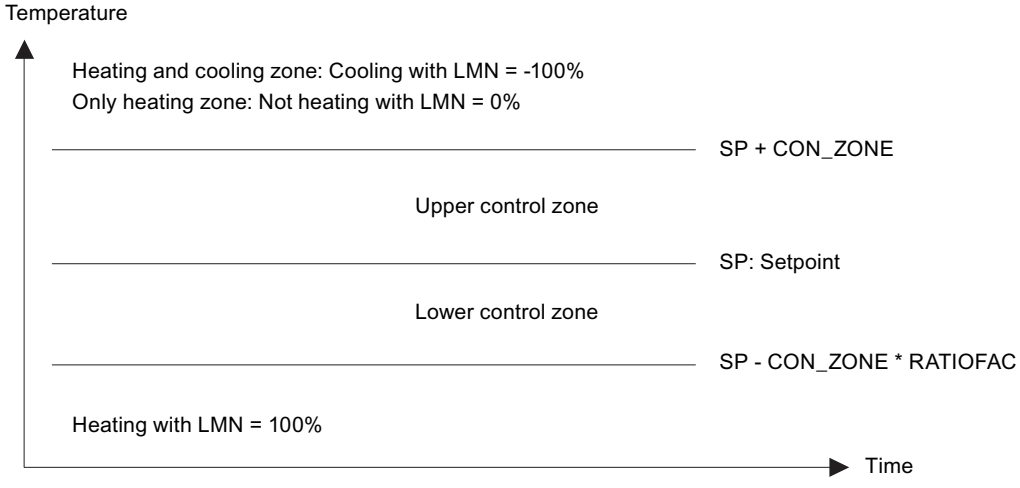

When the control zone is activated (CONZ_ON = TRUE), the structure switchover STRUC_ON is not effective.

Note

When there is a separate controller design for controlled systems with long delay time, a control zone is not determined, for example when STATUS_D = 3 after adaptation.

Before you switch on the control zone manually, make sure that the control zone width is not too small. If the control zone width is set too narrow, oscillations occur in the manipulated variable and the process value trends.

Structure segmentation

Structure segmentation is a function of the controller. In the case of closed-loop control with structure segmentation, the proportional or derivative action lies within the feedback loop, so that a change in the setpoint only acts on the integral action of the controller. When setpoint jumps occur, the manipulated variable does not undergo a jump change but instead follows a ramp-shaped course. The disturbance response is not changed as a result.

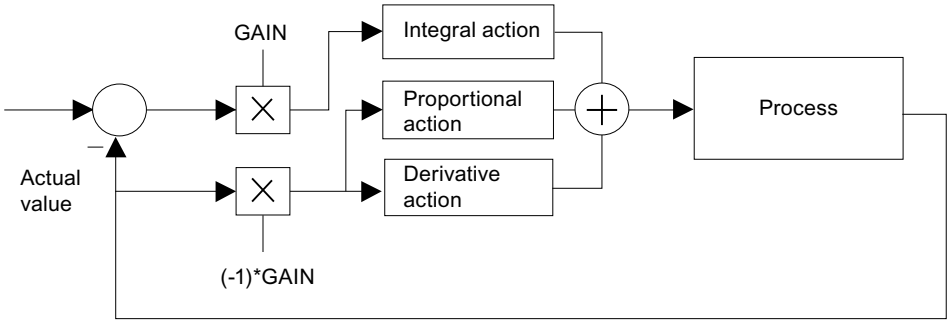

Figure 4-24 Closed-loop control with structure segmentation

In the case of closed-loop control without structure segmentation, the proportional and derivative actions lie within the control deviation, just like the integral action. You can select a structure switchover for this operating mode (STRUC_ON = TRUE).

Closed-loop control with structure segmentation is possible in the following control packages:

- FM 355/455
- Standard PID Control
- Modular PID Control

Note

Only closed-loop control without structure segmentation is possible for the simple controller PID Control of STEP 7 Basic and in the CPU 314 IFM. The structure switchover function should only be selected if there are no other alternatives.

Structure switchover when proportional and derivative actions are within the control deviation

The structure switchover is deactivated in the default setting. It should only be selected if no other alternatives such as, for example, control zone or structure segmentation, are available.

When the control zone is activated (CONZ_ON = TRUE), the structure switchover STRUC_ON is not effective.

You can activate the structure switchover with STRUC_ON = TRUE. The block automatically selects between two control measures depending on the total loop gain (product of the process gain and the controller gain):

- Integral action temporarily deactivated, P(D) control:
  In the case of a positive setpoint jump ≥ MIN_STEP, the integral action of the controller is deactivated temporarily and the gain is increased slightly, which means that a pure P(D) controller is used. In the proximity of the setpoint, the integral action is reactivated and the gain is reduced again (PHASE = 5).
- Temporary predictive manipulated variable specification:
  Processes with a high delay time cannot be controlled well with a P(D) control system. After a positive setpoint jump ≥ MIN_STEP, the manipulated variable required constantly for the new setpoint is therefore output constantly. In the proximity of the setpoint, the system is switched back bumplessly to PI or PID closed-loop control. This results in a slow, but overshoot-free response (PHASE = 6).

Note

If you do not achieve good results (transient response is slow) with the structure switchover in the case of positive setpoint jumps (for example, during heating processes), you can deactivate the structure switchover with STRUC_ON = FALSE, on the condition that small overshoots may occur.

Controller structure modes	CONZ_ON	PFDB_SEL, DFDB_SEL [1)]	STRUC_ON
Activate control zone	TRUE	Any	Any
Activate structure segmentation	Any	TRUE	FALSE
Activate structure switchover	FALSE	FALSE	TRUE

[1)] Controller parameters PFDB_SEL, DFDB_SEL: When CONT_C, these parameters are not available. This corresponds to the PFDB_SEL = DFDB_SEL = FALSE setting (without structure segmentation).

Note

Notes on selecting the operating modes:

- The Self-Tuner calculates a control zone and activates it automatically, if the ratio of the two time constants of the process is f = TM_LAG1 / TM_LAG2 > 10.
- In the case of jumps within the control zone, the structure segmentation that is selected by default acts, if allowed by the controller.
- If you want to compensate for small setpoint jumps within the control zone as quickly as possible and are willing to accept small overshoots, deactivate the structure segmentation. Numerical example: Control zone 20 °C, then overshoots of approx. 10% to 20% of the step size (< 20 °C) occur, i.e. less than 2 to 4 °C.
- The structure segmentation function prevents overshoots at setpoint jumps of all sizes, even if the process does not permit a control zone or you have deactivated it. However, the rise times are slightly slower than with a control zone.
- If the process does not allow a control zone and the controller does not allow a structure segmentation, activate the structure switchover. The structure switchover must not be activated at the same time as the structure segmentation.

Input parameters TUN_EC (S7-300, S7-400)

Table 4-212

Parameter	Address	Data type	Default	Description
SP	0.0	REAL	0.0	Enter the setpoint at the SP input. It is interconnected with the controller via SP_OUT. Note that the setpoint must lie within the setpoint limits of the controller.
PV	4.0	REAL	0.0	PV is interconnected with the process value of the controller.
LMN	8.0	REAL	0.0	LMN is interconnected with the manipulated variable of the controller. Values from -100 to 100% are permitted.
MAN	12.0	REAL	0.0	Enter the manual value at the MAN input. If MAN_ON = TRUE, the manual value MAN is transferred to the manipulated variable of the controller. If MAN_ON = TRUE and MAN = 0.0, the heating/cooling function is off. If MAN_ON = FALSE, the manipulated variable is specified by the controller. Values from -100 to 100% are permitted.
MIN_STEP	16.0	REAL	10.0	You can specify the minimum setpoint jump above which pretuning or adaptation can be carried out in the MIN_STEP input. If structure switchover (STRUC_ON = TRUE) is activated, the setpoint jump must be greater than MIN_STEP in order for the structure switchover activation to take place. Recommendation: MIN_STEP ≥ 10 % of the required setpoint-process value range
LHLM_TUN	20.0	REAL	80.0	At the LHLM_TUN input, specify the manipulated variable jump for the first setting. The manipulated variable jump is calculated automatically during subsequent fine tuning. Values from 0 to 100% are permitted.

Parameter	Address	Data type	Default	Description
LLLM_TUN	24.0	REAL	-20.0	You specify the manipulated variable jump for cooling tuning at the LLM_TUN input. Values from -100 to 0% are permitted.
NORM_FAC	28.0	REAL	1.0	A normalization factor is applied for controllers with a normalized controller gain (Standard PID Control). The normalization factor is calculated on the basis of the range limits of the process value of the controller and is interconnected with the NORM_FAC input.
MAN_ON	32.0	BOOL	TRUE	If MAN_ON = TRUE, the manipulated variable of the controller is specified via the MAN input. If MAN_ON = FALSE, the manipulated variable is specified by the controller.
PID_ON	32.1	BOOL	TRUE	At the PID_ON input, you can specify whether or not the tuned controller will operate as a PI or PID controller. PID controller: PID_ON = TRUE PI controller: PID_ON = FALSE With certain controlled system types, however, it is possible that only a PI controller will be designed, despite PID_ON = TRUE.
STRUC_ON	32.2	BOOL	FALSE	If STRUC_ON = TRUE, a structure switchover (from PI(D) to P(D) control) ensures that overshoots are largely avoided at setpoint jumps. The STRUC_ON input must not be set = TRUE if the proportional action or derivative action is fed into the feedback loop of the controller.
WRITE_DIS	32.3	BOOL	FALSE	In connection with the controller module, the Self-Tuner must know whether the driver block is ready to send data to the FM. This is the case if LOAD_OP = LOAD_PAR = FALSE. The result of the OR operation of LOAD_OP and LOAD_PAR is interconnected with WRITE_DIS.
PRED_ON	32.4	BOOL	TRUE	When changing over to automatic mode you can specify whether you want to change over bumplessly from the last manual value or with bumps with a manipulated variable calculated by the Self-Tuner. If PRED_ON = TRUE (default setting) the changeover is carried out with bumps. The changeover with bumps is only effective after tuning and ensures rapid settling into the desired setpoint, in particular for a PI control system.
CYCLE	34.0	TIME	100 ms	The sampling time must be configured in the CYCLE input. CYCLE must correspond to the sampling time of the controller to be set and the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios). CYCLE > 1 ms

See also

Description TUN_EC (Page 5613)

Mode of operation TUN_EC (Page 5617)

Output parameters TUN_EC (Page 5624)

In/out parameters TUN_EC (Page 5626)

Parameter TUN_EC.PHASE (Page 5627)

Parameter TUN_EC.STATUS_H (Page 5630)

Parameter TUN_EC.STATUS_H (Page 5631)

Parameter TUN_EC.STATUS_D (Page 5631)

Output parameters TUN_EC (S7-300, S7-400)

Table 4-213

Parameter	Address	Data type	Default	Description
MAN_OUT	38.0	REAL	0.0	MAN_OUT is connected to the manual input value of the controller. If MAN_ON = TRUE, the Self-Tuner outputs the value specified at the MAN input to the MAN_OUT output.
SP_OUT	42.0	REAL	0.0	SP_OUT is connected to the setpoint input of the controller. The Self-Tuner writes the setpoint specified at the SP input to the SP_OUT output. If manual mode MAN_ON = TRUE, the setpoint is tracked to the process value: SP_OUT = PV.
GAIN	46.0	REAL	1.5	GAIN is interconnected with the proportional gain (controller gain) of the controller.
TI	50.0	TIME	1 h	TI is interconnected with the integral action time (reset time) of the controller.
TD	54.0	TIME	0 ms	TD is interconnected with the derivative action time (rate time) of the controller.
TM_LAG	58.0	TIME	1 s	TM_LAG is interconnected with the time lag of the controller differentiator.
RATIOFAC	62.0	REAL	1.0	RATIOFAC is interconnected with the heating/cooling ratio factor of the controller or of the PULSEGEN instruction. The parameter is only required for cooling.
PHASE	66.0	INT	0	Parameter TUN_EC.PHASE (Page 5627) displays the current sequence of the controller tuning. During the tuning of Standard PID Control, the PHASE output is interconnected with the PHASE input of the PID_CP instruction.
STATUS_H	68.0	INT	0	Parameter TUN_EC.STATUS_H (Page 5630) displays tuning results regarding the search for the point of inflection during the heating process.
STATUS_D	70.0	INT	0	Parameter TUN_EC.STATUS_D (Page 5631) displays tuning results regarding the controller design during the heating process.
STATUS_C	72.0	INT	0	Parameter TUN_EC.STATUS_H (Page 5631) displays tuning results regarding the search for the point of inflection during the cooling process.
QMAN_ON	74.0	BOOL	FALSE	QMAN_ON is interconnected with the manual/automatic changeover of the controller.
QI_SEL	74.1	BOOL	TRUE	QI_SEL is interconnected with the on/off switch of the controller integral action I_SEL. With the FM355/455, TI must be switched to zero depending on QI_SEL: If QI_SEL = FALSE, then TI = 0.
QD_SEL	74.2	BOOL	FALSE	QD_SEL is interconnected with the on/off switch of the controller derivative action D_SEL. With the FM355/455, TD must be switched to zero depending on QD_SEL: If QD_SEL = FALSE, then TD = 0.
QWRITE	74.3	BOOL	FALSE	QWRITE is interconnected with the LOAD_PAR and LOAD_OP inputs of the PID_FM instruction of FM355/455. As a result, the data records of the controller channel are downloaded to the FM.
PROCESS	76.0	STRUCT		The PROCESS structure contains the parameters of the process.
GAIN	+0.0	REAL	999.0	The process gain is displayed in the PROCESS.GAIN output.
TM_LAG1	+4.0	REAL	0.0	The slow process time constant is displayed at the PROCESS.TM_LAG1 output.
TM_LAG2	+8.0	REAL	0.0	The fast process time constant is displayed at the PROCESS.TM_LAG2 output.

Parameter	Address	Data type	Default	Description
PV00	+12.0	REAL	0.0	The zero mean value of the process is displayed in the PROCESS.PV00 output. The zero mean value is the steady-state process value when manipulated variable = zero. This corresponds to the ambient temperature when the system is cold.
KIG	+16.0	REAL	0.0	The maximum rise ratio of the process value during heating is displayed in the PROCESS.KIG output.
KIG_C	+20.0	REAL	0.0	The maximum fall ratio of the process value during cooling is displayed in the PROCESS.KIG_C output.
PI_CON	100.0	STRUCT		The PI_CON structure contains the parameters of the PI controller.
GAIN	+0.0	REAL	1.5	PI_CON.GAIN displays the proportional gain of the PI controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TI	+4.0	REAL	3600.0	PI_CON.TI displays the integral action time of the PI controller. You can improve the value manually and make it effective by setting LOAD_PAR.
PID_CON	108.0	STRUCT		The PID_CON structure contains the parameters of the PID controller.
GAIN	+0.0	REAL	1.5	PID_CON.GAIN displays the proportional gain of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TI	+4.0	REAL	3600.0	PID_CON.TI displays the integral action time of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TD	+8.0	REAL	0.0	PID_CON.TD displays the derivative action time of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
PI_CON_OLD	120.0	STRUCT		The PI_CON_OLD structure contains the old parameters of the PI controller.
GAIN	+0.0	REAL	1.5	The old proportional gain of the PI controller is displayed at the PI_CON_OLD.GAIN output.
TI	+4.0	REAL	3600.0	The old integral action time of the PI controller is displayed at the PI_CON_OLD.TI output.
PID_CON_OLD	128.0	STRUCT		The PID_CON_OLD structure contains the old parameters of the PID controller.
GAIN	+0.0	REAL	0.0	The old proportional gain of the PID controller is displayed at the PID_CON_OLD.GAIN output.
TI	+4.0	REAL	0.0	The old integral action time of the PID controller is displayed at the PID_CON_OLD.TI output.
TD	+8.0	REAL	0.0	The old derivative action time of the PID controller is displayed at the PID_CON_OLD.TD output.
TU	140.0	REAL	0.0	The determined delay time is displayed at the TU output.
TA	144.0	REAL	0.0	The determined recovery time is displayed at the TA output.

See also

Description TUN_EC (Page 5613)

Mode of operation TUN_EC (Page 5617)

Input parameters TUN_EC (Page 5622)

In/out parameters TUN_EC (Page 5626)

In/out parameters TUN_EC (S7-300, S7-400)

Table 4-214

Parameter	Address	Data type	Default	Description
ADAPT1ST	148.0	BOOL	FALSE	When you carry out the first optimization of the process or do not wish to use the results of previous optimizations such as manipulated value step size or filter time constants, you must carry out a first adaption. To do this, set ADAPT1ST = TRUE. The Self-Tuner then carries out an initialization routine and in the process sets the default values described in the initialization routine. The Self-Tuner sets ADAPT1ST back to FALSE. It does not make sense to set ADAPT1ST to TRUE at a restart of the CPU in OB100.
ADAPT_ON	148.1	BOOL	FALSE	If you want to trigger on-line adaption, set ADAPT_ON = TRUE. If the subsequent setpoint step-change is greater than MIN_STEP, identification is started (PHASE = 2). The Self-Tuner sets ADAPT_ON back to FALSE.
STEADY	148.2	BOOL	FALSE	When the process value has reached a steady state, you can use STEADY to terminate optimization prematurely, even though the Self-Tuner is still in PHASE 3. You can also use STEADY to carry out a renewed controller design in steady state, after the Self-Tuner has passed into PHASE 4.
COOLID_ON	148.3	BOOL	FALSE	If you want to trigger cooling tuning, set COOLID_ON = TRUE. The manipulated value is set to LLLM_TUN and COOLID_ON is reset to FALSE. If there are multiple heating zones, all pretuning and adaptations should have been completed before cooling tuning is started.
UNDO_PAR	148.4	BOOL	FALSE	The old controller parameters are transferred to the current controller parameters and are active immediately depending on PID_ON. PI_CON = PI_CON_OLD PID_CON = PID_CON_OLD The function for undoing changes can only be used in automatic mode (PHASE 4).
SAVE_PAR	148.5	BOOL	FALSE	The current controller parameters are transferred to the old controller parameters. PI_CON_OLD = PI_CON PID_CON_OLD = PID_CON
LOAD_PAR	148.6	BOOL	FALSE	The changed parameters PI_CON and PID_CON are transferred to the controller and are thus active.
CONZ_ON	148.7	BOOL	FALSE	If CONZ_ON = TRUE, the controller operates with a control zone. During controller optimization the Self-Tuner decides whether controlling is carried out with or without control zone and determines an optimum control zone. After optimization has been carried out in PHASE 4 you can de-activate the control zone with CONZ_ON = FALSE or re-configure the control zone.
CON_ZONE	150.0	REAL	0.0	CON_ZONE specifies the width of the control zone. During controller optimization the Self-Tuner decides whether controlling is carried out with or without control zone and determines an optimum control zone. After optimization has been carried out in PHASE 4 you can de-activate the control zone with CONZ_ON = FALSE or re-configure the control zone.

See also

Description TUN_EC (Page 5613)
Mode of operation TUN_EC (Page 5617)
Input parameters TUN_EC (Page 5622)
Output parameters TUN_EC (Page 5624)
Parameter TUN_EC.PHASE (Page 5627)
Parameter TUN_EC.STATUS_H (Page 5630)
Parameter TUN_EC.STATUS_H (Page 5631)
Parameter TUN_EC.STATUS_D (Page 5631)

Parameter TUN_EC.PHASE (S7-300, S7-400)

PHASE	Mode	Description
0	Automatic mode: Close-loop control with preset parameters	During the generation of an instance DB for the Self-Tuner, zero is initially assigned to the parameter PHASE. Since TRUE is initially assigned to MAN_ON, the Self-Tuner will change to PHASE 7 (manual mode) after the first run. You can start the first adaption directly from manual mode (ADAPT_ON = TRUE) or change over to automatic mode. When you change over to automatic mode MAN_ON = FALSE for the first time, the parameters are read in from PI_CON. The default values for GAIN and TI are: GAIN = 1.5, TI = 1h.
2	Controller optimization: Searching for point of inflection with constant manipulated value	As soon as you specify a setpoint step-change ≥ MIN_STEP in the positive direction to the operating point of the warm process state, LHLM_TUN is assigned to MAN_OUT during the first adaption, while a value calculated by Self-Tuner is assigned to MAN_OUT during on-line adaption. QMAN_ON = TRUE is also set. Both values are transferred to the PID controller. The PID controller is controlled with them in manual mode. MIN_STEP should be greater than 10% of the operating range of the setpoint and the process value.

PHASE	Mode	Description
3	Controller optimization: Closed-loop control with a PI coarse controller	If the point of inflection of the step response is recognized (STATUS_H = 2) or if the process value has reached 70% of the setpoint step height (STATUS_H = 3), a prudently set PI controller is designed. The controller gain GAIN is limited to the range of 0.2 to 20.0. The controller operates immediately as a PI controller and tries to settle the process to a steady state. The Self-Tuner saves the newly determined PI controller parameters in GAIN, TI and PI_CON. The previous values of the PI controller are saved in PI_CON_OLD. If it takes a long time until the steady state is reached (creeping transient response during temperature processes) you can initiate the control design with the current data when the state is almost steady by setting STEADY = TRUE. You can also re-initiate the controller design with the current data at a later stage in Phase 3 or 4 by specifying STEADY = TRUE. This often improves the controller design further. If an overshoot occurs or if no point of inflection is found, the cause may be that the manipulated value step size LHLM_TUN was selected too large during the first adaption. This does not necessarily produce a bad controller setting. You should select LHLM_TUN approx. 20% smaller at the next first adaption. If the block has recognized the steady state or if the period 30 × TI (TI: Integral-action time of the PI controller set under PHASE = 3) has passed since the setpoint step-change, an improved controller design is realized and the system changes over to PHASE = 4. The Self-Tuner saves the newly determined PI/PID controller parameters in GAIN, TI, TD and PI_CON or PID_CON. The previous values of the PID controller are saved in PID_CON_OLD. If possible, a PID controller and a PI controller are designed. If PID_ON = TRUE, the PID controller is activated, otherwise the PI controller is activated. In PHASE 4 you can switch over bumplessly at any time between the two controllers. The controller gain GAIN is limited in such a manner that the loop gain of the open loop (product of the process gain and the controller gain) lies under 80.0.
4	Automatic mode: Normal closed-loop control with/without control zone	In this phase the controller controls with its optimized parameters. From this state, you can start cooling tuning, fine tuning or pretuning.
5	Automatic mode: Structure switchover at setpoint step-change: I-action deactivated, P(D) control	In the case of a positive setpoint step-change ≥ MIN_STEP the I-action of the controller is de-activated temporarily and the gain is increased slightly, so that a pure P(D) controller is used. In setpoint proximity the I-action is re-activated and the gain is reduced again
6	Automatic mode: Structure switchover at setpoint step-change: Predictive manipulated value specification	Processes with a high time lag cannot be controlled well with a P(D) control system. After a positive setpoint step-change ≥ MIN_STEP, the steady state manipulated variable required for the new setpoint is therefore output as a constant. In setpoint proximity the system changes back bumplessly to PI or PID controller operating mode. This results in a slow action that is, however, free of overshooting.
7	Manual mode	If you set the input MAN_ON to TRUE, the output QMAN_ON is set to TRUE and the output MAN_OUT to MAN. This changes the PID controller over to manual mode. Manual mode takes priority over all other operating modes. Pretuning, fine tuning and cooling tuning are aborted. When manual operation is disabled (MAN_ON = FALSE), the controller changes over to automatic mode (PHASE = 4 in the operation of the Self-Tuner) and controls with the existing controller parameters.

Heating Optimization Phases during First Adaption

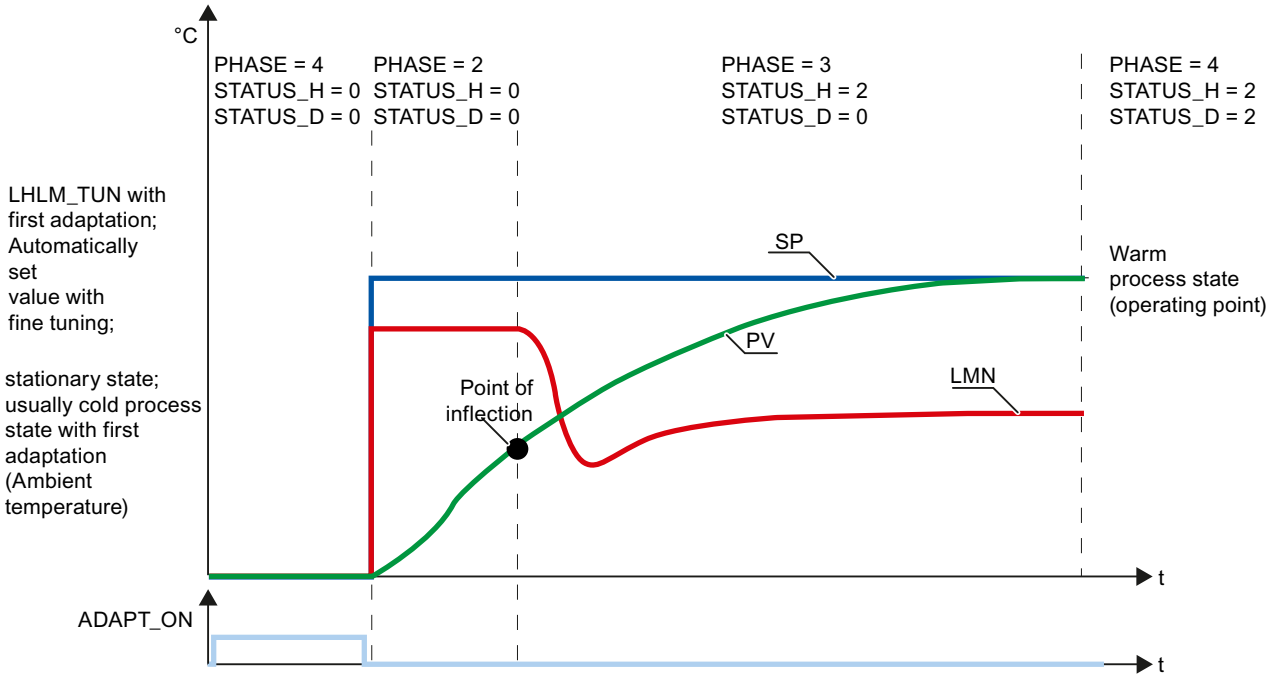

Cooling tuning phases

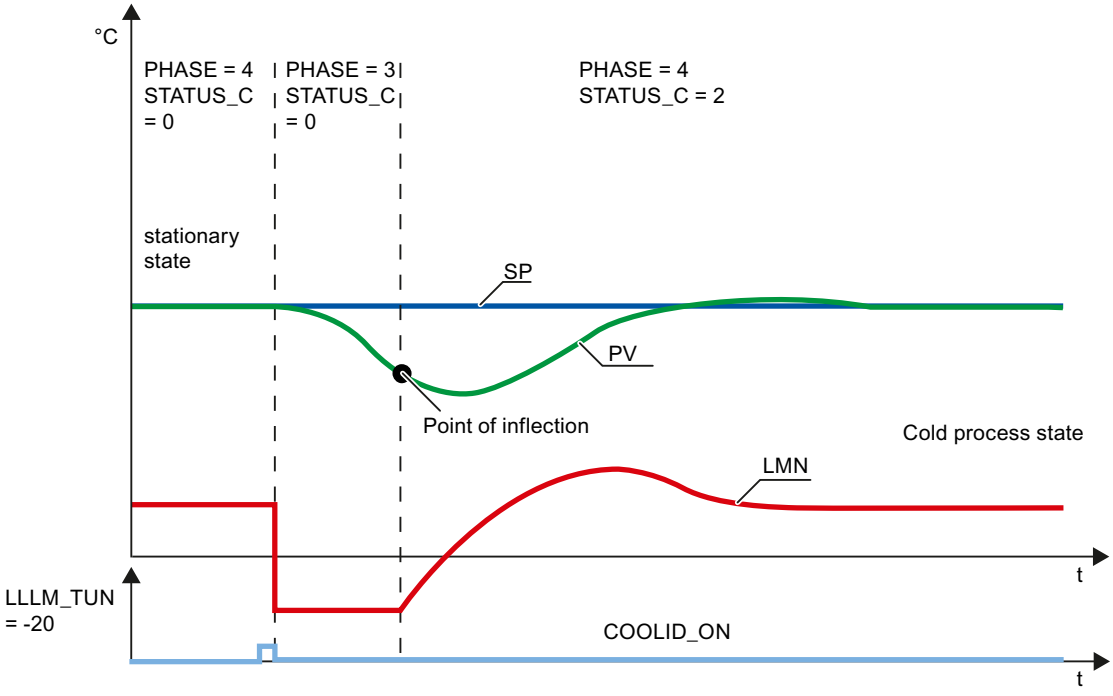

See also

Description TUN_EC (Page 5613)
Mode of operation TUN_EC (Page 5617)
Input parameters TUN_EC (Page 5622)
In/out parameters TUN_EC (Page 5626)
Parameter TUN_EC.STATUS_H (Page 5630)
Parameter TUN_EC.STATUS_H (Page 5631)
Parameter TUN_EC.STATUS_D (Page 5631)

Parameter TUN_EC.STATUS_H (S7-300, S7-400)

STATUS_H	Description	Remedy
0	Initialization	
1	Point of inflection too low In most cases optimization still results in good controller parameters	To get closer the ideal case, increase LHLM_TUN or decrease the setpoint step-change size.
2	Ideal case: Point of inflection found	
3	Point of inflection too high A safety factor is included. In most cases optimization leads to controller parameters that can still be used but are prudent.	To get closer the ideal case, decrease LHLM_TUN or increase the setpoint step-change size.
11, 12	The time lag could not be detected correctly. Operation continues with an estimated value that can lead to controller parameters that are not optimum. If the process has a pure **VZ1 behavior** (no time lag), the Self-Tuner can often design good controller parameters that are still good. Since calculation is continued with a roughly estimated time lag, it is possible that STATUS_D cannot be determined to 1 (controller for approximating VZ1 behavior) and that the control zone is de-activated, although a control zone makes sense for VZ1 processes.	Repeat optimization and ensure that disturbances do not occur at the process value. If you are not satisfied with the controller parameters, you can at least use the values from PROCESS.GAIN and PROCESS.TM_LAG1 for a VZ1 model and use your own controller design in accordance with the text book.
13	The time lag could not be detected correctly. Operation continues with an estimated value that can lead to controller parameters that are not optimum.	Repeat optimization and ensure that disturbances do not occur at the process value.

See also

Description TUN_EC (Page 5613)
Mode of operation TUN_EC (Page 5617)
Input parameters TUN_EC (Page 5622)
In/out parameters TUN_EC (Page 5626)
Parameter TUN_EC.PHASE (Page 5627)

Parameter TUN_EC.STATUS_H (Page 5631)
Parameter TUN_EC.STATUS_D (Page 5631)

Parameter TUN_EC.STATUS_H (S7-300, S7-400)

STATUS_C	Description	Remedy
0	Initialization	
1	Turning point was not found. In most cases, optimization still results in good controller parameters.	To get closer to the ideal case, increase LLLM_TUN (for example from -20.0 to -10.0)
2	Ideal case: Cooling point of inflection found	
11, 12, 13	Incorrect estimate The ratio factor is set to the default value: RATIOFAC = 1.0	Repeat optimization and ensure that disturbances do not occur at the process value.

See also

Description TUN_EC (Page 5613)
Mode of operation TUN_EC (Page 5617)
Input parameters TUN_EC (Page 5622)
In/out parameters TUN_EC (Page 5626)
Parameter TUN_EC.PHASE (Page 5627)
Parameter TUN_EC.STATUS_H (Page 5630)
Parameter TUN_EC.STATUS_D (Page 5631)

Parameter TUN_EC.STATUS_D (S7-300, S7-400)

STATUS_D	Description	Remedy
0	Initialization	
1	The process has almost PT1 behavior;	you can possibly waive the D-action of the contoller.
2	PI and PID controllers possible, VZ2 behavior	
3	Separate PID controller design for very high delay times, dead times or high process order In this case the control zone is switched off.	
4	Incorrect estimate No controller design for on-line adaption. During the first adaption the process is continued with the coarse PI controller from PHASE 3.	Repeat optimization and ensure that disturbances do not occur at the process value.

See also

Description TUN_EC (Page 5613)
Mode of operation TUN_EC (Page 5617)
Input parameters TUN_EC (Page 5622)
In/out parameters TUN_EC (Page 5626)
Parameter TUN_EC.PHASE (Page 5627)
Parameter TUN_EC.STATUS_H (Page 5630)
Parameter TUN_EC.STATUS_H (Page 5631)

TUN_ES (S7-300, S7-400)

Description TUN_ES (S7-300, S7-400)

The TUN_ES instruction is used to tune a PID step controller. Depending on the controller, you interconnect TUN_ES with one of the following instructions:

- FM 355 S or FM 455 S
  - PID_FM
- PID Basic Functions
  - CONT_S
  - TCONT_S
- Standard PID Control (PID Professional optional package)
  - PID_ES
- Modular PID Control (PID Professional optional package)
  - PID
  - LMNGEN_S

Call

TUN_ES is called in the constant time base of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

When ADAPT1ST = TRUE , the initialization routine of the TUN_ES instruction is run. The following pre-assignments are made:

- The system switches to PI controller. The parameters of PI_CON are transferred to the controller for this purpose. If the PI_CON.GAIN and PI_CON.TI parameters both have the value 0.0, they are pre-assigned with PI_CON.GAIN = 1.5 and PI_CON.TI = 3600.0 s.
- PROCESS.GAIN = 999.0
- STATUS_H = 0; STATUS_D = 0

Note

Unless explicitly required, do **not** call TUN_ES during a warm restart of the CPU in its initialization routine as is usual for other instructions. If you do, all process parameters will be lost.

If you call the instruction TUN_ES as a multiple instance DB, no technology object is created. No commissioning interface is available. You must commission TUN_EC using a watch table.

CYCLE sampling time

You enter the sampling time in the CYCLE parameter. The sampling time should not exceed 10% of the determined integral action time of the controller. In addition, the sampling time should not exceed 1% of the motor transition time so that the position feedback has a resolution of at least 1%. You can set the sampling time in the CYCLE parameter of the TUN_ES instruction and the controller. It must agree with the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios).

In connection with the FM355/455 controller modules, the sampling time of TUN_ES must be approximately equal to the resulting cycle time of the controllers on the FM (Parameter setting > Module parameters).

Reaction to error

The error message word RET_VAL is not evaluated by the block.

See also

Mode of operation TUN_ES (Page 5636)

Input parameters TUN_ES (Page 5640)

Output parameters TUN_ES (Page 5641)

In/out parameters TUN_ES (Page 5644)

Parameter TUN_ES.PHASE (Page 5645)

Parameter TUN_ES.STATUS_H (Page 5647)

Parameter TUN_ES.STATUS_D (Page 5648)

Scope of TUN_ES (S7-300, S7-400)

The Self-Tuner can be used particularly well for:

- Temperature control systems, but also for
- Level control systems and
- Flow control systems

In the case of flow control systems a distinction must be made as to whether only the control valve itself is to be controlled or whether a process with associated delay follows the control valve. In the case of control valve control only, the Self-Tuner cannot be used.

Process requirements

The process must meet the following requirements:

- **Stable asymptotic transient response with associated delay**
  The process value must settle to steady state after a manipulated variable jump. This therefore excludes processes that already exhibit an oscillating response in the absence of closed-loop control, as well as controlled systems without inherent regulation (integrator in the controlled system).
- **Delay times not too large**
  The range of application can be specified based on the ratio of the delay time t_u and response time t_a. The delay time also encompasses any existing dead time.
  The pretuning or adaptation is designed for the range $t_u < 1/10\ t_a$, within which most controlled temperature systems fall.
  For the range of especially long delay times $1/10\ t_a < t_u < 1/3\ t_a$, the design of a utilizable PI controller is generally still also possible.
- **Adequate linear response across a sufficiently large operating range**
  This means that non-linear effects within this operating range can be ignored both during identification and during normal closed-loop control operation. However, it is possible to re-identify the process when there is a change from one operating point to another if the adaptation process is carried out again close to the new operating point and provided that the non-linearity is not passed through during adaptation.
  If certain static non-linearities (e.g., valve characteristic curves) are known, it is always advisable to compensate for them in advance with a polyline in order to linearize the process response.
- **Minimal disturbances in temperature processes**
  Disturbances such as thermal transfer to neighboring zones or heat gains or losses due to changes from one state of matter to another must not influence the overall temperature process to any great extent. If necessary, adaptation at the operating point may be necessary. For example, when zones of an extruder are tuned, all zones must be heated up simultaneously.

- **Quality of the measuring signals**
  The quality of the measured signals has to be sufficient; in other words, the signal-to-disturbance ratio has to be sufficiently high.
- **Process gains not too high**
  The process gains must not be too high. Normalization of the process values is not required. The process gain K can thus include physical units, such as:
  K =ΔPV/ΔLMN, [K]= °C/%
  The final controller design is based on a calculation of the process gain K and can thus in principle compensate for any values of K. However, K is initially unknown during the learning phase and overshoots cannot be avoided when there are extreme combinations of gain and learning step changes. The overshoot is also reduced by reducing the parameter LHLM_TUN in the heating process or by increasing LLLM_TUN in the cooling process.

Requirements for processes with a control valve with integral action

In processes with control valves with an integral action, further requirements must be met in addition to those specified above:

The motor run time of the control valve must be less than the time required to find a point of inflection after a manipulated variable jump (also refer to the process response figure). If this is not the case, the process involved is often a flow rate control system in which only the control valve is effective as the dominating process action. It is not advisable to use the Self-Tuner in this case. You can then set the PI step controller manually in accordance with the following rule of thumb:

GAIN = 1, TI = Motor transition time

See also

Mode of operation TUN_ES (Page 5636)

Input parameters TUN_ES (Page 5640)

Output parameters TUN_ES (Page 5641)

In/out parameters TUN_ES (Page 5644)

Parameter TUN_ES.PHASE (Page 5645)

Parameter TUN_ES.STATUS_H (Page 5647)

Parameter TUN_ES.STATUS_D (Page 5648)

Mode of operation TUN_ES (S7-300, S7-400)

TUN_ES has the following functions for the control or tuning of the controller:

- **Manual mode**

  The Self-Tuner forces the controller into manual mode and transfers its manual parameters to it. In the case of step controllers with position feedback you can specify the manual value in floating point format and manual value signals such as open/close. In the case of step controllers without position feedback you can only specify the manual value signals.

- **Controller tuning**
  During tuning of the controller parameters, the actuating time of the actuator is measured and then the controlled system is identified. A controller is designed based on the characteristics of the controlled system. The new controller parameters are calculated on this basis and activated for the controller.

  – **First adaption**
    Pretuning can only be carried out for a positive setpoint jump. It is usually used during initial commissioning or in case of a serious change in the controlled system characteristics.

  – **Fine tuning**

    The controller can be fine tuned online during a positive setpoint jump when there is a change from one operating point to another or when there are changes in the process response. Fine tuning is only possible for step controllers with position feedback.

- **Tuning the response to setpoint changes**
  The controller is designed for optimum disturbance response, in principle. The resulting "stringent" parameters would result in overshoots of 10% to 20% of the jump height at setpoint jumps.
  The **structure segmentation** represents an initial remedy. In this case, the proportional or derivative action is located in the feedback loop. A change in the setpoint thus only acts on the integral action of the controller. Structure segmentation may only be used for step controllers with position feedback.
  If structure segmentation is not possible for the controller, the following function can be activated in order to tune the response to setpoint changes:

- **Structure switchover**
  Changeover to a quasi P(D) closed-loop control or predictive manipulated variable specification at large positive setpoint jumps. In the case of step controllers without position feedback, it is only possible to change over to a quasi P(D) closed-loop control.

- **Saving the controller parameters**

  If you assess the current controller parameters to be usable, you can save these before a manual change of the controller parameters in parameters specially provided in the instance data block of the Self-Tuner. If you tune the controller, the saved parameters are overwritten by the values that were valid prior to tuning.

- **Reloading saved controller parameters**
  The last controller parameter settings you saved can be activated for the controller again using this function.
- **Changing the controller parameters**

  If you want to set the controller parameters yourself for the controller, you can assign these through special parameters in the instance DB of the Self-Tuner and transfer them to the controller by means of this function.

No cooling tuning and no control zone mode can be implemented with TUN_ES.

Enabling the functions

Function	Start bit	Enable bit	Start condition: PHASE	Start condition: Setpoint jump at SP
Manual mode	—	MAN_ON LMNS_ON	0 to 7	—
Pretuning with measurement of the motor transition time	ADAPT1ST	—	0, 4	Pos.: > MIN_STEP
Pretuning without measurement of the actuating time of the actuator or fine tuning (only possible for step controller without position feedback)	ADAPT_ON	—	4	Pos.: > MIN_STEP
Structure switchover	—	STRUC_ON	4	Pos.: > MIN_STEP Neg: > MIN_STEP
Save controller parameters	SAVE_PAR	—	2 to 7	—
Reload controller parameters	UNDO_PAR	—	4	—
Change controller parameters	LOAD_PAR	—	4	—

Note

Start bits are reset after the function has been started successfully. Enable bits remain set.

See also

Description TUN_ES (Page 5632)

Input parameters TUN_ES (Page 5640)

Output parameters TUN_ES (Page 5641)

In/out parameters TUN_ES (Page 5644)

Parameter TUN_ES.PHASE (Page 5645)

Parameter TUN_ES.STATUS_H (Page 5647)

Parameter TUN_ES.STATUS_D (Page 5648)

Structure switchover (S7-300, S7-400)

Structure segmentation

Structure segmentation is a function of the controller. In the case of closed-loop control with structure segmentation, the proportional or derivative action lies within the feedback loop, so that a change in the setpoint only acts on the integral action of the controller. When setpoint jumps occur, the manipulated variable does not undergo a jump change but instead follows a ramp-shaped course.

The disturbance response is not changed as a result.

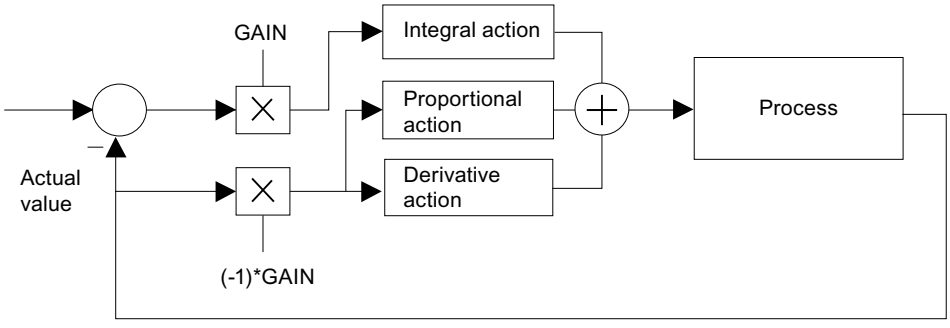

Figure 4-25 Closed-loop control with structure segmentation

In the case of closed-loop control without structure segmentation, the proportional and derivative actions lie within the control deviation, just like the integral action. You can select a structure switchover for this operating mode (STRUC_ON = TRUE).

Closed-loop control with structure segmentation is possible in the following control packages:

- FM355/455
- Standard PID Control
- Modular PID Control

Note

- Structure segmentation must not be used if no position feedback of the control valve is available. This means that you must leave the proportional and derivative actions within the control deviation.
- In closed-loop control with structure segmentation, the dead band in the control deviation is almost without effect.
- With CONT_S and TCONT_S, only closed-loop control without structure segmentation is possible.

Structure switchover when proportional and derivative actions are within the control deviation

The structure switchover is deactivated in the default setting. It should only be selected if structure segmentation is not available. You can activate the structure switchover with STRUC_ON = TRUE. The block automatically selects between two control measures depending on the total loop gain (product of the process gain and the controller gain):

- Integral action temporarily increased, quasi P(D) control:
  In the case of a positive setpoint jump ≥ MIN_STEP, the integral action of the controller is increased temporarily by a significant amount and the gain is decreased slightly, which means that a quasi P(D) controller is used. In the proximity of the setpoint, the system is switched back to the original values for the integral action and the gain (PHASE = 5).
- Temporary predictive manipulated variable specification:
  Processes with long delay times cannot be controlled well with a P(D) control system. After a positive setpoint jump ≥ MIN_STEP, the manipulated variable required constantly for the new setpoint is therefore output constantly. In the proximity of the setpoint, the system is switched back bumplessly to PI or PID closed-loop control. This results in a slow, but overshoot-free response (PHASE = 6).

Note

In the case of step control without position feedback, only the first control measure (PHASE = 5) is possible.

If you do not achieve good results (transient response is slow) with the structure switchover in the case of positive setpoint jumps (for example, during heating processes), you can deactivate the structure switchover with STRUC_ON = FALSE, on the condition that small overshoots may occur.

Controller structure modes	PFDB_SEL, DFDB_SEL [1]	STRUC_ON
Activate structure segmentation	TRUE	FALSE
Activate structure switchover	FALSE	TRUE

[1] Controller parameters PFDB_SEL, DFDB_SEL: With CONT_S, these parameters are not available. This corresponds to the PFDB_SEL = DFDB_SEL = FALSE setting (without structure segmentation).

Note

- Structure segmentation prevents overshooting of the controlled variable when setpoint jumps occur. Select structure segmentation if possible.
- If the controller does not allow structure segmentation, activate structure switchover. The structure switchover must not be activated at the same time as the structure segmentation. You can deactivate structure switchover if rapid tuning is required and small overshoots are permitted at positive setpoint jumps.

Input parameters TUN_ES (S7-300, S7-400)

Table 4-215

Parameter	Address	Data type	Default	Description
SP	0.0	REAL	0.0	Enter the setpoint at the SP input. It is interconnected with the controller via SP_OUT. Note that the setpoint must lie within the setpoint limits of the controller.
PV	4.0	REAL	0.0	PV is interconnected with the process value of the controller.
MAN	8.0	REAL	0.0	Enter the manual value at the MAN input. If MAN_ON = TRUE, the manual value MAN is transferred to the manipulated variable of the controller. When MAN_ON = TRUE and MAN = 0.0, the heating is off. If MAN_ON = FALSE, the manipulated variable is specified by the controller. Values from -100 to 100% are permitted.
LMNR	12.0	REAL	0.0	LMNR is interconnected with the position feedback of the controller (if any). Values from 0.0 to 100% are permitted.
MIN_STEP	16.0	REAL	10.0	You can specify the minimum setpoint jump above which pretuning or adaptation can be carried out in the MIN_STEP input. If structure switchover (STRUC_ON=TRUE) is activated, the setpoint jump must be greater than MIN_STEP in order for the structure switchover activation to take place. Recommendation: MIN_STEP ≥ 10 % of the required setpoint/process value range
LHLM_TUN	20.0	REAL	80.0	At the LHLM_TUN input, specify the manipulated variable jump for the first setting. The manipulated variable jump is calculated automatically during subsequent fine tuning. Values from 0.0 to 100% are permitted.
NORM_FAC	24.0	REAL	1.0	A normalization factor is applied for controllers with a normalized controller gain (Standard PID Control). The normalization factor is calculated on the basis of the range limits of the process value of the controller and is interconnected with the NORM_FAC input.
PULSE_TM	28.0	TIME	0 ms	PULSE_TM is interconnected with the minimum pulse duration of the controller. PULSE_TM ≥ 1 ms
C_LMNUP	32.0	BOOL	FALSE	C_LMNUP is interconnected with the manipulated variable signal QLMNUP of the controller.
C_LMNDN	32.1	BOOL	FALSE	C_LMNDN is interconnected with the manipulated variable signal QLMNDN of the controller.
MAN_ON	32.2	BOOL	FALSE	If MAN_ON = TRUE, the manipulated variable of the controller is specified via the MAN input. If MAN_ON = FALSE, the manipulated variable is specified by the controller.
LMNR_HS	32.3	BOOL	FALSE	LMNR_HS is interconnected with the endstop signal LMNR_HR of the controller.
LMNR_ON	32.4	BOOL	FALSE	If LMNR_ON = TRUE, you can use LMNS_ON = TRUE or MAN_ON = TRUE to switch a step controller with position feedback over to manual mode. If LMNR_ON = FALSE, you can use LMNS_ON = TRUE to switch a step controller without position feedback over to manual mode.
LMNS_ON	32.5	BOOL	TRUE	If LMNS_ON is set, you can use the LMNUP and LMNDN parameters to manipulate the actuating signals of the controller.

Parameter	Address	Data type	Default	Description
LMNUP	32.6	BOOL	FALSE	You can use LMNUP to manipulate the actuating signal OPEN of the controller.
LMNDN	32.7	BOOL	FALSE	You can use LMNDN to manipulate the actuating signal CLOSE of the controller.
PID_ON	33.0	BOOL	FALSE	You can specify at the PID_ON input whether the tuned controller is to operate as a PI controller or a PID controller. PID controller: PID_ON = TRUE PI controller: PID_ON = FALSE With certain controlled system types, however, it is possible that only a PI controller will be designed, despite PID_ON = TRUE.
STRUC_ON	33.1	BOOL	FALSE	If STRUC_ON = TRUE, a structure switchover (from PI(D) to P(D) control) ensures that overshoots are largely avoided at setpoint jumps. The STRUC_ON input must not be set = TRUE if the proportional action or derivative action is fed into the feedback loop of the controller.
WRITE_DIS	33.2	BOOL	FALSE	In connection with the controller module, the Self-Tuner must know whether the driver block is ready to send data to the FM. This is the case if LOAD_OP = LOAD_PAR = FALSE. The result of the OR operation of LOAD_OP and LOAD_PAR is interconnected with WRITE_DIS.
CYCLE	34.0	TIME	100 ms	The sampling time must be configured in the CYCLE input. CYCLE must correspond to the sampling time of the controller to be set and the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios). ≥ 1 ms

See also

Description TUN_ES (Page 5632)
Mode of operation TUN_ES (Page 5636)
Output parameters TUN_ES (Page 5641)
In/out parameters TUN_ES (Page 5644)
Parameter TUN_ES.PHASE (Page 5645)
Parameter TUN_ES.STATUS_H (Page 5647)
Parameter TUN_ES.STATUS_D (Page 5648)

Output parameters TUN_ES (S7-300, S7-400)

Table 4-216

Parameter	Address	Data type	Default	Description
MAN_OUT	38.0	REAL	0.0	MAN_OUT is connected to the manual input value of the controller. If MAN_ON = TRUE, the Self-Tuner outputs the value specified at the MAN input to the MAN_OUT output.
SP_OUT	42.0	REAL	0.0	SP_OUT is connected to the setpoint input of the controller. The Self-Tuner writes the setpoint specified at the SP input to the SP_OUT output. If manual mode MAN_ON = TRUE, the setpoint is tracked to the process value: SP_OUT = PV.

Parameter	Address	Data type	Default	Description
GAIN	46.0	REAL	1.5	GAIN is interconnected with the proportional gain (controller gain) of the controller.
TI	50.0	TIME	1 h	TI is interconnected with the integral action time (reset time) of the controller.
TD	54.0	TIME	0 ms	TD is interconnected with the derivative action time (rate time) of the controller.
TM_LAG	58.0	TIME	1 s	TM_LAG is interconnected with the time lag of the controller differentiator.
MTR_TM	62.0	TIME	30 s	MTR_TM is interconnected with the motor run time of the controller.
DEADB_W	66.0	REAL	0.0	DEADB_W is interconnected with the dead band width of the controller.
PHASE	70.0	INT	0	Parameter TUN_ES.PHASE (Page 5645) displays the current sequence of the controller tuning. During the tuning of Standard PID Control, the PHASE output is interconnected with the PHASE input of the PID_ES instruction.
STATUS_H	72.0	INT	0	Parameter TUN_ES.STATUS_H (Page 5647) displays tuning results regarding the search for the point of inflection during the heating process.
STATUS_D	74.0	INT	0	Parameter TUN_ES.STATUS_D (Page 5648) displays tuning results regarding the controller design during the heating process.
QMAN_ON	76.0	BOOL	FALSE	QMAN_ON is interconnected with the manual/automatic changeover of the controller.
QLMNS_ON	76.1	BOOL		QLMNS_ON is interconnected with the "Enable manipulated signals" parameter of the controller
QLMNUP	76.2	BOOL		QLMNUP is interconnected with the "Manipulated signal open" parameter of the controller
QLMNDN	76.3	BOOL		QLMNDN is interconnected with the "Manipulated signal close" parameter of the controller
QI_SEL	76.4	BOOL	TRUE	QI_SEL is interconnected with the on/off switch of the controller integral action I_SEL. In the case of FM355/455, TI must be switched to zero depending on QI_SEL: If QI_SEL = FALSE, then TI = 0.
QD_SEL	76.5	BOOL	FALSE	QD_SEL is interconnected with the on/off switch of the controller derivative action D_SEL. In the case of FM355/455, TD must be switched to zero depending on QD_SEL: If QD_SEL = FALSE, then TD = 0.
QWRITE	76.6	BOOL	FALSE	QWRITE is interconnected with the LOAD_PAR and LOAD_OP inputs of the PID_FM instruction of FM355/455. As a result, the data records of the controller channel are downloaded to the FM.
PROCESS	78.0	STRUCT		The PROCESS structure contains the parameters of the process.
GAIN	+0.0	REAL	999.0	The process gain is displayed in the PROCESS.GAIN output.
TM_LAG1	+4.0	REAL	0.0	The slow process time constant is displayed at the PROCESS.TM_LAG1 output.
TM_LAG2	+8.0	REAL	0.0	The fast process time constant is displayed at the PROCESS.TM_LAG2 output.
PV00	+12.0	REAL	0.0	The zero mean value of the process is displayed at the PROCESS. PV00 output. The zero mean value is the steady-state process value when manipulated variable = zero. This corresponds to the ambient temperature when the system is cold.
KIG	+16.0	REAL	0.0	The maximum rise ratio of the process value during heating is displayed in the PROCESS.KIG output.
MTR_TM	+20.0	REAL	0.0	The motor run time is displayed at the MTR_TM output.

Parameter	Address	Data type	Default	Description
PI_CON	102.0	STRUCT		The PI_CON structure contains the parameters of the PI controller.
GAIN	+0.0	REAL	1.5	PI_CON.GAIN displays the proportional gain of the PI controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TI	+4.0	REAL	3600.0	PI_CON.TI displays the integral action time of the PI controller. You can improve the value manually and make it effective by setting LOAD_PAR.
PID_CON	110.0	STRUCT		The PID_CON structure contains the parameters of the PID controller.
GAIN	+0.0	REAL	0.0	PID_CON.GAIN displays the proportional gain of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TI	+4.0	REAL	0.0	PID_CON.TI displays the integral action time of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TD	+8.0	REAL	0.0	PID_CON.TD displays the derivative action time of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
PI_CON_OLD	122.0	STRUCT		The PI_CON_OLD structure contains the old parameters of the PI controller.
GAIN	+0.0	REAL	1.5	The old proportional gain of the PI controller is displayed at the PI_CON_OLD.GAIN output.
TI	+4.0	REAL	3600.0	The old integral action time of the PI controller is displayed at the PI_CON_OLD.TI output.
PID_CON_OLD	130.0	STRUCT		The PID_CON_OLD structure contains the old parameters of the PID controller.
GAIN	+0.0	REAL	1.5	The old proportional gain of the PID controller is displayed at the PID_CON_OLD.GAIN output.
TI	+4.0	REAL	3600.0	The old integral action time of the PID controller is displayed at the PID_CON_OLD.TI output.
TD	+8.0	REAL	0.0	The old derivative action time of the PID controller is displayed at the PID_CON_OLD.TD output.
TU	142.0	REAL	0.0	The determined delay time is displayed at the TU output.
TA	146.0	REAL	0.0	The determined recovery time is displayed at the TA output.

See also

Description TUN_ES (Page 5632)

Mode of operation TUN_ES (Page 5636)

Input parameters TUN_ES (Page 5640)

In/out parameters TUN_ES (Page 5644)

In/out parameters TUN_ES (S7-300, S7-400)

Table 4-217

Parameters	Address	Data type	Default	Description
ADAPT1ST	150.0	BOOL	FALSE	When you carry out the first optimization of the process or do not wish to use the results of previous optimizations such as manipulated value step size or filter time constants, you must carry out a first adaption. To do this, set ADAPT1ST = TRUE. The Self-Tuner then carries out an initialization routine and, in the process, sets the default values described in the initialization routine. The Self-Tuner sets ADAPT1ST back to FALSE. It does not make sense to set ADAPT1ST to TRUE at a restart of the CPU in OB100.
ADAPT_ON	150.1	BOOL	FALSE	If you want to trigger on-line adaption, set ADAPT_ON = TRUE. If the subsequent setpoint step-change is greater than MIN_STEP, identification is started (PHASE = 2). The Self-Tuner sets ADAPT_ON back to FALSE.
STEADY	150.2	BOOL	FALSE	When the process value has reached a steady state, you can use STEADY to terminate optimization prematurely, even though the Self-Tuner is still in PHASE 3. After the Self-Tuner has passed into PHASE 4, you can also use STEADY to carry out a renewed controller design in steady state.
UNDO_PAR	150.3	BOOL	FALSE	The old controller parameters are transferred to the current controller parameters and are active immediately depending on PID_ON. PI_CON = PI_CON_OLD PID_CON = PID_CON_OLD The function for undoing changes can only be used in automatic mode (PHASE 4).
SAVE_PAR	150.4	BOOL	FALSE	The current controller parameters are transferred to the old controller parameters. PI_CON_OLD = PI_CON PID_CON_OLD = PID_CON
LOAD_PAR	150.5	BOOL	FALSE	The changed parameters PI_CON und PID_CON are transferred to the controller and are thus active.

See also

Description TUN_ES (Page 5632)
Mode of operation TUN_ES (Page 5636)
Input parameters TUN_ES (Page 5640)
Output parameters TUN_ES (Page 5641)
Parameter TUN_ES.PHASE (Page 5645)
Parameter TUN_ES.STATUS_H (Page 5647)
Parameter TUN_ES.STATUS_D (Page 5648)

Parameter TUN_ES.PHASE (S7-300, S7-400)

PHASE	Mode	Description
0	Automatic mode: Close-loop control with preassigned parameters	During the generation of an instance DB for the Self-Tuner, thePHASE parameter is pre-assigned with zero. Since LMNS_ON is pre-assigned with TRUE, the Self-Tuner will switch to PHASE 7 (manual mode) after the first run. You can start the pretuning directly from manual mode (ADAPT_ON = TRUE) or switch to automatic mode. When you switch to automatic mode MAN_ON = FALSE for the first time, the parameters of PI_CON are applied. The default values for GAIN and TI are: GAIN = 1.5, TI = 1h. Exiting from the phase via MAN_ON, LMNS_ON, or ADAPT1ST
1	Controller tuning: Heating off and waiting for a setpoint jump	After tuning has been started with ADAPT1ST = TRUE, the Self-Tuner closes the control valve (manipulated variable = 0 or LMNS_ON = LMNDN = TRUE) and waits for a positive setpoint jump ≥ MIN_STEP.
2	Controller tuning: Motor transition time measurement, moving to a constant control valve setting and searching for point of inflection	As soon as you specify a setpoint jump ≥ MIN_STEP in the positive direction to the operating point of the warm process state, with pretuning MAN_OUT is set to LHLM_TUN and with fine tuning MAN_OUT is set to a value calculated by the self-tuner. MIN_STEP should be greater than 10% of the operating range of the setpoint and the process value. In the case of step control with position feedback (LMNR_ON = TRUE), the value TRUE is assigned to QMAN_ON. The controller controls the valve to the value of MAN_OUT, and TUN_ES calculates the motor transition time MTR_TM. In the case of step control without position feedback, QLMNS_ON and QLMNUP are set to TRUE, and the valve is moved to the high endstop position. When the high endstop is reached (LMNR_HS =TRUE), TUN_ES calculates the motor transition time and forwards it to the controller. The valve is then moved toward the closed position until the value of MAN_OUT is reached (QLMNDN = TRUE).
3	Controller tuning: Closed-loop control with a PI coarse controller	When the point of inflection of the jump response is recognized (STATUS_H = 2) or when the process value has reached 70% of the setpoint jump height (STATUS_H = 3), a PI controller with conservative settings is designed. The controller gain GAIN is limited to the range from 0.2 to 20.0. As of this point, the controller immediately starts working as a PI controller and attempts to bring the controlled system to a steady-state condition. The Self-Tuner stores the newly determined PI controller parameters in GAIN, TI, and PI_CON. The previous values of the PI controller are saved in PI_CON_OLD. If it takes a long time to reach steady-state condition (creeping transient response during temperature processes), you can set STEADY = TRUE to initiate controller design with the current data when steady-state condition is almost reached. You can also set STEADY = TRUE to re-initiate the controller design with the current data at a later stage in Phase 3 or 4. This will often provide some additional improvement to the controller design. If an overshoot occurs or if no point of inflection is found, the manipulated variable jump size LHLM_TUN selected for pretuning may have been too large. This does not necessarily result in poor controller tuning. For the next pretuning, the value you select for LHLM_TUN should be approximately 20% less. If the block has recognized steady-state condition or if the period 30 × TI (TI: integral action time of the PI controller set in PHASE = 3) has elapsed since the setpoint jump, the controller design has been improved and the system switches to PHASE = 4. The Self-Tuner stores the newly determined PI/PID controller parameters in GAIN, TI, TD, and PI_CON or PID_CON. The previous values of the PID controller are saved in PID_CON_OLD. If possible, a PID controller and a PI controller are designed. If PID_ON = TRUE, the PID controller is activated, otherwise the PI controller is activated. In PHASE 4, it is possible to switch between the two controllers bumplessly at any time. For difficult processes, the block always designs a PI controller, irrespective of PID_ON. The controller gain GAIN is limited in such a manner that the loop gain of the open loop (product of the process gain and the controller gain) is less than 40.0.

PHASE	Mode	Description
4	Automatic mode: Normal closed-loop control with/without control zone	In this phase, the controller controls using its tuned parameters. From this state, you can start fine tuning or pretuning.
5	Automatic mode, structure switchover at setpoint jump: Integral action deactivated, P(D) control	In the case of a positive setpoint jump ≥ MIN_STEP, the integral action of the controller is increased temporarily by a significant amount and the gain is decreased slightly, so that a quasi P(D) controller is used. In the proximity of the setpoint, the system is switched back to the original values for the integral action and the gain (PHASE = 5 in the Self-Tuner sequence).
6	Automatic mode, structure switchover at setpoint jump: Predictive manipulated variable specification	Only for step controller with position feedback Processes with long delay times cannot be controlled well with a P(D) control system. After a positive setpoint jump ≥ MIN_STEP, the manipulated variable required constantly for the new setpoint is therefore output continually. In the proximity of the setpoint, the system is switched back bumplessly to PI or PID closed-loop control. This results in a slow, but overshoot-free response
7	Manual mode	Manual mode takes priority over all other operating modes. Pretuning or fine tuning is canceled. When manual mode is deactivated (MAN_ON = FALSE and LMNS_ON = FALSE), the controller changes over to automatic mode (PHASE = 4 in the Self-Tuner sequence) and controls using the existing controller parameters. If you use a step controller with position feedback (LMNR_ON = TRUE), you can use LMNS_ON = TRUE or MAN_ON = TRUE to switch to manual mode. If you use a step controller without position feedback (LMNR_ON = FALSE), you can only use LMNS_ON = TRUE to switch to manual mode. If you set the MAN_ON input to TRUE, the QMAN_ON output is set to TRUE and the MAN_OUT output is set to MAN. If you set the LMNS_ON input to TRUE, the QLMNUP output is set to LMNUP and the QLMNDN output is set to LMNDN.

Heating tuning with step controller without position feedback

The diagram shows the phase steps of pretuning with measurement of the motor run time.

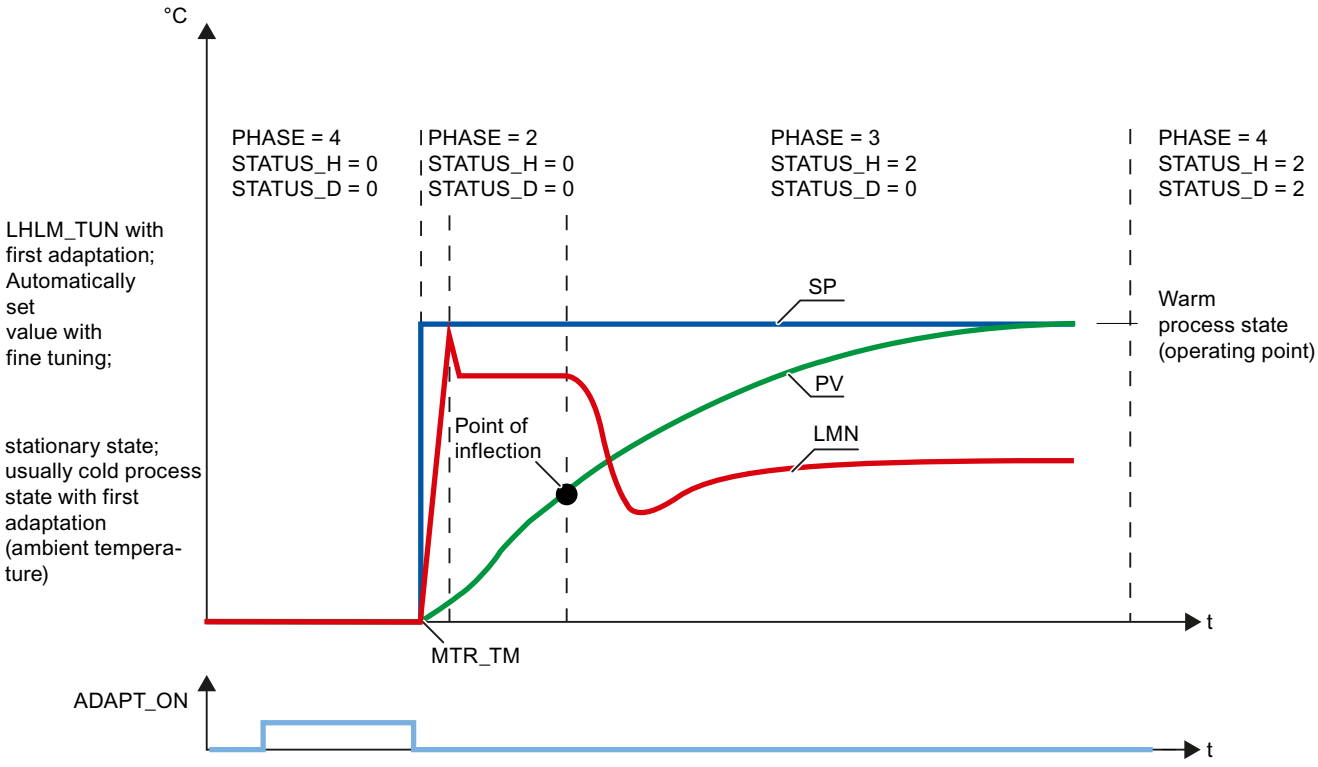

See also

Description TUN_ES (Page 5632)

Mode of operation TUN_ES (Page 5636)

Input parameters TUN_ES (Page 5640)

In/out parameters TUN_ES (Page 5644)

Parameter TUN_ES.STATUS_H (Page 5647)

Parameter TUN_ES.STATUS_D (Page 5648)

Parameter TUN_ES.STATUS_H (S7-300, S7-400)

STATUS_H	Description	Remedy
0	Initialization	
1	Point of inflection too low In most cases optimization still results in good controller parameters.	To get closer the ideal case, increase LHLM_TUN or decrease the setpoint step-change size
2	Ideal case: Point of inflection found	

STATUS_H	Description	Remedy
3	Point of inflection too high A safety factor is included. In most cases optimization leads to controller parameters that can still be used but are prudent.	To get closer the ideal case, decrease LHLM_TUN or increase the setpoint step-change size.
11, 12	The time lag could not be detected correctly. Operation continues with an estimated value that can lead to controller parameters that are not optimum. If the process has a pure VZ1 behavior (no time lag), the Self-Tuner can often design good controller parameters that are still good. Since calculation is continued with a roughly estimated time lag, it is possible that STATUS_D cannot be determined to 1 (controller for approximating VZ1 behavior) and that the control zone is de-activated, although a control zone makes sense for VZ1 processes.	Repeat optimization and ensure that disturbances do not occur at the process value. If you are not satisfied with the controller parameters, you can at least use the values from PROCESS.GAIN and PROCESS.TM_LAG1 for a VZ1 model and use your own controller design in accordance with the text book.
13	The time lag could not be detected correctly. Operation continues with an estimated value that can lead to controller parameters that are not optimum.	Repeat optimization and ensure that disturbances do not occur at the process value.

See also

Description TUN_ES (Page 5632)
Mode of operation TUN_ES (Page 5636)
Input parameters TUN_ES (Page 5640)
In/out parameters TUN_ES (Page 5644)
Parameter TUN_ES.PHASE (Page 5645)
Parameter TUN_ES.STATUS_D (Page 5648)

Parameter TUN_ES.STATUS_D (S7-300, S7-400)

STATUS_D	Description
0	Initialization
1	Only PI controller design, since the process almost has PT1 behavior
2	Ideal case: PI and PID controllers possible, VZ2 behavior
3	Only PI controller design, since there is a very high time lag / dead time or high order
4	Incorrect estimate No controller design for on-line adaption. During the first adaption the process is continued with the coarse PI controller from PHASE = 3. Repeat optimization and ensure that disturbances do not occur at the process value.

See also

Description TUN_ES (Page 5632)
Mode of operation TUN_ES (Page 5636)
Input parameters TUN_ES (Page 5640)
In/out parameters TUN_ES (Page 5644)
Parameter TUN_ES.PHASE (Page 5645)
Parameter TUN_ES.STATUS_H (Page 5647)

Standard PID Control (S7-300, S7-400)

Technical specifications of the instructions (S7-300, S7-400)

CPU utilization

The controlling efficiency and therefore processing speed of actual control loops depends exclusively on the performance of the CPU you are using. To estimate the utilization of a particular CPU due to installation of Standard PID Control, the following information applies:

- The user memory only has to contain a single instance of the instruction, regardless of the number of controllers.
- You will need a DB with approximately 0.5 Kbytes for each controller.
- There is no memory requirement for an L-Stack.
- Alarms are not delayed by PID_CP or PID_ES.
- Key data for typical execution times (process times) of blocks when default parameters for control mode are used:

Instruction	Boundary conditions	Processing time (in ms) CPU 315–2AG10	Processing time (in ms) CPU 416–2XK02
PID_CP	Typical boundary conditions	1.3	0.14
PID_ES	Without position feedback, typical boundary conditions	1.5	0.16

Work memory assignment

Refer to the following tables for the size of the required area in the user memory and, thus, the number of control loops that could theoretically be installed based on the available memory capacity:

Instruction	Load memory requirement	Work memory requirement	Local data
PID_CP	8956 bytes	7796 bytes	122 bytes
PID_ES	9104 bytes	7982 bytes	152 bytes
LP_SCHED	1064 bytes	976 bytes	20 bytes

Instance DB or global DB	Load memory requirement	Work memory requirement:
DB for PID_CP	1168 bytes	510 bytes
DB for PID_ES	1124 bytes	484 bytes
DB_LOOP (for 5 control loops)	184 bytes	100 bytes
DB_RMPSK (with starting point and 4 interpolation points)	142 bytes	78 bytes

Sampling time

The minimum selectable sampling time is dependent on the performance class of the CPU used.

Note

The limited computational accuracy limits the achievable sampling time. As the sampling time becomes shorter, the constants in the algorithms take on smaller and smaller values. This can result in incorrect calculation of the manipulated variable.

Recommendation:

- S7-300: Sampling time ≥ 20 ms
- S7-400: Sampling time ≥ 5 ms

PID_CP (S7-300, S7-400)

Description PID_CP (S7-300, S7-400)

In addition to the functions in the setpoint and process value branches, the instruction implements a complete PID-controller with continuous manipulated variable output and the option of manually influencing the manipulated variable. Subfunctions can be enabled or disabled.

With this instruction you are able to control technical processes and systems with continuous input and output variables on the SIMATIC S7 automation systems. Slow (temperatures, filling levels etc.) as well as fast controlling systems (flow, speed etc.) can be controlled.

The controller can be used individually as a fixed setpoint controller, or in multi-loop control systems as a cascade, blending, or ratio controller.

Startup

The PID_CP instruction features an initialization routine that is executed when input parameter COM_RST = TRUE is set.

The integral action is set to the initialization value I_ITLVAL at startup and read out at output LMN_I. When it is called in a cyclic interrupt level, it then continues to work starting at this value.

When ramp/soak is enabled, the time intervals DB_NBR PI[0 ... NBR_PTS].TMV between the time slices are added up and displayed at the total time T_TM and total remaining time RT_TM outputs. In the case of online changes of PI[n].TMV or specification of TM_CONT and TM_SNBR, the total time and the total remaining time of the ramp/soak profile change. Since the calculation of T_TM and RS_TM with many time slices significantly increases the processing time of the RMP_SOAK function, the calculation is only carried out at a restart or when TUPDT_ON = TRUE.

All other outputs are set to their default values.

Hot restart

With a hot restart, the operating state that existed at the time of the interruption is assumed. The control continues to operate with the values that were calculated at the moment the interruption occurred.

Call

The instruction PID_CP must be called equidistant. To achieve this, use a cyclic interrupt level (for example, OB35).

If you call the instruction PID_CP as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_CP directly in the multiple instance DB and commission it via a watch table.

Generating control deviation PID_CP (S7-300, S7-400)

The following functions are always active and cannot be disabled.

- Setpoint limiting SP_LIMIT,
- Process value monitoring for absolute values PV_ALARM,
- Process value monitoring for size of rate of change ROCALARM,
- Control deviation monitoring for absolute values ER_ALARM,
- Limiting of the output signal LMNLIMIT.

The default values are selected in such a manner that operation requires no individual adjustment. To achieve good controller results, you always have to adapt the parameters of these functions to your process.

Processing the process value

The process value is scaled to the physical value.

AUTOHOTSPOT

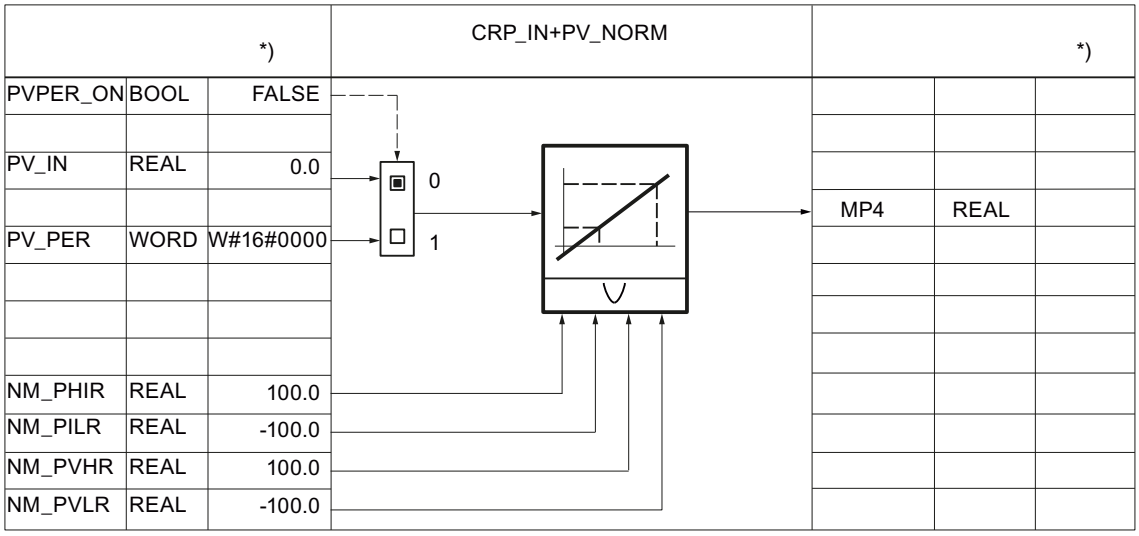

*) Default for creation of new technology object.

The process value can be smoothed with a first-order delay element.

AUTOHOTSPOT

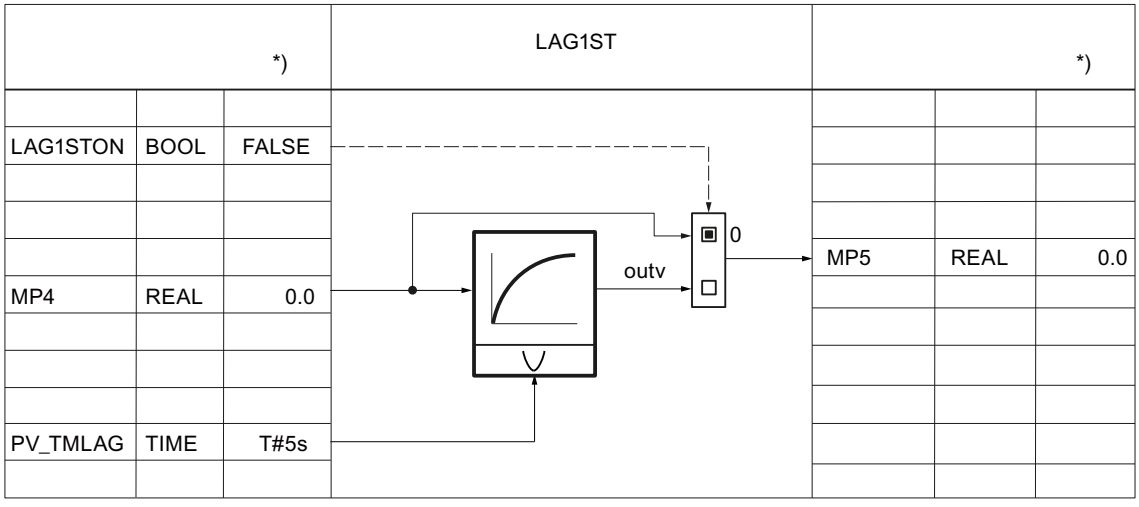

The process value can be linearized with the square root. The following parameters are required for this:

AUTOHOTSPOT

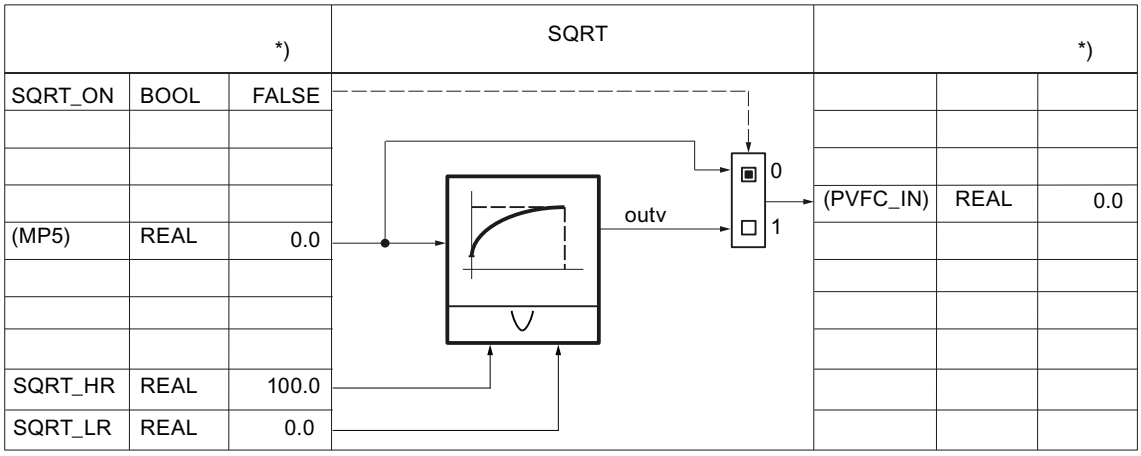

The process value can be prepared with a user-specific function. The following parameters are required for this: The interconnection must be programmed in the user FC.

AUTOHOTSPOT

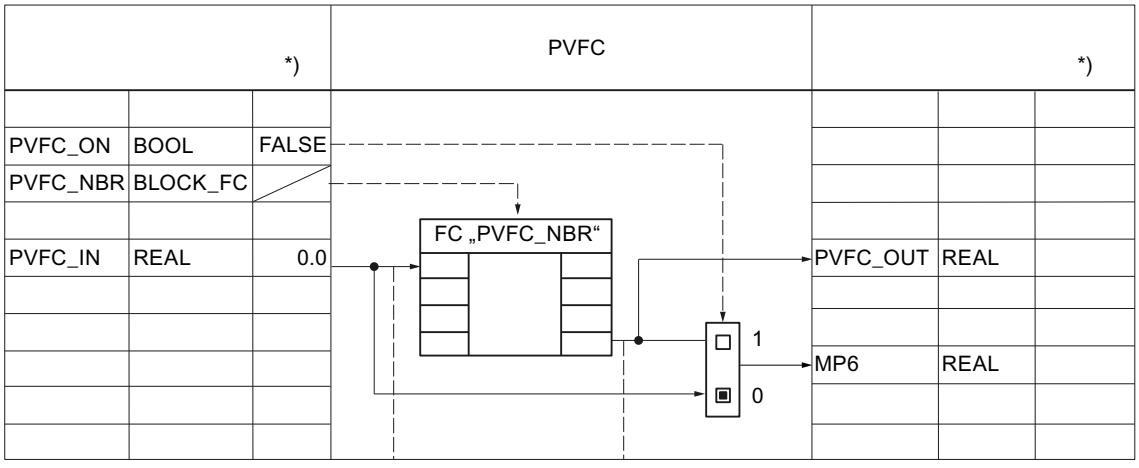

The process value is always monitored with the following parameters for warning and alarm limits.

AUTOHOTSPOT

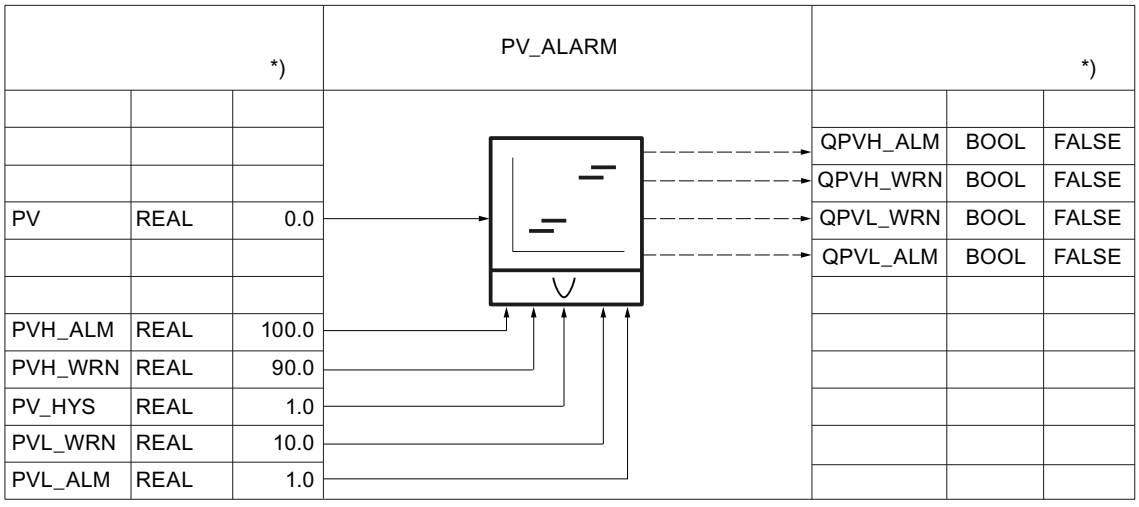

The rate of change of the process value is always limited with the following parameters.

AUTOHOTSPOT

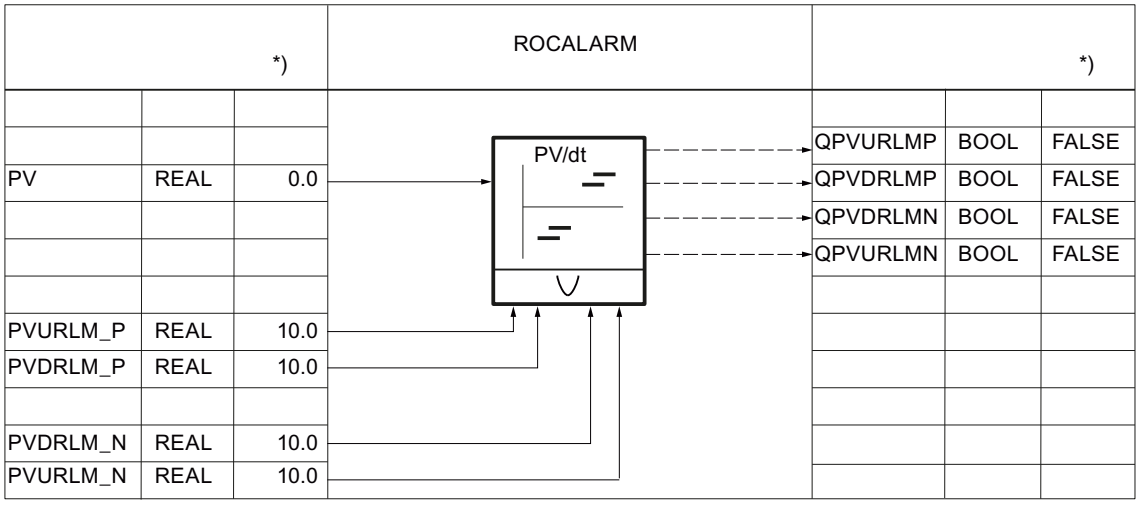

Processing setpoints

The internal setpoint SP_IN can be incremented or decremented with the following parameters.

AUTOHOTSPOT

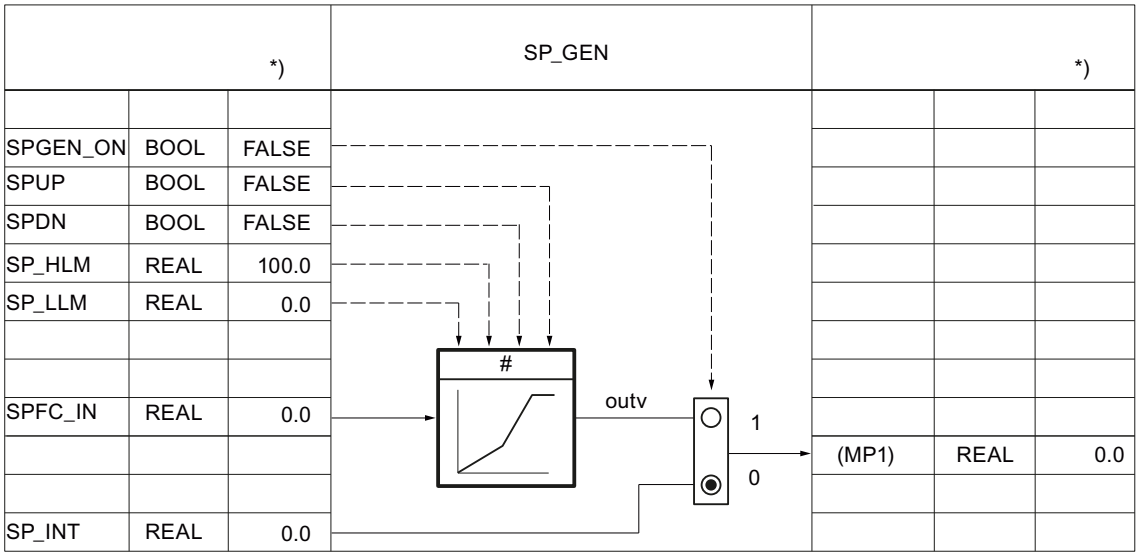

The setpoint can be specified with the following parameters via a ramp/soak function.

AUTOHOTSPOT

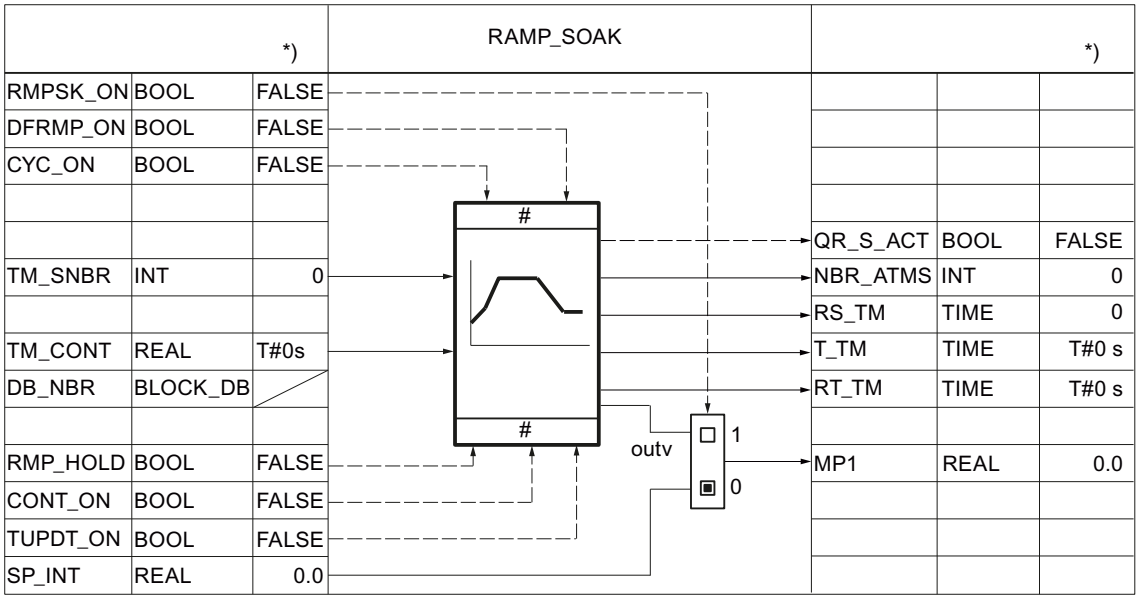

The blending/ratio controller can scale the external setpoint SP_EXT using the following parameters.

AUTOHOTSPOT

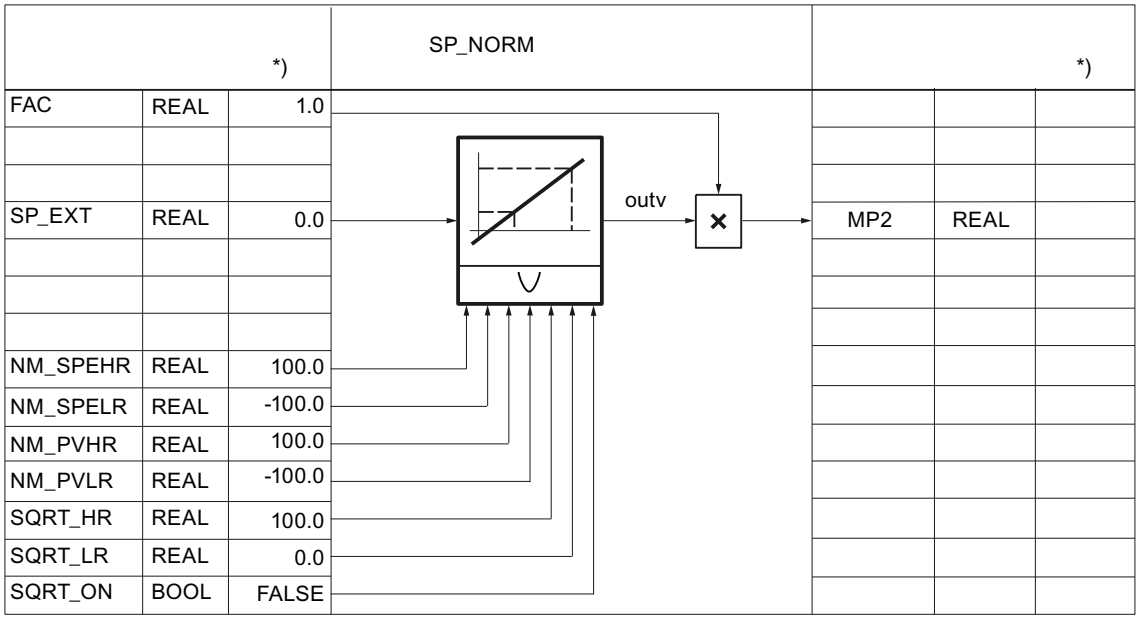

The setpoint can be further processed with a user-specific function. The following parameters are required for this: The interconnection must be programmed in the user FC.

The input value SPFC_IN is an implicit parameter. This can be monitored with the configuration tool, either via the measuring point MP1 (SPEXT_ON = FALSE) or via the measuring point MP2 (SPEXT_ON = TRUE). The output value is accessible at the measuring point MP3. The input SPFC_IN is then switched through on the setpoint branch, if SPFC_ON =FALSE.

AUTOHOTSPOT

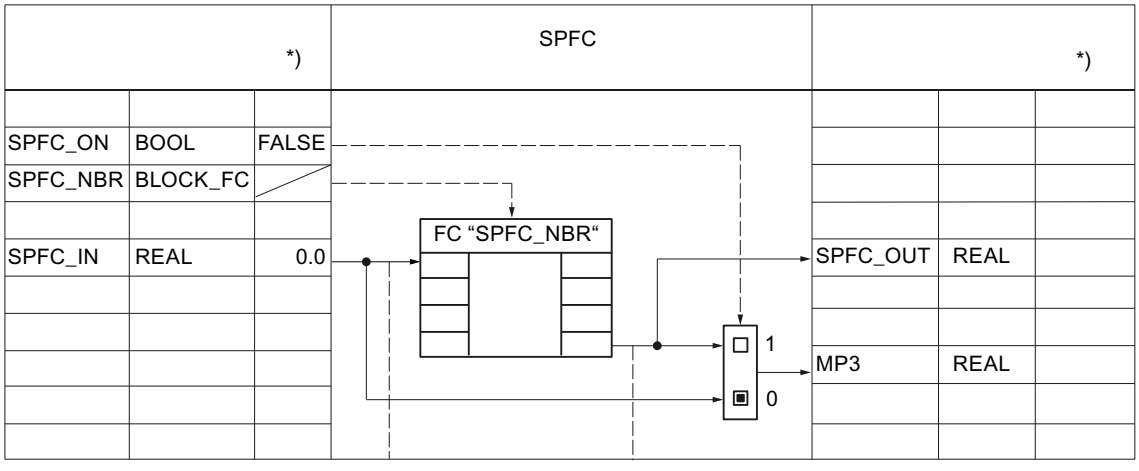

The rate of change of the setpoint can be limited with the following parameters.

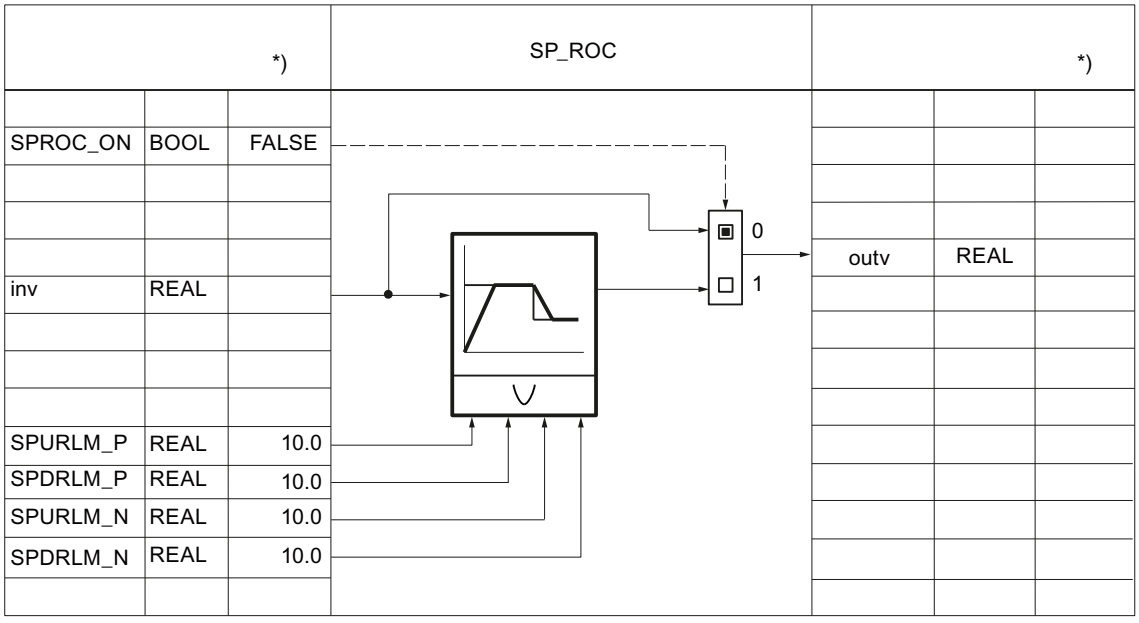

The setpoint value is always monitored with the following parameters for an absolute high and low limit.

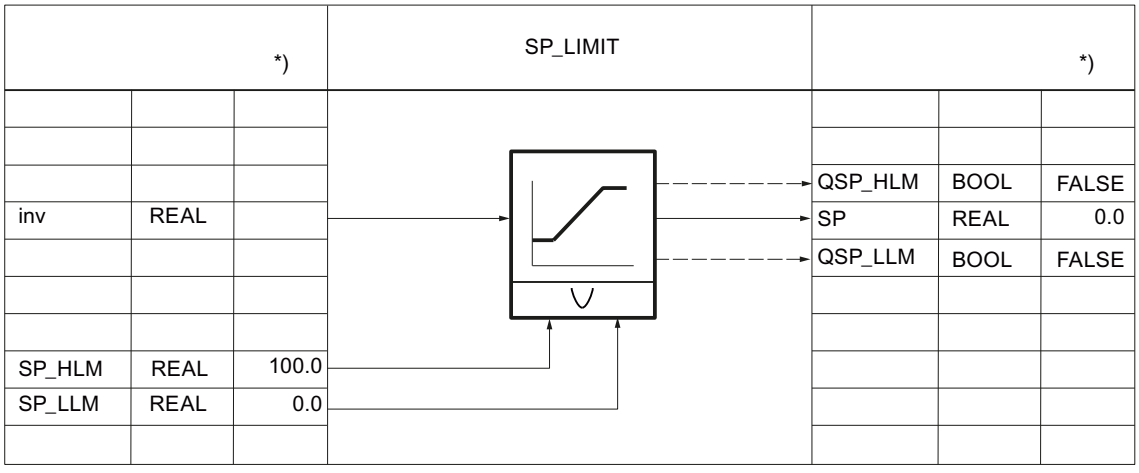

Generating control deviation

The following parameters can be used to route the control deviation over a deadband.

AUTOHOTSPOT

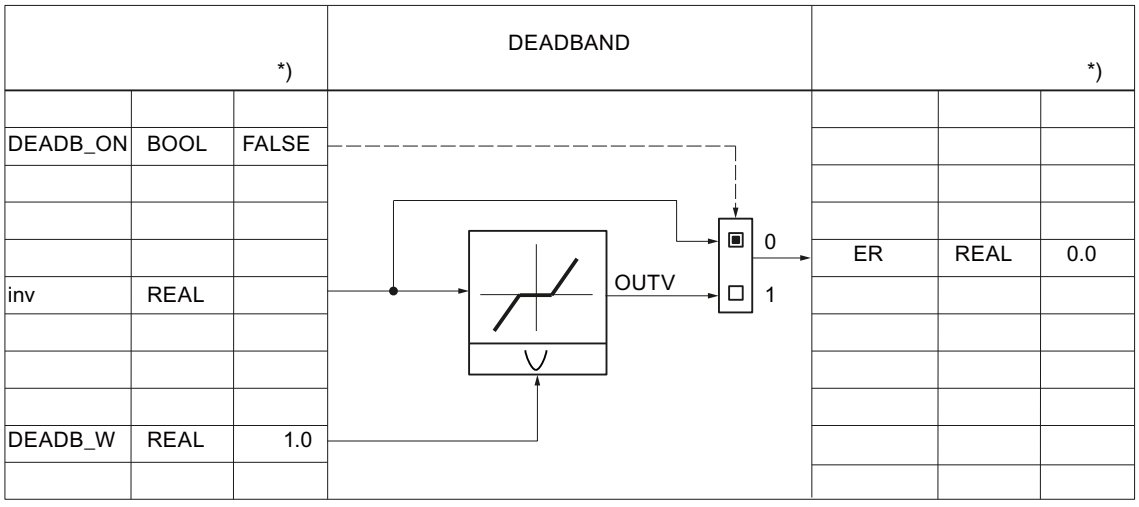

The control deviation is always monitored with the following parameters for warning and alarm limits.

AUTOHOTSPOT

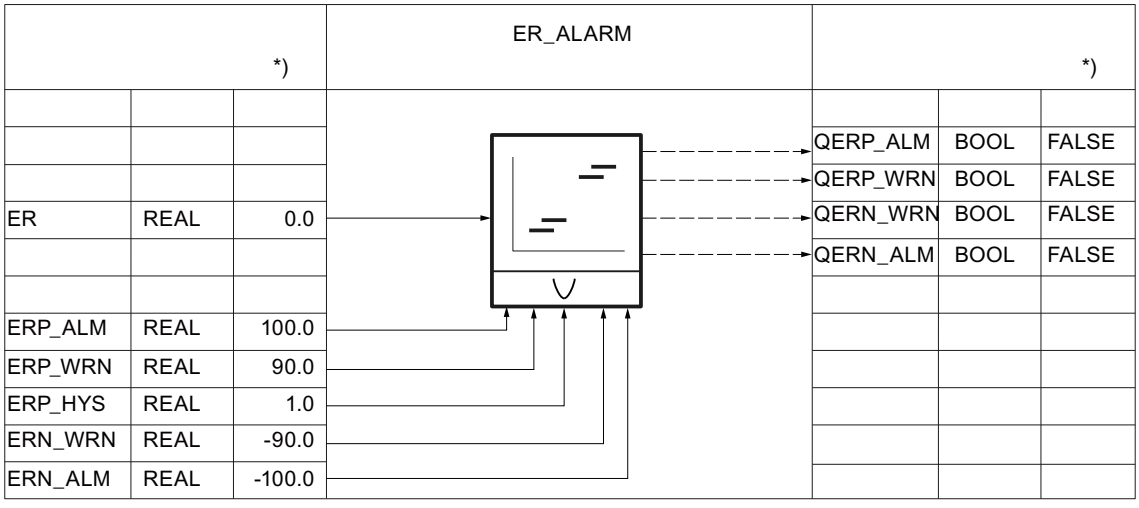

Control algorithm PID_CIP (S7-300, S7-400)

Control algorithm

AUTOHOTSPOT

A total of five switches are available to specify an effective controller structure.

AUTOHOTSPOT

Controller structure	Switch				
	P_SEL	I_SEL	D_SEL	PFDB_SEL	DFDB_SEL
P control	TRUE	FALSE	FALSE	FALSE	FALSE
P control (P in feedback path)	TRUE	FALSE	FALSE	TRUE	FALSE
PI control	TRUE	TRUE	FALSE	FALSE	FALSE
PI control (P in feedback path)	TRUE	TRUE	FALSE	TRUE	FALSE
PD control	TRUE	FALSE	TRUE	FALSE	FALSE
PD control (P in feedback path)	TRUE	FALSE	TRUE	FALSE	TRUE
PID control	TRUE	TRUE	TRUE	FALSE	FALSE
PID control (P/D in feedback path)	TRUE	TRUE	TRUE	FALSE	TRUE

The proportional action is determined by the proportional gain GAIN and can be monitored at LMN_P.

AUTOHOTSPOT

The integral action is calculated using the following parameters.

AUTOHOTSPOT

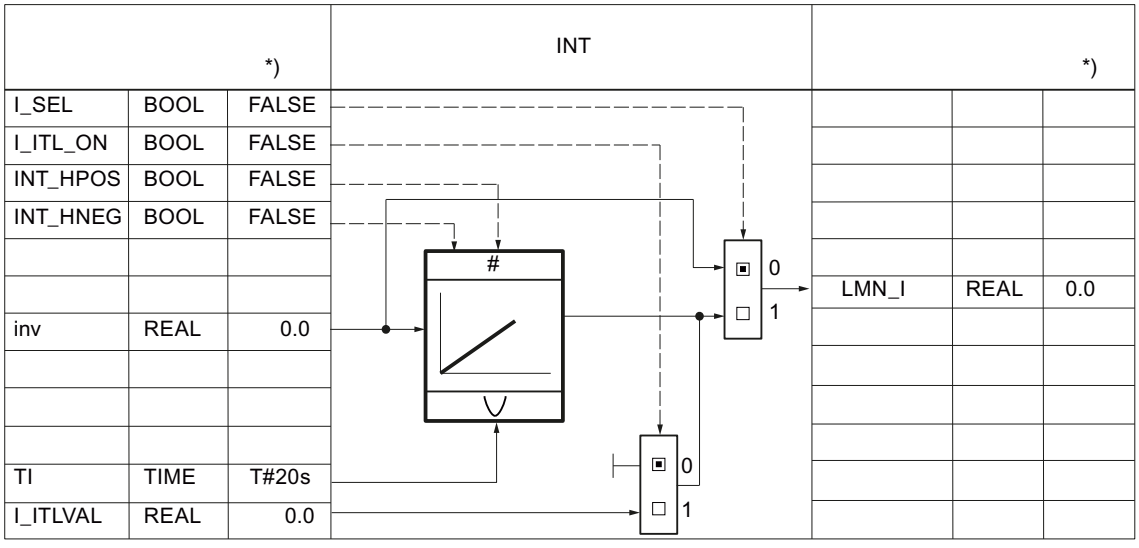

*) Default for creation of new technology object

The derivative action is calculated using the following parameters.

AUTOHOTSPOT

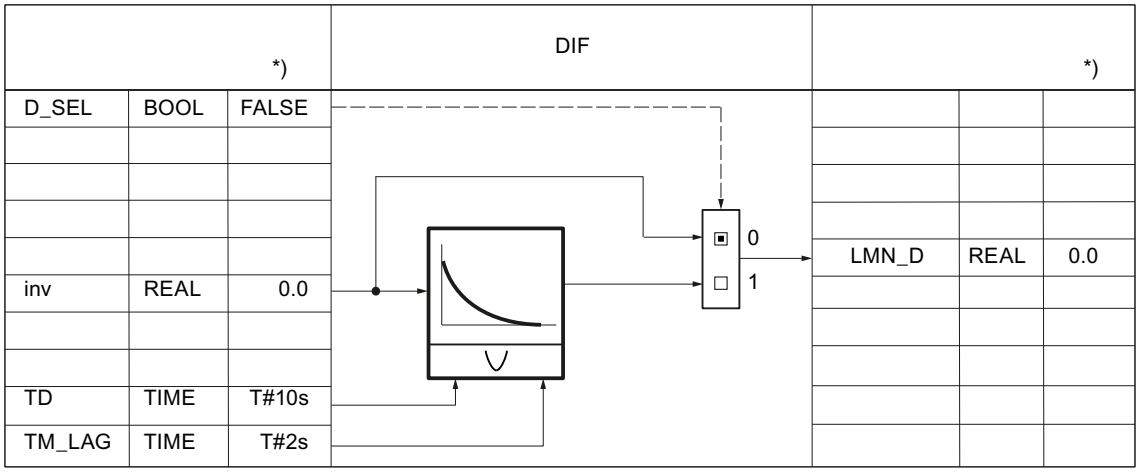

Manipulated variable PID_CP (S7-300, S7-400)

Continuous manipulated variable

The PID control algorithm generates an analog manipulated variable, supplemented if necessary by a pulse generator to generate pulse width modulated output signals for the two-step control or three-step controls with proportional actuators.

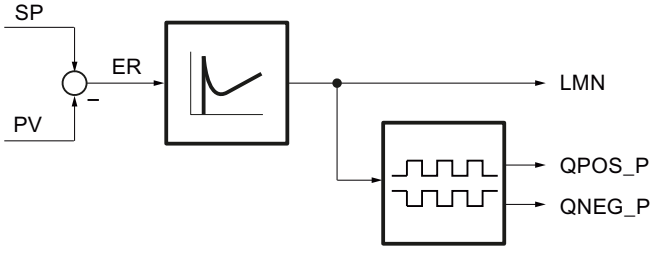

The manipulated variable can be manually influenced using the following parameters.

AUTOHOTSPOT

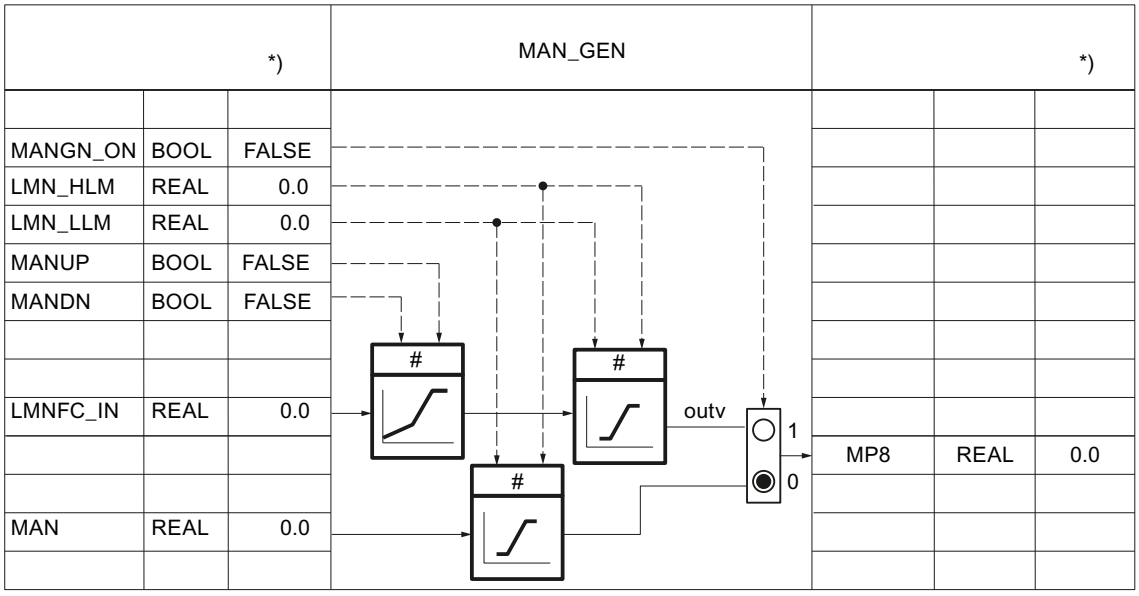

The manipulated variable can be further processed with a user-specific function. The following parameters are required for this: The interconnection must be programmed in the user FC.

AUTOHOTSPOT

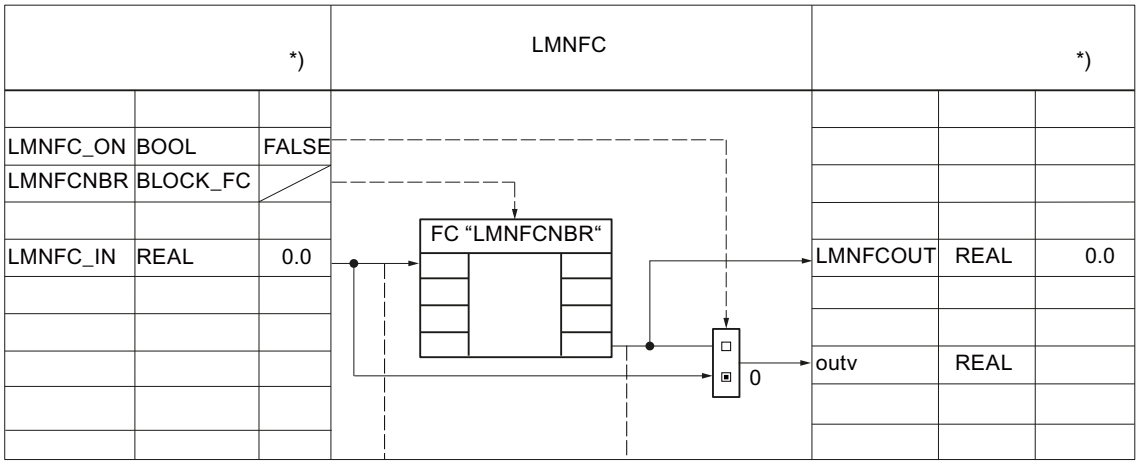

The rate of change of the manipulated variable can be limited using the following parameters.

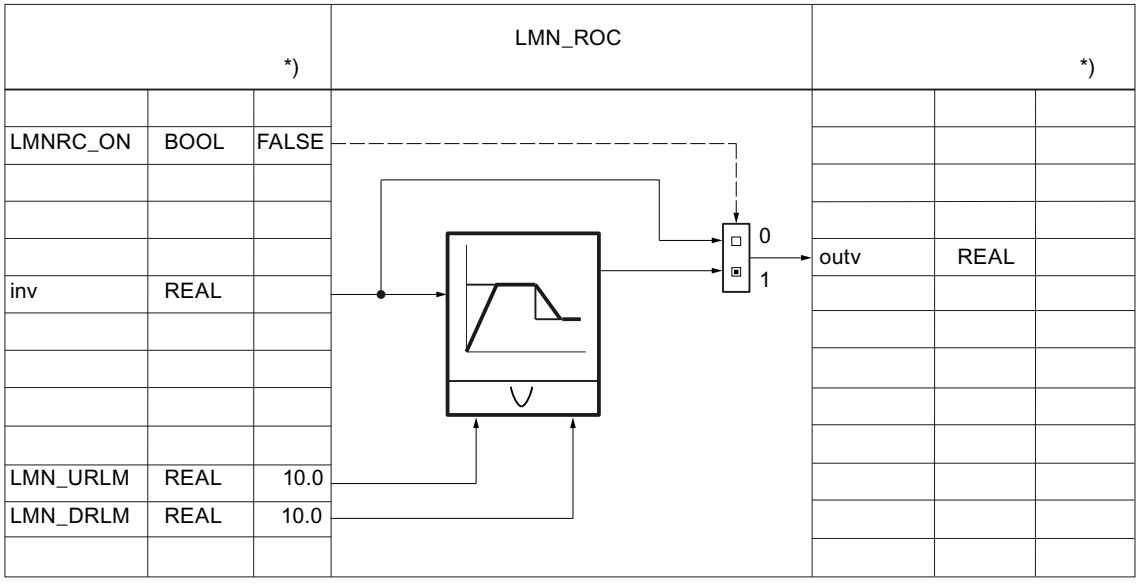

The manipulated variable is always monitored with the following parameters for an absolute high and low limit.

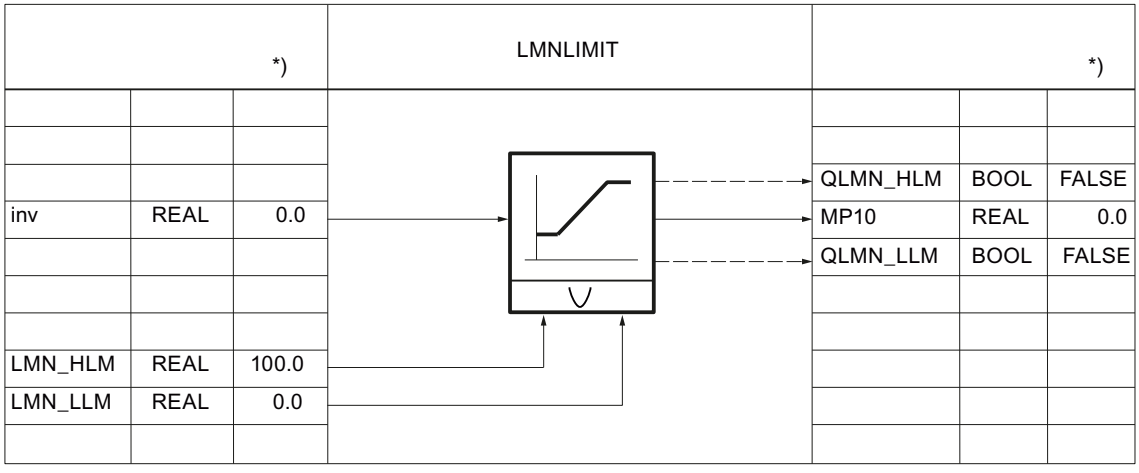

The manipulated variable is scaled to a physical value using the following parameters.

AUTOHOTSPOT

		*)	LMN_NORM			*)
MP10	REAL	0.0		LMN	REAL	0.0
LMN_FAC	REAL	1.0				
LMN_OFF	REAL	0.0				

The manipulated variable in floating-point format is converted to I/O format. In this process, no check is carried out for positive/negative overflow or for reaching the overrange/underrange. Module types are not taken into account.

The following table provides an overview of ranges and numerical values before and after processing.

Manipulated value LMN in %	I/O value LMN_PER
118.515	32767
100.000	27648
0.003617	1
0.000	0
-0.003617	-1
-100.000	-27648
-118.519	-32768

		*)	CRP_OUT			*)
LMN	REAL	0.0	%	LMN_PER	WORD	W#16#0000

The continuous manipulated variable can be output as pulse train via a pulse width modulation. The following parameters are required for this:

AUTOHOTSPOT

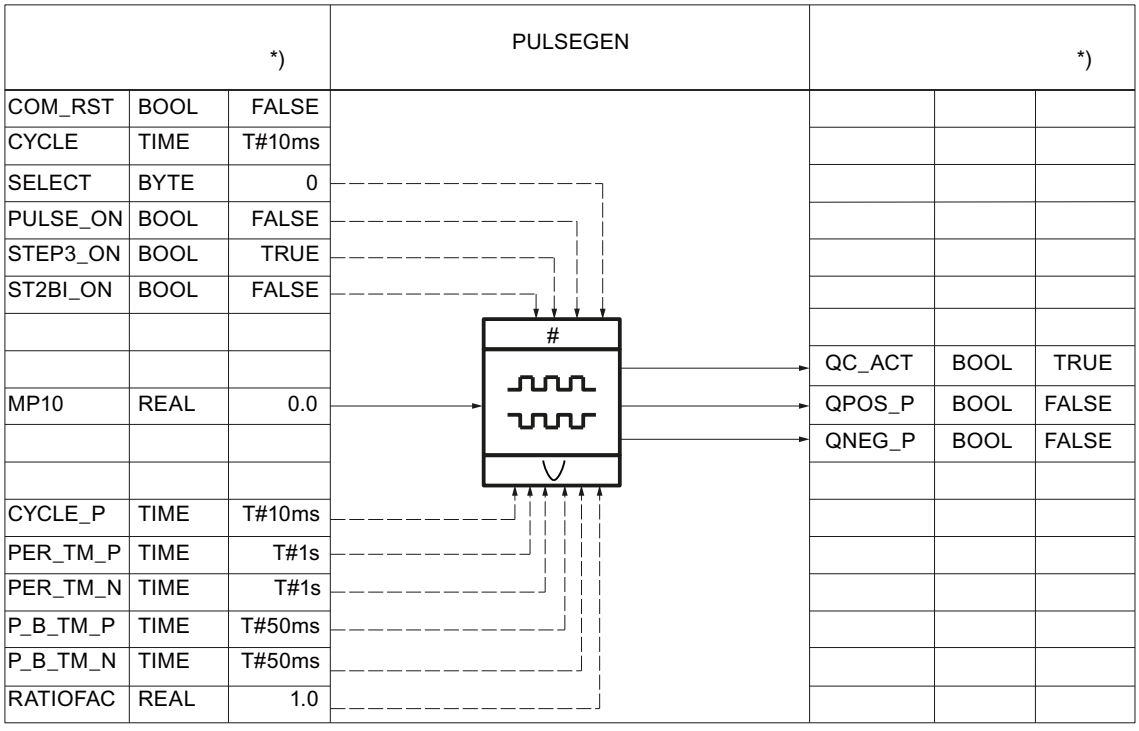

PID_CP in cascade arrangements (S7-300, S7-400)

Interruption of the controller cascade

In a cascade, several controllers are in direct relationship with each other. Therefore, you must take precautions to ensure problem-free cascade operation following an interruption of the cascade structure at any point.

In cascade controls, for this reason, a QCAS signal is generated in the secondary controllers by applying OR logic from the status signals of the switches in the setpoint and manipulated variable branches. This signal actuates a switch in the higher-level controllers that converts the controllers to follow-up mode. The follow-up value is always the setpoint of the lower-level circuit.

A smooth switchover occurs from follow-up mode to automatic mode, just as the switchover from manual to automatic mode.

PID_CP can be used in higher-level control loops as a master controller and in secondary control loops as a follow-up controller.

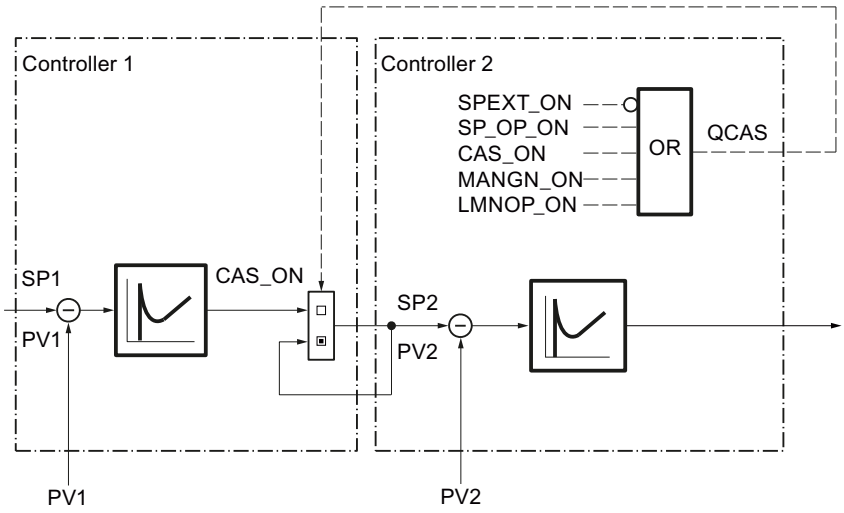

Note

The manipulated variable of the master controller LMN must always be interconnected with the external setpoint SP_EXT of the follow-up controller.

Block interconnection

The following figure shows the principle of the controller interconnection in multi-loop cascades.

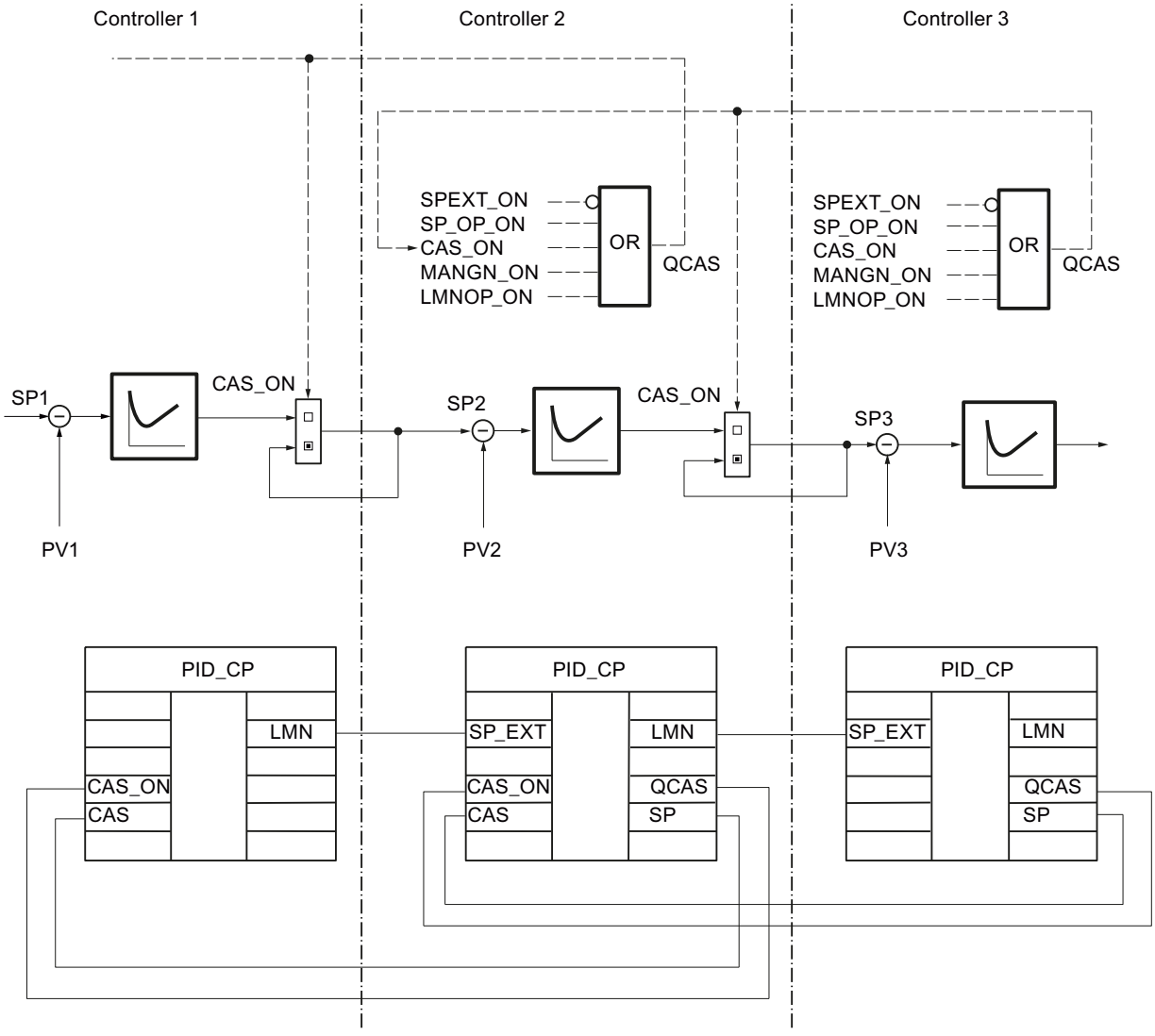

Input parameters for PID_CP (S7-300, S7-400)

Table 4-218

Parameter	Offset	Data type	Range of values	Preassigned value	Description
COM_RST	0.0	BOOL		FALSE	Restart (Initialization routine of the instruction will be processed)
I_SEL	0.1	BOOL		TRUE	Integral action on
D_SEL	0.2	BOOL		FALSE	Derivative action on

Parameter	Offset	Data type	Range of values	Preassigned value	Description
MAN_ON	0.3	BOOL		TRUE	Manual mode on (control loop interrupted, LMN set manually)
CAS_ON	0.4	BOOL		FALSE	Cascade mode ON (interconnected to QCAS of the lower-level controller)
SELECT	1.0	BYTE	0, 1, 2, 3	0	If PULS_ON = TRUE: 0: PID and pulse generator 1: PID (block call in OB 1) 2: Pulse generator (block call in cyclic interrupt OB) 3: PID (block call in cyclic interrupt OB)
CYCLE	2.0	TIME	> 20 ms (S7–300)	T#1s	Sampling time (time between two block calls = constant) Make sure to assign this parameter with the value of the cyclic interrupt cycle clock of the OB in which the PID_CP instruction runs. Otherwise, the time-dependent functions will not work correctly. (Exception: you are using time scaling, e.g., via the loop scheduler)
CYCLE_P	6.0	TIME		T#10 ms	Sampling time of the pulse generator Make sure to assign this parameter with the value of the cyclic interrupt cycle clock of the OB in which the PID_CP instruction runs. Otherwise, the time-dependent functions will not work correctly. (Exception: you are using time scaling, e.g., via the loop scheduler)
SP_INT	10.0	REAL	Engineering value range (physical quantity)	0.0	Internal setpoint SP_INT can be changed via a HMI with SPUP and SPDN
SP_EXT	14.0	REAL	Engineering value range (physical quantity)	0.0	External setpoint SP_EXT is used for blending, ratio and cascade controllers and can be scaled.
PV_IN	18.0	REAL	Engineering value range (physical quantity)	0.0	Process value input (PV in floating-point format)
PV_PER	22.0	INT		W#16#0000	Process value of I/O (PV in I/O format)
GAIN	24.0	REAL	Overall value range (dimensionless)	2.0	Proportional gain (= controller gain)
TI	28.0	TIME	TI ≥ CYCLE	T#20s	Integral action time
TD	32.0	TIME	TD ≥ CYCLE	T#10s	Derivative-action time
TM_LAG	36.0	TIME	TM_LAG ≥ CYCLE/2	T#2s	Time delay of derivative action
DISV	40.0	REAL		0.0	Disturbance variable Valid value range: -100.0 ... 100.0

Parameter	Offset	Data type	Range of values	Preassigned value	Description
CAS	44.0	REAL	Engineering value range (physical quantity)	0.0	Input for cascade mode (interconnected to PV of the lower-level controller)
SP_HLM	48.0	REAL	Engineering value range (physical quantity)	100.0	Setpoint high limit
SP_LLM	52.0	REAL	Engineering value range (physical quantity)	0.0	Setpoint low limit
LMN_HLM	56.0	REAL	LMN_LLM ... 100.0	100.0	Manipulated variable: High limit
LMN_LLM	60.0	REAL	-100.0 ... LMN_HLM	0.0	Manipulated variable: Low limit
DB_NBR	64.0	BLOCK_DB		DB 1	Data block number (DB with the time slices of the ramp/soak profile)
SPFC_NBR	66.0	BLOCK_FC		FC 0	Setpoint FC number (self-defined FC in setpoint branch)
PVFC_NBR	68.0	BLOCK_FC		FC 0	Process value FC number (self-defined FC in process value branch)
LMNFCNBR	70.0	BLOCK_FC		FC 0	Manipulated variable FC number (self-defined FC in manipulated variable branch)

Output parameters PID_CP (S7-300, S7-400)

Table 4-219

Parameter	Offset	Data type	Preassigned value	Description
LMN	72.0	REAL	0.0	Manipulated variable (Manipulated variable in floating-point format)
LMN_PER	76.0	INT	W#16#0000	Manipulated variable for I/O (LMN in I/O format)
SP	78.0	REAL	0.0	Setpoint (effective setpoint)
PV	82.0	REAL	0.0	Process value (output of effective process value in cascade mode)
QCAS	86.0	BOOL	FALSE	Signal for cascade mode (interconnected to CAS_ON of higher-level controller)
QC_ACT	86.1	BOOL	TRUE	Indicates whether the control functions will be processed at the next block call (only relevant if SELECT has the value 0 or 1)
QPOS_P	86.2	BOOL	FALSE	Pulse generator: Positive pulse ON
QNEG_P	86.3	BOOL	FALSE	Pulse generator: Negative pulse ON

In/out parameters PID_CP (S7-300, S7-400)

Table 4-220

Parameter	Offset	Data type	Preassigned value	Description
MAN	88.0	REAL	0.0	Manual manipulated variable (for manipulated variable assignment by means of operator control and monitoring functions)

Static variables PID_CP (S7-300, S7-400)

Parameter	Offset	Data type	Range of values	Preassigned value	Description
PVH_ALM	92.0	REAL	PVH_WRN...100.0	100.0	Process value: High limit 'Alarm'
PVH_WRN	96.0	REAL	PVL_WRN... PVH_ALM	90.0	Process value: High limit 'Warning'
PVL_WRN	100.0	REAL	PVL_ALM... PVH_WRN	-90.0	Process value: Low limit 'Warning'
PVL_ALM	104.0	REAL	-100.0...PVL_WRN	-100.0	Process value: Low limit 'Alarm'
SPGEN_ON	108.0	BOOL		FALSE	Setpoint manipulation ON (adjusting the setpoint using the up/down switch)
SPUP	108.1	BOOL		FALSE	Setpoint UP
SPDN	108.2	BOOL		FALSE	Setpoint DOWN
RMPSK_ON	108.3	BOOL		FALSE	Ramp/soak function ON (setpoint is specified as a ramp/soak profile)
SPEXT_ON	108.4	BOOL		FALSE	External setpoint ON (for interconnection with other controller blocks)
MANGN_ON	108.5	BOOL		FALSE	Manual mode ON (LMN set via switch)
MANUP	108.6	BOOL		FALSE	Manual output value UP
MANDN	108.7	BOOL		FALSE	Manual output value DOWN
DFRMP_ON	109.0	BOOL		FALSE	Preassign output of ramp/soak function (output is preassigned with SP_INT)
CYC_ON	109.1	BOOL		FALSE	Cyclic repetition ON (ramp/soak profile will be repeated automatically)
RMP_HOLD	109.2	BOOL		FALSE	Ramp/soak function HOLD (output of ramp/soak function is frozen)
CONT_ON	109.3	BOOL		FALSE	Continue (with schedule) (ramp/soak profile will be continued at the next time slice)
TUPDT_ON	109.4	BOOL		FALSE	Recalculate total time (total time of ramp/soak profile will be recalculated)

Parameter	Offset	Data type	Range of values	Preassigned value	Description
SPFC_ON	109.5	BOOL		FALSE	Call setpoint FC
SPROC_ON	109.6	BOOL		FALSE	Ramp function ON (rate of change limiting of SP)
PVPER_ON	109.7	BOOL		FALSE	Process value of I/O ON (interconnection to I/O modules)
LAG1STON	110.0	BOOL		FALSE	First-order delay element ON
SQRT_ON	110.1	BOOL		FALSE	Square root function ON
PVFC_ON	110.2	BOOL		FALSE	Call process value FC
DEADB_ON	110.3	BOOL		FALSE	Deadband ON (small disturbances and noise will be filtered)
P_SEL	110.4	BOOL		TRUE	P component ON
PFDB_SEL	110.5	BOOL		FALSE	Switch P component to feedback
INT_HPOS	110.6	BOOL		FALSE	Freeze I component in positive direction
INT_HNEG	110.7	BOOL		FALSE	Freeze I component in negative direction
I_ITL_ON	111.0	BOOL		FALSE	Preassign I component
DFDB_SEL	111.1	BOOL		FALSE	Switch D component to feedback
DISV_SEL	111.2	BOOL		FALSE	Apply disturbance variable
LMNFC_ON	111.3	BOOL		FALSE	Call manipulated variable FC
LMNRC_ON	111.4	BOOL		FALSE	Activate manipulated variable ramp function (rate of change limiting of LMN)
SMOO_CHG	111.5	BOOL		TRUE	Smooth changeover from manual to automatic
PULSE_ON	111.6	BOOL		FALSE	Pulse generator ON
STEP3_ON	111.7	BOOL		TRUE	Pulse generator: Three-step control ON
ST2BI_ON	112.0	BOOL		FALSE	Pulse generator: two-step control for bipolar control range ON (for unipolar control range, STEP3_ON = FALSE must be set)
TM_SNBR	114.0	INT	≥ 0 (dimensionless)	0	No. of next time slice for continuation
TM_CONT	116.0	TIME	Overall value range (dimensionless)	T#0s	Continuation point (Time after time slice TM_SNBR at which the ramp/soak function continues processing the ramp/soak profile)
FAC	120.0	REAL	Overall value range (dimensionless)	1.0	Factor (ratio or blending factor)
NM_SPEHR	124.0	REAL		100.0	Setpoint scaling: Operating range input high
NM_SPELR	128.0	REAL		-100.0	Setpoint scaling: Operating range input low
SPFC_OUT	132.0	REAL	-100.0 ... 100.0	0.0	Setpoint FC output (interconnected to the output of the FC in the setpoint branch)
SPURLM_P	136.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint up rate limit in positive range
SPDRLM_P	140.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint down rate limit in positive range

Parameter	Offset	Data type	Range of values	Preassigned value	Description
SPURLM_N	144.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint up rate limit in negative range
SPDRLM_N	148.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint down rate limit in negative range
NM_PIHR	152.0	REAL		100.0	Process value scaling: Measuring range input high
NM_PILR	156.0	REAL		-100.0	Process value scaling: Measuring range input low
NM_PVHR	160.0	REAL		100.0	Process value scaling: Measuring range output high
NM_PVLR	164.0	REAL		-100.0	Process value scaling: Measuring range output low
PV_TMLAG	168.0	TIME	Overall value range	T#5s	Process value time delay (delay of PT1 element in the PV branch)
SQRT_HR	172.0	REAL		100.0	Square root: Working range output high
SQRT_LR	176.0	REAL		0.0	Square root: Working range output low
PVFC_OUT	180.0	REAL	-100.0 ... 100.0	0.0	Process value FC output (interconnected to the output of the FC in the process value branch)
PVURLM_P	184.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value up rate limit in the positive range
PVDRLM_P	188.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value down rate limit in the positive range
PVURLM_N	192.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value up rate limit in the negative range
PVDRLM_N	196.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value down rate limit in the negative range
PV_HYS	200.0	REAL	≥ 0	1.0	Process value hysteresis (prevents 'flickering' of monitoring display)
DEADB_W	204.0	REAL	0.0 ... 100.0	1.0	Deadband width (= range from zero to deadband high limit) (determines the size of the deadband area)
ERP_ALM	208.0	REAL	0 ... 200.0	100.0	Control deviation: Positive limit 'Alarm'
ERP_WRN	212.0	REAL	0 ... 200.0	90.0	Control deviation: Positive limit 'Warning'
ERN_WRN	216.0	REAL	-200.0 ... 0	-90.0	Control deviation: Negative limit 'Warning'
ERN_ALM	220.0	REAL	-200.0 ... 0	-100.0	Control deviation: Negative limit 'Alarm'
ER_HYS	224.0	REAL	≥ 0 [%]	1.0	Control deviation hysteresis (prevents 'flickering' of monitoring display)
I_ITLVAL	228.0	REAL	-100.0 ... 100.0 [%]	0.0	Initialization value for I action
LMNFCOUT	232.0	REAL	-100.0 ... 100.0 [%]	0.0	Manipulated variable FC output (interconnected to the output of the FC in the manipulated variable branch)
LMN_URLM	236.0	REAL	≥ 0 [%/s]	10.0	Manipulated variable up rate limit
LMN_DRLM	240.0	REAL	≥ 0 [%/s]	10.0	Manipulated variable down rate limit

Parameter	Offset	Data type	Range of values	Preassigned value	Description
LMN_FAC	244.0	REAL	Overall value range (dimensionless)	1.0	Manipulated variable factor (factor for adaptation of the manipulated variable range)
LMN_OFF	248.0	REAL	Overall value range (dimensionless)	0.0	Manipulated variable offset (zero point of manipulated variable scaling)
PER_TM_P	252.0	TIME		T#1s	Pulse generator: Period of positive pulse
PER_TM_N	256.0	TIME		T#1s	Pulse generator: Period of negative pulse
P_B_TM_P	260.0	TIME		T#50ms	Pulse generator: Minimum pulse time or minimum break time of positive pulse
P_B_TM_N	264.0	TIME		T#50ms	Pulse generator: Minimum pulse time or minimum break time of negative pulse
RATIOFAC	268.0	REAL	0.1 ... 10.0 (dimensionless)	1.0	Pulse generator: Ratio factor (ratio of positive pulse time and negative pulse time)
PHASE	272.0	INT		0	Phase of PID self tuner
QPVH_ALM	274.0	BOOL		FALSE	Process value: High limit 'Alarm' triggered
QPVH_WRN	274.1	BOOL		FALSE	Process value: High limit 'Warning' triggered
QPVL_WRN	274.2	BOOL		FALSE	Process value: Low limit 'Warning' triggered
QPVL_ALM	274.3	BOOL		FALSE	Process value: Low limit 'Alarm' triggered
QR_S_ACT	274.4	BOOL		FALSE	Time table for ramp/soak profile will be processed
QSP_HLM	274.5	BOOL		FALSE	Setpoint: High limit triggered
QSP_LLM	274.6	BOOL		FALSE	Setpoint: Low limit triggered
QPVURLMP	274.7	BOOL		FALSE	Process value: Up rate limit in positive range triggered
QPVDRLMP	275.0	BOOL		FALSE	Process value: Down rate limit in positive range triggered
QPVURLMN	275.1	BOOL		FALSE	Process value: Up rate limit in negative range triggered
QPVDRLMN	275.2	BOOL		FALSE	Process value: Down rate limit in negative range triggered
QERP_ALM	275.3	BOOL		FALSE	Control deviation: Positive limit 'Alarm' triggered
QERP_WRN	275.4	BOOL		FALSE	Control deviation: Positive limit 'Warning' triggered
QERN_WRN	275.5	BOOL		FALSE	Control deviation: Negative limit 'Warning' triggered
QERN_ALM	275.6	BOOL		FALSE	Control deviation: Negative limit 'Alarm' triggered
QLMN_HLM	275.7	BOOL		FALSE	Manipulated variable: High limit triggered
QLMN_LLM	276.0	BOOL		FALSE	Manipulated variable: Low limit triggered
NBR_ATMS	278.0	INT		0	Number of the time slice currently being approached by the ramp/soak function
RS_TM	280.0	TIME		T#0s	"Current residual time" of ramp/soak profile until the next time slice
T_TM	284.0	TIME		T#0s	"Total time" of ramp/soak profile

Parameter	Offset	Data type	Range of values	Preassigned value	Description
RT_TM	288.0	TIME		T#0s	"Total residual time" = time remaining until the end of the ramp/soak profile
ER	292.0	REAL		0.0	Control deviation
LMN_P	296.0	REAL		0.0	P component
LMN_I	300.0	REAL		0.0	I component
LMN_D	304.0	REAL		0.0	D component
SPFC_IN	308.0	REAL		0.0	Setpoint FC input (interconnected to input of the self-defined FC)
PVFC_IN	312.0	REAL		0.0	Process value FC input (interconnected to input of the self-defined FC)
LMNFC_IN	316.0	REAL		0.0	Manipulated variable FC input (interconnected to input of the self-defined FC)
SP_OP_ON	320.0	BOOL		FALSE	Setpoint manipulation ON (value of SP_OP will be applied as the setpoint)
PV_OP_ON	320.1	BOOL		FALSE	Process value manipulation ON (value of PV_OP will be applied as the setpoint)
LMNOP_ON	320.2	BOOL		FALSE	Manipulated variable manipulation ON (value of LMN_OP will be applied as the setpoint)
SP_OP	322.0	REAL		0.0	Setpoint manipulation during commissioning
PV_OP	326.0	REAL		0.0	Process value manipulation during commissioning
LMN_OP	330.0	REAL		0.0	Manipulated variable manipulation during commissioning
MP1	334.0	REAL		0.0	Measuring point 1: Internal setpoint
MP2	338.0	REAL		0.0	Measuring point 2: External setpoint
MP3	342.0	REAL		0.0	Measuring point 3: Unlimited setpoint
MP4	346.0	REAL		0.0	Measuring point 4: Process value of the I/O module
MP5	350.0	REAL		0.0	Measuring point 5: Process value after first-order delay element
MP6	354.0	REAL		0.0	Measuring point 6: Effective process value (PV)
MP7	358.0	REAL		0.0	Measuring point 7: Manipulated variable of PID algorithm
MP8	362.0	REAL		0.0	Measuring point 8: Manual output value
MP9	366.0	REAL		0.0	Measuring point 9: Unlimited manipulated variable
MP10	370.0	REAL		0.0	Measuring point 10: Limited manipulated variable

Block diagrams for PID_CP (S7-300, S7-400)

Signal-flow representations

Analogous to the representation of the switches in the configuration tool, the black dot in each switch symbol indicates that the switching signal has the adjacent Boolean value (0=FALSE or 1=TRUE) and that the signal path is switched through this point in each case. The switching signals (binary signals) are identified by dashed lines.

This means that the setpoint is specified as an absolute setpoint via SP_INT; the same applies to the process value input via PV_IN. The default control function is a standard PI controller with P function in the forward branch. The control loop is open and the percentage range of the manipulated variable is influenced by the MAN input. All other functions are passive or, if they cannot be deactivated, marginal signal characteristic parameters are preassigned within the measuring range or operating range so that the functions have no effect.

Symbols and identifiers in the signal flow graphs

The names of the connectable process variables are shown on a shaded background. This allows you to recognize where the controller structure can be connected to the S7 I/O or directly to the measuring and actuators of the process.

Parameter names containing the combination of letters "OP" (for example, SP_OP/ SP_OP_ON) indicate that an intervention using the Standard PID Control configuration tool is possible at this point. The configuration tool has its own interface to the controller FB.

Intermediate variables in the signal characteristic can be monitored at the measuring points MP1 to MP12 marked with circles. These intermediate variables are required in order to match values before triggering "smooth" changeovers or to check the current statuses of the relevant control. The measuring point variables in the configuration tool curve recorder can be represented (statically and dynamically).

For reasons of clarity, the parameters for setting and sizing the processing operations (algorithms) are also indicated in the individual function blocks. Please refer to the descriptions in the reference section and to the representation of the individual subfunctions in the following sections.

Processing the setpoint

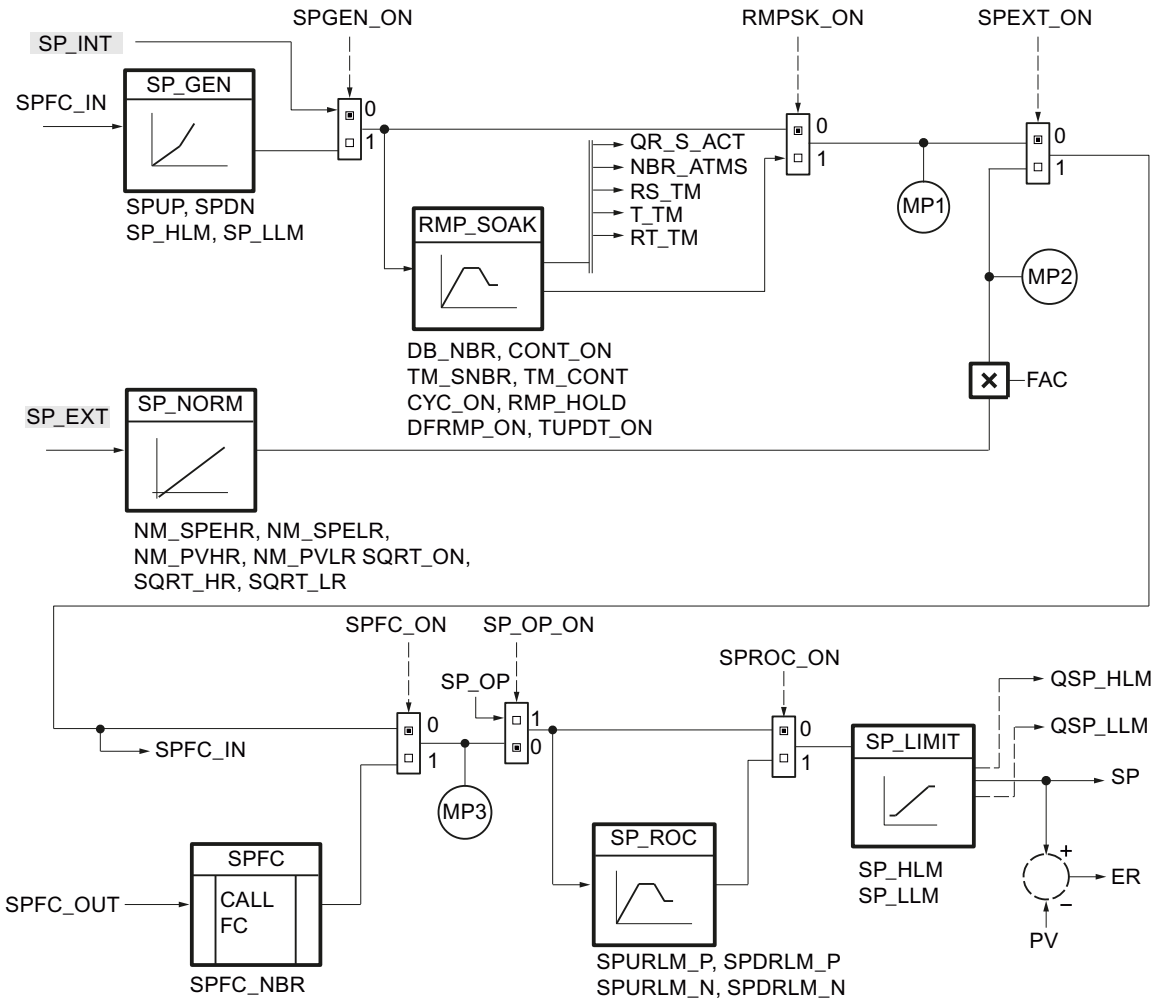

Processing the process value

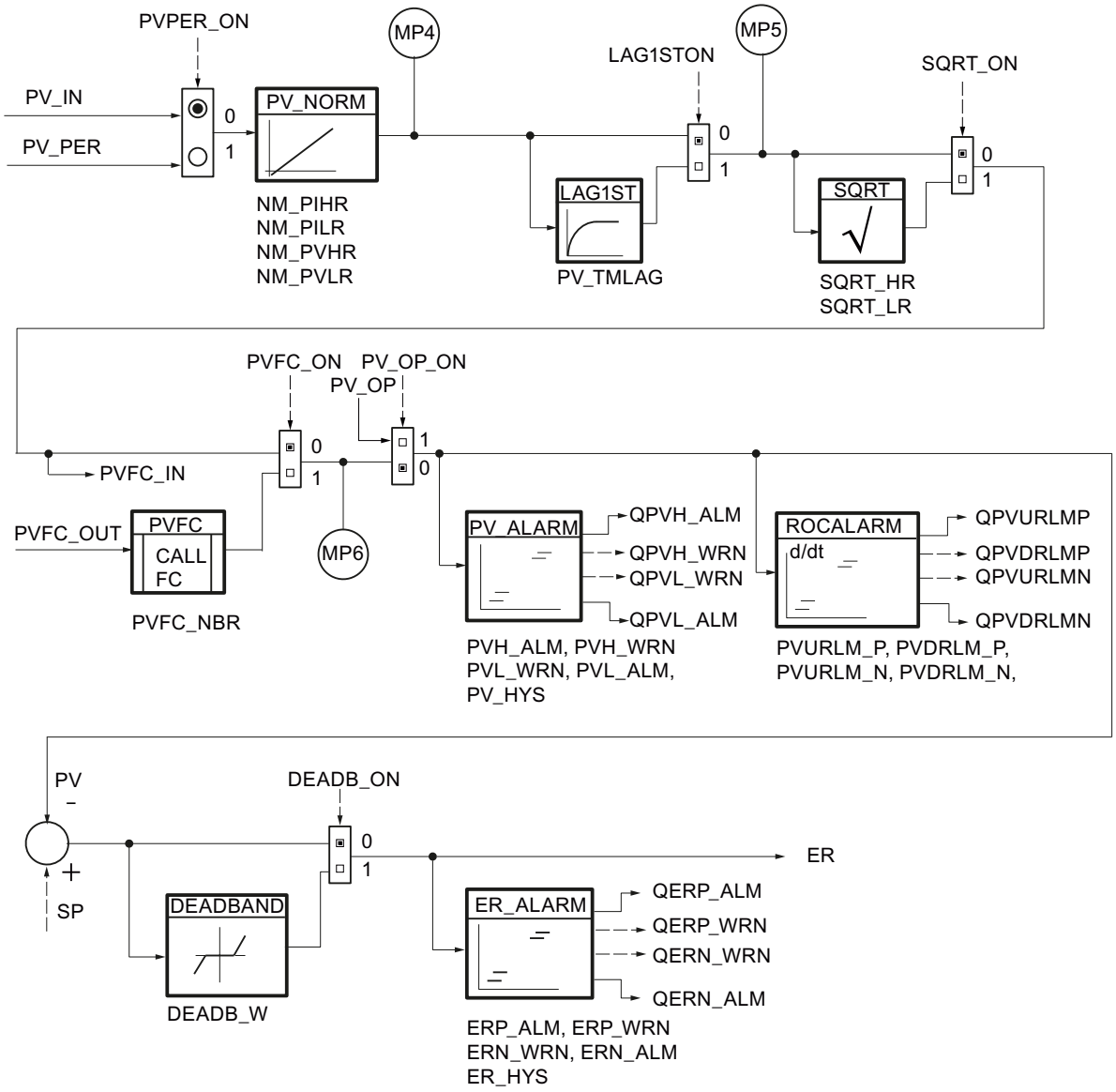

Control algorithm

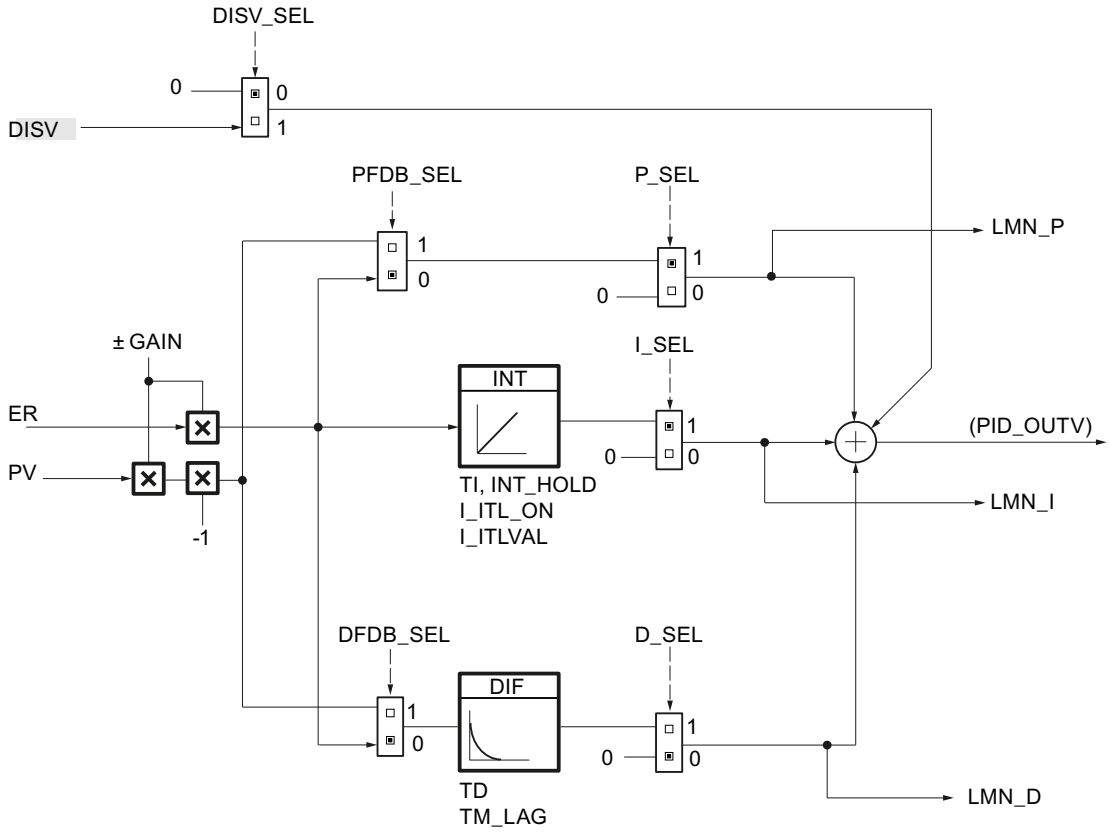

Processing the analog manipulated variable

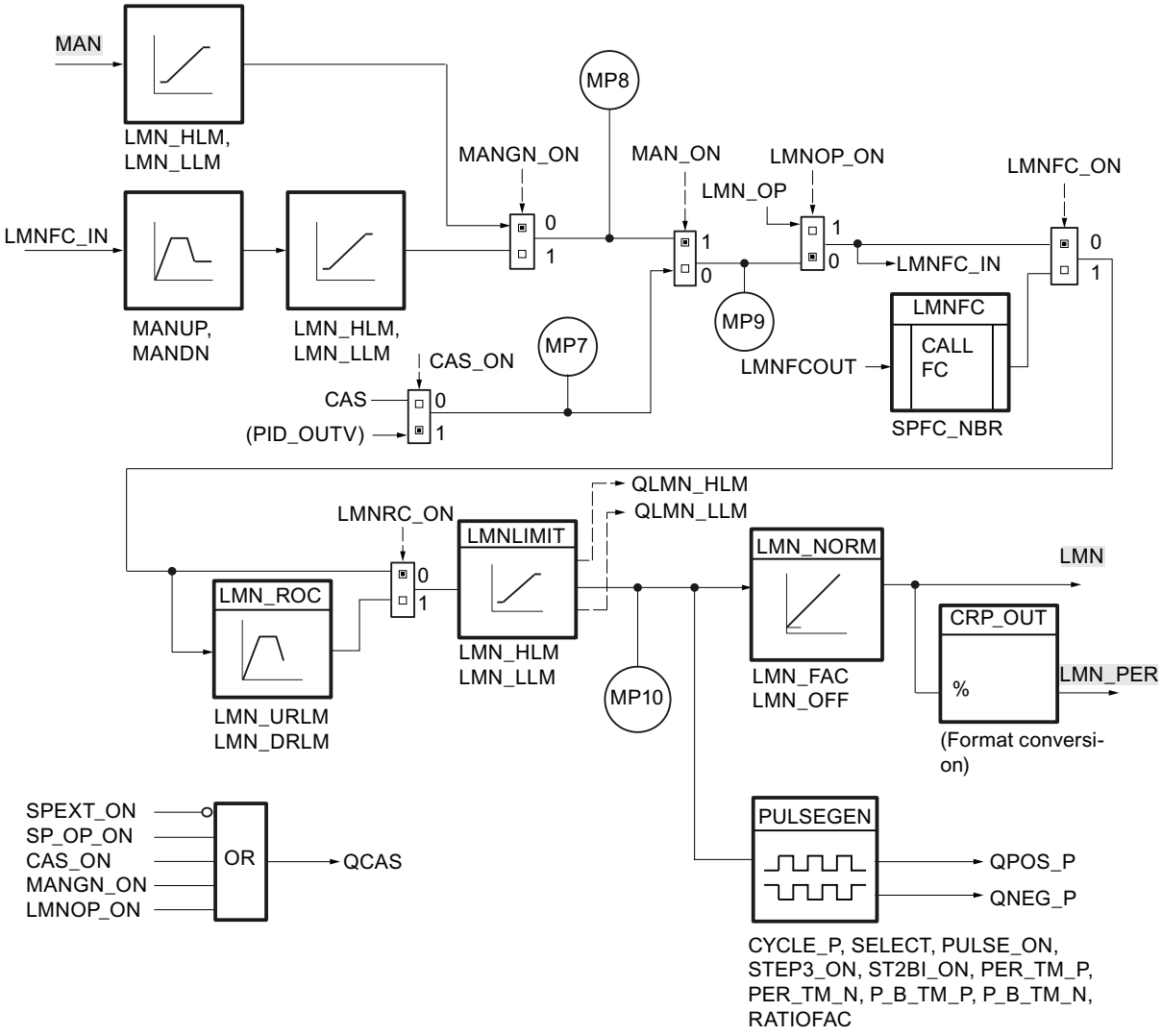

Depending on the combination of the switching states, the OR gate generates an enable signal for the cascade coupling.

Global data block DB_RMPSK (S7-300, S7-400)

Note

Creating a global data block yourself

The time slices for the ramp/soak function (RMP_SOAK) are specified in a global data block. This global data block is not included in the library. You must create the global data block yourself according to the following schema and adapt it to your application.

Parameter	Data type	Default	Description
NBR_PTS	INT	4	Number of time slices -1 1 - 255
PI	Array [0...NBR_PTS] of Struct		Structure for the time slice data. The array must be declared by 0...NBR_PTS.
PI[0].OUTV	REAL	0.0	Output variable [0] Starting point Engineering value range
PI[0].TMV	TIME	T#1s	Time value [0] Starting point
PI[1].OUTV	REAL	0.0	Output variable [1] Time slice 1 Engineering value range
PI[1].TMV	TIME	T#1s	Time value [1] Time slice 1
PI[2].OUTV	REAL	0.0	Output variable [2] Time slice 2 Engineering value range
PI[2].TMV	TIME	T#1s	Time value [2] Time slice 2
PI[3].OUTV	REAL	0.0	Output variable [3] Time slice 3 Engineering value range
PI[3].TMV	TIME	T#1s	Time value [3] Time slice 3
PI[4].OUTV	REAL	0.0	Output variable [4] Time slice 4 Engineering value range
PI[4].TMV	TIME	T#0s	Time value [4] Time slice 4 0 ms

Generating a data block using an external source file

Alternatively, you can generate the global data block using an external source file.

Proceed as follows:

1. Copy the text below to the clipboard.
2. Open an external text editor.
3. Paste the copied text from the clipboard to the text editor.
4. Save the file with the file name extension "DB":
5. Open the "External sources" folder in the project tree of the TIA Portal.
6. Double-click the command "Add new external file".
   The "Open" dialog is displayed.
7. Navigate to the external source file that you have created and select it.
8. Confirm your selection with "Open".
9. Select the external source file.
10. Select the command "Generate blocks from source" in the shortcut menu.
11. A security prompt informs you that any existing blocks will be overwritten.

12. Click "OK" in response to the security prompt.
13. Adapt the created data block to your application.

```
DATA_BLOCK "DB_RMPSK"
TITLE = data block ramp soak
{ S7_Optimized_Access := 'FALSE' }
NAME : DB_RMPSK
    STRUCT
        NBR_PTS : Int;              // number of points (excluding
                                    starting point)
        PI : Array[0..4] of Struct
            OUTV : Real;            // output variable
            TMV : Time;             // output time value
        END_STRUCT;
    END_STRUCT;


BEGIN
    NBR_PTS := 4;
    PI[0].TMV := T#1s;
    PI[1].TMV := T#1s;
    PI[2].TMV := T#1s;
    PI[3].TMV := T#1s;
    PI[4].TMV := T#0s;
END_DATA_BLOCK
```

PID_ES (S7-300, S7-400)

Description PID_ES (S7-300, S7-400)

In addition to the functions in the setpoint and process value branches, the instruction implements a complete PID controller with binary manipulated variable output. It is possible to influence the manipulated variable manually. Subfunctions can be enabled or disabled.

The instruction enables technical processes and systems with integral-action actuators to be controlled on SIMATIC S7 automation systems. Slow (temperatures, filling levels etc.) as well as fast controlling systems (flow, speed etc.) can be controlled.

The controller can be used individually as a fixed setpoint controller, or in secondary control loops in cascade, blending or ratio controls; however it cannot be used as the primary controller.

Signal processing in the setpoint/process value branch as well as handling and monitoring of the control deviation is identical to that of the continuous controller.

Startup

The PID_ES instruction features an initialization routine that is executed when input parameter COM_RST = TRUE is set.

The integral action is set to the initialization value I_ITLVAL at startup and read out at output LMN_I. When it is called in a cyclic interrupt level, it then continues to work starting at this value.

When ramp/soak is enabled, the time intervals DB_NBR PI[0 ... NBR_PTS].TMV between the time slices are added up and displayed at the total time T_TM and total remaining time RT_TM outputs. In the case of online changes of PI[n].TMV or specification of TM_CONT and TM_SNBR, the total time and the total remaining time of the ramp/soak profile change. Since the calculation of T_TM and RS_TM with many time slices significantly increases the processing time of the RMP_SOAK function, the calculation is only carried out at a restart or when TUPDT_ON = TRUE.

All other outputs are set to their default values.

Hot restart

With a hot restart, the operating state that existed at the time of the interruption is assumed. The control continues to operate with the values that were calculated at the moment the interruption occurred.

Call

The instruction PID_ES must be called with a constant bus cycle time. To achieve this, use a cyclic interrupt level (for example, OB35).

If you call the instruction PID_ES as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_ES directly in the multiple instance DB and commission it via a watch table.

Control deviation generation for PID_ES (S7-300, S7-400)

The following functions are always active and cannot be disabled.

- Setpoint limiting SP_LIMIT,
- Process value monitoring for absolute values PV_ALARM,
- Process value monitoring for size of rate of change ROCALARM,
- Control deviation monitoring for absolute values ER_ALARM,
- Limiting of the output signal LMNLIMIT.

The default values are selected in such a manner that operation requires no individual adjustment. To achieve good controller results, you always have to adapt the parameters of these functions to your process.

Processing the process value

The process value is scaled to the physical value.

AUTOHOTSPOT

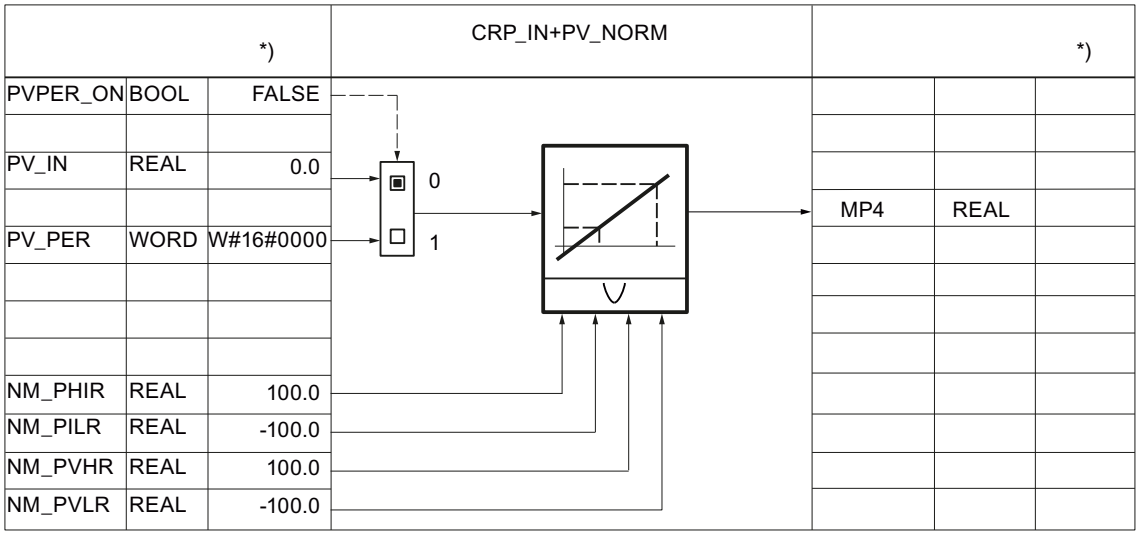

*) Default for creation of new technology object.

The process value can be smoothed with a first-order delay element.

AUTOHOTSPOT

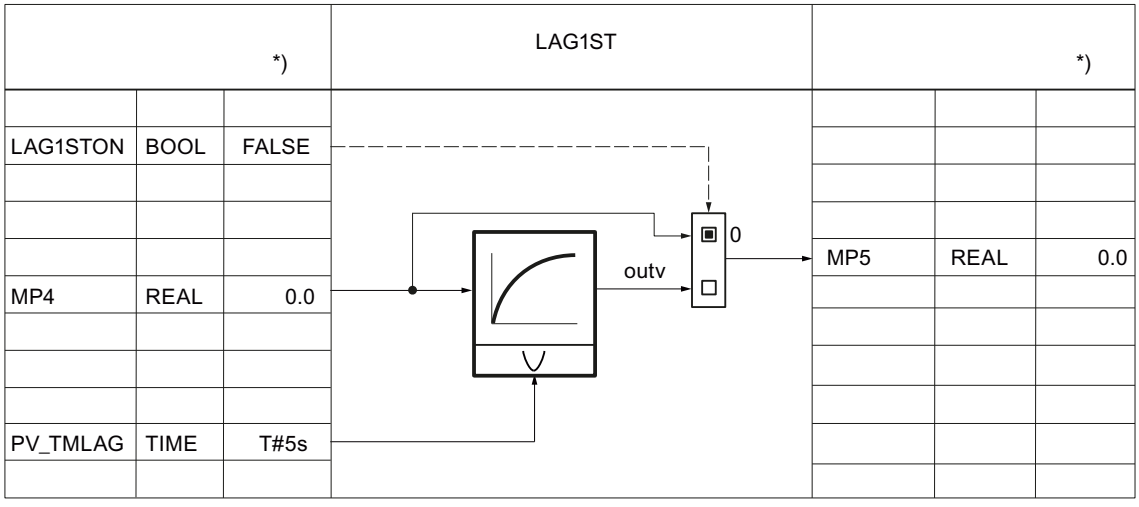

The process value can be linearized with the square root. The following parameters are required for this:

AUTOHOTSPOT

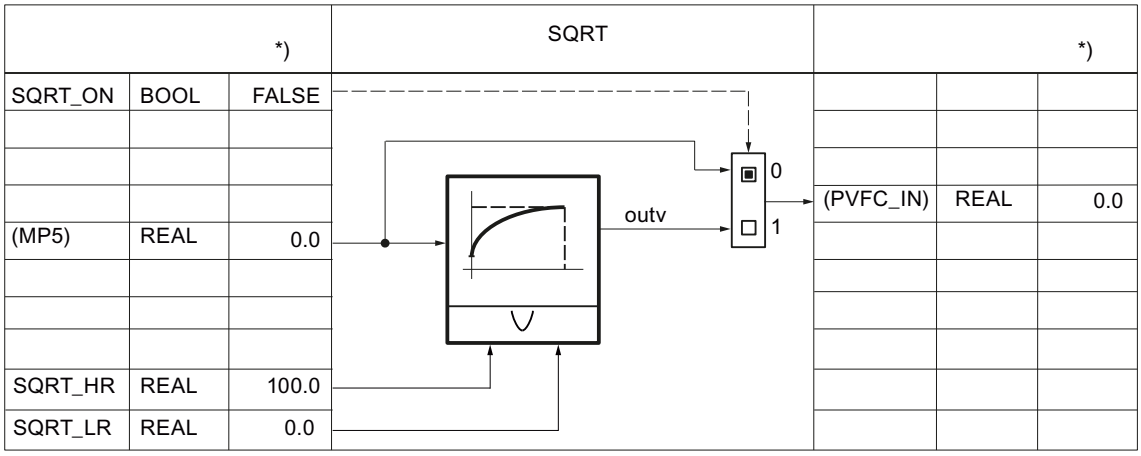

The process value can be prepared with a user-specific function. The following parameters are required for this: The interconnection must be programmed in the user FC.

AUTOHOTSPOT

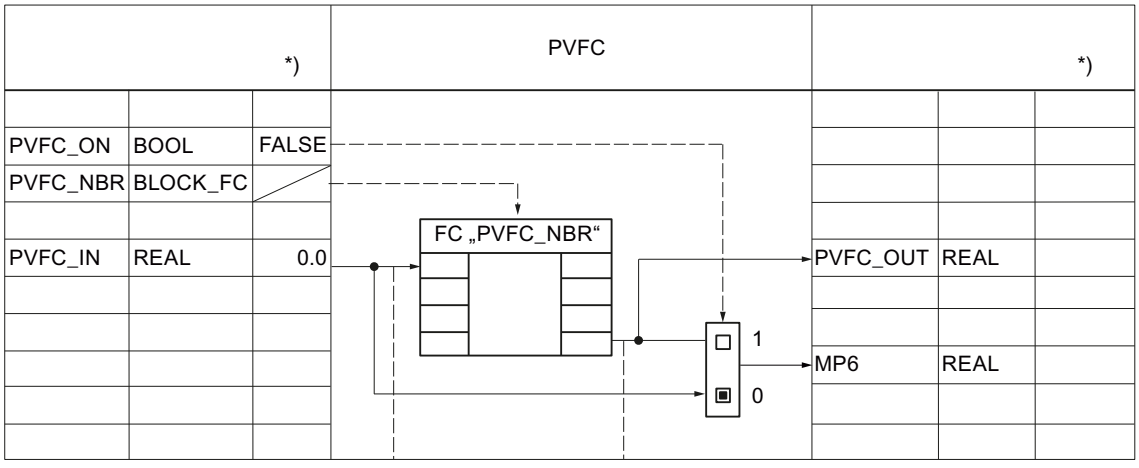

The process value is always monitored with the following parameters for warning and alarm limits.

AUTOHOTSPOT

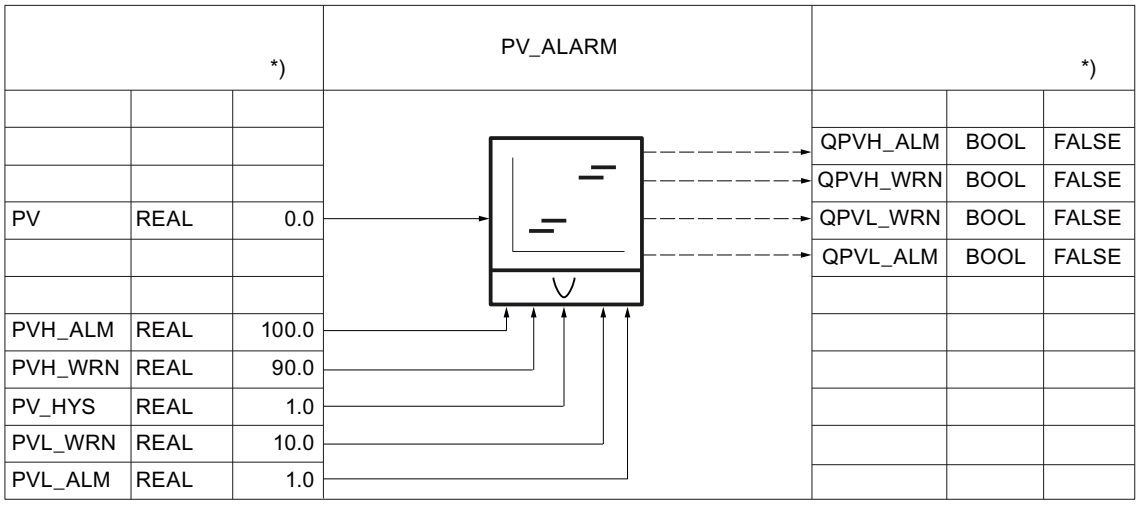

The rate of change of the process value is always limited with the following parameters.

AUTOHOTSPOT

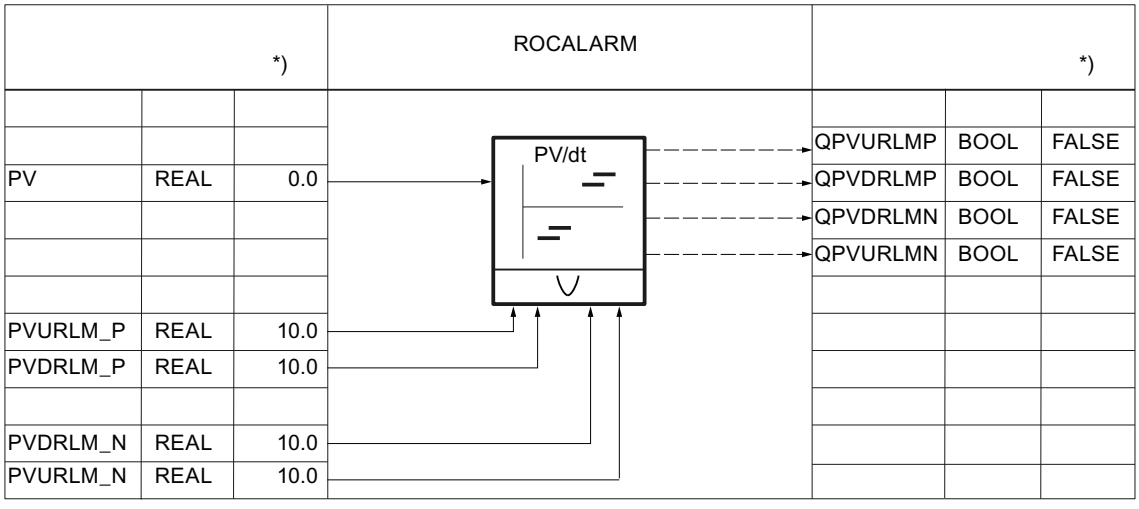

Processing setpoints

The internal setpoint SP_IN can be incremented or decremented with the following parameters.

AUTOHOTSPOT

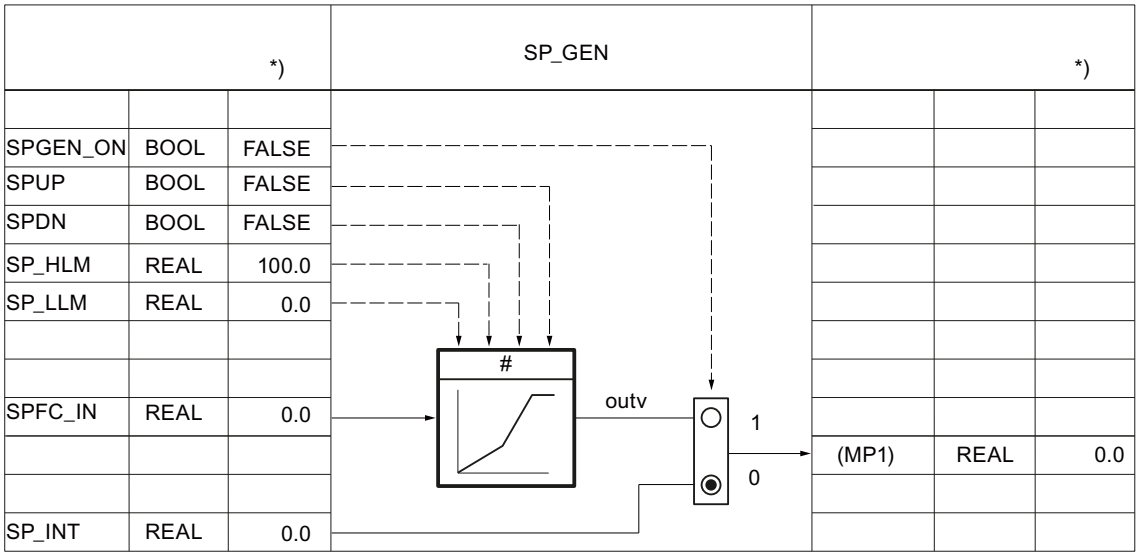

The setpoint can be specified with the following parameters via a ramp/soak function.

AUTOHOTSPOT

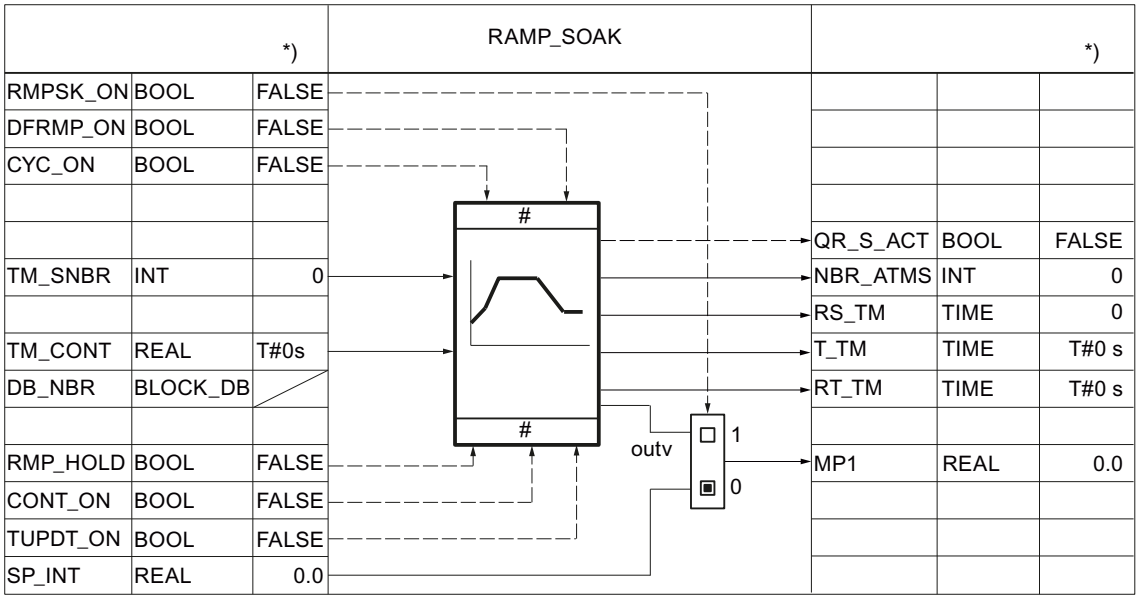

The blending/ratio controller can scale the external setpoint SP_EXT using the following parameters.

AUTOHOTSPOT

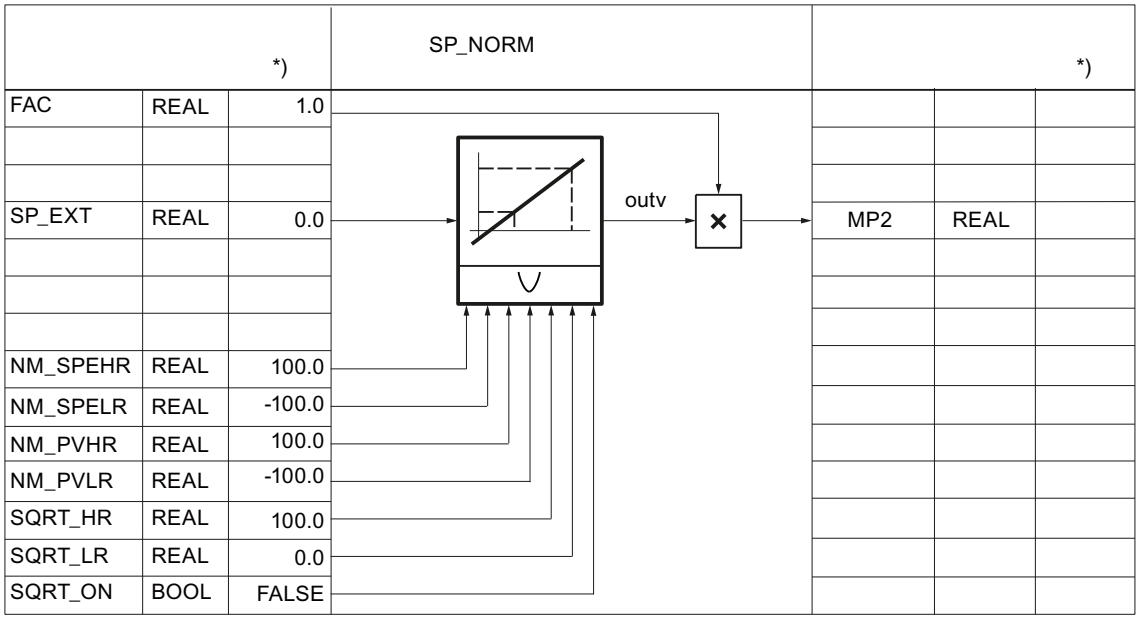

The setpoint can be further processed with a user-specific function. The following parameters are required for this: The interconnection must be programmed in the user FC.

The input value SPFC_IN is an implicit parameter. This can be monitored with the configuration tool, either via the measuring point MP1 (SPEXT_ON = FALSE) or via the measuring point MP2 (SPEXT_ON = TRUE). The output value is accessible at the measuring point MP3. The input SPFC_IN is then switched through on the setpoint branch, if SPFC_ON =FALSE.

AUTOHOTSPOT

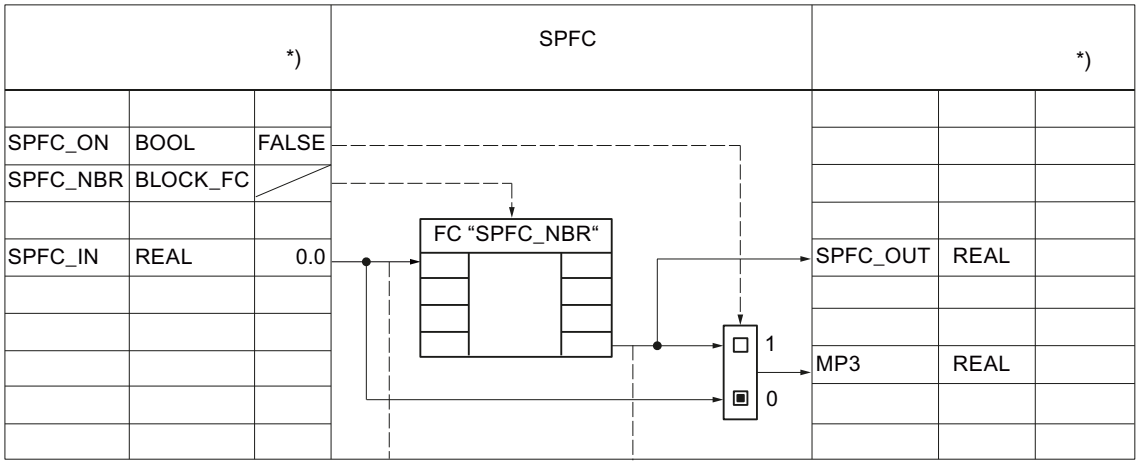

The rate of change of the setpoint can be limited with the following parameters.

		*)	SP_ROC			*)
SPROC_ON	BOOL	FALSE				
inv	REAL			outv	REAL	
SPURLM_P	REAL	10.0				
SPDRLM_P	REAL	10.0				
SPURLM_N	REAL	10.0				
SPDRLM_N	REAL	10.0				

The setpoint value is always monitored with the following parameters for an absolute high and low limit.

		*)	SP_LIMIT			*)
				QSP_HLM	BOOL	FALSE
inv	REAL			SP	REAL	0.0
				QSP_LLM	BOOL	FALSE
SP_HLM	REAL	100.0				
SP_LLM	REAL	0.0				

Generating control deviation

The following parameters can be used to route the control deviation over a deadband.

AUTOHOTSPOT

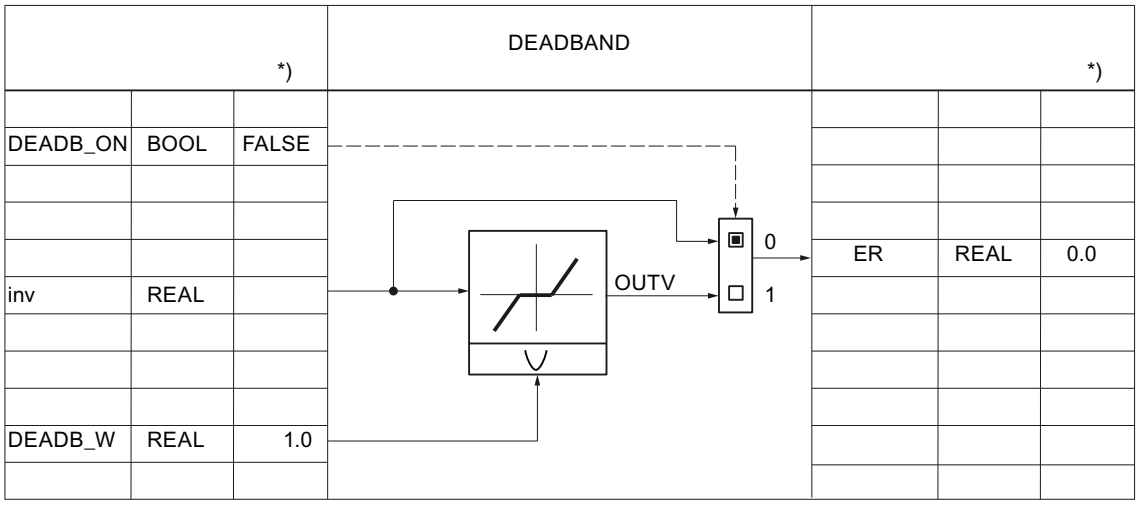

The control deviation is always monitored with the following parameters for warning and alarm limits.

AUTOHOTSPOT

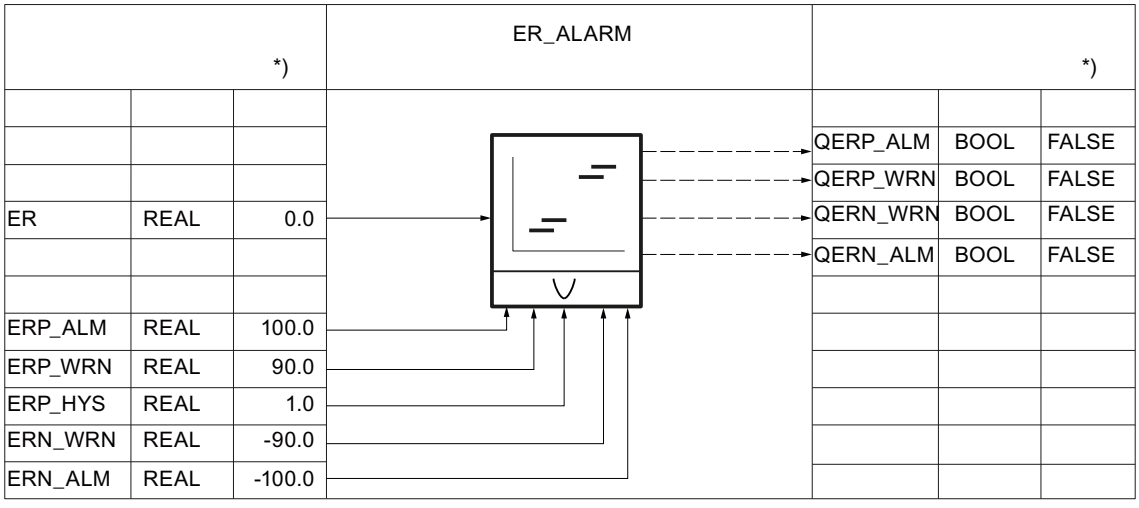

Control algorithm for PID_ES (S7-300, S7-400)

Control algorithm

AUTOHOTSPOT

A total of five switches are available to specify an effective controller structure.

AUTOHOTSPOT

Controller structure	Switch				
	P_SEL	I_SEL	D_SEL	PFDB_SEL	DFDB_SEL
P control	TRUE	FALSE	FALSE	FALSE	FALSE
P control (P in feedback path)	TRUE	FALSE	FALSE	TRUE	FALSE
PI control	TRUE	TRUE	FALSE	FALSE	FALSE
PI control (P in feedback path)	TRUE	TRUE	FALSE	TRUE	FALSE
PD control	TRUE	FALSE	TRUE	FALSE	FALSE
PD control (P in feedback path)	TRUE	FALSE	TRUE	FALSE	TRUE
PID control	TRUE	TRUE	TRUE	FALSE	FALSE
PID control (P/D in feedback path)	TRUE	TRUE	TRUE	FALSE	TRUE

The proportional action is determined by the proportional gain GAIN and can be monitored at LMN_P.

AUTOHOTSPOT

The integral action is calculated using the following parameters.

AUTOHOTSPOT

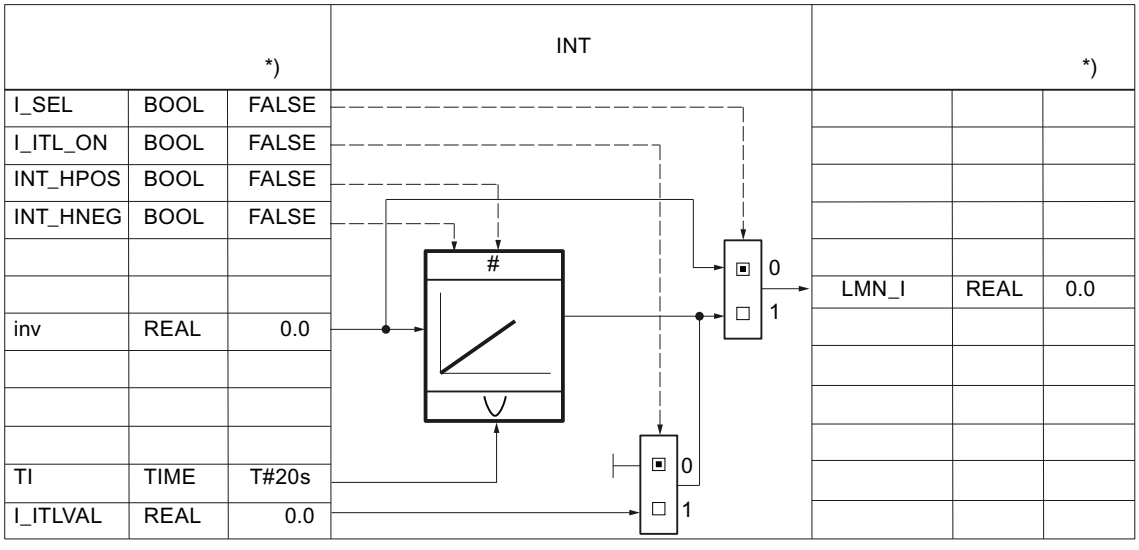

*) Default for creation of new technology object

The derivative action is calculated using the following parameters.

AUTOHOTSPOT

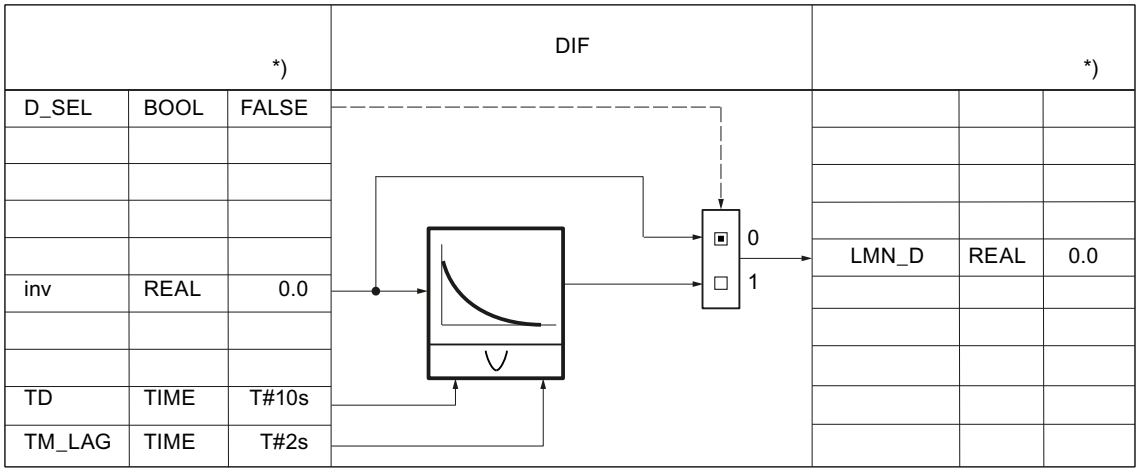

Manipulated variable PID_ES (S7-300, S7-400)

Scaling the position feedback

The position feedback in I/O format is converted to floating-point format. The position feedback in I/O format can be monitored at MP10.

The following table provides an overview of ranges and numerical values before and after processing.

I/O value LMNR_PER	LMNR in %
32767	118.515
27648	100.000
1	0.003617
0	0.000
-1	-0.003617
-27648	-100.000
-32768	-118.519

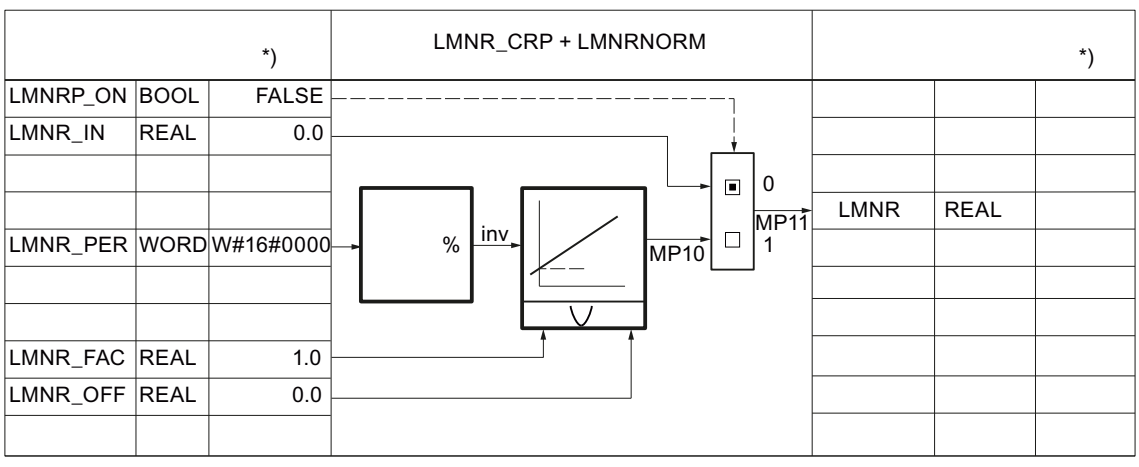

*) Default for creation of new technology object.

Manipulated value limit

The manipulated variable is always monitored with the following parameters for an absolute high and low limit. The implicit parameter can be monitored at MP9.

AUTOHOTSPOT

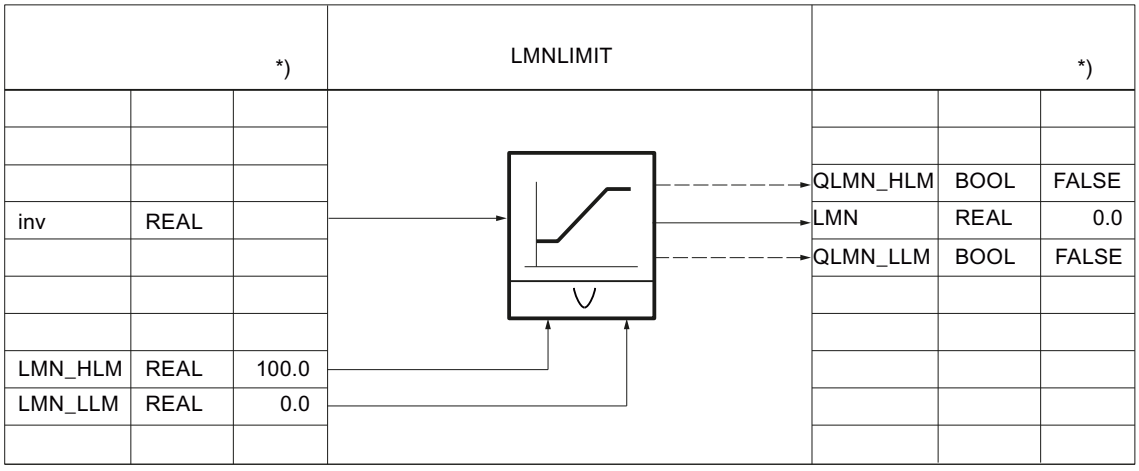

Step controller with position feedback

For step controllers with position feedback, the actuating signals are generated using the following parameters:

AUTOHOTSPOT

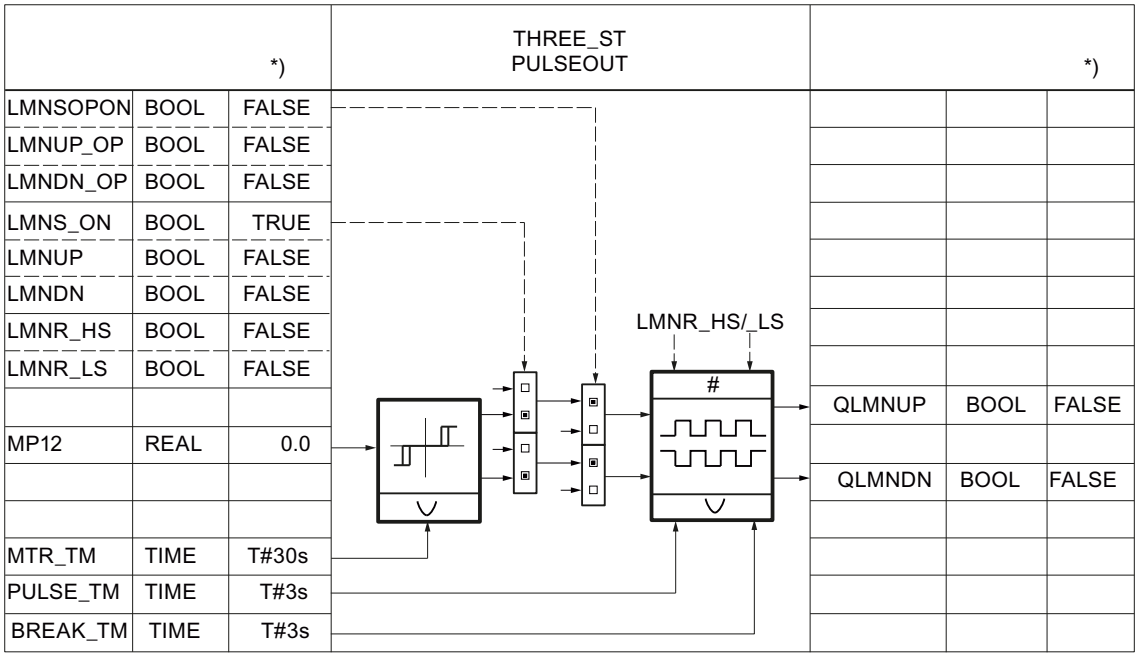

Step controller without position feedback

The position feedback can be simulated. The position feedback is simulated by an integrator with motor transition time MTR_TM as the integral action time constant. When LMNRS_ON = FALSE, the start value of the LMNRSVAL parameter is output at the output of the LMNR_SIM integrator. On switchover to LMNRS_ON = TRUE, the simulation starts up with this start value.

If LMNR_HS = TRUE, the integration is held up, and if LMNR_LS = TRUE, it is held down. The simulated position feedback is not adjusted to the endstop signals.

AUTOHOTSPOT

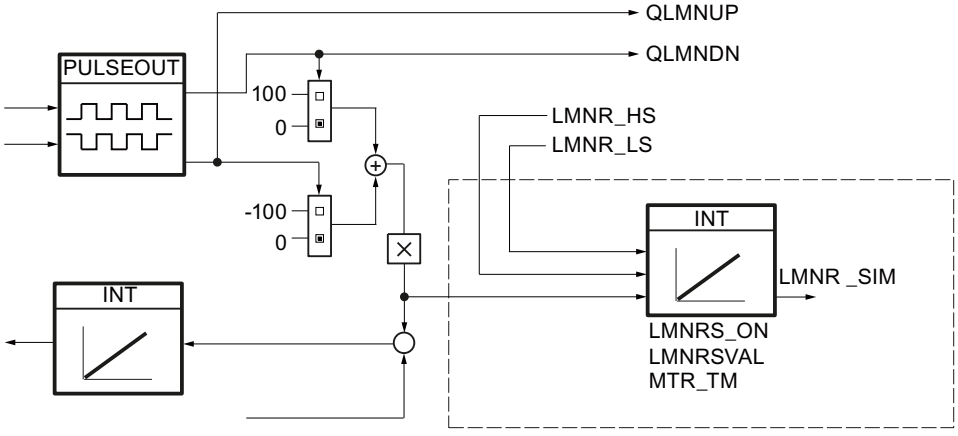

Note

The position feedback is merely simulated. It does not have to match the actual position of the actuator. If a real position feedback is available, it should always be used.

		*)
LMNSOPON	BOOL	FALSE
LMNUP_OP	BOOL	FALSE
LMNDN_OP	BOOL	FALSE
LMNS_ON	BOOL	TRUE
LMNUP	BOOL	FALSE
LMNDN	BOOL	FALSE
LMNR_HS	BOOL	FALSE
LMNR_LS	BOOL	FALSE
MP12	REAL	0.0
MTR_TM	TIME	T#30s
PULSE_TM	TIME	T#3s
BREAK_TM	TIME	T#3s

THREE_ST
PULSEOUT

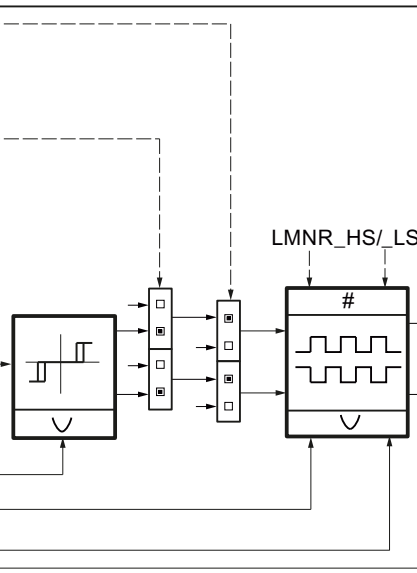

		*)
QLMNUP	BOOL	FALSE
QLMNDN	BOOL	FALSE

PID_ES in cascade arrangements (S7-300, S7-400)

Interruption of the controller cascade

In a cascade, several controllers are in direct relationship with each other. Therefore, you must take precautions to ensure problem-free cascade operation following an interruption of the cascade structure at any point.

In cascade controls, for this reason, a QCAS signal is generated in the secondary controllers by applying OR logic from the status signals of the switches in the setpoint and manipulated variable branches. This signal actuates a switch in the higher-level controllers that converts the controllers to follow-up mode. The follow-up value is always the setpoint SP of the lower-level circuit.

A smooth switchover occurs from follow-up mode to automatic mode, just as the switchover from manual to automatic mode.

Note

PID_ES can only be used in cascade arrangements as a follow-up controller in lower-level control loops.

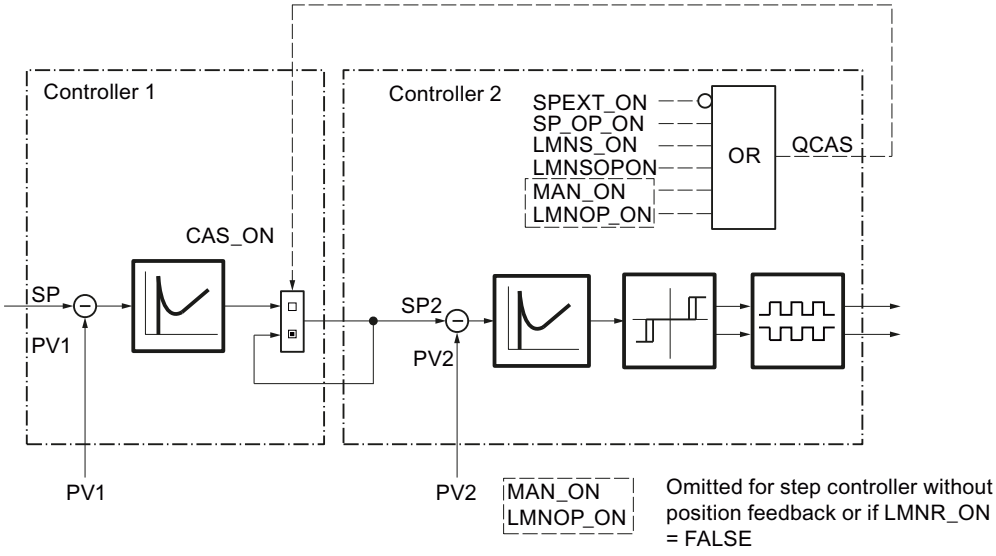

Block interconnection

The following figure shows the principle of the controller interconnection in multi-loop cascades.

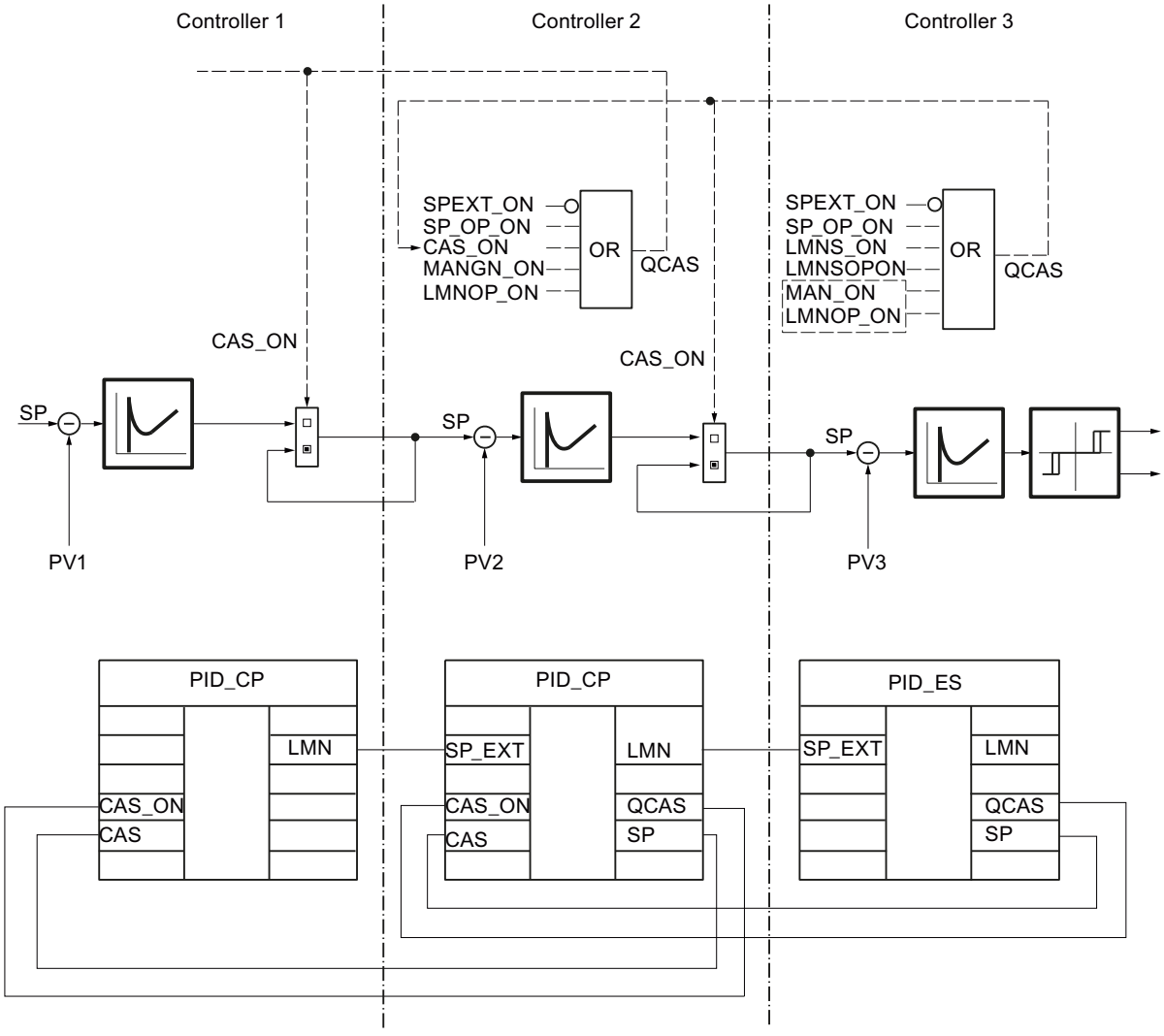

Input parameters PID_ES (S7-300, S7-400)

Table 4-221

Parameter	Offset	Data type	Range of values	Default	Description
COM_RST	0.0	BOOL		FALSE	Restart (Initialization routine of the instruction will be processed)
I_SEL	0.1	BOOL		TRUE	Integral action on
D_SEL	0.2	BOOL		FALSE	Derivative action on
MAN_ON	0.3	BOOL		TRUE	Manual mode on (control loop interrupted, LMN set manually)

Parameter	Offset	Data type	Range of values	Default	Description
LMNR_HS	0.4	BOOL		FALSE	High limit signal of position feedback
LMNR_LS	0.5	BOOL		FALSE	Low limit signal of position feedback
CYCLE	2.0	TIME	> 20 ms (S7–300)	T#1s	Sampling time (time between block calls = constant) Make sure to assign this parameter with the value of the cyclic interrupt cycle clock of the OB in which the PID_CP instruction runs. Otherwise, the time-dependent functions will not work correctly. (Exception: You are using time scaling, e.g., via the loop scheduler)
SP_INT	6.0	REAL	Engineering value range (physical quantity)	0.0	Internal setpoint SP_INT can be changed via a HMI with SPUP and SPDN
SP_EXT	10.0	REAL	Engineering value range (physical quantity)	0.0	External setpoint SP_EXT is used for blending, ratio and cascade controllers and can be scaled.
PV_IN	14.0	REAL	Engineering value range (physical quantity)	0.0	Process value input (PV in floating-point format)
PV_PER	18.0	INT		W#16#0000	Process value of I/O
GAIN	20.0	REAL	Overall value range (dimensionless)	2.0	Proportional gain (= controller gain)
TI	24.0	TIME	TI ≥ CYCLE	T#20s	Integral action time
TD	28.0	TIME	TD ≥ CYCLE	T#10s	Derivative-action time
TM_LAG	32.0	TIME	TM_LAG ≥ CYCLE/2	T#2s	Time delay of derivative action
DISV	36.0	REAL	-100.0 ... 100.0 [%]	0.0	Disturbance variable
LMNR_IN	40.0	REAL	0.0 ... 100.0 [%]	0.0	Position feedback (LMNR in floating-point format)
LMNR_PER	44.0	INT		W#16#0000	Position feedback of I/O (LMNR in I/O format)
SP_HLM	46.0	REAL	Engineering value range (physical quantity)	100.0	Setpoint high limit
SP_LLM	50.0	REAL	Engineering value range (physical quantity)	0.0	Setpoint low limit
LMN_HLM	54.0	REAL	LMN_LLM ... 100.0 [%]	100.0	Manipulated variable: High limit
LMN_LLM	58.0	REAL	0.0 ... LMN_HLM [%]	0.0	Manipulated variable: Low limit
DB_NBR	62.0	BLOCK_DB		DB 1	Data block number (DB with the time slices of the ramp/soak profile)

Parameter	Offset	Data type	Range of values	Default	Description
SPFC_NBR	64.0	BLOCK_FC		FC 0	Setpoint FC number (self-defined FC in setpoint branch)
PVFC_NBR	66.0	BLOCK_FC		FC 0	Process value FC number (self-defined FC in process value branch)

Output parameters PID_ES (S7-300, S7-400)

Table 4-222

Parameter	Offset	Data type	Preassigned value	Description
QLMNUP	68.0	BOOL	FALSE	Manipulated variable signal UP
QLMNDN	68.1	BOOL	FALSE	Manipulated variable signal DOWN
QCAS	68.2	BOOL	FALSE	Signal for cascade mode (interconnected to CAS_ON of higher-level controller)
LMN	70.0	REAL	0.0	Manipulated variable signal (according to control algorithm)
SP	74.0	REAL	0.0	Setpoint (effective setpoint)
PV	78.0	REAL	0.0	Process value (output of effective process value in cascade mode)

In/out parameters PID_ES (S7-300, S7-400)

Table 4-223

Parameter	Offset	Data type	Preassigned value	Description
MAN	82.0	REAL	0.0	Manual output value (for manipulated variable assignment by means of operator control and monitoring functions)

Static variables PID_ES (S7-300, S7-400)

Parameter	Offset	Data type	Range of values	Preassigned value	Description
PVH_ALM	86.0	REAL	PVH_WRN...100.0	100.0	Process value: High limit 'Alarm' Valid value range:
PVH_WRN	90.0	REAL	PVL_WRN... PVH_ALM	90.0	Process value: High limit 'Warning' Valid value range:
PVL_WRN	94.0	REAL	PVL_ALM... PVH_WRN	-90.0	Process value: Low limit 'Warning' Valid value range:
PVL_ALM	98.0	REAL	-100.0...PVL_WRN	-100.0	Process value: Low limit 'Alarm' Valid value range:

Parameter	Offset	Data type	Range of values	Preassigned value	Description
SPGEN_ON	102.0	BOOL		FALSE	Setpoint manipulation ON (adjusting the setpoint using the up/down switch)
SPUP	102.1	BOOL		FALSE	Setpoint UP
SPDN	102.2	BOOL		FALSE	Setpoint DOWN
RMPSK_ON	102.3	BOOL		FALSE	Ramp/soak function ON (setpoint is specified as a ramp/soak profile)
SPEXT_ON	102.4	BOOL		FALSE	External setpoint ON (for interconnection with other controller blocks)
MANGN_ON	102.5	BOOL		FALSE	Manual mode ON (LMN set via switch)
MANUP	102.6	BOOL		FALSE	Manual output value UP
MANDN	102.7	BOOL		FALSE	Manual output value DOWN
LMNS_ON	103.0	BOOL		FALSE	Manual mode for actuating signals ON
LMNUP	103.1	BOOL		FALSE	Manipulated variable signal UP (output signal QLMNUP will be manipulated)
LMNDN	103.2	BOOL		FALSE	Manipulated variable signal DOWN (output signal QLMNDN will be manipulated)
DFRMP_ON	103.3	BOOL		FALSE	Preassign output of ramp/soak function (output is preassigned with SP_INT)
CYC_ON	103.4	BOOL		FALSE	Cyclic repetition ON (ramp/soak profile will be repeated automatically)
RMP_HOLD	103.5	BOOL		FALSE	Ramp/soak function HOLD (output of ramp/soak function is frozen)
CONT_ON	103.6	BOOL		FALSE	Continue (with schedule) (ramp/soak profile will be continued at the next time slice)
TUPDT_ON	103.7	BOOL		FALSE	Recalculate total time (total time of ramp/soak profile will be recalculated)
SPFC_ON	104.0	BOOL		FALSE	Call setpoint FC
SPROC_ON	104.1	BOOL		FALSE	Ramp function ON (rate of change limiting of SP)
PVPER_ON	104.2	BOOL		FALSE	Process value of I/O ON (interconnection with I/O modules)
LAG1STON	104.3	BOOL		FALSE	First-order delay element ON
SQRT_ON	104.4	BOOL		FALSE	Square root function ON
PVFC_ON	104.5	BOOL		FALSE	Call process value FC
DEADB_ON	104.6	BOOL		FALSE	Deadband ON (small disturbances and noise will be filtered)
P_SEL	104.7	BOOL		TRUE	P component ON

Parameter	Offset	Data type	Range of values	Preassigned value	Description
PFDB_SEL	105.0	BOOL		FALSE	Switch P component to feedback
INT_HPOS	105.1	BOOL		FALSE	Freeze I component in positive direction
INT_HNEG	105.2	BOOL		FALSE	Freeze I component in negative direction
I_ITL_ON	105.3	BOOL		FALSE	Preassign I component
DFDB_SEL	105.4	BOOL		FALSE	Switch D component to feedback
DISV_SEL	105.5	BOOL		FALSE	Apply disturbance variable
LMNR_ON	105.6	BOOL		FALSE	Position feedback ON (operating modes: step controller with/without position feedback) Do not switch while in control mode!
LMNRP_ON	105.7	BOOL		FALSE	Position feedback of I/O ON
TM_SNBR	106.0	INT	≥ 0 (dimensionless)	0	No. of next time slice for continuation of the ramp/soak profile
TM_CONT	108.0	TIME	Overall value range (dimensionless)	T#0s	Time span until continuation of the ramp/soak profile (time that elapses before the ramp/soak function continues processing the ramp/soak profile at time slice TM_SNBR following an interruption)
FAC	112.0	REAL	Overall value range (dimensionless)	1.0	Factor (ratio or blending factor)
NM_SPEHR	116.0	REAL		100.0	Setpoint scaling: Input high
NM_SPELR	120.0	REAL		-100.0	Setpoint scaling: Input low
SPFC_OUT	124.0	REAL	-100.0 ... 100.0	0.0	Setpoint FC output (interconnected to the output of the FC in the setpoint branch)
SPURLM_P	128.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint up rate limit in positive range
SPDRLM_P	132.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint down rate limit in positive range
SPURLM_N	136.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint up rate limit in negative range
SPDRLM_N	140.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint down rate limit in negative range
NM_PIHR	144.0	REAL		100.0	Process value scaling: Input high
NM_PILR	148.0	REAL		-100.0	Process value scaling: Input low
NM_PVHR	152.0	REAL		100.0	Process value scaling: Output high
NM_PVLR	156.0	REAL		-100.0	Process value scaling: Output low
PV_TMLAG	160.0	TIME	Overall value range	T#5s	Process value time delay (delay of PT1 element in the PV branch)
SQRT_HR	164.0	REAL		100.0	Square root: Measuring range output high

Parameter	Offset	Data type	Range of values	Preassigned value	Description
SQRT_LR	168.0	REAL		0.0	Square root: Measuring range output low
PVFC_OUT	172.0	REAL	-100.0 ... 100.0	0.0	Process value FC output (interconnected to the output of the FC in the process value branch)
PVURLM_P	176.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value up rate limit in the positive range
PVDRLM_P	180.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value down rate limit in the positive range
PVURLM_N	184.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value up rate limit in the negative range
PVDRLM_N	188.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value down rate limit in the negative range
PV_HYS	192.0	REAL	≥ 0	1.0	Process value hysteresis (prevents 'flickering' of monitoring display)
DEADB_W	196.0	REAL	0.0 ... 100.0	1.0	Deadband width (determines the size of the deadband area)
ERP_ALM	200.0	REAL	0 ... 200.0	100.0	Control deviation: Positive limit 'Alarm'
ERP_WRN	204.0	REAL	0 ... 200.0	90.0	Control deviation: Positive limit 'Warning'
ERN_WRN	208.0	REAL	-200.0 ... 0	-90.0	Control deviation: Negative limit 'Warning'
ERN_ALM	212.0	REAL	-200.0 ... 0	-100.0	Control deviation: Negative limit 'Alarm'
ER_HYS	216.0	REAL	≥0	1.0	Control deviation hysteresis (prevents 'flickering' of monitoring display)
I_ITLVAL	220.0	REAL	-100.0 ... 100.0 [%]	0.0	Initialization value for I action
LMNR_FAC	224.0	REAL	Overall value range (dimensionless)	1.0	Position feedback factor (factor for adaptation of the feedback range)
LMNR_OFF	228.0	REAL	-100.0 ... 100.0 [%]	0.0	Position feedback offset (zero point of feedback scaling)
PULSE_TM	232.0	TIME	= n * CYCLE / n=0,1,2...	T#3s	Minimum pulse time
BREAK_TM	236.0	TIME	= n * CYCLE / n=0,1,2...	T#3s	Minimum break time
MTR_TM	240.0	TIME	≥ CYCLE	T#30s	Motor actuating time
PHASE	244.0	INT		0	Phase of PID self tuner
QPVH_ALM	246.0	BOOL		FALSE	Process value: High limit 'Alarm' triggered
QPVH_WRN	246.1	BOOL		FALSE	Process value: High limit 'Warning' triggered
QPVL_WRN	246.2	BOOL		FALSE	Process value: Low limit 'Warning' triggered
QPVL_ALM	246.3	BOOL		FALSE	Process value: Low limit 'Alarm' triggered
QR_S_ACT	246.4	BOOL		FALSE	Time table for ramp/soak profile will be processed
QSP_HLM	246.5	BOOL		FALSE	Setpoint: High limit triggered

Parameter	Offset	Data type	Range of values	Preassigned value	Description
QSP_LLM	246.6	BOOL		FALSE	Setpoint: Low limit triggered
QPVURLMP	246.7	BOOL		FALSE	Process value: Up rate limit in positive range triggered
QPVDRLMP	247.0	BOOL		FALSE	Process value: Down rate limit in positive range triggered
QPVURLMN	247.1	BOOL		FALSE	Process value: Up rate limit in negative range triggered
QPVDRLMN	247.2	BOOL		FALSE	Process value: Down rate limit in negative range triggered
QERP_ALM	247.3	BOOL		FALSE	Control deviation: Positive limit 'Alarm' triggered
QERP_WRN	247.4	BOOL		FALSE	Control deviation: Positive limit 'Warning' triggered
QERN_WRN	247.5	BOOL		FALSE	Control deviation: Negative limit 'Warning' triggered
QERN_ALM	247.6	BOOL		FALSE	Control deviation: Negative limit 'Alarm' triggered
QLMN_HLM	247.7	BOOL		FALSE	Manipulated variable: High limit triggered
QLMN_LLM	248.0	BOOL		FALSE	Manipulated variable: Low limit triggered
NBR_ATMS	250.0	INT		0	Number of the time slice currently being approached by the ramp/soak function
RS_TM	252.0	TIME		T#0s	"Current residual time" of ramp/soak profile until the next time slice
T_TM	256.0	TIME		T#0s	"Total time" = elapsed time of ramp/soak profile
RT_TM	260.0	TIME		T#0s	"Total residual time" = time remaining until the end of the ramp/soak profile
ER	264.0	REAL		0.0	Control deviation
LMN_P	268.0	REAL		0.0	P component
LMN_I	272.0	REAL		0.0	I component
LMN_D	276.0	REAL		0.0	D component
SPFC_IN	280.0	REAL		0.0	Setpoint FC input (interconnected to input of the self-defined FC)
PVFC_IN	284.0	REAL		0.0	Process value FC input (interconnected to input of the self-defined FC)
SP_OP_ON	288.0	BOOL		FALSE	Setpoint manipulation ON (value of SP_OP will be applied as the setpoint)
PV_OP_ON	288.1	BOOL		FALSE	Process value manipulation ON (value of PV_OP will be applied as the setpoint)
LMNOP_ON	288.2	BOOL		FALSE	Manipulated variable manipulation ON (value of LMN_OP will be applied as the setpoint)

Parameter	Offset	Data type	Range of values	Preassigned value	Description
LMNSO-PON	288.3	BOOL		FALSE	Manipulated variable signal manipulation ON (LMNUP_OP and LMNDN_OP will be applied as actuating signals)
LMNUP_OP	288.4	BOOL		FALSE	Manipulated variable signal UP
LMNDN_OP	288.5	BOOL		FALSE	Manipulated variable signal DOWN
LMNRS_ON	288.6	BOOL		FALSE	Simulation of the position feedback ON
SP_OP	290.0	REAL		0.0	Setpoint manipulation in the configuration tool
PV_OP	294.0	REAL		0.0	Process value manipulation in the configuration tool
LMN_OP	298.0	REAL		0.0	Manipulated variable manipulation in the configuration tool
LMNRSVAL	302.0	REAL		0.0	Start value of the simulated position feedback
LMNR_SIM	306.0	REAL		0.0	Current value of the simulated position feedback
MP1	310.0	REAL		0.0	Measuring point 1: Internal setpoint
MP2	314.0	REAL		0.0	Measuring point 2: External setpoint
MP3	318.0	REAL		0.0	Measuring point 3: Unlimited setpoint
MP4	322.0	REAL		0.0	Measuring point 4: Process value of the I/O module
MP5	326.0	REAL		0.0	Measuring point 5: Process value after first-order delay element
MP6	330.0	REAL		0.0	Measuring point 6: Effective process value (PV)
MP7	334.0	REAL		0.0	Measuring point 7: Manipulated variable of PID algorithm
MP8	338.0	REAL		0.0	Measuring point 8: Manual output value
MP9	342.0	REAL		0.0	Measuring point 9: Unlimited manipulated variable
MP10	346.0	REAL		0.0	Measuring point 10: Position feedback I/O
MP11	350.0	REAL		0.0	Measuring point 11: Feedback value (LMNR_ON = FALSE) Position feedback (LMNR_ON = TRUE)
MP12	354.0	REAL		0.0	Measuring point 12: Three-step element input

Block diagrams for PID_ES (S7-300, S7-400)

Signal flow diagrams

Analogous to the representation of the switches in the configuration tool, the black dot in each switch symbol indicates that the switching signal has the adjacent Boolean value (0=FALSE or 1=TRUE) and that the signal path is switched through this point in each case. The switching signals (binary signals) are identified by dashed lines.

This means that the setpoint is specified as an absolute setpoint via SP_INT; the same applies to the process value input via PV_IN. The default control function is a standard PI controller with P function in the forward branch. The control loop is open and the percentage range of the manipulated variable is influenced by the MAN input. All other functions are passive or, if they cannot be deactivated, marginal signal characteristic parameters are preassigned within the measuring range or operating range so that the functions have no effect.

Symbols and identifiers in the signal flow graphs

The names of the connectable process variables are shown on a shaded background. This allows you to recognize where the controller structure can be connected to the S7 I/O or directly to the measuring elements and actuators of the process.

Parameter names containing the combination of letters "OP" (for example, SP_OP/ SP_OP_ON) indicate that an intervention using the Standard PID Control configuration tool is possible at this point. The configuration tool has its own interface to the controller FB.

Intermediate variables in the signal characteristic can be monitored at the measuring points MP1 to MP12 marked with circles. These intermediate variables are required in order to match values before triggering "smooth" changeovers or to check the current statuses of the relevant control. The measuring point variables in the configuration tool curve recorder can be represented (statically and dynamically).

For reasons of clarity, the parameters for setting and sizing the processing operations (algorithms) are also indicated in the individual function blocks. Please refer to the descriptions in the reference section and to the representation of the individual subfunctions in the following sections.

Processing the setpoint

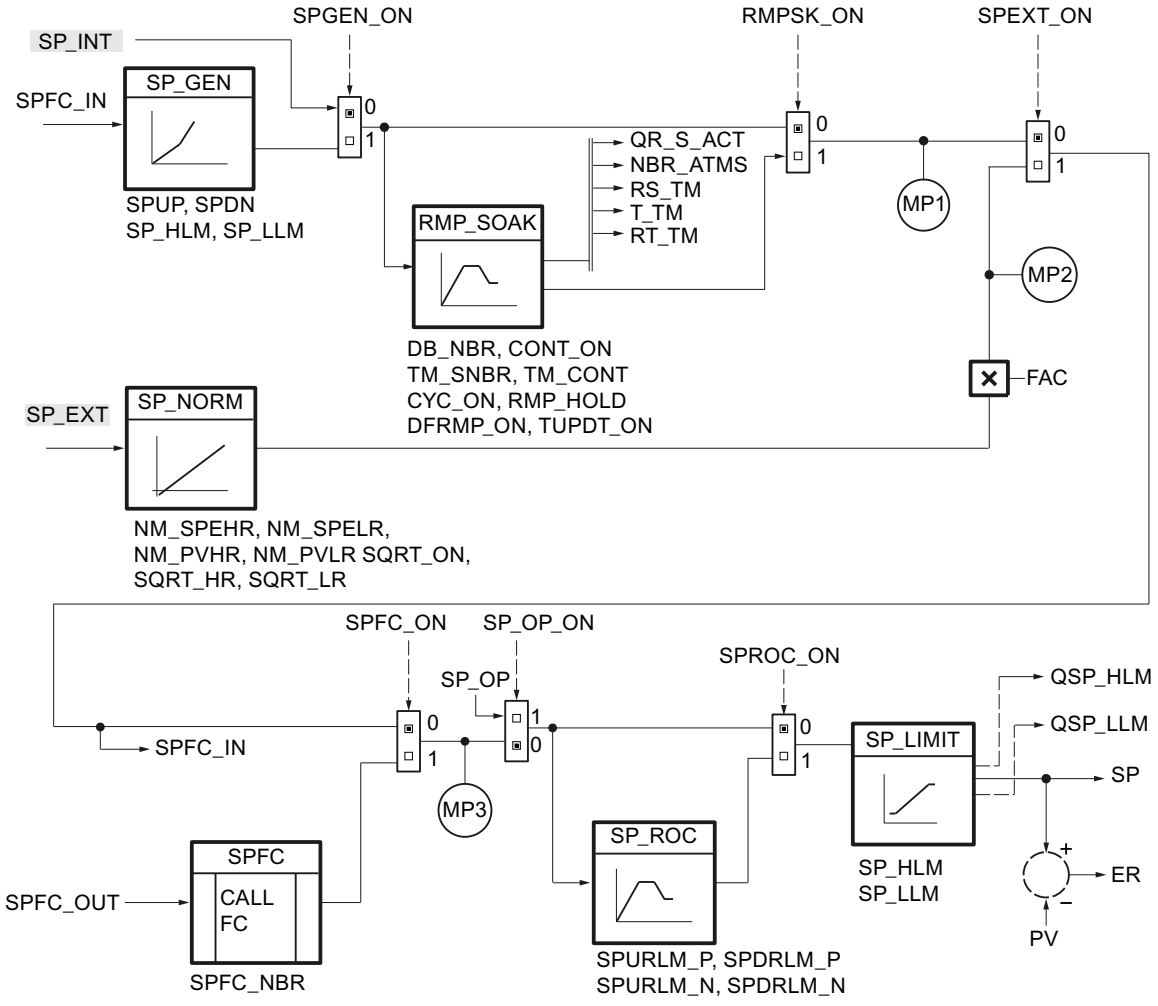

Processing the process value

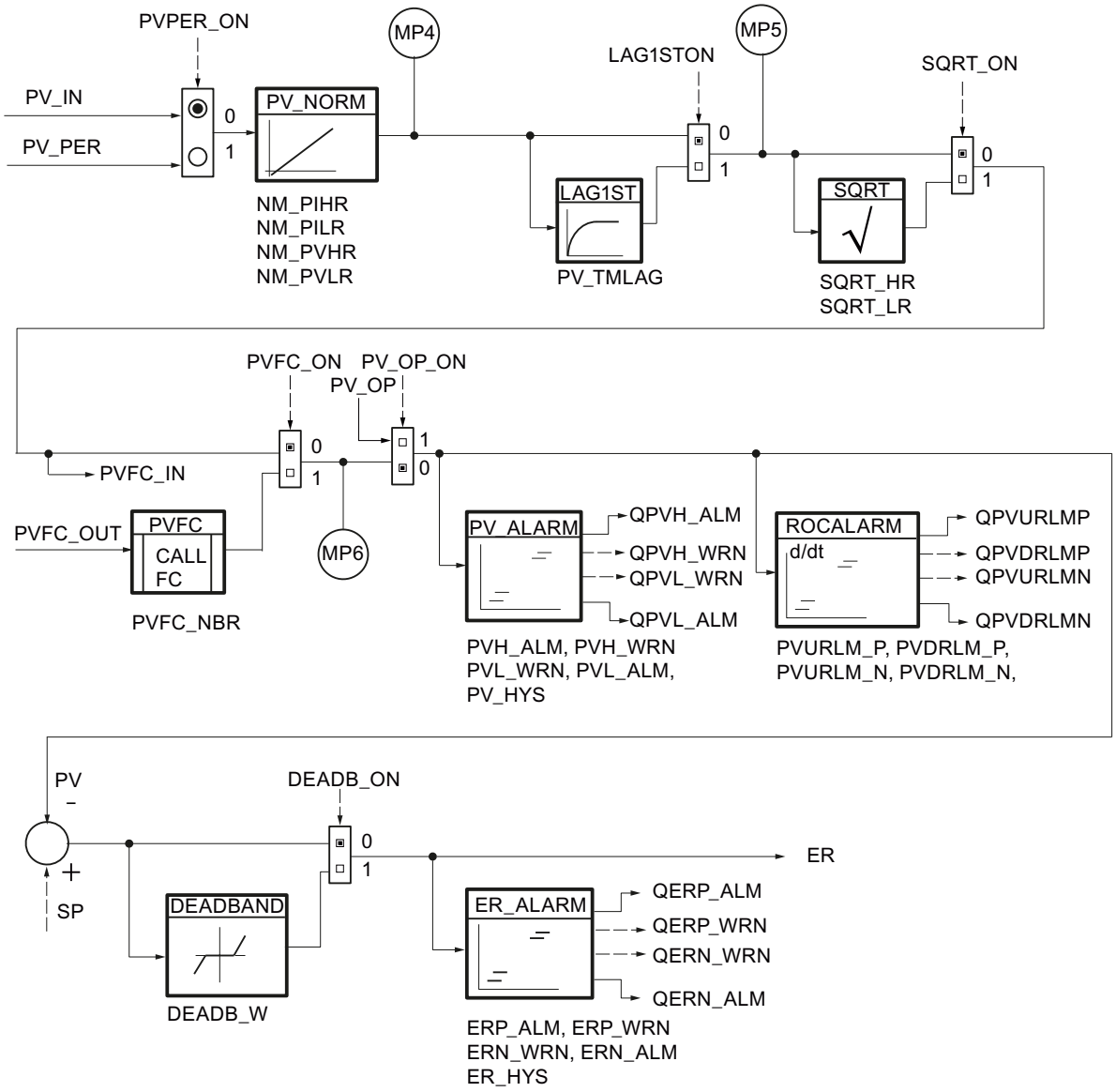

Control algorithm

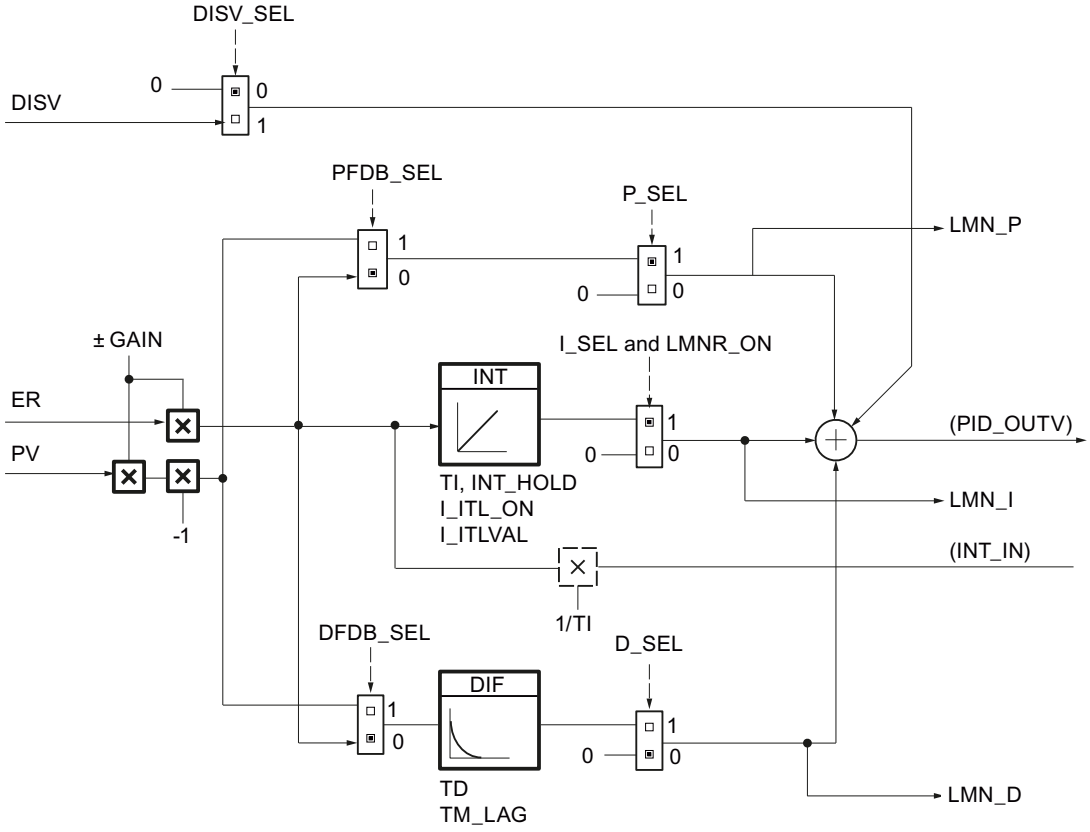

Step controller without position feedback (LMNR_ON = FALSE)

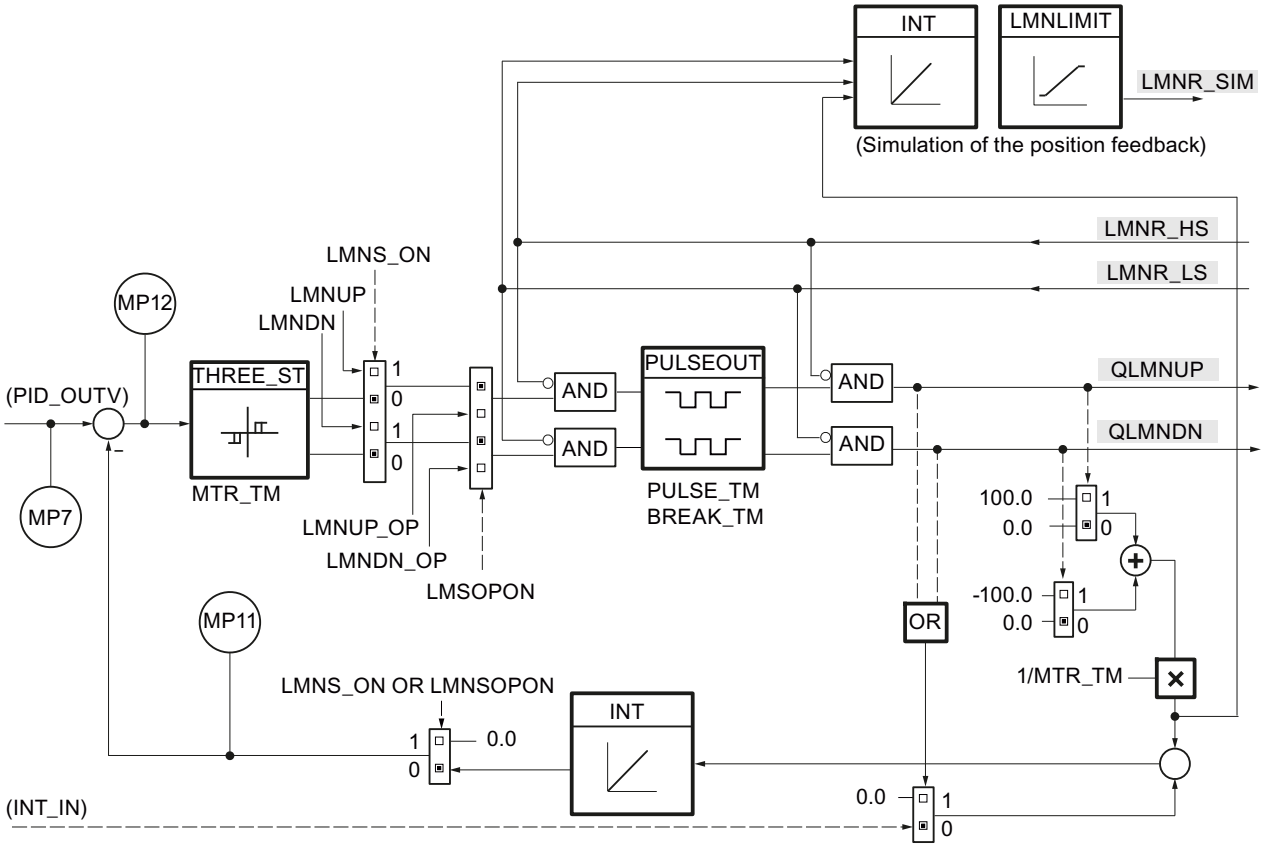

Step controller with position feedback (LMNR_ON = TRUE)

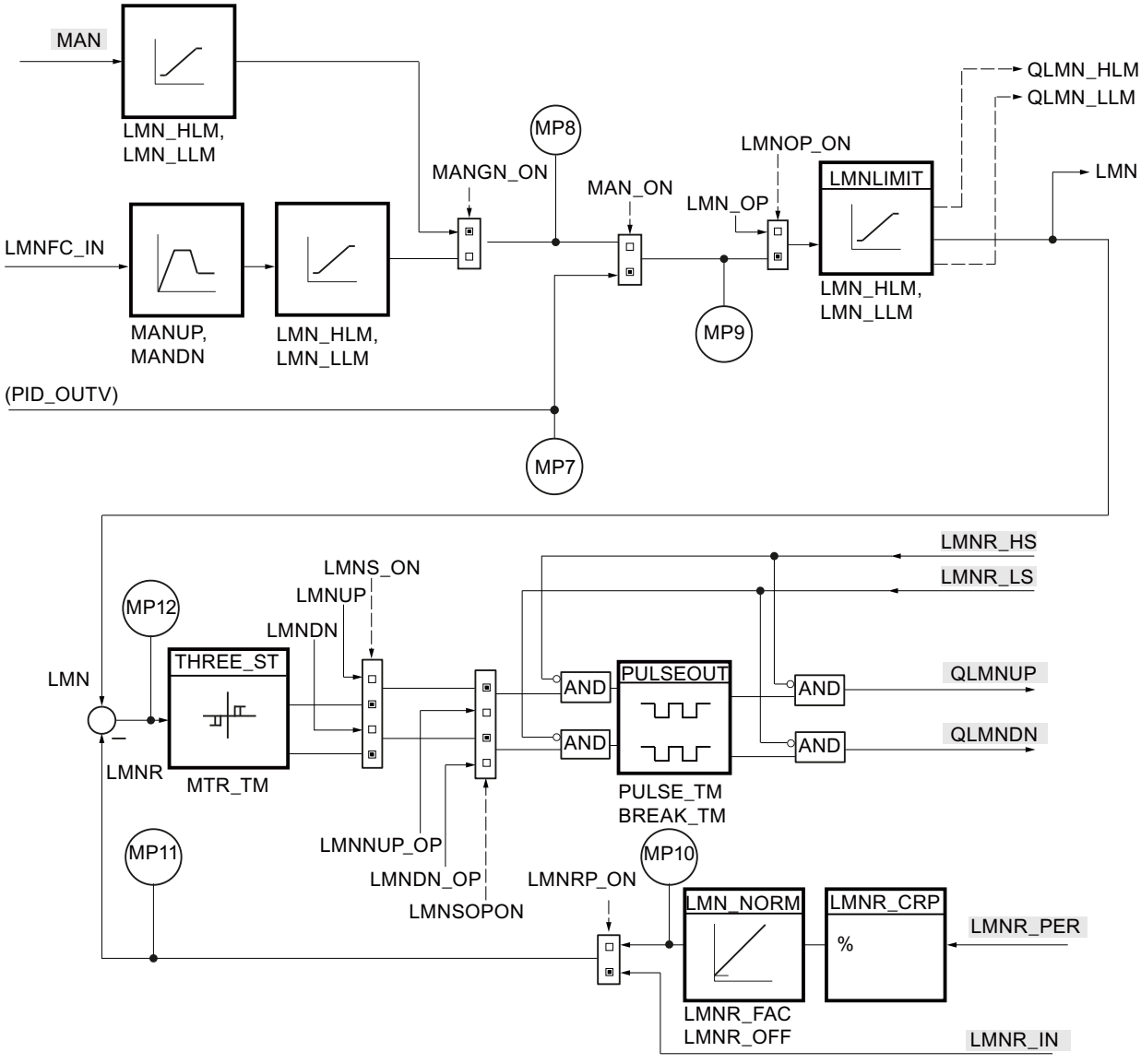

Global data block DB_RMPSK (S7-300, S7-400)

Note

Creating a global data block yourself

The time slices for the ramp/soak function (RMP_SOAK) are specified in a global data block. This global data block is not included in the library. You must create the global data block yourself according to the following schema and adapt it to your application.

Parameter	Data type	Default	Description
NBR_PTS	INT	4	Number of time slices -1 1 - 255
PI	Array [0...NBR_PTS] of Struct		Structure for the time slice data. The array must be declared by 0...NBR_PTS.
PI[0].OUTV	REAL	0.0	Output variable [0] Starting point Engineering value range
PI[0].TMV	TIME	T#1s	Time value [0] Starting point
PI[1].OUTV	REAL	0.0	Output variable [1] Time slice 1 Engineering value range
PI[1].TMV	TIME	T#1s	Time value [1] Time slice 1
PI[2].OUTV	REAL	0.0	Output variable [2] Time slice 2 Engineering value range
PI[2].TMV	TIME	T#1s	Time value [2] Time slice 2
PI[3].OUTV	REAL	0.0	Output variable [3] Time slice 3 Engineering value range
PI[3].TMV	TIME	T#1s	Time value [3] Time slice 3
PI[4].OUTV	REAL	0.0	Output variable [4] Time slice 4 Engineering value range
PI[4].TMV	TIME	T#0s	Time value [4] Time slice 4 0 ms

Generating a data block using an external source file

Alternatively, you can generate the global data block using an external source file.

Proceed as follows:

1. Copy the text below to the clipboard.
2. Open an external text editor.
3. Paste the copied text from the clipboard to the text editor.
4. Save the file with the file name extension "DB":
5. Open the "External sources" folder in the project tree of the TIA Portal.
6. Double-click the command "Add new external file".
   The "Open" dialog is displayed.
7. Navigate to the external source file that you have created and select it.
8. Confirm your selection with "Open".
9. Select the external source file.
10. Select the command "Generate blocks from source" in the shortcut menu.
11. A security prompt informs you that any existing blocks will be overwritten.

12. Click "OK" in response to the security prompt.
13. Adapt the created data block to your application.

```
DATA_BLOCK "DB_RMPSK"
TITLE = data block ramp soak
{ S7_Optimized_Access := 'FALSE' }
NAME : DB_RMPSK
   STRUCT
      NBR_PTS : Int;              // number of points (excluding
                                  starting point)
      PI : Array[0..4] of Struct
         OUTV : Real;             // output variable
         TMV : Time;              // output time value
      END_STRUCT;
   END_STRUCT;


BEGIN
   NBR_PTS := 4;
   PI[0].TMV := T#1s;
   PI[1].TMV := T#1s;
   PI[2].TMV := T#1s;
   PI[3].TMV := T#1s;
   PI[4].TMV := T#0s;
END_DATA_BLOCK
```

LP_SCHED (S7-300, S7-400)

Description of LP_SCHED (S7-300, S7-400)

LP_SCHED is used when the number of cyclic interrupts of a CPU is not sufficient to achieve the desired (different) sampling times. It enables up to 256 control loops to be called with sampling times equal to an integer multiple of the cyclic interrupt cycle.

LP_SCHED reads the parameters you assigned from the global data block DB_LOOP, calculates from these parameters the tags necessary for call scheduling, and stores these tags in the DB_LOOP data block again.

Global data block for DB_LOOP (Page 5716) is not included in the library. You must create this global data block yourself.

Note

The block does not check to determine whether a global DB with the number DB_NBR actually exists or whether the GLP_NBR parameter (maximum control loop number) adapts to the length of the data block. If the parameter assignment is incorrect, the CPU changes to STOP with an "internal system error".

Call

You must call LP_SCHED in a cyclic interrupt OB prior to all controllers.

Note the following when assigning values to the input parameters:

- TM_BASE: At this input, specify the cycle clock of the cyclic interrupt OB in which the LP_SCHED instruction is called.
- COM_RST: At CPU startup, you must call LP_SCHED once with COM_RST = TRUE. During cyclic operation (cyclic interrupt), you must call LP_SCHED with COM_RST = FALSE.
- DB_NBR: At this input, you specify the number of the "DB_LOOP" data block that LP_SCHED is to access.

After the LP_SCHED instruction is called, you must call all associated controllers conditionally. You cannot use LP_SCHED to call the controllers PID_CP or PID_ES, as their input and output parameters must be assigned when the instructions are called.

The following is an example of calling the LP_SCHED instruction and of conditionally calling a controller.

STL

```
CALL "LP_SCHED"
    TM_BASE:=                                   Here, the cycle clock of the cyclic interrupt
                                                is parameterized. Example: T#100ms or #CYCLE
                                                where CYCLE = input parameter of the block in
                                                which LP_SCHED instruction is called.
    COM_RST:=                                   Here, the LP_SCHED instruction is notified
                                                whether or not an initialization pass of the
                                                called control loop is to take place. Exam-
                                                ple: FALSE or #COM_RST where COM_RST = input
                                                parameter of the block in which the LP_SCHED
                                                instruction is called.
    DB_NBR:=                                    Here, the number of the "DB_LOOP" DB that the
                                                LP_SCHED instruction is to process is called.
                                                Example: "DB_LOOP" where DB_LOOP = name of DB
                                                assigned in the symbol table.
A "DB_LOOP".LOOP_DAT[1].ENABLE
SPBN M002                                       Control loop call, if ENABLE = TRUE
//CALL FBx,DBy
  //COM_RST:= "DB_LOOP".LOOP_DAT[1].COM_RST
```

```
STL
      //CYCLE:= "DB_LOOP".LOOP_DAT[1].CYCLE
    CLR
    = "DB_LOOP".LOOP_DAT[1].ENABLE                    Reset ENABLE bit
M002:                                                 Continuing in the program, e.g., conditional
                                                      call of the next control loop FB
```

Startup

When COM_RST = TRUE, the following pre-assignments are set:

Current control loop number: ALP_NBR = 0

The call data of all control loops up to **GLB_NBR** is pre-assigned as follows:

Enable:	ENABLE =	NOT MAN_DIS
Sampling time:	CYCLE =	GV(MAN_CYC)
Restart:	COM_RST =	TRUE
Local call number:	ILP_COU =	0
GV(MAN_CYC) = MAN_CYC is rounded to an integer multiple of TM_BASE*GLP_NBR		

During CPU startup, you must call LP_SCHED from the associated startup OB and, at the same time, assign the value TRUE to the COM_RST input. Then call PID_CP or PID_ES conditionally to initialize the controllers.

In the cyclic interrupt OB, you must assign the value FALSE to COM_RST again.

Operating principle of LP_SCHED (S7-300, S7-400)

Call processing

A controller should be processed if the respective ENABLE bit in the DB_LOOP data block has the value TRUE. This bit was written previously by LP_SCHED. If the controller was processed, you must assign the value FALSE to the ENABLE bit after processing is complete.

When the controller is called, you must interconnect its input parameters COM_RST and CYCLE with the variables COM_RST[x] and CYCLE[x] of the DB_LOOP data block. CYCLE[x] contains the actual sampling time of the controller x and is written by LP_SCHED during each cycle.

During operation, you can manually disable the call of individual controllers and also reset individual controllers.

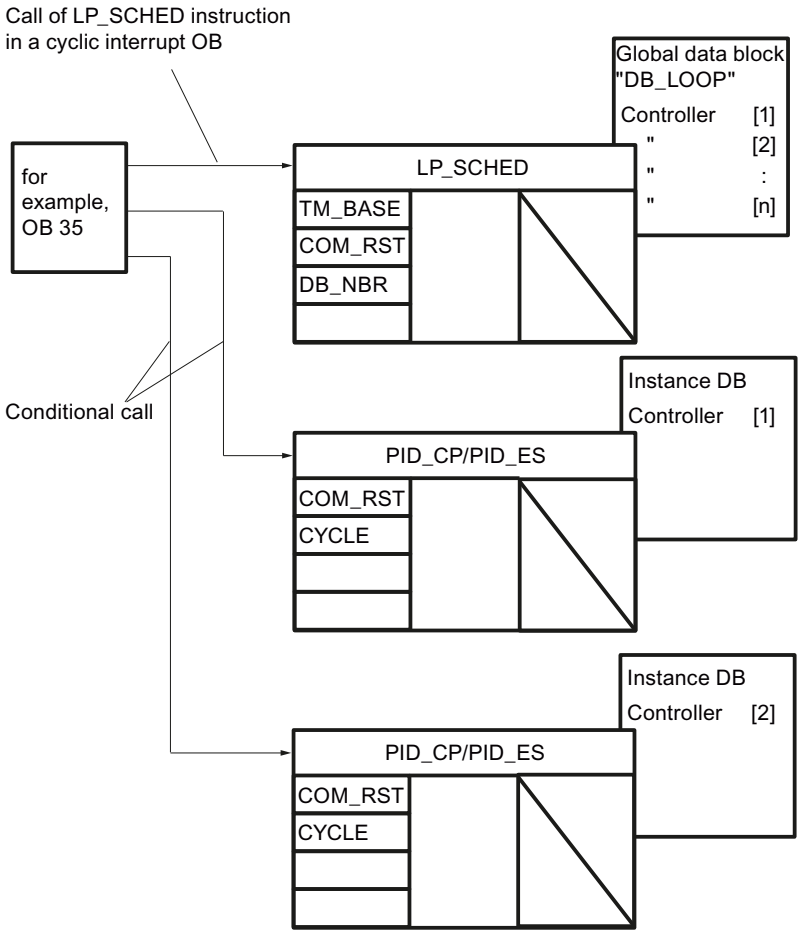

Parameter assignment of the DB_LOOP data block

The parameters of the controller call distributor must be assigned in the DB editor.

You must assign parameters to the following variables in the DB_LOOP data block:

- GLP_NBR: Number of controllers whose call is managed by LP_SCHED (max. 256)
- MAN_CYC[x], x = 1 ... GLP_NBR: The desired sampling time for the individual controllers. Note here that the condition indicated below for MAN_CYC[x] applies to each controller. Otherwise, the assigned sampling time cannot be guaranteed.

The following variables can change during operation:

- MAN_DIS[x] is used to disable the call of control loop x during operation.
- MAN_CRST[x] is used to initiate an initialization pass for control loop x during operation. The change becomes effective with the next call of the instruction LP_SCHED.

The following variables are written by LP_SCHED:

- LP_SCHED enters the call condition for the controller x in the variable ENABLE[x].
- The ALP_NBR and ILP_COU[x] variables are internal variables of the LP_SCHED instruction. You can use them for monitoring.

The COM_RST[x] and CYCLE[x] variables are interconnected with the input parameters COM_RST and CYCLE of the instructions PID_CP and PID_ES.

Sampling time MAN_CYC[x]

LP_SCHED can process a maximum of one controller per call. For this reason, the time

TM_BASE * GLP_NBR, until all controllers have been processed for the first time. Therefore, when you assign parameters for the desired sampling times MAN_CYC[x], you must adhere to the following condition for each controller:

The sampling time of controller x must be an integer multiple of the product obtained from the time base and the number of controllers to be processed.

$$\text{MAN_CYC[x]} \overset{!}{=} \text{GV (TM_BASE * GLP_NBR), x = 1, ... GLP_NBR}$$

The actual sampling time CYCLE[x] of the control loop x is determined as follows by LP_SCHED in each cycle from MAN_CYC[x]:

- If you adhere to this rule, the actual sampling time CYCLE[x] will be identical to the sampling time MAN_CYC[x] you specified previously.
- If you do not adhere to this rule, CYCLE[x] will contain the value that results if MAN_CYC[x] is rounded down to the next integer multiple of TM_BASE * GLP_NBR.

Adding additional control loops

If you want to insert one or more control loops into the DB_LOOP data block, increase the start value for GLP_NBR and expand the LOOP_DAT[1...GLP_NBR] array.

Check the desired sampling time MAN_CYC[x] for each controller. At the same time, note the condition for MAN_CYC.

Pulse generator in conjunction with LP_SCHED

If you have activated the pulse generator for the continuous controller PID_CP, the pulse pattern width CYCLE_P with the LOOP_DAT[x].CYCLE parameter must be interconnected instead of the CYCLE parameter.

Example of a loop schedule

The following example shows the call order of four controllers in a cyclic interrupt OB. A maximum of one controller is processed per unit of the time base TM_BASE. The call order is based on the order of the controllers in the DB_LOOP data block.

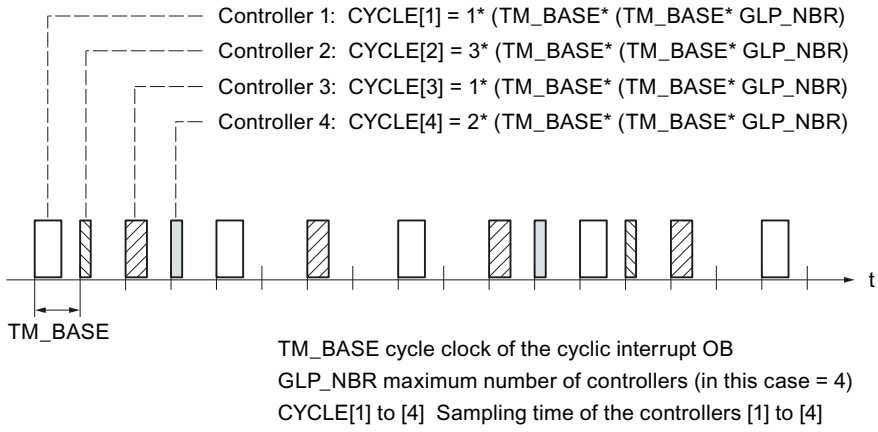

Call of more than one controller per cycle

If more than one controller is to be processed in a cycle, LP_SCHED may then be called multiple times. All calls of the LP_SCHED instruction must be made before calling the controller. In the input parameter TM_BASE, you must then enter the cycle clock of the cyclic interrupt OB, divided by the number of LP_SCHED calls.

Example: LP_SCHED will be called in OB 35 twice; OB 35 will be processed every 100 ms. The input parameter TM_BASE will be configured with 50 ms.

Execution times

The sum of the runtimes of the LP_SCHED instruction and the controller which can be processed in one cycle must not be greater than the cycle clock of the cyclic interrupt OB.

Manipulations during operation

The following changes may be made in the "DB_LOOP" data block during operation, provided that only the respective parameter is changed and the complete data block not downloaded to the CPU:

- Disabling individual controllers
  Writing the TRUE value to the MAN_DIS[x] variable disables the processing of controller x during operation. LP_SCHED does not set the ENABLE bit of this controller to TRUE until you write the value FALSE to MAN_DIS[x] again.
- Initializing a control loop
  You can restart an individual controllers by writing the value TRUE to the MAN_CRST[x] variable: In this case, LP_SCHED writes the value TRUE to the COM_RST[x] variable the next time controller x is processed. During the next subsequent processing of this control loop, LP_SCHED writes the value FALSE to the MAN_CRST[x] and COM_RST[x] variables.
- Change in the sampling time of a control loop
  You can change the MAN_CYC[x] parameter during operation:

  Note

  If the entire DB_LOOP data block is downloaded again to the CPU without requiring a CPU startup, you must pre-assign the internal control loop counters ILP_COU[x], x = 1, ... GLP_NBR and the number of the current control loop ALP_NBR with zero.

Monitoring the instruction LP_SCHED

LP_SCHED enters the number of the controller that is to be processed next in the ALP_NBR variable of the DB_LOOP data block. The number of the respective control loop results from the positioning of its call data in the sequence of the entries in the data block.

The ILP_COU[x] variable is the internal counter. It receives the time period until the next call of the corresponding controller. The time unit of ILP_COU is the product obtained from the time base TM_BASE and the number of controllers GLP_NBR. If ILP_COU = 0, LP_SCHED sets the ENABLE bit of the corresponding controller.

When control loops are not called contrary to expectation

If LP_SCHED is called but individual control loops are not processed, this can be caused by the following:

- Writing the value TRUE to the MAN_DIS[x] variable disables the processing of control loops during operation.
- The number of controllers to be processed by LP_SCHED specified in the GLP_NBR parameter is too low.
- The sampling times MAN_CYC[x] of the individual control loops that you have specified must not be less than the product obtained from the time base TM_BASE and the number of controllers GLP_NBR. Controllers in which this condition is not met will not be processed.

Input parameters LP_SCHED (S7-300, S7-400)

Table 4-224

Parameter	Data type	Range of values	Preassigned value	Explanation
TM_BASE	TIME	≥ 20 ms (S7-300) ≥ 5 ms (S7-400)	100 ms	Time base (Cycle clock of cyclic interrupt level in which LP_SCHED is called) Permitted values are
COM_RST	BOOL		FALSE	Complete restart (Complete restart routine of LP_SCHED will be processed)
DB_NBR	BLOCK_DB		DB 1	Data block number (DB with the call data of the control loop)

Global data block for DB_LOOP (S7-300, S7-400)

The data for the call distribution with LP_SCHED is specified in a global data block. This global data block is not included in the library. You must create the global data block yourself according to the following schema and adapt it to your application.

Parameter	Data type	Default	Description
GLP_NBR	INT	2	Maximum number of controllers 1 - 256
ALP_NBR	INT	0	Number of current controllers 0 - 256
LOOP_DAT	Array [1...GLP_NBR] of Struct		Structure for the data of the individual controllers. The array must be declared by 1...GLP_NBR.
LOOP_DAT[1].MAN_CYC	TIME	T#1s	Data controller 1: Manual sampling time >= 20 ms (S7-300) >= 5 ms (S7-400)
LOOP_DAT[1].MAN_DIS	BOOL	FALSE	Data controller 1: Disable manual controller call
LOOP_DAT[1].MAN_CRST	BOOL	FALSE	Data controller 1: Set manual restart
LOOP_DAT[1].ENABLE	BOOL	FALSE	Data controller 1: Controller enable
LOOP_DAT[1].COM_RST	BOOL	FALSE	Data controller 1: Restart
LOOP_DAT[1].ILP_COU	INT	0	Data controller 1: Internal counter
LOOP_DAT[1].CYCLE	TIME	T#1s	Data controller 1: Sampling time >= 20 ms (S7-300) >= 5 ms (S7-400)
LOOP_DAT[2].MAN_CYC			Data controller 2: Manual sampling time
...			

Alternatively, you can generate the global data block using an external source file. Proceed as follows:

1. Copy the text below to the clipboard.
2. Open an external text editor.

3. Paste the copied text from the clipboard to the text editor.
4. Save the file with the file name extension "DB":
5. Open the "External sources" folder in the project tree of the TIA Portal.
6. Double-click the command "Add new external file".
   The "Open" dialog is displayed.
7. Navigate to the external source file that you have created and select it.
8. Confirm your selection with "Open".
9. Select the external source file.
10. Select the command "Generate blocks from source" in the shortcut menu.
11. A security prompt informs you that any existing blocks will be overwritten.
12. Click "OK" in response to the security prompt.
13. Adapt the created data block to your application.

```
DATA_BLOCK "DB_LOOP"
TITLE = data block loop scheduler
{ S7_Optimized_Access := 'FALSE' }
NAME : DB_LOOP
    STRUCT
        GLP_NBR : Int;                  // greatest loop number
        ALP_NBR : Int;                  // actual loop number
        LOOP_DAT : Array[1..2] of Struct
            MAN_CYC : Time;             // loop data: manual sample time
            MAN_DIS : Bool;             // loop data: manual loop disable
            MAN_CRST : Bool;            // loop data: manual complete restart
            ENABLE : Bool;              // loop data: enable loop
            COM_RST : Bool;             // loop data: complete restart
            ILP_COU : Int;              // loop data: internal loop counter
            CYCLE : Time;               // loop data: sample time
        END_STRUCT;
    END_STRUCT;


BEGIN
    GLP_NBR := 2;
    LOOP_DAT[1].MAN_CYC := T#1S;
    LOOP_DAT[1].CYCLE := T#1S;
    LOOP_DAT[2].MAN_CYC := T#200MS;
    LOOP_DAT[2].CYCLE := T#200MS;
END_DATA_BLOCK
```

Modular PID Control (S7-300, S7-400)

General (S7-300, S7-400)

Conventions for parameter and block names in the block diagram

Not more than 8 characters were selected for the parameter designations.

The following conventions apply for the designation of the parameters:

Initial character:	
Q	General output of the BOOL type (Boolean variable)
SP	Reference variable
PV	Measured value or controlled variable
LMN	Manipulated variable or analog output signal to be output
DISV	Disturbance variable
Subsequent characters:	
MAN	Manual value
INT	Internal
EXT	External
_ON	Boolean variable for activating a function

Call data

Most Modular PID Control instructions require control-loop-specific call data such as the restart bit and sampling time. These values are transferred to the inputs COM_RST and CYCLE.

Remarks on the block parameter bars (input, output, and in/out parameters)

- **Preassigned value:** The preassigned values for the creation of an instance are displayed here.
- **Range of values**: The values at the input parameter should not exceed the permitted range of values. The range is **not** checked in the course of the block execution. If a range of engineering values is specified, a physical variable whose value should lie between approx. $\pm 10^{6}$ is meant.

A_DEAD_B (S7-300, S7-400)

Description A_DEAD_B (S7-300, S7-400)

If the process value is affected by noise and the controller is optimally set, the noise also affects the controller output. This results in increased wear of the actuator through a high frequency of operation (step controllers). Oscillation of the controller output is avoided by suppression of the noise component.

Operating principle

The instruction filters high-frequency interfering signals from the control deviation and produces a deadband around the zero point. If the input variable lies within this range, zero is output at the output. The range width is adapted automatically to the amplitude of the interfering signal.

The instruction operates in accordance with the function:

OUTV = INV + DB_WIDTH	for	INV < -DB_WIDTH
OUTV = 0.0	for	-DB_WIDTH ≤ INV ≤ +DB_WIDTH
OUTV = INV - DB_WIDTH	for	INV > +DB_WIDTH

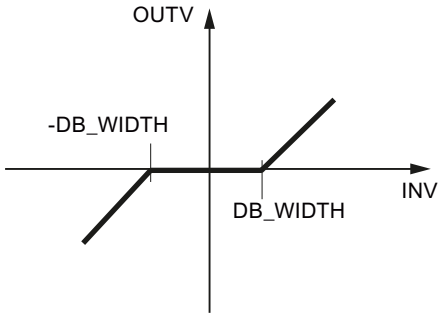

Adaptation of the deadband

The adaptation of the deadband can be activated and deactivated.

- Adaptation off
  If the adaptation is deactivated (ADAPT_ON = FALSE), the last DB_WIDTH value is frozen as the effective deadband width DB_WIDTH.
- Adaptation on
  With ADAPT_ON = TRUE, an adaptation algorithm that calculates the effective deadband width can be activated. The effect is that the deadband width is adapted to the amplitude of the noise signal that is superimposed on the input variable, so that the noise component is also suppressed at a fluctuating noise amplitude.
  In the case of an acyclic block call, the adaptation has to be deactivated (ADAP_ON = FALSE).

If adaptation is activated during commissioning and a stable deadband width is set after a certain time, the adaptation can be deactivated. The deadband width set through the adaptation remains in force as long as a complete restart is not carried out.

Adaptation on

To ensure stability, the effective deadband width DB_WIDTH is subject to a low limit, which is represented by the DB_WL_LM value that can be specified at the input bar. If input signal INV superimposed by the measurement noise exceeds the currently specified deadband width in the negative (1), positive (2), and again negative direction (3) within the period 1/CRIT_FRQ, the effective deadband width is increased by the amount 0.1.

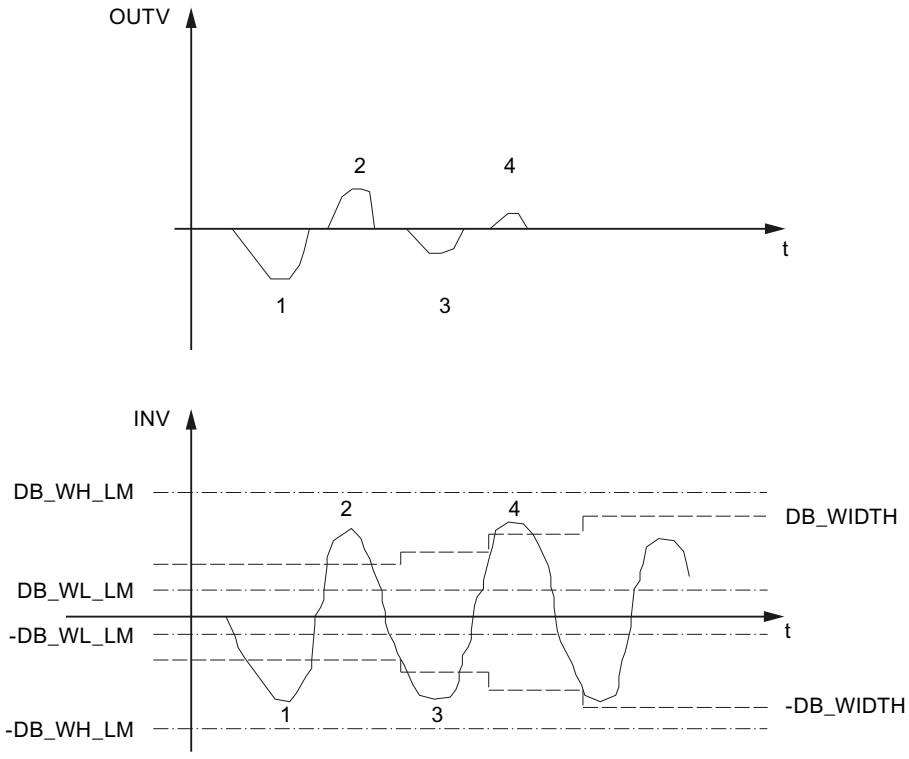

The procedure is started if the limit is exceeded positively or negatively. Whenever the opposing deadband width is exceeded again (3 -> 4) within half the cycle duration, it is increased again by 0.1. This procedure is repeated until the deadband width agrees with the amplitude of the measurement noise. In order not to indiscriminately suppress input signals of any size, the effective deadband width is subject to a high limit, which is represented by input DB_WH_LM. If, on the other hand, the deadband width is not exceeded within time RET_FAC * 1/CRIT_FRQ, it is reduced by the amount 0.1.

CRIT_FRQ specifies the limit frequency from which a signal component is regarded as a disturbance. It is subject to a high limit and low limit as follows:

0.01 ≤ CRIT_FRQ ≤ 1/(3*CYCLE) where CYCLE: Sampling time in sec

The RET_FAC parameter specifies the multiple of 1/CRIT_FRQ based on which the deadband width is reduced again.

The adaptation logic only functions if the input variable without disturbance component lies near zero.

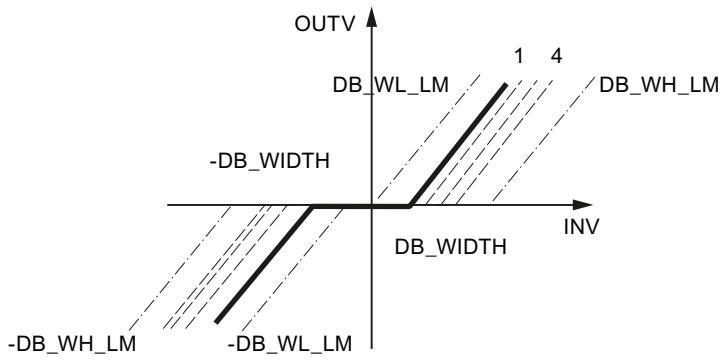

Complete restart

At a complete restart, OUTV=0.0 is set, and the effective deadband width DB_WIDTH = DB_WL_LM is set

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters A_DEAD_B (S7-300, S7-400)

Parameter	Data type	Preassigned value	Description
INV	REAL	0.0	Input variable Engineering value range
DB_WH_LM	REAL	5.0	High limit of the deadband width Engineering value range DB_WH_LM > DB_WL_LM
DB_WL_LM	REAL	1.0	Low limit of the deadband width Engineering value range DB_WL_LM < DB_WH_LM
CRIT_FRQ	REAL	0.1	Limit frequency CRIT_FRQ ≥ 0.01 and CRIT_FRQ ≤ 1/(3 * CYCLE)
RET_FAC	INT	1	Return factor RET_FAC ≥ 1
ADAPT_ON	BOOL	FALSE	Activate adaptation algorithm
COM_RST	BOOL	FALSE	Restart
CYCLE	TIME	T#1s	Sampling time CYCLE ≥ 1ms

Output parameters A_DEAD_B (S7-300, S7-400)

Parameter	Data type	Preassigned value	Description
OUTV	REAL	0.0	Output variable
DB_WIDTH	REAL	0.0	Effective deadband width

CRP_IN (S7-300, S7-400)

Description CRP_IN (S7-300, S7-400)

This instruction adapts the range of values of the analog I/O to the internal representation of the modular control. The instruction can be called in the process value branch, for example.

Operating principle

CRP_IN converts an input value in I/O format into a scaled floating-point value of the modular control.

I/O value	Output value as a %
32767	118.515
27648	100.000
1	0.003617
0	0.000
-1	-0.003617
-27648	-100.000
-32768	-118.519

The floating-point value can be adapted by means of a scaling factor and an offset. The output results from:

OUTV = INV_PER * 100/27648 * FACTOR + OFFSET

Note

A check for positive/negative overflow is not performed.

Using a commissioning value

It is possible to switch over to a commissioning value for purposes of commissioning or testing or in case of I/O faults. If START_ON = TRUE, the value in STARTVAL is output at output OUTV.

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters CRP_IN (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV_PER	0.0	WORD	0	Input variable for I/O Engineering value range
FACTOR	2.0	REAL	1.0	Scaling factor
OFFSET	6.0	REAL	0.0	Offset Engineering value range
START_ON	10.0	BOOL	TRUE	Activate commissioning value
STARTVAL	12.0	REAL	0.0	Commissioning value Engineering value range

Output parameters CRP_IN (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	16.0	REAL	0.0	Output variable

CPR_OUT (S7-300, S7-400)

Description CRP_OUT (S7-300, S7-400)

This instruction adapts a floating-point value of the modular control to the I/O format.

Operating principle

CRP_OUT converts an input value (scaled floating-point value of the modular control) into the I/O format of the analog I/O.

Input value in %	I/O value
118.515	32767
100.000	27648
0.003617	1

Input value in %	I/O value
0.000	0
-0.003617	-1
-100.000	-27648
-118.519	-32768

The floating-point value can be adapted by means of a scaling factor and an offset. The output results from:

OUTV_PER = (INV * FACTOR + OFFSET) * 27648/100

Note

A check for positive/negative overflow is not performed.

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters CRP_OUT (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
FACTOR	4.0	REAL	1.0	Scaling factor
OFFSET	8.0	REAL	0.0	Offset Engineering value range

Output parameters CRP_OUT (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV_PER	12.0	WORD	0	Output variable for I/O

DEAD_T (S7-300, S7-400)

Description DEAD_T (S7-300, S7-400)

This instruction is used in ratio control systems in which the individual components on transport routes of different lengths are brought together.

Operating principle

The instruction delays the output of an input value by an assignable time (dead time). In the case of online changes in the dead time steps can occur at the output value,

The input values are stored temporarily in a global data block.

Global data block DB_DEADT (Page 5727) is not included in the library. You must create this global data block yourself.

The global data block must contain an Array [0..n_{max}] of real.

n_{max} is determined by the dead time DEAD_TM and sampling time CYCLE of the DEAD_T. instruction. The following applies:

- DEAD_TM must be an integer multiple of CYCLE.
- n_{max}=DEAD_TM/CYCLE

In cycle n, the saved input value INV[n] is read from the global data block and output to OUTV. INV[n] is overwritten with the current input value. Counter n is increased by 1. When n = n_{max} , the counter is reset to 0. You specify the number of the global block in input parameter DB_NBR.

Example

At a sampling time of CYCLE=1s and a dead time of DEAD_TM=5s, 5 input values must be buffered. The data area must have a length of 20 bytes.

The data in the global data block are processed like in a ring buffer.

Cycle no.	Counter n	INV of instruction	Tag in global data block	Value of INVI[n] at start of cycle	OUTV of instruction	Value of INVI [n] at end of cycle
0	0	80.0	INVI[0]	0.0	0.0	80.0
1	1	100.0	INVI[1]	0.0	0.0	100.0
2	2	100.0	INVI[2]	0.0	0.0	100.0
3	3	80.0	INVI[3]	0.0	0.0	80.0
4	4	0.0	INVI[4]	0.0	0.0	0.0
5	0	0.0	INVI[0]	80.0	80.0	0.0
6	1	0.0	INVI[1]	100.0	100.0	0.0
7	2	0.0	INVI[2]	100.0	100.0	0.0
8	3	10.0	INVI[3]	80.0	80.0	10.0
9	4	100.0	INVI[4]	0.0	0.0	100.0

The value of INV in cycle 0 is not output at OUTVuntil cycle 5. With a sampling time of 1 s, the output is thus delayed by the dead time of 5 s.

Tracking

When TRACK=TRUE, the input value is output at OUTV without a delay. Buffering of the input values is not interrupted so that, after tracking has been deactivated, the input values can continue to be output after the set dead time. On a falling edge at TRACK, OUTV jumps to INV[DEAD_TM]!

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Note

The instruction does not check whether a global DB with the number DB_NBR actually exists or whether the parameters DEAD_TM (dead time) and CYCLE (sampling time) are suitable for the data block length. If the parameter assignment is incorrect, the CPU changes to the STOP operating mode with the message "Internal system error".

Restart

During a restart, all saved input values are deleted and OUTV=0.0 is output.

Input parameters DEAD_T (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
DB_NBR	4.0	BLOCK_DB	DB 1	Data block number
DEAD_TM	6.0	TIME	10s	Dead time DEAD_TM ≥ CYCLE
TRACK	10.0	BOOL	FALSE	Tracking OUTV = INV
COM_RST	10.1	BOOL	FALSE	Complete restart
CYCLE	12.0	TIME	1s	Sampling time CYCLE ≥ 1ms

Output parameters DEAD_T (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	16.0	REAL	0.0	Output variable

Global data block DB_DEADT (S7-300, S7-400)

The input values for the dead time element DEAD_T are stored temporarily in a global data block. This global data block is not included in the library. You must create the global data block yourself according to the following schema and adapt it to your application.

Parameter	Data type	Default	Description
INVI	Array[0..n_{max}] of real	0.0	Input value [n] Engineering value range

Alternatively, you can generate the global data block using an external source file. Proceed as follows:

1. Copy the text below to the clipboard.
2. Open an external text editor.
3. Paste the copied text from the clipboard to the text editor.
4. Save the file with the file name extension "DB":
5. Open the "External sources" folder in the project tree of the TIA Portal.
6. Double-click the command "Add new external file". The "Open" dialog is displayed.
7. Navigate to the external source file that you have created and select it.
8. Confirm your selection with "Open".
9. Select the external source file.
10. Select the command "Generate blocks from source" in the shortcut menu.
11. A security prompt informs you that any existing blocks will be overwritten.
12. Click "OK" in response to the security prompt.
13. Adapt the created data block to your application.

```
DATA_BLOCK "DB_DEADT"
TITLE = data block dead time
{ S7_Optimized_Access := 'FALSE' }
NAME : DB_DEADT
    STRUCT
        INVI : Array[0..4] of Real;
    END_STRUCT;
BEGIN
END_DATA_BLOCK
```

DEADBAND (S7-300, S7-400)

Description DEADBAND (S7-300, S7-400)

If the process value is affected by noise and the controller is optimally set, the noise also affects the controller output. This results in increased wear of the actuator through a high frequency of operation (step controllers). Oscillation of the controller output is avoided by suppression of the noise component.

Operating principle

The DEADBAND instruction suppresses minor fluctuations of input variable INV around a defined zero point. Outside these fluctuations, output variable OUTV rises proportionally to the input variable. It operates in accordance with the function:

OUTV = INV + DEADB_W - DEADB_O	for	INV < DEADB_O - DEADB_W
OUTV = 0.0	for	DEADB_O - DEADB_W ≤ INV
	and	INV ≤ DEADB_O + DEADB_W
OUTV = INV - DEADB_W - DEADB_O	for	INV > DEADB_O + DEADB_W

The signal corruption corresponds to the value DEADB_W. The center of the deadband is defined with DEADB_O.

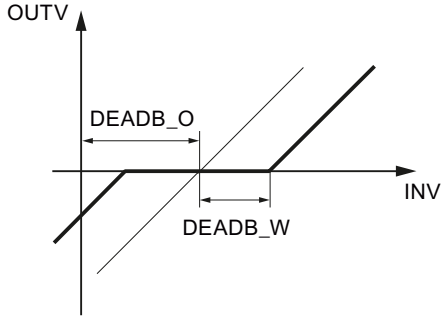

Deadband width DEADB_W and deadband offset DEADB_O can be assigned. When used for generating the control deviation, offset DEADB_O=0.0 must be set.

The following figure shows the bypassing of a noise component with offset

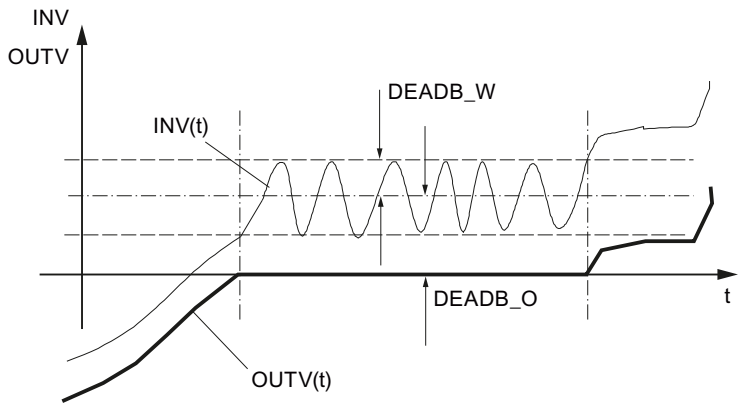

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check. Only positive values are permitted for the deadband width.

Input parameters DEADBAND (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
DEADB_W	4.0	REAL	1.0	Deadband width Engineering value range and ≥ 0.0
DEADB_O	8.0	REAL	0.0	Deadband offset Engineering value range

Output parameters DEADBAND (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	12.0	REAL	0.0	Output variable

DIF (S7-300, S7-400)

Description DIF (S7-300, S7-400)

Process variables are differentiated dynamically. Thus, for example, the velocity can be calculated from the distance traversed. The differentiator can be used for disturbance variable compensation, for feedforward control, and to structure a controller.

Operating principle

This instruction differentiates the input variable with respect to time and smoothes the signal with a first-order delay element. It operates in the Laplace domain according the following transfer function:

- OUTV(s) / INV(s) = TD / (1+TM_LAG*s)

The time response is defined by the derivative-action time TD and the time delay TM_LAG. The following figures show the step response of DIF.

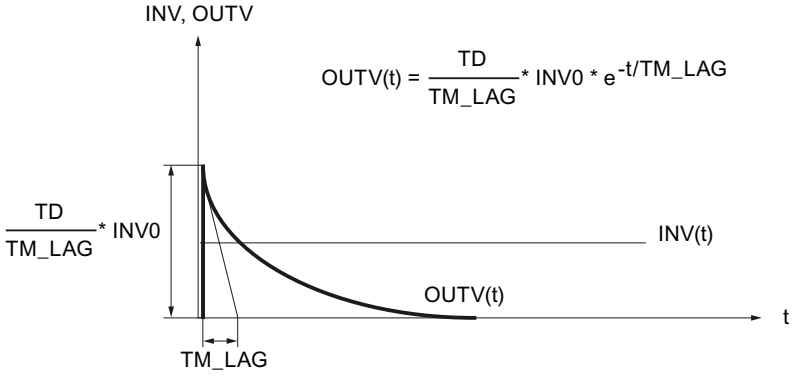

TD	Derivative-action time
TM_LAG	Time delay
INV0	Input step height
INV	Input variable
OUTV	Output variable

If TM_LAG ≤CYCLE/2 is assigned, DIF works without a delay. An input step-change is applied to the output with the factor TD/CYCLE. After one cycle the output returns to 0.0.

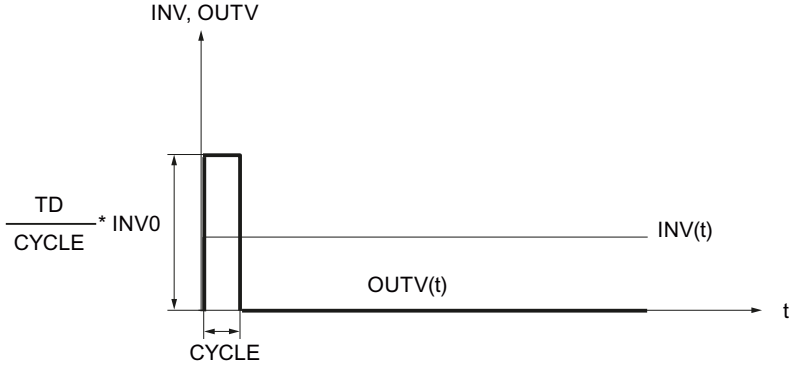

Block-internal limits

The derivative-action time is subject to a low limit equal to the sampling time. The time delay is subject to a low limit equal to half the sampling time.

TD_{intern} = CYCLE	for	TD < CYCLE
TM_LAG_{intern} = CYCLE/2	for	TM_LAG < CYCLE/2

The other values of the input parameters are not limited in the instruction. There is no parameter check.

Complete restart

During a complete restart, all the signal outputs are set to 0. The differentiator is preassigned internally with the current input value INV. The transition to normal operation is thus smooth at a constant input variable.

Input parameters DIF (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
TD	4.0	TIME	T#25s	Derivative-action time ≥ CYCLE
TM_LAG	8.0	TIME	T#5s	Time delay
COM_RST	12.0	BOOL	FALSE	Complete restart
CYCLE	14.0	TIME	T#1s	Sampling time ≥ 1ms

Output parameters DIF (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	18.0	REAL	0.0	Output variable

ERR_MON (S7-300, S7-400)

Description ERR_MON (S7-300, S7-400)

This instruction is used for control deviation generation and monitoring.

Operating principle

The instruction calculates control deviation ER = SP - PV and monitors it for assignable limits. As a result of a setpoint change |ΔSP| > SP_DIFF, the response of limit value signal ER_LM is suppressed for an assignable delay time (TM_DELAY+TM_RAMP). During this time, ER is monitored for the higher limit value ER_LMTD. If ER_LMTD is exceeded, QER_LMTD=TRUE is output. After the delay time has expired, ER_LMTD is transformed into ER_LM along a ramp. The slope of the ramp can be assigned in the TM_RAMP parameter.

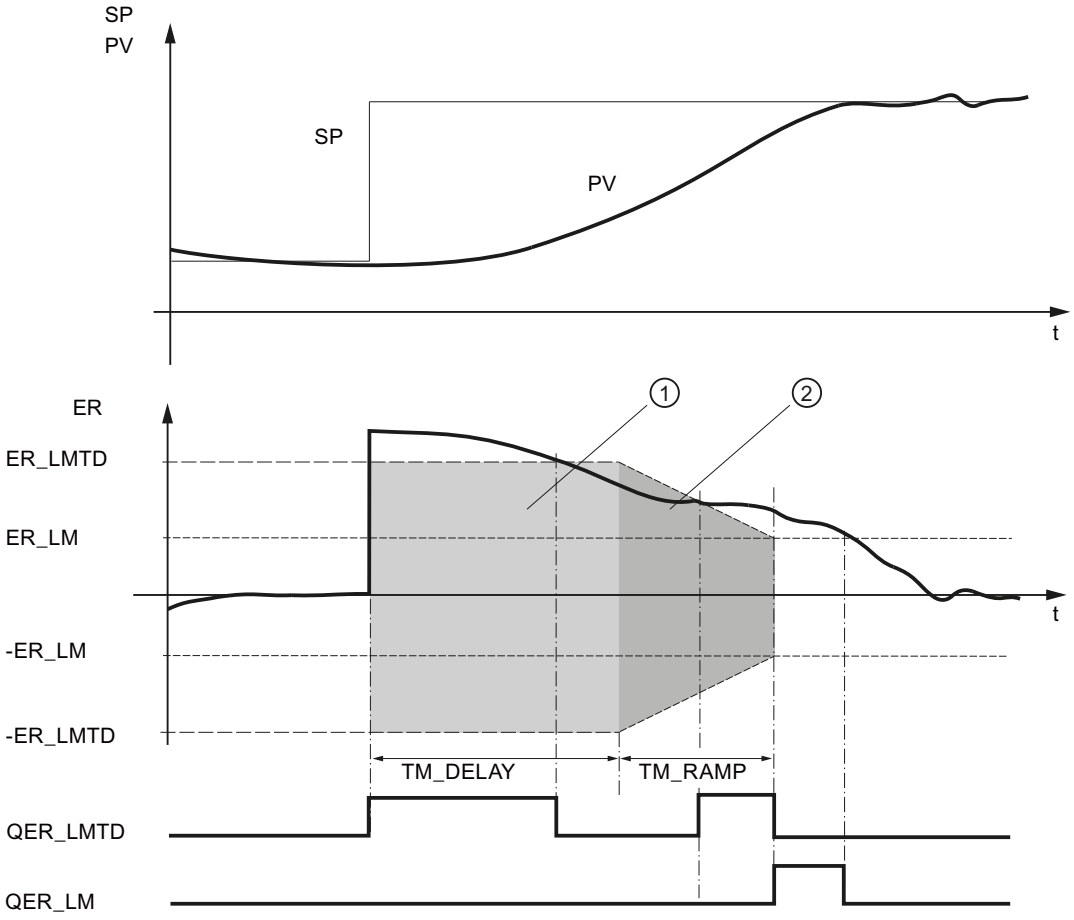

PV:	Process value	TM_DELAY:	On delay of the monitoring signal
SP:	Setpoint		
ER:	Control deviation	TM_RAMP:	Ramp time constant
ER_LM:	Limit value of the control deviation	QER_LM:	Limit value of the control deviation triggered
ER_LMTD:	Limit value of the control deviation during On delay	QER_LMTD:	Limit value of the control deviation during On delay and ramp triggered
①	Suppression of the limit value signal	②	Ramp function

Complete restart

During a complete restart, alarm signals QER_LM and QER_LMTD and control deviation output ER are reset.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters ERR_MON (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
ER_LM	0.0	REAL	10.0	Limit value of the control deviation Engineering value range and > 0.0 and < ER_LMTD
ER_LMTD	4.0	REAL	100.0	Limit value of the control deviation during On delay Engineering value range and > ER_LM
SP	8.0	REAL	0.0	Setpoint Engineering value range
PV	12.0	REAL	0.0	Controlled variable Engineering value range
SP_DIFF	16.0	REAL	10.0	Setpoint change Engineering value range and > 0.0
TM_DELAY	20.0	TIME	T#60s	On delay of the monitoring signal
TM_RAMP	24.0	TIME	T#60s	Ramp time constant
COM_RST	28.0	BOOL	FALSE	Complete restart
CYCLE	30.0	TIME	T#1s	Sampling time ≥ 1ms

Output parameters ERR_MON (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
QER_LM	34.0	BOOL	FALSE	Limit value of the control deviation triggered
QER_LMTD	34.1	BOOL	FALSE	Limit value of the control deviation during On delay and the ramp triggered
ER	36.0	REAL	0.0	Control deviation

INTEG (S7-300, S7-400)

Description INTEG (S7-300, S7-400)

Process variables are integrated dynamically. Thus, for example, the distance traversed is calculated from the velocity. This instruction can be used to set up a controller with integral component.

Operating principle

The instruction integrates the input variable with respect to time and limits the integral to a definable preassigned high and low limit. Limiting of the output variable is indicated by means of message bits.

The instruction contains the following functions:

Functions	DFOUT_ON	HOLD
Integrate	FALSE	FALSE
Freeze integral component	FALSE	TRUE
Preassign output	TRUE	Any

- **Integrate**
  During integration, the instruction works in accordance with the following transfer function:
  In the Laplace domain:
  OUTV(s) / INV(s) = 1 / (TI*s)
  The time response of the integral component is determined by the integral-action time TI.
  The associated step response is shown in the figure below.

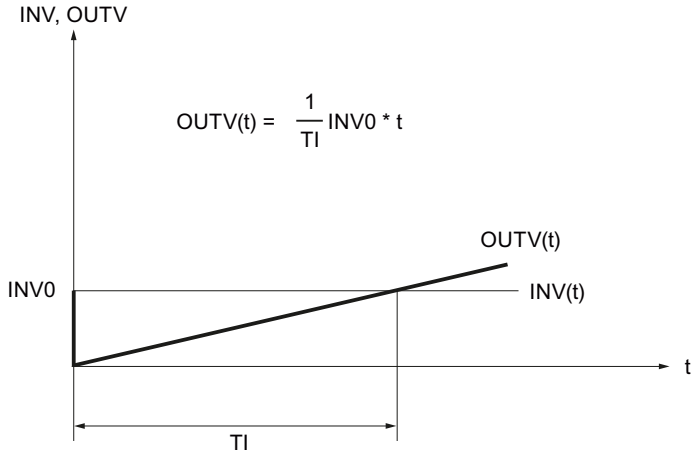

T1:	Integral-action time
INV0:	Input step height
t:	Time
INV:	Input variable
OUTV:	Output variable

The output as well as the memory of the integral component are limited based on the H_LM and L_LM parameters. If the output is outside the range L_LM...H_LM the message bits QH_LM or QL_LM are set to TRUE.

- **Freeze integrator**
  With HOLD = TRUE, the integral component remains at its current output value OUTV. On reset (HOLD = FALSE), integration is continued starting from the current output value OUTV.
- **Preassign output**
  If DFOUT_ON = TRUE is set, DF_OUTV is output at the output. Limit is effective. On reset (DF_OUTV_ON = FALSE), INTEG starts integrating from value DF_OUTV.

The following figure shows an example with DFOUT_ON, HOLD, and limiting

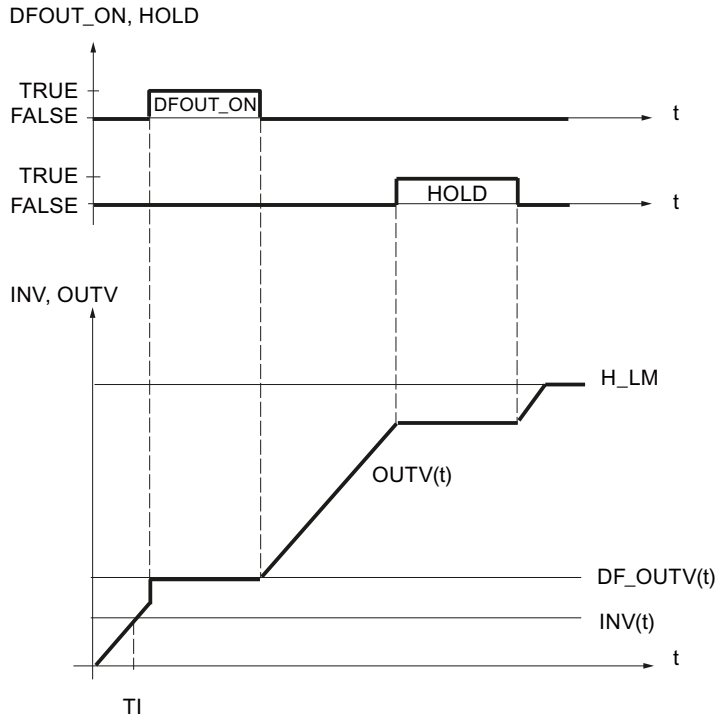

Complete restart

During a complete restart, output OUTV is reset to 0.0. If DFOUT_ON = TRUE is set, DF_OUTV is output. Limiting of the output and its display are also in effect when a complete restart occurs. On transition to normal operation, the instruction continues to integrate from OUTV.

If the integral component is to be started from a specific operating point on a complete restart, the operating point must be entered in input DF_OUTV. When the instruction is called in the complete restart routine, DFOUT_ON= TRUE must be set and reset back to DFOUT_ON= FALSE in the cyclic interrupt level.

Block-internal limits

The integral-action time is subject to a low limit equal to the sampling time.

TI_{intern} = CYCLE for TI < CYCLE

The other values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters INTEG (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
H_LM	4.0	REAL	100.0	High limit Engineering value range H_LM > L_LM
L_LM	8.0	REAL	0.0	Low limit Engineering value range L_HM < H_LM
TI	12.0	TIME	T#25s	Integral-action time TI ≥ CYCLE
DF_OUTV	16.0	REAL	0.0	Default setting of output variable Engineering value range
HOLD	20.0	BOOL	FALSE	Freeze integral component
DFOUT_ON	20.1	BOOL	FALSE	Default setting of output variable On
COM_RST	20.2	BOOL	FALSE	Complete restart
CYCLE	22.0	TIME	T#1s	Sampling time CYCLE ≥ 1ms

Output parameters INTEG (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	26.0	REAL	0.0	Output variable
QH_LM	30.0	BOOL	FALSE	High limit triggered
QL_LM	30.1	BOOL	FALSE	Low limit triggered

LAG1ST (S7-300, S7-400)

Description LAG1ST (S7-300, S7-400)

This instruction can be used as a first-order delay and smoothing element. The time delay is parameterizable.

Operating principle

The instruction smoothes the input variable according to the first-order delay and contains the following functions:

Functions	DFOUT_ON	TRACK
Smoothing	FALSE	FALSE
Tracking	FALSE	TRUE
Preassign output	TRUE	Any

- **Smoothing**
  With smoothing, the instruction operates in the Laplace domain in accordance with the following transfer function:
  OUTV(s) / INV(s) = 1 / (1+TM_LAG*s)
  The time response of the delay element is determined by the time delay TM_LAG. The associated step response is shown in the figure below.

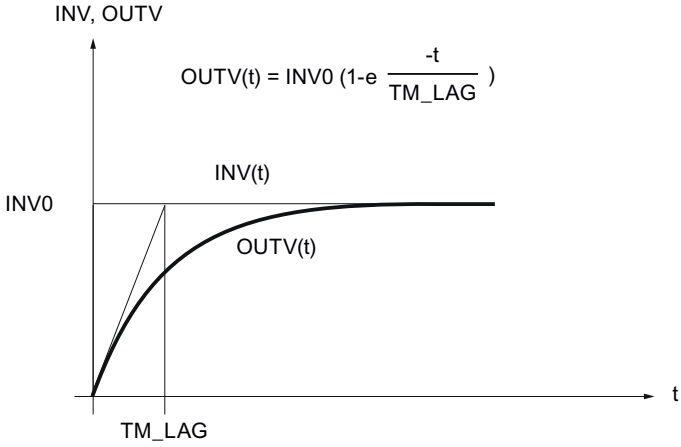

INV	Input variable
OUTV	Output variable
INV0	Input step height
TM_LAG	Time delay

- **Tracking**
  With TRACK = TRUE, input value INV is switched through to output OUTV.
- **Preassign output**
  If DFOUT_ON = TRUE is set, DF_OUTV is output at the output. On reset (DF_OUTV_ON = FALSE), the delay element smoothes starting from value DF_OUTV.
  The following diagram shows the effect of DFOUT_ON and TRACK.

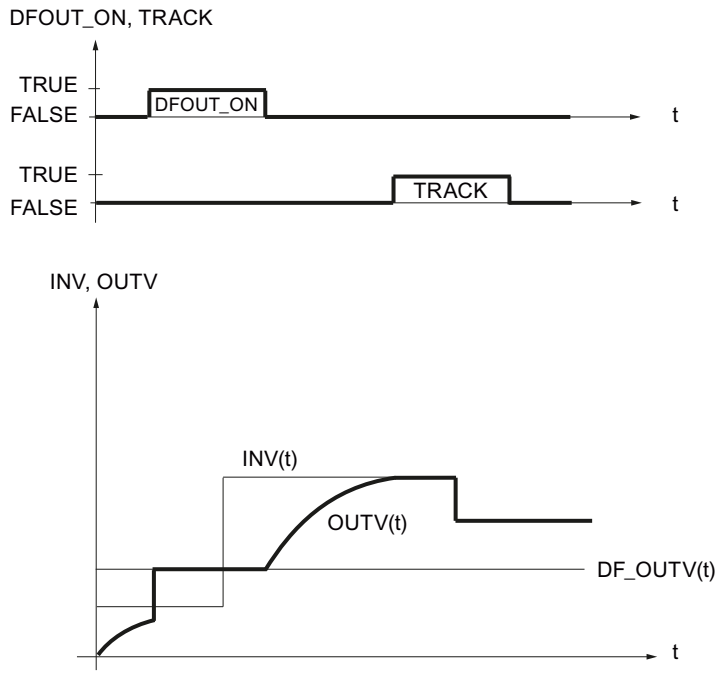

Complete restart

During a complete restart, output OUTV is reset to 0.0. If DFOUT_ON = TRUE is set, DF_OUTV is output. On transition to smoothing, the instruction continues to operate from OUTV.

Block-internal limits

The time delay is subject to a low limit equal to half the sampling time.

TM_LAG_{intern} = CYCLE/**2** for TM_LAG < CYCLE/**2**

The other values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters LAG1ST (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
TM_LAG	4.0	TIME	T#25s	Time delay
DF_OUTV	8.0	REAL	0.0	Default setting of output variable Engineering value range
TRACK	12.0	BOOL	FALSE	Tracking OUTV=INV
DFOUT_ON	12.1	BOOL	FALSE	Default setting of output variable On

Parameter	Offset	Data type	Preassigned value	Description
COM_RST	12.2	BOOL	FALSE	Complete restart
CYCLE	14.0	TIME	T#1s	Sampling time ≥1ms

Output parameters LAG1ST (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	18.0	REAL	0.0	Output variable

LAG2ND (S7-300, S7-400)

Description LAG2ND (S7-300, S7-400)

This instruction is used to simulate controlled system components for feedforward controls and two-circuit control systems.

Operating principle

The instruction implements an oscillating second-order delay element and contains the following functions:

Functions	DFOUT_ON	TRACK
Transferring	FALSE	FALSE
Tracking	FALSE	TRUE
Preassign output	TRUE	Any

- **Transferring**
  The transfer function in the Laplace domain is:
  OUTV(s) / INV(s) = TRANCOEF / (1 + 2*DAM_COEF*TM_CONST*s + TM_CONST^2*s2)
  If DAM_COEF >= 1 (aperiodic case), the transfer element can be represented in a series connection of two first-order delay elements.
  OUTV(s) / INV(s) = TRANCOEF / (1 + T1*s) * 1 / (1 + T2 * s)
  The time constants are converted as follows:

  $$T1 = TM_CONST\ (DAM_COEF + \sqrt{DAM_COEF^2 - 1})$$

  $$T2 = TM_CONST\ (DAM_COEF - \sqrt{DAM_COEF^2 - 1})$$

  The figure below shows the block diagram.

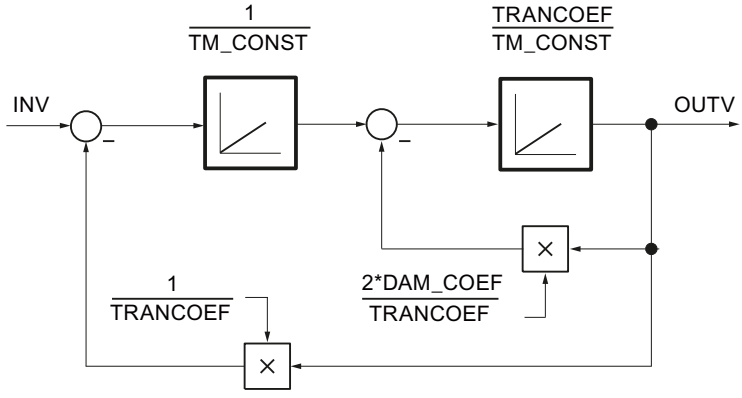

  The following figure shows the step response of the oscillating LAG2ND element.

$$OUTV(t) = TRANCOEF + \frac{TRANCOEF}{\sqrt{1-DAM_COEF^2}} \cdot e^{-\frac{DAM_COEF}{TM_CONST} t} \sin\left(\frac{\sqrt{1-DAM_COEF^2}}{TM_CONST} t - phi\right)$$

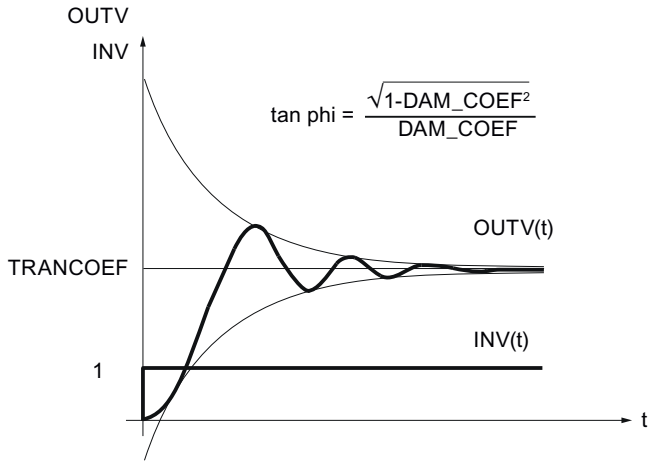

INV	Input variable
OUTV	Output variable
TRANCOEF	Transfer constant
DAM_COEF	Damping constant
phi	Supplement angle

- **Tracking**
  If TRACK =TRUE, OUTV = INV; the internal historical values are set to INV.
- **Preassign output**
  If DFOUT_ON =TRUE, DF_OUTV is output; the internal historical values are set to DF_OUTV. DFOUT_ON has higher priority than TRACK.

Complete restart

During a complete restart, output OUTV is reset to 0.0. If DFOUT_ON = TRUE, DF_OUTV is output.

Block-internal limits

The time constant TM_CONST is subject to a low limit equal to half the sampling time.

$TM_CONST_{intern} = CYCLE/2$ for $TM_CONST < CYCLE/2$

The other values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters LAG2ND (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
TM_CONST	4.0	TIME	T#10s	Time constant
DAM_COEF	8.0	REAL	1.0	Damping constant
TRANCOEF	12.0	REAL	1.0	Transfer constant
DF_OUTV	16.0	REAL	0.0	Default setting of output variable Engineering value range
DFOUT_ON	20.0	BOOL	FALSE	Default setting of output variable On
TRACK	20.1	BOOL	FALSE	Tracking OUTV=INV
COM_RST	20.2	BOOL	FALSE	Complete restart
CYCLE	22.0	TIME	T#1s	Sampling time ≥ 1ms

Output parameters LAG2ND (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	26.0	REAL	0.0	Output variable

LIMALARM (S7-300, S7-400)

Description LIMALARM (S7-300, S7-400)

Illegal or dangerous states can occur in a system if process variables (for example, motor speed, temperature or pressure) exceed or fall below critical values. Such limit violations must be detected and signaled to allow an appropriate reaction.

The instruction checks input variable INV for 4 assignable limits. If one of these limits is reached and exceeded, a limit signal is output. A hysteresis can be set for the off threshold.

Operating principle

The limits are set at inputs H_LM_ALM, H_LM_WRN, L_LM_WRN, and L_LM_ALM. If input variable INV exceeds the limits, the message bits QH_LMALM, QH_LMWRN, QL_LMWRN, and QL_LMALM are set. In order to avoid rapid activating and deactivating of the message bits, the input value must also violate a hysteresis HYS before the outputs are reset.

The figure below shows how the LIMALARM instruction functions:

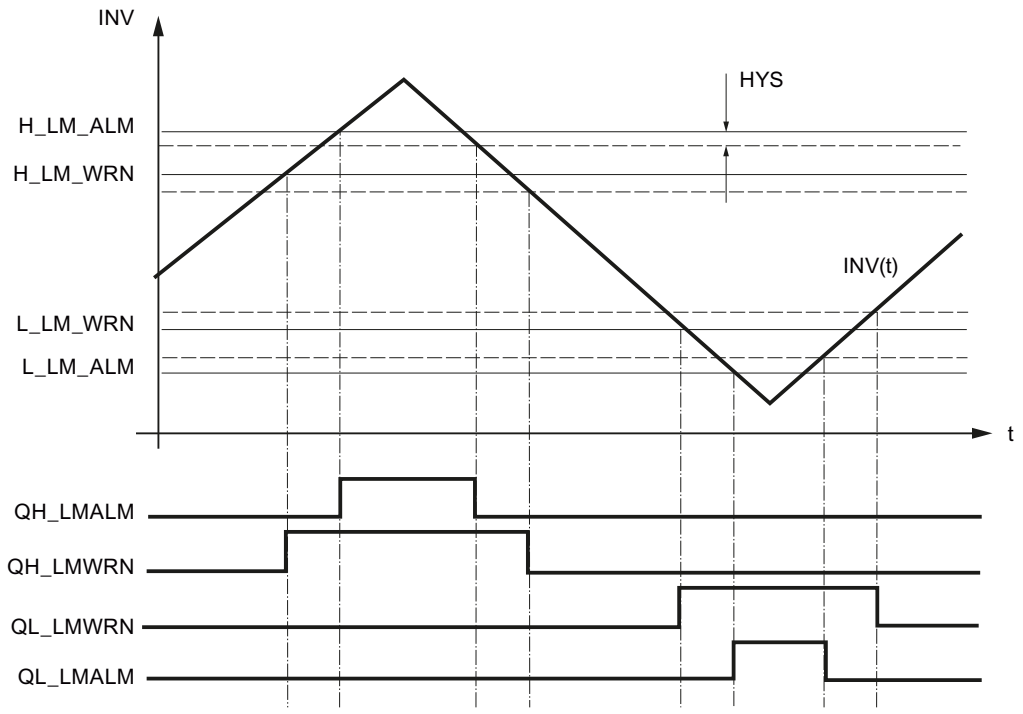

QH_LMALM = TRUE	If	INV rises	and	INV >= H_LM_ALM
	or	INV falls	and	INV >= H_LM_ALM - HYS
QH_LMWRN = TRUE	If	INV rises	and	INV >= H_LM_WRN
	or	INV falls	and	INV >= H_LM_WRN - HYS
QL_LMWRN = TRUE	If	INV falls	and	INV <= L_LM_WRN
	or	INV rises	and	INV <= L_LM_WRN + HYS
QL_LMALM = TRUE	If	INV falls	and	INV <= L_LM_ALM
	or	INV rises	and	INV <= L_LM_ALM + HYS

In order for the instruction to function in a useful way, the following must apply:

L_LM_ALM < L_LM_WRN < H_LM_WRN < H_LM_ALM

Complete restart

During a complete restart, all the signal outputs are set to FALSE.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters LIMALARM (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
H_LM_ALM	0.0	REAL	100.0	High limit alarm Engineering value range > H_LM_WRN
H_LM_WRN	4.0	REAL	90.0	High limit warning Engineering value range > L_LM_WRN
L_LM_WRN	8.0	REAL	10.0	Low limit warning Engineering value range > L_LM_ALM
L_LM_ALM	12.0	REAL	0.0	Low limit alarm Engineering value range < L_LM_WRN
INV	16.0	REAL	0.0	Input variable Engineering value range
HYS	20.0	REAL	1.0	Hysteresis Engineering value range
COM_RST	24.0	BOOL	FALSE	Complete restart

Output parameters LIMALARM (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
QH_LMALM	26.0	BOOL	FALSE	High limit alarm triggered
QH_LMWRN	26.1	BOOL	FALSE	High limit warning triggered
QL_LMWRN	26.2	BOOL	FALSE	Low limit warning triggered
QL_LMALM	26.3	BOOL	FALSE	Low limit alarm triggered

LIMITER (S7-300, S7-400)

Description LIMITER (S7-300, S7-400)

If parameters are defined dynamically (for example calculation of setpoints from process variables), these can assume values that are invalid for the process. The LIMITER instruction can be used to keep parameters within a valid range.

Operating principle

The instruction limits output variable OUTV to a definable high and low limit H_LM and L_LM when input variable INV lies outside these limits. Limiting of OUTV is signaled by outputs QH_LM and QL_LM.

The figure below shows how the LIMITER instruction functions:

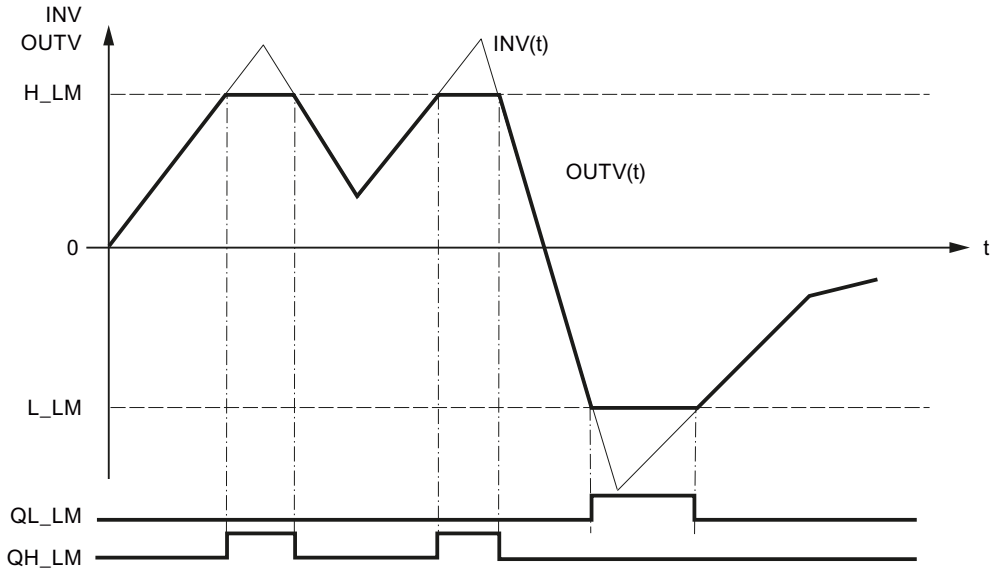

INV	OUTV	QH_LM	QL_LM
≥ H_LM	H_LM,	TRUE	FALSE
≤ L_LM	L_LM,	FALSE	TRUE
≤ L_LM and ≤H_LM	INV	FALSE	FALSE

In order for the instruction to function in a useful way, the following must apply:

L_LM < H_LM

Complete restart

During a complete restart, all signal outputs are set to FALSE; 0.0 is output at output OUTV.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters LIMITER (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
H_LM	4.0	REAL	100.0	High limit Engineering value range > L_LM
L_LM	8.0	REAL	0.0	Low limit Engineering value range < H_LM
COM_RST	12.0	BOOL	FALSE	Complete restart

Output parameters LIMITER (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	14.0	REAL	0.0	Output variable
QH_LM	18.0	BOOL	FALSE	High limit triggered
QL_LM	18.1	BOOL	FALSE	Low limit triggered

LMNGEN_C (S7-300, S7-400)

Description LMNGEN_C (S7-300, S7-400)

This instruction is used to set up a continuous PID controller and determines the manipulated variable or the controller. The manipulated variable can be determined in automatic mode or specified in manual mode. The manual value can be assigned as an absolute value or increased/decreased via a switch. The manipulated variable and the increase in the manipulated variable can be limited to definable values.

Restart

During a complete restart the default value DF_OUTV is switched through to the output LMN irrespective of the default bit DFOUT_ON. Limiting of the output and its display also remains effective during a complete restart. On transition to normal operation, the instruction continues to operate from DF_OUTV.

The output parameters QLMN_HLM and QLMN_LLM are set to FALSE.

Call

While the PID instruction is called in a cyclic interrupt level whose cycle time is adapted to the dominating system time constant, the LMNGEN_C instruction for manual interventions can be called in a faster cyclic interrupt level. Interconnection is carried out with the structured input-output parameters PID_LMNG and LMNG_PID.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Operating principle

The LMNGEN_C instruction is always used in conjunction with the PID instruction.

The following figure shows the interconnection of the continuous PID controller.

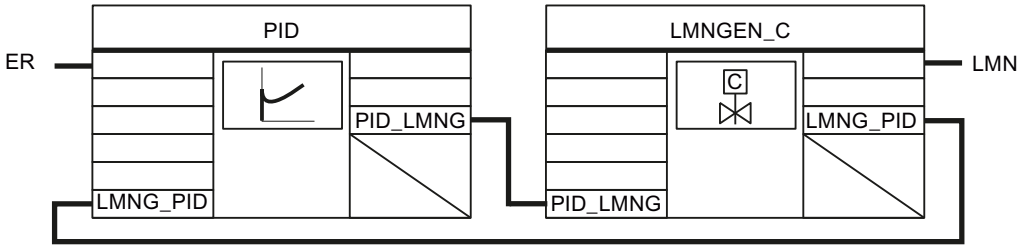

Operating principle LMNGEN_C (S7-300, S7-400)

Limiting the manipulated variable

The manipulated variable is limited in all operating modes. You enter the high limit of the manipulated variable at input LMN_HLM and the low limit at input LMN_LLM. If the calculated manipulated variable exceeds or falls below these values, the high limit or low limit, respectively, is output at LMN. The limiting of the manipulated variable to the high limit is indicated at output QLMN_HLM, and the limiting to the limit limit at output QLMN_LLM.

Manipulated variable ramp

You use the manipulated variable ramp to limit the rate of change of the manipulated variable. You activate the manipulated variable with LMNRC_ON=TRUE. At input parameter LMN_URLM, you specify the maximum rate of change for the rising slope of the manipulated variable, and at input parameter LMN_DRLM the maximum rate of change for the falling slope of the manipulated variable.

Operating modes

The operating modes are activated and deactivated using the DFOUT_ON, MAN_ON, and MANGN_ON parameters.

Operating modes	DFOUT_ON	MAN_ON	MANGN_ON
Automatic mode	FALSE	FALSE	FALSE
Manual mode	FALSE	TRUE	FALSE
Manual value generator	FALSE	TRUE	TRUE
Preassign manipulated variable	TRUE	Not relevant	Not relevant

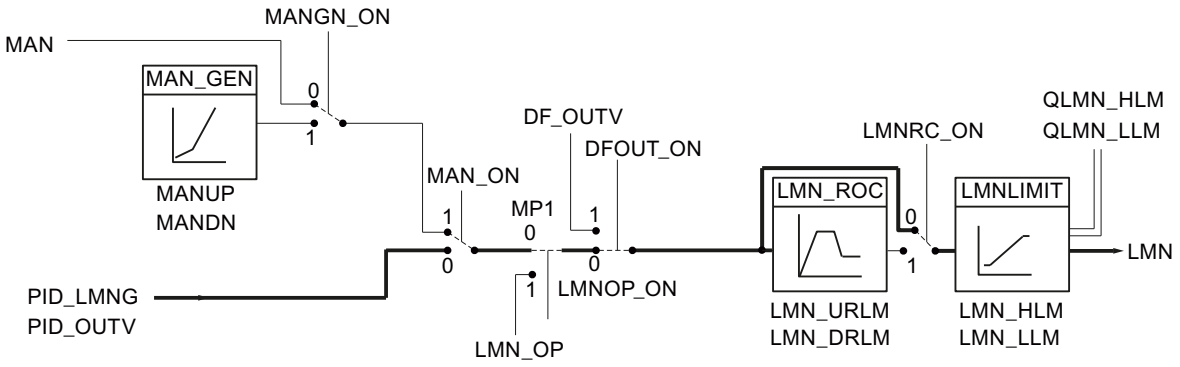

- **Automatic mode**
  The value determined by the PID algorithm is used to determine manipulated variable LMN. The changeover to automatic mode is bumpless if the manipulated variable ramp (LMNRC_ON = TRUE) is activated.
- **Manual mode**
  With MAN_ON = TRUE, you switch to manual mode and thus interrupt the control loop. Manual value MAN is output directly as manipulated variable LMN. MAN must lie within the limits LMN_HLM and LMN_LLM.
- **Manual value generator**
  If MANGN_ON = TRUE as well, the manipulated variable can be increased and decreased using the MANUP and MANDN switches, respectively. The rate of change of the manipulated variable depends on the limits as follows:
  - During the first 3 s after setting of MANUP or MANDN
    dLMN/dt = (LMN_HLM - LMN_LLM) / 100 s
  - Afterwards:
    dLMN/dt = (LMN_HLM - LMN_LLM) / 10 s
- **Preassign manipulated variable**
  When DFOUT_ON = TRUE, the preassigned value DF_OUTV is applied at output LMN. The changeover from or to "Preassign manipulated variable" is only bumpless if the manipulated variable ramp is activated. (LMNRC_ON= TRUE).

Operator control via HMI

You can use static variables to influence the manipulated variable via the HMI (see figure below). The static variables LMN_OP, LMNOP_ON, and MP1 may be operated and monitored via an HMI, for example.

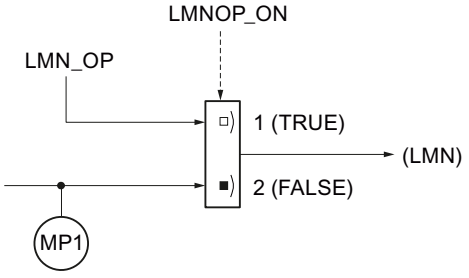

If LMNOP_ON is set to TRUE, LMN_OP is switched through to manipulated variable LMN. MP1 contains the manipulated variable that is being processed by the LMNGEN_C instruction.

If the manipulated variable ramp is activated, bumpless changeover between the values MP1 and LMN_OP is possible. LMN is fed back to the value MP1 in conformance with the specified ramp.

When a static variable is changed, the change takes effect in the process only after the variable has been transferred to the CPU.

Input parameters LMNGEN_C (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
LMN_HLM	0.0	REAL	100.0	High limit of manipulated variable Engineering value range > LMN_LLM
LMN_LLM	4.0	REAL	0.0	Low limit of manipulated variable Engineering value range < LMN_HLM
LMN_URLM	8.0	REAL	10.0	Manipulated variable up rate limit [1/s] > 0.0
LMN_DRLM	12.0	REAL	10.0	Manipulated variable down rate limit [1/s] > 0.0
DF_OUTV	16.0	REAL	0.0	Default setting of output variable Engineering value range
MAN_ON	20.0	BOOL	TRUE	Manual mode ON
MANGN_ON	20.1	BOOL	FALSE	Activate manual value operation
MANUP	20.2	BOOL	FALSE	Manual value up
MANDN	20.3	BOOL	FALSE	Manual value down
LMNRC_ON	20.4	BOOL	FALSE	Manipulated variable rate of change on
DFOUT_ON	20.5	BOOL	FALSE	Default setting of output variable On
COM_RST	20.6	BOOL	FALSE	Complete restart

Parameter	Offset	Data type	Preassigned value	Description
CYCLE	22.0	TIME	T#1s	Sampling time ≥ 1ms
PID_LMNG	26.0	STRUC		PID-LMNGEN interface

Output parameters LMNGEN_C (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
LMN	44.0	REAL	0.0	Manipulated variable
QLMN_HLM	48.0	BOOL	FALSE	High limit of manipulated variable triggered
QLMN_LLM	48.1	BOOL	FALSE	Low limit of manipulated variable triggered
LMNG_PID	50.0	STRUC		PID-LMNGEN interface

In/out parameters LMNGEN_C (S7-300, S7-400)

Table 4-225

Parameter	Offset	Data type	Preassigned value	Description
MAN	68.0	REAL	0.0	Manual value Engineering value range

LMNGEN_C static variables (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
MP1	72.0	REAL	0.0	Manipulated variable processed by LMNGEN_C.
LMM_OP	76.0	REAL	0.0	Operator control via HMI Manual value
LMNOP_ON	80.0	BOOL	FALSE	Operator control via HMI Manual mode ON

LMNGEN_S (S7-300, S7-400)

Description LMNGEN_S (S7-300, S7-400)

This instruction is used to set up a PID step controller for actuators with integral-action response (e.g., motor-driven valves) and determines the manipulated variable of the controller. The manipulated variable can be determined in automatic mode or specified in manual mode. The manual value can be assigned as an absolute value or increased/decreased via a switch. The manipulated variable can be limited to assignable values. The step controller can operate with or without position feedback.

Complete restart

During a complete restart, all the signal outputs are set to 0.

Call

While the PID instruction is called in a cyclic interrupt level whose cycle time is adapted to the dominating system time constant, the LMNGEN_S instruction for manual interventions can be called in a faster cyclic interrupt level. Interconnection is carried out with the structured input-output parameters PID_LMNG and LMNG_PID.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Operating principle

The LMNGEN_S instruction is always used in conjunction with the PID instruction.

The following figure shows the interconnection of a step controller.

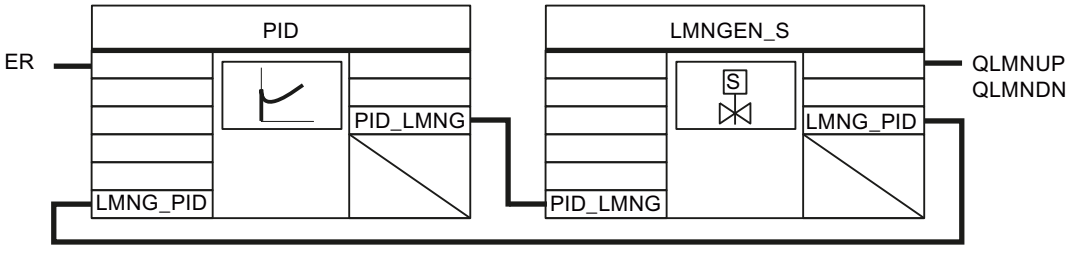

Operating principle LMNGEN_S (S7-300, S7-400)

Limiting the manipulated variable

The manipulated variable is limited in all operating modes. You enter the high limit of the manipulated variable at input LMN_HLM and the low limit at input LMN_LLM. If the calculated manipulated variable exceeds or falls below these values, the high limit or low limit, respectively, is output at LMN. The limiting of the manipulated variable to the high limit is indicated at output QLMN_HLM, and the limiting to the limit limit at output QLMN_LLM.

Limit signals

You interconnect the signals for the upper and lower actuator stop with input parameters LMNR_HS and LMNR_LS. If LMNR_HS= TRUE, the actuator will not open further. If LMNR_LS= TRUE, the actuator will not close further. Output signals QLMNUP and QLMNDN are blocked. The end stops are taken into consideration in all operating modes.

WARNING

If limit signals are not available, the controller cannot recognize a valve stop, and the possibility exists, for example, that the controller will output signals to open the valve even though the valve is at its upper stop. If limit signals are not available, inputs LMNR_HS and LMNR_LS must be assigned the value FALSE.

Step controller with position feedback

If position feedback is available, the instruction uses a three-step element and pulse generator to generate the pulses for controlling the control valve based on the difference between the manipulated variable and the position feedback. You can adapt the response threshold of the three-step element to reduce the switching frequency of the controller.

You activate the position feedback with LMNR_ON = TRUE.

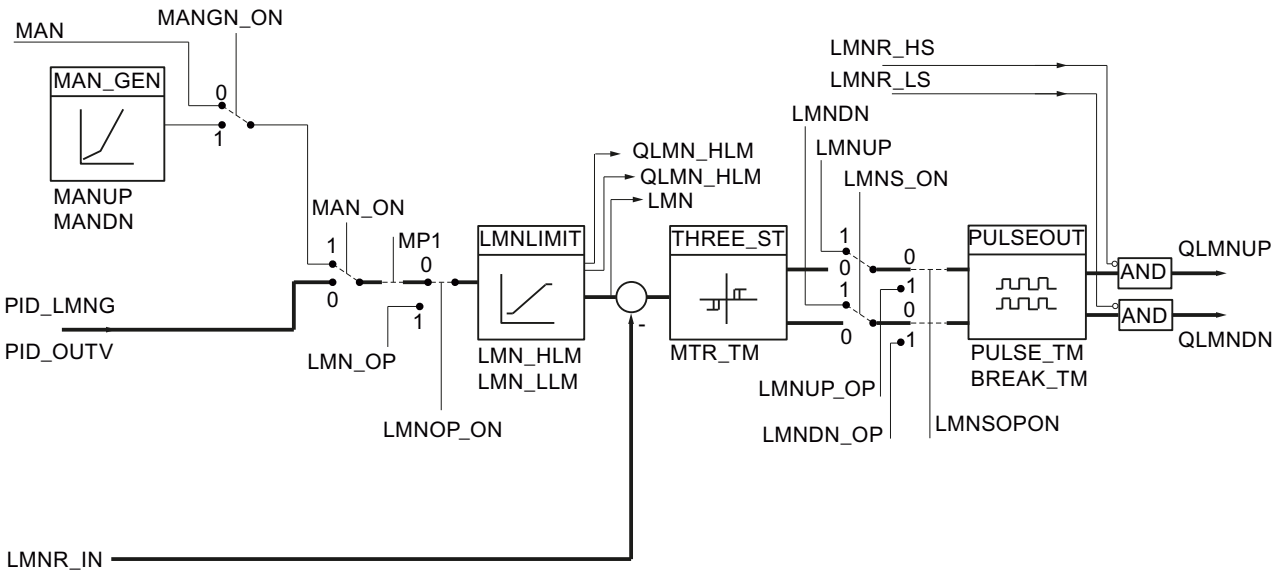

Step controller without position feedback

If position feedback is not available, the instruction adds up the difference between the integral component of the PID algorithm and a simulated position feedback in an integrator and then compares this, as a feedback value, with the remaining proportional-derivative component. The difference is applied in turn to the three-step element and the pulse generator that generates the pulses for the control valve. You can adapt the response threshold of the three-step element to reduce the switching frequency of the controller.

You deactivate the position feedback with LMNR_ON = FALSE.

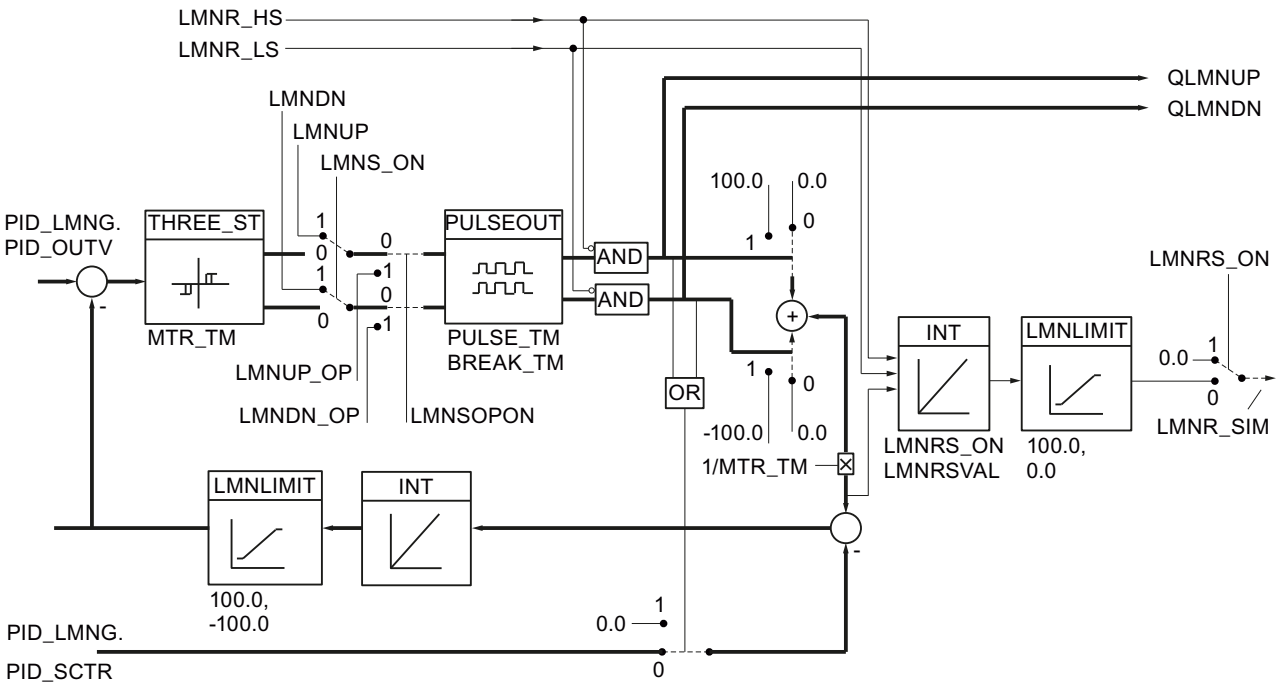

Operating modes

The operating modes are activated and deactivated using the LMN_ON, MAN_ON, and MANGN_ON parameters.

Operating modes	LMN_ON	MAN_ON	MANGN_ON
Automatic mode	FALSE	FALSE	FALSE
Specify manual value	FALSE	TRUE	FALSE
Manual value generator	FALSE	TRUE	TRUE
Manual mode of the manipulated variable output signals	TRUE	Not relevant	Not relevant

The Specify manual value and Manual value generator modes are only possible for step controllers with position feedback. Because information on the valve position is not available in a step controller without position feedback, the actuator cannot be moved to a fixed manual value.

- Specify manual value
  With MAN_ON = TRUE, you switch to manual mode and thus interrupt the conrol loop. Manual value MAN is switched through directly as the manipulated variable. MAN must lie within the limits LMN_HLM and LMN_LLM.
- Manual value generator
  If MANGN_ON = TRUE as well, the manipulated variable can be increased and decreased using the MANUP and MANDN switches, respectively. The rate of change of the manipulated variable depends on the limits as follows:
  - During the first 3 s after setting of MANUP or MANDN
    dLMN/dt = (LMN_HLM - LMN_LLM) / 100 s
  - Afterwards:
    dLMN/dt = (LMN_HLM - LMN_LLM) / 10 s
- Manual mode of the manipulated variable output signals
  If LMNS_ON = TRUE, the binary output signals can be influenced directly. The actuating signal outputs QLMNUP and QLMNDN are set by switches LMNUP and LMNDN. The minimum pulse time PULSE_TM and minimum break time BREAK_TM are also taken into consideration.

- **Automatic operation with position feedback**
  The value PID_LMNG.PID_OUTV sent by the PID algorithm is limited to assignable values LMN_HLM and LMN_LLM. The difference between the manipulated variable and the position feedback is switched to a three-step element with hysteresis. The trip threshold is calculated from the motor actuating time. In order to reduce the switching frequency, the response threshold is adapted. The pulse generator PULSEOUT ensures that the minimum pulse time and minimum break time are observed.
- **Automatic mode without position feedback**
  The difference between the integral component of the PID algorithm and a simulated position feedback is summed in integrator INT. The input of the integrator is the difference of 100.0/0.0/-100.0 (depending on the state of output signals QLMNUP and QLMNDN), referenced to motor actuating time MTR_TM, and the control deviation multiplied times the controller gain, referenced to integral-action time TI of the PID algorithm (PID_LMNG.PID_SCTR). The output of the integrator forms the feedback value that is compared with the remaining proportional-derivative component of the PID algorithm (PID_LMNG.PID_OUTV). The difference is switched to a three-step element with hysteresis. The trip threshold is calculated from the motor actuating time.
  In order to reduce the switching frequency, the response threshold is adapted. The pulse generator PULSEOUT ensures that the minimum pulse time and minimum break time are observed.

Simulation of the position feedback

WARNING

The position feedback is only simulated. It does not have to agree with the real position feedback of the control valve.

If a real position feedback exists, it should also be used.

The simulation of the position feedback is carried out by an integrator with motor actuating time MTR_TM as the integral-action time constant. A rising edge at LMNRS_ON starts the simulation with start value LMNRSVAL. If the value of LMNRS_ON is FALSE, the start value LMNRSVAL is output in parameter LMNR_SIM.

The limit signals LMNR_HS = TRUE and LMNR_LS = TRUE block the integration at high and low limits.

Operator input via HMI

You can use an HMI and static variables to interrupt the manipulated variable branch and specify your own manipulated variables (see figure below).

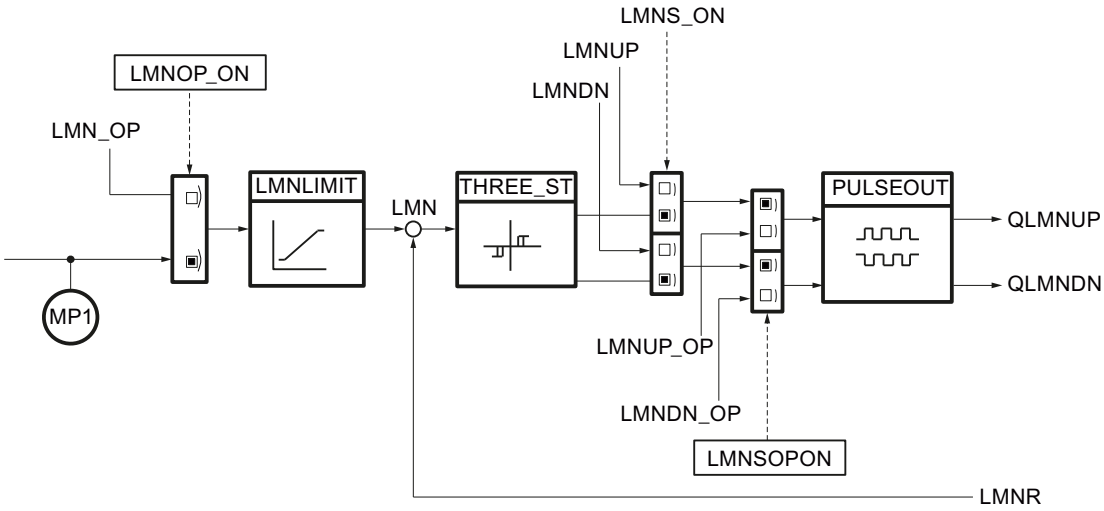

If LMNOP_ON is set to TRUE, LMN_OP is switched through to manipulated variable LMN. MP1 contains the manipulated variable that is being processed by the LMNGEN_S instruction.

If the manipulated variable ramp is activated, bumpless changeover between the values MP1 and LMN_OP is possible. LMN is fed back to the value MP1 in conformance with the specified ramp.

If LMNSOPON is set to TRUE, you can increase the manipulated variable with LMNUP_OP or decrease it with LMNDN_OP. This applies to step controllers with and without position feedback.

The static variables LMNOP_ON, LMN_OP, MP1, LMNSOPON, LMNUP_OP, and LMNDN_OP must not be interconnected and are only manipulated and monitored via an HMI.

When a static variable is changed, the change takes effect in the process only after the variable has been transferred to the CPU.

Input parameters LMNGEN_S (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
LMN_HLM	0.0	REAL	100.0	High limit of manipulated variable Engineering value range > LMN_LLM
LMN_LLM	4.0	REAL	0.0	Low limit of manipulated variable Engineering value range < LMN_HLM
LMNR_IN	8.0	REAL	0.0	Position feedback Engineering value range

Parameter	Offset	Data type	Preassigned value	Description
MTR_TM	12.0	TIME	T#30s	Motor actuating time ≥ CYCLE
PULSE_TM	16.0	TIME	T#3s	Minimum pulse time ≥ CYCLE
BREAK_TM	20.0	TIME	T#3s	Minimum break time ≥ CYCLE
MAN_ON	24.0	BOOL	TRUE	Manual mode ON
MANGN_ON	24.1	BOOL	FALSE	Activate manual value operation
MANUP	24.2	BOOL	FALSE	Manual value up
MANDN	24.3	BOOL	FALSE	Manual value down
LMNR_ON	24.4	BOOL	FALSE	Position feedback on
LMNR_HS	24.5	BOOL	FALSE	High limit signal of position feedback
LMNR_LS	24.6	BOOL	FALSE	Low limit signal of position feedback
LMNS_ON	24.7	BOOL	FALSE	Activate manual mode for manipulated variable signals
LMNUP	25.0	BOOL	FALSE	Manipulated variable signal UP
LMNDN	25.1	BOOL	FALSE	Manipulated variable signal DOWN
COM_RST	25.2	BOOL	FALSE	Complete restart
CYCLE	26.0	TIME	T#1s	Sampling time ≥ 1ms
PID_LMNG	30.0	STRUC		PID-LMNGEN interface

Output parameters LMNGEN_S (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
LMN	48.0	REAL	0.0	Manipulated variable
QLMNUP	52.0	BOOL	FALSE	Manipulated value signal up
QLMNDN	52.1	BOOL	FALSE	Manipulated variable signal DOWN
QLMN_HLM	52.2	BOOL	FALSE	High limit of manipulated variable triggered
QLMN_LLM	52.3	BOOL	FALSE	Low limit of manipulated variable triggered
LMNG_PID	54.0	STRUC		PID-LMNGEN interface

In/out parameters LMNGEN_S (S7-300, S7-400)

Table 4-226

Parameter	Offset	Data type	Preassigned value	Description
MAN	72.0	REAL	0.0	Manual value Engineering value range

LMNGEN_S static variables (S7-300, S7-400)

You must not modify variables that are not listed. These are used for internal purposes only.

Parameter	Offset	Data type	Preassigned value	Description
LMNSOPON	76.0	BOOL	FALSE	Operator control via HMI Manual mode of the manipulated variable output signals ON
LMNUP_OP	76.1	BOOL	FALSE	Operator control via HMI Manipulated variable UP
LMNDN_OP	76.2	BOOL	FALSE	Operator control via HMI Manipulated variable DOWN
LMNOP_ON	76.3	BOOL	FALSE	Operator control via HMI Manual mode ON
LMNRS_ON	76.4	BOOL	FALSE	Simulated position feedback ON
MP1	78.0	REAL	0.0	Operator control via HMI Manipulated variable processed by LMNGEN_C.
LMM_OP	82.0	REAL	0.0	Operator control via HMI Manual value
LMNRSVAL	86.0	REAL	0.0	Initial value for simulate position feedback
LMNR_SIM	9.0	REAL	0.0	Simulated position feedback

NONLIN (S7-300, S7-400)

Description NONLIN (S7-300, S7-400)

The NONLIN instruction can be used to adapt an input value, such as a measured value of a thermocouple, as a function of a definable characteristic (polyline).

Operating principle

NONLIN interpolates the corresponding output variable OUTV for the current input variable INV using time slices on a polyline. The time slices of the polyline are stored in a global data block.

Global data block DB_NONLIN (Page 5761) is not included in the library. You must create this global data block yourself.

Note

The instruction does not check whether a global DB with the number DB_NBR actually exists or whether the DB_NBR. NBR_PTS parameter (number of time slices) is suited to the length of the data block. If the parameter assignment is incorrect, the CPU changes to the STOP operating mode with the message "Internal system error".

NONLIN has the following functions:

Functions	DFOUT_ON	TRACK	OUTV
Interpolation	FALSE	FALSE	OUTV = f(INV)
Preassign output	TRUE	Any	DF_OUTV
Tracking	FALSE	TRUE	OUTV = INV

- Interpolation
  The following figure shows how output variable OUTV is determined from input variable INV.
  OUTV = f(INV)

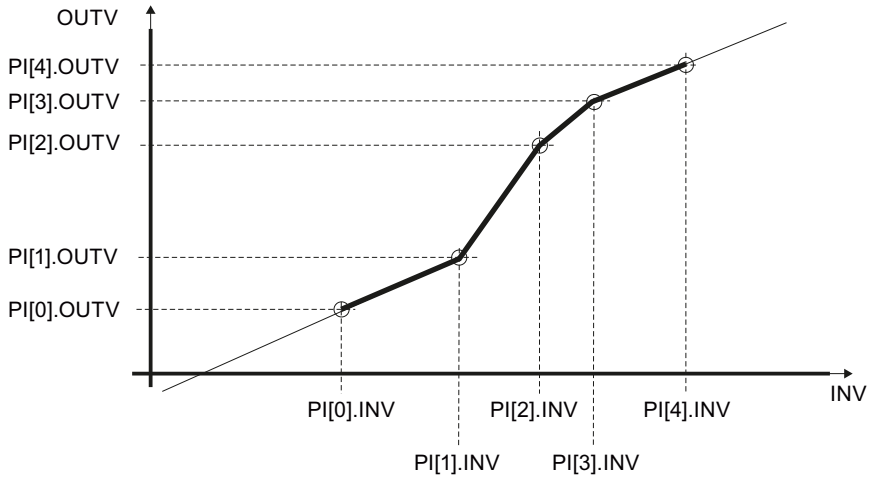

  If the input value lies below time slice 0, extrapolation is carried out with the rise of the time slice pair 0/1. It it lies above the last time slice, extrapolation is carried out with the rise of the last time slice pair. In order for the instruction to produce a useful result, the values of the time slices must exhibit a monotonic rise in ascending order.

- Preassign output
  If DFOUT_ON = TRUE is set, DF_OUTV is output at the output; the change of OUTV is a step change. On changeover to DFOUT_ON = FALSE the change of OUTV is also a step change.

- Tracking
  When TRACK=TRUE, the input value is output directly (OUTV=INV). The change is a step change, as in the case of DFOUT_ON. TRACK has lower priority than DFOUT_ON.

Restart

On a complete restart, OUTV = 0.0 is output. If DFOUT_ON=TRUE, DF_OUTV is output.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters NONLIN (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
DF_OUTV	4.0	REAL	0.0	Default setting of output variable Engineering value range
DB_NBR	8.0	BLOCK_DB	DB 1	Data block number
DFOUT_ON	10.0	BOOL	FALSE	Default setting of output variable On
TRACK	10.1	BOOL	FALSE	Tracking OUTV=INV
COM_RST	10.2	BOOL	FALSE	Complete restart

Output parameters NONLIN (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	12.0	REAL	0.0	Output variable

Global data block DB_NONLIN (S7-300, S7-400)

The time slices of the polyline for the instruction NONLIN are stored in a global data block. This global data block is not included in the library. You must create the global data block yourself according to the following schema and adapt it to your application.

Parameter	Data type	Default	Description
DB_NBR.NBR_PTS	INT	4	Highest time slice number Range of values 1 - 255
DB_NBR.PI[0].OUTV	REAL	0.0	Output variable [0] Starting point 0 Engineering value range
DB_NBR.PI[0].INV	REAL	0.0	Input variable [0] Starting point 0 Engineering value range
DB_NBR.PI[1].OUTV	REAL	0.0	Output variable [1] Time slice 1 Engineering value range
DB_NBR.PI[1].INV	REAL	0.0	Input variable [1] Time slice 1 Engineering value range
DB_NBR.PI[2].OUTV	REAL	0.0	Output variable [2] Time slice 2 Engineering value range
DB_NBR.PI[2].INV	REAL	0.0	Input variable [2] Time slice 2 Engineering value range
DB_NBR.PI[3].OUTV	REAL	0.0	Output variable [3] Time slice 3 Engineering value range

Parameter	Data type	Default	Description
DB_NBR.PI[3].INV	REAL	0.0	Input variable [3] Time slice 3 Engineering value range
DB_NBR.PI[4].OUTV	REAL	0.0	Output variable [4] Time slice 4 Engineering value range
DB_NBR.PI[4].INV	REAL	0.0	Input variable [4] Time slice 4 Engineering value range

Alternatively, you can generate the global data block using an external source file. Proceed as follows:

1. Copy the text below to the clipboard.
2. Open an external text editor.
3. Paste the copied text from the clipboard to the text editor.
4. Save the file with the file name extension "DB":
5. Open the "External sources" folder in the project tree of the TIA Portal.
6. Double-click the command "Add new external file".
   The "Open" dialog is displayed.
7. Navigate to the external source file that you have created and select it.
8. Confirm your selection with "Open".
9. Select the external source file.
10. Select the command "Generate blocks from source" in the shortcut menu.
11. A security prompt informs you that any existing blocks will be overwritten.
12. Click "OK" in response to the security prompt.
13. Adapt the created data block to your application.

```
DATA_BLOCK "DB_NONLIN"
TITLE = data block nonlinear static function
{ S7_Optimized_Access := 'FALSE' }
NAME : DB_NONLI
    STRUCT
        NBR_PTS : Int;                  // number of points (excluding starting point)
        PI : Array[0..4] of Struct
            OUTV : Real;                // output variable
            INV : Real;                 // input variable
        END_STRUCT;
    END_STRUCT;


    BEGIN
        NBR_PTS := 4;
        PI[0].OUTV := 1.0;
```

```
    PI[0].INV := 2.0;
    PI[1].OUTV := 2.0;
    PI[1].INV := 3.0;
    PI[2].OUTV := 4.0;
    PI[2].INV := 4.0;
    PI[3].OUTV := 5.0;
    PI[3].INV := 5.0;
    PI[4].OUTV := 6.0;
    PI[4].INV := 7.0;
END_DATA_BLOCK
```

NORM (S7-300, S7-400)

Description NORM (S7-300, S7-400)

In the case of process values the value supplied by the encoder often lies in a range that is unfavorable for the user (for example 0 to 10 V correspond to 0 to 1200 ºC or 0 to 10 V correspond to 0 to 3000 rpm). By scaling the setpoint or the process value, both process variables lie in the same range of values.

Operating principle

The instruction scales the input variable into an output variable with a different range of values. Input variable INV is transformed to output variable OUTV based on the scaling line. The scaling line is defined by the two value pairs IN_LVAL, OUT_LVAL and IN_HVAL, OUT_HVAL.

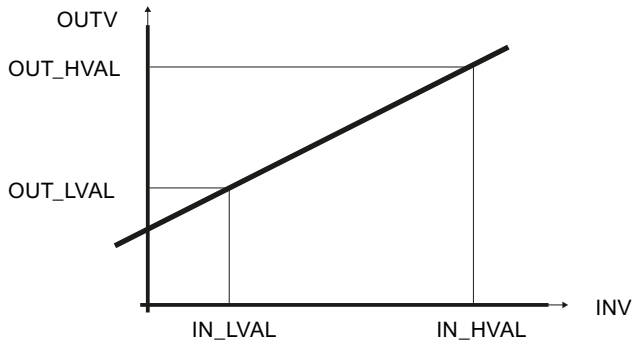

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters NORM (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
IN_HVAL	4.0	REAL	100.0	Physical input value below Engineering value range > IN_LVAL
OUT_HVAL	8.0	REAL	100.0	Physical output value below Engineering value range > OUT_LVAL
IN_LVAL	12.0	REAL	0.0	Physical input value above Engineering value range < IN_HVAL
OUT_LVAL	16.0	REAL	0.0	Physical output value above Engineering value range < OUT_HVAL

Output parameters NORM (S7-300, S7-400)

Parameter	Offset	Data type	Description
OUTV	20.0	REAL	Output variable

OVERRIDE (S7-300, S7-400)

Description OVERRIDE (S7-300, S7-400)

This instruction is required to implement an override control.

Operating principle

Two PID controllers are connected to one actuator. For this purpose, two PID instructions are interconnected by means of instruction OVERRIDE with instruction LMNGEN_C or LMNGEN_S. If you use LMNGEN_S, position feedback must be activated (LMNR_ON=TRUE).

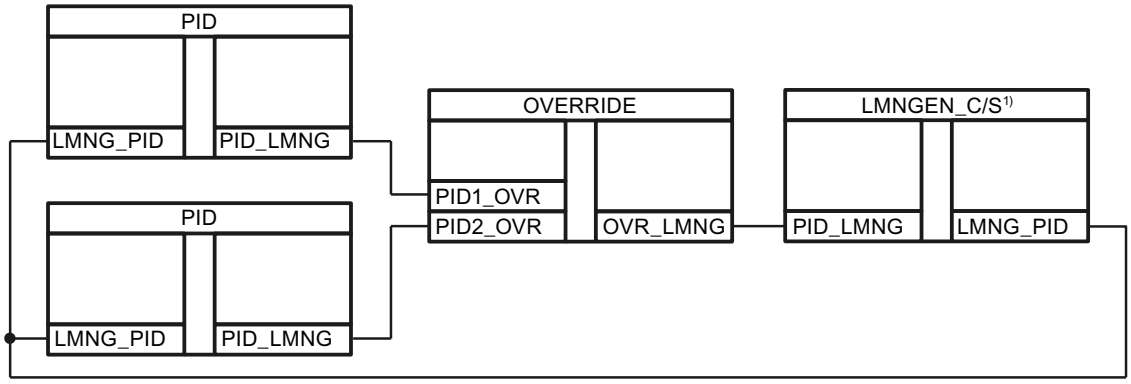

The instruction compares the values PID1_OVR and PID2_OVR and connects either the larger or the smaller value.

The instruction transfers the value in PID1_OVR or in PID2_OVR to LMN_GEN_C or LMN_GEN_S.

If the PID1_ON and PID2_ON switches are unequal, the one value is always transferred. If the PID1_ON and PID2_ON switches are equal, the larger (OVR_MODE=FALSE) or smaller (OVR_MODE=TRUE) of the two values is transferred.

PID1_ON	PID2_ON	OVR_MOD	Function
TRUE	FALSE	Not relevant	PID1_OVR is transferred.
FALSE	TRUE		PID2_OVR is transferred.
FALSE	FALSE	FALSE	The larger value of PID1_OVR and PID2_OVR is transferred.
TRUE	TRUE	FALSE	
FALSE	FALSE	TRUE	The smaller value of PID1_OVR and PID2_OVR is transferred.
TRUE	TRUE	TRUE	

If PID1_OVR is transferred, QPID1=TRUE; if PID2_OVR is transferred, QPID2=TRUE.

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters OVERRIDE (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
PID1_ON	0.0	BOOL	FALSE	PID controller 1 on
PID2_ON	0.1	BOOL	FALSE	PID controller 2 on
OVR_MODE	0.2	BOOL	FALSE	Override mode FALSE=maximum, TRUE=minimum
PID1_OVR	2.0	STRUC		PID-LMNGEN interface
PID2_OVR	20.0	STRUC		PID-LMNGEN interface

Output parameters OVERRIDE (S7-300, S7-400)

Parameter	Offset	Data type	Description
QPID1	38.0	BOOL	PID controller 1 active
QPID2	38.1	BOOL	PID controller 2 active
OVR_LMNG	40.0	STRUC	PID-LMNGEN interface

PARA_CTL (S7-300, S7-400)

Description PARA_CTL (S7-300, S7-400)

The instruction is used in controller structures with parameter changeovers if optimum controller parameters are required for different operating ranges.

Operating principle

Several controller parameter sets (GAIN, TI, TD, and TM_LAG) can be transferred to a PID controller.

The PSET1_ON...PSET4_ON switches can be used to switch through one of the 4 parameter sets to outputs GAIN, TI, TD, and TM_LAG.

If...	Then...
PSET1_ON = TRUE	GAIN = PARASET1.GAIN
	TI = PARASET1.TR
	TD = PARASET1.TD
	TM_LAG = PARASET1.TM_LAG
PSET2_ON = TRUE	GAIN = PARASET2.GAIN
	TI = PARASET2.TR
	TD = PARASET2.TD
	TM_LAG = PARASET2.TM_LAG

PSET3_ON = TRUE	GAIN = PARASET3.GAIN
	TI = PARASET3.TR
	TD = PARASET3.TD
	TM_LAG = PARASET3.TM_LAG
PSET4_ON = TRUE	GAIN = PARASET4.GAIN
	TI = PARASET4.TR
	TD = PARASET4.TD
	TM_LAG = PARASET4.TM_LAG

If 2 or more switches are set to TRUE, the switch with the lowest number always has the highest priority. If no switch is set to TRUE, the first parameter set is transferred.

In order to achieve a bumpless changeover between the parameter sets, a rate of change limiter (ROC_LIM) must be connected between the PID and PARA_CTL instructions for the GAIN parameter.

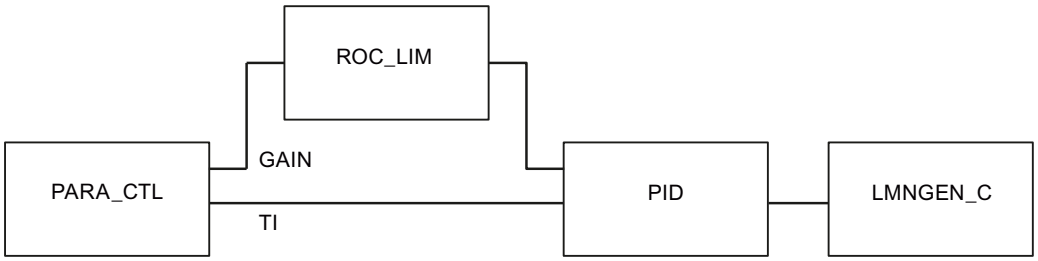

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters PARA_CTL (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
PSET1_ON	0.0	BOOL	FALSE	Transfer Parameter set number 1
PSET2_ON	0.1	BOOL	FALSE	Transfer Parameter set number 2
PSET3_ON	0.2	BOOL	FALSE	Transfer Parameter set number 3
PSET4_ON	0.3	BOOL	FALSE	Transfer Parameter set number 4
PARASET1.GAIN	2.0	REAL		Proportional gain 1
PARASET1.TI	6.0	TIME	5 s	Integral-action time 1
PARASET1.TD	10.0	TIME	10 s	Derivative-action time 1
PARASET1.TM_LAG	14.0	TIME	2 s	Time delay 1

Parameter	Offset	Data type	Preassigned value	Description
PARASET2.GAIN	18.0	REAL		Proportional gain 2
PARASET2.TI	22.0	TIME	5 s	Integral-action time 2
PARASET2.TD	26.0	TIME	10 s	Derivative-action time 2
PARASET2.TM_LAG	30.0	TIME	2 s	Time delay 2
PARASET3.GAIN	34.0	REAL		Proportional gain 3
PARASET3.TI	38.0	TIME	5 s	Integral-action time 3
PARASET3.TD	42.0	TIME	10 s	Derivative-action time 3
PARASET3.TM_LAG	46.0	TIME	2 s	Time delay 3
PARASET4.GAIN	500	REAL		Proportional gain 4
PARASET4.TI	54.0	TIME	5 s	Integral-action time 4
PARASET4.TD	58.0	TIME	10 s	Derivative-action time 4
PARASET4.TM_LAG	62.0	TIME	2 s	Time delay 4

Output parameters PARA_CTL (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
GAIN	66.0	REAL	1.0	Proportional gain
TI	70.0	TIME	10s	Integral action time
TD	74.0	TIME	5s	Derivative-action time
TM_LAG	78.0	TIME	2s	Time delay

PID (S7-300, S7-400)

Description PID (S7-300, S7-400)

This instruction contains the PID algorithm for setting up the following controller types:

Continuous PID controller:	PID + LMNGEN_C
PID step controller for proportional actuators:	PID + LMNGEN_C + PULSEGEN
PID step controller for integrating actuators:	PID + LMNGEN_S

The instruction implements the PID algorithm. It is realized as a pure parallel structure and acts solely as a position algorithm. You can activate and deactivate the proportional, integral and derivative components individually. This allows P, PI, PD, and PID controllers to be configured.

The calculation of the proportional and derivative components can be laid into the feedback. Laying the proportional and derivative components into the feedback loop makes the response to setpoint changes "smooth" while retaining the same speed in settling disturbance variables. It is usually possible to dispense with the common application of a setpoint integrator in order to avoid setpoint steps.

While the PID instruction is placed in a cyclic interrupt level whose cycle time is adapted to the dominating system time constant, the instructions that run the actuators (LMNGEN_C or LMNGEN_S) can be placed in a faster cyclic interrupt level.

Complete restart

During a complete restart zero is pre-assigned to all the output and in/out parameters.

Block-internal limits

The integral-action time is subject to a limit equal to half the sampling time.

The derivative-action time is subject to a low limit equal to the sampling time.

The time lag is subject to a low limit equal to half the sampling time.

TI_{intern} = CYCLE/2	for	TI < CYCLE/2
TD_{intern} = CYCLE	for	TD < CYCLE
TM_LAG_{intern} = CYCLE/2	for	TM_LAG < CYCLE/2

The other values of the input parameters are not limited in the instruction. There is no parameter check.

Operating principle PID (S7-300, S7-400)

Controller structure

The following controller structures are possible with switches P_SEL, I_SEL and D_SEL:

Controller structure	P_SEL	I_SEL	D_SEL
P control	TRUE	TRUE bzw. FALSE	FALSE
PI control	TRUE	TRUE	FALSE
PD control	TRUE	FALSE	TRUE
PID control	TRUE	TRUE	TRUE

Block diagram of the PID algorithm

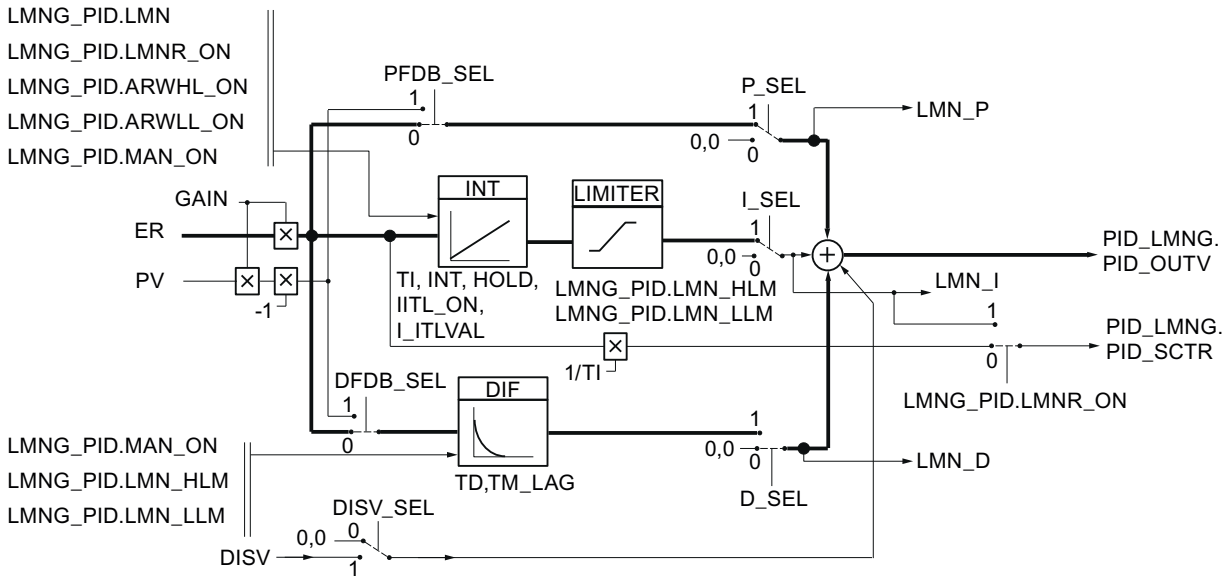

Proportional component

The proportional action can optionally be activated or deactivated using the P_SEL switch. The proportional action is the product of the control deviation ER and the proportional gain GAIN. It can be laid into the feedback path using the PFDB_SEL switch. If the proportional action has been laid into the feedback path, the proportional action is the product of the the process value PV and the proportional gain.

Integral action

The integral action can optionally be activated or deactivated with the I_SEL switch. The time response is determined by the integral action time TI. When deactivated, the integral action and the internal memory of the integral action are set to zero. The integral action can be held using INT_HOLD. You can assign the integral action the value at input I_ITLVAL with the I_ITL_ON switch. In the case manipulated variable limiting the Integral action time remains at the old value (anti reset wind-up).

The following applies to a step controller without position feedback: LMN_I = 0.0

Derivative component

The derivative action can be activated or deactivated with the D_SEL switch. It can be laid into the feedback path by using the DFDB_SEL switch. The input variable of the derivative action is now the controlled variable PV. The time response is determined by the derivative action time TD.

Oscillation in manipulated variable signals may occur, particularly with rapid processes and an activated derivative action. In this case, the control response can be improved by using the first-order delay integrated in the derivative action. Usually a small TM_LAG time delay is sufficient in order to achieve the desired success.

Feedforward control

A disturbance variable DISV can be fed forward additively to the manipulated variable. It can optionally be activated or deactivated by using the DISV_SEL switch.

Input parameters PID (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
ER	0.0	REAL	0.0	Control deviation Engineering value range
PV	4.0	REAL	0.0	Controlled variable (proportional or derivative component in the feedback) Engineering value range
GAIN	8.0	REAL	1.0	Proportional gain
TI	12.0	TIME	T#20s	Integral action time
I_ITLVAL	16.0	REAL	0.0	Initialization value for the integral component Engineering value range
TD	20.0	TIME	T#10s	Derivative-action time
TM_LAG	24.0	TIME	T#2s	Time delay of the derivative component
DISV	28.0	REAL	0.0	Disturbance variable Engineering value range
P_SEL	32.0	BOOL	TRUE	Activate proportional component
PFDB_SEL	32.1	BOOL	FALSE	Switch proportional component into feedback
DFDB_SEL	32.2	BOOL	FALSE	Switch derivative component into feedback
I_SEL	32.3	BOOL	TRUE	Activate integral component
INT_HPOS	32.4	BOOL	FALSE	Hold integral component in positive direction
INT_HNEG	32.5	BOOL	FALSE	Hold integral component in negative direction
I_ITL_ON	32.6	BOOL	FALSE	Set integral component
D_SEL	32.7	BOOL	FALSE	Activate derivative component
DISV_SEL	33.0	BOOL	TRUE	Apply disturbance variable
SMOO_CHG	33.1	BOOL	TRUE	Smooth changeover from manual mode to automatic mode
COM_RST	33.2	BOOL	FALSE	Complete restart
CYCLE	34.0	TIME	T#1s	Sampling time ≥ 1ms
LMNG_PID	38.0	STRUC		PID-LMNGEN interface

Output parameters PID (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
LMN_P	56.0	REAL	0.0	Proportional component
LMN_I	60.0	REAL	0.0	Integral component
LMN_D	64.0	REAL	0.0	Derivative component
PID_LMNG	68.0	STRUC		PID-LMNGEN interface

PULSEGEN_M (S7-300, S7-400)

Description PULSEGEN_M (S7-300, S7-400)

The instruction PULSEGEN_M is used to set up a PID controller with pulse output for proportional actuators. You can use the PULSEGEN_M instruction to configure two- or three-step PID controllers with pulse width modulation. The instruction is usually used in conjunction with the continuous controller PID + LMNGEN_C.

The diagram below shows how the instructions are connected.

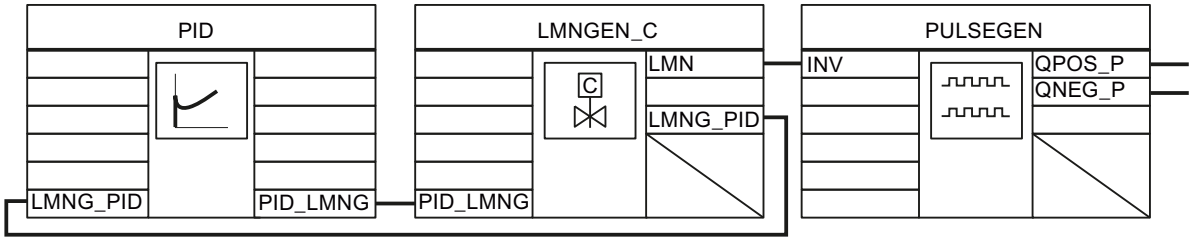

Restart

During a restart, all the signal outputs are set to 0.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Operating principle PULSEGEN_M (S7-300, S7-400)

Pulse width modulation

PULSEGEN_M transforms the input variable INV (= LMN of the PID controller) by modulating the pulse width into a pulse train with a constant period PER_TM. The period corresponds to the cycle time with which the input variable is updated. The duration of a pulse per period is proportional to the input variable. The cycle assigned via PER_TM is not identical to the processing cycle of the PULSEGEN_M instruction. Rather, a PER_TM cycle is made up of several processing cycles of the PULSEGEN_M instruction, whereby the number of PULSEGEN_M calls per PER_TM cycle measures the accuracy of the pulse width.

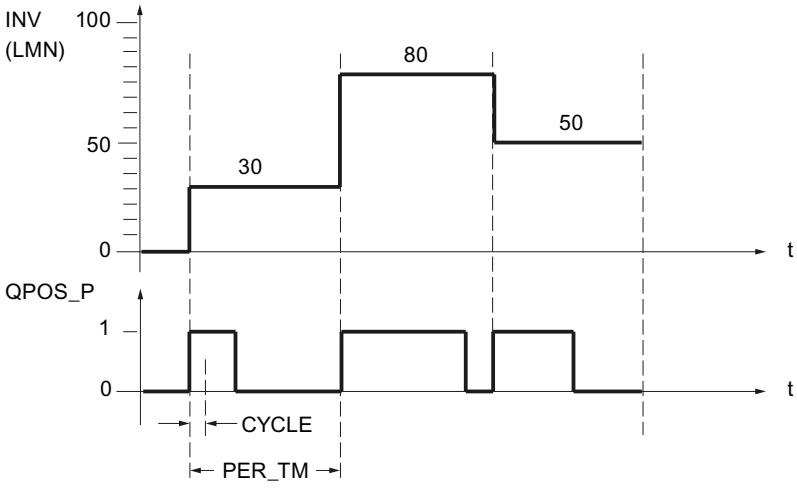

An input variable of 30% and 10 PULSEGEN calls per PER_TM mean the following:

- "One" at the QPOS_P output for the first three calls of PULSEGEN_M (30% of 10 calls)
- "Zero" at the QPOS_P output for seven further calls of PULSEGEN_M (70% of 10 calls)

Minimum pulse time or minimum break time

A correctly configured minimum pulse time or minimum break time P_B_TM can prevent short on/off times, which reduce the working life of actuators. Small absolute values at the input variable LMN that would otherwise generate a pulse duration shorter than P_B_TM are suppressed. Large input values that would generate a pulse duration longer than PER_TM - P_B_TM are set to 100 % or -100 %.

The following formula applies: P_B_TM ≤ 0.1 × PER_TM

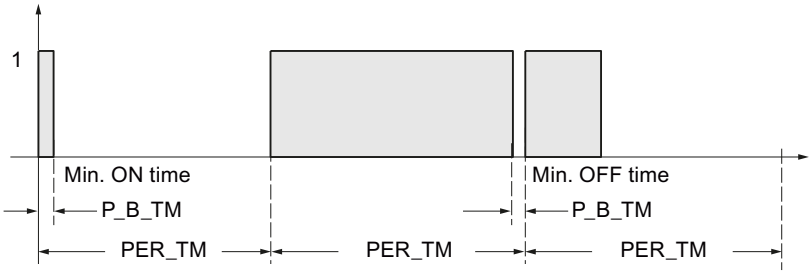

Duration of positive or negative pulses is calculated from the input variable (in %) multiplied by the period duration:

Pulse duration = INV / 100 * PER_TM

Accuracy of pulse width modulation

While the continual controller PID+LMGEN_C is placed in a slow cyclic interrupt level whose cycle time is adapted to the dominating system time constant, the PULSEGEN_M block must be placed in a faster cyclic interrupt level.

With a "Sampling ratio" of 1:10 (controller calls to PULSEGEN_M calls), the accuracy of the manipulated variable is restricted to 10%, in other words, set input values INV can only be simulated by a pulse duration at the QPOS_P output with an accuracy of 10%.

The accuracy is increased as the number of PULSEGEN_M calls per controller call is increased.

The faster the cyclic interrupt level is, the greater the precision with which the manipulated variable can be output. At a cyclic interrupt level that is 100 times faster, a resolution of 1% of the manipulated variable range is achieved.

Automatic synchronization

It is possible to synchronize the pulse output with the controller instruction automatically. This ensures that a modified value of the manipulated variable LMN(t) is output as fast as possible as a proportionally modified pulse duration of the binary signal.

The pulse generator evaluates the input variable INV within the interval of the period PER_TM in each case. Since, however, INV is calculated in a slower cyclic interrupt class, the pulse generator should start the conversion of the discrete value into a pulse signal as soon as possible after the updating of INV. To allow this, the instruction can synchronize the start of the period using the following procedure. If INV changes and if the block call is not in the first or last two call cycles of a period, synchronization is performed. The pulse duration is recalculated and, in the next cycle, is output with a new period.

The period PER_TM must correspond to the sampling time CYCLE of the controller.

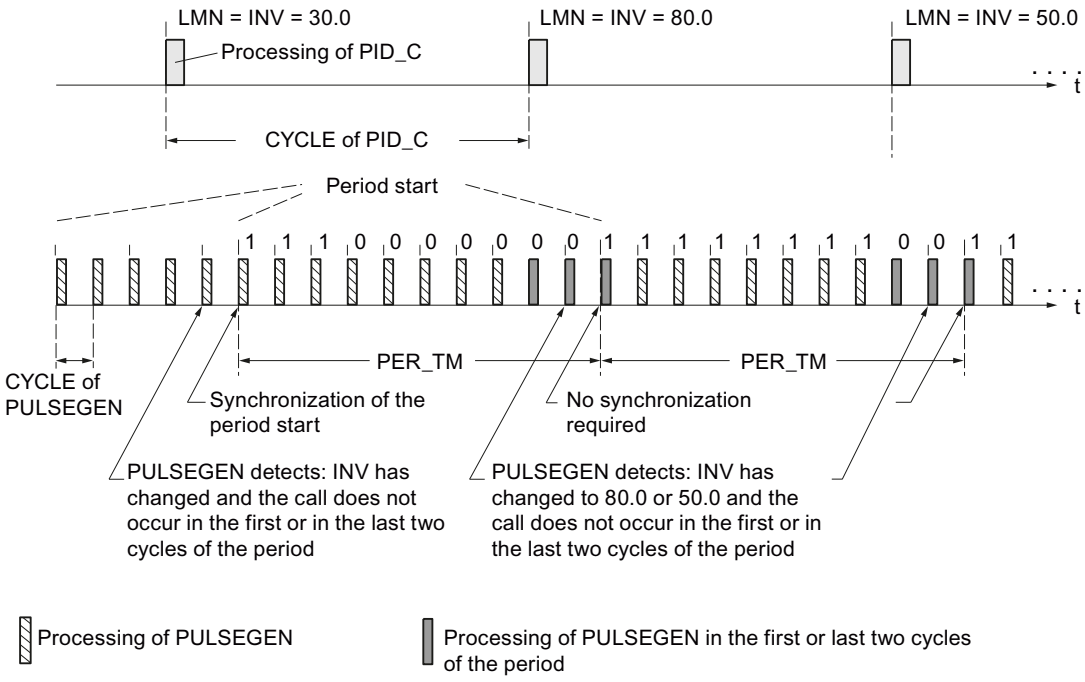

The automatic synchronization is switched off if SYN_ON = FALSE.

Note

The start of a new period and subsequent synchronization usually leads to a certain imprecision when the old value of INV (i.e. of LMN) is mapped to the pulse signal.

PULSEGEN_M modes (S7-300, S7-400)

Operating modes

Depending on the parameters assigned to the pulse shaper, PID controllers with a three-step output or with a bipolar or unipolar two-step output can be configured. The following table illustrates the setting of the switch combinations for the possible modes.

Mode	MAN_ON	STEP3_ON	ST2BI_ON
Three-step control	FALSE	TRUE	Any
Two-step control with bi-polar Manipulating range (-100 % to 100 %)	FALSE	FALSE	TRUE
Two-step control with unipolar Manipulating range (0 % to 100 %)	FALSE	FALSE	FALSE
Manual mode	TRUE	Any	Any

Manual mode in two/three-step control

In the manual mode (MAN_ON = TRUE), the binary outputs of the three-step or two-step controller can be set using the signals POS_P_ON and NEG_P_ON regardless of INV.

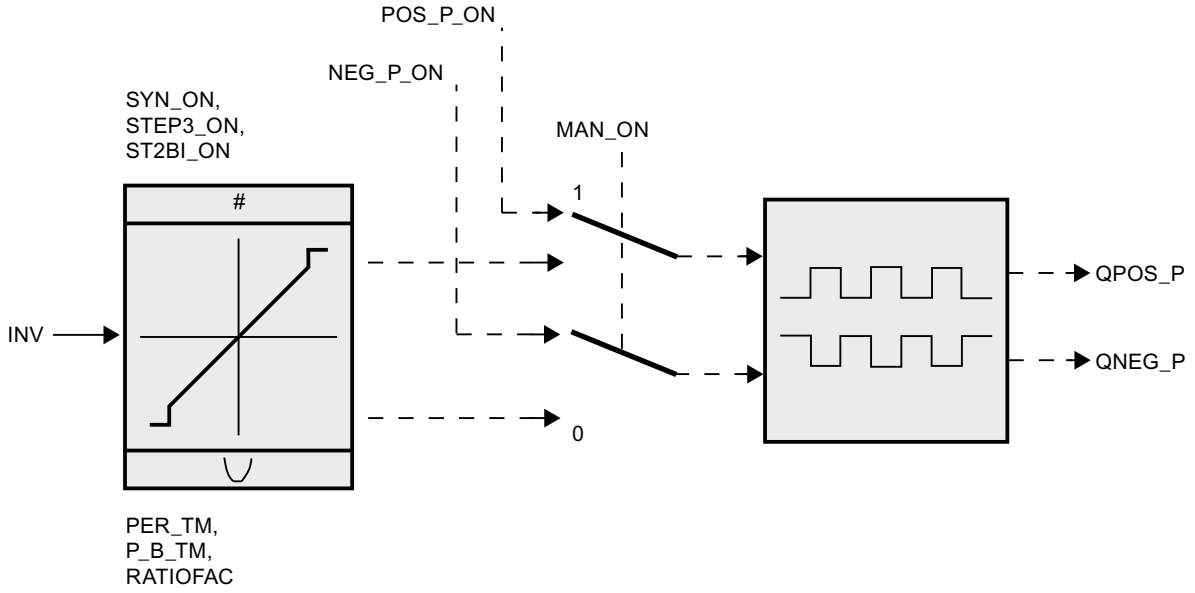

Control	POS_P_ON	NEG_P_ON	QPOS_P	QNEG_P
Three-step control	FALSE	FALSE	FALSE	FALSE
	TRUE	FALSE	TRUE	FALSE
	FALSE	TRUE	FALSE	TRUE
	TRUE	TRUE	FALSE	FALSE
Two-step control	FALSE	Any	FALSE	TRUE
	TRUE	Any	TRUE	FALSE

PULSEGEN_M three-step control (S7-300, S7-400)

Three-step control

Three actuating signal states can be generated in "three-step control" mode, e.g. more - off - less, forward - stop - backward, heat - off - cool. The values of the binary output signals QPOS_P and QNEG_P are assigned to the specific states of the actuator. The table shows the example of a temperature control:

Output signals	Heat	off	Cool
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

Based on the input variable, a characteristic curve is used to calculate pulse duration. The form of the characteristic curve is defined by the minimum pulse time or minimum break time and the ratio factor. The normal value for the ratio factor is 1. The bend points in the characteristic curves are caused by the minimum pulse or minimum break times P_B_TM.

The following figure shows a symmetrical curve of the three-step controller (ratio factor = 1).

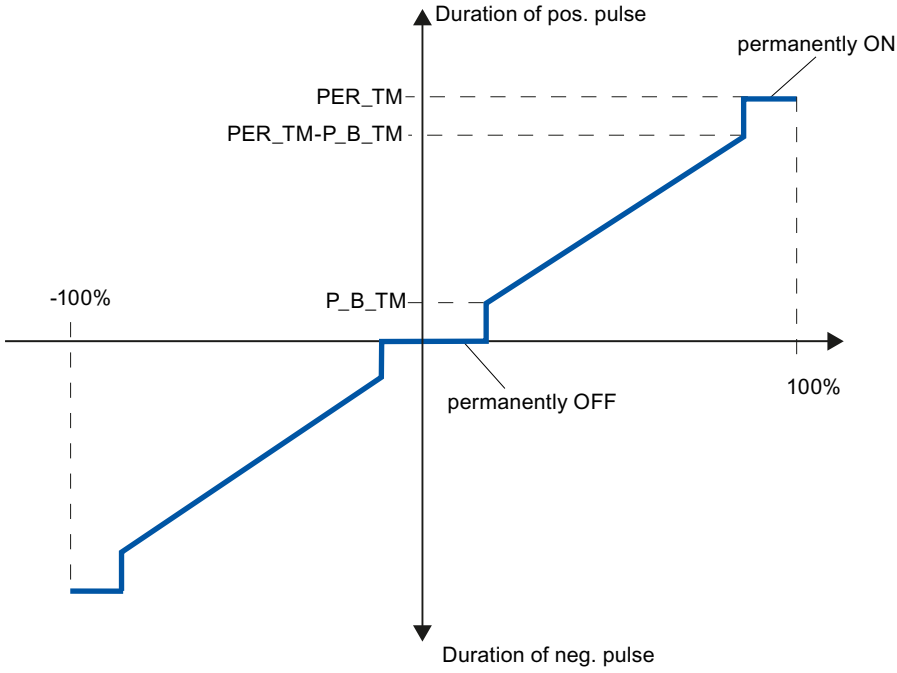

Three-step control asymmetrical

Using the ratio factor RATIOFAC, the ratio of the duration of positive to negative pulses can be changed. In a thermal process, for example, this would allow different system time constants for heating and cooling.

Ratio factor < 1

The pulse duration at the negative pulse output calculated from the input variable multiplied by the period duration is reduced by the ratio factor.

Positive pulse duration = INV /100 * PER_TM

Negative pulse duration = INV / 100 * PER_TM * RATIOFAC

The negative pulse duration is multiplied with the ratio factor.

The following figure shows the asymmetric curve of the three-step controller (ratio factor = 0.5):

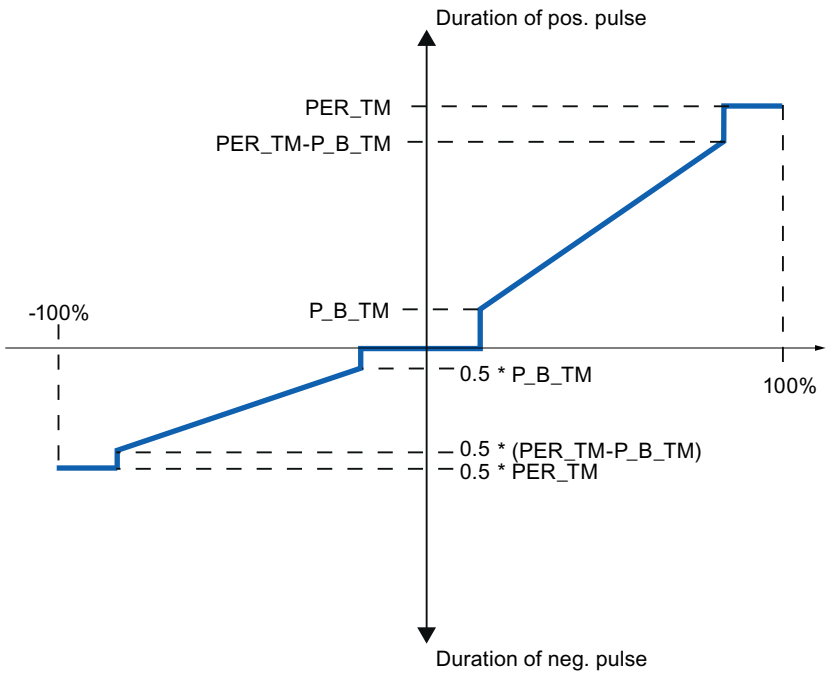

The ratio factor RATIOFAC also influences the minimum pulse or minimum break time. The threshold value for negative pulses is multiplied by the ratio factor.

Ratio factor > 1

The pulse duration at the positive pulse output calculated from the input variable multiplied by the period duration is reduced by the ratio factor.

Positive pulse duration = INV / 100 * PER_TM / RATIOFAC

Negative pulse duration = INV / 100 * PER_TM

The ratio factor RATIOFAC also influences the minimum pulse or minimum break time. The threshold value for positive pulses is divided by the ratio factor.

PULSEGEN_M two-step control (S7-300, S7-400)

In two-step control, only the positive pulse output QPOS_P of PULSEGEN_M is connected to the on/off actuator in question. Depending on the manipulated value range used, the two-step controller has a bipolar or a unipolar manipulated value range.

Two-step control with bipolar manipulated variable range (-100% to 100%)

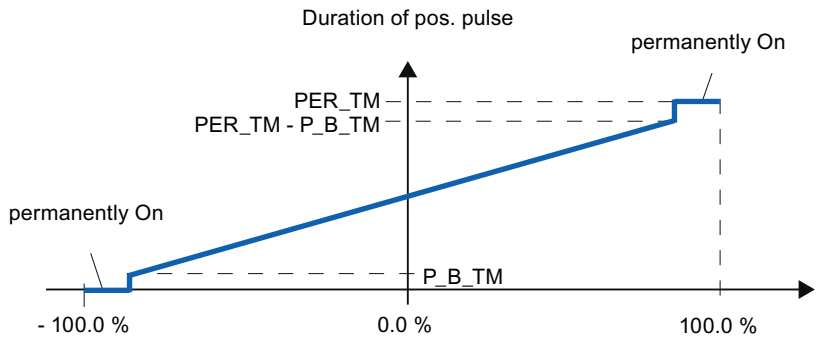

Two-step control with unipolar manipulated variable range (0% to 100%)

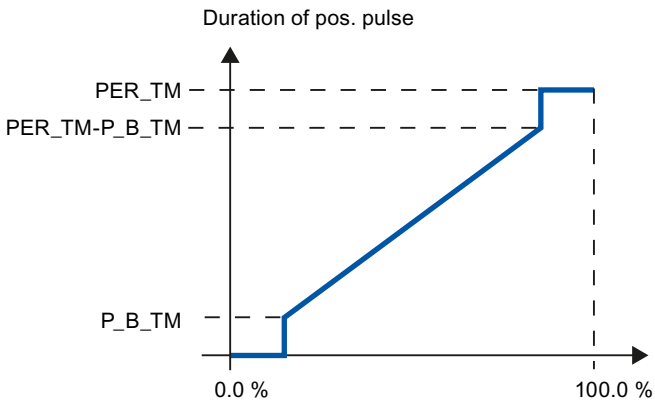

The negated output signal is available at QNEG_P if the connection of the two-step controller in the control loop requires a logically inverted binary signal for the actuating pulses.

Pulse	Actuator On	Actuator Off
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

Input parameters PULSEGEN_M (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable -100.0..100.0 (%)
PER_TM	4.0	TIME	T#1s	Period PER_TM >= 20*CYCLE
P_B_TM	8.0	TIME	T#50ms	Minimum pulse or minimum break time P_B_TM >= CYCLE
RATIOFAC	12.0	REAL	1.0	Ratio factor 0.1..10.0 (dimensionless)

Parameter	Offset	Data type	Preassigned value	Description
STEP3_ON	16.0	BOOL	TRUE	Three-step control ON
ST2BI_ON	16.1	BOOL	FALSE	Activate two-step control for bipolar manipulated variable range
MAN_ON	16.2	BOOL	FALSE	Manual mode ON
POS_P_ON	16.3	BOOL	FALSE	Activate positive pulse
NEG_P_ON	16.4	BOOL	FALSE	Activate negative pulse
SYN_ON	16.5	BOOL	TRUE	Activate synchronization
COM_RST	16.6	BOOL	FALSE	Complete restart
CYCLE	18.0	TIME	T#10ms	Sampling time CYCLE >= 1ms

Output parameters PULSEGEN_M (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
QPOS_P	22.0	BOOL	FALSE	Output signal positive pulse
QNEG_P	22.1	BOOL	FALSE	Output signal negative pulse

RMP_SOAK (S7-300, S7-400)

Description RMP_SOAK (S7-300, S7-400)

You use the RMP_SOAK instruction to specify different setpoints, primarily as a function of the process time.

Operating principle

You specify the time slice of the ramp/soak profile in a global data block. On each call of the instruction, the setpoint valid for this time is output.

The Global data block DB_RMPSK (Page 5787) Is not included in the library. You must create this global data block yourself.

You enter the number of time slices in the global data block in the NBR_PTS tag. You specify the number of the shared block in input parameter DB_NBR.

For each time slice, you enter the setpoint in PI[i].OUTV and the time in PI[i].TMV.

Example of a ramp/soak profile with starting point and time slices.

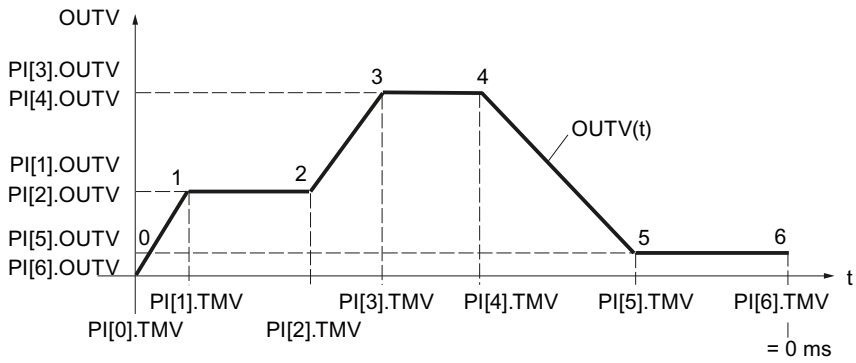

With n time slices, the time value for time slice n = 0 ms.

The method of counting of the time slice values and times is illustrated in the following figure:

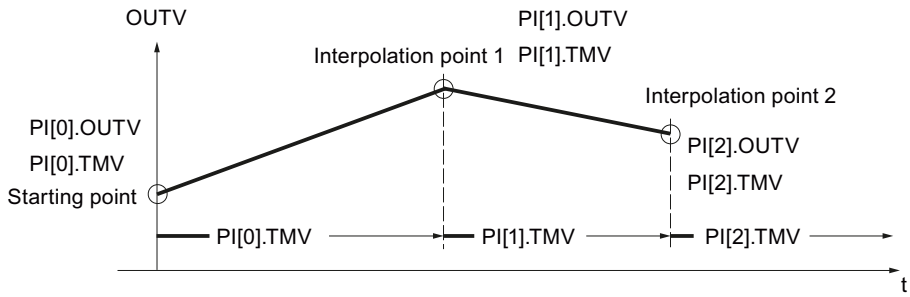

Between the time slices, the ramp/soak function interpolates with 0 ≤ n < (NBR_PTS - 1) according to the following function:

$$OUTV(t) = PI[n + 1].OUTV - \frac{RS_TM}{PI[n].TMV} (PI[n + 1].OUTV - PI[n].OUTV)$$

Note

The instruction does not check whether a global DB with the number DB_NBR actually exists or whether the DB_NBR.NBR_PTS parameter (number of time slices) is suitable for the data block length. If the parameter assignment is incorrect, the CPU changes to the STOP operating mode with the message "Internal system error".

Ramp/soak functions

The instruction has the following functions:

- Activate ramp/soak for single execution
- Pre-assign fixed value to the ramp/soak output
- Activate cyclic operation of the ramp/soak
- Hold processing of the ramp/soak

- Define the processing step and time (the residual time RS_TM and the time slice number TM_SNBR are redefined)
- Update the total processing time and remaining total time

The following table applies for the significance of the control inputs when setting a desired operating mode:

Operating mode	RMPSK_ ON	DFOUT_ ON	RMP_ HOLD	CONT_ ON	CYC_ ON	TUPDT_ ON	Output signal OUTV
Ramp/soak on	TRUE	FALSE	FALSE	**)	FALSE	**)	OUTV(t) End value is held after the end of processing.
Pre-assign output	TRUE	TRUE	**)	**)	**)	**)	DF_OUTV
Cyclic operation on	TRUE	FALSE	FALSE	**)	TRUE	**)	OUTV(t) After end: Automatic start
Hold ramp/soak	TRUE	FALSE	TRUE	FALSE	**)	**)	Current value of OUTV(t) is held *)
Pre-assign processing step and processing time	TRUE	FALSE	TRUE	TRUE	**)	**)	OUTV (old) *)
			FALSE				Operation is continued with newly defined values.
Update total time	**)	**)	**)	**)	**)	FALSE	Does not influence OUTV
						TRUE	Does not influence OUTV

*) The ramp/soak profile does not have the form configured by the user until the next time slice.

**) The respective operating mode set is executed irrespective of the significance of the control signals in the fields marked with **)

Restart

During a complete restart, output OUTV is reset to 0.0. When DFOUT_ON=TRUE, DF_OUTV is output. The time intervals (0 to NBRPTS-1) between the time slices 0 to NBRPTS are totaled and made available under T_TM. The output QR_S_ACT is reset. The outputs NBR_ATMS and RS_TM are pre-assigned with 0.

Activating the ramp/soak

On a positive edge at RMPSK_ON, the ramp/soak function is activated. The ramp/soak profile is terminated after the last time slice has been reached. In order to restart the ramp/soak function, you must first set RMPSK_ON = FALSE and then RMPSK_ON = TRUE.

Pre-assigning the output, starting the ramp/soak profile

If the ramp/soak profile is to start with a particular output value, DFOUT_ON = TRUE must be set. In this case, signal value DF_OUTV is applied at the output of the ramp/soak.

Note

The signal for the output of constant setpoint DFOUT_ON has a higher priority than the start signal for ramp/soak RMPSK_ON.

On changeover (DFOUT_ON = FALSE), OUTV moves linearly from the specified setpoint (DF_OUTV) to the output value of the current time slice number PI[NBR_ATMS].OUTV.

The internal time processing continues running even when a fixed setpoint (RMPSK_ON = TRUE and DFOUT_ON = TRUE) is switched through.

On the start of the ramp/soak profile (RMPSK_ON = TRUE), the fixed setpoint DF_OUTV continues to be output until DFOUT_ON changes from TRUE to FALSE after the duration of T*.

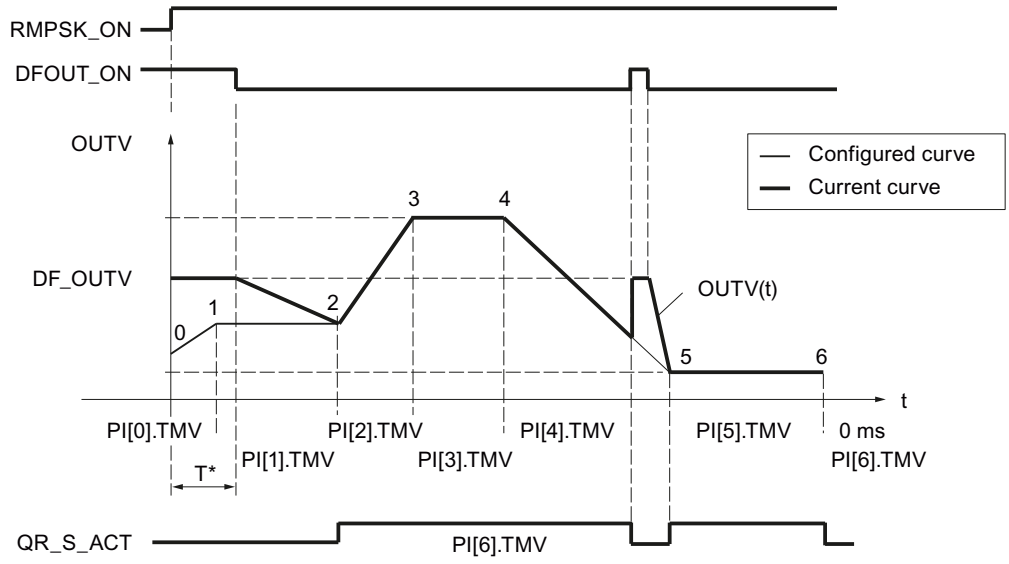

At this moment, the time PI[0].TMV and a part of the time PI[1].TMV has expired. OUTV is moved from DF_OUTV to PI[2].OUTV, which means to the setpoint at time slice 2.

The configured ramp/soak profile is output only starting from time slice 2. The agreement of the configured and current ramp/soak profile is displayed by setting QR_S_ACT = TRUE. On a positive edge at DFOUT_ON during processing of the ramp/soak profile, the output value OUTV jumps to DF_OUTV without a delay.

Cyclic operation activated

If "Cyclic repetition" operating mode (CYC_ON = TRUE) is activated, the ramp/soak function automatically returns to the starting point after the last time slice value has been output and begins a new run.

There is no interpolation between the last time slice and the starting point. For a smooth transition, the following must be true: PI[NBR_PTS].OUTV = PI[0].OUTV.

Hold ramp/soak

With RMP_HOLD = TRUE, the value of the output variable (including the time processing) is frozen. When RMP_HOLD = FALSE is reset, the function resumes at the interruption point PI[x].TMV.

The processing time of the ramp/soak profile is extended by the amount of the time T*. The ramp/soak profile has the configured characteristic from the starting point until the positive

edge at RMP_HOLD and from time slice 5* to time slice 6*, i.e. output signal QR_S_ACT has the value TRUE.

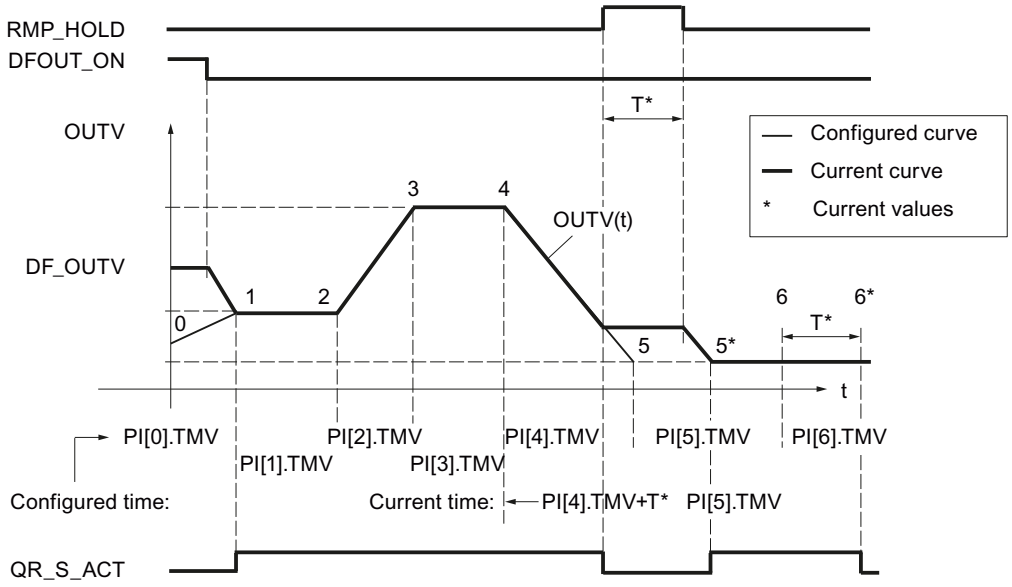

Pre-assigning the processing step and processing time

With CONT_ON, a held ramp/soak profile is restarted at a defined point. If the ramp/soak function is not held, CONT_ON has no effect.

If the control input for continuing (CONT_ON = TRUE) is set, the held ramp/soak function is continued at time slice TM_SNBR. TM_CONT determines the remaining time that the ramp/soak function requires to reach time slice TM_SNBR.

The following figure shows the effect of RMP_HOLD and CONT_ON.

TM_SNBR = 5 and TM_CONT = T*

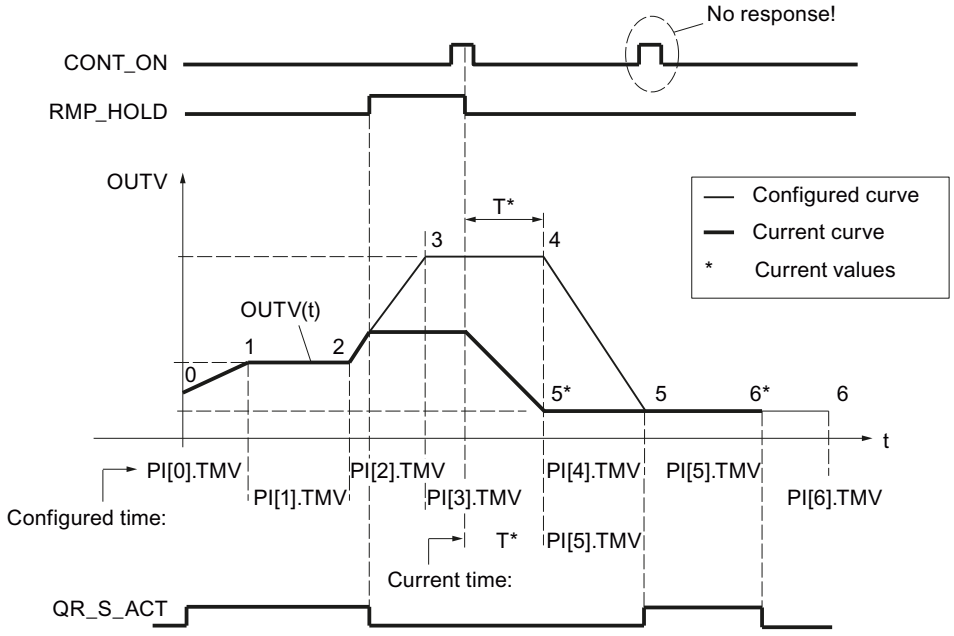

Time slices 3 and 4 are omitted in this cycle of the ramp/soak function.

On a signal change of RMP_HOLD from TRUE to FALSE, the configured characteristic curve is only reached again starting from time slice 5.

Updating the total time and remaining total time

The current time slice number NBR_ATMS, the current remaining time until the time slice is reached (RS_TM), the total time T_TM, and the total remaining time until the end of the ramp/soak profile (RT_TM) are updated in every cycle.

In the case of online changes of PI[n].TMV, the total time and the total remaining time of the ramp/soak profile change. Since the calculation of T_TM and RT_TM with many time slices significantly increases the processing time of the function block, the calculation is only carried after a restart or when TUPDT_ON = TRUE. The time intervals PI[0 ... NBR_PTS].TMV between the individual time slices are added up and indicated at the total time T_TM and total remaining time RT_TM outputs.

Please note that determining the total times requires significant CPU runtime.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters RMP_SOAK (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
DF_OUTV	0.0	REAL	0.0	Default setting of output variable Engineering value range
DB_NBR	4.0	BLOCK_DB	DB 1	Data block number, shared (time slice values) CPU-dependent
TM_SNBR	6.0	INT	0	No. of next time slice for continuation 0 - 255
TM_CONT	8.0	TIME	T#0s	Remaining time for continuing until time slice TM_SNBR Engineering value range
DFOUT_ON	12.0	BOOL	FALSE	Default setting of output variable On
RMPSK_ON	12.1	BOOL	FALSE	Ramp/soak function ON
HOLD	12.2	BOOL	FALSE	Hold the output variable
CONT_ON	12.3	BOOL	FALSE	Continue
CYC_ON	12.4	BOOL	FALSE	Cyclic repetition ON
TUPDT_ON	12.5	BOOL	FALSE	Update total time after change in times (extended block run time!)
COM_RST	12.6	BOOL	FALSE	Complete restart
CYCLE	14.0	TIME	T#1s	Sampling time Engineering value range ≥ 1 ms

Output parameters RMP_SOAK (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	18.0	REAL	0.0	Output variable
QR_S_ACT	22.0	BOOL	FALSE	ramp/soak is being executed
NBR_ATMS	24.0	INT	0	Current time slice number (time slice that is being approached)
RS_TM	26.0	TIME	T#0s	Residual slice time
T_TM	30.0	TIME	T#0s	Shared data: Total time ΣTMVI
RT_TM	34.0	TIME	T#0s	Shared data: Residual total time ΣTMVI

The time slices as well as the number of time slices NBRPTS are stored in a shared data block DB_NBR. Output begins with time slice 0 and ends with the time slice NBR_PTS.

Global data block DB_RMPSK (S7-300, S7-400)

The time slices for the ramp/soak function (RMP_SOAK) are specified in a global data block. This global data block is not included in the library. You must create this global data block yourself according to the following scheme and adapt it to your application.

Parameter	Data type	Default	Description
NBR_PTS	INT	4	Number of time slices -1 1 - 255
PI	Array [0...NBR_PTS] of Struct		Time slice data structure. The array must be declared by 0...NBR_PTS.
PI[0].OUTV	REAL	0.0	Output variable [0] Starting point Engineering value range
PI[0].TMV	TIME	T#1s	Time value [0] Starting point
PI[1].OUTV	REAL	0.0	Output variable [1] Time slice 1 Engineering value range
PI[1].TMV	TIME	T#1s	Time value [1] Time slice 1
PI[2].OUTV	REAL	0.0	Output variable [2] Time slice 2 Engineering value range
PI[2].TMV	TIME	T#1s	Time value [2] Time slice 2
PI[3].OUTV	REAL	0.0	Output variable [3] Time slice 3 Engineering value range
PI[3].TMV	TIME	T#1s	Time value [3] Time slice 3
PI[4].OUTV	REAL	0.0	Output variable [4] Time slice 4 Engineering value range
PI[4].TMV	TIME	T#0s	Time value [4] Time slice 4 0 ms

Alternatively, you can generate the global data block using an external source file. To do so, proceed as follows:

1. Copy the text below to the clipboard.
2. Open an external text editor.
3. Paste the copied text from the clipboard to the text editor.
4. Save the file with the file name extension "DB":
5. Open the "External sources" folder in the project tree of the TIA Portal.
6. Double-click the command "Add new external file".
   The "Open" dialog is displayed.
7. Navigate to the external source file that you have created and select it.
8. Confirm your selection with "Open".
9. Select the external source file.
10. Select the command "Generate blocks from source" in the shortcut menu.
11. A security prompt informs you that any existing blocks will be overwritten.

12. Click "OK" in response to the security prompt.
13. Adapt the created data block to your application.

```
DATA_BLOCK "DB_RMPSK"
TITLE = data block ramp soak
{ S7_Optimized_Access := 'FALSE' }
NAME : DB_RMPSK
 STRUCT
 NBR_PTS : Int; // number of points (excluding starting point)
 PI : Array[0..4] of Struct
 OUTV : Real; // output variable
 TMV : Time; // output time value
 END_STRUCT;
 END_STRUCT;


BEGIN
 NBR_PTS := 4;
 PI[0].TMV := T#1s;
 PI[1].TMV := T#1s;
 PI[2].TMV := T#1s;
 PI[3].TMV := T#1s;
 PI[4].TMV := T#0s;
END_DATA_BLOCK
```

ROC_LIM (S7-300, S7-400)

Description ROC_LIM (S7-300, S7-400)

Ramp functions are used when the process does not tolerate an input step. This is for example the case when a gearbox is inserted between a motor and the load to be driven and an excessively rapid increase of the motor speed would result in overloading of the gearbox. The instruction limits the rate of change of an output value. Step jump functions become ramp functions.

Operating principle

A total of four ramps can be assigned for the positive and negative value range of input variable INV for use in determining the output variable:

- UPRLM_P: Rising values in the positive range
  OUTV > 0 and |OUTV| rising
- DNRLM_P: Falling values in the positive range
  OUTV > 0 and |OUTV| falling
- UPRLM_N: Rising values in the negative range
  OUTV < 0 and |OUTV| rising
- DNRLM_N: Falling values in the negative range
  OUTV < 0 and |OUTV| falling

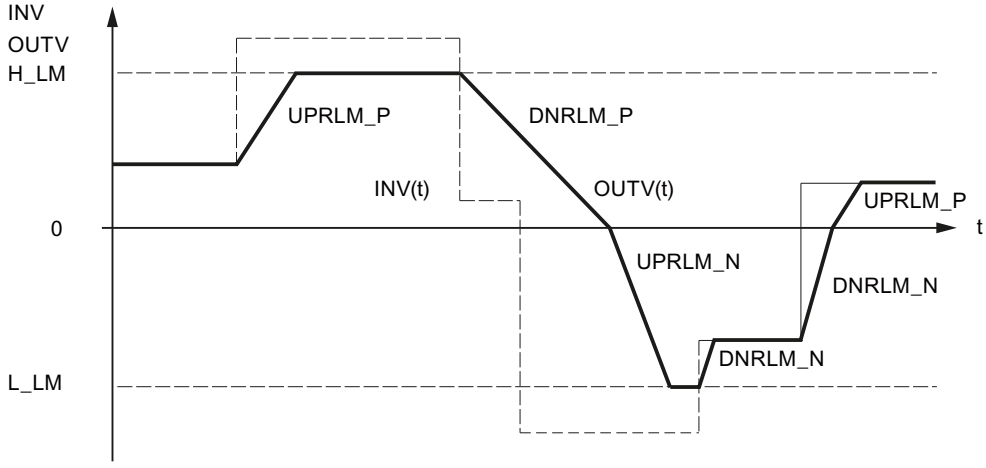

The assigned values of the ramps refer to a rising or falling slope of the output variable per second.

If UPRLM_P = 10.0 and |INV| are rising, OUTV may increase by a maximum of 10.0 in 1 s.

If the input variable rises at a higher rate, OUTV is increased as follows, according to the sampling time CYCLE per call:

CYCLE [s]	Increase of OUTV
1	10.0
0.1	1.0
0.01	0.1

The limiting of the rate of change is indicated in the QUPRLM_P, QDNRLM_P, QUPRLM_N, and QDNRLM_N parameters.

The output variable can be limited by H_LM and L_LM. This is indicated in the QH_LM and QL_LM parameters.

Functions of the instruction

You can use the DFOUT_ON, TRACK, and MAN_ON parameters to influence how the instruction works.

- DFOUT_ON = TRUE
  If DFOUT_ON = TRUE is set, DF_OUTV is output at output OUTV; on a falling edge, OUTV moves from DF_OUTV to INV, and on a rising edge, OUTV moves from INV to DF_OUTV.
- TRACK = TRUE
  Bit TRACK = TRUE is set in order to track OUTV = INV. Since the input variable is switched through directly to the output variable, step changes in the input variable are also output.
- MAN_ON = TRUE
  With MAN_ON = TRUE, you can perform a bumpless changeover between manual and automatic mode. For this purpose, the ROC_LIM instruction is inserted in the setpoint branch directly upstream of the control deviation. The process value is interconnected with input PV and the manual-automatic bit, e.g., LMNGEN_C.MAN_ON is interconnected with input MAN_ON.

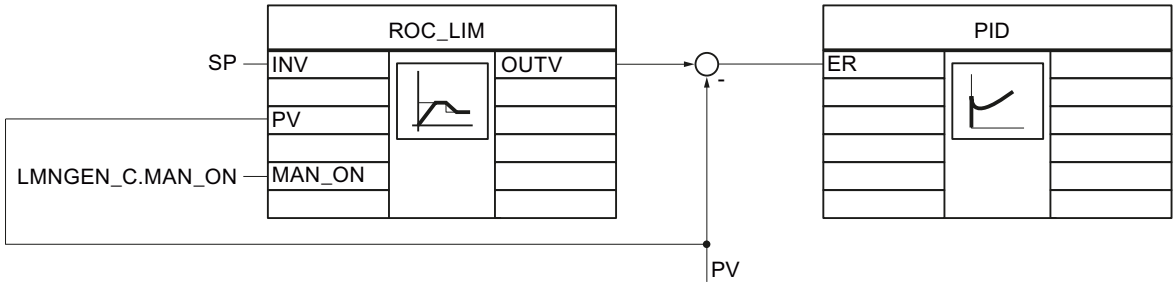

On changeover to manual mode (MAN_ON = TRUE), the value applied to input PV is switched through immediately to output OUTV. Since the setpoint and the process value are equal, the control deviation becomes zero and the controller is in a steady-state idle condition. On reset to automatic mode (MAN_ON = FALSE), output OUTV moves from the current value PV to input value INV along a ramp. This causes a bumpless changeover of the controller structure from manual to automatic mode.
When MAN_ON = TRUE, the H_LM and L_LM limits have no effect; DFOUT_ON is not taken into consideration.

Example 1

When MAN_ON, TRACK, DFOUT_ON equal FALSE , the signal outputs exhibit the following characteristics:

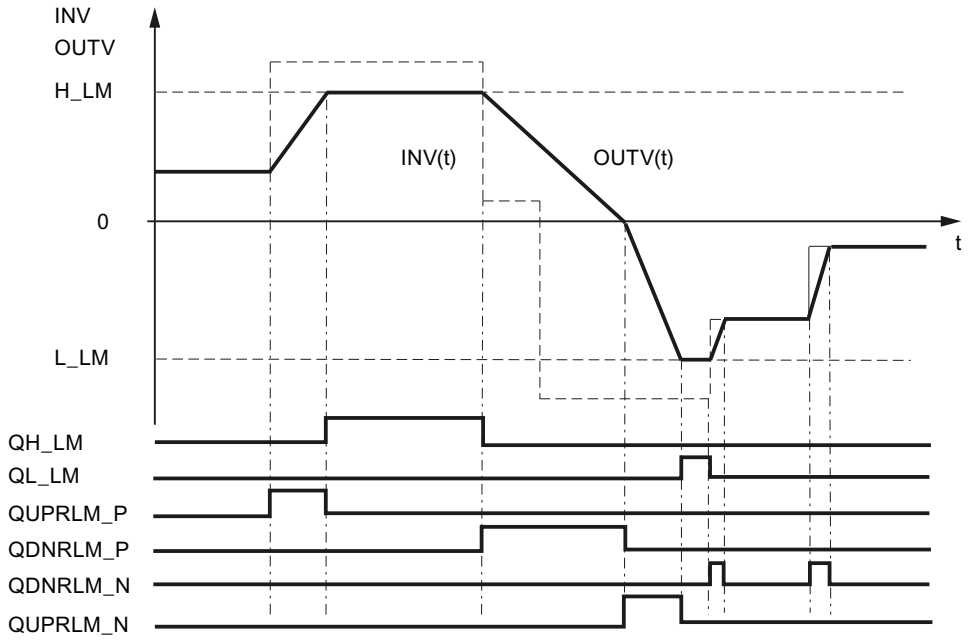

Example 2

Smooth manual-automatic changeover

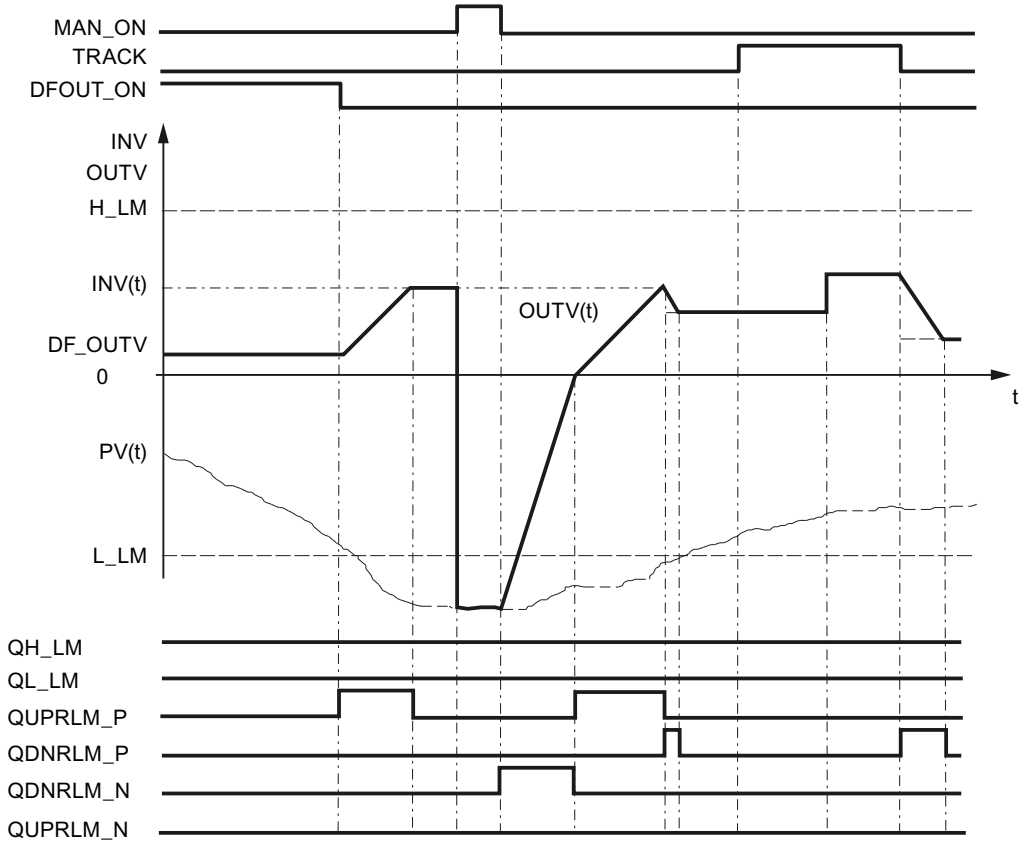

Example 3

Ramp that is only used in the positive range of values.

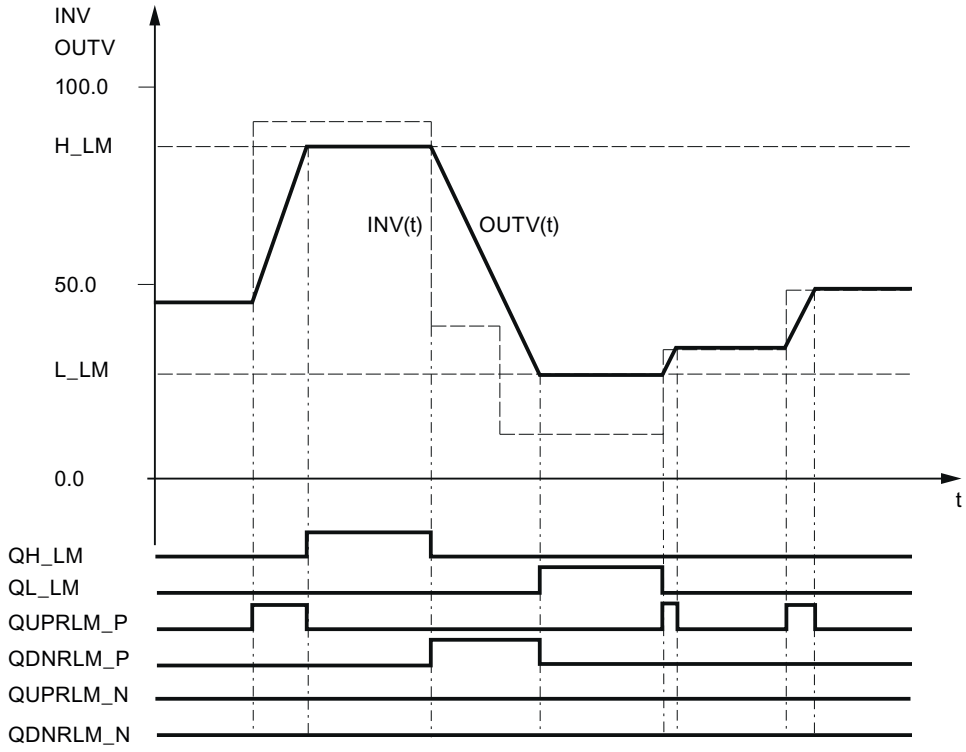

Complete restart

During a complete restart, the output OUTV is reset to 0.0. If DFOUT_ON = TRUE is set, DF_OUTV is output. All the signal outputs are set to FALSE.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters ROC_LIM (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
UPRLM_P	4.0	REAL	10.0	Rise limiter in the positive range Increment per second [1/s] > 0.0

Parameter	Offset	Data type	Preassigned value	Description
DNRLM_P	8.0	REAL	10.0	Fall limiter in the positive range Increment per second [1/s] > 0.0
UPRLM_N	12.0	REAL	10.0	Rise limiter in the negative range Increment per second [1/s] > 0.0
DNRLM_N	16.0	REAL	10.0	Fall limiter in the negative range Increment per second [1/s] > 0.0
H_LM	20.0	REAL	100.0	High limit Engineering value range > L_LM
L_LM	24.0	REAL	0.0	Low limit Engineering value range < H_LM
PV	28.0	REAL	0.0	Controlled variable Engineering value range
DF_OUTV	32.0	REAL	0.0	Default setting of output variable Engineering value range
DFOUT_ON	36.0	BOOL	FALSE	Default setting of output variable On
TRACK	36.1	BOOL	FALSE	Tracking OUTV=INV
MAN_ON	36.2	BOOL	FALSE	Manual mode ON
COM_RST	36.3	BOOL	FALSE	Complete restart
CYCLE	38.0	TIME	T#1s	Sampling time

Output parameters ROC_LIM (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	42.0	REAL	0.0	Output variable
QUPRLM_P	46.0	BOOL	FALSE	Rise limiter in the positive range triggered
QDNRLM_P	46.1	BOOL	FALSE	Fall limiter in the positive range triggered
QUPRLM_N	46.2	BOOL	FALSE	Rise limiter in the negative range triggered
QDNRLM_N	46.3	BOOL	FALSE	Fall limiter in the negative range triggered
QH_LM	46.4	BOOL	FALSE	High limit triggered
QL_LM	46.5	BOOL	FALSE	Low limit triggered

SCALE_M (S7-300, S7-400)

Description SCALE_M (S7-300, S7-400)

In the case of process values, the value supplied by the encoder often lies in a range that is unfavorable for the user (for example, 0 to 10 V corresponds to 0 to 1200 ºC or 0 to 10 V corresponds to 0 to 3000 rpm). By adapting the setpoint or the process value, both process variables lie in the same range of values.

Operating principle

The instruction scales an analog input variable. Analog input variable INV is transformed to output variable OUTV based on the scaling line. The scaling line is defined by the slope (FACTOR) and the distance (OFFSET) between OUTV when INV=0 and coordinate axis OUTV=0 is defined.

Formula:

OUTV = INV * FACTOR + OFFSET

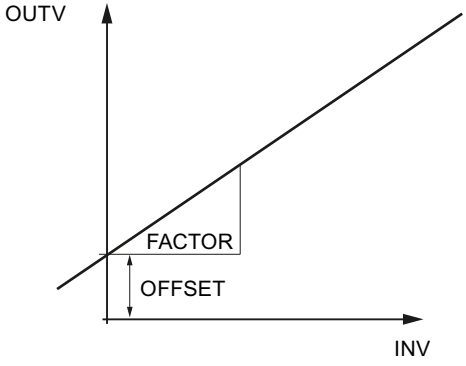

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters SCALE_M (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
FACTOR	4.0	REAL	1.0	Scaling factor
OFFSET	8.0	REAL	0.0	Offset Engineering value range

Output parameters SCALE_M (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV	12.0	REAL	0.0	Output variable

SP_GEN (S7-300, S7-400)

Description SP_GEN (S7-300, S7-400)

An output value can be changed using the SP_GEN instruction and two inputs in order to specify the setpoint manually. To carry out a change in small steps, the instruction should have a sampling time of ≤100 ms.

Operating principle

DFOUT_ON, OUTVUP, and OUTVDN have the following effect on OUTV:

DFOUT_ON	OUTVDN	OUTVUP	OUTV
TRUE	Any	Any	DF_OUTV
FALSE	TRUE	TRUE	OUTV unchanged
	FALSE		OUTV rising
	TRUE	FALSE	OUTV falling
	FALSE		OUTV unchanged

If DFOUT_ON = TRUE is set, DF_OUTV is output at the output. The change in OUTV is a step change. The changeover to DFOUT_ON = FALSE is bumpless.

The OUTVUP input can be used to continuously increase output variable OUTV within the limits H_LM and L_LM; accordingly, the OUTVDN output is used for reducing. The rate of change of OUTV depends on the limits.

- During the first 3 s after setting of OUTVUP or OUTVDN:
  dLMN/dt = (LMN_HLM - LMN_LLM) / 100 s
- Afterwards:
  dLMN/dt = (LMN_HLM - LMN_LLM) / 10 s

The figure shows the change in the output value when OUTVUP is set.

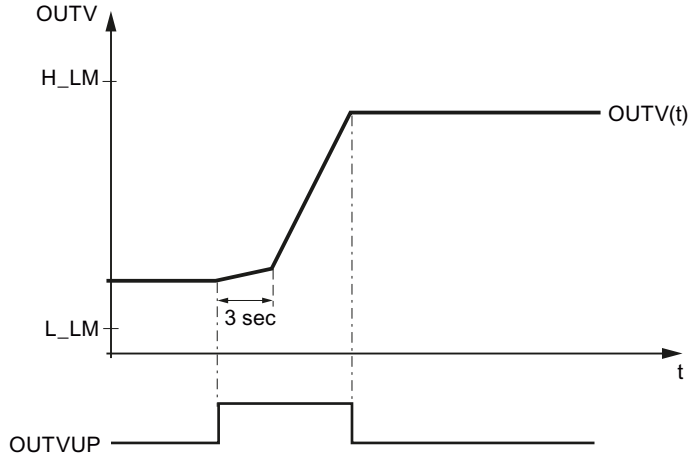

OUTVUP and OUTVDN are executed only if DFOUT_ON = FALSE.

Output variable OUTV is limited by H_LM and L_LM. This is indicated in the QH_LM and QL_LM parameters.

The figure shows how OUTVUP, OUTVDN, and DFOUTV_ON. affect OUTV.

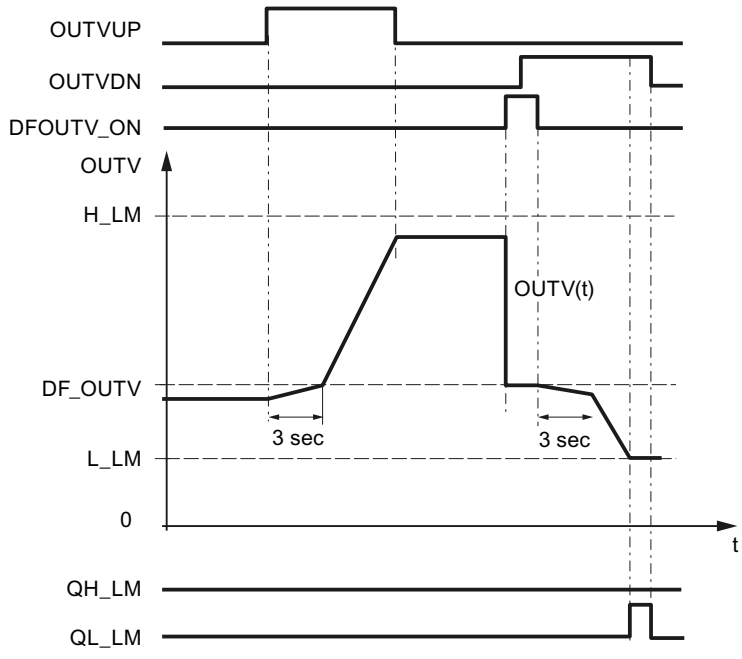

Complete restart

During a complete restart, the output OUTV is set to 0.0. If DFOUT_ON = TRUE, DF_OUTV is output. The limits remain valid at a compete restart.

Block-internal limits

No values are limited block-internally. The parameters are not checked.

Input parameters SP_GEN (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
DF_OUTV	0.0	REAL	0.0	Default setting of output variable Engineering value range
H_LM	4.0	REAL	100.0	High limit Engineering value range > L_LM
L_LM	8.0	REAL	0.0	Low limit Engineering value range < H_LM
OUTVUP	12.0	BOOL	FALSE	OUTV up
OUTVDN	12.1	BOOL	FALSE	OUTV down
DFOUT_ON	12.2	BOOL	FALSE	Default setting of output variable On
COM_RST	12.3	BOOL	FALSE	Complete restart
CYCLE	14.0	TIME	T#100ms	Sampling time

Output parameters SP_GEN (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value (on complete restart)	Description
OUTV	18.0	REAL	0.0	Output variable
QH_LM	22.0	BOOL	FALSE	High limit triggered
QL_LM	22.1	BOOL	FALSE	Low limit triggered

SPLT_RAN (S7-300, S7-400)

Description SPLT_RAN (S7-300, S7-400)

The instruction is required, for example to split a temperature control loop into an actuator for cooling and an actuator for heating.

Operating principle

The manipulated variable of a PID controller is spilt into several ranges. The instruction must be called once per range and be interconnected with manipulated variable processing block LMNGEN_C or LMNGEN_S. If you interconnect LMNGEN_S, position feedback must be activated (LMNR_ON = TRUE).

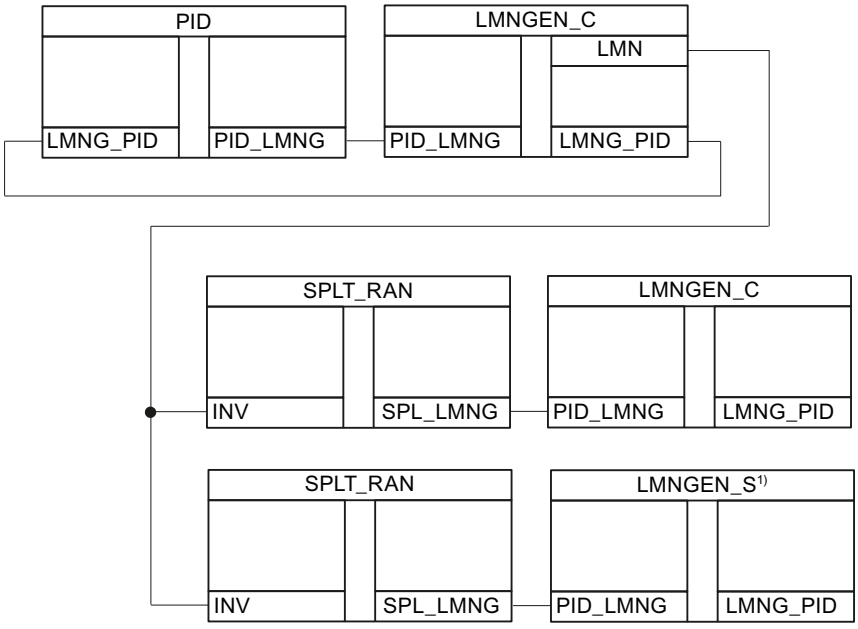

If input value INV is within the range STR_INV and EDR_INV, output value SPL_LMNG.PID_OUTV is output within the range STR_OUTV and EDR_OUTV. If INV < STR_INV,, then STR_OUTV is output. If INV > STR_INV,, then EDR_OUTV is output.

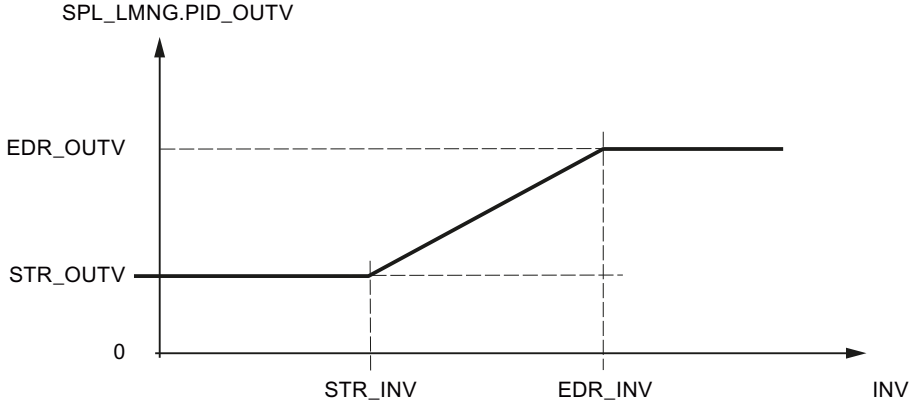

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters SPLT_RAN (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
STR_INV	4.0	REAL	0.0	Start of range INV Engineering value range
EDR_INV	8.0	REAL	50.0	End of range INV Engineering value range
STR_OUTV	12.0	REAL	0.0	Start of range OUTV Engineering value range
EDR_OUTV	16.0	REAL	100.0	End of range OUTV Engineering value range

Output parameters SPLT_RAN (S7-300, S7-400)

Parameter	Offset	Data type	Description
SPL_LMNG	20.0	STRUC	PID-LMNGEN interface

SWITCH (S7-300, S7-400)

Description SWITCH (S7-300, S7-400)

The instruction is used as an input and/or output multiplexer of 2 input/output variables.

Operating principle

The instruction connects one of two analog input values to one of two output values. The INV1_ON and OUTV1_ON switches determine how the instruction operates.

INV1_ON	OUTV1_ON	OUTV1	OUTV2
FALSE	FALSE	Unchanged	INV2
TRUE	FALSE	Unchanged	INV1
FALSE	TRUE	INV2	Unchanged
TRUE	TRUE	INV1	Unchanged

Unchanged means that the value from the last cycle is retained in the output parameter.

Complete restart

On a complete restart, OUTV1 = 0.0 and OUTV2 = 0.0 are set.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters SWITCH (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
INV1	0.0	REAL	0.0	Input variable 1 Engineering value range
INV2	4.0	REAL	0.0	Input variable 2 Engineering value range
INV1_ON	8.0	BOOL	FALSE	Input variable INV1 is switched through
OUTV1_ON	8.1	BOOL	FALSE	Output variable OUTV1 is switched through
COM_RST	8.2	BOOL	FALSE	Complete restart

Output parameters SWITCH (S7-300, S7-400)

Parameter	Offset	Data type	Preassigned value	Description
OUTV1	10.0	REAL	0.0	Output variable 1
OUTV2	14.0	REAL	0.0	Output variable 2

LP_SCHED_M (S7-300, S7-400)

Description LP_SCHED_M (S7-300, S7-400)

If many controllers with different sampling times, in particular slow control systems with high sampling times, have to be called, the scope of the priority class model with regard to usable cyclic interrupt levels is insufficient. The controller call distribution function (LP_SCHED) now makes it possible to integrate several controllers with different sampling times in a cyclic interrupt level. The individual controllers can then called up cyclically with their sampling times.

Use of the controller call distribution function is not mandatory. The controller instructions can also be called directly from the OB without the distribution function.

The call distribution of several controllers on a cyclic interrupt level is implemented in the LP_SCHED instruction. The instruction must be called up **before** all control loops. The data for the individual controller calls is stored in a global data block (DB_LOOP).

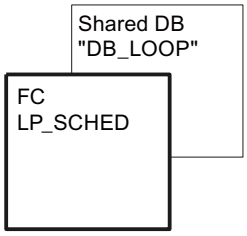

Global data block for DB_LOOP (Page 5806) is not included in the library. You must create this global data block yourself.

The call distributor processes the global data block and sets the ENABLE bits depending on the sequence and the parameterized sampling times of the controllers. The time cycle of the cyclic interrupt level is subordinated by this. The individual control loops on this cyclic interrupt level are called up and processed according to their set sampling time. The ENABLE bit has to be reset again after the block call. The block calls and the reset of the ENABLE bits have to be programmed.

The call of individual control loops can be manually blocked. Individual control loops can also be reset (restart).

Note

The instruction does not check whether a global DB with the number DB_NBR actually exists or whether the GLP_NBR parameter (largest control loop number) is suitable for the data block length. If the parameter assignment is incorrect, the CPU changes to STOP with an "internal system error".

Operating principle LP_SCHED_M (S7-300, S7-400)

Restart

When COM_RST = TRUE, the following pre-assignments are set:

Current control loop number: ALP_NBR = 0

The call data of all control loops up to **GLB_NBR** is pre-assigned as follows:

Enable:	ENABLE =	NOT MAN_DIS
Sampling time:	CYCLE =	GV(MAN_CYC)
Restart:	COM_RST =	TRUE
Local call number:	ILP_COU =	0
GV(MAN_CYC) = MAN_CYC is rounded to an integer multiple of TM_BASE*GLP_NBR		

Control loop call via LP_SCHED_M

The LP_SCHED instruction is included by means of three input parameters in the call scheme of the CPU. The time cycle of the cyclic interrupt level is specified at the TM_BASE input. The execution call of the assigned control loops is carried out via a conditional block call from one cyclic interrupt level (for example OB 35). In the course the ENABLE bits in the global data block are sampled.

If the call is carried out from the restart level, input COM_RST = TRUE is set. This call has to be reset again to FALSE in the cyclic interrupt level. The global data block (see previous table) with the time-relevant data of the control loops in the respective cyclic interrupt level is assigned via the input parameter DB_NBR.

Parameter assignment of the control loop call (global DB)

The parameter assignment of the controller call distribution function has to be carried out **without** support by the configuration software.

The data block (DB_LOOP) contains both a parameter that determines the total number of control loops that have to be processed on the respective cyclic interrupt level (max. 256) and a parameter that displays the control loop that is currently being processed:

GLP_NBR	Largest control loop number
ALP_NBR	Number of the control loop processed in the cycle

The number of the respective control loop results from the positioning of its call data in the sequence of the entries in the DB.

The call data of the individual control loops resides structured in the LOOP_DAT field. If control loops are to be added, the field length of LOOP_DAT has to be adapted. To this purpose the ARRAY data type has to be adapted in the declaration view. For example, with 10 control loops, you must enter ARRAY[1..10]. In addition the GLP_NBR parameter has to be adapted in the data view. It may never be larger than the field length.

The parameters COM_RST and CYCLE in the call data must be interconnected to the corresponding input parameters at the FB of the calling control loop. This interconnection is to be programmed by the user. If the ENABLE parameter is set, the corresponding control loop is to be called. The ENABLE bit has to be reset after the controller call. The conditional controller call and the reset of the ENABLE bit have to be programmed by the user.

With the aid of the manually adjustable parameters MAN_CYC/MAN_DIS/MAN_CRST, one can control whether a control loop can or cannot be called. These call data can be modified online - during operation - so long as only the parameters are overwritten and the entire DB is not regenerated. The following meanings apply:

MAN_CYC	The sampling time of the respective controller (is rounded to an integer multiple of TM_BASE * GLP_NBR in CYCLE).
MAN_DIS	Disabling of the controller call
MAN_CRST	Restart for this controller

Call processing

The corresponding control loop is processed in accordance with the previous DB parameter assignment, depending on the value of the ENABLE signal of the controller call data.

The data block is processed from top to bottom. Per cycle the call distribution function moves one control loop number (ALP_NBR) further in the sequence of the DB. The internal counter ILP_COU is counted down by 1 at the same time. If ILP_COU = 0, the call distribution function sets the ENABLE bit of the corresponding control loop. Resetting of the ENABLE bits has to be programmed by the user after the controller call.

The parameter MAN_CYC is transferred to CYCLE during processing:

CYCLE = GV (MAN_CYC), GV = integer multiple

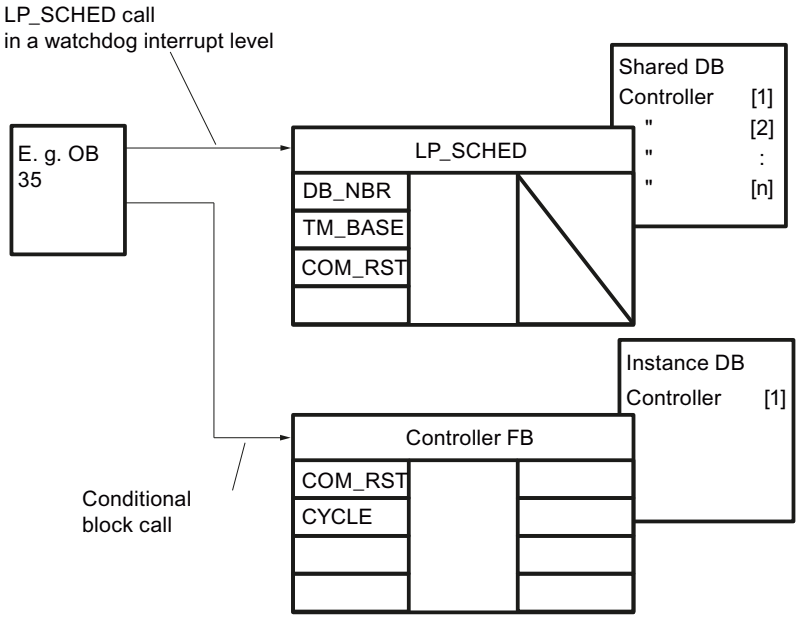

Figure 4-26 Principle of the controller call using the call distribution function LP_SCHED

- Blocking individual control loops:
  If you set the bit "MAN_DIS" by parameter assignment in the DB data block, the "ENABLE" bit is reset to FALSE and the respective control loop is excluded from processing in the call distribution function.
- Resetting individual control loops (restart):
  If you set the "MAN_CRST" bit by parameter assignment in the DB, COM_RST = TRUE and MAN_CRST is subsequently reset. The respective control loop is processed with its restart routine. In the next call cycle, COM_RST = FALSE is also reset automatically.

Note

If a control loop is inserted or deleted, meaning that the whole DB is regenerated, without the call distribution function having to be processed in the restart, the internal control loop counter (ILP_COU[n]) and the parameter for the current control loop number ALP_NBR have to have zero pre-assigned.

Conditions for control loop call via LP_SCHED

In order to ensure that the intervals between the calls of a specific controller remain constant and to utilize the CPU evenly, the cyclic interrupt level may only process one control loop per cycle. The following conditions must therefore be observed during the parameter assignment of the sampling times MAN_CYC in relation to the cycle (TM_BASE):

- The processing times of the individual control loops have to be smaller than the cycle (TM_BASE) of the cyclic interrupt level.
- The sampling time of a control loop (MAN_CYC) has to be an integer multiple (GV) of the product of the time base and number of the controllers to be processed (GLP_NBR): MAN_CYC = GV (TM_BASE * GLP_NBR)

Example for call distribution function

The following example shows the call sequence of four control loops in a cyclic interrupt level (see following figure). Only one control loop is always processed per unit of the time base. The call sequence and thus the displacement times (TD1 ... TD5) result from the sequence of the call data within the global DB.

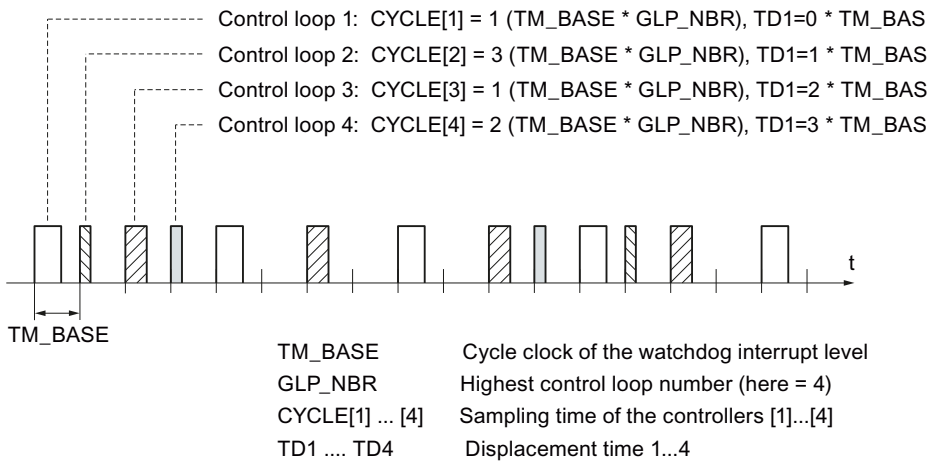

Figure 4-27 Call sequence of four control loops that are processed at varying intervals

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters LP_SCHED_M (S7-300, S7-400)

Parameter	Data type	Description
TM_BASE	TIME	Time base >= 1ms
COM_RST	BOOL	Complete restart
DB_NBR	BLOCK_DB	Data block number

Global data block for DB_LOOP (S7-300, S7-400)

The data for the call distribution with LP_SCHED is specified in a global data block. This global data block is not included in the library. You must create this global data block yourself according to the following scheme and adapt it to your application.

Parameter	Data type	Preassigned value	Description
GLP_NBR	INT	2	Maximum number of controllers 1 - 256
ALP_NBR	INT	0	Number of the current controller 0 - 256
LOOP_DAT	Array [1...GLP_NBR] of Struct		Individual controller data structure. The array must be declared by 1...GLP_NBR.
LOOP_DAT[1].MAN_CYC	TIME	T#1s	Controller 1 data: Manual sampling time >= 1ms
LOOP_DAT[1].MAN_DIS	BOOL	FALSE	Controller 1 data: Disable manual controller call
LOOP_DAT[1].MAN_CRST	BOOL	FALSE	Controller 1 data: Set manual restart
LOOP_DAT[1].ENABLE	BOOL	FALSE	Controller 1 data: Controller enable
LOOP_DAT[1].COM_RST	BOOL	FALSE	Controller 1 data: Restart
LOOP_DAT[1].ILP_COU	INT	0	Controller 1 data: Internal counter
LOOP_DAT[1].CYCLE	TIME	T#1s	Controller 1 data: Sampling time >= 1ms
LOOP_DAT[2].MAN_CYC			Controller 2 data: Manual sampling time
...			

Alternatively, you can generate the global data block using an external source file. To do so, proceed as follows:

1. Copy the text below to the clipboard.
2. Open an external text editor.
3. Paste the copied text from the clipboard to the text editor.
4. Save the file with the file name extension " .DB":
5. Open the "External sources" folder in the project tree of the TIA Portal.
6. Double-click the command "Add new external file".
   The "Open" dialog is displayed.
7. Navigate to the external source file that you have created and select it.
8. Confirm your selection with "Open".
9. Select the external source file.
10. Select the command "Generate blocks from source" in the shortcut menu.
11. A security prompt informs you that any existing blocks will be overwritten.

12. Click "OK" in response to the security prompt.
13. Adapt the created data block to your application.

```
DATA_BLOCK "DB_LOOP"
TITLE = data block loop scheduler
{ S7_Optimized_Access := 'FALSE' }
NAME : DB_LOOP
 STRUCT
 GLP_NBR : Int; // greatest loop number
  ALP_NBR : Int; // actual loop number
 LOOP_DAT : Array[1..2] of Struct
 MAN_CYC : Time; // loop data: manual sample time
 MAN_DIS : Bool; // loop data: manual loop disable
 MAN_CRST : Bool; // loop data: manual complete restart
 ENABLE : Bool; // loop data: enable loop
 COM_RST : Bool; // loop data: complete restart
 ILP_COU : Int; // loop data: internal loop counter
 CYCLE : Time; // loop data: sample time
 END_STRUCT;
 END_STRUCT;


BEGIN
 GLP_NBR := 2;
 LOOP_DAT[1].MAN_CYC := T#1S;
 LOOP_DAT[1].CYCLE := T#1S;
 LOOP_DAT[2].MAN_CYC := T#200MS;
 LOOP_DAT[2].CYCLE := T#200MS;

END_DATA_BLOCK
```

4.2.4.2 Motion Control (S7-300, S7-400)

Easy Motion Control (S7-300, S7-400)

MC_Init (S7-300, S7-400)

Description

The MC_Init prepares the initialization of all axis blocks by setting all bits in the "Init.Ix" bit array of the axis DB (see "Initialization and parameter changes").

Operating principle

instruction MC_Init **prevents the unintentional start of the axis** by setting the error displays "Error" = TRUE and "Err.StoppedMotion" = TRUE in the axis DB. If you want to move the axis after instruction MC_Init has been called, you must first cancel the error displays by acknowledging them (see Error displays and error acknowledgment in the axis DB).

instruction MC_Init checks the parameters in the axis DB. If a parameter error is detected, the "Error" and "Err.ConfigErr" bits are set in the Axis Db for the (group) error display, as well as one of the bits in the "ConfigErr" structure for a more detailed error display.

Parameter errors **cannot be acknowledged**. You must call instruction MC_Init again after correcting the error.

For more information on the error concept, refer to "Error displays and error acknowledgment in the axis DB" and "Parameter errors".

Call

Note

instruction MC_Init may only be called when the axis is at a standstill.

Call instruction MC_Init conditionally in the same program block in which you also call the other instructions of this axis. You must call instruction MC_Init in the following situations:

- After every startup of the CPU (OB 100 and OB 101)
- If you change a parameter that is identified by the following symbol in Configuring the Axis TO without using the configuration software:

Note

Reset the condition used for calling instruction MC_Init once the call has been completed. Otherwise the Easy Motion Control blocks will always repeat only their initialization because the initialization bits in the axis DB are set again.

MC_MoveAbsolute (S7-300, S7-400)

Description

instruction MC_MoveAbsolute can be used to approach a synchronized axis to the absolute target specified in "Position".

Operating principle

On detecting a positive edge at the "Execute" input, instruction MC_MoveAbsolute determines the direction of travel and accelerates the axis in this direction. On reaching the point of deceleration, the axis is slowed down. The specified velocity does not have to be reached during the movement.

Special feature with a linear axis:

- The direction of travel is determined from the setpoint position at the start and the target.

Special features with a rotary axis:

- instruction MC_MoveAbsolute limits the distance by which you can move the axis from a standstill to less than one rotary axis revolution. Use instruction MC_MoveRelative (Page 5812) to position the rotary axis with several revolutions.
- Polarity selection when starting from axis standstill:
  - Positive direction: target approached in a positive direction
  - Negative direction: target approached in a negative direction
  - Shortest distance: The target is approached in the direction in which the distance between start position and target is equivalent to ≤ ½ rev of the rotary axis.
- Polarity selection when overriding a motion
  - Positive direction: target approached in the positive direction. If the previous motion was in the negative direction, the axis is slowed down and the target is approached in the positive direction. If the target is reached exactly when slowing down with a negative travel direction, the axis then travels one revolution in the positive direction.
  - Negative direction: target approached in the negative direction. If the previous motion was in the positive direction, the axis is slowed down and the target is approached in the negative direction. If the target is reached exactly when slowing down with a positive travel direction, the axis then travels one revolution in the negative direction.
  - Shortest distance: The target is approached in the direction in which the distance between start position and target is equivalent to ≤ ½ rev of the rotary axis. If the braking distance is greater, the axis travels to the target in the original direction.
  - If you have selected travel parameters in such a ways that the braking distance of the previous movement is greater than one rotary axis revolution, the target is approached directly after direction reversal.
  - The entire distance between the start position and target may be greater than one rotary axis revolution.

Signal flow diagram for MC_MoveAbsolute

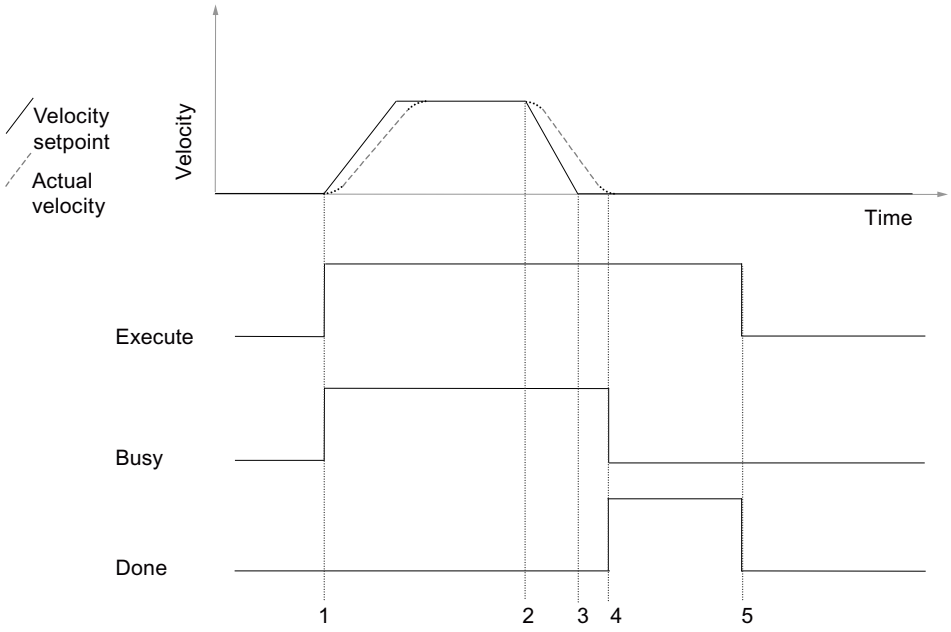

1 Start of the travel FB

2 Brake application point

3 Setpoint has reach target position

4 Axis has reached the target range

5 Reset of "Execute" resets "Done". The process is completed. If "Execute" is reset before the target position has been reached, "Done" is set for one cycle after the target position has been reached and is then reset again.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Start of positioning at a positive edge.
Position	INPUT	REAL	Absolute target position in [length unit] Rotary axis: Start of rotary axis ≤ position < end of rotary axis
Velocity	INPUT	REAL	Axis velocity in [length unit/s]
Acceleration	INPUT	REAL	Axis acceleration in [length unit/s^2]
Deceleration	INPUT	REAL	Axis deceleration in [length unit/s^2]
Direction	INPUT	INT	Polarity selection for travel with rotary axis -1 = Negative direction 0 = Shortest distance 1 = Positive direction

Parameter	Declaration	Data type	Description
Busy	OUTPUT	BOOL	1 = Job in progress.
Done	OUTPUT	BOOL	1 = Job completed without error. The bit remains active at least for one call of the instruction until "Execute" is reset.
CommandAborted	OUTPUT	BOOL	1 = Job was canceled by another job. The bit remains active at least for one call of the instruction until "Execute" is reset.
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active at least for one call of the instruction until "Execute" is reset.

UDT2

A UDT2 data type is listed and reserved in the block properties of MC_MoveAbsolute. This means that no separate UDT2 data type may be used in the same project. This would result in error messages and inconsistencies within the project. If you want to use a UDT2 data type, this UDT must be renamed, for example, to UDT3.

MC_MoveRelative (S7-300, S7-400)

Description

instruction MC_MoveRelative can be used to traverse an axis by the distance that is specified in the "Distance" parameter and that is relative to the setpoint at the start of axis movement. The sign of the "Distance" parameter determines the direction.

Operating principle

On detecting a positive edge at input "Execute", instruction MC_MoveRelative accelerates the axis in the direction that is specified in the "Distance" parameter. On reaching the point of deceleration, the axis is slowed down. The specified velocity does not have to be reached during the movement. The target is determined from the setpoint position at the start of travel and the distance to be traveled.

Special feature of rotary axes:

- You can realize travel over multiple revolutions by setting values in "Distance" that are greater than one revolution of the of the round axis.

Signal flow diagram for MC_MoveRelative

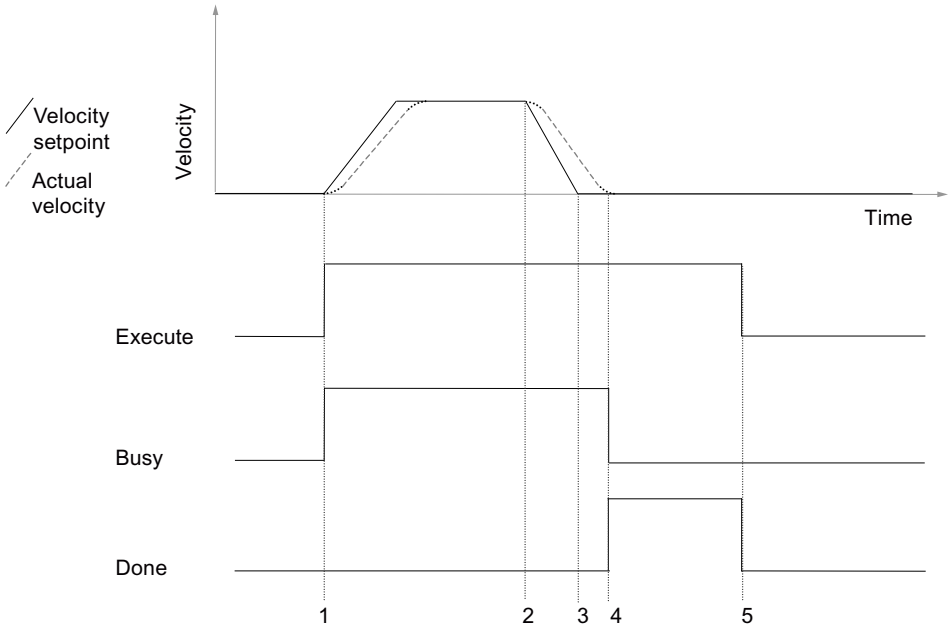

1 Start of the travel FB

2 Brake application point

3 Setpoint has reach target position

4 Axis has reached the target range

5 Reset of "Execute" resets "Done". The process is completed. If "Execute" is reset before the target position has been reached, "Done" is set for one cycle after the target position has been reached and is then reset again.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Start of positioning at a positive edge.
Distance	INPUT	REAL	Distance to be traversed in [length unit], relative to the last position setpoint. The sign determines the direction of travel.
Velocity	INPUT	REAL	Axis velocity in [length unit/s]
Acceleration	INPUT	REAL	Axis acceleration in [length unit/s^2]
Deceleration	INPUT	REAL	Axis deceleration in [length unit/s^2]
Busy	OUTPUT	BOOL	1 = Job in progress.
Done	OUTPUT	BOOL	1 = Job completed without error. The bit remains active at least for one call of the instruction until "Execute" is reset.

Parameter	Declaration	Data type	Description
CommandAborted	OUTPUT	BOOL	1 = Job was canceled by another job. The bit remains active at least for one call of the instruction until "Execute" is reset.
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active at least for one call of the instruction until "Execute" is reset.

UDT2

A UDT2 data type is listed and reserved in the block properties of MC_MoveRelative. This means that no separate UDT2 data type may be used in the same project. This would result in error messages and inconsistencies within the project. If you want to use a UDT2 data type, this UDT must be renamed, for example, to UDT3.

MC_MoveJog (S7-300, S7-400)

Description

instruction MC_MoveJog can be used to move an axis using two level-controlled directional inputs "JogPos" and "JogNeg" (jogging).

Operating principle

Once you have set a directional input, MC_MoveJog accelerates the axis to the specified velocity. The axis moves in this direction as long as the directional input remains set. If you reset the directional input again, the axis slows down.

The two directional inputs are evaluated with the following dependencies:

- As long as one direction is selected, setting the opposite direction has no effect. No error is produced.
- To change direction during travel, set the other directional input to TRUE and at the same time set the originally selected directional input to FALSE.
- A start error will be generated if both inputs are set at the start of a movement ("Err.StartErr"; refer to Errors with soft stop).

Special features of linear axes:

- If the maximum travel distance during travel with a linear axis is exceeded, the MC_MoveJog instruction ramps down the axis to a standstill and the acknowledgeable error message "Err.DistanceErr" is output (refer to AUTOHOTSPOT). Then the axis can be moved again after the error has been acknowledged.
- A non-synchronized linear axis is not monitored and can therefore travel right to the physical end of the axis without slowing down. You must therefore complete the motion in time.

> **WARNING**
>
> Non-synchronized linear axis
>
> Injury to persons and property may occur.
>
> instruction MC_MoveJog can be used to traverse a non-synchronized linear axis to its mechanical end.
>
> To avoid injury to persons and damage to objects, take the following precautions:
>
> - Install an EMERGENCY STOP switch in the vicinity of the computer. This is the only way to ensure that the system is reliably switched off in the event of a computer or software failure.
> - Install safety limit switches for direct control of the power units of all drives.
> - Ensure that nobody has access to the area of the system where there are moving parts.

- A synchronized linear axis is decelerated when approaching the end of the working range and stops at the software limit switch. The acknowledgeable error message "Err.DistanceErr", as well as the "Err.SWLimitMinExceeded" or "Err.SWLimitMaxExceeded" error messages will be output. After the error has been acknowledged, it is only possible to travel in the opposite direction.

Special features of rotary axes:

- instruction MC_MoveJog does not limit the traversing distance of a rotary axis (infinite rotation).

Signal flow diagram for MC_MoveJog

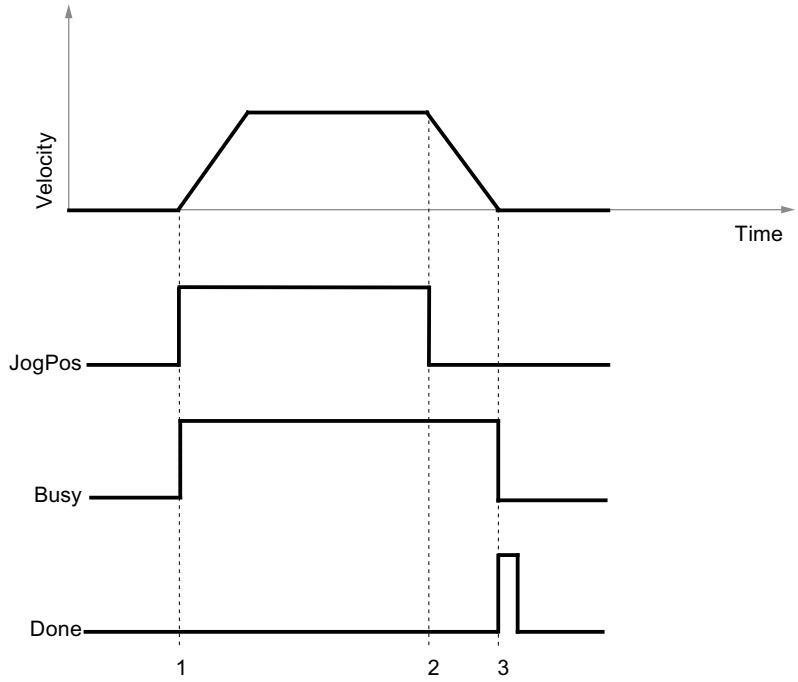

1 Motion is started by setting "JogPos".
2 Motion is stopped by resetting "JogPos".
3 When the axis reaches a standstill, "Busy" is reset, and "Done" is set for one cycle.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
JogPos	INPUT	BOOL	1 = Movement in positive direction as long as "JogPos" is set.
JogNeg	INPUT	BOOL	1 = Movement in negative direction as long as "JogNeg" is set.
Velocity	INPUT	REAL	Axis velocity in [length unit/s]
Acceleration	INPUT	REAL	Axis acceleration in [length unit/s^2]
Deceleration	INPUT	REAL	Axis deceleration in [length unit/s^2]
Busy	OUTPUT	BOOL	1 = Job in progress.
Done	OUTPUT	BOOL	1 = Job completed without error. After the axis has reached a standstill, the bit is set for exactly for the duration of one call of the instruction.

Parameter	Declaration	Data type	Description
CommandAborted	OUTPUT	BOOL	1 = Job was canceled by another job. The bit remains active until the reset of "JogPos" or "JogNeg".
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active until the reset of "JogPos" or "JogNeg".

UDT2

A UDT2 data type is listed and reserved in the block properties of MC_MoveJog. This means that no separate UDT2 data type may be used in the same project. This would result in error messages and inconsistencies within the project. If you want to use a UDT2 data type, this UDT must be renamed, for example, to UDT3.

MC_Home (S7-300, S7-400)

Description

instruction MC_Home is used for the following tasks:

- Reference point approach with incremental encoder ("Velocity" > 0)
- Set reference point with incremental and absolute encoder ("Velocity" = 0)
  The set reference point function for absolute encoders is also known as absolute encoder adjustment.

Operating principle

Reference point approach ("Velocity" > 0)

Once "Execute" has been set, the axis travels at the selected velocity and in the selected direction until the position decoder module distance and the input driver detect that the axis has reached the reference point. Once the reference point has been reached, the actual position value is set to the value specified in the"Position" parameter, and the axis is slowed down. The axis is now synchronized ("Sync" = TRUE), but is not positioned on the reference point.

If the maximum travel distance is exceeded in the reference point approach, instruction MC_Home slows down the axis to a standstill (refer to Sequence of travel movements) and the "Error" and "Err.DistanceErr" displays is set (refer to Errors with soft stop).

The axis will not be synchronized ("Sync" = FALSE) if the reference point approach is stopped before the reference point has been reached.

instruction MC_Home does not stop at the end of the axis, or automatically reverse the direction of movement.

DANGER

If no reference point is found, you must take suitable measures to ensure that the axis will come to a standstill before it reaches the mechanical end position (for example, EMERGENCY STOP switch, safety limit switch).

Set reference point ("Velocity" = 0)

Set "Velocity" to 0. Once "Execute" has been set, the actual position value is set to the value specified in "Position" and the status bit in the axis DB is set simultaneously to "Sync" = TRUE.

Following a replacement of the absolute encoder or mechanical alterations, you must repeat the set reference point task, even if the axis is still synchronized.

Note

Rule for absolute encoders:

The encoder value at the reference point is saved in the axis DB. For this reason, after setting the reference point, you should add the online axis DB to your offline database. Otherwise, the axis is no longer synchronized after you download the DB to the CPU.

Read the information on Loading the axis DB.

Signal flow diagram for reference point approach

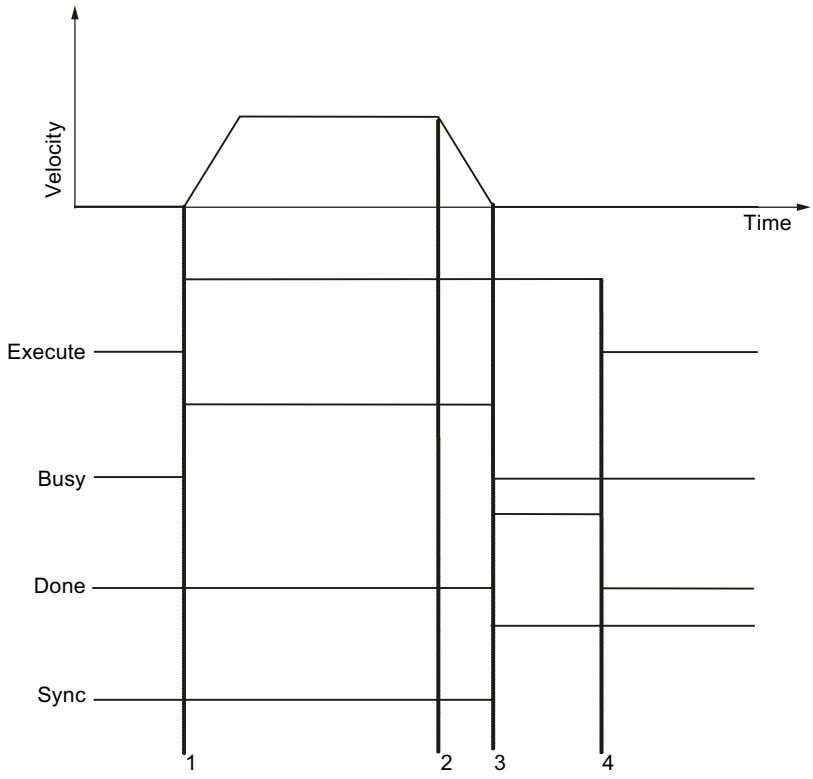

1 Start of the reference point approach with "Execute".
2 Reference position found; axis is decelerating.
3 Axis has reached a standstill; "Busy" is cleared and "Done" is reset.
The axis is synchronized.
4 Reset of "Execute" resets "Done".

Signal flow diagram for reference point setting

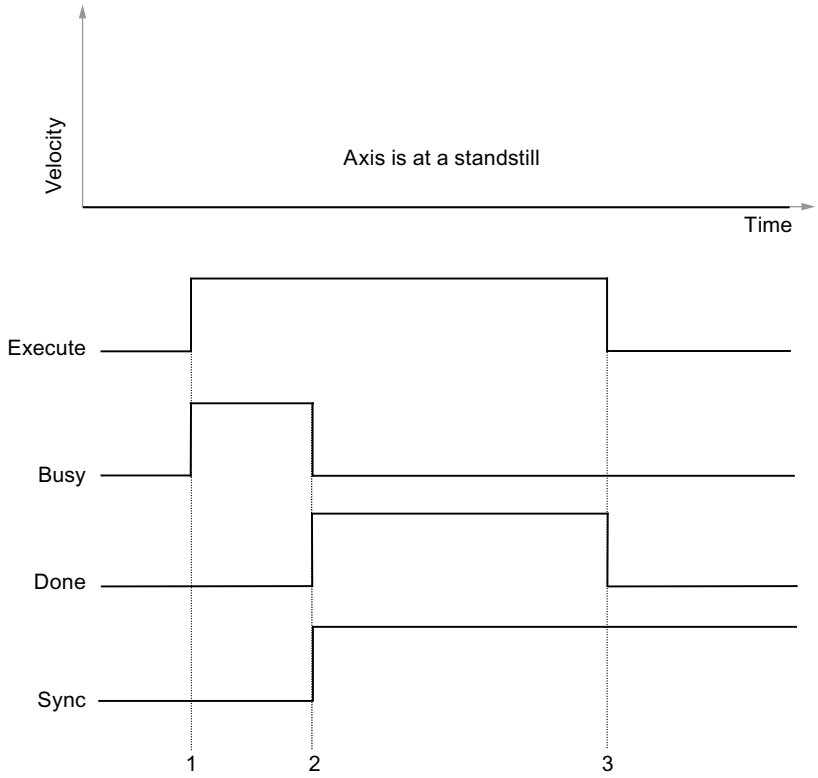

1 Start of the reference point setting or absolute encoder adjustment with "Execute".
2 The axis reports the synchronized state.
3 Reset of "Execute" resets "Done". The process is terminated.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

The reference point approach or reference setting can only be started when the axis is at a standstill.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Start of reference point approach or reference setting with a positive edge.
Position	INPUT	REAL	**Incremental encoder** Reference point coordinate for reference point approach or reference setting. **Absolute encoder** Reference point coordinate for reference setting. The value must be within the working range: • **Linear axis** Within the software limit switch • **Rotary axis:** Rotary axis start ≤ position < Rotary axis end
Velocity	INPUT	REAL	Axis velocity in [length unit/s] Velocity > 0: Reference point approach Velocity = 0: Set reference point
Acceleration	INPUT	REAL	Axis acceleration in [length unit/s^2]
Deceleration	INPUT	REAL	Axis deceleration in [length unit/s^2]
Direction	INPUT	INT	Start direction for reference point approach 1 = Reference point approach in positive direction -1 = Reference point approach in negative direction
Busy	OUTPUT	BOOL	1 = Job in progress.
Done	OUTPUT	BOOL	1 = Job completed without error. The bit remains active at least for one call of the instruction until "Execute" is reset.
CommandAborted	OUTPUT	BOOL	1 = Job was canceled by another job. The bit remains active at least for one call of the instruction until "Execute" is reset.
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active at least for one call of the instruction until "Execute" is reset.

MC_StopMotion (S7-300, S7-400)

Description

instruction MC_StopMotion allows you to interrupt any travel instruction so that the axis comes to a standstill with the set deceleration.

Typical application: A positioning job can by interrupted at any time due to an external event (for example, opening of a safety door).

Operating principle

On detecting the positive edge at input "Execute", the MC_StopMotion instruction takes control of the axis and slows it down to a standstill. The MC_StopMotion instruction can only be interrupted by hard faults and not by a travel instruction. After the axis has reached a standstill, a new travel movement can be started.

You can set a deceleration value at the MC_StopMotion instruction to suit your application. If this deceleration value is less than the one set in the interrupted instruction, the axis is slowed down with this deceleration value. This ensures that the axis does not overshoot the target set in the interrupted block.

If you interrupt a movement with target (MC_MoveAbsolute, MC_MoveRelative) by calling instruction MC_StopMotion, the instruction continues to calculate the remaining distance. On termination of instruction MC_StopMotion, the remaining distance indicates the distance of the current position setpoint to the target of the interrupted block. You can assign this value to an MC_MoveRelative instruction in order to approach the axis to the original target after interruption.

Signal flow diagram for MC_StopMotion

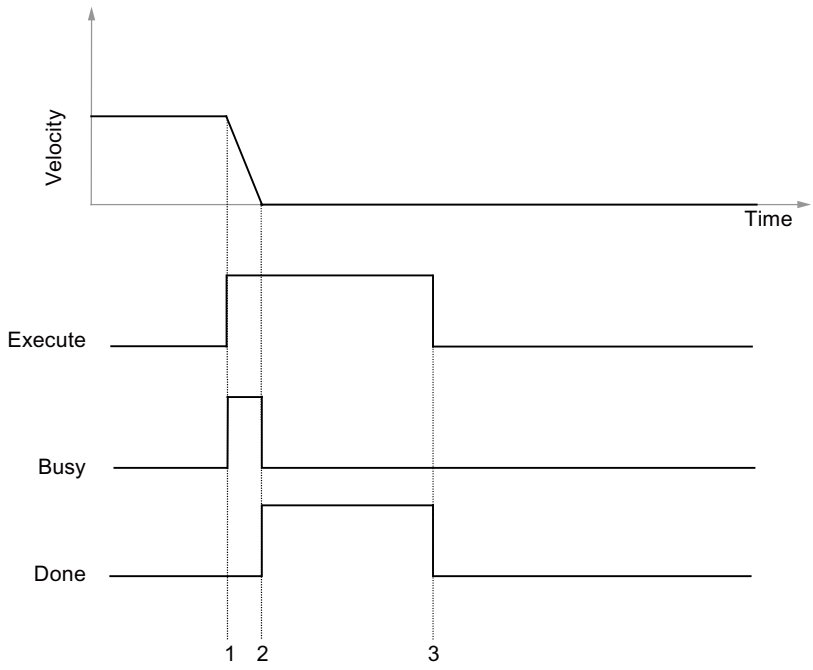

1 FB Stop is started while a movement is active.

2 MC_StopMotion sets output "Done" as soon as standstill is reached.

3 Reset of "Execute" terminates FB Stop.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Start of braking with a positive edge
Deceleration	INPUT	REAL	Axis deceleration for deceleration ramp in [length unit/s^2] If you enter values <= 0.0, the maximum deceleration value from the axis DB will be used.
Busy	OUTPUT	BOOL	1 = Job in progress.
Done	OUTPUT	BOOL	1 = Job completed without error. The bit remains active at least for one call of the instruction until "Execute" is reset.
CommandAborted	OUTPUT	BOOL	1 = Job was cancelled due to manual intervention at the controller.
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active at least for one call of the instruction until "Execute" is reset.

MC_Control (S7-300, S7-400)

Description

instruction MC_Control allows you to implement the position control in combination with the I/O drivers.

instruction MC_Control may assume the following controller states:

- Control mode
- Follow-up mode
- Manual mode

Control mode

The MC_Control instruction

- Calculates the velocity setpoint for the output driver as a function of the position setpoint of a travel instruction and actual position value of the input driver.
- Monitors
  - Standstill
  - Target approach
  - Following error
- Processes error acknowledgment (see Error displays and error acknowledgment)

Follow-up mode

In this state, the setpoint position is corrected according to the actual position. The follow-up mode is activated if "EnableDrive" = FALSE, or after an axis fault with hard stop (refer to Error displays and error acknowledgment). Output "DriveEnabled" = FALSE in follow-up mode.

Manual mode

In this mode, the manual setpoint "ManVelocity" is output as the velocity setpoint. This setpoint is limited to "MaxVelocity". The position setpoint is tracked to the actual position value.

This state is switched on or off using "ManEnable".

⚠ WARNING
Manual control is possible at any time, even if the axis is in error status.

Operating principle

The proportional element of the control algorithm of instruction MC_Control is configured at the "controller gain" parameter ("FactorP").

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Call instruction MC_Control once per axis unconditionally in the selected OB.

Parameter

Parameter	Declaration	Data type	Meaning
EnableDrive	INPUT	BOOL	0 = Disable drive, tracking control is active. 1 = Enable drive, position control is active. If no error is pending, "DriveEnabled" is set.
DriveEnabled	OUTPUT	BOOL	Drive enable, position control is active. If "EnableDrive" is set and no error is pending, "DriveEnabled" is set. If "DriveEnabled" is not set, the axis is in follow-up mode and standstill monitoring is not active for the axis. This signal should be used in order to control the enable input on the power unit.

MC_Simulation (S7-300, S7-400)

Description

instruction MC_Simulation allows you to test your traversing program and driver blocks. You do not require any I/O and your axis is not moved.

Operating principle

instruction MC_Simulation feeds back the controller output value to the input driver by simulating an encoder value corresponding to the encoder type selected in the axis DB.

To activate simulation mode, set the bit "Sim" = TRUE in the axis DB.

When simulation is enabled, a reference point approach is not possible. When in simulation mode, configure the "Velocity" input of instruction MC_Home with the value 0.0. This means that a reference setting will be performed instead of a reference point approach.

If you set a reference point in simulation mode, the reference point of your real axis is also altered. It is therefore necessary to resynchronize your axis after exiting simulation mode.

Whenever you activate and deactivate simulation mode, the axis is set to the stop status that requires acknowledgment (refer to Faults with hard stop) and all bits of the "Init.Ix" bit array is set in the axis DB.

If simulation mode is enabled, the following occurs:

- The output driver does not output values to the I/O.
- The input driver does not read in any encoder values from the I/O.

DANGER

The axis must be at a standstill when simulation mode is enabled.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

MC_GearIn (S7-300, S7-400)

Description

instruction MC_GearIn allows you to link the velocity setpoint of a following axis to the velocity setpoint of a leading axis by means of gear ratio (setpoint coupling).

Operating principle

As soon as a positive edge is detected at the "Execute" input, instruction MC_GearIn couples a following axis (parameter "Slave") to the velocity or position setpoint of a leading axis (parameter "Master") by means of the set gear ratio. The instruction therefore represents a travel instruction of the following axis.

The values in each cycle of a coupled motion are as follows:

- The distance traversed by the following axis, starting at the coupling position =
  the distance traversed by the leading axis, starting at the coupling position x gear ratio
- Velocity of the following axis = gear velocity =
  velocity of the leading axis x gear ratio
- Acceleration of the following axis =

  acceleration of the leading axis x gear ratio
- Deceleration of the following axis =
  deceleration of the leading axis x gear ratio

To allow the following axis to follow the leading axis without restriction, set the following values (with a reserve factor of 1.1) for the following axis. These values must be valid and attainable:

- Velocity(following axis) =
  Velocity(leading axis) × ABS(gear ratio) × 1.1
- Acceleration(following axis) =
  Acceleration(leading axis) × ABS(gear ratio) × 1.1
- Deceleration(following axis) =
  Deceleration(leading axis) × ABS(gear ratio) × 1.1

To cancel the coupling, call the stop instruction, or any other travel instruction for the following axis.

As with all other travel instructions, the MC_GearIn instruction can override a movement and be overridden itself.

The following axis can be coupled to a moving leading axis, whereby the following axis automatically accelerates or decelerates using the ramps configured at the gear instruction until it has reached the gear velocity.

The "InGear" output is set when the gear velocity has been reached for the first time after the start of the motion. If "Execute" is not set at this point, "InGear" remains active for one call of the instruction. "InGear" is reset at the high to low transition at Execute, when an error has occurred, and when a movement is overridden.

Provided the gear velocity can be reached, the following axis travels in speed and position synchronism. The "Coupled" output of the instruction is set.

If the following axis cannot reach the gear velocity, it moves according to the motion parameters set at the gear instruction. The "Coupled" output of the instruction is then reset. As soon as the gear velocity can be reached again, the following axis moves at this velocity. The "Coupled" output of the instruction is set.

As long as instruction MC_GearIn is active at the following axis, you can call any travel instruction for the leading axis, and also override a movement.
Exception: During referencing of the leading axis, the coupling is released with an error ("Err.MasterErr").

The "Override" axis parameter is useless for the following axis and is therefore not evaluated.

Special features when operating a following axis as linear axis:

- If the maximum traversing distance is exceeded during a gear movement, instruction MC_GearIn slows down the following axis to a standstill with the error message (see Sequence of travel movements). The coupling is released.
- A **non-synchronized** linear axis is not monitored and can therefore travel right to the physical end of the axis without slowing down. You must therefore complete the motion in time.
- A **synchronized** linear axis is decelerated when approaching the end of the working range and stops at the software limit switch. The error message "DistanceErr" is output along with "Err.SWLimitMinExceeded", or "Err.SWLimitMaxExceeded". The coupling is released.

WARNING

Injury to persons and property may occur.

instruction MC_GearIn can be used to traverse a non-synchronized linear axis to its mechanical end.

To avoid injury to persons and objects, take the following precautions:

- Install an EMERGENCY STOP switch in the vicinity of the computer. This is the only way to ensure that the system is reliably switched off in the event of a computer or software failure.
- Install safety limit switches for direct control of the power units of all drives.
- Ensure that nobody has access to the area of the system where there are moving parts.

Special feature when operating a following axis as rotary axis:

- The traversing distance of instruction MC_GearIn is not limited (infinite rotation).

Error handling

Master axis errors:

- The leading axis decelerates when it detects errors that result in a soft stop. The coupled following axis follows this motion and does not release the coupling. If the leading axis error is acknowledged, and the leading axis continues its motion, the following axis also resumes the motion.
- If the leading axis detects errors that result in a hard stop, or if it is operated in manual mode, the gear instruction decelerates the following axis and resolves the coupling. The instruction outputs the "Err.MasterErr" error message.

Slave axis errors:

- If the following axis detects errors that result in a soft stop, the gear instruction decelerates the following axis and resolves the coupling.
- If the following axis detects errors that result in a hard stop, the following axis is brought to a stop, and the coupling is released.
- The maximum traversing distance of the coupling movement of a linear axis is limited to 2^{24} increments or 2^{24} [length unit]. If this limit is exceeded, instruction MC_GearIn decelerates the following axis, outputs a "DistanceErr" error message that you can acknowledge, and resolves the coupling.

Coupling errors:

- The following axis travels according to the motion parameters set at the instruction if it cannot reach the gear velocity in the current cycle. This applies particularly when the axis is coupled to a moving leading axis and during a braking operation triggered by an error. The "Coupled" parameter is reset, but the coupling is not released. As soon as the following axis is capable of reaching the necessary values again, it travels according to these values and "Coupled" is set again.

Interaction of instructions for the gear

When using a gear instruction, you should also note the following aspects (refer to "Structure of a user program"):

- The leading and following axes must be called in the **same** execution level.
- In each processing cycle, the following axis adapts itself again to the default values of the leading axis. The following axis instructions should therefore be called after instructions of the leading axis. In particular, the gear instruction of the leading axis should be called **after** the controller of the leading axis.
  Other call sequences, in particular nested calls of the instructions of the leading and following axes, may lead to greater following errors.
- In isochronous applications, the I/O data of the leading axis **and** the following axis must be located in the same process image partition.

This results in one of the following figures, depending on the selected structure:

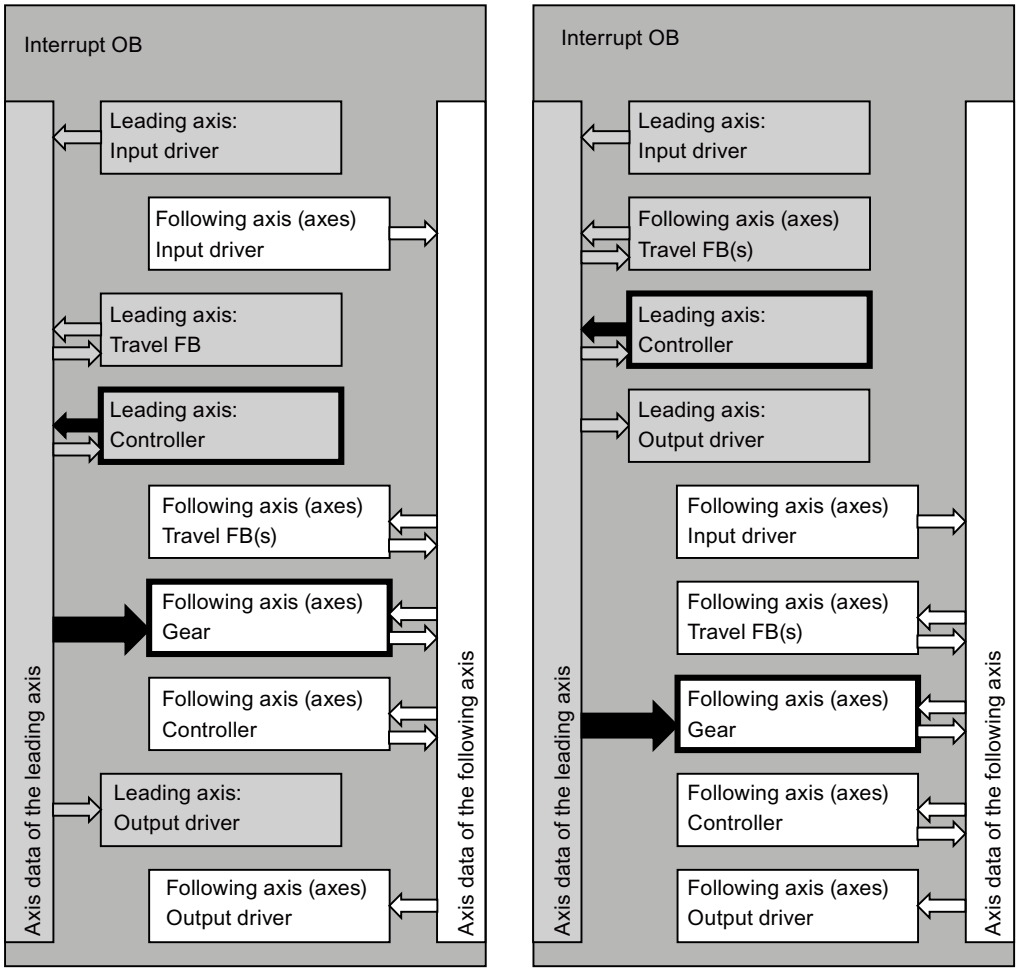

Signal Flow Chart

The chart below shows how the gear instruction takes control of the axis movement and couples to a moving leading axis. In this example, the gear ratio is 0.8. The following axis cannot reach the velocity required to maintain this ratio.

The leading axis is stopped between t = 0.5 and t = 0.72. The coupling is maintained.

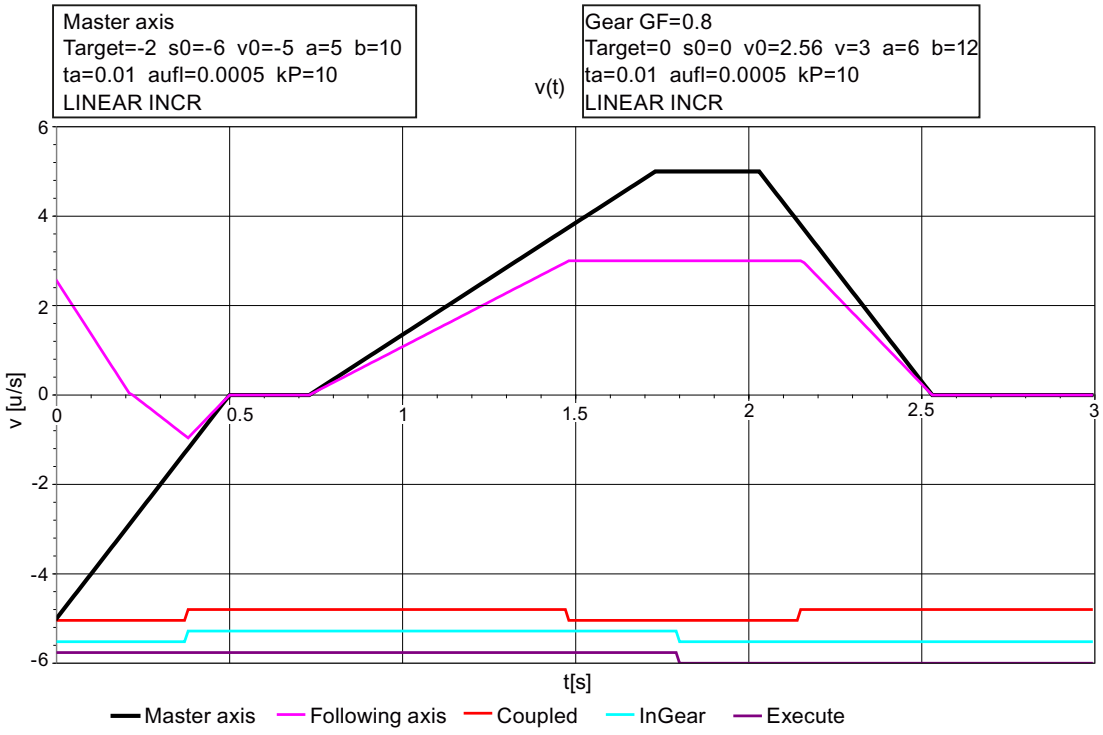

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Start of the gear motion on a positive edge
Rationumerator	INPUT	REAL	Gear ratio numerator Gear ratio = "RatioNumerator" / "RatioDenominator" If "RatioNumerator" < 0, the direction of movement of the leading axis is opposite to that of the following axis. "RatioNumerator" = 0 is rejected with "DataErr" = 1. You can modify the effective gear ratio of the following axis by overriding the active gear instruction with another instance that contains a modified gear ratio.

Parameter	Declaration	Data type	Description
RatioDenominator	INPUT	INT	The gear ratio denominator; must be an integer ≥ 1 You can modify the effective gear ratio of the following axis by overriding the active gear instruction with another instance that contains a modified gear ratio.
Velocity	INPUT	REAL	Axis velocity in [length unit/s] The maximum geared velocity of the following axis is limited to this value.
Acceleration	INPUT	REAL	Axis acceleration in [length unit/s^2] The maximum geared acceleration of the following axis is limited to this value. This parameter is used if: • The following axis is coupled to a faster leading axis. • The acceleration of the leading axis exceeds the capability of the following axis.
Deceleration	INPUT	REAL	Axis deceleration in [length unit/s^2] The maximum geared deceleration of the following axis is limited to this value. This parameter is used if: • A following axis that is already in motion is coupled to a slower leading axis. • The following axis is brought to a stop due to an error. • The deceleration of the leading axis exceeds the capability of the following axis.
Master	IN-OUT	AXIS_REF	Reference to the leading axis data
Slave	IN-OUT	AXIS_REF	Reference to the following axis data
Init	IN-OUT	BOOL	If Init = TRUE, the instruction completes its initialization and then resets the bit.
Busy	OUTPUT	BOOL	1 = Job in progress.
InGear	OUTPUT	BOOL	1 = Gear speed reached Set the first time the gear velocity has been reached. If "Execute" is already reset at this point, "InGear" still remains active for one call of the instruction. It is reset at the high to low transition at Execute, when an error has occurred, and when the movement is overridden.
Coupled	OUTPUT	BOOL	1 = Axis is currently coupled to the leading axis. Reports in each processing cycle whether the gear speed was reached.
CommandAborted	OUTPUT	BOOL	1 = Job was canceled by another job.
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active at least for one call of the instruction until "Execute" is reset.

Input driver (S7-300, S7-400)

Task of the input driver (S7-300, S7-400)

Function

This input driver is used to detect the actual positions of an axis. It reads the non-standardized axis position and converts this to the selected unit of length.

Suitable input drivers are available for a number of position decoder modules. These drivers are described individually in the following sections. Adaptation to configurations that are not directly supported is possible by means of instruction EncoderUniversal (Page 5854).

Initialization

- Synchronization of an **incremental encoder** with the axis is reset (parameter "Sync" = FALSE in the axis data). A new reference point approach or reference setting is then necessary.
- Synchronization of an **absolute encoder** with the axis is retained.

Call

The input driver of an axis must be called unconditionally.

Isochronous mode

The input drivers support the isochronous mode of all suitable I/O modules. The driver automatically recognizes isochronous mode by the fact that it is called in an isochronous OB.

Error handling

Errors can be detected by means of the following mechanisms:

- An error detected by the user program (for example, module removed, or station failure) can be signaled to the input driver at input "EncErr". The input driver will not access the position decoder module as long as input "EncErr" is TRUE.
- Some position decoder modules provide error information that is evaluated automatically by the input driver.
- Other position decoder modules only return their error information by means of diagnostic interrupt and data set 1 (DS1). You must therefore perform the following steps:
  - Enable the diagnostics interrupt for the module in the hardware configuration.
  - In OB 82, compare the address returned in the local data element "OB82_MDL_ADDR" with your module address. If these agree, set the "ReadDiagErr" parameter of the input driver to TRUE (for the incoming and outgoing alarm).
    If "ReadDiagErr" = TRUE, the input driver reads DS 1 of the module by calling instruction RD_REC, saves this record to the static "DiagStatus" parameter
    and then sets "ReadDiagErr" = FALSE. The return value of instruction RD_REC is written to the static "JobErr" parameter.

Error Response

- In the event of an error, the"Error" and "Err.EncoderErr" displays is set in the axis DB.
- With **incremental encoder**, status bit "Sync" is always reset in the axis DB.
- With **absolute encoder**, status bit "Sync" is never reset in the axis DB.

Please also note specific error handling in the following sections on individual drivers.

For detailed information on error response, error display, and error acknowledgment, see "Error Displays and Error Acknowledgment in the axis DB".

Simulation mode

The input driver will not access the position decoder module as long as input "Sim" is set in the axis DB. The encoder values are simulated by means of instruction MC_Simulation (Page 5824).

EncoderIM178 (S7-300, S7-400)

Description

instruction EncoderIM178 is included for reasons of compatibility with STEP7 V5.x. Siemens is no longer actively marketing the IM 178-4 . You can use IM 174 as an alternative.

IM 178-4 must be used with the same channel number and axis both at the input **and** output drivers, as specific address settings refer to both drivers. A combination with other drivers is not possible.

Configuration of IM 178-4 (hardware configuration)

Select the IM 178-4 from the "Distributed I/O devices" section in the hardware catalog. Configure slot 4 of IM 178-4 as "4Word QQ/12Word QI/cons. 1Word".

Parameter	Setting
Synchronization	Software
Edge selection	Rising, falling, or both edges

Configure the other settings depending on the connected encoder.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
RefMode	INPUT	BOOL	Referencing type: 0 = With digital input I0 1 = With digital input I1 and zero mark
DI_0	OUTPUT	BOOL	Digital input 0 of the corresponding channel of IM 178-4
DI_1	OUTPUT	BOOL	Digital input 1 of the corresponding channel of IM 178-4
DI_2	OUTPUT	BOOL	Digital input 2 of the corresponding channel of IM 178-4
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
Status.PARA	STAT	BOOL	PARA = TRUE: IM 178-4 was correctly configured by means of the hardware configuration. PARA = FALSE: see SIMATIC IM 178-4 Drive Interface Manual (http://support.automation.siemens.com/WW/view/en/1117389)
Status.EXTF0	STAT	BOOL	1 = Encoder signal line fault
Status.EXTF1	STAT	BOOL	1 = Encoder error
Status.EXTF2	STAT	BOOL	1 = Zero mark fault

Synchronization

Synchronization can be performed by:

- Reference point approach (with incremental encoder)
- Reference point setting (with incremental and absolute encoder)

Depending on the setting, connect the reference point switch to digital input I0 or I1.

During a reference point approach, IM 178-4 can detect the reference point in either of the following cases:

- At a negative edge, positive edge, or at both edges at digital input I0 (RefMode = 0).
- At the positive edge of the first zero mark after detecting the negative edge of digital input I1 (RefMode = 1).

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Digital Inputs

The digital inputs I0 to I2 of the selected channel of IM 178-4 are indicated at the output parameters "DI_0" to "DI_2" of the input driver.

Error handling

In response to a detected error, the input driver sets the "Error" and "Err.EncoderErr" error codes in the axis DB. The input driver resets the "Sync" status bit in the axis DB if you are using an incremental encoder. For more information, refer to the SIMATIC IM 178-4 Drive Interface Manual (http://support.automation.siemens.com/WW/view/en/1117389).

For detailed information on error response, error display, and error acknowledgment, see "Error Displays and Error Acknowledgment in the axis DB".

Error acknowledgment

If you set error acknowledgment in the axis DB of Easy Motion Control, errors pending in the module are also acknowledged.

EncoderFM450 (S7-300, S7-400)

Description

instruction EncoderFM450 is used to employ a channel of the FM 450-1 as position decoder module for incremental encoders. The channel number is configured at the axis DB.

Configuration of the module (hardware configuration)

The FM 450-1 module should be configured in accordance with the settings listed in the following table.

Dialog	Setting
Interrupt selection	Enable diagnostic interrupt
Operating modes	• Count range: -31 to +31 bits • Operating mode: Continuous counting • Gate control: None
Inputs	If you want to perform a reference point approach, you must select the following setting: • Synchronization: once • The setting "Evaluate zero mark for setting" may be selected or cleared as a reference point criterion.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
ReadDiagErr	IN-OUT	BOOL	Set this bit in OB 82 if the module has triggered a diagnostic interrupt (see below under Error handling).
DI_0	OUTPUT	BOOL	Digital input 0 of the selected channel
DI_1	OUTPUT	BOOL	Digital input 1 of the selected channel
DI_2	OUTPUT	BOOL	Digital input 2 of the selected channel
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
JobErr	STAT	INT	Return value of instruction RD_REC, if "ReadDiagErr" = TRUE was set.
DiagStatus.EXT_VOLTAGE	STAT	STRUCT	External auxiliary voltage faulty
DiagStatus.CONFIG_ERR	STAT	STRUCT	Incorrect configuration
DiagStatus.WTCH_DOG_FLT	STAT	STRUCT	Watchdog timeout
DiagStatus.RAM_FLT	STAT	STRUCT	RAM failure
DiagStatus.CH1_SIGA	STAT	STRUCT	Channel 1 signal A defective
DiagStatus.CH1_SIGB	STAT	STRUCT	Channel 1 signal B defective
DiagStatus.CH1_SIGZ	STAT	STRUCT	Channel 1 signal N defective
DiagStatus.CH1_BETW	STAT	STRUCT	Channel 1 fault between channels
DiagStatus.CH1_5V2	STAT	STRUCT	Channel 1 encoder supply 5.2 V fault

Parameter	Declaration	Data type	Description
DiagStatus.CH2_SIGA	STAT	STRUCT	Channel 2 signal A defective
DiagStatus.CH2_SIGB	STAT	STRUCT	Channel 2 signal B defective
DiagStatus.CH2_SIGZ	STAT	STRUCT	Channel 2 signal N defective
DiagStatus.CH2_BETW	STAT	STRUCT	Channel 2 fault between channels
DiagStatus.CH2_5V2	STAT	STRUCT	Channel 2 encoder supply 5.2 V fault

Synchronization

Synchronization can be performed by:

- Reference point approach or
- Set reference point

Connect the reference point switch to digital input I2.

During a reference point approach, the module can detect the reference point in either of the following cases:

- On a rising edge at digital input I2
- On the rising edge of the zero mark, if digital input I2 is set

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Digital inputs

Digital inputs I0 to I2 of the selected channel are indicated at the output parameters "DI_0" to "DI_2" of the input driver.

Error handling

The module returns its error information by means of dataset 1 (DS1). In the following situations, the input driver reads DS1 from the module by calling instruction RD_REC and saves it to the static "DiagStatus" parameter:

- The input driver is initialized.
- The module reports an error.
- The driver input parameter "ReadDiagErr" is TRUE.

It then sets "ReadDiagErr" = FALSE. The return value of instruction RD_REC is written to the static "JobErr" parameter (see JOB_Err alarms).

If an error display is registered in dataset 1, the input driver sets the error codes "Error" and "Err.EncoderErr" to TRUE in the axis DB. Status bit "Sync" is set to FALSE.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB.

Note

If you are using an FM 450-1 in single-channel mode only, you must configure the second channel with "24 V channel A+B phase offset", as it will otherwise transmit cyclic diagnostic interrupts (for example, due to missing circuitry).

EncoderET200S1SSI (S7-300, S7-400)

Description

instruction EncoderET200S1SSI is used to employ the ET 200S 1SSI Fast Mode as position decoder module for absolute encoders.

The driver can also be used for the isochronous variant of this module.

Configuration the ET 200S 1SSI Fast Mode module (hardware configuration)

The input driver is only suitable for the 1SSI module with "**Fast Mode**" parameter setting as operating mode. In the TIA Portal, the "**Fast Mode**" is no longer selected based on the module, but is now set as operating mode.

Select the module from the "Distributed I/O devices" > ET 200S > FM > Position detection" section in the hardware catalog.

Parameter	Setting
Group diagnostics	You can activate this function if you want to evaluate the data in your program.
Detection	Freewheeling
Reversal of the direction of rotation	Deactivated You can reverse the direction of rotation using the instructions of Easy Motion Control (parameter "Set encoder polarity").
Operating mode	Fast mode

Configure the other settings depending on the connected encoder.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
Status.STS_UP	STAT	BOOL	Forward direction: encoder values increase
Status.STS_DN	STAT	BOOL	Reverse direction: encoder values decrease
Status.STS_DI	STAT	BOOL	Status of digital input
Status.EXTF	STAT	BOOL	Group error or short circuit in encoder supply
Status.ERR_PARA	STAT	BOOL	Parameter errors
Status.RDY	STAT	BOOL	Module is ready for operation

Synchronization

The ET 200S 1SSI Fast Mode module only supports synchronization by means of reference point setting.

Error handling

If the input driver detects an error report of the module, it sets the error codes "Error" and "Err.EncoderErr" in the axis DB.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB.

EncoderAbsSensorDP (S7-300, S7-400)

Description

instruction EncoderAbsSensorDP is used to employ the absolute encoder SIMODRIVE sensor in combination with PROFIBUS DP for position detection.

Configuration of the absolute encoder SIMODRIVE sensor (hardware configuration)

Select the "PROFIBUS DP > ENCODER > SIMODRIVE sensor > Version 2.2 Single-turn or Version 2.2 Multiturn" encoder from the hardware catalog. If the SIMODRIVE sensor is not yet available, implement the device using the corresponding GSD file. The encoder will then be available in the "Additional field devices > PROFIBUS DP > Encoder > Siemens AG > SIMODRIVE sensor" folder.

Parameter	Setting	
Code sequence	Increasing clockwise You can reverse the direction of rotation using the instructions of Easy Motion Control (parameter "Set encoder polarity").	
	Multiturn	**Single-turn**
Scaling function	Enable	Enable
Measuring steps (high)	0	-
Measuring steps (low)	4096	4096
Physical impulses (high)	0	-
Physical impulses (low)	4096	4096
Desired measuring units per	Revolution	Revolution
Total measuring range (high)	256	-
Total measuring range (low)	0	4096
Commissioning mode	Switched off	Switched off
Shorter diagnostics	No	No
Lower limit switch	Switched off	Switched off
Lower limit switch (high)	0	-
Lower limit switch (low)	0	0
Upper limit switch	Switched off	Switched off
Upper limit switch (high)	0	-
Upper limit switch (low)	32767	4096
Velocity output unit	Steps/1000 ms	Steps/1000 ms

Adapt the setting to your selected length unit using the instructions of Easy Motion Control. Fixed parameters can therefore be assigned to the encoder - as indicated above. It is not necessary to perform a "teach-in" of the encoder (refer to the SIMODRIVE sensor absolute encoder with PROFIBUS DP Operating Manual (http://support.automation.siemens.com/WW/view/en/28813901)).

At the parameter "steps per encoder revolution" of Easy Motion Control, enter 4096 ("StepsPerRev").

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
Status.Ready	STAT	BOOL	1 = Encoder is ready for operation
Status.Mode	STAT	BOOL	Operating mode: 0 = Startup mode, 1 = Normal mode
Status.SWLimitExceeded	STAT	BOOL	1 = Software limit switch MIN or MAX has responded
Status.Dir	STAT	BOOL	Direction of rotation: 0 = Clockwise, 1 = Counter-clockwise

Synchronization

Synchronization with the absolute encoder SIMODRIVE sensor is only possible by means of reference point setting.

Error handling

If the input driver detects that an encoder error alarm is pending, it sets the"Error" and "Err.EncoderErr" error codes in the axis DB. For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB.

EncoderSM338 (S7-300, S7-400)

Description

instruction EncoderSM338 is used to employ SM 338 POS_INPUT as position decoder module for absolute encoders.

The driver can also be used for the isochronous variant of this module.

Configuration of SM 338 (hardware configuration)

The FREEZE function must be deactivated.

If the input driver detects a FREEZE status at the encoder input, it sets the "Error" and "Err.EncoderErr" error codes in the axis DB.

Configure the other settings depending on the connected encoder.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
ReadDiagErr	IN-OUT	BOOL	Set this bit in OB 82 if the module has triggered a diagnostic interrupt (see Error handling).
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
Freeze	STAT	BOOL	FREEZE status of the encoder input
JobErr	STAT	INT	Return value of instruction RD_REC, if "ReadDiagErr" = TRUE was set.
DiagStatus.EXT_VOLTAGE	STAT	STRUCT	External auxiliary voltage faulty
DiagStatus.CONFIG_ERR	STAT	STRUCT	Incorrect configuration
DiagStatus.WTCH_DOG_FLT	STAT	STRUCT	Watchdog timeout
DiagStatus.RAM_FLT	STAT	STRUCT	RAM failure
DiagStatus.CH0_INTF	STAT	STRUCT	Configuration or parameter assignment error (internal channel error)
DiagStatus.CH0_EXTF	STAT	STRUCT	Encoder error (external channel error)

Synchronization

Synchronization with SM 338 is only possible by means of reference point setting.

Error handling

This module only provides its error information via diagnostic interrupt and data record 1 (DS1). You must therefore perform the following steps:

- Enable the diagnostics interrupt for the module in the hardware configuration.
- In OB 82, compare the address returned in the local data element "OB82_MDL_ADDR" with your module address. If these agree, set the ReadDiagErr parameter of the input driver to TRUE (for the incoming and outgoing interrupt).
  If "ReadDiagErr" = TRUE, the input driver reads dataset 1 of the module by means of instruction RD_REC and saves it to the static "DiagStatus" parameter.
  It then sets "ReadDiagErr" = FALSE. The return value of instruction RD_REC is written to the static "JobErr" parameter (see JOB_Err alarms).

If an error display is registered in dataset 1, the input driver sets the error codes "Error" and "Err.EncoderErr" to TRUE in the axis DB.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB.

EncoderET200S1Count (S7-300, S7-400)

Description

instruction EncoderET200S1Count is used to employ the ET 200S 1Count5V or 1Count24V module as position decoder module for incremental encoders.

The driver can also be used for the isochronous variant of this module.

Configuration of the ET 200S 1Count module (hardware configuration)

The input driver is only suitable for the "1 Count 5V" or "1 Count 24V" from the hardware catalog. The count mode is now set as the operating mode instead of be selecting via the module.

Select the module from the "Distributed I/O devices" > ET 200S > FM > Counting" section in the hardware catalog.

Parameter	Setting
Group diagnostics	You can activate this function if you want to evaluate the data in your program.
Type of counting mode	Continuous counting
DI function, Synchronization	• If you wish to perform a reference point approach, select: – DI function = Synchronization at positive edge – Synchronization = One-time • Otherwise select: – DI function = input
Gate function	Arbitrary
Operating mode	Count

Configure the other settings depending on the connected encoder.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
DOut_1	INPUT	BOOL	Digital output 1
DOut_2	INPUT	BOOL	Digital output 2 (only available with 1Count5V)
DIn	OUTPUT	BOOL	Digital input of module

Parameter	Declaration	Data type	Description
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
Status.ERR_ENCODER	STAT	BOOL	Short circuit / wire break at encoder signal (only with 1Count5V)
Status.ERR_DO2		BOOL	Short circuit / wire break / overheating DO2 (only with 1Count5V)
Status.ERR_PARA		BOOL	Parameter errors
Status.ERR_DO1		BOOL	Short circuit / wire break / overheating DO1
Status.ERR_24V		BOOL	Short circuit encoder supply

Synchronization

Synchronization can be performed by:

- Reference point approach or
- Reference setting

Connect the reference point switch to digital input DI.

During a reference point approach, ET 200S 1Count is capable of detecting the reference point based a rising edge at digital input DI.

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Digital Input

The digital input of the module is displayed in output parameter "DIn" of the input driver.

Digital outputs

The digital outputs can be controlled by means of the "DOut_1" and "DOut_2" input parameters of the input driver.

Error handling

On detecting an error report of the module, the input driver sets the "Error" and "Err.EncoderErr" error codes and resets the "Sync" status bit in the axis DB.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB.

EncoderFM350 (S7-300, S7-400)

Description

instruction EncoderFM350 is used to employ the FM 350-1 as position decoder module for incremental encoders.

The driver can also be used for the isochronous variant of this module.

Configuration of the module (hardware configuration)

Dialog	Setting
Interrupt selection	Enable diagnostic interrupts
Operating modes	• Count range: -31 to +31 bits • Operating mode: Continuous counting • Gate control: None
Inputs	If you want to perform a reference point approach, you must select the following setting: • Set counter (DI set): once • The setting "Evaluate zero mark for setting" may be selected or cleared as a reference point criterion.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
ReadDiagErr	IN-OUT	BOOL	Set this bit in OB 82 if the module has triggered a diagnostic interrupt (see Error handling).
DI_0	OUTPUT	BOOL	Digital input 0
DI_1	OUTPUT	BOOL	Digital input 1
DI_2	OUTPUT	BOOL	Digital input 2
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
JobErr	STAT	INT	Return value of instruction RD_REC, if "ReadDiagErr" = TRUE was set.
DiagStatus.EXT_VOLTAGE	STAT	STRUCT	External auxiliary voltage faulty
DiagStatus.NO_CONFIG	STAT	STRUCT	Configuration missing
DiagStatus.CONFIG_ERR	STAT	STRUCT	Incorrect configuration
DiagStatus.WTCH_DOG_FLT	STAT	STRUCT	Watchdog timeout
DiagStatus.RAM_FLT	STAT	STRUCT	RAM failure
DiagStatus.CH1_SIGA	STAT	STRUCT	Signal A faulty
DiagStatus.CH1_SIGB	STAT	STRUCT	Signal B faulty
DiagStatus.CH1_SIGZ	STAT	STRUCT	Signal N faulty

Parameter	Declaration	Data type	Description
DiagStatus.CH1_BETW	STAT	STRUCT	Error between channels
DiagStatus.CH1_5V2	STAT	STRUCT	Error at the 5.2 V encoder supply

Synchronization

Synchronization can be performed by:

- Reference point approach or
- Reference setting

Connect the reference point switch to digital input I2.

During a reference point approach, the module can detect the reference point in either of the following cases:

- Rising edge at digital input I2
- Rising edge of the zero mark, if digital input I2 is set

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Digital inputs

The digital inputs I0 to I2 are displayed in the "DI_0" to "DI_2" output parameters of the input driver.

Error handling

The module returns its error information by means of dataset 1 (DS1). In the following situations, the input driver reads DS1 from the module by calling instruction RD_REC and saves it to the static "DiagStatus" parameter:

- The input driver is initialized.
- The module reports an error.
- The driver input parameter "ReadDiagErr" is TRUE.

It then sets "ReadDiagErr" = FALSE. The return value of instruction RD_REC is written to the static "JobErr" parameter (see JOB_Err alarms).

If an error display is registered in dataset 1, the input driver sets the error codes "Error" and "Err.EncoderErr" to TRUE in the axis DB. Status bit "Sync" is set to FALSE.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB.

EncoderCPU314C (S7-300, S7-400)

Description

The instruction EncoderCPU314C is used to integrate a counter channel of CPU 314C for position feedback from 24 V incremental encoders.

Up to 4 counter channels are available.

Configuration of the counter function of CPU 314C (hardware configuration)

This input driver is only supports CPU 314C with firmware V2.0 and higher.

For each counter channel you want to use, set the following parameters in the Properties dialog of the module in slot 2.7 (count):

Parameter	Setting
Operating mode	Continuous counting
Hardware gate	No HW gate (check box cleared)

Customize the other settings according to the encoder used and to the requirements for your application.

Enter the address you have assigned in the hardware configuration to the inputs at slot 2.7 (count) in the axis data of the "Start address of inputs of the position decoder module" parameter ("InputModuleInAddress"). Set the address of the counter outputs at the "Start address of outputs of the position decoder module" parameter ("InputModuleOutAddress") .

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program.

Synchronization

Synchronization can be performed by:

- Reference point approach or
- Reference setting

Connect the reference point switch to the latch digital input of the corresponding channel (DI1.4 for channel 0 to DI1.7 for channel 3).

During a reference point approach, CPU 314C is capable of detecting the reference point at a rising edge at digital input DI.

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Among other things, the driver polls the status of the reference point switch during a reference point approach. The access actions that are necessary for reading the data considerably increase the instruction runtime.

Maximum count frequency

Note

The counter of CPU 314C does not output any values if its maximum counting frequency is exceeded (refer to the SIMATIC S7-300 CPU 31xC Technological Functions Manual (http://support.automation.siemens.com/WW/view/en/12429336)). The axis is halted and the error message "Following error exceeded" is output.

When specifying the maximum velocity ("MaxVelocity") in the axis data, you should therefore account for the limit frequency of the counter.

Error handling

The CPU 314C does not return encoder error information.

EncoderIM174 (S7-300, S7-400)

Description

instruction EncoderIM174 is used to employ IM 174 as position decoder module for incremental and absolute encoders.

IM 174 provides up to 4 axes. You need a driver instruction EncoderIM174 for each axis.

Configuration of IM 174 (hardware configuration)

Insert an IM 174 at an isochronous PROFIBUS DP in your project. Ensure the I/O addresses are in the process image of the CPU. Adjust the addresses manually if necessary. Assign the I/O addresses to a process image partition. An isochronous OB must be assigned to the process image partition in the configuration of the S7 CPU.

Parameter	Setting
I/O address	Specify the first input address for the axis to be used in Easy Motion Control in the axis DB. (Example for slot 4: input address 256 to 273; enter input address 256)
frame selection	You can select PROFIBUS frame 3 or frame 81 in the hardware configuration. Enter this information at input TelNumber of instruction EncoderIM174 as well.
611U compliance mode	When using an external reference mark, activate the 611U compliance mode.

Note

If frame 81 is used, IM 174 only transfers the encoder information to the CPU. Frame 81 is employed if you want to use IM 174 only to read encoder data.
In this case, it is not necessary to provide an output driver for IM 174 (OutputIM174).

Note

The hardware configuration implements IM 174 by default as an isochronous module. Isochronous processing is based on the condition that the I/O addresses of IM 174 are in the process image of the CPU.

For more information on the isochronous mode in SIMATIC S7, refer to the Isochronous Mode Function Manual (http://support.automation.siemens.com/WW/view/en/15218045).

The EncoderIM174 driver instruction demands isochronous processing. If you ignore this rule, the instruction is prevented from accessing the hardware addresses.

Configuration of IM 174 in the axis DB

Specify the input address set for this axis in the hardware configuration in the "Module address inputs" field.

The EncoderIM174 input driver calculates the axis number for IM 174 at runtime. This address can be read from the local data of the corresponding DB.

Input driver EncoderIM174 determines the cycle time that was set in the isochronous PROFIBUS DP at runtime and enters it in the axis DB. You are therefore strongly advised to read back the modified axis DB from the controller in order to apply all changes that may have been made.

Call

The instruction is called in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
RefMode	INPUT	INT	Referencing type: • 0 = Encoder zero mark • 1 = Positive edge at external zero mark (demands 611U compliant mode). • 4 = Positive edge at sensor 1 • 5 = Negative edge at sensor 1 • 6 = Positive edge at sensor 2 • 7 = Negative edge at sensor 2
TelNumber	INPUT	INT	Number of the PROFIBUS frame used: • 3 = PROFIBUS frame 3 • 81 = PROFIBUS frame 81
Err_ID	OUTPUT	WORD	Error code of the instruction: • W#16#0001 = Incorrect OB; instruction may only be installed in OB 6x • W#16#0002 = Incorrect frame number; IM 174 only supports PROFIBUS frames 3 and 81 • W#16#0004 = Incorrect axis number; IM 174 supports up to 4 axes. The hardware address you entered in the axis DB could not be assigned to an axis of IM 174. • W#16#0008 = Incorrect referencing mode setting. Check the parameter at input RefMode.

Synchronization

Synchronization can be performed by:

- Reference point approach (with incremental encoder)
- Reference point setting (with incremental and absolute encoder)

You can use two types of reference point switches for reference point approach:

- If operating with an encoder with zero mark that you want to use for referencing, set the parameter at input "RefMode" = 0. The 611U compliance mode may not be selected in this scenario.
- Configure input "RefMode" = 1 if you want to use an external reference mark (BERO) for referencing. Connect the external reference point switch to digital input B1 to B4, depending on the axis used in your application. For this setup, it is mandatory to select the 611U compliance mode in the hardware configuration.

The digital inputs M1 or M2 must be used for both adjustable sensors.

For more information, refer to the Distributed I/O PROFIBUS Module IM 174 manual (http://support.automation.siemens.com/WW/view/en/35014863).

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Error handling

In response to a detected error, the input driver sets the "Error" and "Err.EncoderErr" error codes in the axis DB. The input driver resets the "Sync" status bit in the axis DB if you are using an incremental encoder.

For more information on error response, error display, and error acknowledgment, see Error displays and error acknowledgment in the axis DB.

Error acknowledgment

If you set error acknowledgment in the axis DB of Easy Motion Control, errors pending in the module are also acknowledged.

EncoderSINAMICS (S7-300, S7-400)

Description

instruction EncoderSINAMICS is used to integrate a SINAMICS drive in Easy Motion Control. The drive may be integrated using PROFIBUS DP or PROFINET IO. You need an EncoderSINAMICS driver instruction for each SINAMICS axis.

Tools required

You need the following tools when using a SINAMICS in Easy Motion Control and to configure a SINAMICS:

- The drive configuration tool STARTER or S7 Technology, including the current SINAMICS Support Package (SSP)

Configuration of SINAMICS (hardware configuration)

Insert a SINAMICS into an isochronous PROFIBUS DP or PROFINET IO segment in your project. The EncoderSINAMICS driver instruction supports the standard frames 3, 5, 102, and 105 for PROFIBUS DP, as well as the standard frames 3 and 102 for PROFINET IO.

You need a suitable frame (5, 105) and an isochronous connection if you are going to use the Dynamic Servo Control (DSC) control method for your application. This DSC functionality is currently not supported for PROFINET IO.

For isochronous processing, make sure that the I/O addresses are in the process image of the CPU. Adjust the addresses manually if necessary. Assign the I/O addresses to a process image partition. An isochronous OB must be assigned to the process image partition in the configuration of the S7 CPU. For more information on the isochronous mode in SIMATIC S7,

refer to the Isochronous Mode Function Manual (http://support.automation.siemens.com/WW/view/en/15218045).

Parameter	Setting
I/O address	Specify the first input address for the axis to be used in Easy Motion Control in the axis DB. (Example for slot 4: input address 256 to 273; enter input address 256)
frame selection	A frame number is selected for communication from the SINAMICS configuration wizard and transferred to the hardware configuration. Enter this information at input TelNumber of instruction EncoderSINAMICS as well.

Note

The standard installation of TIA Portal does not support integration of the SINAMICS in a SIMATIC S7 CPU. Check the availability. Install the corresponding GSD file for SINAMICS. The necessary data is not integrated until you completed this additional installation.

Configuration of SINAMICS in the axis DB

Specify the input address set for this axis in the hardware configuration in the "Module address inputs" field.

Input driver EncoderSINAMICS determines the configured cycle time if a constant bus cycle time or an isochronous mode was detected and enters it in the axis DB. You are therefore strongly advised to read back the modified axis DB from the controller in order to apply all changes that may have been made.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description	
RefMode	INPUT	INT	Referencing type:	
			0 = Function 1 Reference mark 1	Referencing to encoder zero mark or external zero mark (p495[0] in SINAMICS)
			1 = Function 2 Reference mark 2	Second encoder zero mark
			2 = Function 3 Reference mark 3	Third encoder zero mark
			3 = Function 4 Reference mark 4	Fourth encoder zero mark
TelNumber	INPUT	INT	Frame number:	
			The following standard frames are supported: 3, 5, 102, and 105	Only the frames 3 and 102 have been approved for PROFINET IO.

Parameter	Declaration	Data type	Description	
EncoderValid	OUTPUT	BOOL	1= actual position values are valid	SINAMICS reports that the position encoder of the relevant axis returns valid values. This input may be used as activation condition at MC_Control.
Err_ID	OUTPUT	WORD	Error code of the instruction:	
			W#16#0001 = Incorrect OB	The instruction may only be called in OB 6x, OB 3x, or OB 1.
			W#16#0002 = Incorrect frame number	The driver does not support the specified number.
			W#16#0004 = Incorrect referencing mode setting	Check the parameter at input RefMode. Valid values are 0, 1, 2, or 3.

Note

The frame number at the input driver must agree with the frame number configured for SINAMICS. Otherwise, correct operation cannot be guaranteed. No error code is output if you make incorrect entries.

Synchronization

Synchronization can be performed by:

- Reference point approach (with incremental encoder)
- Reference point setting (with incremental and absolute encoder)

The referencing type can be specified using input parameter "RefMode" at the input driver. The following table lists the options available for Easy Motion Control.

RefMode	"Velocity" on MC_Home	Encoder zero mark	External zero mark p495[0]	Description
0	> 0	First encoder zero mark	0	Referencing to first encoder zero mark
1	> 0	Second encoder zero mark	0	Referencing to second encoder zero mark
2	> 0	Third encoder zero mark	0	Referencing to third encoder zero mark
3	> 0	Fourth encoder zero mark	0	Referencing to fourth encoder zero mark

The entry at RefMode controls the corresponding encoder control word for the drive so that the required functionality is realized in the SINAMICS. Input driver EncoderSINAMICS enables only the reference point approach to one or several encoder zero marks and to an external zero mark (alternative zero mark). Referencing on-the-fly is not possible. In order to reference to an external zero mark, set the p495[0] parameter of the axis concerned to a value between 1 and 6 value in the SINAMICS. To enable activation of a reference point approach, a traversing velocity greater than zero must be entered at instruction MC_HOME.

If traversing velocity zero has been entered, the contents of the "RefMode" input and the parameter p495[0] will be ignored.

Error handling

In response to a detected error, the input driver sets the "Error" and "Err.EncoderErr" error codes in the axis DB. The input driver resets the "Sync" status bit in the axis DB if you are using an incremental encoder.

For more information on error response, error display, and error acknowledgment, see Error displays and error acknowledgment in the axis DB.

Error acknowledgment

If you set error acknowledgment in the axis DB of Easy Motion Control, errors pending in the module are also acknowledged.

EncoderUniversal (S7-300, S7-400)

Description

instruction EncoderUniversal is used to integrate position decoder modules for which Easy Motion Control does not provide any special input driver.

Operating principle

instruction EncoderUniversal expects a value at its "EncoderValue" input that corresponds to the actual position of your axis. It does not access the I/O itself. You can interconnect the value provided by your position decoder module directly with the driver, or convert it beforehand.

If your distance measurement module offers special functions you wish to use, you should integrate these into your user program.

Configuration of the module (hardware configuration)

Select parameters according to your hardware and the encoder used.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error With this parameter, you inform the input driver of an error detected by the position decoder module or user program (for example, module removed, or station failure).
EncoderValue	INPUT	DINT	Encoder value

Synchronization

Synchronization with instruction EncoderUniversal is only possible by means of reference point setting.

Error handling

Any errors detected by your distance measurement module should be recorded and processed further in your user program.

If such an error impairs the functions you are performing with Easy Motion Control instructions, you should implement this error into Easy Motion Control error handling by means of "EncErr" input.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB.

Output driver (S7-300, S7-400)

Task of the output driver (S7-300, S7-400)

Function

The output driver is used to output the velocity setpoint of an axis which has been calculated by the controller to the connected output module. The instruction converts the velocity from [length unit/s] to the corresponding range of values of the output module and outputs this value.

If input "EnableDrive" = FALSE, the output driver outputs zero to the output module.

Initialization

The output value is set to zero.

Call

The output driver of an axis must be called unconditionally.

Isochronous Mode

The output drivers support the isochronous mode of all suitable I/O modules. The driver automatically recognizes isochronous mode by the fact that it is called in an isochronous OB.

Error handling

An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver at input "OutErr" . The output driver does not access the output module as long as input "OutErr" = TRUE.

If input"OutErr" = TRUE, the output driver sets the "Error" and "Err.OutputErr" error codes in the axis DB.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB.

Simulation mode

As long as the "Sim" bit is in the axis DB, the output driver does not access the output module (see instruction MC_Simulation (Page 5824)).

OutputIM178 (S7-300, S7-400)

Description

instruction OutputIM178 is used to employ a channel of the IM 178-4 as output module. instruction OutputIM178 is included for reasons of compatibility with STEP7 V5.x. Siemens is no longer actively marketing the IM 178-4. You can use IM 174 as an alternative.

IM 178-4 must be used with the same channel number and axis both at the input **and** output drivers, as specific address settings refer to both drivers. A combination with other drivers is not possible.

Call

The instruction is called in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable If "EnableDrive" = FALSE, a zero value is transferred to the output module. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
Q0	INPUT	BOOL	Digital output 0 of selected channel
Q1	INPUT	BOOL	Digital output 1 of selected channel
Q2	INPUT	BOOL	Digital output 2 of selected channel
OutErr	INPUT	BOOL	1 = Error at output module An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.

Digital outputs

The digital outputs "Q0" to "Q2"of the selected channel of IM 178-4 can be controlled using the input parameters "Q0" to "Q2" of the output driver.

OutputSM432 (S7-300, S7-400)

Description

instruction OutputSM432 is used to employ a channel of the SM 432 as output module.

Configuration of SM 432 (hardware configuration)

Set the output voltage of the corresponding SM 432 channel to ±10 V.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable If "EnableDrive" = FALSE, a zero value is transferred to the output module. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Error at output module An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.

OutputET200S2AO (S7-300, S7-400)

Description

instruction OutputET200S2AO is used to employ a channel of the ET 200S analog output module 2AO U as output module.

The driver can also be used for the isochronous variant of this module.

Configuration of the ET 200S analog output module 2AO U (hardware configuration)

Set the output voltage of the corresponding channel of the module to ±10 V.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable If "EnableDrive" = FALSE, a zero value is transferred to the output module. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Error at output module An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.

OutputCPU314C (S7-300, S7-400)

Description

instruction OutputCPU314C is used to employ an analog output channel of CPU 314C for ±10 V output voltage.

Configuration of the analog outputs of CPU 314C (hardware configuration)

Set the corresponding channel (analog output) of CPU 314C to an output voltage of ±10 V.

Set the address that you have assigned in the hardware configuration to the outputs at slot 2.6 (AI5/AO2) in the axis data parameter "Start address of the outputs of output module" ("OutputModuleOutAddress"). It is not necessary to configure the " Start address of the inputs of the output module" parameter for the output module ("OutputModuleInAddress").

Up to 2 analog output channels are available that allow you to operate the 2 axes entirely by means of CPU 314C.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable If "EnableDrive" = FALSE, the zero value is transferred to the analog output. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Analog output error Errors detected by the user program can be reported to the output driver here.

OutputSM332 (S7-300, S7-400)

Description

instruction OutputSM332 is used to employ a channel of the SM 332 as output module.

The driver can also be used for the isochronous variant of this module.

Configuration of SM 332 (hardware configuration)

Set the output voltage of the corresponding SM 332 channel to ±10 V.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable If "EnableDrive" = FALSE, a zero value is transferred to the output module. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Error at output module An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.

OutputUniversal (S7-300, S7-400)

Description

instruction OutputUniversal allows you to use any analog output module in combination with the instructions of Easy Motion Control.

Operating principle

instruction OutputUniversal calculates a speed setpoint based on the velocity value output by the controller and proportionality ratio "IntAtMaxVelocity" and provides the result at its outputs.

The driver itself does not access the I/O.

"You must use the value of "IntAtMaxVelocity" to control your analog output module so that it outputs a voltage that will move the axis at "MaxVelocity" (you have already declared the value of this voltage in the "DriveInputAtMaxVel" parameter).

The driver curve always passes through zero and follows a linear course.

The speed setpoint is output both as signed integer value and as an amount with a separate sign.

If these output values already correspond to your application, you can output them directly to the I/O via the user program. If the values are not yet suitable, you can also make an additional adjustment via the user program and output the value that is generated.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable A zero value is returned at the outputs as soon as "EnableDrive" = FALSE. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
IntAtMaxVelocity	INPUT	INT	Integer value for converting the speed setpoint (see below).
OutErr	INPUT	BOOL	1 = Error at output module An error detected on the output module by the user program (for example, module removed or station failure) can be signaled to the output driver here.
OutputValue	OUTPUT	INT	Calculated speed setpoint
OutValAbs	OUTPUT	INT	Absolute speed setpoint value
OutValSign	OUTPUT	BOOL	Sign of speed setpoint (TRUE = Negative value)

OutputMM4_DP (S7-300, S7-400)

Description

instruction OutputMM4_DP is used to implement of a MICROMASTER 4 with optional DP add-on module.

Configuration of MM4 (hardware configuration)

For information on MICROMASTER 4, refer to the hardware catalog at:

"Additional field devices > PROFIBUS DP > Drives > Siemens AG > SIMOVERT > MICROMASTER 4".

You may first have to install the corresponding GSD file.

Position the MICROMASTER in your DP master system and assign it a DP address. Insert a "0 PKW, 2 PZD (PPO3)" module into slots 0 and 1. Either accept the default I/O addresses or select your own values.

Transfer these addresses to "Start address of the outputs of the output module" or "Start address of the inputs of the output module" in the axis DB for your axis.

Configuring the MM4

Configure the MICROMASTER (motor data, DP address, operation as DP slave, etc.) using the MICROMASTER documentation according to your requirements.

Transfer the "reference frequency" specified in the P2000 parameter as "reference value for 100 % speed" ("DriveInputAt100") to the axis DB for your axis.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable The zero speed value is output to the MICROMASTER as soon as "EnableDrive" = FALSE. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Output error An error detected on the output module by the user program (for example, station failure) can be signaled to the output driver here.
ON	INPUT	BOOL	1 = Drive switched on "ON" in entered in bit 0 in the controller word.
DriveReady	OUTPUT	BOOL	1 = Drive ready If "DriveReady" = FALSE, starting the travel generates a "DriveError".

OutputIM174 (S7-300, S7-400)

Description

instruction OutputIM174 is used to employ a channel of the IM 174 as output module. IM 174 provides up to 4 axes. You need a driver instruction OutputIM174 for each axis for which PROFIBUS frame 3 is set.

Configuration of IM 174 (hardware configuration)

You only need output driver OutputIM174 if you configure an axis with PROFIBUS frame 3. You do not need an output driver for an axis with PROFIBUS frame 81.

Insert an IM 174 at an isochronous PROFIBUS DP in your project. Ensure the I/O addresses are in the process image of the CPU. Adjust the addresses manually if necessary. Assign the

I/O addresses to a process image partition. An isochronous OB must be assigned to the process image partition in the configuration of the S7 CPU.

Parameter	Setting
I/O address	Specify the first output address for the axis to be used in Easy Motion Control in the axis DB. (Example for slot 5: For start address 256 to 265, you must enter output address 256)

Note

The hardware configuration implements IM 174 by default as an isochronous module. Isochronous processing is based on the condition that the I/O addresses of IM 174 are in the process image of the CPU.

For more information on the isochronous mode in SIMATIC S7, refer to the Isochronous Mode Function Manual (http://support.automation.siemens.com/WW/view/en/15218045).

The OutputIM174 driver instruction demands isochronous processing. If you ignore this rule, the instruction is prevented from accessing the hardware addresses.

Configuration of IM 174 in the axis DB

Specify the output address set for this axis in the hardware configuration in the "Module address outputs" field.

Call

The instruction is called in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable The zero speed value is output to the drive as soon as "EnableDrive" = FALSE. The drive enable is locked. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Error at output module An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.
Err_ID	OUTPUT	WORD	Error code of the instruction W#16#0001 = Incorrect OB; instruction may only be installed in OB 6x

OutputSINAMICS (S7-300, S7-400)

Description

instruction OutputSINAMICS is used to integrate a SINAMICS drive in Easy Motion Control. The drive may be integrated using PROFIBUS DP or PROFINET IO. You need an OutputSINAMICS driver instruction for each SINAMICS axis.

Configuration of SINAMICS (hardware configuration)

Insert a SINAMICS into an isochronous PROFIBUS DP or PROFINET IO segment in your project. The OutputSINAMICS driver instruction supports the standard frames 3, 5, 102, and 105 for PROFIBUS DP, as well as the standard frames 3 and 102 for PROFINET IO.

You need a suitable frame (5, 105) and an isochronous connection if you are going to use the Dynamic Servo Control (DSC) control method for your application. This DSC functionality is currently not supported for PROFINET IO.

For isochronous processing, make sure that the I/O addresses are in the process image of the CPU. Adjust the addresses manually if necessary. Assign the I/O addresses to a process image partition. An isochronous OB must be assigned to the process image partition in the configuration of the S7 CPU. For more information on the isochronous mode in SIMATIC S7, refer to the Isochronous Mode Function Manual (http://support.automation.siemens.com/WW/view/en/15218045).

Parameter	Setting
I/O address	Specify the first output address for the axis to be used in Easy Motion Control in the axis DB. (Example for slot 5: For start address 256 to 265, you must enter output address 256)

Note

The standard installation of TIA Portal does not support integration of the SINAMICS in a SIMATIC S7 CPU. Check the availability. Install the corresponding GSD file for SINAMICS. The necessary data is not integrated until you completed this additional installation.

Configuration of SINAMICS in the axis DB

Specify the output address set for this axis in the hardware configuration in the "Module address outputs" field. Specify the frame number once again to enable stand-alone operation of the output driver. The program checks if the frame number agrees with that of the corresponding EncoderSINAMICS input driver (if that exists).

Activating the axis

Each SINAMICS motor module inevitably assigned a central power supply (line module) for the DC link. Always ensure that the central power supply is switched on before you activate the axis by setting input "EnableDrive" of instruction MC_Control. The line module returns corresponding information on its operating status. Evaluate the necessary feedback signals from the line module and link these as power-on conditions.

For more information, refer to the SINAMICS Drives documentation (http://www.automation.siemens.com/doconweb).

An example of the application for switching on an active line module is available on the Internet (http://support.automation.siemens.com/WW/view/en/21971603).

Switching off the axis - Configuration of the StopMode

Coasting (coast stop) (AUS2) – StopMode = 3:

In this case, the power enable signal is immediately cleared and the drive coasts to a standstill in the power-off state. The mechanical brake control is enabled immediately and simultaneously. This engages the brake for active deceleration of the drive. Mechanical stress on the brake increases in the process depending on its inertia load.

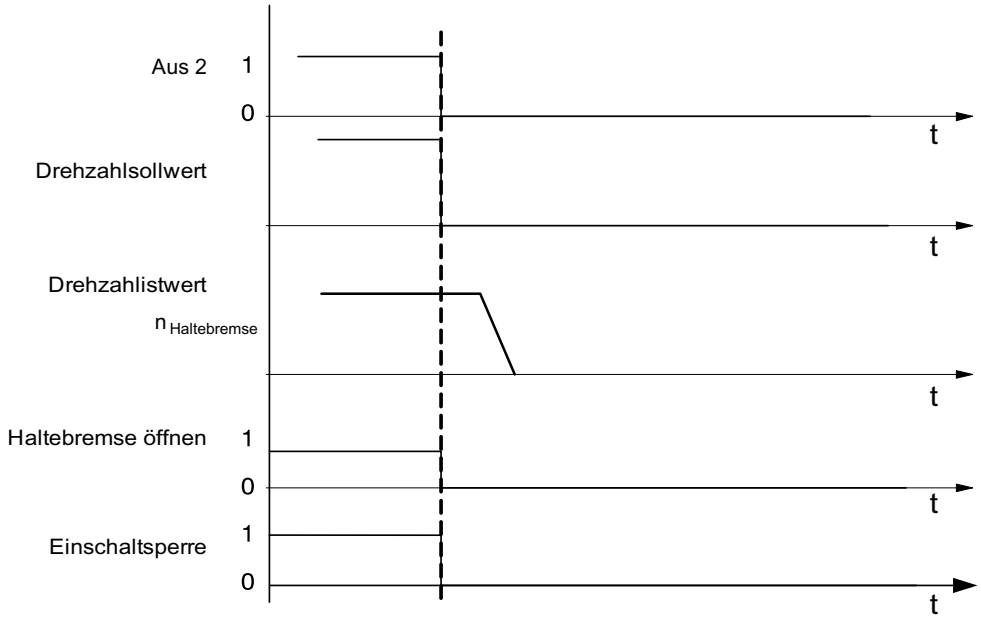

Ramp-down stop (AUS1) – StopMode = 5:

In this mode, the drive is decelerated with its ramp-up function. Once the speed threshold has been reached, the holding brake is engaged and pulse enable is deactivated on expiration of the brake closing time.

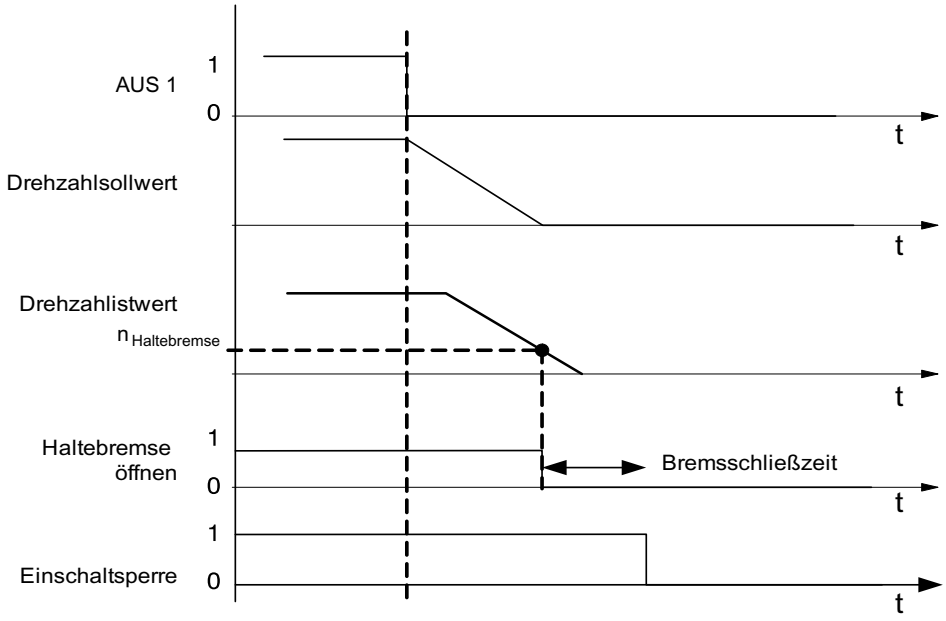

Fast stop (AUS3) – StopMode = 6:

In this mode, the drive is decelerated and brought to a stop with the fast stop ramp (emergency stop). Once the speed threshold has been reached, the holding brake is engaged and pulse enable is deactivated on expiration of the brake closing time.

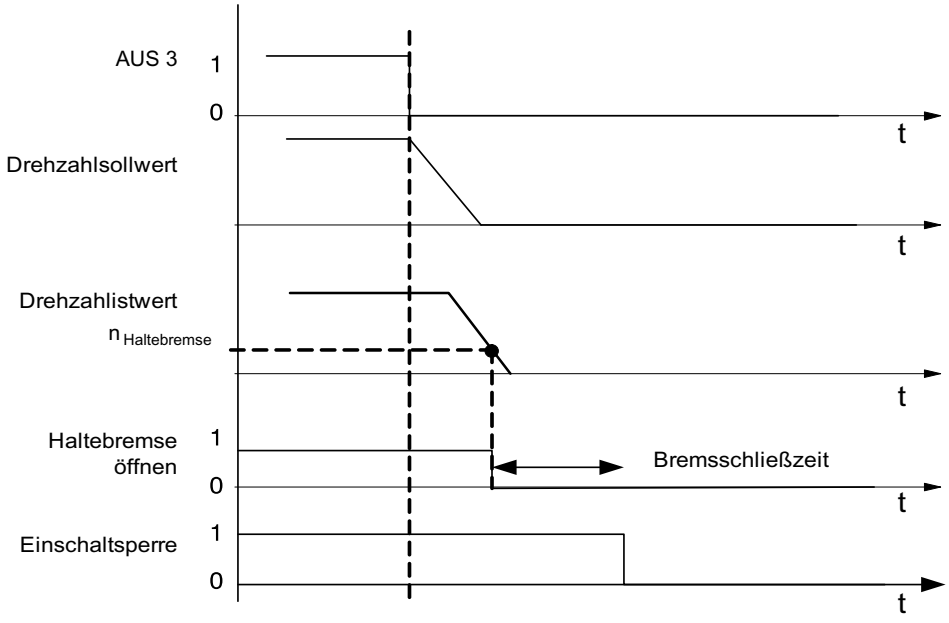

For more information, refer to the SINAMICS S120 product information (http://support.automation.siemens.com/WW/view/en/30119684).

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description	
TelNumber	INPUT	INT	Frame number:	
			The following standard frames are supported: 3, 5, 102,105	Only the frames 3 and 102 have been approved for PROFINET IO. If you are using EncoderSINAMICS as input driver, the number of the frame used must always remain the same.
EnableDrive	INPUT	BOOL	1 = Drive enable	The zero speed value is output to the drive as soon as "EnableDrive" = FALSE. The drive enable is locked. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
StopMode	INPUT	INT	Stop mode:	
			3 = Coast to stop (AUS2)	The drive coasts to a stop in power-off state
			5 = Ramp stop (AUS1)	Stops the axis by using a ramp function generator
			6 = Quick stop (AUS3)	Stops the axis along a quick stop ramp
OutErr	INPUT	BOOL	1 = Error at output module	An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.
Operation_Enabled	OUTPUT	BOOL	1 = SINAMICS reports "Operation enabled"	All axis enable signals are set in the drive and the axis reports that it is ready for movement. The starting current is applied to the axis.
Err_ID	OUTPUT	WORD	Error code of the instruction:	
			W#16#0001 = Incorrect OB	The instruction may only be installed in OB 6x, OB 3x, or OB 1.
			W#16#0002 = Incorrect frame number	The specified number is not supported by the driver, or differs from the frame number that has been configured in the SINAMICS.

OutErr: The drive (SINAMICS) is deactivated on the positive edge of this variable. To reactivate the drive, OutErr = FALSE must be set and the error message must be acknowledged.

Operation_Enabled: This variable reports the current axis status in the SINAMICS. You should not activate a travel instruction while Operation_Enabled = FALSE.

For more information on integration of the SINAMICS, refer to the Internet (http://support.automation.siemens.com/WW/view/en/30119684).

Additional references for Easy Motion Control (S7-300, S7-400)

Axis data block (axis DB) (S7-300, S7-400)

Function

The axis DB is the central database of Easy Motion Control. It contains:

- Parameters that provide basic information about the axis (for example, I/O addresses)
- Position control parameters that are calculated from these basic parameters during initialization of the EMC instructions
- Actual values of the position control (for example, position, error states, etc.)
- Initialization bits for coordinating the start of the instructions

Contents

The following table shows the content of the axis DB.

Note

You may not change any data not listed in this table.

Address	Parameter	Data type	Initial value	Description
0.0	Sample_T	REAL	0.01	Scan time of the axis instructions [s]
4.0	AxisType	BOOL	FALSE	Axis type • FALSE = Linear axis • TRUE = Rotary axis
4.1	EncoderType	BOOL	FALSE	Encoder type • FALSE = Incremental encoder • TRUE = Absolute encoder
4.2	Sim	BOOL	FALSE	TRUE = Simulation mode is active
6.0	AxisLimitMax	REAL	$1.0e^{+6}$	Software limit switch end [length unit]
10.0	AxisLimitMin	REAL	$-1.0e^{+6}$	Software limit switch start [length unit]
14.0	MaxVelocity	REAL	0.1	Maximum axis velocity [unit of length/s]
18.0	MaxAcceleration	REAL	0.1	Max. axis acceleration in [unit of length/s^2]
22.0	MaxDeceleration	REAL	0.1	Max. axis deceleration in [unit of length/s^2]
26.0	MonTimeTargetAppr	REAL	1.0	Monitoring time for target approach [s]
30.0	TargetRange	REAL	1.0	Target range on positioning [unit of length]
34.0	StandstillRange	REAL	1.5	Standstill range without travel order [unit of length]
38.0	MaxFollowingDist	REAL	5.0	Maximum admissible. following error [unit of length]
42.0	MonitorTargetAppr	BOOL	TRUE	TRUE = Enable monitoring of target range
42.1	SWLimitEnable	BOOL	TRUE	TRUE = Enable software limit switch monitoring

Address	Parameter	Data type	Initial value	Description
44.0	FactorP	REAL	1.0	Proportionality factor [1/s]
48.0	ManVelocity	REAL	0.0	Setpoint velocity in manual mode [unit of length/s]
52.0	ManEnable	BOOL	FALSE	TRUE = Enable manual mode
54.0	EmergencyDec	REAL	1000.0	Axis deceleration for emergency stop [length unit/s^2]
58.0	InputModuleInAddr	INT	0	Start address for the position detection module inputs
60.0	InputModuleOutAddr	INT	0	Start address for the position detection module outputs
62.0	InputChannelNo	INT	0	Channel number of position input module
64.0	StepsPerRev	DINT	L#1	Steps per encoder revolution [steps/encoder revolution]
68.0	DisplacementPerRev	REAL	1.0	Distance per encoder revolution [unit of length/encoder revolution]
72.0	NumberRevs	INT	1	Number of revolutions of an absolute encoder
74.0	PolarityEncoder	INT	1	Set encoder polarity
76.0	PositionOffset	DINT	L#0	Position offset / absolute encoder adjustment [step]
80.0	OutputModuleOutAddr	INT	0	Start address for the output module outputs
82.0	OutputModuleInAddr	INT	0	Start address of the output module inputs
84.0	OutputChannelNo	INT	0	Channel number of the output module
86.0	PolarityDrive	INT	1	Set drive polarity
88.0	DriveInputAtMaxVel	REAL	9.0	Reference value for max. axis velocity [V]
92.0	OffsetCompensation	REAL	0.0	Offset compensation [V]
96.0	DriveInputAt100	REAL	10.0	Reference value for 100% speed [V]
100.0	Override	REAL	100.0	Velocity override [%]
104.0	ActPosition	REAL	0.0	Current actual position [unit of length]
108.0	FollowingDistance	REAL	0.0	Current following error [unit of length]
112.0	RemainingDistance	REAL	0.0	Current residual distance to target [unit of length]
116.0	NomVelocity	REAL	0.0	Setpoint velocity of axis [unit of length/s]
120.0	ActVelocity	REAL	0.0	Actual speed of axis [unit of length/s]
124.0	Sync	BOOL	FALSE	• FALSE = Axis not synchronized • TRUE = Axis synchronized
124.1	Error	BOOL	TRUE	• FALSE = No error • TRUE = Group error
124.2	ErrorAck	BOOL	FALSE	• FALSE = No acknowledgment • TRUE = Acknowledge error
128.0	Err.SWLimitMinExceeded	BOOL	FALSE	TRUE = Software limit switch start exceeded
128.1	Err.SWLimitMaxExceeded	BOOL	FALSE	TRUE = Software limit switch end exceeded
128.2	Err.TargetErr	BOOL	FALSE	TRUE = Target outside travel range
128.3	Err.NoSync	BOOL	FALSE	TRUE = Axis not synchronized
128.4	Err.DirectionErr	BOOL	FALSE	TRUE = Travel in specified direction inadmissible
128.5	Err.DataErr	BOOL	FALSE	TRUE = Invalid parameter in travel instruction
128.6	Err.StartErr	BOOL	FALSE	TRUE = Start not possible in current axis status
128.7	Err.DistanceErr	BOOL	FALSE	TRUE = Overtravel
129.0	Err.MasterErr	BOOL	FALSE	TRUE = Master axis in error state

Address	Parameter	Data type	Initial value	Description
130.0	Err.StoppedMotion	BOOL	TRUE	TRUE = Axis in stop status requiring acknowledgment
130.1	Err.EnableDriveErr	BOOL	FALSE	TRUE = Drive enable missing
130.2	Err.FollowingDistErr	BOOL	FALSE	TRUE = Max. following error exceeded
130.3	Err.StandstillErr	BOOL	FALSE	TRUE = Standstill range exceeded
130.4	Err.TargetApproachErr	BOOL	FALSE	TRUE = Error on target approach
130.5	Err.EncoderErr	BOOL	FALSE	TRUE = Encoder error
130.6	Err.OutputErr	BOOL	FALSE	TRUE = Error at output driver
130.7	Err.ConfigErr	BOOL	FALSE	TRUE = Axis data incorrectly configured
131.0	Err.DriveErr	BOOL	FALSE	TRUE = Drive error
132.0	Config.Err_AxisLimit	BOOL	FALSE	TRUE = Axis limits incorrect
132.1	Config.Err_MaxVelocity	BOOL	FALSE	TRUE = Max. velocity incorrect
132.2	Config.Err_MaxAcceleration	BOOL	FALSE	TRUE = Max. acceleration incorrect
132.3	Config.Err_MaxDeceleration	BOOL	FALSE	TRUE = Max. deceleration incorrect
132.4	Config.Err_MonTimeTarge-tAppr	BOOL	FALSE	TRUE = Monitoring time for target approach incorrect
132.5	Config.Err_TargetRange	BOOL	FALSE	TRUE = Target range incorrect
132.6	Config.Err_StandstillRange	BOOL	FALSE	TRUE = Standstill range incorrect
132.7	Config.Err_MaxFollowingDist	BOOL	FALSE	TRUE = Max. following error incorrect
133.0	Config.Err_EmergencyDec	BOOL	FALSE	TRUE = Deceleration for hard stop incorrect
133.1	Config.Err_StepsPerRev	BOOL	FALSE	TRUE = Steps per encoder revolution incorrect
133.2	Config.Err_DisplacementPer-Rev	BOOL	FALSE	TRUE = Axis distance per encoder revolution incorrect
133.3	Config.Err_NumberRevs	BOOL	FALSE	TRUE = Incorrect number of encoder revolutions
133.4	Config.Err_PolarityEncoder	BOOL	FALSE	TRUE = Set encoder polarity incorrect
133.5	Config.Err_PolarityDrive	BOOL	FALSE	TRUE = Set drive polarity incorrect
133.6	Config.Err_DriveInputAtMax-Vel	BOOL	FALSE	TRUE =reference value for maximum velocity incorrect
133.7	Config.Err_AxisLength	BOOL	FALSE	TRUE = Length of axis incorrect
134.0	Config.Err_EncoderRange	BOOL	FALSE	TRUE = Encoder range is not suitable for axis length
134.1	Config.Err_MaxVelRotaryAxis	BOOL	FALSE	TRUE = Max. velocity and scan time do not match rotary axis module
134.2	Config.Err_DriveInputAt100	BOOL	FALSE	TRUE = Reference value for 100 % speed ≤ 0
136.0	EncoderValue	DINT	L#0	Current encoder value
278.0	Init	STRUCT	TRUE	Bit array for initializing 32 instructions

4.2.4.3 Function modules (S7-300, S7-400)

FM x50-1 Counting (S7-300, S7-400)

CNT_CTRL (S7-300, S7-400)

Description

The CNT_CTRL instruction is used to control the counting and measuring modes.

You need to use the CNT_CTL1 (Page 5872) instruction to change parameters in runtime or set and reset the outputs of the counter module.

Operating principle

The CNT_CTRL instruction works only in a non-isochronous OB.
In an isochronous OB, use the instruction CNT_CTL1 (Page 5872) or CNT_CTL2 (Page 5874).

The data needed for the CNT_CTRL instruction is stored in the counter DB (Page 5875) in the CPU. The CNT_CTRL instruction cyclically transfers data from this DB to the counter module and fetches data from the counter module.

Call

The CNT_CTRL instruction can be called once for each counter in the cycle or optionally in a time-controlled program. The call is not allowed in an event-driven interrupt program.

Parameter

Parameter	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	INT	Number of the counter data block	Enters this	Queries this
SW_GATE	INPUT	BOOL	Counter control bit SW gate (start/stop)	Sets and resets this	Queries this
GATE_STP	INPUT	BOOL	Counter control bit gate stop	Sets and resets this	Queries this
OT_ERR_A	INPUT	BOOL	Acknowledge operator error	Sets and resets this	Queries this
OT_ERR	OUTPUT	BOOL	Operator error occurred	Queries this	Sets and resets this
L_DIRECT	IN-OUT	BOOL	Trigger bit for direct loading of a counter	Sets this	Queries and resets this
L_PREPAR	IN-OUT	BOOL	Trigger bit for preparatory loading of a counter	Sets this	Queries and resets this
T_CMP_V1	IN-OUT	BOOL	Transfer trigger bit for comparison value 1	Sets this	Queries and resets this

Parameter	Declaration	Data type	Description	The user ...	The instruction ...
T_CMP_V2	IN-OUT	BOOL	Transfer trigger bit for comparison value 2	Sets this	Queries and resets this
RES_SYNC	IN-OUT	BOOL	Delete synchronization status bit	Sets this	Queries and resets this
RES_ZERO	IN-OUT	BOOL	Delete zero crossing status bit	Sets this	Queries and resets this

Executing jobs

Initiate jobs for the counter module using the instruction parameters L_DIRECT, L_PREPAR, T_CMP_V1, T_CMP_V2, RES_SYNC, RES_ZERO, OT_ERR_A and GATE_STP.

Depending on the command, you need to enter the load value or a comparison value prior to the calling of the instruction in the counter DB.

A configured IN/OUT parameter (L_DIRECT, L_PREPAR, T_CMP_V1, T_CMP_V2, RES_SYNC and RES_ZERO) is deleted by the CNT_CTRL instruction upon completion of the command.

Startup characteristics

As soon as the CNT_CTRL instruction detects a startup (CPU or FM startup), any pending command is deferred and the startup is first acknowledged. A command you have already initiated is not lost, but is not executed until the startup is completed.

Reaction to errors

An operator error occurring during the call of the instruction is reported at the OT_ERR parameter. The error information can be read from the counter DB (OT_ERR_B tag). You can then use the OT_ERR_A parameter to acknowledge the operator error. No additional operator error is reported until you acknowledge the previous error.

DIAG_INF (S7-300, S7-400)

Description

If the trigger parameter is set (IN_DIAG = TRUE), the DIAG_INF instruction reads data record DS 1 from the counter module and makes it available in the counter DB (Page 5875) (starting at DW 54).

Operating principle

The data record is transferred using the RDSYSST instruction. The return code of this instruction (RET_VAL) is copied to the RET_VAL parameter of the DIAG_INF instruction. Once DIAG_INF instruction has been executed, the IN_DIAG trigger parameter is reset and the transfer is reported as completed.

Call

The DIAG_INF instruction can be called in the cycle and in the interrupt program. However, calling it in a time-controlled program is not recommended.

Parameter

Parameter	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	INT	Number of the counter data block	Enters this	Queries this
RET_VAL	OUTPUT	INT	Return code of the instruction RDSYSST	Queries this	Enters this
IN_DIAG	IN-OUT	BOOL	Read trigger bit of diagnostics data record DS 1	Sets and queries this	Resets this

CNT_CTL1 (S7-300, S7-400)

Description

The CNT_CTL1 instruction is used to control the counting and measuring modes.

In addition to the functions in the CNT_CTRL instruction , the CNT_CTL1 instruction enables you to change parameters in runtime and to set and reset the outputs of the counter module.

Operating principle

The data needed for the CNT_CTL1 instruction is stored in the counter DB (Page 5875) in the CPU. The CNT_CTL1 instruction cyclically transfers data from this DB to the counter module and fetches data from the counter module.

Call

The CNT_CTL1 instruction can be called once for each counter in the cycle or optionally in a time-controlled or an isochronous interrupt OB. The call is not allowed in an event-driven interrupt program.

Parameter

Parameter	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	INT	Number of the counter data block	Enters this	Queries this
SW_GATE	INPUT	BOOL	Counter control bit SW gate (start/stop)	Sets and resets this	Queries this
GATE_STP	INPUT	BOOL	Counter control bit gate stop	Sets and resets this	Queries this
OT_ERR_A	INPUT	BOOL	Acknowledge operator error	Sets and resets this	Queries this

Parameter	Declaration	Data type	Description	The user ...	The instruction ...
SET_DO0	INPUT	BOOL	Set/Reset output DO0	Sets and resets this	Queries this
SET_DO1	INPUT	BOOL	Set/Reset output DO1	Sets and resets this	Queries this
OT_ERR	OUTPUT	BOOL	Operator error occurred	Queries this	Sets and resets this
L_DIRECT [2] [3]	IN-OUT	BOOL	**Counting:** Trigger bit for direct and preparatory loading of a counter	Sets this	Queries and resets this
			Measuring: Must NOT be set	-	Queries and resets this
L_PREPAR [2] [3]	IN-OUT	BOOL	**Counting:** Trigger bit for preparatory loading of a counter	Sets this	Queries and resets this
			Measuring: Transfer low limit	Sets this	Queries and resets this
T_CMP_V1 [2]	IN-OUT	BOOL	**Counting:** Transfer trigger bit for comparison value 1	Sets this	Queries and resets this
			Measuring: Transfer high limit	Sets this	Queries and resets this
T_CMP_V2 [2]	IN-OUT	BOOL	**Counting:** Transfer trigger bit for comparison value 2	Sets this	Queries and resets this
			Measuring: Update time	Sets this	Queries and resets this
C_DOPARA [1]	IN-OUT	BOOL	Trigger bit for parameter change	Sets this	Queries and resets this
RES_SYNC	IN-OUT	BOOL	Delete synchronization status bit	Sets this	Queries and resets this
RES_ZERO	IN-OUT	BOOL	Reset status bits for zero pass, overflow, underflow and comparator or measurement end	Sets this	Queries and resets this

[1] You may not set this parameter simultaneously with one of the L_DIRECT, L_PREPAR, T_CMP_V1 or T_CMP_V2 parameters.

[2] You may not set these parameters simultaneously with the C_DOPARA parameter.

[3] You may not set these parameter simultaneously.

Executing jobs

Initiate jobs for FM x50-1 using the instruction parameters L_DIRECT, L_PREPAR, T_CMP_V1, T_CMP_V2, C_DOPARA, RES_SYNC, RES_ZERO and OT_ERR_A.

Depending on the command, enter the corresponding values (load value, comparison value, low limit, high limit and the update time) in the counter DB prior to calling the instruction.

See also:

- Value transfer using CNT_CTL1 (Page 5881)
- DB parameters for value transfer (count modes) (Page 5882)
- DB parameters for value transfer (count modes) (Page 5884)

The CNT_CTL1 instruction deletes an IN-OUT parameter (L_DIRECT, L_PREPAR, T_CMP_V1, T_CMP_V2, C_DOPARA, RES_SYNC and RES_ZERO) when the command is completed. This enables you to recognize that the command has been completed by FM x50-1. If necessary, you can let this information be known to your user program.

Startup characteristics

As soon as the CNT_CTL1 instruction detects a startup (CPU or FM startup), any pending command is deferred and the startup is first acknowledged. A command you have already initiated is not lost, but is not executed until the startup is completed.

Reaction to error

An operator error occurring during the call of the instruction is reported at the OT_ERR parameter. The error information can be read from the counter DB (OT_ERR_B tag). You can then use the OT_ERR_A parameter to acknowledge the operator error. No additional operator error is reported until you acknowledge the previous error.

CNT_CTL2 (S7-300, S7-400)

Description

The function of the CNT_CTL2 instruction is basically the same as the CNT_CTL1 (Page 5872) instruction.

The differences compared to the CNT_CTL1 instruction are discussed in the following section.

Operating principle

The CNT_CTL2 instruction is particularly suitable for applications involving rapid, successive transfer of the same command (for example, load comparison value) to FM x50-1. At the very best, the CNT_CTL1 instruction lets you initiate new jobs at every fifth PROFIBUS DP cycle, while the CNT_CTL2 instruction can handle this at every second PROFIBUS DP cycle.

The instruction is ready to process a command as soon as the corresponding trigger bit is set to 0. The completion of a command is not displayed separately.

If a communication problem or a data or operator error occurs, this can no longer be attributed to a specific command. For this reason, the instruction interrupts command execution and generates an **ID 90** operator error requiring acknowledgment. Once you acknowledged the error by setting the OT_ERR_A parameter, the instruction continues to process any jobs pending.

The acknowledgment of an operator error is successfully completed as soon as OT_ERR parameter is reset. You should keep the OT_ERR_A parameter set until this happens to ensure

reliable acknowledgment. You should not initiate additional jobs until the acknowledgment is successful.

Note

It is not allowed to initiate several value transfers simultaneously with the CNT_CTL2 instruction.

Call

The CNT_CTL2 instruction works only in an isochronous OB.

If the CNT_CTL2 instruction is called in a non-isochronous OB, it generates an **ID 91** operator error. This would also prevent data communication with FM x50-1.

Additional references for FM x50-1 Counter V3 (S7-300, S7-400)

Counter data block (S7-300, S7-400)

Function

The counter DB is the data interface between the user program and FM x50-1. It contains and applies all data necessary to control and operate the module.

Before the parameter assignment of the module, you must provide the following valid data to the counter DB.

- Module address (address 6.0)
- Channel start address (address 8.0)
- Length of the user data interface (address 12.0)

The counter module manual contains an example of how you can implement the transfer of the module address, the channel address, and the length of the user data to the counter DB in OB 100.

Alternatively, you can also have the module address entered automatically using the "Enter module address in DB" function in the device configuration of FM x50-1.

Configuration

The counter DB is divided into different areas:

Channel DB
Instruction parameters, addresses
Transfer area for values to be written
Control interface
Transfer area for values to be read
Error numbers

Channel DB
Feedback interface
Parameters for FM 450-1
Diagnostics data

Contents

The table below shows the contents of the counter DB.

Note

You may not change any data not listed in this table.

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
Instruction parameters, addresses					
0.0	AR1_BUFFER	DWORD	DW#16#0	AR1 Buffer	AR1 Buffer
4.0	FP	BYTE	B#16#0	Flag byte	Flag byte
5.0	RESERVED	BYTE	B#16#0	Reserved	Reserved
6.0	MOD_ADR	WORD	W#16#0	Module address	Module address
8.0	CH_ADR	DWORD	DW#16#0	Channel address	Channel address
12.0	U_D_LGTH	BYTE	B#16#0	Length user data	Length user data
13.0	A_BYTE_0	BYTE	B#16#0	Reserved	Reserved
Transfer area for values to be written					
14.0	LOAD_VAL [1)]	DINT	L#0	New load value (write user)	Low limit (write user)
18.0	CMP_V1 [1)]	DINT	L#0	New comparison value 1 (write user)	High limit (write user)
22.0	CMP_V2 [1)]	DINT	L#0	New comparison value 2 (write user)	Update time (write user)
Control interface					
26.0	A_BIT0_0	BOOL	FALSE	Reserved	Reserved
26.1	TFB	BOOL	FALSE	Reserved / test free	Reserved
26.2	A_BIT0_2	BOOL	FALSE	Reserved	Reserved
26.3	A_BIT0_3	BOOL	FALSE	Reserved	Reserved
26.4	A_BIT0_4	BOOL	FALSE	Reserved	Reserved
26.5	A_BIT0_5	BOOL	FALSE	Reserved	Reserved
26.6	A_BIT0_6	BOOL	FALSE	Reserved	Reserved
26.7	A_BIT0_7	BOOL	FALSE	Reserved	Reserved
27.0	ENSET_UP [1)]	BOOL	FALSE	Enable setting in forward direction (write user)	-

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
27.1	ENSET_DN [1]	BOOL	FALSE	Enable setting in backward direction (write user)	-
27.2	A_BIT1_2	BOOL	FALSE	Reserved	Reserved
27.3	A_BIT1_3	BOOL	FALSE	Reserved	Reserved
27.4	A_BIT1_4	BOOL	FALSE	Reserved	Reserved
27.5	A_BIT1_5	BOOL	FALSE	Reserved	Reserved
27.6	A_BIT1_6	BOOL	FALSE	Reserved	Reserved
27.7	A_BIT1_7	BOOL	FALSE	Reserved	Reserved
28.0	CTRL_DO0 [1]	BOOL	FALSE	Enable digital output DO0 (write user)	Enable digital output DO0 (write user)
28.1	CTRL_DO1 [1]	BOOL	FALSE	Enable digital output DO1 (write user)	Enable digital output DO1 (write user)
28.2	A_BIT2_2	BOOL	FALSE	Reserved	Reserved
28.3	A_BIT2_3	BOOL	FALSE	Reserved	Reserved
28.4	A_BIT2_4	BOOL	FALSE	Reserved	Reserved
28.5	A_BIT2_5	BOOL	FALSE	Reserved	Reserved
28.6	A_BIT2_6	BOOL	FALSE	Reserved	Reserved
28.7	A_BIT2_7	BOOL	FALSE	Reserved	Reserved
29.0	A_BIT3_0	BOOL	FALSE	Reserved	Reserved
29.1	A_BIT3_1	BOOL	FALSE	Reserved	Reserved
29.2	A_BIT3_2	BOOL	FALSE	Reserved	Reserved
29.3	A_BIT3_3	BOOL	FALSE	Reserved	Reserved
29.4	A_BIT3_4	BOOL	FALSE	Reserved	Reserved
29.5	A_BIT3_5	BOOL	FALSE	Reserved	Reserved
29.6	A_BIT3_6	BOOL	FALSE	Reserved	Reserved
29.7	A_BIT3_7	BOOL	FALSE	Reserved	Reserved
Transfer area for values to be read					
30.0	LATCH_LOAD [1]	DINT	L#0	Current load or latch value (read user)	Current measured value (read user)
34.0	ACT_CNTV [1]	DINT	L#0	Current count value (read user)	Current count value (read user)
Error numbers					
38.0	DA_ERR_W [1]	WORD	W#16#0	Data error word (read user)	Data error word (read user)
40.0	OT_ERR_B [1]	BYTE	B#16#0	Operation error byte (read user)	Operation error byte (read user)
Feedback interface					
41.0	E_BIT0_0	BOOL	FALSE	Reserved	Reserved
41.1	STS_TFB	BOOL	FALSE	Reserved / status test free	Reserved

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
41.2	DIAG	BOOL	FALSE	FM 350-1: Diagnostic event (read user) FM 450-1: Reserved	Diagnostic event (read user)
41.3	E_BIT0_3	BOOL	FALSE	Reserved	Reserved
41.4	DATA_ERR [1)]	BOOL	FALSE	Data error bit (read user)	Data error bit (read user)
41.5	E_BIT0_5	BOOL	FALSE	Reserved	Reserved
41.6	E_BIT0_6	BOOL	FALSE	Reserved	Reserved
41.7	PARA [1)]	BOOL	FALSE	Module parameters assigned (read user)	Module parameters assigned (read user)
42.0	E_BYTE_0	BYTE	B#16#0	Reserved	Reserved
43.0	STS_RUN [1)]	BOOL	FALSE	Status counter running (read user) This bit corresponds to bit 2^0 of the counter. 0 = LED CR is off 1 = LED CR is lit	Status counter running (read user) This bit corresponds to bit 2^0 of the counter. 0 = LED CR is off 1 = LED CR is lit
43.1	STS_DIR [1)]	BOOL	FALSE	Status count direction (read user)	Status count direction (read user)
43.2	STS_ZERO [1)]	BOOL	FALSE	Status zero-crossing (read user)	End of measurement (read user)
43.3	STS_OFLW [1)]	BOOL	FALSE	Status overflow (read user)	Status overflow (read user)
43.4	STS_UFLW [1)]	BOOL	FALSE	Status underflow (read user)	Status underflow (read user)
43.5	STS_SYNC [1)]	BOOL	FALSE	Status counter synchronized (read user)	-
43.6	STS_GATE [1)]	BOOL	FALSE	Status internal gate (read user)	Status internal gate (read user)
43.7	STS_SW_G [1)]	BOOL	FALSE	Status SW gate (read user)	Status SW gate (read user)
44.0	STS_SET [1)]	BOOL	FALSE	Status digital input SET (read user)	Status digital input DI Set (read user)
44.1	STS_LATCH [1)]	BOOL	FALSE	New latch value (only in isochronous mode) / Reserved	-
44.2	STS_STA [1)]	BOOL	FALSE	Status digital input START (read user)	Status digital input DI Start (read user)
44.3	STS_STP [1)]	BOOL	FALSE	Status digital input STOP (read user)	Status digital input DI Stop (read user)
44.4	STS_CMP1 [1)]	BOOL	FALSE	Status output comparison value 1 (read user)	Status output comparison value 1 (read user)
44.5	STS_CMP2 [1)]	BOOL	FALSE	Status output comparison value 2 (read user)	Status output comparison value 2 (read user)
44.6	STS_COMP1 [1)]	BOOL	FALSE	Saved status of comparator 1 / Reserved	-

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
44.7	STS_COMP2 [1)]	BOOL	FALSE	Saved status of comparator 2 / Reserved	-
45.0	E_BIT3_0	BOOL	FALSE	Reserved	Reserved
45.1	E_BIT3_1	BOOL	FALSE	Reserved	Reserved
45.2	E_BIT3_2	BOOL	FALSE	Reserved	Reserved
45.3	E_BIT3_3	BOOL	FALSE	Reserved	Reserved
45.4	E_BIT3_4	BOOL	FALSE	Reserved	Reserved
45.5	E_BIT3_5	BOOL	FALSE	Reserved	Reserved
45.6	E_BIT3_6	BOOL	FALSE	Reserved	Reserved
45.7	E_BIT3_7	BOOL	FALSE	Reserved	Reserved
Parameters for FM 450-1					
46.0	ACT_CMP1 [1)]	DINT	L#0	Reserved / Current comparison value 1 (read user)	Reserved
50.0	ACT_CMP2 [1)]	DINT	L#0	Reserved / Current comparison value 2 (read user)	Reserved
The following diagnostic information is entered by the DIAG_INF instruction					
54.0	MDL_DEFECT	BOOL	FALSE	Module fault	Module fault
54.1	INT_FAULT	BOOL	FALSE	Internal error	Internal error
54.2	EXT_FAULT	BOOL	FALSE	External error	External error
54.3	PNT_INFO	BOOL	FALSE	Channel errors (breakdown from DW 58)	Channel errors (breakdown from DW 58)
54.4	EXT_VOLTAGE	BOOL	FALSE	Auxiliary voltage fault	Auxiliary voltage fault
54.5	FLD_CNNCTR	BOOL	FALSE	Front connector	Front connector
54.6	NO_CONFIG	BOOL	FALSE	No parameter assignment	No parameter assignment
54.7	CONFIG_ERR	BOOL	FALSE	Faulty parameters	Faulty parameters
55.0	MDL_TYPE	BYTE	B#16#0	Module type	Module type
56.0	SUB_MDL_ERR	BOOL	FALSE	Wrong/missing interface module	Wrong/missing interface module
56.1	COMM_FAULT	BOOL	FALSE	Communication error	Communication error
56.2	MDL_STOP	BOOL	FALSE	RUN/STOP operating mode display	RUN/STOP operating mode display
56.3	WTCH_DOG_FAULT	BOOL	FALSE	Watchdog (FM)	Watchdog (FM)
56.4	INT_PS_FLT	BOOL	FALSE	Internal power supply fault	Internal power supply fault
56.5	PRIM_BATT_FLT	BOOL	FALSE	Battery monitoring	Battery monitoring
56.6	BCKUP_BATT_FLT	BOOL	FALSE	Bad buffer	Bad buffer
56.7	RESERVED_2	BOOL	FALSE	Reserved	Reserved
57.0	RACK_FLT	BOOL	FALSE	Rack fault	Rack fault
57.1	PROC_FLT	BOOL	FALSE	CPU fault	CPU fault
57.2	EPROM_FLT	BOOL	FALSE	EPROM fault	EPROM fault
57.3	RAM_FLT	BOOL	FALSE	RAM fault	RAM fault
57.4	ADU_FLT	BOOL	FALSE	ADU fault	ADU fault

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
57.5	FUSE_FLT	BOOL	FALSE	Fuse	Fuse
57.6	HW_INTR_FLT	BOOL	FALSE	Hardware interrupt lost	Hardware interrupt lost
57.7	RESERVED_3	BOOL	FALSE	Reserved	Reserved
58.0	CH_TYPE	BYTE	B#16#0	Channel type	Channel type
59.0	LGTH_DIA	BYTE	B#16#0	Length of the diagnostic data per channel	Length of the diagnostic data per channel
60.0	CH_NO	BYTE	B#16#0	Number of channels	Number of channels
61.0	GRP_ERR1	BOOL	FALSE	Group error channel 1	Group error channel 1
61.1	GRP_ERR2	BOOL	FALSE	Reserved / Group error channel 2	Not used with FM 350
61.2	D_BIT7_2	BOOL	FALSE	DS1 byte 7 bit 2	DS1 byte 7 bit 2
61.3	D_BIT7_3	BOOL	FALSE	DS1 byte 7 bit 3	DS1 byte 7 bit 3
61.4	D_BIT7_4	BOOL	FALSE	DS1 byte 7 bit 4	DS1 byte 7 bit 4
61.5	D_BIT7_5	BOOL	FALSE	DS1 byte 7 bit 5	DS1 byte 7 bit 5
61.6	D_BIT7_6	BOOL	FALSE	DS1 byte 7 bit 6	DS1 byte 7 bit 6
61.7	D_BIT7_7	BOOL	FALSE	DS1 byte 7 bit 7	DS1 byte 7 bit 7
62.0	CH1_SIGA	BOOL	FALSE	Channel 1, error signal A	Channel 1, error signal A
62.1	CH1_SIGB	BOOL	FALSE	Channel 1, error signal B	Channel 1, error signal B
62.2	CH1_SIGZ	BOOL	FALSE	Channel 1, error zero signal	Channel 1, error zero signal
62.3	CH1_BETW	BOOL	FALSE	Channel 1, error between channels	Channel 1, error between channels
62.4	CH1_5V2	BOOL	FALSE	Channel 1, error sensor power supply 5.2V	Channel 1, error sensor power supply 5.2V
62.5	D_BIT8_5	BOOL	FALSE	DS1 byte 8 bit 5	DS1 byte 8 bit 5
62.6	D_BIT8_6	BOOL	FALSE	DS1 byte 8 bit 6	DS1 byte 8 bit 6
62.7	D_BIT8_7	BOOL	FALSE	DS1 byte 8 bit 7	DS1 byte 8 bit 7
63.0	D_BYTE9	BYTE	B#16#0	DS1 byte 9	DS1 byte 9
64.0	CH2_SIGA	BOOL	FALSE	Reserved / Channel 2, error signal A	Reserved
64.1	CH2_SIGB	BOOL	FALSE	Reserved / Channel 2, error signal B	Reserved
64.2	CH2_SIGZ	BOOL	FALSE	Reserved / Channel 2, error zero signal	Reserved
64.3	CH2_BETW	BOOL	FALSE	Reserved / Channel 2, error between channels	Reserved
64.4	CH2_5V2	BOOL	FALSE	Reserved / Channel 2, error sensor power supply 5.2V	Reserved
64.5	D_BIT10_5	BOOL	FALSE	DS1 byte 10 bit 5	DS1 byte 10 bit 5
64.6	D_BIT10_6	BOOL	FALSE	DS1 byte 10 bit 6	DS1 byte 10 bit 6
64.7	D_BIT10_7	BOOL	FALSE	DS1 byte 10 bit 7	DS1 byte 10 bit 7
65.0	D_BYTE11	BYTE	B#16#0	DS1 byte 11	DS1 byte 11
66.0	D_BYTE12	BYTE	B#16#0	DS1 byte 12	DS1 byte 12

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
67.0	D_BYTE13	BYTE	B#16#0	DS1 byte 13	DS1 byte 13
68.0	D_BYTE14	BYTE	B#16#0	DS1 byte 14	DS1 byte 14
69.0	D_BYTE15	BYTE	B#16#0	DS1 byte 15	DS1 byte 15
[1] Tags in the DB that you can/must input or read when working with the FM x50-1					

Value transfer using CNT_CTL1 (S7-300, S7-400)

Value transfer

Transfer the operating mode-specific values by setting these instruction parameters:

Mode	Instruction parameters
Counting	L_DIRECT, L_PREPAR, T_CMP_V1, T_CMP_V2, C_DOPARA
Measuring	L_PREPAR, T_CMP_V1, T_CMP_V2, C_DOPARA

You must first place the values to be transferred in the DB beforehand.

See also:

- DB parameters for value transfer (count modes) (Page 5882)
- DB parameters for value transfer (count modes) (Page 5884)

You can transfer several values at the same time:

In operating mode ...	... You can transfer at the same time	
Counting	Load value	(DB parameter LOAD_VAL)
	Comparison value 1	(DB parameter CMP_V1)
	Comparison value 2	(DB parameter CMP_V2)
Measuring	Low limit	(DB parameter LOAD_VAL)
	High limit	(DB parameter CMP_V1)
	Updating time	(DB parameter CMP_V2)

If an incorrect value is transferred, you have to acknowledge this operator error with OT_ERR_A before FM 350-1 can accept further values. You should then correct the value rejected with the operator error and transfer it again.

Note

If you use the L_DIRECT, L_PREPAR, T_CMP_V1 or T_CMP_V2 instruction parameters to load the LOAD_VAL, CMP_V1 or CMP_V2 values, you cannot use the C_DOPARA instruction parameter at the same time to reconfigure parameters.

This would cause an OT_ERR operating error requiring acknowledgment with OT_ERR_A.

Time required to transfer values

Refer to the table below for the time required to transfer values:

Using FM 350-1		Time requirement
Centralized		At least 4 OB 1 cycles
Distributed (non-isochrone mode)		At least 5 PROFIBUS DP cycles
Distributed (isochrone mode)		
• At transmission of only one value		• 5 PROFIBUS DP cycles
• At simultaneous initiation of the transmission of several values		
	• For the 1st value: • For the 2nd value: • For the 3rd value:	• 5 PROFIBUS DP cycles after initiation • 6 PROFIBUS DP cycles after initiation • 7 PROFIBUS DP cycles after initiation

DB parameters for transferring values (counting modes) (S7-300, S7-400)

Preparation

Before calling the CNT_CTRL (Page 5870), CNT_CTL1 (Page 5872) or CNT_CTL2 (Page 5874) instruction, you need to write the following valid data to the data block:

- Module address (DB address 6.0)
- Channel start address (DB address 8.0)
- Length of the user data interface (DB address 12.0)

These data should be transferred to the DB in OB 100 at every startup.

DB parameters for transferring values (counting modes)

The following table shows the DB area to which you transfer the LOAD_VAL, CMP_V1 and CMP_V2 parameters.

Parameter LOAD_VAL (bytes 14 to 17) has two meanings:

- If you set the L_DIRECT or L_PREPAR instruction parameter, LOAD_VAL is interpreted as a load value.
- If you set instruction parameter C_DOPARA, you can use byte 14 to define the characteristics of outputs DO0 and DO1. Bytes 15 and 16 are interpreted as hysteresis and pulse duration.

You can set only one of the L_DIRECT, L_PREPAR and C_DOPARA instruction parameters at any time.

DB address	Parame-ters	Data type	Meaning				
14.0	LOAD_VAL	DINT	Load value; direct and preparatory loading with instruction parameter L_DIRECT Load value; preparatory loading with instruction parameter L_PREPAR				
			Characteristics of the DO0 and DO1 outputs, the hysteresis and the pulse duration; defined with the C_DOPARA instruction parameter:				
			Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	**Reaction of output DO0**
			x	0	0	0	Inactive
			x	0	0	1	Active from the comparison value to high limit
			x	0	1	0	Active from the comparison value to low limit
			x	0	1	1	Active on reaching the comparison value for pulse duration (up/down)
			x	1	0	0	Active on reaching the comparison value for pulse duration (up)
			x	1	0	1	Active on reaching the comparison value for pulse duration (down)
			Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Characteristics of output DO1**
			x	0	0	0	Inactive
			x	0	0	1	Active from the comparison value to high limit
			x	0	1	0	Active from the comparison value to low limit
			x	0	1	1	Active on reaching the comparison value for pulse duration (up/down)
			x	1	0	0	Active on reaching the comparison value of the up count pulse width
			x	1	0	1	Active on reaching the comparison value of the down count pulse width
			x	1	1	0	Switch at comparison values
15.0			Hysteresis (value range 0 to 255)				
16.0			Pulse duration (value range 0 to 250)				
17.0			Reserve = 0				
18.0	CMP_V1	DINT	Comparison value 1; load with the T_CMP_V1 instruction parameter				
22.0	CMP_V2	DINT	Comparison value 2; load with the T_CMP_V2 instruction parameter				
x = irrelevant							

DB parameters for transferring values (measuring modes) (S7-300, S7-400)

Preparation

Before calling the CNT_CTRL (Page 5870), CNT_CTL1 (Page 5872) or CNT_CTL2 (Page 5874) instruction, you need to write the following valid data to the data block:

- Module address (DB address 6.0)
- Channel start address (DB address 8.0)
- Length of the user data interface (DB address 12.0)

These data should be transferred to the DB in OB 100 at every startup.

DB parameters for transferring values (measuring modes)

The following table shows the DB area to which you transfer the LOAD_VAL, CMP_V1 and CMP_V2 parameters.

Parameter LOAD_VAL (bytes 14 to 17) has two meanings:

- If you set the L_PREPAR instruction parameter, LOAD_VAL is interpreted as a low limit.
- If you set the C_DOPARA instruction parameter, you can use byte 14 to specify the characteristics of output DO0.

You can set only one of the L_PREPAR and C_DOPARA instruction parameters at any time.

DB address	Parameters	Data type	Meaning			
14.0	LOAD_VAL	DINT	Low limit; load with the L_PREPAR instruction parameter			
			Characteristics of DO0; specify in the C_DOPARA instruction parameter			
			Bits 2 to 7	**Bit 1**	**Bit 0**	**Reaction of output DO0**
			Irrelevant	0	0	No comparison
			Irrelevant	0	1	Active outside the limits
			Irrelevant	1	0	Active upon violation of low limit
			Irrelevant	1	1	Active upon violation of high limit
15.0			Reserve = 0			
16.0			Reserve = 0			
17.0			Reserve = 0			
18.0	CMP_V1	DINT	High limit; load with the T_CMP_V1 instruction parameter			
22.0	CMP_V2	DINT	Update time; load with the T_CMP_V2 instruction parameter			

FM 350-2 Counting (S7-300, S7-400)

CNT2WRPN (S7-300, S7-400)

Description

You can use the CNT2WRPN instruction to load the counters and comparators of the counter module using write jobs. To do this, you need to call the CNT2WRPN instruction once for each module.

You should integrate the CNT2WRPN instruction in the user program only if it is necessary to reload counters and comparators in runtime.

Operating principle

The CNT2WRPN instruction executes the write command (Page 5896) (JOB_WR) from the counter DB (Page 5887), transfers the corresponding data from the counter DB and indicates thestatus of the write command (Page 5897). The write command is handled internally with the WRREC instruction and the return code is passed to the WRREC instruction.

Call

The CNT2WRPN instruction can be called in the cycle or optionally in a time-controlled program. The call is not allowed in an event-driven interrupt program.

Before executing write jobs, you must assign the appropriate values in the data area associated with the write command. The last write command must be completed, which means that JOB_WR.NO (data byte DBB0) is deleted from the counter DB (Page 5887).

Parameters

Parameters	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	WORD	Number of the counter data block	Enters this	Queries this
RET_VAL	OUTPUT	INT	Return code of the instruction WRREC	Queries this	Enters this

CNT2RDPN (S7-300, S7-400)

Description

You can use the CNT2RDPN instruction to read the counter values and measured values of the counter module using read jobs. Call the CNT2RDPN instruction cyclically once for each module to do this.

Do not integrate the CNT2RDPN instruction in your user program unless you are not processing read jobs.

Operating principle

The instruction executes the read command (Page 5898) (JOB_RD) from the counter DB (Page 5887), transfers the corresponding data to the counter DB and indicates the status of the read command (Page 5898). The read command is handled internally with the RDREC instruction and the return code is passed to the RDREC instruction.

Call

The CNT2RDPN instruction can be called in the cycle or optionally in a time-controlled program. The call is not allowed in an event-driven interrupt program.

The last read command must be completed, which means that JOB_RD.NO (data byte DBB2) is deleted from the counter DB (Page 5887).

Parameters

Parameters	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	WORD	Number of the counter data block	Enters this	Queries this
RET_VAL	OUTPUT	INT	Return code of the instruction RDREC	Queries this	Enters this

CNT2_CTR (S7-300, S7-400)

Description

The CNT2_CTR instruction is used to control the digital outputs (enable and disable) and software gates of the counter module. In addition, you receive the feedback signals of the module.

Operating principle

The CNT2_CTR instruction

- initializes the counter DB (Page 5887).
- Transfers the counter signals from the counter DB (CONTROL_SIGNALS (Page 5894) structure) module.
- Reads the feedback signals The instruction in the counter DB saves the read values to the CHECKBACK_SIGNALS (Page 5895) structure.

Call

You can use the CNT2_CTR instruction cyclically (in OB1 or in the cyclic interrupts) for each module. Calling in an interrupt program is not permitted.

Before calling the CNT2_CTR instruction, enter the current control signals in the CONTROL_SIGNALS structure in the counter DB (Page 5887). When the call of the CNT2_CTR instruction is completed, the feedback signals in the CHECKBACK_SIGNALS structure are updated in the counter DB for further processing.

Parameters

Parame-ters	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	WORD	Number of the counter data block	Enters this	Queries this

DIAG_RD (S7-300, S7-400)

Description

The DIAG_RD instruction can be used to load the data of diagnostic interrupts in the counter DB.

Operating principle

The DIAG_RD instruction reads 16 bytes of diagnostic data from the counter module and enters these data in thecounter DB (Page 5887) in the data area DIAGNOSTIC_IN_INFO. These data contain the diagnostic status of the entire module (across all counters). Diagnostic data is read internally at the call of the RDSYSST instruction. The return code of this instruction is passed as RET_VAL.

Call

The DIAG_RD instruction can be called only in the interrupt program.

Parameters

Parameters	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	WORD	Number of the counter data block	Enters this	Queries this
RET_VAL	OUTPUT	INT	Return code of the instruction RDSYSST	Queries this	Enters this

Additional references for FM 350-2 Counter V2 (S7-300, S7-400)

Counter data block (S7-300, S7-400)

Purpose

The counter DB is the data interface between the user program and FM 350-2. It contains and applies all data necessary to control and operate the module.

Before the parameter assignment of the module, you must provide the following valid data to the counter DB.

- Module address (address 12.0)
- Channel start address (address 14.0)
- DS-OFFSET (address 18.0), set to 0

Contents

The table below shows the contents of the counter DB.

Note

You may not change any data not listed in this table.

Address	Tag	Data type	Initial value	Comment
0.0	NO	BYTE	B#16#0	Number
1.0	BUSY	BOOL	FALSE	TRUE: Write command in progress FALSE: Write command not in progress
1.1	DONE	BOOL	FALSE	TRUE: Write command completed FALSE: Write command not completed
1.2	IMPOSS	BOOL	FALSE	TRUE: Write command not possible FALSE: Write command possible
1.3	UNKNOWN	BOOL	FALSE	TRUE: Write command unknown FALSE: Write command known
2.0	NO	BYTE	B#16#0	Number
3.0	BUSY	BOOL	FALSE	TRUE: Read command in progress FALSE: Read command not in progress
3.1	DONE	BOOL	FALSE	TRUE: Read command completed FALSE: Read command not completed
3.2	IMPOSS	BOOL	FALSE	TRUE: Read command not possible FALSE: Read command possible
3.3	UNKNOWN	BOOL	FALSE	TRUE: Read command unknown FALSE: Read command known
4.0	RESERV_0	ARRAY [1..3] OF WORD	W#16#0	Reserved
10.0	RESERV_1	WORD	W#16#0	Reserved
12.0	MOD_ADR	WORD	W#16#0	Module address
14.0	CH_ADR	DWORD	DW#16#0	Channel address
18.0	DS_OFFS	BYTE	B#16#0	Offset data block
19.0	RESERV_2	BYTE	B#16#0	Reserved
20.0	BIT0_0	BOOL	FALSE	Reserved
20.1	BIT0_1	BOOL	FALSE	Reserved

Address	Tag	Data type	Initial value	Comment
20.2	BIT0_2	BOOL	FALSE	Reserved
20.3	BIT0_3	BOOL	FALSE	Reserved
20.4	BIT0_4	BOOL	FALSE	Reserved
20.5	BIT0_5	BOOL	FALSE	Reserved
20.6	BIT0_6	BOOL	FALSE	Reserved
20.7	BIT0_7	BOOL	FALSE	Reserved
21.0	CTRL_DQ0	BOOL	FALSE	TRUE: Output 0 enabled FALSE: Output 0 not enabled
21.1	CTRL_DQ1	BOOL	FALSE	TRUE: Output 1 enabled FALSE: Output 1 not enabled
21.2	CTRL_DQ2	BOOL	FALSE	TRUE: Output 2 enabled FALSE: Output 2 not enabled
21.3	CTRL_DQ3	BOOL	FALSE	TRUE: Output 3 enabled FALSE: Output 3 not enabled
21.4	CTRL_DQ4	BOOL	FALSE	TRUE: Output 4 enabled FALSE: Output 4 not enabled
21.5	CTRL_DQ5	BOOL	FALSE	TRUE: Output 5 enabled FALSE: Output 5 not enabled
21.6	CTRL_DQ6	BOOL	FALSE	TRUE: Output 6 enabled FALSE: Output 6 not enabled
21.7	CTRL_DQ7	BOOL	FALSE	TRUE: Output 7 enabled FALSE: Output 7 not enabled
22.0	SET_DQ0	BOOL	FALSE	TRUE: Output 0 set FALSE: Output 0 not set
22.1	SET_DQ1	BOOL	FALSE	TRUE: Output 1 set FALSE: Output 1 not set
22.2	SET_DQ2	BOOL	FALSE	TRUE: Output 2 set FALSE: Output 2 not set
22.3	SET_DQ3	BOOL	FALSE	TRUE: Output 3 set FALSE: Output 3 not set
22.4	SET_DQ4	BOOL	FALSE	TRUE: Output 4 set FALSE: Output 4 not set
22.5	SET_DQ5	BOOL	FALSE	TRUE: Output 5 set FALSE: Output 5 not set
22.6	SET_DQ6	BOOL	FALSE	TRUE: Output 6 set FALSE: Output 6 not set
22.7	SET_DQ7	BOOL	FALSE	TRUE: Output 7 set FALSE: Output 7 not set
23.0	SW_GATE0	BOOL	FALSE	TRUE: SW gate counter 0 open FALSE: SW gate counter 0 closed
23.1	SW_GATE1	BOOL	FALSE	TRUE: SW gate counter 1 open FALSE: SW gate counter 1 closed

Address	Tag	Data type	Initial value	Comment
23.2	SW_GATE2	BOOL	FALSE	TRUE: SW gate counter 2 open FALSE: SW gate counter 2 closed
23.3	SW_GATE3	BOOL	FALSE	TRUE: SW gate counter 3 open FALSE: SW gate counter 3 closed
23.4	SW_GATE4	BOOL	FALSE	TRUE: SW gate counter 4 open FALSE: SW gate counter 4 closed
23.5	SW_GATE5	BOOL	FALSE	TRUE: SW gate counter 5 open FALSE: SW gate counter 5 closed
23.6	SW_GATE6	BOOL	FALSE	TRUE: SW gate counter 6 open FALSE: SW gate counter 6 closed
23.7	SW_GATE7	BOOL	FALSE	TRUE: SW gate counter 7 open FALSE: SW gate counter 7 closed
24.0	CTRL_DWORD1	DWORD	DW#16#0	Reserved
28.0	CTRL_DWORD2	DWORD	DW#16#0	Reserved
32.0	CTRL_DWORD3	DWORD	DW#16#0	Reserved
36.0	BIT0_0	BOOL	FALSE	Reserved
36.1	STS_TFB	BOOL	FALSE	TRUE: PG mode active FALSE: No PG mode
36.2	BIT0_2	BOOL	FALSE	Reserved
36.3	BIT0_3	BOOL	FALSE	Reserved
36.4	DATA_ERR	BOOL	FALSE	Data error
36.5	BIT0_5	BOOL	FALSE	Reserved
36.6	BIT0_6	BOOL	FALSE	Reserved
36.7	PARA	BOOL	FALSE	TRUE: Module parameters assigned FALSE: Module parameters not assigned
37.0	STS_CMP0	BOOL	FALSE	TRUE: Comparator 0 addressed FALSE: Comparator 0 not addressed
37.1	STS_CMP1	BOOL	FALSE	TRUE: Comparator 1 addressed FALSE: Comparator 1 not addressed
37.2	STS_CMP2	BOOL	FALSE	TRUE: Comparator 2 addressed FALSE: Comparator 2 not addressed
37.3	STS_CMP3	BOOL	FALSE	TRUE: Comparator 3 addressed FALSE: Comparator 3 not addressed
37.4	STS_CMP4	BOOL	FALSE	TRUE: Comparator 4 addressed FALSE: Comparator 4 not addressed
37.5	STS_CMP5	BOOL	FALSE	TRUE: Comparator 5 addressed FALSE: Comparator 5 not addressed
37.6	STS_CMP6	BOOL	FALSE	TRUE: Comparator 6 addressed FALSE: Comparator 6 not addressed
37.7	STS_CMP7	BOOL	FALSE	TRUE: Comparator 7 addressed FALSE: Comparator 7 not addressed
38.0	STS_UFLW0	BOOL	FALSE	TRUE: Underflow counter 0 FALSE: No underflow counter 0

Address	Tag	Data type	Initial value	Comment
38.1	STS_UFLW1	BOOL	FALSE	TRUE: Underflow counter 1 FALSE: No underflow counter 1
38.2	STS_UFLW2	BOOL	FALSE	TRUE: Underflow counter 2 FALSE: No underflow counter 2
38.3	STS_UFLW3	BOOL	FALSE	TRUE: Underflow counter 3 FALSE: No underflow counter 3
38.4	STS_UFLW4	BOOL	FALSE	TRUE: Underflow counter 4 FALSE: No underflow counter 4
38.5	STS_UFLW5	BOOL	FALSE	TRUE: Underflow counter 5 FALSE: No underflow counter 5
38.6	STS_UFLW6	BOOL	FALSE	TRUE: Underflow counter 6 FALSE: No underflow counter 6
38.7	STS_UFLW7	BOOL	FALSE	TRUE: Underflow counter 7 FALSE: No underflow counter 7
39.0	STS_OFLW0	BOOL	FALSE	TRUE: Overflow counter 0 FALSE: No overflow counter 0
39.1	STS_OFLW1	BOOL	FALSE	TRUE: Overflow counter 1 FALSE: No overflow counter 1
39.2	STS_OFLW2	BOOL	FALSE	TRUE: Overflow counter 2 FALSE: No overflow counter 2
39.3	STS_OFLW3	BOOL	FALSE	TRUE: Overflow counter 3 FALSE: No overflow counter 3
39.4	STS_OFLW4	BOOL	FALSE	TRUE: Overflow counter 4 FALSE: No overflow counter 4
39.5	STS_OFLW5	BOOL	FALSE	TRUE: Overflow counter 5 FALSE: No overflow counter 5
39.6	STS_OFLW6	BOOL	FALSE	TRUE: Overflow counter 6 FALSE: No overflow counter 6
39.7	STS_OFLW7	BOOL	FALSE	TRUE: Overflow counter 7 FALSE: No overflow counter 7
40.0	STS_DIR0	BOOL	FALSE	TRUE: Count direction counter 0 backwards FALSE: Count direction counter 0 forwards
40.1	STS_DIR1	BOOL	FALSE	TRUE: Count direction counter 1 backwards FALSE: Count direction counter 1 forwards
40.2	STS_DIR2	BOOL	FALSE	TRUE: Count direction counter 2 backwards FALSE: Count direction counter 2 forwards
40.3	STS_DIR3	BOOL	FALSE	TRUE: Count direction counter 3 backwards FALSE: Count direction counter 3 forwards
40.4	STS_DIR4	BOOL	FALSE	TRUE: Count direction counter 4 backwards FALSE: Count direction counter 4 forwards
40.5	STS_DIR5	BOOL	FALSE	TRUE: Count direction counter 5 backwards FALSE: Count direction counter 5 forwards

Address	Tag	Data type	Initial value	Comment
40.6	STS_DIR6	BOOL	FALSE	TRUE: Count direction counter 6 backwards FALSE: Count direction counter 6 forwards
40.7	STS_DIR7	BOOL	FALSE	TRUE: Count direction counter 7 backwards FALSE: Count direction counter 7 forwards
41.0	STS_DI0	BOOL	FALSE	TRUE: Digital input 0 set FALSE: Digital input 0 not set
41.1	STS_DI1	BOOL	FALSE	TRUE: Digital input 1 set FALSE: Digital input 1 not set
41.2	STS_DI2	BOOL	FALSE	TRUE: Digital input 2 set FALSE: Digital input 2 not set
41.3	STS_DI3	BOOL	FALSE	TRUE: Digital input 3 set FALSE: Digital input 3 not set
41.4	STS_DI4	BOOL	FALSE	TRUE: Digital input 4 set FALSE: Digital input 4 not set
41.5	STS_DI5	BOOL	FALSE	TRUE: Digital input 5 set FALSE: Digital input 5 not set
41.6	STS_DI6	BOOL	FALSE	TRUE: Digital input 6 set FALSE: Digital input 6 not set
41.7	STS_DI7	BOOL	FALSE	TRUE: Digital input 7 set FALSE: Digital input 7 not set
42.0	STS_DQ0	BOOL	FALSE	TRUE: Digital output 0 set FALSE: Digital output 0 not set
42.1	STS_DQ1	BOOL	FALSE	TRUE: Digital output 1 set FALSE: Digital output 1 not set
42.2	STS_DQ2	BOOL	FALSE	TRUE: Digital output 2 set FALSE: Digital output 2 not set
42.3	STS_DQ3	BOOL	FALSE	TRUE: Digital output 3 set FALSE: Digital output 3 not set
42.4	STS_DQ4	BOOL	FALSE	TRUE: Digital output 4 set FALSE: Digital output 4 not set
42.5	STS_DQ5	BOOL	FALSE	TRUE: Digital output 5 set FALSE: Digital output 5 not set
42.6	STS_DQ6	BOOL	FALSE	TRUE: Digital output 6 set FALSE: Digital output 6 not set
42.7	STS_DQ7	BOOL	FALSE	TRUE: Digital output 7 set FALSE: Digital output 7 not set
43.0	STS_GATE0	BOOL	FALSE	TRUE: Internal gate counter 0 open FALSE: Internal gate counter 0 closed
43.1	STS_GATE1	BOOL	FALSE	TRUE: Internal gate counter 1 open FALSE: Internal gate counter 1 closed
43.2	STS_GATE2	BOOL	FALSE	TRUE: Internal gate counter 2 open FALSE: Internal gate counter 2 closed

Address	Tag	Data type	Initial value	Comment
43.3	STS_GATE3	BOOL	FALSE	TRUE: Internal gate counter 3 open FALSE: Internal gate counter 3 closed
43.4	STS_GATE4	BOOL	FALSE	TRUE: Internal gate counter 4 open FALSE: Internal gate counter 4 closed
43.5	STS_GATE5	BOOL	FALSE	TRUE: Internal gate counter 5 open FALSE: Internal gate counter 5 closed
43.6	STS_GATE6	BOOL	FALSE	TRUE: Internal gate counter 6 open FALSE: Internal gate counter 6 closed
43.7	STS_GATE7	BOOL	FALSE	TRUE: Internal gate counter 7 open FALSE: Internal gate counter 7 closed
44	USER_STAT_WORD0	WORD	W#16#0	Count/measured value, depending on parameter assignment
46	USER_STAT_WORD1	WORD	W#16#0	Count/measured value, depending on parameter assignment
48	USER_STAT_WORD2	WORD	W#16#0	Count/measured value, depending on parameter assignment
50	USER_STAT_WORD3	WORD	W#16#0	Count/measured value, depending on parameter assignment
52	LOAD-VAL0	DINT	L#0	Load counter 0 directly
56	LOAD-VAL1	DINT	L#0	Load counter 1 directly
60	LOAD-VAL2	DINT	L#0	Load counter 2 directly
64	LOAD-VAL3	DINT	L#0	Load counter 3 directly
68	LOAD-VAL4	DINT	L#0	Load counter 4 directly
72	LOAD-VAL5	DINT	L#0	Load counter 5 directly
76	LOAD-VAL6	DINT	L#0	Load counter 6 directly
80	LOAD-VAL7	DINT	L#0	Load counter 7 directly
84	LOAD-PREPARE-VAL0	DINT	L#0	Load counter 0 in preparation
88	LOAD-PREPARE-VAL1	DINT	L#0	Load counter 1 in preparation
92	LOAD-PREPARE-VAL2	DINT	L#0	Load counter 2 in preparation
96	LOAD-PREPARE-VAL3	DINT	L#0	Load counter 3 in preparation
100	LOAD-PREPARE-VAL4	DINT	L#0	Load counter 4 in preparation
104	LOAD-PREPARE-VAL5	DINT	L#0	Load counter 5 in preparation
108	LOAD-PREPARE-VAL6	DINT	L#0	Load counter 6 in preparation
112	LOAD-PREPARE-VAL7	DINT	L#0	Load counter 7 in preparation
116	CMP-VAL0	DINT	L#0	Load comparator 0
120	CMP-VAL1	DINT	L#0	Load comparator 1
124	CMP-VAL2	DINT	L#0	Load comparator 2
128	CMP-VAL3	DINT	L#0	Load comparator 3
132	CMP-VAL4	DINT	L#0	Load comparator 4
136	CMP-VAL5	DINT	L#0	Load comparator 5
140	CMP-VAL6	DINT	L#0	Load comparator 6
144	CMP-VAL7	DINT	L#0	Load comparator 7
148	ACT_CNTV0	DINT	L#0	Current count 0
152	ACT_MSRV0	DINT	L#0	Measurement result 0

Address	Tag	Data type	Initial value	Comment
156	ACT_CNTV1	DINT	L#0	Current count 1
160	ACT_MSRV1	DINT	L#0	Measurement result 1
164	ACT_CNTV2	DINT	L#0	Current count 2
168	ACT_MSRV2	DINT	L#0	Measurement result 2
172	ACT_CNTV3	DINT	L#0	Current count 3
176	ACT_MSRV3	DINT	L#0	Measurement result 3
180	ACT_CNTV4	DINT	L#0	Current count 4
184	ACT_MSRV4	DINT	L#0	Measurement result 4
188	ACT_CNTV5	DINT	L#0	Current count 5
192	ACT_MSRV5	DINT	L#0	Measurement result 5
196	ACT_CNTV6	DINT	L#0	Current count 6
200	ACT_MSRV6	DINT	L#0	Measurement result 6
204	ACT_CNTV7	DINT	L#0	Current count 7
208	ACT_MSRV7	DINT	L#0	Measurement result 7
212.0	BYTE0	BYTE	B#16#0	Reserved
213.0	BYTE1	BYTE	B#16#0	Reserved
214.0	BYTE2	BYTE	B#16#0	Reserved
215.0	BYTE3	BYTE	B#16#0	Reserved
216.0	BYTE4	BYTE	B#16#0	Channel type
217.0	BYTE5	BYTE	B#16#0	Length channel info
218.0	BYTE6	BYTE	B#16#0	Number of channels
219.0	BYTE7	BYTE	B#16#0	Channel error vector
220.0	BYTE8	BYTE	B#16#0	Error counter 0
221.0	BYTE9	BYTE	B#16#0	Error counter 1
222.0	BYTE10	BYTE	B#16#0	Error counter 2
223.0	BYTE11	BYTE	B#16#0	Error counter 3
224.0	BYTE12	BYTE	B#16#0	Error counter 4
225.0	BYTE13	BYTE	B#16#0	Error counter 5
226.0	BYTE14	BYTE	B#16#0	Error counter 6
227.0	BYTE15	BYTE	B#16#0	Error counter 7

CONTROL_SIGNALS (S7-300, S7-400)

CONTROL_SIGNALS

Address	Parameters	Data type	Initial value	Description
21.0 … 7	CTRL_DQ0 … 7	BOOL	FALSE	Enable for digital outputs 0 ... 7
22.0 … 7	SET_DQ0 … 7	BOOL	FALSE	Trigger for digital outputs 0 ... 7
23.0 … 7	SW_GATE0 … 7	BOOL	FALSE	Software gate counters 0 ... 7

CHECKBACK_SIGNALS (S7-300, S7-400)

CHECKBACK_SIGNALS

Address	Parameters	Data type	Initial value	Description
36.1	STS_TFB	BOOL	FALSE	TRUE: Programming device (PG) mode is active, control by CPU is not possible FALSE: Programming device (PG) mode deselected
36.4	DATA_ERR	BOOL	FALSE	TRUE: Data error occurred FALSE: No data error
36.7	PARA	BOOL	FALSE	TRUE: Counter module parameters are assigned, all other CHECKBACK_SIGNALS are valid FALSE: Counter module parameters not assigned
37.0 … 7	STS_CMP0 … 7	BOOL	FALSE	Status comparators 0 ... 7 You need to evaluate the status following the call of the CNT2_CTR instruction because the status is subsequently reset.
38.0 … 7	STS_UFLW0 … 7	BOOL	FALSE	Status of low limit violated counters 0 ... 7 You need to evaluate the status following the call of the CNT2_CTR instruction because the status is subsequently reset.
39.0 … 7	STS_OFLW0 … 7	BOOL	FALSE	Status of high limit violated counters 0 ... 7 You need to evaluate the status following the call of the CNT2_CTR instruction because the status is subsequently reset.
40.0 … 7	STS_DIR0 … 7	BOOL	FALSE	Count direction status of counters 0 ... 7 You need to evaluate the status following the call of the CNT2_CTR instruction because the status is subsequently reset.
41.0 … 7	STS_DI0 … 7	BOOL	FALSE	Status of digital inputs 0 ... 7
42.0 … 7	STS_DQ0 … 7	BOOL	FALSE	Status of digital outputs 0 ... 7
43.0 … 7	STS_GATE0 … 7	BOOL	FALSE	Internal gate status of counters 0 ... 7
44	USER_STAT_DWORD0	WORD	W#16#0	Count/measured value, depending on the parameter assignment
46	USER_STAT_DWORD1	WORD	W#16#0	Count/measured value, depending on the parameter assignment
48	USER_STAT_DWORD2	WORD	W#16#0	Count/measured value, depending on the parameter assignment
50	USER_STAT_DWORD3	WORD	W#16#0	Count/measured value, depending on the parameter assignment

Write job (S7-300, S7-400)

Write command

Initiate a new write command by entering it in JOB_WR.NO. The following write jobs are permissible:

JOB_WR.NO (DBB0)	Entry in CNT2_CHANTYPE	Address in the counter DB	Meaning for the count mode	Meaning for the measuring mode
0	None	None	No write command / last write command finished	
10	LOAD_VAL0	52	Load counter 0 directly	Load low limit 0
11	LOAD_VAL1	56	Load counter 1 directly	Load low limit 1
12	LOAD_VAL2	60	Load counter 2 directly	Load low limit 2
13	LOAD_VAL3	64	Load counter 3 directly	Load low limit 3
14	LOAD_VAL4	68	Load counter 4 directly	Load low limit 4
15	LOAD_VAL5	72	Load counter 5 directly	Load low limit 5
16	LOAD_VAL6	76	Load counter 6 directly	Load low limit 6
17	LOAD_VAL7	80	Load counter 7 directly	Load low limit 7
20	LOAD_PREPARE_VAL0	84	Load counter 0 in preparation	Load high limit 0
21	LOAD_PREPARE_VAL1	88	Load counter 1 in preparation	Load high limit 1
22	LOAD_PREPARE_VAL2	92	Load counter 2 in preparation	Load high limit 2
23	LOAD_PREPARE_VAL3	96	Load counter 3 in preparation	Load high limit 3
24	LOAD_PREPARE_VAL4	100	Load counter 4 in preparation	Load high limit 4
25	LOAD_PREPARE_VAL5	104	Load counter 5 in preparation	Load high limit 5
26	LOAD_PREPARE_VAL6	108	Load counter 6 in preparation	Load high limit 6
27	LOAD_PREPARE_VAL7	112	Load counter 7 in preparation	Load high limit 7
30	CMP_VAL0	116	Load comparator 0	
31	CMP_VAL1	120	Load comparator 1	
32	CMP_VAL2	124	Load comparator 2	
33	CMP_VAL3	128	Load comparator 3	
34	CMP_VAL4	132	Load comparator 4	
35	CMP_VAL5	136	Load comparator 5	
36	CMP_VAL6	140	Load comparator 6	
37	CMP_VAL7	144	Load comparator 7	
40	LOAD_VAL0 to LOAD_VAL3	52 to 67	Load counters 0 to 3 directly	Load low limit 0 to 3
41	LOAD_VAL4 to LOAD_VAL7	68 to 83	Load counters 4 to 7 directly	Load low limit 4 to 7
42	LOAD_VAL0 to LOAD_VAL7	52 to 83	Load counters 0 to 7 directly	Load low limit 0 to 7
50	LOAD_PREPARE_VAL0 to LOAD_PREPARE_VAL3	84 to 99	Load counters 0 to 3 in preparation	Load high limit 0 to 3
51	LOAD_PREPARE_VAL4 to LOAD_PREPARE_VAL7	100 to 111	Load counters 4 to 7 in preparation	Load high limit 4 to 7
52	LOAD_PREPARE_VAL0 to LOAD_PREPARE_VAL7	84 to 111	Load counters 0 to 7 in preparation	Load high limit 0 to 7
60	CMP_VAL0 to CMP_VAL3	116 to 131	Load comparators 0 to 3	

JOB_WR.NO (DBB0)	Entry in CNT2_CHANTYPE	Address in the counter DB	Meaning for the count mode	Meaning for the measuring mode
61	CMP_VAL4 to CMP_VAL7	132 to 147	Load comparators 4 to 7	
62	CMP_VAL0 to CMP_VAL7	116 to 147	Load comparators 0 to 7	

Status of the write job (S7-300, S7-400)

Status of a write job

The status of a write job is indicated in the counter DB (DBB1 data byte).

Bit in JOB_WR (DBX1.)	Meaning
.BUSY, 0	= 1: Write job active. The CNT2WRPN instruction sets this bit as soon as you execute a write job (JOB_WR.NO > 0 and JOB_WR.IMPOSS = 0). The CNT2WRPN instruction deletes this bit as soon as the write job is completed (JOB_WR.NO = 0).
.DONE, 1	= 1: Write job finished. The CNT2WRPN instruction sets this bit as soon as a write job is completed (regardless of errors). The CNT2WRPN instruction deletes the bit when a new write job begins. You can also delete this bit in the user program.
.IMPOSS, 2	= 1: The write job cannot be processed (counter module parameters are not assigned, the call is active or programming device (PG) mode is active). You can leave the write job (JOB_WR) pending or delete it. The CNT2WRPN instruction deletes the bit if the conditions indicated above are fulfilled.
.UNKOWN, 3	= 1: Write job unknown. Your specified write job (JOB_WR) is not in the valid range (see the error message). The CNT2WRPN instruction deletes this bit as soon as a valid number is entered in JOB_WR. The unknown number is retained until this happens.

Error events are indicated in the binary result bit (BR = 0). Possible errors are:

- Unknown write job (see JOB_WR.UNKOWN).
- Error in data transmission with the WRREC instruction. Error information is available at the RET_VAL output parameter.
- The module checks for data errors in the transmitted data transferred and interprets the data. If a data error occurs, the CHECKBACK_SIGNALS.DATA_ERR bit is set to 1 in the counter DB.

Read job (S7-300, S7-400)

Read job

Initiate a new read job by entering it in JOB_RD.NO. The following read jobs are permissible:

JOB_RD.NO (DBB2)	Entry in CNT2_CHAN-TYPE	Address in the counter DB	Meaning
0	None	None	No read job / last read job finished
100	ACT_CNTV0 ACT_MSRV0 ACT_CNTV1 ACT_MSRV1 ACT_CNTV2 ACT_MSRV2 ACT_CNTV3 ACT_MSRV3	148	Current counter value 0 to 3 and measurement result 0 to 3
101	ACT_CNTV4 ACT_MSRV4 ACT_CNTV5 ACT_MSRV5 ACT_CNTV6 ACT_MSRV6 ACT_CNTV7 ACT_MSRV7	180	Current counter value 4 to 7 and measurement result 4 to 7

Status of the write job (S7-300, S7-400)

Status of a read job

The status of a read job is indicated in the counter DB (DBB3 data byte)

Bit in JOB_RD (DBX3.)	Description
.BUSY, 0	= 1: Read job active. The CNT2RDPN instruction sets this bit as soon as you execute a read job (JOB_RD.NO > 0 and JOB_RD.IMPOSS = 0). The CNT2RDPN instruction deletes this bit as soon as the read job is completed (JOB_RD.NO = 0).
.DONE, 1	= 1: Read job finished. The CNT2RDPN instruction sets this bit as soon as a read job is completed (regardless of errors). The CNT2RDPN instruction deletes the bit when a new read job begins. You can also delete this bit in the user program.
.IMPOSS, 2	= 1: The read job cannot be processed (counter module parameters are not assigned, the call is active or programming device (PG) mode is active). You can leave the read job (JOB_RD.NO) pending or delete it. The CNT2RDPN instruction deletes the bit if the conditions indicated above are fulfilled.
.UNKOWN, 3	= 1: Read job unknown. Your specified read job (JOB_RD) is not in the valid range (see Error evaluation). The CNT2RDPN instruction deletes this bit as soon as a valid number is entered in JOB_RD.NO. The unknown number is retained until this happens.

Error events are indicated in the binary result bit (BR = 0). Possible errors are:

- Unknown read job (see JOB_RD.UNKOWN).
- Error in data transmission with the RDREC instruction. Error information is available at the RET_VAL output parameter.

FM x51 Positioning (S7-300, S7-400)

ABS_INIT (S7-300, S7-400)

Description

The ABS_INIT instruction initializes the following data in the channel DB:

- The control signals
- The feedback signals
- The trigger bits, done bits and error bits of the jobs
- The function switches and their done bits and error bits
- Job management for the instruction ABS_CTRL (Page 5900) or ABS_CTRL_451 (Page 5904)

Call

The instruction must be executed for each channel after a startup, namely, after power-on of the module or CPU. For this reason, call the instruction in the restart organization block OB 100 and in pulling/plugging OB 83, for example or in the initialization phase of your user program. This ensures that your user program does not access obsolete data after a CPU or module restart.

Data block used

Channel DB (Page 5908):

The module address must be entered in the channel DB. You should enter it in the startup OB (OB 100).

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the channel DB

Return values

The instruction does not provide a return value.

ABS_CTRL (S7-300, S7-400)

Description

Using the ABS_CTRL instruction, you can read the operational data of all module channels, assign the channel parameters and modify them in runtime. To complete such actions, use control signals, feedback signals, function switches as well as write and read commands.

Operating principle

Each time it is called, the instruction performs the following actions:

- Read feedback signals:
  The ABS_CTRL instruction reads all feedback signals of a channel and enters these in the channel OB. The control signals and commands are not processed unless this step is completed, which means that the feedback signals report the channel status given prior to the instruction call.
- Command management (Page 5917):
  The ABS_CTRL instruction processes the read and write commands and transfers the data between the channel DB, parameter DB and the module.
- Write control signals:
  The control signals that are entered in the channel DB are transferred to the module.

Call

The ABS_CTRL instruction must be called cyclically for each channel, for example, in OB 1.

Before calling the ABS_CTRL instruction, enter all data necessary to execute the selected functions in the channel DB.

Data blocks used

- Channel DB (Page 5908):
  The channel DB is the instance DB of the instruction ABS_CTRL. The module address and the channel number must be entered in the channel DB. Incorrect information could result in I/O access errors or in an access to a different module, which in turn gives rise to data corruption.
- Parameter DB (Page 5914):
  If you want to use commands to write or read the machine data, you need a parameter DB, whose number must be entered in the channel DB.

Return values

The instruction provides the following return values:

RETVAL	BR	Description
1	1	At least 1 command is active
0	1	No active command, no errors
-1	0	Errors: Data error (DATA_ERR) or communication error (JOB_ERR)

Control signals

If a STOP signal is set or an operator error is pending or a drive enable is missing, the instruction resets the control signals START, DIR_M and DIR_P.

To enable the restart of a trip, acknowledge the operator error with OT_ERR_A=1. With this acknowledgement you cannot submit any other commands and control signals.

If there is no operator error pending, the instruction sets the acknowledgment status for the operator error OT_ERR_A to 0.

When the channel reports the start of the trip, the instruction sets and resets the start signals START, DIR_P and DIR_M, except in "jog" mode.

If the axis are not configured, the instruction puts all the control signals on hold, except the operator error acknowledgment OT_ERR_A.

Commands and control signals

You can issue several commands at the same time, also together with the control signals necessary for the positioning. If at least one write command is initiated at the same time as the control signals START, DIR_M or DIR_P, the instruction puts these control signals on hold until the write commands have been executed.

Startup

Call instructionABS_INIT (Page 5899) at the startup of the module or CPU. This action includes a reset of the function switches. The ABS_CTRL instruction acknowledges the module startup. During this time, RETVAL and JOBBUSY = 1.

Reaction to error

If incorrect data is found in a write command, the channel returns feedback signal DATA_ERR = 1 in the channel DB. If an error is found in a write or read command during communication with the module, the cause of error is saved to the JOB_ERR parameter in the channel DB.

- Error with a write command:
  The trigger bit is reset and the error bit _ERR and done bit _D are set if the command is corrupt. For all other pending write commands, the trigger bit is also reset and only the error bit _ERR is set. The awaiting write commands will be withdrawn because commands could stack up on top of each other.
  The execution of the pending read commands is continued. JOB_ERR is retriggered accordingly for all commands.
- Error with a read command:
  The trigger bit is reset and the error bit _ERR and done bit _D are set if the command is corrupt.
  The execution of the pending read commands is continued. JOB_ERR is retriggered accordingly for all commands.

ABS_DIAG (S7-300, S7-400)

Description

Use the ABS_DIAG instruction to read the data from the diagnostics buffer of the module and make it available for display on the HMI system or for programmed evaluation.

Operating principle

The instruction reads the data from the diagnostic buffer when a new entry is displayed in the diagnostics buffer by feedback signal DIAG = 1 in the channel DB. After reading the data from the diagnostics buffer, the module sets the DIAG bit in the channel DB to 0.

Call

The instruction must be called cyclically, for example, in OB 1. Additional calls in an interrupt OB are not allowed. To complete the execution of the instruction, you must call it at least twice (in 2 cycles).

Data block used

Diagnostics DB (Page 5916):

The diagnostics DB is the instance DB of the instruction ABS_DIAG. The module address must be entered in the diagnostics DB. The most recent entry in the diagnostics buffer is written to the DIAG[1] structure and the oldest entry to the DIAG[9] structure.

Return values

The instruction provides the following return values in the RETVAL parameter of the diagnostics DB:

RETVAL	BR	Description
1	1	Command active
0	1	No active command, no errors
-1	0	Errors

Commands

You can read the diagnostic buffer, independent of any new entry, by setting trigger bit DIAGRD_EN in the diagnostics DB. After the diagnostics buffer has been read, the trigger bit is set to 0.

Execute this command after the startup of a CPU and module is completed. This step ensures consistency between the content of the diagnostics DB and the diagnostics buffer of the module, even if the module has not made a new entry in the diagnostics buffer.

Startup

The instruction does not run a startup routine.

Reaction to error

Information about the cause of an execution error is available in the JOB_ERR parameter of the diagnostics DB.

Diagnostics data

Errors	Evaluation using OB 82
Module fault	OB82_MDL_DEFECT
Internal error	OB82_INT_FAULT
External error	OB82_EXT_FAULT
Channel error	OB82_PNT_INFO
External auxiliary voltage is missing	OB82_EXT_VOLTAGE
Watchdog timeout	OB82_WTCH_DOG_FLT
Failure of the internal internal supply voltage of the module	OB82_INT_PS_FLT
Hardware interrupt lost	OB82_HW_INTR_FLT

Error classes

The error events are organized in the following error classes:

- Error class 1: Operational errors (Page 5922)
- Error class 2: Operator errors (Page 5923)

- Error class 4: Data errors (Page 5924)
- Error class 5: Machine data errors (Page 5926)
- Error class 6: Incremental dimension table errors (Page 5928)
- Error class 15: Messages (Page 5928)
- Error class 128: Diagnostics errors (Page 5929)

ABS_CTRL_451 (S7-300, S7-400)

Description

Using the ABS_CTRL_451 instruction, you can read the operational data of all module channels, assign the channel parameters and change them in runtime. To complete such actions, use control signals, feedback signals, function switches as well as write and read jobs.

Operating principle

Each time it is called, the instruction performs the following actions:

- Read feedback signals:
  The ABS_CTRL_451 instruction reads all feedback signals of a channel and enters these in the channel OB. The control signals and jobs are not processed unless this step is completed, which means that the feedback signals report the channel status given prior to the instruction call.
- Job management (Page 5920):
  The ABS_CTRL_451 instruction processes the read and write jobs and transfers the data between the channel DB, parameter DB and the module.
- Write control signals:
  The control signals that are entered in the channel DB are transferred to the module.

Call

The ABS_CTRL_451 instruction must be called cyclically for each channel, for example, in OB 1.

Before calling the ABS_CTRL_451 instruction, enter all data necessary to execute the selected functions in the channel DB.

Data blocks used

- Channel DB (Page 5908):
  The module address and channel number must be entered in the channel DB. Incorrect information can result in I/O access errors or in an access to a different module, which in turn gives rise to data corruption.
- Parameter DB (Page 5914):
  If you want to use jobs to write or read machine data, you need a parameter DB, the number of which must be entered in the channel DB.

Parameter

Parameter	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the channel DB
RET_VAL	OUTPUT	INT	Return value

Return values

The instruction provides the following return values:

RET_VAL	BR	Description
1	1	At least 1 job is active
0	1	No active job, no errors
-1	0	Errors: Data error (DATA_ERR) or communication error (JOB_ERR)

Control signals

If a STOP signal is set or an operator error is pending or a drive enable is missing, the instruction resets the control signals START, DIR_M and DIR_P.

To enable the restart of a trip, acknowledge the operator error with OT_ERR_A=1. With this acknowledgment you cannot submit any other jobs and control signals.

If there is no operator error pending, the instruction sets the acknowledgment status for the operator error OT_ERR_A to 0.

When the channel reports the start of the trip, the instruction sets and resets the start signals START, DIR_P and DIR_M, except in "jog" mode..

If the axis are not configured, the instruction puts all the control signals on hold, except the operator error acknowledgment OT_ERR_A.

Jobs and control signals

You can transfer several jobs simultaneously, also together with the control signals necessary for the positioning. If at least one write job is initiated at the same time as the control signals START, DIR_M or DIR_P, the instruction puts these control signals on hold until the write jobs have been executed.

Startup

Call the ABS_INIT (Page 5899) instruction at the startup of the module or CPU. This action includes a reset the function switches. The ABS_CTRL_451 instruction acknowledges the module startup. During this time, RETVAL and JOBBUSY = 1.

Reaction to error

If incorrect data is found in a write job, the channel returns feedback signal DATA_ERR = 1 in the channel DB. If an error is found in a write or read job during communication with the module, the cause of error is saved to the JOB_ERR parameter in the channel DB.

- Write job errors:
  The trigger bit is reset and the error bit _ERR and done bit _D are set if the job is corrupt. For all other pending write jobs, the trigger bit is also reset and only the error bit _ERR is set. The other pending write jobs are revoked to prevent jobs from overlapping.
  The execution of the pending read jobs is continued. JOB_ERR is retriggered accordingly for all jobs.
- Read job errors:
  The trigger bit is reset and the error bit _ERR and done bit _D are set if the job is corrupt. The execution of the pending read jobs is continued. JOB_ERR is retriggered accordingly for all jobs.

ABS_DIAG_451 (S7-300, S7-400)

Description

Use the ABS_DIAG_451 instruction to read the data from the diagnostics buffer of the module and make it available for display on the HMI system or for programmed evaluation.

Operating principle

The instruction reads the data from the diagnostic buffer when a new entry is displayed in the diagnostics buffer by feedback signal DIAG = 1 in the channel DB. After reading the data from the diagnostics buffer, the module sets the DIAG bit in the channel DB to 0.

Call

The instruction must be called cyclically, for example, in OB 1. Additional calls in an interrupt OB are not allowed. To complete the execution of the instruction, you must call it at least twice (in 2 cycles).

Data block used

Diagnostics DB (Page 5916):

The module address must be entered in the diagnostics DB. The most recent entry in the diagnostics buffer is written to the DIAG[1] structure and the oldest entry to the DIAG[9] structure.

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the diagnostics DB
RET_VAL	OUTPUT	INT	Return value

Return values

The instruction provides the following return values in the RET_VAL parameter of the diagnostics DB:

RET_VAL	BR	Description
1	1	Job active
0	1	No active job, no errors
-1	0	Errors

Jobs

You can read the diagnostic buffer, independent of any new entry, by setting trigger bit DIAGRD_EN in the diagnostics DB. After the diagnostics buffer has been read, the trigger bit is set to 0.

Execute this job after the startup of a CPU and module is completed. This step ensures consistency between the content of the diagnostics DB and the diagnostics buffer of the module, even if the module has not made a new entry in the diagnostics buffer.

Startup

The instruction does not run a startup routine.

Reaction to errors

Information about the cause of an execution error is available in the JOB_ERR parameter of the diagnostics DB.

Diagnostics data

Errors	Evaluation using OB 82
Module fault	OB82_MDL_DEFECT
Internal error	OB82_INT_FAULT
External error	OB82_EXT_FAULT
Channel error	OB82_PNT_INFO
External auxiliary voltage is missing	OB82_EXT_VOLTAGE
Front connector missing	OB82_FLD_CONNCTR
Watchdog timeout	OB82_WTCH_DOG_FLT
Failure of the internal internal supply voltage of the module	OB82_INT_PS_FLT
Hardware interrupt lost	OB82_HW_INTR_FLT

Error classes

The error events are organized in the following error classes:

- Error class 1: Operating error (Page 5922)
- Error class 2: Operator errors (Page 5923)
- Error class 4: Data errors (Page 5924)
- Error class 5: Machine data errors (Page 5926)
- Error class 6: Incremental dimension table errors (Page 5928)
- Error class 15: Messages (Page 5928)
- Error class 128: Diagnostics errors (Page 5929)

Additional references for FM x51 ABS V1 (S7-300, S7-400)

Channel data block (S7-300, S7-400)

Purpose

The channel DB is the data interface between the user program and FM x51. It contains and stores all data necessary to control and operate a channel.

Configuration

The channel DB is divided into different areas:

Channel DB
Addresses *)
Control signals
Feedback signals
Function switches
Trigger bits for write jobs
Trigger bits for read jobs
Done bits
Error bits
Job management for instructions
Data for jobs
*) You can enter the module address in the parameter assignment interface.

Content

The table below shows the content of the channel DB.

Note

You may not change any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
Addresses				
0.0	MOD_ADDR	INT	0	Module address
2.0	CH_NO	INT	1	Channel number
10.0	PARADBNO	INT	-1	Number of the parameter DB
Control signals				
14.3	OT_ERR_A	BOOL	FALSE	1 = acknowledge operator error
15.0	START	BOOL	FALSE	1 = start positioning
15.1	STOP	BOOL	FALSE	1 = stop active traversing
15.2	DIR_M	BOOL	FALSE	1 = minus direction
15.3	DIR_P	BOOL	FALSE	1 = plus direction
15.6	SPEED252	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: Start velocity for incremental approach with increment number 252: – 0 = Creep speed – 1 = Rapid traverse
15.7	DRV_EN	BOOL	FALSE	1 = switch on drive enable
16.0	MODE_IN	BYTE	B#16#0	Required mode • 0 = no mode • 1 = Jog • 3 = reference point approach • 4 = relative incremental approach • 5 = absolute incremental approach
17.0	MODE_TYPE	BYTE	B#16#0	• Initial speed for "jog" mode – 0 = Creep speed – 1 = Rapid traverse • Increment number for incremental approach mode
Feedback signals				
22.2	DIAG	BOOL	FALSE	1 = diagnostic buffer modified
22.3	OT_ERR	BOOL	FALSE	1 = operator error occurred
22.4	DATA_ERR	BOOL	FALSE	1 = data error
22.7	PARA	BOOL	FALSE	1 = axis parameters are assigned
23.0	ST_ENBLD	BOOL	FALSE	1 = start enabled
23.1	WORKING	BOOL	FALSE	1 = positioning is active
23.2	WAIT_EI	BOOL	FALSE	1 = axis waiting for external enable

Address	Parameters	Data type	Initial value	Description
23.4	SPEED_OUT	BOOL	FALSE	• 0 = Creep speed • 1 = Rapid traverse
23.5	ZSPEED	BOOL	FALSE	1 = axis is located in the standstill range
23.6	CUTOFF	BOOL	FALSE	1 = axis is located in the switch-off range
23.7	CHGOVER	BOOL	FALSE	1 = axis is located in the changeover range
24.0	MODE_OUT	BYTE	B#16#0	Active operating mode
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
25.1	MSR_DONE	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Length measurement / edge detection completed
25.2	GO_M	BOOL	FALSE	1 = axis moves in minus direction
25.3	GO_P	BOOL	FALSE	1 = axis moves in plus direction
25.5	FVAL_DONE	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Set actual value on-the-fly completed
25.7	POS_RCD	BOOL	FALSE	1 = In position
26.0	ACT_POS	DINT	L#0	Current actual value (current position of the axis)
Function switch				
34.0	PLOOP_ON	BOOL	FALSE	1 = Loop approach in plus direction
34.1	MLOOP_ON	BOOL	FALSE	1 = Loop approach in minus direction
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input
34.3	EDGE_ON	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Edge detection on
34.4	MSR_ON	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Length measurement on
Trigger bits for write jobs				
35.0	MDWR_EN	BOOL	FALSE	1 = write machine data
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
35.2	DELDIST_EN	BOOL	FALSE	1 = Delete remaining distance
35.3	AVALREM_EN	BOOL	FALSE	1 = Cancel set actual value
35.4	TRGL1WR_EN	BOOL	FALSE	1 = Write incremental dimension table 1 (increment number 1 ... 50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = Write incremental dimension table 2 (increment number 51 ... 100)
35.6	REFPT_EN	BOOL	FALSE	1 = set reference point
35.7	AVAL_EN	BOOL	FALSE	1 = Set actual value
36.0	FVAL_EN	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Set actual value on-the-fly

Address	Parameters	Data type	Initial value	Description
36.1	ZOFF_EN	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Set zero offset
36.2	TRG252_254_EN	BOOL	FALSE	1 = Write increment for increment number 252 or 254
36.3	TRG255_EN	BOOL	FALSE	1 = write increment for increment number 255
36.4	DELDIAG_EN	BOOL	FALSE	1 = delete diagnostic buffer
Activation bits for read jobs				
36.5	MDRD_EN	BOOL	FALSE	1 = Reading machine data
36.6	TRGL1RD_EN	BOOL	FALSE	1 = Read incremental dimension table 1 (increment number 1 ... 50)
36.7	TRGL2RD_EN	BOOL	FALSE	1 = Read incremental dimension table 2 (increment number 51 ... 100)
37.0	MSRRD_EN	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Read measured values
37.1	ACTSPD_EN	BOOL	FALSE	1 = Read actual speed, remaining distance and current increment
37.2	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
Done bits for function switches				
38.0	PLOOP_D	BOOL	FALSE	1 = "loop approach in plus direction" job completed
38.1	MLOOP_D	BOOL	FALSE	1 = "loop approach in minus direction" job completed
38.2	EI_D	BOOL	FALSE	1 = "Do not evaluate enable input" job completed
38.3	EDGE_D	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = "Edge detection ON" job completed
38.4	MSR_D	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = "Length measurement ON" job completed
Done bits for write jobs				
39.0	MDWR_D	BOOL	FALSE	1 = "Write machine data" job completed
39.1	MD_D	BOOL	FALSE	1 = "Enable machine data" job completed
39.2	DELDIST_D	BOOL	FALSE	1 = "Delete remaining distance" job completed
39.3	AVALREM_D	BOOL	FALSE	1 = "Cancel set actual value" job completed
39.4	TRGL1WR_D	BOOL	FALSE	1 = "Write incremental dimension table 1" job completed
39.5	TRGL2WR_D	BOOL	FALSE	1 = "Write incremental dimension table 2" job completed
39.6	REFPT_D	BOOL	FALSE	1 = "Set reference point" job completed
39.7	AVAL_D	BOOL	FALSE	1 = "Set actual value" job completed
40.0	FVAL_D	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = "Set actual value on-the-fly" job completed

Address	Parameters	Data type	Initial value	Description
40.1	ZOFF_D	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = "Set zero offset" job completed
40.2	TRG252_254_D	BOOL	FALSE	1 = "Write increment for increment number 252 or 254" job completed
40.3	TRG255_D	BOOL	FALSE	1 = "Write increment for increment number 255" job completed
40.4	DELDIAG_D	BOOL	FALSE	1 = "Delete diagnostic buffer" job completed
Done bits for read jobs				
40.5	MDRD_D	BOOL	FALSE	1 = "Read machine data" job completed
40.6	TRGL1RD_D	BOOL	FALSE	1 = "Read incremental dimension table 1" job completed
40.7	TRGL2RD_D	BOOL	FALSE	1 = "Read incremental dimension table 2" job completed
41.0	MSRRD_D	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = "Read measured values" job completed
41.1	ACTSPD_D	BOOL	FALSE	1 = "Read actual speed, remaining distance and current increment" job completed
41.2	ENCVAL_D	BOOL	FALSE	1 = "Read encoder values" job completed
Error bits for function switches				
42.0	PLOOP_ERR	BOOL	FALSE	1 = Error with "loop approach in plus direction" job
42.1	MLOOP_ERR	BOOL	FALSE	1 = Error with "loop approach in minus direction" job
42.2	EI_ERR	BOOL	FALSE	1 = Error with "do not evaluate enable input" job
42.3	EDGE_ERR	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Error during "Edge detection ON" job
42.4	MSR_ERR	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Error during "Length measurement ON" job
Error bits for write jobs				
43.0	MDWR_ERR	BOOL	FALSE	1 = Error with "Write machine data" job
43.1	MD_ERR	BOOL	FALSE	1 = Error with "Activate machine data" job
43.2	DELDIST_ERR	BOOL	FALSE	1 = Error with "Delete remaining distance" job
43.3	AVALREM_ERR	BOOL	FALSE	1 = Error with "Cancel set actual value" job
43.4	TRGL1WR_ERR	BOOL	FALSE	1 = Error with "Write incremental dimension table 1" job
43.5	TRGL2WR_ERR	BOOL	FALSE	1 = Error with "Write incremental dimension table 2" job
43.6	REFPT_ERR	BOOL	FALSE	1 = Error with "set reference point" job
43.7	AVAL_ERR	BOOL	FALSE	1 = Error with "set actual value" job
44.0	FVAL_ERR	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Error in "set actual value on-the-fly" job

Address	Parameters	Data type	Initial value	Description
44.1	ZOFF_ERR	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Error in "set zero offset" job
44.2	TRG252_254_ERR	BOOL	FALSE	1 = Error with "Write increment for increment number 252 or 254" job
44.3	TRG255_ERR	BOOL	FALSE	1 = Error with "Write increment for incremental dimension table 255" job
44.4	DELDIAG_ERR	BOOL	FALSE	1 = Error with "Delete diagnostic buffer"
Error bits for read jobs				
44.5	MDRD_ERR	BOOL	FALSE	1 = Error with "read machine data" job
44.6	TRGL1RD_ERR	BOOL	FALSE	1 = Error with "Read incremental dimension table 1" job
44.7	TRGL2RD_ERR	BOOL	FALSE	1 = Error with "Read incremental dimension table 2" job
45.0	MSRRD_ERR	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Error with "Read measured values" job
45.1	ACTSPD_ERR	BOOL	FALSE	1 = Error with "Read actual speed, remaining distance and current increment" job
45.2	ENCVAL_ERR	BOOL	FALSE	1 = Error with "Read current encoder values" job
Job management for the instruction ABS_CTRL or ABS_CTRL_451				
48.0	JOB_ERR	INT	0	Error number of the communication error
50.0	JOBBUSY	BOOL	FALSE	1 = At least 1 job active
50.1	JOBRESET	BOOL	FALSE	1 = Reset all error bits and done bits
Data for "zero offset" job (FM 451)				
80.0	ZOFF	DINT	L#0	• FM 351: 0 (not used) • FM 451: Zero offset
Data for "Set actual value" job				
84.0	AVAL	DINT	L#0	Coordinate for "Set actual value"
Data for "Set actual value on-the-fly" (FM 451)				
88.0	FVAL	DINT	L#0	• FM 351: 0 (not used) • FM 451: Coordinate for "Set actual value on-the-fly"
Data for "Set reference point" job				
92.0	REFPT	DINT	L#0	Coordinate for "Set reference point"
Data for "Write increment for increment number 252 or 254" job				
96.0	TRG252_254	DINT	L#0	Write increment for increment number 252 or 254
Data for "Write increment for increment number 255" job				
100.0	TRG255	DINT	L#0	Increment for increment number 255
104.0	CHGDIF255	DINT	L#0	Changeover difference for increment number 255
108.0	CUTDIF255	DINT	L#0	Switch-off difference for increment number 255
Data for "Read position data" job				
112.0	ACTSPD	DINT	L#0	Current speed
116.0	DIST_TO_GO	DINT	L#0	Remaining distance
120.0	ACT_TRG	DINT	L#0	Current increment

Address	Parameters	Data type	Initial value	Description
Data for "Read encoder data" job				
124.0	ENCVAL	DINT	L#0	Actual encoder value (internal representation)
128.0	ZEROVAL	DINT	L#0	Last zero mark value (internal representation)
132.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment
Data for the "Length measurements / edge detection" job (FM 451)				
136.0	BEG_VAL	DINT	L#0	• FM 351: 0 (not used) • FM 451: Start value of length measurement / edge detection
140.0	END_VAL	DINT	L#0	• FM 351: 0 (not used) • FM 451: End value of length measurement / edge detection
144.0	LEN_VAL	DINT	L#0	• FM 351: 0 (not used) • FM 451: Length

Parameter data block (S7-300, S7-400)

Purpose

In order to change machine data and incremental dimension tables in runtime, you need a parameter DB in which this data is stored. The parameters can be changed from the user program or from an HMI system.

You can export data displayed in the parameter assignment interface to a parameter DB. You can also import a parameter DB into the parameter assignment interface and view it there.

Each module channel can have several sets of parameter assignment data, for example, for different recipes. You can switch among these in your program.

Configuration

Parameter DB
Machine data
incremental dimension tables

Content

The table below shows the content of the parameter DB.

Note

You may not change any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
Machine data				
4.0	EDGEDIST	DINT	L#0	• FM 351: 0 (not used) • FM 451: Minimum edge interval
8.0	UNITS	DINT	L#1	System of units
12.0	AXIS_TYPE	DINT	L#0	0 = Linear axis, 1 = Rotary axis
16.0	ENDROTAX	DINT	L#100 000	End of rotary axis
20.0	ENC_TYPE	DINT	L#1	Encoder type, message length
24.0	DISP_REV	DINT	L#80 000	Length per encoder revolution
32.0	INC_REV	DINT	L#500	Increments per encoder revolution
36.0	NO_REV	DINT	L#1	Number of encoder revolutions
40.0	BAUDRATE	DINT	L#0	Transmission rate
44.0	REFPT	DINT	L#0	Reference point coordinate
48.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment
52.0	REFPT_TYPE	DINT	L#0	Type of reference point approach
59.0	CNT_DIR	BOOL	FALSE	Counting direction: • 0 = Normal • 1 = Inverted
63.0	MON_WIRE	BOOL	TRUE	1 = wire break monitoring
63.1	MON_FRAME	BOOL	TRUE	1 = frame error monitoring
63.2	MON_PULSE	BOOL	TRUE	1 = missing pulses monitoring
64.0	SSW_STRT	DINT	L#-100 000 000	Software limit switch start
68.0	SSW_END	DINT	L#100 000 000	Software limit switch end
76.0	TRG_RANGE	DINT	L#1000	Target range
80.0	MON_TIME	DINT	L#2000	Monitoring time [ms]
84.0	ZSPEED_R	DINT	L#1000	Standstill range
88.0	ZSPEED_L	DINT	L#30 000	High limit of the standstill speed
92.0	CTRL_TYPE	DINT	L#1	Control type (1 - 4)
99.0	REFPT_SPD	BOOL	TRUE	Start speed in the case of a reference point approach: • 0 = Rapid traverse • 1 = Creep speed
99.1	EI_TYPE	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: Enable input: – 0 = Level-controlled – 1 = Edge controlled
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus
Increment dimension table 1				
120.0	TRGL1.TRG[1]	DINT	L#0	Increment number 1

Address	Parameters	Data type	Initial value	Description
. . .	. . .	. . .	. . .	. . .
316.0	TRGL1.TRG[50]	DINT	L#0	Increment number 50
Increment dimension table 2				
320.0	TRGL2.TRG[51]	DINT	L#0	Increment number 51
. . .	. . .	. . .	. . .	. . .
516.0	TRGL2.TRG[100]	DINT	L#0	Increment number 100

Diagnostics data block (S7-300, S7-400)

Purpose

The diagnostics DB is the storage location for the instruction ABS_DIAG (Page 5902) or ABS_DIAG_451 (Page 5906). It contains the module's diagnostics buffer which is processed by the instruction.

Configuration

Diagnostics DB
Module address
Internal data
Job status
Trigger bit
Processed diagnostics buffer

Content

The table below shows the content of the diagnostics DB.

Note

You may not change any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
0.0	MOD_ADDR	INT	0	Module address
256.0	JOB_ERR	INT	0	Error number of the communication error
258.0	JOBBUSY	BOOL	FALSE	1 = Job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = definitely read diagnostics buffer
260.0	DIAG_CNT	INT	0	Number of valid entries in the list

Address	Parameters	Data type	Initial value	Description
262.0	DIAG[1]	STRUCT		Diagnostics data - latest entry
272.0	DIAG[2]	STRUCT		Diagnostics data - second entry
282.0	DIAG[3]	STRUCT		Diagnostics data - third entry
292.0	DIAG[4]	STRUCT		Diagnostics data - fourth entry
302.0	DIAG[5]	STRUCT		Diagnostics data - fifth entry
312.0	DIAG[6]	STRUCT		Diagnostics data - sixth entry
322.0	DIAG[7]	STRUCT		Diagnostics data - seventh entry
332.0	DIAG[8]	STRUCT		Diagnostics data - eighth entry
342.0	DIAG[9]	STRUCT		Diagnostics data - ninth entry

Structure of the diagnostics entry

The table below shows the structure of a diagnostics entry DIAG[n].

Address	Parameters	Data type	Initial value	Description
+0.0	STATE	BOOL	FALSE	0 = incoming event 1 = outgoing event
+0.1	INTF	BOOL	FALSE	1 = internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 2: Operating errors 4: Data error 5: Machine data error 6: Increment dimension table error 15: Messages 128: Diagnostics errors
+4.0	FNO	INT	0	Error number
+6.0	CH_NO	INT	0	Channel number
+8.0	TRG_NO	INT	0	Increment number

Job management (ABS_CTRL) (S7-300, S7-400)

Jobs

Data communication with the module other than the transfer of control and feedback signals is handled with jobs.

In order to initiate a job, set the corresponding trigger bit in the channel DB, including the specific data for write jobs. To execute the job, call the ABS_CTRL (Page 5900) instruction.

If you use the FM 351 centrally, a read job needs precisely one cycle. If you use the FM 351 in a distributed configuration, a read job may need several cycles.

Due to the required acknowledgments of the module, at least 3 calls or OB cycles are required to execute write jobs.

Once a job has been completed, the instruction resets the trigger bit. The next job pending is identified and executed at the next call of the instruction.

In addition to the trigger bit with the _EN extension for "enable", each job is provided a done bit and an error bit. Their names have the extension _D for "done" or _ERR for "error". When the job is completed, the ABS_CTRL instruction updates the done bits and error bits. Following evaluation or prior to the initiation of a job, you should set these bits to 0.

When you set the JOBRESET bit, all done and error bits are reset prior to executing jobs pending. The JOBRESET bit is then reset to 0.

Function switches

The function switches toggle the channel states on and off. A job for writing the function switches is only executed when there is a change to a switch setting. The setting of the function switch is retained after the job has been executed.

Function switches and jobs can be used concurrently at the call of the ABS_CTRL instruction.

Similar to jobs, the function switches are provided bits with name extension _ON/_OFF, done bits with name extension _D and error bits with name extension _ERR.

You should set these bits to 0 before you initiate a job to change a function switch, in order to enable the evaluation of the done bits and error bits of the function switches.

Order of job execution

You can initiate several jobs in a single operation. If no jobs are active, the job management of the ABS_CTRL instruction performs a scan to detect set trigger bits or changes to function switches, starting with the MDWR_EN job. Any job found are executed. When the job is completed, the job management searches for the next job to be executed. After having scanned the last ENCVAL_EN job, the function restarts the scan at the MDWR_EN job. This scan is repeated until all jobs have been executed.

The jobs are processed in the following technologically appropriate order:

Order	Address in the channel DB	Parameters	Description	Reset by
Write jobs				
1	35.0	MDWR_EN	Write machine data	Instruction
2	35.1	MD_EN	Enable machine data	Instruction
	35.2	DELDIST_EN	Delete remaining distance	
	35.3	AVALREM_EN	Cancel set actual value	
	36.4	DELDIAG_EN	Delete diagnostics buffer	
3	35.4	TRGL1WR_EN	Write incremental dimension table 1	Instruction
4	35.5	TRGL2WR_EN	Write incremental dimension table 2	Instruction
5	35.6	REFPT_EN	Set reference point	Instruction
6		Function switch:		User program
	34.0	PLOOP_ON	Loop approach in plus direction	
	34.1	MLOOP_ON	Loop approach in minus direction	
	34.2	EI_OFF	Do not evaluate enable input	
7	35.7	AVAL_EN	Set actual value	Instruction

Order	Address in the channel DB	Parameters	Description	Reset by
10	36.2	TRG252_254_EN	Write increment for increment number 254	Instruction
11	36.3	TRG255_EN	Write increment for increment number 255	Instruction
Read jobs				
12	36.5	MDRD_EN	Reading machine data	Instruction
13	36.6	TRGL1RD_EN	Read incremental dimension table 1	Instruction
14	36.7	TRGL2RD_EN	Read incremental dimension table 2	Instruction
16	37.1	ACTSPD_EN	Read current velocity, remaining distance and increment	Instruction
17	37.2	ENCVAL_EN	Read encoder data	Instruction

This order enables you to completely trigger a positioning motion using a set of jobs and control signals. The jobs encompass the writing and activation of the machine data, the setup of the external enable input and the writing of increments for the incremental approaches.

Jobs during active positioning

If initiated while a positioning job is active, the write jobs listed in the following table are held until the current positioning job is completed and are then executed at the next instruction call.

Address	Parameters	Data type	Initial value	Description
34.0	PLOOP_ON	BOOL	FALSE	1 = Loop approach in plus direction
34.1	MLOOP_ON	BOOL	FALSE	1 = Loop approach in minus direction
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
35.2	DELDIST_EN	BOOL	FALSE	1 = Delete remaining distance
35.3	AVALREM_EN	BOOL	FALSE	1 = Cancel set actual value
35.6	REFPT_EN	BOOL	FALSE	1 = Set reference point coordinate
35.7	AVAL_EN	BOOL	FALSE	1 = Set actual value
36.4	DELDIAG_EN	BOOL	FALSE	1 = Delete diagnostics buffer

Job status

The status of job execution can be read from the return value RETVAL and activity bit JOBBUSY in the channel DB. The status of an individual job can be evaluated based on its trigger bits, done bits and error bits.

	RETVAL	JOBBUSY	Trigger bit _EN	Done bit _D	Error bit _ERR
Job active	1	1	1	0	0
Job completed without errors	0	0	0	1	0
Job completed with errors	-1	0	0	1	1
Write job cancel-led	-1	0	0	0	1

Job management (ABS_CTRL_451) (S7-300, S7-400)

Jobs

Data communication with the module other than the control and feedback signals is handled using jobs.

To initiate a job, set the corresponding trigger bit in the channel DB and the corresponding data for write jobs. Then call the ABS_CTRL_451 (Page 5904) instruction to execute the job.

Due to the required acknowledgments of the module, at least 3 calls or OB cycles are required to execute write jobs. A read job is executed immediately.

Once a job has been completed, the instruction resets the trigger bit. The next job pending is identified and executed at the next call of the instruction.

In addition to the trigger bit with the _EN extension for "enable", each job is provided a done bit and an error bit. Their names have the extension _D for "done" or _ERR for "error". When the job is completed, the ABS_CTRL_451 instruction updates the done bits and error bits. After the evaluation or prior to issuing a job, these bits should be set to 0.

When you set the JOBRESET bit, all done and error bits are reset before pending jobs are executed. The JOBRESET bit is then reset to 0.

Function switches

The function switches switch the states of the channel on and off. A job for writing the function switches is only executed when there is a change to a switch setting. The setting of the function switch is retained after the job has been executed.

Function switches and jobs can be used concurrently at the call of ABS_CTRL_451.

Similar to jobs, the function switches are provided bits with name extension _ON/_OFF, done bits with name extension _D and error bits with name extension _ERR.

To be able to evaluate the done bits and error bits of the function switches, you should set these bits to 0 before you issue a job to change a function switch.

Order of job execution

You can select several jobs simultaneously. If no jobs are active, the job management of the ABS_CTRL_451 instruction performs a scan to detect set trigger bits or changes to function switches, starting with the MDWR_EN job. If a job is found, it is processed. When the job is concluded, the job management searches for the next job to be processed. After having scanned the last ENCVAL_EN job, the function restarts the scan at the MDWR_EN job. This searching process is repeated until all the jobs have been processed.

The jobs are processed in the following technologically appropriate order:

Order	Address in the channel DB	Parameters	Description	Reset from
Write jobs				
1	35.0	MDWR_EN	Write machine data	Instruction

Order	Address in the channel DB	Parameters	Description	Reset from
2	35.1 35.2 35.3 36.4	MD_EN DELDIST_EN AVALREM_EN DELDIAG_EN	Enable machine data Delete remaining distance Cancel set actual value Delete diagnostic buffer	Instruction
3	35.4	TRGL1WR_EN	Write incremental dimension table 1	Instruction
4	35.5	TRGL2WR_EN	Write incremental dimension table 2	Instruction
5	35.6	REFPT_EN	Set reference point	Instruction
6	 34.0 34.1 34.2 34.3 34.4	Function switch: PLOOP_ON MLOOP_ON EI_OFF EDGE_ON MSR_ON	 Loop approach in plus direction Loop approach in minus direction Do not evaluate enable input Edge detection on Length measurement on	User program
7	35.7	AVAL_EN	Set actual value	Instruction
8	36.0	FVAL_EN	Set actual value on-the-fly	Instruction
9	36.1	ZOFF_EN	Set zero point offset	Instruction
10	36.2	TRG252_254_EN	Write increment for increment number 252/254	Instruction
11	36.3	TRG255_EN	Write increment for increment number 255	Instruction
Read jobs				
12	36.5	MDRD_EN	Reading machine data	Instruction
13	36.6	TRGL1RD_EN	Read incremental dimension table 1	Instruction
14	36.7	TRGL2RD_EN	Read incremental dimension table 2	Instruction
15	37.0	MSRRD_EN	Read measured values	Instruction
16	37.1	ACTSPD_EN	Read current velocity, remaining distance and current increment	Instruction
17	37.2	ENCVAL_EN	Read encoder data	Instruction

This order enables you to completely trigger a positioning with a set of jobs and control signals. The jobs go from writing and activating the machine data through the setting of the external enable input up to writing the increments for the incremental approaches.

Jobs during an ongoing positioning

If initiated while a positioning job is active, the write jobs listed in the following table are held until the current positioning job is completed and are then executed at the next instruction call.

Address	Parameters	Data type	Initial value	Description
34.0	PLOOP_ON	BOOL	FALSE	1 = loop approach in plus direction
34.1	MLOOP_ON	BOOL	FALSE	1 = loop approach in minus direction
34.2	EI_OFF	BOOL	FALSE	1 = do not evaluate enable input
34.3	EDGE_ON	BOOL	FALSE	1 = Edge detection on
34.4	MSR_ON	BOOL	FALSE	1 = Length measurement on

Address	Parameters	Data type	Initial value	Description
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
35.2	DELDIST_EN	BOOL	FALSE	1 = delete remaining distance
35.3	AVALREM_EN	BOOL	FALSE	1 = cancel set actual value
35.6	REFPT_EN	BOOL	FALSE	1 = set reference point coordinate
35.7	AVAL_EN	BOOL	FALSE	1 = Set actual value
36.1	ZOFF_EN	BOOL	FALSE	1 = Set zero offset
36.4	DELDIAG_EN	BOOL	FALSE	1 = delete diagnostic buffer

Job status

The status of job execution can be read from the return value RET_VAL and activity bit JOBBUSY in the channel DB. The status of an individual job can be evaluated using the trigger bits, done bits, and error bits of this job.

	RET_VAL	JOBBUSY	Trigger bit _EN	Done bit _D	Error bit _ERR
Job active	1	1	1	0	0
Job completed without errors	0	0	0	1	0
Job completed with errors	-1	0	0	1	1
Write job cancel-led	-1	0	0	0	1

Error classes (S7-300, S7-400)

Error class 1: Process error (S7-300, S7-400)

Description

Operating errors are detected asynchronously to an operation/control. These operating errors cause the positioning to be canceled, except in the case of error number 9, which leads to the shutdown of positioning.

No.	Meaning		Diagnostics interrupt
1	Software limit switch start overrun		Yes
	Cause	The actual value is outside the operating range.	
2	Software limit switch end overrun		Yes
	Cause	The actual value is outside the operating range.	
3	Traversing range start overrun		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits also belong to the traversing range).	
4	Traversing range end overrun		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits also belong to the traversing range).	

No.	Meaning		Diagnostics interrupt
5	**Error on target approach**		Yes
	Cause	Target range has not been reached within the monitoring time.	
6	**Standstill range exited**		Yes
	Cause	The actual value lies outside of the standstill range.	
7	**Positive feedback**		Yes
	Cause	Actual value change > ½ standstill range in the wrong direction.	
8	**Missing or too slight actual value change**		Yes
	Cause	No actual value change or an actual value change against the desired direction within the monitoring time.	
9	**Target overrun (FM 451)**		Yes
	Cause	The target has been overrun with "set actual value on-the-fly".	
10	**Target range overrun**		Yes
	Cause	Target range was overrun after target approach.	
11	**Changeover point switched incorrectly**		Yes
	Cause	Axis swinging in changeover point.	
12	**Switch-off point switched incorrectly**		Yes
	Cause	Axis swinging between switch-off / return point.	
13	**Start of target range switched incorrectly**		Yes
	Cause	Axis oscillating in the target range.	
14	**Change greater than half the rotary axis range**		Yes
	Cause	Velocity/frequency too high or faulty actual value steps.	
15	**Change greater than the rotary axis range**		Yes
	Cause	Velocity/frequency too high or faulty actual value steps.	
16	**Increment for increment number 252 cannot be transferred (FM 451)**		Yes
	Cause	The increment cannot be transferred.	
17	**Increment for increment number 252 cannot be approached (FM 451)**		Yes
	Cause	The distance between the current actual position and the specified increment is less than the changeover difference or switch-off difference.	
18	**Incorrect increment for increment number 252 (FM 451)**		Yes
	Cause	The increment lies outside of the operating range.	

Error class 2: Operator error (S7-300, S7-400)

Description

Operator errors are detected if there is change in the control signals in the user data range. Operator errors lead to the shutdown of positioning.

No.	Meaning		Diagnostics interrupt
1	**Impermissible operating mode**		No
	Cause	The chosen operating mode is inadmissible.	

No.	Meaning		Diagnostics interrupt
3	**Inadmissible interface job**		No
	Cause	The selected signal is not permitted with this operating mode.	
4	**Incorrect operating mode parameter**		No
	Cause	• In "jog" mode, the velocity specification is not equal to the rapid traverse or creep speed. • In "incremental approach" mode, the increment is a value other than 1 - 100, 252, 254 or 255	
5	**Start enable not available**		No
	Cause	Start enable not available upon start.	
7	**Target / target range is outside the operating range**		No
	Cause	Default or calculated target is outside the software limit switches.	
8	**Axis parameters not assigned**		No
	Cause	Incorrect machine data or no machine data has been assigned for the axis.	
9	**Axis not synchronized**		No
	Cause	The "incremental approach" mode is only possible with an already synchronized axis.	
10	**Target / distance increment cannot be positioned**		No
	Cause	The distance between the current actual position and the specified target is less than the switch-off difference.	
17	**Reference point approach not possible**		No
	Cause	An SSI encoder is connected.	
18	**Relative or absolute incremental approach is not possible**		No
	Cause	The increment is invalid.	
19	**Switch-off difference is less than ½ the target range with increment number 255**		No
	Cause	The switch-off difference for the increment number 255 is less than half the target range.	
20	**Inadmissible travel in the specified direction**		No
	Cause	Insufficient distance to the software limit switch.	

Error class 4: Data error (S7-300, S7-400)

Meaning

Operating errors are detected synchronously to an operation / control. Data errors do not trigger an error response.

No.	Meaning		Diagnostics interrupt
6	**Specified increment too great**		No
	Cause	The value is outside the range of ±100 m or ±1000 m. The distance increment / target may not be greater than the traversing range. For a rotary axis, the coordinate must be ≥ 0 and less than the end of the rotary axis.	

No.	Meaning		Diagnostics interrupt
10	Zero offset error		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. After the zero offset, the software limit switches are positioned outside of the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	Incorrect actual value		No
	Cause	Linear axis: The coordinate is outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
12	Incorrect reference point		No
	Cause	Linear axis: The coordinate is outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
20	Enable machine data not permitted		No
	Cause	There are no new (error-free) machine data on the module.	
27	Unauthorized bit-coded setting		No
	Cause	Unused and, in this case, unwritten bits do not equal 0.	
29	Inadmissible bit coding		No
	Cause	Unused and, in this case, unwritten bits do not equal 0.	
34	Not possible to cancel set actual value		No
	Cause	With an SSI encoder and a linear axis, the actual position value would lie outside the operating range after making the setting.	
36	Incorrect changeover difference with the increment number 255		No
	Cause	The value lies outside the permissible range of numbers of ±100 m or ±1000 m. For a rotary axis, the coordinate must be ≥ 0 and less than the end of the rotary axis.	
37	Incorrect switch-off difference with the increment number 255		No
	Cause	The value lies outside the permissible range of numbers of ±100 m or ±1000 m. The switch-off difference must be less than the changeover difference.	
107	Axis parameters not assigned		No
	Cause	On the axis there is either no machine data or it has not been enabled.	
108	Axis not synchronized		No
	Cause	A "set actual value" or "set actual value on-the-fly" job was initiated although the axis is not synchronized.	

Error class 5: Machine data error (S7-300, S7-400)

Meaning

The diagnostic interrupt is only triggered in the event of a faulty system data block (SDB). Machine data errors do not trigger an error response.

No.	Meaning		Diagnostics interrupt
5	Error in hardware interrupt setting		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	Incorrect minimum edge distance (FM 451)		Yes
	Cause	You have entered a value < 0 or > 10^9 µm as the minimum edge distance.	
7	Incorrect system of units		Yes
	Cause	The value for the system of units lies outside of the permissible range of 1 to 4 and 6.	
8	Incorrect axis type		Yes
	Cause	You have entered neither 0 nor 1 as an axis type.	
9	Incorrect end of the rotary axis		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or from 1 to 10^8 µm (depending on the resolution).	
10	Incorrect encoder type		Yes
	Cause	The value for the encoder type lies outside of the permissible range of 1 to 4.	
11	Incorrect distance per encoder revolution		Yes
	Cause	The value for distance / encoder revolution is outside the valid range from 1 to 10^9 µm (regardless of the resolution).	
13	Invalid increments per encoder revolution		Yes
14	Incorrect number of revolutions		Yes
15	Incorrect transmission rate		Yes
	Cause	You have specified a transmission rate outside the permitted range of 0 to 3.	
16	Incorrect reference point coordinate		Yes
	Cause	The coordinate is outside the range from -100 m to +100 m or from 1000 m to +1000 m (depending on the resolution). Linear axis: The coordinate lies outside of the operating range. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
17	Incorrect absolute encoder adjustment		Yes
	Cause	SSI encoder: The value of the absolute encoder adjustment is not in the encoder range: (increments per encoder revolution x number of revolutions - 1).	
18	Incorrect type of reference point approach		Yes
	Cause	You have specified a value outside the permissible value set of 0, 1, 2 and 3.	
19	Incorrect counting direction		Yes
	Cause	You have specified a value outside of the permissible value quantities of 0 and 1.	

No.	Meaning		Diagnostics interrupt
20	It is not possible to monitor the hardware		Yes
	Cause	You have set the monitoring of message frame errors in the parameter DB to "FALSE". The encoder used does not support error pulse monitoring. Disable the MON_PULSE parameter.	
21	Incorrect software limit switch start		Yes
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	Incorrect software limit switch end		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution) or less than the software limit switch start. Linear axis: The software limit switch end (including any zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
23	Incorrect maximum velocity		Yes
	Cause	The data not listed in the parameter DB must be 0.	
24	Incorrect target range		Yes
	Cause	Linear axis: Range from 0 to 100 m or 1000 m, depending on the resolution. Rotary axis: Range greater than the end of the rotary axis.	
25	Incorrect monitoring time		Yes
	Cause	The value for the monitoring time lies outside the permissible range from 0 to 100 000 ms.	
26	Incorrect standstill range		Yes
	Cause	Linear axis: Range from 0 to 100 m or 1000 m, depending on the resolution. Rotary axis: Range greater than the end of the rotary axis.	
127	Incorrect standstill velocity		Yes
	Cause	The value for the standstill velocity lies outside the permissible range from 0 to 100 000 µm/min.	
128	Incorrect control mode		Yes
	Cause	You have specified a value for the control mode that lies outside the permissible range of 1 to 4.	
129	Incorrect start velocity for the reference point approach		Yes
	Cause	You have entered neither 0 nor 1 as a start velocity.	
130	Incorrect changeover difference in the direction +		Yes
	Cause	Linear axis: Range from 0 to 100 m or 1000 m (depending on the resolution). Rotary axis: Range greater than the end of the rotary axis or less than ½ the target range.	
131	Incorrect changeover difference in direction –		Yes
	Cause	Linear axis: Range from 0 to 100 m or 1000 m (depending on the resolution). Rotary axis: Range greater than the end of the rotary axis or less than ½ the target range.	

No.	Meaning		Diagnostics interrupt
132	**Incorrect switch-off difference in direction +**		Yes
	Cause	The switch-off difference is greater than the plus changeover difference, less than ½ the target range or outside the valid range from 0 to 100 m or 0 to 1000 m (depending on resolution).	
133	**Incorrect switch-off difference in direction –**		Yes
	Cause	The switch-off difference is greater than the minus changeover difference, less than ½ the target range or outside the valid range from 0 to 100 m or 0 to 1000 m (depending on resolution).	
200	**Incorrect resolution**		Yes
	Cause	You specified a resolution < 0.1 µm/pulse or > 1000 µm/pulse. You specified a distance / encoder revolution and a number of pulses/encoder revolution, that results in a resolution of < 0.1 or > 1000.	
201	**Encoder does not fit the operating range / rotary axis range**		Yes
	Cause	SSI encoders and rotary axes: The encoder does not precisely cover the rotary axis range. Linear axis: The encoder does not cover at least the operating range (including software limit switches).	

Error class 6: Increment table error (S7-300, S7-400)

Meaning

Increment dimension table errors do not trigger an error response.

No.	Meaning		Diagnostics interrupt
6	**Specified increment in the incremental dimension table too great**		No
	Cause	The value lies outside of ±100 m or ±1000 m. The increment/target may not be greater than the traversing range. For a rotary axis, the coordinate must be ≥ 0 and less than the end of the rotary axis.	

Error class 15: Messages (S7-300, S7-400)

Meaning

The messages do not trigger an error response.

No.	Meaning		Diagnostics interrupt
1	**Start of parameter assignment**		No
	Cause	The module has detected a parameter assignment via a system data block.	

No.	Meaning		Diagnostics interrupt
2	**End of parameter assignment**		No
	Cause	The module has processed the parameter assignment via a system data block without errors.	
11	**Distance to changeover point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	
12	**Distance to reversal point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	
14	**Distance to switch-off point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	
15	**Distance to start of target range too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	

Error class 128: Diagnostics errors (S7-300, S7-400)

No.	Meaning		Diagnostics interrupt
4	**External auxiliary voltage missing**		Yes
	Cause	• External 24 V auxiliary voltage is not connected or has failed • The fuse on the module is defective • Undervoltage • Ground wire break • Short-circuit (for example, at the connected encoder)	
	Effect	• Positioning is cancelled on all channels. • Outputs are switched off • Synchronization is cancelled for incremental encoders when the auxiliary voltage for the encoder supply is missing • Module parameters not assigned. • Start enable is cleared	
	Remedy	Verify that the 24 V connection is correct (faulty module if 24 V connection is correct).	
5	**Front connector missing (FM 451)**		Yes
	Cause	Front connector of the positioning module is not connected.	
	Effect	• Missing external auxiliary 24 V voltage • Module is not ready	
	Correction	Plug the front connector into the positioning module	

No.	Meaning		Diagnostics interrupt
51	**Timeout monitoring responded (watchdog)**		Yes
	Cause	• Module exposed to severe interference • Module error	
	Effect	• Module is reset • Turn off all outputs • If, after resetting the module, no module fault is detected, the module is ready for operation again • The module signals the watchdog timeout with "incoming" and "outgoing"	
	Correction	• Remedying interferences • Contact the relevant sales department who will require details of the circumstances leading to the error. • Replace the module	
144	**Encoder wire break**		Yes
	Cause	• The encoder cable has sheared off or is not plugged in • Encoders without cross signals • Incorrect pin assignment • Cable length too long • Encoder signals short-circuited • Encoder signal edge fault • Maximum input frequency of the encoder input has been exceeded • Sensor voltage supply failure	
	Effect	• The positioning is canceled • Turn off all outputs • Delete the synchronization with incremental encoders • Start enable is deleted	
	Correction	• Check encoder cable • Comply with encoder specification • Monitoring can be temporarily disabled at the responsibility of the operator using the programming interface. • Comply with the technical specifications of the module	

No.	Meaning		Diagnostics interrupt
145	**Absolute encoder message frame error**		Yes
	Cause	Incorrect or interrupted message frame traffic between the module and the absolute encoder (SSI): • The encoder cable has sheared off or is not plugged in • Incorrect encoder type • The encoder is incorrectly set (programmable encoder) • The message length has been incorrectly specified • The encoder is supplying incorrect values (encoder is faulty) • Interference on the measuring system cable • Chosen transmission rate is too high • Monoflop time of the encoder is greater than 64 µs	
	Effect	• Positioning is canceled • Shutdown of the outputs • Start enable is cleared	
	Correction	• Check encoder cable • Check encoder • Check the message frame traffic between the encoder and module	
146	**Missing pulses of incremental encoder**		Yes
	Cause	• Encoder monitoring has detected missing pulses • Number of increments per encoder revolution is incorrectly entered • Encoder faulty: does not return the defined number of pulses • incorrect or no zero mark • Crosstalk on the encoder cable	
	Effect	• Positioning is canceled • Shutdown of the outputs • Start enable is cleared	
	Remedy	• Enter the correct number of increments per encoder revolution (parameter assignment dialog) • Check the encoder and encoder cable • Comply with the shielding and grounding directives • Monitoring can be temporarily disabled at the responsibility of the operator using the programming interface.	

FM x52 Cam Controlling (S7-300, S7-400)

CAM_INIT (S7-300, S7-400)

Description

The CAM_INIT instruction initializes the following data in the channel DB:

- The control signals
- The feedback signals
- The trigger, done and error bits of the jobs
- The function switches and their done and error bits
- Job management and the internal buffers for the CAM_CTRL (Page 5932) or CAM_CTRL_452 (Page 5935) instruction
- Job management and the internal buffers for the CAM_MSRM_452 (Page 5939) instruction (FM 452 only)

Call

The instruction must be executed after a startup, in other words, after the power supply to the module or CPU is switched on. Therefore, you should integrate the instruction in the restart organization block OB 100, for example, and plug/pull organization block OB 83 or call it in the initialization phase of your user program. Before you do that, enter the module address in the DB of OB 100. This ensures that your user program does not access obsolete data after a CPU or module restart.

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the channel DB

Return values

The instruction does not provide a return value.

CAM_CTRL (S7-300, S7-400)

Description

You can call CAM_CTRL instruction to read operating data from the module, initialize the module and control the module in runtime. To complete such actions, use control signals, feedback signals, function switches as well as write and read commands.

Operating principle

Each time it is called, the instruction performs the following actions:

- Read feedback signals:
  The CAM_CTRL instruction reads all feedback signals from the module and enters them in the channel DB. The control signals and commands are not processed unless this step is completed, which means that the feedback signals report the module status given prior to the instruction call.
- Write control signals:
  The control signals that are entered in the channel DB are transferred to the module. Enabling of cam processing, however, is delayed as long as the trigger for a "Set reference point" or "Write cam data" command is set. Activation (reactivation) of cam processing is delayed by this time period.
- Command management (Page 5951):
  The next command is executed based on the command trigger bits entered in the channel DB.

Call

This instruction must be called at cyclic intervals.

Before calling the instruction, enter all the data necessary to execute the selected functions in the channel DB.

Data blocks used

- Channel DB (Page 5940):
  The channel DB is the instance DB of the instruction CAM_CTRL. The module address must be entered in the channel DB. .
- Parameter DB (Page 5947):
  To write or read machine or cam data using commands, you need a parameter DB whose number is entered in the channel DB. The parameter DB must be of sufficient length to handle the existing number of cams.

Return values

The instruction provides the following return values in the RETVAL parameter of the channel DB:

RETVAL	BR	Description
1	1	At least 1 command is active
0	1	No active command, no errors
-1	0	Errors: Data error (DAT_ERR) or Communication error (JOB_ERR) has occurred

Startup

Call the CAM_INIT (Page 5932) instruction at the startup of the module or CPU. This action includes a reset of the function switches.

The CAM_CTRL instruction acknowledges the module startup. During this time, RETVAL and JOBBUSY = 1.

Reaction to error

If corrupt data was written by a write command, the module returns the feedback signal DATA_ERR = 1. If an error occurs during communication with the module for a write or read command, the cause of the error is written to the JOB_ERR parameter of the channel DB.

- Error with a write command:
  For the corrupt command, the trigger bit is reset, and error bit _ERR and done bit _D are set. The trigger bit is canceled and error bit _ERR is set for all pending write commands. The execution of the pending read commands is continued. JOB_ERR is retriggered accordingly for all commands.
- Error with a read command:
  For the corrupt command, the trigger bit is reset, and error bit _ERR and done bit _D are set.
  The execution of the pending read commands is continued. JOB_ERR is retriggered accordingly for all commands.

CAM_DIAG (S7-300, S7-400)

Description

Use the CAM_DIAG instruction to read the data from the diagnostics buffer of the module and make it available for display on the HMI system or for programmed evaluation.

Operating principle

The instruction reads the diagnostics buffer if a new entry is displayed in the diagnostics buffer by feedback signal DIAG = 1. After the diagnostics buffer has been read, the module sets DIAG to 0.

Call

This instruction must be called at cyclic intervals. Additional calls in an event-driven interrupt program are not allowed. To complete the execution of the instruction, you must call it at least twice (in 2 cycles).

Data block used

Diagnostics DB (Page 5949):

The diagnostics DB is the instance DB of the instruction CAM_DIAG. The module address must be entered in the diagnostics DB. The address should entered in the startup OB (OB 100)

or prior to its initial use. The most recent entry in the diagnostics buffer is written to the DIAG[1] structure and the oldest entry to the DIAG[4] structure.

Return values

The instruction provides the following return values in the RETVAL parameter of the diagnostics DB:

RETVAL	BR	Description
1	1	Command active
0	1	No active command, no errors
-1	0	Errors

Commands

You can read the diagnostics buffer regardless of any new entry by setting the DIAGRD_EN trigger bit. After the diagnostics buffer has been read, the trigger bit is set to 0.

Startup

The instruction does not run a startup routine.

Reaction to error

The cause of a command error can be read from the JOB_ERR parameter of the diagnostics DB.

Error classes

The error events are organized in the following error classes:

- Error class 1: Operational errors (Page 5953)
- Error class 4: Data errors (Page 5954)
- Error class 5: Machine data errors (Page 5956)
- Error class 7: Cam data errors (Page 5958)
- Error class 15: Messages (Page 5959)
- Error class 128: Diagnostics errors (Page 5959)

CAM_CTRL_452 (S7-300, S7-400)

Description

You can call CAM_CTRL_452 instruction to read operating data from the module, initialize the module and control the module in runtime. To complete such actions, use control signals, feedback signals, function switches as well as write and read jobs.

Operating principle

Each time it is called, the instruction performs the following actions:

- Read feedback signals:
  The CAM_CTRL_452 instruction reads all feedback signals from the module and enters them in the channel DB. The control signals and jobs are not processed unless this step is completed, which means that the feedback signals report the module status given prior to the instruction call.
- Write control signals:
  The control signals that are entered in the channel DB are transferred to the module. Enabling of cam processing, however, is delayed as long as the trigger for a "Set reference point" or "Write cam data" job is set. Activation (reactivation) of cam processing is delayed by this time period.
- Job management (Page 5952):
  The next job is executed based on the job trigger bits entered in the channel DB.

Call

This instruction must be called at cyclic intervals.

Before calling the instruction, enter all the data necessary to execute the selected functions in the channel DB.

Data blocks used

- Channel DB (Page 5940):
  The module address must be entered in the channel DB.
- Parameter DB (Page 5947):
  To write or read machine or cam data using jobs, you need a parameter DB whose number is entered in the channel DB. The parameter DB must be of sufficient length to handle the existing number of cams.

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the channel DB
RET_VAL	OUTPUT	INT	Return value

Return values

The instruction provides the following return values in the RET_VAL parameter of the channel DB:

RET_VAL	BR	Description
1	1	At least 1 job is active
0	1	No active job, no errors
-1	0	Errors: Data error (DAT_ERR) or Communication error (JOB_ERR) has occurred

Startup

Call the CAM_INIT (Page 5932) instruction at the startup of the module or CPU. This action includes a reset the function switches.

The CAM_CTRL_452 instruction acknowledges the module startup. During this time, RET_VAL and JOBBUSY = 1.

Reaction to errors

If corrupt data was written by a write job, the module returns the feedback signal DATA_ERR = 1. If an error occurs during communication with the module for a write or read job, the cause of the error is written to the JOB_ERR parameter of the channel DB.

- Write job errors:
  For the corrupt job, the trigger bit is reset, and error bit _ERR and done bit _D are set. The trigger bit is canceled and error bit _ERR is set for all pending write jobs.
  The execution of the pending read jobs is continued. JOB_ERR is retriggered accordingly for all jobs.
- Read job errors:
  For the corrupt job, the trigger bit is reset, and error bit _ERR and done bit _D are set.
  The execution of the pending read jobs is continued. JOB_ERR is retriggered accordingly for all jobs.

CAM_DIAG_452 (S7-300, S7-400)

Description

Use the CAM_DIAG_452 instruction to read the data from the diagnostics buffer of the module and make it available for display on the HMI system or for programmed evaluation.

Operating principle

The instruction reads the diagnostics buffer if a new entry is displayed in the diagnostics buffer by feedback signal DIAG = 1. After the diagnostics buffer has been read, the module sets DIAG to 0.

Call

This instruction must be called at cyclic intervals. Additional calls in an interrupt OB are not allowed. To complete the execution of the instruction, you must call it at least twice (in 2 cycles).

Data block used

Diagnostics DB (Page 5949):

The module address must be entered in the diagnostics DB. The most recent entry in the diagnostics buffer is written to the DIAG[1] structure and the oldest entry to the DIAG[4] structure.

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the diagnostics DB
RET_VAL	OUTPUT	INT	Return value

Return values

The instruction provides the following return values in the RET_VAL parameter of the diagnostics DB:

RET_VAL	BR	Description
1	1	Job active
0	1	No active job, no errors
-1	0	Errors

Jobs

You can read the diagnostics buffer regardless of any new entry by setting the DIAGRD_EN trigger bit. After the diagnostics buffer has been read, the trigger bit is set to 0.

Startup

The instruction does not run a startup routine.

Reaction to errors

The cause of a job error can be read from the JOB_ERR parameter of the diagnostics DB.

Error classes

The error events are organized in the following error classes:

- Error class 1: Operating error (Page 5953)
- Error class 4: Data errors (Page 5954)

- Error class 5: Machine data errors (Page 5956)
- Error class 7: Cam data errors (Page 5958)
- Error class 15: Messages (Page 5959)
- Error class 128: Diagnostics errors (Page 5959)

CAM_MSRM_452 (S7-300, S7-400)

Description

Call the CAM_MSRM_452 instruction to immediately evaluate the length measurement or edge detection data of an FM 452 in the hardware interrupt OB.

Call

The instruction is called in a hardware interrupt OB (for example, OB 40).

Data block used

Channel DB (Page 5940):

The module address must be entered in the channel DB. You should enter it in the startup OB (OB 100).

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the channel DB
RET_VAL	OUTPUT	INT	Return value

Return values

The instruction provides the following return values in the RET_VAL parameter of the channel DB:

RET_VAL	BR	Description
1	1	Job active
0	1	No active job, no errors
-1	0	Errors

Measurement results and status information

The measurement results and status information are available in the channel DB:

Address	Name	Type	Initial value	Comment
112.0	BEG_VAL	DINT	L#0	Initial value
116.0	END_VAL	DINT	L#0	End value
120.0	LEN_VAL	DINT	L#0	Length
56.0	JOB_ERR_M	INT	0	Communication errors
58.0	JOBBUSY_M	BOOL	FALSE	Job active

Startup

The instruction does not run a startup routine.

Reaction to errors

The cause of a job error can be read from the JOB_ERR_M parameter of the channel DB.

Additional references for FM x52 CAM V2 (S7-300, S7-400)

Channel data block (S7-300, S7-400)

Purpose

The channel DB is the data interface between the user program and the electronic cam controller FM x52. It contains and saves all data necessary to control and operate the module.

Configuration

The channel DB is divided into different areas:

Channel DB
Addresses* / version switch
Control signals
Feedback signals
Function switches
Trigger bits for write jobs
Trigger bits for read jobs
Done bits
Error bits
Job management for instructions
Data for jobs
* You can enter the module address in the parameter assignment interface.

Content

The table below shows the content of the channel DB.

Note

Do not modify any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
Addresses / version switch				
0.0	MOD_ADDR (Enter!)	INT	0	Module address
2.0	CH_NO	INT	1	Channel number (always 1)
10.0	PARADBNO	INT	-1	Number of the parameter DB -1 = DB not available
12.0	FM_TYPE	BOOL	FALSE	0 = FM 352 up to V4.0 1 = FM 452 or FM 352 V5.0 and higher
Control signals				
15.2	DIR_M	BOOL	FALSE	1 = simulation in negative direction
15.3	DIR_P	BOOL	FALSE	1 = simulation in positive direction
15.4	CAM_EN	BOOL	FALSE	1 = enable cam processing
15.5	CNTC0_EN	BOOL	FALSE	1 = enable count function of counter cam track 0
15.6	CNTC1_EN	BOOL	FALSE	1 = enable count function of counter cam track 1
16.0	TRACK_EN	WORD	W#16#0	Enable cam tracks 0 to 15 Bit 0 = track 0
Feedback signals				
22.2	DIAG	BOOL	FALSE	1 = diagnostic buffer modified
22.4	DATA_ERR	BOOL	FALSE	1 = data error
22.7	PARA	BOOL	FALSE	1 = module has parameters assigned
23.4	CAM_ACT	BOOL	FALSE	1 = cam processing busy
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
25.1	MSR_DONE	BOOL	FALSE	1 = length measurement or edge detection completed
25.2	GO_M	BOOL	FALSE	1 = axis moving in negative direction
25.3	GO_P	BOOL	FALSE	1 = axis moving in positive direction
25.4	HYS	BOOL	FALSE	1 = axis within the hysteresis range
25.5	FVAL_DONE	BOOL	FALSE	1 = set actual value on-the-fly executed
26.0	ACT_POS	DINT	L#0	Current position of axis
30.0	TRACK_OUT	DWORD	DW#16#0	Current signals of tracks 0 to 31 Bit 0 = track 0
Function switches				
34.0	EDGE_ON	BOOL	FALSE	1 = Edge detection on
34.1	SIM_ON	BOOL	FALSE	1 = simulation on

Address	Parameters	Data type	Initial value	Description
34.2	MSR_ON	BOOL	FALSE	1 = Length measurement on
34.3	REFTR_ON	BOOL	FALSE	1 = retrigger reference point
34.4	SSW_OFF	BOOL	FALSE	1 = Software limit switches disabled
Trigger bits for write jobs				
35.0	MDWR_EN	BOOL	FALSE	1 = write machine data
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
35.2	AVALREM_EN	BOOL	FALSE	1 = set actual value, cancel set actual value on-the-fly
35.3	CAM1WR_EN	BOOL	FALSE	1 = write cam data 1 (cams 0 to 15)
35.4	CAM2WR_EN	BOOL	FALSE	1 = write cam data 2 (cams 16 to 31)
35.5	CAM3WR_EN	BOOL	FALSE	1 = write cam data 3 (cams 32 to 47)
35.6	CAM4WR_EN	BOOL	FALSE	1 = write cam data 4 (cams 48 to 63)
35.7	CAM5WR_EN	BOOL	FALSE	1 = write cam data 5 (cams 64 to 79)
36.0	CAM6WR_EN	BOOL	FALSE	1 = write cam data 6 (cams 80 to 95)
36.1	CAM7WR_EN	BOOL	FALSE	1 = write cam data 7 (cams 96 to 111)
36.2	CAM8WR_EN	BOOL	FALSE	1 = write cam data 8 (cams 112 to 127)
36.3	REFPT_EN	BOOL	FALSE	1 = set reference point coordinates
36.4	AVAL_EN	BOOL	FALSE	1 = Set actual value
36.5	FVAL_EN	BOOL	FALSE	1 = set actual value on-the-fly
36.6	ZOFF_EN	BOOL	FALSE	1 = Set zero offset
36.7	CH01CAM_EN	BOOL	FALSE	1 = write cam edge setting (1 cam)
37.0	CH16CAM_EN	BOOL	FALSE	1 = Write setting for fast cam change (16 cams)
Trigger bits for read jobs				
37.1	MDRD_EN	BOOL	FALSE	1 = read machine data
37.2	CAM1RD_EN	BOOL	FALSE	1 = read cam data 1 (cams 0 to 15)
37.3	CAM2RD_EN	BOOL	FALSE	1 = read cam data 2 (cams 16 to 31)
37.4	CAM3RD_EN	BOOL	FALSE	1 = read cam data 3 (cams 32 to 47)
37.5	CAM4RD_EN	BOOL	FALSE	1 = read cam data 4 (cams 48 to 63)
37.6	CAM5RD_EN	BOOL	FALSE	1 = read cam data 5 (cams 64 to 79)

Address	Parameters	Data type	Initial value	Description
37.7	CAM6RD_EN	BOOL	FALSE	1 = read cam data 6 (cams 80 to 95)
38.0	CAM7RD_EN	BOOL	FALSE	1 = read cam data 7 (cams 96 to 111)
38.1	CAM8RD_EN	BOOL	FALSE	1 = read cam data 8 (cams 112 to 127)
38.2	MSRRD_EN	BOOL	FALSE	1 = read measured values
38.3	CNTTRC_EN	BOOL	FALSE	1 = read count values of counter cam tracks
38.4	ACTPOS_EN	BOOL	FALSE	1 = read position and track data
38.5	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
38.6	CAMOUT_EN	BOOL	FALSE	1 = read cam and track data
Done bits for function switches				
40.0	EDGE_D	BOOL	FALSE	1 = "activate edge detection" or "deactivate edge detection" completed
40.1	SIM_D	BOOL	FALSE	1 = "activate simulation" or "deactivate simulation" completed
40.2	MSR_D	BOOL	FALSE	1 = "activate length measurement" or "deactivate length measurement" completed
40.3	REFTR_D	BOOL	FALSE	1 = "activate retrigger reference point" or "deactivate retrigger reference point" completed
40.4	SSW_D	BOOL	FALSE	1 = "Enable software limit switches" or "Disable software limit switches" completed
Done bits for write jobs				
41.0	MDWR_D	BOOL	FALSE	1 = "write machine data" job completed
41.1	MD_D	BOOL	FALSE	1 = "enable machine data" job completed
41.2	AVALREM_D	BOOL	FALSE	1 = "cancel set actual value" or "cancel set actual value on-the-fly" completed
41.3	CAM1WR_D	BOOL	FALSE	1 = "write cam data 1" job completed
41.4	CAM2WR_D	BOOL	FALSE	1 = "write cam data 2" job completed
41.5	CAM3WR_D	BOOL	FALSE	1 = "write cam data 3" job completed
41.6	CAM4WR_D	BOOL	FALSE	1 = "write cam data 4" job completed
41.7	CAM5WR_D	BOOL	FALSE	1 = "write cam data 5" job completed
42.0	CAM6WR_D	BOOL	FALSE	1 = "write cam data 6" job completed
42.1	CAM7WR_D	BOOL	FALSE	1 = "write cam data 7" job completed
42.2	CAM8WR_D	BOOL	FALSE	1 = "write cam data 8" job completed
42.3	REFPT_D	BOOL	FALSE	1 = "set reference point" job completed
42.4	AVAL_D	BOOL	FALSE	1 = "set actual value" job completed
42.5	FVAL_D	BOOL	FALSE	1 = "Set actual value on-the-fly" job completed
42.6	ZOFF_D	BOOL	FALSE	1 = "set zero offset" job completed
42.7	CH01CAM_D	BOOL	FALSE	1 = "change 1 cam" job completed
43.0	CH16CAM_D	BOOL	FALSE	1 = "change 16 cams" completed (fast cam parameter change)
Done bits for read jobs				
43.1	MDRD_D	BOOL	FALSE	1 = "read machine data" job completed

Address	Parameters	Data type	Initial value	Description
43.2	CAM1RD_D	BOOL	FALSE	1 = "read cam data 1" job completed
43.3	CAM2RD_D	BOOL	FALSE	1 = "read cam data 2" job completed
43.4	CAM3RD_D	BOOL	FALSE	1 = "read cam data 3" job completed
43.5	CAM4RD_D	BOOL	FALSE	1 = "read cam data 4" job completed
43.6	CAM5RD_D	BOOL	FALSE	1 = "read cam data 5" job completed
43.7	CAM6RD_D	BOOL	FALSE	1 = "read cam data 6" job completed
44.0	CAM7RD_D	BOOL	FALSE	1 = "read cam data 7" job completed
44.1	CAM8RD_D	BOOL	FALSE	1 = "read cam data 8" job completed
44.2	MSRRD_D	BOOL	FALSE	1 = "read measured values" job completed
44.3	CNTTRC_D	BOOL	FALSE	1 = "read count values of counter cam tracks" job completed
44.4	ACTPOS_D	BOOL	FALSE	1 = "read position and track data" job completed
44.5	ENCVAL_D	BOOL	FALSE	1 = "read actual encoder value" job completed
44.6	CAMOUT_D	BOOL	FALSE	1 = "read position and track data" job completed
Error bits for function switches				
46.0	EDGE_ERR	BOOL	FALSE	1 = error in "activate edge detection" or "deactivate edge detection"
46.1	SIM_ERR	BOOL	FALSE	1 = error in "activate simulation" or "deactivate simulation"
46.2	MSR_ERR	BOOL	FALSE	1 = error in "activate length measurement" or "deactivate length measurement"
46.3	REFTR_ERR	BOOL	FALSE	1 = Error in "Enable retrigger reference point" or "Disable retrigger reference point"
46.4	SSW_ERR	BOOL	FALSE	1 = Error in "Enable software limit switches" or "Disable software limit switches"
Error bits for write jobs				
47.0	MDWR_ERR	BOOL	FALSE	1 = error in "write machine data" job
47.1	MD_ERR	BOOL	FALSE	1 = error in "activate machine data" job
47.2	AVALREM_ERR	BOOL	FALSE	1 = error in "cancel set actual value" or "cancel set actual value on-the-fly"
47.3	CAM1WR_ERR	BOOL	FALSE	1 = error in "write cam data 1" job
47.4	CAM2WR_ERR	BOOL	FALSE	1 = error in "write cam data 2" job
47.5	CAM3WR_ERR	BOOL	FALSE	1 = error in "write cam data 3" job
47.6	CAM4WR_ERR	BOOL	FALSE	1 = error in "write cam data 4" job
47.7	CAM5WR_ERR	BOOL	FALSE	1 = error in "write cam data 5" job
48.0	CAM6WR_ERR	BOOL	FALSE	1 = error in "write cam data 6" job
48.1	CAM7WR_ERR	BOOL	FALSE	1 = error in "write cam data 7" job
48.2	CAM8WR_ERR	BOOL	FALSE	1 = error in "write cam data 8" job
48.3	REFPT_ERR	BOOL	FALSE	1 = error in "set reference point" job
48.4	AVAL_ERR	BOOL	FALSE	1 = error in "set actual value" job
48.5	FVAL_ERR	BOOL	FALSE	1 = error in "set actual value on-the-fly" job
48.6	ZOFF_ERR	BOOL	FALSE	1 = error in "set zero offset" job
48.7	CH01CAM_ERR	BOOL	FALSE	1 = error in "change 1 cam" job

Address	Parameters	Data type	Initial value	Description
49.0	CH16CAM_ERR	BOOL	FALSE	1 = error in "change 16 cams" (fast cam parameter change)
Error bits for read jobs				
49.1	MDRD_ERR	BOOL	FALSE	1 = error in "read machine data" job
49.2	CAM1RD_ERR	BOOL	FALSE	1 = error in "read cam data 1" job
49.3	CAM2RD_ERR	BOOL	FALSE	1 = error in "read cam data 2" job
49.4	CAM3RD_ERR	BOOL	FALSE	1 = error in "read cam data 3" job
49.5	CAM4RD_ERR	BOOL	FALSE	1 = error in "read cam data 4" job
49.6	CAM5RD_ERR	BOOL	FALSE	1 = error in "read cam data 5" job
49.7	CAM6RD_ERR	BOOL	FALSE	1 = error in "read cam data 6" job
50.0	CAM7RD_ERR	BOOL	FALSE	1 = error in "read cam data 7" job
50.1	CAM8RD_ERR	BOOL	FALSE	1 = error in "read cam data 8" job
50.2	MSRRD_ERR	BOOL	FALSE	1 = error in "read measured values" job
50.3	CNTTRC_ERR	BOOL	FALSE	1 = error in "read count values of counter cam tracks" job
50.4	ACTPOS_ERR	BOOL	FALSE	1 = error in "read position and track data" job
50.5	ENCVAL_ERR	BOOL	FALSE	1 = error in "read current encoder value" job
50.6	CAMOUT_ERR	BOOL	FALSE	1 = Error in "Read cam and track data" job
Job management for the instructionCAM_CTRL or CAM_CTRL_452				
52.0	JOB_ERR	INT	0	Communication errors
54.0	JOBBUSY	BOOL	FALSE	1 = At least 1 job active
54.1	JOBRESET	BOOL	FALSE	1 = Reset all error bits and done bits
Job management for the instruction CAM_MSRM_452				
56.0	JOB_ERR_M	INT	0	Communication errors
58.0	JOBBUSY_M	BOOL	FALSE	1 = Job active
Data for "Zero offset" job				
86.0	ZOFF	DINT	L#0	Zero offset
Data for "Set actual value" job				
90.0	AVAL	DINT	L#0	Coordinate for "Set actual value"
Data for "Set actual value on-the-fly" job				
94.0	FVAL	DINT	L#0	Coordinate for "Set actual value on-the-fly"
Data for "Set reference point" job				
98.0	REFPT	DINT	L#0	Coordinate for "Set reference point"
Data for "change cam edges" job				
102.0	CAM_NO	INT	0	Cam number
104.0	CAM_START	DINT	L#0	Cam start
108.0	CAM_END	DINT	L#0	Cam end
Data for the "Length measurement / edge detection" job				
112.0	BEG_VAL	DINT	L#0	Initial value
116.0	END_VAL	DINT	L#0	End value
120.0	LEN_VAL	DINT	L#0	Length
Data for the "read count values" job				
124.0	CNT_TRC0	INT	0	Current count value of counter cam track 0

Address	Parameters	Data type	Initial value	Description
126.0	CNT_TRC1	INT	0	Current count value of counter cam track 1
Data for the "read position and track data" job				
128.0	ACTPOS	DINT	L#0	Current position
132.0	ACTSPD	DINT	L#0	Actual velocity
136.0	TRACK_ID	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31
Data for the "read encoder data" job				
140.0	ENCVAL	DINT	L#0	Encoder value
144.0	ZEROVAL	DINT	L#0	Count value at the last zero mark
148.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment
Data for the "read cam and track data" job				
152.0	CAM_00_31	DWORD	DW#16#0	Cam identifier bits for cams 0 to 31
156.0	CAM_32_63	DWORD	DW#16#0	Cam identifier bits for cams 32 to 63
160.0	CAM_64_95	DWORD	DW#16#0	Cam identifier bits for cams 64 to 95
164.0	CAM_96_127	DWORD	DW#16#0	Cam identifier bits for cams 96 to 127
168.0	TRACK_ID1	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31
172.0	ACTPOS1	DINT	L#0	Current position
Data for the "fast cam parameter change" job				
176.0	C_QTY	BYTE	B#16#0	Number of cams to modify
177.0	DIS_CHECK	BOOL	FALSE	1 = disable data check
180.0	CAM	ARRAY [0...15] STRUCT		Note: The following structure must be completed for each cam to be modified
Relative address				
+0.0	CAM_NO	BYTE	B#16#0	Number of the cam to modify
+1.0	C_EFFDIR	BOOL	FALSE	1 = change the activation direction
+1.1	C_CBEGIN	BOOL	FALSE	1 = change the cam start to the value CBEGIN (new cam start)
+1.2	C_CEND	BOOL	FALSE	1 = change the cam end / activation time to the value CEND (new cam end)
+1.3	C_LTIME	BOOL	FALSE	1 = change the lead time to the LTIME value (new lead time)
+1.4	CAM_OFF	BOOL	FALSE	1 = deactivate the cam during the cam change
+1.5	EFFDIR_P	BOOL	FALSE	1 = new effective direction positive (plus)
+1.6	EFFDIR_M	BOOL	FALSE	1 = new effective direction negative (minus)
+2.0	CBEGIN	DINT	L#0	New cam start
+6.0	CEND	DINT	L#0	New cam end / new activation time
+10.0	LTIME	INT	0	New lead time

Parameter data block (S7-300, S7-400)

Purpose

The machine and cam data is stored in the parameter DB. The parameters can be changed from the user program or from an HMI system. The modified data can be imported to the parameter assignment interface and displayed there. You can export data displayed in the parameter assignment interface to a parameter DB.

A module can contain several parameter data sets (for example, for different recipes) that can be selected from the program.

Configuration

Parameter DB

CAM_P016TYPE (UDT3)
Machine data
Data of cams 0 to 15
CAM_P032TYPE (UDT4)
Machine data
Data of cams 0 to 31
CAM_P064TYPE (UDT5)
Machine data
Data of cams 0 to 63
CAM_P0128TYPE (UDT6)
Machine data
Data of cams 0 to 127

Content

The table below shows the content of the parameter DB.

Note

You may not change any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
Machine data				
3.1	PI_MEND	BOOL	FALSE	• FM 352: 0 • FM 452: 1: Enable hardware interrupt: Full scale value
3.2	PI_CAM	BOOL	FALSE	1: Enable hardware interrupt: Cam on/off

Address	Parameters	Data type	Initial value	Description
3.5	PI_MSTRT	BOOL	FALSE	• FM 352: 0 • FM 452: 1: Enable hardware interrupt: Start of measurement
4.0	EDGEDIST	DINT	L#0	Minimum edge distance for edge detection
8.0	UNITS	DINT	L#1	Physical units system
12.0	AXIS_TYPE	DINT	L#0	0: Linear axis, 1: Rotary axis
16.0	ENDROTAX	DINT	L#100 000	End of rotary axis
20.0	ENC_TYPE	DINT	L#1	Encoder type, frame length
24.0	DISP_REV	DINT	L#80 000	Distance per encoder revolution
32.0	INC_REV	DINT	L#500	Increments per encoder revolution
36.0	NO_REV	DINT	L#1024	Number of encoder revolutions
40.0	BAUDRATE	DINT	L#0	Transmission rate
44.0	REFPT	DINT	L#0	Reference point coordinate
48.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment
52.0	RETR_TYPE	DINT	L#0	Mode for "Retrigger reference point"
59.0	CNT_DIR	DINT	L#0	Count direction: 0: normal, 1: inverted
63.0	MON_WIRE	BOOL	TRUE	1: Wire-break monitoring
63.1	MON_FRAME	BOOL	TRUE	1: Message frame error monitoring
63.2	MON_PULSE	BOOL	TRUE	1: Missing pulse monitoring
64.0	SSW_STRT	DINT	L#-100 000 000	Software limit switch start
68.0	SSW_END	DINT	L#100 000 000	Software limit switch end
76.0	C_QTY	DINT	L#0	Number of cams: 0, 1, 2, 3 = max. 16, 32, 64, 128 cams
80.0	HYS	DINT	L#0	Hysteresis
84.0	SIM_SPD	DINT	L#0	Simulation velocity
90.0	TRACK_OUT	WORD	W#16#0	Control of track outputs: 0 = cam controller, 1 = CPU; Bit number = track number
95.0	EN_IN_I3	BOOL	FALSE	Enable input I3
95.1	EN_IN_I4	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I4
95.2	EN_IN_I5	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I5
95.3	EN_IN_I6	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I6
95.4	EN_IN_I7	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I7

Address	Parameters	Data type	Initial value	Description
95.5	EN_IN_I8	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I8
95.6	EN_IN_I9	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I9
95.7	EN_IN_I10	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I10
99.0	SPEC_TRC0	BOOL	FALSE	1 = track 0 is counter cam track
99.1	SPEC_TRC1	BOOL	FALSE	1 = track 1 is counter cam track
99.2	SPEC_TRC2	BOOL	FALSE	1 = track 2 is brake cam track
100.0	CNT_LIM0	DINT	L#2	High count value for counter cam track 0
104.0	CNT_LIM1	DINT	L#2	High count value for counter cam track 1
Data for cams 0 to 15 / 0 to 31 / 0 to 63 / 0 to 127				
108.0		STRUCT		(12 bytes length per element)
Relative address				
+0.0	CAMVALID	BOOL	FALSE	1: Cam valid
+0.1	EFFDIR_P	BOOL	TRUE	1: Positive effective direction (plus)
+0.2	EFFDIR_M	BOOL	TRUE	1: Negative effective direction (minus)
+0.3	CAM_TYPE	BOOL	FALSE	0: Distance cam, 1: Time-based cam
+0.4	PI_SW_ON	BOOL	FALSE	1: Hardware interrupt on activation
+0.5	PI_SW_OFF	BOOL	FALSE	1: Hardware interrupt on deactivation
+1.0	TRACK_NO	BYTE	B#16#0	Track number
+2.0	CBEGIN	DINT	L#-100 000 000	Cam start
+6.0	CEND	DINT	L#100 000 000	End of output cam / activation time
+10.0	LTIME	INT	0	Lead time

Diagnostics data block (S7-300, S7-400)

Purpose

The diagnostics DB is the storage location for the CAM_DIAG (Page 5934) or CAM_DIAG_452 (Page 5937) instruction. It contains the module's diagnostics buffer which is processed by the instruction.

Configuration

Diagnostics DB
Module address
Internal data
Job status

Diagnostics DB
Trigger bit
Processed diagnostics buffer

Content

The table below shows the content of the diagnostics DB.

Note

You may not change any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
0.0	MOD_ADDR (Enter!)	INT	0	Module address
256.0	JOB_ERR	INT	0	Communication errors
258.0	JOBBUSY	BOOL	FALSE	1 = job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = read diagnostic buffer unconditional
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostic data latest entry
272.0	DIAG[2]	STRUCT		Diagnostic data second entry
282.0	DIAG[3]	STRUCT		Diagnostic data third entry
292.0	DIAG[4]	STRUCT		Diagnostic data oldest entry

Structure of the Diagnostic Entry

The table below shows the structure of a diagnostics entry DIAG[n].

Address	Parame-ters	Data type	Initial value	Description
+0.0	STATE	BOOL	FALSE	0 = outgoing event 1 = incoming event
+0.1	INTF	BOOL	FALSE	1 = Internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 4: Data error 5: Machine data error 7: Cam data error 15: Messages 128: Diagnostics errors
+4.0	FNO	INT	0	Error number 0 ... 255
+6.0	CH_NO	INT	0	Channel number (always 1)
+8.0	CAMNO	INT	0	Cam numbers 0 to 127 with error class = cam data error

Job management (CAM_CTRL) (S7-300, S7-400)

Jobs

Data communication with the module other than the transfer of control and feedback signals is handled by means of jobs.

To initiate a job, set the corresponding trigger bit in the channel DB, including the corresponding data for write jobs. To execute the job, call the CAM_CTRL (Page 5932) instruction.

If operating the module centrally, read jobs are executed immediately. If you use the FM 352 in a distributed configuration, a read job may need several cycles.

Due to the required confirmations from the module, a write job requires at least 3 calls (or OB cycles). If you use the FM 352 in a distributed system, a write job may take more than 3 calls.

You can send several jobs at the same time, if necessary, along with control signals. With the exception of the job for writing the function switches, the jobs are executed in the order of the trigger bits specified in the channel DB. When a job is completed, the trigger bit is reset. The next job pending is identified and executed at the next call of the instruction.

In addition to a trigger bit, all jobs are provided a done bit and an error bit. These are distinguished by their name extension:

Bit	Extension
Trigger bit	_EN ("enable")
Done bit	_D ("done")
Error bit	_ERR ("error")

Done and error bits of the job should be set to 0 after they are evaluated or before this job is initiated.

When you set the JOBRESET bit, all done and error bits are reset before pending jobs are executed. The JOBRESET bit is then reset to 0.

Function switches

The function switches enable and disable module states. A job for writing the function switches is only executed when there is a change to a switch setting. The setting of the function switch is retained after the job has been executed.

Length measurements and edge detection cannot be activated concurrently. For this reason, the CAM_CTRL instruction makes sure that when one of the function switches is enabled, the other is disabled. However, if you enable both function switches (0 → 1), the length measurement is enabled.

Function switches and jobs can be used concurrently at the call of CAM_CTRL.

As with the jobs, the function switches are provided done bits with the name extension _D and error bits with the name extension _ERR.

To be able to evaluate the done and error bits, you should set these bits to 0 when changing a function switch.

Job status

The status of job execution can be read from the return value RETVAL and activity bit JOBBUSY in the channel DB. You can evaluate the status of a single job based on its trigger, done and error bits.

	RETVAL	JOBBUSY	Trigger bit _EN	Done bit _D	Error bit _ERR
Job active	1	1	1	0	0
Job completed without errors	0	0	0	1	0
Job completed with errors	-1	0	0	1	1
Write job cancel-led	-1	0	0	0	1

Job management (CAM_CTRL_452) (S7-300, S7-400)

Jobs

Data communication with the module other than the transfer of control and feedback signals is handled by means of jobs.

To initiate a command, set the corresponding trigger bit in the channel DB, including the corresponding data for write jobs. To execute the command, call the CAM_CTRL_452 (Page 5935) instruction.

A read command is executed immediately. Due to the required confirmations from the module, a write command requires at least 3 calls (or OB cycles).

You can send several jobs at the same time, if necessary, along with control signals. With the exception of the command for writing the function switches, the jobs are executed in the order of the trigger bits specified in the channel DB. When a command is completed, the trigger bit is reset. The next command pending is identified and executed at the next call of the instruction.

In addition to a trigger bit, all jobs are provided a done bit and an error bit. These are distinguished by their name extension:

Bit	Extension
Trigger bit	_EN ("enable")
Done bit	_D ("done")
Error bit	_ERR ("error")

Done and error bits of the command should be set to 0 after they are evaluated or before this command is initiated.

When you set the JOBRESET bit, all done and error bits are reset before pending jobs are executed. The JOBRESET bit is then reset to 0.

Function switches

The function switches enable and disable module states. A command for writing the function switches is only executed when there is a change to a switch setting. The setting of the function switch is retained after the command has been executed.

Length measurements and edge detection cannot be activated concurrently. For this reason, the CAM_CTRL_452 instruction makes sure that when one of the function switches is enabled, the other is disabled. However, if you enable both function switches (0 → 1), the length measurement is enabled.

Function switches and jobs can be used concurrently at the call of CAM_CTRL_452.

As with the jobs, the function switches are provided done bits with the name extension _D and error bits with the name extension _ERR.

To be able to evaluate the done and error bits, you should set these bits to 0 when changing a function switch.

Command status

The status of command execution can be read from the return value RET_VAL and activity bit JOBBUSY in the channel DB. You can evaluate the status of a single command based on its trigger, done and error bits.

	RET_VAL	JOBBUSY	Trigger bit _EN	Done bit _D	Error bit _ERR
Command active	1	1	1	0	0
Command completed without errors	0	0	0	1	0
Command completed with errors	-1	0	0	1	1
Write command cancelled	-1	0	0	0	1

Error classes (S7-300, S7-400)

Error class 1: Operating errors (S7-300, S7-400)

Description

Operating errors are detected asynchronously to an operator input/control.

No.	Meaning	Diagnostics interrupt
1	Software limit switch start overrun	Yes
2	Software limit switch end overrun	Yes
3	Traversing range start overrun	Yes
4	Traversing range end overrun	Yes

No.	Meaning		Diagnostics interrupt
13	Set actual value on-the-fly cannot be executed		Yes
	Cause	After you set the actual value on-the-fly, the software limit switches are outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). The offset derived from the set actual value/set actual value on-the-fly operation is greater than ± 100 m or ± 1000 m.	
	Effect	Axis not synchronized.	

Error class 4: Data error (S7-300, S7-400)

Description

Data errors are detected synchronously to an operator input/control.

No.	Meaning		Diagnostics interrupt
10	Incorrect zero offset		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. After the zero offset, the software limit switches are outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	Incorrect actual value specification		No
	Cause	Linear axis: The coordinate is outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
12	Incorrect reference position		No
	Cause	Linear axis: The coordinate is outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
20	Enable machine data not permitted		No
	Cause	There is no new (error-free) machine data on the module	
21	Set actual value on-the-fly not permitted		No
	Cause	An attempt was made to execute "Set actual value on-the-fly" while "Retrigger reference point" was enabled.	
27	Invalid bit-coded setting		No
	Cause	Unused and, in this case, unwritten bits do not equal 0. An attempt was made to select "length measurement" and "edge detection" at the same time.	
28	Retrigger reference point is not permitted		No
	Cause	An attempt was made to execute "Retrigger reference point" while "Set actual value on-the-fly" was active. An attempt was made to execute "Retrigger reference point" with an SSI encoder.	

No.	Meaning		Diagnostics interrupt
29	**Invalid bit-coded command**		No
	Cause	Unused and, in this case, unwritten bits do not equal 0.	
30	**Incorrect lead time**		No
31	**Incorrect cam number**		No
	Cause	Cam is invalid The cam number is not in the range from 0 to 127.	
32	**Incorrect cam start**		No
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: Cam start is < 0 or greater than the end of the rotary axis.	
33	**Incorrect cam end / cam activation time**		No
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. The cam is not activated at least for the duration of one pulse. With an inverse cam, there are not at least 4 pulses between the cam start and cam end.	
34	**Cancel set actual value not possible**		No
	Cause	The actual position value would be outside the working range with an SSI encoder and linear axis after making the setting.	
35	**Incorrect actual value specified by "Set actual value" / "Set actual value on-the-fly"**		No
	Cause	The specified actual value is outside the valid number range of ±100 m or ±1000 m. If this setting is made, the software limit switches would be outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). The offset derived from set actual value / set actual value on-the-fly would be greater than ±100 m or ±1000 m.	
107	**Axis parameters not assigned**		No
	Cause	No machine data on the axis. No machine data activated on the axis.	
108	**Axis not synchronized**		No
	Cause	One of the settings "Set actual value" or "Set actual value on-the-fly" was initiated although the axis is not synchronized.	
109	**Cam processing active.**		No
110	**Incorrect number of cams to be modified**		No

Error class 5: Machine data error (S7-300, S7-400)

Description

A diagnostics interrupt is only triggered if an error is detected in the system data block (SDB).

No.	Meaning		Diagnostics interrupt
5	**Error in hardware interrupt setting**		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	**Incorrect minimum edge distance**		Yes
	Cause	You have entered a value <0 or >10^9 µm as the minimum edge distance	
8	**Incorrect axis type**		Yes
	Cause	You have specified neither 0, nor 1 as the axis type	
9	**Incorrect end of the rotary axis**		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or from 1 to 10^8 µm (depending on the resolution).	
10	**Incorrect encoder type**		Yes
	Cause	The value for the encoder type is outside the valid range from 1 to 10	
11	**Incorrect distance / encoder revolution**		Yes
	Cause	The value for distance / encoder revolution is outside the valid range from 1 to 10^9 µm (regardless of the resolution).	
13	**Incorrect number of increments / encoder revolution**		Yes
14	**Incorrect number of revolutions**		Yes
15	**Incorrect transmission rate**		Yes
	Cause	You specified a transmission rate outside the valid range from 0 to 3.	
16	**Incorrect reference point coordinate**		Yes
	Cause	The coordinate is outside the range from -100 m to +100 m or -1000 m to +1000 m, depending on the resolution. Linear axis: The coordinate is outside the operating range. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
17	**Incorrect absolute encoder adjustment**		Yes
	Cause	SSI encoder: The value of the absolute encoder adjustment is not in the encoder range: (increments per encoder revolution x number of revolutions - 1).	
18	**Incorrect mode for "Retrigger reference point"**		Yes
	Cause	You specified a value outside the valid quantity of 0, 1, 6 and 7.	
19	**Incorrect direction adaptation**		Yes
	Cause	You have specified a value other than the valid 0 and 1.	
20	**Hardware monitoring not possible**		Yes
	Cause	You set message frame error monitoring = FALSE at the parameter DB. Encoder does not support error pulse monitoring (disable MON_PULSE)	

No.	Meaning		Diagnostics interrupt
21	**Incorrect software limit switch start**		Yes
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	**Incorrect software limit switch end**		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or 1000 m to +1000 m, depending on the resolution) or less than the software limit switch start. Linear axis: The software limit switch end (including any zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
144	**Incorrect quantity structure**		Yes
	Cause	You have specified a quantity structure other than 0 to 3.	
145	**Incorrect hysteresis**		Yes
	Cause	The hysteresis is outside the range 0 to 65535 × resolution. The hysteresis is greater than ¼ × the working range or ¼ × the rotary axis range.	
146	**Incorrect simulation velocity**		Yes
	Cause	The simulation velocity is outside the range 1000 × RES to 3 × 10^7 × RES or is greater than 5 × 10^8 µm/min. The simulation velocity cannot be set internally.	
147	**Incorrect track**		Yes
	Cause	Activation of a track outside 0 to 15 (bits 0 to 15) was selected.	
148	**Incorrect selection of enable inputs**		Yes
	Cause	You wanted to enable a track outside 3 to 10 (bits 0 to 7) using an external signal.	
149	**Incorrect selection of special tracks**		Yes
	Cause	You wanted to define a track outside 0, 1 and 2 (bits 0, 1 and 2) as a special track	
150	**Incorrect high count value track 0**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
151	**Incorrect high count value track 1**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
200	**Incorrect resolution**		Yes
	Cause	You have specified a resolution < 0.1 µm/pulse or >1000 µm/pulse. You specified a distance / encoder revolution and a number of pulses / encoder revolutions, that results in a resolution of < 0.1 or > 1000.	
201	**Encoder does not match the operating range / rotary axis range**		Yes
	Cause	SSI encoder and rotary axis: The encoder does not precisely cover the rotary axis range. Linear axis: The encoder does not cover at least the operating range (including software limit switches).	

Error class 7: Cam data errors (S7-300, S7-400)

Description

A diagnostics interrupt is only triggered if an error is detected in the system data block (SDB).

No.	Meaning		Diagnostics interrupt
1	**Invalid hardware interrupt**		Yes
	Cause	You attempted to define a hardware interrupt for a cam with a cam number > 7.	
2	**Incorrect track number**		Yes
	Cause	The track number is outside the range from 0 to 31	
3	**Incorrect cam start**		Yes
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: Cam start is < 0 or greater than the end of the rotary axis.	
4	**Incorrect cam end**		Yes
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). The cam length is not at least 1 pulse. Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. With an inverse cam, there are not at least 4 pulses between the cam start and cam end.	
5	**Incorrect activation time**		Yes
	Cause	The activation time is < 0 µs. The maximum value depends on the quantity structure.	
6	**Incorrect lead time** [1]		Yes
	Cause	The derivative time is < 0 µs. The maximum value depends on the quantity structure.	
50	**Too many cam records**		Yes
	Cause	You wanted to enter more cam records than is possible with this quantity structure.	
51	**Axis in operation**		Yes
	Cause	You attempted to enter cam records while the cam controller was active.	
52	**Axis parameters not assigned**		Yes
	Cause	You want to enter cam data although no machine parameters have been activated yet.	

[1] The error message can also be generated if you have assigned the parameter "inverted" as the count direction in connection with an absolute encoder (SSI).

Error class 15: Messages (S7-300, S7-400)

No.	Meaning		Diagnostics interrupt
1	Start of parameter assignment		No
	Cause	The module has detected a parameter assignment via a system data block.	
2	End of parameter assignment		No
	Cause	The module has completed processing of the parameter assignment via system data block without errors.	

Error class 128: Diagnostics errors (S7-300, S7-400)

No.	Meaning		Diagnostics interrupt
4	External auxiliary voltage missing		Yes
	Cause	• External 24 V auxiliary voltage is not connected or has failed • Front connector missing • Short-circuit (for example, at the connected encoder)	
	Effect	• Cam processing is disabled • Track outputs is disabled • With incremental encoders, synchronization is canceled • Module parameters not assigned (feedback signal PARA = 0).	
	Remedy	Verify that the 24 V connection is correct (faulty module if 24 V connection is correct).	
5	Front connector missing (FM 452)		Yes
	Cause	Front connector not inserted	
	Effect	• Missing external auxiliary 24 V voltage • Module not ready	
	Remedy	Insert the front connector	
51	Timeout monitoring responded (watchdog)		Yes
	Cause	• Module exposed to severe interference • Module error	
	Effect	• Module is reset • Provided that no module error response is detected after the module reset, the module is ready for operation again. • The module signals the watchdog timeout with "incoming" and "outgoing"	
	Remedy	• Eliminate the interference • Contact the relevant sales department, which will require exact details about the circumstances leading to the error. • Replace the module	

No.	Meaning		Diagnostics interrupt
52	**Internal power supply of the module failed**		Yes
	Cause	Module error	
	Effect	• Module is reset • Provided that no module error response is detected after the module reset, the module is ready for operation again.	
	Remedy	Replace the module	
70	**Hardware interrupt lost**		Yes
	Cause	The module detected a hardware interrupt event that cannot be reported, because the same event has not yet been processed by the user program/ CPU.	
	Effect	• Cam processing is disabled • Track outputs is disabled • With incremental encoders, synchronization is canceled	
	Remedy	• Integrate the hardware interrupt OB in the user program • Check the module's bus connection • Disable the hardware interrupt • Adapt your hardware and software to suit your process requirements (for example, faster CPU, optimize user program).	
144	**Encoder wire break**		Yes
	Cause	• Encoder cable not inserted or sheared off • Encoder has no transverse signals • Incorrect pin assignment • Cable length too long • Short-circuit of the encoder signals	
	Effect	• Cam processing is disabled • Track outputs is disabled • With incremental encoders, synchronization is canceled	
	Remedy	• Check the encoder cable • Comply with the encoder specifications • Monitoring can be temporarily disabled at the responsibility of the operator using the programming interface. • Comply with the technical specifications of the module	

No.	Meaning		Diagnostics interrupt
145	**Absolute encoder message frame error**		Yes
	Cause	Incorrect or interrupted message frame traffic between the module and the absolute encoder (SSI): • Encoder cable not inserted or sheared off • Incorrect encoder type • Incorrect encoder setting (programmable encoders) • Invalid message frame length • Encoder returns incorrect values (faulty encoder) • Interference on the measuring system cable • Selected transmission rate too high	
	Effect	• Cam processing is disabled • Track outputs is disabled • The last valid actual value remains unchanged until the end of the next correct SSI transfer	
	Remedy	• Check the encoder cable • Check the encoder • Check the message frame traffic between the encoder and module	
146	**Incremental encoder missing pulses**		Yes
	Cause	• Encoder monitoring has detected missing pulses • Incorrect entry of the number of increments per encoder revolution • Encoder failure: does not return the specified number of pulses • Incorrect or missing zero mark • Interference on the encoder cable	
	Effect	• Cam processing is disabled • Track outputs is disabled • Cancelation of synchronization	
	Remedy	• Enter the correct number of increments / encoder revolutions • Check the encoder and its cable • Comply with shielding and grounding rules • Monitoring can be temporarily disabled at the responsibility of the operator using the programming interface.	

FM 355 PID Control (S7-300, S7-400)

PID_FM (S7-300, S7-400)

Description PID_FM (S7-300, S7-400)

During operation, the instruction PID_FM is used to:

- Transfer operating parameters to FM 355
- Transfer controlller parameters to FM 355
- Read process values from FM 355

Operating parameters are all in/out parameters that lie between the op_par and cont_par parameters in the instance data block of the instruction.

Controller parameters are all in/out parameters that are located after the cont_par parameter in the instance data block of the instruction.

Process values are all output parameters of the instruction after the out_par parameter.

The required data are stored in an instance DB on the CPU. You need one instance DB for each controller channel used. All the in/out parameters are set to FALSE after an instance DB has been created.

Call

PID_FM is normally called in the cyclic interrupt OB 35 and requires an initialization routine that is started by setting the parameter COM_RST = TRUE in the start-up of the CPU. A calling of the instruction in the start-up OB is possible, but not necessary. After the initialization routine, PID_FM sets the parameter COM_RST to FALSE. If FM 355 is used in distributed I/Os, this can take a few call cycles.

PID_FM has to be called in the same OB as all the other instructions that access the same FM 355.

If you call the instruction PID_FM as a multiple instance DB, no commissioning interface is available. In this case you must commission the FM 355 via a watch table.

Start-up

During CPU start-up and during CPU STOP-RUN transition, the parameters on the FM 355 are overwritten with the parameters from the SDB of the CPU.

Responses in the event of an error

The RET_VALU output parameter contains the 2nd and 3rd bytes of the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated if the READ_VAR and LOAD_PAR parameters are not reset.

When the instruction PID_FM is called, an I/O access error can occur if the FM 355 is not plugged or is not supplied with voltage. In this case the CPU goes into STOP, if no OB 122 is loaded in the CPU.

See also

Mode of operation PID_FM (Page 5963)

Mode of operation PID_FM (S7-300, S7-400)

Effect of parameters LOAD_OP, READ_VAR and LOAD_PAR

In the initialization routine PID_FM reads the parameters from the system data and stores them in its instance DB. Instance DB and system data are synchronized.

The following table shows how the function of the instruction is influenced by the parameters LOAD_OP, READ_VAR and LOAD_PAR.

Parameter	TRUE	FALSE
LOAD_OP	Transfer operating parameters with the instructions WRREC and RDREC.	Transfer operating parameters via direct I/O access in muliplex mode.
	READ_VAR is set to TRUE	READ_VAR is not changed
	Longer run time	Shorter run time
	The values are effective after a cycle of the CPU or FM. The longer cycle is decisive.	A maximum of 3 cycles of the CPU or PM are required until the values are effective. The longer cycle is decisive

Parameter	TRUE	FALSE
READ_VAR	Transfer process values with the instructions WRREC and RDREC.	Process values are transferred via direct I/O accesses in multplex mode.
	Longer processing time	Shorter processing time
	All parameters are read from FM 355	The following parameters are **not** read from FM 355 • SP (setpoint from the FM) • ER (error signal) • DISV (disturbance variable) • LMN_A • LMN_B
	Following parameters are updated at **every** call of the instruction. • PV (process value) • LMN (manipulated value) • The binary displays	Following parameters are updated at every **fourth** call of the instruction. • PV (process value) • LMN (manipulated value) • The binary displays
	At the start-up of the CPU, the following values are applied from the FM 355 to the PID_FM. • SP_OP • LMN_OP • SP_OP_ON • LMNOP_ON If these values were changed on the CPU, they are overwritten with the values from the FM 355 at the next start-up of the CPU	At the start-up of the CPU, the following values are read from the instruction, not from the FM: • SP_OP • LMN_OP • SP_OP_ON • LMNOP_ON
LOAD_PAR	All controller parameters are downloaded from the CPU to the FM 355. This is practical if the controller parameters are to be changed depending on process states.	No transfer of the controller parameters

After data transfer as been completed the parameters LOAD_OP, READ_VAR and LOAD_PAR are reset to FALSE.

Use in distributed I/O

When the FM 355 is used in distributed I/O, the data transfer and the resetting of the parameters can take a few call cycles.

Special features during transfer in multiplex mode

Multiplexing of the data to be transferred during access to the FM 355 via direct I/O accesses is controlled via the PID_FM. This multiplex controlling via PID_FM does not function if two instances of the instruction PID_FM access the same channel number of a module. This results in incorrect parameters in the FM 355 (for example, setpoint and manual manipulated value) and incorrect displays at the output parameters of the instruction PID_FM.

Saving the Parameters in EEPROM

During program controlled re-configuring (LOAD_PAR, LOAD_OP) of the controller module by the instruction PID_FM, its run time is extended. The new parameters are always immediately effective. Controller parameters and the operating parameter SP_ER are stored additionally in a nonvolatile memory (EEPROM). After each saving of the parameters in the EEPROM, a renewed saving is delayed by 30 minutes, since the life of the EEPROM is limited by the number of write processes. It is possible to save the new parameter in the EEPROM as soon as the power supply returns. The selection of the parameters determines whether the reconfiguration of the controller module by the instruction PID_FM is executed without bumps.

Processing times PID_FM

Boundary conditions			Processing time in	
READ_VAR	LOAD_OP	LOAD_PAR	CPU 314	CPU 414-2DP
FALSE	FALSE	FALSE	0.65 ms	0.077 ms
TRUE	FALSE	FALSE	2.85 ms	2.36 ms
TRUE	TRUE	FALSE	4.56 ms	4.48 ms
FALSE	FALSE	TRUE	3.75 ms	2.59 ms
TRUE	FALSE	TRUE	5.95 ms	5.15 ms
TRUE	TRUE	TRUE	7.66 ms	7.1 ms

See also

Description PID_FM (Page 5962)

Input parameters PID_FM (S7-300, S7-400)

Table 4-227

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	1	The number of the controller channel to which the instance DB îs referenced is configured at the "Channel number" input. Values of 1 to 4 are permissible.
PHASE	4.0	INT	0	The parameter PHASE can be interconnected with the output parameter PHASE of the instructions TUN_EC or TUN_ES (PID Self-Tuner). The phase state of the PID Self-Tuner can then be displayed in plain text in the "Commissioning" dialog. This parameter is not relevant for the OP.

See also

Description PID_FM (Page 5962)

Output parameters PID_FM (S7-300, S7-400)

Table 4-228

Parameters	Address	Data type	Default	Description
RET_VALU	6.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.
out_par	8.0	WORD	W#16#3130	The out_par parameter may not be modified by users. It marks the start of the output parameter that is read by the module if READ_VAR = TRUE is set.
SP	10.0	REAL	0.0	The setpoint that is currently in effect is available at the "Setpoint" output.
PV	14.0	REAL	0.0	The process value that becomes effective is issued on the "Process value" output.
ER	18.0	REAL	0.0	The control deviation that becomes effective is issued on the "Control deviation" output.
DISV	22.0	REAL	0.0	The interference that becomes effective is issued on the "Interference" output. Values from -100 to 100 % are permitted.
LMN	26.0	REAL	0.0	The manipulated value that becomes effective is issued on the "Manipulated value" output. Step controllers without analog position feedback output the unlimited P- + D-action at the LMN parameter. Values from -100 to 100 % are permitted.
LMN_A	30.0	REAL	0.0	At the output "Manipulated value A of the split range function / position feedback", continuous controllers display the manipulated value A and step controllers with analog position feedback display the position feedback. The output LMN_A can only be used for an approximate display of a corresponding simulated manipulated variable. The start value LMNRSVAL of the simulated position feedback has to be configured accordingly and becomes effective when LMNRS_ON is set. Values from -100 to 100 % are permitted.
LMN_B	34.0	REAL	0.0	Continuous controllers display the manipulated value B of the split-range function at the output “manipulated value B of the split-range function”. Values from -100 to 100 % are permitted.
QH_ALM	38.0	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_ALM limit is reported at the output "High limit alarm triggered".
QH_WRN	38.1	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_WRN limit is reported at the output "High limit warning triggered".
QL_WRN	38.2	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_WRN limit is reported at output "Low limit warning triggered".
QL_ALM	38.3	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_ALM limit is reported at output "Low limit alarm triggered".
QLMN_HLM	38.4	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the high limit is reported at output "High limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QLMN_LLM	38.5	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the low limit is reported at output "Low limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QPARA_F	38.6	BOOL	FALSE	The module checks the validity of the parameters. A parameter assignment error is displayed at the "Parameter assignment error" output. These parameter assignment errors can also be read out in the module's "Online & Diagnostics" dialog.

Parameters	Address	Data type	Default	Description
QCH_F	38.7	BOOL	FALSE	The "Channel error" output is set if the controller channel cannot provide valid results. A channel error (for example, wire break) is also set if QPARA_F = 1 or QMOD_F = 1. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.
QUPRLM	39.0	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of positive setpoint slew rate triggered" is set, the positive setpoint slew rate is limited.
QDNRLM	39.1	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of negative setpoint slew rate triggered" is set, the setpoint fall is limited.
QSP_HLM	39.2	BOOL	FALSE	The setpoint is always limited to a high and a low limit. The output "High limit of setpoint value triggered" indicates that the high limit has been exceeded.
QSP_LLM	39.3	BOOL	FALSE	The setpoint is always limited to a high and a low limit. Violation of the low limit is reported at output "Setpoint low limit triggered".
QLMNUP	39.4	BOOL	FALSE	This is the "Manipulated value signal up" output. (Only in the case of step controllers or pulse controllers)
QLMNDN	39.5	BOOL	FALSE	This is the "Manipulated value signal down" output. (Only in the case of step controllers or pulse controllers)
QID	39.6	BOOL	FALSE	QID = TRUE indicates that an identification is running (not that it is enabled). On completion of the identification, the identification result can be read out from the IDSTATUS parameter of the instruction CH_DIAG.
QSPOPON	40.0	BOOL	FALSE	Output "Setpoint operation on" indicates whether the operator controls the setpoint using the instruction TUN_EC or TUN_ES. If the bit is set, the value SP_OP is applied as setpoint.
QLMNSAFE	40.1	BOOL	FALSE	If the output "Safety mode" is set, the safety manipulated value is output as the manipulated value.
QLMNO-PON	40.2	BOOL	FALSE	Output "Setpoint operation on" indicates whether the operator controls the manipulated value using the instruction TUN_EC or TUN_ES. If the bit is set, the value LMN_OP is applied as manipulated value.
QLMNTRK	40.3	BOOL	FALSE	The output "Follow-up mode" indicates if the manipulated value is being adjusted via an analog input.
QLMN_RE	40.4	BOOL	FALSE	The output "Manual = 1; Automatic = 0" indicates whether or not the manipulated value is set on the external manipulated value LMN_RE (manual = 1).
QLMNR_HS	40.5	BOOL	FALSE	The output "High endstop signal of position feedback" indicates if the control valve is at its high endstop. QLMNR_HS = TRUE means: The control valve is at high endstop. (For step controllers only)
QLMNR_LS	40.6	BOOL	FALSE	The output "Low endstop signal of position feedback" indicates whether the control valve is at its low endstop. QLMNR_LS = TRUE means: The control valve is at its low endstop. (For step controllers only)
QLMNR_ON	40.7	BOOL	FALSE	Output "Position feedback enabled" indicates the set operating mode "Step controller with position feedback" or "Step controller without position feedback".
QFUZZY	41.0	BOOL	FALSE	If the output parameter is QFUZZY =1, the controller operates with the fuzzy algorithm.
QSPLEPV	41.1	BOOL	FALSE	The output "Display of FUZZY controller: Setpoint < process value" is set when the fuzzy controller is switched on, if the setpoint is less than the effective process value.
QSPR	41.2	BOOL	FALSE	If the output "Split-range operation" is set, the continuous controller is operating in split-range mode.

Parameters	Address	Data type	Default	Description
QMAN_FC	41.4	BOOL	FALSE	The output "QMAN_FC" is set in the following two cases: The slave controller is in manual mode and the master controller is followed up to the process value of the slave controller. The I-action of the master controller is stopped because the setpoint or manipulated variable of the slave controller is limited or because the slave controller is in manual mode.
QPARA-BUB	41.7	BOOL	FALSE	This parameter is set by the FM if the operating parameters are changed via the OP. If READ_VAR = TRUE and if this display is set by the FM, the instruction PID_FM reads out the parameters SP_OP_ON, LMNOP_ON, SP_OP and LMN_OP from the FM and saves them in the instance DB. The instruction thus applies over the operating state of the FM. After the reading process the parameter is set to FALSE.
QMOD_F	42.0	BOOL	FALSE	The instruction checks that the reading and writing of a data record has been completed. Output "Module error" is set if errors are found. The error cause can be: An incorrect module address at parameter MOD_ADDR, or an incorrect channel number at parameter CHANNEL, or a faulty module.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

In/out parameters PID_FM (S7-300, S7-400)

Table 4-229

Parameters	Address	Data type	Default	Description
[1] Default values of the module after the first start-up of the PID_FM with COM_RST = TRUE				
COM_RST	44.0	BOOL	FALSE	If the parameter COM_RST = TRUE, the instruction PID_FM executes an initialization routine. In doing so the controller parameters (these are all the parameters after cont_par) are read from the FM and stored in the instance DB. In addition, the parameters MOD_ADDR and CHANNEL are checked for validity. After the initialization routine the parameter is set to FALSE.
LOAD_OP	44.1	BOOL	FALSE	If the in/out parameter "Load operating parameters to FM 355" is set, the operating parameters are loaded into the module and the in/out parameter is reset.
READ_VAR	44.2	BOOL	FALSE	If the in/out parameter "Read tags from FM 355" is set, the output parameters are read from the module and the in/out parameter is reset.
LOAD_PAR	44.3	BOOL	FALSE	If the in/out parameter "Load control parameters to FM 355" is set, the control parameters are loaded into the module and the in/out parameter is reset.
op_par	46.0	WORD	W#16#3130[1]	The op_par parameter may not be modified by users. It identifies the start of the operating parameters that are transferred to the module, if LOAD_OP = TRUE is set. The end of the operating parameters is identified by cont_par.
SP_RE	48.0	REAL	0.0	An external setpoint is interconnected with the controller at the "External setpoint" input.

Parameters	Address	Data type	Default	Description
LMN_RE	52.0	REAL	0.0	An external manipulated value is interconnected to the controller at the input "External manipulated value". Values from -100.0 to 100.0 (%) are permitted.
SP_OP_ON	56.0	BOOL	FALSE	The instructions of PID Self-Tuner have access to the in/out parameter "Enable setpoint operation". If the bit is set, the value SP_OP is applied as setpoint.
SAFE_ON	56.1	BOOL	FALSE	If the "Adopt safety position" input is set, a safety value is applied as manipulated value. Note: The actuating signal processing via LMNDN_OP, LMNUP_OP and LMNSOPON on a step controller is of higher priority than the safety manipulated value.
LMNOP_ON	56.2	BOOL	FALSE	The instructions of PID Self-Tuner have access to the in/out parameter "Enable manipulated value operation". If the bit is set, the value LMN_OP is applied as manipulated value.
LMNTRKON	56.3	BOOL	FALSE	If input "Follow-up (LMN via AI)" is set, the manipulated value is adjusted to an analog input (AI). (Does not apply to step controllers without analog position feedback)
LMN_REON	56.4	BOOL	FALSE	If input "Enable external manipulated value" is set, the external manipulated value LMN_RE is set as the manipulated value.
LMNRHSRE	56.5	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected at the "High endstop signal of position feedback" input. LMNRHSRE = TRUE means: The control valve is located at the high endstop. (Only in the case of step controllers)
LMNRLSRE	56.6	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected at the "Low endstop signal of position feedback" input. LMNRLSRE = TRUE means: The control valve is located at the low endstop. (Only in the case of step controllers)
LMNSOPON	56.7	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Enable manipulated value operation". If the bit is set, the signals LMNUP_OP and LMNDN_OP are applied as manipulated value signals. (Only in the case of step controllers)
LMNUP_OP	57.0	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Manipulated value signal up". If LMNSOPON is set, the value at input "Manipulated value signal up operation" is set as the manipulated value signal. (Only in the case of step controllers)
LMNDN_OP	57.1	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Manipulated value signal down operation". If LMNSOPON is set, the value at input "Manipulated value signal down operation" is set as the manipulated value signal. (Only in the case of step controllers)
LMNRS_ON	57.3	BOOL	FALSE	You can simulate no position feedback if this is not available. The function is enabled by setting the "Enable simulation of position feedback" input The instructions of PID Self-Tuner have access to this parameter as well since at least one simulated manipulated value is required for optimization if a step controller was configured without position feedback. The simulated value is displayed at parameter LMN_A. When the simulation is enabled the value of the parameter LMNRSVAL is set as the starting value. CAUTION: Offset between the simulation value and real position feedback increases over the time. (Only for step controllers without analog position feedback)

Parameters	Address	Data type	Default	Description
FUZID_ON	57.4	BOOL	FALSE	The identification of the fuzzy algorithm is enabled at the "Enable fuzzy identification" input.
SP_OP	58.0	REAL	0.0	The instructions of PID Self-Tuner have access to the in/out parameter "Setpoint operation". If the SP_OP_ON bit is set, the "Setpoint operation" value is applied as the setpoint.
LMN_OP	62.0	REAL	0.0	The instructions of PID Self-Tuner have access to the in/out parameter "Manipulated value operation". If the LMNOP_ON bit is set, the "Manipulated value operation" value is applied as the manipulated value. Values from -100.0 to 100.0 (%) are permitted
LMNRSVAL	66.0	REAL	0.0	The instructions of PID Self-Tuner have access to input "Starting value of simulated position feedback". The start value of the simulation is entered at the parameter. (Only for step controllers without analog position feedback) Values from -100.0 to 100.0 (%) are permitted
cont_par	70.0	WORD	W#16#3130 [1]	The cont_par parameter may not be modified by users. It indicates the start of the controller parameters which are read from the FM and stored in the instance DB, if COM_RST = TRUE is, and which are transferred to the FM when LOAD_PAR = TRUE is set. The end of the controller parameter is the end of the instance DB.
P_SEL	72.0	BOOL	TRUE [1]	In the PID algorithm, the PID parts can be switched on and off individually. The proportional action is enabled when the "P-action on" input is set.
PFDB_SEL	72.1	BOOL	FALSE [1]	In the PID algorithm, the P- and D-actions can be included in the feedback loop. The proportional action is located in the feedback loop when the input "Enable P-action in feedback loop" is set.
MONER-SEL	72.2	BOOL	FALSE [1]	The controller has a limit indicator that can be applied either for the process value or the control deviation. If the input "Monitoring: Process value = 0, control deviation = 1" is set, the control deviation will be monitored.
D_EL_SEL	74.0	INT	0 [1]	The D-action element in the PID algorithm can be positioned at a separate input. This is selected at the input "D-action element input". 0: Control deviation 1 to 4: Analog input 1 to 4 17: Negative process value, D-action in the feedback
SP_HLM	76.0	REAL	100.0 [1]	The setpoint is always limited to a high and a low limit. The "Setpoint high limit" input specifies the high limit. SP_HLM > SP_LLM
SP_LLM	80.0	REAL	0.0 [1]	The setpoint is always limited to a high and a low limit. The "Setpoint low limit" input specifies the low limit. SP_LLM < SP_HLM
H_ALM	84.0	REAL	100.0 [1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "High limit interrupt" input specifies the highest limit. H_ALM > H_WRN
H_WRN	88.0	REAL	90.0 [1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "High limit warning" input specifies the second highest limit. Values from H_ALM to L_WRN are permitted.
L_WRN	92.0	REAL	10.0 [1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "Low limit warning" input specifies the second lowest limit. Values from H_WRN to L_ALM are permitted.

Parameters	Address	Data type	Default	Description
L_ALM	96.0	REAL	0.0 1)	You can assign parameters for four limits for monitoring the process value or the control deviation. The "Low limit alarm" input specifies the lowest limit. L_ALM < L_WRN
HYS	100.0	REAL	1.0 1)	To prevent flickering of the monitoring displays, a hysteresis can be configured at the "Hysteresis" input. HYS ≥ 0.0
DEADB_W	104.0	REAL	0.0 1)	A deadband is applied to the control deviation. The "Deadband width" input determines the size of the deadband. DEADB_W ≥ 0.0
GAIN	108.0	REAL	1.0 1)	The "Proportional gain" input specifies controller amplification.
TI	112.0	REAL	3000.0 1)	The "Integration time" input determines the time response of the integral action. If TI = 0, the integral action is disabled TI = 0.0 or TI ≥ 0.5
TD	116.0	REAL	0.0 1)	The "Derivative-action time" input determines the time response of the differentiator. If TD = 0, the derivative action is disabled. TD = 0.0 or TD ≥ 1.0
TM_LAG	120.0	REAL	5.0 1)	The algorithm of the derivative action contains a delay for which parameters can be assigned at the input "Time lag of the derivative action". TM_LAG ≥ 0.5
LMN_SAFE	124.0	REAL	0.0 1)	For the manipulated value, a safety value can be configured at the "Safety manipulated value" input. Values from -100.0 to 100.0 (%) are permitted.
LMN_HLM	128.0	REAL	100.0 1)	The manipulated value is always limited to a high and a low limit. The "Manipulated value high limit" input specifies the high limit. (Does not apply to step controllers without analog position feedback) Values from LMN_LLM to 100.0 (%) are permitted.
LMN_LLM	132.0	REAL	0.0 1)	The manipulated value is always limited to a high and a low limit. The "Manipulated value low limit" input specifies the low limit. (Does not apply to step controllers without analog position feedback) Values from -100.0 to LMN_HLM (%) are permitted.
MTR_TM	136.0	REAL	60.0 1)	The runtime from endstop to endstop of the control valve is entered at the "Motor actuating time" parameter. (Only in the case of step controllers) Values of MTR_TM ≥ 0.001 are permitted.
PULSE_TM	140.0	REAL	0.2 1)	A minimum pulse time can be configured at the "Minimum pulse time" parameter. (Applies to step controllers or pulse controllers only) Values of PULSE_TM ≥ 0.0 are permitted.
BREAK_TM	144.0	REAL	0.2 1)	A minimum pulse break duration can be assigned with the parameter "Minimum break time." (Applies to step controllers or pulse controllers only) Values of BREAK_TM ≥ 0.0 are permitted.

Note

If LOAD_PAR = TRUE is set, all the control parameters are loaded permanently to the EEPROM of the FM 355.

If LOAD_OP = TRUE, only the setpoint SP_RE of the operating parameters is loaded permanently to the EEPROM of the FM 355. All the other operating parameters are assigned the values 0 or FALSE as default during the startup of FM 355

The EEPROM of the module could become damaged by too frequent write operations. To prevent this, the module delays another write process by 30 minutes after writing of the EEPROM.

See also

Description PID_FM (Page 5962)

Block diagram PID_FM (S7-300, S7-400)

Overview

The following figures show at which points the parameters of the instruction PID_FM act.

The parameters are effective with three-component controllers, ration/blending controllers, and fixed setpoint or cascade controllers. This also applies for the parameters that exist equally at continuous controllers, at controllers with a pulse output as well as at step controllers. Therefore, not all block diagrams are depicted and not all parameter are marked in all figures.

However, the parameters of the instruction PID_FM are contained in all the figures – with the exception of the parameters MOD_ADDR, CHANNEL, QMOD_F, QPARA_F, QCH_F, QLMNR_ON, RET_VALU, COM_RST, LOAD_PAR, READ_VAR, LOAD_OP.

Action of the input parameters

Error signal generation with fixed setpoint or cascade controllers

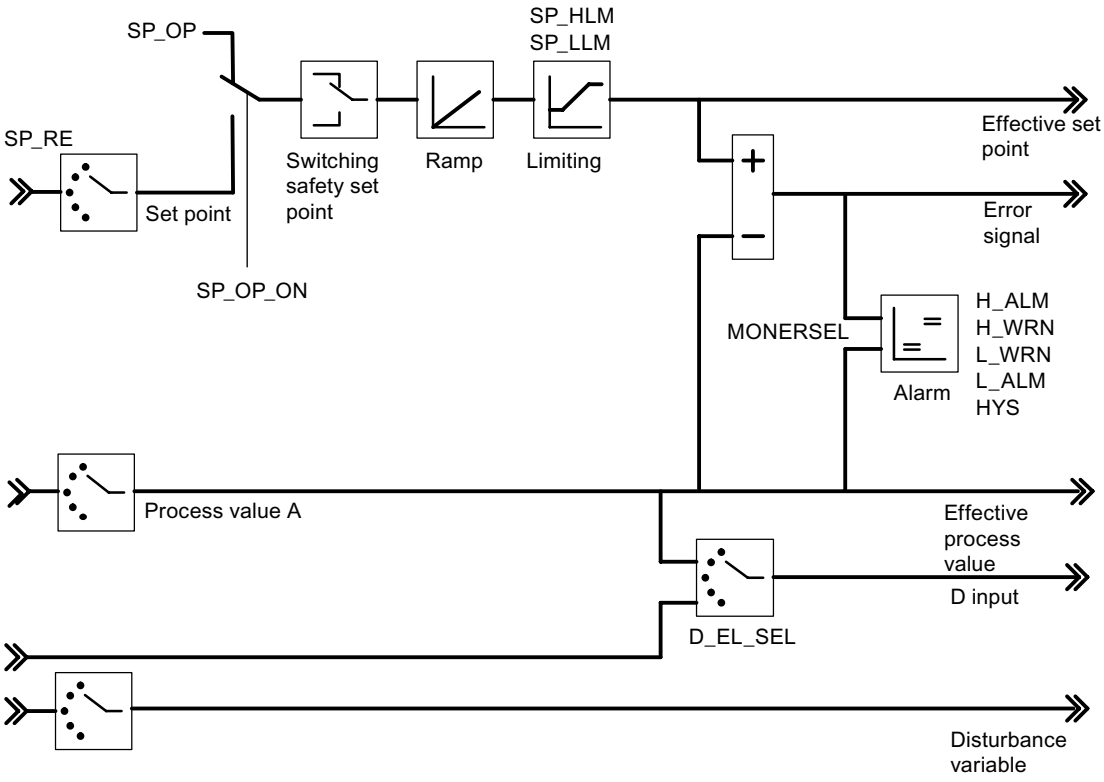

Block diagram of the control algorithm

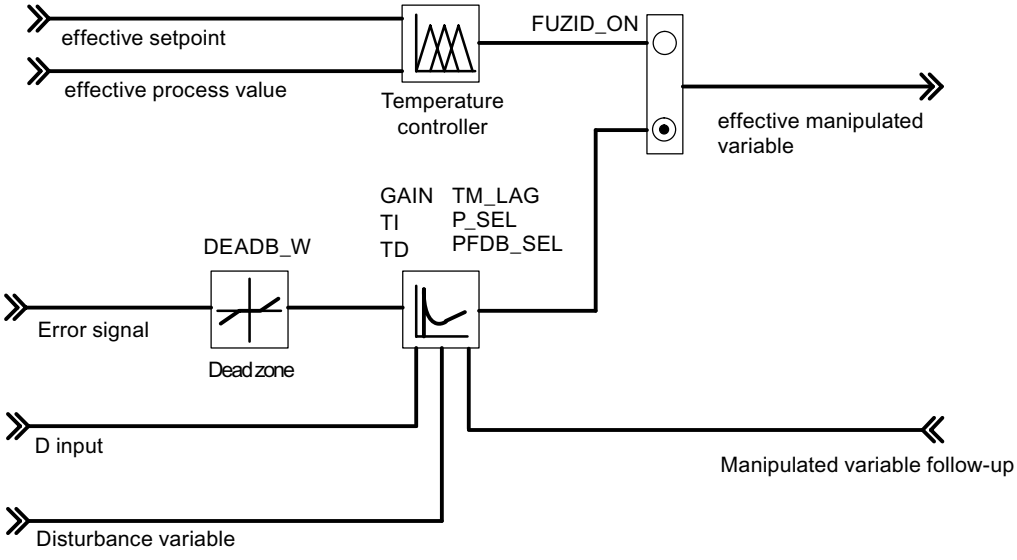

Controller output of the continuous controller

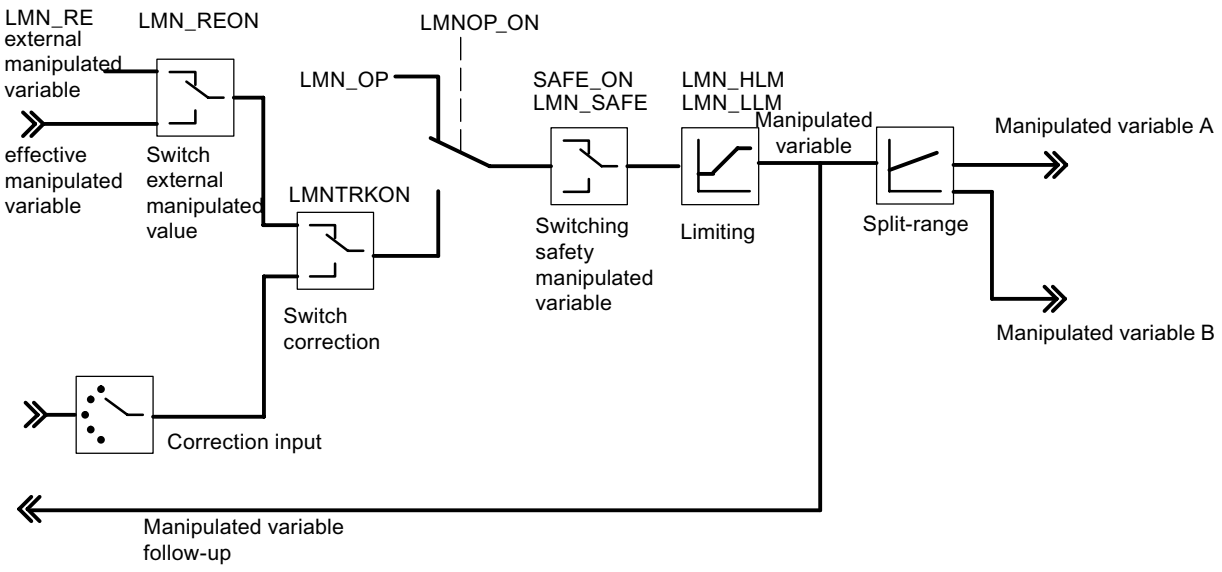

Controller output of the step controller (pulse controller operating mode)

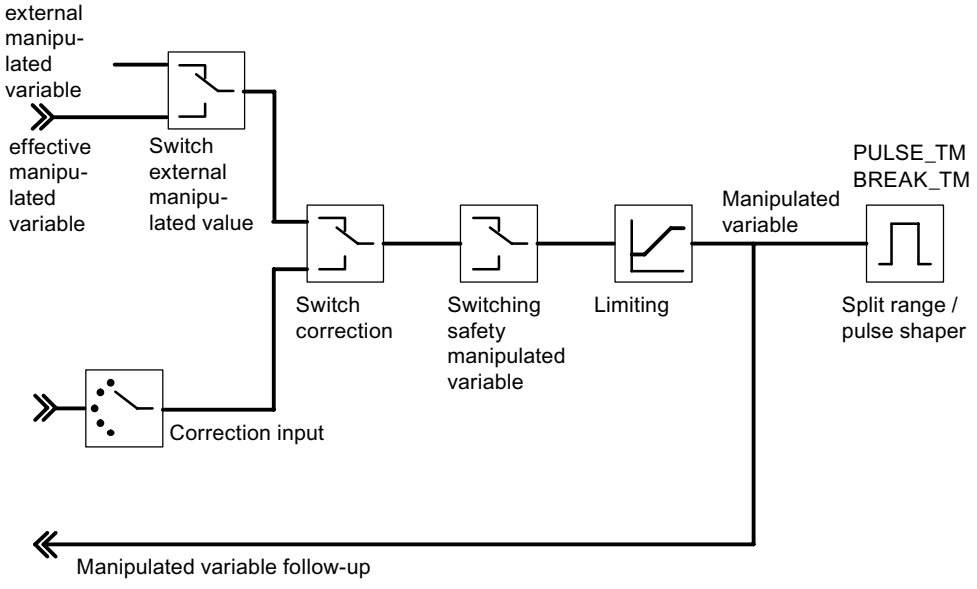

Controller output of the step controller (step controller operating mode with position feedback)

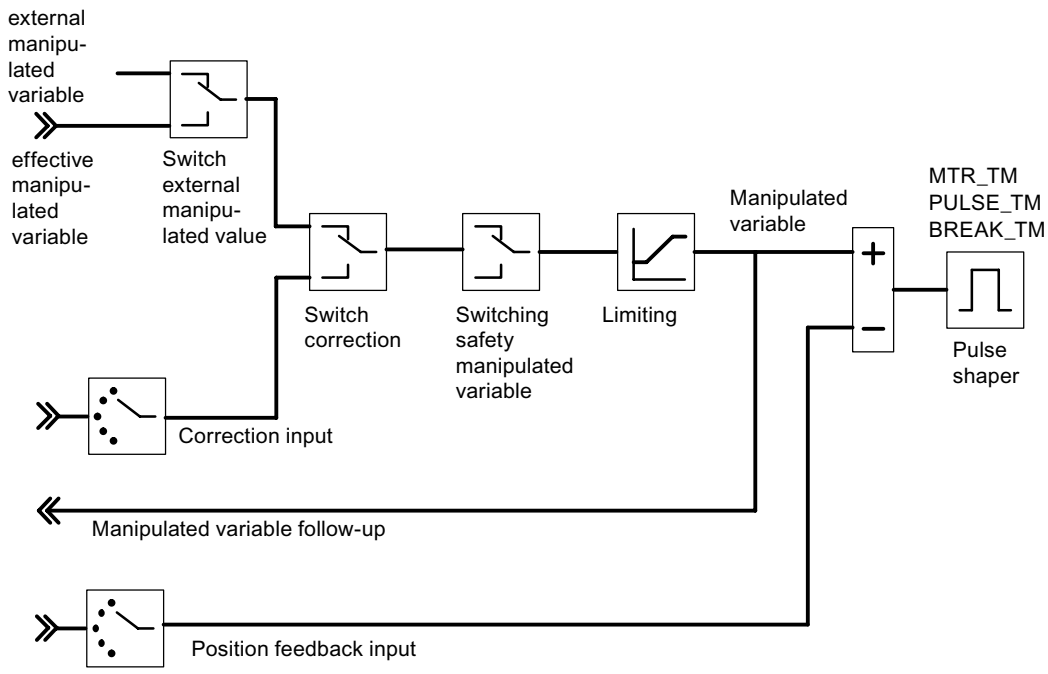

Controller output of the step controller (step controller operating mode without position feedback)

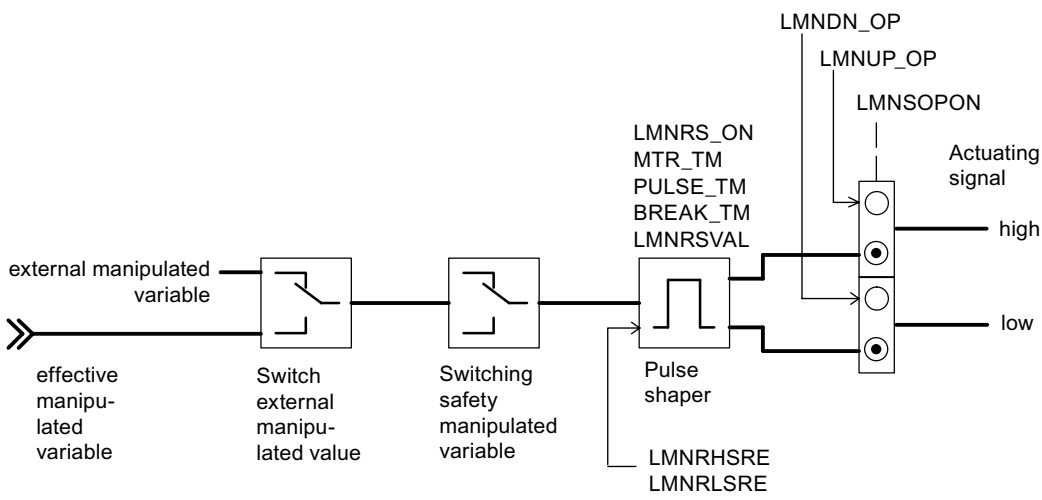

Generating output parameters

The following figures show at which points in the module the output parameters of the instruction PID_FMare generated.

Error signal generation with fixed setpoint or cascade controllers

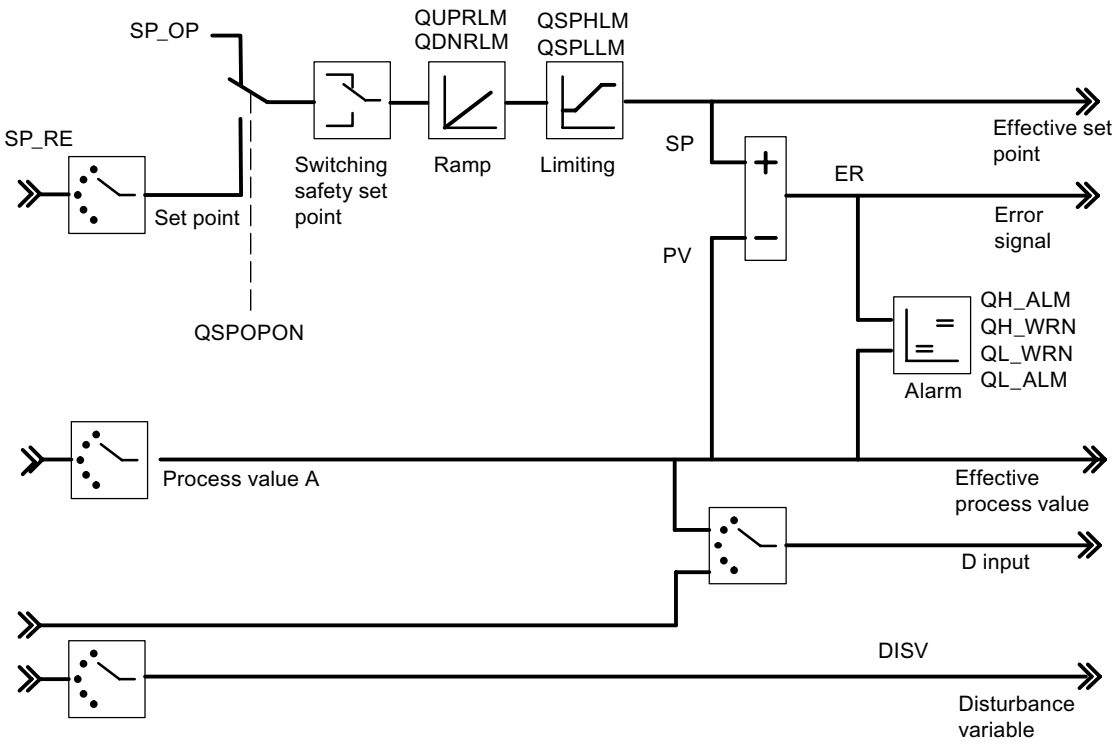

Block diagram of the control algorithm

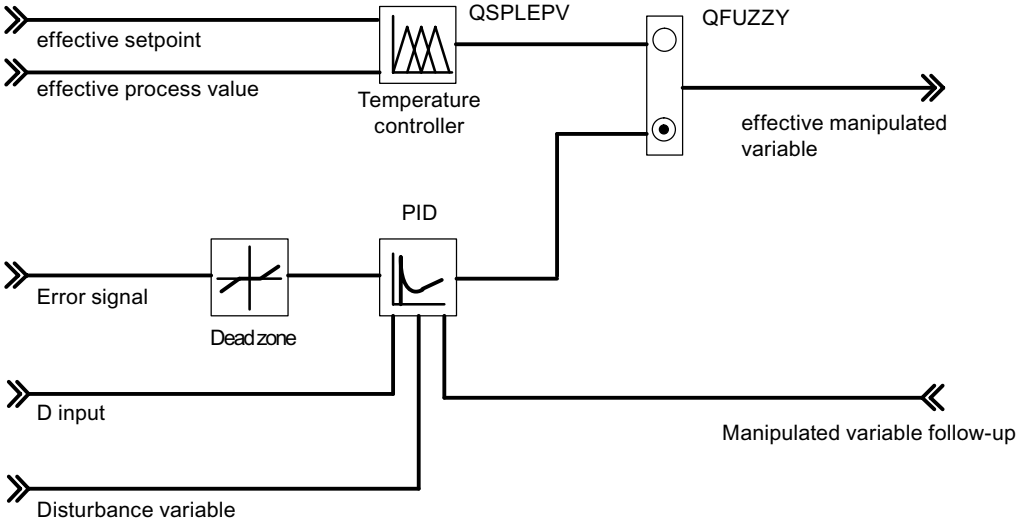

Controller output of the continuous controller

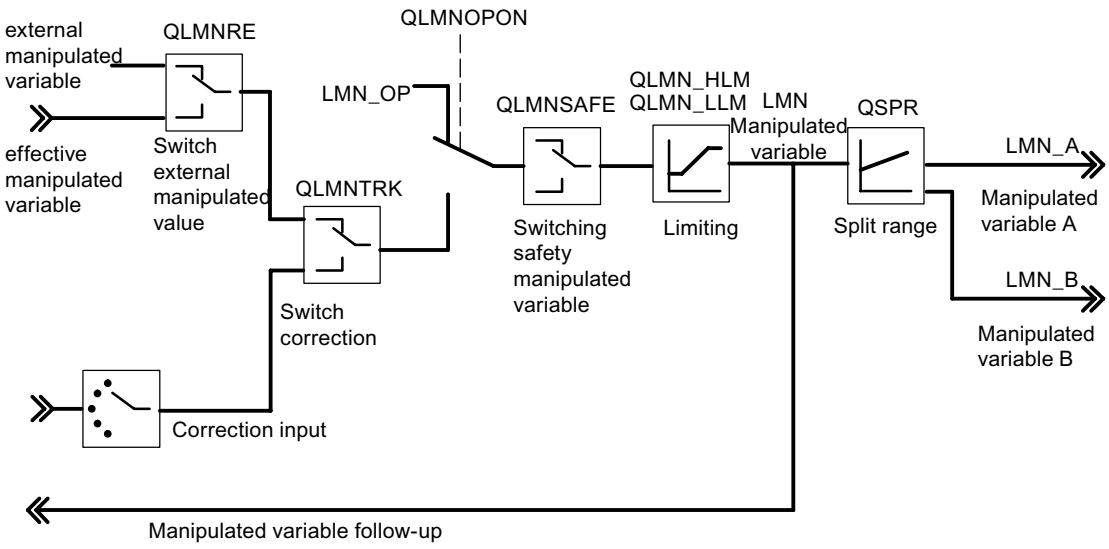

Controller output of the step controller (pulse controller operating mode)

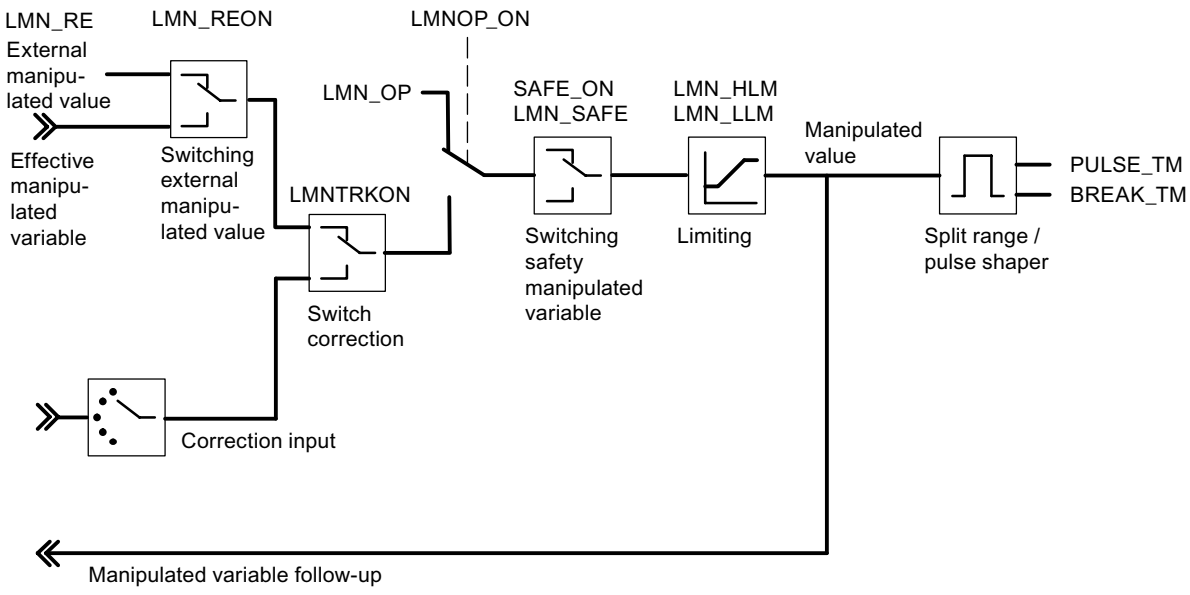

Controller output of the step controller (step controller operating mode with position feedback)

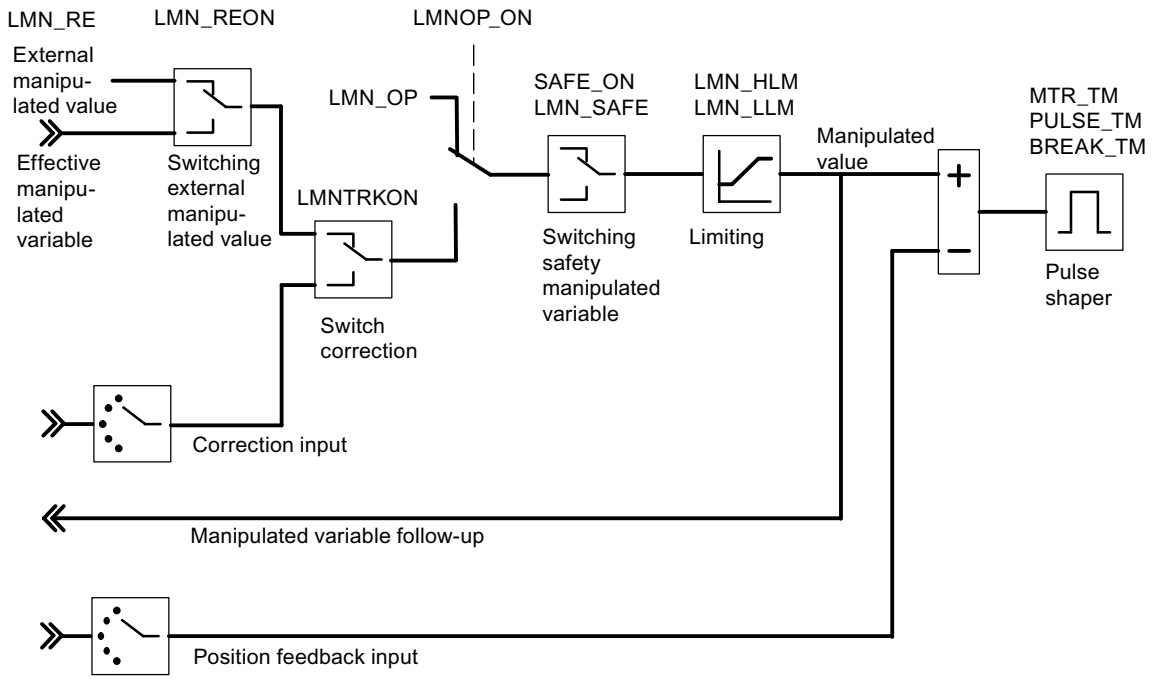

Controller output of the step controller (step controller operating mode without position feedback)

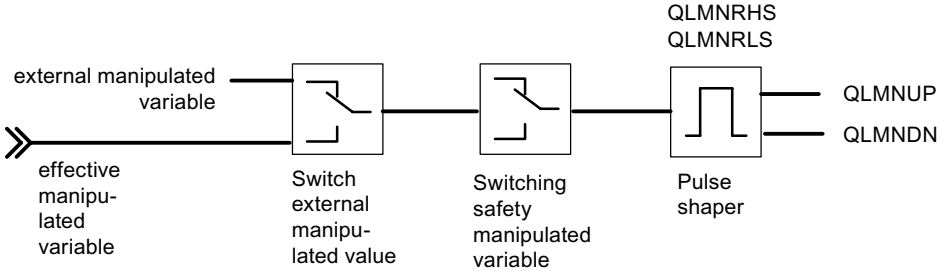

See also

Description PID_FM (Page 5962)

FUZ_355 (S7-300, S7-400)

Description FUZ_355 (S7-300, S7-400)

The instruction FUZ_355 is used to read out and write the controller parameters of the fuzzy temperature controller from FM 355.

This function is suitable for the following applications:

- Transferring the controller parameters of the FM 355 that have been established by identification after the module replacement
- Adapting the FM 355 to different processes

Note

It is not permitted to modify the parameters that have been established by the FM 355 by means of an identification as they have been optimized for the process.

Mode of operation

The parameters READ_PAR and LOAD_PAR are used to control the mode of operation of the instruction FUZ_355

If READ_PAR = TRUE, FUZ_355 reads the parameters of all the temperature controllers of the FM 355 and stores them in the instance DB. After the temperature controller parameters have been successfully read out, FUZ_355 sets the parameter READ_PAR to FALSE.

If LOAD_PAR = TRUE, FUZ_355 writes the parameters of all the temperature controllers of the FM 355 from the instance DB to the FM 355. After the parameters have been successfully transferred, FUZ_355 sets the parameter LOAD_PAR to FALSE.

Call

FUZ_355 requires no initialization routine.

If the FM 355 is used in distributed I/Os, it may take a few call cycles until the parameters READ_PAR and LOAD_PAR are reset to FALSE. Therefore, call the instruction FUZ_355 conditionally, as long as READ_PAR = TRUE or LOAD_PAR = TRUE.

FUZ_355 has to be called in the same OB as all the other instructions that access the same FM 355.

Start-up

During the start-up of the CPU you should set the LOAD_PAR parameter of the FUZ_355 instruction and then call the block in the cyclic program, provided LOAD_PAR = TRUE.

Input parameters FUZ_355 (S7-300, S7-400)

Table 4-230

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters FUZ_355 (S7-300, S7-400)

Table 4-231

Parameters	Address	Data type	Default	Description
RET_VALU	2.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.
PARAFFUZ	4.0	WORD	0	A parameter assignment error created by the the instruction FUZ_355 is displayed at the parameter PARAFFUZ as follows: High byte of PARAFFUZ = 01: There is a parameter assignment error.. High byte of PARAFFUZ = 00: There is no parameter assignment error. The low byte contains the offset of the parameter that caused the parameter assignment error, calculated from the static variable FUZ_PAR[1]. For instance, PARAFFUZ = W#16#0104 means that the second parameter is faulty.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

In/out parameters FUZ_355 (S7-300, S7-400)

Table 4-232

Parameters	Address	Data type	Default	Description
READ_PAR	6.0	BOOL	FALSE	If the READ_PAR parameter is set, the fuzzy parameters are read out from the module and stored in the static variables of the instance DB.
LOAD_PAR	6.1	BOOL	FALSE	If the LOAD_PAR parameter is set, the fuzzy parameters are read out from the static variables of the instance DB module and transferred to the module.

FORCE355 (S7-300, S7-400)

Description FORCE355 (S7-300, S7-400)

The instruction FORCE355 is used to simulate (force) the analog and digital input values to support the start-up.

Simulating Digital Values

The simulation of the values for the digital inputs 1 to 8 is enabled via the switches S_DION[i], where 1 ≤ i ≤ 8. You specify the simulation values at the parameters DI_SIM[i]. If S_DION[i] = TRUE, the value DI_SIM[i] is used Instead of the value of digital input i of the module,. LEDs I1 to I8 also always display the state of the corresponding digital input during simulation.

Simulating Analog Values

The simulation of the analog values for the channels one to 4 is enabled via the switches S_AION[i] or S_PVON[i], where 1 ≤ i ≤ 4.

You specify the simulation values for the channels one to 4 via the parameter PV_SIM[i].

You can have the simulation values become effective at two points:

- S_AION[i] = TRUE (1 ≤ i ≤ 4)
  The value PV_SIM[i] is used instead of the value of analog input i of the module.
- S_PVON[i] = TRUE (1 ≤ i ≤ 4)
  The value PV_SIM[i] is used instead of the conditioned value of analog input i of the module.

The following figure shows at which point the simulated analog value is effective.

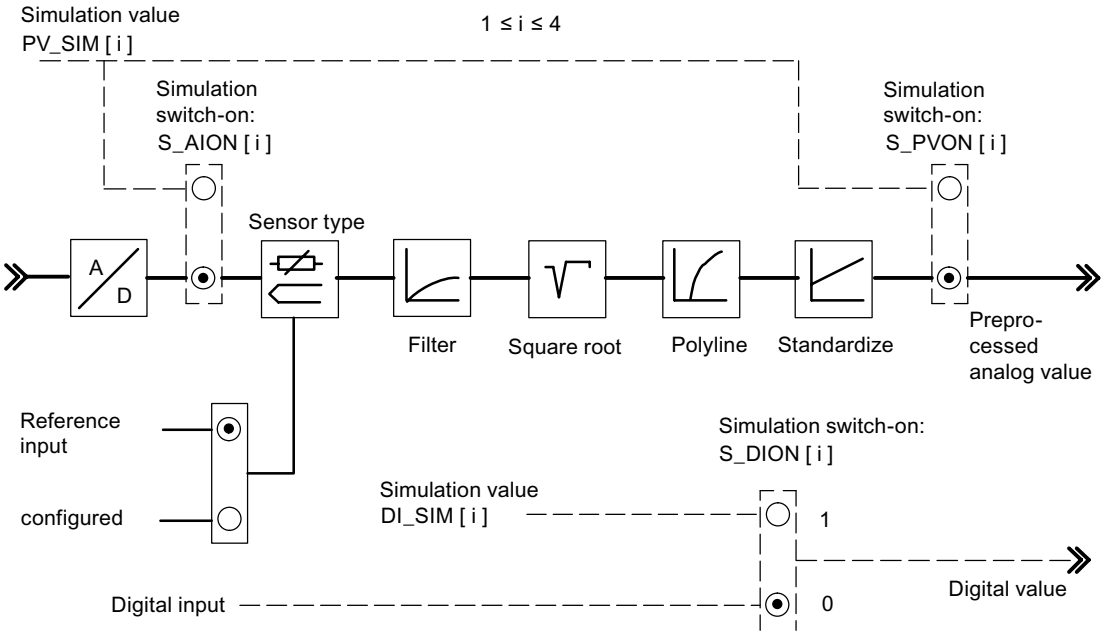

Enabling and specification of the simulation values (forcing) is not carried out via the parameter assignment interface. This is why the relevant switches and connecting lines are drawn as dotted lines.

Call

FORCE355 does not require any initialization routine and is normally called cyclically.

FORCE355 has to be called in the same OB as all the other instructions that access the same FM 355.

Startup characteristics

When FM 355 restarts after power off, the simulation switches on the FM 355 are again set to FALSE.

Input parameters FORCE355 (S7-300, S7-400)

Table 4-233

Parameters	Address	Data type	Default	Description
S_AION	0.0	ARRAY [1..4] of BOOL	FALSE	If, for example, the S_AION[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the analog input value 1 of the module.
S_PVON	2.0	ARRAY [1..4] of BOOL	FALSE	If, for example, the S_PVON[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the conditioned analog input value 1 of the module.
PV_SIM	4.0	ARRAY [1..4] of REAL	0.0	The simulation value for the analog input 1 is specified, for example, at input PV_SIM[1]. If S_PVON = TRUE, then the conditioned analog input value is specified here. If S_PVON = FALSE and S_AION = TRUE, then the analog input value, which is transformed into a conditioned value by means of the conditioning functions, is specified in mA or mV. Permitted are 0.0 to 20.0 [mA], -1500 to +10000 [mV], or a technical range of values.
S_DION	20.0	ARRAY [1..8] of BOOL	FALSE	If, for example, the S_DION[1] switch is set to TRUE, the value DI_SIM[1] is used instead of the digital input value 1 of the module.
DI_SIM	22.0	ARRAY [1..8] of BOOL	FALSE	Simulation value for digital input
MOD_ADDR	24.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters FORCE355 (S7-300, S7-400)

Table 4-234

Parameters	Address	Data type	Default	Description
RET_VALU	26.0	WORD		RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

READ_355 (S7-300, S7-400)

Description READ_355 (S7-300, S7-400)

The instruction READ_355 is used to read out the digital and analog input values to support the start-up.

The following values are displayed:

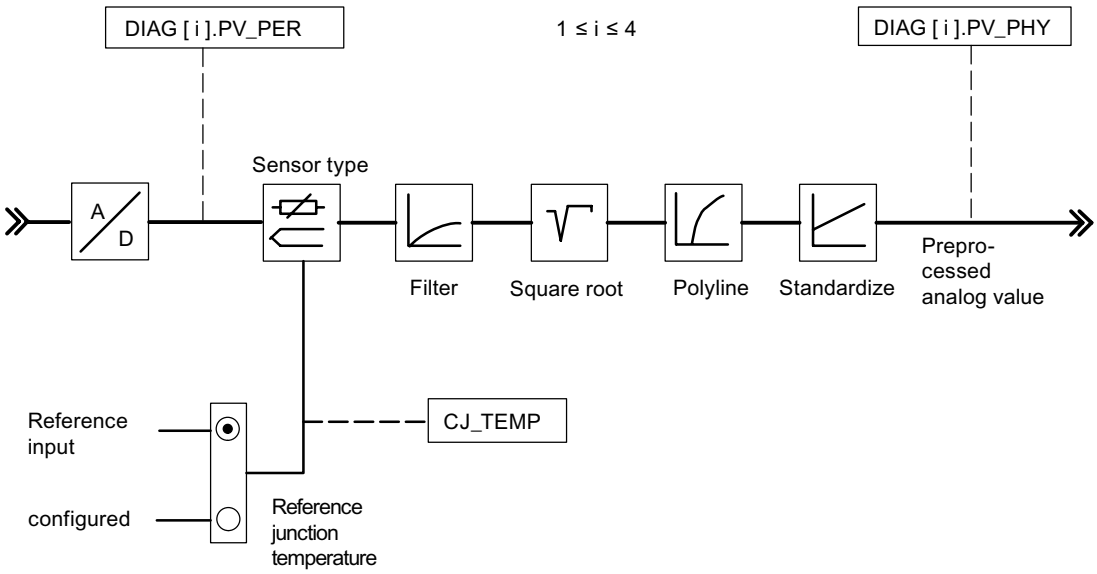

Call

READ_355 does not require any initialization routine and is normally called cyclically.

READ_355 has to be called in the same OB as all the other instructions that access the same FM 355.

Input parameters READ_355 (S7-300, S7-400)

Table 4-235

Parameters	Address	Data type	Default setting	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters READ_355 (S7-300, S7-400)

Table 4-236

Parameters	Address	Data type	Default	Description
CJ_TEMP	2.0	REAL	0.0	At parameter CJ_TEMP the reference junction temperature measured at the reference point is expressed degrees C or in degrees F (depending on which temperature unit has been parameterized). If no "Thermocouple" sensor type was configured or if the configured reference junction temperature was selected at all the analog inputs, 0.0 is displayed at the CJ_TEMP parameter.
STAT_DI	6.0	ARRAY [1..8] of BOOL	FALSE	The actual states of digital inputs 1 to 8 are displayed at parameters STAT_DI[1] to STAT_DI[8], even if these are simulated.
DIAG[x].PV_PER	(channel number) x 8	ARRAY [1..4] of STRUCT	0.0	The values of analog inputs one to four are displayed at parameters DIAG[1].PV_PER bis DIAG[4].PV_PER in the unit mA or mV. If the simulation of the analog input value has been enabled via the instruction FORCE355, the simulated value will be shown.
DIAG[x].PV_PHY	(channel number) x 8 + 4	ARRAY [1..4] of STRUCT	0.0	The conditioned analog input value one to four is displayed in physical unit at the parameters DIAG[1].PV_PHY to DIAG[4].PV_PHY If the simulation of the conditioned physical analog input value is enabled via the instruction FORCE355, the simulated value is displayed.
RET_VALU	40.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

CH_DIAG (S7-300, S7-400)

Description CH_DIAG (S7-300, S7-400)

The instruction CH_DIAG is used to read out additional channel-specific diagnostic variables from the module.

Call

CH_DIAG does not require any initialization routine and is normally called cyclically.

CH_DIAG has to be called in the same OB as all the other instructions that access the same FM 355.

Input parameters CH_DIAG (S7-300, S7-400)

Table 4-237

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	1	The number of the controller channel to which the instance DB is referenced is configured at the "Channel number" input. Values of 1 to 4 are permitted.

See also

Description CH_DIAG (Page 5984)
Output parameters CH_DIAG (Page 5985)
Parameter IDSTATUS (Page 5986)
Block diagrams CH_DIAG (Page 5987)

Output parameters CH_DIAG (S7-300, S7-400)

Table 4-238

Parameters	Address	Data type	Default	Description
SP_R	4.0	REAL	0.0	If a ratio controller is set, the input value of the setpoint value is assigned to the parameter.
PV_R	8.0	REAL	0.0	The following value is only assigned to the parameter value if the ratio controller is set: (Process value A - Setpoint value offset) / Process value D
DIF_I	12.0	REAL	0.0	The input variable of the D-action is displayed at the DIF_I parameter. This is particularly of interest if, for example, an analog input is configured as the input variable of the D-action.
TRACKPER	16.0	REAL	0.0	The TRACKPER parameter shows the input variable to which the output value is tracked if the track-manipulated-variable function is enabled on the controller.
IDSTATUS	20.0	WORD	0.0	The state of the identification is displayed at AUTOHOTSPOT.
LMN_P	22.0	REAL	0.0	The P-action of the manipulated variable is displayed at the LMN_P parameter.
LMN_I	26.0	REAL	0.0	The I-action of the manipulated variable is displayed at the LMN_I parameter.
LMN_D	30.0	REAL	0.0	The D-action of the manipulated variable is displayed at the LMN_D parameter.
RET_VALU	34.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.

See also

RDREC: Read data record (Page 5224)
WRREC: Write data record (Page 5226)

Parameter IDSTATUS (S7-300, S7-400)

The IDSTATUS parameter supplies information on the status of identification and contains the four hexadecimal values X, A, I and F.

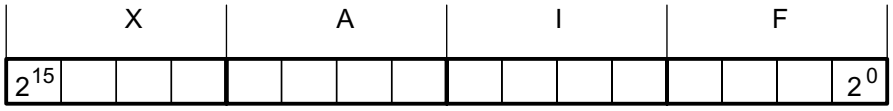

Value	Description
X	Without meaning (always 0)
A	Action number • 0 = Manual mode (or no closed-loop control mode); • 2 = Closed-loop control; • 4 = Tuning enabled (FUZID_ON = true); • 6 = Transition state from manual mode to 2 or 4;
I	Display "Identification running" and "Parameters determined, but not yet stored in EEPROM" • 0 = Identification not running, no new parameters determined • 1 = Identification running, no new parameters determined • 2 = Identification not running, new parameters determined, but not yet stored in EEPROM • 3 = Identification running, new parameters determined, but not yet stored in EEPROM
F	Error number • 0 = No error • 4 = Excessive step-change of the process value during the identification • 5 = Ratio of time lag to system time constant too large or strongly non-linear behavior of the controlled system. • 6 = Temperature drop or rise during identification start too large. System not settled sufficiently

Block diagrams CH_DIAG (S7-300, S7-400)

The following diagnostic values of the control deviation are displayed.

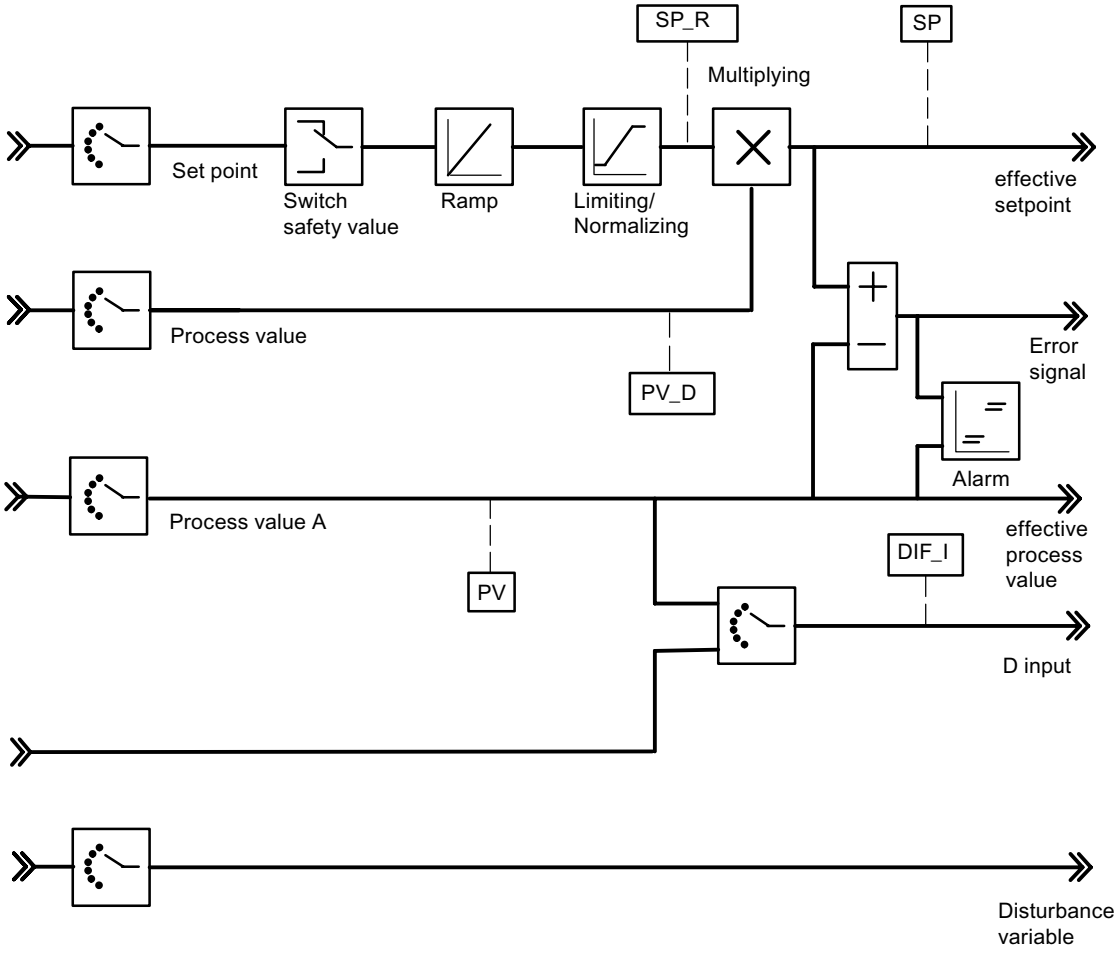

The following values of the control algorithm are displayed.

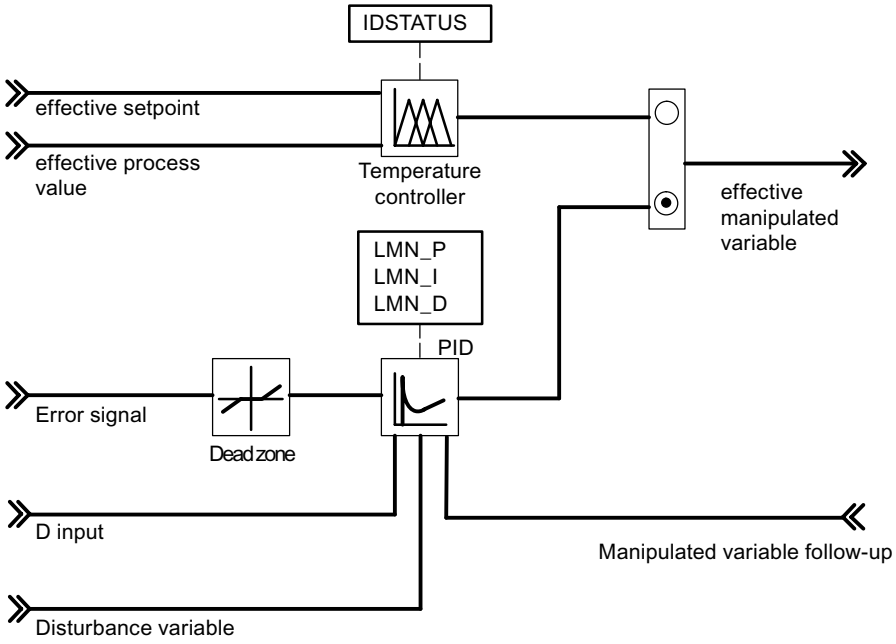

The following values of the continuous controller or step controller are displayed.

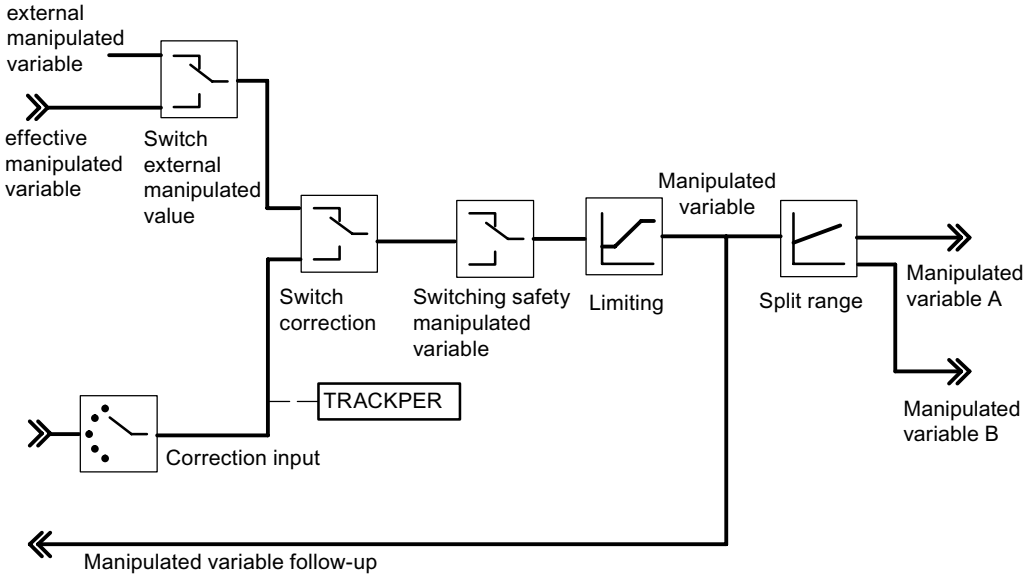

PID_PAR (S7-300, S7-400)

Description PID_PAR (S7-300, S7-400)

The instruction PID_PAR is used for the online modification of parameters that cannot be defined via the instruction PID_FM. To view the parameters that can be modified, refer to the table Changeable parameters (Page 5991)

Call

PID_PAR requires an initialization routine. It is automatically triggered if the system data (SDB default data of the FM 355) has not yet been read from the instruction PID_PAR. You can also start the initialization yourself with COM_RST=TRUE, which is usually done in OB100 since the system data is sent to the FM 355 after STOP-RUN of the CPU. The initialization process lasts several cycles. No data is sent to the FM 355 via WRREC during the initialization. The block automatically resets the COM_RST parameter after the initialization.

To save run time, PID_PAR should not be called cyclically. It should only be called when parameters are to be changed. COM_RST must then be FALSE.

PID_PAR has to be called in the same OB as all the other instructions that access the same FM 355.

If the FM 355 is used in distributed I/Os, it may take a few call cycles until the parameters have been transferred to the FM 355. The BUSY parameter has the value TRUE until the transfer has been completed. You should call PID_PAR as many times as necessary until BUSY = FALSE and RET_VALU = 0.

Mode of operation

In the initialization routine, PID_PAR reads the parameters from the system data and stores them in static variables. Per call, PID_PAR modifies one of these REAL variables and one of these INT variables. At the input parameter INDEX_R or INDEX_I, specify the index numbers of the parameter that you want to modify. At the input parameter VALUE_R or VALUE_I, enter the new values. PID_PAR transfers the entire data record to FM 355 with the modified variables.

To modify additional parameters you have to call **the same** instance DB several times consecutively with COM_RST = FALSE and with different index numbers.

The COM_RST parameter is an input parameter that is not reset by PID_PAR.

Note

Please note that parameters that you have changed using PID_PAR are overwritten by the parameters from the system data during the start-up.

Start-up

During CPU start-up and during CPU STOP-RUN transition, the parameters on the FM 355 are overwritten with the parameters from the SDB of the CPU.

Example

During operation you want to modify the ramp-up time for the reference variable and, depending on the process state, use different analog input values as process value.

- Call PID_PAR using COM_RST = TRUE during the start-up of the CPU.
- To configure the ramp-up time for the reference variable at 10.0, use INDEX_R = 30, VALUE_R = 10.0 and INDEX_I = 0 to call PID_PAR during operation.
- If you want to parameterize the analog input value 4 of the module as the process value, call PID_PAR with INDEX_R = 0, INDEX_I = 50 and VALUE_I = 4 during operation.

See also

Alterable parameters (Page 5991)

Input parameters PID_PAR (Page 5990)

Output parameters PID_PAR (Page 5991)

Input parameters PID_PAR (S7-300, S7-400)

Table 4-239

Parameters	Address	Data type	Default	Description
COM_RST	0.0	BOOL	TRUE	If the parameter COM_RST = TRUE, the instruction PID_PAR executes an initialization routine. During this initialization process the parameters are read from the system data of the CPU and saved in the instance DB.
MOD_ADDR	2.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	4.0	INT	1	The number of the controller channel to which the instance DB is referenced is configured at the "Channel number" input. Values 1 to 4 are permitted.
INDEX_R	6.0	INT	0.0	• 0: No REAL parameters are changed • 1 to 48: Each value corresponds to a changeable REAL parameter.
VALUE_R	8.0	REAL	0.0	Depends on the relevant parameter
INDEX_I	12.0	INT	0.0	• 0: No INT parameters are changed. • 49 to 61: Each value corresponds to a changeable INT parameter.
VALUE_I	14.0	INT	0.0	Depends on the relevant parameter

See also

Description PID_PAR (Page 5988)

Output parameters PID_PAR (Page 5991)

Output parameters PID_PAR (S7-300, S7-400)

Table 4-240

Parameters	Address	Data type	Default	Description
RET_VALU	16.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.
BUSY	18.0	BOOL	FALSE	If BUSY = TRUE, the parameters have not yet been transferred from the module (in the case of distributed I/Os). The instruction PID_PAR should then be called again in the next cycle.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

Alterable parameters (S7-300, S7-400)

Table 4-241 List of the REAL and INT parameters on a FM 355 that are to be modified with PID_PAR

Data type	Description	Index number
–	No parameter selected	0
REAL	Filter time constant for analog input	1
REAL	End of scale (100%)	2
REAL	Start of scale (0%)	3
REAL	Polyline, Interpolation point 1 input side	4
REAL	Polyline, Interpolation point 2 input side	5
REAL	Polyline, Interpolation point 3 input side	6
REAL	Polyline, Interpolation point 4 input side	7
REAL	Polyline, Interpolation point 5 input side	8
REAL	Polyline, Interpolation point 6 input side	9
REAL	Polyline, Interpolation point 7 input side	10
REAL	Polyline, Interpolation point 8 input side	11
REAL	Polyline, Interpolation point 9 input side	12
REAL	Polyline, Interpolation point 10 input side	13
REAL	Polyline, Interpolation point 11 input side	14
REAL	Polyline, Interpolation point 12 input side	15
REAL	Polyline, Interpolation point 13 input side	16
REAL	Polyline, Interpolation point 1 output side	17
REAL	Polyline, Interpolation point 2 output side	18
REAL	Polyline, Interpolation point 3 output side	19
REAL	Polyline, Interpolation point 4 output side	20
REAL	Polyline, Interpolation point 5 output side	21

Data type	Description	Index number
REAL	Polyline, Interpolation point 6 output side	22
REAL	Polyline, Interpolation point 7 output side	23
REAL	Polyline, Interpolation point 8 output side	24
REAL	Polyline, Interpolation point 9 output side	25
REAL	Polyline, Interpolation point 10 output side	26
REAL	Polyline, Interpolation point 11 output side	27
REAL	Polyline, Interpolation point 12 output side	28
REAL	Polyline, Interpolation point 13 output side	29
REAL	Ramp-up time for reference variable	30
REAL	Safety reference variable or safety Reference variable response	31
REAL	Offset for setpoint link (ratio/blending controller)	32
REAL	Factor for process value B (three-component controller)	33
REAL	Factor for process value C (three-component controller)	34
REAL	Offset for process value link (three-component controller)	35
REAL	Factor for disturbance variable link	36
REAL	Operating point	37
REAL	Aggressivity at fuzzy controller	38
REAL	Vertices for split-range function: Start of input signal A range	39
REAL	Vertices for split-range function: End of input signal A range	40
REAL	Vertices for split-range function: Start of output signal A range	41
REAL	Vertices for split-range function: End of output signal A range	42
REAL	Vertices for split-range function: Start of input signal B range	43
REAL	Vertices for split-range function: End of input signal B range	44
REAL	Vertices for split-range function: Start of output signal B range	45
REAL	Vertices for split-range function: End of output signal B range	46
REAL	Minimum pulse time	47
REAL	Minimum break time	48
INT	Selection of the reference variable SP or SP_RE for the controller 0: Setpoint SP_RE of the instruction 1 to 4: Analog input value 1 to 4 17 to 20: Manipulated variable (LMN) of controllers 1 to 4	49
INT	Selection of the main controlled variable process value A for the controller 0: Process value A = 0.0 1 to 4: Analog input value 1 to 4	50
INT	Selection of the auxiliary controlled variable process value B for the controller 0: Process value B = 0.0 1 to 4: Analog input value 1 to 4	51
INT	Selection of the auxiliary controlled variable process value C for the controller 0: Process value C = 0.0 1 to 4: Analog input value 1 to 4	52

Data type	Description	Index number
INT	Selection of the auxiliary controlled variable process value D for the controller 0: Process value D = 0.0 1 to 4: Analog input value 1 to 4 17 to 20: Manipulated variable (LMN) of controllers 1 to 4	53
INT	Selection of the disturbance variable DISV for the controller 0: Disturbance variable = 0.0 1 to 4: Analog input value 1 to 4	54
INT	Selection of position follow-up TRACK_PER for the controller 0: Position follow-up = 0.0 1 to 4: Analog input value 1 to 4	55
INT	Selection of position follow-up LMNR_PER for the controller 0: Position follow-up = 0.0 1 to 4: Analog input value 1 to 4	56
INT	Selection of the signal for switching to the safety value for the manipulated value of the controller 0: Can only be assigned via SAFE_ON of the instruction PID_FM 1 to 8: Assignment via SAFE_ON parameter of the instruction PID_FM ORed with digital input 1 to 8	57
INT	Selecting the signal for switching over to tracking function of the manipulated value of the controller 0: Can only be assigned via LMNTRKON of the instruction PID_FM 1 to 8: Assignment via LMNTRKON parameter of the instruction PID_FM ORed with digital input 1 to 8	58
INT	Selecting the signal for changing over the manipulated value of the controller to LMN_RE 0: Can only be assigned via LMN_REON of the instruction PID_FM 1 to 8: Assignment via LMN_REON parameter of the instruction PID_FM ORed with digital input 1 to 8	59
INT	Selecting the high limit signal of the position feedback 0: Can only be assigned via LMNRHSRE of the instruction PID_FM 1 to 8: Assignment via LMNRHSRE parameter of the instruction PID_FM ORed with digital input 1 to 8	60
INT	Selecting the low limit signal of the position feedback 0: Can only be assigned via LMNRLSRE of the instruction PID_FM 1 to 8: Assignment via LMNRLSRE parameter of the instruction PID_FM ORed with digital input 1 to 8	61

See also

Input parameters PID_PAR (Page 5990)

Output parameters PID_PAR (Page 5991)

CJ_T_PAR (S7-300, S7-400)

Description CJ_T_PAR (S7-300, S7-400)

The instruction CJ_T_PAR is used for online modification of the configured reference junction temperature. This is necessary if a temperature control system with several FM 355 with thermoelement inputs is to be operated without having to connect a Pt 100 to each FM 355.

Call

CJ_T_PAR requires an initialization routine. For this purpose it has to be called once with the COM_RST = TRUE parameter at the start-up of the CPU.

CJ_T_PAR is normally called cyclically. For this purpose COM_RST should be set to FALSE for runtime reasons. The COM_RST parameter is an input parameter that is not reset.

CJ_T_PAR has to be called in the same OB as all the other instructions that access the same FM 355.

If the FM 355 is used in distributed I/Os, it may take a few call cycles until the parameters have been transferred to the FM 355. The BUSY parameter has the value TRUE until the transfer has been completed. You should therefore call the instruction CJ_T_PAR as many times as necessarry when changing parameters until BUSY = FALSE.

Example

If an FM 355 measures the reference junction temperature of an extruder control system with more than four heating zones, this can be read out by READ_355 at parameter CJ_TEMP and can be parameterized for the other FM 355 via CJ_T_PAR.

See also

Input parameters CJ_T_PAR (Page 5994)

Output parameters CJ_T_PAR (Page 5995)

Input parameters CJ_T_PAR (S7-300, S7-400)

Table 4-242

Parameters	Address	Data type	Default	Description
COM_RST	0.0	BOOL	FALSE	If the parameter COM_RST = TRUE, the instruction CJ_T_PAR executes an initialization routine. During this initialization process the parameters are read from the system data of the CPU and saved in the instance DB.
MOD_ADDR	2.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CJ_T	4.0	REAL	0.0	The reference junction temperature can be specified at parameter CJ_T. The permissible value range depends on the sensor type.

See also

Description CJ_T_PAR (Page 5994)

Output parameters CJ_T_PAR (Page 5995)

Output parameters CJ_T_PAR (S7-300, S7-400)

Table 4-243

Parameters	Address	Data type	Default	Description
RET_VALU	8.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.
BUSY	10.0	BOOL	FALSE	If BUSY = TRUE, the parameters have not yet been transferred from the module (in the case of distributed I/Os). The instruction PID_PAR should then be called again in the next cycle.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

DB for HMI (S7-300, S7-400)

Description DB 101 to 104 (S7-300, S7-400)

In order to operate the FM 355 if the CPU fails, you require data blocks 101 to 104. You can establish a maximum of three connections from the FM 355 to OPs via MPI.

Operator control of the FM 355 with the OP is only possible in the STOP mode of the CPU or at a CPU failure.

Monitoring of the FM 355 with the OP is always possible.

The variable interface of the FM 355 contains four data blocks with the block numbers 101 to 104 for the controller channels 1 to 4 (refer to the following figure).

Note

The contents of the data blocks 101 to 104 do not automatically reflect the parameter value effective at the FM 355. Parameters changed with the OP are only transferred to the FM 355 after the operating bits LOAD_PAR and LOAD_OP have been set.

If you change a parameter using OP operation without setting the corresponding operating bit, the changed parameter value is entered in the data block, but the FM 355 continues to operate internally with the unchanged old value of the parameter.

After the operating bits have been set and the parameters have been transferred to the FM 355, the operating bits LOAD_PAR and LOAD_OP are reset by the FM 355.

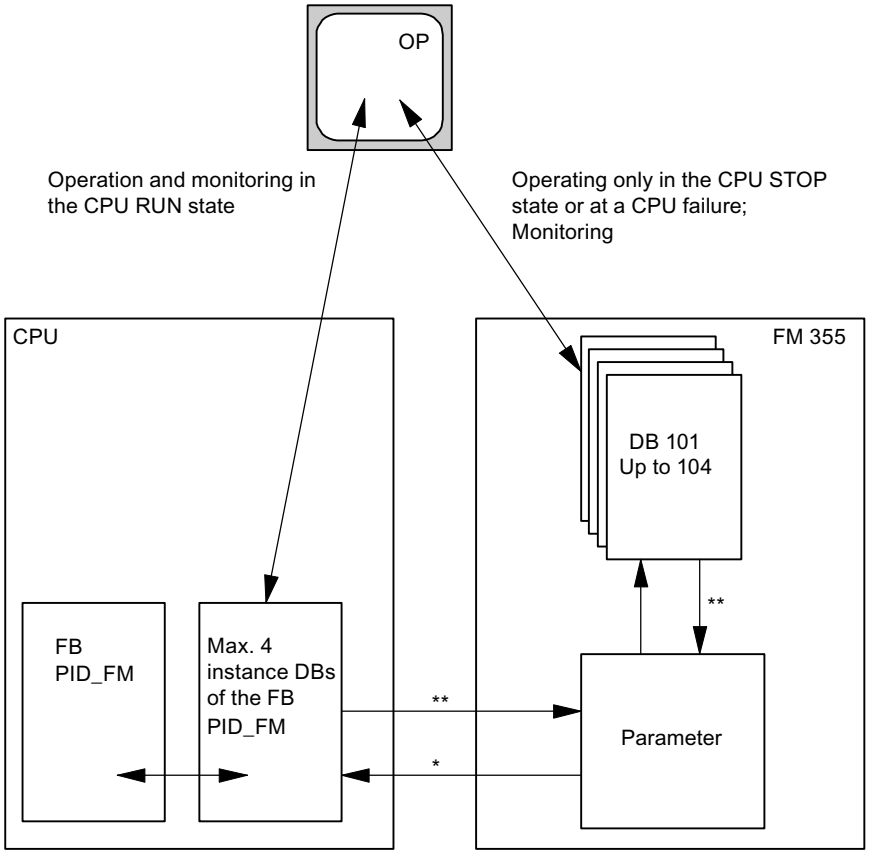

* controlled by the READ_VAR parameter of the instance DB

** controlled by the LOAD_OP and LOAD_PAR parameters

See also

Description PID_FM (Page 5962)

Input parameters DB 101 to 104 (S7-300, S7-400)

Table 4-244

Parameters	Address	Data type	Default	Description
SP_HLM	0.0	REAL	100.0	The setpoint is always limited to a high and a low limit. The "Setpoint high limit" input specifies the high limit. SP_HLM > SP_LLM
SP_LLM	4.0	REAL	0.0	The setpoint is always limited to a high and a low limit. The "Setpoint low limit" input specifies the low limit. SP_LLM < SP_HLM
H_ALM	8.0	REAL	100.0	You can configure four limits for monitoring the process value or error signal. The "High limit interrupt" input specifies the highest limit. H_ALM > H_WRN

Parameters	Address	Data type	Default	Description
H_WRN	12.0	REAL	90.0	You can configure four limits for monitoring the process value or error signal. The "High limit warning" input specifies the second to highest limit. Values from H_ALM to L_WRN are permitted.
L_WRN	16.0	REAL	10.0	You can configure four limits for monitoring the process value or error signal. The "Low limit warning" input specifies the second to lowest limit. Values from H_WRN to L_ALM are permitted.
L_ALM	20.0	REAL	0.0	You can configure four limits for monitoring the process value or error signal. The "Low limit alarm" specifies the lowest limit. L_ALM < L_WRN
HYS	24.0	REAL	1.0	To prevent flickering of the monitoring displays, a hysteresis can be configured at the "Hysteresis" input. HYS ≥ 0.0
DEADB_W	28.0	REAL	0.0	A dead band is applied to the error signal. The "Dead band width" input determines the size of the dead band. DEADB_W ≥ 0.0
GAIN	32.0	REAL	1.0	The "Proportional gain" input specifies controller gain.
TI	36.0	REAL	3000.0	The "Integration time" input determines the time response of the integrator. If TI = 0, the integral action is disabled. TI = 0.0 or TI ≥ 0.5
TD	40.0	REAL	0.0	The "Derivative-action time" input determines the time response of the differentiator. If TD = 0, the derivative action is disabled TD = 0.0 or TD ≥ 1.0
TM_LAG	44.0	REAL	5.0	The algorithm of the D-action includes a time lag that can be configured at the "Time lag of the D-action" input. TM_LAG ≥ 0.5
LMN_SAFE	48.0	REAL	0.0	At the "Safety manipulated value" input, a safety value can be configured for the manipulated value, . Values from -100.0 to 100.0 (%) are permitted.
LMN_HLM	52.0	REAL	100.0	The manipulated value is always limited to a high and a low limit. The "High limit limit of manipulated value" input specifies the high limit. (This does not apply to step controllers without analog position feedback) Values from LMN_LLM to 100.0 (%) are permitted.
LMN_LLM	56.0	REAL	0.0	The manipulated value is always limited to a high and a low limit. The "Low limit of manipulated value" input specifies the low limit. (This does not apply to step controllers without analog position feedback) Values from -100.0 to LMN_HLM (%) are permitted.
MTR_TM	60.0	REAL	60.0	The runtime from endstop to endstop of the control valve is entered at the "Motor actuating time" parameter. (Applies to step controllers only). MTR_TM ≥ 0.001
PULSE_TM	64.0	REAL	0.2	A minimum pulse duration can be configured at the "Minimum pulse time" parameter. (Applies to step controllers or pulse controllers only) PULSE_TM ≥ 0.0

Parameters	Address	Data type	Default	Description
BREAK_TM	68.0	REAL	0.2	A minimum break time can be configured at the "Minimum break time" parameter. (Applies to step controllers or pulse controllers only) BREAK_TM ≥ 0.0
SP_RE	72.0	REAL	0.0	An external setpoint is interconnected with the controller at the "External setpoint" input.
LMN_RE	76.0	REAL	0.0	An external manipulated value is interconnected with the controller at the "External manipulated value" input.
LMNRSVAL	80.0	REAL	0.0	The Commissioning dialog provides you with access to the input "Start value of the simulated position feedback". The start value of the simulation is entered at the parameter. (Only for step controllers without analog position feedback) Values from -100.0 to 100.0 (%) are permitted.
SAFE_ON	84.0	BOOL	FALSE	If the "Adopt safety position" input is set, a safety value is applied as manipulated value. Note: The signal processing of manipulated values via LMNDN_OP , LMNUP_OP and LMNSOPON on a step controller is of higher priority than the safety manipulated value.
LMNTRKON	84.1	BOOL	FALSE	If input "Follow-up (LMN via AI)" is set, the manipulated value is adjusted to an analog input (AI). (This does not apply to step controllers without analog position feedback)
LMN_REON	84.2	BOOL	FALSE	If input "Enable external manipulated value" is set, the external manipulated value LMN_RE is set as the manipulated value.
LMNRHSRE	84.3	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected at the "High endstop signal of position feedback" input. LMNRHSRE = TRUE means: The control valve is at high endstop. (Applies to step controllers only).
LMNRLSRE	84.4	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected at the "Low endstop signal of position feedback" input. LMNRLSRE = TRUE means: The control valve is at the low endstop. (Applies to step controllers only).
LMNSO-PON	84.5	BOOL	FALSE	If the bit is set at input "Enable manipulated value signal operation", the LMNUP_OP and LMNDN_OP signals are set as the manipulated value signals. (Applies to step controllers only).
LMNUP_OP	84.6	BOOL	FALSE	If LMNSOPON is set, the value at input "Manipulated value signal up operation" is set as the manipulated value signal. (Applies to step controllers only).
LMNDN_OP	84.7	BOOL	FALSE	If LMNSOPON is set, the value at input "Manipulated value signal down operation" is set as the manipulated value signal. (Applies to step controllers only).
MONER-SEL	85.0	BOOL	FALSE	The controller possesses a limit value detector that can be applied either for the process value or for the error signal. If the input "Monitoring: Process value = 0, Error signal = 1" is set, the errcr signal will be monitored.
LMNRS_ON	85.1	BOOL	FALSE	You can simulate no position feedback if this is not available. The function is enabled by setting the "Enable simulation of position feedback" input CAUTION: Offset between the simulation value and real position feedback increases over the time. (Only for step controllers without analog position feedback)

Parameters	Address	Data type	Default	Description
FUZID_ON	85.2	BOOL	FALSE	The identification of the fuzzy algorithm is enabled at the input "Enable fuzzy identification".
SPINT_EN	85.3	BOOL	FALSE	The input "Operator input: External = 0, Internal = 1" determines the input that is transferred as a setpoint to the module. SPINT_EN = TRUE: SP_INT is transferred. SPINT_EN = FALSE: SP_RE is transferred.
P_SEL	85.4	BOOL	TRUE	It is possible to enable or disable PID actions individually in the PID algorithm. The proportional action is enabled when the "Enable P-action" input is set.
PFDB_SEL	85.5	BOOL	FALSE	In the PID algorithm, the P- and D-actions can be included in the feedback loop. The proportional action is located in the feedback loop when the "Enable P-action in feedback loop" input is set.
D_EL_SEL	86.0	INT	0	The D-action element in the PID algorithm can be assigned to a separate input. This is selected at the input "D-action element input". 0: Error signal 1 to 4: Analog input 1 to 4 17: negative process value

See also

Description PID_FM (Page 5962)

Output parameters DB 101 to 104 (S7-300, S7-400)

Table 4-245

Parameters	Address	Data type	Default	Description
SP	94.0	REAL	0.0	The setpoint that is currently in effect is available at the "Setpoint" output.
PV	98.0	REAL	0.0	The process value that becomes effective is issued on the "Process value" output.
ER	102.0	REAL	0.0	The control deviation that becomes effective is issued on the "Control deviation" output.
DISV	106.0	REAL	0.0	The interference that becomes effective is issued on the "Interference" output. Values from -100.0 to 100.0 (%) are permitted.
LMN	110.0	REAL	0.0	The manipulated value that becomes effective is issued on the "Manipulated value" output. Step controllers without analog position feedback output the unlimited P- + D-action at the LMN parameter. Values from -100.0 to 100.0 (%) are permitted.
LMN_A	114.0	REAL	0.0	At the output "Manipulated value A of the split range function / position feedback", continuous controllers display the manipulated value A and step controllers with analog position feedback display the position feedback. Step controllers without analog position feedback display the simulated position feedback. Values from -100.0 to 100.0 (%) are permitted.
LMN_B	118.0	REAL	0.0	Continuous controllers display the manipulated value B of the split-range function at the output “Manipulated value B of split-range function”. Values from -100.0 to 100.0 (%) are permitted.

Parameters	Address	Data type	Default	Description
QH_ALM	122.0	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_ALM limit is reported at the output "High limit alarm triggered".
QH_WRN	122.1	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Exceeding of the limit H_WRN is signaled at the "High limit warning triggered" output.
QL_WRN	122.2	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Exceeding of the limit L_WRN is signaled at the "Low limit warning triggered" output.
QL_ALM	122.3	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Exceeding of the limit L_ALM is signaled at the "Low limit alarm triggered" output.
QLMN_HLM	122.4	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the high limit is reported at output "High limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QLMN_LLM	122.5	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the low limit is reported at output "Low limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QSPINTON	122.6	BOOL	FALSE	The output "Internal setpoint on" indicates that SP_INT was transferred to the module.
QPARA_F	123.0	BOOL	FALSE	The module checks the validity of the parameters. A parameter assignment error is displayed at the "Parameter assignment error" output. These parameter assignment errors can also be read out in the module's "Online & Diagnostics" dialog.
QCH_F	123.1	BOOL	FALSE	The "Channel error" output is set if the controller channel cannot provide valid results. A channel error (for example, wire break) is also set if QPARA_F = 1 or QMOD_F = 1. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.
QUPRLM	123.2	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of positive setpoint slew rate triggered" is set, the positive setpoint slew rate is limited.
QDNRLM	123.3	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of negative setpoint slew rate triggered" is set, the setpoint fall is limited.
QSP_HLM	123.4	BOOL	FALSE	The setpoint is always limited to a high and a low limit. The output "High limit of setpoint value triggered" indicates that the high limit has been exceeded.
QSP_LLM	123.5	BOOL	FALSE	The setpoint is always limited to a high and a low limit. Violation of the low limit is reported at output "Setpoint low limit triggered".
QSPOPON	123.6	BOOL	FALSE	The output "Setpoint operation on" shows if the setpoint is being operated via the "Commissioning" dialog. If the bit is set, the value SP_OP is applied as setpoint.
QLMNSAFE	123.7	BOOL	FALSE	If the output "Safety mode" is set, the safety manipulated value is output as the manipulated value.
QLMNO-PON	124.0	BOOL	FALSE	The output "Manipulated value operation on" shows if the manipulated value is being operated via the "Commissioning" dialog. If the bit is set, the value LMN_OP is applied as manipulated value.
QLMNTRK	124.1	BOOL	FALSE	The output "Follow-up mode" indicates if the manipulated value is being adjusted via an analog input.
QLMN_RE	124.2	BOOL	FALSE	The output "Manual = 1; Automatic = 0" indicates whether or not the manipulated value is set on the external manipulated value LMN_RE (manual = 1).
QLMNR_HS	124.3	BOOL	FALSE	The output "High endstop signal of position feedback" indicates if the control valve is at its high endstop. QLMNR_HS = TRUE means: The control valve is at high endstop. (For step controllers only)

Parameters	Address	Data type	Default	Description
QLMNR_LS	124.4	BOOL	FALSE	The output "Low endstop signal of position feedback" indicates whether the control valve is at its low endstop. QLMNR_LS = TRUE means: The control valve is at its low endstop. (For step controllers only)
QLMNR_ON	124.5	BOOL	FALSE	Output "Position feedback enabled" indicates the set operating mode "Step controller with position feedback" or "Step controller without position feedback".
QFUZZY	124.6	BOOL	FALSE	If the output "PID algorithm = 0; fuzzy = 1", the controller is working with the fuzzy algorithm.
QSPLEPV	124.7	BOOL	FALSE	The output "Display of FUZZY controller: Setpoint < process value" is set when the fuzzy controller is switched on, if the setpoint is less than the effective process value.
QSPR	125.0	BOOL	FALSE	If the output "Split-range operation" is set, the continuous controller is operating in split-range mode.
QLMNUP	125.1	BOOL	FALSE	This is the "Manipulated value signal up" output. (Only in the case of step controllers or pulse controllers)
QLMNDN	125.2	BOOL	FALSE	This is the "Manipulated value signal down" output. (Only in the case of step controllers or pulse controllers)
QBACKUP	125.4	BOOL	FALSE	0= No backup state (CPU in RUN) 1= Backup state (CPU in STOP or has failed)
QID	125.5	BOOL	FALSE	QID = TRUE indicates that an identification is running (not that it is enabled). On completion of the identification, the identification result can be read out from the IDSTATUS parameter of the instruction CH_DIAG.
QMAN_FC	125.6	BOOL	FALSE	The controller is a master controller that is adjusted by the manual mode of a slave controller to its process value or its I-action is stopped, because the setpoint or manipulated value of the slave controller lies in the limit.
RET_VALU	126.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

Description PID_FM (Page 5962)

In/out parameters DB 101 to 104 (S7-300, S7-400)

Table 4-246

Parameters	Address	Data type	Default	Description
SP_INT	128.0	REAL	0.0	The in/out parameter "Internal setpoint" is used to specify a setpoint by means of operating and monitoring functions.
SP_OP	132.0	REAL	0.0	The "Commissioning" dialog can access the in/out parameter "Setpoint operation". If the SP_OP_ON bit is set, the "Setpoint operation" value is applied as the setpoint.

Parameters	Address	Data type	Default	Description
LMN_OP	136.0	REAL	0.0	The "Commissioning" dialog can access the in/out parameter "Manipulated value operation". If the LMNOP_ON bit is set, the "Manipulated value operation" value is applied as the manipulated value. Values from -100.0 to 100.0 (%) are permitted.
SP_OP_ON	140.0	BOOL	FALSE	The "Commissioning" dialog can access the in/out parameter "Enable setpoint operation". If the bit is set, the value SP_OP is applied as setpoint.
LMNOP_ON	140.1	BOOL	FALSE	The "Commissioning" dialog can access the in/out parameter "Enable manipulated value operation". If the bit is set, the value LMN_OP is applied as manipulated value.
LOAD_PAR	140.2	BOOL	FALSE	If the in/out parameter "Load control parameters to FM 355" is set, the control parameters are loaded into the module and the in/out parameter is reset.
LOAD_OP	140.3	BOOL	FALSE	If the in/out parameter "Load operating parameters to FM 355" is set, the operating parameters are loaded into the module and the in/out parameter is reset.

See also

Description DB 101 to 104 (Page 5995)

Description PID_FM (Page 5962)

FM 355-2 Temperature Control (S7-300, S7-400)

FMT_PID (S7-300, S7-400)

Description FMT_PID (S7-300, S7-400)

During operation, the instruction FMT_PID is used to:

- Transfer operating parameters to FM 355-2
- Transfer controlller parameters to FM 355-2
- Read process values from FM 355-2

Operating parameters are all static variables of instance DB in the structure OP.

Controller parameters are all static variables of instance DB in the structure PAR.

Process values are all static variables of instance DB in the structure OUT.

The data required for the FMT_PID are stored in an instance DB on the CPU. The FMT_PID is program controlled to read data from the FM 355-2 and to write data to the FM 355-2.

Call

The instruction FMT_PID has to be called in the same OB as all the other instructions that access the same FM 355-2.

FMT_PID is called in the cyclic interrupt OB. It requires an initialization routine that is started by setting the COM_RST = TRUE parameter in the start-up of the CPU. After the initialization routine, FMT_PID sets the parameter COM_RST to FALSE.

If you call the instruction FMT_PID as a multiple instance DB, no commissioning interface is available. In this case you must commission the FM 355-2 via a watch table or with the instruction FMT_TUN.

Calling in Distributed I/O

The following has to be noted in the case of a distributed structure and simultaneous calling of FMT_PID and FMT_TUN:

- It is not permitted to set LOAD_OP simultaneously with FMT_PID and FMT_TUN.
- It is not permitted to set READ_OUT simultaneously with FMT_PID and FMT_TUN.

Reason: Both FBs access FM 355-2 via the same data records.

You must therefore ensure that only one of the two FBs reads or writes a data record at the same time.

Start-up

During CPU start-up and during CPU STOP-RUN transition, the parameters on the FM 355-2 are overwritten with the parameters from the SDB of the CPU.

Responses in the event of an error

The RET_VALU output parameter contains the 2nd and 3rd bytes of the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated if the READ_PAR and LOAD_PAR parameters are not reset.

When the instruction FMT_PID is called, an I/O access error can occur if the FM 355-2 is not plugged or is not supplied with voltage. In this case the CPU goes into STOP, if no OB 122 is loaded in the CPU.

If an error occurs when reading/writing a data set, QMOD_F and QCH_F must be set. In the case of parameter assignment errors, QPAR_F and QCH_F are set.

See also

Block diagrams (Page 6013)

Mode of operation FMT_PID (S7-300, S7-400)

Effect of parameters LOAD_OP, READ_OUT, LOAD_PAR and READ_PAR

In the initialization routine FMT_PID reads the parameters from the system data and stores them in its instance DB. Instance DB and system data are synchronized.

The following figure shows how the function of the instruction is influenced by the parameters LOAD_OP, READ_OUT, LOAD_PAR and READ_PAR.

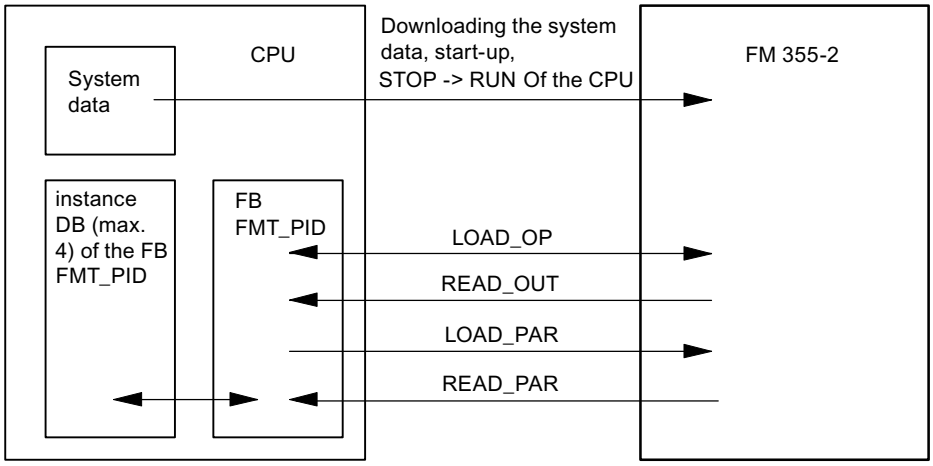

The following table contains detailed information depending on the values of the parameters.

Parameters	TRUE	FALSE
LOAD_OP	The static variables of the structure OP are transferred cyclically FM 355-2 with the instructions WRREC and RDREC. The output parameters are read from the FM 355-2.	The static variables of the structure OP are transferred via the input and output areas of the module. This is not possible if two instances of the instruction FMT_PID access the same channel number of a module.
	READ_VAR is set to TRUE	READ_VAR is not changed
	In the following situations LOAD_OP is automatically TRUE: • If you have set one of the following parameters: TUN_ST, TUN_CST, SAVE_PAR,UNDO_PAR and LOAD_PID • If you modify TUN_ON • If you modify the setpoint SP_RE.	
	Longer run time	Shorter run time
	The values are effective after a cycle of the CPU or FM. The longer cycle is decisive.	A maximum of 3 to 4 cycles of the CPU or FM are required until the values are effective. The longer cycle is decisive
	All operating parameters are transferred	Only the operating parameters up to and including LMN_DN are transferred. In automatic mode, only the setpoint SP_RE is transferred. In manual mode, only the manual manipulated value LMN_RE is transferred.

Parameters	TRUE	FALSE
READ_OUT	The static variables of the structure OUT are read cyclically from FM 355-2 with the instructions WRREC and RDREC and stored in the structure OUT.	The static variables of the structure OUT are transferred via the input and output areas of the module and stored in the structure OUT.
	In the following situation READ_OUT is automatically TRUE: • During an optimization QTUN_ON = TRUE • When you modify the setpoint SP_RE	This is not possible if two instances of the instruction FMT_PID access the same channel number of a module.
	Longer processing time	Shorter processing time
	All parameters are read from FM 355	The following parameters are **not** read from FM 355 • SP (setpoint from the FM) • ER (error signal) • DISV (disturbance variable) • LMN_A • LMN_B • PHASE • STATUS_H • STATUS_C • STATUS_D • ZONE_TUN
READ_PAR	All static variables of the structure PAR are read from the FM 355-2 an stored in the instance DB on the CPU.	No transfer of the controller parameters to the CPU You can change the individual parameters in the user program.
LOAD_PAR	All static variables of the structure PAR are downloaded from the CPU to the FM 355-2. This is necessary if the control parameters on the CPU have changed based on the process states.	No transfer of the controller parameters to the FM 355-2

After data transfer has been completed, the parameters LOAD_OP, READ_OUT, LOAD_PAR and READ_PAR are reset to FALSE.

Use in distributed I/O

When the FM 355-2 is used in distributed I/O, the data transfer and the resetting of the parameters can take a few call cycles.

Processing times FMT_PID

During program controlled re-configuring (LOAD_PAR, LOAD_OP) of the FM 355-2 by FMT_PID, its runtime is extended. The new parameters are always immediately effective.

Boundary conditions			Processing time in	
READ_OUT	LOAD_OP	LOAD_PAR READ_PAR	CPU 315-2-DP	CPU 416-2 DP
FALSE	FALSE	FALSE	0.65 ms	0.04 ms
TRUE	FALSE	FALSE	2.85 ms	0.30 ms

Boundary conditions			Processing time in	
READ_OUT	LOAD_OP	LOAD_PAR READ_PAR	CPU 315-2-DP	CPU 416-2 DP
TRUE	TRUE	FALSE	4.56 ms	0.55 ms
FALSE	FALSE	TRUE	3.58 ms*)	0.30 ms
TRUE	FALSE	TRUE	5.8 ms*)	0.56 ms
TRUE	TRUE	TRUE	7.41 ms*)	0.82 ms
*) If READ_PAR is set instead of LOAD_PAR, the CPU 315-2 DP processing time is increased by 0.35 ms.				

See also

Block diagrams (Page 6013)

Input parameters FMT_PID (S7-300, S7-400)

Table 4-247

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	At this input you will find the module address resulting from configuration under STEP 7.
CHANNEL	2.0	INT	0	Number of the controller channel to which the instance DB is referenced. Values from 0 to 3 are permitted

See also

Block diagrams (Page 6013)

Output parameters FMT_PID (S7-300, S7-400)

Table 4-248

Parameters	Address	Data type	Default	Description
QMOD_F	4.0	BOOL	FALSE	The instruction checks that the reading and writing of a data record has been completed. If errors are identified, the QMOD_F output is set. The cause of errors can be: An incorrect module address on the MOD_ADDR parameter, an incorrect channel number on the CHANNEL parameter or a defective module.
RET_VALU	6.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the instructions RDREC and WRREC.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

In/out parameters FMT_PID (S7-300, S7-400)

Table 4-249

Parameters	Address	Data type	Default	Description
COM_RST	8.0	BOOL	FALSE	If COM_RST = TRUE, the instruction FMT_PID executes an initialization routine and resets COM_RST. This initialization routine is imperative every time the CPU starts up.
LOAD_OP	8.1	BOOL	FALSE	If the in/out parameter LOAD_OP is set, the operating parameters are downloaded to the module, the output parameters are read and the in/out parameters are reset.
READ_OUT	8.2	BOOL	FALSE	If the in/out parameter READ_OUT is set, the output parameters are read from the module and stored in the OUT structure of the instance DB and the I/O parameters are reset.
LOAD_PAR	8.3	BOOL	FALSE	If the in/out parameter LOAD_PAR is set, the controller parameters are downloaded to the module and the in/out parameter is reset.
READ_PAR	8.4	BOOL	FALSE	If the parameter READ_PAR = TRUE is set, then the controller parameters are read from FM 355-2 and stored in the PAR structure of the instance DB. The in/out parameter is then reset.

See also

Block diagrams (Page 6013)

Static variables FMT_PID.OP (S7-300, S7-400)

Table 4-250

Parameters	Address	Data type	Default	Description
vers_nr	10.0	WORD	W#16#3230	The vers_nr parameter may not be modified by users. It identifies the start of the operating parameters that are transferred to the module, if LOAD_OP = TRUE is set.
SP_RE	12.0	REAL	0.0	An external setpoint is interconnected with the controller at input SP_RE.
LMN_RE	16.0	REAL	0.0	An external manipulated value is interconnected with the controller at input LMN_RE. Values from -100.0 to 100.00 (%) are permitted.
SAFE_ON	20.0	BOOL	FALSE	When SAFE_ON is set, a safety value is applied as manipulated value.
LMNTRKON	20.1	BOOL	FALSE	If LMNTRKON is set, the manipulated value is corrected to an analog input (AI) (This does not apply to step controllers without analog position feedback)
LMN_REON	20.2	BOOL	FALSE	If LMN_REON is set, the external manipulated value LMN_RE is applied (manual mode).
LMNRHSRE	20.3	BOOL	FALSE	The "Control valve at high endstop" signal can be interconnected with a digital input of the FM 355-2 or at the LMNRHSRE input. LMNRHSRE = TRUE means: The control valve is at high endstop. (Applies to step controllers only)

Parameters	Address	Data type	Default	Description
LMNRLSRE	20.4	BOOL	FALSE	The "Control valve at high endstop" signal can be interconnected with a digital input of the FM 355-2 or at the LMNRLSRE input. LMNRLSRE = TRUE means: The control valve is at the low endstop. (Applies to step controllers only)
LMNS_ON	20.5	BOOL	FALSE	Enable manipulated value operation (Applies to step controllers only; manual mode).
LMN_UP	20.6	BOOL	FALSE	Manipulated value signal up operation (Applies to step controllers only).
LMN_DN	20.7	BOOL	FALSE	Manipulated value signal down operation (Applies to step controllers only).
SAVE_PAR	22.0	BOOL	FALSE	Saves the current contoller parameters to the output parameters of the instruction FMT_TUN. • FMT_TUN.SAV_PFAC=PFAC_SP • FMT_TUN.SAV_GAIN =GAIN • FMT_TUN.SAV_TI =TI • FMT_TUN.SAV_TD = TD • FMT_TUN.SAV_D_F =D_F • FMT_TUN.SAV_CONZ = CON_ZONE • FMT_TUN.SAV_RATI =RATIOFAC • FMT_TUN.SAV_CZON=CONZ_ON • FMT_TUN.SAV_PSEL=P_SEL
UNDO_PAR	22.1	BOOL	FALSE	The saved controller parameters are retrieved again, also in automatic mode. • PFAC_SP= FMT_TUN.SAV_PFAC • GAIN = FMT_TUN.SAV_GAIN • TI = FMT_TUN.SAV_TI • TD = FMT_TUN.SAV_TD • D_F = FMT_TUN.SAV_D_F • CON_ZONE = FMT_TUN.SAV_CONZ • RATIOFAC = FMT_TUN.SAV_RATI • CONZ_ON= FMT_TUN.SAV_CZON • P_SEL= FMT_TUN.SAV_PSEL
LOAD_PID	22.2	BOOL	FALSE	If PID_ON=TRUE, it loads the PID parameters (PID_GAIN, PID_TI, PID_TD). If PID_ON=FALSE, the PI parameters (PI_GAIN, PI_TI) are loaded. (Even in automatic mode)
TUN_ON	22.3	BOOL	FALSE	TUN_ON=TRUE switches the FM 355-2 to ready for optimization (PHASE=1): The manipulated value is averaged until the manipulated value excitation is applied.
TUN_ST	22.4	BOOL	FALSE	If the setpoint is to remain constant during controller optimization at the operating point, a manipulated value step-change by the amount of TUN_DLMN is enabled by TUN_ST=TRUE (PHASE = 1 or PHASE = 2)
TUN_CST	22.5	BOOL	FALSE	Starts cooling optimization by applying a manipulated manipulated value step-change in the amount of TUN_CLMN (PHASE = 1 or PHASE = 2)

See also

Description FMT_PID (Page 6002)
Block diagrams (Page 6013)

Static variables FMT_PID.PAR (S7-300, S7-400)

Table 4-251

Parameters	Address	Data type	Default	Description
vers_nr	24.0	WORD	W#16#3230	The vers_nr parameter may not be modified by users. It identifies the start of the controller parameters, which are read from the FM with READ_PAR = TRUE and stored in the instance DB or transferred to the FM with LOAD_PAR = TRUE.
P_SEL	26.0	BOOL	TRUE	It is possible to enable or disable PID actions individually in the PID algorithm. The P-action is enabled if the input P_SEL is set.
MONERSEL	26.1	BOOL	FALSE	The controller possesses a limit value detector that can be applied either for the process value or for the error signal. If MONERSEL is set, the error signal is monitored.
PID_ON	26.2	BOOL	TRUE	After optimization or in the case of LOAD_PID =TRUE, the PID parameters are enabled. If PID_ON =FALSE, the PI parameters are loaded. With certain process types it is nevertheless possible that only a PI controller will be designed despite PID_ON = TRUE.
CONZ_ON	26.3	BOOL	FALSE	TRUE = control zone enabled FALSE = control zone off
D_EL_SEL	28.0	INT	16	The D-action element in the PID algorithm can be assigned to a separate input This is selected at the D_EL_SEL parameter. • D_EL_SEL = 0 to D_EL_SEL = 3: Analog input 0 to 3 • D_EL_SEL = 16: error signal • D_EL_SEL = 17: negative process value
SP_HLM	30.0	REAL	100.0	The setpoint is always limited to a high and a low limit. The SP_HLM parameter specifies the high limit. SP_HLM > SP_LLM
SP_LLM	34.0	REAL	0.0	The setpoint is always limited to a high and a low limit. The SP_LLM parameter specifies the low limit. SP_LLM < SP_HLM
H_ALM	38.0	REAL	100.0	Highest limit for monitoring the process value or the error signal. H_ALM > H_WRN
H_WRN	42.0	REAL	90.0	Second to highest limit for monitoring the process value or the error signal. H_WRN > H_ALM
L_WRN	46.0	REAL	10.0	Second to lowest limit for monitoring the process value or the error signal. Values from H_WRN to L_ALM are permitted.
L_ALM	50.0	REAL	0.0	Lowest limit for monitoring the process value or the error signal. Values from L_ALM to L_WRN are permitted.
HYS	54.0	REAL	1.0	The parameters can be assigned for a hysteresis to prevent the monitoring displays from flickering. HYS >= 0.0

Parameters	Address	Data type	Default	Description
DEADB_W	58.0	REAL	0.0	A dead band is applied to the error signal. The DEADB_W parameter determines the size of the dead band. DEADB_W >= 0.0
PFAC_SP	62.0	REAL	1.0	PFAC_SP specifies the effective P-action when there is a setpoint change. Values from 0.0 to 1.0 are permitted.
GAIN	66.0	REAL	1.0	The GAIN parameter specifies the controller gain. The control direction is inverted by means of a negative sign of GAIN.
TI	70.0	REAL	3000.0	The TI parameter (integral-action time) determines the time response of the integrator. If TI = 0.0, the integral action is disabled. TI = 0.0 or TI >= 0.5
TD	74.0	REAL	0.0	The TD (rate time) paramter determines the time response of the derivative action (D-action element). If TD = 0.0, the derivative action is disabled TD = 0.0 or TD >= 1.0
D_F	78.0	REAL	5.0	The time constant TD/D_F that is effective for the D-action element is limited internally to ≥ sampling time/2. Values from 5.0 to 10.0 are permitted.
LMN_SAFE	82.0	REAL	0.0	A safety value can be parameterized for the manipulated value. Values from -100.0 to 100.0 (%) are permitted.
LMN_HLM	86.0	REAL	100.0	The manipulated value is always limited to a high and a low limit. The LMN_HLM parameter specifies the high limit. (This does not apply to step controllers without analog position feedback). Values from LMN_LLM to 100.0 (%) are permitted.
LMN_LLM	90.0	REAL	0.0	The manipulated value is always limited to a high and a low limit. The LMN_LLM parameter specifies the low limit. (This does not apply to step controllers without analog position feedback). Values from -100.0 to LMN_HLM (%) are permitted.
MTR_TM	94.0	REAL	60.0	Runtime of the control valve from stop to stop (Applies to step controllers only). MTR_TM >= 0.001
PULSE_TM	98.0	REAL	0.0	Minimum pulse duration (Applies to step controllers or pulse controllers only)
BREAK_TM	102.0	REAL	0.0	Minimum pulse duration (Applies to step controllers or pulse controllers only)
RATIOFAC	106.0	REAL	0.0	Ratio of heating gain/cooling gain for the process. RATIOFAC <> 0.0 has the following effect: • LMN = LMN * RATIOFAC for LMN<0.0; • CON_ZONE 50 % greater Values from 0.01 to 100.0 or 0.0 are permitted.
CON_ZONE	110.0	REAL	100.0	Control zone width If ER >= CON_ZONE, then LMN = LMN_HLM. If ER <= -CON_ZONE, then LMN = LMN_LLM. If RATIOFAC <> 0.0 and ER <= -CON_ZONE/RATIOFAC, then LMN = LMN_LLM. CON_ZONE > 0.0

Parameters	Address	Data type	Default	Description
TUN_DLMN	114.0	REAL	20.0	Process excitation for controller optimization results from a manipulated value step-change of TUN_DLMN. Values from -100.0 to 100.0 (%) are permitted.
TUN_CLMN	118.0	REAL	-20.0	Manipulated variable step-change after start of cooling optimization by TUN_CST. Values from -100.0 to 100.0 (%) are permitted.

See also

Block diagrams (Page 6013)

Static variables FMT_PID.OUT (S7-300, S7-400)

Table 4-252

Parameters	Address	Data type	Default	Description
vers_nr	122.0	WORD	W#16#3230	The vers_nr parameter may not be modified by users. It marks the start of the output parameter that is read by the module if READ_OUT = TRUE is set.
PV	124.0	REAL	0.0	Effective process value.
LMN	128.0	REAL	0.0	The effective output value is output at the LMN output. In the case of step controllers without analog position feedback, the unlimited P- + D-action is output at the LMN output. Values from -100.0 to 100.0 (%) are permitted.
QLMNSAFE	132.1	BOOL	FALSE	If the QLMNSAFE output is set, the safety output value is output as output value.
QLMNTRK	132.3	BOOL	FALSE	Output QLMNTRK indicates if the output value is being corrected via an analog input.
QLMN_RE	132.4	BOOL	FALSE	The QLMN_RE output indicates if LMN_REON is set and the external output value LMN_RE is applied as the output value.
QLMNR_HS	132.5	BOOL	FALSE	QLMNR_HS = TRUE means: The control valve is at high endstop. (Applies to step controllers only)
QLMNR_LS	132.6	BOOL	FALSE	QLMNR_LS = TRUE means: The control valve is at the low endstop. (Applies to step controllers only)
QLMNR_ON	132.7	BOOL	FALSE	QLMNR_ON = TRUE means: Step controller with position feedback.
QSTEPCON	133.0	BOOL	FALSE	Step controller / pulse controller • QSTEPCON = 0: Pulse controller or continuous controller • QSTEPCON = 1: Step controller
QSPR	133.1	BOOL	FALSE	If the QSPR output is set, the continuous controller operates in split-range mode.
QMAN_FC	133.3	BOOL	FALSE	This controller is a master controller. It is manually followed up to the process value of a slave controller, or its I-action is disabled because the setpoint or manipulated variable of the slave controller has reached the limit.
QH_ALM	134.0	BOOL	FALSE	High limit H_ALM overshot by process value or error value.

Parameters	Address	Data type	Default	Description
QH_WRN	134.1	BOOL	FALSE	Process value or control deviation exceeds second to highest limit H_WRN.
QL_WRN	134.2	BOOL	FALSE	Process value or control deviation has dropped below the second to lowest limit L_WRN.
QL_ALM	134.3	BOOL	FALSE	Process value or control deviation has dropped below the lowest limit L_ALM.
QLMN_HLM	134.4	BOOL	FALSE	The QLMN_HLM output reports that the high output value limit LMN_HLM has been reached. (This does not apply to step controllers without analog position feedback).
QLMN_LLM	134.5	BOOL	FALSE	The QLMN_LLM output reports that the low output value limit LMN_LLM has been reached. (This does not apply to step controllers without analog position feedback)
QPAR_F	134.6	BOOL	FALSE	The module checks the reliability of the parameters. A parameter assignment error is indicated at output QPAR_F.
QCH_F	134.7	BOOL	FALSE	The QCH_F output is set if the controller channel is unable to supply any valid results. A channel error (for example, wire break) is also set if QPAR_F = TRUE or QMOD_F = TRUE. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.
QUPRLM	135.0	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the QUPRLM output is set, the setpoint rise is limited.
QDNRLM	135.1	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the QDNRLM output is set, the setpoint decay is limited.
QSP_HLM	135.2	BOOL	FALSE	The QSP_HLM output reports that the high setpoint limit SP_HLM has been reached.
QSP_LLM	135.3	BOOL	FALSE	The QSP_LLM output reports that the low setpoint limit SP_LLM has been reached.
QLMNUP	135.4	BOOL	FALSE	"Output value signal up" output (Applies to step controllers or pulse controllers only)
QLMNDN	135.5	BOOL	FALSE	"Output value signal down" output (Applies to step controllers or pulse controllers only)
QTUN_ON	135.6	BOOL	FALSE	QTUN_ON = TRUE indicates that tuning is in progress.
PAR_ACT	135.7	BOOL	FALSE	Updating controller parameters
SP	136.0	REAL	0.0	Effective setpoint
ER	140.0	REAL	0.0	Effective control deviation (SP- PV)
DISV	144.0	REAL	0.0	Effective disturbance variable Values from -100.0 to 100.0 (%) are permitted.
LMN_A	148.0	REAL	0.0	Continuous or pulse controllers display the output value A of the split-range function at the LMN_A output; step controllers with analog position feedback display the position feedback there. Values from -100.0 to 100.0 (%) are permitted.
LMN_B	152.0	REAL	0.0	Continuous or pulse controllers display the output value B of the split-range function at output LMN_B. Values from -100.0 to 100.0 (%) are permitted.

Parameters	Address	Data type	Default	Description
PHASE	156.0	INT	0	Indicates the controller tuning phases. • PHASE = -1: Canceling of controller tuning, for example with parallel tuning of several channels. The value lasts for a few cycles and is used as cancelation signal between the instruction FMT_PID and the module. • PHASE = 0: No tuning mode; automatic or manual mode • PHASE = 1: Ready to start tuning; check parameters, wait for excitation, measure the sampling times • PHASE = 2: Actual tuning: Searching for point of inflection with constant output value. • PHASE = 3: Calculating process parameters. Saving valid controller parameters prior to tuning. • PHASE = 4: Controller design • PHASE = 5: Following up the controller to the new manipulated variable • PHASE = 6: Following up the controller to the new manipulated variable • PHASE = 7: Check of process type, if process type II or III was determined (during heating optimiation only).
STATUS_H	158.0	INT	0	AUTOHOTSPOT displays a diagnostic value via the search for the point of inflection during heating process.
STATUS_C	160.0	INT	0	AUTOHOTSPOT displays a diagnostic value via the search for the point of inflection during cooling process.
STATUS_D	162.0	INT	0	AUTOHOTSPOT indicates a diagnostic value of the controller design during the heating process.
ZONE_TUN	164.0	WORD	W#16#0	In HEX code, each of the four digits represents one channel, arranged in the following order from left to right: Channel 0, 1, 2, 3. ZONE_TUN = 0 means that the selected channel will not be tuned as a group with other channels. A value <> 0000 shows - in each case with a 1 - the channels that will be tuned as a group.

See also

Block diagrams (Page 6013)

Block diagrams (S7-300, S7-400)

Overview

The following figures show at which points the parameters of the instruction FMT_PID act.

As far as three-component controllers and ratio/mixing controllers are concerned, the parameters affect the same point as with fixed setpoint or cascade controllers. This also applies to the parameters that are equally in place for continuous controllers, controllers with pulse output as well as step controllers. It is generally correct that identical buttons also contain identical parameters. For reasons of clarity, not all structure diagrams are depicted, thereore, and not all parameters are marked in all figures.

However, the parameters of the instruction FMT_PID are contained in all the figures – with the exception of the parameters MOD_ADDR, CHANNEL, QMOD_F, QPAR_F, QCH_F, QLMNR_ON, RET_VALU, COM_RST, LOAD_PAR, READ_PAR, READ_OUT and LOAD_OP.

The following figures show the places in the module affected by the parameters of the FMT_PID.

Action of the input parameters

The following figure shows the control deviation for a fixed-value or cascade controller.

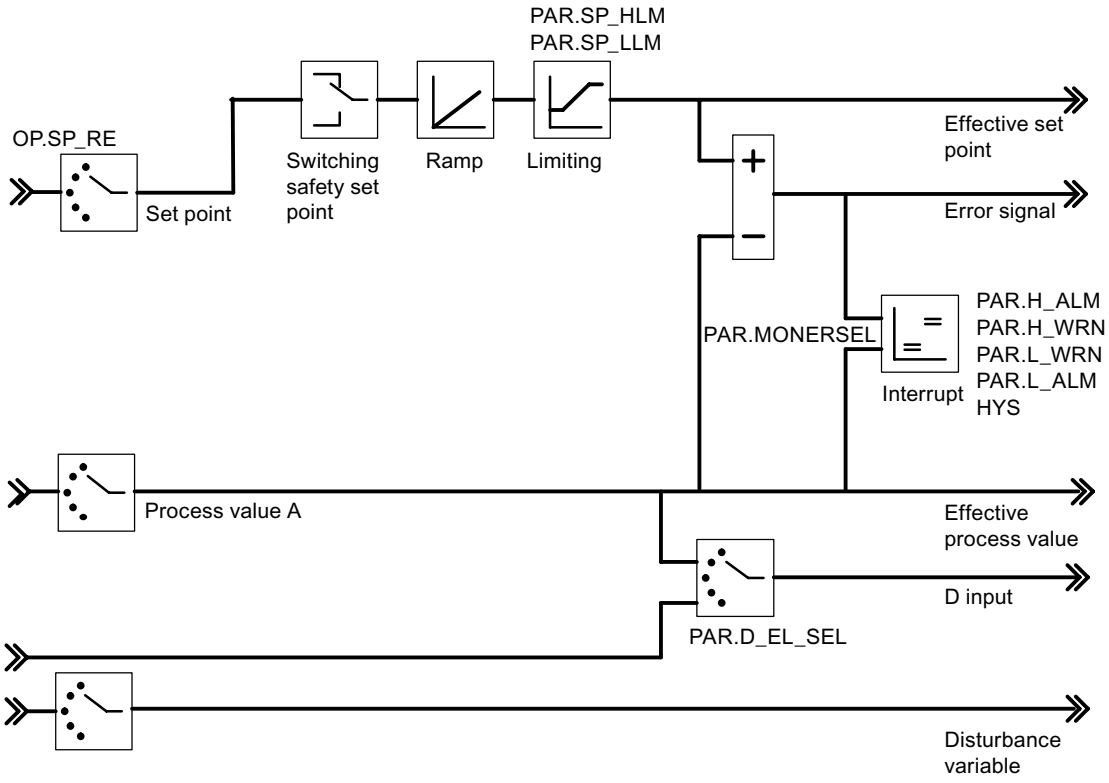

The following figure shows the block diagram of the control algorithm.

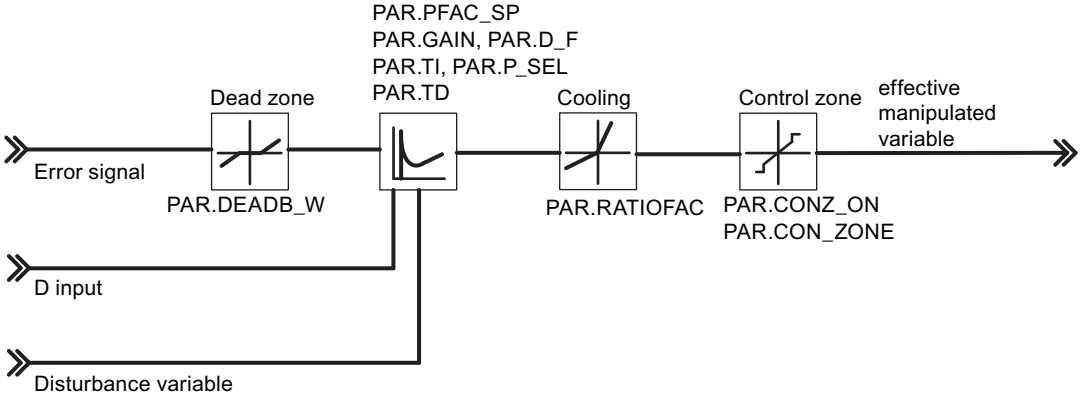

The following figure shows the controller output of the continuous controller (FM 355-2 C).

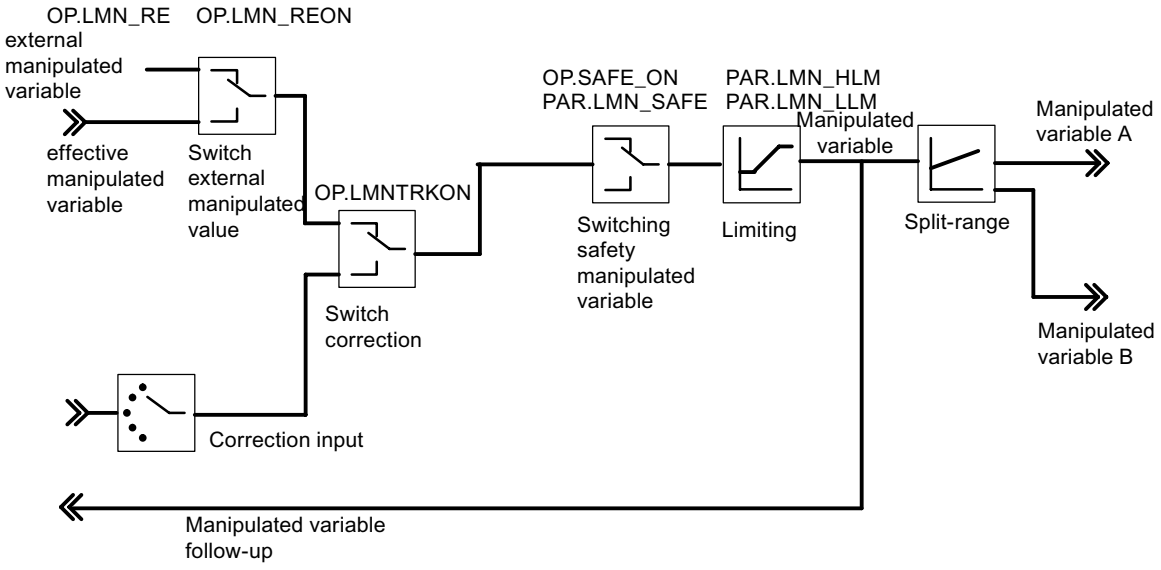

The following figure shows the controller output of the pulse controller (FM 355-2 S).

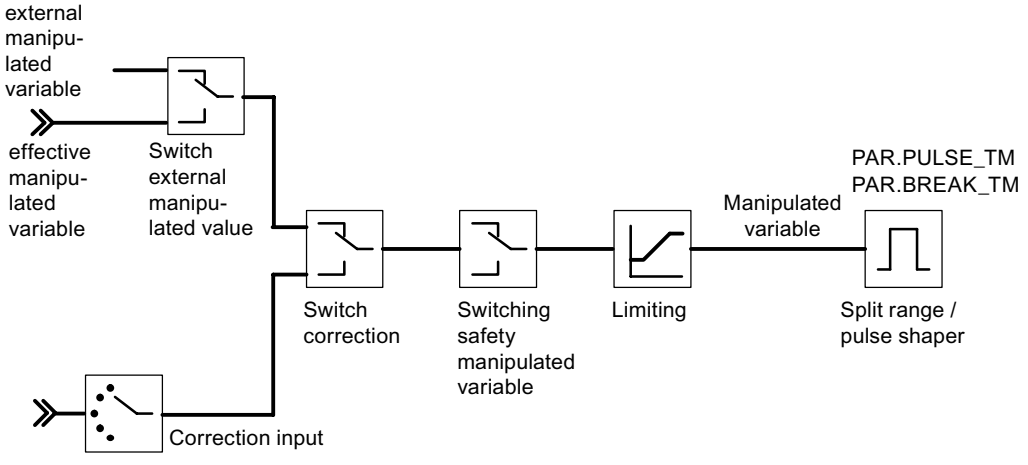

The following figure shows the controller output of the step controller with position feedback (FM 355-2 S).

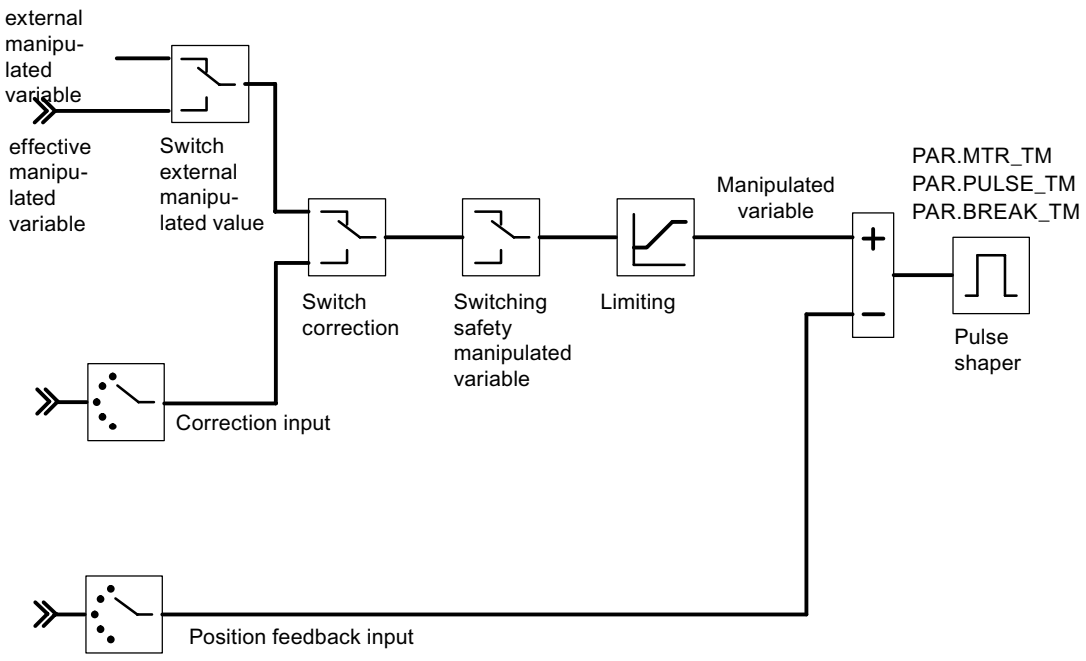

The following figure shows the controller output of the step controller without position feedback (FM 355-2 S).

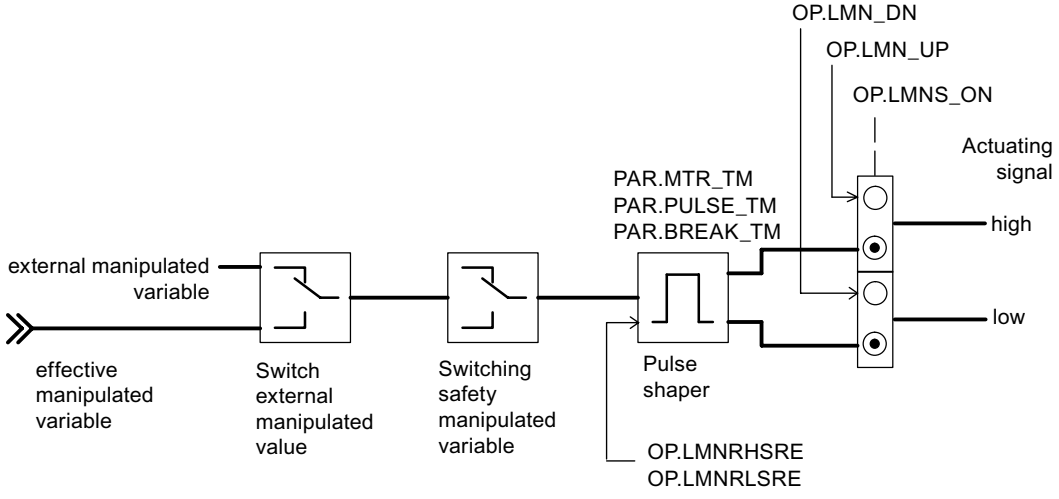

Generating output parameters

The following figures show at which points in the module the output parameters of the instruction FMT_PID are generated.

The following figure shows the control deviation for a fixed-value or cascade controller.

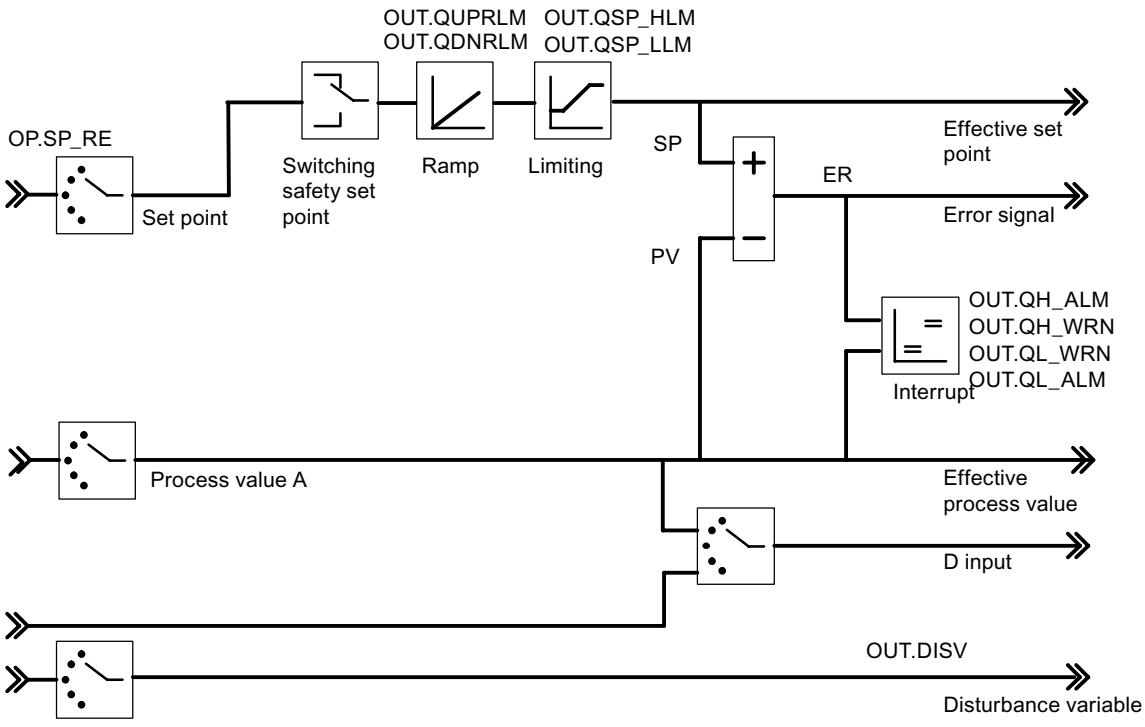

The following figure shows the controller output of the continuous controller (FM 355-2 C).

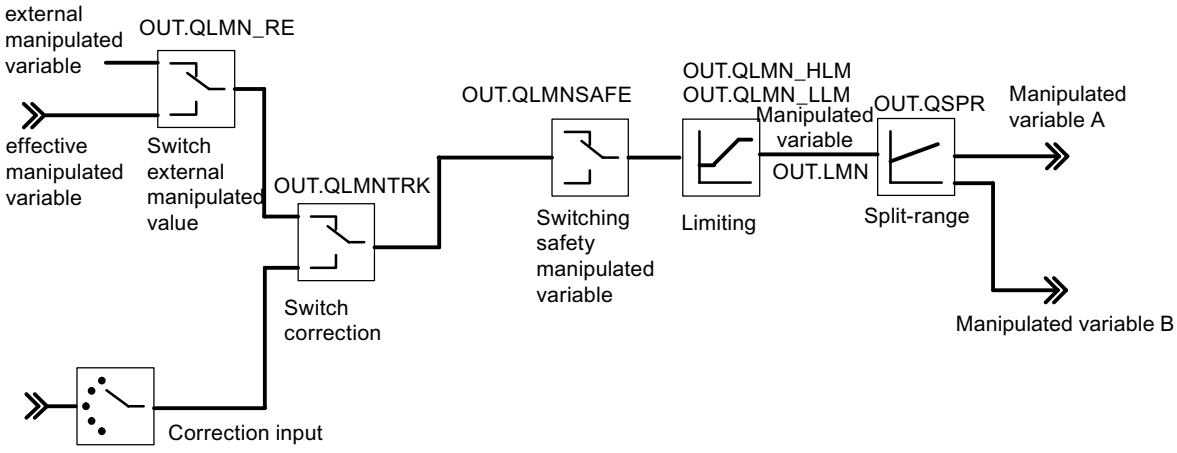

The following figure shows the controller output of the pulse controller (FM 355-2 S).

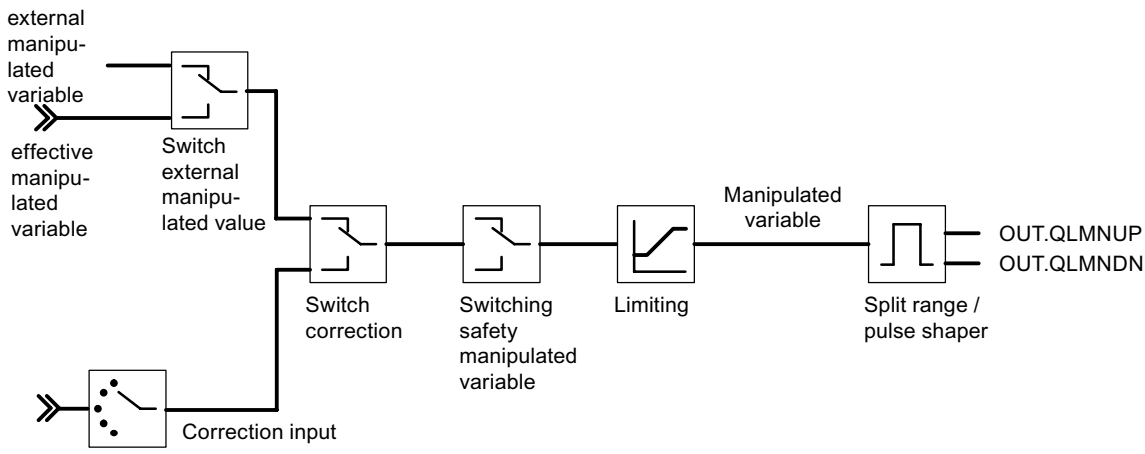

The following figure shows the controller output of the step controller with position feedback (FM 355-2 S).

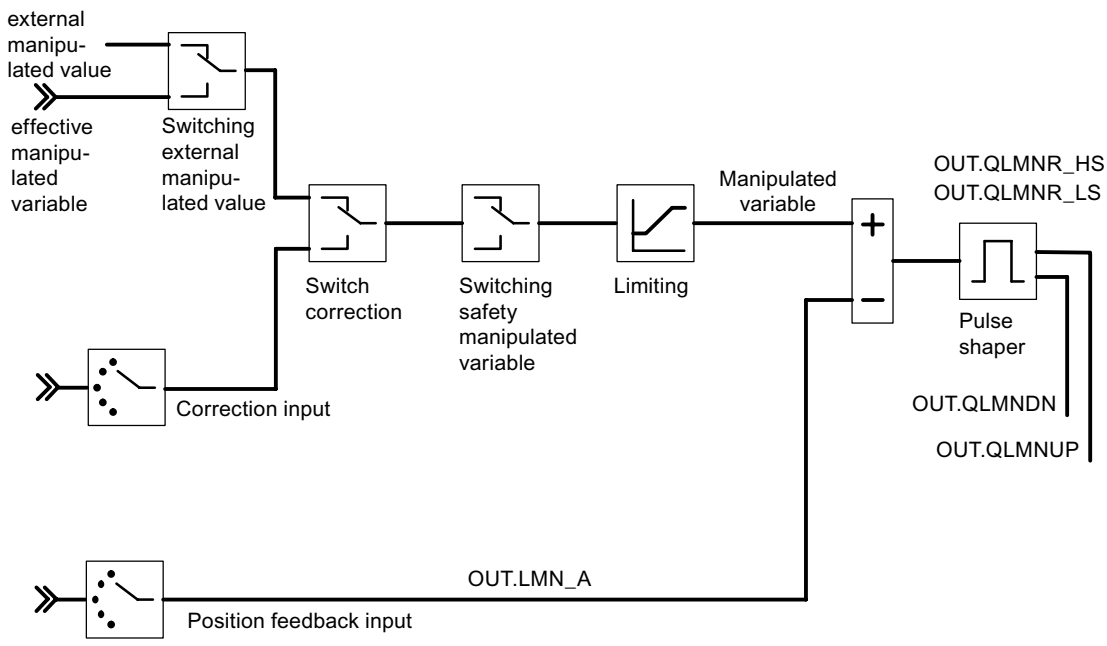

The following figure shows the controller output of the step controller without position feedback (FM 355-2 S).

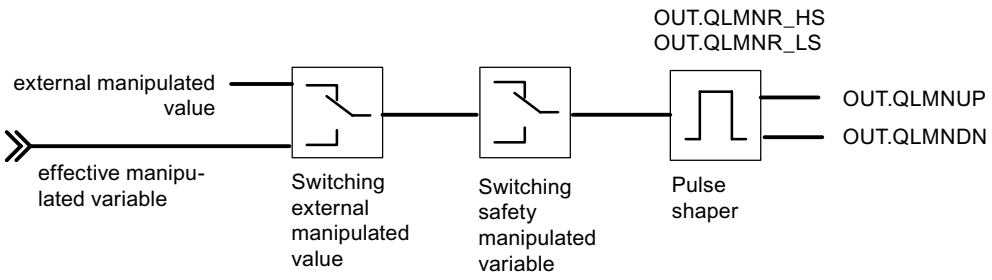

See also

Description FMT_PID (Page 6002)

Parameters STATUS_H (S7-300, S7-400)

STATUS_H indicates a diagnostic value via the search for the point of inflection during the heating process.

If a diagnostic value 2xxxx is displayed, optimization is completed but the controller parameters are uncertain.

If a diagnostic value 3xxxx is displayed, the optimization is canceled (PHASE = 0 and TUN_ON = FALSE) and the controller switches to the previous operating mode.

STATUS_ H	Description	Remedy
0	Default or no or still no new controller parameters	
10000	Optimization completed + suitable controller parameters found	
2xxxx	Optimization completed + controller parameters uncertain	
2xxx1	effective motor actuating time >= 65% of the point of inflection time T_P_INF	Compensation of TU and T_P_INF is not performed any more. This leads to the design of a softer controller
2xx2x	Point of inflection not reached (only with excitation via setpoint step-change)	If the controller oscillates, reduce the controller parameters or repeat test with lower TUN_DLMN.
2x1xx	Estimation error TU <= 3*Sampling time	Repeat test.
2x2xx	Serious estimation error (TU < sampling time)	Repeat test. Special case PT1-only process: Do not repeat the test; reduce the controller parameters if necessary
2x3xx	Estimation error TU too high	Repeat test under better conditions.
21xxx	Estimation error N_PTN < 1	Repeat test under better conditions.
22xxx	Estimation error N_PTN > 10	Repeat test under better conditions.
3xxxx	Optimization interrupted in phase 1 due to incorrect parameter assignment	
30001	Simultaneous setting of TUN_ON and TUN_ST	Restart optimization.

STATUS_ H	Description	Remedy
30002	|TUN_DLMN| < 5 % or value of effective LMN modification < 5 % • Transition in phase 99 (Phase 0 in a parallel optimization process) • Output LMN = LMN0	Either you configured for TUN_DLMN a value that is too low, or the manipulated variable was near to a control limit prior to start of optimization. In this case, you must prevent the controller from settling to the new setpoint and from unnecessary leaving the stationary operating point (not possible with parallel optimization process). You should now proceed as follows: • If necessary, cancel setpoint and set TUN_ON to FALSE. • TUN_DLMN correct. • If TUN_DLMN >= 5%, check manipulated value limits. • Restart optimization.
30004	|TUN_DLMN| < 5 % or effective LMN modification < 5 % The effective manipulated value difference limited by split–range limits and not by manipulated value limits. • Transition in phase 99 (Phase 0 in a parallel optimization process) • Output LMN = LMN0	See STATUS_H=30002: Also take into account that, for example, with LMN_A < 5 % a heating optimization with negative TUN_DLMN is not possible (reason: the cooling power must not be adjusted).
30009	Safety mode	Close safety mode and restart optimization.

Note

At the end of phase 1, STATUS_H = 0 is reset.

If you cancel optimization in phase 2, STATUS_H = 0. However, STATUS_D still displays the status of the last controller calculation.

See also

Block diagrams (Page 6013)

Parameters STATUS_C (S7-300, S7-400)

STATUS_C indicates a diagnostic value via the search for the point of inflection during cooling process.

If a diagnostic value 2xxxx is displayed, optimization is completed, but the controller parameters are uncertain.

If a diagnostic value 3xxxx is displayed, the optimization is canceled (PHASE = 0 and TUN_ON = FALSE) and the controller switches to the previous operating mode.

STATUS_ C	Description	Remedy
0	Default or no or still no new controller parameters	
10000	Optimization completed + suitable controller parameters found	
2xxxx	Optimization completed + controller parameters uncertain	
2x3xx	Estimation error TU too high	Repeat test under better conditions.
21xxx	Estimation error N_PTN < 1	Repeat test under better conditions.
22xxx	Estimation error N_PTN > 10	Repeat test under better conditions.
3xxxx	Optimization interrupted in phase 1 due to incorrect parameter assignment	
30001	Simultaneous setting of TUN_ON and TUN_CST	Restart optimization.
30002	Only for step controllers without analog position feedback: |TUN_CLMN| < 5 % or value of effective LMN modification < 5 % • Transition to phase 0 • TUN_ON = FALSE	Either you configured for TUN_CLMN a value that is too low, or the manipulated variable was near to a control limit prior to start of optimization. In this case, you must prevent the controller from settling to the new setpoint and from unnecessary leaving the stationary operating point (not possible with parallel optimization process). You should now proceed as follows: • If necessary cancel setpoint and set TUN_ON to FALSE. • TUN_CLMN correct. • Check the manipulated value limits, if | TUN_CLMN| >= 5% and the sign was correct. • Restart optimization.
30003	TUN_CLMN <= -5%, butLMN_LLM > -5%. • Transition to phase 0 • TUN_ON = FALSE	The low limit LMN_LLM is too high (e.g. 0%) and therefore prevents the output of cooling power. • Correct low limit LMN_LLM. • Restart optimization.
30004	TUN_CLMN > -5 % or effective LMN modification > -5 % The effective manipulated value difference limited by split–range limits and not by manipulated value limits. • Transition to phase 0 • TUN_ON = FALSE	See STATUS_C=30002: Also take into account that, for example, with LMN_A < 5 % a heating optimization with negative TUN_DLMN is not possible (reason: the cooling power must not be adjusted).
30008	TUN_CST but without previous heating optimization • Transition to phase 0 • TUN_ON = FALSE	First start heating optimization
30009	Safety mode	Close safety mode and restart optimization.

See also

Block diagrams (Page 6013)

Parameters STATUS_D (S7-300, S7-400)

STATUS_D	Description
0	No controller parameters have been calculated.
110	N_PTN <= 1.5 Process type I fast
121	N_PTN > 1.5 Process type I
122	N_PTN = 1.9 process type I after phase 7 (before N_PTN > 1.9)
200	N_PTN > 1.9 Process type II (transition range)
310	N_PTN >= 2.1 Process type III fast
320	N_PTN > 2.6 Process type III

Note

If you interrupt optimization in phase 2, STATUS_H = 0 is. However, STATUS_D still displays the status of the last controller calculation.

The higher the value of STATUS_D, the higher the order of the control process, the greater the ratio TU/TA and the gentler the controller parameters will be.

See also

Block diagrams (Page 6013)

FMT_PAR (S7-300, S7-400)

Description FMT_PAR (S7-300, S7-400)

The instruction FMT_PAR is used for online modification of parameters that cannot be defined via instruction FMT_PID.

Call

FMT_PAR has to be called in the same OB as all the other instructions that access the same FM 355-2.

To save runtime, FMT_PAR should only be called using LOAD=TRUE, if parameters are to be modified.

Operating principle

With FMT_PAR, you can modify one REAL-parameter and one of the INT parameters for each call.

At AUTOHOTSPOT specify the index numbers of the parameter you want to modify. At the input parameter VALUE_R or VALUE_I, specify the new values.

If you want to modify several parameters, you must call the same instance DB several times in succession using LOAD_PAR = TRUE and different index numbers.

If FM 355-2 is used in distributed I/O, it may take a few call cycles until the parameters are applied to FM 355-2. As long as the transfer has not been completed, parameter LOAD_PAR retains the value TRUE. When modifying parameters, keep repeating the call of instruction FMT_PAR until LOAD_PAR = FALSE is.

Note

Please note that parameters, modified using FMT_PAR,are overwritten at startup of CPU with the parameters from the system data.

Example

You want to modify in operating mode the startup time of the ramp for the reference variable as well as,depending on the process state, use different analog input values as process value.

- To configure the startup time for the reference variable at 10.0 call FMT_PAR in the current mode with INDEX = 30 and VALUE_R = 10.0.
- If you want to configure the analog input value 4 of the module as process value, call FMT_PAR with INDEX = 50 and VALUE_I = 4 during operation.

Input parameters FMT_PAR (S7-300, S7-400)

Table 4-253

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	0	Number of controller channel to which the instance DB refers. Values from 0 to 3 are permitted.
INDEX	4.0	INT	0	0 to 100
VALUE_R	6.0	REAL	0.0	Depending on respective parameter
VALUE_I	10.0	INT	0	Depending on respective parameter

Output parameters FMT_PAR (S7-300, S7-400)

Table 4-254

Parameters	Address	Data type	Default	Description
RET_VALU	12.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from parameter STATUS of instruction WRREC.

See also

WRREC: Write data record (Page 5226)

In/out parameters FMT_PAR (S7-300, S7-400)

Table 4-255

Parameters	Address	Data type	Default	Description
LOAD_PAR	+14.0	BOOL	FALSE	If the in/out parameters LOAD_PAR is set, VALUE_R or VALUE_I are written on the module.INDEX indicates which parameter is overwritten. Thereafter LOAD_PAR is reset.

Parameters INDEX (S7-300, S7-400)

INDEX	Description	Data type
0	no parameter selected	-
1	Filter time constant for the analog input	REAL
2	Scaling of analog input end of measuring range equals 100%	REAL
3	Scaling of analog input end of measuring range equals 0%	REAL
4	Polyline, interpolation point 1 supply side	REAL
5	Polyline, interpolation point 2 supply side	REAL
6	Polyline, interpolation point 3 supply side	REAL
7	Polyline, interpolation point 4 supply side	REAL
8	Polyline, interpolation point 5 supply side	REAL
9	Polyline, interpolation point 6 supply side	REAL
10	Polyline, interpolation point 7 supply side	REAL
11	Polyline, interpolation point 8 supply side	REAL
12	Polyline, interpolation point 9 supply side	REAL
13	Polyline, interpolation point 10 supply side	REAL
14	Polyline, interpolation point 11 supply side	REAL
15	Polyline, interpolation point 12 supply side	REAL
16	Polyline, interpolation point 13 supply side	REAL
17	Polyline, interpolation point 1 output side	REAL
18	Polyline, interpolation point 2 output side	REAL
19	Polyline, interpolation point 3 output side	REAL
20	Polyline, interpolation point 4 output side	REAL
21	Polyline, interpolation point 5 output side	REAL
22	Polyline, interpolation point 6 output side	REAL
23	Polyline, interpolation point 7 output side	REAL
24	Polyline, interpolation point 8 output side	REAL
25	Polyline, interpolation point 9 output side	REAL
26	Polyline, interpolation point 10 output side	REAL
27	Polyline, interpolation point 11 output side	REAL

INDEX	Description	Data type
28	Polyline, interpolation point 12 output side	REAL
29	Polyline, interpolation point 13 output side	REAL
30	Startup time of the ramp for the reference variable	REAL
31	Safety manipulated value	REAL
32	Offset for setpoint link (ratio/blending controller)	REAL
33	Factor for process value B (three component controller)	REAL
34	Factor for process value C (three component controller)	REAL
35	Offset for process value link (three-component controller)	REAL
36	Factor for disturbance variable link	REAL
37	Operating point	REAL
38	unused	REAL
39	Vertices for split range function Start of input signal A range	REAL
40	Vertices for split range function End of input signal A range	REAL
41	Vertices for split range function Start of output signal A range	REAL
42	Vertices for split range function: End of output signal A range	REAL
43	Vertices for split range function Start of input signal B range	REAL
44	Vertices for split range function End of input signal B range	REAL
45	Vertices for split range function: Start of output signal B range	REAL
46	Vertices for split range function: End of output signal B range	REAL
47	Minimum pulse time	REAL
48	Minimum break time	REAL
49	Selection of the reference variable SP or SP_RE for the controller • -1: Setpoint SP_RE of the instruction • 0 to 3: Analog input value 0 to 3 • 16 to 19: Manipulated variable (LMN) of controllers 0 to 3 • 32 to 35: Control variable A of controllers 0 to 3 • 48 to 51: Control variable B of controllers 0 to 3	INT
50	Selection of the main controlled variable process value A for the controller • -1: Process value A = 0.0 • 0 to 3: Analog input value 0 to 3	INT
51	Selection of the auxiliary controlled variable process value B for the controller • -1: Process value B = 0.0 • 0 to 3: Analog input value 0 to 3	INT
52	Selection of the auxiliary controlled variable process value C for the controller • -1: Process value C = 0.0 • 0 to 3: Analog input value 0 to 3	INT
53	Selection of the auxiliary controlled variable process value D for the controller • -1: Process value D = 0.0 • 0 to 3: Analog input value 0 to 3 • 16 to 19: Manipulated variable (LMN) of controllers 0 to 3	INT

INDEX	Description	Data type
54	Selection of the disturbance variable DISV for the controller • -1: Disturbance variable = 0.0 • 0 to 3: Analog input value 0 to 3	INT
55	Selection on the correction input switch. On the measuring point MP TRACKPER the value can be read off with tool tip. • -1: Position follow-up = 0.0 • 0 to 3: Analog input value 0 to 3	INT
56	Selection on the position feedback input switch for step controllers with position feedback. • -1: Position follow-up = 0.0 • 0 to 3: Analog input value 0 to 3	INT
57	Selecting the signal for transition to safety value for the manipulated value of the controller • -1: Can only be assigned via SAFE_ON of instruction FMT_PID • 0 to 7: Assignment via SAFE_ON parameter of the instruction FMT_PID ORed with digital input 0 to 7	INT
58	Selecting signal for switching over to adjustment function of the manipulated value of the controller. • -1: Can only be assigned via LMNTRKON of instruction FMT_PID • 0 to 7: Assignment via LMNTRKON parameter of the instruction FMT_PID ORed with digital input 0 to 7	INT
59	Selecting the signal for changing over the manipulated value of the controller to LMN_RE • -1: Can only be assigned via LMN_REON of instruction FMT_PID • 0 to 7: Assignment via LMN_REON parameter of the instruction FMT_PID ORed with digital input 0 to 7	INT
60	Selecting the high limit signal of the position feedback • -1: Can only be assigned via LMNRHSRE of instruction FMT_PID • 0 to 7: Assignment via LMNRHSRE parameter of the instruction FMT_PID ORed with digital input 0 to 7	INT
61	Selecting the low limit signal of the position feedback • -1: Can only be assigned via LMNRLSRE of instruction FMT_PID • 0 to 7: Assignment via LMNRLSRE parameter of the instruction FMT_PID ORed with digital input 0 to 7	INT

FMT_DS1 (S7-300, S7-400)

Description FMT_DS1 (S7-300, S7-400)

Instruction FMT_DS1 is available to you for reading the diagnostic data record DS1.

Mode of operation

If parameter READ_DS1 = TRUE, then diagnostic data record DS1 is read from FM 355-2 and written to the structure DS1 of the instance DB. Thereafter the in/out parameter is reset.

This only makes sense, if an error is reported in a channel in DS0.

Call

FMT_DS1 has to be called in the same OB as all the other instructions that access the same FM 355-2.

FMT_DS1 requires no initialization routine.

To always obtain the updated diagnostic values, set the parameter READ_DS1 cyclically to TRUE in OB 82.

After the diagnostic values have been successfully read, FMT_DS1 resets the parameter READ_DS1.

Input parameters FMT_DS1 (S7-300, S7-400)

Table 4-256

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

Output parameter FMT_DS1 (S7-300, S7-400)

Table 4-257

Parameters	Address	Data type	Default	Description
DS1	2.0	STRUCT		The diagnostic data record DS1 contains bytes from 0 to 12. The first 4 bytes are identical with diagnostic data record DS0. AUTOHOTSPOT
RET_VALU	16.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from parameter STATUS of instruction RDREC.

See also

RDREC: Read data record (Page 5224)

In/out parameters FMT_DS1 (S7-300, S7-400)

Table 4-258

Parameters	Address	Data type	Default	Description
READ_DS1	18.0	BOOL	FALSE	If parameter READ_DS1 = TRUE, then diagnostic data record DS1 is read from FM 355-2 and written to the structure DS1of the instance DB. Thereafter the in/out parameter is reset.

Assignments of DS1 (S7-300, S7-400)

The following table shows the assignments of diagnostic data record DS1. The first 4 bytes are identical to diagnostic data record DS0. Any bits not listed have no meaning and are zero.

Byte	Bit	Meaning	Remark	Event no.
0	0	Module fault	Is set at each diagnostics event	8:x:00
	1	Internal error	Is set at every internal error: • Time monitoring addressed • EPROM error • ADU/DAU error • Analog input hardware error	8:x:01
	2	External error	Is set at every external error: • Missing external auxiliary voltage • Parameter assignment error • Analog input wire break (only in range 4 to 20 mA) • Analog input undershoot of measuring range • Analog input overshoot of measuring range • Analog output load break • Analog output short circuit	8:x:02
	3	Error in one channel	See from byte 7 for further breakdown	8:x:03
	4	Missing external auxiliary voltage	24 V supply for FM 355-2 failed	8:x:04
	6	Unused	-	-
	7	Parameter assignment error	The module is unable to use a parameter. Reason: Parameters unknown or illegal combination of parameters.	8:x:07
1	0 ... 3	Module class	8 is always assigned.	-
	4	Channel-specific diagnosis	Is set, when the module can provide additional channel information and a channel error exists (see byte 7 to 12)	-
2	3	Time monitoring addressed (watchdog)	Hardware error	8:x:33
3	2	EPROM error	Module defect	8:x:42
	4	ADU/DAU error	Module defect	8:x:44
4	0 ... 7	Channel type	75H is always assigned.	-

Byte	Bit	Meaning	Remark		Event no.
5	0 ... 7	Length of the diagnostics information	8 is always assigned.		-
6	0 ... 7	Number of channels	5 always assigned (4 controllers + 1 reference channel)		-
7	0 ... 4	Channel error vector	One bit is assigned to each channel (0...3; 4 for reference channel).		-
8	0	Analog input hardware error	Channel-specific diagnostics channel 0		8:x:B0
	1	Unused			8:x:B1
	2	Analog input wire break (only in range 4 to 20 mA)			8:x:B2
	3	Unused			8:x:B3
	4	Analog input undershoot of measuring range			8:x:B4
	5	Analog input overshoot of measuring range			8:x:B5
	6	Analog output wire break		Only with the current output of the continuous-action controller.	8:x:B6
	7	Analog output short circuit		Only with the voltage output of the continuous-action controller	8:x:B7
9	0 ... 7	See byte 8	Channel-specific diagnostics channel 1		See above
10	0 ... 7	See byte 8	Channel-specific diagnostics channel 2		See above
11	0 ... 7	See byte 8	Channel-specific diagnostics channel 3		See above
12	0 ... 5	See byte 8	Diagnostics for reference channel		See above

FMT_TUN (S7-300, S7-400)

Description FMT_TUN (S7-300, S7-400)

With instruction FMT_TUN you obtain additional detailed information (e.g. backup controller parameters) during controller optimization.

Note

However, for performing controller optimization the instruction FMT_PID is adequate. You can start optimization and observe the status information with this instruction.

Call

FMT_TUN has to be called in the same OB as all the other instructions that access the same FM 355-2.

FMT_TUN requires no initialization routine.

Set parameter READ_OUT cyclically to TRUE to receive constantly the updated values.

After the parameters have been successfully read out, FMT_TUN resets the parameter READ_OUT.

Mode of operation

After downloading the controller parameters with LOAD_PAR = TRUE or the operating parameters with LOAD_OP = TRUE the parameters do not take effect immediately. Only after the cycle time of FM355-2 (depending on the number of channels up to 500 ms) the controller parameters are available for reading back with READ_PAR or the output parameters with READ_OUT. This also applies after downloading the system data (CPU transition from STOP to RUN) as well as when reading the output parameters FMT_TUN.

Call at distributed configuration

In the case of disturbed configuration and simultaneous call of FMT_PID and FMT_TUN, the following must be noted.

- LOAD_OP must not be set simultaneously at FMT_PID and FMT_TUN.
- READ_OUT must not be set simultaneously at FMT_PID and FMT_TUN.

Reasons: Both instructions access FM355-2 via the same data records. You must therefore make sure that at the same time only one of the two FBs reads or writes a data record at the same time.

Input parameter FMT_TUN (S7-300, S7-400)

Table 4-259

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	0	Number of controller channel to which the instance DB refers. Values from 0 to 3 are permitted.

Output parameters FMT_TUN (S7-300, S7-400)

Table 4-260

Parameters	Address	Data type	Default	Description
PI_GAIN	4.0	REAL	0.0	PI controller parameters
PI_TI	8.0	REAL	0.0	PI controller parameters
PID_GAIN	12.0	REAL	0.0	PID controller parameters
PID_TI	16.0	REAL	0.0	PID controller parameters

Parameters	Address	Data type	Default	Description
PID_TD	20.0	REAL	0.0	PID controller parameters
SAV_PFAC	24.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.PFAC_SP
SAV_GAIN	28.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.GAIN
SAV_TI	32.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.TI
SAV_TD	36.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.TD
SAV_D_F	40.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.D_T
SAV_RATI	44.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.RATIOFAC
SAV_CONZ	48.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.CON_ZONE
SAV_PSEL	52.0	BOOL	FALSE	Saved controller parameters FMT_PID.PAR.P_SEL
SAV_CZON	52.1	BOOL	FALSE	Saved controller parameters FMT_PID.PAR.CONZ_ON
RET_VALU	54.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from parameter STATUS of instruction RDREC.

See also

RDREC: Read data record (Page 5224)

In/out parameters FMT_TUN (S7-300, S7-400)

Table 4-261

Parameters	Address	Data type	Default	Description
READ_OUT	56.0	BOOL	FALSE	If the in/out parameter READ_OUT is set, the output parameters are read from FM 355-2.

Static variables FMT_TUN.OUT (S7-300, S7-400)

Table 4-262

Parameters	Address	Data type	Default	Description
vers_nr	58.0	WORD	W#16#3230	The vers_nr parameter may not be modified by users. It marks the start of the output parameter that is read by the module if READ_OUT = TRUE is set.
PV	60.0	REAL	0.0	Process value that becomes effective.
LMN	64.0	REAL	0.0	The effective manipulated value that is currently in effect is output at output LMN. For step controller without analog position feedback the unlimited P- + D-action is output at output LMN. Values from -100.0 to 100.0 (%) are permitted.
QLMNSAFE	68.1	BOOL	FALSE	If output QLMNSAFE is set, the safety manipulated value is output as manipulated value.

Parameters	Address	Data type	Default	Description
QLMNTRK	68.3	BOOL	FALSE	Output QLMNTRK indicates if the manipulated value is being compensated via an analog input.
QLMN_RE	68.4	BOOL	FALSE	The output QLMN_RE indicates if LMN_REON is set and the external manipulated value LMN_RE is applied as the manipulated value.
QLMNR_HS	68.5	BOOL	FALSE	QLMNR_HS = TRUE means: Control valve is at the high endstop. (Only in the case of step controllers)
QLMNR_LS	68.6	BOOL	FALSE	QLMNR_LS = TRUE means: The control valve is located at the low endstop. (Only in the case of step controllers)
QLMNR_ON	68.7	BOOL	FALSE	QLMNR_ON = TRUE means: Step controller with position feedback.
QSTEPCON	69.0	BOOL	FALSE	Step controllers/pulse controllers 0 = Pulse controller or continuous controller 1 = step controller
QSPR	69.1	BOOL	FALSE	If output QSPR is set, the continuous controller works in the split-range mode.
QMAN_FC	69.3	BOOL	FALSE	The controller is a master controller that is adjusted by the manual mode of a slave controller to its process value or its I-action is stopped, because the setpoint or manipulated value of the slave controller lies in the limit.
QH_ALM	70.0	BOOL	FALSE	High limit H_ALM exceeded by process value or error signal.
QH_WRN	70.1	BOOL	FALSE	Process value or error signal exceeds second to highest limit H_WRN.
QL_WRN	70.2	BOOL	FALSE	Process value or error signal has dropped below the second to lowest limit L_WRN.
QL_ALM	70.3	BOOL	FALSE	Process value or error signal has dropped below the lowest limit L_ALM.
QLMN_HLM	70.4	BOOL	FALSE	The output QLMN_HLM reports that the high manipulated value limit LMN_HLM. has been reached. (Does not apply to step controllers without analog position feedback).
QLMN_LLM	70.5	BOOL	FALSE	The output QLMN_LLM reports that the high manipulated value limit LMN_LLM has been reached. (Does not apply to step controllers without analog position feedback)
QPAR_F	70.6	BOOL	FALSE	The module checks the parameters for validity. A parameter assignment error is indicated on output QPAR_F.
QCH_F	70.7	BOOL	FALSE	The output QCH_F is set if the controller channel is unable to supply any valid results. A channel error (for example, wire break) is also set if QPAR_F = 1 or QMOD_F = 1. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.
QUPRLM	71.0	BOOL	FALSE	The setpoint is limited to a positive and negative gradient. If the output QUPRLM is set, the setpoint rise is limited.
QDNRLM	71.1	BOOL	FALSE	The setpoint is limited to a positive and negative gradient. If the output QDNRLM is set, the setpoint fall is limited.
QSP_HLM	71.2	BOOL	FALSE	The output QSP_HLM reports that the high setpoint limit SP_HLM has been reached.
QSP_LLM	71.3	BOOL	FALSE	The output QSP_LLM reports that the low setpoint limit SP_LLM has been reached.

Parameters	Address	Data type	Default	Description
QLMNUP	71.4	BOOL	FALSE	"Manipulated value signal up" output. (Only in the case of step controllers or pulse controllers)
QLMNDN	71.5	BOOL	FALSE	"Manipulated value signal down" output. (Only in the case of step controllers or pulse controllers)
QTUN_ON	71.6	BOOL	FALSE	QTUN_ON = TRUE indicates that optimization is in progress.
PAR_ACT	71.7	BOOL	FALSE	Updating controller parameters
SP	72.0	REAL	0.0	Effective actual setpoint.
ER	76.0	REAL	0.0	Negative deviation
DISV	80.0	REAL	0.0	Effective actual disturbance variable. Values from -100.0 to 100.0 (%) are permitted.
LMN_A	84.0	REAL	0.0	In the case of continuous or pulse controllers of manipulated value A of the split-range function and for step controllers with analog position feedback, the position feedback is displayed on output LMN_A. Values from -100.0 to 100.0 (%) are permitted.
LMN_B	88.0	REAL	0.0	On output LMN_B, in the case of continuous or pulse controllers, manipulated value B of the split-range function is displayed. Values from -100.0 to 100.0 (%) are permitted.
PHASE	92.0	INT	0	Indicates the controller optimization phases. *) 0..7
STATUS_H	94.0	INT	0	Diagnostics value for searching for point of inflection during heating optimization See parameter FMT_PID.STATUS_H (Page 6019)
STATUS_C	96.0	INT	0	Diagnostics value for searching for point of inflection during cooling optimization. See parameter FMT_PID.STATUS_C (Page 6020)
STATUS_D	98.0	INT	0	Diagnostics value for controller design during heating optimization See parameter FMT_PID.STATUS_D (Page 6022)
ZONE_TUN	100.0	WORD	W#16#0	In HEX code, each one of the four digits represents one channel, arranged in the following order from left to right: Channel 0, 1, 2, 3. ZONE_TUN = 0 means that the channel selected will not be optimized together with other channels. A value <> 0000 shows - in each case with a 1 - the channels which will be optimized as a group.
GAIN_P	152.0	REAL	0.0	Identified steady state gain (phys. variable) /%. In the case of process type I, GAIN_P tends to be estimated too low.
TU	156.0	REAL	0.0	Identified time lag of process. TU≥ 3*Sampling time
TA	160.0	REAL	0.0	Identified recovery time of process. In the case of process type I, TA tends to be estimated too low.
KIG	164.0	REAL	0.0	Maximum possible rate of rise of the process value. (phys. variable) /s GAIN_P = 0.01 * KIG * TA
N_PTN	168.0	REAL	0.0	Order of the process "Non-integer values" are also possible. Values from 0.1 to 10.0 are permitted.
TM_LAG_P	172.0	REAL	0.0	Time constant of a PTN model. (practical values for N_PTN>=2 only).

Parameters	Address	Data type	Default	Description
T_P_INF	176.0	REAL	0.0	Time from process excitation to inflection point.
P_INF	180.0	REAL	0.0	Change in process value from process excitation to inflection point.
LMN0	184.0	REAL	0.0	Manipulated value at the start of optimization Is established in phase 1 (average) Values from -100.0 to 100.0 (%) are permitted.
PV0	188.0	REAL	0.0	Process value at the start of optimization Is established in phase 1 (average).
PVDT0	192.0	REAL	0.0	Process value slew rate at the start of optimization Sign is adapted. (phys. variable) /s
PVDT	196.0	REAL	0.0	Current process value slew rate Sign is adapted. (phys. variable) /s
PVDT_MAX	200.0	REAL	0.0	Maximal process value modification (phys. variable) /s Maximum derivative of the process value at the point of inflection (sign adapted, always > 0); is used to calculate TU and KIG.
NOI_PVDT	204.0	REAL	0.0	Noise action in PVDT_MAX in % The larger the noise action, the less precise (softer) are the controller parameters.
NOISE_PV	208.0	REAL	0.0	Absolute process value noise action (phys. variable) /s The difference between maximum and minimum process value in phase 1.
FIL_CYC	212.0	INT	0	Number of cycles of the mean value filter The process value is determined through FIL_CYC cycles. FIL_CYC is increased from 1 to a max. of 1024 if needed.
POI_CMAX	214.0	INT	0	Maximum number of cycles after inflection point This time is used to find another (i.e. better) inflection point in case of measuring noise. Only then is optimization completed.
POI_CYCL	216.0	INT	0	Number of cycles after inflection point

See also

Parameters STATUS_H (Page 6019)

Parameters STATUS_C (Page 6020)

Parameters STATUS_D (Page 6022)

FMT_PV (S7-300, S7-400)

Description FMT_PV (S7-300, S7-400)

Instruction FMT_PV is for reading or writing process values (analog and digital input values) to support startup.

Call

FMT_PV has to be called in the same OB as all the other instructions that access the same FM 355-2.

FMT_PV requires no initialization routine.

Set parameter LOAD_PV cyclically to TRUE if you wish to write cyclically simulated process values to FM 355-2.

Set parameter READ_PV cyclically to TRUE to receive constantly the updated process values.

After the diagnostic values have been successfully written or read, FMT_PV resets the parameter LOAD_PV or READ_PV .

Simulation of the analog values (LOAD_PV = TRUE)

The simulation of the analog value for the channels 0 to 3 is enabled via the switches S_AION[i] or S_PVON[i], where 0 ≤ i ≤ 3. The figure below shows the point at which the simulated analog value becomes effective.

You specify the simulation values for the channels 0 to 3 via the parameter PV_SIM[i].

You can have the simulation values become effective at two points:

- S_AION[i] = TRUE (0 ≤ i ≤ 3)
  The value PV_SIM[i] is used instead of the value of analog input i of the module.
- S_PVON[i] = TRUE (0 ≤ i ≤ 3)
  The value PV_SIM[i] is used instead of the conditioned value of analog input i of the module.

Simulation of the digital values (LOAD_PV = TRUE)

The simulation of the values for the digital inputs 0 to 7 is enabled via the switches S_DION[i], where 0 ≤ i ≤ 7 is.

You lay down the simulation values via parameter DI_SIM[i].

- S_DION[i] = TRUE (0 ≤ i ≤ 7)
  Instead of the value of digital input i of the module, the value DI_SIM[i] is used.

Note

LEDs I0 to I7 always display the state of the associated digital input, also during simulation.

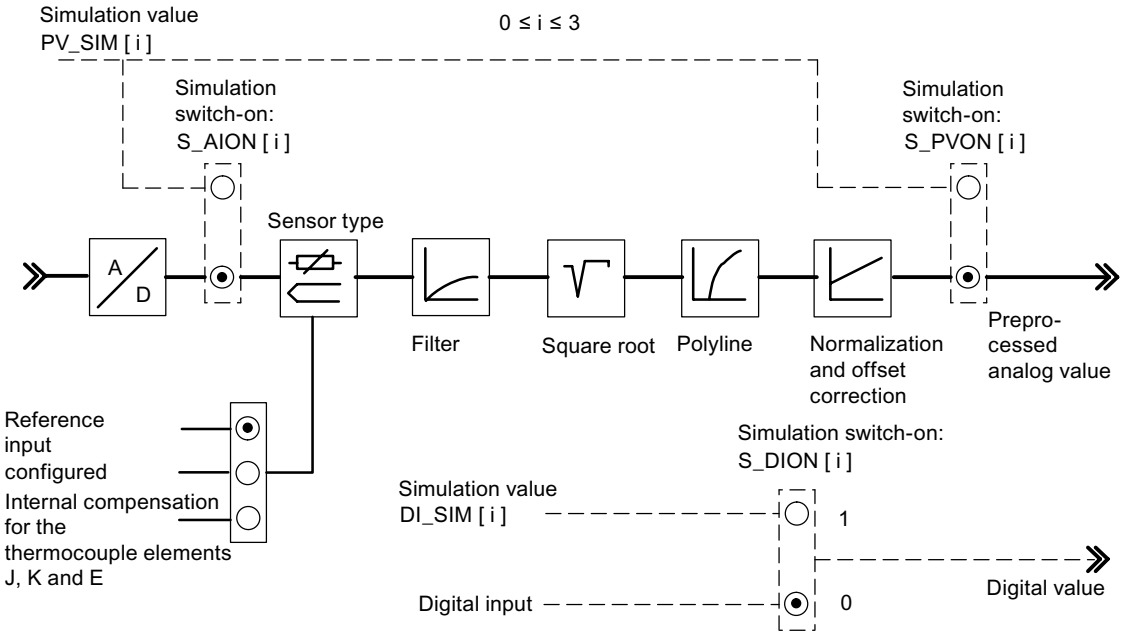

Note

Enabling and specification of the simulation values (Force) does not take place via the configuration software. This is why the relevant switches and connecting lines are drawn as dotted lines.

Displaying the process values (READ_PV = TRUE)

The following values are displayed:

- The actual state of digital inputs 0 to 7 are displayed at parameters STAT_DI[0] to STAT_DI[7], even if these are simulated.
- The value of analog inputs 0 to 3 is displayed on parameters DIAG[0].PV_PER to DIAG[3].PV_PER in mA or mV units. If the simulation of the analog input value was enabled, the simulated value is displayed.
- The conditioned analog input value 0 to 3 is displayed on parameters DIAG[0].PV_PHY to DIAG[3].PV_PHY in physical unit. If the simulation of the conditioned physical analog input value was enabled, the simulated value is displayed.

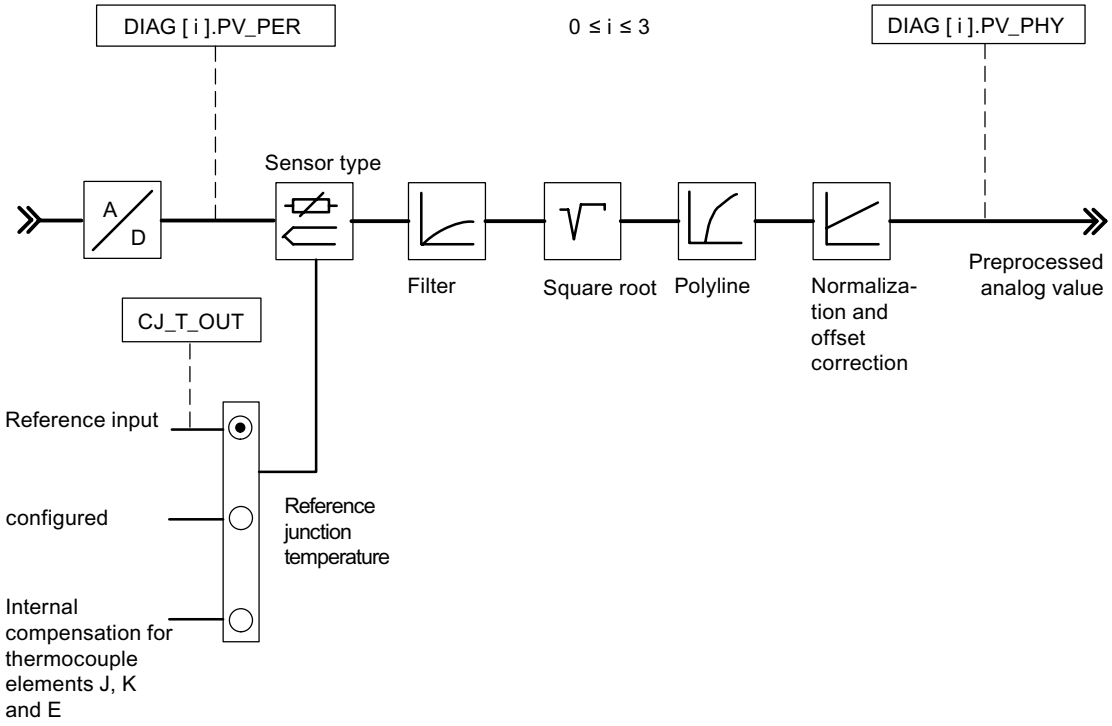

Note

The updating of the process values can be delayed if FM 355-2 is loaded compared to instruction FMT_PID.

Start-up

When FM 355-2 restarts after power off, the simulation switches on the FM 355-2 are again set to FALSE.

Input parameter FMT_PV (S7-300, S7-400)

Table 4-263

Parameters	Address	Data type	Default	Description
S_AION	0.0	ARRAY [0..3] of BOOL	FALSE	If, for example, the S_AION[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the analog input value 1 of the module.
S_PVON	2.0	ARRAY [0..3] of BOOL	FALSE	If, for example, the S_PVON[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the conditioned analog input value 1 of the module.
PV_SIM	4.0	ARRAY [0..3] of REAL	0.0	The simulation value for the analog input 1 is specified, for example, at input PV_SIM[1]. If S_PVON = TRUE, then the conditioned analog input value is specified here. If S_PVON = FALSE and S_AION = TRUE, then the analog input value, which is transformed into a conditioned value by means of the conditioning functions, is specified in mA or mV. 0.0 to 20.0 [mA] or -1500 to +10000 [mV] or technical value range
S_DION	20.0	ARRAY [0..7] of BOOL	FALSE	If, for example, the S_DION[1] switch is set to TRUE, the value DI_SIM[1] is used instead of the digital input value 1 of the module.
DI_SIM	22.0	ARRAY [0..7] of BOOL	FALSE	
MOD_ADDR	24.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

See also

Output parameter FMT_PV (Page 6039)

in/out parameters FMT_PV (Page 6039)

Output parameter FMT_PV (S7-300, S7-400)

Table 4-264

Parameters	Address	Data type	Default	Description
STAT_DI	26.0	ARRAY [0..7] of BOOL	FALSE	The states of digital inputs 0 to 7 are displayed at the STAT_DI parameters.
DIAG[x].PV_PER	28+ (Channel number x8)	ARRAY [0..3] of STRUCT	0.0	The analog input value of the module is displayed, for example, in the units mA or mV, at the DIAG[x].PV_PER parameter.
DIAG[x].PV_PHY	32+ (Channel number x8)	ARRAY [0..3] of STRUCT	0.0	The DIAG[x].PV_PHY parameter, for example, displays the conditioned analog input value of the module in the physical unit.
RET_VALU	60.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the instructions RDREC and WRREC.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

in/out parameters FMT_PV (S7-300, S7-400)

Table 4-265

Parameters	Address	Data type	Default	Description
LOAD_PV	62.0	BOOL	FALSE	When the in/out parameter LOAD_PV is set, the simulation values in PV_SIM and DI_SIM are written to FM 355-2 in accordance with the switch positions and the in/out parameter is reset.
READ_PV	62.1	BOOL	FALSE	If the parameter READ_PV = TRUE, the reference junction temperature is read from the module. Thereafter the in/out parameter is reset.

See also

Input parameter FMT_PV (Page 6038)

Output parameter FMT_PV (Page 6039)

FMT_CJ_T (S7-300, S7-400)

Description FMT_CJ_T (S7-300, S7-400)

Instruction FMT_CJ_T is for reading the measured reference junction temperature and for changing the parameterized reference junction temperature online. This is necessary if a temperature control system with several FM 355-2 with thermoelement inputs is to be operated without having to connect a Pt 100 to each FM 355-2.

Call

FMT_PAR has to be called in the same OB as all the other instructions that access the same FM 355-2.

If FM 355-2 is used in distributed I/O, it may take a few call cycles until the parameters are applied to FM 355-2. As long as the transfer has not been completed, parameter LOAD_CJ retains the value TRUE. So keep repeating your call of FMT_CJ_T when modifying parameters until the block sets LOAD_CJ = FALSE.

Mode of operation

If, for example, an FM 355-2 measures the reference junction temperature of an extruder control system with more than four heating zones, it can be read with READ_CJ = TRUE on parameter CJ_T_OUT and can be parameterized for the other FM 355-2 via CJ_TEMP and LOAD_CJ.

On parameter CJ_T_OUT the reference junction temperature measured at the reference point is expressed in degrees C or in degrees F (depending on which temperature unit has been parameterized).

On parameter CJ_T_OUT 0.0. is displayed if:

- no sensor of the "Thermoelement" type has been parameterized,
- the parameterized reference junction temperature was selected on all analog inputs,
- the internal compensation was selected on all analog inputs.

Input parameters FMT_CJ_T (S7-300, S7-400)

Table 4-266

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CJ_TEMP	2.0	REAL	0.0	The reference junction temperature can be specified at parameter CJ_TEMP.

Output parameters FMT_CJ_T (S7-300, S7-400)

Table 4-267

Parameters	Address	Data type	Default	Description
CJ_T_OUT	6.0	REAL	0.0	At the CJ_T_OUT output, the measured reference junction temperature is displayed by the module if a thermocouple element input is configured and the reference junction temperature is not configured.
RET_VALU	10.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the instructions RDREC and WRREC.

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

In/out parameters FMT_CJ_T (S7-300, S7-400)

Table 4-268

Parameters	Address	Data type	Default	Description
LOAD_CJ	12.0	BOOL	FALSE	If the in/out parameter LOAD_CJ has been set, the configured reference junction temperature on the module is overwritten with the value CJ_TEMP and the in/out parameter is reset.
READ_CJ	12.1	BOOL	FALSE	If the parameter READ_CJ = TRUE, the reference junction temperature is read from the module. Thereafter the in/out parameter is reset.

FM 455 PID Control (S7-300, S7-400)

PID_FM_455 (S7-300, S7-400)

Description PID_FM_455 (S7-300, S7-400)

During operation, the instruction PID_FM_455 is used to:

- Transfer operating parameters to FM 455
- Transfer controller parameters to FM 455
- Read process values from FM 455

Operating parameters are all in/out parameters that lie between the op_par and cont_par parameters in the instance data block of the instruction.

Controller parameters are all in/out parameters that are located after the cont_par parameter in the instance data block of the instruction.

Process values are all output parameters of the instruction after the out_par parameter.

The required data are stored in an instance DB on the CPU. You need one instance DB for each controller channel used. All the in/out parameters are set to FALSE after an instance DB has been created.

Call

PID_FM_455 is normally called in the cyclic interrupt OB 35 and requires an initialization routine that is started by setting the parameter COM_RST = TRUE in the start-up of the CPU. A calling of the instruction in the start-up OB is possible, but not necessary. After the initialization routine, PID_FM_455 sets the parameter COM_RST to FALSE.

PID_FM_455 has to be called in the same OB as all the other instructions that access the same FM 455.

If you call the instruction PID_FM_455 as a multiple instance DB, no commissioning interface is available. In this case you must commission the FM 455 via a watch table.

Start-up

During CPU start-up and during CPU STOP-RUN transition, the parameters on the FM 455 are overwritten with the parameters from the SDB of the CPU.

Responses in the event of an error

The output parameter RET_VALU includes the return value RET_VAL of the RD_REC und WR_REC instructions. RET_VALU can be evaluated if theREAD_VAR and LOAD_PAR parameters are not reset.

When the instruction PID_FM_455 is called, an I/O access error can occur if the FM 455 is not plugged or is not supplied with voltage. In this case the CPU goes into STOP, if no OB 122 is loaded in the CPU.

Mode of operation PID_FM_455 (S7-300, S7-400)

Effect of parameters LOAD_OP, READ_VAR and LOAD_PAR

In the initialization routine PID_FM_455 reads the parameters from the system data and stores them in its instance DB. Instance DB and system data are synchronized.

The following table shows how the function of the instruction is influenced by the parameters LOAD_OP, READ_VAR and LOAD_PAR.

Parameters	TRUE	FALSE
LOAD_OP	Using the instruction WR_REC to transfer the operating parameters from the CPU to FM 455	Using direct I/O access in multiplex mode to transfer the operating parameters from the CPU to FM 455
	Longer run time	Shorter run time
	The values are effective after a cycle of the CPU or FM. The longer cycle is decisive.	A maximum of 3 cycles of the CPU or FM are required until the values are effective. The longer cycle is decisive

Parameters	TRUE	FALSE
READ_VAR	Using the instruction RD_REC to read the process value from FM 455	Using direct I/O access in multiplex mode to read the process values from FM 455
	Longer processing time	Shorter processing time
	All parameters are read from FM 455	The following parameters are **not** read from FM 455 • SP (setpoint from the FM) • ER (error signal) • DISV (disturbance variable) • LMN_A • LMN_B
	Following parameters are updated at **every** call of the instruction. • PV (process value) • LMN (manipulated value) • The binary displays	Following parameters are updated at every **fourth** call of the instruction. • PV (process value) • LMN (manipulated value) • The binary displays
	At the start-up of the CPU, the following values are applied from the FM 455 to the PID_FM_455. • SP_OP • LMN_OP • SP_OP_ON • LMNOP_ON If these values were changed on the CPU, they are overwritten with the values from the FM 455 at the next start-up of the CPU	At the start-up of the CPU, the following values are read from the instruction, not from the FM: • SP_OP • LMN_OP • SP_OP_ON • LMNOP_ON
LOAD_PAR	All controller parameters are downloaded from the CPU to FM 455. This is practical if the controller parameters are to be changed depending on process states.	No transfer of the controller parameters

After data transfer as been completed the parameters LOAD_OP, READ_VAR and LOAD_PAR are reset to FALSE.

Special features during transfer in multiplex mode

Multiplexing of the data to be transferred during access to the FM 455 via direct I/O accesses is controlled via the PID_FM_455. This multiplex controlling via PID_FM_455 does not function if two instances of the instruction PID_FM_455 access the same channel number of a module. This results in incorrect parameters in the FM 455 (for example, setpoint and manual manipulated value) and incorrect displays at the output parameters of the instruction PID_FM_455.

Saving the Parameters in EEPROM

During program controlled re-configuring (LOAD_PAR, LOAD_OP) of the controller module by the instruction PID_FM_455, its run time is extended. The new parameters are effective immediately and are stored additionally in a nonvolatile memory (EEPROM). After each saving of the parameters in the EEPROM, a renewed saving is delayed by 30 minutes, since the life of the EEPROM is limited by the number of write processes. It is possible to save the new parameter in the EEPROM as soon as the power supply returns. The selection of the parameters determines whether the reconfiguration of the controller module by the instruction PID_FM_455 is executed without bumps.

Processing times of the PID_FM_455

Boundary conditions			Processing time in
READ_VAR	LOAD_OP	LOAD_PAR	CPU 414-2DP
FALSE	FALSE	FALSE	0.077 ms
TRUE	FALSE	FALSE	2.36 ms
TRUE*)	TRUE	FALSE	4.48 ms
FALSE	FALSE	TRUE	2.59 ms
TRUE	FALSE	TRUE	5.15 ms
TRUE*)	TRUE	TRUE	7.1 ms
*) If LOAD_OP = TRUE, READ_VAR is also set to TRUE by the PID_FM_455.			

Input parameters PID_FM_455 (S7-300, S7-400)

Table 4-269

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	1	The number of the controller channel to which the instance DB is referenced is configured at the "Channel number" input. Values of 1 to 16 are permitted.
PHASE	4.0	INT	0	The parameter PHASE can be interconnected with the output parameter PHASE of the instructions TUN_EC or TUN_ES (PID self-tuner). The phase state of the PID Self-Tuner can then be displayed in plain text in the "Commissioning" dialog. This parameter is of no significance for the OP.

Output parameters PID_FM_455 (S7-300, S7-400)

Parameters	Address	Data type	Default	Description
RET_VALU	6.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.
out_par	8.0	WORD	W#16#3130	The out_par parameter may not be modified by users. It marks the start of the output parameter that is read by the module if READ_VAR = TRUE is set.

Parameters	Address	Data type	Default	Description
SP	10.0	REAL	0.0	The setpoint that is currently in effect is available at the "Setpoint" output.
PV	14.0	REAL	0.0	The process value that becomes effective is issued on the "Process value" output.
ER	18.0	REAL	0.0	The control deviation that becomes effective is issued on the "Control deviation" output.
DISV	22.0	REAL	0.0	The interference that becomes effective is issued on the "Interference" output. Values from -100 to 100 % are permitted.
LMN	26.0	REAL	0.0	The manipulated value that becomes effective is issued on the "Manipulated value" output. Step controllers without analog position feedback output the unlimited P- + D-action at the LMN parameter. Values from -100 to 100 % are permitted.
LMN_A	30.0	REAL	0.0	At the output "Manipulated value A of the split range function / position feedback", continuous controllers display the manipulated value A and step controllers with analog position feedback display the position feedback. The output LMN_A can only be used for an approximate display of a corresponding simulated manipulated variable. The start value LMNRSVAL of the simulated position feedback has to be configured accordingly and becomes effective when LMNRS_ON is set. Values from -100 to 100 % are permitted.
LMN_B	34.0	REAL	0.0	Continuous controllers display the manipulated value B of the split-range function at the output “Manipulated value B of split-range function”. Values from -100 to 100 % are permitted.
QH_ALM	38.0	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_ALM limit is reported at the output "High limit alarm triggered".
QH_WRN	38.1	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_WRN limit is reported at the output "High limit warning triggered".
QL_WRN	38.2	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_WRN limit is reported at output "Low limit warning triggered".
QL_ALM	38.3	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_ALM limit is reported at output "Low limit alarm triggered".
QLMN_HLM	38.4	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the high limit is reported at output "High limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QLMN_LLM	38.5	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the low limit is reported at output "Low limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QPARA_F	38.6	BOOL	FALSE	The module checks the validity of the parameters. A parameter assignment error is displayed at the "Parameter assignment error" output. These parameter assignment errors can also be read out in the module's "Online & Diagnostics" dialog.
QCH_F	38.7	BOOL	FALSE	The "Channel error" output is set if the controller channel cannot provide valid results. A channel error (for example, wire break) is also set if QPARA_F = 1 or QMOD_F = 1. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.
QUPRLM	39.0	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of positive setpoint slew rate triggered" is set, the positive setpoint slew rate is limited.
QDNRLM	39.1	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of negative setpoint slew rate triggered" is set, the setpoint fall is limited.

Parameters	Address	Data type	Default	Description
QSP_HLM	39.2	BOOL	FALSE	The setpoint is always limited to a high and a low limit. The output "High limit of setpoint value triggered" indicates that the high limit has been exceeded.
QSP_LLM	39.3	BOOL	FALSE	The setpoint is always limited to a high and a low limit. Violation of the low limit is reported at output "Setpoint low limit triggered".
QLMNUP	39.4	BOOL	FALSE	This is the "Manipulated value signal up" output. (Only in the case of step controllers or pulse controllers)
QLMNDN	39.5	BOOL	FALSE	This is the "Manipulated value signal down" output. (Only in the case of step controllers or pulse controllers)
QID	39.6	BOOL	FALSE	QID = TRUE indicates that an identification is running (not that it is enabled). On completion of the identification, the identification result can be read out from the IDSTATUS parameter of the instruction CH_DIAG_455.
QSPOPON	40.0	BOOL	FALSE	The output "Setpoint operation on" shows if the setpoint is being operated by the configuration tool. If the bit is set, the value SP_OP is applied as setpoint.
QLMNSAFE	40.1	BOOL	FALSE	If the output "Safety mode" is set, the safety manipulated value is output as the manipulated value.
QLMNO-PON	40.2	BOOL	FALSE	The output “Manipulated value operation on” indicates whether the manipulated value is being operated via the configuration tool. If the bit is set, the value LMN_OP is applied as manipulated value.
QLMNTRK	40.3	BOOL	FALSE	The output "Follow-up mode" indicates if the manipulated value is being adjusted via an analog input.
QLMN_RE	40.4	BOOL	FALSE	The output "Manual = 1; Automatic = 0" indicates whether or not the manipulated value is set on the external manipulated value LMN_RE (manual = 1).
QLMNR_HS	40.5	BOOL	FALSE	The output "High endstop signal of position feedback" indicates if the control valve is at its high endstop. QLMNR_HS = TRUE means: The control valve is at high endstop. (For step controllers only)
QLMNR_LS	40.6	BOOL	FALSE	The output "Low endstop signal of position feedback" indicates whether the control valve is at its low endstop. QLMNR_LS = TRUE means: The control valve is at its low endstop. (For step controllers only)
QLMNR_ON	40.7	BOOL	FALSE	Output "Position feedback enabled" indicates the set operating mode "Step controller with position feedback" or "Step controller without position feedback".
QFUZZY	41.0	BOOL	FALSE	If the output parameter is QFUZZY = 1, the controller operates with the fuzzy algorithm.
QSPLEPV	41.1	BOOL	FALSE	The output "Display of FUZZY controller: Setpoint < process value" is set when the fuzzy controller is switched on, if the setpoint is less than the effective process value.
QSPR	41.2	BOOL	FALSE	If the output "Split-range operation" is set, the continuous controller is operating in split-range mode.
QMAN_FC	41.4	BOOL	FALSE	The output "QMAN_FC" is set in the following two cases: • The slave controller is in manual mode and the master controller is followed up to the process value of the slave controller. • The I-action of the master controller is stopped because the setpoint or manipulated variable of the slave controller is limited or because the slave controller is in manual mode.

Parameters	Address	Data type	Default	Description
QPARABUB	41.7	BOOL	FALSE	This parameter is set by the FM if the operating parameters are changed via the OP. If READ_VAR = TRUE and if this display is set by the FM, the instruction PID_FM_455 reads out the parameters SP_OP_ON, LMNOP_ON, SP_OP and LMN_OP from the FM and saves them in the instance DB. The instruction thus applies over the operating state of the FM. After the reading process the parameter is set to FALSE.
QMOD_F	42.0	BOOL	FALSE	The instruction checks that the reading and writing of a data record has been completed. Output "Module error" is set if errors are found. The error cause can be: An incorrect module address at parameter MOD_ADDR, or an incorrect channel number at parameter CHANNEL, or a faulty module.

See also

RD_REC: Read data record from I/O (Page 5256)

WR_REC: Write data record to I/O (Page 5260)

In/out parameters PID_FM_455 (S7-300, S7-400)

Table 4-270

Parameters	Address	Data type	Default	Description
[1] Default values of the module after the first start-up of the PID_FM_455 with COM_RST = TRUE				
COM_RST	44.0	BOOL	FALSE	If the parameter COM_RST = TRUE, the instruction PID_FM_455 executes an initialization routine. In doing so the controller parameters (these are all the parameters after cont_par) are read from the FM and stored in the instance DB. In addition, the parameters MOD_ADDR and CHANNEL are checked for validity. After the initialization routine the parameter is set to FALSE.
LOAD_OP	44.1	BOOL	FALSE	If the in/out parameter "Load operating parameters to FM 455" is set, the operating parameters are loaded into the module and the in/out parameter is reset.
READ_VAR	44.2	BOOL	FALSE	If the in/out parameter "Read tags from FM 455" is set, the output parameters are read from the module and the in/out parameter is reset.
LOAD_PAR	44.3	BOOL	FALSE	If the in/out parameter "Load control parameters to FM 455" is set, the control parameters are loaded into the module and the in/out parameter is reset.
op_par	46.0	WORD	W#16#3130 [1]	The op_par parameter may not be modified by users. It identifies the start of the operating parameters that are transferred to the module, if LOAD_OP = TRUE is set. The end of the operating parameters is identified by cont_par.
SP_RE	48.0	REAL	0.0	An external setpoint is interconnected with the controller at the "External setpoint" input.
LMN_RE	52.0	REAL	0.0	An external manipulated value is interconnected to the controller at the input "External manipulated value". Values from -100 to 100 % are permitted.
SP_OP_ON	56.0	BOOL	FALSE	The instructions of PID Self-Tuner have access to the in/out parameter "Enable setpoint operation". If the bit is set, the value SP_OP is applied as setpoint.

Parameters	Address	Data type	Default	Description
SAFE_ON	56.1	BOOL	FALSE	If the "Adopt safety position" input is set, a safety value is applied as manipulated value. Note: The actuating signal processing via LMNDN_OP, LMNUP_OP and LMNSOPON on a step controller is of higher priority than the safety manipulated value.
LMNOP_ON	56.2	BOOL	FALSE	The instructions of PID Self-Tuner have access to the in/out parameter "Enable manipulated value operation". If the bit is set, the value LMN_OP is applied as manipulated value.
LMNTRKON	56.3	BOOL	FALSE	If input "Follow-up (LMN via AI)" is set, the manipulated value is adjusted to an analog input (AI). (Does not apply to step controllers without analog position feedback)
LMN_REON	56.4	BOOL	FALSE	If input "Enable external manipulated value" is set, the external manipulated value LMN_RE is set as the manipulated value.
LMNRHSRE	56.5	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected at the "High endstop signal of position feedback" input. LMNRHSRE = TRUE means: The control valve is located at the high endstop. (Only in the case of step controllers)
LMNRLSRE	56.6	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected at the "Low endstop signal of position feedback" input. LMNRLSRE = TRUE means: The control valve is located at the low endstop. (Only in the case of step controllers)
LMNSO-PON	56.7	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Enable manipulated value operation". If LMNSOPON is set, the signals LMNUP_OP and LMNDN_OP are applied as manipulated value signals. (Only in the case of step controllers)
LMNUP_OP	57.0	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Manipulated value signal up". If LMNSOPON is set, the value at input "Manipulated value signal up operation" is set as the manipulated value signal. (Only in the case of step controllers)
LMNDN_OP	57.1	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Manipulated value signal down operation". If LMNSOPON is set, the value at input "Manipulated value signal down operation" is set as the manipulated value signal. (Only in the case of step controllers)
LMNRS_ON	57.3	BOOL	FALSE	You can simulate no position feedback if this is not available. The function is enabled by setting the "Enable simulation of position feedback" input The instructions of PID Self-Tuner have access to this parameter as well since at least one simulated manipulated value is required for optimization if a step controller was configured without position feedback. The simulated value is displayed at parameter LMN_A. When the simulation is enabled the value of the parameter LMNRSVAL is set as the starting value. CAUTION: Offset between the simulation value and real position feedback increases over the time. (Only for step controllers without analog position feedback)
FUZID_ON	57.4	BOOL	FALSE	The identification of the fuzzy algorithm is enabled at the "Enable fuzzy identification" input.
SP_OP	58.0	REAL	0.0	The instructions of PID Self-Tuner have access to the in/out parameter "Setpoint operation". If the SP_OP_ON bit is set, the "Setpoint operation" value is applied as the setpoint.

Parameters	Address	Data type	Default	Description
LMN_OP[)]	62.0	REAL	0.0	The instructions of PID Self-Tuner have access to the in/out parameter "Manipulated value operation". If the LMNOP_ON bit is set, the "Manipulated value operation" value is applied as the manipulated value. Values from -100 to 100 % are permitted.
LMNRSVAL	66.0	REAL	0.0	The instructions of PID Self-Tuner have access to input "Starting value of simulated position feedback". The start value of the simulation is entered at the parameter. (Only for step controllers without analog position feedback) Values from -100 to 100 % are permitted.
cont_par	70.0	WORD	W#16#3130 [1)]	The cont_par parameter may not be modified by users. It indicates the start of the controller parameters which are read from the FM and stored in the instance DB, if COM_RST = TRUE is, and which are transferred to the FM when LOAD_PAR = TRUE is set. The end of the controller parameter is the end of the instance DB.
P_SEL	72.0	BOOL	TRUE [1)]	In the PID algorithm, the PID parts can be switched on and off individually. The proportional action is enabled when the “P-action on” input is set.
PFDB_SEL	72.1	BOOL	FALSE [1)]	In the PID algorithm, the P- and D-actions can be included in the feedback loop. The proportional action is located in the feedback loop when the input "Enable P-action in feedback loop" is set.
MONER-SEL	72.2	BOOL	FALSE [1)]	The controller has a limit indicator that can be applied either for the process value or the control deviation. If the input "Monitoring: Process value = 0, control deviation = 1" is set, the control deviation will be monitored.
SDB_SEL	72.4	BOOL	FALSE	SDB_SEL=TRUE: After CPU STOP-RUN, the controller parameters of instruction PID_FM_455 are not overwritten by the SDB parameters in the FMx55.
D_EL_SEL	74.0	INT	0 [1)]	The D-action element in the PID algorithm can be positioned at a separate input. This is selected via the input “D-action element input for the controller”. • 0: Control deviation • 1 to 16: Analog input 1 to 16 • 17: Negative process value, D-action in the feedback
SP_HLM	76.0	REAL	100.0 [1)]	The setpoint is always limited to a high and a low limit. The "Setpoint high limit" input specifies the high limit. SP_HLM > SP_LLM
SP_LLM	80.0	REAL	0.0 [1)]	The setpoint is always limited to a high and a low limit. The "Setpoint low limit" input specifies the low limit. SP_LLM < SP_HLM
H_ALM	84.0	REAL	100.0 [1)]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "High limit interrupt" input specifies the highest limit. H_ALM > H_WRN
H_WRN	88.0	REAL	90.0 [1)]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "High limit warning" input specifies the second highest limit. Values from H_ALM to L_WRN are permitted.
L_WRN	92.0	REAL	10.0 [1)]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "Low limit warning" input specifies the second lowest limit. Values from H_WRN to L_ALM are permitted.
L_ALM	96.0	REAL	0.0 [1)]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "Low limit alarm" input specifies the lowest limit. L_ALM < L_WRN

Parameters	Address	Data type	Default	Description
HYS	100.0	REAL	1.0[1)]	To prevent flickering of the monitoring displays, a hysteresis can be configured at the "Hysteresis" input. HYS ≥ 0.0
DEADB_W	104.0	REAL	0.0[1)]	A deadband is applied to the control deviation. The "Deadband width" input determines the size of the deadband. DEADB_W ≥ 0.0
GAIN	108.0	REAL	1.0[1)]	The "Proportional gain" input specifies controller amplification.
TI	112.0	REAL	3000.0[1)]	The "Integration time" input determines the time response of the integral action. If TI = 0, the integral action is disabled. TI = 0.0 or TI ≥ 0.5
TD	116.0	REAL	0.0[1)]	The "Derivative-action time" input determines the time response of the differentiator. If TD = 0, the derivative action is disabled. TD = 0.0 or TD ≥ 1.0
TM_LAG	120.0	REAL	5.0[1)]	The algorithm of the derivative action contains a delay for which parameters can be assigned at the input "Time lag of the derivative action". TM_LAG ≥ 0.5
LMN_SAFE	124.0	REAL	0.0[1)]	For the manipulated value, a safety value can be configured at the "Safety manipulated value" input. Values from -100 to 100 % are permitted.
LMN_HLM	128.0	REAL	100.0[1)]	The manipulated value is always limited to a high and a low limit. The "Manipulated value high limit" input specifies the high limit. (Does not apply to step controllers without analog position feedback) Values from LMN_LLM to 100 % are permitted.
LMN_LLM	132.0	REAL	0.0[1)]	The manipulated value is always limited to a high and a low limit. The "Manipulated value low limit" input specifies the low limit. (Does not apply to step controllers without analog position feedback) Values from -100 to LMN_HLM are permitted.
MTR_TM	136.0	REAL	60.0[1)]	The runtime from endstop to endstop of the control valve is entered at the "Motor actuating time" parameter. (Only in the case of step controllers)
PULSE_TM	140.0	REAL	0.2[1)]	A minimum pulse time can be configured at the "Minimum pulse time" parameter. (Applies to step controllers or pulse controllers only)
BREAK_TM	144.0	REAL	0.2[1)]	A minimum pulse break duration can be assigned with the parameter "Minimum break time." (Applies to step controllers or pulse controllers only)

Note

If LOAD_PAR = TRUE is set, all the control parameters are loaded permanently to the EEPROM of FM 455.

If LOAD_OP = TRUE is set, only the setpoint SP_RE of the operating parameters is loaded permanently to the EEPROM of the FM 455. All the other operating parameters are assigned the values 0 or FALSE as default during the startup of FM 455

The EEPROM of the module could become damaged by too frequent write operations. To prevent this, the module delays another write process by 30 minutes after writing of the EEPROM.

Block diagram PID_FM_455 (S7-300, S7-400)

Overview

The following figures show at which points the parameters of the instruction PID_FM_455 act.

The parameters are effective with three-component controllers, ration/blending controllers, and fixed setpoint or cascade controllers. This also applies for the parameters that exist equally at continuous-action controllers, at controllers with a pulse output as well as at step controllers. Therefore, not all block diagrams are depicted and not all parameter are marked in all figures.

However, the parameters of the instruction PID_FM_455 are contained in all the figures – with the exception of the parameters MOD_ADDR, CHANNEL, QMOD_F, QPARA_F, QCH_F, QLMNR_ON, RET_VALU, COM_RST, LOAD_PAR, READ_VAR, LOAD_OP.

Action of the input parameters

Control deviation generation with fixed-value or cascade controller

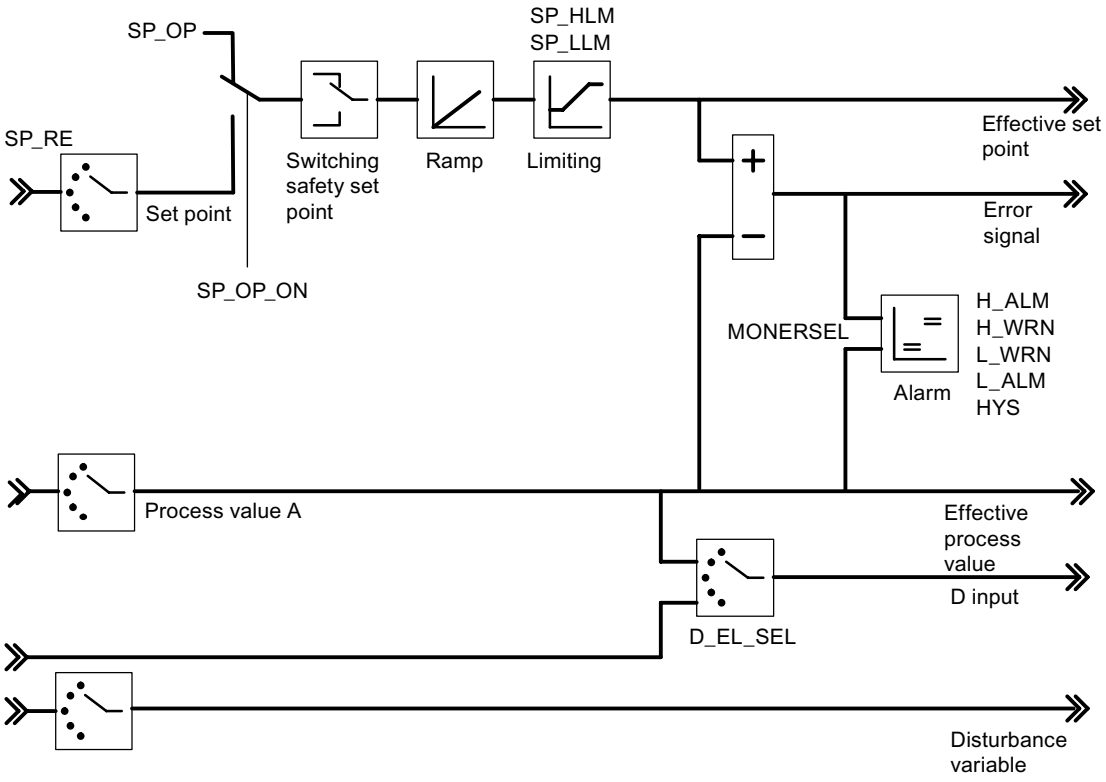

Block diagram of the control algorithm

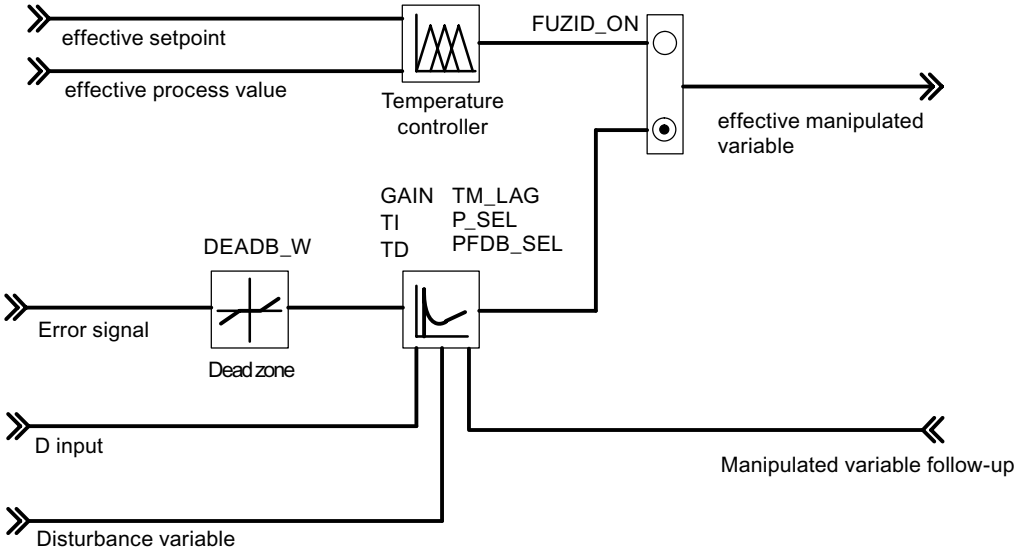

Controller output of the continuous-action controller

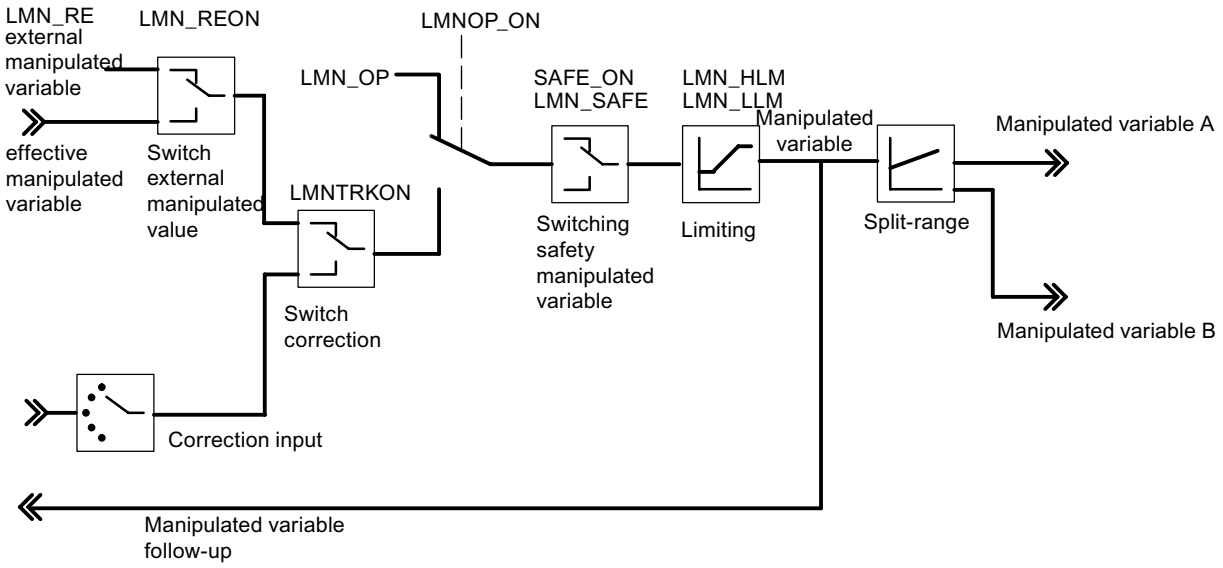

Controller output of the step controller (pulse controller operating mode)

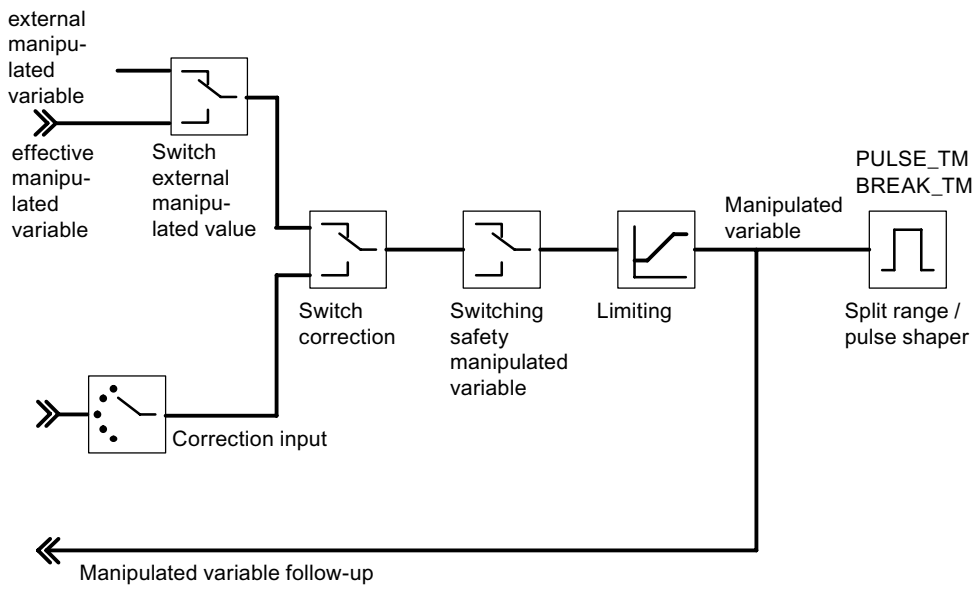

Controller output of the step controller (step controller operating mode with position feedback)

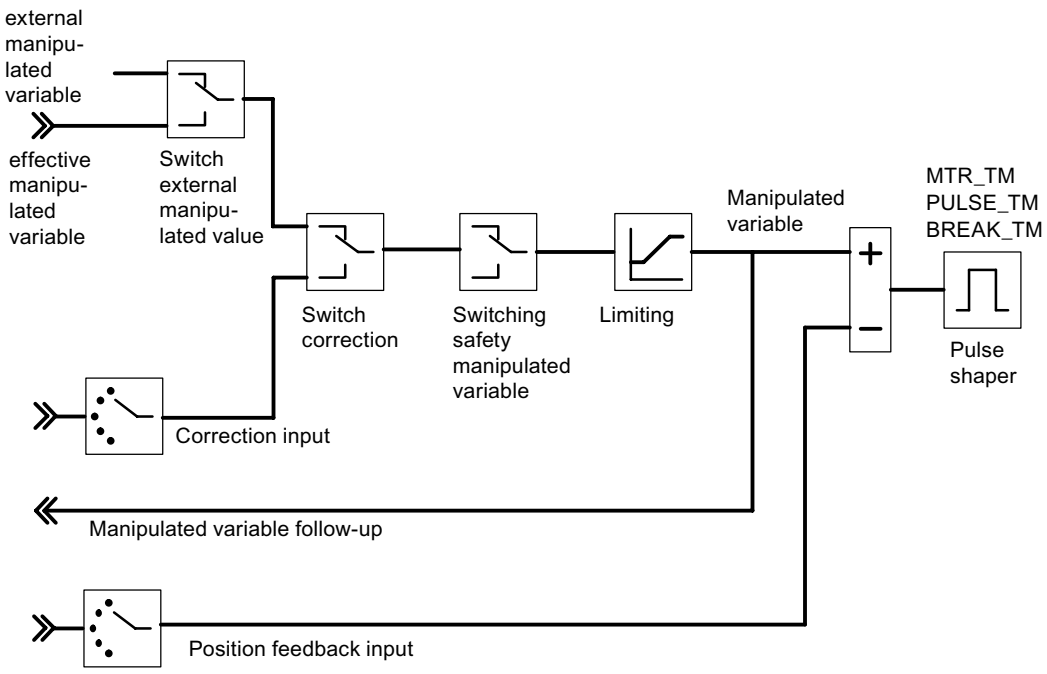

Controller output of the step controller (step controller operating mode without position feedback)

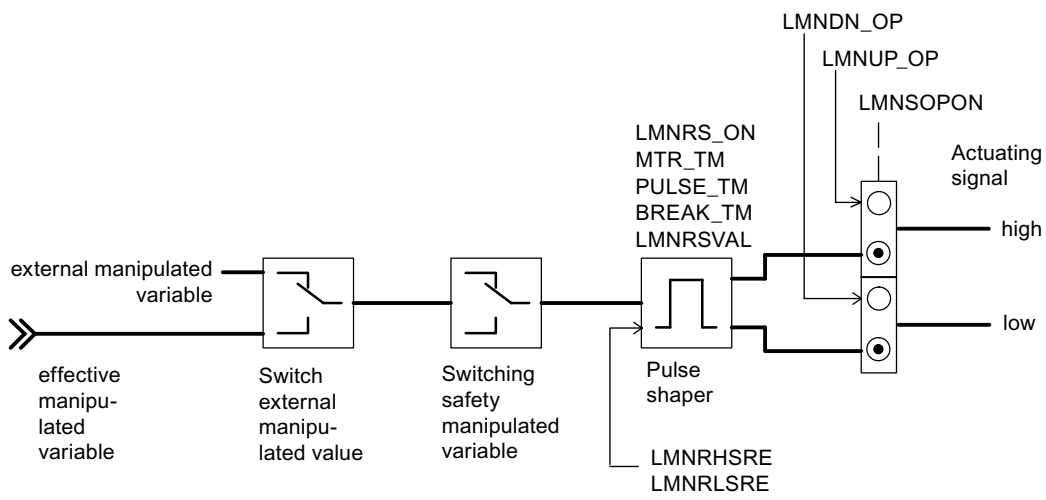

Generating output parameters

The following figures show at which points in the module the output parameters of the instruction PID_FM_455are generated.

Control deviation generation with fixed-value or cascade controller

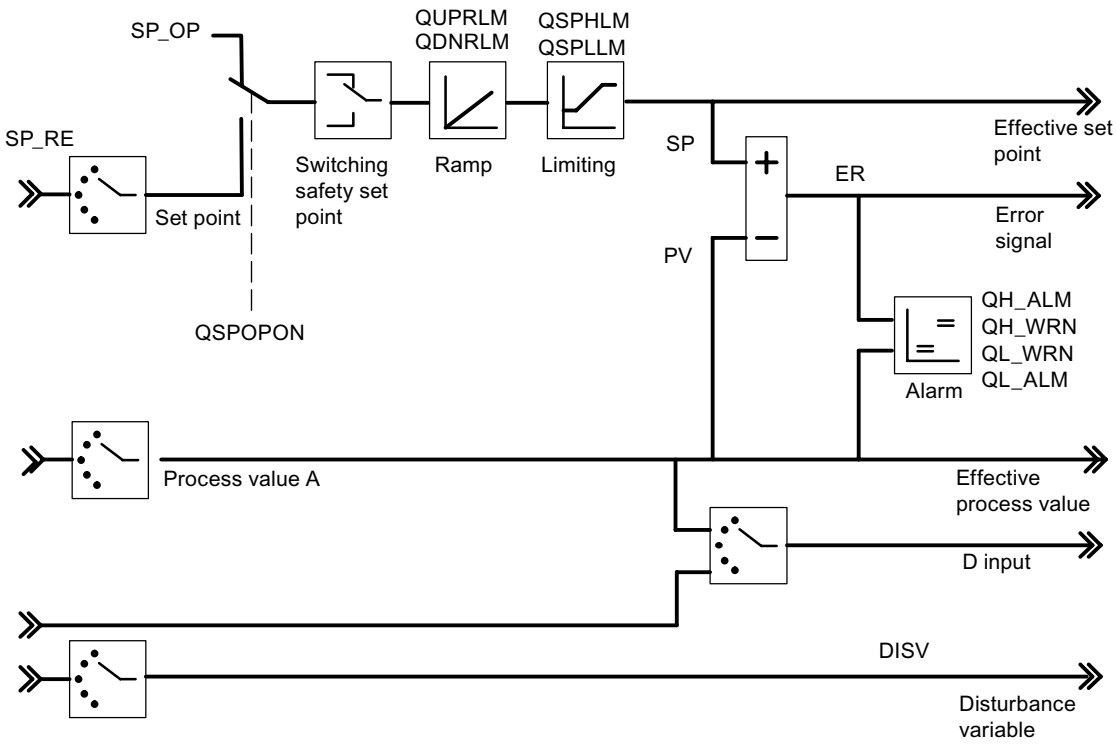

Block diagram of the control algorithm

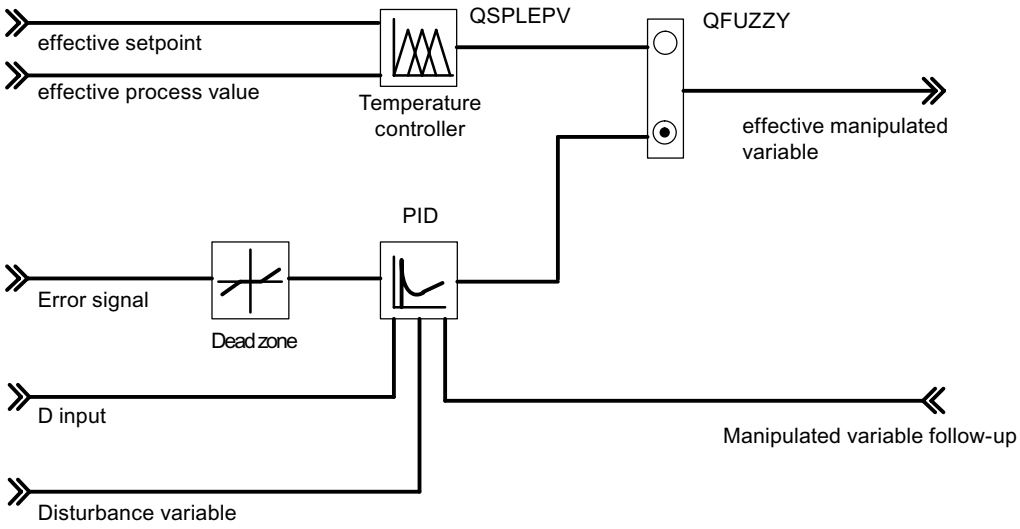

Controller output of the continuous-action controller

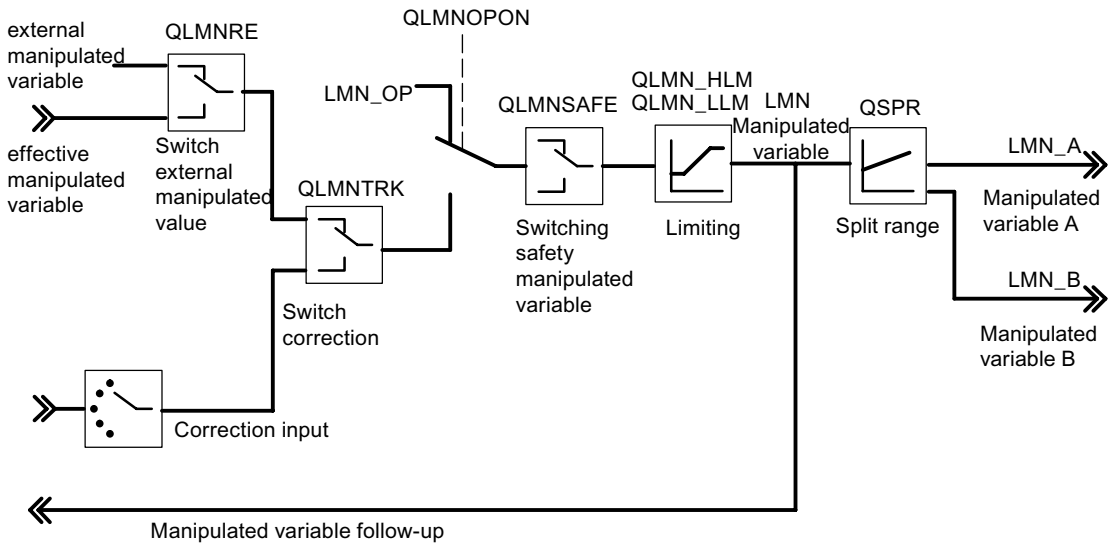

Controller output of the step controller (pulse controller operating mode)

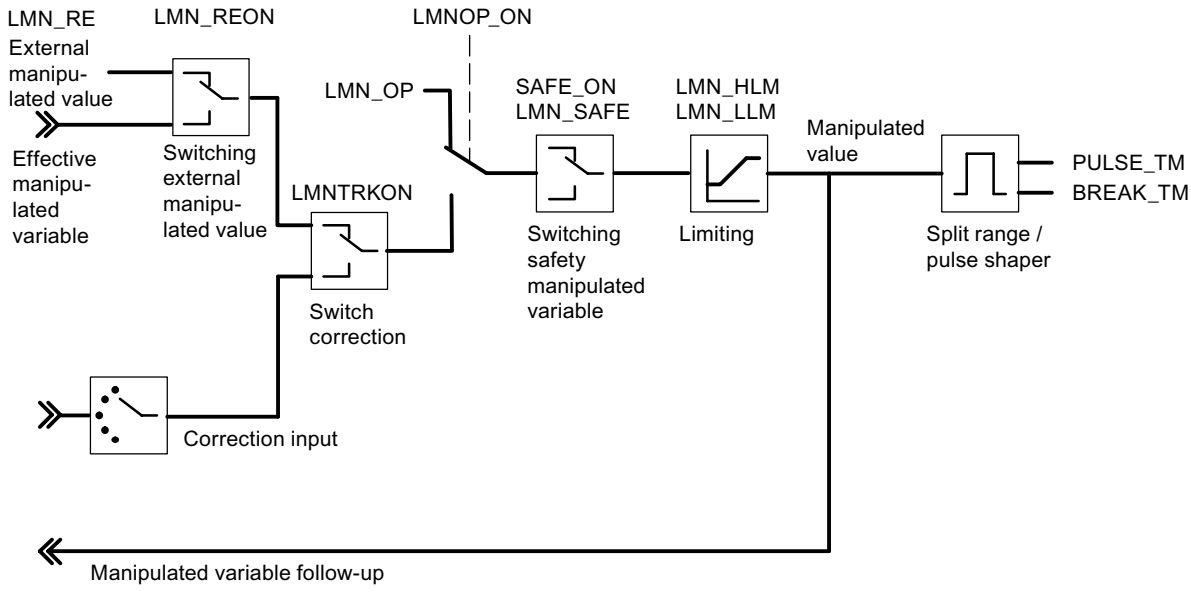

Controller output of the step controller (step controller operating mode with position feedback)

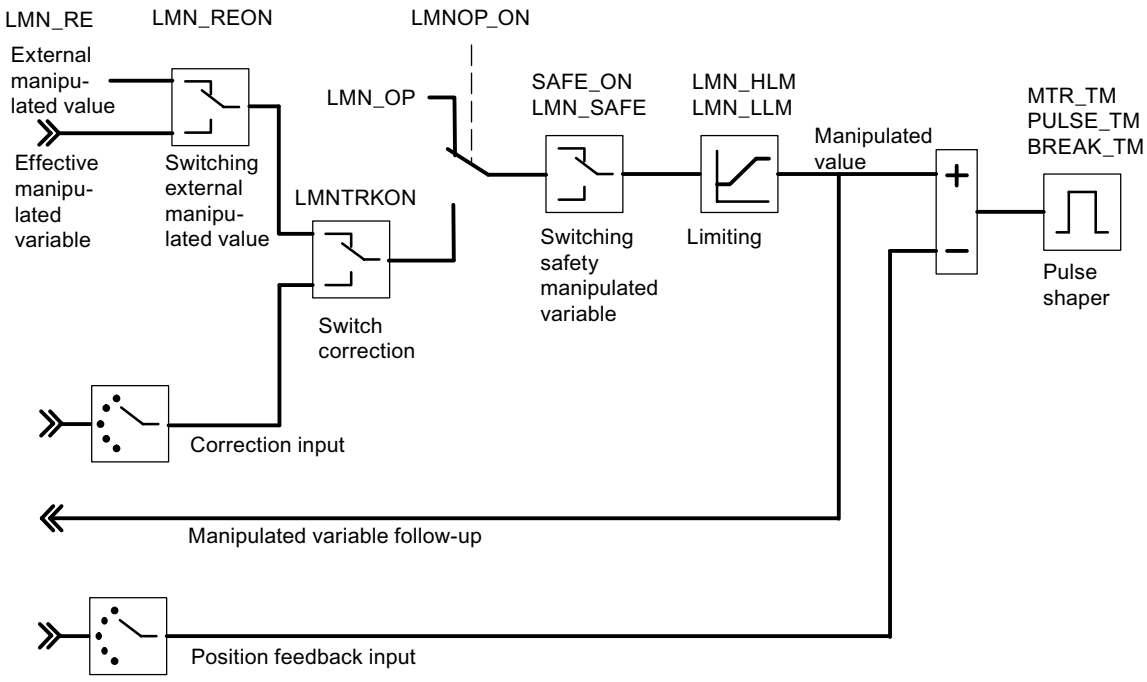

Controller output of the step controller (step controller operating mode without position feedback)

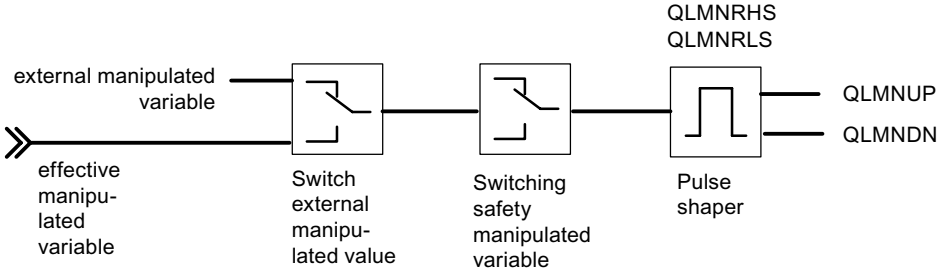

FUZ_455 (S7-300, S7-400)

Description FUZ_455 (S7-300, S7-400)

The instruction FUZ_455 is used to read out and write the controller parameters of the fuzzy temperature controller from FM 455.

This function is suitable for the following applications:

- Transferring the controller parameters of the FM 455 that have been established by identification after the module replacement
- Adapting the FM 455 to different processes

Note

It is not permitted to modify the parameters that have been established by the FM 455 by means of an identification as they have been optimized for the process.

Mode of operation

The parameters READ_PAR and LOAD_PAR are used to control the mode of operation of the instruction FUZ_455

If READ_PAR = TRUE, FUZ_455 reads the parameters of all the temperature controllers of the FM 455 and stores them in the instance DB. After the temperature controller parameters have been successfully read out, FUZ_455 sets the parameter READ_PAR to FALSE.

If LOAD_PAR = TRUE, FUZ_455 writes the parameters of all the temperature controllers of the FM 455 from the instance DB to the FM 455. After the parameters have been successfully transferred, FUZ_455 sets the parameter LOAD_PAR to FALSE.

Call

FUZ_455 requires no initialization routine.

FUZ_455 has to be called in the same OB as all the other instructions that access the same FM 455.

Start-up

During the start-up of the CPU you should set the LOAD_PAR parameter of the FUZ_455 instruction and then call the block in the cyclic program, provided LOAD_PAR = TRUE.

Input parameters FUZ_455 (S7-300, S7-400)

Table 4-271

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	512	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters FUZ_455 (S7-300, S7-400)

Table 4-272

Parameters	Address	Data type	Default	Description
RET_VALU	2.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.
PARAFFUZ	4.0	WORD	0	A parameter assignment error created by the the instruction FUZ_455 is displayed at the parameter PARAFFUZ as follows: High byte of PARAFFUZ = 01: There is a parameter assignment error.. High byte of PARAFFUZ = 00: There is no parameter assignment error. The low byte contains the offset of the parameter that caused the parameter assignment error, calculated from the static variable FUZ_PAR[1]. For instance, PARAFFUZ = W#16#0104 means that the second parameter is faulty.

See also

RD_REC: Read data record from I/O (Page 5256)

WR_REC: Write data record to I/O (Page 5260)

In/out parameters FUZ_455 (S7-300, S7-400)

Table 4-273

Parameters	Address	Data type	Default	Description
READ_PAR	6.0	BOOL	FALSE	If the READ_PAR parameter is set, the fuzzy parameters are read out from the module and stored in the static variables of the instance DB.
LOAD_PAR	6.1	BOOL	FALSE	If the LOAD_PAR parameter is set, the fuzzy parameters are read out from the static variables of the instance DB module and transferred to the module.

FORCE455 (S7-300, S7-400)

Description FORCE455 (S7-300, S7-400)

The instruction FORCE455 is used to simulate (force) the analog and digital input values to support the start-up.

Simulating Digital Values

The simulation of the values for the digital inputs 1 to 16 is enabled via the switches S_DION[i], where 1 ≤ i ≤ 16. You specify the simulation values at the parameters DI_SIM[i]. If S_DION[i] = TRUE, the value DI_SIM[i] is used Instead of the value of digital input i of the module,. The LEDs I1 to I16 always show the status of the associated digital input, even in the case of a simulation.

Simulating Analog Values

The simulation of the analog value for the channels 1 to 16 is enabled via the switches S_AION[i] or S_PVON[i], where 1 ≤ i ≤ 16.

You specify the simulation values for the channels 1 to 16 via the parameter PV_SIM[i].

You can have the simulation values become effective at two points:

- S_AION[i] = TRUE (1 ≤ i ≤ 16)
  The value PV_SIM[i] is used instead of the value of analog input i of the module.
- S_PVON[i] = TRUE (1 ≤ i ≤ 16)
  The value PV_SIM[i] is used instead of the conditioned value of analog input i of the module.

The following figure shows at which point the simulated analog value is effective.

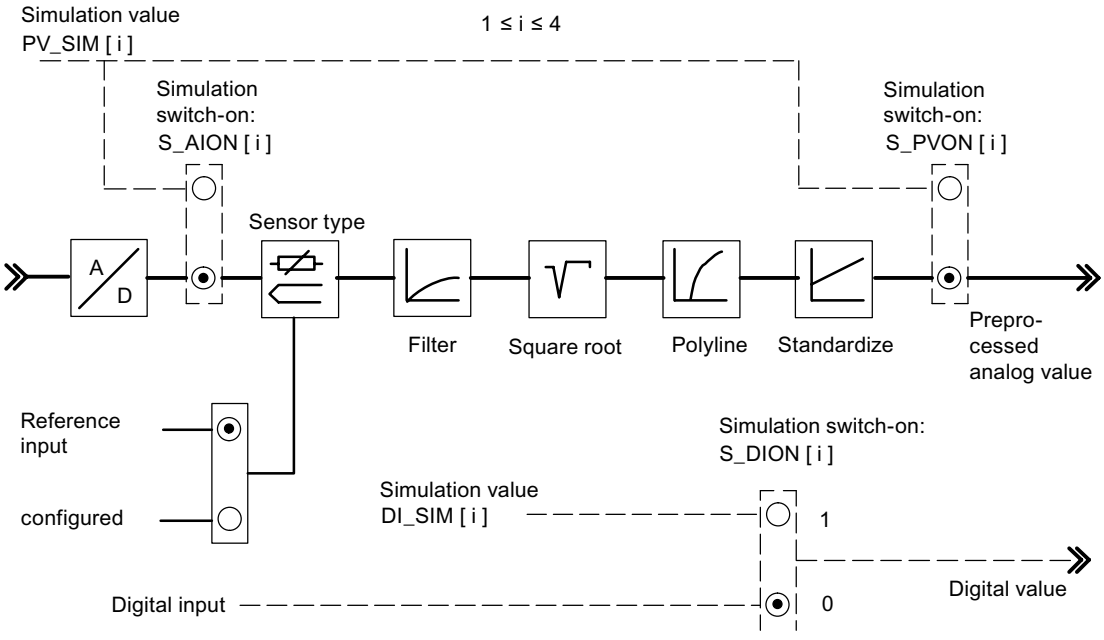

Enabling and specification of the simulation values (forcing) is not carried out via the parameter assignment interface. This is why the relevant switches and connecting lines are drawn as dotted lines.

Call

FORCE455 does not require any initialization routine and is normally called cyclically.

FORCE455 has to be called in the same OB as all the other instructions that access the same FM 455.

Startup characteristics

When FM 455 restarts after power off, the simulation switches on the FM 455 are again set to FALSE.

Input parameters FORCE455 (S7-300, S7-400)

Table 4-274

Parameters	Address	Data type	Default	Description
S_AION	0.0	ARRAY [1..16] of BOOL	FALSE	If, for example, the S_AION[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the analog input value 1 of the module.
S_PVON	2.0	ARRAY [1..16] of BOOL	FALSE	If, for example, the S_PVON[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the conditioned analog input value 1 of the module.
PV_SIM	4.0	ARRAY [1..16] of REAL	0.0	The simulation value for the analog input 1 is specified, for example, at input PV_SIM[1]. If S_PVON = TRUE, then the conditioned analog input value is specified here. If S_PVON = FALSE and S_AION = TRUE, then the analog input value, which is transformed into a conditioned value by means of the conditioning functions, is specified in mA or mV. Permitted are 0.0 to 20.0 [mA], -1500 to +10000 [mV], or a technical range of values.
S_DION	68.0	ARRAY [1..16] of BOOL	FALSE	If, for example, the S_DION[1] switch is set to TRUE, the value DI_SIM[1] is used instead of the digital input value 1 of the module.
DI_SIM	70.0	ARRAY [1..16] of BOOL	FALSE	Simulation value for digital input
MOD_ADDR	72.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters FORCE455 (S7-300, S7-400)

Table 4-275

Parameters	Address	Data type	Default	Description
RET_VALU	74.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.

See also

RD_REC: Read data record from I/O (Page 5256)

WR_REC: Write data record to I/O (Page 5260)

READ_455 (S7-300, S7-400)

Description Ready_455 (S7-300, S7-400)

The instruction READ_455 is used to read out the digital and analog input values to support the start-up.

The following values are displayed:

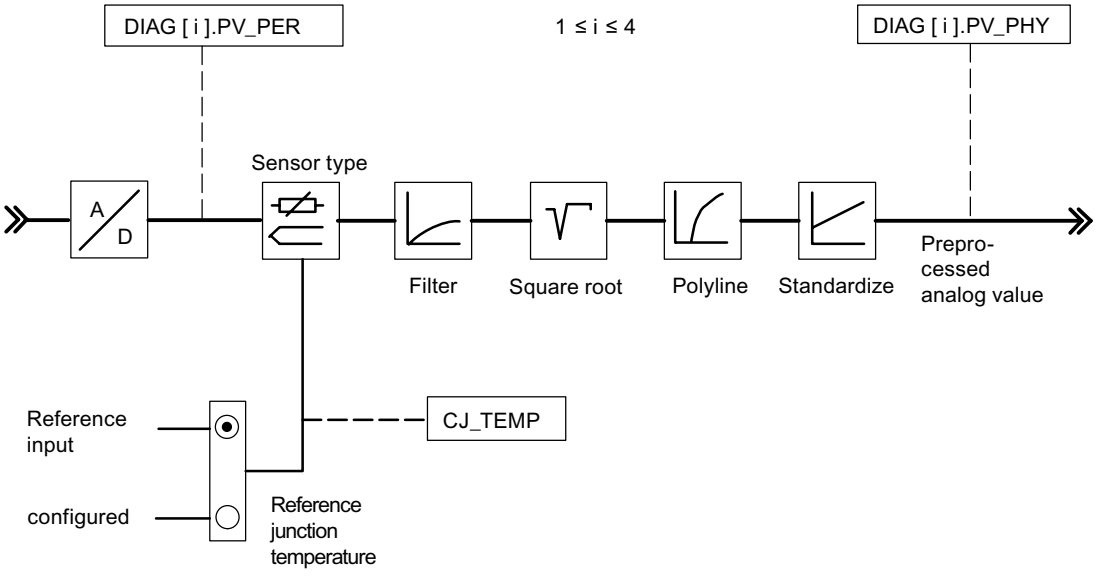

Call

READ_455 does not require any initialization routine and is normally called cyclically.

READ_455 has to be called in the same OB as all the other instructions that access the same FM 455.

Input parameters READ_455 (S7-300, S7-400)

Table 4-276

Parameters	Address	Data type	Default setting	Description
MOD_ADDR	0.0	INT	512	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters READ_455 (S7-300, S7-400)

Table 4-277

Parameters	Address	Data type	Default	Description
CJ_TEMP	2.0	REAL	0.0	At the CJ_TEMP output, the measured reference junction temperature is displayed by the module if a thermocouple element input is configured and the reference junction temperature is not configured.
STAT_DI	6.0	ARRAY [1..16] of BOOL	FALSE	The states of digital inputs 1 to 16 are displayed at the STAT_DI parameters.
DIAG[x].PV_PER	(channel number) x 8	ARRAY [1..16] of STRUCT	0.0	The analog input value of the module is displayed, for example, in the units mA or mV, at the DIAG[1].PV_PER parameter.
DIAG[x].PV_PHY	(channel number) x 8 + 4	ARRAY [1..16] of STRUCT	0.0	The DIAG[1].PV_PHY parameter, for example, displays the conditioned analog input value of the module in the physical unit.
RET_VALU	136.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.

See also

RD_REC: Read data record from I/O (Page 5256)

WR_REC: Write data record to I/O (Page 5260)

CH_DIAG_455 (S7-300, S7-400)

Description CH_DIAG_455 (S7-300, S7-400)

The instruction CH_DIAG_455 is used to read out additional channel-specific diagnostic variables from the module.

Call

CH_DIAG_455 does not require any initialization routine and is normally called cyclically.

CH_DIAG_455 has to be called in the same OB as all the other instructions that access the same FM 455.

Input parameters CH_DIAG_455 (S7-300, S7-400)

Table 4-278

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	1	The number of the controller channel to which the instance DB is referenced is configured at the "Channel number" input. Values of 1 to 16 are permitted.

Output parameters CH_DIAG_455 (S7-300, S7-400)

Table 4-279

Parameters	Address	Data type	Default	Description
SP_R	4.0	REAL	0.0	If a ratio controller is set, the input value of the setpoint value is assigned to the parameter.
PV_R	8.0	REAL	0.0	The following value is only assigned to the parameter value if the ratio controller is set: (Process value A - Setpoint value offset) / Process value D
DIF_I	12.0	REAL	0.0	The input variable of the D-action is displayed at the DIF_I parameter. This is particularly of interest if, for example, an analog input is configured as the input variable of the D-action component.
TRACKPER	16.0	REAL	0.0	The TRACKPER parameter shows the input variable to which the manipulated value is tracked if the track-manipulated-variable function is enabled on the controller.
IDSTATUS	20.0	WORD	0.0	The state of the identification is displayed at AUTOHOTSPOT
LMN_P	22.0	REAL	0.0	The P-action of the manipulated variable is displayed at the LMN_P parameter.
LMN_I	26.0	REAL	0.0	The I-action of the manipulated variable is displayed at the LMN_I parameter.
LMN_D	30.0	REAL	0.0	The D-action of the manipulated variable is displayed at the LMN_D parameter.
RET_VALU	34.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.

See also

RD_REC: Read data record from I/O (Page 5256)

WR_REC: Write data record to I/O (Page 5260)

Parameter IDSTATUS (S7-300, S7-400)

The IDSTATUS parameter supplies information on the status of identification and contains the four hexadecimal values X, A, I and F.

X				A				I				F			
2^{15}															2^{0}

Value	Description
X	Without meaning (always 0)
A	Action number • 0 = Manual mode (or no closed-loop control mode); • 2 = Closed-loop control; • 4 = Tuning enabled (FUZID_ON = true); • 6 = Transition state from manual mode to 2 or 4;
I	Display "Identification running" and "Parameters determined, but not yet stored in EEPROM" • 0 = Identification not running, no new parameters determined • 1 = Identification running, no new parameters determined • 2 = Identification not running, new parameters determined, but not yet stored in EEPROM • 3 = Identification running, new parameters determined, but not yet stored in EEPROM
F	Error number • 0 = No error • 4 = Excessive step-change of the process value during the identification • 5 = Ratio of time lag to system time constant too large or strongly non-linear behavior of the controlled system. • 6 = Temperature drop or rise during identification start too large. System not settled sufficiently

Block diagrams CH_DIAG_455 (S7-300, S7-400)

Block diagrams CH_DIAG_455

The following diagnostic values of the control deviation are displayed.

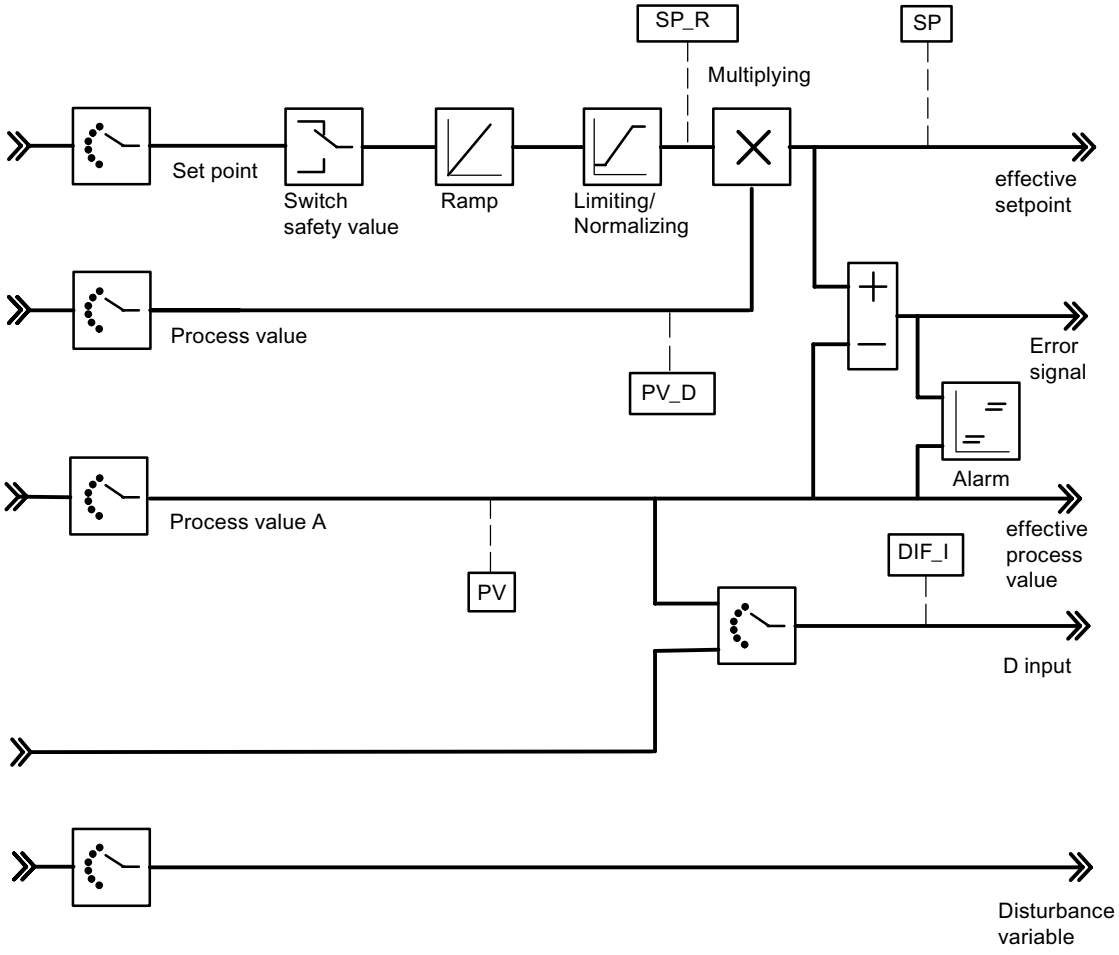

The following values of the control algorithm are displayed.

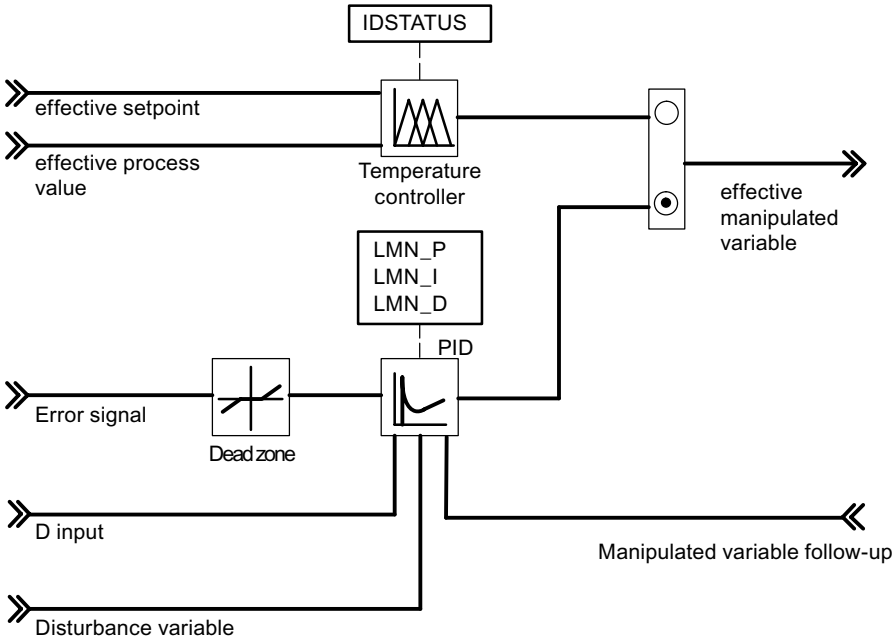

The following values of the continuous-action controller or step controller are displayed.

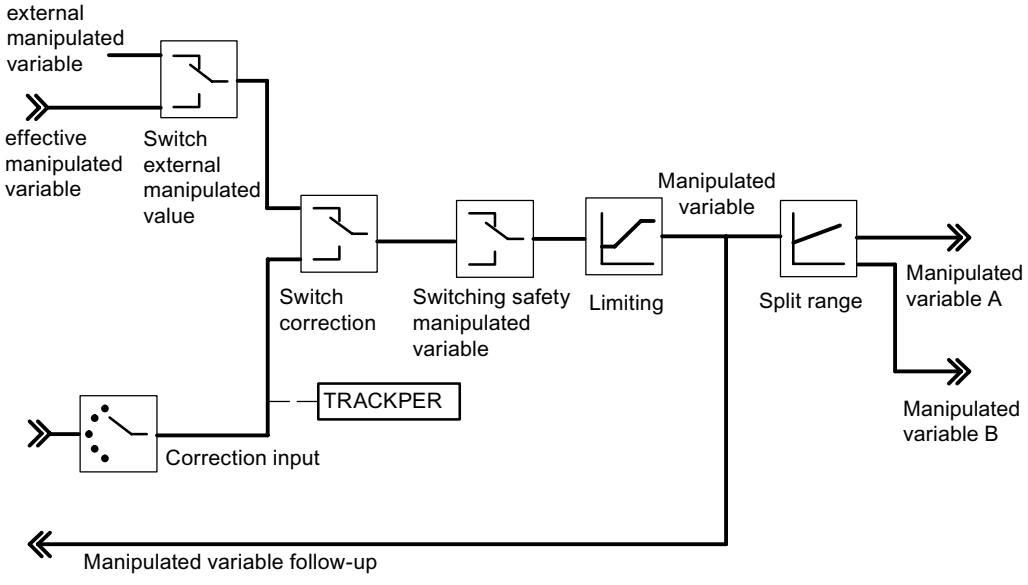

PID_PAR_455 (S7-300, S7-400)

Description PID_PAR_455 (S7-300, S7-400)

The instruction PID_PAR_455 is used for the online modification of parameters that cannot be defined via the instruction PID_FM_455. To view the parameters that can be modified, refer to the table Alterable parameters

Call

PID_PAR_455 requires an initialization routine, which is initiated by setting the parameter COM_RST = TRUE. To save run time, PID_PAR_455 should not be called cyclically. It should only be called when parameters are to be changed. COM_RST must then be FALSE.

PID_PAR_455 has to be called in the same OB as all the other instructions that access the same FM 455.

Mode of operation

In the initialization routine, PID_PAR_455 reads the parameters from the system data and stores them in static variables. Per call, PID_PAR_455 modifies one of these REAL variables and one of these INT variables. At the input parameter INDEX_R or INDEX_I, specify the index numbers of the parameter that you want to modify. At the input parameter VALUE_R or VALUE_I, enter the new values. PID_PAR_455 transfers the entire data record to FM 455 with the modified variables.

To modify additional parameters you have to call **the same** instance DB several times consecutively with COM_RST = FALSE and with different index numbers.

The COM_RST parameter is an input parameter that is not reset by PID_PAR_455.

Note

Please note that parameters that you have changed using PID_PAR_455 are overwritten by the parameters from the system data during the start-up.

Start-up

During CPU start-up and during CPU STOP-RUN transition, the parameters on the FM 455 are overwritten with the parameters from the SDB of the CPU.

Example

During operation you want to modify the ramp-up time for the reference variable and, depending on the process state, use different analog input values as process value.

- Call PID_PAR_455 using COM_RST = TRUE during the start-up of the CPU.
- To configure the ramp-up time for the reference variable at 10.0, use INDEX_R = 30, VALUE_R = 10.0 and INDEX_I = 0 to call PID_PAR_455 during operation.
- If you want to parameterize the analog input value 4 of the module as the process value, call PID_PAR_455 with INDEX_R = 0, INDEX_I = 50 and VALUE_I = 4 during operation.

Input parameters PID_PAR_455 (S7-300, S7-400)

Table 4-280

Parameters	Address	Data type	Default	Description
COM_RST	0.0	BOOL	TRUE	If the parameter COM_RST = TRUE, the instruction PID_PAR_455 executes an initialization routine. During this initialization process the parameters are read from the system data of the CPU and saved in the instance DB.
MOD_ADDR	2.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	4.0	INT	1	The number of the controller channel to which the instance DB is referenced is configured at the "Channel number" input. Values of 1 to 16 are permitted.
INDEX_R	6.0	INT	0.0	• 0: No REAL parameters are changed • 1 to 48: Each value corresponds to an alterable REAL parameter.
VALUE_R	8.0	REAL	0.0	Depends on the relevant parameter
INDEX_I	12.0	INT	0.0	• 0: No INT parameters are changed. • 49 to 61:: Each value corresponds to an alterable INT parameter.
VALUE_I	14.0	INT	0.0	Depends on the relevant parameter

Output parameters PID_PAR_455 (S7-300, S7-400)

Table 4-281

Parameters	Address	Data type	Default	Description
RET_VALU	16.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.
BUSY	18.0	BOOL	FALSE	If BUSY = TRUE, the parameters have not yet been applied by the module. The instruction PID_PAR_455 should then be called again in the next cycle.

See also

RD_REC: Read data record from I/O (Page 5256)

WR_REC: Write data record to I/O (Page 5260)

Alterable parameters (S7-300, S7-400)

Table 4-282 List of the REAL and INT parameters on a FM 455 that are to be modified with PID_PAR_455

Data type	Description	Index number
-	No parameter selected	0
REAL	Filter time constants for analog input	1
REAL	End of scale (100%)	2

Data type	Description	Index number
REAL	Start of scale (0%)	3
REAL	Polyline, interpolation point 1 input side	4
REAL	Polyline, interpolation point 2 input side	5
REAL	Polyline, interpolation point 3 input side	6
REAL	Polyline, interpolation point 4 input side	7
REAL	Polyline, interpolation point 5 input side	8
REAL	Polyline, interpolation point 6 input side	9
REAL	Polyline, interpolation point 7 input side	10
REAL	Polyline, interpolation point 8 input side	11
REAL	Polyline, interpolation point 9 input side	12
REAL	Polyline, interpolation point 10 input side	13
REAL	Polyline, interpolation point 11 input side	14
REAL	Polyline, interpolation point 12 input side	15
REAL	Polyline, interpolation point 13 input side	16
REAL	Polyline, interpolation point 1 output side	17
REAL	Polyline, interpolation point 2 output side	18
REAL	Polyline, interpolation point 3 output side	19
REAL	Polyline, interpolation point 4 output side	20
REAL	Polyline, interpolation point 5 output side	21
REAL	Polyline, interpolation point 6 output side	22
REAL	Polyline, interpolation point 7 output side	23
REAL	Polyline, interpolation point 8 output side	24
REAL	Polyline, interpolation point 9 output side	25
REAL	Polyline, interpolation point 10 output side	26
REAL	Polyline, interpolation point 11 output side	27
REAL	Polyline, interpolation point 12 output side	28
REAL	Polyline, interpolation point 13 output side	29
REAL	Ramp-up time for control variable	30
REAL	Safety reference variable or safety reference variable ratio	31
REAL	Offset for the setpoint logic operation (ratio / mixed controllers)	32
REAL	Factor for process value B (three-component controllers)	33
REAL	Factor for process value C (three-component controllers)	34
REAL	Offset for process value logic operation (three-component controllers)	35
REAL	Factor for disturbance variable logical operation	36
REAL	Operating point	37
REAL	Aggressiveness at fuzzy controller	38
REAL	Vertices for split-range function: Start of input signal A area	39
REAL	Vertices for split-range function: End of input signal A area	40
REAL	Vertices for split-range function: Start of output signal A area	41
REAL	Vertices for split-range function: End of output signal A area	42
REAL	Vertices for split-range function: Start of input signal B area	43
REAL	Vertices for split-range function: End of input signal B area	44
REAL	Vertices for split-range function: Start of output signal B area	45

Data type	Description		Index number
REAL	Vertices for split-range function: End of output signal B area		46
REAL	Minimum pulse time		47
REAL	Minimum break time		48
INT	Selection of the reference variable SP or SP_RE for the controller		49
	0	Setpoint SP_RE of the instruction	
	1 to 16	Analog input value 1 to 16	
	17 to 32	Manipulated variable (LMN) of controllers 1 to 16	
INT	Selecting the main controlled variable process value A for the controllers		50
	0:	Process value A = 0.0	
	1 to 16:	Analog input value 1 to 16	
INT	Selecting the main controlled variable process value B for the controllers		51
	0	Process value B = 0.0	
	1 to 16	Analog input value 1 to 16	
INT	Selecting the auxiliary controlled variable process value C for the controllers		52
	0	Process value C = 0.0	
	1 to 16	Analog input value 1 to 16	
INT	Selecting the auxiliary controlled variable process value D for the controllers		53
	0	Process value D = 0.0	
	1 to 16	Analog input value 1 to 16	
	17 to 32	Manipulated variable (LMN) of controllers 1 to 16	
INT	Selection of the disturbance variable DISV for the controller		54
	0	Disturbance variable = 0.0	
	1 to 16	Analog input value 1 to 16	
INT	Selection of position follow-up TRACK_PER for the controller		55
	0	Position follow-up = 0.0	
	1 to 16	Analog input value 1 to 16	
INT	Selection of position follow-up LMNR_PER for the controller		56
	0	Position follow-up = 0.0	
	1 to 16	Analog input value 1 to 16	
INT	Selecting the signal for changeover to safety value for the manipulated value of the controller		57
	0	Can only be assigned via SAFE_ON of the instruction PID_FM_455	
	1 to 16	Assignment via SAFE_ON parameter of the instruction PID_FM_455 ORed with digital input 1 to 16	
INT	Selecting the signal for changeover to the follow-up function of the manipulated value of the controller		58
	0	Can only be assigned via LMNTRKON of the instruction PID_FM_455	
	1 to 16	Assignment via LMNTRKON parameter of the instruction PID_FM_455 OR-ed with digital input 1 to 16	
INT	Selecting the signal for changing over the manipulated value of the controller to LMN_RE		59
	0	Can only be assigned via LMN_REON of the instruction PID_FM_455	
	1 to 16	Assignment via LMN_REON parameter of the instruction PID_FM_455 OR-ed with digital input 1 to 16	

Data type	Description		Index number
INT	Selecting the high limit signal of the position feedback		60
	0	Can only be assigned via LMNRHSRE of the instruction PID_FM_455	
	1 to 16	Assignment via LMNRHSRE parameter of the instruction PID_FM_455 OR-ed with digital input 1 to 16	
INT	Selecting the low limit signal of the position feedback		61
	0	Can only be assigned via LMNRLSRE of the instruction PID_FM_455	
	1 to 16	Assignment via LMNRLSRE parameter of the instruction PID_FM_455 OR-ed with digital input 1 to 16	

CJ_T_PAR_455 (S7-300, S7-400)

Description CJ_T_PAR_455 (S7-300, S7-400)

The instruction CJ_T_PAR_455 is used for online modification of the configured reference junction temperature. This is necessary if a temperature control system with several FM 455 with thermoelement inputs is to be operated without having to connect a Pt 100 to each FM 455.

Call

CJ_T_PAR_455 requires an initialization routine. To this purpose it has to be called once with the COM_RST = TRUE parameter at the start-up of the CPU.

CJ_T_PAR_455 is normally called cyclically. For this purpose COM_RST should be set to FALSE for runtime reasons. The COM_RST parameter is an input parameter that is not reset.

CJ_T_PAR_455 has to be called in the same OB as all the other instructions that access the same FM 455.

Example

If an FM 455 measures the reference junction temperature of an extruder control system with more than 16 heating zones, this can be read out by READ_455 at parameter CJ_TEMP and can be parameterized for the other FM 455 via CJ_T_PAR_455.

Input parameters CJ_T_PAR_455 (S7-300, S7-400)

Table 4-283

Parameters	Address	Data type	Default	Description
COM_RST	0.0	BOOL	–	If the parameter COM_RST = TRUE, the instruction CJ_T_PAR_455 executes an initialization routine. During this initialization process the parameters are read from the system data of the CPU and saved in the instance DB.
MOD_ADDR	2.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CJ_T	4.0	REAL	0.0	The reference junction temperature can be specified at parameter CJ_T. The permissible value range depends on the sensor type.

Output parameters CJ_T_PAR_455 (S7-300, S7-400)

Table 4-284

Parameters	Address	Data type	Default	Description
RET_VALU	8.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.
BUSY	10.0	BOOL	FALSE	If BUSY = TRUE, the parameters have not yet been applied by the module. The instruction CJ_T_PAR_455 should then be called again in the next cycle.

See also

RD_REC: Read data record from I/O (Page 5256)

WR_REC: Write data record to I/O (Page 5260)

LP_ZONE (S7-300, S7-400)

Description LP_ZONE (S7-300, S7-400)

To optimize load on CPU resources and processing of the instruction ZONE at constant bus cycle times at during controller optimization, it is necessary to use the loop scheduler LP_ZONE.

LP_ZONE ensures that only one zone is processed per call of the cyclic interrupt OB.

The instruction LP_ZONE calculates the zone sampling times and transfers them to the instances of instruction ZONE.

Call

LP_ZONE is called in a start-up or cyclic interrupt OB. The instructions LP_ZONE, ADM_ZONE and ZONE must all be called at the same cyclic interrupt level, preferably directly in the cyclic interrupt OB. It is not permitted to program separate time reduction ratios for the blocks.

LP_ZONE requires an initialization routine after the insertion or removal of a zone or during the the modification of a call sequence. This initialization routine must be called with the parameter COM_RST = TRUE. The COM_RST bit is passed to the individual zones and their administration block via the LP_SCH interface.

The zone numbers NBR_CALL of the instances of the instructions ZONE and ADM_ZONE and the maximum zone number MNR_CALL of the instruction LP_ZONE are automatically set according to the following routine:

Initial execution of the cyclic interrupt OB after COM_RST of the instruction LP_ZONE has been set:

- LP_ZONE sets the call counter LP_SCH. CALL_CNT to 1.
- ZONE and ADM_ZONE read the call counter and write the value to their in/out parameterNBR_CALL. ZONE increments the call counter by the count by 1. ADM_ZONE leaves it unchanged.

Second run of the cyclic interrupt OB after COM_RST of the instruction LP_ZONE has been set:

- LP_ZONE reads the call counter (total number of zones) and writes the value to MNR_CALL.

Mode of operation

Parameter LP_SCH.CALL_CNT is incremented by the count of 1 at every call of the instruction LP_ZONE. The counter is reset to 1 after it has reached the maximum zone number of MNR_CALL. When the ZONE instruction is called, it first check the counter LP_SCH.CALL_CNT. If the counter value and internal zone number NBR_CALL match, the instruction executes its program. The parameter MNR_CALL of the instruction LP_ZONE and NBR_CALL of the instruction ADM_ZONE and ZONE are automatically set by the blocks after a restart.

Note

Interconnect the in/out parameter LP_SCH of the instructions ZONE and ADM_ZONE to the internal structure LP_SCH of the instance DB of the instruction LP_ZONE.

Interconnect input parameter OB_EXC_FREQ of the instruction LP_ZONE to the temporary parameter OBxx_EXC_FREQ of the cyclic interrupt OB.

Example

Call sequence in OB33

```
OB33
 ...
 call LP_ZONE, DB_LP_ZONE
 call ADM_ZONE, DB_ADM_ZONE
 call ZONE, DB_ZONE1
 call ZONE, DB_ZONE2
 call ZONE, DB_ZONE3
 …
 BE
```

leads the following zone processing cycle:

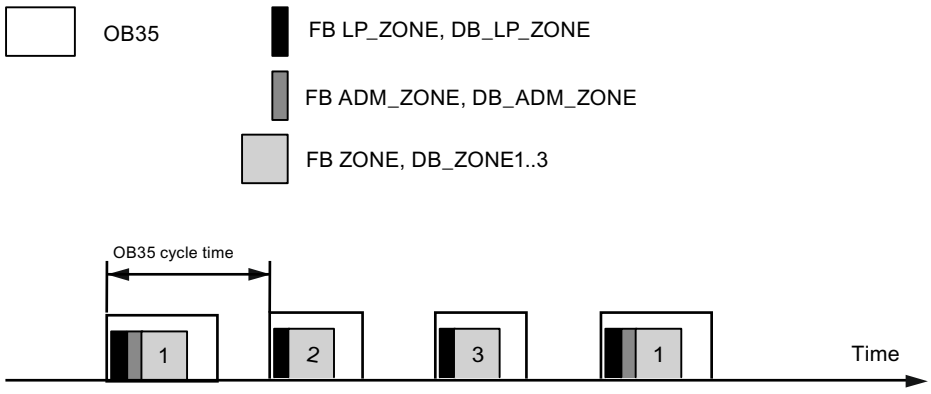

Input parameters LP_ZONE (S7-300, S7-400)

Table 4-285

Parameters	Address	Data type	Default	Description
OB_EXC_FREQ	0.0	INT	0	Sampling time [ms]; interconnect the parameter with the corresponding local variables of the cyclic interrupt OB.

Output parameters LP_ZONE (S7-300, S7-400)

Table 4-286

Parameters	Address	Data type	Default	Description
MNR_CALL	2.0	INT	0	Number of zones. Defined automatically after restart.

In/out parameters LP_ZONE (S7-300, S7-400)

Table 4-287

Parameters	Address	Data type	Default	Description
COM_RST	4.0	BOOL	TRUE	Restart; make sure that this bit is set in OB100.

Static variables LP_ZONE (S7-300, S7-400)

Table 4-288

Parameters	Address	Data type	Default	Description
LP_SCH	6.0	STRUCT		Interface between the instructions LP_ZONE and ADM_ZONE or ZONE .

This is followed by internal parameters which may not be used by users.

ADM_ZONE (S7-300, S7-400)

Description ADM_ZONE (S7-300, S7-400)

In practice, heating/cooling zones are designed to allow relatively unhindered heat transfer between zones. In such cases, the thermally coupled zones must be organized into a group. The instruction ADM_ZONE manages a group of heating/cooling zones and handles the following functions:

- Switching the heating of all zones on and off
- Enabling and disabling controller optimization
- Group display for channel faults
- Group display for controller optimization errors
- Group display for controller optimization warnings
- Control of the global manipulated value step-changes during controller optimization

Call

Call the instructions LP_ZONE, ADM_ZONE and ZONE at the same cyclic interrupt level, preferably immediately in the cyclic interrupt OB. It is not permitted to program separate time reduction ratios for the blocks. Call ADM_ZONE after LP_ZONE and before ZONE.

Start-up

Set the restart bit of ADM_ZONE, for example in OB100, or in the first cycle of the cyclic interrupt OB. The heating is disabled at the selected zones after restart and any active controller optimization is canceled.

Mode of operation

The actual program is only executed if the call counter of the instruction LP_ZONE has the same value as the internal call number: LP_SCH.CALL_CNT = NBR_CALL.

Switching the heating of all zones on and off

Changes at switch HEAT_OFF are transferred as actions to the ADMIN interface and are successively passed to the zones to be administered. If no action is pending, the result is transferred from the ADMIN.QHEAT_OFF zones to in/out parameter HEAT_OFF. Group switch HEAT_OFF is only set to TRUE if the heating is shut down at all zones.

User actions at the HEAT_OFF switch are indicated at the done parameter QHO_DONE. The parameter is set to TRUE by default. It is set to FALSE after a user action. QHO_DONE is reset to TRUE after the heating has been switched on or off at all associated zones.

A zone heating is only in disabled state if the following condition is met: cont_FM455.QLMN_RE = TRUE and cont_FM455.LMN = 0.0

Switching controller optimization on and off at all zones

Changes at switch TUN_ON are transferred as actions to the ADMIN interface and are successively passed to the zones to be administered. If no action is pending, the result is transferred from the ADMIN.QTUN_ON zones to in/out parameter TUN_ON. Group switch TUN_ON is only set to FALSE if controller optimization is disabled at all zones.

Group display for channel faults

Bit QCH_F indicates whether a channel error of FM 455 is pending on at least one zone. Check on which zone a channel error has occurred. If QCH_F = TRUE, then the precise error information is read out in the diagnostic record DS1 of the module.

Group display for controller optimization errors

Bit QTUN_ERR indicates if an error has occurred in at least one instance DB of the ZONESTATUS_H = 3xxxx or STATUS_C = 3xxxx instruction. Check the corresponding zones and to remedy errors follow the steps described at AUTOHOTSPOT.

Group display for controller optimization warnings

Bit QTUN_ERR indicates if an error has occurred in at least one instance DB of the ZONESTATUS_H = 2xxxx or STATUS_C = 2xxxx instruction. Check the corresponding zones and to remedy errors follow the steps described at AUTOHOTSPOT.

Sequential control during controller optimization

The administration block ensures that the manipulated value step-change is carried out simultaneously at all zones of this group during controller optimization. After having found their point of inflection, all zones change over to phase 7.

Note

Interconnect the in/out parameter ADMIN of the instruction ZONE with the internal structure ADMIN of the instance DB of ADM_ZONE.

ADM_ZONE is always used in conjunction with LP_ZONE.

Output parameters ADM_ZONE (S7-300, S7-400)

Table 4-289

Parameters	Address	Data type	Default	Description
QCH_F	0.0	BOOL	FALSE	Group message: Channel error
QTUN_ERR	0.1	BOOL	FALSE	Group message: Error during controller optimization

Parameters	Address	Data type	Default	Description
QTUN_WRN	0.2	BOOL	FALSE	Group message: Warning during controller optimization
QHO_DONE	0.3	BOOL	TRUE	Feedback of a HEAT_OFF operation • 0: HEAT_OFF operation is triggered • 1: HEAT_OFF operation is active at all zones
NBR_CALL	2.0	INT	0	Call number; entered automatically by the instruction after restart.

In/out parameters ADM_ZONE (S7-300, S7-400)

Table 4-290

Parameters	Address	Data type	Default	Description
HEAT_OFF	4.0	BOOL	TRUE	Heating off/on; 1 = off
TUN_ON	4.1	BOOL	FALSE	Controller optimization on/off; 1 = on
COM_RST	4.2	BOOL	TRUE	Restart; make sure that this bit is set in OB100.
LP_SCH	6.0	STRUCT	P#P 0.0	Interface between the instructions LP_ZONE and ADM_ZONE or ZONE.

Static variables ADM_ZONE (S7-300, S7-400)

Table 4-291

Parameters	Address	Data type	Default	Description
ADMIN	12.0	STRUCT		Interface between the instructions ADM_ZONE and ZONE .

This is followed by internal parameters which may not be used by users.

ZONE (S7-300, S7-400)

Description ZONE (S7-300, S7-400)

The instance DB of the instruction ZONE contains the controller data of an individual heating or heating/cooling zone. The instruction ZONE reads and writes data to a channel of FM 455.

The controller functions of FM 455 are supplemented with the following functions:

- Control zone operation
- Ratio factor for heating/cooling zones
- Controller optimization

ZONE is therefore particularly suitable for controlling the heating and heating/cooling zones of extruders.

Requirements

The channel of the FM 455 is configured as pulse controller or continuous controller.

The instructions PID_FM_455 and PID_PAR_455 are copied to the user program.

Call

Call the instructions LP_ZONE, ADM_ZONE and ZONE at the same cyclic interrupt level, preferably immediately in the cyclic interrupt OB. It is not permitted to program separate time reduction ratios for the blocks. It is essential to position the associated zones in the block after the call of the instruction ADM_ZONE.

Start-up

In OB100, set the restart bit of ZONE. The bit is set automatically if you are using LP_ZONE.

Following actions are carried out when initialization routine COM_RST=TRUE

is initiated:

- cont_FM455.SDB_SEL = TRUE: SDB controller parameters cont_FM455.P_SEL...cont_FM455.BREAK_TM are not applied by FM 455. The parameters active in the FM remain active.
- cont_FM455.SP_OP_ON = FALSE: This setting activates setpoint SP_RE.
- cont_FM455.LOAD_PAR = TRUE: The controller parameters are transferred from the instance DB to FM 455.

Mode of operation

ZONE includes the instructions PID_FM_455 and PID_PAR_455 as local instances.

ZONE sets the parameters cont_FM455.LOAD_OP = TRUE and cont_FM455.READ_VAR = TRUE at each call. All operating parameters (cont_FM455.SP_RE...cont_FM455.LMNRSVAL) and all process values (cont_FM455.SP...cont_FM455.QMOD_F) are updated as a result.

Controller parameters (cont_FM455.P_SEL...cont_FM455.BREAK_TM) can be modified directly by entering a new value at the instance DB on the CPU and setting the cont_FM455.LOAD_PAR bit for the duration of one cycle. You do not require an auxiliary DB. The bit is reset on successful completion of the transfer. After the restart of the CPU the controllers are not overwritten by the controller parameters of the SDB.

If you are working without LP_ZONE, you must set parameter NBR_CALL=0.

Parameters of instruction PID_FM_455 that are alterable with ZONE (S7-300, S7-400)

Table 4-292 The following list contains the parameters of instruction PID_FM_455 that are affected by ZONE:

Parameters	Comment
cont_FM455.READ_VAR	Set cyclically to TRUE
cont_FM455.LOAD_OP	Set cyclically to TRUE

Parameters	Comment
cont_FM455.LOAD_PAR	Set to TRUE during controller optimization and during modification of the CONZONE and RATIOFAC parameters.
cont_FM455.MOD_ADDR	Set cyclically at input MOD_ADDR.
cont_FM455.CHANNEL	Set cyclically at input CHANNEL.
cont_FM455.SDB_SEL	Set to TRUE at restart. At the CPU transition from RUN to STOP, FM 455 rejects the controller parameters (P_SEL...BREAK_TM) of the SDB.
cont_FM455.SP_OP_ON	Set to FALSE at the restart so that SP_RE is active.
cont_FM455.SP_RE	Set cyclically to SP_IN
cont_FM455.GAIN cont_FM455.TI cont_FM455.TD cont_FM455.TM_LAG cont_FM455.D_EL_SEL cont_FM455.PFDB_SEL cont_FM455.LOAD_PAR	Set during controller optimization, when restoring controller parameters from backup, and when loading PI/PID controller parameters.
cont_FM455.LMN_LLM cont_FM455.LOAD_PAR	Set after modification of RATIOFAC and COOLZONE
cont_FM455.LMN_REON cont_FM455.LMN_RE cont_FM455.LOAD_OP	Set cyclically

Parameters of instruction PID_PAR_455 that are alterable with ZONE (S7-300, S7-400)

Table 4-293 The following list contains the parameters of instruction PID_PAR_455, that are affected by ZONE:

Parameters	Comment
pidpar_FM455.COM_RST	Set to TRUE at restart
pidpar_FM455.MOD_ADDR	Set cyclically at input MOD_ADDR.
pidpar_FM455.CHANNEL	Set cyclically at input CHANNEL.
pidpar_FM455. INDEX_R	Set to 43 after modification of RATIOFAC and COOLZONE
pidpar_FM455. VALUE _R	Set after modification of RATIOFAC and COOLZONE:

Input parameters ZONE (S7-300, S7-400)

Table 4-294

Parameters	Ad-dress	Data type	Default	Description
MOD_ADDR	0.0	INT	512	Module address
CHANNEL	2.0	INT	1	Channel number

Parameters	Address	Data type	Default	Description
LPSCH_EN	4.0	BOOL	TRUE	If LPSCH_EN = TRUE, ZONE is controlled via the instruction LP_ZONE. The in/out parameter LP_SCH must be interconnected with the static variable LP_SCH of the instruction LP_ZONE. Otherwise the CPU goes to STOP with "Range error while reading". If LPSCH_EN = FALSE, ZONE is processed at each cyclic interrupt cycle. It is not necessary to supply the in/out parameter LP_SCH. The sampling time must be entered at a static variable CYCLE.
ADMIN_EN	4.1	BOOL	TRUE	If ADMIN_EN = TRUE, several heating/cooling zones are handled via the instruction ADM_ZONE. The in/out parameter ADMIN must be interconnected with the static variable ADMIN of the instruction ADM_ZONE. Otherwise the CPU goes to STOP with "Range error while reading". If ADMIN_EN = FALSE, ZONE works without the instruction ADM_ZONE. It is not necessary to supply the in/out parameter ADMIN.

In/out parameters ZONE (S7-300, S7-400)

Table 4-295

Parameters	Address	Data type	Default	Description
LP_SCH	6.0	STRUCT	P#P 0.0	Interface between the instructions LP_ZONE and ADM_ZONE or ZONE.
ADMIN	12.0	STRUCT	P#P 0.0	Interface between the instructions ADM_ZONE and ZONE .

Static variables ZONE (S7-300, S7-400)

Table 4-296

Parameters	Address	Data type	Default	Description
cont_FM455	18.0	FB PID_FM_455		Local instance of the instruction PID_FM_455
pid-par_FM455	172.0	FB PID_PAR_455		Local instance of the instruction PID_PAR_455
SP_IN	426.0	REAL	0.0	The input SP_IN is used to specify a setpoint.
CYCLE	430.0	REAL	0.1	Sets the sampling time for the PID algorithm. If used, the instruction LP_ZONE automatically calculates the sampling time of LP_ZONE.
H_OFF_EN	434.0	BOOL	TRUE	TRUE = switch HEAT_OFF is active and affects the parameters MAN_ON and MAN.
TUN_EN	434.1	BOOL	TRUE	No new controller parameters are calculated if TUN_EN=0 during controller optimization. Caution: the manipulated value step-change is enabled regardless of this setting.

Parameters	Address	Data type	Default	Description
COOL-ZONE	434.2	BOOL	FALSE	COOLZONE = FALSE: Self-contained heating zone COOLZONE = TRUE: Heating/cooling zone
PID_ON	434.3	BOOL	TRUE	At the PID_ON input, you can specify whether or not the tuned controller will operate as a PI or PID controller. PID controller: PID_ON = TRUE PI controller: PID_ON = FALSE With certain process types it is nevertheless possible that only a PI controller will be designed despite PID_ON = TRUE.
QCH_F	434.4	BOOL	FALSE	QCH_F = TRUE: A channel error of FM 455 has occurred during the calling of the local instance of the instruction PID_FM_455 (parameter cont_FM455.QCH_F).
QTUN_ERR	434.5	BOOL	FALSE	QTUN_ERR = TRUE: Incorrect operator action occurred during controller optimization. STATUS_H or STATUS_C = 3xxxx
QTUN_WRN	434.6	BOOL	FALSE	QTUN_WRN = TRUE: : Warning message occurred during controller optimization. STATUS_H or STATUS_C = 2xxxx
QTUNSTCP	434.7	BOOL	FALSE	QTUNSTCP = TRUE: Tuning was initiated in cold process state and the controller is to be corrected to the operating point after optimization. MAN_ON = FALSE; MAN = 0 (Heating off) SP_IN > PV0 +10 (At the start of optimization, the operating point is greater than the process value by the amount of 10)
QCONTINU	435.0	BOOL	TRUE	Indicates whether the zone is ready for the next phase of controller optimization
PHASE	436.0	INT	0	The currently active phase of controller optimization is indicated at the Parameter PHASE (0 to 7, 11 to13).
STATUS_H	438.0	INT	0	Parameter STATUS_H indicates a diagnostic value via the search for the point of inflection during the heating process.
STATUS_C	440.0	INT	0	Parameter STATUS_C indicates a diagnostic value via the search for the point of inflection during the cooling process.
STATUS_D	442.0	INT	0	Parameter STATUS_D indicates a diagnostic value via the controller design during the heating process.
PI_CON	444.0	STRUCT		PI controller parameters
GAIN	+0.0	REAL	0.0	PI controller gain
TI	+4.0	REAL	0.0	PI integration time [s]
	=8.0	END_STRUCT		
PID_CON	452.0	STRUCT		PID controller parameters
GAIN		REAL	0.0	PID controller gain
TI		REAL	0.0	PID integration time [s]
TD		REAL	0.0	PID derivative action time [s]
	=12.0	END_STRUCT		
PAR_SAVE	464.0	STRUCT		The PID parameters are saved in this structure.
GAIN	+0.0	REAL	0.0	Controller gain

Parameters	Address	Data type	Default	Description
TI	+4.0	REAL	0.0	Integration time [s]
TD	+8.0	REAL	0.0	Derivative action time (s)
TM_LAG	+12.0	REAL	0.0	Time lag of the D-action [s]
D_EL_SEL	+16.0	INT	0	D-action element input
PFDB_SEL	+18.0	BOOL	FALSE	P-action in feedback loop
CONZ_ON	+18.1	BOOL	FALSE	Enable control zone
CON_ZONE	+20.0	REAL	100.0	Control zone band
RATIOFAC	+24.0	REAL	1.0	Ratio of heating gain/cooling gain of the process.
	=28.0	END_STRUCT		
HEAT_OFF	492.0	BOOL	FALSE	TRUE = disable heating You must set H_OFF_EN = TRUE to activate HEAT_OFF. • MAN_ON=TRUE and MAN=0.0 when HEAT_OFF is set. • MAN_ON=FALSE when HEAT_OFF is reset.
TUN_ON	492.1	BOOL	FALSE	Enable controller optimization Set TUN=TRUE only for one cycle. TUN_ON is reset automatically after tuning.
CTUN_ON	492.2	BOOL	FALSE	Enable only cooling optimization Set CTUN =TRUE only for one cycle. CTUN _ON is reset automatically after tuning.
COM_RST	492.3	BOOL	FALSE	A initialization routine is initiated
TUN_LMN	494.0	REAL	80.0	Process excitation for controller optimization starting in cold process state is initiated by a manipulated value step by the amount of TUN_LMN.
TUN_DLMN	498.0	REAL	20.0	Process excitation for controller optimization at the operating point is initiated by a manipulated value step by the amount of TUN_DLMN.
TUN_CLMN	502.0	REAL	-10.0	Manipulated value after the start of cooling optimization.
STAT_TM	506.0	REAL	30.0	Time for steady state condition for controller optimization phase 1 or 11 The process is considered in steady state if the process value or manipulated value do not change by more than the amount of 2*STAT_D_PV or 2*STATDLMN within the time STAT_TM.
STAT_DPV	510.0	REAL	1.0	Process value delta for the steady state condition in controller optimization phase 1 or 11 See STAT_TM .
STATDLMN	514.0	REAL	4.0	Manipulated value delta for the steady state condition in controller optimization phase 1 or 11 See STAT_TM.
RATIOFAC	518.0	REAL	1.0	Ratio of heating gain/cooling gain of the process. The low output limit and "Range start of the input signal of manipulated value B of the split-range function" are adapted automatically.

Parameters	Address	Data type	Default	Description
CON_ZONE	522.0	REAL	100.0	Control zone If ER >= CON_ZONE, then LMN = LMN_HLM. If ER <= -CON_ZONE (or -CON_ZONE /RATIOFAC), then LMN = LMN_LLM.
MAN	526.0	REAL	0.0	Manual value HEAT_OFF affects the parameter if H_OFF_EN=TRUE .
MAN_ON	530.0	BOOL	TRUE	Manual/automatic mode changeover HEAT_OFF affects the parameter if H_OFF_EN=TRUE .
CONZ_ON	530.1	BOOL	FALSE	TRUE = control zone enabled
UNDO_PAR	530.2	BOOL	FALSE	Copies the controller parameters of structure PAR_SAVE to the following parameters: • cont_FM455.GAIN • cont_FM455.TI • cont_FM455.TD • cont_FM455.TM_LAG • cont_FM455.D_EL_SEL • cont_FM455.PFDB_SEL • CONZ_ON • CON_ZONE • RATIOFAC The parameters are then downloaded to FM 455. This function is not executed in automatic mode.
SAVE_PAR	530.3	BOOL	FALSE	Saves the following controller parameters to the structure PAR_SAVE: • cont_FM455.GAIN • cont_FM455.TI • cont_FM455.TD • cont_FM455.TM_LAG • cont_FM455.D_EL_SEL • cont_FM455.PFDB_SEL • CONZ_ON • CON_ZONE • RATIOFAC

Parameters	Address	Data type	Default	Description
LOAD_PID	530.4	BOOL	FALSE	Load PI/PID parameters: If PID_ON=TRUE, it copies the PID parameters of structure PID_CON to the following parameters: • cont_FM455.GAIN • cont_FM455.TI • cont_FM455.TD • cont_FM455.TM_LAG If PID_ON=FALSE, it copies the PI parameters of structure PI_CON to the following parameters: • cont_FM455.GAIN • cont_FM455.TI • cont_FM455.TD = 0.0 • cont_FM455.TM_LAG = 0.5 The parameters are then downloaded to FM455. This function is not executed in automatic mode.
BTUN_DPV	532	REAL	0.0	Process value difference for automatic cancellation of optimization.
BTUN_TM	536	REAL	300.0	Time following the manipulated value step-change for automatic cancellation of optimization.

Variables for controller optimization ZONE (S7-300, S7-400)

Table 4-297

Parameters	Address	Data type	Default	Description
GAIN_P	540.0	REAL	0.0	Identified process gain.
TU	544.0	REAL	0.0	Identified time lag of the process.
TA	548.0	REAL	0.0	Identified recovery time of the process.
KIG	552.0	REAL	0.0	Maximum possible rate of rise of the process value during heating.
KIG_C	556.0	REAL	0.0	Maximum decay rate of the process value during cooling.
N_PTN	560.0	REAL	0.0	Order of the process. "Non-integer values" are also possible.
TM_LAG_P	564.0	REAL	0.0	Time constant of a PTN model. (practical values for N_PTN>=2 only).
T_P_INF	568.0	REAL	0.0	Time from process excitation until the point of inflection.
P_INF	572.0	REAL	0.0	Process value change from process excitation until the point of inflection.
LMN0	576.0	REAL	0.0	Manipulated value at the start of optimization Calculated in phase 1 (mean value).
PV0	580.0	REAL	0.0	Process value at the start of optimization Calculated in phase 1 (mean value).
PVDT0	584.0	REAL	0.0	Process value slew rate at the start of optimization
PVDT	588.0	REAL	0.0	Current process value slew rate
PVDT_MAX	592.0	REAL	0.0	Maximum derivative of the process value at the point of inflection (sign adapted, always > 0); is used to calculate TU and KIG.

Parameters	Address	Data type	Default	Description
NOI_PVDT	596.0	REAL	0.0	Noise action in PVDT_MAX in % The higher the noise action, the less accurate (less aggressive) the control parameters.
NOISE_PV	600.0	REAL	0.0	Absolute process value noise Difference between maximum and minimum process value in phase 1.
FIL_CYC	604.0	INT	1	Number of cycles of mean value filters 1 to 1024; The process value is determined through FIL_CYC cycles. FIL_CYC is increased from 1 to a max. of 1024 if needed.
POI_CMAX	606.0	INT	0	Maximum number of cycles after inflection point This time is used to find another (i.e. better) inflection point for measuring noise. The optimization is completed only after this time.
POI_CYCL	608.0	INT	0	Number of cycles after inflection point
NBR_CALL	610.0	INT	-1	Call number; is entered automatically after restart.

This is followed by internal parameters which may not be used by users.

Parameters STATUS_H and STATUS_C (S7-300, S7-400)

STATUS_H indicates a diagnostic value via the search for the point of inflection during the heating process, STATUS_C beim during the cooling process.

If a diagnostic value 2xxxx is displayed, QTUN_WRN = TRUE

If a diagnostic value 3xxxx is displayed, QTUN_ERR = TRUE

STATUS_ H STATUS_C	Description	Remedy
0	Default, or no/no new controller parameters	
10000	Tuning completed + suitable controller parameters found	
2xxxx	Tuning completed + controller parameters uncertain	
2xx2x	Point of inflection not reached (only when optimization from cold process state to the operating point QTUNSTCP =TRUE)	If the controller oscillates, weaken the controller parameters, or repeat the test with a smaller manipulated value difference TUN_LMN.
2x1xx	Estimation error TU too small TU < 3*CYCLE	Reduce CYCLE and repeat the test. Special case self-contained PT1 process:: Do not repeat the test; weaken the controller parameters if need be.
2x2xx	Estimation error T_P_INF too small T_P_INF < 72.8*CYCLE	see 2x1xx
2x3xx	Estimation error TU too small and estimation error T_P_INF too small	see 2x1xx
3xxxx	Tuning canceled. No controller parameter found.	
30002	For heating optimization: Effective manipulated variable difference < 5% For cooling optimization: TUN_CLMN > -5 %	Correct manipulated value difference TUN_LMN, TUN_DLMN or TUN_CLMN.

STATUS_ H STATUS_C	Description	Remedy
30003	Incorrect manipulated variable low output limit for cooling optimization. cont_FM455.LMN_LLM > -5.0	Set zone as heating/cooling zone COOLZONE =TRUE and check cont_FM455.LMN_LLM.
30004	Effective manipulated value difference is limited by split-range limits and not by manipulated value limits.	Check limits of the manipulated values and split-range. Correct manipulated value difference TUN_LMN, TUN_DLMN or TUN_CLMN.
30005	Sampling time CYCLE deviates by more than 5% from the measured value.	Compare CYCLE with the cycle time of the cyclic interrupt level and monitor possibly existing loop scheduler. Check CPU load. Excessive load on CPU resources leads to extended sampling times that no longer match CYCLE.
30008	An attempt is being made to initiate cooling optimization. However, no heating optimization was carried out beforehand (KIG=0.0).	Start with a heating optimization session.
30009	Safety mode: cont_FM455.QLMNSAFE = TRUE	Exit the safety mode and restart optimization.
30010	No adequate temperature change after modification of heating/cooling output.	Check the signal profile of the manipulated variable and time. Check plausibility of the parameters BTUN_TM and BTUN_DPV.

See also

Output parameters ADM_ZONE (Page 6077)

Parameters STATUS_D (S7-300, S7-400)

STATUS_D	Description
0	No controller parameters were calculated.
110	N_PTN <= 1.5 process type I fast
121	N_PTN > 1.5 process type I
200	N_PTN > 1.9 process type II (transition range)
310	N_PTN >= 2.1 process type III fast
320	N_PTN > 2.6 process type III
111, 122, 201, 311, 321	Parameters have been corrected from phase 7.

Note

STATUS_H = 0 is reset at the end of phase 1. If you cancel optimization in phase 2, STATUS_H = 0 is set. However, STATUS_D still displays the status of the last controller calculation.

The higher the value of STATUS_D, the higher the order of the control process, the greater the TU/TA ratio and the gentler the controller parameters will be.

Parameter PHASE (S7-300, S7-400)

PHASE	Description
0	No optimization mode; automatic or manual mode
1	Steady state test; parameter check; feed forwarding of a constant heating manipulated variable
2	Searching for point of inflection with constant manipulated value
3 (1 cycle per instruction ZONE)	Calculating the process parameters. Saving currently valid controller parameters prior to optimization
4 (1 cycle per instruction ZONE)	Controller design
5 (1 cycle per instruction ZONE)	Following up the controller to the new manipulated variable
7	Check of the process type (only for heating optimization)
10	Cooling optimization Wait until all zones zones are in automatic mode (phase > 2).
11	Cooling optimization Steady state test; parameter check; feed forwarding of a constant cooling manipulated variable
12	Cooling optimization: Searching for point of inflection with constant manipulated value
13	Cooling optimization: Calculating the heating/cooling ratio

Heating Tuning Phases during First Adaption

The following figure shows the phases of self-contained heating optimization as a function of the ambient temperature and operating point:

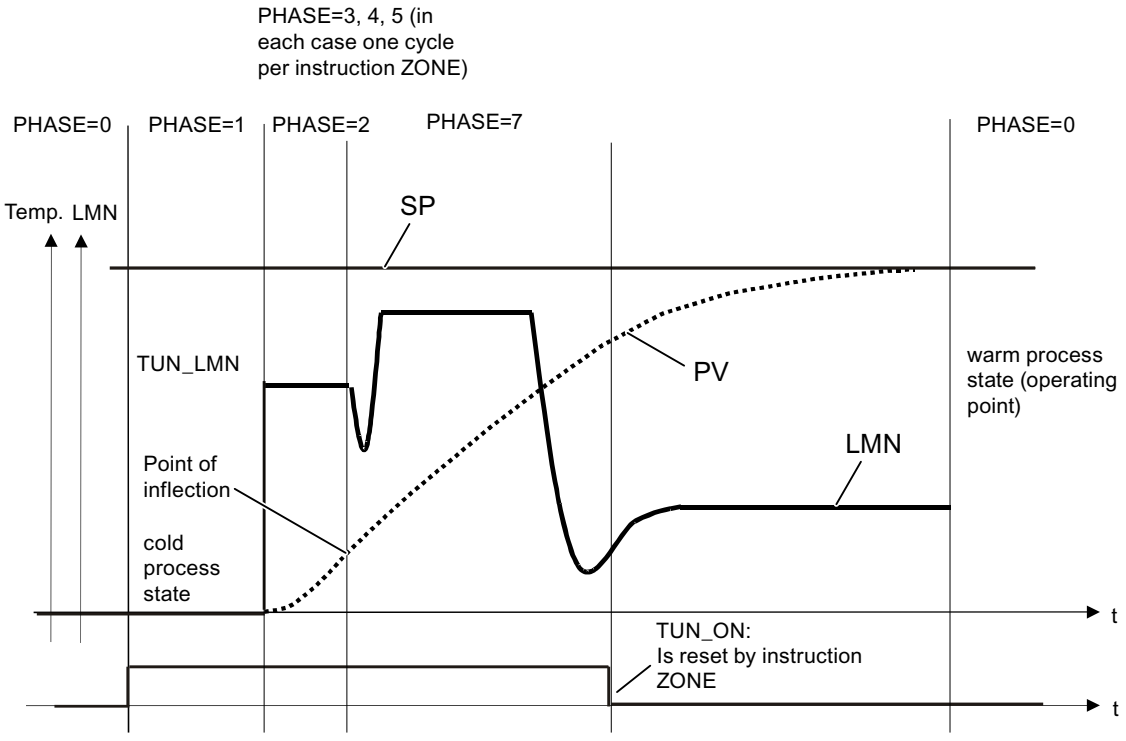

Phases of Heating Tuning at the Operating Point

The following figure shows the phases of a self-contained heating optimization at the operating point.

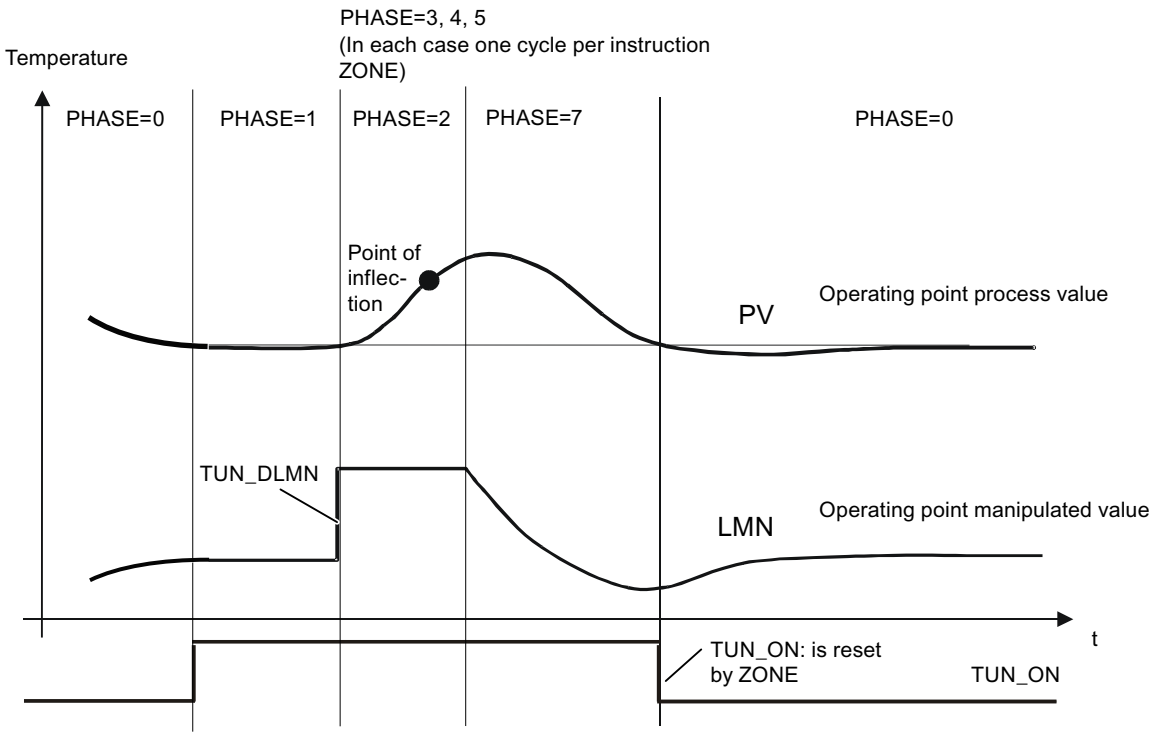

After optimization was completed and the instruction has returned to Phase 0 and TUN_ON=FALSE is set, the parameters STATUS_H, STATUS_C or QTUN_WRN, QTUN_ERR indicate whether optimization was completed without errors.

DB for HMI (S7-300, S7-400)

Description DB 101 to 116 (S7-300, S7-400)

In order to operate the FM 455 if the CPU fails, you require data blocks 101 to 104. You can establish a maximum of three connections from the FM 455 to OPs via MPI.

Operator control of the FM 455 with the OP is only possible in the STOP mode of the CPU or at a CPU failure.

Monitoring of the FM 455 with the OP is always possible.

The variable interface of the FM 455 contains 16 data blocks with the block numbers 101 to 116 for the controller channels 1 to 16 (refer to the following figure).

Note

The contents of the data blocks 101 to 116 do not automatically reflect the parameter value effective at the FM 455. Parameters changed with the OP are only transferred to the FM 455 after the operating bits LOAD_PAR or LOAD_OP have been set.

If you change a parameter using OP operation without setting the corresponding operating bit, the changed parameter value is entered in the data block, but the FM 455 continues to operate internally with the unchanged old value of the parameter.

After the operating bits have been set and the parameters have been transferred to the FM 455, the operating bits LOAD_PAR and LOAD_OP are reset by the FM 455.

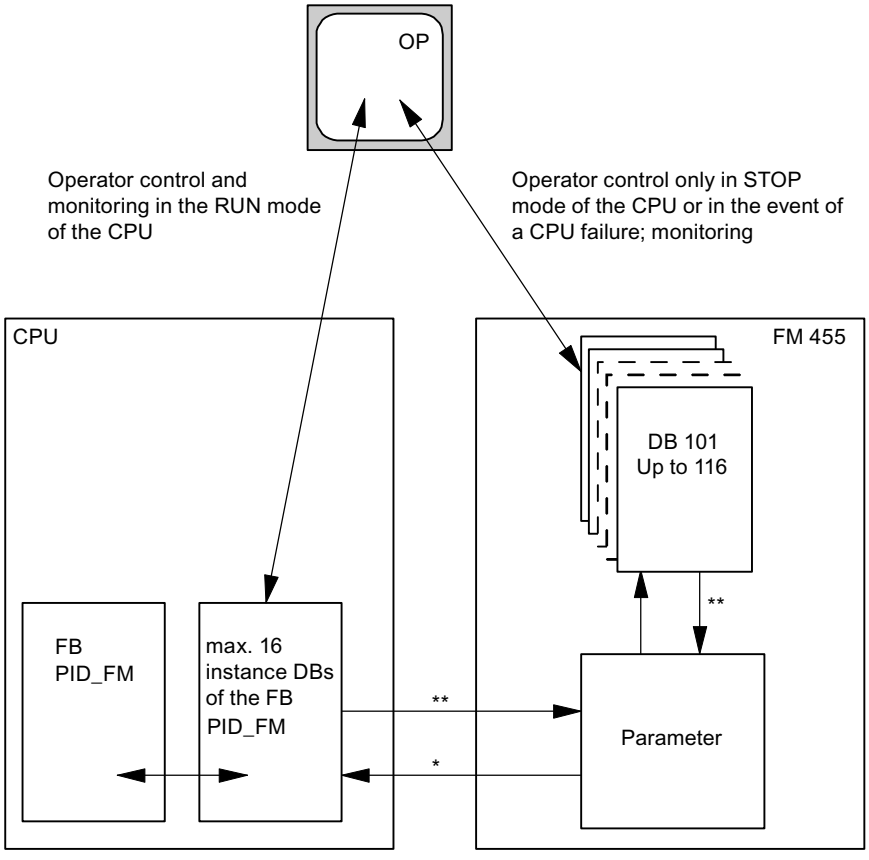

* controlled by the READ_VAR parameter of the instance DB

** controlled by the LOAD_OP and LOAD_PAR parameters

Input parameters DB 101 to 116 (S7-300, S7-400)

Note

The EEPROM of the module could become damaged in the case of too frequent write operations. To prevent this, after the description of the EEPROM, the module delays another description by 30 minutes.

Table 4-298

Parameters	Address	Data type	Default	Description
[1] Control parameters; [2] Operating parameters				
SP_HLM[1]	0.0	REAL	100.0	The setpoint is always limited to a high and a low limit. The input "Setpoint high limit" indicates the high limit. SP_HLM > SP_LLM
SP_LLM[1]	4.0	REAL	0.0	The setpoint is always limited to a high and a low limit. The input "Setpoint low limit" specifies the low limit. SP_LLM < SP_HLM
H_ALM[1]	8.0	REAL	100.0	You can assign parameters for four limits for monitoring the process value or the error signal. The input "High limit interrupt" indicates the highest limit. H_ALM > H_WRN
H_WRN[1]	12.0	REAL	90.0	You can assign parameters for four limits for monitoring the process value or the error signal. The input “High limit warning” specifies the second to highest limit. Values from H_ALM to L_WRN are permitted.
L_WRN[1]	16.0	REAL	10.0	You can assign parameters for four limits for monitoring the process value or the error signal. The input “Low limit warning” specifies the second to highest limit. Values from H_WRN to L_ALM are permitted.
L_ALM[1]	20.0	REAL	0.0	You can assign parameters for four limits for monitoring the process value or the error signal. The input "Low limit alarm" specifies the lowest limit. L_ALM < L_WRN
HYS[1]	24.0	REAL	1.0	To prevent flickering of the monitoring displays, a hysteresis can be configured at the "Hysteresis" input. HYS ≥ 0.0
DEADB_W[1]	28.0	REAL	0.0	A dead band is applied to the error signal. The "Dead band width" input determines the size of the dead band. DEADB_W ≥ 0.0
GAIN[1]	32.0	REAL	1.0	The "Proportional gain" input specifies controller amplification.
TI[1]	36.0	REAL	3000.0	The "Integration time" input determines the time response of the integral action. If TI = 0, the integral action is disabled. TI = 0.0 or TI ≥ 0.5
TD[1]	40.0	REAL	0.0	The "Derivative-action time" input determines the time response of the differentiator. If TD = 0, the derivative action is disabled TD = 0.0 or TD ≥ 1.0

Parameters	Address	Data type	Default	Description
TM_LAG[1)]	44.0	REAL	5.0	The algorithm of the derivative action contains a delay for which parameters can be assigned at the input "Time lag of the derivative action". TM_LAG ≥ 0.5
LMN_SAFE[1)]	48.0	REAL	0.0	For the manipulated value, a safety value can be configured at the "Safety manipulated value" input. Values from -100 to 100 % are permitted.
LMN_HLM[1)]	52.0	REAL	100.0	The manipulated value is always limited to a high and a low limit. The input "Manipulated value high limit" specifies the high limit. (Does not apply to step controllers without analog position feedback) Values from LMN_LLM to 100 % are permitted.
LMN_LLM[1)]	56.0	REAL	0.0	The manipulated value is always limited to a high and a low limit. The input "Manipulated value low limit" indicates the low limit. (Does not apply to step controllers without analog position feedback) Values from -100 to LMN_HLM are permitted.
MTR_TM[1)]	60.0	REAL	60.0	The runtime of the control valve from limit to limit is entered at the "Motor actuating time" parameter. (Only in the case of step controllers) MTR_TM ≥ 0.001
PULSE_TM[1)]	64.0	REAL	0.2	A minimum pulse duration can be configured on the "Minimum pulse time" parameter. (Only in the case of step controllers or pulse controllers)
BREAK_TM[1)]	68.0	REAL	0.2	A minimum break time can be set at the "Minimum break time" parameter. (Only in the case of step controllers or pulse controllers)
SP_RE[2)]	72.0	REAL	0.0	An external setpoint is interconnected at input "External setpoint" with the controller. Technical range of values (physic. variable)
LMN_RE[2)]	76.0	REAL	0.0	An external manipulated value is interconnected at the input "External manipulated value" with the controller. Values from -100 to 100 % are permitted.
LMNRSVAL[2)]	80.0	REAL	0.0	The Commissioning dialog provides you with access to the input "Start value of the simulated position feedback". The start value of the simulation is entered at the parameter. (only for step controllers without analog position feedback). Values from -100 to 100 % are permitted.
SAFE_ON[2)]	84.0	BOOL	FALSE	If the "Enable safety position" input is set, a safety value is applied as the manipulated value. Note: The signal processing of maninpulated values via LMNDN_OP, LMNUP_OP and LMNSOPON on a step controller is of higher priority than the safety manipulated value.
LMNTRKON[2)]	84.1	BOOL	FALSE	If input "Follow-up (LMN via AI)" is set, the manipulated value is adjusted to an analog input (AI). (Does not apply to step controllers without analog position feedback)
LMN_RE-ON[2)]	84.2	BOOL	FALSE	If input "Enable external manipulated value" is set, the external manipulated value LMN_RE is set as the manipulated value.

Parameters	Address	Data type	Default	Description
LMNRHSRE[2)]	84.3	BOOL	FALSE	The signal “Control valve at high endstop” is interconnected at the input “High stop signal of position feedback”. LMNRHSRE = TRUE means: The control valve is at high endstop. (Only in the case of step controllers)
LMNRLSRE[2)]	84.4	BOOL	FALSE	The signal “Control valve at low endstop” is interconnected at the input “Low endstop signal of position feedback”. LMNRLSRE = TRUE means: The control valve is at low endstop. (Only in the case of step controllers)
LMNSOPON[2)]	84.5	BOOL	FALSE	If the bit is set at input "Enable manipulated value signal operation", the LMNUP_OP and LMNDN_OP signals are set as the manipulated value signals. (Only in the case of step controllers)
LMNUP_OP[2)]	84.6	BOOL	FALSE	If LMNSOPON is set, the value at input "Manipulated value signal up operation" is set as the manipulated value signal. (Only in the case of step controllers)
LMNDN_OP[2)]	84.7	BOOL	FALSE	If LMNSOPON is set, the value at input "Manipulated value signal down operation" is set as the manipulated value signal. (Only in the case of step controllers)
MONERSEL[1)]	85.0	BOOL	FALSE	The controller has a limit indicator that can be applied either for the process value or the error signal. If the input "Monitoring: Process value = 0, control error = 1" is set, the control error is monitored.
LMNRS_ON[2)]	85.1	BOOL	FALSE	You can simulate position feedback if this is not available. The function is enabled by setting the "Enable simulation of position feedback" input CAUTION: Offset between the simulation value and real position feedback increases over the time. (only for step controllers without analog position feedback).
FUZID_ON[2)]	85.2	BOOL	FALSE	The "Enable fuzzy identification" input is used to enable identification of the fuzzy algorithm.
SPINT_EN[2)]	85.3	BOOL	FALSE	The input "Operator input: External = 0, Internal = 1" determines the input that is transferred as a setpoint to the module. SPINT_EN = TRUE: SP_INT is transferred. SPINT_EN = FALSE: SP_RE is transferred.
P_SEL[1)]	85.4	BOOL	TRUE	In the PID algorithm, the PID actions can be enabled and disabled separately. The proportional action is enabled if the "Enable P-action" input is set
PFDB_SEL[1)]	85.5	BOOL	FALSE	In the PID algorithm, the P and D actions can be positioned in the feedback. If input "Enable P-action in feedback" is set, the proportional action is located in the feedback loop.
D_EL_SEL[1)]	86.0	INT	0	The D-action element in the PID algorithm can be positioned at a separate input. This is selected at the input "D-action element input". • 0: Error signal • 1 to 16:: Analog input 1 to 16 • 17: Negative process value

Output parameters DB 101 to 116 (S7-300, S7-400)

Table 4-299

Parameters	Address	Data type	Default	Description
SP	94.0	REAL	0.0	The setpoint that is currently in effect is available at the "Setpoint" output.
PV	98.0	REAL	0.0	The process value that becomes effective is issued on the "Process value" output.
ER	102.0	REAL	0.0	The control deviation that becomes effective is issued on the "Control deviation" output.
DISV	106.0	REAL	0.0	The interference that becomes effective is issued on the "Interference" output. Values from -100 to 100 % are permitted.
LMN	110.0	REAL	0.0	The manipulated value that becomes effective is issued on the "Manipulated value" output. Step controllers without analog position feedback output the unlimited P- + D-action at the LMN parameter. Values from -100 to 100 % are permitted.
LMN_A	114.0	REAL	0.0	At the output "Manipulated value A of the split range function / position feedback", continuous controllers display the manipulated value A and step controllers with analog position feedback display the position feedback. Step controllers without analog position feedback display the simulated position feedback. Values from -100 to 100 % are permitted.
LMN_B	118.0	REAL	0.0	Continuous controllers display the manipulated value B of the split-range function at the output “Manipulated value B of split-range function”. Values from -100 to 100 % are permitted.
QH_ALM	122.0	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_ALM limit is reported at the output "High limit alarm triggered".
QH_WRN	122.1	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_WRN limit is reported at the output "High limit warning triggered".
QL_WRN	122.2	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_WRN limit is reported at output "Low limit warning triggered".
QL_ALM	122.3	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_ALM limit is reported at output "Low limit alarm triggered".
QLMN_HLM	122.4	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the high limit is reported at output "High limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QLMN_LLM	122.5	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the low limit is reported at output "Low limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QSPINTON	122.6	BOOL	FALSE	The output "Internal setpoint on" indicates that SP_INT was transferred to the module.
QPARA_F	123.0	BOOL	FALSE	The module checks the validity of the parameters. A parameter assignment error is displayed at the "Parameter assignment error" output. These parameter assignment errors can also be read out in the module's "Online & Diagnostics" dialog.
QCH_F	123.1	BOOL	FALSE	The "Channel error" output is set if the controller channel cannot provide valid results. A channel error (for example, wire break) is also set if QPARA_F = 1 or QMOD_F = 1. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.

Parameters	Address	Data type	Default	Description
QUPRLM	123.2	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of positive setpoint slew rate triggered" is set, the positive setpoint slew rate is limited.
QDNRLM	123.3	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of negative setpoint slew rate triggered" is set, the setpoint fall is limited.
QSP_HLM	123.4	BOOL	FALSE	The setpoint is always limited to a high and a low limit. The output "High limit of setpoint value triggered" indicates that the high limit has been exceeded.
QSP_LLM	123.5	BOOL	FALSE	The setpoint is always limited to a high and a low limit. Violation of the low limit is reported at output "Setpoint low limit triggered".
QSPOPON	123.6	BOOL	FALSE	The output "Setpoint operation on" shows if the setpoint is being operated via the "Commissioning" dialog. If the bit is set, the value SP_OP is applied as setpoint.
QLMNSAFE	123.7	BOOL	FALSE	If the output "Safety mode" is set, the safety manipulated value is output as the manipulated value.
QLMNO-PON	124.0	BOOL	FALSE	The output "Manipulated value operation on" shows if the manipulated value is being operated via the "Commissioning" dialog. If the bit is set, the value LMN_OP is applied as manipulated value.
QLMNTRK	124.1	BOOL	FALSE	The output "Follow-up mode" indicates if the manipulated value is being adjusted via an analog input.
QLMN_RE	124.2	BOOL	FALSE	The output "Manual = 1; Automatic = 0" indicates whether or not the manipulated value is set on the external manipulated value LMN_RE (manual = 1).
QLMNR_HS	124.3	BOOL	FALSE	The output "High endstop signal of position feedback" indicates if the control valve is at its high endstop. QLMNR_HS = TRUE means: The control valve is at high endstop. (For step controllers only)
QLMNR_LS	124.4	BOOL	FALSE	The output "Low endstop signal of position feedback" indicates whether the control valve is at its low endstop. QLMNR_LS = TRUE means: The control valve is at its low endstop. (For step controllers only)
QLMNR_ON	124.5	BOOL	FALSE	Output "Position feedback enabled" indicates the set operating mode "Step controller with position feedback" or "Step controller without position feedback".
QFUZZY	124.6	BOOL	FALSE	If the output "PID algorithm = 0; fuzzy = 1", the controller is working with the fuzzy algorithm.
QSPLEPV	124.7	BOOL	FALSE	The output "Display of FUZZY controller: Setpoint < process value" is set when the fuzzy controller is switched on, if the setpoint is less than the effective process value.
QSPR	125.0	BOOL	FALSE	If the output "Split-range operation" is set, the continuous controller is operating in split-range mode.
QLMNUP	125.1	BOOL	FALSE	This is the "Manipulated value signal up" output. (Only in the case of step controllers or pulse controllers)
QLMNDN	125.2	BOOL	FALSE	This is the "Manipulated value signal down" output. (Only in the case of step controllers or pulse controllers)
QBACKUP	125.4	BOOL	FALSE	• 0= No backup state (CPU in RUN) • 1= Backup state (CPU in STOP or has failed)
QID	125.5	BOOL	FALSE	QID = TRUE indicates that an identification is running (not that it is enabled). On completion of the identification, the identification result can be read out from the IDSTATUS parameter of the instruction CH_DIAG_455.

Parameters	Address	Data type	Default	Description
QMAN_FC	125.6	BOOL	FALSE	The controller is a master controller that is adjusted by the manual mode of a slave controller to its process value or its I-action is stopped, because the setpoint or manipulated value of the slave controller lies in the limit.
RET_VALU	126.0	WORD	0	RET_VALU contains the return value RET_VAL of the RD_REC and WR_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.

See also

RD_REC: Read data record from I/O (Page 5256)

WR_REC: Write data record to I/O (Page 5260)

In/out parameters DB 101 to 116 (S7-300, S7-400)

Table 4-300

Parameters	Address	Data type	Default	Description
1) Operating parameters				
SP_INT	128.0	REAL	0.0	The in/out parameter "Internal setpoint" is used to specify a setpoint by means of operating and monitoring functions.
SP_OP[1)]	132.0	REAL	0.0	The "Commissioning" dialog can access the in/out parameter "Setpoint operation". If the SP_OP_ON bit is set, the "Setpoint operation" value is applied as the setpoint.
LMN_OP[1)]	136.0	REAL	0.0	The "Commissioning" dialog can access the in/out parameter "Manipulated value operation". If the LMNOP_ON bit is set, the "Manipulated value operation" value is applied as the manipulated value. Values from -100 to 100 % are permitted.
SP_OP_ON [1)]	140.0	BOOL	FALSE	The "Commissioning" dialog can access the in/out parameter "Enable setpoint operation". If the bit is set, the value SP_OP is applied as setpoint.
LMNOP_ON [1)]	140.1	BOOL	FALSE	The "Commissioning" dialog can access the in/out parameter "Enable manipulated value operation". If the bit is set, the value LMN_OP is applied as manipulated value.
LOAD_PAR	140.2	BOOL	FALSE	If the in/out parameter "Load control parameters to FM 455" is set, the control parameters are loaded into the module and the in/out parameter is reset.
LOAD_OP	140.3	BOOL	FALSE	If the in/out parameter "Load operating parameters to FM 455" is set, the operating parameters are loaded into the module and the in/out parameter is reset.

Note

The EEPROM of the module could become damaged by too frequent write operations. To prevent this, the module delays another write process by 30 minutes after writing of the EEPROM.

4.2.4.4 300C functions (S7-300)

ANALOG: Position with analog output (S7-300)

Description

You use the ANALOG instruction to control the positioning functions of the user program.

A fixed assigned analog output controls the power section with a voltage (voltage signal) of ±10 V or with a current (current signal) of ±20 mA.

- After the acceleration phase (RAM_UP), the drive approaches the destination at the speed ($V_{setpoint}$).
- At the braking point that is calculated by the CPU, the deceleration (RAMP_DN) to the changeover point is initialized.
- Once the changeover point has been reached, the run will continue at creep speed (V_{creep}).
- The drive is switched off at the cut-off point.
- The changeover point and the cut-off point are determined for every target that will be approached in the values you have specified for the changeover difference and cut-off difference parameters. The changeover difference and cut-off difference can be determined differently for forward motion (in plus direction) and for reverse motion (in minus direction).
- Traverse is complete (WORKING = FALSE) when the cut-off point is reached. A new traverse can then be started after this time.
- The specified target has been reached (POS_RCD = TRUE) when the actual position value has reached the target area. If the actual position value again deviates from the target area without a new traverse having been started, the signal "position reached" is not reset.

If the change-over difference is less than the cut-off difference, the drive is slowed linearly to speed setpoint 0 starting at the braking point.

Basic parameters

The parameters of the instruction that are identical for all modes are described here. The parameters specific to the operating mode are described for the individual operating modes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter

The following table shows the parameters of the instruction ANALOG:

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
LADDR	Input	WORD	0	CPU-specific	W#16#0310	The submodule I/O address, which you have specified during hardware configuration. If the I and O addresses are not equal, the lesser of the two addresses must be specified.
CHANNEL	Input	INT	2	0	0	Channel number
STOP	Input	BOOL	4.4	TRUE/FALSE	FALSE	Stop traverse Use STOP = TRUE to stop/interrupt the traverse prematurely.
ERR_A	Input	BOOL	4.5	TRUE/FALSE	FALSE	Group acknowledgement of external error External errors are acknowledged with ERR_A = TRUE.
SPEED	Input	DINT	12	Creep speed to 1 000 000 pulses/s No higher than the maximum speed that will be assigned in the parameter	1000	The axis will be accelerated to the speed "$V_{setpoint}$". It is not possible to change the speed during traverse.
WORKING	Output	BOOL	16.0	TRUE/FALSE	FALSE	Traverse in process
ACT_POS	Output	DINT	18	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	22	0, 1, 3, 4, 5	0	Active/set operating mode
ERR	Output	WORD	24	Each bit 0 or 1	0	External error: Bit 2: Zero mark monitoring Bit 11: Traverse range monitoring (always 1) Bit 12: Work range monitoring Bit 13: Process value monitoring Bit 14: Destination run-up monitoring Bit 15: Destination area monitoring Remaining bits reserved

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
ST_ENBLD	Output	BOOL	26.0	TRUE/FALSE	TRUE	The CPU sets start enable if all the following conditions apply: • STOP is not active (STOP = FALSE) • No external error pending (ERR = 0) • Drive enable is set (DRV_EN = TRUE) • Positioning not in process (WORKING = FALSE)
ERROR	Output	BOOL	26.1	TRUE/FALSE	FALSE	Error when starting /resuming a traverse
STATUS	Output	WORD	28.0	W#16#0000 to W#16#FFFF	W#16#0000	Error number

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
ACCEL	STATIC	DINT	30	1 to 100 000 pulses/s^2	100	Acceleration Change during traverse not possible.
DECEL	STATIC	DINT	34	1 to 100 000 pulses/s^2	100	Deceleration Change during traverse not possible.
CHGDIFF_P	STATIC	DINT	38	0 to $+10^8$ pulses	1000	Changeover difference plus: "Changeover difference plus" defines the changeover point from which the drive continues forward traverse in creep speed.
CUTOFF-DIFF_P	STATIC	DINT	42	0 to $+10^8$ pulses	100	Cut-off difference plus: "Cut-off difference plus" defines the cut-off point at which the drive is shut off from creep speed in the forward direction.
CHGDIFF_M	STATIC	DINT	46	0 to $+10^8$ pulses	1000	Changeover difference minus: "Changeover difference minus" defines the changeover point from which the drive continues in reverse motion at creep speed.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
CUTOFF-DIFF_M	STATIC	DINT	50	0 to $+10^8$ pulses	100	Cut-off difference minus: "Cut-off difference minus" defines the point at which the drive is switched out of creep speed in the reverse direction.
PARA	STATIC	BOOL	54.0	TRUE/FALSE	FALSE	Parameters have been assigned to the axis
DIR	STATIC	BOOL	54.1	TRUE/FALSE	FALSE	Actual/last direction of movement FALSE = forward (in plus direction) TRUE = reverse (in minus direction)
CUTOFF	STATIC	BOOL	54.2	TRUE/FALSE	FALSE	Drive in cut-off area (from the cut-off point to the start of the next run)
CHGOVER	STATIC	BOOL	54.3	TRUE/FALSE	FALSE	Drive in changeover range (from reaching creep speed to start of next traverse)
RAMP_DN	STATIC	BOOL	54.4	TRUE/FALSE	FALSE	The drive is decelerated (from braking point to changeover point)
RAMP_UP	STATIC	BOOL	54.5	TRUE/FALSE	FALSE	The drive is accelerated (from start until it reaches the speed SPEED ($V_{setpoint}$))
DIST_TO_GO	STATIC	DINT	56	$-5x10^8$ to $+5x10^8$ pulses	0	Actual residual distance
LAST_TRG	STATIC	DINT	60	$-5x10^8$ to $+5x10^8$ pulses	0	Last/current target • Absolute incremental mode: At start of traverse LST_TRG = current absolute target (TARGET). • Relative incremental mode: At start of traverse LST_TRG = LAST_TRG is the specified +/- distance of the previous traverse (TARGET).

Parameters for "jog" mode

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Jog in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Jog in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 1 = jog
WORKING	Output	BOOL	16.0	TRUE/FALSE	FALSE	Traverse in process
ACT_POS	Output	DINT	18	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	22	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "search for reference" mode

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Search for reference in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Search for reference in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 3 = "search for reference"
WORKING	Output	BOOL	16.0	TRUE/FALSE	FALSE	Traverse in process
SYNC	Output	BOOL	16.3	TRUE/FALSE	FALSE	SYNC = TRUE: Axis is synchronized
ACT_POS	Output	DINT	18	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	22	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "relative incremental mode"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Traverse in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Traverse in minus direction (positive edge)

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 4 = relative incremental mode
TARGET	Input	DINT	8	0 to 10^9 pulses	1000	Distance in pulses (only positive values allowed)
WORKING	Output	BOOL	16.0	TRUE/FALSE	FALSE	Traverse in process
POS_RCD	Output	BOOL	16.1	TRUE/FALSE	FALSE	Position reached
ACT_POS	Output	DINT	18	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	22	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "absolute incremental mode"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
START	Input	BOOL	4.1	TRUE/FALSE	FALSE	Start traverse (positive edge)
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Traverse in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Traverse in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 5 = absolute incremental mode
TARGET	Input	DINT	8	Linear axis : $-5x10^8$ to $+5x10^8$ Rotary axis: 0 to rotary axis end -1	1000	Target in pulses
WORKING	Output	BOOL	16.0	TRUE/FALSE	FALSE	Traverse in process
POS_RCD	Output	BOOL	16.1	TRUE/FALSE	FALSE	Position reached
ACT_POS	Output	DINT	18	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	22	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for the job "set reference point"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
SYNC	Output	BOOL	16.3	TRUE/FALSE	FALSE	Axis is synchronized

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_REQ	STATIC	BOOL	76.0	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_DONE	STATIC	BOOL	76.1	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	STATIC	BOOL	76.2	TRUE/FALSE	FALSE	Faulty job
JOB_ID	STATIC	INT	78	1, 2	0	Job, 1 = "set reference point"
JOB_STAT	STATIC	WORD	80	W#16#0000 to W#16#FFFF	W#16#0000	Job error number
JOB_VAL	STATIC	DINT	82	$5x10^8$ to $+5x10^8$ pulses	0	Job parameter for reference point coordinates

Parameters for the job "delete residual distance"

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_REQ	STATIC	BOOL	76.0	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_DONE	STATIC	BOOL	76.1	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	STATIC	BOOL	76.2	TRUE/FALSE	FALSE	Faulty job
JOB_ID	STATIC	INT	78	1, 2	0	Job, 2 = "delete residual distance"
JOB_STAT	STATIC	WORD	80	W#16#0000 to W#16#FFFF	W#16#0000	Job error number
JOB_VAL	STATIC	DINT	82	-	0	Any setting

Parameters for the "length measurement" function

This operation is started at the positive edge on the digital input. There are no specific input parameters.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
MSR_DONE	OUT	BOOL	16.2	TRUE/FALSE	FALSE	Length measurement complete

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
BEG_VAL	STATIC	DINT	64	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value start of length measurement
END_VAL	STATIC	DINT	68	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value length measurement end
LEN_VAL	STATIC	DINT	72	0 to 10^9 pulses	0	Measured length

Parameter ERROR and STATUS

Operating mode error (ERROR = TRUE)

The output parameter ERROR is set to TRUE if an error is detected. The parameter STATUS shows the cause of the error.

Event class error code	Description
W#16#2002	Incorrect instruction, use ANALOG
W#16#2004	Incorrect channel number (CHANNEL). Set channel number "0".
W#16#3001	Traverse job rejected because of job error in the same call. Correct the respective JOB parameters.
W#16#3002	A change of Output _IN is not permitted while the drive is still running. Wait for the end of the current positioning.
W#16#3003	Unknown operating mode (MODE_IN). Permitted are 1 (jog), 3 (search for reference), 4 (relative incremental mode) and 5 (absolute incremental mode).
W#16#3004	Only one start request is allowed at a time. Valid start requests are DIR_P, DIR_M or START.
W#16#3005	START is only permitted in operating mode "absolute incremental mode”. Start the traverse with DIR_P or DIR_M.
W#16#3006	DIR_P and DIR_M are not permitted for linear axis and "absolute incremental mode". Start the traverse with START.
W#16#3007	Axis not synchronized. "Absolute incremental mode" is only possible with a synchronized axis.
W#16#3008	Leave work area. Return traverse to work area is only allowed in jog mode.
W#16#3101	Start is not enabled because parameters have not been assigned for the axis. Assign parameters for the "positioning" submodule via HW Config.
W#16#3102	Start not enabled because drive enable is not set Set the "drive enable" on the instruction (DRV_EN=TRUE).
W#16#3103	Start not enabled because STOP is set. Delete STOP on the instruction (STOP=FALSE)
W#16#3104	Start is not enabled because the axis is currently positioning (WORKING=TRUE). Wait for the end of the current positioning.
W#16#3105	Start not enabled because at least one error that is pending has not been acknowledged. First, eliminate and clear all external errors and then restart traverse.

Event class error code	Description
W#16#3202	Wrong SPEED default. The speed default is out of the permitted creep speed range of up to 1000000 pulses/s, though not higher than the parameter-assigned maximum speed.
W#16#3203	The acceleration default in ACCEL is outside the permitted range of 1 to 100,000 pulses/s^2.
W#16#3204	The deceleration default in DECEL is outside the permitted range of 1 to 100,000 pulses/s^2.
W#16#3206	The speed default in SPEED must be greater than or equal to the assigned referencing frequency parameter.
W#16#3301	Changeover/cut-off difference is too high. Set a maximum changeover/cut-off difference of 10^8.
W#16#3304	Cut-off difference too low. The cut-off difference must be at least half the value of the target range.
W#16#3305	Changeover difference too low. The changeover difference must be at least half the value of the target range.
W#16#3401	Target default outside of work range. For a linear axis and absolute incremental mode the target default must be within the range of the software limit switches (inclusive).
W#16#3402	Wrong target default. For a rotary axis the target default must be greater than 0 and less than the rotary axis end value.
W#16#3403	Wrong distance specification. The traverse distance for relative incremental mode must be positive.
W#16#3404	Wrong distance specification. The resulting absolute target coordinate must be greater than $-5x10^8$.
W#16#3405	Wrong distance specification. The resulting absolute target coordinate must be greater than $-5x10^8$.
W#16#3406	Wrong distance specification. The resulting absolute target coordinate must be within the work range (+/- half of the target range).
W#16#3501	Travel range too great. Target coordinate + actual residual distance must be greater than or equal to $-5x10^8$.
W#16#3502	Travel range too great. Target coordinate + actual residual distance must be less than or equal to $5x10^8$.
W#16#3503	Traverse distance insufficient. Traverse distance in plus direction must be greater than the specified cut-off difference for direction plus.
W#16#3504	Traverse distance insufficient. The traverse distance in minus direction must be greater than the specified cut-off difference for the minus direction
W#16#3505	Traverse distance insufficient or the limit switch is already overtraveled in plus direction. The last approachable target in plus direction (work range or traverse range limit) is too close to the current position.
W#16#3506	Traverse distance insufficient, or the limit switch is already overtraveled in the minus direction. The last approachable target in the minus direction (work range or traverse range limit) is too close to the current position.

Job error (JOB_ERR = TRUE)

The output parameter JOB_ERR is set to TRUE if an error is detected. The parameter JOB_STAT shows the cause of the error.

Event class error code	Description
W#16#4001	Parameters not assigned for axis. Assign parameters for the "positioning" submodule via HW Config.
W#16#4002	Job not possible because positioning is in progress. Wait until WORKING = FALSE, then repeat the job.
W#16#4004	Unknown job. Check the job ID and then repeat the job.
W#16#4101	For a linear axis the reference point coordinate should not be outside of the work area limits.
W#16#4102	For a linear axis the specified reference point coordinate + actual residual distance must be greater than or equal to $-5x10^8$.
W#16#4103	For a linear axis the specified reference point coordinate + actual residual distance must be greater than or equal to $5x10^8$.
W#16#4104	For a linear axis the specified reference point coordinate + actual difference to the starting point of traverse must be greater than or equal to $-5x10^8$.
W#16#4105	For a linear axis the specified reference point coordinate + actual difference to the starting point of traverse must be less than or equal to $5x10^8$.
W#16#4106	For a rotary axis the reference point coordinate should not be lower than 0 and greater than or equal to the rotary axis end.

External error (ERR)

The technology monitors traverse, traverse range, and the connected peripheral devices. The prerequisite is that you have switched on monitoring in the "Drive", "Axis" and "Encoder" parameter screen forms beforehand.

An external fault is reported when the monitoring unit is tripped. External errors can occur at anytime regardless of the started functions. You must always acknowledge external errors with ERR_A = TRUE.

External errors are indicated by a set bit in parameter ERR (WORD).

Monitoring	Error code	Bit in ERR-WORD
Missing pulse (zero mark)	W#16#0004	2
Traverse range	W#16#0800	11
Work range	W#16#1000	12
Process value	W#16#2000	13
Destination run-up	W#16#4000	14
Target range	W#16#8000	15

System error

A system error is indicated with BIE (binary result) = FALSE. A system error is triggered by errors while reading/writing the instance DB or by multiple calls of the instruction.

DIGITAL: Position with digital output (S7-300)

Description

You use the DIGITAL instruction to control the positioning functions of the user program.

Four 24 V digital outputs are permanently assigned to the drive and control the power stage (you can find additional information in the documentation for the hardware used). Depending on the trigger mode, the digital outputs control the direction and speed levels (rapid/creep speed).

The distance is measured via an asymmetrical 24 V incremental transducer with two out-of-phase signals at 90 degrees.

- First, the target is approached with the speed (V_{rapid}).
- At the changeover point, the system switches to creep speed (V_{creep}).
- The drive is switched off at the cut-off point.
- You can set the changeover point and cut-off point with the parameter values CHGDIFF_P/M and CUTOFFDIFF_P/M. The changeover difference and cut-off difference can be defined differently for forward motion (in plus direction) and for reverse motion (in minus direction).
- Traverse is complete (WORKING = FALSE) when the cut-off point is reached. A new traverse can then be started after this time.
- The specified target has been reached (POS_RCD = TRUE) when the actual position value has reached the target area. If the actual position value again deviates from the target area without a new traverse having been started, the signal "position reached" is not reset.

Basic parameters:

The parameters of the instruction that are identical for all modes are described here. The parameters specific to the operating mode are described for the individual operating modes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
LADDR	Input	WORD	0	CPU-specific	W#16#0310	The submodule I/O address, which you have specified during hardware configuration. If the I and O addresses are not equal, the lesser of the two addresses must be specified.
CHANNEL	Input	INT	2	0	0	Channel number
STOP	Input	BOOL	4.4	TRUE/FALSE	FALSE	Stop traverse Use STOP = TRUE to stop/interrupt the traverse prematurely.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
ERR_A	Input	BOOL	4.5	TRUE/FALSE	FALSE	Group acknowledgement of external error External errors are acknowledged with ERR_A = TRUE.
SPEED	Input	BOOL	12.0	TRUE/FALSE	FALSE	Two speed stages for rapid traverse/creep speed TRUE = rapid traverse FALSE= creep speed
WORKING	Output	BOOL	14.0	TRUE/FALSE	FALSE	Traverse in process
ACT_POS	Output	DINT	16	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	20	0, 1, 3, 4, 5	0	Active/set operating mode
ERR	Output	WORD	22	Each bit 0 or 1	0	External error • Bit2: Zero mark monitoring • Bit11: Traverse range monitoring (always 1) • Bit12: Work range monitoring • Bit13: Process value monitoring • Bit14: Target area monitoring • Bit15: Target area monitoring Remaining bits reserved
ST_ENBLD	Output	BOOL	24.0	TRUE/FALSE	TRUE	The CPU sets start enable if all the following conditions apply: • STOP is not active (STOP = FALSE) • No external error pending (ERR = 0) • Drive enable is set (DRV_EN = TRUE) • Positioning not in process (WORKING = FALSE)
ERROR	Output	BOOL	24.1	TRUE/FALSE	FALSE	Error when starting /resuming a traverse
STATUS	Output	WORD	26.0	W#16#0000 to W#16#FFFF	W#16#0000	Error number

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
CHGDIFF_P	STATIC	DINT	28	0 to $+10^8$pulses	1000	Changeover difference plus: "Changeover difference plus" defines the point at which the drive continues its forward traverse with creep speed.
CUTOFF-DIFF_P	STATIC	DINT	32	0 to $+10^8$ pulses	100	Cut-off difference plus: "Cut-off difference plus" defines the cut-off point at which the drive is shut off from creep speed in the forward direction.
CHGDIFF_M	STATIC	DINT	36	0 to $+10^8$ pulses	1000	Changeover difference minus: "Changeover difference minus" defines the point at which the drive changes over from rapid traverse to creep speed in the reverse direction.
CUTOFF-DIFF_M	STATIC	DINT	40	0 to $+10^8$ pulses	100	Cut-off difference minus: "Cut-off difference minus" defines the point at which the drive is switched out of creep speed in the reverse direction.
PARA	STATIC	BOOL	44.0	TRUE/FALSE	FALSE	Parameters have been assigned to the axis
DIR	STATIC	BOOL	44.1	TRUE/FALSE	FALSE	Actual/last direction of movement FALSE = forward (in plus direction) TRUE = reverse (in minus direction)
CUTOFF	STATIC	BOOL	44.2	TRUE/FALSE	FALSE	Drive in cut-off area (from the cut-off point to the start of the next run)
CHGOVER	STATIC	BOOL	44.3	TRUE/FALSE	FALSE	Drive in changeover range (from reaching creep speed to start of the next traverse)

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DIST_TO_GO	STATIC	DINT	46	$-5x10^8$ to $+5x10^8$ pulses	0	Actual residual distance
LAST_TRG	STATIC	DINT	50	$-5x10^8$ to $+5x10^8$ pulses	0	Last/current target • Absolute incremental mode: At start of traverse LST_TRG = current absolute target (TARGET). • Relative incremental mode: At start of traverse LST_TRG = LAST_TRG is the specified +/- distance of the previous traverse (TARGET).

Parameters for "jog" mode

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Jog in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Jog in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 1 = jog
WORKING	Output	BOOL	14.0	TRUE/FALSE	FALSE	Traverse in process
ACT_POS	Output	DINT	16	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	20	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "search for reference" mode

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Search for reference in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Search for reference in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 3 = "search for reference"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
WORKING	Output	BOOL	14.0	TRUE/FALSE	FALSE	Traverse in process
SYNC	Output	BOOL	14.3	TRUE/FALSE	FALSE	SYNC = TRUE: Axis is synchronized
ACT_POS	Output	DINT	16	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	20	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "relative incremental mode"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Traverse in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Traverse in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 4 = relative incremental mode
TARGET	Input	DINT	8	0 to 10^9 Pulse	1000	Distance in pulses (only positive values allowed)
WORKING	Output	BOOL	14.0	TRUE/FALSE	FALSE	Traverse in process
POS_RCD	Output	BOOL	14.1	TRUE/FALSE	FALSE	Position reached
ACT_POS	Output	DINT	16	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	20	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "absolute incremental mode"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
START	Input	BOOL	4.1	TRUE/FALSE	FALSE	Start traverse (positive edge)
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Traverse in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Traverse in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 5 = absolute incremental mode

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
TARGET	Input	DINT	8	Linear axis : $-5x10^8$ to $+5x10^8$ Rotary axis: 0 to rotary axis end -1	1000	Target in pulses
WORKING	Output	BOOL	14.0	TRUE/FALSE	FALSE	Traverse in process
POS_RCD	Output	BOOL	14.1	TRUE/FALSE	FALSE	Position reached
ACT_POS	Output	DINT	16	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	20	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for the job "set reference point"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
SYNC	Output	BOOL	14.3	TRUE/FALSE	FALSE	Axis is synchronized

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_REQ	STATIC	BOOL	66.0	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_DONE	STATIC	BOOL	66.1	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	STATIC	BOOL	66.2	TRUE/FALSE	FALSE	Faulty job
JOB_ID	STATIC	INT	68	1, 2	0	Job, 1 = "set reference point"
JOB_STAT	STATIC	WORD	70	W#16#0000 to W#16#FFFF	W#16#0000	Job error number
JOB_VAL	STATIC	DINT	72	$-5x10^8$ to $+5x10^8$ pulses	0	Job parameter for reference point coordinates

Parameters for the job "delete residual distance"

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_REQ	STATIC	BOOL	66.0	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_DONE	STATIC	BOOL	66.1	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	STATIC	BOOL	66.2	TRUE/FALSE	FALSE	Faulty job

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_ID	STATIC	INT	68	1, 2	0	Job, 2 = "delete residual distance"
JOB_STAT	STATIC	WORD	70	0 to FFFF hex	0	Job error number
JOB_VAL	STATIC	DINT	72	-	0	None

Parameters for the "length measurement" function

This operation is started at the positive edge on the digital input. There are no specific input parameters.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
MSR_DONE	Output	BOOL	14.2	TRUE/FALSE	FALSE	Length measurement complete

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
BEG_VAL	STATIC	DINT	54	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value start of length measurement
END_VAL	STATIC	DINT	58	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value length measurement end
LEN_VAL	STATIC	DINT	62	0 to 10^9pulses	0	Measured length

Parameter ERROR and STATUS

The output parameter ERROR is set to TRUE if an error is detected. The parameter STATUS shows the cause of the error.

STATUS Error code	Description
W#16#2001	Incorrect instruction, use DIGITAL instruction.
W#16#2004	Incorrect channel number (CHANNEL). Set channel number "0".
W#16#3001	Traverse job rejected because of job error in the same call. Correct the respective JOB parameters.
W#16#3002	A change of MODE_IN is not permitted while the drive is still running. Wait for the end of the current positioning.
W#16#3003	Unknown operating mode (MODE_IN). Permitted are 1 (jog), 3 (search for reference), 4 (relative incremental mode) and 5 (absolute incremental mode).
W#16#3004	Only one start request is allowed at a time. Valid start requests are DIR_P, DIR_M or START.
W#16#3005	START is only permitted in operating mode "absolute incremental mode”. Start the traverse with DIR_P or DIR_M.

STATUS Error code	Description
W#16#3006	DIR_P and DIR_M are not permitted for linear axis and "absolute incremental mode". Start the traverse with START.
W#16#3007	Axis not synchronized. "Absolute incremental mode" is only possible with a synchronized axis.
W#16#3008	Leave work area. Return traverse to work area is only allowed in jog mode.
W#16#3101	Start is not enabled because parameters have not been assigned for the axis. Assign parameters for the "positioning" submodule via the hardware configuration.
W#16#3102	Start not enabled because drive enable is not set Set the "drive enable" on the instruction (DRV_EN=TRUE).
W#16#3103	Start not enabled because STOP is set. Delete STOP on the instruction (STOP = FALSE).
W#16#3104	Start is not enabled because the axis is currently positioning (WORKING=TRUE). Wait for the end of the current positioning.
W#16#3105	Start not enabled because at least one error that is pending has not been acknowledged. First, eliminate and clear all external errors and then restart traverse.
W#16#3201	Wrong SPEED default. For positioning with digital outputs only "creep speed" (0) and "rapid traverse" (1) are allowed.
W#16#3301	Changeover/cut-off difference is too high. Set a maximum changeover/cut-off difference of 10^8.
W#16#3303	Changeover difference too low. The changeover difference must be greater than or equal to the cut-off difference.
W#16#3304	Cut-off difference too low. The cut-off difference must be at least half the value of the target range.
W#16#3401	Target default outside of work range. For a linear axis and absolute incremental mode the target default must be within the range of the software limit switches (inclusive).
W#16#3402	Wrong target default. For a rotary axis the target default must be greater than "0" and less than the rotary axis end value.
W#16#3403	Wrong distance specification. The traverse distance for relative incremental mode must be positive.
W#16#3404	Wrong distance specification. The resulting absolute target coordinate must be greater than $-5x10^8$.
W#16#3405	Wrong distance specification. The resulting absolute target coordinate must be greater than $-5x10^8$.
W#16#3406	Wrong distance specification. The resulting absolute target coordinate must be within the work range (+/- half of the target range).
W#16#3501	Traverse distance too great. Target coordinate + actual residual distance must be greater than or equal to $-5x10^8$.
W#16#3502	Traverse distance too great. Target coordinate + actual residual distance must be less than or equal to $5x10^8$.
W#16#3503	Traverse distance insufficient. Traverse distance in plus direction must be greater than the specified cut-off difference for plus direction.
W#16#3504	Traverse distance insufficient. The traverse distance in minus direction must be greater than the specified cut-off difference for the minus direction.

STATUS Error code	Description
W#16#3505	Traverse distance insufficient or the limit switch is already overtraveled in plus direction. The last approachable target in plus direction (work range or traverse distance limit) is too close to the current position.
W#16#3506	Traverse distance insufficient, or the limit switch is already overtraveled in the minus direction. The last approachable target in minus direction (work range or traverse distance limit) is too close to the current position.

Job error (JOB_ERR = TRUE)

The parameter JOB_ERR is set to TRUE if an error is detected. The parameter JOB_STAT shows the cause of the error.

Event class error code	Description
W#16#4001	Parameters not assigned for axis. Assign parameters for the "positioning" submodule via the hardware configuration.
W#16#4002	Job not possible because positioning is in progress. Jobs can only be carried out if positioning is not in progress. Wait until WORKING = FALSE, then repeat the job.
W#16#4004	Unknown job. Check the job ID and then repeat the job.
W#16#4101	For a linear axis the reference point coordinate should not be outside of the work area limits.
W#16#4102	For a linear axis the specified reference point coordinate + actual residual distance must be greater than or equal to $-5x10^8$.
W#16#4103	For a linear axis the specified reference point coordinate + actual residual distance must be greater than or equal to $5x10^8$.
W#16#4104	For a linear axis the specified reference point coordinate + actual difference to the starting point of traverse must be greater than or equal to $-5x10^8$.
W#16#4105	For a linear axis the specified reference point coordinate + actual difference to the starting point of traverse must be less than or equal to $5x10^8$.
W#16#4106	For a rotary axis the reference point coordinate should not be lower than 0 and greater than or equal to the rotary axis end.

External error (ERR)

The technology monitors traverse, traverse range, and the connected peripheral devices. The prerequisite is that you have switched on monitoring in the "Drive", "Axis" and "Encoder" parameter screen forms beforehand.

An external fault is reported when the monitoring unit is tripped. External errors can occur at anytime regardless of the started functions. You must always acknowledge external errors with ERR_A = TRUE.

External errors are indicated by a set bit in parameter ERR (WORD).

Monitoring	Error code	Bit in ERR-WORD
Missing pulse (zero mark)	W#16#0004	2
Traverse range	W#16#0800	11
Work range	W#16#1000	12

Monitoring	Error code	Bit in ERR-WORD
Process value	W#16#2000	13
Destination run-up	W#16#4000	14
Target range	W#16#8000	15

System error

A system error is indicated with BIE (binary result) = FALSE. A system error is triggered by errors while reading/writing the instance DB or by multiple calls of the instruction.

COUNT: Control counter (S7-300)

Description

You use the "COUNT" instruction to control the counter from the user program.

The following functionality is available:

- Starting/stopping the counter via the software gate SW_GATE
- Enabling/controlling the output DO
- Retrieving status bits STS_CMP, STS_OFLW, STS_UFLW and STS_ZP
- Retrieving the current counter value COUNTVAL
- Jobs for reading/writing the internal counter registers
- Retrieving the current period duration TIMEVAL

Parameter

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
LADDR	Input	WORD	0	CPU-specific	W#16#0300	The submodule I/O address, which you have specified during hardware configuration. If the I and O addresses are not equal, the lesser of the two addresses must be specified.
CHANNEL	Input	INT	2	CPU 312C: 0 to 1 CPU 313C: 0 to 2 CPU 314C: 0 to 3	0	Channel number
SW_GATE	Input	BOOL	4.0	TRUE/FALSE	FALSE	Software gate for starting/ stopping the counter
CTRL_DO	Input	BOOL	4.1	TRUE/FALSE	FALSE	Enable output
SET_DO	Input	BOOL	4.2	TRUE/FALSE	FALSE	Control output
JOB_REQ	Input	BOOL	4.3	TRUE/FALSE	FALSE	Job trigger (positive edge)

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_ID	Input	WORD	6	W#16#0000 Job without function W#16#0001 Write count value W#16#0002 Write load value W#16#0004 Write comparison value W#16#0008 Write hysteresis W#16#0010 Write pulse duration W#16#0082 Read load value W#16#0084 Read comparison value W#16#0088 Read hysteresis W#16#0090 Read pulse duration	W#16#0000	Job number
JOB_VAL	Input	DINT	8	-2^{31} to $+2^{31}$ -1	0	Value for write jobs.
STS_GATE	Output	BOOL	12.0	TRUE/FALSE	FALSE	Status of the internal gate
STS_STRT	Output	BOOL	12.1	TRUE/FALSE	FALSE	Status of the hardware gate (start input)
STS_LTCH	Output	BOOL	12.2	TRUE/FALSE	FALSE	Status of the latch input
STS_DO	Output	BOOL	12.3	TRUE/FALSE	FALSE	Output status
STS_C_DN	Output	BOOL	12.4	TRUE/FALSE	FALSE	Status reverse direction. The last count direction is always displayed. The value of STS_C_DN is FALSE after the first call of the instruction.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
STS_C_UP	Output	BOOL	12.5	TRUE/FALSE	FALSE	Status forward direction The last count direction is always displayed. The value of STS_C_UP is TRUE after the first call of the instruction.
COUNTVAL	Output	DINT	14	-2^{31} to $+2^{31}-1$	0	Actual count value
LATCHVAL	Output	DINT	18	-2^{31} to $+2^{31}-1$	0	Actual latch value
JOB_DONE	Output	BOOL	22.0	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	Output	BOOL	22.1	TRUE/FALSE	FALSE	Faulty job
JOB_STAT	Output	WORD	24	0 to W#16#FFFF	0	Job error number

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Note

If you have set the parameter "response of the output" to "no comparison" via the configuration interface, the following is valid:

- The output will be switched in the same manner that a normal output is switched.
- The input parameters CTRL_DO and SET_DO are not active.
- The status bits STS_DO and STS_CMP (status comparator in the IDB) remain reset.

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
STS_CMP	STATIC	BOOL	26.3	TRUE/FALSE	FALSE	Status comparator. Reset with RES_STS. The status bit STS_CMP indicates that the conditions for comparison of the comparator are satisfied or have been satisfied. STS_CMP also indicates that the output was set (STS_DO = TRUE).
STS_OFLW	STATIC	BOOL	26.5	TRUE/FALSE	FALSE	Status overflow Reset with RES_STS.
STS_UFLW	STATIC	BOOL	26.6	TRUE/FALSE	FALSE	Status underflow Reset with RES_STS.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
STS_ZP	STATIC	BOOL	26.7	TRUE/FALSE	FALSE	Status zero-crossing Reset with RES_STS. Only set for counters without main count direction. Indicates the zero crossing. Is also set when the counter is set to 0 or if the counter starts counting as of load value=0.
JOB_OVAL	STATIC	DINT	28	-2^{31} to $+2^{31}-1$	0	Output value for read jobs.
RES_STS	STATIC	BOOL	32.2	TRUE/FALSE	FALSE	Reset status bits. Resets the status bits STS_CMP, STS_OFLW, STS_UFLW and STS_ZP. Two calls of the instruction are required to reset the status bits.

Parameter JOB_ERR and JOB_STAT

JOB_ERR = TRUE if a job error occurs. The precise error cause is displayed in JOB_STAT.

Event class error code	Description
W#16#0121	Compare value too low.
W#16#0122	Compare value too great.
W#16#0131	Hysteresis too small.
W#16#0132	Hysteresis too great.
W#16#0141	Pulse duration too low.
W#16#0142	Pulse duration too great.
W#16#0151	Load value too low.
W#16#0152	Load value too high.
W#16#0161	Counter value too low.
W#16#0162	Counter value too high.
W#16#01FF	Invalid job number.

System error

If a system error occurs, BIE (binary result) = FALSE.

Event class error code	Description
W#16#8001	Wrong operating mode or faulty parameters. Set the correct operating mode during hardware configuration or use the instruction that matches the set operating mode.
W#16#8009	Invalid channel number. Set a channel number ≤ 3 (CPU-specific value).

FREQUENC: Control frequency measurement (S7-300)

Description

You use the "FREQUENC" instruction to operate the frequency meter of the user program.

The following functionality is available:

- Starting/stopping via the software gate SW_GATE
- Enabling/controlling the output DO
- Retrieving the status bits STS_CMP, STS_OFLW and STS_UFLW
- Retrieving the current frequency value MEAS_VAL
- Jobs for reading/writing the internal frequency counter registers

Parameter

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
LADDR	Input	WORD	0	CPU-specific	W#16#0300	The submodule I/O address, which you have specified during hardware configuration. If the I and O addresses are not equal, the lesser of the two addresses must be specified.
CHANNEL	Input	INT	2	CPU 312C: 0 to 1 CPU 313C: 0 to 2 CPU 314C: 0 to 3	0	Channel number
SW_GATE	Input	BOOL	4.0	TRUE/FALSE	FALSE	Software gate for starting/stopping the frequency measurement
MAN_DO	Input	BOOL	4.1	TRUE/FALSE	FALSE	Enable manual output control
SET_DO	Input	BOOL	4.2	TRUE/FALSE	FALSE	Control output
JOB_REQ	Input	BOOL	4.3	TRUE/FALSE	FALSE	Job trigger (positive edge)

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_ID	Input	WORD	6	W#16#0000 = Job without function W#16#0001 = Write low limit W#16#0002 = Write high limit W#16#0004 = Write integration time W#16#0081 = Read integration time W#16#0082 = Read high limit W#16#0084 = Read integration time	0	Job number
JOB_VAL	Input	DINT	8	-2^{31} to $+2^{31}-1$	0	Value for write jobs.
STS_GATE	Output	BOOL	12.0	TRUE/FALSE	FALSE	Status of the internal gate
STS_STRT	Output	BOOL	12.1	TRUE/FALSE	FALSE	Status of the hardware gate (start input)
STS_DO	Output	BOOL	12.2	TRUE/FALSE	FALSE	Output status
STS_C_DN	Output	BOOL	12.3	TRUE/FALSE	FALSE	Status reverse direction. The last count direction is always displayed. The value of STS_C_DN is FALSE after the first call of the instruction.
STS_C_UP	Output	BOOL	12.4	TRUE/FALSE	FALSE	Status forward direction The last count direction is always displayed. The value of STS_C_UP is TRUE after the first call of the instruction.
MEAS_VAL	Output	DINT	14	0 to $+2^{31}-1$	0	Actual frequency value
COUNTVAL	Output	DINT	18	-2^{31} to $+2^{31}-1$	0	Actual count value (starts every time the gate opens at 0)
JOB_DONE	Output	BOOL	22.0	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	Output	BOOL	22.1	TRUE/FALSE	FALSE	Faulty job
JOB_STAT	Output	WORD	24	W#16#0000 to W#16#FFFF	W#16#0000	Job error number

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Note

If you have set the parameter "response of the output" to "no comparison" via the configuration interface, the following is valid:

- The output will be switched in the same manner that a normal output is switched.
- The input parameters MAN_DO and SET_DO are not active.
- The status bit STS_DO remains reset.

Parameters not interconnected to the block (static local data without the reserved parameter RESxx):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
STS_CMP	STATIC	BOOL	26.3	TRUE/FALSE	FALSE	End of measurement status Reset with RES_STS . The measured value is updated after every expired time interval. Here, the end of measurement is reported by the status bit STS_CMP.
STS_OFLW	STATIC	BOOL	26.5	TRUE/FALSE	FALSE	Status overflow Reset with RES_STS.
STS_UFLW	STATIC	BOOL	26.6	TRUE/FALSE	FALSE	Status underflow Reset with RES_STS.
JOB_OVAL	STATIC	DINT	28	-2^{31} to 2^{31} -1	0	Output value for read jobs.
RES_STS	STATIC	BOOL	32.2	TRUE/FALSE	FALSE	Reset status bits. Resets the status bits STS_CMP, STS_OFLW and STS_UFLW. Two calls of the instruction are required to reset the status bits.

Parameter JOB_ERR and JOB_STAT

JOB_ERR = TRUE if a job error occurs. The precise error cause is displayed in JOB_STAT.

Event class error code	Description
W#16#0221	Integration time too low.
W#16#0222	Integration time too great.
W#16#0231	Low limit of the frequency is too low.
W#16#0232	High limit of the frequency is too high.
W#16#0241	High limit of the frequency is too low.

Event class error code	Description
W#16#0242	High limit of the frequency is too high.
W#16#02FF	Invalid job number.

System error

If a system error occurs, BIE (binary result) = FALSE.

Event class error code	Description
W#16#8001	Wrong operating mode or faulty parameters. Set the correct operating mode during hardware configuration or use the instruction that matches the set operating mode.
W#16#8009	Invalid channel number. Set a channel number ≤ 3 (CPU-specific value).

PULSE: Control pulse width modulation (S7-300)

Description

You use the "PULSE" instruction to control the pulse width modulation of the user program.

The following functionality is available:

- Starting/stopping via the software gate SW_EN
- Enabling/controlling the output DO
- Retrieving the status bits STS_EN, STS_STRT and STS_DO
- Input of the output value
- Jobs for reading/writing the registers

Parameter

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Meaning
LADDR	Input	WORD	0	CPU-specific	W#16#0300	The submodule I/O address, which you have specified during hardware configuration. If the I and O addresses are not equal, the lesser of the two addresses must be specified.
CHANNEL	Input	INT	2	CPU 312C: 0 to 1 CPU 313C: 0 to 2 CPU 314C: 0 to 3	0	Channel number

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Meaning
SW_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Software gate for starting/ stopping the output
MAN_DO	Input	BOOL	4.1	TRUE/FALSE	FALSE	Enable manual output control
SET_DO	Input	BOOL	4.2	TRUE/FALSE	FALSE	Control output
OUTP_VAL	Input	INT	6.0	in ppm: 0 to 1000 as S7 analog value: 0 to 27648	0	Default output value if you enter an output value > 1 000 or 27648, the CPU will limit it to 1 000 or 27648
JOB_REQ	Input	BOOL	8.0	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_ID	Input	WORD	10	W#16#0000 = Job without function W#16#0001 = Write period time W#16#0002 = Write switch-on delay W#16#0004 = Write minimum pulse duration W#16#0081 = Read period time W#16#0082 = Read switch-on delay W#16#0084 = Read minimum pulse duration	W#16#0000	Job number
JOB_VAL	Input	DINT	12	-2^{31} to$+2^{31}$ -1	0	Value for write jobs.
STS_EN	Output	BOOL	16.0	TRUE/FALSE	FALSE	Enable status
STS_STRT	Output	BOOL	16.1	TRUE/FALSE	FALSE	Status of the hardware gate (start input)
STS_DO	Output	BOOL	16.2	TRUE/FALSE	FALSE	Output status
JOB_DONE	Output	BOOL	16.3	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	Output	BOOL	16.4	TRUE/FALSE	FALSE	Faulty job
JOB_STAT	Output	WORD	18	W#16#0000 to W#16#FFFF	W#16#0000	Job error number

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_OVAL	Output	DINT	20	-2^{31} to 2^{31} -1	0	Output value for read jobs.

Parameter JOB_ERR and JOB_STAT

JOB_ERR = TRUE if a job error occurs. The precise error cause is displayed in JOB_STAT.

Event class error code	Description
W#16#0411	Period too low.
W#16#0412	Compare value too great.
W#16#0421	On delay too short.
W#16#0422	On delay too long.
W#16#0431	Minimum pulse duration too low.
W#16#0432	Minimum pulse duration too great.
W#16#04FF	Invalid job number.

System error

If a system error occurs, BIE (binary result) = FALSE.

Event class error code	Description
W#16#8001	Wrong operating mode or faulty parameters. Set the correct operating mode during hardware configuration or use the instruction that matches the set operating mode.
W#16#8009	Invalid channel number. Set a channel number ≤ 3 (CPU-specific value).

4.2.5 Communication

4.2.5.1 Common parameter and error information

Common parameters of instructions for S7 basic communication

REQ parameter

The REQ input parameter (request to activate) is a level-triggered control parameter. It is used to trigger the job (the data transfer or the connection termination):

- If you call the instruction for a job that is not currently active, the job is started with REQ = 1. If at the time of the first call of an instruction there is still no connection to the communication partner, then the connection will be established before starting the data transmission.
- If you have initiated a job and it is not yet completed when you call the instruction again for the same job, REQ is not evaluated by the instruction.

Parameters REQ_ID (only "X_SEND" and "X_RCV")

The input parameter REQ_ID is used to identify your send data. It is passed by the operating system of the sending CPU to the "X_RCV (Page 6795)" instruction of the CPU of the communication partner.

You require the REQ_ID parameter at the receiving end

- if you call several "X_SEND (Page 6794)" instructions with different REQ_ID parameters on one send CPU and transfer the data to a communication partner.
- if you use the "X_SEND (Page 6794)" instruction to send data to one communication partner from several send CPUs.

By evaluating REQ_ID, you can save the received data in different memory areas.

Parameters RET_VAL and BUSY

The instructions for S7 basic communication asynchronous, in other words, the processing of a job extends over more than one call. The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

CONT parameter

The CONT input parameter (continue) is a control parameter. Using this parameter, you decide whether or not a connection to the communication partner remains established after the job has completed.

- If you select CONT=0 at the first call, the connection is terminated again after the data transfer is completed. The connection is then available again for data exchange with a new communication partner.
  This method ensures that connection resources are only occupied when they are actually in use.
- If you select CONT=1 at the first call, the connection remains established after the completion of the data transfer.
  This method is, for example, useful when you exchange data cyclically between two stations.

Note

You can also use "X_ABORT (Page 6803)" or "I_ABORT (Page 6787)" to explicitly abort a connection established with CONT=1.

Error information of communication instructions for non-configured S7 connections

Display

You can classify the "real" error information for the instructions "X_SEND (Page 6794)", "X_RCV (Page 6795)", "X_GET (Page 6799)", "X_PUT (Page 6801)", "X_ABORT (Page 6803)", "I_PUT (Page 6786)", "I_GET (Page 6784)", and "I_ABORT (Page 6787)" specified in the following table, as follows:

Error code (W#16#...)	Explanation
809x	Error on the CPU on which the instruction is running
80Ax	Permanent communication error
80Bx	Error on the communication partner
80Cx	Temporary error

The following table contains specific error information for the instructions.

Error code (W#16#...)	Explanation (general)	Explanation (instruction-specific)
0000	Execution completed without errors.	"X_ABORT (Page 6794)" and "I_ABORT (Page 6787)": REQ=1, and the specified connection is not established.
		"X_RCV (Page 6795)": EN_DT=1 and RD=NIL

Error code (W#16#...)	Explanation (general)	Explanation (instruction-specific)
00xy	-	"X_RCV (Page 6795)" when NDA=1 and RD<>NIL: RET_VAL contains the length of the received data block (with EN_DT=0) or the data block copied to RD (with EN_DT=1).
		"X_GET (Page 6799)": RET_VAL contains the length of the received block of data.
		"I_GET (Page 6784)": RET_VAL contains the length of the received block of data.
7000	-	"X_SEND (Page 6794)", "X_GET (Page 6799)", "X_PUT (Page 6801)", "X_ABORT (Page 6803)", "I_GET (Page 6784)", "I_PUT (Page 6786)" and "I_ABORT (Page 6787)": Call with REQ=0 (call without processing), BUSY has the value "0", no data transfer active.
		"X_RCV (Page 6795)": EN_DT=0/1 and NDA=0
7001	First call with REQ=1: Data transfer was triggered; BUSY has the value "1".	-
7002	Intermediate call (REQ irrelevant): Data transfer is already active; BUSY has the value "1".	"X_ABORT (Page 6803)" and "I_ABORT (Page 6787)": Intermediate call with REQ=1
8090	Specified destination address of the communication partner is invalid, for example: • Wrong IOID • Wrong base address exists • Wrong MPI address (> 126)	-

Error code (W#16#...)	Explanation (general)	Explanation (instruction-specific)
8092	Error in SD or RD, for example: addressing of the local data area is not permitted.	"X_SEND (Page 6794)", for example • illegal length for SD • SD=NIL is illegal
		"X_RCV (Page 6795)", for example • More data was received than can fit in the area specified by RD. • RD is of the data type BOOL, the received data is, however, longer than one byte.
		"X_GET (Page 6799)" and "I_GET (Page 6784)", for example • illegal length for RD • The length or the data type of RD does not match the received data. • RD=NIL is illegal.
		"X_PUT (Page 6801)" and "I_PUT (Page 6786)", for example • illegal length for SD • SD=NIL is illegal
8095	The block is already being executed in a lower priority class.	
80A0	Error in the received acknowledgment	"X_PUT (Page 6801)" and "I_PUT (Page 6786)": The data type specified in the SD of the sending CPU is not supported by the communication partner.
80A1	Communications problems: Instruction called after existing connection was canceled	"AS_DIAL (Page 6808)": • The processing of a communication instruction was rejected by the local TS Adapter because no remote connection has been set up by "AS_DIAL". • The remote connection to the remote TS Adapter was aborted during the processing of the communication instruction.
80B0	Object is not reachable, for example, DB not loaded	Possible for "X_GET (Page 6799)" and "X_PUT (Page 6801)" and "I_GET (Page 6784)" and "I_PUT (Page 6786)"
80B1	Error in the ANY pointer. The length of the data area to be transferred is incorrect.	-

Error code (W#16#...)	Explanation (general)	Explanation (instruction-specific)
80B2	Hardware error: module does not exist • The configured slot is not occupied. • Actual module type does not match expected type • Distributed I/O is not available. • No entry for the module in the corresponding SDB.	Possible for "X_GET (Page 6799)" and "X_PUT (Page 6801)" and "I_GET (Page 6784)" and "I_PUT (Page 6786)".
80B3	Data may either only be read or only written, for example, write-protected DB	Possible for "X_GET (Page 6799)" and "X_PUT (Page 6801)" and "I_GET (Page 6784)" and "I_PUT (Page 6786)".
80B4	Data type error in the ANY pointer, or ARRAY of the specified data type not permitted.	"X_GET (Page 6799)" and "X_PUT (Page 6801)" and "I_GET (Page 6784)" and "I_PUT (Page 6786)": The data type specified in VAR_ADDR is not supported by the communication partner.
80B5	Execution rejected due to illegal mode	Possible for "X_SEND (Page 6794)". With "AS_DIAL (Page 6808)": • The processing of a communication instruction was rejected by local TS Adapter because the "DIAL" function of "AS_DIAL" is not yet completed. • The DIAL function of "AS_DIAL" was requested even though a remote connection is already set up for AS-AS remote coupling.
80B6	The received acknowledgment contains an unknown error code.	-
80B7	Data type and/or length of the transferred data does not fit in the area on the partner CPU to which it should be written.	Possible for "X_PUT (Page 6801)" and "I_PUT (Page 6786)"
80B8	-	"X_SEND (Page 6794)": "X_RCV (Page 6795)" of the communication partner rejected the data acceptance (RD=NIL).
80B9	-	"X_SEND (Page 6794)": The block of data was identified by the communication partner (calling "X_RCV (Page 6795)" with EN_DT=0), it has not yet been entered in the user program because the partner is in STOP mode.
80BA	The response of the communication partner does not fit in the communication frame.	-
80C0	The specified connection is being used by another job.	-

Error code (W#16#...)	Explanation (general)	Explanation (instruction-specific)
80C1	Lack of resources on the CPU on which the instruction is executed, for example: • The maximum number of different send jobs is already being executed on the module. • The connection resource is in use, for example, receiving data.	-
80C2	Temporary lack of resources on the communication partner, for example: • The communication partner is currently processing the maximum number of jobs. • The required resources (memory, etc.) are being used. • Not enough work memory. (Compress memory).	With "AS_DIAL (Page 6808)": • Temporary lack of resources on the remote CPU. • The remote CPU with the MPI address is not available or does not exist.
80C3	Error in establishing a connection, for example: • The local S7 station is not attached to the MPI subnet. • You have addressed your own station on the MPI subnet. • The communication partner is no longer reachable. • Temporary lack of resources on the communication partner	-

4.2.5.2 S7 communication

Overview of the S7 communication instructions

Classification

For S7 communication, connection configuration is needed. The integrated communication functions are called in the user program via instructions.

These instructions can be classified in the following categories:

- Instructions for data exchange
- Instructions for changing the operating mode

- Instructions for querying the operating mode
- One instruction each for querying the connection

Note

If your S7-300 CPU does not have an Ethernet interface, you will need a SIMATIC NET CP (Page 6499) from the S7-300 product range to execute S7-300 instructions. For further information, please see the related documentation.

Instructions for data exchange

Communication instructions for data exchange are used to exchange data between two communication partners. If a communication instruction exists only on the local module, this is known as data exchange configured at one end. If it exists on the local as well as on the remote module, this is known as data exchange configured at both ends.

Instruction	Brief description
"USEND (Page 6146)" / "URCV (Page 6149)"	Fast unacknowledged exchange of data regardless of the sequential processing of the communication function ("URCV") on the communication partner (for example, operational and maintenance alarms). This means that the data can be overwritten by more recent data at the communication partner.
"BSEND (Page 6151)" / "BRCV (Page 6154)"	Secure transfer of a data block to the communication partner. This means that data transmission is not complete until the receive function ("BRCV") on the communication partner has accepted the data.
"GET (Page 6141)"	Program-controlled reading of tags without additional communication function in the user program of the communication partner.
"PUT (Page 6144)"	Program-controlled writing of tags without additional communication function in the user program of the communication partner.
"PRINT (Page 6170)"	Send data to a printer (S7-400 only: CP441).

Instructions for changing the operating mode

With instructions for changing the operating mode, you control the operating mode of a remote device.

The data exchange instructions for changing the operating mode are configured at one end.

Instruction	Brief description
"START (Page 6176)"	Trigger a WARM RESTART of an S7-300/400 CPU if it is in STOP mode.
"STOP (Page 6178)"	STOP of an S7-300/400 CPU if it is in RUN, HOLD, or startup mode
"RESUME (Page 6180)"	Trigger a hot restart of an S7-400 CPU if it is in STOP mode.

Instructions for querying the operating mode

With instructions for querying the operating mode, you can obtain information about the operating mode of a remote device.

With the "STATUS" instruction, data exchange is configured at one end, while with the "USTATUS (Page 6184)" instruction, data exchange is configured at both ends.

Instruction	Brief description
"STATUS (Page 6181)"	Supplies the operating mode of a communication partner (S7-400 CPU) when requested by the user.
"USTATUS (Page 6184)"	Receives the operating mode of an S7-400 CPU when it changes its mode, if the corresponding connection attribute (send operating mode messages) has been set.

Instruction for querying the connection

Instruction for S7-400	Instruction for S7-300	Brief description
"CONTROL (Page 6167)"	-	Queries the state of a connection that belongs to an instance of an instruction.
-	"C_CNTRL (Page 6157)"	Query the status of a connection using the connection ID.

Note

You can also use the "C_DIAG (Page 5510)" instruction to diagnose the current connection status (for S7-400 only).

Data consistency

Definition

The size of the data area which can be modified simultaneously by concurrent processes is called the consistent data area. Data areas which are larger than the consistent data area can thus be falsified as a whole.

This means that a data area which belongs together and which is larger than consistent data area can consist in part of new and of old consistent data blocks at the same time.

Example

An inconsistency can arise if a communication block is interrupted, for example, by a hardware interrupt OB with a higher priority. If the user program in this OB now changes the data which have already been processed in part by the communication block, the transferred data originate:

- In part from the time before the hardware interrupt was processed
- And in part from the time after the hardware interrupt was processed

This means that these data are inconsistent (not coherent).

Effect

If larger packages of data are to be transferred in a consistent form, the transfer should not be interrupted. This can, for example, increase the interrupt reaction time of the CPU.

Data consistency with SIMATIC

- If the user program includes a communication function, for example "BSEND (Page 6151)" / "BRCV (Page 6154)" that accesses common data, access to this data area can, for example, be coordinated using the DONE parameter. The data consistency of the communication areas which are transferred locally with a communication block can therefore be ensured in the user program.
- However, in the case of S7 communication functions, for example "PUT (Page 6144)" / "GET (Page 6141)" or write/read using OP communication, the size of the consistent data areas must already be taken into consideration during programming or configuration, since no communication block is available in the user program of the target device (server) to synchronize communication data in the user program.
- With the S7-300, the communication data are copied consistently into the user memory in blocks of 32 bytes in the cycle control point of the operating system. Data consistency is not guaranteed for larger data areas. If a defined data consistency is required, the communication data in the user program may not exceed 32 bytes (maximum of 8 bytes, depending on the version).
- In the S7-400 by contrast the communication data are not processed in the cycle control point, but in fixed time slices during the program cycle. The consistency of a tag is ensured by the system.
- These communication areas can then be accessed consistently using the "PUT (Page 6144)" / "GET (Page 6141)" instructions or when reading/writing tags, for example by an OP or an OS.

Note

Additional information on data consistency is provided in the description of the specific instructions.

Common parameters of instructions for S7 communication

Classification

The parameters of the instructions for S7 communication can be divided into the following five categories according to their functions:

1. Control parameters are used to activate an instruction.
2. Addressing parameters are used to address the remote communication partner.
3. Send parameters point to the data areas that are to be sent to the remote partner.

4. Receive parameters point to the data areas where the data received from remote partners will be entered.
5. Status parameters are used to monitor whether the instruction has completed its task without error or for the analysis of any errors that have occurred.

Control parameters

Data exchange will only be activated if the appropriate control parameters have a defined value (for example, are set) when the instruction is called or when the value has undergone a specific change since the previous call (for example, a positive edge).

Note

First call for S7-300

For the first call, set the REQ parameter to FALSE .

Addressing parameters

Parameters	Description
ID	Reference to the local connection description (specified by the configuration of the connection). Note: The ID W#16#EEEE is not permitted for S7communication instructions.
R_ID	With the R_ID parameter, you specify that a send and a receive instruction belong together: The R_ID parameter must match in the instruction at the sending end and the instruction at the receiving end. This allows the communication of several instruction pairs via the same logic connection. • R_ID must be specified in the form DW#16#wxyzWXYZ. • The instruction pairs of a logical connection specified in R_ID must be unique for this connection.

The PI_NAME parameter is only described for the relevant instructions (S7-400 only).

Note

Addressing parameters ID and R_ID

S7-300: You can change the settings of the addressing parameters ID and R_ID during runtime. The new parameters are validated with each new job after the previous job has been closed.

You can use the following options to reduce the number of instance DBs and therefore the work memory required:

1. With tag IDs, you can use several connections via one data instance block.
2. With tag R_IDs, you can define several pairs of sending and receiving instructions for one job with a single instance.
3. You can combine cases 1 and 2.

Please note that the new parameters are valid after the last job is executed. When you activate the sending operation, the R_ID parameter in the instruction at the sending end must match its counterpart at the receiving end.

S7-400: The ID and R_ID addressing parameters are evaluated only when the instruction is called the first time or when loading the instance over another (the actual parameters or the predefined values from the instance). The first call therefore specifies the communication relation (connection) with the remote partner until the next warm or cold restart.

Status parameters

With the status parameters, you monitor whether the instruction has completed its task correctly or whether it is still active. The status parameters also indicate errors.

Note

The status parameters are valid for one cycle only, namely from the first command following the call until the next call. As a result, you must evaluate these parameters after each instruction cycle.

Send and receive parameters

If you do not use all send or receive parameters of an instruction, the first unused parameter must be a NIL pointer and the parameters used must be located one after the other without any gaps.

Note

First call for S7-400

During the first call, the ANY pointer specifies the maximum amount of user data that can be transferred for the job. On the basis of this, a communication data buffer is created in the work memory of the CPU to ensure data consistency. This buffer occupies up to 480 bytes of work memory. We recommend you run the first call in the warm or cold restart OB if the block is not reloaded with the instruction call when the CPU is in RUN mode.

At subsequent calls you can send/receive any amount of data, however, no more than with the first call.

The "BSEND (Page 6151)" and "BRCV (Page 6154)" instructions are an exception to this rule. You can use them to transfer up to 64 kbytes per job.

For communication instructions configured at both ends

- The number of the SD_i and RD_i parameters used must match at the send and receive end.
- The data types of the SD_i and RD_i parameters that belong together must match at the send and receive end.
- The amount of data to be sent according to the SD_i parameter must not exceed the range made available by the corresponding RD_i (does not apply to "BSEND (Page 6151)" / "BRCV (Page 6154)").

If you do not keep to the rules above, this is indicated by ERROR = 1 and STATUS = 4.

User data size

With the "USEND (Page 6146)", "URCV (Page 6149)", "GET (Page 6141)" and "PUT (Page 6144)" instructions, the amount of data to be transmitted must not exceed a defined user data length. The maximum user data size depends on:

- The instruction used
- The communication partner.

The guaranteed minimum size of the user data for an instruction with 1-4 tags is listed in the following table:

Instruction	Partner: S7-300	Partner: S7-400
PUT / GET	160 bytes	400 bytes
USEND / URCV	160 bytes	440 bytes
BSEND / BRCV	32768/65534 bytes	65534 bytes

Further information on the user data size can be found in the technical data of the respective CPU.

Exact user data size

If the user data size specified above is insufficient you can determine the maximum byte length of the user data as follows:

- First read the data block size valid for communication from the following table:

Local CPU	Remote CPU	Data block size in bytes
S7-300	Any	240 (S7-300)
S7-400	S7-300	240 (S7-400)
S7-400	S7-400	480

- Use this value in the following table to read the maximum possible user data length in bytes as total of the parameters used. It applies for even lengths of the areas SD_i, RD_i, and ADDR_i. For each range of uneven length the maximum possible user data length is reduced by one byte.

		Number of SD_i, RD_i, ADDR_i parameters used			
Data block size	Instruction	1	2	3	4
240 (S7-300)	PUT/GET/ USEND	160	-	-	-
240 (S7-300 via integrated interface)	PUT	212	-	-	-
	GET	222	-	-	-
	USEND	212	-	-	-
240 (S7-400)	PUT	212	196	180	164
	GET	222	218	214	210
	USEND	212	-	-	-
480	PUT	452	436	420	404
	GET	462	458	454	450
	USEND	452	448	444	440

Startup characteristics of instructions for S7 communication

Requirements

In the following description for S7-400, it is assumed that:

- The connection descriptions (SDBs) exist on the modules.
- The configured connections have been established.
- The actual parameter for ID matches the configured connection ID for each instruction.

Warm restart and cold restart

During a warm and a cold restart, all instructions are set to the NO_INIT status. The actual parameters stored in the instance DBs are unchanged.

Warm restart and cold restart with data exchange instructions configured at both ends

With data exchange instructions configured at both ends, the two modules do not normally both run a warm or cold restart at he same time. The adjustment is made implicitly by the instruction based on the following rules.

- The receive blocks ("URCV (Page 6149)", "BRCV (Page 6154)") react as follows:
  - If the instruction has received a job but has not acknowledged this job at the time of the warm or cold restart, it generates a sequence abort frame ("BRCV (Page 6154)") and then immediately branches to the NO_INIT status.
  - With "BRCV (Page 6154)", it is possible that another data segment will be received despite the transferred sequence abort. This will be discarded locally.
  - With "URCV (Page 6149)", the change to the NO_INIT status occurs immediately.
- The send blocks ("USEND (Page 6146)", "BSEND (Page 6151)") react as follows:
  - If the "BSEND (Page 6151)" instruction has started a job sequence that has not yet been completed, it sends a sequence abort during the warm or cold restart. It then branches to the NO_INIT status immediately afterwards. An acknowledgement that arrives at a later time is discarded locally.
  - If the "BSEND (Page 6151)" instruction has already sent or received a sequence abort when the warm or cold restart is requested, it changes immediately to the NO_INIT status.
  - In all other cases and if the instruction sends only alarms (for example, "USEND (Page 6146)"), local processing is aborted and the instruction immediately branches to the NO_INIT status.

Warm restart and cold restart with data exchange instructions configured at one end

It is assumed that the server on the communication partner is operational after the connections have been established, in other words that it can process jobs or output alarms at any time.

Instructions that send out jobs and expect acknowledgements respond as follows:

- The current processing is aborted followed by a branch to the NO_INIT status immediately afterwards. If an acknowledgement for the job sent prior to the warm or cold restart arrives later, it is discarded locally.
- A new job may have been sent before the acknowledgement of the earlier job is received.

Instructions that output or receive alarms react as follows:

- The current processing is aborted followed by a branch to the NO_INIT status immediately afterwards.
- With "USTATUS (Page 6184)", alarms that arrive during the NO_INIT and DISABLED statuses are discarded locally.

Hot restart

The instructions for S7 communication are set to the NO_INIT status only during a warm or cold restart. This means that they react like user function blocks that can be resumed following a hot restart.

Memory reset

A memory reset always causes all connections to be terminated. Since a warm or cold restart is the only possible startup type for the user program after a memory reset, all instructions for S7 communication (if they still exist) are set to the NO_INIT status and initialized. Partner blocks in a module whose memory was not reset change to the IDLE or ENABLED or DISABLED statuses as a reaction to the connection being aborted.

Error behavior of instructions for S7 communication

Introduction

The following section describes how instructions for S7 communication in S7-400 react to problems.

Connection abort

The connections assigned to the instruction instances are monitored for aborts.

If a connection is aborted, the reaction of the instruction depends on its internal status:

- If the connection abort is detected in the IDLE or ENABLED status, the instruction reacts as follows:
  - It branches to the ERROR status and outputs the error ID "Communication problems" at the ERROR and STATUS output parameters.
  - When it is next called, the instruction returns to its original status and checks the connection again.
- An instruction that is not in the IDLE or DISABLED status reacts as follows:
  - It aborts its processing, changes to the ERROR status immediately or at the next call, and outputs the error ID "Communication problems" at the ERROR and STATUS output parameters.
  - When it is next called, the instruction changes to the IDLE, DISABLED or ENABLED status. In the IDLE and ENABLED statuses, the connection is checked again.

This procedure will also be executed if the connection has again been set up in the meantime.

Power down

A power down with battery backup followed by a restart causes all established connections to be terminated. Therefore, the points made above apply to all instructions involved.

If there is a power down with battery backup followed by an automatic warm or cold restart, the points made about terminated connections and warm or cold restarts apply.

In the special case of an automatic warm or cold restart without battery backup, where a memory reset is executed automatically after power returns, the instructions for S7 communication react as described in the "Startup characteristics of instructions for S7 communication (Page 6138)" section.

Reaction to operating mode changes

If there is an operating mode change between the STOP, STARTUP, RUN, and HOLD modes, the instruction remains in its current status (exception: During a warm or cold restart, it changes to the NO_INIT status). This applies both to communications instructions configured at one end and at both ends.

Error interface to the user program

If an error occurs during the processing of an instruction, it always changes to the ERROR status. At the same time, the ERROR output parameter is set to "1" and the corresponding error ID is entered in the STATUS output parameter. You can evaluate this information in your program.

Examples of possible errors:

- Error when collecting send data.
- Error when copying receive data into the receive areas (for example, attempting to access a DB that does not exist)
- The length of the data area sent does not match the length of the receive area specified in the partner instruction.

GET: Read data from a remote CPU

Description

With the help of the instruction, you can read data from a remote CPU if the connection does not take place via a CP.

- **S7-300:** The data is read on a positive edge at REQ. The parameter values ID, ADDR_1 and RD_1 are adopted on each positive edge at REQ . After a job has been completed, you can assign new values to the ID, ADDR_1 and RD_1 parameters.
- **S7-400:** The instruction is started on a positive edge at control input REQ . The relevant pointers to the areas to be read out (ADDR_i) are then sent to the partner CPU.

The remote partner returns the data. The received data is copied to the configured receive areas (RD_i) at the next call. Make sure that the areas defined with the parameters ADDR_i and RD_i match in terms of number, length and data type.

Completion of this action is indicated by the status parameter NDR having the value "1". Reading can only be activated again after the previous reading process has been completed.

The remote CPU can be in RUN or STOP mode. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being read or if the data type check results in an error.

Parameters

The following table shows the parameters of the "GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR / STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
S7-300: ADDR_1 **S7-400:** ADDR_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointers to the areas on the partner CPU that are to be read. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10). When transferring data structures (e.g., Struct, Array), the data type CHAR, BYTE, WORD or DWORD must be used at the ADDR parameter.
S7-300: RD_1 **S7-400:** RD_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointers to the areas on the local CPU in which the read data is entered. Only the data types BOOL are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "GET" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: – Maximum number of parallel jobs/instances exceeded.
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the receive area pointers RD_i relating to the data length or the data type.
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "GET" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	20	• **S7-400** Not enough work memory. Remedy: Reduce the program code in the memory. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) – Possible when first called
1	27	**For S7-300 only:** No function code for this instruction exists in the CPU.

Data consistency

The data is received in a consistent state if the following point is observed:

Evaluate the part of the receive area RD_i currently being used completely before initiating another job.

PUT: Write data to a remote CPU

Description

With the help of the instruction, you can write data to a remote CPU if the connection does not take place via a CP.

- **S7-300:** The data is sent on a positive edge at REQ. The parameter values ID, ADDR_1 and SD_1 are adopted on each positive edge at REQ. After a job has been completed, you can assign new values to the ID, ADDR_1 and SD_1 parameters.
- **S7-400:** The instruction is started on a positive edge at control input REQ. The pointers to the areas to be written (ADDR_i) and the data (SD_i) are then sent to the partner CPU.

The remote partner saves the required data under the addresses supplied with the data and returns an execution acknowledgement. Make sure that the areas defined with the parameters ADDR_i and SD_i match in terms of number, length and data type.

If no errors occur, this is indicated at the next call with status parameter DONE = "1". The writing process can only be activated again after the last job is complete.

The remote CPU can be in RUN or STOP mode. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being written or if the execution check results in an error.

Parameters

The following table shows the parameters of the "PUT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job executed without errors.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.

Parameter	Declaration	Data type	Memory area	Description
S7-300: ADDR_1 **S7-400:** ADDR_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointers to the areas on the partner CPU to which the data will be written. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10). When transferring data structures (e.g., Struct, Array), the data type CHAR, BYTE, WORD or DWORD must be used at the ADDR parameter.
S7-300: SD_1 **S7-400:** SD_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointers to the areas on the local CPU which contain the data to be sent. Only the data types BOOL are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for PUT that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: – Maximum number of parallel jobs/instances exceeded.
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the send area pointers SD_i relating to the data length or the data type
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB)

ERROR	STATUS (decimal)	Explanation
1	12	When the instruction was called • An instance DB was specified that does not belong to "PUT" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	20	• **S7-400** Not enough work memory. Remedy: Load the associated instance DB again. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded at CPU-RUN (STOP-RUN transition of the CPU or CP required). – Possible when first called
1	27	**For S7-300 only:** No function code for this instruction exists in the CPU.

Data consistency for S7-300:

To ensure data consistency, send area SD_1 may not be written to again until the current send job has been completed. This is the case when the value of the status parameter DONE changes to "1".

Data consistency for S7-400 and S7-300 via an integrated interface:

When a send operation is activated (rising edge at REQ), the data to be sent from the send area SD_i is copied from the user program. After the instruction call, you can write to these areas again without corrupting the current send data.

Note

The send operation is only complete when the DONE status parameter has the value "1".

USEND: Send data uncoordinated

Description

The "USEND" instruction sends data to a remote partner instruction of type "URCV (Page 6149)". The sending process is carried out without coordination with the partner instruction. This means that the data transfer is carried out without acknowledgment by the partner instruction.

- **S7-300:** The data is sent on a positive edge at REQ. The parameter values R_ID, ID and SD_1 are adopted on each positive edge at REQ . After a job has been completed, you can assign new values to the R_ID, ID and SD_1 parameters.
- **S7-400:**The data is sent on a positive edge at control input REQ. The data to be sent is referenced by the parameters SD_1, ... SD_4 (it is not necessary for all four send parameters to have a value).

You must, however, make sure that the areas defined by the parameters SD_1 to SD_4 / SD_1 and RD_1 to RD_4 / RD_1 (with the associated partner instruction "URCV (Page 6149)") match in terms of:

- Number
- Length and
- Data type

The R_ID parameter must be identical in both instructions.

Successful completion of the send operation is indicated when the value of the DONE status parameter is set to logical "1".

Parameters

The following table shows the parameters of the "USEND" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a positive edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 6134)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
S7-300: SD_1 **S7-400:** SD_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointer on the i-th send area. Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: – Maximum number of parallel jobs/instances exceeded
1	4	Errors in the send area pointers SD_i relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "USEND" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	18	• R_ID already exists in the connection ID. • Also for S7-300: The instances were overloaded at CPU-RUN (STOP-RUN transition of the CPU or CP required).
1	20	• **S7-400** Not enough work memory. Remedy: Reduce the program code in the memory. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded at CPU-RUN (STOP-RUN transition of the CPU or CP required). – Possible when first called
1	27	**For S7-300 only:** No function code for this instruction exists in the CPU.

Data consistency

S7-300: To ensure data consistency, send area SD_1 may not be written to again until the current send job has been completed. This is the case when the value of the status parameter DONE changes to "1".

S7-400 and S7-300 via an integrated interface: When a send operation is activated (positive edge at REQ), the data to be sent from the send area SD_i is copied from the user program.

After the instruction call, you can write to these areas again without corrupting the current send data.

Note

The send operation is only complete when the DONE status parameter has the value "1".

URCV: Receive data uncoordinated

Description

The "URCV" instruction receives data asynchronously from a remote partner instruction of type "USEND (Page 6146)" and copies this data into the configured receive areas.

The instruction is ready to receive if there is a logical 1 at the EN_R input. An active job can be canceled with EN_R=0.

- **S7-300:** The parameters R_ID, ID and RD_1 are adopted on each positive edge at EN_R. After a job has been completed, you can assign new values to the R_ID, ID and RD_1 parameters.
- **S7-400:** The receive data areas are referenced by the parameters RD_1 to RD_4.

Make sure that the areas defined by the parameters RD_i / RD_1 and SD_i / SD_1 (with the associated partner instruction "USEND (Page 6146)") match in terms of:

- Number
- Length and
- Data type

Successful completion of the copy operation is indicated when the value of the NDR status parameter is set to logical "1". The R_ID parameter must be identical in both instructions.

Parameters

The following table shows the parameters of the "URCV" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 6134)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.

Parameters	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
S7-300: RD_1 **S7-400:** RD_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointer to the i-th receive area: Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS (decimal)	Explanation
0	9	Overrun warning: older received data were overwritten by newer received data.
0	11	Warning: The received data is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: Maximum number of parallel jobs/instances exceeded
1	4	Errors in the receive area pointers RD_i relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "URCV" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	18	• R_ID already exists in the connection ID. • Also for S7-300: The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.)

ERROR	STATUS (decimal)	Explanation
1	19	The associated "USEND (Page 6146)" instruction sends data faster than this can be copied to the receive areas by "URCV".
1	20	• **S7-400** Not enough work memory. Remedy: Reduce the program code in the memory. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) – Possible when first called
1	27	**For S7-300 only:** No function code for this block exists in the CPU.

Data consistency

The data is received in a consistent state if the following point is observed:

- **S7-300:** After the status parameter NDR has changed to value "1", you must call "URCV" again immediately with value "0" at EN_R. This ensures that the receive area is not already overwritten before you have evaluated it. Evaluate the receive area RD_1 completely before you call the block with value "1" at control input EN_R.
- **S7-400:** After the status parameter NDR has changed to the value "1", there is new receive data in your receive areas (RD_i). A new block call may cause these data to be overwritten with new receive data. If you want to prevent this, call "URCV" (for example, during cyclic block processing) with the value 0 at EN_R until you have finished processing the received data.

BSEND: Send data in segments

Description

The "BSEND" instruction sends data to a remote partner instruction of type "BRCV (Page 6154)". With this type of data transfer, more data can be transported between the communication partners than is possible with all other communication instructions for configured S7 connections. The following aggregates can be transferred:

- 32768 bytes for S7-300 via SIMATIC-Net CPs
- 65534 bytes for S7-400 and S7-300 via an integrated interface

Functional description

The data area to be transmitted is segmented. Each segment is sent individually to the partner. The last segment is acknowledged by the partner when it is received, regardless of the associated "BRCV (Page 6154)" call.

- **S7-300:** The data is sent on a positive edge at REQ. The parameters R_ID, ID, SD_1 and LEN are adopted on each positive edge at REQ . After a job has been completed, you can assign new values to the R_ID, ID, SD_1 and LEN parameters. To transfer segmented data, the instruction must be called cyclically in the user program.
  The start address and the maximum length of the data to be sent are specified by SD_1. The length of the data block is specified by LEN based on the job.
- **S7-400 and S7-300 via an integrated interface:** The send operation is activated after the instruction is called and a positive edge on control input REQ. Sending the data from the user memory is asynchronous to the execution of the user program.
  The start address of the data to be sent is specified by SD_1. The length of the send data is specified by LEN based on the job. In this case, LEN replaces the length section of SD_1.

The R_ID parameter must be identical in the related instructions. If there is a positive edge at control input R, the current sending process is canceled. Successful completion of the send operation is indicated by the value of the DONE status parameter being set to "1".

A new send job cannot be processed until the previous send operation has been completed and the status parameter DONE or ERROR has the value "1".

Due to the asynchronous data transfer, a new data transfer can only be initiated if the previous data has been fetched by a partner instruction call. Until the data is fetched, the status value "7" (see below) will be output when "BSEND" is called.

Parameters

The following table shows the parameters of the "BSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange at a positive edge
R	Input	BOOL	I, Q, M, D, L	Control parameter reset, activates an abort of the current data exchange at a positive edge
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 6134) If there is a connection via a CP 441 to S5 or third-party devices, R_ID contains the address information of the remote device. For further information, refer to the description of the CP 441.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job completed without error.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
SD_1	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointer to the send area. Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** When the ANY pointer accesses a DB, the DB must always be specified. (For example: P#DB10.DBX5.0 byte 10)
LEN	InOut	WORD	I, Q, M, D, L	Length of the data field to be sent in bytes

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for "BSEND" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	• Communication problems, for example: Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: Maximum number of parallel jobs/instances exceeded.
1	2	Negative acknowledgment of partner instruction. The function cannot be executed.
1	3	R_ID is unknown on the connection specified by the ID or the receive block has not yet been called.
1	4	Error in the send area pointer SD_1 involving the data length or the data type, or the value "0" was transferred at the LEN parameter.
1	5	Reset request was executed.
1	6	Partner instruction is in DISABLED status (EN_R has the value "0"). Also check the input parameters of BRCV for consistency with BSEND.

ERROR	STATUS (decimal)	Explanation
1	7	Partner instruction is in the wrong state. The receive instruction was not called again after the last data transmission.
1	8	Access to remote object in the user memory was rejected: The destination area for the associated "BRCV" is too small. The corresponding BRCV reports ERROR = 1, STATUS = 4 or ERROR = 1, STATUS = 10.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • an instance DB was specified that does not belong to "BSEND" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	18	• R_ID already exists in the connection. • Also for S7-300: The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.)
1	20	• **S7-400** Not enough work memory. Remedy: Reduce the program code in the memory. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) – Possible when first called – Memory bottleneck in CP
1	27	**For S7-300 only:** No function code for this instruction exists in the CPU.

Data consistency

To ensure data consistency, you may only write to the part of the currently used sending area SD_1 after the current send operation is complete. This is the case when the value of the status parameter DONE changes to "1".

BRCV: Receive data in segments

Description

The "BRCV" instruction receives data from a remote partner instruction of type "BSEND (Page 6151)". After each received data segment, an acknowledgment is sent to the partner instruction and the LEN parameter is updated.

The instruction is ready to receive data after it is called with the value "1" at control input EN_R. An active job can be canceled with EN_R=0.

The start address and the maximum length of the receive area are specified by RD_1. The length of the received block of data is shown in LEN .

- **S7-300:** The parameters R_ID, ID and RD_1 are adopted on each positive edge at EN_R. After a job has been completed, you can assign new values to the R_ID, ID and RD_1 parameters. For the transmission of segmented data, the instruction must be called cyclically in the user program.
- **S7-400 and S7-300 via an integrated interface:** Receipt of the data from the user memory is asynchronous to the execution of the user program.

The R_ID parameter must be identical in the related instructions.

Value "1" at the NDR status parameter indicates that all data segments have been received without error. The received data remains unchanged until the next call with EN_R=1.

If the instruction is called again while data is being received asynchronously, this leads to a warning being output in the STATUS parameter; if the call is made with EN_R=0, receiving is canceled and the instruction returns to its initial state.

Parameters

The following table shows the parameters of the "BRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 6134) If there is a connection via a CP 441 to S5 or third-party devices, R_ID contains the address information of the remote device. For further information, refer to the description of the CP 441.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	

Parameter	Declaration	Data type	Memory area	Description
RD_1	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointer to the receive area. The length information specifies the maximum length of the block to be received. Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** When the ANY pointer accesses a DB, the DB must always be specified. (For example: P# DB10.DBX5.0 byte 10).
LEN	InOut	WORD	I, Q, M, D, L	Length of the data already received in bytes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for "BRCV" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: The received data is already being processed in a priority class with lower priority.
0	17	Warning: Instruction receives data asynchronously. The LEN parameter shows the amount of data already received in bytes.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: Maximum number of parallel jobs/instances exceeded.
1	2	Function cannot be executed (protocol error)
1	4	Error in the receive area pointer RD_1 relating to the data length or data type. The block of data sent is longer than the receive area.
1	5	Reset request received, incomplete transfer.
1	8	Access error in associated "BSEND (Page 6151)": After the last valid data segment has been sent, ERROR = 1 and STATUS = 4 or ERROR = 1 and STATUS = 10 is reported.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "BRCV" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).

ERROR	STATUS (decimal)	Explanation
1	18	• R_ID already exists in the connection. • Also for S7-300: The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.)
1	20	• **S7-400** Not enough work memory. Remedy: Reduce the program code in the memory. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) – Possible when first called – Memory bottleneck in CP
1	27	**For S7-300 only:** No function code for this instruction exists in the CPU.

Data consistency

The data is received in a consistent state if the following point is observed: Evaluate the currently used part of the receive area RD_1 completely before you call the block again with the value 1 at control input EN_R.

Special case for receiving data (S7-400 only)

If the receiving CPU has a "BRCV" instruction that is ready to receive data (in other words, a call with EN_R =1 has already been made) and the CPU changes to STOP mode before the corresponding send instruction has sent the first data segment of a job, the following will occur:

- The data in the first job after the receiving CPU has gone into STOP mode are fully entered in the receive area.
- The "BSEND (Page 6151)" partner instruction receives a positive acknowledgment of this.
- No more "BSEND (Page 6151)" jobs can be accepted by a receiving CPU in STOP mode.
- As long as the CPU remains in STOP mode, both NDR and LEN have the value "0".

To prevent information about the received data being lost, perform a hot restart on the receiving CPU and call "BRCV" with EN_R=1.

C_CNTRL: Query connection status (S7-300)

Description

With this instruction, you query the status of a connection **for S7-300**. The current status of the connection addressed by ID is obtained following the instruction call with value "1" at the EN_R control input.

Parameters

The following table shows the parameters of the "C_CNTRL" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter, signals ready to receive if the input is set.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
RET_VAL	Output	INT	I, Q, M, D, L	Error information
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
C_CONN	Output	BOOL	I, Q, M, D, L	Status of the corresponding connection. Possible values: • 0: Connection aborted or not yet established. • 1: Connection available.
C_STATUS	Output	WORD	I, Q, M, D, L	Connection status: • W#16#0000: Connection is not set up • W#16#0001: Connection is being set up • W#16#0002: Connection is set up • W#16#000F: No data on connection status available (for example during CP startup) • W#16#00FF: Connection is not configured

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters RET_VAL, ERROR and STATUS

The RET_VAL output parameter can have the following two values for the "C_CNTRL" instruction:

- 0000H: No error occurred during execution.
- 8000H: An error occurred during execution.

ERROR	STATUS (decimal)	Explanation
1	10	CP access error. Another job is currently running. Repeat job later.
1	27	There is no function code on the CPU for this instruction.

Note

The output parameters ERROR and STATUS need to be evaluated even if the output parameter RET_VAL shows the value 0000H .

Others (S7-300)

GET_S: Read data from a remote CPU (S7-300)

Description

With this instruction, you can read data from a remote CPU **with S7-300**. The connection can take place via the integrated interface as well as a CP.

The data is read on a rising edge at REQ. The parameter values ID, ADDR_1 and RD_1 are adopted on each positive edge at REQ . After a job has been completed, you can assign new values to the ID, ADDR_1 and RD_1 parameters.

The remote partner returns the data. The received data is copied to the configured receive areas (RD_1) at the next call. Make sure that the areas defined with the parameters ADDR_1 and RD_1 match in terms of number, length and data type.

Completion of this action is indicated by the status parameter NDR having the value "1". Reading can only be activated again after the previous reading process has been completed.

The remote CPU can be in RUN or STOP mode. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being read or if the data type check results in an error.

Parameters

The following table shows the parameters of the "GET_S" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.

Parameter	Declaration	Data type	Memory area	Description
ERROR / STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
ADDR_1	InOut	ANY	M, D	Pointers to the areas on the partner CPU that are to be read.
RD_1	InOut	ANY	M, D	Pointers to the areas on the local CPU in which the read data is entered. Only the data types BOOL are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "GET_S" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Maximum number of parallel jobs/instances exceeded.
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the receive area pointers RD_1 relating to the data length or the data type.
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).

ERROR	STATUS (decimal)	Explanation
1	12	When the instruction was called • An instance DB was specified that does not belong to "GET_S" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	20	• Maximum number of parallel jobs/instances exceeded • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) • Possible when first called
1	27	There is no function code on the CPU for this instruction.

Data consistency

The data is received in a consistent state if the following point is observed:

Evaluate the part of the receive area RD_1 currently being used completely before initiating another job.

PUT_S: Write data to a remote CPU (S7-300)

Description

With this instruction, you can write data to a remote CPU **with S7-300**. The connection can take place via the integrated interface as well as a CP.

The data is sent on a rising edge at REQ. The parameter values ID, ADDR_1 and SD_1 are adopted on each positive edge at REQ. After a job has been completed, you can assign new values to the ID, ADDR_1 and SD_1 parameters.

The remote partner saves the required data under the addresses supplied with the data and returns an execution acknowledgement. Make sure that the areas defined with the parameters ADDR_1 and SD_1 match in terms of number, length and data type.

If no errors occur, this is indicated at the next call with status parameter DONE = "1". The writing process can only be activated again after the last job is complete.

The remote CPU can be in RUN or STOP mode. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being written or if the execution check results in an error.

Parameters

The following table shows the parameters of the "PUT_S" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job executed without errors.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
ADDR_1	InOut	ANY	M, D	Pointers to the areas on the partner CPU to which the data will be written.
SD_1	InOut	ANY	M, D	Pointers to the areas on the local CPU which contain the data to be sent. Only the data types BOOL are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for "PUT_S" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.

ERROR	STATUS (decimal)	Explanation
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Maximum number of parallel jobs/instances exceeded.
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the send area pointers SD_1 relating to the data length or the data type.
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB)
1	12	When the instruction was called • An instance DB was specified that does not belong to "PUT_S" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	20	• Maximum number of parallel jobs/instances exceeded • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) • Possible when first called
1	27	There is no function code on the CPU for this instruction.

Data consistency

To ensure data consistency, send area SD_1 may not be written to again until the current send job has been completed. This is the case when the value of the status parameter DONE changes to "1".

USEND_S: Send data uncoordinated (S7-300)

Description

The instruction "USEND_S" sends data to a remote partner instruction of type "URCV_S (Page 6165)" **with S7-300**. The sending process is carried out without coordination with the partner instruction. This means that the data transfer is carried out without acknowledgement by the partner instruction.

The data is sent on a rising edge at REQ. The parameter values R_ID, ID and SD_1 are adopted on each positive edge at REQ . After a job has been completed, you can assign new values to the R_ID, ID and SD_1 parameters.

Make sure that the areas defined by the parameters SD_1 and RD_1 (for the associated partner instruction "URCV_S (Page 6165)") match in terms of:

- Number
- Length and
- Data type

The R_ID parameter must be identical in both instructions.

Successful completion of the send operation is indicated when the value of the DONE status parameter is set to logical "1".

Parameters

The following table shows the parameters of the "USEND_S" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 6134)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job has been executed error-free.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
SD_1	InOut	ANY	M, D	Pointer to the send area. Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS(decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Maximum number of parallel jobs/instances exceeded
1	4	Errors in the send area pointers SD_1 relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "USEND_S" • A global DB was specified instead of an instance DB • No instance DB found (Solution: Load the associated instance DB again).
1	18	• R_ID already exists in the connection ID. • The instances were overloaded at CPU-RUN (STOP-RUN transition of the CPU or CP required.)
1	20	• Maximum number of parallel jobs/instances exceeded • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) • Possible when first called
1	27	There is no function code on the CPU for this instruction.

Data consistency

To ensure data consistency, send area SD_1 may not be written to again until the current send job has been completed. This is the case when the value of the status parameter DONE changes to "1".

URCV_S: Receive data uncoordinated (S7-300)

Description

The instruction "URCV_S" receives data asynchronously from a remote partner instruction of type "USEND_S (Page 6163)" and copies this data into the configured receive areas **with S7-300**.

The instruction is ready to receive if there is a logical 1 at the EN_R input. An active job can be canceled with EN_R=0.

The parameter values R_ID, ID and RD_1 are adopted on each positive edge at EN_R. After a job has been completed, you can assign new values to the R_ID, ID and RD_1 parameters.

Make sure that the areas defined by the parameters RD_1 and SD_1 (for the associated partner instruction "USEND_S (Page 6163)") match in terms of:

- Number
- Length
- Data type

Successful completion of the copy operation is indicated when the value of the NDR status parameter is set to logical "1". The R_ID parameter must be identical in both instructions.

Parameters

The following table shows the parameters of the "URCV_S" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 6134)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still executing. • 1: Job completed without errors.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
RD_1	InOut	ANY	M, D	Pointer to the i-nth receive area: Only the data types BOOL are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS(decimal)	Explanation
0	9	Overrun warning: Older received data are overwritten by newer received data.
0	11	Warning: The received data is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Maximum number of parallel jobs/instances exceeded
1	4	Errors in the receive area pointers RD_1 relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "URCV_S" • A global DB instead of an instance DB was specified. • No instance DB found (Solution: Load the associated instance DB again).
1	18	• R_ID already exists in the connection ID. • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.)
1	19	The associated "USEND_S (Page 6163)" instruction sends data faster than this can be copied to the receive areas by "URCV_S".
1	20	• Maximum number of parallel jobs/instances exceeded • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) • Possible when first called
1	27	There is no function code in the CPU for this block.

Data consistency

The data is received in a consistent state if the following point is observed:

After the status parameter NDR has changed to "1", call "URCV_S" again immediately with the value "0" at EN_R. This ensures that the receive area is not already overwritten before you have evaluated it. Evaluate the receive area RD_1 completely before you call the block with value "1" at control input EN_R.

CONTROL: Determine connection status for the instance of an instruction (S7-400)

Description

With the "CONTROL" instruction, you can obtain the status of the connection belonging to a local communication instruction instance on an **S7-400**.

After the instruction is called with the value "1" at control input EN_R, the current status of the connection belonging to the communication instruction instance selected with I_DB is queried.

Parameters

The following table shows the parameters of the "CONTROL" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter, signals ready to receive if the input is set.
I_DB	Input	WORD	I, Q, M, D, L or constant	Number of the instance DB
OFFSET	Input	WORD	I, Q, M, D, L or constant	Offset of the data record in bytes in the multi-instance DB (if no multi-instance DB exists, "0" must be entered here).
RET_VAL	Return	INT	I, Q, M, D, L	Error information
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
I_TYP	Output	BYTE	I, Q, M, D, L	Identifier for the block type belonging to the selected instance.
I_STATE	Output	BYTE	I, Q, M, D, L	• = 0: The corresponding instruction instance has not been called since the last cold/warm restart or loading. • <> 0: The corresponding instruction instance has been called at least once since the last cold/warm restart or loading.
I_CONN	Output	BOOL	I, Q, M, D, L	Status of the corresponding connection. Possible values: • 0: Connection aborted or not yet established. • 1: Connection available.
I_STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS of the queried communication instruction instance.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Output parameter I_TYP

The following table lists the different instruction types and the corresponding identifiers

Instruction type	Identifier (W#16#...)
USEND (Page 6146)	00
URCV (Page 6149)	01
BSEND (Page 6151)	04
BRCV (Page 6154)	05
GET (Page 6141)	06
PUT (Page 6144)	07
PRINT (Page 6170)	08
START (Page 6176)	0B
STOP (Page 6178)	0C
RESUME (Page 6180)	0D
STATUS (Page 6181)	0E
USTATUS (Page 6184)	0F
ALARM (Page 5412)	15
ALARM_8 (Page 5404)	16
ALARM_8P (Page 5406)	17
NOTIFY (Page 5409)	18
AR_SEND (Page 5432)	19
NOTIFY_8P (Page 5402)	1A
(no instruction exists; I_DB or OFFSET wrong)	FF

Parameters RET_VAL, ERROR and STATUS

The output parameter RET_VAL can have the following two values for the "CONTROL" instruction:

- 0000H: No error occurred during execution.
- 8000H: An error occurred during execution.

ERROR	STATUS (decimal)	Explanation
1	10	Access to local user memory not possible (e.g.: A memory byte that does not exist in the CPU being used was specified as the actual parameter for I_TYP).
1	12	When the instruction was called: • An instance DB was specified at the I_DB parameter that does not belong to the "CONTROL" instruction. • A global DB was specified instead of an instance DB. • No instance DB found (Remedy: Load the associated instance DB again).

Note

The output parameters ERROR and STATUS need to be evaluated even if the output parameter RET_VAL shows the value 0000H .

PRINT: Send data to printer (S7-400)

Description of PRINT (S7-400)

Description

With the "PRINT" instruction, you can send data and a formatting instruction to a remote printer **with S7-400**, for example, using the CP 441.

When there is a rising edge at control input REQ , the format description (FORMAT parameter) and the data (SD_i) are sent to the printer selected with ID and PRN_NR .

If you do not use all four send areas, make sure that the first area is described by the SD_1 parameter, the second area (if it exists) by SD_2, the third area (if it exists) by SD_3.

Successful execution of the job is indicated by the status parameter DONE having the value "1". If any errors occur, they are indicated in the status parameters ERROR and STATUS.

Parameters

The following table shows the parameters of the "PRINT" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job has been executed error-free.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
PRN_NR	InOut	BYTE	I, Q, M, D, L	Printer number

Parameters	Declaration	Data type	Memory area	Description
FORMAT (Page 6172)	InOut	STRING	I, Q, M, D, L	Format description
SD_i (1≤ i ≤4)	InOut	ANY	M, D, T, C	Pointer to the "i-nth" send data area. Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P# DB10.DBX5.0 byte 10).

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for "PRINT" that can be output via the ERROR and STATUS parameters.

ERROR	STA-TUS(deci-mal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	2	Negative acknowledgement from printer. The function cannot be executed.
1	3	PRN_NR is unknown on the communication link specified by the ID.
1	4	Data length or data type related error in the FORMAT in/out parameter or in the send area pointers SD_i.
1	6	The remote printer is in OFFLINE mode.
1	7	The remote printer is not in the correct status (for example, paper out).
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called: • An instance DB was specified that does not belong to "PRINT" • A global DB instead of an instance DB was specified. • No instance DB found (Solution: Load the associated instance DB again).
1	13	Error in the FORMAT (Page 6172) in/out parameter.
1	20	**S7-400** Not enough work memory. Solution: Reduce the program code in the memory.

Amount of transferable data

The amount of data that can be transferred to a remote printer must not exceed a maximum length. This maximum data length is calculated as follows:

```
maxleng = 420 - format
```

In this case `format` is the current length of the FORMAT parameter in bytes. The data to be printed can be distributed over one or more send areas.

FORMAT parameter (S7-400)

Description

The character string FORMAT contains character and format elements that are to be printed. It has the following structure:

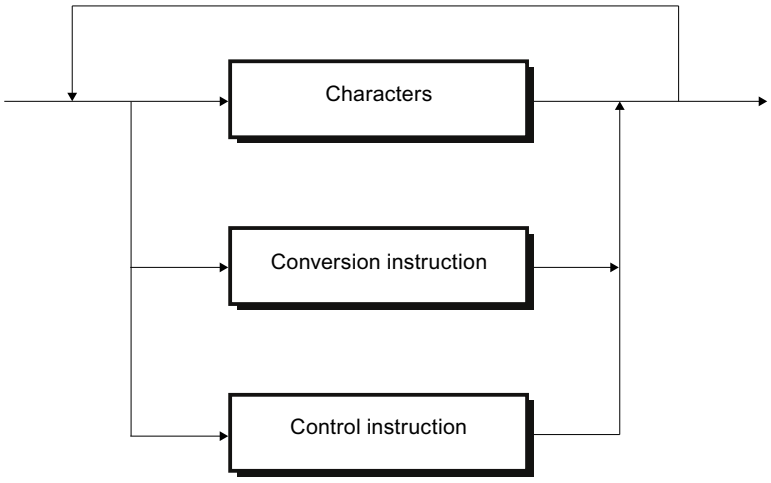

Figure 4-28 Structure of characters and format elements

Exactly one conversion instruction must exist in FORMAT for each send area SD_1 to SD_4 that is to be printed. The conversion instructions are used on the send areas SD_i according to order specified. Characters and instructions can follow each other in any order.

Characters

Permissible:

- All printable characters
- $$ (Dollar character), $' (single inverted comma),$L and $l (line feed), $P and $p (page), $R and $r (carriage return), $T and $t (tabulator)

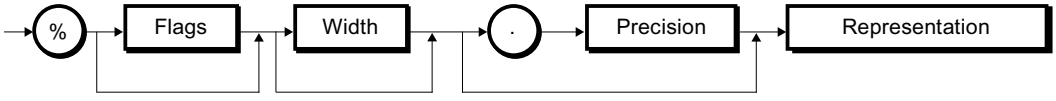

Figure 4-29 Syntax diagram of a conversion instruction

Element of a conversion instruction	Meaning	
Flags	• None: • -:	right-justified output left-justified output
Width	• None: • n:	output in standard representation Exactly n characters are output. If the output is right-justified, this may be preceded by blanks, with left- justified output the blanks come after the characters.
Precision	Precision is only relevant for representations A, D, F and R (see following table).	
	• None: • 0: • n:	Output is in standard representation no output of the decimal point or of decimal places for F and R representations • with F and R: output of the decimal point and n decimal places • with A and D (date): number of digits for the year: Possible values: 2 and 4.
Representation	The following table contains: • The possible representations • The data types possible for each representation • for each representation, the standard representation (the printer output is in standard representation if you specify no width and no precision in the FORMAT parameter.	

Conversion instruction

The following table shows the possible representations in the conversion instruction of the FORMAT parameter.

Representation	Possible data types	Example	Length	Comments
A, a	DATE	25.07.1996	10	-
	DWORD			
C, c	CHAR	K	1	-
	BYTE	M	1	
	WORD	KL	2	
	DWORD	KLMN	4	
	ARRAY of CHAR	KLMNOP	Number of characters	
	ARRAY of BYTE			
D, d	DATE	1996-07-25	10	-
	DWORD			
F, f	REAL	0.345678	8	-
	DWORD			
H, h	all data types incl. ARRAY of BYTE	Depending on data type	Depending on data type	Hexadecimal representation
I, i	INT	- 32 768	max. 6	-

Representation	Possible data types	Example	Length	Comments
	WORD	- 2 147 483 648	max. 11	
N, n	WORD	Text output	-	The relevant send area SD_i contains a reference (number) to a text to be printed. The text is on the module (for example, CP 441), which creates a printable string. If no text is found under the specified number, **** will be output.
R, r	REAL	0.12E-04	8	-
	DWORD			
S, s	STRING	Text output		-
T, t	TIME	2d_3h_10m_5s_250ms	max. 21	If an error occurs, **** will be output.
	DWORD			
U, u	BYTE	255	max. 3	-
	WORD	65 535	max. 5	
	DWORD	4 294 967 295	max. 10	
X, x	BOOL	1	1	-
	BYTE	101 ..	8	
	WORD	101 ..	16	
	DWORD	101 ..	32	
Z, z	TIME_OF_DAY (TOD)	15:38:59.874	12	-

At the points in this table at which a maximum length is specified for the representation, the actual length can of course be shorter.

Note

With the data types C and S, the following points depend on the printer being used:

- which characters can be printed
- what the printer prints for non-printable characters, unless the printer driver has a conversion table for these characters.

Control instruction

Using the control instruction you can do the following:

- Print the characters % and \
- Change the printer settings.

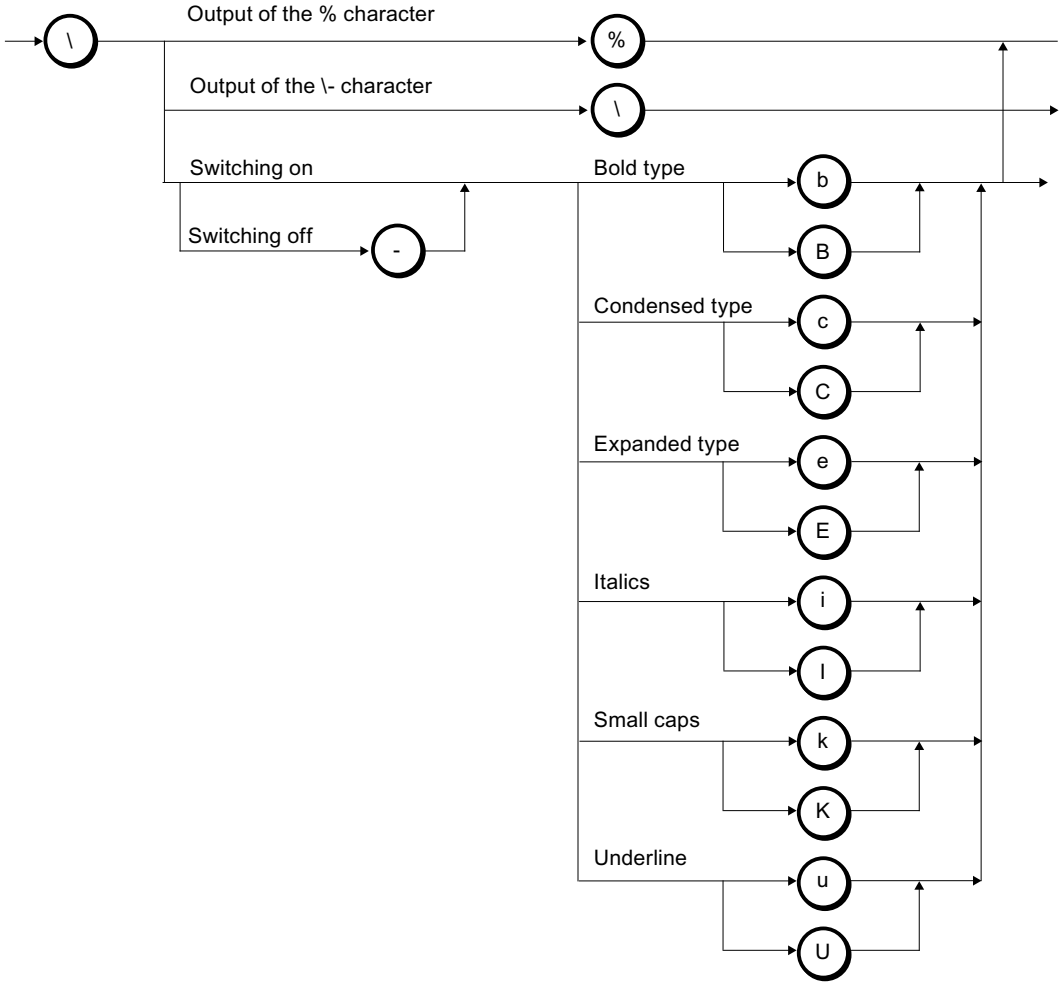

Figure 4-30 Syntax diagram of the control instruction

If you attempt to disable, for example, a font that is not enabled or execute a function that the printer does not recognize, the control instruction is ignored. The following table show the possible errors for the FORMAT in/out parameter.

Error	Printer output
Conversion instruction cannot be executed	* characters are output according to the (maximum) length of the default representation or the specified width.
Specified width too small	In the representations A, C, D, N, S, T, and Z, as many characters are printed as specified by the selected width. With all other representations, * characters are printed across the specified width.
Too many conversion instructions	The conversion instructions to which no send area pointer SD_i belongs are ignored.

Error	Printer output
Too few conversion instructions	Send areas for which there is no conversion instruction are not printed out.
Undefined or unsupported conversion instructions	****** is printed out.
Incomplete conversion instruction	****** is printed out.
Undefined or unsupported control instructions	Control instructions that do not comply with the syntax shown in the figure above are ignored.

START: Execute a warm restart or cold restart on a remote device (S7-400)

Description

If there is a positive edge at control input REQ, the "START" instruction activates a warm restart or a cold restart on the remote device addressed by ID **with S7-400**.

You can find additional information on the topic of warm and cold restarts here: Cold and warm restart (http://support.automation.siemens.com/WW/view/de/34053759)

The following conditions must be met if the remote device is a CPU:

- The CPU must be in STOP mode.
- The mode selector of the CPU must be set to "RUN".

Once the warm or cold restart is complete, the device changes to RUN mode and sends a positive execution acknowledgement. The status parameter DONE is set to "1" with the evaluation of the positive acknowledgement. The status parameters ERROR and STATUS show any errors that have occurred.

Another activation of a warm or cold restart is not possible until the last activation is completed.

Parameters

The following table shows the parameters of the "START" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the instruction on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without errors.

Parameters	Declaration	Data type	Memory area	Description
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
PI_NAME	InOut	ANY	I, Q, M, D, T, C	Pointer to memory area in which the name of the program (ASCII code) to be started is located. This name must not contain more than 32 characters. With a standard system from the S7 family, it must be P_PROGRAM.
ARG	InOut	ANY	I, Q, M, D, T, C	Execution argument. • If you do not assign a value to ARG , a warm restart is run on the remote device. • If you assign the value "C," a cold restart is run on the remote device (if the remote device is capable of this type of startup).
IO_STATE	InOut	BYTE	I, Q, M, D, L	Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 series automation system.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

ERROR and STATUS parameters

The following table contains all specific error information for START that can be output via the ERROR and STATUS parameters.

ERROR	STATUS(decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	3	The program name entered in PI_NAME is unknown.
1	4	Error in the pointers PI_NAME or ARG relating to the data length or the data type.
1	7	No complete restart possible on the partner device.

ERROR	STATUS(decimal)	Explanation
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "START" • A global DB was specified instead of an instance DB • No instance DB found (Solution: Load the associated instance DB again).
1	20	S7-400: Not enough work memory. Solution: Reduce the program code in the memory.

STOP: Change a remote device to STOP mode (S7-400)

Description

If there is a rising edge at control input REQ , the "STOP" instruction activates a change to STOP mode on the remote device addressed by ID**with S7-400**. The mode change is possible when the device is in RUN, HOLD, or startup.

Successful execution of the job is indicated by the status parameter DONE having the value "1". If any errors occur, they are indicated in the status parameters ERROR and STATUS.

The mode change can only be repeated on the same remote device if the previous "STOP" instruction has been completed.

Parameters

The following table shows the parameters of the "STOP" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the instruction on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job has been executed error-free.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 There is an error. STATUS supplies detailed information on the type of error.

Parameters	Declaration	Data type	Memory area	Description
PI_NAME	InOut	ANY	I, Q, M, D	Pointer to the memory area in which the name of the program (ASCII code) to be started is located. This name must not contain more than 32 characters. With a standard system from the S7 family, it must be P_PROGRAM.
IO_STATE	InOut	BYTE	I, Q, M, D, L	Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 series automation system.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "STOP" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	3	The program name entered in PI_NAME is unknown.
1	4	Error in the pointer PI_NAME relating to the data length or the data type.
1	7	The partner device is already in the STOP state.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to STOP • A global DB was specified instead of an instance DB • No instance DB found (Solution: Load the associated instance DB again).
1	20	S7-400: Not enough work memory. Solution: Reduce the program code in the memory.

RESUME: Initiate a hot restart on a remote device (S7-400)

Description

If there is a rising edge at control input REQ, the "RESUME" instruction activates a hot restart in the remote device addressed by ID**with S7-400**. The following conditions must be present if the remote device is a CPU:

- The CPU must be in STOP mode.
- The mode selector of the CPU must be set to RUN.
- You must have enabled manual hot restart during configuration.
- There must be no condition preventing a hot restart.

Once the hot restart has been completed, the device changes to RUN mode and sends a positive execution acknowledgement. The status parameter DONE is set to "1" with the evaluation of the positive acknowledgement. The status parameters ERROR and STATUS show any errors that have occurred.

Another activation of the hot restart in the same remote device is not possible until the last activation is complete.

Parameters

The following table shows the parameters of the "RESUME" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the instruction on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job has been executed error-free.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
PI_NAME	InOut	ANY	I, Q, M, D	Pointer to memory area in which the name of the program (ASCII code) to be started is located. This name must not contain more than 32 characters. With an S7, it must be P_PROGRAM.

Parameters	Declaration	Data type	Memory area	Description
ARG	InOut	ANY	I, Q, M, D, T, C	Execution argument. Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 series automation system.
IO_STATE	InOut	BYTE	I, Q, M, D, L	Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 series automation system.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

The following table contains all specific error information for "RESUME" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	• Communications problems, for example, connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	3	The program name entered in PI_NAME is unknown.
1	4	Error in the pointers PI_NAME or ARG relating to the data length or the data type.
1	7	Hot restart not possible
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to RESUME. • A global DB was specified instead of an instance DB • No instance DB found (Solution: Load the associated instance DB again).
1	20	S7-400: Not enough work memory. Solution: Reduce the program code in the memory.

STATUS: Query the status of a remote partner (S7-400)

Description

With the "STATUS" instruction, you can query the device status of a remote communication partner **with S7-400**.

If there is a rising edge at control input REQ, a job is sent to the remote partner. The reply is evaluated to determine whether problems have occurred. If no errors occurred, the received

status is copied to the tags of the PHYS, LOG and LOCAL parameters with the next call. Completion of this action is indicated by the status parameter NDR having the value "1".

You can only query the status of the same communications partner again after the last query is complete.

Parameters

The following table shows the parameters of the "STATUS" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the instruction on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still executing. • 1: Job completed without errors.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 There is an error. STATUS supplies detailed information on the type of error.
PHYS	InOut	ANY	I, Q, M, D	Physical status (minimum length: one byte) Possible values: • 10H fully functional. • 13H service required.
LOG	InOut	ANY	I, Q, M, D	Logical status (minimum length: one byte) Possible value: • 00H status change permitted.
LOCAL	InOut	ANY	I, Q, M, D	Status if the partner device is an S7 CPU (minimum length: two bytes)

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

In/out parameter LOCAL

If the communications partner is an S7 CPU, the in/out parameter LOCAL contains its current status: The first byte is reserved, the second byte contains an ID for the status.

Operating mode	Corresponding identifier
STOP	00H
STARTUP (warm restart)	01H
RUN	02H
STARTUP (hot restart)	03H
HOLD	04H
STARTUP (cold restart)	06H
RUN	09H
LINK-UP	0BH
UPDATE	0CH

Parameters ERROR and STATUS

The following table contains all specific error information for STATUS that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Error in PHYS, LOG or LOCAL relating to the data length or the data type.
1	8	Access to a remote object was rejected.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to STATUS • A global DB was specified instead of an instance DB • No instance DB found (Solution: Load the associated instance DB again).
1	20	S7-400: Not enough work memory. Solution: Reduce the program code in the memory.

USTATUS: Uncoordinated receiving of remote device status (S7-400)

Description

The "USTATUS" instruction receives the device status change of a remote communication partner **with S7-400**. The partner sends its status unsolicited when a change occurs if you have configured this behavior.

If EN_R 1 is set at the control input during the call and a frame of the partner is pending, the status information is entered in the tags of the PHYS, LOG and LOCAL parameters at the next call. Completion of this action is indicated by the status parameter NDR having the value "1".

The transfer of the operating status messages must be enabled on the connection used by the "USTATUS" instruction.

Note

You can only place one instance of the "USTATUS" instruction per connection.

Parameters

The following table shows the parameters of the "USTATUS" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter (enabled to receive), signals ready to receive if the input is set.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 6134)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still executing. • 1: Job successfully completed.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
PHYS	InOut	ANY	I, Q, M, D	Physical status (minimum length: one byte) Possible values: • 10H fully functional. • 13H service required.

Parameter	Declaration	Data type	Memory area	Description
LOG	InOut	ANY	I, Q, M, D	Logical status (minimum length: one byte) Possible value: • 00H status change permitted.
LOCAL	InOut	ANY	I, Q, M, D	Status if the partner device is an S7 CPU (minimum length: two bytes)

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

In/out parameter LOCAL

If the communication partner is an S7 CPU, the in/out parameter LOCAL contains its current status: The first byte is reserved, the second byte contains an ID for the status.

Operating mode	Corresponding identifier
STOP	00H
STARTUP (warm restart)	01H
RUN	02H
STARTUP (hot restart)	03H
HOLD	04H
STARTUP (cold restart)	06H
RUN	09H
LINK-UP	0BH
UPDATE	0CH

Parameters ERROR and STATUS

The following table contains all specific error information for "USTATUS" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	9	Overrun warning: an older device status has been overwritten by a more recent device status.
0	11	The received data is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	4	Error in PHYS, LOG or LOCAL relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "USTATUS" • A global DB was specified instead of an instance DB. • No instance DB found (Remedy: Load the associated instance DB again).

ERROR	STATUS (decimal)	Explanation
1	18	There is already an instance for "USTATUS" for the connection identified by ID.
1	19	The remote CPU sends data faster than it can be accepted in the user program by the instruction.
1	20	S7-400: Not enough work memory. Remedy: Reduce the program code in the memory.

4.2.5.3 Open User Communication

Overview

Open User Communication via Industrial Ethernet

The following instructions and UDTs are available for exchanging data with other communication partners by means of the user program via Ethernet (TCP, ISO-on-TCP, UDP protocols):

- Connection-oriented protocols: TCP as per RFC 793, ISO on TCP as per RFC 1006:
  - UDT 65 "TCON_PAR (Page 6196)" with the data structure for assigning connection parameters
  - UDT 651 to UDT 656 with protocol-specific defaults
  - "TCON (Page 6208)" for establishing the connection
  - "TDISCON (Page 6212)" for terminating the connection
  - "TSEND (Page 6213)" for sending data
  - "TRCV (Page 6216)" for receiving data
- Connectionless protocol: UDP as per RFC 768
  - UDT 65 "TCON_PAR (Page 6196)" with the data structure for assigning parameters of the local communications access point
  - UDT 657 with protocol-specific defaults
  - UDT 66 "TCON_ADR (Page 6198)" with the data structure for assigning addressing parameters of the remote partner
  - UDT 661 with protocol-specific defaults
  - "TCON (Page 6208)" for setting up the local communication access point
  - "TDISCON (Page 6212)" for canceling the local communication access point
  - "TUSEND (Page 6220)" for sending data
  - "TURCV (Page 6223)" for receiving data

Functional description of instructions for Open User Communication via Industrial Ethernet

Connection-oriented and connectionless protocols

The following protocol variants are distinguished in the data communication:

- Connection-oriented protocol variants:
  They establish a logical connection to the communication partner before data transmission is started. After the data transmission is complete, they then terminate the connection, if necessary. Connection-oriented protocols are used for data transmission when reliable, guaranteed delivery is of particular importance. Usually, there can be several logical connections over one physical line.
  With the instructions for Open User Communication via Industrial Ethernet, the following connection-oriented protocols are supported:
  - TCP as per RFC 793
  - ISO on TCP as per RFC 1006
- Connectionless protocol variants:
  They work without a connection. There is thus no establishment and termination of a connection with a remote partner. Unacknowledged connectionless protocols transmit the data to the remote partner and are therefore unprotected. This means the data can be lost without any indication at the block.
  With the instructions for Open User Communication via Industrial Ethernet, the following connectionless protocols are supported:
  - UDP per RFC 768 (broadcast or unicast, no support for multicast)

The functional description of the instructions depends on the protocol variant being used. This is discussed in more detail in the following section.

Receive area

This term will be used repeatedly in the following sections. This means the area in which the instruction enters the received data.

The receive area is specified by the following two variables:

- Pointer to the start of the area
- Length of the area

Depending on the protocol variant being used, the length of the area is specified either by the LEN parameter (if LEN <> 0) or the length information of the DATA parameter (if LEN = 0).

The following data types are permitted in the ANY pointer: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME.

TCP

During data transmission, no information about the length or about the start and end of a message is transmitted. This is not a problem during sending because the sender knows how many data bytes it will be sending. However, the receiver has no means of detecting where one message ends in the data stream and the next one begins. We therefore recommend that you select the size of receive area of the "TRCV (Page 6216)" instruction to match the value of the LEN parameter of the "TSEND (Page 6213)" instruction on the communication partner (number of bytes to be sent).

- Date reception in ad-hoc mode:
  The receive area is identical with the area specified by the DATA parameter of the "TRCV (Page 6216)" instruction.
  - Immediately after receiving a block of data, the "TRCV (Page 6216)" instruction transfers this data block to the receive area and sets NDR to "1". The maximum length is 8192 bytes.
  - If you have selected a receive area that is longer than the sent data, the "TRCV (Page 6216)" instruction copies all received data completely in the receive area. It then sets NDR to TRUE and writes the length of the sent data to RCVD_LEN.
  - If you have selected a receive area that is smaller than the sent data, the "TRCV (Page 6216)" instruction copies as many bytes into the receive area until it is full. It then sets NDR to TRUE and writes the length of the receive area to RCVD_LEN. With each subsequent call, you receive a further block of the sent data.
- Data reception with specified length:
  - The receive area is formed by the DATA parameters (start address of the receive area) and LEN (length of the receive area) by "TRCV (Page 6216)".
  - If data are received that do not completely fill the receive area, then these data will not be made available to you initially. They will only become available when additional data has filled up the receive area completely. Please note that in this case data from two different send jobs are located in the same receive area. If you cannot determine the end of the first message or the start of the second message, you will not be able to recognize the first or the second message.
  - If you have selected a receive area that is smaller than the sent data, the "TRCV (Page 6216)" instruction copies as many bytes into the receive area until it is completely full.
  - If "TRCV (Page 6216)" has completely filled the receive area, it will set NDR to TRUE and describes RCVD_LEN with the value of LEN. Thus with each subsequent call, you get an additional block of the sent data.

ISO on TCP

During data transfer, information on the length and end of a message is also transferred.

If you have selected a receive area that is longer than the sent data, then the "TRCV (Page 6216)" instruction enters all the sent data completely in the receive area. It then sets NDR to TRUE and writes the length of the sent data to RCVD_LEN.

If you have selected a receive area shorter than the length of the sent data, then "TRCV (Page 6216)" will not copy any data to the receive area and returns the following error information: ERROR=1, STATUS=W#16#8088.

UDP

There is no connection establishment as is the case with the TCP and ISO on TCP protocols. You must therefore specify a reference to the address parameters of the recipient (IP address and port no.) when you call the send block "TUSEND (Page 6220)". In the same way, on completion of the receive block "TURCV (Page 6223)", you will receive a reference to the address parameters of the sender (IP address and port no.).

- To use the "TUSEND (Page 6220)" and "TURCV (Page 6223)" instructions, you must first call the "TCON (Page 6208)" instruction both at the sender and receiver end to set up the local communication access point.
- Each time you call "TUSEND (Page 6220)", you can reference the remote partner once again by specifying its IP address and its port number.
- During data transfer, information on the length and end of a message is also transferred.

If you have selected a receive area that is longer than the sent data, "TURCV (Page 6223)" will copy all sent data completely to the receive area. It then sets NDR to TRUE and writes the length of the sent data to RCVD_LEN.

If you have selected a receive area shorter than the length of the sent data, TURCV does not copy any data to the receive area and returns the following error information: ERROR = 1, STATUS = W#16#8088.

Port numbers for TCP and UDP

The address assignment for a connection is made as follows:

- With TCP:
  - Active connection: Via the remote IP address, the remote port and the local port (specify the port number of the local port explicitly or let the operating system set it).
  - Passive connection: Via the local port. If you want to use the local port several times, you have to either pre-define the remote IP address or the remote IP address together with the remote port.
- With UDP: Via the remote IP address, the remote port and the local port (specify the port number of the local port explicitly).

The areas listed below apply to the local port. These areas apply to the remote port of the remote partner is an S7 CPU. Otherwise, there are no restrictions for the remote port.

- For CPUs 31x-2 PN/DP up to and including firmware version V2.6 and CPUs 41x-3 PN/DP up to and including firmware version V5.1:
  - Permissible port numbers in the configuration (UDT 65): 2000 to 5000
  - The operating system assigns a port from the dynamic range of numbers between 49152 and 65534 if a port number is not assigned explicitly.
- For CPUs 31x-2 PN/DP as of firmware version V2.7 and CPUs 41x-3 PN/DP as of firmware version V5.2:
  - Permissible port numbers in the configuration (UDT 65): 1 to 49151
    We recommend restricting the port numbers from 2000 to 5000, since Siemens reserves ports from the number ranges 1 to 1999 and 5001 to 49151 for use by the system.
  - The operating system assigns a port from the dynamic range of numbers between 49152 and 65534 if a port number is not assigned explicitly.
- For CPUs 31x-2 PN/DP as of firmware version V3.2, CPUs 41x-3 PN/DP as of firmware version V6.0 and CPUs 41x-5H PN/DP as of firmware version V6.0:
  - A port can be used several times.

You can find the port numbers reserved by the system in the following list:

Protocol	Port number	Service
-	0	(This port is reserved. It cannot be used.)
TCP	20 and 21	FTP
TCP	25	SMTP
TCP	80	HTTP
TCP	102	RFC 1006
UDP	135	RPC-DCOM
UDP	161	SNMP REQUEST
TCP, UDP	34962	PNIO
TCP, UDP	34963	PNIO
TCP, UDP	34964	PNIO
UDP	65532	NTP
UDP	65533	NTP
UDP	65534	NTP
UDP	65535	NTP

Also see http://www.iana.org/assignments/port-numbers.

Note

Do not use port numbers reserved for the system.

Identical port numbers must not be used for active and passive connection establishment in a device.

Assigning parameters for communications connections with TCP and ISO on TCP

Data block for assigning parameters

To be able to assign parameters for TCP and ISO on TCP for communication connections, create a DB that contains the data structure from the UDT 65 "TCON_PAR". This data structure contains the parameters necessary for configuring the connection. You will need such a data structure for every connection. You can assemble this structure in a shared DB.

The CONNECT connection parameter of the "TCON (Page 6208)" instruction contains a reference to the address of the associated connection description (e.g. P#DB100.DBX0.0 byte 64).

Form of the connection description (UDT 65)

Byte	Parameters	Data type	Start value	Description
0 to 1	block_length	WORD	W#16#40	Length of UDT 65: 64 bytes (fixed)
2 to 3	id	WORD	W#16#0001	Reference to the connection (range of values: W#16#0001 to W#16#0FFF) You must specify the value of this parameter in the respective block for the ID.
4	connection_type	BYTE	B#16#11	Protocol variant: • B#16#11: TCP • B#16#12: ISO on TCP • B#16#01: TCP (compatibility mode)
5	active_est	BOOL	FALSE	ID for the manner in which the connection is established: • FALSE: Passive connection establishment • TRUE: Active connection establishment

Byte	Parameters	Data type	Start value	Description
6	local_device_id	BYTE	B#16#02	• B#16#00: Communication via CP 443-1EX (only with S7-400 and connection_type = B#16#12) Permitted CPs: CP443-1EX4x, CP443-1EX20, CP443-1GX20, CP443-1EX30, CP443-1GX30 • B#16#01: Communication via IE interface on interface slot 1 (IF1) for WinAC RTX (only TCP) • B#16#02: Communication over the integrated IE interface with CPUs 315-2 PN/DP and 317-2 PN/DP • B#16#03: Communication via the integrated IE interface for CPU 319-3 PN/DP • B#16#05: Communication via the integrated IE interface with CPUs 414-3 PN/DP, 416-3 PN/DP, 416-3F PN/DP and 41x-5H PN/DP (Rack 0) • B#16#06: Communication via IE interface on interface slot 2 (IF2) for WinAC RTX (only TCP) • B#16#0B: Communication via IE interface on interface slot 3 (IF3) for WinAC RTX (only TCP) • B#16#0F: Communication via IE interface on interface slot 4 (IF4) for WinAC RTX (only TCP) • B#16#15: Communication via the integrated IE interface for CPUs 41x-5H PN/DP (Rack 1)
7	local_tsap_id_len	BYTE	B#16#02	Length of parameter local_tsap_id used; possible values: • 0 or 2, for connection_type = B#16#01 (active end: 0, passive end: 2) • 0 or 2, for connection_type = B#16#11 (active end: 0 or 2, passive end: 2) • 2 to 16. if connection_type = B#16#12
8	rem_subnet_id_len	BYTE	B#16#00	This parameter is currently not used. Assign with B#16#00.
9	rem_staddr_len	BYTE	B#16#00	Length of the address of the remote connection end point: • 0: unspecified, i.e. parameter rem_staddr is irrelevant. • 4: Valid IP address in the rem_staddr parameter
10	rem_tsap_id_len	BYTE	B#16#00	Length of parameter rem_tsap_id used; possible values: • 0 or 2, for connection_type = B#16#01 For the passive end only value B#16#00 is permitted. • 0 or 2. if connection_type = B#16#11 • 0 or 2 to 16, if connection_type = B#16#12 (active end: 2 to 16; passive end: 0 or 2 to 16, 0 means unspecified)
11	next_staddr_len	BYTE	B#16#00	length of the next_staddr parameter used (only relevant for ISO-on-TCP)

Byte	Parameters	Data type	Start value	Description
12 to 27	local_tsap_id	ARRAY [1..16] of BYTE	B#16#07 B#16#D0 B#16#00 ...	With connection_type = • B#16#11: Local port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 6187)), local_tsap_id[1] = high byte of the port no. in hexadecimal, local_tsap_id[2] = low byte of the port number in hexadecimal, local_tsap_id[3-16] = B#16#00 • B#16#12: local TSAP ID: see below. • B#16#01: Local port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 6187)), local_tsap_id[1] = low byte of the port no. in hexadecimal, local_tsap_id[2] = high byte of the port number in hexadecimal, local_tsap_id[3-16] = B#16#00 Note: If there are multiple connections to the same communication partner, you have to ensure that each value of local_tsap_id that you use in your CPU is unique.
28 to 33	rem_subnet_id	ARRAY [1..6] of BYTE	B#16#00 ...	This parameter is currently not used. Assign 0 to it.
34 to 39	rem_staddr	ARRAY [1..6] of BYTE	B#16#00 ...	IP address of the remote connection end point, for example, 192.168.0.1: With connection_type = • B#16#1x: rem_staddr[1] = B#16#C0 (192), rem_staddr[2] = B#16#A8 (168), rem_staddr[3] = B#16#00 (0), rem_staddr[4] = B#16#01 (1), rem_staddr[5-6]= B#16#00 (reserved) • B#16#01: rem_staddr[1] = B#16#01 (1), rem_staddr[2] = B#16#00 (0), rem_staddr[3] = B#16#A8 (168), rem_staddr[4] = B#16#C0 (192), rem_staddr[5-6]= B#16#00 (reserved)

Byte	Parameters	Data type	Start value	Description
40 to 55	rem_tsap_id	ARRAY [1..16] of BYTE	B#16#00 ...	With connection_type = • B#16#11: remote port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 6187)), rem_tsap_id[1] = high byte of the port no. in hexadecimal, rem_tsap_id[2] = low byte of the port number in hexadecimal, rem_tsap_id[3-16] = B#16#00 • B#16#12: remote TSAP ID: see below. • B#16#01: remote port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 6187)), local_tsap_id[1] = low byte of the port no. in hexadecimal, local_tsap_id[2] = high byte of the port number in hexadecimal, local_tsap_id[3-16] = B#16#00
56 to 61	next_staddr	ARRAY [1..6] of BYTE	B#16#00 ...	With local_device_id = • B#16#00: next_staddr[1]: Rack and slot of the associated (local) CP (Bits 0 to 4: slot, bits 5 to 7: rack number) next_staddr[2-6]: B#16#00 • B#16#02, B#16#03, B#16#05: next_staddr[1-6]: B#16#00 • B#16#01, B#16#06, B#16#0B, B#16#0F: next_staddr[1-6]: B#16#00 (In this case, next_staddr_len=B#16#00)
62 to 63	spare	WORD	W#16#0000	Standby: Assign "0" to this parameter.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

Length and form of the local and remote TSAP ID for ISO on TCP

- Active connection establishment:
  - Remote TSAP-ID:
    All character strings with a length between 1 and 16 bytes are permitted. Each character can have a value between B#16#00 and B#16#FF.
  - Remote TSAP-ID:
    For integrated IE interfaces, all character strings with a length between 1 and 16 bytes are permitted. Each character can have a value between B#16#00 and B#16#FF.
- Passive connection establishment:
  - Remote TSAP-ID:
    All character strings with a length between 0 and 16 bytes are permitted. Each character can have a value between B#16#00 and B#16#FF.
  - Local TSAP-ID:

loc_tsap_id_len	local_tsap_id[1]	local_tsap_id[2]	local_tsap_id[3 bis 16]
2	B#16#E0 (connection type T connection)	0 (only for integrated IE interfaces) or rack and slot of the local CPU (bits 0 to 4 slot, bits 5 to 7 rack number)	Does not exist
> 2	B#16#E0 (connection type T connection)	0 (only for integrated IE interfaces) or rack and slot of the local CPU (bits 0 to 4 slot, bits 5 to 7 rack number)	TSAP extension
	Only for integrated IE interfaces: a ASCII character (B#16#20 to B#16#7E)	irrelevant	TSAP extension

CPU dependencies for protocol variants TCP and ISO on TCP

To determine which protocol variants (TCP and ISO on TCP) can be used on which CPU, refer to the following paragraph:

Relationship between CPU and protocol variant used (connection_type) and transferable data length (Page 6199).

For information on the number of possible connections, please refer to the specifications for your CPU.

Connection establishment

A communication partner A must trigger the active connection establishment. A communication partner B must trigger the passive connection establishment. If both communication partners have triggered their connection establishment then the operating system can completely establish the communication connection.

When assigning parameters to the connection specify which communication partner will activate the connection establishment and which will execute a passive communication establishment on the request of the communication partner.

In UDP, both partners must initiate passive connection establishment.

Assigning parameters for the local communications access point with UDP

Data structure for configuring the local communications access point

To assign parameters for the local communications access point, create a DB that contains the data structure from the UDT 65 "TCON_PAR". This data structure contains the parameters necessary for configuring the connection between the user program and the communications layer of the operating system.

The CONNECT parameter of the "TCON (Page 6208)" instruction contains a reference to the address of the associated connection description (e.g. P#DB100.DBX0.0 byte 64).

Form of the connection description for UDP (UDT 65)

Byte	Parameters	Data type	Initial value	Description
0 to 1	block_length	WORD	W#16#40	Length of UDT 65: 64 bytes (fixed)
2 to 3	id	WORD	W#16#0001	Reference to this connection between the user program and the communications level of the operating system (range of values: W#16#0001 to W#16#0FFF) You must specify the value of this parameter in the respective block for the ID.
4	connection_type	BYTE	B#16#13	Protocol variant: • B#16#13: UDP
5	active_est	BOOL	FALSE	ID for the manner in which the connection is established: You must assign FALSE to this parameter.
6	local_device_id	BYTE	B#16#02	• B#16#01: Communication via IE interface on interface slot 1 (IF1) for WinAC RTX • B#16#02: Communication via the integrated IE interface for CPU 317-2 PN/DP • B#16#03: Communication via the integrated IE interface for CPU 319-3 PN/DP • B#16#05: Communication via the integrated IE interface with CPUs 414-3 PN/DP, 416-3 PN/DP, 416-3F PN/DP and 41x-5H PN/DP (Rack 0) • B#16#06: Communication via IE interface on interface slot 2 (IF2) for WinAC RTX • B#16#0B: Communication via IE interface on interface slot 3 (IF3) for WinAC RTX • B#16#0F: Communication via IE interface on interface slot 4 (IF4) for WinAC RTX • B#16#15: Communication via the integrated IE interface for CPUs 41x-5H PN/DP (Rack 1)

Byte	Parameters	Data type	Initial value	Description
7	local_tsap_id_len	BYTE	B#16#02	Length of local_tsap_id parameter (local port) used; 2 bytes
8	rem_subnet_id_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
9	rem_staddr_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
10	rem_tsap_id_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
11	next_staddr_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
12 to 27	local_tsap_id	ARRAY [1..16] of BYTE	B#16#07 B#16#D0 B#16#00 ...	Local port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 6187)), local_tsap_id[1] = high byte of the port no. in hexadecimal, local_tsap_id[2] = low byte of the port number in hexadecimal, local_tsap_id[3-16] = B#16#00 (reserved) Note: Ensure that each value of local_tsap_id that you use in your CPU is unique.
28 to 33	rem_subnet_id	ARRAY [1..6] of BYTE	B#16#00 ...	This parameter is not used. Assign "0" to it.
34 to 39	rem_staddr	ARRAY [1..6] of BYTE	B#16#00 ...	This parameter is not used. Assign "0" to it.
40 to 55	rem_tsap_id	ARRAY [1..16] of BYTE	B#16#00 ...	This parameter is not used. Assign "0" to it.
56 to 61	next_staddr	ARRAY [1..6] of BYTE	B#16#00 ...	This parameter is not used. Assign "0" to it.
62 to 63	spare	WORD	W#16#0000	Standby: Assign "0" to this parameter.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

CPU dependency of the UDP protocol variant

To determine which CPUs are suitable for use with the UDP protocol variant, refer to the following paragraph:

Relationship between CPU and protocol variant used (connection_type) and transferable data length (Page 6199)

For information on the number of possible connections between the user program and the communications layer of the operating system, please refer to the specifications for your CPU.

Configuring the local communications access point

Each communications partner must configure its local communications point independently of the other partner. This pertains to establishing the connection between the user program and communications layer of the operating system.

In UDP, both partners must initiate passive connection establishment.

Structure of the address information for the remote partner with UDP

Overview

- With "TUSEND (Page 6220)", you transfer the address information of the receiver at the ADDR parameter. This address information must have the structure specified below.
- With "TURCV (Page 6216)", you receive the address of the sender of the received data at the ADDRparameter. This address information must have the structure specified below.

Data block for the address information of the remote partner

You must create a DB that contains one or more data structures as per UDT 66 "TADDR_PAR".

In parameter ADDR of "TUSEND (Page 6220)", you transfer a pointer to the address of the associated remote partner (e.g. P#DB100.DBX0.0 byte 8). This pointer is received in the ADDRparameter of "TURCV (Page 6216)".

Form of the address information for the remote partner (UDT 66)

Byte	Parameter	Data type	Initial value	Description
0 to 3	rem_ip_addr	ARRAY [1..4] of BYTE	B#16#00 ...	IP address of the remote partner, e.g. 192.168.2.3: • rem_ip_addr[1] = B#16#C0 (192) • rem_ip_addr[2] = B#16#A8 (168) • rem_ip_addr[3] = B#16#02 (2) • rem_ip_addr[4] = B#16#03 (3)
4 to 5	rem_port_nr	ARRAY [1..2] of BYTE	B#16#00 ...	remote port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 6187)) • rem_port_nr[1] = high byte of the port no. in hexadecimal notation • rem_port_nr[2] = low byte of port no. in hexadecimal notation
6 to 7	spare	ARRAY [1..2] of BYTE	B#16#00 ...	Standby: Assign "0" to this parameter.

Relationship between CPU and protocol variant used (connection_type) and transferable data length

Relationship between CPU and protocol variant used (connection_type) and transferable data length

The table below shows which protocol variants you can use on which CPU, and which data lengths you can transfer.

Protocol variant	Parameter "connection _type" in UDT 65	CPU	Data length in bytes for communication over integrated IE interface	Data length in bytes for communication via CP
TCP	B#16#11	CPUs 31x-2 PN/DP as of firmware version V2.4	1 to 8192	-
		CPUs 31x-2 PN/DP as of firmware version V3.1	1 to 32767	
		CPUs 31x-3 PN/DP as of firmware version V3.2	1 to 32767	
		CPUs 41x-3 PN/DP	1 to 32767	-
		CPUs 41x-5H PN/DP	1 to 32767	-
	B#16#01	CPUs 31x-2 PN/DP	1 to 1472	-
ISO on TCP	B#16#12	CPUs 31x-2 PN/DP as of firmware version V2.4	1 to 8192	-
		CPUs 31x-2 PN/DP as of firmware version V3.1	1 to 32767	
		CPUs 31x-3 PN/DP as of firmware version V3.2	1 to 32767	
		CPUs 41x as of firmware version V4.1 (without CPUs 41x-4H)	-	1 to 1452
		CPUs 41x-3 PN/DP	1 to 32767	1 to 1452
		CPUs 41x-5H PN/DP	1 to 32767	1 to 1452
UDP	B#16#13	CPUs 31x-2 PN/DP as of firmware version V2.4, CPUs 41x-3 PN/DP and 41x-5H PN/DP	1 to 1472	-

Examples of parameters for communications connections

Example 1: Communication via ISO on TCP and CP 443-1 between two S7-400 CPUs

The two communication partners are two CPUs 414-2. The communication is handled by two CPs 443-1.

The following table shows the most important data for both communications partners:

Property	Communication partner A: CPU 414-2 with CP 443-1	Communication partner B: CPU 414-2 with CP 443-1
Connection establishment	active	passive
IP address	192.168.4.14	192.168.4.16
Physical address of the CPU	Rack 0, slot 3	Rack 0, slot 4
Physical address of the associated CP	Rack 0, slot 6	Rack 1, slot 8
Local TSAP-ID (Note: the coding of the actual TSAP to distinguish the connection occurs as of the third byte)	0xE0 03 54 43 50 2D 31	0xE0 04 54 43 50 2D 31

The following table shows the parameter entries in the DB relevant for active establishment of a connection by communication partner A:

Parameters	Data type	Value in example	Description
id	WORD	W#16#0414	Reference to this connection
connection_type	BYTE	B#16#12	Protocol variant: ISO on TCP
active_est	BOOL	TRUE	Active connection establishment
local_device_id	BYTE	B#16#00	Communication AS-internal via CP
local_tsap_id_len	BYTE	B#16#07	Length of parameter local_tsap_id
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#07	Length of parameter rem_tsap_id
next_staddr_len	BYTE	B#16#01	Length of parameter next_staddr
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#E0 • local_tsap_id[2] = B#16#03 • local_tsap_id[3] = B#16#54 (ASCII equivalent of "T") • local_tsap_id[4] = B#16#43 (ASCII equivalent of "C") • local_tsap_id[5] = B#16#50 (ASCII equivalent of "P") • local_tsap_id[6] = B#16#2D (ASCII equivalent of "-") • local_tsap_id[7] = B#16#31 (ASCII equivalent of "1") • local_tsap_id[8-16] = B#16#00	local TSAP-ID: 0xE0035443502D31

Parameters	Data type	Value in example	Description
rem_staddr	ARRAY [1..6] of BYTE	"192.168.4.16" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#04 (4) • rem_staddr[4] = B#16#10 (16) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point
rem_tsap_id	ARRAY [1..16] of BYTE	• rem_tsap_id[1] = B#16#E0 • rem_tsap_id[2] = B#16#04 • rem_tsap_id[3] = B#16#54 (ASCII equivalent of "T") • rem_tsap_id[4] = B#16#43 (ASCII equivalent of "C") • rem_tsap_id[5] = B#16#50 (ASCII equivalent of "P") • rem_tsap_id[6] = B#16#2D (ASCII equivalent of "-") • rem_tsap_id[7] = B#16#31 (ASCII equivalent of "1") • rem_tsap_id[8-16] = B#16#00	remote TSAP-ID: 0xE0045443502D31
next_staddr	ARRAY [1..6] of BYTE	• next_staddr[1] = B#16#06 • next_staddr[2-6] = B#16#00	Rack = 0, slot = 6 (bits 7 to 5: Rack no., bits 4 to 0: slot no.)

The following table shows the parameter entries in the DB relevant for passive establishment of a connection by communication partner B:

Parameters	Data type	Value in example	Description
id	WORD	W#16#0416	Reference to this connection
connection_type	BYTE	B#16#12	Protocol variant: ISO on TCP
active_est	BOOL	FALSE	Passive connection establishment
local_device_id	BYTE	B#16#00	Communication AS-internal via CP
local_tsap_id_len	BYTE	B#16#07	Length of parameter local_tsap_id used
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#07	Length of parameter rem_tsap_id
next_staddr_len	BYTE	B#16#01	Length of parameter next_staddr

Parameters	Data type	Value in example	Description
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#E0 • local_tsap_id[2] = B#16#04 • local_tsap_id[3] = B#16#54 (ASCII equivalent of "T") • local_tsap_id[4] = B#16#43 (ASCII equivalent of "C") • local_tsap_id[5] = B#16#50 (ASCII equivalent of "P") • local_tsap_id[6] = B#16#2D (ASCII equivalent of "-") • local_tsap_id[7] = B#16#31 (ASCII equivalent of "1") • local_tsap_id[8-16] = B#16#00	local TSAP-ID: 0xE0045443502D31
rem_staddr	ARRAY [1..6] of BYTE	"192.168.4.14" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#04 (4) • rem_staddr[4] = B#16#0E (14) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point
rem_tsap_id	ARRAY [1..16] of BYTE	• rem_tsap_id[1] = B#16#E0 • rem_tsap_id[2] = B#16#03 • rem_tsap_id[3] = B#16#54 (ASCII equivalent of "T") • rem_tsap_id[4] = B#16#43 (ASCII equivalent of "C") • rem_tsap_id[5] = B#16#50 (ASCII equivalent of "P") • rem_tsap_id[6] = B#16#2D (ASCII equivalent of "-") • rem_tsap_id[7] = B#16#31 (ASCII equivalent of "1") • rem_tsap_id[8-16] = B#16#00	remote TSAP-ID: 0xE0035443502D31
next_staddr	ARRAY [1..6] of BYTE	• next_staddr[1] = B#16#28 • next_staddr[2-6] = B#16#00	Rack = 1, slot = 8 (bits 7 to 5: Rack no., bits 4 to 0: slot no.)

Example 2: Communication over TCP and integrated PROFINET interfaces between an S7-400 CPU and an S7-300 CPU

One partner is a CPU 416-3 PN/DP, the other is a CPU 319-3 PN/DP. The following table shows the most important data for both communications partners:

Property	Communication partner A: CPU 416-3 PN/DP	Communication partner B: CPU 319-3 PN/DP
Connection establishment	active	passive
IP address	192.168.3.142	192.168.3.125
Local port no.	irrelevant	2005

The following table shows the parameter entries in the DB relevant for active establishment of a connection by communication partner A:

Parameters	Data type	Value in example	Description
id	WORD	W#16#0014	Reference to this connection
connection_type	BYTE	B#16#11	Protocol variant: TCP
active_est	BOOL	TRUE	Active connection establishment
local_device_id	BYTE	B#16#05	Communication via the integrated Ethernet interface with S7-400 CPUs
local_tsap_id_len	BYTE	B#16#00 (only this value is possible)	Parameter local_tsap_id is not used
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#02 (only this value is possible)	Length of parameter rem_tsap_id
rem_staddr	ARRAY [1..6] of BYTE	"192.168.3.125" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#03 (3) • rem_staddr[4] = B#16#7D (125) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point
rem_tsap_id	ARRAY [1..16] of BYTE	"2005" • rem_tsap_id[1] = B#16#07 • rem_tsap_id[2] = B#16#D5 • rem_tsap_id[3-16] = B#16#00	Remote port no.: 2005 = W#16#07D5

The following table shows the parameter entries in the DB relevant for passive establishment of a connection by communication partner B:

Parameters	Data type	Value in example	Description
id	WORD	W#16#000F	Reference to this connection
connection_type	BYTE	B#16#11	Protocol variant: TCP
active_est	BOOL	FALSE	Passive connection establishment

Parameters	Data type	Value in example	Description
local_device_id	BYTE	B#16#03	Communication via the integrated Ethernet interface with the CPU 319-3 PN/DP
local_tsap_id_len	BYTE	B#16#02 (only this value is possible)	Length of parameter local_tsap_id
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#00 (only this value is possible)	Length of parameter rem_tsap_id
local_tsap_id	ARRAY [1..16] of BYTE	"2005" • local_tsap_id[1] = B#16#07 • local_tsap_id[2] = B#16#D5 • local_tsap_id[3-16] = B#16#00	Local port no.: 2005 = W#16#07D5
rem_staddr	ARRAY [1..6] of BYTE	"192.168.3.142" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#03 (3) • rem_staddr[4] = B#16#8E (142) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point

Example 3: Communication over UDP and integrated PROFINET interfaces between an S7-300 CPU and an S7-400 CPU

One partner is a CPU 319-3 PN/DP, the other is a CPU 414-3 PN/DP. The following table shows the most important data for both communications partners:

Property	Communication partner A: CPU 319-3 PN/DP	Communication partner B: CPU 414-3 PN/DP
Sender/receiver	Sender	Receiver
IP address	192.168.3.142	192.168.3.125
Local port no.	2004	2005

The following table shows the parameter entries in the DB relevant for the sender (communication partner A) for assigning parameters to the local communications access point:

Parameter	Data type	Value in example	Description
id	WORD	W#16#0014	Reference to this connection between the application program and the communication layer of the operating system.
connection_type	BYTE	B#16#13	Protocol variant: UDP
active_est	BOOL	FALSE	Only this value can be used with the protocol variant UDP.
local_device_id	BYTE	B#16#03	Communication via the integrated Ethernet interface with the CPU 319-3 PN/DP

Parameter	Data type	Value in example	Description
local_tsap_id_len	BYTE	B#16#02	Length of parameter local_tsap_id
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#07 • local_tsap_id[2] = B#16#D4 • local_tsap_id[3-16] = B#16#00	Local port no.: 2004 = W#16#07D4

The following table shows the parameter entries in the DB relevant for the receiver (communication partner B) for assigning parameters to the local communications access point:

Parameter	Data type	Value in example	Description
id	WORD	W#16#000F	Reference to this connection between the application program and the communication layer of the operating system.
connection_type	BYTE	B#16#13	Protocol variant: UDP
active_est	BOOL	FALSE	Only this value can be used with the protocol variant UDP.
local_device_id	BYTE	B#16#05	Communication via the integrated Ethernet interface with S7-400 CPUs
local_tsap_id_len	BYTE	B#16#02	Length of parameter local_tsap_id
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#07 • local_tsap_id[2] = B#16#D5 • local_tsap_id[3-16] = B#16#00	Local port no.: 2005 = W#16#07D5

When "TUSEND (Page 6220)" is called on the sender, you transfer the following address parameters of the receiver in a DB:

Parameters	Data type	Value in example	Description
rem_ip_addr	ARRAY [1..4] of BYTE	• rem_ip_addr[1] = B#16#C0 (192) • rem_ip_addr[2] = B#16#A8 (168) • rem_ip_addr[3] = B#16#3 (3) • rem_ip_addr[4] = B#16#7D (125)	IP address of the receiver: 192.168.3.125
rem_port_nr	ARRAY [1..2] of BYTE	• rem_port_nr[1] = B#16#07 • rem_port_nr[2] = B#16#D5	Port no. of the receiver: 2005 = W#16#07D5

When "TURCV (Page 6223)" is called on the receiver, you receive the following address parameters of the sender in a DB:

Parameters	Data type	Value in example	Description
rem_ip_addr	ARRAY [1..4] of BYTE	• rem_ip_addr[1] = B#16#C0 (192) • rem_ip_addr[2] = B#16#A8 (168) • rem_ip_addr[3] = B#16#3 (3) • rem_ip_addr[4] = B#16#8E (142)	IP address of the sender: 192.168.3.142
rem_port_nr	ARRAY [1..2] of BYTE	• rem_port_nr[1] = B#16#07 • rem_port_nr[2] = B#16#D4	Port no. of the sender: 2004 = W#16#07D4

Example 4: Communication over ISO on TCP and integrated PROFINET interfaces between two S7-400 CPUs

One of the communication partners is a CPU 414-3 PN/DP as of firmware version V5.0, the other is a CPU 416-3 PN/DP with firmware version V5.0. Communication is handled over the integrated PROFINET interfaces.

The following table shows the most important data of the two communication partners.

Property	Communication partner A: CPU 414-3 PN/DP	Communication partner B: CPU 416-3 PN/DP
Sender/receiver	Sender	Receiver
Connection establishment	passive	active
IP address	192.168.0.14	192.168.0.16

The following table shows all parameter entries of communication partner A in the corresponding DB.

Parameter	Data type	Value in example	Description
block_length	WORD	W#16#40	Length of UDT65: 64 bytes (fixed)
id	WORD	W#16#0001	Reference to this connection
connection_type	BYTE	B#16#12	Protocol variant: • B#16#12: ISO on TCP
active_est	BOOL	FALSE	Passive connection establishment
local_device_id	BYTE	B#16#05	Communication via the integrated IE interface for CPUs 41x PN/DP
local_tsap_id_len	BYTE	B#16#03	Length of parameter local_tsap_id
rem_subnet_id_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#03	Length of parameter rem_tsap_id

Parameter	Data type	Value in example	Description
next_staddr_len	BYTE	B#16#00	This parameter is not used (because communication is over the integrated interface). Assign with B#16#00 .
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#E0 • local_tsap_id[2] = B#16#03 • local_tsap_id[3] = B#16#01 • local_tsap_id[4-16] = B#16#00	local TSAP-ID: 0xE00301
rem_subnet_id	ARRAY [1..6] of BYTE	rem_subnet_id[1-6] = B#16#00	This parameter is not used. Assign "0" to it.
rem_staddr	ARRAY [1..6] of BYTE	"192.168.0.16" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#00 (0) • rem_staddr[4] = B#16#10 (16) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point
rem_tsap_id	ARRAY [1..16] of BYTE	• rem_tsap_id[1] = B#16#E0 • rem_tsap_id[2] = B#16#03 • rem_tsap_id[3] = B#16#01 • rem_tsap_id[4-16] = B#16#00	remote TSAP-ID: 0xE00301
next_staddr	ARRAY [1..6] of BYTE	next_staddr[1-6] = B#16#00	This parameter is not used (because communication is over the integrated interface)
spare	WORD	W#16#0000	This parameter is not used. Assign "0" to it.

The following table shows all parameter entries of communication partner B in the corresponding DB.

Parameter	Data type	Value in example	Description
block_length	WORD	W#16#40	Length of UDT65: 64 bytes (fixed)
id	WORD	W#16#0001	Reference to this connection
connection_type	BYTE	B#16#12	Protocol variant: • B#16#12: ISO on TCP
active_est	BOOL	TRUE	Active connection establishment
local_device_id	BYTE	B#16#05	Communication via the integrated IE interface for CPUs 41x PN/DP
local_tsap_id_len	BYTE	B#16#03	Length of parameter local_tsap_id
rem_subnet_id_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#03	Length of parameter rem_tsap_id

Parameter	Data type	Value in example	Description
next_staddr_len	BYTE	B#16#00	This parameter is not used (because communication is over the integrated interface). Assign with B#16#00.
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#E0 • local_tsap_id[2] = B#16#03 • local_tsap_id[3] = B#16#01 • local_tsap_id[4-16] = B#16#00	local TSAP-ID: 0xE00301
rem_subnet_id	ARRAY [1..6] of BYTE	rem_subnet_id[1-6] = B#16#00	This parameter is not used. Assign "0" to it.
rem_staddr	ARRAY [1..6] of BYTE	"192.168.0.14" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#00 (0) • rem_staddr[4] = B#16#0E (14) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point
rem_tsap_id	ARRAY [1..16] of BYTE	• rem_tsap_id[1] = B#16#E0 • rem_tsap_id[2] = B#16#03 • rem_tsap_id[3] = B#16#01 • rem_tsap_id[4-16] = B#16#00	remote TSAP-ID: 0xE00301
next_staddr	ARRAY [1..6] of BYTE	next_staddr[1-6] = B#16#00	This parameter is not used (because communication is over the integrated interface)
spare	WORD	W#16#0000	This parameter is not used. Assign "0" to it.

Data types

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

TCON: Establish communication connection

Number of connections possible

For information on the number of possible TCP or UDP connections, refer to the specifications for your CPU.

Use with TCP and ISO on TCP

Both communication partners call the "TCON" instruction to set up and establish the communication connection. In the parameter assignment, you specify which partner is the active communication end point and which is passive.

After the connection is set up and established, it is automatically maintained and monitored by the CPU.

If the connection is terminated, for example due to a line break or due to the remote communication partner, the active partner attempts to re-establish the configured connection. You do not have to call "TCON" again.

An existing connection is terminated and the connection is removed when the "TDISCON" instruction is called or when the CPU has gone into STOP mode. To set up and establish the connection again, call "TCON" again.

Use with UDP

Both communication partners call the "TCON" instruction to configure their local communication access point (local port). A connection is configured between the user program and the communication layer of the operating system. No connection is established to the remote partner.

The local access point is used to send and receive UDP message frames.

The following parameters must match in the configuration of the local access point (UDT 65) or in the configuration of the remote partner (UDT 66).

- With UDP: Reference to the connection and the local port no.

Functional description

The "TCON" instruction is asynchronous, in other words, job processing extends over multiple calls. You start the job for setting up and establishing a connection by calling "TCON" with REQ = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can recognize the current status of "TCON" or when the connection was set up or established (for TCP and ISO-on-TCP).

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Parameters

The following table shows the parameters of the "TCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C	Control parameter REQUEST starts the job for establishing the connection specified by ID. The job starts on a rising edge.
ID	Input	WORD	M, D or constant	Reference to the connection to be established to the remote partner, or between the user program and the communication layer of the operating system. ID must be identical to the corresponding parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. • BUSY = 0: The job is completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during job processing. STATUS supplies detailed information on the type of error
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information
CONNECT	InOut	ANY	D	Pointers to the associated connection description (UDT 65) See also: • Assigning parameters for communications connections with TCP and ISO on TCP (Page 6191) • Assigning parameters for the local communications access point with UDP (Page 6196) Note: You can enter the CONNECT parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: CONN_DB.Conn1

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	Connection was established successfully
0	7000	No job processing active
0	7001	Start job processing, establishing connection
0	7002	Intermediate call (REQ irrelevant), connection being established
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8089	The CONNECT parameter does not point to a data block.
1	809A	The CONNECT parameter points to a field that does not match the length of the connection description (UDT65).
1	809B	The local_device_id in the connection description does not match the CPU.
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2
1	80A1	Connection or port is already being used by the user
1	80A2	Local or remote port is being used by the system
1	80A3	Attempt being made to re-establish an existing connection
1	80A4	IP address of the remote connection end point is invalid, e.g., it matches the local partner's own IP address
1	80A7	Communication error: You have called "TDISCON (Page 6212)" before "TCON" was completed. Connection establishment was aborted by calling "TDISCON (Page 6212)".
1	80B2	The CONNECT parameter points to a data block that was generated with the keyword UNLINKED.
1	80B3	Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9
1	80B4	You have violated one or more of the following conditions when establishing a connection passively (active_est = FALSE) with the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0 • With local_tsap_id_len >= B#16#03, local_tsap_id[1] is an ASCII character. • local_tsap_id[1] is an ASCII character and local_tsap_id_len >= B#16#03
1	80B5	Error in parameter active_est (UDT 65) with the UDP protocol variant
1	80B6	Parameters assignment error in parameter connection_type (UDT 65)
1	80B7	Error in one of the following parameters of UDT 65: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len
1	80B8	Parameters in the local connection description (UDT 65) and Parameter ID are different
1	80C3	All connection resources are in use.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters. • The configured connection is currently being removed by a "TDISCON (Page 6212)". • The H system is connecting and updating.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

See also

Assigning parameters for communications connections with TCP and ISO on TCP (Page 6191)

Assigning parameters for the local communications access point with UDP (Page 6196)

TDISCON: Terminate communication connection

Use with TCP and ISO on TCP

The "TDISCON" instruction terminates a communication connection from the CPU to a communication partner.

Use with UDP

The "TDISCON" instruction closes the local communication access point. This means that the connection between the user program and the communication layer of the operating system is terminated.

Functional description

The "TDISCON" instruction is asynchronous, in other words, job processing extends over multiple calls. You start the job for terminating a connection by calling the "TDISCON" instruction with REQ = 1.

After the "TDISCON" instruction has been successfully executed, the ID specified for the "TCON (Page 6208)" instruction is no longer valid and cannot be used for sending or receiving.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions (see also: Difference between synchronous and asynchronous instructions (Page 4186)).

Parameters

The following table shows the parameters of the "TDISCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C	Starts the job to terminate the connection specified in the ID on a rising edge.
ID	Input	WORD	M, D or constant	Reference to the connection established with "TCON (Page 6208)". Range of values: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	M, D	Status of the instruction

Parameters BUSY, DONE and ERROR

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of TDISCON. The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	Connection successfully terminated
0	7000	No job processing active
0	7001	Start of job execution, connection is being terminated.
0	7002	Connection is being terminated (REQ irrelevant).
1	8086	The ID parameter is outside the permitted address range.
1	80A3	Attempt being made to terminate a non-existent connection.
1	80C4	Temporary communication error: The interface is currently receiving new parameters, or the connection is currently being established / the H system is connecting and updating.

See also

Overview of the valid data types (Page 231)

TSEND: Send data via communication connection

Description

The "TSEND" instruction sends data over an existing communication connection.

Functional description

"TSEND" is an asynchronous instruction, in other words, its processing extends over multiple calls. You start the job by calling "TSEND" with REQ = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

The DONE output parameter is set if the data to be sent to the local interface was transferred completely.

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current "TSEND" status.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TSEND", make sure the data in the send area remains consistent until the DONE parameter or the ERROR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C	Control parameter REQUEST starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
ID	Input	WORD	M, D or constant	Reference to the associated connection. ID must be identical to the corresponding parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	INT	I, Q, M, D, L	Maximum number of bytes to be sent with the job See also: Relationship between CPU and protocol variant used (connection_type) and transferable data length (Page 6199)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: The data was transferred completely to the local interface for sending.

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information
DATA	InOut	ANY	I, Q, M, D	Send area, contains address and length The address refers to: • The process image of the inputs • The process image of the outputs • A bit memory • A data block Note: Do not use ARRAY of BOOL as send area. Note: You can enter the DATA parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	Send job completed without error
0	7000	No job processing active
0	7001	Start of job processing, data being sent Note: During this processing phase, the operating system accesses the data in the DATA send area.
0	7002	Intermediate call (REQ irrelevant), job is being processed Note: During this processing phase, the operating system accesses the data in the DATA send area.
1	8085	The LEN parameter has the value "0" or is greater than the highest permitted value.
1	8086	The ID parameter is not in the permitted range
1	8088	The LEN parameter is greater than the memory area specified in DATA.

ERROR	STATUS (W#16#...)	Explanation
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is currently being terminated. Transmission over this connection is not possible. • The interface is being reinitialized.
1	80B3	The connection type (connection_type parameter in the connection description) is set to UDP. Please use "TUSEND (Page 6220)".
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection to the partner cannot be established at the moment. • The interface is receiving new parameter settings or the connection is currently being established.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

TRCV: Receive data via communication connection

Description

The "TRCV" instruction receives data over an existing communication connection.

Receive area

This means the area in which the function block enters the received data.

The receive area is specified by the following two variables:

- Pointer to the start of the area
- Length of the area

Depending on the protocol variant being used, the length of the area is specified either by the LEN parameter (if LEN <> 0) or the length information of the DATA parameter (if LEN = 0).

Receive modes of TRCV

The following table shows how "TRCV" enters the received data in the receive area.

Protocol variant	Entering the data in the receive area	connection_type parameter	Value of the LEN parameter
TCP	Ad-hoc mode	B#16#01, B#16#11	0
TCP	Data reception with specified length	B#16#01, B#16#11	<> 0
ISO on TCP	protocol-controlled	B#16#12	0 (recommended) or <> 0

TCP / ad-hoc mode

The ad-hoc mode exists only with the TCP protocol variant. You set ad-hoc mode by assigning the value "0" to the LEN parameter.

The receive area is identical to the area formed by DATA. A maximum of 8192 bytes are received. Immediately after receiving a block of data, the "TRCV" instruction transfers this to the receive area and sets NDR to "1".

TCP / data reception with specified length

You set data reception with a specified length by assigning a value other than zero to the LEN parameter.

The receive area is defined by the LEN and DATA parameters. As soon as LEN bytes have been received, TRCV transfers them to the receive area and sets NDR to "1".

ISO on TCP / protocol-controlled data transfer

With the ISO-on-TCP protocol variant, data is transferred protocol-controlled.

The receive area is defined by the LEN and DATA parameters. As soon as all job data has been received, "TRCV" transfers it to the receive area and sets NDR to "1".

Data exchange of data type STRING

If you want to exchange data of the data type STRING with the "TSEND" and "TRCV" instructions, only the Ad-hoc-Mode with LEN=0 is appropriate. Besides, you have to set the string length for "TRCV" at least that large as for the "TSEND" instruction. If you do not follow this rule, it can lead to errors during the further string processing.

If you use ARRAY of BYTE as receive area for a sent character string, the first byte contains the maximum length that is set by the sender abd the secind byte the actual character string length.

Functional description

"TRCV" is an asynchronous instruction, in other words, its processing extends over multiple calls. You start the receive job by calling TRCV with EN_R = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

The following table shows the relationship between BUSY, NDR and ERROR. Using this table, you can determine the current status of "TRCV" or when the receive operation is completed.

BUSY	NDR	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.

BUSY	NDR	ERROR	Description
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TRCV", the data in the receive area is only consistent when the NDR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L, T, C	Control parameter enabled to receive: With EN_R = 1, "TRCV" is ready to receive. The receive job is being processed.
ID	Input	WORD	M, D or constant	Reference to the associated connection. ID must be identical to the associated parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	INT	I, Q, M, D, L	Length of the receive area in bytes For the meaning of LEN = 0 or LEN <> 0, see above (receive modes of "TRCV"). For the value range, see also: Relationship between CPU and protocol variant used (connection_type) and transferable data length (Page 6199).
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • NDR = 0: Job not yet started or still running • NDR = 1: Job successfully completed
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information

Parameter	Declaration	Data type	Memory area	Description
RCVD_LEN	Output	INT	I, Q, M, D, L	Amount of data actually received in bytes
DATA	InOut	ANY	I, Q, M, D	Receive area (definition see above), contains address and length The address refers to: • The process image of the inputs • The process image of the outputs • A bit memory • A data block Note: Do not use ARRAY of BOOL as receive area. Note: You can enter the DATA parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	New data was accepted. The current length of the received data is shown in RCVD_LEN.
0	7000	Block not ready to receive
0	7001	Block is ready to receive, receive job was activated
0	7002	Intermediate call, receive job being processed Note: During this processing phase the instruction writes data to the receive area. For this reason, an error could result in inconsistent data in the receive area.
1	8085	The LEN parameter is greater than the largest permitted value, or you changed the value of the LEN or DATA parameter since the first call
1	8086	The ID parameter is not in the permitted range
1	8088	• Receive area is too small • Value in LEN is higher than the receive area specified by DATA
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is currently being terminated. A receive job over this connection is not possible. • New parameter settings are being assigned to the interface.
1	80B3	The protocol variant (connection_type parameter in the connection description) is set to UDP. Please use "TURCV (Page 6223)".
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.

ERROR	STATUS (W#16#...)	Explanation
1	80C4	Temporary communication error: • The connection to the partner cannot be established at the moment. • The interface is receiving new parameter settings or the connection is currently being established.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

TUSEND: Send data via Ethernet (UDP)

Description

The "TUSEND" instruction sends data to the remote partner addressed by the ADDR parameter using UDP.

WARNING

Data transfer via UDP

When transferring data with UDP according to RFC 768, the data is transferred to the remote partner without acknowledgement and is therefore unreliable. This means that data can be lost without this being indicated by the block.

Note

For sequential send operations to different partners, you only need to adjust the ADDR parameter when calling "TUSEND". You do not need to call the "TCON (Page 6208)" and "TDISCON (Page 6212)" instructions again.

Functional description

"The TUSEND" instruction works asynchronously, which means its job processing extends over multiple calls. You start the job by calling "TUSEND" with REQ = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current status of "TUSEND" or when the send process is concluded.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.

BUSY	DONE	ERROR	Description
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TUSEND", make sure the data in the send area remains consistent until the DONE parameter or the ERROR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TUSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C	Control parameter REQUEST starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
ID	Input	WORD	M, D or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the corresponding parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	INT	I, Q, M, D, L	Number of bytes to be sent with the job Range of values: 1 to 1472
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information

Parameter	Declaration	Data type	Memory area	Description
DATA	InOut	ANY	I, Q, M, D	Send area, contains address and length The address refers to: • The process image of the inputs • The process image of the outputs • A bit memory • A data block Note: You can enter the DATA parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable
ADDR	InOut	ANY	D	Pointers to the address of the receiver (for example, P#DB100.DBX0.0 byte 8) See also: Structure of the address information for the remote partner with UDP (Page 6198) Note: You can enter the ADDR parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	Send job completed without error
0	7000	No job processing active
0	7001	Start of job processing, data being sent Note: During this processing phase, the operating system accesses the data in the DATA send area.
0	7002	Intermediate call (REQ irrelevant), job is being processed Note: During this processing phase, the operating system accesses the data in the DATA send area.
1	8085	The LEN parameter has the value "0" or is greater than the highest permitted value.
1	8086	The ID parameter is not in the permitted range
0	8088	The LEN parameter is greater than the memory area specified in DATA.
1	8089	The ADDR parameter does not point to a data block

ERROR	STATUS (W#16#...)	Explanation
1	80A1	Communication error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. Transmission over this connection is not possible. • The interface is being reinitialized.
1	80A4	IP address of the remote connection end point is invalid, e.g., it matches the local partner's own IP address.
1	80B3	• The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TSEND (Page 6213)". • ADDR parameter: Invalid information for port no.
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection between the user program and the communication layer of the operating system cannot be established at this time. • New parameter settings are being assigned to the interface.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

TURCV: Receive data via Ethernet (UDP)

Description

The "TURCV" instruction receives data via UDP. After successful completion of "TURCV", the ADDR parameter will show you the address of the remote partner (the sender).

WARNING

Data transfer via UDP according to RFC 768

When transferring data with UDP according to RFC 768, the data is transferred to the remote partner without acknowledgement and is therefore unreliable. This means that data can be lost without this being indicated by the block.

Functional description

"TURCV" is an asynchronous instruction, in other words, its job processing extends over multiple calls. You start the receive job by calling "TURCV" with EN_R = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions

See also: Difference between synchronous and asynchronous instructions (Page 4186).

The following table shows the relationship between BUSY, NDR and ERROR. Using this table, you can determine the current status of TURCV or when the receive process is concluded.

BUSY	NDR	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TURCV", the data in the receive area is only consistent when the NDR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TURCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L, T, C	Control parameter enabled to receive: With EN_R = 1, "TURCV" is ready to receive. The receive job is being processed.
ID	Input	WORD	M, D or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the associated parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	INT	I, Q, M, D, L	Length of the receive area in bytes: 0 (recommended) or 1 to 1472 See also: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 6187)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • NDR = 0: Job not yet started or still running • NDR = 1: Job successfully completed
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information
RCVD_LEN	Output	INT	I, Q, M, D, L	Amount of data actually received in bytes

Parameter	Declaration	Data type	Memory area	Description
DATA	InOut	ANY	I, Q, M, D	Receive area (for definition, see Functional description of instructions for Open User Communication via Industrial Ethernet (Page 6187)) The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block Note: You can enter the DATA parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable
ADDR	InOut	ANY	D	Pointers to the address of the receiver (for example, P#DB100.DBX0.0 byte 8), see also: Structure of the address information for the remote partner with UDP (Page 6198) Note: You can enter the ADDR parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	New data was accepted. The current length of the received data is shown in RCVD_LEN.
0	7000	Block not ready to receive
0	7001	Block is ready to receive, receive job was activated
0	7002	Intermediate call, receive job being processed Note: During this processing phase, "TURCV" writes data to the receive area. For this reason, an error could result in inconsistent data in the receive area.
1	8085	The LEN parameter is greater than the largest permitted value, or you changed the value of the LEN or DATA parameter since the first call
1	8086	The ID parameter is not in the permitted range
1	8088	• Receive area is too small • Value in LEN is higher than the receive area specified by DATA
1	8089	The ADDR parameter does not point to a data block

ERROR	STATUS (W#16#...)	Explanation
1	80A1	Communication error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. A receive job over this connection is not possible. • New parameter settings are being assigned to the interface.
1	80B3	The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TRCV (Page 6216)".
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: New parameter settings are being assigned to the interface.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

IP_CONF: Change IP configuration parameters

Description of IP_CONF

Description

The "IP_CONF" instruction is used for the program-controlled configuration of the integrated PROFINET interface of the CPU. The existing configuration data is overwritten.

You can make the following interface configuration settings:

- IP parameters: IP address, subnet mask, router address
- PROFINET IO device name (if the CPU is operated as a PROFINET IO device)

You have to store the configuration data in a configuration DB.

You can make the program-controlled setting of the IP configuration with the "IP_CONF" instruction as an alternative to configuration with STEP 7. It only takes effect, however, if you explicitly specified in the hardware configuration that IP parameters are assigned via the user program.

Note

As an alternative to project engineering, the configuration data of the communication processor can be set using the "IP_CONFIG" instruction (see SIMATIC NET CP).

Functional description

The "IP_CONF" instruction works asynchronously, that is, its execution extends over multiple calls. You start the transfer operation by calling "IP_CONF" with REQ = 1.

Only one job can be active at any time.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions (see also: Difference between synchronous and asynchronous instructions (Page 4186)).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current status of the instruction and when the transfer of configuration data is concluded.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Parameters

The following table shows the parameters of the "IP_CONF" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C	Control parameter REQUEST. starts the instruction on a rising edge.
LADDR	Input	WORD	M, D or constant	Diagnostics address of the PROFINET interface
CONF_DB (Page 6229)	Input	ANY	D	Pointer to the configuration data (permitted data types: BYTE, WORD, BLOCK_DB) Note: You can enter the CONF_DB parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	• 0: Job is complete. • 1: The job is not yet completed. A new job cannot be triggered.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
STATUS	Output	DWORD	I, Q, M, D, L	Error information
ERR_LOC	Output	DWORD	I, Q, M, D, L	Error source (field_id and subfield_type_id of the parameter block causing the error)

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters ERROR and STATUS

ERROR	STATUS (DW#16#..)	ERR_LOC*	Explanation
0	00000000	0	Job execution terminated without error
0	00700000	0	No job processing active
0	00700100	0	Start of job execution
0	00700200	0	Intermediate call (REQ irrelevant)
1	C08xyy00	0	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)
1	C0808000	0	LADDR is invalid
1	C0808100	0	LADDR is not assigned to the supported PROFINET interface
1	C0808200	0	Error in the CONF_DB parameter: Data type is not supported.
1	C0808300	0	Error in the CONF_DB parameter: The pointer points to an unsupported area.
1	C0808400	0	Error in the CONF_DB parameter: Incorrect length of the ANY pointer
1	C0808700	0	Error in the CONF_DB parameter: Inconsistency of the length specified in the configuration data
1	C0808800	s, 0	field_type_id has an illegal value
1	C0808900	s, 0	field_id has an illegal value
1	C0808A00	s, 0	Wrong number for subfield_cnt
1	C0808B00	s, t	subfield_id has an illegal value
1	C0808C00	s, t	Subfield used more than once
1	C0808D00	s, t	subfield_len has an incorrect or illegal value
1	C0808E00	s, t	subfield_mode has an illegal value
1	C0808F00	s, t	There is a conflict with an earlier subfield in the subfield.
1	C0809000	s, t	The parameters of the subfield are write-protected. E.g. parameters are specified via configuration or PNIO mode is enabled.
1	C0809400	s, t	The parameter value in the subfield is not defined or illegal
1	C080C200	0	Transfer cannot be carried out (e.g. because the interface is not reachable).
1	C080C300	0	Insufficient resources (for example, multiple calling of "IP_CONF" with different parameters)
1	C080C400	0	Temporary communication error
* In the table above, f is the field_id and s the subfield_type_id of the parameter block causing the error.			

Data block CONF_DB

Description

The following figure shows how the configuration data to be transferred is stored in the configuration DB.

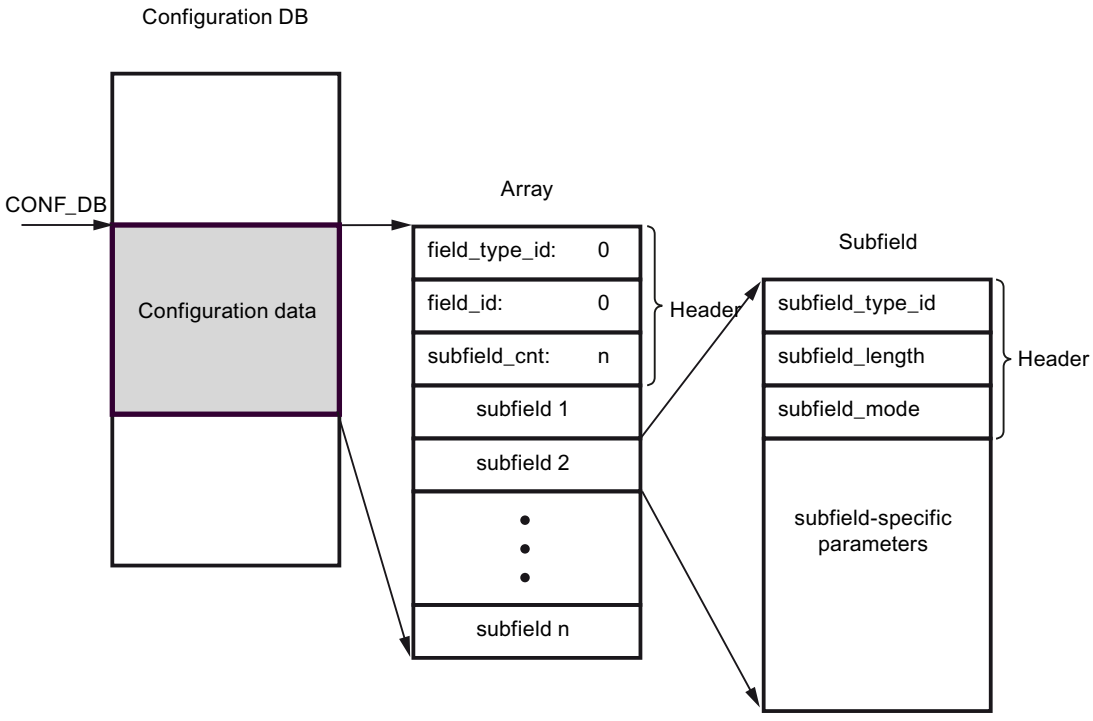

The configuration data therefore comprises exactly one field and several subfields:

- The field consists of a header and the subfields. The header, in turn, consists of the following elements:
  - field_type_id (data type INT): Zero
  - field_id (data type INT): Zero
  - subfield_cnt (data type INT): Number of subfields:
- Each subfield consists, in turn, of a header (subfield_type_id, subfield_length, subfield_mode) and the subfield-specific parameters. Each subfield must consist of an even number of bytes.
  The following values are permitted for subfield_mode:
  - 1: permanent validity of the configuration data
  - 2: temporary validity of the configuration data including the deletion of existing permanent configuration data

Note

Exactly one field is currently permitted. Its parameters field_type_id and field_id must have the value zero. Other fields with different values for field_type_id and field_id are subject to future extensions.

Permitted subfields

subfield_type_id	Name of the subfield	Explanation
30	SUB_IP_SUITE_IPV4	IP parameters: IP address, subnet mask, router address
40	SUB_NOS	PROFINET IO device name (Name of Station)

Type definition for the subfield "SUB_IP_SUITE_IPV4"

Name		Data type	Start value	Description
SUB_IP_SUITE_V4		Struct		
	Id	INT	30	subfield_type_id
	len	INT	18	subfield_length
	mode	INT	1	subfield_mode (1: permanent, 2: temporary)
	ipaddr_3	BYTE	b#16#C8	IP address high byte: 200
	ipaddr_2	BYTE	b#16#0C	IP address high byte: 12
	ipaddr_1	BYTE	b#16#01	IP address low byte: 1
	ipaddr_0	BYTE	b#16#90	IP address low byte: 144
	snmask_3	BYTE	b#16#FF	Subnet mask high byte: 255
	snmask_2	BYTE	b#16#FF	Subnet mask high byte: 255
	snmask_1	BYTE	b#16#FF	Subnet mask low byte: 255
	snmask_0	BYTE	b#16#00	Subnet mask low byte: 0
	router_3	BYTE	b#16#C8	Router high byte: 200
	router_2	BYTE	b#16#0C	Router high byte: 12

Name		Data type	Start value	Description
	router_1	BYTE	b#16#01	Router low byte: 1
	router_0	BYTE	b#16#01	Router low byte: 1

Type definition for the subfield "SUB_NOS"

Name		Data type	Start value	Description
SUB_NOS		Struct		
	id	INT	40	subfield_type_id
	len	INT	246	subfield_length
	mode	INT	1	subfield_mode (1: permanent, 2: temporary)
	nos	ARRAY[1..240] of BYTE	0	Device name: You must occupy the ARRAY from the first byte. If the ARRAY is longer than the device name to be assigned, enter a zero byte after the actual device name (complying with IEC 61158-6-10). Otherwise, nos is rejected and the "IP_CONF (Page 6226)" instruction enters the error code DW#16#C0809400 in STATUS. If you enter zero the first byte, the device name is deleted.

The device name is subject to the following limitations:

- Restricted to a total of 240 characters (lower case letters, numbers, dash, or dot)
- A name component within the device name, in other words, a character string between two dots, must not exceed 63 characters. The name components must not begin or end with the "-" character.
- No special characters such as umlauts, brackets, underscore, slash, blank space, etc. The only special character permitted is the dash.
- The device name must not begin or end with the "-" or "." characters.
- The device name must not begin with a number.
- The device name form n.n.n.n (n = 0, ... 999) is not permitted.
- The device name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).
- The character string ".." is not permitted in the device name.

Note

You can also create an ARRAY "nos" that is shorter then 240 bytes, but not less than 2 bytes. In this case, you must adjust the "len" (length of subfield) tag accordingly.

Data exchange using FETCH and WRITE

FW_TCP: Swap data using FETCH and WRITE via TCP

Description

With the "FW_TCP" instruction, the FETCH/WRITE functions are provided via the TCP connection.

Call

Since a separate TCP connection is required for each of the services FETCH/WRITE, the instruction is called twice in OB1:

- The first call with the FETCH_TCP_DB instance data block provides the FETCH service via the TCP connection.
- The second call with the WRITE_TCP_DB instance data block provides the WRITE service via the TCP connection.

The TCP connection is automatically established after a cold restart, warm restart and hot restart of the CPU, as well as after changes to the connection parameters. The parameters for the connection establishment are stored in the "ConnectParam" data block (description: See below). The input parameter ENABLE provides the option of manually controlling the establishment and termination of the connection. When the connection parameters are changed, the connection is automatically terminated and re-established.

The following figure is a schematic representation of the "FW_TCP" instruction, which is called in OB1 and implements the FETCH and WRITE services via TCP connections.

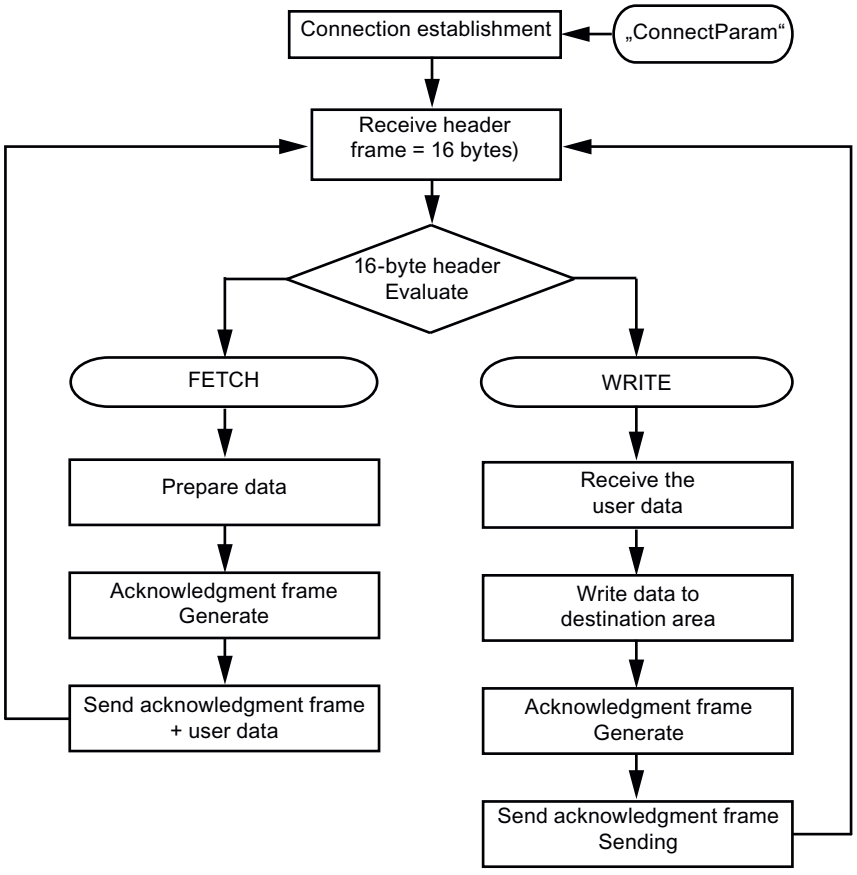

Connection parameters of the "ConnectParam" instance data block

The following connection parameters are checked before the connection is established:

- connection_type
- id
- local_tsap_id

If one of these parameters is wrongly defined, the communication connection will not be established. No data is sent or received.

- After the communication connection has been established, a 16-byte header frame of the FETCH/WRITE services is received. The header frame contains the information indicating whether a FETCH or WRITE job is pending.
- If a FETCH job is pending, the data requested by the communication partner is prepared in the instance data block DB210 of the instruction and the acknowledgement frame generated. The acknowledgement frame is sent including the prepared data.
- If a WRITE job is pending, the user data is received and written to the destination area after the evaluation of the 16-byte header frame. The acknowledgement frame is then generated in instance data block DB310 and sent.

The following table shows the data structure of the connection parameters for a TCP connection. This is 64 bytes long.

Parameter	Data type	Start value	Description
id	WORD	W#16#1	Connection ID: The value of the parameter must be within the following range: id = W#16#0001 to W#16#0FFF (1 to 4095)
connection_type	BYTE	B#16#11	Connection type. The following is valid for the TCP connection: • connection_type = B#16#11 (TCP native) or • connection_type = B#16#1 (TCP native or compatibility mode)
active_est	BOOL	FALSE	Passive connection establishment
local_device_id	BYTE	B#16#3	ID for the local PN/IE interface (here: CPU 319-3PN/DP).
local_tsap_id_len	BYTE	B#16#2	Length of the local_tsap_id parameter used in bytes: • 0 or 2, if connection type = 17 (TCP) • Only the value "0" is permitted for the active end.
rem_subnet_id_len	BYTE	B#16#0	This parameter is not used.
rem_staddr_len	BYTE	B#16#0	Length of address of partner end point (rem_staddr parameter) in bytes. Since this is an unspecified connection, the parameter is irrelevant.
rem_tsap_id_len	BYTE	B#16#0	Length of the rem_tsap_id parameter used in bytes: • 0 or 2, if connection type = 17 (TCP). • Only the value 0 is permitted for the passive end.
next_staddr_len	BYTE	B#16#0	This parameter is not used.
local_tsap_id	ARRAY [1..16] of BYTE	B#16#7 B#16#D0 B#16#0 : : B#16#0	Local port for the TCP connection. Possible values: 1 to 49151 (recommended values: 2000...5000). • local_tsap_id[1] = high byte of the port no. in hexadecimal notation; • local_tsap_id[2] = low byte of port no. in hexadecimal notation; • local_tsap_id[3-16] = irrelevant Note: Make sure that every value of local_tsap_id is unique within the CPU.
rem_subnet_id	ARRAY [1..6] of BYTE	B#16#0 …	This parameter is not used.
rem_staddr	ARRAY [1..6] of BYTE	B#16#0	IP address of the partner end point, for example, for 192.168.2.3: • rem_staddr[1] = 192 • rem_staddr[2] = 168 • rem_staddr[3] = 2 • rem_staddr[4] = 3 • rem_staddr[5-6] = irrelevant Not relevant for unspecified connections with passive connection establishment.

Parameter	Data type	Start value	Description
rem_tsap_id	ARRAY [1..16] of BYTE	B#16#0	Port of the communication partner. For 17 (TCP) possible values: 1 to 49151 (recommended values: 2000...5000). • rem_tsap_id[1] = high byte of the port no. in hexadecimal notation; • rem_tsap_id[2] = low byte of port no. in hexadecimal notation; • rem_tsap_id[3-16] = irrelevant Not relevant for unspecified connections with passive connection establishment.
next_staddr	ARRAY [1..6] of BYTE	B#16#0	This parameter is not used.
spare	WORD	W#16#0	This parameter is not used.

Parameters

The following table shows the parameters of the "FW_TCP" instruction:

Parameter	Declaration	Data type	Memory area	Description
ENABLE	Input	BOOL	I, Q, M, D, L, T, C	The establishment and termination of the connection is initiated with this parameter: • 0 = Connection is terminated • 1 = Connection is established
CONNECT	Input	ANY	D	ANY pointer to the data area of DB211 that contains the parameters for establishing the TCP connection. This data area must be at least 64 bytes in size.
ADDRMODE	Input	INT	I, Q, M, D, L or constant	This parameter defines how the data of the FETCH or WRITE job are addressed (in S7 or S5 address mode). • 0 = S7 address mode (Start address of the data is interpreted as a byte address) • 1 = S5 address mode (Start address of the data is interpreted as a word address)
NDR	Output	BOOL	I, Q, M, D, L	new data record parameter: This parameter shows that the data of the WRITE job has been successfully adopted and that the acknowledgement frame was generated and sent.
ERROR	Output	BOOL	I, Q, M, D, L	This parameter is set when: • An error occurred establishing or terminating the connection • An error occurred during the sending or receiving of data. • An invalid FETCH/WRITE header frame was received.

Parameter	Declaration	Data type	Memory area	Description
MODE	Output	BYTE	I, Q, M, D, L	This parameter shows whether a FETCH or WRITE job was executed. • 0 = No job active • 1 = WRITE job • 2 = FETCH job
STATUS (Page 6240)	Output	WORD	M, D	Internal communication status bits are output in the STATUS parameter.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

FW_IOT: Swap data using FETCH and WRITE via ISO-on-TCP

Description

With the "FW_IOT" instruction, the FETCH/WRITE services are provided via the ISO-on-TCP connection.

Call

Since a separate TCP connection is required for each of the FETCH/WRITE services, the "FW_IOT" instruction is called twice in OB1:

- The first call with the FETCH_IOT_DB instance data block provides the FETCH service via the ISO-on-TCP connections.
- The second call with the WRITE_IOT_DB instance data block provides the WRITE service via the ISO-on-TCP connection.

The ISO-on-TCP connection is automatically established after a cold restart, warm restart and hot restart of the CPU, as well as after the changes to the connection parameters. The parameters for the establishment of the connection are stored in the "ConnectParam" DB. With the ENABLE input parameter, the user has the option of manually controlling connection establishment and termination. When the connection parameters are changed, the connection is automatically terminated and re-established. The following table shows the data structure of the connection parameters for an ISO-on-TCP connection: This is 64 bytes long.

The following figure is a schematic representation of the "FW_IOT" instruction, which is called in OB1 and implements the FETCH and WRITE services via ISO-on-TCP connections.

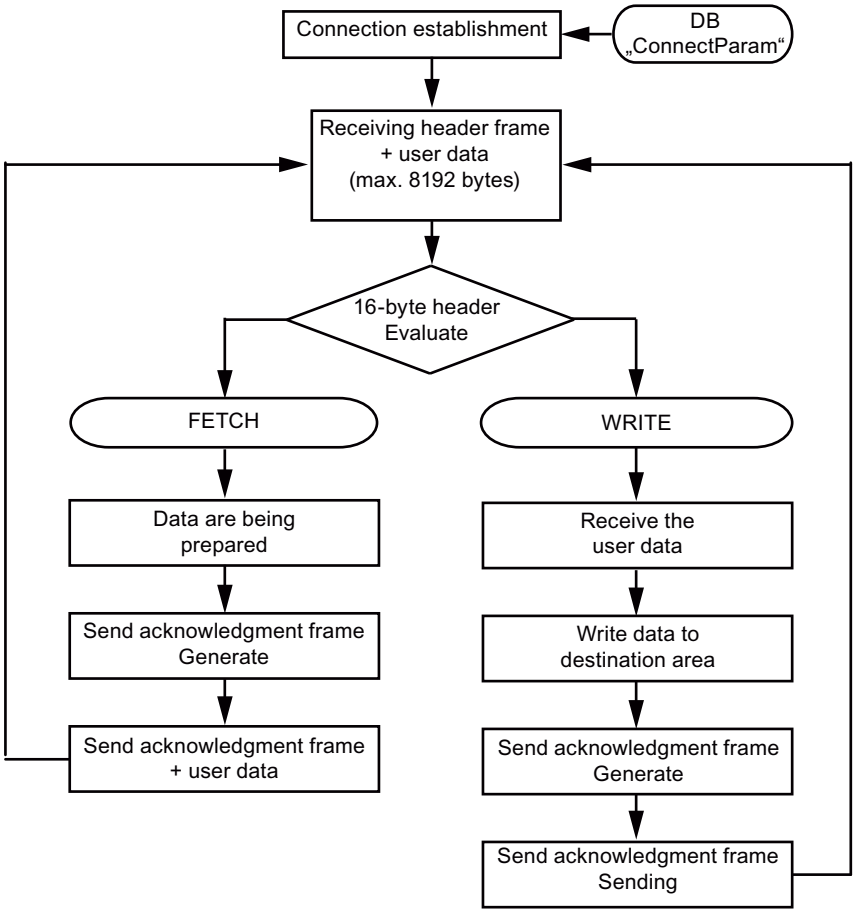

Connection parameters of the "ConnectParam" instance data block

The following connection parameters are checked before the connection is established:

- connection_type
- id
- local_tsap_id
- local_tsap_id _len

If one of these parameters is wrongly defined, the communication connection will not be established. No data is sent or received.

- After the communication connection has been established, a 16-byte header frame of the FETCH/WRITE services is received. The header frame contains the information indicating whether a FETCH or WRITE job is pending.
- If a FETCH job is pending, the data requested by the communication partner is prepared in the FETCH_IOT_DB instance data block of the instruction and the acknowledgement frame generated. The acknowledgement frame is sent including the prepared data.
- If a WRITE job is pending, the user data is received and written to the destination area after the evaluation of the 16-byte header frame. The acknowledgement frame is then generated in the WRITE_IOT_DB instance data block and sent.

The following table shows the data structure of the connection parameters for a TCP connection. This is 64 bytes long.

Parameter	Data type	Start value	Description
id	WORD	W#16#1	Connection ID: The value of the parameter must be within the following range: id = W#16#0001 to W#16#0FFF (1 to 4095)
connection_type	BYTE	B#16#12	Connection type. The following is valid for an ISO-on-TCP connection: connection_type = B#16#12
active_est	BOOL	FALSE	Passive connection establishment
local_device_id	BYTE	B#16#3	ID for the local PN/IE interface (here: CPU 319-3PN/DP).
local_tsap_id_len	BYTE	B#16#2	Length of the local_tsap_id parameter used in bytes: • 2 to 16, if connection type = 18 (ISO-on-TCP)
rem_subnet_id_len	BYTE	B#16#0	This parameter is not used.
rem_staddr_len	BYTE	B#16#0	Length of address of partner end point (rem_staddr parameter) in bytes. Since this is an unspecified connection, the parameter is irrelevant.
rem_tsap_id_len	BYTE	B#16#0	Length of the rem_tsap_id parameter used in bytes: • 2 to 16, if connection type = 18 (ISO-on-TCP)
next_staddr_len	BYTE	B#16#0	This parameter is not used.
local_tsap_id	ARRAY [1..16] of BYTE	B#16#7 B#16#D0 B#16#0 : : B#16#0	Local TSAP ID for the ISO-on-TCP connection. • local_tsap_id[1] = B#16#E0; • local_tsap_id[2] = rack and slot of local end points (bits 0 to 4: slot number, bits 5 to 7: rack number); • local_tsap_id[3-16] = TSAP extension, optional Note: Make sure that every value of local_tsap_id is unique within the CPU.
rem_subnet_id	ARRAY [1..6] of BYTE	B#16#0 …	This parameter is not used.
rem_staddr	ARRAY [1..6] of BYTE	B#16#0	IP address of the partner end point, for example, for 192.168.2.3: • rem_staddr[1] = 192 • rem_staddr[2] = 168 • rem_staddr[3] = 2 • rem_staddr[4] = 3 • rem_staddr[5-6] = irrelevant Not relevant for unspecified connections with passive connection establishment.
rem_tsap_id	ARRAY [1..16] of BYTE	B#16#0	Port of the communication partner. For partner TSAP ID with ISO-on-TCP: • rem_tsap_id[1] = B#16#E0; • rem_tsap_id[2] = rack and slot of partner end point (bits 0 to 4: slot number, bits 5 to 7: rack number); • rem_tsap_id[3-16] = TSAP extension, optional Not relevant for unspecified connections with passive connection establishment.

Parameter	Data type	Start value	Description
next_staddr	ARRAY [1..6] of BYTE	B#16#0	This parameter is not used.
spare	WORD	W#16#0	This parameter is not used.

Parameters

The following table shows the parameters of the "FW_IOT" instruction:

Parameter	Declaration	Data type	Memory area	Description
ENABLE	Input	BOOL	I, Q, M, D, L, T, C	The establishment and termination of the connection is initiated with this parameter: • 0 = Connection is terminated • 1 = Connection is established
CONNECT	Input	ANY	D	ANY pointer to the data area of the DB that contains the parameters for establishing the TCP connection. This data area must be at least 64 bytes in size.
ADDRMODE	Input	INT	I, Q, M, D, L or constant	This parameter defines how the data of the FETCH or WRITE job are addressed (in S7 or S5 address mode). • 0 = S7 address mode (Start address of the data is interpreted as a byte address) • 1 = S5 address mode (Start address of the data is interpreted as a word address)
NDR	Output	BOOL	I, Q, M, D, L	new data record parameter: This parameter shows that the data of the WRITE job has been successfully adopted and that the acknowledgement frame was generated and sent.
ERROR	Output	BOOL	I, Q, M, D, L	This parameter is set when: • An error occurred establishing or terminating the connection • An error occurred during the sending or receiving of data. • An invalid FETCH/WRITE header frame was received.
MODE	Output	BYTE	I, Q, M, D, L	This parameter shows whether a FETCH or WRITE job was executed. • 0 = No job active • 1 = WRITE job • 2 = FETCH job
STATUS (Page 6240)	Output	WORD	M, D	Internal communication status bits are output in the STATUS parameter.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

STATUS parameter

Structure of the STATUS parameter

The STATUS output parameter of the "FW_TCP" and "FW_IOT" instructions provides information on the status of the communication between S7 station and PC station.

This status word is structured as follows:

Bit	Instance DB.FBStatus	Description
High byte		
0	NotConnected	No connection established or connection interrupted • The FETCH/WRITE client has not yet initiated the establishment of the connection • Error during connection establishment
1	ErrorConnect	Error in connection establishment (for example: Error in the data structure of the connection parameter)
2	ErrorDisConnect	Error during connection termination
3	ErrorConType	Wrong connection type in the data structure of the connection parameters)
4	Not assigned	
5	BusyConnect	Connection is being established
6	BusyDisConnect	Connection is being terminated
7	Not assigned	
Low byte		
8	ErrorRec1	TCP connection: • Error in receive job • FETCH/WRITE header frame could not be received ISO-on-TCP connection • Error in receive job • FETCH/WRITE header frame including user data could not be received
9	ErrorRec2	TCP connection • Error in receive job • User data could not be received
10	ErrorSend	TCP and ISO-on-TCP connection • Error in send job • FETCH/WRITE acknowledgement frame including user data could not be sent
11	Not assigned	
12	BusyRec1	Receive job is running
13	BusyRec2	Receive job is running
14	BusySend	Send job is running
15	ErrorHeader	Undefined FETCH/WRITE header frame received.

Addresses and tags of the instance data block

The error status of the connection establishment and termination, and the error status of the send and receive job are stored in the instance data block of the instructions "FW_TCP" and "FW_IOT".

- The instance data block of the "FW_TCP" instruction comprises 8348 bytes.
- The instance data block of the "FW_IOT" instruction comprises 16544 bytes.

The following table shows an overview of the addresses and tags of the instance data block where the status is stored.

Tag	Data type	Address	Description
DoneError.STATUS_Connect	WORD	22.0	Connection establishment status
DoneError.STATUS_REC_1	WORD	24.0	Status of the first receive job
DoneError.STATUS_REC_2	WORD	26.0	Status of the second receive job (only for TCP)
DoneError.STATUS_SEND	WORD	28.0	Status of the send job
DoneError.STATUS_DisConnect	WORD	30.0	Connection termination status

Value of the STATUS parameter

The following table provides an overview of the status values of the connection establishment.

STATUS (W#16#...)	Description
0000	Connection was established successfully.
8086	The ID parameter is outside the permitted range.
8087	Maximum number of connections reached; no additional connection possible.
8089	The CONNECT parameter does not point to a data block.
809A	The CONNECT parameter points to a field that does not match the length of the connection description.
809B	The local_device_id in the connection description does not match the CPU.
80A1	Connection or port is already in use.
80A2	Local or remote port is being used by the system.
80A3	Attempt is being made to terminate a non-existent connection.
80A4	IP address of the remote connection end point is invalid; for example, it matches the local partner's own IP address
80B2	The CONNECT parameter points to a data block that was generated with the keyword UNLINKED.
80B3	Inconsistent parameter assignment
80B4	You have violated one or more of the following conditions when establishing a connection passively (active_est = FALSE) with the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0
80B6	Parameter assignment error relating to the connection_type parameter
80B7	Error in the connection description
80C3	All connection resources are in use: • Another block with this ID is already being processed in a different priority class. • Internal lack of resources.

STATUS (W#16#...)	Description
80C4	Temporary communications error: • The connection cannot be established at this time. • New parameter settings are being assigned to the interface. • The configured connection is currently being removed by the "TDISCON" instruction.
82xx	General error information

4.2.5.4 Web server

WWW: Synchronizing user-defined web pages

Description

The instruction WWW initializes the Web server of the CPU or synchronizes user-defined web pages with the user program in the CPU.

User-defined web pages together with the Web server make it possible for the CPU to access freely designed web pages of the CPU with a web browser.

Use script instructions (such as Javascript) and HTML code in user-defined web pages to transfer data via a web browser for further processing to the CPU and to display data from the operand area of the CPU in the web browser. Call the WWW instruction in the user program for synchronization of the user program and the Web server as well as initialization.

Initialization

User-defined web pages are "packaged" in data blocks for processing by the CPU. You will have to generate appropriate data blocks from the source files (HTML files, screens, Javascript files, ...) during configuration. The Web Control DB takes on a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. The data blocks with coded web pages are called fragment DBs.

When the data block is downloaded into the CPU, the CPU does not "know" that user-defined web pages are coded inside it. The instruction "WWW" in the startup OB, for example, will inform the CPU which DB is the Web Control DB. The user-defined web pages can be accessed via a web browser after this initialization.

Synchronization

If you want the user program to interact with the user-defined web pages, then the instruction WWW must be used in the cyclical program part.

Examples of interaction between user program and web page:

- Check received data
- Assemble and send back data to the web browser making the request

In this case it must be possible to evaluate the current status information and the Web server must receive control information, such as release of a web page requested by a web browser.

Parameter

The following table shows the parameters of the instruction "WWW":

Parameter	Declaration	Data type	Memory area	Description
CTRL_DB	Input	BLOCK_DB	I, Q, M, D, L or constant	Data block that describes the user-defined web pages (Web Control DB)
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred. There are no web page requests that have to be released by the user program.
00xy	x: indicates whether an error has occurred during initialization of the Web Control DB (CTRL_DB): x=0: No errors occurred. x=1: Error occurred. The error is coded in the byte "CTRL_DB.commandstate.last_error" of the Web Control DB, see description of Web Control DB. y: Number of the pending request. Several requests are possible (e. g. requests "0" and "1" are pending: y="3". y="1": Request "0" y="2": Request "1" y="4": Request "2" y="8": Request "3"
803A	The specified Web Control DB does not exist on the CPU.
8081	Incorrect version or incorrect format of the Web Control DB
80C1	There are no resources to initialize the web application, for example, because only two or four web applications may be running.

4.2.5.5 Additional communication

4.2.5.6 Communications processor

Point-to-point (S7-300, S7-400)

Overview of Freeport communication (S7-300, S7-400)

STEP 7 offers extended instructions that can be used for Freeport communication with a protocol specified in the user program. These instructions can be divided into two categories:

- Configuration instructions
- Communication instructions

Note

CPU capacity

The Freeport instructions communicate with the communication modules by reading or writing data records.
When using the instructions, you should therefore check the capacity of the CPU for reading and writing data records.
If several instructions need to read or write data records simultaneously on a CPU, the user program may need to create a time offset between the calls of each instruction.

Configuration instructions

Before the user program can start the Freeport communication, the communication interface and the parameters for sending and receiving of data must be configured.

The interface configuration and the data configuration can be set for each CM in the device configuration or with the following instructions of your user program:

- Port_Config (Page 6249)
- Send_Config (Page 6252)
- Receive_Config (Page 6253)
- P3964_Config (Page 6258)

NOTICE
Device configuration <-> Configuration instructions The device configuration parameters are transferred to the CM upon each Power On of the CPU (return of voltage). The parameters of the configuration instructions are transferred to the CM as defined in your user program. The parameters of the device configuration are not synchronized with the parameters of the configuration instructions, which means the parameters of the configuration instructions are not applied to the CPU device configuration. With your user program, you determine the parameters that apply in the CM and when they apply.

Communication instructions

The user program uses the instructions for Freeport communication to send data to and receive data from the communication interfaces. The CMs send data to and receive data from the communication stations.

- Send_P2P (Page 6260)
- Receive_P2P (Page 6263)

Note

Data consistency

- If the data to be sent is transmitted consistently, it cannot be changed after the positive edge at the REQ parameter until DONE has been set by the Send_P2P instruction.
- If the receive data is to be read consistently, it may only be evaluated in the cycle in which NDR = TRUE.

The receive buffer can be reset with additional instructions and special RS232 signals can be queried and set.

- Receive_Reset (Page 6265)
- Signal_Get (Page 6266)
- Signal_Set (Page 6267)

The following instructions let you read or write extended functions, as long as these are supported by the module.

- Get_Features (Page 6269)
- Set_Features (Page 6270)

All Freeport instructions work asynchronously. The instructions must therefore be called until the DONE output parameter indicates that execution is complete.

The user program can determine the send and receive status with the help of the query architecture. Send_P2P and Receive_P2P can be run at the same time. The communication

modules buffer the send and receive data as required until a module-specific maximum buffer size has been reached.

Note

Resolution of bit times

The number of bit times is specified with the configured data transmission rate for different parameters. Specifying the parameter in bit times makes it independent of the data transmission rate. All parameters with unit of bit times can be specified with a maximum number of 65535.

Using the instructions (S7-300, S7-400)

The Freeport instructions must be called cyclically to query received data or the end of transmission for a send process.

Depending on the data volume, data transmission may take place over several calls (program cycles).
If a command is completed with DONE = TRUE or NDR = TRUE, it has been executed without errors.

Note

Backing up STATUS

The DONE, NDR, ERROR and STATUS parameters are only available for one block cycle. To display the STATUS, you should therefore copy it to a free data area.

Master

Typical sequence for a master:

1. The Send_P2P instruction triggers transmission to the CM.
   Data transmission is initiated by a positive edge at the REQ input.
2. The Send_P2P instruction is executed in subsequent cycles to query the status of the transmission process.
3. When the Send_P2P instruction signals that transmission is complete at the DONE output, the user code can prepare the receipt of the answer.
4. The Receive_P2P instruction is run repeatedly to query an answer. If the CM has acquired response data, the Receive_P2P instruction copies the response to the CPU and signals that new data has been received at the NDR output.
5. The user program can process the response.
6. Back to step 1 and repetition of the sequence.

Slave

Typical sequence for a slave:

1. The user program runs the Receive_P2P instruction in each cycle.
2. If the CM has received a request, the Receive_P2P signals that new data is available at the NDR output and the request is copied to the CPU.
3. The user program processes the request and creates a response.
4. The response is returned to the master with the Send_P2P instruction.
5. The Send_P2P instruction must be run repeatedly to ensure that the send process is actually taking place.
6. Back to step 1 and repetition of the sequence.

The slave must ensure that Receive_P2P is called up often enough so that a transmission can be received by the master before it cancels the process due to a timeout while waiting for the response. To do so, the user program Receive_P2P can be called from within a cycle OB whose cycle time is sufficiently short so that the master can receive a transmission before the timeout setting expires. If the OB cycle time is set so that two runs can take place during the timeout setting of the master, the user program can receive all transmissions without any losses.

General parameters for Freeport operations (S7-300, S7-400)

Table 4-301 General input parameters of the Freeport instructions

Parameter	Description
REQ	Data transmission is initiated by a positive edge at the REQ input. Another edge at REQ may only be generated after the command has been completed (DONE or ERROR). Data transmission can take several calls (program cycles), depending on the data volume. When you add a Freeport instruction to your program, STEP 7 prompts you to specify the instance DB (or to have STEP 7 create a corresponding instance DB). Use a unique DB for each PtP instruction call.
PORT	A port address is assigned during configuration of the communication module. The PORT parameter communicates assignment to a specific communication module to the instruction. You can select a symbolic name for the standard port after the configuration. The assigned CM port value is the "Hardware ID" property of the device configuration with S7-1200/1500 and the "Input address" with S7-300/400. The symbolic port name is assigned in the symbol table.

The output parameters DONE, NDR, ERROR and STATUS of the Freeport instructions indicate the execution status of the Freeport functions.

Table 4-302 Output parameters DONE, NDR, ERROR and STATUS

Parameter	Data type	Standard	Description
DONE	Bool	FALSE	Set to TRUE for one cycle to indicate that the last request was completed with errors; otherwise FALSE.
NDR	Bool	FALSE	Set to TRUE for one cycle to indicate that new data has been received; otherwise FALSE.

Parameter	Data type	Standard	Description
ERROR	Bool	FALSE	Set to TRUE for one cycle to indicate that the last request was completed with errors; the corresponding error code can be found in STATUS; otherwise FALSE.
STATUS	Word	16#0000 or 16#7000	Result status: • If the DONE or NDR bit is set, STATUS is set to 0/16#7000 or to a specific error code. • If the ERROR bit is set, STATUS displays an error code. • If none of the bits listed above is set, the instruction can return status results that describe the current status of the function. The value in STATUS is valid until you call this instruction again (with one and the same port address).

Table 4-303 In/out parameter COM_RST

Parameter	Data type	Standard	Description
COM_RST	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. COM_RST is then set back to FALSE. Note: You must set COM_RST to TRUE during startup and should not subsequently change the parameter (that is, do not assign a value when you call the instruction). COM_RST is reset by the instruction following initialization of the instance DB.

Note

Please note that the parameters DONE, NDR, ERROR and STATUS are only set for one cycle.

Table 4-304 Shared error codes

Error code	Description
16#0000	No error
16#7000	Function not active
16#7001	Initial call after request started.
16#7002	Subsequent call after request started.
16#8x3A	Invalid pointer in parameter x

Table 4-305 Shared error classes of the STATUS parameter

Description of the class	Error classes	Description
Port configuration	16#81Ax	For the description of frequent errors in the interface configuration
Send configuration	16#81Bx	For the description of errors in the send configuration
Receive configuration	16#81Cx	For the description of errors in the receive configuration
Sending	16#81Dx	For the description of runtime errors during sending

Description of the class	Error classes	Description
Receiving	16#81Ex	For the description of runtime errors during receiving
RS232 accompanying signals	16#81Fx	For the description of errors in connection with signal processing

Port_Config: Configure PtP communication port (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Port_Config instruction (port configuration) allows you to change parameters such as the data transmission rate in runtime using your program. The data pending in the CM is deleted with the execution of Port_Config.

Configuration changes of Port_Config are saved on the CM and not in the CPU. When the voltage returns, the CM is configured with the data saved in the device configuration.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
PROTOCOL	IN	UInt	Word	0	Protocol • 0 = Freeport protocol • 1 = Protocol 3964(R)

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
BAUD	IN	UInt	Word	6	Data transmission rate of the port: • 1 = 300 bps • 2 = 600 bps • 3 = 1200 bps • 4 = 2400 bps • 5 = 4800 bps • 6 = 9600 bps • 7 = 19200 bps • 8 = 38400 bps • 9 = 57600 bps • 10 = 76800 bps • 11 = 115200 bps
PARITY	IN	UInt	Word	1	Parity of the port: • 1 = no parity • 2 = even parity • 3 = odd parity • 4 = mark parity • 5 = space parity • 6 = any
DATABITS	IN	UInt	Word	1	Bits per character: • 1 = 8 data bits • 2 = 7 data bits
STOPBITS	IN	UInt	Word	1	Stop bits: • 1 = 1 stop bit • 2 = 2 stop bits
FLOWCTRL	IN	UInt	Word	1	Flow control: • 1 = no flow control • 2 = XON/XOFF • 3 = Hardware RTS always ON • 4 = Hardware RTS switched • 5 = Hardware RTS always ON, ignore DTR/DSR
XON-CHAR	IN	Char		16#0011	Specifies the character that serves as XON character. It is typically a DC1 character (11H). This parameter is only evaluated when software flow control is active.
XOFF-CHAR	IN	Char		16#0013	Specifies the character that serves as XOFF character. It is typically a DC3 character (13H). This parameter is only evaluated when software flow control is active.

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
WAITIME	IN	UInt	Word	2000	Specifies how long to wait for an XON character after receipt of an XOFF character or how long to wait for a CTS = ON signal after CTS = OFF (0 to 65535 ms). This parameter is only evaluated when flow control is active.
MODE	IN	USInt	Byte	0	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master; CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave; CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode [1)]
LINE_PRE	IN	USInt	Byte	0	Receive line initial state Valid initial states are: • 0 = "No" initial state [1)] • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire mode (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
BRK_DET	IN	USInt	Byte	0	Break detection The following settings are permitted: • 0 = break detection deactivated • 1 = break detection activated
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))

[1)] Required setting for the use of PROFIBUS cables with CM 1241 for RS485

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

Send_Config: Configure PtP sender (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Send_Config instruction (send configuration) allows you to change send parameters in runtime using your program (conditions that identify the start and the end of the data to be sent). Any data pending in a CM is deleted when Send_Config is executed.

Configuration changes of Send_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Activates the configuration change upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
RTSOND-LY	IN	UInt	Word	0	Number of milliseconds to wait after activation of RTS before a transmission of send data is started. This parameter is only valid if the hardware flow control is active. The valid range is 0 to 65535 ms. The value 0 deactivates the function.
RTSOFFDLY	IN	UInt	Word	0	Number of milliseconds to wait after transmission of send data before RTS is deactivated: This parameter is only valid if the hardware flow control is active. The valid range is 0 to 65535 ms. The value 0 deactivates the function.

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
BREAK	IN	UInt	Word	0	This parameter specifies that a BREAK is to be sent at the start of each frame for the specified number of bit times. The maximum is 65535 bit times. The value 0 deactivates the function.
IDLELINE	IN	UInt	Word	0	This parameter specifies that the line is to remain in idle for the specified number of bit times prior to the start of each frame. The maximum is 65535 bit times. The value 0 deactivates the function.
USR_END	IN	STRING[2]		0	Input of end delimiters. No more than 2 end delimiters can be configured. All data including the end delimiter(s) is sent, independent of the configured frame length.
APP_END	IN	STRING[5]		0	Input of characters to be appended. You can append up to 5 characters.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

Receive_Config: Configure PtP recipient (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Receive_Config instruction (receive configuration) allows you to change receive parameters in runtime using your program. This instruction configures the conditions that mark the start and the end of received data. Any data pending in a CM is deleted when Receive_Config is executed.

Configuration changes of Receive_Config are saved **non-retentive** on the CM. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module. The Receive_Config instruction therefore must be called again from the user program when the voltage returns to the CPU or to the communication module in order to overwrite the parameters stored in the device configuration.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Activates the configuration change upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
RECEIVE_CONDITIONS	IN	Variant	Any	-	The data structure of Receive_Conditions specifies the start and end conditions used to identify the start and end of a frame.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

Start conditions for the Receive_P2P instruction

The Receive_P2P instruction uses the configuration specified in the device configuration or by the Receive_Config instruction to determine the start and end of Freeport communication frames. The start of the frame is defined by the start conditions. The start of the frame can be determined with one or several start conditions.
If Break as well as Idle Line is activated, Break must be met first and then Idle Line as well. After that, one of the other conditions (start character or start sequence) is sufficient to start data transmission.
The start condition "Any character" cannot be combined with other start conditions.

Data type structure of the Receive_Conditions parameter, part 1 (start conditions)

Table 4-306 Structure of Receive_Conditions for start conditions

Parameter	Declaration	Data type	Default	Description
START.START-COND	IN	Word	16#0002	Specifying the start condition • 01H - detection of the start character • 02H - Any character • 04H - detection of a line break • 08H - detection of an idle line • 10H - detection of start sequence 1 • 20H - detection of start sequence 2 • 40H - detection of start sequence 3 • 80H - detection of start sequence 4 The start conditions can be combined by adding the values together.
START.IDLE-TIME	IN	Word	16#0028	The number of bit times required in the idle state for a new frame start to be detected (default value: W#16#28). Only in connection with the condition "Detection of an idle line". 0 to FFFF
START.START CHAR	IN	Byte	16#0002	The start character for the condition "Start character". (default value: B#16#2)
START.SEQ[1]. CTL	IN	Byte	0	Start sequence 1, deactivate/activate comparison for each character: (default value: B#16#0) These are the activation bits for each character of the start character string. • 01H - character 1 • 02H - character 2 • 04H - character 3 • 08H - character 4 • 10H - character 5 When a bit is deactivated for a specific character, this means that each character in this position in the character string represents a valid start character string (e.g. 1FH = all 5 characters interpreted).
START.SEQ[1]. STR[1] .. START.SEQ[1]. STR.[5]	IN	Char[5]	0	Start sequence 1, start character (5 characters).
START.SEQ[2]. CTL	IN	Byte	0	Start sequence 2, deactivate/activate comparison for each character. Default value: B#16#0)
START.SEQ[2]. STR[1] .. START.SEQ[2]. STR.[5]	IN	Char[5]	0	Start sequence 2, start character (5 characters).
START.SEQ[3]. CTL	IN	Byte	0	Start sequence 3, deactivate/activate comparison for each character. Default value: B#16#0

Parameter	Declaration	Data type	Default	Description
START.SEQ[3]. STR[1] .. START.SEQ[3]. STR.[5]	IN	Char[5]	0	Start sequence 3, start character (5 characters).
START.SEQ[4]. CTL	IN	Byte	0	Start sequence 4, deactivate/activate comparison for each character. Default value: B#16#0
START.SEQ[4]. STR[1] .. START.SEQ[4]. STR.[5]	IN	Char[5]	0	Start sequence 4, start character (5 characters),

Example

Have a look at the following received data in hexadecimal coding: "68 10 aa 68 bb 10 aa 16". The configured start character strings are available in the following table. Start character strings are evaluated once the first character 68H has been successfully received. After the fourth character has been successfully received (the second 68H), start condition 1 has been met. Once the start conditions have been met, the evaluation of the end conditions starts.

Processing of the start character string can be canceled due to different errors in parity, framing or time intervals between characters. These errors prevent receipt of the data because the start condition has not been met (an error message is output).

Table 4-307 Start conditions

Start condition	First character	First character +1	First character +2	First character +3	First character +4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the Receive_P2P instruction

The end of a frame is defined by the first occurrence of one or more configured end conditions.

You can configure the end conditions either in the properties of the communication interface in the device configuration, or with the Receive_Config instruction. The receive parameters (start and end conditions) are reset to the settings in the device configuration each time the voltage returns to the CPU or the communication module. When the STEP 7 user program executes Receive_Config, the settings are set to the parameters of Receive_Config .

Data type structure of the Receive_Conditions parameter, part 2 (end conditions)

Table 4-308 Structure of Receive_Conditions for end conditions

Parameter	Declaration	Data type	Default	Description
END.END-COND	IN	Word	0	This parameter specifies the condition for the frame end: • 01H - response timeout • 02H - message timeout • 04H - character delay time • 08H - maximum frame length • 10H - read message length from message (N+LEN+M) • 20H - end sequence • 40H - fixed frame length
END.FIXLEN	IN	Word	1	Fixed frame length: Only used if the end condition "Fixed frame length" has been selected. 1 to 4000 bytes (up to 4 KB depending on the module)
END.MAXLEN	IN	Word	1	Maximum frame length: Only used if the end condition "Maximum frame length" has been selected. 1 to 4000 bytes (up to 4 KB depending on the module)
END.N	IN	Word	0	Byte position of the length field in the frame. Only used with end condition N+LEN+M. 1 to 4000 bytes (up to 4 KB depending on the module)
END.LENGTH-SIZE	IN	Word	0	Size of the length field (1, 2, or 4 bytes). Only used with end condition N+LEN+M.
END.LENGTHM	IN	Word	0	Number of characters after the length field that are not included in the value of the length field. This entry is only used with end condition N+LEN+M. 0 to 255 bytes
END.RCVTIME	IN	Word	200	Specify the wait time for the first received character after a frame has been sent. The receive instruction is terminated with an error message if a character is not received within the specified time. This information is used only with the condition "Response timeout". (0 to 65535 ms). Note: This parameter cannot be used as sole end criterion but only in connection with at least one other end condition.
END.MSGTIME	IN	Word	200	Specify how long to wait for receipt of the complete frame after receipt of the first character. This parameter is used only if the condition "Message timeout" is selected. (0 to 65535 ms)
END.CHARGAP	IN	Word	12	Enter the maximum number of bit times between characters. If the number of bit times between characters exceeds the specified value, the end condition has been met. This information is used only with the condition "Character delay time". (0 to 65535 bit times)

Parameter	Declaration	Data type	Default	Description
END.SEQ.CTL	IN	Byte	0	Character sequence 1, deactivate/activate comparison for each character: These are the activation bits for each character of the end character string. Character 1 is bit 0, character 2 is bit 1, ..., character 5 is bit 4. If a bit is deactivated for a specific character, this means that each character represents a congruence at this position of the character string.
END.SEQ.STR[1] .. END.SEQ.STR[5]	IN	Char[5]	0	Character string 1, start character (5 characters)

Table 4-309 General parameters of the Receive_P2P instruction

Parameter	Declaration	Data type	Default	Description
GENERAL.MBUF_SIZE	IN	Byte	255	Input number of frames that are to be buffered in the receive buffer of the CM. If no other conditions are active that influence the reaction of the receive buffer (prevent timeout, data flow control), additional frames are discarded once the limit has been reached. (1 to 255 frames)
GENERAL.OW_PROT	IN	Byte	0	Activates the no overwriting function of the buffered frame if the CM receives a new frame and the receive buffer of the CM was not yet read. This step prevents already buffered received frames from being lost. • 0 - not activated • 1 - activated
GENERAL.CLR_MBUF	IN	Byte	0	Activates deletion of the receive buffer during CPU startup. The receive buffer is automatically deleted when the CPU switches from STOP to RUN. The receive buffer only contains frames received after CPU startup. • 0 - not activated • 1 - activated

P3964_Config: Configuring the 3964(R) protocol (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The P3964_Config instruction (protocol configuration) allows you to change protocol parameters for 3964(R), such as character delay time, priority and block check, in runtime using your program.

Configuration changes of P3964_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
BCC	IN	USInt	Byte	1	Activates/deactivates the use of the block check • 0 = without block check • 1 = with block check
Priority	IN	USInt	Byte	1	Selection of the priority • 0 = low priority • 1 = high priority
CharacterDelayTime	IN	UInt	Word	16#00DC	Setting the character delay time (depending on the set data transmission rate) (default value: 220 ms) 1 ms to 65535 ms
AcknDelayTime	IN	UInt	Word	16#07D0	Setting the acknowledgment delay time (depending on the set data transmission rate) (default value: 2000 ms) 1 ms to 65535 ms
BuildupAttempts	IN	USInt	Byte	16#0006	Setting the number of connection attempts (default value: 6 connection attempts) 1 to 255
RepetitionAttempts	IN	USInt	Byte	16#0006	Setting the number of transmission attempts (default value: 6 connection attempts) 1 to 255
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

Send_P2P: Sending data (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Send_P2P instruction (send point-to-point data) starts the transmission of data and transmits the contents of the assigned buffer to the communication module. The CPU program is still being executed while the CM sends the data with the data transmission rate. Only one send instruction per communication module may be pending at any time. The CM signals an error if a second Send_P2P instruction is executed while the CM is already sending a frame.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
BUFFER	IN	Variant	Any	0	This parameter points to the memory area of the send buffer. Notes: • Boolean data and Boolean fields are not supported. • If the send buffer is in the optimized memory area, the maximum permitted length of the sent data is 1024 bytes. Exception: Arrays of Byte, Word or DWord are supported up to a length of 4096 bytes. • If the send buffer is a String or WString, the content of the string is transferred without the current and maximum length. Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 6262)"
LENGTH	IN	UInt	Word	0	Length in bytes of the data to be transferred. The memory area addressed in the BUFFER parameter is completely transmitted with LENGTH = 0. Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 6262)"
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Send_P2P instruction The instruction is initialized with 1. The instruction then resets COM_RST to 0. Note: The parameter is only available for S7-300/400 instructions.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

Parameters

The DONE and ERROR outputs are in FALSE status when a send instruction is being processed. At the end of the send instruction, one of the DONE or ERROR outputs is set to TRUE for one cycle to signal the status of the send instruction. The error code at the STATUS output can be evaluated when the status of ERROR is TRUE.

The instruction outputs the status 16#7001 when the communication interface accepts the send data. Subsequent executions of Send_P2P output the value 16#7002 if the CM is still sending. At the end of the send instruction, the CM outputs the status 16#0000 for the send

instruction (if no error has occurred). Subsequent executions of Send_P2P with REQ = 0 output the status 16#7000 (free).

The diagram below shows the relationship between the output values and REQ. It is based on the assumption that the instruction is called cyclically to check the status of the send process (indicated by the STATUS values).

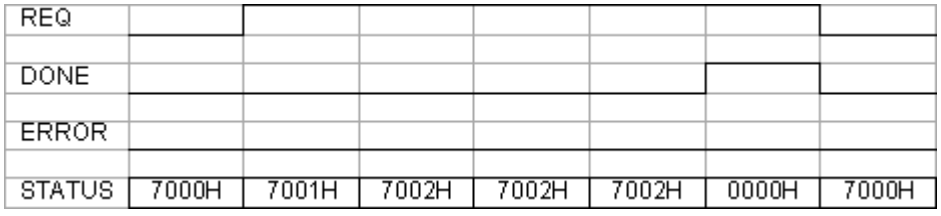

The figure below shows how the DONE and STATUS parameters are only valid for one cycle if a pulse is pending at the REQ line (for one cycle) to trigger the send instruction.

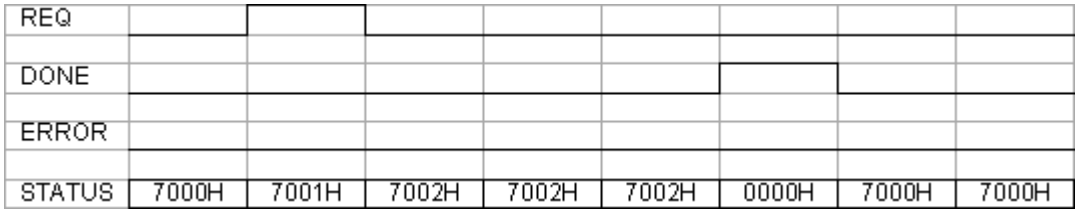

The figure below shows the relationship of the DONE, ERROR and STATUS parameters in case of an error.

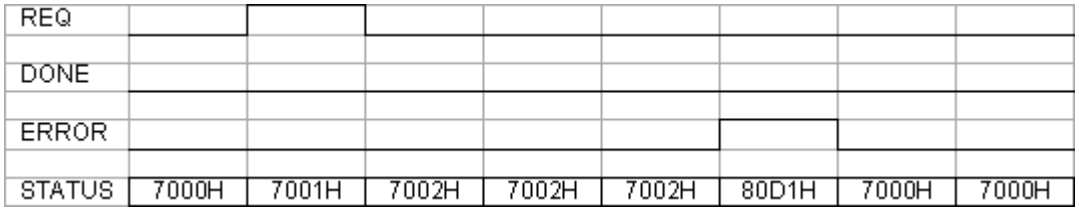

The DONE, ERROR and STATUS values are only valid until Send_P2P is executed again with the same instance DB.

Using the BUFFER and LENGTH parameters for communication operations (S7-300, S7-400)

Interaction of BUFFER and LENGTH parameters for Send_P2P

The minimum data size sent by the Send_P2P instruction is one byte.

The BUFFER parameter specifies the size of the data to be sent if a "0" is passed at the LENGTH parameter during call. The specification of a tag is sufficient for this.

You cannot use the data type Bool or arrays of the Bool type for the BUFFER parameter. If large amounts of data are being transferred we recommend the mapping to the array or structure data types.

Table 4-310 BUFFER parameter

BUFFER	Description
Elementary data type	When sending: The LENGTH value must include the byte size of this data type. Example: For a Word value, the LENGTH must be two. For a DWord value or Real value, the LENGTH must be four.
Structure	For optimized memory: The maximum permitted length of the BUFFER is 1024 Byte; otherwise 4 KB are permitted depending on the module. For sending: The LENGTH value can include a byte size smaller than the complete byte length of the structure; in this case, only the first LENGTH bytes of the structure from BUFFER are sent.
Array	For optimized memory: If the array data type is not equal to Byte, Word or DWord, the maximum permitted buffer length is 1024 Byte. Depending on the module, up to 4 KB can be transmitted if the memory is not optimized, independent of the data structure. For sending: The LENGTH value can include a byte size smaller than the complete byte length of the array, whereby, this byte size is a multiple of the byte size of the data element. Example: The LENGTH parameter of an array of the Word type must be a multiple of two and a multiple of four for an array of the Real type. If LENGTH is specified, the number of array elements is transmitted. If BUFFER includes an array with 15 DWord elements (a total of 60 Bytes), for example, and you specify LENGTH = 20, the first five DWord elements from the array are transmitted.
String	The LENGTH parameter includes the number or characters to be sent. Only the characters of the String are transmitted. The bytes with the maximum and actual length of the String are not sent.

Table 4-311 LENGTH parameter

LENGTH	Description
= 0	The complete content of the memory area specified by BUFFER is transferred. If BUFFER points to a string, the entire content of the string is transferred, without the bytes with the maximum and actual length.
> 0	The content up to the configured length of the memory area specified by BUFFER is transferred.

Receive_P2P: Receiving data (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Receive_P2P instruction (receive data using point-to-point communication) checks the frames received in the CM. If a frame is available, it is transmitted from the CM to the CPU. A receive error is indicated at the STATUS parameter.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
BUFFER	IN	Variant	Any	0	This parameter points to the start address of the receive buffer. This buffer must be large enough to receive the maximum frame length. Note: • Boolean data or Boolean fields are not supported. • If the receive buffer is in the optimized memory area, the maximum permitted length of the received data is 1024 bytes. Exception: Arrays of Byte, Word or DWord are supported up to a length of 4096 bytes. • If the receive buffer is a String or WString, the received data is written to the content of the string and the current length of the string is set accordingly. Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 6262)"
LENGTH	OUT	UInt	Word	0	Length of the frame received in bytes Additional information under "Using the BUFFER and LENGTH parameters for communication operations (Page 6262)"
NDR	OUT	Bool		FALSE	TRUE for one cycle if new data is available and the instruction has been completed without errors.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

The error code at the STATUS output can be evaluated if the status of ERROR is TRUE. The STATUS value provides the reason for terminating the receive operation in the CM.
This is usually a positive value which indicates that the receive operation has been successful and the frame criterion that has been detected.
If the STATUS value is negative (the most significant bit of the hexadecimal value is set), the receive operation was terminated due to an error condition, such as a parity, framing or overflow error.

Each communication module can buffer a module-specific number for frames. If several frames are available in the CM, the Receive_P2P instruction outputs the oldest available frame (FIFO).

Receive_Reset: Clear receive buffer (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Receive_Reset instruction (reset receiver) clears the receive buffer in the CM.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle means that the last request was completed without errors.

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
ERROR	OUT	Bool		FALSE	TRUE means that the last request was completed with errors. If this output is TRUE, the STATUS output contains the corresponding error codes.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

Signal_Get: Read status (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Signal_Get instruction (get RS232 signals) reads the current states of the RS232 accompanying signals and displays them at the corresponding instruction outputs.

Note

Restriction

- This instruction can only be used with CMs RS232 BA and RS232 HF.
- If RS232C is set for the operating mode, this instruction can also be used with CM PtP (ET200SP).

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
NDR	OUT	Bool		FALSE	TRUE for one cycle if the RS232 accompanying signals have been read and the instruction has been completed without errors.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))
DTR	OUT	Bool		FALSE	Data device ready, module ready (output)
DSR	OUT	Bool		FALSE	Data device ready, communication station ready (input)
RTS	OUT	Bool		FALSE	Send request, module ready to send (output)
CTS	OUT	Bool		FALSE	Ready to send, communication station can receive data (input)
DCD	OUT	Bool		FALSE	Data carrier signal detected, signal level received
RING	OUT	Bool		FALSE	Call display, signaling incoming call

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

Signal_Set: Set accompanying signals (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Signal_Set instruction (set RS232 signals) allows you to set the RS232 communication signals.

Note

Restrictions

- This instruction can only be used with CMs RS232 BA and RS232 HF.
- If RS232C is set for the operating mode, this instruction can also be used with CM PtP (ET200SP).

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
SIGNAL	IN	Byte		0	Selection of the signal to be set (more than one possible): • 01H = RTS • 02H = DTR • 04H = DSR (for interface type DCE only)
RTS	IN	Bool		FALSE	Send request, module ready to send Set this value at the output (TRUE or FALSE), default value: FALSE
DTR	IN	Bool		FALSE	Data terminal ready, module ready Set this value at the output (TRUE or FALSE), default value: FALSE
DSR	IN	Bool		FALSE	Data terminal ready (for DCE interface type only), not used.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

Get_Features: Get extended functions (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

If supported by the module, you can use the Get_Features instruction (get extended functions) to get information on the ability of the module to support CRC and to generate diagnostic messages.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	PORT	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
NDR	OUT	Bool		FALSE	TRUE for one cycle if new data is available and the instruction has been completed without errors

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
MODBUS_CRC	OUT	Bool		FALSE	Modbus CRC support
DIAG_ALARM	OUT	Bool		FALSE	Generation of diagnostic messages
SUPPLY_VOLT	OUT	Bool		FALSE	Diagnostics for missing supply voltage L+ is available
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

Set_Features: Set extended functions (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

If supported by the module, you can use the Set_Features instruction (select extended functions) to activate CRC support and the generation of diagnostic messages.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	The instruction to set extended functions is started upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
EN_MODBUS_CRC	IN	Bool		FALSE	Activate Modbus CRC support
EN_DIAG_ALARM	IN	Bool		FALSE	Activate generation of diagnostic messages
EN_SUPPLY_VOLT	IN	Bool		FALSE	Enable diagnostics for missing supply voltage L+
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one execution once the last request is completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6271))

Additional information about the general parameters is available at "General parameters for Freeport operations (Page 6247)".

Error messages (S7-300, S7-400)

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		

Error code	Description	Remedy
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#7002	Interim call: Data transmission running	-
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.

Error code	Description	Remedy
16#8193	The module does not support supply voltage diagnostics L+.	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire operation (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire operation. (4)

Error code	Description	Remedy
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see AUTOHOTSPOT).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)

Error code	Description	Remedy
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (bytes) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer on startup is deactivated (0) • Clear receive buffer on startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)

Error code	Description	Remedy
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end delimiters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
16#81DF	The module has reset the interface to the FB for one of the following reasons: • Module was restarted • Module parameters were reassigned • CPU STOP	—
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Break in receive line to partner. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (bytes) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		

Error code	Description	Remedy
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
16#81F2	The DSR signal cannot be set because the module has the type DTE.	Check the configured type of the module. The module type must be DCE (data communication equipment).
16#81F3	The DTR signal cannot be set because the module has the type DCE.	Check the configured type of the module. The module type must be DTE (data terminal equipment).
16#81F4	Block header error (e.g. incorrect block type or incorrect block length)	Check the instance DB and the block header.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditions points to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 KB • Non-optimized memory area: 4 KB Note: If the pointer points to an optimized memory area, do not send more than 1 KB.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in the RDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter before the 1st call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.

Error code	Description	Remedy
16#8301 [1]	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 [1]	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 [1]	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 [1]	Receive_Conditions does not point to a structure of the typeReceive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.

Error code	Description	Remedy
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the frame header and data string or the connection is terminated with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

1) Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.

Error code	Description	Remedy
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.

Error code	Description	Remedy
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 6323) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 6323) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 6323) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#80C9	The slave does not respond within the time set by Blocked_Proc_Timeout.	Check the setting for Blocked_Proc_Timeout. Check if the module has been configured with the Modbus_Comm_Load instruction. The module may possibly need to be reconfigured using Modbus_Comm_Load after a pull/plug or after voltage recovery.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.

Error code	Description	Remedy
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1)]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1)]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1)]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1)]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1)]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		
16#8428 [1)]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during call of SFC51	Call the Modbus_Slave instruction again
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200% to +200%
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request. Check if the instructions USS_Read_Param, USS_Write_Param or USS_Port_Scan have reported an error. If they have, check the value of the static tag USS_DB. w_USSExtendedError of the USS_Drive_Control instruction.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.

Error code	Description	Remedy
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

USS (S7-300, S7-400)

Overview of USS communication (S7-300, S7-400)

USS communication

The USS instructions control the operation of drives which support the protocol of the universal serial interface (USS). You can communicate with several drives by means of RS485 connections and the USS instructions with the PtP communication modules. Each RS485 port can operate up to 16 drives.

The USS protocol uses a master-slave network for communication via a serial bus. The master uses an address parameter to send data to a selected slave. A slave cannot send without having first received a send request. Communication between individual slaves is not possible. USS communication takes place in half-duplex mode. The figure below shows a network diagram for a sample application with 16 drives.

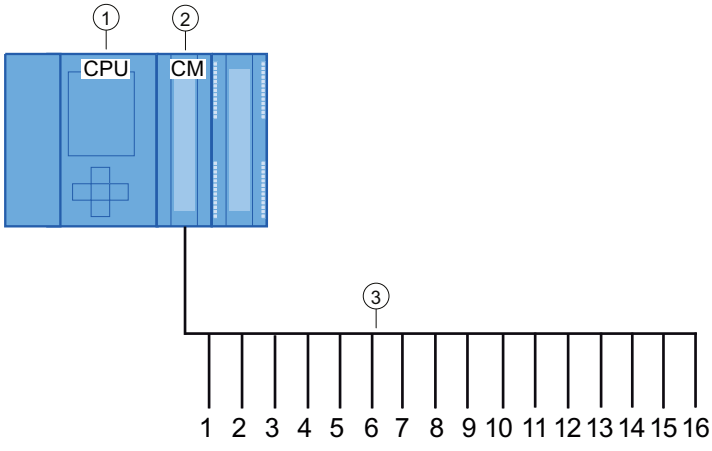

① CPU
② CM
③ USS drives in a USS network

Figure 4-31 Wiring example with S7-1500 communication module

Note

Communicating with a drive via RS232

You can basically also use CM PtP RS232 BA and CM PtP RS232 HF for communication with a drive. However, only **one** drive can be connected to an RS232 port.

Communicating with a drive via RS422

You can basically also use the RS422 interface of the CM PtP RS422/485 BA and CM PtP RS422/485 HF for communication with a drive. However, only **one** drive can be connected to an RS422 port.

USS instructions in your program

- USS_Port_Scan: The instruction USS_Port_Scan allows you to communicate via a communication module with up to 16 drives using a USS network (must be called cyclically). There is only one USS_Port_Scan instruction per PtP communication port in the program, and it controls transmission to all drives.
- USS_Drive_Control: The instruction USS_Drive_Control allows you to prepare the send data from USS_Port_Scan for a drive and display its receive data. USS_Drive_Control configures the data to be sent and evaluates the data received in a previous request from USS_Port_Scan.
- USS_Read_Param: The instruction USS_Read_Param allows you to read parameters from a drive.
- USS_Write_Param: The instruction USS_Write_Param allows you to change parameters in a drive.

Requirements for using the USS protocol (S7-300, S7-400)

The four USS instructions use 2 FBs and 2 FCs to support the USS protocol. For each USS network, one instance data block (DB) is used for USS_Port_Scan and one instance data block for all calls of USS_Drive_Conrol.

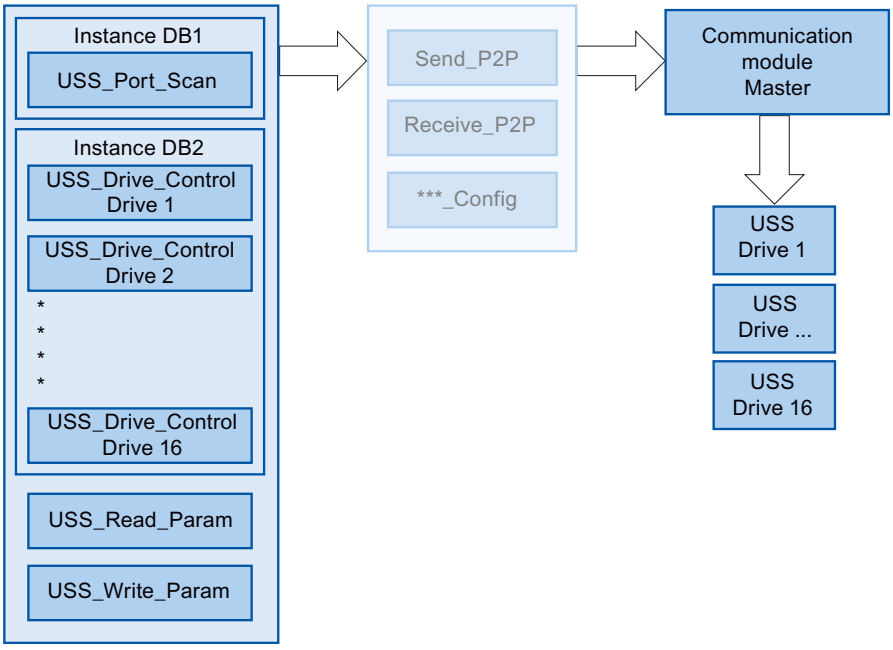

Figure 4-32 USS program sequence

All drives (max. 16) that are connected to one RS485 port are part of the same USS network. All drives that are connected to another RS485 port are part of a different USS network. Each USS network is managed using a unique instance data block for all USS_Drive_Control instructions and one further instance data block for the USS_Port_Scan instruction. All instructions that are part of a USS network must share this instance data block for USS_Drive_Conrol. The USS_Port_Scan, USS_Read_Param and USS_Write_Param instructions have the USS_DB parameter for this function. This parameter must be connected to the (static) USS_DB parameter of the instance DB of the USS_Drive_Control instruction.

- The instructions USS_Drive_Control and USS_Port_Scan are function blocks (FB). If you add the USS_Drive_Control or USS_Port_Scan instruction to the programming editor, you are prompted to assign a DB for this FB in the "Call options" dialog. If it is the first USS_Drive_Control instruction in this program for this USS network, you can apply the DB standard assignment (or change the name, if necessary) and the new DB is created for you. If, however, it is not the first USS_Drive_Control instruction for this drive, you need to select the DB which has already been assigned to this USS network from the drop-down list in the "Call options" dialog.
- The USS_Write_Param and USS_Read_Param instructions are functions (FCs). No DB is assigned when you add these FCs in the editor. If you add these FCs or the USS_Port_Scan instruction in the editor, you need to assign the USS_DB parameter of the corresponding USS_Drive_Control instance DB to the USS_DB input of these instructions. Double-click the parameter field and click the symbol to display the available DBs. Enter a period "." and select the USS_DB parameter from the drop-down list.

- The USS_Port_Scan function controls communication between the CPU and the drives via the point-to-point RS485 communication port. A communication with the drive is processed every time you call this function. Your program must call this function fast enough so that the drives do not signal a timeout. To ensure a constant time response for frame communication, you should call this instruction in a cyclic interrupt OB.
- The USS_Drive_Control instruction gives your program access to a specified drive in the USS network. Its inputs and outputs correspond to the states and operating functions of the drive. If there are 16 drives in your network, you must call USS_Drive_Control at least 16 times in your program, i.e. once for each drive.
  You should only call the USS_Drive_Control instruction from a cyclic OB.
- The operating parameters of the drive are read and written with the USS_Read_Param and USS_Write_Param functions. These parameters control the internal operation of the drive. See the drive manual for a definition of these parameters. Your program may include any number of these functions, but only one read or write request may be active for a drive at any particular time. You may only call the USS_Read_Param and USS_Write_Param functions from the cycle OB of a main program.

NOTICE
USS instruction calls Only ever call USS_Drive_Control, USS_Read_Param or USS_Write_Param from a cycle OB of the main program. The USS_Port_Scan instruction function can be called from any OB, but it is usually called from a cyclic interrupt OB. Do not use the USS_Drive_Control, USS_Read_Param or USS_Write_Param instruction in an OB with a higher priority than the corresponding USS_Port_Scan instruction. For example, do not add USS_Port_Scan to the main program or USS_Read_Param to a cyclic interrupt OB. You may encounter unexpected errors if the execution of USS_Port_Scan is interrupted by another instruction.

Note

Parameter ID value

You need to configure the use of 4 PIV words (ParameterIDValue) for the drives.

Calculating time for communication with the drive

Communication with the drive takes place asynchronously to the cycle of the CPU. The CPU usually runs through several cycles before communication with a drive is complete.

To ensure that the watchdog set for the drive is not triggered, the send frames must be sent to the drive within the watchdog time. You must allow for the number of any retries which may be needed to complete the transaction if communication errors occur. By default, up to 2 retries are made for each transaction with the USS protocol.
The maximum time interval between two send frames is calculated as follows:

N * (5 * cycle time + frame runtime + timeout of the receive frame max.) * (number of transmission attempts)

N	Number of drives in this network
Factor 5	5 cycles are typically needed for sending and receiving frames.
Cycle time	Max. cycle time of the cyclic interrupt OB in which the USS_Port_Scan instruction is called.
Frame runtime	Frame time = (number of characters per frame) * (11 Bit per character) / (data transmission rate in Bit/s)
Number of transmission attempts	Number of retries + 1
Timeout of the receive frame	RCVTIME (if no response is received from the drive)
Timeout of the receive frame max.	RCVTIME + MSGTIME (if an incomplete answer is received shortly before the expiration of RCVTIME and monitoring of MSGTIME has expired **or** if a response is still being processed at the expiration of RCVTIME, the timeout is extended by the MSGTIME)

The following times apply for "Received frame timeout" (ms):

Bit/s	115200	57600	38400	19200	9600	4800	2400	1200
Receive_Conditions.END.RCVTIME	25	29	33	56	72	100	100	100
Receive_Conditions.END.MSGTIME	25	29	33	56	72	124	240	460

Timeout of the receive frame max. = (Receive_Conditions.END.RCVTIME (0.072 s) + Receive_Conditions.END.MSGTIME (0.072 s))

Example:

5 drives
Data transmission rate = 9600 bps
28 characters per frame
Cycle time = 0.020 s
Number of retries = 2

Time interval = 5 * (5 * 0.02 s + 1 * 28 * 11 / 9600 s + 0.072 s + 0.072 s) * 3 = 3 s

The time monitoring of the drive must be set to about 3 seconds in this case.

USS_Port_Scan / USS_Port_Scan_31: Communication by means of a USS network (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Port_Scan_31 instruction

It is only possible to use the USS_Port_Scan_31 instruction on an S7-1500 CPU.

Description

The USS_Port_Scan instruction processes the communication by means of a USS network for max. 16 drives.

The USS_Port_Scan_31 instruction processes the communication by means of a USS network for max. 31 drives.

STEP 7 automatically creates the instance DB when you add the instruction.

Parameters

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
PORT	IN	Port	Word	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.
BAUD	IN	DInt		9600	Data transmission rate for USS communication The following are valid: • 1200 bps • 2400 bps • 4800 bps • 9600 bps • 19200 bps • 38400 bps • 57600 bps • 115200 bps
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control / USS_Drive_Control_31 instruction to your program.

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
COM_RST	INOUT	---	Bool	FALSE	Initialization of the USS_Port_Scan / USS_Port_Scan_31 instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.
ERROR	OUT	Bool		FALSE	If TRUE, this output indicates that an error has occurred and that the STATUS output is valid. You may need to check the value of the static tag USS_DB.w_USSExtendedError in the instance DB of the USS_Drive_Control / USS_Drive_Control_31 instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 6304)).

There is only one USS_Port_Scan / USS_Port_Scan_31 instruction per PtP communication port in the program and each call of this instruction controls a transmission to or from all drives in this network. All USS functions assigned to one USS network and one PtP communication port must use the same instance DB.

Your program must execute the USS_Port_Scan / USS_Port_Scan_31 instruction often enough to prevent a timeout in the drive (see Requirements for using the USS protocol (Page 6287) "Calculating time for communication with the drive").

The USS_Port_Scan / USS_Port_Scan_31 instruction is usually called from a cyclic interrupt OB to prevent drive timeouts and to make the last USS data updates available for calls of USS_Drive_Control / USS_Drive_Control_31 .

USS_Port_Scan / USS_Port_Scan_31 data block tags

The table below shows the public static tags in the instance DB of USS_Port_Scan / USS_Port_Scan_31 that you can use in your program.

Table 4-312 Static tags in the instance DB

Tag	Data type	Standard	Description
MODE	USInt	4	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master; CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave; CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode [1]
LINE_PRE	USInt	2	Receive line initial state Valid initial states are: • 0 = "No" initial state [1] • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire mode (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
BRK_DET	USInt	0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
RETRIES_MAX	SInt/Byte	2	Number of retries when communication errors occur. You can use this parameter to set the number of send retries for a request frame if the response frame is not received within the set time.
EN_DI-AG_ALARM	Bool	0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
EN_SUP-PLY_VOLT	Bool	0	Enable diagnostics for missing supply voltage L+ • 0 - not activated • 1 - activated

[1] Required setting for the use of PROFIBUS cables with CM 1241 for RS485

Version 2.5 is functionally identical to version 2.4 and its version number was only incremented due to internal measures.

Instruction versions

USS_Port_Scan:

Version 2.5 is functionally identical to version 2.4 and its version number was only incremented due to internal measures.

USS_Port_Scan_31:

Version 1.2 is functionally identical to version 1.1 and its version number was only incremented due to internal measures.

USS_Drive_Control / USS_Drive_Control_31: Preparing and displaying data for the drive (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Drive_Control_31 instruction

It is only possible to use the USS_Drive_Control_31 instruction on an S7-1500 CPU.

Description

The USS_Drive_Control instruction prepares send data for max. 16 drives and evaluates the response data of the drives.

The USS_Drive_Control_31 instruction prepares send data for max. 31 drives and evaluates the response data of the drives.

You need to use a separate instance of the instruction for each drive, and all USS functions assigned to one USS network and one PtP communication port must use the same instance DB. You must enter the DB name when you add the first USS_Drive_Control / USS_Drive_Control_31 instruction. You then refer to this DB that was created when the first instruction was added.

STEP 7 automatically creates the DB when you add the instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/ WinAC		
RUN	IN	Bool		FALSE	Start bit of the drive: If this parameter is TRUE, this input allows you to operate the drive with the preset speed. If RUN changes to FALSE during operation of the drive, the motor coasts to a standstill. This behavior differs from disconnection of the power supply (OFF2) and braking of the motor (OFF3).
OFF2	IN	Bool		FALSE	"Coast to standstill" bit: If this parameter is FALSE, this bit causes the drive to coast to a standstill without braking.
OFF3	IN	Bool		FALSE	Fast stop bit: If this parameter is FALSE, this bit causes a fast stop by braking the drive.
F_ACK	IN	Bool		FALSE	Error acknowledgment bit: This bit resets the error bit of a drive. The bit is set after clearing the error and the drive detects this way that the previous error no longer has to be reported.
DIR	IN	Bool		FALSE	Direction control of the drive: This bit is set if the drive is to run forward (if SPEED_SP is positive; see table "Interaction of SPEED_SP and DIR parameters").
DRIVE	IN	USInt	Byte	1	Address of the drive: This input is the address of the USS drive. The valid range is between drive 1 and drive 16.
PZD_LEN	IN	USInt	Byte	2	Word length: This is the number of PZD data words. Valid values are 2, 4, 6 or 8 words.
SPEED_SP	IN	Real		0.0	Speed setpoint: This is the speed of the drive as a percentage of the configured frequency. A positive value means that the drive runs forward (if DIR is true). The valid range is 200.00 to -200.00.
CTRL3	IN	Word		0	Control word 3: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL4	IN	Word		0	Control word 4: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL5	IN	Word		0	Control word 5: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL6	IN	Word		0	Control word 6: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL7	IN	Word		0	Control word 7: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL8	IN	Word		0	Control word 8: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/ WinAC		
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the USS_Drive_Control / USS_Drive_Control_31 instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.
NDR	OUT	Bool		FALSE	New data available: If this parameter is TRUE, the bit signals that data of a new communication request is available at the output.
ERROR	OUT	Bool		FALSE	Error occurred: If TRUE, this indicates that an error has occurred and that the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan / USS_Port_Scan_31 instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 6304)).
RUN_EN	OUT	Bool		FALSE	Operation enabled: This bit signals if the drive is running.
D_DIR	OUT	Bool		FALSE	Drive direction: This bit signals if the drive is running forward. • FALSE – forward • TRUE – backward
INHIBIT	OUT	Bool		FALSE	Drive blocked: This bit signals the status of the block bit for the drive. • FALSE – not blocked • TRUE – blocked
FAULT	OUT	Bool		FALSE	Drive error: This bit signals that an error occurred in the drive. You must remedy the fault and set the F_ACK bit to clear this bit.
SPEED	OUT	Real		0.0	Actual value drive speed (scaled value of STATUS 2 of the drive): This is the speed of the drive as a percentage of the configured speed.
STATUS1	OUT	Word		0	STATUS 1 of the drive This value includes fixed status bits of a drive.
STATUS3	OUT	Word		0	STATUS 3 of the drive This value includes a user-definable status word of the drive.
STATUS4	OUT	Word		0	STATUS 4 of the drive This value includes a user-definable status word of the drive.
STATUS5	OUT	Word		0	STATUS 5 of the drive This value includes a user-definable status word of the drive.
STATUS6	OUT	Word		0	STATUS 6 of the drive This value includes a user-definable status word of the drive.
STATUS7	OUT	Word		0	STATUS 7 of the drive This value includes a user-definable status word of the drive.
STATUS8	OUT	Word		0	STATUS 8 of the drive This value includes a user-definable status word of the drive.

When the initial execution of USS_Drive_Control / USS_Drive_Control_31 takes place, the drive specified by the USS address (DRIVE parameter) is initialized in the instance DB. After initialization, subsequent USS_Port_Scan / USS_Port_Scan_31 instructions can start communication with the drive from this drive number.

If you change the drive number, you must first place the CPU in STOP and then back in RUN to initialize the instance DB. The input parameters are configured in the USS send buffer and any outputs are read from a "previous" valid response buffer. USS_Drive_Control / USS_Drive_Control_31 only configures the data to be sent and evaluates data received in a previous request.

You can control the drive's direction of rotation by using the D_IR input (Bool) or the sign (positive or negative) at the SPEED_SP input (Real). The table below explains how these inputs work together to determine the direction of rotation of the drive, provided the motor rotates forward.

Table 4-313 Interaction of SPEED_SP and DIR parameters

SPEED_SP	DIR	Direction of rotation of the drive
Value > 0	0	Backward
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Backward

USS_Drive_Control / USS_Drive_Control_31 data block tags

The table below shows the public static tags in the instance DB of USS_Drive_Control / USS_Drive_Control_31 that you can use in your program.

Table 4-314 Static tags in the instance DB

Tag	Data type	Standard	Description
USS_DB. w_US-SExtendedError	Word	16#0	USS Drive Extended Error Code - drive-specific value The meaning of the error message depends on which instruction reported an error **first** (ERROR = TRUE). The following cases are distinguished: • USS_Write_Param / USS_Write_Param_31: You can find the meaning of the error code in the description of the drive. • USS_Read_Param / USS_Read_Param_31: You can find the meaning of the error code in the description of the drive. • USS_Port_Scan / USS_Port_Scan_31: Number of the drive affected by the error message.

Instruction versions

USS_Drive_Control:

Version 2.0 is functionally identical to version 1.2 and its version number was only incremented due to internal measures.

USS_Drive_Control_31:

Version 2.0 is functionally identical to version 1.0 and its version number was only incremented due to internal measures.

USS_Read_Param / USS_Read_Param_31: Read data from drive (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Read_Param_31 instruction

It is only possible to use the USS_Read_Param_31 instruction on an S7-1500 CPU.

Description

The USS_Read_Param instruction reads a parameter from one of max. 16 drives.

The USS_Read_Param_31 instruction reads a parameter from one of max. 31 drives.

All USS functions assigned to one USS network and one PtP communication port must use the instance data block of the USS_Drive_Control / USS_Drive_Control_31 instruction. USS_Read_Param / USS_Read_Param_31 must be called from a cycle OB of the main program.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		–	A positive edge at REQ creates a new read request.
DRIVE	IN	USInt	Byte	–	Address of the drive: DRIVE is the address of the USS drive. The valid range is between drive 1 and drive 16.
PARAM	IN	UInt		–	Parameter number: PARAM specifies the drive parameter to write. The range for this parameter is between 0 and 2047. With some drives, the most significant byte of the INDEX parameter can be used to access parameter values greater than 2047. Additional information on access to an extended range is available in your drive manual.
INDEX	IN	UInt		–	Parameter index: INDEX specifies the drive parameter index to which to write. It is a 16-bit value in which the least significant bit is the actual index value with a range of (0 to 255). The drive can also use the most significant byte, which is drive-specific. Additional information is available in your drive manual.

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control / USS_Drive_Control_31 instruction to your program.
DONE[1]	OUT	Bool		FALSE	If this parameter is TRUE, the previously requested value of the read parameter is available at the VALUE output. This bit is set when the USS_Drive_Control / USS_Drive_Control_31 instruction recognizes the read response of the drive. This bit is reset the next time USS_Read_Param / USS_Read_Param_31 is called.
ERROR	OUT	Bool		FALSE	ERROR = TRUE: An error has occurred and the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan / USS_Port_Scan_31 instruction. You may need to check the value of the static tag USS_DB. w_USSExtendedError in the instance DB of the USS_Drive_Control / USS_Drive_Control_31 instruction.
STATUS	OUT	UInt		0	Error code (see Error messages (Page 6304)).
VALUE	OUT	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, UInt, DWord, DInt, UDInt, Real)	–	This is the value of the parameter that was read and is only valid if the DONE bit is true.

[1] The DONE bit indicates that valid data was read out of the referenced motor drive and transmitted to the CPU. It does not indicate that the instruction is cabable of immediately read an additional parameter. An empty read request must be sent to the motor drive and acknowledged by the instruction before the parameter channel is freed up for use by the respective drive. The immediate call of USS_Read_Param / USS_Read_Param_31 or USS_Write_Param / USS_Write_Param_31 for the specific motor drive results in error 16#818A.

Instruction versions

USS_Read_Param:

Version 1.5 is functionally identical to version 1.4 and its version number was only incremented due to internal measures.

USS_Read_Param_31:

Version 1.1 is functionally identical to version 1.0 and its version number was only incremented due to internal measures.

USS_Write_Param / USS_Write_Param_31: Change data in drive (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Note

Using the USS_Write_Param_31 instruction

It is only possible to use the USS_Write_Param_31 instruction on an S7-1500 CPU.

Note

For EEPROM write instructions (EEPROM in a USS drive):

Keep the number of EEPROM write operations to a minimum in order to maximize the EEPROM service life.

Description

The USS_Write_Param instruction changes a parameter in one of max. 16 drives.

The USS_Write_Param_31 instruction changes a parameter in one of max. 31 drives.

All USS functions assigned to one USS network and one PtP communication port must use the instance data block of the USS_Drive_Control / USS_Drive_Control_31.

USS_Write_Param / USS_Write_Param_31 must be called from the cycle OB of a main program.

Parameters

Table 4-315 Data types for the parameters

Parame-ters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/ WinAC		
REQ	IN	Bool		–	A positive edge at REQ creates a new write request.
DRIVE	IN	USInt	Byte	–	Address of the drive: DRIVE is the address of the USS drive. The valid range is between drive 1 and drive 16.
PARAM	IN	UInt		–	Parameter number: PARAM specifies the drive parameter to write. The range for this parameter is between 0 and 2047. With some drives, the most significant byte of the INDEX parameter can be used to access parameter values greater than 2047. Additional information on access to an extended range is available in your drive manual.

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/ WinAC		
INDEX	IN	UInt		–	Parameter index: INDEX specifies the drive parameter index to which to write. It is a 16-bit value in which the least significant bit is the actual index value with a range of (0 to 255). The drive can also use the most significant byte, which is drive-specific. Additional information is available in your drive manual.
EEPROM	IN	Bool		–	Save in EEPROM of the drive: If TRUE, the transaction of a parameter for writing to the drive is saved in the EEPROM of the drive. If FALSE, the written value is saved only temporarily and is lost the next time you switch on the drive.
VALUE	IN	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, UInt, DWord, DInt, UDInt, Real)	–	Value of the parameter in which you want to write. It must be valid with a positive edge of REQ.
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control / USS_Drive_Control_31 instruction to your program.
DONE[1]	OUT	Bool		FALSE	If TRUE, the VALUE input is written to the drive. This bit is set when the USS_Drive_Control / USS_Drive_Control_31 instruction recognizes the write response of the drive. This bit is reset the next time USS_Write_Param / USS_Write_Param_31 is called.
ERROR	OUT	Bool		FALSE	If TRUE, an error has occurred and the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan / USS_Port_Scan_31 instruction. You may need to check the value of the static tag USS_DB.w_USSExtendedError in the instance DB of the USS_Drive_Control / USS_Drive_Control_31 instruction.
STATUS	OUT	UInt		0	Error code (see Error messages (Page 6304)).

[1] The DONE bit indicates that valid data was read out of the referenced motor drive and transmitted to the CPU. It does not indicate that the USS library is able to read out an additional parameter immediately. An empty write request must be sent to the motor drive and acknowledged by the instruction before the parameter channel is freed up for use by the respective drive. The immediate call of USS_Read_Param / USS_Read_Param_31 or USS_Write_Param / USS_Write_Param_31 FC for the specific motor drive results in error 0x818A.

Instruction versions

USS_Write_Param:

Version 1.6 is functionally identical to version 1.5 and its version number was only incremented due to internal measures.

USS_Write_Param_31:

Version 1.1 is functionally identical to version 1.0 and its version number was only incremented due to internal measures.

General information on drive setup (S7-300, S7-400)

Requirements for the drive setup

- You need to configure the use of 4 PIV words (ParameterIDValue) for the drives.
- The drives can be configured for 2, 4, 6 or 8 PZD words (Process data area).
- The number of PZD words in the drive must correspond to the PZD_LEN input of the USS_Drive_Control instruction of the drive.
- Make sure that the data transmission rate of all drives corresponds to the BAUD input of the USS_Port_Scan instruction.
- Make sure that the drive is set up for USS communication.
- Make sure that it is specified in the drive that the frequency setpoint is provided by the USS interface.
- Make sure that the drive address is specified (areas: 1-16).
  This address must correspond to the DRIVE input at the USS_Drive_Control block of the drive.
- Make sure that the RS485 network is terminated correctly.

Connection and setup of SINAMICS V20 drive

An application example for operation of a SINAMICS V20 at an S7-1200 is available on the Internet. (http://support.automation.siemens.com/WW/view/en/63696870)

Connecting the SINAMICS V20 drive

Connection example of a SIEMENS G120(C) drive to a USS network. Connection examples for other drives are available in the manual of the respective drive.

The connection of a SINAMICS G120(C) drive to the USS network takes place via plug-in connection. The connection is short-circuit proof and isolated.

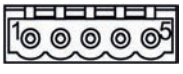

1 0 V reference potential
2 RS485N, receiving and sending (-)
3 RS485N, receiving and sending (+)
4 Cable shield
5 Not used

Figure 4-33 USS connection

NOTICE
Different reference voltages If you connect devices that do not have the same reference voltage, you may create unwanted currents in the connection cable. These unwanted currents may cause communication errors or damages in the devices. Make sure that all devices you connect with a communication cable either have the same reference conductor in the circuit or are electrically disconnected to prevent the occurrence of unwanted currents. Make sure that the shield is connected to the ground or pin 1 of the bus connector on the drive. Make sure that wiring terminal 2 (GND) of the G120(C) is connected to the ground.

If the RS485 master (e.g., S7-1200 CPU with CM1241 communication module) is connected by means of a PROFIBUS connector, wire the bus cables as follows:

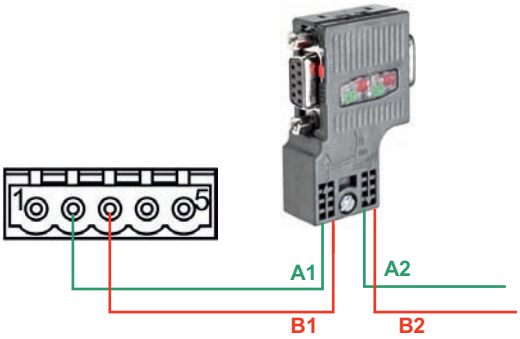

Figure 4-34 Connection of communication module

If the RS485 master is a terminating station in the network or a point-to-point connection, you must use terminals A1 and B1 (not A2 and B2) of the PROFIBUS connector, as these terminals provide termination settings (for example, with the DP plug connector 6ES7972-0BB52-0XA0).

If the G120(C) is configured as terminating station in the network, you must set the switch for the bus terminating resistor to "ON".

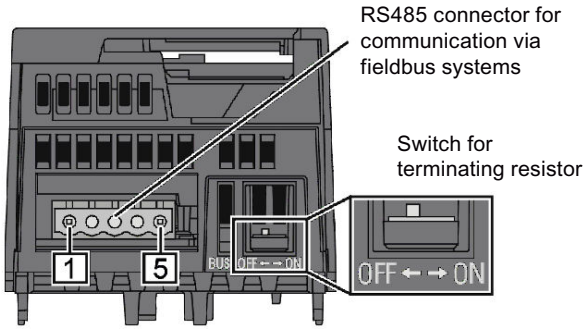

Figure 4-35 Connection of terminating stations

Setup of a G120(C) drive

Before you connect a drive to the S7-1500 or ET 200SP, make sure that the drive has the following system parameters.

Step	Instruction	Operating Instructions	
		G120 [1]	G120C [2]
1	Perform a basic commissioning of the drive with the Operator Panel BOP-2. The converter offers different defaults (macros) for its inputs and outputs and the fieldbus interface. In the ninth step of the basic commissioning (MAC PAR p15), select macro 21 for USS communication. This determines the default for the following parameters: • Data transmission rate (p2020): 38400 bps • Number of PZD (p2022): 2 • Number of PIV (p2023): variable Note: You can also conduct basic commissioning with the STARTER commissioning software or with SINAMICS Startdrive.	Chapter 4.4.3	Chapter 6.4.1
2	Specify the USS address of the converter with the address switch on the control unit of the G120 or on the G120(C). • Valid address range: 1 … 30 Note: Yon can also set the USS address with the parameter p2021 or with STARTER or SINAMICS Startdrive.	Chapter 6.2.2.1	Chapter 8.4.2.1
	With the following steps you are accessing the parameters directly with the BOP-2 by entering their numbers and modifying their values.	Chapter 4.4.2	Chapter 6.4.2
3	Adapt the following communication-related converter parameters of your application: • Data transmission rate (p2020), of ≠ 38400 bps (Make sure that setting is identical with the BAUD parameter of the USS_Port_Scan communication instruction.) • Number of PZD (p2022), if ≠ 2 (Make sure that the setting is identical with the PZD_LEN parameter of the USS_Drive_Control communication instruction.) • Number of PIV (p2023) = 4 (Change the value set to "variable" (127) with macro 21 by default to "4" (required by instructions USS_Read_Param and USS_Write_Param.) • Fieldbus SS monitoring time [ms] (p2040)	Chapter 6.2.2.2	Chapter 8.4.2.1
4	Specify the source for the speed setpoint. • n_set Eval (p1000[0]) = 6 (The speed setpoint is provided by the USS bus.)		

Step	Instruction	Operating Instructions	
		G120 [1]	G120C [2]
5	Set the reference value for speed and frequency. • n_reference f_reference (p2000) = (6,00 min^{-1} to 210000,00 min^{-1}) (all relative speeds or frequencies refer to this reference value. The reference value corresponds to 100% or 4000_{hex} (word) or $4000\ 0000_{hex}$ (double word). The following applies: $f_{reference_value}$ (in Hz) = $n_{reference_value}$ (in ((min^{-1}) / 60) x number of pole pairs))		
6	Transfer the parameters to the non-volatile memory. • Save par (p0971) = 1		

1) G120 (http://support.automation.siemens.com/WW/view/en/62089662)

2) G120(C) (http://support.automation.siemens.com/WW/view/en/61462568)

Error messages (S7-300, S7-400)

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)

Error code	Description	Remedy
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#7002	Interim call: Data transmission running	-
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
16#8193	The module does not support supply voltage diagnostics L+.	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)

Error code	Description	Remedy
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire operation (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire operation. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		

Error code	Description	Remedy
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see AUTOHOTSPOT).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (bytes) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)

Error code	Description	Remedy
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer on startup is deactivated (0) • Clear receive buffer on startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end delimiters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
16#81DF	The module has reset the interface to the FB for one of the following reasons: • Module was restarted • Module parameters were reassigned • CPU STOP	—
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.

Error code	Description	Remedy
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Break in receive line to partner. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (bytes) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
16#81F2	The DSR signal cannot be set because the module has the type DTE.	Check the configured type of the module. The module type must be DCE (data communication equipment).
16#81F3	The DTR signal cannot be set because the module has the type DCE.	Check the configured type of the module. The module type must be DTE (data terminal equipment).
16#81F4	Block header error (e.g. incorrect block type or incorrect block length)	Check the instance DB and the block header.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING

Error code	Description	Remedy
16#8225	Receive_Conditions points to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 KB • Non-optimized memory area: 4 KB Note: If the pointer points to an optimized memory area, do not send more than 1 KB.
16#8229 [1]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in the RDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter before the 1st call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 [1]	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 [1]	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 [1]	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 [1]	Receive_Conditions does not point to a structure of the typeReceive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		

Error code	Description	Remedy
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the frame header and data string or the connection is terminated with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.

Error code	Description	Remedy
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.

Error code	Description	Remedy
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.

Error code	Description	Remedy
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 6323) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 6323) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 6323) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).

Error code	Description	Remedy
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#80C9	The slave does not respond within the time set by Blocked_Proc_Timeout.	Check the setting for Blocked_Proc_Timeout. Check if the module has been configured with the Modbus_Comm_Load instruction. The module may possibly need to be reconfigured using Modbus_Comm_Load after a pull/plug or after voltage recovery.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.

Error code	Description	Remedy
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200% to +200%
16#8187	Incorrect drive number returned	Check the response frame of the drive.

Error code	Description	Remedy
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request. Check if the instructions USS_Read_Param, USS_Write_Param or USS_Port_Scan have reported an error. If they have, check the value of the static tag USS_DB. w_USSExtendedError of the USS_Drive_Control instruction.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

MODBUS (RTU) (S7-300, S7-400)

Overview of the Modbus RTU communication (S7-300, S7-400)

Modbus RTU communication

Modbus RTU (Remote Terminal Unit) is a standard protocol for communication in the network and uses the RS232 or RS422/485 connection for serial data transmission between Modbus devices in the network.

Modbus RTU uses a master/slave network in which all communication is triggered by a single master device while the slaves can only respond to the request of the master. The master sends a request to a slave address and only the slave with this slave address responds to the command.

Exception: Modbus slave address 0 sends a broadcast frame to all slaves (without slave response).

Modbus function codes

- A CPU that is operated as a Modbus RTU master can read and write data and I/O states in a Modbus RTU slave connected by means of a communication connection.
- A CPU operated as a Modbus RTU slave allows a Modbus RTU master connected over a communication connection to read and write data and I/O states in its own CPU.

Table 4-316 Functions for reading data: Reading distributed I/O and program data

Modbus function code	Functions for reading data from the slave (server) - standard addressing
01	Read output bits: 1 to 2000/1992[1)] bits per request
02	Read input bits: 1 to 2000/1992[1)] bits per request
03	Read hold register: 1 to 125/124[1)] words per request
04	Read input words: 1 to 125/124[1)] words per request

1) for extended addressing

Table 4-317 Functions for writing data: Changing distributed I/O and program data

Modbus function code	Functions for writing of data in the slave (server) - standard addressing
05	Write one output bit: 1 bit per request
06	Write one hold register: 1 word per request
15	Write one or several output bits: 1 to 1960 bits per request
16	Write one or several hold registers: 1 to 122 words per request

- The Modbus function codes 08 and 11 offer diagnostic options for communication with the slave device.
- Modbus slave address 0 sends a broadcast frame to all slaves (without slave response; for function codes 5, 6, 15, 16).

Table 4-318 Station addresses in the Modbus network

Station		Address
RTU station	Standard station address	1 to 247 and 0 for broadcast
	Extended station address	1 to 65535 and 0 for broadcast

Modbus memory addresses

The number of Modbus memory addresses (input/output addresses) that is actually available depends on the CPU version and the available work memory.

Modbus RTU instructions in your program

- Modbus_Comm_Load: You need to run Modbus_Comm_Load to set up PtP parameters such as data transmission rate, parity and data flow control. Once you have configured the communication module for the Modbus RTU protocol, it can only be used by the Modbus_Master instruction or the Modbus_Slave instruction.
- Modbus_Master: The CPU can be used as Modbus RTU master device with the Modbus master instruction for communication with one or more Modbus slave devices.
- Modbus_Slave: The CPU can be used as Modbus RTU slave device with the Modbus slave instruction for communication with one Modbus master device.

Modbus_Comm_Load: Configure communication module for Modbus (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Modbus_Comm_Load instruction configures a communication module for communication by means of the Modbus RTU protocol. An instance data block is automatically assigned when you add the Modbus_Comm_Load instruction in your program.

Configuration changes of Modbus_Comm_Load are saved on the CM and not in the CPU. With voltage recovery and pulling/plugging, the CM is configured with the data saved in the device configuration. The Modbus_Comm_Load instruction must be called in these scenarios.

Parameters

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	Port	Laddr	0	Specifies the communication module which is used for the communication: • For S7-1500/S7-1200: "HW identifier" from the device configuration. The symbolic port name is assigned in the "System constants" tab of the PLC tag table and can be applied from there. • For S7-300/S7-400: "Input address" from the device configuration. In the S7-300/400/WinAC systems the PORT parameter is assigned the input address assigned in HWCN.

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
BAUD	IN	UDInt	DWord	9600	Selection of the data transmission rate Valid values are: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 bit/s.
PARITY	IN	UInt	Word	0	Selection of parity: • 0 – None • 1 – Odd • 2 – Even
FLOW_CTRL	IN	UInt	Word	0	Selection of flow control: • 0 – (default) no flow control • 1 – Hardware flow control with RTS always ON (not with RS422/485 CMs) • 2 – Hardware flow control with RTS switched (not with RS422/485 CMs)
RTS_ON_DLY	IN	UInt	Word	0	Selection RTS ON delay: • 0 – No delay from "RTS active" until the first character of the frame is sent. • 1 to 65535 – Delay in milliseconds from "RTS active" until the first character of the frame is sent (not with RS422/485 CMs). RTS delays must be used independent of the selection FLOW_CTRL.
RTS_OFF_DLY	IN	UInt	Word	0	Selection RTS OFF delay: • 0 – No delay after transmission of last character until "RTS inactive" • 1 to 65535 – Delay in milliseconds after transmission of last character until "RTS inactive" (not with RS422/485 ports). RTS delays must be used independent of the selection FLOW_CTRL.
RESP_TO	IN	UInt	Word	1000	Response timeout: 5 ms to 65535 ms - Time in milliseconds that Modbus_Master waits for a response from the slave. If the slave does not respond within this period, Modbus_Master repeats the request or terminates the request with an error if the specified number of repetitions (see below, RETRIES parameter) has been sent.
MB_DB	IN/OUT	MB_BASE		-	A reference to the instance data block of the Modbus_Master or Modbus_Slave instructions. The MB_DB parameter must be connected with the (static and therefore not visible in the instruction) MB_DB parameter of the Modbus_Master or Modbus_Slave instruction.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Comm_Load instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.

Parameter	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
DONE	OUT	Bool		FALSE	The DONE bit is TRUE for one cycle after the last request has been completed without errors.
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. The error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 6341))

Modbus_Comm_Load is executed to configure a port for the Modbus RTU protocol. Once you have configured the port for the Modbus RTU protocol, it can only be used by the Modbus_Master or Modbus_Slave instructions.

You have to run Modbus_Comm_Load for the configuration of each communication port that is to be used for Modbus communication. You must assign a unique Modbus_Comm_Load instance DB to each port that you use. Only run Modbus_Comm_Load again if you need to change communication parameters, such as data transmission rate or parity, or in case the network has returned.

For example, an instance data block is assigned to the instruction if you add Modbus_Master or Modbus_Slave to your program. You need to connect the MB_DB parameter of the Modbus_Comm_Load instruction to the MB_DB parameter of the Modbus_Master or Modbus_Slave instruction.

Modbus_Comm_Load data block tags

The table below shows the public static tags in the instance DB of Modbus_Comm_Load that you can use in your program.

Table 4-319 Static tags in the instance DB

Tag	Data type		Standard	Description
	S7-1200/1500	S7-300/400/WinAC		
ICHAR_GAP	Word		0	Maximum character delay time between characters. This parameter is specified in milliseconds and increases the anticipated period between the received characters. The corresponding number of bit times for this parameter is added to the Modbus default value of 35 bit times (3.5 character times).
RETRIES	Word		2	Number of retries that the master executes before the error code 0x80C8 for "No response" is returned.
EN_SUPPLY_VOLT	Bool		0	Enable diagnostics for missing supply voltage L+

Tag	Data type		Stand-ard	Description
	S7-1200/1500	S7-300/400/WinAC		
MODE	USInt	Byte	0	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master, CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave, CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode [1]
LINE_PRE	USInt	Byte	0	Receive line initial state Valid initial states are: • 0 = "No" initial state [1] • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire mode (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
BRK_DET	USInt	Byte	0	Break detection The following are valid: • 0 = break detection deactivated • 1 = break detection activated
EN_DI-AG_ALARM	Bool		0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
STOP_BITS	USINT	Byte	1	Number of stop bits; • 1 = 1 stop bit, • 2 = 2 stop bits, • 0, 3 to 255 = reserved

[1] Required setting for the use of PROFIBUS cables with CM 1241 for RS485

Instruction versions

Version 3.1 is functionally identical to version 3.0 and its version number was only incremented due to internal measures.

Modbus_Master: Communicate as Modbus master (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

The Modbus_Master instruction communicates as Modbus master via a port configured by the Modbus_Comm_Load instruction. An instance data block is automatically assigned when you add the Modbus_Master instruction in your program. The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the (static) MB_DB parameter of the Modbus_Master instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	FALSE = no request TRUE = request to send data to the Modbus slave
MB_ADDR	IN	UInt	Word	-	Modbus RTU station address: Standard addressing range (1 to 247 as well as 0 for Broadcast) Extended addressing range (1 to 65535 as well as 0 for Broadcast) The value 0 is reserved for the broadcast of a frame to all Modbus slaves. Only the Modbus function codes 05, 06, 15 and 16 are supported for the broadcast.
MODE	IN	USInt	Byte	0	Mode selection: Specifies the type of request (read, write or diagnostics). Additional information is available in the table of Modbus functions below.
DATA_ADDR	IN	UDInt	DWord	0	Start address in the slave: Specifies the start address of the data that is accessed in the Modbus slave. The valid addresses are listed in the table of Modbus functions below.
DATA_LEN	IN	UInt	Word	0	Data length: Specifies the number of bits or words this instruction is to access. The valid lengths are listed in the table of Modbus functions below.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Master instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
DATA_PTR	IN/OUT	Variant	Any	-	Data pointer: Points to the flag or DB address for the data to be written or read. As of instruction version V3.0: The parameter may point to an optimized memory area. In the optimized memory area, a single element or an array is permitted with the following data types: Bool, Byte, Char, Word, Int, DWord, DInt, Real, USInt, UInt, UDInt, SInt, WChar. Every other data type results in error message 16#818C.
DONE	OUT	Bool		FALSE	The DONE bit is TRUE for one cycle after the last request has been completed without errors.
BUSY	OUT	Bool		-	• FALSE – no command active for Modbus_Master • TRUE – command for Modbus_Master in progress
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. The error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		0	Error code (see Error messages (Page 6341))

Tags in the data block of the Modbus master

The table below shows the public static tags in the instance DB of Modbus_Master that you can use in your program.

Table 4-320 Static tags in the instance DB

Tag	Data type	Standard	Description
Blocked_Proc_Timeout	Real	3.0	Duration (in seconds) for which to wait for a blocked Modbus master instance before this instance is removed as ACTIVE. This may happen, for example, if a master request was output and the program then stops to call the master function before it has completely finished the request. The time value must be greater than 0 and less than 55 seconds to avoid an error to occur. See also "Rules for communication by the Modbus-Master" and "Calling theModbus_Master instruction with different parameter settings".
Extended_Addressing	Bool	FALSE	Configures the slave station address as single or double byte. • FALSE = One-byte address; 0 to 247 • TRUE = Two-byte address (corresponds to extended addressing); 0 to 65535

Tag	Data type	Standard	Description
Compatibility_Mode [1]	Bool	FALSE	Compatibility mode with CP 341 and CP 441-2 and ET 200SP 1SI with driver for Modbus RTU and with ET 200SP 1SI for Modbus. The default value is 0. • FALSE = as per Modbus specification, not compatible • TRUE = compatible – For FC1 and FC2: The data read from the received frame is written word for word to the addressed CPU memory and exchanged byte by byte. If the number of bits to be transmitted is not a multiple of 16, the bits which are not relevant are set to null in the last word. – For FC15: The words to be transmitted are read word by word from the addressed memory and written byte by byte to the send frame. If the number of bits to be transmitted is not a multiple of 8, the bits in the last byte which are not relevant are read unchanged from the addressed memory and entered in the send frame.
MB_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to this MB_DB parameter of the Modbus_Master instruction.

[1] The PtP communication modules respond as defined in the Modbus specification. To retain a response as with CP 341, CP 441-2 and ET 200SP 1SI for Modbus, use the "Compatibility_Mode" parameter.

You program can write values to the Blocked_Proc_Timeout and Extended_Addressing tags to control the Modbus master operations.

Rules for communication by the Modbus-Master

- Modbus_Comm_Load must be run to configure a port so that the Modbus_Master instruction can communicate with this port.
- A port which is to be used as Modbus master must not be used by Modbus_Slave . You can use one or several instances of Modbus_Master [1] with this port. But all versions of the Modbus_Master must use the same instance DB for the port.
- The Modbus instructions do not use communication alarm events to control the communication process. Your program must query the Modbus_Master instruction for completed commands (DONE, ERROR).
- We recommend to call all executions of Modbus_Master for a specific port from a program cycle OB. Modbus master instructions can only be executed in one program cycle or in one cyclical/time-controlled processing level. They may not be processed in different processing levels. The priority interruption of a Modbus master instruction by another Modbus master instruction in a processing level with higher priority results in improper operation. Modbus master instructions may mot be processed in startup, diagnostic or time error levels.

[1] "Instance of Modbus master" here means a call of the Modbus_Master instruction with the same interconnection to a Modbus_Comm_Load instruction and the same setting for the MB_ADDR, MODE, DATA_ADDR and DATA_LEN parameters.

Example

Modbus_Master is called with MODE=0 and DATA_ADDR=10

This job is now active until it is completed with DONE=1 or ERROR=1 or until the time monitoring configured at the Blocked_Proc_Timeout parameter has expired. If a new command is started after the watchdog time expires and before the previous command has been completed, the previous command is aborted without an error message.

If, while this command is running, the instruction is now called a second time with the same instance data but different MODE and DATA_ADDR parameter settings, this second call is terminated with ERROR=1 and STATUS=8200.

Calling the Modbus_Master instruction with different parameter settings

If multiple calls of the Modbus_Master instruction with different settings for MB_ADDR, MODE, DATA_ADDR or DATA_LEN are placed in your program, you must ensure that only one of these calls is active at any given time. Otherwise, the error message 16#8200 is output (interface is busy with an ongoing request).

If a call cannot be processed in full, the watchdog is activated by the Blocked_Proc_Timeout parameter and terminates the ongoing command.

REQ parameter

FALSE = no request; TRUE = request to send data to the Modbus slave

Enable the requested transmission. This transmits the contents of the buffer to the point-to-point communication interface.

You use the DATA_ADDR and MODE parameters to select the Modbus function code.

DATA_ADDR (Modbus start address in the slave): Specifies the start address of the data that is accessed in the Modbus slave.

The Modbus_Master instruction uses the MODE input instead of a function code input. The combination of MODE and DATA_ADDR specifies the function code that is used in the actual Modbus frame. The table below shows how the MODE parameter, the Modbus function code and the Modbus address range in DATA_ADDR are related.

Table 4-321 Modbus functions

MODE	DATA_ADDR (Modbus address)			DATA_LEN (data length)			Modbus function code	Operation and data		
0				Bits per request			01	Read output bits:		
	1	to	9999	1	to	2000/1992 [1]		0	to	9998
0				Bits per request			02	Read input bits:		
	10001	to	19999	1	to	2000/1992 [1]		0	to	9998
0				Words per request			03	Read hold register:		
	40001	to	49999	1	to	125/124 [1]		0	to	9998
	400001	to	465535	1	to	125/124 [1]		0	to	65534
0				Words per request			04	Read input words:		
	30001	to	39999	1	to	125/124 [1]		0	to	9998

MODE	DATA_ADDR (Modbus address)			DATA_LEN (data length)			Modbus function code	Operation and data		
1				Bits per request			05	Write one output bit:		
	1	to	9999	1				0	to	9998
1				1 word per request			06	Write one hold register:		
	40001	to	49999	1				0	to	9998
	400001	to	465535	1				0	to	65524
1				Bits per request			15	Write multiple output bits:		
	1	to	9999	2	to	1968/1960 [1]		0	to	9998
1				Words per request			16	Write multiple hold registers:		
	40001	to	49999	2	to	123/122		0	to	9998
	400001	to	465534	2	to	123/122 [1]		0	to	65534
2 [2]				Bits per request			15	Write one or several output bits:		
	1	to	9999	1	to	1968/1960 [1]		0	to	9998
2 [2]				Words per request			16	Write one or several hold registers:		
	40001	to	49999	1	to	123		0	to	9998
	400001	to	465535	1	to	122 [1]		0	to	65534
11	Both DATA_ADDR and DATA_LEN operands of the Modbus_Master are ignored with this function.						11	Read status word and event counter of the slave communication. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful processing of a frame.		
80				1 word per request			08	Check slave status with data diagnostic code 0x0000 (loopback test – slave returns an echo of the request)		
	-			1				-		
81				1 word per request			08	Reset slave event counter using data diagnostic code 0x000A		
	-			1				-		
104 [3]				Words per request			04	Read input words		
	0	to	65535	1	to	125/124 [1]		0	to	65535
3 to 10, 12 to 79, 82 to 103, 105 to 255	-			-				Reserved		

[1] In extended addressing, see the Extended_Adressing parameter, the maximum data length is shorter by 1 byte or 1 word depending on the data type of the function.

[2] MODE 2 allows you to write one or more output bits and one or more holding registers using the Modbus functions 15 and 16. MODE 1 uses the Modbus functions 5 and 6 to write 1 output bit and 1 holding register, and Modbus functions 15 and 16 to write multiple output bits and multiple holding registers.

[3] The following applies to S7-300/400/WinAC: Reserved

DATA_PTR parameter

The DATA_PTR parameter points to the DB or bit memory address in which reading or writing is performed. If you use a data block, you must create a global data block that provides the data memory for read and write processes on Modbus slaves.

Note

S7-1200/1500 - The data block addressed using DATA_PTR must support direct addressing

The data block must permit direct (absolute) and symbolic addressing.

Note

Using function code 5

Function code 5 is used to set or delete individual bits.

When a bit is set, the value "16#FF00" must be specified in the first word of the addressed DB or bit memory area via DATA_PTR.

- With S7-1200, the value "16#0100" can also be specified to set a bit.
- To reset a bit, the value "16#0000" must be specified in the first word of the DB or bit memory area addressed via DATA_PTR.

All other values are rejected with ERROR = TRUE and STATUS = 16#8384.

Data block structures for the DATA_PTR parameter

- These data types are valid for reading words of the Modbus address range (DATA_PTR) 30001 to 39999, 40001 to 49999 and 400001 to 465535 as well as for writing words to the Modbus address range (DATA_PTR parameter) 40001 to 49999 and 400001 to 465535.
  - Standard array of data types WORD, UINT or INT
  - Named structure of the WORD, UINT or INT type in which each element has a unique name and a 16-bit data type.
  - Named complex structure in which each element has a unique name and a 16-bit or 32-bit data type.
- For reading and writing bits for the Modbus address range (DATA_PTR parameter) 00001 to 09999 and for reading bits from 10001 to 19999.
  - Standard field from Boolean data types.
  - Named Boolean structure from clearly named Boolean tags.
- It is not essential, but nevertheless advisable, to allocate each Modbus_Master instruction its own separate memory area. The reason for this is that data destruction is far more likely if multiple Modbus_Master instructions are reading and writing in the same memory area.
- It is not necessary for the data areas for DATA_PTR to be located in the same global data block. You can create a data block with several areas for Modbus read processes, a data block for Modbus write processes or a data block for each slave station.

Instruction versions

Version 3.0 is functionally identical to version 2.4 and its version number was only incremented due to internal measures.

See also

Modbus_Slave: Communicate as Modbus slave (Page 6329)

Modbus_Slave: Communicate as Modbus slave (S7-300, S7-400)

Note

Use with CM1241

The use of this instruction with a CM1241 is only possible from firmware version V2.1 of the module.

Description

Your program can use the Modbus_Slave instruction to communicate as a Modbus slave by using a CM (RS422/485 or RS232). STEP 7 automatically creates an instance DB when you add the instruction. The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the (static) MB_DB parameter of the Modbus_Slave instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
MB_ADDR	IN	UInt	Word	-	Standard address of the Modbus slave: Standard addressing range (1 to 247) Extended addressing range (0 to 65535) Note: 0 is the broadcast address
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Slave instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE. Note: The parameter is only available for S7-300/400 instructions.

Parameters	Declaration	Data type		Standard	Description
		S7-1200/1500	S7-300/400/WinAC		
MB_HOLD_REG	IN/OUT	Variant	Any	-	Pointer to the Modbus hold register DB: The Modbus hold register may be the memory area of the flags or a data block. As of instruction version V4.0: The parameter must point to a memory area that has a length of at least 16 bits. A shorter length results in error message 16#8187. This applies to single elements, arrays, STRUCTs and UDTs. For example, a Single Bool or an array consisting of less than 16 Boolean elements results in the error message. If the length is not a multiple of 16 bits, the remaining bits at the end of the memory area cannot be read or written by the Modbus_Slave instruction. The parameter may point to an optimized memory area. In the optimized memory area, a single element or an array is permitted with the following data types: Bool, Byte, Char, Word, Int, DWord, DInt, Real, USInt, UInt, UDInt, SInt, WChar. Every other data type results in error message 16#818C.
NDR	OUT	Bool		FALSE	New data available: • FALSE – No new data • TRUE – Indicates that new data was written by the Modbus master The NDR bit is TRUE for one cycle after the last request has been completed without errors.
DR	OUT	Bool		FALSE	Read data: • FALSE – No data read • TRUE – Indicates that the instruction has stored the data received by the Modbus master in the target area. The DR bit is TRUE for one cycle after the last request has been completed without errors.
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. If the execution was terminated with an error, the error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		0	Error code (see Error messages (Page 6341))

The function codes of the Modbus communication (1, 2, 4, 5 and 15) can read and write bits and words directly in the process image input and in the process image output of the CPU. The MB_HOLD_REG parameter must be defined as data type greater than one byte for these

function codes. The table below shows the sample assignment of Modbus addresses to the process image in the CPU.

Table 4-322 Assignment of Modbus addresses to the process image

Modbus functions						S7-1200			
Code	Function	Data area	Address area			Data area	CPU address		
01	Read bits	Output	0	to	8191	Process image output	O0.0	to	O1023.7
02	Read bits	Input	0	to	8191	Process image input	I0.0	to	I1023.7
04	Read words	Input	0	to	511	Process image input	IW0	to	IW1022
05	Write bit	Output	0	to	8191	Process image output	O0.0	to	O1023.7
15	Write bits	Output	0	to	8191	Process image output	O0.0	to	O1023.7

Table 4-323 Assignment of Modbus addresses to the process image

Modbus functions						S7-1500 / S7-300 / S7-400			
Function code	Function	Data area	Address area			Data area	CPU address		
01	Read bits	Output	0	to	9998	Process image output	O0.0	to	A1249.6
02	Read bits	Input	0	to	9998	Process image input	I0.0	to	I1249.6
04	Read words	Input	0	to	9998	Process image input	IW0	to	IW19996
05	Write bit	Output	0	to	9998	Process image output	O0.0	to	A1249.6
15	Write bits	Output	0	to	9998	Process image output	O0.0	to	A1249.6

Note

The available address area may be smaller, depending on the memory configuration of the CPU.

The function codes of the Modbus communication (3, 6, 16) use a Modbus hold register which is an address area in the memory area of the flags or a data block. The type of holding register is specified by the MB_HOLD_REG parameter of the Modbus_Slave instruction.

Note

S7-1200/1500 - type of the MB_HOLD_REG data block

The data block with Modbus hold register must permit direct (absolute) and symbolic addressing.

Table 4-324 Diagnostics functions

Modbus diagnostic functions of the S7-1200 Modbus_Slave		
Function codes	**Subfunction**	**Description**
08	0000H	Output request data of echo test: The Modbus_Slave instruction returns the echo of a received data word to the Modbus master.
08	000AH	Clear communication event counter: The Modbus_Slave instruction clears the communication event counter used for Modbus function 11.
11		Call communication event counter: The Modbus_Slave instruction uses an internal communication event counter to detect the number of successful Modbus read and Modbus write requests that are sent to the Modbus slave. The counter is not incremented for function 8, function 11 and broadcast requests. It is also not incremented for requests that result in communication errors (for example, parity or CRC errors).

The Modbus_Slave instruction supports broadcast write requests from Modbus masters as long as the requests include access to valid addresses. Modbus_Slave generates error code 16#8188 for function codes that are not supported by the broadcast function.

Variables of the Modbus slave in instruction version V3.0

This table below shows the public static tags in the instance data block of Modbus_Slave that you can use in your program.

Table 4-325 Variables of the Modbus slave

Tag	Data type	Standard	Description
HR_Start_Offset	Word	0	Specifies the start address of the Modbus hold register (default = 0)
QB_Start	Word	0	Start address of the valid writable addressing range of the outputs (byte 0 to 65535) Note: The variable is not available for S7-300, S7-400 and WinAC.
QB_Count	Word	0xFFFF	Number of output bytes that can be written by the Modbus master. Note: The variable is not available for S7-300, S7-400 and WinAC.
Extended_Addressing	Bool	FALSE	Extended addressing, configures slave addressing as single or double byte (FALSE = single byte address, TRUE = double byte address)
Request_Count	Word	0	The number of all requests received by this slave
Slave_Message_Count	Word	0	The number of requests received for this specific slave
Bad_CRC_Count	Word	0	The number of received requests that have a CRC error
Broadcast_Count	Word	0	The number of received broadcast requests
Exception_Count	Word	0	Modbus-specific errors that are acknowledged with an exception to the master
Success_Count	Word	0	The number of received requests without protocol errors for this specific slave
MB_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to this MB_DB parameter of the Modbus_Master instruction.

You program can write values to the HR_Start_Offset and Extended_Addressing tags and control the Modbus slave operations. The other tags can be read to monitor the Modbus status.

Rules for Modbus slave communication

- Modbus_Comm_Load must be run to configure a port so that the Modbus_Slave instruction can communicate by means of this port.
- If a port is to respond as slave to a Modbus master, this port may not be programmed with the Modbus_Master instruction.
- Only one instance of Modbus_Slave can be used with a specific port; otherwise you may encounter unexpected behavior.
- The Modbus instructions do not use communication alarm events to control the communication process. Your program must control the communication process by querying the Modbus_Slave instruction for completed send and receive processes.
- The Modbus_Slave instruction must be executed regularly with a frequency that allows a timely response to incoming requests of a Modbus master. We recommend executing Modbus_Slave in each cycle from a program cycle OB. Modbus_Slave can be executed from a cyclic interrupt OB but we do not recommend it, because excessive time delays in the interrupt program can temporarily block the execution of other interrupt programs.

Time control of the Modbus signal

Modbus_Slave must be executed regularly to receive each request of the Modbus master and respond accordingly. The frequency with which Modbus_Slave is executed depends on the timeout value specified for the response by the Modbus master. This can be seen in the figure below.

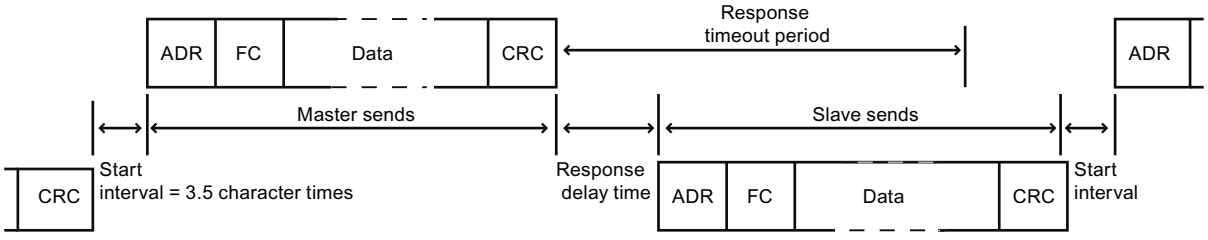

The timeout period of the (RESP_TO) response is the duration that a Modbus master waits for the beginning of an answer from a Modbus slave. This period is not defined by the Modbus protocol, but by a parameter of the Modbus_Comm_Load instruction. As both receiving and sending a frame requires multiple calls of the Modbus_Slave instruction (at least three), you should execute Modbus_Slave at least twelve times during the timeout period for the response of the Modbus master so that the Modbus slave receives and sends data twice as many times as specified by the timeout period.

HR_Start_Offset

The Modbus holding register addresses start at 40001 or 400001. These addresses correspond to the start address of the holding register in the target system memory. But you can configure the HR_Start_Offset tag to configure a start address different than 40001 or 400001 for the Modbus hold register.

The address 0 in the received frame correspond to the start address of the hold register in the target system memory. Use the tag HR_Start_Offset to configure a start address other than 0 for the Modbus hold register.

You can, for example, configure a hold register with start at MW100 and a length of 100 words. With HR_Start_Offset = 20, the address 20 in the received frame corresponds to the start address of the holding register in the target memory (MW100). Each address in the received frame below 20 and above 119 results in an addressing error.

Table 4-326 Example for addressing the Modbus hold register when DATA_PTR is a pointer to MW100 with a length of 100 words

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (word)	0	99
	S7-1500 address	MW100	MW298
20	Modbus address (word)	20	119
	S7-1500 address	MW100	MW298

HR_Start_Offset is a word value which specifies the start address of the Modbus hold register and is saved in the Modbus_Slave instance data block. You select this public static tag by means of the parameter drop-down list once you have added Modbus_Slave in your program.

If you have added Modbus_Slave to an LAD network, for example, you can go to a previous network and assign the value HR_Start_Offset using the move command. The value must be assigned prior to execution of Modbus_Slave.

Enter Modbus slave tag using the standard DB name:

1. Position the cursor in the OUT1 parameter field and enter the character m.
2. Select the required instance DB of the Modbus_Slave instruction from the drop-down list.
3. Position the cursor to the right of the DB name (after the quotation mark) and enter a point.
4. Select "Modbus_Slave_DB.HR_Start_Offset" in the drop-down list.

Instruction versions

Version 4.0 is functionally identical to version 3.0 and its version number was only incremented due to internal measures.

Frame structure (S7-300, S7-400)

Extended_Addressing

You access the Extended_Addressing tag as described for the HR_Start_Offset reference, except that the Extended_Addressing tag is a Boolean value.

You can configure a single byte (Modbus standard) or two bytes (Extended_Adressing = TRUE) with Extended_Adressing = FALSE for addressing the Modbus slave. Extended addressing is used to address more than 247 devices in a single network. With

Extended_Adressing = TRUE you can address up to 65535 addresses. The following example shows a Modbus frame.

Table 4-327 Slave address with one byte (Byte 0)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	
Request	Slave address	Function code	Start address		Data		
Valid response	Slave address	Function code	Length	Data...			
Error message	Slave address	0xxx	Exception code				

Table 4-328 Slave address with two bytes (Byte 0 and Byte 1)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Request	Slave address		Function code	Start address		Data	
Valid response	Slave address		Function code	Length	Data...		
Error message	Slave address		0xxx	Exception code			

Frame description

Data traffic between master and slave / slave and master starts with the slave address, following by the function code. The data is then transferred. The structure of the data field depends on the function code used. The checksum (CRC) is transmitted at the end of the frame.

Function code 1 - This function allows individual output bits to be read

Table 4-329 FC 1 - Read output bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 1	Start address		Number of outputs	
Valid response	Slave address	Function code 1	Length [1]	Output data [3)]		
Error message	Slave address	0x81	Exception code [2]	---		

[1] Length: If there is a remainder when the number of outputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

[3] The output data can comprise multiple bytes

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 1	Start address		Number of outputs	
Valid response	Slave address		Function code 1	Length [1]	Output data		
Error message	Slave address		0x81	Exception code [2]	---		

[1] Length: If there is a remainder when the number of outputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

[3] The output data can comprise multiple bytes

Function code 2 - This function allows individual input bits to be read

Table 4-330 FC 2 - Read input bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 2	Start address		Number of inputs	
Valid response	Slave address	Function code 2	Length [1]	Input data		
Error message	Slave address	0x82	Exception code [2]	---		

[1] Length: If there is a remainder when the number of inputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 2	Start address		Number of inputs	
Valid response	Slave address		Function code 2	Length [1]	Input data		
Error message	Slave address		0x82	Exception code [2]	---		

[1] Length: If there is a remainder when the number of inputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

Function code 3 - This function allows individual registers to be read

Table 4-331 FC 3 - Read hold register

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 3	Start address		Number of registers	
Valid response	Slave address	Function code 3	Length [1]	Register data		
Error message	Slave address	0x83	Exception code [2]	---		

[1] Length: Number of bytes

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 3	Start address		Number of registers	
Valid response	Slave address		Function code 3	Length [1]	Register data		
Error message	Slave address		0x83	Exception code [2]	---		

[1] Length: Number of bytes

[2] E code: 01 or 02 or 03 or 04

Function code 4 - This function allows individual registers to be read

Table 4-332 FC 4 - Read input words

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 4	Start address		Number of input words	
Valid response	Slave address	Function code 4	Length [1]	Input data		
Error message	Slave address	0x84	Exception code [2]	---		

[1] Length: 2 * number of input words

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 4	Start address		Number of input words	
Valid response	Slave address		Function code 4	Length [1]	Input data		
Error message	Slave address		0x84	Exception code [2]	---		

[1] Length: 2 * number of input words

[2] E code: 01 or 02 or 03 or 04

Function code 5 - This function can set or delete individual bits

Table 4-333 FC 5 - Write an output bit

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 5	Start address		Value	
Valid response	Slave address	Function code 5	Length	Value		
Error message	Slave address	0x85	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 5	Start address		Value	
Valid response	Slave address		Function code 5	Length	Value		
Error message	Slave address		0x85	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

Function code 6 - This function allows individual registers to be written

Table 4-334 FC 6 - Write hold register

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 6	Address		Register	
Valid response	Slave address	Function code 6	Address		Register	
Error message	Slave address	0x86	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 6	Address		Register	
Valid response	Slave address		Function code 6	Address		Register	
Error message	Slave address		0x86	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

Function code 8 - This function is used to check the communication connection

Table 4-335 FC 8 - Slave status

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 8	Diagnostic code		Test value	
Valid response	Slave address	Function code 8	Diagnostic code		Test value	
Error message	Slave address	0x88	Exception code [1]	---		

[1] E code: 01 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 8	Diagnostic code		Test value	
Valid response	Slave address		Function code 8	Diagnostic code		Test value	
Error message	Slave address		0x88	Exception code [1]	---		

[1] E code: 01 or 03 or 04

Function code 11 - This function can read 2 bytes of "Status word" and 2 bytes of "Event counter"

Table 4-336 FC 11 - Event counter for slave communication

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 11	---			
Valid response	Slave address	Function code 11	Status		Event counter	
Error message	Slave address	0x8B	Exception code [1]	---		

[1] E code: 01 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 11	---			
Valid response	Slave address		Function code 11	Status		Event counter	
Error message	Slave address		0x8B	Exception code [1]	---		

[1] E code: 01 or 04

Function code 15 - This function allows multiple bits to be written

Table 4-337 FC 15 - Write one/multiple output bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte n
Query	Slave address	Function code 15	Start address		Number of output words		Byte counter [1]	Value	
Valid response	Slave address	Function code 15	Start address		Number of output words		---		
Error message	Slave address	0x8F	Exception code [2]	---					

[1] Byte counter: If there is a remainder when the number of bytes is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte n
Query	Slave address		Function code 15	Start address		Number of output words		Byte counter [1]	Value	
Valid response	Slave address		Function code 15	Start address		Number of output words		---		
Error message	Slave address		0x8F	Exception code [2]		---				

[1] Byte counter: If there is a remainder when the number of bytes is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

Function code 16 - This function allows individual or multiple registers to be written

Table 4-338 FC 16 - Write one/multiple hold registers

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte n
Query	Slave address	Function code 16	Start address		Number of registers		Byte counter [1]	Value	
Valid response	Slave address	Function code 16	Start address		Number of registers		---		
Error message	Slave address	0x90	Exception code [2]	---					

[1] Byte counter: Number of registers * 2

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte n
Query	Slave address		Function code 16	Start address		Number of registers		Byte counter [1]	Value	
Valid response	Slave address		Function code 16	Start address		Number of registers		---		
Error message	Slave address		0x90	Exception code [2]		---				

[1] Byte counter: Number of registers * 2

[2] E code: 01 or 02 or 03 or 04

Error messages (S7-300, S7-400)

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		

Error code	Description	Remedy
16#7002	Interim call: Data transmission running	-
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (bytes)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
16#8193	The module does not support supply voltage diagnostics L+.	Select a suitable value for "Diagnostics interrupt". The following are valid: Diagnostics interrupt deactivated or diagnostic interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)

Error code	Description	Remedy
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire operation (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire operation. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		

Error code	Description	Remedy
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see AUTOHOT-SPOT).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (bytes) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)

Error code	Description	Remedy
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer on startup is deactivated (0) • Clear receive buffer on startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end delimiters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
16#81DF	The module has reset the interface to the FB for one of the following reasons: • Module was restarted • Module parameters were reassigned • CPU STOP	—
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Break in receive line to partner. Reconnect or switch on partner.

Error code	Description	Remedy
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (bytes) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
16#81F2	The DSR signal cannot be set because the module has the type DTE.	Check the configured type of the module. The module type must be DCE (data communication equipment).
16#81F3	The DTR signal cannot be set because the module has the type DCE.	Check the configured type of the module. The module type must be DTE (data terminal equipment).
16#81F4	Block header error (e.g. incorrect block type or incorrect block length)	Check the instance DB and the block header.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditions points to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 KB • Non-optimized memory area: 4 KB Note: If the pointer points to an optimized memory area, do not send more than 1 KB.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in the RDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter before the 1st call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 [1)]	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 [1)]	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 [1)]	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 [1)]	Receive_Conditions does not point to a structure of the typeReceive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)

Error code	Description	Remedy
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the frame header and data string or the connection is terminated with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.

Error code	Description	Remedy
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.

Error code	Description	Remedy
Configuration error - Modbus_Slave		
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.

Error code	Description	Remedy
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 6323) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 6323) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 6323) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80D1	The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that the same data transmission rate, parity and stop bit number are configured for both devices.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
Communication error - Modbus_Master		

Error code	Description	Remedy
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#80C9	The slave does not respond within the time set by Blocked_Proc_Timeout.	Check the setting for Blocked_Proc_Timeout. Check if the module has been configured with the Modbus_Comm_Load instruction. The module may possibly need to be reconfigured using Modbus_Comm_Load after a pull/plug or after voltage recovery.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1)]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1)]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1)]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer

Error code	Description	Remedy
16#8855 [1]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200% to +200%
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.

Error code	Description	Remedy
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request. Check if the instructions USS_Read_Param, USS_Write_Param or USS_Port_Scan have reported an error. If they have, check the value of the static tag USS_DB. w_USSExtendedError of the USS_Drive_Control instruction.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.
Error codes, general		
16#8FFF	The module is not ready temporarily due to a reset.	Repeat the request.

[1] Only with instructions for S7-300/400 CPUs

PtP data link CP 340 (S7-300, S7-400)

P_SEND: Sending Data (S7-300, S7-400)

Description

The P_SEND instruction transfers data blocks from a DB to CP 340 as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the P_SEND instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

Data transmission is initiated by a positive edge at the input REQ. Data transmission can take several calls (program cycles), depending on the data volume.

The P_SEND instruction can be called in the cycle by setting signal state "1" at the R parameter input. This setting cancels the transmission to CP 340 and resets the P_SEND instruction to the initial state. Data that has already been received by CP 340 is still sent to the communication partner. A static signal state "1" at the input R indicates that data transmission is disabled.

Parameter LADDR specifies the address of the CP 340 to be addressed.

Output DONE indicates "Job completed without errors". ERROR indicates error events. A corresponding event number is displayed in STATUS parameter (Page 6367) if an error occurs. If no error occurs, STATUS has a value 0. DONE and ERROR/STATUS are also output for RESET of the P_SEND instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of CP 340 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8190 send data, starting from data byte
LEN	INPUT	INT	Data length 1 ≤ LEN ≤ 1024, specified in number of bytes
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors STATUS parameter (Page 6367) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6367) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

The P_SEND instruction works in combination with an I_SEND instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.

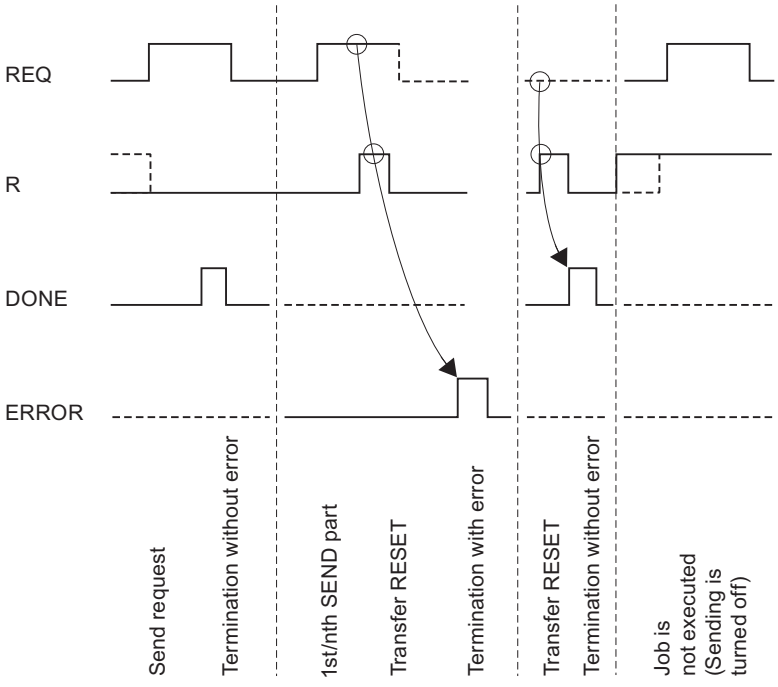

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rules

Note

The P_SEND instruction has no parameter check. If incorrect parameters are assigned, the CPU can change to STOP mode.

Note

The CP-CPU startup routine of the P_SEND instruction must be completed before the CP 340 can execute a job triggered after the CPU changed from STOP to RUN . A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the CP 340 .

P_RCV: receiving data (S7-300, S7-400)

Description

The P_RCV instruction transfers data from the CP 340 to an S7 data area as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the P_RCV instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from CP 340. An active transmission can be canceled by setting signal state "0" at parameter EN_R. (Because transmission may sometimes not start up again after setting parameter EN_R to "1", you should set parameter R to "1" in addition to setting EN_R to "0" to cancel an active transmission.) The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R. Data transmission can take several calls (program cycles), depending on the data volume.

If the instruction detects signal state "1" at the R parameter, the current transmission job is canceled and the P_RCV instruction is reset to the initial state. Receiving is disabled as long as signal state "1" is set at parameter R. If the signal state returns to "0", the canceled message frame is received again from the beginning.

Parameter LADDR is used to select the CP 340 to be addressed.

Output NDR indicates "Job completed without errors/data received" (all data read) and ERROR indicates error events. STATUS (Page 6367) indicates the error event numbers. If no error occurs, STATUS has a value 0. NDR and ERROR/STATUS are also output by the RESET of P_RCV (parameter LEN == 16#00). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enable data reading
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled.
LADDR	INPUT	INT	Start address of CP 340 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Receive data block no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8190 Receive data starting from data byte
NDR [1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00;
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6367) contains the error information.
LEN [1]	OUTPUT	INT	Length of the message frame received 1 ≤ LEN ≤ 1024, specified in number of bytes
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6367) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

The P_RCV instruction works in combination with an I_RCV instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0E occurs, you can examine the SFCERR or SFCSTATUS tag for additional details.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the input circuit of EN_R and R.

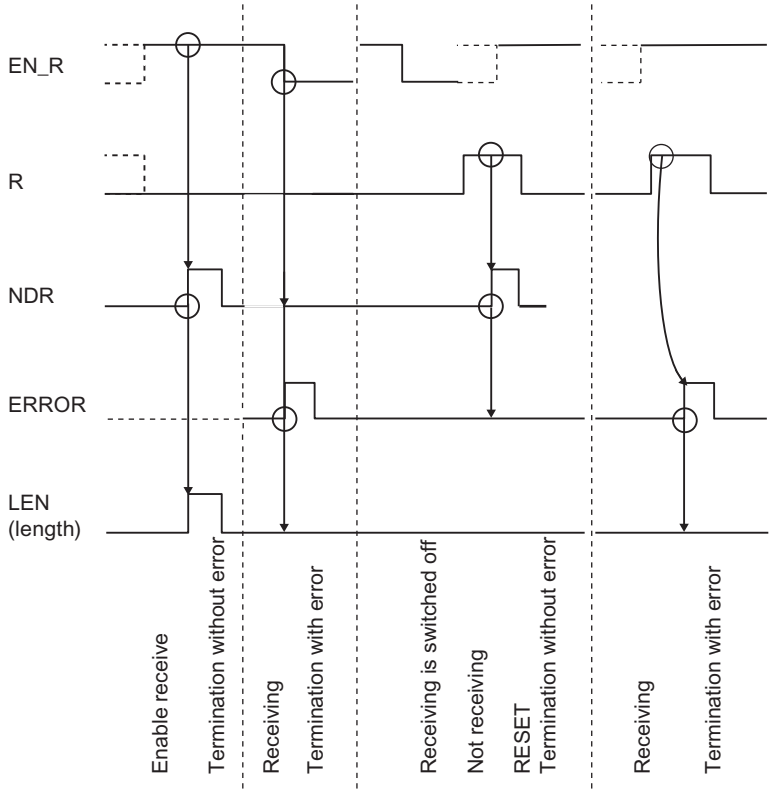

Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

Rules

Note

The P_RCV instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CP-CPU startup routine of the P_RCV instruction must be completed before the CP 340 can execute a job triggered after the CPU changed from STOP to RUN.

P_PRINT: Print message text with up to 4 variables (S7-300, S7-400)

Description

The P_PRINT instruction is available for printing message texts. The P_PRINT instruction transfers a process message to the CP 340, for example. CP 340 logs the process message to the connected printer.

The P_PRINT instruction sends up to four variables to the CP 340. For data transmission, the P_PRINT instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The DB_NO and DBB_NO parameters enable access to the pointers (to data blocks) for the format string and the four variables. The pointers must be stored without gaps and in a specific sequence in the parameterized data block. This is the pointer DB (see Figure "Pointer DB").

A positive edge at the input REQ initiates the transmission of the message text. The frame starts with the format string of the message text, This is followed by tags 1 to 4.

Data transmission can take several calls (program cycles), depending on the data volume.

The P_PRINT instruction can be called in the cycle by setting signal state "1" at parameter input R. This setting cancels the transmission to CP 340 and resets the P_PRINT instruction to the initial state. Data that has already been received by CP 340 is still sent to the communication partner. A static signal state "1" at the input R indicates that the transmission of print jobs is disabled.

Parameter LADDR specifies the address of the CP 340 to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, STATUS has a value 0. DONE and ERROR/STATUS are also output for RESET of the P_PRINT instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Printing is disabled
LADDR	INPUT	INT	Start address of CP 340 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Pointer to pointer DB: CPU-specific, zero not allowed (The pointers to variables and format string are stored in the pointer DB in a fixed order (see previous figure).)

Parameters	Declaration	Data type	Description
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8162 Pointer as of data byte
DONE [1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6367) contains the error information.
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6367) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Assignment in the Data Area, Instance DB

The P_PRINT instruction works in combination with an I_PRINT instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Assignment in the Data Area, Pointer DB

The P_PRINT instruction uses the DB_NO and DBB_NO parameters to access a pointer DB in which the pointers to the data blocks containing the message texts and variables are stored in a fixed sequence. You have to create the pointer DB.

The figure shows the syntax of the pointer DB that is addressed by means of the DB_NO and DBB_NO parameters of the P_PRINT instruction.

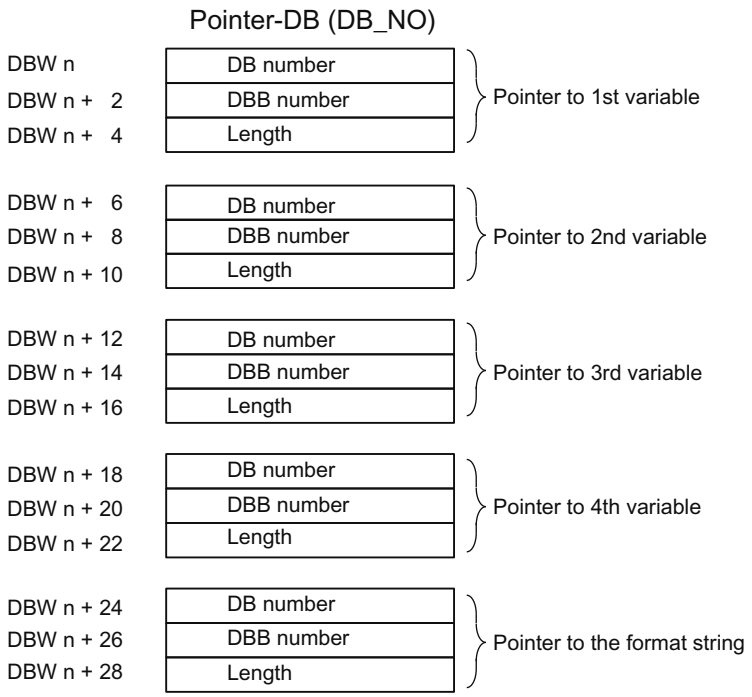

Permissible DB Number

The permissible DB numbers are CPU-specific. If the value 16#00 is specified as the DB number for "Pointer to variable", this variable is interpreted as not present and the pointer is set to the next variable or the format string.

If the DB number is equal to the value 16#00 in "Pointer to format string", the print job is canceled and event ID 16#1E43 is indicated at parameter output STATUS of the P_PRINT instruction.

Permissible DBB Number

The variable or format string begins at the parameterized DBB number. The variables can have a maximum length of 32 bytes and the format string can have a maximum length of 150 bytes.

If the maximum length is exceeded, the print job is canceled and event ID 16#1E41 is indicated at parameter output STATUS of the P_PRINT instruction.

Permissible Length

The entry length in the pointer DB is to be set for each display type (data type) independently from the precision used.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.

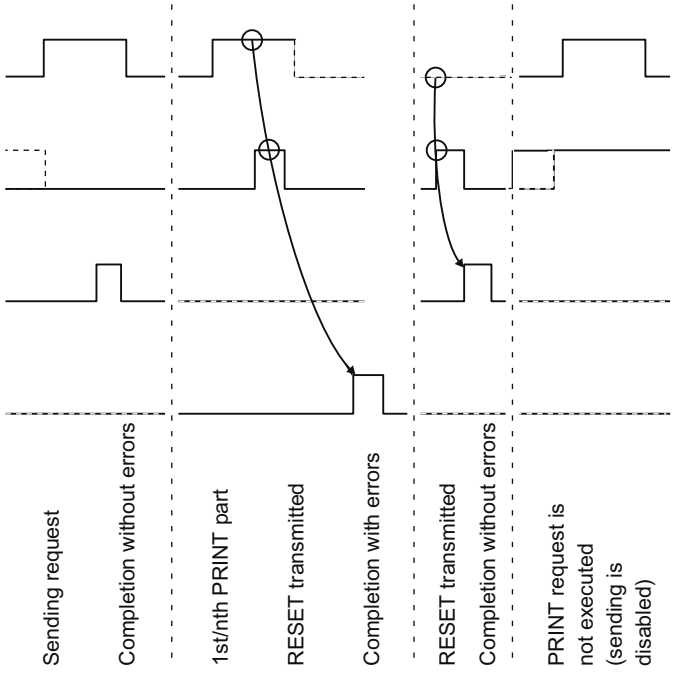

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. It does not have to have a signal state of "1" during the entire transmission operation.

Rules

Note

The P_PRINT instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CP-CPU startup routine of the P_PRINT instruction must be completed before the CP 340 can execute a job triggered after the CPU changed from STOP to RUN. A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the CP 340.

P_RESET: Delete receive buffer (S7-300, S7-400)

Description

The P_RESET instruction clears the entire input buffer of CP 340. All message frames in memory are discarded. A message frame received at the time the P_RESET instruction is called is saved.

A positive edge at the input REQ enables the instruction. The job can run over several calls (program cycles).

Parameter LADDR specifies the address of the CP 340 to be addressed.

The DONE output indicates "Job completed without errors". ERROR indicates error events. A corresponding event number is displayed in STATUS parameter (Page 6367) if an error occurs. If no errors occurs, STATUS has the value 0. If an error occurs, the binary result BR is reset. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
LADDR	INPUT	INT	Start address of CP 340 The start address is taken from STEP 7.
DONE [1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR [1]	OUTPUT	BOOL	Job canceled with errors STATUS parameter (Page 6367) contains the error information.
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6367) contains the error information
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

The P_RESET instruction works in combination with an I_P_RESET instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ.

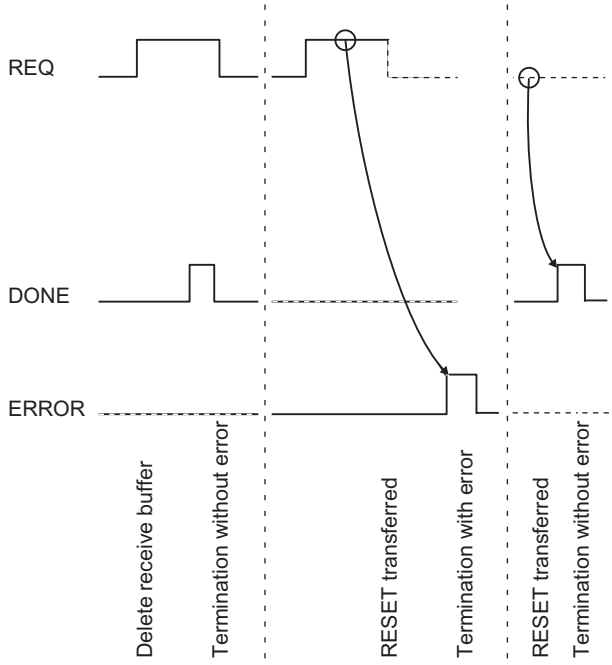

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rule

Note

The P_RESET instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

V24_STAT_340 / V24_STAT: Reading accompanying signals from the RS232C interface (S7-300, S7-400)

Description

The V24_STAT_340 (CP 340) / V24_STAT (CP 341) instruction reads the RS 232C accompanying signals from the CP 34x and makes them available to you at the block parameters. The V24_STAT_340 / V24_STAT instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The RS 232C accompanying signals are updated each time the instruction is called (cyclic polling). The CP 34x updates the I/O status at intervals of 20 ms. The inputs/outputs are constantly updated independently of this.

The binary result BR is not affected. The instruction does not output an error message.

Parameter LADDR is used to select the CP 34x be addressed.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 34x The start address is taken from STEP 7.
DTR_OUT	OUTPUT	BOOL	**D**ata **t**erminal **r**eady, CP 34x ready (CP 34x output)
DSR_IN	OUTPUT	BOOL	**D**ata **s**et **r**eady, Communication partner ready (CP 34x input)
RTS_OUT	OUTPUT	BOOL	**R**equest **t**o **s**end, CP 34x ready to send (CP 34x output)
CTS_IN	OUTPUT	BOOL	**C**lear **t**o **s**end, Communication partner can receive data from CP 34x (response to RTS = ON of CP 34x) (CP 34x input)
DCD_IN	OUTPUT	BOOL	**D**ata **C**arrier **d**etect, Receive signal level (CP 34x input)
RI_IN	OUTPUT	BOOL	**R**ing **I**ndicator, Ring indicator (CP 34x input)

Allocation in the data area

The V24_STAT_340 / V24_STAT instruction does not occupy any data areas.

Rule

Note

A minimum pulse duration is necessary to detect a signal change. The CPU cycle time, the update time on CP 34x and the response time of the communication partner are decisive variables.

V24_SET_340 / V24_SET: Writing accompanying signals to the RS232C interface (S7-300, S7-400)

Description

You can set or reset the interface outputs via the parameter inputs of instruction V24_SET_340 (CP 340) / V24_SET (CP 341). The V24_SET_340 / V24_SET instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The binary result BR is not affected. The instruction does not output an error message.

Parameter LADDR is used to select the CP 34x be addressed.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 34x The start address is taken from STEP 7.
RTS	INPUT	BOOL	Request to send, CP 34x ready to send (Control CP 34x output)
DTR	INPUT	BOOL	Data terminal ready, CP 34x ready (Control CP 34x output)

Allocation in the data area

The V24_SET_340 / V24_SET instruction does not occupy any data areas.

STATUS parameter (S7-300, S7-400)

Event classes

Event class	Meaning
5	Error executing a CPU job
7	Send errors

Event class	Meaning
8	Receive errors
30 ($1E_H$)	Errors in communication between the CP and CPU

STATUS parameter

Error code (W#16#...)	Description	Remedy
0502	Job not permitted in this CP operating mode (e.g., device interface not parameterized)	Evaluate the diagnostic interrupt and rectify the error accordingly.
0505	**Only for printer drivers:** System data block with message texts not available on the CP	Use the parameter assignment software to configure the message texts and then carry out a warm restart.
0506	**Only for printer drivers:** Message text not available	Use the parameter assignment software to configure the message texts and then carry out a warm restart.
0507	**Only for printer drivers:** Message text too long	Edit the message text to reduce it to a length of fewer than 150 characters (or no more than 250 characters if it contains variables).
0508	**Only for printer drivers:** Too many conversion instructions	You have configured more conversion instructions than variables. Conversion instructions without associated variables will be ignored.
0509	**Only for printer drivers:** Too many variables	You have configured more variables than conversion instructions. Variables without associated conversion instructions will not be output.
050A	**Only for printer drivers:** Unknown conversion instruction	Check the conversion instruction. Undefined or unsupported conversion instructions are replaced in the printout with ******.
050B	**Only for printer drivers:** Unknown control instruction	Check the control instruction. Undefined or unsupported control instructions will be ignored. The control instruction will not be output as text either.
050C	**Only for printer drivers:** Conversion instruction not executable	Check the conversion instruction. Conversion instructions that cannot be executed are output in the printout in accordance with the defined width and the valid remainder of the conversion instruction or the default display with * characters.
050D	**Only for printer drivers:** Width in conversion instruction too small or too great	Correct the specified width of the variable in the conversion instruction on the basis of the variable's maximum possible number of characters in text-based display types (A, C, D, S, T, Y, Z). Only as many characters as will fit in the specified width appear in the printout; the text is truncated to this width. In all other cases, * characters are output corresponding to the width.

Error code (W#16#...)	Description	Remedy
050E	**Only for 3964(R) and ASCII drivers:** Invalid frame length	Frame length is > 1,024 bytes. The rest of the frame (> 1,024 bytes) is still being received by the CP 340; this means that the first part of the frame is discarded. Select a shorter frame length.
051B	**Only for printer drivers:** Precision invalid	Correct the specified precision in the conversion instruction. The precision is initialized with a dot prefix to identify and limit the width (example: ".2" to output the decimal point and 2 decimal places). Precision is only relevant to display types A, D, F and R. It is ignored otherwise.
051C	**Only for printer drivers:** Variable invalid (variable length incorrect/incorrect type)	Correct the specified variable. The corresponding table indicates the possible data types for each display type.
051E	**Only for printer drivers:** The "line end sequences" sent with this job (i.e.: $R / $L / $N) do not fit (any longer) on the (initial) page.	Increase the length of your page, reduce the number of lines (or line feeds) or spread your printout over a number of pages.
0701	**Only for 3964(R):** Sending of the first repetition: • An error was detected when transmitting the frame or • The partner requested a repetition by means of a negative acknowledgment character (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0702	**Only for 3964(R):** Error establishing connection: After STX was sent, NAK or any other code (except for DLE or STX) was received.	Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) interconnected in the data link.
0703	**Only for 3964(R):** Acknowledgment delay time (QVZ) exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive or there is a break in the transmission line, for example. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) interconnected in the data link.
0704	**Only for 3964(R):** Termination by partner: One or more codes were received from the partner during sending.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to check this state.

Error code (W#16#...)	Description	Remedy
0705	**Only for 3964(R):** Negative acknowledgment when sending	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to check this state.
0706	**Only for 3964(R):** End-of-connection error: • Partner rejected frame at end of connection with NAK or a random string (except for DLE) or • Acknowledgment characters (DLE) received too early.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to check this state.
0707	**Only for 3964(R):** Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send frame: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device too slow or faulty. You can interconnect an interface tester (FOXPG) in the transmission line to check this state.
0708	**Only for ASCII drivers and printer drivers:** The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
070B	**Only for 3964(R):** Initialization conflict cannot be resolved because both partners have high priority.	Change the parameter assignment.
070C	**Only for 3964(R):** Initialization conflict cannot be resolved because both partners have low priority.	Change the parameter assignment.
0801	**Only for 3964(R):** Expectation of the first repetition: An error was detected on receiving a frame and the CP requested repetition from the partner via a negative acknowledgment (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.

Error code (W#16#...)	Description	Remedy
0802	**Only for 3964(R):** Error establishing connection: • In idle mode, one or more random codes (other than NAK or STX) were received or • After an STX was received, the partner sent more characters without waiting for the response DLE. After partner power ON: • While partner is being switched on, the CP receives an undefined character.	Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) interconnected in the data link.
0805	**Only for 3964(R):** Logical error while receiving: After DLE was received, a further random character (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the frame header and data string or the connection is terminated with DLE ETX. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) interconnected in the data link.
0806	**Character delay time (ZVZ) exceeded:** • Two successive characters were not received within character delay time or **Only for 3964(R):** • First character after sending of DLE during connection setup was not received within character delay time.	Partner device too slow or faulty. Check this using an interface test device (FOXPG) interconnected in the data link.
0807	**Illegal frame length:** A zero-length frame has been received.	Receipt of a zero-length frame is not an error. Check why the communication partner is sending frames without user data.
0808	**Only for 3964(R):** Error in block check character (BCC): The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) interconnected in the data link.
0809	**Only for 3964(R):** The number of repetitions must the set to the same value.	Declare a block waiting time at the communication partner identical to that of the CP 340. Check for malfunction of the communication partner; you may need to use an interface test device (FOXPG) that is switched into the transmission line.
080A	There is no free receive buffer available: No receive buffer space available for receiving data.	Increase the call rate for the P_RCV instruction.

Error code (W#16#...)	Description	Remedy
080C	Transmission error: • A transmission error (parity error, stop bit error, overflow error) was detected. **Only for 3964(R):** • If this happens during send or receive operations, repetition is started. • If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. • If the SF LED (red) and the RxD LED (green) light up, there is a break in the connecting cable (cable break) between the two communication partners.	Disturbances on the data link cause frame repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits.
080D	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
0810	**Only for ASCII drivers:** Parity error: If the SF LED (red) and the RxD LED (green) light up, there is a break in the connecting cable (cable break) between the two communication partners.	Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0811	**Only for ASCII drivers:** Character frame error: If the SF LED (red) and the RxD LED (green) light up, there is a break in the connecting cable (cable break) between the two communication partners.	Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0812	**Only for ASCII drivers:** More characters were received after the CP had sent XOFF or set CTS to OFF.	Reset the parameters for the communication partner or read data from CP more quickly.
0818	**Only for ASCII drivers:** DSR = OFF or CTS = OFF	The partner has switched the DSR or CTS signal to "OFF" before or during a transmission. Check the partner's control of the RS 232C accompanying signals.
1E0D	Job aborted due to warm restart, hot restart or reset	
1E0E	Static error when calling RD_REC SFC or RDREC SFB. The RET_VAL return value for the SFC/SFB is made available to you for evaluation in the SFCERR or SFCSTATUS variable respectively on the instance DB.	Load the SFCERR or SFCSTATUS variable from the instance DB.

Error code (W#16#...)	Description	Remedy
1E0F	Static error when calling WR_REC SFC or RDREC SFB. The RET_VAL return value for the SFC/SFB is made available to you for evaluation in the SFCERR or SFCSTATUS variable respectively on the instance DB.	Load the SFCERR or SFCSTATUS variable from the instance DB.
1E41	The number of bytes specified at the FBs' LEN parameter is not permissible.	You must stay within a range of values of 1 to 1,024 bytes.
1E41	P_PRINT instruction: The number of bytes specified for the variable or format string in the pointer DB under length is not permissible.	You must specify a permissible length: 32 bytes for variables, 150 bytes for format strings.
1E43	P_PRINT instruction: No pointer available for format string.	Enter the data block no. and data word no. for the format string in the pointer DB.

Note

An error message is only output if the ERROR bit (job canceled with error) is also set. In all other cases the STATUS word is zero.

Calling the SFCERR or SFCSTATUS variable

You can call the SFCERR or SFCSTATUS variable to obtain more detailed information about the pending event class 30 error, 14 (1E0EH) or 15 (1E0FH).

You can load the SFCERR or SFCSTATUS variable only by means of symbolic access to instance DB of the corresponding instruction.

Error messages entered in the SFCERR variable can be found in the system functions SFC 58 under "WR_REC: Write data record to I/O" and SFC 59 under "RD_REC: Read data record from I/O".

Error messages entered in the SFCSTATUS variable can be found in the system functions SFB 52 under "RDREC: Read data record" and SFB 53 under "WRREC: Write data record".

See also

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

WR_REC: Write data record to I/O (Page 5260)

RD_REC: Read data record from I/O (Page 5256)

PtP data link CP 341 (S7-300, S7-400)

P_SND_RK: Fetching or sending data (S7-300, S7-400)

P_SND_RK

The P_SND_RK instruction distinguishes the following applications:

- P_SND_RK: Send data with 3964(R) and ASCII drivers (Page 6374)
- P_SND_RK: Send data with RK 512 (Page 6377)
- P_SND_RK: Fetch data with RK 512 (Page 6381)

P_SND_RK: Send data with 3964(R) and ASCII drivers (S7-300, S7-400)

Description

The P_SND_RK instruction transfers data blocks from a DB to the CP 341 as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the P_SND_RK instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

Data transmission is initiated by a positive edge at the input REQ. Data transmission can take several calls (program cycles), depending on the data volume.

The P_SND_RK instruction can be called in the cycle by setting signal state "1" at the R parameter input. This setting cancels the transmission to CP 341 and resets the P_SND_RK instruction to the initial state. Data that has already been received by CP 341 is still sent to the communication partner. If the signal state remains static at "1" at the input R, sending is disabled.

Parameter LADDR specifies the address of the CP 341 to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for a RESET of the P_SND_RK instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Send DB no.: CPU-specific, zero is not allowed
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8190 Send data starting from data byte
LEN	INPUT	INT	Data length 1 ≤ LEN ≤ 4096, specified in number of bytes
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6401) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6401) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Note

The R_CPU_NO, R_TYP, R_NO, R_OFFSET, R_CF_BYT and R_CF_BIT parameters are irrelevant for procedure 3964(R) and the ASCII driver and require no data. Similarly, the SF parameter requires no data since 'S' for Send is set by default.

Allocation in the data area

The P_SND_RK instruction works in combination with an I_SND_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR or SFCSTATUS variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.

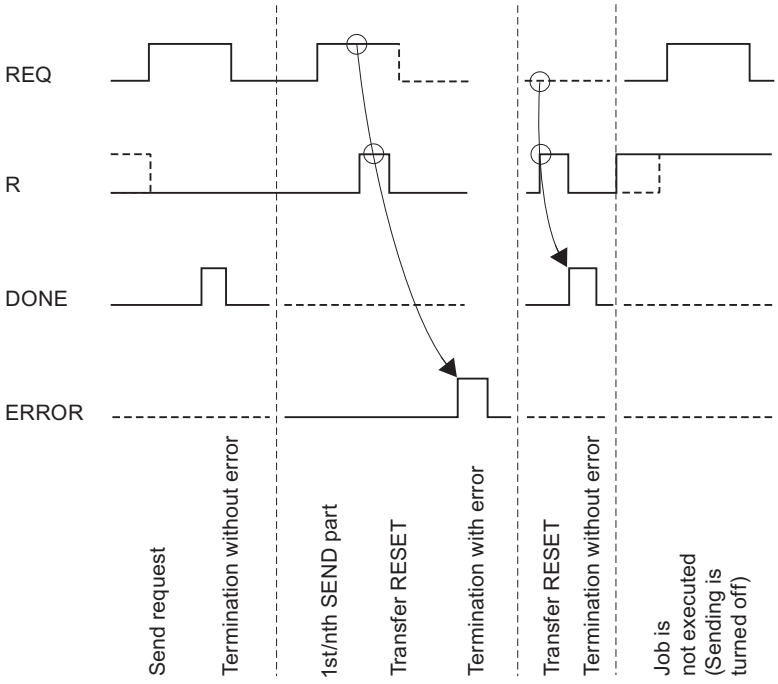

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rules

Note

The P_SND_RK instruction does not have a parameter check. If the parameter assignment is incorrect, the CPU can go to STOP mode.

Note

The CP-CPU startup routine of the P_SND_RK instruction must be completed before the CP 341 can execute a job triggered after the CPU changed from STOP to RUN . A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the CP 341.

P_SND_RK: Send data with RK 512 (S7-300, S7-400)

Description

The P_SND_RK instruction can be used with parameter setting SF = 'S' to send data from an S7 data area to a CP 341.

Operating principle

Data transmission is initiated by a positive edge at the input REQ. Data transmission can take several calls (program cycles), depending on the data volume (LEN).

Parameter LADDR specifies the address of the CP 341 to be addressed.

The area of the data blocks is the only permissible source for data to be sent. The source is appropriately specified by setting the DB number (DB_NO) and offset (DBB_NO) of the first data byte to be transmitted from this data block.

Data blocks (DB) and extended data blocks (DX) are valid destination data types (R_TYP). The destination is appropriately specified by means of the CPU number (R_CPU_NO, relevant only to multi-processor communication), the data type (R_TYP: DB or DX), the DB number (R_NO) and the offset (R_OFFSET) to which the first byte is to be written.

The interprocessor communication flag byte and bit on the partner CPU are specified in R_CF_BYT and R_CF_BIT.

The P_SND_RK instruction can be called in the cycle by setting signal state "1" at the R parameter input. This setting cancels the transmission to CP 341 and resets the P_SND_RK instruction to the initial state. Data that has already been received by CP 341 is still sent to the communication partner. A static signal state "1" at the input R indicates that data transmission is disabled.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for a RESET of the P_SND_RK instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
SF	INPUT	CHAR	Selection for Send data or Fetch data SF = 'S' (Send) Default: 'S'
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled. Default: 0
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Data block number of source Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number of source 0 ≤ DBB_NO ≤ 8190 Send data starting from data byte
LEN	INPUT	INT	Data length of the message frame to be sent 1 ≤ LEN ≤ 4096, specified in number of bytes; only even-numbered values are appropriate
R_CPU_NO	INPUT	INT	CPU number of the partner CPU 0 ≤ R_CPU_NO ≤ 4, only for multiprocessor mode. Default: 1
R_TYP	INPUT	CHAR	Address type on the partner CPU 'D': Data block ‘X’: Expanded data block
R_NO	INPUT	INT	Data block number on the partner CPU 0 ≤ R_NO ≤ 255
R_OFFSET	INPUT	INT	Data byte number on the partner CPU 0 ≤ R_OFFSET ≤ 510, even-numbered values only
R_CF_BYT	INPUT	INT	Interprocessor communication flag byte on partner CPU 0 ≤ R_CF_BYTE ≤ 255 Default: 255 (means: without interprocessor communication flag)
R_CF_BIT	INPUT	INT	Interprocessor communication flag bit on the partner CPU 0 ≤ R_CF_BIT ≤ 7
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6401) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6401) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Information in the message frame header

The table below shows the information in the header of the RK 512 message frame.

Table 4-339 Information in the RK 512 message frame header for "Send data" request

Source on your S7 automation system (local CPU)	To the destination, partner CPU	Message frame header, bytes		
		3/4 command type	5/6 Z-DBNR/Z offset	7/8 Number in
Data block	Data block	AD	DB/DW	Words

Source on your S7 automation system (local CPU)	To the destination, partner CPU	Message frame header, bytes		
Data block	Expanded data block	AD	DB/DW	Words
D-DBNO: Destination data block number D-Offset: Destination start address DW: Offset in words				

Allocation in the data area

The P_SND_RK instruction works in combination with an I_SND_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Special features for sending data

Note the following special features with regard to sending data:

- RK 512 allows only an even-numbered amount of data to be sent. If you specify an odd-numbered amount of data for the length (LEN), an additional filler byte with a value of "0" is sent at the end of the data.
- RK 512 allows only an even-numbered offset. If you specify an odd-numbered offset, the data is stored in the partner starting from the next lower even-numbered offset.
  Example: Offset is 7, data is stored from byte 6.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.

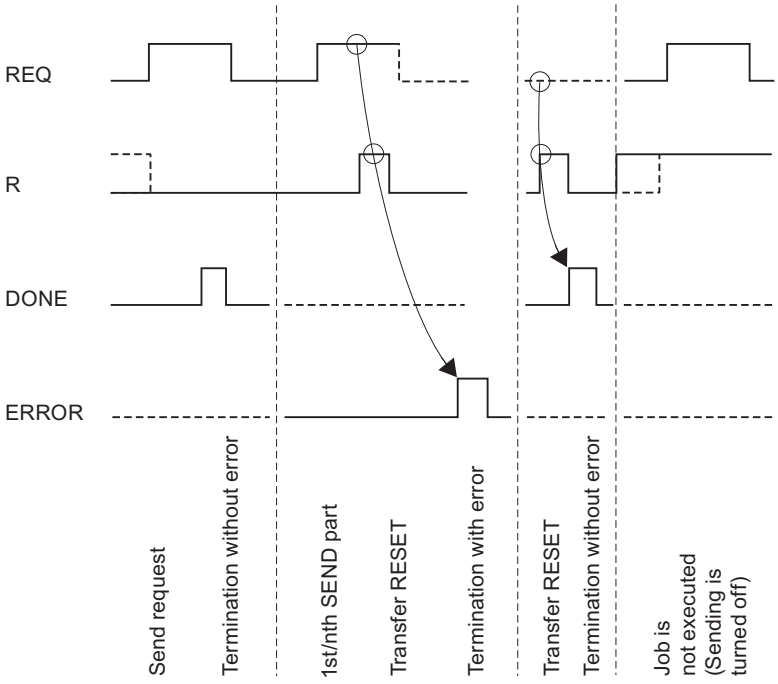

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rule

Note

The P_SND_RK instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

P_SND_RK: Fetch data with RK 512 (S7-300, S7-400)

Description

The P_SND_RK instruction can be used with parameter setting SF = F to fetch data from a remote communication partner and save it to an S7 data area of your automation system.

Note

To fetch data from a CP 341, you must always program a P_RCV_RK instruction on the CP 341.

Operating principle

Data transmission is initiated by a positive edge at the input REQ. Data transmission can take several calls (program cycles), depending on the data volume (LEN).

Parameter LADDR specifies the address of the CP 341 to be addressed.

The communication partner from which the data is fetched is specified by the CPU number (R_CPU_NO, relevant only to multiprocessor communication). The following data types (R_TYP) are valid sources for the data to be fetched: data blocks, extended data blocks, bit memory, inputs, outputs, counters and timers. The source is fully specified by the data type (R_TYP), the data block number (R_NO, relevant only to data blocks and expanded data blocks) and the offset(R_OFFSET) of the first data byte in this area to transmit.

The interprocessor communication flag byte and bit on the partner CPU are specified in R_CF_BYT and R_CF_BIT.

The only permissible destination areas are data blocks (DB). The destination is specified by the data block number (DB_NO) and offset (DBB_NO) to which the first byte is written.

The P_SND_RK instruction can be called in the cycle by setting signal state "1" at the R parameter input. This setting cancels the transmission from CP 341 and resets the P_SND_RK instruction to the initial state. A static signal status "1" at the input R indicates that fetching is disabled.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for a RESET of the P_SND_RK instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
SF	INPUT	CHAR	Selection for Send data or Fetch data SF = 'F' (Fetch)default value: 'S' (Send)
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Fetching is disabled. Default: 0

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number of destination Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number of destination 0 ≤ DBB_NO ≤ 8190 Send data starting from data byte
LEN	INPUT	INT	Data length of the message frame to be fetched 1 ≤ LEN ≤ 4096, specified in number of bytes[1]
R_CPU_NO	INPUT	INT	CPU number of the partner CPU 0 ≤ R_CPU_NO ≤ 4, only for multiprocessor mode. Default: 1
R_TYP	INPUT	CHAR	Address type on the partner CPU 'D': Data block 'X': Expanded data block 'M': Memory bit 'I': Inputs 'O': Outputs 'C': Counters 'T': Timers
R_NO	INPUT	INT	Data block number on the partner CPU 0 ≤ R_NO ≤ 255
R_OFFSET	INPUT	INT	Data byte number on the partner CPU
R_CF_BYT	INPUT	INT	Interprocessor communication flag byte on partner CPU 0 ≤ CF_BYTE ≤ 255 Default: 255 (means: without interprocessor communication flag)
R_CF_BIT	INPUT	INT	Interprocessor communication flag bit on the partner CPU 0 ≤ CF_BIT ≤ 7
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6401) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6401) contains the error information.
[1] The parameter is available until the next call of the instruction!			

R_TYP, R_NO and R_OFFSET parameters

Source on the partner CPU	R_TYP	R_NO	R_OFFSET (in bytes) (This value is specified by the partner CPU.)
Data block	'D'	0 - 255	0 - 510 (only even-numbered values are appropriate!)
Expanded data block	'X'	0 - 255	0 - 510 (only even-numbered values are appropriate!)
Bit memory	'M'	irrelevant	0 - 255
Inputs	'I'	irrelevant	0 - 255
Outputs	'O'	irrelevant	0 - 255
Counters	'C'	irrelevant	0 - 255
Timers	'T'	irrelevant	0 - 255

Information in the message frame header

The table below shows the information in the header of the RK 512 message frame.

Source on the partner CPU	To the destination, your S7 automation system (local CPU)	Message frame header, bytes		
		3/4 com-mand type	5/6 Q-DBNR/Q offset	7/8 Number in
Data block	Data block	ED	DB/DW	Words
Expanded data block	Data block	EX	DB/DW	Words
Bit memory	Data block	EM	Byte address	Bytes
Inputs	Data block	EI	Byte address	Bytes
Outputs	Data block	EO	Byte address	Bytes
Counters	Data block	EC	Counter no.	Words
Timers	Data block	ET	Timer number	Words
S-DBNO: Source data block number S-Offset: Source start address				

Allocation in the data area

The P_SND_RK instruction works in combination with an I_SND_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Special features for (expanded) data blocks

Note the following special features with regard to fetching data from data blocks and expanded data blocks:

- RK 512 allows only an even-numbered amount of data to be fetched. If you specify an odd-numbered amount for the length (LEN), an extra byte is always sent. In the destination DB, however, the correct amount of data is entered.
- RK 512 allows only an even-numbered offset. If you specify an odd-numbered offset, the data is fetched from the partner starting from the next lower even-numbered offset. Example: Offset is 7, data is fetched as of byte 6.

Special features for timers and counters

If you fetch timers or counters from the communication partner, remember that 2 bytes are fetched for each timer or counter. For example, if you want to fetch 10 counters, you must enter 20 as the length.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERRORparameters, depending on the input circuit of REQ and R.

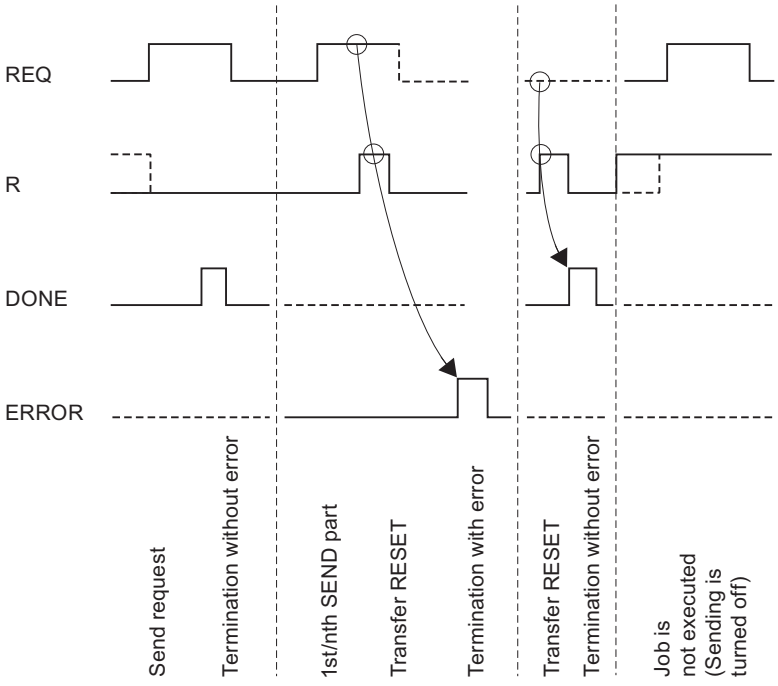

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rule

Note

The P_SND_RK instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

P_RCV_RK: Receive data or Provide data (S7-300, S7-400)

P_RCV_RK

The P_RCV_RK instruction distinguishes the following applications:

- P_RCV_RK: Receiving data with 3964(R) or ASCII driver (Page 6385)
- P_RCV_RK: Receive data with RK 512 (Page 6388)
- P_RCV_RK: Provide data with RK 512 (Page 6392)

P_RCV_RK: Receiving data with 3964(R) or ASCII driver (S7-300, S7-400)

Description

The P_RCV_RK instruction transfers data from CP 341 to an S7 data area as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the P_RCV_RK instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from CP 341 . An active transmission can be canceled by setting signal state "0" at parameter EN_R . (Because transmission may sometimes not start up again after setting parameter EN_R to "1", you should set parameter R to "1" in addition to setting EN_R to "0" to cancel an active transmission.) The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R . Data transmission can take several calls (program cycles), depending on the data volume.

If the instruction detects signal state "1" at parameter R, the current transmission job is canceled and the P_RCV_RK instruction is reset to the initial state. Receiving is disabled as long as signal state "1" is set at parameter R.

Parameter LADDR is used to select the CP 341 to be addressed.

Output NDR indicates "Job completed without errors/data received" (all data read) and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. NDR and ERROR/STATUS are also output for the RESET of the P_RCV_RK instruction (parameter LEN == 16#00). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enable data reading
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled.
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Receive DB No.: CPU-specific, zero is not allowed
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8190 Receive data starting from data byte
NDR [1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00;
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6401) contains the error information.
LEN [1]	OUTPUT	INT	Length of the message frame received 1 ≤ LEN ≤ 4096, specified in number of bytes
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6401) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Note

The L_TYP, L_NO, L_OFFSET, L_CF_BYT and L_CF_BIT parameters are irrelevant for the 3964(R) procedure and ASCII driver and require no data.

Allocation in the data area

The P_RCV_RK instruction works in combination with an I_RCV_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0E occurs, you can examine the SFCERR or SFCSTATUS tag for additional details. This error tag must be loaded via symbolic access to the instance DB.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the input circuit of EN_R and R.

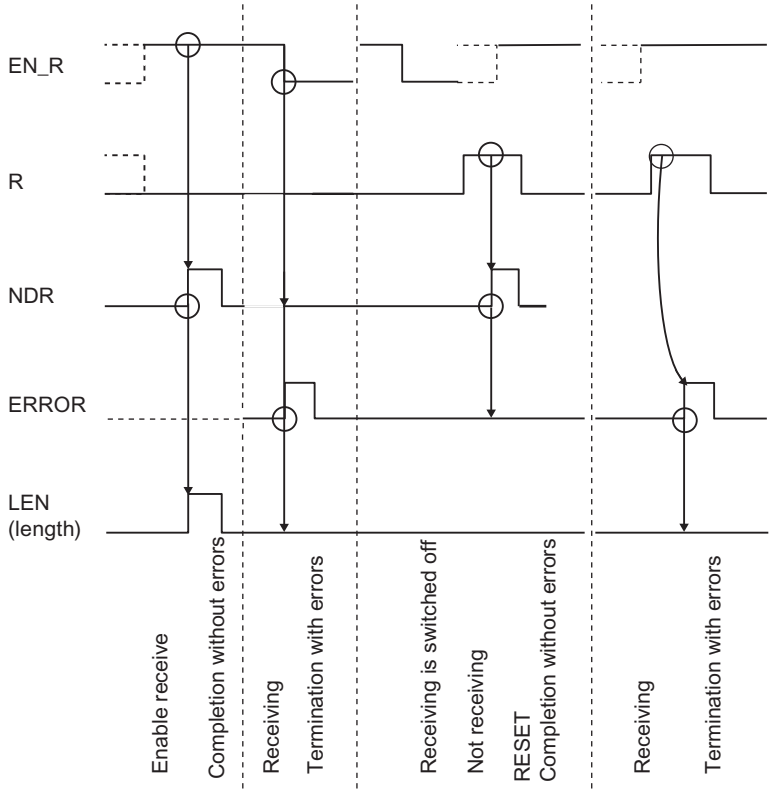

Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

Rules

Note

The P_RCV_RK instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CP-CPU startup routine of the P_RCV_RK instruction must be completed before the CP 341 can execute a job triggered after the CPU changed from STOP to RUN.

P_RCV_RK: Receive data with RK 512 (S7-300, S7-400)

Description

The P_RCV_RK instruction sends data from the CP 341 to an S7 data area. For data transmission, the P_RCV_RK instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from CP 341 . An active transmission can be canceled by setting signal state "0" at parameter EN_R . The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R . Data transmission can take several calls (program cycles), depending on the data volume.

Parameter LADDR specifies the address of the CP 341 to be addressed.

If the communication partner specifies the destination "DB", the data is placed in the data area specified in the RK 512 message frame header. Using the (L_...) parameter, you can identify the destination area type (L_TYP), the destination data block number (L_NO, only relevant for L_TYP = DB), the offset in the destination area (L_OFFSET) and the length (LEN) of transmitted data. If the partner specifies the data destination "DX", the data is saved to the data block (DB) that is specified at the DB_NO and DBB_NO parameters.

If the instruction detects signal state "1" at parameter R, the current transmission job is canceled and the P_RCV_RK instruction is reset to the initial state. Receiving is disabled as long as the signal state at the R parameter is "1".

Output NDR indicates "Job completed without errors/data received" (all data read). The L_TYP, L_NO and L_OFFSET parameters indicate the data storage location for the duration of one cycle. The L_CF_BYT and L_CF_BIT and the length LEN parameters of the respective job are also indicated for one cycle.

Using interprocessor communication flags

The interprocessor communication flag specified in the RK 512 message frame header is verified prior to data reception. The data is not transmitted unless the value of the interprocessor communication flag is "0". When transmission is completed, the interprocessor communication flag is set to the value "1" and output for one cycle (NDR) at the instruction.

The user program can evaluate the interprocessor communication flag in order to find out if the transmitted data can be processed. Once the data has been processed, the user must reset the interprocessor communication flag to "0". The communication partner can now issue a SEND request again.

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enables data receipt
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled. Default: 0
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number of the receive data (destination) Receive data block no.: CPU-specific, zero not allowed (Relevant only for DX data destination)
DBB_NO	INPUT	INT	Data byte number of the receive data (destination) 0 ≤ DBB_NO ≤ 8190 Receive data starting from data byte (Relevant only for DX data destination)
L_TYP[1]	OUTPUT	CHAR	Type of area on local CPU (destination) 'D': Data block
L_NO [1]	OUTPUT	INT	Data block number on local CPU (destination) 0 ≤ L_NO ≤ 255
L_OFFSET[1]	OUTPUT	INT	Data byte number on local CPU (destination) 0 ≤ L_OFFSET ≤ 510
L_CF_BYT[1]	OUTPUT	INT	Interprocessor communication flag byte on local CPU 0 ≤ L_CF_BYTE ≤ 255255 means: Without interprocessor communication flag

Parameters	Declaration	Data type	Description
L_CF_BIT [1]	OUTPUT	INT	Interprocessor communication flag bit on local CPU 0 ≤ L_CF_BIT ≤ 7
NDR[1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6401) contains the error information.
LEN[1]	OUTPUT	INT	Length of the message frame received 0 ≤ LEN ≤ 4096, specified in number of bytes
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6401) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

The P_RCV_RK instruction works in combination with an I_RCV_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0E occurs, you can examine the SFCERR or SFCSTATUS variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the input circuit of EN_R and R.

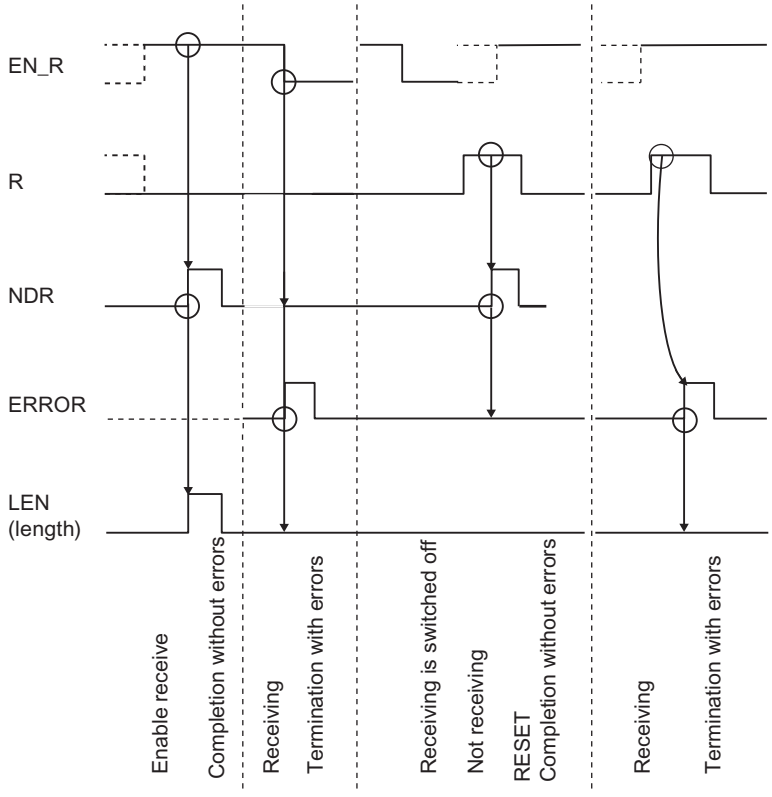

Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

How to Handle an Error

ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, STATUS has a value 0. NDR and ERROR/STATUS are also output for the RESET of the P_RCV_RK instruction (parameter LEN == 16#00). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Rule

Note

The P_RCV_RK instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

P_RCV_RK: Provide data with RK 512 (S7-300, S7-400)

Description

The P_RCV_RK instruction must be called whenever the communication partner executes a "Fetch data" job (FETCH job).

The P_RCV_RK instruction provides data from an S7 data area to CP 341 . For data transmission, the P_RCV_RK instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

A (static) signal state "1" at the EN_R parameter enables a check that determines whether data is to be provided to CP 341 . An active transmission can be canceled by setting signal state "0" at parameter EN_R . The canceled request is terminated with an error message (STATUS output). The job is disabled as long as signal state "0" is set at parameter EN_R . Data transmission can take several calls (program cycles), depending on the data volume.

The first RK 512 message frame specifies the type of source area (L_TYP), the source data block number (L_NO, relevant only if L_TYP = DB), the offset in the source area (L_OFFSET) and the length (LEN) of data to be provided. The instruction evaluates this message frame information and transmits the requested data to the CP 341. The DB_NO and DBB_NO parameters are insignificant for the P_RCV_RK instruction.

Parameter LADDR specifies the address of the CP 341 to be addressed.

If the instruction detects signal state "1" at parameter R, the current transmission job is canceled and the P_RCV_RK instruction is reset to the initial state. The request is disabled as long as the signal state at the R parameter is "1".

Output NDR indicates "Job completed without errors/data received" (all data read). The source of fetched data is indicated at the L_TYP, L_NO and L_OFFSET parameters for the duration of one cycle (valid data types: data blocks, input bytes, output bytes, timers and counters). The L_CF_BYT and L_CF_BIT and the length LEN parameters of the respective job are also indicated for one cycle.

Note

The length of timers or counters the communication partner fetches from the CP 341 is limited to 32 bytes (16 timers or counters, 2 bytes each).

Using interprocessor communication flags

Following receipt of the message frame, the interprocessor communication flags specified in the RK 512 message frame header are checked. The data is not provided unless the value of the interprocessor communication flag is "0". When transmission is completed, the interprocessor communication flag is set to the value "1" and output for one cycle (NDR) at the instruction.

The user program can evaluate the interprocessor communication flag in order to find out if the provided data can be accessed again. Once the data has been processed, the user must reset the interprocessor communication flag to "0". The communication partner can now issue a FETCH request again.

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enables data provision
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Disables providing. Default: 0
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	irrelevant
DBB_NO	INPUT	INT	irrelevant
L_TYP[1]	OUTPUT	CHAR	Type of area on local CPU (source) 'D': Data block 'M' Memory bit 'I': Inputs 'O': Outputs 'C': Counters 'T': Timers
L_NO[1]	OUTPUT	INT	Data block number on local CPU (source) 0 ≤ L_NO ≤ 255 (only relevant, if L_TYP = D)
L_OFFSET[1]	OUTPUT	INT	Data byte number on local CPU (source) 0 ≤ L_OFFSET ≤ 510 (dependent on the area type)
L_CF_BYT[1]	OUTPUT	INT	Interprocessor communication flag byte on local CPU 0 ≤ CF_BYTE ≤ 255 255 means: Without interprocessor communication flag
L_CF_BIT[1]	OUTPUT	INT	Interprocessor communication flag bit on local CPU 0 ≤ CF_BIT ≤ 7
NDR[1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6401) contains the error information.
LEN[1]	OUTPUT	INT	Length of the message frame received 0 ≤ LEN ≤ 4096, specified in number of bytes
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6401) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

The P_RCV_RK instruction works in combination with an I_RCV_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0E occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the input circuit of EN_R and R.

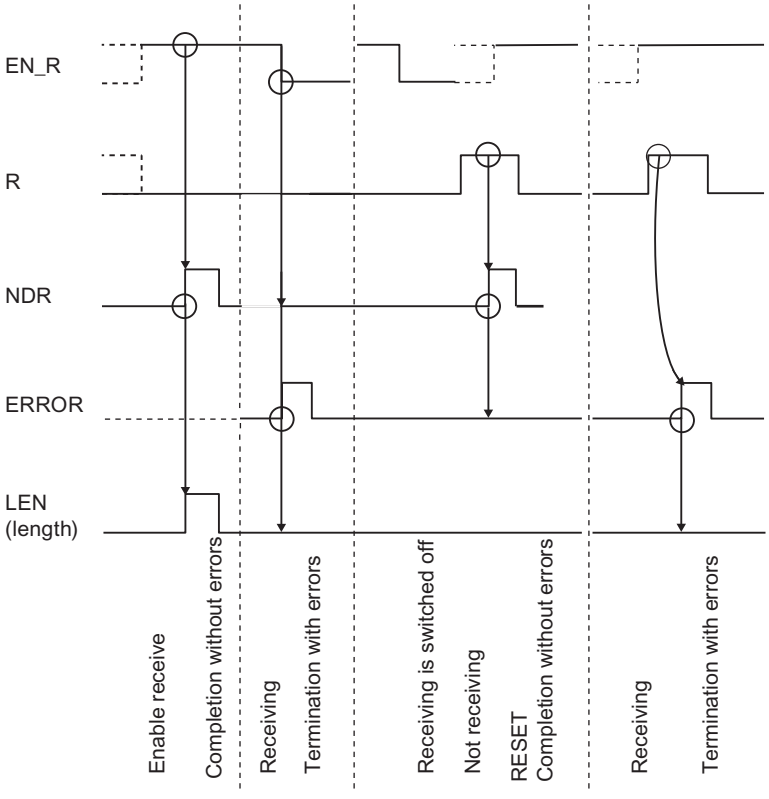

Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

How to Handle an Error

ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. NDR and ERROR/STATUS are also output for the RESET of the P_RCV_RK instruction (parameter LEN == 16#00). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Rule

Note

The P_RCV_RK instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

P_PRT341: Print message text with up to 4 variables (S7-300, S7-400)

Description

The P_PRT341 instruction is available for printing message texts. The P_PRT341 instruction transfers a process message to the CP 341, for example. CP 341 logs the process message to the connected printer.

Operating principle

The P_PRT341 instruction transmits a message text with up to four variables to CP 341. For data transmission, the P_PRT341 instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The DB_NO and DBB_NO parameters enable access to the pointers (to data blocks) for the format string and the four variables. The pointers must be stored without gaps and in a specific sequence in the parameterized data block. This is the pointer DB (see Figure "Pointer DB").

A positive edge at the input REQ initiates the transmission of the message text. The frame starts with the format string of the message text, This is followed by tags 1 to 4.

Data transmission can take several calls (program cycles), depending on the data volume.

The P_PRT341 instruction can be called in the cycle by setting signal state "1" at the R parameter input. This setting cancels the transmission to CP 341 and resets the P_PRT341 instruction to the initial state. Data that has already been received by CP 341 is still sent to the communication partner. A static signal state "1" at the input R indicates that the transmission of print jobs is disabled.

Parameter LADDR specifies the address of the CP 341 to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for a RESET of the

P_PRT341 instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Printing is disabled
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Pointer to pointer DB: CPU-specific, zero is not allowed (The pointers to variables and format string are stored in the pointer DB in a fixed order.)
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8162 Pointer as of data byte
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6401) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6401) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Assignment in the Data Area, Instance DB

The P_PRT341 instruction works in combination with an I_PRINT instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Assignment in the Data Area, Pointer DB

The P_PRT341 instruction uses the DB_NO and DBB_NO parameters to access a pointer DB in which the pointers to the data blocks containing the message texts and variables are stored in a fixed sequence. You have to create the pointer DB.

The figure shows the syntax of the pointer DB that is addressed by means of the DB_NO and DBB_NO parameters of the P_PRT341 instruction.

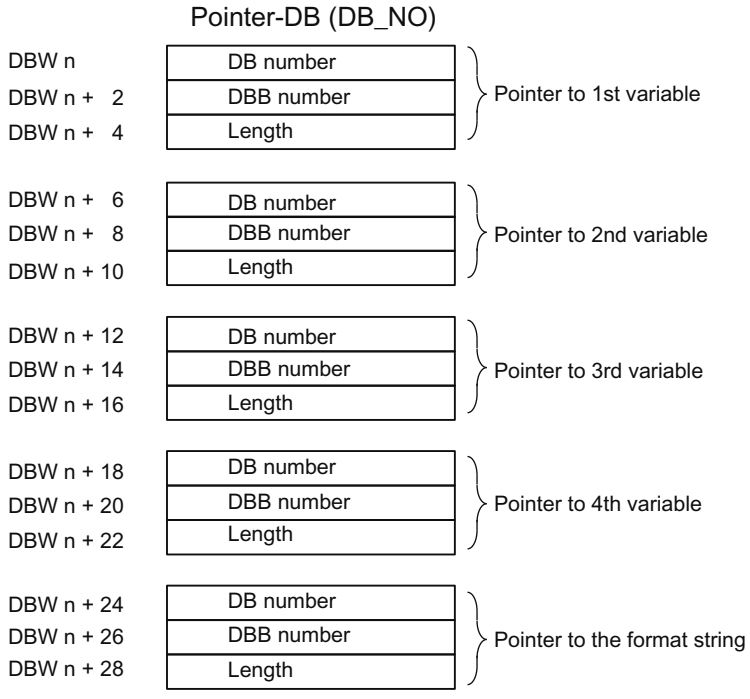

Permissible DB Number

The permissible DB numbers are CPU-specific. If the value 16#00 is specified as the DB number for "Pointer to variable", this variable is interpreted as not present and the pointer is set to the next variable or the format string.

If the DB number is equal to the value 16#00 in "Pointer to format string", the print job is canceled and event ID 16#1E43 is indicated at parameter output STATUS of the P_PRT341 instruction.

Permissible DBB Number

The variable or format string begins at the parameterized DBB number. The variables can have a maximum length of 32 bytes and the format string can have a maximum length of 150 bytes.

If the maximum length is exceeded, the print job is canceled and event ID 16#1E41 is indicated at parameter output STATUS of the P_PRT341 instruction.

Permissible Length

The entry length in the pointer DB is to be set for each display type (data type) independently from the precision used.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.

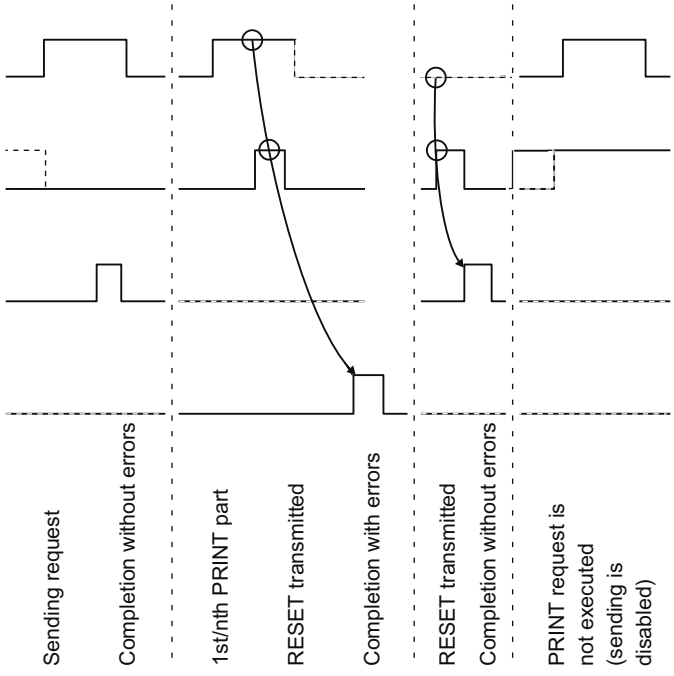

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. It does not have to have a signal state of "1" during the entire transmission operation.

Rules

Note

The P_PRT341 instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CP-CPU startup routine of the P_PRT341 instruction must be completed before the CP 341 can execute a job triggered after the CPU changed from STOP to RUN. A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the CP 341.

V24_STAT_340 / V24_STAT: Reading accompanying signals from the RS232C interface (S7-300, S7-400)

Description

The V24_STAT_340 (CP 340) / V24_STAT (CP 341) instruction reads the RS 232C accompanying signals from the CP 34x and makes them available to you at the block parameters. The V24_STAT_340 / V24_STAT instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The RS 232C accompanying signals are updated each time the instruction is called (cyclic polling). The CP 34x updates the I/O status at intervals of 20 ms. The inputs/outputs are constantly updated independently of this.

The binary result BR is not affected. The instruction does not output an error message.

Parameter LADDR is used to select the CP 34x be addressed.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 34x The start address is taken from STEP 7.
DTR_OUT	OUTPUT	BOOL	**D**ata **t**erminal **r**eady, CP 34x ready (CP 34x output)
DSR_IN	OUTPUT	BOOL	**D**ata **s**et **r**eady, Communication partner ready (CP 34x input)
RTS_OUT	OUTPUT	BOOL	**R**equest **t**o **s**end, CP 34x ready to send (CP 34x output)
CTS_IN	OUTPUT	BOOL	**C**lear **t**o **s**end, Communication partner can receive data from CP 34x (response to RTS = ON of CP 34x) (CP 34x input)
DCD_IN	OUTPUT	BOOL	**D**ata **C**arrier **d**etect, Receive signal level (CP 34x input)
RI_IN	OUTPUT	BOOL	**R**ing **I**ndicator, Ring indicator (CP 34x input)

Allocation in the data area

The V24_STAT_340 / V24_STAT instruction does not occupy any data areas.

Rule

Note

A minimum pulse duration is necessary to detect a signal change. The CPU cycle time, the update time on CP 34x and the response time of the communication partner are decisive variables.

V24_SET_340 / V24_SET: Writing accompanying signals to the RS232C interface (S7-300, S7-400)

Description

You can set or reset the interface outputs via the parameter inputs of instruction V24_SET_340 (CP 340) / V24_SET (CP 341). The V24_SET_340 / V24_SET instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The binary result BR is not affected. The instruction does not output an error message.

Parameter LADDR is used to select the CP 34x be addressed.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 34x The start address is taken from STEP 7.
RTS	INPUT	BOOL	Request to send, CP 34x ready to send (Control CP 34x output)
DTR	INPUT	BOOL	Data terminal ready, CP 34x ready (Control CP 34x output)

Allocation in the data area

The V24_SET_340 / V24_SET instruction does not occupy any data areas.

STATUS parameter (S7-300, S7-400)

Event classes

Event class	Meaning
0	CP startup
1	CP hardware fault
2	Error initializing
3	Incorrect parameter assignment for the instruction
4	The CP detected errors in CP - CPU data traffic
5	Error executing a CPU job
6	Error executing a partner job (for RK 512 only)
7	Send errors
8	Receive errors
9	Received response message frame with error or error message frame from the interconnection partner
10 ($0A_H$)	The CP detected errors in the response message frame of the partner
30 ($1E_H$)	Errors in communication between the CP and CPU

STATUS parameter

Error code(W#16#...)	Description	Remedy
0003	PtP parameters applied	-
0004	Parameter already on CP (timers match)	-
0007	CPU status transition to STOP	-
0008	CPU status transition to RUN/STARTUP	-
0101	Fault while testing operating system EPROM of CP	CP defective; replace CP.
0102	RAM test of CP errored	
0103	Request interface of CP defective	
0110	Fault in CP firmware	Switch module off and on again. If necessary, replace module.
020F	Invalid parameter assignment detected at start of parameter communication. Interface could not be parametered.	Correct invalid parameters and restart.
0301	Invalid source/destination data type or area does not exist (start address, length) Invalid or no DB (e.g. DB 0) or Other data type invalid or missing Interprocessor communication byte number invalid or Invalid interprocessor communication flag bit number or 'S' or 'F' was not selected (for the P_SND_RK instruction)	Check parameter assignment on CPU and CP and correct if necessary. **RK 512 only:** Partner returns invalid parameters in frame header. Check parameters on CPU and CP; possibly create block. See request tables for valid data types. **RK 512 only:** Partner provides incorrect parameters in the message frame header.

Error code(W#16#...)	Description	Remedy
0403	Incorrect, unknown or illegal data type	Check the program, for example to find incorrect parameter assignments in the instruction.
0407	An error has occurred during data transmission between the CPU and CP	If the error message persists, check the parameter assignment of the instructions called in the user program. If the error is signaled immediately after POWER ON, a connection to the CPU is not set up at this time. With the 3964(R) procedure and the ASCII driver, the data transmission of the receiving CP 341 is repeated until the data are transferred to the CPU. With RK 512, the request is acknowledged negatively and must be repeated in the user program. If the message occurs sporadically during active data transmission, the CPU can not accept the data at times. With the 3964(R) procedure and the ASCII driver, the data transmission of the receiving CP 341 is repeated until the data are transferred to the CPU. With RK 512, the request is acknowledged negatively and must be repeated in the user program. To correct or avoid errors, you should increase the call rate for the P_RCV_RK instruction in the user program.
0408	Error during data transmission between the CPU and the CP (receipt)	
	• CPU is temporarily overloaded, request is repeated	• Reduce the number of communication calls
	• The data area of the CPU cannot be accessed for the time being, e.g., because the receive block is not called often enough.	• Increase the call rate for the receive block
	• The data area of the CPU cannot be accessed for the time being, e.g., because the receive block is disabled in the interim (EN=false).	• Check if the receive block has been disabled too long

Error code(W#16#...)	Description	Remedy
0409	Data reception is not possible. Error during data transmission between the CPU and the CP (receipt). Data reception is not possible. After multiple attempts, the request was cancelled after 10 s, because	
	• The receive block is not called.	• Check your user program to determine whether the receive block is being executed.
	• The receive block is disabled.	• Check if the receive block is disabled.
	• The data area of the CPU cannot be accessed.	• Check if the data area to which the data is to be transferred is available.
	• Insufficient length of the data area on the CPU.	• Check the length of the data area
040A	An error has occurred during data transmission between the CPU and the CP. The data transmission was canceled due to a RESET, because: • Destination DB is not available • Destination DB is too short • RESET bit set at the instruction	Create destination DB in the user program or increase the length of the existing destination DB.
0501	Current request aborted as a result of CP restart.	No help available during POWER ON. When changing the parameters of the CP in the programming device, before writing an interface you should ensure there are no more requests running from the CPU.
0502	Job not permitted in this CP operating mode (e.g., device interface not parameterized)	Set the parameters for the device interface.
0505	**Only for printer drivers:** System data block with message texts not available on the CP	Use the parameter assignment software to configure the message texts and then carry out a warm restart.
0506	**Only for printer drivers:** Message text not available	Use the parameter assignment software to configure the message texts and then carry out a warm restart.
0507	**Only for printer drivers:** Message text too long	Edit the message text to reduce it to a length of fewer than 150 characters (or no more than 250 characters if it contains variables).
0508	**Only for printer drivers:** Too many conversion instructions	You have configured more conversion instructions than variables. Conversion instructions without associated variables will be ignored.
0509	**Only for printer drivers:** Too many variables	You have configured more variables than conversion instructions. Variables without associated conversion instructions will not be output.

Error code(W#16#...)	Description	Remedy
050A	**Only for printer drivers:** Unknown conversion instruction	Check the conversion instruction. Undefined or unsupported conversion instructions are replaced in the printout with ******.
050B	**Only for printer drivers:** Unknown control instruction	Check the control instruction. Undefined or unsupported control instructions will be ignored. The control instruction will not be output as text either.
050C	**Only for printer drivers:** Conversion instruction not executable	Check the conversion instruction. Conversion instructions that cannot be executed are output in the printout in accordance with the defined width and the valid remainder of the conversion instruction or the default display with * characters.
050D	**Only for printer drivers:** Width in conversion instruction too small or too great	Correct the specified width of the variable in the conversion instruction on the basis of the variable's maximum possible number of characters in text-based display types (A, C, D, S, T, Y, Z). Only as many characters as will fit in the specified width appear in the printout; the text is truncated to this width. In all other cases, * characters are output corresponding to the width.
050E	**Only for ASCII drivers:** An error occurred while sending. The defined end-of-text characters did not occur within the maximum allowed length or in the case of automatic appending, the maximum allowed transmission length was exceeded.	Extend the end-of-text characters in the transmission buffer at the desired point or select a shorter frame length for automatic appending.
0514	Specified start addresses too high for desired data type or start address or DB/DX number too low.	Refer to the request tables for the permissible start addresses and DB/DX numbers that can be specified in the program.
0515	**RK 512 only:** Incorrect bit number specified in the interprocessor communication flag.	Permissible bit numbers: 0 to 7
0516	**RK 512 only:** Specified CPU number too high.	Permissible CPU no.: none, 0, 1, 2, 3 or 4
0517	Transmission length > 1 kbyte is too long for CP or length is too short for interface parameter.	Split the request up into several shorter requests.
051A	**RK 512 only:** Error sending a command message frame An associated procedure number has just been entered in STATUS.	See the solution for the previous error number.

Error code(W#16#...)	Description	Remedy
051B	**Only for printer drivers:** Precision invalid	Correct the specified precision in the conversion instruction. The precision is initialized with a dot prefix to identify and limit the width (example: ".2" to output the decimal point and 2 decimal places). Precision is only relevant to display types A, D, F and R. It is ignored otherwise.
051C	**Only for printer drivers:** Variable invalid (variable length incorrect/incorrect type)	Correct the specified variable. The corresponding table indicates the possible data types for each display type.
051E	**Only for printer drivers:** The "line end sequences" sent with this job (i.e.: $R / $L / $N) do not fit (any longer) on the (initial) page.	Increase the length of your page, reduce the number of lines (or line feeds) or spread your printout over a number of pages.
0601	Error in 1st command byte (not 00 or FFH)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0602	Error in 3rd command byte (not A, 0 or FFH)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0603	Error in 3rd command byte in the case of continuation frames (command not as for 1st frame)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0604	Error in 4th command byte (command letter incorrect)	Header layout error at partner or a command combination has been requested that is not permitted at the CP. Check the permissible commands. You can interconnect an interface tester in the transmission line to debug the partner device.
0605	Error in 4th command byte in the case of continuation frames (command not as for 1st frame)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0606	Error in 5th command byte (DB number not permissible)	Refer to the request tables for the permissible DB numbers, start addresses or lengths.
0607	Error in the 5th or 6th command byte (start address too high)	Refer to the request tables for the permissible DB numbers, start addresses or lengths.
0608	Error in the 7th or 8th command byte (impermissible length)	Obtain from the request tables the permissible DB/DX numbers, start addresses or lengths.
0609	Error in the 9th and 10th command bytes (illegal coordination flag for this data type or bit number too high).	Header layout error at partner. Find out from the request tables when a coordination flag is permitted.
060A	Error in the 10th command byte (illegal CPU number)	Header layout error at partner.
060B	SEND message frame was longer/shorter than expected (more/less data received than announced in message frame header).	Correction required at the partner

Error code(W#16#...)	Description	Remedy
060C	FETCH command message frame with user data received.	Correction required at the partner
060D	The CP received a message frame during an invalid operating mode:	
	• Receive connection between the CPU and CP is not set up or not yet correctly set up	• Check the parameter assignment of the addressed connection
	• CP startup is not yet completed	• This error message can occur only during CP startup. Repeat the request.
	• The receiving CPU is in STOP mode	• Place the CPU back in RUN mode and repeat the request.
	• Parameters for the interface are currently being assigned	• This is a temporary error. Repeat the request.
060E	Synchronous fault of partner • New (continuation) command message frame received before response message frame was sent. • First command frame expected and continuation frame came. • Continuation command expected and 1st frame came.	This error may be reported after your own automation system is restarted in the case of long message frames or when the partner is restarted. These cases represent normal system startup behavior. The error can also occur during operation as a consequence of error statuses only recognized by the partner. Otherwise, you must assume an error on the part of the partner device. The error may not occur in the case of commands < 128 bytes.
060F	DB locked by coordination function	In local program: Reset the interprocessor communication flag after the last transmission data was processed. In partner program: Repeat the request
0610	Message frame received too short (length < 4 bytes in the case of continuation or response message frames or <10 bytes in the case of command message frames)	You can interconnect an interface tester in the transmission line to debug the partner device.
0611	Frame length and length specified in telegram header are not the same.	You can interconnect an interface tester in the transmission line to debug the partner device.
0612	Error while sending the (continuation) response message frame. An associated procedure error number has been entered in STATUS immediately beforehand.	See solution for the error number entered immediately beforehand in STATUS.
0701	Sending of the first repetition: • An error was detected when transmitting the frame or • The partner requested a repetition by means of a negative acknowledgment character (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame still has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.

Error code(W#16#...)	Description	Remedy
0702	**Only for 3964(R):** Error establishing connection: After STX was sent, NAK or any other code (except for DLE or STX) was received.	Check for malfunction of the partner device, possibly by using interface test device interconnected in the data link.
0703	**Only for 3964(R):** Acknowledgment delay time (QVZ) exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive or there is a break in the transmission line, for example. You can interconnect an interface tester in the transmission line to debug the partner device.
0704	**Only for 3964(R):** Termination by partner: One or more codes were received from the partner during sending.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. If necessary, use an interface test device switched into the transmission line to check.
0706	**Only for 3964(R):** End-of-connection error: • Partner rejected frame at end of connection with NAK or a random string (except for DLE) or • Acknowledgment characters (DLE) received too early.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. If necessary, use an interface test device switched into the transmission line to check.
0707	**Only for 3964(R):** Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send frame: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device too slow or faulty. If necessary, use an interface test device switched into the transmission line to check.
0708	**ASCII driver and printer driver only:** The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
0709	Connection attempt not possible. Number of permitted connection attempts exceeded.	Check the interface cable or the transmission parameters. Also check that receive function between CPU and CP is correctly configured at the partner device.
070A	The data could not be transmitted. The permitted number of transfer attempts was exceeded.	Check the interface cable or the transmission parameters.

Error code(W#16#...)	Description	Remedy
0801	Expectation of the first repetition: An error was detected on receiving a message frame and the CP requested repetition from the partner via a negative acknowledgment (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame still has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0802	**Only for 3964(R):** Error establishing connection: • In idle mode, one or more random codes (other than NAK or STX) were received or • After an STX was received, the partner sent more characters without waiting for the response DLE. After partner power ON: • While partner is being switched on, the CP receives an undefined character.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0805	**Only for 3964(R):** Logical error while receiving: After DLE was received, a further random character (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the frame header and data string or the connection is terminated with DLE ETX. Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0806	Character delay time (ZVZ) exceeded: • Two successive characters were not received within character delay time or Only for 3964(R): • First character after sending of DLE during connection setup was not received within character delay time.	Partner device too slow or faulty. Interconnect an interface tester in the transmission line to see if this is the case.
0808	**Only for 3964(R):** Error in block check character (BCC): The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
080A	There is no free receive buffer available: No receive buffer space available for receiving data.	Increase the call rate for the P_RCV_RK instruction.

Error code(W#16#...)	Description	Remedy
080C	Transmission error: • A transmission error (parity error, stop bit error, overflow error) was detected. **Only for 3964(R):** • If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. **Only in the case of RK 512 and 3964(R):** • If this happens during send or receive operations, repetition is started.	Disturbances on the data link cause frame repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits.
080D	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
0815	Discrepancy between settings for transmission attempts of the CP and the communication partner.	Set the same number of transfer attempts at communications partner as at CP. You can interconnect an interface tester in the transmission line to debug the partner device.
0816	The length of a received message frame exceeded the maximum specified length.	Correction required at the partner
0818	**For ASCII driver only:** DSR = OFF or CTS = OFF	The partner has switched the DSR or CTS signal to "OFF" before or during a transmission. Check the partner's control of the RS 232C accompanying signals.
0902	**Only with RK 512:** Memory access error in partner (memory does not exist) **With SIMATIC S5 as partner:** • Incorrect area at status word or • Data area does not exist (except DB/DX) or • Data area too short (except DB/DX)	Check that the partner has the desired data area and that it is big enough or check the parameters of the called system function block. Check the specified length at the system function block.
0903	**Only with RK 512:** DB/DX access error in the partner (DB/DX does not exist or is too short) **With SIMATIC S5 as partner:** • DB/DX does not exist or • DB/DX too short or • DB/DX number impermissible Permissible source area exceeded with FETCH request	Check that the partner has the desired data area and that it is big enough or check the parameters of the called system function block. Check the specified length at the system function block.
0904	**Only with RK 512:** Partner reports "Illegal request type".	Partner malfunction, because the CP never outputs a system command.

Error code(W#16#...)	Description	Remedy
0905	**Only with RK 512:** Error at the partner or SIMATIC S5 as partner: • Source/destination type not permissible or • Memory error in partner programmable controller or • Error notifying CP/CPU at the partner or • Partner programmable controller is in STOP mode	Check if the partner can transmit the desired data type. Check the structure of the hardware at the partner. Set the mode selector switch of the partner programmable controller to RUN.
0908	**Only with RK 512:** Partner detects synchronous error: Frame sequence error.	This error occurs at restart of your own programmable controller or of the partner. This represents normal system startup behavior. You do not need to correct anything. The error is also conceivable during operation as a consequence of previous errors. Otherwise, you can assume an error on the part of the partner device.
0909	**Only with RK 512:** DB/DX locked at the partner by coordination flag	In partner program: Reset the coordination memory after the last transmission data was processed! In own program: Repeat the request.
090A	**Only with RK 512:** Errors in message frame header that are detected by the partner: Third command byte in the header is incorrect	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090B	**Only with RK 512:** Error in message frame header: 1st or 4th command byte in header is incorrect	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090C	**Only with RK 512:** Partner detects incorrect message frame length (total length).	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090D	**Only with RK 512:** Partner was not yet restarted.	Restart the partner programmable controller or set the mode selector switch on the CP to RUN.
090E	**Only with RK 512:** Unknown error number received in the response message frame.	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.

Error code(W#16#...)	Description	Remedy
0A01	**Only with RK 512:** Synchronization error of partner, because: • Response frame without request • Response frame received before continuation frame sent • Continuation response message frame received after an initial message frame was sent • A first response frame was received after a continuation frame was sent	This error is reported after your own programming device is restarted in the case of long frames or when the partner is restarted. This represents normal system startup behavior. You do not have to correct anything. The error can also occur during operation as a consequence of error statuses only recognized by the partner. Otherwise, you can assume an error on the part of the partner device. The error may not occur in the case of commands < 128 bytes.
0A02	RK 512 only: Error in the structure of the received response message frame (1st byte not 00 or FF)	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A03	RK 512 only: Received response message frame has too many or too few data.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A04	RK 512 only: Response message frame for SEND request arrived with data.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A05	RK 512 only: No response message frame received from partner within monitoring time.	Is the partner a slow device? This error is also often displayed as a consequence of a previous error. For example, procedure receive errors (event class 8) can be displayed after a FETCH message frame was sent. Reason: As a result of disturbances, the response frame could not be received, and the monitoring time elapsed. This error also possibly occurs if the partner is restarted before it could respond to the last received FETCH frame.
1E0D	Job aborted due to warm restart, hot restart or reset	
1E0E	Static error when calling RD_REC SFC. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
1E0F	Static error when calling WR_REC SFC. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
1E41	Invalid number of bytes specified at instruction parameter LEN	You must stay within a range of values of 1 to 4096 bytes.
1E42	P_PRINT_RK instruction: The number of bytes specified for the variable or format string in the pointer DB under length is not permissible.	You must specify a permissible length: 32 bytes for variables, 150 bytes for format strings.
1E43	P_PRINT_RK instruction: No pointer available for format string.	Enter the data block no. and data word no. for the format string in the pointer DB.

Note

An error message is only output if the ERROR bit (job canceled with error) is also set. In all other cases, the STATUS word is zero.

Calling the SFCERR or SFCSTATUS variable

You can call the SFCERR or SFCSTATUS variable to obtain more detailed information about the pending event class 30 error, 14 (1E0EH) or 15 (1E0FH).

You can load the SFCERR or SFCSTATUS variable only by means of symbolic access to instance DB of the corresponding instruction.

Error messages entered in the SFCERR variable can be found in the system functions SFC 58 under "WR_REC: Write data record to I/O" and SFC 59 under "RD_REC: Read data record from I/O".

Error messages entered in the SFCSTATUS variable can be found in the system functions SFB 52 under "RDREC: Read data record" and SFB 53 under "WRREC: Write data record".

See also

WR_REC: Write data record to I/O (Page 5260)

RD_REC: Read data record from I/O (Page 5256)

RDREC: Read data record (Page 5224)

WRREC: Write data record (Page 5226)

PtP data link CP 440 (S7-300, S7-400)

SEND_440: Sending Data (S7-300, S7-400)

Description

The SEND_440 instruction transmits a data block from a DB to the CP 440 as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the SEND_440 instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

Data transmission is initiated by a positive edge at the input REQ. Data transmission can take several calls (program cycles), depending on the data volume.

The SEND_440 instruction can be called in the cycle by setting signal state "1" at parameter input R. This setting cancels the transmission to CP 440 and resets the SEND_440 instruction to the initial state. Data that has already been received by CP 440 is still sent to the

communication partner. A static signal state "1" at the input R indicates that data transmission is disabled.

Parameter LADDR specifies the address of the CP 440 to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for a RESET of the SEND_440 instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of CP 440 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8190 Transmitted data as of data byte; Offset is CPU-specific
LEN	INPUT	INT	Data length 1 ≤ LEN ≤ 400, specified in number of bytes
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6420) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6420) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

The SEND_440 instruction works in combination with an I_SEND_440 instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.

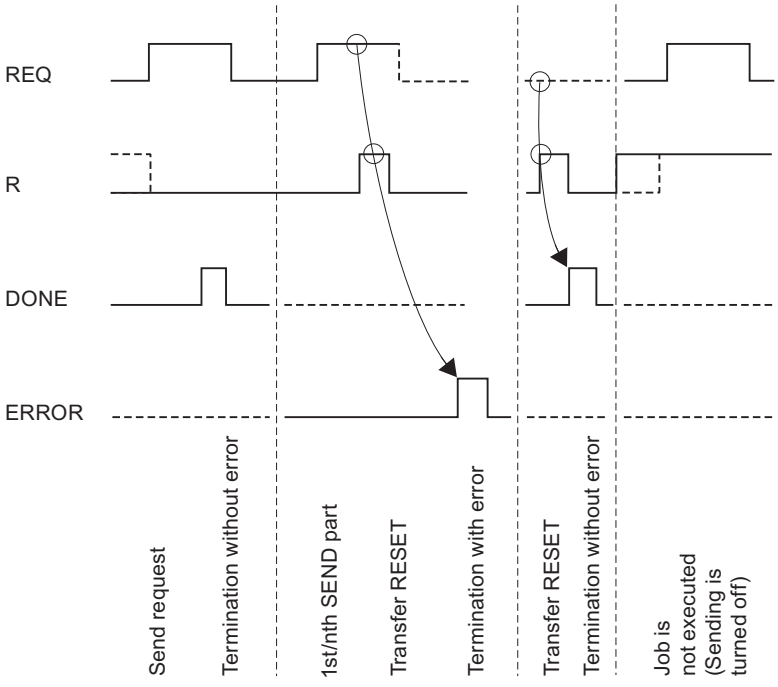

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rules

Note

The SEND_440 instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

RECV_440: receiving data (S7-300, S7-400)

Description

The RECV_440 instruction transfers data from the CP 440 to an S7 data area as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the RECV_440 instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program. .

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from CP 440. An active transmission can be canceled by setting signal state "0" at parameter EN_R. The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R. Data transmission can take several calls (program cycles), depending on the data volume.

If the instruction detects signal state "1" at the R parameter, the current transmission job is canceled and the RECV_440 instruction is reset to the initial state. Receiving is disabled as long as signal state "1" is set at parameter R.

Parameter LADDR is used to select the CP 440 to be addressed.

Output NDR indicates "Job completed without errors/data received" (all data read) and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, STATUS has a value 0. NDR and ERROR/STATUS are also output for the RESET of the RECV_440 instruction (parameter LEN == 16#00). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enable data reading
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled.
LADDR	INPUT	INT	Start address of CP 440 The basic address can be found in the configuration table in STEP 7.

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Data block number Receive data block no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number Offset is CPU-specific
NDR[1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6420) contains the error information.
LEN[1]	OUTPUT	INT	Length of the message frame received 1 ≤ LEN ≤ 400, specified in number of bytes
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6420) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

The RECV_440 instruction works in combination with an I_RECV_440 instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0E occurs, you can examine the SFCERR variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the input circuit of EN_R and R.

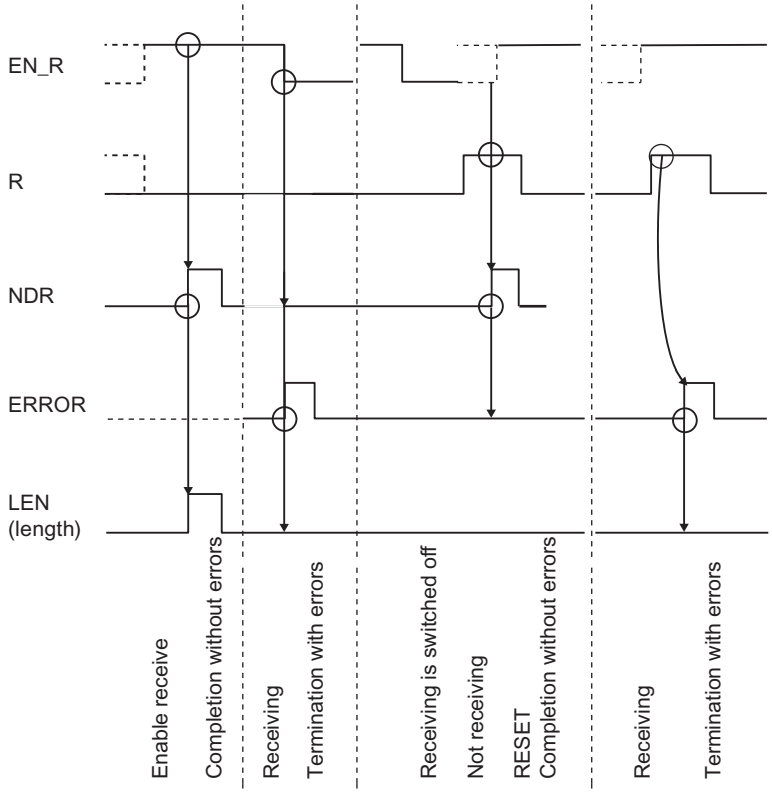

Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

Rule

Note

The RECV_440 instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

RES_RECV: Delete receive buffer (S7-300, S7-400)

Description

The RES_RECV instruction clears the entire input buffer of CP 440. All message frames in memory are discarded. A message frame received at the time the RES_RECV instruction is called is saved.

Operating principle

A positive edge at the input REQ enables the instruction. The job can run over several calls (program cycles).

The RES_RECV instruction can be called in the cycle by setting signal state "1" at parameter input R. This action cancels the delete operation and resets the RES_RECV instruction to the initial state.

Parameter LADDR specifies the address of the CP 440 to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, STATUS has a value 0. DONE and ERROR/STATUS are also output for RESET of the RES_RECV instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of CP 440 The start address is taken from STEP 7.
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6420) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6420) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

The RES_RECV instruction works in combination with an I_RES_RECV instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.

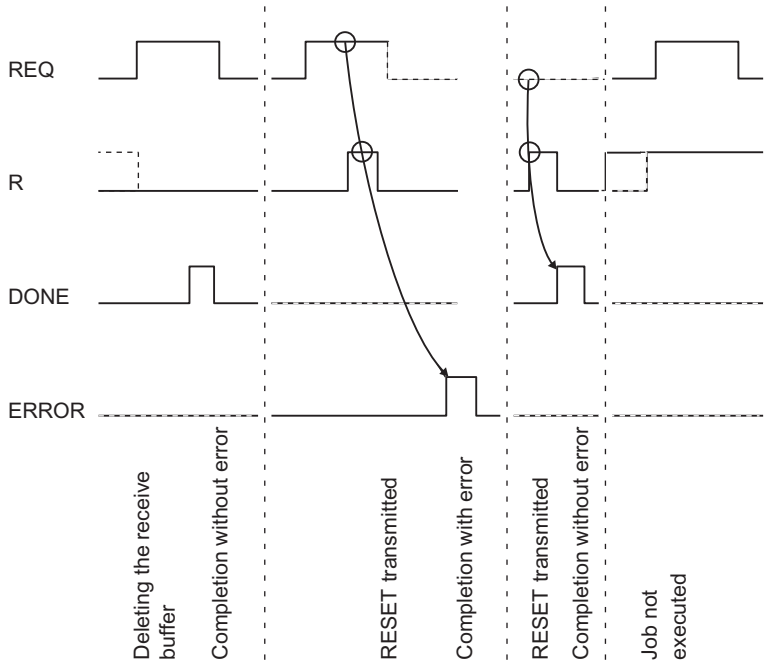

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rule

Note

The RES_RECV instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

STATUS parameter (S7-300, S7-400)

Event classes

Event class	Meaning
0 (0_H)	CP startup
2 (02_H)	Error initializing
5 (05_H)	Error executing a CPU job
7 (07_H)	Send errors
8 (08_H)	Receive errors
11 ($0B_H$)	Warnings
30 ($1E_H$)	Errors in communication between the CP and CPU
128 (80_H)	Module fault

STATUS parameter

Error code(W#16#...)	Description	Remedy
0001	Initialization of the CP completed	-
0003	PtP parameters applied	-
0004	Parameter already on CP (timers match)	-
0007	CPU status transition to STOP	-
0008	CPU status transition to RUN/STARTUP	-
0201	No parameters Parameter memory empty or has unknown contents	Load interface parameters.
0204	Impermissible character frame	Correct the impermissible parameter assignment, load the parameters onto the module and carry out a cold restart.
0205	Impermissible transmission rate	Correct the impermissible parameter assignment, load the parameters onto the module and carry out a cold restart.
020F	Invalid parameter assignment detected at start of parameter communication. Interface could not be parametered.	Correct invalid parameters and restart.
0212	Impermissible • Monitoring time • Character Delay Time Only for 3964(R): • Acknowledgment delay	Correct the impermissible parameter assignment, load the parameters onto the module and carry out a cold restart.

Error code(W#16#...)	Description	Remedy
0501	Current request aborted as a result of CP restart.	No help available during POWER ON. When changing the parameters of the CP in the programming device, before writing an interface you should ensure there are no more requests running from the CPU.
0502	Job not permitted in this CP operating mode (e.g., device interface not parameterized)	Set the parameters for the device interface.
050E	• Invalid frame length or	• Frame length is > 400 bytes. Reduce the message frame length or
	• The configured end-of-text characters did not occur within the maximum permissible length.	• Add the end-of-text characters at the place you want in the send buffer.
0517	Transmission length > 400 bytes exceeds CP capacity.	Split the job up into several shorter jobs.
051D	Send/receive job interrupted by: • Communication block reset • Parameter reassignment	Repeat the call for the communication block.
0522	A new SEND job has been started, although the old job has not yet been completed.	Only start the new SEND job when the old job has been completed with DONE or ERROR.
0701	Only for 3964(R): Sending of the first repetition: • An error was detected when transmitting the frame or • The partner requested a repetition by means of a negative acknowledgment character (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0702	Only for 3964(R): Error establishing connection: After STX was sent, NAK or any other code (except for DLE or STX) was received.	Check for malfunctioning of the partner device, possibly using an interface test device switched into the transmission line.
0703	Only for 3964(R): Acknowledgment delay time (QVZ) exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive or there is a break on the send line, for example. Identify the malfunctioning of the partner device, possibly by using an interface test device switched into the transmission line.
0704	Only for 3964(R): Termination by partner: One or more codes were received from the partner during sending.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester in the transmission line to check the status.

Error code(W#16#...)	Description	Remedy
0705	Only for 3964(R): Negative acknowledgment when sending	Check if the partner also indicates an error; possibly it has not received all of the transmitted data (for example, due to an interrupted data link) or because fatal errors are pending or the characteristics of the partner device is faulty. You can interconnect an interface tester in the transmission line to check the status.
0706	Only for 3964(R): End-of-connection error: • Partner rejected frame at end of connection with NAK or a random string (except for DLE) or • Acknowledgment characters (DLE) received too early.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester in the transmission line to check the status.
0707	Only for 3964(R): Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send frame: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device faulty or too slow. You can interconnect an interface tester in the transmission line to check the status.
0708	For ASCII driver only: The wait time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
0709	Only for 3964(R): Connection attempt not possible. Number of permitted connection attempts exceeded.	Check the interface cable or the transmission parameters. Also check that receive function between CPU and CP is correctly configured at the partner device.
070A	Only for 3964(R): The data could not be transmitted. The permitted number of transfer attempts was exceeded.	Check the interface cable or the transmission parameters.
070B	Only for 3964(R): Initialization conflict cannot be resolved because both partners have high priority.	Change the parameter assignment.
070C	Only for 3964(R): Initialization conflict cannot be resolved because both partners have low priority.	Change the parameter assignment.
0801	Only for 3964(R): Expectation of the first repetition: An error was detected on receiving a frame and the CP requested repetition from the partner via a negative acknowledgment (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.

Error code(W#16#...)	Description	Remedy
0802	Only for 3964(R): Error establishing connection: • In idle mode, one or more random codes (other than NAK or STX) were received or • After an STX was received, the partner sent more characters without waiting for the response DLE. After partner power ON: • While partner is being switched on, the CP receives an undefined character.	You can interconnect an interface tester in the transmission line to check the error response of the partner device.
0805	Only for 3964(R): Logical error while receiving: After DLE was received, a further random character (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the frame header and data string or the connection is terminated with DLE ETX. You can interconnect an interface tester in the transmission line to check the error response of the partner device.
0806	Character delay time (ZVZ) exceeded: • Two successive characters were not received within character delay time or Only for 3964(R): • First character after sending of DLE during connection setup was not received within character delay time.	Partner device faulty or too slow. You can interconnect an interface tester in the transmission line to check the status.
0807	Illegal frame length: A zero-length frame has been received.	Receipt of a zero-length frame is not an error. Check why the communication partner is sending message frames without user data.
0808	Only for 3964(R): Error in block check character (BCC): The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. You can interconnect an interface tester in the transmission line to check the error response of the partner device.
0809	Only for 3964(R): Waiting time for block repetition has expired	Declare a block waiting time at the communication partner identical to that of the CP 440. Check for malfunctioning of the communication partner, possibly by using an interface tester interconnected in the data link.
080A	There is no free receive buffer available: No receive buffer space available for receiving data.	Increase the call rate for the RECV_440 instruction.

Error code(W#16#...)	Description	Remedy
080C	Transmission error: • A transmission error (parity error, stop bit error, overflow error) was detected. Only for 3964(R): • If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. Only for 3964(R): • If this happens during send or receive operations, repetition is started.	Disturbances on the data link cause frame repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits.
080D	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
080E	Receive buffer overflow when flow control not enabled.	Increase the call rate for the receive instruction in the user program or configure the parameters for communication with flow control.
0810	Parity error	Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits.
0811	Character frame error	Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0812	For ASCII driver only: More characters were received after the CP had sent XOFF or set CTS to OFF.	Reassign the parameters for the communication partner or deal with the data in the CP more quickly.
0814	For ASCII driver only: A message frame or several message frames have got lost, because you have been working without flow control.	Work with flow control as much as you can. Use the entire receive buffer. Set the "Reaction to CPU Stop" parameter in the basic parameters to "Continue".
0816	The length of a received message frame was longer than the maximum specified length.	Correction required at the partner
0820	Static error when calling RD_REC SFC. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Analyze the SFCERR variable from the instance DB.
0B01	Receive buffer more than 2/3 full	Call the receive block more often to avoid the receive buffer overflowing.
8000	Module firmware not found	Perform a firmware update

Error code(W#16#...)	Description	Remedy
1E0D	• Job canceled due to a cold restart, warm restart, reset or	• Repeat the request.
	• Module firmware does not exist	• Check if the module is inserted.
1E0E	Static error when calling RD_REC SFC. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
1E0F	Static error when calling WR_REC SFC. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
1E41	Invalid number of bytes specified at instruction parameter LEN.	Keep to the value range 1 to 400 bytes.

Note

An error message is only output if the ERROR bit (job canceled with error) is also set. In all other cases, the STATUS word is zero.

Calling the SFCERR Variable

You can obtain more information on errors 14 (1E0EH) and 15 (1E0FH) in event class 30 by means of the SFCERR variable.

You can load the SFCERR variable from the instance DB of the corresponding instruction.

Error messages entered in the SFCERR variable can be found in the system functions SFC 58 under "WR_REC: Write data record to I/O" and SFC 59 under "RD_REC: Read data record from I/O".

See also

WR_REC: Write data record to I/O (Page 5260)

RD_REC: Read data record from I/O (Page 5256)

PtP data link CP 441 (S7-300, S7-400)

Communication instructions (S7-300, S7-400)

Introduction

Instead of using its own blocks, the CP 441 communicates by means of instructions of the S7-400 automation system

The S7-400 automation system provides a number of instructions which can be called in the user program to initiate and control communication between the CPU and CP 441 communication processor. The instructions are saved to non-volatile memory of the CPU.

Instructions from the automation system S7-400

The table below lists the instructions that the S7-400 automation system provides for communication between the CPU and CP 441.

Table 4-340 Instructions from the automation system S7-400

SFB	Meaning
BSEND	With the instruction BSEND, you can send data from a S7-data range to a communication partner with a destination.
BRCV	With the instruction BRCV, you can receive the data from a communication partner and transfer it to a S7-data range.
FETCH	RK 512 only: With the instruction FETCH, you can fetch the data from a communication partner.
PUT	RK 512 only: With the instruction PUT, you can send the data to the communication partner with dynamic destination addresses..
PRINT	With the instruction PRINT, you can print a message text on a printer with up to four variables.
STATUS	With the instruction STATUS, you can call the status of a communication connection.

Further Information

A detailed description of the instructions from the automation system S7-400 and their settings can be found under "Overview of instructions for S7 communication".

See also

Overview of the S7 communication instructions (Page 6131)

V24_STAT_441: Reading accompanying signals from the RS232C interface (S7-300, S7-400)

Description

The V24_STAT_441 instruction reads the RS 232C accompanying signals from an interface of the CP 441 and makes them available to you at the block parameters. The V24_STAT_441 instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The RS 232C accompanying signals are updated each time the instruction is called (cyclic polling). At the V24_STAT_441 instruction, select the interface by setting an ID that represents the "local ID" of one of the connections operating on this interface.

The binary result BR is not affected.

Note

A positive voltage at the RS232 C input signals DSR, CTS, DCD and RI is mapped accordingly to signal status "1" of the input signals DSR_IN, CTS_IN, DCD_IN and RI_IN of the V24_STAT_441 instruction.

Parameters

Parameters	Declaration	Data type	Description
REQ	VAR_INPUT	BOOL	Enables transmission at the rising edge
ID	VAR_INPUT	WORD	Clear communication connection to a communication partner
NDR	VAR_OUTPUT	BOOL	Rising edge indicates that new receive data is available to the user program
ERROR	VAR_OUTPUT	BOOL	Rising edge indicates errors
STATUS	VAR_OUTPUT	WORD	Contains detailed error information or a warning
DTR_OUT	OUTPUT	BOOL	**D**ata **t**erminal **r**eady, CP 441 ready (CP 441 output)
DSR_IN	OUTPUT	BOOL	**D**ata **s**et **r**eady, Communication partner ready (CP 441 input)
RTS_OUT	OUTPUT	BOOL	**R**equest **t**o **s**end, CP 441 ready to send[1] (CP 441 output)
CTS_IN	OUTPUT	BOOL	**C**lear **t**o **s**end, Communication partner is ready to receive data from CP 441 (response to RTS = ON of the CP 441) (CP 441 input)
DCD_IN	OUTPUT	BOOL	**D**ata **C**arrier **d**etect, receive signal level (CP 441 input)
RI_IN	OUTPUT	BOOL	**R**ing **I**ndicator, Indication of incoming call (CP 441 input)

Example

Table 4-341 Example of the call of the V24_STAT_441 instruction

STL

```
        CALL FB 5, DB55
                 REQ                 := DB30.DBX10.0
```

```
STL
                    ID                  :=W#16#1000
                    NDR                 := DB30.DBX10.1
                    ERROR               := DB30.DBX10.2
                    STATUS              := DB30.DBW20
                    DTR_OUT             := DB30.DBX30.0
                    DSR_IN              := DB30.DBX30.1
                    RTS_OUT             := DB30.DBX30.2
                    CTS_IN              := DB30.DBX30.3
                    DCD_IN              := DB30.DBX30.4
                    RI_IN               := DB30.DBX30.5
```

V24_SET_441: Writing accompanying signals to the RS232C interface (S7-300, S7-400)

Description

You can set or reset the interface outputs of CP 441 via the parameter inputs of the V24_SET_441 instruction. The V24_SET_441 instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

At the V24_SET_441 instruction, select the interface by setting an ID that represents the "local ID" of one of the connections operating on this interface.

The binary result BR is not affected.

Parameters

Parameters	Declaration	Data type	Description
REQ	VAR_INPUT	BOOL	Enables transmission at the rising edge
ID	VAR_INPUT	WORD	Clear communication connection to a communication partner
DONE	VAR_OUTPUT	BOOL	Signals error-free completion of the job at a rising edge
ERROR	VAR_OUTPUT	BOOL	Rising edge indicates errors
STATUS	VAR_OUTPUT	WORD	Contains detailed error information or a warning
RTS	INPUT	BOOL	Request to send, CP 441 ready to send (Control CP 441 output)
DTR	INPUT	BOOL	Data terminal ready, CP 441 ready (Control CP 441 output)

Example

Table 4-342 Example of the call of the V24_SET_441 instruction

```
STL
        CALL FB 6, DB56
                 REQ         := DB40.DBX10.0
                 ID          :=W#16#1000
                 DONE        := DB40.DBX10.1
                 ERROR       := DB40.DBX10.2
                 STATUS      := DB40.DBW20
                 RTS         := DB40.DBX30.2
                 DTR         := DB40.DBX30.0
```

SYSTAT error message area (S7-300, S7-400)

Error-Signaling Area SYSTAT

The error-signaling area SYSTAT is an error area which is available for every interface (ID number). The SYSTAT records all errors/events which can occur during data transmission on an interface.

Note

Because the STATUS request is executed asynchronously to the rest of the requests running at an interface, an SFB with a specific R_ID cannot be assigned to the error messages. This means that although SYSTAT can display which errors have occurred at an interface, it cannot show which SFB call (R_ID number) triggered the error.

Errors/Events

The SYSTAT messages are entered in bytes 2 to 15 of the LOCAL parameter when the STATUS SFB is called. In addition to the error byte (byte 2), the first six errors/events are displayed. Error event 1 is the oldest.

If other error events occur, these cannot be reported until the "old" entries are deleted. The error-signaling area must therefore be deleted in good time. This is done when the STATUS SFB is called.

The errors/events are stored as follows:

- Byte 0 — Operating state of CP (02H for RUN, 05H for defective)
- Byte 1 — Reserved
- Byte 2 — Bit 0 -F Enter error in SYSTAT
  Bit 1 -U Error overflow
  Bit 2 -B Break
- Byte 3 — Reserved
- Byte 4/5 — Event 1

- Byte 6/7 Event 2
- Byte 8/9 Event 3
- Byte 10/11 Event 4
- Byte 12/13 Event 5
- Byte 14/15 Event 6

Event classes

Event class	Meaning
1	CP hardware fault
2	Error initializing
3	Incorrect parameter assignment for the instruction
4	The CP detected errors in CP - CPU data traffic
5	Error executing a CPU job
6	Error executing a partner job (for RK 512 only)
7	Send errors
8	Receive errors
9	Received response message frame with error or error message frame from the interconnection partner
10 ($0A_H$)	The CP detected errors in the response message frame of the partner

Event numbers

Error code(W#16 #...)	Description	Remedy
0101	Fault while testing operating system EPROM of CP	CP defective; replace CP.
0102	RAM test of CP errored	
0103	Request interface of CP defective	
0104	No interface submodule inserted	Insert suitable interface submodule for CP.
0105	• Parameter memory defective • Interface submodule unplugged after parameter assignment	Exchange CP or insert suitable interface submodule for CP.
0110	Fault in CP firmware	Switch module off and on again. If necessary, replace module.
0201	No parameters Parameter memory empty or has unknown contents	Load interface parameters.
0208	Parameter assignment and interface submodule incompatible	Check parameters set for interface submodule.
020F	Invalid parameter assignment detected at start of parameter communication. Interface could not be parametered.	Correct invalid parameters and restart.

Error code(W#16 #...)	Description	Remedy
0210	Total baud rate exceeded	Reduce the baud rates of the two interfaces so that the total baud rate is not exceeded.
0211	Total baud rate back in range	Total baud rate was out of range. It is now back in range after the excursion.
0301	Invalid or no source/destination data type Invalid area (start address, length) Invalid or no DB (e.g. DB 0) or Other data type invalid or missing	Check parameters on CPU and CP and correct if necessary. **RK 512 only:** Partner returns invalid parameters in telegram header. Check parameters on CPU and CP; possibly create block. See request tables for valid data types. **RK 512 only:** Partner provides incorrect parameters in the message frame header.
0302	Area too short	Check parameters on CPU and CP; possibly check block/area. **RK 512 only:**Partner provides incorrect parameters in the message frame header.
0303	Area cannot be accessed	Check parameters on CPU and CP. Obtain the permissible start addresses and lengths from the request tables. **RK 512 only:** Partner provides incorrect parameters in the message frame header.
0401	CP cannot accept requests (overload)	In your user program, reduce number of requests called concurrently for CP.
0402	CP cannot process request type	Check if the system function blocks you have called in user program are valid for CP.
0403	Incorrect, unknown or illegal data type	Check program, e.g. for incorrect parameters of SFB.
0407	An error has occurred during data transmission between the CPU and the CP. Data cannot be received because there is no access to the CPU destination frame or the CPU destination frame does not exist or is too short. Writing data to the CPU destination frame or reading data from the CPU source frame during a CPU stop when assigning parameters is not permitted.	Check the destination frame on the CPU. Check the parameter assignment for "Response to a CPU Stop".
0408	• **Only for 3964R and ASCII drivers** • A temporary error has occurred during data transmission between the CPU and the CP (receive). The request is queued for repetition because the CPU is temporarily overloaded or the receive block (BRCV) is requested too infrequently or the receive block has been temporarily blocked.	• Reduce the number of communication requests. Call the receive block more often. Check if the receive block has been blocked too long.

Error code(W#16 #...)	Description	Remedy
0409	• **Only for 3964R and ASCII drivers** • An error has occurred during data transmission between the CPU and the CP. Data reception is not possible. After multiple attempts (see (04)07H), the request has canceled after 10 seconds because the receive block (BRCV) could not be called or is blocked.	• Check if the receive block has been called or has been blocked.
040B	Error during data transmission between CPU and CP, because • no connection has been configured • no receipt possible via configured connection	• Configure the connection in "NetPro" • Enter in "NetPro" under "Object Properties Connection) as communication direction: – 2: Partner → local or – 3: Local ↔ partner
0501	Current request aborted as a result of CP restart.	No remedy is possible at POWER ON. When changing the parameters of the CP in the programming device, before writing an interface you should ensure there are no more requests running from the CPU.
0502	Job not permitted in this CP operating mode (e.g., device interface not parameterized)	Set the parameters for the device interface.
0503	Wrong time, incorrect format	Check the time parameters.
0505	**Only for printer drivers:** System data block with message texts not available on the CP	Use the parameter assignment software to configure the message text and then carry out a restart.
0506	**Only for printer drivers:** Message text not available	Use the parameter assignment software to configure the message texts and then carry out a warm restart.
0507	**Only for printer drivers:** Message text too long	Edit the message text to reduce it to a length of fewer than 150 characters (or no more than 250 characters if it contains variables).
0508	**Only for printer drivers:** Too many conversion instructions	You have configured more conversion instructions than variables. Conversion instructions without associated variables will be ignored.
0509	**Only for printer drivers:** Too many variables	You have configured more variables than conversion instructions. Variables without associated conversion instructions will not be output.
050A	**Only for printer drivers:** Unknown conversion instruction	Check the conversion instruction. Undefined or unsupported conversion instructions are replaced in the printout with ******.

Error code(W#16 #...)	Description	Remedy
050B	**Only for printer drivers:** Unknown control instruction	Check the control instruction. Undefined or unsupported control instructions will be ignored. The control instruction will not be output as text either.
050C	**Only for printer drivers:** Conversion instruction not executable	Check the conversion instruction. Conversion instructions that cannot be executed are output in the expression in accordance with the defined width and the valid remainder of the conversion instruction or the standard representation with * characters.
050D	**Only for printer drivers:** Width in conversion instruction too small or too great	Correct the specified width of the variable in the conversion instruction on the basis of the variable's maximum possible number of characters in text-based display types (A, C, D, S, T, Y, Z). Only as many characters as will fit in the specified width appear in the printout; the text is truncated to this width. In all other cases, * characters are output corresponding to the width.
050E	**Only for ASCII drivers:** An error occurred while sending. The defined end-of-text characters did not occur within the maximum allowed length or in the case of automatic appending, the maximum allowed transmission length was exceeded.	Extend the end-of-text characters in the transmission buffer at the desired point or select a shorter telegram length for automatic appending.
050F	Number of requests that can be processed simultaneously too great	Change your STEP 7 program so that fewer requests can run simultaneously.
0510	Area occupied (resource)	Repeat the request.
0511	Length not permissible for this request type	Divide up the data to be transmitted into several requests.
0512	**RK 512 only:**Mismatch between SFB's source and destination parameters.	Obtain the permissible values from the request tables.
0513	Data type error (DB ...): Unknown or impermissible data type (e.g. DE) **RK 512 only:** Mismatch between SFB's source and destination data types.	Obtain the permissible data types and their combinations from the request tables.
0514	Specified start addresses too high for desired data type or start address or DB/DX number too low.	Obtain from the request tables the permissible start addresses and DB/DX numbers that can be specified in the program.
0515	**RK 512 only:** Wrong bit number specified for coordination flag.	Permissible bit numbers: 0 to 7
0516	**RK 512 only:** Specified CPU too high.	Permissible CPU numbers: none, 1, 2, 3 or 4

Error code(W#16 #...)	Description	Remedy
0517	An error occurred while receiving. The receive telegram is longer than 4 KB or is longer than the defined "fixed receive length" or the receive telegram does not fit into the destination frame.	Reduce the length of your connection partner's telegram or increase the length of your receive DB.
0518	Transmission length at sending too great (>4 KB)	RK 512 only: Obtain the permissible lengths from the request tables. Split the request up into several shorter requests.
0519	CP in wrong mode for PLC request	Check if the addressed interface is parametered.
051A	**RK 512 only:** Error sending a command telegram An associated procedure number has just been entered in STATUS.	See the remedy for the previous error number.
051B	**Only for printer drivers:** Precision invalid	Correct the specified precision in the conversion instruction. The precision is always initialized with a dot prefix to identify and limit the width (for example, ".2" to output the decimal point and two decimals.) Precision is only relevant to display types A, D, F and R. It is ignored otherwise.
051C	**Only for printer drivers:** Variable invalid (variable length incorrect/incorrect type)	Correct the specified send variable.
0601	Error in 1st command byte (not 00 or FFH)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0602	Error in 3rd command byte (not A, 0 or FFH)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0603	Error in 3rd command byte in the case of continuation telegrams (command not as for 1st telegram)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0604	Error in 4th command byte (command letter incorrect)	Header layout error at partner or a command combination has been requested that is not permitted at the CP. Check the permissible commands. You can interconnect an interface tester in the transmission line to debug the partner device.
0605	Error in 4th command byte in the case of continuation telegrams (command not as for 1st telegram)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0606	Error in 5th command byte (DB number not permissible)	Obtain from the request tables the permissible DB numbers, start addresses or lengths.
0607	Error in 5th or 6th command byte (start address too high)	Obtain from the request tables the permissible DB numbers, start addresses or lengths.

Error code(W#16 #...)	Description	Remedy
0608	Error in 7th or 8th command byte (impermissible length)	Obtain from the request tables the permissible DB/DX numbers, start addresses or lengths.
0609	Error in 9th and 10th command byte (coordination flag for this data type impermissible or bit number too high)	Header layout error at partner. Find out from the request tables when a coordination flag is permitted.
060A	Error in 10th command byte (CPU number not permitted)	Header layout error at partner.
060B	SEND telegram was longer/shorter than expected (more/less data received than announced in telegram header).	Correction required at the partner
060C	GET command telegram received with user data	Correction required at the partner
060D	The CP received a telegram during an invalid operating mode. • Receive connection between CPU and CP not set up or not yet correctly set up • CP startup is not fully completed. • Parameters for the interface are currently being assigned	• Check if the addressed connection has been assigned the correct parameters. • This error message can occur only during CP startup. Repeat the request. • This is a temporary error. Repeat the request.
060E	Synchronous fault of partner • New (continuation) command telegram received before response telegram sent. • 1. 1st command telegram expected and continuation telegram came. • Continuation command telegram expected and 1st telegram came	This error may be reported after your own programming device is restarted in the case of long telegrams or when the partner is restarted. These cases represent normal system start-up characteristics. The error can also occur during operation as a consequence of error statuses only recognized by the partner. Otherwise, you have to assume an error on the part of the partner device. The error may not occur in the case of requests <128 bytes.
060F	DB locked by coordination function	In local program: After processing of the last transmission data, enable the last receive block with "EN". In partner program: Repeat the request
0610	Telegram received too short (length <4 bytes in the case of continuation or response telegrams or <10 bytes in the case of command telegrams)	You can interconnect an interface tester in the transmission line to debug the partner device.
0611	Telegram length and length specified in telegram header are not the same.	You can interconnect an interface tester in the transmission line to debug the partner device.
0612	Error sending the (continuation) response telegram. An associated procedure error number has been entered in STATUS immediately beforehand.	See remedy for the error number entered immediately beforehand in STATUS.

Error code(W#16 #...)	Description	Remedy
0701	Sending of the first repetition: • An error was detected when transmitting the telegram or • The partner requested a repetition by means of a negative acknowledgment character (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0702	Error establishing connection: • After STX was sent, NAK or another code (except DLE or STX) was received or • The response came too early or • An initialization conflict occurred	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0703	Acknowledgment delay time (QVZ) exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive or there is a break in the transmission line, for example. You can interconnect an interface tester in the transmission line to debug the partner device.
0704	Termination by partner: One or more codes were received from the partner during sending	Check if the partner also indicates an error; possibly it has not received all of the transmitted data (for example, due to an interrupted data link) or because fatal errors are pending or the characteristics of the partner device is faulty. If necessary, use an interface test device switched into the transmission line to check.
0705	Negative acknowledgment when sending	Check if the partner also indicates an error; possibly it has not received all of the transmitted data (for example, due to an interrupted data link) or because fatal errors are pending or the characteristics of the partner device is faulty. If necessary, use an interface test device switched into the transmission line to check.
0706	End-of-connection error: • Partner rejected telegram at end of connection with NAK or a random string (except for DLE) or • Acknowledgment characters (DLE) received too early.	Check if the partner also indicates an error; possibly it has not received all of the transmitted data (for example, due to an interrupted data link) or because fatal errors are pending or the characteristics of the partner device is faulty. If necessary, use an interface test device switched into the transmission line to check.
0707	Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send telegram: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device too slow or faulty. If necessary, use an interface test device switched into the transmission line to check.

Error code(W#16 #...)	Description	Remedy
0708	ASCII Driver and printer driver only: The wait time for XON or CTS = ON has expired.	The communication partner is faulty, too slow or has been taken offline. Check the communication partner; you may need to change the parameter assignment.
0709	Connection setup not possible. Number of permitted setup attempts exceeded.	Check the interface cable or the transmission parameters.
070A	The data could not be transmitted. The permitted number of transfer attempts was exceeded.	Check the interface cable or the transmission parameters.
0801	Expectation of the first repetition: An error was detected on receiving a telegram and the CP requested repetition from the partner via a negative acknowledgment (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0802	**Error establishing connection:** • In idle mode, one or more random codes (other than NAK or STX) were received or • After an STX was received, the partner sent more characters without waiting for the response DLE. **After POWER ON of the partner:** • While partner is being switched on, the CP receives an undefined character.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0805	Logical error while receiving: After DLE was received, a further random code (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the telegram header and data string or the connection is terminated with DLE ETX. Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0806	Character delay time (ZVZ) exceeded: • Two successive characters were not received within character delay time or • 1. character after sending of DLE while establishing connection was not received within the character delay time.	Partner device too slow or faulty. Interconnect an interface tester in the transmission line to see if this is the case.
0808	Block check character (BCC) error (only in the case of RK 512 with the 3964R procedure and the 3964R procedure) The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check if there is a serious problem with the connection. In this case, error codes of the event class 8/event number 12 sometimes occur. Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.

Error code(W#16 #...)	Description	Remedy
080A	There is no free receive buffer available: After receipt of STX, there was no empty receive buffer available to the procedure at connection setup and after an additional waiting time	Increase the call rate for the receive instruction in the user program.
080C	Transmission error: A transmission error (parity error, stop bit error, overflow error) was detected. If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. **Only in the case of RK 512 and 3964(R):** If this occurs during send or receive operation, repetitions are initiated.	Disturbances on the data link cause telegram repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check that the settings for baud rate, parity and number of stop bits are the same on both devices.
080D	BREAK The connection line (receive line) to the partner device is interrupted	Establish the connection between the devices or switch on the partner device. In the case of TTY, Check if there is a current loop in the idle state.
0812	**Only for ASCII drivers:** More characters were received after the CP had sent XOFF or set CTS to OFF.	Reset the parameters for the communication partner or read data from CP more quickly.
0815	Discrepancy between settings for transfer attempts at CP and communication partner.	Set the same number of transfer attempts at communications partner as at CP. You can interconnect an interface tester in the transmission line to debug the partner device.
0816	• The length of a received telegram was longer than the length agreed upon or • the length of the parametered receive buffer (with CP 441 only) is too short.	• A correction is necessary at the partner or • the length of the receive buffer (with CP 441 only) must be enlargened
0818	**Only for ASCII drivers:** DSR = OFF or CTS = OFF	The partner has switched the DSR or CTS signal to "OFF" before or during a transmission. Check the partner's control of the RS 232C accompanying signals.
0902	**RK 512 only:** Memory access error at partner (memory does not exist) With SIMATIC S5 as partner: • Incorrect area at status word or • Data area does not exist (except DB/DX) or • Data area too short (except DB/DX)	Check that the partner has the desired data area and that it is big enough or check the parameters of the called system function block. Check the specified length at the system function block.

Error code(W#16 #...)	Description	Remedy
0903	**RK 512 only:** DB/DX access error at the partner (DB/DX does not exist or is too short) With SIMATIC S5 as partner: • DB/DX does not exist or • DB/DX too short or • DB/DX number impermissible Permissible source area for GET request exceeded	Check that the partner has the desired data area and that it is big enough or check the parameters of the called system function block. Check the specified length at the system function block.
0904	**RK 512 only:** Partner returns "Request type not permitted".	Partner malfunction, because a system command is never issued from the CP 441.
0905	**RK 512 only:** Error at partner or at SIMATIC S5 as partner: • Source/destination type not permissible or • Memory error in partner programmable controller or • Error notifying CP/CPU at the partner or • Partner programmable controller is in STOP state	Check if the partner can transmit the desired data type. Check the structure of the hardware at the partner. Set the partner programmable controller to RUN.
0908	**RK 512 only:** Partner detecting synchronization error: Telegram sequence error.	This error occurs at restart of your own programmable controller or of the partner. This represents normal system start-up characteristics. You do not need to correct anything. The error is also conceivable during operation as a consequence of previous errors. Otherwise, you can assume an error on the part of the partner device.
0909	**RK 512 only:** DB/DX disabled at partner by coordination flag	In partner program: After processing of the last transmission data, reset the coordination flag. In own program: Repeat the request.
090A	**RK 512 only:** Error detected by partner in telegram header: 3rd command byte in header is incorrect	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090B	**RK 512 only:** Error in telegram header: 1. or incorrect command byte 4 in header	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090C	**RK 512 only:** Partner detects incorrect telegram length (total length).	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090D	**RK 512 only:** Partner has not yet restarted.	Restart the partner programmable controller or set the mode selector on the CP or CPU to RUN.

Error code(W#16 #...)	Description	Remedy
090E	**RK 512 only:** Unknown error number received in response telegram.	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
0A01	**RK 512 only:** Synchronization error of partner, because: • Response telegram without request • Response telegram received before continuation telegram sent • Continuation response telegram received after an initial telegram was sent • A first response telegram was received after a continuation telegram was sent	This error is reported after your own programming device is restarted in the case of long telegrams or when the partner is restarted. This represents normal system start-up characteristics. You do not have to correct anything. The error can also occur during operation as a consequence of error statuses only recognized by the partner. Otherwise, you can assume an error on the part of the partner device. The error may not occur in the case of requests <128 bytes.
0A02	**RK 512 only:** Error in the structure of the received response telegram (1st byte not 00 or FF)	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A03	**RK 512 only:** Received response telegram has too many data or not enough data.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A04	**RK 512 only:** Response telegram for SEND request arrived with data.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A05	**RK 512 only:** No response telegram from partner within monitoring time.	Is the partner a slow device? This error is also often displayed as a consequence of a previous error. For example, procedure receive errors (event class 8) can be displayed after a GET telegram was sent. Reason: As a result of disturbances, the response telegram could not be received and the monitoring time elapsed. This error can also occur if a restart was carried out at the partner before it could respond to the most recently received GET telegram.
0A06	**RK 512 only:** Received response telegram after GET request has not enough data.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.

Modbus slave (RTU) (S7-300, S7-400)

Modbus instructions (S7-300, S7-400)

Introduction

The Modbus masters do not need separate Modbus instructions for communication.

Instructions of CP 341

The data transfer between CP 341 and the CPU is handled using the instructions P_SND_RK and P_RCV_RK.

Table 4-343 Instructions of CP 341

Instruction	Meaning
P_RCV_RK	Using the P_RCV_RK instruction, you can receive data from a communication partner and save the data to a data block or provide data for the communication partner.
P_SND_RK	Using the P_SND_RK instruction, you can send the entire area or a section of a data block to a communication partner or fetch data from a communication partner.

Instructions from the automation system S7-400

The data transfer between CP 441 and the CPU is handled using the instructions BSEND and BRCV.

Table 4-344 Instructions from the automation system S7-400

SFB	Meaning
BSEND	With the instruction BSEND, you can send data from a S7-data range to a communication partner with a destination.
BRCV	With the instruction BRCV, you can receive the data from a communication partner and transfer it to a S7-data range.

MODB_341: Modbus slave instruction for CP 341 (S7-300, S7-400)

Description

The MODB_341 instruction and associated driver enable a communication connection between a Modbus master control system and the CP 341 communication module as "Modbus-capable" slave system.

Operating principle

In the user program, the MODB_341 instruction is is called in the cyclic routine. The instruction uses the associated IDB_MODB_341 instance DB as the working area.

The MODB_341 instruction initializes the CP and executes the Modbus functions which the driver cannot execute by itself.

The MODB_341 instruction must be performed following every cold or warm restart of the CPU. Initialization is triggered by a rising edge at the input CP_START. The instruction deletes the instance DB, calls system instruction SZL_LESEN to read the I, Q, F, T and C address areas from the CPU and saves these to the instance DB. This procedure allows you to check whether the write requests from the Modbus master system are out of range.

A SEND job reports the number of the instance DB and the currently successful initialization progress to the CP.

Once the SEND job is completed without errors, output CP_START_OK is set and initialization of the instruction is completed.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	Int	Base address of the CP Use HW Config assignment
START_TIMER	INPUT	Timer	Timer for start of monitoring time
START_TIME	INPUT	S5Time	Time value for start of monitoring time
OB_MASK	INPUT	BOOL	Mask I/O access errors, delay interrupts FALSE: I/O access errors are not masked. TRUE: I/O access errors are masked and interrupts are delayed.
CP_START	INPUT	BOOL	Start instruction initialization
CP_START_FM	INPUT	BOOL	Initialization is triggered with a rising edge at CP_START.
CP_START_NDR	OUTPUT	BOOL	Info: Write request from the CP
CP_START_OK	OUTPUT	BOOL	Startup completed without error TRUE: The initialization job could be completed without error before the monitoring time elapsed.
CP_START_ERROR	OUTPUT	BOOL	Startup canceled with error TRUE: The initialization job could not be completed without error even after the monitoring time had elapsed.
ERROR_NR	OUTPUT	Word	Error number Allocation, see ERROR_NR, ERROR_INFO parameters (Page 6445)
ERROR_INFO	OUTPUT	Word	Additional error info Allocation, see ERROR_NR, ERROR_INFO parameters (Page 6445)

How to Handle an Error

If the SEND job is terminated with errors, CP_START is reset and CP_START_ERROR is set.

If initialization ends with errors, Modbus communication is not possible. All requests from the Modbus master system are answered with an exception code frame.

After power on, the CP needs several seconds to complete the hardware and memory test before it is ready for operation. The initialization attempts carried out by the MODB_341 instruction within this period are terminated with error. For this reason, the instruction repeats its initialization job several times within the monitoring time.

If the initialization is successfully completed within the configured START-TIME of the START-TIMERS, CP_START_OK is set. If the initialization job cannot be completed without errors even after the monitoring time has elapsed, CP_START_ERROR is set.

MODB_441: Modbus slave instruction for CP 441 (S7-300, S7-400)

Description

The MODB_441 instruction and associated driver enable a communication connection between a Modbus master control system and the CP 441-2 communication module as "Modbus-capable" slave system.

Operating principle

In the user program, the MODB_441 instruction is is called in the cyclic routine. The instruction uses the associated IDB_MODB_441 instance DB as the working area.

The Modbus function codes 01, 02, 03, 04, 06 and 16 are processed directly by the CP. For function codes 05 and 15, the instruction handles the bit-by-bit entry of the data in the SIMATIC memory range.

The MODB_441 instruction initializes the CP and executes the Modbus functions which the driver cannot execute by itself.

The MODB_441 instruction must be performed following every cold or warm restart of the CPU. Initialization is triggered by a rising edge at the input CP_START. The instruction deletes the instance DB, calls system instruction SZL_LESEN to read the I, Q, F, T and C address areas from the CPU and saves these to the instance DB. This procedure allows you to check whether the write requests from the Modbus master system are out of range.

A SEND job reports the number of the instance DB and the currently successful initialization progress to the CP.

Once the SEND job is completed without errors, output CP_START_OK is set and initialization of the instruction is completed.

Parameters

Parameters	Declaration	Data type	Description
ID	INPUT	Word	Connection ID Connection ID from configuration of the data link.
START_TIMER	INPUT	Timer	Timer for start of monitoring time
START_TIME	INPUT	S5Time	Time value for start of monitoring time
STATUS_TIMER	INPUT	Timer	Read Timer for SYSTAT
STATUS_TIME	INPUT	S5Time	Read time interval for SYSTAT

Parameters	Declaration	Data type	Description
OB_MASK	INPUT	BOOL	Mask I/O access errors, delay interrupts FALSE: I/O access errors are not masked. TRUE: I/O access errors are masked and interrupts are delayed.
CP_START	INPUT	BOOL	Start instruction initialization
CP_START_FM	INPUT	BOOL	Initialization is triggered with a rising edge at CP_START.
CP_START_NDR	OUTPUT	BOOL	Info: Write request from the CP
CP_START_OK	OUTPUT	BOOL	Startup completed without error TRUE: The initialization job could be completed without error before the monitoring time elapsed.
CP_START_ERROR	OUTPUT	BOOL	Startup canceled with error TRUE: The initialization job could not be completed without error even after the monitoring time had elapsed.
ERROR_NR	OUTPUT	Word	Error number Allocation, see ERROR_NR, ERROR_INFO parameters (Page 6445)
ERROR_INFO	OUTPUT	Word	Additional error info Allocation, see ERROR_NR, ERROR_INFO parameters (Page 6445)

How to Handle an Error

If the SEND job is terminated with errors, CP_START is reset and CP_START_ERROR is set.

If initialization ends with errors, Modbus communication is not possible. All requests from the Modbus master system are answered with an exception code frame.

After power on, the CP needs several seconds to complete the hardware and memory test before it is ready for operation. The initialization attempts carried out by the MODB_441 instruction within this period are terminated with error. For this reason, the instruction repeats its initialization job several times within the monitoring time.

If the initialization is successfully completed within the configured START-TIME of the START-TIMERS, CP_START_OK is set. If the initialization job cannot be completed without errors even after the monitoring time has elapsed, CP_START_ERROR is set.

ERROR_NR, ERROR_INFO parameters (S7-300, S7-400)

Error classes

Error numbers from ... to	Description	Meaning
1 ... 9	Instruction and CP initialization error	Error numbers 1 to 9 indicate that initialization was terminated with errors. The CP_START_ERROR parameter is set to 1. Modbus communication to the master system is not possible.
10 ... 19	Error executing a function code	Error numbers 10 to 19 indicate that an error occurred during the processing of a function code. The CP transmitted an invalid execution job to the communication instruction. The error is also reported to the driver. Subsequent execution jobs continue to be processed.
90 ... 99	Other faults/errors	A processing error has occurred. The error is not reported to the driver. Subsequent execution jobs continue to be processed.

ERROR_NR, ERROR_INFO parameters

ERROR_ No (decimal)	ERROR_INFO	Error Text	Remedy
0	0	No error	
1	SFC 51->RET_VAL	Error reading the system status list with SFC 51.	Analyze RET_VAL in ERROR_INFO; eliminate the cause.
2		Timeout of CP initialization or error during CP initialization	Check if the CP interface was assigned the "Modbus slave" protocol in the parameters. Analyze ERROR_INFO.
	P_SND_RK -> STATUS	Error in P_SND_RK job	Check module address LADDR at the communication instruction.
	SFB 12->STATUS	Error in BSEND job	Check the connection ID in the communication instruction.
10	Processing Code	The driver transferred an invalid processing function to the communication instruction.	CP restart (Power_On)
11	Start address	The driver transferred an invalid start address to the communication instruction.	Check Modbus address of Modbus master system.
12	Register number	The driver transferred an invalid number of registers to the communications instruction: Number of registers = 0.	Check number of registers of Modbus master system, if required restart CP (Power_ON)

ERROR_ No (decimal)	ERROR_INFO	Error Text	Remedy
13	Register number	The driver transferred an invalid number of registers to the communications instruction: Number of registers > 128.	Check number of registers of Modbus master system, if required restart CP (Power_ON)
14	Memory bits M - End Address	Attempted access to SIMATIC memory area "memory bits" in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
15	Outputs Q – End Address	Attempted access to SIMATIC memory area "outputs" in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
16	Timers T – End Address	Attempted access to SIMATIC memory area "timers" in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
17	Counters C – End Address	Attempted access to SIMATIC memory area "counters" in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
18	0	The driver transferred an invalid SIMATIC memory area to the communication instruction.	If required, restart CP (Power_On)
19		Error during access to SIMATIC I/Os.	Check if the required I/Os exist and are error-free.
90	P_SND_RK -> STATUS	Error transmitting an acknowledgment message to the driver using the P_SND_RK instruction	Analyze the STATUS information
	SFB 12->STATUS	Error transmitting an acknowledgment message to the driver using SFB 12 BSEND	Analyze the STATUS information
91	SFB 22->STATUS	Error reading SYSTAT with SFB 22 (STATUS)	Analyze the STATUS information
92	P_RCV_RK -> STATUS	Error executing a RECEIVE/FETCH call using the P_RCV_RK instruction	Analyze P_RCV_RK-STATUS

SYSTAT error message area (S7-300, S7-400)

Event classes

Event class	Meaning
5	Error executing a CPU job
8	Receive errors
14 ($0E_H$)	General processing error

STATUS parameter

Error code(W#16#...)	Description	Remedy
0518	Transmission frame length exceeded (> 4 KB) or insufficient transmission frame length for SEND.	Check the communication instruction; reload it if necessary.
0806	Character delay time ZVZ exceeded	Error by the partner device or repair disruptions on the transmission line.
080C	A character transmission error was detected (parity error, overrun error or stop bit error (frame)).	Check if the transmission lines are affected by interference. Modify the system configuration or the cable routing. Check the consistency of the settings for the protocol parameters baud rate, number of data bits, parity and number of stop bits at the CP and link partner.
080D	BREAK The receive line to the partner device is disrupted.	Establish the connection between the devices or switch on the partner device. In TTY mode, check if there is a line current flow in idle state. For an RS422/485(X27) connection, check or modify the default configuration of the 2-wire receive line R(A) and R(B).
0830	Broadcast is not allowed with this function code.	The MODBUS master system can use the broadcast function only for the function codes enabled for this purpose.
0831	Function code received is invalid.	This function code cannot be used with this driver.
0832	Maximum number of bits or registers exceeded or the number of bits cannot be divided by the value 16 for access to the SIMATIC memory area for timers and counters.	Limit the maximum number of bits to 2040 and of the registers to 127. Access to SIMATIC timers/counters only in 16-bit intervals only.
0833	The number of bits or registers for function code FC 15/16 and frame element byte_count do not match.	Adjust the number of bits, number of registers or the byte_count.
0834	Illegal bit coding for "Set bit / Reset bit" recognized.	Always use the codes 0000 hex or FF00 hex for FC 05.
0835	Invalid diagnostics subcode (≠ 0000 hex) detected for function code FC 08 "Loop back test".	Always use subcode 0000 hex for FC 08.
0836	The internally calculated value of the CRC 16 checksum does not match the CRC checksum received.	Check the calculation of the CRC checksum at the MODBUS master system.

Error code(W#16#...)	Description	Remedy
0837	Message frame sequence error: The MODBUS master system sent a new request message frame before the driver transmitted the last response message frame.	Increase the timeout for the slave response message frame at the MODBUS master system.
0E01	Error during initialization of the driver-specific SCC process.	Reassign parameters of driver and reload.
0E02	Error during startup of driver: Wrong SCC process active (SCC driver). The driver cannot function with this SCC driver.	Reassign parameters of driver and reload.
0E03	Driver startup error: Incorrect data transfer process is active (interface to the SFBs). The driver does not work with this data transfer process.	Reassign parameters of driver and reload.
0E04	Driver startup error: Invalid interface module. The driver cannot be operated with the configured interface module.	Check and correct the parameter assignment.
0E05	Error with driver dongle: No dongle plugged in, or inserted dongle is faulty. The driver is not ready to run.	Check if a driver dongle is plugged into the CP. If the inserted dongle is faulty, replace it with a correct dongle.
0E06	Error with driver dongle: The dongle has no valid content. The driver is not ready to run.	Obtain a correct dongle from the Siemens office which supplied you with the driver.
0E10	Internal error procedure: default branch in Send automatic device.	CP restart (Power_On)
0E11	Internal error procedure: default branch in Receive automatic device.	CP restart (Power_On)
0E12	Internal error active automatic device: Default branch	CP restart (Power_On)
0E13	Internal error passive automatic device: Default branch	CP restart (Power_On)
0E20	This connection requires 8 data bits. The driver is not ready for operation.	Correct the parameter assignment of the driver.
0E21	The parameterized multiplication factor for the character delay time is not within the range of 1 to 10. The driver is operating with default setting 1.	Correct the parameter assignment of the driver.
0E22	The operating mode set for the driver is illegal. Specify "Normal" or "Interference Suppression". The driver is not ready for operation.	Correct the parameter assignment of the driver.
0E23	Parameter assignment with invalid value for the slave address. Slave address 0 is invalid. The driver is not ready for operation.	Correct the parameter assignment of the driver.
0E24	Invalid limits were configured for write access. The driver is not ready.	Correct the parameter assignment of the driver.
0E25	An invalid "from/to" combination was configured in the specification of the "Conversion of the Modbus addresses with function code FC 01, 05, 15" area. (Memory bits, outputs, timers, counters area). The driver is not ready.	Correct the parameter assignment of the driver.

Error code(W#16#...)	Description	Remedy
0E26	An invalid "from/to" combination was configured in the specification of the "Conversion of the Modbus addresses with function code FC 02" area. (Memory bits, inputs area). The driver is not ready.	Correct the parameter assignment of the driver.
0E27	An overlap was configured in the specification of the "Conversion of the Modbus addresses with function code FC 01, 05, 15" area. (Memory bits, outputs, timers, counters area). The driver is not ready.	Correct the parameter assignment of the driver.
0E28	An overlaps was configured in the specification of the "Conversion of the Modbus addresses with function code FC 02" area. (Memory bits, inputs area). The driver is not ready.	Correct the parameter assignment of the driver.
0E2E	An error occurred when reading the interface parameter file. The driver is not ready to run.	Restart CP (Power_On).
0E30	Internal error during data transmission to CPU: Unexpected acknowledgment Passive.	Can be ignored if it happens intermittently.
0E31	Timeout during data transfer to the CPU.	Check CP-CPU interface.
0E32	Error occurred during data transmission to CPU with RCV: Exact failure reason (detailed error) is in SYSTAT before this entry.	Check CP-CPU interface.
0E33	Internal error during data transfer to the CPU: Illegal status of automatic device	Check CP-CPU interface.
:		
0E38	An error has occurred accessing SIMATIC areas "Memory bit, output, timers, counters, input" with the function code FC 01 or FC 02: For example, input not available or attempt to read beyond the end of the area.	Check whether the addressed SIMATIC area exists and that no attempt was made to access beyond the end of the area.
0E39	An error has occurred accessing the SIMATIC "Data block" area with the function code FC 03, 04, 06, 16: Data block not available or too short.	Check if the addressed data block exists and is sufficiently long.
0E3A	Error executing a write job with function codes FC 05, 15: Instance DB of the MODBUS instruction does not exist or is of insufficient length.	Check if the instance DB assigned in the parameters of the MODBUS communications instruction exists and whether it is of sufficient length.
0E3B	Timeout (waiting for acknowledgement) in the execution of a write job by the Modbus communication instruction.	Check the connection configuration and the CP-CPU interface (SFB SEND); if necessary, reload the MODBUS communication instruction.
0E3C	Illegal job with this driver.	Only SFB SEND, STATUS (CP 441-2 only) are permitted.
0E50	In the word-based SIMATIC timers/counters areas, the resulting residual bit number from the Modbus address is ≠ 0.	Only use MODBUS addresses which result in valid bit numbers.
0E51	The received Modbus address is outside the configured "from/to" areas.	Only use addresses as address information in the request message frame that have previously been defined in the configuration.

Error code(W#16#...)	Description	Remedy
0E52	• A SIMATIC area limit was exceeded during an access attempt by the Modbus master system: Resulting DB number < 1 • Write access to an unreleased area (configuration) • Write access to the instance DB of the communication instruction	Access area limited to valid SIMATIC memory areas.
0E53	A SIMATIC area limit was exceeded during an access attempt by the Modbus master system: For example, overflow from the formation of the resulting DB no. (> 65535)	Access area limited to valid SIMATIC memory areas.
0E54	Access beyond the configured area end or access beyond the SIMATIC area end.	Access area limited to valid SIMATIC memory areas.
0E55	Write access to this SIMATIC memory area is not allowed.	Write only to the SIMATIC memory bits, outputs data areas.
0E56	Data link operation not possible because the communication instruction is inactive.	Call the MODBUS communication instruction at cyclic intervals in the STEP 7 user program. If necessary, re-initialize the communication instruction.
0E57	An error has occurred executing the MODBUS function code in the communication instruction.	Analyze the exact cause.

ET 200S serial interface (S7-300, S7-400)

ET 200S 1SI for 3964(R) and ASCII (S7-300, S7-400)

S_RCV: receiving data (S7-300, S7-400)

Description

The S_RCV instruction sends data from the ET 200S 1SI module to an S7 data area as specified at the DB_NO and DBB_NO parameters. For data transmission, the S_RCV instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from the ET 200S 1SI module. An active transmission can be canceled by setting signal state "0" at parameter EN_R and with signal state "1" at parameter R. (Because transmission may sometimes not start up again after setting parameter EN_R to "1", you should set parameter R to "1" in addition to setting EN_R to "0" to cancel an active transmission.) The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R. Data transmission can take several calls (program cycles), depending on the data volume.

If the instruction detects signal state "1" at the R parameter, the current transmission job is canceled and the S_RCV instruction is reset to the initial state. Receiving is disabled as long as signal state "1" is set at parameter R. If the signal state returns to "0", the canceled message frame is received again from the beginning.

Parameter LADDR specifies the address of the ET 200S 1SI to be addressed.

Output NDR indicates "Job completed without errors/data received" (all data read) and ERROR indicates error events. A corresponding error number is displayed in STATUS if an error occurs. If the receive buffer is filled to more than 2/3 of its capacity, STATUS returns a warning at each call of S_RCV. If no errors or warnings are pending, STATUS has a value of "0".

NDR and ERROR/STATUS are also output after the reset of the S_RCV instruction (LEN == 16#00) (see time sequence chart). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enable data reading
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Receive data block no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number (received data) **S7-1500**: 0 ≤ DBB_NO and DBB_NO + LEN ≤ size of the DB **S7-300/400**: 0 ≤ DBB_NO and DBB_NO + LEN < 8190
NDR [1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6464) contains the error information.
LEN [1]	OUTPUT	INT	Length of the message frame received 1 ≤ LEN ≤ 224 specified in number of bytes
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6464) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a receive job, the NDR, ERROR, LEN and STATUS parameters remain available for the duration of **one** CPU cycle!			

Allocation in the data area

The S_RCV instruction works in combination with an I_RCV instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If the error STATUS == W#16#1Exx occurs, you can examine the SFCERR variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Startup

The COM_RST parameter of S_RCV instruction is used to report a startup to the instruction.

In the startup OB, set parameter COM_RST to 1.

Call the instruction in the cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the wiring of the EN_R and R inputs.

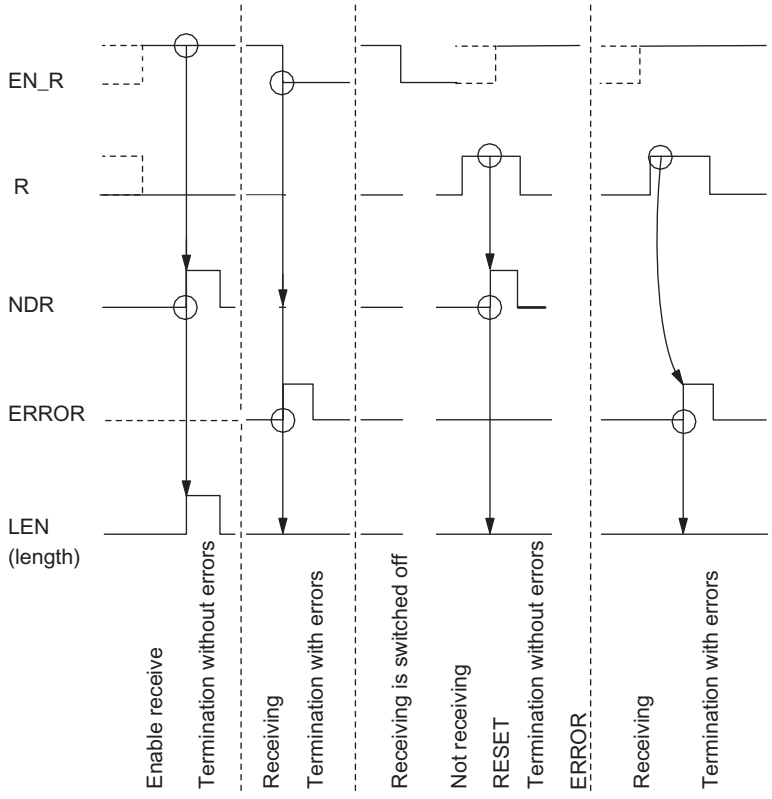

Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

Rules

Note

The S_RCV instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CPU startup routine of the ET 200S module for the S_RCV instruction must be completed before the ET 200S 1SI module can receive a job initiated after the status transition of the CPU from STOP to RUN.

S_SEND: Sending Data (S7-300, S7-400)

Description

The S_SEND instruction transmits a data block from a DB to the ET 200S 1SI as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the S_SEND instruction is called in a static (unconditional) operation in the cycle or **alternatively**, in a time-controlled program.

Operating principle

Data transmission is initiated by a positive edge at the REQ input. Data transmission can take several calls (program cycles), depending on the data volume.

The S_SEND instruction can be called in the cycle by setting signal state "1" at parameter input R. This setting cancels the transmission to the ET 200S 1SI module and resets the S_SEND instruction to the initial state. Data that has already been received by the ET 200S 1SI module is still transmitted to the communication partner. A static signal state "1" at the input R indicates that data transmission is disabled.

Parameter LADDR specifies the address of the ET 200S 1SI to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no errors occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for RESET of the S_SEND instruction (see the time sequence chart). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number (transmit data) **S7-1500**: 0 ≤ DBB_NO and DBB_NO + LEN ≤ size of the DB **S7-300/400**: 0 ≤ DBB_NO and DBB_NO + LEN < 8190
LEN	INPUT	INT	Data length 1 ≤ LEN ≤ 224 specified in number of bytes

Parameters	Declaration	Data type	Description
DONE [1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6464) contains the error information.
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6464) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a transmission job, the DONE, ERROR and STATUS parameters remain available for the duration of **one** CPU cycle!			

Allocation in the data area

The S_SEND instruction works in combination with an I_SEND instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If the error STATUS == W#16#1Exx occurs, you can examine the SFCERR variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Startup

The COM_RST parameter of S_SEND instruction is used to report a startup to the instruction.

In the startup OB, set parameter COM_RST to 1.

Call the instruction in the cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the wiring of the REQ and R inputs.

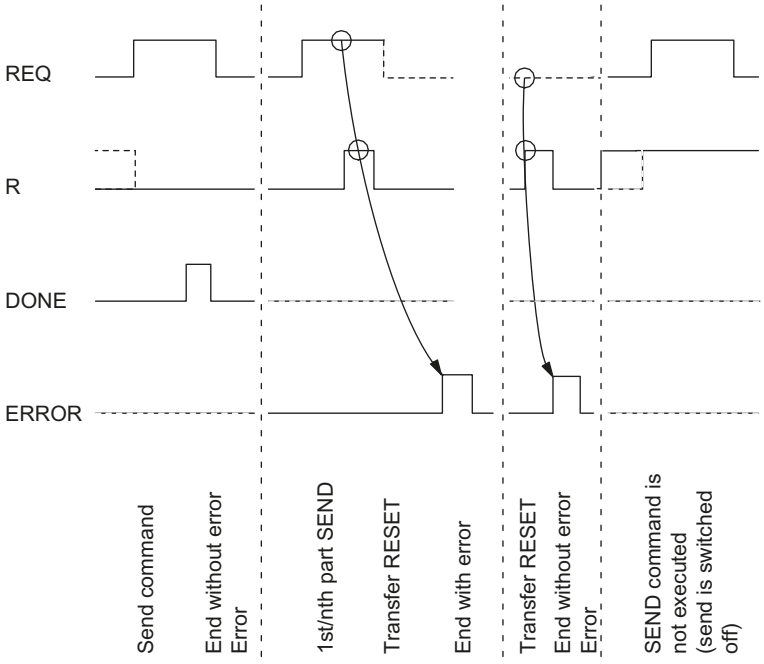

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rules

Note

The S_SEND instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CPU startup routine of the ET 200S module for the S_SEND instruction must be completed (see above) before the ET 200S 1SI module can execute a job initiated after the status transition of the CPU from STOP to RUN. A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the ET 200S 1SI module.

S_VSTAT: Read accompanying signals at the RS 232C interface (S7-300, S7-400)

Description

The S_VSTAT instruction reads the RS 232C accompanying signals of the ET 200S 1SI module and makes them available to you in the block parameters. For data transmission, the S_VSTAT instruction is called by means of (unconditional) static operation in the cycle or alternatively, in a time-controlled program.

Operating principle

The RS 232C accompanying signals are updated each time the instruction is called (cyclic polling).

The address of the ET 200S 1SI module to be addressed is specified in the LADDR parameter.

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
DONE[1]	OUTPUT	BOOL	Indicates that the instruction has terminated. (ET 200S 1SI output)
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6464) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6464) contains the error information.
DTR_OUT [1]	OUTPUT	BOOL	Data terminal ready, ET 200S 1SI ready for operation. (ET 200S 1SI output)
DSR_IN [1]	OUTPUT	BOOL	Data set ready; communication partner ready for operation. (ET 200S 1SI input)
RTS_OUT [1]	OUTPUT	BOOL	Request to send, ET 200S 1SI clear to send. (ET 200S 1SI output)
CTS_IN [1]	OUTPUT	BOOL	Clear to send; communication partner can receive data from the ET 200S 1SI module (response to RTS = ON of the ET 200S 1SI) (ET 200S 1SI input)

Parameters	Declaration	Data type	Description
DCD_IN [1]	OUTPUT	BOOL	Data carrier detect (ET 200S 1SI input)
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a job, these parameters remain available for the duration of **one** CPU cycle.			

Allocation in the data area

The S_VSTAT instruction works in combination with an I_STAT instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

A minimum pulse duration is necessary to detect a signal change. Determining factors are the CPU cycle time, the update time on the ET 200S 1SI module and the response time of the communication partner.

Startup

The COM_RST parameter of the S_VSTAT instruction is used to report a startup to the instruction.

Set the COM_RST parameter in the startup OB to 1.

Call the instruction in cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

S_VSET: Write accompanying signals at the RS 232C interface (S7-300, S7-400)

Description

You can set and reset the interface outputs using the corresponding parameter inputs of the S_VSET instruction. The S_VSET instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The address of the ET 200S 1SI module to be addressed is specified in the LADDR parameter.

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
RTS	INPUT	BOOL	Request to send, ET 200S 1SI clear to send. (Controls ET 200S 1SI output)
DTR	INPUT	BOOL	Data terminal ready, ET 200S 1SI ready for operation. (Controls ET 200S 1SI output)
DONE[1]	OUTPUT	BOOL	(ET 200S 1SI output)
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6464) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6464) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a job, these parameters remain available for the duration of **one** CPU cycle.			

Allocation in the data area

The S_VSET instruction works in combination with an I_SET instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Startup

The COM_RST parameter of the S_VSET instruction is used to report a startup to the instruction.

Set the COM_RST parameter in the startup OB to 1.

Call the instruction in cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

S_XON: Assign parameters for XON/XOFF data flow control (S7-300, S7-400)

Description

Using the S_XON instruction, you can set additional parameters if the module is configured for XON/XOFF data flow control.

Note

The instruction S_XON must be started before the instructions S_SEND, S_RCV, S_VSET and S_VSTAT and have completed its task (DONE = TRUE).

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
XON	INPUT	BYTE	XON character 0 to $7F_H$ (7 data bits) 0 to FF_H (8 data bits) Default: 11 (DC1)
XOFF	INPUT	BYTE	XOFF character 0 to $7F_H$ (7 data bits) 0 to FF_H (8 data bits) Default: 13 (DC3)
WAIT_FOR_XON	INPUT	TIME	Wait time for XON after XOFF 20 ms to 10 min 55 s 350 ms Default: 2 s
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6464) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6464) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart

[1] The DONE, ERROR and STATUS parameters are available for **one** CPU cycle following a successful request.

Allocation in the data area

The S_XON instruction works in combination with an I_XON instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for more precise error information.

Startup

The COM_RST parameter of the S_XON instruction is used to report a startup to the instruction.

Set the COM_RST parameter in the startup OB to 1.

Call the instruction in cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

S_RTS: Assign parameters for RTS/CTS data flow control (S7-300, S7-400)

Description

Using the S_RTS instruction, you can configure additional parameters if the module is configured for RTS/CTS data flow control.

Note

The instruction S_RTS must be started before the instructions S_SEND, S_RCV, S_VSET and S_VSTAT and have completed its task (DONE = TRUE).

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S SI module The start address is taken from STEP 7.

Parameters	Declaration	Data type	Description
WAIT_FOR_CTS	INPUT	TIME	Wait time for CTS = ON 20 ms to 10 min 55 s 350 ms Default: 2 s
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6464) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6464) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] The DONE, ERROR and STATUS parameters are available for **one** CPU cycle following a successful request.			

Allocation in the data area

The S_RTS instruction works in combination with an I_RTS instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for more precise error information.

Startup

The COM_RST parameter of the S_RTS instruction is used to report a startup to the instruction.

Set the COM_RST parameter in the startup OB to 1.

Call the instruction in cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

S_V24: Assign parameters for data flow control using automatic operation of the RS 232C accompanying signals (S7-300, S7-400)

Description

Using the S_V24 instruction, you can configure additional parameters if the module is configured for automatic control of the RS 232C accompanying signals.

Note

The instruction S_V24 must be started before the instructions S_SEND, S_RCV, S_VSET and S_VSTAT and have completed its task (DONE = TRUE).

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
TIME_RTS_OFF	INPUT	TIME	Time that must elapse after transmission before RTS is disabled. 0 ms to 10 min 55 s 350 ms Default: 10 ms
DATA_WAIT_TIME	INPUT	TIME	Time to wait until the partner sets CTS = ON once RTS has been set. 0 ms to 10 min 55 s 350 ms Default: 10 ms
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6464) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6464) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart

[1] The DONE, ERROR and STATUS parameters are available for **one** CPU cycle following a successful request.

Allocation in the data area

The S_V24 instruction works in combination with an I_V24 instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for more precise error information.

Startup

The COM_RST parameter of the S_V24 instruction is used to report a startup to the instruction.

Set the COM_RST parameter in the startup OB to 1.

Call the instruction in cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

STATUS parameter (S7-300, S7-400)

Event classes

Event class	Meaning
2	Error initializing
5	Error executing a CPU job
7	Send errors
8	Receive errors
11 ($0B_H$)	Warning
30 ($1E_H$)	Errors in communication between the module and CPU

STATUS parameter

Error code(W#16#...)	Description	Remedy
0201	No (valid) configuration available.	Provide the module with correct parameters. If necessary, ensure the system is correctly installed.
0502	Request not permitted in this operating mode of the ET 200S 1SI module (e.g., device interface not parameterized).	The send message frame is longer than 224 bytes. The ET 200S 1SI module canceled the transmission job. Select a shorter message frame length.
050E	Invalid telegram length	The send message frame is longer than 224 bytes. The ET 200S 1SI module canceled the transmission job. Select a shorter telegram length.
0550	The parameter update request is invalid for the currently selected form of ET 200S 1SI module data flow control.	Edit the instruction parameters (S_XON, S_RTS, S_V24) in the AS program or modify the data flow control function of the ET 200S 1SI module in the hardware configuration to ensure compatibility.
0551	Frame execution error during communication between the ET 200S 1SI module and the automation system. The error occurred in the automation system during transmission of a received message frame from the ET 200S 1SI module.	The module and the automation system have canceled the transmission. Repeat the receive request; the ET 200S 1SI module sends the received message again.
0702	Only for 3964(R): Error establishing connection: After STX was sent, NAK or any other code (except for DLE or STX) was received.	You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0703	Only for 3964(R): Acknowledgment delay time (QVZ) exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive or there is a break in the transmission line, for example. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0704	Only for 3964(R): Termination by partner: One or more characters were received from the partner during sending.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0705	Only for 3964(R): Negative acknowledgment when sending	Check if the partner is also indicating errors, possibly because not all transmit data has arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0706	Only for 3964(R): End-of-connection error: • Partner rejected telegram at end of connection with NAK or a random string (except for DLE) or • Acknowledgment characters (DLE) received too early.	Check if the partner is also indicating errors, possibly because not all transmit data has arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.

Error code(W#16#...)	Description	Remedy
0707	Only for 3964(R): Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send telegram: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device too slow or faulty. If necessary, use an interface test device switched into the transmission line to check.
0708	Only for ASCII drivers: The wait time for XON or CTS = ON has expired.	The communication partner is faulty, too slow or has been taken offline. Check the communication partner; you may need to change the parameter assignment.
070B	Only for 3964(R): Initialization conflict cannot be resolved because both partners have high priority.	Change the parameter assignment.
070C	Only for 3964(R): Initialization conflict cannot be resolved because both partners have low priority.	Change the parameter assignment.
0802	Only for 3964(R): Error establishing connection: • In idle mode, one or more random codes (other than NAK or STX) were received or • After an STX was received, partner sent more characters without waiting for response DLE. After partner power ON: • The module receives an undefined character while the partner is being switched on.	You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0805	Only for 3964(R): Logical error while receiving: After DLE was received, a further random code (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the telegram header and data string or the connection is terminated with DLE ETX. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0806	Character delay time (ZVZ) exceeded: • Two successive characters were not received within character delay time or Only for 3964(R): • 1. character after sending of DLE while establishing connection was not received within the character delay time.	Partner device too slow or faulty. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0807	Only for 3964(R): Illegal telegram length: A zero-length telegram has been received.	Receipt of a zero-length telegram is not an error. Check why the communication partner is sending telegrams without user data.

Error code(W#16#...)	Description	Remedy
0808	Only for 3964(R): Error in block check character (BCC): The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0809	Only for 3964(R): The number of repetitions must the set to the same value.	Declare a block wait time at the communication partner identical to that in your module. Check for malfunction of the communication partner; you may need to use an interface test device that is switched into the transmission line.
080A	There is no free receive buffer available: No receive buffer space available for receiving data.	Increase the call rate for the S_RCV instruction.
080C	Transmission error: • Transmission error (parity/stop bit/overflow error) detected. Only for 3964(R): • If this happens during send or receive operations, repetition is started. • If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. • If the SF LED (red) lights up, there is a break in the connecting cable between the two communication partners.	Disturbances on the transmission line cause message frame repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits.
080D	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
0810	Only for ASCII drivers: Parity error: • If the SF LED (red) lights up, there is a break in the connecting cable between the two communication partners.	Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0811	Only for ASCII drivers: Character frame error: • If the SF LED (red) lights up, there is a break in the connecting cable between the two communication partners.	Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0812	Only for ASCII drivers: More characters were received after the module had sent XOFF or set CTS to OFF.	Reconfigure the communication partner or read module data more rapidly.
0818	Only for ASCII drivers: DSR = OFF or CTS = OFF	The partner has switched the DSR or CTS signal to "OFF" before or during a transmission. Check the partner's control of the RS 232C accompanying signals.

Error code(W#16#...)	Description	Remedy
0850	The length of the receive message frame is greater than 224 bytes or the defined message frame length.	Adjust the message frame length of the partner
0B01	Reception buffer is more than 2/3 full	
1E0D	"Job aborted due to warm restart, hot restart or reset"	
1E0E	Static error at the call of SFC DPRD_DAT. The RET_VAL return value of the SFC is available for evaluation in the SFCERR variable of the instance DB.	Load the SFCERR tag from the instance DB.
1E0F	Static error at the call of SFC DPWR_DAT. The RET_VAL return value of the SFC is available for evaluation in the SFCERR variable of the instance DB.	Load the SFCERR tag from the instance DB.
1E10	Static error during the call of SFC RD_LGADR. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.
1E11	Static error during the call of SFC RDSYSST. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.
1E20	Parameter out of range.	Modify the input of the instruction so that it is within the valid range.
1E41	Invalid number of bytes specified at instruction parameter LEN	You must stay within a range of values of 1 to 224 bytes.

Evaluating the SFCERR variable

More detailed information on the error events (1E) $0E_H$, (1E) $0F_H$, (1E) 10_H and (1E) 11_H of event class 30 is available in the SFCERR variable.

You can load the SFCERR variable from the instance DB of the corresponding instruction.

The error messages entered in the SFCERR variable are described in the sections on the "DPRD_DAR" and SFC15 "DPWR_DAT" system functions in the *System Software for S7-300/400, System and Standard Functions* Reference Manual.

ET 200S 1SI for Modbus (S7-300, S7-400)

Sending and receiving as Modbus master (S7-300, S7-400)

Description

Data is transferred between the module and CPU by means of the S_SEND and S_RCV instructions. The S_SEND instruction is activated by an edge at the REQ input if data is to be output. The S_RCV instruction is set ready to receive with EN_R=1. An S_RCV is required for all reading function codes.

Sending and receiving data using the Modbus master protocol

The S_SEND and S_RCV instructions must be activated in order to execute a Modbus master job. The S_SEND instruction is activated by an edge at the REQ input when data is to be output to the module. The S_RCV instruction is made ready to receive data from the module with EN_R=1. An S_RCV is required for all reading function codes. The figure below shows the overall characteristics of the S_SEND and S_RCV parameters when a Modbus job is executed.

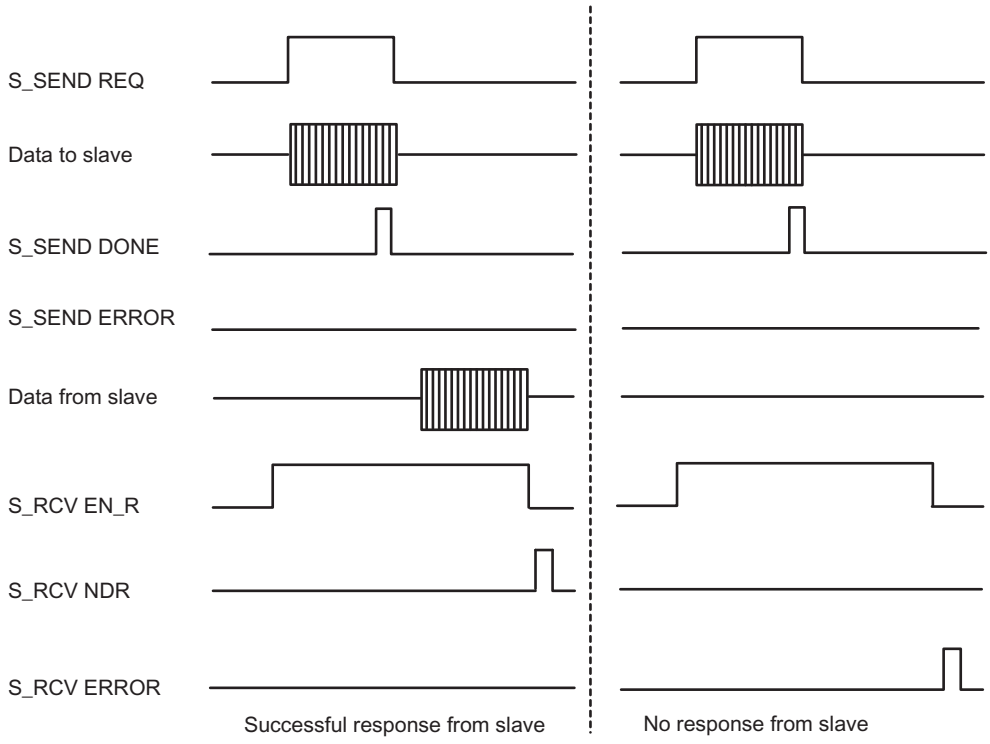

Note

Send and receive sequence

As the interface between the user program and the interface module operates in half-duplex mode you must observe the following items:

After positive acknowledgment of the Modbus master read job, you first have to fetch the receive data from the interface module by calling the S_RCV instruction before starting a new Modbus master send job.

S_RCV: receiving data (S7-300, S7-400)

Description

The S_RCV instruction sends data from the ET 200S 1SI module to an S7 data area as specified at the DB_NO and DBB_NO parameters. For data transmission, the S_RCV instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from the ET 200S 1SI module. An active transmission can be canceled by setting signal state "0" at parameter EN_R and with signal state "1" at parameter R. (Because transmission may sometimes not start up again after setting parameter EN_R to "1", you should set parameter R to "1" in addition to setting EN_R to "0" to cancel an active transmission.) The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R. Data transmission can take several calls (program cycles), depending on the data volume.

If the instruction detects signal state "1" at the R parameter, the current transmission job is canceled and the S_RCV instruction is reset to the initial state. Receiving is disabled as long as signal state "1" is set at parameter R. If the signal state returns to "0", the canceled message frame is received again from the beginning.

Parameter LADDR specifies the address of the ET 200S 1SI to be addressed.

Output NDR indicates "Job completed without errors/data received" (all data read) and ERROR indicates error events. A corresponding error number is displayed in STATUS if an error occurs. If the receive buffer is filled to more than 2/3 of its capacity, STATUS returns a warning at each call of S_RCV. If no errors or warnings are pending, STATUS has a value of "0".

NDR and ERROR/STATUS are also output after the reset of the S_RCV instruction (LEN == 16#00) (see time sequence chart). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enable data reading
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Receive data block no.: CPU-specific, zero not allowed

Parameters	Declaration	Data type	Description
DBB_NO	INPUT	INT	Data byte number (received data) **S7-1500**: 0 ≤ DBB_NO and DBB_NO + LEN ≤ size of the DB **S7-300/400**: 0 ≤ DBB_NO and DBB_NO + LEN < 8190
NDR [1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6477) contains the error information.
LEN [1]	OUTPUT	INT	Length of the message frame received 1 ≤ LEN ≤ 224 specified in number of bytes
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6477) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a receive job, the NDR, ERROR, LEN and STATUS parameters remain available for the duration of **one** CPU cycle!			

Allocation in the data area

The S_RCV instruction works in combination with an I_RCV instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If the error STATUS == W#16#1Exx occurs, you can examine the SFCERR variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Startup

The COM_RST parameter of S_RCV instruction is used to report a startup to the instruction.

In the startup OB, set parameter COM_RST to 1.

Call the instruction in the cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the wiring of the EN_R and R inputs.

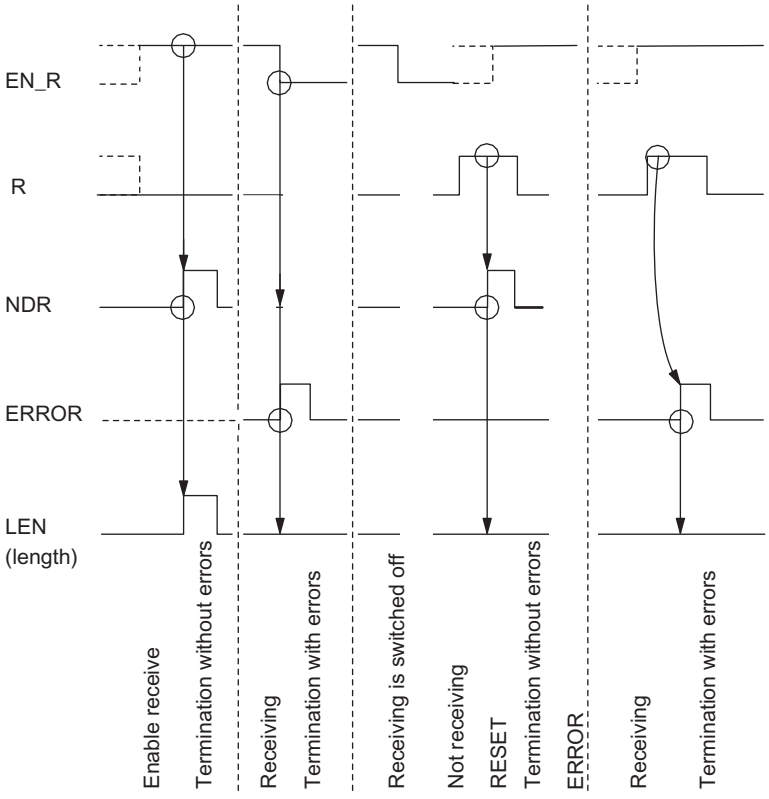

Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

Rules

Note

The S_RCV instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CPU startup routine of the ET 200S module for the S_RCV instruction must be completed before the ET 200S 1SI module can receive a job initiated after the status transition of the CPU from STOP to RUN.

S_SEND: Sending Data (S7-300, S7-400)

Description

The S_SEND instruction transmits a data block from a DB to the ET 200S 1SI as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the S_SEND instruction is called in a static (unconditional) operation in the cycle or **alternatively**, in a time-controlled program.

Operating principle

Data transmission is initiated by a positive edge at the REQ input. Data transmission can take several calls (program cycles), depending on the data volume.

The S_SEND instruction can be called in the cycle by setting signal state "1" at parameter input R. This setting cancels the transmission to the ET 200S 1SI module and resets the S_SEND instruction to the initial state. Data that has already been received by the ET 200S 1SI module is still transmitted to the communication partner. A static signal state "1" at the input R indicates that data transmission is disabled.

Parameter LADDR specifies the address of the ET 200S 1SI to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no errors occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for RESET of the S_SEND instruction (see the time sequence chart). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number (transmit data) **S7-1500**: 0 ≤ DBB_NO and DBB_NO + LEN ≤ size of the DB **S7-300/400**: 0 ≤ DBB_NO and DBB_NO + LEN < 8190
LEN	INPUT	INT	Data length 1 ≤ LEN ≤ 224 specified in number of bytes

Parameters	Declaration	Data type	Description
DONE [1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 6477) contains the error information.
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 6477) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a transmission job, the DONE, ERROR and STATUS parameters remain available for the duration of **one** CPU cycle!			

Allocation in the data area

The S_SEND instruction works in combination with an I_SEND instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If the error STATUS == W#16#1Exx occurs, you can examine the SFCERR variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Startup

The COM_RST parameter of S_SEND instruction is used to report a startup to the instruction.

In the startup OB, set parameter COM_RST to 1.

Call the instruction in the cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the wiring of the REQ and R inputs.

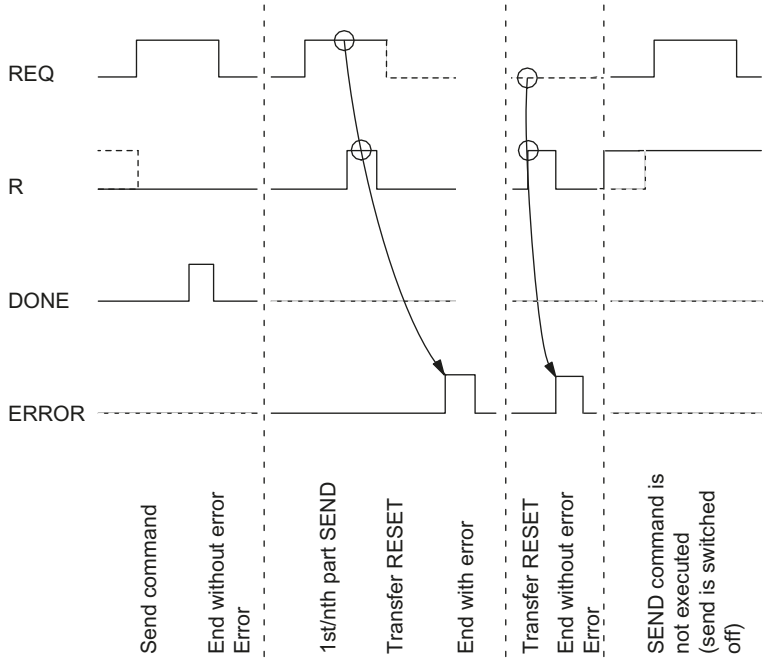

Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rules

Note

The S_SEND instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CPU startup routine of the ET 200S module for the S_SEND instruction must be completed (see above) before the ET 200S 1SI module can execute a job initiated after the status transition of the CPU from STOP to RUN. A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the ET 200S 1SI module.

S_MODB: Modbus slave instruction for ET 200S 1SI (S7-300, S7-400)

Description

To enable execution of a Modbus slave job, the S_MODB instruction must be called cyclically in the user program. The S_MODB instruction receives the job from serial interface module ET200S Modbus/USS, executes it and returns the response to the module. Communication between the CPU and module is handled by means of the S_SEND and S_RCV instructions which are called by S_MODB .

Operating principle

Following each warm restart of the CPU, the user program must initialize the Modbus communication instruction. Initialization is triggered by a rising edge at the CP_START input. The instruction records the size of the I, Q, F, T and C address areas of the CPU in its instance data block. On successful completion of initialization, the FB sets the CP_START_OK output.

An initialization error is specified by the CP_START_ERROR output. In this case, Modbus communication is not possible and all requests from the Modbus master are answered with an exception code message.

S_MODB uses a Modbus data conversion table that is located in the data block in order to map the Modbus addresses to the SIMATIC S7 memory areas.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of serial interface module ET 200S 1SI. The start address is taken from STEP 7.
START_TIMER	INPUT	TIMER	Time for initialization timeout: CPU time that the instruction uses to prevent the initialization from freezing.
START_TIME	INPUT	S5TIME	Time value for initialization timeout: Valid maximum time for the instruction's initialization
DB_No	INPUT	BLOCK_DB	Number of the data block that contains the Modbus conversion table: See the data areas in the SIMATIC CPU
OB_MASK	INPUT	BOOL	Masks I/O access errors and delays interrupts: TRUE can be set to specify that access errors to non-existing I/O are reported to the Modbus master only
CP_START	INPUT	BOOL	Start initialization of the instruction
CP_START_FM	INPUT	BOOL	Initialization is triggered by a rising edge at the input START: Memory element to which the instruction saves the previous value of CP_START
CP_NDR	OUTPUT	BOOL	Receive order frame

Parameters	Declaration	Data type	Description
CP_START_OK	OUTPUT	BOOL	Initialization successfully completed: Initialization was completed without errors before expiration of the monitoring time.
CP_START_ERROR	OUTPUT	BOOL	Initialization terminated with error: Initialization could not be completed without errors, even after expiration of the monitoring time.
ERROR_NR	OUTPUT	WORD	Error number See Diagnostics
ERROR_INFO	OUTPUT	WORD	Additional error information See Diagnostics

Allocation in the data area

The S_MODB instruction works in combination with an IDB_MODB instance DB, the number of which is specified in the call.

Access to the data in the instance DB is prohibited.

How to Handle an Error

Input parameter OB_MASK can be used to instruct the Modbus instruction to mask I/O access errors. If an attempt is made to write to an I/O that is not available, the CPU does not go into STOP mode or call an error OB. The instruction detects the access error, cancels the job and returns an error response to the Modbus master.

STATUS parameter (Modbus / USS) (S7-300, S7-400)

Event classes

Event class	Meaning
2	Error initializing
5	Error executing a CPU job
8	Receive errors
14 ($0E_H$)	General processing errors: Parameter assignment, processing of an S_SEND job. Evaluation of data received, receipt of an exception code message
30 ($1E_H$)	Errors in communication between the module and CPU

STATUS parameter

Error code(W#16#...)	Description	Remedy
0201	No (valid) configuration available.	Provide the module with correct parameters. If necessary, ensure the system is correctly installed.
0502	Order inadmissible in this operating mode of the serial interface module ET 200S Modbus/USS (example: device interface is not configured).	Evaluate the diagnostic interrupt and rectify the error accordingly.
050E	Invalid telegram length	The send message frame is longer than 224 bytes. The send job was aborted by the ET 200S Modbus/USS module. Select a shorter telegram length.
0530	Modbus master send job rejected because the response of the communication partner to a previous reading Modbus master send job has not yet been retrieved.	After a successful read Modbus master send job, you must first read the response from the communication partner from the module before you start a new Modbus master send job.
0551	Frame sequence errors in the communication between the serial interface module ET 200S Modbus/USS and the CPU. The error occurred while transmitting a received message frame of the ET 200S SI serial interface module to the CPU.	The module and the CPU have canceled the transmission. Repeat the receive job. The ET 200S Modbus/USS serial interface module sends the received message again.
0806	Character delay time exceeded. Two successive characters were not received within the character delay time.	Partner device too slow or faulty. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
080A	Overflow of the receive buffer at the master during the reception of the response message frame.	Check the protocol settings of the slave.
080C	Transmission error (parity/stop bit/overflow error) detected.	Disturbances on the data link cause telegram repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits.
080D	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
0810	Parity error: If the SF LED (red) lights up, there is a break in the cable that connects the two communication partners.	Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits. Change your system setup or cable wiring.

Error code(W#16#...)	Description	Remedy
0811	Character frame error: If the SF LED (red) lights up, there is a break in the cable that connects the two communication partners.	Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0812	Additional characters were received after the CTS serial port was set to OFF.	Reconfigure the communications partner or fetch from the serial interface faster.
0830	Master: A request message frame has been sent and the response monitoring time has elapsed without the start of a response message frame being detected. Slave: Broadcast not permitted with this function code.	Check if the transmission line has been interrupted (interface analysis may be necessary). Check that the same settings have been made on the module and communication partner for the following protocol parameters: transmission rate, number of data bits, parity and number of stop bits. Check that the value for the response monitoring time in PtP_PARAM is large enough. Check to see if the specified slave address is available. The Modbus master system is only allowed to use the broadcast function for the function codes enabled for this purpose.
0831	Master: The first character in the slave response message frame is different from the slave address that was sent in the request message frame (for normal operation). Slave: Function code received not permissible.	The wrong slave has responded. Check if the transmission line has been interrupted (interface analysis may be necessary). This function code cannot be used for this driver.
0832	Maximum number of bits or registers exceeded or number of bits cannot be divided by 16 if the SIMATIC timers or counters memory areas are accessed.	Limit the maximum number of bits to 2,040 and the maximum number of registers to 127. Access to SIMATIC timers, counters in 16-bit intervals only.
0833	Bit or register number for function code FC 15/16 and message frame element byte_count do not match.	Correct the bit/register number or byte_count.
0834	Illegal bit coding for "Set bit / Reset bit" recognized.	Use only the 0000Hex or FF00Hex codes for FC05.
0835	Illegal diagnostic subcode (not 0000Hex) at function code FC 08 "Loop back test" recognized.	Use only the subcode 0000Hex for FC08.
0836	The internally generated value of the CRC 16 checksum does not match the CRC checksum received.	Check the formation of the CRC checksum on the Modbus master system.

Error code(W#16#...)	Description	Remedy
0837	Message frame sequence error: The Modbus master system sent a new request message frame before the driver transmitted the last response message frame.	Increase the timeout on the slave response message frame for Modbus master system.
0850	Length of the receive message frame greater than 224 bytes or greater than the configured message frame length.	Adjust the message frame length of the partner.
0E20	This connection requires 8 data bits. The driver is not ready for operation.	Correct the parameter assignment of the driver.
0E21	The parameterized multiplication factor for the character delay time is not within the range of 1 to 10. The driver is operating with a default setting of 1.	Correct the parameter assignment of the driver.
0E22	The operating mode set for the driver is illegal. "Normal operation" or "Noise suppression operation" must be specified. The driver is not ready for operation.	Correct the parameter assignment of the driver.
0E23	**Master**: An illegal value has been set for the response monitoring time: Valid values are between 50 and 655,000 ms. The driver is not ready for operation. **Slave**: An illegal value has been set for the slave address. Slave address 0 is illegal. The driver is not ready for operation.	Correct the parameter assignment of the driver. Correct the parameter assignment of the driver.
0E2E	An error occurred while the interface parameter file was being read. The driver is not ready for operation.	Restart the master (Mains_ON).
0E40	The value specified for LEN with S_SEND is too small.	The minimum length is 2 bytes.
0E41	The value specified for LEN with S_SEND is too small. A greater length is required for the transmitted function code.	The minimum length for this function code is 6 bytes.
0E42	The transmitted function code is illegal.	Only use permissible function codes.
0E43	Slave address 0 (= broadcast) is not permissible with this function code.	Only use the slave address 0 with suitable function codes.
0E44	The value of the "number of bits" transmitted is not within the range of 1 to 2,040.	The "number of bits" must be within the range of 1 to 2,040.
0E45	The value of the "number of registers" transmitted is not within the range of 1 to 127.	The "number of registers" must be within the range of 1 to 127.

Error code(W#16#...)	Description	Remedy
0E46	Function code 15 or 16: The values of the "number of bits"/"number of registers" transmitted is not in the range of 1 to 2,040 or 1 - 127.	The "number of bits"/"number of registers" must be within the range of 1 to 2,040 or 1 to 127.
0E47	Function code 15 or 16: LEN for S_SEND does not correspond to the "number of bits"/"number of registers" transmitted. LEN is too small.	Increase LEN for SEND until enough user data is transmitted to the module. More user data must be sent to the module because of the "number of bits"/"number of registers".
0E48	Function code 5: The code specified in the SEND source DB for "set bit" (FF00H) or "delete bit" (0000H) is incorrect.	The only permissible codes are "set bit" (FF00H), "delete bit" or 0000H.
0E49	Function code 8: The code specified in the SEND source code for "diagnostic code" is incorrect.	The only permissible code is "diagnostic code" 0000H.
0E4A	The length of this function code is greater than the maximum length.	The manual contains the maximum lengths for each function code.
0E50	The master received a response without sending anything.	A slave or another master is on the network. Check if the transmission line has been interrupted (interface analysis may be necessary).
0E51	Function code incorrect: The function code received in the response message frame is different from the function code sent.	Check the slave device.
0E52	Byte underflow: The number of characters received is less than indicated by the byte counter of the response message frame or expected for this function code.	Check the slave device.
0E53	Byte overflow: The number of characters received is greater than indicated by the byte counter of the response message frame or expected for this function code.	Check the slave device.
0E54	Byte counter incorrect: The byte counter received in the response message frame is too small.	Check the slave device.
0E55	Byte counter wrong: The byte counter received in the response message frame is wrong.	Check the slave device.
0E56	Echo incorrect: The response message frame data echoed by the slave (number of bits, etc.) is different from the data sent in the response message frame.	Check the slave device.

Error code(W#16#...)	Description	Remedy
0E57	CRC check incorrect: An error occurred when checking the CRC 16 checksum of the response message frame from the slave.	Check the slave device.
0E61	Response message frame with exception code 01: Illegal function	Refer to the manual for the slave device.
0E62	Response message frame with exception code 02: Illegal data address	Refer to the manual for the slave device.
0E63	Response message frame with exception code 03: Illegal data value	Refer to the manual for the slave device.
0E64	Response message frame with exception code 04: Failure in associated device	Refer to the manual for the slave device.
0E65	Response message frame with exception code 05: Acknowledgement	Refer to the manual for the slave device.
0E66	Response message frame with exception code 06: Occupied; message frame rejected	Refer to the manual for the slave device.
0E67	Response message frame with exception code 07: Negative acknowledgement	Refer to the manual for the slave device.
1E0D	"Job aborted due to warm restart, hot restart or reset"	
1E0E	Static error during call of the SFC DP_RDDAT. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.
1E0F	Static error during call of the SFC DP_WRDAT. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.
1E10	Static error during the call of SFC RD_LGADR. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.
1E11	Static error during the call of SFC RDSYSST. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.

Error code(W#16#...)	Description	Remedy
1E20	Parameter out of range.	Enter a parameter within the valid range for the instruction.
1E41	Invalid number of bytes specified at instruction parameter LEN	You must stay within a range of values of 1 to 256 bytes.

Evaluating the SFCERR variable

More detailed information on errors (1E) $0E_H$, (1E) $0F_H$, (1E) 10_H and (1E) 11_H belonging to event class 30 can be obtained via the SFCERR variable.

You can load the SFCERR variable from the instance DB of the corresponding instruction.

Error messages entered in the SFCERR variable can be found in the system functions "DPRD_DAT: Read consistent data of a DP standard slave (Page 5262)" and SFC15 "DPWR_DAT: Write consistent data of a DP standard slave (Page 5265)" and RD_LGADR: Determine all logical addresses of a module (Page 5553).

ERROR_NR, ERROR_INFO parameters (S7-300, S7-400)

Error classes for diagnostics

Error numbers from ... to	Description	Meaning
1 ... 9	Instruction and CP initialization error	Error numbers 1 to 9 indicate that initialization was terminated with errors. The CP_START_ERROR parameter is set to 1. Modbus communication to the master system is not possible.
10 ... 19	Error executing a function code	Error numbers 10 to 19 indicate that an error occurred during the processing of a function code. The CP transmitted an invalid execution job to the communication instruction. The error is also reported to the driver. Subsequent execution jobs continue to be processed.
90 ... 99	Other faults/errors	A processing error has occurred. The error is not reported to the driver. Subsequent execution jobs continue to be processed.

Error numbers

ERROR_ No (decimal)	ERROR_INFO	Error Text	Remedy
0	0	No error	
1	SFC 51->RET_VAL	Error reading the system status list with SFC 51.	Analyze RET_VAL in ERROR_INFO; eliminate the cause.

ERROR_ No (decimal)	ERROR_INFO	Error Text	Remedy
2	S_SEND->STATUS, S_RCV->STATUS	Timeout or error during module initialization (error in S_SEND request).	Check if this interface was assigned parameters for operation with "Modbus slave" protocol. Verify the "ID" specified in the communication instruction. Analyze ERROR_INFO.
11	Start address	The driver transferred an invalid start address to the communication instruction.	Check the Modbus address of the Modbus master system.
12	Number of registers	The driver transferred an invalid number of registers to the communication instruction. Number of registers = 0.	Check the number of registers of the Modbus master system. If necessary, restart the module (Mains_ON).
13	Number of registers	The driver transferred an invalid number of registers to the communication instruction: Number of registers > 128.	Check the number of registers of the Modbus master system. If necessary, restart the module (Mains_ON).
14	Bit memory M – End address	Attempt to access the SIMATIC flags memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.
15	Outputs Q – End address Inputs I – End address	Attempt to access the SIMATIC outputs memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.
16	Timers T – End address	Attempt to access the SIMATIC timers memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.
17	Counters C – End address	Attempt to access the SIMATIC counters memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.
18	0	The driver transferred an invalid SIMATIC memory area to the communication instruction.	If necessary, restart the module (Mains_ON).
19		Error during access to SIMATIC I/Os.	Check if the required I/Os exist and are error-free.
20	DB#	DB does not exist.	Add the DB to your project.
21	DB#	DB length invalid	Increase the DB length.
22	DB#	DB# is below the minimum DB value.	Change the minimum DB value.

ERROR_ No (decimal)	ERROR_INFO	Error Text	Remedy
23	DB#	DB# is above the maximum DB value.	Change the maximum DB value.
24	Flag address	Flag is below the lower limit.	Change the lower limits of the flags in the conversion DB.
25	Flag address	Flag is above the upper limit.	Change the upper limits of the flags in the conversion DB.
26	Output address	Output is below the lower limit.	Change the lower limits of the outputs in the conversion DB.
27	Output address	Output is above the upper limit.	Change the upper limits of the outputs in the conversion DB.
90	S_SEND-> STATUS	Error during transmission of an acknowledgement message frame to the driver with S_SEND.	Analyze the STATUS information.
94	S_RCV->STATUS	Error reading SYSTAT with S_RCV (STATUS).	Analyze the STATUS information.

ET 200S 1SI for USS (S7-300, S7-400)

S_USST: Send data to a USS slave (S7-300, S7-400)

Description

The S_USST instruction handles the transmission of network data (PZD and any PKW data) to the slaves, depending on the network data structure used.

Operating principle

The instruction fetches the parameter assignment of the current slave from the polling list (parameter DB) and transmits the data from the network data DB. It evaluates the communication control word of the current slave (triggering of a PKW request/ acknowledgement of a parameter change report), completes the USS transmit data and transmits it to the send buffer of the communications processor DB. It then initiates network data transmission to the slave by calling the S_SEND instruction.

If the instruction detects a parameter assignment error in the parameter DB, an error signal is stored in the parameter assignment error 2 byte of the network data DB.

The instruction is called once per automation system cycle.

Parameters

Parame-ters	Declaration	Data type	Description
DBPA	INPUT	INT	Block number of the parameter DB CPU-specific (zero is not allowed)
SYPA	INPUT	INT	Start address of the system parameters in the parameter DB 0 <= SYPA <= 8174
SLPA	INPUT	INT	Start address of the slave parameters in the parameter DB 0 <= SLPA <= 8184

Program structure of S_USST

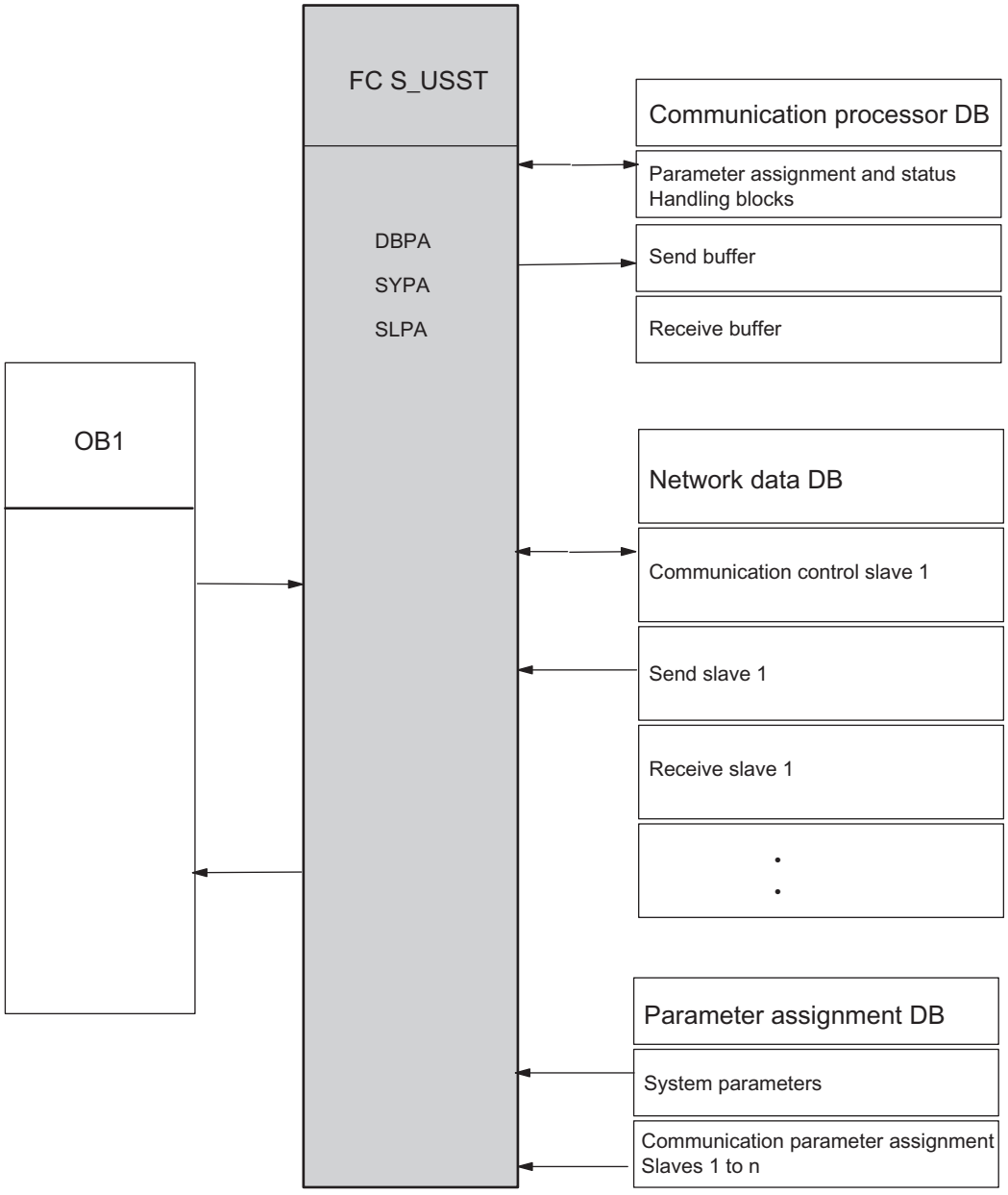

S_USSR: Receive data from a USS slave (S7-300, S7-400)

Description

The S_USSR instruction handles the receipt of network data (PZD and any PKW data) from the slaves, depending on the network data structure used.

Operating principle

The instruction fetches the parameter assignment of the current slave from the polling list (parameter DB) and evaluates the status word of the TRANSMIT block.

If the current request has been terminated without errors (bit 9 = 0 in the communication status word of the network data DB), the incoming data from the receive buffer of the communications DB is transmitted to the network data DB and evaluated. The communication status word is then updated in the network data DB.

If the current request has been terminated with an error (bit 9 = 1 in the communication status word of the network data DB), the data of the current slave is not accepted from the receive buffer of the communications processor DB. The instruction specifies this in the communication status word of the network data DB and enters the cause of error in the communication error word.

If the block detects a parameter assignment error in the parameter DB, an error signal is stored in the parameter assignment error 1 byte of the network data DB.

The instruction is called once per automation system cycle.

Parameters

Parame-ters	Declaration	Data type	Description
DBPA	INPUT	INT	Block number of the parameter DB CPU-specific (zero is not allowed)
SYPA	INPUT	INT	Start address of the system parameters in the parameter DB 0 <= SYPA <= 8174
SLPA	INPUT	INT	Start address of the slave parameters in the parameter DB 0 <= SLPA <= 8184

Program syntax of S_USSR

The diagram below shows the program syntax of S_USSR.

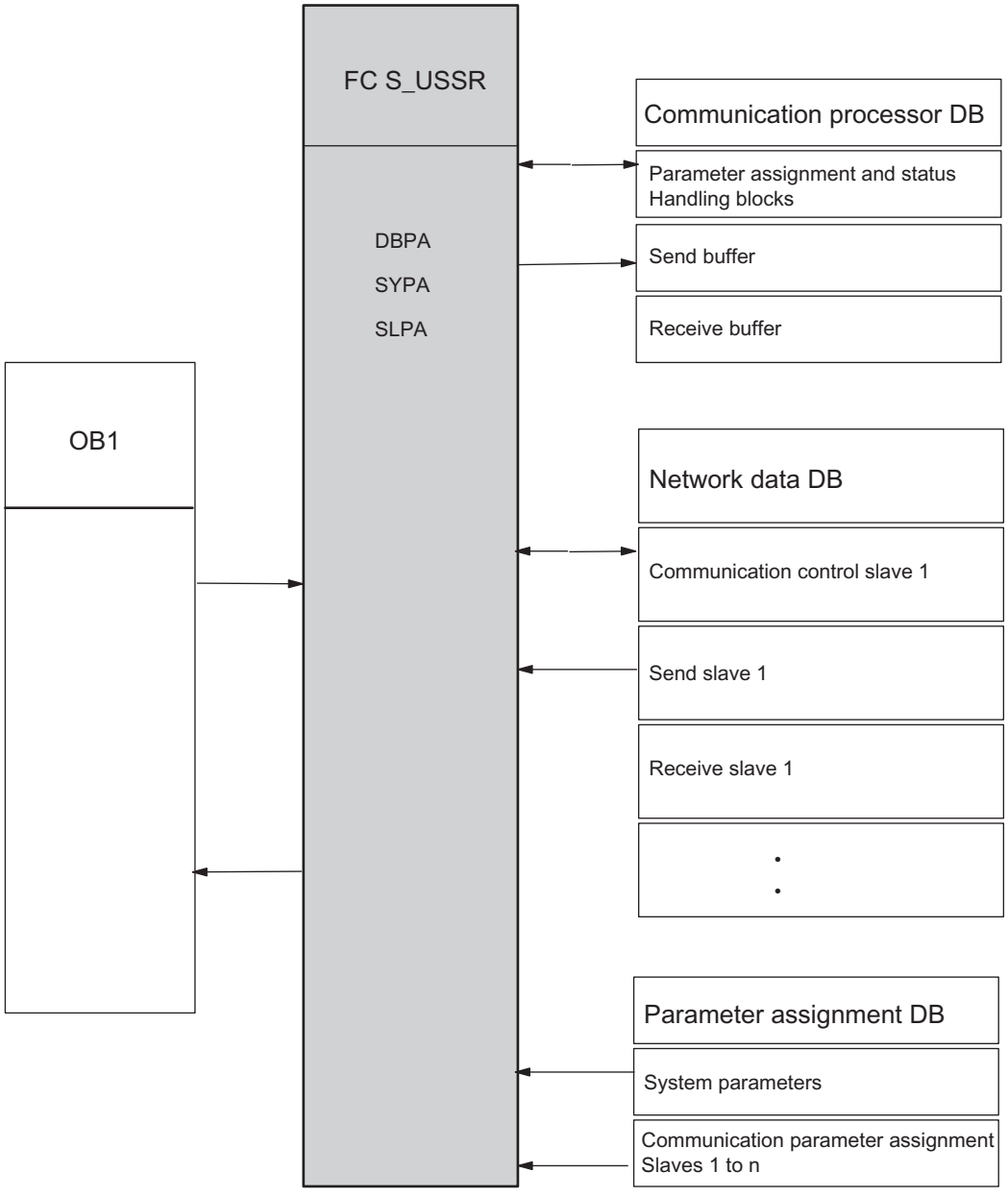

Rule

Note

The parameters of the U_USST instruction correspond to the parameters of the S_USSR instruction. The two functions access the same configuration (system and slave parameters) in the parameter DB and, consequently, have to be assigned identical parameters.

S_USSI: Initialize USS (S7-300, S7-400)

Description

The S_USSI instruction is an optional function.

Call of this instruction during the startup of the S7 system initiates the generation of the communications processor DB, network data DB and parameter DB. These data blocks are indispensible for communication. The DBPA is also preset. The S_USSI instruction can be used to generate and preset the specified data area, provided all slaves have the same network data structure.

Note

If you do not use S_USSI, you have to pre-assign a value (e.g., 20000) to DMW 2 in the communications processor DB.

Note

S_USSI on S7-1500

Note the following when using the S_USSI instruction of the distributed I/O ET 200S 1SI on an S7-1500.

The parameter ANZ shows the value 0 even in the case of an error.

If the CPU is switched to STOP with the S_USSI instruction, check all possible errors listed at the parameter ANZ.

Operating principle

When called, the instruction first checks the plausibility of its parameter assignment in terms of the number of slaves, network data structure, start node number and PKW repetitions. If the block detects an error during this, the data blocks are not generated or preset. The CPU goes into STOP mode and the user receives an error message via the error byte of the S_USSI instruction. Once the parameter error has been eliminated, all data blocks that have already been generated must be deleted before a restart.

After the plausibility check, the block checks whether the data blocks to be generated already exist:

- If the data blocks to be generated do not already exist, they are generated and the DBPA is preset.
- If the data blocks to be generated already exist, the length of each data block is checked. If the DB is long enough, the parameter DB is preset again and the contents of the network data DB and the communications processor DB are deleted. If a DB is too short, the CPU goes into STOP mode. You can identify the faulty DB by the condition code byte of the S_USSI instruction. For the error to be eliminated, the three data blocks must be completely deleted. The data blocks are then generated again at the next restart and the parameter DB is preset.

The S_USSI instruction must be called once during system startup (OB 100).

Parameters

Parameters	Declaration	Data type	Description
SANZ	INPUT	INT	Number of slaves with the same network data structure (system parameters in the DBPA) 1 ≤ SANZ ≤ 31
TNU1	INPUT	INT	Start node number (station number) 0 ≤ TNU1 ≤ 31
PKW	INPUT	INT	PKW, number Number of words of the PKW interface 0, 3 or 4
PZD	INPUT	INT	PZD, number Number of words of PZD interface 0 <= PZD <= 16
DBND	INPUT	INT	Network data DB number CPU-specific (zero is not allowed) Permitted range: 60000 to 60999
DBPA	INPUT	INT	Parameter DB number CPU-specific (zero is not allowed) Permitted range: 60000 to 60999
DBCP	INPUT	INT	Communications processor DB number CPU-specific (zero is not allowed) Permitted range: 60000 to 60999
WDH	INPUT	INT	Number of permissible PKW request repetitions 0 ≤ WDH ≤ 32767
ANZ	OUTPUT	BYTE	Error byte 0: No error 1: Too many slaves 2: Invalid data for the network data structure 3: Configuration DB: Invalid number or DB too short 4: Network DB: Invalid number or DB too short 5: Incorrect station number 6: Communication processor DB: Invalid number or DB too short 7: Free 8: Repetition counter: Incorrect value

Network data DB (S7-300, S7-400)

Description

The network data DB forms the interface between the communication and control programs. The user must make this block available as "empty" and it must be sufficiently long.

Operating principle

You can generate and preset these blocks by calling the S_USSI instruction during CPU startup (DBPA only) or enter them manually.

Only the transmit data for a slave is entered in the network data DB send buffer that is assigned to the slave by the control program. The response data from the slave is accepted from the appropriate reception buffer (after evaluation of bit 9 in the communication control word). Status words can be used to precisely monitor communication, while the control word allows you to precisely control the beginning of a parameter assignment request.

The communication interface contains the following data for each slave:

- Slave-related communication data (communication control, tracking, 6 data words)
- Buffer for the current PKW job (only if a PKW area exists)
- Send buffer for network data (maximum of 20 data words)
- Reception buffer for network data (maximum of 20 data words)

The lengths of the send and receive buffer depend on the network data structure selected. If the PKW interface does not exist, the buffer for the current PKW job is not used.

The total length of the network data DB required depends on the number of slaves and the network data structure used.

Number of data words per slave = 2 x (PKW + PZD) + PKW + 6

where PKW = 0, 3 or 4 and 0 <= PZD <= 16

Example: A drive with a PKW area of 3 words and a PZD area of 2 words requires 19 data words in the network data DB.

With 31 slaves and the maximum network data length, the network data DB is 1,550 data words long. DBW0 is reserved.

Slave data assignment in the network data DB with 4 words in the PKW area and 0 to 16 words in the PZD area.

DBWn	Internal		Communication control
DBWn+2	Communication control word (KSTW)		
DBWn+4	Communication status word		Communication tracking
DBWn+6	Communication error word		Error status
DBWn+8	Internal		PKW attempt counter
DBWn+10	Parameter error 1 byte, parameter error 2 byte		Parameter error
DBWn+12	Parameter ID	PKE	Buffer for current PKW job
DBWn+14	Index	IND	
DBWn+16	Parameter value 1	PWE1	
DBWn+18	Parameter value 2	PWE2	

DBWn+20	Parameter ID	PKE	PKW area	Send buffer
DBWn+22	Index	IND		
DBWn+24	Parameter value 1	PWE1		
DBWn+26	Parameter value 2	PWE2		
DBWn+28	Control word (STW)	PZD1	PZD area (max. 16 words PZD)	
DBWn+30	Main setpoint (HSW)	PZD2		
DBWn+32	Setpoint/Additional control word	PZD3		
DBWn+34	Setpoint/Additional control word	PZD4		
...	...			
DBWn+58	Setpoint/Additional control word	PZD16	PKW area	Receive buffer
DBWn+60	Parameter ID	PKE		
DBWn+62	Index	IND		
DBWn+64	Parameter value 1	PWE1		
DBWn+66	Parameter value 2	PWE2		
DBWn+68	Status word (ZSW)	PZD1	PZD area (max. 16 words PZD)	
DBWn+70	Main process value (HIW)	PZD2		
DBWn+72	Process value/Additional status word	PZD3		
DBWn+74	Process value/Additional status word	PZD4		
...	...			
DBWn+98	Process value/Additional status word	PZD16		
	•			
(n = 2, 4 , 6...)	•			

Note

If there is no PKW area, neither the buffer for current PKW jobs nor the PKW area in the send buffer exists.

Communication control word KSTW (DBWn)

The bits in the communication control word coordinate the user program and the S_USST instruction.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Bit 0: Triggers the PKW job
  Bit 0 is set by the user if a new PKW job is in the send buffer and is to be processed. The instruction resets the bit if the PKW job is accepted.
- Bit 1: Acceptance of parameter change report
  Bit 1 is set by the user if the parameter change report has been accepted. The instruction resets the bit in order to acknowledge acceptance. Following this acknowledgement, the slave either continues to process the current job or transmits the next parameter change report.

Communication status word (DBWn+4)

The S_USST and S_USSR instructions set the bits in the communication status word.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Bit 0: PKW job in progress
  If the PKW job was accepted and the parameter ID (PKE) contains a valid job ID, instruction S_USST sets the bit to 0. Instruction S_USSR resets the bit after the PKW job has been executed (with or without errors) or if there are problems with the PKW interface.
- Bit 1: PKW job completed without errors
  Instruction S_USSR sets bit 1 if a PKW job has been executed without errors. The response should be taken from the receive buffer. Instruction S_USST resets the bit when a new PKW job is initiated.

  Note

  The PKW jobs for the slave are processed in the order specified in the polling list (DBPA). Only one job is active for each slave at any one time. If more than one slave is entered in the polling list, the response data for a new PKW job is only available on a positive edge of bit 1 (or bit 2).

- Bit 2: PKW job completed with errors
  Instruction S_USSR sets bit 2 for the response ID in the PKE. The error number is in the PWE of the slave response. Instruction S_USST resets the bit when a new PKW job is initiated.

  Note

  The last PKW job transmitted by the user is stored in the send interface after processing. Transmission to the slave is repeated until a new job is entered. This might require additional responses in the user program in the event of the status PKW job being terminated with an error (bit 2) and a PKW interface error (bit 4).

- Bit 3: PKW job ID is invalid.
  Instruction S_USST sets bit 3 if job ID 15 is set in the PKE or index 255 is entered in job ID 4. Instruction S_USST resets the bit if the next PKW job is triggered with a valid job ID in the PKE.

- Bit 4: PKW interface with an error (counter overflow).
  Instruction S_USSR sets bit 4 if the slave does not respond to the PKW job within a configurable number of job repetitions (WDH parameter in the parameter DB) or in the case of response ID 8 in the PKE. Instruction S_USSR resets the bit when a new PKW job is initiated and correctly executed.
- Bit 5: Response data contains a parameter change report.
  Instruction S_USSR sets bit 5 if a parameter change report from the slave is present (response IDs 9 to 12 and toggle bit 11 inverted). Instruction S_USST resets the bit when the user acknowledges the parameter change report (communication control word, bit 1).
- Bit 6: Operational fault on the slave.
  Instruction S_USSR sets and resets bit 6. The instruction evaluates the slave's status word (bit 3).
- Bit 7: There is a warning from the slave.
  Instruction S_USSR sets and resets bit 7. The instruction evaluates the slave's status word (bit 7).
- Bit 8: Automation system control requested.
  Instruction S_USSR sets and resets bit 8. The instruction evaluates the status word (bit 9) and the control word (bit 10).
- Bit 9: Group communication fault.
  Instruction S_USSR sets and resets bit 9. The instruction evaluates the feedback messages from the S_SEND and S_RCV standard blocks and checks the message frame received with respect to ADR, STX, BCC and LGE. The instruction also reports a timeout of the message frame monitoring time.

Note

The receive data from the network data DB is only valid where bit 9 = 0.

Structure of the communication error word (DBWn+6)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Bit 0: Addressing error (ADR)
- Bit 3: Start of message frame not detected (no STX for first character)
- Bit 4: Incorrect block check character (BCC)
- Bit 6: Incorrect message frame length (LGE)
  Instruction S_USSR sets bits 0, 3, 4 and 6 if an error is detected in the received frame (ADR, STX, BCC, LGE).
- Bit 7: Message frame monitoring time has elapsed
  Instruction S_USSR sets bit 7 if the time between the transmission of the message frame from the master to the slave and the arrival of the response from the slave exceeds the valid time calculated by the program (frame monitoring time).

The remaining bits are not used.

Parameter error 1 byte

Error message from the S_USSR instruction, parameter assignment error in parameter DB

- Value 0: No error
- Value 1: Incorrect data for PKW/PZD

Parameter error 2 byte

Error message from the S_USSR instruction, parameter assignment error in parameter DB

- Value 0: No error
- Value 1: Incorrect data for PKW/PZD

Parameter ID PKE in the send buffer

The user must assign the parameter number (bits 0 to 10) and the job ID (bits 12 to 15). The toggle bit for the parameter change report (bit 11) is masked by the S_USSR and S_USST instructions.

Network data DB (S7-300, S7-400)

Description

The arameter DB contains the program parameters required for communication control. The user needs to generate this block and set suitable defaults for the communication system (S_USSI or manually). The slaves on the bus are processed in the sequence in which they are entered in the DBPA (polling list).

A slave can also be entered more than once in the parameter DB in order to effectively step up its priority class.

Operating principle

The length of the parameter DB depends on the number (n) of slaves to be addressed in a bus cycle.
Number of data words of the parameter DB = (n x 4) + 5.

4 data words are required for each instance of slave communication and 4 data words are assigned once for the system parameters. DBW0 is reserved.

Data block allocation

DBW 0	Free	System parameters
DBW 2	DBCP	
DBW 4	SANZ	
DBW 6	SLAV	
DBW 8	WDH	

DBW 10	Number of PKW, number of PZD	Communications Parameter set slave 1
DBW 12	TUN	
DBW 14	DBND	
DBW 16	KSTW	
DBW 18	Number of PKW, number of PZD	Communications Parameter set slave 2
DBW 20	TUN	
DBW 22	DBND	
DBW 24	KSTW	
	Number of PKW, number of PZD	Communications Parameter set slave n
	TUN	
	DBND	
DBW (n x 8 + 8)	KSTW	

System parameters

DBCP	Block number of the communications processor DB
SANZ	Total number of slave parameter sets in the parameter DB. If some individual slaves are to be addressed more frequently than others in a bus cycle, their slave parameters must be entered more than once in the parameter DB. The SANZ system parameter must be adjusted accordingly.
SLAV	Number of the current slave (consecutive). Required by the S_USST FC and the S_USSR FC to calculate the current parameter set. This data word must be preset to 1. This is carried out by the S_USSI FC, if it is being used.
WDH	Number of permissible PKW job repetitions (value range: 0 through 32,767). If the current PKW job is not terminated within the set number, problems at the PKW interface are reported.

Slave communication parameter

Number of PKW, number of PZD	Definition of network data structure Left byte: Number of words for PKW area (0, 3, 4) Right byte: Number of words for PZD area (0 to 16) Any deviations from this are detected as parameter assignment errors (by the S_USST and S_USSR FCs) and entered in the parameter assignment error 1 byte, parameter assignment error 2 byte of the network data DB.
TUN	Node number corresponding to the bus address set on the drive (0 to 31).
DBND	Block number of the network data DB.
KSTW	Address of the communication control word KSTW for the slave in the network data DB.

Communication processor DB (S7-300, S7-400)

Description

This data block is used to handle data traffic between the CPU and serial interface module ET 200S Modbus/USS. Provide this block with a sufficient length. The communication processor DB must have a length of at least 50 words (DBW 0 to 98).

Syntax of the communication processor DB

DBW 0	Communication status		SEND and RECEIVE
DBW 2	Maximum number of cycles to expire when waiting to receive data	Cycle counter for timeout formation when waiting to receive data	FC17
DBW 4	Measured start interval		FC17
DBW 6	Duration of the last cycle (OB1_MIN_CYCLE)		FC17, OB1
DBW 8	Transmission frame length (LEN)		SEND
DBB10	Free		
DBB 11 : : DBB 54	Send buffer		Send frame to module (length is determined by the network data structure of the current slave)
DBB 55 : : DBB 98	Receive buffer		Received frame from module (length is determined by the network data structure of the current slave)

Communication status DBW0

Bit no.		
0	Input REQ for S_SEND	This bit is reset if bit 8 is set.
1	Input R for S_SEND	S_USST resets this bit at cyclic intervals.
2	Output DONE of S_SEND	
3	Output ERROR of S_SEND	
4	Input EN_R for S_RCV	S_USSR sets this bit at cyclic intervals.
5	Input R for S_RCV	S_USSR resets this bit at cyclic intervals.
6	Output NDR of S_RCV	
7	Output ERROR of S_RCV	
8	Job in progress (S_SEND saves the DONE bit)	S_USST resets this bit at cyclic intervals.

Duration of the last cycle DBW6

S_USST uses this parameter to measure slave response times. Before each call of S_USST , the user program must copy the cycle time of the automation system (OB1_MIN_CYCLE) to this parameter.

SIMATIC NET CPs

Industrial Ethernet

Introduction to Ethernet

General notes on the FCs / FBs

Available instructions

The following list shows the numbers of the program blocks (instructions) as supplied. You can change these numbers.

Please note that you must use different instructions the S7300 and S7400 (separate libraries).

Communication service / functional area	Type of instruction	Library	
		CP 300	CP 400
SEND / RECEIVE (open communications services)	AG_SEND	x	x
	AG_RECV	x	x
	AG_LSEND	x [2)]	x
	AG_LRECV	x [2)]	x
	AG_LOCK	x	x
	AG_UNLOCK	x	x
	AG_CNTEX	x	x [3)]
	AG_CNTRL	x	x [3)]
Programmed communication connections	IP_CONFIG	x	x
S7 communication	BSEND	x	
	BRCV	x	
	PUT	x	
	GET	x	
	USEND	x	
	URCV	x	
	C_CNTRL	x	
FTP	FTP_CMD	x	x

Communication service / functional area	Type of instruction	Library	
		CP 300	CP 400
PROFINET IO	PNIO_SEND	x	
	PNIO_RECV	x	
	PNIO_RW_REC	x	
	PNIO_ALARM	x	

1) PN_InOut / PN_InOut_Fast is supplied along with the SIMATIC iMap engineering tool.

2) Not to be used with the current CPs and no longer part of the current SIMATIC_NET_CP library

3) Depending on the CP type

Which version of the instructions should I use?

The following descriptions also include information on differences in behavior between the various versions of the instructions. Please check and note the version identifiers of the instructions you are using.

The latest release of the version history for SIMATIC NET instructions can be found on the Internet using the following entry ID:

9836605 (http://support.automation.siemens.com/WW/view/de/9836605)

The global libraries installed with the Automation Workbench contain the versions of the instructions current when the Workbench was released.

Note

We recommend that you always use the latest versions of the instructions for all module types.

You will find information on the current versions of the instructions and the current instructions to download from the Internet in our customer support:

http://support.automation.siemens.com/WW/view/en/8797900

With older module types, this recommendation assumes that you are using the latest firmware for the particular module type.

Instructions when modules are replaced

Module replacement in this sense means the replacement of a module with another module that may be a more recent version.

Note

Remember that if you replace a module, you must only use the instructions permitted for the configured CP type in the user program.

We recommend that you always use the latest versions of the instructions for all module types. With older module types, this recommendation assumes that you are using the latest firmware for the particular module type.

You will find more information on replacing modules in our Customer Support on the Internet.

The manuals for specific S7 CPs contain information on the compatibility of the S7CPs and the corresponding instructions.

See also

Programming manual "Functions (FC) and Function Blocks (FB) for SIMATIC NET S7 CPs" (http://support.automation.siemens.com/ww/view/de/30564821)

Setting parameters for FC calls

Before describing the instruction in detail, a few general comments on calling and setting parameters for instructions will be useful at this point.

The general information below applies to the following parameter groups that exist for all instructions:

- Parameters for CP and connection assignment (input parameters)
- Parameters for specifying a CPU data area (input parameters)
- Status information (output parameters)

Calling communications instruction for an S7300

NOTICE
Calling the communications instructions for S7-300 in more than one priority class is not permitted! If, for example, you call a communication instruction in OB1 and in OB35, execution of the instruction could be interrupted by the higherpriority OB. If you call instructions in more than one OB, write your program so that a communications instruction that is currently executing cannot be interrupted by another communications instruction (for example by disabling/enabling interrupts with system instructions).

Parameters for specifying a CPU data area (input parameters)

Specifying the data area on the CPU

When you call an instruction, you transfer the address and length of the data area on the CPU in which the user data is available or will be stored or which can contain further parameter information

The ANY pointer data type is used to address this area.

Status information (output parameters)

Evaluating status codes

For status evaluation, the following parameters must be evaluated in the user program:

- DONE or NDR
  These parameters (DONE with send jobs and NDR with receive jobs) signal (successful) completion of the job.
- ERROR
  This indicates that the job could not be executed errorfree.
- STATUS
  This parameter supplies detailed information about the execution of the job. Status codes can be returned during execution of the job (DONE=0 and ERROR=0).

Note

Remember that the status codes DONE, NDR, ERROR, STATUS are updated at each instruction call.

Status codes during CP startup

With a warm or hot restart on the Ethernet CP, the output parameters of the instruction are reset as follows:

- DONE = 0
- NDR = 0
- ERROR = 0
- STATUS =
  - 8180_H for AG_RECV / AG_LRECV
  - 8181_H for AG_SEND / AG_LSEND / AG_SSEND

Instructions for open communications services (SEND/RECEIVE interface)

Instructions for the SEND/RECEIVE interface

Overview

The following instructions are available for transferring data on the SEND/RECEIVE interface:

Instruction	Can be used with [1]		Meaning
S7-300	S7-400		
AG_SEND	x	x	for sending data
AG_RECV	x	x	for receiving data
AG_LSEND		x	for sending data
AG_LRECV		x	for receiving data
AG_SSEND		x	for sending data
AG_SRECV		x	for receiving data

1) Notes on the Instructions for an S7300 and S7400

- The following applies to S7300:
  - With the latest versions of the Ethernet CPs, only AG_SEND and AG_RECV are used; the data length can be up to 8192 bytes.
  - With S7–300 CPs (up to 6GK7 343–1EX10–0XE0 with firmware V2.2), use the instruction AG_LRECV on TCP connections instead of AG_RECV.
- The following applies to S7400:
  - With AG_SEND / AG_RECV, the data length per job is restricted to <=240 bytes. Longer data records (up to 8192 bytes) can be transferred with the AG_LSEND or AG_LRECV instructions.
  - AG_SSEND and AG_SRECV are for accelerated transfer of data by using optimized communication between CPU and CP in the S7 station. The fast communication has no effect on LAN communication. These two instructions are supported as of STEP 7 V5.4 SP3.
  - On an S7–400, AG_RECV cannot be used on TCP connections but only the AG_LRECV or AG_SRECV instructions.

Further information

Please check the supported data area for the S7CP you are using in the manual for the specific device. You will find an overview of the versions of the instructions in the SIMATIC NET block history.

Application

The following diagram illustrates the use of the instructions described here for bidirectional data transfer on a configured connection.

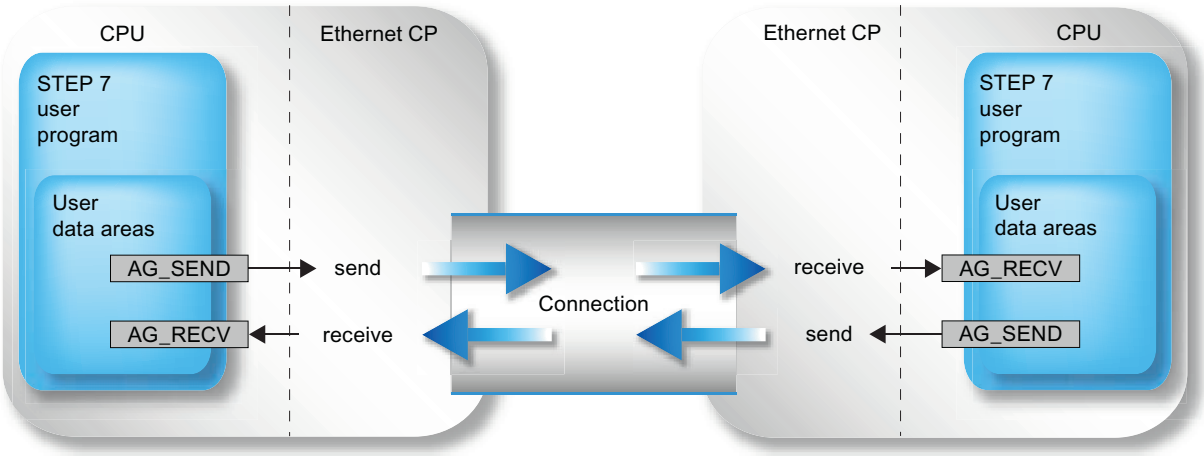

Specifying the data area on the CPU

When you call an instructions, you transfer the address and length of the data area in the CPU. Remember, that the maximum length of the data area depends on the type and version of the instruction being used.

- AG_SEND and AG_RECV
  Up to version V3.0 of these instructions, a maximum of 240 bytes can be sent or received. The current versions allow a data area of up to 8192 bytes for an S7300. With an S7400, AG_LSEND / AG_LRECV must still be used for larger data areas.
- AG_LSEND / AG_LRECV
  Using the CPs of the S7400 and with earlier versions of the S7300, larger data areas can only be transferred with AG_LSEND or AG_LRECV. Please check the length of the data area in the product information of the CP.
- AG_SSEND / AG_SRECV
  With CPs of the S7−400 that support PROFINET communication in conjunction with CPUs as of version 5.1, data can be transferred at higher transmission speeds with AG_SSEND or AG_SRECV (does not apply to the CP 443−1 Advanced 6GK7 443–1EX41–0XE0). You can check which CP types are supported by CPUs as of version 5.1 in the manual of your CP (Section "Requirements for use").

The following table shows the limit values of the various connection types.

INSTRUCTION	ISO transport	ISO-on-TCP	TCP	UDP
AG_LSEND (S7-400) AG_SEND (S7-300)	8192 bytes	8192 bytes	8192 bytes	2048 bytes
AG_SEND (S7-400)	240 bytes	240 bytes	240 bytes	240 bytes
AG_LRECV (S7-400) AG_RECV (S7-300)	8192 bytes	8192 bytes	8192 bytes	2048 bytes

INSTRUCTION	ISO transport	ISO-on-TCP	TCP	UDP
AG_RECV (S7-400)	240 bytes	240 bytes	240 bytes	240 bytes
AG_SSEND (S7-400) AG_SRECV (S7-400)	1452 bytes	1452 bytes	1452 bytes	1452 bytes

Note

For information on the length of the data area you can transfer with older versions of the Ethernet CPs, refer to the product information / manual of the Ethernet CP you are using.

Use without job header

On specified connections, the address and job parameters are specified by the connection configuration. The user program only provides the user data in the UDP data area when sending with AG_SEND / AG_LSEND / AG_SSEND or receives the data with AG_RECV / AG_LRECV / AG_SRECV.

Use with header

Free UDP connections require a job header in the user data area.

The following schematic illustrates the structure of the job buffer and the meaning and location (high byte / low byte) of the parameters in the job header.

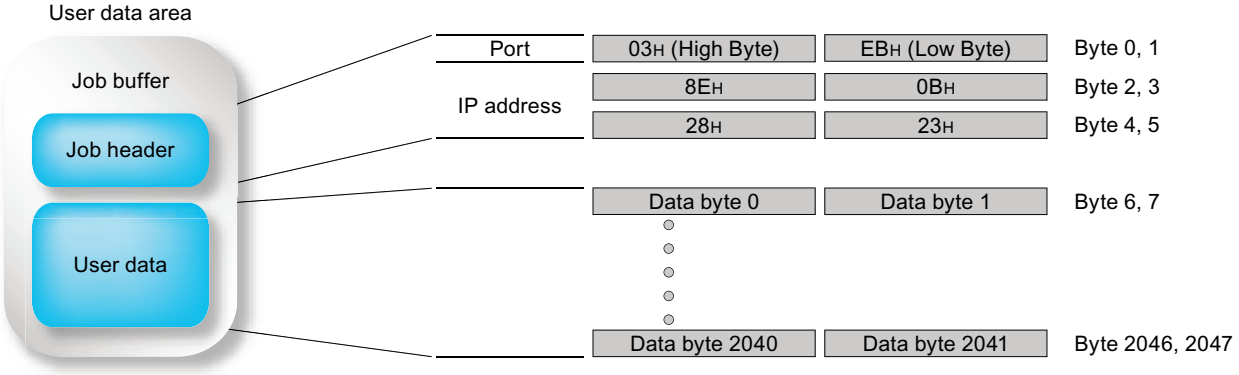

Figure 4-36 Sending and receiving on a free UDP connection with programmed addresses

- In the diagram (entries in hexadecimal) the following IP address is assumed as an example: 142.11.40.35;
- For the port address 1003, the following would be entered: For high byte: 03_H; for low byte: EB_H.
- The user data area can be up to 2048 bytes. Up to 2042 bytes of user data can be transferred. 6 bytes are reserved for the job header.
  Please note that the data length specified in the instruction call (LEN parameter) must include the header and the user data!

Change call parameters only after job confirmation

Note

Once the job has been triggered, you can only change the call parameters of the call interface of the AG_SEND or AG_RECV instruction after the instruction has confirmed completion of the job with DONE=1 or with ERROR=1.

If you do not keep to this rule, it is possible that the job will be aborted with an error.

Status display on the call interface of the instruction; Special situations with AG_SEND / AG_RECV (S7-300*)

The following behavior applies only to the S7-300 with program blocks as of version V4.0.

With AG_SEND and AG_RECV, you receive the following status codes for the operating situations mentioned:

- CP is in STOP.
- Connection is not configured.
- Connection is not established.
- Connection is aborted.

Codes:

- AG_SEND:
  DONE=0; ERROR=1; Status=8183_H
- AG_RECV:
  DONE=0; ERROR=0; Status=8180_H
  or
  DONE=0; ERROR=1; Status=8183_H

See also

General notes on the FCs / FBs (Page 6499)

AG_SEND / AG_LSEND / AG_SSEND

AG_SEND / AG_LSEND / AG_SSEND (Ind. Ethernet)

Description

AG_SEND / AG_LSEND / AG_SSEND pass data to the Ethernet CP for transfer over a configured connection.

The selected data area can be a memory bit area or a data block area.

Errorfree execution is indicated when the entire user data area could be sent over Ethernet.

Note:
Unless otherwise stated, all the following information applies equally to AG_SEND, AG_LSEND and AG_SSEND.

Call

Call interface in FBD representation

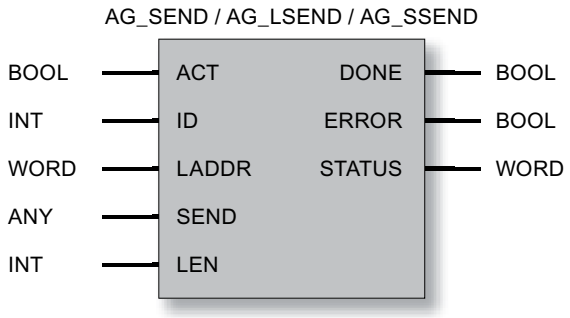

Note

If you want to use the AG_SSEND instruction, you will need to select the "SPEED SEND/ RECV" mode in the connection properties during configuration of the connection.

See also

Parameters for AG_SEND / AG_LSEND / AG_SSEND (IE) (Page 6508)

Parameters DONE, ERROR, STATUS (Page 6509)

AG_SEND / AG_LSEND (PROFIBUS) (Page 6646)

Programming manual "Functions (FC) and Function Blocks (FB) for SIMATIC NET S7 CPs" (http://support.automation.siemens.com/ww/view/de/30564821)

Parameters for AG_SEND / AG_LSEND / AG_SSEND (IE)

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_SEND / AG_LSEND / AG_SSEND instructions:

Parameter	Declaration	Data type	Range of values	Description
ACT	INPUT	BOOL	0,1	If the instruction is called with ACT=1, LEN bytes are sent from the ISO transport data area specified with the SEND parameter. If the instruction is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.
ID	INPUT	INT	1,2...64 (S7-400) 1,2...16 (S7-300)	The connection number of the connection is specified in the parameter ID.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.
SEND	INPUT	ANY		Specifies the address and length The address of the data area points to one of the alternatives: • Memory bit area • Data block area
LEN	INPUT	INT	On ISO transport and ISOonTCP / TCP: 1, 2 to 8192 (or up to "length specified for SEND parameter") On UDP: 1, 2 to 2048 (or up to "length specified for SEND parameter")	Number of bytes to be sent from the data area with this job. The possible values range from 1 to length specified for the SEND parameter. • Note the type of instruction: – For S7-300 The current versions of AG_SEND allow up to 8192 bytes (2048 bytes for UDP) to be transferred. – For S7-400 With AG_SEND, the data area is restricted to a maximum of 240 bytes. Note the following with an S7-400: • Improved performance with shorter data records: Transfer of data records up to 240 bytes results in better performance! This applies regardless of the type used (AG_SEND / AG_LSEND). • With AG_SSEND, the data area is restricted to a maximum of 1452 bytes.
DONE	OUTPUT	BOOL	0: Job active 1: Job done	The status parameter indicates whether or not the job was completed without errors. As long as DONE = 0, no further job can be triggered. When the job is accepted, DONE is set to 0 by the CP.

Parameter	Declaration	Data type	Range of values	Description
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 6509)
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Parameters DONE, ERROR, STATUS (Page 6509)

See also

AG_SEND / AG_LSEND / AG_SSEND (Ind. Ethernet) (Page 6506)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 4-345 AG_SEND / AG_LSEND / AG_SSEND status codes

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job completed without error.
0	0	0000_H	No job being executed.
0	0	8181_H	Job active.
0	1	7000_H	The condition code is possible only with S7-400: The instruction was called with ACT=0; the job has not yet been processed.
0	1	8183_H	No configuration or the ISO/TCP service has not yet started on the Ethernet CP.
0	1	8184_H	• Illegal data type specified for the SEND parameter. • System error (the source data area is incorrect).
0	1	8185_H	LEN parameter longer than SEND source area.
0	1	8186_H	ID parameter invalid. • ID != 1, 2 to 16 (S7-300) • ID != 1, 2 to 64.(S7-400)
0	1	8302_H	No receive resources on the destination station. Receiving station cannot process received data quickly enough or has not made any receive resources available.

DONE	ERROR	STATUS	Meaning
0	1	8304_H	The connection is not established. The send job should only be attempted again after waiting for at least 100 ms.
0	1	8311_H	The destination station cannot be obtained under the specified Ethernet address.
0	1	8312_H	Ethernet error on the CP.
0	1	$8F22_H$	Source area invalid, e.g.: Area does not exist in the DB LEN parameter < 0
0	1	$8F24_H$	Area error reading a parameter.
0	1	$8F28_H$	Alignment error reading a parameter.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	Wrong DB number
0	1	$8F3A_H$	Area not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference e.g. parameter LEN=0.
0	1	8090_H	• Module with this module start address does not exist; • The instruction being used does not match the system family being used (remember to use different instructions for S7300 and S7400).
0	1	8091_H	Module start address not at a doubleword boundary.
0	1	8092_H	In the ANY reference, a type other than BYTE is specified. (S7-400 only)
0	1	$80A4_H$	The communication bus connection between the CPU and CP is not established. (With newer CPU versions)
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	The specified length (in the LEN parameter) is incorrect.
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	CPU resources (memory) occupied.
0	1	$80C4_H$	Communications error Occurs temporarily and a repetition in the user program will often remedy the problem
0	1	$80D2_H$	Module start address incorrect.

See also

AG_SEND / AG_LSEND / AG_SSEND (Ind. Ethernet) (Page 6506)

AG_RECV / AG_LRECV / AG_SRECV

AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet)

Description

The AG_RECV / AG_LRECV / AG_SRECV instruction receives the data transferred on a configured connection from the Ethernet CP.

The data area specified for the receive data can be a memory bit area or a data block area.

Errorfree execution is indicated when the data could be received from the Ethernet CP.

Note:
Unless otherwise stated, all the following information applies equally to AG_RECV and AG_LRECV / AG_SRECV.

Call

Call interface in FBD representation

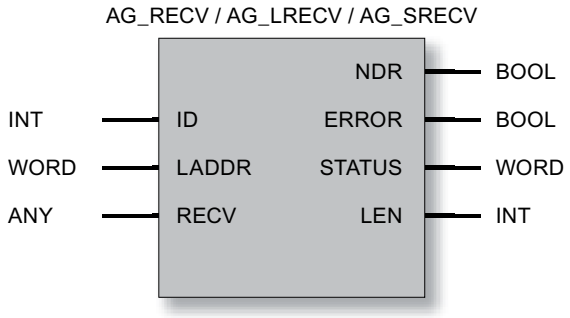

Note

If you want to use the AG_SRECV instruction, you will need to select the "SPEED SEND/ RECV" mode in the connection properties during configuration of the connection.

See also

AG_RECV / AG_LRECV (PROFIBUS) (Page 6653)

Parameters for AG_RECV / AG_LRECV / AG_SRECV (IE) (Page 6512)

Parameters DONE, ERROR, STATUS (IE) (Page 6513)

Parameters for AG_RECV / AG_LRECV / AG_SRECV (IE)

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_RECV / AG_LRECV / AG_SRECV instructions:

Parameter	Declaration	Data type	Range of values	Description
ID	INPUT	INT	1, 2 to 64 (S7-400) 1, 2 to 16 (S7-300)	The connection number of the ISO transport connection is specified in the ID parameter.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.
RECV	INPUT	ANY		Specifies the address and length The address of the data area points to one of the alternatives: • Memory bit area • Data block area Note on length: Performance is improved when transferring data records up to 212 bytes if you also restrict the length to 212 bytes at the RECV parameter. Point to note with AG_SRECV: With AG_SRECV, always set RECV to the maximum receive buffer length of 1452 bytes. Otherwise, the following error can occur in certain situations: NDR=0; ERROR=1; STATUS=8185_H
NDR	OUTPUT	BOOL	0: - 1: new data	The parameter indicates whether or not new data was accepted. For the meaning in conjunction with the ERROR and STATUS parameters, refer to AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet) (Page 6511)
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the NDR and STATUS parameters, refer to AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet) (Page 6511)

Parameter	Declaration	Data type	Range of values	Description
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the NDR and ERROR parameters, refer to AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet) (Page 6511)
LEN	OUTPUT	INT	**On ISO Transport and ISOonTCP:** 1,2,...8192 **On UDP:** 1,2,...2048	Specifies the number of bytes accepted from the Ethernet CP and entered in the data area. Note the type of instruction: • For S7-300 The current versions of AG_RECV allow up to 8192 bytes (2048 bytes for UDP) to be transferred. • For S7-400 With AG_RECV, the data area is restricted to a maximum of 240 bytes. With AG_SRECV, the data area is restricted to a maximum of 1452 bytes.

Parameters DONE, ERROR, STATUS (IE)

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 4-346 AG_RECV / AG_LRECV / AG_SRECV condition codes

NDR	ERROR	STATUS	Meaning
1	0	0000_H	New data accepted.
0	0	8180_H	There is no data available yet (not with AG_SRECV).
0	0	8181_H	Job active.
0	1	8183_H	• The configuration is missing; • The ISO transport service has not yet started on the Ethernet CP; • The connection is not established.
0	1	8184_H	• Illegal type specified for the RECV parameter; • System error.
0	1	8185_H	Destination buffer (RECV) is too short.

NDR	ERROR	STATUS	Meaning
0	1	8186_H	ID parameter invalid. ID != 1, 2 to 16 (S7-300). ID != 1, 2 to 64.(S7-400)
0	1	8304_H	The connection is not established. The send job should only be attempted again after waiting for at least 100 ms.
0	1	$8F23_H$	Source area invalid, e.g.: Area does note exist in the DB.
0	1	$8F25_H$	Area error writing a parameter.
0	1	$8F29_H$	Alignment error writing a parameter
0	1	$8F30_H$	Parameter is in the writeprotected first current data block.
0	1	$8F31_H$	Parameter is in the writeprotected second current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error
0	1	$8F3A_H$	Destination area not loaded (DB).
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F45_H$	Address of the parameter to be written is disabled in the access track.
0	1	$8F7F_H$	Internal error, for example illegal ANY reference.
0	1	8090_H	• No module with this module start address exists or the CPU is in STOP mode • The instruction being used does not match the system family being used (remember to use different instructions for S7300 and S7400).
0	1	8091_H	Module start address not at a doubleword boundary.
0	1	8092_H	In the ANY reference, a type other than BYTE is specified. (S7-400 only)
0	1	$80A0_H$	Negative acknowledgment reading from the module.
0	1	$80A4_H$	The communication bus connection between the CPU and CP is not established.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	Possible causes: • The destination area is invalid. • The destination area is too short. • The destination area for the received data was adequately dimensioned. Remedy: Run another receive call with maximum receive buffer size. This applies regardless of the connection type (unicast / multicast / broadcast) and the device family (S7-300 / S7-400).
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	CPU resources (memory) occupied.
0	1	$80C4_H$	Communications error Occurs temporarily and a repetition in the user program will often remedy the problem
0	1	$80D2_H$	Module start address incorrect.

See also

AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet) (Page 6511)

Instructions for access coordination with FETCH/WRITE

Instructions for access coordination with FETCH/WRITE

Overview

The following instructions are available for the FETCH/WRITE function to coordinate access:

Instruction	can be used with:		Meaning
S7-300	S7-400		
AG_LOCK	x	x	Locks external data access with FETCH/WRITE.
AG_UNLOCK	x	x	Releases external data access with FETCH/WRITE.

Caution when Configuring

If you use AG_LOCK and AG_UNLOCK, you must specify the following information for CPs in S7400 stations in the configuration:

- Under "Properties > Addresses"
  The "Address setting for LOCK/UNLOCK" option must be selected if the selection is available.

How It works

With these instructions, you can coordinate access to system memory areas so that no inconsistent data is created and transferred. The control is from the user program in the S7 CPU that can, if necessary, disable an external FETCH/WRITE access using an AG_LOCK call. After a certain time or after the local write/read access is completed, an AG_UNLOCK job can be used to enable external access again.

Another advantage is that this access lock only applies to the FETCH/WRITE connection specified in the call. If more than one FETCH/WRITE connection is configured, these can, for example, be used for certain specific system areas and a selective access coordination can be implemented.

The following diagram illustrates the usual chronological sequence of memory access coordination controlled in the user program with AG_LOCK and AG_UNLOCK.

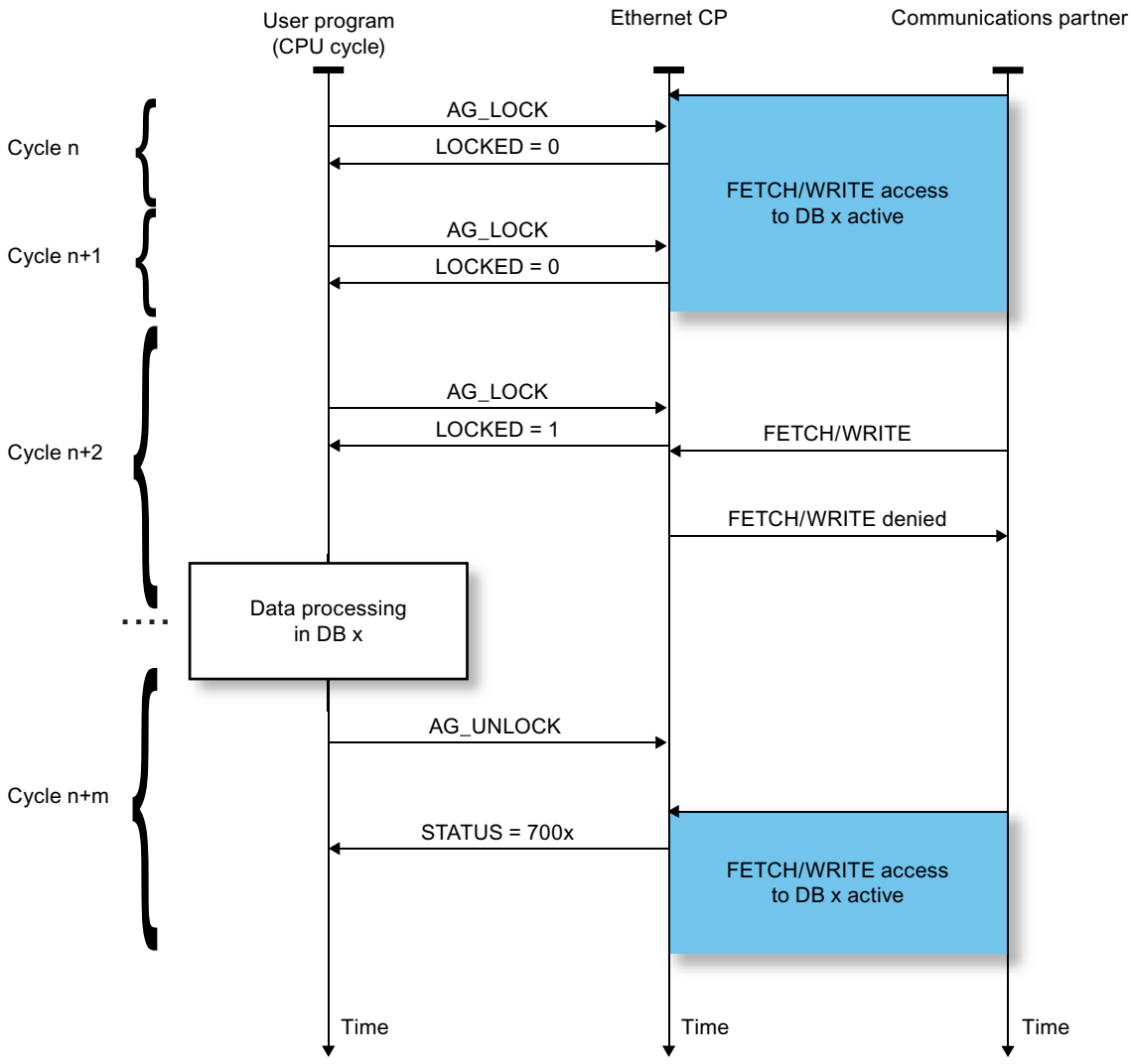

The lock job must first be monitored in the user program using the code in the return parameter LOCKED. As long as LOCKED=0 is indicated, it must be assumed that there is still an external FETCH/WRITE access active.

If LOCKED=1 is indicated, this shows that the lock is active; data can now be modified by the user program.

The status code is updated at each instruction call.

AG_LOCK

AG_LOCK

Description

Using the AG_LOCK instruction, the data exchange using FETCH or WRITE on the connection selected with the parameter ID is disabled. The LOCKED output indicates whether or not the lock was successful. If the lock was not successful, the job must be triggered again in a later CPU cycle.

The STATUS output indicates the status of the CP for this connection.

Call

Call interface in FBD representation

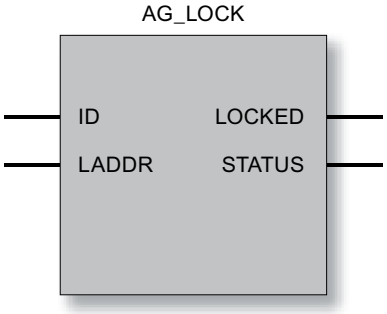

See also

Parameters for AG_LOCK (Page 6517)

STATUS parameter (Page 6518)

Parameters for AG_LOCK

Explanation of the formal parameters

The following table explains all the formal parameters for AG_LOCK:

Parameter	Declaration	Data type	Range of values	Description
ID	INPUT	INT	1, 2 to 16 for S7300 1, 2 to 64 for S7400	The connection number of the connection is specified in the parameter ID.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.

Parameter	Declaration	Data type	Range of values	Description
LOCKED	OUTPUT	BOOL	0: not (yet) locked 1: locked	Shows the status of the access lock requested on the specified FETCH/WRITE connection.
STATUS	OUTPUT	WORD		Status code For the meaning, see STATUS parameter (Page 6518)

See also

AG_LOCK (Page 6517)

STATUS parameter

Condition codes

The following table shows the STATUS code that must be evaluated by the user program.

Table 4-347 AG_LOCK condition codes

STATUS	Meaning
7000_H	CP is not processing a job
7001_H	FETCH active
7002_H	WRITE active
8183_H	FETCH/WRITE not configured for this connection (S7-400 only)
8186_H	ID number not in permitted range (e.g. 1...64 for S7-400 Industrial Ethernet CPs)
$80A4_H$	The communication bus connection between the CPU and CP is not established (with newer CPU versions).
$80B0_H$	The module does not recognize the data record.
$80B1_H$	The specified length (in the LEN parameter) is incorrect.
$80B2_H$	The communication bus connection between the CPU and CP is not established.
$80C0_H$	The data record cannot be read.
$80C1_H$	The specified data record is currently being processed.
$80C2_H$	There are too many jobs pending.
$80C3_H$	CPU resources (memory) occupied.
$80C4_H$	Communications error Occurs temporarily and a repetition in the user program will often remedy the problem.
$80D2_H$	Module start address incorrect.

See also

AG_LOCK (Page 6517)

AG_UNLOCK

AG_UNLOCK

Description

With the aid of the AG_UNLOCK instruction, you enable external access to user memory areas of the S7-CPU. With FETCH or WRITE, access via the connection selected with the ID parameter is then possible.

The AG_UNLOCK follows an access lock with AG_LOCK.

Call

Call interface in FBD representation

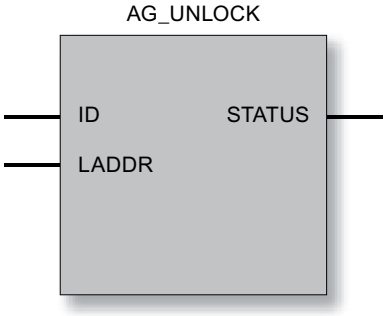

How It works

To release the connection again, the instruction must clear the LOCK request bit again. The instruction also shows the current status using error messages.

See also

STATUS parameter (Page 6520)

Parameters for AG_UNLOCK (Page 6520)

Parameters for AG_UNLOCK

Explanation of the formal parameters

The following table explains all the formal parameters for AG_UNLOCK:

Parameter	Declaration	Data type	Range of values	Description
ID	INPUT	INT	1, 2 to 16 for S7300 1, 2 to 64 for S7400	The connection number of the connection is specified in the parameter ID. (See Configuration)
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.
STATUS	OUTPUT	WORD		Status code: For the meaning, see STATUS parameter (Page 6520)

See also

AG_UNLOCK (Page 6519)

STATUS parameter

Condition codes

The following table shows the STATUS code that must be evaluated by the user program.

Table 4-348 AG_UNLOCK condition codes

STATUS	Meaning
7000_H	CP is not processing a job
7001_H	FETCH active
7002_H	WRITE active
8183_H	FETCH/WRITE not configured for this connection (S7-400 only)
8186_H	ID number not in permitted range (e.g. 1...64 for S7-400 Industrial Ethernet CPs)
$80A4_H$	The communication bus connection between the CPU and CP is not established (with newer CPU versions).
$80B0_H$	The module does not recognize the data record.
$80B1_H$	The specified length (in the LEN parameter) is incorrect.
$80B2_H$	The communication bus connection between the CPU and CP is not established.
$80C0_H$	The data record cannot be read.
$80C1_H$	The specified data record is currently being processed.
$80C2_H$	There are too many jobs pending.
$80C3_H$	CPU resources (memory) occupied.
$80C4_H$	communications error Occurs temporarily and a repetition in the user program will often remedy the problem.
$80D2_H$	Module start address incorrect.

See also

AG_UNLOCK (Page 6519)

Instructions for connection diagnostics

AG_CNTEX

AG_CNTEX

Significance and how It works

With the AG_CNTEX instruction, it is possible run diagnostics on connections and to address devices using the ping command via the network. When necessary, you can initialize connection establishment again using AG_CNTEX.

Note

AG_CNTRL and AG_CNTEX

The AG_CNTEX program block provides expanded functionality compared with the AG_CNTRL program block.

All the functions of AG_CNTRL are included in AG_CNTEX and they can be used identically in the user program.

The following actions are possible by setting commands:

- Reading out connection information
  Based on status information, you can decide whether or not it would be useful to reset all or individual connections of the CP.
- Resetting configured connections
  You can reset individual connections or all connections of a CP.
- Aborting the active connection and establishing it again
- Reading out connection types configured on the CP (expanded functionality compared with AG_CNTRL)
- Send PING command (expansion compared with AG_CNTRL)
  You can check whether a specific node is reachable in the network.

The commands of the AG_CNTEX program block are permitted only for SEND/RECV connections based on the ISO / RFC / TCP / UDP protocols.

Note

Availability in the block library

If the program block AG_CNTEX is not yet available in the SIMATIC_NET_CP block library, install the current SIMATIC NET block library; you will find this under the following entry ID on the Internet:

8797900 (http://support.automation.siemens.com/WW/view/de/8797900)

Call interface

Call interface in FBD representation

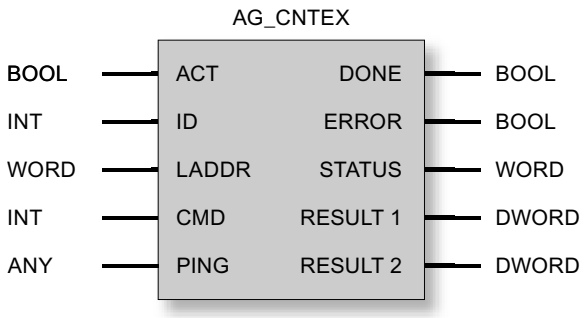

See also

How the PING function works (Page 6522)

Parameters for AG_CNTEX (Page 6524)

Parameters DONE, ERROR, STATUS (Page 6525)

Commands and job results - AG_CNTEX (Page 6526)

How the PING function works

Mode of operation / call sequence

With the CMD=8 PING command, you instruct the CP to send 4 successive PING requests over the network to the IP address specified in the job. The PING echo is expected by the CP within the period of time you set in the PING job field.

The CP registers the response times and enters these in the RESULT 1/2 parameters.

The RESULT 1/2 parameters can be queried with PING command CMD=9. As soon as the 4 PING requests have been replied to, or their set monitoring time has been exceeded, execution

is confirmed in the DONE=1 parameter. The PING result can then be queried within a maximum of 30 seconds; afterwards, the RESULT entries become invalid.

Note

PING is only possible over a configured connection

The PING command is only possible if at least **one** connection for the SEND/RECEIVE interface (TCP, ISO-on-TCP, ISO-Transport, UDP) is configured.

Several PING requests at the same time

You can send up to 4 PING requests at the same time to different IP addresses. To do this, you must use the same instance DB for the ping requests. Further PING requests are possible only after completion of at least one of the current PING requests.

If too many PING requests are sent at the same time, an error message to this effect is output (STATUS parameter = $828A_H$).

When are PING requests completed?

PING requests count as being completed when one of the following conditions is met:

- The PING result was read out:
- The PING result was not read out but 30 seconds have elapsed since the Ping result was available.

See also

AG_CNTEX (Page 6521)

Commands and job results - AG_CNTEX (Page 6526)

Parameters for AG_CNTEX

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_CNTEX function:

Parameter	Declaration	Data type	Range of values	Description
ACT	INPUT	BOOL	0, 1	The FB must be called with ACT=1. If it is called with ACT=0, there is no function call and the block is exited again immediately.
ID	INPUT	INT	• 1, 2, .., n, or • 0	The connection number of the connection is specified in the parameter ID. The connection number can be found in the configuration. n is the maximum number of connections and is dependent on the product (S7-300 or S7-400). For a call that addresses all connections, 0 must be specified as the ID. This affects: • the functions CN_STATUS_ALL (CMD 3) and CN_RESET_ALL (CMD 4) • PING command with CMD 8 or CMD 9
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.
CMD	INPUT	INT		Command to FB AG_CNTEX
PING	INPUT	ANY		References a block of data (for example DB) that contains the data structure for the PING command. The block of data contains the IP address and optional information about the time monitoring and the number of bytes to be transferred in the PING request. Data structure, see below
DONE	OUTPUT	BOOL	0: Job still being processed or not yet triggered 1: Job done	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 6525) Note: If DONE=1, RESULT can be evaluated
ERROR	OUTPUT	BOOL	0: No error 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 6525)
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Parameters DONE, ERROR, STATUS (Page 6525)
RESULT1	OUTPUT	DWORD		Information returned to AG_CNTEX according to the command.
RESULT2	OUTPUT	DWORD		Part 2 of the information returned to AG_CNTEX according to the command.

PING block of data

The PING parameter references a block of data with the following data structure:

Parameter	Data type	Range of values	Note
IP address	ARRAY [1..4] of Byte		
TIMEOUT	INT	1..60000 ms	Can be specified as an option; default value = 1000 ms
Size	INT	1..1000 bytes	Can be specified as an option; default value = 32 bytes

See also

AG_CNTEX (Page 6521)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

The command results in the RESULT1/2 parameters must also be evaluated according to Commands and job results - AG_CNTEX (Page 6526).

Table 4-349 AG_CNTEX codes

DONE	ERROR	STATUS	Meaning
1	0	0000_H	A job (CMD) was transferred successfully to the CP (for example RESET) or a status was read successfully from the CP. The RESULT1/2 parameters can be evaluated.
0	0	0000_H	There has been no block call yet or the program block was called with ACT=0.
0	0	8181_H	Job active The block call must be repeated with the same parameters until DONE or ERROR is signaled.
0	1	8183_H	No configuration or the service has not yet started on the Ethernet CP.
0	1	8184_H	System error or wrong parameter type. The cause can be: • Data type of the ANY pointer not correct for the PING parameter. • The ANY pointer references an odd bit address.
0	1	8186_H	The ID parameter is invalid. The permitted ID depends on the selected command.
0	1	8187_H	The CMD parameter is invalid.
0	1	8090_H	Possible meanings: • No module with this module start address exists; • The program block being used does not match the system family being used (remember to use different program blocks for S7300 and S7400); • The function is not supported by this module.
0	1	8091_H	The module start address is not at a doubleword boundary.

DONE	ERROR	STATUS	Meaning
0	1	8092_H	The module start address is incorrect.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established. The corresponding CPU in the H system is in STOP mode.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	CPU resources (memory) occupied.
0	1	$80C4_H$	Communications error The error occurs temporarily; it is usually best to repeat the job in the user program.
0	1	8286_H	The value for the "Timeout" in the PING data block is outside the valid range of values.
0	1	8287_H	The IP address specified in the PING DB is reserved and therefore not permitted.
0	1	8288_H	The display occurs only with the PING result request command. Possible meanings: • The IP address was not reached by the PING command (for example because the CP was changed to STOP mode after the PING request command was sent); • The PING result has already been read out; • The PING result was not read out within the maximum time of 30 seconds.
0	1	8289_H	The data volume for the PING request has exceeded the permitted range (maximum 1000 bytes; see data structure for the PING command)
0	1	$828A_H$	There are already 4 ping requests being processed. New requests are only possible again after processing the existing requests.
0	1	$828B_H$	There is already a PING request being processed for the specified IP address. Use the PING result request to complete the current processing.

See also

AG_CNTEX (Page 6521)

Commands and job results - AG_CNTEX

Commands and evaluating the job results

The following table shows you the possible commands and the results that can be evaluated in the RESULT1/2 parameters.

Note

Command evaluation with older CP types or firmware versions

The commands described below are supported by the current CP types or firmware versions. You should also check the more detailed information under the following entry ID:

http://support.automation.siemens.com/WW/view/en/33414377 (http://support.automation.siemens.com/ww/view/en/33414377)

Table 4-350 Commands to FC AG_CNTRL

CMD	Meaning		
0	NOP – no operation The block executes without a job being sent to the CP.		
	RESULT (for CMD = 0)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	$0000\ 0001_H$	Executed without error
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning			
1	CN_STATUS – connection status This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	$0000\ 000*_H$	Bits 0-3: Codes for the send direction (excluded values: 0x2)	
			Bit 0	Connection Type • 0: No send and receive connection • 1: Connection reserved for send and receive jobs
			Bit 1	Status of current job • 0: No send job being executed • 1: Send job being executed
			Bits 2+3	Previous job: • 00: No information available on previous send job • 01: previous send job completed successfully • 10: previous send job not completed successfully

CMD	Meaning			
1	CN_STATUS – connection status **(continued for CMD=1)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 00*0_H	Bits 4−7: Codes for the receive direction (excluded values: 0x2)	
			Bit 4	Connection Type • 0: No send and receive connection • 1: Connection reserved for send and receive jobs
			Bit 5	Status of current job • 0: No receive job being executed • 1: Receive job being executed
			Bits 6+7	Previous job: • 00: No information available on previous receive job • 01: previous receive job completed successfully • 10: previous receive job not completed successfully

CMD	Meaning			
1	CN_STATUS – connection status **(continued for CMD=1)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 0*00$_H$	Bits 8-11: Codes for FETCH/WRITE (excluded values: 0x3, 0x7, 0x8, 0xB, 0xF)	
			Bit 8	Connection type: • 0: No FETCH connection • 1: Connection reserved for FETCH jobs
			Bit 9	Connection type: • 0: No WRITE connection • 1: Connection reserved for WRITE jobs
			Bit 10	Job status (FETCH/WRITE): • 0: Job status OK • 1: Job status NOT OK This ID is set in the following situations: – The job was acknowledged negatively by the CPU – The job could not be forwarded to the CPU because the connection was in the "LOCKED" status. – The job was rejected because the FETCH/WRITE header did not have the correct structure.
			Bit 11	Status of FETCH/WRITE job • 0: No job active • 1: Job from LAN active

CMD	Meaning			
1	CN_STATUS – connection status **(continued for CMD=1)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 *000$_H$	Bits 12-15: General CP information (excluded values: 0x3, 0xB)	
			Bit 12 + 13	Information on connection status: (only available for SEND/RECV connections based on the ISO/RFC/TCP protocols, with UDP, the corresponding internal information is output) • 00: Connection is terminated • 01: Connection establishment active • 10: Connection termination active • 11: Connection is established
			Bit 14	CP information: • 0: CP in STOP • 1: CP in RUN
			Bit 15	Reset ID • 0: AG_CNTEX has not yet reset a connection or the reset ID was cleared. • 1: AG_CNTEX caused a connection reset.
	RESULT1	**** 0000$_H$		Bits 16-31: Reserved 0 – reserved for later expansions
	RESULT2	0000 0000$_H$		- reserved for later expansions

CMD	Meaning		
2	CN_RESET – connection reset This command resets the connection selected with ID. The CP is selected in the LADDR parameter. Resetting the connection means that a connection is aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted. An entry is also generated in the diagnostics buffer in which the job result can be found.		
	RESULT (for CMD = 2)		Meaning
	Parameter	Hex value/range	
	RESULT1	$0000\ 0001_H$	The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment were triggered.
		$0000\ 0002_H$	The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning		
3	CN_STATUS_ALL – all connections status This command returns the connection status of all connections (established/terminated) in the RESULT1/2 parameters (at total of 8 bytes of group information). The ID parameter must be set to "0" (checked for 0). The CP is selected in the LADDR parameter. When necessary, you can obtain detailed information about a terminated or unconfigured connection using a further connection status call with CMD=1.		
	RESULT (for CMD = 2)		Meaning
	Parameter	Hex value/range	
	RESULT1	$**** ****_H$ 32 bits with the following validity: • For S7-400: Bits 0-31 for connections 1 - 32 • For S7-300: Bits 0-15 for connections 1 - 16	For the relevant connection: • 0 – connection terminated / not configured • 1 – connection established
	RESULT2	$**** ****_H$ 32 bits with the following validity: • For S7-400: Bits 0-31 for connections 33 - 64	For the relevant connection: • 0 – connection terminated / not configured • 1 – connection established

CMD	Meaning		
4	CN_RESET_ALL – all connections reset This command resets all connections. The ID parameter must be set to "0" (checked for 0). The CP is selected in the LADDR parameter. Resetting the connections means that connections are aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted. An entry is also generated in the diagnostics buffer in which the job result can be found.		
	RESULT (for CMD = 4)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	$0000\ 0001_H$	The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment of all connections were triggered.
	RESULT1	$0000\ 0002_H$	The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning		
5	CN_CLEAR_RESET - Clear the reset ID This command resets the reset ID (bit 15 in RESULT1) for the connection selected with ID. The CP is selected in the LADDR parameter. This job executes automatically when the connection status is read (CMD=1); the separate job described here is therefore only required in special situations.		
	RESULT (for CMD = 5)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	$0000\ 0001_H$	The clear job was transferred to the CP successfully.
	RESULT1	$0000\ 0002_H$	The Clear job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning		
6	CN_DISCON − connection disconnect This command resets the connection selected with ID and LADDR. Resetting the connection is achieved by aborting the connection. Any data in the stack is lost without any message being displayed. The connection is not established again automatically afterwards. The connection can be established again with the CN_STARTCON control job. A diagnostics buffer entry is created in which you will find the job result.		
	RESULT (for CMD = 6)		Meaning
	Parameter	Hex value/range	
	RESULT1	0000 0001_H	The job was transferred to the CP successfully. The connection abort was initiated.
	RESULT1	0000 0002_H	The job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	0000 0000_H	Default

CMD	Meaning		
7	CN_STARTCON − start connection This command establishes a connection selected with ID and LADDR and aborted earlier with the control job CN_DISCON. A diagnostics buffer entry is created in which you will find the job result.		
	RESULT (for CMD = 7)		Meaning
	Parameter	Hex value/range	
	RESULT1	0000 0001_H	The connection establishment job was transferred to the CP successfully. The connection establishment was initiated.
	RESULT1	0000 0002_H	The connection establishment job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	0000 0000_H	Default

CMD	Meaning		
8	PING_REQUEST - Send a PING request This command sends a PING command to the CP. The CP then initiates 4 PING echo requests to the specified IP address.		
	RESULT (for CMD = 8)		Meaning
	Parameter	Hex value/range	
	RESULT1	0000 0001_H	The ping request was sent to the CP successfully.
	RESULT2	0000 0002_H	The ping request could not be sent to the CP because the corresponding service was not available on the CP. A possible cause might, for example, be: CP in STOP mode

CMD	Meaning		
9	PING_RESULT - Query ping result This command sends a PING result request to the CP. The CP transfers the results of the 4 executed PING echo requests in the RESULT parameter. The call is successful when the 4 ping echo requests have been completed on the part of the CP.		
	RESULT (for CMD = 9)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	**** ****$_H$	1. Word: Reply time in ms for the 1st PING echo request. 2. Word: Reply time in ms for the 2nd PING echo request. Example: 0005 FFFF$_H$ Echo 1 -> received after 5 ms Echo 2 -> no echo in the set monitoring time
	RESULT2	**** ****$_H$	1. Word: Reply time in ms for the 3rd PING echo request. 2. Word: Reply time in ms for the 4th PING echo request. Example: 0002 3456$_H$ Echo 3 -> received after 2 ms Echo 4 -> received after 13398 ms
	Range of values for data words in RESULT1 / RESULT 2:		
		0000$_H$	not used
		0001$_H$ ... EA60$_H$	Reply time in ms 0001$_H$ = 1 ms EA60$_H$ = 60000 ms
		EA61$_H$ ... FFFE$_H$	not used
		FFFF$_H$	Timeout: No echo within the specified monitoring time.

CMD	Meaning		
10	CONN_TYPE - Connection type This command requests the CP to specify the current connection type for the specified connection ID.		
	RESULT (for CMD = 10)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	0000 000*$_H$	The request returns the following values for the possible connection types: 0: no connection set up 1: UDP connection 2: SMTP connection 3: TCP connection 4: Free UDP connection 5: FTP connection 6: ISO transport connection 7: ISOonTCP connection
	RESULT2	0000 0000$_H$	Default

See also

You will find further information in the FAQ under the following entry ID: http://support.automation.siemens.com/WW/view/en/33414377 (http://support.automation.siemens.com/ww/view/en/33414377)

See also

AG_CNTEX (Page 6521)

How the PING function works (Page 6522)

AG_CNTRL for connection diagnostics

AG_CNTRL

Description

With the AG_CNTRL instruction, you can diagnose connections. When necessary, you can initialize connection establishment again using the instruction.

Note

AG_CNTRL and AG_CNTEX

The AG_CNTEX instruction provides expanded functionality compared with the AG_CNTRL instruction.

All the functions of AG_CNTRL are included in AG_CNTEX and they can be used identically in the user program.

The following actions are possible by setting commands:

- Reading out connection information
  Based on status information, you can decide whether or not it would be useful to reset all or individual connections of the CP.
- Resetting configured connections
  You can reset individual connections or all connections of a CP.
- Aborting the active connection and establishing it again

The commands of the AG_CNTRL instruction are permitted only for SEND/RECV connections based on the ISO / RFC / TCP / UDP protocols.

Call interface

Call interface in FBD representation

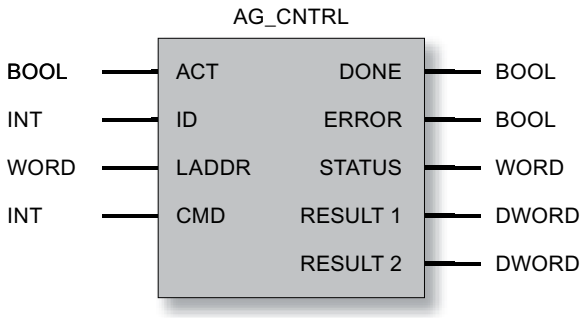

Further information

FAQ under entry ID 33414377 (http://support.automation.siemens.com/WW/view/de/33414377)

See also

Parameters for AG_CNTRL (Page 6537)
Parameters DONE, ERROR, STATUS (Page 6538)
Commands and job results for AG_CNTRL (Page 6539)

Parameters for AG_CNTRL

Explanation of the formal parameters

The following table explains all the formal parameters for AG_CNTRL:

Parameter	Declaration	Data type	Range of values	Description
ACT	INPUT	BOOL	0, 1	AG_UNLOCK must be called with ACT=1. If it is called with ACT=0, there is no function call and the instruction is exited again immediately.
ID	INPUT	INT	• 1, 2, .., n, or • 0	The connection number of the connection is specified in the parameter ID. The connection number can be found in the configuration. n is the maximum number of connections and is dependent on the product (S7-300 or S7-400). If the call addresses all connections (_ALL function with CMD 3 or 4), 0 must be specified as the ID.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.
CMD	INPUT	INT		Command to AG_CNTRL.
DONE	OUTPUT	BOOL	0: Job still being processed or not yet triggered 1: Job done	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 6538) Note: If DONE=1, RESULT can be evaluated
ERROR	OUTPUT	BOOL	0: No error 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 6538)
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Parameters DONE, ERROR, STATUS (Page 6538)
RESULT1	OUTPUT	DWORD		Information returned according to the command sent to AG_CNTRL.
RESULT2	OUTPUT	DWORD		Only to be evaluated for S7-400: Part 2 of the information returned according to the command sent to AG_CNTRL.

See also

AG_CNTRL (Page 6536)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

The command results in the RESULT1/2 parameters must also be evaluated according to "Commands and job results for AG_CNTRL (Page 6539)".

Table 4-351 AG_CNTRL codes

DONE	ERROR	STATUS	Meaning
1	0	0000_H	A job (CMD) was transferred successfully to the CP (for example RESET) or a status was read successfully from the CP. The RESULT1/2 parameters can be evaluated.
0	0	0000_H	There has been no instruction call yet or the instruction was called with ACT=0.
0	0	8181_H	Job active The instruction call must be repeated with the same parameters until DONE or ERROR is signaled.
0	1	8183_H	No configuration or the service has not yet started on the Ethernet CP.
0	1	8186_H	The ID parameter is invalid. The permitted ID depends on the selected command; see CMD parameter in "Commands and job results for AG_CNTRL (Page 6539)"
0	1	8187_H	The CMD parameter is invalid.
0	1	8188_H	Sequence error in the ACT control (Note: this code does not occur in the product version of the CP / firmware).
0	1	8189_H	The CP version / firmware used does not support AG_CNTRL. The code is set when you call a CP 3431-EX20 with firmware as of V1.3.9; with other CP types, the code 80B0H is set instead. Note: AG_CNTRL in version V1.0 is supported by the CPs as of CP 343-1EX21/GX21; this code does not occur with these modules.
0	1	8090_H	• No module with this module start address exists. or • The instruction being used does not match the system family being used (remember to use different instructions for S7300 and S7400). or • The function is not supported by this module.
0	1	8091_H	The module start address is not at a doubleword boundary.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established. The corresponding CPU in the H system is in STOP mode.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.

DONE	ERROR	STATUS	Meaning
0	1	$80C3_H$	CPU resources (memory) occupied.
0	1	$80C4_H$	Communications error The error occurs temporarily; it is usually best to repeat the job in the user program.
0	1	$80D2_H$	The module start address is incorrect.

See also

AG_CNTRL (Page 6536)

Commands and job results for AG_CNTRL

Commands and evaluating the job results

The following table shows you the possible commands and the results that can be evaluated in the RESULT1/2 parameters.

Table 4-352 Commands to AG_CNTRL

CMD	Meaning		
0	NOP – no operation AG_CNTRL executes without a job being sent to the CP.		
	RESULT (for CMD = 0)		Meaning
	Parameter	Hex value/range	
	RESULT1	$0000\ 0001_H$	Executed without error
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning			
1	CN_STATUS – connection status This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 000*$_H$	Bits 0-3: Codes for the send direction (excluded values: 0x2)	
			Bit 0	Connection Type • 0: No send and receive connection • 1: Connection reserved for send and receive jobs
			Bit 1	Status of current job • 0: No send job being executed • 1: Send job being executed
			Bits 2+3	Previous job: • 00: No information available on previous send job • 01: previous send job completed successfully • 10: previous send job not completed successfully

CMD	Meaning			
1	CN_STATUS – connection status **(CMD=1 continued)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 00*0$_H$	Bits 4−7: Codes for the receive direction (excluded values: 0x2)	
			Bit 4	Connection Type • 0: No send and receive connection • 1: Connection reserved for send and receive jobs
			Bit 5	Status of current job • 0: No receive job being executed • 1: Receive job being executed
			Bits 6+7	Previous job: • 00: No information available on previous receive job • 01: previous receive job completed successfully • 10: previous receive job not completed successfully

CMD	Meaning			
1	CN_STATUS – connection status **(CMD=1 continued)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 0*00$_H$	Bits 8-11: Codes for FETCH/WRITE (excluded values: 0x3, 0x7, 0x8, 0xB, 0xF)	
			Bit 8	Connection type: • 0: No FETCH connection • 1: Connection reserved for FETCH jobs
			Bit 9	Connection type: • 0: No WRITE connection • 1: Connection reserved for WRITE jobs
			Bit 10	Job status (FETCH/WRITE): • 0: Job status OK • 1: Job status NOT OK This ID is set in the following situations: – The job was acknowledged negatively by the CPU – The job could not be forwarded to the CPU because the connection was in the "LOCKED" status. – The job was rejected because the FETCH/WRITE header did not have the correct structure.
			Bit 11	Status of FETCH/WRITE job • 0: No job active • 1: Job from LAN active

CMD	Meaning			
1	CN_STATUS – connection status **(CMD=1 continued)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 *000$_H$	Bits 12-15: General CP information (excluded values: 0x3, 0xB)	
			Bit 12 + 13	Information on connection status: (only available for SEND/RECV connections based on the ISO/RFC/TCP protocols, with UDP, the corresponding internal information is output) • 00: Connection is terminated • 01: Connection establishment active • 10: Connection termination active • 11: Connection is established
			Bit 14	CP information: • 0: CP in STOP • 1: CP in RUN
			Bit 15	Reset ID • 0: AG_CNTRL has not yet reset a connection or the reset ID was cleared. • 1: The control instruction caused a connection reset.
	RESULT1	**** 0000$_H$		Bits 16-31: Reserved 0 – reserved for later expansions
	RESULT2	0000 0000$_H$		- reserved for later expansions

CMD	Meaning		
2	CN_RESET – connection reset This command resets the connection selected with ID. The CP is selected in the LADDR parameter. Resetting the connection means that a connection is aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted. An entry is also generated in the diagnostics buffer in which the job result can be found.		
	RESULT (for CMD = 2)		Meaning
	Parameter	Hex value/range	
	RESULT1	$0000\ 0001_H$	The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment were triggered.
		$0000\ 0002_H$	The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning		
3	CN_STATUS_ALL – all connections status This command returns the connection status of all connections (established/terminated) in the RESULT1/2 parameters (at total of 8 bytes of group information). The ID parameter must be set to "0" (checked for 0). The CP is selected in the LADDR parameter. When necessary, you can obtain detailed information about a terminated or unconfigured connection using a further connection status call with CMD=1.		
	RESULT (for CMD = 3)		Meaning
	Parameter	Hex value/range	
	RESULT1	$**** ****_H$	32 bits: Connection 1 - 32 • 0 – connection terminated / not configured • 1 – connection established
	RESULT2	$**** ****_H$	32 bits: Connection 33 - 64 • 0 – connection terminated / not configured • 1 – connection established

CMD	Meaning		
4	CN_RESET_ALL – all connections reset: This command resets all connections. The ID parameter must be set to "0" (checked for 0). The CP is selected in the LADDR parameter. Resetting the connections means that connections are aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted. An entry is also generated in the diagnostics buffer in which the job result can be found.		
	RESULT (for CMD = 4)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	$0000\ 0001_H$	The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment of all connections were triggered.
	RESULT1	$0000\ 0002_H$	The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning		
5	CN_CLEAR_RESET – Clear the reset ID This command resets the reset ID (bit 15 in RESULT1) for the connection selected with ID. The CP is selected in the LADDR parameter. This job executes automatically when the connection status is read (CMD=1); the separate job described here is therefore only required in special situations.		
	RESULT (for CMD = 5)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	$0000\ 0001_H$	The clear job was transferred to the CP successfully.
	RESULT1	$0000\ 0002_H$	The Clear job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning		
6	CN_DISCON − connection disconnect This command resets the connection selected with ID and LADDR. Resetting the connection is achieved by aborting the connection. Any data in the stack is lost without any message being displayed. The connection is not established again automatically afterwards. The connection can be established again with the CN_STARTCON control job. A diagnostics buffer entry is created in which you will find the job result.		
	RESULT (for CMD = 6)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	$0000\ 0001_H$	The job was transferred to the CP successfully. The connection abort was initiated.
	RESULT1	$0000\ 0002_H$	The job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning		
7	CN_STARTCON − start connection This command establishes a connection selected with ID and LADDR and aborted earlier with the control job CN_DISCON. A diagnostics buffer entry is created in which you will find the job result.		
	RESULT (for CMD = 6)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	$0000\ 0001_H$	The connection establishment job was transferred to the CP successfully. The connection establishment was initiated.
	RESULT1	$0000\ 0002_H$	The connection establishment job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

Further information

You will find further information in the FAQ under the following entry ID: 33414377 (http://support.automation.siemens.com/WW/view/en/33414377)

See also

AG_CNTRL (Page 6536)

Instructions for PROFINET IO (S7-300)

Instructions for PROFINET IO (S7-300)

Overview

The instructions listed below are available for transferring data cyclically on the PROFINET IO interface. The significance of the instructions differs depending on how you use the CP (as a PROFINET IO controller or PROFINET IO device) in an S7 station.

Instruction	can be used with:		Meaning
S7-300	S7-400		
PNIO_SEND	x	-	Depending on the mode of the CP: • On a PROFINET IO controller Sending output data to the PROFINET IO devices. • On a PROFINET IO device Forwarding process input data to the PROFINET IO controller.
PNIO_RECV	x	-	Depending on the mode of the CP: • On a PROFINET IO controller Receiving process input data from the PROFINET IO devices. • On a PROFINET IO device Receiving process output data from the PROFINET IO controller.

For CPs operating as PROFINET IO controller and IO device at the same time, the instructions are available as of version 2.0.

The instructions listed below are available for transferring data (data records, interrupt information) acyclically on the PROFINET IO interface. The two instructions can only be used in PROFINET IO controller mode.

Instruction	can be used with:		Meaning
S7-300	S7-400		
PNIO_RW_REC	x	-	• Read data record (from a PROFINET IO device) • Write data record (to a PROFINET IO device)
PNIO_ALARM	x	-	Receive alarm information from the PROFINET IO devices

Instructions for cyclic transmission of user data (S7-300)

PNIO_SEND (S7-300)

PNIO_SEND (S7-300)

Description

The PNIO_SEND instruction is used to transfer data in the PROFINET IO controller or PROFINET IO device modes of the CP.

- Operating as PROFINET IO controller
  The instruction transfers the process data (outputs) of a specified output area to the CP to be forwarded to PROFINET IO devices and as status display returns the IO consumer status (IOCS) of the outputs of the PROFINET IO devices.
- Operation as PROFINET IO device
  The instruction reads the preprocessed inputs of the CPU on the PROFINET IO device and transfers them to the PROFINET IO controller (configured I-addresses). In addition to this, the instruction also returns the IO Consumer Status (IOCS) of the PROFINET IO controller as a status code.

The preprocessed process data is available in a DB or bit memory area.

For CPs operating as PROFINET IO controller and IO device at the same time, the instruction is available as of version 2.0. With the additional MODE parameter, you set the mode for which the instruction will be called.

Call interface (version 2.0)

Call interface in FBD representation

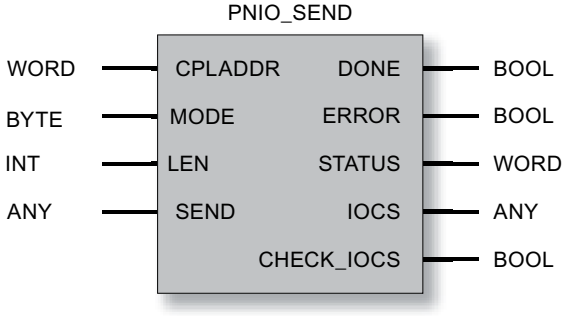

See also

General characteristics of the instructions for PROFINET IO (Page 6559)

Data consistency (Page 6560)

Parameters for PNIO_SEND (Page 6548)

Parameters DONE, ERROR, STATUS (IE) (Page 6552)

Parameters for PNIO_SEND (S7-300)

Explanation of the formal parameters

The following table explains all the formal parameters for the PNIO_SEND function:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	-	Module start address
MODE (parameters version 2.0 or later)	INPUT	BYTE	Values for X: • $0Y_H$: Status bits are transferred in IOCS. • $8Y_H$: Restriction to group message in CHECK_IOCS; no status bits in IOCS. Values for Y: • $X0_H$: – IO controller mode – IO device mode (without parallel operation) There is compatibility with the instruction in version 1.0. • $X1_H$: IO device mode (with parallel operation)	Specified in the form XY (hexadecimal): • X sets the transfer of status information. • Y specifies the mode of the CP as IO controller or IO device. Notes on compatibility;: • The version 1.0 instruction can continue to be used as long as the CP is not being operated as an IO controller and IO device at the same time. • When MODE=0, the instruction as of version 2.0 behaves like the instruction version 1.0.

Parameter	Declaration	Data type	Range of values	Description
SEND	IN_OUT	ANY (as VARTYPE only BYTE is permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area	Specifies the address and length **IO controller mode:** The length should match the total length of the distributed IO configured, whereby address gaps are also transmitted. The length can also be shorter than the total length of the distributed IO, for example when the instruction is called more than once in one OB. It must, however, have the total length in at least one call. **IO device mode:** The data structure results from the order of the slots of the input modules configured for this PROFINET IO device on the PROFINET IO controller line and their length without address gaps. (Please note the more extensive explanations or examples for your CP in the devicespecific Part B of this manual) Notes: • The instruction begins to transfer the data at address 0 regardless of how you configured the addresses (regardless of the lowest configured address). • Specifying an I/O area is not permitted since you must first check the IOCS for GOOD before data can be accepted in the I/O.

Parameter	Declaration	Data type	Range of values	Description
LEN	INPUT	INT	Value > 0 The maximum total length of the data areas to be transferred can be found in the devicespecific Part B of the device manual in the "Performance data" chapter. This may differ for controller or device mode.	Length of the data area to be transferred in bytes. The transfer of the data always begins with address 0 regardless of the configuration. Please note that the IO address "0" with a length of 1 is included. **IO controller mode:** • The highest configured address of the devices must be specified here. The individual areas are not grouped together. If the instruction is called more than once, LEN can also be shorter than the highest address. The highest address should be specified in at least one call (compare "SEND" parameter). • The data is transferred in the order of the logical addresses (as with PROFIBUS DP). **IO device mode:** • The data is transferred in the order of the slots corresponding to the configuration of the input modules on the PROFINET IO controller line for this PROFINET IO device. Note: Make sure that the length programmed here and the configuration of the PROFINET IO controller are consistent. The entire data area length including any gaps is transferred for the device.
DONE	OUTPUT	BOOL	0: - 1: New data accepted	This parameter indicates whether or not the job was completed without errors.
ERROR	OUTPUT	BOOL	0: -1: Error	Error code
STATUS	OUTPUT	WORD	-	Status code

Parameter	Declaration	Data type	Range of values	Description
CHECK_IOCS	OUTPUT	BOOL	0: All IOCS set to GOOD 1: At least one IOCS set to BAD	Auxiliary bit that indicates whether or not it is necessary to evaluate the IOCS status area.
IOCS	OUTPUT	ANY (as VARTYPE only BYTE is permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area Length: For the maximum value, refer to the devicespecific Part B of this manual in the section "Performance data". This may differ for controller or device mode.	A status bit is transferred per byte of user data. The length information depends on the length in the LEN parameter (one bit per byte) = (Length LEN + 7/ 8) Controller mode: Address gaps are also transferred according to the SEND parameter. Address gaps are transferred with the status GOOD. Device mode: Address gaps are not transferred. The instruction begins the transfer of the status for address 0. Note: The minimum length of the ANY pointer is (length LEN + 7/8)

Note

Remember that all output parameters may only be evaluated when the instruction signals either DONE = 1 or ERROR = 1.

Note

You must assume that the returned IOCS status does not arrive timesynchronized with the data (SEND parameter) but delayed by one user program cycle. This means: User data and IOCS are not consistent.

See also

PNIO_SEND (Page 6547)

Parameters DONE, ERROR, STATUS (IE) (S7-300)

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

To find out which system instructions are used and are relevant for error evaluation, display the properties dialog of the instruction described here in the "Calls" tab.

Table 4-353 Condition codes PNIO_SEND

DONE	ERROR	STATUS	Meaning
0	0	8180_H	• Data transfer active; or • The CP is in STOP mode.
0	0	8181_H	Module does not support version 2.0 of the instruction Solution: Use version 1.0.
1	0	0000_H	New data transferred without error.
0	1	8183_H	• PROFINET IO configuration missing; or • wrong CPLADDR; or • The CP is in STOP mode. or • Interconnection of MODE does not match module configuration or incorrect interconnection with MODE > 1 Extra in device mode: • The connection between PROFINET IO controller and PROFINET IO device is down, or • PROFINET IO controller not reachable or • Total lengths (configuration and LEN parameter) are not consistent.
0	1	8184_H	System error or bad parameter type.
0	1	8185_H	Parameter LEN is greater than source area SEND or target buffer (IOCS) is too small.
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Range error when reading a parameter.
0	1	$8F25_H$	Range error when writing a parameter.

DONE	ERROR	STATUS	Meaning
0	1	$8F28_H$	Alignment error when reading a parameter.
0	1	$8F29_H$	Alignment error when writing a parameter.
0	1	$8F30_H$	Parameter is in the write-protected 1st current data block.
0	1	$8F31_H$	Parameter is in the write-protected 2nd current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F3A_H$	Destination area is not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented.
0	1	$8F45_H$	Access to a parameter to be written during execution of the instruction is prevented.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference.
0	1	8090_H	Module with this address does not exist.
0	1	$80A0_H$	Negative acknowledgment writing to the module.
0	1	$80A1_H$	Negative acknowledgment writing to the module.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	• The specified data record length is incorrect. or • The CP changes to STOP.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).

See also

PNIO_SEND (Page 6547)

PNIO_RECV (S7-300)

PNIO_RECV (S7-300)

Description

The PNIO_RECV instruction is used to receive data in the PROFINET IO controller or PROFINET IO device modes of the CP.

- Operating as PROFINET IO controller
  The instruction accepts the process data from PROFINET IO devices (inputs of the controller) and transfers the IO provider status (IOPS) from the PROFINET IO devices to the specified input areas.
- Operating as PROFINET IO device
  The instruction receives the data transferred by the PROFINET IO controller (configured O-addresses) as well as the IO Provider Status (IOPS) of the PROFINET IO controller and writes them to the data areas on the CPU of the PROFINET IO device reserved for the process outputs.

For CPs operating as PROFINET IO controller and IO device at the same time, the instruction is available as of version 2.0. With the additional MODE parameter, you set the mode for which the instruction will be called.

Call interface (version 2.0)

Call interface in FBD representation

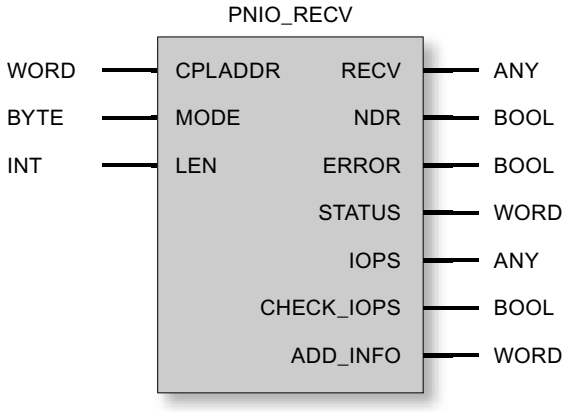

See also

General characteristics of the instructions for PROFINET IO (Page 6559)

Data consistency (Page 6560)

Parameters for PNIO_RECV (Page 6555)

Parameters NDR, ERROR, STATUS (IE) (Page 6558)

Parameters for PNIO_RECV (S7-300)

Explanation of the formal parameters

The following table explains all the formal parameters for the PNIO_RECV instruction:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	-	Module start address
MODE (parameters version 2.0 or later)	INPUT	BYTE	Values for X: • $0Y_H$: Status bits are transferred in IOCS. • $8Y_H$: Restriction to group message in CHECK_IOCS; no status bits in IOCS. Values for Y: • $X0_{H:}$ – IO controller mode – IO device mode (without parallel operation) Compatible with instruction in version 1.0 • $X1_H$: IO device mode (with parallel operation)	Specified in the form XY (hexadecimal): • X sets the transfer of status information. • Y specifies the mode of the CP as IO controller or IO device. Notes on compatibility;: • The version 1.0 instruction can continue to be used as long as the CP is not being operated as an IO controller and IO device at the same time. • When MODE=0, the instruction as of version 2.0 behaves like the instruction version 1.0.
RECV	IN_OUT	ANY (as VAR-TYPE only BYTE is permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area	Specifies the address and length IO controller mode: The length should match the total length of the distributed IO configured, whereby address gaps are also transmitted. The length can also be shorter than the total length of the distributed IO, for example when the instruction is called more than once in one OB. It must, however, have the total length in at least one call. IO device mode: The data structure results from the order of the slots of the output modules configured for this PROFINET IO device on the PROFINET IO controller line and their length without address gaps. Notes: • The instruction begins to transfer the data at address 0 regardless of how you configured the addresses (regardless of the lowest configured address). • Specifying an I/O area is not permitted since you must first change the IOPS for GOOD before data can be accepted in the I/O.

Parameter	Declaration	Data type	Range of values	Description
LEN	INPUT	INT	Value > 0 The maximum total length of the data to be transferred can be found in Part B of the device manual in the "Performance data" chapter. This may differ for controller or device mode.	Length of the data area to be transferred in bytes. The transfer of the data always begins with address 0 regardless of the configuration. Please note that the IO address "0" with a length of 1 is included. IO controller mode: • The highest configured address of the devices must be specified here. The individual areas are not grouped together. If the instruction is called more than once, LEN can also be shorter than the highest address. The highest address should be specified in at least one call (compare "RECV" parameter). • The data is transferred in the order of the logical addresses (as with PROFIBUS DP). IO device mode: • The data is transferred in the order of the slots corresponding to the configuration of the input modules on the PROFINET IO controller line for this PROFINET IO device. • Note: Make sure that the length programmed here and the configuration of the PROFINET IO controller are consistent. The entire data area length including any gaps is transferred for the device.
NDR	OUTPUT	BOOL	0: - 1: Data accepted	This parameter indicates whether or not the job was completed without errors.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code
STATUS	OUTPUT	WORD	-	Status code
CHECK_ IOPS	OUTPUT	BOOL	0: All IOPS set to GOOD 1: At least one IOPS set to BAD	Auxiliary bit that indicates whether or not it is necessary to evaluate the IOPS status area.

Parameter	Declaration	Data type	Range of values	Description
IOPS	OUTPUT	ANY (as VAR-TYPE only BYTE is permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area Length: For the maximum value, refer to Part B of the device manual in the section "Performance data". This may differ for controller or device mode.	A status bit is transferred per byte of user data. The length information depends on the length in the RECV parameter (one bit per byte) = (Length LEN + 7/ 8) Controller mode: Address gaps are also transferred according to the RECV parameter. Address gaps are transferred with the status GOOD. Device mode: Address gaps are not transferred. The instruction begins the transfer of the status for address 0. Note: • The minimum length of the ANY pointer is (length LEN + 7/8)
ADD_INFO	OUTPUT	WORD	Additional Diagnostic Information In controller mode: • 0: No alarm • >0: Number of pending alarms In device mode, the parameter is always = 0.	Parameter expansion Note: The ADD_INFO parameter is also updated when there are no INPUT addresses configured on the PROFINET IO controller. In this case, the PNIO_RECV instruction is called with a length LEN > 0 (for example LEN = 1 byte). It then transfers an address gap of 1 byte. The parameter expansion can be used for CPs as of the following firmware version: • CP 343−1 (EX30) as of firmware V2.0 • CP 343−1 Lean (CX10) as of firmware V2.0 • CP 343−1 Advanced (GX30) as of firmware V1.0 In older firmware versions, the parameter is reserved.

Note

Remember that all output parameters may only be evaluated when the instruction signals either NDR = 1 or ERROR = 1.

See also

PNIO_RECV (Page 6554)

Parameters NDR, ERROR, STATUS (IE) (S7-300)

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

To find out which system instructions are used and are relevant for error evaluation, display the properties dialog of the instruction described here in the "Calls" tab.

Table 4-354 Condition codes PNIO_RECV

NDR	ERROR	STATUS	Meaning
0	0	8180_H	• Data acceptance active; or • The CP is in STOP mode.
0	0	8181_H	Module does not support version 2.0 of the instruction. Solution: Use version 1.0.
1	0	0000_H	New data accepted without error.
0	1	8183_H	• PROFINET IO configuration missing; or • wrong CPLADDR; or • The CP is in STOP mode. or • Interconnection of MODE does not match module configuration or incorrect interconnection with MODE > 1. Extra in device mode: • The connection between PROFINET IO controller and PROFINET IO device is down or • PROFINET IO controller not reachable or • Total lengths (configuration and LEN parameter) are not consistent
0	1	8184_H	System error or bad parameter type.
0	1	8185_H	Destination buffer (RECV of IOCS) is too small.
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Range error when reading a parameter.

NDR	ERROR	STATUS	Meaning
0	1	$8F25_H$	Range error when writing a parameter.
0	1	$8F28_H$	Alignment error when reading a parameter.
0	1	$8F29_H$	Alignment error when writing a parameter.
0	1	$8F30_H$	Parameter is in the write-protected 1st current data block.
0	1	$8F31_H$	Parameter is in the write-protected 2nd current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F3A_H$	Destination area is not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented.
0	1	$8F45_H$	Access to a parameter to be written during execution of the instruction is prevented.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference.
0	1	8090_H	Module with this address does not exist.
0	1	$80A0_H$	Negative acknowledgment writing to the module.
0	1	$80A1_H$	Negative acknowledgment writing to the module.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	• The specified data record length is incorrect. or • The CP changes to STOP.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).

See also

PNIO_RECV (Page 6554)

General characteristics of the instructions for PROFINET IO (S7-300)

IO Consumer Status (IOCS) and IO Provider Status (IOPS)

For both communication partners - CPU/CP on the one hand and IO device on the other - there is the status information GOOD or BAD for the data. This status information is transferred parallel to the data. The status of the partner that sends the data is called IOPS (IO Provider Status), the status of the receiving partner is called IOCS (IO Consumer Status).

The IOPS and IOCS status are not necessarily identical. It is, for example, possible that the S7-300 CPU is in STOP mode (output disable or no PROFINET IO instructions active). In this case, the CP as PROFINET IO controller transfers the BAD status to the IO devices.

Relationship between instruction call and IO data

- Operation as PROFINET IO controller
  As a PROFINET IO controller, the CP does not monitor the cyclic calls of the PNIO_SEND/RECV instructions. If the instructions are not called, the last transferred IO data and IOCS/IOPS data are taken as valid.
- Operation as PROFINET IO device
  PNIO_SEND and PNIO_RECV each have their own watchdog. Depending on the CPU cycle time, the connection to the PROFINET IO controller is terminated if one of the two instructions is no longer called following the initialization phase.

Optimizing data transfer (only when operating as PROFINET IO controller)

It is possible to call the instructions with a length (LEN parameter) that is shorter than the configured total length of the IO data on the PNIO chain.

You can use this so that timecritical data is transferred in every CPU cycle whereas non critical data is not transferred in every cycle.

Example:
You could, for example, transfer only the first data area (timecritical data) in every cycle and the total length of the configured IO data in every second cycle. To do this, place the timecritical data in the lower area (starting at IO address 0) during configuration.

Data consistency (S7-300)

The entire input or output data area of the PROFINET IO controller is always transferred in its entirety and is therefore consistent.

- Operating as PROFINET IO controller
  Regardless of this, using the length information in the instruction call, you can also read or output an input or output area smaller than the configured area consistently.

Note: You should, however, bear in mind that in terms of the "IO user data" within a PROFINET IO system, data consistency can only be guaranteed within the individual IO slots. This applies regardless of the fact that consistent data transfer between CPU and IO controller is guaranteed for the instructions described here.

Instruction call

To guarantee data consistency, you may, however, only access the IO data when the instruction has completed free of errors (output parameter NDR = TRUE). You must also check that the IOCS or IOPS status for the data is GOOD.

Example

In a normal situation (depending on the total length of the IO data), the instruction will run over several user program cycles until the condition code DONE/NDR = 1 is signaled.

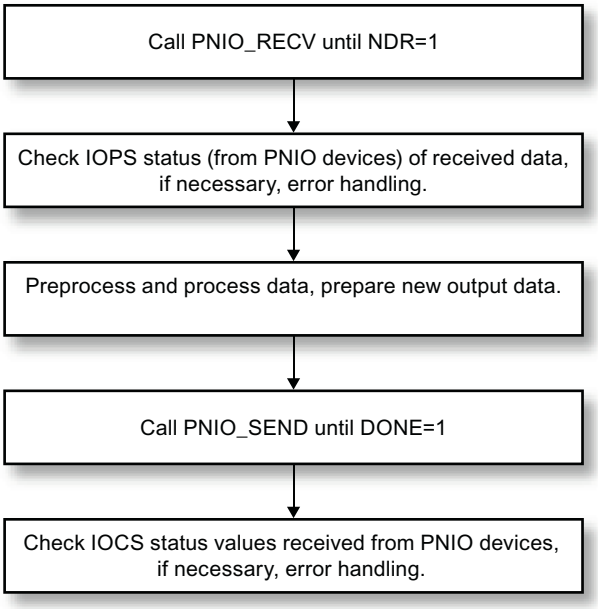

Note: The user program cycle and the cycle of the IO data exchange between the PROFINET IO controller and PROFINET IO devices are independent of each other.

Substitute values (S7-300)

Operational situations

The setting of substitute values is supported for the two following operational situations:

- Substitute values during startup (mode change on the CPU from STOP to RUN)
- Substitute values if problems are detected (remove/insert or station failure/return)

Substitute values during startup

You can initialize the outputs with substitute values by setting a memory bit ("startup" memory bit) in the startup OB. In cyclic mode (OB1), evaluate this "startup" memory bit to call the PNIO_SEND instruction with the initialization values when appropriate.

Substitute values if a problem occurs (only when operating as PROFINET IO controller)

If there is a fault (device/submodule failed), you can find out which submodules have failed by querying the status information IOCS / IOPS status. You then have the option of setting substitute values.

PNIO_RW_REC (S7-300)

PNIO_RW_REC (S7-300)

Description

RW_REC is used both for the "read data record" and the "write data record" function in PROFINET IO controller mode. RW_REC can only execute one of the functions at any one time. The "read data record" or "write data record" function is controlled by the WRITE_REC parameter.

Example: The CP can be informed of the location ID and plant designation using the "write data record" function (if this parameter was not already set in the properties dialog of the CP in STEP 7). This is done using the maintenance data record "IM1" with index AFF1H.

You will find details of the supported data records and their structure at the following Internet address:

`http://support.automation.siemens.com/WW/view/en/19289930`

Call interface

Call interface in FBD representation:

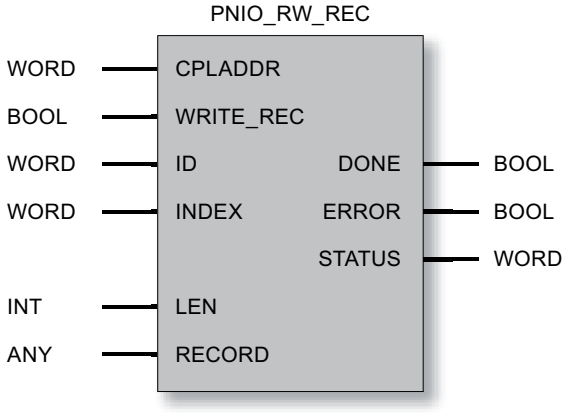

See also

Parameters DONE, ERROR, STATUS (IE) (Page 6564)

Parameters for PNIO_RW_REC (Page 6563)

Parameters for PNIO_RW_REC (S7-300)

Explanation of the formal parameters

The following table explains all the formal parameters for the PNIO_RW_REC instruction:

Parameter	Declaration	Data type	Possible values	Description
CPLADDR	INPUT	WORD	-	Module start address
WRITE_REC	INPUT	BOOL	0: Read data record 1: Write data record	Job type; The parameter must not be changed while the instruction is executing.
ID	INPUT	WORD		Logical address of the PROFINET IO component (module or submodule). For an output module, bit 15 is set (example of output address 5: ID:=DW#16#8005). For a mixed module, the lower of the two addresses must be specified.
INDEX	INPUT	WORD	See vendor information for the data record numbers supported by the module.	Data record number that the user wants to read or write.
DONE	OUTPUT	BOOL	0: - 1: Data record transferred successfully	This parameter indicates whether or not the job was completed without errors.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code
STATUS	OUTPUT	WORD	0: No error Different value: Error (see "Parameters DONE, ERROR, STATUS (IE) (Page 6564)")	Status code
LEN	IN_OUT	INT	The maximum length is 480 bytes.	• Read data record: OUTPUT parameter only; after a successful read, the length of the read data record is indicated; otherwise 0. • Write data record: INPUT parameter only; length of the data record to be written is entered here by the user. The length must match the definition of the data record.
RECORD	IN_OUT	ANY (as VARTYPE, BYTE, WORD and DWORD are permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area The length of the ANY pointer must be greater than or equal to the definition of the data record.	• Read data record: OUTPUT parameter only; after a successful read, the data of the data record is stored here. If the ANY pointer is too short, as much data as possible is transferred. • Write data record: INPUT parameter only; the data to be written from the data record is stored here by the user. The ANY pointer must be at least as long as specified in the LEN parameter.

See also

PNIO_RW_REC (Page 6562)

Parameters DONE, ERROR, STATUS (IE) (S7-300)

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries with the coding $8Fxx_H$ under STATUS, note the information in the Reference Manual "STEP 7 - System and Standard Functions for S7-300 and S7-400". The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 4-355 PNIO_RW_REC condition codes

DONE	ERROR	STATUS	Meaning
0	0	8180_H	Data transfer active
1	0	0000_H	Data record transferred successfully
0	1	8183_H	• No PROFINET IO controller configuration, • wrong CPLADDR or • CP in STOP mode
0	1	8184_H	System error or illegal parameter type
0	1	8185_H	Destination buffer (RECORD) is too short
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short)
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short)
0	1	$8F24_H$	Area error reading a parameter
0	1	$8F25_H$	Area error writing a parameter
0	1	$8F28_H$	Orientation error when reading a parameter
0	1	$8F29_H$	Alignment error writing a parameter
0	1	$8F30_H$	Parameter is in the write-protected first active data block
0	1	$8F31_H$	Parameter is in the write-protected second active data block
0	1	$8F32_H$	The DB number in the parameter is too high
0	1	$8F3A_H$	Destination area not loaded (DB)
0	1	$8F42_H$	Timeout reading a parameter from the I/O area
0	1	$8F43_H$	Timeout writing a parameter to the I/O area
0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented
0	1	$8F45_H$	Access to a parameter to be written during execution of the instruction is prevented
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference
0	1	8090_H	Module with this address does not exist

DONE	ERROR	STATUS	Meaning
0	1	$80A0_H$	Negative acknowledgment reading from the module
0	1	$80A1_H$	Negative acknowledgment writing to the module
0	1	$80A3_H$	General PROFINET IO context management error
0	1	$80A9_H$	PROFINET IO device or module reports an illegal type
0	1	$80B0_H$	Module does not recognize the data record
0	1	$80B1_H$	• The specified data record length is incorrect or • The CP changes to STOP
0	1	$80B2_H$	The logical address or the configured slot is not in use
0	1	$80B4_H$	PROFINET IO device or module signaling access to an illegal area
0	1	$80B6_H$	PROFINET IO device or module denies access
0	1	$80B8_H$	The module is signaling an illegal parameter
0	1	$80C0_H$	The data record cannot be read
0	1	$80C1_H$	The specified data record is being processed
0	1	$80C2_H$	Too many jobs pending
0	1	$80C3_H$	Resources (memory) occupied
0	1	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).

See also

PNIO_RW_REC (Page 6562)

PNIO_Alarm (S7-300)

PNIO_ALARM (S7-300)

Description

The PNIO_ALARM instruction is used for evaluation of alarms by a CP 343-1 operating as PROFINET IO controller and should be called in its user program if the parameter ADD_INFO does not equal 0 in PNIO_RECV. After full and error-free transfer of all OUTPUT parameters of the PNIO_ALARM, the received alarms are automatically acknowledged.

The alarms are forwarded to the user program in the chronological order in which they were signaled. Older alarms that have not yet been signaled to the user program and that become invalid due to more recent alarms are not deleted by the newer alarms.

Note

As long as PNIO_ALARM has not yet been called, the alarms are acknowledged automatically (internally) the CP.

If PNIO_ALARM has been called (at least) once in the user program, it must continue to be called to acknowledge pending alarms. This is the situation when the PNIO_RECV instruction signals a value not equal to "0" in the ADD_INFO parameter.

If PNIO_ALARM is no longer called after it has been called once or more in the user program, alarms are not acknowledged and there is no guarantee that the IO image will be updated correctly. The can occur, for example, following a station return alarm. The need to call PNIO_ALARM can only be reset by restarting the CP (power cycle).

Call interface

Call interface in FBD representation

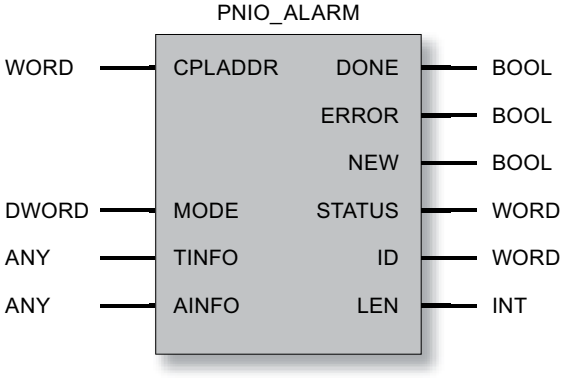

See also

Parameters DONE, NEW, ERROR, STATUS (IE) (Page 6568)

Parameters for PNIO_ALARM (Page 6567)

Parameters for PNIO_ALARM (S7-300)

Explanation of the formal parameters

The following table explains all the formal parameters for the PNIO_ALARM instruction:

Parameter	Declaration	Data type	Possible values	Description
CPLADDR	INPUT	WORD	-	Start address of the module that caused the error
DONE	OUTPUT	BOOL	0: - 1: Alarm information transferred successfully	This parameter indicates whether or not the job was completed without errors. If DONE = 1, the NEW parameter must also be checked.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code
NEW	OUTPUT	BOOL	0: Data transfer active or no new alarm 1: New alarm received and acknowledged	If DONE = 1 and NEW = 1, a new received alarm is signaled.
STATUS	OUTPUT	WORD	0: No error Different value: Error (see Table 1–14)	Status code
ID	OUTPUT	WORD		Logical start address of the PNIO component that triggers the alarm (module or submodule). For an output module, bit 15 is set (example of output address 5: ID:=DW#16#8005). For a mixed module, the lower of the two addresses is specified.
LEN	OUTPUT	INT		Length of the received alarm information (AINFO)
MODE	IN_OUT	DWORD	0	Reserved
TINFO	IN_OUT	ANY (as VARTYPE, BYTE, WORD and DWORD are permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area The length of the ANY pointer must be >= 32 bytes.	(task information) Destination area for the alarm management information. The error OB start information (OB header = byte 0 to 19 of TINFO) is reproduced as far as possible by the CP firmware. See also 1)
AINFO	IN_OUT	ANY (as VARTYPE, BYTE, WORD and DWORD are permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area The length of the ANY pointer must be greater than or equal to the maximum additional alarm information that can be expected, maximum 1432 bytes (see LEN parameter)	(alarm information) Destination area for header information and additional alarm information. If the ANY pointer AINFO is too low, the information will be truncated. See also 1)

1) Reference Manual "STEP 7 - System and Standard Functions for S7-300 and S7-400", receiving an alarm with the "RALRM" instruction

See also

PNIO_ALARM (Page 6565)

Parameters DONE, NEW, ERROR, STATUS (IE) (S7-300)

Condition codes

The following table shows the condition codes formed by the DONE, NEW, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries with the coding $8Fxx_H$ under STATUS, note the information in the Reference Manual "STEP 7 - System and Standard Functions for S7-300 and S7-400". The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 4-356 PNIO_ALARM status codes

DONE	NEW	ERROR	STATUS	Meaning
0	0	0	8180_H	Data transfer active
1	1	0	0000_H	Alarm data successfully transferred and alarm acknowledged
1	0	0	0000_H	No alarm data exist
0	0	1	8183_H	• No PROFINET IO controller configuration, • wrong CPLADDR or • CP in STOP mode
0	0	1	8184_H	System error or illegal parameter type
0	0	1	8185_H	Destination buffer (TINFO or AINFO) is too short
0	0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short)
0	0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short)
0	0	1	$8F24_H$	Area error reading a parameter
0	0	1	$8F25_H$	Area error writing a parameter
0	0	1	$8F28_H$	Orientation error when reading a parameter
0	0	1	$8F29_H$	Alignment error writing a parameter
0	0	1	$8F30_H$	Parameter is in the write-protected first active data block
0	0	1	$8F31_H$	Parameter is in the write-protected second active data block
0	0	1	$8F32_H$	The DB number in the parameter is too high
0	0	1	$8F3A_H$	Destination area not loaded (DB)
0	0	1	$8F42_H$	Timeout reading a parameter from the I/O area

DONE	NEW	ERROR	STATUS	Meaning
0	0	1	$8F43_H$	Timeout writing a parameter to the I/O area
0	0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented.
0	0	1	$8F45_H$	Access to a parameter to be written during execution of the instruction is prevented.
0	0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference
0	0	1	8090_H	Module with this address does not exist
0	0	1	$80A0_H$	Negative acknowledgment reading from the module
0	0	1	$80A1_H$	Negative acknowledgment writing to the module
0	0	1	$80B0_H$	Module does not recognize the data record
0	0	1	$80B1_H$	• The specified data record length is incorrect or • The CP changes to STOP
0	0	1	$80C0_H$	The data record cannot be read
0	0	1	$80C1_H$	The specified data record is being processed
0	0	1	$80C2_H$	Too many jobs pending
0	0	1	$80C3_H$	Resources (memory) occupied
0	0	1	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).

See also

PNIO_ALARM (Page 6565)

PROFIenergy (S7-300)

PROFIenergy

The PROFIenergy functions in PROFINET are used for energy management of plants. These include the planned or spontaneous shutdown of individual field devices, units or plant sections to save energy. The shutdown takes place during times without production or during breaks in production. Energy and diagnostics data can be read from devices included in the energy concept and that support these functions.

PROFIenergy controller

The commands for shutting down are output by the higher-level controller, in PROFINET IO, the IO controller.

With the SIMATIC S7-300, an S7-300 CPU with PROFIenergy functionality can be the PROFIenergy controller.

PROFIenergy devices

The commands of the PROFIenergy controller are processed by the IO devices with PROFIenergy functionality to shut down connected devices in the field.

In the context of the PROFIenergy program blocks, an IO device with PROFIenergy functionality is known as a PROFIenergy device. With the SIMATIC S7-300, an S7-300 CPU with PROFIenergy functionality can be the PROFIenergy device.

I-devices

In SIMATIC S7, an intelligent device (I-device) itself can have subordinate PROFIenergy devices. In this case, the I-device also has the function of a PROFIenergy controller.

Energy saving modes and PE_MODE_ID

Many devices support only the operating states "ready to operate" (power ON) and "pause" (power OFF). Scaled energy-saving states with different energy-saving modes can be specified for devices that support this or for groups of units in the controller of the PROFIenergy device. With PROFIenergy, these various states of energy consumption can be assigned to the devices that will be shut down in the field.

The various states of energy consumption are known as "energy-saving modes". For each individual energy-saving mode, a defined "PE_MODE_ID" is specified.

Programming of the features of the energy saving modes

The details of the energy-saving modes (addressed field device, pause duration etc) are programmed in the user program of the CPU of the PROFIenergy device.

PROFIenergy program blocks for the CP 300 (S7-300)

Implementation of the PROFIenergy functions in S7-300

With a SIMATIC S7-300, the PROFIenergy functions are provided by program blocks for the IO controller and the IO device.

Note that an S7-300 CPU and a CP 300 use different PROFIenergy program blocks.

PROFIenergy specification

The functions of the PROFIenergy program blocks for the CP 300 are based on the following specification of the PROFIBUS Users Organization (PNO):

Common Application Profile PROFIenergy, Technical Specification for PROFINET, Version 1.0, January 2010, Order No. 3.802

PROFIenergy program blocks for the CP 300

The PROFIenergy program blocks are called by the user program of the CPU. The following PROFIenergy program blocks are available for the PROFIenergy functions of the CP 300:

- CP 300 as IO controller:
  - PE_START_END_CP
    Program block for initiating and ending pauses for power supply and setting defined energy-saving modes for the PROFIenergy device.
  - PE_CMD_CP
    Program block for initiating and ending pauses for power supply and setting defined energy-saving modes and for querying measured energy values from the PROFIenergy device.

  The two program blocks can be used as alternatives. Compared with PE_START_END_CP, PE_CMD_CP has an expanded range of functions for the integration of measured energy values.
  For each PROFIenergy device, the program block must be called separately.

  - PE_DS3_WRITE_ET200_CP
    Does not belong to the PROFIenergy function blocks, but expands the PROFIenergy functions for an ET 200S.
    With PE_DS3_WRITE_ET200_CP, the settings for the switching behavior of up to 8 slots (in this case: power modules) of the ET 200S are specified.
- CP 300 as IO device:
  - PE_I_DEV_CP
    Receives all PROFIenergy commands and allows the user program to execute the PROFIenergy functions.
    Makes the response frames of the IO device available to the IO controller.
    PE_I_DEV_CP is called cyclically by the user program of the IO device.
  - Supplementary program blocks (FC 0...FC 8) for PE_I_DEV_CP:
    These FCs make the response data available for PE_I_DEV_CP. The FCs must be called in the user program and linked with PE_I_DEV_CP.

  If the PROFIenergy device is an I-device and itself has subordinate PROFIenergy devices, PE_START_END_CP or PE_CMD_CP is called in the CPU of the I-device for the subordinate PROFIenergy devices.

System and program blocks for transferring data records

The PROFIenergy commands and status information between IO controller and IO device are exchanged by reading and writing data records. This is implemented using the program blocks RDREC and RWREC.

The PROFIenergy data records are described below along with the response data of the individual program blocks.

Note

Block calls

PE_START_END_CP, PE_CMD_CP, PE_I_DEV_CP and PE_DS3_WRITE_ET200_CP must not be called at the same time. The next program block can only be called after one of these program blocks as signaled "no error" (VALID = 1) or "error" (ERROR = 1).

The program block PNIO_RW_REC must also not be called at the same time as PE_START_END_CP, PE_CMD_CP, PE_I_DEV_CP or PE_DS3_WRITE_ET200_CP.

PE_START_END_CP (S7-300)

Meaning and call - PE_START_END_CP (S7-300)

Significance and how it works

PE_START_END_CP can be used as an alternative to PE_CMD_CP.

PE_START_END_CP is used on the IO controller. It triggers an energy saving pause or ends a pause on the assigned PROFIenergy device.

The program block can be used ideally on IO controllers with IO devices that have only field devices connected to them and no energy data needs to be or can be read out from them.

The energy-saving modes are configured in the user program of the IO device. The energy-saving mode actually adopted is reported back by the IO device after execution of PE_START_END_CP and output at the PE_MODE_ID parameter.

The Pause_Time parameter specifies the length of the energy-saving pause for the IO device. On the IO device, the PE_I_DEV_CP program block checks whether or not the specified duration of the pause is adequately long and can be implemented.

Sequence

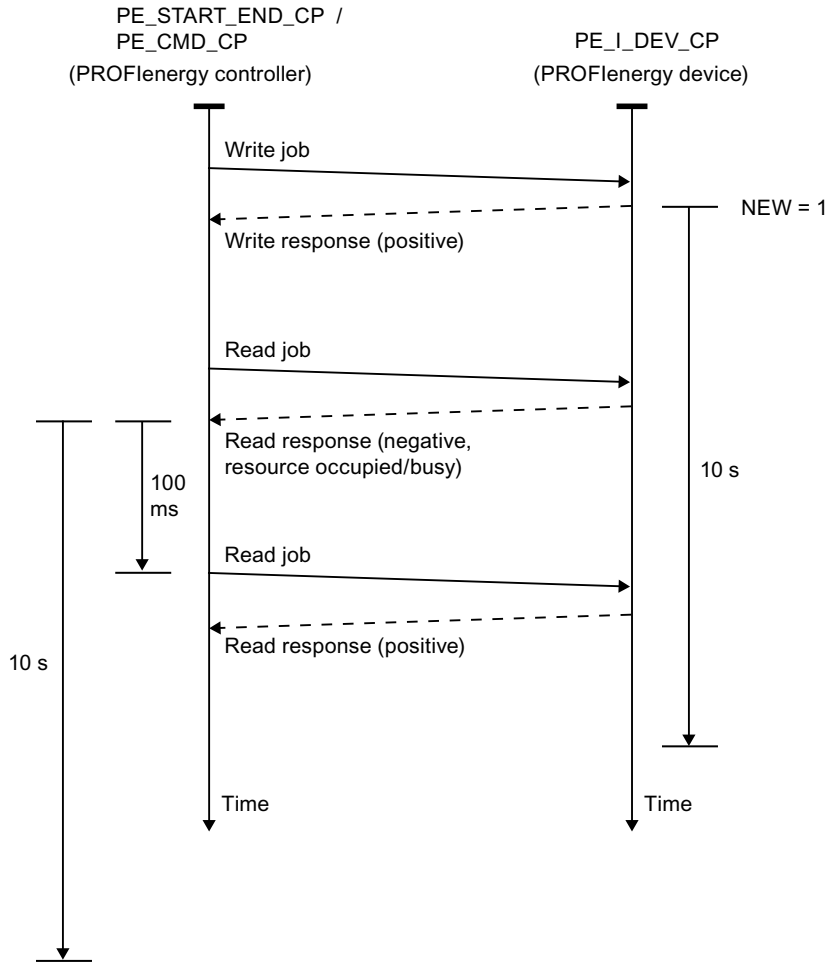

Figure 4-37 Flow chart of the write/read jobs of PE_START_END_CP and PE_CMD_CP

Using WRREC, PE_START_END_CP sends a PROFIenergy command as a write job to the IO device. Following this, PE_START_END_CP waits for the acknowledgment from the IO device. To achieve this, the acknowledgement data record is read every 100 milliseconds using the program block RDREC.

As long as no acknowledgement has arrived from the IO device, the read job is repeated for 10 seconds at intervals of 100 ms.

The response data of the IO device is read with RDREC.

Call interface in FBD representation

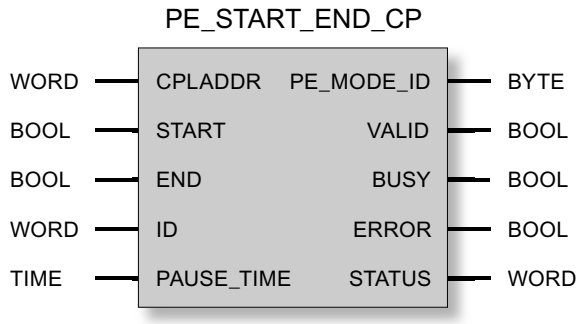

Explanation of the formal parameters of PE_START_END_CP (S7-300)

Explanation of the formal parameters of PE_START_END_CP

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	I, Q, M, D, L, const.	Module start address of the CP
START	INPUT	BOOL	• 1 = command active • 0 = command not active	A rising edge enables the "Start_Pause" command
END	INPUT	BOOL	• 1 = command active • 0 = command not active	A rising edge enables the "End_Pause" command
ID	INPUT	WORD		Logical address of the destination PROFIenergy device
PAUSE_ TIME	INPUT	TIME	T#-24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS	IEC time in steps of 1 ms, integer with sign
PE_MODE_ ID	OUTPUT	BYTE	• 00_h: Power OFF (pause) • 01_h...FE_h: Configurable • FF_h: Ready for operation	ID of the energy-saving mode adopted by the IO device after execution of the command.
VALID	OUTPUT	BOOL	0: - 1: Execution completed successfully	This parameter indicates whether or not the job was completed without errors.
BUSY	OUTPUT	BOOL	0: Execution completed, aborted or not yet started 1: Execution active	Condition code of the processing status of the program block
ERROR	OUTPUT	BOOL	0: - 1: Errors	Error code For the meaning in conjunction with the STATUS parameter, refer to Condition codes of PE_START_END_CP (Page 6575).
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the ERROR parameter, refer to Condition codes of PE_START_END_CP (Page 6575).

See also

Response data (Page 6581)

Condition codes of PE_START_END_CP (S7-300)

Condition codes of PE_START_END_CP

PE_START_END_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

The following additional PROFIenergy-specific condition codes are output. The error codes of STATUS are valid only in conjunction with ERROR = 1.

Table 4-357 Specific condition codes of PE_START_END_CP

STATUS	Meaning
Block-specific errors	
8080_h	Rising edge at START and END at the same time
8081_h	Length conflict between CMD_PARAM and CMD_PARAM_LEN
PROFIenergy-specific errors	
$FE01_h$	Invalid Service_Request_ID
$FE02_h$	Invalid Request_Reference
$FE03_h$	Invalid CMD_MODIFIER
$FE04_h$	Invalid information about the data structure of a command (Data_Structure_Identifier_RQ) in the frame for writing the PROFIenergy data record
$FE05_h$	Invalid information about the data structure of a command (Data_Structure_Identifier_RS) in the frame for reading the PROFIenergy data record
$FE06_h$	Energy saving mode (PE_Mode_ID) not supported
$FE07_h$	Response longer than max transfer length
$FE08_h$	Invalid number of commands
$FE09_h$	Invalid block type (see frame header)
$FE0A_h$	Invalid block length (see frame header)
$FE0B_h$	Invalid block version (see frame header)
$FE50_h$	Not a suitable energy saving mode (PE_Mode_ID)
$FE51_h$	Value for PAUSE_TIME not supported
$FE52_h$	PE_Mode_ID not supported

Details on the parameters of the PROFIenergy-specific errors can be found in the section AUTOHOTSPOT.

PE_CMD_CP (S7-300)

Meaning and call - PE_CMD_CP (S7-300)

Significance and how it works

PE_CMD_CP can be used as an alternative to PE_START_END_CP.

PE_CMD_CP is used on the IO controller and initiates an energy-saving pause or ends a pause on the assigned PROFIenergy device. PE_CMD_CP can also read out further information and energy measured values from an IO device.

The program block can be used ideally on IO controllers with IO devices that have field devices connected to them and energy data needs to be read out from them.

You will find a flowchart of the write/read jobs of PE_CMD_CP in section AUTOHOTSPOT.

The individual commands that can be transferred to the IO device with the program block are assigned defined "Service_Request_IDs". The Service_Request_IDs 01...05 and 16 are assigned in the CMD parameter.

The CMD_MODIFIER parameter specifies the two commands 04 (Query_Modes) and 16 (Query_Measurement) in greater detail.

The CMD_PARA parameter assigns the values for certain parameters to some commands using an Any pointer. The CMD_PARA_LEN parameter specifies the length of this parameter.

The RESPONSE_DATA parameter points to the data area of the response data of the IO device.

Call interface in FBD representation

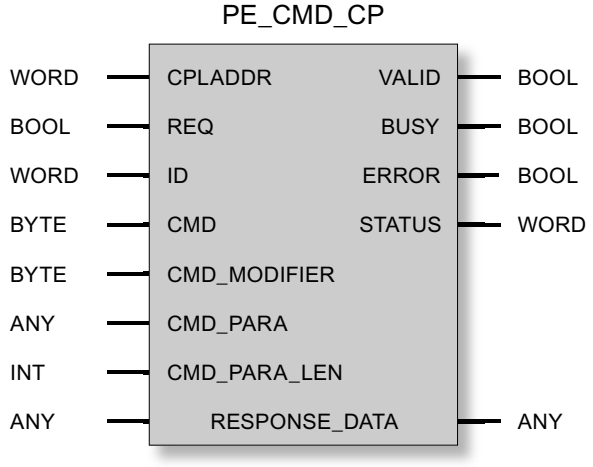

See also

Explanation of the formal parameters of PE_CMD_CP (Page 6577)

Explanation of the formal parameters of PE_CMD_CP (S7-300)

Explanation of the formal parameters of PE_CMD_CP

Parameters	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	I, Q, M, D, L, const.	Module start address of the CP
REQ	INPUT	BOOL		Starts the transfer of the PROFIenergy commands on a rising edge.
ID	INPUT	WORD		Logical address of the destination PROFIenergy device
CMD	INPUT	BYTE	• 01: Start_Pause • 02: End_Pause • 03: Query_Modes • 04: PEM_Status • 05: PE_Identity • 16: Query_Measurement	Service_Request_ID of the PROFIenergy command. You will find the meaning of the commands below this table.
CMD_ MODIFIER	INPUT	BYTE	For "Start_Pause": 00 For "End_Pause": 00 For "Query_Modes": • 01: List_Energy_Saving _Modes • 02: Get_Mode For "PEM_Status": 00 For "PE_Identity": 00 For "Query_Measurement": • 01: Get_Measurement_List • 02: Get_Measurement_Values	Modifier of the PROFIenergy command, meaning: • "Query_Modes" command – Modifier 01: Reads all supported energy-saving modes (PE_Mode_ID). – Modifier 02 reads the parameters of the selected PE_Mode_ID. • "Query_Measurement" command – Modifier 01: Reads the configured Measurement_IDs. – Modifier 02: Reads the measured values of the selected Measurement_IDs. You will find information on the parameters in AUTOHOTSPOT in the section for the particular command. Modifier 00 means "no options".

Parameters	Declaration	Data type	Range of values	Description
CMD_PARA	INPUT	ANY		Any pointer to parameters for commands • For command 01 Start_Pause: "Pause_Time" • For command 02 End_Pause: Irrelevant • For command 03 Query_Modes: – For modifier 01: Irrelevant – For modifier 02 Get_Mode: "PE_Mode_ID" • For command 04 PEM_Status: Irrelevant • For command 05 PE_Identity: Irrelevant • For command 16 Query_Measurement: – For modifier 01: Irrelevant – For modifier 02 Get_Measurement_Values: ANY pointer to the data structure with the parameters "Count" and "Measurement_IDs" You will find information on the parameters in AUTOHOTSPOT in the section for the particular command. The entire data area of the data record to be written (Service_Data_Request) is entered. Maximum length: 234 bytes
CMD_PARA _LEN	INPUT	INT		Actual length of the parameters in CMD_PARA. Max. length: 234 bytes
RESPONSE _DATA	INOUT	ANY		Pointer to the address of the response data of the IO device (complete frame including block header) **Note:** If the area selected is not large enough, only the configured number of bytes is saved.
VALID	OUTPUT	BOOL	0: - 1: Execution completed successfully	The status parameter of the program block indicates whether or not the job was completed without errors.
BUSY	OUTPUT	BOOL	0: Execution not yet started, completed or aborted 1: Execution active	Condition code of the processing status of the program block
ERROR	OUTPUT	BOOL	0: - 1: Errors	Error code For the meaning in conjunction with the STATUS parameter, refer to AUTOHOTSPOT.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the ERROR parameter, refer to AUTOHOTSPOT.

Service_Request_IDs and meaning of the PROFIenergy commands

The PROFIenergy commands with Service_Request_ID 01...05 and 16 have the following significance:

- 01 = Start_Pause
  Command for starting an energy-saving pause.
  The IO device selects the configured energy-saving mode. The energy-saving mode is reported back to the controller in the response data.
- 02 = End_Pause
  Command for ending an energy saving pause
- 03 = Query_Modes
  Queries the configured energy-saving modes with all corresponding time and energy information on the IO device.
  The queried information is detailed using the CMD_MODIFIER parameter:
  - List_Energy_Saving_Modes
    Reads all supported PROFIenergy modes of the IO device.
  - Get_Mode
    Reads the data of the selected PROFIenergy mode.
- 04 = PEM_Status
  Query of the energy-saving mode actually adopted by the field device or the unit group.
- 05 = PE_Identity
  Queries the PROFIenergy services supported by the IO device.
- 16 = Query_Measurement
  Queries the energy data of the IO device.
  The queried information is detailed using the CMD_MODIFIER parameter:
  - Get_Measurement_List
    Reads all the configured Measurement_IDs on the device.
  - Get_Measurement_Values
    Reads the measured energy values of the selected Measurement_IDs.

Commands for various device classes

The devices that can be included in PROFIenergy concepts can be divided into three classes that are addressed by the IO controller with different commands:

- IO modules, actuators, motor starters
  Supported commands:
  - Start_Pause, End_Pause
  - Query_Modes, PEM_Status, PE_Identify
- Measuring devices for electrical variables
  Supported commands:
  - Query_Measurement
- Frequency converters
  Supported commands:
  - Start_Pause, End_Pause
  - Query_Modes, PEM_Status, PE_Identify
  - Query_Measurement
    Data of electrical variables data acquired by frequency converters can also be queried.

Condition codes of PE_CMD_CP (S7-300)

Condition codes of PE_CMD_CP

PE_CMD_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

The following additional PROFIenergy-specific condition codes are output. The error codes of STATUS are valid only in conjunction with ERROR = 1.

Table 4-358 Specific condition codes of PE_CMD_CP

STATUS	Meaning
Block-specific errors	
8081_h	Length conflict between CMD_PARAM and CMD_PARAM_LEN
PROFIenergy-specific errors	
$FE01_h$	Invalid Service_Request_ID
$FE02_h$	Invalid Request_Reference
$FE03_h$	Invalid CMD_MODIFIER
$FE04_h$	Invalid information on the data structure of a command (Data_Structure_Identifier_RQ) in the frame for the PROFIenergy data record to be written
$FE05_h$	Invalid information on the data structure of a command (Data_Structure_Identifier_RS) in the frame for the PROFIenergy data record to be read
$FE06_h$	Energy saving mode (PE_Mode_ID) not supported
$FE07_h$	Response longer than max transfer length
$FE08_h$	Invalid number of commands
$FE09_h$	Invalid block type (see frame header)

STATUS	Meaning
$FE0A_h$	Invalid block length (see frame header)
$FE0B_h$	Invalid block version (see frame header)
$FE50_h$	Not a suitable energy saving mode (PE_Mode_ID)
$FE51_h$	Value for PAUSE_TIME not supported
$FE52_h$	PE_Mode_ID not supported

Details on the parameters of the PROFIenergy-specific errors can be found in the section AUTOHOTSPOT.

Response data (S7-300)

Structure of the response data

The following table shows the structure of the data record ($80A0_h$) of the response data of PE_START_END_CP and PE_CMD_CP.

The following table shows an overview of the structure of the data record of the returned response data according to the PROFIenergy specification. The composition of the "Service Data Response" area is described below for the individual PROFIenergy commands.

Table 4-359 Structure of the response data

Block definitions	Attributes	Value	Data type	Description
Block header	BlockType	0801_h	Unsigned16	
	BlockLength		Unsigned16	Frame length (without the "BlockType" and "BlockLength" fields)
	BlockVersionHigh	01_h	Unsigned8	
	BlockVersionLow	00_h	Unsigned8	
Response header	Service_Request_ID	01_h...FF_h	Unsigned8	01_h: Start_Pause 02_h: End_Pause 03_h: Query_Modes 04_h: PEM_Status 05_h: PE_Identify 06_h...09_h: - Reserved - 10_h: Query_Measurement 11_h...CF: - Reserved - $D0_h$...FF_h: Vendor-specific
	Request_Reference	01_h...FF_h	Unsigned8	Identification number of the query (mirrored in the response of the IO device)

Block definitions	Attributes	Value	Data type	Description
Service header response	Status	01_h...FF_h	Unsigned8	00_h: - Reserved - 01_h: Done 02_h: Done with error(s) 03_h: Data incomplete 04_h...CF_h: - Reserved - $D0_h$...FF_h: Depends on the Service_Request_ID
	Data_Structure_Identifier_RS	01_h...FF_h	Unsigned8	00_h: - Reserved - 01_h...FF_h: Data structure dependent on the Service_Request_ID FF_h: error
Service data response				Response data of the IO device Depending on the particular PROFIenergy command (described below)

Meaning of "Service data request" and "Service data response"

The following sections explain the parameter values for the queries of the IO controller to the IO device (Service Data Request) for each PROFIenergy command and the structure of the response data of the IO device (Service Data Response).

- Service data request
  Parameter values for IO controller queries
- Service data response
  Structure of the response data of the IO device

PROFIenergy command "Start_Pause"

- Service data request
  - CMD = 01
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 04
  - CMD_PARA = Any pointer to the value for "Pause_Time" (data type "TIME")
    IEC time in steps of 1 ms, integer with sign
    Value: T#-24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS
- Service data response

Parameters	Value	Data type
PE_Mode_ID *	01_h...FF_h	Unsigned8
- Reserved -	00_h	Unsigned8

* Identification number of the energy-saving mode

PROFIenergy command "End_Pause"

- Service data request
  - CMD = 02
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant
- Service data response

Parameters	Value	Data type
Time_to_operate *		Unsigned32

* Expected time for switching over the PROFIenergy device to "ready to operate"

PROFIenergy command "Query_Modes" – List_Energy_Saving_Modes

- Service data request
  - CMD = 03
  - CMD_MODIFIER = 01
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant
- Service data response

Parameters	Value	Data type
Number_of_PE_Mode_IDs *	01_h	Unsigned8
PE_Mode_IDs		Unsigned8 array of Number_of_PE_Mode_IDs (unique ID for mode)

* Number of energy-saving modes

PROFIenergy command "Query_Modes" – Get_Mode

- Service data request
  - CMD = 03
  - CMD_MODIFIER = 02
  - CMD_PARA_LEN = 01
  - CMD_PARA = Any pointer to value for PE_MODE_ID (unsigned8)
- Service data response

Parameters	Value	Data type
PE_Mode_ID	01_h...FF_h	Unsigned8
PE_Mode_Attributes *	00_h...01_h	Unsigned8

Parameters	Value	Data type
Time_min_Pause		Unsigned32
Time_to_Pause		Unsigned32
Time_to_operate		Unsigned32
Time_min_length_of_stay		Unsigned32
Time_max_length_of_stay		Unsigned32
Mode_Power_Consumption		Float32
Energy_Consumption_to_pause		Float32
Energy_Consumption_to_operate		Float32

* Coding of bit 0:
0 = Only static time and energy measured values available.
1 = Dynamic time and energy measured values available.
Bits 1...7: Reserved

PROFIenergy command "PEM_Status"

- Service data request
  - CMD = 04
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant
- Service data response

Parameters	Value	Data type
PE_Mode_ID_Source *	00_h 01_h...FE_h FF_h	Unsigned8
PE_Mode_ID_Destination *	00_h 01_h...FE_h FF_h	Unsigned8
Time_to_operate		Unsigned32
Remaining_time_to_destination		Unsigned32
Mode_Power_Consumption		Float32
Energy_Consumption_to_Destination		Float32
Energy_Consumption_to_operate		Float32

* Possible values for "PE_Mode_ID_Source" and "PE_Mode_ID_Destination":
00_h: PE_Power_off
01_h...FE_h: Freely configurable
FF_h: PE_Ready_to_operate

PROFIenergy command "PE_Identify"

- Service data request
  - CMD = 05
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant
- Service data response

Parameters	Value	Data type
Count *	6	Unsigned8
Start_Pause **	01_h	Unsigned8
End_Pause	02_h	Unsigned8
Query_Modes	03_h	Unsigned8
PEM_Status	04_h	Unsigned8
PE_Identify	05_h	Unsigned8
Query_Measurement ***	10_h	Unsigned8

* Number of supported PROFIenergy commands

** Service_Request_ID of the first supported PROFIenergy command

*** Service_Request_ID of the last supported PROFIenergy command

PROFIenergy command "Query_Measurement" – Get_Measurement_List

- Service data request
  - CMD = 16
  - CMD_MODIFIER = 01
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant
- Service data response

Parameters	Value	Data type
Count *		Unsigned8
- Reserved -		Unsigned8
Measurement_ID **		Unsigned16
Accuracy_Domain [1]		Unsigned8
Accuracy_Class [2]		Unsigned8
Range [3]		Float32
…		
Measurement_ID ***		Unsigned16
Accuracy_Domain [1]		Unsigned8

Parameters	Value	Data type
Accuracy_Class [2]		Unsigned8
Range [3]		Float32

* Number of Measurement_IDs

** First supported Measurement_ID

*** Last supported Measurement_ID

[1] Accuracy domain (range 1...4):
0 = Reserved
1 = Percentage of the measuring range
2 = Percentage of the current measured values
3 = Accuracy according to IEC 61557-12
4 = Accuracy according to EN 50470-3 section 8

[2] Accuracy class (range 1...15):
0 = Reserved
1 (0.01%) ... 15 (>20%)

[3] Measuring range if Accuracy_Domain = 1; otherwise undefined

PROFIenergy command "Query_Measurement" – Get_Measurement_Values

- Service data request
  - CMD = 16
  - CMD_MODIFIER = 02
  - CMD_PARA_LEN = length of the data structure in bytes
  - CMD_PARA = Any pointer to data structure with the following structure:

Parameters	Value	Data type
Count *		Unsigned8
- Reserved -		Unsigned8
Measurement_ID **		Unsigned16
…		
Measurement_ID ***		Unsigned16

* Number of Measurement_IDs

** First queried measured value

*** Last queried measured value

- Service data response

Parameters	Value	Data type
Count *		Unsigned8
- Reserved -		Unsigned8
Length_of_Structure	0002_h...$FFFF_h$	Unsigned16
Measurement_Data_Structure_ID	1 = simple value	Unsigned8
Measurement_ID **	00_h...FF_h	Unsigned16

Parameters	Value	Data type
Status_of_Measurement_Value	1 = valid 2 = not available 3 = not available at times	Unsigned8
Transmission_Data_Type		Float32
End_of_demand		Unsigned32 or Unsigned16
Length_of_Structure		Unsigned16
Measurement_Data_Structure_ID		Unsigned8
Measurement_ID ***		Unsigned16
Status_of_Measurement_Value		Unsigned8
Transmission_Data_Type		Float32
End_of_demand		Unsigned32 or Unsigned16

* Number of Measurement_IDs

** First queried measured value

*** Last queried measured value

See also

Explanation of the formal parameters of PE_START_END_CP (Page 6574)

PE_I_DEV_CP (S7-300)

Meaning and call - PE_I_DEV_CP (S7-300)

Significance and how it works

The program block PE_I_DEV_CP is used on the PROFIenergy device where it handles the PROFIenergy commands of the IO controller. The PROFIenergy data records ($80A0_h$) sent by the IO controller are forwarded by the CP firmware to PE_I_DEV_CP. The PROFIenergy data of the IO device is made available to the IO controller as the response by PE_I_DEV_CP using the PROFIenergy data record ($80A0_h$).

The response data of PE_I_DEV_CP is generated by the supplementary functions FC 0 to FC 8, see section AUTOHOTSPOT.

Call interface in FBD representation

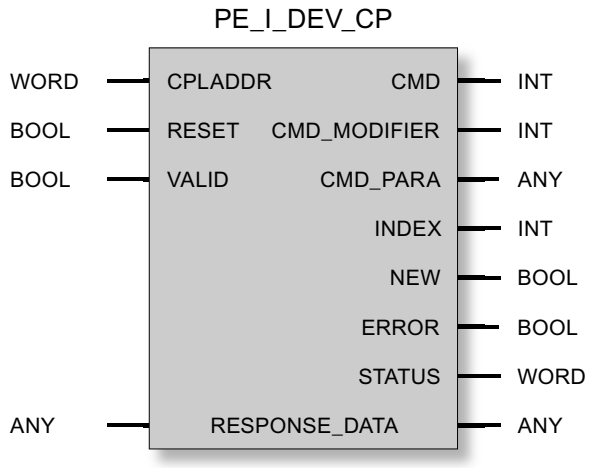

Explanation of the formal parameters of PE_I_DEV_CP (S7-300)

Explanation of the formal parameters of PE_I_DEV_CP

Parameters	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	I, Q, M, D, L, const.	Module start address of the CP
RESET	INPUT	BOOL		Resets processing of the program block. NEW is set to 0.
VALID	INPUT	BOOL		If the response data was written to the relevant memory area of the IO device, VALID = 1 must be set by the user program. Following this, the program block makes the data available to the IO controller. NEW is set to 0.
CMD	OUTPUT	INT	• 01: Start_Pause • 02: End_Pause • 03: Query_Modes • 04: PEM_Status • 05: PE_Identity • 16: Query_Measurement	Service ID of the PROFIenergy command

Parameters	Declaration	Data type	Range of values	Description
CMD_ MODIFIER	OUTPUT	INT	• Modifier for Start_Pause: 00 • Modifier for End_Pause: 00 • Query_Modes, Modifier: – 01 (List_Energy_ Saving_Modes) – 02 (Get_Mode) • Modifier for PEM_Status: 00 • Modifier for PE_Identity: 00 • Query_Measurement, Modifier: – 01 (Get_Measurement _List) – 02 (Get_Measurement _Values)	Modifier of the PROFIenergy commands Meaning of the modifiers for commands: • "Query_Modes" command, Modifier: – 01 (List_Energy_Saving_Modes): Reads all supported PROFIenergy modes – 02 (Get_Mode): Reads the data of the selected PROFIenergy mode • "Query_Measurement" command, Modifier: – 01 (Get_Measurement_List): Reads all configured Measurement_IDs. – 02 (Get_Measurement_Values): Reads the measured values of the selected Measurement_IDs.
CMD_PARA	OUTPUT	ANY		Any pointer to parameters for the following command modifiers (compare CMD_MODI-FIER parameter): • For "Get_Mode": PE_Mode_ID (ID of the energy-saving mode) length = 1 • For "Get_Measurement_Values": measured values of the Measurement_IDs length = max. 236 bytes (complete frame of the controller command without header)
INDEX	OUTPUT	INT		Number of the PROFIenergy data record ($80A0_h$)
NEW	OUTPUT	BOOL	0: Execution not yet started, completed or aborted 1: Execution active	Condition codes of the processing status of the program block
ERROR	OUTPUT	BOOL	0: - 1: Errors	Error code For the meaning in conjunction with the STA-TUS parameter, refer to Condition codes of PE_I_DEV_CP (Page 6590).

Parameters	Declaration	Data type	Range of values	Description
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the ERROR parameter, refer to Condition codes of PE_I_DEV_CP (Page 6590).
RESPONSE_DATA	INOUT	ANY	See "Response data" of the program block	Pointer to the data area of the response of the IO device (complete response frame including header). The data area must match the data area of the supplementary program blocks FC 0 - FC 8 (parameter "DATA_ERRORRSP"). Recommended size: At least 244 bytes. If the data area is too small, only the data of the configured bytes are transferred.

Condition codes of PE_I_DEV_CP (S7-300)

Condition codes of PE_I_DEV_CP

PE_I_DEV_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

Supplementary program blocks for PE_I_DEV_CP (S7-300)

Overview of the FCs (S7-300)

Function

The supplementary program blocks FC 0 to FC 8 support the preparation of the response data made available to the controller by PE_I_DEV_CP:

- For the response data of each PROFIenergy command, there is a separate program block (FC 1 - FC 8).
- FC 0 generates a common negative response for all PROFIenergy commands.

The FCs are called in the user program. In STEP 7 V5.5, they are available in the standard library in the "PROFIenergy" folder.

The FCs have several common parameters as well as individual parameters. Some of the common parameters of the FCs are interconnected with parameters of PE_I_DEV_CP. With some of the individual input parameters of the FCs, the response data is entered as plain language for the user or stored in the memory area of the IO device.

Overview of the FCs

The following supplementary program blocks are made available:

Table 4-360 Overview of the supplementary FCs

Number	Name
FC 0	PE_ERROR_RSP
FC 1	PE_START_RSP
FC 2	PE_END_RSP
FC 3	PE_LIST_MODES_RSP
FC 4	PE_GET_MODE_RSP
FC 5	PE_PEM_STATUS_RSP
FC 6	PE_IDENTIFY_RSP
FC 7	PE_MEASUREMENT_LIST_RSP
FC 8	PE_MEASUREMENT_VALUE_RSP

Interconnection of the FCs with PE_I_DEV_CP (S7-300)

Interconnection of the FCs with the program block PE_I_DEV_CP

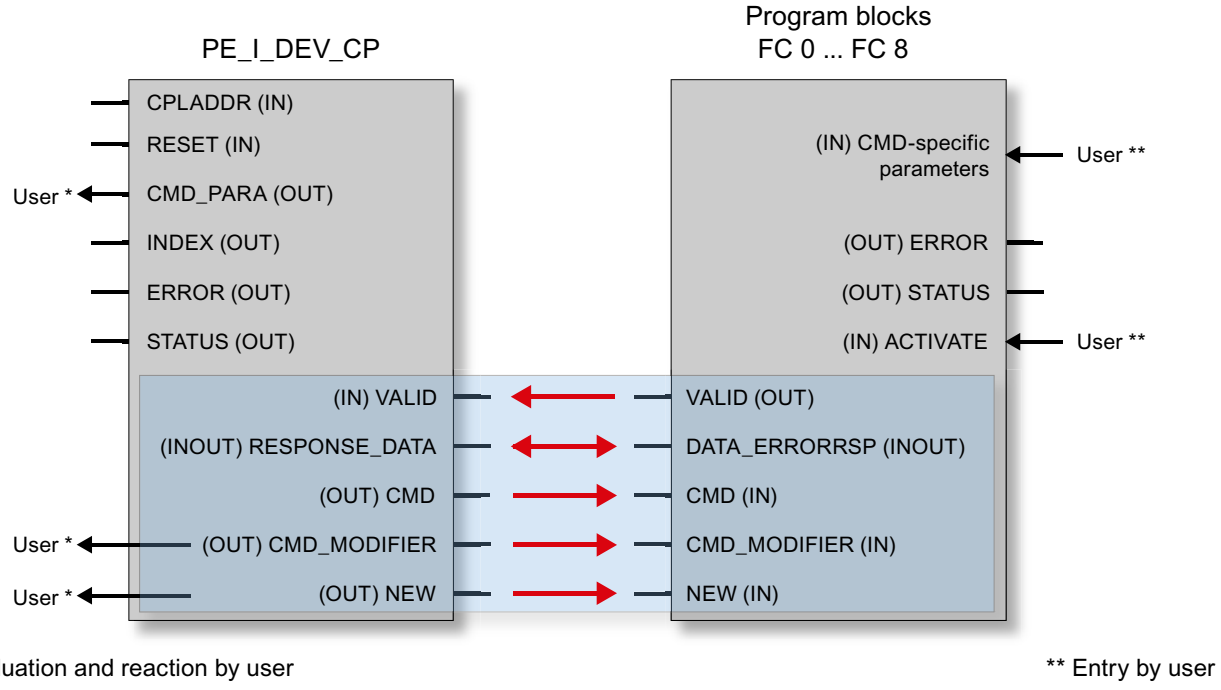

* Evaluation and reaction by user

** Entry by user

Figure 4-38 Interconnection of the FCs with PE_I_DEV_CP

Note

Interconnection of the program blocks is an absolute necessity

PE_I_DEV_CP must be interconnected with FC 0...FC 8 at the parameters shown on a light blue background that are assigned to the corresponding parameters of the FCs indicated by red arrows.

Common parameters of the FCs (S7-300)

Common parameters of the supplementary program blocks FC 0 - FC 8

Parameters	Declaration	Data type	Range of values	Description
Input parameters				
ACTIVATE	INPUT	BOOL		Instructs the block to copy the input parameters to the "DATA_ERRORRSP" data area are on a rising edge. Is then reset by the block. Must be set by the user within 10 seconds after a positive edge was detected at PE_I_DEV_NEW.
PE_I_DEV_NEW	INPUT	BOOL		Must be interconnected with the NEW output parameter of PE_I_DEV_CP. The block is processed only when 1 is set.
CMD	INPUT	INT		Must be interconnected with the CMD output parameter of PE_I_DEV_CP.
CMD_MODIFIER	INPUT	INT		Must be interconnected with the CMD_MODIFIER output parameter of PE_I_DEV_CP.
Output parameters				
DATA_ERRORRSP	OUTPUT	ANY		Pointer to the data area in which the response data will be stored (complete response frame including header). Must be interconnected with the RESPONSE_DATA output parameter of PE_I_DEV_CP. Recommended size: At least 244 bytes.
VALID	OUTPUT	BOOL	0: - 1: No error	Is set by the block. Must be interconnected with the VALID input parameter of PE_I_DEV_CP.
ERROR	OUTPUT	BOOL	0: No error 1: Errors	Error code
STATUS	OUTPUT	WORD	0: No error	Status code $80B1_h$: Error in ANY information (for example wrong area)

Individual parameters of the FCs (S7-300)

Individual parameters of FC 0 to FC 8

Below you will find a description of the individual parameters of the FCs.

PE_ERROR_RSP

Generates a negative response if the required PROFIenergy command is generally or temporarily not supported. The negative response is not dependent on the requesting command.

Table 4-361 Individual parameters of FC 0 PE_ERROR_RSP

Parameters	Declaration	Data type	Range of values	Description
ERROR_CODE	INPUT	BYTE		Error number

PE_START_RSP

Initiates an energy saving pause. Generates the response to the "Start_Pause" command. Returns the energy-saving mode adopted by the device.

Table 4-362 Individual parameters of FC 1 PE_START_RSP

Parameters	Declaration	Data type	Range of values	Description
PE_Mode_ID	INPUT	BYTE		ID of the energy-saving mode that the device or the unit group adopts.

Return message with the PE_Mode_ID of the energy-saving mode that the field devices or the unit group have adopted.

PE_END_RSP

Generates the response to the "End_Pause". command

Table 4-363 Individual parameters of FC 2 PE_END_RSP

Parameters	Declaration	Data type	Range of values	Description
Time_to_Operate	INPUT	DWORD		Time required to change from the current energy-saving mode after "ready to operate".

PE_LIST_MODES_RSP

Generates the response to the "Query_Modes" > modifier "List_Modes" command (list of the supported energy-saving modes).

The IDs of the energy-saving modes must be specified in the user program.

Table 4-364 Individual parameters of FC 3 PE_LIST_MODES_RSP

Parameters	Declaration	Data type	Range of values	Description
Number_of_PE_Mode_IDs	INPUT	BYTE		Number of supported energy-saving modes
PE_Mode_ID	INPUT	ANY	• 00_h • 01_h...FE_h • FF_h	Pointer to the area in which the energy-saving modes are stored. As the user, you will need to store the IDs of the energy-saving modes here. An energy-saving mode ID is configured in the Unsigned8 format. Permitted range: 1 to 254 bytes.

If the devices or a group need to react differently to different lengths of pause you can set up different energy-saving modes (PE_Mode) to achieve this. You assign a different PE_Mode_ID to the various energy-saving modes.

Possible values for "PE_Mode_ID":

- 00_h: PE_Power_off
- 01_h...FE_h: Freely configurable or vendor-specific
- FF_h: PE_Ready_to_operate

PE_GET_MODE_RSP

Generates the response to the "Query_Modes" > Modifier "Get_Mode". command

Table 4-365 Individual parameters of FC 4 PE_GET_MODE_RSP

Parameters	Declaration	Data type	Range of values	Description
PE_Mode_ID	INPUT	BYTE		Currently used energy-saving mode ID
Time_Min_Pause *	INPUT	Unsigned32		Minimum pause duration for this PE energy-saving mode. It is the sum of the three parameters: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay
Time_to_Pause *	INPUT	Unsigned32		Time from the START edge until the requested energy-saving mode is reached
Time_to_operate *	INPUT	Unsigned32		Max. time after turn on before PE_ready_to_operate Time_to_operate can be used directly for the relevant calculations. The value can either be a static MAX value or can be calculated dynamically by the PE device.
Time_min_length_of_stay *	INPUT	Unsigned32		Minimum time that the PE device must remain in this PE_Mode.
Time_max_length_of_stay *	INPUT	Unsigned32		Maximum time that the PE device can remain in this PE_Mode.

Parameters	Declaration	Data type	Range of values	Description
Mode_Power _Consumption **	INPUT	Float32		Energy consumption in current PE_Mode [kW]
Energy_Consumption _to_pause **	INPUT	Float32		Energy consumption of PE_ready_to_operate until the current PE_Mode [kWh]
Energy_Consumption _to_operate **	INPUT	Float32		Energy consumption from current PE_Mode until PE_ready_to_operate [kWh]

* The PROFIenergy profile does not specify an invalid time format.
If the time is unlimited, the maximum value $FFFFFFFF_h$ can be specified.
If the time is zero, 00_h can be used.

** If an energy consumption value is not defined, 0.0 (Float32) can be specified.

PE_PEM_STATUS_RSP

Generates the response to the "PEM_STATUS". command

Table 4-366 Individual parameters of FC 5 PE_PEM_STATUS_RSP

Parameters	Declaration	Data type	Range of values	Description
PE_Mode_ID_Source	INPUT	BYTE	• 00_h • 01_h...FE_h • FF_h	ID of the energy-saving mode actually adopted
PE_Mode_ID_Destination	INPUT	BYTE	• 00_h • 01_h...FE_h • FF_h	ID of the energy-saving mode set by the controller
Time_to_operate *	INPUT	Unsigned32		Max. time after turn on before PE_ready_to_operate Time_to_operate can be used directly for the relevant calculations. The value can either be a static MAX value or can be calculated dynamically by the PE device.
Remaining_time_to _destination *	INPUT	Unsigned32		Optional: Time remaining until the requested PE_Mode. Dynamic value or static MAX value
Mode_Power _Consumption **	INPUT	Float32		Energy consumption in current PE_Mode [kW]
Energy_Consumption _to_Destination **	INPUT	Float32		Energy consumption until the requested PE_Mode [kWh]
Energy_Consumption _to_operate **	INPUT	Float32		Energy consumption from current PE_Mode until PE_ready_to_operate [kWh]

* The PROFIenergy profile does not specify an invalid time format.
If the time is unlimited, the maximum value $FFFFFFFF_h$ can be specified.
If the time is zero, 00_h can be used.

** If an energy consumption value is not defined, 0.0 (Float32) can be specified.

Possible values for "PE_Mode_ID_Source" and "PE_Mode_ID_Destination":

- 00_h: PE_Power_off
- 01_h...FE_h: Freely configurable or vendor-specific
- FF_h: PE_Ready_to_operate

PE_IDENTIFY_RSP

Generates the response to the "PE_Identify". command

As the user you need to specify which PROFIenergy commands are supported.

Table 4-367 Individual parameters of FC 6 PE_IDENTIFY_RSP

Parameters	Declaration	Data type	Range of values	Description
Count	INPUT	BYTE	0...6	Meaning of supported PROFIenergy commands
Start_Pause	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported
End_Pause	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported
Query_Modes	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported
PEM_Status	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported
PEM_Identify	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported
Query_Measurement	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported

PE_MEASUREMENT_LIST_RSP

Generates the response to the "Query_Measurement" > Modifier "Get_Measurement_List". command

Table 4-368 Individual parameters of FC 7 PE_MEASUREMENT_LIST_RSP

Parameters	Declaration	Data type	Range of values	Description
Count	INPUT	BYTE		Number of supported measured value IDs (Measurement_ID)
Measurement_List	INPUT	ANY		Pointer to the data area with the supported measured value IDs. As the user, you store the measured value IDs in this data area. Per frame, a maximum of 29 measured value IDs can be transferred. For information on the structure of the array, refer to section AUTOHOTSPOT > "Query_Measurement" – Get_Measurement_List.

PE_MEASUREMENT_VALUE_RSP

Generates the response to the "Query_Measurement" > Modifier "Get_Measurement_Values". command

Table 4-369 Individual parameters of FC 8 PE_MEASUREMENT_VALUE_RSP

Parameters	Declaration	Data type	Range of values	Description
Count	INPUT	BYTE		Number of supported Measurement_Values
Measurement_Values	INPUT	ANY		Pointer to the data area of the measured values (Measurement_Values). As the user, you store the measured values in this data area. Per frame, a maximum of 116 measured values can be transferred. For information on the structure of the array, refer to section AUTOHOTSPOT > "Query_Measurement" – Get_Measurement_List.

PE_DS3_WRITE_ET200S_CP (S7-300)

Meaning and call - PE_DS3_WRITE_ET200_CP (S7-300)

Significance and how it works

PE_DS3_WRITE_ET200_CP is used in the CPU of the CP 300 as a PROFIenergy controller to transfer the settings for switching behavior from power modules to an ET 200S. The switching behavior for up to 8 slots (in this case: power modules) can be transferred.

PE_DS3_WRITE_ET200_CP is not a PROFIenergy program block.

Call interface in FBD representation

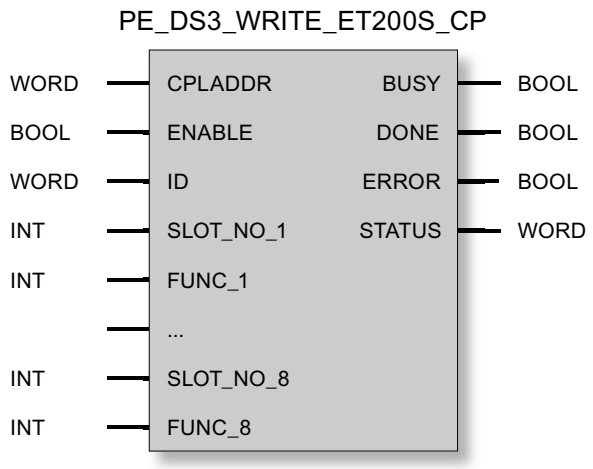

Explanation of the formal parameters of PE_DS3_WRITE_ET200S_CP (S7-300)

Explanation of the formal parameters of PE_DS3_WRITE_ET200_CP

Parameters	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	I, Q, M, D, L, const.	Module start address of the CP
ENABLE	INPUT	BOOL		Starts the processing of the program block on a rising edge.
ID	INPUT	WORD		Logical address of the header slot of the IO device
SLOT_NO_1	INPUT	INT		Slot number of the first power module

Parameters	Declaration	Data type	Range of values	Description
FUNC_1	INPUT	INT		Specifies the switching behavior for the power module in terms of starting or ending energy-saving pauses • 0 (FALSE) – PAUSE_START: No influence (power module remains turned on) – PAUSE_STOP: Turns the power module on again. • 1 (TRUE) – PAUSE_START: Turns the power module off. – PAUSE_STOP: Turns the power module on again.
...	INPUT	INT		
...	INPUT	INT		
SLOT_NO_8	INPUT	INT		Slot number of the eighth power module
FUNC_8	INPUT	INT		See "FUNC_1"
BUSY	OUTPUT	BOOL	0: Execution not yet started, completed or aborted 1: Execution active	Condition code of the processing status of the program block
DONE	OUTPUT	BOOL	0: - 1: Data record transferred successfully	This parameter indicates whether or not the job was completed without errors.
ERROR	OUTPUT	BOOL	0: - 1: Errors	Error code
STATUS	OUTPUT	WORD		Status code

For the meaning of DONE, ERROR and STATUS, see Condition codes of PE_DS3_WRITE_ET200S_CP (Page 6599).

Condition codes of PE_DS3_WRITE_ET200S_CP (S7-300)

Condition codes of PE_DS3_WRITE_ET200_CP

PE_DS3_WRITE_ET200_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

Instructions for programmed connections

Overview

Interplay between programming and configuration

You configure connections on the SEND/RECEIVE interface and the IP configuration of a CP either with STEP 7 or you configure them during runtime of the S7 station via the user program. Mixing these variants on a CP is not possible!

Principle of programmed configuration

Configuration data for communication connections and the IP configuration can be transferred to the CPU using an instruction called in the user program.

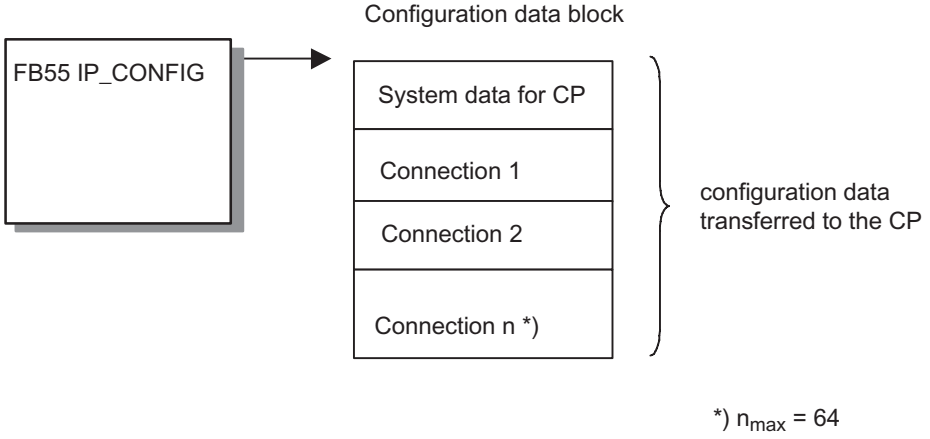

The configuration DB can be loaded on the CP at any time. The previously applicable connections and configuration data (IP address, subnet mask, default router, NTP time server and other parameters) are overwritten.

Based on the configuration data, the Ethernet CP recognizes that the communication connections must be set up by the user program.

Note

The functions can only be executed if "Not locked" was configured for the module access protection. Refer to the "Options" tab in the properties dialog of the CP (not available for every CP).

The "Set IP address in user program" option must also be enabled (see properties dialog of the CP or the Ethernet interface of the CP, "IP Configuration" tab).

Based on the configuration data, the Ethernet CP recognizes that the communication connections must be set up by the user program.

Note

As soon as the user program transfers the connection data via the IP_CONFIG instruction, the CPU switches the CP briefly to STOP. The CP receives the system data (including the IP address) and the new connection data and processes them during startup (RUN).

Quantity framework

A maximum of 64 connections can be specified in the CP_CONFIG instruction. The most important factor, however, is the maximum number of connections supported by the CP type you are using.

Special features / restrictions

- Consistency check only in the configuration
  The connection configuration involves consistency checks that are not possible or only possible with restrictions when using the programmed configuration!
- Connection configuration required on the partner
  When configuring specified connections, you implicitly create the connection for the partner; with a programmed configuration, this is not possible! In this case, you must configure suitable connections for the partner.
- Configuring IP access protection
  Using IP access protection gives you the opportunity of restricting communication over the CP of the local S7 station to partners with specific IP addresses. This parameter assignment also applies to programmed communications connections. You either disable IP access protection in the configuration (= default) or authorize the communications partner.
- DHCP / DNS is supported
  With a programmed configuration, IP addressing is also possible using DHCP (and DNS for the mail service).
  The use of a DHCP server is defined in this case in the IP_CONFIG instruction (not in the configuration).
- No connection information when uploading
  When you upload the S7 station data into the project engineering, this does not contain the data of the programmed configuration.
- Configuring connections for CPs with several interfaces
  If you are using CPs with several interfaces (for example with a gigabit interface), check the device manual to see whether or not the connection configuration is supported for both interfaces.

- PROFINET IO is not possible at the same time
  On a device you intend to operate as a PROFINET IO controller or IO device, it is not possible to set up the connection using FB55 as described here.
- No use of the IP_CONFIG instruction when operating the CP with fault-tolerant S7 connections
  If you configure fault-tolerant S7 connections via the CP, you cannot use the IP_CONFIG instruction for IP configuration of the CP.

See also

IP_CONFIG for programmed communications connections (Page 6602)

Configuration data block (Page 6607)

IP_CONFIG

IP_CONFIG for programmed communications connections

Description

The IP_CONFIG instruction transfers the connection data specified in a data block (configuration DB) to the CP. The configuration DB contains all the connection data to allow the connections for the SEND/RECEIVE interface of an Ethernet CP to be set up.

You can use this variant of programmed communication connections as an alternative to connection configuration with STEP 7.

Depending on the size of the configuration DB, the data may be transferred to the CP in several segments. This means that the instruction must continue to be called until it signals complete transfer by setting the DONE bit to 1.

Call

Call interface in FBD representation

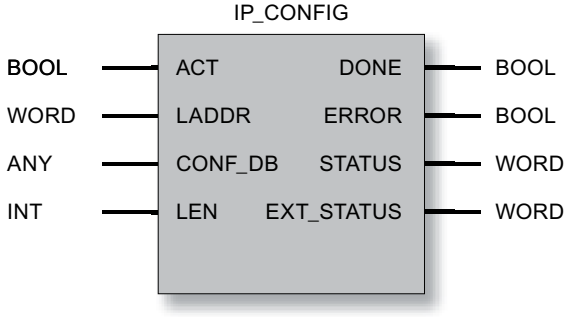

Note

No use of the IP_CONFIG instruction when operating the CP with fault-tolerant S7 connections

If you configure fault-tolerant S7 connections via the CP, you cannot use the IP_CONFIG instruction for IP configuration of the CP.

Note

Avoid possible double addressing

If you use the IP_CONFIG instruction, make sure that the assignment of IP addresses is unique. If an address is detected twice, it is possible that the CP will not become active in the network.

See also

Overview (Page 6600)

Configuration data block (Page 6607)

Parameters for IP_CONFIG (Page 6603)

Reserved port numbers - IP_CONFIG (Page 6604)

Parameters DONE, ERROR, STATUS (Page 6605)

Parameters for IP_CONFIG

Explanation of the formal parameters

The following table explains the formal parameters for the call interface of the "IP_CONFIG" instruction:

Parameter	Declaration	Data type	Range of values	Description
ACT	INPUT	BOOL, TIMER, COUN-TER	0,1	When the instruction is called with ACT = 1, DBxx is sent to the CP. If the instruction is called with ACT = 0, only the status codes DONE, ERROR and STATUS are updated.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.
CONF_DB	INPUT	ANY		The parameter points to the start address of the configuration data area in a data block (data type: byte).
LEN	INPUT	INT		Length information in bytes for the configuration data area.

Parameter	Declaration	Data type	Range of values	Description
DONE	OUTPUT	BOOL	0: - 1: Job completed with data transfer.	The parameter indicates whether the configuration data area was completely transferred. Remember that it may be necessary to call the instruction several times depending on the size of the configuration data area (in several cycles) until the DONE parameter is set to 1 to signal completion of the transfer. For the meaning of this parameter in conjunction with the ERROR and STATUS parameters, refer to the following table.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning of this parameter in conjunction with the DONE and STATUS parameters, refer to the following table.
STATUS	OUTPUT	WORD	See following table	Status code For the meaning of this parameter in conjunction with the DONE and ERROR parameters, refer to the following table.
EXT_Status	OUTPUT	WORD		If an error occurs in the execution of a job, the parameter indicates which parameter was detected as the cause of the error in the configuration DB. High byte: Index of the parameter field Low byte: Index of the subfield within the parameter field

See also

IP_CONFIG for programmed communications connections (Page 6602)

Reserved port numbers - IP_CONFIG

Reserved Port Numbers

The following local port numbers are reserved; do not use these in the connection project engineering.

Table 4-370 Reserved Port Numbers

Protocol	Port number	Service
TCP	20, 21	FTP
TCP	25	SMTP
TCP	80	HTTP
TCP	102	RFC1006
TCP	135	RPC-DCOM
TCP	502	ASA application protocol
UDP	161	SNMP_REQUEST
UDP	34964	PN IO
UDP	65532	NTP

Protocol	Port number	Service
UDP	65533	NTP
UDP	65534	NTP
UDP	65535	NTP

See also

IP_CONFIG for programmed communications connections (Page 6602)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Table 4-371 IP_CONFIG status codes

DONE	ERROR	STATUS	Meaning
General codes relating to job execution			
1	0	0000_H	Job completed without errors
0	0	8181_H	Job active
Errors detected on the interface between CPU and CP.			
0	1	$80A4_H$	• Communication error on the K-bus or • Data error: Configuration by the user program is not set.
0	1	$80B1_H$	The number of data bytes to be sent exceeds the upper limit for this service. (upper limit = 16 Kbytes)
0	1	$80C4_H$	Communication error The error can occur temporarily; it is usually best to repeat the job in the user program.
0	1	$80D2_H$	Configuration error The module you are using does not support this service.
Errors detected in the evaluation of the IP_CONFIG in the CPU or on the interface between CPU and CP.			
0	1	8183_H	The CP rejects the requested data record number.
0	1	8184_H	System error or bad parameter type. (data type of the ANY pointer CONF_DB not OK) (Currently only the byte data type is accepted)
0	1	8185_H	The value of the LEN parameter is larger than the CONF_DB less the reserved header (4 bytes) or the length information is incorrect.
0	1	8186_H	Illegal parameter detected The ANY pointer CONF_DB does not point to a data block.
0	1	8187_H	Illegal status of the IP_CONFIG Data in the header of CONF_DB was possibly overwritten.
Further errors detected on the interface between the CPU and CP.			
0	1	$8A01_H$	The status code in the data record is invalid (value is >= 3).

DONE	ERROR	STATUS	Meaning
0	1	$8A02_H$	There is no job running on the CP. IP_CONFIG did, however, expect an acknowledgment for the executed job.
0	1	$8A03_H$	There is no job running on the CP and the CP is not ready. IP_CONFIG triggered the first job to read a data record.
0	1	$8A04_H$	There is no job running on the CP and the CP is not ready. IP_CONFIG did, however, expect an acknowledgment for the executed job.
0	1	$8A05_H$	A job is running, there has, however, not been an acknowledgment yet. IP_CONFIG, however, triggered the first job to read a data record.
0	1	$8A06_H$	A job is completed. IP_CONFIG, however, triggered the first job to read a data record.
Errors detected when evaluating IP_CONFIG on the CP.			
0	1	$8B01_H$	Communication error The DB could not be transferred.
0	1	$8B02_H$	Parameter error Double parameter field
0	1	$8B03_H$	Parameter error The subfield in the parameter field is not permitted.
0	1	$8B04_H$	Parameter error The length specified in the instruction does not match the length of the parameter fields / subfields.
0	1	$8B05_H$	Parameter error The length of the parameter field is invalid.
0	1	$8B06_H$	Parameter error The length of the subfield is invalid.
0	1	$8B07_H$	Parameter error The ID of the parameter field is invalid
0	1	$8B08_H$	Parameter error The ID of the subfield is invalid
0	1	$8B09_H$	System error The connection does not exist
0	1	$8B0A_H$	Data error The content of the subfield is not correct.
0	1	$8B0B_H$	Structure error A subfield exists twice.
0	1	$8B0C_H$	Data error The parameter does not contain all the necessary parameters.
0	1	$8B0D_H$	Data error The CONF_DB does not contain a parameter field for system data.
0	1	$8B0E_H$	Data error / structure error The CONF_DB type is invalid.
0	1	$8B0F_H$	System error The CP does not have enough resources to process CONF_DB completely.
0	1	$8B10_H$	Data error Configuration by the user program is not set.

DONE	ERROR	STATUS	Meaning
0	1	$8B11_H$	Data error The specified type of the parameter field is invalid.
0	1	$8B12_H$	Data error Too many connections were specified (either in total or too many for a specific type; for example, only one Email connection is possible).
0	1	$8B13_H$	CPinternal error
0	1	$8B14_H$	The active protection level does not permit the action of making changes.
Further errors detected on the program interfaces within the CPU (system instruction errors).			
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Area error reading a parameter.
0	1	$8F25_H$	Area error writing a parameter.
0	1	$8F28_H$	Alignment error reading a parameter.
0	1	$8F29_H$	Alignment error writing a parameter.
0	1	$8F30_H$	The parameter is in the writeprotected first current data block.
0	1	$8F31_H$	The parameter is in the writeprotected second current data block.
0	1	$8F32_H$	The parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error
0	1	$8F3A_H$	The target area was not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented.
0	1	$8F45_H$	Access to a parameter to be written during execution of the instruction is prevented.
0	1	$8F7F_H$	Internal error For example, an illegal ANY reference was detected.

See also

IP_CONFIG for programmed communications connections (Page 6602)

Configuration data block

Configuration data block (CONF_DB)

Meaning

The configuration data block (CONF_DB) contains all the connection data and configuration data (IP address, subnet mask, default router, NTP time server and other parameters) for an Ethernet CP. The configuration data block is transferred to the CP with the IP_CONFIG instruction.

Block and Data Structure

The graphic below shows the following:

- Structure resulting from parameter fields and subfields
  - The connections and specific system data are is described by an identically structured parameter field.
  - Individual parameters are characterized by subfields.
- Offset range
  The CONF_DB can start at any point within a data block as specified by an offset range. The address (or offset) must simply be an even number.

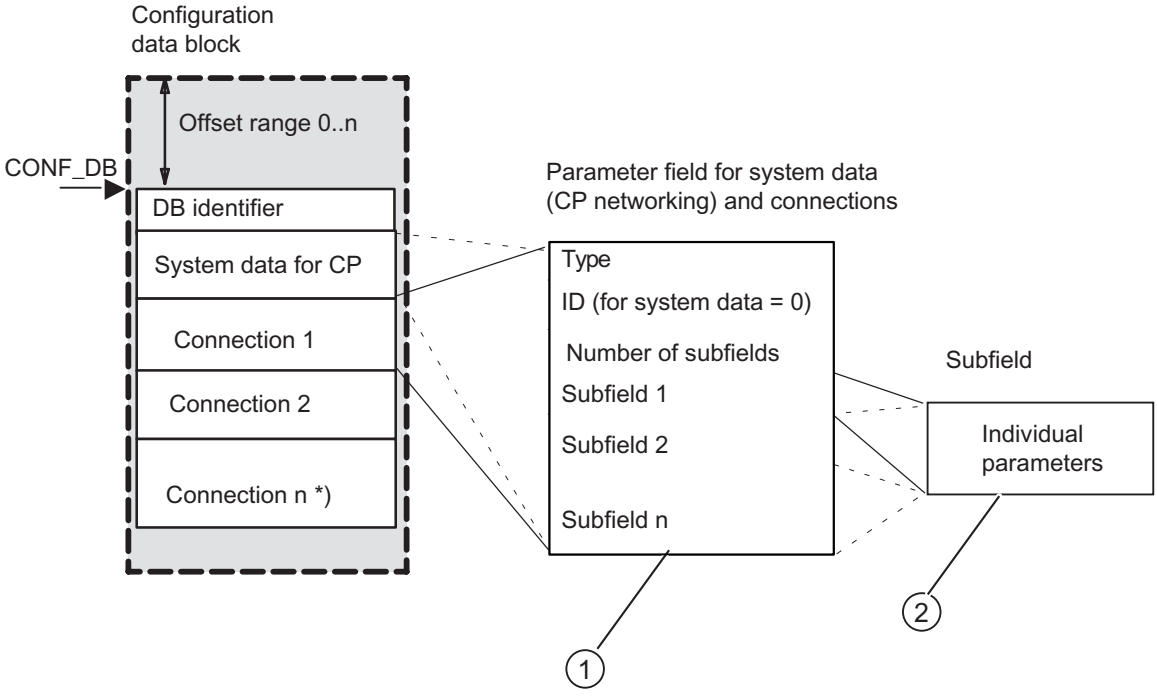

① Parameter fields are described below in Parameter field for system data (IP configuration) (Page 6608)

② Subfield types are described below in Parameter fields for connection types (Page 6612)

Parameter field for system data (IP configuration)

Meaning

Below, you will find the parameter field for system data relevant to the IP configuration of the CP and the subfields that need to be specified in it.

Some applications do not require all the subfield types - refer to the table for details.

Layout

On CPs with several interfaces, the structure of the parameter field described below applies only to the PROFINET interface.

• **Type = 0**
• **ID = 0**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
1	SUB_IP_V4	Local IP address	++
2	SUBNET_MASK	-	++
8	SUB_DEF_ROUTER	-	+
4	SUB_DNS_SERV_ADDR *)	This subfield can occur from 0 to 4 times. The first entry is the primary DNS server.	+
14	SUB_DHCP_ENABLE	0: No DHCP 1: DHCP	+
15	SUB_CLIENT_ID	- Note: Only useful when SUB_DHCP_ENABLE = 1	+
30**)	SUB_DEVICE_NAME	Device name complying with PROFINET IO convention Enter a device name to make the device individually recognizable for analysis and diagnostics in the network.	+
*)The subfield type is used only for E-mail connections. **) ID is supported only be certain CP types. ***) ++ = mandatory; + = optional			

See also

Subfield types (Page 6609)

Subfield types

Different parameters are required depending on the parameter field. Each parameter is described by a subfield. Which subfields are required is explained in the descriptions of the system data and the connection types in the previous sections.

Each subfield consists of the specific parameter section and the header (4 Byte).

Example

The following excerpt from a CONF_DB illustrates the structure of a subfield based on the example of the SUBNET_MASK subfield type.

Address	Name	Type	Initial value	Comment
+14.0	Sub_field_2	STRUCT		// Subfield 2 type SUB-NET_MASK
+0.0	Sub_field_ID	INT	2	// Subfield ID
+2.0	Sub_field_len	INT	8	// Total length of the subfield in bytes
+4.0	Parameter	STRUCT		Parameter range of SUB-NET_MASK
+0.0	Value_1	BYTE	B#16#FF	
+1.0	Value_2	BYTE	B#16#FF	
+2.0	Value_3	BYTE	B#16#FF	
+3.0	Value_4	BYTE	B#16#0	
=4.0	END_	STRUCT		
=8.0	END_	STRUCT		

In total, the following subfield types are available:

Subfield ID [1)]	Subfield type	Subfield length (in bytes)	Meaning of the Parameter
1	SUB_IP_V4	4 + 4	IP address according to IPv4
2	SUBNET_MASK	4 + 4	Subnet mask
3	SUB_DNS_NAME	Length of DNS name + 4	DNS name
4	SUB_DNS_SERV_ADDR	4 + 4	DNS server address
8	SUB_DEF_ROUTER	4 + 4	IP address of default router
9	SUB_LOC_PORT	2 + 4	Local port
10	SUB_REM_PORT	2 + 4	Remote port, also for Email connections
11	SUB_LOC_TSAP	TSAP length + 4	Local TSAP
12	SUB_REM_TSAP	TSAP length + 4	Remote TSAP
13	SUB_EMAIL_SENDER	Length of the sender E-mail address + 4	Email address of the sender
14	SUB_DHCP_ENABLE	2 + 4	Obtain an IP address from a DHCP server • Range of values: 0 = no DHCP 1 = DHCP (optional)
15	SUB_CLIENT_ID	Length of the client ID + 4	(optional)
18	SUB_CONNECT_NAME	Length of the name + 4	Name of the connection Possible characters are: a...z, A...Z, 0...9, -, _

Subfield ID [1)]	Subfield type	Subfield length (in bytes)	Meaning of the Parameter
19	SUB_LOC_MODE	1 + 4	Local mode of the connection • Range of values: 0x00 = SEND/RECV 0x01 = FTP protocol (TCP connection only) 0x10 = S5 addressing mode for FETCH/WRITE *) 0x20 = SPEED SEND/RECV (permitted only for CP 443-1 Advanced) 0x80 = FETCH *) 0x40 = WRITE *) If you do not set the parameter, the default setting is SEND/RECV. Note: FETCH / WRITE require the passive connection establishment setting (see SUB_CON_ESTABL.
20	SUB_REM_MODE	1 + 4	Setting the mode on the communication partner. (not currently supported)
21	SUB_KBUS_ADR	5	KBUS address of the CPU (relevant only for S7-400)
22	SUB_CON_ESTABL	1 + 4	Type of connection establishment. With this option, you specify whether connection establishment from this S7 station is active or passive. • Range of values: 0 = passive 1 = active
23	SUB_ADDR_IN_DATA-BLOCK	1 + 4	Select free UDP connection. The remote node is entered in the job header of the job buffer by the user program when it calls AG_SEND. This allows any node on Ethernet/LAN/WAN to be reached. • Range of values: 1 = free UDP connection 0 = other The parameter is practical only for a UDP connection.
24	SUB_NTP_SERVER	4 + 4	The subfield defines an NTP server from which the CP can obtain its time via the NTP protocol. For the situation when one or more NTP servers are defined, up to 4 subfields of ID 24 can be defined. The subfields of ID 24 may only be installed in the system parameter field type 0 / ID 0.

Subfield ID [1]	Subfield type	Subfield length (in bytes)	Meaning of the Parameter
30	SUB_DEVICE_NAME	Length of the name + 4	Device name complying with PROFINET IO convention The device name must comply with DNS conventions, in other words; • Restriction to a total of 127 characters (letters, numbers, hyphen or period) • Parts of the name within the device name; in other words, a string between two periods, must not exceed a maximum of 63 characters. • No special characters such as umlauts (ä, ö etc.), brackets, underscore, slash, blank etc. The dash (hyphen) is the only permitted special character. • The device name must not begin or end with the "-" or "." character, nor may either of these be the last character. • The device name must not begin with numbers. • The device name must not have the format n.n.n.n (n = 0...999). • The device name must not begin with the character string "port-xyz-" (x, y, z = 0...9).
1) Note: ID numbers not listed are not currently used.			

Parameter fields for connection types

General

Below, you will see which values need to be entered in the parameter fields and which subfields are used for the various connection types.

Some applications do not require all the subfield types - refer once again to the table for details.

Connection ID

The ID parameter that precedes each connection parameter field beside the type ID is particularly important.

On programmed connections, you can assign this ID freely within the permitted range of values. You must then use this ID on the call interface of the FCs for the SEND/RECV interface to identify the connection.

Range of values for the connection ID:

- S7-400: 1,2...64
- S7-300: 1,2...16

See also

Subfield types (Page 6609)

Parameter field for TCP connection

Layout

Enter the parameters in the parameter field for TCP connections as follows:

• **Type = 1 -> ①**
• **ID = connection ID -> ②**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Legend:
① Identifier for the connection type
② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.
Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
1	SUB_IP_V4	IP address of the partner	++ *)
9	SUB_LOC_PORT	-	++
10	SUB_REM_PORT	-	++ **)
18	SUB_CONNECT_NAME	-	+
19	SUB_LOC_MODE	-	+
21	SUB_KBUS_ADR	This value is always set to 2 for CPs for the S7300 and does not need to be specified.	++ (for S7-400)
22	SUB_CON_ESTABL	-	++
*) optional for a passive connection. ***) ++ = mandatory; + = optional			

See also

Subfield types (Page 6609)

Parameter field for UDP connection

Layout

Enter the parameters in the parameter field for UDP connections as follows:

• **Type = 2 -> ①**
• **ID = connection ID -> ②**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Legend:
① Identifier for the connection type
② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.
Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
1	SUB_IP_V4	IP address of the partner	++
9	SUB_LOC_PORT	-	++
10	SUB_REM_PORT	-	++
18	SUB_CONNECT_NAME	-	+
19	SUB_LOC_MODE	-	+
21	SUB_KBUS_ADR	This value is always set to 2 for CPs for the S7300 and does not need to be specified.	++ (for S7-400)
23	SUB_ADDR_IN_DATABLOCK	If the "Free UDP connection" is selected for this parameter, the parameters SUB_IP_V4 and SUB_REM_PORT are omitted.	+
***) ++ = mandatory; + = optional			

See also

Subfield types (Page 6609)

Parameter field for an ISOonTCP connection

Layout

Enter the parameters in the parameter field for ISO-on-TCP connections as follows:

• **Type = 3 -> ①**
• **ID = connection ID -> ②**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Legend:
① Identifier for the connection type
② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.
Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
1	SUB_IP_V4	IP address of the partner	++ *)
11	SUB_LOC_TSAP	-	++
12	SUB_REM_TSAP	-	++ *)
18	SUB_CONNECT_NAME	-	+
19	SUB_LOC_MODE	-	+
21	SUB_KBUS_ADR	This value is always set to 2 for CPs for the S7300 and does not need to be specified.	++ (for S7-400)
22	SUB_CON_ESTABL	-	++
*) optional for a passive connection. ***) ++ = mandatory; + = optional			

See also

Subfield types (Page 6609)

Parameter field for an Email connection

Meaning

To send Emails, one Email connection must be set up per Advanced CP. The Email connection specifies the mail server via which all the mails sent by the Advanced CP are delivered.

Layout

Enter the parameters in the parameter field for E-mail connections as follows:

• **Type = 4 -> ①**
• **ID = connection ID -> ②**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Legend:
① Identifier for the connection type
② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.
Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
1	SUB_IP_V4	IP address of the mail server, over which the Emails are sent. You can specify an absolute or alias IP address. The use of an alias assumes that the Advanced CP knows the address of the domain name server (DNS). This entry must be made when configuring the Advanced CP in HW Config. For more detailed information refer to the online help.	++ / + *)
3	SUB_DNS_NAME	DNS name of the Email server	++ / + *)
13	SUB_EMAIL_SENDER	Email address of the sender	++
18	SUB_CONNECT_NAME	-	+
21	SUB_KBUS_ADR	This value is always set to 0 for CPs for the S7300 and does not need to be specified.	++ (for S7-400)

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
22	SUB_CON_ESTABL	-	++
) The parameters SUB_IP_V4 and SUB_DNS_NAME are mutually exclusive; one or the other parameter must be specified. *) ++ = mandatory; + = optional			

Note

Mail server ports are "wellknown ports" and do not need to be specified.

See also

Subfield types (Page 6609)

Parameter field for FTP connection

Meaning

To run an FTP job sequence between the S7 station acting as the FTP client and an FTP server, the Advanced CP must establish a connection to the S7 CPU. This connection is known as an FTP connection.

FTP connections are TCP connections, with the parameter SUB_LOC_MODE set to the "FTP" mode.

Layout

Enter the parameters in the parameter field for FTP connections as follows:

• **Type = 1 -> ①**
• **ID = connection ID -> ②**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Legend:

① Identifier for the connection type

② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.
Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
18	SUB_CONNECT_NAME	-	+
19	SUB_LOC_MODE	here: 0x01 = FTP protocol	++
21	SUB_KBUS_ADR	This value is always set to 0 for CPs for the S7300 and does not need to be specified.	++ (for S7-400)
***) ++ = mandatory; + = optional			

See also

Subfield types (Page 6609)

Instructions for FTP services (S7-300, S7-400)

FTP_CMD for FTP services (S7-300, S7-400)

Migration from FC 40-44 to FB40 (S7-300, S7-400)

Comparison of the FTP_CMD program block with older functions FC40...44

All CPs with FTP functionality support the functions FC40...44. This means that existing user programs can be used unchanged.

If you want to convert from the FTP functions FC40...44 to FTP_CMD, you will need to modify your user program.

The following table shows the FTP_CMD commands used to replace the functions FC40...44.

- Correlation is indicated by "X".
- Where there is no correlation, this is indicated by "-".

	Commands of the "CMD" parameter of FTP_CMD					
Older FTP functions FC40...44	CMD = 1	CMD = 2	CMD = 3	CMD = 4	CMD = 5	CMD = 6 CMD = 7 CMD = 17
FC40	X [1)]					
FC41		X [2)]				
FC42			X [3)]			
FC43				X [4)]		
FC44					X	
						-
[1)]...[4)] The parameters of FC40...43 and CMD 1...4 (FTP_CMD) are not identical. (See table below)						

The corresponding parameters that specify a particular function in the functions FC40...FC43 or in the commands of FTP_CMD are listed in the following table.

Parameters of the FC			Parameters in FTP_CMD (with CMD 1...4)	
FC40:	LOGIN	→	CMD = 1:	NAME_STR
FC41:	FILE_NAME	→	CMD = 2:	NAME_STR
FC42:	FILE_NAME	→	CMD = 3:	NAME_STR
FC43:	FILE_NAME	→	CMD = 4:	NAME_STR
FC40...43:	BUFFER_DB_NR	→	Omitted (replaced by instance DB)	

See also

Overview of FTP_CMD (Page 6619)

Overview of FTP_CMD (S7-300, S7-400)

Meaning

Using the FTP_CMD instruction, you can establish FTP connections and transfer files from and to an FTP server.

The advantages of FTP_CMD are as follows:

- Simplification in the user program by using a command variable instead of different function calls
- Additional function "APPEND"
  "APPEND" allows data to be appended to an existing file.
- Additional function "RETR_PART"
  "RETR_PART" allows selected data areas to be retrieved from a file.
- Additional function "CONNECT_TLS_PRIVATE"
  "CONNECT_TLS_PRIVATE" allows the establishment of secure SSL connections
- The AG_SEND (FC5) function is not required here.

Validity

FB40 can be used as of the following module types:

- As of CP 343-1 Advanced (GX30* / GX31)
- As of CP 443-1 Advanced (GX20* / GX30)
  *) "CONNECT-TLS-PRIVATE" function cannot be used.

Call interface

Call interface in FBD representation

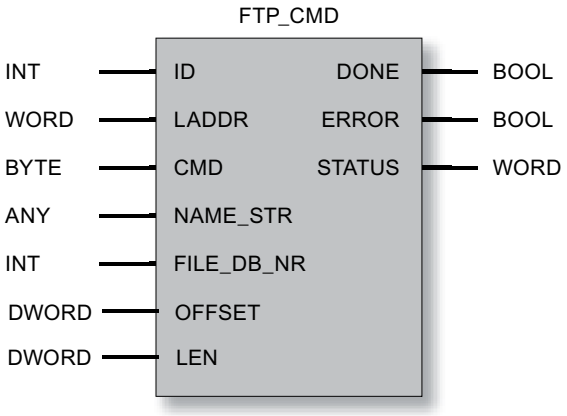

System functions called

The following system functions are called by the program block FTP_CMD:

SFC 1, SFC 20, SFC 24, SFC 58, SFC 59

Note

Note that the FTP client services of old SIMATIC S7-300 CPUs, for example the CPU 312 or CPU 315-1AF01, cannot be used because they do not support SFC24.

See also

Structure and use of the file DB (Page 6628)

Output parameters and status information FTP_CMD (Page 6625)

Input parameter - FTP_CMD (Page 6621)

Migration from FC 40-44 to FB40 (Page 6618)

Input parameter - FTP_CMD (S7-300, S7-400)

Explanation of the input parameters

Each FTP block call must be supplied with the following input parameters:

Table 4-372 Formal parameters of FB40 (FTP_CMD) - input parameters

Parameter	Declaration	Type	Range of values	Meaning / remarks
ID	INPUT	INT	For S7-300: 1, 2...16 For S7-400: 1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
LADDR	INPUT	WORD		Module start address When you call an FC, you transfer the module start address of the ADVANCED-CP in the LADDR parameter. You will find the module start address of the ADVANCED CP in the configuration of the ADVANCED CP in "Properties>Addresses>Inputs".
CMD	INPUT	BYTE	See table below - FTP commands in the "CMD" parameter	FTP commands executed when FTP_CMD is called. You will find further information following the table. If a command is not supported by the CP firmware, an error message with STATUS = $8F6B_H$ is output. Examples of FTP commands: • RETRIEVE: B#16#3 • CONNECT_TLS_PRIVATE: B#16#11

Parameter	Declaration	Type	Range of values	Meaning / remarks
NAME_STR	INPUT	ANY	Only "BYTE" is permitted as VARTYPE.	The address references a data block area. Here, you specify the address and length of the data area in which the target data is entered. • When CMD = 1, 17: With this command, the "NAME_STR" parameter specifies the FTP server to be addressed over the FTP connection with the following attributes: - IP address of the FTP server - User name - Password for the login These values must be specified as three consecutive strings in the destination range of the ANY pointer. • When CMD = 2, 3, 4, 6, 7: With this command, the "NAME_STR" parameter specifies the file name on the FTP server, in other words, the data source or data destination. The file name is specified as a string in the destination range of the ANY pointer. • When CMD = 5: Parameter not relevant You will find example of content further below.
FILE_DB_NR	INPUT	INT		The data block specified here contains the file DB to be read / written. The parameter is relevant only when CMD = 2, 3, 6 and 7.

Parameter	Declaration	Type	Range of values	Meaning / remarks
OFFSET	INPUT	DWORD		Only when CMD = 7: Offset in bytes starting at which the file will be read.
LEN	INPUT	DWORD		Only when CMD = 7: Sublength in bytes that is read starting at the value specified in "OFFSET". Special features: • If "DW#16#FFFFFFFF" is specified, the available rest of the file will be read. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET > length of the original file: The length of the destination file is displayed in this case in the ACT_LENGTH parameter in the file DB: 0 bytes on the CPU. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET + LEN > length of the original file (and LEN ≠ 0xFFFFFFFF): The length of the destination file is displayed in this case in the ACT_LENGTH parameter in the file DB: Available bytes starting at "OFFSET". Result OK (DONE = 1, STATUS = 0) if no other error occurred.

FTP commands in the "CMD" parameter

The following table explains the meaning of the commands of the "CMD" parameter and which input parameters need to be supplied. The ID and LADDR parameters must always be set to identify the connection.

CMD	Relevant input parameters (in addition to ID and LADDR)	Meaning / handling
0 (NOOP)	-	The called instruction does not execute any action. The status codes are set as follows when these parameters are supplied: • DONE=1; ERROR=0; STATUS=0
1 (CONNECT)	NAME_STR	With this command, the FTP client establishes an FTP connection to an FTP server (port 21). The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user.
2 (STORE)	NAME_STR FILE_DB_NR	This function call transfers a data block (file DB) from the FTP client (S7-CPU) to the FTP server. Caution: If the file (file DB) already exists on the FTP server, it will be overwritten.

CMD	Relevant input parameters (in addition to ID and LADDR)	Meaning / handling
3 (RETRIEVE)	NAME_STR FILE_DB_NR	This function call transfers a file from the FTP server to the FTP client (S7-CPU). Caution: If the data block (file DB) on the FTP client already contains a file, it will be overwritten.
4 (DELETE)	NAME_STR	With this function call, you delete a file on the FTP server.
5 (QUIT)	No others	With this function call, you establish the FTP connections selected with the ID.
6 (APPEND)	NAME_STR FILE_DB_NR	Similar to "STORE", the "APPEND" command saves a file on the FTP server. With "APPEND", the file on the FTP server is, however, not overwritten. The new content is appended to the existing file. If the file (file DB) does not exist on the FTP server, it will be created.
7 (RETR_PART)	NAME_STR FILE_DB_NR OFFSET LEN	Using the "RETR_PART" command (retrieve part) , you can request a section of a file from the FTP server. If very large files are involved, this allows you to restrict the read to the part you currently require. To do this, you need to know the structure of the file. Enter the required part of the file using the two parameters "OFFSET" and "LEN" in FB40.
17 (CONNECT_TLS_PRIVATE)	NAME_STR	With the "CONNECT_TLS_PRIVATE" command, the FTP client sets up an SSL-secured FTP connection (FTPS) to an FTP server (port 21). The data of the control connection and the data connection is secure. The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user. Requirements: • The "Security" option must be activated on the CP. • For the FTP connection secured with SSL, certificates must be provides in the CP configuration.

Examples of the content of the "NAME_STR" parameter

The parameter record has the following content:

Table 4-373 Content of the parameter record for CMD = 1, 17

Relative address 2)	Name	Type 1)	Example	Meaning
0.0	ip_address	STRING[100]	’142.11.25.135’	IP address of the FTP server
102.0	username	STRING[32]	'user'	User name for the login on the FTP server
136.0	password	STRING[32]	'password'	Password for the login on the FTP server
1) The maximum possible string length is specified 2) The specified values relate to the string lengths specified in "Type".				

Table 4-374 Content of the parameter record for CMD = 2, 3, 4, 6, 7

Relative address 2)	Name	Type 1)	Example	Meaning
170.0	filename	STRING[220] or STRING[212]	'plant1/tank2/press.dat'	File name of the source or destination file Note: If CMD=7 (RETR_PART), the maximum length of the file name is limited to 212 characters.
1) The maximum possible string length is specified 2) The specified values relate to the string lengths specified in "Type".				

See also

Overview of FTP_CMD (Page 6619)

Output parameters and status information FTP_CMD (S7-300, S7-400)

Introduction

For status evaluation, the following parameters must be evaluated in the user program:

Table 4-375 Formal parameters of FB40 (FTP_CMD) - output parameters

Parameter	Declaration	Type	Range of values	Meaning / remarks
DONE	OUTPUT	BOOL	0: - 1: Job executed	This parameter indicates whether or not the job was completed without errors.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code This parameter signals that the job could not be executed error-free.
STATUS	OUTPUT	WORD	See following table	Status code This parameter supplies detailed information about the execution of the job.

The DONE, ERROR and STATUS parameters are updated at every block call.

Example

During job execution, FTP_CMD returns the following codes:

- DONE=0
- ERROR=0
- STATUS=8181_H

Meaning: Job still running.

Evaluating status codes

Note

For entries coded with 8FxxH in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 4-376 FB 40: Meaning of the STATUS parameter in conjunction with DONE and ERROR

DONE	ERROR	STATUS	Meaning
0	0	0000_H	No job being executed.
1	0	0000_H	Job completed without error.
0	0	8181_H	Job active. If 8181_H is indicated permanently: The CP is not released for FTP_CMD (an illegal command for the firmware version (CMD 6, CMD 7 or CMD 17) was called.)
0	1	8090_H	• No module with this module start address exists. • The block being used does not match the system family being used (remember to use different blocks for S7300 and S7400).
0	1	8091_H	Module start address not at a doubleword boundary
0	1	8092_H	Type information in the ANY pointer is not byte
0	1	$80A4_H$	The communication bus connection between the CPU and CP is not established (with newer CPU versions). This can, for example, be caused by the following: • No connection configuration • Maximum number of CPs operating at the same time was exceeded
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	Destination area invalid; for example, destination area > 240 bytes.
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established (with older CPU versions). (with newer CPU versions, see $80A4_H$)
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	Module start address incorrect.
0	1	8183_H	The configuration does not match the job parameters.
0	1	8184_H	Illegal data type specified for the NAME_STR parameter.
0	1	8186_H	ID parameter invalid. ID = 1, 2....64
0	1	8302_H	

DONE	ERROR	STATUS	Meaning
0	1	$8F22_H$	Source area invalid, for example: Area does not exist in the DB
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Area error reading a parameter
0	1	$8F28_H$	Alignment error reading a parameter
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error
0	1	$8F3A_H$	Area not loaded (DB)
0	1	$8F50_H$	File DB DB 0 or DB does not exist
0	1	$8F51_H$	Specified file DB data area larger than existing data area
0	1	$8F52_H$	File DB in write-protected memory
0	1	$8F53_H$	File DB max. length < current length
0	1	$8F54_H$	File DB does not contain any valid data.
0	1	$8F55_H$	Header status bit: Locked
0	1	$8F56_H$	The NEW bit in the file DB header was not reset
0	1	$8F57_H$	The FTP client does not have write access to the file DB but rather the FTP server (header status bit: WriteAccess).
0	1	$8F60_H$	Bad user data, for example bad IP address of the FTP server
0	1	$8F61_H$	FTP server not obtainable
0	1	$8F62_H$	Possible meanings: • Job not supported or rejected by FTP server • The FTP server does not support SSL-secured connections.
0	1	$8F63_H$	File transfer aborted by the FTP server
0	1	$8F64_H$	Error on the FTP control connection; data could not be sent or received; the FTP control connection must be established again after such an error.
0	1	$8F65_H$	Error on the FTP data connection; data could not be sent or received. The job must be called again. This error can, for example, be caused by RETRIEVE (CMD=3) when the addressed file is already open on the FTP server.
0	1	$8F66_H$	Error reading/writing data from/to the CPU (for example DB does not exist or too short)
0	1	$8F67_H$	Error in the FTP client on the CP; for example attempting to open more than the maximum number of FTP connections.
0	1	$8F68_H$	The job was rejected by the FTP client. This error can, for example, be caused by RETRIEVE (CMD=3) when the value for the parameter MAX_LENGTH was selected too low in the file DB header.
0	1	$8F69_H$	The FTP connection is in an incorrect status, for example: • The connection is called without a previous connection termination (with the same connection ID) • There is a connection termination for a connection that has already been terminated: • A STORE command was sent on a connection that is not established.

DONE	ERROR	STATUS	Meaning
0	1	$8F6A_H$	The connection could not be established due to a temporary resource bottleneck. Remedy: Repeat the block call.
0	1	$8F6B_H$	Possible causes: • Wrong value for the CMD parameter • An FTP_CMD command is not supported. Possible cause: Wrong firmware on the CP Remedy: Firmware update
0	1	$8F6C_H$	A value > 7FFF FFF_H was set in the OFFSET parameter.
0	1	$8F6D_H$	The FTP client does not support SSL-secured connections.
0	1	$8F6F_H$	Possible causes: • The certificate contains an invalid value for "notBefore". • The certificate is invalid. The "notBefore" entry contains a time after the current time.
0	1	$8F70_H$	Possible causes: • The certificate contains an invalid value for "notAfter". • The certificate has elapsed: The "notAfter" entry contains a time before the current time.
0	1	$8F71_H$	The issuer certificate of a non-trustworthy certificate could not be found.
0	1	$8F72_H$	The original CA certificate is invalid. This is either not a CA certificate or its expansions are not consistent with the intended purpose.
0	1	$8F73_H$	The original CA certificate is marked as non-trustworthy for the specified purpose.
0	1	$8F74_H$	Other errors occurred during the verification of a certificate.
0	1	$8F7F_H$	Internal error, for example illegal ANY reference

See also

Overview of FTP_CMD (Page 6619)

Structure and use of the file DB (S7-300, S7-400)

Structure of the data blocks (file DBs) for FTP services - FTP server mode (S7-300, S7-400)

Procedure

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length (20 bytes) and structure)
- Section 2: User data (has a variable length and structure)

File DB header for FTP server mode

Note: The file DB header described here is largely identical to the file DB header for client mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data ("file does not exist"). • 1: The file DB contains valid user data ("file exists").	The dele FTP command sets EXIST=0; The store FTP command sets EXIST=1;
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The stor and retr FTP commands set LOCKED=1 when they are executed. The following function is also possible when writing from the user program: The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program: 1. Check LOCKED bit; if = 0 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit; if = 0 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0

Parameter	Type	Value / meaning	Supply
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program on the S7-CPU must set NEW=0 to allow store to be used again or to be able to delete the file with the dele FTP command.
WRITE_ACCESS	BOOL	0: The FTP client on the PG/PC has no write access rights for the file DBs on the S7 CPU. 1: The FTP client on the PG/PC has write access rights for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	This parameter is irrelevant in FTP server mode.	Is set to "0" by the FTP server.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the IT-CP (the initialization value is 1.1.1994 0.0 (midnight)).

Structure of the data blocks (file DBs) for FTP services - FTP client mode (S7-300, S7-400)

Procedure

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)

File DB header for FTP client mode

Note: The file DB header described here is largely identical to the file DB header for server mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data ("file does not exist"). • 1: The file DB contains valid user data ("file exists").	The dele FTP command sets EXIST=0; The stor FTP command sets EXIST=1;
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The stor and retr FTP commands set LOCKED=1 when they are executed. The following function is also possible when writing from the user program: The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program: 1. Check LOCKED bit; if = 0 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit; if = 0 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program in the S7-CPU must set NEW=0 to allow a new retr command.

Parameter	Type	Value / meaning	Supply
WRITE_ACCESS	BOOL	0: The user program (FTP client blocks) has write access rights for the file DBs on the S7 CPU. 1: The user program (FTP client blocks) has no write access rights for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	Unsigned integer (16-bit) containing the last reply code from FTP as a binary value. The content of this field is only valid when EXIST = 1.	This is updated by the FTP client when the FTP command is executed.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the IT-CP (the initialization value is 1.1.1994 0.0 (midnight)).

FILE_DB_HEADER data block as a template (S7-300, S7-400)

Meaning

The data type FILE_DB_HEADER is predefined for creating a file DB header.

Procedure

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)

Follow the steps outlined below:

1. For the PLC CPU on which you create the user program with the FTP instructions, create a data block of the type "Global DB".
2. Select the line you want to use as the start line for the file DB.
3. From the drop-down list in the "Data type" column, select a structure element of the type FILE_DB_HEADER.

Result: A data structure with the header structure required for the file DB is created.

Note

"Add new block" function - type selection

When you create new data blocks, the "FILE_DB_HEADER" block type is also available under the "Type" entry in the drop-down list. Do not use this option! The data block created in this way only contains the header structure and cannot be expanded by the area required for storing user data.

FILE_DB_HEADER data block - example and template for the file DB header

In the declaration view, you will see the following structure:

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	bit08	BOOL	FALSE	reserved
+0.1	bit09	BOOL	FALSE	reserved
+0.2	bit10	BOOL	FALSE	reserved
+0.3	bit11	BOOL	FALSE	reserved
+0.4	bit12	BOOL	FALSE	reserved
+0.5	bit13	BOOL	FALSE	reserved
+0.6	bit14	BOOL	FALSE	reserved
+0.7	bit15	BOOL	FALSE	reserved
+1.0	EXIST	BOOL	FALSE	if TRUE: FileDB content is valid data
+1.1	LOCKED	BOOL	FALSE	if TRUE: FileDB is locked caused by changes of the content
+1.2	NEW	BOOL	FALSE	if TRUE: FileDB content is new and may not be overwritten
+1.3	WRITE_ACCESS	BOOL	FALSE	if TRUE: Ftp-Server of the IT-CP has write access, else Ftp-Server
+1.4	bit04	BOOL	FALSE	reserved
+1.5	bit05	BOOL	FALSE	reserved
+1.6	bit06	BOOL	FALSE	reserved
+1.7	bit07	BOOL	FALSE	reserved

Address	Name	Type	Initial value	Comment
+2.0	ACT_LENGTH	DINT	L#0	actual size of the content in bytes (not including the header of 20 bytes)
+6.0	MAX_LENGTH	DINT	L#0	max. size of the content in bytes (not including the header of 20 bytes)
+10.0	FTP_REPLY_CODE	INT	0	last reply code from the remote FTP server
+12.0	DATE_TIME	DATE_AND_TIME	DT#00-1-1-0:0:0.000	date and time of last change of the content of the FileDB
=20.0		END_STRUCT		

Differences in the modes

File DB header for FTP client mode

The file DB header described here is largely identical for the FTP client mode and FTP server mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Note the ways in which the descriptions of the parameters differ:

- Structure of the data blocks (file DBs) for FTP services - FTP server mode (Page 6628)
- Structure of the data blocks (file DBs) for FTP services - FTP client mode (Page 6631)

Instructions for ERPC-CP (S7-300)

LOGICAL_TRIGGER for the logical trigger (S7-300)

Meaning of the instruction

The LOGICAL_TRIGGER instruction is available if you want to use a logical trigger for ERPC communication.

To start a logical trigger, call the LOGICAL_TRIGGER instruction in the user program of the CPU in OB1.

Further blocks are required for the LOGICAL_TRIGGER instruction call:

- An automatically generated instance DB
- A data block "CONF_DB"
  This configuration DB contains the configuration data of the logical trigger. You create and configure the configuration DB available in the STEP 7 project.
  If you want to call more than one logical trigger, you will also need to make more than one configuration DB available.

You can change the numbers of the LOGICAL_TRIGGER instruction and the instance DB.

Validity

The LOGICAL_TRIGGER instruction can be used with the following module types:

- CP 343-1 ERPC

Call

Call interface in FBD representation

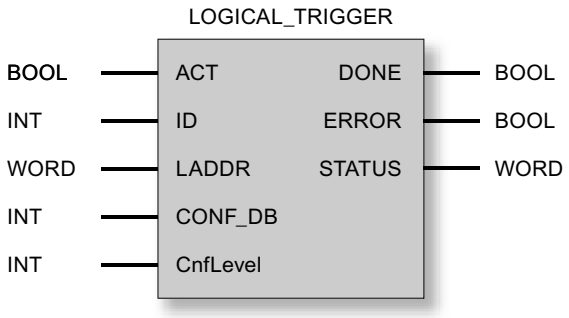

See also

The configuration data block (Page 6640)

How LOGICAL_TRIGGER works (Page 6636)

Explanation of the formal parameters for LOGICAL_TRIGGER (Page 6637)

LOGICAL_TRIGGER instruction codes (Page 6638)

How LOGICAL_TRIGGER works (S7-300)

Operating principle

The following table shows the steps involved in a trigger call by the user program of the CPU.

Step	Meaning
1	The LOGICAL_TRIGGER instruction is called at the intended point in the user program of the CPU with the corresponding instance DB and the selected configuration data block CONF_DB. • If the LOGICAL_TRIGGER instruction is called with ACT = 1, the current trigger data is read and sent to the CP firmware. • If the LOGICAL_TRIGGER instruction is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.
2	The LOGICAL_TRIGGER instruction reads the current trigger data.
3	the LOGICAL_TRIGGER instruction creates the PDU that will be sent to the CP firmware with the current trigger data.
4	The CP firmware creates the data frame and transfers it to the ERPC application.
5	The ERPC application sends the data frame to the ERP subscriber (ERP system or MES).

See also

LOGICAL_TRIGGER for the logical trigger (Page 6635)

Explanation of the formal parameters for LOGICAL_TRIGGER (S7-300)

Explanation of the formal parameters

The following table explains the formal parameters for the call interface of the LOGICAL_TRIGGER instruction:

Parameter	Declaration	Data type	Possible values	Description
ACT	INPUT	BOOL	0	If the LOGICAL_TRIGGER instruction is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.
			1	If the LOGICAL_TRIGGER instruction is called with ACT = 1, the current trigger data is read in and sent to the CP.
ID	INPUT	INT		Trigger ID This value identifies the logical trigger configured in the ILS Workbench.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table by HW Config. Specify this address here.
CONF_DB	INPUT	INT		This data block contains the configuration data of the configured logical trigger.
CnfLevel	INPUT	INT	0: Transport acknowledgment 1: End-to-end acknowledgment	Acknowledgment mode You can find the relevant acknowledgment based on the STATUS value in the codes of the LOGICAL_TRIGGER instruction. • 0 = transport acknowledgment (STATUS = 0000_H) The job is reported as successful, as soon as the data is transferred to the ERPC application. This does not necessarily mean that the data frame was sent to the ERP subscriber (ERP system or MES) and does not preclude the ERPC application detecting an error later. • 1 = end-to-end acknowledgment (STATUS = 0001_H) The job is only acknowledged after the ERPC application has checked the data. The "TriggerResponse" variable of the configuration DB (DB_CONF) is used to report whether or not the ERP subscriber could be reached an whether the ERPC application is in store-and-forward mode. Compared with the transport acknowledgment, the end-to-end acknowledgment means a longer job execution time.

Parameter	Declaration	Data type	Possible values	Description
DONE	OUTPUT	BOOL	0: Job active 1: Job completed	The parameter indicates whether or not the job for transferring the configuration data area was handled free of errors. When the job is accepted, DONE is set to 0 by the CP. As long as DONE = 0, no further job can be triggered. For the meaning in the context of the ERROR and STATUS parameters, refer to the table "LOGICAL_TRIGGER instruction codes".
ERROR	OUTPUT	BOOL	0: - 1: Error situation	Error code For the meaning in the context of the DONE and STATUS parameters, refer to the table "LOGICAL_TRIGGER instruction codes".
STATUS	OUTPUT	WORD	Refer to the table "LOGICAL_TRIGGER instruction codes".	Status code For the meaning in the context of the DONE and ERROR parameters, refer to the table "LOGICAL_TRIGGER instruction codes".

See also

LOGICAL_TRIGGER for the logical trigger (Page 6635)

LOGICAL_TRIGGER instruction codes (Page 6638)

LOGICAL_TRIGGER instruction codes (S7-300)

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Table 4-377 LOGICAL_TRIGGER instruction codes

DONE	ERROR	STATUS	Meaning
Codes relating to job execution			
1	0	0000_H	Job completed without error. The logical trigger was completed successfully.
1	0	0001_H	Job completed without error. The database is unreachable (store-and-forward mode).
0	0	8181_H	Job active.
0	1	7000_H	The LOGICAL_TRIGGER instruction was called with ACT = 0. The job will, however, not be executed. Call the LOGICAL_TRIGGER instruction at least once with ACT = 1.
Codes relating to configuration and the sequence of the logical trigger			
0	1	$80D2_H$	The CP in use does not support ERPC communication (wrong CP type).
0	1	8183_H	The CP in use does not support ERPC communication (wrong CP type).
0	1	8187_H	Invalid status of the LOGICAL_TRIGGER instruction (unknown LOGICAL_TRIGGER_STATE). Call the LOGICAL_TRIGGER instruction again.
0	1	$8A01_H$	The number of configured logical triggers equals 0.

DONE	ERROR	STATUS	Meaning
0	1	$8A02_H$	The is no configuration in the configuration DB for this logical trigger. Check the ILS Workbench configuration.
0	1	$8A03_H$	The structure of the configuration DB is incorrect. The "header identifier" does not have the correct value. Correct the value of the "ident" variable in the configuration DB (see manual of the ERPC-CP).
0	1	$8A04_H$	The structure of the configuration DB is incorrect. Download the ILS Workbench configuration to the CP again, create and configure the configuration DB(s) again (see "ERPC-CP" manual).
0	1	$8A05_H$	The configured configuration DB does not exist on the CPU.
0	1	$8A06_H$	The next call called a trigger that is still running with a different ID. Check the "ID" in the called LOGICAL_TRIGGER instruction.
0	1	$8A08_H$	The configuration data in the configuration DB does not exist or is incomplete. If the error occurs only during startup of the S7 station, the cause may be that the configuration data of the logical trigger was not completely transferred to the configuration DB. If the error continues to occur, check the configuration of the ERPC symbols.
0	1	$8A09_H$	An unknown error was reported in the configuration DB.
0	1	$8A0A_H$	The logical trigger cannot be started because a new trigger configuration is currently being loaded.
0	1	$8A0B_H$	Error identifying the time stamp of the current data record (CPU data)
0	1	$8A0C_H$	The configuration DB was created with the "Unlinked" property. Correct the object properties of the LOGICAL_TRIGGER instruction.
0	1	$8A0D_H$	Error in the the input parameter CONF_DB of the LOGICAL_TRIGGER instruction. The parameter has the value "0" or higher than the maximum DB number for the CPU.
0	1	$8A0E_H$	The transferred trigger ID is not in the permitted range of 1...16. Correct the value in the LOGICAL_TRIGGER instruction call in the user program.
0	1	$8A0F_H$	The set acknowledgment mode (CnfLevel) is invalid. Correct the value in the LOGICAL_TRIGGER instruction call in the user program.
0	1	$8Bxx_H$	Error copying the current variable values to the PDU of the logical trigger. The last two places (xx) are the variable number. Check the configuration of the symbol involved in the symbol table of the CPU and in the list of ERPC symbols in the properties dialog of the CP.
0	1	$8C01_H$	The internal status code of the LOGICAL_TRIGGER instruction is invalid. Download the ILS Workbench configuration to the CP again, create and configure the configuration DB(s) again (see "ERPC-CP" manual).
0	1	$8C02_H$	The return value of the end-to-end acknowledgment is invalid. Download the ILS Workbench configuration to the CP again, create and configure the configuration DB(s) again (see "ERPC-CP" manual).
0	1	$8C03_H$	The logical trigger contains more than 255 variables.
0	1	$8C06_H$	Error reading the data record.
0	1	$8D03_H$	The firmware is signaling a timeout during a database action.
0	1	$8D04_H$	The database application is signaling a general error in the acknowledgment of the current action.
0	1	$8E01_H$	The configured configuration DB on the CPU is not large enough. Change the size of the configuration DB.

DONE	ERROR	STATUS	Meaning
0	1	$8E06_H$	No connection has yet been established to the logical trigger.
0	1	$8EXX_H$	These status codes with values for XX_H in the range $02_H..FF_H$ are generated from an internal trigger response. If such values occur, they are relevant for service purposes.

See also

LOGICAL_TRIGGER for the logical trigger (Page 6635)

Explanation of the formal parameters for LOGICAL_TRIGGER (Page 6637)

The configuration data block (S7-300)

Preparing the configuration data block "CONF_DB"

If you use the "logical trigger" ERPC function, you will need to create a data block (DB) in STEP 7 for the configuration data of the logical trigger and specify it in the call parameters of the LOGICAL_TRIGGER instruction. The LOGICAL_TRIGGER instruction accesses DB CONF_DB. CONF_DB has no further significance for the user program.

Programming the configuration data block

To identify the newly created DB, you will need to open the DB and specify the "header identifier" and the DB size in the first two free lines.

Open the DB in STEP 7 and configure the first two free lines with the variables "ident" and "data" as follows:

Address	Name	Type	Initial value	Comment (*optional*)
*)		`STRUCT` *)		
*)	`ident`	`DWORD`	`DW#16#45525043`	*header identifier*
*)	`data`	`array[1..2048]`		*DB size (see warning below)*
*)		`Byte`		
*)		`END_STRUCT` *)		
*) Values are entered by the program				

Note

DB size

2 048 bytes are recommended as the DB size. If it becomes apparent during commissioning that this value is not enough, increase it. If a value is too low, this is reported by the LOGICAL_TRIGGER instruction with an error and the STATUS "$8A05_H$".

See also

LOGICAL_TRIGGER for the logical trigger (Page 6635)

PROFIBUS

Introduction to PROFIBUS

General information on the FCs / FBs for PROFIBUS CPs

Available instructions

The following list shows the symbolic names of the program blocks (instructions) as supplied. You cannot change these symbolic names.

Communication service / functional area	Instruction (symbolic name) 1)	Library	
		CP 300	CP 400
PROFIBUS DP	DP_SEND	x	x
	DP_RECV	x	x
	DP_DIAG	x	x
	DP_CTRL	x	x
SEND / RECEIVE (open communications services)	AG_SEND		x
	AG_RECV		x
	AG_LSEND		x
	AG_LRECV		x
S7 communication	BSEND		x
	BRCV		x
	PUT		x
	GET		x
	USEND		x
	URCV		x
	C_CNTRL		x

1) Note:
The following descriptions also include information on differences in behavior between the various versions of the instructions. Please check and note the version identifiers of the instructions you are using.
The SIMATIC Manager global libraries installed with the Automation Workbench contain the versions of the instructions current when STEP 7 was released.

Which block version should I use?

The following descriptions also include information on differences in behavior between the various versions of the instructions. Please check and note the version identifiers of the instructions you are using.

Note

We recommend that you always use the latest versions of the instructions for all module types.

You will find information on the current versions of the instructions and the current instructions to download from the Internet in our customer support:

http://support.automation.siemens.com/WW/view/en/8797900

With older module types, this recommendation assumes that you are using the latest firmware for the particular module type.

Instructions when modules are replaced

Module replacement in this sense means the replacement of a module with another module that may be a more recent version.

Note

Please remember that if you replace a module, you must only use the instructions permitted for the configured CP type in the user program.

We recommend that you always use the latest versions of the instructions for all module types. With older module types, this recommendation assumes that you are using the latest firmware for the particular module type.

You will find more information on replacing modules in our Customer Support on the Internet.

http://support.automation.siemens.com/WW/view/en/7806643

The manuals contain information on the compatibility of the S7-CPs and the corresponding instructions.

Setting parameters for block / function calls

Before describing the instruction in detail, a few general comments on calling and setting parameters for instructions will be useful at this point.

The general information below applies to the following parameter groups that exist for all instructions:

- Parameters for CP and connection assignment (input parameters)
- Parameters for specifying a CPU data area (input parameters)
- Status information (output parameters)

Calling communications instruction for an S7300

NOTICE
Calling the communications instructions for S7-300 in more than one priority class is not permitted! If, for example, you call a communication instruction in OB1 and in OB35, execution of the instruction could be interrupted by the higherpriority OB. If you call instructions in more than one OB, write your program so that a communications instruction that is currently executing cannot be interrupted by another communications instruction (for example by disabling/enabling interrupts with system instructions).

Parameters for specifying a CPU data area (input parameters)

Specifying the data area on the CPU

When you call an instruction, you transfer the address and length of the data area on the CPU in which the user data is available or will be stored or which can contain further parameter information.

The ANY pointer data type is used to address this area.

Status information (output parameters)

Evaluating status codes

For status evaluation, the following parameters must be evaluated in the user program:

- DONE or NDR
  These parameters (DONE with send jobs and NDR with receive jobs) signal (successful) completion of the job.
- ERROR
  This indicates that the job could not be executed errorfree.
- STATUS
  This parameter supplies detailed information about the execution of the job. Status codes can be returned during execution of the job (DONE=0 and ERROR=0).

Note

Remember that the status codes DONE, NDR, ERROR, STATUS are updated at each instruction call.

Status codes during CP startup

With a warm or hot restart on the Ethernet CP, the output parameters of the instruction are reset as follows:

- DONE = 0
- NDR = 0
- ERROR = 0
- STATUS = 8180H or 8181H

Instructions for open communications services (SEND/RECEIVE interface)

Instructions for FDL connections (SEND/RECEIVE interface)

Overview

The following instructions are available for the SEND/RECEIVE interface for transferring data on configured FDL connections:

Instruction	Can be used with [1]		Meaning
S7-300	S7-400		
AG_SEND	x	x	for sending data
AG_RECV	x	x	for receiving data
AG_LSEND		x	for sending data
AG_LRECV		x	for receiving data

1) Notes on the Instructions for an S7300 and S7400

To ensure the compatibility of PROFIBUS and Industrial Ethernet on the interface in the user program, you can use the AG_LSEND and AG_LRECV instructions on PROFIBUS as alternatives to AG_SEND and AG_RECV. There is no difference in the interface or the way they function. On PROFIBUS, however, you can only transfer data up to a maximum of 240 bytes even with these instructions although they are intended for longer data records on Industrial Ethernet.

This is only possible if the type and version of the instructions are permitted for the CP type you are using.

With the S7 CPs for S7300, only the AG_SEND and AG_RECV instructions are used; on Industrial Ethernet even for the transfer of longer data records.

The manuals contain information on the compatibility of the S7-CPs and the corresponding instructions. You will find an overview of the versions of the instructions in the documentation and block history.

Application

The following diagram illustrates the use of the AG_SEND / AG_LSEND and AG_RECV / AG_LRECV instructions for bi-directional data transfer on a configured FDL connection. With certain connection types, a job header should be included in the user data area.

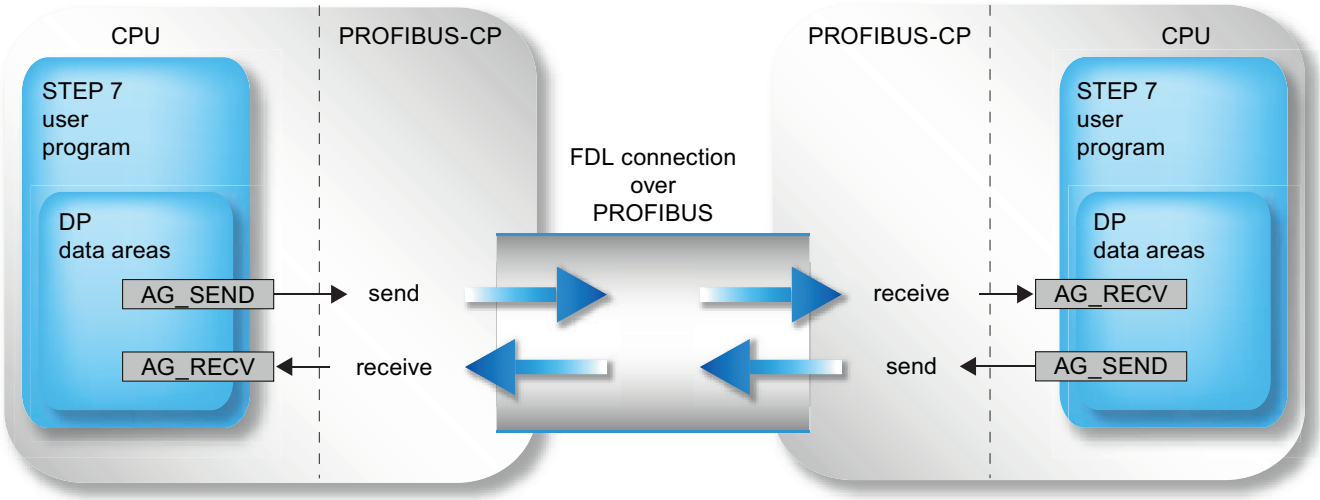

Figure 4-39 Using AG_SEND and AG_RECV on both communications partners

Application without job header

With a specified FDL connection, the address and job parameters are specified by the configuration of the connection. The user program only provides the user data in the FDL data area when sending with AG_SEND / AG_LSEND or receives the data with AG_RECV / AG_LRECV.

Up to 240 bytes of user data can be transferred. This information applies to AG_SEND and AG_LSEND on PROFIBUS.

Working with the job header

The following connection types require a job header in the FDL (user) data area:

- Unspecified FDL connection with free layer 2 access
- FDL connection with broadcast
- FDL connection with multicast

The following schematic illustrates the structure of the job buffer and the meaning and location of the parameters in the job header.

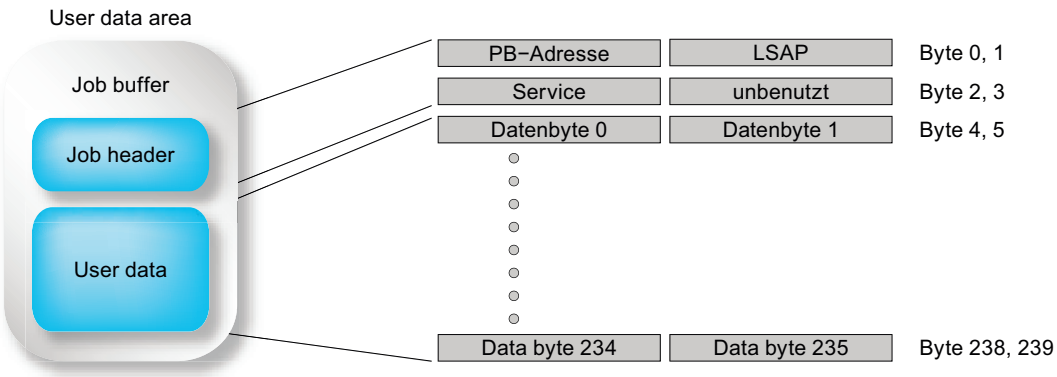

Figure 4-40 Sending and receiving via an FDL connection with programmed broadcast addressing

|The user data area can be up to 240 bytes. Up to 236 bytes of user data can be transferred. 4 bytes are reserved for the job header.

Please note that the data length specified in the instruction call (LEN parameter) must include the header and the user data!

AG_SEND / AG_LSEND

AG_SEND / AG_LSEND (PROFIBUS)

Description

The FC AG_SEND / AG_LSEND instruction transfers data to the PROFIBUS CP for transmission on a configured FDL connection.

The selected data area can be a process image area, a memory bit area or a data block area.

Error free execution of the function is indicated when the entire FDL data area could be sent on PROFIBUS.

Note:
Unless otherwise stated, all the following information applies equally to AG_SEND and AG_LSEND.

Call

Call interface in FBD representation

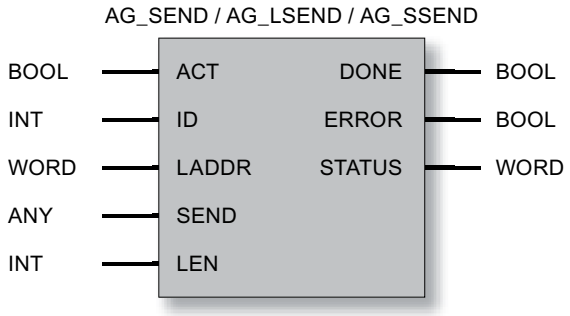

Calls with job header

The following table shows the connection types and job types for which parameters must be supplied in the job header.

The job header is located in the FDL (user) data area. It occupies the first 4 bytes and must be added to the length specified in the LEN parameter. The maximum user data length is therefore reduced for jobs with a job header to 236 bytes.

Table 4-378 Supplying the job header in the user data area

Parameter	FDL connection type		
Unspecified: free layer 2 [2]	**Broadcast**	**Multicast**	
PB address	Address of the destination station Range of values: 0 to 126 depending on node / 127 for broadcast/multicast	For AG_SEND no relevance; but area must be reserved.	For AG_SEND no relevance; but area must be reserved.
LSAP	LSAP of the destination station Range of values: 0 to 62 depending on node / 63 for broadcast	No significance but area must be reserved.	No significance but area must be reserved.
Service [1]	SDA (Send Data with Acknowledge): Value: 00_H SDN (Send Data with No Acknowledge): Value: 01_H	No significance but area must be reserved.	No significance but area must be reserved.

[1] for broadcast and multicast, only the SDN service is possible.

[2] The information on broadcast and multicast in this column is relevant only when an unspecified FDL connection is used for broadcast or multicast. On a configured FDL connection (recommended application) with broadcast or multicast as the connection partner, the address parameters are assigned automatically according to the configuration.

See also

Parameters for AG_SEND / AG_LSEND (PB) (Page 6648)
Parameters DONE, ERROR, STATUS (PB) (Page 6649)
Example of AG_SEND (PB) (Page 6651)
AG_SEND / AG_LSEND / AG_SSEND (Ind. Ethernet) (Page 6506)

Parameters for AG_SEND / AG_LSEND (PB)

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_SEND /AG_LSEND instructions:

Parameter	Declaration	Data type	Possible values	Description
ACT	INPUT	BOOL	0, 1	If the instruction is called with ACT=1, LEN bytes are sent from the ISO transport data area specified with the SEND parameter. If the instruction is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.
ID	INPUT	INT	1,2...64 (S7-400) 1,2...16 (S7-300)	The connection number of the FDL connection is specified in the parameter ID.
LADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
SEND	INPUT	ANY (only the following are permitted as VARTYPE: WORD and DWORD are permitted)		Specifies the address and length The address of the data area points to one of the alternatives: • PI area • Memory bit area • Data block area With a call with job header, the FDL data area contains the job header and the user data.
LEN	INPUT	INT	1, 2, to 240 (or up to "length specified for SEND parameter")	Number of bytes to be sent from the FDL data area with this job. The possible values range from 1 to length specified for the SEND parameter. In a call, with job header, the length information is made up of the job header (4 bytes) + user data (1 to 236 bytes). Therefore LEN >= 4 !
DONE	OUTPUT	BOOL	0: - 1: new data	The status parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the ERROR and STATUS parameters, refer to the following table.

Parameter	Declaration	Data type	Possible values	Description
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the DONE and STATUS parameters, refer to the following table.
STATUS	OUTPUT	WORD	See following table	Status code For the meaning in conjunction with the DONE and ERROR parameters, refer to the following table.

See also

AG_SEND / AG_LSEND (PROFIBUS) (Page 6646)

Parameters DONE, ERROR, STATUS (PB)

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 4-379 AG_SEND condition codes

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job completed without error.
0	0	0000_H	No job being executed.
0	0	8181_H	Job active.
0	1	7000_H	The condition code is possible only with S7-400: The instruction was called with ACT=0; the job has not yet been processed.
0	1	8183_H	No configuration or the FDL service has not yet started on the PROFIBUS CP.
0	1	8184_H	Possible causes: • Illegal data type specified for the SEND parameter. • FDL connection without job buffer: System error. • FDL connection with job buffer: Parameter LEN<4 or illegal parameter in job header (with free layer 2 access).
0	1	8185_H	LEN parameter longer than SEND source area.
0	1	8186_H	ID parameter invalid. ID!=1, 2 to 15, 16.
0	1	8301_H	SAP not activated on destination station.

DONE	ERROR	STATUS	Meaning
0	1	8302_H	No receive resources on the destination station; the receiving station cannot process received data quickly enough or has not prepared any receive resources.
0	1	8303_H	The PROFIBUS service (SDA Send Data with Acknowledge) is not supported on this SAP by the destination station. This condition code can also occur temporarily when connections or gateways are downloaded "in RUN".
0	1	8304_H	The FDL connection is not established.
0	1	8311_H	The destination station is not obtainable at the specified PROFIBUS address or the service is not possible for the specified PROFIBUS address.
0	1	8312_H	PROFIBUS error on the CP: for example, bus short-circuit, own station not in ring.
0	1	8315_H	Possible causes: • Internal parameter error on an FDL connection with job header: Parameter LEN<4 or illegal parameter in job header (with free layer 2 access). • Bus disruption Possible additional meaning: • This error code can also occur with bus problems (for example physical disturbances due to bad power connections or different settings for the transmission speed on the nodes).
0	1	$8F22_H$	Source area invalid, e.g.: Area does not exist in the DB LEN parameter < 0
0	1	$8F24_H$	Area error reading a parameter.
0	1	$8F28_H$	Alignment error reading a parameter.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error.
0	1	$8F3A_H$	Area not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F44_H$	Address of the parameter to be read is disabled in the access track.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference e.g. parameter LEN=0
0	1	8090_H	• No module with this module start address exists. • The instruction being used does not match the system family being used (remember to use different instructions for S7300 and S7400).
0	1	8091_H	Module start address not at a doubleword boundary.
0	1	8092_H	In the ANY reference, a type other than BYTE is specified. (S7-400 only)
0	1	$80A4_H$	The communication bus connection between the CPU and CP is not established. (with newer CPU versions). This can, for example, be caused by the following: • No connection configuration; • The maximum number of CPs that can be operated at one time has been exceeded (for further information, refer to the CP manual).
0	1	$80B0_H$	The module does not recognize the data record.

DONE	ERROR	STATUS	Meaning
0	1	$80B1_H$	The destination area is invalid. The amount of data to be sent exceeds the upper limit permitted for this service (e.g. destination area > 240 bytes).
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established (with older CPU versions; otherwise $80A4_H$; for further information, refer to this code)
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	Module start address incorrect.

See also

AG_SEND / AG_LSEND (PROFIBUS) (Page 6646)

Example of AG_SEND (PB)

Example of call and evaluation of parameters

Below you will find an executable example of an AG_SEND instruction call and parameter evaluation.

The OB100 listed below belongs to the "AE_460_1" instruction selected here in which the send call takes place; OB100 sets the ACT bit correctly when the CPU starts up.

To function correctly, a DB100 with a size of at least 240 bytes must be loaded.

The program requires a CP at address 256 and a configured connection of the type ISO Transport / ISOonTCP / TCP or FDL with ID=1 (please adapt your configuration where necessary !).

```
//-----------------------------------------------------------------
---
FUNCTION FC 100: VOID
TITLE = SENDE_DEMO
AUTHOR : Tester
FAMILY : S7300
NAME : FC5_Demo
VERSION : 1.0
//-----------------------------------------------------------------
---
BEGIN
    CALL FC 5 (
    ACT        := M100.0,
    ID            := 1,
    LADDR        := W#16#100,
    SEND        := P#DB100.dbx0.0 BYTE 240,
    LEN        := 240,
    DONE        := M100.1,
    ERROR    := M100.2,
    STATUS    := MW102 );
//-----------------------------------------------------------------
---
    R M100.0;          //   Reset parameter ACT for all further
    SET;               //   AG_SEND calls
    A M100.1;          //
    JC done;           //   Test for DONE = TRUE
    SET;               //
    A M100.2;          //
    JC err;            //   Test for ERROR = TRUE
//-----------------------------------------------------------------
---
    BEU;               //   Neither DONE nor ERROR is set, the job is
                       //   still running.
//-----------------------------------------------------------------
---
done:    S M100.0;     //   Job completed without error. Set ACT =
    BEU;               //   TRUE so that the following call can
                           trigger the new job.
//-----------------------------------------------------------------
---
err:    NOP 1;         //   An error occurred. The status word can be
    NOP 1;             //   evaluated here. Set ACT to TRUE in any
    S M100.0;          //   case, so that a new send job can be
    BEU;               //   triggered if the error disappears.
```

```
//-------------------------------------------------------------------
---
END_FUNCTION
ORGANIZATION_BLOCK OB100
TITLE         = Init_for_FC100
FAMILY:    S7300
NAME:        SENDE_DEMO_INIT
VERSION:    1.0
VAR_TEMP
OB1_System: array [1..20] of byte;
END_VAR
//-------------------------------------------------------------------
---
BEGIN                  //
    SET                //
    S M100.0           //   Initialize ACT parameter
END_ORGANIZATION_BLOCK
```

AG_RECV / AG_LRECV

AG_RECV / AG_LRECV (PROFIBUS)

Description

The AG_RECV / AG_LRECV instruction receives the data transferred on a configured FDL connection from the PROFIBUS CP.

The data area specified for the receive data can be a process image area, a bit address area or a data block area.

Errorfree execution is indicated when the data could be received from the PROFIBUS CP.

Note:

Unless otherwise stated, all the following information applies equally to AG_RECV and AG_LRECV.

Call interface

Call interface in FBD representation

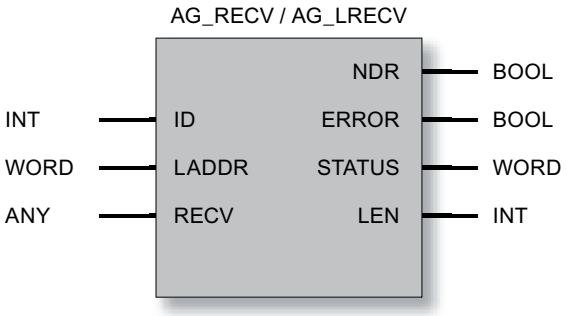

Calls with job header

Table 4-380 Return parameters in the job header in the FDL (user) data area

Parameter	FDL connection type		
Unspecified: free layer 2)	**Broadcast**	**Multicast**	
PB address	Address of the sender Range of values: 0 to 126 depending on node		
LSAP	LSAP of the sender Values: 0 to 63 depending on node		
Service	SDN indication (Send Data with No Acknowledge - Indication): Value: 01_H or SDA indication (Send Data with Acknowledge - Indication): Value: 00_H	SDN indication (Send Data with No Acknowledge - Indication): Value: $7F_H$	SDN indication (Send Data with No Acknowledge - Indication): Value: $7F_H$

See also

Parameters for AG_RECV / AG_LRECV (PB) (Page 6655)

Parameters DONE, ERROR, STATUS (PB) (Page 6656)

AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet) (Page 6511)

Parameters for AG_RECV / AG_LRECV (PB)

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_RECV / AG_LRECV instruction:

Parameter	Declaration	Data type	Possible values	Description
ID	INPUT	INT	1, 2 to 16 (S7-300) 1, 2 to 32 (S7-400)	The connection number of the FDL connection is specified in the parameter ID.
LADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
RECV	INPUT	ANY (only the following are permitted as VAR-TYPE: WORD and DWORD are permitted)		Specifies the address and length The address of the FDL data area points to one of the alternatives: • PI area • Memory bit area • Data block area With a call with job header, the FDL data area contains the job header and the user data.
LEN	OUTPUT	INT	1,2,...240	Specifies the number of bytes to be received in the FDL data area from the PROFIBUS CP. In a call, with job header, the length information is made up of the job header (4 bytes) + user data (1 to 236 bytes). Therefore LEN >= 4 !
NDR	OUTPUT	BOOL	0: - 1: new data	This parameter indicates whether new data were received. For the meaning in conjunction with the ERROR and STATUS parameters, refer to the following table.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning of this parameter in conjunction with the NDR and STATUS parameters, refer to the following table.
STATUS	OUTPUT	WORD	See following table	Status code For the meaning in conjunction with the NDR and ERROR parameters, refer to the following table.

See also

AG_RECV / AG_LRECV (PROFIBUS) (Page 6653)

Parameters DONE, ERROR, STATUS (PB)

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 4-381 AG_RECV / AG_LRECV condition codes

NDR	ERROR	STATUS	Meaning
1	0	0000_H	New data accepted.
0	0	8180_H	• There is no data available yet. The configuration is missing or the FDL service has not started on the PROFIBUS CP (occurs here instead of the code 0,1,8183_H).
0	0	8181_H	Job active.
0	1	8183_H	No configuration or the FDL service has not yet started on the PROFIBUS CP.
0	1	8184_H	• Illegal data type specified for the RECV parameter. • System error.
0	1	8185_H	Destination buffer (RECV) is too short.
0	1	8186_H	ID parameter invalid. ID!=1, 2 to 15, 16.
0	1	8303_H	The PROFIBUS service (SDA - Send Data with Acknowledge) is not supported on this SAP. This condition code can also occur temporarily when connections or gateways are downloaded "in RUN".
0	1	8304_H	The FDL connection is not established.
0	1	$8F23_H$	Source area invalid, e.g.: Area does note exist in the DB.
0	1	$8F25_H$	Area error writing a parameter.
0	1	$8F29_H$	Alignment error writing a parameter
0	1	$8F30_H$	Parameter is in the writeprotected 1st current data block.
0	1	$8F31_H$	Parameter is in the writeprotected 2nd current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error.
0	1	$8F3A_H$	Destination area not loaded (DB).
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F45_H$	Address of the parameter to be written is disabled in the access track.

NDR	ERROR	STATUS	Meaning
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference.
0	1	8090_H	• No module with this module start address exists. • The instruction being used does not match the system family being used (remember to use different instructions for S7300 and S7400).
0	1	8091_H	Module start address not at a doubleword boundary.
0	1	8092_H	In the ANY reference, a type other than BYTE is specified. (S7-400 only)
0	1	$80A0_H$	Negative acknowledgment reading from the module.
0	1	$80A4_H$	The communication bus connection between the CPU and CP is not established. (with newer CPU versions). This can, for example, be caused by the following: • No connection configuration; • The maximum number of CPs that can be operated at one time has been exceeded (for further information, refer to the CP manual).
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	Possible causes: • The destination area is invalid. • The destination area is too short. • The destination area for the received data was adequately dimensioned. Remedy: Run another receive call with maximum receive buffer size. This applies regardless of the connection type (unicast / multicast / broadcast) and the device family (S7-300 / S7-400).
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	80C3H	Resources occupied (memory).
0	1	$80C4_H$	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	Module start address incorrect.

See also

AG_RECV / AG_LRECV (PROFIBUS) (Page 6653)

Instructions for DP (distributed I/O) with S7-300

Instructions for the S7300 DP mode

Overview

The following instructions are available for the DP master and DP slave modes with an S7300:

Instruction	can be used with:		Meaning
	DP master	DP slave	
DP_SEND	X	X	for sending data
DP_RECV	X	X	for receiving data
DP_DIAG	X	-	for diagnostic functions initiated by the DP master
DP_CTRL	X	-	for control functions

Application

The following diagram illustrates the use of the DP_SEND and DP_RECV instructions on the DP master and DP slave.

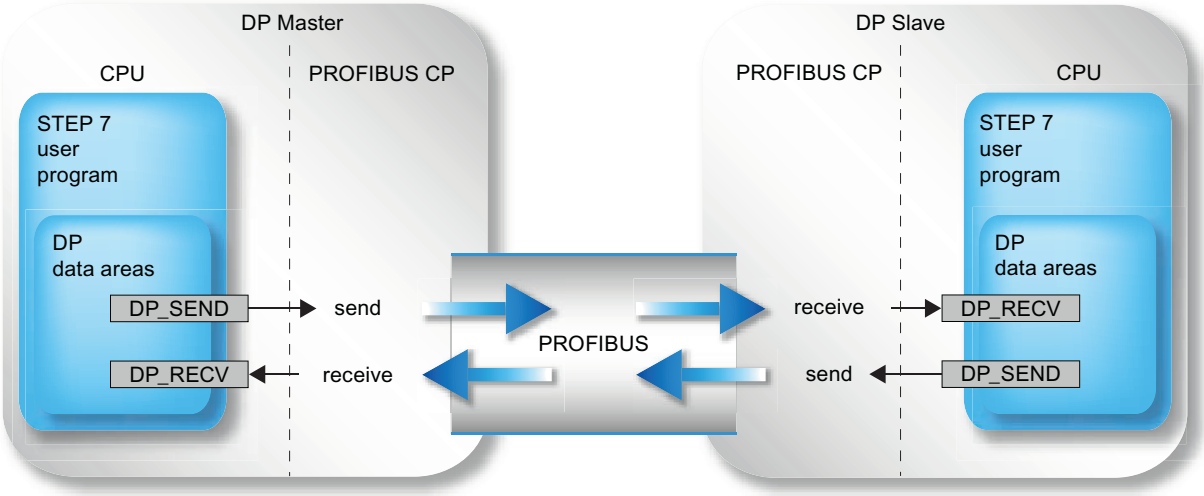

Figure 4-41 Using the DP_SEND and DP_RECV instructions with DP master and DP slave

DP_SEND

DP_SEND

Description

The DP_SEND instruction transfers data to the PROFIBUS CP. Depending on the mode of the PROFIBUS CP, DP_SEND has the following significance:

- When used in the DP master

The instruction transfers the data of a specified DP output area to the PROFIBUS CP for output to the distributed I/O system.

- When used in the DP slave

The instruction transfers the input data of the DP slave to the PROFIBUS CP for transfer to the DP master

The selected data area can be a process image area, a memory bit area or a data block area.

Correct execution is signaled when the entire DP data area could be accepted by the PROFIBUS CP.

To start the DP master exactly one DP-SEND or DP-RECV call must precede the call sequence. The following applies to this first call:

If DP-SEND is used for initialization, the transferred data area is not accepted and "0" is sent to the slaves. The user data to be transferred is accepted only with the second block call.

Call interface

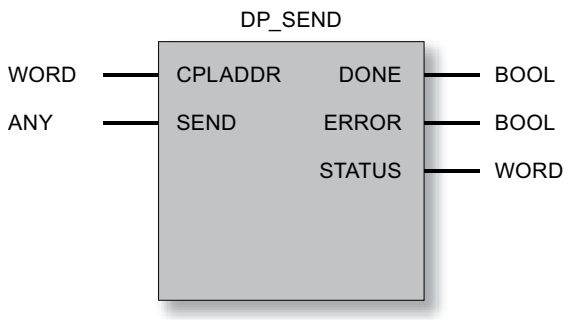

See also

Parameters for DP_SEND (Page 6660)

Parameters DONE, ERROR, STATUS (Page 6661)

Parameters for DP_SEND

Explanation of the formal parameters

The following table explains all the formal parameters for the DP_SEND instruction:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
SEND	INPUT	ANY (only the following are permitted as VAR-TYPE: With DP_SEND as of V3: BYTE With DP_SEND up to V2.x: BYTE, WORD and DWORD)		Specifies the address and length The address of the DP data area points to one of the alternatives: • PI area • Memory bit area • Data block area The length must be set for • DP master: 1...21600 • DP slave: 1...240
DONE	OUTPUT	BOOL	0: - 1: new data	The status parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the ERROR and STATUS parameters, refer toParameters DONE, ERROR, STATUS (Page 6661).
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the DONE and STATUS parameters, refer to Parameters DONE, ERROR, STATUS (Page 6661)
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the DONE and ERROR parameters, refer to Parameters DONE, ERROR, STATUS (Page 6661)

See also

DP_SEND (Page 6659)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 4-382 DP_SEND condition codes

DONE	ERROR	STATUS	Meaning
0	0	8180_H	• Startup: The DP service was started but data acceptance is not yet possible. • Normal operation Data transfer active. • DP has not started due to: – CP STOP or – "No parameter assignment" (occurs here instead of the code 0,1,8183_H).
1	0	0000_H	New data transferred without error.
0	1	8183_H	No configuration or the DP service has not yet started on the PROFIBUS CP.
0	1	8184_H	System error or bad parameter type.
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Area error reading a parameter.
0	1	$8F25_H$	Area error writing a parameter.
0	1	$8F28_H$	Alignment error reading a parameter.
0	1	$8F29_H$	Alignment error writing a parameter.
0	1	$8F30_H$	Parameter is in the writeprotected 1st current data block.
0	1	$8F31_H$	Parameter is in the writeprotected 2nd current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error.
0	1	$8F3A_H$	Destination area not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F44_H$	Address of the parameter to be read is disabled in the access track.
0	1	$8F45_H$	Address of the parameter to be written is disabled in the access track.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference.
0	1	8090_H	No module with this address exists.

DONE	ERROR	STATUS	Meaning
0	1	8091_H	Logical base address not at a double word boundary.
0	1	$80A1_H$	Negative acknowledgment writing to the module.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	The number of data bytes to be sent exceeds the upper limit for this service (applies to DP master and DP slave mode).
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	Logical base address incorrect.

See also

DP_SEND (Page 6659)

DP_RECV

DP_RECV

Description

The DP_RECV instruction receives data over PROFIBUS. DP_RECV has the following significance depending on the mode of the PROFIBUS CP:

- When used in the DP master
  DP_RECV receives the process data from the distributed I/O along with status information and enters this in a specified DP input area.
- When used on the DP slave
  DP_RECV accepts the output data transferred by the DP master in the DP data area specified in the instruction.

The data area specified for the receive data can be a process image area, a bit address area or a data block area.

Errorfree execution is signaled when the entire DP data input area could be transferred by the PROFIBUS CP.

Note that the DP_RECV instruction must be called successfully at least once on the DP slave in the user program if output data was configured for this DP slave. Please read the information in the manual.

To start the DP master exactly one DP-SEND or DP-RECV call must precede the call sequence. The following applies to this first call:

- If DP-RECV is used for initialization, the received data is not adopted. The user data to be received is accepted only with the second block call.

Additional task: Entering the status byte

The DP_RECV instruction has the following additional task:

- Updating the DP status byte DPSTATUS. This means that DP_RECV handles tasks for DP diagnostics
  If no receive data is configured, DP_RECV must be called with a length of 1 to update the DPSTATUS status byte (applies only to DP masters).
  Please read the information in the manual as well.
- Enabling the station list (see DP_DIAG (Page 6668)).

Call interface

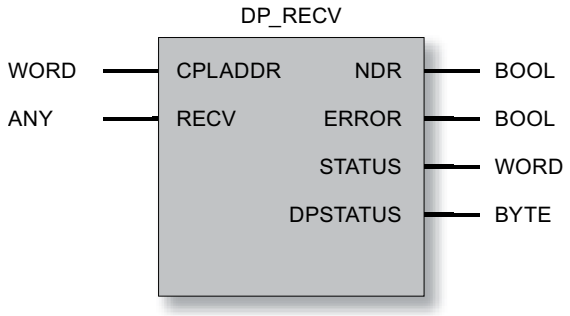

See also

Parameters for DP_RECV (Page 6664)

Parameters NDR, ERROR, STATUS (Page 6665)

DPSTATUS - DP_RECV parameters (Page 6666)

Parameters for DP_RECV

Explanation of the formal parameters

The following table explains all the formal parameters for the DP_RECV instruction:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
RECV	INPUT	ANY (only the following are permitted as VAR-TYPE: With DP_RECV as of V3: BYTE With DP_RECV up to V2.x: BYTE, WORD and DWORD)		Specifies the address and length The address of the DP data area points to one of the alternatives: • PI area • Memory bit area • Data block area The length must be set for: • DP master: 1...2160 • DP slave: 1...240 • DP master; only read status byte: 1 (see also CP manual)
NDR	OUTPUT	BOOL	0: - 1: New data accepted	The status parameter indicates whether or not new data was accepted. For the meaning in conjunction with the ERROR and STATUS parameters, refer to Parameters NDR, ERROR, STATUS (Page 6665).
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the NDR and STATUS parameters, refer to Parameters NDR, ERROR, STATUS (Page 6665)
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters NDR and ERROR, refer to Parameters NDR, ERROR, STATUS (Page 6665)
DPSTATUS	OUTPUT	Byte	For coding, see below under DPSTATUS	DP status code

See also

DP_RECV (Page 6662)

Parameters NDR, ERROR, STATUS

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

NDR	ERROR	STATUS	Meaning
0	0	8180_H	• Startup: The DP service was started but data acceptance is not yet possible. • Normal operation Data transfer active. • DP has not started due to: – CP STOP or – "No parameter assignment" (occurs here instead of the code 0,1,8183_H).
1	0	0000_H	New data accepted without error.
0	1	8183_H	No configuration or the DP service has not yet started on the PROFIBUS CP.
0	1	8184_H	System error or bad parameter type.
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Area error reading a parameter.
0	1	$8F25_H$	Area error writing a parameter.
0	1	$8F28_H$	Alignment error reading a parameter.
0	1	$8F29_H$	Alignment error writing a parameter.
0	1	$8F30_H$	Parameter is in the writeprotected 1st current data block.
0	1	$8F31_H$	Parameter is in the writeprotected 2nd current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error.
0	1	$8F3A_H$	Destination area not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F44_H$	Address of the parameter to be read is disabled in the access track.
0	1	$8F45_H$	Address of the parameter to be written is disabled in the access track.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference.
0	1	8090_H	No module with this address exists.
0	1	8091_H	Logical base address not at a double word boundary.

NDR	ERROR	STATUS	Meaning
0	1	$80A0_H$	Negative acknowledgment reading the module.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	The number of data bytes to be sent exceeds the upper limit for this service (applies to DP master and DP slave mode).
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	Logical base address incorrect.

See also

DP_RECV (Page 6662)

DPSTATUS - DP_RECV parameters

DPSTATUS

The coding of the DPSTATUS output parameter is different for the DP master mode and DP slave mode.

DP master mode

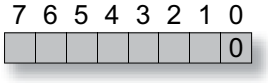

Table 4-383 Meaning of the bits in DPSTATUS in DP master mode

Bit	Meaning
7	not used
6	This bit is not set. Please read the information in the manual as well.
5, 4	Values for DPSTATUS of the DP master: 00 RUN 01 CLEAR 10 STOP (this is now the OFFLINE mode) 11 OFFLINE Please read the information in the manual as well.
3	Value 1: Cyclic synchronization is active
2	Value 0: No new diagnostic data exists Value 1: Evaluation of diagnostic list useful; at least one station has new diagnostic data

Bit	Meaning
1	Value 0: All DP slaves are in the data transfer phase Value 1: Evaluating the station list is useful
0	DP mode Value 0: DP master mode The other bits only have the specified meaning when this bit is not set.

DP slave mode

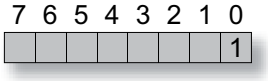

Table 4-384 Meaning of the bits in DPSTATUS in DP slave mode

Bit	Meaning
7-5	not used
4	This bit is not set. Please read the information in the manual as well.
3	This bit is not set. Please read the information in the manual as well.
2	Value 1: DP master 1 is in the CLEAR mode. The DP slave receives the value 0 in the DP data intended for the outputs. This has no effect on the send data.
1	Value 1: The configuration/parameter assignment is not yet completed.
0	Value 1: DP slave mode. **The other bits only have the specified meaning when this bit is set.**

Note

Please note, that DPSTATUS must not be evaluated until the return parameter NDR=1 is set.

See also

DP_RECV (Page 6662)

DP_DIAG

DP_DIAG

Description

The DP_DIAG instruction is used to request diagnostic information. The following types of job are possible:

- Request DP station list
- Request DP diagnostics list;
- Request DP single status;
- Read input/output data of a DP slave acyclically
- Read older DP single diagnostic information
- Read DP status.
- Read DP mode for PLC/CP stop
- Read current status of the DP slave.

Diagnostics data can also be requested for a specific slave by specifying a station address.

To transfer the diagnostic data to the CPU, you should reserve a memory area in the CPU and specify this area in the call. This memory area can be a data block area or a bit memory area. The maximum length of the available memory area must also be specified in the job.

Note

The DP_DIAG instruction is only of practical use in the DP master mode.

Exclusion

As long as this instruction is running, it must not be supplied with new job data.

Exception: Requesting the DP station list or DP diagnostics list.

Call interface

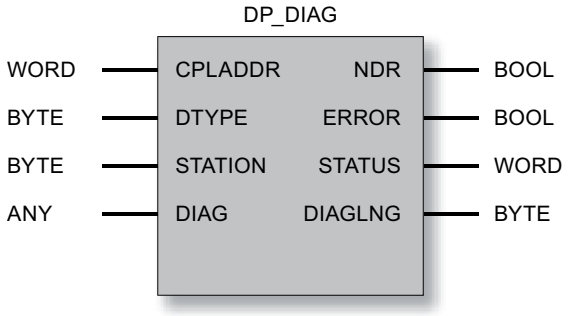

See also

Parameters for DP_DIAG (Page 6669)
Job types for DP_DIAG (Page 6670)
Ring buffer for diagnostics data - DP_DIAG (Page 6672)
Parameters NDR, ERROR, STATUS (Page 6673)

Parameters for DP_DIAG

Explanation of the formal parameters

The following table explains all the formal parameters for the DP_DIAG instruction:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
DTYPE	INPUT	BYTE	0: Station list 1: Diagnostic list 2: Current diagnostic info 3: Older diagnostics info 4: Read status 5: Read status for CPU STOP 6: Read status for CP STOP 7: Read input data (acyclically) 8: Read output data (acyclically) 10: Read current status of the DP slave	Diagnostics type
STATION	INPUT	BYTE		Station address of the DP slave
DIAG	INPUT	ANY (only the following are permitted as VARTYPE: BYTE, WORD and DWORD)	The length must be set from 1 to 240	Specifies the address and length Address of the data area. References the following alternatives: • PI area • Memory bit area • Data block area
NDR	OUTPUT	BOOL	0: - 1: new data	This parameter indicates whether or not new data were accepted. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Parameters NDR, ERROR, STATUS (Page 6673)

Parameter	Declaration	Data type	Range of values	Description
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters NDR and STATUS, refer to Parameters NDR, ERROR, STATUS (Page 6673)
STATUS	OUTPUT	WORD	See list	Status code For the meaning in conjunction with the parameters NDR and ERROR, refer to Parameters NDR, ERROR, STATUS (Page 6673)
DIAGLNG	OUTPUT	BYTE	See list	This contains the actual length (in bytes) of the data made available by the PROFIBUS CP, regardless of the buffer size specified in the DIAG parameter. The following applies to job types with DTYPE 4, 5 and 6 Here, DIAGLNG always has the value "1". The value returned in the DIAG parameter is not relevant for the evaluation in these cases. In these cases, the relevant value is contained in the STATUS parameter.

See also

DP_DIAG (Page 6668)

Job types for DP_DIAG

Job types

The following overview of the specifications for DTYPE, STATION and DIAGLNG shows the permitted or useful entries.

Table 4-385 Job types for DP_DIAG

DTYPE	Corresponds to job	Parameter STATION	DIAGLNG	Acknowledgement code (contained in the STATUS parameter; shown in Table "DP_DIAG codes")
0	Read DP station list	---	- ignored -	With the DP station list, you obtain information in the user program on the status and availability of DP slaves. The information in the DP station list relates to all DP slaves assigned to the DP master by the configuration.
1	Read DP diagnostics list	---	- ignored -	The DP diagnostics list informs the CPU program about the DP slaves with new diagnostics data.
2	Read current DP single diagnostics data	1...126	>=6	The current DP single diagnostics informs the CPU program of the current diagnostics data of a DP slave.

DTYPE	Corresponds to job	Parameter STATION	DIAGLNG	Acknowledgement code (contained in the STATUS parameter; shown in Table "DP_DIAG codes")
3	Read older DP single diagnostic information	1...126	>=6	The older DP single diagnostics informs the CPU program of the older diagnostics data of a DP slave. This data is stored on the PROFIBUS CP and read according to the "last in - first out" principle in the ring buffer. The structure of the ring buffer is explained below. If changes occur quickly in the DP slave diagnostic data, this function allows the diagnostic data of a DP slave to be acquired and evaluated in the CPU program of the DP master.
4	Read the operating status requested with DP-CTRL job (CTYPE=4)		=1	With this job, the DP operating status can be read that was set previously with the DP-CTRL job (CTYPE=4). Note: The operating status that is read out does not necessarily match the current operating status. The following operating statuses are possible: • RUN • CLEAR • STOP (is mapped to the OFFLINE status) *) • OFFLINE
5	Read DP status for CPU STOP		=1	With this job you can find out the DP status to which the PROFIBUS CP changes if the CPU changes to STOP: • RUN • CLEAR • STOP (is mapped to the OFFLINE status) *) • OFFLINE As default, the PROFIBUS CP changes to the DP status CLEAR if the CPU changes to STOP.
6	Read DP status for CP STOP		=1	With this job you can find out the DP status to which the PROFIBUS CP changes if the CP changes to STOP: • STOP (is mapped to the OFFLINE status) *) • OFFLINE As default, the PROFIBUS CP changes to the DP status OFFLINE if the CP changes to STOP.
7	Read input data	1...126	>=1	With this job, the DP master (class 2) reads the input data of the DP slave. This function is also known as shared input.

DTYPE	Corresponds to job	Parameter STATION	DIAGLNG	Acknowledgement code (contained in the STATUS parameter; shown in Table "DP_DIAG codes")
8	Read output data	1...126	>=1	With this job, the DP master (class 2) reads the output data of a DP slave. This function is also known as shared output.
10	Read current status of the DP slave	1...126	>=0	With this job, you can read out the current status of the DP slave. The following statuses are possible : • The DP master exchanges data with the DP slave cyclically. • The DP master reads the input data of the DP slave cyclically. • The DP master reads the output data of the DP slaves cyclically. • The DP master is not currently processing this DP slave cyclically.

*) The STOP status is no longer supported on the latest modules (as of module type DA02).

See also

DP_DIAG (Page 6668)

Ring buffer for diagnostics data - DP_DIAG

Ring Buffer for Diagnostic Data

The following diagram illustrates how diagnostic data is read using the "read older DP single diagnostic data" function. The first access reads the **most recent of the older diagnostic data**.

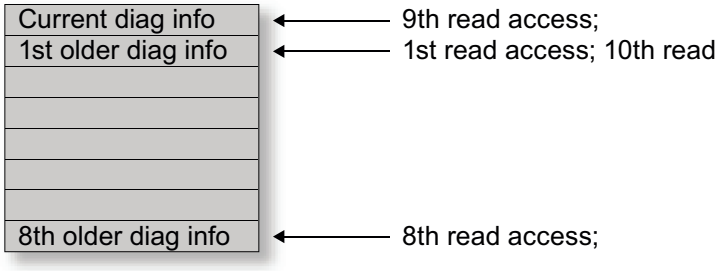

Figure 4-42 Ring Buffer for Diagnostic Data

When the current diagnostic data is read out, the read pointer is reset to the first older diagnostic data.

See also

DP_DIAG (Page 6668)

Parameters NDR, ERROR, STATUS

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 4-386 DP_DIAG codes

NDR	ERROR	STATUS	Possible with DTYPE	Meaning
0	0	8181_H	2-10	Job active. DP master not started due to • CP STOP or • No parameter assignment (occurs here instead of the code 0,1,8183_H).
0	0	8182_H	0	Triggering job pointless. DP master not started due to • CP STOP or • No parameter assignment (occurs here instead of the code 0,1,8183_H).
0	0	8182_H	1	No new diagnostic data exist. DP master not started due to • CP STOP or • No parameter assignment (occurs here instead of the code 0,1,8183_H).
1	0	0000_H	0, 1 and 4-9	Job completed without error. Note: With DTYPE 2, 3 and 10, error-free execution is indicated by a status code other than "0". Below you will see the detailed status codes for error-free execution for the range: 82XXH If an error occurs in execution, you receive status codes in the following ranges: 80XXH, 83XXH, 8FXXH

NDR	ERROR	STATUS	Possible with DTYPE	Meaning
1	0	8222_H	7,8	Job completed without error. The length of the DP slave data that was read is not the same as the data length expected by the DP master based on the module list of the DP slave in the CP database.
1	0	8227_H	7,8	Job completed without error. Message: No data exists.
1	0	8231_H	4,5,6	Job completed without error. Message: The DP status is already "RUN"
1	0	8232_H	4,5,6	Job completed without error. Message: The DP status is already "CLEAR"
1	0	8233_H	4,5,6	Job completed without error. Message: The DP status is already STOP The STOP status is now the OFFLINE status (here code 8234_H). Please read the information in the manual as well.
1	0	8234_H	4,5,6	Job completed without error. Message: The DP status is already "OFFLINE"
1	0	$823A_H$	2,3,7,8	Job completed without error. Message: 241 or 242 bytes of data were read. 240 bytes of data are available.
1	0	8241_H	2,3,10	Job completed without error. Message: The specified DP slave was not configured.
1	0	8243_H	2,3,10	Job completed without error. Message: The module list of the DP slave in the CP database only contains empty modules.
1	0	8245_H	2,3,10	Job completed without error. Message: The DP slave is in the "read input data cyclically" mode.
1	0	8246_H	2,3,10	Job completed without error. Message: The DP slave is in the "read output data cyclically" mode.
1	0	8248_H	2,3,10	Job completed without error. Note: This is the default code for the named diagnostics types if there is no special situation to signal.
1	0	8249_H	2,3,10	Job completed without error. Message: The DP slave is deactivated due to a DP mode change (e.g. CP mode selector set to STOP).
1	0	$824A_H$	2,3,10	Job completed without error. Message: The DP slave is deactivated due to a DP_CTRL job in the CPU program.
0	1	8090_H	0-10	Logical base address of the module is invalid
0	1	$80B0_H$	0-10	The module does not recognize the data record or is changing from RUN --> STOP.
0	1	$80B1_H$	0-10	Specified data record length incorrect
0	1	$80C0_H$	0-10	Data record cannot be read
0	1	$80C1_H$	0-10	The specified data record is being processed
0	1	$80C2_H$	0-10	Too many jobs pending
0	1	$80C3_H$	0-8	Resources (memory) occupied

NDR	ERROR	STATUS	Possible with DTYPE	Meaning
0	1	$80C4_H$	0-10	Communications error
0	1	$80D2_H$	0-10	Logical base address wrong
0	1	8183_H	0-10	DP master not configured.
0	1	8184_H	0-8	System error or bad parameter type.
0	1	8311_H	>=2	DTYPE parameter outside range of values.
0	1	8313_H	2,3,7,8,10	STATION parameter outside range of values.
0	1	8321_H	>=2	The DP slave is not providing any valid data.
0	1	8326_H	7,8	The DP slave has more than 242 bytes of data available. The PROFIBUS CP supports a maximum of 242 bytes.
0	1	8335_H	7,8	The PROFIBUS CP is in PROFIBUS status: "Station not in ring".
0	1	8341_H	2,3,7,8,10	The specified slave was not configured
0	1	8342_H	7,8	The DP slave with the PROFIBUS address specified in the STATION parameter is not obtainable.
0	1	8349_H	7,8	The DP master is in the OFFLINE mode.
0	1	$8F22_H$	0-10	Area length error reading a parameter (e.g. DB too short)
0	1	$8F23_H$	0-10	Area length error writing a parameter (e.g. DB too short)
0	1	$8F24_H$	0-10	Area error reading a parameter
0	1	$8F25_H$	0-10	Area error writing a parameter
0	1	$8F28_H$	0-10	Orientation error when reading a parameter
0	1	$8F29_H$	0-10	Alignment error writing a parameter
0	1	$8F30_H$	0-10	Parameter is in the writeprotected 1st current data block
0	1	$8F31_H$	0-10	Parameter is in the writeprotected 2nd current data block
0	1	$8F32_H$	0-10	The DB number in the parameter is too high
0	1	$8F33_H$	0-10	DB number error
0	1	$8F3A_H$	0-10	Area not loaded (DB)
0	1	$8F42_H$	0-10	Timeout reading a parameter from the I/O area
0	1	$8F43_H$	0-10	Timeout writing a parameter to the I/O area
0	1	$8F44_H$	0-10	Address of the parameter to be read locked in the access track
0	1	$8F45_H$	0-10	Address of the parameter to be written is disabled in the access track
0	1	$8F7F_H$	0-10	Internal error, e.g. illegal ANY reference

See also

DP_DIAG (Page 6668)

DP_CTRL

DP_CTRL

Description

The DP_CTRL instruction transfers control jobs to the PROFIBUS CP. You specify a job field (CONTROL parameter) to specify the control job in greater detail.

The following types of job are possible:

- Global control acyclic/cyclic;
- Delete older diagnostic data;
- Set current DP mode;
- Set DP mode for PLC/CP STOP;
- Read input/output data cyclically;
- Set the operating mode of the DP slave.

There are restrictions relating to the job types listed here (please refer to the information in the manual for the module).

Note

The DP_CTRL instruction is only of practical use in the DP master mode.

Connector

As long as this instruction is running, it must not be supplied with new job data.

Call interface

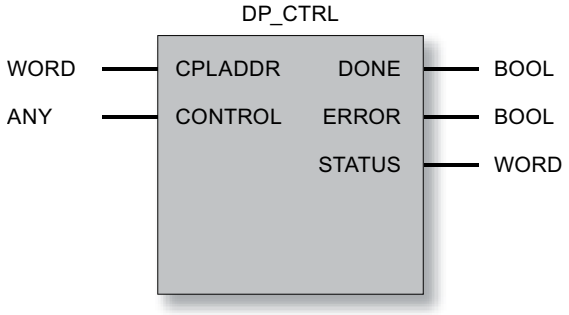

See also

Parameters for DP_CTRL (Page 6677)

Job types for DP_CTRL (Page 6679)

Command mode and group select - DP_CTRL (Page 6681)
Parameters DONE, ERROR, STATUS (Page 6682)

Parameters for DP_CTRL

Explanation of the formal parameters

The following table explains all the formal parameters for the DP_CTRL instruction:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
CONTROL	INPUT	ANY (only the following are permitted as VAR-TYPE: BYTE, WORD and DWORD)	The length must be set from 1 to 240	Specifies the address and length of the CONTROL job field Address of the data area. References the following alternatives: • PI area • Memory bit area • Data block area The length must be at least as long as the number of parameters.
DONE	OUTPUT	BOOL	0: - 1: Job executed without error.	Indicates whether the job was sent and completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 6682)
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the DONE and STATUS parameters, refer to Parameters DONE, ERROR, STATUS (Page 6682)
STATUS	OUTPUT	WORD	See following table 'Return Codes'	Status code For the meaning in conjunction with the DONE and ERROR parameters, refer to Parameters DONE, ERROR, STATUS (Page 6682)

Structure of the CONTROL job field

The control job has the following structure:

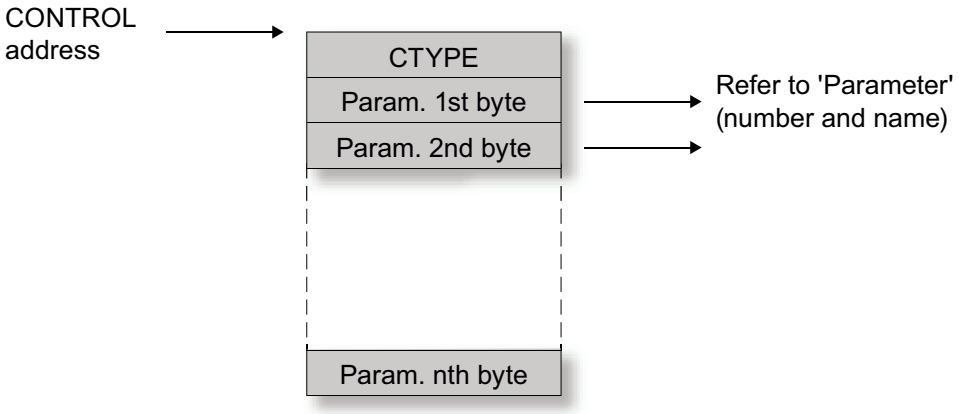

Example of the job field

With a job field as shown below, a cyclic global control job SYNC and UNFREEZE is sent for group 4 and group 5 without the autoclear option.

DB 14

Byte 0	01H	CTYPE
Byte 1	24H	Command Mode
Byte 2	18H	Group Select
Byte 3	00H	Autoclear

The length in the ANY pointer must be at least 4 (in the example, 30 has been selected).

See also

DP_CTRL (Page 6676)

Job types for DP_CTRL

Job types

Permitted or feasible specifications for the job are shown in the following overview based on the specification for CTYPE and the information in the job field.

CTYPE	Corresponds to job	Parameter in job field		Description
Name	Quantity			
0	Trigger global control	1. byte: command mode 2nd byte: group select (See section following this table.)	2	A single global control job is sent to the DP slaves selected with group select. The command mode parameter specifies the following global control jobs: • SYNC • UNSYNC • FREEZE • UNFREEZE • CLEAR - is not supported (please read the information in the manual as well) It is possible to specify more than one job in the command mode parameter.
1*)	Trigger cyclic global control	1. byte: command mode 2nd byte: group select 3rd byte: autoclear (See section following this table.)	3	The sending of cyclic global control jobs to the DP slaves selected with group select is triggered on the PROFIBUS CP. The autoclear parameter is only evaluated with the SYNC global control job. If at least one DP slave in the selected group is not in the data transfer phase and autoclear = 1 is set, the CLEAR mode is activated. In other words, the output data of the DP slaves is set to "0". The following global jobs can be activated in the command mode parameter: • SYNC • FREEZE • CLEAR (CLEAR-Bit = 1) - is not supported (please read the information in the manual as well) or deactivated: • UNSYNC • UNFREEZE • UNCLEAR (CLEAR bit = 0) It is possible to specify more than one job in the command mode parameter. An active cyclic global control job can only be terminated by a further global control job (cyclic or acyclic). To terminate the job set in the command mode, the job must be canceled. For example, the SYNC job is canceled by an UNSYNC job.
3	Delete older DP single diagnostic data	1. byte: Slave address 1..126 127 = all slaves	1	The older diagnostic data stored on the PROFIBUS CP is deleted for one or all DP slaves.

CTYPE	Corresponds to job	Parameter in job field		Description
4	Set current DP mode	1. byte: RUN = 00_H CLEAR = 01_H OFFLINE = 03_H RUN with AUTO-CLEAR = 04_H RUN without AUTO-CLEAR = 04_H	1	The DP mode can be set with this job as follows: • RUN • CLEAR • OFFLINE The AUTOCLEAR parameter means that the DP master class 1 changes to the CLEAR status automatically when the following condition is met: at least one of the DP slaves with which the DP master class 1 wants to exchange data is not in the data transfer phase. The RUN without AUTOCLEAR parameter resets AUTOCLEAR. Notes: The STOP = 02_H mode is no longer supported on the later modules (as of module type DA02). STOP = 02_H is mapped to the OFFLINE mode.
5	Set DP mode for CPU STOP	1. byte: RUN = 00_H CLEAR = 01_H OFFLINE = 03_H	1	This job specifies which DP mode the PROFIBUS CP changes to if the CPU changes to STOP: • RUN • CLEAR • OFFLINE As default, the PROFIBUS CP changes to the DP status CLEAR if the CPU changes to STOP. This mode remains set during a CP mode change from RUN --> STOP --> RUN. Notes: The STOP = 02_H mode is no longer supported on the later modules (as of module type DA02). STOP = 02_H is mapped to the OFFLINE mode.
6	Set DP mode for CP STOP	1. byte: OFFLINE=03_H	1	This job specifies which DP mode the PROFIBUS CP changes to if the CP changes to STOP. • OFFLINE As default, the PROFIBUS CP changes to the DP status OFFLINE if the CP changes to STOP. This mode remains set during a CP mode change from RUN --> STOP --> RUN. Notes: The STOP = 02_H mode is no longer supported on the later modules (as of module type DA02). STOP = 02_H is mapped to the OFFLINE mode.
7 *)	Read input data cyclically (DP master class 2)	1. byte: slave address 1 to 125	1	This job is not supported. Please read the information in the manual as well.
8 *)	Read output data cyclically (DP master class 2)	1. byte: slave address 1 to 125	1	This job is not supported. Please read the information in the manual as well.

CTYPE	Corresponds to job	Parameter in job field		Description
9	Terminate cyclic processing of the DP slave by the DP master (class 1, class 2)	1. byte: slave address 1 to 125	1	This job terminates the cyclic reading of the input data or output data of the addressed DP slave or the data transfer (DP master class 1). The DP slave is then no longer processed by the PROFIBUS CP acting as DP master (class 2). **This deactivates the DP slave.**
10	Start cyclic processing as DP master (class 1)	1. byte: slave address 1 to 125	1	The PROFIBUS CP acting as the DP master (class 1) then assigns parameters to the addressed DP slave and starts cyclic data transfer (writing outputs/reading inputs). **This activates the DP slave.**

*) This CTYPE is no longer supported on the latest modules (as of module type DA02).

See also

DP_CTRL (Page 6676)

Command mode and group select - DP_CTRL

Structure of command mode

In the command mode parameter, you specify the modes for input and output data for the cyclic and acyclic global control jobs.

The meaning is as follows:

1 = activated
0 = not activated

Bit number: 7 6 5 4 3 2 1 0

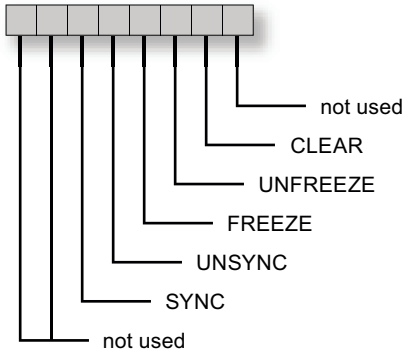

Structure of group select

In the group select parameter, you specify the group to be addressed by the control job specified in the command mode parameter. The group select parameter occupies the second byte in the control job. Each bit defines a possible DP slave group.

The meaning is as follows:

1 = assigned
0 = not assigned

Bit number:	7	6	5	4	3	2	1	0
Group;	8	7	6	5	4	3	2	1

See also

DP_CTRL (Page 6676)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the return codes formed by the DONE, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 4-387 DP_CTRL condition codes

DONE	ERROR	STATUS	Possible with CTYPE	Meaning
0	0	8181_H	0..10	Job active. DP master not started due to: • CP STOP or • "no parameter assignment" Note: The code described here occurs instead of one of the codes described later: 0, 1, 8183_H 0, 1, 8333_H 0, 1, 8334_H
1	0	0000_H	0..10	**Job completed without error.**
1	0	8214_H	0,1	**Job completed without error.** Message: Cyclic global control job is sent as acyclic global control job
1	0	8215_H	0,1	**Job completed without error.** The slaves addressed in the selected group are all deactivated.
1	0	8219_H	0,1	**Job completed without error.** An attempt was made to send an already active cyclic global control again. The global control continues unchanged.

DONE	ERROR	STATUS	Possible with CTYPE	Meaning
1	0	8228_H	0,1	**Job completed without error.** Message: The DP slaves addressed in the selected groups do not have any input modules.
1	0	8229_H	0,1	**Job completed without error.** Message: The DP slaves addressed in the selected groups do not have any output modules.
1	0	8231_H	4,5,6	**Job completed without error.** Message: The DP status is already "RUN"
1	0	8232_H	4,5,6	**Job completed without error.** Message: The DP status is already "CLEAR"
1	0	8233_H	4,5,6	**Job completed without error.** Message: The DP status is already "STOP"
1	0	8234_H	4,5,6	**Job completed without error.** Message: The DP status is already "OFFLINE"
1	0	8235_H	4	**Job completed without error.** Message: The DP status is already "RUN" with activated AUTOCLEAR
1	0	8236_H	4	**Job completed without error.** Message: The DP status is already "RUN" with deactivated AUTOCLEAR
1	0	8241_H	7-10	**Job completed without error.** Message: The specified DP slave was not configured.
1	0	8243_H	7-10	**Job completed without error.** Message: The DP slave is already deactivated since the module list of the DP slave in the CP database only contains empty modules.
1	0	8245_H	7-10	**Job completed without error.** Message: The DP slave is already in the "read input data cyclically" mode
1	0	8246_H	7-10	**Job completed without error.** Message: The DP slave is already in the "read output data cyclically" mode
1	0	8248_H	7-10	**Job completed without error.** Message: The module list of the DP slave in the CP database contains input, output, or input/output modules.
1	0	8249_H	7-10	**Job completed without error.** Message: This slave is deactivated due to a change in the DP mode.
1	0	$824A_H$	7-10	**Job completed without error.** Message: The DP slave is already deactivated due to a DP_CTRL job in the CPU program.
0	1	8090_H	0..10	No module with this address exists.
0	1	8091_H	0..10	Logical address not at a double word boundary.
0	1	$80B0_H$	0..10	The module does not recognize the data record.
0	1	$80B1_H$	0..10	The specified data record length is incorrect.
0	1	$80C0_H$	0..10	The data record cannot be read.
0	1	$80C1_H$	0..10	The specified data record is currently being processed.
0	1	$80C2_H$	0..10	There are too many jobs pending.
0	1	$80C3_H$		Resources occupied (memory).
0	1	8183_H	0..10	The DP master is not configured... Note: If the DP master is in "STOP" status, the status 8181_H can also be output.
0	1	8184_H		System error or illegal parameter type...

DONE	ERROR	STATUS	Possible with CTYPE	Meaning
0	1	8311_H	0..10	CTYPE parameter outside the range of values
0	1	8312_H	0..10	The length of the area in the CONTROL parameter is too short.
0	1	8313_H	3,7,8,9, 10	The slave address parameter is outside the range of values.
0	1	8315_H	0,1	All DP slaves of the group specified in the global control are deactivated (always occurs with an empty group).
0	1	8317_H	8	The length of the configured output data is greater than the configured receive area of the DP slave. Activating the slave mode "Read output data" is not possible.
0	1	8318_H	0,1,4,5,6	The parameter 1st byte of the job data field is outside the range of values. With GLOBAL CONTROL, CLEAR was used with SYNC or a GLOBAL CONTROL with CLEAR set was sent to group 0.
0	1	$831A_H$	0,1	At least one DP slave cannot handle FREEZE.
0	1	$831B_H$	0,1	At least one DP slave cannot handle SYNC.
0	1	8333_H	0,1	This job is not permitted in the DP "STOP" mode. Note: If no DP master is configured, the status 8181_H can also be output.
0	1	8334_H	0, 1	This job is not permitted in the DP "OFFLINE" mode. Note: If no DP master is configured, the status 8181_H can also be output.
0	1	8335_H	0, 1	The PROFIBUS CP is in PROFIBUS status: "Station not in ring".
0	1	8339_H	0, 1	At least one DP slave in the selected group is not in the data transfer phase.
0	1	$833C_H$	1	Cyclic global control must not be used in the "PLC <-> CP free running" mode. This error does not occur on the CP 3425 because this mode is not possible with this CP (PBUS data records are always used for data transfer).
0	1	8341_H	7-10	The specified DP slave was not configured.
0	1	8183_H	0..10	DP master not configured.
0	1	8184_H	-	System error or bad parameter type.
0	1	$8F22_H$	0..10	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	0..10	Area length error writing a parameter.
0	1	$8F24_H$	0..10	Area error reading a parameter.
0	1	$8F25_H$	0..10	Area error writing a parameter.
0	1	$8F28_H$	0..10	Alignment error reading a parameter.
0	1	$8F29_H$	0..10	Alignment error writing a parameter.
0	1	$8F30_H$	0..10	The parameter is in the writeprotected first current data block.
0	1	$8F31_H$	0..10	The parameter is in the writeprotected second current data block.
0	1	$8F32_H$	0..10	Parameter contains a DB number that is too high.
0	1	$8F33_H$	0..10	DB number error.
0	1	$8F3A_H$	0..10	Area not loaded (DB).
0	1	$8F42_H$	0..10	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	0..10	Timeout writing the parameter to the I/O area.
0	1	$8F44_H$	0..10	Access to a parameter to be read during execution of the instruction is prevented.

DONE	ERROR	STATUS	Possible with CTYPE	Meaning
0	1	$8F45_H$	0..10	Access to a parameter to be written during execution of the instruction is prevented.
0	1	$8F7F_H$	0..10	Internal error, e.g. illegal ANY reference.
0	1	$80C4_H$	0..10	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	0..10	Logical base address incorrect.

See also

DP_CTRL (Page 6676)

SIMATIC NET CM/CP

S7-1500 CM/CP

Industrial Ethernet

Instructions for FTP services

FTP_CMD for FTP services (S7-1500)

Overview of FTP_CMD (S7-1500)

Meaning

Using the FTP_CMD instruction, you can establish FTP connections and transfer files from and to an FTP server.

Note

Block versions

You can use the version V2.x of FTP_CMD in a station only in conjunction with a CPU and a CP V2.x V2.x.

As soon as the station obtains a CPU V1.x or CP V1.x, you must use FTP_CMD in the older version V1.x (e.g. V1.4). To do this, change the version of the "SIMATIC NET CP" library to V3.4. You can then select an older version of the block.

Data transfer is possible using FTP or FTPS (secure SSL connections).

Note

FTPS: Comparing certificates

FTPS requires a comparison of the certificates between FTP server and FTP client. If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server. Import the certificate of the FTP server as a trusted certificate in the certificate manager.

How it works

The FTP_CMD instruction references a job block (ARG) in which the FTP command is specified. Depending on the type of FTP command (CMD), this job block uses different data structures for parameter assignment. Suitable data types (UDTs) are available for these various structures.

The following diagram shows the call structure:

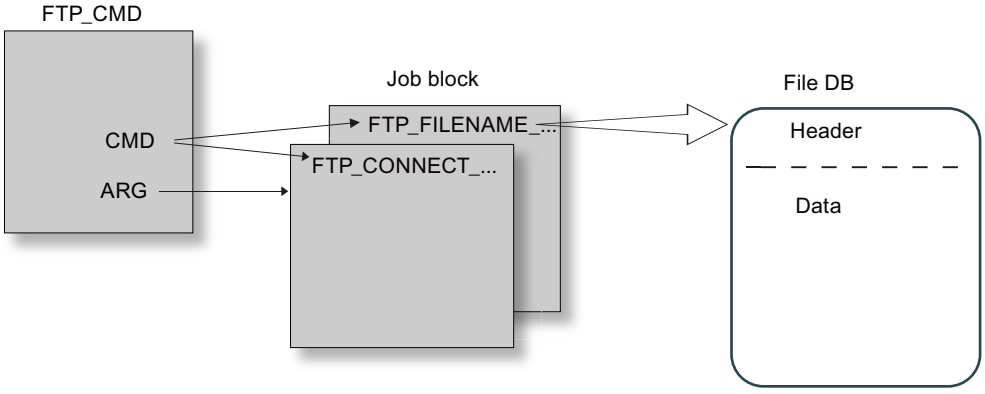

Job blocks

The following data structures are used for the job blocks:

- Connection establishment
  Various data structures are available for the connection establishment using the following types of access:
  - FTP_CONNECT_IPV4: Connection establishment with IP addresses according to IPv4
  - FTP_CONNECT_IPV6: Connection establishment with IP addresses according to IPv6
  - FTP_CONNECT_NAME: Connection establishment with server name (DNS)
- Data transfer
  For the data transfer, two different data structures are available:
  - FTP_FILENAME: Data structure for access to a complete file
  - FTP_FILENAME_PART: Data structure for read access to a data area

Data transfer in the File_DB

The data transfer is achieved using data blocks containing a header for job data and the area for the user data. The data block is specified in the job buffer.

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- For all data blocks being used as file DBs, disable the "Optimized block access" attribute.
- Only when using a CPU V1.x and a CP V1.1.x:
  Disable the "Disable PUT/GET communication" option in the configuration data of the CPU under "Properties > General > Protection" (PUT/GET must be released).

Validity

The FTP_CMD instruction can be used with the following module types:

- CP 1543-1

Call interface

Call interface in FBD representation

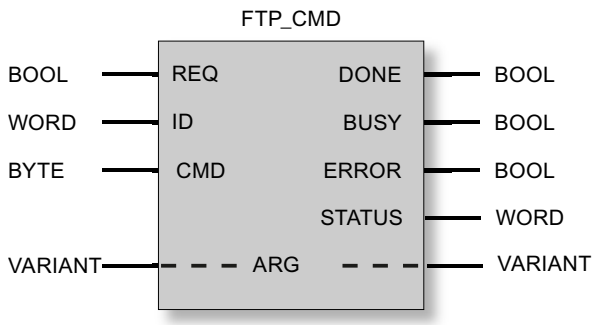

See also

Output parameters and status information FTP_CMD (Page 6694)

Input parameter - FTP_CMD (Page 6688)

Structure of the data blocks (file DBs) for FTP services - FTP client mode (Page 6698)

Input parameter - FTP_CMD (S7-1500)

Explanation of the input parameters

You supply the FTP_CMD instruction with the following input parameters:

Table 4-388 Formal parameters of the FTP_CMD instruction - input parameters

Parameter	Declaration	Data type	Memory area	Meaning / remarks
REQ	Input	BOOL	I, Q, M, D, L	Starts the send job on a rising edge.
ID *	INPUT	INT	1, 2 ... 64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
CMD *	INPUT	BYTE	See following table "Commands".	FTP command to be executed when the instruction is called. You will find value ranges for the FTP command types after the table. Note: The FTP command specified here must be specified identically in the job block (ARG parameter). If a command is not supported by the CP firmware, an error message with STATUS = $8F6B_H$ is output.
ARG *	INPUT	VARIANT	See following table "Commands".	Job block References the data area with the execution parameters suitable for the FTP command. Specific data types (UDT) are used depending on the FTP command. The UDTs are shown below. The ANY data type is not permitted for the pointer to be specified here!

* The values of the input parameters "ID" and "CMD" overwrite the value of the input parameter "ARG".

FTP commands in the "CMD" parameter

The following table shows you the significance of the commands of the "CMD" parameter and which UDTs you use to supply the job blocks.

Table 4-389 Command types

CMD (command type)	Relevant job blocks / UDT	Meaning / handling
0 (NOOP)	*	The called FC does not execute any action. The status codes are set as follows when these parameters are supplied: DONE=1; ERROR=0; STATUS=0
1 (CONNECT)	FTP_CONNECT_IPV4 FTP_CONNECT_IPV6 FTP_CONNECT_NAME	FTP connection establishment With this command, the FTP client establishes an FTP connection to an FTP server (port 21). The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user.
2 (STORE)	FTP_FILENAME	This function call transfers a data block (file DB) from the FTP client (S7-CPU) to the FTP server. Caution: If the file (file DB) already exists on the FTP server, it will be overwritten.
3 (RETRIEVE)	FTP_FILENAME	This function call transfers a file from the FTP server to the FTP client (S7-CPU). Caution: If the data block (file DB) on the FTP client already contains a file, it will be overwritten.
4 (DELETE)	FTP_FILENAME	With this function call, you delete a file on the FTP server.
5 (QUIT)	*	With this function call, you establish the FTP connections selected with the ID.
6 (APPEND)	FTP_FILENAME	Similar to "STORE", the "APPEND" command saves a file on the FTP server. With "APPEND", the file on the FTP server is, however, not overwritten. The new content is appended to the existing file. If the file (file DB) does not exist on the FTP server, it will be created.
7 (RETR_PART)	FTP_FILENAME_PART	Using the "RETR_PART" command (retrieve part) , you can request a section of a file from the FTP server. If very large files are involved, this allows you to restrict the read to the part you currently require. To do this, you need to know the structure of the file. Enter the required part of the file using the two parameters "OFFSET" and "LEN" in FB40.

* With the command types 0 (NOOP) and 5 (QUIT) a freely selectable job block (UDT) must be specified. This is not evaluated.

See also

Overview of FTP_CMD (Page 6685)

Job blocks for FTP_CMD (S7-1500)

Meaning

You supply the FTP_CMD instruction with a job block using the ARG parameter. The structure depends on the FTP command type. By using the default data types (UDT), the instruction recognizes the type of the job block. Below, you will find the relevant data types (UDTs) for the following job blocks:

- FTP connection establishment with IP address according to IPv4
- FTP connection establishment with IP address according to IPv6
- FTP connection establishment with server name
- Write and read access and other FTP commands
- FTP command RETR_PART

Job block for FTP connection establishment with IP address according to IPv4

For FTP connection establishment with IP address according to IPv4, the following data structure is used.

Table 4-390 FTP_CONNECT_IPV4

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPCmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You can find the value ranges for the command types in the section Input parameter - FTP_CMD (Page 6688). Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	'benutzer'	User name for the login on the FTP server

Parameter	Type	Range of values	Meaning / remarks
Password	STRING[32]	'passwort'	Password for the login on the FTP server
FTPserverIPaddr	IP_V4	ADDR(1) ... ADDR(4)	IP address of the FTP server as Array[1..4] of Byte, where 1 byte specifies one block of the address. Example: ADDR(1) specifies the first address block (the first byte of the address).

Job block for FTP connection establishment with IP address according to IPv6

For FTP connection establishment with IP address according to IPv6, the following data structure is used.

Table 4-391 FTP_CONNECT_IPV6

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPCmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You can find the value ranges for the command types in the section Input parameter - FTP_CMD (Page 6688). Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	'user'	User name for the login on the FTP server
Password	STRING[32]	'password'	Password for the login on the FTP server
FTPserverIPaddr	IP_V6	ADDR(1) ... ADDR(16)	IP address of the FTP server as Array[1..16] of Byte, where 2 bytes specify one block of the address. Example: ADDR(1) + ADDR(2) specify the first address block.

Job block for FTP connection establishment with server name

For FTP connection establishment specifying the server name, the following data structure is used. The server name is assigned to an IP address using DNS.

Table 4-392 FTP_CONNECT_NAME

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPcmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You can find the value ranges for the command types in the section Input parameter - FTP_CMD (Page 6688). Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	’benutzer’	User name for the login on the FTP server
Password	STRING[32]	’passwort’	Password for the login on the FTP server
FTPserverName	STRING[254]		IP address of the FTP server

Job block for write and read access and other FTP commands

The following data structure is used for the FTP commands store, retrieve, delete and append.

Table 4-393 FTP_FILENAME

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	

Parameter	Type	Range of values	Meaning / remarks
FTPcmd	BYTE	2, 3, 4, 6	FTP command "STORE / RETRIEVE / DELETE / APPEND" FTP command that executes when the instruction is called. You can find the value ranges for the command types in the section Input parameter - FTP_CMD (Page 6688). Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
DataBlockNumber	UINT		The data block specified here contains the file DB to be read / written.
LenFilename	UINT	0...1000	The "LenFilename" parameter for specifying the total length of the file name is not evaluated. Instead the length information in the string of the "Filename" parameter is evaluated.
Filename	ARRAY[0..3] OF STRING[220]		File name of the destination or source file.

Job block for the RETR_PART FTP command

The following data structure is used for the RETR_PART FTP command.

Table 4-394 FTP_FILENAME_PART

Parameter	Type	Range of values	Meaning / remarks
InterfaceID	HW_ANY		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	CONN_OUC	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablishment	BOOL	TRUE	
FTPcmd	BYTE	7	FTP command "RETR_PART" FTP command that executes when the instruction is called. You can find the value ranges for the command types in the section Input parameter - FTP_CMD (Page 6688). Note: The FTP command specified here must be specified identically in the CMD input parameter.

Parameter	Type	Range of values	Meaning / remarks
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
Offset	DWORD		Offset in bytes starting at which the file will be read.
Length	DWORD		Sublength in bytes that is read starting at the value specified in "OFFSET". Special features: • If "DW#16#FFFFFFFF" is specified, the available rest of the file will be read. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET > length of the original file: Length of the destination file (ACT_LENGTH in file DB): 0 bytes on the CPU. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET + LEN > length of the original file (and LEN ≠ 0xFFFFFFFF): Length of the destination file (ACT_LENGTH in file DB): Available bytes starting at "OFFSET". Result OK (DONE = 1, STATUS = 0) if no other error occurred.
DataBlockNumber	UINT		The data block specified here contains the file DB to be read / written.
LenFilename	UINT	0...1000	The "LenFilename" parameter for specifying the total length of the file name is not evaluated. Instead the length information in the string of the "Filename" parameter is evaluated.
Filename	ARRAY[0..3] OF STRING[220]		File name of the destination or source file.

See also

Overview of FTP_CMD (Page 6685)

Output parameters and status information FTP_CMD (S7-1500)

Parameters BUSY, DONE and ERROR

You control the execution status with the parameters BUSY, DONE, ERROR and STATUS. The BUSY parameter indicates the processing status. With the DONE parameter, you check whether or not a job was correctly executed. The ERROR parameter is set if errors occur during execution of "FTP_CMD". Error information is output in the STATUS parameter.

The following table shows the relationship between the parameters BUSY, DONE and ERROR:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was terminated with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Evaluating status codes

Note

For entries coded with 8FxxH in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 4-395 FTP_CMD: Meaning of the STATUS parameter in conjunction with DONE and ERROR

DONE	ERROR	STATUS	Meaning
0	0	0000_H	No job is being executed.
1	0	0000_H	the job was completed without error.
0	0	7001_H	The job was triggered the first time (BUSY=1)
0	0	7002_H	The job is still being executed (BUSY=1)
0	0	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).
0	0	8183_H	The configuration does not match the job parameters.
0	1	8401_H	Unknown error Possible causes: • A timeout was detected on the connection. • The FTP server has aborted the connection. Remedy: • Send the QUIT and CONNECT commands again to re-establish the connection.
0	1	8402_H	The connection has an error status. The timeout of the connection may have been exceeded or the FTP server may have terminated the connection. Remedy: Send the QUIT and CONNECT commands to re-establish the connection.
0	1	8403_H	Login has failed.
0	1	8404_H	FTP server is not obtainable.
0	1	8405_H	Transfer has failed.

DONE	ERROR	STATUS	Meaning
0	1	8406_H	Timeout for current action
0	1	8407_H	File was not found on the FTP server.
0	1	8408_H	Transfer not possible.
0	1	8409_H	File could not be fetched.
0	1	8410_H	Setting the TCP port for the data connection has failed.
0	1	8411_H	Offset information does not match.
0	1	8412_H	Error changing the specified directory
0	1	8413_H	Error receiving data
0	1	8414_H	Error sending data
0	1	8415_H	Specified CMD (command type) was rejected by the client.
0	1	8416_H	Connection was closed by the FTP server.
0	1	8418_H	Error in the user data. Possible causes: • File name is empty. • Data length is "0" • etc.
0	1	8419_H	There is no socket resource for opening a data connection.
0	1	8420_H	There is no socket resource for opening a control connection.
0	1	8421_H	Error opening the file DB to be read.
0	1	8422_H	Error opening the file DB to be written.
0	1	8423_H	Connection establishment to the FTP server has failed.
0	1	8424_H	Internal error
0	1	8425_H	Format error in the domain name
0	1	8426_H	There are too many DNS queries pending.
0	1	8427_H	The specified DNS server could not assign the specified domain name.
0	1	8428_H	There is no connection resource available.
0	1	8429_H	Unknown channel ID
0	1	8430_H	The file DB is too short.
0	1	8431_H	Error when writing to the file DB.
0	1	8432_H	Error when reading from the file DB.
0	1	8433_H	Error when accessing the file DB.
0	1	8434_H	Action was aborted.
0	1	8435_H	Channel will be reset.
0	1	8436_H	Unexpected server reply
0	1	8437_H	Certificate could not be verified.
0	1	8438_H	Unknown error occurred
0	1	8439_H	The FTP command causes an error. The cause must be looked for on the FTP server (REST command).
0	1	8440_H	The FTP server does not support the requested SSL protocol.
0	1	8446_H	After the FTP password was sent to the FTP server, an unexpected code was returned by the FTP server.
0	1	8451_H	An error was signaled when attempting to change the transmission mode from binary to ASCII.

DONE	ERROR	STATUS	Meaning
0	1	8455_H	A memory request has failed on the CM/CP.
0	1	8460_H	A problem has occurred handling SSL/TLS.
0	1	8469_H	Interface error The specified output interface could not be used. Remedy: Set the interface to be used for outgoing connections.
0	1	8475_H	The SSL certificate or the SSH md5 fingerprint was not considered trusted.
0	1	8476_H	Nothing was received from the FTP server. In the current status, an incorrect response must be assumed.
0	1	8477_H	The specified "Crypto engine" (cryptographic module) was not found.
0	1	8478_H	The attempt to set the selected SSL "Crypto engine" as the default failed.
0	1	8480_H	A problem has occurred with the certificate of the FTP client.
0	1	8481_H	The specified number could not be used.
0	1	8482_H	The FTP server uses a coding that is not supported.
0	1	8484_H	The maximum file size was exceeded.
0	1	8485_H	The file DB was modified while being processed to be sent or the file DB is incorrectly structured.
0	1	8489_H	Data could not be sent. There is not enough memory available for the action on the FTP server.
0	1	8492_H	The file already exists. The file will not be overwritten.
0	1	8496_H	A problem occurred reading the SSL CA certificate.
0	1	8497_H	An unexpected error occurred in the SSH session.
0	1	8498_H	It was not possible to terminate the SSL connection.
0	1	8499_H	The socket is not ready for sending/receiving. Wait until it is ready and try again.
0	1	8501_H	The SSL certificate check by the FTP server has failed.
0	1	8507_H	A timeout has occurred establishing the connection during the active FTP session while waiting for the FTP server.
0	1	$8F55_H$	Header status bit: Locked
0	1	$8F56_H$	The NEW bit in the file DB header was not reset
0	1	$8F6B_H$	Possible causes: • Bad value for the CMD parameter Values from 0 to 15 are permitted. • An FB40 command is not supported. Possible cause: Wrong firmware on the CP Remedy: Firmware update (with older CPs, use the functions FC 40...FC 44 instead of FB 40.)
0	1	$8F7F_H$	Internal error, for example illegal ANY reference

See also

Overview of FTP_CMD (Page 6685)

Structure and use of the file DB - FTP client mode

Structure of the data blocks (file DBs) for FTP services - FTP client mode

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data as "Array [..] of Byte" or "Array [..] of Char" (has a variable length and structure)

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.
- For all data blocks being used as file DBs, disable the "Optimized block access" attribute.

File DB header for FTP client mode

The file DB header described here is identical to the file DB header described for server mode.

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data (file does not exist). • 1: The file DB contains valid user data (file exists).	The DELETE FTP command sets EXIST=0. The STORE FTP command sets EXIST=1.
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The "STORE" and "RETRIEVE" FTP commands set LOCKED=1 when they are executed if the bit was previously at 0. The user program on the S7 CPU can also set or reset LOCKED during write access to achieve data consistency. This results in mutual locking between the user program and FTP handling to ensure consistency. Recommended sequence in the user program: 1. Check LOCKED bit (if = 0) 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit (if = 0) 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program in the S7-CPU must set NEW=0 to allow a new "RETRIEVE" command.
WRITE_ACCESS	BOOL	• 0: The user program has write access rights for the file DBs on the S7 CPU. • 1: The user program has no write access rights for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.

Parameter	Type	Value / meaning	Supply
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	Unsigned integer (16-bit) containing the last reply code from FTP as a binary value. The content of this field is only valid when EXIST = 1.	Updated by the FTP protocol handler with the FTP command processing of the server.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. The reference is the startup time of the CP (initialization value: 01.01.1994 00:00h).

See also

FILE_DB_HEADER data block as template - FTP client mode (Page 6700)

FILE_DB_HEADER data block as template - FTP client mode

Meaning

The file DB header described here is identical for the FTP client mode and FTP server mode.

The data type FILE_DB_HEADER is predefined for creating a file DB header.

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)

Follow the steps outlined below:

1. In the CPU on which you create the user program with the FTP instructions, create a data block of the type "Global DB".
2. Open the properties dialog of the newly created data block (shortcut menu), disable the "Optimized block access" attribute and then close the properties dialog.
3. In the block editor of the DB, select the line you want to use as the start line for the file DB.
4. From the drop-down list in the "Data type" column, select a structure element of the type "FILE_DB_HEADER" or enter the type "FILE_DB_HEADER" using the keyboard.
   Result: A data structure with the header structure required for the file DB is created.
5. Set the "WRITE_ACCESS" parameter to "true" to enable access.
6. Enter a value for the length of the user data at the "MAX_LENGTH" parameter, see next item.
7. Below this, create a data field of the type "Array [..] of Byte" or "Array [..] of Char" for the user data to be sent.
   The size of the field must match the specification of MAX_LENGTH in the header.

Note

"Add new block" function - type selection

When you create new data blocks, the "FILE_DB_HEADER" block type is also available under the "Type" entry in the drop-down list. Do not use this option! The data block created in this way only contains the header structure and cannot be expanded by the area required for storing user data.

FILE_DB_HEADER data block - example and template for the file DB header

In the declaration view, you will see the following structure:

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	bit08	BOOL	FALSE	reserved
+0.1	bit09	BOOL	FALSE	reserved
+0.2	bit10	BOOL	FALSE	reserved
+0.3	bit11	BOOL	FALSE	reserved
+0.4	bit12	BOOL	FALSE	reserved
+0.5	bit13	BOOL	FALSE	reserved
+0.6	bit14	BOOL	FALSE	reserved
+0.7	bit15	BOOL	FALSE	reserved
+1.0	EXIST	BOOL	FALSE	if TRUE: File DB does not contain valid data
+1.1	LOCKED	BOOL	FALSE	if TRUE: The file DB is blocked due to a change to the content.

Address	Name	Type	Initial value	Comment
+1.2	NEW	BOOL	FALSE	if TRUE: The content of the File DB is new and must not be overwritten.
+1.3	WRITE_ACCESS	BOOL	FALSE	if TRUE: The FTP server has write access.
+1.4	bit04	BOOL	FALSE	reserved
+1.5	bit05	BOOL	FALSE	reserved
+1.6	bit06	BOOL	FALSE	reserved
+1.7	bit07	BOOL	FALSE	reserved
+2.0	ACT_LENGTH	DINT	L#0	Current length of the content in bytes (not including 20 bytes for the header)
+6.0	MAX_LENGTH	DINT	L#0	Maximum length of the content in bytes (not including 20 bytes for the header)
+10.0	FTP_REPLY_CODE	INT	0	Updated by the FTP protocol handler with the FTP command processing, last response information from the FTP server.
+12.0	DATE_TIME	DATE_AND_TIME	DT#00-1-1-0:0:0.000	Date and time of the last change to the content of the file DB.
=20.0		END_STRUCT		

S7-1200 CM/CP

Telecontrol

Telecontrol instructions

TC_CON: Establish connection via the GSM network

Meaning

The TC_CON instruction allows an S7-1200 with a CP 1242-7 to establish a connection of the following types:

- ISO-ON-TCP
  Connection partner is a CP 1242-7.
  ISO-ON-TCP connections are used only in "GPRS direct" mode.
- UDP
  Any connection partner is possible.

- SMS
  The connection partner is an SMS client.
- Telecontrol connection
  The connection partner is either a telecontrol server or another station that can be reached via the telecontrol server.

A TC_CON establishes exactly one connection. Depending on the mode of the CP 1242-7 and the protocol you are using, a maximum of 3 to 5 simultaneous connections with unique IDs (see below) are supported per CP. You will find the maximum number of simultaneous connections in the performance data of the CP.

The CONNECT parameter uses a data block (DB) with the structure of a system data type (SDT) for the connection description.

The required connection type is defined using a connection-specific SDT "TCON_..." (see below). For each of the connection types listed above, one of the following SDTs must be assigned:

- TCON_IP_RFC for ISO-ON-TCP connections
- TCON_IP_V4 for UDP connections
- TCON_PHONE for SMS connections
- TCON_WDC for telecontrol connections

The "ActiveEstablished" parameter of these SDTs also specifies whether or not connection establishment is active or passive.

For parameter settings for these SDTs, see TCON_...: SDTs for the telecontrol connection establishment (Page 6718).

The ID parameter references the GPRS connection. The ID is assigned and must be unique within the CPU.

The INTERFACE parameter references the GPRS interface of the required local CP. This must be taken from STEP 7.

Call interface in FBD representation

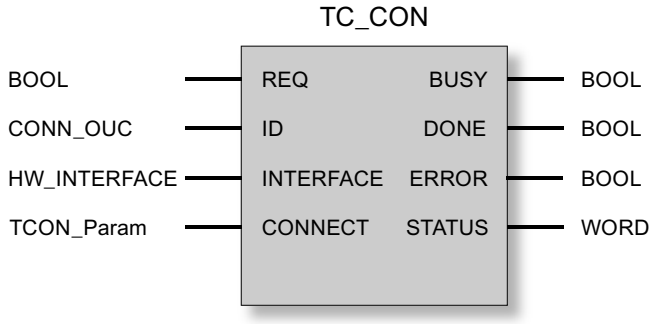

Explanation of the formal parameters

The following table explains all the formal parameters for the TC_CON instruction.

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection. The ID is assigned. The value of ID is also required by the system data type (SDT) of the CONNECT parameter.
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
CONNECT	INOUT	TCON_Param	See also "TCON_...: SDTs for telecontrol connection establishment"	Reference to a data block for connection establishment. The SDTs of the type TCON_IP_RFC, TCON_IP_V4, TCON_PHONE or TCON_WDC specify the structure of the data block suitable for the relevant connection. In the SDTs, note the parameter "Active-Established" (active / passive connection establishment).
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

When called, the instruction remains in the BUSY = 1 state for several seconds. In the following situations, the BUSY state = 1 can last for a longer time:

- On active ISO-on-TCP connections if the partner cannot be reached.
- On passive connections when no frame is received.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	8087_H	Maximum number of connections reached, no further connection possible
0	1	$80E3_H$	The ID is already being used by another connection. This means for TC_SEND, TC_RECV or TC_DISCON, BUSY is currently TRUE. The status code is output when EN_R of TC_RECV is permanently TRUE. This mostly results in TC_RECV being called. Remedy for this situation: Turn EN_R off before TC_CON or TC_DISCON is called. EN_R may only be turned on again if TC_CON executed free of errors.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80E8_H$	Remote partner cannot be reached. Check the connection parameters. In the "GPRS direct" mode, the message is output if the partner can be reached but is not accepting a connection request.
0	1	$80EB_H$	Request temporarily denied (TC_CON has already been called with the same destination address.)
0	1	$80EC_H$	Opening the Listener Port failed: Check the connection parameters.
0	1	$80F2_H$	The CP is in the wrong mode: • Telecontrol connections are permitted only in "Telecontrol" mode. • ISO-ON-TCP connections are permitted only in "GPRS direct" mode.
0	1	$80F3_H$	No free connection endpoint for sending data: • Use less connections or • Use less passive connections or • Turn off NTP. Remember the maximum number of simultaneous connections of the CP 1242-7.

DONE	ERROR	STATUS	Meaning
0	1	$80F4_H$	Connection endpoint cannot be generated: Repeat the call. If necessary, check the connection parameters.
0	1	$80F5_H$	Invalid connection endpoint: Connection establishment by TC_CON failed. Repeat the block call.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format, invalid value or phone number in TCON_Phone longer than 20 characters) Check the configuration of the "TC_CON..." SDT.

TC_DISCON: Terminate connection via the GSM network

Meaning

The TC_DISCON instruction on an S7-1200 with CP 1242-7 terminates an ISO-ON-TCP, UDP, SMS or telecontrol connection that was established with the TC_CON instruction.

You will find detailed information on the connection types in the description of the TC_CON instruction.

TC_DISCON terminates the connection to the telecontrol server only logically. At the TCP/IP level, the connection is retained.

If you want the connection to the telecontrol server to be terminated physically, configure the connection as a "Temporary connection" in the "telecontrol server" parameter group in STEP 7. Temporary stations terminate the connection automatically after sending the data.

Note

Stopping the execution of further program blocks by TC_DISCON

Calling up TC_DISCON ends the execution of TC_CON, TC_SEND, and TC_RECV blocks, that were called up with the same connection ID (parameter "ID") and interface (parameter "INTERFACE"). These blocks then signal an ERROR.

Do not call up TC_DISCON if TC_CON indicates "Error = 1".

If TC_CON indicates "ERROR", then the connection is not established. In this case TC_DISCON must not be called.

If TC_DISCON is called in this case, the connection ID ("ID") is briefly reserved, and a TC_CON called up immediately afterwards would indicate ERROR and STATUS 80E3.

The ID parameter references the GPRS connection. The ID must be unique within the CPU and the same as the ID used with TC_CON.

The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

Call interface in FBD representation

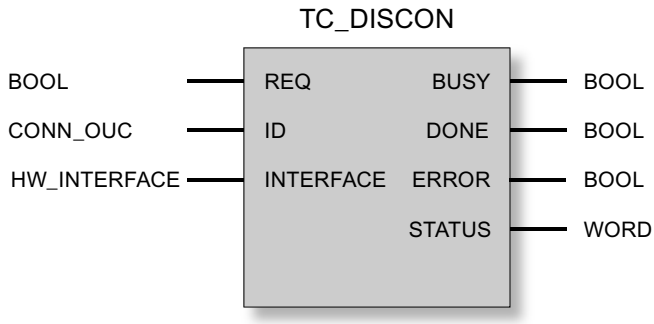

Explanation of the formal parameters

The following table explains all the formal parameters for the TC_DISCON instruction

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	The instruction has not yet been called.

You will find all other code combinations of DONE and ERROR in the following table.

Note

When called, the instruction remains in the BUSY = 1 state for several seconds.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	$80E4_H$	Unknown ID: No connection with this ID has been established by TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed or • Connection terminated by remote partner.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the configuration of the "TC_CON..." SDT.

TC_SEND: Send data via the GSM network

Meaning

The TC_SEND instruction allows the sending of data via programmed connections of the following types:

- ISO-ON-TCP connections
- UDP connections

- SMS connections
  The sending of SMS messages is supported only if this was set up in the STEP 7 configuration of the CP.
- Telecontrol connections

Note

Sending SMS messages to multiple recipients

If you want to send an identical SMS message to several recipients, you need to establish a connection to each recipient.

You will find more detailed information on the connection types in the description of the TC_CON instruction.

The ID parameter references the GPRS connection. The value of ID must correspond to the value used for ID by TC_CON.

The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

The amount of data to be sent is specified with the LEN parameter.

The size of the data area specified in DATA must be at least as large as the number of bytes configured for LEN. Permitted data types in the data area specified in DATA are all except BOOL and ARRAY of BOOL.

The destination address (connection partner) for the data to be sent is configured in the TC_CON instruction.

Call interface in FBD representation

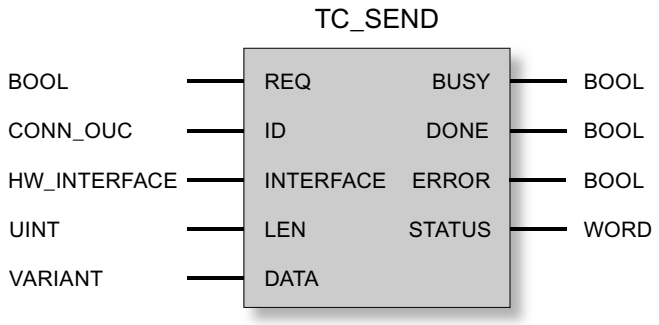

Explanation of the formal parameters

The following table explains all the formal parameters for the TC_SEND instruction.

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
LEN	INPUT	UINT	1...2048	Number of bytes of data to be sent. The value must be ≥ 1 and ≤ 2048. The value should match the size of the range of DATA.
DATA	INOUT	VARIANT		Address reference to the send data area of the CPU *
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. ** For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

* For special features of the DATA parameter for SMS texts, refer to the next section.

** After sending a frame, TC_SEND sets DONE = 1. Note the following response:
The loss of an ISO-on-TCP connection is only recognized by the sender after 1 to 2 minutes. The transferred data may be lost although TC_SEND has set DONE = 1 at the sender.
If an ISO-on-TCP connection is aborted after receiving a frame before TC_RECV was started, the transferred data may be lost even if TC_SEND sets DONE = 1 at the sender.

Configuring SMS texts with the DATA parameter

The instruction sends the data referenced by the pointer of the type VARIANT of the DATA parameter as an SMS text.

If an operand of the data type STRING is referenced by DATA for SMS texts, the first two bytes are transferred with length information of the string.

One option for the correct text representation of SMS messages to be sent is to convert the text string into an Array of BYTE or Array of CHAR using the conversion function Strg_TO_Chars. Strg_TO_Chars at the EN parameter is linked to the output parameter ENO by TC_SEND.

For SMS texts, the CP does not support all special characters, for example umlauts (ü, ä etc.). The specification GSM 03.38 applies. There may be additional restrictions imposed by the GSM network provider.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS *	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	$80E0_H$	Internal error If you send the frames directly to the telecontrol server (mode "Telecontrol"), make sure that the send cycle time is ≥ 1 second.
0	1	$80E1_H$	Timeout: • Increase the value of the "Connection monitoring time" in the configuration of the CP 1242-7 or • Check the connection partner.
0	1	$80E4_H$	Unknown ID: First call TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80E7_H$	Data to be sent not completely transferred: Repeat the job.

DONE	ERROR	STATUS *	Meaning
0	1	$80E8_H$	Remote partner cannot be reached. Check the connection parameters. In the "GPRS direct" mode, the message is output if the partner can be reached but is not accepting a connection request.
0	1	$80E9_H$	Connection establishment by remote partner: Check the connection partner. If necessary, terminate the connection with TC_DISCON and establish it again with TC_CON.
0	1	$80EA_H$	Error message from remote partner: • Check the connection partner. Enable the "TC_RECV" instruction on the communications partner. • If necessary, terminate the connection with TC_DISCON and establish it again with TC_CON.
0	1	$80EF_H$	SMS could not be sent: • Check whether the destination address (telephone number of the destination subscriber) exists. • Check whether the inserted SIM card allows sending of SMS. • Check the length of the SMS text that was sent. SMS texts > 160 characters will not be sent. • Make sure that when the data block TCON_PHONE was created, the "Standard" option was selected for block access.
0	1	$80F1_H$	Sending of SMS messages is not enabled in the STEP 7 configuration of the CP: Enalble the "Allow SMS" option in the configuration of the CP.
0	1	$80F4_H$	Connection endpoint cannot be generated: Check the connection partner.
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed. or • Connection terminated by remote partner: First call TC_DISCON.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value): Check the configuration of the "TC_CON..." SDT.

* Further statuses that are not listed here can be found in the status display is of the "RDREC" or "WRREC" instructions in the two middle status bytes (STATUS[2], STATUS[3]).

TC_RECV: Receive data via the GSM network

Meaning

The TC_RECV instruction allows the reception of data via programmed connections of the following types:

- ISO-ON-TCP connections
- SMS connections
  To receive SMS messages, the phone number of the sender must be configured in the STEP 7 configuration of the receiving CP (authorized phone numbers). The sender must support the CLIP function.
  The phone number of the connection partner must be entered in the "TCON_PHONE" SDT.
  Wake-up SMS messages are filtered out.
- Telecontrol connections

Note

Receiving SMS messages from different senders

If you want to receive SMS messages from different senders, you have two alternatives:

- You configure several connections (TC_CON, TC_RECV, TC_DISCON).
  or
- You may only enter no telephone number for only one configured connection in the required data block "TCON_PHONE" in the "PhoneNumber" parameter. When receiving messages, this is then interpreted as a placeholder for all authorized connection partners.

You will find more detailed information on the connection types in the description of the TC_CON instruction.

The ID parameter references the GPRS connection. The value of ID must correspond to the value used for ID by TC_CON.

The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

The maximum amount of data to be received is specified with the LEN parameter.

The size of the data area specified in DATA must be at least as large as the number of bytes configured for LEN. Permitted data types in the data area specified in DATA are all except BOOL and ARRAY of BOOL. The received data is interpreted as if the sending partner had used the same data types.

The DB (system data type) used for the connection description of TC_RECV must differ from a DB used for TC_SEND.

Storing SMS messages

Received SMS messages are stored retentively in the CP 1242-7 (25 storage spaces) and on the SIM card (varying number of storage spaces).

- After an SMS message has been read by TC_RECV the SMS message will be deleted from its storage space.
- If all of the storage spaces have been allocated and a new SMS message is received, the oldest SMS message is deleted.

Call interface in FBD representation

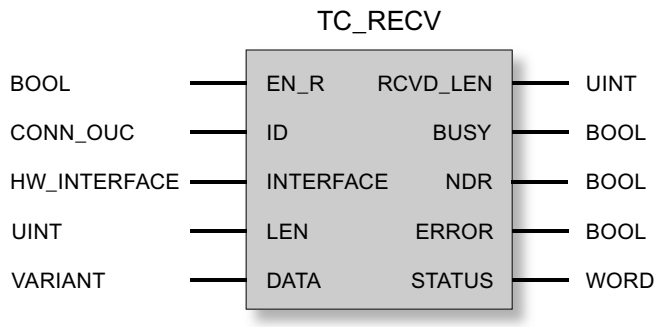

Explanation of the formal parameters

The following table explains all the formal parameters for the TC_RECV instruction

Parameter	Declaration	Data type	Range of values	Description
EN_R	INPUT	BOOL	0: Data reception locked 1: Data reception enabled	Enables / locks the reception of data. • Block version 1.1: After setting 1 to 0, the block is inactive. • Block version 1.0: After setting 1 to 0, the program block receives data again (until DONE = 0 and ERROR = 0). Note the information on the status code 80E3 of TC_CON.
ID	INPUT	CONN_OUC	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTER-FACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
LEN	INPUT	UINT	1...2048	(minimum) number of bytes of data to be received, maximum 2048
DATA	INOUT	VARIANT		Address reference to the receive data area of the CPU *
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
RCVD_LEN	OUTPUT	UINT		Number of bytes of received data

Parameter	Declaration	Data type	Range of values	Description
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

* For special features of the DATA parameter for SMS texts, refer to the next section.

Configuring SMS texts with the DATA parameter

The instruction references the received SMS text with the pointer of the type VARIANT of the DATA parameter to the data area of the CPU.

If DATA references an operand of the data type STRING for the SMS text, the first two bytes of the SMS text will be interpreted as length information of the data type STRING and not as SMS text.

One option for the correct text representation of SMS messages to be received is to convert an Array of BYTE or Array of CHAR to a text string using the conversion function Chars_TO_Strg. Chars_TO_Strg at the EN parameter is linked to the output parameter ENO of TC_RECV.

For SMS texts, the CP does not support all special characters, for example umlauts (ü, ä etc.). The specification GSM 03.38 applies. There may be additional restrictions imposed by the GSM network provider.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS *	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	$80A3_H$	• An attempt is made to re-establish an existing connection. • An attempt is made to terminate a non-existent connection.
0	1	$80E0_H$	Internal error
0	1	8086_H	Illegal value for ID
0	1	$80E4_H$	Unknown ID: First call TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed. or • Connection terminated by remote partner: First call TC_DISCON.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the configuration of the "TC_CON..." SDT.

* Further statuses that are not listed here can be found in the status display is of the "RDREC" or "WRREC" instructions in the two middle status bytes (STATUS[2], STATUS[3]).

TC_CONFIG: Transferring configuration data to a CP

Meaning

With the instruction TC_CONFIG the parameters of an S7-1200 mobile wireless CP configured in STEP 7 can be changed. The configured values are not overwritten retentively. The overwritten values remain valid until TC_CONFIG is called again or until the station starts up again (cold restart after cycling power).

To be able to use the function, there must already be values present in the STEP 7 basic configuration of the CP.

If the STEP 7 configuration data of the CP needs to be changed permanently, the instruction needs to be called again each time the station restarts (cold restart) or a modified project must be downloaded to the station.

The CONFIG parameter points to the memory area with the configuration data. The configuration data is stored in a data block (DB). The structure of the DB is specified by the system data type (SDT) IF_CONF.

The configuration data to be modified on the CP is put together as necessary in blocks in IF_CONF "IF_CONF_..." for the individual parameters.

Parameters that are not intended to change as a result of the instruction are not entered in IF_CONF. They retain the value configured in STEP 7.

For detailed information on settings for IF_CONF, refer to the section IF_CONF: SDT for telecontrol configuration data (Page 6723).

The INTERFACE parameter references the GPRS interface of the required local CP.

Call interface in FBD representation

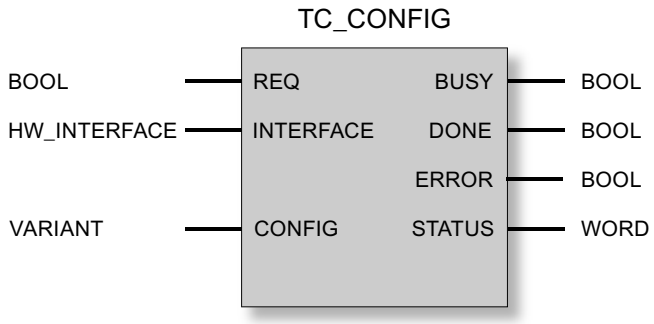

Explanation of the formal parameters

The following table explains all the formal parameters for the TC_CONFIG instruction

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
INTERFACE	INPUT	HW_INTERFACE (WORD)		Reference to the interface of the local CP 1242-7
CONFIG	INOUT	VARIANT	See also "IF_CONF: SDT for telecontrol configuration data	Reference to the memory area with the collected configuration data to be modified
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.

Parameter	Declaration	Data type	Range of values	Description
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	$80E0_H$	Internal error (for example no value in the basic configuration of a parameter)
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80EB_H$	Query temporarily rejected (the CP is currently being configured by STEP 7).
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the "IF_CONF" SDT.
0	1	$80F7_H$	Wrong ID in the parameter fields of the configuration data: Check the "IF_CONF" SDT.

TCON_...: SDTs for the telecontrol connection establishment

System data types TCON_... for telecontrol connections

The four system data types TCON_... described below are used among other things for the connection description of telecontrol connections of the CP 1242-7.

TCON_PHONE is also used with the mobile wireless CPs of the S7-1200 for the description of a connection of Open User Communication to transfer SMS messages (TCON / TDISCON / TSEND or TSEND_C).

To configure a telecontrol connection for the CP 1242-7 using the TC_CON instruction, the CONNECT parameter of the instruction is used for the connection description.

The connection description is specified by the structure of the system data type (SDT). The structure of the relevant SDT contains the parameters necessary to establish the connection with the remote communications partner.

For different connection types that depend on the remote communications partner, the following SDTs are used:

- TCON_IP_RFC for ISO-on-TCP connections to IPv4 stations with CP 1242-7
- TCON_IP_V4 for UDP connections to IPv4 stations (sending only)
- TCON_PHONE for connections to SMS clients
- TCON_WDC for connections to telecontrol servers or stations that can be reached via the telecontrol server.

The parameter assignment of the connection description is made in a data block of the same type as the SDT.

Creating a DB of the type TCON_...

You will need to type in the data types of the relevant DBs with the keyboard. They are not displayed in the selection list. The data types are not case-sensitive.

To create a TCON_... DB, follow the steps outlined below:

1. Create a data block of the type "global DB" with block access "Standard".
2. Create an SDT in the table of the parameter configuration of the DB by assigning the name and typing in the required type (for example "TCON_IP_RFC") in the cell of the data type. The SDT and its parameters are created (see below).
3. Configure the parameters that are described below for each SDT type.

Reserved bits are not displayed.

System data type TCON_IP_RFC for connections to IPv4 stations

This connection type is supported only on ISO-on-TCP connections to communications partners with a fixed IP address. The CP must be configured for the "GPRS direct" mode.

Table 4-396 Parameters of TCON_IP_RFC

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0C	Protocol variant 12 (C_h): ISO-on-TCP connection
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment • 1: Active connection establishment
6 ... 7	-	-	-	- reserved -
8 ... 11	RemoteAddress	IP_V4		IP address of the connection partner
	ADDR	Array [1...4] of Byte		IP address of the relevant connection partner
12 ... 13	RemoteTSelector	TSelector		Remote T selector
	TSelLen	UINT		Length of the remote T selector "RemoteTSelector"
14 ... 45	TSel	Array [1...32] of Byte	any	Remote transport selector of the connection • When "ActiveEstablished" = 1: With active connection establishment, the T selector of the local partner must be the same as the T selector of the connection partner (passive connection establishment on the remote partner). • When "ActiveEstablished" = 0 correspondingly (passive connection establishment local, active connection establishment remote)
46 ... 47	LocalTSelector	TSelector		Local T selector
	TSelLen	UINT		Length of the local T selector "LOCAL_TSel"
48 ... 79	TSel	Array [1...32] of Byte	any	Local transport selector of the connection • When "ActiveEstablished" = 1: With active connection establishment, the T selector of the local partner must be the same as the T selector of the connection partner (passive connection establishment on the remote partner). • When "ActiveEstablished" = 0 correspondingly (passive connection establishment local, active connection establishment remote)

System data type TCON_IP_V4 for connections to IPv4 stations

This connection type is supported only for sending on UDP connections to communications partners with a fixed IP address.

To receive, ActiveEstablished = 0 must be set.

Table 4-397 Parameters of TCON_IP_V4

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0B	Protocol variant 11 (B_h): UDP connection
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment Setting for sending and receiving data. • 1: Active connection establishment Setting for sending data only.
6 ... 9	RemoteAddress	IP_V4		IP address of the connection partner
	ADDR	Array [1...4] of Byte		The four bytes (ADDR[1] ... ADDR[4]) specify the four blocks of the IP address.
10 ... 11	RemotePort	UINT	1...65535	IP port of the connection partner Not relevant if ActiveEstablished = 0.
12 ... 13	LocalPort	UINT	1...65535	Local IP port ("0" is not permitted) Not relevant if ActiveEstablished = 1.

System data type TCON_PHONE for SMS connections

Note

Authorized phone numbers

The CP only accepts an SMS if the sending communication partner is authorized based on its phone number. These numbers are in configured for the CP in STEP 7 in the "authorized phone numbers" list.

SMS text

- Programmed SMS texts for SMS messages to be sent are accessed using the DATA parameter of the TC_SEND instruction.
- The text of a received SMS message is assigned to the address area of the CPU by the DATA parameter of the TC_RECV instruction.

Table 4-398 Parameters of TCON_PHONE

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the local interface (telecontrol interface or Ethernet interface of the CP > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the local mobile wireless connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0E	Protocol variant 14 (E$_h$): SMS connection
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment (not relevant for the CP 1242-7): • 0: Passive connection establishment (not relevant here) • 1: Active connection establishment
6...7	-	-	-	- reserviert -
8 ... 31	PhoneNumber	STRING[22]		Call number of the connection partner Permitted values: Plus character (+) and numbers Note the exact notation of the international dialing code of the relevant phone number assigned by the network provider ("+" character or zeros). Without an entry for the PhoneNumber parameter, no connection partner is specified and SMS messages can be received reception from all authorized connection partners. Note the following during startup: Without an entry, TC_RECV first delivers the oldest received SMS message.

System data type TCON_WDC for connections to telecontrol servers or remote stations

You can configure the connection to the telecontrol server assigned to the S7-1200 or to a remote station that can be reached via the telecontrol server with TCON_WDC. The address data of the telecontrol server assigned to the CP can be found in STEP 7 in the "Telecontrol interface > Mode" tab of the CP. The telecontrol server or a remote station is addressed using the host name or the IP address.

The "RemoteWdcAddress" parameter of TCON_WDC specifies the Access ID of the connection partner.

Table 4-399 Parameters of TCON_WDC

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.

Byte	Parameter	Data type	Initial value	Description
4	ConnectionType	BYTE	W#16#0F	Protocol variant 15 (F_h): Telecontrol connection using an IP address
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment • 1: Active connection establishment
6 ... 7	-	-	-	- reserved -
8 ... 11	RemoteWdcAddress	DWORD		Specifies the Access ID (hex). The access ID depends on the connection partner. • Connection to a remote CP: The access ID is made up of the following: – STEP 7 project number – Station number – Slot If the remote station has more than one GPRS-CP and you do not want to specify the path, the last byte for the slot must be set to 0. You will find the access ID in the STEP 7 project in the "CP authentication of the CP" parameter group. • Connection to the telecontrol server: Access ID = 0 • To only write to the process image of the CP: Access ID = DW#16#FEEDDADA

IF_CONF: SDT for telecontrol configuration data

Structure of the system data type IF_CONF for the TC_CONFIG instruction

The TC_CONFIG parameter of the CONFIG instruction references the memory area containing the CP configuration data to be modified. The configuration data stored in a data block is described as a structure of the IF_CONF system data type (SDT).

To be able to use the function, there must already be values present in the STEP 7 basic configuration of the CP.

IF_CONF is made up of a header followed by fields that correspond to the parameters or parameter areas of the CP in the device properties of the STEP 7 project.

The CP configuration data to be modified is collected together as IF_CONF fields. Parameters that will not be modified are ignored in the IF_CONF structure and remain as they were configured in the STEP 7 project.

Creating the DB and the IF_CONF structures

You can create the parameters of the CP within the IF_CONF DB in one or more structures each with one or more fields.

You will need to type in the data types of the fields using the keyboard. They are not displayed in the selection list. The data types are not case-sensitive.

Follow the steps below to create IF_CONF:

1. Create a data block of the type "global DB" with block access "Standard".
2. Create a structure (data type "Struct") in the table of the parameter configuration of the DB. You can specify any name.
3. Under this structure add a header by assigning the name of the header and typing it in the cell of the data type "IF_CONF_Header".
   The header of the structure and its three parameters (see below) is created.
4. Create a field for the first parameter to be changed by typing in the required data type (for example "IF_CONF_APN") in the cell of the data type.
5. Repeat the last step for all parameters you want to change on the CP using the TC_CONFIG instruction.
6. Finally, update the number of fields in the header in the "subfieldCnt" parameter.

Header of IF_CONF

Table 4-400 IF_CONF_Header

Byte	Parameter	Data type	Initial value	Description
0 ... 1	fieldType	UINT		Field type: Must always be 0.
2 ... 3	fieldId	UINT		Field ID: Must always be 0.
4 ... 5	subfieldCnt	UINT		Total number of fields contained in the structure

General parameters of the parameter fields

Each field has the following general parameters:

- Id
  This parameter identifies the field and must not be modified.
- Length
  This parameter indicates the length of the field. The value serves as information.
  Fields with strings and / or arrays have a variable length. Due to hidden bytes, the actual length of fields can be greater than the sum of the displayed parameters.
- Mode
  The following values are permitted to these parameters:

Table 4-401 Values of "Mode"

Value	Meaning
1	Permanent validity of the configuration data Not relevant for the CP
2	Temporary validity of the configuration data, including deleting of existing permanent configuration data The permanent configuration data is replaced by the parameter fields of IF_CONF.

Field for the parameter area "GPRS access"

Table 4-402 IF_CONF_APN

Parameter	Data type	Initial value	Description
Id	UINT	4	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 174
Mode	UINT		Validity (1: permanent, 2: temporary)
AccesspointGPRS	STRING [98]		APN: Name of the access point of the GSM network provider to the Internet
AccesspointUser	STRING [42]		APN user name
AccesspointPassword	STRING [22]		APN password

Field for the parameter area "CP identification"

Table 4-403 IF_CONF_Login

Parameter	Data type	Initial value	Description
Id	UINT	5	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 54
Mode	UINT		Validity (1: permanent, 2: temporary)
ModemName	STRING [22]		Access ID The value cannot be set.
ModemPassword	STRING [22]		Telecontrol password (max. 20 characters) The password with the CP 1242-7 (6GK7 242-7KX30-0XE0) cannot be changed via the SDT.

Field for the parameter area "Telecontrol server access"

This field is only used when the telecontrol server is addressed with a name that can be resolved by DNS. If the telecontrol server is addressed with its IP address, the "IF_CONF_TCS_IP_V4" field is used.

In STEP 7, the corresponding data is located in the "Mode" parameter area.

If there is more than one telecontrol server, use the field once per server.

Table 4-404 IF_CONF_TCS_Name

Parameter	Data type	Initial value	Description
Id	UINT	6	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 266
Mode	UINT		Validity (1: permanent, 2: temporary)
TcsName	-	-	- reserved -
	STRING [254]		Name of the telecontrol server that can be resolved by DNS

Parameter	Data type	Initial value	Description
RemotePort	UINT		Port of the telecontrol server
Rank	UINT		Priority of the server [1, 2] 1 = first telecontrol server, 2 = second telecontrol server

Field for the parameter area "Telecontrol server access"

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

This field is only used when the telecontrol server is addressed by its IP address. If the telecontrol server is addressed by its DNS name, the "IF_CONF_TCS_Name" field is used.

In STEP 7, the corresponding data is located in the "Mode" parameter area.

If there is more than one telecontrol server, use the field once per server.

Table 4-405 IF_CONF_TCS_IP_v4

Parameter	Data type	Initial value	Description
Id	UINT	7	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
RemoteAddress	IP_V4		IP address of the telecontrol server
RemotePort	UINT		Port of the telecontrol server
Rank	UINT		Priority of the server [1, 2] 1 = first telecontrol server, 2 = second telecontrol server

Field for the "Mode" parameter area

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

In STEP 7, the corresponding data is located in the parameter areas "Mode" and Modem settings".

Table 4-406 IF_CONF_GPRS_Mode

Parameter	Data type	Initial value	Description
Id	UINT	8	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 10
Mode	UINT		Validity (1: permanent, 2: temporary)
GPRSmode	UINT		Mode of the CP: • 0 = Telecontrol • 1 = GPRS direct

Parameter	Data type	Initial value	Description
TemporaryStation	BOOL		Bit 0: Temporary connection If this option is selected, the CP only establishes a temporary connection to send data. Once the messages have been transferred, the CP terminates the connection again. • 1: activated (temporary connection) • 0: deactivated (permanent connection)
SMS_Enabled	BOOL		Bit 1: Allow SMS Selecting the option allows the S7 station to send SMS messages. • 1: activated (SMS allowed) • 0: deactivated (no SMS)

Field for the "SMSC" parameter

In STEP 7, the corresponding data is located in the parameter area "Modem settings".

Table 4-407 IF_CONF_SMS_Provider

Parameter	Data type	Initial value	Description
Id	UINT	10	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 28
Mode	UINT		Validity (1: permanent, 2: temporary)
SMSProvider	STRING [20]		Node number of the SMS center (SMSC) of the GSM network provider with which the contract was signed for this station.

Field for the "PIN" parameter

In STEP 7, the corresponding data is located in the parameter area "Modem settings".

Table 4-408 IF_CONF_PIN

Parameter	Data type	Initial value	Description
Id	UINT	11	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 16
Mode	UINT		Validity (1: permanent, 2: temporary)
Pin	STRING [8]		PIN of the SIM card inserted in the SIM card The parameter is not relevant if the PIN was correctly configured. If the PIN was incorrectly configured, the correct PIN can be entered.

Field for monitoring times

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

In STEP 7, the corresponding data is located in the parameter areas "Keepalive timeout" and Operating mode".

Table 4-409 IF_CONF_TC_Timeouts

Parameter	Data type	Initial value	Description
Id	UINT	12	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 12
Mode	UINT		Validity (1: permanent, 2: temporary)
KeepAliveTimeout	-	-	- Reserved - (cannot be set)
SendTimeout	UINT		Connection monitoring time: Monitoring time of the connection to the communications partner (seconds) Relevant in the modes "Telecontrol" and "GPRS direct"
RedialTimeout	UINT		Reconnection delay: Basic value for the wait time until the next attempt to establish a connection following an unsuccessful connection establishment. After every 3 attempts, the basic value is doubled up to a maximum of 900 s. Range of values: 10 to 600 s. Example: Basic value 20 results in the following dialing intervals: three times 20 s, three times 40 s, three times 80 s etc. up to a maximum of 900 s. If a second telecontrol server / router is configured, the CP attempts to connect to the second partner at the 4th attempt. If the second partner is also unreachable, the 7th time the CP attempts to connect to the first partner again. Not relevant for SMS connections

Field for the "Wake up right" parameter area

Table 4-410 IF_CONF_WakeupList

Parameter	Data type	Initial value	Description
Id	UINT	13	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 246
Mode	UINT		Validity (1: permanent, 2: temporary)
WakeupPhone [1...10]	ARRAY [1...10] of STRING [22]		Phone number subscriber authorized to wake up The asterisk (*) at the end of a call number is used a placeholder for direct dialing numbers.

Field for the "Preferred GSM networks" parameter area

Table 4-411 IF_CONF_PrefProvider

Parameter	Data type	Initial value	Description
Id	UINT	14	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 46

Parameter	Data type	Initial value	Description
Mode	UINT		Validity (1: permanent, 2: temporary)
Provider [1...5]	ARRAY [1...5] of STRING [6]		Alternative GSM networks with priority 1 to 5 into which the CP dials. Up to 5 networks can be configured. No. 1 with highest priority, no. 5 with lowest priority. Entry of the Public Land Mobile Network (PLMN) of the network provider consisting of Mobile Country Code (MCC) and Mobile Network Code (MNC). Example (test network of Siemens AG): 26276

Field for the "DNS configuration" parameter area

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

Table 4-412 IF_CONF_DNS

Parameter	Data type	Initial value	Description
Id	UINT	16	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
DNS_IP [1]	IP_V4		IP address of the 1st Domain Name System server
DNS_IP [2]	IP_V4		IP address of the 2nd Domain Name System server

Field for the "Time-of-day synchronization" parameter area

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

Table 4-413 IF_CONF_NTP

Parameter	Data type	Initial value	Description
Id	UINT	17	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 24
Mode	UINT		Validity (1: permanent, 2: temporary)
NTP_IP [1]	ARRAY [1...4] of IP_V4		IP address of NTP server 1
...	...		(IP address of NTP server 2...3)
NTP_IP [4]	ARRAY [1...4] of IP_V4		IP address of NTP server 4

Block for activating / deactivating TeleService users

The SDT can only be used for configuration at runtime with the CP 1242-7 (6GK7 242-7KX30-0XE0).

SDT for activating or deactivating TeleService users already configured in the STEP 7 project of the CP. In STEP 7, the corresponding data can be found in the parameter area "TeleService settings" > "TeleService user management".

Table 4-414 IF_CONF_GPRS_UserList

Parameter	Data type	Initial value	Description
Id	UINT	19	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 506
Mode	UINT		Validity (1: permanent, 2: temporary)
GPRS_User [1...10]	ARRAY [1...10] of GPRS_User		TeleService user no. 1 to max. no. 10

The array is formed from the parameter records for the TeleService users ("GPRS_User" [1...n]).

Table 4-415 GPRS_User [n] (parameter for TeleService user)

Parameter	Data type	Initial value	Description
UserName [n]	STRING [22]		TeleService user name
Password [n]	STRING [22]		- The string must be empty!
Diag_Allowed [n]	BOOL		- Reserved - (cannot be set)
Teleserv_Allowed [n]	BOOL		Activation of the TeleService user • 0 = user is deactivated • 1 = user is activated
FW_Load_Allowed [n]	BOOL		- Reserved - (cannot be set)

Field for setting the parameters for TeleService access (DNS name of the server)

Access data of the TeleService server (switching station).

In STEP 7, the corresponding data is located in the parameter area "TeleService settings".

If there is more than one TeleService server, use the field once per server.

Table 4-416 IF_CONF_TS_Name

Parameter	Data type	Initial value	Description
Id	UINT	20	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 266
Mode	UINT		Validity (1: permanent, 2: temporary)
ts_name	String [254]		Name of the TeleService server that can be resolved by DNS
RemotePort	UINT		Port of the engineering station
Rank	UINT		Priority of the server [1] or [2] 1 = server 1, 2 = server 2

Field for setting the parameters for TeleService access (IP address of the server)

Access data of the TeleService server (switching station).

In STEP 7, the corresponding data is located in the parameter area "TeleService settings".

If there is more than one TeleService server, use the field once per server.

Table 4-417 IF_CONF_TS_IF_V4

Parameter	Data type	Initial value	Description
Id	UINT	21	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
RemoteAddress	IP_V4		IP address of the TeleService server
RemotePort	UINT		Port of the TeleService server
Rank	UINT		Priority of the server [1] or [2] 1 = server 1, 2 = server 2

S7-300/400-CP

CP 443-1 OPC UA

Program blocks for the OPC UA client

Overview of the program blocks for the OPC UA client function

The function blocks listed below are available in the library "SIMATIC NET CP" for the OPC UA client function of the CP. Use the blocks on the CPU.

Some blocks require special user-defined PLC data types (UDTs) that you will find in the following below the relevant program block. The UDTs are available as pre-assembled data types. To do this under the entry "PLC data types" of the station create a new data type and in the "data type" box, type in the name of the UDT. for example "UA_SessionConnectInfo". The required parameters will then be created automatically in the declaration table of the UDT.

- UA_Connect
  Establishes a connection for a session with an OPC UA server.
  - UA_SessionConnectInfo
    Contains connection information for the block parameter "SessionConnectInfo".
  - UA_UserIdentityToken
    Contains the data of the user authentication for the block parameter "UserIdentityToken".
- UA_NamespaceGetIndex
  Fetches the name space index of a name space URI.

- UA_NodeGetHandleList
  Registers node IDs on the connected server and fetches the node handles in the form of a list.
  - UA_NodeID
    Contains the parameters for identifying the target node for the block parameter "NodeID".
- UA_NodeReleaseHandleList
  Releases the node handles of the used list on the server.
- UA_Disconnect
  Terminates the connection of a current session with an OPC UA server.
- UA_ReadList
  Using the list of node handles, reads the data from nodes of the connected server.
  - UA_NodeAdditionalInfo
    Specifies the item attribute and the index range for the block parameter "NodeAddInfos".
  - UA_IndexRange
    Specifies the start and end index for the block parameter "IndexRange”.
  - UA_AnyPointer
    References the memory area of the CPU in which the process data for the block parameter "Variables" is stored.
  - UA_TimeStamps
    Contains the time stamps of the elements of the data area of the block parameter "Variables".
- UA_WriteList
  Using the list of node handles, writes data to nodes of the connected server.
  - UA_NodeAdditionalInfo
    Specifies the item attribute and the index range for the block parameter "NodeAddInfos".
  - UA_IndexRange
    Specifies the start and end index for the block parameter "IndexRange”.
  - UA_AnyPointer
    References the memory area of the CPU in which the process data for the block parameter "Variables" is stored.

Additionally required system functions (SFC)

The following system functions are required for the full functionality of the program blocks for the OPC UA client:

- BLKMOV (SFC20)
- TIME_TCK (SFC64)

For its function, UA_Connect also requires:

- LGC_GADR (SFC49)
- RDSYSST (SFC51)

Creating SFCs

Create the system functions by opening an organization block in the block directory of the CPU, for example OB1.

In the block catalog that is displayed by opening OB1, expand the corresponding block group.

Find the SFCs in the following block order:

- BLKMOV (SFC20): "Simple instructions" group > "Move" folder
- TIME_TCK (SFC64): "Extended instructions" group > "Day and time" folder
- LGC_GADR (SFC 49): "Extended instructions" > "Addressing" folder
- RDSYSST (SFC 51): "Extended instructions" group > "Diagnostics" folder

Drag the SFCs to the network of the organization block and supply the parameters of the SFC.

Calling the program blocks and their interaction

Note

General notes on the program blocks and instance DBs

Create the function blocks with "Default access method". In the assigned instance data blocks leave the option "Retentivity" disabled.

Each program block must be called cyclically until the status parameter "Done" or "Error" changes to 1.

Within a session only a single block can be called at any one time. Different blocks can only be processed one after another in a session.

With UA_Connect you establish a connection from the CP as OPC UA client to a UA server and open a session. The block must be called for every connection to a UA server. When necessary for more than one connection to a UA server, the block can be called more than once with different instance data blocks (IDBs). You can use this, for example, to read via one connection and to write via another connection or to read or write different data via multiple connections.

For connections to more than one UA server, the UA_Connect block must be called several times with different input parameters. As a UA client, the CP can establish connections to up to five UA servers.

To query the name space index of a name space URI of the UA server, call a UA_NamespaceGetIndex within a session. To query multiple name space URIs you can call the block more than once with different input parameters.

To create a handle list as preparation for the read and write services call up UA_NodeGetHandleList. The block is called separately with suitable information at the "NodeIDs" input parameter for each target node.

Following this you can use the blocks UA_ReadList und UA_WriteList to read and write the data of the items. You can call these two blocks as often as necessary within a session.

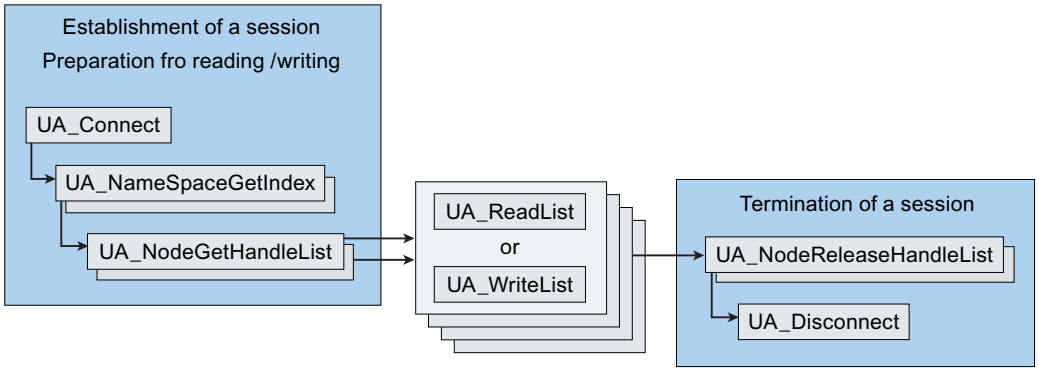

Figure 4-43 Calling the client program blocks and their interaction

If the connection is no longer required, release the handles on the server again using the block UA_NodeReleaseHandleList. This deletes the handles on the server.

The connection is terminated with UA_Disconnect and the session is ended.

See also

Time monitoring of the blocks (Page 6734)

Time monitoring of the blocks

Parameters for the time response of the blocks

To control and monitor the time response of the program blocks, the the following three parameters are available:

- Timeout
  Input parameter in all blocks for the OPC UA client
- SessionTimeout
  Input parameter in the block UA_Connect
- MonitorConnection
  Input parameter in the block UA_Connect

Since the parameters influence the running of the OPC UA communication as a whole, below notes on these parameters precede the block description.

Timeout

With this input parameter you monitor every block call. If a block call cannot be completed within the configured maximum permissible time, the status parameter "Error" is set to 1 and processing aborted.

SessionTimeout

With this input parameter you monitor the duration of a session without data traffic without the block "UA_Disconnect" being called to terminate a connection.

The minimum value of SessionTimeout" is 30 seconds. If the value is exceeded, the connection is terminated.

The parameter allows a server to reduce bound resources if the client does not use the session for a longer period of time.

An adequate time, however, ensures that the session is not terminated immediately if, for example, data cannot be transferred immediately due to network disruptions.

The value of "SessionTimeout" should generally be higher than the values of "Timeout" for the "UA_Connect" blocks used.

MonitorConnection

With this input parameter the status of a connection without data traffic is checked. When this connection monitoring time elapses, a frame is sent to check the status of the connection to the server.

To ensure that the session is not terminated immediately using the "SessionTimeout" parameter when there is no data traffic and the connection monitoring time is addressed, it is recommended that you select a value for "MonitorConnection" that is lower than half of "SessionTimeout".

See also

Program blocks for the OPC UA client (Page 6731)

UA_Connect

Function

Function of the block

With the UA_Connect block you establish a connection from the CP as OPC UA client to a UA server and open a session.

As the target address, you specify the URL of the UA server at the "ServerEndpointUrl” parameter.

You store the connection information at the "SessionConnectInfo” parameter in a data block.

UA_Connect also requires the system functions LGC_GADR (SFC 49) and RDSYSST (SFC 51).

See also

Parameter (Page 6736)

Program blocks for the OPC UA client (Page 6731)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-418 Parameters of the block UA_Connect

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ServerEndpointUrl	IN	STRING	Max. 254 characters	Address (URL) of the connection partner (server). Only IPv4 addresses are permitted.
SessionConnectInfo	IN	UDT	See "Meaning"	Connection information, see UASessionConnectInfo (Page 6749).
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
ConnectionHdl	OUT	DWORD	1 .. 5	Unique identifier of an established connection It is required by other blocks as an input parameter.
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 6755). Note: If the error code B000F002 is output, the OPC UA session is not yet established. Call the block again.

UA_NamespaceGetIndex

Function

Function of the block

With the UA_NamespaceGetIndex block you fetch the name space index of the name space URI of the connected UA server.

See also

Parameter (Page 6737)

Program blocks for the OPC UA client (Page 6731)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-419 Parameters of the block UA_NamespaceGetIndex

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block. As long as Execute = 1, the other input parameters cannot be changed.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
NamespaceUri	IN	STRING	Max. 254 characters	Specifies the address (name space URI) of the target server Example of the name space of the CPU symbols in the S7-400 with CP 443-1 OPC UA: http://www.siemens.com/simatic-classic-s7-opcua
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
NamespaceIndex	OUT	WORD	0 .. 65535	Output of the name space index of the name space specified by the "NamespaceUri" parameter. The namespace indexes 0 and 1 are reserved and specified by the OPC Foundation.

Parameter	Declaration	S7 data type	Range of values	Meaning
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 6755).

UA_NodeGetHandleList

Function

Function of the block

With the UA_NodeGetHandleList block, you register nodeIDs on the connected UA server.

The UA server returns the handles. These are output by the block with the "NodeHdls" parameter. You access the items of the node handles with the blocks "UA_ReadList" and "UA_WriteList" at their input parameter "NodeHdls".

The nodeIDs of the target server that you specify in the "NodeIDs" input parameter must be known to you. You store these in a UDT "UANodeIDs".

The number of target nodes at the input parameter "NodeIDCount" is always 1 so you need to call the block separately for each nodeID

See also

Parameter (Page 6739)

Program blocks for the OPC UA client (Page 6731)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-420 Parameters of the block UA_NodeGetHandleList

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
NodeIDCount	IN	WORD	1	Number of elements in the data area von "NodeIDs". The number of must be 1.
NodeIDs	IN	ARRAY of UDT	See "Meaning"	Structure of the parameters of the target node of the OPC server. The number of nodeIDs (1) must be identical to the number at the output parameter "NodeHdls". For information on the structure of the UDT, see UANodeID (Page 6751).
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
NodeHdls	OUT	ARRAY of DWORD	0 .. 4 294 967 295	Array of the handles that the OPC UA server assigns as the reply to the client's query. The handles serve to uniquely identify the item on the server. The handles remain valid until they are released by calling the UA_NodeReleaseHandleList block, in other words become invalid. When a session is terminated, the CP cancels all registered node handles of this session. The parameter is used for the input parameter "NodeHdls" of the "UA_ReadList" and "UA_WriteList" blocks.
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".

Parameter	Declaration	S7 data type	Range of values	Meaning
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 6755).
NodeErrorIDs	OUT	ARRAY of DWORD	See "Meaning"	Contains the error numbers for all elements of the data area of "NodeIDs". For the significance of the numbers, refer to the section Error numbers (Page 6755).

UA_NodeReleaseHandleList

Function

Function of the block

With the UA_NodeReleaseHandleList block you release the node handles of the current session on the connected UA server. This deletes the list.

The input parameter "NodeHdls" references the data block to which the handles of the output parameter "NodeHdls" of the "UA_NodeGetHandleList" block are written.

The number of handles at the input parameter "NodeHdlCount" is always 1 so you need to call the block separately for each handle.

See also

Parameter (Page 6740)

Program blocks for the OPC UA client (Page 6731)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-421 Parameters of the block UA_NodeReleaseHandleList

Parameter	Declaration	S7 data type	Range of values	Meaning / remarks
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
NodeHdlCount	IN	WORD	1	Number of elements in the data area von "NodeHdls". The number of must be 1.
NodeHdls	IN	ARRAY of DWORD	0 .. 4 294 967 295	Array of the IDs of the handles to be canceled. If execution of the block is successful, the handles are canceled on the server and therefore become invalid.

Parameter	Declaration	S7 data type	Range of values	Meaning / remarks
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 6755).
NodeErrorIDs	OUT	ARRAY of DWORD	See "Meaning"	Contains error codes for all elements of the data area of "NodeHdls".

UA_Disconnect

Function

Function of the block

With the UA_Disconnect block you terminate an existing connection between the CP as OPC UA client and a UA server. This ends the existing session.

See also

Program blocks for the OPC UA client (Page 6731)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-422 Parameters of the block UA_Disconnect

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 6755).

UA_ReadList

Function

Function of the block

With the UA_ReadList block you read data from the connected UA server.

You program the attributes to be read in a UDT "UANodeAdditionalInfo" that is referenced by the input parameter "NodeAddInfos".

The handles required at the input parameter "NodeHdls" are read from array of the output parameter "NodeHdls" of the "UA_NodeGetHandleList" block.

The number of nodeIDs at the input parameter "NodeIDCount" is always 1 so you need to call the block separately for each target node.

See also

Parameter (Page 6743)

Program blocks for the OPC UA client (Page 6731)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-423 Parameters of the block UA_ReadList

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
NodeHdlCount	IN	WORD	1	Number of elements in the target area to be read. The number of must be 1.
NodeHdls	IN	ARRAY of DWORD	1	Array of the handles of the target nodes (items) of the OPC UA server. The handles are read from array of the output parameter "NodeHdls" of the "UA_NodeGetHandleList" block.
NodeAddInfos	IN	ARRAY of UDT	See "Meaning"	Specifies the attribute of the items to be read and the index range. For information on the structure of the UDT, see UANodeAdditionalInfo (Page 6752).
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
Variables	IN_OUT	UDT	See "Meaning"	Reference to the data to be read. For information on the structure of the UDT, see UAAnyPointer (Page 6753). For the supported data types of the process data, refer to the section Data types (Page 6746).
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing

Parameter	Declaration	S7 data type	Range of values	Meaning
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "ErrorID" parameter.
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 6755).
NodeErrorIDs	OUT	ARRAY of DWORD	See "Meaning"	Contains error numbers for all elements of the data area of "Variables". For the significance of the numbers, refer to the section Error numbers (Page 6755).
TimeStamps	OUT	UDT	See "Meaning"	Contains time stamps for all elements of the data area of "Variables". For information on the structure of the UDT, see UATimeStamp (Page 6753).

UA_WriteList

Function

Function of the block

With the UA_WriteList block you write data to the connected UA server.

You program the attributes to be written in a UDT "UANodeAdditionalInfo" that is referenced by the input parameter "NodeAddInfos".

The handles required at the input parameter "NodeHdls" are read from array of the output parameter "NodeHdls" of the "UA_NodeGetHandleList" block.

The number of nodeIDs at the input parameter "NodeIDCount" is always 1 so you need to call the block separately for each target node.

See also

Parameter (Page 6745)

Program blocks for the OPC UA client (Page 6731)

Parameter

Block parameters

The following table explains the formal parameters of the program block.

Table 4-424 Parameters of the block UA_WriteList

Parameter	Declaration	S7 data type	Range of values	Meaning
Execute	IN	BOOL	0, 1	A rising edge 0 → 1 at the parameter starts processing of the block.
ConnectionHdl	IN	DWORD	1 .. 5	Connection identifier supplied by UA_Connect
NodeHdlCount	IN	WORD	1	Number of elements in the target area to be written to. The number of must be 1.
NodeHdls	IN	ARRAY of DWORD	1	Array of the handles of the target nodes (items) of the OPC UA server. The handles are read from array of the output parameter "NodeHdls" of the "UA_NodeGetHandleList" block.
NodeAddInfos	IN	ARRAY of UDT	See "Meaning"	Specifies the attribute of the items to be written and the index range. For information on the structure of the UDT, see UANodeAdditionalInfo (Page 6752).
Timeout	IN	TIME	5000 .. 120000	Maximum time for establishing the connection in milliseconds. If the value is exceeded, processing of the block is aborted with error number B0007001. If a value is entered outside the permitted range, the default value of 60000 (60 seconds) is used.
Variables	IN_OUT	UDT	See "Meaning"	Reference to the data to be written. For information on the structure of the UDT, see UAAnyPointer (Page 6753). For the supported data types of the process data, refer to the section Data types (Page 6746).
Done	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block execution aborted, not yet completed or not yet started • 1: Block processing completed without errors
Busy	OUT	BOOL	0, 1	Status parameter for block processing • 0: Block not being processed • 1: Block currently executing
Error	OUT	BOOL	0, 1	Error code • 0: No error • 1: An error has occurred. See "" parameter "ErrorID".

Parameter	Declaration	S7 data type	Range of values	Meaning
ErrorID	OUT	DWORD	See "Meaning"	Output of the error number when "Error" = 1. For the significance of the numbers, refer to the section Error numbers (Page 6755).
NodeErrorIDs	OUT	ARRAY of DWORD	See "Meaning"	Contains error numbers for all elements of the data area of "Variables". For the significance of the numbers, refer to the section Error numbers (Page 6755).

Data types

Data types and memory areas

The CP 443-1 OPC UA supports the data types listed below for process data.

The corresponding symbols on the CPU are permitted for the operand areas listed. The memory areas have the following abbreviations. The short forms in brackets are abbreviations depending on the mnemonics set (German/English).

- I = Process input image (E/I)
- Q = Process output image (A/Q)
- PI = I/O area inputs (PE/PI)
- PQ = I/O area outputs (PA/PQ)
- M = Memory bit
- DB = Data block / system data block / instance data block

I/O areas are only supported by the OPC UA server.

Table 4-425 Data types

Size [bit], type	Area	Data type			Remark	Operand area of the CPU
		SIMATIC S7	IEC 61131-3	OPC UA		
1	0 .. 255	BOOL	BOOL	Boolean	0 = false 1 .. 255 = true	I, Q, PI, PQ, M, DB
Numbers						
8, UInt8	0 .. 255	BYTE	BYTE, USINT	Byte		I, Q, PI, PQ, M, DB
8, UInt8	0 .. 255	CHAR	CHAR	Byte	ASCII characters 32 .. 255	I, Q, PI, PQ, M, DB
16, UInt16	0 .. 65535	WORD	WORD, UINT	UInt16		I, Q, PI, PQ, M, DB
16, Int16	-32768 .. 32767	INT	INT	Int16		I, Q, PI, PQ, M, DB
32, UInt32	0 .. 4294967295	DWORD	DWORD, UDINT	UInt32		I, Q, PI, PQ, M, DB
32, Int32	2147483648 .. 2147483647	DINT	DINT	Int32		I, Q, PI, PQ, M, DB

Size [bit], type	Area	Data type			Remark	Operand area of the CPU
		SIMATIC S7	IEC 61131-3	OPC UA		
32, Float	±1.175 495e-38 .. 3.402823e+38	REAL	REAL	Float		DB
Time and date						
16, UInt16	0 .. 999	S5TIME	TIME	UInt16	SIMATIC time For the coding see online help of STEP 7.	DB
16, UInt16	D#1990-1-1 .. D#2168-12-31 (0 .. 65378)	DATE	DATE	UInt16	IEC date [d]	DB
32, UInt32	TOD#0:0:0.0 .. TOD#23:59:59.999 (0 .. 86399999)	TIME_OF_DAY (TOD)	TIME_OF_DAY	UInt32	Time of day [ms]	DB
32, UInt32	-24D_20H_31M_23S_648MS .. T#24D_20H_31M_23S_647MS (-2147483648 .. 2147483647)	TIME	TIME	UInt32	IEC time [ms]	DB
Complex data types						
64	1990-1-1-0:0:0.0 .. 2089-12-31-23:59:59.999	DATE_AND_TIME	DT DATE_AND_TIME	UInt32+ UInt32	Date and time of day For the coding see online help of STEP 7.	DB
254 * 8		STRING *	STRING	String	Max. 254 bytes user data	DB
		ARRAY *			Multidimensional field of elementary or complex data types	DB
		UDT *			User-defined data type for specific parameter groups	DB

* For the coding see below.

You will find a general description of the coding of the data types in the STEP 7 online help.

Special features of complex data types

With the complex data types used by the OPC UA function of the CP. the special coding rules apply that are described below.

STRING

A STRING can contain a maximum of 254 characters.

ASCII characters from the range 32 (0x20) .. 255 (0xFF) are permitted.

Of these, the following characters are not permitted: 127 (0x7F), 129 (0x81), 141 (0x8D), 143 (0x8F), 144 (0x90), 157 (0x9D)

Special features of the OPC UA server

The characters are coded in the format UTF-8 in the OPC UA server of the CP.

Special features of the OPC UA client

With process data with the data type STRING that are referenced by the "UAAnyPointer", the data type "2" (BYTE) is specified.

As the repetition factor in the UAAnyPointer always select the maximum value of 256 (0x100); 254 for the user data + 2 for the header.

ARRAY

Special features of the OPC UA client

The client program blocks support only the transfer of one dimensional arrays from the memory area "DB".

The transfer of the data type "ARRAY of BOOL" is not supported by the client program blocks.

With a ARRAY of STRING, with the limits of the dimension of every string element the maximum length (256) must be specified "n" times, where "n" is the number of strings in the array.

Special features of the OPC UA server

The data type "ARRAY of BOOL" is readable in all cases.

The data type "ARRAY of BOOL" can only be written when the length is a multiple of 8.

UDT

You will find the structure of the UDTs required by the program blocks for the OPC UA client function in the description of the relevant block, see for example AUTOHOTSPOT.

Parameter types

The parameter types "TIMER" and "COUNTER" are only supported for the OPC UA server of the CP.

UDTs for the CP 443-1 OPC UA

UASessionConnectInfo

UASessionConnectInfo

The following table shows the meaning of the connection information for the parameter "SessionConnectInfo" of the UA_Connect block.

Table 4-426 Parameters of UDT UASessionConnectInfo

Parameter	S7 data type	Range of values	Meaning
SessionName	STRING	0 .. 254 characters	Name of the session If the string remains empty, the following is entered by the system as the session name: <Connection>+<decimal value of "ConnectionHdl">
ApplicationName	STRING	0 .. 254 characters	The parameter is given the application name of the CP (OPC UA client) configured In HW Config by STEP 7.
SecurityMsgMode	WORD	0 .. 3	Security process • 0 = Best possible procedure • 1 = No security process • 2 = Authenticate • 3 = Authenticate and encrypt
SecurityPolicy	WORD	0 .. 4	Securityprofile • 0 = Best possible security profile • 1 = No security profile • 2 = Basic128Rsa15 • 3 = Basic256 • 4 = Basic256Sha256
CertificateStore	STRING	0 .. 254 characters	Certificate store of the CP (OPC UA client) The parameter is inserted by STEP 7.
ClientCertificateName	STRING	0 .. 254 characters	Name of the client certificate The parameter is inserted by STEP 7.
ServerUri	STRING	0 .. 254 characters	Server address (URI) that is also stored in the server certificate. The parameter my only be left empty for sessions without a security process. Example: urn:<ApplicationName>:GUID

Parameter	S7 data type	Range of values	Meaning
CheckServerCertificate	BOOL	0, 1	Comparison (check) of the certificate sent by the server with the server certificate stored in the certificate store of the client CP: • 0 = No check • 1 = Check **Note** The settings of this parameter are overwritten by the security settings in the configuration. If you enable the check (1) but the security functions are disabled in the configuration of the CP, connection establishment is aborted with an error message. Recommendation: Leave the parameter at the default setting (0) since the security functions are enabled by the configuration of the CP.
TransportProfile	WORD	1	1 = UATP_UATcp According to the PLCopen specification, only this transport profile is supported.
UserIdentityToken	UDT	See "Meaning"	Data of the user authentication, see UAUserIdentityToken (Page 6751).
VendorSpecificParameter	WORD	See "Meaning"	Entry of the logical address of the CP 443-1 OPC UA. You will find this in STEP 7 properties dialog of the CP as an input address in the "Addresses" tab.
SessionTimeout	TIME	30000 .. 86 400 000	Maximum time that a session remains established when there is no data traffic (milliseconds). If the value is exceeded, the session (connection) is aborted. In this case, you will need to re-establish the connection by calling UA_Connect. If you program values outside the permitted range of values (30 seconds ... 1 day), the value 1200000 (20 minutes) is used.
MonitorConnection	TIME	5000 .. 86 400 000	Connection monitoring time (milliseconds) Time without data traffic after which the client checks the connection to the server by sending a frame. Default setting: 15000 (15 seconds)
LocaleIDs	ARRAY[1..5] of String[6]	Examples: en-US, de-DE, zh-CHS ...	Optional language and regional identifier acc. to RFC 3066. 0 = no or unknown LocaleID.

UAUserIdentityToken

UAUserIdentityToken

The following table shows the meaning of the user authentication data that supplies the UserIdentityToken parameter of the UDT "UASessionConnectInfo".

Table 4-427 Parameters of UDT UAUserIdentityToken

Parameter	S7 data type	Range of values	Meaning
UserIdentity TokenType	WORD	0, 1	0 = No authentication 1 = Authentication When the communications partner (server) demands authentication with a user name and password, set this parameter to 1 and the two following according to the requirements of the server.
TokenParam1	STRING	1 .. 254 characters	User name
TokenParam2	STRING	1 .. 254 characters	Password

UANodeID

UANodeID

The following table shows the meaning of the parameters of UANodeID to identify the target node on the OPC UA server. UANodeID supplies the parameter "NodeIDs" of the UA_NodeGetHandleList" block.

Table 4-428 Parameters of UDT UANodeID

Parameter	S7 data type	Range of values	Meaning
NamespaceIndex	WORD	0 .. 65535	Index of the name space of the server
Identifier	STRING	Max. 254	Specifies the nodeID in the name space index.
IdentifierType	WORD	1 .. 2	Specifies the format and the area of application (generally the server) of the node ID. Supported types: • 1: UAIdentifierType_String String identifiers distinguish upper and lower case. • 2: UAIdentifierType_Numeric Numeric identifier

UANodeAdditionalInfo

UANodeAdditionalInfo

Specifies the attribute of the item and the index range for the block parameter "NodeAddInfos".

Table 4-429 Parameters of UDT UANodeAdditionalInfo

Parameter	S7 data type	Range of values	Meaning
AttributeID	WORD	13	Attribute of the items Only attribute 13 (UAAI_Value) for the value of the item is supported.
IndexRangeCount	WORD	0 .. 1	Number of index ranges The value is unimportant with elementary data types. For items data type ARRAY the following applies: • 0: A single index The entire array is read/written • 1: A subsection of an array defined by "IndexRange".
IndexRange	UDT	See “Meaning”	Area of the array for the parameter IndexRangeCount. For information on the structure of the UDT, see UAIndexRange (Page 6752).

UAIndexRange

UAIndexRange

Specifies the start and end index for the parameter "IndexRange” of the UDT "UANodeAdditionalInfo".

Table 4-430 Parameters of UDT UAIndexRange

Parameter	Data type	Range of values	Meaning
StartIndex	WORD	0 .. 65535	Index as of which reading will take place.
EndIndex	WORD	0 .. 65535	Index up to which reading will take place.

Rules:

If IndexRangeCount = 1 is used, the following rules apply to programming the index range:

- StartinIndex and EndIndex must be assigned.
- For access to more than one element the StartIndex must be lower than the EndIndex. If this is not the case, this causes an error with the number 80360000.

- If you want to read a single element of an array, enter the same value for the StartIndex and EndIndex.
- To be able to access all elements of the array, the StartIndex and EndIndex must be assigned based on the total number of elements of the array.
  If values are specified outside the size of the array, this causes an error with the number 80370000.

Example of assigning "StartIndex" and EndIndex" with arrays.

From an array with a size of 10 elements (no. 1 .. 10) the elements 3 to 5 need to be read. The two indexes are programmed as follows:

- StartIndex = 2 (element no. 3)
- EndIndex = 4 (element no. 5)

UATimeStamp

UATimeStamp

The UDT contains the time stamps of the elements of the data area in milliseconds. UATimeStamp supplies the parameter "TimeStamps" of the block "UA_ReadList".

Table 4-431 Parameters of UDT UATimeStamp

Parameter	S7 data type	Range of values	Meaning
TimestampL	WORD	see below	Least significant word (bits 0..15)
TimestampH	WORD	see below	Most significant word (bits 16..31)

Range of values of the entire double word:

TOD#0:0:0.0 .. TOD#23:59:59.999 (0 .. 86 399 999 milliseconds)

UAAnyPointer

UAAnyPointer

The UDT "UAAnyPointer" is referenced by the "Variables" parameter of the program block.

The UDT references the memory area of the process data in the CPU. To do this process data in the operand areas "inputs" or "outputs" must be copied to a data block.

Store the UDT in a data block.

Table 4-432 Parameters of UDT UAAnyPointer

Parameter	S7 data type	Range of values	Meaning
SyntaxID	WORD	10	The value for the syntax ID is always 10.
DataType	WORD	See "Meaning"	Data type of the target node For the range of values see below, table "Coding of the data type".

Parameter	S7 data type	Range of values	Meaning
RepetitionFactor	WORD	0 .. 65535	Repetition factor For further information, refer to the STEP 7 online help.
DB_Number	WORD	0 .. 65535	Number of the data block (DB) Enter either the number of a DB or a memory area. If you specify a DB, enter a zero in "memArea".
MemArea	WORD	See "Meaning"	Memory area Enter either the number of a DB or a memory area. If you specify a memory area, enter a zero in "DB_Number". For the range of values of the memory area see below, table "Coding of the memory area".
ByteOffset	WORD	Depends on the memory area, see "Meaning"	Byte offset in the specified memory area, as of which the data is accessed.
BitOffset	WORD	0 .. 7	Bit offset in the specified memory area

Data type

The following table explains the coding of the data type in the "DataType" parameter of the UDT UAAnyPointer.

Table 4-433 Coding of the data type

Hexadecimal code	S7 data type	Description
b#16#01	BOOL	Bit
b#16#02	BYTE	Byte (8 bits)
b#16#03	CHAR	Character (8 bits)
b#16#04	WORD	Word (16 bits)
b#16#05	INT	Integer (16 bits)
b#16#06	DWORD	Word (32 bits)
b#16#07	DINT	Integer (32 bits)
b#16#08	REAL	Floating-point number (32 bits)
b#16#09	DATE	Date
b#16#0A	TIME_OF_DAY (TOD)	Time of day
b#16#0B	TIME	Time
b#16#0C	S5TIME	Data type S5TIME
b#16#0E	DATE_AND_TIME (DT)	Date and time (64 bits)
b#16#02 *	STRING	Character string

* The string uses the lower-level data type BYTE. Refer to the special features in the section Data types (Page 6746).

Memory area

The following table explains the coding of the memory area in the "MemArea" parameter of the UDT UAAnyPointer.

Table 4-434 Coding of the memory area

Hexadecimal code	Area	Description
b#16#80	P	Memory area of the I/O
b#16#81	E	Memory area of the inputs
b#16#82	A	Memory area of the outputs
b#16#83	M	Memory area of the bit memory
b#16#84	DB	Data block

Error numbers

Evaluation of the output parameters

The output parameters Done (BOOL), Error (BOOL) and ErrorID (DWORD) of the OPC UA client program blocks are updated each time the block is called. Evaluate the states of the output parameters in the user program.

Example:

After it is called, the program block UA_Connect returns the following values at the three output parameters.

- Done = 1
- Error = 0
- ErrorID = 00000000

Meaning: The job was completed successfully.

Meaning of the error numbers (hexadecimal) of ErrorID

Table 4-435 Meaning of ErrorID in conjunction with Done and Error

Done	Error	ErrorID (hex)	Name	Meaning
1	0	00000000		The block execution was completed successfully.
CP-specific error numbers				
0	1	B0006001		Missing positive edge for the program block call
0	1	B0007001		Timeout
0	1	B0008004		Invalid NodeIDCount
0	1	B0008086		Invalid node handle
0	1	B0008087		The maximum number of connections (5) has been reached.
0	1	B0008090		Wrong logical address. Connection establishment aborted.
0	1	B0008093		The logical address cannot be processed. An impermissible module is being used.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	B00080A1		The node handle was not set up for this connection.
0	1	B00080C3		The connection resources of the CPU are temporarily exhausted.
0	1	B0009085		The maximum memory space for items (64000 bytes) of the OPC UA client has been exceeded.
0	1	B000800A		The OPC client was not enabled in the configuration.
0	1	B0008723		The read/write job references a non-existent data area in a DB.
0	1	B000F001		Two function blocks were called in one cycle.
0	1	B000F002		OPC UA session not yet established. Call the UA_Connect block again.
0	1	B000F003		Communication error
0	1	B000F004		Invalid value for the parameter "SyntaxID" in the UDT "UAAnyPointer"
0	1	B000F005		Invalid data type in UAAnyPointer
0	1	B000F006		Invalid value for the parameter "RepetitionFactor" in the UDT "UAAnyPointer"
0	1	B000F007		Invalid operand area of UAAnyPointer
0	1	B000F010		Invalid operand area in a read or write job: Arrays of inputs, arrays of outputs, arrays of memory bits, arrays of local data and arrays of local data of the predecessor block are not permitted.
0	1	B000F011		The bit offset must be zero.
0	1	B0010001		Wrong data packet in UA_Connect
0	1	B0010002		Invalid URL of a server end point
0	1	B0010003		Invalid value for Timeout
0	1	B0010004		Invalid value for SessionTimeout
0	1	B0010005		Invalid value for MonitorConnection
0	1	B0010006		Invalid transport profile
0	1	B0010007		Invalid server URL
0	1	B0010008		Invalid Security procedure
0	1	B0010009		Invalid Security profile
0	1	B001000A		Invalid value for UserIdentityToken
0	1	B001000B		Invalid application name
0	1	B0010101		Invalid format of the server end point URL. It must start with "opc.tcp://".
0	1	B0010102		The path of the server end point URL is not supported.
0	1	B0010103		The client must not connect itself to the server of its own CP.
0	1	B0010104		Invalid port
0	1	B0010105		Invalid address. DNS is not supported.
0	1	B0020001		Wrong data packet in UA_NameSpaceGetIndex
0	1	B0030001		Wrong data packet in UA_NodeGetHandleList
0	1	B0030002		Invalid NodeID Identifier
0	1	B0030003		Max. number of items (64000) exceeded
0	1	B0030004		Unsupported attribute "ValueRank" of a tag. The ValueRank values 1 (one dimensional array) and -1 (scalar) are supported.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	B0040001		Wrong data packet in UA_NodeReleaseHandleList
0	1	B0050001		Wrong data packet in UA_Disconnect
0	1	B0050002		UA_Disconnect cannot be called, because there is no connection.
0	1	B0060001		Wrong data packet in UA_ReadList
0	1	B0060002		The length of the data to be read exceeds the maximum of 64000.
0	1	B0060003		Invalid range of IndexRangeCount
0	1	B0060004		Invalid IndexRange. StartIndex must not be higher than EndIndex and EndIndex must not be higher than the array length.
0	1	B0060005		Invalid AttributeID
0	1	B0060006		The read data area is too small. Correct the data in the UDT "UAIndexRange".
0	1	B0060007		Invalid string length. The value (user data) must be 254.
0	1	B0060008		Invalid value of the node. The value cannot be read into the S7 tag.
0	1	B0060010		Invalid operand area in a read or write job: Arrays of inputs, arrays of outputs, arrays of memory bits, arrays of local data and arrays of local data of the predecessor block are not permitted.
0	1	B0070001		Wrong data packet in UA_WriteList
0	1	B0070002		The length of the data to be written exceeds the maximum.
0	1	B0070003		Invalid range of IndexRangeCount
0	1	B0070004		Invalid IndexRange. StartIndex must not be higher than EndIndex and EndIndex must not be higher than the array length.
0	1	B0070005		Invalid AttributeID
0	1	B0070006		The data area is outside the permitted area.
0	1	B0070009		Invalid character in the string
Error numbers according to IEC 61131-3 for OPC UA				
0	1	002D0000	GoodSubscriptionTransferred	The subscription was transferred to another session.
0	1	002E0000	GoodCompletesAsynchronously	The processing will complete asynchronously.
0	1	002F0000	GoodOverload	Sampling has slowed down due to resource limitations.
0	1	00300000	GoodClamped	The value written was accepted but was clamped.
0	1	00960000	GoodLocalOverride	The value has been overridden.
0	1	00A20000	GoodEntryInserted	The data or event was successfully inserted into the historical database.
0	1	00A30000	GoodEntryReplaced	The data or event field was successfully replaced in the historical database.
0	1	00A50000	GoodNoData	No data exists for the requested time range or event filter.
0	1	00A60000	GoodMoreData	The data or event field was successfully replaced in the historical database.
0	1	00A70000	GoodCommunicationEvent	The communication layer has raised an event.
0	1	00A80000	GoodShutdownEvent	The system is shutting down.
0	1	00A90000	GoodCallAgain	The operation is not finished and needs to be called again.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	00AA0000	GoodNonCriticalTimeout	A non-critical timeout occurred.
0	1	00BA0000	GoodResultsMayBe Incomplete	The server should have followed a reference to a node in a remote server but did not. The result set may be incomplete.
0	1	406C0000	UncertainReferenceOutOf Server	One of the references to follow in the relative path references to a node in the address space in another server.
0	1	408F0000	UncertainNoCommunication LastUsableValue	Communication to the data source has failed. The variable value is the last value that had a good quality.
0	1	40900000	UncertainLastUsableValue	Whatever was updating this value has stopped doing so.
0	1	40910000	UncertainSubstituteValue	The value is an operational value that was manually overwritten.
0	1	40920000	UncertainInitialValue	The value is an initial value for a variable that normally receives its value from another variable.
0	1	40930000	UncertainSensorNotAccurate	The value is at one of the sensor limits.
0	1	40940000	UncertainEngineeringUnits Exceeded	The value is outside of the range of values defined for this parameter.
0	1	40950000	UncertainSubNormal	The value is derived from multiple sources and has less than the required number of Good sources.
0	1	40A40000	UncertainDataSubNormal	The value is derived from multiple values and has less than the required number of Good values.
0	1	40BC0000	UncertainReferenceNot Deleted	The server was not able to delete all target references.
0	1	40C00000	UncertainNotAllNodes Available	The list of references may not be complete because the underlying system is not available.
0	1	80010000	BadUnexpectedError	An unexpected error occurred
0	1	80020000	BadInternalError	An internal error occurred as a result of a programming or configuration error.
0	1	80030000	BadOutOfMemory	Not enough memory to complete the operation.
0	1	80040000	BadResourceUnavailable	An operating system resource is not available
0	1	80050000	BadCommunicationError	A low level communication error occurred.
0	1	80060000	BadEncodingError	Encoding halted because of invalid data in the objects being serialized.
0	1	80070000	BadDecodingError	Decoding halted because of invalid data in the stream.
0	1	80080000	BadEncodingLimitsExceeded	The message encoding/decoding limits imposed by the stack have been exceeded.
0	1	80B80000	BadRequestTooLarge	The resources (memory) of the CPU are temporarily occupied
0	1	80B90000	BadResponseTooLarge	The response message size exceeds limits set by the client.
0	1	80B90000	BadUnknownResponse	An unrecognized response was received from the server.
0	1	800A0000	BadTimeout	The operation timed out.
0	1	800B0000	BadServiceUnsupported	The server does not support the requested service.
0	1	800C0000	BadShutdown	The operation was cancelled because the application is shutting down.
0	1	800D0000	BadServerNotConnected	The operation could not complete because the client is not connected to the server.
0	1	800E0000	BadServerHalted	The server has stopped and cannot process any requests.
0	1	800F0000	BadNothingToDo	There was nothing to do because the client passed a list of operations with no elements.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	80100000	BadTooManyOperations	The request could not be processed because it specified too many operations.
0	1	80110000	BadDataTypeIdUnknown	The extension object cannot be (de)serialized because the data type id is not recognized.
0	1	80120000	BadCertificateInvalid	The certificate provided as a parameter is not valid.
0	1	80130000	BadSecurityChecksFailed	The certificate provided as a parameter is not valid.
0	1	80140000	BadCertificateTimeInvalid	The Certificate has expired or is not yet valid.
0	1	80150000	BadCertificateIssuer TimeInvalid	An Issuer Certificate has expired or is not yet valid.
0	1	80160000	BadCertificateHost NameInvalid	The HostName used to connect to a Server does not match a HostName in the Certificate.
0	1	80170000	BadCertificateUriInvalid	The URI specified in the Application Description does not match the URI in the Certificate.
0	1	80180000	BadCertificateUseNotAllowed	The Certificate may not be used for the requested operation.
0	1	80190000	BadCertificateIssuerUse NotAllowed	The Issuer Certificate may not be used for the requested operation.
0	1	801A0000	BadCertificateUntrusted	The Certificate is not trusted.
0	1	801B0000	BadCertificateRevocation Unknown	It was not possible to determine if the Certificate has been revoked.
0	1	801C0000	BadCertificateIssuer RevocationUnknown	It was not possible to determine if the Issuer Certificate has been revoked.
0	1	801D0000	BadCertificateRevoked	The Certificate has been revoked.
0	1	801E0000	BadCertificateIssuerRevoked	The Issuer Certificate has been revoked.
0	1	801F0000	BadUserAccessDenied	User does not have permission to perform the requested operation.
0	1	80200000	BadIdentityTokenInvalid	The user identity token is not valid.
0	1	80210000	BadIdentityTokenRejected	The user identity token is valid but the server has rejected it.
0	1	80220000	BadSecureChannelIdInvalid	The specified secure channel is no longer valid.
0	1	80230000	BadInvalidTimestamp	The timestamp is outside the range allowed by the server.
0	1	80240000	BadNonceInvalid	The nonce does appear to be not a random value or it is not the correct length.
0	1	80250000	BadSessionIdInvalid	The session id is not valid.
0	1	80260000	BadSessionClosed	The session was closed by the client.
0	1	80270000	BadSessionNotActivated	The session cannot be used because ActivateSession has not been called.
0	1	80280000	BadSubscriptionIdInvalid	The subscription id is not valid.
0	1	802A0000	BadRequestHeaderInvalid	The header for the request is missing or invalid.
0	1	802B0000	BadTimestampsTo ReturnInvalid	The timestamps to return parameter is invalid.
0	1	802C0000	BadRequestCancelled ByClient	The request was cancelled by the client.
0	1	80310000	BadNoCommunication	Communication with the data source is defined, but not established, and there is no last known value available.
0	1	80320000	BadWaitingForInitialData	Waiting for the server to obtain values from the underlying data source.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	80330000	BadNodeIdInvalid	The syntax of the node id is not valid.
0	1	80340000	BadNodeIdUnknown	The node id refers to a node that does not exist in the server address space.
0	1	80350000	BadAttributeIdInvalid	The attribute is not supported for the specified Node.
0	1	80360000	BadIndexRangeInvalid	The syntax of the index range parameter is invalid.
0	1	80370000	BadIndexRangeNoData	No data exists within the range of indexes specified.
0	1	80380000	BadDataEncodingInvalid	The data encoding is invalid.
0	1	80390000	BadDataEncoding Unsupported	The server does not support the requested data encoding for the node.
0	1	803A0000	BadNotReadable	The access level does not allow reading or subscribing to the Node.
0	1	803B0000	BadNotWritable	The access level does not allow writing to the Node.
0	1	803C0000	BadOutOfRange	The value was out of range.
0	1	803D0000	BadNotSupported	The requested operation is not supported.
0	1	803E0000	BadNotFound	A requested item was not found or a search operation ended without success.
0	1	803F0000	BadObjectDeleted	The object cannot be used because it has been deleted.
0	1	80400000	BadNotImplemented	Requested operation is not implemented.
0	1	80410000	BadMonitoringModeInvalid	The monitoring mode is invalid.
0	1	80420000	BadMonitoredItemIdInvalid	The monitoring item id does not refer to a valid monitored item.
0	1	80430000	BadMonitoredItem FilterInvalid	The monitored item filter parameter is not valid.
0	1	80440000	BadMonitoredItem FilterUnsupported	The server does not support the requested monitored item filter.
0	1	80450000	BadFilterNotAllowed	A monitoring filter cannot be used in combination with the attribute specified.
0	1	80460000	BadStructureMissing	A mandatory structured parameter was missing or null.
0	1	80470000	BadEventFilterInvalid	The event filter is not valid.
0	1	80480000	BadContentFilterInvalid	The content filter is not valid.
0	1	80490000	BadFilterOperandInvalid	The operand used in a content filter is not valid.
0	1	804A0000	BadContinuation PointInvalid	The continuation point provide is longer valid.
0	1	804B0000	BadNoContinuationPoints	The operation could not be processed because all continuation points have been allocated.
0	1	804C0000	BadReferenceTypeIdInvalid	The operation could not be processed because all continuation points have been allocated.
0	1	804D0000	BadBrowseDirectionInvalid	The browse direction is not valid.
0	1	804E0000	BadNodeNotInView	The node is not part of the view.
0	1	804F0000	BadServerUriInvalid	The ServerUri is not a valid URI.
0	1	80500000	BadServerNameMissing	No ServerName was specified
0	1	80510000	BadDiscoveryUrlMissing	No DiscoveryUrl was specified.
0	1	80520000	BadSempahoreFileMissing	The semaphore file specified by the client is not valid.
0	1	80530000	BadRequestTypeInvalid	The security token request type is not valid.
0	1	80540000	BadSecurityModeRejected	The security mode does not meet the requirements set by the Server.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	80550000	BadSecurityPolicyRejected	The security policy does not meet the requirements set by the Server.
0	1	80560000	BadTooManySessions	The server has reached its maximum number of sessions.
0	1	80570000	BadUserSignatureInvalid	The user token signature is missing or invalid.
0	1	80580000	BadApplicationSignature Invalid	The signature generated with the client certificate is missing or invalid.
0	1	80590000	BadNoValidCertificates	The client did not provide at least one software certificate that is valid and meets the profile requirements for the server.
0	1	805A0000	BadRequestCancelled ByRequest	The request was cancelled by the client with the Cancel service.
0	1	805B0000	BadParentNodeIdInvalid	The parent node id does not to refer to a valid node.
0	1	805C0000	BadReferenceNotAllowed	The reference could not be created because it violates constraints imposed by the data model.
0	1	805D0000	BadNodeIdRejected	The requested node id was reject because it was either invalid or server does not allow node ids to be specified by the client.
0	1	805F0000	BadNodeClassInvalid	The node class is not valid.
0	1	80600000	BadBrowseNameInvalid	The browse name is invalid.
0	1	80610000	BadBrowseNameDuplicated	The browse name is not unique among nodes that share the same relationship with the parent.
0	1	80620000	BadNodeAttributesInvalid	The node attributes are not valid for the node class.
0	1	80630000	BadTypeDefinitionInvalid	The type definition node id does not reference an appropriate type node.
0	1	80640000	BadSourceNodeIdInvalid	The source node id does not reference a valid node.
0	1	80650000	BadTargetNodeIdInvalid	The target node id does not reference a valid node.
0	1	80660000	BadDuplicateReference NotAllowed	The reference type between the nodes is already defined.
0	1	80670000	BadInvalidSelfReference	The server does not allow this type of selfreference on this node.
0	1	80680000	BadReferenceLocalOnly	The reference type is not valid for a reference to a remote server.
0	1	80690000	BadNoDeleteRights	The server will not allow the node to be deleted.
0	1	806A0000	BadServerIndexInvalid	The server index is not valid.
0	1	806B0000	BadViewIdUnknown	The view id does not refer to a valid view node.
0	1	806D0000	BadTooManyMatches	The requested operation has too many matches to return.
0	1	806E0000	BadQueryTooComplex	The requested operation requires too many resources in the server.
0	1	806F0000	BadNoMatch	The requested operation has no match to return.
0	1	80700000	BadMaxAgeInvalid	The max age parameter is invalid.
0	1	80710000	BadHistoryOperationInvalid	The history details parameter is not valid.
0	1	80720000	BadHistoryOperation Unsupported	The server does not support the requested operation.
0	1	80730000	BadWriteNotSupported	The server not does support writing the combination of value, status and timestamps provided.
0	1	80740000	BadTypeMismatch	The value supplied for the attribute is not of the same type as the attribute's value.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	80750000	BadMethodInvalid	The method id does not refer to a method for the specified object.
0	1	80760000	BadArgumentsMissing	The client did not specify all of the input arguments for the method.
0	1	80770000	BadTooManySubscriptions	The server has reached its maximum number of subscriptions.
0	1	80780000	BadTooManyPublish Requests	The server has reached the maximum number of queued publish requests.
0	1	80790000	BadNoSubscription	There is no subscription available for this session.
0	1	807A0000	BadSequenceNumber Unknown	The sequence number is unknown to the server.
0	1	807B0000	BadMessageNotAvailable	The requested notification message is no longer available.
0	1	807C0000	BadInsufficientClientProfile	The Client of the current Session does not support one or more Profiles that are necessary for the Subscription.
0	1	80BF0000	BadStateNotActive	The sub-state machine is not currently active.
0	1	807D0000	BadTcpServerTooBusy	The server cannot process the request because it is too busy.
0	1	807E0000	BadTcpMessageTypeInvalid	The type of the message specified in the header invalid.
0	1	807F0000	BadTcpSecureChannel Unknown	The SecureChannelId and/or TokenId are not currently in use.
0	1	80800000	BadTcpMessageTooLarge	The size of the message specified in the header is too large.
0	1	80810000	BadTcpNotEnough Resources	There are not enough resources to process the request.
0	1	80820000	BadTcpInternalError	An internal error occurred.
0	1	80830000	BadTcpEndpointUrlInvalid	The Server does not recognize the QueryString specified.
0	1	80840000	BadRequestInterrupted	The request could not be sent because of a network interruption.
0	1	80850000	BadRequestTimeout	Timeout occurred while processing the request.
0	1	80860000	BadSecureChannelClosed	The secure channel has been closed.
0	1	80870000	BadSecureChannelToken Unknown	The token has expired or is not recognized.
0	1	80880000	BadSequenceNumberInvalid	The sequence number is not valid.
0	1	80890000	BadConfigurationError	There is a problem with the configuration that affects the usefulness of the value.
0	1	808A0000	BadNotConnected	The variable should receive its value from another variable, but has never been configured to do so.
0	1	808B0000	BadDeviceFailure	There has been a failure in the device/data source that generates the value that has affected the value.
0	1	808C0000	BadSensorFailure	There has been a failure in the sensor from which the value is derived by the device/data source.
0	1	808D0000	BadOutOfService	The source of the data is not operational.
0	1	808E0000	BadDeadbandFilterInvalid	The dead band filter is not valid.
0	1	80970000	BadRefreshInProgress	This Condition refresh failed, a Condition refresh operation is already in progress.
0	1	80980000	BadConditionAlreadyDisabled	This condition has already been disabled.
0	1	80990000	BadConditionDisabled	Property not available, this condition is disabled.
0	1	809A0000	BadEventIdUnknown	The specified event id is not recognized.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	809B0000	BadNoData	No data exists for the requested time range or event filter.
0	1	809D0000	BadDataLost	Data is missing due to collection started/stopped/lost.
0	1	809E0000	BadDataUnavailable	Expected data is unavailable for the requested time range due to an unmounted volume an off-line archive or tape or similar reason for temporary unavailability.
0	1	809F0000	BadEntryExists	The data or event was not successfully inserted because a matching entry exists.
0	1	80A00000	BadNoEntryExists	The data or event was not successfully updated because no matching entry exists.
0	1	80A10000	BadTimestampNotSupported	The client requested history using a timestamp format the server does not support (i. e. requested ServerTimestamp when server only supports SourceTimestamp).
0	1	80AB0000	BadInvalidArgument	One or more arguments are invalid.
0	1	80AC0000	BadConnectionRejected	Could not establish a network connection to remote server.
0	1	80AD0000	BadDisconnect	The server has disconnected from the client.
0	1	80AE0000	BadConnectionClosed	The network connection has been closed.
0	1	80AF0000	BadInvalidState	The operation cannot be completed because the object is closed uninitialized or in some other invalid state.
0	1	80B00000	BadEndOfStream	Cannot move beyond end of the stream.
0	1	80B10000	BadNoDataAvailable	No data is currently available for reading from a non-blocking stream.
0	1	80B20000	BadWaitingForResponse	The asynchronous operation is waiting for a response.
0	1	80B30000	BadOperationAbandoned	The asynchronous operation was abandoned by the caller.
0	1	80B40000	BadExpectedStreamToBlock	The stream did not return all data requested (possibly because it is a non-blocking stream).
0	1	80B50000	BadWouldBlock	Non-blocking behavior is required and the operation would block.
0	1	80B60000	BadSyntaxError	A value had an invalid syntax.
0	1	80B70000	BadMaxConnections Reached	The operation could not be finished because all available connections are in use.
0	1	80BB0000	BadEventNot Acknowledgeable	The event cannot be acknowledged.
0	1	80BD0000	BadInvalidTimestamp Argument	The defined timestamp to return was invalid.
0	1	80BE0000	BadProtocolVersion Unsupported	The applications do not have compatible protocol versions.
0	1	80C10000	BadFilterOperatorInvalid	An unrecognized operator was provided in a filter.
0	1	80C20000	BadFilterOperator Unsupported	A valid operator was provided, but the server does not provide support for this filter operator.
0	1	80C30000	BadFilterOperandCount Mismatch	The number of operands provided for the filter operator was less than expected for the operand provided.
0	1	80C40000	BadFilterElementInvalid	The referenced element is not a valid element in the content filter.
0	1	80C50000	BadFilterLiteralInvalid	The referenced literal is not a valid value.
0	1	80C90000	BadViewTimestampInvalid	The view timestamp is not available or not supported.
0	1	80CA0000	BadViewParameterMismatch	The view parameters are not consistent with each other.
0	1	80CB0000	BadViewVersionInvalid	The view version is not available or not supported.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	80CC0000	BadConditionAlready Enabled	This condition has already been enabled.
0	1	80CD0000	BadDialogNotActive	The dialog condition is not active.
0	1	80CF0000	BadConditionBranch AlreadyAcked	The condition branch has already been acknowledged.
0	1	80D00000	BadConditionBranch AlreadyConfirmed	The condition branch has already been confirmed.
0	1	80D10000	BadConditionAlready-Shelved	The condition has already been shelved.
0	1	80D20000	BadConditionNotShelved	The condition is not currently shelved.
0	1	80D30000	BadShelvingTimeOutO-fRange	The shelving time not within an acceptable range.
0	1	80D40000	BadAggregateListMismatch	The requested number of Aggregates does not match the requested number of NodeIds.
0	1	80D50000	BadAggregateNotSupported	The requested Aggregate is not support by the server.
0	1	80D60000	BadAggregateInvalidInputs	The aggregate value could not be derived due to invalid data inputs.
0	1	80DB0000	BadTooManyMonitoredItems	The request could not be processed because there are too many monitored items in the subscription.
0	1	80D70000	BadBoundNotFound	No data found to provide upper or lower bound value.
0	1	80D80000	BadBoundNotSupported	The server cannot retrieve a bound for the variable.
0	1	00D90000	GoodDataIgnored	The request specifies fields which are not valid for the EventType or cannot be saved by the historian.
0	1	80DA0000	BadAggregateConfiguration Rejected	The aggregate configuration is not valid for specified node.
Error numbers of PLCOpen for OPC UA				
0	1	A0000001	PLCopenUA_Bad_FW_ PermanentError	Internal, permanent error.
0	1	A0000002	PLCopenUA_Bad_FW_ TempError	Temp. Error; FB could retry to reach FW.
0	1	A0000003	PLCopenUA_Bad_ ConnectionError	Connection could not be established.
0	1	A0000004	PLCopenUA_Bad_ HostNotFound	The requested hostname could not be found.
0	1	A0000005	PLCopenUA_Bad_ AlreadyConnected	Connection was already established.
0	1	A0000006	PLCopenUA_Bad_ SecurityFailed	Connection failed due to security setup.
0	1	A0000007	PLCopenUA_Bad_ Suspended	Connection is suspended.
0	1	A0000008	PLCopenUA_Bad_ ConnectionInvalidHdl	Provided ConnectionHdl is not known.
0	1	A0000009	PLCopenUA_Bad_ NSNotFound	A namespace with the requested name cannot be found on server.
0	1	A000000A	PLCopenUA_Bad_ ResultTooLong	Target PLC variable is too short for retrieved data.

Done	Error	ErrorID (hex)	Name	Meaning
0	1	A000000B	PLCopenUA_Bad_ InvalidType	Invalid or unsupported Type.
0	1	A000000C	PLCopenUA_Bad_ NodeInvalidHdl	Provided NodeHdl is not known.
0	1	A000000D	PLCopenUA_Bad_ MethodInvalidHdl	Provided MethodHdl is not known.
0	1	A000000E	PLCopenUA_Bad_ ReadFailed	Read failed for unknown reason.
0	1	A000000F	PLCopenUA_Bad_ WriteFailed	Write failed for unknown reason.
0	1	A0000010	PLCopenUA_Bad_ CallFailed	Method Call failed for unknown reason.
0	1	A0000011	PLCopenUA_Bad_ InParamFailed	Method Call Input parameter conversion failed.
0	1	A0000012	PLCopenUA_Bad_ OutParamFailed	Method Call Output parameter conversion failed. ATTEN-TION: this means the MethodCall was executed successfully but the returned values could not be converted.
0	1	A0000013	PLCopenUA_Bad_ SubscriptionInvalidHdl	Provided SubscriptionHdl is not known.
0	1	A0000014	PLCopenUA_Bad_ MonitoredItemInvalidHdl	Provided MonitoredItemHdl is not known.

4.2.5.7 300C functions

ASCII, 3964(R)

SEND_PTP: Send data (ASCII, 3964(R))

Description

You use the SEND_PTP instruction to send a block of data from a data block.

- The sending process is activated after the block is called and a positive edge is detected at the control input REQ.
- The range of data to be transmitted is specified by SD_1 (DB number and start address), and the length of the data block by LEN.
- For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at the control input R, an active sending process will be aborted and the instruction reset to the initial state. A canceled job is terminated with an error message (STATUS output).
- Use LADDR to specify the submodule I/O address, which you have defined during hardware configuration.
- DONE will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.

If the job was processed with DONE = TRUE this means that:

- When using the ASCII driver: The data where transmitted to the communication partner. It is not ensured that all data were received by the communication partner.
- When using the procedure 3964(R): The data have been transmitted to the communication partner and they where acknowledged positively by the partner. It is not ensured that the data were also passed onto the partner CPU.

In STATUS, the CPU indicates the respective event ID as the result of an error or warning. DONE or ERROR/STATUS are also output on RESET of the instruction (R = TRUE).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the instruction was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect.

Instance DB

The "SEND_PTP" instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Parameters

The following table shows the parameters of the instruction "SEND_PTP":

Parameters	Declaration	Data type	Range of values	Default	Description
REQ	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Request: activates data exchange at the positive edge.
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset. Job is aborted. Transmission is disabled.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The I/O address of the submodule, which you have specified during hardware configuration.
DONE	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): • FALSE: Job not yet started or still running. • TRUE: Job was executed error-free.
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with error

Parameters	Declaration	Data type	Range of values	Default	Description
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR = TRUE: An error has occurred, STATUS supplies detailed information on the type of error.
SD_1	InOut	ANY	CPU-specific	0	Send parameters: Here you enter the following values: • The number of the DB from which the data will be transmitted. • The data byte number, starting from which data will be transmitted. Possible values: 0 to 8190 For example: DB 10 from byte 2 -> DB10.DBB2 Note: Note that for S7-300 CPUs, the parameter SD_1 always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
LEN	InOut	INT	1 to 1024	1	Here you specify the length of the data block that is to be transmitted (length is set here indirectly).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Data consistency

Data consistency is limited to 206 bytes. If you want consistent data transmission exceeding 206 bytes, you must take the following into account:

Do not write to the currently used section of the send range SD_1 again until the sending process has concluded. This is the case when the status parameter DONE assumes the value TRUE.

RCV_PTP: Fetch data (ASCII, 3964(R))

Description

You use the RCV_PTP instruction to receive data and then store them in a data block.

- After the instruction has been called with the value TRUE at control input EN_R, it is ready to receive data. You cancel the current transmission by setting the signal status of parameter EN_R to FALSE. A canceled job is terminated with an error message (STATUS output). The input is switched off as long as the signal status of parameter EN_R is set to FALSE.
- The receive area is specified by RD_1 (DB number and start address), and the length of the data block by LEN.
- For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at control input R, the current transmission will be canceled and the instruction reset to its initial state. A canceled receive job is terminated with an error message (STATUS output).
- Use LADDR to specify the submodule I/O address, which you have defined during hardware configuration.
- NDR will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.
- In STATUS , the CPU indicates the respective event ID as the result of an error or warning.
- NDR or ERROR/STATUS are also output on RESET of the instruction (R = TRUE) (parameter LEN == 16#00).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the instruction was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect.

Instance DB

The "RCV_PTP" instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Parameters

The following table shows the parameters of the instruction "RCV_PTP":

Parameters	Declaration	Data type	Range of values	Default	Description
EN_R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Enable to receive: Receive enable
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset: Job is aborted.

Parameters	Declaration	Data type	Range of values	Default	Description
LADDR	Input	WORD	CPU-specific	W#16#03FF	The I/O address of the submodule, which you have specified during hardware configuration.
NDR	Output	BOOL	TRUE/FALSE	FALSE	Job done without error, data were accepted • FALSE: Job not started yet or still running • TRUE: Job completed without errors
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with error
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR = TRUE: An error has occurred, STATUS supplies detailed information on the type of error.
RD_1	InOut	ANY	CPU-specific	0	Receive parameter: Here you specify: • The number of the DB in which the received data are to be stored. • The data byte number starting at which the received data will be stored. Possible values: 0 to 8190 For example: DB 20 starting with byte 5 -> DB20.DBB5 Note: Note that for S7-300 CPUs, the parameter RD_1 always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
LEN	InOut	INT	0 to 1024	0	Output of the data length (number of bytes)

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Data consistency

Data consistency is limited to 206 bytes. If you want consistent data transmission exceeding 206 bytes, you must take the following into account:

Do not access the receive DB again until the data have been completely received (NDR = TRUE). Then, disable the receiving DB (EN_R = FALSE) until you have processed the data.

RES_RCVB: Reset receive buffer (ASCII, 3964(R))

Description

You use the RES_RECV instruction to delete the entire receive buffer of the module. All stored message frames are discarded. A message frame that is incoming when the RES_RCVB instruction is called will be saved.

- The job is activated after the instruction is called and a positive edge is detected at control input REQ. The job can run across multiple calls (program cycles).
- For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at control input R, the deleting process will be canceled and the instruction reset to its initial state. A canceled job is terminated with an error message (STATUS output).
- Use LADDR to specify the submodule I/O address, which you have defined during hardware configuration.
- DONE will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.
- In STATUS the CPU indicates the respective event ID as the result of an error or warning.
- DONE or ERROR/STATUS are also output on RESET of the instruction (R = TRUE).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the block was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect.

Instance DB

The RES_RCVB instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Parameter

The following table shows the parameters of the instruction RES_RCVB:

Parameter	Declaration	Data type	Range of values	Default	Description
REQ	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Request: Activates the job on a positive edge.
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset: Job is aborted.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The I/O address of the submodule, which you have specified during hardware configuration.
DONE	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): • FALSE: Job not yet started or still running. • TRUE: Job was executed error-free.
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with error
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR = TRUE: An error has occurred, STATUS supplies detailed information on the type of error.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

RK 512

SEND_RK: Send data (RK 512)

Description

You use the SEND_RK to send a block of data from a data block.

- The send job is activated after the instruction is called and a positive edge is detected at control input REQ.
- The range of data to be transmitted is specified by SD_1 (DB number and start address), and the length of the data block by LEN.
- You also specify the receive area on the partner in the instruction. The CPU enters this information in the message frame header and transfers it to the partner.
- The destination is specified by the CPU number in R_CPU (only relevant for multiprocessor communication), the data type in R_TYPE (data blocks (DB) and expanded data blocks (DX)), the data block number in R_DBNO and the offset in R_OFFSET to which the first byte is to be written.
- In R_CF_BYT and R_CF_BIT you declare the interprocessor communication flag byte and bit on the partner CPU.
- With the SYNC_DB parameter, you specify the DB where the data that are common to all instructions used for initialization during startup and synchronization will be stored. The DB number must be identical for all instructions used in your user program.
- For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at control input R, the active sending process will be aborted and the instruction reset to the initial state. A canceled job is terminated with an error message (STATUS output).
- Use LADDR to specify the submodule I/O address, which you have defined during hardware configuration.
- DONE will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.
- If the job was processed with DONE = TRUE, the data were sent to the communication partner, the communication partner acknowledged them positively, and they were then passed on to the partner CPU.
- In STATUS, the CPU indicates the respective event ID as the result of an error or warning.
- DONE or ERROR/STATUS are also output on RESET of the instruction (R = TRUE).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the block was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect.

Instance DB

The SEND_RK instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Special features when sending data

Take the following special features into account when "sending data":

- With RK 512 you can only send an even number of data. If you declare an odd length (LEN) of data, an additional fill byte with the value "0" will be appended to the transmitted data.
- In RK 512 you can only declare an even offset. If you declare an odd offset the data are stored in the partner station from the next lower even offset.
  Example: Offset is "7", the data will be stored as of byte "6".

Parameters

The following table shows the parameters of the instruction SEND_RK:

Parameters	Declaration	Data type	Range of values	Default	Description
SYNC_DB	Input	INT	CPU-specific	0	Number of the DB where the common data for the synchronization of the RK instructions are stored (minimum length = 240 bytes).
REQ	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Request: Activates the job on a positive edge.
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset: Job is aborted.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The I/O address of the submodule, which you have specified during hardware configuration.
R_CPU	Input	INT	0 to 4	1	CPU number of the partner CPU (only for multiprocessor operation)
R_TYPE	Input	CHAR	‘D’, ‘X’	'D'	Address type on the partner CPU (only uppercase allowed) 'D': Data block 'X': Expanded data block
R_DBNO	Input	INT	0 to 255	0	Data block number on the partner CPU
R_OFFSET	Input	INT	0 to 510 (even values only)	0	Data byte number on the partner CPU
R_CF_BYT	Input	INT	0 to 255	255	Communication flag byte on partner CPU (255: Means: without interprocessor communication flag)
R_CF_BIT	Input	INT	0 to 7	0	Communication flag bit on the partner CPU

Parameters	Declaration	Data type	Range of values	Default	Description
DONE	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): • FALSE: Job not yet started or still running. • TRUE: Job was executed error-free.
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with error
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR=TRUE: An error has occurred, STATUS supplies detailed information on the type of error.
SD_1	Input	ANY	CPU-specific	0	Send parameters: Here you specify: • The number of the DB from which the data will be transmitted. • The data byte number, starting from which data will be transmitted. Possible values: 0 to 8190 For example: DB 10 from byte 2 -> DB10.DBB2 Note: Note that for S7-300 CPUs, the parameter SD_1 always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
LEN	Input	INT	1 to 1024	1	Here you specify the length of the data block that is to be transmitted (length is set here indirectly).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Data in the message frame header

The table below shows the data in the message frame header of the RK 512 message frame.

Source on your S7 automation system (local CPU)	To target, partner CPU	Message frame header, bytes		
		3/4 command type	5/6 D-DBNO[1] / D offset[2]	7/8 number in
Data block	Data block	AD	DB/DW[3]	Words
Data block	Expanded data block	AD	DB/DW[3]	Words

[1] D-DBNO: Destination data block number
[2] D offset: Destination start address
[3] DW: Offset in words

Data consistency

Data consistency is limited to 128 bytes. If you want consistent data transmission exceeding 128 bytes, you must take the following into account:

Do not write to the currently used section of the send range SD_1 again until the sending process has concluded. This is the case when the status parameter DONE assumes the value TRUE.

FETCH_RK: Fetch data (RK 512)

Description

You use the FETCH_RK instruction to retrieve a block of data from the partner and store the data in a data block.

- The sending process is activated after the block is called and a positive edge is detected at the control input REQ.
- The area in which the retrieved data are stored is specified by RD_1 (DB number and start address), and the length of the data block by LEN.
- You also specify the partner area from which the data will be fetched for the instruction. The CPU enters this information in the RK512 message frame header and transfers it to the partner.
- The partner area is determined by the CPU number in R_CPU (only relevant for multiprocessor communication), the data type in R_TYPE (data blocks, expanded data blocks, memory bits, inputs, outputs, counters and timers), the data block number in R_DBNO (only relevant for data blocks and expanded data blocks) and the offset in R_OFFSET, from where the first byte is to be retrieved.
- Use R_CF_BYT and R_CF_BIT to declare the interprocessor communication flag byte and bit on the partner CPU.
- With the SYNC_DB parameter, you specify the DB where the data that are common to all instructions used for initialization during startup and synchronization will be stored. The DB number must be identical for all instructions used in your user program.

- For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at control input R, the current transmission will be canceled and the instruction reset to its initial state. A canceled job is terminated with an error message (STATUS output).
- Use LADDR to specify the submodule I/O address, which you have defined during hardware configuration.
- DONE will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.
- In STATUS, the CPU indicates the respective event ID as the result of an error or warning.
- DONE or ERROR/STATUS are also output on RESET of the instruction (R = TRUE).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the instruction was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect. If data will be retrieved from your CPU, you must program a SERVE_RK (Page 6780) instruction on your CPU.

Instance DB

The FETCH_RK instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Special features for (expanded) data blocks

Note the following special features when "retrieving data" from a data block and expanded data blocks:

- With RK 512 you can only send an even number of data. If you declare an odd length (LEN) of data, an additional byte will always be appended to the transmitted data. In the destination DB, however, the correct number of data is always entered.
- In RK 512 you can only declare an even offset. If you declare an odd offset the data are stored in the partner station starting with the next smaller even offset.
  Example: Offset is 7, the data are stored from byte 6.

Special features for timers and counters

When you retrieve times or counters from your communication partner, you must take into account that you must retrieve two bytes for every time or counter. For example, if you want to retrieve 10 counters you must declare a length of 20.

Parameters

The following table shows the parameters of the instruction "FETCH_RK":

Parameters	Declaration	Data type	Range of values	Default	Description
SYNC_DB	Input	INT	CPU-specific	0	Number of the DB where the common data for the synchronization of the RK instructions are stored (minimum length = 240 bytes).
REQ	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Request: Activates the job on a positive edge.
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset: Job is aborted.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The I/O address of your sub-module that you specified during hardware configuration.
R_CPU	Input	INT	0 to 4	1	CPU number of the partner CPU (only for multiprocessor operation)
R_TYPE	Input	CHAR	'D', 'X', 'M', 'I', 'Q', 'C', 'T'	'D'	Address type on the partner CPU • 'D': Data block • 'X': Expanded data block • 'M': Bit memory • 'I': Inputs • 'Q': Outputs • 'C': Counters • 'T': Timers
R_DBNO	Input	INT	0 to 255	0	Data block number on the partner CPU
R_OFFSET	Input	INT	See table: "Parameters for data source (partner CPU)"	0	Data byte number on the partner CPU
R_CF_BYT	Input	INT	0 to 255	255	Communication flag byte on partner CPU (255: Means: without interprocessor communication flag)
R_CF_BIT	Input	INT	0 to 7	0	Communication flag bit on the partner CPU
DONE	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): • FALSE: Job not yet started or still running. • TRUE: Job was executed error-free.
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with errors

Parameters	Declaration	Data type	Range of values	Default	Description
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR=TRUE: An error has occurred, STATUS supplies detailed information on the type of error.
RD_1	Input	ANY	CPU-specific	0	Receive parameter: Here you specify: • The number of the DB in which the retrieved data are to be stored. • The data byte number starting at which the retrieved data are to be stored. Possible values: 0 to 8190 For example: DB 10 from byte 2 -> DB10.DBB2 Note: Note that for S7-300 CPUs, the parameter RD_1 always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
LEN	Input	INT	1 to 1024	1	Here you specify the length of the data block that is to be retrieved (length is set here indirectly). A length of two bytes must be declared for each timer and counter.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameters for data source (partner CPU)

The data types that can be transmitted are shown in the table below.
The value R_OFFSET is specified by the partner CPU.

Source on the partner CPU	R_TYPE	R_NO	R_OFFSET (in bytes)
Data block	'D'	0 - 255	0 - 510, only even values
Expanded data block	'X'	0 - 255	0 - 510, only even values
Bit memory	’M’	irrelevant	0 - 255
Inputs	'I'	irrelevant	0 - 255
Outputs	’Q’	irrelevant	0 - 255
Counters	'C'	irrelevant	0 - 255
Timers	'T'	irrelevant	0 - 255

Data in the message frame header

The table below shows the data in the message frame header of the RK 512 message frame.

Source on the partner CPU	To the destination, your S7 automation system (local CPU)	Message frame header, bytes		
		3/4 command type	5/6 S-DBNO[1] / S offset[2]	7/8 number in
Data block	Data block	ED	DB/DW	Words
Expanded data block	Data block	EX	DB/DW	Words
Bit memory	Data block	EM	Byte address	Bytes
Inputs	Data block	EE	Byte address	Bytes
Outputs	Data block	IO	Byte address	Bytes
Counters	Data block	EC	Counter number	Words
Timers	Data block	ET	Time number	Words

[1] S-DBNO: Source Data Block Number
[2] S offset: Source start address

Data consistency

Data consistency is limited to 128 bytes. If you want consistent data transmission exceeding 128 bytes, you must take the following into account:

Do not write to the currently used section of the receive range RD_1 again until the transmission process has concluded. This is the case when the status parameter DONE assumes the value TRUE.

SERVE_RK: Receive and provide data (RK 512)

Description

You use the SERVE_RK instruction to receive and provide data.

- Receiving data: The data are stored in the area that is specified by the partner in the RK 512 message frame header. A call of the instruction is required when the communication partner executes a "Send data" job (SEND job).
- Serving data: The data are retrieved from the area that is specified by the partner in the RK 512 message frame header. A call of the instruction is required when the communication partner executes a "Fetch data" job (FETCH job).

After the instruction has been called with the value TRUE at control input EN_R, it is ready to fetch data. You cancel the current transmission by setting the signal status of parameter EN_R to FALSE. A canceled job is terminated with an error message (STATUS output). The input is switched off as long as the signal status of parameter EN_R is set to FALSE.

With the SYNC_DB parameter, you specify the DB where the data that are common to all instructions used for initialization during startup and synchronization will be stored. The DB number must be identical for all instructions used in your user program.

For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at control input R, the current transmission will be canceled and the instruction reset to its initial state. A canceled job is terminated with an error message (STATUS output).

Use LADDR to specify the submodule I/O address, which you have defined in HW Config.

NDR will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.

With NDR = TRUE for an instruction call, the CPU indicates in the parameters L_TYPE, L_DBNO and L_OFFSET the area where data were stored or from where they were retrieved. Also shown are the parameters L_CF_BYT and L_CF_BIT of a call as well as the length LEN of the respective job.

In STATUS, the CPU indicates the respective event ID as the result of an error or warning.

NDR or ERROR/STATUS are also output on RESET of the instruction (R = TRUE) (parameter LEN == 16#00).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the block was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect.

Instance DB

The SERVE_RK instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Using interprocessor communication flags

You can lock or enable SEND and FETCH jobs of your communication partner via an interprocessor communication flag. Thus, you can prevent overwriting or reading of data that have not yet been processed.

You can specify an interprocessor communication flag for every job.

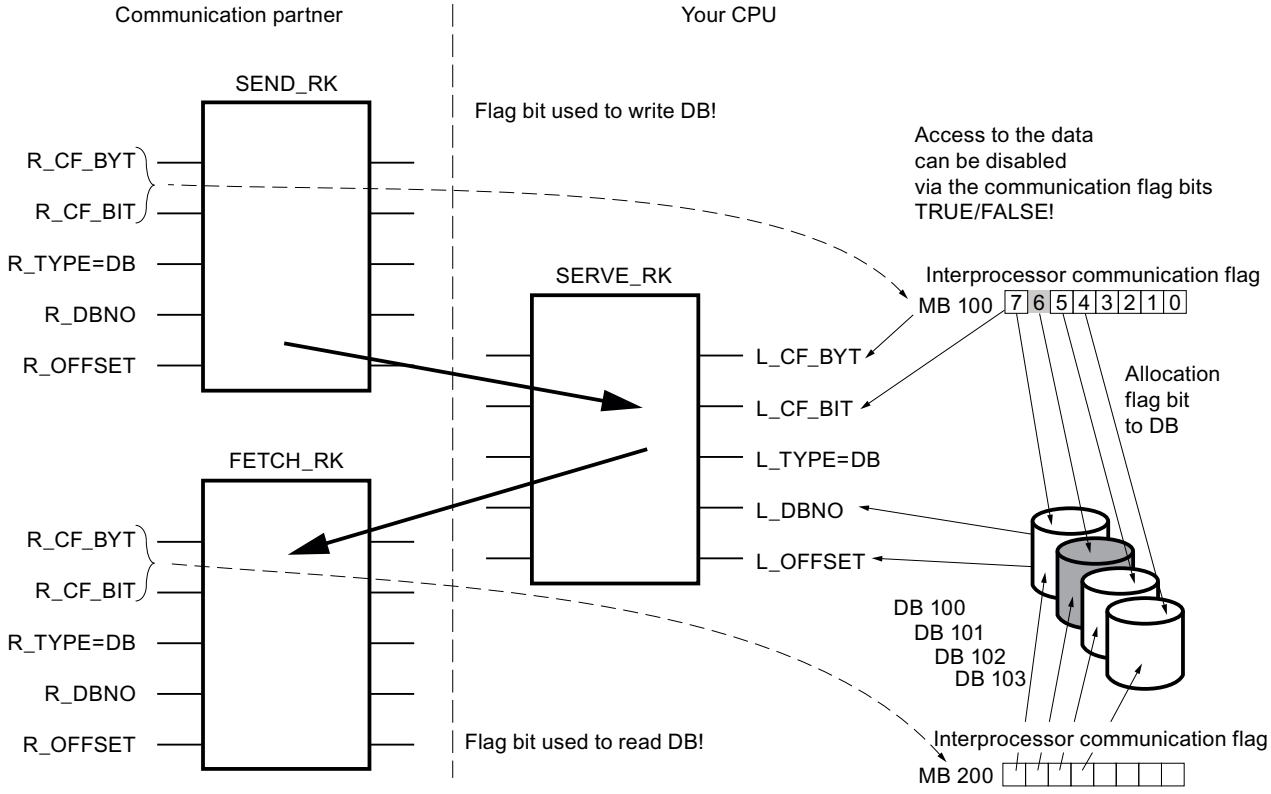

Example: SEND_RK with interprocessor communication flag:

In this example the communication partner transmits data to DB 101 on your CPU

1. On your CPU, set the interprocessor communication flag 100.6 to FALSE.
2. On your communication partner, specify interprocessor communication flag 100.6 (parameters R_CF_BYT, R_CF_BIT) for the SEND job.
   The interprocessor communication flag is transferred to your CPU in the RK 512 message frame header.
   Before it processes the job, the CPU verifies the interprocessor communication flag that is specified in the RK 512 message frame header. The job is only processed if the interprocessor communication flag is set to FALSE on your CPU. If the interprocessor communication flag is set to TRUE, then the error message "32 hex" will be transmitted to the communication partner in the response message frame.
   After the data have been transferred to DB 101, interprocessor communication flag 100.6 will be set to TRUE on your CPU by SERVE. Also, the communication flag byte and bit on SERVE are output for the duration of one call (if NDR = TRUE).

3. When you evaluate the interprocessor communication flag (interprocessor communication flag 100.6 = TRUE) in your user program you can see whether the job is complete and the transmitted data can be processed.
4. After you have processed the data in your user program you must reset the interprocessor communication flag 100.6 to FALSE. Your communication partner can only execute the job again without error when this has been done.

Parameter

The following table shows the parameters of the instruction "SERVE_RK":

Parameter	Declaration	Data type	Range of values	Default	Description
SYNC_DB	Input	INT	CPU-specific	0	Number of the DB where the common data for the synchronization of the RK instructions are stored (minimum length = 240 bytes).
EN_R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Enable to receive: Job enable
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset: Job is aborted.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The submodule I/O address, which you have specified during hardware configuration.
NDR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter New Data Ready (this parameter is only set for the duration of one call): • FALSE: Job not yet started or still running. • TRUE: Job was executed error-free.
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with errors
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR=TRUE: An error has occurred, STATUS supplies detailed information on the type of error.

Parameter	Declaration	Data type	Range of values	Default	Description
L_TYPE	Output	CHAR	'D' 'D', 'M', 'I', 'Q', 'C', 'T',	' '	**Receiving data:** Type of the destination area on the local CPU (only upper case allowed): • 'D': Data block **Serving data:** Type of the source area on the local CPU (only uppercase allowed): • 'D': Data block • 'M': Bit memory • 'I': Inputs • 'Q': Outputs • 'C': Counters • 'T': Timers This parameter is only set for the duration of one call.
L_DBNO	Output	INT	CPU-specific	0	Data block number on local CPU. This parameter is only set for the duration of one call.
L_OFFSET	Output	INT	0 - 510	0	Data byte number on local CPU. This parameter is only set for the duration of one call.
L_CF_BYT	Output	INT	0 to 255	0	Communication flag byte on local CPU. This parameter is only set for the duration of one call. (255: Means: without interprocessor communication flag)
L_CF_BIT	Output	INT	0 to 7	0	interprocessor communication flag on local CPU. This parameter is only set for the duration of one call.
LEN	InOut	INT	0 to 1024	0	Length of message frame, number in bytes (this parameter is only set for the duration of one call).

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Data consistency

Data consistency is limited to 128 bytes. If you want consistent data transmission exceeding 128 bytes, you must take the following into account:

Use the connection memory function. Do not access the data again until they have been completely transmitted (evaluation of the interprocessor communication flag specified for this job; the interprocessor communication flag is active for one instruction call if NDR = TRUE). Do not reset the interprocessor communication flag to FALSE until you have processed the data.

4.2.5.8 Communication with iSlave / iDevice

I_GET: Read data from a communication partner within the local S7 station

Description

With the "I_GET" instruction, you can read data from a communication partner that is inside the local S7 station. The communication partner can be located in the central controller, an expansion unit, or be part of the distributed configuration. Please make sure that you have assigned distributed communication partners to the local CPU. There is no corresponding instruction on the communication partner.

Receiving is activated by calling the instruction with REQ=1. Then call the instruction until receipt of data is indicated with BUSY=0. RET_VAL Then contains the length of the received block of data in bytes.

Make sure that the receive area defined with the RD parameter (on the receiving CPU) is at least as long as the read area defined by the VAR_ADDR parameter (on the communication partner). The data types of RD and VAR_ADDR must also match.

Parameter

The following table shows the parameters of the "I_GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: AUTOHOTSPOT
CONT	Input	BOOL	I, Q, M, D, L	Control parameter "continue" See also: AUTOHOTSPOT
IOID	Input	BYTE	I, Q, M, D, L, P or constant	Identifier of the address area on the partner module: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L, P or constant	Logical address of the partner module For a mixed module, the lower of the two addresses must be specified.
VAR_ADDR	Input	ANY	I, Q, M, D	Reference to the area on the partner CPU from which the data will be read. Choose a data type that is supported by the communication partner.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code. If no error occurs, RET_VAL contains the length in bytes of the block of data copied to the receive area RD as a positive number.

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY=1: Receiving is not yet complete. • BUSY=0: Receiving is complete or there is no receiving process active.
RD	Output	ANY	I, Q, M, D	Reference to the receive area (receive data area). The following data types are permitted: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME as well as arrays of the data types listed with the exception of BOOL. The receive area RD must be at least as long as the read area VAR_ADDR on the communication partner. The data types for RD and VAR_ADDR must also match. The maximum size of the receive area is 76 bytes.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Change to STOP mode

If the CPU changes to STOP mode, the connection established by "I_GET" is terminated. Whether the received data located in a buffer of the operating system are lost depends on the type of subsequent restart performed:

- Following a hot restart (not on the S7-300 and the S7-400H), the data is copied to the area defined by RD.
- Following a warm or cold restart, the data are discarded.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the data transfer with "I_GET": The data will also be read in STOP mode.

Data consistency

The data are received in a consistent state.

Parameter RET_VAL

See also:

- AUTOHOTSPOT
- Evaluating errors with output parameter RET_VAL (Page 162)

See also

Difference between synchronous and asynchronous instructions (Page 4186)

I_PUT: Write data to a communication partner within the local S7 station

Description

With the "I_PUT" instruction, you write data to a communication partner that is inside the local S7 station. The communication partner can be located in the central controller, an expansion unit, or be part of the distributed configuration. Please make sure that you have assigned distributed communication partners to the local CPU. There is no corresponding instruction on the communication partner.

The send operation is started after the instruction is called with signal level 1 at the REQ control input.

Make sure that the send area defined with the SD parameter (on the sending CPU) is the same length as the receive area defined by the VAR_ADDR parameter (on the communication partner). The data types of SD and VAR_ADDR must also match.

Parameter

The following table shows the parameters of the "I_PUT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: AUTOHOTSPOT
CONT	Input	BOOL	I, Q, M, D, L	Control parameter "continue" See also: AUTOHOTSPOT
IOID	Input	BYTE	I, Q, M, D, L, P or constant	Identifier of the address area on the partner module: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L, P or constant	Logical address of the partner module. For a mixed module, the lower of the two addresses must be specified.
VAR_ADDR	Input	ANY	I, Q, M, D, L	Reference to the area on the partner CPU to which the data will be written. Choose a data type that is supported by the communication partner.
SD	Input	ANY	I, Q, M, D	Reference to the area on the local CPU that contains the data to be sent. The following data types are permitted: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME as well as arrays of the data types listed with the exception of BOOL. SD must be the same length as the VAR_ADDR parameter of the communication partner. The data types for SD and VAR_ADDR must also match. The maximum size of the send area is 76 bytes.

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY=1: Sending is not yet complete. • BUSY=0: Sending is complete or there is no sending process active.

You can find additional information on valid data types under "Overview of the valid data types (Page 231)".

Change to STOP mode

If the CPU changes to STOP mode, the connection established by "I_PUT" is terminated. Data can no longer be sent. If the send data have already been copied to the internal buffer when the CPU changes mode, the contents of the buffer are discarded.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the data transfer with "I_PUT". The transmitted data will be written nevertheless.

Data consistency

The data are sent in a consistent state.

Parameter RET_VAL

See also:

- AUTOHOTSPOT
- Evaluating errors with output parameter RET_VAL (Page 162)

See also

Difference between synchronous and asynchronous instructions (Page 4186)

I_ABORT: Abort an existing connection to a communication partner within the local S7 station

Description

With the "I_ABORT" instruction, you abort a connection that was established with an "I_GET (Page 6784)" or "I_PUT (Page 6786)" instruction to a communication partner that is within the local S7 station. If the job belonging to "I_GET (Page 6784)" or "I_PUT (Page 6786)" is completed (BUSY = 0), the connection resources used at both ends are released after "I_ABORT" is called.

If the job belonging to "I_GET (Page 6784)" or "I_PUT (Page 6786)" is not yet completed (BUSY = 1), you will need to call the relevant instruction again with REQ = 0 and CONT = 0 after the connection has been aborted and then wait for BUSY = 0. Only then are all the connection resources released again.

You can only call "I_ABORT" at the end that is executing "I_PUT (Page 6786)" or "I_GET (Page 6784)" (in other words, at the client end).

The connection abort is activated by calling the instruction with REQ=1.

Parameters

The following table shows the parameters of the "I_ABORT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: AUTOHOTSPOT
IOID	Input	BYTE	I, Q, M, D, L, P or constant	Identifier of the address area on the partner module: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L, P or constant	Logical address of the partner module. For a mixed module, the lower of the two addresses must be specified.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The connection abort is not yet completed. • BUSY = 0: The connection abort is completed.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Change to STOP mode

If the CPU changes to STOP mode, the connection abort started with "I_ABORT" will be completed.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the connection abort with "I_ABORT". The connection is aborted.

RET_VAL parameter

See also:

- AUTOHOTSPOT
- Evaluating errors with output parameter RET_VAL (Page 162)

See also

Difference between synchronous and asynchronous instructions (Page 4186)

4.2.5.9 PROFINET CBA

Information regarding PN_IN, PN_OUT, and PN_DP

Note

By default, the operating system updates both the PROFINET CBA component as well as the DP interconnections at the cycle control point. If, however, you have disabled these automatic updates in the configuration (for example to improve the time response of the CPU), you will have to perform the updates yourself. This is done by calling the "PN_IN (Page 6791)", "PN_OUT (Page 6792)" and "PN_DP (Page 6793)" instructions at appropriate times. These updates can only be disabled as a group and not individually in the configuration.

Shadow memory

The interface DB is the user program interface for the PROFINET CBA component. To ensure that the inputs and outputs are consistent when the associated program sections are running, each interface DB has an identically structured memory area that is managed by the operating system. This area is called "shadow memory". In your program, you can only access the interface DB, and other (external) PROFINET CBA components can only access the shadow memory. This arrangement prevents conflicts in accessing the inputs and outputs of the interface DB.

Updating the PROFINET CBA component

The data consistency provided by the use of the shadow memory means that updating the PROFINET CBA component takes place in the following two steps:

- The inputs in the shadow memory are copied to the inputs of the interface DB before your program for the PROFINET CBA component starts
- The outputs of the interface DB are copied to the outputs of the shadow memory after your program for the PROFINET CBA component has run

The PROFINET CBA component is updated on CPUs with an integrated PROFINET interface either by the operating system or by the PN_IN (Page 6791)" and "PN_OUT (Page 6792)" instructions (depending on how you configured the PROFINET CBA component when you

created it). The following schematic diagram illustrates the update by the "PN_IN (Page 6791)" and "PN_OUT (Page 6792)" instructions.

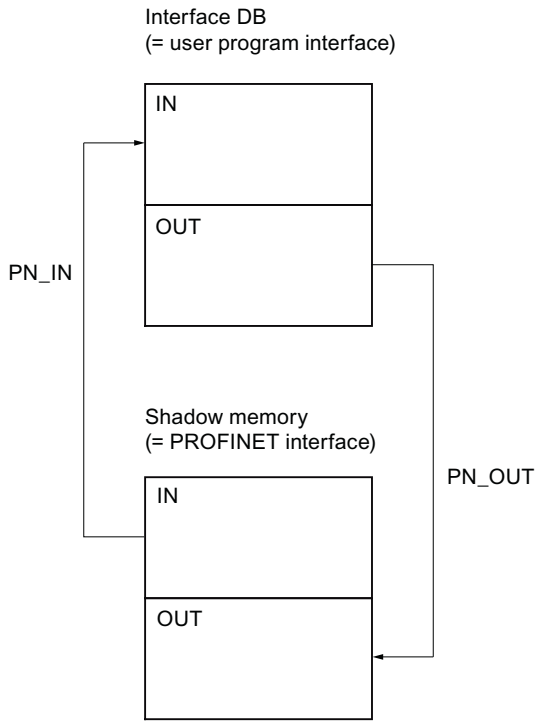

The system-side update always occurs at the cycle control point.

If, on the other hand, you perform the update using the "PN_IN (Page 6791)" and "PN_OUT (Page 6792)" instructions, you will need to call "PN_IN (Page 6791)" at the start of the OB containing the program associated with the PROFINET CBA component and "PN_OUT (Page 6792)" at the end of this OB. This procedure is shown using OB 30 as an example in the following Fig.

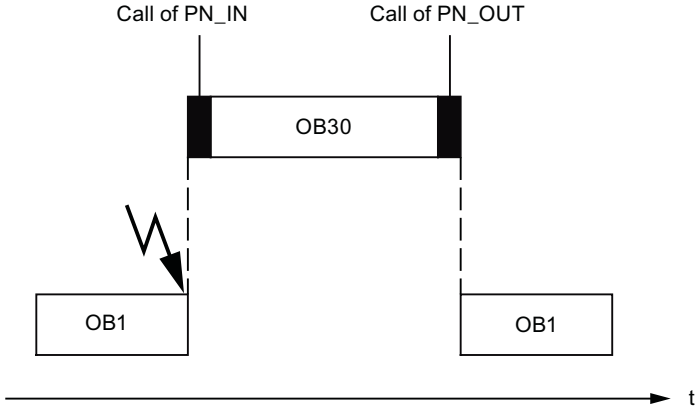

Note

(More than one PROFINET interface DB on a device):

With multifunction components, it is possible to update all interface DBs with one job (DBNO=0) – similar to the update at the cycle control point but triggered by an instruction. If an error occurs when updating an interface DB, the updating of the other interface DBs continues. A negative RET_VAL relates to one of the updated interface DBs.

Updating the DP interconnections

The DP interconnections are updated either by the operating system or by the "PN_DP (Page 6793)" instruction, depending on how you configured the PROFINET CBA component when you created it.

PN_IN: Update the inputs of the user program interface

Description

With the "PN_IN" instruction, you copy the input data in PROFINET CBA from the shadow memory of the PROFINET CBA component to the associated interface DB. After conclusion of the instruction, your application has the current input data available to it.

Parameters

The following table shows the parameters of the "PN_IN" instruction:

Parameters	Declaration	Data type	Memory area	Description
DBNO	Input	WORD	I, Q, M, D, L or constant	DB no. of the interface DB (DBNO=0: Updating all PROFINET CBA interface DBs)
RET_VAL	Return	INT	I, Q, M, D, L	Error information

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
8001	PROFINET CBA configuration is not available or invalid.
8002	The DB number does not match the one in the component configuration.
8004	The DB number matches the one in the component configuration, but the DB is not yet loaded.
8005	The interface DB was compiled with the keyword UNLINKED, in other words, it is only in the load memory but not in the work memory.

Error code (W#16#...)	Explanation
8006	The interface DB is write-protected in the CPU.
80B1	Length error when reading or writing. The component configuration does not fit the loaded DB.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

PN_OUT: Update the outputs of the user program interface

Description

With the instruction "PN_OUT", you copy the output data generated in your application from the interface DB of the PROFINET CBA component to the associated shadow memory. After conclusion of the instruction, the other PROFINET CBA components have the current output data available to them.

Parameters

The following table shows the parameters of the "PN_OUT" instruction:

Parameters	Declaration	Data type	Memory area	Description
DBNO	Input	WORD	I, Q, M, D, L or constant	DB no. of the interface DB (DBNO=0: Updating of all PROFINET interface DBs)
RET_VAL	Return	INT	I, Q, M, D, L	Error information

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
8001	PROFINET CBA configuration is not available or invalid.
8002	The DB number does not match the one in the component configuration.
8004	The DB number matches the one in the component configuration, but the DB is not yet loaded.
8005	The interface DB was compiled with the keyword UNLINKED, in other words, it is only in the load memory but not in the work memory.
8006	The interface DB is write-protected in the CPU.
80B1	Length error when reading or writing. The component configuration does not fit the loaded DB.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

PN_DP: Update DP interconnections

Description

With the "PN_DP" instruction you update all

- interconnections between PROFINET CBA components on the local PROFIBUS and
- interconnections with cyclic transmission between PROFINET CBA components on the local PROFIBUS and external PROFINET CBA components. The interconnections apply between networks (between Industrial Ethernet and PROFIBUS DP).

Functional description

"PN_DP" is an asynchronous instruction, its processing extends, if necessary, over multiple calls. To start updating the DP interconnections, call "PN_DP" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Difference between synchronous and asynchronous instructions (Page 4186).

Parameters

The following table shows the parameters of the "PN_DP" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Initiate update of the DP interconnections
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The updating of the DP interconnection is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	Job completed without error.
7000	First call with REQ = 0: Update of DP interconnections not initiated. BUSY has the value "0".
7001	(only relevant for S7-400:) First call with REQ = 1: BUSY has the value "1".
7002	(only relevant for S7-400:) Intermediate call (REQ irrelevant). The updating of the DP interconnections is not yet completed. BUSY has the value "1".
8001	PROFINET CBA configuration is not available or invalid.

Error code (W#16#...)	Explanation
8095	You have initiated another update of the DP interconnections in a higher priority class. However, the update in the priority class with lower priority (by the operating system or a "PN_DP" processing) is still running.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 162)

4.2.5.10 MPI communication

X_SEND: Send data to a communication partner outside the local S7 station

Description

With the "X_SEND" instruction, you send data to a communication partner outside the local S7 station. The data is received on the communication partner using the "X_RCV (Page 6795)" instruction. The send operation is started after the instruction is called with REQ=1.

Make sure that the send area defined by the SD parameter (on the sending CPU) is smaller than or the same size as the receive area defined by the RD parameter (on the communication partner). If SD is data type BOOL, RD must also be data type BOOL.

Parameters

The following table shows the parameters of the "X_SEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: AUTOHOTSPOT
CONT	Input	BOOL	I, Q, M, D, L	Control parameter "continue" See also: AUTOHOTSPOT
DEST_ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameter "destination ID". It contains the configured MPI address of the communication partner.
REQ_ID	Input	DWORD	I, Q, M, D, L or constant	Job identifier to identify the data on the communication partner.
SD	Input	ANY	I, Q, M, D	Reference to the send area. The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME and arrays of the specified data types with the exception of BOOL. The maximum size of the send area is 76 bytes.

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: Sending is not yet completed. • BUSY = 0: Sending is completed or there is no send operation active.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Data consistency

The data are sent in a consistent state.

Parameter RET_VAL

See also:

- AUTOHOTSPOT
- Evaluating errors with output parameter RET_VAL (Page 162)

See also

Difference between synchronous and asynchronous instructions (Page 4186)

X_RCV: Receive data from a communication partner outside the local S7 station

Description

With the "X_RCV" instruction, you receive the data sent by one or more communication partners outside the local S7 station using the "X_SEND (Page 6794)" instruction.

With the "X_RCV" instruction

- you can check whether data are available at the current time. The data were entered in an internal queue by the operating system.
- You can copy the oldest block of data from the queue to a selected receive area.

Parameters

The following table shows the parameters of the "X_RCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_DT	Input	BOOL	I, Q, M, D, L	Control parameter "enable data transfer". Use the value 0 to verify whether at least one data block is available. The value 1 causes the oldest data block in the queue to copied into the work memory area that you have specified with RD.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code. If no error occurs, RET_VAL contains • if EN_DT=0/1 and NDA = 0: W#16#7000. In this case, no block of data is in the queue. • If EN_DT=0 and NDA=1 the length of the oldest block of data entered in the queue as a positive number in bytes. • If EN_DT=1 and NDA=1 the length of the oldest block of data copied to the receive area RD as a positive number in bytes.
REQ_ID	Output	DWORD	I, Q, M, D, L	Job identifier of the "X_SEND (Page 6794)" instruction whose sent data is first in the queue, in other words, the oldest data in the queue. If there is no block of data in the queue, REQ_ID has the value "0".
NDA	Output	BOOL	I, Q, M, D, L	Status parameter "new data arrived". NDA=0: • There is no block of data in the queue. NDA=1: • The queue contains at least one block of data. ("X_RCV" call with EN_DT=0). • The oldest block of data in the queue was copied to the user program ("X_RCV" call with EN_DT=1).
RD	Output	ANY	I, Q, M, D	Reference to the receive area (receive data area). The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of the specified data types with the exception of BOOL. If you want to discard the oldest block of data in the queue, assign RD the value NIL. .The maximum size of the receive area is 76 bytes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Show data reception with EN_DT=0

As soon as data from a communication partner arrive, they are entered in the queue by the operating system in the order in which they are received.

If you want to check whether at least one block of data is in the queue, call "X_RCV" with EN_DT=0 and evaluate the output parameter NDA as follows:

- NDA=0 means that the queue does not contain a block of data. REQ_ID is irrelevant, RET_VAL contains W#16#7000.
- NDA=1 means that there is at least one block of data that can be fetched in the queue . In this case, you also evaluate the output parameters RET_VAL and possibly REQ_ID. RET_VAL contains the length of the data block in bytes, REQ_ID has the job ID of the send block. If there are several blocks of data in the queue, REQ_ID and RET_VAL belong to the oldest block of data in the queue.

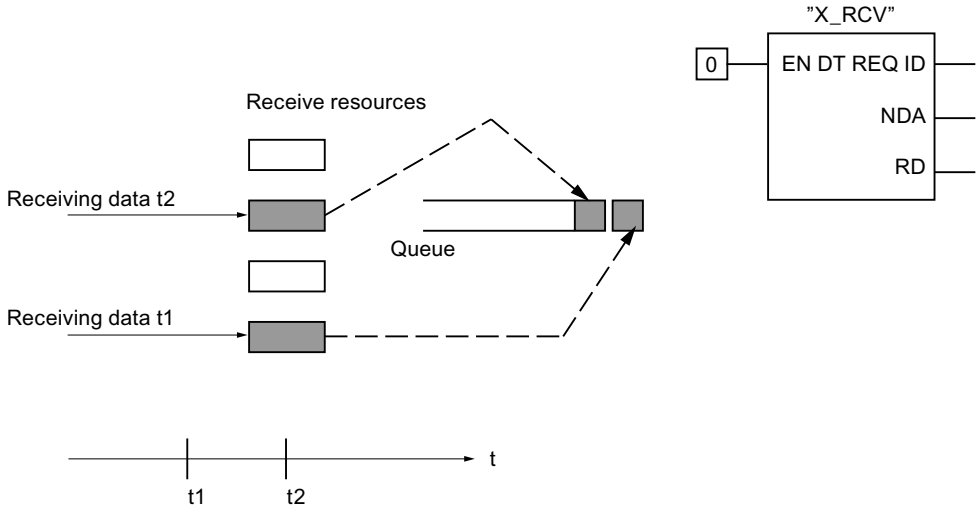

Accepting data in the receive area with EN_DT=1

If you call "X_RCV" with EN_DT=1 , the oldest available block of data in the queue is copied to the area of the work memory specified by RD. RD must be greater than or equal to the send area of the associated "X_SEND (Page 6794)" instruction that is defined by the SD parameter. If its input parameter SD is data type BOOL, RD must also be data type BOOL. If you want to store the received data in different areas, you can determine REQ_ID (instruction call with EN_DT=0) and select RD (with EN_DT=1) accordingly in the subsequent call. If an error occurred when copying, then the length of the copied block of data in bytes is in RET_VAL and a positive acknowledgment will be sent to the sender.

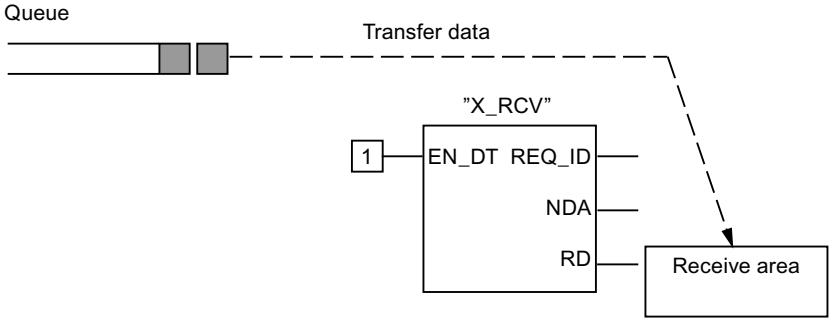

Discarding data

If you do not want to accept the data, assign the value NIL to RD. In this case, the sender receives a negative acknowledgment (RET_VAL of the corresponding "X_SEND (Page 6794)" instruction has the value W#16#80B8) and "0" is entered in RET_VAL of the "X_RCV" instruction.

Data consistency

After a call with EN_DT=1 and RETVAL=W#16#00xy, there is new data in the receive area RD. A further block call can overwrite this data. To prevent this, do not call "X_RCV" with the same receive area RD until you have evaluated the received data.

Change to STOP mode

If the CPU changes to STOP mode

- All newly arriving jobs are acknowledged negatively.
- Applies for the jobs that have already arrived: All jobs that have arrived and are in the queue are acknowledged negatively.
  - Following a warm or cold restart, the data are discarded.
  - If there is a subsequent hot restart (not on an S7-300 or S7-400H), the data block belonging to the oldest job is entered in the user program, if it was queried before the transition to STOP mode (by calling "X_RCV" with EN_DT=0). Otherwise it is discarded. All other blocks of data are discarded.

Connection abort

If the connection is terminated a job belonging to the connection that is already in the queue is discarded.

Exception: If this job is the oldest in the queue and you have already detected its presence by calling X_RCV" with EN_DT=0, you can enter it in the receive area with EN_DT=1.

Parameter RET_VAL

See also:

- AUTOHOTSPOT
- Evaluating errors with output parameter RET_VAL (Page 162)

X_GET: Read data from a communication partner outside the local S7 station

Description

With the "X_GET" instruction, you can read data from a communication partner that is inside the local S7 station. There is no corresponding instruction on the communication partner.

The read operation is started after the instruction is called with REQ=1. Following this, you continue to call "X_GET" until the receipt of the data is indicated with BUSY=0. RET_VAL contains the length of the received block of data in bytes.

Make sure that the receive area defined with the RD parameter (on the receiving CPU) is at least as long as the read area defined by the VAR_ADDR parameter (on the communication partner). The data types of RD and VAR_ADDR must also match.

Parameters

The following table shows the parameters of the "X_GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: AUTOHOTSPOT
CONT	Input	BOOL	I, Q, M, D, L	Control parameter "continue" See also: AUTOHOTSPOT
DEST_ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameter "destination ID". It contains the MPI address of the communication partner. You have configured this.
VAR_ADDR	Input	ANY	I, Q, M, D	Reference to the area on the partner CPU from which the data will be read. Choose a data type that is supported by the communication partner.

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code. If no error occurs, RET_VAL contains the length in bytes of the block of data copied to the receive area RD as a positive number.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: Receiving is not yet completed. • BUSY = 0: Receiving is completed or there is no receive operation active.
RD	Output	ANY	I, Q, M, D	Reference to the receive area (receive data area). The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of the specified data types with the exception of BOOL. The receive area RD must be at least as long as the read area VAR_ADDR on the communication partner. The data types for RD and VAR_ADDR must also match. The maximum size of the receive area is 76 bytes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Change to STOP mode

If the CPU changes to STOP mode, the connection established by "X_GET" is terminated. Whether the received data located in a buffer of the operating system are lost depends on the type of subsequent restart performed:

- Following a hot restart (not on the S7-300 and the S7-400H), the data is copied to the area defined by RD.
- Following a warm or cold restart, the data are discarded.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the data transfer with "X_GET": The data will also be read in STOP mode.

Data consistency

The data are received in a consistent state.

Parameter RET_VAL

See also:

- AUTOHOTSPOT
- Evaluating errors with output parameter RET_VAL (Page 162)

See also

Difference between synchronous and asynchronous instructions (Page 4186)

X_PUT: Write data to a communication partner outside the local S7 station

Description

With the "X_PUT" instruction, you write data to a communication partner that is outside the local S7 station. There is no corresponding instruction on the communication partner.

The read operation is started after the instruction is called with REQ=1. Following this, you continue to call "X_PUT" until the receipt of the data is indicated with BUSY=0.

Make sure that the send area defined with the SD parameter (on the sending CPU) is the same length as the receive area defined by the VAR_ADDR parameter (on the communication partner). The data types of SD and VAR_ADDR must also match.

Parameters

The following table shows the parameters of the "X_PUT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: AUTOHOTSPOT
CONT	Input	BOOL	I, Q, M, D, L	Control parameter "continue" See also: AUTOHOTSPOT.
DEST_ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameter "destination ID". It contains the MPI address of the communication partner. You have configured this.
VAR_ADDR	Input	ANY	I, Q, M, D	Reference to the area on the partner CPU to which the data will be written. Choose a data type that is supported by the communication partner.
SD	Input	ANY	I, Q, M, D	Reference to the area on the local CPU that contains the data to be sent. The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME,S5_TIME, DATE_AND_TIME and arrays of the specified data types with the exception of BOOL. SD must be the same length as the VAR_ADDR parameter of the communication partner. The data types for SD and VAR_ADDR must also match. The maximum size of the send area is 76 bytes.

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: Sending is not yet completed. • BUSY = 0: Sending is completed or there is no send operation active.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Change to STOP mode

If the CPU changes to STOP mode, the connection established by "X_PUT" is terminated. Data can no longer be sent. If the send data have already been copied to the internal buffer when the CPU changes mode, the contents of the buffer are discarded.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the data transfer with "X_PUT": The transmitted data will be written nevertheless.

Data consistency

The data are sent in a consistent state.

Parameter RET_VAL

See also:

- AUTOHOTSPOT
- Evaluating errors with output parameter RET_VAL (Page 162)

See also

Difference between synchronous and asynchronous instructions (Page 4186)

X_ABORT: Abort an existing connection to a communication partner outside the local S7 station

Description

With the "X_ABORT" instruction, you abort a connection that was established with an "X_SEND (Page 6794)", "X_GET (Page 6799)" or "X_PUT (Page 6801)" instruction to a communication partner that is outside the local S7 station.

- If the job belonging to "X_SEND (Page 6794)", "X_GET (Page 6799)" or "X_PUT (Page 6801)" is completed (BUSY=0), the connection resources used at both ends are released again after the "X_ABORT" instruction is called.
- If the job belonging to "X_SEND (Page 6794)", "X_GET (Page 6799)" or "X_PUT (Page 6801)" is not yet completed (BUSY=1), you will need to call the relevant instruction again with REQ=0 and CONT=0 after the connection has aborted and then wait for BUSY=0. Only then are all the connection resources released again.

You can only call "X_ABORT" at the end that is executing "X_SEND (Page 6794)", "X_PUT (Page 6801)" or "X_GET (Page 6799)". The connection abort is activated by calling the instruction with REQ=1.

Parameters

The following table shows the parameters of the "X_ABORT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: AUTOHOTSPOT
DEST_ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameter "destination ID". It contains the MPI address of the communication partner. You have configured this.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The connection abort is not yet completed. • BUSY = 0: The connection abort is completed.

For additional information on valid data types, refer to "Overview of the valid data types (Page 231)".

Change to STOP mode

If the CPU changes to STOP mode, the connection abort started with "X_ABORT" will be completed.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the connection abort with "X_ABORT". The connection is aborted.

Parameter RET_VAL

See also:

- AUTOHOTSPOT
- Evaluating errors with output parameter RET_VAL (Page 162)

See also

Difference between synchronous and asynchronous instructions (Page 4186)

4.2.5.11 TeleService

PG_DIAL: Establish remote connection to programming device/PC

Description of PG_DIAL

Description

The "PG_DIAL" instruction transfers a telephone number and an event ID to a TS Adapter. Using the specified telephone number, the TS Adapter establishes a remote connection to a programming device/PC. The event ID is transferred to the programming device/PC and passed to a waiting application.

If the event ID (EVENT_ID (Page 6806) parameter) was successfully transferred to the application, the TS Adapter receives an acknowledgement that is passed to the "PG_DIAL" instruction. The processing of "PG_DIAL" then is terminated and the status transferred to the caller of "PG_DIAL". The application on the programming device/PC is responsible for terminating the remote connection.

If an error occurs during the processing that leads to the processing aborting, the error code is transferred to the caller of "PG_DIAL". Any established remote connection will be terminated by the TS Adapter.

Calling "PG_DIAL"

You can call the "PG_DIAL" instruction in the cycle or a time-controlled program. An instance DB must be specified when "PG_DIAL" is called.

"PG_DIAL" must be called more than once to process a block function. This means that calling "PG_DIAL" in a "queue" is pointless. The end of the block function is indicated by BUSY = 0.

Communication connection abort

If the CPU changes to STOP mode while "PG_DIAL" is active, the communication connection to the TS Adapter is aborted.

The communication connection to the TS Adapter is also lost if serious communication problems arise on the MPI bus or, for example, the CPU is switched off. In these cases the telephone number and event ID that were already transferred to the TS Adapter and not

discarded by the adapter. The TS Adapter establishes the remote connection to the PG/PC and transfers the event ID. However, the acknowledgement that is transferred from the programming device/PC to the TS Adapter, is discarded by the TS Adapter.

If a user program attempts to establish a remote connection while the TS Adapter is in the status described above, "PG_DIAL" is terminated with the return value W#16#B10A. The user program can later repeat the establishment of a remote connection. You also receive the return value W#16#B10A, if various CPUs attempt to establish a remote connection at the same time via the same TS Adapter.

NOTICE
Changing user programs You can change the parts of your user program that directly affect calls of "PG_DIAL" only when the CPU is in STOP mode. This relates, in particular, to deleting and replacing program blocks that contain "PG_DIAL" calls. Ignoring this restriction can tie up connection resources. The status of the automation system relating to the communication instructions for non-configured S7 connections may become undefined. A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The input parameters of the function block are copied to in internal buffer when "PG_DIAL" is first called. Do not change this data before the first call is completed (return value W#16#7001), otherwise inconsistent data could be transferred.

Parameters

The following table shows the parameters of the "PG_DIAL" instruction:

Parameters	Declaration	Data type	Memory area	Description
MPI_TS_ADAPTER	Input	INT	I, Q, M, D, L or constant	MPI address of the TS Adapter
PHONE_NO (Page 6806)	Input	ANY	D	Reference to a data string with a maximum length of 31 characters
EVENT_ID (Page 6806)	Input	ANY	I, Q, M, D	Reference to a byte array with a maximum length of 16 characters
BUSY (Page 6807)	Output	BOOL	I, Q, M, D, L	• BUSY=1: The establishment of the remote connection is not yet completed. • BUSY=0: The processing of "PG_DIAL" was stopped.
STATUS (Page 6807)	Output	INT	I, Q, M, D, L	Return value of "PG_DIAL"

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

Virtual block parameters EN and ENO and the BR bit

The virtual block parameters EN and ENO occur only when "PG_DIAL" is included in the LAD or FBD representation of the STEP 7 editor. They are closely linked with the BR bit (binary result) of the status word.

- Input parameters EN
  The status of the result of logic operation (RLO) is saved to the BR bit during the block call.
- Output parameters ENO
  If "PG_DIAL" is processed error-free, the status of the BR bit that existed when the block was called is restored again at the end of the block call.
  If an error message is output via the output parameter STATUS, the BR bit is set to "0" immediately before exiting "PG_DIAL".

PHONE_NO parameter

Description

The input parameter PHONE_NO specifies the telephone number to which a remote connection will be established. Enter the complete telephone number including country code, area code and subscriber number. The string is transferred unchanged to the modem. If you use characters that are not numerals, make sure that the modem used supports these characters.

The TS Adapter uses the following parameter values to establish a remote connection:

- Location: Dial procedure, access code
- Call settings: "Wait for dial tone before dialing", "Number of redial attempts" and "Redial after"

EVENT_ID parameter

Description

The EVENT_ID input parameter specifies the event ID. This is transferred transparently from the user program of the automation system to the user program on the PG/PC by "PG_DIAL", the TS Adapter and the TeleService application on the PG/PC. You can structure the event ID as required and transfer any information from the automation system to the PG/PC.

If a length less than 16 is transferred when "PG_DIAL" is called, the remaining bytes of the array are padded with B#16#00.

BUSY parameter

Description

"PG_DIAL" is an asynchronous instruction, in other words, its processing extends over multiple calls.

- If the BUSY output parameter = 1, the internal status of "PG_DIAL" is output in the output parameter STATUS (Page 6807).
- When processing is completed, this is indicated by the output parameter BUSY = 0. The STATUS (Page 6807) output parameter then indicates whether the job was completed error-free (STATUS = W#16#0000) or with errors.

STATUS parameter

Description

The return values of "PG_DIAL" can be classified as follows:

- W#16#0000: "PG_DIAL" was completed successfully
- W#16#7xxx: Status of "PG_DIAL"
- W#16#8xxx: An error was reported during the internal call of a communication instruction or the "BLKMOV" instruction
- W#16#9xxx: Parameter error when calling "PG_DIAL"
- W#16#Bxxx: An error was reported by the TS Adapter

The following tables shows the return values of "PG_DIAL" except for error codes of the communication instructions used.

Return value (W#16#...)	Explanation	Notes
0000	The processing of "PG_DIAL" was completed without errors.	
7000	"PG_DIAL" was reset (the communication with the TS Adapter was terminated in the meantime).	Call "PG_DIAL" again.
7001	"PG_DIAL" is active (first call, BUSY = 1). The function has just been started.	
7002	"PG_DIAL" is active (subsequent call, BUSY = 1). The processing of the function is not yet completed.	
8xxx or 8zxx	The processing of "PG_DIAL" was completed with an error code of the internally called communication instructions. If the error message originates from the "BLKMOV" instruction, the meaning is as follows • z = 2: Error copying the PHONE_NO parameter to the internal buffer • z = 3: Error copying the EVENT_ID parameter to the internal buffer	You will find more detailed information in the error information of the communication instructions: AUTOHOTSPOT

Return value (W#16#...)	Explanation	Notes
9001	Length of PHONE_NO = 0 or > 31	The telephone number must consist of at least 1 character and not more than 31 characters.
9002	Length of EVENT_ID = 0 or > 16	The event ID must consist of at least 1 character and not more than 16 characters.
B000	It was possible to establish the remote connection to the PG/PC. With TeleService, however, no program logged on for the transferred event ID.	
B10A	A job is already stored on the TS Adapter and this job is currently being processed by the TS Adapter.	Call "PG_DIAL" again.
B10B	After the establishment of the remote connection, the connection was terminated by the partner before the event ID could be transferred.	Call "PG_DIAL" again.
B206	It was not possible to establish the remote connection to the PG/PC.	Check the parameter assignment of the modem in the TS Adapter (local and remote).
B302	The TS Adapter is directly connected to a PG/PC (direct connection).	Terminate the communication on the direct connection.
B305	A remote connection to a PG/PC is still established.	Call "PG_DIAL" again later.

AS_DIAL: Establish remote connection to AS

Description of AS_DIAL

Definition

Local and remote automation system

- Local: The S7 automation system from which the initiative to establish the remote connection originates is described as local.
- Remote: The S7 automation system to which the remote connection will be established is described as remote.

Description

With the "AS_DIAL" instruction, you can establish a remote connection from a local S7 automation system to a remote S7 automation system and then exchange process data. You can use the "X_GET (Page 6799)", "X_PUT (Page 6801)" and "X_SEND (Page 6794)" instructions to exchange process data.

The "AS_DIAL" instruction has two functions:

- **Function DIAL:** Establishing a remote connection to a remote TS Adapter. You use the REQ_DIAL (Page 6811) input parameter to request this function.
- **Function HANGUP:** Terminating an existing remote connection to a remote TS Adapter. You use the REQ_HANGUP (Page 6811) input parameter to request this function.

Only one of the functions can be active at one time. The DIAL function is terminated by the HANGUP function request.

If an error occurs during the processing that leads to the processing aborting, the error code is transferred to the caller of "AS_DIAL". Any established remote connection will be terminated by the TS Adapter.

Calling "AS_DIAL"

You can call "AS_DIAL" in the cycle or in a time-controlled program. If you call "AS_DIAL" in different tasks, make sure that the calls are interlocked.

- An instance DB must be specified when "AS_DIAL" is called. Always use the same instance for a remote connection.
  Exception: If the local CPU communicates via multiple local TS Adapters, assign a separate instance to each TS Adapter.
- "AS_DIAL" must be called more than once to process a block function. This means that calling "AS_DIAL" in a "queue" is pointless. The end of the block function is indicated by BUSY = 0.
- To communicate with the TS Adapter, "AS_DIAL" also uses communication instructions for non-configured S7 connections.
  Therefore, make sure that no communication instructions for non-configured S7 connections ("X_PUT (Page 6801)", "X_GET (Page 6799)", "X_SEND (Page 6794)", "X_RCV (Page 6795)", "X_ABORT (Page 6803)") that were started with DEST_ID = "MPI address of the local TS Adapter" are active when the remote connection is being established or terminated by "AS_DIAL".

Parameters

The following table shows the parameters of the "AS_DIAL" instruction:

Parameters	Declaration	Data type	Memory area	Description
ADDR_TS_ ADAPTER	Input	INT	I, Q, M, D, L or constant	Input parameters of the DIAL and HANGUP functions: MPI address of the TS Adapter on the local MPI network
PHONE_NO (Page 6810)	Input	ANY	D	Input parameters of the DIAL function: Telephone number of the modem of the remote automation system (max. 31 characters)
LOGIN	Input	ANY	D	Input parameters of the DIAL function: User name for authorization on the remote TS Adapter (max. 8 characters)
PASSWORD	Input	ANY	D	Input parameters of the DIAL function: Password for authorization on the remote TS Adapter (max. 8 characters)
ADDR_CPU	Input	INT	I, Q, M, D, L or constant	Input parameters of the DIAL function: MPI address of the CPU on the remote MPI network to which the connection is to be established.

Parameters	Declaration	Data type	Memory area	Description
REQ_DIAL (Page 6811)	Input	BOOL	I, Q, M, D	Input parameters for requesting the DIAL function: Establishing a remote connection to a remote TS Adapter
REQ_HANGUP (Page 6811)	Input	BOOL	I, Q, M, D	Input parameters for requesting the HANG-UP function: Terminating an existing remote connection to a remote TS Adapter
STATUS (Page 6812)	Output	INT	I, Q, M, D, L	Return value of "AS_DIAL"
BUSY (Page 6812)	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The DIAL or HANGUP function is still active; in other words, establishment/termination of the remote connection is not yet completed. • BUSY=0: Processing of the "AS_DIAL" instruction is completed. No function is still active.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

Virtual block parameters EN and ENO and the BR bit

The virtual block parameters EN and ENO occur only with the integration of "AS_DIAL" in the LAD or FBD representation.

- Input parameter EN: The status of the result of logic operation (RLO) is saved to the BR bit during the block call.
- Output parameter ENO:
  If "AS_DIAL" is processed error-free, the status of the BR bit that existed when the block was called is restored again at the end of the block call.
  If an error message is output via the STATUS (Page 6812) output parameter, the BR bit is set to "0" immediately before exiting "AS_DIAL".

PHONE_NO parameter

Description

The input parameter PHONE_NO specifies the telephone number to which a remote connection will be established. Enter the complete telephone number including country code, area code and subscriber number. The string is transferred unchanged to the modem. If you use characters that are not numerals, make sure that the modem used supports these characters.

The TS Adapter uses the following parameter values to establish a remote connection:

- Location: Dial procedure, access code
- Call settings: "Wait for dial tone before dialing", "Number of redial attempts" and "Redial after"

Parameter REQ_DIAL

Description

The DIAL function can only be requested if BUSY (Page 6812) = 0 was reported at the last of "AS_DIAL" call. With the REQ_DIAL input parameter, you start the DIAL function. The REQ_DIAL input parameter is level-triggered.

REQ_DIAL triggers the DIAL function of the "AS_DIAL" instruction. The "AS_DIAL" instruction checks the parameters for plausibility and transfers these to the local TS Adapter.

If an error is detected during the processing of the DIAL function, this causes processing of the DIAL function to be aborted. "AS_DIAL" automatically executes the HANGUP (Page 6811) function and then outputs the error message in the STATUS (Page 6812) return value.

Result: With BUSY (Page 6812) = 0, the result of the connection establishment is displayed in the STATUS (Page 6812) return value.

REQ_HANGUP parameter

Description

With the REQ_HANGUP input parameter, you start the HANGUP function. The HANGUP function terminates a remote connection to the remote TS Adapter, which was established earlier by the DIAL (Page 6811) function.

The REQ_HANGUP input parameter is level-triggered. The HANGUP function can also be started if the DIAL (Page 6811) function was already started.

The end of the HANGUP function is notified by the output parameter BUSY (Page 6812) = 0. The STATUS (Page 6812) output parameter then supplies the result of the function processing.

Value	Meaning
W#16#0000	The remote connection was terminated normally by the HANGUP function.
W#16#3007	The remote connection was terminated by the local TS Adapter because an error occurred.
W#16#3008	The remote connection has broken down spontaneously or the remote TS Adapter has hung up.

BUSY parameter

Description

"AS_DIAL" is an asynchronous instruction, in other words, its processing extends over multiple calls.

- If BUSY = 1, the internal STATUS (Page 6812) of "AS_DIAL" is output in the STATUS (Page 6812) output parameter. At least one further "AS_DIAL" call is necessary.
- If BUSY = 0, the processing is completed. The STATUS (Page 6812) output parameter then indicates whether the job was completed error-free or with errors.

STATUS parameter

Description

The return values of "AS_DIAL" can be classified as follows:

- W#16#0000: "AS_DIAL" was completed without errors
- W#16#3xxx: Most recent status of remote connection in the TS Adapter
- W#16#7xxx: Status of "AS_DIAL"
- W#16#8xxx: An error was reported during the internal call of a communication instruction ("X_ABORT (Page 6803)", "X_GET (Page 6799)", "X_PUT (Page 6801)", "X_RCV (Page 6795)" and "X_SEND (Page 6794)").
- W#16#9xxx: Parameter error when calling "AS_DIAL"
- W#16#Bxxx: Error during connection establishment

The following table shows the return values of "AS_DIAL" except for error codes of the communication instructions used:

Return value (W#16#...)	Explanation	Notes
0000	No function is active. The processing of "AS_DIAL" was completed without errors.	This message is only displayed once after successful completion of the relevant function.
3007	The local TS Adapter has hung up.	Is only displayed by the HANGUP function.
3008	The remote connection has broken down spontaneously or the remote TS Adapter has hung up.	Is only displayed by the HANGUP function.
7000	No function is active.	No function was processed or requested via the input parameters.
7001	DIAL function is active (first call, BUSY = 1). The function has just been started.	The REQ_DIAL input parameter is not evaluated until the end of the (BUSY = 0) function.
7002	DIAL function is active (subsequent call, BUSY = 1). The processing of the function is not yet completed.	The REQ_DIAL input parameter is not evaluated until the end of the (BUSY = 0) function.
7701	The HANGUP function is active (first call). The function has just been started.	The input parameters REQ_DIAL and REQ_HANGUP are not evaluated until the end of the (BUSY = 0) function.

Return value (W#16#...)	Explanation	Notes
7702	HANGUP function is active (subsequent call). The processing of the function is not yet completed.	The input parameters REQ_DIAL and REQ_HANGUP are not evaluated until the end of the (BUSY = 0) function.
8xxx	Error message: The processing of the active block function was terminated because of an error message (RET_VAL < 0) of an instruction called within the block.	You will find more detailed information in the error information of the communication instructions: AUTOHOTSPOT.
9001	Length of PHONE_NO = 0 or > 31	The telephone number must consist of at least 1 character and not more than 31 characters.
9003	Length of LOGIN > 8	The user name for authorization on the remote TS Adapter must not exceed 8 characters.
9004	Length of PASSWORD > 8	The password for authorization on the remote TS Adapter must not exceed 8 characters.
B100	TS Adapter cannot currently establish a connection.	The TS Adapter cannot currently call a connection to the modem. Call "AS_DIAL" again.
B10A	An event ID of a "PG_DIAL" function is already stored in the TS Adapter or an "AS_DIAL" function is active that is still to be transferred.	Call "AS_DIAL" again.
B10B	The remote connection was terminated again after successful establishment of the remote connection.	Call "AS_DIAL" again.
B206	It was not possible to establish the remote connection to the TS Adapter.	Check the parameter assignment of the modem in the TS Adapter (local and remote).
B20A	The remote TS Adapter type does not support AS-AS remote coupling.	Use a TS Adapter that supports the AS-AS remote coupling.
B20B	The "AS_DIAL" instruction and the version of the TS Adapter are incompatible.	Use a TS Adapter that supports the AS-AS remote coupling.
B20C	The remote TS Adapter cannot join the MPI network. Cause: Network parameter error	Check the parameter assignment of the network parameters in the remote TS Adapter.
B20D	The version of the remote TS Adapter does not support an AS-AS remote coupling.	Use a TS Adapter that supports the AS-AS remote coupling.
B20E	Data transfer to the remote TS Adapter was aborted. Cause: Transmission error	Call "AS_DIAL" again.
B20F	The remote TS Adapter has not called back.	Check the access protection parameter settings (callback number) and modem in the remote TS Adapter.
B210	The data transmission to the remote TS Adapter is disrupted. Cause: Timeout	Call "AS_DIAL" again.
B252	The remote TS Adapter rejects the job due to lack of authorization.	You have not specified a user name and password for the "AS_DIAL" call.
B253	The remote TS Adapter rejects the authorization. Cause: Unknown user name	Check the call parameters of "AS_DIAL" in your user program.
B254	The remote TS Adapter rejects the authorization. Cause: Incorrect password	Check the call parameters of "AS_DIAL" in your user program.

Return value (W#16#...)	Explanation	Notes
B302	The TS Adapter is directly connected to a PG/PC (direct connection).	Terminate the communication on the direct connection.
B305	A remote connection to a PG/PC is still established by the local TS Adapter.	Call "AS_DIAL" again later.

SMS_SEND: Send text (SMS) message

Description of SMS_SEND

Description

The "SMS_SEND" instruction transfers a telephone number, a service center number and an SMS message to a TS Adapter. The TS Adapter transfers this data via GSM commands to a wireless modem.

After the SMS message has been sent, the TS Adapter receives an acknowledgement that is forwarded to the "SMS_SEND" instruction. The processing of "SMS_SEND" is then terminated and the status communicated to the caller. This status is only an acknowledgement that the SMS has been sent, not a confirmation of receipt.

If an error occurs during the processing that leads to the processing aborting, the error code is transferred to the caller of "SMS_SEND".

Calling "SMS_SEND"

You can call the "SMS_SEND" instruction in the cycle or a time-controlled program.

An instance DB must be specified when "SMS_SEND" is called.

"SMS_SEND" must be called more than once to process a block function. This means that calling "SMS_SEND" in a "queue" is pointless. The end of the block function is indicated by BUSY = 0.

Communication connection abort

If the CPU changes to STOP mode while "SMS_SEND" is active, the communication connection to the TS Adapter is aborted. The communication connection to the TS Adapter is also lost if serious communication problems arise on the MPI bus or, for example, the CPU is switched off.

In these cases the telephone number, service center number and SMS message that were already transferred to the TS adapter are not discarded by the TS adapter. The TS Adapter transfers the data to the wireless modem. However, the acknowledgement that is transferred from the wireless modem to the TS Adapter, is discarded by the TS Adapter.

If a user program attempts to send an SMS to the CPU while the TS Adapter is in the status described above, "SMS_SEND" is terminated with the return value W#16#B10A. The user program can later repeat the sending of an SMS. You also receive the return value

W#16#B10A, if different CPUs attempt to send an SMS at the same time via the same TS Adapter.

NOTICE
Changing the STEP 7 user program You can only change the parts of your STEP 7 user program that directly affect calls of "SMS_SEND" when the CPU is in STOP mode. This relates, in particular, to deleting and replacing program blocks that contain "SMS_SEND" calls. Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status for the communication instructions for non-configured S7 connections ("X_PUT (Page 6801)", "X_GET (Page 6799)", "X_SEND (Page 6794)", "X_RCV (Page 6795)", "X_ABORT (Page 6803)"). A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The input parameters of the function block are copied to in internal buffer when "SMS_SEND" is first called. Do not change this data before the first call is completed (return value W#16#7001), otherwise inconsistent data could be transferred.

Transfer of PIN to the wireless modem

If you use a SIM card on which the PIN check is enabled, the PIN must be transferred during the initialization phase of the wireless modem. This is possible using the init string of the TS Adapter, which you can set using TeleService. Set the init string as follows (example for PIN = 4711):

- AT+CPIN="4711";AT&F.....

When power is restored, the TS Adapter sends the string "AT+CPIN="4711" once, along with the PIN for the SIM card used in the wireless modem, to the connected components. The wireless modem is then initialized with "AT&F...".

Caution: An incorrect PIN is not reported during the initialization, but only when an SMS is sent via the feedback message of "SMS_SEND".

Sending a fax

By prefixing a special number of the network provider (for example, "99") before actual number, the SMS is converted by the network provider into a fax and sent to a fax machine. This is a service of the network provider, not a function of the TS Adapter.

The fax number is entered with the prefix of the network provider in the PHONE_NO parameter.

Example of a network with the fax number 07214711:

- PHONE_NO = '9907214711'

Sending an e-mail

By dialing a special number of the network provider (for example, "8000") and appending an e-mail address before the SMS message, the SMS can be sent to an e-mail address. This is a service of the network provider, not a function of the TS Adapter.

You enter the corresponding number in the PHONE_NO (Page 6817) parameter.

In the SMS message, the e-mail address prefixes the actual text of the message and is delimited from this by a separator. This shortens the maximum length of the SMS payload by the length of the e-mail address + separator. Separators are, for example, blanks (" ") or colons (":").

Some service providers demand a "*" instead of the "@" character.

Example:

- The text 'CPU in STOP' is to be sent to the e-mail address "Surname*provider.com".
- PHONE_NO = 8000
- MESSAGE = Surname*provider.comCPU in STOP

Parameters

The following table shows the parameters of the "SMS_SEND" instruction:

Parameters	Declaration	Data type	Memory area	Description
MPI_TS_ ADAPTER	Input	INT	I, Q, M, D, L or constant	MPI address of the TS Adapter
PHONE_NO (Page 6817)	Input	ANY	D	Reference to a data string with a maximum length of 31 characters
SCENTER_NO (Page 6817)	Input	ANY	D	Reference to a data string with a maximum length of 20 characters
MESSAGE (Page 6817)	Input	ANY	D	Reference to a data string with a maximum length of 160 characters
BUSY (Page 6818)	Output	BOOL	I, Q, M, D, L	• BUSY=1: The sending of the SMS is not yet completed • BUSY=0: Processing of "SMS_SEND" was terminated
STATUS (Page 6818)	Output	INT	I, Q, M, D, L	Return value of "SMS_SEND"

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

Virtual parameters EN and ENO and the BR bit

The virtual block parameters EN and ENO occur only with the integration of "SMS_SEND" in the LAD or FBD representation.

- Input parameter EN:
  The status of the result of logic operation (RLO) is saved to the BR bit during the block call.
- Output parameter ENO:
  If "SMS_SEND" is processed error-free, the status of the BR bit that existed when the block was called is restored again at the end of the block call.
  If an error message is output via the output parameter STATUS, the BR bit is set to "0" immediately before exiting "SMS_SEND".

PHONE_NO parameter

Description

The input parameter PHONE_NO specifies the telephone number to which an SMS is to be sent. Enter the complete telephone number including country code, area code and subscriber number. The string is transferred unchanged to the wireless modem. If you use characters that are not numerals, make sure that the wireless modem used supports these characters.

Parameter SCENTER_NO

Description

The SCENTER_NO input parameter specifies the service center number to which the SMS will be sent. The string is transferred unchanged to the wireless modem.

MESSAGE parameter

Description

The MESSAGE input parameter specifies the SMS message. This is transferred using the "SMS_SEND" instruction, via the TS Adapter and the GSM wireless modem to the network provider.

BUSY parameter

Description

"SMS_SEND" is an asynchronous instruction, in other words, its processing extends over multiple calls.

- If the BUSY output parameter = 1, the internal status of "SMS_SEND" is output in the output parameter STATUS (Page 6818).
- When processing is completed, this is indicated by the output parameter BUSY = 0. The STATUS (Page 6818) output parameter then indicates whether the job was completed error-free (STATUS = W#16#0000) or with errors.

STATUS parameter

Description

The return values of "SMS_SEND" can be classified as follows:

- W#16#0000: "SMS_SEND" was ended successfully.
- W#16#7xxx: Status of "SMS_SEND"
- W#16#8xxx: when calling a communication instruction or the instruction "BLKMOV" an error was reported.
- W#16#9xxx: parameter error when calling "SMS_SEND"
- W#16#Bxxx: an error was reported by the TS adapter.
- W#16#Cxxx: an error was reported by the wireless modem.

The following table shows the return values of "SMS_SEND" except for error codes of the communication instructions used or the wireless modem.

Return value (W#16#...)	Explanation	Notes
0000	The processing of "SMS_SEND" was completed without errors.	
7000	"SMS_SEND" was reset (the communication with the TS adapter was terminated in the meantime).	Call "SMS_SEND" again.
7001	"SMS_SEND" is active (first call, BUSY = 1). The function has just been started.	
7002	"SMS_SEND" is active (next call, BUSY = 1). The processing of the function is not yet completed.	

Return value (W#16#...)	Explanation	Notes
8xxx or 8zxx	Processing of "SMS_SEND" was completed with an error code of the communication instructions called internally or the "BLKMOV" instruction. If the error message originates from the "BLKMOV" instruction, the meaning is as follows • z = 2 : Error copying the parameter PHONE_NO to the internal buffer • z = 3 : Error copying the parameter SCENTER_NO to the internal buffer • z = 4 : Error copying the parameter MESSAGE to the internal buffer	For detailed information, refer to the description of the "BLKMOV" instruction and the error information of the communication instructions: AUTOHOTSPOT
9001	Length of PHONE_NO = 0 or > 31	The telephone number must consist of at least 1 character and not more than 31 characters.
9002	Length of SCENTER_NO = 0 or > 20	The service center number must consist of at least 1 character and not more than 20 characters.
9003	Length of MESSAGE = 0 or > 160	The SMS message must consist of at least 1 character and not more than 160 characters.
B10A	A job is already stored on the TS Adapter and this job is currently being processed by the TS Adapter.	Call "SMS_SEND" again.
B301	A remote connection to a PG/PC is still established.	Call "SMS_SEND" again later.
B302	The TS Adapter is directly connected to a PG/PC (direct connection).	Connect the TS adapter to a GSM wireless modem.
B303	The TS Adapter is not connected to a wireless modem or a PG/PC.	Connect the TS adapter to a GSM wireless modem.
B304	The interface to the wireless modem is not currently ready to send an SMS.	Call "SMS_SEND" again later.
B614	The connected modem did not reply to the SMS-specific command within the monitoring period.	Check whether you have connected a GSM wireless modem. Call "SMS_SEND" again later.
B615	The TS Adapter has received an unspecific error message from the connected modem.	Check whether you have connected a GSM wireless modem.
Cxxx	The processing of "SMS_SEND" was terminated with an error code of the GSM modem. xxx: Error number	For a list of the error numbers, refer to the your wireless modem manual or to the GSM standards GSM 04.11, GSM 03.40 and GSM 07.05. Examples: • C136 = No SIM card present in the GSM modem • C137 = PIN for GSM modem not correct (see transfer of the PIN to the wireless modem) • C001 = Wrong phone number for the service center

AS_MAIL: Transfer email

Description of AS_MAIL

Required instructions

The "AS_MAIL" instruction uses the Simple Mail Transfer Protocol (SMTP) to transfer an e-mail from a CPU to a mail server. "To run, "AS_MAIL" requires the following instructions:

- "S_COMP: Compare character strings (Page 5205)"
- "FIND: Find characters in a character string (Page 5217)"
- "INSERT: Insert characters in a character string (Page 5216)"
- "LEFT: Read the left character of a character string (Page 5213)"
- "LEN: Determine the length of a character string (Page 5212)"
- "RIGHT: Read the right characters of a character string (Page 5214)"

Description

"AS_MAIL" is an asynchronous instruction, in other words, its processing extends over multiple calls. When you call the "AS_MAIL" instruction, you must specify an instance DB.

You initialize the block by a call with COM_RST = 1.

You start the sending of an e-mail with an edge change from "0" to "1" for the REQ parameter.

The job status is indicated by the output parameters BUSY , "DONE", "ERROR", "STATUS" and "SFC_STATUS". "SFC_STATUS" corresponds in this case to the "STATUS" output parameter of the called communication blocks or the "BLKMOV" instruction.

The output parameters DONE, ERROR, STATUS, and SFC_STATUS are each displayed for only one cycle if the status of the BUSY output parameter changes from "1" to "0".

The following table shows the relationship between BUSY, DONE, and ERROR. Using this table, you can determine the current status of the instruction "AS_MAIL" or when the sending of the e-mail is completed.

BUSY	DONE	ERROR	Description
1	Irrelevant	Irrelevant	The job is being processed.
0	1	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS and SFC_STATUS parameters.
0	0	0	The "AS_MAIL" instruction was not assigned a (new) job.

If the CPU changes to STOP mode while "AS_MAIL" is active, the communication connection to the mail server aborts. The communication connection to the mail server will also be lost if

communication problems occur on the Industrial Ethernet bus. In this case, the transfer of the e-mail will be interrupted and it will not reach its recipient.

NOTICE
Changing user programs You can change the parts of your user program that directly affect calls of "AS_MAIL" only: • when the CPU is in "STOP" mode or • when no mail is being sent (REQ = 0 and BUSY = 0). This relates, in particular, to deleting and replacing program blocks that contain "AS_MAIL" calls or calls for the instance DB of "AS_MAIL". Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status for the TPC/IP communication functions via Industrial Ethernet. After transferring the changes, run a warm or cold restart on the CPU or set the COM_RST parameter of the "AS_MAIL" instruction.

Data consistency

The ADDR_MAIL_SERVER input parameter of the instruction is re-adopted by the "AS_MAIL" instruction each time the sending of an e-mail is triggered. If a change is made during operation, the "new" value only becomes effective once an e-mail is triggered again.

In contrast, the parameter settings of WATCH_DOG_TIME, TO_S, CC, FROM, SUB, TEXT, ATTACHMENT, and, if applicable, USERNAME and PASSWORD are adopted while the "AS_MAIL" instruction is running, which means that they may only be changed when the job has completed (BUSY = 0)

Setting the parameters of the TS Adapter IE

On the TS Adapter IE, you need to specify parameters for outgoing calls in such a way as to enable the TS Adapter IE to establish a connection to the dial-up server of your service provider.

If you set "When required" for connection establishment, the connection will only be established when an e-mail needs to be sent.

Connection establishment can take longer (approx. one minute) for an analog modem connection. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection.

Parameters

The following table shows the parameters of the "AS_MAIL" instruction:

Parameter	Declaration	Data type	Memory area	Description
COM_RST	Input / Output	BOOL	I, Q, M, D, L	Input/output parameter "COMPLETE" RESTART: The block has an initialization routine that is processed when the COM_RST input is set. The COM_RST parameter of the "AS_MAIL" instruction must be set to "1": • once, before the first call and • every time that the corresponding instance DB is reloaded. After initialization, the "AS_MAIL" instruction, resets the COM_RST parameter.
REQ	Input	BOOL	I, Q, M, D, L	Control parameter REQUEST: Activates the sending of an e-mail on a rising edge.
ADDR_MAIL_SERVER	Input	DWORD	I, Q, M, D, L	IP address input parameter of the mail server: Specify as a data word in HEX format, for example: IP address = 192.168.0.200. ADDR_MAIL_SERVER = DW#16#C0A800C8, where: • 192 = 16#C0, • 168 =16#A8 • 0 = 16#00 and • 200 = 16#C8.
WATCH_DOG_TIME	Input	TIME	I, Q, M, D, L	Input parameter for max. period of time: The "AS_MAIL" instruction should establish a connection within the period specified by WATCH_DOG_TIME. The block is exited with an error if this period is exceeded. The time before the block is exited and the error is output can exceed the WATCH_DOG_TIME because connection termination also takes a certain amount of time. To begin with, you should set a period of 2 minutes. If you have an ISDN telephone connection, a significantly shorter period can be selected.
USERNAME	Input	ANY	D	Input parameter for user name: Reference to a STRING type tag with a maximum length of 180 characters. A user name is an absolute requirement for authentication.
PASSWORD	Input	ANY	D	Input parameter for password: Reference to a STRING type tag with a maximum length of 180 characters. A password is an absolute requirement for authentication.
TO_S	Input	ANY	D	Input parameter for receiver addresses: Reference to a STRING type tag with a maximum length of 240 characters (see call example).

Parameter	Declaration	Data type	Memory area	Description
CC	Input	ANY	D	Input parameter for CC recipient addresses: Reference to a STRING type tag with a maximum length of 240 characters (see call example).
FROM	Input	ANY	D	Input parameter for sender address: Reference to a STRING type tag with a maximum length of 240 characters (see call example).
SUB	Input	ANY	D	Input parameter for subject of the e-mail: Reference to a STRING type tag with a maximum length of 240 characters.
TEXT	Input	ANY	D	Input parameter for text of the e-mail: Reference to a data string with a maximum length of 240 characters.
ATTACHMENT	Input	ANY	I, Q, M, D, L	Input parameter for e-mail attachment: Reference to a byte/word/double word field with a maximum length of 65534 bytes.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: E-mail transmission is not yet complete. • BUSY=0: The processing of "AS_MAIL" was stopped.
DONE	Output	BOOL	I, Q, M, D, L	• DONE = 0: Job not yet started or still executing. • DONE = 1: Job was executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR=1: An error occurred during processing. STATUS and SFC_STATUS supply detailed information about the type of error.
STATUS (Page 6825)	Output	INT	I, Q, M, D, L	Output/status parameter STATUS: Return value or error information of the "AS_MAIL" instruction.
SFC_STATUS (Page 6825)	Output	INT	I, Q, M, D, L	Output/status parameter "SFC_STATUS": Error information of the called communication blocks or the "BLKMOV" instruction.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 231)".

Note

Notes to the parameters for the instruction

- If the assignment of the parameters USERNAME and PASSWORD is removed, re-entered or changed in the block in which "AS_MAIL" is called, this change will only become effective if the corresponding instance DB is reloaded and the COM_RST parameter is set. Otherwise, the pointer remains on the corresponding parameter in the instance DB.
- Because of runtime and memory space, the "AS_MAIL" instruction does not perform any syntax check of the parameters TO_S, CC and FROM. Please note the correct syntax of the parameter.
- The optional parameters CC, TEXT and ATTACHMENT are only sent with the e-mail if they are assigned in the block in which the "AS_MAIL" instruction is called. If such an assignment is removed from the call interface of the "AS_MAIL" instruction, this change only becomes effective when the corresponding instance DB is reloaded and the COM_RST parameter is set. Otherwise, the pointer remains on the corresponding parameter in the instance DB. Please note that empty strings are not permitted in a parameter assignment.
- If the ATTACHMENT parameter points to an array in a data block and if the size of this array is changed and stored, the ANY pointer of the ATTACHMENT parameter must be entered again at the interface of the "AS_MAIL" instruction.

SMTP authentication

Authentication refers here to a procedure for ensuring an identity, for example, by a password query.

The "AS_MAIL" instruction supports the SMTP authentication procedure AUTH-LOGIN, that is requested by most mail servers. For information about the authentication procedure of your mail server, please refer to your mail server manual or the website of your Internet service provider.

To use the AUTH-LOGIN authentication procedure, the "AS_MAIL" instruction requires the user name with which it can log in with the mail server. This user name corresponds to the user name with which you set up a mail account on your mail server. It is available to the "AS_MAIL" instruction in the USERNAME parameter.

To log on, the "AS_MAIL" instruction also requires the associated password. This password corresponds to the password you specified when you set up your mail account. It is made available to the "AS_MAIL" instruction in the PASSWORD parameter.

The user name and the password are each transferred to the mail server unencrypted (BASE64-coded).

If no user name is specified in the DB, the AUTH-LOGIN authentication procedure is not used. The e-mail is then sent without authentication.

STATUS and SFC_STATUS parameters

Description

The return values of the "AS_MAIL" instruction can be classified as follows:

- W#16#0000: "AS_MAIL" was completed successfully
- W#16#7xxx: Status of "AS_MAIL"
- W#16#8xxx: An error was reported during the internal call of a communication instruction, during the "BLKMOV" instruction call, or by the mail server.

The following table shows the return values of "AS_MAIL" except for error codes of the communication blocks called internally and the "BLKMOV" instruction.

Return value STATUS (W#16#...):	Return value SFC_STATUS (W#16#...):	Explanation	Notes
0000	-	The processing of "AS_MAIL" was completed without errors.	A error-free completion of "AS_MAIL" does not mean that the sent e-mail will necessarily arrive (see below - note on point 1)
7001		"AS_MAIL" is active (BUSY = 1).	
7002	7002	"AS_MAIL" is active (BUSY = 1).	
8xxx	xxxx	Processing of "AS_MAIL" was completed with an error code of the communication instructions called internally or the "BLKMOV" instruction.	For detailed information on the evaluation of the SFC_STATUS parameter, refer to the descriptions of the STATUS parameter of the "BLKMOV" instruction.
800z	xxxx	The error message originates from the "BLKMOV" instruction and means: • z = 1 Error copying the TO_S parameter to the internal buffer • z = 2 Error copying the CC parameter to the internal buffer • z = 3 Error copying the FROM parameter to the internal buffer • z = 4 Error copying the SUB parameter to the internal buffer • z = 5 Error copying the TEXT parameter to the internal buffer • z = 6 Error copying the ATTACHMENT parameter to the internal buffer • z = 7 Error copying the USERNAME parameter to the internal buffer • z = 8 Error copying the PASSWORD parameter to the internal buffer	
8010	xxxx	Error during connection establishment.	COM_RST was possibly not set after loading instance DB.

Return value STATUS (W#16#...):	Return value SFC_STATUS (W#16#...):	Explanation	Notes
8011	xxxx	Error sending the data.	For detailed information on the evaluation of SFC_STATUS, refer to the descriptions of the STATUS parameter of the "TSEND (Page 6213)" instruction.
8012	xxxx	Error receiving the data.	For more information on the evaluation of SFC_STATUS, refer to the descriptions of the STATUS parameter of the "TRCV (Page 6216)" instruction.
8013	xxxx	Error during connection establishment.	For more information on the evaluation of SFC_STATUS, refer to the descriptions of the STATUS parameter of the "TCON (Page 6208)" and "TDISCON (Page 6212)" instructions.
8014	-	Establishment of a connection is not possible.	You have possibly entered an incorrect mail server IP address (ADDR_MAIL_SERVER) or a time span that is too short (WATCH_DOG_TIME) to establish the connection. It is also possible that the CPU has no connection to the network or that the CPU configuration is incorrect.
82xx, 84xx, or 85xx	-	The error message originates from the mail server and corresponds, except for the "8", to the error number of the SMTP protocol. The following columns list several error codes that can occur:	See point 2 in the note.
8450	-	Action not executed: Mailbox not available/ not reachable.	Try again later.
8451	-	Action aborted: Local processing error	Try again later.
8500	-	Syntax error: Error not recognized. This also includes the error when a command string is too long. This can also be occur when the e-mail server does not support the LOGIN authentication procedure.	Check the parameters of "AS_MAIL". Try to send an e-mail without authentication. To do this, replace the USERNAME parameter with an empty string.
8501	-	Syntax error: Parameter or argument incorrect	You have possibly entered an incorrect address in TO_S or CC.
8502	-	Command unknown or not implemented.	Check your entries, in particular the FROM parameter. This is possibly incomplete and you have forgotten "@" or ".".
8535	-	SMTP authentication incomplete.	You have possibly entered an incorrect user name or incorrect password.
8550	-	Mail server cannot be reached, you have no access rights.	You have possibly entered an incorrect user name or password, or the mail server does not support your LOGIN. Another cause of error could be an incorrect entry of the domain name after the "@" in TO_S or CC.

Return value STATUS (W#16#...):	Return value SFC_STATUS (W#16#...):	Explanation	Notes
8552	-	Action aborted: Assigned memory size has been exceeded	Try again later.
8554	-	Transmission failed.	Try again later.

Note

Status error

1. An incorrect entry of the recipient addresses does not generate a status error of the "AS_MAIL" instruction. In this case, there is no guarantee that the e-mail will be sent to other recipients, even if these were entered correctly.
2. You will find more detailed information on the SMTP error code and other error codes in SMTP protocol on the Internet or in the error documentation of the mail server. You can also view the most recent message from the mail server in your instance DB in the BUFFER1 parameter. If you look under "Data", you will find the data most recently sent by the "AS_MAIL" instruction.

Example of "AS_MAIL" call

Description

The data that is available to the "AS_MAIL" instruction based on the ANY pointer is stored in a data block. In this example, DB 2 is used for this.

Because of runtime and memory space, the "AS_MAIL" instruction does not perform any syntax check of the parameters TO_S, CC and FROM.

Parameters TO_S, CC and FROM

The TO_S, CC and FROM parameters are ANY pointers to strings with, for example, the following content:

- TO: <wenna@mydomain.com>, <ruby@mydomain.com>,
- CC: <admin@mydomain.com>, <judy@mydomain.com>,
- FROM: <admin@mydomain.com>

Note the following rules when entering the parameters:

- The "TO:", "CC:" and "FROM:" characters must be entered.
- A space and an opening angled bracket "<" must be entered before each address.
- A closing angled bracket ">" must be entered after each address.
- A comma must be entered after each address in TO_S and CC.
- Only one e-mail may be entered in FROM, and there must be no comma at the end of this address.

Initialization / startup in OB 100

The COM_RST parameter of the "AS_MAIL" instruction must be set before the first call of the instruction, or always after a reloading of the associated instance DB.

In the following example, DB 1 is the instance DB of the "AS_MAIL" instruction.

SET		//Set RLO
R	DB1.REQ	//REQ is reset so that the "AS_MAIL" instruction does not start to send an e-mail immediately after COM_RST is set.
S	DB1.COM_RST	//COM_RST is set; this must be set once for initialization before the first call of the "AS_MAIL" instruction, or always after reloading the associated instance DB.

Cyclic call

This example of the "AS_MAIL" instruction call must be called cyclically (for example, in OB1). In the user program, use "Modify tags" to set the REQ parameter to "1".

You can copy these lines to an STL source file and compile them. To do this, however, the blocks DB 1 and DB 2 must be entered in the project's tag table.

Example of "AS_MAIL" call	Explanation
`DATA_BLOCK "DB2"`	`//Global data block with the data for the "AS_MAIL" instruction`
`TITLE =`	
`VERSION : 1.0`	
`STRUCT`	
`TO_S : STRING [240 ] := 'TO: <wenna@mydomain.com>, <ruby@mydomain.com>,';`	`//Address of recipient`
`CC : STRING [240 ] := 'CC: <admin@mydomain.com>, <judy@mydomain.com>,';`	`//CC Address of recipient`
`FROM : STRING [60 ] := 'FROM: <admin@mydomain.com>';`	`//Address of sender`
`SUB : STRING [60 ] := 'Status plant 7';`	`//Subject`
`TEXT : STRING [240 ] := 'Fault in plant 7 sector 2';`	`//Text`
`USERNAME : STRING [60 ] := 'admin';`	`//User name for LOGIN authentication`
`PASSWORD : STRING [60 ] := 'test';`	`//Password for LOGIN authentication`
`ATTACHMENT : ARRAY [1 .. 178 ] OF BYTE;`	`//Attachment`
`END_STRUCT ;`	
`BEGIN`	
`END_DATA_BLOCK`	
`DATA_BLOCK "DB1"`	`//Instance DB of the "AS_MAIL" instruction`
`TITLE =`	

Example of "AS_MAIL" call	Explanation
AUTHOR : TELESERV	
FAMILY : TELESERV	
VERSION : 1.0	
"AS_MAIL"	
BEGIN	
END_DATA_BLOCK	
FUNCTION_BLOCK FB 1	
TITLE =Example AS_MAIL	The COM_RST parameter of the "AS_MAIL" instruction must be set before the first "AS_MAIL" call, or always after reloading the associated instance DB. The call condition is set during the execution of the user program.
VERSION : 1.0	
VAR	
CallCondition: BOOL ;	
END_VAR	
BEGIN	
NETWORK	
CALL AS_MAIL, DB1 (	//Calling AS_MAIL
REQ := # CallCondition,	//Job triggered on rising edge
ADR_MAIL_SERVER := DW#16#C0A800C8,	//Corresponds to the IP address 192.168.0.200 in HEX format
WATCH_DOG_TIME := T#2M,	//Time span of 2 minutes within which sending should be completed
USERNAME := DB2.USERNAME,	//Pointer to user name of the sender. If there is no DB2.USERNAME , in other words, no parameter is assigned, the LOGIN authentication procedure is not used.
PASSWORD := DB2.PASSWORD,	//Pointer to the password
TO_S := DB2.TO_S,	//Pointer to the recipient address
CC := DB2.CC,	//Pointer to the CC recipient address (optional). If there is no DB2.CC, in other words, if this is not assigned, the e-mail is not sent to the CC recipient.
FROM := DB2.FROM,	//Pointer to the address of the sender
SUB := DB2.SUB,	//Pointer to the subject of the mail
TEXT := DB2.TEXT,	//Pointer to the text of the mail (optional). If there is no DB2.TEXT, in other words, if this is not assigned, the e-mail is sent without text.
ATTACHMENT := DB2.ATTACHMENT,	//Pointer to the mail attachment (optional). Example: 101 bytes in the mail attachment - ATTACHMENT := P#DB2.DBX974.0 BYTE 101 (example relates to the above DB). //If there is no DB2.ATTACHMENT, in other words, if this is not assigned, the e-mail is sent without an attachment.

Example of "AS_MAIL" call	Explanation
BUSY := M46.0,	//Output/status parameter BUSY
DONE := M46.1,	//Output/status parameter DONE
ERROR := M46.2,	//Output/status parameter ERROR
STATUS := MW48,	//Output/status parameter STATUS
SFC_STATUS := MW50	//Output/status parameter SFC_STATUS
);	
	//Checking the call results at the end of the processing
U M 46.0;	//Is BUSY ==0?
SPB End;	//No, BUSY ==1, "AS_MAIL" is still processing the job
U M 46.1;	//Was the processing of "AS_MAIL" completed without errors?
SPB End;	
NOP 0;	//No, an error occurred during processing, BUSY ==0, BR ==0.
	//Add further error evaluation here, if necessary, STATUS, SFC_STATUS
End:BE;	
END_FUNCTION_BLOCK	

Note

Optional parameters

For the optional parameters USERNAME, PASSWORD, CC, TEXT and ATTACHMENT, the following applies:

If an assignment is removed or re-inserted, this change will only be valid when the associated instance DB (in this example, DB 1) is reloaded and the "COM_RST" parameter is set.

Program editor 5

5.1 Overview of the program editor

Function of the program editor

The program editor is the integrated development environment for programming functions, function blocks, and organization blocks. If offers comprehensive support for programming and troubleshooting.

The appearance and functionality of the program editor can vary depending on the CPU, programming language and block type used.

Structure of the program editor

Using LAD as an example, the following figure shows the components of the program editor:

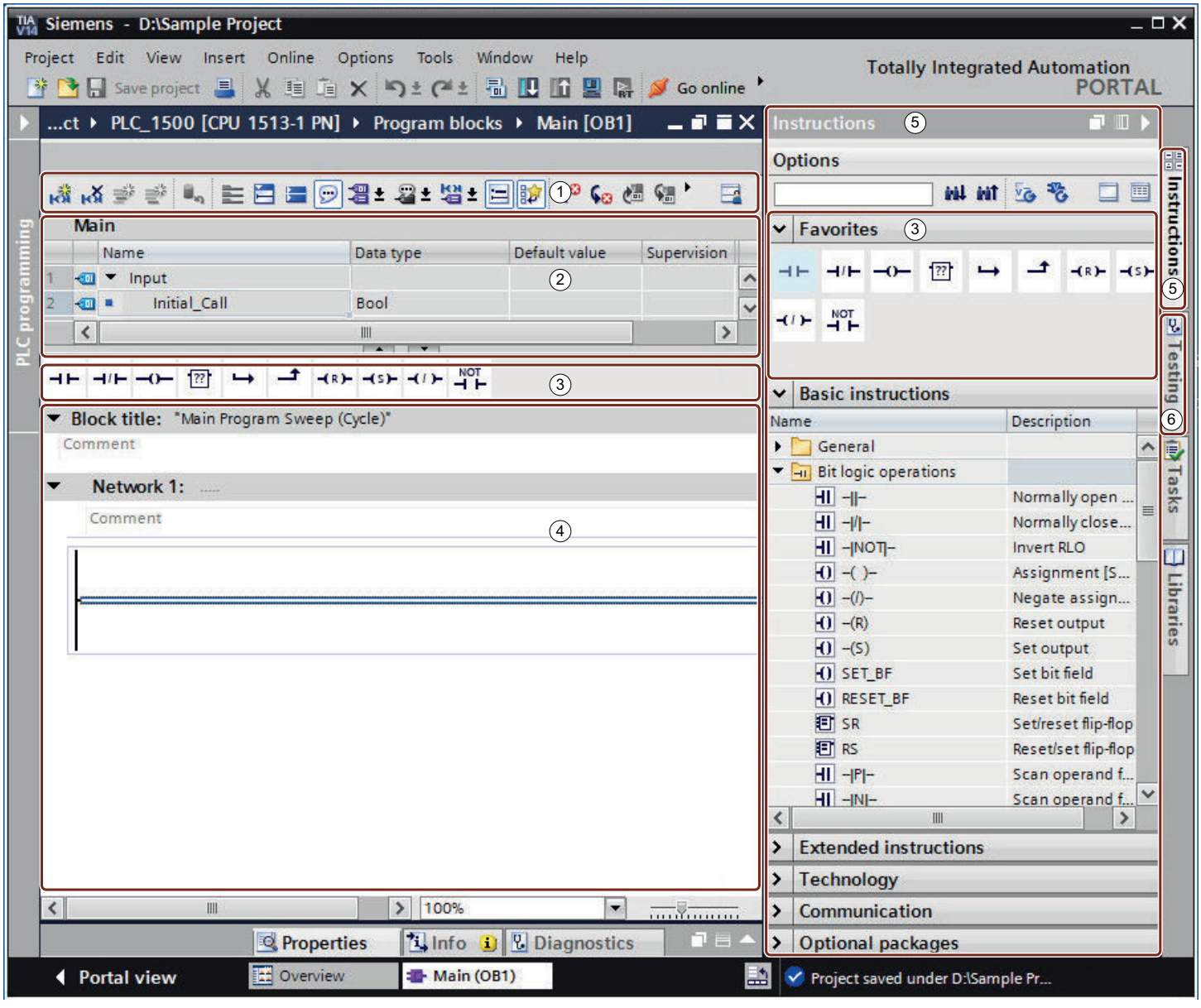

① Toolbar
② Block interface
③ "Favorites" pane in the "Instructions" task card and favorites in the programming window
④ Programming window
⑤ "Instructions" task card
⑥ "Testing" task card

Toolbar

The toolbar allows you to access the principal functions of the program editor, such as:

- Show and hide absolute operands
- Showing and hiding favorites
- Skip to syntax errors
- Update block calls
- Show and hide program status

The functions available in the toolbar can vary depending on the programming language used.

Block interface

The block interface contains the declarations for local tags used solely within the block. The sections available depend on the block type.

Favorites

You can save frequently used instructions as favorites. These favorites are then displayed in the "Instructions" task card and the "Favorites" pane. You can also display favorites in the programming window using the program editor toolbar. This allows you to access your favorites even when the "Instructions" task card is not visible.

Programming window

The programming window is the work area of the program editor. You can enter the program code in this window. The appearance and functionality of the program window can vary depending on the programming language used.

"Instructions" task card

The "Instructions" task card offers you easy access to all instructions available for creating your program. The instructions are broken down by area into a number of different palettes. If an instruction profile is active, the offered instructions may vary. You receive a brief description for each instruction. If several versions of an instruction are available, the versions are listed in a drop-down list and the currently selected version is used in the program. If required, you can select a different instruction version. Read the information in the section "Basic information on instruction versions (Page 6850)".

With the task card toolbar, you can access the following functions:

- Search up or down for instructions
  You have the option of searching the panes for specific instructions.
- Upgrade project and instructions
  If you are still working with a project which was created with TIA Portal V14, only the instruction versions from V14 will be available. Using the "Upgrade project and instructions" button you can easily upgrade your project to TIA Portal V14 SP1. The "Instructions" task card then receives the new instruction versions of TIA Portal V14 SP1.

  Note

  Please note the following:
  - Your program is not changed by the upgrade, i.e. it still contains the old instruction versions from V14. To use the new instruction versions you must update your program.
  - The "project and operation upgrade" button is only active if you are still working with a TIA Portal V14 project. The button becomes inactive as soon as you upgrade your project to TIA Portal V14 SP1.
- Update program in the current PLC
  Updates the CPU program to the new instruction versions.
- Changing the pane mode
  You can choose between single and multi-pane mode.
  See also: Change pane mode
- Showing/hiding column headers
  You can show or hide column headers in the panes. You can modify the arrangement of columns by clicking a column header and moving the column to the new position by means of drag-and-drop.

"Testing" task card

You can set settings in the "Testing" task card for troubleshooting using the program status. The functions of the "Testing" task card are only available in online mode. It contains the following panes, which are displayed depending on the selected CPU and the configured programming language of the block:

- CPU operator panel
  The CPU operator panel allows you to switch the CPU operating mode.
- Breakpoints
  You can test blocks you created in the textual programming languages STL and SCL in single step mode. To do this, set breakpoints in the program code.
  In the "Breakpoints" pane, you can find all breakpoints that you have set and you can activate, delete, navigate to, or set the call environment for the individual breakpoints.
- PLC register
  This pane allows you to read off the values for the PLC registers and the accumulators.
- Sequence control
  In this pane you can set the operating mode for testing sequencers for GRAPH blocks.
- Test settings
  This pane allows you to specify the test settings for the GRAPH block.

- Call environment
  This pane allows you to specify the call environment for the block.
- Call hierarchy
  In this pane, you can trace the call hierarchy of the blocks. You only see the call hierarchy during block monitoring.

See also

Overview of the block interface (Page 6965)

Enlarging the programming window (Page 6841)

Find instructions (Page 6846)

Using instruction profiles (Page 6853)

5.2 Using the keyboard in the program editor

Navigating in the editor

Function	Shortcut key
Open "Instructions" task card	<Ctrl+Shift+C>
Open "Testing" task card	<Ctrl+Shift+O>
Add new block	<Ctrl+N>
Open block/PLC data type	<F7>
Expand all networks	<Alt+F11>
Collapse all networks	<Alt+F12>
Navigate to the next point of use of the selected block or operand	<Ctrl+Shift+G>
Navigate to the previous point of use of the selected block or operand	<Ctrl+Shift+F>
Navigate to the next read/write access	<Alt+F8>
Navigate to the previous read/write access	<Alt+F9>
Navigate to the definition of the selected block or operand	<Ctrl+Shift+D>

Navigating in the program code (LAD/FBD)

Function	Selected object	Shortcut key
Navigating between objects in the network	Object in the network	Arrow keys
Navigate to the first element of the network	Object in the network	<Home>
Navigate to the last element of the network	Object in the network	<End>
Navigate to the next element of the network	Object in the network	<Tab>
Navigate to the previous element of the network	Object in the network	<Shift+Tab>
Insert network	Any	<Ctrl+R>

Navigating in the program code (STL/SCL)

Function	Position of the cursor	Shortcut key
Navigating in the program code	Line	Arrow keys
One word to the right/left	Line	<Ctrl+arrow keys>
Cursor to start of line	Line	<Home>
Cursor to end of line	Line	<End>
Cursor to start of code section	Line	<Ctrl+Home>
Cursor to end of code section	Line	<Ctrl+End>
To the next network (STL only)	Network title	<Down arrow>
To the next network (STL only)	Line	<Tab> Repeat the shortcut keys until the insertion point is in the next network.
To the previous network (STL only)	Network title	<Up arrow>

Function	Position of the cursor	Shortcut key
To the previous network (STL only)	Line	<Shift+Tab> Repeat the shortcut keys until the insertion point is in the previous network.
Insert network	Any	<Ctrl+R>

Inserting instructions (LAD)

Function	Selected object	Shortcut key
Insert normally open contact	Rung	<Shift+F2>
Insert normally closed contact	Rung	<Shift+F3>
Insert empty box	Rung	<Shift+F5>
Insert assignment	Rung	<Shift+F7>
Insert "Open branch"	Rung	<Shift+F8>
Insert "Close branch"	Rung	<Shift+F9>

Inserting instructions (FBD)

Function	Selected object	Shortcut key
Insert AND logic operation	Network, input or output	<Shift+F2>
Insert OR logic operation	Network, input or output	<Shift+F3>
Insert empty box	Network	<Shift+F5>
Insert assignment	Network, input or output	<Shift+F7>
Insert "Open branch"	Connection line between two boxes	<Shift+F8>
Invert RLO	Network, input or output	<Ctrl+Shift+4>
Insert input	Network, input or output	<Ctrl+Shift+3>

Entering operands (LAD/FBD/GRAPH)

Function	Selected object	Shortcut key
Activate the input field for the first operand of the instruction	Instruction	<Return> Or <Any letters/numbers> An empty input field opens on <Return>, any letters or numbers will be entered in the input field.
Activate input field for the operand	Operand	<F2>

Function	Selected object	Shortcut key
Delete operand	Operand	<Del>
Define tag	Operand	<Ctrl+Shift+I>
Rewire tag	Operand	<Ctrl+Shift+P>
Rename tag	Operand	<Ctrl+Shift+T>
Entering operands	Input field for operands	<Any letters/numbers>
Confirm entry of the operand	Input field for operands	<Return>
Open autocompletion	Input field for operands	<Ctrl+I>
Discard current change	Input field for operands	<Esc> The input field is deactivated and the previous contents restored.

Process instructions (STL/SCL)

Function	Selected object	Shortcut key
Indent line (SCL only)	Line	<Tab> or <Ctrl+R>
Outdent line (SCL only)	Line	<Shift+Tab> or <Ctrl+Shift+R>
Automatically formatting selected text (SCL only)		<Ctrl+Shift+W>
Open "Call options" dialog	Cursor after block call	<Return>
Define tag	Operand	<Ctrl+Shift+I>
Rewire tag	Operand	<CtrlShift+P>
Rename tag	Operand	<Ctrl+Shift+T>
Expand/collapse parameter list (SCL only)	Operand	<Ctrl+Shift+Space>
Expand/collapse code section	Insertion mark within the code section	<Ctrl+Shift+Num+> <Ctrl+Shift+Num->
Expand/collapse all code sections	Any	<Ctrl+Shift+Num*> <Ctrl+Shift+Num/>
Open autocompletion	Any	<Ctrl+I> or <Ctrl+Spacebar>
Set/delete bookmarks		<Ctrl+Shift+M>
To the next bookmark		<Ctrl+Shift+6>
To the previous bookmark		<Ctrl+Shift+5>
Comment out selection	Line	<Ctrl+Shift+Y>
Remove comment	Line	<Ctrl+Shift+U>

Programming window of GRAPH

Function	Block	Shortcut key
Page Up/Down	Navigation view, single step view, sequence view, permanent instructions	<Page Up>/ <Page Down>
Navigate in the navigation view	Navigation view	<Up arrow> <Down arrow>
Expand/collapse object	Navigation view	<+> or <Right arrow> <-> or <Left arrow>
Switch between single step view and sequence view when a step or a transition is selected	Navigation view	<Return>
Switch between navigation view and work area	Navigation view, single step view, sequence view, permanent instructions	<ALT+F6>
Go to first element in a network	Single step view	<Home>
Go to last element in a network	Single step view	<End>
Switch to interlock	Single step view	<Ctrl+Home>
Switch to transition	Single step view	<Ctrl+End>
Navigate in the structure	Sequence view	Arrow keys
Go to first step	Sequence view	<Home> or <Ctrl+Home>
Go to last step	Sequence view	<End> or <Ctrl+End>
Open branch	Sequence view	<Shift+F8>
Close branch	Sequence view	<Shift+F9>
Insert sequence end	Sequence view	<Shift+F7>
Insert jump	Sequence view	<Shift+F12>
Insert step and transition	Sequence view	<Shift+F5>
Delete element	Sequence view	<Del>
Go to first editable element	Permanent instructions	<Home>
Go to next editable element	Permanent instructions	<Tab>
Go to last editable element	Permanent instructions	<End>
Go to previous editable element	Permanent instructions	<Shift+Tab>
Jump to the start of the "Action" cell	Actions	<Home>
Jump to the end of the "Action" cell	Actions	<End>
Insert new action	Actions	<Return>

Programming window of SCL

Function	Block	Shortcut key
Expand block	Block overview, block in the programming window	<Ctrl+Shift+Num+>
Collapse block	Block overview, block in the programming window	<Ctrl+Shift+Num->
Expand all blocks	Block overview, programming window	<Ctrl+Shift+Num*>
Collapse all blocks	Block overview, programming window	<Ctrl+Shift+Num/>

Monitor program

Function	Shortcut key
Set/remove breakpoint (STL, SCL)	<Ctrl+Shift+F9>
Step over breakpoint (STL, SCL)	<Ctrl+Shift+F10>
Jump to lower-level block (STL, SCL)	<Ctrl+Shift+F11>
Return to calling block (STL, SCL)	<Ctrl+Shift+F12>
Run program up to marker (cursor position) (STL, SCL)	<Ctrl+F3>
Show program status (STL, SCL, PLC tag table, watch table and data block)	<Ctrl+T>
Enable all breakpoints (STL, SCL)	<Ctrl+Shift+F2>
Disable all breakpoints (STL, SCL)	<Ctrl+Shift+F3>
Modify to 0 (LAD, FBD)	<Ctrl+Shift+9>
Modify to 1 (LAD, FBD)	<Ctrl+Shift+1>
Modify operand (LAD, FBD)	<Ctrl+Shift+2>

5.3 Enlarging the programming window

Introduction

The programming window is relatively small when all components of the application are shown. If the program code is large, you may find you have to rearrange the work area constantly. To avoid this problem, you can hide or minimize the display of the following components of the application and of the program editor:

- Project tree
- Task cards
- Block interface
- Favorites
- Comments
- Networks

Note

You can also use the "Reduce automatically" option for the task cards, project tree, and Inspector window. These windows will then be minimized automatically when you do not need them.

See also: Maximizing and minimizing the work area

Hiding and showing the project tree

The project tree allows you to access all areas of the project. You can hide the project tree while you are creating a program so you have more space for the programming window.

To show and hide the project tree, follow these steps:

1. To hide the project tree, deselect the "Project tree" check box in the "View" menu, or click on "Collapse" on the project tree title bar.
2. To show the project tree, select the "Project tree" check box in the "View" menu or click on "Extend" on the project tree title bar.

Opening and closing task cards

The task cards are located at the right-hand edge of the programming window.

To open or close the task cards, follow these steps:

1. To close the task cards, deselect the "Task card" check box in the "View" menu or click "Collapse" on the task cards title bar.
2. To open the task cards, select the "Task card" check box in the "View" menu or click "Expand" on the task cards title bar.

Hiding and showing the block interface

The block interface is shown in the upper section of the program editor. During programming you can show and hide it as required.

To show and hide the block interface, follow these steps:

1. In the lower part of the interface within the window splitter, click on the Up arrow or Down arrow.

Showing and hiding favorites

To hide or show the favorites in the program editor, follow these steps:

1. Click the "Display favorites in the editor" button in the program editor toolbar.

Showing and hiding comments

Within a block you can enter a comment for the block or for each network. These two types of comments are shown and hidden differently.

To show or hide a block comment, follow these steps:

1. Click the the triangle at the start of the line with the block title.

To show or hide network comments, follow these steps:

1. Click "Network comments on/off" on the program editor toolbar.

Note

The comments available can vary depending on the programming language used.

Opening and closing networks

Some programming languages use networks. You can open or close these networks as required.

To open or close networks, follow these steps:

1. If you want to open a network, click the right arrow in front of the network title. If you want to close a network, click the down arrow in front of the network title.

To open and close all networks, follow these steps:

1. Click "Open all networks" or "Close all networks" in the program editor toolbar.

Note

Networks are not used in every programming language.

See also

Overview of the program editor (Page 6831)

5.4 Setting the mnemonics

You can program blocks using German or international mnemonics. When you open the TIA Portal for the first time, international mnemonics is set as default. You can change the mnemonics at any time.

Procedure

To set the mnemonics, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation.
3. In the "General settings" group, select the mnemonics that you want to use.
   The mnemonics is changed in all blocks.

5.5 Displaying symbolic and absolute addresses

You have the following options for displaying operands in the program editor:

- Symbolic representation
  The symbolic operands are displayed in the program. The corresponding absolute addresses are shown in tooltips if you hold the mouse pointer over the operand.
- Absolute representation
  The absolute addresses are displayed in the program. The corresponding symbolic operands are displayed in tooltips.
- Symbolic and absolute representation
  Symbolic operands and absolute addresses are displayed in program. This setting only has an effect on blocks that were programmed with LAD, FBD, STL, and GRAPH.

Note

Note that absolute operands are not displayed for GRAPH blocks if you have disabled the option "Display absolute operands" in the "Action" column.

You can set the display of operands centrally for all new blocks of the project or for individual blocks.

Setting operand representation for new blocks

To set the operand representation for all new blocks in the project, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. Select the desired representation in the "Operand representation" drop-down list.
   The selected representation is enabled for all new blocks in the program.

Setting operand representation for an individual block

To change the representation of the operands, follow these steps:

1. Open the block in the program editor.
2. Click the "Absolute/symbolic operands" button in the program editor toolbar.
   Each time you click the button, the representation and the symbol on the button change.

Or:

1. Open the block in the program editor.
2. Click the small arrow next to the "Absolute/symbolic operands" button in the program editor toolbar.
   A drop-down list is displayed.
3. Select the required representation from the drop-down list.
   The symbol on the button changes.

See also

Basic information about operands (Page 89)

5.6 Find instructions

You have the option of finding specific instructions in the "Instructions" task card and then inserting them in your program. Note the following rules when entering search terms:

- No distinction is made between upper and lower case text.
- The search function considers parts of a search term.
- You cannot use placeholders, such as "*" and "?".
- If an instruction name includes underscores, the instruction is found even if you do not enter the underscore.
- The text in the columns "Name" and "Description" are taken into consideration in the search.

Requirement

- A block is open.
- The "Instructions" task card is open.

Procedure

To search for a specific instruction in the "Instructions" task card, follow these steps:

1. Select a starting point for the search if you want to start the search at a specific point. If you do not select a starting point, the search starts either at the top or bottom in the task card, depending on the type of search you have selected.
2. Enter a search term in the text box of the task card toolbar.
3. Click "Search down" if you want to search the task card from top to bottom.
4. Click "Search up" if you want to search the task card from bottom to top.
   The first match with the search term found is displayed as the result. If you want to continue the search, click the "Search down" or "Search up" button again. If no matches are found, you will receive a corresponding message.

See also

Overview of the program editor (Page 6831)

5.7 Editing multilingual project texts in blocks

Introduction

A block can contain the following translatable project texts:

- Title of the block and networks
- Comments about the block and the networks
- Comments in the block interface
- Comments on the operands used
- Freely defined comments

To translate these texts into another language, you must activate this language as a project language. You can find additional information on activating project languages under AUTOHOTSPOT.

Note that the texts may not exceed the length of 32767 Unicode characters even after translation.

Requirement

- Several project languages are activated.
- A block is open and it contains at least one translatable project text.
- The block has no know-how protection.

Procedure

To edit a text in all project languages, follow these steps:

1. Select the desired element in the open block. The following table provides an overview of the selection options:

Selected element	Displayed project texts
"Block title" pane	If you select the "Block title" pane, you can edit the following project texts: • Title of the block and networks • Comments about the block and the networks • Freely defined comments anywhere in the block • Comments in the block interface For multi-instances and parameter instances, only the comments you have changed on an instance-specific basis are displayed. You can recognize these comments from the fact that they are not grayed out. You can only select and edit grayed-out comments in the associated function block.
Network pane	If you select one or more network panes, you can edit the following project texts: • Titles and comments of the selected networks • Freely defined comments in the networks
Language element	If you select one or more language elements (e.g. boxes in LAD/FBD) or select a part of the program code in STL or SCL, you can edit the following project texts: • Comments on the operands used, including on global PLC tags • For block calls: Comments on the block parameters that are initialized with actual parameters • Freely defined comments for the language element
Step or transition	If you select a step or a transition in the GRAPH block, you can edit the following project texts: • Step name • Transition name

2. Open the "Properties" tab in the Inspector window.
3. Open the "Texts" tab.
   The "Texts" tab displays the texts in the active project languages and, if available, their translations.
4. You can edit the translations directly in the table on the "Texts" tab.
5. To edit the translations with an external editor, you can export the displayed texts to OOXML format using the "Export/Import project texts" buttons.

Note

Editing all project texts in the global "Project texts" table

You can also edit the translations for project texts in the global "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts". It contains all translatable texts of the entire project.

You can find additional information on translation of texts under AUTOHOTSPOT.

5.8 Use instruction versions

5.8.1 Basic information on instruction versions

Introduction

The instructions available to you for programming the user program are managed in system libraries. If a new version of a system library is installed by an update, the newer versions of the instructions of this system library may also be installed.

If there are several versions for an instruction, these are listed in a drop down list in the "Instructions" task card after the respective instruction. If the instruction versions are not shown, you can show them again using the shortcut menu of the column header. In the drop-down list, select the version of an instruction which is to be used in the program. If you do not select any versions, the most recent versions are used.

Note

Please note the following:

- You can only ever use the same version of an instruction within a device.
- If you use an instruction in a know-how-protected block with a version other than the version set in the project, you can compile the know-how-protected block without a password if the interfaces of the instruction versions are identical. The next time you open the know-how-protected block, you will need to update the call for the instruction. The corresponding call is marked in red. The call is updated automatically by the system by compiling the entire program.
- If you change the version of an instruction that other instructions depend on, the versions of the dependent instructions are also changed.
- If you select a version for an instruction that can not be run on the CPU used, the instruction is shaded out. This means that you cannot use this instruction in this version with your CPU.
- The system block in the project tree displays the block version number in its properties; it does not have to match the instruction version number of the associated instruction in the task card.
- If you change the version of an instruction in the task card, you must compile the associated system block in the project tree before you update its block version number.

Changes in the versions

New versions can be main versions or secondary versions. New versions, such as 2.0 or 3.0, have more substantial changes to them. New main versions may therefore result in changes to the block interface. New secondary versions, such as 1.3 or 1.4, contain lesser changes or remedies to errors.

Using instruction versions

You can decide within a device which version of an instruction you want to use. If you select another version for an instruction, the new version is specified for all locations of use of this instruction within your program. These instructions are identified in the program by a red frame. You must then download your program to the device to use the new instruction version.

If you open your project with a newer TIA Portal version, you can upgrade your project and the instructions to the new TIA Portal version within the "Instructions" task card. Newer versions for some instructions are then available. These do not, however, automatically replace the old versions in your program. To use the new instruction versions in your program, you must explicitly update your program.

See also

Upgrade project and instructions (Page 6851)

Update program in the current PLC (Page 6852)

5.8.2 Upgrade project and instructions

You can upgrade a project and the instruction versions to a newer TIA Portal version in one step. This will not change your program.

Note

No backward compatibility

Projects that are saved with the current version of the TIA Portal are not backward compatible with older versions due to their enhanced functionality. With an upgrade, a new project is saved in the current project version. The original project is not overwritten and can still be edited with a compatible older version of the TIA Portal.

Procedure

To upgrade a project and the instructions, follow these steps:

1. Open a code block.
2. Open the "Instructions" task card.
3. Click "Upgrade project and instructions" in the toolbar.
   The "Project compatibility" dialog opens.
4. Click "OK" to continue with the upgrade.

Result

The project and instructions are upgraded. The "Upgrade project and instructions" button becomes inactive and the function can no longer be executed.

As the next step, you can update your program so that the new instruction versions can be used.

See also

Basic information on instruction versions (Page 6850)

Update program in the current PLC (Page 6852)

5.8.3 Update program in the current PLC

If there are new instruction versions in your project, you can update the program in the PLC so that the new instruction versions are used. This may be the case, for example after upgrading your project to a new TIA Portal version. Please note that you have to compile your program after update.

Procedure

Proceed as follows to update the program in the current PLC:

1. Right-click on the PLC for which you would like to update the program in the project tree.
2. Select the "Update program" command in the shortcut menu.
   The "Update program in the current CPU" dialog opens.
3. Click "OK" to perform the update.
4. Compile your program to complete the upgrade.

Or:

1. Open a code block.
2. Open the "Instructions" task card.
3. Click on "Update program in the current PLC" in the toolbar.
   The "Update program in the current CPU" dialog opens.
4. Click "OK" to perform the update.
5. Compile your program to complete the upgrade.

See also

Basic information on instruction versions (Page 6850)

Upgrade project and instructions (Page 6851)

5.9 Using instruction profiles

5.9.1 Basics of instruction profiles

Introduction

The TIA Portal provides you with numerous instructions that you can use to program the user program. However, you may filter out specific instructions that you do not want to use. To this end you can create instruction profiles in which you can explicitly specify the instructions to be listed in the "Instructions" task card. However, although you may create several instruction profiles in a project, only one of these profiles may be active at any given time. You can exchange instruction profiles with other users by means of shared libraries.

Note

Please note the following:

- The use of instructions that are not allowed in the active profile in a block will trigger the output of a block compilation error. Such a situation may be triggered if you drag-and-drop a block from the library to your program.
- Instructions of a profile that are not supported by the currently installed products are deleted from the profile the next time it is edited. If you transfer this profile to an engineering system in which these instructions are supported by the installed products, the instructions are again present in the profile but they are disabled. You can enable these instructions as required at any time.
- If you want to make changes to the active profile, you must recompile the blocks in the project. This is also necessary when you disable or delete the active profile or when you enable a profile.

See also

Creating new instruction profiles (Page 6853)

Opening and editing instruction profiles (Page 6855)

Activating and deactivating instruction profiles (Page 6856)

Deleting instruction profiles (Page 6857)

5.9.2 Creating new instruction profiles

Requirement

The "Common data > Instruction profiles" folder is open in the project navigation.

Procedure

Proceed as follows to create a new instruction profile:

1. Double-click the "Add new profile" command.
   The Instruction Profile Editor opens and displays the new instruction profile. All instructions are activated for the new instruction profile.
2. Edit the new instruction profile to suit your requirements.

If necessary, you can rename the new instruction profile. To do this, follow these steps:

1. Right-click on the new instruction profile.
2. Select the "Rename" command in the shortcut menu.
3. Enter a name for the new instruction profile.

Note

The first instruction profile that you create will be used as active profile. In this case, compile all blocks in the project. If other instruction profiles are already available you must explicitly activate the new one in order to use it as active profile. You can identify the active profile by its icon in the project navigation.

See also

Basics of instruction profiles (Page 6853)

Opening and editing instruction profiles (Page 6855)

Activating and deactivating instruction profiles (Page 6856)

Deleting instruction profiles (Page 6857)

5.9.3 Opening and editing instruction profiles

Once you have opened an instruction profile, you can edit it as follows:

- Activating and deactivating instructions
  You can explicitly specify the instructions to be allowed in the instruction profile.

  Note

  Note that dependencies exist between some instructions. As a result, it is possible that several instructions may be activated or deactivated by an action. The check box icon indicates the folders in which instructions are deactivated.

- Activating and deactivating instruction versions
  Certain instructions are available with different versions. You can explicitly specify the instruction versions to be allowed in the instruction profile.
- Renumbering blocks
  An instruction representing an internal function block (FB) or function (FC) in the system is assigned a specific block number by the system. You can replace this block number with your own block number. Within a version, there are several implementations for certain instructions. The block numbers in such instructions can only be changed for the specific implementation.

  Note

  If an instruction from the instruction profile is used in the program and the specified block number is already in use by a different block, the specified block number of the instruction will be replaced by a free block number.

Requirement

The "Common data > Instruction profiles" folder is open in the project navigation.

Opening instruction profiles

Proceed as follows to open an instruction profile:

1. Double-click the instruction profile that you want to edit.
   The instruction profile opens in the Instruction Profile Editor.

Editing instruction profiles

Proceed as follows to edit an instruction profile in the Instruction Profile Editor:

1. Select the device that you want to edit from the "Device family" drop-down list box.
2. Select the programming language for which you want to edit the instruction profile from the "Language" drop-down list box.
3. Deactivate the instructions or instruction versions that you want to exclude from the instruction profile. You can deactivate a folder to deactivate all subordinate instructions.

4. Activate the instructions or instruction versions that you want to allow in the instruction profile.
5. You may assign your own block numbers.

Note

You can assign the number up to 65535 for CPUs of the S7-1200/1500 series. For CPUs of the S7300/400 series you find the restrictions of the number ranges in the respective CPU manual.

Note

A new compilation process is required for all blocks in the project when you change the active profile.

See also

Basics of instruction profiles (Page 6853)

Creating new instruction profiles (Page 6853)

Activating and deactivating instruction profiles (Page 6856)

Deleting instruction profiles (Page 6857)

Basic information on instruction versions (Page 6850)

5.9.4 Activating and deactivating instruction profiles

You first need to activate an instruction profile in order to include filtering of its instructions. You can always deactivate the instruction profile to reset the instructions task card to the default scope of instructions.

Note

A new compilation process is required for all blocks in the project.

Requirement

The "Common data > Instruction profiles" folder is open in the project navigation.

Activating instruction profiles

Proceed as follows to activate an instruction profile:

1. Right-click on the instruction profile that you want to activate.
2. Select the "Activate profile" command from the shortcut menu.
   The selected instruction profile is now active. Instructions can now only be used in accordance with the settings of this profile.

Deactivating instruction profiles

Proceed as follows to deactivate the instruction profile:

1. Right-click on the instruction profile that you want to deactivate.
2. Select the "Deactivate profile" command from the shortcut menu.
   No instruction profile is active and the "Instructions" task card once again shows all instructions that are available for use.

See also

Basics of instruction profiles (Page 6853)

Creating new instruction profiles (Page 6853)

Opening and editing instruction profiles (Page 6855)

Deleting instruction profiles (Page 6857)

5.9.5 Deleting instruction profiles

Requirement

The "Common data > Instruction profiles" folder is open in the project tree.

Procedure

Proceed as follows to delete an instruction profile:

1. Right-click on the instruction profile that you want to delete.
2. Select the "Delete" command in the shortcut menu.

Note

A new compilation process is required for all blocks in the project when you delete the active profile.

Result

The selected instruction profile is deleted. If you deleted the active instruction profile, no more active profiles are available and the "Instructions" task card once again shows all instructions that are available for use.

See also

Basics of instruction profiles (Page 6853)

Creating new instruction profiles (Page 6853)

Opening and editing instruction profiles (Page 6855)
Activating and deactivating instruction profiles (Page 6856)

5.10 Using autocompletion

5.10.1 Basics of autocompletion

Function

You can use autocompletion in the program window of the program editor as an easy way to access available tags or instructions during programming. Autocompletion means a context-specific list appears in a dialog from which you can select the tags or instructions you need.

See also

Using autocompletion in graphic programming languages (Page 6859)

Using autocompletion in textual programming languages (Page 6860)

5.10.2 Using autocompletion in graphic programming languages

Inserting tags using autocompletion

To insert tags in graphic programming languages using autocompletion, follow these steps:

1. Select an operand of the instruction to which you wish to assign a tag.
   The input field for the operand opens. The autocompletion button will appear beside the input field.
2. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the local and global tags, data blocks and multiple instances which are admissible for the operand in the given context. You can exit autocompletion at any time by pressing <Esc>.
3. Select the required tag from the list. If necessary, you can also filter the list:
   - For example, enter the first few letters of the name of the tag or instruction you wish to insert. Autocompletion will be filtered further with each letter entered. If there is no tag or instruction starting with the letters entered, autocompletion will remain at the last match.
   - Enter # to access the local tags from the block interface.
   - Enter " to access the global tags.
   - Enter % to access absolute addresses.

   If the tag is a structured tag, a data block or a multiple instance, then an arrow is displayed at the end of the row. Click on the arrow to display the lower-level elements. You can navigate to the very last level in this way. If a structure is allowed as a data type for the operand, you can choose "None" from the list. This assigns the entire structure to the operand as a tag. Use the <Backspace> key to return to the previous level.
4. Press the <Return> key to apply the tag.

See also

Basics of autocompletion (Page 6859)

Using autocompletion in textual programming languages (Page 6860)

5.10.3 Using autocompletion in textual programming languages

Inserting tags and instructions using autocompletion

To insert tags and instructions in textual programming languages using autocompletion, follow these steps:

1. Enter the first few letters of the name of the tag or instruction you wish to insert. If necessary, you can directly filter the kind of tags:
   - Enter # to access the local tags from the block interface.
   - Enter " to access the global tags.
   - Enter % to access absolute addresses.

   Autocompletion opens. It contains only the local and global tags, data blocks, multiple instances and instructions which are admissible at the current position. You can exit autocompletion at any time by pressing <Esc>.
2. Enter more letters of the name of the tag or instruction you wish to insert. You can use <Enter> or <Tab> to apply the tag or instruction and close autocompletion.
   Autocompletion will be filtered further with each letter entered. If there is no tag or instruction starting with the letters entered, autocompletion only contains the previous matches.
3. Select the tag or instruction required from the list.
   If a tag is a structured tag, a data block or a multiple instance, first select the tag, the data block or multiple instance from the autocompletion and apply the selection with <Enter>.
   To select the additional components of the structure, data block, or multiple instance, enter a period. Autocompletion then reopens and you can select the next component.
4. Press the <Return> key to apply the tag.

See also

Basics of autocompletion (Page 6859)

Using autocompletion in graphic programming languages (Page 6859)

5.11 General settings for the PLC programming

5.11.1 Overview of the general settings

Overview

The following table shows the general settings that you can make:

Group	Setting	Description
View	Operand representation	Representation of the operand in the program editor. You can select between the following options: • Symbolic and absolute • Symbolic • Absolute (only for LAD, FBD, STL and GRAPH)
	Tag information	Additional information for the tags used is displayed in the program editor. When you select the option "Tag information with hierarchy", the comments of the higher structure levels are also displayed for structured tags.
	Location of tag information (LAD/FBD)	Specifies the position at which the tag information is to be displayed. You can display the tag information either collectively below a LAD/FBD network or directly at the respective operand.
	with network comments	Network comments are shown.
Compilation	Delete actual parameters on interface update	Actual parameters are deleted if the associated formal parameters were deleted in the called block, and you run the "Update block call" function or compile the block.

Group	Setting	Description
Default settings for new blocks	IEC check for code blocks	The compatibility of operands in comparison operations and arithmetic operations are tested according to IEC rules. You have to explicitly convert non-compatible operands. See also: Overview of the valid data types (Page 231)
	Set "DB Accessible from OPC UA" for new data blocks	When the check box is selected, the attribute "DB Accessible from OPC UA" is set for new global data blocks. The data block is thus accessible as complete object from OPC UA. The individual tags of the data block can then be individually released or locked for OPC UA. See also: AUTOHOTSPOT
	Set instance data block elements with access type "Standard" to retentive.	When this check box is selected, the "Retain" attribute is set for newly created tags. The setting influences the following types of data blocks: • S7-1200/S7-1500: Instance data blocks without optimized access – the setting influences all data block tags. • S7-1200/S7-1500: Instance data blocks with optimized access – the setting influences all tags with the attribute "Retentivity: set in IDB". **Note** With other types of data blocks you cannot specify any default for the retentivity. • S7-300/S7-400: Tags in instance data blocks and global data blocks are retentive by default. If you change the retentivity setting, the change affects all tags in the block. • S7-1200/S7-1500: Tags in instance data blocks with optimized access are non-retentive by default. You can set the retentivity for each tag individually. See also: Setting retentivity (Page 7037)

Group	Setting	Description
Additional settings	Set network title automatically	Sets the title of a network based on the comment of the output parameter of the first writing instruction in the network. See also: Inserting network title (Page 7131)
	Show autocompletion list	The autocompletion list is displayed.
	Mnemonics	German or international designation of operations and operands
Download without reinitialization	Memory reserve	Defines the size of reserve in volatile memory that can be used for interface extensions. See also: Downloading blocks for S7-1200/1500 (Page 7544)
Block interface / data block elements	Set "Accessible from HMI/OPC UA" for new elements and ARRAY data block elements.	Enables the option "Accessible from HMI" for new tags in the block interface and in data blocks. The use of this option makes sense especially when working with large amounts of data in ARRAY data blocks. See also: Basic principles for programming of data blocks (Page 7019)
	Set "Writable from HMI/OPC UA" for new elements and ARRAY data block elements.	When the check box is selected, the attribute "Writable from HMI/OPC UA" is set for new elements in the block interface and in data blocks. The use of this option makes sense especially when working with large amounts of data in ARRAY data blocks. See also: Basic principles for programming of data blocks (Page 7019)

See also

Overview of the block interface (Page 6965)

Changing the settings (Page 6864)

Permissible addresses and data types of PLC tags (Page 7073)

Basics of block access (Page 55)

Activate or deactivate IEC check (Page 491)

Using autocompletion (Page 6859)

5.11.2 Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the general settings (Page 6861)

5.12 Eliminating syntax errors in the program

5.12.1 Basic information on syntax errors

Syntax errors

Below are some examples of syntax errors:

- Missing separators or the use of too many separators
- Incorrect keyword spelling
- Incorrect jump label spelling/notation
- Notation which does not match the set mnemonics (for example, "I2.3" instead of "E2.3")
- The use of key words as operands

Identification of syntax errors

Syntax errors are underlined in red or appear in red type.

This identification allows you to recognise incorrect inputs at a glance and jump from error to error to eliminate them. Syntax errors are also listed in the "Info" tab of the inspector window with an error message.

See also

Finding syntax errors in the program (Page 6865)

5.12.2 Finding syntax errors in the program

Procedure

To find syntax errors in the program, follow these steps:

1. Select the position in the program in which you wish to look for errors.
2. Click "Go to next error" in the toolbar.
   The first error after the position you have selected will be marked.

You can use "Go to next error" and "Go to previous error" in the toolbar to find and correct all errors in the block.

Or:

1. Open the error list in the inspector window with "Info > Syntax".
   All syntax errors are listed in the table with a short description of the error.
2. If there are any errors, click on the blue question mark next to the error text to obtain information on eliminating the problem.
3. Double-click the error you want to correct.
   The corresponding error is highlighted.

See also

Basic information on syntax errors (Page 6865)

5.13 Changing the programming language

5.13.1 Rules for changing the programming language

Rules

Observe the following rules if you want to change the programming language for a block:

- All CPU series:
  - You can only change the programming language of entire blocks. The programming language cannot be changed for individual networks.
  - You cannot switch blocks programmed in the programming languages SCL or GRAPH. In GRAPH blocks, however, you can change between LAD and FBD as network languages.
- S7-300/400:
  - You can only change between the programming languages LAD, FBD and STL.
  - You can create networks within a block using another programming language and then copy them into the desired block.
  - If the language of individual networks of the block cannot be changed, these networks is displayed in their original language.
- S7-1200/1500:
  - You can change between the programming languages LAD and FBD.
- S7-1500:
  - You can create STL networks within the LAD and FBD blocks. However, you cannot copy between STL and LAD/FBD.

5.13.2 Change the programming language

Procedure

To change the programming language, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.
3. Select the "General" entry in the area navigation.
4. Select the new programming language in the "Language" drop-down list.
5. Confirm your selection with "OK".

See also

Rules for changing the programming language (Page 6867)

Use software units (S7-1500) 6

6.1 Basics on software units (S7-1500)

6.1.1 Introduction to software units (S7-1500)

Programming with software units

With the help of software units, you can subdivide your user program into individual program units, which you can edit and download independent of each other. For this purpose a new "Software Units" folder is provided in the project tree in which you can create and program your software units.

The following figure shows the "Software units" folder in the project tree:

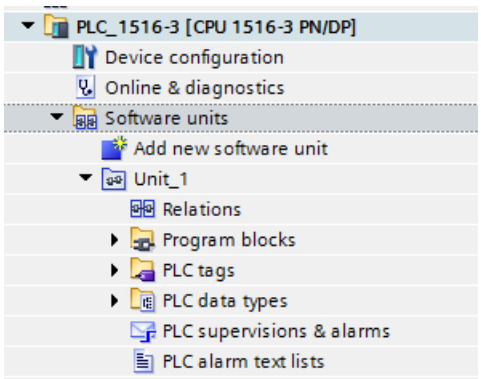

Each software unit contains the following main elements:

- Relations: By means of the relational table, you can set up access from your software unit to the following objects:
  - Published blocks and PLC data types of other software units
  - Global data blocks outside of the software unit
  - Technology objects
- "Program blocks" folder: You create the program blocks (organization blocks, function blocks, functions, data blocks) in this folder.
- "PLC tags" folder: You create the PLC tag tables, PLC tags and user constants in this folder.
- "PLC data types" folder: You create the PLC data types in this folder.
- PLC supervisions and alarms: Opens the alarm and supervision editor.
- PLC alarm text lists: Opens the text lists editor.

Creation of the user program inside software units represents an extension of the existing programming options. This means you are free to choose which program sections you want to create inside software units and which you want to create outside of software units.

Note

Note the following information when using software units:

- You can use software units with all versions of the S7-1500 CPU with a firmware version V2.6 and higher.
- You can use up to 255 software units per CPU.
- The "Optimized block access" attribute is selected for all program blocks of a software unit and cannot be disabled.
- If you have created an online backup for a device that supports software units, you cannot use this backup for a device that does not support software units.
- The program information contains information about all blocks, regardless as to whether or not they were created in a software unit.
- The data blocks and tags from software units are also visible in a device proxy and can therefore be exported and imported. It is irrelevant whether the data blocks are published or not.
- OPC UA client interface: Configuration data blocks that are generated from OPC UA client interfaces by the system ("<Name of the client interface_Configuration>") are always created outside of the software units. If you want to access them you can create a relation to these client interfaces. To do so, use the "Data block outside of the software unit" option as relation type.

Program structure

Regardless of whether you want to split an existing program into software units or start a new project, you should plan the program structure at the start. A practical division of your user program into software units can be based on the following criteria, for example:

- Functional and technological units
  In software units, you can create functional or technological units that you can edit and load independently of each other. If the blocks of an existing program have already been divided into groups, these groups may result in useful software units.
- Expected frequency of changes
  Another possibility for splitting is the expected change frequency of program sections. To create units that can be loaded as independently of each other as possible, you should create your own software units for program sections with different change intervals. In one software unit, you can create blocks with basic functions that rarely need to be changed and include blocks that need to be changed frequently for commissioning in other software units. This division can also facilitate troubleshooting.

- Editor
  Splitting your program based on editors can significantly reduce the work required for coordination. In order to further optimize the cooperation of the project staff, you can use software units in conjunction with multiuser engineering.
- Widest possible encapsulation of the software units
  To enable you to edit and load the software units independently of one another, there should be no cross-relationships between the software units. Therefore, only use published program elements and relations if this is really necessary. Try to encapsulate the individual software units as far as possible.

You can also mix the individual criteria or create your own criteria for the division. However, you should consider the encapsulation of the software units as a main criterion in order to derive the greatest possible benefit from the software units.

See also: Possible program structures with software units (Page 6875)

Using software units for shared commissioning

By structuring your user program into software units, you also simplify the commissioning of the user program on a PLC by several users. Such structuring and the division into different areas of responsibility enables changes to different software units to be loaded into the PLC independently. The TIA Portal ensures that changes made by other users are not overwritten. In most cases, this eliminates the need to synchronize the user program via the PLC and accelerates the loading process to the PLC. Note the requirements for loading software units.

You can find additional information in the following sections.

- Downloading software units (Page 6900)
- Introduction to Multiuser Commissioning

Data access

You can create the following program elements in software units:

- Blocks (OBs, FBs, FCs and DBs)
- PLC tag tables and PLC tags
- PLC data types

To allow these program elements to be accessed outside the respective software unit, they need to be published. However, this is not possible for organization blocks and PLC tags. They can only be accessed in the software unit in which they are defined.

In observation and force tables, you can access all PLC data, regardless of its position or publication status.

To access the FBs, FCs, global DBs and PLC data types of other software units, these must be published and there must also be a relation to the respective software unit. You can also use relations to access global data blocks and technology objects that are located outside of software units.

The following figure shows the possible data accesses between programs inside and outside software units:

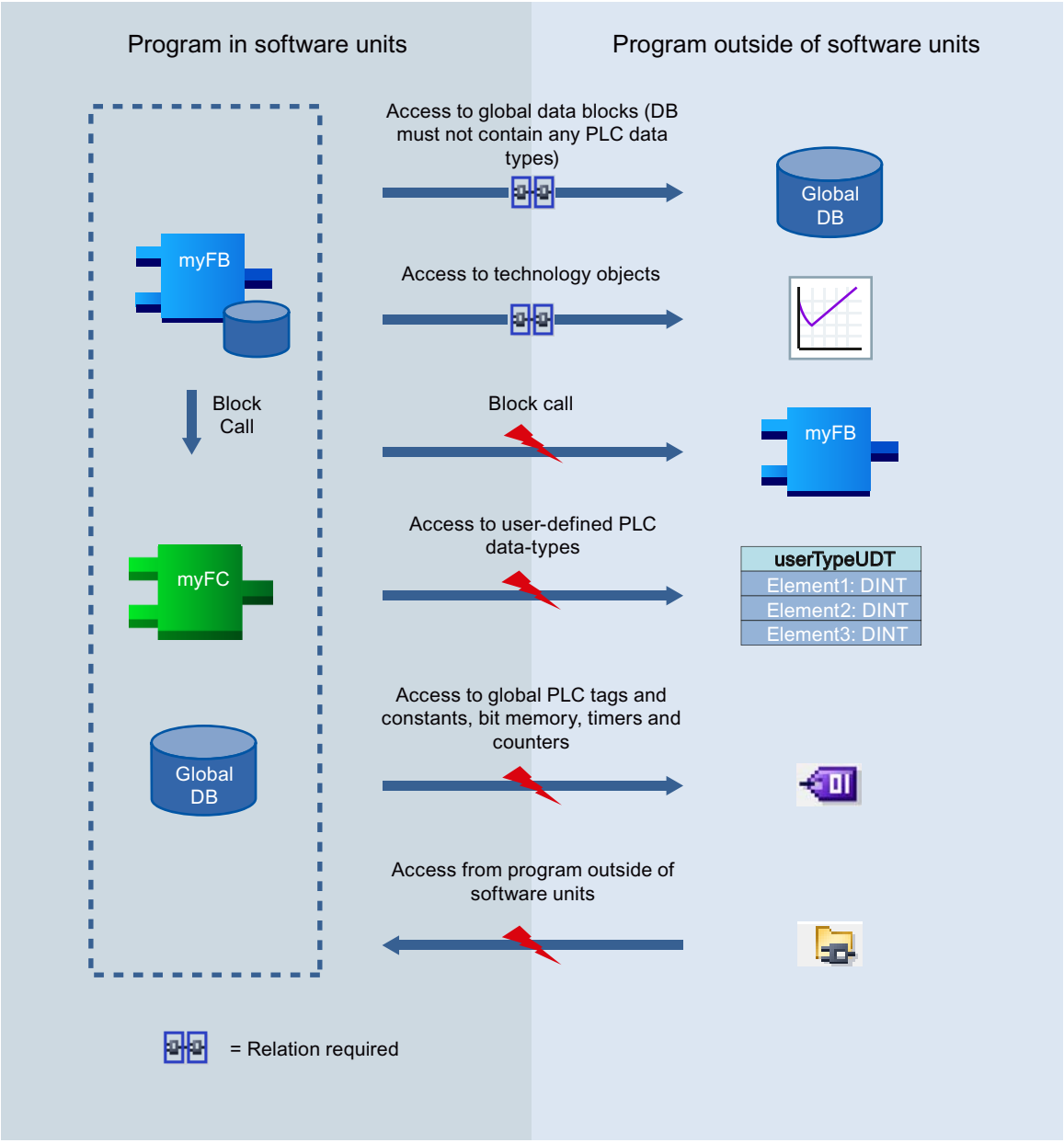

See also

Creating a PLC program in software units (Page 6873)

6.1.2 Creating a PLC program in software units (S7-1500)

The general programming concepts remain the same during program creation in software units. You create your user program in the programming editor in the usual manner. You have the same support as outside of software units, e.g. dragging-and-dropping instructions from the "Instructions" task card and auto-complete when entering data types and operand names.

Special considerations during program creation in software units

Note the following points when programming inside software units:

- The "Optimized block access" attribute is selected for all program blocks of a software unit and cannot be disabled. This also means that this attribute must be enabled when moving blocks from outside software units into a software unit. Therefore, do not use instructions with parameters that require absolute addressing, such as "PUT" or "GET".
- Program sections that you create outside a software unit have no access to the program sections you create inside a software unit. Blocks that you have programmed inside a software unit cannot access blocks outside of software units.
- However, you can selectively allow a software unit to access particular program elements outside of software units. Thus, for example, you can access global data blocks and technology objects that exist outside of software units.
- The program code inside the software units is part of the overall user program. That is why, for example, block names have to be unique CPU-wide.
- Inside of software units, you can only use symbolic programming.
- Inside of software units, you cannot use fail-safe blocks (F blocks).
- With the exception of organization blocks (OBs), all blocks are automatically numbered. The automatically assigned block numbers are not displayed, but you can see them in the properties of a block.
- The parameter types "TIMER", "COUNTER", "BLOCK_FB" and "BLOCK_FC" are not allowed in the block interface.
- Bit memory, the "Timer" function and the "Counter" function are not permitted in the PLC tag table. Using IEC timers and counters instead.
- You cannot use system and clock bit memories. Siemens Industry Online Support provides you with ready-made functions that you can use alternatively.
  See also: https://support.industry.siemens.com/cs/ww/en/view/109479728 (https://support.industry.siemens.com/cs/ww/en/view/109479728)
- Software units can be edited and loaded separately. If this results in blocks of the same software unit being changed by multiple editors, you have the option of synchronizing your data when loading.
- System constants are created in the unit in which the associated object is located. They cannot be published, which means that access from outside the respective software unit is not possible.

- When instructions that are implemented within the system as functions or function blocks are used, these system functions or system function blocks are created in the "System blocks" folder outside the software units. In contrast, the single instance of the data block of a system function block is created unpublished in the "System blocks" folder of the corresponding software unit. If you want to move the data block and access it, you may need to change the access to "Published" and create a relation to this data block.
- The use of software units does not change the access options for tags via HMI or OPC UA. You still define these options in the block interface for local tags or in the PLC tag table for global tags.

Programming languages

You can use the following programming languages in software units:

- LAD
- FBD
- SCL
- GRAPH
- ProDiag

There is no difference in using the programming languages LAD, FBD, SCL and GRAPH in software units and elsewhere. Note the following particularities for ProDiag when programming in software units:

- User-defined text lists that are referenced in supervision-specific texts can be stored in any software unit but not outside of software units.
- Every software unit has its own ProDiag OB to which a number >= 250 is assigned.
- The global tag supervisions for a ProdDiag FB can only supervise operands of the same software unit.
- A ProDiag FB only contains block supervisions that reference parameters from the same software unit.
- You cannot publish a ProDiag FB. Therefore, access from other software units is not possible.
- The initial value acquisition for a ProDiag FB only takes into consideration writing within the software unit in which the ProDiag FB is located at global supervisions and calls.

PLC alarm text lists

Each software units has a "PLC alarm text lists" entry in the project tree. This means you can create the corresponding text lists for each software unit. Text lists are always published, which means you can access them from other software units without having created relations beforehand. However, they cannot be accessed from outside of software units.

Only the own text lists are displayed in each software unit. To create or delete text lists, you must open the text list editor of the corresponding software unit or the CPU. Text list names have to be unique CPU-wide.

PLC supervisions and alarms

Each software unit contains its own ProDiag overview editor, which you can open in the project tree via the "PLC supervision & alarms" entry. You can add supervisions for a software unit in this editor. The supervisions that you create in one software unit are not displayed in other software units. The ProDiag overview editor outside software units, in contrast, displays all PLC supervisions, including supervisions within the software units.

You can export the created supervisions to an Excel spreadsheet and then import them back into the TIA Portal. If you export in the ProDiag overview editor of a software unit, only the supervisions of this software unit are exported. To export all supervision data from a PLC, start the export process in the ProDiag overview editor outside the software units. The Excel spreadsheet contains an additional column in which the software unit is specified.

See also:

- Displaying ProDiag supervisions (Page 7918)
- Exporting and importing ProDiag supervisions and properties of a ProDiag FB (Page 7928)

See also

Introduction to software units (Page 6869)

Possible program structures with software units (Page 6875)

6.1.3 Possible program structures with software units (S7-1500)

The following two simplified examples show how you can structure a user program in software units and thereby reduce cross-references between the individual software units as far as possible.

Both example programs control a production plant consisting of two stations. The stations perform various processing steps on workpieces and perform standardized tasks, e.g. adding up the produced quantities.

Program structure - Example 1

In the first example, the central unit "UnitMain" performs central control tasks. It contains an organizational block for cyclic program processing and a "LineControl" function block that coordinates the two stations.

The two stations are mapped in one software unit each. The program sections in the two software units are encapsulated so far that they can be processed by two different programmers.

The "UnitStandard" has the function of a library that provides general functionality. In the example it contains the FB "PartCount" to count quantities. This FB is instantiated once by each station. In practice, such a "standard software unit" may contain a large number of reusable program sections that can be called up whenever needed.

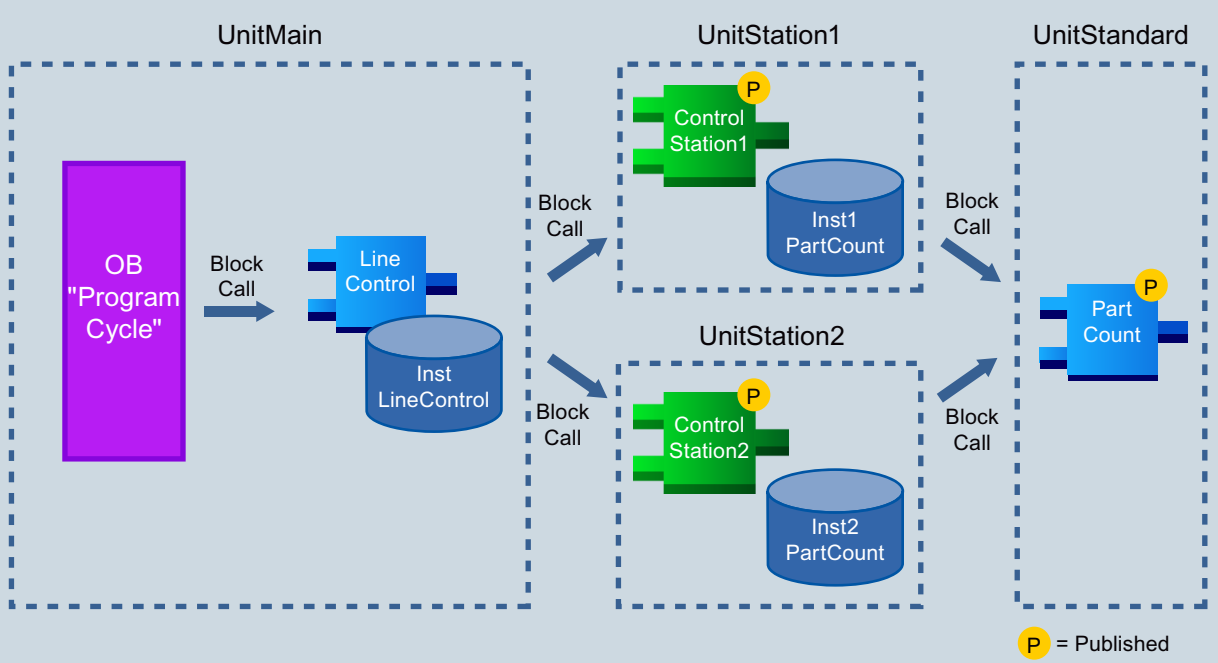

Required relations

Relation source	Relation target
UnitMain	UnitStation1
	UnitStation2
UnitStation1	UnitStandard
UnitStation2	

Program structure - Example 2

In the second example, each station is also controlled encapsulated in its own software unit. In the second example, however, each station has its own cyclic organizational block. If a program contains several organizational blocks, the operating system calls them one after the other in the order of their OB numbers.

As in the first example, there is also the "UnitStandard", which is used as a library for reused program sections. In addition to the "PartCount" FB, it also contains a PLC data type (UDT) that maps the production data of a workpiece. The PLC data type can be used in the interface of each program block and thus enables standardized storage of the workpiece data.

"UnitGlobalMachineData" is used for central data storage and processing. It contains a global data block in which each software unit can store its workpiece data.

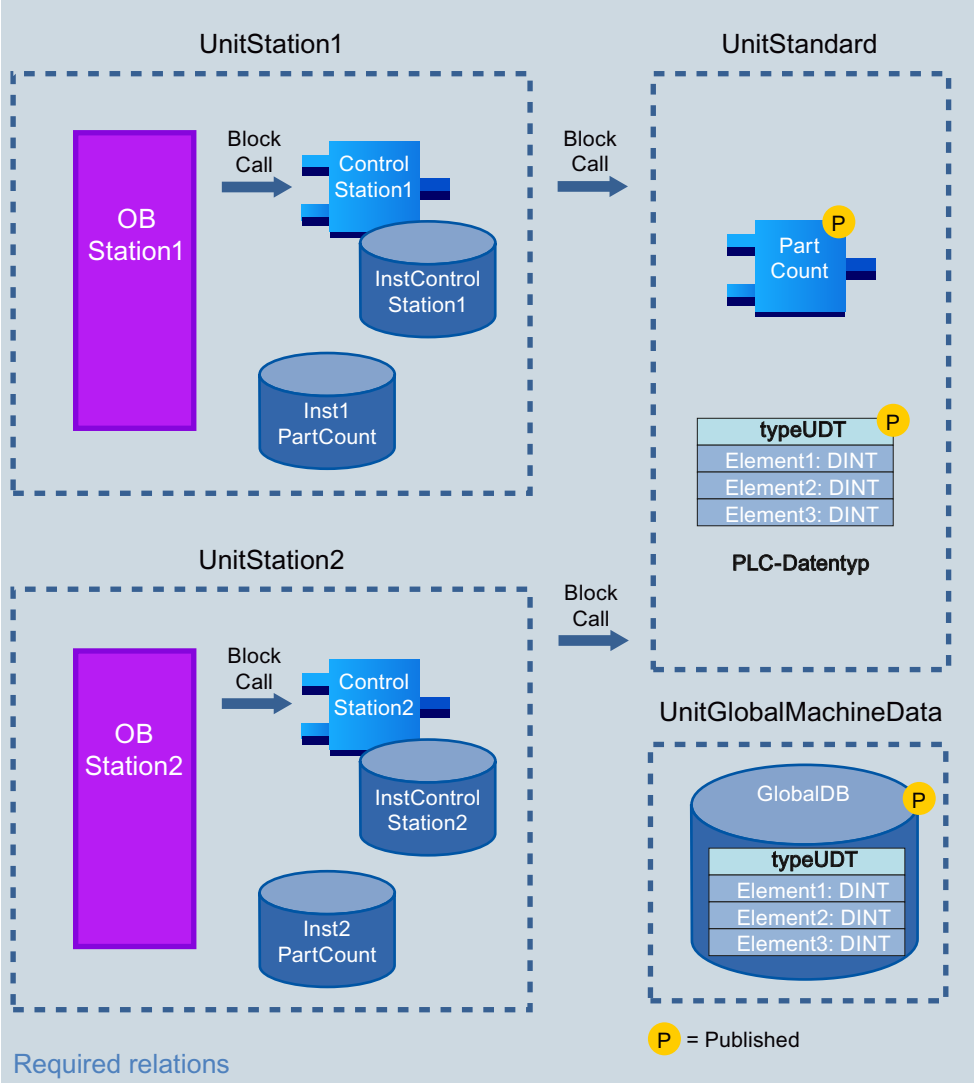

Required relations

Relation source	Relation target
UnitStation1	UnitStandard
	UnitGlobalMachineData
UnitStation2	UnitStandard
	UnitGlobalMachineData
UnitGlobalMachineData	UnitStandard

See also

Introduction to software units (Page 6869)

Creating a PLC program in software units (Page 6873)

6.2 Creating and managing software units (S7-1500)

6.2.1 Creating software units (S7-1500)

Requirement

The "Software Units" folder in the project tree is open.

Procedure

To create a new software unit, follow these steps:

1. Double-click the "Add new software unit" entry.
   The "Add new software unit" dialog opens.
2. Enter a name for the new software unit.
3. To enter additional properties for the new software unit, click "Additional information".
   An area with additional entry fields is displayed.
4. Enter all the desired properties.
5. Confirm your entries with "OK".

Result

The new software unit is created. You can now add additional program elements to the software unit and create the necessary relations.

See also

Introduction to software units (Page 6869)

Renaming software units (Page 6878)

Copying and pasting software units (Page 6879)

Deleting software units (Page 6880)

Displaying and modifying properties of a software unit (Page 6881)

6.2.2 Renaming software units (S7-1500)

Requirement

The "Software Units" folder in the project tree is open.

Procedure

To rename a software unit, follow these steps:

1. Right-click on the software unit you want to rename.
2. Select the "Rename" command in the shortcut menu.
   The name of the software unit in the project tree changes to an entry field.
3. Enter the new name of the software unit.
4. Confirm your entry with the Enter key.

Result

The name of the software unit is changed at all points of use in the program.

See also

Creating software units (Page 6878)

Copying and pasting software units (Page 6879)

Deleting software units (Page 6880)

Displaying and modifying properties of a software unit (Page 6881)

6.2.3 Copying and pasting software units (S7-1500)

You can also create new software units by copying existing software units and pasting the copy. You have the following options for this:

- Copying a software unit and pasting it within the same CPU
- Copying a software unit and pasting it into another CPU within the same project
- Copying a software unit of a reference project and pasting it into the open project
- Copying a software unit from a TIA Portal instance and pasting it into another TIA Portal instance

When software units with names identical to existing software units are pasted, the resulting name conflicts are handled as follows:

- Pasting the copied software unit into the same CPU:
  The copy of the software unit is assigned a name to which a number is added. For example, if software unit "SoftwareUnit_1" is copied, a possible name for the copy is "SoftwareUnit_2". A consecutive number range is not used for the number. Rather the smallest free number is always used. The copy of software unit "SoftwareUnit_1" can thus be assigned the name "SoftwareUnit_25", if a lower number is not free.
- Pasting the copied software unit into another CPU:
  A dialog box opens in which you can select whether the software unit with the same name is to be replaced or the copied software unit is to be pasted with a new name.

Copying software units

To copy a software unit, follow these steps:

1. Right-click on the software unit you want to copy.
2. Select the "Copy" command in the shortcut menu.
   A copy of the software unit is now on the clipboard and can be pasted into either the same CPU or another CPU.

Pasting software units

To paste a software unit, follow these steps:

1. In the project tree, open the folder structure for the CPU in which you want to paste the copied software unit.
2. Right-click on the "Software Units" folder.
3. Select the "Paste" command in the shortcut menu.
   - If you are pasting the software unit into the same CPU, the copy is pasted with the name extension "_<consecutive number>".
   - If you are pasting the software unit into another CPU and a software unit with this name already exists, the "Paste" dialog box opens. Select the desired option and confirm your selection with "OK".

See also

Creating software units (Page 6878)

Renaming software units (Page 6878)

Deleting software units (Page 6880)

Displaying and modifying properties of a software unit (Page 6881)

6.2.4 Deleting software units (S7-1500)

Requirement

The "Software Units" folder in the project tree is open.

Procedure

To delete a software unit, follow these steps:

1. In the "Software Units" folder in the project tree, right-click on the software unit that you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. Confirm the confirmation prompt with "Yes".
   The software unit is deleted from the project.

See also

Creating software units (Page 6878)
Renaming software units (Page 6878)
Copying and pasting software units (Page 6879)
Displaying and modifying properties of a software unit (Page 6881)

6.2.5 Displaying and modifying properties of a software unit (S7-1500)

You can display and modify the following properties for software units:

- Name
- Author
- Comment

Requirement

The "Software Units" folder in the project tree is open.

Procedure

To display and modify the properties of a software unit, follow these steps:

1. Right-click on the software unit whose properties you want to display or edit.
2. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the block opens and the properties are displayed.
3. Change the desired property as needed.
4. Confirm your entries with "OK".

See also

Creating software units (Page 6878)
Renaming software units (Page 6878)
Copying and pasting software units (Page 6879)
Deleting software units (Page 6880)

6.2.6 Managing software unit elements in libraries (S7-1500)

You can manage software units in libraries to reuse them, if necessary, together with their program elements. You can paste software units as master copies in the project library or in a global library.

You can choose from the following methods for creating master copies:

- You select one or more software units and generate separate master copies from them.
- You select multiple software units and create a single master copy from it that contains all selected elements.

You can also paste the blocks and PLC data types contained in software units independently of their software unit in the project library or in a global library. There is no difference when it comes to using blocks and PLC data types that were not programmed within software units. They can, for example, also be pasted as master copies or types in the project library or global libraries and then reused in the project. If the block or the PLC data type already exists in the project, you will receive a message informing you about it. In this message you can decide whether the existing object is moved to the selected folder or is to remain where it is.

Creating separate master copies from one or more software units

To create separate master copies from one or more software units, follow these steps:

1. Open the project library or a global library in the "Libraries" task card.
2. Select the desired software units in the project tree.
3. Using a drag-and-drop operation, move the software units to the "Master copies" folder or any subfolder of "Master copies".

Or:

1. Select the desired software units in the project tree.
2. Copy the software units to the clipboard and paste them to the "Master copies" folder or any subfolder of "Master copies".

The software units are pasted as separate master copies in the project library or global library.

Creating a single master copy from multiple software units

To create a single master copy from multiple software units, follow these steps:

1. Open the project library or a global library in the "Libraries" task card.
2. Copy to the clipboard the software units that you want to create as master copies.
3. Right-click on the "Master copies" folder or any of its subfolders in the library.
4. In the shortcut menu, select "Paste as a single master copy" command.

Or:

1. Select the desired software units in the project tree.
2. Press the <Alt> key and use a drag-and-drop operation to move the elements to the "Master copies" folder or any subfolder of "Master copies".

The elements are pasted in the library as a single master copy. The single master copy contains all selected elements.

6.3 Inserting program elements (S7-1500)

6.3.1 Creating blocks (S7-1500)

You can use the following block types inside a software unit:

- Organization blocks (OB)
- Functions (FC)
- Function blocks (FB)
- Data blocks (DB)

You can use the following programming languages to program the blocks: LAD, FBD, SCL, GRAPH and PRODIAG.

The numbers for FBs, FCs and DBs are always automatically assigned and cannot be changed. On the other hand, you can assign the block numbers for OBs yourself based on the available number range in each case. In a software unit, the block numbers are not displayed in the project tree. However, you can open the properties of a block to display its number.

Note that all restrictions for organization blocks are also retained in software units. This means that you can only use the maximum number of an organization block type, regardless as to whether the organization blocks exist inside or outside of software units.

If you want to access a block of another software unit, the "Published" option must be selected for this block under "Access". Note the following particularities for this:

- Organization blocks:
  The "Access" property is not available for organization blocks. They cannot be accessed from another software unit.
- ProDiag function blocks:
  The "Published" option is always deselected for ProDiag function blocks. This means they also cannot be accessed from another software unit.

Requirement

The "Software Units" folder in the project tree is open.

Procedure

To create a new block in a software unit, follow these steps:

1. Open the software unit to which you want to add a new block.
2. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
3. Click the corresponding button for the desired block type.
4. Enter a name for the new block.
5. Select the language for the new block.

6. Select the "Published" check box if you want to access the new block from other software units.
7. To enter additional properties of the new block, click "Additional information".
   An area with additional entry fields is displayed.
8. Enter all the desired properties.
9. Select the "Add new and open" check box if the block is to be opened as soon as it is created.
10. Confirm your entries with "OK".

Result

The new block is created. You can find the block in the project tree inside the corresponding software unit in the "Program blocks" folder.

See also

Declaring PLC tags (Page 6884)

Creating PLC data types (Page 6886)

6.3.2 Declaring PLC tags (S7-1500)

You can work with PLC tags and global constants inside software units in the same way as outside of software units. Note the following particularities:

- The "Show all tags" tag table is not available inside a software unit.
- The "Show all tags" tag table outside of software units also shows the tags and global constants of the software units. The tags and global constants of the software units can also be edited here.
- Tags and global constants of a software unit can only be used inside this software unit. Access to the tags and global constants of a software unit from outside this software unit is not allowed. Access via HMI and OPC UA is not restricted by this, however.
- S5 timers and s5 counters are not permitted as data types for operands.

- The use of bit memory is not allowed.
- You can also export PLC tag tables to software units for further processing with external applications. You can also import PLC tag tables that have been processed externally into software units. The following formats are supported for import and export:
  - XLSX (Excel)
  - XML
  - SDF

  In "XLSX" format, an additional column is created in which the software unit from which the PLC tag table was exported is specified. The other two formats do not contain any information about the exporting software unit. Note that you can only import PLC tags from an Excel file with the additional column for the software unit into other software units, but not into PLC tag tables outside software units.

Creating a PLC tag table

To create a new PLC tag table inside a software unit, follow these steps:

1. In the project tree, open the "PLC tags" folder inside the software unit to which you want to add a PLC tag table.
2. Double-click the "Add new tag table" entry.
   A new PLC tag table with the default name "TagTable_x" is created.
3. Select the PLC tag table in the project tree.
4. Select the "Rename" command in the shortcut menu.
5. Enter a name that is unique CPU-wide.
   A new PLC tag table is created. You can declare tag and constants in this table.

Declaring tags

To declare PLC tags inside a software unit, follow these steps:

1. Open a PLC tag table that is located inside a software unit.
2. Open the "Tags" tab.
3. Enter a tag name in the "Name" column.
4. Enter the desired data type in the "Data type" column. You will be supported by autocompletion when entering the data type.
   An address corresponding to the data type is automatically completed.
5. Optional: Click on the arrow key in the "Address" column and enter an operand identifier, an operand type, an address and a bit number in the subsequent dialog.
6. Optional: Disable the options for HMI and OPC UA.
7. Optional: Enter a comment in the "Comments" column.
8. Repeat steps 3 to 7 for all the needed tags.

Declaring global constants

To declare global constants inside a software unit, follow these steps:

1. Open a PLC tag table that is located inside a software unit.
2. Open the "User constants" tab.
   The constants table opens.
3. Enter a constant name in the "Name" column.
4. Enter the desired data type in the "Data type" column. You will be supported by autocompletion when entering the data type.
5. In the "Value" column, enter a constant value that is permitted for the selected data type.
6. Optional: Enter a comment in the "Comments" column if you have comments about the constant.
7. Repeat steps 3 to 6 for all other constants you want to declare.

Result

The PLC tag table, the PLC tag or the global constant has been created and can be edited in the same way as is outside a software unit. Access, however, to the tags and constants is only possible within the own software unit. You can insert tags into a watch table or force table in order to supervise or force them.

See also

Creating blocks (Page 6883)

Creating PLC data types (Page 6886)

Declaring PLC tags (Page 7079)

Exporting and importing PLC tags (Page 7098)

6.3.3 Creating PLC data types (S7-1500)

You can also work with PLC data types inside a software unit. If you want to access them from other software units, you can set the "Published" property and set up a relation with the corresponding software unit. You cannot use this PLC data type outside of software units.

Procedure

1. In the project tree, open the "PLC data types" folder inside a software unit.
2. In the "PLC data types" folder, double-click the "Add new data type" command.
   The "Add new PLC data type" dialog opens.
3. Enter a name for the new PLC data type.
4. Select the "Published" check box if you want to access the new block from other software units.

5. To enter additional properties of the new PLC data type, click "Additional information". An area with additional entry fields is displayed.
6. Enter all the desired properties.
7. Select the "Add new and open" check box if the PLC data type is to be opened as soon as it is created.
8. Confirm your entries with "OK".

Result

The PLC data type has been created and you can edit it in the same way as outside of a software unit.

See also

Creating blocks (Page 6883)

Declaring PLC tags (Page 6884)

Creating PLC data types (Page 7108)

6.3.4 Inserting block calls (S7-1500)

You can use block calls inside as well as outside of software units. Note the following particularities:

- Within a software unit, you can call up blocks independent of their publishing status. This means that the called block does not have to be published.
- When block calls take place between two software units, there must be a relation between the software units and the called block has to be published. The calling block must have access to the called block and the corresponding instance data block. The instance data block must also have access to its function block. Make sure that there are no circular relations.
- Calls between blocks of software units and blocks outside software units are not possible.
- You cannot publish an instance data block when the associated function block is not published.

Inserting a call of a function (FC)

To insert a call of a function (FC), follow these steps:

1. Drag the function from the project tree to the required network.

Inserting a call of a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Drag the function block from the project tree into the program code.
   The "Call options" dialog opens.
2. In the dialog, enter whether you want to call the block as a single, multiple or parameter instance.
   - If you click the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block.
     If you call a block that contains supervisions, assign a ProDiag function block to the supervision functions in the "ProDiag FB" text box.
   - If you click the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
   - If you click the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
3. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You can then assign the parameters.

Note

If you specify an instance data block that does not exist when calling a function block, it is created but not published. You can publish it subsequently if required.

See also

Block calls (Page 60)

6.3.5 Copying and moving program elements (S7-1500)

You can copy or move the following program elements:

- Blocks
- PLC tag tables and tags
- PLC data types
- Text lists

The following copying and moving operations are possible:

- Copying or moving a program element from one software unit to another software unit
  All set properties of the program element remain unchanged.
- Copying or moving a program element from a software unit to an area outside of software units.
  The "Access" property is removed for blocks and PLC data types. Automatic numbering is set for blocks.
- Copying or moving a program element from an area outside of software units into a software unit.
  You can only insert and edit blocks in a software unit that are automatically numbered and for which the attribute "Optimized block access" is enabled. You can insert blocks that do not meet these requirements, but they are not supported and therefore cannot be edited. For supported blocks and PLC data types, the "Access" property is added and the "Published" check box is cleared.

Note that copying is always performed when dragging-and-dropping into a different CPU. The object is not deleted at the original location by this operation.

Copying or moving blocks

To copy a block, follow these steps:

1. Right-click the block that you want to copy.
2. Select the "Copy" command in the shortcut menu.
3. In the project tree, right-click the program block folder into which you want to insert the copied block.
4. Select the "Paste" command in the shortcut menu.
   - If you paste the block into the same CPU, the copy is pasted with the name extension "_<consecutive number>".
   - If you paste the block into a different CPU where a block of the same name already exists, the "Paste" dialog box opens. Select the desired option and confirm your selection with "OK".

Alternatively, you can also copy the block via drag-and-drop while holding down the <Ctrl> key.

To move a block, follow these steps:

1. Drag-and-drop the block to the new position.

Copying or moving PLC tag tables and tags

To copy PLC tag tables, follow these steps:

1. Select the PLC tag table you want to copy.
2. Select the "Copy" command in the shortcut menu.

3. Position the cursor at the location where you want to insert the PLC tag table.
   You can insert the PLC tag table into the same software unit or into a different software unit or outside a software unit into the "PLC tags" folder of a CPU.
4. Select the "Paste" command in the shortcut menu.
   - If you paste the PLC tag table into the same CPU, the copy is pasted with the name extension "_<consecutive number>".
   - If you paste the PLC tag table into a different CPU where a PLC tag table of the same name already exists, the "Paste" dialog box opens. Select the desired option and confirm your selection with "OK".

Alternatively, you can also copy the PLC tag table via drag-and-drop while holding down the <Ctrl> key.

To move tag tables, follow these steps:

1. Drag the PLC tag table to the new position.

To copy tags, follow these steps:

1. Select the tags that you want to copy.
   You can also select several tags by clicking them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking the first and last tag.
2. Select the "Copy" command in the shortcut menu.
3. Position the cursor at the location where you want to insert the tags.
   You can insert the tags into every PLC tag table, irrespective of whether these are in the same software unit or into a different software unit or outside a software unit.
4. Select the "Paste" command in the shortcut menu.
   If you paste the tags into the same PLC tag table or into a PLC tag table in which tags with the same name already exist, the copy is inserted with the extension "_<consecutive number>".

Alternatively, you can also copy the tags via drag-and-drop while holding down the <Ctrl> key.

To move tags, follow these steps:

1. Select the tag that you want to move.
   You can also select several tags by clicking them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking the first and last tag.
2. Select the "Cut" command in the shortcut menu.
3. Position the cursor at the location where you want to insert the tags.
   You can insert the tags into every PLC tag table, irrespective of whether these are in the same software unit or into a different software unit or outside a software unit.
4. Select the "Paste" command in the shortcut menu.
   If you paste the tags into the same PLC tag table or into a PLC tag table in which tags with the same name already exist, the copy is inserted with the extension "_<consecutive number>".

Alternatively, you can also move the tags using drag-and-drop.

Copying or moving PLC data types

To copy a PLC data type, follow these steps:

1. Right-click the PLC data type that you want to copy.
2. Select the "Copy" command in the shortcut menu.
3. In the project tree, right-click the "PLC data types" folder into which you want to insert the copied PLC data type.
4. Select the "Paste" command in the shortcut menu.
   - If you paste the PLC data type into the same CPU, the copy is pasted with the name extension "_<consecutive number>".
   - If you paste the PLC data type into a different CPU where a PLC data type of the same name already exists, the "Paste" dialog box opens. Select the desired option and confirm your selection with "OK".

Alternatively, you can also copy the PLC data type via drag-and-drop while holding down the <Ctrl> key.

To move a PLC data type, follow these steps:

1. Drag the PLC data type to the new position.

Copying or moving text lists

To copy text lists, follow these steps:

1. In the project tree, double-click the "PLC alarm text lists" entry for which you want to copy a text list in the CPU or software unit.
   All the text lists assigned to the CPU or software unit are displayed in the work area listed in a table.
2. Right-click the text list that you want to copy.
3. Select the "Copy" command in the shortcut menu.
4. Position the cursor at the location where you want to insert the text list.
5. Select the "Paste" command in the shortcut menu.
   - If you paste the text list into the same CPU, the copy is pasted with the name extension "_<consecutive number>".
   - If you paste the text list into a different CPU where a text list of the same name already exists, the "Paste" dialog box opens. Select the desired option and confirm your selection with "OK".

Alternatively, you can also copy the text list via drag-and-drop while holding down the <Ctrl> key.

Follow the steps below to move text lists:

1. In the CPU or software unit double-click the "PLC alarm text lists" entry in the project tree for which you want to move a text list.
   All the text lists assigned to the CPU or software unit are displayed in the work area listed in a table.
2. Drag the text list you want to move to the new position.

6.4 Creating relations (S7-1500)

6.4.1 Basics of relations (S7-1500)

Function of relations

With the help of software units, you can subdivide your user program into individual program units that you can edit, compile and download individually. You can use relations to create connections that allow you access to the following objects outside of the own software unit:

- Function blocks (FB), functions (FC)
- Global data blocks (DB)
- PLC data types
- Technology objects

Note the following particularities when using relations:

- A software unit cannot create a relation with itself.
- A relation can only be created in one direction at a time. This means that a software unit cannot create a relation to another software unit if a relation back from it already exists. Example: Software Unit A has a relation with Software Unit B. Software Unit B cannot establish an additional relation with Software Unit A and can also not access the program elements of Software Unit A.
- Relations that would lead to circular references cannot be created.

You can create, change and delete relations in the relational table.

Access to function blocks (FB) and functions (FC)

From a software unit, you can access the function blocks and functions of other software units if these are published and there is a relation to the software units. You can create a new data block as an instance of the function block for this of the other software unit in the accessing software unit. Alternatively, you can also publish the associated instance data block of an FB to access it.

It is not possible to access a function block or a function of a software unit from a block that was not created in a software unit. Likewise, a function block or a function inside a software unit cannot access a block outside of a software unit. Blocks outside of a software unit are the blocks that you have created directly in the "Program blocks" folder below the CPU.

System functions and system blocks are an exception; you can access them as usual.

Access to global data blocks (DB)

You have the following options to access global data blocks that are not located in the own software unit.

- You can access global data blocks outside of software units from a software unit if you have created a corresponding relation. The following rules apply to access of global data blocks:
  - Access to a global data block that contains PLC data types is not permitted.
  - Access to an unoptimized data block is not permitted.
  - Access to global ARRAY data blocks is not permitted.
  - Access to a global fail-safe data block (F-DB) is not permitted.
- The global data blocks of a software unit can be accessed from another software unit if the data blocks are published and there is a relation to the software units. However, access to the global data blocks of a software unit from blocks that are not stored in software units is not possible.

Access to PLC data types

You can access the PLC data types of other software units from a software unit if these are published and there is a relation to the software units. However, access to the PLC data types of a software unit from blocks that are not stored in software units is not possible.

Blocks and the PLC data types used by them in the block sections "Input", "Output", "InOut" and "Static" must have the same publication status. If a "Motor" block is published and uses the PLC data type "UDT1" of the same software unit, the PLC data type "UDT1" must also be published.

Access to technology objects

You can access technology objects from a software unit by creating a relation to the corresponding object in the software unit.

See also

Publish program elements (Page 6893)

Setting up relations (Page 6895)

6.4.2 Publish program elements (S7-1500)

These program elements must be published so that other software units can access the function blocks (FBs), functions (FCs), global data blocks (DBs) and PLC data types of a software unit. If this was not done when these elements were created, you can change the setting at any time. However, you can also cancel the publication again for an element. Access from outside is then no longer possible.

You cannot change the publication status for the following program elements:

- Organization blocks (OB)
- PLC tag tables
- ProDiag function blocks

These program elements are always unpublished. They cannot be accessed from outside their own software unit.

You can change the publication status for a program item at the following locations:

- In the properties of the program element
- In the overview window
- In the portal view

You see the publishing status for each program element in the detail view.

Changing the publication status in the properties of the program element

To change the publication status for a program element by its properties, follow these steps:

1. In the project tree, open the "Software Units" folder and navigate to the program element for which you want to change the publishing status.
2. Right-click on the program element.
3. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the program element opens.
4. Click the "General" group in the area navigation if it is not displayed.
5. Select or clear the "Published" check box below "Access".
6. Confirm your entries with "OK".
   The publication status for the selected program element is changed.

Changing the publishing status in the browser

To change the publishing status for a program element in the browser, follow these steps:

1. In the project tree, open the folder "Software Units" and navigate to the software unit of the program element with the publication status you want to change.
2. Select the "Program blocks" or "PLC data types" folder, depending on the type of program element.
3. Open the overview window using the "Maximizes/minimizes the overview" button or the "View > Overview" menu.
4. Open the "Details" tab.
   The publication status is displayed for each element in the "Published" column.
5. In the "Published" column, select or clear the check box of the program element you want to change.
6. Alternatively, you can select several program elements and change the publication status for all selected elements simultaneously using the shortcut menu command "Publication".

Changing the publishing status in the portal view

To change the publishing status for a program element in the portal view, follow these steps:

1. Change to the portal view.
2. Click on "PLC programming".
3. Click on "Show all objects".
4. Open the "Details" tab.
   The publication status is displayed for each element in the "Published" column.
5. In the "Published" column, select or clear the check box of the program element you want to change.

See also

Basics of relations (Page 6892)

Setting up relations (Page 6895)

6.4.3 Setting up relations (S7-1500)

You have the option of setting up relations with other software units, global data blocks and technology objects. Note that you can only access objects of another software unit for which the "Published" property is set. You can change or delete relations at any time.

Requirement

The "Software Units" folder in the project tree is open.

Creating relations

To create a new relation, follow these steps:

1. In the project tree, open the software unit for which you want to create a relation.
2. Double click on "Relations".
   The relational table opens.
3. Click "<Add new relation>".
4. Select the type of relation in the "Relation type" column.
5. In the "Accessible element" column, select or enter the element you want to access.
   The relation has been created and you can access the element.

Changing relations

To change a relation, follow these steps:

1. In the project tree, open the software unit for which you want to change a relation.
2. Double click on "Relations".
   The relational table opens.
3. In the "Accessible element" column, change the element you want to access as required.

Deleting relations

To delete a relation, follow these steps:

1. In the project tree, open the software unit for which you want to delete a relation.
2. Double click on "Relations".
   The relational table opens.
3. Select the relations that you want to delete.
4. In the shortcut menu, select the "Delete" command or press the <Del> key.
5. Confirm the confirmation prompt with "Yes".
   The selected relations are deleted.

See also

Basics of relations (Page 6892)

Publish program elements (Page 6893)

6.5 Compiling and downloading software units (S7-1500)

6.5.1 Basics for compiling and downloading software units (S7-1500)

Introduction

Software units are program units that you can compile and download independent of the remaining program code. If there are dependencies between software units, a corresponding message is displayed during compiling. After successful compilation, you can load software units into a device or onto a memory card. Loading can only be performed if the following requirements are met:

- The loading operation does not involve a program element that exists at a different position on the device or the memory card with the same name or the same number.
- If a software unit has relations to objects outside the software unit and these objects have been changed since their last download, these objects have to be included in the loading. This applies to published blocks of other software units, but also to global data blocks and technology objects outside of software units.
- The loading does not include a system block that already exists on the device or the memory card with a different number or version.
- The loading does not include any organization blocks with numbers that are used in the device or on the memory card for organization blocks of other types.
- If the loading includes an organization block with a specific start event and further software units with organization blocks with the same start event exist, these further software units also have to be included in the loading. This applies, for example, to the cycle OB and the startup OB.
- The maximum number of organization blocks of a specific type is not exceeded during the loading to the device or the memory card. If you still want to perform the loading, load the entire CPU into the device or onto the memory card. This restores the consistency between the project and device or memory card.
- Loading all the software units via the "Software Units" folder is only possible if the device or memory card does not contain any software units that are not in the project. Otherwise, this software unit will be deleted. If you still want to perform the loading, load the entire CPU into the device or onto the memory card. This restores the consistency between the project and device or memory card.
- All the PLC data types on the device or the memory card is also in the offline project with the same version.
- There is already a hardware configuration for the device in the device or on the memory card. If this is not the case, you must load the entire CPU.

If there are no dependencies to other program parts, the only software units included in the loading are those for which you have initiated the loading. This means that the rest of the

program may differ between the program elements in the project and on the device or memory card.

Note

Checksum for individually loaded software units

In order to check the identity or integrity of PLC programs, a checksum is generated during compiling of a CPU in the project. This is loaded as well into the device or onto the memory card. The "GetChecksum" instruction can be used to read this checksum during runtime in the program. If, by contrast, you load individual software units into a device or onto a memory card, the checksum is reset to "0x0000000000000000". The reason for this is that the checksum in the project no longer matches the checksum of the device. The next time you load the complete PLC program, you again receive a valid checksum. Be aware of this when using "GetChecksum".

Loading software units in "RUN" mode:

The same rules apply for the loading of software units in the "RUN" mode as when loading the different program elements outside of the software units. In the process, take into account the descriptions on the pages "Downloading blocks in the "RUN" mode to the device" of the various CPU series in the information system.

Software units in connection with OPC UA

If you wish to use OPC UA communication in connection with the software units, please note the following special points when compiling your user program:

- When the client interfaces are compiled, the relevant data blocks for OPC UA are created. However, these are stored outside of the software units. Therefore, move them to the desired software units with drag-and-drop.
- New PLC data types are also created in the "System data types" folder outside of the software units when the client interfaces are compiled. Move these also into the "PLC data types" folder of the corresponding software units.
- If you change the name of read, write or method lists, new PLC data types are created. You need to move these into the corresponding software units as well.
- If you create new nodes or change the names of existing nodes in the lists, the corresponding client interface needs to be compiled again for the names to be updated in the data blocks and PLC data types in the software units.

See also

Compiling software units (Page 6899)

Basics for compiling and downloading PLC programs (Page 7537)

6.5.2 Compiling software units (S7-1500)

The following possibilities are available to compile software units:

- Compiling the program elements of a device. This includes the program elements of the software units.
- Compiling the program elements of all the software units of a device.
- Compiling the program elements of individual software units.
- Compiling the blocks in the "Program blocks" folder of a software unit

Depending on which options you use, you get different options for the compilation process.

- Hardware and software (only changes)
- Hardware (only changes)
- Hardware (rebuild all)
- Software (only changes)
- Software (rebuild all)
- Software (reset memory reserve)

If a software unit accesses program elements of a different software unit, inconsistencies can arise during compiling. In this occurs, compile the dependent software unit as well.

Compiling the program elements of a device.

To compile the program elements of a device including the software units, follow these steps:

1. Select the device with the program elements you want to compile in the project tree.
2. Select the desired option in the "Compile" submenu of the shortcut menu.

Compiling the program elements of all the software units of a device

To compile the program elements of all the software units of a device, follow these steps:

1. Select the "Software Units" folder in the project tree.
2. Select the desired option in the "Compile" submenu of the shortcut menu.

Compiling the program elements of individual software units

To compile the program elements of individual software units, follow these steps:

1. In the "Software Units" folder in the project tree, select the software units whose program elements you want to compile.
2. Select the desired option in the "Compile" submenu of the shortcut menu.

Compiling the blocks in the "Program blocks" folder of a software unit

To compile the blocks of a software unit, follow these steps:

1. Select the blocks within a software unit that you want to compile in the "Program blocks" folder of the project tree.
2. Select the desired option in the "Compile" submenu of the shortcut menu.

Result

The program elements are compiled. You can check whether or not the compilation was successful in the Inspector window with "Info > Compile".

See also

Basics for compiling and downloading software units (Page 6897)

6.5.3 Downloading software units (S7-1500)

6.5.3.1 Loading software units to a device (S7-1500)

You can load one, multiple or all software units to a device in one load operation. In order to load all the software units at one time, the device may not contain any software unit that does not exist in the offline project. In this case, you can either load the entire CPU or individual software units.

Requirement

A hardware configuration exists on the device.

Loading one or more software units to a device

To load one or more software units to the device, follow these steps:

1. Open the "Software Units" folder in the project tree.
2. Select the software units you want to load.

3. Select one of the following commands in the shortcut menu of the "Download to device" submenu.
   - "Software (only changes)": This option only loads the selected software units. However, if changes have been made to other software units that result in inconsistencies with the selected units, these software units must also be loaded. In this case, the loading process cannot be performed and you must restart the loading with the option "Software (with software unit relations)".
   - "Software (with software unit relations)": This option loads the selected software units together with the software units to which there are relations. This also loads the changes in the software units connected by relations that do not result in inconsistencies with the selected units.

   If an online connection is already defined, the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays messages and proposes actions required for loading. If you have not yet established an online connection, the "Extended download" dialog opens. Then perform step 4.
4. Set all parameters required for the connection and click "Load".
   The "Load preview" dialog opens. This dialog displays messages and recommends actions needed for the loading operation.
5. Check the messages and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Load" button is enabled.
6. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the load without synchronization by clicking "Force download to device". If you have performed the suggested actions, you are asked whether you want to continue with the download. Click "Continue download" to download the software units. The "Load results" dialog then opens and shows you the status and the actions after the load operation.
7. If the download was not performed in "RUN" mode and you want to start the modules again immediately after the download, select the "Start module" entry in the "Start modules after download" drop-down list.
8. To close the "Load results" dialog box, click "Finish".

Loading all software units to a device

Proceed as follows to load all software units in one loading operation to the device:

1. Select the "Software Units" folder in the project tree.
2. Select the "Download to device > Software (only changes)" command from the shortcut menu.
   If an online connection is already defined, the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays messages and recommends actions needed for the loading operation. If you have not already established an online connection, the "Extended download to device" dialog opens. Then perform step 3.
3. Set all parameters required for the connection and click "Load".
   The "Load preview" dialog opens. This dialog displays messages and recommends actions needed for the loading operation.

4. Check the messages and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Load" button is enabled.
5. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the load without synchronization by clicking "Force download to device". If you have performed the suggested actions, you are asked whether you want to continue with the download. Click "Continue download" to download the software units. The "Load results" dialog then opens and shows you the status and the actions after the load operation.
6. If the download was not performed in "RUN" mode and you want to start the modules again immediately after the download, select the "Start module" entry in the "Start modules after download" drop-down list.
7. To close the "Load results" dialog box, click "Finish".

Result

The software units are loaded to the device. If the changes affect additional objects, these are compiled and also loaded to the device. Objects that only exist online in the device are deleted. Be aware that loading individual software units can result in inconsistencies between the user program in the device and the user program in the offline project.

The messages under "About > General" in the Inspector window show whether the load process was successful.

6.5.3.2 Loading software units from a device (S7-1500)

You can load either all or individual software units from a device or the software units together with all the project data. In order to load the software units together with the project data from a device, read the description on the "Uploading project data from a device" page in the information system.

Loading all software units from a device

To load all software units from a device, follow these steps:

1. Establish an online connection with the device from which you want to load the software units.
   See also: "Go online and Go offline" in the information system
2. In the project tree, select the "Software Units" folder from which you want to load software units.
3. In the "Online" menu, select the "Upload from device" command.
   The "Upload preview" dialog box opens. This dialog displays messages and recommends actions needed for the loading operation.
4. Check the messages and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Upload from device" button is enabled.
5. Click "Upload from device".
   The loading operation is performed.

Loading single software units from a device

To load individual software units from a device, follow these steps:

1. Establish an online connection with the device from which you want to load the software units.
   See also: "Go online and Go offline" in the information system
2. In the project tree, select the software units that you want to load from the device.
3. In the "Online" menu, select the "Upload from device" command.
   The "Upload preview" dialog box opens. This dialog displays messages and recommends actions needed for the loading operation.
4. Check the messages and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Upload from device" button is enabled.
5. Click "Upload from device".
   The loading operation is performed.

Result

The software units are loaded from the device to the project. You can edit and recompile them and load them to the device again.

6.5.3.3 Loading software units to a memory card (S7-1500)

You have the following options of loading project data together with software units to a memory card.

- Dragging project data to a memory card
- Writing project data to a memory card

Requirement

A memory card is displayed.

Procedure

To load software units together with other project data to a memory card, follow these steps:

1. In the project tree, drag the project data you want to load to the memory card. Please note that you cannot use the "Program blocks" folder as destination.
   If necessary, the project data is compiled. The "Load preview" dialog then opens. This dialog displays messages and recommends actions needed for the loading operation.
2. Check the messages and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Load" button is enabled.
3. Click "Load".
4. The loading operation is performed.

Or:

1. In the project navigation, select the project data that you want to load.
2. To do this, right-click the selection and select the "Copy" command from the shortcut menu. You can also use the key combination <Ctrl+C>.
3. Right-click the memory card and select the "Paste" command from the shortcut menu. You can also use the key combination <Ctrl+V>.
   If necessary, the project data is compiled. The "Load preview" dialog then opens. This dialog displays messages and recommends actions needed for the loading operation.
4. Check the messages and, where necessary, enable the actions in the "Action" column. As soon as loading becomes possible, the "Load" button is enabled.
5. Click "Load".
   The loading operation is performed.

Or:

1. In the project navigation, select the project data that you want to load.
2. Select the "Card Reader/USB memory > Write to memory card" command in the "Project" menu.
   The "Select memory card" dialog opens.
3. Select a memory card that is compatible with the CPU.
   A button with a green check mark is activated at the bottom of the dialog.
4. Click the button with the green check mark.
   If necessary, the project data is compiled. The "Load preview" dialog then opens. This dialog displays messages and recommends actions needed for the loading operation.
5. Check the messages and, where necessary, enable the actions in the "Action" column. As soon as loading becomes possible, the "Load" button is enabled.
6. Click "Load".
   The loading operation is performed.

6.5.3.4 Loading software units from a memory card (S7-1500)

To load software units together with other project data to a memory card, follow these steps: You have the following options for this:

- Load project data of the memory card as a new station
  With this option you upload the project data of a memory card to your project as a new station.
- Load project data of the memory card to an existing device
  With this option, you upload project data of a memory card to an existing device in your project.

Requirement

- A project is open.
- The memory card is displayed.

Loading as new station

To load project data from a memory card to your project, follow these steps:

1. In the project navigation, select the project data you want to upload.
2. Select "Upload device as new station (hardware and software)" in the "Online" menu.

Or:

1. In the project navigation, drag the memory card folder to the project.

Or:

1. Right-click the memory card.
2. Select the "Copy" command in the shortcut menu.
3. Right-click the project.
4. Select the "Paste" command in the shortcut menu.

Uploading to an existing device

To upload the project data of a memory card to an existing device, follow these steps:

1. In the project navigation, drag the folder of the memory card to a device in the project or copy the memory card and paste the data into a device.
   The "Upload preview" dialog box opens.
2. Check the messages in the "Upload preview" dialog, and select the necessary actions in the "Action" column.
   As soon as loading becomes possible, the "Upload from device" button is enabled.
3. Click "Upload from device".
   The loading operation is performed.

6.6 Comparing software units (S7-1500)

6.6.1 Basics for software unit comparison (S7-1500)

Offline/online and offline/offline comparison of software units

The comparison of software units is basically the same as the comparison of other project data. This means you also get a simple offline/online comparison, for example, when you create an online connection. During this process, comparable objects in the project tree are marked with icons that represent the result of the comparison. The following rules apply here:

- The software units are shown as identical when their properties and all lower-level objects are identical.
- The software units are shown as different when their properties are different.
- An exclamation mark is shown for a software unit when the lower-level objects are different but their properties are identical.

You can also run a more comprehensive offline/online and offline/offline comparison in the compare editor.

When you start an offline/online or offline/offline comparison on a device, the software units are integrated into the comparison. In the compare editor, the software units are shown in both the left and the right comparison table. In the "Property Comparison" section of the comparison editor under "Source Data > Software unit" you can see the checksums of the compared objects.

During the comparison, the software units are assigned to each other based on their name and their objects are compared with each other. Objects with the same name in different software units are not compared with each other. This means, for example, that when the block "Block_1" is contained in the software unit "Unit1" in the offline project and online in the software unit "Unit2" in the device, they are considered as different objects.

You can perform actions to synchronize objects that are not identical. When you specify the action for a software unit, it may happen that multiple objects with the same name have been created during synchronization. In an offline/online comparison, a selection dialog box appears in which you can decide whether the existing objects are to be deleted or renamed. If you make an offline/offline comparison, however, the action is aborted and a corresponding message is displayed.

See also

Basic information on comparing PLC programs (Page 7585)

Creating and managing blocks 7

7.1 Creating blocks

7.1.1 Block folder

Function

You can find a "Program blocks" folder in the project tree, in which you can create and manage the following blocks:

- Organization blocks (OB)
- Function blocks (FB)
- Functions (FCs)
- Data blocks (DB)

A "System blocks" subfolder containing another subfolder, "Program resources", is also created in the "Program blocks" folder the first time you drag an instruction to your program which is an internal system function block. The instance data block of the internal system function block is also pasted to the "Program resources" folder. You can copy, rename or delete such instance data blocks in the "Program resources" folder. The "Program resources" folder is deleted together with the "System blocks" folder if contains no more blocks.

A program cycle OB is automatically generated for each device and inserted in the "Program blocks" folder.

Note

Moving blocks from the "Program resources" folder or to the "Program resources" folder

You can move blocks between the "Program resources" folder and other folders. When moving blocks from the "Program resources" folder or to the "Program resources" folder, note the following:

- Moving blocks from the "Program resources" folder into another folder: When the next time the user program is compiled, the shifted instance data blocks are moved back into the "Program resources" folder.
- Moving by any blocks to the "Program blocks" folder: Blocks in the "Program resources" folder that are not required to run the user program are removed during the next compilation. This may cause your moved block to be deleted.

See also

Creating functions and function blocks (Page 6909)

Creating data blocks (Page 6910)

Creating organization blocks (Page 6908)

Using blocks from libraries (Page 6911)

7.1.2 Creating organization blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create an organization block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Organization block (OB)" button.
3. Select the type of new organization block.
4. Enter a name for the new organization block.
5. Enter the properties of the new organization block.
6. To enter additional properties for the new organization block, click "Additional information". An area with further input fields is displayed.
7. Enter all the properties you require.
8. Activate the "Add new and open" check box if the organization block does not open as soon as it is created.
9. Confirm your entries with "OK".

Result

The new organization block is created. You can find the organization block in the project tree in the "Program blocks" folder. You can assign additional parameters to some organization blocks in the inspector window or device view after they have been created. The organization block description will state whether the newly created organization block has additional parameters.

See also

Organization blocks (OB) (Page 39)

Block folder (Page 6907)

Creating functions and function blocks (Page 6909)

Creating data blocks (Page 6910)

Using blocks from libraries (Page 6911)

Entering a block title (Page 6917)

Entering a block comment (Page 6918)

7.1.3 Creating functions and function blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a function (FC) or a function block (FB), follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Function block (FB)" or "Function (FC)" button.
3. Enter a name for the new block.
4. Enter the properties of the new block.
5. To enter additional properties for the new block, click "Additional information".
   An area with further input fields is displayed.
6. Enter all the properties you require.
7. Activate the "Add new and open" check box if the block does not open as soon as it is created.
8. Confirm your entries with "OK".

Result

The new block is created. You can find the block in the project tree in the "Program blocks" folder.

See also

Function blocks (FB) (Page 41)

Functions (FCs) (Page 40)

Basics of block access (Page 55)

Block folder (Page 6907)

Creating organization blocks (Page 6908)

Creating data blocks (Page 6910)

Using blocks from libraries (Page 6911)

Entering a block title (Page 6917)

Entering a block comment (Page 6918)

7.1.4 Creating data blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a data block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Select the type of the data block. You have the following options available to you:
   – To create a global data block, select the list entry "Global DB".
   – To create an ARRAY data block, select the "ARRAY DB" entry in the list.
   – To create an instance data block, select the function block to which you want to assign the instance data block from the list. The list contains only the function blocks that were previously created for the CPU.
   – To create a data block based on a PLC data type, select the PLC data type from the list. The list contains only the PLC data types that were previously created for the CPU.
   – To create a data block based on a system data type, select the system data type from the list. The list contains only those system data types that have already been inserted to program blocks in the CPU.
4. Enter a name for the data block.
5. Enter the properties of the new data block.
6. If you have selected an ARRAY DB as the data block type, enter the ARRAY data type and the high limit for the ARRAY.
   You can change the high limit for the ARRAY at any time in the property window of the created block. The ARRAY data type cannot be changed subsequently.
7. If you have selected a block that contains monitoring as the "Type", assign a ProDiag function block to the monitoring functions.
8. To enter additional properties of the new data block, click "Additional information".
   An area with further input fields is displayed.
9. Enter all the properties you require.
10. Activate the "Add new and open" check box if the block does not open as soon as it is created.
11. Confirm your entry with "OK".

Result

The new data block is created. You can find the data block in the project tree in the "Program blocks" folder.

Note

Retentivity of ARRAY data blocks

ARRAY data blocks and their components cannot be set retentively.

See also

Basic principles for programming of data blocks (Page 7019)

Global data blocks (DB) (Page 42)

Instance data blocks (Page 53)

System data types (Page 349)

Using blocks from libraries (Page 6911)

7.1.5 Using blocks from libraries

You can save blocks in the project library or in a global library, so that you can use them more than once within a user program. You can insert the blocks as master copies or as types.

See also: Library basics

Requirement

- The "Libraries" task card is displayed.
- No write protection is set for global libraries.

Adding blocks as master copies to the project library or to a global library

To add new blocks as master copies to the project library or to a global library, follow these steps:

1. Maximize the project library or the global library.
2. Use drag-and-drop to move the block you wish to add to the library to the "Master copies" folder or any one of the "Master copies" subfolders in the project library or a global library. Do not release the left mouse button until a small plus sign appears underneath the mouse pointer.

Or:

1. Copy the block you want to add as master copy.
2. Maximize the project library or the global library.

3. Right-click the "Master copies" folder or any of its subfolders.
4. Select "Paste" in the shortcut menu.

Adding blocks as types to the project library or to a global library

To add new blocks as types to the project library or to a global library, follow these steps:

1. Maximize the project library.
2. Drag-and-drop the block you want to add as a type into the "Types" folder or any subfolder of "Types" in the project library. Do not release the left mouse button until a small plus sign appears underneath the mouse pointer.
3. If necessary, drag the block from the "Types" folder of the project library into the "Types" folder of a global library. Or you can copy the block in the project library, and add it to a global library.

Or:

1. Copy the block you want to add as type.
2. Maximize the project library.
3. Right-click the "Types" folder or any of its subfolders.
4. Select "Paste" in the shortcut menu.
5. If necessary, drag the block from the "Types" folder of the project library into the "Types" folder of a global library. Or you can copy the block in the project library, and add it to a global library.

Using blocks of the project library or a global library

To use a block from the project library or a global library in your project, follow these steps:

1. Maximize the project library or the global library so that you can see the block you wish to use.
2. Use a drag-and-drop operation to move the block to the CPU block folder. If the selected insertion points is not allowed, the mouse pointer will appear as a circle with a slash.

Note

If you derive an instance from a type in a global library, the type is also inserted into the project library. The instance is then only linked to the type in the project library.

7.1.6 Copying and pasting blocks

7.1.6.1 Basics of copying and pasting blocks

Function

You can also create new blocks by copying existing blocks and pasting the copy. Note the following principles when copying to CPUs of the same device family:

- You can copy organization blocks (OBs), functions (FCs), function blocks (FBs), and global data blocks (DBs) without restriction.
- You can copy instance data blocks only for the same function block, since the assignment to the function block cannot be changed afterwards. However, the assignment is canceled if you copy the instance data block to a different CPU. If a function block with the same name exists there, the instance data block will be assigned to this function block. If you copy the instance data block together with the function block into the other CPU, the instance data block is assigned to the copy of the function block.

The various device families sometimes support different blocks, especially in the case of organization blocks. However, function blocks and functions can also be programmed on the

various devices with different access types. Therefore, not all blocks are supported on all devices. Note the following principles when copying to a different device family:

- Copying to an S7-1200 CPU:
  - Organization blocks with "Optimized" access type can be copied to an S7-1200. If the copied OB type is supported by the S7-1200 CPU, the copied OB retains the properties of its event. You must, however, compile it again.
  - Although organization blocks with the "Standard" access type can be copied to an S7-1200, they are not supported by the CPU.
  - Function blocks (FBs), functions (FCs) and global data blocks (DBs) with "Optimized" access type can be copied to an S7-1200. However, they must be recompiled after this.
  - Instance data blocks: If there is a function block in the target CPU with the name that was assigned to the instance data block in the source CPU, the instance data block is assigned to the function block in the target CPU. If you copy the instance data block together with the function block to which it was assigned in the source CPU into the target CPU, the instance data block is assigned to the copy of the function block.
- Copying to an S7-1500 CPU:
  - Organization blocks with "Optimized" access type can be copied to an S7-1500. If the copied OB type is supported by the S7-1500 CPU, the copied OB retains the properties of its event. You must, however, compile it again. OB types that are not supported receive a no parking symbol.
  - Organization blocks with "Standard" access type can be copied to an S7-1500. If the OB derives from an S7-300/400 CPU, it receives the standard event of the corresponding OB type. If the OB derives from an S7-1200/1500 CPU, it receives the properties of its event. However, it must be compiled again.
  - Function blocks (FBs), functions (FCs) and global data blocks (DBs) with "Optimized" access type can be copied to an S7-1500. However, they must be recompiled after this.
  - Instance data blocks: If there is a function block in the target CPU with the name that was assigned to the instance data block in the source CPU, the instance data block is assigned to the function block in the target CPU. If you copy the instance data block together with the function block to which it was assigned in the source CPU into the target CPU, the instance data block is assigned to the copy of the function block.
- Copying to S7-300/400 CPUs:
  - Organization blocks can be copied as required between S7-300 and S7-400.
  - Although organization blocks from S7-1200/1500 CPUs can be copied to S7-300/400 CPUs, they are not supported by the target CPU.
  - Function blocks (FBs), functions (FCs) and global data blocks (DBs) can be copied as required between S7-300 and S7-400.
  - Although function blocks (FBs), functions (FCs) and global data blocks (DBs) can be copied from S7-1200/1500 CPUs to S7-300/400 CPUs, they are not supported by the target CPU.

- Instance data blocks: If there is a function block in the target CPU with the name that was assigned to the instance data block in the source CPU, the instance data block is assigned to the function block in the target CPU. If you copy the instance data block together with the function block to which it was assigned in the source CPU into the target CPU, the instance data block is assigned to the copy of the function block.
- You may be able to cop function blocks (FBs) and functions (FCs) that are programmed with LAD or FBD and contain SCL networks to S7-300/400 CPUs, but they are not supported by the target CPU.

In the project tree, blocks that are not supported are indicated by the no parking symbol. Blocks with a no parking symbol cannot be edited, but only used again as copy source.

Note

When blocks are copied between different device families, it is possible that the copied block needs to be recompiled. This applies also to the coping of blocks between CPUs and software controllers. If the block has know-how protection, re-compilation is only possible with the correct password.

Copying data

With paste, all the block data is copied and forwarded to the copy. This data includes:

- Block interface tags
- All networks
- Comments in all existing compilations
- Messages defined in the block
- The entire program code of the copied block including the call instructions contained in the block.
  However, the following objects are not copied:
  - Called blocks and associated instance data blocks
  - Global tags that are referenced in the copied block
  - The PLC data types used in the copied block for the declaration of the block parameters and local tags
  - The technology objects used in the copied block

Avoiding name conflicts during pasting

When pasting copied blocks with identical names to already existing blocks, the following mechanisms are used to avoid name conflicts:

- Pasting the copied block into the same CPU:
  The copy of the block gets a name that is extended by a number. For example, if block "A" is copied, a possible name for the copy is "A_1". Consecutive numbering is not used, but rather the smallest free number. The copy of block "A" can also get the name "A_25", if no lower number is available.
- Pasting the copied block into another CPU:
  A dialog box opens in which you can select whether the block with the same name will be replaced or the copied block will be pasted with a duplicate designation (Name_Number).

Note

Number conflicts

Number conflicts may occur, if the pasted block has the same block number as an existing block. The block number is not automatically changed during pasting. This double number may have an effect on block calls. When you copy blocks you should therefore check the block number carefully and correct duplicate block numbers manually or using the block properties. Duplicate block numbers lead to a compilation error.

See also

Copying blocks (Page 6916)

Pasting blocks (Page 6917)

7.1.6.2 Copying blocks

Requirement

The "Program blocks" folder is opened in the project tree.

Procedure

To copy a block, follow these steps:

1. Right-click the block that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

A copy of the block is now on the clipboard and can be pasted either into the same CPU or into another one.

See also

Basics of copying and pasting blocks (Page 6913)

Pasting blocks (Page 6917)

7.1.6.3 Pasting blocks

Requirement

You have copied a block.

Procedure

To paste a copied block and its data into a CPU, follow these steps:

1. In the project tree, open the folder structure for the CPU into which you want to paste the copied block.

   Note

   Please note that you can only paste the copied block into a CPU which supports the block programming language and type.

2. Right-click on the "Program blocks" folder.
3. Select "Paste" in the shortcut menu.
   - If you are pasting the block into the same CPU as the original block, "_<consecutive number>" will be appended to the name of the copy.
   - If you are pasting the block into a different CPU where a block of the same name already exists, the "Paste" dialog box opens. Select the required option and confirm with "OK".

See also

Basics of copying and pasting blocks (Page 6913)

Copying blocks (Page 6916)

7.1.7 Entering a block title

The block title is the title of the block. It is not the same thing as the block name, which was assigned when the block was created. The length of the block title is restricted to one row. You can enter the block title for open and closed blocks.

Requirement

A code block is available.

Enter block title for open block

To insert the block title in an open block, follow these steps:

1. Click on the title bar of the block in the program editor.
2. Enter the block title.

Enter block title for closed blocks

To insert the block title in a closed block, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.
3. Select the entry "Information" in the area navigation.
4. Enter the block title in the "Title" input field.
5. Confirm your entry with "OK".

See also

Creating organization blocks (Page 6908)

Creating functions and function blocks (Page 6909)

Entering a block comment (Page 6918)

7.1.8 Entering a block comment

You can use the block comment to document the entire code block. For example, you can indicate the purpose of the block or draw attention to special characteristics. You can enter the block comment for open and closed blocks.

Note

The block comment should not exceed 32,767 Unicode characters.

Requirement

A code block is available.

Enter block comment for open blocks

To insert a block comment in an open block, proceed as follows:

1. Click the small arrow in front of the block title.
   The right arrow becomes a down arrow, and the comment area is displayed.
2. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
3. Enter the block comment.

Enter block comments for closed blocks

To insert the block comment in a closed block, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.
3. Select the entry "Information" in the area navigation.
4. Enter the block comment in the "Comment" input field.
5. Confirm your entry with "OK".

See also

Creating organization blocks (Page 6908)

Creating functions and function blocks (Page 6909)

Entering a block title (Page 6917)

7.2 Specifying block properties

7.2.1 Basics of block properties

Block properties

Each block has certain properties, which you can display and edit. These properties are used amongst other things to:

- Identify the block
- Display the memory requirements and the compilation status of the block
- Display the time stamp
- Display the reference information
- Specify the access protection

See also

Overview of block properties (Page 6921)

Block time stamps (Page 6925)

Displaying and editing block properties (Page 6928)

Setting the mnemonics (Page 6843)

7.2.2 Overview of block properties

Overview

The properties of the blocks are block and CPU-specific. Not all properties are therefore available for all blocks or in all CPU families. The following table gives an overview of block properties:

Group	Property	Description
Properties	Name	Unique block name within the station
	Constant name	Name of the constant pasted for the OB in the PLC tag table
	Type	Block type (cannot be changed)
	Number	Block number
	Event class	Event class of an OB (cannot be changed)
	Language	Programming language of the block
	Language in networks	Language used to program conditions in GRAPH blocks.
	Version	Version of the GRAPH block
	Process image partition number	Display of the process image partitions that are assigned to the organization block (cannot be changed)
	ARRAY data type	Data type of an ARRAY data block (cannot be changed)
	ARRAY limit	High limit of an ARRAY data block The "Move operations" section of the "Instructions" task card offers options for addressing of ARRAY data blocks.
Information	Title	Block title
	Comment	Block comment
	Version	Version number of the block
	Family	Block family name
	Author	Name of the author, company name, department name, or other names
	User-defined ID	ID created by the user
Time stamp	Block	Times of creation and time of change of the block (cannot be changed)
	Interface	Time of creation of the block interface (cannot be changed)
	Code	Code modification time (non-editable)
	Data	Data modification time (non-editable)
	Load-relevant	Time of the last load-relevant modification (cannot be changed)

Group	Property	Description
Compilation	Status	Details of the last compilation run (cannot be changed)
	Lengths	Details of the block lengths (cannot be changed)
	Simulation	Indicates whether the block can be simulated with SIMATIC S7-PLCSIM Advanced for compiling. This requires the "Support simulation during block compilation" option to be selected in the project properties. See also: Displaying properties of the project
	Library compliance	Indicates whether the block in the library can be used with know-how protection.
Protection	Protection	Setting up know-how protection and copy protection for the block See also: Protecting blocks (Page 6943)

Group	Property	Description
Attributes	IEC check	The compatibility of the operands in comparison operations and arithmetic operations are tested according to IEC 61131. You have to explicitly convert non-compatible operands. See also: Overview of data type conversion (Page 489)
	Local error handling within block	Troubleshooting inside the block with the GET_ERROR or GET_ERR_ID instruction (cannot be changed). See also: Handling program execution errors (Page 136)
	Optimized block access	The tag declaration for blocks with optimized access contains only the symbolic names of the data elements. The system automatically optimizes and manages the addresses. The CPU performance increases and access errors, for example, from SIMATIC HMI, are prevented. See also: Blocks with optimized access (Page 55)
	Capable of multi-instance	The block can be called with the "Multi-instance" call option. This attribute cannot be changed.
	Create extended status information	Allows you to monitor all tags in an SCL block. This option does, however, increase the program memory space required and the execution times.
	Check ARRAY limits	Checks whether array indexes are within the range declared for an ARRAY during the runtime of an SCL block. The block enable output ENO is set to "0" if an array index is outside of the permitted range.
	Set ENO automatically/Set ENO automatically for SCL blocks and SCL networks	Checks whether errors occur in the processing of certain instructions during the runtime of an SCL block. The block enable output ENO is set to "0" if a runtime error occurs.
	Create minimized DB	Creates instance data blocks for GRAPH blocks of the S7-300 and S7-400 in minimized format. This option reduces the GRAPH FB memory space required, however you will only receive limited program status information.
	Skip steps	If the transitions before and after a step become valid simultaneously in a GRAPH block, the step will not be activated and will be skipped.
	Acknowledge supervision alarms	Any supervision error which occurs during the operation of a GRAPH block must be acknowledged before the program can continue.
	Permanent processing of all interlocks in manual operation	Permanently monitors all interlock conditions in a GRAPH block in manual mode.
	Initial value acquisition (Can only be activated as of block version V4.0)	The signal state of operands and the results of comparisons from a step or a transition are stored in the static parameters "CRIT_LOC", "CRIT_LOC_ERR", "CRIT" and "CRIT_FLT" of the GRAPH block interface.

Group	Property	Description
	Lock operating mode selection	Prevents a GRAPH block operating mode being selected.
	Name of the expansion block	Defines the extension block via which you can access the invisible parameters in the block interface of the GRAPH function block.
	Data block write-protected in the device	Indicates whether the data block is read-only in the target system, and cannot be overwritten while the program is running. The option is available only for data blocks.
	Only store in load memory	On activation the data block is stored only in the load memory, occupies no space in the work memory, and is not linked into the program. The Task Card "Instructions" in the "Extended Instructions > Data block functions" pane provide options for handling the data blocks. The option is available only for data blocks.
	DB Accessible from OPC UA	When the check box is selected the data block is accessible as complete object from OPC UA. The individual tags of the data block can then be individually released or locked for OPC UA. The option is available only for data blocks.
	Start information	Here you specify the structure of the start information of the OB for S7-1500 CPUs: either like for S7-300 and S7-400 CPUs or optimized start information.
	Priority	Shows the priority set for organization blocks. The CPU family used and the type of the organization block determine whether you can change the priority.
	Parameter input via registers	Enables parameter input by means of registers in an STL block of S7-1500. This feature allows you to use the "Conditional call of blocks" (CC) and "Unconditional call of blocks" (UC) instructions in the block.
	Enable readback	Enables you to mark individual parameters of the block for readback. The "Tag readback" function is relevant if the block is used in a CFC chart.
	Block representation	Specifies how the block is represented in a CFC chart. You can find or more information in the help for CFC on the page "Display variants of the block symbol in the CFC chart".
Time-of-day interrupt	Time-of-day interrupt	Parameters of the time-of-day interrupt OB: active (yes or no), execution, start date and time, local time or system time
Cyclic interrupt	Cyclic interrupt	Cycle time and phase shift of the cyclic interrupt OB
Triggers	Triggers	Display of start events of the hardware interrupt OB
Isochronous mode	Isochronous mode	Parameters of the isochronous mode interrupt OB: application cycles, automatic setting (yes or no), delay time. Also indicated is the PROFINET IO system or DP master system whose IO devices or DP slaves are assigned to the isochronous mode interrupt OB.

Group	Property	Description
Download without reinitialization	Memory reserve	Defines the reserve in volatile memory that can be used for interface extensions. The number of currently available bytes is displayed in parentheses. This information is updated with each compilation.
	Activate download without reinitialization for retentive tags	Enables definition of a reserve in retentive memory.
	Reserve in retentive memory	Defines the reserve in retentive memory that can be used for interface extensions. The number of currently available bytes is displayed in parentheses. This information is updated with each compilation.

See also

Basics of block properties (Page 6920)

Block time stamps (Page 6925)

Displaying and editing block properties (Page 6928)

Basics of block access (Page 55)

General settings for the PLC programming (Page 6861)

7.2.3 Block time stamps

Introduction

Blocks receive a number of different time stamps which show you when the block was created and when it was last changed. These time stamps are also used for automatic consistency checks before compilation.

Code block time stamps

The following time stamps are generated for code blocks (OBs, FBs, FCs):

- Block: Created on, Modified on
- Interface: Modified on
- Code/data: Modified on
- Load-relevant: Modified on

A time stamp conflict is displayed during compilation if the time stamp for the block calling is older than that of the interface for the block called.

Time stamps for code blocks are updated as follows:

- Block: The time stamp for the last block modification is always the same as the time stamp either of the interface or of the code depending on which area was modified last.
- Interface: The interface time stamp is updated each time the interface is modified. Even if you manually undo a change to the interface, for example change the name back, that is also a change which updates the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.
- Code/data: The code time stamp is updated each time the block code is changed. Even if you manually undo a change to the code, for example remove an instruction, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.
- Load-relevant: The time stamp for "Load-relevant" is updated each time a code block is changed. These changes can affect the code, the data or the interface.

Global data block time stamps

The following time stamps are generated for global data blocks:

- Block: Created on, Modified on
- Interface: Modified on
- Data: Modified on
- Load-relevant: Modified on

Time stamp conflict is reported during compilation of a global data block based on a PLC data type if the time stamp of the global data block is older than the time stamp of the PLC data type used.

The time stamps for global data blocks are updated as follows:

- Block: The time stamp for the last change to a global data block is always the same as the time stamp for the interface and the data.
- Interface and data: The interface time stamps and the data are updated each time the global data block is changed. Even if you manually undo a change, for example remove a tag, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamps will be reset to the value they had before the undone change.
- Load-relevant: The time stamp for "Load-relevant" is updated each time a global data block is changed. These changes can affect the data or the interface.

Instance data block time stamps

The following time stamps are generated for instance data blocks:

- Block: Created on, Modified on
- Interface: Modified on
- Data: Modified on
- Load-relevant: Modified on

A time stamp conflict will be reported during compilation of an instance data block if the interface time stamps of the instance data block are not identical to those of the function block.

The time stamps for instance data blocks are updated as follows:

- Block: The time stamp for the last change to an instance data block is always the same as the time stamp for the interface and the data.
- Interface and data: The interface time stamps and the data are updated each time the instance data block is changed. Even if you manually undo a change, for example cancel the tag retain setting, that is also a change which will update the time stamps. However, if you undo the change using the "Undo" function, the time stamps will be reset to the value they had before the undone change.
- Load-relevant: The time stamp for "Load-relevant" is updated each time an instance data block is changed. These changes can affect the data or the interface.

PLC data type time stamps

The following time stamps are generated for PLC data types:

- Block: Created on, Modified on
- Interface: Modified on
- Load-relevant: Modified on

The time stamps for PLC data types are updated as follows:

- Block: The time stamp for the last change to a PLC data type is always the same as the interface time stamp.
- Interface: The interface time stamp is updated each time the PLC data type is changed. Even if you manually undo a change, for example delete the content of a PLC data type, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.
- Load-relevant: The time stamp for "Load-relevant" is updated each time a PLC data type is changed.

See also

Basics of block properties (Page 6920)

Overview of block properties (Page 6921)

Displaying and editing block properties (Page 6928)

Basic information on compiling blocks (Page 7539)

7.2.4 Displaying and editing block properties

The properties of the blocks are block and CPU-specific. Not all properties are therefore available for all blocks or in all CPU families. Properties that can only be displayed are write-protected.

Displaying and editing properties of a closed block

To display and edit the properties of a closed block, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Right-click the block whose properties you want to display or edit.
3. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the block opens.
4. In the area navigation, click a group whose properties you want to display or edit.
5. Change the relevant property.
6. Confirm your entries with "OK".

Displaying and editing properties of an open block

To display or edit the properties of an open block, follow these steps:

1. Select the "Inspector window" check box in the "View" menu.
   The Inspector window opens.
2. Click the "Properties" tab.
   The properties of the block are shown in the "Properties" tab of the Inspector window.
3. In the area navigation, click a group whose properties you want to display or edit.
4. Change the relevant property.

Result

The properties of the block will be changed. The changes are not saved until the project is saved.

See also

Basics of block properties (Page 6920)

Overview of block properties (Page 6921)

Block time stamps (Page 6925)

7.3 Managing blocks

7.3.1 Opening blocks

You can open both the blocks in the project (offline blocks) and also the blocks in the device (online blocks).

To open an offline block, you have the following options:

- Open offline block directly
  You can open a block directly if the corresponding block folder is open in the project tree or in the overview window.
- Find and open offline block
  You can search for blocks within a project, a device, the "Program blocks" folder and in the Program editor, and then open these blocks.

Note the following special features when opening online blocks:

- You cannot edit online blocks.
- It is possible that another user can carry out a loading process on the selected CPU through joint parallel working on a CPU. Consequently, it can happen that the online block that you have open is deleted by the loading process if the block only exists in the device. In this case, the online block is closed and a message is displayed in the Inspector window.

Open offline block directly

To open an offline block directly, you have the following options:

1. Open the folder with the block you wish to open in the project tree or in the overview window.
2. Double-click on the block you wish to open.
   The block will open in the program editor.

Find and open offline block

To find an offline block and then open it, follow these steps:

1. You can start the search process for the following three types:
   - In the shortcut menu of the project, a device or "Program block" folder, select the "Open block/PLC data type" command or press the <F7> key.
   - Press the <F7> key in the open program editor.

   The "Open block/PLC data type" dialog box opens.
2. Enter the name, the address or the type of block you are looking for.
   The block list is filtered further with each letter entered. The block list is closed if there is no block that matches the input. You can show the complete block list at any time by clicking the button to the right of the text box. Keep in mind that there is no filtering in this case. If you want to filter for your inputs again, click the button once again.
3. In the block list, click the block you wish to open.
   The block is opened in the program editor and displayed as selection in the project tree.

Open online block

To open an online block directly, you have the following options:

1. Open the "Online access" folder in the project tree.
2. Click on the arrow symbol to the left of the interface to show all the objects that are arranged below the interface.
3. Double-click on the "Update accessible devices" command below the interface.
   All devices that are accessible via this interface are displayed.
4. Open the folder of the device that contains the block you want to open.
5. Open the "Program blocks" folder.
   All the device's blocks are displayed.
6. Double-click on the block you wish to open.
   The block will open in the program editor.

See also

Saving blocks (Page 6930)

Closing blocks (Page 6931)

Renaming blocks (Page 6931)

Deleting blocks offline (Page 6932)

Deleting blocks online (Page 6933)

Opening know-how protected blocks (Page 6960)

7.3.2 Saving blocks

Blocks are always saved together with the project. Faulty blocks can also be saved. This allows the error to be resolved at a convenient time.

Procedure

To save a block, follow these steps:

1. Select the "Save" or "Save as" command in the "Project" menu.
   See also: Saving projects

See also

Opening blocks (Page 6929)

Closing blocks (Page 6931)

Renaming blocks (Page 6931)

Deleting blocks offline (Page 6932)

Deleting blocks online (Page 6933)

7.3.3 Closing blocks

Procedure

To close a block, follow these steps:

1. Click the "Close" button in the title bar of the program editor.

Note

Note that the block will not be saved on closing.

See also

Opening blocks (Page 6929)
Saving blocks (Page 6930)
Renaming blocks (Page 6931)
Deleting blocks offline (Page 6932)
Deleting blocks online (Page 6933)

7.3.4 Renaming blocks

Requirement

The "Program blocks" folder is opened in the project tree.

Procedure

To change the name of a block, follow these steps:

1. Right-click the block that you want to rename.
2. Select the "Rename" command in the shortcut menu.
   The block name in the project tree changes to an input field.
3. Input the new name for the block.
4. Confirm your entry with the Enter key.

Result

The name of the block is now changed at all points of use in the program.

See also

Opening blocks (Page 6929)
Saving blocks (Page 6930)
Closing blocks (Page 6931)
Deleting blocks offline (Page 6932)
Deleting blocks online (Page 6933)

7.3.5 Deleting blocks offline

Requirement

The "Program blocks" folder opens in the project tree.

Procedure

To delete a block that exists offline, proceed as follows:

1. In the project tree in the "Program blocks" folder, right-click on the block that you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. Confirm the safety prompt with "Yes".
   The block is deleted offline from the project.

Note

If you are deleting organization blocks, note that events may be assigned to these blocks. If you delete such organization block the program cannot respond to parameterized events.

See also

Opening blocks (Page 6929)
Saving blocks (Page 6930)
Closing blocks (Page 6931)
Renaming blocks (Page 6931)
Deleting blocks online (Page 6933)
Downloading blocks for S7-1200/1500 (Page 7544)

7.3.6 Deleting blocks online (S7-300, S7-400)

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To delete a block that exists online, follow these steps:

1. In the "Program blocks" folder in the project tree, right-click on the block that you wish to delete from the device.
2. Select the "Delete" command from the shortcut menu.
   The "Delete" dialog opens.
3. Select the "Delete from device" check box.
4. Click "Yes".
   The block will be deleted from the device online.

See also

Opening blocks (Page 6929)

Saving blocks (Page 6930)

Closing blocks (Page 6931)

Renaming blocks (Page 6931)

Deleting blocks offline (Page 6932)

7.3.7 Deleting CPU data blocks

You may delete CPU data blocks both in offline and online mode.

Deleting CPU data blocks in offline mode

Proceed as follows to delete a CPU data block from the offline project:

1. Right-click the CPU data blocks that you want to delete in the project navigation, "Program blocks" folder.
2. Select the "Delete" command from the shortcut menu.
3. Confirm the safety prompt with "Yes".
   The CPU data block is deleted from the offline project.

Deleting CPU data blocks in online mode

Proceed as follows to delete a CPU data block from the online project:

1. Establish an online connection to the device containing the CPU data block that you want to delete.
2. Right-click the CPU data block that you want to delete from the device in the project navigation, "Program blocks" folder.
3. Select the "Delete" command from the shortcut menu.
   The "Delete" dialog opens.
4. Select the "Delete from device" check box.
5. Click "Yes".
   The CPU data block is deleted from the online device.

See also

CPU data blocks (Page 53)

7.4 Using external source files

7.4.1 Basics of using external source files

Function

The textual programming languages STL and SCL allow you to enter the program code in any ASCII editor and save it as an external source file. This enables you to perform a range of tasks, for example:

- Declaring tags
- Specify block properties
- Programming blocks

Note

For blocks to be generated from the external source file in the TIA Portal, the source file must be saved with one of the following character encodings:

- ANSI
- UTF-8 with BOM (Byte Order Mark)
- UTF-16 with BOM, little and big endian format
- UTF-32 with BOM, little and big endian format
- UCS2 in little and big endian format

You can import these source files to your project and use them to generate blocks. You can generate a number of different blocks from one source file. Observe the following special features when generating blocks from a source file:

- A block that exists under the same name in the project will be overwritten.
- If a block was programmed with its absolute block number instead of a symbolic name in the source file and this number is already assigned by a block in the project, the new generated block is initially assigned the next free symbolic name.
- If you have not explicitly defined the access mode for a block in the external source file, the block access mode is set depending on the CPU series used:
  - Blocks generated for a CPU of the S7-1200/1500 series are assigned "optimized" access mode by default.
  - Blocks generated for a CPU of the S7-300/400 product range are assigned "standard" access mode by default.

  Organization blocks are the exception in this case, as they are always assigned the "standard" access mode by default, regardless of the CPU series. You have the option of changing the block access mode manually.
- It is possible that not all comments from the source file will be applied in the block.

- If you use absolute addressing in the external source file, a symbolic tag is created for each absolute address during the generation of the block. The names of these tags are made up of "Tag_" and a time stamp. This may result in relatively long tag names, which you can change manually if required.
- If you are using instructions in a different version in the external source file than in the target device, this may result in compilation errors. Correct the respective instructions in this case and start the compilation process once again. You can also select a different version for the target device.
- CPUs of the S7-300/400 series accept organization blocks with any name during import. If you import an OB with a standard name (for example, CYCL_EXC or CYC_INT5), the correct OB class is automatically assigned. If you import an OB with an unknown name, an OB 0 is created. You can change this number after the import and hereby assign the desired OB class. To change the OB number, select "Properties" in the shortcut menu.
- No automatic checking of the source is made when and SCL source is imported. As a result, blocks generated from this may contain errors which are only displayed at the next compilation.

You also have the option of saving existing blocks as external source files. You can either write only the program code of the selected block to a source file or additionally the program code of the blocks and PLC data types dependent on the selected block. When saving several blocks to a source file, all blocks must use the same programming language and none of the blocks may have know-how protection.

See also

Rules for programming external source files (Page 6936)

Saving blocks as external source files (Page 6938)

Inserting external source files (Page 6939)

Opening and editing external source files (Page 6940)

Generating blocks from external source files (Page 6941)

7.4.2 Rules for programming external source files

An external source file basically consists of continuous text. To compile the source into blocks, certain structures and syntax rules must however be adhered to.

Syntax rules

The syntax of the instructions in external source files is very similar to that in the creation of user programs in the program editor with STL or SCL. Note, however, the following additional syntax rules:

- Block call
  When calling a block, transfer the parameters in the defined order in the ASCII editor. If you do not, the comment assignments for these lines may not match.
  Enter the parameters in brackets. The individual parameters are separated by a comma.
- Upper or lower case
  The program editor generally disregards upper or lower case. Jump labels are an exception to this. Character string entries are also case-sensitive ("STRING" data type). Keywords are displayed in upper case. For compilation purposes, however, case is disregarded; you can therefore specify keywords in upper or lower case or a mixture of the two.
- Semicolon
  Mark the end of every instruction and every tag declaration with a semicolon. You can enter several instructions per line.
- Forward slashes
  Begin every comment with two forward slashes (//) and end the comment with the <Enter> key.
- Use of String constants
  To avoid compilation errors when using String constants, enter the text in the language of the target project. You can explicitly use the data type WSTRING for CPUs of the S7-1200/1500 series by using the prefix "WString#":
  `Operand := WString#'<String constant>';`

See also

Basics of using external source files (Page 6935)

Saving blocks as external source files (Page 6938)

Inserting external source files (Page 6939)

Opening and editing external source files (Page 6940)

Generating blocks from external source files (Page 6941)

7.4.3 Saving blocks as external source files

You have the following options for saving STL and SCL blocks as external source files:

- Copying a block as text
  Note that with protected blocks only the information that you can also see without a password is copied to the source file.
- Generating external source file from blocks
  You can either write only the program code of the selected block to a source file or additionally the program code of all the blocks and PLC data types dependent on the selected block. All blocks must use the same programming language and none of the blocks may have know-how protection.

Note

When using the tabular block interface, note that not all initial values are exported to the source, e.g. for PLC data types.

Copying a block as text

To copy a block as text and save it to an external source file, follow these steps:

1. In the project tree, right-click on the block that you want to save in an external source file.
2. Select the "Copy as text" command in the shortcut menu.
3. Open an external text editor.
4. Paste the copied text from the clipboard.
5. Save the file with one of the following file name extensions:
   - ".scl", if you want to generate a source file for an SCL block
   - ".stl", if you want to generate a source file for an STL block
   - ".DB" if you wish to generate a source file for a data block
   - ".UDT", if you want to generate a source file for a PLC data type

Generating external source file from blocks

To generate external source files from STL or SCL blocks, follow these steps:

1. In the project tree or in the overview window, select the STL or SCL blocks from which you want to generate an external source file.

   Note

   Select only blocks with the same programming language and without know-how protection.

2. Select the "Generate source from blocks > Selected blocks only" command from the shortcut menu to save only the selected block to an external source file. Or select the command "Generate source from blocks > Including dependent blocks" if you also want to save the program code of the dependent blocks and referenced PLC data types to the external source file.
   The "Save as" dialog opens.

3. Specify a path and a name for the external source file.
4. Click "OK".

You can also generate an opened STL or SCL block as external source file. To do this, follow these steps:

1. In the program editor, select the "Generate source from this block" command in the toolbar in the drop-down list "Generate source from block" to save only the open block in an external source file. Or select the command "Generate source from this block and all dependent blocks" if you also want to save the program code of the dependent blocks and referenced PLC data types to the external source file.
   The "Save as" dialog opens.
2. Specify a path and a name for the external source file.
3. Click "OK".

Result

Depending on the selected option, either only the block or the block together with the dependent blocks and PLC data types are saved as an external source file. You can include this source file in a project in the TIA Portal and use it to generate other blocks. However, please note that you can use STL source files only in S7-300/400/1500 CPUs.

See also

Basics of using external source files (Page 6935)

Rules for programming external source files (Page 6936)

Inserting external source files (Page 6939)

Opening and editing external source files (Page 6940)

Generating blocks from external source files (Page 6941)

7.4.4 Inserting external source files

Requirement

- An external source file is available and complies with the syntax and structure rules.
- The "External source files" folder is open in the project tree.

Procedure

Follow these steps to insert an external source file:

1. Double-click on the "Add new external file" command.
   The "Open" dialog box opens.
2. Navigate to and select existing external source files.

3. Confirm your selection with "Open".
   If there is already an external source file with this name in the project, the "Add external file" dialog opens. In this case, continue with step 4.
4. Select whether the new external source is to be renamed and inserted or whether the existing external source to be replaced.
5. Click "OK".

Result

The new source file will be added to the "External source files" folder.

See also

Basics of using external source files (Page 6935)

Rules for programming external source files (Page 6936)

Saving blocks as external source files (Page 6938)

Opening and editing external source files (Page 6940)

Generating blocks from external source files (Page 6941)

7.4.5 Opening and editing external source files

By linking the files with the file name extensions "stl" and "scl" to an editor you will be able to open and edit external source files with these formats directly. Use Notepad as editor because other text editors may not let you open several sources at the same time.

This means you do not need to insert the external source files again after editing.

Linking files with the file name extensions "stl" and "scl" file types to an editor

Proceed as follows to link files with the file name extensions "stl" and "scl" to an editor:

1. Open Windows Explorer.
2. Right-click on an STL file.
3. Select "Properties" in the shortcut menu.
   The "Properties" dialog box opens.
4. Click "Change" in the "File type" area on the "General" tab.
   The "Open with" dialog box opens.
5. Select the text editor you want to link to the "stl" file type.
6. Confirm your selection with "OK".
7. Close the "Properties" dialog with "OK".
8. Repeat steps 2 to 7 with an SCL file.

Opening and editing an external source file

To open an external source file, follow these steps:

1. Open the "External source files" folder in the project tree.
2. Double-click on the external source file you want to open.
   The external source file will open in the linked editor and can be edited.

See also

Basics of using external source files (Page 6935)

Rules for programming external source files (Page 6936)

Saving blocks as external source files (Page 6938)

Inserting external source files (Page 6939)

Generating blocks from external source files (Page 6941)

7.4.6 Generating blocks from external source files

Requirement

- The "External source files" folder is open in the project tree.
- An external source file is available.
- The external source file was saved with a permitted character encoding.

Procedure

To generate blocks from an external source file, follow these steps:

1. Open the external source file from which you wish to generate blocks.
2. Select the "Generate blocks from source" command in the "Edit" menu.
3. A prompt will appear telling you any existing blocks will be overwritten.
4. Confirm the safety prompt with "Yes".

Result

The external source file blocks will be generated and inserted in the "Program blocks" folder in the project tree. In the event of errors, information about the errors which have occurred will be displayed in the inspector window. This information, however, relates to the external source file and not to the block generated.

See also

Basics of using external source files (Page 6935)
Rules for programming external source files (Page 6936)
Saving blocks as external source files (Page 6938)
Inserting external source files (Page 6939)
Opening and editing external source files (Page 6940)
Specifying block properties (Page 6920)

Protecting blocks 8

8.1 Protecting blocks

Introduction

You have the following options for protecting your blocks

- Know-how protection
  With the help of know-how protection you can use a password to protect blocks of the OB, FB, FC type and global data blocks from unauthorized access.
- Copy protection:
  For S7-1200/1500 CPUs, you can also set up copy protection for code blocks which binds execution of the block to the CPU or the memory card with the defined serial number. When using copy protection for a block it makes sense to also know-how-protect the block to ensure that copy protection can only be revoked with a password.
- Write protection:
  You can assign write protection for code blocks, which prevent unintentional changes to the block.

Note the following points for know-how protection:

- You can not manually protect instance data blocks; they depend on the know-how protection of the assigned FB. This means that when you create an instance data block for a know-how protected FB, the instance data block also receives this know-how protection. This is independent of whether you explicitly create the instance data block or if it is created by a block call.
- The know-how protection acts as write protection in global blocks and in instance data blocks.
- You cannot provide ARRAY data blocks with know-how protection.
- The load memory requirement can be higher for know-how-protected blocks.

If a block is know-how protected, only the following data is readable without the correct password:

- Interface parameters Input, Output, InOut, Return, Static
- Block title
- Block comment
- Block properties
- Tags of global data blocks without specification of the location of use

The following actions can be performed with a know-how protected block:

- Copying and deleting
- Calling in a program

- Compare offline/online
- Downloading

The code of the block, on the other hand, is protected from unauthorized reading and modification.

Note

Please note the following:

- S7-1200 version 1.0 and S7-300/400 (only GRAPH and SCL blocks): If you download a know-how-protected block to a device, no restore information is loaded along with it. This means that you cannot open a know-how-protected block again even with the correct password if you upload it from the device.
- Only the non-protected data is compared in offline-online comparison of know-how protected blocks.
- You will no longer be able to access the block if you do not have the password.
- If you add a know-how-protected block to a library, the master copy created will also be know-how protected.
- You will find information on whether or not the block can be used as protected library item in the block properties under "Compilation > Library conformance". For this purpose, the block cannot use any tags from the operand areas Output (Q), Input (I), Bit memory (M), Timer function (T) or Count function (C), and cannot access data blocks.
- Cross-references to used tags, bit memories, inputs and outputs in know-how-protected blocks are only displayed if the relevant blocks were unlocked before the cross-reference list was created by entering the correct password.
- Automatic renumbering and manual renumbering without password of know-how-protected blocks is only possible for CPUs of the series S7-1500 and S7-1200 (V4). The know-how protection must also be set with a TIA Portal version V13 SP1 or higher. If these requirements for a know-how-protected block are not met, the loadable binary component of the block is no longer up-to-date when you change the number. In this case, the block must be recompiled before downloading it to a device. For know-how-protected blocks, this is only possible with the correct password. Keep this in mind particularly if you want to copy a know-how-protected block to another device in which there is already a block with the same number.
- Always pass on a project that includes know-how-protected blocks as a project archive or library archive. In this way, you ensure that the know-how protection cannot be bypassed.
- You cannot change the know-how protection settings for an open, know-how protected block.

Using a password provider

You have the option of connecting the TIA Portal to an external password management system via a password provider. For this, a project product version V14 SP1 and higher is required.

If you are using a password provider, you can select the password from a list of passwords when setting know-how protection, copy or write protection. Only the names of the passwords are displayed in the list, the passwords themselves are not visible. When a block is opened, the TIA Portal connects to the password provider and retrieves the corresponding password. If the password provider is not available at this time, you can only open the block fully if manual

input is defined as the fallback strategy and the corresponding password is known. The name of the password is not sufficient here. Depending on the selected security setting in the password provider, you can change the password for a know-how protected block or revoke protection again.

The use of a password provider increases security for your program code, as it is not necessary to pass on passwords for know-how protected blocks to other editors. Instead, all editors can connect the same password provider in order to obtain access to the blocks. If an editor leaves the project, there is no need to change the password.

In the event of problems with connected password providers, you can record the activities of password providers and forward the recording to Customer Support for analysis.

Note

Safety note for block protection

If a block is protected by a password that was entered manually, manual input is always required to edit the block or its protection settings. This is true even if a password provider contains a password with the same content at a later time.

See also

Connecting the password provider (Page 6946)

Show available password provider (Page 6949)

Recording activities of the password providers (Page 6950)

Define the type of password assignment (Page 6952)

Setting up and removing block copy protection (Page 6954)

Setting up and removing block write protection (Page 6956)

Setting up block know-how protection (Page 6959)

Opening know-how protected blocks (Page 6960)

Printing know-how protected blocks (Page 6962)

Changing a password (Page 6963)

Removing block know-how protection (Page 6964)

8.2 Connecting the password provider

As an alternative to manual password input, a password provider can be connected to the TIA Portal. When using a password provider, you can select a password from a list of available passwords. When a know-how protected block is opened, the TIA Portal connects to the password provider and retrieves the corresponding password.

To connect a password provider you have to install and activate it. A settings file in which the use of a password provider is defined is also required.

Installing and activating password providers

To install and activate a password provider, proceed as follows:

1. Install the desired password provider in accordance with the installation instructions of the password provider.
2. Open the TIA Portal.
3. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
4. In the area navigation, select the "Password provider > Available password providers" group.
   The newly installed password provider is displayed in the table of available password providers.
5. Select the check box in the "Status" column to use the password provider.
   The installed password provider is now active and can be used.
6. Set the preferred password input such that the password provider is used.
   See also: Define the preferred password assignment (Page 6952)

Create a settings file for using password providers

To create a settings file for the use of password providers, proceed as follows:

1. Create an XML file with the content specified below. Use the coding "UTF-8".
2. In the setting tag with the name "EnablePasswordPreference", define whether a user can switch between manual password assignment and password assignment via a password provider.
3. In the setting tag with the name "PasswordPreference", define which type of password assignment is to be used.
4. Save the XML file.
5. Save the file named "CorporateSettings.xml" in the following directory on your computer:
   C:\ProgramData\Siemens\Automation\Portal V15_1\CorporateSettings\

Content of settings file

The settings file must have the following content:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
   <Settings.Settings ID="0">
       <ObjectList>
          <Settings.General ID="1" AggregationName="General">
             <ObjectList>
                <Settings.PasswordProviders ID="3" AggregationeName="PasswordProviders">
                   <AttributeList>
                      <EnablePasswordPreference>
                         <Value>true</Value>
                      </EnablePasswordPreference>
                      <DefaultPasswordPreference>
                         <Value>Advanced</Value>
                      </DefaultPasswordPreference>
                   </AttributeList>
                </Settings.PasswordProviders>
             </ObjectList>
          </Settings.General>
       </ObjectList>
    </Settings.Settings>
</Document>
```

You specify the settings for the password provider in the element "Settings.PasswordProviders":

- The following values are possible for "EnablePasswordPreference":
  - "true": In the TIA Portal the user can switch between manual password assignment and password assignment via password providers.
  - "false": Switching between manual password assignment and password assignment via password providers is not possible in the TIA Portal.

  The default setting is "false".
- The following values are possible for "DefaultPasswordPreference":
  - "Manual": Assignment of passwords in the TIA Portal is done manually.
  - "Advanced": Passwords are assigned in the TIA Portal via a password provider.
  - "AdvancedWithManualFallback": Passwords are assigned in the TIA Portal via a password provider. If the password provider is not available, the protected block can be accessed via manual password input.

  The default setting is "Manual".
  If the "CorporateSettings.xml" file already exists and has different settings, supplement the element "Settings.PasswortProviders" in the "ObjectList>" element.

See also

Protecting blocks (Page 6943)

Show available password provider (Page 6949)

Recording activities of the password providers (Page 6950)
Setting up and removing block copy protection (Page 6954)
Setting up and removing block write protection (Page 6956)
Setting up block know-how protection (Page 6959)
Opening know-how protected blocks (Page 6960)
Printing know-how protected blocks (Page 6962)
Changing a password (Page 6963)
Removing block know-how protection (Page 6964)

8.3 Show available password provider

You can display the password providers which are available in the TIA Portal.

Requirement

Password assignment via password provider was defined using a settings file.

Procedure

Proceed as follows to display the available password providers:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "Password provider > Available password providers" group.
   The available password providers are displayed in the overview table.

See also

Protecting blocks (Page 6943)

Connecting the password provider (Page 6946)

Recording activities of the password providers (Page 6950)

Define the type of password assignment (Page 6952)

Setting up and removing block copy protection (Page 6954)

Setting up and removing block write protection (Page 6956)

Setting up block know-how protection (Page 6959)

Opening know-how protected blocks (Page 6960)

Printing know-how protected blocks (Page 6962)

Changing a password (Page 6963)

Removing block know-how protection (Page 6964)

8.4 Recording activities of the password providers

In the event of problems in connection with connected password providers, you can record the activities of password providers. No passwords are recorded. You can then forward the recording to Customer Support for analysis.

Requirement

A password provider is connected and activated.

Activate recording of the activities of the password providers

Proceed as follows to activate the recording:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "Password providers > Analysis of the password providers" group.
3. Select the "Log activities of the password provider" check box.
   The text field for the storage location of the recording file and the "Browse" button become active.
4. Enter a storage location for the recording file or select it using the "Browse" button.

Deactivate recording of the activities of password providers

Proceed as follows to deactivate recording:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "Password providers > Analysis of the password providers" group.
3. Deselect the "Log activities of the password provider" check box.
   The text field for the storage location of the recording file and the "Browse" button become inactive.

See also

Protecting blocks (Page 6943)

Connecting the password provider (Page 6946)

Show available password provider (Page 6949)

Define the type of password assignment (Page 6952)

Setting up and removing block copy protection (Page 6954)

Setting up and removing block write protection (Page 6956)

Setting up block know-how protection (Page 6959)

Opening know-how protected blocks (Page 6960)

Printing know-how protected blocks (Page 6962)
Changing a password (Page 6963)
Removing block know-how protection (Page 6964)

8.5 Define the type of password assignment

The TIA Portal offers the following options for the type of password assignment for blocks for know-how protection, copy protection and write protection:

- "Manual password input"
  With manual password input, you enter the password directly in the dialog for the password request.
- "Selection via password provider"
  With password input via an external password provider, you can select the password using a name from the list of available passwords. If the password provider is not available, it is not possible to set protection. The block can only be accessed later if there is a connection to the password provider.
- "Selection via password provider with manual input as fallback strategy for access"
  Like password input via an external password provider. If the password provider is not available and you know the password, you can, however, open the block for editing by entering the password manually.

Manual password input is the default setting.

Requirement

The following requirements must be met in order that password assignment via an external password provider can be used

- Password assignment via password provider was defined using a settings file.
- The change of the type of password assignment was allowed in the settings file.
- A password provider is connected.

Procedure

Proceed as follows to define the type of password assignment:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "Password provider > Password assignment" group.
3. Activate the desired type of password assignment.

See also

Protecting blocks (Page 6943)

Connecting the password provider (Page 6946)

Show available password provider (Page 6949)

Recording activities of the password providers (Page 6950)

Setting up and removing block copy protection (Page 6954)

Setting up and removing block write protection (Page 6956)

Setting up block know-how protection (Page 6959)
Opening know-how protected blocks (Page 6960)
Printing know-how protected blocks (Page 6962)
Changing a password (Page 6963)
Removing block know-how protection (Page 6964)

8.6 Setting up and removing block copy protection (S7-1200, S7-1500)

For S7-1200 /1500 CPUs, you can set up copy protection which binds execution of the block to a specific CPU or a specific memory card. The block can then only be executed if it is in the device with the set serial number. If you want to add the serial number automatically through a download process, you must specify a copy protection password. This password is queried during a download process of the block. You can change the password without having to remove the know-how protection of the block. When you change the copy protection settings or remove the copy protection, the password is retained until the next compilation process.

It is important that you also know-how-protect any block for which you have set up copy protection. If you do not, anyone can reset the copy protection.

Note

S7-1500 and S7-1200 V2.2 and higher: If you download a copy protected block to a device that does not match the specified serial number, the entire download operation will be rejected. This means that blocks without copy protection, too, will not be downloaded.

Setting up copy protection

To set up copy protection for a block, follow these steps:

1. Open the properties of the block for which you wish to set up copy protection.
2. Select the "Protection" entry in the area navigation in the block properties.
3. Select either "Bind to serial number of the CPU" or "Bind to serial number of the memory card" from the drop-down list in the "Copy protection" area.
4. Enter either the serial number directly or enable the option "Serial number inserted when downloading to a device or memory card" if the serial number is to be inserted automatically when you load.
   Continue with step 7 if you have entered the serial number directly. If the serial number is to be added automatically during the download process, you must also execute steps 5 and 6.
5. Click "Define password".
   The "Define protection" dialog opens.
6. If you are using manual password input, enter the password in the fields "New password" and "Confirm password".
7. If you are using a password provider, first select the password provider and then the password.
8. Confirm your entries with "OK".
   The status box provides the information that a copy protection password is defined.
9. You can now set up the know-how protection for the block in the "Know-how protection" area, if the block does not already have know-how protection.

Removing copy protection

To remove copy protection, follow these steps:

1. Open the properties of the block for which you wish to remove copy protection.

   Note

   If the block has know-how protection, you must open the block with the correct password in order to be able to change the copy protection settings.

2. Select the "Protection" entry in the area navigation in the block properties.
3. Select "No binding" in the drop-down list in the "Copy protection" area.

See also

Protecting blocks (Page 6943)

Connecting the password provider (Page 6946)

Show available password provider (Page 6949)

Recording activities of the password providers (Page 6950)

Define the type of password assignment (Page 6952)

Setting up and removing block write protection (Page 6956)

Setting up block know-how protection (Page 6959)

Opening know-how protected blocks (Page 6960)

Printing know-how protected blocks (Page 6962)

Changing a password (Page 6963)

Removing block know-how protection (Page 6964)

8.7 Setting up and removing block write protection

You can set up write protection for blocks of the type OB, FB, FC to prevent them from being changed unintentionally. Blocks with write protection can only be opened as "read only", the block properties can however still be edited.

Please note that write protection is not know-how protection, as you can remove write protection again at any time. If a block is write-protected, you cannot additionally set up know-how protection. Remove the write protection for the block if you want to set up know-how protection for it.

Setting up write protection

To set write protection for a block, follow these steps:

1. Open the properties of the block for which you wish to set up write protection.
2. Select the "Protection" entry in the area navigation in the block properties.
3. In the area "Write protection" select "Define password".
   The "Define protection" dialog opens.
4. If you are using manual password input, enter the password in the fields "New password" and "Confirm password".
5. If you are using a password provider, first select the password provider and then the password.
6. Confirm your entries with "OK".
   The status box provides the information that a write protection password is defined. The text of the button changes to "Change password".
7. Select the "Write protection" check box.
   The "Access protection" dialog will open.
8. Enter the correct password or select the correct password if you are using a password provider.
   Write protection is enabled and will be active the next time the block is opened.

Disabling write protection

Proceed as follows to disable write protection:

1. Open the properties of the block for which you wish to disable write protection.
2. Select the "Protection" entry in the area navigation in the block properties.
3. In the area "Write protection" select the "Write protection" check box.
   The "Access protection" dialog will open.
4. Enter the correct password or select the correct password if you are using a password provider.
   The write protection for the block is disabled the block can be edited again the next time the block is opened. The password is retained, so that you can enable write protection again at any time.

Enable write protection.

Proceed as follows to enable write protection:

1. Open the properties of the block for which you wish to enable write protection.
2. Select "Protection" in the area navigation in the inspector window.
3. In the area "Write protection" select the "Write protection" check box.
   The "Access protection" dialog will open.
4. Enter the correct password or select the correct password if you are using a password provider.
   The write protection for the block is reset and will be active the next time the block is opened.

Changing the write protection password

To change the write protection password, follow these steps:

1. Open the properties of the block for which you wish to change the write protection password.
2. Select the "Protection" entry in the area navigation in the block properties.
3. Click the "Change password" button.
   The "Change protection" dialog opens.
4. If you are using manual password input, enter the old password in the field "Old password" and the new password in the field "New password". Also enter the new password in the "Confirm password" box
5. If you are using a password provider, first select the password provider and then the new password.
6. Confirm your entries with "OK".
   The write protection password for the block is changed.

Note

It is always possible to change the write protection password for the block. This depends on whether write protection is active or not.

Removing write protection

To set write protection for a block, follow these steps:

1. Open the properties of the block for which you wish to remove write protection.
2. Select the "Protection" entry in the area navigation in the block properties.
3. Click the "Change password" button.
   The "Change protection" dialog opens.
4. If you are using manual password input, enter the current password in the field "Old password".

5. If you are using a password provider, first select the password provider and then the current password.
6. Click "Remove"
   The write protection for the block is removed. Unlike deactivation of write protection, the password is also deleted.

See also

Protecting blocks (Page 6943)

Connecting the password provider (Page 6946)

Show available password provider (Page 6949)

Recording activities of the password providers (Page 6950)

Define the type of password assignment (Page 6952)

Setting up and removing block copy protection (Page 6954)

Setting up block know-how protection (Page 6959)

Opening know-how protected blocks (Page 6960)

Printing know-how protected blocks (Page 6962)

Changing a password (Page 6963)

Removing block know-how protection (Page 6964)

8.8 Setting up block know-how protection

You can set up know-how protection for blocks in the devices in your project.

Procedure

To set up block know-how protection, follow these steps:

1. Select the blocks with no know-how protection which you want to protect.
2. Select the command "Know-how protection" in the "Edit" menu.
   The "Define protection" dialog opens.
3. If you are using manual password input, enter the password in the fields "New password" and "Confirm password".
4. If you are using a password provider, first select the password provider and then the password.
5. Confirm your entries with "OK".
6. Close the "Define protection" dialog with "OK".

Result

The blocks selected will be know-how-protected. Know-how protected blocks are marked with a lock in the project tree. The password entered is valid for all blocks selected.

See also

Protecting blocks (Page 6943)

Connecting the password provider (Page 6946)

Show available password provider (Page 6949)

Recording activities of the password providers (Page 6950)

Define the type of password assignment (Page 6952)

Setting up and removing block copy protection (Page 6954)

Setting up and removing block write protection (Page 6956)

Opening know-how protected blocks (Page 6960)

Printing know-how protected blocks (Page 6962)

Changing a password (Page 6963)

Removing block know-how protection (Page 6964)

8.9 Opening know-how protected blocks

If the use of a password provider is set as the type of password assignment, the TIA Portal connects to the password provider when a protected block is opened. If the connection is successful, there is no need to enter a password. If the password provider is not available, you can only open the block fully if you had defined manual input as the fallback strategy and know the corresponding password. The name of the password is not sufficient here.

You can only open multiple know-how protected blocks at once if they are protected with the same password.

Requirement

- "Manual password input" is set as the type of password assignment.
- "Selection via password provider with manual input as fallback strategy for access" is set and there is no connection to the password provider.

Procedure

To open a know-how protected block, follow these steps:

1. Double-click on the block you wish to open.
   The "Access protection" dialog will open.
2. Enter the password for the know-how protected block.
3. Confirm your entry with "OK".

Result

The know-how protected block will open provided you have entered the correct password. However, the block will remain know-how protected. If you copy the block or add it to a library, for example, the copies will also be know-how protected.

Once you have opened the block, you can edit the program code and the block interface of the block for as long as the block or TIA Portal is open. The password must be entered again the next time the block is opened. If you close the "Access protection" dialog with "Cancel", the block will open but the block code will not be displayed and you will not be able to edit the block.

See also

Protecting blocks (Page 6943)

Connecting the password provider (Page 6946)

Show available password provider (Page 6949)

Recording activities of the password providers (Page 6950)

Define the type of password assignment (Page 6952)

Setting up and removing block copy protection (Page 6954)

Setting up and removing block write protection (Page 6956)

Setting up block know-how protection (Page 6959)
Printing know-how protected blocks (Page 6962)
Changing a password (Page 6963)
Removing block know-how protection (Page 6964)

8.10 Printing know-how protected blocks

You can only print complete know-how protected blocks if they have been opened with the correct password. If you print a closed block or if the block was not opened with the correct password, only the non-protected block data will be printed.

Procedure

To print a know-how protected block in full, follow these steps:

1. Open the know-how protected block you wish to print.
   See also: Opening know-how protected blocks (Page 6960)
2. Select the "Print" command in the "Project" menu.
   The "Print" dialog will open.
3. Select the printer in the "Name" field.
4. Click "Advanced" to modify the Windows printer settings.
5. Select the documentation information set in the "Document information" drop-down list that you want to use for the frame layout.
6. Under "Print objects/area" select whether you want to print all objects or the complete area, or only a selection.
7. Under "Properties" select the print scope.
   - Select "All" to print the complete block.
   - Choose "Visible" to print all the information within the block that is visible on the screen.
   - Select "Compact" to print a shortened form of the block.
8. Click "Preview" to generate a print preview in advance.
   A print preview is created in the work area.
9. Click "Print" to start the printout.

See also

Protecting blocks (Page 6943)

Connecting the password provider (Page 6946)

Show available password provider (Page 6949)

Recording activities of the password providers (Page 6950)

Define the type of password assignment (Page 6952)

Setting up and removing block copy protection (Page 6954)

Setting up and removing block write protection (Page 6956)

Setting up block know-how protection (Page 6959)

Changing a password (Page 6963)

Removing block know-how protection (Page 6964)

8.11 Changing a password

If you are using a password provider, whether you are able to change the password for know-how protected blocks depends on the specifications of the password provider.

Procedure

To change the password, follow these steps:

1. Select the know-how protected blocks for which you want to change the password.

   Note

   You can only change the password for several blocks at once if all blocks selected have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.
   The "Change protection" dialog opens.
3. If you are using manual password input, enter the old password in the field "Old password" and the new password in the field "New password". Also enter the new password in the "Confirm password" box
4. If you are using a password provider, first select the password provider and then the new password.
5. Close the "Change protection" dialog with "OK".

See also

Protecting blocks (Page 6943)

Connecting the password provider (Page 6946)

Show available password provider (Page 6949)

Recording activities of the password providers (Page 6950)

Define the type of password assignment (Page 6952)

Setting up and removing block copy protection (Page 6954)

Setting up and removing block write protection (Page 6956)

Setting up block know-how protection (Page 6959)

Opening know-how protected blocks (Page 6960)

Printing know-how protected blocks (Page 6962)

Removing block know-how protection (Page 6964)

8.12 Removing block know-how protection

If you are using a password provider, whether you are able to remove the know-how protection of a block depends on the specifications of the password provider.

Procedure

To remove block know-how protection, follow these steps:

1. Select the blocks for which you want to remove know-how protection.

   Note

   You can only remove know-how protection for several blocks at once if all blocks selected have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.
   The "Change protection" dialog opens.
3. If you are using manual password input, enter the password for know-how protection of the block in the field "Old password". This is not required if you are using a password provider.
4. Click the "Remove" button.

Result

Know-how protection will be disabled for the blocks selected.

See also

Protecting blocks (Page 6943)

Connecting the password provider (Page 6946)

Show available password provider (Page 6949)

Recording activities of the password providers (Page 6950)

Define the type of password assignment (Page 6952)

Setting up and removing block copy protection (Page 6954)

Setting up and removing block write protection (Page 6956)

Setting up block know-how protection (Page 6959)

Opening know-how protected blocks (Page 6960)

Printing know-how protected blocks (Page 6962)

Changing a password (Page 6963)

Declaring the block interface 9

9.1 Overview of the block interface

Introduction

The interface contains the declarations of local variables and constants that are used within the block. The variables are subdivided into two groups:

- Block parameters that form the block interface when it is called in the program.
- Local data that are used for storage of intermediate results.

You use the variable declaration to define the call interface of a block in the program and the names and data types of variables and constants that you want to use in the block.

The interface of function blocks also defines the structure of the instances that are assigned to the function block.

Tabular or textual representation of the block interface

The interface is displayed by default in the tabular view. In SCL the interface may also be displayed in a textual view.

See also:

Selecting tabular or textual representation of the block interface (Page 6973)

Block parameters

The following table shows the types of block parameters:

Type	Section	Function	Available in
Input parameters	Input	Parameters whose values are read by the block.	Functions, function blocks and some types of organization blocks
Output parameters	Output	Parameters whose values are written by the block.	Functions and function blocks
In/out parameters	InOut	Parameters whose values are read by the block when it is called, and whose values are written again by the block after execution.	Functions and function blocks
Return value	Return	Value that is returned to the calling block.	Functions

Depending on the type of block opened, additional sections may be displayed.

Local data

The following table shows the types of local data:

Type	Section	Function	Available in
Temporary local data	Temp	Tags that are used to store temporary intermediate results. Temporary local data are retained for only one cycle. If you use temporary local data, you have to make sure that the values are written within the cycle in which you want to read them. Otherwise the values will be random.	Functions, function blocks and organization blocks **Note**: Temporary local data is not shown in instance data blocks.
Static local data	Static	Tags that are used for storage of static intermediate results in the instance data block. Static data is retained until overwritten, which may be after several cycles. The names of the blocks, which are called in this code block as multiple instance, will also be stored in the static local data.	Function blocks
Constant	Constant	Constants with declared symbolic names that are used within the block.	Functions, function blocks and organization blocks **Note**: Local constants are not shown in instance data blocks.

See also

Using tags within the program (Page 95)

Keywords (Page 89)

Valid data types in the block interface (Page 6969)

Setting the retentivity of local tags (Page 6971)

Basics of constants (Page 97)

Parameter transfer at block call (Page 76)

Using setpoints during commissioning (Page 7063)

9.2 Rules for declaring the block interface

9.2.1 General rules for declaring the block interface

Using block parameters

The following rules apply to the use of block parameters within the block:

- Input parameters may only be read.
- Output parameters may only be written.
- In/out parameters may be read and written.

Note

Maximum number of parameters per block

The maximum permitted number of formal parameters is sufficient for all usual application scenarios. The exact number depends on the selected data types, declaration subsections and other factors.

If the maximum number is exceeded, you receive a message during the compilation process. In this case, you can combine several parameters in a PLC data type (UDT) or in a global data block (DB) and transfer this as a block parameter.

Assigning default values to block parameters

You can assign default values to specific parameters in the function block interface. The possibility of the assignment depends on the declaration subsection and data type of the parameter. If you do not specify a default value the pre-defined value for the specified data type is used. For example, the value "false" is predefined for BOOL.

The following table shows which parameters can be assigned a default value:

Parameter type	Section	Assignment of a default value is possible		
		Elementary data types	Structured data types	Parameter types
Input parameters	Input	X	X	-
Output parameters	Output	X	X	-
In/out parameters	InOut	X	- (1)	-
Static local data	Static	X	X	-
Temporary local data	Temp	-	-	-
Constants	Constant	X	-	-

(1) Exception: In blocks with optimized access, you have the option to use PLC data types as default values under certain conditions.

Note

You can find more information on permissible characters in tag names in the following entry at Siemens Industry Online Support:

When should identifiers and operands be used in "quotation marks" in STEP 7 (TIA Portal)?

https://support.industry.siemens.com/cs/ww/en/view/109477857 (https://support.industry.siemens.com/cs/ww/en/view/109477857)

You can find recommendations for uniform, project-wide tag naming in the programming style guide:

https://support.industry.siemens.com/cs/document/109478084 (https://support.industry.siemens.com/cs/ww/en/view/81318674)

See also

Using tags within the program (Page 95)

Keywords (Page 89)

Declaring predefined actual parameters (Page 6987)

9.2.2 Valid data types in the block interface (S7-300, S7-400)

Valid data types in the block interface in S7-300/400

The following table shows which data types you can assign to the parameters in the individual sections of the interface.

Section	Standard Data types	ARRAY STRUCT STRING DT	Parameter types	VOID	POINTER	ANY
Organization block						
Temp	X	X	-	-	-	X
Constant	X	X (3)	-	-	-	-
Function block						
Input	X	X	X	-	X	X
Output	X	X	-	-	-	-
InOut	X	X (1)	-	-	X	X
Static	X	X	-	-	-	-
Temp	X	X	-	-	-	X
Constant	X	X (3)	-	-	-	-
Function						

Section	Standard Data types	ARRAY STRUCT STRING DT	Parameter types	VOID	POINTER	ANY
Input	X	X (1)	X	-	X	X
Output	X	X (1)	-	-	X	X
InOut	X	X (1)	-	-	X	X
Temp	X	X	-	-	-	X
Return	X	X	-	X	X	X (2)
Constant	X	X (3)	-	-	-	-

(1) STRING can only be defined in the standard length of 254 characters.
(2) In SCL, ANY is not permitted as a function value.
(3) Constants with the ARRAY or STRUCT data type are not permitted.

See also

Parameter transfer at block call (Page 76)

9.2.3 Valid data types in the block interface (S7-1200, S7-1500)

Valid data types in the block interface in S7-1200

The following table shows which data types you can assign in the parameters in the individual sections of the interface.

Section	Standard Data types	ARRAY STRUCT STRING / WSTRING DT	ARRAY [*]	VOID	VARIANT
Organization block					
Temp	X	X	-	-	X
Constant	X	X (1) (2)	-	-	-
Function block					
Input	X	X	-	-	X
Output	X	X	-	-	-
InOut	X	X (1)	X (4)	-	X
Static	X	X	-	-	-
Temp	X	X	-	-	X
Constant	X	X (1) (2)	-	-	-
Function					
Input	X	X (1)	X (4)	-	X
Output	X	X (1)	X (4)	-	X

Section	Standard Data types	ARRAY STRUCT STRING / WSTRING DT	ARRAY [*]	VOID	VARIANT
InOut	X	X (1)	X (4)	-	X
Temp	X	X	-	-	X
Return	X	X (3)	-	X	-
Constant	X	X (1) (2)	-	-	-

(1) You cannot make length declarations for STRING and WSTRING in these sections. STRINGs always have the standard length 254; WSTRINGs the standard length 16832. A declaration in the format MyString[3] would not be permitted. WSTRING is only permitted in blocks with optimized access in these sections.

(2) Constants with the ARRAY or STRUCT data type are not permitted.

(3) Function values of the WSTRING data type must not be longer than 1022 characters.

(4) ARRAY[*] is available in blocks with optimized access as of firmware V4.2.

Valid data types in the block interface in S7-1500

The following table shows which data types you can assign in the parameters in the individual sections of the interface.

Section	Stand-ard Data types	ARRAY STRUCT STRING / WSTRING DT	AR-RAY[*]	Param-eter types	VOID	DB_ANY	POINT-ER	ANY	VAR-IANT	REF_TO
Organization block										
Temp	X	X	-	- (4)	-	X	-	X (3)	X	X
Constant	X	X (1) (5)	-	-	-	-	-	-	-	-
Function block										
Input	X	X	-	X	-	X	X	X	X	-
Output	X	X	-	-	-	X	-	-	-	-
InOut	X	X (1)	X(7)(8)	- (4)	-	X	X	X	X	-
Static	X	X	-	-	-	X	-	-	-	-
Temp	X	X	-	- (4)	-	X	-	X (3)	X	X
Constant	X	X (1) (5)	-	-	-	-	-	-	-	-
Function										
Input	X	X (1)	X (7)	X	-	X	X	X	X	X
Output	X	X (1)	X (7)	-	-	X	X	X	X	X
InOut	X	X (1)	X (7)	- (4)	-	X	X	X	X	-
Temp	X	X	-	- (4)	-	X	-	X (3)	X	X
Return	X	X (6)	-	-	X	X	X	x (2)	-	X

Section	Standard Data types	ARRAY STRUCT STRING / WSTRING DT	ARRAY[*]	Parameter types	VOID	DB_ANY	POINTER	ANY	VARIANT	REF_TO
Constant	X	X (1) (5)	-	-	-	-	-	-	-	-
(1) You cannot make length declarations for STRING and WSTRING in these sections. STRINGs always have the standard length 254; WSTRINGs the standard length 16832. A declaration in the format MyString[3] would not be permitted. WSTRING is only permitted in blocks with optimized access in these sections. (2) In SCL, ANY is not permitted as a function value. (3) ANY can only be used in blocks with standard access in the "Temp" section. (4) The "INSTANCE" parameter type is the only exception permissible in the "TEMP" and "InOut" sections. (5) Constants with the ARRAY or STRUCT data type are not permitted. (6) Function values of the WSTRING data type must not be longer than 1022 characters. (7) ARRAY[*] is available in blocks with optimized access as of firmware V2.0. (8) ARRAY[*] is only available when the "Set in IDB" attribute is not set.										

See also

Parameter transfer at block call (Page 76)

9.2.4 Setting the retentivity of local tags

Introduction

Function blocks store their data in an instance. To prevent data loss in the event of power failure, you can mark the data as retentive. This data is stored in a retentive memory area. The option of setting the retentivity depends on the set access type of the function block.

Retentive behavior in blocks with standard access

In blocks with standard access you cannot set the retentive behavior of individual tags. You can only define them as retentive in the assigned instance. All tags contained in the block are then considered as retentive.

Retentivity for optimized block access

In data blocks with optimized access you can define the retentive behavior of individual tags.

For structured data type tags, the retentivity setting always applies to the entire structure. You can make no individual retentivity setting for individual elements within the structure.

You cannot create retentive tags of the structured data type in the "InOut" section. In/out parameters with structured data type, for example ARRAY, STRUCT, or STRING, are always non-retentive.

The following settings are available:

- Retentive
  The values of the tags or the structure are available even after a power failure.
- Non-retentive
  The values of the tags or the structure are lost in the event of a power failure.
- Set in IDB
  The retentivity can be set in the instance data block. The setting that is made in the instance data block than applies, however, centrally to all tags that are selected with "Set in IDB".

See also

Properties of local tags and constants (Page 7009)

Basics of block access (Page 55)

9.3 Selecting tabular or textual representation of the block interface

Tabular or textual representation of the block interface

By default, the interface is displayed in the table view. The interface may also be displayed in a textual view in SCL.

The representation of the interface of an SCL block can be set centrally. If you select textual representation as the default, the interface of all newly created blocks in the future will be displayed in the textual view.

The following elements can only be used in the text view:

- Comments between sections or at the beginning of a section
- Several sections of a type (e.g. VAR_INPUT)

Note

Fail-safe blocks and system blocks

For fail-safe blocks and system blocks, the interface is only displayed in tabular form.

Setting the display of the interface for new blocks

To configure the representation of the new blocks, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > SCL" group in the area navigation.
3. Select the desired view of the interface under "Interface > Default settings for new blocks".

When you create new blocks, their interface is shown in the selected view.

Note

Display of the interface after importing a block

After a block is imported, its interface is displayed in the tabular view.

See also

Layout of the block interface (Page 6974)

Structure of the textual block interface (Page 6991)

9.4 Declaring the block interface

9.4.1 Layout of the block interface

Layout of the block interface

The following figure shows the structure of the block interface. The number of columns and sections varies depending on the type of block.

Name	Data type	Default value	Retain
Input			
My_Input1	Bool	false	Non-retain
Output			
My_Output1	Bool	1	Retain
InOut			
<Hinzufügen>			
Static			
Temp			
Constant			
My_Constant1	Int	22	

Meaning of the columns

The following table shows the meaning of the individual columns. You can show or hide the columns as required. The number of columns displayed varies depending on the CPU series and the type of the open object.

Column	Explanation
	Symbol you can click on to drag-and-drop an element to a program for use as an operand.
Name	Name of the element.
Data type	Data type of the element.
Offset	Relative address of a tag within the block. The column is only visible in blocks with standard access. Note: Many instructions from the SIMATIC system libraries have the "optimized block access" property and therefore do not occupy any fixed memory addresses. No offset is displayed for these instructions, even if you use them as multi-instance in a block with standard access.

Column	Explanation
Default value	Value with which you can preassign specific tags in the interface of the code block, or the value of a local constant. Specification of the default value is optional for tags. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL. The default value of a tag is applied as the start value in the corresponding instance data block. You can replace these values with instance-specific start values in the instance data block. Constants always have the default value declared in the block interface. They are not shown in instance data blocks and cannot be assigned instance-specific values there.
Retain	Marks a tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This column is only visible in the interface of the function block with optimized access.
Visible in HMI engineering	Indicates whether a tag is visible by default in the HMI selection list.
Accessible from HMI/OPC UA	Indicates whether HMI/OPC UA can access this tag during runtime.
Writable from HMI/OPC UA	Indicates whether the tag from HMI/OPC UA can be written during runtime.
Setpoint	Marks a tag as a setpoint value Setpoint values are the values that will probably have to be fine tuned during commissioning. The column is only available in the interface of function blocks.
Supervision	Indicates whether Supervision has been created for process diagnostics for the tag.
Comment	Comments on documentation of the element.

See also

Using tags within the program (Page 95)

Keywords (Page 89)

Valid data types in the block interface (Page 6969)

Setting the retentivity of local tags (Page 6971)

Basics of constants (Page 97)

Selecting tabular or textual representation of the block interface (Page 6973)

Parameter transfer at block call (Page 76)

Using setpoints during commissioning (Page 7063)

9.4.2 Declaring local tags and constants in the block interface

Requirement

The block interface is open.

Procedure

To declare a tag or constant, follow these steps:

1. Select the desired declaration section in the block interface.
   You must declare a constant in the "Constant" section.
2. Enter a name for the element in the "Name" column.
3. Enter the required data type in the "Data type" column. You will be supported by Autocomplete during input.
4. For constants, enter a value in the "Default value" column.
5. Optional: Change the properties that are displayed in the other columns of the block interface.

Result

The element is created.

Syntax check

A syntax check is performed after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. However, you will not be able to compile the program if the tag declaration contains syntax errors.

Note

If you change the interface of a block, the calls of the block in the program will possibly become inconsistent. The call locations are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent blocks have to be updated manually.

See also:

Updating block calls in LAD (Page 7144)

Updating block calls in FBD (Page 7207)

See also

Rules for declaring the block interface (Page 6967)

Using tags within the program (Page 95)

Constants (Page 97)

Declaring tags of the ARRAY data type (Page 6978)

Declaring tags of STRUCT data type (Page 6980)

Declare variables of the STRING and WSTRING data types (Page 6981)

Declaring tags based on a PLC data type (Page 6982)

Declaring higher-level tags (Page 6983)

Declaring multi-instances (Page 6983)
Declaring parameter instances (Page 6985)
Basic information on start values (Page 7035)
Setting the retentivity of local tags (Page 6971)
Properties of local tags and constants (Page 7009)

9.4.3 Declaring a local tag in the program editor

Requirement

The program editor is open.

Procedure

To declare a local tag, follow these steps:

1. Insert an instruction in your program.
   The "<???>", "<??.?>" or "..." strings represent operand placeholders.
2. Replace an operand placeholder with the name of the tag to be created.
3. Select the name of the element.
   If you want to declare multiple elements, select the names of all the elements to be declared.
4. Select the "Define tag" command in the shortcut menu.
   The "Define tag" dialog box opens. It displays a declaration table in which the name of the element is already entered.
5. To declare a local tag, select one of the following sections:
   - Local In
   - Local Out
   - Local InOut
   - Local Static
   - Local Temp
6. In the other columns, enter data type and comments.
7. Click the "Define" button to complete your entry.

Result

The declaration is written directly into the block interface and is valid within the entire block.

Note

If you change the interface of a block, the calls of the block in the program will possibly become inconsistent. The call locations are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent blocks have to be updated manually.

See also:

Updating block calls in LAD (Page 7144)

Updating block calls in FBD (Page 7207)

See also

Using tags within the program (Page 95)

Keywords (Page 89)

Basic information on start values (Page 7035)

Properties of local tags and constants (Page 7009)

Setting the retentivity of local tags (Page 6971)

Edit table of the block interface (Page 6988)

Rules for declaring the block interface (Page 6967)

9.4.4 Declaring tags of the ARRAY data type

Requirement

The block interface is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Select the desired declaration section in the block interface.
2. Enter a tag name in the "Name" column.
3. In the "Data type" column, click the button for the data type selection.
   A list of the permissible data types is opened.
4. Select the "Array" data type.
   The "Array" dialog opens.
5. In the "Data type" text box, specify the data type of the ARRAY elements.

6. In the "ARRAY limits" input field, specify the high and low limit for each dimension.
   Example of a one-dimensional ARRAY:
   `Array[0..3] of Bool`
   Example of a three-dimensional ARRAY:
   `Array[0..3, 0..15, 0..33]of Bool`
   Example of a one-dimensional ARRAY with variable limits:
   `Array[*]of Bool`
   Example of a three-dimensional ARRAY with variable limits:
   `Array[*, *, *]of Bool`
   Example of a one-dimensional ARRAY with local constants as ARRAY limits
   `Array[#My_local_const1..#My_local_const2]of Bool`
   Example of a one-dimensional ARRAY with global constants as ARRAY limits
   `Array["My_global_const1".."My_global_const1"]of Bool`
7. Confirm your entry.
8. Optional: Change the properties of the tags that are displayed in the other columns of the block interface.

The tag is created but remains collapsed. To expand the ARRAY, click the triangle in front of the tag. Note that you cannot expand ARRAYs with more than 10,000 elements for reasons of clarity.

Note

Availability of ARRAY[*]

ARRAY[*] is available in optimized blocks for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0. In functions (FC) you can use ARRAY[*] in all declaration subsections. In function blocks (FB) you can declare ARRAY[*] only as an in-out parameter in the "InOut" section.

Entering default values of ARRAY elements

To pre-assign default values to the individual elements of an ARRAY, follow these steps:

1. Click the triangle in front of the ARRAY data type tags.
   The ARRAY is expanded and the individual ARRAY elements are shown in separate lines.
2. Enter the desired values in the "Default value" column.

Adding comments for ARRAYs

To add a comment for the entire ARRAY, enter the comment in the top line of the ARRAY declaration. The comment is used as a preset comment for all lower-level ARRAY elements.

To provide specific comments for the individual elements of an ARRAY, follow these steps:

1. Click the triangle in front of the ARRAY data type tags.
   The ARRAY is expanded and the individual ARRAY elements are shown in separate lines.
2. Enter the desired values in the "Comment" column.

Displaying ARRAYs in expanded mode

In "Expanded mode" all ARRAYs are displayed expanded by default.

To enable the expanded mode, follow these steps:

1. Click the "Expanded mode" button in the toolbar.

Note

Restriction in "Expanded mode" display

Depending on the amount of RAM available on your computer, the following maximum limits apply to expanded mode:

- RAM < 8 GB: A maximum of 10,000 rows are displayed.
- RAM >= 8 GB: A maximum of 40,000 rows are displayed.

See also

ARRAY (Page 283)

Rules for declaring the block interface (Page 6967)

Using tags within the program (Page 95)

Keywords (Page 89)

Properties of local tags and constants (Page 7009)

Setting the retentivity of local tags (Page 6971)

9.4.5 Declaring tags of STRUCT data type

Requirement

The block interface is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Select the desired declaration section in the block interface.
2. Enter a tag name in the "Name" column.
3. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.
   An empty, indented row is inserted after the new tag.
4. Insert the first structural element in the first empty row.
   An additional empty row is inserted after the element.
5. Select a data type for the structure element.

6. Optional: Change the properties of the structural element that is displayed in the other columns of the block interface.
7. Repeat the step 4 to 6 for all additional structure elements.
   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
8. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

Note

S7-1500: A maximum of 252 structures in one data block

A maximum of 252 structures is permitted in one data block. If you need additional structures, you must restructure your program. You can, for example, create the structures in several global data blocks.

See also

Declaration of STRUCT (Page 6995)

Rules for declaring the block interface (Page 6967)

Using tags within the program (Page 95)

Keywords (Page 89)

Properties of local tags and constants (Page 7009)

Setting the retentivity of local tags (Page 6971)

9.4.6 Declare variables of the STRING and WSTRING data types

Requirement

The block interface is open.

Procedure

To declare a variable the STRING or WSTRING data type, proceed as follows:

1. Select the desired declaration section in the block interface.
2. Enter a variable name in the "Name" column.

3. Enter "STRING" or "WSTRING" in the "Data type" column. You will be supported by autocompletion during input.
4. Optional: Specify the maximum length of the string using square brackets after the keyword STRING or WSTRING. If you do not specify a maximum length, the string has a default length of 254 characters.
   Example of a WSTRING with the maximum length of 4:
   `WSTRING[4]`
   Example of a string whose maximum length is defined by a local constant:
   `STRING[#My_local_const1]`
   Example of a string whose maximum length is defined by a global constant:
   `STRING["My_global_const1"]`

Result

The variable of the STRING or WSTRING data type is applied.

See also

Declaration of STRING and WSTRING (Page 7000)

9.4.7 Declaring tags based on a PLC data type

Requirement

A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a PLC data type, follow these steps:

1. Select the desired declaration section in the block interface.
2. Enter the PLC data type in the "Data type" column. You will be supported by Autocomplete during input.
   The tag is created.
3. If you have already defined default values or comments for the tags within a PLC data type during declaration of the PLC data type, these are shown in gray. You can change these values in the block interface.
   The changed values are displayed in black and apply only to the specific point of use.

Note

If you change or delete PLC data types that are used in the block interface, the interface becomes inconsistent. To eliminate this inconsistency, you must recompile the program.

See also: Updating the block interface (Page 7015)

9.4.8 Declaring higher-level tags

Introduction

To access data areas within a declared tag, you can overlay the declared tags with an additional declaration. This provides you with the option of addressing an already declared tag with a different data type. You can, for example, address the individual bits of a tag of WORD data type with an ARRAY of BOOL.

Overlaying tags

To overlay a tag with a new data type, follow these steps:

1. Open the block interface.
2. In the interface, select the tag that you want to overlay with a new data type.
3. Click "Add row" in the toolbar.
   A row is inserted after the tag to be overlaid. The overlaying tag must be declared in the row directly after the tag that is to be overlaid.
4. Enter a tag name in the "Name" column.
5. Enter the "AT" entry in the "Data type" column. You will be supported by Autocomplete in this step.
   The following is added to the entry in the "Name" column.
   `"AT<Name of the higher-level tag>"`
6. Click the data type selection button again and select the data type for the new tag.
   The tag is created. It points to the same data as the higher-level tag, however interprets this data with the new data type.

Removing overlay

To remove the overlay of a tag, follow these steps:

1. Select the higher-level tag in the block interface.
2. Select the "Delete" command in the shortcut menu.

The overlay is removed.

See also

Overlaying tags with AT (Page 111)

9.4.9 Declaring multi-instances

Requirement

The function block to be called exists in project tree and is multi-instance capable.

Declaring a multi-instance

To declare a function block to be called as a multi-instance, follow these steps:

1. Open the block interface of the calling function block.
2. In the "Name" column of the "Static" section, enter a designation for the block call.
3. In the "Data type" column, enter the symbolic name for the function block to be called.

Note

The program editor will declare the multi-instance automatically if you program a block call in a network and then specify in the "Call options" dialog that you want to call the block as a multiple instance.

Declaring multi-instances as ARRAY

To declare an ARRAY of multi-instances, follow these steps:

1. Open the block interface of the calling function block.
2. In the "Name" column of the "Static" section, enter a designation for the ARRAY.
3. Select the ARRAY data type in the "Data type" column.
   The "Array" dialog opens.
4. In the "Data type" input field, enter the name of the function block for which an instance should be defined. The block name must be in quotation marks.
5. In the "ARRAY limits" input field, specify the high and low limit for each dimension.
   Example of a one-dimensional ARRAY:[0..3]
   Example of a three-dimensional ARRAY:[0..3, 0..15, 0..33]
6. Confirm your entry.

See also

Updating the block interface (Page 7015)

Declaration of instances (Page 7001)

9.4.10 Declaring parameter instances

Introduction

The following table shows the options for declaring and using parameter instances in the block interface:

Declaration	Block type	Application
InOut	FB FC	You can then call the instance that you transfer as actual parameter within the function block.
Input	FC	You have read access to the data of the instance that you transfer as actual parameter. However, you cannot call the instance from within the block.
Output	FC	You have write access to the data of the instance that you transfer as actual parameter. However, you cannot call the instance from within the block.

Requirement

The block interface is open.

Procedure

To declare a parameter instance, follow these steps:

1. Open the required section of the block interface (see table).
2. In the "Name" column, enter a designation for the in/out parameter to which the instance is transferred.
3. In the "Data type" column, enter the name of the function block for which an instance should be defined. The block name must be in quotation marks.

Note

Declaring parameter instances automatically

The program editor will declare the parameter instance automatically if you program a block call in a network and then specify in the "Call options" dialog that you want to call the block as a parameter instance.

See also

Parameter instances (Page 68)

Declaration of instances (Page 7001)

9.4.11 Hiding parameters during the block call

Introduction

You can hide the block parameters when calling the block in LAD or FBD. Hidden parameters are not visible initially, but can be displayed via a small arrow at the bottom edge of the call box.

Two options are available for hiding a parameter:

- Hide
  The parameter is always hidden.
- Hide, if no parameter is assigned
  The parameter is hidden as long as it has not yet been interconnected. If you assign an actual parameter, the parameter is displayed at the call box.

Procedure

Proceed as follows to define whether a block parameter is to be displayed or hidden during the call:

1. Open the block interface.
2. Select a parameter in the block interface.
3. Open the "Properties" tab in the Inspector window.
4. Select the "Attributes" group in the area navigation.
5. Select one of the following options under "Visibility at block calls in LAD / FBD":
   - "Show"
   - "Hide"
   - "Hide, if no parameter is assigned"

Result

The parameter is displayed or hidden at the block call in LAD or FBD depending on the setting.

If a syntax error is found during the block call, the parameter cannot be hidden.

See also

Wiring hidden parameters (Page 7160)

Declaring predefined actual parameters (Page 6987)

9.4.12 Declaring predefined actual parameters

Introduction

You can specify during the declaration of a block parameter which actual parameter is to be used during the block call.

This can be useful if you are using program blocks as library elements and want to store information about the actual parameters to be used along with the library element.

In addition you have the option in LAD and FBD to hide parameters that have a valid predefined actual parameter during the block call. Hidden parameters are not visible initially, but can be displayed via a small arrow at the bottom edge of the call box.

Two options are available for hiding a predefined parameter:

- The parameter is always hidden.
- The parameter is hidden as long as the predefined actual parameter is assigned. However, when a different parameter is assigned, the parameter becomes visible.

Requirement

The parameter is located in the section "Input", "Output" or "InOut".

Procedure

Proceed as follows to predefine an actual parameter:

1. Open the block interface.
2. Select a parameter in the block interface.
3. Open the "Properties" tab in the Inspector window.
4. Select the "Attributes" group in the area navigation.
5. Enter the required actual parameter in the "Predefined actual parameter" input box.
6. Select the "Hide" option under "Visibility at block calls in LAD / FBD".
7. As an additional option, select "Show if parameter assigned on block call is not identical to the predefined actual parameter". This option is only available if the "Hide" option was activated beforehand.

Result

- An actual parameter is predefined. If you save the program block as a library element, it thus also contains information about the actual parameter to be used.
- If the library element is used in the program, a check is made to determine whether the actual parameter you have predefined can be addressed. If so, it is automatically used as the actual parameter.
- If the actual parameter If the actual parameter is not found in the program, a syntax error is signaled and you have to carry out the parameter assignment manually. Parameters with an invalid assignment are not hidden.

See also

Hiding parameters during the block call (Page 6986)

9.4.13 Edit table of the block interface

9.4.13.1 Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

9.4.13.2 Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

9.4.13.3 Deleting tags

Procedure

To delete a tag, follow these steps:

1. Select the row with the tag to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

9.4.13.4 Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

9.4.13.5 Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.
5. To hide or show several columns, click "More" and activate or deactivate the check box of the corresponding columns in the "Show/Hide" dialog.

9.4.13.6 Editing tags with external editors

To edit individual tags in external table editors, such as Excel, you can export or import these tags using copy and paste.

Note

Restriction when displaying and copying ARRAY elements

Depending on the amount of work memory available on your computer, the following maximum limits apply when displaying and copying ARRAYs from the block interface in an Excel table:

- RAM < 8 GB: A maximum of 10,000 elements are displayed and copied.
- RAM >= 8 GB: A maximum of 40,000 elements are displayed and copied.

Requirements

The block interface and an external editor are opened.

Procedure

To export individual tags to an external editor and import them again, follow these steps:

1. Click the "Expanded mode" button to show all elements of structured data types.
2. Select one or more tags.
3. Select "Copy" in the shortcut menu.
4. Switch to the external editor and paste the copied tags.
5. Edit the tags as required.
6. Copy the tags in the external editor.
7. Select the tags in the external editor.
8. Switch back to the block interface.
9. Select "Paste" in the shortcut menu.

9.5 Declare block interface in the textual view

9.5.1 Structure of the textual block interface

Structure of the textual block interface

The following figure shows the structure of the textual block interface, as it is appears after a new function block is created:

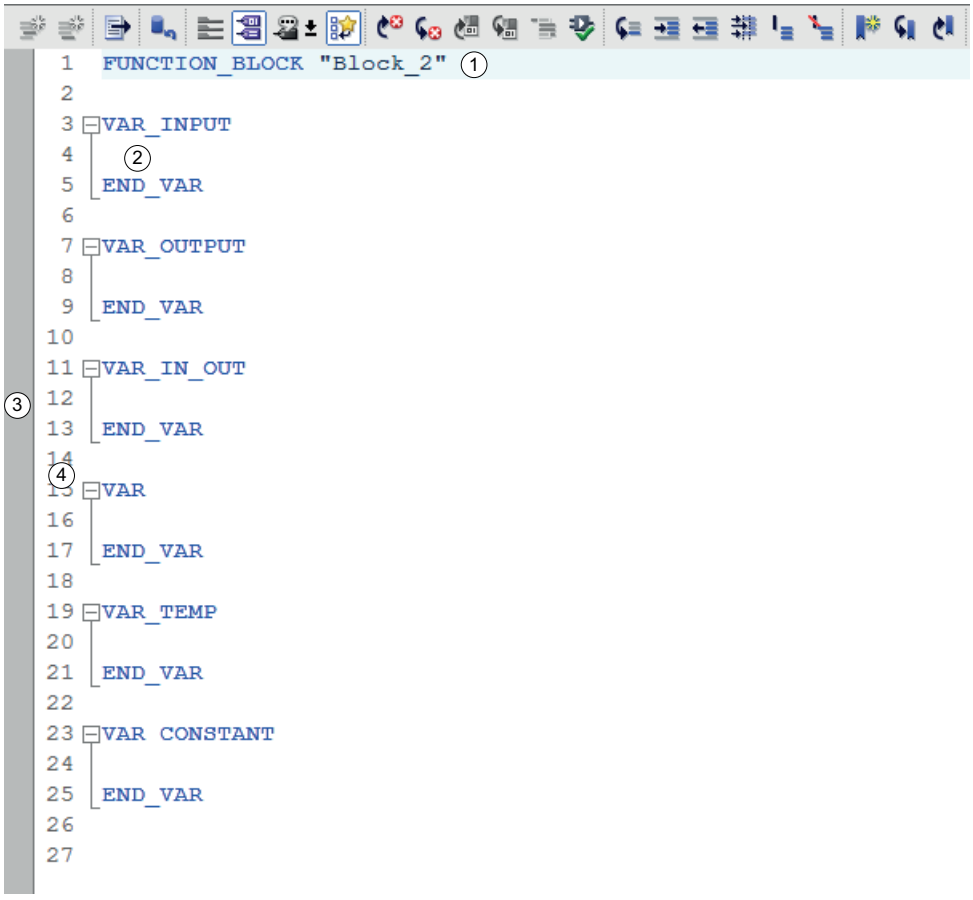

The textual block interface consists of the following areas:

Area	Meaning
① Block declaration	The block declaration defines the name and type of block.
② Declaration sections	Declare the parameters and local data of the block in the declaration sections. The declaration sections are already entered in the empty block interface. However, you can move, copy or delete the sections any way you wish.

Area	Meaning
③ Sidebar	You can set bookmarks in the sidebar.
④ Line numbers	The line numbers are displayed to the left of the declaration sections.

See also

Selecting tabular or textual representation of the block interface (Page 6973)

9.5.2 Block declaration and return value

Block declaration

The textual block interface begins with the block declaration. You do not have to enter the block declaration yourself. It is already included in the template of the interface.

Return value

For functions (FC), you specify the return value of the function block after the declaration. The return value is the value that is returned to the calling block. If you specify a "VOID" value, the function does not return a value.

Syntax

The following syntax is used for the block declaration and the return value:

Block type	Syntax	Example
Function (FC)	FUNCTION <Name> [:Data type return value>] <Declaration>	`FUNCTION "My_Function" : Int` `FUNCTION "My_Function" : Void`
Function block (FB)	FUNCTION_BLOCK <Name> <Declaration>	`FUNCTION_BLOCK "My_FunctionBlock"`
Organization block (OB)	ORGANIZATION_BLOCK <Name> <Declaration>	`ORGANIZATION_BLOCK "My_OrganizationBlock"`

See also

Example of a textual block interface (Page 7007)

9.5.3 Declaration sections

Declaration sections

The textual block interface is divided into different declaration sections, which are characterized by keyword pairs. Different sections are allowed depending on the type of block.

The order of declaration sections is not of importance. A section can occur several times in the block interface.

The following syntax is used for the declaration section:

Declaration section	Syntax
Input parameters	VAR_INPUT [<ATTRIBUTE>] <Declaration> END_VAR
Output parameters	VAR_OUTPUT [<ATTRIBUTE>] <Declaration> END_VAR
In/out parameters	VAR_IN_OUT [<ATTRIBUTE>] <Declaration> END_VAR
Temporary local data	VAR_TEMP <Declaration> END_VAR
Static local data	VAR [<ATTRIBUTE>] <Declaration> END_VAR
Constant	VAR CONSTANT <Declaration> END_VAR

You can find information on the individual declaration sections at:

Overview of the block interface (Page 6965)

Attributes for declaration section

Optionally, you can assign attributes to sections. The attributes apply to all variables that are declared in this section. If a section appears more than once, you can use other attributes for each occurrence.

The following attributes are available:

Attribute	Meaning
RETAIN	The tags in this section are retentive, i.e. their values are still available after a power failure.
NON_RETAIN	The tags in this section are not retentive, i.e. their values are lost after a power failure.
DB_SPECIFIC	Retentivity can be set in the instance data block

If you do not specify an attribute, the data of the declaration section is non-retentive.

See also

Setting the retentivity of local tags (Page 6971)

Example of a textual block interface (Page 7007)

9.5.4 Variable declaration and initialization

Declaration and initialization of variables or constants

You declare the local variables and constants that are used within the block in the individual declaration sections.

A variable declaration consists of a symbolic name and a data type. Optionally, you can specify a default value for initializing the variable.

A constant declaration always contains a value besides the symbolic name and the data type. Constant declarations are only permitted in the "VAR CONSTANT" section.

Overlaying variables with AT

To access data areas within a declared variable, you can overlay the declared variables with an additional declaration. This provides you with the option of addressing an already declared variable with a different data type. You can, for example, address the individual bits of a variable of WORD data type with an ARRAY of BOOL.

While an AT declaration must always be directly after the referenced tag in the table view, it can be at any position in the text view.

AT declarations cannot be initialized. Note that AT declarations can only be used in non-optimized blocks.

Syntax

The following syntax is used to declare a variables and constants:

Declaration	Syntax	Examples
Tag	<Name> : <Data type> [:= <Value>];	`myBit : BOOL;` `myBit : BOOL := true;`

Constant	<Name> : <Data type> := <Value>;	`PI : REAL := 3.141592;`
		`myInt: INT := INT#16#7FFF;`
		`myString: STRING := 'hello';`

AT declaration	<Name> AT <Name of the referenced variable> : <Data type>;	`myReferenceToVar2 AT Var_2 : Int;`

See also

Overview of the valid data types (Page 231)

General rules for declaring the block interface (Page 6967)

Overlaying tags with AT (Page 111)

Declaring higher-level tags (Page 6983)

Example of a textual block interface (Page 7007)

Constants (Page 97)

9.5.5 Declaration of STRUCT

Declaration of STRUCT

The STRUCT data type represents a data structure that consists of a fixed number of components of different data types.

A structure can also be created based on a PLC data type (UDT). To do this, assign the PLC data type as a data type of the structure. A tooltip shows you the individual elements of the PLC data type.

Syntax

The following syntax is used to declare a STRUCT:

Declaration	Syntax	Examples

STRUCT	<Name> : Struct
	<Element name> : <Data type> [:= <Value>];
	<Element name> : <Data type> [:= <Value>];
	...
	END_Struct := (<Initialization list>);
		`myStruct : Struct`
		`mem_1 : Int;`
		`mem_2 : Int;`
		`mem_3 : BOOL;`
		`END_Struct := (6,0, TRUE) ;`

STRUCT based on a PLC data type (UDT)	<Name> : "<UDT name>" := (<initialization list>);	`myStruct : "myType" := (6,0, TRUE);`

Note

S7-1500: A maximum of 252 structures in one data block

A maximum of 252 structures is permitted in one data block in CPUs of the S7-1500 series. If you need additional structures, you must restructure your program. You can, for example, create the structures in several global data blocks.

Initialization of STRUCT

The initialization of STRUCT is optional. For initialization, enter a value directly after the tag declaration or use an initialization list.

An initialization list is placed after the keyword END_STRUCT in parentheses and := introduces it. If the structure contains lower-level structures, use nested brackets for initialization. If you unintentionally use both options for the initialization, the value from the initialization list is used.

The following syntax is used for the initialization list:

Declaration	Syntax	Examples	Comment

Initialization of STRUCT	<Name> : Struct
	<Element name> : <Data type> [:= <Value>];
	...
	END_Struct
		`myStruct : Struct`
		`mem_1 : Int := 1;`
		`mem_2 : Int := 2;`
		`mem_3 : BOOL := FALSE`
		`End_Struct`
			The structure elements are initialized as follows:
			// mem_1 := 1
			// mem_2 := 2
			// mem_3 := FALSE

Initialization list for STRUC	:= (<Value>,<Value>,<Value>...)	`myStruct : Struct`
		`mem_1 : Int;`
		`mem_2 : Int;`
		`mem_3 : BOOL`
		`End_Struct:=(2,0,TRUE);`
			The structure elements are initialized as follows:
			// mem_1 := 2
			// mem_2 := 0
			// mem_3 := TRUE

Declaration	Syntax	Examples	Comment

Initialization list for nested STRUC	:= (<Value>,(<Value>,<Value>...))	`myStruct : Struct`
		`mem_1 : Int;`
		`mem_2 : Int;`
		`mem_3 : Struct`
		`mem_4 : BOOL;`
		`mem_5 : BOOL;`
		`End_Struct;`
		`End_Struct:=(2,0, (TRUE,TRUE));`
			The structure elements are initialized as follows:
			// mem_1 := 2
			// mem_2 := 0
			// mem_4 := TRUE
			// mem_5 := TRUE

Initialization list for STRUCT with element name specification	:= (<Element name> := <Value>), (<Element name> := <Value>)...	`myStruct : Struct`
		`mem_1 : Int;`
		`mem_2 : Int;`
		`End_Struct:=(mem_2:=55);`
			The structure elements are initialized as follows:
			// mem_2 := 55

Initialization list for PLC data type (UDT)	:= (<Value>,<Value>,<Value>...)	`myStruct : "myType" := (2,0,TRUE);`	The structure elements are initialized as follows:
			// mem_1 := 2
			// mem_2 := 0
			// mem_3 := TRUE

Initialization list for nested PLC data type (UDT)	:= (<Value>,(<Value>,<Value>...))	`myStruct : "myType" := (2,0, (TRUE,TRUE);`	The structure elements are initialized as follows:
			// mem_1 := 2
			// mem_2 := 0
			// mem_3 := TRUE
			// mem_4 := TRUE

Initialization list for PLC data type (UDT) with element name specification	:= (<Element name> := <Value>), (<Element name> := <Value>)...	`myStruct : "myType" := (mem_1:=22,mem_2:=55);`	The structure elements are initialized as follows:
			// mem_1 := 22
			// mem_2 := 55

See also

Declaring tags of STRUCT data type (Page 6980)

STRUCT data structure (anonymous structures) (Page 278)

Rules for declaring the block interface (Page 6967)

Example of a textual block interface (Page 7007)

9.5.6 Declaration of ARRAY

Declaration of ARRAY

The ARRAY data type represents a data structure that consists of a fixed number of elements of the same data type.

The following syntax is used to declare an ARRAY:

Declaration	Syntax	Examples
ARRAY	<Name> : ARRAY [Low limit..High limit] OF <Data type> := [<Initialization list>];	`MyARRAY_1 : ARRAY[0..7] OF BOOL;` `MyARRAY_1 : ARRAY[0..7] OF BOOL := [1,1,0,0,0,1,0,0];`
ARRAY with variable limits	<Name> : ARRAY [*] OF <Data type>;	`MyARRAY_1 : ARRAY[*] OF INT;` `MyARRAY_2 : ARRAY[*, *, *] OF INT;`

ARRAY with local constants as limits	<Name> : ARRAY [#<Constant name>..#<Constant name>] OF <Data type> := [<Initialization list>];	`MyARRAY_1 : ARRAY[#LocConst1..#LocConst2] OF INT;`
		`MyARRAY_2 : ARRAY[1..#LocConst] OF INT;`
		`MyARRAY_3 : ARRAY[1..#LocConst] OF INT := [1,1,0,0];`

ARRAY with global constants as limits	<Name> : ARRAY ["<Constant name>".."<Constant name>"] OF <Data type> := [<Initialization list>];	`MyARRAY_1 : ARRAY["GlobConst1".."GlobConst2"] OF INT;`
		`MyARRAY_2 : ARRAY[1.."GlobConst", 2..5,#l..#u] OF INT;`
		`MyARRAY_3 : ARRAY[1.."GlobConst"] OF INT:= [1,1,0,0];`

ARRAY OF STRUCT	<Name> : ARRAY[Low limit..high limit] OF Struct
	<Element name> : <Data type>;
	<Element name> : <Data type>;
	...
	END_Struct := [<Initialization list>];
		`MyARRAY_1 : Array[0..1] OF Struct`
		`mem_1 : Int;`
		`mem_2 : Int;`
		`END_STRUCT := [ (2,4),(22,44) ];`

ARRAY OF UDT	<Name> : ARRAY[Low limit..High limit] OF "<UDT_Name>" := [<Initialization list>];	`MyARRAY_1 : Array[0..1] OF MyType := [ (2,4),(22,44) ];`

Initialization of ARRAY

The initialization of an ARRAY is optional. You use an initialization list for this. It is enclosed in square brackets after the declaration and starts with :=. The list contains comma-separated values for all elements of the ARRAY. You can initialize a maximum of 10000 elements.

Optionally, you can use repetition factors in parentheses to initialize several successive elements with the same value. The repetition factor must be a positive integer.

With an ARRAY OF STRUCT, you optionally also the opportunity to initialize element by element. You have centrally assign all structural elements of a type a value.

The following syntax is used for the initialization list:

Declaration	Syntax	Examples	Comment

Initialization list	:= [<Value>,<Value>,<Value>...]	`myArray : ARRAY[0..2] OF BOOL := [1,1,0];`	The ARRAY elements are initialized as follows:
			// myArray[0] := 1
			// myArray[1] := 1
			// myArray[2] := 0

Initialization list with repetition factor	:= [<Value>,<Repetition factor>(Value),<Value>...]	`myArray : ARRAY[1..2, 1..3] OF INT := [9,8,3(10),6];`	The ARRAY elements are initialized as follows:
			// myArray[1,1] := 9
			// myArray[1,2] := 8
			// myArray[1,3] := 10
			// myArray[2,1] := 10
			// myArray[2,2] := 10
			// myArray[2,3] := 6

Initialization list for ARRAY of STRUCT	:= [(< Value list element 1>), (< Value list element 2>)...]	`myArray : Array[0..1] OF Struct`
		`  Element1 : Int;`
		`  Element2 : Int;`
		`END_Struct := [ (2,4), (22,44) ];`
			The ARRAY elements are initialized as follows:
			// myArray[0].Element1 := 2
			// myArray[0].Element2 := 4
			// myArray[1].Element1 := 22
			// myArray[1].Element2 := 44

Initialization list for ARRAY of STRUCT with element name specification	:= [(<Element name> := <Value>), (<Element name> := <Value>)...]	`myArray : Array[0..1] OF Struct`
		`  Element1 : Int;`
		`  Element2 : Int;`
		`END_Struct := [ (Element1 := 2 , Element2:=4), (Element1 := 22, Element2:=44) ];`
			The ARRAY elements are initialized as follows:
			// myArray[0].Element1 := 2
			// myArray[0].Element2 := 4
			// myArray[1].Element1 := 22
			// myArray[1].Element2 := 44

Initialization list for ARRAY of STRUCT with repetition factor	:= [<Repetition factor>(Value list element 1),<Repetition factor>(Value list element 2),...]	`myArray : Array[0..1, 0..1] OF Struct`
		`  Element_x : Int;`
		`  Element_y : Int;`
		`End_Struct:=[2((1, 11)), 2((2, 22))];`
			The ARRAY elements are initialized as follows:
			// myArray[0,0].Element_x:= 1
			// myArray[0,0].Element_y:= 11
			// myArray[0,1].Element_x:= 1
			// myArray[0,1].Element_y:= 11
			// myArray[1,0].Element_x:= 2
			// myArray[1,0].Element_y:= 22
			// myArray[1,1].Element_x:= 2
			// myArray[1,1].Element_y:= 22

Declaration	Syntax	Examples	Comment

Initialization list for ARRAY of UDT	:= [(Value list element 1>), (<Value list element 2>)...]	`myArray : Array[0..1] OF MyType := [ (2,4),(22,44) ];`	The ARRAY elements are initialized as follows:
			// myArray[0].UDT-Element1 := 2
			// myArray[0].UDT-Element2 := 4
			// myArray[1].UDT-Element1 := 22
			// myArray[1].UDT-Element2 := 44

Initialization list for ARRAY of UDT with element name specification	:= [(<Element name> := <Value>), (<Element name> := <Value>)...]	`myArray : Array[0..1] OF MyType := [ (UDT-Element1 := 2,UDT-Element2:=4),(UDT-Element1 := 22,UDT-Element2:=44) ];`	The ARRAY elements are initialized as follows:
			// myArray[0].UDT-Element1 := 2
			// myArray[0].UDT-Element2 := 4
			// myArray[1].UDT-Element1 := 22
			// myArray[1].UDT-Element2 := 44

Initialization list for ARRAY of UDT with repetition factor	:= [<Repetition factor>(Value list element 1),<Repetition factor>(Value list element 2),...]	`myArray : Array[0..1] OF myType :=[2((),1))];`	The ARRAY elements are initialized as follows:
			// myArray[0].UDT-Element1 := 0
			// myArray[0].UDT-Element2 := 1
			// myArray[1].UDT-Element1 := 0
			// myArray[1].UDT-Element2 := 1

See also

ARRAY (Page 283)

Rules for declaring the block interface (Page 6967)

Example of a textual block interface (Page 7007)

9.5.7 Declaration of STRING and WSTRING

Declaration of STRING and WSTRING

The STRING and WSTRING data types store multiple characters in a string. Any character of the ASCII code is permitted in the string. The characters are given in single quotes.

Optionally, you can specify a default value for initializing the string.

The maximum length of the character string can be specified during the declaration of an operand using square brackets after the keyword STRING or WSTRING (for example, STRING[4]). To declare the maximum length, you can enter an absolute value or use local or global constants.

If the specification of the maximum length is omitted, the standard length of 254 characters is set for the respective operand.

Syntax

The following syntax is used to declare a STRING and WSTRING:

Declaration	Syntax	Examples
STRING	`<Name> : STRING [:= <Value>];`	`myString: STRING;` `myString: STRING := 'hello';`
WSTRING	`<Name> : WSTRING [:= <Value>];`	`myWstring: WSTRING;` `myWstring_var: WSTRING := 'helloWorld';`

STRING with defined maximum length	`<Name> : STRING[[Constant]];`	`myString: STRING[10];`
		`myString: STRING["globConst"];`
		`myString: STRING[#locConst];`

See also

STRING (Page 264)

WSTRING (Page 267)

Declare variables of the STRING and WSTRING data types (Page 6981)

Rules for declaring the block interface (Page 6967)

Example of a textual block interface (Page 7007)

9.5.8 Declaration of instances

Declaration of multi-instances

You declare multi-instances in the "VAR ...END_VAR" section of the textual block interface. Simply enter the name of the called block as the data type.

Multi-instances can also be created as an ARRAY.

See also:

Multi-instances (Page 65)

Declaring multi-instances (Page 6983)

Declaration of parameter instances

It is best to declare parameter instances as in/out parameters in "VAR_IN_OUT...END_VAR" of the textual block interface. However, they are also possible as input or output parameters in functions (FC). Simply enter the name of the called block as the data type.

See also:

Parameter instances (Page 68)

Declaring parameter instances (Page 6985)

Instance	Syntax	Examples	Meaning

Multi-instance	VAR
	<Multi-instance name>:"<Block name>";
	END_VAR
		`VAR`
		`instFB : "MyFB";`
		`END_VAR`
			"instFB" is a multi-instance of the "MyFB" block.

ARRAY of multi-instances	VAR
	<Multi-instance name>: ARRAY [n...m] of "<Block name>";
	END_VAR
		`VAR`
		`instArray : ARRAY [0…4] of "MyFB";`
		`END_VAR`
			"instArray" is an array of multi-instances of the "MyFB" block.

Parameter instance	VAR
	<Parameter instance name>:"<Block name>";
	END_VAR
		`VAR_IN_OUT`
		`instParam:"MyFB";`
		`END_VAR`
			"instParam" is a parameter instance of the "MyFB" block.

See also

Example of a textual block interface (Page 7007)

9.5.9 Attributes for variables

Attributes for variables

Optionally, you can assign attributes to each variable. If you do not explicitly assign an attribute to a variable, it has the preassigned attribute values. You can change the default settings of some attributes in the general settings for PLC programming.

The following attributes are available:

Attribute	Meaning	Default
Visible in HMI engineering	Indicates whether a variable is visible by default in the HMI selection list.	TRUE
Accessible from HMI / OPC UA	Indicates whether HMI/OPC UA can access this variable during runtime.	TRUE

Attribute	Meaning	Default
Writable from HMI/OPC UA	Indicates whether the tag from HMI/OPC UA can be written during runtime.	FALSE
Setpoint	Marks a variable as a setpoint. Setting values are the values that will probably have to be fine tuned during commissioning. The column is only available in the interface of function blocks.	FALSE

Syntax

The following syntax is used for attributes:

Attribute	Syntax	Examples
Visible in HMI engineering	<Name> {EXTERNALVISIBLE:=<value>}: <Data type>;	`myInt {EXTERNALVISIBLE:= 'False'} : INT;`
Accessible from HMI / OPC UA	<Name> {EXTERNALACCESSIBLE:=<value>}: <Data type>;	`myInt {EXTERNALACCESSIBLE:= 'False'} : INT;`
Writable from HMI/OPC UA	<Name> {EXTERNALWRITABLE:=<value>}: <Data type>;	`myInt {EXTERNALWRITABLE:= 'False'} : INT;`
Setpoint	<Name> {S7_SETPOINT:=<value>}: <Data type>;	`myInt {S7_SETPOINT:= 'False'} : INT;`

See also

Example of a textual block interface (Page 7007)

General settings for the PLC programming (Page 6861)

9.5.10 Comments

Comments

There are several ways to comment a variable in the textual block interface:

- Line comment
  A line comment starts with "//" and extends to the end of the line.
- Comment section
  A comment section is started with "(*" and ended with "*)". It can span several lines.

The following rules apply to comments:

- Comments cannot be nested.
- Multilingual comments are not possible in the textual block interface.
- Comments are not permitted between the block declaration and the first declaration section.

Syntax

The following syntax is used for comments:

Attribute	Syntax	Examples
Line comment	<Declaration>; //<Comment>	`MyInt : INT; // this is my comment`

Comment section	<Declaration>; (*<Comment>*)	`MyInt : INT; (* these are my comments:`
		`1st comment`
		`2nd comment*)`

See also

Editing multilingual comments in the block interface (Page 7013)

Example of a textual block interface (Page 7007)

9.5.11 Edit textual block interface

When editing the textual block interface, you can use the same editing functions as those for editing the program code.

Use autocomplete

1. Enter the syntax of the declaration on the keyboard.
   The autocomplete opens during the entry. It offers all the syntax elements that are allowed at the current location.
2. Select the desired element from the autocomplete.

See also: Entering SCL instructions manually (Page 7329)

Automatically format lines

1. Select the text that you want to format or place the insertion point in the appropriate line.
2. Select the "Format selected text automatically" button into the toolbar of the programming editor.
   All selected lines are formatted as long as they are syntactically correct.

See also: Formatting SCL code (Page 7321)

Indenting or outdenting lines

1. Click in the line that you want to indent or outdent.
2. Press the "Indent text", "Outdent text" button into the toolbar of the programming editor.
   You can set the width of the indent in "Options > Settings".

See also: Formatting SCL code (Page 7321)

Expanding and collapsing sections of code

1. Click the minus sign in the outline view.
   The code section closes.
2. Click the plus sign in the outline view.
   The code section opens.

See also: Expanding and collapsing sections of code (Page 7322)

Using bookmarks

1. Click on the line in which you want to place the bookmark.
2. Click the "Set/delete bookmark" button in the toolbar.

Note

Navigation view between interface and program code

If you have set bookmarks both in the interface and in the program code, you cannot use the commands "Go to next/previous bookmark" to navigate between the two windows.

See also: Using bookmarks (Page 7324)

Copying or moving tags between textual and tabular interfaces

You can copy or move tags between a textual interface and a tabular interface.

1. Select one or more tags in a declaration section.
2. In the shortcut menu, select the "Copy" or "Cut" command.
3. Place the mouse pointer at the position where you want to insert the tag.
4. Select "Paste" in the shortcut menu.

Note

Copying or moving structured data types with initializations

When nested structured data types (for example, ARRAY of STRUCT) are copied from a textual interface to a tabular interface, the default values of the lower-level structures are not copied or moved.

9.5.12 Restore default interface of an organization block

Introduction

Organization blocks have defined interface parameters that include the start information of the OB. If you change or delete these predefined parameters in the textual block interface, you have the option to restore the defined standard interface with optimized organization blocks.

Procedure

To restore the default interface of an organization block, proceed as follows:

1. Select the "Restore default interface" command in the shortcut menu.

Result

- Deleted or changed OB parameters are restored.
- Parameters which you have added manually are not deleted but are retained.

9.5.13 Example of a textual block interface

Example

The following figure shows an example of a textual block interface:

```
1   FUNCTION_BLOCK "MyFB"
2
3   VAR_INPUT
4      myIn1 : Int := 5; //Declaration of an Input tag with initialization
5   END_VAR
6
7   VAR_INPUT RETAIN
8      myIn2 {EXTERNALACCESSIBLE := 'False'; EXTERNALVISIBLE := 'False'} : B
9      //Declaration of an Input tag with attributes
10  END_VAR
11
12  VAR_OUTPUT RETAIN
13     myOut : Array[0..2] OF Byte; //Declaration of an ARRAY
14  END_VAR
15
16  VAR_IN_OUT
17
18  END_VAR
19
20  VAR
21     myStatic : "FB_SCL"; //Declaration of a multi-instance
22  END_VAR
23
24  VAR_TEMP
25
26  END_VAR
27
28  VAR CONSTANT
29     myConstant : Byte := 44; // Declaration of a constant
30  END_VAR
31
```

9.6 Editing the properties of local tags and constants

9.6.1 Properties of local tags and constants

Properties

The table below provides an overview of the properties of local tags and constants:

Group	Property	Description
General	Name	Name of the element
	Data type	Data type of the element
	Default value	Value with which you can pre-assign specific tags in the interface of the code block, or the value of a local constant. Specification of the default value is optional for tags. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL. The default value of a tag is applied as the start value in the corresponding instance. You can then replace these adopted values with instance-specific start values.
	Comment	Comment for the element

Group	Property	Description
Attributes	Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This attribute is only available in the interface of the function block with optimized access.
	Visible in HMI engineering	Shows whether the tag is visible by default in the HMI selection list.
	Writable from HMI/ OPC UA	Indicates whether the tag from HMI/OPC UA can be written during runtime.
	Accessible from HMI/OPC UA	Indicates whether HMI/OPC UA can access this tag during runtime. Please note, however, that you cannot implement general access protection for the tag with the "Accessible from HMI/OPC UA" attribute. Read or write access from other applications is possible even if the attribute is not enabled.
	Visibility at block calls in LAD/FBD	Show: The parameter is always visible at block calls in LAD or FBD.
		Hide: The parameter is always hidden at block calls in LAD or FBD. The instruction box then has a small arrow on the lower edge. You can have the parameters displayed and interconnect them by clicking this arrow.
		Hide if no parameter is assigned: The parameter is always hidden at block calls in LAD or FBD as long as it has not yet been interconnected. If you assign an actual parameter, the parameter is displayed at the call box.
	Predefined actual parameter	Defines a parameter that is to be used as actual parameter during the block call.
	Show if the assigned actual parameter is not identical with the predefined actual parameter	The parameter is hidden at block calls in LAD or FBD as long as the predefined actual parameter has not yet been assigned. However, when a different parameter is assigned, the parameter becomes visible. This option is only active if you have selected the "Hide" parameter and have defined a predefined actual parameter.

Group	Property	Description
User-defined attributes	CFC_Configurable	Configurable Indicates whether a parameter is configurable in CFC.
	CFC_ForTest	For test Indicates whether a parameter is registered for the CFC test mode.
	CFC_Visible	Visible Indicates whether a parameter is visible in CFC.
	CFC_Interconnectable	Interconnectable Indicates whether a parameter is interconnectable in CFC.
	CFC_EnableTagReadback	Enable tag readback Indicates whether a parameter is relevant for the "Read back chart" function in CFC.
	CFC_EnumerationTexts	Enumeration texts Assigns a parameter to an enumeration in CFC.
	CFC_EngineeringUnit	Engineering unit Assigns a parameter to a unit in CFC.
	CFC_LowLimit	Low limit Defines the low limit for the parameter in CFC.
	CFC_HighLimit	High limit Defines the high limit for the parameter in CFC.

See also

Setting the retentivity of local tags (Page 6971)

Changing properties of local tags and constants (Page 7011)

Keywords (Page 89)

Hiding parameters during the block call (Page 6986)

Declaring predefined actual parameters (Page 6987)

9.6.2 Changing properties of local tags and constants

Editing properties of an element in the tabular block interface

To edit the properties of an element in the block interface, follow these steps:

1. Open the block interface.
2. Select the required element in the table.
3. Change the entries in the columns.

Editing properties of multiple elements in the tabular block interface

You can also simultaneously set or reset some properties for multiple selected elements.

To change one of these properties for several elements, follow these steps:

1. Open the block interface.
2. Hold down the <Ctrl> key.
3. In the required column, select each of the table cells whose value you want to change.
4. Select the "Set <property>" or "Reset <property>" command in the shortcut menu.

Editing properties in the properties window

To edit the properties of an individual tag or constant, follow these steps:

1. Select the tag or constant.
   The properties of the element are shown in the Inspector window.
2. Change the entries in the inspector window.

Renaming tags directly in the program editor

To rename one or more elements, follow these steps:

1. Select one or more elements in the program.
2. Select the "Rename tag" command in the shortcut menu.
   The "Rename tag" dialog opens. It displays a declaration table with the selected elements.
3. Change the entries in the "Name" column.
4. Confirm the input by clicking the "Change" button.

Editing the data type or comment in the program editor

Proceed as follows to edit the data type or tag comment in the program editor:

1. Select the name of the tag.
2. Select the "Rewire tag" command in the shortcut menu.
   The "Rewire tag" dialog will open. The dialog shows a declaration table.
3. Change the entry in the "Data type" or "Comment" columns.
4. Click the "Change" button to confirm the input.

Effect in the program

In case of a change of the name, data type or address of a tag or constant, each location of use of the tag is automatically updated in the program.

Note

If you change the interface of a block, the program may become inconsistent. The inconsistencies are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent calls are marked in red. You than have to manually updated the inconsistencies.

See also:

Updating block calls in LAD (Page 7144)

Updating block calls in FBD (Page 7207)

See also

Overview of the block interface (Page 6965)

Properties of local tags and constants (Page 7009)

Setting the retentivity of local tags (Page 6971)

Basic information on start values (Page 7035)

Using tags within the program (Page 95)

Keywords (Page 89)

Updating the block interface (Page 7015)

Edit table of the block interface (Page 6988)

Rules for declaring the block interface (Page 6967)

9.6.3 Editing multilingual comments in the block interface (S7-1500)

You can edit the comments from the block interface in all project languages that you have activated in the project language settings. You can find additional information on activating project languages under AUTOHOTSPOT.

Note that the texts must not exceed a length of 32767 Unicode characters even after translation.

Multilingual comments are not possible in the textual block interface.

Requirement

- You have activated multiple project languages.
- The block interface is open and it contains at least one comment.
- The block does not have know-how protection.

Procedure

To edit a comment in all project languages, follow these steps:

1. Click on the comment whose translation you want to edit.
   If you want to edit the translations of several comments at the same time, select all the desired comments.
2. Open the "Properties" tab in the Inspector window.
3. Open the "Texts" tab.
   The "Texts" tab displays the selected comments in the active project languages and, if available, the translations of the comments.
4. You can edit the translations directly in the table in the "Texts" tab.
5. To edit the translations with an external editor, you can export the displayed texts to OOXML format using the "Export/Import project texts" buttons.

Note

Editing all project texts in the global "Project texts" table

You can also edit the translations for the individual project languages in the global "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts". It contains all translatable texts of the entire project.

You can find additional information on translation of texts under AUTOHOTSPOT.

9.7 Updating the block interface

Introduction

If you change or delete PLC data types or multiple instances that are used in the block interface, the interface will become inconsistent. To remedy this inconsistency, the interface has to be updated.

You have two options for updating the block interface:

- Explicit updating of the block interface.
  The used PLC data types and multiple instances will be updated. The instance data blocks that belong to the block are not implicitly updated during this process.
- Implicit updating during compilation.
  All used PLC data types and multiple instances as well as the related instance data blocks will be updated.

Explicit updating of the block interface

To explicitly update the block interface, follows these steps:

1. Open the block interface.
2. Select the "Update" command in the shortcut menu.

Implicit Updating during Compilation

Proceed as follows to implicitly update all uses of PLC data types and multiple instances as well as the instance data blocks during compilation:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Declaring tags based on a PLC data type (Page 6982)

Basic information on start values (Page 7035)

Using tags within the program (Page 95)

Keywords (Page 89)

Properties of local tags and constants (Page 7009)

Setting the retentivity of local tags (Page 6971)

Updating block calls in LAD (Page 7144)

Declaring multi-instances (Page 6983)

Edit table of the block interface (Page 6988)

Rules for declaring the block interface (Page 6967)

9.8 Extending the block interface (S7-1200, S7-1500)

Description

In order to enable the editing of PLC programs that have already been commissioned and that are running without error on a system, CPUs of the S7-1500 series and most CPUs of the S7-1200 V4 and higher series support the option of extending the interfaces of function blocks during runtime.

You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Principle of operation

Each function block is always assigned a default memory reserve. The memory reserve is not used initially. Activate the memory reserve if you have compiled and loaded the block and then decide that you want to load interface changes later. All tags that you subsequently declare will be saved to the memory reserve. The subsequent download does not influence any tags that are already loaded or therefore have a negative impact on the ongoing operation.

If you decide to review your program at a later time while the plant is not in operation, you are also provided an option of reworking the memory layout of individual or several blocks in a single pass. With this action, you move all tags from the reserve area to the regular area. The memory reserve is now cleared and made available for further interface extensions.

Requirements

This "Download without reinitialization" function is available if the following requirements are met:

- The project is in the "TIA Portal V12" format or a higher version.
- You are working with a CPU that supports "Download without reinitialization".
- The blocks were created in LAD, FBD, STL, or SCL.
- The blocks were created by the user, i.e. they are not included with the blocks delivered in your package.
- These blocks are assigned the optimized access attribute.

Basic steps

To extend the interface of a function block and then load the block without re-initialization, follows these steps:

1. All blocks have a default memory reserve of 100 bytes. You can adapt this memory reserve to suit your requirements.
2. Activate the memory reserve.
3. Extend the block interface.

4. Compile the block.
5. Download the block to the CPU as usual.

For more information on the various steps, refer to chapter "Loading blocks (S7-1200/1500) ".

Note

The full scope of the "Download without reinitialization" function is only available on CPUs of the S7-1500 and S7-1200 V4 series.

However, all CPU families support the option of extending the interface of function blocks and downloading newly declared tags without repercussion:

- You may add new tags in the "Temp" section and download these without influencing the process.
- You may create new tags of a structured data type in the "InOut" section and download these without influencing the process.

Programming data blocks 10

10.1 Basic principles for programming of data blocks

A data block (DB) is used to save the values that are written during execution of the program.

In contrast to the code block, the data block contains only tag declarations. It contains no networks or instructions. The tag declarations define the structure of the data block.

Types of data blocks

There are two types of data blocks:

- Global data blocks
  The global data block is not assigned to a code block. You can access the values of a global data block from any code block. A global data block contains only static tags.
  The structure of the global data block can be freely defined. In the declaration table for data blocks, you declare the data elements that are to contained in the global data block.
- Instance data blocks
  The instance data block is assigned directly to a function block (FB). The structure of an instance data block cannot be freely defined, but is instead determined by the interface declaration of the function block. The instance data block contains exactly those block parameters and tags that are declared there.
  However, you can define instance-specific values in the instance data block, for example, start values for the declared tags.

ARRAY data blocks (S7-1500)

ARRAY data blocks are global data blocks that consist of an ARRAY. This ARRAY can be based on any data type. For example, an ARRAY of a PLC data type (UDT) is possible. The DB contains no other elements besides the ARRAY. Because of their flat structure, ARRAY data blocks facilitate access to the ARRAY elements and their transfer to called blocks.

You use ARRAY data blocks to address ARRAYs indirectly: You can determine during runtime which ARRAY DB is read or written and how large it is.

The "Move operations" section of the "Instructions" task card offers options for addressing of ARRAY DBs.

Note

Retentivity of ARRAY data blocks

ARRAY data blocks and their components cannot be set retentively.

See also:

Example of the use of ARRAY data blocks (Page 44)

PLC data types as a template for global data blocks

PLC data types can be used as templates for the creation of global data blocks with identical data structures. You create the structure as PLC data type only once and then generate the required data blocks by assigning the PLC data type.

System data types as a template for global data blocks

System data types can also be used as templates for creating global data blocks with identical data structure. System data types already have a pre-defined structure. You insert the system data type in the program only once and then generate additional data blocks with an identical structure by assigning the system data type.

Access modes

There are two different modes of accessing data values in data blocks:

- Data blocks with optimized access (only S7-1200)
  Data blocks with optimized access have no fixed defined structure. In the declaration, the data elements are assigned only a symbolic name and no fixed address within the block. You access the data values in these block via symbolic names.
  The "Optimized block access" attribute is always enabled for ARRAY data blocks.
- Data blocks with standard access (all CPU families)
  Data blocks with standard access have a fixed structure. In the declaration, the data elements are assigned both a symbolic name and a fixed address within the block. You can access the data values in this block via the symbolic names or addresses.
  Data blocks with standard access are tailored to WORD limits. This means that they always occupy one WORD (16 bits) or a multiple thereof. If necessary, tags are automatically added to the data block during compilation so that the block fills the space to the next WORD.
  ARRAY data blocks with standard access are not possible.

Retentivity of data values

To prevent data loss in the event of power failure, you can store the data values in a retentive memory area.

See also

Creating data blocks (Page 6910)

Global data blocks (DB) (Page 42)

Instance data blocks (Page 53)

Blocks with optimized access (Page 55)

10.2 Structure of the declaration table for data blocks

Structure of the declaration table for data blocks

The figure below shows the structure of the declaration table for data blocks. The display will vary depending on type of block and type of access.

	Name	Data type	Start value	Retain
	Input			
	MyInput1	Bool	false	☑
	Output			
	MyOutput1	Byte	0	☐
	InOut			
	Static			

Display of instance-specific values

In instance data blocks, you can apply the already defined values from the interface of the assigned function block or define instance-specific start values. Values that are applied from the function block cannot be edited. You can replace the grayed-out values with instance-specific values. Values that were already changed instance specific are not grayed out.

Meaning of the columns

The following table shows the meaning of the individual columns. You can show or hide the columns as required. The number of columns displayed varies depending on the CPU series.

Column	Explanation
	Symbol you can click to move or copy the tag. You can, for example, drag-and-drop the tag into a program and use it there as operand.
Name	Name of the tags.
Data type	Data type of the tags.
Offset	Relative address of the tags. The column is only visible in data blocks with standard access. Note: Many instructions from the SIMATIC system libraries have the "optimized block access" property and therefore do not occupy any fixed memory addresses. No offset is displayed for these instructions, even if you use them as multi-instance in a block with standard access.
Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.

Column	Explanation
Start value	Value that the tag should assume at startup. The default values defined in a code block are used as start values during the creation of the data block. You can then replace these adopted values with instance-specific start values. Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.
Monitor value	Current data value in the CPU. This column only appears if an online connection is available and you click "Monitor".
Snapshot	Shows values that were loaded from the device.
Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off.
Visible in HMI engineering	Shows whether the tag is visible by default in the HMI selection list.
Accessible from HMI/ OPC UA	Indicates whether HMI/OPC UA can access this tag during runtime.
Writable from HMI/ OPC UA	Indicates whether the tag from HMI/OPC UA can be written during runtime.
Setpoint value	Setpoint values are the values that will probably have to be fine tuned during commissioning. After commissioning, the values of these tags can be transferred to the offline program as start values and stored there.
Supervision	Indicates whether Supervision has been created for process diagnostics for the tag.
Comment	Comment to document the tags.

See also

Editing the properties of tags in data blocks (Page 7040)

Using setpoints during commissioning (Page 7063)

Creating data blocks (Page 6910)

Creating a data structure for global data blocks (Page 7029)

Basic information on start values (Page 7035)

Defining start values (Page 7035)

Setting retentivity (Page 7037)

10.3 Creating data blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a data block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Select the type of the data block. You have the following options available to you:
   - To create a global data block, select the list entry "Global DB".
   - To create an ARRAY data block, select the "ARRAY DB" entry in the list.
   - To create an instance data block, select the function block to which you want to assign the instance data block from the list. The list contains only the function blocks that were previously created for the CPU.
   - To create a data block based on a PLC data type, select the PLC data type from the list. The list contains only the PLC data types that were previously created for the CPU.
   - To create a data block based on a system data type, select the system data type from the list. The list contains only those system data types that have already been inserted to program blocks in the CPU.
4. Enter a name for the data block.
5. Enter the properties of the new data block.
6. If you have selected an ARRAY DB as the data block type, enter the ARRAY data type and the high limit for the ARRAY.
   You can change the high limit for the ARRAY at any time in the property window of the created block. The ARRAY data type cannot be changed subsequently.
7. If you have selected a block that contains monitoring as the "Type", assign a ProDiag function block to the monitoring functions.
8. To enter additional properties of the new data block, click "Additional information".
   An area with further input fields is displayed.
9. Enter all the properties you require.
10. Activate the "Add new and open" check box if the block does not open as soon as it is created.
11. Confirm your entry with "OK".

Result

The new data block is created. You can find the data block in the project tree in the "Program blocks" folder.

Note

Retentivity of ARRAY data blocks

ARRAY data blocks and their components cannot be set retentively.

See also

Global ARRAY data blocks (DB) (Page 43)

Basic principles for programming of data blocks (Page 7019)

Global data blocks (DB) (Page 42)

Instance data blocks (Page 53)

System data types (Page 349)

Using blocks from libraries (Page 6911)

10.4 Updating data blocks

Introduction

Changes in the interface of a function block or a PLC data type can lead to the corresponding data blocks becoming inconsistent. These inconsistencies are marked in red in the declaration table and at the call point of the block. To remedy these inconsistencies, the data blocks must be updated.

You have three options to update block calls:

- Explicit updating in the declaration table for data blocks.
  The data block is updated. Changes from the interface of the assigned function block and changes to the used PLC data types are applied.
- Explicit updating in the program editor.
  The block calls in the open block will be updated. The associated instance data block is also adjusted in the process.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types and the corresponding instance data blocks are updated.

Explicit Updating in the Declaration Table for Data Blocks

To explicitly update an individual data block, follow these steps:

1. Open the data block.
2. Select "Update interface" in the shortcut menu.

Explicit Updating in the Program Editor

To update all block calls or a specific call within a block, follow these steps:

1. Open the block in the program editor.
2. Right-click on the instruction with the block call.
3. Select the "Update" command in the shortcut menu.
4. The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
5. If you want to update the block call, click "OK". To cancel the update, click "Cancel".

Implicit Updating during Compilation

To implicitly update all block calls and uses of PLC data types as well as the instance data blocks during the compiling, follow these steps:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Changing the properties of tags in instance data blocks (Page 7042)

10.5 Extending data blocks (S7-1200, S7-1500)

Description

In order to enable the editing of PLC programs that have already been commissioned and that are running without error on a system, CPUs of the S7-1500 series and most CPUs of the S7-1200 V4 and higher series support the option of extending global data blocks during runtime.

You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Principle of operation

Each data block is always assigned a default memory reserve. The memory reserve is not used initially. Activate the memory reserve if you decide on loading interface changes after having compiled and downloaded the block. All tags that you subsequently declare will be saved to the memory reserve. A subsequent download has no impact on the values of tags that have already been loaded.

If you decide to review your program at a later time while the plant is not in operation, you are also provided an option of reworking the memory layout of individual or several blocks in a single pass. With this action, you move all tags from the reserve area to the regular area. The memory reserve is now cleared and made available for further interface extensions.

Requirements

This "Download without reinitialization" function is available if the following requirements are met:

- The project is in the "TIA Portal V12" format or a higher version.
- You are working with a CPU that supports "Download without reinitialization".
- The blocks were created in LAD, FBD, STL, or SCL.
- The blocks were created by the user, i.e. they are not included with the blocks delivered in your package.
- These blocks are assigned the optimized access attribute.

Basic steps

Perform the following steps if you want to extend the data block and then load the block without re-initialization.

1. All blocks have a default memory reserve of 100 bytes. You can adapt this memory reserve to suit your requirements.
2. Activate the memory reserve.
3. Extend the block interface.

4. Compile the block.
5. Download the block to the CPU as usual.

Reference

For more information on the various steps, refer to chapter "Loading blocks (S7-1200/1500) ".

10.6 Creating a data structure for global data blocks

10.6.1 Declaring tags of elementary data type

Requirement

A global data block is open.

Note

You cannot change the structure of instance data blocks and of data blocks based on a PLC data type directly, since the structures of these blocks are defined by the respective function block or the PLC data type.

The type of the data block is entered in the block properties.

Procedure

To declare a tag of the elementary data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. In the "Data type" column, click the button for the data type selection.
   A list of the permissible data types is opened.
3. Select the desired data type.
4. Optional: Change the properties of the tags that are displayed in the other columns.
5. Repeat steps 1 to 4 for all tags that are to be declared.

See also

Displaying and editing block properties (Page 6928)

Declaring tags of the ARRAY data type (Page 7030)

Declaring tags of STRUCT data type (Page 7031)

Editing the declaration table for data blocks (Page 7047)

Defining start values (Page 7035)

Setting retentivity (Page 7037)

Online and diagnostic functions for data blocks (Page 7051)

10.6.2 Declaring tags of the ARRAY data type

Requirement

A global data block is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the "Array" data type in the "Data type" column. You will be supported by Autocomplete in this step.
   The "Array" dialog opens.
3. In the "Data type" text box, specify the data type of the array elements.
4. In the "ARRAY limits" input field, specify the high and low limit for each dimension.
   Example of a one-dimensional ARRAY:
   `ARRAY [0..3] of Bool`
   Example of a three-dimensional ARRAY:
   `ARRAY[0..3, 0..15, 0..33] of Bool`
   Example of a one-dimensional ARRAY with local constants as ARRAY limits
   `ARRAY[#My_local_const1..#My_local_const2] of Bool`
   Example of a one-dimensional ARRAY with global constants as ARRAY limits
   `ARRAY["My_global_const1".."My_global_const1"] of Bool`
5. Confirm your entry.
6. Optional: Change the properties of the tags that are displayed in the other columns.

The tag is created but remains collapsed. To expand the ARRAY, click the triangle in front of the tag. Note that you cannot expand ARRAYs with more than 10,000 elements for reasons of clarity.

Entering start values of ARRAY elements

To set default start values for the individual elements of an ARRAY, follow these steps:

1. Click the triangle in front of the ARRAY data type tags.
   The ARRAY is expanded and the individual ARRAY elements are shown in separate lines.
2. Enter the required values in the "Start value" column.

Adding comments for ARRAYs

To add a comment for the entire ARRAY, enter the comment in the top line of the ARRAY declaration. The comment is used as a preset comment for all lower-level ARRAY elements.

To provide specific comments for the individual elements of an ARRAY, follow these steps:

1. Click the triangle in front of the ARRAY data type tags.
   The ARRAY is expanded and the individual ARRAY elements are shown in separate lines.
2. Enter the desired values in the "Comment" column.

Displaying ARRAYs in expanded mode

In "Expanded mode" all ARRAYs are displayed expanded by default.

To enable the expanded mode, follow these steps:

1. Click the "Expanded mode" button in the toolbar.

Note

Restriction in "Expanded mode" display

Depending on the amount of RAM available on your computer, the following maximum limits apply to expanded mode:

- RAM < 8 GB: A maximum of 10,000 rows are displayed.
- RAM >= 8 GB: A maximum of 40,000 rows are displayed.

See also

Defining start values (Page 7035)

Setting retentivity (Page 7037)

Editing the declaration table for data blocks (Page 7047)

ARRAY (Page 283)

Using autocompletion (Page 6859)

10.6.3 Declaring tags of STRUCT data type

Requirement

A global data block is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.
   An empty, indented row is inserted after the new tag.

3. Insert the first structural element in the first empty row.
   An additional empty row is inserted after the element.
4. Select a data type for the structure element.
5. Optional: Change the properties of the structural element that is displayed in the other columns of the block interface.
6. Repeat the step 4 to 7 for all additional structure elements.
   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
7. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

Enter start values of structure elements

To set default start values for the individual elements of a structure, follow these steps:

1. Click the triangle in front of the STRUCT data type tags.
   The structure opens and the individual structure elements are shown in separate rows.
   Note that, for reasons of clarity, you cannot expand very large structures.
2. Enter the required values in the "Start value" column.

Note

S7-1500: A maximum of 252 structures in one data block

A maximum of 252 structures is permitted in one data block. If you need additional structures, you must restructure your program. You can, for example, create the structures in several global data blocks.

See also

Basic information on STRUCT (Page 278)

Defining start values (Page 7035)

Setting retentivity (Page 7037)

Editing the declaration table for data blocks (Page 7047)

Using autocompletion (Page 6859)

10.6.4 Declaring tags of the STRING data type

Requirement

A global data block is open.

Procedure

To declare a tag of STRING data type, proceed as follows:

1. Select the appropriate declaration section in the interface.
2. Enter a tag name in the "Name" column.
3. Enter "STRING" in the "Data type" column. You will be supported by autocompletion during input.
4. Optional: Specify the maximum length of the string using square brackets after the keyword STRING. If you do not specify a maximum length, the string has a default length of 254 characters.
   Example of a STRING with the maximum length of 4:
   `STRING[4]`
   Example of a string whose maximum length is defined by a local constant:
   `STRING[#My_local_const1]`
   Example of a STRING whose maximum length is defined by a global constant:
   `STRING["My_global_const1"]`

Result

The tag of STRING data type is created.

10.6.5 Declaring tags based on a PLC data type

Requirement

- A global data block is open.
- A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a PLC data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the PLC data type in the "Data type" column. You will be supported by Autocomplete during input.

3. The tag is created.
4. Optional: Change the properties of the tags that are displayed in the other columns of the table.
   If you have already defined default values or comments for the tags within a PLC data type during declaration of the PLC data type, these are shown in gray. You can change these values in the block interface.
   The changed values are displayed in black and apply only to the specific point of use.

Note

If you change or delete PLC data types that are being used in the data block, the data block becomes inconsistent. To eliminate this inconsistency, you must recompile the program.

See also

Overview of the block interface (Page 6965)

Editing the declaration table for data blocks (Page 7047)

Using autocompletion (Page 6859)

10.7 Defining start values

10.7.1 Basic information on start values

Definition of "Start value"

The start value of a tag is a value defined by you which the tag assumes after a CPU startup.

The retentive tags have a special status. Their values take the defined start value only after a "cold restart". After a "warm restart", they retain their values and are not reset to the start value.

Definition of "Default value"

The structure of the data blocks can be derived from higher-level elements.

- An instance data block is based, for example, on the interface of a higher-level code block.
- A global data block can be based on a predefined PLC data type.

In this case you can define a default value for each tag in the higher-level element. These default values are used as start values during the creation of the data block. You can then replace these values with instance-specific start values in the data block.

Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.

See also

Defining start values (Page 7036)

Structure of the declaration table for data blocks (Page 7021)

Declaring local tags and constants in the block interface (Page 6975)

Applying values from the online program as start values (Page 7066)

10.7.2 Defining start values

Defining start values

To define the start values for the tags of a data block, follow these steps:

1. Open the data block.
   The "Default value" column shows the default values that were defined for the tags in the interface of a higher-level code block or in a PLC data type.
2. Click the "Expanded mode" button to show all elements of structured data types.
3. Enter the desired start values in the "Start value" column. The value must match the data type of the tag and should not exceed the range of the data type.
   The start values are defined. The tag takes the defined value at startup, provided it was not declared as retentive.

Resetting a tag to the default value

To reset a tag for which you have defined a start value to the default value, follow these steps:

1. Select a modified value in the table.
2. Delete the value.
   The default value is entered. The default value is displayed.

Resetting all tags to the default value

To reset to the default value all tags for which you have defined an start value, follow these steps:

1. Select the "Reset start values" icon in the toolbar.
   The default values are transferred to the "Start value" column. Write-protected start values are not overwritten.

See also

Basic information on start values (Page 7035)

Applying values from the online program as start values (Page 7066)

Editing the declaration table for data blocks (Page 7047)

10.8 Setting retentivity

10.8.1 Retentivity of tags in data blocks

Retentive behavior

To prevent data loss in the event of power failure, you can mark the data as retentive. This data is stored in a retentive memory area. The options for setting the retentivity depend on the type of data block and the type of block access that is set.

See also

Setting retentivity in an instance data block (Page 7037)

Setting retentivity in a global data block (Page 7038)

10.8.2 Setting retentivity in an instance data block

Introduction

In an instance data block, the editability of the retentive behavior depends on the type of access of the higher-level function block:

- Function block with standard access
  You can define the instance data both as retentive or non-retentive. Individual retentivity settings are not possible for individual tags.
- Function block with optimized access
  In the instance data block, you can define the retentivity settings of the tags that are selected in the block interface with "Set in IDB". With these tags also, you cannot individually set the retentive behavior for each tag. The retentivity setting has an impact on all tags that are selected in the block interface with "Set in IDB".

Setting Retentivity for Standard Access

To centrally set the retentivity of all tags in the data block with standard access, follow these steps:

1. Open the instance data block.
2. Select the check box in the "Retain" column of a tag.
   All tags are defined as retentive.
3. To reset the retentivity setting for all tags, clear the check box in the "Retain" column of a tag.
   All tags will be defined as non-retentive.

Setting Retentivity for Optimized Access

To set the retentive behavior of the tags that are selected with "Set in IDB" in data blocks with optimized access, follow these steps:

1. Open the instance data block.
2. Select the check box in the "Retain" column of a tag.
   All tags selected with "Set in IDB" in the data block interface are defined as retentive.
3. To reset the retentivity setting for the tags, clear the check box in the "Retain" column of a tag.
   All tags selected with "Set in IDB" in the data block interface will be defined as non-retentive.

See also

Basics of block access (Page 55)

Retentivity of tags in data blocks (Page 7037)

Editing the declaration table for data blocks (Page 7047)

10.8.3 Setting retentivity in a global data block

Introduction

In a global data block, the editability of the retentive behavior depends on the type of access:

- Global data block with standard access
  You can define the data both as retentive or non-retentive. Individual retentivity settings are not possible for individual tags.
- Global data block with optimized access
  You can individually define the retentivity settings of the tags. For tags with structured data types, retentivity settings are transferred for all tag elements.

Note

Retentivity of ARRAY data blocks

ARRAY data blocks and their components cannot be set retentively.

Setting Retentivity for Standard Access

To centrally set the retentivity of all tags in the data block with standard access, follow these steps:

1. Open the global data block.
2. Select the check box in the "Retain" column of a tag.
   All tags are defined as retentive.
3. To reset the retentivity setting for all tags, clear the check box in the "Retain" column of a tag.
   All tags are defined as non-retentive.

Setting Retentivity for Optimized Access

To individually set the retentivity of all tags in data blocks with optimized access, follow these steps:

1. Open the global data block.
2. In the "Retain" column, select the check box for the tags for which you want to set a retentive behavior.
   The selected tag is defined as retentive.
3. To reset the retentivity setting for the tags, clear the check box in the "Retain" column of a tag.
   All selected tags are defined as non-retentive.

See also

Basics of block access (Page 55)

Retentivity of tags in data blocks (Page 7037)

Editing the declaration table for data blocks (Page 7047)

10.9 Editing the properties of tags in data blocks

10.9.1 Properties of the tags in data blocks

Properties

The properties of a tag are displayed in the Inspector window when you select the tag in the declaration table or at a point of use in the program editor.

Note

Properties of ARRAY elements

When you use ARRAY accesses with variable index in the form `#Program_Alarm_1[#myIndex]` in your program, the specific ARRAY element to be used during runtime is not yet known at the time of programming. For this reason, when you select this type of ARRAY access in the program editor, the properties of the first ARRAY element are always displayed.

The following table provides an overview of the properties of tags in data blocks:

Group	Property	Description
General	Name	Name of the tags.
	Data type	Data type of the tags.
	Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
	Start value	Value that the tag should assume at CPU startup. The default values defined in a code block are used as start values during the creation of the data block. You can then replace these adopted values with instance-specific start values. Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as default value for BOOL.
	Comment	Comment on the tag.

Group	Property	Description
Attributes	Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This attribute is only available in the interface of the function block with optimized access.
	Visible in HMI engineering	Shows whether the tag is visible by default in the HMI selection list.
	Writable from HMI/OPC UA	Indicates whether or not the tag can be written from HMI/OPC UA during runtime.
	Accessible from HMI/OPC UA	Indicates whether HMI/OPC UA can access this tag during runtime. Please note, however, that you cannot implement general access protection for the tag with the "Accessible from HMI/OPC UA" attribute. Read or write access from other applications is possible even if the attribute is not enabled.
	Visibility in block calls in LAD/FBD	Show: The parameter is always visible at block calls in LAD or FBD.
		Hide: The parameter is always hidden at block calls in LAD or FBD. The instruction box then has a small arrow on the lower edge. You can have the parameters displayed and interconnect them by clicking this arrow.
		Hide if no parameter is assigned: The parameter is always hidden at block calls in LAD or FBD as long as it has not yet been interconnected. If you assign an actual parameter, the parameter is displayed at the call box.
	Predefined actual parameter	Defines a parameter that is to be used as actual parameter during the block call.
	Show, if parameter assigned on block call is not identical to the predefined actual parameter	The parameter is hidden at block calls in LAD or FBD as long as the predefined actual parameter has not yet been assigned. However, when a different parameter is assigned, the parameter becomes visible. This option is only active if you have selected the "Hide" parameter and have defined a predefined actual parameter.

Group	Property	Description
User-defined attributes	CFC_Visible	Visible Indicates whether a parameter is visible in CFC.
	CFC_Configurable	Configurable Indicates whether a parameter is configurable in CFC.
	CFC_ForTest	For test Indicates whether a parameter is registered for the CFC test mode.
	CFC_Interconnectable	Interconnectable Indicates whether a parameter is interconnectable in CFC.
	CFC_EnableTagReadback	Enable tag readback Indicates whether a parameter is relevant for the "Read back chart" function in CFC.
	CFC_EnumerationTexts	Enumeration texts Assigns a parameter to an enumeration in CFC.
	CFC_EngineeringUnit	Engineering unit Assigns a parameter to a unit in CFC.
	CFC_LowLimit	Low limit Defines the low limit for the parameter in CFC.
	CFC_HighLimit	High limit. Defines the high limit for the parameter in CFC.

See also

Changing the properties of tags in instance data blocks (Page 7042)

Changing the properties of tags in global data blocks (Page 7044)

Setting retentivity (Page 7037)

Defining start values (Page 7035)

10.9.2 Changing the properties of tags in instance data blocks

Instance-specific tag properties

Two options are available for defining the tag properties:

- The tag properties are applied from the interface of the assigned function block.
  Properties that are applied from the function block are displayed grayed out out in the columns of the declaration table. The "Name" and "Data type" properties are always applied.
- You define instance-specific properties.
  You can change some properties instance specific. For example, the comment is a changeable value. Properties that were changed instance specific are not grayed out in the columns of the declaration table. The instance-specific changes are retained, even if the interface of the higher-level function block is changed and the instance data blocks are subsequently updated.

Editing properties of an element in the declaration table

To edit the properties of an element, follow these steps:

1. Open the instance data block.
2. Select the required element in the table.
3. Change the entries in the columns.

Editing properties of several elements in the declaration table

You can also simultaneously set or reset some properties for multiple selected elements.

To change one of these properties for several elements, follow these steps:

1. Open the data block.
2. Hold down the CTRL key.
3. In the required column, select each of the table cells whose value you want to change.
4. Select the "Set <property>" or "Reset <property>" command in the shortcut menu.

Editing properties in the properties window

To edit the properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.
   The properties window opens. It shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

Reset individual properties to the default value.

To reset individual tag properties to the value that was defined as default in the function block, follow these steps:

1. Select an instance-specific, modified value in the table.
2. Delete the value.
   The instance-specific value will be deleted and the default value from the interface of the function block entered. The default value is displayed grayed out.

See also

Updating data blocks (Page 7025)

Properties of the tags in data blocks (Page 7040)

Editing the declaration table for data blocks (Page 7047)

10.9.3 Changing the properties of tags in global data blocks

Introduction

Two options are available for defining the tag properties:

- The tag properties are applied from the PLC data type.
  Properties that are applied from the PLC data type are shown grayed out in the columns of the declaration table. The "Name" and "Data type" properties are always applied.
- You define specific properties.
  You can change some properties in the global data block. Changeable values are, for example, "Comment" or "Visible in HMI". Properties that were changed are not grayed out in the columns of the declaration table. The changes are retained, even if the PLC data type changes and the global data block is subsequently updated.

Editing properties of an element in the declaration table

To edit the properties of an element, follow these steps:

1. Open the global data block.
2. Select the required element in the table.
3. Change the entries in the columns.

Editing properties of several elements in the declaration table

You can also simultaneously set or reset some properties for multiple selected elements.

To change one of these properties for several elements, follow these steps:

1. Open the data block.
2. Hold down the CTRL key.
3. In the required column, select each of the table cells whose value you want to change.
4. Select the "Set <property>" or "Reset <property>" command in the shortcut menu.

Editing properties in the properties window

To edit the properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.
   The properties window opens. It shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

Reset individual properties to the default value.

To reset individual tag properties to the value that was defined as default in the PLC data type, follow these steps:

1. Select a modified value in the table.
2. Delete the value.
   The default value from the PLC data type is entered. The default value is displayed grayed out.

See also

Properties of the tags in data blocks (Page 7040)

Editing the declaration table for data blocks (Page 7047)

10.9.4 Editing multilingual comments in the data block (S7-1500)

You can edit the comments from a data block in all project languages that you have activated in the project language settings. You can find additional information on activating project languages under AUTOHOTSPOT.

Note that the texts must not exceed a length of 32767 Unicode characters even after translation.

Requirement

- You have activated multiple project languages.
- The data block is open and it contains at least one comment.
- The block does not have know-how protection.

Procedure

To edit a comment in all project languages, follow these steps:

1. Click on the comment whose translation you want to edit.
   If you want to edit the translations of several comments at the same time, select all the desired comments.
   In instance data blocks, you can select only comments you have changed on an instance-specific basis. You can identify these comments by the fact that they are not grayed out. You can only select and edit grayed-out comments in the associated function block.
2. Open the "Properties" tab in the Inspector window.
3. Open the "Texts" tab.
   The "Texts" tab displays the selected comments in the active project languages and, if available, the translations of the comments.
4. You can edit the translations directly in the table in the "Texts" tab.
5. To edit the translations with an external editor, you can export the displayed texts to OOXML format using the "Export/Import project texts" buttons.

Note

Editing all project texts in the global "Project texts" table

You can also edit the translations for the individual project languages in the global "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts". It contains all translatable texts of the entire project.

You can find additional information on translation of texts under AUTOHOTSPOT.

10.10 Editing the declaration table for data blocks

10.10.1 Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

10.10.2 Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

10.10.3 Deleting tags

Requirements

A global data block is open.

Procedure

To delete a tag, follow these steps:

1. Select the row with the tag to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

Note

You cannot directly change the structure of instance data blocks and of global data blocks based on a PLC data type, since the structures of these blocks are defined in the higher-level object.

The type of the data block is entered in the block properties.

See also: Displaying and editing block properties (Page 6928)

10.10.4 Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

10.10.5 Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.
5. To hide or show several columns, click "More" and activate or deactivate the check box of the corresponding columns in the "Show/Hide" dialog.

10.10.6 Editing tags with external editors

To edit individual tags in external table editors, such as Excel, you can export or import these tags using copy and paste.

Note

Restriction when displaying and copying ARRAY elements

Depending on the amount of work memory available on your computer, the following maximum limits apply when displaying and copying ARRAYs from the data block editor in an Excel table:

- RAM < 8 GB: A maximum of 10,000 elements are displayed and copied.
- RAM >= 8 GB: A maximum of 40,000 elements are displayed and copied.

Requirement

The data block and an external editor are opened.

Procedure

To export and re-import individual tags by drag-and-drop operation, follow these steps:

1. Click the "Expanded mode" button to show all elements of structured data types.
2. Select one or more tags.
3. Select "Copy" in the shortcut menu.
4. Switch to the external editor and paste the copied tags.
5. Edit the tags as required.
6. Copy the tags in the external editor.

7. Switch back to the declaration table.
8. Select "Paste" in the shortcut menu.

10.11 Online and diagnostic functions for data blocks

10.11.1 Editing data blocks in online mode

Introduction

When you load a data block to the CPU, there are two versions of the block:

- An offline version in the project on your programming device
- An online version on the CPU
  Data blocks can either be located only in the load memory, only in the work memory or in both memory areas.

Start values and actual values of tags

The values of the tags in the offline and online version of a data block are different because the tags in the online block can be changed by the user program during runtime.

When you declare a data block, you assign a start value to the tags. The start value is the value of the tag when the data block is called for the first time. This means the tag is "initialized" with the start value.

The start values of the tag are declared in the offline version of the block. When the data block is loaded to the CPU, it is transferred to the load memory with the start values. From there, the data block is loaded to the work memory of the CPU. When the block is loaded for the first time, the tags in the work memory are initialized with the declared start values. They can now be changed with the user program. The values of the tags in the work memory are referred to as "actual values".

Loading changed data blocks

When you make changes to a data block after initial loading, you are editing the offline version of the block. Depending on the existing conditions, subsequent loading of the data block has different effects on the actual values.

- Loading changes in "RUN":
  - If you have not made any changes to the structure of the data block, the actual values of all tags are retained. The program is processed further "bumplessly" Any changes to the start values in the offline data block are written to the load memory. They are not transferred to the CPU until the next transition from "STOP" to "RUN".
  - If you have changed the structure of the data block, all tags are reinitialized with their start values. Initializing the tags may have unwanted effects on the process because the process and the program may not be synchronous any longer.
    You can prevent this by using the memory reserve of the block. The memory reserve enables you to load data blocks in "RUN" without reinitialization. This means you can reload "bumplessly" even if you have changed the structure of the data block and have added new tags, for example.
    See also: Basic information on loading block extensions without reinitialization (Page 7560)
- Loading changes in "STOP":
  - The start values of non-retentive tags are placed in the load memory of the CPU. The program runs with the new start values at the next transition from "STOP" to "RUN".
  - Retentive tags are not reinitialized. Your actual values are retained after loading in "STOP".
    To also reinitialize the retentive tags when loading in "STOP", select the blocks you want to load in the project tree and select the command "Compile > Software (rebuild all blocks)" in the shortcut menu. In the "Online" menu, select the command "Download and reset PLC program". The online blocks are deleted and overwritten by the new blocks. This reinitializes all tags, including the retentive tags.

Editing actual values with online and diagnostic functions

To manipulate actual values directly in online mode without having to reload data blocks, you can use the online and diagnostic functions of the TIA Portal.

See also: Overview of the online and diagnostic functions in data blocks (Page 7053)

Loading data blocks from a device

You can also load data blocks from a device again. This makes sense, for example, if the original offline project is not available during servicing. Please note that the two CPU families handle the values of tags differently when loading blocks from a device:

- S7-1200/1500
  CPUs of the S7-1200 and S7-1500 series store the start values with which a data block is loaded to the CPU in the load memory. If you load a data block from an S7-1200/1500 CPU, these start values are applied from the load memory to the offline data block again. Even if you have made value changes in the load memory during runtime by means of the "WRIT_DBL" instruction, the values originally loaded to the device are applied to the offline data block.
- S7-300/400
  CPUs of the S7-300/400 series also store the start values with which a data block is loaded to the CPU in the load memory. However, they cannot read the values back from the load memory. If you load a data block from a device, the current monitored values are loaded from the CPU and entered in the "Start values" column in the offline data block.

See also

Downloading blocks for S7-1200/1500 (Page 7544)

10.11.2 Overview of the online and diagnostic functions in data blocks

Overview of functions

The data block editor offers different options for editing data blocks in online mode. These functions directly access the actual values of the tags in the online program. Actual values are the values which the tags have at the current time during program execution in the CPU work memory.

The following table provides an overview of the online and diagnostic functions. Detailed descriptions of the individual functions can be found in the following chapters.

Button	Function	Description	S7-300/400	S7-1200/1500
	Monitor tags online (Page 7054)	Displays the actual values which the tags currently have in the CPU.	X	X
-	Modify individual actual values (Page 7055)	Modifies individual tags immediately and once only to specific values in the declaration table. The CPU then uses these values as actual values in the online program.	X	X
	Create a snapshot of the actual values (Page 7058)	Saves the actual values present at the current time as snapshot in the offline project. The snapshot always captures the actual values of all tags of the data block.	X	X

Button	Function	Description	S7-300/400	S7-1200/1500
	Reinitialize actual values with a snapshot (Page 7059)	Loads the snapshot to the CPU as actual values. The CPU then uses these values as actual values in the online program.	-	S7-1200 V4.1 and higher S7-1500 V1.7 and higher
	Apply snapshot values as start values (Page 7061)	Copies the snapshot to the start values in the offline program. The program runs with the new start values at the next transition from STOP to RUN. You can copy all start values or only the start values of the tags identified as "setpoint".	X	X
	Reinitialize actual values with start values (Page 7056)	Reinitializes the actual values of tags. The start values are written directly to the CPU work memory. You can copy all actual values or only the actual values of the tags identified as "setpoint".	X	X

See also

Initializing setpoint values in the online program (Page 7065)

Applying values from the online program as start values (Page 7066)

Editing data blocks in online mode (Page 7051)

Using setpoints during commissioning (Page 7063)

Defining start values (Page 7035)

10.11.3 Monitoring tags

You can monitor the actual values of the tags in the CPU directly in the declaration table.

Requirement

- An online connection is available.
- The data block has been loaded to the CPU.
- The program execution is active (CPU in "RUN").
- The data block is open.

Procedure

To monitor the tags, follow these steps:

1. Start monitoring by clicking the "Monitor all" button.
   The additional "Monitor value" column is displayed in the table. This shows the current data values.
   See also: Structure of the declaration table for data blocks (Page 7021)
2. End monitoring by clicking the "Monitor all" button again.

See also

Overview of the online and diagnostic functions in data blocks (Page 7053)

10.11.4 Modifying tags

You can modify an individual tag in the data block to a specific value. The CPU then uses this value as the actual value in the online program.

DANGER

Danger when changing tag values

Changing the tag values while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors.

Make sure that no dangerous situations can arise before you modify the tags.

Requirement

- An online connection to the CPU is available.
- The data block whose tags you wish to modify is identical offline and online.
- The data block is open.

Procedure

To modify an individual tag in the data block, follow these steps:

1. Start monitoring by clicking the "Monitor all" button.
   The additional "Monitor value" column is displayed in the table. This shows the current data values.
2. Double-click the tag to be modified.
   - In the case of Boolean tags, the "Toggle value" dialog opens. Confirm this dialog by clicking "OK".
     If you select the option "Do not show this message again" in this dialog, you can switch the value in the future by double-clicking.
   - In the case of non-Boolean tags, the "Modify value" dialog opens. Enter the desired value in the "Modify value" input field and confirm with "OK".
     If there is a snapshot, the value is already entered as default.

Result

The tag will have the specified value once when you execute the modify job. The job is executed immediately and is not tied to the next cycle control point.

See also

Introduction to modifying tags (Page 7759)

Overview of the online and diagnostic functions in data blocks (Page 7053)

10.11.5 Loading start values as actual values

You can load the start values from the offline program to the CPU as actual values. The start values are loaded directly to the CPU work memory. The tags in the online block are reinitialized, and the CPU uses the new values as actual values in the online program. No distinction is made between retentive and non-retentive values. You can copy all actual values or only the actual values of the tags identified as "setpoint".

You can quickly bring the controlled process into a defined state this way.

You cannot reinitialize the following types of data blocks:

- Data blocks that are part of a reference project
- Fail-safe data blocks
- Data blocks generated by the CPU
- Data blocks with the attribute "Only store in load memory"
- Data blocks that only exist online

You can execute this function in "RUN" mode as well as "STOP" mode.

DANGER

Danger when changing tag values

Changing the tag values while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors.

If the volume of data to be transmitted is too large, the values may be transmitted in several cycles. If the program accesses tags before all values are completely transmitted, there is a risk that inconsistent value combinations may be created and processed.

Copying the values of elementary data types may also take several cycles. These values are potentially invalid until they have been completely transmitted. Dangerous states may occur if the program accesses these values before they have been completely transmitted.

To prevent dangerous states from occurring, please note the following:

- Make sure that the plant is in a safe state before you overwrite the actual values.
- Make sure that the program does not read or write the affected data during transmission.
- You may want to use the "Modify tag" function in the watch table or in the DB editor as an alternative.

Requirement

- You are using an S7-1200 V4.1 or higher or an S7-1500 V1.7 or higher.
- An online connection to the CPU is available.
- The data blocks whose actual values you want to reinitialize are identical offline and online.

Reinitializing actual values of an open block

To reinitialize all actual values or only the setpoints of a block with start values, follow these steps:

1. Open the data block.
2. Click the "Load start values as actual values > All values" button.

Or:

1. Open the data block.
2. Click the "Load start values as actual values > Only setpoints" button.

Reinitializing actual values of multiple blocks

To reinitialize all actual values of multiple blocks in one step, proceed as follows:

1. Select multiple blocks or the entire block folder in the project tree.
2. Select the "Load start values as actual values" command in the shortcut menu.

Result

The actual values in the online program are reinitialized with the start values.

The maximum number of tags that can be initialized is dependent on the CPU. If there are too many tags, an alarm informs you about this. In this case, you can insert the tags in a watch table and initialize them using the "Modify" function in the watch table.

See also

Overview of the online and diagnostic functions in data blocks (Page 7053)

10.11.6 Creating a snapshot of the actual values

You can save the actual values of the tags of one or more data blocks as snapshot. Actual values are the values which the tags have at the current time during program execution in the CPU work memory.

A snapshot cannot be created for data blocks that are stored only in the load memory and are not integrated in the program.

You have the following options for creating a snapshot:

- Create a snapshot of an open data block
- Create a snapshot of several selected data blocks

A snapshot is also created automatically when you download a block or a program from the device.

⚠ CAUTION
Creating the snapshot
The values in the snapshot may originate in several cycles.

Requirement

- An online connection to the CPU is available.
- The data blocks for which you want to create a snapshot are identical offline and online.

Procedure

To create a snapshot of an open data block, follow these steps:

1. Open the data block.
2. Click "Snapshot".

To create a snapshot of several selected data blocks, follow these steps:

1. Select the blocks in the project tree.
   You can select the blocks individually or select devices, groups or folders in the project tree which include the data blocks.
2. In the shortcut menu, select "Snapshot of the current values" or select the menu command "Online > Snapshot of the current values".

Result

- The latest monitored values will be applied in the "Snapshot" column.
- An alarm is shown in the Inspector window after the operation is complete.
- The time stamp of the snapshot is displayed above the declaration table.

Note

Changes to the data block structure

If you change the structure of the data block after creation of a snapshot, individual values in the snapshot may become invalid. For example, if you change the data type of a tag, the value of this tag becomes invalid. No value is displayed for this tag in the "Snapshot" column.

See also

Overview of the online and diagnostic functions in data blocks (Page 7053)

10.11.7 Loading snapshots as actual values

You can load the snapshots from the offline program to the CPU as actual values. The values from the snapshot are loaded directly to the CPU work memory. The tags in the online block are reinitialized, and the CPU uses the new values as actual values in the online program.

You can quickly bring the controlled process into a defined state this way.

You cannot reinitialize the following types of data blocks with snapshots:

- Data blocks that are part of a reference project
- Fail-safe data blocks
- Data blocks generated by the CPU
- Data blocks with the attribute "Only store in load memory"
- Data blocks that only exist online

DANGER

Danger when changing tag values

Changing the tag values while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors.

- Make sure that the plant is in a safe state before you overwrite the actual values.
- Make sure that the program does not read or write the affected data during transmission.
- You may want to use the "Modify tag" function in the watch table or in the DB editor as an alternative.

Dependencies on the CPU mode

You can execute this function in "RUN" mode as well as "STOP" mode. The table below shows the reactions of the CPU in the different modes:

Action	System reaction	Consequences for the online program
Overwrite actual values in "RUN" mode	The values of all DB tags are overwritten in the current program. No distinction is made between retentive and non-retentive values.	Changing the actual values can result in inconsistencies between the program and the actual process. If the volume of data to be transmitted is too large, the values may be transmitted in several cycles. If the program accesses tags before all values are completely transmitted, there is a risk that inconsistent value combinations may be created and processed. Copying the values of elementary data types may also take several cycles. These values are potentially invalid until they have been completely transmitted. Dangerous states may occur if the program accesses these values before they have been completely transmitted.
Overwrite actual values in "STOP" mode	Only the actual values of the retentive tags are overwritten by the snapshot. Non-retentive tags are initialized with their start values during the transition from "STOP" to "RUN". The values from the snapshot are not taken into consideration.	Because only the retentive data from the snapshot is transmitted, there is a risk that inconsistent value combinations may be created and processed.

Requirement

- You are using an S7-1200 V4.1 or higher or an S7-1500 V1.7 or higher.
- An online connection to the CPU is available.
- The data blocks whose actual values you want to reinitialize are identical offline and online.
- A snapshot of the data blocks has been created.

Reinitializing actual values of an open block

To reinitialize all actual values of a block with snapshots, follow these steps:

1. Open the data block.
2. Click "Load snapshots as actual values".

Reinitializing actual values of multiple blocks

To reinitialize the actual values of multiple blocks in one step, follow these steps:

1. Select multiple blocks or the entire block folder in the project tree.
2. Select the "Load snapshots as actual values" command in the shortcut menu.

Result

The actual values in the online program are reinitialized with the values from the snapshot. Tags for which a valid snapshot does not exist are reinitialized with their start values.

See also

Overview of the online and diagnostic functions in data blocks (Page 7053)

10.11.8 Copying the snapshot to the start values

You can copy the snapshot to the start values in the offline program. The program runs with the new start values at the next transition from "STOP" to "RUN".

You can copy all start values, the start values of the retentive tags or only the start values of the tags identified as "setpoint".

Note that the values from the snapshot are always copied. There is no check to determine whether all values originate from the same cycle.

Write-protected start values are not overwritten.

You have the following basic options for applying the values:

- Applying the values of an open data block
  You can apply all values or only the values of the tags marked as "setpoint" as start values in an open data block.
- Applying the values of multiple blocks in the project tree
  You can either apply all values, all setpoints or all retentive values as start values in the project tree.

Requirement

A snapshot has been created for the data block.

Procedure

To apply the snapshots as start values in a data block, follow these steps:

1. Open a data block.
2. In the toolbar, click "Copy snapshots to start values > All values".

Or:

1. Open a data block.
2. In the toolbar, click "Copy snapshots to start values > Only setpoints".

To apply the monitored values of multiple data blocks in the project tree, follow these steps:

1. Select the blocks in the project tree.
2. Then select one of the following commands in the shortcut menu:
   - "Copy snapshots to start values > All values"
   - "Copy snapshots to start values > Only setpoints"
   - "Copy snapshots to start values > Only retain values"

Result

The values from the "Snapshot" column are applied to the "Start value" column. The new start values are stored in the offline program.

Note

Applying values of individual tags

You can also apply the values of individual tags from the "Snapshot" column to the "Start values" column. Use the "Copy" and "Paste" commands from the shortcut menu to copy the values and insert them in the "Start value" column. Note that only the values that are currently located in the visible area of the table are copied.

See also

Overview of the online and diagnostic functions in data blocks (Page 7053)

10.11.9 Using setpoints during commissioning

10.11.9.1 Basic information on adjusting setpoints during commissioning

Introduction

During commissioning of a plant, data values have to be frequently adjusted in order to optimally adapt the program to the general operating conditions on site. To this end, you can use the online and diagnostic functions of the declaration table for data blocks.

To use the functions, first define specific tags as "setpoints" in the program. Setpoints are the values that will probably have to be fine-tuned during commissioning.

The following table provides an overview of the functions for tuning values during commissioning. Detailed descriptions of the individual functions can be found in the following chapters.

In addition, there are also the general functions for monitoring and controlling the data block:

Overview of the online and diagnostic functions in data blocks (Page 7053)

Button	Function	Description
	Initialize setpoints in "RUN" mode (Page 7065)	This function enables you to change the values of individual tags online to quickly determine the optimum tag values.
	Apply values from the online program as start values to the offline program (Page 7066)	When you have determined the optimum tag values, you can apply these as start values in the offline program. This allows you to ensure that the program starts with the optimized values the next time it is loaded.

See also

Marking data as values that can be set (Page 7063)

10.11.9.2 Marking data as values that can be set

You can mark specific tags in the program as "Setpoints". Setpoints are the values that will probably have to be fine-tuned during commissioning.

Rules

You can mark tags as "setpoints" in the following block types:

- In function blocks (FB), but only in the "Static" section
- in global data blocks (DB)
- in PLC data types (UDT)
  In the case of PLC data types, however, the setting is only effective if the UDT is used in the "Static" section of a function block or data block.

It is not possible to define setpoints in the following block types:

- In data blocks based on a PLC data type and in instance data blocks. These inherit the setting from the higher-level FB or UDT.
- You cannot mark tags as "setpoints" in ARRAY data blocks.
- You also cannot mark tags as "setpoints" at the call point of a multi-instance. You have to make the setting in the interface of the function block that is called as multiple instance.
- You cannot change the "setpoint" marking in know-how-protected blocks. To do so, you must first remove the know-how protection.

Requirement

A function block, a global data block or a PLC data type (UDT) is open.

Procedure

To mark a tag as "setpoint", follow these steps:

1. Select a tag from the "Static" section.
2. Select the check box in the "Setpoint" column.
   - You cannot define the higher-level element of a structure or a PLC data type as "setpoint". You have to make the setting for the lower-level elements individually.
   - In the case of ARRAYs, you can only mark the higher-level element as "setpoint". The lower-level elements inherit the setting.
   - For ARRAYs of STRUCT, you can only mark the elements below the first structure as setpoints. The elements of other structures inherit the setting.

Result

The tags are marked as setpoints. During commissioning, these tags can be initialized online. You do not need to set the CPU to "STOP" mode; it can remain in "RUN". In addition, the current tag values can be transferred as start values to the offline program and saved there.

See also

Overview of the online and diagnostic functions in data blocks (Page 7053)

Online and diagnostic functions for data blocks (Page 7051)

Defining start values (Page 7035)

10.11.9.3 Initializing setpoint values in the online program

Basics on initializing setpoints

You can initialize all tags marked as "setpoints" with new values in the online program. At the same time, the start values will be loaded from the offline program to the online program. The CPU remains in "RUN" mode. All tags that are marked as setpoints are initialized once at the next cycle control point. This applies both to retentive and non-retentive tags. The program execution is then continued with the new tag values.

The maximum number of initializable tags is CPU-dependent: For S7-300/400 CPUs, you can initialize up to 35 setpoints. For S7-1200/1500 CPUs, the maximum amount is 200 setpoints. The maximum amount may be less, however, depending on the data types used.

If too many setpoints are selected, an alarm informs you of this. In this case, you can insert the tags in a watch table and initialize them using the "Modify" function in the watch table.

DANGER

Danger when changing tag values

Changing the tag values while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors.

Make sure that no dangerous situations can arise before you reinitialize the setpoints.

Requirement

- An online connection to the CPU exists.
- The structure of the data block is identical offline and online.
- One or more tags are marked as "setpoints".

Procedure

To initialize all setpoints of the data block, follow these steps:

1. Open a global data block or an instance data block.
2. Enter the required values in the "Start value" column. The start values must correspond to the indicated data type.
3. Click the "Load start values as actual values > Only setpoints" button.

Result

The setpoints in the online program are initialized with the start values from the offline program at the next cycle control point.

See also

Overview of the online and diagnostic functions in data blocks (Page 7053)

10.11.9.4 Applying values from the online program as start values

In order to apply tag values from the online program to the offline program as start values, first create a snapshot of the tag values from the online program. You can then apply them to the offline program. Note that the values from the snapshot are always copied. There is no check to determine whether all values originate from the same cycle.

Write-protected start values are not overwritten.

You have the following basic options for applying the values:

- Applying the setpoints of an open data block
- Applying the setpoints of multiple blocks in the project tree

Requirement

- An online connection to the CPU is available.
- As least one data block has been loaded to the CPU.

Procedure

To apply all setpoints as start values in a data block, follow these steps:

1. Open the data block.
2. Start monitoring by clicking the "Monitor all" button.
   The "Monitor value" column is displayed in the table. This shows the current data values.
3. In the toolbar, click "Snapshot".
   The actual values are applied in the "Snapshot" column.
4. In the toolbar, click "Copy snapshots to start values > Only setpoints".

The values from the "Snapshot" column are applied to the "Start values" column.

To apply the monitored values of multiple data blocks in the project tree, follow these steps:

1. Select the blocks in the project tree.
2. Select the "Snapshot" command in the shortcut menu.
   The actual values of all selected blocks are applied in the "Snapshot" column.
   An alarm is shown in the Inspector window after the operation is complete.
3. Next, select the command "Copy snapshot values as start values > Only setpoints" in the shortcut menu.

The values from the "Snapshot" column are applied to the "Start value" column.

Result

The new start values are stored in the offline program.

Note

Applying values of individual tags

You can also apply the values of individual tags that were not marked as setpoints beforehand from the "Snapshot" column to the "Start values" column. Use the "Copy" and "Paste" commands from the shortcut menu to copy the values and insert them in the "Start value" column. Note that only the values that are currently located in the visible area of the table are copied.

See also

Basic information on start values (Page 7035)

Defining start values (Page 7036)

Overview of the online and diagnostic functions in data blocks (Page 7053)

Online and diagnostic functions for data blocks (Page 7051)

Declaring PLC tags 11

11.1 Overview of PLC tag tables

Introduction

PLC tag tables contain the definitions of the PLC tags and symbolic constants that are valid throughout the CPU. A PLC tag table is created automatically for each CPU used in the project. You can create additional tag tables and use these to sort and group tags and constants.

In the project tree there is a "PLC tags" folder for each CPU of the project. The following tables are included:

- "All tags" table
- Standard tag table
- Optional: Other user-defined tag tables

All tags

The "All tags" table gives an overview of all PLC tags, user constants and system constants of the CPU. This table cannot be deleted or moved.

Standard tag table

There is one standard tag table for each CPU of the project. It cannot be deleted, renamed or moved. The default tag table contains PLC tags, user constants and system constants. You can declare all PLC tags in the default tag table, or create additional user-defined tag tables as you want.

User-defined tag tables

You can create multiple user-defined tag tables for each CPU to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. User-defined tag tables can contain PLC tags and user constants.

See also

Structure of the PLC tag tables (Page 7070)

Using tags within the program (Page 95)

Basics of constants (Page 97)

11.2 Structure of the PLC tag tables

Introduction

Each PLC tag table contains a tab for tags and a tab for user constants. The default tag table and the "All tags" table also have a "System constants" tab.

Structure of the "PLC tags" tab

In the "Tags" tab you declare the global PLC tags that you require in the program. The following figure shows the tab structure. The number of columns shown may vary.

	Name	Data type	Address	Retain
	Motor1	Bool	%Q3.1	
	Motor2	Bool	%Q3.2	
	Control	Bool	%I3.3	

The following table shows the meaning of the individual columns. The number of columns shown may vary. You can show or hide the columns as required.

Column	Explanation
	Symbol you can click on to drag-and-drop a tag to a program for use as an operand.
Name	Unique name for the constants throughout the CPU.
Data type	Data type of the tags.
Address	Tag address.
Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off.
Visible in HMI engineering	Shows whether the tag is visible by default in the operand selection of HMI.
Accessible from HMI/OPC UA	Indicates whether HMI/OPC UA can access this tag during runtime.
Writable from HMI/OPC UA	Indicates whether the tag from HMI/OPC UA can be written during runtime.
Supervision	Indicates whether Supervision has been created for process diagnostics for the tag.
Monitor value	Current data value in the CPU. This column only appears if an online connection is established and you select the "Monitor all" button.
Tag table	Shows which tag table includes the tag declaration. This column is only available in the "All tags" table.
Comment	Comment to document the tags.

Structure of the "User constants" and "System constants" tabs

In the "User constants" you define symbolic constants that are valid throughout the CPU. The constants required by the system are shown in the "Systems constants" tab. System constants can be hardware IDs, for example, which can be used for identification of modules.

The following figure shows the structure of both tabs. The number of columns shown may vary.

	Name	Data type	Value	Comment
	Const_1	Bool	true	
	Const_2	Byte	12	
	Const_3	Bool	false	
	Const_4	Real	1.0	

The following table shows the meaning of the individual columns. You can show or hide the columns as required.

Column	Explanation
	Symbol you can click to move a tag into a network via a drag-and-drop operation for use as an operand.
Name	Unique name for the constants throughout the CPU.
Data type	Data type of the constants
Value	Value of the constants
Tag table	Shows which tag table includes the constant declaration. This column is only available in the "All tags" table.
Comment	Comments to document the tags.

See also

Hardware data types (Page 351)

Using tags within the program (Page 95)

Basics of constants (Page 97)

Overview of PLC tag tables (Page 7069)

Show and hide table columns (Page 7098)

11.3 Rules for PLC tags

11.3.1 Valid names of PLC tags

Permissible characters

The following rules apply to the use of names for PLC tags:

- Letters, numbers, special characters are permitted.
- The use of reserved keywords is not recommended.
- Quotation marks are not permitted as a component of the tag name.

Unique tag names

The names of the PLC tags must be unique throughout the CPU, even if the tags are located in different tag tables of a CPU. A name that is already used for a block, another PLC tag or a constant within the CPU, cannot be used for a new PLC tag. The uniqueness check does not differentiate between use of small and capital letters.

If you enter an already assigned name another time, a sequential number is automatically appended to the second name entered. For example, if you enter the name "Motor" a second time, the second entry is changed to "Motor(1)".

Unique table names

The names of the PLC tag tables must also be unique throughout the CPU. A unique name is automatically suggested when user-defined PLC tag tables are being created.

Note

You can find more information on permissible characters in tag names in the Siemens Industry Online Support in the following entries:

When should identifiers and operands be used in "quotation marks" in STEP 7 (TIA Portal)?

https://support.industry.siemens.com/cs/ww/en/view/109477857 (https://support.industry.siemens.com/cs/ww/en/view/109477857)

See also

Using tags within the program (Page 95)

Keywords (Page 89)

Permissible addresses and data types of PLC tags (Page 7073)

Entering a PLC tag declaration (Page 7079)

11.3.2 Permissible addresses and data types of PLC tags

The addresses of PLC tags are made up of the particulars of the operand area and the address within this area.

The addresses must be unique throughout the CPU. If you enter an address that is already assigned to another tag, the address will be highlighted at both places in yellow and an error message will be issued.

Operand areas

The following table shows the possible operand areas. The available data types depend on the CPU you use:

Operand area		Explanation	Data type	Format	Address area:		
International mnemonics	German mnemonics				S7-1200	S7-300/400	S7-1500
I	E	Input bit	BOOL	I x.y E x.y	0.0..1023.7	0.0..65535.7	0.0..32767.7
I	E	Input (64-bit)	LWORD, LINT, ULINT, LTIME, LTOD, LDT, LREAL, PLC data type	I x.0 E x.0	-	-	0.0..32760.0
IB	EB	Input byte	BYTE, CHAR, SINT, USINT, PLC data type	IB x EB y	0..1023	0..65535	0..32767
IW	EW	Input word	WORD, INT, UINT, DATE, WCHAR, S5TIME, PLC data type	IW x EW y	0..1022	0..65534	0..32766
ID	ED	Input double word	DWORD, DINT, UDINT, REAL, TIME, TOD, PLC data type	ID x ED y	0..1020	0..65532	0..32764
Q	A	Output bit	BOOL	Q x.y A x.y	0.0..1023.7	0.0..65535.7	0.0..32767.7
Q	A	Output (64-bit)	LWORD, LINT, ULINT, LTIME, LTOD, LDT, LREAL, PLC data type	Q x.0 A x.0	-	-	0.0..32760.0

Operand area		Explanation	Data type	Format	Address area:		
International mnemonics	German mnemonics				S7-1200	S7-300/400	S7-1500
QB	AB	Output byte	BYTE, CHAR, SINT, USINT, PLC data type	QB x AB y	0..1023	0..65535	0..32767
QW	AW	Output word	WORD, INT, UINT, DATE, WCHAR, S5TIME, PLC data type	QW x AW y	0..1022	0..65534	0..32766
QD	AD	Output double word	DWORD, DINT, UDINT, REAL, TIME, TOD, PLC data type	QD x AD y	0..1020	0..65532	0..32764
M	M	Memory bit	BOOL	M x.y	0.0..8191.7	0.0..65535.7	0.0..16383.7
M	M	Bit memory (64-bit)	LREAL	M x.0	0.0..8184.0	-	0.0..16376.0
M	M	Bit memory (64-bit)	LWORD, LINT, ULINT, LTIME, LTOD, LDT	M x.0	-	-	0.0..16376.0
MB	MB	Memory byte	BYTE, CHAR, SINT, USINT	MB x	0..8191	0..65535	0..16383
MW	MW	Memory word	WORD, INT, UINT, DATE, WCHAR, S5TIME	MW x	0..8190	0..65534	0..16382
MD	MD	Memory double word	DWORD, DINT, UDINT, REAL, TIME, TOD	MD x	0..8188	0..65532	0..16380
T	T	Time function (for S7-300/400 only)	Timer	T n	-	0..65535	0..2047
C	Z	Counter function (for S7-300/400 only)	Counter	Z n C n	-	0..65535	0..2047

Addresses

The following table shows the possible addresses of tags:

Data type	Address	Example
BOOL	Tags with BOOL data type are addressed with a byte number and a bit number. The numbering of the bytes begins at 0 for each operand range. The numbering of the bits is from 0 to 7.	A 1.0
BYTE, CHAR, SINT, USINT	Tags with BYTE, CHAR, SINT, and USINT data type are addressed with a byte number.	MB 1
WORD, INT, UINT, DATE, WCHAR, S5TIME	Tags with WORD, INT, UINT, DATE, S5TIME data type consist of two bytes. They are addressed with the number of the lowest byte.	IW 1
DWORD, DINT, UDINT, REAL, TIME, TOD	Tags with DWORD, DINT, UDINT, REAL, TIME, TOD data type consist of four bytes. They are addressed with the number of the lowest byte.	AD 1
LWORD, LINT, ULINT, LTIME, LTOD, LDT, LREAL	Tags of data type LWORD, LINT, ULINT, LTIME, LTOD, LDT, and LREAL consist of eight bytes. They are addressed with it number 0 and the number of the lower byte.	I 1.0

Mnemonics used

The addresses that you enter in the PLC tag table are automatically adapted to the set mnemonics.

See also

Setting the mnemonics (Page 6843)

Using tags within the program (Page 95)

Valid names of PLC tags (Page 7072)

Overview of the valid data types (Page 231)

Entering a PLC tag declaration (Page 7079)

11.4 Creating and managing PLC tag tables

11.4.1 Creating a PLC tag table

You can created multiple user-defined PLC tag tables in a CPU. Each tag table must have have a unique name throughout the CPU.

Requirement

The project view is open.

Procedure

To created a new PLC tag table, follow these steps:

1. Open the "PLC tags" folder under the CPU in the project tree.
2. Double-click the "Add new tag table" entry.
   A new PLC tag table with the default name "TagTable_x" is created.
3. Select the PLC tag table in the project tree.
4. Select the "Rename" command in the shortcut menu.
5. Type in a name that is unique throughout the CPU.

Result

A new PLC tag table is created. You can now declare tag and constants in this table.

See also

Overview of PLC tag tables (Page 7069)

Structure of the PLC tag tables (Page 7070)

Declaring PLC tags (Page 7079)

Declaring global constants (Page 7088)

Exporting and importing PLC tags (Page 7098)

11.4.2 Grouping PLC tag tables

You can gather the user-defined tag tables of the CPU into groups. You cannot, however, move the standard tag table and the "All tags" table into a group.

Requirement

Multiple user-defined tag tables are contained in the "PLC tags" folder of the CPU.

Procedure

To gather multiple PLC tag tables into a group, follow these steps:

1. Select the "PLC tags" folder under the CPU in the project tree.
2. Select the "Insert > Group" menu command.
   A new group with the standard name "Group_x" is inserted.
3. Select the newly inserted group in the project tree.
4. Select the "Rename" command in the shortcut menu.
5. Assign the new group a unique name throughout the CPU.
6. Drag to the new group the tables you want to group together.

Result

The tag tables are gathered in the new group.

See also

Overview of PLC tag tables (Page 7069)

Structure of the PLC tag tables (Page 7070)

Declaring PLC tags (Page 7079)

Declaring global constants (Page 7088)

11.4.3 Opening the PLC tag table

Procedure

To open the PLC tag table in a CPU, proceed as follows:

1. Open the "PLC tags" folder under the CPU in the project tree.
2. Double-click the PLC tag table in the folder.
3. Select the desired tab in the upper corner.

Result

The PLC tag table associated with the CPU opens. You can declare the required tags and constants.

See also

Overview of PLC tag tables (Page 7069)

Structure of the PLC tag tables (Page 7070)

Declaring PLC tags (Page 7079)
Declaring global constants (Page 7088)

11.5 Declaring PLC tags

11.5.1 Entering a PLC tag declaration

11.5.1.1 Declaring tags in the PLC tag table

Requirement

The "Tags" tab of the PLC tag table is open.

Procedure

To define PLC tags, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
   An address corresponding to the data type is automatically appended.
3. Optional: Click on the arrow key in the "Address" column and enter an operand identifier, an operand type, an address and a bit number in the dialog which then opens.
4. Optional: Disable the options for HMI and OPC UA.
5. Optional: Enter a comment in the "Comments" column.
6. Repeat steps 1 - 5 for all the required tags.
   See also: Permissible addresses and data types of PLC tags (Page 7073)

Syntax check

A syntax check is performed automatically after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. As long as the tag declaration contains syntax errors and the tag is used in the program, you will not be able to compile the program.

See also

Exporting and importing PLC tags (Page 7098)

Using autocompletion (Page 6859)

Valid names of PLC tags (Page 7072)

Declaring PLC tags in the program editor (Page 7080)

Editing PLC tag tables (Page 7096)

Structure of the PLC tag tables (Page 7070)

11.5.1.2 Declaring PLC tags in the program editor

Requirement

- The program editor is open.

Procedure

To declare operands as global PLC tags, follow these steps:

1. Insert an instruction in your program.
   The "<???>", "<??.?>" or "..." strings represent operand placeholders.
2. Replace an operand placeholder with the name of the PLC tag to be created.
3. Select the tag name.
   If you want to declare multiple PLC tags, select the names of all the tags to be declared.
4. Select the "Define tag" command in the shortcut menu.
   The "Define tag" dialog box opens. This dialog displays a declaration table in which the name of the tag is already entered.
5. Click the arrow key in the "Section" column and select one of the following entries:
   - Global Memory
   - Global Input
   - Global Output
   - Global Timer
   - Global Counter
6. In the other columns, enter the address, data type, and comments.
   See also: Permissible addresses and data types of PLC tags (Page 7073)
7. If the CPU contains multiple PLC tag tables, you can use an entry in the "PLC tag table" column to indicate in which table the tag is to be inserted. If you make no entry in the column, the new tag will be inserted in the default tag table.
8. Click the "Define" button to complete your entry.

Result

The tag declaration is written to the PLC tag table and is valid for all blocks in the CPU.

See also

Valid names of PLC tags (Page 7072)

Declaring tags in the PLC tag table (Page 7079)

Using autocompletion (Page 6859)

11.5.1.3 Navigate to the device view

You can navigate from a PLC tag table to the device and network editor and display the configured input or output in the device view.

Requirements

- The tag is declared as input or output in a PLC tag table.
- An input or output group is configured in the hardware and network editor.

Procedure

To display a PLC tag in the device view, follow these steps:

1. Select the tag in the PLC tag table.
2. Select the shortcut menu command "Go to device view".

The device and network editor is opened and the input or output for which the tag was declared is displayed.

11.5.2 Setting the retentivity of PLC tags

11.5.2.1 Retentive behavior of PLC tags

Retentive PLC tags

Each CPU has a memory area whose content remains available even after the supply voltage has been switched off. This area is referred to as retentive memory area.

To avoid data loss during power failure, you can save specific PLC tags to this memory area. You make the retentivity setting of PLC tags in the PLC tag table (S7-1200/S7-1500) or in the CPU properties (S7-300/S7-400).

Depending on the CPU family, the retentive memory area can accommodate various type of PLC tags. The following table provides an overview of the options of the various CPUs:

CPU type	Retentive bit memories	Retentive SIMATIC timers	Retentive SIMATIC counters
S7-300/400 series	✓	✓	✓
S7-1200 series	✓	-	-
S7-1500 series	✓	✓	✓

See also

Setting the retentive behavior of PLC tags (Page 7082)

Setting the retentive behavior of PLC tags (Page 7082)

11.5.2.2 Setting the retentive behavior of PLC tags (S7-1200, S7-1500)

Introduction

You set the retentivity for PLC tags in the PLC tag table for the S7-1200/S7-1500 CPU series. You can recognize the retentivity setting of a tag by the check mark set in the "Retain" column of the PLC tag table.

Requirement

The "PLC tags" tab of the PLC tag table is open.

Procedure

To define the width of the retentive memory area for PLC tags of CPU S7-1200/S7-1500, follow these steps:

1. On the toolbar, click the "Retain" button.
   The "Retain memory" dialog will open.
2. Set the width of the retentive memory area.
   - Enter the number of retentive memory bytes.
     Example: If you enter "2", memory bytes MB 0 and MB 1 are retentive.
   - Enter the number of retentive timers. Each S7 timer occupies two bytes.
     Example: If you enter "2", S7 timers T 0 and T 1 are retentive.
   - Enter the number of retentive counters. Each S7 counter occupies two bytes.
   - Example: If you enter "2", S7 counters C 0 and C 1 are retentive.
3. Click the "OK" button.

Result

The width if the retentive memory area is defined. All tags with addresses in this memory area are then designated retentive. In the "Retain" column of the tag table a check mark is automatically set for all tags that are located within the retentive memory area.

See also

Retentive behavior of PLC tags (Page 7081)

11.5.2.3 Setting the retentive behavior of PLC tags (S7-300, S7-400)

Introduction

You set the retentivity for PLC tags in the CPU settings for the S7-300/400 CPU series.

Procedure

To define the width of the retentive memory area for PLC tags of CPU S7-300/400, follow these steps:

1. Select the CPU in project tree.
2. Select the "Properties" menu command.
3. Select the section "Retentivity".
   - Enter the number of retentive memory bytes.
     Example: If you enter "2", memory bytes MB 0 and MB 1 are retentive.
   - Enter the number of retentive timers. Each S7 timer occupies two bytes.
     Example: If you enter "2", S7 timers T 0 and T 1 are retentive.
   - Enter the number of retentive counters. Each S7 counter occupies two bytes.
     Example: If you enter "2", S7 counters C 0 and C 1 are retentive.

Result

The width if the retentive memory area is defined. All tags within this memory area are then declared retentive.

See also

Retentive behavior of PLC tags (Page 7081)

11.6 Grouping PLC tags for inputs and outputs in structures

11.6.1 Other useful information regarding structured PLC tags

Use of structured PLC tags (S7-1200 V4 and higher/S7-1500)

To make your program easier to view, you can group several input or output addresses in a higher-level PLC tag. The higher-level PLC tag represents a structure that contains several logically related inputs or outputs. When the block is called, you transfer the higher-level tag and thus need only an input or output parameter for all associated inputs or outputs.

Principle of operation

To create a structured PLC tag, you initially define a PLC data type (UDT). In it you declare the necessary data elements and specify their names and data types.

Then, you switch to the PLC tag table and specify the higher-level PLC tag there. As a data type for the tag, you select your PLC data types. The system now reserves a certain number of input or output addresses starting from the start address of the higher-level tag. The number of reserved addresses depends on the length of your PLC data type.

If you call a block that requires the reserved inputs or outputs for program execution, you transfer the higher-level tag as a block parameter.

You can address the individual PLC tags like structure elements in the program code.

A detailed description of the individual handling steps can be found in the following chapters.

Application example

You can use structured PLC tags in order to group the inputs or outputs of a function module. The following figure shows the schematic representation of a motor: A component was created in the "Datatype_Motor" PLC data type for each of the three inputs.

The memory areas of the declared tags must not overlap. In the example, you see that the "Speed" component has the data type "Integer" and therefore must start at a word address. For this reason, the first input word has been filled with the "Dummy" fill tag. This means that "Speed" is located on the second input word.

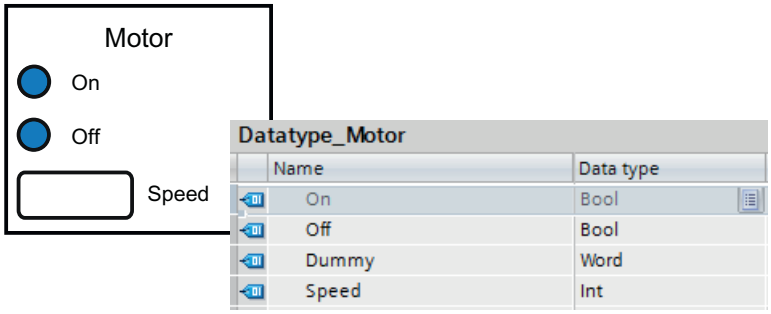

The following figure shows the higher-level "Motor" PLC tag that is based on the "Data type_Motor" data type. With the declaration of "Motor", the addresses IW0 and IW1 are reserved on the input module.

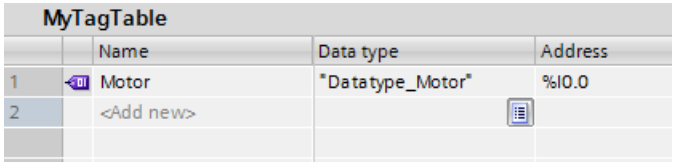

The following figure shows the transfer of the "Motor" PLC tag as an input parameter of the "Motor_Control" block.

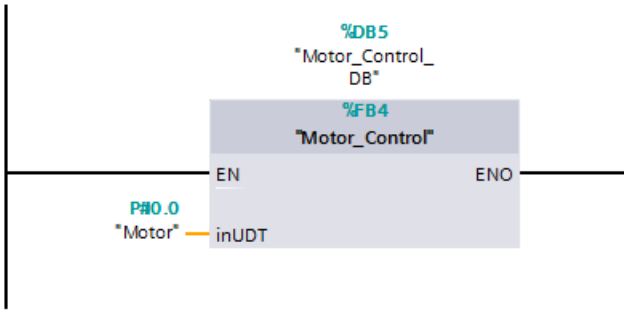

You can address the individual components of the tag in the "Motor_Control" block.

Addressing	Description
"Motor"	Addressing the higher-level PLC tag.
"Motor".On	Addressing a component of a structured PLC tag.
"Motor".On:P	Addressing an I/O input or output (PI or PQ).

Rules for use of structured PLC tags

Note the following rules when creating and using structured PLC tags.

- Structured PLC tags can be used in the "Inputs" and "Outputs" operand areas.
- Structured tags are not permitted in the bit memory address area.
- Structured PLC tags cannot be addressed from HMI.

Observe the following rules when creating the PLC data type that is to serve as the basis for a PLC tag:

- The memory areas of the individual elements must not overlap.
  See also: Permissible addresses and data types of PLC tags (Page 7073)
- Do not group inputs and outputs in one PLC data type but create different PLC data types for inputs and outputs.

- Do not group inputs or outputs from different modules in a PLC data type because it is not guaranteed that the process images of the modules are updated synchronously.
- In the lower-level PLC data types, all data types are permitted except for "STRING" and "WSTRING".

See also

Creating structured PLC tags (Page 7086)

Addressing global variables (Page 103)

11.6.2 Creating structured PLC tags

Rules

Note the following rules when creating structured PLC tags:

- Use separate PLC data types for the "Inputs" and "Outputs" operand areas.
- Structured tags are not permitted in the bit memory address area.
- Do not group inputs or outputs from different modules in a PLC data type because it is not guaranteed that the process images of the modules are updated synchronously.

Procedure

To create a structured PLC tag, follow these steps:

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare all the necessary components in the PLC type. All data types except for "STRING" and "WSTRING" are permitted.
3. Select the PLC data type in the project tree and select the "Compile > Software (only changes)" command from the shortcut menu.
   The PLC data type is compiled and can then be used in the PLC tag table.
   Even when you make changes to existing PLC data types, you must recompile the program. This updates all locations of use of the PLC data type.
4. Open a PLC tag table within the same CPU.
5. Declare a new tag or select an existing tag.
6. In the "Data type" column, select the PLC data type and assign it to the PLC tag.
   The PLC tag receives the structure of the PLC data type. A suitable address is assigned automatically. Structured PLC tags always start at word addresses.
   The highest structure element is displayed without its subelements in the table.

Note

Assignment rules and default values

- For the declaration of the PLC data type, note that the memory areas of the individual tags must not overlap. For example, tags of data type "Integer" must start at a word limit. If necessary, insert "fill tags" to prevent overlaps.
  See also: Permissible addresses and data types of PLC tags (Page 7073)
- It is not possible to assign default values for the individual components. Values that you enter in the "Default value" column are not evaluated. Tags of data types "DT" and "DTL" may therefore contain invalid values.

See also

Other useful information regarding structured PLC tags (Page 7084)

Addressing global variables (Page 103)

Declaring PLC tags (Page 7079)

11.7 Declaring global constants

11.7.1 Rules for global user constants

Permitted characters

Names of global constants may consist of the following characters:

- Letters, numbers, special characters are permitted.
- Quotation marks are not permitted.

Unique constant names

The names of global constants must be unique throughout the CPU, even if the constants are located in different tag tables of a CPU. A name that is already used for a block, a PLC tag or another constant within the CPU, cannot be used for new constant. The uniqueness check does not differentiate between use of small and capital letters.

If you enter an already assigned name another time, a sequential number is automatically appended to the second name entered. For example, if you enter the name "Motor" a second time, the second entry is changed to "Motor(1)".

Permitted data types

For constants, all data types supported by the CPU are permitted, with the exception of structured data types.

Permitted values

You can select any value from the value range of the specified data type as constant value. For information on the value ranges, refer to the "Data types" chapter.

See also: Overview of the valid data types (Page 231)

See also

Basics of constants (Page 97)

Declaring global constants (Page 7089)

Constants (Page 97)

11.7.2 Rules for global system constants

Definition

System constants are global constants, unique throughout the CPU, that are required and automatically created by the system. System constants can, for example, be used to address and identify hardware objects.

Rules

System constants are assigned automatically when components are inserted in the device or network view, and entered in the default tag table ("System constants" tab). A system constant is created for each module, but also for each submodule. An integrated counter, for example, therefore receives a system constant as well. System constants consist of a symbolic name and a numeric HW identifier and cannot be changed.

Names of system constants

The names of the system constants are structured hierarchically. They consist of a maximum of four hierarchical levels, each of which is separated by a tilde "~". Based on the name, you can therefore recognize the "path" to the relevant hardware module.

Example

A system constant with the name "`Local~PROFINET_interface_1~Port_1`" designates Port 1 of the PROFINET interface 1 of the local CPU.

See also

Instructions for address conversion (Page 2738)

11.7.3 Declaring global constants

Introduction

You declare constants in the "User constants" tab of a PLC tag table. During declaration you have to to enter a symbolic name, a data type and a fixed value for each constant. You can select any value from the value range of the specified data type as constant value. For information on the value ranges, refer to the "Data types" chapter.

See also: Overview of the valid data types (Page 231)

Procedure

To declare constants, follow these steps:

1. Open a PLC tag table.
2. Open the "User constants" tab.
   The constants table opens.
3. Enter a constant name in the "Name" column.
4. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
5. Enter a constant value in the "Value" column; this constant value must be valid for the selected data type.
6. If you want, enter comments on the constants in the "Comments" column. The entry of a comment is optional.
7. If you want to declare additional constants, place the cursor in the next row and repeat steps 3 to 6.

Syntax check

A syntax check is performed automatically after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. As long as the tag declaration contains syntax errors and the constant is used in the program, you will not be able to compile the program.

See also

Opening the PLC tag table (Page 7077)

Inserting a table row in the PLC tag table (Page 7096)

Structure of the PLC tag tables (Page 7070)

Rules for global user constants (Page 7088)

Using autocompletion (Page 6859)

Constants (Page 97)

11.8 Editing properties

11.8.1 Editing the properties of PLC tags

11.8.1.1 Properties of PLC tags

Overview

The following table provides an overview of the properties of PLC tags. The display of properties may vary depending on the CPU type.

Group	Property	Description
General	Name	A unique name within the CPU.
	Data type	Data type of the tags.
	Address	Tag address.
	Retain	Shows whether the tag is in the retentive memory area.
	Comment	Comment on the tag.
History	Date created	Time when the tag was created (cannot be changed).
	Last modified	Time when the tag was last changed (cannot be changed).
Application	Visible in HMI engineering	Shows whether the tag is visible by default in the HMI selection list.
	Accessible from HMI/OPC UA	Indicates whether HMI/OPC UA can access this tag during runtime.
	Writable from HMI/OPC UA	Indicates whether the tag from HMI/OPC UA can be written during runtime.

See also

Editing the properties of PLC tags (Page 7091)

11.8.1.2 Editing the properties of PLC tags

Editing properties in a PLC tag table

To edit the properties of one or more tags, follow these steps:

1. In the project tree, double-click the PLC tag table that contains the tags.
   The PLC tag table opens.
2. Change the entries in the columns.

Editing addresses in the program editor

To edit the address of a tag in the program editor, follow these steps:

1. Select the tag name.
2. Select the "Rewire tag" command in the shortcut menu.
   The "Rewire tag" dialog will open. The dialog shows a declaration table.
3. Enter the new address in the "Address" column.
4. Click the "Change" button to confirm the input.

Editing names in the program editor

To edit the name of a tag in the program editor, follow these steps:

1. Select the tag name.
2. Select the "Rename tag" command in the shortcut menu.
   The "Rename tag" dialog opens. The dialog shows a declaration table.
3. Enter the new name in the "Name" column.
4. Click the "Change" button to confirm the input.

Effect in the program

In the case of a change of the tag's name, data type or address, each location of use of the tag is automatically updated in the program.

See also

Properties of PLC tags (Page 7091)

11.8.2 Editing properties of global constants

11.8.2.1 Properties of global constants

Overview

The following table gives an overview of the properties of constants:

Group	Property	Description
General	Name	A unique name within the table
	Data type	Data type of the constants
	Value	Value that you defined for the constants. This value must be compatible with the declared data type. See also: Overview of the valid data types (Page 231)
	Comment	Comment on the constants

Group	Property	Description
History	Date created	Time when the constant was created (cannot be changed)
	Last modified	Time when the constant was last changed (cannot be changed)

11.8.2.2 Editing properties of global constants

Editing properties in a PLC tag table

To edit the properties of one or more constants, follow these steps:

1. In the project tree, double-click the PLC tag table that contains the constants.
   The PLC tag table opens.
2. Open the "User constants" tab.
3. Change the entries in the "Name", "Data type", "Value", or "Comments" column.

Effect in the program

In the case of a change of a constant's name, data type or value, each location of use of the constant is automatically updated in the program.

11.8.3 Editing multilingual comments in the PLC tag table (S7-1500)

You can edit the comments from a PLC tag table in all project languages that you have activated in the project language settings. You can find additional information on activating project languages under AUTOHOTSPOT.

Note that the texts must not exceed a length of 32767 Unicode characters even after translation.

Requirement

- You have activated multiple project languages.
- A PLC tag table is open and it contains at least one comment.
- The block does not have know-how protection.

Procedure

To edit a comment in all project languages, follow these steps:

1. Click on the comment whose translation you want to edit.
   If you want to edit the translations of several comments at the same time, select all the desired comments.
   Comments of a PLC data type (UDT) cannot be selected in the PLC tag table. You can only edit these comments in the PLC data type itself.
2. Open the "Properties" tab in the Inspector window.
3. Open the "Texts" tab.
   The "Texts" tab displays the selected comments in the active project languages and, if available, the translations of the comments.
4. You can edit the translations directly in the table in the "Texts" tab.
5. To edit the translations with an external editor, you can export the displayed texts to OOXML format using the "Export/Import project texts" buttons.

Note

Editing all project texts in the global "Project texts" table

You can also edit the translations for the individual project languages in the global "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts". It contains all translatable texts of the entire project.

You can find additional information on translation of texts under AUTOHOTSPOT.

11.9 Monitoring of PLC tags

11.9.1 Monitoring of PLC tags

You can monitor the current data values of the tags on the CPU directly in the PLC tag table.

Requirement

An online connection to the CPU exists or is possible.

Procedure

To monitor the data values, proceed as follows:

1. Open a PLC tag table.
2. Start monitoring by clicking the "Monitor all" button.
   - If no online connection to the CPU exists, this is established.
   - Monitoring is started with the trigger setting "Permanent".
   - The additional "Monitor value" column is displayed in the table. This shows the current data values.
   - The symbol for the forcing of tags is displayed if a tag is currently being forced.
3. End the monitoring by clicking the "Monitor all" button again.

Note

Editing PLC tags during the monitoring of tags

If the editing of tags is started and the PLC tag table is edited, for example by adding new tags, the monitoring is re-started after editing is complete.

Note

You also have the option of copying PLC tags to a monitor or force table so that you can monitor, control or force them in the table.

See also

Structure of the PLC tag tables (Page 7070)

Introduction to testing with the watch table (Page 7739)

Introduction for testing with the force table (Page 7768)

Copying entries in the PLC tag table (Page 7096)

11.10 Editing PLC tag tables

11.10.1 Inserting a table row in the PLC tag table

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

11.10.2 Copying entries in the PLC tag table

You can copy PLC tags within a table or to other tables.

Procedure

To copy a tag, follow these steps:

1. Select the tags you want to copy.
   You can also select several tags by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last tag.
2. Select "Copy" in the shortcut menu.
3. Position the insertion pointer at the location where you want to insert the tags.
4. Select "Paste" in the shortcut menu.

Or

1. Select the tag.
2. Hold down the left mouse button.
3. At the same time, press <Ctrl>.
4. Drag the tag to the destination.

Result

- The tag is copied to the destination.
- If there is a name conflict, a number is automatically appended to the tag name. For example, "Tag" becomes "Tag(1)".
- All other properties of the tag remain unchanged.

11.10.3 Deleting entries in the PLC tag table

Procedure

Follow the steps below to delete elements:

1. Select the row with the element to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

11.10.4 Sorting rows in the PLC tag tables

You can sort the rows in the tables alphanumerically by name, data type, or address.

Procedure

To sort the table rows, follow these steps:

1. Select the column by which you want to sort.
2. Click the column header.
   The column will be sorted in order of increasing values.
   An up arrow shows the sort sequence.
3. In order to change the sort sequence, click the arrow.
   The column will be sorted in order of decreasing values. A down arrow shows the sort sequence.
4. To restore the original sequence, click a third time on the column header.

11.10.5 Automatically filling in cells in the PLC tag table

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

If you fill the cells in the column "Address" automatically, the addresses will be increased depending on the indicated data type.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.

3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically. If entries are already present in the cells that are to be automatically updated, a dialog appears in which you can indicate whether you want to overwrite the existing entries or whether you want to insert new rows for the new tags.

11.10.6 Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.
5. To hide or show several columns, click "More" and activate or deactivate the check box of the corresponding columns in the "Show/Hide" dialog.

11.10.7 Exporting and importing PLC tags

11.10.7.1 Basics for exporting and importing PLC tags

Introduction

You can export PLC tag tables to edit them further with external applications. Similarly, you can import PLC tag tables edited externally to the TIA Portal.

The following formats are supported for import and export:

- XLSX (Excel)
- XML
- SDF

For the export you can specify which elements you want to export:

- Tags and/or constants
- All elements or just the elements that are actually being used in the program on the current CPU.

Overwriting existing PLC tags and constants during import

Existing entries of the same name will be overwritten during import if they have the same name as the entries that will be imported.

Link to existing objects

References to PLC tags or constants that already exist in the project are updated automatically during import. The update is executed based on the name of the PLC tags and constants.

See also

Format of the export file (*.xlsx) (Page 7099)

Exporting PLC tags (Page 7101)

Importing PLC tags (Page 7101)

11.10.7.2 Format of the export file (*.xlsx)

Introduction

During the export of PLC tag tables, a standardized XSLX format will be generated that you can edit with external table editors.

This format is also expected during the import of tables.

Format of the export file

The sheet is always named "PLCTags". This sheet can contain the displayed columns. The sorting order of columns can vary. Not all columns must exist, but the table may not contain any additional columns. During import, the following values will be identified by a <no value> entry.

The names of the column headers are also clearly defined and are always expected in English.

The following table specifies the contents expected for the individual columns:

Element	Description
Name	Name of the tags
Path	Group and name of the PLC tag table
Data Type	The notation of the data type corresponds to the notation used in the PLC tag table.
Logical Address	The address can be specified with German or international mnemonics.
Comment	Free-form comments
Hmi Visible	The value TRUE or FALSE is expected.
Hmi Accessible	The value TRUE or FALSE is expected.
Hmi Writeable	The value TRUE or FALSE is expected.

See also

Basics for exporting and importing PLC tags (Page 7098)

Exporting PLC tags (Page 7101)

Importing PLC tags (Page 7101)

11.10.7.3 Format of the export file (*.xml)

Introduction

During the export of PLC tag tables, a standardized xml format will be generated. This format is also expected during the import of xml files.

Example of an xml export file

The following example shows a PLC tag table which was exported to the *.xml format:

```
1  <?xml version='1.0' encoding='utf-8'?>
2  <Tagtable name='Default tag table'>
3    <!-- exported by 'Totally Integrated Automation Portal' version 15.0.1.0
     2004-2016 Siemens AG - all rights reserved -->
4    <Tag type='Bool' hmiVisible='True' hmiWriteable='True' hmiAccessible='True'
     retain='False' remark='' addr='%I0.0'>myTag</Tag>
5    <Constant type='' remark='' value=''>myConstant</Constant>
6  </Tagtable>
```

11.10.7.4 Format of the export file (*.sdf)

Introduction

During the export of PLC tag tables, a standardized sdf format will be generated. This format is also expected during the import of sdf files.

Example of an sdf export file

The following example shows a PLC tag table which was exported to the *.sdf format:

```
"Tag_1","%I0.0","Bool","True","True","False","Comment","","True"
"Tag_2","%I0.1","Bool","True","True","False","","","True"
"Const_1","CONSTANT","Bool","","","","Comment1","TRUE",""
"Const_2","CONSTANT","Bool","","","","","TRUE",""
```

11.10.7.5 Exporting PLC tags

Requirement

A PLC tag table is open.

Procedure

To export PLC tags and constants, follow these steps:

1. In the PLC tag table, click the "Export" button.
   The "Export" dialog box opens.
2. Click the selection button under "Path of export file".
   The "Save as" dialog opens.
3. Enter the name of the export file and select the file in which it is to be stored.
4. Select *.xlsx, *.sdf or *.xml as the file type.
5. Click the "Save" button.
6. Now select the export options:
   - Select whether to export tags and/or constants.
   - Select whether you want to export all elements of the table or just the elements that are actually being used in the program on the current CPU.
7. Click the "OK" button.

Result

The export file will be generated. Errors and warnings generated during export are indicated in the "Info" tab of the Inspector window.

See also

Basics for exporting and importing PLC tags (Page 7098)

Format of the export file (*.xlsx) (Page 7099)

Importing PLC tags (Page 7101)

11.10.7.6 Importing PLC tags

Requirement

A PLC tag table is open.

Procedure

To import PLC tags, follow these steps:

1. Click the "Import" button.
   The "Import" dialog box opens.
2. Select the file you want to import.
3. Specify whether you want to import tags and/or constants.
4. Click the "OK" button.

Result

The tags from the import file are imported into the TIA Portal.

They are inserted into the PLC tag table which has the same name as the import file. Existing entries of the same name are overwritten if they have the same name as the entries to be imported. If there is no PLC tag table with the name of the import file in the project, it is created and the tags are inserted.

References to PLC tags or constants that already exist in the project are updated automatically during import. The update is executed based on the name of the PLC tags and constants.

Errors and warnings generated during import are indicated in the import log.

See also

Basics for exporting and importing PLC tags (Page 7098)

Format of the export file (*.xlsx) (Page 7099)

Exporting PLC tags (Page 7101)

11.10.7.7 Editing individual PLC tags with external editors

To edit individual PLC tags in external editors outside the TIA portal, you can export or import these tags using copy and paste. However, you cannot copy structured tags to an editor.

Requirement

A PLC tag table and an external editor are opened.

Procedure

To export and import individual PLC tags, follow these steps:

1. Select one or more PLC tags.
2. Select "Copy" in the shortcut menu.
3. Switch to the external editor and paste the copied tags.

4. Edit the tags as required.
5. Copy the tags in the external editor.
6. Switch back to the PLC tag table.
7. Select "Paste" in the shortcut menu.

Note

You also have the option of export or importing PLC tags as mass data.

See also: Exporting and importing PLC tags (Page 7098)

Declaring PLC data types (UDT) 12

12.1 Structure of the declaration table for PLC data types

Structure of the declaration table for PLC data types

The figure below shows the structure of the declaration table for PLC data types.

Name	Data type	Default value
Motor	Bool	false
MyTag1	Struct	
Element1	Bool	false
Element2	Bool	false
MyTag2	Bool	false

Meaning of the columns

The following table shows the meaning of the individual columns. You can show or hide the columns as required. The number of columns displayed varies depending on the CPU series.

Column	Explanation
	Symbol you can click to move or copy the tag.
Name	Name of the tags.
Data type	Data type of the tags.
Default value	Value with which you predefine the tag in the declaration of the PLC data type. Specification of the default value is optional. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL.
Visible in HMI engineering	Shows whether the tag is visible by default in the HMI selection list.
Accessible from HMI/OPC UA	Indicates whether HMI/OPC UA can access this tag during runtime.
Writable from HMI/OPC UA	Indicates whether the tag from HMI/OPC UA can be written during runtime.
Setpoint value	Setpoint values are the values that will probably have to be fine tuned during commissioning. After commissioning, the values of these tags can be transferred to the offline program as start values and stored there.
Comment	Comment to document the tags.

See also

Creating PLC data types (Page 7108)
Show and hide table columns (Page 7119)
Basics of PLC data types (UDT) (Page 273)
Using setpoints during commissioning (Page 7063)

12.2 PLC data types in the project tree

"PLC data types" folder

The project tree contains a "PLC data types" folder in which you can create and manage the following PLC data types (UDT):

- User-created PLC data types
  PLC data types that you create yourself are located directly in the folder and can be opened, deleted and renamed there.
- System data types
  PLC data types created by the system are located in the "System data types" subfolder. This folder is created when you create a data block based on a system data type for the first time or when you drag an instruction that uses a system data type into the program. If the system data type is no longer being used in the program, it is deleted automatically during compilation. The "System data types" folder is deleted when it no longer contains any data types.

Note

Visibility of system data types in the project tree

Note that not all system data types are visible in the project tree. Internal system data types such as "IEC_COUNTER" are not displayed in the "System data types" folder.

Moving or copying PLC data types

You can move or copy PLC data types between the "System data types" folder and other folders. However, the following rules must be observed:

- Moving or copying PLC data types to the "System data types" folder:
  PLC data types in the "System data types" folder that are not required for user program execution are removed during the next compilation. This may cause your moved PLC data type to be deleted.
- Moving data types from the "System data types" folder:
  System data types can be moved, but not copied, from the "System data types" folder to the "PLC data types" folder above it. Data types not located in the "System data types" folder are not managed by the system, however. They are thus not automatically deleted when they are no longer used in the program.

See also

Basics of PLC data types (UDT) (Page 273)

Creating PLC data types (Page 7108)

Structure of the declaration table for PLC data types (Page 7105)

12.3 Creating PLC data types

Requirement

The "PLC data types" folder opens in the project tree.

Procedure

To create a PLC data type, proceed as follows:

1. In the "PLC data types" folder, click the "Add new data type" command.
   A new declaration table for creating a PLC data type will be created and opened.
2. Select the PLC data type and select the "Rename" command in the shortcut menu.
3. Enter the name of the PLC data type.

The new PLC data type is created. You can find the PLC data type in the project tree in the "PLC data types" folder.

Adding comments for a PLC data type

To enter a comment for a PLC data type, follow these steps:

1. Right-click the PLC data type in the project tree.
2. Select the "Properties" command in the shortcut menu.
   A dialog containing the properties of the PLC data type opens.
3. Select the "Information" entry in the area navigation.
4. Enter the comment in the "Comment" input field.
5. Confirm your entry with "OK".

See also

Structure of the declaration table for PLC data types (Page 7105)

Basics of PLC data types (UDT) (Page 273)

12.4 Delete PLC data types

Requirement

The PLC data type you want to delete is not open.

Procedure

To delete a PLC data type, follow these steps:

1. In the project tree, open the "PLC data types" folder.
2. Select the PLC data type to be deleted. You can also select several PLC data types by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last data type.
3. Select the "Delete" command in the shortcut menu.

Note

If you delete a PLC data type, the blocks that use the data type will become inconsistent. These inconsistencies are marked in red in the block used. To remedy these inconsistencies, the data blocks have to be updated.

See also

Updating the block interface (Page 7015)

Updating data blocks (Page 7025)

Basics of PLC data types (UDT) (Page 273)

12.5 Renumbering PLC data types

For performance reasons, PLC data types are processed internally with numbers. If there are number conflicts, these are resolved automatically. But this is not possible for a PLC data type which is used by a know-how protected block. The block must be recompiled when you change the number of the PLC data type which results in a password prompt for the know-how protected block. You can bypass this step by setting up a separate numbering scheme for your PLC data types. Use numbers greater than 5000.

Procedure

To change the default number of a PLC data type, follow these steps:

1. Open the project library in the "Libraries" task card.
2. Drag the compilable PLC data type to the "Types" folder.
   The "Add type" dialog opens.
3. Enter the properties of the new type.
4. Click "OK" to confirm.
5. Right-click the PLC data type in the project library and select the "Edit type" command from the shortcut menu.
6. Click "OK" to confirm your selection of the instance.
   The library view opens.
7. Close the library view.
   The extension "in test" is now added to the name of the PLC data type.
8. Right-click the PLC data type and select the "Properties" command from the shortcut menu.
9. Select the "General" group in the area navigation.
   You can now edit the number of the PLC data type.
10. Change the number of the PLC data type.
11. Click "OK" to confirm.
12. Right-click the PLC data type in the project library and select the "Release version" command from the shortcut menu.
    The PLC data type has a new number. The assigned number is retained even if the type of the PLC data type is revoked.

12.6 Programming the structure of PLC data types

12.6.1 Declaring tags of elementary data type

Requirement

A PLC data type is open.

Procedure

To declare a tag, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
3. Optional: Change the properties of the tags that are displayed in the other columns.
4. Repeat steps 1 to 3 for all tags that are to be declared.

See also

Editing the declaration table for PLC data types (Page 7118)

12.6.2 Declaring tags of the ARRAY data type

Requirement

A PLC data type is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the "Array" data type in the "Data type" column. You will be supported by Autocomplete in this step.
   The "Array" dialog opens.
3. In the "Data type" text box, specify the data type of the array elements.
4. In the "ARRAY limits" input field, specify the high and low limit for each dimension.
   Example of a one-dimensional ARRAY:
   `[0..3]`
   Example of a three-dimensional ARRAY:
   `[0..3, 0..15, 0..33]`

5. Confirm your entry.
6. Optional: Change the properties of the tags that are displayed in the other columns.

The tag is created but remains collapsed. To expand the ARRAY, click the triangle in front of the tag. Note that you cannot expand ARRAYs with more than 10,000 elements for reasons of clarity.

Entering default values of ARRAY elements

To pre-assign default values to the individual elements of an ARRAY, follow these steps:

1. Click the triangle in front of the ARRAY data type tags.
   The ARRAY is expanded and the individual ARRAY elements are shown in separate lines.
2. Enter the desired values in the "Default value" column.

Adding comments for ARRAYs

To add a comment for the entire ARRAY, enter the comment in the top line of the ARRAY declaration. The comment is used as a preset comment for all lower-level ARRAY elements.

To provide specific comments for the individual elements of an ARRAY, follow these steps:

1. Click the triangle in front of the ARRAY data type tags.
   The ARRAY is expanded and the individual ARRAY elements are shown in separate lines.
2. Enter the desired values in the "Comment" column.

Displaying ARRAYs in expanded mode

In "Expanded mode" all ARRAYs are displayed expanded by default.

To enable the expanded mode, follow these steps:

1. Click the "Expanded mode" button in the toolbar.

Note

Restriction in "Expanded mode" display

Depending on the amount of RAM available on your computer, the following maximum limits apply to expanded mode:

- RAM < 8 GB: A maximum of 10,000 rows are displayed.
- RAM >= 8 GB: A maximum of 40,000 rows are displayed.

See also

Structure of the declaration table for PLC data types (Page 7105)

ARRAY (Page 283)

Using autocompletion (Page 6859)

12.6.3 Declaring tags of STRUCT data type

Requirements

A PLC data type is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.
   An empty, indented row is inserted after the new tag.
3. Insert the first structural element in the first empty row.
   An additional empty row is inserted after the element.
4. Select a data type for the structure element.
5. Optional: Change the properties of the structural element that is displayed in the other columns.
6. Repeat steps 3 to 5 for all additional structure elements.
   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
7. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

See also

Basic information on STRUCT (Page 278)

Structure of the declaration table for PLC data types (Page 7105)

Using autocompletion (Page 6859)

12.6.4 Declaring tags based on a different PLC data type

Requirement

- A global data block is open.
- A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a different PLC data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the PLC data type in the "Data type" column. You will be supported by Autocomplete during input.
   The tag is created.
3. Optional: change the properties of the tags that are displayed in the other columns of the table.
   If you have already defined default values or comments for the tags within a PLC data type during declaration of the PLC data type, these are shown in gray. You can change these values here.
   The changed values are displayed in black and apply only to the specific point of use.

Note

If you change or delete PLC data types that are being used in the data block, the data block becomes inconsistent. To eliminate this inconsistency, you must recompile the program.

See also

Structure of the declaration table for PLC data types (Page 7105)

Using autocompletion (Page 6859)

12.7 Editing tag properties in PLC data types

12.7.1 Properties of tags in PLC data types

Properties

The following table gives an overview of tag properties in PLC data types:

Group	Property	Description
General	Name	Name of the tags.
	Data type	Data type of the tags.
	Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
	Start value	Not relevant in PLC data types
	Comment	Comment on the tag.
Attributes	Retain	Not relevant in PLC data types
User-defined attributes	CFC_Visible	Visible Indicates whether a parameter is visible in CFC.
	CFC_Configurable	Configurable Indicates whether a parameter is configurable in CFC.
	CFC_ForTest	For test Indicates whether a parameter is registered for the CFC test mode.
	CFC_Interconnectable	Interconnectable Indicates whether a parameter is interconnectable in CFC.
	CFC_EnableTagReadback	Enable tag readback Indicates whether a parameter is relevant for the "Read back chart" function in CFC.
	CFC_EnumerationTexts	Enumeration texts Assigns a parameter to an enumeration in CFC.
	CFC_EngineeringUnit	Engineering unit Assigns a parameter to a unit in CFC.
	CFC_LowLimit	Low limit Defines the low limit for the parameter in CFC.
	CFC_HighLimit	High limit Defines the high limit for the parameter in CFC.

See also

Changing the properties of tags in PLC data types (Page 7116)

Structure of the declaration table for PLC data types (Page 7105)

12.7.2 Changing the properties of tags in PLC data types

Editing general properties in the declaration table

To edit the general properties of one or more tags, follow these steps:

1. Open the PLC data type.
2. Change the entries in the columns.

Editing detailed properties in the properties window

To edit the detailed properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.
   The inspector window shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

See also

Updating the block interface (Page 7015)

Updating data blocks (Page 7025)

12.7.3 Editing multilingual comments in the PLC data type (S7-1500)

You can edit the comments from the PLC data type in all project languages that you have activated in the project language settings. You can find additional information on activating project languages under AUTOHOTSPOT.

Note that the texts must not exceed a length of 32767 Unicode characters even after translation.

Requirement

- You have activated multiple project languages.
- The PLC data type is open and it contains at least one comment.
- The data type does not have know-how protection.

Procedure

To edit a comment in all project languages, follow these steps:

1. Click on the comment whose translation you want to edit.
   If you want to edit the translations of several comments at the same time, select all the desired comments.
2. Open the "Properties" tab in the Inspector window.
3. Open the "Texts" tab.
   The "Texts" tab displays the selected comments in the active project languages and, if available, the translations of the comments.
4. You can edit the translations directly in the table in the "Texts" tab.
5. To edit the translations with an external editor, you can export the displayed texts to OOXML format using the "Export/Import project texts" buttons.

Note

Editing all project texts in the global "Project texts" table

You can also edit the translations for the individual project languages in the global "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts". It contains all translatable texts of the entire project.

You can find additional information on translation of texts under AUTOHOTSPOT.

12.8 Editing the declaration table for PLC data types

12.8.1 Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

12.8.2 Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

12.8.3 Deleting tags

Procedure

Follow the steps below to delete elements:

1. Select the row with the element to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

See also

Updating the block interface (Page 7015)

Updating data blocks (Page 7025)

12.8.4 Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

12.8.5 Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.
5. To hide or show several columns, click "More" and activate or deactivate the check box of the corresponding columns in the "Show/Hide" dialog.

Creating LAD programs 13

13.1 Basic information on LAD

13.1.1 LAD programming language

Overview of the Ladder Logic (LAD) programming language

LAD is a graphical programming language. The representation is based on circuit diagrams.

The program is mapped in one or more networks. A network contains a power rail on the left where the rungs originate. The binary signal scans are arranged in the form of contacts on the rungs. The serial arrangement of the elements on a rung creates a series connection; arrangement on simultaneous branches creates a parallel connection. Complex functions are represented by boxes.

Example of networks in LAD

The following figure shows a LAD network with two normally open contacts, one normally closed contact and one coil:

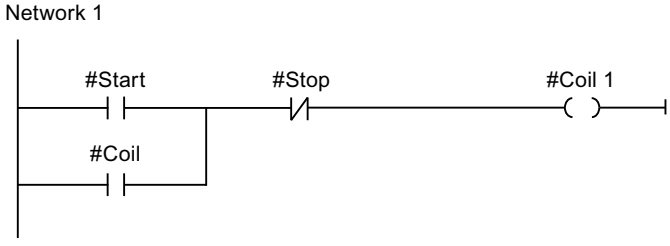

13.1.2 Overview of the LAD elements

LAD elements

A LAD program consists of separate elements that you can arrange in series or parallel on the power rail of a network. Most program elements must be supplied with tags.

There is at least one rung from the power rail. Network programming starts at the left edge of the rung. You can expand the power rail by several rungs and branches.

For example, the following figure shows elements of a LAD network:

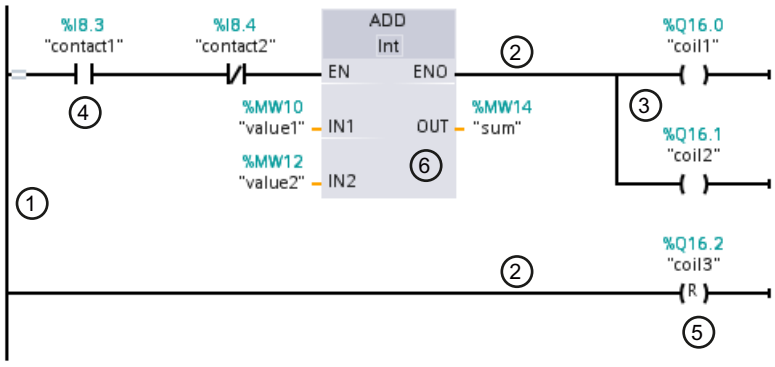

1) Power rail

2) Rung

3) Branch

4) Contact

5) Coil

6) Box

Power rail

Each LAD network consists of a power rail that contains at least one rung. A network can be extended by adding additional rungs. You can use branches to program parallel connections in the specific rungs.

Contacts

You can use contacts to create or interrupt a current-carrying connection between two elements. The current is relayed from left to right. You can use contacts to query the signal state or the value of an operand and control it depending on the result of the current flow.

The following types of contact are available to you in a LAD program:

- Normally open contact:
  Normally open contacts forward the current if the signal state of a specified binary operand is "1".
- Normally closed contacts:
  Normally closed contacts forward the current if the signal state of a specified binary operand is "0".
- Contact with additional function:
  Contacts with additional function forward the current if a specific condition is met. With these contacts you can also execute an additional function, such as an RLO edge detection and a comparison.

Coils

You can use coils to control binary operands. Coils can set or reset a binary operand depending on the signal state of the result of logic operation.

The following types of coils are available to you in a LAD program:

- Standard coils:
  Standard coils set a binary operand if current flows in the coil. The "Assignment" instruction is an example of a standard coil.
- Coils with additional function:
  These coils have additional functions in addition to the evaluation of the logic operation result. Coils for RLO edge detection and program control are examples of coils with additional function.

Boxes

Boxes are LAD elements with complex functions. The empty box is an exception. You can use the empty box as a placeholder in which you can select the required instruction.

The following types of boxes are available to you in a LAD program:

- Boxes without EN/ENO mechanism:
  A box is executed depending on the signal state at the box inputs. The error status of the processing cannot be queried.
- Boxes with EN/ENO mechanism:
  A box is only executed if the enable input "EN" carries the signal state "1". If the box is processed correctly, the "ENO" enable output has signal state "1". If an error occurs during the processing, the "ENO" enable output is reset.

Calls of code block are also shown in the network as boxes with EN/ENO mechanism.

See also

Rules for the use of LAD elements (Page 7134)

EN/ENO mechanism (Page 141)

13.2 Settings for LAD

13.2.1 Overview of the settings for LAD

Overview

The following table shows the settings that you can make:

Group	Setting	Description
Font	Font size	Font size in program editor
View	Layout	Compact or wide Changes the vertical spacing between operands and other objects (such as operand and contact). The change becomes visible once the block is reopened.
Operand field	Maximum width	Maximum number of characters that can be entered horizontally in the operand field. This setting recalculates the layout of the networks.
	Maximum height	Maximum number of characters that can be entered vertically in the operand field. This setting recalculates the layout of the networks.

See also

Changing the settings (Page 7124)

13.2.2 Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for LAD (Page 7124)

13.3 Working with networks

13.3.1 Using networks

Function

The user program is created in the block within networks. For a code block to be programmed, it must contain at least one network. To achieve a better overview of the user program, you can also subdivide your program into several networks.

You have the option to insert networks for the textual programming languages SCL and STL in LAD and FBD blocks, and then use instructions in these programming languages. Depending on the CPU you are using, you can insert networks for the following programming languages:

- S7-300/400: STL networks
- S7-1200: SCL networks
- S7-1500: STL and SCL networks

The programming editor always adapts to the network that currently has the focus. This means that you have access to the SCL instructions as well as the SCL functions when programming an SCL network.

Note

You cannot use the instruction "Goto" in SCL networks within LAD or FBD blocks.

EN/ENO mechanism in blocks with different network languages

You can also use the EN/ENO mechanism in blocks with different network languages. Each programming language displays the error status differently:

- SCL has an "ENO" tag that stores the error status and can be queried. Direct access to this tag is only possible with SCL.
- LAD/FBD/STL have no special tag for "ENO". However, you can read the error status for STL from the BR bit, and query it via the RET coil for LAD/FBD.

The following rules apply to reading out the error status for the entire block:

- The last network in the block is a LAD/FBD network:
  If you do not use RET coils, the error status is "TRUE" by default.
- The last network in the block is an STL network:
  The BR bit determines the error status. The BR bit can be edited in STL networks using the BR tab.
- The last network in the block is an SCL network:
  The "ENO" tag determines the error status of the block.

See also

Inserting networks (Page 7127)
Inserting network title (Page 7131)
Entering a network comment (Page 7132)
Navigating networks (Page 7133)

13.3.2 Inserting networks

You can either insert a network of the same programming language or a network for a textual programming language in an LAD or FBD block. Keep in mind, however, that not all CPU types support all textual languages.

Requirement

An LAD or FBD block is open.

Insert LAD or FBD network

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Insert STL or SCL network

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert STL network" or "Insert SCL network" command in the shortcut menu.

Result

A new, empty network is inserted in the block in the corresponding programming language.

See also

Using networks (Page 7126)
Selecting networks (Page 7128)
Copying and pasting networks (Page 7129)
Deleting networks (Page 7129)
Expanding and collapsing networks (Page 7130)
Inserting network title (Page 7131)

Entering a network comment (Page 7132)
Navigating networks (Page 7133)

13.3.3 Selecting networks

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.
   The first and last networks and all those in between are selected.

See also

Using networks (Page 7126)
Inserting networks (Page 7127)
Copying and pasting networks (Page 7129)
Deleting networks (Page 7129)
Expanding and collapsing networks (Page 7130)
Inserting network title (Page 7131)
Entering a network comment (Page 7132)
Navigating networks (Page 7133)

13.3.4 Copying and pasting networks

Copied networks can be pasted within the block or in another block. Networks that were created in LAD or FBD can also be inserted in blocks of the respective other programming language.

Requirement

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Using networks (Page 7126)
Inserting networks (Page 7127)
Selecting networks (Page 7128)
Deleting networks (Page 7129)
Expanding and collapsing networks (Page 7130)
Inserting network title (Page 7131)
Entering a network comment (Page 7132)
Navigating networks (Page 7133)

13.3.5 Deleting networks

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Using networks (Page 7126)
Inserting networks (Page 7127)
Selecting networks (Page 7128)
Copying and pasting networks (Page 7129)
Expanding and collapsing networks (Page 7130)
Inserting network title (Page 7131)
Entering a network comment (Page 7132)
Navigating networks (Page 7133)

13.3.6 Expanding and collapsing networks

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Using networks (Page 7126)
Inserting networks (Page 7127)
Selecting networks (Page 7128)
Copying and pasting networks (Page 7129)
Deleting networks (Page 7129)
Inserting network title (Page 7131)
Entering a network comment (Page 7132)
Navigating networks (Page 7133)

13.3.7 Inserting network title

The network title is the header of a network. The length of the network title is limited to one line. You can enter the title manually or set it automatically. When you set it automatically, you can do this for individual networks or use the settings to specify that the network title is always set automatically.

For automatic insertion of the network title, the comment of the operand in one of the following instructions in the network is evaluated:

- Assignment
- Set output
- Reset output

The instruction that is listed first in the network is used.

The network title is only inserted automatically when the following conditions are fulfilled:

- The network does not have a title yet.
- The operand of the instruction used for the comment has a comment.

Note

Note the following restrictions for automatic insertion of the network title:

- The network title is not adapted if you change the comment of the operand at a later time.
- The network title is not adapted if you change the operand of the instruction.
- The network title is only set by the writing instructions listed above.
- If the operand is of the data type array, the comment of the array is used and not the comments of the array elements.
- Comments of invalid operands are not taken into consideration.

Entering the network title manually

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

Setting the network title automatically

To specify that the network titles are always set automatically, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Select the "Set network title automatically" check box in the "Additional settings" group.
   The network titles are set automatically as of this time if the conditions listed above are fulfilled.

To set an individual network title automatically, follow these steps:

1. Right-click "Network <Number of the network>" in the title bar of a network.
2. Select the "Set network title automatically" command in the shortcut menu.
   The title of the selected network is set based on the comment of the operand if the conditions listed above are fulfilled.

See also

Using networks (Page 7126)

Inserting networks (Page 7127)

Selecting networks (Page 7128)

Copying and pasting networks (Page 7129)

Deleting networks (Page 7129)

Expanding and collapsing networks (Page 7130)

Entering a network comment (Page 7132)

Navigating networks (Page 7133)

13.3.8 Entering a network comment

You can use network comments to provide comments on the program contents of individual networks. For example, you can indicate the function of the network or draw attention to special characteristics.

Requirement

A network is available.

Procedure

To enter a network comment, follow these steps:

1. Click on the right arrow before the network title.
2. If the comment area is not visible, click "Network comments on/off" in the toolbar.
   The comment area is displayed.
3. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
4. Enter the network comment.

See also

Using networks (Page 7126)
Inserting networks (Page 7127)
Selecting networks (Page 7128)
Copying and pasting networks (Page 7129)
Deleting networks (Page 7129)
Expanding and collapsing networks (Page 7130)
Inserting network title (Page 7131)
Navigating networks (Page 7133)

13.3.9 Navigating networks

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click in the code area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.
   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 7126)
Inserting networks (Page 7127)
Selecting networks (Page 7128)
Copying and pasting networks (Page 7129)
Deleting networks (Page 7129)
Expanding and collapsing networks (Page 7130)
Inserting network title (Page 7131)
Entering a network comment (Page 7132)

13.4 Inserting LAD elements

13.4.1 Rules for the use of LAD elements

Rules

Note the following rules when inserting LAD elements:

- Every LAD network must terminate with a coil or a box. However, the following LAD elements must not be used to terminate a network:
  - Comparator boxes
  - Instructions for positive and negative RLO edge detection
- The starting point of the branch for a box connection must always be the power rail. Logic operations or other boxes can be present in the branch before the box.
- Only contacts can be inserted into simultaneous branches with preceding logic operations. The contact for negating the result of logic operation (-|NOT|-) is an exception here. The contact for negating the result of logic operation, as well as coils and boxes, can be used in simultaneous branches if they originate directly from the power rail.
- Constants (e.g. TRUE or FALSE) cannot be assigned to normally closed or normally open contacts. Instead, use operands of the BOOL data type.
- Only one jump instruction can be inserted in each network.
- Only one jump label can be inserted in each network.
- Instructions with positive or negative edge detection may not be arranged directly at the left margin of the rung as they requires a prior logic operation.

Placement rules for S7-1200/1500 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
SET_BF	Set bit field	No
RESET_BF	Reset bit field	No
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
JMP_LIST	Define jump list	No
SWITCH	Jump distributor	No
RET	Return	No

Placement rules for S7-300/400 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
S	Set output	Yes
R	Reset output	Yes
SP	Start pulse timer	Yes
SE	Start extended pulse timer	Yes
SD	Start on-delay timer	Yes
SS	Start retentive on-delay timer	Yes
SF	Start off-delay timer	Yes
SC	Set counter value	Yes
CU	Count up	Yes
CD	Count down	Yes
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
RET	Return	No
OPN	Open global data block	No
OPNI	Open instance data block	No
CALL	Call block	No
SAVE	Save RLO in BR bit	No
MCRA	Enable MCR range	No
MCRD	Disable MCR range	No
MCR<	Open MCR ranges	No
MCR>	Close MCR ranges	No

Note

Calling blocks with binary inputs in S7-300/400 CPUs

The following FAQ describes the special features when calling blocks with binary inputs:

FAQ 13988019: What should you watch out for when programming nested FB/FC calls with binary inputs in FBD? (Page 7136)

See also

Overview of the LAD elements (Page 7121)

FAQ 13988019 (https://support.industry.siemens.com/cs/de/de/view/13988019)

EN/ENO mechanism (Page 141)

13.4.2 Prohibited interconnections in LAD

Power flow from right to left

No branches can be programmed that could result in a power flow in the reverse direction.

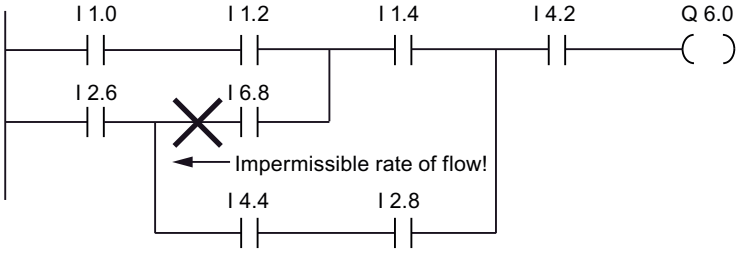

Short-circuit

No branches may be programmed that would cause a short-circuit.

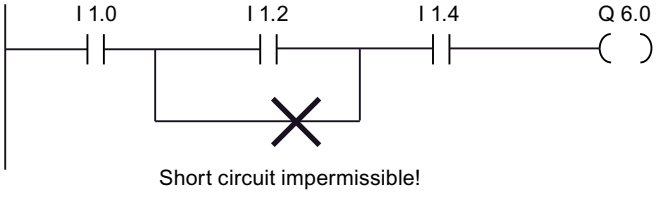

Logic operations

The following rules apply to logic operations:

- Only Boolean inputs can be combined with preceding logic operations.
- Only the first Boolean output can be combined with a further logic operation.
- Only one complete logical path can exist per network. Paths that are not connected can be linked.

See also

Rules for the use of LAD elements (Page 7134)

13.4.3 Inserting LAD elements using the "Instructions" task card

Requirement

A network is available.

Procedure

To insert a LAD element into a network using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to the LAD element that you want to insert.
3. Use drag-and-drop to move the element to the desired place in the network.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog, you can determine if the data of the inserted element is to be stored in a single, multi or parameter instance. If you select "Single instance", you can find the new instance data block once it is created in the "Program resources" folder of the project tree under "Program blocks > System blocks". You can find multi-instances in the block interface in the "Static" section and parameter instances in the "InOut" section.

Or:

1. Select the point in the network at which you want to insert the element.
2. Open the "Instructions" task card.
3. Double-click on the element you want to insert.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog, you can determine if the data of the inserted element is to be stored in a single, multi or parameter instance. If you select "Single instance", you can find the new instance data block once it is created in the "Program resources" folder of the project tree under "Program blocks > System blocks". You can find multi-instances in the block interface in the "Static" section and parameter instances in the "InOut" section.

Result

The selected LAD element is inserted with placeholders for the parameters.

See also

Single instances (Page 63)

Multi-instances (Page 65)

Parameter instances (Page 68)

13.4.4 Inserting LAD elements using an empty box

Requirement

A network is available.

Procedure

To insert an LAD element into a network using an empty box, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Empty box" in the "Basic instructions" palette.
3. Use a drag-and-drop operation to move the "Empty box" element to the desired place in the network.
4. Position the cursor over the triangle in the top right-hand corner of the empty box.
   A drop-down list is displayed.
5. Select the required LAD element from the drop-down list.
   If the element is an internal system function block (FB), the "Call option" dialog opens. In this dialog, you can determine if the data of the inserted element is to be stored in a single, multi or parameter instance. If you select "Single instance", you can find the new instance data block once it is created in the "Program resources" folder of the project tree under "Program blocks > System blocks". You can find multi-instances in the block interface in the "Static" section and parameter instances in the "InOut" section.

Result

The empty box is changed to the respective LAD element. Placeholders are inserted for the parameters.

See also

Single instances (Page 63)

Multi-instances (Page 65)

Parameter instances (Page 68)

13.4.5 Selecting the data type of a LAD element

13.4.5.1 Selecting a data type

Introduction

Some instructions can be executed with several different data types. If you use one of these instructions in the program, you have to specify a valid data type for the instruction at the specific point in the program. For some instructions, you have to select the data types for the inputs and outputs separately.

Note

The valid data type (BOOL) for the tags on the enable input EN and the enable output ENO is predefined by the system and cannot be changed.

The valid data types for an instruction are listed in the instruction drop-down list. You specify the data type of the instruction by selecting an entry from the drop-down list. If the data type of an operand differs from the data type of the instruction and cannot be converted implicitly, the operand is displayed in red and a rollout with the corresponding error message appears.

Data type selection of mathematical instructions

Some mathematical instructions provide you with the option of having the data type automatically set corresponding to the data types of the operand. In the drop-down list for data type selection, these instructions have the entry "Auto" in addition to the actual data types. If you select this entry and then allocate the first operand, the data type of the operand is selected as data type for the instruction. The entry in the drop-down list changes to "Auto (<Data type>)", e.g. "Auto (Real)". If you allocate additional operands, the automatically set data type of the instruction is adjusted according to the following criteria:

- You supply all other operands with tags of the same data type:
  The data type of the instruction is not changed.
- You supply all other operands with tags whose data type is smaller than the data type of the instruction:
  The data type of the instruction is not changed. For the operand with the smaller data type, an implicit conversion is conducted if necessary.
- You supply an additional operand with a tag whose data type is greater than the data type of the instruction:
  The data type of the instruction is changed to the larger data type. An implicit conversion is performed, if necessary, for operands that deviate from the newly set data type of the instruction.

Each change in the data type of an operand can result in a change of the data type of the instruction. Other operands may possibly be implicitly converted as a result. Operands for which an implicit conversion is performed are marked with a gray square.

Note

Please also observe the information on data type conversion for your device and, in particular, the notes on the IEC check.

See also: Data type conversion (Page 489)

See also

Defining the data type of an instruction (Page 7139)

13.4.5.2 Defining the data type of an instruction

Introduction

Some instructions can be executed with several different data types. When you insert such instructions into your program, you must specify the data type for these instructions at the actual point in the program.

Specifying the data type by means of the drop-down list

To define the data type of an instruction using the drop-down list, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Click the triangle in the upper corner of the drop-down list.
   The drop-down list will open to display the data types valid for the instruction.
3. Select a data type from the drop-down list.
   The selected data type is displayed.
4. If the instruction has two drop-down lists, select the data type for the instruction inputs in the left-hand drop-down list and the data type for the instruction outputs in the right-hand drop-down list.

Specifying data type by assigning tags

To define the data type of an instruction by assigning tags, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. At an input or output, specify a valid tag, the data type of which is to be applied as the instruction data type.
   The data type of the tag is displayed in the drop-down list.
3. Enter a valid tag at an input and a valid tag at an output if data types need to be defined for both the inputs and outputs of the instruction. The tag specified at the input determines the data type of the inputs; the tag specified at the output determines the data type of the outputs of the instruction.

Automatically specifying the data type of mathematical instructions

To automatically specify the data type for mathematical instructions, follow these steps:

1. Insert the mathematical instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Select the "Auto" entry from the drop-down list.
3. Enter a valid tag at an input or output.
   The data type of the tag is applied as data type of the instruction. The entry in the drop-down list changes to "Auto (<Data type>)".

See also: Selecting a data type (Page 7138)

See also

Selecting a data type (Page 7138)

13.4.6 Using favorites in LAD

13.4.6.1 Adding LAD elements to Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing LAD elements from Favorites (Page 7142)

Overview of the program editor (Page 6831)

13.4.6.2 Inserting LAD elements using favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing LAD elements from Favorites (Page 7142)

Overview of the program editor (Page 6831)

13.4.6.3 Removing LAD elements from Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding LAD elements to Favorites (Page 7141)

Inserting LAD elements using favorites (Page 7141)

Overview of the program editor (Page 6831)

13.4.7 Insert block calls in LAD

13.4.7.1 Inserting block calls using a drag-and-drop operation

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree. If you call function blocks from other function blocks, you can either call them as a single instance, multi-instance or parameter instance.

See also: Basics of instances (Page 62)

Requirement

- A network is available.
- The block that is to be called is available.

Inserting a call of a function (FC)

To insert a call of a function (FC) into a network using a drag-and-drop operation, follow these steps:

1. Drag the function from the project tree to the required network.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Drag the function block from the project tree to the required network.
   The "Call options" dialog opens.
2. In the dialog, enter whether you want to call the block as a single, multi or parameter instance.
   - If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block.
     If you call a block that contains monitoring, assign a ProDiag function block to the monitoring functions in the "ProDiag FB" text box.
   - If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
   - If you click on the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
3. If you call a block that contains monitoring, assign a ProDiag function block to the monitoring functions.
4. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You can then assign the parameters.

See also: Parameter transfer at block call (Page 76)

Note

If when calling a function block you specify an instance data block that does not exist, it will be created.

See also

Updating block calls in LAD (Page 7144)

Changing the instance type (Page 7146)

Single instances (Page 63)

Multi-instances (Page 65)

Block calls (Page 60)

13.4.7.2 Updating block calls in LAD

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have the following options for updating the block calls:

- Explicit updating of all inconsistent block calls in the program editor.
  The inconsistent block calls within the open block are updated. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are removed if they are not connected.
  - Renamed parameters get the new parameter names.
- Explicit updating of a block call in the program editor.
  The "Interface update" dialog is displayed. In this dialog, you have the option of changing the connection of the operands of the new interface. The inconsistent call of this block is then updated. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are removed if they are not connected.
  - Renamed parameters get the new parameter names.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated. Note that when you call functions (FCs) before the next compilation process, all new formal parameters must be supplied with actual parameters.

Updating all inconsistent block calls in the program editor

To open all block calls in a block, follow these steps:

1. Open the calling block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

Updating a specific block call in the program editor

To update a specific block call in the program editor, follow these steps:

1. Open the calling block in the program editor.
2. Right-click on the block call that you want to update.
3. Select the "Update" command in the shortcut menu.
   The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
4. If necessary, change the connection of the operands. To do this, you have the following options:
   - You can use either a drag-and-drop operation or a cut/copy-and-paste operation to move the operand from the old interface to the new interface.
   - You can delete an operand.
   - You can rename an operand.
   - You can specify a new operand via autocompletion.
5. Click "OK" to update the block call. If you want to cancel the update, click "Cancel".

Note

Note that the "Update block call" command is only available provided you did not previously update all block calls in the editor with the "Update inconsistent block calls" command.

Update block calls during compilation

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Inserting block calls using a drag-and-drop operation (Page 7143)

Changing the instance type (Page 7146)

Block calls (Page 60)

13.4.7.3 Changing the block call

You have the option of changing the called block for a block call. But keep in mind that no new instance data blocks are created, for example, when changing from a function (FC) to a function block (FB).

Procedure

To change the called block of a block call, follow these steps:

1. Click on the name of the called block within the block call and press the <F2> key. Or double-click the name of the called block.
   A text box opens, and the name of the currently called block is selected.
2. Enter the name of the block you want to call or select a block in the autocompletion.
3. If you want to call an FB, create a new instance data block, if necessary, and specify it as operand.

13.4.7.4 Changing the instance type

Instance type

There are two ways of calling function blocks:

- As a single instance
- As a multiple instance
- As parameter instance

See also: Basics of instances (Page 62)

You can modify a defined instance type at any time.

Requirement

The user program contains a block call.

Procedure

To change the instance type of a function block, follow these steps:

1. Open the code block and select the block call.
2. Select the "Change instance" command in the shortcut menu.
   The "Call options" dialog opens.

3. Click on the button "Single instance", "Multi-instance" or "Parameter instance".
   - If you select the "Single instance" instance type, enter a name for the data block that is to be assigned to the function block.
   - If you select "Multiple instance" as the instance type, enter in the "Name in the interface" text field the name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
   - If you select "Parameter instance" as the instance type, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
4. Confirm your entries with "OK".

Note

The previous instances are not deleted automatically.

See also

Inserting block calls using a drag-and-drop operation (Page 7143)

Updating block calls in LAD (Page 7144)

Instances (Page 62)

Block calls (Page 60)

13.4.8 Inserting complex LAD instructions

13.4.8.1 Using the "Calculate" instruction (S7-1200, S7-1500)

Requirement

A network is available.

Procedure

Proceed as follows to use the "Calculate" instruction:

1. Open the "Instructions" task card.
2. Navigate to "Math functions > CALCULATE" in the "Basic instructions" pane.
3. Use drag-and-drop to move the element to the desired place in the network.
   The instruction "Calculate" will be inserted for the data type with a placeholder expression and question mark.
4. Enter the data type for the calculation.

5. Enter the operands for the calculation.

   Note

   The calculation is run with the inputs of the "Calculate" instruction. If you want to use constants you must also insert appropriate inputs for them.

6. Click on the "Edit 'Calculate' instruction" button to replace the placeholder expression with the correct expression.
   The "Edit 'Calculate' instruction" dialog will open.
7. Enter the required expression in the "OUT:= " text box.

   Note

   In the "Example" area you can find an example of a valid expression and possible instructions that you can use.

   To determine a value with the help of Pythagoras' theorem, for example, enter "OUT := SQRT (SQR (IN1) + SQR (IN2))".

8. Confirm your entry with "OK".

See also

CALCULATE: Calculate (Page 726)

13.4.9 Using free-form comments

13.4.9.1 Basic information on using free-form comments in LAD

Introduction

Free-form comments allow you to add comments to the source code for graphic programming languages similar to line comments for textual languages.

Free-form comments can be used for the following elements:

- Boxes
- Coils

Commenting out program code

To deactivate individual instructions or entire program sections in the LAD and FBD programming languages, you must set a bit parallel to the instruction or in series to an instruction block. Commenting out, as in STL and SCL, is not possible in LAD and FBD.

Examples of commenting out program code in all programming languages are available in the Siemens Industry Online under the following FAQ ID: 109482004 (https://support.industry.siemens.com/cs/hu/en/view/109482004)

See also

Inserting free-form comments (Page 7149)
Editing free-form comments (Page 7149)
Deleting free-form comments (Page 7151)

13.4.9.2 Inserting free-form comments

Requirement

A network with instructions is available.

Procedure

To insert a free comment on an instruction, proceed as follows:

1. If necessary, activate the "Free-form comments on/off" button in the toolbar.
2. Right-click on the instruction for which you want to insert a free-form comment.
3. Select the "Insert comment" command in the shortcut menu.
   A comment box with a standard comment opens. The comment box is connected by an arrow to the corresponding instruction.
4. Enter the required comment in the comment box.

See also

Basic information on using free-form comments in LAD (Page 7148)
Editing free-form comments (Page 7149)
Deleting free-form comments (Page 7151)

13.4.9.3 Editing free-form comments

Introduction

Free-form comments can be edited as follows:

- Changing the comment text
- Changing the position and size of the comment box
- Attaching a comment to another element
- Showing and hiding free comments

Changing the comment text

To change the text of free-form comments, follow these steps:

1. Click on the comment box.
2. Enter the desired text.

Changing the position of the comment box

To change the positioning of the comment box, follow the steps below:

1. Left-click the comment box and keep the mouse button pressed.
2. Drag the comment box to the desired location.

Changing the size of the comment box

To change the size of the comment box, follow the steps below:

1. Click on the comment box.
2. Drag the comment box on the move handle in the lower right corner to the desired size.

Attaching a comment to another element

To attach a free-form comment to another element, follow these steps:

1. Left-click the point of the arrow that links the comment box with the instruction and keep the mouse button pressed.
2. Drag the arrow to the element to which you want to attach the comment. Possible insertion points are marked with a green square.
3. Release the mouse button.

Showing and hiding free comments

To show or hide a free-form comments, follow these steps:

1. Click the "Free-form comment on/off" button in the toolbar.

See also

Basic information on using free-form comments in LAD (Page 7148)

Inserting free-form comments (Page 7149)

Deleting free-form comments (Page 7151)

13.4.9.4 Deleting free-form comments

Procedure

To delete a free-form comment, proceed as follows:

1. Right-click on the free-form comment that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on using free-form comments in LAD (Page 7148)
Inserting free-form comments (Page 7149)
Editing free-form comments (Page 7149)

13.5 Editing LAD elements

13.5.1 Selecting LAD elements

You can select several individual elements or all elements in a network.

Requirement

LAD elements are available

Selecting several individual LAD elements

To select several individual LAD elements, follow these steps:

1. Press and hold down the <Ctrl> key.
2. Click on all the LAD elements you wish to select.
3. Now release the <Ctrl> key.

Selecting all LAD elements in a network

To select all LAD elements in a network, follow these steps:

1. Go to the network whose elements you wish to select.
2. Select the "Select all" command in the "Edit" menu or press <Ctrl A>.

See also

Copying LAD elements (Page 7153)
Cutting LAD elements (Page 7153)
Pasting an LAD element from the clipboard (Page 7154)
Replacing LAD elements (Page 7155)
Inserting additional inputs and outputs in LAD elements (Page 7155)
Removing inputs and outputs (Page 7156)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 144)
Deleting LAD elements (Page 7157)

13.5.2 Copying LAD elements

Requirement

An LAD element is available.

Procedure

To copy a LAD element, follow these steps:

1. Right-click the LAD element that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

The LAD element will be copied and saved to the clipboard.

See also

Selecting LAD elements (Page 7152)
Cutting LAD elements (Page 7153)
Pasting an LAD element from the clipboard (Page 7154)
Replacing LAD elements (Page 7155)
Inserting additional inputs and outputs in LAD elements (Page 7155)
Removing inputs and outputs (Page 7156)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 144)
Deleting LAD elements (Page 7157)

13.5.3 Cutting LAD elements

Requirement

An LAD element is available.

Cutting

To cut a LAD element, follow these steps:

1. Right-click the LAD element that you want to cut.
2. Select "Cut" in the shortcut menu.

Result

The LAD element will be cut and saved to the clipboard.

See also

Selecting LAD elements (Page 7152)
Copying LAD elements (Page 7153)
Pasting an LAD element from the clipboard (Page 7154)
Replacing LAD elements (Page 7155)
Inserting additional inputs and outputs in LAD elements (Page 7155)
Removing inputs and outputs (Page 7156)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 144)
Deleting LAD elements (Page 7157)

13.5.4 Pasting an LAD element from the clipboard

Requirement

An LAD element is available.

Procedure

To paste an LAD element from the clipboard, follow these steps:

1. Copy a LAD element or cut a LAD element.
2. Right-click the point in the network where you want to paste the element.
3. Select "Paste" in the shortcut menu.

See also

Selecting LAD elements (Page 7152)
Copying LAD elements (Page 7153)
Cutting LAD elements (Page 7153)
Replacing LAD elements (Page 7155)
Inserting additional inputs and outputs in LAD elements (Page 7155)
Removing inputs and outputs (Page 7156)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 144)
Deleting LAD elements (Page 7157)

13.5.5 Replacing LAD elements

You can easily exchange LAD elements with other LAD elements of the same type. This has the advantage that the parameters are retained and need not be entered again. For example, you can exchange normally open contacts and normally closed contacts or RS FlipFlop and SR FlipFlop.

Requirements

A network with at least one LAD element is present.

Procedure

To replace an LAD element with another LAD element, follow these steps:

1. Select the LAD element that you want to replace.
2. Position the cursor over the triangle in the top right-hand corner of the LAD element.
   A drop-down list is displayed.
3. From the drop-down list, select the LAD element that you want to use to replace the existing LAD element.

See also

Selecting LAD elements (Page 7152)

Copying LAD elements (Page 7153)

Cutting LAD elements (Page 7153)

Pasting an LAD element from the clipboard (Page 7154)

Inserting additional inputs and outputs in LAD elements (Page 7155)

Removing inputs and outputs (Page 7156)

Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 144)

Deleting LAD elements (Page 7157)

13.5.6 Inserting additional inputs and outputs in LAD elements (S7-1200, S7-1500)

Introduction

You can expand LAD elements which execute commutative arithmetic instructions by adding additional inputs. Such elements are, for example, the instructions "Add" (ADD) and "Multiply" (MUL). You can expand the MOVE and DEMUX instruction boxes by adding additional outputs.

Requirement

An LAD element is available that permits the insertion of additional inputs and outputs.

Inserting an additional input

To add an additional input to the box of a LAD element, follow these steps:

1. Right-click on an existing input of the LAD element.
2. Select "Insert input" in the shortcut menu.
   An additional input is added to the box of the LAD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.
   An additional input is added to the box of the LAD element.

Inserting an additional output

To add an additional output to the box of a LAD element, follow these steps:

1. Right-click on an existing output of the LAD element.
2. Select "Insert output" from the shortcut menu.
   An additional output is added to the box of the LAD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.
   An additional output is added to the box of the LAD element.

See also

Inserting LAD elements (Page 7134)

Selecting LAD elements (Page 7152)

Copying LAD elements (Page 7153)

Cutting LAD elements (Page 7153)

Pasting an LAD element from the clipboard (Page 7154)

Replacing LAD elements (Page 7155)

Removing inputs and outputs (Page 7156)

Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 144)

Deleting LAD elements (Page 7157)

13.5.7 Removing inputs and outputs (S7-1200, S7-1500)

Introduction

Inputs and outputs which you have added to an instruction can be removed.

Requirement

An LAD element is available to which you have added additional inputs and outputs.

Remove input

To remove an input, follow these steps:

1. Select the input that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The input of the LAD element is removed.

Remove output

To remove an output, follow these steps:

1. Select the output that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The output of the LAD element will be removed.

See also

Inserting LAD elements (Page 7134)

Selecting LAD elements (Page 7152)

Copying LAD elements (Page 7153)

Cutting LAD elements (Page 7153)

Pasting an LAD element from the clipboard (Page 7154)

Replacing LAD elements (Page 7155)

Inserting additional inputs and outputs in LAD elements (Page 7155)

Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 144)

Deleting LAD elements (Page 7157)

13.5.8 Deleting LAD elements

Requirement

An LAD element is available.

Procedure

To delete a LAD element, follow these steps:

1. Right-click the LAD element that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting LAD elements (Page 7134)
Selecting LAD elements (Page 7152)
Copying LAD elements (Page 7153)
Cutting LAD elements (Page 7153)
Pasting an LAD element from the clipboard (Page 7154)
Replacing LAD elements (Page 7155)
Inserting additional inputs and outputs in LAD elements (Page 7155)
Removing inputs and outputs (Page 7156)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 144)

13.6 Inserting operands into LAD instructions

13.6.1 Inserting operands

The character strings "<???>", "<??.?>" and "..." are inserted as placeholders for the parameters when an LAD element is inserted. The "<???>" and "<??.?>" strings displayed in red indicate parameters that need to be connected. The "..." string displayed in black indicates parameters that may be connected. "<??.?>" stands for Boolean placeholders. In addition, LAD elements can have the following different I/Os for the parameters:

- Orange I/Os: Only tags or constants can be interconnected at this I/O.
- Black I/Os: Boolean input or output to which other elements can be connected upstream or downstream.

Note

If you position the cursor over the placeholder, the expected data type will be displayed.

Requirement

An LAD element is available.

Procedure

To connect the parameters of a LAD element, follow these steps:

1. Double-click the placeholder of the parameter.
   An entry field opens, and the placeholder is selected.
2. Enter the appropriate parameter.

   Note

   If you enter the absolute address of a parameter that has already been defined, this absolute address will be changed to the symbolic name of the parameter as soon as the input is confirmed. If you have not yet defined the parameter, a new tag with this absolute address and the default name "Tag_<n>" will be entered in the PLC tag table. When you confirm your input, the absolute address will be replaced with the symbolic name "Tag_<n>".

3. Confirm the parameter with the Enter key.
4. If you have not yet defined the parameter, you can define it directly in the program editor using the shortcut menu.
   See also:
   Declaring PLC tags in the program editor (Page 7080)
   Declaring local tags in the program editor (Page 6977)

Or drag from it the PLC tag table:

1. In the project tree, select the "PLC tags" folder or open the PLC tag table.
2. If you have opened the PLC tag table, drag the symbol from the first column of the desired tag to the appropriate place in your program. If you have not opened the PLC tag table yet, open the detail view now. Drag the desired tag from the detail view to the appropriate place in your program.

Or drag from it the block interface:

1. Open the block interface.
2. Drag the required operand from the block interface to the instruction window.

Result

- If the syntax is error-free, the displayed parameter is black. The editor then jumps to the next placeholder.
- If there is an error in the syntax, the cursor stays in the entry field and a corresponding error message is displayed in the status line. If you press the Enter key again, the entry field is closed and the faulty entry is displayed in red italics.

See also

Using and addressing operands (Page 89)

Using autocompletion (Page 6859)

13.6.2 Wiring hidden parameters

Introduction

If an instruction contains hidden parameters, the instruction box has a small arrow on the lower edge. You can recognize hidden parameters by their white font.

You can show and wire the hidden parameters at any time.

Showing or hiding hidden parameters

To show or hide hidden parameters, follow these steps:

1. Click on the down arrow at the bottom edge of the instruction box to show hidden parameters.
2. Click on the up arrow at the bottom edge of the instruction box to hide hidden parameters.

Wiring hidden parameters

To wire parameters, follow these steps:

1. Wire the hidden parameters like normally visible parameters.
   Its visibility may change due to the interconnection of the parameter:
   - The parameter is visible if you have selected the "Hide, if no parameter is assigned" option in the properties.
   - The parameter remains hidden if you have selected the "Hide" option in the properties.

See also

Hiding parameters during the block call (Page 6986)

13.6.3 Displaying or hiding tag information

Introduction

You can display the following information about the tags to be used in the Program editor:

- Name of the tag
- Tag addresses
- Simple or hierarchical comments to document the tags

The information is taken from the block interface for local tags and DB tags and from the PLC tag table for tags that are valid CPU-wide.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all the blocks currently opened and opened in future is shown. You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual blocks that are open.

When you select the display of tag information with hierarchical comments, the comments of the higher structure levels are also displayed for structured tags. The comments are shown in brackets after the tag comment; the comments of the individual levels are separated by a dot. If there is no comment for a tag on a structure level, it is not displayed. This is indicated by two successive dots.

You can display the tag information either collectively below a LAD/FBD network or directly at the respective operand. As with showing and hiding tag information, you can set this via the settings for all block or change specifically for an opened block.

Displaying or hiding tag information for all the blocks

Proceed as follows to display or hide the tag information for all the blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.

3. If you want to display the tag information, select the "Show" option or the "Tag information with hierarchical comments" option in the drop-down list, depending on whether you want to display simple or hierarchical comments.
4. Use the "Position of the tag information (LAD/FBD)" drop-down list to specify whether the tag information is to be displayed below the LAD/FBD network or on the operand.
   For all open blocks, the tag information is shown and displayed at the selected position. If you open additional blocks the tag information for these blocks is also shown.
5. If you want to hide the tag information, select the option "Hide" in the "Tag information" drop-down list.
   The tag information is hidden for all open blocks. If you open additional blocks the tag information for these blocks is also hidden.

Displaying or hiding tag information for an opened block

Proceed as follows to display or hide the tag information for an opened block:

1. If you want to display the tag information, select the "Show tag information" option or the "Tag information with hierarchical comments" option in the "Shows the tag information" drop-down list, depending on whether you want to display simple or hierarchical comments.
2. To define the position of the tag information, click the "Position of the tag information (LAD/FBD)" button in the program editor toolbar. The position changes with each click on the button. Alternatively, you can click the small arrow beside the "Position of the tag information (LAD/FBD)" button to open the drop-down list. Than select the position from the drop-down list.
   The tag information is shown and displayed at the selected position.
3. If you want to hide the tag information, select the option "Hide tag information" in the "Shows the tag information" drop-down list.
   The tag information is hidden.

13.7 Branches in LAD

13.7.1 Basic information on branches in LAD

Definition

You use branches to program parallel circuits with the Ladder Logic (LAD) programming language. Branches are inserted in the main rung. You can insert several contacts into the branch and thus achieve a parallel circuit of series connections. This allows you to program complex ladder logic.

The figure below shows an example of the use of branches:

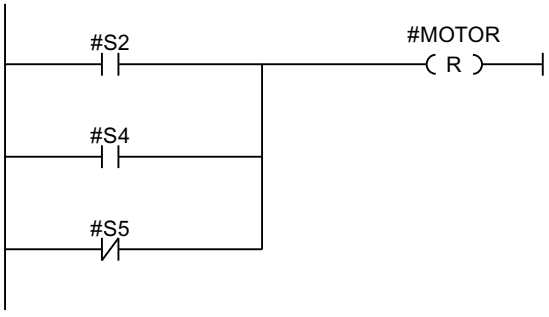

MOTOR carries signal 1, if one of the following conditions is fulfilled:

- Signal 1 is pending on S2 or S4
- Signal 0 is pending on S5.

See also

Rules for branches in LAD (Page 7163)

Inserting branches into the LAD network (Page 7164)

Closing branches in the LAD network (Page 7165)

Deleting branches in LAD networks (Page 7165)

13.7.2 Rules for branches in LAD

Rules

The following rules apply to simultaneous branches:

- A simultaneous branch can only be inserted if the main branch already contains an LAD element.
- Simultaneous branches are opened downwards or are connected directly to the power rail. They are terminated upwards.

- Simultaneous branches are opened after the selected LAD element.
- Simultaneous branches are terminated after the selected LAD element.
- To delete a simultaneous branch, you must delete all LAD elements of this branch. When the last LAD element is removed from the branch, the rest of the branch is also removed.
- You can only insert a coil in a parallel connection when the parallel connection starts directly at the power rail.

See also

Basic information on branches in LAD (Page 7163)

Inserting branches into the LAD network (Page 7164)

Deleting branches in LAD networks (Page 7165)

Closing branches in the LAD network (Page 7165)

13.7.3 Inserting branches into the LAD network

You can create several branches in a network.

Requirement

- A network is available.
- The network contains elements.

Procedure

To insert a new branch in a network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Open branches" in the "Simple instructions" palette.
3. Use drag-and-drop to move the element to the desired place in the network.
   If you want to connect the new branch directly to the power rail, drag the element to the power rail.

See also

Basic information on branches in LAD (Page 7163)

Rules for branches in LAD (Page 7163)

Deleting branches in LAD networks (Page 7165)

13.7.4 Closing branches in the LAD network

Branches must be closed again at suitable places. If necessary, branches will be arranged so that they do not cross each other.

Requirement

A branch is available.

Procedure

To close an open branch, follow these steps:

1. Select the open branch.
2. Press and hold down the left mouse button.
   A dashed line will appear as soon as the cursor is moved.
3. Drag the dashed line to a suitable place on the network. Permissible connections are indicated by green lines.
4. Release the left mouse button.

See also

Basic information on branches in LAD (Page 7163)

Rules for branches in LAD (Page 7163)

13.7.5 Deleting branches in LAD networks

Requirement

A branch is available.

Procedure

To delete a branch, follow these steps:

1. Select the connection line that links the branch to the main branch.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on branches in LAD (Page 7163)

Rules for branches in LAD (Page 7163)

Inserting branches into the LAD network (Page 7164)

13.8 Crossings in LAD

13.8.1 Basic information on crossings in LAD

Definition

A crossing is a place in a LAD network where one branch is closed and at the same time another branch is opened.

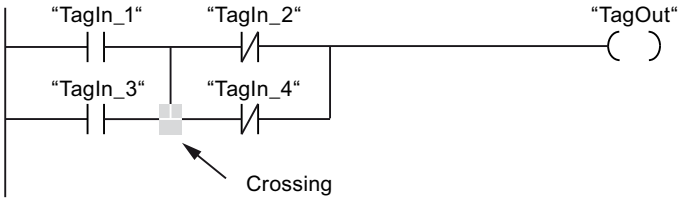

"TagOut" receives signal 1, if the following two conditions are met:

- "TagIn_1" or "TagIn_3" has signal 1
- "TagIn_2" or "TagIn_4" has signal 0

13.8.2 Inserting crossings

You can insert crossings in a LAD network by creating connections between the main branch and an additional branch or between different branches.

Requirements

A branch is available.

Procedure

To insert a new crossing in an LAD network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Open branches" in the "Simple instructions" palette.
3. Drag the element behind the existing branch.
4. Insert any element into the open branch.
5. Click the arrow of the open branch after the inserted element.
6. Hold down the left mouse button and drag the dashed connecting line to the main branch.
7. Release the left mouse button.

See also

Rearranging crossings (Page 7167)
Deleting crossings (Page 7167)
Inserting branches into the LAD network (Page 7164)

13.8.3 Rearranging crossings

Requirement

A crossing is available.

Procedure

To rearrange a connection, follow these steps:

1. Select the connection line that defines the crossings in the respective branches.
2. Select the "Delete" command in the shortcut menu.
3. Open the "Instructions" task card.
4. Navigate to "General > Open branches" in the "Simple instructions" palette.
5. Use a drag-and-drop operation to move the element to the place in the network where you want to insert the new crossing.
6. Click on the arrow for the open branch.
7. Hold down the left mouse button and drag the dashed connecting line to the subsidiary branch in which you wish to insert the new crossing.
8. Release the left mouse button.

See also

Inserting crossings (Page 7166)
Deleting crossings (Page 7167)

13.8.4 Deleting crossings

Requirement

A crossing is available.

Procedure

To delete a crossing, follow these steps:

1. Select the connection line that defines the crossings in the respective branches.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting crossings (Page 7166)
Rearranging crossings (Page 7167)

13.9 Rungs in LAD

13.9.1 Basic information on rungs in LAD

Using rungs

The program is mapped in one or more networks. A network contains a power rail on the left where one or more rungs originate. The binary signal scans are arranged in the form of contacts on the rungs. The serial arrangement of the elements on a rung creates a series connection; arrangement on simultaneous branches creates a parallel connection. A rung is closed by a coil or a box in which the result of logic operation will be written.

The figure below shows an example of the use of several rungs within a network:

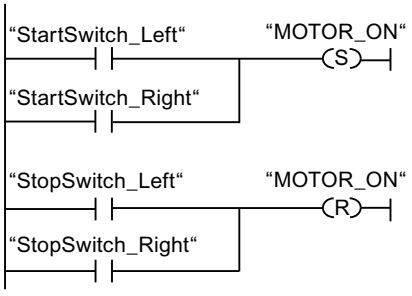

Rules

Remember the following rules when using several rungs:

- Connections are not permitted between rungs.
- Only one jump instruction is permissible per network. The positioning rules for jump instructions remain valid.

Running rungs

Rungs and networks are executed from top to bottom and from left to right. This means that the first instruction in the first rung of the first network is processed first. All instructions of this rung are then processed. After this come all other rungs of the first network. The next network is processed only after all rungs have first been run.

Differences between branches and rungs

The difference between branches and rungs is that the rungs are independent branches that can also stand in a different network. Branches, on the other hand, permit the programming of a parallel connection.

See also

Insert rung (Page 7170)

Deleting a rung (Page 7170)

13.9.2 Insert rung

Requirement

- A block is open.
- A network is available.

Procedure

To insert a new rung in a network, proceed as follows:

1. Insert any coil on the power rail.
   A new rung will be inserted and the coil positioned at the end of the rung.
2. Insert additional instructions in the new rung.

See also

Basic information on rungs in LAD (Page 7169)

Deleting a rung (Page 7170)

13.9.3 Deleting a rung

Requirement

A rung is available.

Procedure

To delete a rung, proceed as follows:

1. Hold down the left mouse button and draw a frame around the rung. At the same time, make sure that you select all instructions. Alternatively, you can hold down the <Shift> key and select the first the last instruction of the rung.
2. Right-click on one of the instructions in the rung.
3. Select the "Delete" command in the shortcut menu.

See also

Basic information on rungs in LAD (Page 7169)
Insert rung (Page 7170)

13.10 LAD programming examples

13.10.1 Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbuttons at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbuttons at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.

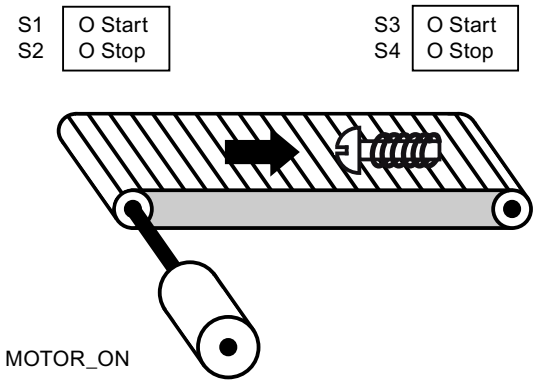

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
StartPushbutton_Left (S1)	Input	BOOL	Start pushbutton on the left side of the conveyor belt
StopPushbutton_Left (S2)	Input	BOOL	Stop pushbutton on the left side of the conveyor belt
StartPushbutton_Right (S3)	Input	BOOL	Start pushbutton on the right side of the convey-or belt
StopPushbutton_Right (S4)	Input	BOOL	Stop pushbutton on the right side of the convey-or belt
MOTOR_ON	Output	BOOL	Turn on the conveyor belt motor

The following networks show the LAD programming for solving this task:

Network 1:

The conveyor belt motor is switched on when start pushbutton "S1" or "S3" is pressed.

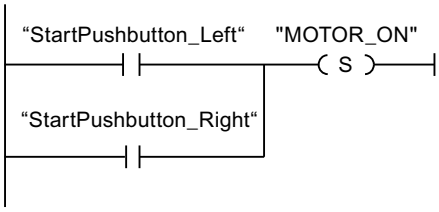

Network 2:

The conveyor belt motor is switched off when stop pushbutton "S2" or "S4" is pressed.

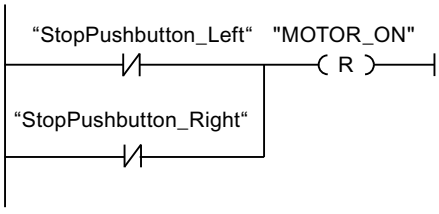

See also

Basic information on LAD (Page 7121)
Settings for LAD (Page 7124)
Working with networks (Page 7126)
Inserting LAD elements (Page 7134)
Editing LAD elements (Page 7152)
Inserting operands into LAD instructions (Page 7159)
Branches in LAD (Page 7163)
Crossings in LAD (Page 7166)
Rungs in LAD (Page 7169)
Example of detecting the direction of a conveyor belt (Page 7174)
Example of detecting the fill level of a storage area (Page 7176)
Example of calculating an equation (Page 7179)
Example of controlling room temperature (Page 7181)

13.10.2 Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The detected running direction of the belt is indicated by a RIGHT arrow or a LEFT arrow. If additional conveyed material is approaching PEB1 from the right or PEB2 from the left, the displayed arrow is initially switched off until, after both photoelectric barriers are passed, the running direction can be detected again and the corresponding arrow displayed. For the solution of the task, 2 edge memory bits are needed that detect the signal change from "0" to "1" at the two photoelectric barriers.

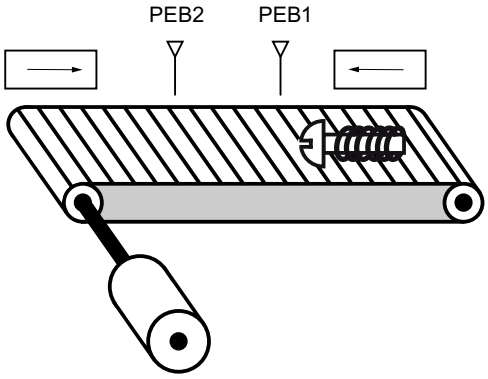

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RIGHT	Output	BOOL	Display during movement to right
LEFT	Output	BOOL	Display during movement to left
CM1	Input	BOOL	Edge bit memory 1
CM2	Input	BOOL	Edge bit memory 2

The following networks show the LAD programming for solving this task:

Network 1:

If the signal state changes from "0" to "1" (positive edge) at photoelectric barrier "PEB1" and, at the same time, the signal state at "PEB2" is "0", the object on the belt is moving to the left.

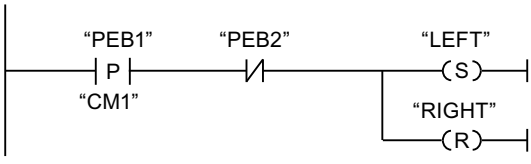

Network 2:

If the signal changes from "0" to "1" (positive edge) at photoelectric barrier "PEB2" and, at the same time, the signal state at "PEB1" is "0", the object on the belt is moving to the right.

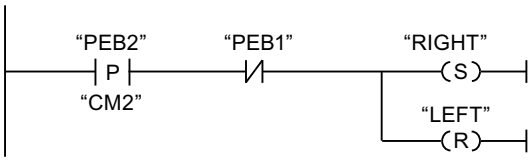

See also

Basic information on LAD (Page 7121)

Settings for LAD (Page 7124)

Working with networks (Page 7126)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

Branches in LAD (Page 7163)

Crossings in LAD (Page 7166)

Rungs in LAD (Page 7169)

Example of controlling a conveyor belt (Page 7172)

Example of detecting the fill level of a storage area (Page 7176)

Example of calculating an equation (Page 7179)

Example of controlling room temperature (Page 7181)

13.10.3 Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock where they are loaded onto trucks. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

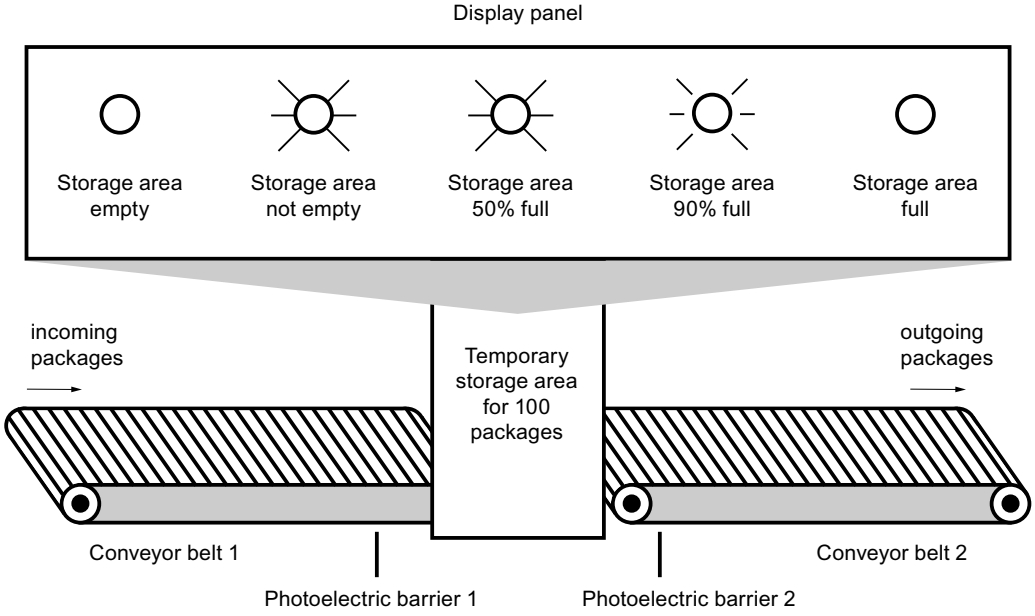

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RESET	Input	BOOL	Reset counter
LOAD	Input	BOOL	Adjust the current counter value to the value of the PV parameter.
MAX STORAGE AREA FILL AMOUNT	Input	INT	Maximum possible number of packages in the storage area
PACKAGECOUNT	Output	INT	Number of packages in the storage area (current count value)

Name	Declaration	Data type	Description
STOCK_PACKAGES	Output	BOOL	Is set when the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag.
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50 % full
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90 % full
STOR_FULL	Output	BOOL	Display lamp: Storage area full
VOLUME_50	Input	INT	Comparison value: 50 packages
VOLUME_90	Input	INT	Comparison value: 90 packages
VOLUME_100	Input	INT	Comparison value: 100 packages

The following networks show the LAD programming for activating the lamps:

Network 1:

When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "MAX STORAGE AREA FILL AMOUNT" tag. As long as the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".

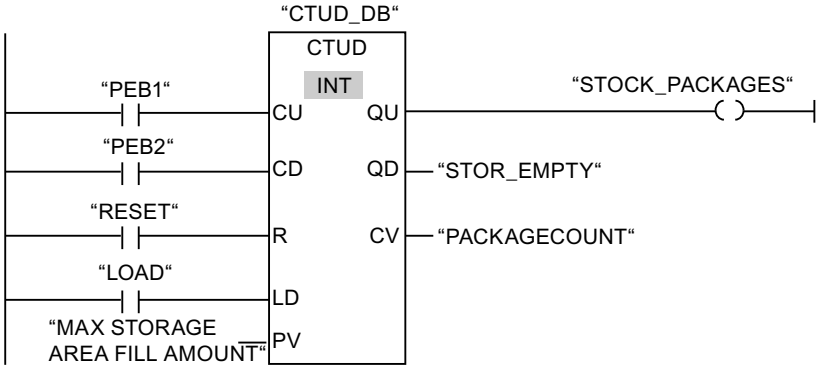

Network 2:

As long as there are packages in the storage area, the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.

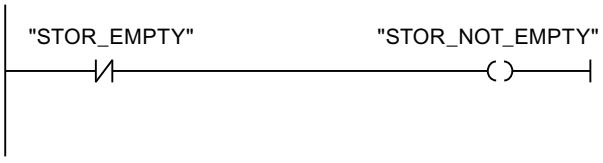

Network 3:

If the number of packages in the storage area is greater than or equal to 50, the "Storage area 50 % full" lamp switches on.

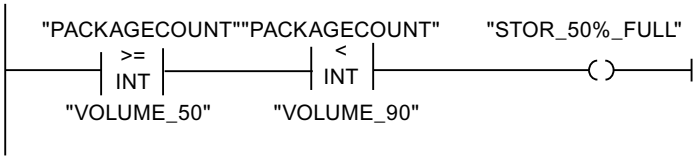

Network 4:

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.

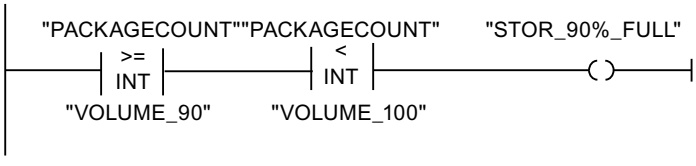

Network 5:

If the number of packages in the storage area reaches 100, the lamp for the "Storage area full" message switches on.

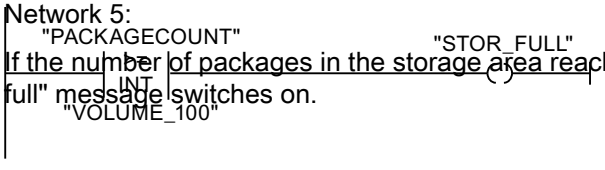

See also

Basic information on LAD (Page 7121)
Settings for LAD (Page 7124)
Working with networks (Page 7126)
Inserting LAD elements (Page 7134)
Editing LAD elements (Page 7152)
Inserting operands into LAD instructions (Page 7159)
Branches in LAD (Page 7163)
Crossings in LAD (Page 7166)
Rungs in LAD (Page 7169)
Example of controlling a conveyor belt (Page 7172)
Example of detecting the direction of a conveyor belt (Page 7174)
Example of calculating an equation (Page 7179)
Example of controlling room temperature (Page 7181)

13.10.4 Example of calculating an equation

Calculating an equation

The following program example shows you how to solve equations of the following type with the LAD programming language:

X= ((A + B) x C) / D

You can always solve this equation with mathematical operations. To do so, you can use the instructions "ADD", "MUL" and "DIV". If you are using a CPU of the S7-1200 or S7-1500 series, you can also use the instruction "CALCULATE". Depending on the data type you choose, different mathematical operations are available for this instruction which can be combined with one another.

Implementation with the instructions "ADD", "MUL" and "DIV"

You want to calculate the following equation:

RESULT = ((A + B) x 15) / E

The following table shows the definition of the tags used:

Name	Data type	Comment
A	INT	First value for addition
B	INT	Second value for addition
C	INT	First intermediate result
15	INT	Multiplier

Name	Data type	Comment
D	INT	Second intermediate result
E	INT	Divisor
RESULT	INT	End result

The following network shows the LAD programming for calculating the equation:

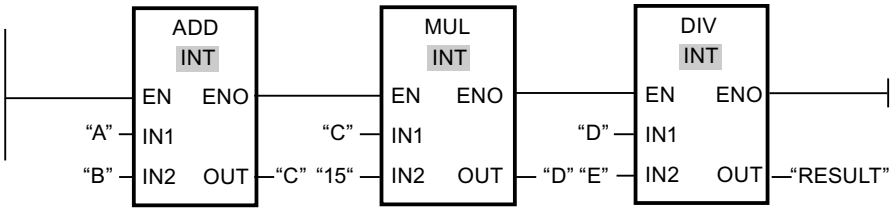

The value of operand "A" is added to the value of operand "B". The sum is stored in operand "C". The value of operand "C" is multiplied by "15". The multiplication result is stored in operand "D". The value stored in operand "D" is then divided by the value of operand "E". The end result is saved in the "RESULT" operand.

Implementation with the instruction "CALCULATE" (S7-1200/1500 only)

You want to calculate the following equation:

RESULT = ((5 + 10) x 4) / 6

The following table shows the definition of the tags used:

Name	Declaration	Data type	Comment	Value
A	Input	INT	First value for addition	5
B	Input	INT	Second value for addition	10
C	Input	INT	Multiplier	4
D	Input	INT	Divisor	6
RESULT	Output	INT	End result	10

To program the equation with the instruction "CALCULATE", follow these steps:

1. Drag the instruction "CALCULATE" from the "Instructions" task card to an LAD network.
2. You can select the data type INT for the instruction from the "<???>" drop-down list.
3. Interconnect the tags declared in the block interface with the inputs and outputs of the instruction box.
4. Click the "Calculator" icon on the top right corner of the instruction box to enter the equation to be calculated.
   The dialog "Edit 'Calculate' instruction" opens.
5. Enter the following expression in the "OUT:=" field:
   ((IN1 + IN2) * IN4) / IN3
   The equation is displayed in the instruction box.

The following network shows the result in the LAD programming language:

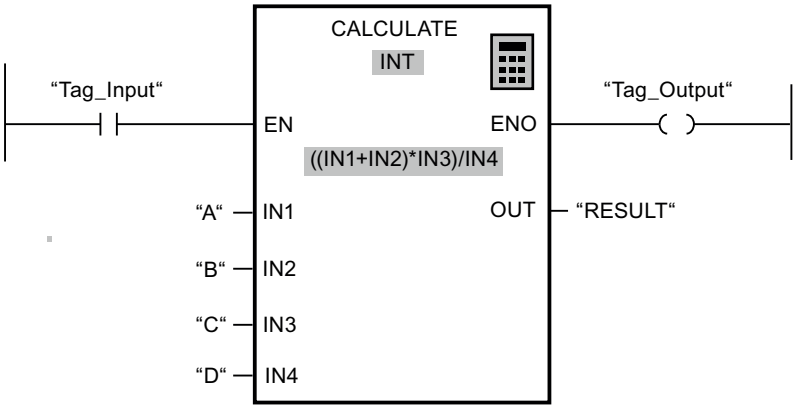

If the "Tag_Input" has signal state "1", the instruction is executed. The value of operand "A" is added to the value of operand "B". The subtotal is multiplied by "C" and then divided by the value of the "D" operand. The end result is saved in the "RESULT" operand.

See also

Basic information on LAD (Page 7121)

Settings for LAD (Page 7124)

Working with networks (Page 7126)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

Branches in LAD (Page 7163)

Crossings in LAD (Page 7166)

Rungs in LAD (Page 7169)

Example of controlling a conveyor belt (Page 7172)

Example of detecting the direction of a conveyor belt (Page 7174)

Example of detecting the fill level of a storage area (Page 7176)

Example of controlling room temperature (Page 7181)

13.10.5 Example of controlling room temperature

Controlling room temperature

In a cold room, the temperature must be maintained below zero degrees Celsius. A sensor is used to check for any temperature fluctuations. If the temperature rises above zero degrees Celsius, the cooling system switches on for a preset time. The "Cooling system on" lamp is lit during this time.

The cooling system and the lamp are turned off if one of the following conditions is met:

- The sensor reports a temperature fall below zero degrees Celsius.
- The preset cooling time has elapsed.
- The "STOP" pushbutton is pressed.

If the preset cooling time has expired and the temperature in the cold room is still too high, the cooling system can be restarted with the "RESET" pushbutton.

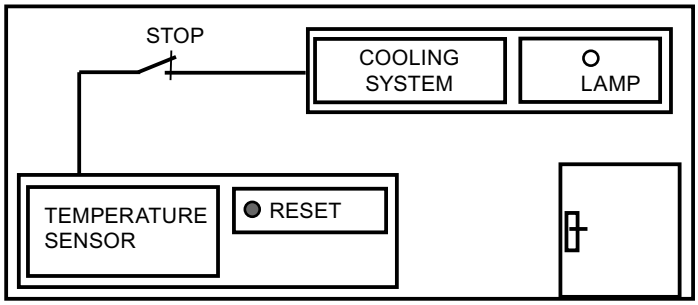

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Comment
Sensor	Input	BOOL	Temperature sensor signal
RESET	Input	BOOL	Restart
STOP	Input	BOOL	The cooling system is switched off.
MaxCoolTime	-	TIME	Preset cooling time This tag is defined in the "DB_Cool" data block.
CurrCoolTime	-	TIME	Currently elapsed cooling time This tag is defined in the "DB_Cool" data block.
Cooling system	Output	BOOL	The cooling system is switched on.
Lamp	Output	BOOL	The lamp for the "Cooling system on" message is switched on.
TempVariable	Temp	BOOL	Temporary tag This tag stores the signal state of the IEC timer TP.

The following network shows the LAD programming for controlling room temperature:

Network 1:

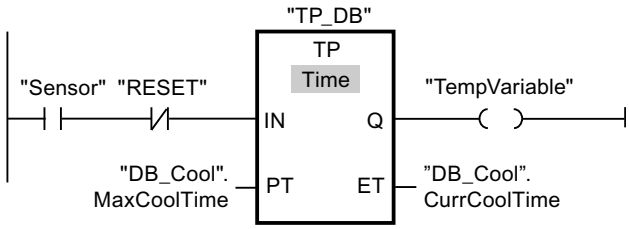

Network 2:

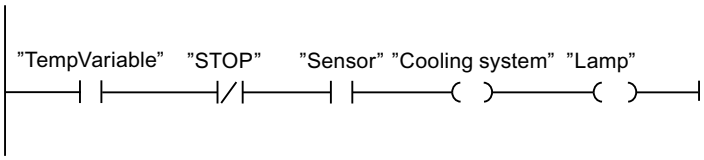

When the temperature in the cold room rises above zero degrees Celsius, the signal state at the "Sensor" operand switches from "0" to "1" (positive signal edge). In the case of a positive signal edge at the IN input of the timer function, the preset cooling time is started and the "TempVariable" receives the signal state "1". The signal state "1" of the "TempVariable" causes the cooling system as well as the display lamp to be turned on in network 2. The "Sensor", "Cooling system" and "Lamp" outputs must be programmed in network 2, because program only one coil can be programmed at the Q output of the timer function.

If the temperature in the cold room falls below zero degrees Celsius, the signal state of the sensor switches back to "0". This switches the cooling system and lamp off.

If the sensor does not signal a temperature drop, the cooling system and lamp are switched off after the preset cooling time has elapsed, at the latest. In this case, the cooling process can be restarted with the "RESET" pushbutton. Pressing and releasing the pushbutton generates a new positive signal edge at the IN input that restarts the cooling system.

The cooling system and the display lamp can be turned off with the "STOP" pushbutton at any time.

See also

Basic information on LAD (Page 7121)

Settings for LAD (Page 7124)

Working with networks (Page 7126)

Inserting LAD elements (Page 7134)

Editing LAD elements (Page 7152)

Inserting operands into LAD instructions (Page 7159)

Branches in LAD (Page 7163)

Crossings in LAD (Page 7166)

Rungs in LAD (Page 7169)

Example of controlling a conveyor belt (Page 7172)

Example of detecting the direction of a conveyor belt (Page 7174)

Example of detecting the fill level of a storage area (Page 7176)
Example of calculating an equation (Page 7179)

Creating FBD programs 14

14.1 Basic information on FBD

14.1.1 FBD programming language

Overview of the Function Block Diagram (FBD) programming language

FBD is a graphical programming language. The representation is based on electronic circuit systems.

The program is mapped in one or more networks. A network contains one or more logic operation paths. The binary signal scans are linked by boxes. The representation of the logic is based on the graphical logic symbols used in Boolean algebra.

Example of networks in FBD

The following figure shows an FBD network with AND and OR boxes and an assignment:

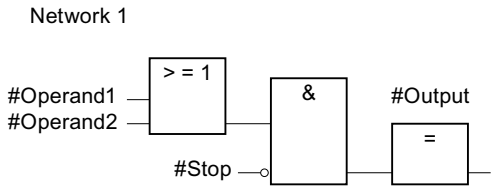

See also

Working with networks (Page 7190)

14.1.2 Overview of the FBD elements

FBD elements

An FBD program consists of separate elements that are linked by means of a binary signal flow. Most program elements must be supplied with tags.

A FBD network is programmed from left to right.

For example, the following figure shows elements of an FBD network:

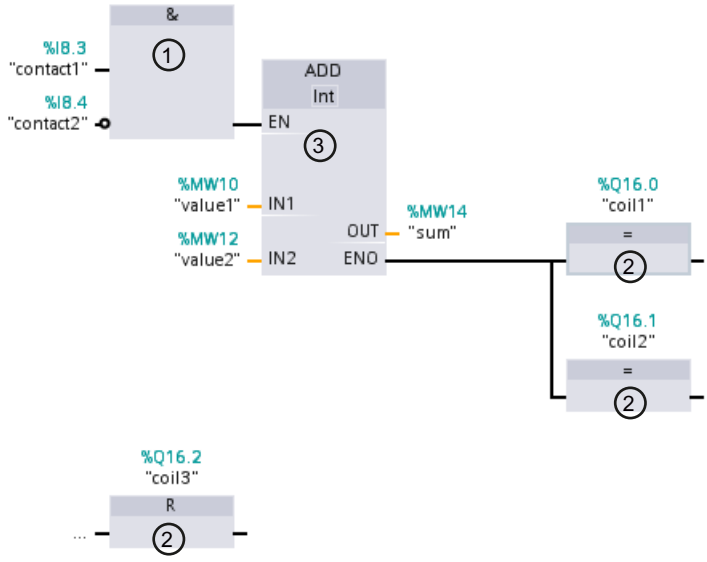

1) Binary function

2) Standard box

3) Complex box

Binary functions

You can use binary functions to query binary operands and to combine their signal states. The following operations are examples of binary functions: "AND operation", "OR operation" and "EXCLUSIVE OR operation".

Standard boxes:

You can use standard boxes to control binary operands, perform RLO edge detection or execute jump functions in the program. Standard boxes generally have only one single input.

Complex boxes

Complex boxes represent program elements with complex functions. The empty box is an exception. You can use the empty box as a placeholder in which you can select the required instruction.

The following types of boxes are available to you in an FBD program:

- Complex boxes without EN/ENO mechanism:
  A box is executed independently of the signal state at the box inputs. The error status of the processing cannot be queried.
- Complex boxes with EN/ENO mechanism:
  A box is only executed if the enable input "EN" has the signal state "1". If the box is processed correctly, the "ENO" enable output has signal state "1". If an error occurs during processing, the "ENO" output is reset.
  If the EN enable input is not interconnected, the box is always executed.

Calls of code block are also shown in the network as complex boxes with EN/ENO mechanism.

See also

EN/ENO mechanism (Page 141)

14.2 Settings for FBD

14.2.1 Overview of the settings for FBD

Overview

The following table shows the settings that you can make:

Group	Setting	Description
Font	Font size	Font size in program editor
View	Layout	Compact or wide Changes the vertical spacing between operands and other objects (such as operand and contact). The change becomes visible once the block is reopened.
Operand field	Maximum width	Maximum number of characters that can be entered horizontally in the operand field. This setting recalculates the layout of the networks.
	Maximum height	Maximum number of characters that can be entered vertically in the operand field. This setting recalculates the layout of the networks.

See also

Changing the settings (Page 7188)

14.2.2 Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for FBD (Page 7188)

14.3 Working with networks

14.3.1 Using networks

Function

The user program is created in the block within networks. For a code block to be programmed, it must contain at least one network. To achieve a better overview of the user program, you can also subdivide your program into several networks.

You have the option to insert networks for the textual programming languages SCL and STL in LAD and FBD blocks, and then use instructions in these programming languages. Depending on the CPU you are using, you can insert networks for the following programming languages:

- S7-300/400: STL networks
- S7-1200: SCL networks
- S7-1500: STL and SCL networks

The programming editor always adapts to the network that currently has the focus. This means that you have access to the SCL instructions as well as the SCL functions when programming an SCL network.

Note

You cannot use the instruction "Goto" in SCL networks within LAD or FBD blocks.

EN/ENO mechanism in blocks with different network languages

You can also use the EN/ENO mechanism in blocks with different network languages. Each programming language displays the error status differently:

- SCL has an "ENO" tag that stores the error status and can be queried. Direct access to this tag is only possible with SCL.
- LAD/FBD/STL have no special tag for "ENO". However, you can read the error status for STL from the BR bit, and query it via the RET coil for LAD/FBD.

The following rules apply to reading out the error status for the entire block:

- The last network in the block is a LAD/FBD network:
  If you do not use RET coils, the error status is "TRUE" by default.
- The last network in the block is an STL network:
  The BR bit determines the error status. The BR bit can be edited in STL networks using the BR tab.
- The last network in the block is an SCL network:
  The "ENO" tag determines the error status of the block.

See also

Inserting network title (Page 7194)
Entering a network comment (Page 7196)
Navigating networks (Page 7197)

14.3.2 Inserting networks

You can either insert a network of the same programming language or a network for a textual programming language in an LAD or FBD block. Keep in mind, however, that not all CPU types support all textual languages.

Requirement

An LAD or FBD block is open.

Insert LAD or FBD network

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Insert STL or SCL network

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert STL network" or "Insert SCL network" command in the shortcut menu.

Result

A new, empty network is inserted in the block in the corresponding programming language.

See also

Inserting network title (Page 7194)
Entering a network comment (Page 7196)
Navigating networks (Page 7197)

14.3.3 Selecting networks

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.
   The first and last networks and all those in between are selected.

See also

Using networks (Page 7190)

Inserting networks (Page 7191)

Inserting network title (Page 7194)

Entering a network comment (Page 7196)

Navigating networks (Page 7197)

14.3.4 Copying and pasting networks

Copied networks can be pasted within the block or in another block. Networks that were created in LAD or FBD can also be inserted in blocks of the respective other programming language.

Requirement

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Using networks (Page 7190)
Inserting networks (Page 7191)
Selecting networks (Page 7192)
Inserting network title (Page 7194)
Entering a network comment (Page 7196)
Navigating networks (Page 7197)

14.3.5 Deleting networks

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Using networks (Page 7190)
Inserting networks (Page 7191)
Selecting networks (Page 7192)
Copying and pasting networks (Page 7192)
Inserting network title (Page 7194)
Entering a network comment (Page 7196)
Navigating networks (Page 7197)

14.3.6 Expanding and collapsing networks

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Using networks (Page 7190)
Inserting networks (Page 7191)
Selecting networks (Page 7192)
Copying and pasting networks (Page 7192)
Deleting networks (Page 7193)
Inserting network title (Page 7194)
Entering a network comment (Page 7196)
Navigating networks (Page 7197)

14.3.7 Inserting network title

The network title is the header of a network. The length of the network title is limited to one line. You can enter the title manually or set it automatically. When you set it automatically, you can do this for individual networks or use the settings to specify that the network title is always set automatically.

For automatic insertion of the network title, the comment of the operand in one of the following instructions in the network is evaluated:

- Assignment
- Set output
- Reset output

The instruction that is listed first in the network is used.

The network title is only inserted automatically when the following conditions are fulfilled:

- The network does not have a title yet.
- The operand of the instruction used for the comment has a comment.

Note

Note the following restrictions for automatic insertion of the network title:

- The network title is not adapted if you change the comment of the operand at a later time.
- The network title is not adapted if you change the operand of the instruction.
- The network title is only set by the writing instructions listed above.
- If the operand is of the data type array, the comment of the array is used and not the comments of the array elements.
- Comments of invalid operands are not taken into consideration.

Entering the network title manually

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

Setting the network title automatically

To specify that the network titles are always set automatically, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Select the "Set network title automatically" check box in the "Additional settings" group.
   The network titles are set automatically as of this time if the conditions listed above are fulfilled.

To set an individual network title automatically, follow these steps:

1. Right-click "Network <Number of the network>" in the title bar of a network.
2. Select the "Set network title automatically" command in the shortcut menu.
   The title of the selected network is set based on the comment of the operand if the conditions listed above are fulfilled.

See also

Using networks (Page 7190)

Inserting networks (Page 7191)

Selecting networks (Page 7192)

Copying and pasting networks (Page 7192)

Deleting networks (Page 7193)

Expanding and collapsing networks (Page 7194)

Entering a network comment (Page 7196)

Navigating networks (Page 7197)

14.3.8 Entering a network comment

You can use network comments to provide comments on the program contents of individual networks. For example, you can indicate the function of the network or draw attention to special characteristics.

Requirement

A network is available.

Procedure

To enter a network comment, follow these steps:

1. Click on the right arrow before the network title.
2. If the comment area is not visible, click "Network comments on/off" in the toolbar.
   The comment area is displayed.
3. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
4. Enter the network comment.

See also

Using networks (Page 7190)

Inserting networks (Page 7191)

Selecting networks (Page 7192)

Copying and pasting networks (Page 7192)

Deleting networks (Page 7193)

Expanding and collapsing networks (Page 7194)

Inserting network title (Page 7194)

Navigating networks (Page 7197)

14.3.9 Navigating networks

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click in the code area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.
   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 7190)

Inserting networks (Page 7191)

Selecting networks (Page 7192)

Copying and pasting networks (Page 7192)

Deleting networks (Page 7193)

Expanding and collapsing networks (Page 7194)

Inserting network title (Page 7194)

Entering a network comment (Page 7196)

14.4 Inserting FBD elements

14.4.1 Rules for the use of FBD elements

Rules

Note the following rules when inserting FBD elements:

- An FBD network can consist of several elements. All elements of a logic path must be linked to each other according to IEC 61131-3.
- Standard boxes (flip flops, counters, timers, math operations, etc.) can be added as output to boxes with binary logic operations (for example, AND, OR). Comparison boxes are excluded from this rule.
- Only Boolean inputs in an instruction can be combined with preceding logic operations.
- Only the bottom Boolean output in an instruction can be combined with an additional logic operation.
- Enable input EN or enable output ENO can be connected to boxes, but this is not mandatory.
- Constants (for example, TRUE or FALSE) cannot be assigned to binary logic operations. Instead, use tags of the BOOL data type.
- Only one jump instruction can be inserted in each network.
- Only one jump label can be inserted in each network.
- Instructions for positive or negative RLO edge detection may not be arranged right at the left of the network as this requires a prior logic operation.

Placement rules for S7-1200/1500 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
SET_BF	Set bit field	No
RESET_BF	Reset bit field	No
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
JMP_LIST	Define jump list	No
SWITCH	Jump distributor	No
RET	Return	No

Placement rules for S7-300/400 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
S	Set output	Yes
R	Reset output	Yes
SP	Start pulse timer	Yes
SE	Start extended pulse timer	Yes
SD	Start on-delay timer	Yes
SS	Start retentive on-delay timer	Yes
SF	Start off-delay timer	Yes
SC	Set counter value	Yes
CU	Count up	Yes
CD	Count down	Yes
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
RET	Return	No
OPN	Open global data block	No
OPNI	Open instance data block	No
CALL	Call block	No
SAVE	Save RLO in BR bit	No
MCRA	Enable MCR range	No
MCRD	Disable MCR range	No
MCR<	Open MCR ranges	No
MCR>	Close MCR ranges	No

Note

Calling blocks with binary inputs in S7-300/400 CPUs

The following FAQ describes the special features when calling blocks with binary inputs:

FAQ 13988019: What should you watch out for when programming nested FB/FC calls with binary inputs in FBD? (https://support.industry.siemens.com/cs/de/de/view/13988019)

14.4.2 Inserting FBD elements using the "Instructions" task card

Requirement

A network is available.

Procedure

To insert FBD elements into a network using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to the FBD element that you want to insert.
3. Use drag-and-drop to move the element to the desired place in the network.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog, you can determine if the data of the inserted element is to be stored in a single, multi or parameter instance. If you select "Single instance", you can find the new instance data block once it is created in the "Program resources" folder of the project tree under "Program blocks > System blocks". You can find multi-instances in the block interface in the "Static" section and parameter instances in the "InOut" section.

Or:

1. Select the point in the network at which you want to insert the element.
2. Open the "Instructions" task card.
3. Double-click on the element you want to insert.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog, you can determine if the data of the inserted element is to be stored in a single, multi or parameter instance. If you select "Single instance", you can find the new instance data block once it is created in the "Program resources" folder of the project tree under "Program blocks > System blocks". You can find multi-instances in the block interface in the "Static" section and parameter instances in the "InOut" section.

Result

The selected FBD element is inserted with dummy entries for the parameters.

See also

Rules for the use of FBD elements (Page 7198)

Single instances (Page 63)

Multi-instances (Page 65)

Parameter instances (Page 68)

14.4.3 Inserting FBD elements using an empty box

Requirement

A network is available.

Procedure

To insert FBD elements into a network using an empty box, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Empty box" in the "Basic instructions" palette.
3. Use a drag-and-drop operation to move the "Empty box" element to the desired place in the network.
4. Position the cursor over the triangle in the top right-hand corner of the empty box.
   A drop-down list is displayed.
5. Select the desired FBD element from the drop-down list.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog, you can determine if the data of the inserted element is to be stored in a single, multi or parameter instance. If you select "Single instance", you can find the new instance data block once it is created in the "Program resources" folder of the project tree under "Program blocks > System blocks". You can find multi-instances in the block interface in the "Static" section and parameter instances in the "InOut" section.

Result

The empty box is changed to the respective FBD element. Placeholders are inserted for the parameters.

See also

Single instances (Page 63)

Multi-instances (Page 65)

Parameter instances (Page 68)

14.4.4 Selecting the data type of an FBD element

14.4.4.1 Selecting a data type

Introduction

Some instructions can be executed with several different data types. If you use one of these instructions in the program, you have to specify a valid data type for the instruction at the specific point in the program. For some instructions, you have to select the data types for the inputs and outputs separately.

Note

The valid data type (BOOL) for the tags on the enable input EN and the enable output ENO is predefined by the system and cannot be changed.

The valid data types for an instruction are listed in the instruction drop-down list. You specify the data type of the instruction by selecting an entry from the drop-down list. If the data type of an operand differs from the data type of the instruction and cannot be converted implicitly, the operand is displayed in red and a rollout with the corresponding error message appears.

Data type selection of mathematical instructions

Some mathematical instructions provide you with the option of having the data type automatically set corresponding to the data types of the operand. In the drop-down list for data type selection, these instructions have the entry "Auto" in addition to the actual data types. If you select this entry and then allocate the first operand, the data type of the operand is selected as data type for the instruction. The entry in the drop-down list changes to "Auto (<Data type>)", e.g. "Auto (Real)". If you allocate additional operands, the automatically set data type of the instruction is adjusted according to the following criteria:

- You supply all other operands with tags of the same data type:
  The data type of the instruction is not changed.
- You supply all other operands with tags whose data type is smaller than the data type of the instruction:
  The data type of the instruction is not changed. For the operand with the smaller data type, an implicit conversion is conducted if necessary.
- You supply an additional operand with a tag whose data type is greater than the data type of the instruction:
  The data type of the instruction is changed to the larger data type. An implicit conversion is performed, if necessary, for operands that deviate from the newly set data type of the instruction.

Each change in the data type of an operand can result in a change of the data type of the instruction. Other operands may possibly be implicitly converted as a result. Operands for which an implicit conversion is performed are marked with a gray square.

Note

Please also observe the information on data type conversion for your device and, in particular, the notes on the IEC check.

See also: Data type conversion (Page 489)

See also

Defining the data type of an instruction (Page 7202)

14.4.4.2 Defining the data type of an instruction

Introduction

Some instructions can be executed with several different data types. When you insert such instructions into your program, you must specify the data type for these instructions at the actual point in the program.

Specifying the data type by means of the drop-down list

To define the data type of an instruction using the drop-down list, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Click the triangle in the upper corner of the drop-down list.
   The drop-down list will open to display the data types valid for the instruction.
3. Select a data type from the drop-down list.
   The selected data type is displayed.
4. If the instruction has two drop-down lists, select the data type for the instruction inputs in the left-hand drop-down list and the data type for the instruction outputs in the right-hand drop-down list.

Specifying data type by assigning tags

To define the data type of an instruction by assigning tags, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. At an input or output, specify a valid tag, the data type of which is to be applied as the instruction data type.
   The data type of the tag is displayed in the drop-down list.
3. Enter a valid tag at an input and a valid tag at an output if data types need to be defined for both the inputs and outputs of the instruction. The tag specified at the input determines the data type of the inputs; the tag specified at the output determines the data type of the outputs of the instruction.

Automatically specifying the data type of mathematical instructions

To automatically specify the data type for mathematical instructions, follow these steps:

1. Insert the mathematical instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Select the "Auto" entry from the drop-down list.
3. Enter a valid tag at an input or output.
   The data type of the tag is applied as data type of the instruction. The entry in the drop-down list changes to "Auto (<Data type>)".

See also: Selecting a data type (Page 7201)

See also

Selecting a data type (Page 7201)

14.4.5 Using favorites in FBD

14.4.5.1 Adding FBD elements to Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing FBD elements from Favorites (Page 7205)

Overview of the program editor (Page 6831)

14.4.5.2 Inserting FBD elements using favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 6831)

Removing FBD elements from Favorites (Page 7205)

14.4.5.3 Removing FBD elements from Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding FBD elements to Favorites (Page 7204)

Inserting FBD elements using favorites (Page 7204)

Overview of the program editor (Page 6831)

14.4.6 Inserting block calls in FBD

14.4.6.1 Inserting block calls using a drag-and-drop operation

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree. If you call function blocks from other function blocks, you can either call them as a single instance, multi-instance or parameter instance.

See also: Basics of instances (Page 62)

Requirement

- A network is available.
- The block that is to be called is available.

Inserting a call of a function (FC)

To insert a call of a function (FC) into a network using a drag-and-drop operation, follow these steps:

1. Drag the function from the project tree to the required network.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Drag the function block from the project tree to the required network.
   The "Call options" dialog opens.
2. In the dialog, enter whether you want to call the block as a single, multi or parameter instance.
   - If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block.
     If you call a block that contains monitoring, assign a ProDiag function block to the monitoring functions in the "ProDiag FB" text box.
   - If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
   - If you click on the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
3. If you call a block that contains monitoring, assign a ProDiag function block to the monitoring functions.
4. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You can then assign the parameters.

See also: Parameter transfer at block call (Page 76)

Note

If when calling a function block you specify an instance data block that does not exist, it will be created.

See also

Updating block calls in FBD (Page 7207)

Changing the instance type (Page 7209)

Single instances (Page 63)

Multi-instances (Page 65)

14.4.6.2 Updating block calls in FBD

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have the following options for updating the block calls:

- Explicit updating of all inconsistent block calls in the program editor.
  The inconsistent block calls within the open block are updated. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are removed if they are not connected.
  - Renamed parameters get the new parameter names.
- Explicit updating of a block call in the program editor.
  The "Interface update" dialog is displayed. In this dialog, you have the option of changing the connection of the operands of the new interface. The inconsistent call of this block is then updated. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are removed if they are not connected.
  - Renamed parameters get the new parameter names.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated. Note that when you call functions (FCs) before the next compilation process, all new formal parameters must be supplied with actual parameters.

Updating all inconsistent block calls in the program editor

To open all block calls in a block, follow these steps:

1. Open the calling block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

Updating a specific block call in the program editor

To update a specific block call in the program editor, follow these steps:

1. Open the calling block in the program editor.
2. Right-click on the block call that you want to update.
3. Select the "Update" command in the shortcut menu.
   The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
4. If necessary, change the connection of the operands. To do this, you have the following options:
   – You can use either a drag-and-drop operation or a cut/copy-and-paste operation to move the operand from the old interface to the new interface.
   – You can delete an operand.
   – You can rename an operand.
   – You can specify a new operand via autocompletion.
5. Click "OK" to update the block call. If you want to cancel the update, click "Cancel".

Note

Note that the "Update block call" command is only available provided you did not previously update all block calls in the editor with the "Update inconsistent block calls" command.

Update block calls during compilation

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Inserting block calls using a drag-and-drop operation (Page 7206)

Changing the instance type (Page 7209)

Block calls (Page 60)

14.4.6.3 Changing the block call

You have the option of changing the called block for a block call. But keep in mind that no new instance data blocks are created, for example, when changing from a function (FC) to a function block (FB).

Procedure

To change the called block of a block call, follow these steps:

1. Click on the name of the called block within the block call and press the <F2> key. Or double-click the name of the called block.
   A text box opens, and the name of the currently called block is selected.
2. Enter the name of the block you want to call or select a block in the autocompletion.
3. If you want to call an FB, create a new instance data block, if necessary, and specify it as operand.

14.4.6.4 Changing the instance type

Instance type

There are two ways of calling function blocks:

- As a single instance
- As a multiple instance
- As parameter instance

See also: Basics of instances (Page 62)

You can modify a defined instance type at any time.

Requirement

The user program contains a block call.

Procedure

To change the instance type of a function block, follow these steps:

1. Open the code block and select the block call.
2. Select the "Change instance" command in the shortcut menu.
   The "Call options" dialog opens.

3. Click on the button "Single instance", "Multi-instance" or "Parameter instance".
   - If you select the "Single instance" instance type, enter a name for the data block that is to be assigned to the function block.
   - If you select "Multiple instance" as the instance type, enter in the "Name in the interface" text field the name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
   - If you select "Parameter instance" as the instance type, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
4. Confirm your entries with "OK".

Note

The previous instances are not deleted automatically.

See also

Inserting block calls using a drag-and-drop operation (Page 7206)

Updating block calls in FBD (Page 7207)

Instances (Page 62)

14.4.7 Inserting complex FBD instructions

14.4.7.1 Using the "Calculate" instruction (S7-1200, S7-1500)

Requirement

A network is available.

Procedure

Proceed as follows to use the "Calculate" instruction:

1. Open the "Instructions" task card.
2. Navigate to "Math functions > CALCULATE" in the "Basic instructions" pane.
3. Use drag-and-drop to move the element to the desired place in the network.
   The instruction "Calculate" will be inserted for the data type with a placeholder expression and question mark.
4. Enter the data type for the calculation.

5. Enter the operands for the calculation.

   Note

   The calculation is run with the inputs of the "Calculate" instruction. If you want to use constants you must also insert appropriate inputs for them.

6. Click on the "Edit 'Calculate' instruction" button to replace the placeholder expression with the correct expression.
   The "Edit 'Calculate' instruction" dialog will open.
7. Enter the required expression in the "OUT:= " text box.

   Note

   In the "Example" area you can find an example of a valid expression and possible instructions that you can use.

   To determine a value with the help of Pythagoras' theorem, for example, enter "OUT := SQRT (SQR (IN1) + SQR (IN2))".

8. Confirm your entry with "OK".

See also

CALCULATE: Calculate (Page 1073)

14.4.8 Using free-form comments

14.4.8.1 Basic information on using free comments in FBD

Introduction

Free-form comments allow you to add comments to the source code for graphic programming languages similar to line comments for textual languages.

Free-form comments can be used for all non-binary boxes.

Commenting out program code

To deactivate individual instructions or entire program sections in the LAD and FBD programming languages, you must set a bit parallel to the instruction or in series to an instruction block. Commenting out, as in STL and SCL, is not possible in LAD and FBD.

Examples of commenting out program code in all programming languages are available in the Siemens Industry Online under the following FAQ ID: 109482004 (https://support.industry.siemens.com/cs/hu/en/view/109482004)

See also

Inserting free-form comments (Page 7212)
Editing free-form comments (Page 7212)
Deleting free-form comments (Page 7214)

14.4.8.2 Inserting free-form comments

Requirement

A network with instructions is available.

Procedure

To insert a free comment on an instruction, proceed as follows:

1. If necessary, activate the "Free-form comments on/off" button in the toolbar.
2. Right-click on the instruction for which you want to insert a free-form comment.
3. Select the "Insert comment" command in the shortcut menu.
   A comment box with a standard comment opens. The comment box is connected by an arrow to the corresponding instruction.
4. Enter the required comment in the comment box.

See also

Basic information on using free comments in FBD (Page 7211)
Editing free-form comments (Page 7212)
Deleting free-form comments (Page 7214)

14.4.8.3 Editing free-form comments

Introduction

Free-form comments can be edited as follows:

- Changing the comment text
- Changing the position and size of the comment box
- Attaching a comment to another element
- Showing and hiding free comments

Changing the comment text

To change the text of free-form comments, follow these steps:

1. Click on the comment box.
2. Enter the desired text.

Changing the position of the comment box

To change the positioning of the comment box, follow the steps below:

1. Left-click the comment box and keep the mouse button pressed.
2. Drag the comment box to the desired location.

Changing the size of the comment box

To change the size of the comment box, follow the steps below:

1. Click on the comment box.
2. Drag the comment box on the move handle in the lower right corner to the desired size.

Attaching a comment to another element

To attach a free-form comment to another element, follow these steps:

1. Left-click the point of the arrow that links the comment box with the instruction and keep the mouse button pressed.
2. Drag the arrow to the element to which you want to attach the comment. Possible insertion points are marked with a green square.
3. Release the mouse button.

Showing and hiding free comments

To show or hide a free-form comments, follow these steps:

1. Click the "Free-form comment on/off" button in the toolbar.

See also

Basic information on using free comments in FBD (Page 7211)

Inserting free-form comments (Page 7212)

Deleting free-form comments (Page 7214)

14.4.8.4 Deleting free-form comments

Procedure

To delete a free-form comment, proceed as follows:

1. Right-click on the free-form comment that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on using free comments in FBD (Page 7211)
Inserting free-form comments (Page 7212)
Editing free-form comments (Page 7212)

14.5 Editing FBD elements

14.5.1 Selecting FBD elements

You can select several individual elements or all elements in a network.

Requirement

FBD elements are available

Selecting several individual FBD elements

To select several individual FBD elements, follow these steps:

1. Press and hold down the <Ctrl> key.
2. Click on all the FBD elements you wish to select.
3. Now release the <Ctrl> key.

Selecting all FBD elements in a network

To select all FBD elements in a network, follow these steps:

1. Go to the network whose elements you wish to select.
2. Select the "Select all" command in the "Edit" menu or press <Ctrl+A>.

See also

Copying FBD elements (Page 7215)
Cutting FBD elements (Page 7216)
Pasting an FBD element from the clipboard (Page 7217)
Replacing FBD elements (Page 7217)
Adding additional inputs and outputs to FBD elements (Page 7218)
Removing instruction inputs and outputs (Page 7219)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 149)
Deleting FBD elements (Page 7220)

14.5.2 Copying FBD elements

Requirement

An FBD element is available.

Procedure

To copy an FBD element, follow these steps:

1. Right-click the FBD element that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

The FBD element will be copied and saved to the clipboard.

See also

Selecting FBD elements (Page 7215)
Cutting FBD elements (Page 7216)
Pasting an FBD element from the clipboard (Page 7217)
Replacing FBD elements (Page 7217)
Adding additional inputs and outputs to FBD elements (Page 7218)
Removing instruction inputs and outputs (Page 7219)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 149)
Deleting FBD elements (Page 7220)

14.5.3 Cutting FBD elements

Requirement

An FBD element is available.

Cutting

To cut an FBD element, follow these steps:

1. Right-click the FBD element that you want to cut.
2. Select "Cut" in the shortcut menu.

Result

The FBD element will be cut and saved to the clipboard.

See also

Selecting FBD elements (Page 7215)
Copying FBD elements (Page 7215)

Pasting an FBD element from the clipboard (Page 7217)
Replacing FBD elements (Page 7217)
Adding additional inputs and outputs to FBD elements (Page 7218)
Removing instruction inputs and outputs (Page 7219)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 149)
Deleting FBD elements (Page 7220)

14.5.4 Pasting an FBD element from the clipboard

Requirement

An FBD element is available.

Procedure

To paste an FBD element from the clipboard, follow these steps:

1. Copy an FBD element or cut an FBD element.
2. Right-click the point in the network where you want to paste the element.
3. Select "Paste" in the shortcut menu.

See also

Selecting FBD elements (Page 7215)
Copying FBD elements (Page 7215)
Cutting FBD elements (Page 7216)
Replacing FBD elements (Page 7217)
Adding additional inputs and outputs to FBD elements (Page 7218)
Removing instruction inputs and outputs (Page 7219)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 149)
Deleting FBD elements (Page 7220)

14.5.5 Replacing FBD elements

You can easily exchange FBD elements with other FBD elements of the same type. This has the advantage that the parameters are retained and need not be entered again. For example, you can exchange OR and AND, RS-FlipFlop and SR-FlipFlop, comparison functions or jump instructions.

Requirements

A network with at least one FBD element is present.

Procedure

To replace an FBD element with another FBD element, follow these steps:

1. Select the FBD element that you want to replace.
   If elements compatible with the selected FBD element are available, a triangle will appear in the upper right-hand corner of the element.
2. Position the cursor above the triangle of the FBD element.
   A drop-down list is displayed.
3. From the drop-down list, select the FBD element that you want to use to replace the existing FBD element.

See also

Selecting FBD elements (Page 7215)

Copying FBD elements (Page 7215)

Cutting FBD elements (Page 7216)

Pasting an FBD element from the clipboard (Page 7217)

Adding additional inputs and outputs to FBD elements (Page 7218)

Removing instruction inputs and outputs (Page 7219)

Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 149)

Deleting FBD elements (Page 7220)

14.5.6 Adding additional inputs and outputs to FBD elements (S7-1200, S7-1500)

Introduction

You can expand several FBD elements with additional inputs that execute arithmetic or binary operations. Such elements are, for example, the instructions "Add" (ADD), "Multiply" (MUL), AND or OR. You can expand the "MOVE value" (MOVE) and "Demultiplex" (DEMUX) instruction boxes by adding additional outputs.

The name of the new inputs and outputs is comprised of the type of inserted element and a consecutive number. The name of a new input is may be "IN2"; the name of a new output may be "OUT2".

Requirements

An FBD element is available that permits the insertion of additional inputs and outputs.

Inserting an additional input

To add an additional input to the box of an FBD element, follow these steps:

1. Right-click on an existing input of the FBD element.
2. Select "Insert input" in the shortcut menu.
   An additional input is added to the box of the FBD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.
   An additional input is added to the box of the FBD element.

Inserting an additional output

To add an additional output to the box of an FBD element, follow these steps:

1. Right-click on an existing output of the FBD element.
2. Select "Insert output" from the shortcut menu.
   An additional output is added to the box of the FBD element.

Or:

1. Click on the yellow star symbol beside the last output of the instruction box.
   An additional output is added to the box of the FBD element.

See also

Inserting FBD elements (Page 7198)

Selecting FBD elements (Page 7215)

Copying FBD elements (Page 7215)

Cutting FBD elements (Page 7216)

Pasting an FBD element from the clipboard (Page 7217)

Replacing FBD elements (Page 7217)

Removing instruction inputs and outputs (Page 7219)

Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 149)

Deleting FBD elements (Page 7220)

14.5.7 Removing instruction inputs and outputs (S7-1200, S7-1500)

Introduction

Inputs and outputs which you have added to an instruction can be removed.

Requirement

An FBD element is available, which you have expanded with additional inputs or outputs.

Remove input

To remove an input, follow these steps:

1. Select the input that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The input of the FBD element is removed.

Remove output

To remove an output, follow these steps:

1. Select the output that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The output of the FBD element will be removed.

See also

Inserting FBD elements (Page 7198)
Selecting FBD elements (Page 7215)
Copying FBD elements (Page 7215)
Cutting FBD elements (Page 7216)
Pasting an FBD element from the clipboard (Page 7217)
Replacing FBD elements (Page 7217)
Adding additional inputs and outputs to FBD elements (Page 7218)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 149)
Deleting FBD elements (Page 7220)

14.5.8 Deleting FBD elements

Requirement

An FBD element is available.

Procedure

To delete an FBD element, follow these steps:

1. Right-click the FBD element that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting FBD elements (Page 7198)
Selecting FBD elements (Page 7215)
Copying FBD elements (Page 7215)
Cutting FBD elements (Page 7216)
Pasting an FBD element from the clipboard (Page 7217)
Replacing FBD elements (Page 7217)
Adding additional inputs and outputs to FBD elements (Page 7218)
Removing instruction inputs and outputs (Page 7219)
Enabling and disabling the EN/ENO mechanism in LAD and FBD (Page 149)

14.6 Inserting operands in FBD instructions

14.6.1 Inserting operands

The character strings "<???>", "<??.?>" and "..." are inserted as placeholders for the parameters when a FBD element is inserted. The "<???>" and "<??.?>" strings displayed in red indicate parameters that need to be connected. The "..." string displayed in black indicates parameters that may be connected. "<??.?>" stands for Boolean placeholders. In addition, FBD elements can have the following different I/Os for the parameters:

- Orange I/Os: Only tags or constants can be interconnected at this I/O.
- Black I/Os: Boolean input or output to which other elements can be connected upstream or downstream.

Note

To display the available data types in a tooltip, move the cursor over the placeholder.

Requirement

An FBD element is available.

Procedure

To connect the parameters of an FBD element, follow these steps:

1. Click the placeholder of the parameter.
   An input field is opened.
2. Enter the corresponding parameters, for example a PLC tag, a local tag or a constant.

   Note

   If you enter the absolute address of a parameter that has already been defined, this absolute address will be changed to the symbolic name of the parameter as soon as the input is confirmed. If you have not yet defined the parameter, a new tag with this absolute address and the default name "Tag_1" will be entered in the PLC tag table. When you confirm your input, the absolute address will be replaced with the symbolic name "Tag_1".

3. Confirm the parameter with the Enter key.
4. If you have not yet defined the parameter, you can define it directly in the program editor using the shortcut menu.
   See also: "Declaring PLC tags in the program editor (Page 7080)".

Or drag from it the PLC tag table:

1. In the project tree, select the "PLC tags" folder and open the PLC tag table.
2. If you have opened the PLC tag table, drag the desired tag to the corresponding location in your program. If you have not opened the PLC tag table yet, open the detail view now. Drag the desired tag from the detail view to the appropriate place in your program.

Or drag from it the block interface:

1. Open the block interface.
2. Drag the desired operand from the block interface to the corresponding location in your program.

Result

- If the syntax is error-free, the displayed parameter is black.
- If there is an error in the syntax, the cursor stays in the input field and a corresponding error message is displayed in the inspector window in the "Info > Syntax" register.

See also

Using and addressing operands (Page 89)

Using autocompletion (Page 6859)

14.6.2 Wiring hidden parameters

Introduction

If an instruction contains hidden parameters, the instruction box has a small arrow on the lower edge. You can recognize hidden parameters by their white font.

You can show and wire the hidden parameters at any time.

Showing or hiding hidden parameters

To show or hide hidden parameters, follow these steps:

1. Click on the down arrow at the bottom edge of the instruction box to show hidden parameters.
2. Click on the up arrow at the bottom edge of the instruction box to hide hidden parameters.

Wiring hidden parameters

To wire parameters, follow these steps:

1. Wire the hidden parameters like normally visible parameters.
   Its visibility may change due to the interconnection of the parameter:
   - The parameter is visible if you have selected the "Hide, if no parameter is assigned" option in the properties.
   - The parameter remains hidden if you have selected the "Hide" option in the properties.

See also

Hiding parameters during the block call (Page 6986)

14.6.3 Displaying or hiding tag information

Introduction

You can display the following information about the tags to be used in the Program editor:

- Name of the tag
- Tag addresses
- Simple or hierarchical comments to document the tags

The information is taken from the block interface for local tags and DB tags and from the PLC tag table for tags that are valid CPU-wide.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all the blocks currently opened and opened in future is shown. You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual blocks that are open.

When you select the display of tag information with hierarchical comments, the comments of the higher structure levels are also displayed for structured tags. The comments are shown in brackets after the tag comment; the comments of the individual levels are separated by a dot. If there is no comment for a tag on a structure level, it is not displayed. This is indicated by two successive dots.

You can display the tag information either collectively below a LAD/FBD network or directly at the respective operand. As with showing and hiding tag information, you can set this via the settings for all block or change specifically for an opened block.

Displaying or hiding tag information for all the blocks

Proceed as follows to display or hide the tag information for all the blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to display the tag information, select the "Show" option or the "Tag information with hierarchical comments" option in the drop-down list, depending on whether you want to display simple or hierarchical comments.

4. Use the "Position of the tag information (LAD/FBD)" drop-down list to specify whether the tag information is to be displayed below the LAD/FBD network or on the operand.
   For all open blocks, the tag information is shown and displayed at the selected position. If you open additional blocks the tag information for these blocks is also shown.
5. If you want to hide the tag information, select the option "Hide" in the "Tag information" drop-down list.
   The tag information is hidden for all open blocks. If you open additional blocks the tag information for these blocks is also hidden.

Displaying or hiding tag information for an opened block

Proceed as follows to display or hide the tag information for an opened block:

1. If you want to display the tag information, select the "Show tag information" option or the "Tag information with hierarchical comments" option in the "Shows the tag information" drop-down list, depending on whether you want to display simple or hierarchical comments.
2. To define the position of the tag information, click the "Position of the tag information (LAD/FBD)" button in the program editor toolbar. The position changes with each click on the button. Alternatively, you can click the small arrow beside the "Position of the tag information (LAD/FBD)" button to open the drop-down list. Than select the position from the drop-down list.
   The tag information is shown and displayed at the selected position.
3. If you want to hide the tag information, select the option "Hide tag information" in the "Shows the tag information" drop-down list.
   The tag information is hidden.

14.7 Branches in FBD

14.7.1 Basic information on branches in FBD

Definition

You can use the Function Block Diagram (FBD) programming language to program parallel branches. This is done using branches that are inserted between the boxes. You can insert additional boxes within the branch and in this way build up complex function block diagrams.

The figure below shows an example of the use of branches:

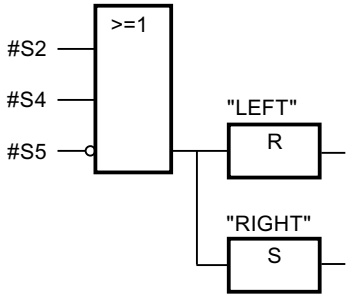

See also

Rules for branches in FBD (Page 7226)

Inserting branches in FBD networks (Page 7227)

Deleting branches in FBD networks (Page 7227)

14.7.2 Rules for branches in FBD

Rules

The following rules apply to the use of branches in FBD:

- Branches are opened downward.
- Branches can be inserted only between FBD elements.
- To delete a branch, you must delete all FBD elements, including the branch itself.
- If you delete the connection between two branches, the FBD elements of the interrupted branch will be positioned freely in the network.

See also

Basic information on branches in FBD (Page 7226)
Inserting branches in FBD networks (Page 7227)
Deleting branches in FBD networks (Page 7227)

14.7.3 Inserting branches in FBD networks

Requirement

A network is available.

Procedure

To insert a new branch in a network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Branch" in the "Basic instructions" palette.
3. Drag the element from the "Elements" pane to the a required location on a connection line between two boxes.

See also

Rules for branches in FBD (Page 7226)
Basic information on branches in FBD (Page 7226)
Deleting branches in FBD networks (Page 7227)

14.7.4 Deleting branches in FBD networks

Requirement

A branch is available.

Procedure

To delete a branch, follow these steps:

1. Select the connection line that links the branch to the main branch.
2. Select the "Delete" command in the shortcut menu.

Result

The branch is now deleted. Boxes connected to the deleted branch are placed freely within the network.

See also

Rules for branches in FBD (Page 7226)

Basic information on branches in FBD (Page 7226)

Inserting branches in FBD networks (Page 7227)

14.8 Logic paths in FBD

14.8.1 Basic information on logic paths in FBD

Use of logic paths

The user program will be mapped in one or more networks. The networks can contain one or more logic paths on which the binary signals are arranged in the form of boxes.

The following figure shows an example of the use of several logic paths within a network:

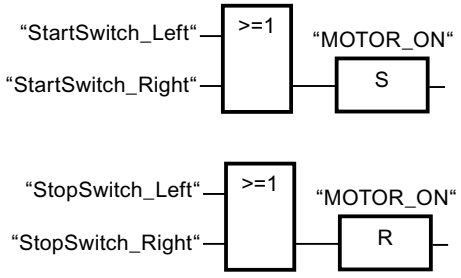

Rules

Remember the following rules when using logic paths:

- Connections are not permitted between logic paths.
- Only one jump instruction is permissible per network. The positioning rules for jump instructions remain valid.

Executing logic paths

Logic paths are executed from top to bottom and from left to right. This means that the first instruction in the first logic path of the first network is executed first. All instructions of this logic path are then executed. After this come all other logic paths of the first network. The next network is executed only after all logic paths have first been executed.

When jumps are used the regular execution of the logic paths is circumvented and the instruction is executed at the jump destination.

Differences between branches and logic paths

The difference between branches and logic paths is that the logic paths are independent branches that can also stand in a different network. Branches, on the other hand, permit the programming of a parallel connection and have a common preceding logic operation.

See also

Inserting a logic path (Page 7230)

Deleting a logic operation path (Page 7230)

14.8.2 Inserting a logic path

Requirement

- A block is open.
- A network is available.

Procedure

To insert a new logic path in a network, follow these steps:

1. Insert any instruction in a network in such a way that it has no connection to existing instructions.
   A new logic path is inserted.
2. Insert an assignment at the end of the new logic path.
3. Insert additional instructions in the new logic path.

See also

Basic information on logic paths in FBD (Page 7229)

Deleting a logic operation path (Page 7230)

14.8.3 Deleting a logic operation path

Requirement

A logic path is available.

Procedure

To delete a logic path, proceed as follows:

1. Hold down the left mouse button and draw a frame around the logic path. At the same time, make sure that you select all instructions of the logic path. Alternatively, you can hold down the <Shift> key and select the first the last instruction of the logic path.
2. Right-click on one of the instructions in the logic path.
3. Select the "Delete" command in the shortcut menu.

See also

Basic information on logic paths in FBD (Page 7229)
Inserting a logic path (Page 7230)

14.9 FBD programming examples

14.9.1 Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbuttons at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbuttons at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.

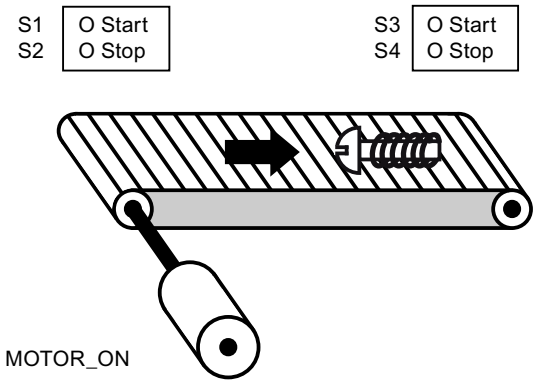

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
StartPushbutton_Left (S1)	Input	BOOL	Start pushbutton on the left side of the conveyor belt
StopPushbutton_Left (S2)	Input	BOOL	Stop pushbutton on the left side of the conveyor belt
StartPushbutton_Right (S3)	Input	BOOL	Start pushbutton on the right side of the convey-or belt
StopPushbutton_Right (S4)	Input	BOOL	Stop pushbutton on the right side of the convey-or belt
MOTOR_ON	Output	BOOL	Turn on the conveyor belt motor

The following networks show the FBD programming for solving this task:

Network 1:

The conveyor belt motor is switched on when start pushbutton "S1" or "S3" is pressed.

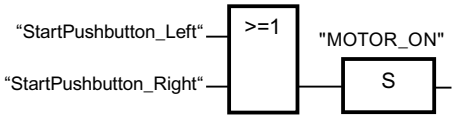

Network 2:

The conveyor belt motor is switched off when stop pushbutton "S2" or "S4" is pressed.

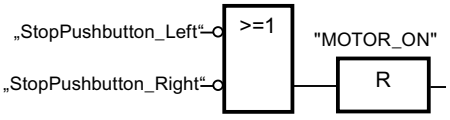

See also

Basic information on FBD (Page 7185)
Settings for FBD (Page 7188)
Working with networks (Page 7190)
Inserting FBD elements (Page 7198)
Editing FBD elements (Page 7215)
Inserting operands in FBD instructions (Page 7222)
Branches in FBD (Page 7226)
Logic paths in FBD (Page 7229)
Example of detecting the direction of a conveyor belt (Page 7234)
Example of detecting the fill level of a storage area (Page 7236)
Example of calculating an equation (Page 7240)
Example of controlling room temperature (Page 7242)

14.9.2 Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The detected running direction of the belt is indicated by a RIGHT arrow or a LEFT arrow. If additional conveyed material is approaching PEB1 from the right or PEB2 from the left, the displayed arrow is initially switched off until, after both photoelectric barriers are passed, the running direction can be detected again and the corresponding arrow displayed. For the solution of the task, 2 edge memory bits are needed that detect the signal change from "0" to "1" at the two photoelectric barriers.

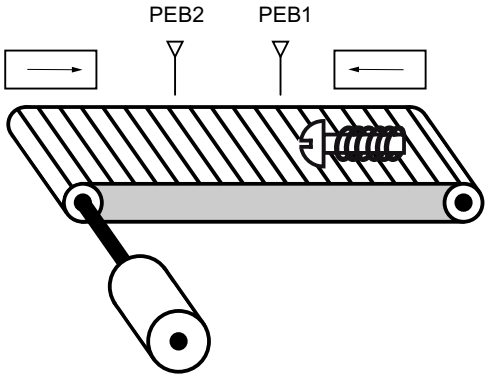

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RIGHT	Output	BOOL	Display during movement to right
LEFT	Output	BOOL	Display during movement to left
CM1	Input	BOOL	Edge bit memory 1
CM2	Input	BOOL	Edge bit memory 2

The following networks show the FBD programming for solving this task:

Network 1:

If the signal state changes from "0" to "1" (positive edge) at photoelectric barrier "PEB1" and, at the same time, the signal state at "PEB2" is "0", the object on the belt is moving to the left.

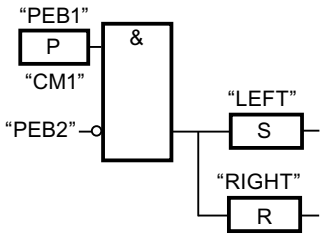

Network 2:

If the signal changes from "0" to "1" (positive edge) at photoelectric barrier "PEB2" and, at the same time, the signal state at "PEB1" is "0", the object on the belt is moving to the right.

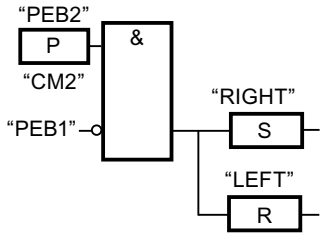

See also

Basic information on FBD (Page 7185)

Settings for FBD (Page 7188)

Working with networks (Page 7190)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

Branches in FBD (Page 7226)

Logic paths in FBD (Page 7229)

Example of controlling a conveyor belt (Page 7232)

Example of detecting the fill level of a storage area (Page 7236)

Example of calculating an equation (Page 7240)

Example of controlling room temperature (Page 7242)

14.9.3 Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock where they are loaded onto trucks. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

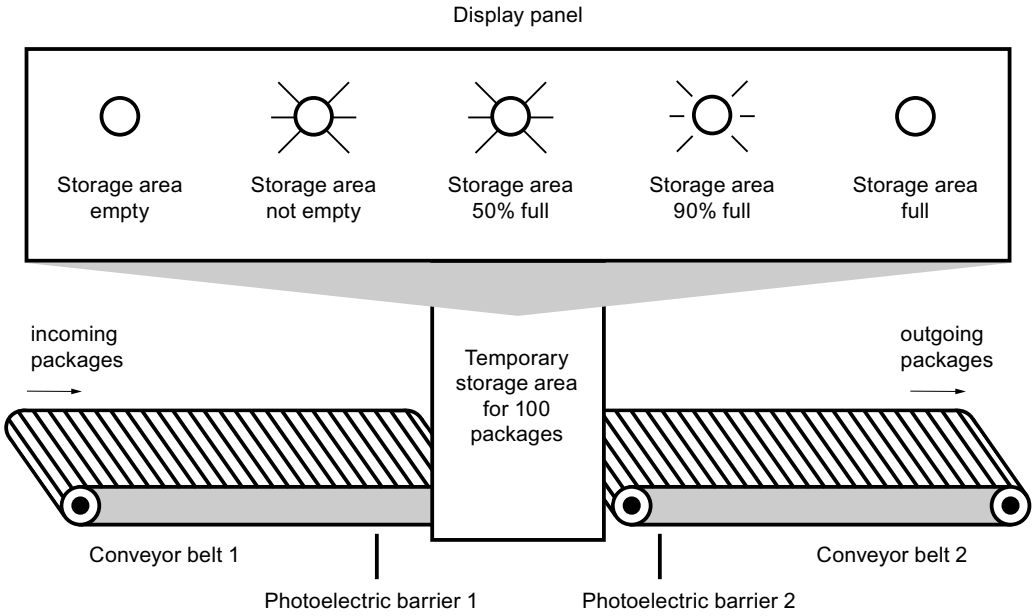

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RESET	Input	BOOL	Reset counter
LOAD	Input	BOOL	Adjust the current counter value to the value of the PV parameter.
MAX STORAGE AREA FILL AMOUNT	Input	INT	Maximum possible number of packages in the storage area
PACKAGECOUNT	Output	INT	Number of packages in the storage area (current count value)

Name	Declaration	Data type	Description
STOCK_PACKAGES	Output	BOOL	Is set when the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag.
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50 % full
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90 % full
STOR_FULL	Output	BOOL	Display lamp: Storage area full
VOLUME_50	Input	INT	Comparison value: 50 packages
VOLUME_90	Input	INT	Comparison value: 90 packages
VOLUME_100	Input	INT	Comparison value: 100 packages

The following networks show the FBD programming for activating the lamps:

Network 1:

When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "MAX STORAGE AREA FILL AMOUNT" tag. As long as the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".

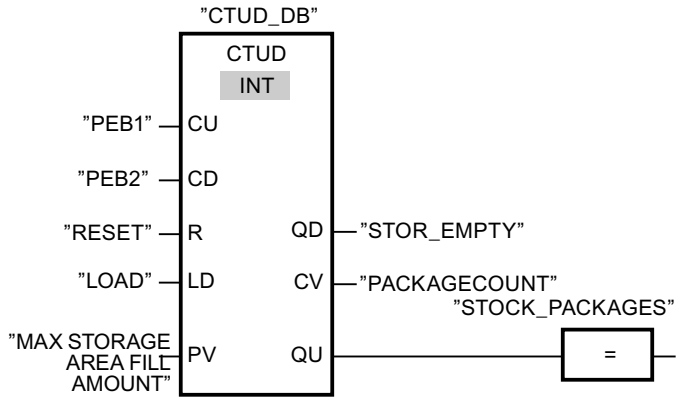

Network 2:

As long as there are packages in the storage area, the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.

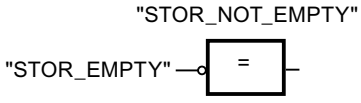

Network 3:

If the number of packages in the storage area is greater than or equal to 50, the "Storage area 50 % full" lamp switches on.

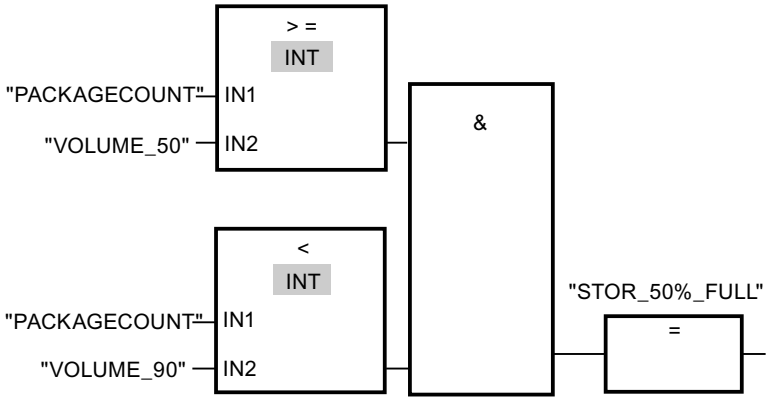

Network 4:

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.

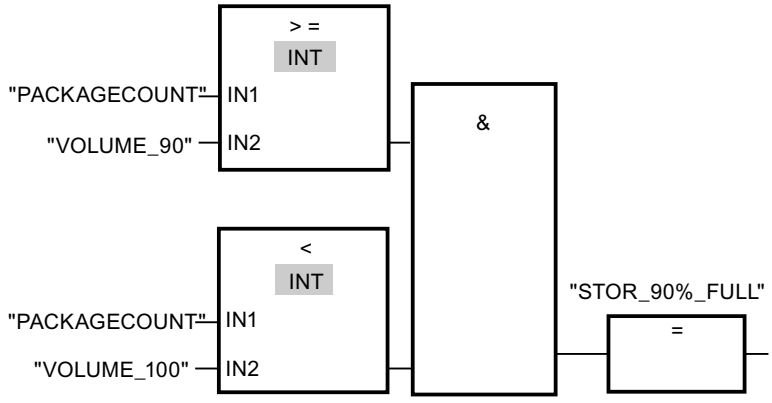

Network 5:

If the number of packages in the storage area reaches 100, the lamp for the "Storage area full" message switches on.

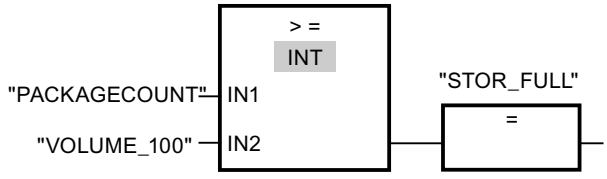

See also

Basic information on FBD (Page 7185)

Settings for FBD (Page 7188)

Working with networks (Page 7190)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

Branches in FBD (Page 7226)

Logic paths in FBD (Page 7229)

Example of controlling a conveyor belt (Page 7232)

Example of detecting the direction of a conveyor belt (Page 7234)

Example of calculating an equation (Page 7240)

Example of controlling room temperature (Page 7242)

14.9.4 Example of calculating an equation

Calculating a complex equation

The following program example shows you how to solve equations of the following type with the FBD programming language:

X= ((A + B) x C) / D

You can always solve this equation with mathematical operations. To do so, you can use the instructions "ADD", "MUL" and "DIV". If you are using a CPU of the S7-1200 or S7-1500 series, you can also use the instruction "CALCULATE". Depending on the data type you choose, different mathematical operations are available for this instruction which can be combined with one another.

Implementation with the instructions "ADD", "MUL" and "DIV"

You want to calculate the following equation:

RESULT = ((A + B) x 15) / E

The following table shows the definition of the tags used:

Name	Data type	Comment
A	INT	First value for addition
B	INT	Second value for addition
C	INT	First intermediate result
15	INT	Multiplier
D	INT	Second intermediate result
E	INT	Divisor
RESULT	INT	End result

The following network shows the FBD programming for calculating the equation:

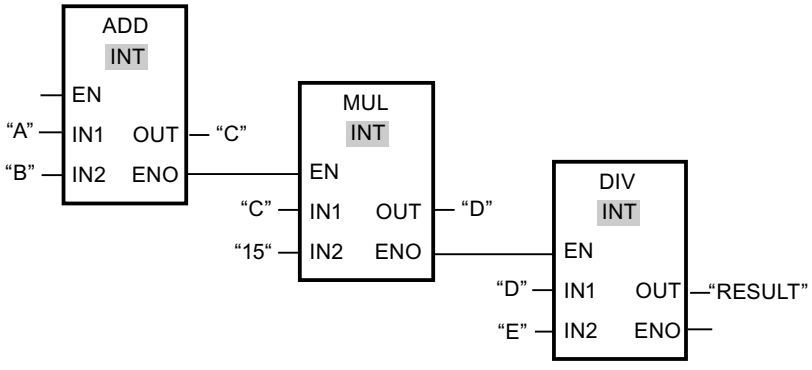

The value of operand "A" is added to the value of operand "B". The sum is stored in operand "C". The value of operand "C" is multiplied by "15". The multiplication result is stored in operand "D". The value stored in operand "D" is then divided by the value of operand "E". The end result is saved in the "RESULT" operand.

Implementation with the instruction "CALCULATE" (S7-1200/1500 only)

You want to calculate the following equation:

RESULT = ((5 + 10) x 4) / 6

The following table shows the definition of the tags used:

Name	Declaration	Data type	Comment	Value
A	Input	INT	First value for addition	5
B	Input	INT	Second value for addition	10
C	Input	INT	Multiplier	4
D	Input	INT	Divisor	6
RESULT	Output	INT	End result	10

To program the equation with the instruction "CALCULATE", follow these steps:

1. Drag the instruction "CALCULATE" from the "Instructions" task card to an FBD network.
2. You can select the data type INT for the instruction from the "<???>" drop-down list.
3. Interconnect the tags declared in the block interface with the inputs and outputs of the instruction box.
4. Click the "Calculator" icon on the top right corner of the instruction box to enter the equation to be calculated.
   The dialog "Edit 'Calculate' instruction" opens.
5. Enter the following expression in the "OUT:=" field:
   ((IN1 + IN2) * IN4) / IN3
   The equation is displayed in the instruction box.
   The following network shows the result in the FBD programming language:

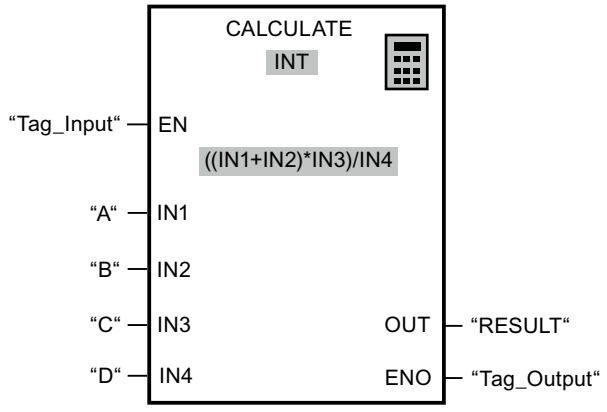

If the "Tag_Input" has signal state "1", the instruction is executed. The value of operand "A" is added to the value of operand "B". The subtotal is multiplied by "C" and then divided by the value of the "D" operand. The end result is saved in the "RESULT" operand.

See also

Basic information on FBD (Page 7185)
Settings for FBD (Page 7188)
Working with networks (Page 7190)
Inserting FBD elements (Page 7198)
Editing FBD elements (Page 7215)
Inserting operands in FBD instructions (Page 7222)
Branches in FBD (Page 7226)
Logic paths in FBD (Page 7229)
Example of controlling a conveyor belt (Page 7232)
Example of detecting the direction of a conveyor belt (Page 7234)
Example of detecting the fill level of a storage area (Page 7236)
Example of controlling room temperature (Page 7242)

14.9.5 Example of controlling room temperature

Controlling room temperature

In a cold room, the temperature must be maintained below zero degrees Celsius. A sensor is used to check for any temperature fluctuations. If the temperature rises above zero degrees Celsius, the cooling system switches on for a preset time. The "Cooling system on" lamp is lit during this time.

The cooling system and the lamp are turned off if one of the following conditions is met:

- The sensor reports a temperature fall below zero degrees Celsius.
- The preset cooling time has elapsed.
- The "Stop" pushbutton is pressed.

If the preset cooling time has expired and the temperature in the cold room is still too high, the cooling system can be restarted with the "RESET" pushbutton.

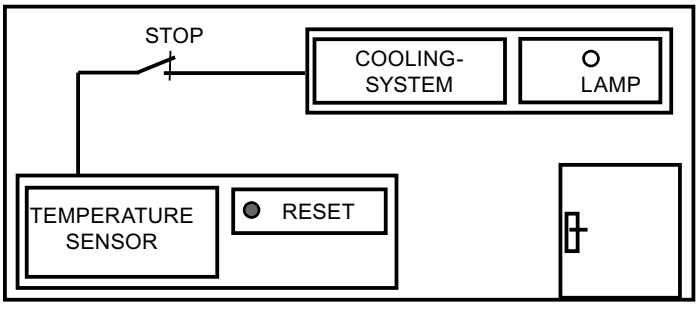

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Comment
Sensor	Input	BOOL	Temperature sensor signal
RESET	Input	BOOL	Restart
STOP	Input	BOOL	The cooling system is switched off.
MaxCoolTime	-	TIME	Preset cooling time This tag is defined in the "DB_Cool" data block.
CurrCoolTime	-	TIME	Currently elapsed cooling time This tag is defined in the "DB_Cool" data block.
Cooling system	Output	BOOL	The cooling system is switched on.
Lamp	Output	BOOL	The lamp for the "Cooling system on" message is switched on.
TempVariable	Temp	BOOL	Temporary tag This tag stores the signal state of the IEC timer TP.

The following network shows the FBD programming for controlling room temperature:

Network 1:

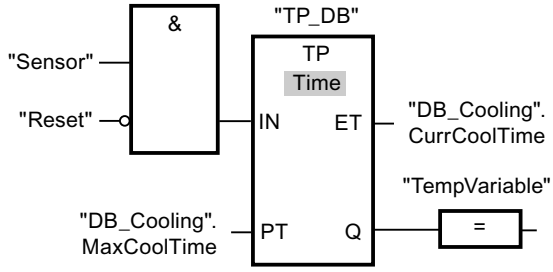

Network 2:

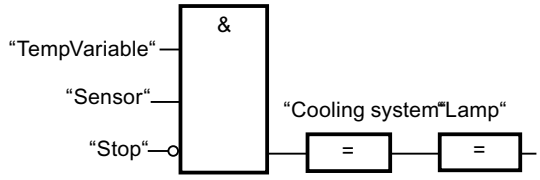

When the temperature in the cold room rises above zero degrees Celsius, the signal state at the "Sensor" operand switches from "0" to "1" (positive signal edge). In the case of a positive signal edge at the IN input of the timer function, the preset cooling time is started and the "TempVariable" receives the signal state "1". The signal state "1" of the "TempVariable" causes

the cooling system as well as the display lamp to be turned on in network 2. The "Sensor", "Cooling system" and "Lamp" outputs must be programmed in network 2, because program only one coil can be programmed at the Q output of the timer function.

If the temperature in the cold room falls below zero degrees Celsius, the signal state of the sensor switches back to "0". This switches the cooling system and lamp off.

If the sensor does not signal a temperature drop, the cooling system and lamp are switched off after the preset cooling time has elapsed, at the latest. In this case, the cooling process can be restarted with the "RESET" pushbutton. Pressing and releasing the pushbutton generates a new positive signal edge at the IN input that restarts the cooling system.

The cooling system and the display lamp can be turned off with the "STOP" pushbutton at any time.

See also

Basic information on FBD (Page 7185)

Settings for FBD (Page 7188)

Working with networks (Page 7190)

Inserting FBD elements (Page 7198)

Editing FBD elements (Page 7215)

Inserting operands in FBD instructions (Page 7222)

Branches in FBD (Page 7226)

Logic paths in FBD (Page 7229)

Example of controlling a conveyor belt (Page 7232)

Example of detecting the direction of a conveyor belt (Page 7234)

Example of detecting the fill level of a storage area (Page 7236)

Example of calculating an equation (Page 7240)

Creating STL programs (S7-300, S7-400, S7-1500) 15

15.1 STL Basics (S7-300, S7-400, S7-1500)

15.1.1 STL programming language (S7-300, S7-400, S7-1500)

Overview of the STL (Statement List) programming language

STL is a text-based programming language you can use to program logic blocks.

The STL program is divided into networks. Each network can contain one or more rows. The numbering of the rows begins with 1 in each network and is continued in ascending order with each new row. The individual STL instructions are programmed in the rows of a network, where only one STL instruction can be specified per row. Each statement represents an instruction to the CPU. The CPU executes the instructions from top to bottom.

The following example shows the STL programming of a network:

STL	Explanation
A"Tag_Input_1"	// Check whether the signal state of the operand is "1" and AND with current RLO
A "Tag_Input_2"	// Check whether the signal state of the operand is "1" and AND with current RLO
S "Tag_Output"	// Sets the operand to "1" if the RLO is "1"

15.1.2 Data exchange in STL (S7-300, S7-400, S7-1500)

S7-300/400 data exchange

For S7-300/400 CPUs, information is exchanged between different memory areas in STL via accumulators. Accumulators are special registers in the process which serve as buffers. All S7-300/400 CPUs have two buffers, namely accumulator 1 (ACCU 1) and accumulator 2 (ACCU 2). The S7-400 CPUs and the S7-318 CPU also have two additional buffers, accumulators 3 (ACCU 3) and 4 (ACCU 4).

S7-1500 data exchange

Although S7-1500 CPUs no longer have a register in the processor, they can emulate the address register, ACCU 1, ACCU 2, the status word and the data block registers DB and DI. Data is exchanged primarily via the block interface, global data blocks or via PLC tags. However, using accumulators for data exchange slows downs the program processing.

When programming in STL for S7-1500, note the following basic instructions regarding the register:

- The contents of the registers, accumulators and the status word are only available within STL networks. When an LAD or FBD network follows an STL network, you cannot access the register contents that were previously set in STL from the LAD or FBD network. However, the register contents are available again in a downstream STL network.
  The RLO bit is an exception: It is set to "undefined" at a language change and is no longer available in downstream networks.
- The values from the registers, accumulators and the status word are never transferred to called blocks. The instructions "CC" and "UC" are the only exceptions. If you use "UC" or "CC" and want to pass the parameters to the called block by using registers, the status word or accumulators, you have to select the "Parameter passing via registers" option in the properties of the called block. Note that this option is only available for STL blocks with standard access and the block can have no formal parameters. If the option is enabled, you can transfer register contents between blocks. The RLO bit is an exception here as well: It is set to "undefined" when the block is exited and is no longer available after a block call.
- The data block register DB is set to "0" after each access to the data block with specification of a fully qualified address (for example, %DB10.DBW10). Subsequent partially qualified access leads to an error during compilation.
- To pass an error message to the calling block, you can use the BR bit. You first need to store the error message in the called block in the BR bit with the instruction "SAVE". You can then read the BR bit in the calling block.
- If you symbolically address (e.g. with the instruction L #myIn) a local formal parameter from the block interface in S7-1500, you always access the data block that you specified as an instance in the block call. Although the OPN DI, L AR2, +AR2, CDB, CAR instructions change the contents of the DI or address register, the registers are no longer evaluated when local formal parameters are addressed.

Load and transfer

During data exchange, the direction of the information flow is marked. If data is transferred from a memory area to accumulator 1, this is referred to as loading data. Data transfer from accumulator 1 to a memory area is referred to as "transfer". Load and transfer instructions are used for load and transfer in STL.

Load and transfer is necessary above all for editing digital values. If you want to add two values, for example, you have to load them to the buffer and execute the appropriate instruction. First, the first value to be added is loaded to accumulator 1. When the second summand is loaded, the first value, which is located in accumulator 1, is moved to accumulator 2. After execution of the add instruction (e.g. I+), the sum is saved in accumulator 1. The sum can be transferred to any operand using the transfer instruction.

All accumulators are 32 bits long and are organized byte-oriented. Data transfer between the memory areas and accumulator 1 may be byte-oriented, word-oriented or double word-oriented.

Data exchange between accumulators

Data can be exchanged between individual accumulators. The accumulator instructions "TAK", "PUSH" and "POP" are used for this purpose. This allows the transfer of data between the available accumulators and the exchange of bytes in Accumulator 1. Please note that the accumulator instructions "ENT" and "LEAVE" are then no longer available for the S7-1500.

See also

Parameter assignment via registers during block call in STL (Page 7247)

Load (Page 4788)

Transfer (Page 4797)

Accumulators (Page 4883)

15.1.3 Parameter passing via registers (S7-300, S7-400, S7-1500)

15.1.3.1 Parameter assignment via registers during block call in STL (S7-300, S7-400, S7-1500)

Block calls can be implemented using the following instructions in STL:

- CALL: Call block
- UC: Unconditional block call
- CC: Conditional block call

In a block call, values are usually transferred between the called and the calling block. The data exchange options are different for the various CPU families.
See also: Data exchange in STL (Page 7245)

Values transferred from the calling block to the called block

The following table indicates which registers, accumulators and status bits are directly available after a block call in the called block prior to processing the first network:

	S7-300/400		S7-1500	
	UC / CC	CALL	UC / CC with option "Parameter passing via registers"	CALL
Status bit OS	0	0	0	0
Status bit OV	W	W	W	0
Status bit CC0	W	W	W	0
Status bit CC1	W	W	W	0
BR	W	W	W	1
RLO	W	W	W	A
Accu 1	W	W	W	A

	S7-300/400		S7-1500	
Accu 2	W	W	W	A
Address register 1	W	W	W	0
Address register 2	W	Call FC: W Call FB: Instance offset	W	Call FC: 0 Call FB: Instance offset
DB register	W	W	W	A
DI register	W	Call FC: W Call FB: Instance DB number	W	Call FC: A Call FB: Instance DB number

Legend:
0 = Register is set to "0" at block change.
1 = Register is set to "1" at block change.
W = Register content can be set in the calling block and read in the called block.
A = Register content is not transferred to the called block. The contents are undefined, access results in an error during compilation.
Instance offset = Offset in instance DB
Instance DB number = Number of the instance DB

Values returned from the called block to the calling block

The following table indicates which registers, accumulators and status bits are directly available after processing of the block call in the calling block:

	S7-300/400		S7-1500	
	UC / CC	CALL	UC / CC with option "Parameter passing via registers"	CALL
Status bit OS	0	0	0	0
Status bit OV	W	W	W	-
Status bit CC0	W	W	W	-
Status bit CC1	W	W	W	-
BR	W	W	W	W
RLO	W	W1	W	A
Accu 1	W	W1	W	-
Accu 2	W	W1	W	-
Address register 1	W	W1	W	-
Address register 2	W	Call FC: W Call FB: -	W	-
DB register	-	X	-	-
DI register	-	X	-	-

Legend:
0 = Register is set to "0" at block change.
1 = Register is set to "1" at block change.
W = Register content can be set in the called block and read in the calling block.
W1 = The returned value can be changed by the output parameters and in/out parameters of the called block.
- = Register contents is not returned to the calling block, content is not changed by the block call.

A = Register content is not transferred to the called block. The content is undefined, access results in an error during compilation.
X = Register content is not returned to the calling block. The value can be changed implicitly by the parameters of the called block.

See also

Activating parameter passing via registers (Page 7249)

15.1.3.2 Activating parameter passing via registers (S7-1500)

Requirement

- The "Optimized block access" option is disabled.
- The block interface does not include any parameters.

Procedure

To enable parameter passing via registers, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Right-click the block for which you want to set up the parameter passing via registers.
3. Select the "Properties" command in the shortcut menu.
4. The properties dialog box of the block opens.
5. Click "Attributes" in the area navigation.
6. Select the option "Parameter passing via registers".
7. Confirm your entries with "OK".

See also

Parameter assignment via registers during block call in STL (Page 7247)

Data exchange in STL (Page 7245)

15.2 Settings for STL (S7-300, S7-400, S7-1500)

15.2.1 Overview of the settings for STL (S7-300, S7-400, S7-1500)

Overview

The following table shows the settings that you can make for STL:

Group	Setting	Description
View	Line break on Enter	Inserts a line break when you press the Enter key.

See also

Changing the settings (Page 7250)

15.2.2 Changing the settings (S7-300, S7-400, S7-1500)

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for STL (Page 7250)

15.3 Working with networks (S7-300, S7-400, S7-1500)

15.3.1 Using networks (S7-300, S7-400, S7-1500)

Function

The user program is created in the block within networks. For a code block to be programmed, it must contain at least one network. To achieve a better overview of the user program, you can also subdivide your program into several networks.

See also

Inserting networks (Page 7251)
Selecting networks (Page 7252)
Copying and pasting networks (Page 7253)
Deleting networks (Page 7253)
Expanding and collapsing networks (Page 7254)
Inserting network title (Page 7255)
Entering a network comment (Page 7256)
Navigating networks (Page 7257)

15.3.2 Inserting networks (S7-300, S7-400, S7-1500)

Requirement

A block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Result

A new empty network is inserted into the block.

See also

Using networks (Page 7251)
Selecting networks (Page 7252)
Copying and pasting networks (Page 7253)
Deleting networks (Page 7253)
Expanding and collapsing networks (Page 7254)
Inserting network title (Page 7255)
Entering a network comment (Page 7256)
Navigating networks (Page 7257)

15.3.3 Selecting networks (S7-300, S7-400, S7-1500)

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.
   The first and last networks and all those in between are selected.

See also

Using networks (Page 7251)
Inserting networks (Page 7251)
Copying and pasting networks (Page 7253)
Deleting networks (Page 7253)

Expanding and collapsing networks (Page 7254)
Inserting network title (Page 7255)
Entering a network comment (Page 7256)
Navigating networks (Page 7257)

15.3.4 Copying and pasting networks (S7-300, S7-400, S7-1500)

Copied networks can be pasted within the block or in another block. Networks that were created in LAD or FBD can also be inserted in blocks of the respective other programming language.

Requirement

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Using networks (Page 7251)
Inserting networks (Page 7251)
Selecting networks (Page 7252)
Deleting networks (Page 7253)
Expanding and collapsing networks (Page 7254)
Inserting network title (Page 7255)
Entering a network comment (Page 7256)
Navigating networks (Page 7257)

15.3.5 Deleting networks (S7-300, S7-400, S7-1500)

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Using networks (Page 7251)
Inserting networks (Page 7251)
Selecting networks (Page 7252)
Copying and pasting networks (Page 7253)
Expanding and collapsing networks (Page 7254)
Inserting network title (Page 7255)
Entering a network comment (Page 7256)
Navigating networks (Page 7257)

15.3.6 Expanding and collapsing networks (S7-300, S7-400, S7-1500)

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Using networks (Page 7251)
Inserting networks (Page 7251)
Selecting networks (Page 7252)

Copying and pasting networks (Page 7253)

Deleting networks (Page 7253)

Inserting network title (Page 7255)

Entering a network comment (Page 7256)

Navigating networks (Page 7257)

15.3.7 Inserting network title (S7-300, S7-400, S7-1500)

The network title is the header of a network. The length of the network title is limited to one line. You can enter the title manually or set it automatically. When you set it automatically, you can do this for individual networks or use the settings to specify that the network title is always set automatically.

For automatic insertion of the network title, the comment of the operand in one of the following instructions in the network is evaluated:

- Assignment
- Set output
- Reset output

The instruction that is listed first in the network is used.

The network title is only inserted automatically when the following conditions are fulfilled:

- The network does not have a title yet.
- The operand of the instruction used for the comment has a comment.

Note

Note the following restrictions for automatic insertion of the network title:

- The network title is not adapted if you change the comment of the operand at a later time.
- The network title is not adapted if you change the operand of the instruction.
- The network title is only set by the writing instructions listed above.
- If the operand is of the data type array, the comment of the array is used and not the comments of the array elements.
- Comments of invalid operands are not taken into consideration.

Entering the network title manually

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

Setting the network title automatically

To specify that the network titles are always set automatically, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Select the "Set network title automatically" check box in the "Additional settings" group.
   The network titles are set automatically as of this time if the conditions listed above are fulfilled.

To set an individual network title automatically, follow these steps:

1. Right-click "Network <Number of the network>" in the title bar of a network.
2. Select the "Set network title automatically" command in the shortcut menu.
   The title of the selected network is set based on the comment of the operand if the conditions listed above are fulfilled.

See also

Using networks (Page 7251)

Inserting networks (Page 7251)

Selecting networks (Page 7252)

Copying and pasting networks (Page 7253)

Deleting networks (Page 7253)

Expanding and collapsing networks (Page 7254)

Entering a network comment (Page 7256)

Navigating networks (Page 7257)

15.3.8 Entering a network comment (S7-300, S7-400, S7-1500)

You can use network comments to provide comments on the program contents of individual networks. For example, you can indicate the function of the network or draw attention to special characteristics.

Requirement

A network is available.

Procedure

To enter a network comment, follow these steps:

1. Click on the right arrow before the network title.
2. If the comment area is not visible, click "Network comments on/off" in the toolbar.
   The comment area is displayed.

3. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
4. Enter the network comment.

See also

Using networks (Page 7251)
Inserting networks (Page 7251)
Selecting networks (Page 7252)
Copying and pasting networks (Page 7253)
Deleting networks (Page 7253)
Expanding and collapsing networks (Page 7254)
Inserting network title (Page 7255)
Navigating networks (Page 7257)

15.3.9 Navigating networks (S7-300, S7-400, S7-1500)

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click in the code area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.
   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 7251)
Inserting networks (Page 7251)
Selecting networks (Page 7252)

Copying and pasting networks (Page 7253)
Deleting networks (Page 7253)
Expanding and collapsing networks (Page 7254)
Inserting network title (Page 7255)
Entering a network comment (Page 7256)

15.4 Inserting STL instructions (S7-300, S7-400, S7-1500)

15.4.1 Rules for STL (S7-300, S7-400, S7-1500)

Rules

The following rules must be observed when entering STL instructions incrementally:

- An instruction consists of the specification of an optional jump label, the operation, the operand, and an optional comment. Example:
  `M001: A I1.0 //comment`
- Each instruction is in its own separate line.
- You can enter a maximum of 999 networks per block.
- The entry of operations or absolute addresses is not case-sensitive.
- If you would like support for block calls, you must program the called block before the calling block.

Impermissible instructions are highlighted in red. A rollout and the inspector window give information for troubleshooting.

Jumps

For jump operations, make sure that the jump destination is always at the start of a logic sequence and is within a different nesting level (not applicable to CPU 318-2). The jump designation must not be located within a logic sequence or a nesting level.

See also

Entering STL instructions manually (Page 7262)

Inserting STL instructions using the "Instructions" task card (Page 7263)

Inserting comments (Page 7275)

Inserting operands (Page 7265)

Using favorites in STL (Page 7268)

Inserting block calls in STL (Page 7270)

15.4.2 Instructions from the global library "Long Functions" (S7-300, S7-400, S7-1500)

Description

The table below shows the instructions included in the global library "Long Functions":

Instruction	Explanation
ABS_LINT	Forming absolute value with the LINT data type
ABS_LREAL	Forming absolute value with the LREAL data type
ACOS_LREAL	Forming arccosine value with the LREAL data type
ADD_LINT	Adding with the LINT data type
ADD_LREAL	Adding with the LREAL data type
ADD_ULINT	Adding with the ULINT data type
AND_LWORD	AND logical operation with the LWORD data type
ASIN_LREAL	Forming arccosine value with the LREAL data type
ATAN_LREAL	Forming arctangent value with the LREAL data type
COS_LREAL	Forming cosine value with the LREAL data type
DINT_to_LINT	Data type conversion from DINT to LINT
DIV_LINT	Dividing with the LINT data type
DIV_LREAL	Dividing with the LREAL data type
DIV_ULINT	Dividing with the ULINT data type
EQ_LINT	Compare for equivalence with the LINT data type
EQ_LREAL	Compare for equivalence with the LREAL data type
EQ_LWORD	Compare for equivalence with the LWORD data type
EQ_ULINT	Compare for equivalence with the ULINT data type
EXP_LREAL	Forming exponential value with the LREAL data type
GE_LINT	Compare for greater than or equal to with the LINT data type
GE_LREAL	Compare for greater than or equal to with the LREAL data type
GE_ULINT	Compare for greater than or equal to with the ULINT data type
GT_LINT	Compare for greater than with the LINT data type
GT_LREAL	Compare for greater than with the LREAL data type
GT_ULINT	Compare for greater than with the ULINT data type
LE_LINT	Compare for less than or equal to with the LINT data type
LE_LREAL	Compare for less than or equal to with the LREAL data type
LE_ULINT	Compare for less than or equal to with the ULINT data type
LINT_to_DINT	Data type conversion from LINT to DINT

Instruction	Explanation
LINT_to_LREAL	Data type conversion from LINT to LREAL
LINT_to_ULINT	Data type conversion from LINT to ULINT
LN_LREAL	Forming natural logarithm with the LREAL data type
LREAL_to_LINT	Data type conversion from LREAL to LINT
LREAL_to_REAL	Data type conversion from LREAL to REAL
LREAL_to_ULINT	Data type conversion from LREAL to ULINT
LT_LINT	Compare for less than with the LINT data type
LT_LREAL	Compare for less than with the LREAL data type
LT_ULINT	Compare for less than with the ULINT data type
LWORD_to_ULINT	Data type conversion from LWORD to ULINT
LWORD_to_WORD	Data type conversion from LWORD to WORD
MOVE_LINT	Move value with the LINT data type
MOVE_LREAL	Move value with the LREAL data type
MOVE_LWORD	Move value with the LWORD data type
MOVE_ULINT	Move value with the ULINT data type
MUL_LINT	Multiply by the LINT data type
MUL_LREAL	Multiply by the LREAL data type
MUL_ULINT	Multiply by the ULINT data type
NE_LINT	Compare for non-equivalence with the LINT data type
NE_LREAL	Compare for non-equivalence with the LREAL data type
NE_LWORD	Compare for non-equivalence with the LWORD data type
NE_ULINT	Compare for non-equivalence with the ULINT data type
NEG_LINT	Create two's complement with the LINT data type
NEG_LREAL	Create two's complement with the LREAL data type
OR_LWORD	OR logical operation with the LWORD data type
RDN_LREAL	Round number with the LREAL data type
REAL_to_LREAL	Data type conversion from REAL to LREAL
SIN_LREAL	Form sine value with the LREAL data type
SQR_LREAL	Form square with the LREAL data type
SQRT_LREAL	Form square root with the LREAL data type
SUB_LINT	Subtract with the LINT data type
SUB_LREAL	Subtract with the LREAL data type
SUB_ULINT	Subtract with the ULINT data type
TAN_LREAL	Form tangent value with the LREAL data type
TRUNC_LREAL	Create integer with the LREAL data type
UDINT_to_ULINT	Data type conversion from UDINT to ULINT
ULINT_to_LINT	Data type conversion from ULINT to LINT
ULINT_to_LREAL	Data type conversion from ULINT to LREAL

Instruction	Explanation
ULINT_to_LWORD	Data type conversion from ULINT to LWORD
ULINT_to_UDINT	Data type conversion from ULINT to UDINT
WORD_to_LWORD	Data type conversion from WORD to LWORD
XOR_LWORD	EXCLUSIVE OR logical operation with the LWORD data type

See also

Entering STL instructions manually (Page 7262)

Inserting STL instructions using the "Instructions" task card (Page 7263)

Inserting operands (Page 7265)

15.4.3 Entering STL instructions manually (S7-300, S7-400, S7-1500)

When an empty network is created automatically or manually, a two-line entry field is generated within the network. The STL instructions are entered in this entry field.

Note

S7-1500: Instructions from the "Instructions" task card that use emulated registers and accumulators cannot use 64-bit tags, such as LInt, as operands. If you want to use 64-bit tags, use the instructions from the "Long Functions" global library.

Requirement

A network is available.

Procedure

To enter STL instructions in a network, follow these steps:

1. Click the first empty row of the entry field in a network.
2. Enter an instruction.
3. Enter a space.
4. Enter the operand.

   Note

   You can also use a drag-and-drop operation to insert an already defined operand from the PLC tag table or from the detail view.

5. Enter a line comment. This step is optional.
6. Confirm the instruction with the Enter key.
7. Repeat steps 2 to 6 until you have entered all the required instructions.

Result

After the instruction is closed by pressing the Enter key, the program editor performs the following actions:

- Syntax check: After the check, faulty entries are displayed in red italics. In addition, you also receive a detailed error message in the "Syntax" tab of the information window.
- Change of lower and upper case: Instructions are changed to upper case.
- If you have entered absolute addresses in symbolic programming, these will be converted into symbolic names. If no symbolic name is available, "Tag_< number>" is inserted.
- Block call: the necessary parameters are inserted and displayed.
- An additional line is inserted in the entry field.

See also

Instructions from the global library "Long Functions" (Page 7260)

Rules for STL (Page 7259)

Inserting STL instructions using the "Instructions" task card (Page 7263)

Inserting operands (Page 7265)

Using favorites in STL (Page 7268)

Inserting block calls in STL (Page 7270)

Inserting comments (Page 7275)

15.4.4 Inserting STL instructions using the "Instructions" task card (S7-300, S7-400, S7-1500)

When an empty network is created automatically or manually, a two-line entry field is generated within the network. The STL instructions are entered in this entry field.

Note

S7-1500: Instructions from the "Instructions" task card that use emulated registers and accumulators cannot use 64-bit tags, such as LInt, as operands. If you want to use 64-bit tags, use the instructions from the "Long Functions" global library.

Requirement

A network is available.

Procedure

To insert STL instructions into a network using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. To insert the instruction, select one of the following steps:
   – Go to the STL element you want to insert and drag it to the required location in the network.
   – Select the location in the network at which you want to insert the element and then double-click on the element you want to insert.
3. Enter the required operands.

   Note

   You can also use a drag-and-drop operation to insert an already defined operand from the PLC tag table, a data block, the detail view or the block interface.

4. Enter a line comment. This step is optional.
5. Repeat steps 2 to 4 until you have entered all the required instructions.

Result

After the instruction is closed by pressing the Enter key, the program editor performs the following actions:

- Syntax check: After the check, faulty entries are displayed in red italics. In addition, you also receive a detailed error message in the information window.
- Change of lower and upper case: Instructions are changed to upper case.
- If you have entered absolute addresses in symbolic programming, these will be converted into symbolic names. If no symbolic name is available, "Tag_< number>" is inserted.
- Block call: the necessary parameters are inserted and displayed.
- An additional line is inserted in the entry field.

See also

Instructions from the global library "Long Functions" (Page 7260)

Rules for STL (Page 7259)

Entering STL instructions manually (Page 7262)

Inserting operands (Page 7265)

Using favorites in STL (Page 7268)

Inserting block calls in STL (Page 7270)

Inserting comments (Page 7275)

15.4.5 Selecting the data type of an STL instruction (S7-300, S7-400, S7-1500)

15.4.5.1 Selecting a data type (S7-300, S7-400, S7-1500)

Introduction

Some instructions can be executed with several different data types. If you use one of these instructions in the program, you have to specify a valid data type for the instruction at the specific point in the program.

The valid data types for an instruction are listed in the instruction drop-down list. You specify the data type of the instruction by selecting an entry from the drop-down list. If the data type of an operand differs from the data type of the instruction and cannot be converted implicitly, the operand is displayed in red and a rollout with the corresponding error message appears.

15.4.5.2 Defining the data type of an instruction (S7-300, S7-400, S7-1500)

Introduction

Some instructions can be executed with several different data types. When you insert such instructions into your program, you must specify the data type for these instructions at the actual point in the program.

Procedure

To define the data type of an instruction using the drop-down list, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Click the triangle in the upper corner of the drop-down list.
   The drop-down list opens to display the data types valid for the instruction.
3. Select a data type from the drop-down list.
   The selected data type is displayed.

15.4.6 Inserting operands (S7-300, S7-400, S7-1500)

You can enter the operands of an STL instruction either manually or by dragging them from the PLC tag table, a data block, the detail view, or the block interface.

Note

S7-1500: Instructions from the "Instructions" task card that use emulated registers and accumulators cannot use 64-bit tags, such as LInt, as operands. If you want to use 64-bit tags, use the instructions from the "Long Functions" global library.

Procedure

To enter the operand of an STL instruction manually, follow these steps:

1. Position the cursor to the right of the operation.
2. Enter a space and then the operand.

   Note

   If you enter the absolute address of a parameter that has already been defined in symbolic programming, this absolute address will be changed to the symbolic name of the parameter as you confirm your input. If you have not yet defined the parameter, a new tag with this absolute address and the default name "Tag_1" will be entered in the PLC tag table. When you confirm your input, the absolute address will be replaced with the symbolic name "Tag_1".

3. Confirm the parameter with the Enter key.
4. If you have not yet defined the parameter, you can define it directly in the program editor using the shortcut menu.
   See also: "Declaring tags directly in the program editor (Page 7080)"

Or drag from it the PLC tag table:

1. In the project tree, select the "PLC tags" folder or open the PLC tag table.
2. If you have opened the PLC tag table, drag the symbol from the first column of the desired tag to the appropriate place in your program. If you have not opened the PLC tag table yet, open the detail view now. Drag the desired tag from the detail view to the appropriate place in your program.

Or drag from it the block interface:

1. Open the block interface.
2. Drag the required operand from the block interface to the instruction window.

See also

Instructions from the global library "Long Functions" (Page 7260)

Rules for STL (Page 7259)

Entering STL instructions manually (Page 7262)

Inserting STL instructions using the "Instructions" task card (Page 7263)

Using favorites in STL (Page 7268)

Inserting block calls in STL (Page 7270)

Inserting comments (Page 7275)

Using and addressing operands (Page 89)

Using autocompletion (Page 6859)

15.4.7 Displaying or hiding tag information (S7-300, S7-400, S7-1500)

Introduction

You can hide or show simple or hierarchical comments on the documentation of the tags, regardless of the absolute or symbolic representation of the operands. For local tags and DB tags, this information is obtained from the block interface. For tags that are valid for the entire CPU, the information is obtained from the PLC tag table.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all blocks currently opened and opened in future is shown.

You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual ones that you have opened.

If you select the display of tag information with hierarchical comments, the comments of the higher structure levels of structured tags will also be displayed. The display is in brackets after the comment of the tags; the comments of the individual levels are separated by a period. If there is no comment at a structure level for a tag, it is omitted in the display and this is recognizable because there are two periods.

Displaying or hiding tag information for all blocks

Follow the steps below to display or hide the tag information for all blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to show the tag information, either select the "Expand" option in the "Tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
4. If you want to hide the tag information, select the "Collapse" option in the "Tag information" drop-down list.
   The tag information is displayed or hidden for all blocks. When you open further blocks, the tag information is displayed or hidden depending on the selected setting.

Displaying or hiding tag information for an opened block

Follow the steps below to display or hide the tag information for an opened block:

1. If you want to show the tag information, either select the "Show tag information" option in the "Shows the tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
2. If you want to hide the tag information, select the "Hide tag information" option in the "Hides tag information" drop-down list.
   The tag information is displayed or hidden.

15.4.8 Using favorites in STL (S7-300, S7-400, S7-1500)

15.4.8.1 Adding STL elements to Favorites (S7-300, S7-400, S7-1500)

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Inserting STL instructions using favorites (Page 7268)

Removing STL instructions from Favorites (Page 7269)

Overview of the program editor (Page 6831)

15.4.8.2 Inserting STL instructions using favorites (S7-300, S7-400, S7-1500)

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 6831)

Adding STL elements to Favorites (Page 7268)

Entering STL instructions manually (Page 7262)

Inserting STL instructions using the "Instructions" task card (Page 7263)

Removing STL instructions from Favorites (Page 7269)

15.4.8.3 Removing STL instructions from Favorites (S7-300, S7-400, S7-1500)

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding STL elements to Favorites (Page 7268)

Inserting STL instructions using favorites (Page 7268)

Overview of the program editor (Page 6831)

15.4.9 Inserting block calls in STL (S7-300, S7-400, S7-1500)

15.4.9.1 Manually inserting block calls (S7-300, S7-400, S7-1500)

You can insert calls for functions (FCs) and function blocks (FBs). Function blocks can be called either as a single, multi or parameter instance.

See also: Basics of instances (Page 62)

Requirement

A network is available.

Inserting a call for a function (FC)

To insert a function call in an STL network, follow these steps:

1. Enter "CALL <Name of the function to be called>".
   Replace "<Name of the function to be called>" with the name of the function that you want to call.
2. Confirm your entry with the Enter key.
   - If the function already exists, all parameters of the function are inserted in the call.
   - If the function does not already exist, you must enter the parameters yourself.

Inserting a call for a function block (FB)

To insert the call for a function block (FB), follow these steps:

1. Enter "CALL <Name of the function block to be called>[, <Name of the instance data block>]".
   Replace "<Name of the function to be called>" with the name of the function block that you want to call. The instance data block entry is optional.
2. Confirm your entry with the Enter key.
   If you have not specified the instance data block or the instance data block you have specified does not exist, the "Call options" dialog is opened. In this case, perform steps 3 and 4.

3. In the dialog, enter whether you want to call the block as a single, multi or parameter instance.
   - If you click the "Single instance" button, enter a name for the data block that will be assigned to the call in the entry field "Name".
   - If you click the "Multi instance" button, enter the name of the tag in the "Name in the interface" entry field, with which the called function block will be entered as a static tag in the interface of the calling block.
   - If you click on the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
4. If you call a block that contains monitoring, assign a ProDiag function block to the monitoring functions.
5. Confirm your entries with "OK".

Result

When you call existing functions or function blocks with an instance data block, the parameters of the function or function block are inserted into the call. You must then assign these parameters.

See also: Parameter transfer at block call (Page 76)

If when calling a function block you specify an instance data block that does not exist, it will be created.

See also

Inserting block calls using a drag-and-drop operation (Page 7271)

Updating block calls in STL (Page 7272)

Changing the instance type (Page 7274)

Parameter assignment via registers during block call in STL (Page 7247)

15.4.9.2 Inserting block calls using a drag-and-drop operation (S7-300, S7-400, S7-1500)

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree. Function blocks can be called either as a single, multi or parameter instance.

See also: Basics of instances (Page 62)

Requirement

- A network is available.
- The function to be called (FC) or the function block (FB) to be called is present.

Inserting a call for a function (FC)

To insert a function call in an STL network using a drag-and-drop operation, follow these steps:

1. Drag the function from the project tree into the network.

Inserting a call for a function block (FB)

To insert the call for a function block (FB), follow these steps:

1. Drag the function block from the project tree into the network.
   The "Call options" dialog opens.
2. In the dialog, enter whether you want to call the block as a single, multi or parameter instance.
   - If you click the "Single instance" button, enter a name for the instance data block to be assigned to the call in the entry field "Name".
   - If you click the "Multi-instance" button, in the "Name in the interface" box, enter a name of the tag with which the called function block will be entered as a static tag in the interface of the calling block.
   - If you click on the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
3. If you call a block that contains monitoring, assign a ProDiag function block to the monitoring functions.
4. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You must then assign these parameters.

See also: Parameter transfer at block call (Page 76)

If when calling a function block you specify an instance data block that does not exist, it will be created.

See also

Manually inserting block calls (Page 7270)

Updating block calls in STL (Page 7272)

Changing the instance type (Page 7274)

15.4.9.3 Updating block calls in STL (S7-300, S7-400, S7-1500)

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have the following options for updating the block calls:

- Explicit updating of all inconsistent block calls in the program editor.
  The inconsistent block calls within the open block are updated. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are removed if they are not connected.
  - Renamed parameters get the new parameter names.
- Explicit updating of a block call in the program editor.
  The "Interface update" dialog is displayed. In this dialog, you have the option of changing the connection of the operands of the new interface. The inconsistent call of this block is then updated. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are removed if they are not connected.
  - Renamed parameters get the new parameter names.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated. Note that when you call functions (FCs) before the next compilation process, all new formal parameters must be supplied with actual parameters.

Updating all inconsistent block calls in the program editor

To open all block calls in a block, follow these steps:

1. Open the calling block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

Updating a specific block call in the program editor

To update a specific block call in the program editor, follow these steps:

1. Open the calling block in the program editor.
2. Right-click on the block call that you want to update.
3. Select the "Update" command in the shortcut menu.
   The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
4. If necessary, change the connection of the operands. To do this, you have the following options:
   - You can use either a drag-and-drop operation or a cut/copy-and-paste operation to move the operand from the old interface to the new interface.
   - You can delete an operand.
   - You can rename an operand.
   - You can specify a new operand via autocompletion.
5. Click "OK" to update the block call. If you want to cancel the update, click "Cancel".

Note

Note that the "Update block call" command is only available provided you did not previously update all block calls in the editor with the "Update inconsistent block calls" command.

Update block calls during compilation

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Manually inserting block calls (Page 7270)

Inserting block calls using a drag-and-drop operation (Page 7271)

Changing the instance type (Page 7274)

Block calls (Page 60)

15.4.9.4 Changing the instance type (S7-300, S7-400, S7-1500)

Instance type

There are two ways of calling function blocks:

- As a single instance
- As a multiple instance
- As parameter instance

See also: Basics of instances (Page 62)

You can modify a defined instance type at any time.

Requirement

The user program contains a block call.

Procedure

To change the instance type of a function block, follow these steps:

1. Open the code block and select the block call.
2. Select the "Change instance" command in the shortcut menu.
   The "Call options" dialog opens.
3. Click on the button "Single instance", "Multi-instance" or "Parameter instance".
   - If you select the "Single instance" instance type, enter a name for the data block that is to be assigned to the function block.
   - If you select "Multiple instance" as the instance type, enter in the "Name in the interface" text field the name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
   - If you select "Parameter instance" as the instance type, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
4. Confirm your entries with "OK".

Note

The previous instances are not deleted automatically.

See also

Manually inserting block calls (Page 7270)

Inserting block calls using a drag-and-drop operation (Page 7271)

Updating block calls in STL (Page 7272)

Instances (Page 62)

15.4.10 Inserting comments (S7-300, S7-400, S7-1500)

Commenting program code

Various options are available for commenting STL programs:

- Line comment
  A line comment starts with "//" and extends to the end of the line.
- Comment section within a line
  A comment section is started with "(* and completed by "*)". A comment section can be located at any point in the line, but cannot extend over several lines.

Requirement

A network is available.

Inserting line comments

To insert a line comment into an instruction line, follow these steps:

1. Enter "//" at the end of the instruction line.
2. Enter the comment.

Inserting a comment section within a line

Proceed as follows to insert a comment section within an instruction line:

1. Enter a "(*" at any point in the instruction line.
2. Enter the comment after "(*".
3. Enter "*)" after the comment.

Disabling one or more lines with comments

To disable program code with with comments, follow these steps:

1. Select the code lines you want to comment out.
2. Click the "Comment selection" button in the editor.
3. "//" is inserted at the beginning of the line in the selected lines. The code that follows is interpreted as a comment. If lines already containing a line comment are disabled, "//" is inserted as well. If the lines are then enabled again, original comments are retained.

Enabling comment lines

To enable lines that have been commented out to be enabled as code again, proceed as follows:

1. Select the code lines you want to enable.
2. Click the "Remove comment" button in the editor.
3. The "//" mark for line comments at the beginning of the line is removed.

Result

Comments are formatted with highlighting.

See also

Rules for STL (Page 7259)

Entering STL instructions manually (Page 7262)

Inserting STL instructions using the "Instructions" task card (Page 7263)

Inserting operands (Page 7265)

Using favorites in STL (Page 7268)

Inserting block calls in STL (Page 7270)

15.5 Editing STL instructions (S7-300, S7-400, S7-1500)

15.5.1 Selecting STL instructions (S7-300, S7-400, S7-1500)

You can select individual instructions or all the instructions of a network.

Requirement

STL instructions are available.

Selecting individual STL instructions

To select STL instructions, follow these steps:

1. Set the insertion mark before the first character that you want to select.
2. Press and hold down the left mouse button.
3. Move the cursor to a position after the last character that you want to select.
4. Release the left mouse button.

Selecting all STL instructions of a network

To select all instructions of a network, follow these steps:

1. Click in the network whose instructions you want to select.
2. Select "Select All" from the "Edit" menu or use the keyboard shortcut <Ctrl+A>.

See also

Copying STL instructions (Page 7277)
Cutting STL instructions (Page 7278)
Pasting STL instructions from the Clipboard (Page 7279)
Deleting STL instructions (Page 7279)

15.5.2 Copying STL instructions (S7-300, S7-400, S7-1500)

Requirement

An STL instruction is available.

Procedure

To copy an STL instruction, follow these steps:

1. Select the STL instruction you want to copy.
2. Select "Copy" in the shortcut menu.

Result

The STL instruction is copied and saved to the clipboard.

See also

Selecting STL instructions (Page 7277)
Cutting STL instructions (Page 7278)
Pasting STL instructions from the Clipboard (Page 7279)
Deleting STL instructions (Page 7279)

15.5.3 Cutting STL instructions (S7-300, S7-400, S7-1500)

Requirement

An STL instruction is available.

Procedure

To cut an STL instruction, follow these steps:

1. Select the STL instruction you want to cut.
2. Select "Cut" in the shortcut menu.

Result

The STL instruction is cut and saved to the clipboard.

See also

Selecting STL instructions (Page 7277)
Copying STL instructions (Page 7277)
Pasting STL instructions from the Clipboard (Page 7279)
Deleting STL instructions (Page 7279)

15.5.4 Pasting STL instructions from the Clipboard (S7-300, S7-400, S7-1500)

Requirement

An STL instruction is available.

Procedure

To paste an STL instruction from the Clipboard, follow these steps:

1. Copy an STL instruction or cut an STL instruction.
2. Click the place in the network where you want to insert the STL instruction.
3. Select "Paste" in the shortcut menu.

See also

Selecting STL instructions (Page 7277)
Copying STL instructions (Page 7277)
Cutting STL instructions (Page 7278)
Deleting STL instructions (Page 7279)

15.5.5 Deleting STL instructions (S7-300, S7-400, S7-1500)

Requirement

An STL instruction is available.

Procedure

To delete an STL instruction, follow these steps:

1. Select the STL instruction you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Selecting STL instructions (Page 7277)
Copying STL instructions (Page 7277)
Cutting STL instructions (Page 7278)
Pasting STL instructions from the Clipboard (Page 7279)

15.6 STL programming examples (S7-300, S7-400, S7-1500)

15.6.1 Example of controlling a conveyor belt (S7-300, S7-400, S7-1500)

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbuttons at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbuttons at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.

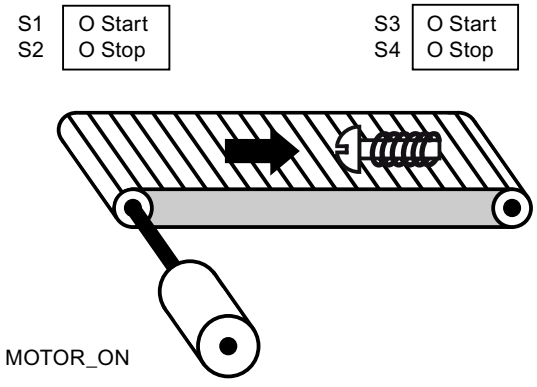

Implementation

The following table shows the definition of the tags used:

Operand	Declaration	Data type	Description
StartPushbutton_Left (S1)	Input	BOOL	Start pushbutton on the left side of the conveyor belt
StopPushbutton_Left (S2)	Input	BOOL	Stop pushbutton on the left side of the conveyor belt
StartPushbutton_Right (S3)	Input	BOOL	Start pushbutton on the right side of the convey-or belt
StopPushbutton_Right (S4)	Input	BOOL	Stop pushbutton on the right side of the convey-or belt
MOTOR_ON	Output	BOOL	Turn on the conveyor belt motor

The following STL program shows how to implement this task:

```
STL                 Explanation
O #S1               // Scan start pushbutton S1 for "1".
```

STL	Explanation
O #S3	// Scan start pushbutton S3 for "1".
S #MOTOR_ON	// If one of the start pushbuttons (S1 or S3) returns the signal state "1", the conveyor belt motor is switched on.
O #S2	// Scan stop pushbutton S2 for "1".
O #S4	// Scan stop pushbutton S4 for "1".
R #MOTOR_ON	// If one of the stop pushbuttons (S2 or S4) returns the signal state "1", the conveyor belt motor is switched off.

See also

STL Basics (Page 7245)
Settings for STL (Page 7250)
Working with networks (Page 7251)
Inserting STL instructions (Page 7259)
Editing STL instructions (Page 7277)
Example of detecting the direction of a conveyor belt (Page 7282)
Example of detecting the fill level of a storage area (Page 7284)
Example of calculating an equation (Page 7286)
Example for heating an oven (Page 7288)
Example of a step sequence (Page 7290)

15.6.2 Example of detecting the direction of a conveyor belt (S7-300, S7-400, S7-1500)

Detecting the direction of a conveyor belt

The detected running direction of the belt is indicated by a RIGHT arrow or a LEFT arrow. If additional conveyed material is approaching PEB1 from the right or PEB2 from the left, the displayed arrow is initially switched off until, after both photoelectric barriers are passed, the running direction can be detected again and the corresponding arrow displayed. For the solution of the task, 2 edge memory bits are needed that detect the signal change from "0" to "1" at the two photoelectric barriers.

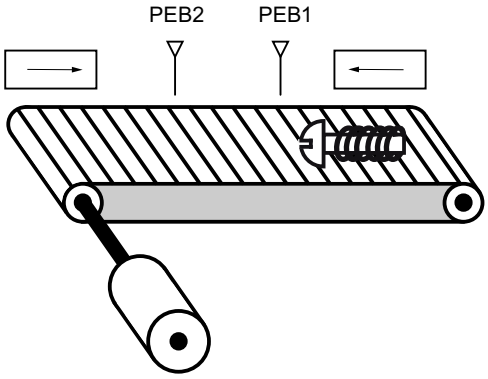

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
S1	Input	BOOL	Photoelectric barrier 1
S2	Input	BOOL	Photoelectric barrier 2
TM1	Input	BOOL	Edge bit memory 1
TM2	Input	BOOL	Edge bit memory 2
RIGHT	Output	BOOL	Display for movement to the right
LEFT	Output	BOOL	Display for movement to the left

The following STL program shows how to implement this example:

```
STL                 Explanation
A #S1               // Scan photoelectric barrier "S1" for "1"
FP #TM1             // Query positive edge
AN #S2              // Scan photoelectric barrier "S2" for "0"
S #LEFT             //If the signal state changes from "0" to "1" (positive edge)
                    at the photoelectric barrier "S1" and, at the same time, the
                    signal state at the photoelectric barrier "S2" is "0", the ob-
                    ject on the belt is moving to the left.
                    // The display for movement to the left is activated.
A #S2               // Scan photoelectric barrier "S2" for "1"
```

STL	Explanation
FP #TM2	// Query positive edge
AN #S1	// Scan photoelectric barrier "S1" for "0"
S #RIGHT	//If the signal state changes from "0" to "1" (positive edge) at photoelectric barrier "S2" and, at the same time, the signal state at photoelectric barrier "S1" is "0", the object on the belt is moving to the right.
	// The display for movement to the right is activated.
AN #S1	// Scan photoelectric barrier "S1" for "0"
AN #S2	// Scan photoelectric barrier "S2" for "0"
R #LEFT	// The display for movement to the left will be turned off when the signal state at both photoelectric barriers is "0".
R #RIGHT	// The display for movement to the right will be turned off when the signal state at both photoelectric barriers is "0".

See also

STL Basics (Page 7245)
Settings for STL (Page 7250)
Working with networks (Page 7251)
Inserting STL instructions (Page 7259)
Editing STL instructions (Page 7277)
Example of controlling a conveyor belt (Page 7280)
Example of detecting the fill level of a storage area (Page 7284)
Example of calculating an equation (Page 7286)
Example for heating an oven (Page 7288)
Example of a step sequence (Page 7290)

15.6.3 Example of detecting the fill level of a storage area (S7-300, S7-400, S7-1500)

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock where they are loaded onto trucks. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

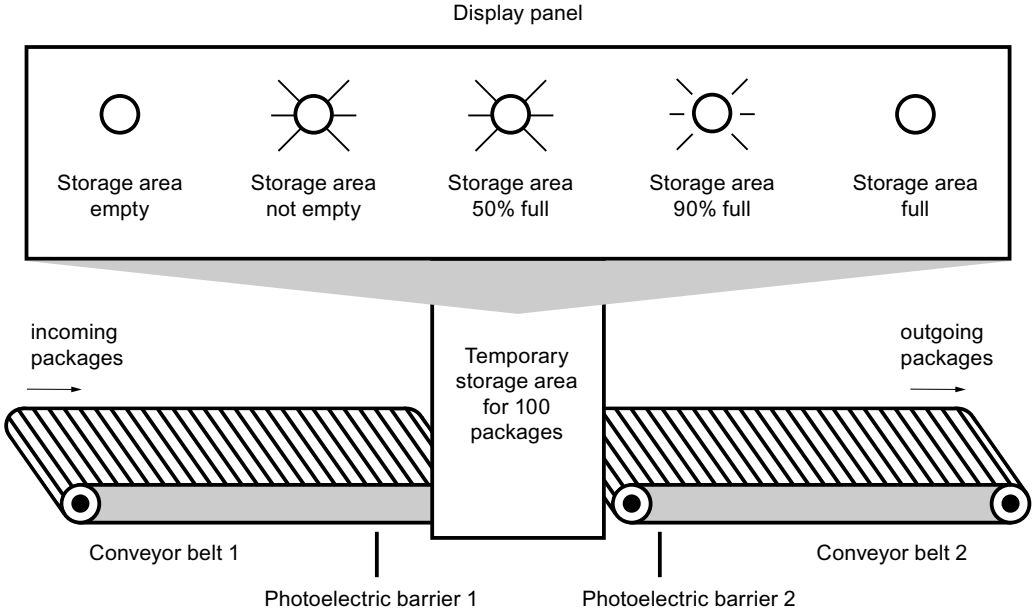

Implementation

The following table shows the definition of the tags used:

Name	Data type	Address	Description
PACKAGECOUNT	COUNTER	C1	Number of packages in the storage area (current count value)

Name	Section	Data type	Description
LS1	Input	BOOL	Photoelectric barrier 1
LS2	Input	BOOL	Photoelectric barrier 2
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty

Name	Section	Data type	Description
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50% full
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90% full
STOR_FULL	Output	BOOL	Display lamp: Storage area full

The following STL program shows how to implement this example:

```
STL                 Explanation
A #LS1              // Scan photoelectric barrier "LS1" for "1".
CU "PACKAGECOUNT"   // In the event of a positive signal edge at photoelectric bar-
                    rier "LS1", the count value of counter "PACKAGECOUNT" is in-
                    creased by one.
A #LS2              // Scan photoelectric barrier "LS2" for "1".
CD "PACKAGECOUNT"   // In the event of a positive signal edge at photoelectric bar-
                    rier "LS2", the count value of counter "PACKAGECOUNT" is de-
                    creased by one.
AN "PACKAGECOUNT"   // Scan count value for "0".
= #STOR_EMPTY       // With a count value of "0" the display lamp "storage area emp-
                    ty" is switched on.
A "PACKAGECOUNT"    // Scan count value for "1".
= #STOR_NOT_EMPTY   // With a count value greater than "0" the display lamp "Storage
                    area not empty" is switched on.
L 50                // Load the comparison value "50" to accumulator 1.
L "PACKAGECOUNT"    // Move the comparison value to accumulator 2.
                    // Load the current count value to accumulator 1.
<=I                 // Compare values
= #"STOR_50%_FULL"  // With a count value greater than or equal to "50" the display
                    lamp "Storage area 50% full" is switched on.
L 90                // Move the counter value to accumulator 2.
                    // Load the comparison value "90" to accumulator 1.
>=I                 // Compare values
= #"STOR_90%_FULL"  // With a count value greater than or equal to "90" the display
                    lamp "Storage area 90% full" is switched on.
L "PACKAGECOUNT"    // Load the current count value to accumulator 1.
L 100               // Move the counter value to accumulator 2.
                    // Load the comparison value "100" to accumulator 1.
>=I                 // Compare values
= #STOR_FULL        // At a count value greater than "100" the display lamp "Storage
                    area full" is switched on.
```

See also

STL Basics (Page 7245)

Settings for STL (Page 7250)

Working with networks (Page 7251)

Inserting STL instructions (Page 7259)
Editing STL instructions (Page 7277)
Example of controlling a conveyor belt (Page 7280)
Example of detecting the direction of a conveyor belt (Page 7282)
Example of calculating an equation (Page 7286)
Example for heating an oven (Page 7288)
Example of a step sequence (Page 7290)

15.6.4 Example of calculating an equation (S7-300, S7-400, S7-1500)

Calculating an equation

The sample program shows you how to use three math instructions to calculate the following equation:

RESULT = ((A + B) x C) /D

Implementation

The following table shows the declaration of the operands used in the PLC tag table:

Name	Data type	Comment
A	INT	First value for addition
B	INT	Second value for addition
C	INT	Multiplier
D	INT	Divisor
RESULT	INT	End result

The following STL program shows how to implement this example:

```
 STL                 Explanation
L "A"                // Load value of the operand "A" to accumulator 1
L "B"                // Load value of operand "A" to accumulator 2
                     // Load value of the operand "B" to accumulator 1
+I                   // Add the values of accumulator 1 and 2
                     // Store sum in accumulator 1
L "C"                // Move the sum to accumulator 2
                     // Load value of the operand "C" to accumulator 1
*I                   // Multiply the values of accumulator 1 and 2
                     // Store product in accumulator 1
L "D"                // Move the product to accumulator 2
                     // Load value of the operand "D" to accumulator 1
```

STL	Explanation
/I	// Divide the value of accumulator 2 by the value of accumulator 1
	// Store result in accumulator 1
T "RESULT"	// Transfer the result to operand "RESULT"

See also

STL Basics (Page 7245)
Settings for STL (Page 7250)
Working with networks (Page 7251)
Inserting STL instructions (Page 7259)
Editing STL instructions (Page 7277)
Example of controlling a conveyor belt (Page 7280)
Example of detecting the direction of a conveyor belt (Page 7282)
Example of detecting the fill level of a storage area (Page 7284)
Example for heating an oven (Page 7288)
Example of a step sequence (Page 7290)

15.6.5 Example for heating an oven (S7-300, S7-400, S7-1500)

Heating an oven

The following illustration shows an oven that is turned on with a start switch. The heating process is started when the start switch is pressed. The heating period is set with digital thumbwheels. The heating period is set in seconds using the BCD format.

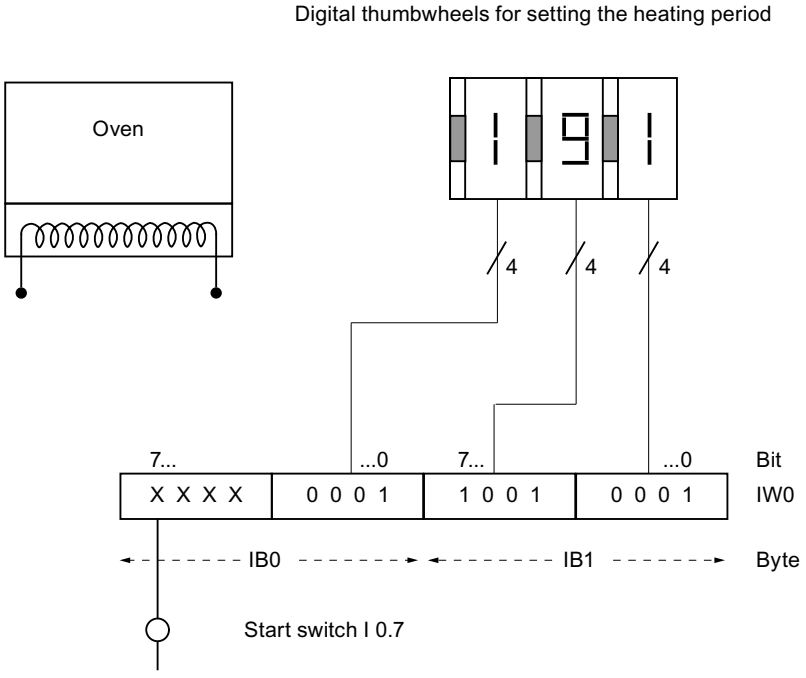

Implementation

The following table shows the declaration of the operands used in the PLC tag table:

Name	Data type	Address	Comment
DURATION	WORD	EW0	Heating period in seconds • I1.0 to I1.3: Thumbwheels for ones • I1.4 to I1.7: Thumbwheels for tens • I0.0 to I0.3: Thumbwheels for hundreds
HEATING	TIMER	T1	Time that is started with the preset heating period.

The following table shows the declaration of the operands used in the block interface of the code block:

Name	Section	Data type	Comment
START	Input	BOOL	Start switch
START_HEATING	Output	BOOL	Start of the heating process

The following STL program shows how to implement this example:

```
 STL                 Explanation
A "HEATING"          // Scan to see if time has started
= #START_HEATING     // Start the heating process
BEC                  // For RLO=1 End processing of block
                     // This prevents the "HEATING" time from being restarted if the
                     button is pressed
L "DURATION"         // Load the heating period in accumulator 1
AW W#16#0FFF         // Reset input bits I0.4 to I0.7 to "0"
OW W#16#4000         // Set seconds in bits I1.2 and I1.3 of accumulator 1
A #START             // Scan start switch for "1"
SE "HEATING"         // Start heating process as an extended pulse upon a positive
                     signal edge at the start switch
```

See also

STL Basics (Page 7245)

Settings for STL (Page 7250)

Working with networks (Page 7251)

Inserting STL instructions (Page 7259)

Editing STL instructions (Page 7277)

Example of controlling a conveyor belt (Page 7280)

Example of detecting the direction of a conveyor belt (Page 7282)

Example of detecting the fill level of a storage area (Page 7284)

Example of calculating an equation (Page 7286)

Example of a step sequence (Page 7290)

15.6.6 Example of a step sequence (S7-300, S7-400, S7-1500)

Programming a step sequence

The following figure shows a station for removing glass containers from a pallet. The pallets are transported on a conveyor belt to the station. When a pallet with glass containers reaches the station, the conveyor belt will be stopped and a gripper moves from its basic position (P0) to the position above the pallet (P2). Once the gripper is above the pallet, the gripping clamps will open and the gripper lowered. Sensors detect the actual position of the gripper and the status of the gripping clamps. The sequence of the gripper movement in this example is implemented by a step sequence. You can program the additional steps required for removing the bottles and transporting them on an additional conveyor belt.

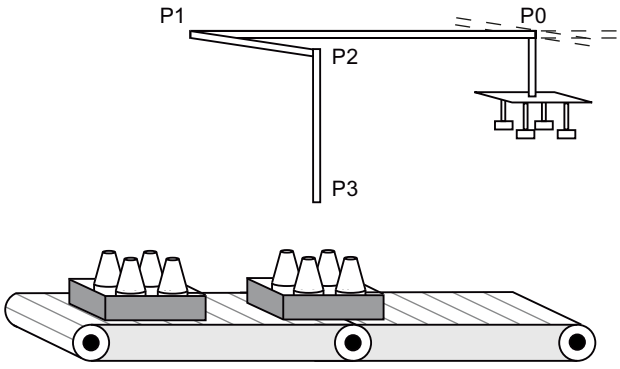

Implementation

The following table shows the declaration of the operands used in the PLC tag table:

Name	Data type	Comment
NUMBER	INT	Step number
Tag_Error	BOOL	Operand that is set when the step number is greater than 3 or if one of the steps was not executed.

The following table shows the declaration of the operands used in the block interface of the code block:

Name	Section	Data type	Comment
POS_0	Input	BOOL	Gripper in basic position (P0)
POS_1	Input	BOOL	Gripper in position 1 (P1)
POS_2	Input	BOOL	Gripper in position 2 (P2)
GRIPPER_OPEN	Input	BOOL	Gripping clamps open
OUT_POS_1	Output	BOOL	Move gripper to position 1

Name	Section	Data type	Comment
OUT_POS_2	Output	BOOL	Move gripper to position 2
OUT_GRIPPER	Output	BOOL	Open gripping clamps
OUT_POS_3	Output	BOOL	Move gripper to position 3

The following STL program shows how to implement this example:

```
STL                          Explanation
L "NUMBER"                   // Load the step number to accumulator 1.
JL END                       // Start of the jump list
JU POSITION_0                // At a value of "0" in the accumulator 1 jump to jump
                             label "POSITION_0".
JU POSITION_1                // At a value of "1" in the accumulator 1 jump to jump
                             label "POSITION_1".
JU POSITION_2                // At a value of "2" in the accumulator 1 jump to jump
                             label "POSITION_2".
JU POSITION_3                // At a value of "3" in the accumulator 1 jump to jump
                             label "POSITION_3".
END: JU ERROR                // End of the jump list
                             // At a step number greater than 3 jump to jump label
                             "ERROR".
POSITION_0: A #POS_0         // Jump label "POSITION_0"
                             // Scan to see if gripper is in basic position (P0).
= #OUT_POS_1                 // If condition is met, set output "OUT_POS_1" and
                             move gripper to position 1 (P1).
JCN ERROR                    // With RLO "0" jump to jump label "ERROR".
JC NEXT                      // With RLO "1" jump to jump label "NEXT".
POSITION_1: A #POS_1         // Jump label "POSITION_1"
                             // Scan to see if gripper is in position 1 (P1).
= #OUT_POS_2                 // If condition is met, set output "OUT_POS_2" and
                             move gripper to position 2 (P2).
JCN ERROR                    // With RLO "0" jump to jump label "ERROR".
JC NEXT                      // With RLO "1" jump to jump label "NEXT".
POSITION_2: A #POS_2         // Jump label "POSITION_2"
                             // Scan to see if gripper is in position 2 (P2).
= #OUT_GRIPPER               // If condition is met, set output "OUT_GRIPPER" and
                             open gripping clamps.
JCN ERROR                    // With RLO "0" jump to jump label "ERROR".
JC NEXT                      // With RLO "1" jump to jump label "NEXT".
POSITION_3: A #POS_2         // Jump label "POSITION_3"
                             // Scan to see if gripper is in position 2 (P2).
A #GRIPPER_OPEN              // Scan to see if the gripping clamps are open
= #OUT_POS_3                 // If condition is met, set output "OUT_POS_3" and
                             move gripper to position 3 (P3).
JCN ERROR                    // With RLO "0" jump to jump label "ERROR".
JC NEXT                      // With RLO "1" jump to jump label "NEXT".
```

STL	Explanation
NEXT: INC 1	// Jump label "NEXT"
	// Increase step number in accumulator 1 by one.
T "NUMBER"	// Transfer step number to operand "NUMBER".
L 3	// Move the current step number to accumulator 2.
	// Load value 3 to accumulator 1.
>I	// Scan to see if current step number is greater than 3.
JC RESET_NUMBER	// With a scan result "1" jump to jump label "RESET_NUMBER" and continue with program processing
BEU	// End block
RESET_NUMBER: L 0	// Jump label "RESET_NUMBER"
	// Load value "0" in accumulator 1.
T "NUMBER"	// Assign value "0" to operands "NUMBER" (step number).
BEU	// End block
ERROR: NOT	// Jump label "ERROR"
= "Tag_Error"	// Assign negated RLO to the operand "Tag_Error".
BEU	// End block

See also

STL Basics (Page 7245)
Settings for STL (Page 7250)
Working with networks (Page 7251)
Inserting STL instructions (Page 7259)
Editing STL instructions (Page 7277)
Example of controlling a conveyor belt (Page 7280)
Example of detecting the direction of a conveyor belt (Page 7282)
Example of detecting the fill level of a storage area (Page 7284)
Example of calculating an equation (Page 7286)
Example for heating an oven (Page 7288)

Creating SCL programs 16

16.1 Basics of SCL

16.1.1 Programming language SCL

Programming language SCL

SCL (Structured Control Language) is a high-level programming language based on PASCAL. The language is based on DIN EN 61131-3 (international IEC 1131-3).

The standard standardizes programming languages for programmable logic controllers. The SCL programming language fulfills the PLCopen Basis Level of ST language (Structured Text) defined in this standard.

Language elements

SCL also contains higher programming languages in addition to the typical elements of the PLC, such as inputs, outputs, timers or memory bits.

- Expressions
- Value assignments
- Operators

Program control

SCL provides convenient instructions for controlling the program allowing you, for example, to create program branches, loops or jumps.

Application

SCL is therefore particularly suitable for the following areas of application:

- Data management
- Process optimization
- Recipe management
- Mathematical / statistical tasks

16.1.2 Expressions

Description

Expressions are calculated during the runtime of the program and return a value. An expression consists of operands (such as constants, tags or function calls) and optionally out of operators (such as *, /, + or -). Expressions can be linked together or nested within each other by operators.

Evaluation order

The evaluation of the expression occurs in a specific order that is defined by the following factors:

- Priority of the operators involved
- Left-to-right order
- Brackets

Within an expression, do not use function calls that influence global or static tags.

Types of expressions

The following expression types are available depending on the operator:

- Arithmetic expressions
  Arithmetic expressions consist of either a numerical value or combine two values or expressions with arithmetic operators.
- Relational expressions
  Relational expressions compare the values of two operands and yield a Boolean value. The result is TRUE if the comparison is true, and FALSE if it is not met.
- Logical expressions
  Logical expressions combine two operands with logical operators (AND, OR, XOR) or negating operands (NOT).

How expressions are used

You can use the result of an expression in different ways:

- As a value assignment for a tag
- As as a condition for a control instruction
- As a parameter for a calling a block or instruction

See also

Operators and operator precedence (Page 7304)

Arithmetic expressions (Page 7295)

Relational expressions (Page 7297)

Logical expressions (Page 7303)

16.1.3 Arithmetic expressions

Description

Arithmetic expressions consist of either a numerical value or combine two values or expressions with arithmetic operators.

Arithmetic operators can process the data types that are allowed in the CPU in use. If two operands are involved in the operation, the data type of the result is determined based on the following criteria:

- If both operands are integers with sign and have different lengths, the result receives the data type of the longer integer (e. g. INT + DINT = DINT).
- If both operands are integers without sign and have different lengths, the result receives the data type of the longer integer (e. g. USINT + UDINT = UDINT).
- If one operand is an integer with sign and the other integer is an operand without sign, the result receives the next larger data type with sign that covers the integer without sign (e. g. SINT + USINT = INT).
  You can only execute an operation with such operands if the IEC check is not set.
- If one operand is an integer and the other operand is a floating-point number, the result receives the data type of the floating-point number (e. g. INT + REAL = REAL).
- If both operands are floating-point numbers and have different lengths, the result receives the data type of the longer floating-point number (e. g. REAL + LREAL = LREAL).
- The data type of the result of an operation that involves operands of the data type groups "Times" and "Date and time" can be found in the table in section "Data types of arithmetic expressions".
  You cannot use data types of the data type groups "Times" and "Date and time" when the IEC check is set.

Data types of arithmetic expressions

The following table shows the data types you can use in arithmetic expressions:

Operation	Operator	1st operand	2nd operand	Result
Power	**	Integer/floating-point number	Integer/floating-point number	Floating-point number
Unary plus	+	Integer/floating-point number TIME, LTIME	-	Integer/floating-point number TIME, LTIME
Unary minus	-	Integer/floating-point number TIME, LTIME	-	Integer/floating-point number TIME, LTIME

Operation	Operator	1st operand	2nd operand	Result
Multiplication	*	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME, LTIME	Integer	TIME, LTIME
Division	/	Integer/floating-point number	Integer/floating-point number (not equal 0)	Integer/floating-point number
		TIME, LTIME	Integer	TIME, LTIME
Modulo function	MOD	Integer	Integer	Integer
Addition	+	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME	TIME	TIME
		TIME	DINT	TIME
		LTIME	TIME, LTIME	LTIME
		LTIME	LINT	LTIME
		TOD	TIME	TOD
		TOD	DINT	TOD
		LTOD	TIME, LTIME	LTOD
		LTOD	LINT	LTOD
		DATE	LTOD	DTL
		DATE	TOD	• S7-300/400: DT • S7-1200/1500: DTL
		DT	TIME	DT
		LDT	TIME, LTIME	LDT
		DTL	TIME, LTIME	DTL

Operation	Operator	1st operand	2nd operand	Result
Subtraction	-	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME	TIME	TIME
		TIME	DINT	TIME
		LTIME [1]	TIME, LTIME	LTIME
		LTIME	LINT	LTIME
		TOD	TIME	TOD
		TOD	DINT	TOD
		TOD	TOD	TIME
		LTOD	TIME, LTIME	LTOD
		LTOD	LINT	LTOD
		LTOD	LTOD	LTIME
		DATE	DATE	• S7-300/400/1200: TIME • S7-1500: LTIME
		DT	TIME	DT
		DT	DT	TIME
		LDT	TIME, LTIME	LDT
		DTL	TIME, LTIME	DTL
		DTL	DTL	• S7-1200: TIME • S7-1500: LTIME
[1] Combinations between nanoseconds and milliseconds are not possible within expressions.				

For additional information on valid data types, refer to "See also".

Example

The following example shows an arithmetic expression:

```
SCL
"MyTag1":= "MyTag2" * "MyTag3";
```

See also

Expressions (Page 7294)

Operators and operator precedence (Page 7304)

16.1.4 Relational expressions

Description

Relational expressions compare the values or the data types of two operands and yield a Boolean value. The result is TRUE if the comparison is true, and FALSE if it is not met.

Relational operators can process the data types that are allowed in the CPU in use. The data type of the result always is BOOL.

Note the following rules when forming relational expressions:

- All tags are comparable within the following data type groups:
  - Integers/floating-point numbers
  - Binary numbers
  - String
- With the following data types/data groups, only tags of the same type can be compared:
  - TIME, LTIME
  - Date and time
  - PLC data types
  - ARRAY
  - STRUCT
  - Tag to which ANY is pointing
  - Tag to which VARIANT is pointing
- The comparison of STRINGs takes place according to the character encoding in the character set that is set in Windows. The comparison of WSTRINGs takes place according to the character encoding in UTF-16. The length of the tags and the numerical value of each character are used for the comparison.
- S5TIME tags are not permitted as comparison operands. An explicit conversion from S5TIME to TIME or LTIME is necessary.

Comparison of floating-point numbers

When floating-point numbers are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check.

The special bit patterns of invalid floating-point numbers (NaN) that are the outcome of undefined results (e.g. root of -1) are not comparable. This means if one of the operands has the value NaN, the comparison expression "==: Equal" as well as "<>: Not equal" has the result FALSE.

Comparison of character strings

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table shows examples of "==" comparison of strings:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	0
'aa'	'aaa'	0

The following table shows examples of "<>" comparison of strings:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0
'aa'	'aaa'	1

You can also compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

Comparison of timers, date and time

Bit patterns of invalid timers, date and times, e.g. DT#2015-13-33-25:62:99.999_999_999, cannot be compared. This means if one of the operands has an invalid value, the instruction "==: Equal" as well as "<>: Not equal" has the result FALSE.

Not all times can be compared directly with each other, such as S5TIME. In this case they are implicitly converted into another time so that they can be compared, for example to TIME.

If you want to compare dates and times of different data types, the value of the smaller date or time data type is implicitly converted into the larger date or time data type. This means the two date and time data types DATE and DTL, for example, are compared on the basis of DTL.

When the implicit conversions fail, the comparison result is FALSE.

Comparing tags of WORD data type to tags of the S5TIME data type

Both tags are converted to the TIME data type when comparing a tag of WORD data type to a tag of S5TIME data type. The WORD tag is interpreted as an S5TIME value. If one of the two tags cannot be converted, the comparison is not performed and the result is FALSE. If the conversion is successful the comparison is performed based on the selected comparison instruction.

Comparison of structures

Note

Availability of comparison of structures

The option to compare structures is available for a CPU of the S7-1200 series as of firmware version >= 4.2, and for a CPU of the S7-1500 series as of firmware version >= 2.0.

You can compare the values of two structured operands with each other when both tags are of the same structure data type. When structures are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check. An exception are the comparisons in which one of the operands is a VARIANT or an ANY. If the data type is not

yet known at the time when the program is created, you can use the VARIANT data type. In this case you can also compare the operands with a structured tag of any data type. You can also compare two tags of the data type VARIANT or ANY with each other.

You can select the data type VARIANT for the comparison of structures from the drop-down list in the instruction box. Tags from the following data types are possible:

- PLC data type (UDT)
- STRUCT (either the structure of the STRUCT data type must be contained in a PLC data type or the two structures to be compared must be two elements of an ARRAY of STRUCT. Instance data blocks and tags of anonymous structures are not permitted.)
- Tag to which ANY is pointing
- Tag to which VARIANT is pointing

The following requirements must be met so that the two tags of the data type ARRAY can be compared with each other:

- The elements must each have the same data type.
- The two ARRAYs must have the same dimension.
- All dimensions must have the same number of elements. The exact ARRAY limits do not have to match.

Note

ARRAY of BOOL

When you compare two operands of the data type ARRAY of BOOL with each other and the number of elements cannot be divided by 8, the fill bits are compared as well. This may have an effect on the comparison result.

The table below shows an example of a comparison of structures for "==: Equal":

<Operand1>			**<Operand2>**			**RLO of the instruction**
Tag of data type A <PLC data type>		Tag value	Tag of data type A <PLC data type>		Tag value	1
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

<Operand1>			**<Operand2>**			**RLO of the instruction**
Tag of data type A <PLC data type>		Tag value	Tag of data type B <PLC data type>		Tag value	0
	BOOL	FALSE		BOOL	TRUE	
	INT	2		INT	3	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	VARIANT (supplied with tag of data type A)		Tag value	1
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

The table below shows an example of a comparison of structures for "<>: Not equal":

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type A <PLC data type>		Tag value	0
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	Tag of data type B <PLC data type>		Tag value	1
	BOOL	FALSE		BOOL	TRUE	
	INT	2		INT	3	

<Operand1>			<Operand2>			RLO of the instruction
Tag of data type A <PLC data type>		Tag value	VARIANT (supplied with tag of data type A)		Tag value	0
	BOOL	FALSE		BOOL	FALSE	
	INT	2		INT	2	

Data types of relational expressions

The following table shows the data types/data type groups you can use in relational expressions:

Operation	Operator	1st operand	2nd operand	Result
Compare for equal, not equal	=, <>	Integer/floating-point number	Integer/floating-point number	BOOL
		Bit strings	Bit strings	BOOL
		String	String	BOOL
		TIME, LTIME	TIME, LTIME	BOOL
		Date and time	Date and time	BOOL
		VARIANT/ANY	VARIANT/ANY	BOOL
		Any data type (but must correspond to the data type with which the VARIANT is supplied)	VARIANT/ANY	BOOL
		VARIANT/ANY	Any data type	BOOL
		PLC data type	PLC data type	BOOL
		ARRAY of <data type> with fixed and variable ARRAY limits	ARRAY of <data type> with fixed and variable ARRAY limits	BOOL
		STRUCT	STRUCT	BOOL
Compare for less than, less than-equal to, greater than, greater than or equal to	<, <=, >, >=	Integer/floating-point number	Integer/floating-point number	BOOL
		Bit strings (S7-1200/1500 only)	Bit strings (S7-1200/1500 only)	BOOL
		String	String	BOOL
		TIME, LTIME	TIME, LTIME	BOOL
		Date and time	Date and time	BOOL

Example

The following example shows a relational expression:

SCL

```
IF a > b THEN c:= a;
IF A > 20 AND B < 20 THEN C:= TRUE;
IF A<>(B AND C) THEN C:= FALSE;
```

Note

The comparison for STRING and DT are executed internally in the S7-300/400 by extended instructions. The following operands are not permitted for these functions:

- Parameter of a FC
- In-out parameter of an FB of type STRUCT or ARRAY

Note

Comparing the hardware data types HW_IO and HW_DEVICE

If you want to compare the two data types, you must first create a tag of data type HW_ANY in the "Temp" section of the block interface and then copy the LADDR (of data type HW_DEVICE) to the tag. It is then possible to compare HW_ANY and HW_IO.

See also

Expressions (Page 7294)

Operators and operator precedence (Page 7304)

16.1.5 Logical expressions

Description

Logical expressions combine two operands with logical operators AND OR XOR or negating operands NOT.

Logical operators can process the data types that are allowed in the CPU in use. The result of a logical expression is of BOOL data type, if both operands are of BOOL data type. If at least one of both operands is a bit string, then the result is also a bit string and is determined by the type of the highest operand. For example, when you link a BYTE type operand to a WORD type operand, the result is type WORD.

To link a BOOL type operand with a bit string, you must first explicitly convert it to a bit string.

Data types of logical expressions

The following table shows the data types you can use in logical expressions:

Operation	Operator	1st operand	2nd operand	Result
Negation (generate one's complement)	NOT	BOOL	-	BOOL
		Bit string	-	Bit string
AND logic operation	AND or &	BOOL	BOOL	BOOL
		Bit string	Bit string	Bit string
OR logic operation	OR	BOOL	BOOL	BOOL
		Bit string	Bit string	Bit string

Operation	Operator	1st operand	2nd operand	Result
EXCLUSIVE OR logic operation	XOR	BOOL	BOOL	BOOL
		Bit string	Bit string	Bit string

Example

The following example shows a logical expression:

SCL

```
IF "MyTag1" AND NOT "MyTag2" THEN c := a;
MyTag := ALPHA OR BETA;
```

See also

Expressions (Page 7294)

Operators and operator precedence (Page 7304)

SCL programming examples (Page 7360)

16.1.6 Operators and operator precedence

Operators and their order of evaluation

Expressions can be linked together or nested within each other by operators.

The order of evaluation for expressions depends on the precedence of operators and brackets. The following basic rules apply:

- Arithmetic operators are evaluated before relational operators and relational operators are evaluated before logical operators.
- Operators with no precedence are evaluated according to their occurrence from left to right.
- Value assignments are evaluated from right to left.
- Operations in brackets are evaluated first.

The following table provides an overview of the operators and their precedence:

Operator	Operation	Precedence
Arithmetic expressions		
+ (Page 7295)	Unary plus	2
- (Page 7295)	Unary minus	2
** (Page 7295)	Power	3
* (Page 7295)	Multiplication	4
/ (Page 7295)	Division	4
MOD (Page 7295)	Modulo function	4
+ (Page 7295)	Addition	5
- (Page 7295)	Subtraction	5

Operator	Operation	Precedence
+= (Page 7295), -= (Page 7295), *= (Page 7295), /= (Page 7295)	Combined value assignments	11
Relational expressions		
< (Page 7297)	Less than	6
> (Page 7297)	Greater than	6
<= (Page 7297)	Less than or equal	6
>= (Page 7297)	Greater than or equal	6
= (Page 7297)	Equal	7
<> (Page 7297)	Not equal	7
Logical expressions		
NOT (Page 7303)	Negation	3
AND (Page 7303) or & (Page 7303)	Boolean AND	8
XOR (Page 7303)	Exclusive OR	9
OR (Page 7303)	Boolean OR	10
Reference expressions		
REF (Page 305)	Reference	
^ (Page 309)	Dereference	1
?= (Page 314)	Assignment attempt	11
Miscellaneous operations		
( ) (Page 7294)	Brackets	1
:= (Page 7305)	Assignment	11

16.1.7 Value assignments

Definition

You can use a value assignment to assign the value of an expression to a tag. On the left side of the assignment is the tag that takes the value of the expression on the right.

The name of a function can also be specified as an expression. The function is called by the value assignment and returns its function value to the tag on the left.

The data type of value assignment is defined by the data type of the tag on the left. The data type of the expression on the right must match this type.

You have the following options for programming value assignments:

- Single value assignments
  In case of single value assignments, an expression or tag is assigned to a single tag only:
  Example: `a := b;`
- Multiple value assignments
  In case of multiple value assignments, multiple assignments can be executed with one instruction.
  Example: `a := b := c;`
  This corresponds to the following notation:
  ```
  b := c;
  a := b;
  ```
- Combined value assignments
  In case of combined value assignments, you can combine the operators "+", "-", "*" and "/" with the assignment operator:
  Example: `a += b;`
  This corresponds to the following notation:
  ```
  a := a + b;
  ```
  You can also assign combined value assignments multiple times:
  ```
  a += b += c *= d;
  ```
  This corresponds to the following sequence of assignments:
  ```
  c := c * d;
  b := b + c;
  a := a + b;
  ```

Value assignments for STRUCT data type or PLC data types

An entire structure can be assigned to another if the structures are identically organized and the data types as well as the names of the structural components match.

You can assign a tag, an expression or another structural element to an individual structural element.

Value assignments for the ARRAY data type

An entire ARRAY can be assigned to another ARRAY if both the data types of the ARRAY elements as well as the ARRAY limits match.

You can assign a tag, an expression or another ARRAY element to an individual ARRAY element.

Value assignments for the STRING data type

An entire STRING can be assigned to another STRING.

You can assign another STRING element to an individual STRING element.

Value assignment for data type WSTRING (S7-1200/1500)

An entire WSTRING can be assigned to another WSTRING.

You can assign another WSTRING element to an individual WSTRING element.

Value assignments for the ANY data type

You can assign a tag with the ANY data type only to the following objects:

- Input parameters or temporary local data of FBs that also have the data type ANY.
- Temporary local data of FCs that also have the data type ANY.

Note that you can only point to memory areas with "standard" access mode with the ANY pointer.

Value assignments for the POINTER data type

You cannot use POINTER in value assignments in SCL.

Examples

The following table shows examples of single value assignments:

SCL	
"MyTag1" := "MyTag2";	(* Assignment of a tag*)
"MyTag1" := "MyTag2" * "MyTag3";	(* Assignment of an expression*)
"MyTag" := "MyFC"();	(* Call for a function that assigns its function value to the "MyTag" tag*)
#MyStruct.MyStructElement := "MyTag";	(* Assignment of a tag to a structure element*)
#MyArray[2] := "MyTag";	(* Assignment of a tag to an ARRAY element*)
"MyTag" := #MyArray[1,4];	(* Assignment of an ARRAY element to a tag*)
#MyString[2] := #MyOtherString[5];	(* Assignment of a STRING element to another STRING element*)

The following table shows examples of multiple value assignments:

SCL	
"MyTag1" := "MyTag2" := "MyTag3";	(* Assignment of a tag*)
"MyTag1" := "MyTag2" := "MyTag3" * "MyTag4";	(* Assignment of an expression*)
"MyTag1" := "MyTag2" := "MyTag3 := "MyFC"();	(* Call of a function that assigns its function value to the tags "MyTag1", "MyTag1" and "MyTag1" *)
#MyStruct.MyStructElement1 := #MyStruct.MyStructElement2 := "MyTag";	(* Assignment of a tag to two structure elements*)
#MyArray[2] := #MyArray[32] := "MyTag";	(* Assignment of a tag to two ARRAY elements*)
"MyTag1" := "MyTag2" := #MyArray[1,4];	(* Assignment of an ARRAY element to two tags*)
#MyString[2] := #MyString[3]:= #MyOtherString[5];	(* Assignment of a STRING element to two STRING elements*)

The following table shows examples of combined value assignments:

SCL	
"MyTag1" += "MyTag2";	(* "MyTag1" and "MyTag2" are added and the result of the addition is assigned to "MyTag1".*)
"MyTag1" -= "MyTag2" += "MyTag3";	(* "MyTag2" and "MyTag3" are added. The result of the addition is assigned to ""MyTag2"" and then subtracted from "MyTag1". The result is assigned to "MyTag1".*)
#MyArray[2] += #MyArray[32] += "MyTag";	(* The ARRAY element "MyArray[32]" is added to "MyTag". The result is assigned to "MyArray[32]". The two ARRAY elements are then added and the result is assigned to the ARRAY element "MyArray[2]". The data types must be compatible for this operation.*)
#MyStruct.MyStructElement1 /= #MyStruct.MyStructElement2 *= "MyTag";	(* The structure element "MyStructElement2" is multiplied by "MyTag". The result is assigned to "MyStructElement2". The structure element "MyStructElement1" is then divided by "MyStructElement2" and the result is assigned to "MyStructElement1". The data types must be compatible for this operation.*)

See also

Operators and operator precedence (Page 7304)

Addressing operands (Page 103)

16.1.8 Calculating with floating-point numbers (REAL and LREAL) in SCL

Representation of the accuracy of floating-point numbers

The data type REAL, for example, is specified and calculated in the program with an accuracy of 6 decimal places. For the calculation of floating-point numbers (REAL and LREAL), it should be noted that this accuracy applies in general to each individual step of the calculation.

The exponents are adjusted when floating-point numbers are added and subtracted. The base and the exponents are thus equal during the addition and subtraction and only the mantissas are added. For additional information on the structure of floating-point numbers, refer to "See also".

Programming example

In the following programming example, you are to perform a calculation in which two operands of the data type REAL are to be added and one is to be subtracted. In the next step of the calculation, the constant 1 is divided by the previous result. To do this, create a global data block in which you declare your operands and a function in which you program the calculation operations.

Calculation formulas

```
y = a + b - c
Z = 1/y
```

The operands are stored with the following values:

Operand	Value	REAL value
a	100 000 000	$1.000000*10^8$
b	1	$1.000000*10^0$
c	100 000 000	$1.000000*10^8$

Procedure

Create the data block "DB_GlobalData":

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Specify the name "DB_GlobalData".
4. Select "Global DB" as the type of the data block.
5. Click "OK".
6. Create the following tags in the data block and enter the corresponding start values:

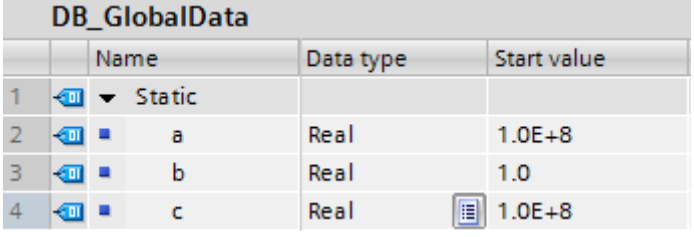

DB_GlobalData

	Name	Data type	Start value
1	Static		
2	a	Real	1.0E+8
3	b	Real	1.0
4	c	Real	1.0E+8

The start value of both tags is 100000000.0 and is converted into 1.0E+8 according to the data type REAL.

Create an SCL function and name it "FC_Calculate".

1. Declare the block interface as follows:

FC_Calculate		
	Name	Data type
1	Input	
2	Output	
3	InOut	
4	Temp	
5	z	Real
6	y	Real
7	Constant	
8	Return	
9	FC_Calculate	Void

2. Write the following formulas in your program code and establish an online connection to see the result:

SCL

```
#y := "DB_GlobalData".a + "DB_GlobalData".b - "DB_GlobalData".c;
#z := 1/#y;
```

#y	0.000000E+000
"Global data".a	1.000000E+008
"Global data".b	1.000000E+000
"Global data".c	1.000000E+008

As you can see, the result at the operand is #y = 0, even though the number 1 is actually expected as result.

The incorrect result comes about as follows:

1. In the first step of the calculation, the operands a + b are added. The REAL values of the two operands (a = $1.000000*10^8$ and b = $1.000000*10^0$) look as follows after the exponents have been adjusted:
   a = $1.000000*10^8$ and b = $0.00000001*10^8$. The last two places of the second number (operand b) are truncated, as they can no longer be represented due to the accuracy of 6 decimal places. Therefore, a 0 instead of a 1 is added to the operand.
2. In the second calculation step, the operand C is subtracted from the result of the preceding calculation step (intermediate result = $1.000000*10^8$ - c = $1.000000*10^8$ is $0.000000e^0$).
3. If you now calculate the operand z in the next calculation step, you try to divide by zero.

#z	16#7F800000
#y	0.000000E+000

1. Possible solution

To work around such cases, you can simply adjust your calculation formula. Write the formula as follows instead:

Calculation formulas

```
y = a - c + b
Z = 1/y
```

Since the result $0.000000e^0$ is available in this case after the first calculation step (operand a - c), the addition of the REAL number in the second calculation step (intermediate result + b) leads to the correct result ($y = 0.000000*10^0 + 1.000000*10^0 = 1.000000*10^0$).

	#y	1.000000E+000
	"DB_GlobalData".a	1.000000E+008
	"DB_GlobalData".c	1.000000E+008
	"DB_GlobalData".b	1.000000E+000
	#z	1.000000E+000
	#y	1.000000E+000

We recommend that you check how the calculation can be made most effectively before you program a calculation.

2. Possible solution

To calculate the above named formulas, use the LREAL data type instead of the REAL data type. Since the data type is processed with an accuracy of 15 decimal places, this problem does not even arise.

1. In the global data block "DB_GlobalData", create three new tags with the same values, each with the data type LREAL.

DB_GlobalData			
	Name	Data type	Start value
1	Static		
2	a	Real	1.0E+8
3	a_LREAL	LReal	100000000.0
4	b	Real	1.0
5	b_LREAL	LReal	1.0
6	c	Real	1.0E+8
7	c_LREAL	LReal	100000000.0

2. In the block interface of the FC "FC_Calculate", also declare two new tags with the data type LREAL.

FC_Calculate		
	Name	Data type
1	Input	
2	Output	
3	InOut	
4	Temp	
5	z_LREAL	LReal
6	z	Real
7	y_LREAL	LReal
8	y	Real
9	Constant	
10	Return	
11	FC_Calculate	Void

3. Use the new LREAL tags for the formulas in your program code and establish an online connection to see the result.

SCL

```
#y_LREAL := "DB_GlobalData".a_LREAL + "DB_GlobalData".b_LREAL -
"DB_GlobalData".c_LREAL;
#z_LREAL := 1/#y_LREAL;
```

▼	#y_LREAL	1.00000000000000E+000
	"DB_GlobalData".a_LREAL	1.00000000000000E+008
	"DB_GlobalData".b_LREAL	1.00000000000000E+000
	"DB_GlobalData".c_LREAL	1.00000000000000E+008
▼	#z_LREAL	1.00000000000000E+000
	#y_LREAL	1.00000000000000E+000

See also

REAL (Page 251)

LREAL (Page 252)

Invalid floating-point numbers (Page 253)

16.1.9 Calculating with constants in SCL

Interpretation of typed and non-typed constants

Constants are data with a fixed value that cannot change during program runtime. Constants can be read by various program elements during the execution of the program but cannot be overwritten. There are defined notations for the value of a constant, depending on the data type and data format. A distinction is made between the typed and non-typed notation.

We recommend that you do not mix typed and non-typed constants within mathematical functions as you may otherwise be faced with unwanted implicit conversions and thus end up with incorrect values.

Programming example

In the following programming example, you see a calculation operation with a typed and a non-typed constant.

1. Create an SCL function block and name it "FB_MathsFunctions".
2. Declare the "Variable_DINT" tag in the "Temp" section of the block interface.

FB_MathsFunctions		
	Name	Data type
1	Input	
2	Output	
3	InOut	
4	Static	
5	Temp	
6	Variable_DINT	DInt
7	Constant	

3. Write the following program code:
   Variable_DINT := INT#1 +50000;
   In this math operation, the typed constant INT#1 and the non-typed constant 50000 are to be added. The non-typed constant 50000 is underlined in yellow in the software to indicate that the constant value is outside the permitted range of the data type INT.

```
1
2  #Variable_DINT := INT#1 + 50000;
```

To see the result, go online.

1. Compile the SCL function block "FB_MathsFunctions" by right-clicking the command "Compile > Software (only changes)" to execute it.
2. Download the block with the command "Download to device > Software (only changes)".
3. Go online and monitor your block.

#Variable_DINT	-15535

The data type of the typed constant defines the data type of the addition. This means that the addition is performed in the data type area INT. In the first step, the non-typed constant 50000 is implicitly converted into the data type INT. However, the conversion leads to a negative value (-15536). This value is then added to the typed constant (INT#1). The result is -15535. Since the tag to which the result of the addition is to be written is declared with the data type DINT, the number -15535 is implicitly converted into the data type DINT and written to the tag "Variable_DINT". However, the result remains negative.

1. Possible solution

One option for avoiding this undesired result is to type both constants. If you type both constants, the longer data type determines the calculation operation.

1. Write the following program code in the "FB_MathsFunctions" SCL function block:

```
3
4  #Variable_DINT := INT#1 + DINT#50000;
```

   In this calculation operation, the typed constant INT#1 and the typed constant DINT#50000 are to be added.

To see the result, go online.

1. Compile the SCL function block "FB_MathsFunctions" by right-clicking the command "Compile > Software (only changes)" to execute it.
2. Download the block with the command "Download to device > Software (only changes)".
3. Go online and monitor your block.

#Variable_DINT	50001

   The constant INT#1 is converted into the DINT data type and the addition of the two constants is executed in the DINT data type area.

2. Possible solution

Another option for avoiding this undesired result is not to type both constants. If you do not type both constants, these are then interpreted as the widest possible data type on the current CPU. This means that on an S7-1500 series CPU, the two constants are interpreted as LINT data type.

1. Write the following program code in the "FB_MathsFunctions" SCL function block:

```
1
2  #Variable_DINT := 1 + 50000;
```

   In this calculation operation, the non-typed constant 1 and the non-typed constant 50000 are to be added.

To see the result, go online.

1. Compile the SCL function block "FB_MathsFunctions" by right-clicking the command "Compile > Software (only changes)" to execute it.
2. Download the block with the command "Download to device > Software (only changes)".
3. Go online and monitor your block.

#Variable_DINT	50001

The constants 1 and 50000 are interpreted as LINT data type and the result of the addition is again converted into the DINT data type.

See also

Basics of constants (Page 97)

Overview of the valid data types (Page 231)

16.2 Settings for SCL

16.2.1 Overview of the settings for SCL

Overview

The following tables show the settings you can make for SCL:

Editor settings

Group	Setting	Description
View	Keyword highlighting	Notation used to represent the keywords of the programming language. You can choose between uppercase and lowercase letters or a notation corresponding to the conventions of the Pascal programming language.
	Left-align actual parameters	Left-aligns the actual parameters for a block call. Only has this effect when the "Smart" option is selected in the settings under "General > Script/text editors > Indent".

Default settings for new blocks

If you create new blocks, the following settings are set as default values. You can change these in the block properties at a later point in time.

Group	Setting	Description
Compile	Create extended status information	Allows all tags in a block to be monitored. The memory requirements of the program and execution times increase, however, with this option.
	Check ARRAY limits[1)]	Checks at runtime whether array indices are within the declared range for an ARRAY. If an array index exceeds the permissible range, the enable output ENO of the block is set to "0".
	Set ENO automatically	Checks at runtime whether errors occur in the processing of certain instructions. If a runtime error occurs, the enable output ENO of the block is set to "0".
Block interface	View of the block interface	Defines whether the block interface of newly created blocks is shown in the table view or the text view.
[1)]For CPUs of the S7-300/400 series: When the ARRAY limits are violated, the enable output ENO is set to FALSE. For CPUs of the S7-1200/1500 series: When the ARRAY limits are violated, the enable output ENO is not set to FALSE. See "Addressing ARRAY components (Page 289)" for error query options.		

See also

Changing the settings (Page 7318)

EN/ENO mechanism (Page 141)

16.2.2 Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for SCL (Page 7317)

16.3 The programming window of SCL

16.3.1 Overview of the programming window

Function

The programming window is the work area, where you enter the SCL program.

The following figure shows the programming window of SCL:

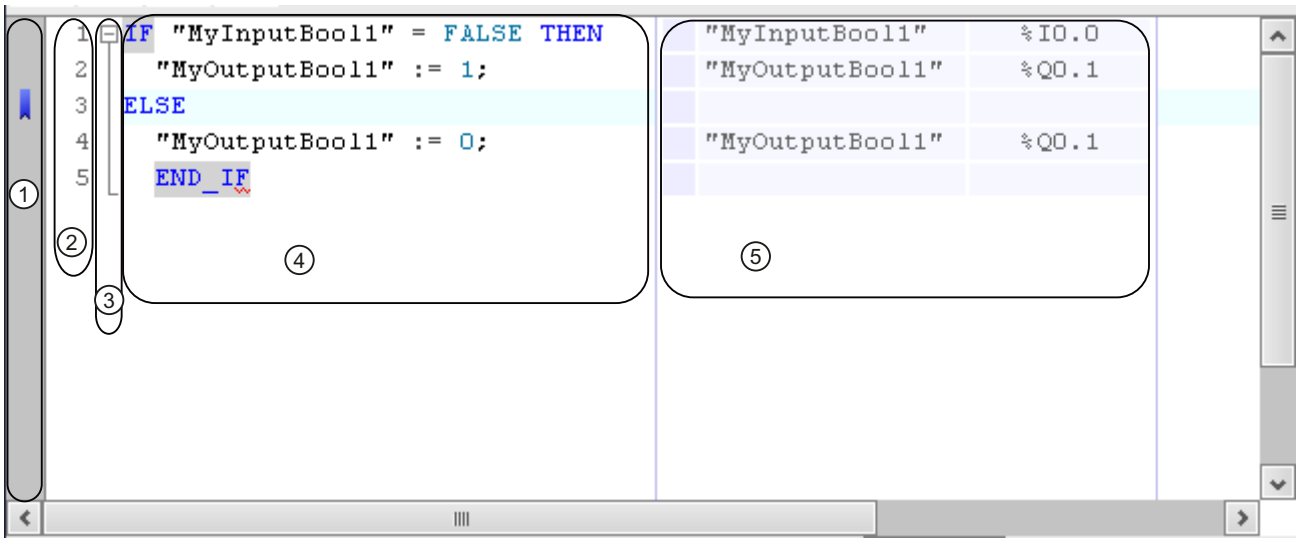

The programming window consists of the following sections:

Section	Meaning
① Sidebar	You can set bookmarks and breakpoints in the sidebar.
② Line numbers	The line numbers are displayed to the left of the program code.
③ Outline view	The outline view highlights related code sections.
④ Code area	You edit the SCL program in the code area.
⑤ Display of the absolute operands	This table shows the assignment of symbolic operands to absolute addresses.

See also

Customizing the programming window (Page 7320)

Formatting SCL code (Page 7321)

Expanding and collapsing sections of code (Page 7322)

Navigate to definitions (Page 7323)

Using bookmarks (Page 7324)

16.3.2 Customizing the programming window

Introduction

You can customize the appearance of the programming window and the program code in the following way:

- By setting the font, size and color
- By setting the tab spacing
- By displaying the line numbers
- By showing or hiding the absolute operands

Setting the font, size and color

To set the font, size and color, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Script/text editors" group.
3. Select the desired font and font size or choose a font color for the individual language elements.

Setting the tab spacing

To provide a better overview of the program, lines are indented according to syntax. Define the depth of indentation with the tab spacing.

To set the tab spacing, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Script/text editors" group.
3. Set the tab spacing.

Show line numbers

To display the line numbers, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Script/text editors" group.
3. Select the "Show line numbers" option.

Show or hide the absolute operands

You can show the assignment of symbolic and absolute operands in a table next to the program code, if required.

To hide or show the display of the absolute operands, follow these steps:

1. Click the "Absolute/symbolic operands" icon in the toolbar.
   The display of the absolute operands appears.
2. To move the display, click the table and drag it to the desired position while holding down the mouse button.
3. To change the width of the table, click on the right or left table border and drag it to the right or left while holding down the mouse button.

See also

Overview of the programming window (Page 7319)

Formatting SCL code (Page 7321)

Expanding and collapsing sections of code (Page 7322)

Navigate to definitions (Page 7323)

Using bookmarks (Page 7324)

16.3.3 Formatting SCL code

Introduction

To make the program clearer, you can indent or outdent individual lines manually or format code sections. Note the following information about formatting code sections:

- The type of formatting is based on the general settings for indents, but at least the line or the section is always indented. If you selected the setting "Smart", unnecessary spaces within the SCL instruction are also removed.
- Only syntactically correct code sections can be formatted.
- If you place the insertion point in the first or last line of an instruction for program control, for example in an IF instruction in the line with the "IF", the entire instruction is formatted.
- If you select text, only the selected text is formatted.

In addition, you can set that the actual parameters of block calls are generally to be left-aligned. To do this, "Smart" must be selected as the option for indents.

Indenting or outdenting lines

To indent or outdent individual lines, follow these steps:

1. Click on the line you want to indent or outdent.
2. Press the "Indent text", "Outdent text" button into the toolbar of the programming editor.

Note

You can set the width of the indent in "Options > Settings".

Formatting code sections

To format code sections, follow these steps:

1. Select the text that you want to format or place the insertion point in the appropriate line.
2. Select the "Format selected text automatically" button into the toolbar of the programming editor.

Left-aligning actual parameters of block calls

To left-align the actual parameters of block calls, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Script/text editors" group in the area navigation.
3. Under "Indent", select the "Smart" option.
4. Select the "PLC programming > SCL (Structured Control Language)" group in the area navigation.
5. Select the "Left-align actual parameters" check box under "View".
   The actual parameters are left-aligned for newly inserted block. To left-align the actual parameters of existing block calls, you can select the block calls and click the "Automatically format selected text" button in the toolbar of the program editor.

See also

Overview of the programming window (Page 7319)

Customizing the programming window (Page 7320)

Expanding and collapsing sections of code (Page 7322)

Navigate to definitions (Page 7323)

Using bookmarks (Page 7324)

16.3.4 Expanding and collapsing sections of code

Introduction

SCL instructions can span several lines. Examples for this are program control instructions or block calls.

Such instructions that belong together are identified as follows:

- An outline view between the display line number and the program code marks the entire code section.
- When you select the opening keyword, the closing keyword is automatically highlighted.

To improve clarity, you can expand or collapse sections of code that belong together in the outline display. The selected outline display is retained when you close the block or the project so that, the next time you open the block, the sections of code are displayed in exactly the same way as they were when you closed it.

Procedure

To expand or collapse the code section, follow these steps:

1. Click the minus sign in the outline view.
   The code section closes.
2. Click the plus sign in the outline view.
   The code section opens.

See also

Overview of the programming window (Page 7319)

Customizing the programming window (Page 7320)

Formatting SCL code (Page 7321)

Navigate to definitions (Page 7323)

Using bookmarks (Page 7324)

16.3.5 Navigate to definitions

You can use elements in your program code that were defined in other editors, for example, tags or data blocks (DBs) of called function blocks (FBs). To view the definition instances in the corresponding editors, you can specifically navigate to these instances.

Procedure

To navigate to the definition of a code element, follow these steps:

1. Right-click on the code element.
2. Select the "Go to > Definition" command in the shortcut menu.
   The editor in which the code element was defined opens and the definition instance is displayed.

Or:

1. Press and hold down the <Ctrl> key.
2. Move the mouse pointer over your program code.
   If the mouse pointer moves across a code element whose definition can be displayed, it is shown as underlined and the name of the element turns into a link.
3. Click on the link.
   The editor in which the code element was defined opens and the definition instance is displayed.

See also

Overview of the programming window (Page 7319)

Customizing the programming window (Page 7320)

Formatting SCL code (Page 7321)

Expanding and collapsing sections of code (Page 7322)

Using bookmarks (Page 7324)

16.3.6 Using bookmarks

16.3.6.1 Basics of bookmarks

Function

You can use bookmarks to mark program locations in extensive programs so that you can find them quickly later if they need revising. Bookmarks are displayed in the sidebar of the programming window. You can navigate between multiple bookmarks within a block using menu commands.

Bookmarks are saved with the project and are therefore available for anyone who wants to edit the block. However, they are not loaded to a device.

Bookmarks are not evaluated when blocks are compared.

See also

Setting bookmarks (Page 7325)

Navigating between bookmarks (Page 7325)

Deleting bookmarks (Page 7326)

16.3.6.2 Setting bookmarks

Requirement

The SCL block is open.

Procedure

To set a bookmark, follow these steps:

1. Right-click on the desired line in the sidebar.
2. Select the "Bookmarks > Set" command in the shortcut menu.

Or:

1. Click on the line in which you want to place the bookmark.
2. Click the "Set/delete bookmark" button in the toolbar.

Or:

1. Hold down the <Ctrl> key.
2. Click on the line in the sidebar in which you want to place the bookmark.

Result

A bookmark is placed in the program code.

See also

Basics of bookmarks (Page 7324)
Navigating between bookmarks (Page 7325)
Deleting bookmarks (Page 7326)

16.3.6.3 Navigating between bookmarks

Requirement

Several bookmarks are set in a block.

Procedure

To navigate between bookmarks, follow these steps:

1. Set the insertion cursor in the program code.
2. In the "Edit" menu, select the "Go to > Next bookmark" or "Go to > Previous bookmark" command.

Or:

1. Set the insertion cursor in the program code.
2. In the toolbar of the programming editor, click the "Go to next bookmark", "Go to previous bookmark" button.

Or:

1. Click in the sidebar.
2. Select the "Bookmarks > Next" or "Bookmarks > Previous" command in the shortcut menu.

Result

The line with the bookmark is highlighted.

See also

Basics of bookmarks (Page 7324)

Setting bookmarks (Page 7325)

Deleting bookmarks (Page 7326)

16.3.6.4 Deleting bookmarks

You can delete individual bookmarks or all bookmarks from the block or the CPU.

Deleting individual bookmarks

To delete an individual bookmark, follow these steps:

1. Right-click in the sidebar on the line in which you want to delete the bookmark.
2. Select the "Bookmarks > Remove" command in the shortcut menu.

Or:

1. Click on the line in which you want to delete the bookmark.
2. In the "Edit" menu, select the "Bookmarks > Remove" command.

Or:

1. Click on the line in which you want to delete the bookmark.
2. Click the "Set/delete bookmark" button in the toolbar.

Deleting all bookmarks from the block

To delete all bookmarks from the block, follow these steps:

1. Right-click in the sidebar.
2. Select the "Bookmarks > Delete all from block" command in the shortcut menu.

Or:

1. In the "Edit" menu, select the "Bookmarks > Delete all from block" command.

See also

Basics of bookmarks (Page 7324)
Setting bookmarks (Page 7325)
Navigating between bookmarks (Page 7325)

16.4 Entering SCL instructions

16.4.1 Rules for SCL instructions

Instructions in SCL

SCL recognizes the following types of instructions:

- Value assignments
  Value assignments are used to assign a tag a constant value, the result of an expression or the value of another tag.
- Instructions for program control
  Instructions for program control are used to implement program branches, loops or jumps.
- Additional instructions from the "Instructions" task card
  The "Instructions" task card offers a wide selection of standard instructions that you can use in your SCL program.
- Block calls
  Block calls are used to call up subroutines that have been placed in other blocks and to further process their results.

Rules

You need to observe the following rules when entering SCL instructions:

- Instructions can span several lines.
- Each instruction ends with a semicolon (;).
- No distinction is made between upper and lower case.
- Comments serve only for documentation of the program. They do not affect the program execution.

Examples

The following examples shows the various types of instructions:

SCL

```
// Example of a value assignment
"MyTag":= 0;
// Example of a block call
"MyDB"();
// Example of a program control instruction
WHILE "Counter" < 10 DO
    "MyTAG" := "MyTag" + 2;
END_WHILE;
```

See also

Basics of SCL (Page 7293)

16.4.2 Entering SCL instructions manually

Requirement

An SCL block is open.

Procedure

To enter SCL instructions, follow these steps:

1. Enter the syntax of the instruction using the keyboard.
   You are supported by the auto-complete function when performing this task. It offers all the instructions and operands that are allowed at the current location.
2. Select the required instruction or the desired operand from the auto-complete function.
   If you select an instruction that requires specification of operands, placeholders for the operands are inserted into the program. The placeholders for the operands are highlighted in yellow. The first placeholder is selected.
3. Replace this placeholder with an operand.
4. Use the <TAB> key to navigate to all other placeholders and replace them with operands.

Note

You can also drag-and-drop a defined operand from the PLC tag table or from the block interface into the program. To replace an operand that has already been inserted, hover the mouse pointer briefly over the operand to be replaced before releasing the mouse button. This selects the operand and when you release the mouse button it is replaced by the new operand.

Result

The instruction is inserted.

The programming editor performs a syntax check. Incorrect entries are displayed in red and italics. In addition, you also receive a detailed error message in the inspector window.

See also

Using autocompletion in textual programming languages (Page 6860)

Expanding and reducing the parameter list (Page 7354)

Surround program code with structure elements (Page 7331)

Data type conversion for S7-1200 (Page 489)

16.4.3 Inserting SCL instructions using the "Instructions" task card

The "Instructions" task card offers a wide selection of instructions that you can use in your SCL program. The SCL-specific instructions for program control are available in the "Instructions" task card.

Requirement

An SCL block is open.

Procedure

To insert SCL instructions into a program using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. To insert the instruction, select one of the following steps:
   - Navigate to the SCL instruction you want to insert and drag-and-drop it to the required line in the program code. The insertion location is highlighted by a green rectangle.
   - Select the location in the program code where you want to insert the instruction and then double-click on the instruction you want to insert.

   The instruction is inserted in the program. The placeholders for the operands are highlighted in yellow. A light yellow indicates the optional parameters that you do not necessarily have to interconnect. A darker yellow indicates the mandatory parameters that you must interconnect. The first placeholder is selected.
3. Replace this placeholder with an operand. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
4. Use the <TAB> key to navigate to all other placeholders and replace them with operands.

Result

The instruction is inserted.

The programming editor performs a syntax check. Incorrect entries are displayed in red and italics. In addition, you also receive a detailed error message in the inspector window.

See also

Using autocompletion in textual programming languages (Page 6860)

Expanding and reducing the parameter list (Page 7354)

Surround program code with structure elements (Page 7331)

Data type conversion for S7-1200 (Page 489)

16.4.4 Surround program code with structure elements

You have the option of surrounding your program code with the following structure elements:

- Ranges
  - REGION
- Control structures
  - IF ... THEN
  - CASE ... OF ...
  - FOR ... TO ... DO ...
  - WHILE ... DO ...
- Comment section
  - /**/

Procedure

To surround your program code with a structure element, proceed as follows:

1. Select the program code that you want to surround with a structure element.
2. Right-click the selected program code.
3. Select the required structure element under "Surround with" in the shortcut menu.
4. Alternatively, you can also select the structure element in the favorites or in the "Instructions" task card in the "Simple instruction > Program control" pane.

See also

Entering SCL instructions manually (Page 7329)

Inserting SCL instructions using the "Instructions" task card (Page 7330)

16.4.5 Working with regions

16.4.5.1 Using regions

Function

In SCL, you can structure your program code in regions. This improves the clarity of your program because you combine related parts of the program and can expand and collapse regions as needed. A region overview is available for fast navigation in the regions. The region overview appears next to the program window and can be displayed or hidden as needed.

The following figure shows the region overview and the programming window:

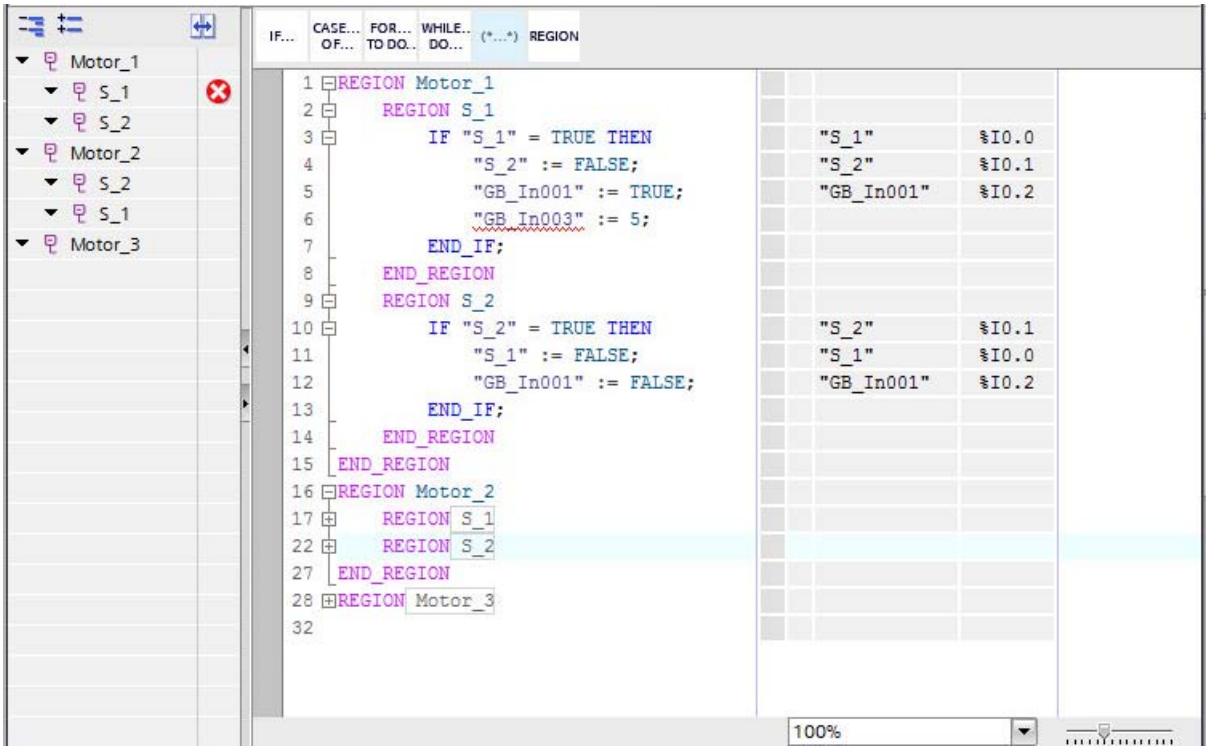

The region overview also shows the regions in which there are syntax errors. This allows you to quickly navigate to faulty code positions and rectify errors.

To summarize a program part in a region, enclose it with the keywords "REGION" and "END_REGION". Depending on your settings in the keyword highlighting, the notation of the keywords can also be "region" and "end_region" or "Region" and "End_Region". The keywords are automatically converted to the notation that corresponds to the current setting. To find "REGION" and "END_REGION" keywords that belong together, click either "REGION" or "END_REGION". Both keywords are highlighted in color as a result.

You can give each region a name. This has the following advantages:

- The name is also visible when a region is collapsed. This allows you to keep an overview of your program code.
- You can identify the regions in the region overview more easily and navigate to specific regions.

Specifying a name is optional.

Syntax

Use the following syntax to combine the program parts into regions:

```
REGION <Name>
     <Instructions>
END_REGION
```

Note the following information when using regions:

- The keywords "REGION" and "END_REGION" must be at the start of a line. Only spaces are allowed before the keywords.
- All characters after the keyword "END_REGION" are considered to be a comments and therefore have no effect on the execution of the program.
- All instructions must be completed within a region.
- Within CASE statements the instructions can be enclosed by regions. The constants which are used for comparison of the CASE statement must be outside the regions, however. The regions may not cover several branches.
  The following example shows the permitted use of regions within a CASE statement:
  ```
  CASE <Expression> OF
    <Constant1> : REGION <Instructions1> END_REGION
    <Constant2> : REGION <Instructions2> END_REGION
  ELSE REGION <Instructions0> END_REGION
  END_CASE
  ```
  The following example shows non-permitted uses of regions within CASE statements:
  ```
  CASE <Expression> OF
   REGION <Constant1> : <Instructions1> END_REGION
   <Constant2> : REGION <Instructions2> END_REGION
  ELSE <Instructions0>
  END_CASE
  CASE <Expression> OF
   <Constant1> : <Instructions1> REGION
   <Constant2> : END_REGION <Instructions2>
  ELSE <Instructions0>
  END_CASE
  ```
  In these cases, the constants within a region are interpreted as jump labels which can be reached with GOTO, but which are not are taken into consideration by the CASE statements.

You also have the option to nest regions. However, make sure that regions within other regions are successfully completed:

```
REGION <Name>
     <Instructions>
     REGION <Name>
           <Instructions>
     END_REGION
     <Instructions>
     REGION <Name>
          REGION
               <Instructions>
          END_REGION
          REGION <Name>
               <Instructions>
          END_REGION
     END_REGION
END_REGION
```

See also

Inserting regions (Page 7334)
Copying and inserting regions (Page 7335)
Navigating in regions (Page 7336)
Deleting regions (Page 7338)

16.4.5.2 Inserting regions

You have the following options to insert regions in your program code:

- You can insert regions manually via the keyboard.
- You can insert regions via the "Instructions" task card. You can hereby insert either empty regions or surround existing program code with regions.
- You can insert regions via the favorites You can hereby insert either empty regions or surround existing program code with regions.

Inserting regions manually via the keyboard

To insert a region into your program code manually via the keyboard, follow these steps:

1. Place the cursor at the position where you want to insert a region.
2. Enter the keyword "REGION".

   Note

   Depending on your settings in the keyword highlighting, the notation of the keywords can also be "region" and "end_region" or "Region" and "End_Region". The keywords are automatically converted to the notation that corresponds to the current setting.

3. Enter a name for the region. This step is optional.
4. Place the cursor at the end of the region.
5. Enter the keyword "END_REGION".
   The corresponding program code is combined in a region that you can expand and collapse. The new region is visible in the region overview. If you have not assigned a name, it is indicated in the region overview with the name "Unnamed".

Inserting regions via the "Instructions" task card

To insert a region in your program code via the "Instructions" task card, proceed as follows:

1. Place the cursor at the position where you want to insert an empty region or select the program code which you want to surround with a region.
2. Open the "Instructions" task card.
3. Navigate to "Control Panel > REGION".

4. Drag the instruction REGION to the position in your program code at which you want to insert a region, or insert the instructionREGION by double-clicking.
   Either an empty region is inserted or the selected program code is surrounded with a region. The new region is visible in the region overview and the placeholder "_name_" is used as name.
5. Replace the placeholder "_name_" with the required name or delete the placeholder if you do not want to assign a name. If you have assigned no name, the new region appears in the region overview with the name "Unnamed".

Insert via the favorites

To insert a region in your program code via the favorites, proceed as follows:

1. Place the cursor at the position where you want to insert an empty region or select the program code which you want to surround with a region.
2. Click the "REGION" instruction in the Favorites pane or on the Favorites bar in the program editor.
   Either an empty region is inserted or the selected program code is surrounded with a region. The new region is visible in the region overview and the placeholder "_name_" is used as name.
3. Replace the placeholder "_name_" with the required name or delete the placeholder if you do not want to assign a name. If you have assigned no name, the new region appears in the region overview with the name "Unnamed".

See also

Using regions (Page 7331)

Copying and inserting regions (Page 7335)

Navigating in regions (Page 7336)

Deleting regions (Page 7338)

16.4.5.3 Copying and inserting regions

You can also copy regions and their contents and insert them back into your program code.

Procedure

To copy regions and their contents, follow these steps:

1. Open the region overview.
2. Right-click the region you want to copy and select the "Copy" command from the shortcut menu.
3. Place the cursor at the position in your program code at which you want to insert the region.

4. Insert the region either via the "Insert" shortcut menu command or via the key combination <Ctrl+V> in your program code.
   The copied region is inserted expanded together with its contents into your program code. In addition, the region is inserted in the region overview.
5. If necessary, assign a new name for the inserted region.

Or:

1. Select the region in the program window. The region can be expanded or collapsed.
2. Copy the region either via the shortcut menu "Copy" or the key combination <Ctrl+C>.
3. Place the cursor at the position in your program code at which you want to insert the region.
4. Insert the region either via the "Insert" shortcut menu command or via the key combination <Ctrl+V> in your program code.
   The copied region is inserted expanded together with its contents into your program code. In addition, the region is inserted in the region overview.
5. If necessary, assign a new name for the inserted region.

See also

Using regions (Page 7331)

Inserting regions (Page 7334)

Navigating in regions (Page 7336)

Deleting regions (Page 7338)

16.4.5.4 Navigating in regions

You can use the region overview to navigate quickly in the regions. The region overview displays all regions that exist in your program code. In addition, it offers the following functions:

- Synchronizing the display of the regions in the region overview and in the programming window:
  You can specify whether expanding and collapsing regions affects only the region overview or the programming window, or both.
- Expanding and collapsing an individual region:
  You can expand and collapse individual regions in either the region overview or in the programming window. If you have switched on synchronization, the region is expanded or collapsed in both windows.
- Expanding or collapsing all regions:
  You can expand or collapse all regions simultaneously either in the region overview or in the programming window. If you have switched on synchronization, the regions are expanded or collapsed in both windows.

Synchronizing the display of the regions in the region overview and in the programming window

To define the synchronization of the display for the regions in the region overview and in the programming window, proceed as follows:

1. Open the region overview.
2. To switch on the synchronization of the display, click "Synchronization on/off" on the toolbar in the region overview.
   The button for the synchronization is displayed as active.
3. To switch off the synchronization of the display, click "Synchronization on/off" on the toolbar in the region overview.
   The button for the synchronization is displayed as inactive.

Expanding and collapsing an individual region

To expand or collapse an individual region, follow these steps:

1. Open the region overview.
2. In the region overview, right-click on the region that you want to expand or collapse.
3. Select the command "Expand" or "Collapse" in the shortcut menu.

Or:

1. Open the region overview.
2. In the region overview, select the region that you want to expand or collapse.
3. Press the shortcut <Ctrl+Shift+Num+> to expand or <Ctrl+Shift+Num-> to collapse.

Or:

1. In the programming window, place the cursor in the region that you want to expand or collapse.
2. Press the shortcut <Ctrl+Shift+Num+> to expand or <Ctrl+Shift+Num-> to collapse.

Depending on the synchronization settings, the region is either expanded or collapsed in the region overview or in the programming window or in both windows.

Expanding or collapsing all regions:

To expand or collapse all regions simultaneously, follow these steps:

1. Open the region overview.
2. To expand all regions, click "Expand all" on the toolbar in the region overview.
3. To collapse all regions, click "Collapse all" on the toolbar in the region overview.

Or:

1. Open the region overview.
2. Select any region in the region overview.
3. To expand all regions, press the shortcut <Ctrl+Shift+Num*>.
4. To collapse all regions, press the shortcut <Ctrl+Shift+Num/>.

Or:

1. Place the cursor in the programming window.
2. To expand all regions, press the shortcut <Ctrl+Shift+Num*>.
3. To collapse all regions, press the shortcut <Ctrl+Shift+Num/>.

Depending on the synchronization settings, all regions are either expanded or collapsed in the region overview or in the programming window or in both windows.

See also

Using regions (Page 7331)

Inserting regions (Page 7334)

Copying and inserting regions (Page 7335)

Deleting regions (Page 7338)

16.4.5.5 Deleting regions

You can remove regions from program code at any time.

Procedure

To delete a region, proceed as follows:

1. In your program code, remove the keyword "REGION" and the name of the region if you have assigned a name.
2. Delete the corresponding keyword "END_REGION" in your program code.
   The region is deleted from the program code and the region overview.

Note

As soon as you have removed either the keyword "REGION" or "END_REGION" from the program code, you can in each case recognize the respective associated keyword by the fact that it is marked with a red line for a syntax error.

See also

Using regions (Page 7331)

Inserting regions (Page 7334)

Copying and inserting regions (Page 7335)

Navigating in regions (Page 7336)

16.4.6 Defining the data type of an SCL instruction

16.4.6.1 Basic information on the data types of SCL instructions

Introduction

The SCL instructions that you employ for block programming use specific data types to calculate function values. Certain SCL instructions only support the use of a specific data type. You cannot change the data type for these instructions. However, most of the SCL instructions support the use of different data types. We differentiate between the following two types of such instructions:

- Instructions for which the data type of the function value is determined by the data type of the input parameters. This is the case for most instructions.
- Instructions with default data type. The instructions listed in the following table are of this type.

You will have to change the default data type if this is incompatible with the data type of the input parameter used. You can always change the data type based on the following syntax:
`_<data type>`

SCL instructions with default data type

The following table lists the SCL instructions with default data types:

Instruction	Default data type
CEIL	DINT
DECO	DWORD
FLOOR	DINT
NORM_X	REAL
PEEK	BYTE
SCALE_X	INT
TRUNC	DINT
CONCAT	STRING
T_DIFF	TIME

See also

Changing the data type of an SCL instruction (Page 7340)

Example for changing the data type of an SCL instruction (Page 7340)

16.4.6.2 Changing the data type of an SCL instruction

Procedure

Proceed as follows to insert an SCL instruction and change its data type:

1. Insert the instruction at the required point in the program using drag-and-drop.
2. Specify the operands for the instruction.
   The data type of the function value is specified based on the input parameters, or the default data type of the instruction is used.
3. Append the "_<data type>" string to the instruction name.
   "<data type>" represents the data type you need for the instruction.

See also

Basic information on the data types of SCL instructions (Page 7339)

Example for changing the data type of an SCL instruction (Page 7340)

16.4.6.3 Modifying the data types of IEC timers and IEC counters

IEC timers and IEC counters are internal system function blocks and require an instance data block. You can create the instance data blocks either as single or multi-instance. The data type of the instance data block is determined according to the associated instruction. For CPUs of the S7-1200 and S7-1500 series, you can, however execute the instructions with different data types, depending on your requirements.

If the newly set data type of the instance data block does not match the data type of the input parameter, an implicit conversion takes place if possible. If the conversion is not possible, you will receive an error message.

Procedure

To change the data type of an IEC timer or IEC-counter instance data block, proceed as follows:

1. Open the block in which you call the IEC timer or IEC counter.
   Depending on the instance type of the instance data block, there is a green-bordered box before (multi-instance) or after (single instance) the name of the instance data block.
2. Click the green-bordered box.
   A drop-down list box with the valid data types for the instance data block is opened.
3. Select the desired data type.

16.4.6.4 Example for changing the data type of an SCL instruction

Changing the default data type of the "Decode" instruction (DECO)

Data type DWORD is set as default if you insert the "Decode" instruction in the program.

```
"Tag_Result" := DECO(IN := "Tag_Value");
```

Modify the program code as follows to convert the data type from DWORD to BYTE:

```
"Tag_Result_BYTE" := DECO_BYTE(IN := "Tag_Value");
```

See also

Basic information on the data types of SCL instructions (Page 7339)

Changing the data type of an SCL instruction (Page 7340)

16.4.7 Displaying or hiding tag information

Introduction

Regardless of whether the operands are represented in absolute or symbolic form, you can show and hide simple or hierarchical comments used to document global tags. This information is taken from the PLC tag table.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all blocks currently opened and opened in future is shown.

You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual ones that you have opened.

If you select the display of tag information with hierarchical comments, the comments of the higher structure levels of structured tags will also be displayed. The display is in brackets after the comment of the tags; the comments of the individual levels are separated by a period. If there is no comment at a structure level for a tag, it is omitted in the display and this is recognizable because there are two periods.

Displaying or hiding tag information for all blocks

Follow the steps below to display or hide the tag information for all blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to show the tag information, either select the "Expand" option in the "Tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
4. If you want to hide the tag information, select the "Collapse" option in the "Tag information" drop-down list.
   The tag information is displayed or hidden for all blocks. When you open further blocks, the tag information is displayed or hidden depending on the selected setting.

Displaying or hiding tag information for an opened block

Follow the steps below to display or hide the tag information for an opened block:

1. If you want to show the tag information, either select the "Show tag information" option in the "Shows the tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
2. If you want to hide the tag information, select the "Hide tag information" option in the "Hides tag information" drop-down list.
   The tag information is displayed or hidden.

16.4.8 Using Favorites in SCL

16.4.8.1 Adding SCL instructions to the Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 6831)

Inserting SCL instructions using Favorites (Page 7343)

Removing SCL instructions from the Favorites (Page 7343)

16.4.8.2 Inserting SCL instructions using Favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 6831)

Adding SCL instructions to the Favorites (Page 7342)

Removing SCL instructions from the Favorites (Page 7343)

16.4.8.3 Removing SCL instructions from the Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 6831)

Adding SCL instructions to the Favorites (Page 7342)

Inserting SCL instructions using Favorites (Page 7343)

16.4.9 Insert block calls in SCL

16.4.9.1 Basic information on the block call in SCL

Calling function blocks

Syntax of a call

The following syntax is used to call a function block as a single or multi-instance:

- Single instance:
  - If the function block originates from the project:
    `<DBName> (Parameter list)`
  - If the function block originates from the "Instructions" task card:
    `<DB name>.<Instruction name> (Parameter list)`
    or
    `<Instruction name> (Parameter list)`
- Multi-instance
  `<#Instance name> (Parameter list)`

Calling as single instance or multi-instance

Function blocks can be called either as a single instance or a multi-instance.

- Calling as a single instance
  The called function block stores its data in a data block of its own.
- Calling as a multi-instance
  The called function block stores its data in the instance data block of the calling function block.

For additional information on the types of calls, refer to "See also".

Recursive block calls

You have the option of calling a block recursively. This means a block can call itself again. Keep in mind that the call depth is limited to 24 and that you cannot use multi-instances.

Parameter list

If you call another code block from a SCL block, you can supply the formal parameters of the called block with actual parameters.

The specification of the parameters has the form of a value assignment. This value assignment enables you to assign values (actual parameters) to the parameters you have defined in the called block.

The formal parameters of the called code block are listed in brackets directly after the call. Input and in-out parameters have the assignment identifier ":=", output parameters have the assignment identifier "=>". A placeholder placed after the parameter shows the required data type and the type of the parameter.

Rules for supplying parameters

The following rules apply to supplying parameters:

- Constants, tags and expressions can be used as actual parameters.
- The assignment order is not of importance.
- The data types of formal and actual parameters match. You can also select data types for the actual parameter for which implicit data type conversion into the data type of the formal parameter is possible.
- The individual assignments are separated by commas.
- If the called block has only one parameter, it is sufficient to specify the actual parameter in the brackets. The formal parameter need not be specified.

See also

Manually inserting block calls (Page 7349)

Inserting block calls with drag-and-drop (Page 7351)

Examples for calling a function block in SCL (Page 7347)

Block calls (Page 60)

Using and addressing operands (Page 89)

Calling functions

Syntax of a call

The following syntax is used to call a function:

```
<Function name> (Parameter list);     //Standard call
<Operand>:=<Function name> (Parameter list); // Call in an expression
```

Function value

Functions that provide a return value can be used in any expression in place of an operand in SCL. For this reason, the return value is also known as the "function value" in SCL.

The call options of functions depend on whether the function returns a function value to the calling block.

The function value is defined in the RET_VAL parameter. If the RET_VAL parameter is of the VOID data type, then the function will not return a value to the calling block. If the RET_VAL parameter has another data type, then the function returns a function value of this data type.

In SCL, all data types are permitted for the RET_VAL parameter except ANY, ARRAY, STRUCT and VARIANT, as well as the parameter types TIMER and COUNTER.

Call options

There are two possibilities for calling functions in SCL:

- Standard call for functions with and without a function value
  With a standard call, the results of the function is made available as an output and in-out parameter.
- Call in an expression for functions with a function value
  Functions that return a function value can be used in any expression in place of an operand, for example, a value assignment.
  The function calculates the function value, which has the same name as the function and returns it to the calling block. There the value replaces the function call.
  Following the call, the results of the function is made available as a function value or as an output and in-out parameter.

Recursive block calls

You have the option of calling a block recursively. This means a block can call itself again. Keep in mind that the call depth is limited to 24 and that you cannot use multi-instances.

Parameter list

If you call another code block from a SCL block, you need to supply the formal parameters of the called block with actual parameters.

The specification of the parameters has the form of a value assignment. This value assignment enables you to assign values (actual parameters) to the parameters you have defined in the called block.

The formal parameters of the called code block are listed in brackets directly after the call. Input and in-out parameters have the assignment identifier ":=", output parameters have the assignment identifier "=>". A gray placeholder placed after the parameter shows the required data type and the type of the parameter.

Rules for supplying parameters

The following rules apply to supplying parameters to functions:

- All parameters of the function must be supplied.
- The assignment order is not of importance.
- Constants, tags and expressions can be used as actual parameters.
- The data types of formal and actual parameters match. You can also select data types for the actual parameter for which implicit data type conversion into the data type of the formal parameter is possible.
- The individual assignments are separated by commas.
- If the called block has only one parameter, it is sufficient to specify the actual parameter in the brackets. The formal parameter need not be specified.
- When you call functions in SCL, you cannot use the release mechanism via EN. Use an IF statement instead to call functions conditionally.

See also

Manually inserting block calls (Page 7349)

Inserting block calls with drag-and-drop (Page 7351)

Examples for calling functions in SCL (Page 7348)

Block calls (Page 60)

Using and addressing operands (Page 89)

Examples for calling a function block in SCL

Calling as a single instance

The following example shows the call of an FB as a single instance:

SCL

```
// Call as a single instance
"MyDB" (MyInput:=10, MyInout:= "Tag1");
```

Result

After the call is executed, the value determined for the "MyInout" in/out parameter is available in "Tag1" in the "MyDB" data block.

Calling as a multi-instance

The following example shows the call of an FB as a multi-instance:

SCL

```
// Call as a multi-instance
#MyFB (MyInput:= 10, MyInout:= "Tag1");
```

Result

After the "#MyFB" block is executed, the value determined for the "MyInout" in/out parameter is made available in "Tag1" in the data block of the calling code block.

See also

Calling function blocks (Page 7344)

Manually inserting block calls (Page 7349)

Inserting block calls with drag-and-drop (Page 7351)

Block calls (Page 60)

Using and addressing operands (Page 89)

Examples for calling functions in SCL

Standard call

The following example shows a standard function call:

SCL

```
// Standard function call
"MyFC" (MyInput := 10, MyInOut := "Tag1");
```

Result

After the "MyFC" block is executed, the value determined for the "MyInOut" in/out parameter is available in "Tag1" in the calling block and needs to be further processed there.

Call in a value assignment

The following example shows a function call in a value assignment:

SCL

```
(*Call in a value assignment, a function value was defined for "MyFC" *)
#MyOperand := "MyFC" (MyInput1 := 3, MyInput2 := 2, MyInput3 := 8.9,
MyInOut := "Tag1");
```

Result

The function value of "MyFC" is transferred to "#MyOperand".

Call in an arithmetic expression

The following example shows a function call in an arithmetic expression:

```
SCL
(*Call in a mathematical expression, a function value was defined for
"MyFC"  *)
#MyOperand := "Tag2" + "MyFC" (MyInput1 := 3, MyInput2 := 2, MyInput3 :=
8.9);
```

Result

The function value of "MyFC" will be added to "Tag2" and the result will be transferred to "MyOperand".

See also

Calling functions (Page 7345)

Manually inserting block calls (Page 7349)

Inserting block calls with drag-and-drop (Page 7351)

16.4.9.2 Manually inserting block calls

You can insert calls for functions (FCs) and function blocks (FBs). Function blocks can be called either as a single, multi or parameter instance.

See also: Basics of instances (Page 62)

Inserting a call for a function (FC)

Proceed as follows to insert a function call:

1. Enter the function name.
2. Confirm your entry with the Return key.
3. Select the "Show all parameters" command from the context menu or press the key combination <Ctrl+Shift+space bar>. Alternatively, you can click "Expands/collapses the parameter list of block calls" in the function bar.
   The syntax for the function call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.

4. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
5. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters. You can navigate to the previous placeholder with <Shift + TAB>.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Enter the name of the function block.
2. Enter an opening parenthesis "(".
   The "Call options" dialog opens.
3. Confirm your entries with "OK".
   The syntax for the function block call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
4. In the dialog, enter whether you want to call the block as a single, multi or parameter instance.
   - If you click the "Single instance" button, enter a name for the data block that will be assigned to the call in the entry field "Name".
   - If you click the "Multi-instance" button, in the "Name in the interface" field, enter the name of the tag with which the called function block will be entered as a static tag in the interface of the calling block.
   - If you click on the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
5. If you call a block that contains monitoring, assign a ProDiag function block to the monitoring functions.
6. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
7. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters. You can navigate to the previous placeholder with <Shift + TAB>.

Result

The block call is inserted.

If you specify an instance data block that does not exist when calling a function block, it is created.

See also

Updating block calls (Page 7352)

Expanding and reducing the parameter list (Page 7354)

Basic information on the block call in SCL (Page 7344)

Block calls (Page 60)

Using autocompletion in textual programming languages (Page 6860)

16.4.9.3 Inserting block calls with drag-and-drop

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree.

Function blocks can be called either as a single, multi or parameter instance.

See also: Basics of instances (Page 62)

Requirement

The function to be called (FC) or the function block (FB) to be called is present.

Inserting a call for a function (FC)

To insert a function call using drag-and-drop, follow these steps:

1. Drag the function from the project tree into the program.
   The syntax for the function call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
2. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
3. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters. You can navigate to the previous placeholder with <Shift + TAB>.

Inserting a call for a function block (FB)

To insert a call for a function block (FB) using drag-and-drop, follow these steps:

1. Drag the function block from the project tree and drop it into the program.
   The "Call options" dialog opens.
2. In the dialog, enter whether you want to call the block as a single, multi or parameter instance.
   - If you click the "Single instance" button, enter a name for the data block that will be assigned to the call in the entry field "Name".
   - If you click the "Multi-instance" button, in the "Name in the interface" field, enter the name of the tag with which the called function block will be entered as a static tag in the interface of the calling block.
   - If you click on the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
3. If you call a block that contains monitoring, assign a ProDiag function block to the monitoring functions.

4. Confirm your entries with "OK".
   The syntax for the function block call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
5. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
6. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters. You can navigate to the previous placeholder with <Shift + TAB>.

Result

The block call is inserted.

If you specify an instance data block that does not exist when calling a function block, it is created.

See also

Updating block calls (Page 7352)

Expanding and reducing the parameter list (Page 7354)

Basic information on the block call in SCL (Page 7344)

Block calls (Page 60)

Using autocompletion in textual programming languages (Page 6860)

16.4.9.4 Updating block calls

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have the following options for updating the block calls:

- Explicit updating of all inconsistent block calls in the programming editor.
  The inconsistent block calls within the open block are updated. The following actions are carried out in the process:
  - New parameters are added. Please note, however, that the parameters are hidden for function blocks (FBs) and the parameters are supplied via the corresponding instance data block (DB). If required, you can show the parameters using the shortcut menu command "Show all parameters".
  - Deleted parameters are not removed. If necessary, expand the parameter list to remove deleted parameters manually.
  - Renamed parameters get the new parameter names.
- Explicit updating of a block call in the programming editor.
  The inconsistent call of this block is updated at all call locations. The following actions are carried out in the process:
  - New parameters are added.
  - Deleted parameters are not removed. If necessary, expand the parameter list to remove deleted parameters manually.
  - Renamed parameters get the new parameter names.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated. Make sure that you manually remove deleted parameters before the compilation process and supply all new formal parameters with actual parameters when you call functions.

Updating all inconsistent block calls in the programming editor

To update all block calls in a block, follow these steps:

1. Open the calling block in the programming editor.
2. Click "Update inconsistent block calls" in the toolbar.
   All inconsistent calls are updated. If necessary, supply new formal parameters of functions (FCs) with actual parameters.

Updating a specific block call in the programming editor

To update a specific block call in the programming editor, follow these steps:

1. Open the calling block in the programming editor.
2. Right-click on the block call that you want to update.
3. Select the "Update block call" command in the shortcut menu.
4. If parameters were added, enter the values for the new block parameters.

Note

Note that the "Update block call" command is only available as long as you did not previously update all block calls in the editor with the "Update inconsistent block calls" command.

Updating block calls during compilation

To implicitly update all block calls and uses of PLC data types during compilation, follow these steps:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the "Compile > Software (rebuild all blocks)" command in the shortcut menu.

See also

Manually inserting block calls (Page 7349)

Inserting block calls with drag-and-drop (Page 7351)

Expanding and reducing the parameter list (Page 7354)

16.4.10 Expanding and reducing the parameter list

In SCL, if you call blocks or insert instructions that are system-internal function blocks, the syntax and the parameter list with the placeholders for the actual parameters are inserted in the SCL program. To make the program code easier to read, the unused optional parameters are removed from the parameter list when you edit other instructions. You can restore these at any time. You can also explicitly reduce the parameter list when you have finished assigning the parameters.

Expanding the parameter list

To expand the parameter list, follow these steps:

1. Right-click in the block call or the instruction.
2. Select the "Expand parameter list" command from the shortcut menu or press the key combination <Ctrl+Shift+Space bar>.
   The parameter list is displayed in full again.

Reducing the parameter list

To reduce the parameter list, follow these steps:

1. Right-click in the block call or the instruction.
2. Select the "Reduce parameter list" command from the shortcut menu or press the key combination <Ctrl+Shift+Space bar>.
   All unused optional parameters are hidden.

See also

Entering SCL instructions manually (Page 7329)

Inserting SCL instructions using the "Instructions" task card (Page 7330)

Manually inserting block calls (Page 7349)

Inserting block calls with drag-and-drop (Page 7351)

16.4.11 Insert pragmas (S7-1200, S7-1500)

You have the option of assigning validities to your program code using pragmas. To do so, surround the program code with Pragma keywords and assign at least one parameter. Optionally, you can assign additional parameters to define the content. Note that the character "$" is used as escape sequence in string constants and cannot therefore be used as normal sign for the parameter name.

The general syntax for pragmas is as follows:

```
{PRAGMA_BEGIN 'Param1', 'Param2', ... , 'ParamN'}
      //Programcode
{PRAGMA_END}
```

Pragmas can also be nested. However, make sure that pragmas within other pragmas are successfully completed. Pragmas can be expanded and collapsed as needed.

Note

Pragmas are currently evaluated only in SiVArc. The "SIVARC" parameter and another parameter must be assigned.

For more information, refer to the SiVArc help.

Procedure

To insert a pragma, follow these steps:

1. Place the cursor at the position where you want to insert a pragma.
2. Enter "{ PRAGMA_BEGIN".
3. Enter at least one parameter with single quotation marks.

4. Enter any number of additional parameters in single quotation marks. This step is optional.
5. Enter "}".
6. Place the cursor at the end of the program code that you want to surround with the pragma.
7. Enter "{ PRAGMA_END}".
   The corresponding program code is combined in a pragma.

16.4.12 Inserting comments

Commenting program code

You have various options for commenting SCL programs:

- Line comment
  A line comment starts with "//" and extends to the end of the line.
- Comment section
  A comment section is started with "(* and completed by "*)". It can span several lines.

Inserting a line comment

To insert line comments, follow these steps:

1. Type "//" at the position where you want to place the comment. This does not have to be the beginning of the line.
2. Enter the comment text.

Inserting a comment section

To insert a comment section, follow these steps:

1. Type "(*" at the position where you want to place the comment. This does not have to be the beginning of the line.
2. Enter the comment text.
3. Complete the comment with "*)".

Disabling one or more lines with comments

To disable program code with comments, follow these steps:

1. Select the code lines you want to comment out.
2. Click the "Comment selection" button in the editor.
   "//" is inserted at the beginning of the line in the selected lines. The code that follows is interpreted as a comment. If lines already containing a line comment are disabled, "//" is inserted as well. If these lines are enabled again, the original comments are retained.

Enabling comment lines

To enable lines that have been commented out to be enabled as code again, proceed as follows:

1. Select the code lines you want to enable.
2. Click the "Remove comment" button in the editor.
   The "//" mark for line comments at the beginning of the line is removed.

Example

The following code contains comment sections and line comments

```
(*******************************************************************************
  A description of the instructions that follow can be placed here
*******************************************************************************)
IF "MyVal1" > 0 THEN //No division by 0
    "MyReal" := "MyVal2" (* input value *) / "MyVal1" (* measured value *);
END_IF;
//Data type conversion
"MyInt" := REAL_TO_INT("MyReal");
```

See also

(*...*): Insert a comment section (Page 1845)

16.5 Editing SCL instructions

16.5.1 Selecting instructions

You can select individual instructions or all instructions of a block.

Requirement

An SCL block is open.

Selecting individual instructions

To select individual instructions, follow these steps:

1. Set the insertion mark before the first character that you want to select.
2. Press and hold down the left mouse button.
3. Move the cursor to a position after the last character that you want to select.
4. Release the left mouse button.

Selecting all the instructions of a program

To select all instructions, follow these steps:

1. In the "Edit" menu, select the "Select All" command or use the keyboard shortcut <Ctrl+A>.

Note

When you select the opening keyword of an instructing, the closing keyword is automatically highlighted.

16.5.2 Copying, cutting and pasting instructions

Copying an instruction

To copy an instruction, follow these steps:

1. Select the instruction you want to copy.
2. Select "Copy" in the shortcut menu.

Cutting an instruction

To cut an instruction, follow these steps:

1. Select the instruction you want to cut.
2. Select the "Cut" command in the shortcut menu.

Inserting an instruction from the clipboard

To insert an instruction from the clipboard, follow these steps:

1. Copy or cut an instruction.
2. Click on the position at which you want to insert the instruction.
3. Select "Paste" in the shortcut menu.

16.5.3 Deleting instructions

Requirement

An SCL block is open.

Procedure

To delete an instruction, follow these steps:

1. Select the instruction you want to delete.
2. Select the "Delete" command in the shortcut menu.

16.6 SCL programming examples

16.6.1 Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbuttons at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbuttons at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.

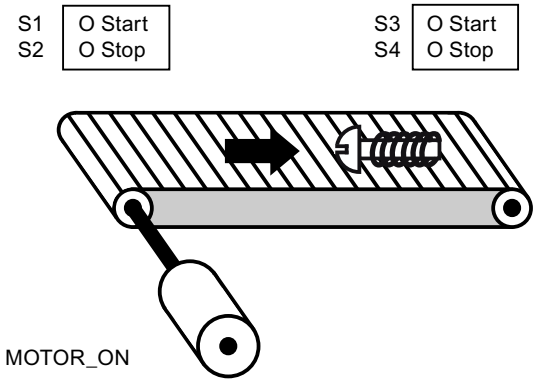

Implementation

The following table shows the definition of the tags used:

Operand	Declaration	Data type	Description
StartPushbutton_Left (S1)	Input	BOOL	Start pushbutton on the left side of the conveyor belt
StopPushbutton_Left (S2)	Input	BOOL	Stop pushbutton on the left side of the conveyor belt
StartPushbutton_Right (S3)	Input	BOOL	Start pushbutton on the right side of the convey-or belt
StopPushbutton_Right (S4)	Input	BOOL	Stop pushbutton on the right side of the convey-or belt
MOTOR_ON	Output	BOOL	Turn on the conveyor belt motor
MOTOR_OFF	Output	BOOL	Turn off the conveyor belt motor

The following SCL program shows how to implement this task:

SCL

```
IF "StartPushbutton_Left_S1" OR "StartPushbutton_Right_S3" THEN
"MOTOR_ON" := 1;
"MOTOR_OFF" := 0;
END_IF;
IF "StopPushbutton_Left_S2" OR "StopPushbutton_Right_S4" THEN
"MOTOR_ON" := 0;
"MOTOR_OFF" := 1;
END_IF;
```

The conveyor belt motor is switched on when start pushbutton "StartPushbutton_Left_S1" or "StartPushbutton_Right_S3" is pressed. The conveyor belt motor is switched off when stop pushbutton "StopPushbutton_Left_S2" or "StopPushbutton_Right_S4" is pressed.

See also

Logical expressions (Page 7303)

Basics of SCL (Page 7293)

Settings for SCL (Page 7317)

The programming window of SCL (Page 7319)

Entering SCL instructions (Page 7328)

Editing SCL instructions (Page 7358)

16.6.2 Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The detected running direction of the belt is indicated by a RIGHT arrow or a LEFT arrow. If additional conveyed material is approaching PEB1 from the right or PEB2 from the left, the displayed arrow is initially switched off until, after both photoelectric barriers are passed, the running direction can be detected again and the corresponding arrow displayed. For the solution of the task, 2 edge memory bits are needed that detect the signal change from "0" to "1" at the two photoelectric barriers.

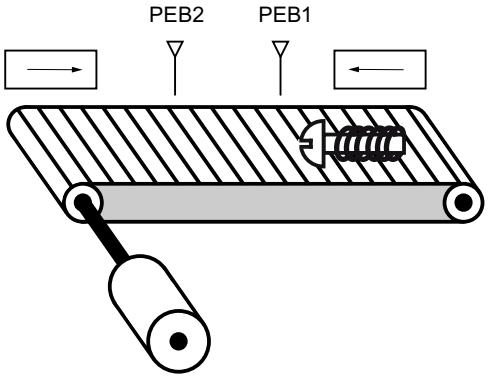

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
Photoelectric barrier PEB1	Input	BOOL	Photoelectric barrier 1
Photoelectric barrier PEB2	Input	BOOL	Photoelectric barrier 2
RIGHT	Output	BOOL	Display for movement to the right
LEFT	Output	BOOL	Display for movement to the left
Auxiliary flag PEB1	Input	BOOL	Edge bit memory 1
Auxiliary flag PEB2	Input	BOOL	Edge bit memory 2

The following SCL program shows how to implement this example:

SCL

```
// Program code for left running
IF "Photolelectric barrier PEB1" = 1 AND "Auxiliary flag PEB2" = 0 THEN
          "Auxiliary flag PEB1" := 1; // Set auxiliary flag for PEB1
          "LEFT" := 0; // Switch off arrow display left
          "RIGHT" := 0; // Switch off arrow display right
END_IF;
```

SCL

```
IF "Auxiliary flag PEB1" = 1 AND "Photoelectric barrier PEB2" = 1 THEN // The conveyor belt
is running to the left
          "LEFT" = 1;
          "RIGHT" := 0;
END_IF;

IF "LINKS" = 1 AND "Photoelectric barrier PEB2" = 0 THEN // Reset auxiliary flag for PEB1
          "Auxiliary flag PEB1" = 0
END_IF;
```

SCL

```
// Program code for right running
IF "Photolelectric barrier PEB2" = 1 AND "Auxiliary flag PEB1" = 0 THEN
          "Auxiliary flag PEB2" := 1; // Set auxiliary flag for PEB2
          "LEFT" := 0; // Switch off arrow display left
          "RIGHT" := 0; // Switch off arrow display right
END_IF;

IF "Auxiliary flag PEB2" = 1 AND "Photoelectric barrier PEB1" = 1 THEN // The conveyor belt
is running to the right
          "LEFT" = 0;
          "RIGHT" := 1;
END_IF;

IF "RIGHT" = 1 AND "Photoelectric barrier PEB1" = 0 THEN // Reset auxiliary flag for PEB2
          "Auxiliary flag PEB2" := 0;
END_IF;
```

If the photoelectric barrier "PEB1" has signal state "1" and the photoelectric barrier "PEB2" has signal state "0" at the same time, the object on the belt is moving to the left. If the photoelectric barrier "PEB2" has signal state "1" and the photoelectric barrier "PEB1" has signal state "0" at the same time, the object on the belt is moving to the right. The displays for a movement to the left or right will be turned off when the signal state at both photoelectric barriers is "0".

See also

Logical expressions (Page 7303)

Basics of SCL (Page 7293)

Settings for SCL (Page 7317)

The programming window of SCL (Page 7319)

Entering SCL instructions (Page 7328)

Editing SCL instructions (Page 7358)

16.6.3 Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock where they are loaded onto trucks. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

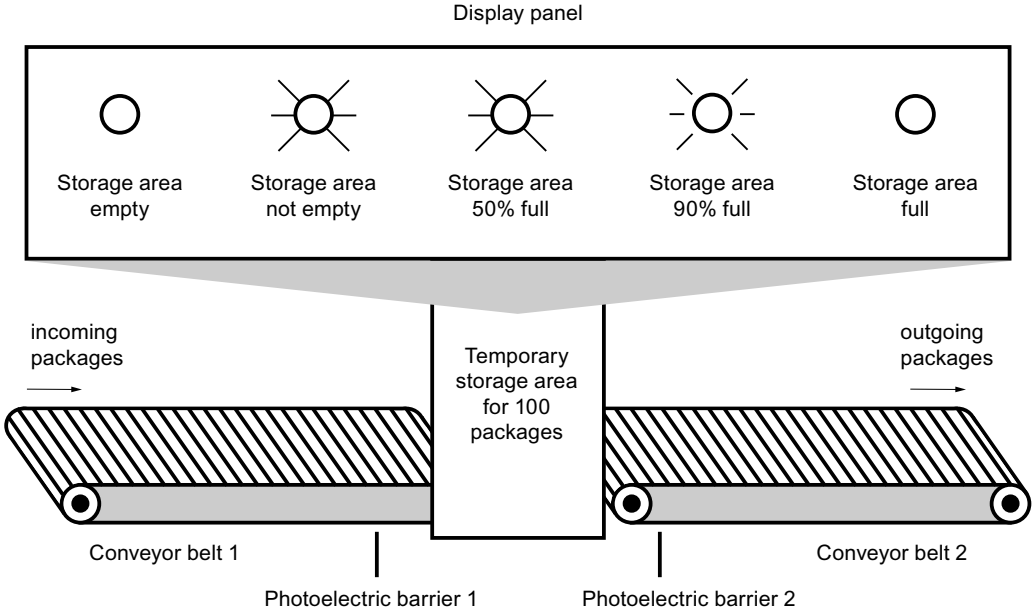

Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RESET	Input	BOOL	Reset counter
LOAD	Input	BOOL	Adjust the current counter value to the value of the PV parameter.
MAX STORAGE AREA FILL AMOUNT	Input	INT	Maximum possible number of packages in the storage area
PACKAGECOUNT	Output	INT	Number of packages in the storage area (current count value)

Name	Declaration	Data type	Description
STOCK_PACKAGES	Output	BOOL	Is set when the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag.
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50 % full
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90 % full
STOR_FULL	Output	BOOL	Display lamp: Storage area full
VOLUME_50	Input	INT	Comparison value: 50 packages
VOLUME_90	Input	INT	Comparison value: 90 packages
VOLUME_100	Input	INT	Comparison value: 100 packages

The following SCL program shows how to implement this example:

When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "MAX STORAGE AREA FILL AMOUNT" tag. As long as the current count value is greater than or equal to the value of the "MAX STORAGE AREA FILL AMOUNT" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".

SCL

```
"VOLUME_50" := 5; // Preassigning of the comparison value to 50 packages (for the test only 5 packages)
"VOLUME_90" := 9; // Preassigning of the comparison value to 90 packages (for the test only 9 packages)
"VOLUME_100" := 10; // Preassigning of the comparison value to 100 packages (for the test only 10 packages)
"MAX STORAGE AREA FILL AMOUNT" := 10; // Preassigning of the maximum amount in storage area to 100 packages (for the test only 10 packages)
```

SCL

```
"IEC_Counter_0_DB".CTUD(CU := "PEB1",
                        CD := "PEB2",
                        R := "RESET",
                        LD := "LOAD",
                        PV := "MAX STORAGE AREA FILL AMOUNT",
                        QU => "STOCK_PACKAGES",
                        QD => "STOR_EMPTY",
                        CV => "PACKAGECOUNT");
```

As long as the storage area contains packages, the "Storage area not empty" lamp is switched on.

SCL

```
"STOR_NOT_EMPTY" := NOT "STOR_EMPTY"
```

If the number of packages in the storage area is lower than 50%, the lamps for the alarms "Storage area 50% full", "Storage area 90% full" and "Storage area full" switch off.

SCL

```
IF "PACKAGECOUNT" < "VOLUME_50" THEN
"STOR_50%_FULL" := 0;
"STOR_90%_FULL" := 0;
"STOR_FULL" := 0;
END_IF;
```

If the number of packages in the storage area is greater than or equal to 50 %, the "Storage area 50 % full" lamp switches on.

SCL

```
IF "PACKAGECOUNT" >= "VOLUME_50" AND "PACKAGECOUNT <= "VOLUME_90" THEN
"STOR_50%_FULL" := 1;
"STOR_90%_FULL" := 0;
"STOR_FULL" := 0;
END_IF;
```

If the number of packages in the storage area is greater than or equal to 90 %, the "Storage area 90 % full" lamp switches on. The display lamp for 50 % full also remains on.

SCL

```
IF "PACKAGECOUNT" >= "VOLUME_90" AND "PACKAGECOUNT < "VOLUME_100" THEN
"STOR_50%_FULL" := 1;
"STOR_90%_FULL" := 1;
"STOR_FULL" := 0;
END_IF;
```

If the number of packages in the storage area reaches 100 %, the lamp for the "Storage area full" message switches on. The display lamps for 50 % and 90 % full also remain on.

SCL

```
IF "PACKAGECOUNT" >= "VOLUME_100" THEN
"STOR_50%_FULL" := 1;
"STOR_90%_FULL" := 1;
"STOR_FULL" := 1;
END_IF;
```

See also

Basics of SCL (Page 7293)

Settings for SCL (Page 7317)

The programming window of SCL (Page 7319)

Entering SCL instructions (Page 7328)

Editing SCL instructions (Page 7358)

Creating GRAPH programs (S7-300, S7-400, S7-1500)

17

17.1 Basic information on GRAPH (S7-300, S7-400, S7-1500)

17.1.1 GRAPH programming language (S7-300, S7-400, S7-1500)

Overview

GRAPH is a graphic programming language for creating sequential control systems. Sequences can be programmed in a fast straightforward manner using sequencers. The process is broken down into individual steps, each with a clear scope of functions, and organized into sequencers. The actions to be executed are defined within the individual steps. There are transitions between the steps. These contain conditions for advancing to the next step.

Sequential control system modules

A sequential control system controls the process in a pre-defined order and subject to certain conditions. The complexity of the sequential control system is specified by the automation task. At least three blocks always belong to a sequential control system.

- GRAPH function block
  In the GRAPH function block, you describe the individual steps and transitions of the sequential control system within one or more sequencers.
- Instance data block
  The instance DB contains the data and parameters of the sequential control system. It is assigned to the GRAPH function block and can be automatically generated by the system.
- Calling code block
  In order for the GRAPH function block to be executed within the cycle, it must be called from a higher-level code block. This block can be an organization block (OB), a function (FC), or another function block (FB). The GRAPH function block is always called as a single instance.

The following figure shows how the blocks work together in a sequential control system:

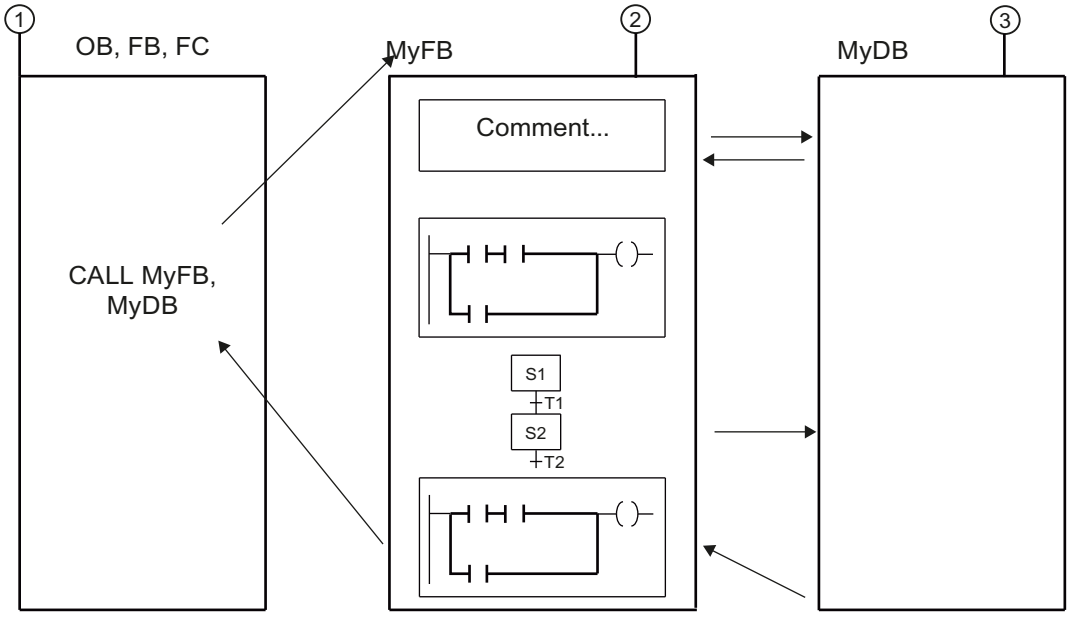

① Calling block

② GRAPH function block

③ Instance data block

The cycle of a GRAPH function block is defined by the cycle of the calling block. In each cycle, the permanent pre-instructions are executed first in the GRAPH function block. Then the actions of the active steps are processed. Last, the permanent post-instructions are executed.

Note

Permanent instructions are executed in each cycle, even if no step is active.

See also

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

Sequencers (Page 7371)

Steps and transitions (Page 7375)

Actions (Page 7378)

Block interface of a GRAPH function block (Page 7390)

17.1.2 Sequencers (S7-300, S7-400, S7-1500)

17.1.2.1 Basics on sequencers (S7-300, S7-400, S7-1500)

Introduction

Within a GRAPH function block, you can write your program in the form of sequencers. The sequencers can either process independent tasks, or you can break down a complex task into multiple sequencers. If the sequencers are processing independent tasks, they are processed in parallel in the program flow. If you are using multiple sequencers to break down a complex task into smaller segments, you must link the associated sequencers with jumps.

You subdivide the program into individual steps within a sequencer. In the simplest case, the steps are processed linearly one after the other. However, the use of alternative or simultaneous branches allows you to create sequencers with more complex structures. The program execution always begins with a step that you have defined as the initial step. A sequencer can have one or more initial steps. The initial steps can be at any position in the sequencer.

When a step is active, its actions are executed. More than one step can be be active simultaneously, e.g., within a simultaneous branch. While a step is active, the conditions of the following transition are constantly checked. As soon as all conditions are satisfied and no supervision errors are pending, the transition advances to the next step. This now becomes the active step. You can complete a sequencer either with a jump or a sequence end. The destination of the jump can be any step of the same sequencer or another sequencer. This enables cyclic processing of the sequencer.

Processing principle of a sequencer

The processing of a sequencer begins when the initial step(s) of the sequence become active. Several initial steps are permitted only in simultaneous branches. As long as a step is active, its actions are executed. The interlock conditions for the individual actions are taken into consideration here. Once all possible actions have been executed, a check is first made for the existence of a supervision error. If no such error exists and the following transition condition is satisfied, the next step of the sequence becomes active. If a supervision error exists or the transition condition is not satisfied, the step remains active until the error is eliminated or the transition condition is satisfied. At the end of a sequencer, you can use jumps to enable cyclic processing of the sequencer or a sequence end to terminate the sequencer.

See also

Elements of a sequencer (Page 7372)

Programming the structure of a sequencer (Page 7434)

GRAPH programming language (Page 7369)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

Steps and transitions (Page 7375)

Actions (Page 7378)

Block interface of a GRAPH function block (Page 7390)

17.1.2.2 Elements of a sequencer (S7-300, S7-400, S7-1500)

Overview

The following table shows the possible elements of a sequencer along with their icons:

Icon	Name
	Step and transition (Page 7375)
	Step (Page 7375)
	Transition (Page 7375)
	Simultaneous branch
	Alternative branch
	Close branch
	Jump to step
	Sequence end

Simultaneous branch

You can use simultaneous branches to program AND branches. This means that you use one transition to activate multiple steps whose actions are then executed. Simultaneous branches always start with a step.

The subsequent transitions of the simultaneous branches are located on the main branch, in which case the various simultaneous branches can be connected to various points of the main branch. Note that branches that you join together in a transition will only advance to the next step when all branches are completely processed.

You can program up to 249 simultaneous branches in a sequencer.

The following figure shows the graphic representation of a simultaneous branch:

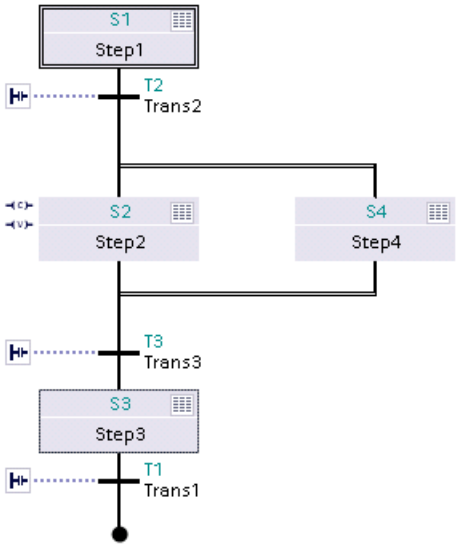

Alternative branch

You can use alternative branches to program OR branches. This means that you insert branches that start with a transition after a step. Depending of which transition's condition is satisfied first, the corresponding branch is executed. If multiple transitions are fulfilled at the same time, the set operating mode determines which branch is executed:

- Automatic or semi-automatic mode: The transition furthest to the left receives the highest priority and the corresponding branch is executed.
- Manual mode: The transition that is fulfilled first receives the highest priority and the corresponding branch is executed.

You can program up to 125 alternative branches in a sequencer.

The following figure shows the graphic representation of an alternative branch:

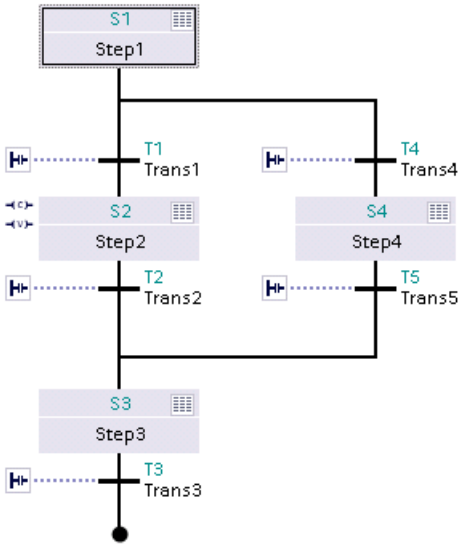

Close branch

You can use the "Close branch" element to close simultaneous and alternative branches to their parent branch. This is necessary if you do not want to complete the branch with a jump or a sequence end.

Jump to step

You can use a jump to continue the program execution at any step within the GRAPH function block. Jumps can be inserted at the end of the main branch, an alternative branch, or a simultaneous branch to enable cyclic processing of the sequencer. The jump and jump destination are represented in the sequencer as arrows, in which case the return transition is specified for the jump destination and the destination step is specified for the return.

Note

Avoid jumps from a transition to a directly preceding step. If you need this type of jump, insert an empty step with a transition without condition.

The following figure shows the graphic representation of a jump:

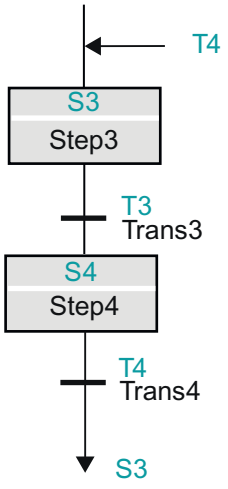

Sequence end

You can complete the sequencer or a branch with the "Sequence end" element. This terminates the sequencer or the branch. In a simultaneous branch, a transition must precede the sequence end. The "Sequence end" element is represented in the sequencer with a black dot.

The following figure shows the graphic representation of a sequence end:

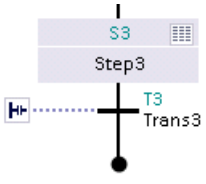

Note

If all branches of a sequencer are completed with a sequence end, the sequencer can be restarted by the "INIT_SQ" parameter or using the "Initialize" button in the "Sequence control" pane of the "Testing" task card.

See also

Basics on sequencers (Page 7371)

Programming the structure of a sequencer (Page 7434)

GRAPH programming language (Page 7369)

Steps and transitions (Page 7375)

Actions (Page 7378)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

Block interface of a GRAPH function block (Page 7390)

17.1.3 Steps and transitions (S7-300, S7-400, S7-1500)

17.1.3.1 Basics on steps (S7-300, S7-400, S7-1500)

Introduction

Steps are used to subdivide complex automation tasks into straightforward subtasks that can then be accomplished using actions. The individual steps are organized in sequencers so that every step can be executed in the specified order in the program flow. Each step must be assigned a unique name and number.

The actions of a step are carried out when the step becomes active. For this a step must either be an initial step or the transition condition preceding it must be satisfied. A step can also be called by an event-dependent action. In addition, you can program interlock and supervision conditions in a step.

You are free to choose the step at which the processing of the sequencer is to start. For this you specify any step as the initial step. In simultaneous branches you can specify multiple steps as initial steps. During cyclic operation of a sequencer, the initial step is treated like every other step, provided there is no initialization of the sequencer using the "INIT_SQ" parameter. If the sequencer is reinitialized, all initial steps are activated when there is a positive signal edge at "INIT_SQ". All other steps are deactivated. Initial steps are framed in the navigation view and sequence view to make them easy to recognize.

A step in which no actions are programmed is an empty step. Since an empty step does not have to process an action, the sequencer advances immediately to the following transition. This allows you to program two consecutive transitions.

You can change the name and number of a step at any time. In addition, you have the option of documenting steps using comments.

See also

Step elements (Page 7376)

Basics on transitions (Page 7377)

GRAPH programming language (Page 7369)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

Programming the structure of a sequencer (Page 7434)

Sequencers (Page 7371)

Actions (Page 7378)

Block interface of a GRAPH function block (Page 7390)

17.1.3.2 Step elements (S7-300, S7-400, S7-1500)

You can program the following components in a step:

- Interlock
- Supervision
- Actions

Interlock

You use an interlock to program conditions on which the execution of the individual actions depends. Actions of a step that are linked with the interlock are only executed when the interlock condition is satisfied. If the interlock condition is not satisfied, a fault exists. You can specify alarms for this case, which are then displayed. The fault does not affect the advancement to the next step, however. Note also that the interlock is automatically canceled when the step is deactivated. Optionally, you can specify that the operator must acknowledge an interlock alarm in order to close it.

You also have the option to not program any conditions in the interlock but to still link the interlock with actions. In this case, the interlock condition is always satisfied.

Supervision

You use a supervision to program conditions with which you can monitor a whole step. If the condition is satisfied, a fault exists and the step remains active. That is, in this case the sequencer does not advance to the next step but instead remains in the active step until the fault has been eliminated. You can also specify alarms and mandatory acknowledgement for the supervision.

Actions

Actions are programmed within the steps. For example, actions control inputs or outputs, activate or deactivate other steps of the sequencer, or call blocks. Actions thus contain instructions for process control.

Actions are differentiated into standard actions and event-dependent actions. Standard actions are executed as long as the step is active. Event-dependent actions are those that you have linked with an event.

See also

Basics on steps (Page 7375)

Basics on transitions (Page 7377)

GRAPH programming language (Page 7369)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

Programming the structure of a sequencer (Page 7434)

Sequencers (Page 7371)

Actions (Page 7378)

Block interface of a GRAPH function block (Page 7390)

17.1.3.3 Basics on transitions (S7-300, S7-400, S7-1500)

Introduction

Transitions are located between the steps and contain the conditions for advancing to the next step. That is, the sequencer only switches to the subsequent steps when the conditions of the transition are satisfied. In the process, the steps belonging to the transition are deactivated and the subsequent steps are activated. If the conditions are not satisfied, the steps belonging to the transition remain active. Each transition must be assigned a unique name and number. You can program the conditions of a transition in LAD or FBD.

A transition in which no conditions are programmed is an empty transition. In this case, the sequencer advances immediately to the subsequent steps.

You can change the name and number of a transition.

See also

GRAPH programming language (Page 7369)
Permanent instructions (Page 7388)
Conditions (Page 7389)
Interlock and supervision alarms (Page 7389)
Step elements (Page 7376)
Programming the structure of a sequencer (Page 7434)
Sequencers (Page 7371)
Actions (Page 7378)
Block interface of a GRAPH function block (Page 7390)

17.1.4 Actions (S7-300, S7-400, S7-1500)

17.1.4.1 Basics on actions (S7-300, S7-400, S7-1500)

Introduction

You can use actions, for example, to control inputs or outputs, activate or deactivate other steps of the sequencer, or call blocks. Actions are executed when the associated step becomes active (standard actions) or when an event linked with the action occurs (event-dependent actions). You can also use timers and counters in actions.

The following table shows examples of programming of actions:

Interlock	Event	Identifier	Action	Description
		N	MyTag1	Action consisting of instruction and operand without interlock
-(C)-		N	MyTag2	Action consisting of instruction and operand with interlock
	S1	S	MyTag3	Action consisting of event, instruction, and operand without interlock

Actions are executed in order from "top" to "bottom" when the step is active or the event occurs.

The following figure shows an example of the order in which actions are processed in a step:

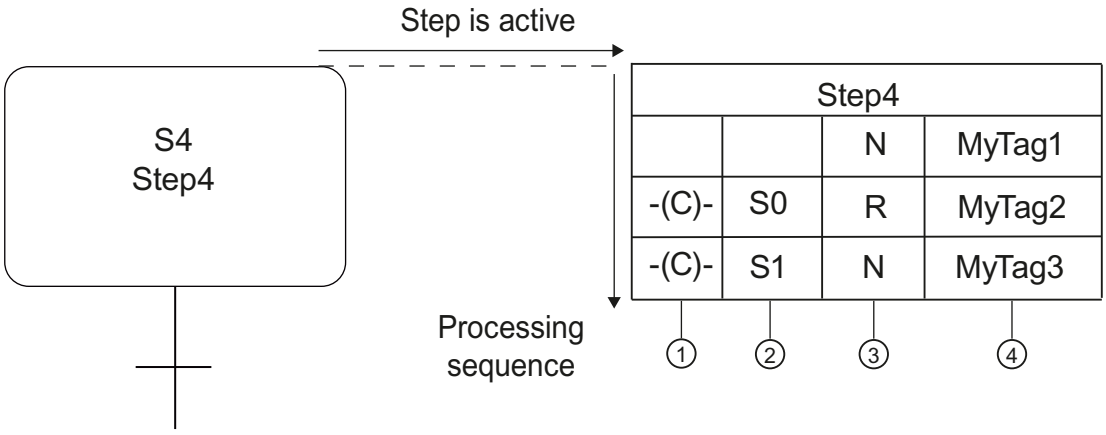

① Interlock

② Optional event

③ Identifier

④ Operand, assignment, or block call

An action consists of the following components:

- Interlock (optional)
  You can link an action with an interlock in order to influence the execution of the action.
- Event (optional)
  An event defines when an action is to be executed. You must specify an event for some identifiers.
- Identifier (mandatory)
  The identifier defines the type of action to be executed, e.g., the setting or resetting of an operand.
- Action (mandatory)
  The action determines the operands used to execute the action.

See also

Standard actions (Page 7380)

Event-dependent actions (Page 7381)

Block calls (Page 7384)

Timers (Page 7385)

Counter operations (Page 7387)

GRAPH programming language (Page 7369)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

Programming actions (Page 7454)

Sequencers (Page 7371)

Steps and transitions (Page 7375)

Block interface of a GRAPH function block (Page 7390)

17.1.4.2 Standard actions (S7-300, S7-400, S7-1500)

Introduction

Standard actions are executed as long as a step is active. The following table shows the possible standard actions and their identifiers:

Identifier	Data type of the operand	Meaning
N - Set as long as step is active	BOOL FB, FC, SFB, SFC	The signal state of the operand is "1" as long as the step is active. The specified block is called as long as the step is active. The step also counts as being active in the cycle in which the S1 event occurs.
S - Set to 1	BOOL	As soon as the step is active, the operand is set to "1" and then remains at "1".
R - Set to 0	BOOL	As soon as the step is active, the operand is set to "0" and then remains at "0".
D - On delay	BOOL and Time/ DWORD	n seconds after the step activation, the operand is set to "1" and remains at "1" for the duration of the step activation. This does not apply if the duration of the step activation is shorter than n seconds. You can specify the time as a constant or as a PLC tag of data type TIME/ DWORD.
L - Set for limited time	BOOL and Time/ DWORD	When the step is active, the operand is set to "1" for n seconds. The operand is then reset. The operand is also reset if the step activation time is less than the time duration. You can specify the time as a constant or as a PLC tag of data type TIME/ DWORD.

You can link all of these standard actions with an interlock. If an action is linked with an interlock, the action is only executed if the conditions of the interlock are satisfied.

The following table shows some examples of the use of time constants:

Identifier	Action	Description
D	"MyTag", T#2s	2 seconds after step activation, the "MyTag" operand is set to "1" and remains at "1" for the duration of the step activation. This does not apply if the duration of the step activation is shorter than 2 seconds. The operand is reset with step deactivation (not latching).
L	"MyTag", T#20s	If the step is active, the "MyTag" operand is set to "1" for 20 seconds. The operand is then reset (not latching). The operand is also reset if the step activation time is less than 20 seconds.

See also

Basics on actions (Page 7378)
Event-dependent actions (Page 7381)
Block calls (Page 7384)
Timers (Page 7385)
Counter operations (Page 7387)
GRAPH programming language (Page 7369)
Permanent instructions (Page 7388)
Conditions (Page 7389)
Interlock and supervision alarms (Page 7389)
Programming actions (Page 7454)
Sequencers (Page 7371)
Steps and transitions (Page 7375)
Block interface of a GRAPH function block (Page 7390)

17.1.4.3 Event-dependent actions (S7-300, S7-400, S7-1500)

Introduction

You have the option of linking an action with an event in order to make the execution of the action dependent on certain conditions. For actions with identifiers "D", "L", and "TF", the linking with events is not possible. If an action is linked with an event, the signal state of the event is detected by edge evaluation. This means that the instructions are only executed in the cycle in which the event occurs.

The following table shows the events that you can link with actions:

Event	Signal evaluation	Description
S1	Positive edge	Step is activated (signal state = "1")
S0	Negative edge	Step is deactivated (signal state = "0")
V1	Positive edge	Supervision condition is satisfied, i.e., fault occurs (signal state = "1")
V0	Negative edge	Supervision condition is no longer satisfied, i.e., fault is eliminated (signal state = "0")
L0	Positive edge	Interlock condition is satisfied, i.e., fault is eliminated (signal state = "1")
L1	Negative edge	Interlock condition is not satisfied, i.e., fault occurs (signal state = "0")
A1	Positive edge	An alarm is acknowledged.
R1	Positive edge	Incoming registration.

The signal states of an event can be used to program additional actions. This allows you to monitor and influence not only the individual steps but also the entire sequential control system.

Actions for which you use the identifiers "ON" (activate step) and "OFF" (deactivate step) must always be linked with an event. The event determines the timing of the activation or deactivation

of the step. If a step is both activated and deactivated in a cycle, the deactivation has higher priority.

You can additionally connect actions for which you use the "S1", "V1", "A1", or "R1" events with an interlock. As a result of this, the actions are only executed if the conditions of the interlock are satisfied.

The following figure shows an example of a step that contains event-dependent actions:

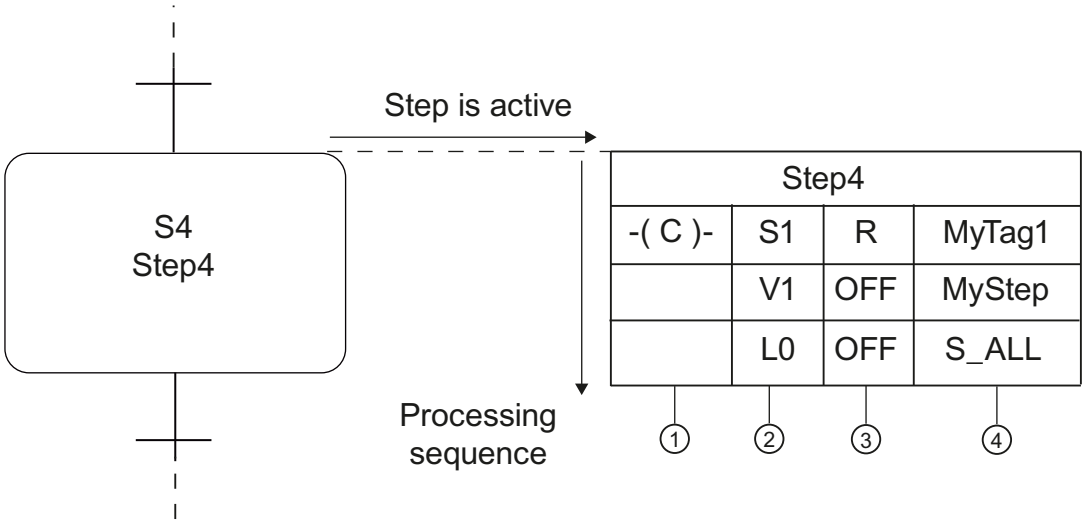

As soon as the "Step4" step becomes active and the interlock condition is satisfied, "MyTag1" is set to "0" and then remains at "0". If a supervision error occurs, the "MyStep" step is deactivated. If an interlock error occurs, all steps are deactivated.

Registration

A registration is an event that is triggered outside the block. Registrations are queried via a positive signal edge at the "REG_S" or "REG_EF" input parameter. If the registration occurs via the "REG_S" input parameter, the event is only forwarded to the active step that is indicated in the "S_NO" output parameter. If the registration is via the "REG_EF" input parameter, the event is forwarded to all currently active steps.

Examples of the signal evaluation of events

The following figure shows the signal evaluation for a step:

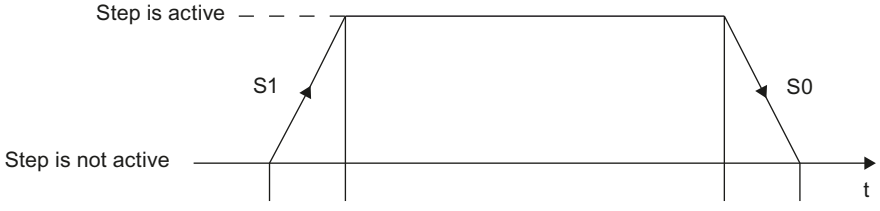

- S1: Step becomes active
- S0: Step becomes inactive

The following figure shows the signal evaluation for a supervision:

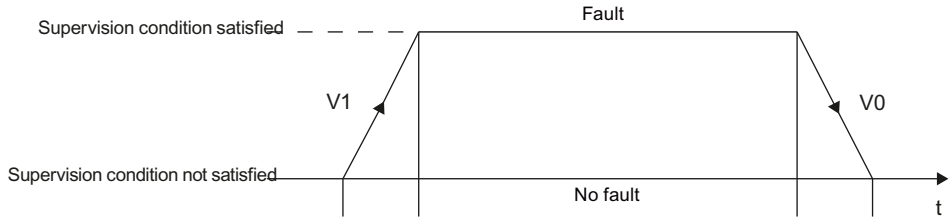

- V1: Supervision error occurs
- V0: Supervision error is eliminated

The following figure shows the signal evaluation for an interlock:

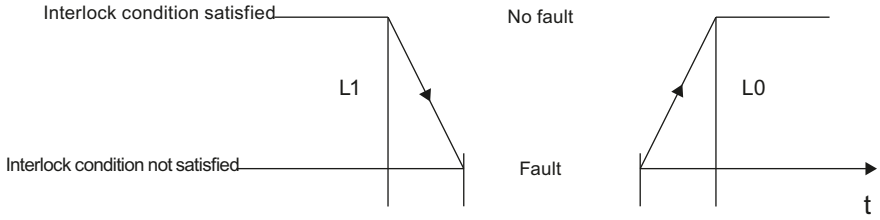

- L1: Interlock condition no longer satisfied (incoming fault)
- L0: Interlock condition satisfied (outgoing fault)

The following figure shows the signal evaluation for an alarm and the registration:

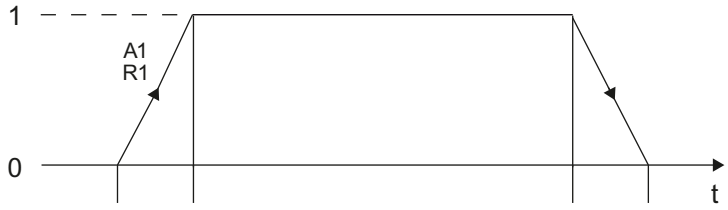

- A1: An alarm is acknowledged.
- R1: Incoming registration (positive edge at input REG_EF/REG_S)

See also

Basics on actions (Page 7378)
Standard actions (Page 7380)
Block calls (Page 7384)
Timers (Page 7385)
Counter operations (Page 7387)
GRAPH programming language (Page 7369)

Permanent instructions (Page 7388)
Conditions (Page 7389)
Interlock and supervision alarms (Page 7389)
Programming actions (Page 7454)
Sequencers (Page 7371)
Steps and transitions (Page 7375)
Block interface of a GRAPH function block (Page 7390)

17.1.4.4 Block calls (S7-300, S7-400, S7-1500)

Introduction

Within actions you can call other function blocks and functions in which you perform certain subtasks. This allows you to better structure your program.

Call of function blocks (FB)

Use the following syntax for calling a function block:
```
CALL "<FBName>", "<DBName>" (list of parameters)
```

The following table shows an example of the call of a function block:

Interlock	Event	Identifier	Action
		N	CALL "FB_KOP", "FB_KOP_DB" (MyInOut := "MyTag2")

The "FB_KOP" function block is called using the "CALL" instruction. The "CALL" instruction is always connected with the "N" identifier. After the name of the called block is the name of the data block containing the data and parameters of the function block. In the list of parameters, the "MyTag2" tag is assigned to the "MyInOut" parameter.

Call of functions (FC)

Use the following syntax for the call of a function:
```
CALL "<FCName>" (list of parameters)
```

The following table shows an example of the call of a function:

Inter-lock	Event	Identifier	Action
		N	CALL "FC_KOP" (MyInput := "MyTag1")

The "FC_KOP" function is called using the "CALL" instruction. In the list of parameters, the "MyTag1" tag is assigned to the "MyInput" parameter.

See also

Block calls (Page 60)
Inserting block calls manually (Page 7480)
Inserting block calls using drag-and-drop operation (Page 7481)
Updating block calls in GRAPH (Page 7483)
Basics on actions (Page 7378)
Standard actions (Page 7380)
Event-dependent actions (Page 7381)
Timers (Page 7385)
Counter operations (Page 7387)
GRAPH programming language (Page 7369)
Permanent instructions (Page 7388)
Conditions (Page 7389)
Interlock and supervision alarms (Page 7389)
Programming actions (Page 7454)
Sequencers (Page 7371)
Steps and transitions (Page 7375)
Block interface of a GRAPH function block (Page 7390)

17.1.4.5 Timers (S7-300, S7-400, S7-1500)

Introduction

You can use timers within actions. Except for "TF", all timers are dependent on an event that determines when the timer is activated. The "TF" timer is activated by the step itself.

When the "TL", "TD", and "TF" timers are used, you must specify the duration. Use the following syntax for this:

```
<Duration> = IWy, QWy, MWy, LWy, DBWy, DIWy; tag of type S5TIME or
WORD; S5T#Constant
```

y = 0 ... 65534

You can also enter the constants as normal time information, such as "5 s".

The following table shows the timers you can use in actions:

Event	Identifier	Description
S1, S0, L1, L0, V1, V0, A1, R1	TL	Extended pulse: As soon as the defined event occurs, the timer starts. During the specified duration, the timer status has signal state "1". After the time expires, the timer status has signal state "0".
S1, S0, L1, L0, V1, V0, A1, R1	TD	Retentive on-delay: As soon as the defined event occurs, the timer starts. During the specified duration, the timer status has signal state "0". After the time expires, the timer status has signal state "1".
S1, S0, L1, L0, V1, V0, A1, R1	TR	Stop timer and reset: As soon as the defined event occurs, the timer is stopped. Timer status and time value are reset to 0.
-	TF	Off delay: The counter status is set to "1" as soon as the step is activated. The timer starts running when the step is deactivated, but the timer status is reset to "0" only after the time expires.

See also

Basics on actions (Page 7378)

Standard actions (Page 7380)

Event-dependent actions (Page 7381)

Block calls (Page 7384)

Counter operations (Page 7387)

GRAPH programming language (Page 7369)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

Programming actions (Page 7454)

Sequencers (Page 7371)

Steps and transitions (Page 7375)

Block interface of a GRAPH function block (Page 7390)

17.1.4.6 Counter operations (S7-300, S7-400, S7-1500)

Introduction

You can use counters within actions. Counters always require an event in order to specify when the counter is activated. This means that the counter is activated when the associated event occurs. You can additionally connect actions for which you use the S1, V1, A1, or R1 events with an interlock. As a result of this, the actions are only executed if the conditions of the interlock are satisfied.

The following table shows the counters you can use in actions:

Event	Identifier	Data type of the operand	Meaning
S1, S0, L1, L0, V1, V0, A1, R1	CS	COUNTER	Set initial counter value: As soon as the defined event occurs, the counter is set to the specified count value. You can specify the counter value as a tag or constant of data type WORD (C#0 to C#999).
S1, S0, L1, L0, V1, V0, A1, R1	CU	COUNTER	Count up: As soon as the defined event occurs, the counter is incremented by "1". The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive edge.
S1, S0, L1, L0, V1, V0, A1, R1	CD	COUNTER	Count down: As soon as the defined event occurs, the counter is decremented by "1". The counter value can be decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer decremented on a positive edge.
S1, S0, L1, L0, V1, V0, A1, R1	CR	COUNTER	Reset counter: As soon as the defined event occurs, the counter is reset to "0".

See also

Basics on actions (Page 7378)

Standard actions (Page 7380)

Event-dependent actions (Page 7381)

Block calls (Page 7384)

Timers (Page 7385)

GRAPH programming language (Page 7369)

Permanent instructions (Page 7388)
Conditions (Page 7389)
Interlock and supervision alarms (Page 7389)
Programming actions (Page 7454)
Sequencers (Page 7371)
Steps and transitions (Page 7375)
Block interface of a GRAPH function block (Page 7390)

17.1.5 Permanent instructions (S7-300, S7-400, S7-1500)

Introduction

Permanent instructions allow you to create program code that is executed before or after the sequencer is processed. This means you can program conditions and block calls that are processed in every cycle independently of the sequencer.

You can create permanent instructions with the LAD and FBD programming languages. A maximum of 32 LAD or FBD instructions can be programmed in one permanent instruction. You can use up to 250 permanent pre-instructions and 250 permanent post-instructions within a GRAPH function block. You transfer the result of logic operation of the conditions to an assignment for which the "Set" and "Reset" memory functions are also available. The operand used (e.g., a bit memory) can be queried in transition, interlock, supervision, and other permanent networks.

See also

"Permanent pre-instructions" view (Page 7423)
"Permanent post-instructions" view (Page 7428)
GRAPH programming language (Page 7369)
Sequencers (Page 7371)
Steps and transitions (Page 7375)
Actions (Page 7378)
Conditions (Page 7389)
Interlock and supervision alarms (Page 7389)
Block interface of a GRAPH function block (Page 7390)

17.1.6 Conditions (S7-300, S7-400, S7-1500)

Introduction

Conditions are binary states of the process that can be linked together as LAD or FBD elements (e.g., normally open contact, normally closed contact, AND logic operation, OR logic operation, comparator) according to the Boolean logic in the ladder/function block diagram. The result of logic operation (RLO) can influence individual actions of a step, the whole step, the advancement to the next step, or the entire sequencer.

You use conditions to program the following components of your GRAPH program:

- Permanent instructions
- Interlocks
- Supervisions
- Transitions

Depending on the network, you can program up to 32 instructions to which a maximum of 32 operands are interconnected. You can have additional information about the utilized tags displayed by showing the tag information.

See also

GRAPH programming language (Page 7369)

Sequencers (Page 7371)

Steps and transitions (Page 7375)

Actions (Page 7378)

Permanent instructions (Page 7388)

Interlock and supervision alarms (Page 7389)

Block interface of a GRAPH function block (Page 7390)

17.1.7 Interlock and supervision alarms (S7-300, S7-400, S7-1500)

Introduction

You can use interlocks and supervisions to detect faults. In order to quickly eliminate these faults, you have the option of specifying alarms that describe the faults in detail. To do this, you first activate the alarm generation and then adapt the alarm texts to meet your requirements. You can also specify that the alarms have to be acknowledged.

See also

Specifying alarms (Page 7485)

GRAPH programming language (Page 7369)

Sequencers (Page 7371)

Steps and transitions (Page 7375)

Actions (Page 7378)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Block interface of a GRAPH function block (Page 7390)

17.1.8 Block interface of a GRAPH function block (S7-300, S7-400, S7-1500)

17.1.8.1 Basics on the block interface of a GRAPH function block (S7-300, S7-400, S7-1500)

Interface parameter sets

In the settings "Options > Settings > PLC programming > GRAPH" you can choose between the following interface parameter sets, each of which provides you with a different set of input and output parameters:

- Minimum interface parameter set
  The minimum interface parameter set contains only the input parameter "INIT_SQ" and no output parameters.
- Standard interface parameter set
  The standard interface parameter set enables operation of the sequencer in various operating modes and provides the option of acknowledging alarms.
- Maximum interface parameter set
  The maximum interface parameter set provides additional diagnostics.

You can manually delete or insert individual parameters in all parameter sets.

Note

When selecting, also take into consideration the available memory space in your CPU, because the memory space required for the GRAPH function block and the associated instance data block increases as the number of parameters increases.

See also:

Input parameters (Page 7393)

Output parameters (Page 7397)

Memory space models for the instance data block (S7-300/400)

For CPUs of the S7-300/400 series, you can additionally specify the memory space model of the instance data block. You have the following options:

- Standard data block
- Minimized data block

You define the number of static parameters via the selection of the memory model. Generally, both memory models store information on the individual steps and transitions as static parameters. A separate structure with detailed information is created for each step and each transition. You receive more detailed information in a standard data block than in a minimized data block. However a standard data block requires considerably more memory space.

The following restrictions apply to a minimized data block:

- Some elements of the steps and transition structures are unavailable.
- The "Skip steps" option is unavailable.
- No alarms are available.
- Step and transition numbers must be consecutive. If not, the steps and transitions are automatically renumbered during compilation.
- No data is generated for the criteria analysis.

See also: Static parameters (Page 7400)

Access to interface parameters

Generally, with an S7-1500 CPU, you do not have write access to the GRAPH-internal interface parameters. However, the following components of the "RT_DATA > MOP" structure in the "Static" section that manage the operating modes allow write access:

- AUTO
- MAN
- TAP
- TOP
- LOCK
- SUP
- OFF
- INIT
- HOLD
- TMS_HALT
- OPS_ZERO
- S_PREV
- S_NEXT
- S_ON
- S_OFF

- T_PUSH
- ACK
- IL_PERM
- T_PERM
- ILP_MAN

These parameters react to a positive signal edge. For S7-300/400 CPUs, also avoid write access to parameters other than those listed. Otherwise, error-free processing of the sequencer cannot be guaranteed.

See also: Other static parameters (Page 7404)

Resetting registration requirements

When errors and faults are registered via the "REG_EF" input parameter, the following differences exist between the CPU families with respect to resetting the registration requirement:

- S7-300/400
  The registration requirement is retained until an interlock or supervision error occurs in a step. Only then is the "R1" event triggered, the associated action executed, and the registration requirement reset.
- S7-1500
  The registration requirement is always reset, regardless of whether interlock or supervision errors occur in the current cycle.

See also

Specifying the interface parameter set (Page 7417)

Specifying the memory space model of the data block (Page 7418)

GRAPH programming language (Page 7369)

Sequencers (Page 7371)

Actions (Page 7378)

Steps and transitions (Page 7375)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

17.1.8.2 Input parameters (S7-300, S7-400, S7-1500)

Overview of the input parameters of a GRAPH function block

The input parameters of a GRAPH function block react to a positive signal edge. The following table provides an overview of the input parameters of a GRAPH function block:

Parameters	Data type	Description	Minimum parameter set	Standard parameter set	Maximum parameter set
OFF_SQ	BOOL	OFF_SEQUENCE: Switch off sequencer, i.e., deactivate all steps Type: Request	-	x	x
INIT_SQ	BOOL	INIT_SEQUENCE: Activate initial steps, reset sequencer Type: Request	x	x	x
ACK_EF	BOOL	ACKNOWLEDGE_ERROR_FAULT: Acknowledge a fault, force advance to next step Type: Request	-	x	x
REG_EF	BOOL	REGISTRATE_ERROR_FAULT: Register all errors and faults Type: Request	-	-	x
ACK_S	BOOL	ACKNOWLEDGE_STEP: Acknowledge step indicated in "S_NO" output parameter Type: Request	-	-	x
REG_S	BOOL	REGISTRATE_STEP: Register step indicated in "S_NO" output parameter Type: Request	-	-	x
HALT_SQ	BOOL	HALT_SEQUENCE: Stop/reactivate sequencer Type: Status, this is reset by the next positive edge	-	-	x
HALT_TM	BOOL	HALT_TIMES: Stop/reactivate all step activation times and time-dependent operations (L and D) of the sequencer Type: Status, this is reset by the next positive edge	-	-	x
ZERO_OP	BOOL	ZERO_OPERANDS: Reset to zero all operands with identifier N, D, L in active steps and do not execute CALL instructions in actions/reactivate operands and CALL instructions Type: Status, this is reset by the next positive edge	-	-	x
EN_IL	BOOL	ENABLE_INTERLOCKS: Disable interlock (sequencer behavior is same as when interlock condition is satisfied)/re-enable Type: Status, this is reset by the next positive edge	-	-	x

Parameters	Data type	Description	Minimum parameter set	Standard parameter set	Maximum parameter set
EN_SV	BOOL	ENABLE_SUPERVISIONS: Disable supervision (sequencer behavior is same as when supervision condition is not satisfied)/re-enable Type: Status, this is reset by the next positive edge	-	-	x
EN_ACKREQ	BOOL	ENABLE_ACKNOWLEDGE_REQUIRED: Activate mandatory acknowledgment Type: Status, this is reset by the next positive edge	-	-	x
EN_SSKIP	BOOL	ENABLE_STEP_SKIPPING: Enable step skipping Type: Status, this is reset by the next positive edge	-	-	x
DISP_SACT	BOOL	DISPLAY_ACTIVE_STEPS: Display active steps only Type: Status, this is reset by the next positive edge at DISP_SEF, DISP_SALL	-	-	x
DISP_SEF	BOOL	DISPLAY_STEPS_WITH_ERROR_OR_FAULT: Display only steps with errors and disrupted steps Type: Status, this is reset by the next positive edge at DISP_SACT, DISP_SALL	-	-	x
DISP_SALL	BOOL	DISPLAY_ALL_STEPS: Display all steps Type: Status, this is reset by the next positive edge at DISP_SACT, DISP_SEF	-	-	x
S_PREV	BOOL	PREVIOUS_STEP:Automatic mode: Page up through the currently active steps, display of step number in "S_NO" parameter Manual mode: Display preceding step in "S_NO" (smaller number) Type: Request	-	x	x
S_NEXT	BOOL	NEXT_STEP: Automatic mode: Page down through the currently active steps, display of step number in "S_NO" parameter Manual mode: Display next step in S_NO (larger number) Type: Request	-	x	x
SW_AUTO	BOOL	SWITCH_MODE_AUTOMATIC: Operating mode switchover: Automatic mode Type: Status, this is reset by the next positive edge at SW_TAP, SW_TOP, SW_MAN	-	x	x
SW_TAP	BOOL	SWITCH_MODE_TRANSITION: Operating mode switchover: Semiautomatic mode Type: Status, this is reset by the next positive edge at SW_AUTO, SW_TOP, SW_MAN	-	x	x

Parameters	Data type	Description	Minimum parameter set	Standard parameter set	Maximum parameter set
SW_TOP	BOOL	SWITCH_MODE_TRANSITION_OR_PUSH: Operating mode switchover: Automatic or semi-automatic mode Type: Status, this is reset by the next positive edge at SW_AUTO, SW_TAP, SW_MAN	-	x	x
SW_MAN	BOOL	SWITCH_MODE_MANUAL: Operating mode switchover: Manual mode, an independent sequence is not initiated Type: Status, this is reset by the next positive edge at SW_AUTO, SW_TAP, SW_TOP	-	x	x
S_SEL	INT	STEP_SELECT: Select step number for "S_NO" output parameter in manual mode, enable/disable with "S_ON", "S_OFF" Type: N/A	-	x	x
S_SELOK	BOOL	STEP_SELECT_OK: Apply value in "S_SEL" for "S_NO" output parameter Type: Request	-	-	x
S_ON	BOOL	STEP_ON: Manual mode: Activate displayed step Type: Request	-	x	x
S_OFF	BOOL	STEP_OFF: Manual mode: Deactivate displayed step Type: Request	-	x	x
T_PREV	BOOL	PREVIOUS_TRANSITION: Show previous valid transition in "T_NO" output parameter Type: Request	-	-	x
T_NEXT	BOOL	NEXT_TRANSITION: Show next valid transition in "T_NO" output parameter Type: Request	-	-	x
T_PUSH	BOOL	PUSH_TRANSITION: Transition advances to next step, when condition is satisfied and "T_PUSH" (edge) Requirement: Automatic mode or manual mode Type: Request	-	x	x

Parameters	Data type	Description	Minimum parameter set	Standard parameter set	Maximum parameter set
EN_LMODE (S7-1500 only as of block version V2.0)	BOOL	ENABLE_LEARNING_MODE: Enable the learning mode Type: Status, this is reset by the next positive edge	-	-	x
RESET_CRIT (S7-1500 only; GRAPH FB as of block version V5.0)	BOOL	RESET_CRITERIA: Reset initial values of interlocks and transitions Type: Request	-	-	x

See also

Resetting initial values (Page 7490)
Basics on the block interface of a GRAPH function block (Page 7390)
Output parameters (Page 7397)
Static parameters (Page 7400)
Specifying the interface parameter set (Page 7417)
Specifying the memory space model of the data block (Page 7418)
GRAPH programming language (Page 7369)
Sequencers (Page 7371)
Steps and transitions (Page 7375)
Actions (Page 7378)
Permanent instructions (Page 7388)
Conditions (Page 7389)
Interlock and supervision alarms (Page 7389)

17.1.8.3 Output parameters (S7-300, S7-400, S7-1500)

Overview of the output parameters of a GRAPH function block

The following table provides an overview of the output parameters of a GRAPH function block:

Parameters	Data type	Description	Minimum parameter set	Standard parameter set	Maximum parameter set
S_NO	INT	STEP_NUMBER: Display of step number	-	x	x
S_MORE	BOOL	MORE_STEPS: Additional steps are active	-	x	x
S_ACTIVE	BOOL	STEP_ACTIVE: Displayed step is active	-	x	x
S_TIME	TIME	STEP_TIME: Step activation time	-	-	x
S_TIMEOK	TIME	STEP_TIME_OK: Step activation time is error free	-	-	x
S_CRITLOC (S7-300/400 or S7-1500 as of block version V4.0)	DWORD	STEP_CRITERIA_INTERLOCK: Interlock criteria bits	-	-	x
S_CRITLOCERR (S7-300/400 or S7-1500 as of block version V4.0)	DWORD	S_CRITERIA_IL_LAST_ERROR: Interlock criteria bits for event L1	-	-	x
S_CRITSUP (S7-300/400 only)	DWORD	STEP_CRITERIA_SUPERVISION: Supervision criteria bits	-	-	x
S_STATE	WORD	STEP_STATE: Step state bits	-	-	x
T_NO	INT	TRANSITION: Valid transition number	-	-	x
T_MORE	BOOL	MORE_TRANSITIONS: More valid transitions available for display	-	-	x

Parameters	Data type	Description	Minimum parameter set	Standard parameter set	Maximum parameter set
T_CRIT (S7-300/400 or S7-1500 as of block version V4.0)	DWORD	TRANSITION_CRITERIA: Status of the max. 32 LAD/FBD elements of the transition criteria bits in the current process cycle You can find additional information on the order of the criteria bits in the DWORD here: Example of order of the list (Page 7491)	-	-	x
T_CRITOLD (S7-300/400 or S7-1500 as of block version V4.0)	DWORD	T_CRITERIA_LAST_CYCLE: Status of the max. 32 LAD/FBD elements of the transition criteria bits of the last cycle You can find additional information on the order of the criteria bits in the DWORD here: Example of order of the list (Page 7491)	-	-	x
T_CRITFLT (S7-300/400 or S7-1500 as of block version V4.0)	DWORD	T_CRITERIA_LAST_FAULT: Status of the max. 32 LAD/FBD elements of the transition criteria bits for event V1 from the process cycle when the supervision error occurred in the previous step. You can find additional information on the order of the criteria bits in the DWORD here: Example of order of the list (Page 7491)	-	-	x
ERROR	BOOL	INTERLOCK_ERROR: Interlock error (any step)	-	-	x
FAULT	BOOL	SUPERVISION_FAULT: Supervision error (any step)	-	-	x
ERR_FLT	BOOL	IL_ERROR_OR_SV_FAULT: General fault	-	x	x
SQ_ISOFF	BOOL	SEQUENCE_IS_OFF: Sequencer is switched off (no step is active)	-	-	x

Parameters	Data type	Description	Minimum parameter set	Standard parameter set	Maximum parameter set
SQ_HALTED	BOOL	SEQUENCE_IS_HALTED: Sequencer is stopped	-	-	x
TM_HALTED	BOOL	TIMES_ARE_HALTED: Timers are stopped	-	-	x
OP_ZEROED	BOOL	OPERANDS_ARE_ZEROED: Operands are reset	-	-	x
IL_ENABLED	BOOL	INTERLOCK_IS_ENABLED: Interlock is taken into consideration	-	-	x
SV_ENABLED	BOOL	SUPERVISION_IS_ENABLED: Supervision is taken into consideration	-	-	x
ACKREQ_ENABLED	BOOL	ACKNOWLEDGE_REQUIRED_IS_ENABLED: Mandatory acknowledgement is enabled	-	-	x
SSKIP_ENABLED	BOOL	STEP_SKIPPING_IS_ENABLED: Step skipping is enabled	-	-	x
SACT_DISP	BOOL	ACTIVE_STEPS_WERE_DISPLAYED: Only active steps are displayed in "S_NO"	-	-	x
SEF_DISP	BOOL	STEPS_WITH_ERROR_FAULT_WERE_DISPLAYED: Only steps with errors and faults are displayed in "S_NO"	-	-	x
SALL_DISP	BOOL	ALL_STEPS_WERE_DISPLAYED: All steps are displayed in "S_NO"	-	-	x
AUTO_ON	BOOL	AUTOMATIC_IS_ON: Display of automatic mode	-	x	x
TAP_ON	BOOL	T_AND_PUSH_IS_ON: Display of semiautomatic mode	-	x	x
TOP_ON	BOOL	T_OR_PUSH_IS_ON: Display of semiautomatic mode	-	x	x

Parameters	Data type	Description	Minimum parameter set	Standard parameter set	Maximum parameter set
MAN_ON	BOOL	MANUAL_IS_ON: Display of manual mode	-	x	x
LMODE_ENABLED (S7-1500 only as of block version V2.0)	BOOL	LEARNING_MODE_IS_ENABLED: The learning mode is enabled	-	-	x

See also

Basics on the block interface of a GRAPH function block (Page 7390)
Input parameters (Page 7393)
Static parameters (Page 7400)
Specifying the interface parameter set (Page 7417)
Specifying the memory space model of the data block (Page 7418)
GRAPH programming language (Page 7369)
Sequencers (Page 7371)
Steps and transitions (Page 7375)
Actions (Page 7378)
Permanent instructions (Page 7388)
Conditions (Page 7389)
Interlock and supervision alarms (Page 7389)

17.1.8.4 Static parameters (S7-300, S7-400, S7-1500)

Static parameters of a step (S7-300, S7-400, S7-1500)

Overview of the static parameters of a step

The following table shows the parameters that are created for each step of the GRAPH function block in the "Static" section of the block interface:

Component	Description	Data type	Memory space minimized DB	Standard DB
S1	Step is activated	BOOL	x	x
L1	Interlock is outgoing	BOOL	x	x
V1	Supervision incoming	BOOL	x	x
R1	Reserved	BOOL	x	x
A1	Error is acknowledged	BOOL	x	x
S0	Step is deactivated	BOOL	x	x

Component	Description	Data type	Memory space minimized DB	Standard DB
L0	Incoming interlock	BOOL	x	x
V0	Outgoing supervision	BOOL	x	x
X	Step is active	BOOL	x	x
LA	Interlock is unsatisfied	BOOL	x	x
VA	Pending supervision	BOOL	x	x
RA	Reserved	BOOL	x	x
AA	Reserved	BOOL	x	x
SS	System internal	BOOL	x	x
LS	Direct result of the programmed interlock	BOOL	x	x
VS	Direct result of the programmed supervision	BOOL	x	x
SNO	User step number	INT	-	x
T	Total step activation time	TIME	x	x
U	Uninterrupted step activation time	TIME	-	x
T_MAX (S7-1500 only as of block version V2.0)	Maximum step activation time	TIME	-	x
T_WARN (S7-1500 only as of block version V2.0)	Warning time	TIME	-	x
CRIT_LOC (S7-300/400) (S7-1500 as of block version V4.0)	State of the max. 32 LAD/FBD elements in interlock in the current process cycle	DWORD	-	x
CRIT_LOC_ERR (S7-300/400 only) (S7-1500 as of block version V4.0)	Copy of CRIT_LOC, if interlock is outgoing	DWORD	-	x
CRIT_SUP (S7-300/400 only)	State of the max. 32 LAD/FBD elements in supervision in the current process cycle	DWORD	-	x
SM	System internal	BOOL	-	x
LP (S7-300/400 only)	System internal	BOOL	x	x
LN (S7-300/400 only)	System internal	BOOL	x	x

Component	Description	Data type	Memory space minimized DB	Standard DB
VP (S7-300/400 only)	System internal	BOOL	x	x
VN (S7-300/400 only)	System internal	BOOL	x	x
H_IL_ERR	System internal	BYTE	-	x
H_SV_FLT	System internal	BYTE	-	x
RESERVED (S7-300/400 only)	Reserve	DWORD	-	x

Access to components

Write access is not recommended because the parameter values influence the sequence. Reading access takes place via the step name followed by the structure element name. In conditions or actions, e.g. the following accesses are thereby possible:

- TROCKNEN.X
- TROCKNEN.T

See also

Resetting initial values (Page 7490)

Basics on the block interface of a GRAPH function block (Page 7390)

Input parameters (Page 7393)

Output parameters (Page 7397)

Static parameters of a transition (Page 7403)

Other static parameters (Page 7404)

Specifying the interface parameter set (Page 7417)

Specifying the memory space model of the data block (Page 7418)

GRAPH programming language (Page 7369)

Sequencers (Page 7371)

Steps and transitions (Page 7375)

Actions (Page 7378)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

Static parameters of a transition (S7-300, S7-400, S7-1500)

Overview of the static parameter of a transition

The following table shows the parameters that are created for each transition of the GRAPH function block in the "Static" section of the block interface:

Component	Description	Data type	Memory space minimized DB	Standard DB
TV	Transition is valid	BOOL	x	x
TT	Transition is satisfied	BOOL	x	x
TS	Transition is switching	BOOL	x	x
CF_IV (S7-300/400 only)	Entry "CRIT_FLT" is invalid	BOOL	-	x
TNO	Display of the user-defined transition number	INT	-	x
CRIT (S7-300/400 only) (S7-1500 as of block version V4.0)	State of the max. 32 LAD/FBD transition elements in the current process cycle	DWORD	-	x
CRIT_OLD (S7-300/400 only) (S7-1500 as of block version V4.0)	State of the max. 32 LAD/FBD transition elements in the previous process cycle Which element ultimately triggered the relaying.	DWORD	-	x
CRIT_FLT (S7-300/400 only) (S7-1500 as of block version V4.0)	Copy of "CRIT", if an error occurs	DWORD	-	x

Access to components

Write access is not recommended because the parameter values influence the sequence. Reading access takes place via the transition name followed by the structure element name. In conditions or actions, e.g. the following accesses are thereby possible:

- VENTIL_EIN.TT
- VENTIL_EIN.CRIT

See also

Basics on the block interface of a GRAPH function block (Page 7390)

Input parameters (Page 7393)

Output parameters (Page 7397)

Static parameters of a step (Page 7400)

Other static parameters (Page 7404)

Specifying the interface parameter set (Page 7417)

Specifying the memory space model of the data block (Page 7418)

GRAPH programming language (Page 7369)

Sequencers (Page 7371)

Steps and transitions (Page 7375)

Actions (Page 7378)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

Other static parameters (S7-300, S7-400, S7-1500)

Overview

For an S7-1500 CPU the following static parameters are available in the "RT_DATA" structure.

The static parameters include the step and transition parameters as well as the following information:

- Internal parameters "VERSION", "S_DISPLAY", "S_SEL_OLD", "S_DISPIDX", "T_DISPIDX"
- "MOP >Operating mode" component
- "SQ_FLAGS > Sequence bit memory"
- Component "Clock speeds"

Note

Additional parameters are contained in a GRAPH-specific instance data block. They are used however only for internal application. The use of these parameters in the program is not recommended.

Internal parameters

The following table shows the internal parameters that are located in the "Static" section. They are used to manage the step and transition numbers:

Component	Description	Data type	Length in bytes	Memory space minimized DB	Standard DB
VERSION	Block version	STRING[10]	2	x	x
S_DISPLAY	Copy of the "S_NO" output parameter	INT	2	x	x
S_SEL_OLD	Step number from the "S_SEL" input parameter from the last CPU cycle	INT	2	x	x
S_DISPIDX	Step index for the step in "S_NO" output parameter	BYTE	1	x	x
T_DISPIDX	Transition index for the transition in "T_NO" output parameter	BYTE	1	x	x

Operating mode

The following table shows the "MOP" structure components in the section "Static" that manage the operating modes:

Component	Description	Data type	Memory space minimized DB	Standard DB
AUTO	Switch to automatic mode	BOOL	x	x
MAN	Switch to manual mode	BOOL	x	x
TAP	Switch to "Jog" mode	BOOL	x	x
TOP	Switch to "Automatic" or "Advance" mode	BOOL	x	x
ACK_S	Requirement: Acknowledge step that is indicated in "S_NO" output parameter	BOOL	x	x
REG_S	Requirement: Register step that is indicated in "S_NO" output parameter	BOOL	x	x
T_PREV	Requirement: Output previous valid transition in "T_NO" output parameter	BOOL	x	x
T_NEXT	Requirement: Output next valid transition in "T_NO" output parameter	BOOL	x	x
LOCK	Turn on/turn off interlock handling	BOOL	x	x
SUP	Turn on/turn off supervision handling	BOOL	x	x
ACKREQ	Requirement: Acknowledge step that is indicated in "S_NO" output parameter	BOOL	x	x
SSKIP	Status: "Skip step" is activated	BOOL	x	x
OFF	Deactivate sequencer	BOOL	x	x
INIT	Activate initial steps	BOOL	x	x

Component	Description	Data type	Memory space minimized DB	Standard DB
HOLD	Stop/resume sequence	BOOL	x	x
TMS_HALT	Stop/resume timers	BOOL	x	x
OPS_ZERO	Switch off/switch on actions	BOOL	x	x
SACT_DISP	Status: Display active steps only	BOOL	x	x
SEF_DISP	Status: Display only steps with errors and disrupted steps	BOOL	x	x
SALL_DISP	Status: Display all steps	BOOL	x	x
S_PREV	"Automatic" mode: Display previous simultaneously active step in "S_NO" output parameter "Manual" mode: Display next smaller step number	BOOL	x	x
S_NEXT	"Automatic" mode: Display next simultaneously active step in "S_NO" output parameter "Manual" mode: Display next higher step number	BOOL	x	x
S_SELOK	Requirement: Output step number that is indicated in "S_SEL" input parameter in "S_NO" output parameter	BOOL	x	x
S_ON	"Manual" mode: Activate displayed step	BOOL	x	x
S_OFF	"Manual" mode: Deactivate displayed step	BOOL	x	x
T_PUSH	"Jog" mode: Transition switches, when the satisfied condition is applied with "T_PUSH"	BOOL	x	x
REG	Requirement: Register all interlock and supervision windows	BOOL	x	x
ACK	Acknowledgement of the fault	BOOL	x	x
IL_PERM	Permanent interlock processing	BOOL	x	x
T_PERM	Permanent transition processing	BOOL	x	x
ILP_MAN	Permanent interlock processing in "Manual" mode	BOOL	x	x

Component	Description	Data type	Memory space minimized DB	Standard DB
LMODE (S7-1500 only, as of block version V2.0)	Learning mode is active	BOOL	-	-
RESET_CRIT (S7-1500 only; as of block version V5.0)	RESET_CRITERIA: Reset initial values of interlocks and transitions Type: Request	BOOL	-	-

Clock pulses (S7-300/400 only)

The following table shows the "TICKS" structure components in the section "Static" that manages the information for activation time calculation. This information can be evaluated only internally in the system.

Component	Description	Data type	Memory space minimized DB	Standard DB
DELTA	Time difference between cycles	TIME	x	x
OLD	10 ms clock counter value in last cycle	TIME	x	x
NEW	10 ms clock counter value in this cycle	TIME	x	x

Sequence bit memory

The following table shows the "SQ_FLAGS" structure components in the section "Static" that monitor the internal sequence status:

Component	Description	Data type	Memory space minimized DB	Standard DB
ERR_FLT	General fault	BOOL	x	x
ERROR	Interlock error	BOOL	x	x
FAULT	Supervision error	BOOL	x	x
RT_FAIL	Runtime error	BOOL	x	x
NO_SNO	Selected step does not exist	BOOL	x	x
NF_OFL	Too many steps that have to be activated or deactivated	BOOL	x	x
SA_OFL	Too many active steps	BOOL	x	x
TV_OFL	Too many valid transitions	BOOL	x	x
MSG_OFL	Too many alarms	BOOL	x	x
NO_SWI	Do not switch in this cycle	BOOL	x	x
CYC_OP	Cyclic processing of the sequencer after initialization	BOOL	x	x

Component	Description	Data type	Memory space minimized DB	Standard DB
AS_MSG	Alarms during runtime enabled or disabled by instruction	BOOL	x	x
AS_SEND	Send alarms from "WR_USMSG" or only enter in diagnostics buffer	BOOL	x	x
SQ_BUSY	Internal edge memory bit for sequence processing	BOOL	x	x
SA_BUSY	Internal edge memory bit for sequence processing	BOOL	x	x
AS_SIG	Edge memory bit for alarms from "Alarm_S" and "Alarm_SQ"	BOOL	x	x

Access to components

Write access is not recommended because the parameter values influence the sequence. Read access is however possible.

See also

Basics on the block interface of a GRAPH function block (Page 7390)

Input parameters (Page 7393)

Output parameters (Page 7397)

Static parameters of a step (Page 7400)

Static parameters of a transition (Page 7403)

Specifying the interface parameter set (Page 7417)

Specifying the memory space model of the data block (Page 7418)

GRAPH programming language (Page 7369)

Sequencers (Page 7371)

Steps and transitions (Page 7375)

Actions (Page 7378)

Permanent instructions (Page 7388)

Conditions (Page 7389)

Interlock and supervision alarms (Page 7389)

17.1.8.5 Extending the standard behavior of GRAPH (S7-1500)

Basic information on extending the standard behavior of GRAPH (S7-1500)

Introduction

GRAPH function blocks contain both visible and invisible internal parameters in the block sections in the "Static" section. Visible parameters are:

- Internal data area "RT_DATA"
- Transition structure "Trans1"
- Step structure "Step1"

Invisible parameters are:

- Description and status arrays

If required, as in STEP 7 Classic, you can also access the invisible parameters. This requires that the GRAPH block as of version 4.0 and higher is used. In addition to the GRAPH function block, an SCL function block is also required as an expansion. Accesses to the individual arrays within the extension block are via offsets, which are available in the "RT_DATA" structure.

Arrays

Among other things you can access the following arrays via the extension block for GRAPH:

Array	Description
SINI	Array of initial steps
LSTT	Assignment of jump transition to jump step
ATAJ	Assignment of all alternative transitions of an alternative junction
ATAB	Assignment of all alternative transitions of an alternative branch
PSTT	Assignment of direct preceding step to transition
NSTT	Assignment of direct following step to transition
ASSJ	Assignment of all simultaneous steps of a simultaneous junction
ASSB	Assignment of all simultaneous steps of a simultaneous branch
PTTS	Assignment of direct preceding transition to step
NTTS	Assignment of direct following transition to step
SW_SQTS	Assignment of step to sequential part
SWITCH	Data for safe activation mode
TVX	Array of valid transitions
TTX	Array of satisfied transitions
TSX	Array of switching transitions
S00X	Array of steps to be deactivated (second last cycle)
SOFFX	Array_1 of the steps to be deactivated
SONX	Array_1 of the steps to be activated
SAX	Array of active steps

Array	Description
SERRX	Array of disturbed steps
SMX	Array of marked steps
S0X	Array_2 of the steps to be deactivated
S1X	Array_2 of the steps to be activated

Creating an extension function block and assigning it to a GRAPH FB (S7-1500)

You can create an extension block and assign it to the GRAPH function block in order to access the GRAPH-internal parameters. The extension block can be used as a single instance or multi-instance.

Requirement

The current project is a V14 SP1 project.

Procedure

To create an extension block and assign it to a GRAPH FB, proceed as follows:

1. Create a new function block (FB) in the "GRAPH" programming language.
2. Open the block properties of the new GRAPH FB.
3. Under "General > Version" set the version of the GRAPH FB to >= V4.0.

   Note

   Note on project compatibility

   If your current project is not yet a V14 SP1 project, a corresponding message will be displayed. In order to be able to access the GRAPH-internal parameters via the extension block, your project must be upgraded to a V14 SP1 project. After upgrading, however, you are no longer able to work with a TIA Portal version lower than V14 SP1.

4. Create a new function block (FB) in the "SCL" programming language. You can assign any name to this FB. This FB serves as an extension block.
5. Enter the following in/out parameters in the extension block in the block interface in the "InOut" section:
   - "io_RT_Data", data type "G7_RTDataPlus_V4" (e.g. for the block version V4.0)
   - "io_G7T", data type "Array[*] of G7_TransitionPlus_V4" (e.g. for the block version V4.0)
   - "io_G7S", data type "Array[*] of G7_StepPlus_V4" (e.g. for the block version V4.0)
   - "io_G7Arrays", data type "Array[*] of USInt"
6. Open the block properties of the GRAPH FB.
7. Enter the name of the SCL function block under "Attributes > Name of extension block".

8. Instantiate the extension block either as a single instance or multi-instance:
   - Single instance:
     Create an instance data block of the type extension block. Enter a parameter with extension block as the data type in the GRAPH FB in the block interface in the "InOut" section.

     Note

     If you want to use an extension block for several GRAPH FBs, enter the extension block in the attributes of the corresponding GRAPH FBs. However, create a separate instance data block of the extension block for each GRAPH FB. To define different behavior for the various GRAPH FBs, you can use the "Static" section in the block interface of the extension block.

   - Multi-instance:
     Enter a parameter with extension block as the data type in the GRAPH FB in the block interface in the "Static" section.
9. Call the GRAPH FB in any other block, the instance data block for the GRAPH FB is automatically created.
10. If you have instantiated the extension block as a single instance, provide the corresponding parameter of the block call, which you defined in the "InOut" section of the GRAPH FB, with the instance data block of the extension block.

Result

The extension block is prepared and you can access the GRAPH-internal parameters within it. The individual description and status arrays are contained in the collective array "io_G7Arrays". Access to them is via the offsets. An access of this type could appear, for example, as follows:

```
#t_NumberOfActiveSteps :=
#io_G7Arrays[#io_RT_DATA.OFFSETS.SAX_OFFSET] - 1;
```

17.2 Settings for GRAPH (S7-300, S7-400, S7-1500)

17.2.1 Overview of the settings for GRAPH (S7-300, S7-400, S7-1500)

Overview

The following table shows the settings that you can make:

Group	Area	Setting	Description
View		Length of step and transition names	Specifies whether step and transition names are to be displayed in one or two lines. The "Show two lines" setting allows longer names to be viewed.
Load		Turn off sequence before downloading DB	All active steps are deactivated before the instance data block is loaded.
Default for time monitoring		Uninterrupted step activation time (U)	Activation time for the time monitoring of a step by the "Greater than uninterrupted step activation time" instruction (CMP>U).
		Step activation time (T)	Activation time for the time monitoring of a step by the "Greater than step activation time" instruction (CMP>T).
		Maximum step activation time (T_MAX)	Specifies the maximum time allowed to elapse after the activation of a step. You can monitor this time with the instruction "Greater than maximum step activation time" (CMP>T_MAX). You can also have this value determined by the system. See also: Basics of step time monitoring (Page 7465) This setting is relevant only for CPUs of the S7-1500 series.
		Warning time (T_WARN)	Specifies the time that must elapse after the activation of a step before a warning is output. You can monitor this time with the instruction "Greater than warning time" (CMP>T_WARN). You can also have this value determined by the system. See also: Basics of step time monitoring (Page 7465) This setting is relevant only for CPUs of the S7-1500 series.

Group	Area	Setting	Description
Default settings for new blocks	Editor	Language in networks	Selection of the programming language for the programming of conditions.
	Interface	Minimum interface parameters	Inserts the minimum parameter set when creating new GRAPH function blocks.
		Default interface parameters	Inserts the default parameter set when creating new GRAPH function blocks.
		Maximum interface parameters	Inserts the maximum number of parameters into the block interface when new GRAPH function blocks are created.
	Compile	Create minimized DB	Creates an instance data block with a reduced set of static parameters when compiling. This limits the functionality, however. You can change this setting again subsequently for individual blocks. See also: Basics on the block interface of a GRAPH function block (Page 7390)
	Sequence properties	Skip steps	Skips steps in which the next transition condition is already satisfied when the step is activated. In this case the step does not become active. You can change this setting subsequently for individual blocks.
		Acknowledge supervision alarms	Specifies that supervision errors must be acknowledged before the sequencer will continue to be processed.
		Permanent processing of all interlocks in manual mode	Allows the processing of all interlocks in manual mode.
		Lock operating mode selection	Prevents the manual change of operating mode via the "Testing" task card. You can change this setting subsequently for individual blocks.
		Set ENO automatically	Displays the additional "ENO" operand for S7-1500 CPUs when testing with program status.
	Alarm properties	Enable alarms	Enables the interlock and supervision alarms for new GRAPH function blocks. You can change this setting subsequently for individual blocks.
Synchronization		Preceding transition satisfied	Marks the steps whose preceding transition condition is satisfied when determining possible synchronization points. The subsequent transition condition is not satisfied in this case.
		Interlock condition satisfied	Marks the steps whose interlock condition is satisfied when determining possible synchronization points. The subsequent transition condition is satisfied in this case.

See also

Overview of block properties (Page 6921)
Changing the settings (Page 7414)
Specifying the programming language in networks (Page 7416)
Specifying the interface parameter set (Page 7417)
Specifying the memory space model of the data block (Page 7418)
Modifying settings for internal parameters (Page 7419)

17.2.2 Changing the settings (S7-300, S7-400, S7-1500)

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Change the settings.

Result

The change is applied and does not have to explicitly saved.

See also

Overview of the settings for GRAPH (Page 7412)
Specifying the programming language in networks (Page 7416)
Specifying the interface parameter set (Page 7417)
Specifying the memory space model of the data block (Page 7418)
Modifying settings for internal parameters (Page 7419)

17.2.3 GRAPH versions (S7-300, S7-400, S7-1500)

17.2.3.1 Overview of GRAPH versions (S7-300, S7-400, S7-1500)

Overview

The following table lists the available versions for GRAPH function blocks and the functionality these contain:

Version	Description
V1.0	Includes the basic functionality for GRAPH function blocks.
V2.0	Provides you with the learning mode for determining the values of "T_MAX" and "T_Warn" in addition to the basic functionality.
V3.0 *	Includes the functionality of V1.0 and V2.0 and also extends the range of functions for the PLC code view and improves the processing of alarms.
V4.0 *	Trigger value acquisition Requirement for enabling initial value acquisition. There are two new static parameters for "STEPx" and three new parameters for "Transx". These are needed in order to save the signal state of operands or the result of comparators from a step or transition. Extending the standard behavior of GRAPH
V5.0 *	Required for: • Translating GRAPH step and GRAPH transition names • Resetting initial values of interlocks and transitions
V6.0 *	Enables multi-language subcategories for interlock and supervision errors, and GRAPH warnings.

* Only available for S7-1500 as of firmware V2.0

The GRAPH function block is automatically generated with version 6.0 in projects that are created with TIA Portal V15.1.

See also

Defining the version of the GRAPH function block (Page 7415)

17.2.3.2 Defining the version of the GRAPH function block (S7-300, S7-400, S7-1500)

You can use various versions for a GRAPH function block. Different functions are available to you depending on the selected version. You can specify the version either via the project tree or in the inspector window.

Specify version of the GRAPH function block via the project tree

To specify the version for a GRAPH function block, follow these steps:

1. Right-click the GRAPH function block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.

3. Select the "General" entry in the area navigation.
4. In the "Block" area, select the version you want to use from the drop-down list.
5. Confirm your selection with "OK".

Specify the version of the GRAPH function block in the inspector window

To specify the version for a GRAPH function block in the inspector window, follow these steps:

1. Open the GRAPH function block whose version you want to change.
2. Open the "Properties" tab in the Inspector window.
3. Select the "General" entry in the area navigation in the inspector window.
4. In the "Block" area, select the version you want to use from the drop-down list.

See also

Overview of GRAPH versions (Page 7415)

17.2.4 Specifying the programming language in networks (S7-300, S7-400, S7-1500)

For GRAPH function blocks, you cannot change the programming language of the block. However, you can specify the programming language of the networks for conditions. You can choose here between LAD and FBD. You can specify the programming language either in the project tree or in the inspector window.

Specifying the programming language in networks in the project tree

To specify the programming language in networks in the project tree, follow these steps:

1. Right-click the GRAPH function block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   A dialog containing the properties of the block opens.
3. Select the "General" entry in the area navigation.
4. In the "Block" area, select the programming language you want to use from the "Language in networks" drop-down list.
5. Confirm your selection with "OK".

Specifying the programming language in networks in the inspector window

To specify the programming language in networks in the inspector window, follow these steps:

1. Open the GRAPH function block whose programming language you want to change.
2. Open the "Properties" tab in the inspector window.

3. Select the "General" entry in the area navigation in the inspector window.
4. In the "Block" area, select the programming language you want to use from the "Language in networks" drop-down list.

See also

Overview of the settings for GRAPH (Page 7412)

Changing the settings (Page 7414)

Specifying the interface parameter set (Page 7417)

Specifying the memory space model of the data block (Page 7418)

Modifying settings for internal parameters (Page 7419)

17.2.5 Specifying the interface parameter set (S7-300, S7-400, S7-1500)

You can specify the interface parameter set for GRAPH function blocks as follows:

- Globally for all new GRAPH function blocks
- Locally for an open GRAPH function block

The local setting has priority over the globally setting.

Note

You can manually add or delete individual input and output parameters in the block interface at any time.

Specifying the interface parameter set globally for all new GRAPH function blocks

In order to specify the interface parameter set globally for all new GRAPH function blocks, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. If you want to use the minimum interface parameter set, select the "Minimum interface parameters" option.
4. If you want to use the default interface parameter set, select the "Default interface parameters" option.
5. If you want to use the maximum interface parameter set, select the "Maximum interface parameters" option.

Specifying the interface parameter set locally for an open GRAPH function block

In order to specify the interface parameter set for an open GRAPH function block, follow these steps:

1. If you want to use the minimum interface parameter set, select the "Minimum interface parameters" command in the "Edit > Interface parameters" menu.
2. If you want to use the default interface parameter set, select the "Default interface parameters" command in the "Edit > Interface parameters" menu.
3. If you want to use the maximum interface parameter set, select the "Maximum interface parameters" command in the "Edit > Interface parameters" menu.

The block interface of the GRAPH function block is adapted.

See also

Basics on the block interface of a GRAPH function block (Page 7390)

Overview of the settings for GRAPH (Page 7412)

Changing the settings (Page 7414)

Specifying the programming language in networks (Page 7416)

Specifying the memory space model of the data block (Page 7418)

Modifying settings for internal parameters (Page 7419)

17.2.6 Specifying the memory space model of the data block (S7-300, S7-400)

You can specify the memory space model for GRAPH function blocks as follows:

- Globally for all new GRAPH function blocks
- Locally for an open GRAPH function block

The local setting has priority over the globally setting.

Note

GRAPH-specific parameters that are not available in the minimized data block cannot be manually inserted in the block interface.

Specifying the memory space model globally for all new GRAPH function blocks

In order to specify the memory space model globally for all new GRAPH function blocks, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.

3. If you want to use a minimized data block, select the "Create minimized DB" option.
4. If you want to use a data block with the standard memory space model, clear the "Create minimized DB" option.

Specifying the memory space model locally for an open GRAPH function block

In order to specify the memory space model for an open GRAPH function block, follow these steps:

1. Open the "Properties" tab in the inspector window.
2. Select the "Attributes" entry in the area navigation in the inspector window.
3. If you want to use a minimized data block, select the "Create minimized DB" option in the "Compile" group.
4. If you want to use a data block with the standard memory space model, clear the "Create minimized DB" option in the "Compile" group.

The interface of the GRAPH block is adapted.

See also

Basics on the block interface of a GRAPH function block (Page 7390)

Overview of the settings for GRAPH (Page 7412)

Changing the settings (Page 7414)

Specifying the programming language in networks (Page 7416)

Specifying the interface parameter set (Page 7417)

Modifying settings for internal parameters (Page 7419)

17.2.7 Modifying settings for internal parameters (S7-300, S7-400, S7-1500)

You can declare the internal parameters of a GRAPH block as visible and accessible from HMI. For S7-1500 CPUs, you can also declare the internal parameters as retentive. If you declare the internal parameters as retentive, their values are retained even after an operating state change from "STOP" to "RUN", for example following a power failure. As a result, the sequencer remains in the current state and continues running from the point at which it was interrupted after the change to "RUN".

Parameters that you have declared yourself in the interface of a GRAPH block are not affected by these settings.

Declaring internal parameters as visible and accessible from HMI

Proceed as follows to declare all internal parameters of a GRAPH block as visible and accessible from HMI:

1. Open the GRAPH block for which you want to change the setting.
2. In the "Edit" menu, select the "Internal parameters visible/accessible from HMI" check box. Clear this check box if you want to reset the setting.

Declaring internal parameters as retentive (S7-1500)

Proceed as follows to declare all internal parameters of a GRAPH block as retentive:

1. Open the GRAPH block for which you want to change the setting.
2. In the "Edit" menu, select the "Retentive internal parameters" check box. Clear this check box if you want to reset the setting.

Note

Points to note for S7-1500 CPUs with a firmware version lower than V1.5

When you declare the internal parameters of a GRAPH block as retentive, the sequence control remains in the current status and continues from the point at which it was interrupted after a change to "RUN". Please note that supervisions should not be used if you activate the new parameter "Retentive internal parameters" in a GRAPH block.

Set data in standard range as retentive (S7-300/400)

To set the data within the standard range as retentive, follow these steps:

1. Open the "Program blocks" folder in the project navigator.
2. Right-click on the instance data block of the GRAPH block.
3. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the block opens.
4. Click "Attributes" in the area navigation.
5. Select the check box "Set data in standard range as retentive".
6. Confirm your entries with "OK".

See also

Overview of the settings for GRAPH (Page 7412)

Changing the settings (Page 7414)

Specifying the programming language in networks (Page 7416)

Specifying the interface parameter set (Page 7417)

Specifying the memory space model of the data block (Page 7418)

17.3 Programming window of GRAPH (S7-300, S7-400, S7-1500)

17.3.1 Overview of the programming window (S7-300, S7-400, S7-1500)

Function

The programming window is the area, where you create the GRAPH program. In addition to the actual work area, it contains a navigation view that allows you to easily access all areas of the GRAPH function block.

You can perform the following tasks in the programming window:

- Programming of permanent pre-instructions and post-instructions
- Programming of the sequencers
- Specifying of the interlock and supervision alarms

Depending on what you want to program, you can switch between the following views:

- Permanent pre-instructions (Page 7423)
- Sequence view (Page 7424)
- Single step view (Page 7426)
- Permanent post-instructions (Page 7428)
- Alarm view (Page 7429)

The work area and the available instructions and favorites vary depending on the view.

Structure of the programming window

The following figure shows the programming window of GRAPH in the sequence view:

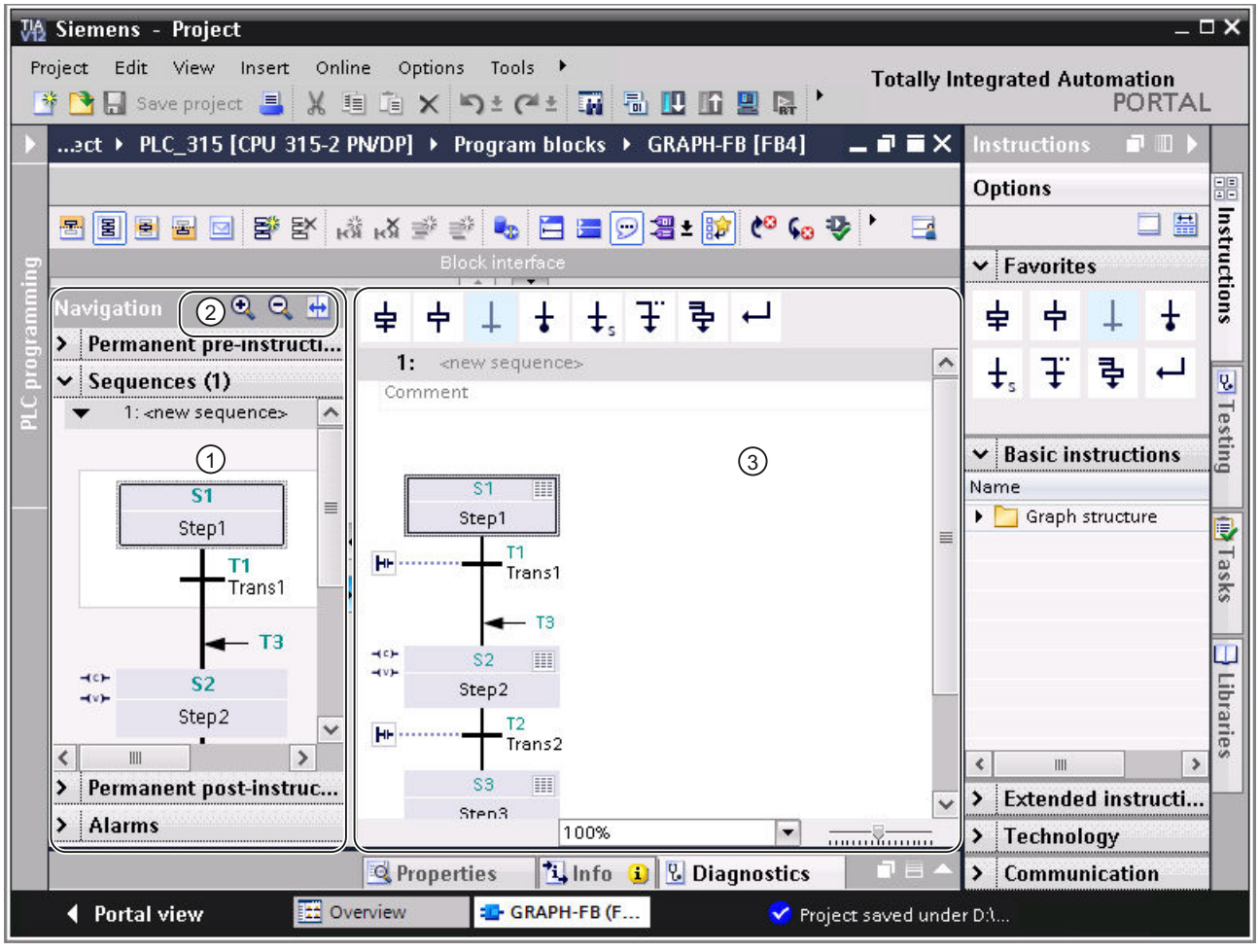

① Navigation view

② Navigation toolbar

③ Work area

Navigation view

The navigation view contains the following pallets you can use to open the following views:

- Permanent pre-instructions
- Sequence view
- Permanent post-instructions
- Alarm view

In addition, the navigation view shows graphic overviews of the permanent instructions and sequencers and offers you basic processing options via the shortcut menu.

You can double-click a step in the navigation view to switch between the sequence and single step views.

Navigation toolbar

You can perform the following actions in the navigation toolbar:

- Zoom in or zoom out of elements within the navigation
  You can use the zoom in and zoom out buttons to zoom in on or zoom out from the graphic elements of the permanent instructions and sequences within the navigation view. You can set the zoom factor separately for each pallet. When the GRAPH function block is closed, any changes to the zoom factor setting are discarded. If you want to retain the zoom factor, you can do so via the "Remember layout" button.
- Synchronizing navigation
  When the button is enabled, the navigation and the work area are synchronized so that the same elements are always displayed. When you disable the button, you can display different objects in navigation and work area.

Work area

You program the individual components of your sequencer in the work area. You can display the GRAPH program in different views for this. You can scale these views using a zoom function.

17.3.2 "Permanent pre-instructions" view (S7-300, S7-400, S7-1500)

Function of the "Permanent pre-instructions" view

In the "Permanent pre-instructions" view, you can program instructions that are executed before the sequencer is processed.

Structure of the "Permanent pre-instructions" view

The following figure shows an example of the "Permanent pre-instructions" view with the FBD programming language:

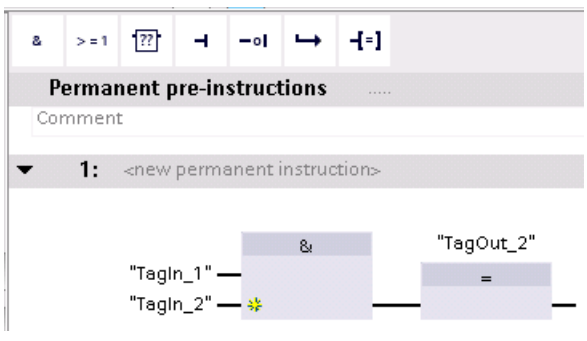

See also

Permanent instructions (Page 7388)
Overview of the programming window (Page 7421)
Sequence view (Page 7424)
Single step view (Page 7426)
"Permanent post-instructions" view (Page 7428)

17.3.3 Sequence view (S7-300, S7-400, S7-1500)

Function of the sequence view

The sequence view shows you the structure of your sequencer in an easy to read format and enables you to add the following elements to it:

- Steps
- Transitions
- Jumps
- Branches
- Sequence ends

In addition, you can expand steps and transitions at the click of a mouse in order to display or edit the actions of a step and the conditions of a transition.

Structure of the sequence view

The following figure shows an example of the sequence view:

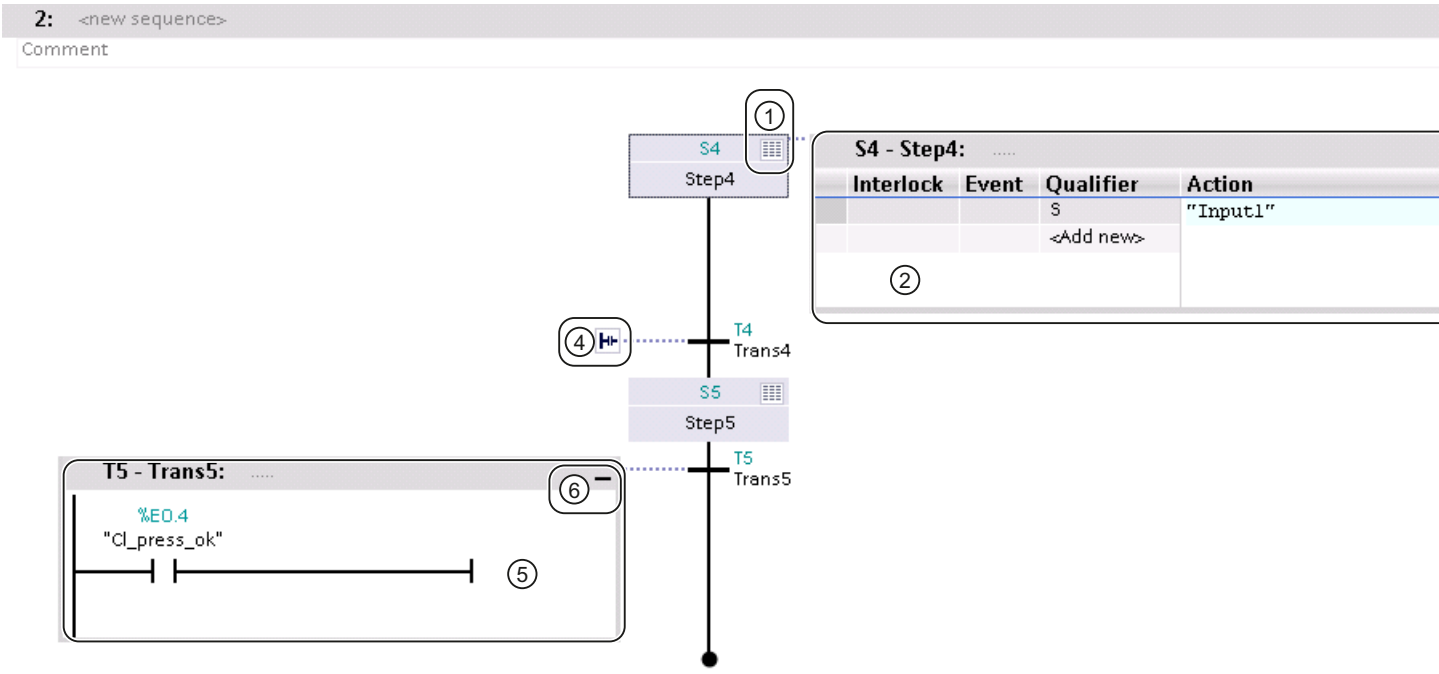

① Button for expanding or reducing a step

② Step in the expanded view

③ Button for reducing a step

④ Button for expanding a transition

⑤ Transition in the expanded view

⑥ Button for reducing a transition

See also

Overview of the programming window (Page 7421)

"Permanent pre-instructions" view (Page 7423)

Single step view (Page 7426)

"Permanent post-instructions" view (Page 7428)

17.3.4 Single step view (S7-300, S7-400, S7-1500)

Function of the single step view

The single step view provides you the option of programming the following elements of a step:

- Interlock
- Supervision
- Actions
- Transition

In addition, you can specify the title of the step and a comment for the step

Note

You can program the actions of the step and the associated transition additionally in the sequence view.

Structure of the single step view

The following figure shows a step in the single step view:

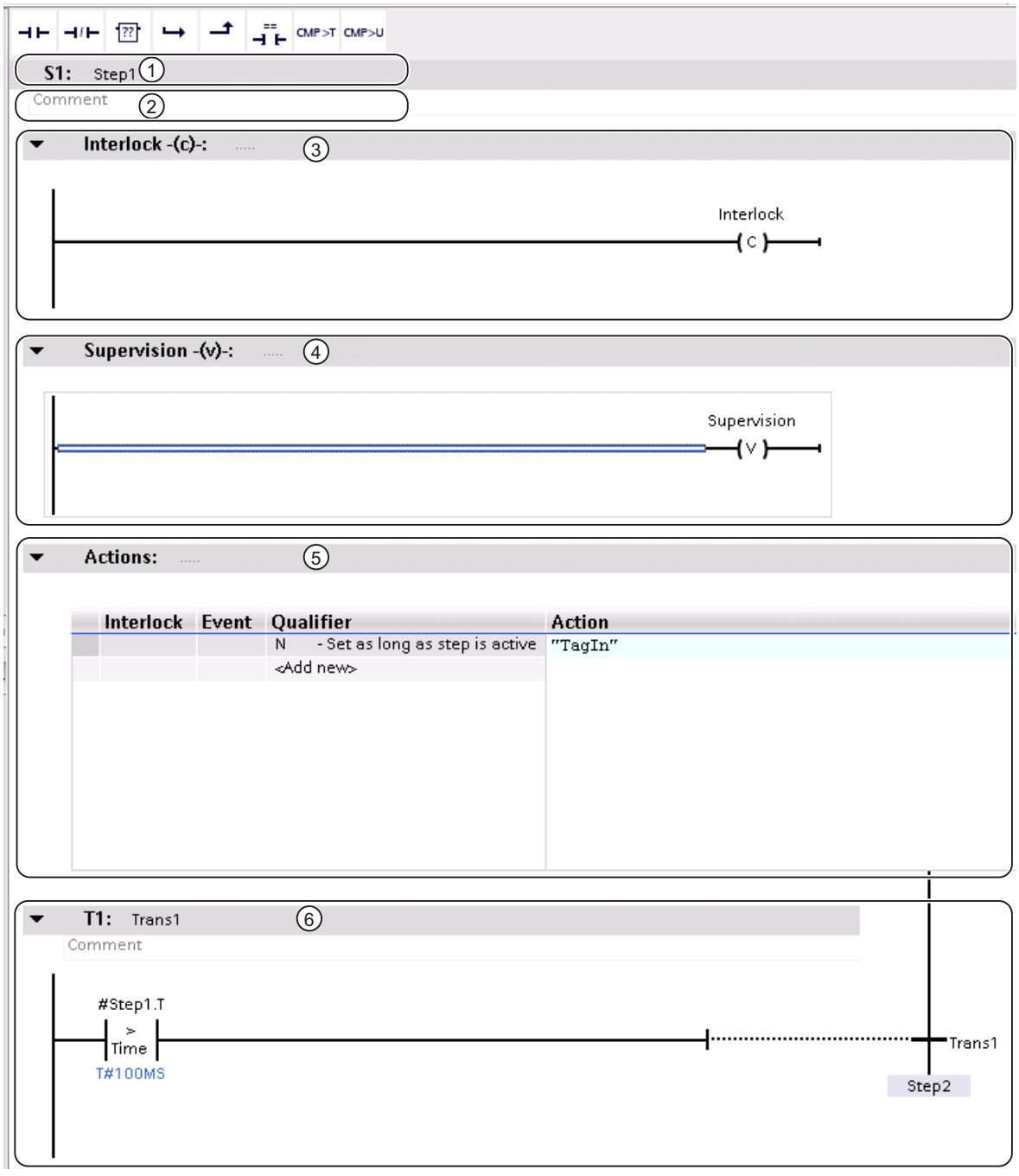

① Title of the step

② Comment for the step

③ Interlock

④ Supervision

⑤ Actions

⑥ Transition

See also

Step elements (Page 7376)

Overview of the programming window (Page 7421)

"Permanent pre-instructions" view (Page 7423)

Sequence view (Page 7424)

"Permanent post-instructions" view (Page 7428)

17.3.5 "Permanent post-instructions" view (S7-300, S7-400, S7-1500)

Function of the "Permanent post-instructions" view

In the "Permanent post-instructions" view, you can program instructions that are executed after the sequencer is processed.

Structure of the "Permanent post-instructions" view

The following figure shows an example of the "Permanent post-instructions" view with the LAD programming language:

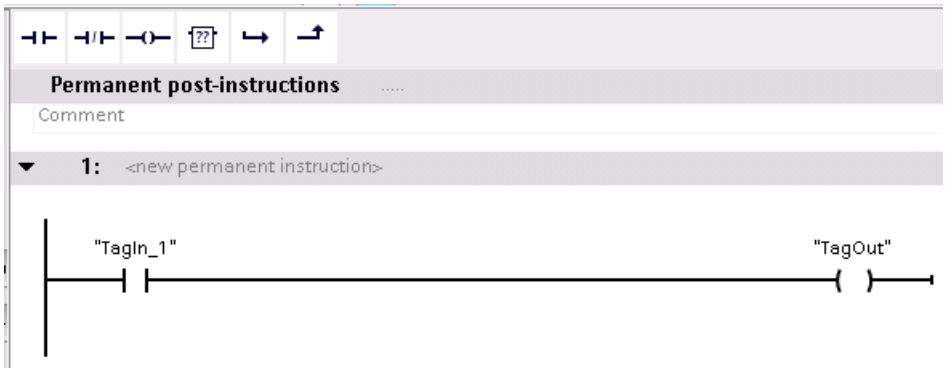

See also

Permanent instructions (Page 7388)

Overview of the programming window (Page 7421)

"Permanent pre-instructions" view (Page 7423)

Sequence view (Page 7424)

Single step view (Page 7426)

17.3.6 Alarm view (S7-300, S7-400, S7-1500)

Function of the alarm view

You can perform the following tasks in the alarm view:

- Enable the alarm display
- Specify category enablers
- Select display classes
- Select category and subcategories for interlocks
- Select category and subcategories for supervisions
- Select category and subcategories for GRAPH warnings

Structure of the alarm view

The figure below shows the alarm view of a GRAPH function block:

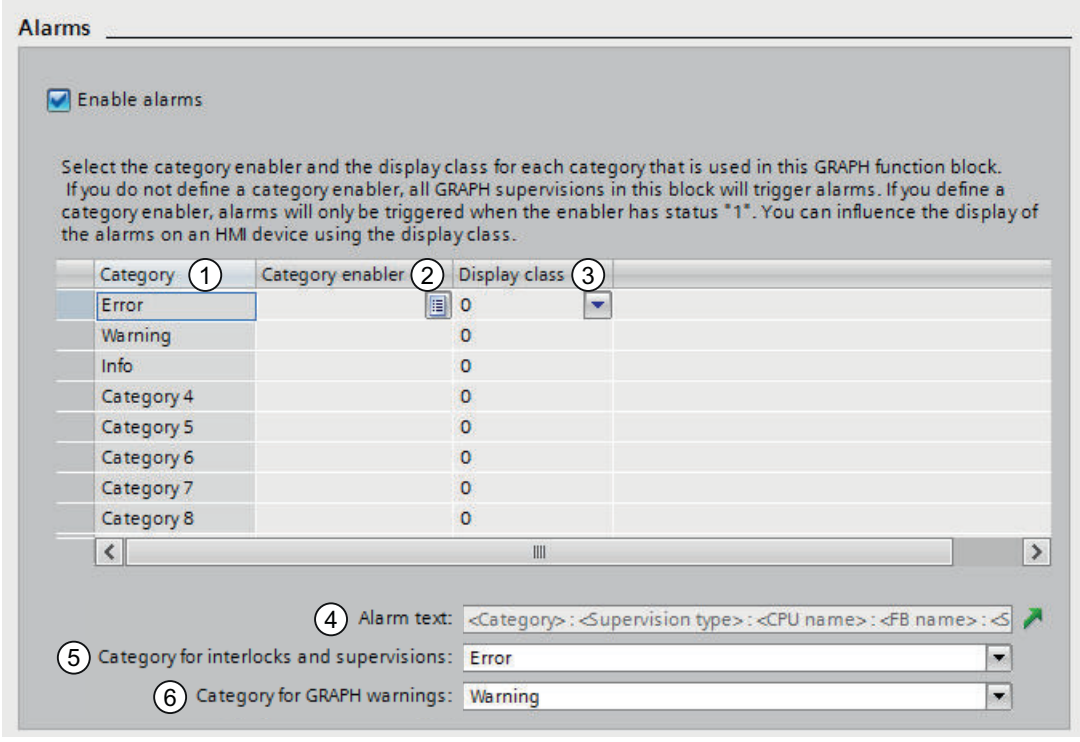

Numbering	Explanation	Additional information
1	You can use categories to divide faults into different priority levels such as errors, warnings, etc.	You can find additional information on the categories here: Categories (Page 7870)
2	You can use the category enabler to define an enabler (global Boolean tag) for the ProDiag function block and for each individual category contained therein and use this to activate or deactivate the supervisions of this category.	For additional information on the enabler, refer to: Using the category enabler (Page 7865)
3	You can use the display class to define which supervisions are to be displayed on an HMI device.	You can find additional information on the display classes here: Setting display classes (Page 7867)
4	Alarm text structure of GRAPH supervisions With the green arrow, you can jump directly to the global ProDiag supervision settings, where you can specify the alarm text structure.	You can find additional information on creating the alarm text structure here: Structure of a GRAPH supervision alarm (Page 7881)
5	You can specify one category and two subcategories for interlocks. You can select from a maximum of eight categories.	You can find additional information here: Programming interlocks (Page 7463)
6	You can specify one category and two subcategories for supervisions. You can select from a maximum of eight categories.	You can find additional information here: Programming supervisions (Page 7464)
7	You can specify one category and two subcategories for GRAPH warnings that are output by the "T_WARN" instruction. You can select from a maximum of eight categories.	You can find additional information here: Using step time monitoring (Page 7465)

See also

Specifying alarms (Page 7485)

Overview of the programming window (Page 7421)

"Permanent pre-instructions" view (Page 7423)

Sequence view (Page 7424)

Single step view (Page 7426)

"Permanent post-instructions" view (Page 7428)

17.4 Using Favorites in GRAPH (S7-300, S7-400, S7-1500)

17.4.1 Adding GRAPH elements to Favorites (S7-300, S7-400, S7-1500)

Requirement

- A GRAPH function block is open in one of the following views:
  - Permanent pre-instructions
  - Sequence view
  - Single step view
  - Permanent post-instructions
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the GRAPH programming window.

Procedure

In order to add instructions to Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to Favorites.
4. Drag the instruction to the "Favorites" pane or to the Favorites area in the GRAPH programming window.

Note

To additionally display the Favorites in the programming window, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Inserting GRAPH elements using Favorites (Page 7432)

Removing GRAPH elements from Favorites (Page 7432)

17.4.2 Inserting GRAPH elements using Favorites (S7-300, S7-400, S7-1500)

Requirement

- A GRAPH function block is open in one of the following views:
  - Permanent pre-instructions
  - Sequence view
  - Single step view
  - Permanent post-instructions
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In Favorites, click the instruction to be inserted.

Note

To additionally display the Favorites in the programming window, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding GRAPH elements to Favorites (Page 7431)

Removing GRAPH elements from Favorites (Page 7432)

17.4.3 Removing GRAPH elements from Favorites (S7-300, S7-400, S7-1500)

Requirement

- A GRAPH function block is open in one of the following views:
  - Permanent pre-instructions
  - Sequence view
  - Single step view
  - Permanent post-instructions
- Favorites are available.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click the instruction in the Favorites area that you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the programming window, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding GRAPH elements to Favorites (Page 7431)

Inserting GRAPH elements using Favorites (Page 7432)

17.5 Programming the structure of a sequencer (S7-300, S7-400, S7-1500)

17.5.1 Inserting the sequencers (S7-300, S7-400, S7-1500)

When you open a GRAPH function block, the "Sequences" pane is expanded by default in the navigation view. A new GRAPH function block already contains a sequencer with one step and one transition. You can insert additional sequencers at any time.

Requirement

A GRAPH function block is open.

Procedure

To insert a new sequencer, follow these steps:

1. In the navigation view, right-click the title bar of the "Sequences" pane or the title bar of a sequencer that is already present.
2. Select the "Insert sequence" command in the shortcut menu.
   The sequencer is inserted after the selected sequencer. If no sequencer is selected, it is inserted before the first sequencer.

See also

Basics on sequencers (Page 7371)

Elements of a sequencer (Page 7372)

Renaming sequencers (Page 7434)

Adding comments for sequencers (Page 7435)

Inserting elements in a sequencer (Page 7436)

Ending sequencers (Page 7437)

Deleting sequencers (Page 7437)

Using steps and transitions (Page 7438)

Using branches (Page 7448)

Using jumps (Page 7451)

17.5.2 Renaming sequencers (S7-300, S7-400, S7-1500)

Requirement

A GRAPH function block is open in the sequence view.

Procedure

To rename a sequencer, follow these steps:

1. Click the title bar of the sequencer in the sequence view.
   The "<new sequence>" placeholder is provided by default.
2. Enter the new name of the sequencer.

See also

Basics on sequencers (Page 7371)
Elements of a sequencer (Page 7372)
Inserting the sequencers (Page 7434)
Adding comments for sequencers (Page 7435)
Inserting elements in a sequencer (Page 7436)
Ending sequencers (Page 7437)
Deleting sequencers (Page 7437)
Using steps and transitions (Page 7438)
Using branches (Page 7448)
Using jumps (Page 7451)

17.5.3 Adding comments for sequencers (S7-300, S7-400, S7-1500)

Requirement

A GRAPH function block is open in the sequence view.

Procedure

To enter a comment for a sequencer, follow these steps:

1. Click inside the comment area of the sequencer in the sequence view.
   A "Comment" placeholder is provided by default.
2. Enter a comment on the sequencer.

See also

Basics on sequencers (Page 7371)
Elements of a sequencer (Page 7372)
Inserting the sequencers (Page 7434)
Renaming sequencers (Page 7434)
Inserting elements in a sequencer (Page 7436)

Ending sequencers (Page 7437)
Deleting sequencers (Page 7437)
Using steps and transitions (Page 7438)
Using branches (Page 7448)
Using jumps (Page 7451)

17.5.4 Inserting elements in a sequencer (S7-300, S7-400, S7-1500)

Requirement

A sequencer is open in the sequence view.

Procedure

To insert elements in a sequencer, follow these steps:

1. Right-click the point in the sequencer where you want to insert a new element.
2. Select the element you want to insert in the "Insert element" submenu of the shortcut menu. Only the elements that are permitted at the selected point in the sequencer are available for selection.

Or:

1. Open the "Basic instructions" pane of the "Instructions" task card.
2. Open the "Graph structure" folder.
3. Drag the element you want to insert to the desired point in the sequencer. Note here that not all elements can be inserted at all points in the sequencer. Allowed points in the sequencer are indicated by a small green square.

See also

Basics on sequencers (Page 7371)
Elements of a sequencer (Page 7372)
Inserting the sequencers (Page 7434)
Renaming sequencers (Page 7434)
Adding comments for sequencers (Page 7435)
Ending sequencers (Page 7437)
Deleting sequencers (Page 7437)
Using steps and transitions (Page 7438)
Using branches (Page 7448)
Using jumps (Page 7451)

17.5.5 Ending sequencers (S7-300, S7-400, S7-1500)

Every sequencer must be ended with a jump to another step in the same sequencer or another sequencer or with the "Sequence end" element. A jump to another step enables cyclic processing of the sequencer.

Requirement

A sequencer is open in the sequence view.

Procedure

To end a sequence, follow these steps:

1. Right-click the double arrow at the end of the sequencer.
2. Select the "Sequence end" element in the "Insert element" submenu of the shortcut menu.

Or:

1. Right-click the double arrow at the end of the sequencer.
2. Select the "Jump" element in the "Insert element" submenu of the shortcut menu.
   An overview containing all available steps of the GRAPH function block opens.
3. Select a step as the jump destination.

See also

Basics on sequencers (Page 7371)

Elements of a sequencer (Page 7372)

Inserting the sequencers (Page 7434)

Renaming sequencers (Page 7434)

Adding comments for sequencers (Page 7435)

Inserting elements in a sequencer (Page 7436)

Deleting sequencers (Page 7437)

Using steps and transitions (Page 7438)

Using branches (Page 7448)

Using jumps (Page 7451)

17.5.6 Deleting sequencers (S7-300, S7-400, S7-1500)

You can delete sequencers again at any time. However, the GRAPH function block must always contain at least one sequencer. Note that when you delete sequencers, this may result in the deletion of jump destinations.

Procedure

To delete a sequencer, follow these steps:

1. Right-click the title bar of the sequencer in the navigation view.
2. Select the "Delete" command in the shortcut menu.

See also

Basics on sequencers (Page 7371)
Elements of a sequencer (Page 7372)
Inserting the sequencers (Page 7434)
Renaming sequencers (Page 7434)
Adding comments for sequencers (Page 7435)
Inserting elements in a sequencer (Page 7436)
Ending sequencers (Page 7437)
Using steps and transitions (Page 7438)
Using branches (Page 7448)
Using jumps (Page 7451)

17.5.7 Using steps and transitions (S7-300, S7-400, S7-1500)

17.5.7.1 Insert steps and transitions (S7-300, S7-400, S7-1500)

When you insert a new GRAPH function block in your project, it already contains a sequencer with one step and one transition. You can insert additional steps and transitions into the sequencer.

Note the following rules when inserting steps and transitions:

- Steps and transitions must always alternate. This means that you cannot insert two steps or two transitions in a row.
- It follows from this that you can insert steps and transitions only at the end of a branch or at positions at which a step or transition is missing.
- A simultaneous branch must always start with a step.
- An alternative branch must always start with a transition.
- You can insert a step and a transition together as a pair. Because this method ensures that the step is followed by a transition, it is possible to do this at any position in the sequencer.

Requirement

A GRAPH function block is open in the sequence view.

Inserting a step

To insert a step, follow these steps:

1. Right-click a position in the sequencer where it is permitted to insert a step.
2. Select the "Insert element > Step" command from the shortcut menu.

Or:

1. Drag the "Step" element from Favorites or the "Instructions" task card to the appropriate position in the sequencer.

Inserting a transition

To insert a transition, follow these steps:

1. Right-click a position in the sequencer where it is permitted to insert a transition.
2. Select the "Insert element > Transition" command from the shortcut menu.

Or:

1. Drag the "Transition" element from Favorites or the "Instructions" task card to the appropriate position in the sequencer.

Inserting a step and transition together as a pair

To insert a step and a transition together as a pair, follow these steps:

1. Right-click the position in the sequencer where you want to insert the step and the transition.
2. Select the "Insert element > Step and transition" command from the shortcut menu.

Or:

1. Drag the "Step and transition" element from Favorites or the "Instructions" task card to the appropriate position in the sequencer.

See also

Steps and transitions (Page 7375)

Specifying and canceling the initial step (Page 7440)

Renumbering steps and transitions (Page 7441)

Renaming steps and transitions (Page 7443)

Adding comments for steps (Page 7444)

Copying and inserting steps and transitions (Page 7444)

Skip steps (Page 7445)

Deleting steps and transitions (Page 7446)

17.5.7.2 Specifying and canceling the initial step (S7-300, S7-400, S7-1500)

When you insert a new GRAPH function block in your project, it already contains a sequencer with one step and one transition. By default, this step is specified as the initial step. This means that the sequencer is processed starting with this step. You can define a different step as the initial step at any time. In simultaneous branches, it is also permitted to have multiple initial steps. However, the maximum number of initial steps is 250. You can change an initial step to a normal step subsequently by canceling it.

Requirement

A GRAPH function block is open in the sequence view.

Specifying an initial step

To specify a step as an initial step, follow these steps:

1. Right-click the step you want to specify as an initial step.
2. Select the "Initial step" check box in the shortcut menu.

Canceling an initial step

To cancel an initial step and use it as a normal step again, follow these steps:

1. Right-click the step you want to specify as an initial step.
2. Clear the "Initial step" check box in the shortcut menu.

See also

Steps and transitions (Page 7375)
Insert steps and transitions (Page 7438)
Renumbering steps and transitions (Page 7441)
Renaming steps and transitions (Page 7443)
Adding comments for steps (Page 7444)
Copying and inserting steps and transitions (Page 7444)
Skip steps (Page 7445)
Deleting steps and transitions (Page 7446)

17.5.7.3 Renumbering steps and transitions (S7-300, S7-400, S7-1500)

When you insert steps and transitions in your sequencer, the next available number is assigned. You can change this number subsequently. You have the following two options for this:

- You manually assign new numbers for the steps and transitions you want to renumber. Note that the same number must not be assigned more than once.
- You perform program-supported renumbering This method allows you to have all steps and transitions of a branch, sequencer, or block to be renumbered automatically. Observe the following differences when using this method:
  - Renumbering within a branch:
    Within the branch, all steps or transitions starting from the selected step or transition are renumbered. The renumbering ends at the start of a new branch.
  - Renumbering within a sequencer:
    Within the sequencer, all steps or transitions are renumbered. The first step or the first transition of the sequencer is assigned the start number you entered. Note that steps in other sequencers may also be renumbered in the process if duplicate numbers would otherwise result.
  - Renumbering within a block:
    Within the GRAPH block, all steps or transitions are renumbered. The first step or the first transition within the first sequencer of the block is assigned the start number you entered.

  You can also renumber only the steps or only the transitions.

Note

Note, however, that this option is not available for GRAPH function blocks for which you have selected the "Create minimized DB" attribute.

Requirement

A GRAPH function block is open in the sequence view.

Manual renumbering of steps and transitions

To manually renumber steps or transitions, follow these steps:

1. Select the step whose number you want to change.
2. Click the number of the step.
3. Enter the new number.

Program-supported renumbering of steps and transitions within a branch

For program-supported renumbering of steps or transitions within a branch, follow these steps:

1. Right-click the step or transition where you want to start renumbering the branch.
2. Select the "Renumber" command in the shortcut menu.
   The "Renumber" dialog opens.

3. If you want to renumber the steps, select the "Steps" check box and enter the number to be assigned to the selected step.
4. If you want to renumber the transitions, select the "Transitions" check box and enter the number to be assigned to the transition following the selected step.
5. Select the "Branch" option button in the "Renumber in" area.
6. Click "OK" to confirm.

Program-supported renumbering of steps and transitions within a sequencer

For program-supported renumbering of steps or transitions within a sequencer, follow these steps:

1. Right-click a step or transition within the sequencer for which you want to carry out renumbering.
2. Select the "Renumber" command in the shortcut menu.
   The "Renumber" dialog opens.
3. If you want to renumber the steps, select the "Steps" check box and enter the number to be assigned to the first step of the sequencer.
4. If you want to renumber the transitions, select the "Transitions" check box and enter the number to be assigned to the first transition of the sequencer.
5. Select the "Sequence" option button in the "Renumber in" area.
6. Click "OK" to confirm.

Program-supported renumbering of steps and transitions within a GRAPH function block

For program-supported renumbering of steps or transitions within a GRAPH function block, follow these steps:

1. Right-click a step or transition within the GRAPH function block for which you want to carry out renumbering.
2. Select the "Renumber" command in the shortcut menu.
   The "Renumber" dialog opens.
3. If you want to renumber the steps, select the "Steps" check box and enter the number to be assigned to the first step of the GRAPH function block.
4. If you want to renumber the transitions, select the "Transitions" check box and enter the number to be assigned to the first transition of the GRAPH function block.
5. Select the "Block" option button in the "Renumber in" area.
6. Click "OK" to confirm.

See also

Steps and transitions (Page 7375)

Insert steps and transitions (Page 7438)

Specifying and canceling the initial step (Page 7440)

Renaming steps and transitions (Page 7443)
Adding comments for steps (Page 7444)
Copying and inserting steps and transitions (Page 7444)
Skip steps (Page 7445)
Deleting steps and transitions (Page 7446)

17.5.7.4 Renaming steps and transitions (S7-300, S7-400, S7-1500)

When you insert steps and transitions into your sequencer, they are initially named "Step<StepNumber>" and "Trans<StepNumber>". You can change these names in order to make your program easier to understand. Note that the same name must not be assigned more than once.

Requirement

A GRAPH function block is open in the sequence view.

Renaming a step

To rename a step, follow these steps:

1. Select the step whose name you want to change.
2. Click the step name.
3. Enter a new name.

Renaming a transition

To rename a transition, follow these steps:

1. Double-click the transition name you want to change.
2. Enter a new name.

See also

Steps and transitions (Page 7375)
Insert steps and transitions (Page 7438)
Specifying and canceling the initial step (Page 7440)
Renumbering steps and transitions (Page 7441)
Adding comments for steps (Page 7444)
Copying and inserting steps and transitions (Page 7444)
Skip steps (Page 7445)
Deleting steps and transitions (Page 7446)

17.5.7.5 Adding comments for steps (S7-300, S7-400, S7-1500)

You have the option of adding comments for steps. For example, you can use comments to indicate the purpose of the step or to reference particular details.

Requirement

A step is open in the single step view.

Procedure

To add a comment for a step, follow these steps:

1. Click "Comment" in the comment area.
   The "Comment" text is deleted and an insertion point appears.
2. Enter the comment.

See also

Steps and transitions (Page 7375)

Insert steps and transitions (Page 7438)

Specifying and canceling the initial step (Page 7440)

Renumbering steps and transitions (Page 7441)

Renaming steps and transitions (Page 7443)

Copying and inserting steps and transitions (Page 7444)

Skip steps (Page 7445)

Deleting steps and transitions (Page 7446)

17.5.7.6 Copying and inserting steps and transitions (S7-300, S7-400, S7-1500)

You can copy steps and transitions and insert them in the same sequencer or a different sequencer. This allows you to reduce the programming effort required in cases where a new step is to perform similar tasks as an existing step. Note the following particularities:

- When steps and transitions are copied, the alternating sequence of a step and a transition must also be retained. For this reason, placeholders are inserted ("Missing Step", "Missing Transition"), if two steps or two transitions in a row result after the insertion operation.
- A renaming and renumbering of the inserted element is carried out to prevent duplicate names and numbers.

Requirement

A GRAPH function block is open in the sequence view.

Copying steps and transitions

To copy a step or a transition, follow these steps:

1. To copy a step, right-click the step you want to copy. To copy a transition, right-click the transition you want to copy.
2. Select the "Copy" command in the shortcut menu.

Insert steps and transitions

To insert a step or transition, follow these steps:

1. Right-click the position in the sequencer where you want to insert the step or transition.
2. Select the "Insert" command in the shortcut menu.
3. If necessary, replace the placeholders with valid elements.

See also

Steps and transitions (Page 7375)

Insert steps and transitions (Page 7438)

Specifying and canceling the initial step (Page 7440)

Renumbering steps and transitions (Page 7441)

Renaming steps and transitions (Page 7443)

Adding comments for steps (Page 7444)

Skip steps (Page 7445)

Deleting steps and transitions (Page 7446)

17.5.7.7 Skip steps (S7-300, S7-400, S7-1500)

You have the option of preventing the execution of steps for which the preceding and subsequent transitions are simultaneously valid. For CPUs of the S7-300/400 series, however, this option cannot be used in combination with the "Create minimized DB" option.

Procedure

To skip steps for which the preceding and subsequent transitions are simultaneously valid, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Select the "Skip steps" option under "Sequence properties".

See also

Steps and transitions (Page 7375)
Insert steps and transitions (Page 7438)
Specifying and canceling the initial step (Page 7440)
Renumbering steps and transitions (Page 7441)
Renaming steps and transitions (Page 7443)
Adding comments for steps (Page 7444)
Copying and inserting steps and transitions (Page 7444)
Deleting steps and transitions (Page 7446)

17.5.7.8 Deleting steps and transitions (S7-300, S7-400, S7-1500)

You can remove steps and transitions from a sequencer again. Note the following particularities:

- When steps and transitions are deleted, the alternating sequence of a step and a transition must also be retained. For this reason, placeholders are inserted ("Missing Step", "Missing Transition"), if two steps or two transitions in a row occur after the deletion operation.
- You cannot delete all the steps of a sequencer since it must always contain at least one step.

Requirement

A GRAPH function block is open in the sequence view.

Procedure

To delete a step or a transition, follow these steps:

1. To delete a step, right-click the step you want to delete. To delete a transition, right-click the transition you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. If necessary, replace the placeholders with valid elements.

See also

Steps and transitions (Page 7375)
Insert steps and transitions (Page 7438)
Specifying and canceling the initial step (Page 7440)
Renumbering steps and transitions (Page 7441)
Renaming steps and transitions (Page 7443)
Adding comments for steps (Page 7444)

Copying and inserting steps and transitions (Page 7444)

Skip steps (Page 7445)

17.5.7.9 Displaying and editing translations of step and transition names (S7-1500)

You can display the names of all steps and transitions in all project languages that have been activated in the setting options for the project languages and for which translations already exist. You can find additional information on activating project languages under AUTOHOTSPOT.

If you change the name of the step or the transition directly in the program editor and have not made any changes to the names on the "Texts" tab, then the name is automatically adjusted in all project languages. This is advantageous, for example, if you use language-neutral step and transition names.

Requirement

You have created at least one step and one transition in a GRAPH FB of the version 5.0 and the GRAPH FB is open.

Procedure

To display the name of a step in all project languages, follow these steps:

1. Select the line with the required step or the transition whose translations you want to display.
   If you want to display the translations of multiple steps or transitions simultaneously, you need to select the corresponding steps and transitions.
   If you want to display the translations of all steps and transitions simultaneously, click in the GRAPH FB outside the sequencer.
2. Open the "Properties" tab in the Inspector window.
3. Open the "Texts" tab.
   If the names of the steps and transitions appear only in the current user interface language, either you have not yet selected any project languages or translations are not yet available. You can enter the translations directly in the table on the "Texts" tab or you can enter the translations for the individual project languages in the global "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts".

You can find additional information on translation of texts under AUTOHOTSPOT.

Result

The names of the selected steps and transitions are displayed in the required project languages and, if available, the translations of the names on the lower-level "Texts" tab.

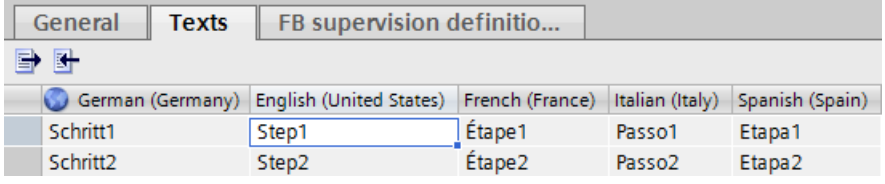

German (Germany)	English (United States)	French (France)	Italian (Italy)	Spanish (Spain)
Schritt1	Step1	Étape1	Passo1	Etapa1
Schritt2	Step2	Étape2	Passo2	Etapa2

17.5.8 Using branches (S7-300, S7-400, S7-1500)

17.5.8.1 Creating alternative branches (S7-300, S7-400, S7-1500)

Requirement

A sequencer is open in the sequence view.

Procedure

To create an alternative branch, follow these steps:

1. Right-click the step after which you want to insert an alternative branch.
2. Select the "Open alternative branch" command in the "Insert element" submenu of the shortcut menu.

Or:

1. Open the "Basic instructions" pane of the "Instructions" task card.
2. Open the "Graph structure" folder.
3. Drag the "Alternative branch" element to the desired position in the sequencer. Allowed positions are indicated by a green square.

See also

Elements of a sequencer (Page 7372)

Creating a simultaneous branch (Page 7448)

Closing branches (Page 7449)

Deleting branches (Page 7450)

17.5.8.2 Creating a simultaneous branch (S7-300, S7-400, S7-1500)

Requirements

A sequencer is open in the sequence view.

Procedure

To create a simultaneous branch, follow these steps:

1. Right-click the transition after which you want to insert a simultaneous branch.
2. Select the "Open simultaneous branch" command in the "Insert element" submenu of the shortcut menu.

Or:

1. Open the "Basic instructions" pane of the "Instructions" task card.
2. Open the "Graph structure" folder.
3. Drag the "Simultaneous branch" element to the desired position in the sequencer. Allowed positions are indicated by a green square.

See also

Elements of a sequencer (Page 7372)

Creating alternative branches (Page 7448)

Closing branches (Page 7449)

Deleting branches (Page 7450)

17.5.8.3 Closing branches (S7-300, S7-400, S7-1500)

You can close simultaneous and alternative branches as follows:

- You can join the branch to its parent branch.
- You can close the branch with a sequence end.
- You can close the branch with a jump.

In a simultaneous branch, however, you can only insert a sequence end or a jump after a transition.

Requirement

An alternative or simultaneous branch is present.

Joining a branch to its parent branch

To join an open branch to its parent branch, follow these steps:

1. Select the double arrow at the end of the branch.
2. Press and hold down the left mouse button.
   A dashed line will appear as soon as the cursor is moved.
3. Drag the dashed line to a suitable position in the network. Allowed positions are indicated by a green square.
4. Release the left mouse button.

Closing a branch with a sequence end

To close a branch with a sequence end, follow these steps:

1. If required, insert a transition at the end of the branch in a simultaneous branch.
2. Right-click the double arrow at the end of the branch.
3. Select the "Sequence end" element in the "Insert element" submenu of the shortcut menu.

Closing a branch with a jump

To close a branch with a jump, follow these steps:

1. If required, insert a transition at the end of the branch in a simultaneous branch.
2. Right-click the double arrow at the end of the branch.
3. Select the "Jump" element in the "Insert element" submenu of the shortcut menu.
   An overview containing all available steps of the GRAPH function block opens.
4. Select a step as the jump destination.

See also

Elements of a sequencer (Page 7372)
Creating alternative branches (Page 7448)
Creating a simultaneous branch (Page 7448)
Deleting branches (Page 7450)

17.5.8.4 Deleting branches (S7-300, S7-400, S7-1500)

Requirement

An alternative or simultaneous branch is present.

Procedure

To delete a branch, follow these steps:

1. Select the branch you want to delete.
2. Right-click the selected branch.
3. Select the "Delete" command in the shortcut menu.

See also

Elements of a sequencer (Page 7372)
Creating alternative branches (Page 7448)

Creating a simultaneous branch (Page 7448)

Closing branches (Page 7449)

17.5.9 Using jumps (S7-300, S7-400, S7-1500)

17.5.9.1 Inserting jumps (S7-300, S7-400, S7-1500)

You can insert jumps only at the end of a sequencer or branch. In a simultaneous branch, the last element must be a transition for this.

Requirement

A sequencer is open in the sequence view.

Procedure

To insert a jump in a sequencer or branch, follow these steps:

1. If required, insert a transition at the end of the branch in a simultaneous branch.
2. Right-click the double arrow at the end of the sequencer or branch.
3. Select the "Jump" element in the "Insert element" submenu of the shortcut menu.
   An overview containing all available steps of the GRAPH function block opens.
4. Select a step as the jump destination.

Result

The jump is inserted. The jump and jump destination are represented in the sequencer as arrows, in which case the return transition is indicated for the jump destination and the destination step is indicated for the return.

See also

Elements of a sequencer (Page 7372)

Changing the jump destination (Page 7451)

Navigating to the jump destination (Page 7452)

Deleting jumps (Page 7453)

17.5.9.2 Changing the jump destination (S7-300, S7-400, S7-1500)

You can change the destination of a jump subsequently.

Requirement

- A GRAPH function block is open in the sequence view.
- A jump is available.

Procedure

To change the jump destination subsequently, follow these steps:

1. Double-click the step number of the current jump destination.
   An overview containing all available steps of the GRAPH function block opens.
2. Select a step as the new jump destination.

See also

Elements of a sequencer (Page 7372)
Inserting jumps (Page 7451)
Navigating to the jump destination (Page 7452)
Deleting jumps (Page 7453)

17.5.9.3 Navigating to the jump destination (S7-300, S7-400, S7-1500)

You can navigate from a jump to the step that you have specified as the jump destination.

Requirement

- A GRAPH function block is open in the sequence view.
- A jump is available.

Procedure

To navigate from a jump to the step specified as the destination, follow these steps:

1. Right-click on the jump.
2. Select the "Go to > Jump destination" command in the shortcut menu.

See also

Elements of a sequencer (Page 7372)
Inserting jumps (Page 7451)
Changing the jump destination (Page 7451)
Deleting jumps (Page 7453)

17.5.9.4 Deleting jumps (S7-300, S7-400, S7-1500)

Requirement

- A GRAPH function block is open in the sequence view.
- A jump is available.

Procedure

To delete a jump, follow these steps:

1. Right-click on the jump you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Elements of a sequencer (Page 7372)
Inserting jumps (Page 7451)
Changing the jump destination (Page 7451)
Navigating to the jump destination (Page 7452)

17.6 Programming actions (S7-300, S7-400, S7-1500)

17.6.1 Inserting actions (S7-300, S7-400, S7-1500)

Introduction

You can specify actions that will be implemented when the step is processed. In order to add comments for actions or adapt block calls, you can enable a multi-line input. Single-line input is the default setting. You can insert actions either in the single step view or the sequence view.

For a better overview of your programmed actions you can show descriptions for the events and identifiers.

In actions you can assign an operand with the value of another operand or mathematical expression or the result of an instruction. Use the following syntax for this:

- Assigning the value of an operand: A:=B
- Assigning the value of a mathematical expression: A:=B<Operator>C
  You can use the following operators: + (Addition), - (Subtraction), * (Multiplication), / (Division), % (Modulo), AND (AND logic operation), OR (OR logic operation), XOR (Exclusive OR logic operation)
- Assigning the result of an instruction: A:=<Instruction>(B)
  You obtain the available instructions from the "Instructions" task card.

Note that the data types of the operands must be compatible with one another.

Inserting actions in the single step view

To insert an action in the single step view, follow these steps:

1. Open the single step view.
2. If the actions are collapsed, click the small arrow in front of "Actions".
   The Right arrow becomes a Down arrow, and the table containing the actions is displayed.
3. If you want to enable multi-line input, right-click in the line of the new action and select the "Allow multi-line mode" command in the shortcut menu.
4. If you want to link the new action with an interlock, click in the cell of the "Interlock" column and select the "-(C)- - Interlock" entry in the drop-down list.
5. If you want to link the new action with an event, click in the cell of the "Event" column and select the appropriate event in the drop-down list.

6. Click in the cell of the "Identifier" column and select the identifier of the new action in the drop-down list.
7. In the "Action" column, specify the action that is to be performed. You have the following options:
   – You can replace the placeholders with the operands or values that you want to use for the action. You can also insert these using drag-and-drop operation or via autocompletion.
   – You can use instructions from the "Instructions" task card. To do so, drag-and-drop them from the task card into the "Action" column.
   – You can drag-and-drop blocks from the project tree into the "Action" column in order to call them.

Note

When data-type-dependent instructions are called, you must specify a data type. You can either manually enter the data type or select it from a drop-down list by clicking the small green rectangle that is shown below the existing data type.

Inserting actions in the sequence view

To insert an action in the sequence view, follow these steps:

1. Open the sequence view.
2. Expand the step in which you want to insert actions. To do so, click the button at the top right corner of the step or select the step and press <Enter>.
3. If you want to enable multi-line input, right-click in the line of the new action and select the "Allow multi-line mode" command in the shortcut menu.
4. If you want to link the new action with an interlock, click in the cell of the "Interlock" column and select the "-(C)- - Interlock" entry in the drop-down list.
5. If you want to link the new action with an event, click in the cell of the "Event" column and select the appropriate event in the drop-down list.

6. Click in the cell of the "Identifier" column and select the identifier of the new action in the drop-down list.
7. In the "Action" column, specify the action that is to be performed. You have the following options:
   - You can replace the placeholders with the operands or values that you want to use for the action. You can also insert these using drag-and-drop operation or via autocompletion.
   - You can use instructions from the "Instructions" task card. To do so, drag-and-drop them from the task card into the "Action" column.
   - You can drag-and-drop blocks from the project tree into the "Action" column in order to call them.

Note

When data-type-dependent instructions are called, you must specify a data type. You can either manually enter the data type or select it from a drop-down list by clicking the small green rectangle that is shown below the existing data type.

Showing the event descriptions

To show the description of the events, follow these steps:

1. Right-click in the action table.
2. Select the "Show event descriptions" command in the shortcut menu.

Showing the identifier descriptions

To show the description of the identifiers, follow these steps:

1. Right-click in the action table.
2. Select the "Show qualifier descriptions" command in the shortcut menu.

See also

Changing actions (Page 7459)

Copying, cutting, and inserting actions (Page 7460)

Moving actions (Page 7461)

Deleting actions (Page 7462)

Actions (Page 7378)

17.6.2 Defining the data type of a GRAPH instruction (S7-300, S7-400, S7-1500)

17.6.2.1 Basic information on the data types of GRAPH instructions (S7-300, S7-400, S7-1500)

Introduction

GRAPH instructions can be used by means of the following two mechanisms:

- As block call
  For instructions that you insert in your program as block calls, you can select the data type via a drop-down list.
- As GRAPH-internal function
  You assign the GRAPH-internal functions directly to a tag. The operand must have a data type which supports this function. The following table provides an overview of the permitted data types of GRAPH-internal functions.

GRAPH-internal functions

The following table shows the GRAPH-internal functions and their data types:

Function	S7-300/400	S7-1500
BCD_TO_INT	WORD	WORD
BCD_TO_DINT	DWORD	DWORD
INT_TO_BCD	INT	INT
DINT_TO_BCD	DINT	DINT
INT_TO_DINT	INT	INT
DINT_TO_REAL	DINT	DINT
ROUND	REAL	REAL, LREAL
TRUNC	REAL	REAL, LREAL
CEIL	REAL	REAL, LREAL
FLOOR	REAL	REAL, LREAL
NEG	INT, DINT, REAL	SINT, INT, DINT, REAL, LINT, LREAL
ABS	REAL	SINT, INT, DINT, LINT, REAL, LREAL
SQR	REAL	REAL, LREAL
SQRT	REAL	REAL, LREAL
LN	REAL	REAL, LREAL
EXP	REAL	REAL, LREAL
SIN	REAL	REAL, LREAL
COS	REAL	REAL, LREAL
TAN	REAL	REAL, LREAL
ASIN	REAL	REAL, LREAL
ACOS	REAL	REAL, LREAL
ATAN	REAL	REAL, LREAL

Function	S7-300/400	S7-1500
NOT	WORD, DWORD	USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, BYTE, WORD, DWORD, LWORD
RLDA	DWORD	-
RRDA	DWORD	-
SWAP	WORD, DWORD	WORD, DWORD, LWORD
AND	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD
OR	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD
XOR	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD
SHL	WORD, DWORD	BYTE, USINT, SINT, WORD, UINT, INT, DWORD, UDINT, DINT, LWORD, ULINT, LINT
SHR	WORD, DWORD	BYTE, WORD, DWORD, USINT, UINT, UDINT, SINT, INT, DINT, LWORD, ULINT, LINT
ROL	DWORD	BYTE, WORD, DWORD, LWORD
ROR	DWORD	BYTE, WORD, DWORD, LWORD
MOD	DINT	USINT, UINT, UDINT, SINT, INT, DINT, ULINT, LINT
-: Function not available		

17.6.2.2 Changing the data type of a GRAPH instruction (S7-300, S7-400, S7-1500)

GRAPH instructions that you insert in your program via a block call are inserted with a preset data type. If necessary, you can change this data type. Data types which can be changed are indicated by a green rectangle under the first letter of the data type. You can change the operand for GRAPH-internal functions.

Changing the data type of a called instruction

To change the data type of a called instruction, follow these steps:

1. Click on the data type of the instruction.
   The drop-down list for data type selection opens.
2. Click on the data type that you wish to use.

Changing the data type of a GRAPH-internal instruction

To change the data type of a GRAPH-internal function, follow these steps:

1. Enter a value or a tag as the operand. Observe the permitted range of values for the operand.

Or:

1. Holding the right mouse button down, drag a tag with the permitted data type over the operand.
2. Wait until the current operand has a green border.
3. Release the mouse button.

17.6.3 Changing actions (S7-300, S7-400, S7-1500)

You can make the following changes for actions:

- Linking an action with an interlock or canceling the link with an interlock
- Linking an action with an event or canceling or changing the link with an event.
- Changing the action identifier
- Changing the action, e.g., replacing operands or using a different instruction

Note that certain action identifiers require the use of events and that the link to an event cannot be canceled for these identifiers.

Requirement

A GRAPH function block is open in the single step view.

Procedure

To change an action, follow these steps:

1. If you want to link an action with an interlock, click in the cell of the "Interlock" column and select the "-(C)- - Interlock" entry in the drop-down list.
2. If you want to cancel the link with an interlock, click in the cell of the "Interlock" column and select the "No condition" entry in the drop-down list.
3. If you want to link the action with an event or change the linked event, click in the cell of the "Event" column and select the appropriate event in the drop-down list.
4. If you want to cancel the link with an event, click in the cell of the "Event" column and select the "No event" entry in the drop-down list. If the action identifier requires an event, the background of the cell turns red and "<???>" is displayed. In this case, you must assign an event again or choose a different action identifier.

5. If you want to change the action identifier, click in the cell of the "Qualifier" column and select the new action identifier in the drop-down list.
6. If you want to change the action, click in the action in the "Action" column and make one of the following changes:
   - Change the operand(s) of the action.
   - To change the data type of an operand, click the small green rectangle and select the new data type in the drop-down list.
   - To use a different instruction, delete the existing action and drag a different instruction from the "Instructions" task card into the "Action" column. Enter the operands.
   - To call a different block, delete the existing action and drag-and-drop a block from the project tree into the "Action" column in order to call it.

See also

Inserting actions (Page 7454)

Copying, cutting, and inserting actions (Page 7460)

Moving actions (Page 7461)

Deleting actions (Page 7462)

Actions (Page 7378)

17.6.4 Copying, cutting, and inserting actions (S7-300, S7-400, S7-1500)

Requirement

A GRAPH function block is open in the single step view.

Copying actions

To copy an action, follow these steps:

1. Right-click in the first cell of the action you want to copy.
2. Select the "Copy" command in the shortcut menu.

Cutting actions

To cut an action, follow these steps:

1. Right-click in the first cell of the action you want to cut.
2. Select the "Cut" command in the shortcut menu.

Pasting actions from the clipboard

To insert an action from the clipboard, follow these steps:

1. Right-click the position at which you want to insert the action.
2. Select the "Insert" command in the shortcut menu.

See also

Inserting actions (Page 7454)
Changing actions (Page 7459)
Moving actions (Page 7461)
Deleting actions (Page 7462)
Actions (Page 7378)

17.6.5 Moving actions (S7-300, S7-400, S7-1500)

You can move actions within the action table.

Requirement

A GRAPH function block is open in the single step view.

Procedure

To move an action within the action table, follow these steps:

1. Click in the first cell of the action you want to move and keep the mouse button pressed. This selects the action.
2. Drag the action to the desired position.

See also

Inserting actions (Page 7454)
Changing actions (Page 7459)
Copying, cutting, and inserting actions (Page 7460)
Deleting actions (Page 7462)
Actions (Page 7378)

17.6.6 Deleting actions (S7-300, S7-400, S7-1500)

Requirement

A GRAPH function block is open in the single step view.

Procedure

To delete an action, follow these steps:

1. Right-click in the first cell of the action you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting actions (Page 7454)
Changing actions (Page 7459)
Copying, cutting, and inserting actions (Page 7460)
Moving actions (Page 7461)
Actions (Page 7378)

17.7 Programming conditions (S7-300, S7-400, S7-1500)

17.7.1 Programming interlocks (S7-300, S7-400, S7-1500)

You can use up to 32 instructions in your interlock network to which a maximum of 32 operands are interconnected. In LAD, the instructions can be inserted both in a rung and in the power rail.

Requirement

A GRAPH function block is open in the single step view.

Procedure

To program interlocks, follow these steps:

1. Open the interlock network by clicking the small arrow in front of "Interlock -(c)-".
2. Open the "Instructions" task card.
3. Navigate to the instruction you want to insert.
4. Drag the instruction to the desired position in the network.
5. In the "Interlock" column of the action table, click in the cell of the action you want to link with the interlock and select the "-(C)- - Interlock" entry in the drop-down list.

Or:

1. Open the interlock network by clicking the small arrow in front of "Interlock -(c)-".
2. Open the "Instructions" task card.
3. Navigate to "LAD/FBD > General > Empty box" in the "Basic instructions" pane.
4. Drag the "Empty box" element to the desired position in the network.
5. Click on "??".
   A text box opens.
6. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the permitted elements for the particular context.
7. Select the desired element from the drop-down list.
8. In the "Interlock" column of the action table, click in the cell of the action you want to link with the interlock and select the "-(C)- - Interlock" entry in the drop-down list.

See also

Conditions (Page 7389)

Inserting supervision conditions (Page 7464)

Programming transitions (Page 7469)

Programming permanent instructions (Page 7471)

Displaying or hiding tag information (Page 7478)

Specifying the programming language in networks (Page 7416)

17.7.2 Programming supervisions (S7-300, S7-400, S7-1500)

17.7.2.1 Inserting supervision conditions (S7-300, S7-400, S7-1500)

You can use up to 32 instructions in your supervision network to which a maximum of 32 operands are interconnected. In LAD, the instructions can be inserted both in a rung and in the power rail.

Note

You can program step time monitoring for CPUs of the S7-1500 series. To do this, you can use the two instructions "Greater than maximum step activation time" (CMP>T_MAX), "Greater than warning time" (CMP>T_WARN) and a warning coil (-(w)-). You can either specify the comparison values for the two instructions manually or have them determined by the system in learning mode.

Requirement

A GRAPH function block is open in the single step view.

Procedure

To program supervisions, follow these steps:

1. Open the supervision network by clicking the small arrow in front of "Supervision -(v)-".
2. Open the "Instructions" task card.
3. Navigate to the instruction you want to insert.
4. Drag the instruction to the desired position in the network.

Or:

1. Open the supervision network by clicking the small arrow in front of "Supervision -(v)-".
2. Open the "Instructions" task card.
3. Navigate to "LAD/FBD > General > Empty box" in the "Basic instructions" pane.
4. Drag the "Empty box" element to the desired position in the network.
5. Click on "??".
   A text box opens.
6. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the permitted elements for the particular context.
7. Select the desired element from the drop-down list.

See also

Conditions (Page 7389)
Programming interlocks (Page 7463)
Programming transitions (Page 7469)
Programming permanent instructions (Page 7471)
Displaying or hiding tag information (Page 7478)
Specifying the programming language in networks (Page 7416)

17.7.2.2 Using step time monitoring (S7-1500)

Basics of step time monitoring (S7-1500)

Introduction

With supervision conditions, you can monitor the step activation time; in other words, the time that has elapsed since activation of the step. To do this, you can specify the maximum time "T_MAX" and use this as a comparison value. To make the comparison, you can use the instruction "Greater than maximum step activation time" (CMP>T_MAX). If the maximum step activation time is exceeded, a supervision error is triggered and a supervision alarm is output, if you have enabled this. To receive an alarm before the supervision error occurs, you can specify a warning time "T_WARN" and monitor this, once again using a supervision condition. To do this, you can use the instruction "Greater than warning time" (CMP>T_WARN) and a warning coil.

Initially, the values specified in the settings in "PLC programming > GRAPH > Default for time monitoring" are used for "T_MAX" and "T_WARN". You have the following options for changing the times for "T_MAX" and "T_WARN":

- Manually using the "Define time" dialog (Page 7467)
- Using the learning mode of the interface (Page 7467)
- Using the learning mode when testing (Page 7706)

If you use the learning mode, the activation time of the steps is measured as the program runs. The longer you leave the learning mode activated, the more accurately the activation time can be determined. The times for "T_MAX" and "T_WARN" result from the sums of the measured times and the additive thresholds that you can set.

If you use the learning mode to determine the times during testing, you can, if necessary, reset the learned times. This returns "T_MAX" and "T_WARN" to their default values in the instance data block.

Note

For GRAPH blocks with a version lower than 2.0, the learning mode for determining the values of "T_MAX" and "T_WARN" is not available. You can change the version in the properties of the GRAPH block under "General > Block".

See also

Programming step time monitoring (Page 7466)

CMP>T_MAX: Greater than maximum step activation time (Page 2075)

CMP>T_WARN: Greater than warning time (Page 2076)

CMP>T_MAX: Greater than maximum step activation time (Page 2086)

CMP>T_WARN: Greater than warning time (Page 2087)

Programming step time monitoring (S7-1500)

Requirement

- The alarm view is enabled for the GRAPH function block.
  See also: Enabling and disabling the alarm view (Page 7485)
- Alarm texts are defined for the GRAPH function block.
  See also: Editing alarms (Page 7485)
- The GRAPH function block is open in the single step view.

Procedure

To program step time monitoring including a supervision and a warning, follow these steps:

1. Open the supervision network by clicking the small arrow in front of "Supervision -(V)-".
2. Open the "Instructions" task card.
3. Insert the instruction "CMP>T_MAX" in the rung with the supervision coil "-(V)-".
4. Insert the instruction "CMP>T_WARN" in a new rung.
5. Insert the instruction "-(W)-" in the new rung.
6. Specify the times for "T_MAX" and "T_WARN" either manually or using the learning mode.

Result

If the step activation time is greater than the time specified for "T_WARN", a warning is displayed. If it also exceeds the time of "T_MAX", the supervision alarm is output.

See also

Basics of step time monitoring (Page 7465)

Specifying times T_MAX and T_WARN manually (Page 7467)

Determining times for T_MAX and T_WARN using the learning mode of the interface (Page 7467)

Calculating times for T_MAX and T_WARN when testing (Page 7706)

CMP>T_MAX: Greater than maximum step activation time (Page 2075)

CMP>T_WARN: Greater than warning time (Page 2076)

CMP>T_MAX: Greater than maximum step activation time (Page 2086)

CMP>T_WARN: Greater than warning time (Page 2087)

Specifying times T_MAX and T_WARN manually

Requirement

- The GRAPH function block is open in the single step view.
- Step time monitoring is programmed for the step.

Procedure

To specify the times for "T_MAX" and "T_WARN", follow these steps:

1. Right-click in the supervision network.
2. Select the "Define time" command in the shortcut menu.
   The "Define time" dialog opens.
3. Enter the times for "T_MAX" and "T_WARN".
4. Confirm your entries with "OK".
   The entered times are adopted in the block interface as default values for the static step parameters "T_MAX" and "T_WARN".

See also

Basics of step time monitoring (Page 7465)

Programming step time monitoring (Page 7466)

Determining times for T_MAX and T_WARN using the learning mode of the interface (Page 7467)

CMP>T_MAX: Greater than maximum step activation time (Page 2075)

CMP>T_WARN: Greater than warning time (Page 2076)

CMP>T_MAX: Greater than maximum step activation time (Page 2086)

CMP>T_WARN: Greater than warning time (Page 2087)

Determining times for T_MAX and T_WARN using the learning mode of the interface (S7-1500)

You can have the times for "T_MAX" and "T_WARN" determined directly using the block call of the GRAPH function block. However, note the following restrictions as compared to the use of the learning mode when testing that you can start in the "Testing" task card:

- The determined values are only saved in the online instance data block.
- You cannot reset the learned times.

To make sure that "T_MAX" and "T_WARN" are greater than the learned activation times, you can define additive thresholds that are added to the learned times. Enter a lower value for the warning than for the supervision so that the warning is displayed before the supervision.

Requirement

The GRAPH function block is called in the program.

Defining additive thresholds

To define additive thresholds, follow these steps:

1. Insert the "MOVE" instruction in the block that calls the GRAPH function block.
2. For the "IN" parameter, enter the additive threshold for the supervision as a percentage.
3. For the "OUT1" parameter, enter the static parameter "RT_DATA > TRESHOLD_SUP" of the instance data block.
4. Insert another "MOVE" instruction.
5. For the "IN" parameter, enter the additive threshold for the warning as a percentage.
6. For the "OUT1" parameter, enter the static parameter "RT_DATA > TRESHOLD_WARN" of the instance data block.
   The additive thresholds are taken into account in the calculation of "T_MAX" and "T_WARN" in learning mode.

Enabling learning mode

To activate learning mode via the interface, follow these steps:

1. Set the interface parameter set to "Maximum interface parameters" or insert the two parameters manually in the block interface of the GRAPH function block:
   - "EN_LMODE" input parameter
   - "LMODE_ENABLED" output parameter
2. Open the calling block.
3. Link the "EN_LMODE" input parameter to a tag.
4. Download the block to the device.
5. In the toolbar, click "Monitoring on/off" to start testing with program status for the calling block.
6. Right click on the tag you linked to the "EN_LMODE" input parameter.
7. In the shortcut menu, select the command "Modify > Modify to 1" to obtain a rising edge for the "EN_LMODE" input parameter.
   The signal state of "LMODE_ENABLED" is set to "1" and the learning mode is activated.
8. Wait until each step in the GRAPH function block has been active at least once.
9. Right click again on the tag you linked to the "EN_LMODE" input parameter.
10. In the shortcut menu, select the command "Modify > Modify to 0" to reset the "EN_LMODE" input parameter.

11. Right click again on the tag you linked to the "EN_LMODE" input parameter.
12. In the shortcut menu, select the command "Modify > Modify to 1" to obtain a rising edge for the "EN_LMODE" input parameter.
    Learning mode is deactivated. The learned times are entered in the online instance data block of the GRAPH function block.

See also

Basics of step time monitoring (Page 7465)

Programming step time monitoring (Page 7466)

Specifying times T_MAX and T_WARN manually (Page 7467)

CMP>T_MAX: Greater than maximum step activation time (Page 2075)

CMP>T_WARN: Greater than warning time (Page 2076)

CMP>T_MAX: Greater than maximum step activation time (Page 2086)

CMP>T_WARN: Greater than warning time (Page 2087)

17.7.3 Programming transitions (S7-300, S7-400, S7-1500)

You can use up to 32 instructions in your transition network to which a maximum of 32 operands are interconnected. In LAD, the instructions can be inserted both in a rung and in the power rail. You can program transitions in either the single step view or the sequence view.

Programming transitions in the single step view

To program transitions in the single step view, follow these steps:

1. Open the single step view.
2. Open the transition network by clicking the small arrow in front of "<TransitionNumber> - <TransitionName>".
3. Open the "Instructions" task card.
4. Navigate to the instruction you want to insert.
5. Drag the instruction to the desired position in the network.

Or:

1. Open the transition network by clicking the small arrow in front of "<TransitionNumber> - <TransitionName>".
2. Open the "Instructions" task card.
3. Navigate to "LAD/FBD > General > Empty box" in the "Basic instructions" pane.
4. Drag the "Empty box" element to the desired position in the network.
5. Click on "??".
   A text box opens.

6. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the permitted elements for the particular context.
7. Select the desired element from the drop-down list.

Programming transitions in the sequence view

To program transitions in the sequence view, follow these steps:

1. Open the sequence view.
2. Expand the transition you want to program. To do so, click the button next to the transition or select the transition and press <Enter>.
3. Open the "Instructions" task card.
4. Navigate to the instruction you want to insert.
5. Drag the instruction to the desired position in the network.

Or:

1. Open the "Instructions" task card.
2. Navigate to "LAD/FBD > General > Empty box" in the "Basic instructions" pane.
3. Drag the "Empty box" element to the desired position in the network.
4. Click on "??".
   A text box opens.
5. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the permitted elements for the particular context.
6. Select the desired element from the drop-down list.

See also

Conditions (Page 7389)

Programming interlocks (Page 7463)

Inserting supervision conditions (Page 7464)

Programming permanent instructions (Page 7471)

Displaying or hiding tag information (Page 7478)

Specifying the programming language in networks (Page 7416)

17.7.4 Programming permanent instructions (S7-300, S7-400, S7-1500)

17.7.4.1 Working with networks (S7-300, S7-400, S7-1500)

Using networks (S7-300, S7-400, S7-1500)

Function

Permanent instructions are programmed inside networks in a GRAPH function block. At least one network must be available in order to program a permanent instruction. To achieve a better overview of the program, you can also subdivide your program into several networks.

See also

Inserting networks (Page 7471)
Selecting networks (Page 7472)
Copying and pasting networks (Page 7473)
Deleting networks (Page 7473)
Expanding and collapsing networks (Page 7474)
Inserting network title (Page 7475)
Navigating networks (Page 7476)

Inserting networks (S7-300, S7-400, S7-1500)

Requirement

A block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Result

A new empty network is inserted into the block.

See also

Using networks (Page 7471)
Selecting networks (Page 7472)
Copying and pasting networks (Page 7473)
Deleting networks (Page 7473)
Expanding and collapsing networks (Page 7474)
Inserting network title (Page 7475)
Navigating networks (Page 7476)

Selecting networks (S7-300, S7-400, S7-1500)

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.
   The first and last networks and all those in between are selected.

See also

Using networks (Page 7471)
Inserting networks (Page 7471)
Copying and pasting networks (Page 7473)
Deleting networks (Page 7473)
Expanding and collapsing networks (Page 7474)

Inserting network title (Page 7475)
Navigating networks (Page 7476)

Copying and pasting networks (S7-300, S7-400, S7-1500)

Copied networks can be pasted within the block or in another block. Networks that were created in LAD or FBD can also be inserted in blocks of the respective other programming language.

Requirement

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Using networks (Page 7471)
Inserting networks (Page 7471)
Selecting networks (Page 7472)
Deleting networks (Page 7473)
Expanding and collapsing networks (Page 7474)
Inserting network title (Page 7475)
Navigating networks (Page 7476)

Deleting networks (S7-300, S7-400, S7-1500)

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Using networks (Page 7471)
Inserting networks (Page 7471)
Selecting networks (Page 7472)
Copying and pasting networks (Page 7473)
Expanding and collapsing networks (Page 7474)
Inserting network title (Page 7475)
Navigating networks (Page 7476)

Expanding and collapsing networks (S7-300, S7-400, S7-1500)

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Using networks (Page 7471)
Inserting networks (Page 7471)
Selecting networks (Page 7472)
Copying and pasting networks (Page 7473)
Deleting networks (Page 7473)
Inserting network title (Page 7475)
Navigating networks (Page 7476)

Inserting network title (S7-300, S7-400, S7-1500)

The network title is the header of a network. The length of the network title is limited to one line. You can enter the title manually or set it automatically. When you set it automatically, you can do this for individual networks or use the settings to specify that the network title is always set automatically.

For automatic insertion of the network title, the comment of the operand in one of the following instructions in the network is evaluated:

- Assignment
- Set output
- Reset output

The instruction that is listed first in the network is used.

The network title is only inserted automatically when the following conditions are fulfilled:

- The network does not have a title yet.
- The operand of the instruction used for the comment has a comment.

Note

Note the following restrictions for automatic insertion of the network title:

- The network title is not adapted if you change the comment of the operand at a later time.
- The network title is not adapted if you change the operand of the instruction.
- The network title is only set by the writing instructions listed above.
- If the operand is of the data type array, the comment of the array is used and not the comments of the array elements.
- Comments of invalid operands are not taken into consideration.

Entering the network title manually

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

Setting the network title automatically

To specify that the network titles are always set automatically, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Select the "Set network title automatically" check box in the "Additional settings" group.
   The network titles are set automatically as of this time if the conditions listed above are fulfilled.

To set an individual network title automatically, follow these steps:

1. Right-click "Network <Number of the network>" in the title bar of a network.
2. Select the "Set network title automatically" command in the shortcut menu.
   The title of the selected network is set based on the comment of the operand if the conditions listed above are fulfilled.

See also

Using networks (Page 7471)

Inserting networks (Page 7471)

Selecting networks (Page 7472)

Copying and pasting networks (Page 7473)

Deleting networks (Page 7473)

Expanding and collapsing networks (Page 7474)

Navigating networks (Page 7476)

Navigating networks (S7-300, S7-400, S7-1500)

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click in the code area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.
   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 7471)

Inserting networks (Page 7471)

Selecting networks (Page 7472)

Copying and pasting networks (Page 7473)

Deleting networks (Page 7473)

Expanding and collapsing networks (Page 7474)

Inserting network title (Page 7475)

17.7.4.2 Inserting permanent instructions (S7-300, S7-400, S7-1500)

When you expand a pane for permanent pre-instructions or post-instructions, it always contains at least one empty network by default. You can use up to 32 instructions per network to which a maximum of 32 operands are interconnected. In LAD, the instructions can be inserted both in a rung and in the power rail. You can use up to 250 permanent pre-instructions and 250 permanent post-instructions within a GRAPH function block.

Requirement

A GRAPH function block is open in the sequence view.

Procedure

To program permanent instructions, follow these steps:

1. In the navigation view, open the "Permanent pre-instructions" or "Permanent post-instructions" pane.
2. Click on a network.
   The view for permanent pre-instructions or post-instructions opens in the work area.
3. Open the "Instructions" task card.
4. Navigate to the instruction you want to insert.
5. Drag the instruction to the desired position in the network.

Or:

1. Open the "Instructions" task card.
2. Navigate to "General > Empty box" in the "Basic instructions" pane.
3. Drag the "Empty box" element to the desired position in the network.
4. Click on "??".
   A text box opens.
5. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the permitted elements for the particular context.
6. Select the desired element from the drop-down list.

See also

Conditions (Page 7389)

Programming interlocks (Page 7463)

Inserting supervision conditions (Page 7464)

Programming transitions (Page 7469)

Displaying or hiding tag information (Page 7478)

Specifying the programming language in networks (Page 7416)

17.7.5 Displaying or hiding tag information (S7-300, S7-400, S7-1500)

Introduction

For permanent instructions, interlocks, supervisions, and transitions, you can display the following information on the utilized tags in the programming window:

- Name of the tag
- Tag addresses
- Simple or hierarchical comments to document the tags

The information is taken from the block interface for local tags and DB tags and from the PLC tag table for tags that are valid CPU-wide.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all the blocks currently opened and opened in future is shown. You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual blocks that are open.

When you select the display of tag information with hierarchical comments, the comments of the higher structure levels are also displayed for structured tags. The comments are shown in brackets after the tag comment; the comments of the individual levels are separated by a dot. If there is no comment for a tag on a structure level, it is not displayed. This is indicated by two successive dots.

You can display the tag information either collectively below a LAD/FBD network or directly at the respective operand. As with showing and hiding tag information, you can set this via the settings for all block or change specifically for an opened block.

Displaying or hiding tag information for all the blocks

Proceed as follows to display or hide the tag information for all the blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to display the tag information, select the "Show" option or the "Tag information with hierarchical comments" option in the drop-down list, depending on whether you want to display simple or hierarchical comments.

4. Use the "Position of the tag information (LAD/FBD)" drop-down list to specify whether the tag information is to be displayed below the LAD/FBD network or on the operand.
   For all open blocks, the tag information is shown and displayed at the selected position. If you open additional blocks the tag information for these blocks is also shown.
5. If you want to hide the tag information, select the option "Hide" in the "Tag information" drop-down list.
   The tag information is hidden for all open blocks. If you open additional blocks the tag information for these blocks is also hidden.

Displaying or hiding tag information for an opened block

Proceed as follows to display or hide the tag information for an opened block:

1. If you want to display the tag information, select the "Show tag information" option or the "Tag information with hierarchical comments" option in the "Shows the tag information" drop-down list, depending on whether you want to display simple or hierarchical comments.
2. To define the position of the tag information, click the "Position of the tag information (LAD/FBD)" button in the program editor toolbar. The position changes with each click on the button. Alternatively, you can click the small arrow beside the "Position of the tag information (LAD/FBD)" button to open the drop-down list. Than select the position from the drop-down list.
   The tag information is shown and displayed at the selected position.
3. If you want to hide the tag information, select the option "Hide tag information" in the "Shows the tag information" drop-down list.
   The tag information is hidden.

See also

Programming interlocks (Page 7463)

Inserting supervision conditions (Page 7464)

Programming transitions (Page 7469)

Programming permanent instructions (Page 7471)

17.8 Inserting block calls in GRAPH (S7-300, S7-400, S7-1500)

17.8.1 Inserting block calls manually (S7-300, S7-400, S7-1500)

You can call other blocks in actions and permanent instructions.

Inserting block calls manually in an action

To manually insert a block call in an action, follow these steps:

1. Open the GRAPH function block in the single step view.
2. Click in the cell of the action in the "Action" column.
3. Enter "CALL <Name of block to be called>" and confirm your entry with the Enter key.
4. Replace "<Name of block to be called>" with the name of the block you want to call.
   If the block to be called is a function block, the "Call options" dialog opens. In this case, perform steps 5 and 6.
5. In the dialog, enter whether you want to call the block as a single, multi or parameter instance.
   - If you click the "Single instance" button, enter a name for the data block that will be assigned to the call in the entry field "Name".
   - If you click the "Multi-instance" button, in the "Name in the interface" box, enter a name of the tag with which the called function block will be entered as a static tag in the interface of the calling block.
   - If you click on the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
6. Confirm your entry with "OK".

Inserting a block call manually in a permanent instruction

To manually insert a block call in a permanent instruction, follow these steps:

1. Open the view for "Permanent pre-instructions" or "Permanent post-instructions".
2. Right-click the position in the network where you want to insert the block call.
3. Select the "Insert empty box" command in the shortcut menu.
4. Click on "??".
   A text box opens.
5. Enter the name of the block you want to call or open the autocompletion and select the block.
   If the block to be called is a function block, the "Call options" dialog opens. In this case, perform steps 6 and 7.

6. In the dialog, enter whether you want to call the block as a single, multi or parameter instance.
   - If you click the "Single instance" button, enter a name for the data block that will be assigned to the call in the entry field "Name".
   - If you click the "Multi-instance" button, in the "Name in the interface" box, enter a name of the tag with which the called function block will be entered as a static tag in the interface of the calling block.
   - If you click on the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
7. If you call a block that contains monitoring, assign a ProDiag function block to the monitoring functions.
8. Confirm your entry with "OK".

See also

Basics of instances (Page 62)

Block calls (Page 7384)

Inserting block calls using drag-and-drop operation (Page 7481)

Updating block calls in GRAPH (Page 7483)

17.8.2 Inserting block calls using drag-and-drop operation (S7-300, S7-400, S7-1500)

You can call other blocks in actions and permanent instructions.

Inserting block calls in an action using drag-and-drop operation

To insert a block call in an action using drag-and-drop operation, follow these steps:

1. Open the GRAPH function block in the single step view.
2. Drag the block you want to call from the project tree to the cell of the action in the "Action" column.
   If the block to be called is a function block, the "Call options" dialog opens. In this case, perform steps 3 and 4.

3. In the dialog, enter whether you want to call the block as a single, multi or parameter instance.
   - If you click the "Single instance" button, enter a name for the data block that will be assigned to the call in the entry field "Name".
   - If you click the "Multi-instance" button, in the "Name in the interface" box, enter a name of the tag with which the called function block will be entered as a static tag in the interface of the calling block.
   - If you click on the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
4. Confirm your entry with "OK".

Inserting block calls in a permanent instruction using drag-and-drop operation

To insert a block call in a permanent instruction using drag-and-drop operation, follow these steps:

1. Open the view for "Permanent pre-instructions" or "Permanent post-instructions".
2. Drag the block you want to call from the project tree to the position of the network of the permanent instruction where you want to insert the block call.
   If the block to be called is a function block, the "Call options" dialog opens. In this case, perform steps 3 and 4.
3. In the dialog, enter whether you want to call the block as a single, multi or parameter instance.
   - If you click the "Single instance" button, enter a name for the data block that will be assigned to the call in the entry field "Name".
   - If you click the "Multi-instance" button, in the "Name in the interface" box, enter a name of the tag with which the called function block will be entered as a static tag in the interface of the calling block.
   - If you click on the "Parameter instance" button, enter the name of the in/out (InOut) parameter to which the instance should be passed during runtime in the "Name in the interface" text box.
4. If you call a block that contains monitoring, assign a ProDiag function block to the monitoring functions.
5. Confirm your entry with "OK".

See also

Basics of instances (Page 62)

Block calls (Page 7384)

Inserting block calls manually (Page 7480)

Updating block calls in GRAPH (Page 7483)

17.8.3 Updating block calls in GRAPH (S7-300, S7-400, S7-1500)

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have the following options for updating the block calls:

- Explicit updating of all inconsistent block calls in the programming editor.
  The inconsistent block calls within the open block are updated. The following actions are carried out in the process:
  - New parameters are added. Please note, however, that the parameters are not shown for function blocks (FBs) and the parameters are supplied via the corresponding instance data block (DB).
  - Deleted parameters are not removed.
  - Renamed parameters get the new parameter names.
- Explicit updating of a block call in the programming editor.
  The inconsistent call of this block is then updated. The following actions are carried out in the process:
  - New parameters are added. Please note, however, that the parameters are not shown for function blocks (FBs) and the parameters are supplied via the corresponding instance data block (DB).
  - Deleted parameters are not removed.
  - Renamed parameters get the new parameter names.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated. Note that when you call functions (FCs) before the compilation process, all new formal parameters must be supplied with actual parameters.

Updating all inconsistent block calls in the programming editor

To update all block calls in a block, follow these steps:

1. Open the calling block in the programming editor.
2. Click the "Update inconsistent block calls" button in the toolbar.

Updating a specific block call in the programming editor

To update a specific block call in the programming editor, follow these steps:

1. Open the calling block in the programming editor.
2. Right-click on the block call that you want to update.
3. Select the "Update block call" command in the shortcut menu.
4. If parameters were added for a function (FC), enter the values for the new block parameters.

Note

Note that the "Update" command is only available as long as you did not previously update all block calls in the editor with the "Update inconsistent block calls" command.

Updating block calls during compilation

To implicitly update all block calls and uses of PLC data types during compilation, follow these steps:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the "Compile > Software (rebuild all blocks)" command in the shortcut menu.

See also

Block calls (Page 7384)

Inserting block calls manually (Page 7480)

Inserting block calls using drag-and-drop operation (Page 7481)

17.8.4 Changing the block call (S7-300, S7-400, S7-1500)

You have the option of changing the called block for a block call. But keep in mind that no new instance data blocks are created, for example, when changing from a function (FC) to a function block (FB).

Procedure

To change the called block of a block call, follow these steps:

1. Click on the name of the called block within the block call and press the <F2> key. Or double-click the name of the called block.
   A text box opens, and the name of the currently called block is selected.
2. Enter the name of the block you want to call or select a block in the autocompletion.
3. If you want to call an FB, create a new instance data block, if necessary, and specify it as operand.

17.9 Specifying alarms (S7-300, S7-400, S7-1500)

17.9.1 Enabling and disabling the alarm view (S7-300, S7-400, S7-1500)

You can enable or disable the alarm view globally for new GRAPH function blocks or locally for individual blocks. The local setting of a block has priority over the global setting.

Enabling or disabling the alarm view globally

To globally enable or disable the alarm view for new GRAPH function blocks, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. If you want to enable the alarm view, select the "Enable alarms" check box in the "Default settings for new blocks > Alarm properties" area. If you want to disable the alarm view, clear the "Enable alarms" check box.

Enabling or disabling the alarm view locally

To enable or disable the alarm view for an existing GRAPH function block, follow these steps:

1. Open the alarm view for the GRAPH function block.
2. If you want to enable the alarm view, select the "Enable alarms" check box in the "Alarms" area. If you want to disable the alarm view, clear the "Enable alarms" check box.

See also

Interlock and supervision alarms (Page 7389)

Editing alarms (Page 7485)

17.9.2 Editing alarms (S7-300, S7-400, S7-1500)

You specify the alarm for interlocks, supervisions and GRAPH warnings (step time monitoring) in the global supervision settings. You assign the category and the subcategories for interlocks, supervisions and GRAPH warnings in the alarm view of the GRAPH function block.

In addition, you can create step-specific alarm texts for interlocks and supervisions within a GRAPH step.

Note

Step time monitoring is only available for the GRAPH function blocks of the S7-1500 series.

Editing alarms

To edit the alarms for all GRAPH function blocks, follow these steps:

1. Double-click the "Common data > Supervision settings" entry in the project tree.
   The ProDiag supervision settings are opened.
2. Select the "Alarm texts > GRAPH supervisions" group in the area navigation.
3. Use drag-and-drop to move the desired alarm text fields from the "Supported alarm text fields" field to the "Alarm text" text frame.
   You can change the order of the individual text fields at any time.
4. If you want to delete a text field again, right-click the corresponding text field and select "Delete" from the shortcut menu.
5. Select the desired delimiter for the text fields.

or:

1. Open the alarm view for the GRAPH function block.
2. Click the green arrow "Go to supervision settings" to the right of the alarm text.
3. From this point, the procedure described above from step 3 applies.

Select categories and subcategories

To select the categories and subcategories for interlocks, supervisions and GRAPH warnings, follow these steps:

1. Open the GRAPH function block.
2. Open the alarm view of the GRAPH function block.
3. Select the desired entry for interlocks, supervision and for the GRAPH warnings from the drop-down lists for the categories or the subcategories.

Editing the interlock alarm of a single step

To edit the interlock alarm of a single step, follow these steps:

1. Open the step in the single step view.
2. Select the "-(C)-" interlock.
3. Open the Inspector window and switch to the "Properties" tab, if necessary.
4. Enter the desired alarm text in the "Alarm text" text box.
   This text is appended to the end of the GRAPH alarm with the "<Step specific field>" alarm text box.

Editing the supervision alarm of a single step

To edit the supervision alarm of a single step, follow these steps:

1. Open the step in the single step view.
2. Select the "-(v)-" supervision.

3. Open the Inspector window and switch to the "Properties" tab, if necessary.
4. Enter the desired alarm text in the "Alarm text" text box.
   This text is appended to the end of the GRAPH alarm with the "<Step specific field>" alarm text box. With an S7-1500, the text is also used as a step-specific text for step time monitoring if this is used as a supervision condition.

See also

Interlock and supervision alarms (Page 7389)

Enabling and disabling the alarm view (Page 7485)

Structure of the alarm text editor (Page 7879)

Creating alarm texts (Page 7885)

17.10 Acquiring initial values (S7-1500)

17.10.1 Overview of initial value acquisition (S7-1500)

Requirement

In order to enable initial value acquisition, you need to set a version >= 4.0 in the GRAPH function block.

You can find additional information here: Activating initial value acquisition (Page 7489)

Description

You have the option of testing the execution of your user program on the HMI device. The data and values on the HMI device are continuously synchronized with the CPU and updated. You can always see the current program status on the HMI device for each GRAPH function block. The actual values of the signal states of Boolean operands and the results of comparator operations in interlocks and transitions are recorded continuously from the time of activation of the initial value acquisition, even if no error occurs.

The signal states are stored for each cycle in the GRAPH instance data block. A maximum of 32 signal states can be recorded per interlock or per transition. The actual values are stored in the static parameters CRIT_LOC (step) or CRIT (transition) in a DWORD and each signal state occupies one bit. As soon as an interlock error occurs, the values of the CRIT_LOC parameter are copied to the CRIT_LOC_ERR parameter. As soon as a supervision error occurs, the values of the CRIT parameter are copied to the CRIT_FLT parameter in the next transition. These initial values are no longer changed when a new error occurs in the active step.

You can display the actual values with the help of the PLC code view. As soon as an error occurs, you can have the initial values displayed and the bad operands can be immediately recognized (criteria analysis). You can switch between display of the actual values and display of the initial values.

You can find additional information on the visualization of the HMI device here: "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag"

For the named static parameters of the step or transition structures to be displayed, you need to select the maximum interface parameter set for the block interface.

You can find additional information on the interface parameters here: Specifying the interface parameter set (Page 7417)

You can find additional information on the static parameters of a GRAPH function block here: Static parameters (Page 7400)

Supported instructions

All instructions supported in a GRAPH function block for transitions and interlocks are supported for the initial value acquisition.

For bit logic operations, the status of the operand is stored. For comparators, the result of the comparison is stored.

Recording actual values or initial values

The respective interlock of the active steps and subsequent transitions is recorded in each program cycle. If you enable the block property "Permanent processing of all interlocks in manual mode" or the test settings "Edit all interlocks" or "Edit all transitions", all interlocks and all transitions of the GRAPH sequencer will be recorded in every program cycle. That may result in a longer cycle time.

Evaluation order on the HMI device in the PLC code view

The recording of the signal status in the respective DWORD or the list of operands in the symbol table of the PLC code view is carried out in a defined order. In a LAD interlock, for example, the process always starts in the power rail at the top left and ends at the bottom right.

Note

Observe the evaluation order during programming

Also in the criteria analysis on the HMI device, this order is adhered to for the list of the bad operands in the symbol table. Observe this evaluation order for the programming of the interlock or the transition.

Additional information on the order of the list is available here: Example of order of the list (Page 7491)

17.10.2 Activating initial value acquisition (S7-1500)

Procedure

Proceed as follows to activate initial value acquisition:

1. Right-click the GRAPH function block and select "Properties..." from the shortcut menu. The properties dialog of the GRAPH function block opens.
2. Select a version >= V4.0 in the "General > Block" tab.
3. Click on the "Attributes" tab.
4. Activate "Initial value acquisition".
5. Click "OK".
6. Compile and download the user program.

Result

The recording of the initial values begins once the user program has been compiled.

17.10.3 Resetting initial values (S7-1500)

Description

Each GRAPH function block contains the two system-internal parameters "H_IL_ERR" and "H_SV_FLT" in its block parameter set for each individual step. If you have activated the "Initial value acquisition" option in the block properties of the GRAPH function block, the Bit 2 at the "H_IL_ERR" (Interlock) parameter is set to TRUE in the case of an interlock, and the Bit 5 at the "H_SV_FLT" (Transition) parameter is set to TRUE in the case of a monitoring error, as soon as a corresponding event exists in the active step. This enables you to visualize the initial values on your HMI device with the PLC code view.

As of a block version >= 5.0 of a GRAPH function block, you have the option of using input parameter "RESET_CRIT" to reset the signal state of the two static parameters "H_IL_ERR" and "H_SV_FLT" to FALSE again after the criteria analysis. You should use this option, in particular, when you change the programming of interlocks and transitions and you no longer need the initial values that were made obsolete by this. The reprogramming does not automatically reset the signal states back to FALSE.

Procedure

To reset the signal states of the two system-internal static parameters, follow these steps:

1. Check that the version of the GRAPH function block is a version >= 5.0.
2. Select the maximum block parameter record:
   - When you want to create a new GRAPH function block, first select the maximum block parameter record in the general settings for GRAPH.
   - If you want to change the parameter record at a GRAPH function block that has already been created, open the block, click in the block interface and in the menu bar select "Edit > Interface parameters > Maximum interface parameters". Or create the RESET_CRIT parameter manually in the IN section.
3. Connect input parameter "RESET_CRIT" in your user program at the call of the GRAPH function block.

Result

As soon as a positive signal edge is present at input parameter "RESET_CRIT", the signal states of the two system-internal static parameters "H_IL_ERR" and "H_SV_FLT" are reset to FALSE in all the programmed GRAPH steps and the PCL code view on your HMI device no longer displays any obsolete events.

See also

Basics on the block interface of a GRAPH function block (Page 7390)

Overview of initial value acquisition (Page 7488)

Input parameters (Page 7393)

Static parameters of a step (Page 7400)

17.10.4 Example of order of the list (S7-1500)

The recording of the actual or initial values and the list of operands in the symbol table of the PLC code view are subject to the same defined order.

Using the following examples you can see how the recording of the signal states within an LAD interlock and the listing of operands on the HMI device work.

Recording of all signal states

The signal states are written to the static parameter CRIT_LOC in the following or in the T_CRIT output parameter in the following order:

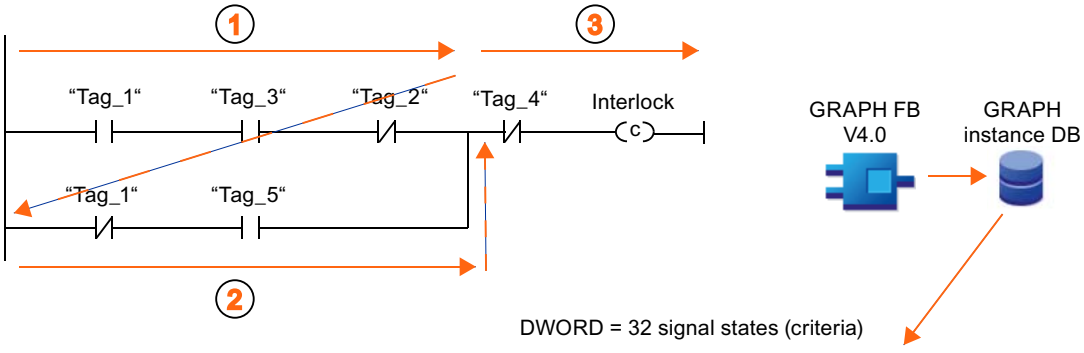

Bit 31	Bit ...	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	Tag_4	Tag_5	Tag_1	Tag_2	Tag_3	Tag_1

On the HMI device, all operands are listed in exactly this order in the PLC code view:

Symbol name	Operand	Comment
Tag_1	e.g. %M2xy	Automatic mode / manual mode
Tag_3		Switch on the motor
Tag_2		Interlock_1
Tag_1		Automatic mode / manual mode
Tag_5		Manual switch
Tag_4		Light barrier

List of operands in case of error

If errors occur in the interlock, the bad operands are marked in color and only the bad operands are listed in the symbol table:

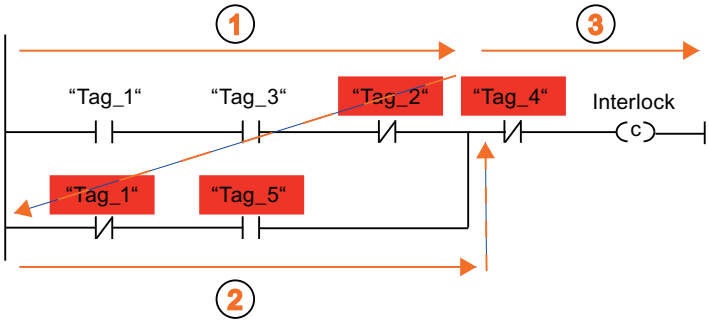

List of the bad operands in the symbol table of the PLC code view:

Symbol name	Operand	Comment
Tag_2	e.g. %M2xy	Interlock_1
Tag_1		Automatic mode / manual mode
Tag_5		Manual switch
Tag_4		Light barrier

Optimized programming of the interlock in case of error

In order that you can see at first glance that the light barrier is responsible for the error, in this example it is beneficial to program the operand "Tag_4: Light barrier" first:

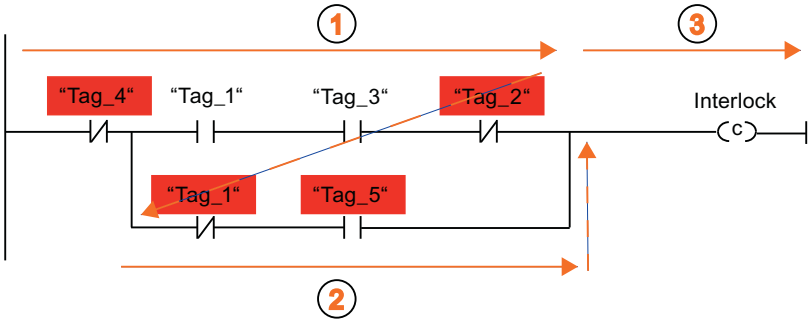

The information about the open light barrier is then displayed in the first line of the symbol table.

Symbol name	Operand	Comment
Tag_4	e.g. %M2xy	Light barrier
Tag_2		Interlock_1
Tag_1		Automatic mode / manual mode
Tag_5		Manual switch

Configuring alarms 18

18.1 Introduction to alarm configuration (S7-300, S7-400, S7-1500)

Overview

Alarms allow you to detect errors in process control in the automation system quickly, to localize them precisely, and to eliminate them. This leads to a significant reduction in down times in the plant.

Before alarms can be output, they need to be configured.

You can create, edit and compile event-dependent alarms along with their alarm texts and alarm attributes and display them on display devices.

The table below lists the alarm types along with a brief description of their functions.

Alarm type	Description
Program alarms	Program alarms are used to report program-synchronous events and are each assigned to a block. They are created in the program editor and edited in the alarm editor.
System diagnostic alarms	System diagnostic alarms are configuration-dependent module events and are activated or deactivated in the hardware configuration. They can only be viewed, not edited, in the alarm editor.
User diagnostic alarms	By means of user diagnostic alarms, you can write a user entry to the diagnostics buffer and send a corresponding alarm. They are assigned to a CPU. They are created in the alarm editor and can be edited there.

18.2 Assigning alarm numbers (S7-300, S7-400, S7-1500)

Assigning numbers

The alarms are identified by a number that is unique within the CPU. This means that it is not necessary to recompile after copying complete programs. Copying a single block is an exception to this rule. Here, recompilation is necessary to include the changed alarm number (ID) in the program.

18.3 Components of an alarm (S7-300, S7-400, S7-1500)

Overview

How an alarm is displayed depends on the alarm method, the alarm block used and the display device.

The possible components of an alarm are listed in the following table:

Component	Description
Time stamp	Generated when the alarm event occurs in the automation system.
Alarm state	The following are possible: Incoming, incoming with acknowledgment, outgoing, outgoing without acknowledgment, outgoing with acknowledgment.
Associated value	With some alarms, it is possible to add an associated value that can be evaluated by the alarm instruction being used.
Image	If there is a system crash, the resulting alarms can be displayed later on HMI devices.
Alarm number	A number (ID) assigned by the system that is unique within the CPU and identifies an alarm.
Alarm texts	Are configured by the user.

18.4 Available alarm blocks (S7-300, S7-400, S7-1500)

S7-300/400: Overview of the alarm blocks

You can choose from the following alarm blocks:

- ALARM
- ALARM_8
- ALARM_8P
- NOTIFY
- ALARM_S
- ALARM_SQ
- AR_SEND (for sending archives; no configuration of alarm texts and alarm attributes possible)
- NOTIFY_8P
- ALARM_DQ
- ALARM_D
- ALARM_T

S7-300/400: When do I use which alarm block?

The following table will help you to decide which alarm block to use for your application. The choice of alarm block depends on the following factors:

- The number of channels available in the block and therefore the number of signals to be monitored per block call.
- The possibility of acknowledging alarms.
- The possibility of including associated values.
- The display devices to be used.
- The configuration limits of the CPU.

Alarm block	Channels	Acknowl-edgment	Associated values	Special features
ALARM *)	1	Yes	up to 10	Sends an alarm on rising or falling edge
ALARM_8 *)	8	Yes	-	Sends an alarm on rising or falling edge of one or more signals
ALARM_8P *)	8	Yes	up to 10	as ALARM_8
NOTIFY	1	no	up to 10	as ALARM
NOTIFY_8P	8	no	up to 10	as ALARM

Alarm block	Channels	Acknowl-edgment	Associated values	Special features
AR_SEND	1	-	-	Used to send an archive; no configuration of alarm texts and alarm attributes possible
ALARM_SQ	1	Yes	1	With each block call and a signal change compared with the previous block call, an alarm is generated
ALARM_S	1	no	1	as ALARM_SQ
ALARM_DQ	1	Yes	1	as ALARM_SQ
ALARM_D	1	no	1	as ALARM_SQ
ALARM_T	-	-	-	Reserved for system test purposes
*) If acknowledgment-triggered alarming is activated, the alarm is not shown in the alarm view.				

S7-1500: Overview of the alarm blocks

The following alarm block is available:

- Program_Alarm

The selection of the corresponding alarm class determines whether the block requires acknowledgment or not. The default setting is that the block does not require acknowledgement.

Alarm block	Channels	Acknowl-edgment	Associated values	Special features
Program_Alarm	1	possible	up to 10	Sends an alarm with time stamp on rising or falling edge

18.5 Alarm type and alarms (S7-300, S7-400, S7-1500)

Using the program editors, you have the option of first creating an alarm type (which means an FB as template for instance DBs) and then alarms (which means instance DBs from the template of the alarm type).

The block with alarm capability can be an FB or an instance DB

- With an FB you create an alarm type that serves as a template for alarms. All the input you make for the alarm type is automatically included for alarms derived from it. If you assign an instance DB to the FB, messages are generated automatically for the instance DB according to the template of the alarm type and alarm numbers are assigned.
- With an instance DB, you can modify the alarms generated based on the alarm type for the specific instance if they are not locked.

The visible difference is that alarm numbers are assigned to alarms whereas they are not assigned to an alarm type. The alarm types and corresponding instances are arranged one under the other in the alarm editor. By default, the table with the alarm instances is collapsed.

Changing the data for an alarm type

If you change data for an alarm type, these changes are automatically included in the instances. Exceptions: You have already changed this data in the instance or have locked or unlocked data later in the alarm type.

Note

If you copy instances to a different program without copying the alarm type, then the instance will not be completely displayed. In this case, copy the alarm type to the new program as well.

Resetting instance-specific data to the value of the alarm type

If attributes or texts were overwritten in an alarm instance, a type icon is displayed beside the attribute in the alarm editor. You can decide whether to use the value of the alarm type again for each attribute. In this case, no type icon is displayed.

See also

Editing the alarm type (Page 7505)

Creating an instance DB and editing alarm instances (Page 7505)

Basics for locking/unlocking texts and attributes (Page 7520)

Locking new alarms and show/hide interlock symbols (Page 7521)

Creating alarm classes (Page 7528)

Editing alarm classes (Page 7529)

Working with the alarm editor (Page 7502)

18.6 Formal parameters, alarm data types and alarm blocks (S7-300, S7-400, S7-1500)

Formal parameters as alarm number inputs (S7-300/400)

For each program alarm or group of program alarms, you require a formal parameter (name of the alarm) in your program that you specify as a STATIC parameter in the tag overview of your program. The formal parameter is used as an alarm number input and forms the basis for an alarm.

Supplying the formal parameter with the suitable data type (S7-300/400)

The formal parameter must be assigned an alarm data type in keeping with the alarm block being used.

Alarm data types and corresponding alarm blocks

The following table shows the alarm data types with their corresponding alarm blocks and their properties. The values of the data types have the same names as the alarm blocks (exception: see table) and have the prefix "C_".

SIMATIC system	Data type	Alarm block	Properties
S7-400	C_Alarm_8	ALARM_8	8 channels, with acknowledgment, no associated values
S7-400	C_Alarm_8p	ALARM_8P	8 channels, with acknowledgment, up to 10 associated values per channel
S7-400	C_Notify	NOTIFY	1 channel, no acknowledgment, up to 10 associated values
S7-400	C_Alarm	ALARM	1 channel, with acknowledgment, up to 10 associated values
S7-300/400	C_Alarm_s	ALARM_S	1 channel, no acknowledgment, up to 1 associated value
S7-300/400	C_Alarm_s	ALARM_SQ	1 channel, with acknowledgment, up to 1 associated value
S7-400	C_Ar_Send	AR_SEND	Used to send an archive
S7-400	C_Notify_8p	NOTIFY_8P	8 channels, no acknowledgment, up to 10 associated values
S7-300/400	C_Alarm_s	ALARM_DQ	1 channel, with acknowledgment, up to 1 associated value

SIMATIC system	Data type	Alarm block	Properties
S7-300/400	C_Alarm_s	ALARM_D	1 channel, no acknowledgment possible, up to 1 associated value
S7-1500	Program_Alarm	Program_Alarm	1 channel, acknowledgment possible by parameter assignment, up to 10 associated values

18.7 Layout of the alarm editor (S7-300, S7-400, S7-1500)

Layout of the alarm editor

The following graphic shows an example of the components in the alarm editor:

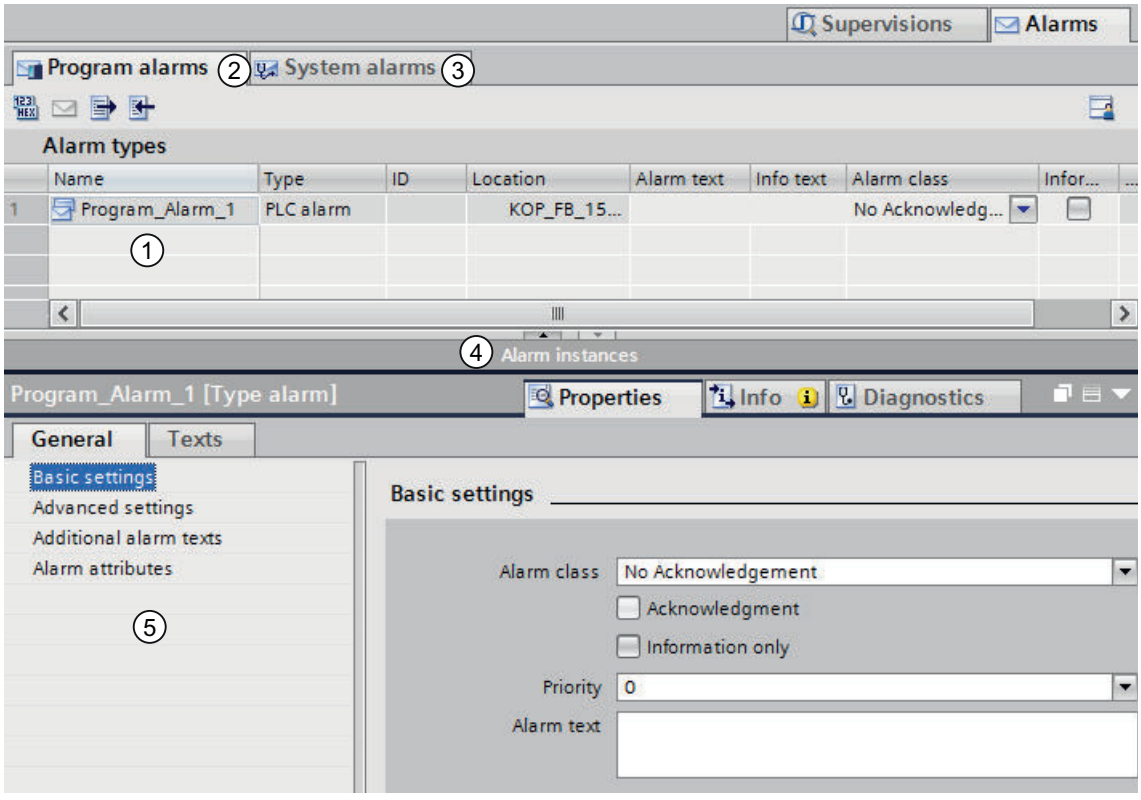

1		Tabular display of the alarms in the work area, see: Working with the alarm editor (Page 7502)
2		"Program alarms" tab: You can edit program alarms here. See Basics of the work area
3		"System alarms" tab: System alarms can only be viewed and cannot be edited. These alarms are configured in the device properties (shortcut menu command "Go to Device") as well as in the project navigation under "Common data > System diagnostics settings".
4		The "Alarm instances" table contains the instances (instance DBs) derived from the alarm type (FB). As default, the table is collapsed.
5		Inspector window
		"User diagnostics alarms" tab: User diagnostics alarms can be created and edited here. The tab is not available for S7-1500 CPUs.
		Save window settings: The current window settings will be restored when the PLC alarms are opened again.

You can enter or modify the necessary parameters, texts and attributes in the table or in the Inspector window. You can sort the table columns in ascending or descending order.

18.8 Working with the alarm editor (S7-300, S7-400, S7-1500)

Below are some basic procedures for working with the alarm editor.

Show/hide alarm instances

The table with the alarm instances is collapsed when you open the alarm editor for the first time. You can change the split between the two tables with the mouse. To do this, click between the two areas and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize the table view, maximize the table view or restore the last selected split.

Sorting tables in ascending or descending order

To sort a table by a column in ascending or descending order, follow the steps below:

1. Click the table header of a column if you want to sort the column in ascending order.
2. Click again on the same column of the table header to sort the column in descending order.
3. Click a third time on the table header of the same column to cancel the sorting.

Copying/pasting cells

You can select one or more cells and insert them at a different position. The data is checked and, if it is valid, the shortcut menu command "Copy" is enabled. The data is checked again at the point of insertion and, if it is valid, the shortcut menu command "Paste" is enabled. If the data is invalid, the shortcut menu commands are grayed out.

18.9 Creating and editing alarms (S7-300, S7-400)

18.9.1 Creating program alarms (S7-300, S7-400)

Requirement

You have created a function block.

Procedure

To create a program alarm, follow these steps:

1. In the "Program blocks" folder in the project navigation, select the function block (FB) for which you want to create a program alarm and double-click the block to open it.
2. Click "Extended instructions" > "Alarms" in the "Instructions" task card and drag the required alarm block (e.g., "Alarm_S") into the instruction window or network (depending on the block language).
   Result: The instruction part of the FB displays the inputs of the called alarm block, in this case the ALARM_S block.
3. Fill in the block interface. For each alarm block that will be called in the FB, you will need to declare tags in the calling FB. For this purpose, enter the following tags, for example:
   – For the "STATIC" parameter, a name for the alarm block input, for example, "Alarm01" and the data type (C_Alarm_s).
   – Assign the name that you entered for the alarm block input to the "EV_ID" tag, in this case "Alarm01" and ID_16#eeee.

   Note

   If, instead of an alarm block in the CPU, you call an FB with multiple instances in which alarms are also configured, you also need to configure the alarms of the FB with multiple instances in the calling block.

4. Repeat steps 2 through 3 for all alarm block calls in this FB.

18.9.2 Editing program alarms in the alarm editor (S7-300, S7-400)

Requirement

You have created a program alarm.

Procedure

To edit program alarms, follow these steps:

1. Double-click "PLC supervisions & alarms" in the project navigation. Select the "Alarms" tab. The alarm editor opens.
2. Enter the required texts and attributes in the appropriate columns.

See also

Basics for locking/unlocking texts and attributes (Page 7520)

Locking new alarms and show/hide interlock symbols (Page 7521)

18.9.3 Editing program alarms in the program editor (S7-300, S7-400)

Requirement

You have created a program alarm.

Procedure

To edit program alarms, follow these steps:

1. Select the appropriate line in the block interface.
2. Switch to the "Alarm" tab in the Inspector window.
3. Enter the required texts and attributes in the appropriate fields.

18.9.4 Deleting program alarms (S7-300, S7-400)

Procedure

To delete a program alarm, follow these steps:

1. Open the block containing the alarm you want to delete.
2. Delete the alarm block inserted during creation in the instruction window or network (depending on the block language).
3. Select the corresponding row in the block interface and select "Delete" in the shortcut menu.

Result

The alarm is deleted.

18.9.5 Editing the alarm type (S7-300, S7-400)

Procedure

To edit an alarm type, follow these steps:

1. Select the required alarm block.
2. Enter the texts you require or select the required attributes.
   If you have selected a multichannel alarm block (for example, "ALARM_8"), you can assign a separate text and certain separate attributes to each subalarm.

   Note

   You must be working in the alarm editor to edit subalarms.

3. If you do not want the texts or attributes of the instance to be changed, lock them in the alarm type.

See also

Basics for locking/unlocking texts and attributes (Page 7520)

Locking new alarms and show/hide interlock symbols (Page 7521)

18.9.6 Creating an instance DB and editing alarm instances (S7-300, S7-400)

Requirement

You have already created an FB and created at least one alarm in it.

Procedure

To assign instance data blocks (DBs) to an alarm type and to edit the alarms for these DBs for specific instances, follow the steps below:

1. Double-click on "Add new block" in the project navigation, click on the "Data block (DB)" button that appears in the dialog and select the function block (alarm type) to which you want to assign the instance DB from the "Type" drop-down list.
2. Double-click "PLC supervisions & alarms" in the project navigation. Select the "Alarms" tab to open the alarm configuration.
3. Enter the desired changes for the respective alarm instance.

Result

This completes the alarm configuration for the created instance DB.

See also

Basics for locking/unlocking texts and attributes (Page 7520)

Locking new alarms and show/hide interlock symbols (Page 7521)

18.9.7 Creating user diagnostic alarms (S7-300, S7-400)

User diagnostics alarms are assigned to a CPU. They are created and edited in the alarm editor.

Requirement

You have created a function block.

Procedure

To create a user diagnostics alarm, follow these steps:

1. In the "Program blocks" folder in the project navigation, select the function block (FB) for which you want to create a user diagnostics alarm and double-click the block to open it.
2. Insert a configuration of the outgoing and incoming user diagnostics alarm in the instruction window of the FB by calling the "WR_USMSG" alarm block twice (each in connection with a negative and a positive edge).
3. Double-click "PLC supervisions & alarms" in the project navigation. Select the "Alarms" tab to open the alarm editor.
4. Select the "User diagnostics alarms" tab in the alarm editor.
5. Click in the table and select "Insert new alarm" in the shortcut menu.

Result

You have created a user diagnostics alarm.

18.9.8 Editing user diagnostic alarms (S7-300, S7-400)

Requirement

You have created a user diagnostic alarm.

The alarm editor is open.

Procedure

To edit a user diagnostic alarm, follow these steps:

1. Enter the required texts and attributes in the appropriate columns.

18.9.9 Deleting user diagnostic alarms (S7-300, S7-400)

You can delete a selected alarm. The texts you configured for this alarm are deleted and the alarm number is released for use.

Procedure

To delete a user diagnostics alarm, follow these steps:

1. Delete the alarm blocks inserted during creation in the instruction window of the FB.
2. In the alarm editor, select the corresponding row in the table and select "Delete" in the shortcut menu.

Result

The alarm is deleted. It is no longer displayed in the table.

18.9.10 Insert associated values in alarms (S7-300, S7-400)

To add current information to alarms, e.g. from the process, you can insert associated values at any location within an alarm text.

Procedure

To insert an associated value into an alarm, follow these steps:

Alarm SFB for the S7-400 (e.g. Alarm, Alarm_8P)

There are 10 associated values whose length is limited to 12 bytes each. They cannot be addressed internally.

1. Configure a block as follows:
   @<No. of the associated value><element type>%<format>@.
2. Insert this block at the locations in the alarm text at which the associated value is to be shown.

Alarm SFC for the S7-300/400 (e.g. Alarm_S)

There is one associated value whose length is limited to 12 bytes. It can be addressed internally with the index and the element type.

1. Configure a block as follows:
   @<Index><element type>%<format>@.
   Index: Index in the associated value. The index is interpreted as an array index (starting with 1).
   The byte offset is calculated with the following formula:
   Offset = ((Index - 1) * data width (element type))
   Use only element types Y, W, X, or R. The element type is used here only to determine the data width. The display type is determined by the format specification.
2. Insert this block at the locations in the alarm text at which the associated value is to be shown.

See also

Structure of associated values (Page 7508)

Examples of associated values (Page 7509)

18.9.11 Structure of associated values (S7-300, S7-400)

Associated values are comprised of the following:

Element type

This uniquely configures the data type of the associated value:

Element type	Data type
Y	BYTE
W	WORD
X	DWORD
I	Integer
D	DINT
B	BOOL
C	CHAR
R	REAL
O	LREAL

The element type only uniquely identifies the data type transferred by the AS. It is not used as Casting Operator.

Format

Determine the output format for the associated value on the display device. The format is preceded by the "%" sign. The following fixed formats apply to alarm texts:

Format	Description
%[i]X	Hexadecimal number with i digits
%[i]u	Decimal number without sign with i digits
%[i]d	Decimal number with sign with i digits
%[i]b	Binary number with i digits
%[i][.y]f	Floating-point number with sign with y digits after the decimal point and total number of digits i
%[i]s	String (ANSI string) with i digits Characters are printed up to the first 0 Byte (00Hex).
%t#<Name of text list>	Access to text list

If the number of digits [i] is too small, the value is nevertheless output in full.

If the number of digits [i] is too large, an appropriate number of fill characters is output before the value.

If the number of digits [i] is too large and if a sign is output (for decimal/floating-point numbers), the number of fill characters is reduced by one. The same applies to periods or commas within floating-point numbers.

Example: Value = 1234.567 Format=%8.3f Output=1234.567; that is, no fill character is output even though there are only 7 digits.

Note

Please note that you can optionally enter "[i]" (without the square brackets).

See also

Insert associated values in alarms (Page 7507)

Examples of associated values (Page 7509)

18.9.12 Examples of associated values (S7-300, S7-400)

Examples of associated values:

@1I%6d@: The value from associated value 1 is shown as a decimal number with a max. 6 digits.

@2R%6f@: The value "5.4", for instance from associated value 2, is shown as a fixed number of points "5.4" (three leading spaces).

@2R%2f@: The value "5.4", for instance from associated value 2, is shown as a fixed number of points "5.4" (if number of digits is too small then these are not cut off).

@1W%t#Textbib1@: Associated value 1 of the WORD data type is the index by which the text to be inserted is referenced in the Textbib1 text library.

Examples of formats and output values:

Value=255 Format=%5X Output="16#000FF"

Value=123 Format=%5u Output=" 123"

Value=-200 Format=%4d Output="-200"

Value=19 Format=%8b Output="2#00010011"

Value = 1234,567 Format=%8.3f Output="1234.567"; that is, no empty character is output even though there are only 7 digits.

Note

If you want to transfer more than one associated value to an ALARM_S block then you can transfer an array with a max. length of 12 bytes. These could be, e.g. maximum 12 Byte or Char, maximum 6 Word or Int or maximum 3 DWord, Real or Dint.

See also

Insert associated values in alarms (Page 7507)

Structure of associated values (Page 7508)

18.9.13 Deleting associated values (S7-300, S7-400)

You can delete associated values by deleting the placeholder that represents an associated value in the alarm text.

Procedure:

Follow the steps below to delete associated values:

1. Find the placeholder in the alarm text that corresponds to the associated value you want to delete.
   The placeholder starts with the "@" character followed by the index and element type by which you can recognize the associated value, then a format specification, and it ends with the "@" character.
2. Delete the placeholder you have found from the alarm text.

18.10 Creating and editing alarms (S7-1500)

18.10.1 Creating program alarms (S7-1500)

Requirement

You have created a function block.

Procedure

To create a program alarm, follow these steps:

1. In the "Program blocks" folder in the project navigation, select the function block (FB) for which you want to create a program alarm and double-click the block to open it.
2. In the instruction window or network of the FB (depending on the block language), insert the call for the selected alarm block ("Program_Alarm"). It is available in the "Instructions" task card under "Extended instructions" > "Alarms". The block interface is automatically provided with the necessary information in the "Static" area.
3. If necessary, change the alarm name in the dialog that opens.
   Result: The instruction part of the FB displays the input tags of the called alarm block, in this case the Program_Alarm block.
4. Repeat steps 2 through 3 for all alarm block calls in this FB.

Please note that you must specify the alarm class when you create alarms for S7-1500 CPUs. Indicates whether the alarm requires acknowledgment. Alarms that do not require acknowledgment can also be used only for information purposes.

As a result, there are three types of alarm:

- The alarm has to be acknowledged and has the status "incoming" or "outgoing".
- The alarm does not have to be acknowledged and has the status "incoming" or "outgoing".
- The alarm is used only for information purposes and has no status. The check mark must be set in the "Information only" column.

Note

The default setting is "without acknowledgment".

See also

Creating alarm classes (Page 7528)

18.10.2 Editing program alarms in the alarm editor (S7-1500)

Requirement

You have created a program alarm.

Procedure

To edit program alarms, follow these steps:

1. Double-click "PLC supervisions & alarms" in the project navigation. Select the "Alarms" tab. The alarm editor opens.
2. Enter the required texts and attributes in the appropriate columns.

See also

Basics for locking/unlocking texts and attributes (Page 7520)

Locking new alarms and show/hide interlock symbols (Page 7521)

18.10.3 Editing program alarms in the program editor (S7-1500)

Requirement

You have created a program alarm.

Procedure

To edit program alarms, follow these steps:

1. Select the appropriate line in the block interface.
2. Switch to the "Alarm" tab in the Inspector window.
3. Enter the required texts and attributes in the appropriate fields.

"Program_Alarm" alarm block

If you use the "Program_Alarm" alarm block, the additional "Configuration" tab is displayed in the Inspector window. You can enter additional associated values, which for example do not occur in texts, at the empty input channels SD_1 to SD_10.

For this, click the selection box next to the parameter and select the desired value.

See also

Basics for locking/unlocking texts and attributes (Page 7520)

Locking new alarms and show/hide interlock symbols (Page 7521)

18.10.4 Deleting program alarms (S7-1500)

Procedure

To delete a program alarm, follow these steps:

1. Open the block containing the alarm you want to delete.
2. Select the corresponding row in the block interface and select "Delete" in the shortcut menu.
3. Delete the alarm block from the instruction window or network (depending on the block language).

Result

The alarm is deleted.

18.10.5 Editing the alarm type (S7-1500)

Procedure

To edit an alarm type, follow these steps:

1. Select the required alarm block.
2. Enter the texts you require or select the required attributes.

See also

Basics for locking/unlocking texts and attributes (Page 7520)

Locking new alarms and show/hide interlock symbols (Page 7521)

18.10.6 Creating an instance DB and editing alarm instances (S7-1500)

Requirement

You have already created an FB and created at least one alarm in it.

Procedure

To assign instance data blocks (DBs) to an alarm type and to edit the alarms for these DBs for specific instances, follow the steps below:

1. Double-click on "Add new block" in the project navigation, click on the "Data block (DB)" button that appears in the dialog and select the function block (alarm type) to which you want to assign the instance DB from the "Type" drop-down list.
2. Double-click "PLC supervisions & alarms" in the project navigation. Select the "Alarms" tab to open the alarm configuration.
3. Enter the desired changes for the respective alarm instance.

Result

This completes the alarm configuration for the created instance DB.

See also

Basics for locking/unlocking texts and attributes (Page 7520)

Locking new alarms and show/hide interlock symbols (Page 7521)

18.10.7 Insert associated values in alarms (S7-1500)

To add current information to alarms, for example from the process, you can insert associated values at various positions within an alarm text.

Procedure using the shortcut menu

To insert an associated value into an alarm, follow these steps:

1. Open the shortcut menu at the desired position, and use the corresponding menu command to select whether you want to insert a keyword, a tag, or an entry from a text list.
2. Enter the desired parameters in the dialog that appears, and confirm your entries.

Note

You can configure associated values both in the alarm editor and the block editor. If you work in the alarm editor, you should have opened the block for which you are configuring the alarm in the block editor. Otherwise, only already configured tags, but no newly added or created tags, can be interconnected or shown in the "Tag" selection box.

If you enter associated values in the alarm editor, you must be in edit mode (click the required box twice) to display the required shortcut menu.

Result

The associated value is integrated into the alarm.

If there is a syntax error, the text is displayed with a red background and must be corrected. Not every combination of tag and formatting is possible.

When tags and text list entries are entered, the "Address" and alarm parameter fields are automatically filled out. They are not displayed until the dialog is opened again. With open block editor, the associated value is added automatically at input channel SD1 to SD10.

Note

If you have set different project languages, the associated values are copied to all project languages. Any changes you make are automatically applied to all project languages. When translating the texts, make sure that the same order is adhered to in all the languages.

Copying embedded data between different Program_Alarm blocks results in errors, unless the two blocks have an identical arrangement of SD tags of the same data type and format.

Procedure without the shortcut menu

You can insert associated values at any location within an alarm text or infotext. To do this, follow these steps:

1. Insert your "Program_Alarm" block in your PLC program.
2. Assign the tag whose value you want to display in the "Program_Alarm" alarm text or in the associated infotext to the SD input of the "Program_Alarm" block.
3. Double-click "PLC supervisions & alarms" in the project navigation. Select the "Alarms" > "Program alarms" tab.
4. Select your new "Program_Alarm" and insert the reference to the required associated value in the format "@SD_number%format_information@" at the required location in the alarm text or infotext.

Note

If you have set different project languages, the associated values are not copied to all project languages. You must insert the associated values in the adapted classical notation for each language. There is no stipulation that the associated values must be in the same order in each language.

See also

Examples of associated values (Page 7517)

Settings when entering associated values (Page 7516)

Special features of associated values (Page 7517)

Use of the associated values (Page 7898)

18.10.8 Settings when entering associated values (S7-1500)

You can insert associated values in alarms both in the alarm editor and the block editor. If you work in the alarm editor, you should have opened the block for which you are configuring the alarm in the block editor. As a result of this, the "Tag" selection box will display the available parameters, which you can then apply. If the block editor is closed, you will see only the previously used parameters.

Entering keyword

The possible keywords that you can define in the text are available for selection in the drop-down list.

Entering a text list entry as an associated value

You select the desired text list in the drop-down list. In the "Tag" selection box select the tag which includes the index for the text list during run time.

Entering a tag as an associated value

You select the desired tag in the "Tag" selection box. The address and the alarm parameter are added automatically by the program. In the "Format" section, you must now define the display type suitable for the specified tag, the minimum length, and, if applicable, the number of decimal places.

Note

Projects created with an older version of the TIA Portal can still contain the manual notation with embedded associated values. This corresponds to the syntax @<Number of the associated value>%<Format specification>@. For information on the "Format specification" parameter, refer to Structure of associated values (Page 7508).

We recommend that you use only the above described dialog boxes to enter associated values in new projects.

When you use ARRAY accesses with variable index in the form #Program_Alarm_1[#myIndex] in your program, the specific ARRAY element to be used during runtime is not yet known at the time of programming. For this reason, when you select this type of ARRAY access in the program editor, the properties of the first ARRAY element are always displayed in the inspector window.

See also

Keywords for alarm texts (Page 7519)

Insert associated values in alarms (Page 7514)

18.10.9 Special features of associated values (S7-1500)

Upgrading older projects

Projects created with an older version of the TIA Portal can contain embedded associated values with different structures in the different project languages. The associated values may have a different order or may be missing in a different language. When a project of this type is upgraded, the order of the associated values is harmonized. It is based on the reference language set in the original project. In this case, it is recommended to have the translated texts checked and corrected.

Importing a translated text list < V14

It may occur that you have exported, translated and re-imported a text list for alarms in an older version. If you try to import an old file in a current project (e.g. due to identical names), this is rejected due to data incompatibility.

18.10.10 Examples of associated values (S7-1500)

Examples of associated values that can be entered in the shortcut menu:

Keyword:

<Keyword: CpuName>

The name of the CPU is entered as an associated value.

Tag:

<Tag: #Temperature>

The value of the "#Temperature" tag is displayed at this point in the alarm text.

Text list entry:

<Text list: USER_1: "tag_1">

The content of the "tag_1" tag is used as index for the "USER_1" text list. The entry from the text list is displayed at this point in the alarm text.

Examples of associated values that can be entered without the shortcut menu:

@1%6d@: The value from associated value 1 (SD_1) is shown as a decimal number with maximum 6 digits.

@2%6f@: The value "5.4", for instance from associated value 2 (SD_2), is shown as a fixed number of points "5.4" (three leading spaces).

@2%2f@: The value "5.4", for instance from associated value 2 (SD_2), is shown as a fixed number of points "5.4" (if number of digits is too small then these are not cut off).

@10%t#Textbib1@: The associated value 10 (SD_10) is shown with the text to be inserted from the "Textbib1" text library.

Examples of formats and output values:

Value=255 Format=%5X Output="16#000FF"

Value=123 Format=%5u Output=" 123"

Value=-200 Format=%4d Output="-200"

Value=19 Format=%8b Output="2#00010011"

Value = 1234.567 Format=%8.3f Output="1234.567"; that is, no empty character is output even though there are only 7 digits.

See also

Insert associated values in alarms (Page 7514)

18.10.11 Deleting associated values (S7-1500)

Procedure:

Follow the steps below to delete associated values:

1. Open the "Configuration" tab in the program editor.
2. Select the required associated value and delete it.
3. Also delete the associated value from the alarm texts in the alarm editor.

Or

1. Select the required input channel in the program editor.
2. Delete the associated value.
3. Also delete the associated value from the alarm texts in the alarm editor.

18.11 Texts and attributes (S7-300, S7-400, S7-1500)

18.11.1 Keywords for alarm texts (S7-300, S7-400, S7-1500)

Overview of keywords for alarm texts

You can enter the following keywords in event texts, info texts, or additional texts. These are then replaced with their real values by the engineering system.

PLC name

Name of the configured PLC.

Example: <Keyword: CpuName> = "PLC_1"

Name of the instance DB

Location of the alarm instance, usually a data block.

Example: <Keyword: Instance> = "Block_3_DB"

FB nesting for alarm

If you are using nested blocks in the project, the tag path of the alarm-defining block is shown that leads from the function block from which the data block is generated through all multi-instances.

Example: <Keyword: Path> = "myFB2\myFB1"

Note

For blocks that are not nested, the "Path" keyword is replaced with an empty string.

Name of the alarm

Name of the alarm.

Example: <Keyword: Name> = "myAlarm"

Note

Keywords cannot be entered in text lists.

18.11.2 Entering texts (S7-300, S7-400, S7-1500)

You can enter the texts for the alarm instances manually or you can use the default values which you input in the texts at the alarm type.. Click in the relevant row and type in the text. If you want to protect the text from being overwritten, click the option in the column. The texts can include line breaks.

Note

Using the character @ in a text box

Please use the @ character only for declaring the associated values, for example @4%6d@. Also, if you use the @ character in the free text of a text field, you might not be able to resolve the associated values contained in the free text of the alarm under certain circumstances.

Text template from the alarm type

All the texts in the alarm type are available as a template for creating alarm texts. If the alarm type already contains a general text, all instances of this alarm type include the same attributes and texts. You can adapt it at the alarm instance, if necessary.

Event text

Event texts can be shown on all display devices.

Info text

The info text is a text that can be specified for certain display devices.

Additional texts

Additional texts are texts that can be displayed by certain display devices.

18.11.3 Basics for locking/unlocking texts and attributes (S7-300, S7-400, S7-1500)

You can lock texts and attributes at the alarm type to prevent them from being altered at the alarm instances. As of version V13 SP1, the interlock symbols are not displayed by default and set automatically for newly generated alarms.

Locking the data for an alarm type

Use the alarm types as templates to facilitate creation of alarms.

- When you enter data (attributes and texts) for the alarm type, you can specify whether or not they are locked. If you lock data, the icon of a closed chain link is shown beside the input box. If the attribute is not locked, the chain link is open.
- If you lock data in the alarm type, you can no longer change this in the alarms of the specific instance. The data is simply displayed.
- If, however, you want to make changes, you will need to go back to the alarm type, cancel the lock and make any changes there.

Note

If you have edited alarm instances in the alarm editor in which the texts/attributes are not locked in the alarm type, a type icon is displayed in front of the relevant column. When you click on this icon, modified texts/attributes are reset to the values of the alarm type.

See also

Locking new alarms and show/hide interlock symbols (Page 7521)

18.11.4 Locking new alarms and show/hide interlock symbols (S7-300, S7-400, S7-1500)

Show/hide interlock symbols

Procedure

To show or hide the interlock symbols, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC alarms" group in the area navigation, click the "Show interlock symbols" check box and set or remove the check mark. Setting the check mark causes the interlock symbols to be displayed; removing the check mark hides the symbols.

Result

The change is applied and does not have to explicitly saved.

Note

As default, the option is disabled.

Set/remove interlock for new program alarms

Procedure

To set or remove the interlock for new program alarms, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC alarms" group in the area navigation, click the "Show/hide interlock symbols on creation of a new program alarm" check box and set or remove the check mark.

Note

The option is activated by default.

18.11.5 Locking texts (S7-300, S7-400, S7-1500)

"Locked" option in the alarm type

You can only lock texts at alarm types. The icon displayed beside the input field indicates whether or not they are locked. The locked texts in all alarm instances derived from the alarm type are write-protected.

Requirement

The interlocking symbols are activated.

Note

The order of the procedure described below is random.

Locking texts

Follow the steps below to lock texts:

- Start by editing the alarm types.
- Click on the icon beside the input box you want to lock.
  Result: The icon changes to a closed chain link.

Unlocking texts

To unlock texts, follow the steps below:

- Start by editing the alarm types.
- Click on the icon beside the input box you want to unlock.
  Result: The icon changes to an open chain link.

See also

Basics for locking/unlocking texts and attributes (Page 7520)

Locking new alarms and show/hide interlock symbols (Page 7521)

Entering texts (Page 7520)

18.11.6 Locking attributes (S7-300, S7-400, S7-1500)

Locking attributes in the alarm type

You can only lock attributes at alarm types. The icon displayed beside the input field indicates whether or not they are locked. The locked attributes in all alarm instances derived from the alarm type are write-protected.

Requirement

The interlocking symbols are activated.

Note

The order of the procedure described below is random.

Locking attributes

Follow the steps below to lock attributes:

- Start by editing the alarm types.
- Click on the icon to the left of the input box you want to lock in the table.
  Result: The icon changes to a closed chain link.

Unlocking attributes

Follow the steps below to unlock attributes:

- Start by editing the alarm types.
- Click on the icon to the left of the input box you want to unlock in the table.
  Result: The icon changes to an open chain link.

See also

Basics for locking/unlocking texts and attributes (Page 7520)

Locking new alarms and show/hide interlock symbols (Page 7521)

18.12 Text lists for alarms (S7-300, S7-400, S7-1500)

18.12.1 Basics of alarm text lists (S7-300, S7-400, S7-1500)

You can adapt existing text lists (user-defined and system-defined text lists) to your requirements and edit texts and attributes. You can then translate the texts into the project language(s) you want to use.

If a signaling block which references a text list is copied into another program, you must either also copy the associated text lists, create another text list with the same name and contents, or change the associated value in the alarm text.

Note

Please note that the names of text lists for messages are not permitted to contain any of the following special characters:

\ / : * ? " < > | @

If you reference a text list that contains one of these characters, a system error is detected. The text is displayed with a red background and must be corrected.

Detailed information on text lists is contained in the "Working with text lists" chapter.

See also

Basics for exporting and importing alarm texts (Page 7533)

18.12.2 Editing text lists for alarms (S7-300, S7-400, S7-1500)

Requirement

- You have created a user-defined text list.

Procedure

Follow the steps below to edit user-defined text lists:

1. Double-click on the "PLC alarm text lists" below the device in the project navigation.
   The text list editor opens.
2. Select the text list you want to edit from the table.
3. Change the desired values.
   You can change the previously created texts or insert new texts in an existing text list.

Note

Please note that text lists for messages are not permitted to contain any of the following special characters:

\ / : * ? " < > | @

If you reference a text list that contains one of these characters, a system error is detected. The text is displayed with a red background and must be corrected.

18.12.3 Integrating texts from text lists in alarms (S7-300, S7-400, S7-1500)

You can integrate texts from various text lists in an alarm. The texts can be positioned anywhere you want which means that they can be used in alarms in foreign languages.

Procedure S7-300/400

To integrate texts from text lists in alarms, follow the steps below:

1. Double-click "PLC supervisions & alarms" in the project navigation. Select the "Alarms" tab. The alarm editor opens.
2. Put an associated value in the format @[Index][Data type]%t#[Text list]@ at the point in the alarm at which you want the text from the text list to appear. The data type must be WORD (W).

   Note

   S7-300/400: [Index] = e.g. 1, where 1 is the first associated value of the alarm.

Procedure S7-1500

To integrate texts from text lists in alarms, follow the steps below:

1. Double-click "PLC supervisions & alarms" in the project navigation. Select the "Alarms" tab. The alarm editor opens.
2. Go to the point in the alarm at which you want the text from the text list to appear, and select from the shortcut menu the option to insert a text list.
3. You select the desired text list in the drop-down list. Then, in the "Tag" selection field, select the tag that will contain the index of the text list entry in the automation system.

   Note

   If you work in the alarm editor, you should have opened the block for which you are configuring the alarm in the block editor.

   If you enter associated values in the alarm editor, you must be in edit mode (click the required box twice) to display the required shortcut menu.

See also

Structure of associated values (Page 7508)

18.12.4 Example of integrating texts from text lists in alarms (S7-300, S7-400, S7-1500)

S7-300/400: Configured alarm text: Pressure is @2W%t#USER_1@.

S7-1500: Configured alarm text: Pressure has <Text list: USER_1: "tag_1">.

Text list with the name "USER_1":

Index	German	English
1734	zu hoch	too high
1735	zu niedrig	too low

Assumption: The associated value of "tag_1" is supplied with the value 1734.

Result: The following alarm text is displayed in run time on the HMI on the operator panel (if the project language is set to "English"): Pressure is too high.

Assumption: The associated value of "tag_1" is supplied with the value 1735.

Result: The following alarm text is displayed in run time on the HMI on the operator panel (if the project language is set to "English"): Pressure is too low.

See also

Integrating texts from text lists in alarms (Page 7526)

18.13 Alarm classes (S7-300, S7-400, S7-1500)

18.13.1 Creating alarm classes (S7-300, S7-400, S7-1500)

You can configure alarm classes to suit your purposes. You can create and edit them in the alarm class editor. An alarm can then by assigned to an alarm class in the alarm editor.

Two alarm classes "Acknowledgement" (for alarms with acknowledgment) and "No Acknowledgement" (for alarms without acknowledgment) are already created by the system. Only the display name can be modified here.

A total of 16 alarm classes can be created to which you can assign the corresponding priority. The "With acknowledgment" property of the alarm classes indicates that it is mandatory to acknowledge the assigned alarms.

Requirement

You have opened the "Common data" folder in project navigation.

Procedure

To configure an alarm class, follow these steps:

1. Double click on the "Alarm classes" entry in project navigation.
   The alarm class editor opens.
2. Select "Insert new alarm class" in the shortcut menu.
   Result: A new alarm class with an automatically assigned name is created.
3. If necessary, enter a different name for the new alarm class in the "Name" column.
   The language of the name you assign here is neutral.
4. If necessary, enter a display name in the "Display name" column. This name is translatable.
5. Specify whether or not alarms of this alarm class require acknowledgment in the "With acknowledgment" column.
6. If necessary, set a priority. This is only possible with user-defined alarm classes.
   The priority is taken over for the alarm type.

Note

Please note that you must specify the alarm class when you create alarms for S7-1500 CPUs. Indicates whether the alarm requires acknowledgment.

S7-1500: Program alarms where the option "Information only" is selected, should be set for as short a time as possible! This avoids multiple archiving (alarm log) on download after changes to program alarm text.

18.13.2 Editing alarm classes (S7-300, S7-400, S7-1500)

You can change the settings (name, display name, priority or acknowledgment) of an alarm class at any time, even if alarms have already been assigned to this alarm class. The changes are applied automatically to the alarms (if the settings at the alarm were not overwritten).

Copying alarm classes

To copy alarm classes, follow these steps:

1. Select the row with the alarm class you want to copy.
2. Select "Copy" in the shortcut menu.
3. Select "Paste" in the shortcut menu.

Result

The copied alarm class is appended to the end of the table under a new name.

The name of the new alarm class is made up as follows:

<old name>_<no.>

no.: This is the lowest free natural number.

Note

- Please note that you must specify the alarm class when you create alarms for S7-1500 CPUs. Indicates whether the alarm requires acknowledgment.
- You can only change the display name for alarm classes generated by the system.

18.13.3 Deleting alarm classes (S7-300, S7-400, S7-1500)

Because you can only create up to 16 alarm classes, it is necessary to delete alarm classes that are no longer required.

Procedure

To delete an alarm class, follow these steps:

1. Double click on the "Alarm classes" entry in project navigation.
   The alarm class editor opens.
2. Select the required alarm class and click the "Delete" icon in the toolbar.

Result

The alarm class is deleted.

Note

Alarm classes generated by the system cannot be deleted.

18.13.4 Priority of the alarm class (S7-300, S7-400, S7-1500)

You can define a priority (0 to 16) for every alarm type for program alarms and user diagnostics alarms. The priority of an alarm class is usually specified or changed in the alarm class editor, but it can be changed later in the alarm editor for individual alarms. The block assigned to the alarm may not be write-protected. When you create new alarms, the priority is set by default to "0".

Changing the priority for alarm types and alarms

If you change data for an alarm type later, these changes are automatically included in the instances. Exception: You have already changed this data in the instance.

The following table lists the possible actions and their effects:

Initial status	Action	Result
No alarm type exists.	The alarm type is created.	The priority of the alarm class in the alarm class editor and the priority at the alarm type in the alarm editor in the "Priority" column are identical.
The alarm type exists, the priority is not changed.	The alarm class of the alarm type is changed.	The priority of the alarm class in the alarm class editor and the priority at the alarm type in the alarm editor in the "Priority" column are identical.
The alarm type exists, the priority is not changed.	The priority of the alarm class is changed in the alarm class editor.	The priority of the alarm class in the alarm class editor and the priority at the alarm type in the alarm editor in the "Priority" column are identical.
The alarm type exists, the priority is not changed.	The alarm class is deleted in the alarm class editor.	The priority of the alarm type is set to "0" and is marked as "invalid".
The alarm type exists, the priority is not changed.	The priority is changed for the alarm type.	The priority of the alarm class in the alarm class editor and the priority at the alarm type in the alarm editor in the "Priority" column are different.
The alarm type exists, the priority for the alarm type was changed.	The alarm class of the instance is changed.	The priority of the alarm class in the alarm class editor and the priority at the instance in the alarm editor in the "Priority" column are different.

Initial status	Action	Result
The alarm (instance) exists, the priority for the alarm type was changed.	The priority of the alarm class is changed.	The priority of the alarm class in the alarm class editor and the priority at the instance in the alarm editor in the "Priority" column are different.
The alarm (instance) exists, the priority for the alarm type was changed.	The alarm class is deleted in the alarm class editor.	The priority for the alarm type is displayed, the alarm class is marked as "invalid".
The alarm (instance) exists, the priority for the alarm type was changed.	The priority for the alarm type is changed to the same value as the priority of the alarm class.	The priority of the alarm class and the priority in the "Priority" column in the alarm editor are identical but not coupled (*).
The alarm (instance) exists, the priority for the alarm type was changed.	The priority for the alarm type is reset to the priority of the alarm class.	The priority of the alarm class and the priority in the "Priority" column in the alarm editor are identical.

If you change the priority at the alarm later, the value is marked with an asterisk (*).

You can reset the changed values again at any time to the priority of the alarm class. To do this, select the "Reset to type alarm value" from the drop-down list.

18.13.5 Exporting and importing alarm classes (S7-300, S7-400, S7-1500)

18.13.5.1 Basics for exporting and importing alarm classes (S7-300, S7-400, S7-1500)

Description

You can export the alarm classes of a project and import them into another project. The advantage of importing alarm classes is that they do not have to be created separately for each project.

The alarm classes are saved in a data file (*.dat).

18.13.5.2 Exporting alarm classes (S7-300, S7-400, S7-1500)

Requirement

A project is open.

Procedure

To export alarm classes, follow these steps:

1. In the project tree, click on the "Shared data" folder and double-click "Alarm classes".
2. In the alarm classes editor, click the "Export of alarm class settings" icon in the toolbar. The "Save as" dialog opens.

3. Select the path to which you want to save the export file.
4. Click the "Save" button.

Result

A .dat file is created at the selected memory location.

18.13.5.3 Importing alarm classes (S7-300, S7-400, S7-1500)

Requirement

The project into which you want to import the alarm classes is open.

Procedure

To import alarm classes, follow these steps:

1. In the project tree, click on the "Shared data" folder and double-click "Alarm classes".
2. In the alarm classes editor, click the "Import of alarm class settings" icon in the toolbar. The "Open" dialog opens.
3. Select the path where the file to be imported is stored.
4. Select the required .dat file.
5. Click the "Open" button.

Note

The "Import" icon is grayed out when the open project is read-only (e.g. reference object).

If an error occurs during the import, an alarm is displayed and the process is aborted.

Result

The alarm classes are displayed.

18.14 Exporting and importing alarm texts (S7-300, S7-400, S7-1500)

18.14.1 Basics for exporting and importing alarm texts (S7-300, S7-400, S7-1500)

Introduction

You can export program alarm texts of the alarm type to a standardized XLSX format for editing with external table editors (for example, translation). Similarly, you can import these alarm texts edited with external table editors to the TIA Portal.

Format of the export file

The names of the column titles are fixed and should not be modified. The sorting order of columns can vary within the spreadsheet. The number of columns depends on the selection made during export. Each alarm is assigned a separate row in the file.

The following table specifies the contents expected for the individual columns:

<table>
<tr><th>Column</th><th colspan="2">Explanation</th><th>Translation-related</th></tr>
<tr><td>Location</td><td colspan="2">Name of the block to which the alarm is assigned.</td><td>No</td></tr>
<tr><td>Alarm name</td><td colspan="2">Name of the alarm</td><td>No</td></tr>
<tr><td>"Alarm text" - English (United States) / [en-US] / Event text</td><td>Event text of the alarm</td><td rowspan="3">The field designation contains a language ID. Alarm texts must be assigned a language ID for import.
An expression with a reference ID will be added to the text for S7-1500 if the alarm text has an associated value. Example: text <field ref="0" />. The associated value is assigned to an alarm text with the ID.</td><td>Yes</td></tr>
<tr><td>"Info text" - English (United States) / [en-US] / Info text</td><td>Info text of the alarm</td><td>Yes</td></tr>
<tr><td>"Additional text" - English (United States) / [en-US] / Additional text</td><td>Additional text of the alarm</td><td>Yes</td></tr>
<tr><td>FieldInfo (for S7-1500 only)</td><td colspan="2">Specifies whether the alarm text contains associated values. The settings are separated by a semicolon ";".
Example for associated values with S7-1500:
Tag: <ref id = 0; type = AlarmTag; Tag = Tag1; DisplayType = Decimal; Length = 5;>
Text list: <ref id = 1; type = CommonTextList; TextList = Textlist1; Tag = tag 2; Length = 5;></td><td>No</td></tr>
</table>

Example for S7-300/400

The figure below shows an excerpt of an export file for S7-300/400 as an example.

	A	B	C	D
1	Location	Alarm name	"Alarm text" - Englisch (USA) / [en-US] / Event text	"Alarm text" - Deutsch (Deutschland) / [de-DE]
2	FB1	BoilerAlarm	Boiler temperature is @1R%6.2f@!	Heizertemperatur ist @1R%6.2f@!
3	FB1	MotorAlarm	Motor @1W%t#MotorNames@ is running at @2W%	Motor @1W%t#MotorNames@ läuft mit @2W%

Note

Note that the keywords (enclosed by $$ or @) are also not to be translated in the example above.

Example for S7-1500

The figure below shows an excerpt of an export file for S7-1500 as an example.

	A	B	C	D	
1	Location	Alarm name	"Alarm text" - English (United States) / [en-US] /	"Meldetext" - German (Germany)	FieldInfo
2	FB1	BoilerAlarm	Boiler temperature is <field ref="0" />!	Heizertemperatur ist <field ref="0	<ref id =
3	FB1	MotorAlarm	Motor <field ref="0" /> is running at <field ref="1	Motor <field ref="0" /> läuft mit <	<ref id =
4					

Translating the alarm texts in an external table editor

The texts are translated in the respective column with the language ID specified for the language to be translated. When translating the texts, make sure that the same order of the associated values is adhered to in all the languages; otherwise, the texts can no longer be imported!

See also

Exporting alarm texts (Page 7534)

Importing alarm texts (Page 7535)

Special features of associated values (Page 7517)

18.14.2 Exporting alarm texts (S7-300, S7-400, S7-1500)

Requirement

The alarm editor is open.

Procedure

To export alarm texts, follow these steps:

1. Click the "Export" button in the "Program alarms" tab in the toolbar. The "Export alarm texts" dialog box opens.
2. Select the path to which you want to save the export file.
3. Select the required languages.
4. If necessary, specify if event texts, info texts and/or additional texts are to be exported as well.
5. Click "OK".

Note

- All project languages selected in TIA Portal are displayed in the export dialog. You can add additional project languages in the project language editor so that they are displayed in this selection. All available project languages are selected by default; you can deselect languages for which the current translation process is of no importance.
- Only export data to an area that is protected with corresponding access mechanisms.

See also

Basics for exporting and importing alarm texts (Page 7533)

18.14.3 Importing alarm texts (S7-300, S7-400, S7-1500)

Requirement

The alarm editor is open.

Procedure

To import alarm texts, follow these steps:

1. Click the "Import" button in the "Program alarms" tab in the toolbar. The "Import alarm texts" dialog box opens.
2. Select the path in which you want to save the file to be imported and click "Import".
3. In the next dialog, select the required language(s).
4. Click "OK".

Note

- You do not have to use the same settings for import as you do for export.
- Only import files from trusted sources.

See also

Basics for exporting and importing alarm texts (Page 7533)

Compiling and downloading PLC programs 19

19.1 Basics for compiling and downloading PLC programs

Compiling and downloading PLC programs

In order for the PLC program you have created to become executable in the automation system, you must first compile the program data that you have created offline and then download it to a device. You can load the program data to devices and memory cards.

Compiling the PLC program

Blocks are always compiled completely to keep the program consistent.

You can find additional information on compiling the PLC program here: Basic information on compiling blocks (Page 7539)

Downloading the PLC program

When the program is downloaded for the first time, the program data is loaded completely. During later download operations, only changes are downloaded.

When downloading to the device, all changes to blocks, PLC data types and PLC tags are included to ensure consistency.

Note

Downloading to device

- When we refer to downloading "blocks" in the following descriptions, we mean the entire PLC program including the PLC data types and the PLC tags.

During the download operation, all information that is required for the reconstruction of the program, including symbolic information such as the names and comments for code and data blocks, is also downloaded in the up to three configured project languages. If you change the project language, you must therefore re-load the program.

The symbolic information is not loaded to the work memory, but rather to the load memory.

After the data has been loaded from a device, the symbolic information is available again in your program, which increases the readability of your program code. Please note, however, that loading to and from a device is not a substitute for storing data in an offline project, as watch tables or multi-language capability of projects cannot be reproduced by loading to and from a device.

After loading from a device, you can only display all data from know-how-protected blocks by entering the correct password.

As of TIA Portal V15, the PLC tags are stored at the time of download according to the tag tables configured offline.

You can find additional information on downloading the PLC program here:

Introduction to downloading blocks (Page 7544) (S7-1200/1500)

Introduction to loading of blocks (Page 7568) (S7-300/400)

Introduction to downloading PLC tags (Page 7582) (S7-1200/1500)

See also

Basic information on compiling blocks (Page 7539)

Introduction to loading of blocks (Page 7568)

Introduction to downloading blocks (Page 7544)

Introduction to downloading PLC tags (Page 7582)

19.2 Compiling blocks

19.2.1 Basic information on compiling blocks

Introduction

A user program must first be compiled before the CPU can execute it. You need to recompile your program each time you make a change.

The following procedures take place during compilation:

- The user program is checked for syntax errors.
- Unneeded instructions are removed from the user program.
- All the block calls within the compiled blocks are checked. In case of changes to the interface of called blocks, errors will be shown in the "Compilation" tab of the information window. You have to correct these errors first.
- The blocks must be numbered uniquely in the user program. If more than one block has the same number, the blocks with number conflicts are renumbered automatically during compilation. A block will not be renumbered in the following cases:
  - The block was selected either individually or as part of a multiselection for the compilation.
  - The number assignment is set to "manual" in the properties of the block.

  Number conflicts that cannot be resolved by automatic renumbering must be corrected manually. Note the messages in the Inspector window for this.

  Note

  Renumbering know-how-protected blocks

  Automatic renumbering and manual renumbering without password of know-how-protected blocks is only possible for CPU of the series S7-1500 and S7-1200 (V4). The know-how protection must also be set with a TIA Portal version V13 SP1 or higher.

- Finally, the user program is compiled into a code that can be read by the CPU.

Compilation methods

You can start compilation in the following windows or editors:

- Compiling blocks in the project tree
  Serves to compile individual blocks or the simultaneous compilation of one or several blocks in the "Program blocks" folder.
- Compiling blocks in the program editor
  This is intended for compilation of a single open block.
- Compiling blocks in the call or dependency structure
  Used to compile individual blocks.
  See also: Call structure (Page 7633), Dependency structure (Page 7640)

Compilation options

If you are compiling blocks in project tree, you have further options:

- Software (only changes)
  All program changes of the selected blocks are compiled. If you have selected a block folder, all program changes to the bocks contained in the folder are compiled.
- Software (compile all blocks)
  All blocks are compiled. This is recommended for the first compilation and after major revisions.
- Software (reset memory reserve)
  All tags declared in the reserve area of the interface of selected blocks are moved to the standard area of the interface. Memory reserve is now available for further interface extensions.

  Note

  This option is only available for CPUs of the S7-1500 and S7-1200 V4 and higher series.

Consistency check

Changing the interfaces of blocks called or PLC data types used can result in inconsistencies between calling blocks and called blocks or between the PLC data types and the global data blocks which use these PLC data types.

To avoid such inconsistencies in the user program, the system performs an automatic consistency check before each compilation process. The time stamps are compared and compilation is then either carried out or canceled depending on the results of the comparison.

- The calling block can only be compiled if the time stamps of the interfaces of the called blocks are older than those of the calling block.
- A global data block based on a PLC data type can only be compiled correctly if the time stamp of the global data block is newer than the time stamp of the PLC data type used.
- The instance data block can only be compiled correctly if the interface time stamps for the interface of the instance data block are identical to those of the assigned function block.

If the compilation process is cancelled, an alarm is displayed in the inspector window. Update the block calls in the relevant blocks and the PLC data types in the global data blocks and then restart compilation. The consistency check also finds know-how protected blocks which cannot be compiled. The corresponding messages will also be shown in the inspector window.

If you start loading immediately instead of first compiling, the blocks selected will be automatically compiled and the block call and global data blocks implicitly updated. Please note the following differences between the CPU families:

- S7-1200/1500: All blocks affected are loaded to ensure no inconsistencies can arise.
- S7-300/400: Only the block selected is loaded.

See also

Compiling blocks in the project tree (Page 7541)
Compiling blocks in the program editor (Page 7542)
Correcting compilation errors (Page 7543)
Block time stamps (Page 6925)
Updating block calls in LAD (Page 7144)
Updating block calls in FBD (Page 7207)
Updating block calls in STL (Page 7272)

19.2.2 Compiling blocks in the project tree

You can compile one block, multiple blocks or all of the blocks in the project tree.

For CPUs of the S7-1500 and S7-1200 V4 series, you can also reset the memory layout of blocks with memory reserve by running a compilation. For more information on memory reserve, refer to chapter "Loading blocks (S7-1200/1500) > Loading block changes without reinitialization".

Requirement

The project tree is open.

Compiling one or more blocks in the project tree

To compile multiple blocks in the project tree, follow these steps:

1. Open the "Program blocks" folder in project tree.
2. Select the blocks you want to compile.
3. Select the "Compile > Software (only changes)" command from the shortcut menu.

Compiling all blocks in the project tree

To compile all blocks in the "Program blocks" folder in project tree, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. You can select one of two different options for the compilation:
   - If you want to compile only the changes since the last compilation, select the "Compile > Software (only changes)" command in the shortcut menu.
   - If you want to compile all blocks completely, select the "Compile > Software (compile all blocks)" command in the shortcut menu.

Resetting memory layout (S7-1500/S7-1200 V4)

Proceed as follows to reset the memory layout of blocks:

1. Select the "Program blocks" folder, or specific blocks in this folder.
2. Select the "Compile > Software (Reset memory reserve)" command from the shortcut menu.

Result

The code for the blocks will be generated if the consistency check has been successful. Instance data blocks generated by the system which are no longer needed will be deleted.

The message under "Info > Compilation" in the inspector window reports whether the compilation was successful.

See also

Basic information on compiling blocks (Page 7539)

Compiling blocks in the program editor (Page 7542)

Correcting compilation errors (Page 7543)

Finding syntax errors in the program (Page 6865)

19.2.3 Compiling blocks in the program editor

Note

Note that the block is recompiled even if you have not made any changes and the time stamp of the block is modified.

Requirement

The block to be compiled is open.

Procedure

To compile a block in the program editor, follow these steps:

1. Right-click in the instruction window of the programming editor.
2. Select the "Compile" command in the shortcut menu.

Result

The code for the block is generated. Instance data blocks generated by the system which are no longer needed will be deleted.

The message under "Info > Compilation" in the inspector window reports whether the compilation was successful.

See also

Basic information on compiling blocks (Page 7539)

Compiling blocks in the project tree (Page 7541)

Correcting compilation errors (Page 7543)

19.2.4 Correcting compilation errors

In the Inspector window in "Info > Compile", you can see whether any compilation was successful or whether errors were detected in the program. If errors occur, you will need to correct them and then start the compilation again.

Procedure

To correct errors following compilation, follow these steps:

1. Open the error list in the Inspector window with "Info > Compile".
2. If there is one, click on the blue question mark next to the error text for information on remedying errors.
3. Double-click the error you want to correct.
   The corresponding error is highlighted.
4. Correct the error.
5. Restart compilation.

See also

Basic information on compiling blocks (Page 7539)

Compiling blocks in the program editor (Page 7542)

Compiling blocks in the project tree (Page 7541)

19.3 Downloading blocks for S7-1200/1500 (S7-1200, S7-1500)

19.3.1 Introduction to downloading blocks (S7-1200, S7-1500)

Downloading blocks to device

So that the CPU can execute the user program, the program data must first be compiled and then downloaded to the device.

You can load the program data to devices and memory cards.

There are different ways to download from and upload to the device:

- By using the commands in the "Online" menu
- By using the shortcut menu
- By using the "Load" button in the toolbar

Loading can be started in the project tree from the following objects:

- From devices
- From the "Program blocks" folder
- From individual blocks
- From the "PLC data types" folder
- From the "PLC tags" folder

Depending on the selected object and the command selected, you can load the following components:

- Downloading to device
  Both the hardware configuration and the software are downloaded to the destination if differences exist between the online and offline versions.
- Extended download to device
  Both the hardware configuration and software are downloaded to the destination if differences exist between the online and offline versions. With extended loading, you can also select a target device and the settings for a target device.
- Load PLC program to device and reset
  The PLC program including all blocks, PLC data types and PLC tags is downloaded to the target and all values are reset to their initial values. Be aware that this also applies to retentive values. The CPU goes to "STOP".
- Download user program to memory card
  The entire program is downloaded onto a plugged-in memory card.

Depending on the selected object and the command selected, you can download the following components from the shortcut menu:

- Hardware and software (only changes)
  Both the hardware configuration and software are downloaded to the destination if differences exist between the online and offline versions.
- Hardware configuration
  Only the hardware configuration is downloaded to the destination.
- Software (only changes)
  Only the objects that differ online and offline are downloaded to the destination.
- Software (all)
  The software is downloaded completely to the destination.

Note

No valid hardware configuration available

If no valid hardware configuration was found on the CPU while loading the software, the existing hardware is loaded as well during a "Download to device".

Note that, in this case, hardware components are also loaded during the download, even though you executed the command for loading software.

Note

To avoid inconsistencies between calling and called blocks, all blocks affected are compiled and loaded after each global change, such as a change in the block interface.

Note

S7-1500

The load memory of S7-1500 series CPUs is on the SIMATIC memory card. Therefore, a SIMATIC memory card absolutely must be inserted in order to operate the CPU.

Uploading blocks from device

You can load the blocks of a device to your project. This is necessary, for example, if you want to edit blocks that only exist in this device. You have the option of loading either all available blocks (organization blocks, function blocks, functions and data blocks) and global PLC tags or individual blocks to the project.

Uploading blocks from or downloading blocks to a memory card

Memory cards are plug-in cards used with an S7-1200 series CPU, for example, to replace the load memory of a device. In the case of S7-1500 series CPUs, they contain the load memory. Only the SIMATIC memory card from Siemens can be used for devices of the S7-1200 and S7-1500 series.

To use a memory card as load memory, you must download the user program or individual blocks to a memory card. You can just as well upload blocks from a memory card back into the project.

Note

S7-1200

Note the following when uploading to or downloading from a memory card:

- If the CPU contains no previous program and you insert an empty memory card in the CPU the program will be loaded from the PG/PC to the memory card and not to the CPU.
- If you insert an empty memory card prior to the startup of the CPU, the program that is on the CPU will be transferred automatically to the memory card. The program on the CPU will then be deleted.
- If you insert a memory card with a program in the CPU prior to the startup of the CPU and the CPU already contains a program, the program on the memory card will be executed and not the program on the CPU. The program on the CPU will be deleted.

Loading GRAPH function blocks

If you load a GRAPH function block together with its instance data block, the processing of the sequencer starts over at the initial step. As a result, problems may occur when synchronizing the sequencer with the process. You can avoid these problems by switching off the sequencer before loading.

Downloading changes without reinitialization

It often proves necessary to edit or expand a PLC program that was already commissioned and that is running on the plant without error. Such operations should be performed without causing any major interruptions of current operations.

S7-1500 therefore offers the option of extending the interfaces of function or data blocks during runtime and loading the modified blocks without setting the CPU to STOP or affecting the value of tags that are already loaded. This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Effects of a load operation on the tag values of a data blocks

When data blocks are downloaded to a device in STOP mode, the next transition of the device to RUN affects the current tag values as follows:

- Tags not marked as being retentive retain their defined start values.
- Retentive tags of the S7-1200 only retain their values if the following conditions are met:
  - You loaded the data block by means of "Download to device > Software (changes only)".
  - You made no changes to the DB structure.

  Otherwise the retentive tags will also retain their defined start values.
- Retentive tags of the S7-1500 only retain their values if the following conditions are met:
  - You loaded the data block by means of "Download to device > Software (changes only)".
  - You made no changes to the structure of the data block or modified it within the memory reserve.

  Otherwise the retentive tags will also retain their defined start values.

Loading blocks with synchronization

In team engineering, it is possible for several users to work on one project with several engineering systems at the same time and access one S7-1500 CPU. To ensure consistency within the shared project, it is necessary to synchronize the changed data prior to loading so that nothing gets overwritten unintentionally.

If differences are determined between the online and offline data management within the shared project during loading that were caused by a different engineering system, automatic synchronization of the data to be loaded is offered during loading.

In this case, the "Synchronization" dialog displays the data to be synchronized with the current status (online-offline comparison) and the possible actions.

The following options are available for synchronization:

Application case	Recommendation	Synchronization
One or more blocks on the CPU (online) are more recent than in the engineering system (offline).	These blocks should be downloaded from the CPU to the engineering system before loading.	Automatic synchronization is possible: The blocks in the engineering system are updated prior to loading.
One or more new blocks have been created and exist only in the CPU (online).	These blocks should be downloaded from the CPU to the engineering system before loading.	Automatic synchronization is possible: The new blocks are added to the engineering system prior to the download.
One or more blocks on the CPU have been deleted.	The blocks should also be deleted prior to the download in the engineering system.	Automatic synchronization is not possible. The blocks deleted on the CPU should be manually deleted in the offline project in the engineering system.

Application case	Recommendation	Synchronization
One or more blocks on the CPU and in the engineering system are different. This is the case when a different user has changed blocks to which you have also made corrections and has already downloaded them to the CPU.	These blocks with competing changes must be adapted manually. You decide in this case which changes you are going to accept. If the blocks on the CPU are to be retained, you should accept these blocks prior to download from the CPU to your engineering system. If the blocks that you have changed are to be applied, you can continue with the download without synchronization.	Automatic synchronization is not possible: The affected blocks on the CPU or in the engineering system must be adapted manually. One of the existing block versions (online or offline) will be overwritten in the process.
There are differences in the hardware configuration on the CPU (online) and in the engineering system (offline).	Differences in the hardware configuration must be adapted manually. You decide in this case which hardware configuration you are going to accept. If the existing hardware configuration on the CPU is to be retained, you should apply these in your engineering system prior to loading. If you want to apply the changed hardware configuration, you can continue with the download without synchronization.	Automatic synchronization is not possible: The hardware configuration must be adapted manually. One of the existing hardware configurations (online or offline) will be overwritten in the process.

You can use the "Force download to device" command to download blocks without synchronization, if desired.

Loading I&M data in PROFINET IO devices to load your modules

When loaded in a device, you can load I&M data in PROFINET IO devices and their modules in addition to the above-mentioned components.

19.3.2 Downloading blocks in the "RUN" operating mode to the device (S7-1200, S7-1500)

Basics on downloading blocks in the "RUN" operating mode

When you download modified blocks to the device, it is not always necessary to switch the device to the "STOP" operating mode. Prior to a download operation, the Engineering System checks whether the device must be stopped before downloading. The result of this check is displayed in the "Load preview" dialog.

If it is necessary to change to the "STOP" operating mode, you cannot continue the download process until you have stopped the CPU.

If the requirements are met, you can also download a modified program or program parts to a CPU in "RUN" operating mode.

The quantity structures may be exceeded for very complex programs which prevents a download in "RUN". In this case you must first create the prerequisites for a download in "RUN".

Tips:

- Use a memory card with sufficient capacity.
- Select a CPU with sufficient work memory.
- If necessary, reduce the number of objects to be loaded (blocks, constants, tags, data types).
  If you cannot download all objects at once, proceed in several steps and download a smaller number of objects in each step.

Note

Actual parameters are not overwritten by a download process in the "RUN" operating mode. Changes to the actual parameters are not made until the next operating state transition from "STOP" to "RUN".

Downloading changes in "RUN"

The table below shows which program and configuration changes can be downloaded in "RUN" operating mode, sorted by CPU family and taking into consideration the firmware versions of the CPUs.

Explanations on the table:

- "RUN": Change can be downloaded to the CPU not only in the STOP operating mode but also in the "RUN" operating mode.
- "RUN (< 57)": The CPU can integrate up to 56 new or modified objects/blocks in one program cycle. If you download more objects/blocks, they are integrated in several successive program cycles. If you want to download all objects/blocks consistently, you must set the CPU to "STOP" operating mode. This number depends on the setting for S7-300 CPUs with configuration option "Process mode/test mode".
- "RUN (Init)": Change can be downloaded in "RUN" operating mode; downloaded data blocks are re-initialized.
- "STOP": Change can only be downloaded in "STOP" operating mode.
- "STOP (Reset)": Change can only be downloaded in "STOP" operating mode; all data including retentive data is reset.

	S7-300	S7-400	S7-1200 V4.0 and higher	S7-1500
Action/type of change	Download possible in mode ...			
Modified properties of HW components. This includes changes to comments in the HW configuration.	STOP	STOP	STOP	STOP
Added HW components	STOP	STOP	STOP	STOP
New/revised text lists (messages)	RUN	RUN	STOP	RUN (V1.1 and higher)

	S7-300	S7-400	S7-1200 V4.0 and higher	S7-1500
Action/type of change	**Download possible in mode ...**			
Revised comments (new, revised, deleted) with the exception of comments in the HW configuration	-	-	**RUN**	**RUN**
Number of blocks downloaded at the same time	**RUN (<17)**	**RUN (<57)**	**RUN (<21)**	**RUN (all)**
Download PLC program to the device and reset	STOP (Reset)	STOP (Reset)	STOP (Reset)	STOP (Reset)
New OB	**RUN**	**RUN**	STOP	**RUN**
Modified OB: Code changes	**RUN**	**RUN**	**RUN**	**RUN**
OB with modified properties (e.g., cycle time change)	**RUN**	**RUN**	STOP	**RUN**
Deleted OB	**RUN**	**RUN**	STOP	**RUN**
New FB/FC/DB/User Data Type (UDT)	**RUN**	**RUN**	**RUN**	**RUN**
Deleted FB/FC/DB/User Data Type (UDT)	**RUN**	**RUN**	**RUN**	**RUN**
Revised FB/FC: Code change	**RUN**	**RUN**	**RUN**	**RUN**
Revised FB/FC: Interface change*	STOP	STOP	**RUN**	**RUN**
Modified DB: Modified property ("Only store in load memory" attribute changed)	STOP	STOP	**RUN (Init)**	**RUN (Init)**
Modified DB (memory reserve not enabled): Name/type of tags modified, tags added or deleted **	**RUN (Init)**	**RUN (Init)**	**RUN (Init)**	**RUN (Init)**
Modified DB (memory reserve enabled): New tag added**	-	-	**RUN**	**RUN**
Modified User Data Type (UDT)	STOP	STOP	**RUN (Init)**	**RUN (Init)**
Add new PLC tags (timer, counter, bit memory)	**RUN**	**RUN**	**RUN**	**RUN**
Modified retentivity settings (timer, counter, bit memory, DB area)	STOP	STOP	STOP	STOP
Motion Control technology objects: Changes to MC Servo cycle clock, change from free-running to cyclical (and vice versa). Changes to the HW interface of the TO	-	-	-	STOP

* If the interface change results in structural changes at the instance DB, see "Modified DB...".

** For the effect of download of data block changes to the data block content, see section "Downloading data blocks to the CPU"

Download of changes in "RUN" with older CPU firmware versions

The table below shows which changes you can download in "RUN" operating mode, in particular, for which older CPU firmware versions.

	S7-1200 V1.0 - 2.1	S7-1200 V2.2 - V3.0
Action/type of change	Download possible in mode ...	
Modified properties of HW components. This includes changes to comments in the HW configuration.	STOP	STOP
Added HW components	STOP	STOP
New/revised text lists (messages)	STOP	STOP
Revised comments (new, revised, deleted) with the exception of comments in the HW configuration	STOP	RUN
Number of blocks downloaded at the same time	STOP	RUN (<11)
Download PLC program to the device and reset	STOP (Reset)	STOP (Reset)
New OB	STOP	STOP
Modified OB: Code changes	STOP	RUN
OB with modified properties (e.g., cycle time change)	STOP	STOP
Deleted OB	STOP	STOP
New FB/FC/DB/User Data Type (UDT)	STOP	RUN
Deleted FB/FC/DB/User Data Type (UDT)	STOP	RUN
Revised FB/FC: Code change	STOP	RUN
Revised FB/FC: Interface change	STOP	STOP
Modified DB: Modified property ("Only store in load memory" attribute changed)	STOP	STOP
Modified DB (memory reserve not enabled): Name/type of tags modified, tags added or deleted	STOP	STOP
Modified User Data Type (UDT)	STOP	STOP
Add new PLC tags (timer, counter, bit memory)	STOP	STOP
Modified retentivity settings (timer, counter, bit memory, DB area)	STOP	STOP

Downloading data blocks to the CPU

Depending on the conditions, the download of new or modified data blocks has an effect on the actual values in the data block:

Download new data blocks	Actual values in the new data blocks are set to start values.
Download structurally modified data blocks (memory reserve not enabled)	Actual values of added tags in the structurally modified data blocks are set to start values.

Download structurally modified data blocks (memory reserve enabled)	The following actual values are retained: • Actual values of tags outside the memory reserve • Actual values of tags that were not modified within the memory reserve Actual values of added tags within the memory reserve are set to start values.
Download of simply modified data blocks (no structural modification)	Actual values are retained.

Additional information

Additional information on download of block extensions without re-initialization and download of modified values for data blocks is available at "See also".

See also

Downloading blocks from program editor to device (Page 7552)

Downloading blocks from the project tree to the device (Page 7554)

19.3.3 Downloading blocks from program editor to device (S7-1200, S7-1500)

Requirement

The block to be downloaded is open.

Procedure

To download a block from the program editor to the device, follow these steps:

1. Right-click in the instruction window of the programming editor.
2. Select the "Download to device" command in the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load".
   - You can save your preferred connection parameters as default under "Options > Settings > Online & Diagnostics". When you first load with new connection parameters, a query is automatically displayed whether you want to store it as a default setting. Click "Yes", if you want to save the current connection parameters as default.
   - You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command. You can also open the "Extended download to device" dialog explicitly via the "Online" menu.
     See also: Establishing and terminating an online connection
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays alarms and proposes actions necessary for loading.
3. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as downloading becomes possible, the "Load" button is enabled.

   Note

   Actions

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   Note

   To avoid inconsistencies between calling and called blocks, download all affected blocks each time you make global changes, such as changes in the block interface. Select the "Consistent download" action.

4. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
5. If you want to start the modules again directly after downloading, select the "Start module" check box.
6. To close the "Load results" dialog box, click "Finish".

Result

The code for the block will be downloaded to the device. If the changes affect additional blocks, these will be compiled and also downloaded to the device. Blocks that only exist online in the device are deleted. Existing CPU data blocks are retained, however. Inconsistencies between the blocks in the user program are avoided by loading all blocks affected and deleting the unneeded blocks in the device.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Downloading blocks from the project tree to the device (Page 7554)

Downloading blocks in the "RUN" operating mode to the device (Page 7548)

19.3.4 Downloading blocks from the project tree to the device (S7-1200, S7-1500)

In the project tree you can download one block, multiple blocks or all blocks to a device.

Downloading one or more blocks from the project tree to the device

To download one block or multiple blocks to the device from the project tree, follow these steps:

1. Open the "Program blocks" folder in project tree.
2. Select the blocks you want to download.
3. Select the "Download to device > Software (only changes)" command from the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load".
   - You can save your preferred connection parameters as default under "Options > Settings > Online & Diagnostics". You will be prompted as such when you download new connection parameters for the first time. Select "Yes", if you want to save the current connection parameters as default.
   - You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command. You can also open the "Extended download to device" dialog explicitly via the "Online" menu.
     See also: Establishing and terminating an online connection
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays alarms and proposes actions necessary for loading.

4. Check the alarms and, where necessary, enable the actions in the "Action" column.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   As soon as downloading becomes possible, the "Load" button is enabled.

5. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
6. If you want to start the modules again directly after downloading, select the "Start all" check box.
7. To close the "Load results" dialog box, click "Finish".

Downloading blocks from the project tree to the device

To download all blocks in the "Program blocks" folder to the device from the project tree, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. Select the "Download to device" submenu in the shortcut menu.
3. If you only want to download the changes since the last download, select the "Software (only changes)" option. If all blocks are to be fully loaded and all values are to be reset to their start values, select "Download PLC program to the device and reset".
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load". You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command. You can also open the "Extended download to device" dialog explicitly via the "Online" menu.
     See also: Establishing and terminating an online connection
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   As soon as downloading becomes possible, the "Load" button is enabled.

5. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
6. If you want to start the modules again directly after downloading, select the "Start module" check box.
7. To close the "Load results" dialog box, click "Finish".

Result

The code for the blocks is downloaded to the device. If the changes affect additional blocks, these will be compiled and also downloaded to the device. Blocks that only exist online in the device are deleted. Inconsistencies between the blocks in the user program are avoided by loading all blocks affected and deleting the unneeded blocks in the device.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Downloading blocks from program editor to device (Page 7552)

Downloading blocks in the "RUN" operating mode to the device (Page 7548)

19.3.5 Uploading blocks from device (S7-1200, S7-1500)

You can load either all the blocks or only individual blocks from a device to your project.

Note

Please note the following:

- Please note that when you load individual blocks, no tags or other required blocks to which you may refer are loaded together with the individual blocks. During the loading operation, references to tags and blocks are reassigned where possible based on the names. After the loading operation, check whether these assignments are correct.
- When loading from a device to an empty project, any existing folder structures for blocks and objects from libraries might not be loaded.
- S7-1500: When GRAPH function blocks are loaded from a device to your project, the step-specific alarm texts for the interlock and supervision alarms are not loaded.
- S7-1500: The start values that you have changed with the instruction "WRIT_DBL: Write to data block in the load memory" will be lost during execution of the action "Upload from device".

Requirement

The online and offline versions of the blocks to be loaded differ or the blocks only exist online.

Loading all blocks from a device

To load all blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to load the blocks.
   See also: Establishing and terminating an online connection
2. In the project tree, select the device folder from which you want to load blocks.
3. In the "Online" menu, select the "Upload from device" command.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Upload from device" button is enabled.
5. Click the "Upload from device" button.
   The loading operation is performed.

Loading individual blocks from a device

To load individual blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to load the blocks.
   See also: Establishing and terminating an online connection
2. In the project tree, select the blocks that you want to load from the device.
3. In the "Online" menu, select the "Upload from device" command.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Upload from device" button is enabled.
5. Click the "Upload from device" button.
   The loading operation is performed.

Result

The blocks will be loaded from the device to the project. You can edit them as normal, recompile them and load them to the device again.

19.3.6 Downloading blocks to a memory card (S7-1200, S7-1500)

Requirement

- The memory card is marked as a program card.
- The "Program blocks" folder of the memory card is open.

Procedure

To download blocks to a memory card, follow these steps:

1. Open the "Program blocks" folder of the device in the project tree.
2. Select the blocks you want to download to the memory card.
3. Drag the blocks in project tree to the "Program blocks" folder of the memory card. You can also copy the blocks and add them to the memory card.
   If necessary, the blocks are compiled. The "Load preview" dialog then opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
5. As soon as downloading becomes possible, the "Load" button is enabled.
6. Click the "Load" button.
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
7. Click "Finish".

Or:

1. In the project tree, select the blocks that you want to upload.
2. Select the "Card reader/USB memory > Write to memory card" command in the "Project" menu.
   The "Select memory card" dialog opens.
3. Select a memory card which is compatible with the CPU.
   A button with a green check mark is activated at the bottom of the dialog.
4. Click the button with the green check mark.
   If necessary, the project data are compiled. The "Load preview" dialog then opens. This dialog displays alarms and proposes actions necessary for loading.
5. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as downloading becomes possible, the "Load" button is enabled.
6. Click the "Load" button.
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
7. Click "Finish".

Result

The blocks are downloaded to the memory card. If the changes affect additional blocks, these will also be downloaded to the memory card. Blocks that exist only on the memory card are deleted. Inconsistencies between the blocks in the user program are avoided by downloading all affected blocks and the deleting of the non-required blocks on the memory card.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Uploading blocks from a memory card (Page 7559)

19.3.7 Uploading blocks from a memory card (S7-1200, S7-1500)

You can only upload all blocks from one memory card back into your project.

Requirement

The memory card is displayed.

See also: Accessing memory cards

Procedure

To upload blocks from a memory card to your project, follow these steps:

1. In the project tree, drag the folder of the memory card to the folder of the device in the project. You can also copy the memory card and insert it in the device.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for loading.
2. Check the alarms and, where necessary, enable the actions in the "Action" column.
   The "Upload from device" button will be enabled as soon as uploading becomes possible.
3. Click on the "Upload from device" button.

See also

Downloading blocks to a memory card (Page 7557)

19.3.8 Switching off the sequencer prior to loading a GRAPH DB (S7-1500)

You can specify the switching off of the sequencer prior to loading an instance data block either globally or during the load operation.

Globally switching off the sequencer

To switch off the sequencer globally for each loading operation of an instance data block, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Select the "Turn off sequence before downloading DB" check box.
   For future loading operations, the sequencer is switched off prior to loading of the instance data block.

Switching off the sequencer during the loading operation

To switch off the sequencer during the loading operation, follow these steps:

1. Load the GRAPH function block to the device.
   During the loading operation, the "Load preview" dialog opens. This dialog displays alarms and suggests the required actions for loading. If the instance data block must be loaded together with the GRAPH function block, the "Load preview" dialog suggests the action "Turn off sequence before downloading the DB".
2. Select the "Turn off sequence before downloading DB" check box.

19.3.9 Loading block extensions without reinitialization (S7-1200, S7-1500)

19.3.9.1 Basic information on loading block extensions without reinitialization (S7-1200, S7-1500)

Description

To enable the subsequent editing of PLC programs that have already been commissioned and are running without error on a system, CPUs of the S7-1200 V4 and higher and S7-1500 series support the option of extending the interfaces of function or data blocks during runtime.

You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Principle of operation

In principle, every function or data block has a memory reserve by default, which can be used for subsequent interface changes. The memory reserve is not used initially. Activate the memory reserve if you have compiled and loaded the block and then decide that you want to load interface changes later. All tags that you subsequently declare will be saved to the memory reserve. A subsequent download has no impact on the values of tags that have already been loaded.

If you decide to revise your program at a later time while the plant is not in operation, you also have the option of resetting the memory layout of one or more blocks in one step. With this action, you move all tags from the reserve area to the regular area. The memory reserve is now cleared and made available for further interface extensions.

Requirements

This "Download without reinitialization" function is available if the following requirements are met:

- The project is in the "TIA Portal V12" format or a higher version.
- You are working with a CPU of the S7-1200 V4 or higher or S7-1500 series.
- The blocks were created in LAD, FBD, STL, or SCL.
- The blocks were created by the user, i.e. they were not included in your delivery package.
- They are blocks with "optimized access".

The following blocks do not offer the option of downloading block extensions without reinitialization:

- GRAPH blocks
- ARRAY data blocks
- ProDiag blocks

Note

Restriction regarding use of the memory reserve

It is not possible to assign the contents of a global data block to a structurally identical data block, e.g. using a move box, if one of the two DBs has a memory reserve.

Basic steps

Perform the following steps if you want to extend the interface of a function or data block and then load the block without reinitialization. A detailed description of the individual steps can be found in the following chapters:

1. By default, all blocks have a memory reserve of 100 bytes in the non-retentive memory. If needed, you can adjust the amount of memory reserve or define additional memory reserve in the retentive memory.
2. Activate the memory reserve.
3. Extend the block interface.
4. Compile the block.
5. Download the block to the CPU as usual.

Overview of settings for memory reserve

The overview of the project tree shows the settings for memory reserve for all program blocks at a glance:

To display the overview, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. Click "Maximizes/minimizes the Overview".

See also

Initializing setpoint values in the online program (Page 7065)

Setting the reserved memory for block extensions (Page 7562)

Activating memory reserve (Page 7564)

Deactivating memory reserve (Page 7565)

Extending the block interface or data block (Page 7565)

Resetting the reserved memory (Page 7566)

19.3.9.2 Setting the reserved memory for block extensions (S7-1200, S7-1500)

Introduction

Essentially, each function or data block has a reserve of 100 bytes by default. This reserve is not located in the block's retentive memory area. By default, no reserve is available in the retentive memory area, since the retentive memory is restricted to the CPU.

You can change the default value of 100 bytes, which applies to all newly created blocks in the project. Furthermore, you can change the size of the reserved memory individually for specific blocks and define a reserve in the retentive memory area for specific blocks.

Setting the amount of reserved memory for new blocks

To set the size of the reserved memory for all newly created blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. In the "Reserved memory for download without reinitialization" group, enter in the "Memory reserve" input box the desired number of bytes to be allocated for a later extension of the block interface.
   Newly created function blocks and global data blocks now get a reserved memory in the specified size. Blocks that already exist are not affected by this change.

Note

The setting is only valid for blocks that support the "Download without reinitialization" function. Other blocks are not influenced by this setting.

Setting the amount of reserved memory for an existing block

To set the amount of reserved memory for an existing block, proceed as follows:

1. Select the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The "Properties" dialog opens.
3. Select the "Download without reinitialization" group in the area navigation.
4. Enter the desired number of bytes in the "Memory reserve" input box.
5. To define a reserve in retentive memory, select the "Enable download without reinitialization for retentive tags" check box and enter the desired number of bytes in the "Retentive reserved memory" input box.

Note

Please note the following:

- If you have already activated reserved memory for the current block, you can no longer change the size of the reserved memory.
- In the memory, data elements are sorted according to specific rules that allow the fastest possible access while the program is running. This is why the available memory is sometimes reduced by more than the actual size of the data element when a new data element is added.
  For example, the available memory area may be reduced by four bytes when an element of the "WORD" type and an element of the "BYTE" type are added, since all data elements are aligned to a WORD limit.

See also

Basic information on loading block extensions without reinitialization (Page 7560)

19.3.9.3 Activating memory reserve (S7-1200, S7-1500)

Introduction

Each function or data block is always assigned a default memory reserve. However, this memory reserve is not used initially. Activate the memory reserve if you have compiled and loaded the block and then decide that you want to load interface changes later. All tags that you subsequently declare will be saved to the memory reserve.

Requirements

- The block is currently compiled.
- The block contains a memory reserve.

Procedure

To activate the memory reserve for a function block or global data block, proceed as follows:

1. Open the block.
2. Select the "Keep actual values" button in the toolbar:

Result

- As of now, all newly declared tags will be stored in the memory reserve and can therefore be downloaded without affecting the process that is running.
- It is no longer possible to delete existing tags. Their properties can no longer be changed, except for the properties "Default value", "Start value" and "Comment".
- Local constants which are used as ARRAY limits can no longer be changed, because changing the constant value could have an effect on the memory layout of the block.

See also

Basic information on loading block extensions without reinitialization (Page 7560)

19.3.9.4 Deactivating memory reserve (S7-1200, S7-1500)

Introduction

In order to revise the memory layout of a block at a later time when the plant is not in operation, you can deactivate the reserved memory. With this action, you move all tags from the reserve area to the regular area. The reserved memory remains in existence and is now made available for further interface extensions. It then has the size which is defined in the block properties again.

Note

Please note that the block has to be compiled and loaded again after you have deactivated the reserved memory. It is not possible to perform a "Download without reinitialization" after deactivating the reserved memory.

Requirement

The block's reserved memory is activated.

Procedure

To activate the reserved memory for a function block or global data block, proceed as follows:

1. Open the block.
2. Select the "Keep actual values" button in the toolbar:

Result

With this action, you move all tags that were previously located in the reserve area to the regular area. As a result, the block has to compiled and loaded again. The values of the tags in the CPU are reinitialized during loading.

See also

Basic information on loading block extensions without reinitialization (Page 7560)

19.3.9.5 Extending the block interface or data block (S7-1200, S7-1500)

Introduction

You can add new tags after you have activated the reserved memory of a function block or global data block.

Requirement

The reserved memory is activated.

Procedure

To declare additional tags, follow these steps:

1. Select a declaration section, for example "Input", "Output", "InOut" or "Static".
2. In the selected section, declare as usual one or more tags and enter their properties in the columns. In the "Retain" column of an FB, you can select only between the "Retain" and "Non-retain" settings. The "Set in IDB" setting is not available in the reserved memory.

   Note

   It is not possible to add new tags within a tag with structured data type. You cannot declare any new structure elements within an existing structure, for example.

   However, you can create new structures within the reserved memory.

3. Compile the changed blocks, for example by selecting them in the project tree and selecting the command "Compile> Software (only changes)" in the shortcut menu.
   During compiling, the newly declared tags are inserted into the reserved memory. If the reserved memory is too small, the compiling is terminated and an error message notifies you about this.
4. Load the changed blocks, for example by selecting them in the project tree and selecting the command "Load > Software (only changes)" in the shortcut menu.
   When you load block extensions, only the newly added tags with defined start values are initialized. The existing online tags are not reinitialized.

See also

Basic information on loading block extensions without reinitialization (Page 7560)

19.3.9.6 Resetting the reserved memory (S7-1200, S7-1500)

Introduction

If you decide to revise your program at a later time while the plant is not in operation, you have the option of resetting the reserved memory of one or more blocks in one step. With this action, you move all tags from the reserved memory to the regular memory area. The reserved memory is now cleared and made available for further interface extensions.

Requirement

- The block's reserved memory is activated.
- The reserved memory contains tags.

Procedure

To reset the reserved memory of one or more blocks, follow these steps:

1. Select the "Program blocks" folder, or specific blocks in this folder.
2. Select the "Compile > Software (Reset memory reserve)" command from the shortcut menu.

Result

All tags in the selected blocks that were previously located in the reserved memory area are moved from this area to the regular area. The block is recompiled. The tags are reinitialized during the next loading. The block's configured reserved memory is retained and continues to be active.

DANGER

Reinitialization at the next load

When you reset the reserved memory, all the tags that were previously in the reserved memory are reinitialized the next time you load. This is true for both retentive and non-retentive tags.

Changing tag values during ongoing plant operation can cause serious material damage and personal injuries! It is therefore very important that you check the program thoroughly in a test environment after resetting the reserved memory before putting it in operation.

See also

Basic information on loading block extensions without reinitialization (Page 7560)

19.4 Downloading blocks for S7-300/400 (S7-300, S7-400)

19.4.1 Introduction to loading of blocks (S7-300, S7-400)

Downloading blocks to device

So that the CPU can execute the user program, the program data must first be compiled and then downloaded to the device.

You can load the program data to devices and memory cards.

There are different ways to download from and upload to the device:

- By using the commands in the "Online" menu
- By using the shortcut menu
- By using the "Load" button in the toolbar

Loading can be started in the project tree from the following objects:

- From devices
- From the "Program blocks" folder
- From individual blocks
- From the "PLC data types" folder

Depending on the selected object and the command selected, you can load the following components:

- Downloading to device
  Both the hardware configuration and the software are downloaded to the destination if differences exist between the online and offline versions.
- Extended download to device
  Both the hardware configuration and software are downloaded to the destination if differences exist between the online and offline versions. With extended loading, you can also select a target device and the settings for a target device.
- Load PLC program to device and reset
  The PLC program including all blocks and PLC data types is downloaded to the target and all values are reset to their initial values. Be aware that this also applies to retentive values. The CPU goes to "STOP".
- Download user program to memory card
  The entire program is downloaded onto a plugged-in memory card.

Depending on the selected object and the command selected, you can download the following components from the shortcut menu:

- Hardware and software (only changes)
  Both the hardware configuration and software are downloaded to the destination if differences exist between the online and offline versions.
- Hardware configuration
  Only the hardware configuration is downloaded to the destination.
- Software (only changes)
  Only the objects that differ online and offline are downloaded to the destination.
- Software (all)
  The software is downloaded completely to the destination.

Note

No valid hardware configuration available

If no valid hardware configuration was found on the CPU while loading the software, the existing hardware is loaded as well during a "Download to device".

Note that, in this case, hardware components are also loaded during the download, even though you executed the command for loading software.

Note

Downloading more than 16 blocks in RUN mode

If more blocks are to be downloaded in RUN than permitted by the CPU capacity for a download process, the blocks to be downloaded can be downloaded asynchronously over multiple cycles. When this error occurs, you can select this option in the "Load preview" dialog from the drop-down list.

Keep in mind that a consistent state of the CPU is not reached again until the download over multiple cycles is complete.

Note

Downloading know-how-protected blocks

If you download a know-how-protected block to a device, no restore information is installed along with it. This means that you cannot reopen a know-how-protected block if you upload it from the device.

Note

Loading CPU S7-300/400

When loading from a device, please note that alarm information may be lost and that this could lead to errors during compilation.

Loading PLC programs from device

You can download the program data loaded on a device back to your project.

This includes the existing blocks and PLC data types.

This is necessary, for example, if you want to edit blocks that only exist in this device. You can either upload all existing blocks (organization blocks, function blocks, functions, data blocks) and PLC data types or just individual blocks to the project.

Note

Loading CPU S7-300/400

When downloading from a device, please note that alarm information may be lost and that this could lead to errors during compilation.

Uploading blocks from or downloading blocks to a memory card

Memory cards are insertable cards used, for example, to expand the memory of a CPU and to protect data against power failures. You can use different types of memory cards depending on the CPU. For additional information on memory cards approved for your CPU, see "Additional information on configurations" in the section "Functional description of S7-300/400 CPUs".

Loading GRAPH function blocks

If you load a GRAPH function block together with its instance data block, the processing of the sequencer starts over at the initial step. As a result, problems may occur when synchronizing the sequencer with the process. You can avoid these problems by switching off the sequencer before loading.

Effects of a load operation on the tag values of a data blocks

When data blocks are downloaded to a device in STOP mode, the next transition of the device to RUN affects the current tag values as follows:

- Tags of non-retentive data blocks retain their defined start values.
- Tags of retentive data blocks only retain their values if the following conditions are met:
  - You loaded the data block by means of "Download to device > Software (changes only)".
  - You made no changes to the DB structure.

  Otherwise the tags of retentive data blocks also retain their defined start values.

Loading I&M data in PROFINET IO devices to load your modules

When loaded in a device, you can load I&M data in PROFINET IO devices and their modules in addition to the above-mentioned components.

19.4.2 Downloading blocks in the "RUN" operating state to the device (S7-300, S7-400)

Basics about loading blocks in RUN mode

When you load modified blocks to a device, it is not always necessary to set the device to STOP mode. During the loading process, the engineering system checks whether the device has to be stopped before the loading. The result of the check is displayed in the "Load preview" dialog.

If it is necessary to change to STOP mode, you cannot continue the loading process until you have stopped the CPU.

If the requirements are met, you can also load a modified program or program parts in a CPU in RUN mode.

Basically, there are the following restriction when loading in RUN mode:

- Depending on the device used, the number and the type of blocks that you can load in RUN mode can be limited.
- You have the option to download the blocks to the device in RUN over multiple cycles. This is only possible when you use the "Download > Software (only changes)" command.
- It is not possible to load all the blocks using "Download PLC program to the device and reset".
- If you have changed the hardware configuration of the device that is to be used as the target of the loading process, loading in RUN mode is not possible.

Note

Actual parameters are not overwritten by a loading process in RUN mode. Modifications to the actual parameters only take place at the next change of operating mode from STOP to RUN.

Loading changes in RUN mode

The following table shows which program or configuration changes can be loaded in RUN mode, sorted according to CPU series, and taking into account the firmware versions of the CPUs.

Explanations on the table:

- RUN: Change can be loaded in the CPU in STOP mode as well as in RUN mode.
- RUN (< 57): The CPU can integrate up to 56 new or modified objects/blocks in one program cycle. If you download more objects/blocks, they are integrated in several successive program cycles. If you want to load all objects/blocks consistently, you must set the CPU to STOP mode. This number depends on the setting for S7-300 CPUs with configuration option "Process mode/test mode".
- RUN (init): Change can loaded in RUN mode; loaded data blocks are re-initialized.
- STOP: Change can only be loaded in the STOP mode.
- STOP (reset): Change can only be loaded in STOP mode. All data including retentive data is reset.

	S7-300	**S7-400**	**S7-1200 V4.0 and higher**	**S7-1500**
Action/type of change	**Download possible in operating state ...**			
Modified properties of HW components. This includes changes to comments in the HW configuration.	STOP	STOP	STOP	STOP
Added HW components	STOP	STOP	STOP	STOP
New/revised text lists (alarms)	**RUN**	**RUN**	STOP	**RUN (V1.1 and higher)**
Revised comments (new, revised, deleted) with the exception of comments in the HW configuration	-	-	**RUN**	**RUN**
Number of blocks downloaded at the same time	**RUN (<17)**	**RUN (<57)**	**RUN (<21)**	**RUN (all)**
Download PLC program to the device and reset	STOP (Reset)	STOP (Reset)	STOP (Reset)	STOP (Reset)
New OB	**RUN**	**RUN**	STOP	**RUN**
Modified OB: Code changes	**RUN**	**RUN**	**RUN**	**RUN**
OB with modified properties (e.g., cycle time change)	**RUN**	**RUN**	STOP	**RUN**
Deleted OB	**RUN**	**RUN**	STOP	**RUN**
New FB/FC/DB/User Data Type (UDT)	**RUN**	**RUN**	**RUN**	**RUN**
Deleted FB/FC/DB/User Data Type (UDT)	**RUN**	**RUN**	**RUN**	**RUN**
Revised FB/FC: Code change	**RUN**	**RUN**	**RUN**	**RUN**
Revised FB/FC: Interface change*	STOP	STOP	**RUN**	**RUN**

	S7-300	S7-400	S7-1200 V4.0 and higher	S7-1500
Action/type of change	**Download possible in operating state ...**			
Modified DB: Modified property ("Only store in load memory" attribute changed)	STOP	STOP	**RUN (Init)**	**RUN (Init)**
Modified DB (memory reserve not enabled): Name/type of tags modified, tags added or deleted **	**RUN (Init)**	**RUN (Init)**	**RUN (Init)**	**RUN (Init)**
Modified DB (memory reserve enabled): New tags added**	-	-	**RUN**	**RUN**
Modified User Data Type (UDT)	STOP	STOP	**RUN (Init)**	**RUN (Init)**
Add new PLC tags (timer, counter, bit memory)	**RUN**	**RUN**	**RUN**	**RUN**
Modified retentivity settings (timer, counter, bit memory, DB area)	STOP	STOP	STOP	STOP
Motion Control technology objects: Changes to MC Servo cycle clock, change from free-running to cyclical (and vice versa). Changes to the HW interface of the TO	-	-	-	STOP

* If the interface change results in structural changes at the instance DB, see "Modified DB...".

** For information on the effect of downloading data block changes on the data block content, see section "Downloading data blocks to the CPU".

Loading changes in RUN with older CPU firmware versions

The table below shows the changes you can load in RUN mode and for older CPU firmware versions.

	S7-1200 V1.0 - 2.1	S7-1200 V2.2 - V3.0
Action/type of change	**Download possible in operating state ...**	
Modified properties of HW components. This includes changes to comments in the HW configuration.	STOP	STOP
Added HW components	STOP	STOP
New/revised text lists (alarms)	STOP	STOP
Revised comments (new, revised, deleted) with the exception of comments in the HW configuration.	STOP	**RUN**
Number of blocks downloaded at the same time	STOP	**RUN (<11)**
Download PLC program to the device and reset	STOP (Reset)	STOP (Reset)
New OB	STOP	STOP
Modified OB: Code changes	STOP	**RUN**
OB with modified properties (e.g., cycle time change)	STOP	STOP
Deleted OB	STOP	STOP
New FB/FC/DB/User Data Type (UDT)	STOP	**RUN**

	S7-1200 V1.0 - 2.1	S7-1200 V2.2 - V3.0
Action/type of change	**Download possible in operating state ...**	
Deleted FB/FC/DB/User Data Type (UDT)	STOP	**RUN**
Revised FB/FC: Code change	STOP	**RUN**
Revised FB/FC: Interface change	STOP	STOP
Modified DB: Modified property ("Only store in load memory" attribute changed)	STOP	STOP
Modified DB (memory reserve not enabled): Name/type of tags modified, tags added or deleted	STOP	STOP
Modified User Data Type (UDT)	STOP	STOP
Add new PLC tags (timer, counter, bit memory)	STOP	STOP
Modified retentivity settings (timer, counter, bit memory, DB area)	STOP	STOP

Downloading data blocks to the CPU

Depending on the conditions, the download of new or modified data blocks has an effect on the actual values in the data block:

Download new data blocks	Actual values in the new data blocks are set to the initial values.
Download structurally modified data blocks (memory reserve not enabled)	Actual values of added tags in the structurally modified data blocks are set to initial values.
Download structurally modified data blocks (memory reserve enabled)	The following actual values are retained: • Actual values of tags outside the memory reserve • Actual values of tags that were not modified within the memory reserve. Actual values of added tags within the memory reserve are set to initial values.
Download of simply modified data blocks (no structural modification)	Actual values are retained.

Additional information

Additional information on download of block extensions without re-initialization and download of modified values for data blocks is available at "See also".

See also

Downloading blocks in the program editor to device (Page 7575)

Downloading blocks from the project tree to the device (Page 7576)

19.4.3 Downloading blocks in the program editor to device (S7-300, S7-400)

Requirement

The block to be downloaded is open.

Procedure

To load a block from the program editor to the device, follow these steps:

1. Right-click in the Instruction window of the program editor.
2. Select the "Download to device" command in the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load".
   - You can save your preferred connection parameters as default under "Options > Settings > Online & Diagnostics". When you first load with new connection parameters, a query is automatically displayed whether you want to store it as a default setting. Click "Yes", if you want to save the current connection parameters as default.
   - You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command. You can also open the "Extended download to device" dialog with the "Online" menu.
     See also: Establishing and terminating an online connection
   - If the device is already online, the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays messages and recommends actions necessary for loading.
3. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon loading becomes possible, the "Load" button is enabled.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   Note

   To avoid inconsistencies between calling and called blocks, download all affected blocks each time you make global changes, such as changes in the block interface. Select the "Consistent download" action.

4. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays alarms and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.

5. If you want to start the modules again directly after downloading, select the "Start all" check box.
6. To close the "Load results" dialog box, click "Finish".

Result

The code for the block is downloaded to the device. If you have selected "Consistent download", then all blocks that are subject to this change will be compiled and downloaded to the device.

The messages under "Info > General" in the Inspector window report indicate whether the loading process was successful.

See also

Downloading blocks from the project tree to the device (Page 7576)

Downloading blocks in the "RUN" operating state to the device (Page 7571)

19.4.4 Downloading blocks from the project tree to the device (S7-300, S7-400)

In the project tree you can download one block, multiple blocks or all blocks to a device.

Loading one or more blocks from the project tree to the device

To load one or more blocks to the device from the project tree, follow these steps:

1. Open the "Program blocks" folder in project tree.
2. Select the blocks you want to download.
3. Select the "Download to device > Software (only changes)" command from the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load". You can also open the "Extended download to device" dialog with the "Online" menu.
     See also: Establishing and terminating an online connection
   - If the device is already online, the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays messages and recommends actions necessary for loading.

4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Load" button is enabled.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   Note

   To avoid inconsistencies between calling and called blocks, download all affected blocks each time you make global changes, such as changes in the block interface. Select the "Consistent download" action.

5. Click "Load".
   The block is loaded and the "Load results" dialog opens. This dialog shows you the status and the actions after downloading.
6. To close the "Load results" dialog box, click "Finish".

Loading blocks from the project tree to the device

To download all blocks in the "Program blocks" folder to the device from the project tree, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. Select the "Download to device" submenu in the shortcut menu.
3. If you only want to download the changes since the last download, select the "Software (only changes)" option. If all blocks are to be fully loaded and all values are to be reset to their start values, select "Download PLC program to the device and reset".
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load". You can also open the "Extended download to device" dialog with the "Online" menu.
     See also: Establishing and terminating an online connection
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays alarms and proposes actions necessary for loading.

4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Load" button is enabled.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   Note

   To avoid inconsistencies between calling and called blocks, download all affected blocks each time you make global changes, such as changes in the block interface. Select the "Consistent download" action.

5. Click "Load".
   The block is loaded and the "Load results" dialog opens. This dialog shows you the status and the actions after downloading.
6. To close the "Load results" dialog box, click "Finish".

Result

The code for the blocks is downloaded to the device. If the changes affect additional blocks, these will be compiled and also downloaded to the device. Blocks that only exist online in the device are deleted. Inconsistencies between the blocks in the user program are avoided by loading all blocks affected and deleting the unneeded blocks in the device.

The messages under "Info > General" in the Inspector window report indicate whether the loading process was successful.

See also

Downloading blocks in the program editor to device (Page 7575)

Downloading blocks in the "RUN" operating state to the device (Page 7571)

19.4.5 Uploading blocks from device (S7-300, S7-400)

You can upload either all the blocks or only individual blocks from a device to your project.

Note

Although SCL blocks and GRAPH function blocks can be loaded from an S7-300/400, they cannot be edited once loaded.

Requirement

The online and offline versions of a blocks to be loaded differ or the blocks only exist online.

Downloading all blocks from a device

To download all blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to upload the blocks.
   See also: Establishing and terminating an online connection
2. In the project tree, select the device folder from which you want to upload blocks.
3. In the "Online" menu, select the "Upload from device" command.
   The "Upload preview" dialog box opens. This dialog displays messages and recommends actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as uploading becomes possible, the "Upload from device" button is enabled.
5. Click the "Upload from device" button.
   The load process is executed.

Downloading individual blocks from a device

To upload individual blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to upload the blocks.
   See also: Establishing and terminating an online connection
2. In the project tree, select the blocks that you want to upload from the device.
3. In the "Online" menu, select the "Upload from device" command.
   The load process is executed.

Result

The blocks will be uploaded from the device to the project. You can edit, recompile and load the blocks in the device again.

19.4.6 Downloading blocks to a memory card (S7-300, S7-400)

Requirement

The "Program blocks" folder of the memory card is open.

Procedure

To download blocks to a memory card, follow these steps:

1. Open the "Program blocks" folder of the device in the project tree.
2. Select the blocks you want to download to the memory card.
3. Drag the blocks in project tree to the "Program blocks" folder of the memory card. You can also copy the blocks and add them to the memory card.
   If necessary, the blocks are compiled. The "Load preview" dialog then opens. This dialog displays messages and recommends actions necessary for loading.

4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Load" button is enabled.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

5. Click the "Load" button.
   The load process is executed.

Or:

1. In the project tree, select the blocks that you want to upload.
2. Select the "Card Reader/USB memory > Write to memory card" command in the "Project" menu.
   The "Select memory card" dialog opens.
3. Select a memory card that is compatible with the CPU.
   A button with a green check mark is activated at the bottom of the dialog.
4. Click the button with the green check mark.
   If necessary, the project data is compiled. The "Load preview" dialog then opens. This dialog displays messages and recommends actions necessary for loading.
5. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as loading becomes possible, the "Load" button is enabled.
6. Click the "Load" button.
   The load process is executed.

Result

The selected blocks will be downloaded to the memory card. The messages under "Info > General" in the Inspector window report indicate whether the loading process was successful.

See also

Uploading blocks from a memory card (Page 7580)

19.4.7 Uploading blocks from a memory card (S7-300, S7-400)

You can only upload all blocks from a memory card to your project. Note that no symbolic information is loaded when you upload from a memory card.

Requirement

The memory card is displayed.

See also: Accessing memory cards

Procedure

To upload blocks from a memory card to your project, follow these steps:

1. In the project tree, drag the memory card folder to the device folder in the project. You can also copy the memory card and insert it in the device.
   The "Upload preview" dialog box opens. This dialog displays messages and recommends actions necessary for loading.
2. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as uploading becomes possible, the "Upload from device" button is enabled.
3. Click the "Upload from device" button.

See also

Downloading blocks to a memory card (Page 7579)

19.4.8 Switching off the sequencer prior to loading a GRAPH DB (S7-300, S7-400)

You can specify the switching off of the sequencer prior to loading an instance data block either globally or during the load operation.

Globally switching off the sequencer

To switch off the sequencer globally for each loading operation of an instance data block, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Select the "Turn off sequence before downloading DB" check box.
   For future loading operations, the sequencer is switched off prior to loading of the instance data block.

Switching off the sequencer during the loading operation

To switch off the sequencer during the loading operation, follow these steps:

1. Load the GRAPH function block to the device.
   During the loading operation, the "Load preview" dialog opens. This dialog displays alarms and suggests the required actions for loading. If the instance data block must be loaded together with the GRAPH function block, the "Load preview" dialog suggests the action "Turn off sequence before downloading the DB".
2. Select the "Turn off sequence before downloading DB" check box.

19.5 Downloading PLC tags (S7-1200, S7-1500)

19.5.1 Introduction to downloading PLC tags (S7-1200, S7-1500)

Introduction

PLC tags, as well as blocks, can be downloaded to a device and uploaded back to a project from a device. If you select a PLC tag table in the project navigation, the "Download to device > Software (only changes)" command becomes available.

You can use the command from the "Online" menu or from the shortcut menu or the "Load" button in the toolbar.

When you start the download to device from another object, for example from the block folder, the tag tables are always loaded consistently.

This functionality is supported by the following devices:

- CPU S7-1200
- CPU S7-1500

In the case of an S7-1500 as of firmware V2.5, the information to which PLC tag table the tags belong is downloaded to the CPU along with the PLC tags.

The command is not available for devices that do not support this functionality.

Rules for downloading to a device from PLC tag tables (S7-1500 as of V2.5 only)

Offline tag tables can be downloaded to the CPU in the specified structure.

The following rules apply:

- Downloading PLC tags is subject to the same requirements and procedures as downloading blocks.
- If inconsistencies are detected during loading of PLC tags, the synchronization dialog is displayed. To ensure consistency within a shared project, it is necessary to synchronize the changed data prior to loading so that nothing gets unintentionally overwritten. In this case, the data to be synchronized is displayed with the current status (offline/online comparison) and the actions that are possible. The comparison editor allows you to see which tags have changed in which tag table. The tags are always compared by name. In the event of a conflict, the user must decide which tags he would like to keep and load.

See also:

- Comparing PLC tags (Page 7616)
- Downloading blocks from the project tree to the device (Page 7554)
- Downloading blocks to a memory card (Page 7557)

Rules for uploading from a device for PLC tag tables (S7-1500 as of V2.5 only)

Tag tables that are available online can be uploaded to the project in the specified structure.

The following rules apply:

- Uploading PLC tags is subject to the same requirements and procedures as uploading blocks.
- For PLC tags, the following special considerations apply to uploading from devices:
  - Tags that exist online and offline with the same name are overwritten offline.
  - Tags that are only available offline remain unchanged.
  - Tags that are only available online are created offline again.
  - Tags that exist online in another tag table are inserted in the uploaded tag table and are deleted from the other tag table.

See also:

- Uploading blocks from device (Page 7556)
- Uploading blocks from a memory card (Page 7559)

Example for uploading from a device for PLC tag tables (S7-1500 as of V2.5 only)

The following example illustrates the system characteristics when uploading tag tables which differ online/offline:

- "Tag_A" tag is only available offline in tag table 1.
- "Tag_A" tag is only available online in tag table 2.

Result when uploading from tag table 1:

- "Tag_A" is only present offline in tag table 1 and therefore remains unchanged in tag table 1 after uploading.

Result when uploading from tag table 2:

- "Tag_A" is only available online in tag table 2 and is therefore added to tag table 2 after the upload; it is deleted in tag table 1.

See also

Comparing PLC tags (Page 7616)

Downloading blocks from the project tree to the device (Page 7554)

Uploading blocks from device (Page 7556)

Uploading blocks from a memory card (Page 7559)

Downloading blocks to a memory card (Page 7557)

Online/offline visibility of PLC tags (Page 7584)

19.5.2 Online/offline visibility of PLC tags (S7-1200, S7-1500)

Introduction

PLC tag tables, as well as blocks, can be downloaded to a device and uploaded back to a project from a device.

Depending on whether tag tables are only available offline or online, they are displayed differently in the TIA Portal.

Visibility of PLC tag tables (S7-1500 as of V2.5 only)

PLC tag tables are displayed in the TIA Portal as follows:

Display of PLC tag tables	Visibility	Possible actions
Offline view in the project tree	Offline data is permanently displayed, if available. After uploading from the device, the online data is also visible offline. However, there may still be differences because only tags that are available offline are retained after the upload from device.	It is possible to open and edit in the editor. It is possible to download and upload to and from the device as well as to and from memory cards.
Online view in the project tree	Online data is displayed in online mode. Online data is also displayed offline if there is no corresponding offline tag table.	The editor is opened in read-only mode, editing is not possible. However, you can determine which tags are available.
For accessible devices	After downloading to device.	The editor is opened in read-only mode, editing is not possible. However, you can determine which tags are available.
On local memory cards	After downloading to the memory card.	The editor is opened in read-only mode, editing is not possible. However, you can determine which tags are available.
In the comparison editor	If there is one or more tag tables offline and/or online, comparative operations can be performed.	You can open the comparison editor and the comparison results for the selected tags are displayed by name.

You can find additional information about how you can use the comparison editor for comparing PLC tags here: Comparing PLC tags (Page 7616)

See also

Comparing PLC tags (Page 7616)

Introduction to downloading PLC tags (Page 7582)

Comparing PLC programs 20

20.1 Basic information on comparing PLC programs

20.1.1 Introduction to comparing PLC programs

Function

You can compare the following objects of a PLC program in order to detect any differences:

- Code blocks with other code blocks
- Data blocks with other data blocks
- PLC tags of a PLC tag table with the PLC tags of another PLC tag table
- PLC data types with other PLC data types

Comparison of tags takes place over tag names.

Types and levels of comparison

Two different basic types of comparison can be used:

- Compare offline/online
  The objects in the project are compared with the objects of the corresponding online device. An online connection to the device is necessary for this comparison.
- Offline/offline comparison
  With offline/offline comparison you can compare the objects of two devices in the project that is currently open. A device from a reference project or from a library can also be dragged onto the right drop area. You can only compare devices within a TIA Portal instance.

Please note that you cannot carry out an unlimited number of comparisons at the same time, but only one comparison per comparison type (offline/online or offline/offline).

You can choose between the following levels of comparison depending on how in-depth an object comparison you require:

- Compare editor
  If you start a comparison on a device, the compare editor opens in which the result of the comparison is displayed.
- Detailed comparison
  For some comparison objects, you can then start a detailed comparison in the compare editor in which the objects compared will be opened side-by-side, each in its own program editor instance. Any differences will be highlighted. For blocks, PLC tags and PLC data types, you can also start the detailed comparison directly from the project tree. Note, however, that you can only start a detailed comparison for SCL blocks if the type of block interface is identical for both blocks.

The table below gives an overview of the types and levels of comparison you can apply for each object:

Object	Offline/online		Offline/offline	
Compare editor	Detailed comparison	Compare editor	Detailed comparison	
LAD block	X	X	X	X
FBD block	X	X	X	X
STL block[1]	X	X	X	X
SCL block	X	X[3 4 7]	X	X[7]
GRAPH block[2]	X	X[4]	X	X[5]
Global data block	X	X	X	X
Instance data block	X	X	X	X
PLC tags	X[6]	X[6]	X	X
PLC data type	X[4]	X[4]	X	X

Legend:

X: Available
-: Not available

[1]: STL is not available for S7-1200

[2]: GRAPH is not available for S7-1200

[3]: Not for S7-1200 older than version 2.0

[4]: Not for S7-300/400

[5]: Only comparable with a GRAPH block from the same CPU family

[6]: Only for S7-1500 as of version 2.5

[7]: The type of block interface must be identical

Note

Please note the following:

- You cannot perform a detailed comparison for know-how protected blocks.
- If the detail comparison detects differences only with respect to the data types of local tags, with offline being an interrupt data type (C_ALARM C_ALARM_S C_ALARM_8 C_ALARM_8P C_ALARM_T C_AR_SEND C_NOTIFY C_NOTIFY_8P) and online a DWORD, this difference is not marked as such.
- You cannot run a detailed comparison for types and master copies from libraries.

Comparing PLC programs based on checksums

In addition to the comparison options listed above, TIA Portal gives provides you the option of a simple checksum comparison: The system automatically determines a checksum for the entire PLC program. Based on this checksum, you can easily determine whether two PLC programs are identical.

See also: Comparing PLC programs based on checksums (Page 7588)

See also

Comparing blocks and PLC data types (Page 7587)

Comparing PLC tags (Page 7588)

Performing detailed block comparisons (Page 7591)

20.1.2 Comparing blocks and PLC data types

Assignment criteria

The blocks and PLC data blocks to be compared are assigned to each other on the basis of the following criteria:

- Blocks:
  - Compare offline/online: Addresses, e.g. FB100 or DB100
  - Offline/offline comparison: Symbolic names of the blocks
- PLC data types: PLC data types are assigned to each other by name

Comparison

When comparing blocks and PLC data types, checksums are formed for specific properties that serve as the basis for comparison. Some of these properties are grouped into categories in the Comparison Editor. No checksums are available with an offline/online comparison for CPUs of the S7-300/400 series; the time stamp is evaluated in this case. Basically, the time stamp is issued as information for an offline/online comparison for all CPUs as a separate category.

After performing the comparison, you can use actions to define what is to be done about the differences. You can also start a detailed comparison for blocks. The versions of a block compared are opened beside each other and the differences are highlighted. For blocks, PLC tags and PLC data types, you can also start the detailed comparison directly from the project tree.

See also

Introduction to comparing PLC programs (Page 7585)

Comparing PLC tags (Page 7588)

Performing detailed block comparisons (Page 7591)

20.1.3 Comparing PLC tags

Introduction

The device PLC tag tables will also be shown in the comparison editor if you carry out an offline/offline comparison. The PLC tag tables are assigned to each by name and you can see at a glance whether the PLC tag tables are available in both devices. If necessary, you can also perform a detailed comparison for a PLC tag table. For blocks, PLC tag tables and PLC data types, you can also start the detailed comparison directly from the project tree.

With an offline/online comparison, the overall status of all PLC tags is displayed for CPUs of the S7-1200 and S7-1500 series with a firmware version older than V2.5. For S7-1500-CPUs as of firmware version 2.5 and higher, you have the same options as with an offline/online comparison.

See also

Introduction to comparing PLC programs (Page 7585)

Comparing blocks and PLC data types (Page 7587)

Performing detailed block comparisons (Page 7591)

20.1.4 Comparison of PLC programs based on checksums

Introduction

PLC programs are automatically identified with unique checksums during compilation. Based on this checksum, you can identify your program and determine whether two PLC programs are identical.

Because the checksum is loaded to the CPU together with the PLC program, it can also serve as important information in case of a service call. You can, for example, easily determine whether the program that is currently running on the CPU is the same program you loaded some time ago or if it has been changed in the meantime. The original program does not need to be installed on your programming device (PG) in this case.

Creation of the system diagnostics blocks

PLC programs are automatically given a unique checksum during compilation. If the next compilation run detects that the PLC program has been changed, the program is given a new checksum. If the PLC program has not changed but is still compiled again, the checksum remains the same.

The checksum also remains unchanged when you make changes and undo them.

The check takes place every time the PLC program is compiled, for example, when you select a CPU or a block folder and use the menu command "Compile > Software".

Two checksums are generated:

- Checksum for software: A checksum is generated across all blocks in the block folder.
  - Blocks that are not called in the program but exist in the block folder are also included in the checksum.
  - Safety blocks and safety PLC data types (UDT) are not taken into consideration.
  - Changes to watch tables or force tables do not change the checksum.
  - Data blocks that are created or modified during runtime, for example, by the instructions "WRIT_DBL", "CREAT_DB" and "DELETE_DB", do not influence the formation of the checksum.
- Checksum for text lists: A checksum is generated for all text lists in the project.

Loading the checksum

The checksum is loaded to the CPU together with the PLC program and is available in the online program. During download from the CPU, the checksum is not applied to the offline project because it is generated again automatically during the next compilation.

Evaluating the checksum

The checksum is displayed in the CPU properties (Properties > General > Checksums). From there, you can apply it manually to your documents. The "GetChecksum" extended instruction is available to read out the checksum while the program is running.

See also

GetChecksum: Read out checksum (Page 2606)

20.2 Comparing blocks

20.2.1 Comparing blocks in the compare editor

You have the following options for comparing blocks in the compare editor:

- Compare offline/online
  The blocks in the project are compared with the blocks of the selected device.
- Automatic offline/offline comparison
  All blocks of the selected devices are compared offline.
- Manual offline/offline comparison
  The selected blocks of the devices are compared offline.

Performing an offline/online comparison of blocks

To perform an offline/online comparison, follow these steps:

1. In the project tree, select a device that allows offline/online comparison.
2. Select the "Compare > Offline/online" command in the shortcut menu.
3. If you have not already established an online connection to this device, the "Go online" dialog opens. In this case, set all the necessary parameters for the connection and click "Connect".
   The online connection is established and the compare editor opens.
4. Open the "Program blocks" folder.
   You can identify the status based on the symbols in the status and action area. You can define certain actions depending on the status of the objects. Note, however, that you can only perform actions in one direction in a synchronization action.

Performing an automatic offline/offline comparison of blocks

To perform an automatic offline/offline comparison of blocks, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. Open the "Program blocks" folder.
   You can identify the status of the objects based on the symbols in the status and action area. You can define certain actions depending on the status of the objects. When you select an object, the object's properties and the corresponding object of the assigned device are clearly shown in the properties comparison.

You can drag any other device to the drop area at any time to perform further comparisons.

Carrying out a manual offline/offline comparison of blocks

To perform a manual offline/offline comparison of blocks, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. In the status and action area, click on the button for switching between automatic and manual comparison.
5. Select the objects that you want to compare.
   The properties comparison is displayed. You can identify the status of the objects based on the symbols. You can define certain actions depending on the status of the objects.

You can drag any other device to the drop area at any time to perform further comparisons.

See also

Introduction to comparing PLC programs (Page 7585)

Performing detailed block comparisons (Page 7591)

Comparing PLC tags (Page 7616)

Comparing PLC data types (Page 7619)

20.2.2 Performing detailed block comparisons

20.2.2.1 Start a detailed comparison for LAD/FBD/STL/SCL blocks

You can start a detailed comparison for blocks. The versions of a block compared are opened beside each other and the differences highlighted. You have the following options for starting a detailed comparison for blocks:

- Starting detailed comparisons using the compare editor
  The detailed comparison for blocks is possible with the compare editor, both for offline/online and for offline/offline comparisons.
- Starting detailed comparison in the project tree
  The detailed comparison for blocks is possible with the project tree, both for offline/online and for offline/offline comparisons. An existing online connection is required for the offline/online comparison.
- Starting detailed comparisons in the program editor
  In this case the opened block is compared with its online version.

Note

Please note the following:

- For blocks that are created in the programming language SCL, the detail comparison is not available for S7-1200 series CPUs with a version older than 2.0.
- You can only start a detailed comparison for two SCL blocks if both blocks use the same interface type, either both in table form or both in text form.
- S7-1500: It is possible that another user can carry out a loading process on the CPU through joint parallel working on a CPU. If the currently compared block is changed or deleted by this loading process, the detailed comparison is terminated and a message is displayed. In this case, re-start the detailed comparison, if required.

Starting detailed comparisons using the compare editor

To start a detailed comparison for a block using the compare editor, follow these steps:

1. First, perform an offline/online or an offline/offline comparison.
   The compare editor opens.
2. In the compare editor, select the block for which you want to perform a detailed comparison.
3. Click "Start detailed comparison" in the toolbar or select the "Start detailed comparison" from the shortcut menu.

Starting detailed comparison in the project tree

To start an offline/offline detailed comparison for a block directly in the project tree, follow these steps:

1. Right-click the block that you want to compare. This can also be a block from a reference project.
2. Select the command "Quick compare > Select as left object" in the shortcut menu.
3. Right-click the block that you want to compare with the block that you previously selected as left object.
4. Select the command "Quick compare > Compare with <selected object>" in the shortcut menu. "<selected object>" stands for the left comparison object.

To start an offline/online detailed comparison for a block directly in the project tree, follow these steps:

1. Establish an online connection to the device where the block is located.
2. Right-click the block that you want to compare with its online object.
3. Select the command "Quick compare > Compare with the online object" in the shortcut menu.

Starting detailed comparisons in the program editor

For the comparison type offline/online, you can start the detailed comparison directly in the program editor. To do this, follow these steps:

1. Open the block for which you wish to carry out a detailed comparison.
2. Establish an online connection.
   See also: Go online and Go offline

   Note

   Please note that the block must be available online in order for you to be able to start the detailed comparison for the block within the programming editor.

3. Click "Detailed comparison" in the toolbar.
4. Confirm the dialog for closing the block with "Yes".

Result

One instance of the program editor will be opened for each version of the block compared and the two instances are displayed side by side.

See also

Basic information on comparing PLC programs (Page 7585)

Visualization of the comparison result for LAD/FBD (Page 7596)

Visualization of the comparison result for STL (Page 7600)

Visualization of the comparison result for SCL (Page 7603)

Navigating in the detailed comparison (Page 7612)

Changing blocks during the detailed comparison (Page 7613)

Updating comparison results (Page 7615)

20.2.2.2 Start detailed comparison for GRAPH blocks (S7-300, S7-400, S7-1500)

You can start a detailed comparison for GRAPH blocks. The versions of a block compared are opened beside each other and the differences highlighted. You have the following options for starting a detailed comparison for GRAPH blocks:

- Starting detailed comparisons using the compare editor
  The detailed comparison for blocks is possible with the compare editor, both for online/offline and for offline/offline comparisons.
- Starting detailed comparison in the project tree
  The detailed comparison for blocks is possible with the project tree, both for online/offline and for offline/offline comparisons. An existing online connection is required for the online/offline comparison.
- Starting detailed comparisons in the program editor
  In this case the opened block is compared with its online version.

The following comparison methods are available for GRAPH blocks:

- Compare sequence
  Complete sequencers are compared with each other in this mode. A detailed comparison for a GRAPH block always starts in the "Compare sequence" comparison mode. This means the comparison starts at the beginning of the sequencer and the differences between the sequencers are displayed. If there are structural differences between the sequencers, the comparison results are only displayed up to the first structural difference.
- Compare selection
  You can use the "Compare selection" comparison mode to compare any areas with each other. This means you can compare sections of the sequencer which are downstream from a structural difference in the sequence.

You can toggle at random between both comparison modes.

Note

S7-1500: It is possible that another user can carry out a loading process on the CPU through joint parallel working on a CPU. If the currently compared block is changed or deleted by this loading process, the detailed comparison is terminated and a message is displayed. In this case, re-start the detailed comparison, if required.

Starting detailed comparisons using the compare editor

To start a detailed comparison, follow these steps:

1. First, perform an online/offline or an offline/offline comparison between the two devices. The compare editor opens.
2. In the compare editor, select the GRAPH block for which you want to perform a detailed comparison.

   Note

   The manual comparison is also available for an offline/offline comparison. It lets you select any GRAPH block in the comparison editor for the comparison.

3. Click the "Start detailed comparison" button in the toolbar.
   One instance of the program editor is opened for each version of the GRAPH blocks compared and the two instances are displayed side by side. Any differences will be highlighted. The comparison takes place in the "Compare sequence" mode.

Starting detailed comparison in the project tree

To start an offline/offline detailed comparison for a GRAPH block directly in the project tree, follow these steps:

1. Right-click the block that you want to compare. This can also be a block from a reference project.
2. Select the command "Quick compare > Select as left object" in the shortcut menu.

3. Right-click the block that you want to compare with the block that you previously selected as left object.
4. Select the command "Quick compare > Compare with <selected object>" in the shortcut menu. "<selected object>" stands for the left comparison object.
   One instance of the program editor is opened for each version of the GRAPH blocks compared and the two instances are displayed side by side. Any differences will be highlighted. The comparison takes place in the "Compare sequence" mode.

To start an online/offline detailed comparison for a GRAPH block directly in the project tree, follow these steps:

1. Establish an online connection to the device where the block is located.
2. Right-click the block that you want to compare with its online object.
3. Select the command "Quick compare > Compare with the online object" in the shortcut menu.
   One instance of the program editor is opened for each version of the GRAPH blocks compared and the two instances are displayed side by side. Any differences will be highlighted. The comparison takes place in the "Compare sequence" mode.

Starting detailed comparisons in the program editor

For the comparison type online/offline, you can start the detailed comparison directly in the program editor. To do this, follow these steps:

1. Open the GRAPH block for which you wish to carry out a detailed comparison.
2. Establish an online connection.
   See also: Go online and Go offline

   Note

   Please note that the block must be available online in order for you to be able to start the detailed comparison for the block within the programming editor.

3. Click "Detailed comparison" in the toolbar.
4. Confirm the dialog for closing the block with "Yes".
   One instance of the program editor is opened for each version of the GRAPH blocks compared and the two instances are displayed side by side. Any differences will be highlighted. The comparison takes place in the "Compare sequence" mode.

Comparing regions

To compare specific areas within the displayed sequencers, follow these steps:

1. In each sequencer, select the step from which you want to start the comparison.
2. Click "Comparison mode" in the toolbar.
   The comparison mode changes to "Compare selection" and the comparison result is updated. The sequencers are compared as of the selected steps. The compared substrings are highlighted.
3. To compare other areas, select the steps which are to serve as starting point for the comparison.

4. Click "Update the comparison result" in the toolbar.

   Note

   When you change the comparison mode, you are also updating the comparison results at the same time. If you stay in "Compare selection" mode, you must manually update the comparison results every time you change the starting point.

5. To compare the complete sequencers once again, click "Comparison mode" in the toolbar. Each time you click "Comparison mode" you are changing the comparison mode. You can also open the drop-down list with the arrow and select the required mode from the list.

See also

Basic information on comparing PLC programs (Page 7585)

Visualization of the comparison result for GRAPH (Page 7606)

Navigating in the detailed comparison (Page 7612)

Changing blocks during the detailed comparison (Page 7613)

Updating comparison results (Page 7615)

20.2.2.3 Visualization of the comparison result

Visualization of the comparison result for LAD/FBD

Introduction

The detailed comparison allows you to identify the exact places where versions of a block differ. The following color coding allows you to find these places as quickly as possible:

- The lines where there are differences are highlighted in gray.
- Differing operands and instructions are highlighted in green.
- If the number of networks differs, pseudo-networks are added to allow the display of identical networks to be synchronized. These pseudo-networks are highlighted in gray and contain the text "No corresponding network found" in the title bar of the network. Pseudo-networks cannot be edited.

  Note

  If there is a difference of more than 50 percent between two networks, these networks are regarded as not belonging together and there is no comparison between these networks. In this case, pseudo networks are also inserted until a suitable network is found.

For the sake of clarity, not all the differences are highlighted but rather the first difference of an operation in each case. For example, if all the inputs in an instruction with multiple inputs are different in the offline and online versions of the block, only the first input is highlighted as

a difference. You can resolve this difference and update the comparison list. The next input will then be highlighted as a difference.

The number of differences highlighted within a network therefore depends on the number of instructions.

Structure of the detailed comparison

The following figure shows an example of the offline/online detailed comparison for the LAD programming language:

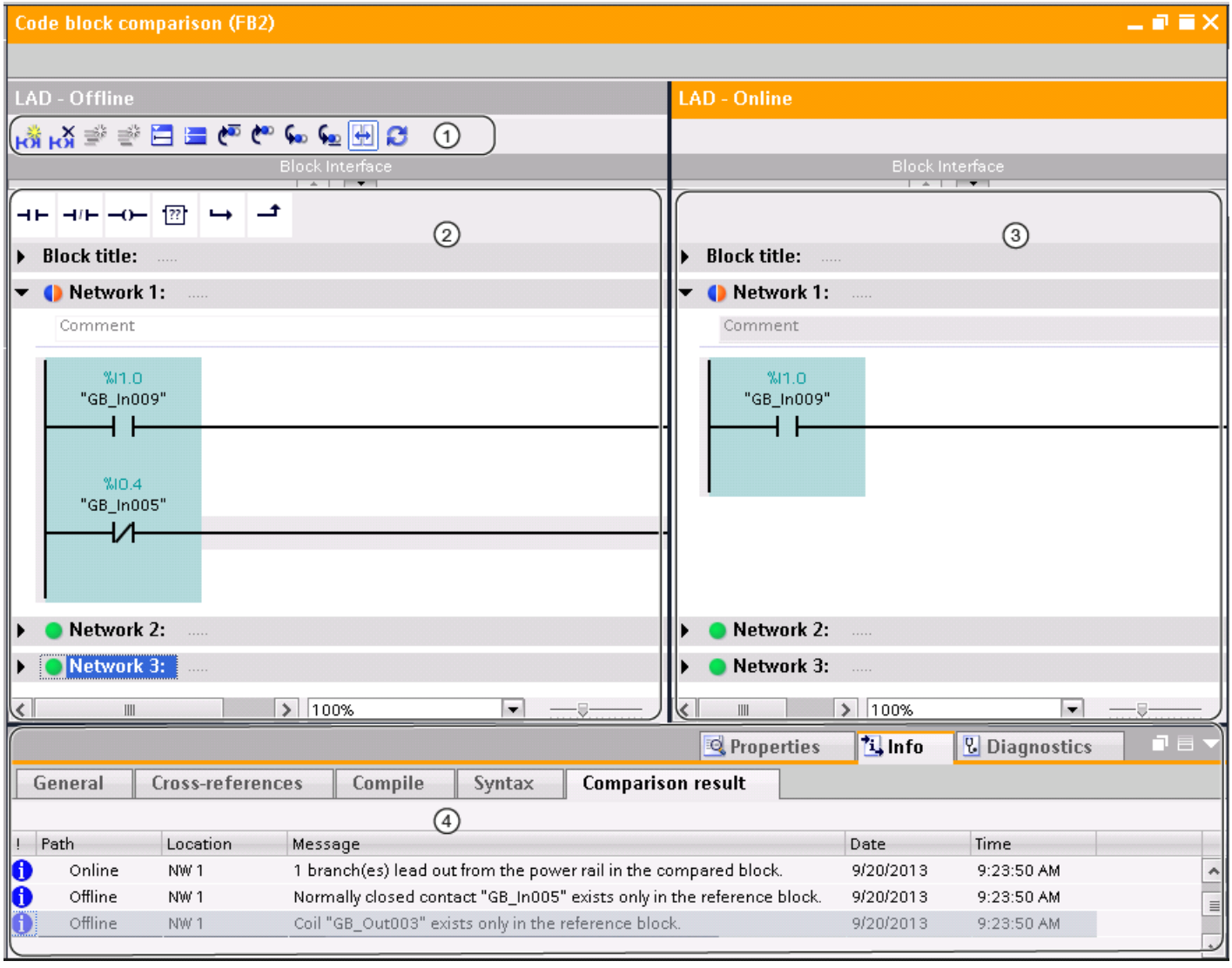

① Toolbar of the detailed comparison for LAD

② Reference block

③ Compared block

④ Comparison result in the Inspector window

The following figure shows an example of the offline/online detailed comparison for the FBD programming language:

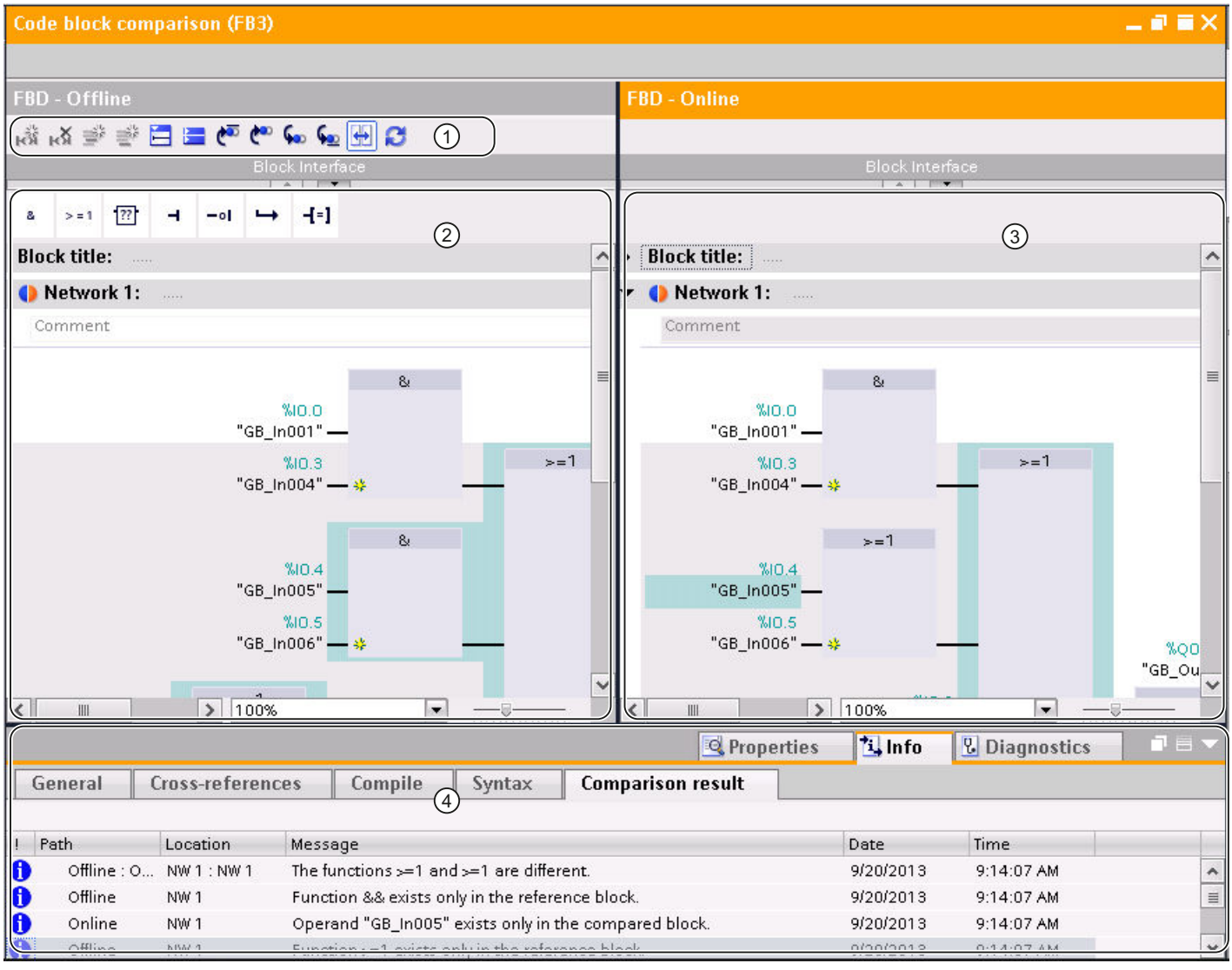

① Toolbar of the detailed comparison for FBD

② Reference block

③ Compared block

④ Comparison result in the Inspector window

Note

Display of the symbolic names of the online version of the block is only possible for S7-1200 and S7-1500.

Toolbar of the detailed comparison

With the toolbar, you can access the following functions:

- General functions
  - Insert network
  - Delete network
  - Insert row
  - Add row
  - Open all networks
  - Close all networks
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

Reference block

The reference block is displayed in the left window. In an offline/online comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an offline/online comparison, the compared block is the online version of the block.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Basic information on comparing PLC programs (Page 7585)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 7591)

Navigating in the detailed comparison (Page 7612)

Changing blocks during the detailed comparison (Page 7613)

Updating comparison results (Page 7615)

Visualization of the comparison result for STL (S7-300, S7-400, S7-1500)

Introduction

The detailed comparison allows you to identify the exact places where versions of a block differ. The following color coding allows you to find these places as quickly as possible:

- The lines where there are differences are highlighted in gray.
- Differing operands and instructions are highlighted in green.
- If the number of networks differs, pseudo-networks are added to allow the display of identical networks to be synchronized. These pseudo-networks are highlighted in gray and contain the text "No corresponding network found" in the title bar of the network. Pseudo-networks cannot be edited.

Structure of the detailed comparison

The following figure shows an example of the online/offline detailed comparison for the STL programming language:

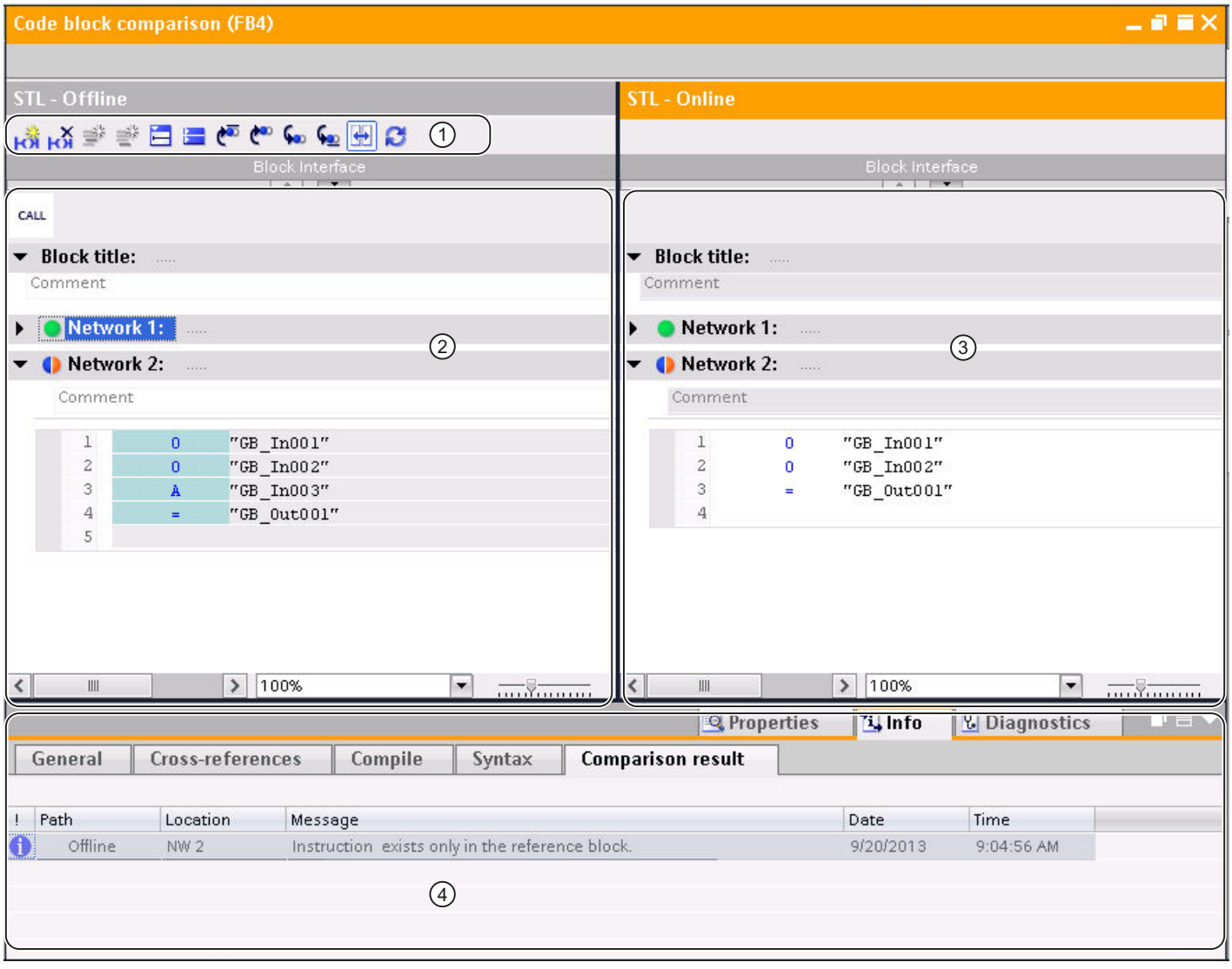

① Toolbar of the detailed comparison for STL

② Reference block

③ Compared block

④ Comparison result in the Inspector window

Note

The display of the symbolic names of the online version of the block is only possible for S7-1500.

Toolbar of the detailed comparison

With the toolbar, you can access the following functions:

- General functions
  - Insert network
  - Delete network
  - Insert row
  - Add row
  - Open all networks
  - Close all networks
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

Reference block

The reference block is displayed in the left window. In an online/offline comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an online/offline comparison, the compared block is the online version of the block.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Basic information on comparing PLC programs (Page 7585)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 7591)

Navigating in the detailed comparison (Page 7612)

Changing blocks during the detailed comparison (Page 7613)

Updating comparison results (Page 7615)

Visualization of the comparison result for SCL

Introduction

The detailed comparison allows you to identify the exact places where versions of a block differ. The following color coding allows you to find these places as quickly as possible:

- The lines where there are differences are highlighted in gray.
- Differing operands and instructions are highlighted in green.

Note

The offline/online detailed comparison is not available for the CPU families S7-300/400 and for S7-1200 with a version less than 2.0.

Structure of the detailed comparison

The following figure shows an example of the offline/online detailed comparison for the SCL programming language:

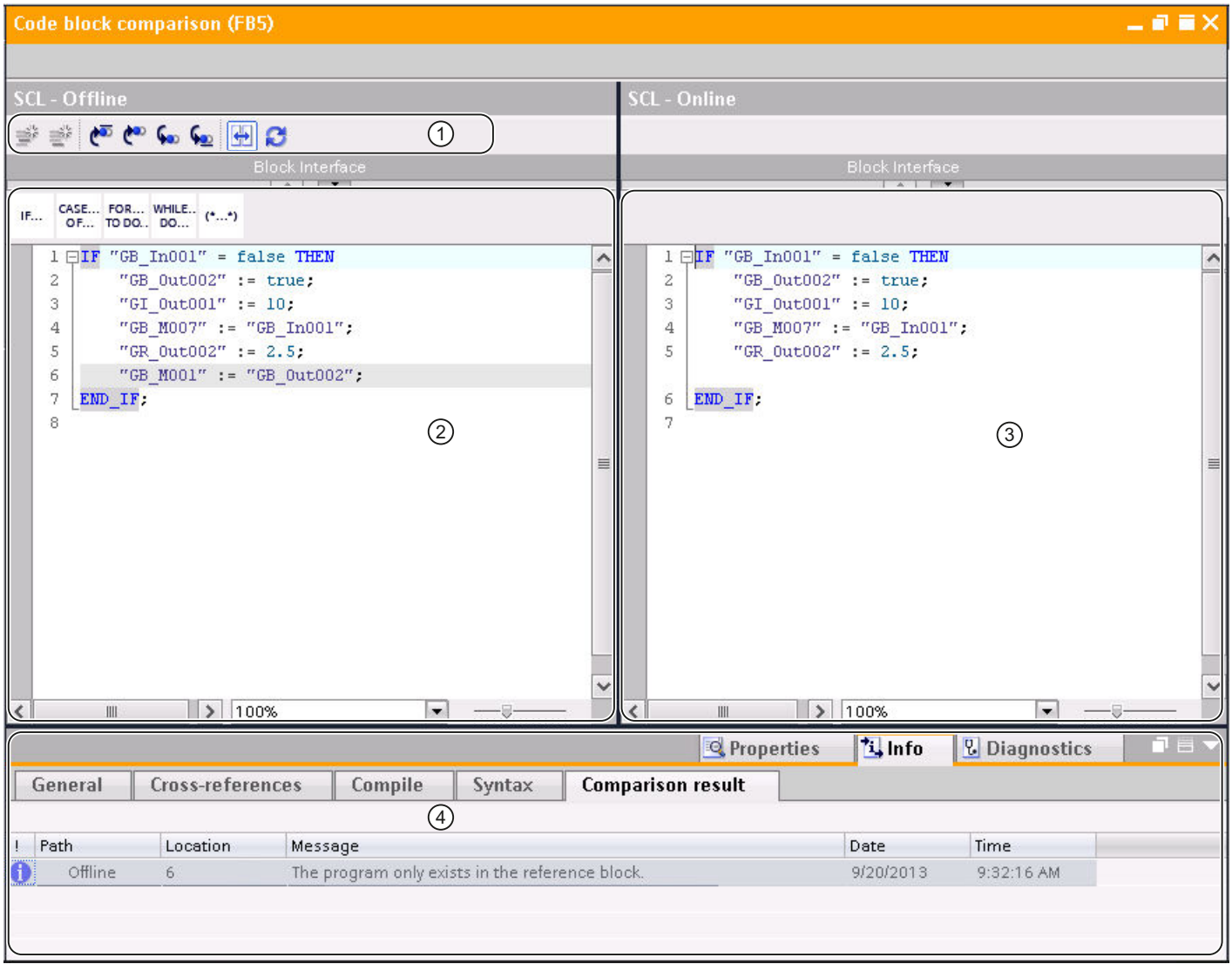

① Toolbar of the detailed comparison for SCL

② Reference block

③ Compared block

④ Comparison result in the Inspector window

Note

The display of the symbolic name of the online version of the block is only possible for S7-1200 and S7-1500.

Toolbar of the detailed comparison

With the toolbar, you can access the following functions:

- General functions
  - Insert row
  - Add row
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

Reference block

The reference block is displayed in the left window. In an offline/online comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an offline/online comparison, the compared block is the online version of the block.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Basic information on comparing PLC programs (Page 7585)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 7591)

Navigating in the detailed comparison (Page 7612)

Changing blocks during the detailed comparison (Page 7613)

Updating comparison results (Page 7615)

Visualization of the comparison result for GRAPH (S7-300, S7-400, S7-1500)

Introduction

The detailed comparison allows you to identify the exact places where versions of a block differ. When you start a detailed comparison for a GRAPH block, navigation is opened first. Use the dividers to toggle between the navigation and the currently set view. You can select other views with the toolbar of the detailed comparison.

The result of the comparison is indicated by the comparison symbols.

See also: Overview of the comparison editor

Structure of the detailed comparison

The following figure shows an example for the navigation view with an online/offline detailed comparison for the GRAPH programming language:

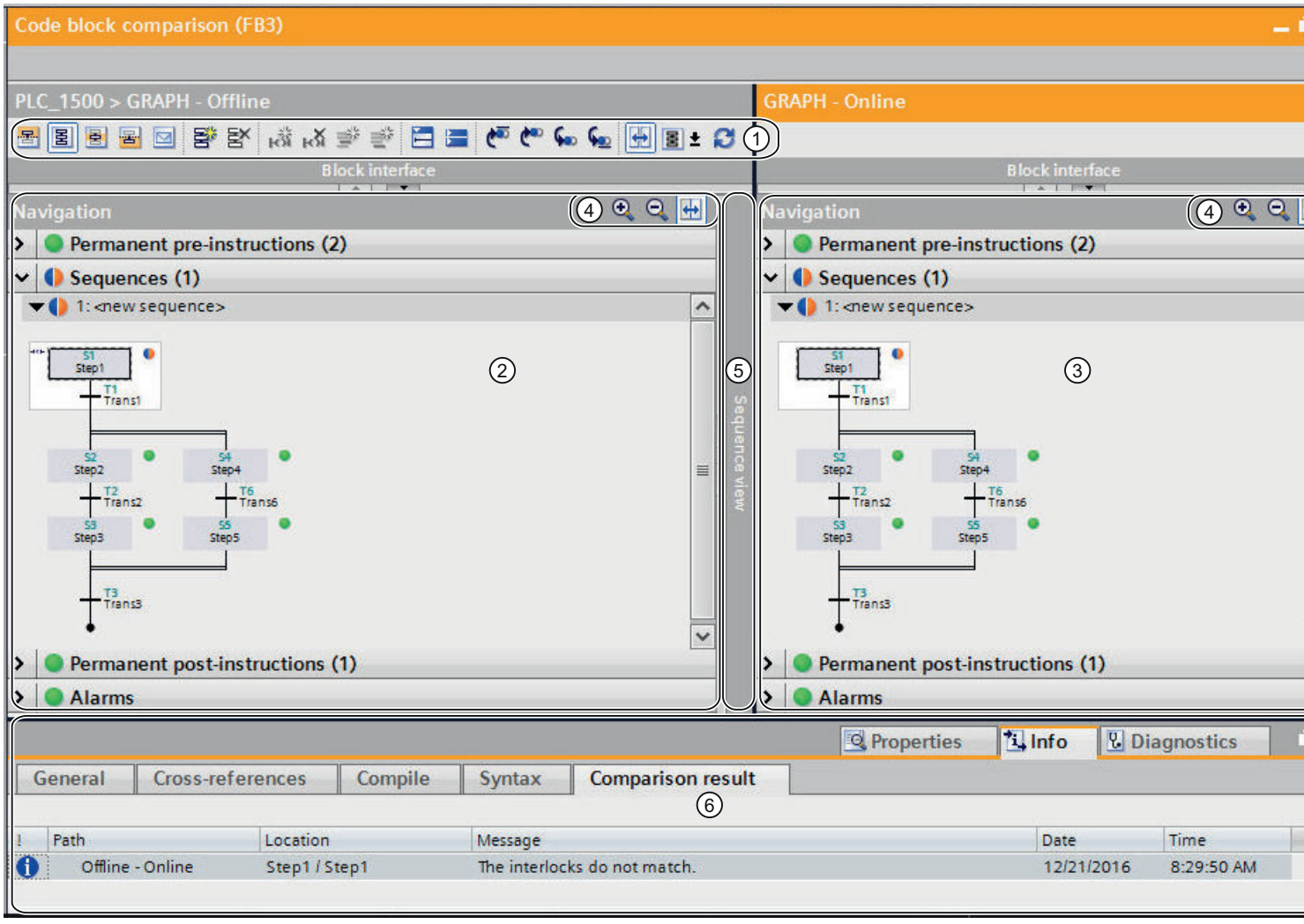

① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

④ Navigation toolbar

⑤ Dividers

⑥ Comparison result in the Inspector window

The following figure shows an example for the sequence view with an online/offline detailed comparison for the GRAPH programming language:

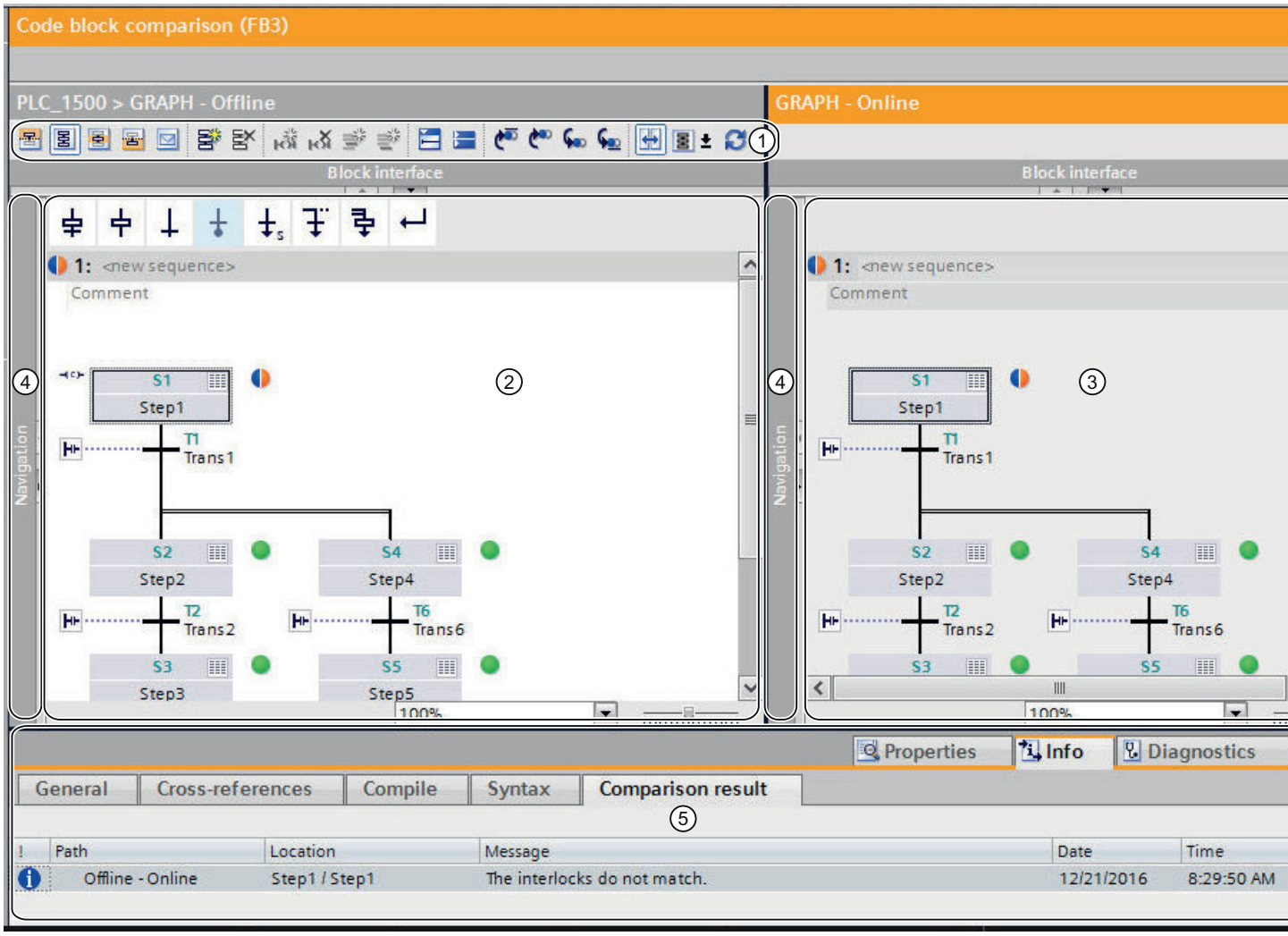

① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

④ Dividers

⑤ Comparison result in the Inspector window

Note

If there are structural differences between the blocks, the comparison results are only displayed up to the first structural difference in the sequence view.

The following figure shows an example for the single step view with an online/offline detailed comparison for the GRAPH programming language:

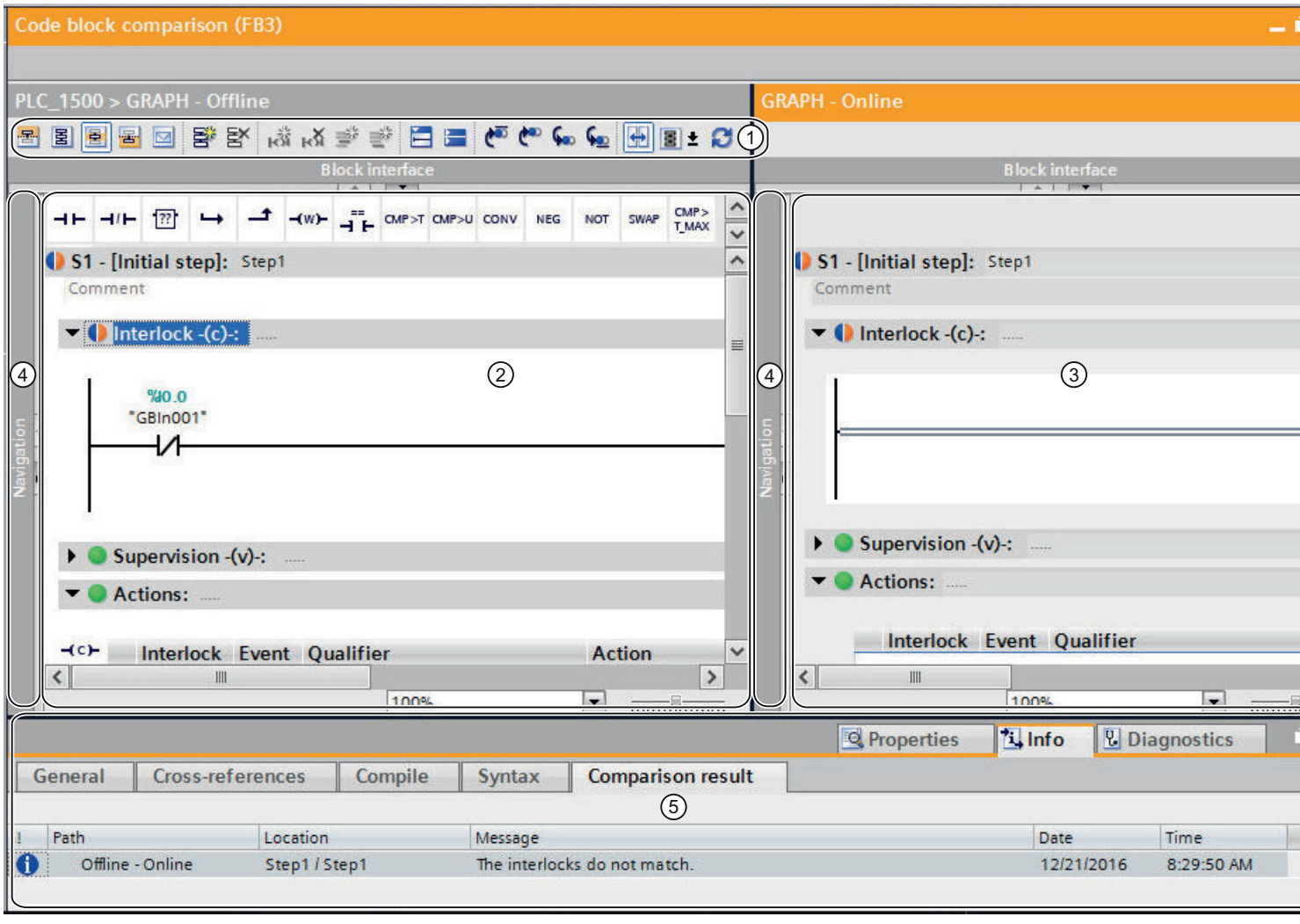

① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

④ Dividers

⑤ Comparison result in the Inspector window

Note

The result of the comparison refers to the complete network Differences within the networks are not identified.

The following figure shows an example for the view of permanent instructions with an online/offline detailed comparison for the GRAPH programming language:

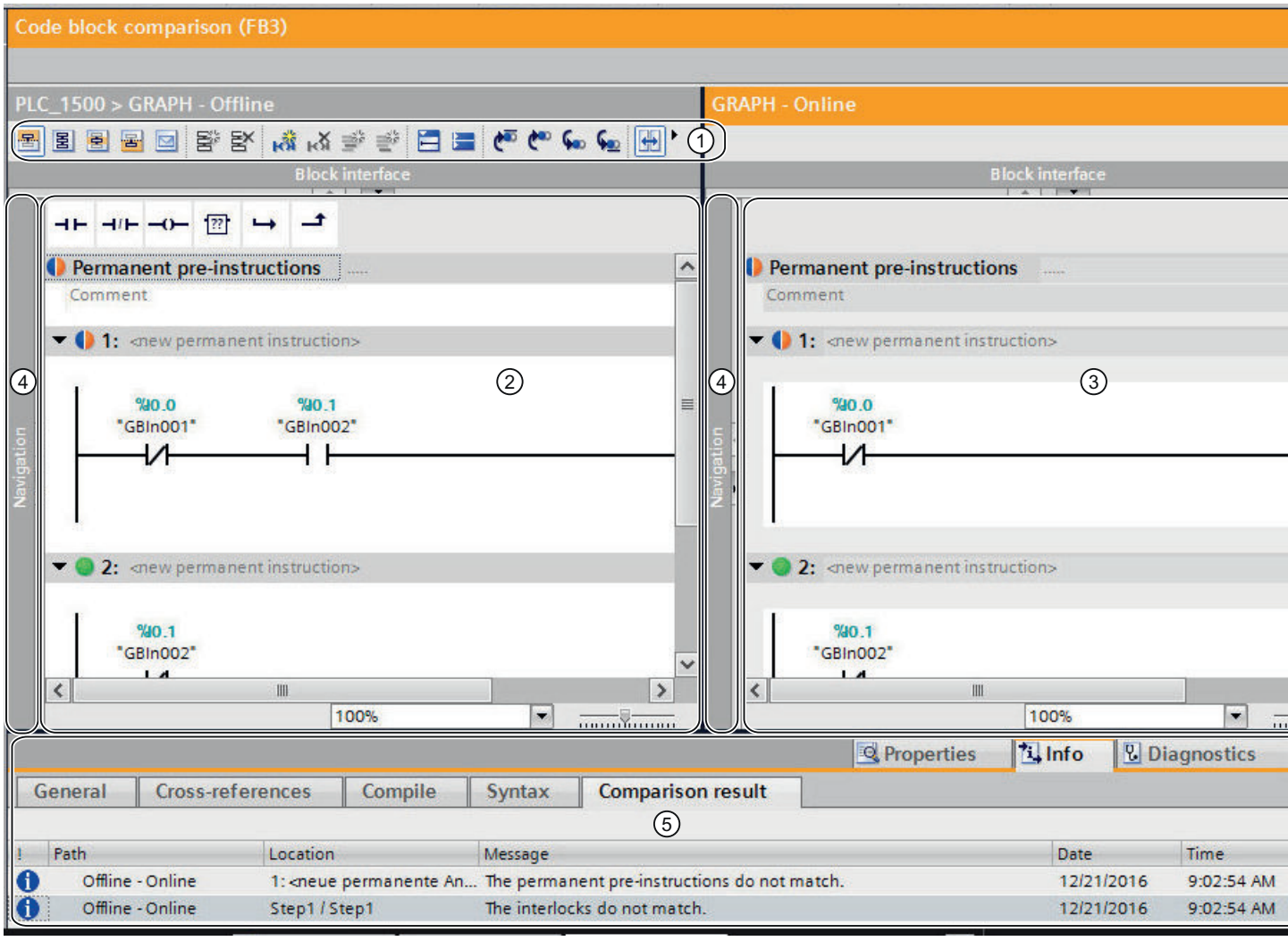

① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

④ Dividers

⑤ Comparison result in the Inspector window

Note

The result of the comparison refers to the complete network Differences within the networks are not identified.

The following figure shows an example for the alarm view with an online/offline detailed comparison for the GRAPH programming language:

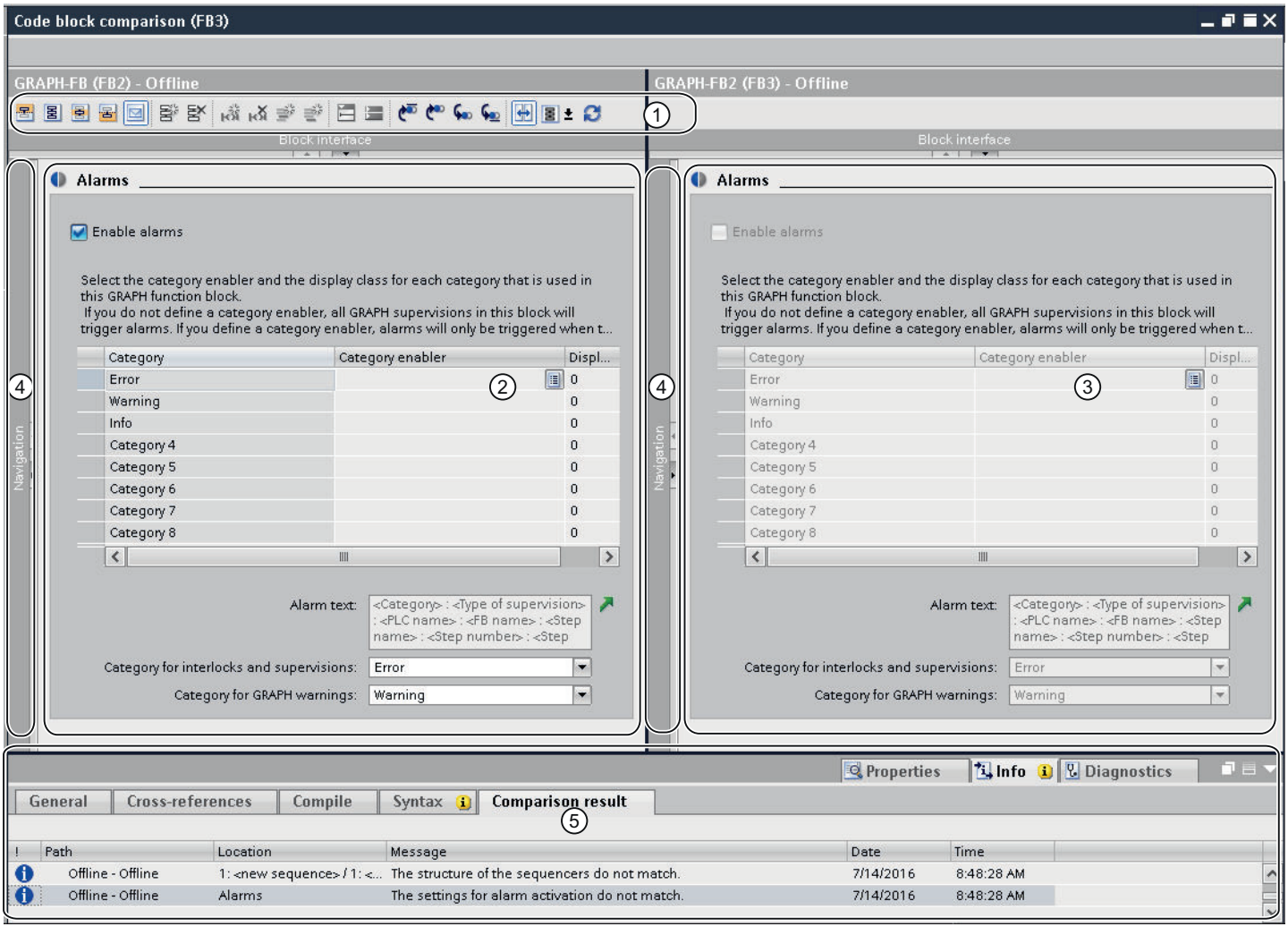

① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

④ Dividers

⑤ Comparison result in the Inspector window

Toolbars

With the toolbar of the detailed comparison, you can access the following functions:

- General functions
  - Change to permanent pre-instructions
  - Change to sequence view
  - Change to single step view
  - Change to permanent post-instructions
  - Change to alarm view
  - Add sequence
  - Delete sequence
  - Insert network
  - Delete network
  - Insert row
  - Add row
  - Open all networks
  - Close all networks
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results
  - Select the comparison mode

The navigation has its own toolbar with the following functions:

- Zoom in or zoom out of elements within the navigation
- Synchronize navigation

Reference block

The reference block is displayed in the left window. In an online/offline comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an online/offline comparison, the compared block is the online version of the block.

Dividers

You can click the dividers to toggle quickly between the navigation and the current view.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Basic information on comparing PLC programs (Page 7585)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 7591)

Start detailed comparison for GRAPH blocks (Page 7593)

Navigating in the detailed comparison (Page 7612)

Changing blocks during the detailed comparison (Page 7613)

Updating comparison results (Page 7615)

20.2.2.4 Navigating in the detailed comparison

Requirement

You have run a detailed comparison.

Navigate to the differences

To navigate to a difference between the two blocks, follow these steps:

1. Open the list of results for the detailed comparison under "Info > Comparison result" in the Inspector window.
2. Double-click a difference.
   The difference is selected in both editors.

Or:

1. Click one of the following navigation buttons on the toolbar:
   - Position on first difference
     Navigates to the first difference in the block, and displays the difference in both editors.
   - Position on previous difference
     Navigates to the previous difference starting from the current position, and displays the difference in both editors.
   - Position on next difference
     Navigates to the next difference starting from the current position, and displays the difference in both editors.
   - Position on last difference
     Navigates to the last difference in the block, and displays the difference in both editors.

Switching off/on the synchronization of the vertical scrolling between the editors

The scrolling for both editors is synchronized to ensure that the corresponding networks are visible parallel to each other during vertical scrolling. You can switch this mode off and on. To do this, follow these steps:

1. To switch off synchronized scrolling, click the "Synchronize scrolling between editors" button in the toolbar.
2. To switch on synchronized scrolling again, click the "Synchronize scrolling between editors" button one more time in the toolbar.

See also

Basic information on comparing PLC programs (Page 7585)

Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 7591)

Start detailed comparison for GRAPH blocks (Page 7593)

Visualization of the comparison result for LAD/FBD (Page 7596)

Visualization of the comparison result for STL (Page 7600)

Visualization of the comparison result for SCL (Page 7603)

Visualization of the comparison result for GRAPH (Page 7606)

Changing blocks during the detailed comparison (Page 7613)

Updating comparison results (Page 7615)

20.2.2.5 Changing blocks during the detailed comparison

While you are carrying out the detailed comparison, you can make changes to the blocks that are being compared. Remember the following:

- Compare offline/online: You can change only the offline block.
- Offline/offline comparison: You can change only the offline block in the left-hand area.

After a block change, it may be necessary to manually update the comparison result in the comparison editor to ensure that the comparison status is displayed correctly. You can then specify actions to synchronize the objects.

Note

You cannot change SCL blocks manually. However, you can apply changes from one block to the other block. In this regard, note the following points:

- You cannot apply any changes in an online block.
- You can only apply changes in an offline block if this is not write-protected. This is the case, for example, if the blocks of the detailed comparison come from different CPUs. It is then also possible to apply the changes to the block in the right-hand area.

Changing LAD, FBD or STL blocks

To change LAD, FBD or STL blocks, follow these steps:

1. Change the block in the left-hand area according to your requirements.
2. If necessary, click on "Update comparison results" in the toolbar.

Changing GRAPH blocks

To change a GRAPH block, follow these steps:

1. Switch to sequence view by clicking "Sequence view" between the two blocks.
2. Change the block in the left-hand area according to your requirements.
3. If necessary, click on "Update comparison results" in the toolbar.

Changing SCL blocks

To apply a change from one block to another one, follow these steps:

1. In the sidebar of the block from which you want to apply a change to another block, click on the arrow in the corresponding line.
   The line is inserted in the other block and the arrow buttons are removed.

   Note

   The colors of the arrows mean the following:

   - Gray: The changes cannot be applied to the other block, as the other block is either an online block or write-protected.
   - Blue: The changes are applied from an offline block to the other block.
   - Orange: The changes are applied from an online block to the other block.

2. If necessary, click on "Update comparison results" in the toolbar.

See also

Basic information on comparing PLC programs (Page 7585)
Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 7591)
Start detailed comparison for GRAPH blocks (Page 7593)
Visualization of the comparison result for LAD/FBD (Page 7596)
Visualization of the comparison result for STL (Page 7600)
Visualization of the comparison result for SCL (Page 7603)
Visualization of the comparison result for GRAPH (Page 7606)
Navigating in the detailed comparison (Page 7612)
Updating comparison results (Page 7615)

20.2.2.6 Updating comparison results

As soon as you change an object, the comparison results are no longer valid and must be updated.

Requirement

You have run a detailed comparison.

Procedure

To update the comparison results, follow these steps:

1. Click "Update the comparison result" in the toolbar.

See also

Basic information on comparing PLC programs (Page 7585)
Start a detailed comparison for LAD/FBD/STL/SCL blocks (Page 7591)
Start detailed comparison for GRAPH blocks (Page 7593)
Visualization of the comparison result for LAD/FBD (Page 7596)
Visualization of the comparison result for STL (Page 7600)
Visualization of the comparison result for SCL (Page 7603)
Visualization of the comparison result for GRAPH (Page 7606)
Navigating in the detailed comparison (Page 7612)
Changing blocks during the detailed comparison (Page 7613)

20.3 Comparing PLC tags

You have the following options for comparing PLC tags:

- Automatic offline/offline comparison in the compare editor
  The PLC tag tables of the selected devices are compared offline.
- Manual offline/offline comparison in the compare editor
  The selected PLC tag tables of the devices are compared offline.
- Offline/online comparison (only for S7-1500 version 2.5 and higher)
  The PLC tags in the project are compared with the PLC tags of the selected device.
- Detailed comparison
  Use the detailed comparison to determine differences within the PLC tag tables. You can start the detailed comparison either in the compare editor, in the project tree or in the PLC tag table "PLC tags" (only offline/online). The offline/online detailed comparison is available for all devices; the offline/online detailed comparison for S7-1500 CPUs as of version 2.5.

Performing automatic offline/offline comparison in the compare editor

To perform an automatic offline/offline comparison of PLC tag tables, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. Open the "PLC tags" folder.
   You can identify the status of the PLC tag tables based on the symbols in the status and action area. You can define certain actions depending on the status.

You can drag any other device to the drop area at any time to perform further comparisons.

Performing manual offline/offline comparison in the compare editor

To perform a manual offline/offline comparison of PLC tag tables, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. In the status and action area, click on the button for switching between automatic and manual comparison.
5. Select the PLC tag tables that you want to compare.
   The properties comparison is displayed. You can identify the status based on the symbols.

You can drag any other device to the drop area at any time to perform further comparisons.

Run offline/online comparison (only for S7-1500 version 2.5 and higher)

To perform an offline/offline comparison of PLC tag tables, follow these steps:

1. In the project tree, select a device that allows offline/online comparison.
2. Select the "Compare > Offline/online" command in the shortcut menu.
3. If you have not already established an online connection to this device, the "Go online" dialog opens. In this case, set all the necessary parameters for the connection and click "Connect".
   The online connection is established and the compare editor opens.
4. Open the "PLC tags" folder.
   You can identify the status of the PLC tag tables based on the symbols in the status and action area. You can define certain actions depending on the status of the objects. Note, however, that you can only perform actions in one direction in a synchronization action.

Starting detailed comparisons using the compare editor

To start a detailed comparison for a PLC tag table using the compare editor, follow these steps:

1. Perform an offline/offline or an offline/online comparison.
   The compare editor opens.
2. For an automatic offline/offline comparison or offline/online comparison in the compare editor, select the PLC tag table for which you want to run a detailed comparison. Note that two PLC tag tables must be selected for comparison for a manual offline/offline comparison.
3. Click the "Start detailed comparison" button in the toolbar.
   A separate compare editor opens. All existing PLC tags and user constants of the selected PLC tag tables are displayed depending on the settings of the compare editor. However, system constants are not shown. You can identify the status of the PLC tags based on the symbols. You can define certain actions for an offline/offline comparison depending on the status of the PLC tags.

Starting detailed comparison in the project tree

To start an offline/offline detailed comparison for a PLC tag table in the project tree, follow these steps:

1. Right-click the PLC tag table that you want to compare. This can also be a PLC tag table from a reference project.
2. Select the command "Quick compare > Select as left object" in the shortcut menu.
3. Right-click the PLC tag table that you want to compare with the PLC tag table that you previously selected as left object.
4. Select the command "Quick compare > Compare with <selected object>" in the shortcut menu. "<selected object>" stands for the left comparison object.
   A separate compare editor opens. All existing PLC tags and user constants of the selected PLC tag tables are displayed depending on the settings of the compare editor. However, system constants are not shown. You can identify the status of the PLC tags based on the symbols. You can define certain actions depending on the status of the PLC tags.

To start an offline/online detailed comparison for a PLC tag table directly in the project tree, follow these steps:

1. Establish an online connection to the device where the PLC tag table is located.
2. Right-click the PLC tag table that you want to compare with your online object.
3. Select the command "Quick compare > Compare with the online object" in the shortcut menu.
   A separate compare editor opens. All existing PLC tags and user constants of the selected PLC tag tables are displayed depending on the settings of the compare editor. However, system constants are not shown. You can identify the status of the PLC tags based on the symbols.

Start detailed comparison in the PLC tag table "PLC tags"

To start an offline/online detailed comparison for a PLC tag table in the PLC tag table "Show all tags", follow these steps:

1. Establish an online connection to the device where the PLC tag table is located.
2. Open the "PLC tags" folder under the CPU in the project tree.
3. Double-click the entry "Show all tags".
   The PLC tag table "PLC tags" opens.
4. Click the "Start detailed comparison" button in the toolbar.
   A separate compare editor opens. All existing PLC tags and user constants of the selected PLC tag tables are displayed depending on the settings of the compare editor. However, system constants are not shown. You can identify the status of the PLC tags based on the symbols.

See also

Introduction to comparing PLC programs (Page 7585)

Comparing PLC data types (Page 7619)

Comparing blocks (Page 7590)

20.4 Comparing PLC data types

You have the following options for comparing PLC data types:

- Offline/online comparison (S7-1200/1500 only)
  The PLC data types in the project are compared with the PLC data types of the selected device.
- Automatic offline/offline comparison in the compare editor
  The PLC data types of the selected devices are compared offline.
- Manual offline/offline comparison in the compare editor
  The selected PLC data types of the devices are compared offline.
- Detailed comparison
  Use the detailed comparison to determine differences between PLC data types. You can start the detailed comparison either via the compare editor or in the project tree.

Performing offline/online comparison of PLC data types in the compare editor

To perform an offline/online comparison, follow these steps:

1. In the project tree, select a device that allows offline/online comparison.
2. Select the "Compare > Offline/online" command in the shortcut menu.
   If you have not already established an online connection to this device, the "Go online" dialog opens. In this case, set all the necessary parameters for the connection and click "Connect".
   The online connection is established and the compare editor opens.
3. Open the "PLC data types" folder.
   You can identify the status of the PLC data types based on the symbols in the status and action area. When you select an object, the properties of the PLC data type and the corresponding PLC data type of the assigned device are displayed in the properties comparison.

Performing automatic offline/offline comparison in the compare editor

To perform an automatic offline/offline comparison of PLC data types, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. Open the "PLC data types" folder.
   You can identify the status of the PLC data types based on the symbols in the status and action area. You can define certain actions depending on the status. You can drag any other device to the drop area at any time to perform further comparisons.

Performing manual offline/offline comparison in the compare editor

To perform a manual offline/offline comparison of PLC data types, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. In the status and action area, click on the button for switching between automatic and manual comparison.
5. Select the PLC data types that you want to compare.
   The properties comparison is displayed. You can identify the status based on the symbols. You can drag any other device to the drop area at any time to perform further comparisons.

Starting detailed comparisons using the compare editor

To start a detailed comparison for a PLC data type using the compare editor, follow these steps:

1. Perform an offline/offline comparison. You can also perform an offline/online comparison for CPUs of the S7-1200/1500 series.
2. For an automatic offline/offline comparison in the compare editor, select the PLC data type for which you want to run a detailed comparison. Note that two PLC data types must be selected for comparison with a manual offline/offline comparison.
3. Click the "Start detailed comparison" button in the toolbar.
   The "PLC data type comparison" window opens and all tags of the compared PLC data types are displayed. You can view the differences under "Info > Result of comparison" in the inspector window.

Starting detailed comparison in the project tree

To start a offline/offline detailed comparison for a PLC data type directly in the project tree, follow these steps:

1. Right-click the PLC data type that you want to compare. This can also be a PLC data type from a reference project.
2. Select the command "Quick compare > Select as left object" in the shortcut menu.
3. Right-click the PLC data type that you want to compare with the PLC data type that you previously selected as left object.
4. Select the command "Quick compare > Compare with <selected object>" in the shortcut menu. "<selected object>" stands for the left comparison object.
   The "PLC data type comparison" window opens and all tags of the compared PLC data types are displayed. You can view the differences under "Info > Result of comparison" in the inspector window.

To start an offline/online detailed comparison for a PLC data type for a CPU of the S7-1200/1500 series directly in the project tree, follow these steps:

1. Establish an online connection to the device where the block is located.
2. Right-click the PLC data type that you want to compare with its online object.
3. Select the command "Quick compare > Compare with the online object" in the shortcut menu.
   The "PLC data type comparison" window opens and all tags of the compared PLC data types are displayed. You can view the differences under "Info > Result of comparison" in the inspector window.

See also

Introduction to comparing PLC programs (Page 7585)

Comparing blocks (Page 7590)

Comparing PLC tags (Page 7616)

Displaying program information 21

21.1 Overview of available program information

Program information

The program information of a user program contains the view specified in the following table.

View	Application
Assignment list (Page 7624)	Provides an overview of the address bits for the I, Q, and M memory areas already allocated within the user program. Also indicates if an address has been allocated by access from an S7 program or if the address has been assigned to a SIMATIC S7 module.
Call structure (Page 7633)	Shows the call structure of the blocks within the user program and provides an overview of the blocks used and their relationships.
Dependency structure (Page 7640)	Shows the list of blocks used in the user program. A block is shown at the first level and blocks that call or use this block are indented below it. In contrast to the call structure, instance blocks are listed separately.
Resources (Page 7646)	Shows the hardware resources of the CPU for objects (OB, FC, FB, DB, user-defined data types and PLC tags), for CPU memory areas and for the existing I/O modules.

Displaying several views simultaneously

You can generate and display several views for one or more user programs to facilitate testing and changing your user program.

Displaying multiple views, for example, enables you to:

- Display all program information for a user program next to one another
- Compare different user programs

21.2 Displaying an assignment list

21.2.1 Introduction to the assignment list

Program information in the assignment list

The assignment list shows if an address has been allocated by access from an S7 program or if the address has been assigned to a SIMATIC S7 module. It is therefore an important basis for locating errors or changes in the user program.

In the assignment list, you have a CPU-specific overview of which bit is used in which byte of the memory areas listed below:

- Input (I)
- Output (O)
- Bit memory (M)
- Timer (T)
- Counter (C)
- I/O (P)

Display of the assignment list

The assignment list of inputs, outputs, and bit memory is displayed in several separate work windows.

Filters

You can filter the display within the assignment list. You can use predefined filters or create your own.

Displaying cross-reference information

You have the option of displaying cross-reference information for selected addresses in the assignment list.

You can display the cross-references for a selected address in the Inspector window using the "Cross-reference information" shortcut menu command. The command "Tools > Cross-references" allows you to also open the cross-reference list for the selected object.

Displaying the PLC tag table

You can open the PLC tag table from the assignment list and edit the properties of the tags used.

To do this select an address of the assignment list and select the "Open editor" command in the shortcut menu.

Enabling the display of retentivity

You can enable and disable the display of the retentive state of bit memory by selecting the "Hide/show retain area" toolbar button.

See also

Symbols in the assignment list (Page 7626)

Layout of the assignment list (Page 7625)

21.2.2 Layout of the assignment list

Layout of the assignment list

Depending on the CPU, the assignment list is displayed in several work windows with the following operands.

For S7-300/400 and S7-1500 CPUs:

- Inputs
- Outputs
- Bit memories
- Timers
- Counters

For S7-1200 CPUs:

- Inputs
- Outputs
- Bit memories

Displaying inputs, outputs, bit memory, timers and counters

It shows all operands used and their assignment in the S7 program.

For all displayed operands, each line in the assignment list is dedicated to a byte of the memory area, in which the corresponding eight bits from 7 to 0 are labeled according to their access. Afterwards, a "bar" indicates whether access is made by a byte (B), word (W) or double word (D).

You can find an explanation of the symbols in the assignment list here: (Page 7626)

See also

Introduction to the assignment list (Page 7624)

21.2.3 Symbols in the assignment list

Meaning of the symbols in the assignment list

The following table shows the meaning of the symbols in the assignment list:

Symbol	Meaning
	Indicates the address assignment in the selected state.
	Indicates the address assignment in the non-selected state.
	Indicates that a pointer start address and a tag address access the same address range and that they are selected.
	Indicates that a pointer start address and a tag address access the same address range and that they are not selected.
	Indicates the pointer assignment in the selected state.
	Indicates the pointer assignment in the non-selected state.
	Indicates that the byte is in use with byte access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
	Indicates that the byte is in use with byte access and the corresponding tag is not selected.
	Indicates that the byte is in use with word access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
	Indicates that the byte is in use with word access and the corresponding tag is not selected.
	Indicates that the byte is in use with double word access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
	Indicates that the byte is in use with double word access and the corresponding tag is not selected.
Background color: gray	Indicates that a byte is in use with byte, word or double word access and that the address is also in use by the hardware. The gray background color indicates overlapping memory access.
Background color: yellow	Indicates that the address is not in use by the hardware.
	Indicates that the memory area has been defined as system memory.
	Indicates that the memory area has been defined as clock memory.

See also

Layout of the assignment list (Page 7625)

Introduction to the assignment list (Page 7624)

21.2.4 Displaying an assignment list

Requirement

A project has been created with programmed blocks.

Procedure

Proceed as follows to display the assignment list:

1. Select the "Program blocks" folder or one or more of the blocks it contains.
2. Select the "Assignment list" command in the "Tools" menu.

Result

The assignment list for the selected program is displayed.

View options in the assignment list

Refer to view respective view options that are set to display the desired information in the assignment list.

See also

Setting the view options for the assignment list (Page 7627)

Layout of the assignment list (Page 7625)

21.2.5 Setting the view options for the assignment list

Introduction

The following view options are available for the assignment list:

- Used addresses:
  When this check box is activated, the addresses, I/Os and pointers used in the program are displayed.
- Free hardware addresses:
  When this check box is activated, only the free hardware addresses are displayed.

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Procedure

Proceed as follows to set the view options for the assignment list:

1. Click on the arrow of the symbol ("View options") in the task bar.
   The view options for the assignment list are opened. Check marks are set in front of the activated view options.
2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the desired information is displayed in the assignment list.

21.2.6 Filter options in the assignment list

Filter settings

You can define your own filter settings for the assignment list. The following options are available for defining filters:

- Display all addresses of the address areas specified.
- Display of single, defined addresses from the selected address area, for example, "0" and "200".
- Display of complete areas from the selected address area, for example, "0 - 256".

The following table provides an overview of all available options:

Selection in the	Selection	Symbol	Meaning
Address area	All CPU-dependent displayed addresses (I, O, M, T, C) can be activated as they are by default, or individual address areas can be activated.	Check box is activated	Only the activated address areas (I, O, M, T, C) are shown in the assignment list.
Filter area	Show assignment for all addresses	*	Displays the assignment of all addresses of the enabled address areas (I, Q, M).
	Show assignment for selected addresses, for example, for the inputs "IB 0" and "IB 256"	0;256 Separate individual addresses and areas by a semicolon.	Assignments of selected addresses for the activated address areas (I) are shown.
	Show assignment for selected areas, for example, for the inputs "IB 0 to IB 100" and "IB 200 to IB 256".	0-100;200-256 Contiguous areas should be connected by a hyphen.	Assignments of selected areas for the activated address areas (I) are shown.

21.2.7 Defining filters for assignment list

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Defining filter

Proceed as follows to define a filter for the assignment list:

1. Click on the symbol ("Filter") in the task bar.
   The "Assignment List Filter" dialog opens.
2. Click on the symbol ("Create new filter") in the task bar.
   A new filter is created with the name "Filter_1". The check boxes for all addresses (inputs, outputs, memory bits, timers and counters) are activated by default for the filter.
3. If you want to change the name of the filter, click on the drop-down list in the task bar and enter a new filter name.
4. Deactivate the check boxes of addresses that are not to be affected by the filter.

5. Enter one of the following options in the filter area of the activated address:
   - Show all addresses used = "*"
   - Show single, defined addresses, for example, IB 0" and IB 25 = "0.25". Individual addresses and address areas are separated by commas or semicolons.
   - Show complete address areas, for example, IB 0 to IB 256 = "0-256". Complete address areas should be connected by a hyphen.
6. Confirm your entries with "OK".
   The newly defined filter is shown in the task bar of the assignment list under the specified name.

Delete filter

Proceed as follows to delete a filter:

1. Click on the symbol ("Filter") in the task bar.
   The filter dialog for the assignment list opens.
2. In the drop-down list of the task bar, select the filter you want to delete.
3. Click on the symbol ("Delete selected filter") in the task bar.
   The selected filter is deleted.

See also

Filter options in the assignment list (Page 7628)

Displaying an assignment list (Page 7627)

Introduction to the assignment list (Page 7624)

21.2.8 Filtering an assignment list

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Procedure

1. Click on the arrow on the drop-down list.
   The available filter are displayed.
2. Select the desired filter.

Result

The assignment list is filtered according to the settings of the selected filter.

Note

The filter settings are saved when the project is closed.

21.2.9 Defining retentive memory areas for bit memories

Introduction

In the assignment list you can define the width of the retentive memory area for bit memories. The content of tags which are addressed in retentive memory is retained after power off and at the STOP to RUN transition after power on.

The display of retentive bit memories can be enabled and disabled in the assignment list. If their display is enabled, retentive bit memories are identified by an icon in the "Address" column.

Requirement

The assignment list is open.

Procedure

Proceed as follows to define the width of the retentive memory area for bit memories:

1. Click "Retain" in the toolbar.
   The "Retain memory" dialog will open.
2. Starting at the count of 0, define the width of the retentive memory area by entering the last byte of this area in the input field. Watch out for any addresses of tags already assigned to the retentive area.
3. Load the block to the target system. Select the "Program blocks" folder in the Project tree and select the "Download to device" submenu in the shortcut menu.

Result

The width of the retentive memory area is defined. If enabled in the assignment list, an icon will indicate the retentive state of all tags in the "Address" column.

21.2.10 Enabling the display of retentive bit memories

Introduction

In the assignment list you can enable and disable the display of retentive bit memories. The retentive bit memories are identified by means of an icon in the "Address" column if the display of retentivity is enabled.

Requirement

The assignment list is open.

Procedure

Proceed as follows to enable and disable the display of retentive bit memories:

1. Click "Display/hide retentivity" in the toolbar.

Result

The retentive tags are identified by means of an icon in the "Address" column of the bit memory area if the display of retentivity is enabled. The icons in the "Address" column are hidden if the display of retentivity is disabled.

21.3 Displaying the call structure

21.3.1 Introduction to the call structure

Call structure

The call structure describes the call hierarchy of the blocks within an S7 program.

It provides an overview of:

- The blocks used
- Jumps to the places of use of the blocks
- Dependencies between blocks
- Local data requirements of the blocks
- Status of the blocks

Information in the call structure

Displaying the call structure provides you with a list of the blocks used in the user program. The first level of the call structure is highlighted in color and shows the blocks that are not called by any other block in the program. Organization blocks are always shown on the first level of the call structure. Functions, function blocks and data blocks are only shown on the first level if they are not called by an organization block. When a block calls other blocks or functions, they are listed indented under the calling block. Instructions and blocks are shown in the call structure only if they are called by a block.

Note

Please note that no call structure is displayed for know-how-protected blocks.

View options

The following view options are available for the call structure:

- Show conflicts only:
  When this check box is activated, only conflicts within the call structure are displayed.
- Group multiple calls together:
  When this check box is activated, several block calls are grouped together. The number of block calls is displayed in the "Call frequency" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Displaying the block calls

You can display the block calls in a block by clicking on the arrow in front of the block title. To display the call information of all blocks, click on the "Expand list" icon in the toolbar.

You can hide the total overview by clicking the "Collapse list" icon.

Displaying cross-reference information

You can display the cross-reference information for a block in the Inspector window by right-clicking on the relevant block and selecting the "Cross-reference information" command from the shortcut menu.

To open the "Cross-references" view, click the "Cross-references" command in the shortcut menu.

Displaying blocks in the program editor

You can open the program editor from the call structure and edit blocks there.

To do this, select the required block in the call structure and select the "Open" command in the shortcut menu.

Displaying deleted blocks

The rows belonging to deleted blocks are identified by an icon.

Note

Please note that any existing local data can only be displayed or updated after a block is compiled.

See also

Symbols in the call structure (Page 7634)

21.3.2 Symbols in the call structure

Meaning of the symbols in the call structure

The following table shows the meaning of the symbols in the call structure:

Symbol	Meaning
	Indicates an organization block (OB).
	Indicates a function block (FB).
	Indicates a function (FC).
	Indicates a data block (DB).
	Indicates that the block is declared as a multiple instance or as a parameter instance.

Symbol	Meaning
	The object has an interface dependency to an object connected to the left.
	Indicates that the block needs to be compiled again.
	Indicates that the data block needs to be compiled again.
	Indicates that the object is not available.
	Indicates that the interface causes a time stamp conflict.
	Indicates that the variable causes a time stamp conflict.
	Indicates that the block is not called directly or indirectly from an OB.
	Indicates that an object has know-how protection.
	Indicates that a tag declaration in the interface has a recursive dependency: • Scenario 1: FB1 calls FB2 and this then calls FB1. The instance data blocks of these FBs have a recursion in the interface. • Scenario 2: A multiple instance FB uses the instance DB of its parent FB as a global DB.
	Indicates that the block is normally called recursively.
	Indicates that a block is called recursively with a condition.
	Indicates that a block is called recursively without a condition.

21.3.3 Layout of the call structure

Layout of the call structure

The view of the call structure consists of the following columns:

Column	Content/meaning
Call structure	Shows an overview of the blocks called If the viewing option "Group multiple calls together" is enabled, several block calls are grouped together and the "Number of calls" column is displayed.
Call type (!)	Shows the type of call, for example recursive block call.
Address	Shows the absolute address of the block. With a function block, the absolute address of the corresponding instance data block is also shown.
Call frequency	Indicates the number of multiple calls of blocks.

Column	Content/meaning
Details	Shows the network or interface of the calling block. All information are offered as a link in this column. With this link, you can jump to the location of the block call in the program editor. If the viewing option "Group multiple calls together" option is enabled, the calls are grouped together and are available as links in a drop-down list.
Local data (in path)	Indicates the local data requirement of the full path. Blocks with optimized access have higher local data requirements because the information for the symbolic addressing is stored with them. Please note that any existing local data can only be displayed or updated after compiling a block.
Local data (for blocks)	Show the local data requirements of the block. Blocks with optimized access have higher local data requirements because the information for the symbolic addressing is stored with them. Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Symbols in the call structure (Page 7634)

Introducing the consistency check in the call structure (Page 7638)

21.3.4 Displaying the call structure

Requirement

A project has been created with blocks.

Procedure

Proceed as follows to display the call structure:

1. Select the "Program blocks" folder or one or more of the blocks it contains.
2. Select the "Call structure" command in the "Tools" menu.

Result

The call structure for the selected program is displayed.

Note

Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Setting the view options for the call structure (Page 7637)

21.3.5 Setting the view options for the call structure

Introduction

The following view options are available for the call structure:

- Show conflicts only:
  Only the blocks causing conflicts within the call structure are displayed if this check box is activated.
  The following blocks cause conflicts:
  - Blocks executing any calls with older or newer code time stamps.
  - Blocks calling a block with modified interface.
  - Blocks using a tag with modified address and/or data type.
  - Block called neither directly, nor indirectly by an OB.
  - Blocks calling a block which no longer exists.
- Group multiple calls together:
  When this viewing option is enabled, several block calls and data block accesses are grouped together. The number of block calls is displayed in the "Call frequency" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Requirement

- A project has been created with programmed blocks.
- The call structure is open.

Procedure

Proceed as follows to set the view options for the call structure:

1. Click on the arrow of the symbol ("View options") in the task bar.
   The view options for the call structure opens. Check marks are set in front of the activated view options.
2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the required information is displayed in the call structure.

21.3.6 Introducing the consistency check in the call structure

Consistency check

Changing the time stamp of a block during or after the program is generated can lead to time stamp conflicts, which in turn cause inconsistencies among the blocks that are calling and being called.

Using the consistency check

The "Consistency check" function is used to visualize inconsistencies when time stamp conflicts occur. Whe the consistency check is performed, the inconsistent blocks are shown in the call structure and marked with the correspoinding symbols.

- Most time stamp and interface conflicts can be rectified by recompiling the blocks.
- If compilation fails to clear up inconsistencies you can use the link in the "Details" column to go to the source of the problem in the program editor and manually eliminate any inconsistencies.
- The blocks marked in red must be recompiled.

See also

Symbols in the call structure (Page 7634)

21.3.7 Checking block consistency in the call structure

Requirement

- A project has been created with programmed blocks.
- The call structure is open.

Procedure

Proceed as follows to check the block consistency:

1. Click on the symbol ("Consistency check") in the task bar.
   The block consistency is checked. Blocks found to be inconsistent are marked accordingly by a symbol.
2. If a block is inconsistent, click on the arrow in front of the block title in the call structure.
   The inconsistent blocks are displayed. The exact problem locations are listed as links in the "Details" column.
3. Click on the respective link in the "Details" column to jump to the location in the block requiring correction.
4. Check and correct the inconsistencies in the blocks.

5. Recompile the blocks by selecting the required blocks and clicking on the command "Compile" in the shortcut menu.
6. Download the corrected blocks to the target system by clicking the command "Download to device" in the shortcut menu.

Result

The block consistency is checked. The inconsistencies in the blocks are corrected. The corrected blocks are loaded to the target system.

See also

Symbols in the call structure (Page 7634)

21.4 Displaying the dependency structure

21.4.1 Introduction to the dependency structure

Introduction

The dependency structure shows the dependencies each block in the program has to other blocks.

Information in the dependency structure

Displaying the dependency structure provides you with a list of the blocks used in the user program. A block is shown at the far left, and blocks that call or use this block are indented below it.

The dependency structure also shows the status of the individual blocks using symbols.

Objects that cause a time stamp conflict and could lead to an inconsistency in the program are marked with various symbols.

The dependency structure is an extension of the cross-reference list for objects.

View options

The following view options are available for the dependency structure:

- Show conflicts only:
  When this check box is activated, only the conflicts within the dependency structure are displayed.
- Group multiple calls together:
  When this check box is activated, several block calls are grouped together. The number of block calls is shown numerically in the "Dependency structure" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Displaying the dependencies

Clicking on the arrow in front of the block title displays the blocks that call or use this block. To display the dependencies of all blocks,

click the "Expand list" icon in the toolbar.

You can hide the total overview by clicking the "Collapse list" icon.

Displaying cross-reference information

You can display the cross-reference information for a block in the Inspector window by right-clicking on the relevant block and selecting the "Cross-reference information" command from the shortcut menu.

To open the "Cross-references" view, click the "Cross-references" command in the shortcut menu.

Displaying blocks in the program editor

You can open the program editor from the dependency structure and edit blocks there. To do this, select the required block in the dependency structure and select the "Open" command in the shortcut menu.

21.4.2 Layout of the dependency structure

Layout of the dependency structure

The view of the dependency structure consists of the following columns:

Column	Content/meaning
Dependency	It indicates the dependencies between each block and the other blocks in the program.
Call type (!)	Shows the type of call, for example recursive block call.
Address	Shows the absolute address of the block.
Call frequency	Indicates the number of multiple calls of blocks.
Details	Shows the network or interface of the called block. All information are offered as a link in this column. With this link, you can jump to the location of the block call in the program editor. If the viewing option "Group multiple calls together" option is enabled, the calls are grouped together and are available as links in a drop-down list.

See also

Symbols in the dependency structure (Page 7641)

21.4.3 Symbols in the dependency structure

Meaning of the symbols in the dependency structure

The following table shows the meaning of the symbols in the dependency structure:

Symbol	Meaning
	Indicates an organization block (OB).
	Indicates a function block (FB).
	Indicates a function (FC).

Symbol	Meaning
	Indicates a data block (DB).
	Indicates that the block is declared as a multiple instance or as a parameter instance.
	The object has an interface dependency to an object connected to the left.
	Indicates that the block needs to be compiled again.
	Indicates that the data block needs to be compiled again.
	Indicates that there is an inconsistency with this object.
	Indicates that an object has know-how protection.
	Indicates that a tag declaration in the interface has a recursive dependency: • Scenario 1: FB1 calls FB2 and this then calls FB1. The instance data blocks of these FBs have a recursion in the interface. • Scenario 2: A multiple instance FB uses the instance DB of its parent FB as a global DB.
	Indicates that the block is normally called recursively.
	Indicates that a block is called recursively with a condition.
	Indicates that a block is called recursively without a condition.

21.4.4 Displaying the dependency structure

Requirement

A project has been created with programmed blocks.

Procedure

Proceed as follows to display the dependency structure:

1. Select the block folder or one or more of the blocks contained therein.
2. Select the "Dependency structure" command in the "Tools" menu.

Result

The dependency structure for the selected program is displayed.

See also

Setting the view options for the dependency structure (Page 7643)

21.4.5 Setting the view options for the dependency structure

Introduction

The following view options are available for the dependency structure:

- Show conflicts only:
  When this check box is activated, only the conflicts within the dependency structure are displayed.
  The following blocks cause conflicts:
  - Blocks executing any calls with older or newer code time stamps.
  - Blocks called by a block with modified interface.
  - Blocks using a tag with modified address and/or data type.
  - Block called neither directly, nor indirectly by an OB.
- Group multiple calls together:
  When this check box is activated, several block calls are grouped together. The number of block calls is shown in the relevant column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Requirement

- A project has been created with programmed blocks.
- The dependency structure is open.

Procedure

Proceed as follows to set the view options for the dependency structure:

1. Click on the arrow of the symbol ("View options") in the task bar.
   The view options for the dependency structure are opened. Check marks are set in front of the activated view options.
2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the required information is displayed in the dependency structure.

21.4.6 Introducing the consistency check in the dependency structure

Consistency check

Changing the time stamp of a block during or after program creation can lead to time stamp conflicts, which in turn cause inconsistencies among the blocks that are calling and being called.

Using the consistency check

The "Consistency check" function is used to visualize inconsistencies. When the consistency check is performed, the inconsistent blocks are shown in the dependency structure and marked with the corresponding symbols.

- Most time stamp and interface conflicts can be rectified by recompiling the blocks.
- If compilation fails to clear up inconsistencies, you can use the link in the "Details" column to go to the source of the problem in the program editor and manually eliminate any inconsistencies.
- The blocks marked in red must be recompiled.

See also

Layout of the dependency structure (Page 7641)

Symbols in the dependency structure (Page 7641)

21.4.7 Checking block consistency in the dependency structure

Requirement

- A project has been created with programmed blocks.
- The dependency structure is open.

Procedure

Proceed as follows to check the block consistency:

1. Click on the symbol ("Consistency check") in the task bar.
   The block consistency is checked. Blocks found to be inconsistent are marked accordingly by a symbol.
2. If a block is inconsistent, click on the arrow in front of the block title in the dependency structure.
   The inconsistent blocks are displayed. The exact problem locations are listed as links in the "Details" column.
3. Check and correct the inconsistencies in the blocks.
4. Recompile the blocks by selecting the required blocks and clicking on the command "Compile" in the shortcut menu.
5. Download the corrected blocks to the target system by clicking the command "Download to device" in the shortcut menu.

Result

The block consistency is checked. The inconsistencies in the blocks are corrected. The corrected blocks are loaded to the target system.

See also

Symbols in the dependency structure (Page 7641)

21.5 Displaying CPU resources

21.5.1 Introducing resources

Introduction

The "Resources" tab indicates the hardware resources of the configured CPU for:

- the used programming objects,
- the assignment of the different memory areas within the CPU and
- the assigned inputs and outputs of the existing input and output modules.

Information provided in the "Resources" tab

The resources tab provides an overview of the hardware resources. The display in this tab depends on the CPU which you are using. The following information is displayed:

- the programming objects used in the CPU (e.g. OB, FC, FB, DB, data types and PLC tags)
- the memory areas available on the CPU (load memory, work memory - divided into code work memory and data work memory depending on the CPU -, retentive memory), their maximum size and utilization by the programming objects stated above
- the I/O of modules which can be configured for the CPU (I/O modules, digital input modules, digital output modules, analog input modules, and analog output modules), including the I/O already in use.

Display of the maximum available load memory

The maximum size of available load memory can be selected from a drop-down list box in the "Total" row of the "Load memory" column.

Note

Displaying the allocated load memory in the CPU

Note that the sum of the allocated load memory cannot be precisely determined if not all blocks are compiled.

In this case, a preceding ">" in front of the sum indicates that the amount of allocated space may be greater than that shown because uncompiled blocks are not taken into account in the total.

Display of the maximum available work memory

The maximum size of available work memory is displayed in the "Work memory" column or in the "Code work memory" and "Data work memory" columns in the "Total" row.

Display of the maximum available retentive memory

The maximum size of available retentive memory is displayed in the "Retain memory" column in the "Total" row.

Note

Retentive memory data

All bit memories and data blocks specified as retentive will be integrated in the calculation of the retentive data.

Updating the display in the "Resources" tab

Click the "Update view" toolbar button to update the display of objects.

Benefits of the display in the "Resources" tab

The "Resources" tab of the program information dialog provides a detailed list of all objects and of the corresponding memory area used.

The tab also indicates shortage of resources and helps to avoid such states.

Blocks which are not compiled can be identified as their size is indicated by a question mark.

See also

Layout of the "Resources" tab (Page 7648)

Displaying resources (Page 7649)

Selecting the maximum load memory available (Page 7650)

21.5.2 Layout of the "Resources" tab

Layout of the "Resources" tab in the program information

The view of the "Resources" tab consists of the following columns:

Column	Content/meaning
Objects	The "Details" area provides an overview of the programming objects available in the CPU, including their memory assignments.
Load memory	Displays the maximum load memory resources of the CPU as a percentage and as absolute value. The values displayed under "Total" provide information on the maximum memory available in the load memory. The values displayed under "Used" provide information on the memory actually used in the load memory. If a value is displayed in red, the available memory capacity has been exceeded.
Work memory or code and data work memory	Displays the maximum work memory resources of the CPU as a percentage and as absolute value. The work memory depends on the CPU and is divided into "Code work memory" and "Data work memory" for a CPU from the S7-400 or S7-1500 series, for example. The values displayed under "Total" provide information on the maximum memory available in the work memory. The values displayed under "Used" provide information on the memory space actually used in the work memory. If a value is displayed in red, the available memory capacity has been exceeded.
Retentive memory	Displays the maximum resources for retentive memory in the CPU as a percentage and as absolute value. The values displayed under "Total" provide information on the maximum memory available in the retentive memory. The values displayed under "Used" provide information on the memory actually used in the retentive memory. If a value is displayed in red, the available memory capacity has been exceeded.

Column	Content/meaning
I/O	Displays the I/Os which are available on the CPU, including their module-specific availability in the next columns. The values displayed at "Configured" provide information about the maximum number of I/O available. The values displayed under "Used" provide information on the actually used inputs and outputs.
DI / DQ / AI / AQ	Displays the number of configured and used inputs/outputs: DI = Digital inputs DQ = Digital outputs AI = Analog inputs AQ = Analog outputs The values displayed at "Configured" provide information about the maximum number of I/O available. The values displayed under "Used" provide information on the actually used inputs and outputs.

See also

Displaying resources (Page 7649)

Selecting the maximum load memory available (Page 7650)

Introducing resources (Page 7646)

21.5.3 Displaying resources

Introduction

The memory use shows you which space is allocated to specific objects in the CPU.

Note

Displaying the allocated load memory in the CPU

Note that the sum of the allocated load memory cannot be precisely determined if not all blocks are compiled.

In this case, a preceding ">" in front of the sum indicates that the amount of allocated space may be greater than that shown because uncompiled blocks are not taken into account in the total.

Requirement

A project with programmed blocks has been created.

Procedure

Proceed as follows to display the resources of the respective CPU memory areas:

1. Select the block folder below the relevant CPU, or one or several of the blocks contained therein.
2. Select the "Resources" command in the "Tools" menu.

Result

The memory resources of the assigned CPU are displayed.

21.5.4 Selecting the maximum load memory available

Requirement

A project with programmed blocks has been created.

Procedure

Proceed as follows to display the available maximum of load memory resources:

1. Select the block folder below the relevant CPU, or one or several of the blocks contained therein.
2. Select the "Resources" command in the "Tools" menu.
3. In the dialog that is displayed, open the drop-down list in the "Total" field of the "Load memory" column by clicking the icon.
4. Select a corresponding value for the CPU used by clicking it in the drop-down list box.

Result

The "Total" field displays the selected maximum memory resources.

Note

Display of maximum memory resources

If a value is displayed in red for the maximum memory resources, the available memory capacity has been exceeded.

In this case, adapt the memory capacity as described above.

Displaying cross-references 22

22.1 General information about cross references

Introduction

The cross-reference list provides an overview of the use of objects and devices within the user program.

You can see which objects are interdependent, what relationships they have with one another and where the individual objects are located.

You can also jump directly to the reference location of an object.

Uses of cross-references

The cross-reference list offers you the following advantages:

- When creating and changing a program, you retain an overview of the devices and objects used such as the blocks, operands and tags etc.
- You can see whether the object uses other objects or is used by itself.
- You can filter the displayed cross-reference information by specific criteria. Predefined filters for source and reference objects are available to you to this purpose.
- In addition, you can also create user-defined filters to find relevant references more easily.
- From the cross-references, you can use the link highlighted in blue to jump directly to the reference location of the selected object.
- During a program test or when troubleshooting, you are informed of the following, for example:
  - In which block and by which command is an operand processed.
  - Which tag is used how and where.
  - Which block is called by which other block.
  - Cross-reference information for subordinate and higher-level structures.
- As part of the project documentation, the cross-references provide a comprehensive overview of all operands, memory areas, blocks, tags, screens etc. used.

Note

Objects displayed in the cross-reference list

- Which objects you can display in the cross-reference list depends on the installed products.
- As of TIA Portal V15 instructions with version identification can also be displayed with the cross-references. Instructions without version identification are **not** displayed.

See also

Structure of the cross-reference list (Page 7653)
Displaying the cross-reference list (Page 7658)
Displaying and editing cross-references in the Inspector window (Page 7682)

22.2 Structure of the cross-reference list

Introduction

The cross-reference list enables the display of reference objects and lower-level objects for a source object selected in the project tree.

The source object is the object displayed first in the cross-reference list.

The display of the cross-references for an object is project- and device-specific.

You can also have multiple cross-reference lists open at the same time.

Structure of the cross-reference list

When you have selected a source object and open the cross-reference list, the objects are displayed as follows:

- The source object and all the lower-level objects of this source object and the associated references are structured in a standard view.
- The source object is the object for which you have opened the cross-references and is displayed as the first object in a row with a gray background.
- Lower-level objects are marked with a preceding blue square.
- If lower-level objects also have reference objects, they are displayed with a gray background in the object column and in an hierarchical structure.
- If an object is used several times, it is also displayed several times.
- The "Add new source object" button is displayed at the end of a filled cross-reference list. When you click on this button, you can add a new object to the cross-reference list.
- The source object and the lower-level object are collapsed in the Inspector window. Lower-level objects with reference objects are show with a gray background in the object column.

Example

The graphic below shows an example of the structure of the cross-reference list to a source object:

Display of the newly added source objects in the cross-reference list

Objects which were inserted using the "Add new source object" button are displayed as follows:

- They are always at the end of the cross-reference list and are automatically expanded.
- They are initially displayed independently of the selected filter setting.
- Newly added objects are correctly sorted and the filter settings displayed accordingly after an update or when another filter is used.

Columns in the cross-reference list

The cross-reference list has the following structure:

Column	Content/meaning
Object	Displays the name of the source object which was selected when the cross-reference list was opened, with all lower-level objects and the associated reference objects.
Reference location	Shows the reference location of the objects, e.g. a network.
Reference type	Shows the relationship between the source object and the referenced objects: • "Uses": The source object uses this object. • "Used by": The source object is used by this object. • "Type -> Instance": The source object is a type of the referenced object. • "Instance -> Type": The source object is an instance of the referenced object. • "Group -> Element": The source object is a group of the referenced object. • "Element -> Group": The source object is an element of the referenced object. • "Defines": The source object defines the referenced object. • "Defined by": The source object is defined by the referenced object.
As	Shows additional information on the referenced objects.
Access	Shows the type of access, e.g. whether access to the operand is read access (R) and/or write access (W).
Address	Shows the address of the relevant object.
Type	Shows the type and the language used to create the object.
Device	Shows the associated device name, e.g. "CPU_1".
Path	Displays the path of the object in the project tree with specification of folders and groups.
Comment	Displays comments on individual objects, if available.

See also

General information about cross references (Page 7651)

Displaying the cross-reference list (Page 7658)

22.3 Settings in the cross-reference list

Settings in the cross-reference list

Settings can be made in the cross-reference list using the buttons in the toolbar and the selection of pre-defined filter settings.

Settings using the buttons in the cross-reference list

The following settings can be made using the buttons:

- **"Update" button**
  Updates the contents displayed in the open cross-reference list.
  When you press this button, the displayed contents of the cross-reference list are updated with the current contents from the associated project. Automatic update is not carried out within the cross-reference list if data management has changed in the project.
- **"Expand all" button**
  Expands the entries in the current cross-reference list by opening all lower-level objects.
- **"Collapse all" button**
  Reduces the entries in the current cross-reference list by closing the lower-level objects.
- **"Save window settings" button**
  Saves the current window settings in the cross-reference list.
  This includes, for example, the columns displayed, the width of the columns and the arrangement of the columns.
  The relevant filter settings in the cross-reference list are **not** saved.

Filter settings in the cross-reference list

You have the possibility of selecting predefined filters for displaying objects.

You have the following options for filter selection, which relate to source objects:

- **Show objects with references:**
  This setting shows all the lower-level objects below the source object with existing references. It is the default setting whenever the cross-reference list is opened.
- **Show objects without references:**
  This setting shows all the lower-level objects below the source object without references. These non-referenced objects are assigned the entry "no references available" in the column "Reference location". Lower-level elements without references are also displayed if their parent objects contain references.
- **Show unused objects:**
  This setting shows all the lower-level but unused objects below the source object. These unused objects are assigned the entry "Object is not used" in the column "Reference location".
- **Show all objects:**
  This setting shows all the lower-level objects below the source object with or without references.

You have the following options for filter selection, which relate to reference objects:

- **Show only "uses":**
  This setting only shows the used reference objects with the reference type "Uses", "Element -> Group", "Instance -> Type" and "Defines".
- **Show only "used by":**
  This setting shows all the reference objects with the reference type "Used by", "Group->Element", "Type->Instance", and "Defined by".
- **Show only local references:**
  This setting only shows local references which belong to a specific device (e.g. to CPU_1).

Sorting in the cross-reference list

The entries are displayed in alphabetical order, depending on the associated object, the associated device and the object type. Associated objects are displayed one after another.

The following sorting criteria are applied:

- For source objects: Sorting by object type, object name and number
- For referenced objects: Sorting by assignment to device, object name and number

---

Note

No sorting by name if more than 10,000 objects are displayed

Please note that no sorting by name can take place if more than 10,000 objects are displayed in the cross-reference list.

In this case, a banner text notifies you that no sorting is currently available for the displayed objects.

If you click the link in the banner text, you can enable sorting by name. A dialog box is then displayed which informs you about the progress of the sorting.

Display of columns in the cross-reference list

The columns displayed in the cross-reference list can be customized using the shortcut menu.

The shortcut menu offers the following options:

- Showing or hiding selected columns
- Displaying all existing columns
- Optimizing the column width for the selected column or for all columns.

Note

Showing or hiding selected columns

The columns "Object" and "Reference location" cannot be hidden in the cross-reference list.

You can save the settings you made by using the "Save window settings" button.

Adding new objects to the cross-reference list

You have the following options for inserting new source objects in an existing cross-reference list:

- By clicking the "Add new source object" button which is displayed at the end of the cross-reference list.

Note

Multiple insertion of identical objects

If you insert an identical object multiple times in the cross-reference list, the duplicates are automatically removed after the next update.

22.4 Displaying the cross-reference list

Introduction

Depending on the object selected in the project tree or in the "Instructions" task card, all the cross-references belonging to this source object are displayed.

Alternatively, you have the following option to open the cross-reference list:

- Select the "Cross-reference" command from the "Tools" menu.
- Select "Cross-references" in the shortcut menu.
- Click on the icon for cross-references in the toolbar.
- Select an object in the project tree and click "F11".
- Select an instruction with version identification in the "Instructions" task card and click "F11".

The cross-references for the relevant selected object and for all the lower-level objects below it are always displayed.

The display of the cross-references for an object is project- and device-specific. You can also have multiple cross-reference lists open at the same time.

Note

Displaying cross-references

- Which objects you can display in the cross-reference list depends on the installed products.
- If more than 50 objects were selected, these are collapsed in the cross-reference list.
- If fewer than 50 objects were selected, these are expanded in the cross-reference list.

Requirement

You have created a project which contains objects with references.

Displaying cross-references

Proceed as follows to display cross-references in the cross-reference list:

1. Select the required object and open the cross-references, e.g. using the shortcut menu.
2. The cross-reference list for the selected object is opened.
   - The relevant selected object is the "source object" and is at the top in the "Object" column.
   - In the column "Reference location", you can see the locations at which the objects shown in the cross-reference list can be used.
   - "Used by" and "Uses", for example, show the relations between the objects in the "Reference type" column.
3. To jump to the reference location of the relevant object, click on the corresponding link.

4. You can perform the following actions using the buttons in the toolbar:
   - Update cross-reference list
   - Defining filters for the cross-reference list
   - Collapse entries
   - Expand entries
   - Saving settings for the cross-reference list
   - Check if overlapping accesses are available in the currently opened cross-reference list.
   - Display of the overlapping accesses in a separate area below the project window
   - Adding new source objects
5. You can sort the entries in the "Object" column, for example, in ascending or descending order by clicking on the corresponding column header.

See also

General information about cross references (Page 7651)

Structure of the cross-reference list (Page 7653)

22.5 Sorting the cross-reference list

Introduction

Sorting for the displayed source and reference objects is executed by default for the cross-reference list.

This sorting is also displayed accordingly in the Inspector window.

Note

No sorting by name if more than 10,000 objects are displayed

Please note that no sorting by name is possible if more than 10,000 objects are displayed in the cross-reference list.

In this case, a banner text notifies you that no sorting is currently available for the displayed objects.

If you click the link in the banner text, you can switch on the sorting. A dialog box is then displayed which informs you about the progress of the sorting.

Sorting criteria for displaying cross-references

Sorting takes place according to the following criteria:

For source objects and assigned objects:

- By object type: The source objects are ordered by their object type and displayed together.
- By name: In alphanumeric order

For reference objects:

- By device: The reference objects for a device are displayed together.
- By object type: The reference objects are ordered by their object type and displayed together.
- By name: In alphanumeric order

For the reference location:

- By name: In alphanumeric order

Requirements

You have created a project which contains objects with references.

Sorting the cross-reference list

To sort the cross-reference list when more than 10,000 objects are displayed, follow these steps:

1. To sort the cross-reference list by name, click the link in the displayed banner text.
   Result: A dialog notifies you about the progress in sorting the cross-references.
2. The displayed objects are also shown sorted by name after the completion of the sorting process.

22.6 Filter options for cross-references

22.6.1 Introduction to filtering of cross-references

Introduction

You have the option of filtering the cross-reference list in order to simplify the search for specific cross-references and to structure it better.

The valid filter setting is displayed in the drop-down list with a yellow background.

Filter settings predefined by the system (referred to as system filters below) are available via the drop-down list in the toolbar of the cross-references.

In addition to the filter options provided by the system, you can also create user-defined filters in accordance with your own requirements.

The system filters are always available in the drop-down list for filter selection and cannot be deleted.

User-defined filters are displayed after creation in the drop-down list for filter selection below the system filters and can be modified, renamed and deleted as required.

In the figure below you can see the system filters that are always available as well as two user-defined filters "Filter_1" and "Filter_2".

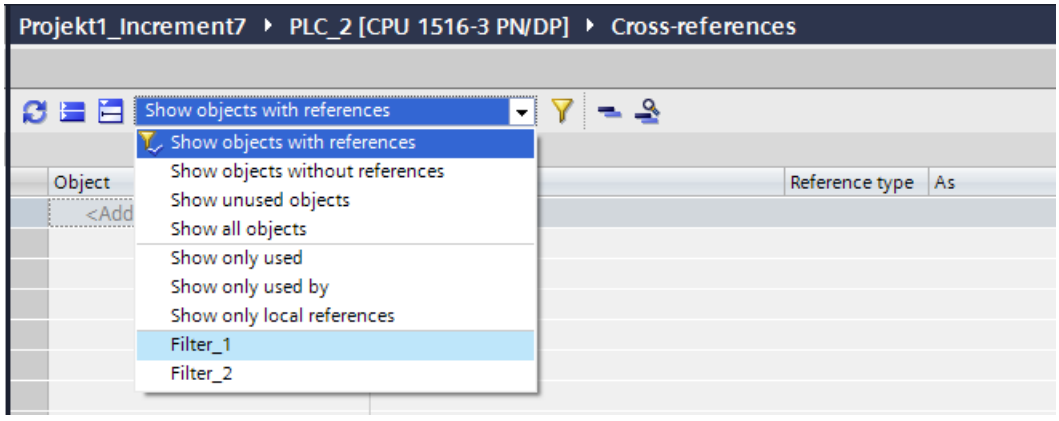

System-defined filter settings in the cross-reference list

The system filters shown in the upper area of the drop-down list are applicable for the displayed source objects; the filter objects in the lower area are used for the reference objects.

The following system filters are available:

- **Show objects with references:**
  This setting shows all the lower-level objects below the source object with existing references. It is the default setting whenever the cross-reference list is opened.
- **Show objects without references:**
  This setting shows all the lower-level objects below the source object without references. These non-referenced objects are assigned the entry "no references available" in the column "Reference location". Lower-level elements without references are also displayed if their parent objects contain references.
- **Show unused objects:**
  This setting shows all the lower-level but unused objects below the source object. These unused objects are assigned the entry "Object is not used" in the column "Reference location".
- **Show all objects:**
  This setting shows all the lower-level objects below the source object with or without references.
- **Show only "uses":**
  This setting only shows the used reference objects with the reference type "Uses", "Element -> Group", "Instance -> Type" and "Defines".
- **Show only "used by":**
  This setting shows all the reference objects with the reference type "Used by", "Group->Element", "Type->Instance", and "Defined by".
- **Show only local references:**
  This setting only shows local references which belong to a specific device (e.g. to CPU_1).

As soon as you have selected one of these system filters via the drop-down list, the associated cross-references are filtered and displayed in accordance with the selected criteria.

See also Filtering the cross-reference list (Page 7675).

User-defined filters in the cross-reference list

You can also create user-defined filters for the cross-references in addition to the system filters that are always available.

This is possible both in the cross-reference editor and in the Inspector window in the "Info" > "Cross-references" tab.

Open the cross-references and click the filter icon in the toolbar of the cross-reference editor.

In the subsequent dialog click "Create new filter" and enter the desired filter criteria. Click "Apply" to display the associated cross-references in accordance with the selected filter criteria.

User-defined filters are displayed in the drop-drop-down list below the system filters with the preset name, for example as "Filter_1".

In contrast to the system filters, user-defined filters can be renamed, modified and deleted freely.

Invalid filters are displayed with a red background after being opened. These filters cannot be applied.

See also Creating user-defined filters for cross-references (Page 7667).

22.6.2 Structure of the filter dialog

Introduction

You can also create user-defined filters to display the cross-references.

Below you can find information on the structure of the filter dialog and the symbols used in the filter dialog.

Structure of the filter dialog

The filter dialog has the following structure:

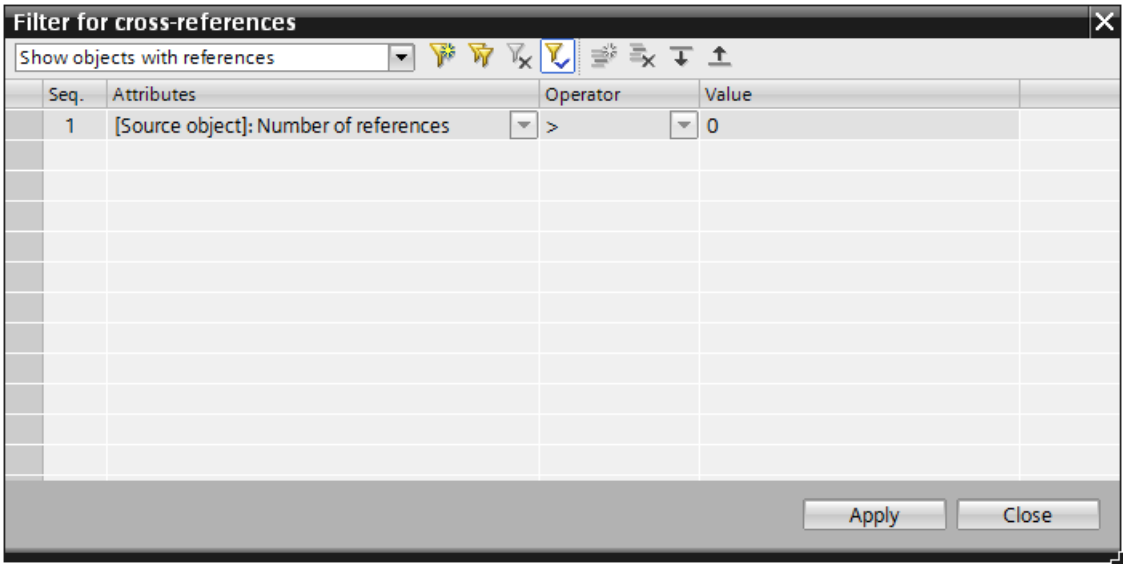

The filter dialog contains:

- The drop-down list for selecting existing filters.
  The filter displayed at the top in the drop-down list is the system filter taken from the open cross-reference list.
- The icons for creating, duplicating and deleting filters.
- The icon for specifying a filter as the default.
- The icons for adding, deleting and moving filter criteria.
- The columns "Seq." (sequence), "Attributes", "Operator " and "Value".
  The "Seq." column shows the number of the respective filter criteria.
  In the remaining columns you define the criteria required for new filters.
- "Apply" button: When you exit the filter dialog with "Apply", the changes are saved in the current user profile and are applied in the lower-level editor in the cross-reference list.
- "Close" button: When you exit the filter dialog with "Close", the changes are applied in the current user profile but are not applied in the editor.

Icons in the filter dialog

The filter dialog shows the following icons in the toolbar:

Icon	Meaning	Action
	Create new filter	Creates a new user-defined filter.
	Duplicate selected filter	Duplicates the selected filter so that existing filter criteria can be applied and processed further.
	Delete filter	Deletes the selected user-defined filter. The filter cannot be restored after it has been deleted! Note: System filters cannot be deleted!
	Apply filter as default setting	Applies the selected filter as default setting. This filter is used when the cross-references are opened again.
	Insert new criterion	Inserts a new filter criterion.
	Delete criterion	Deletes the selected filter criterion.
	Move criterion up	Moves the selected filter criterion one line up.
	Move criterion down	Moves the selected filter criterion one line down.

22.6.3 Attributes, operators and values in the filter dialog

Core statement

You can also create user-defined filters to display the cross-references.

Below you can find information on attributes, operators and values in the filter dialog, as well as on the criteria you can use.

Attributes, operators and values in the filter dialog

To create a valid filter you have to define the associated attributes, operators and values.

When you click in the column "Attribute" or "Operator", a drop-down list is displayed from which you can select the desired entries.

The "Value" column is displayed with a red background when you have selected an attribute and an operator, but have not yet entered a valid value.

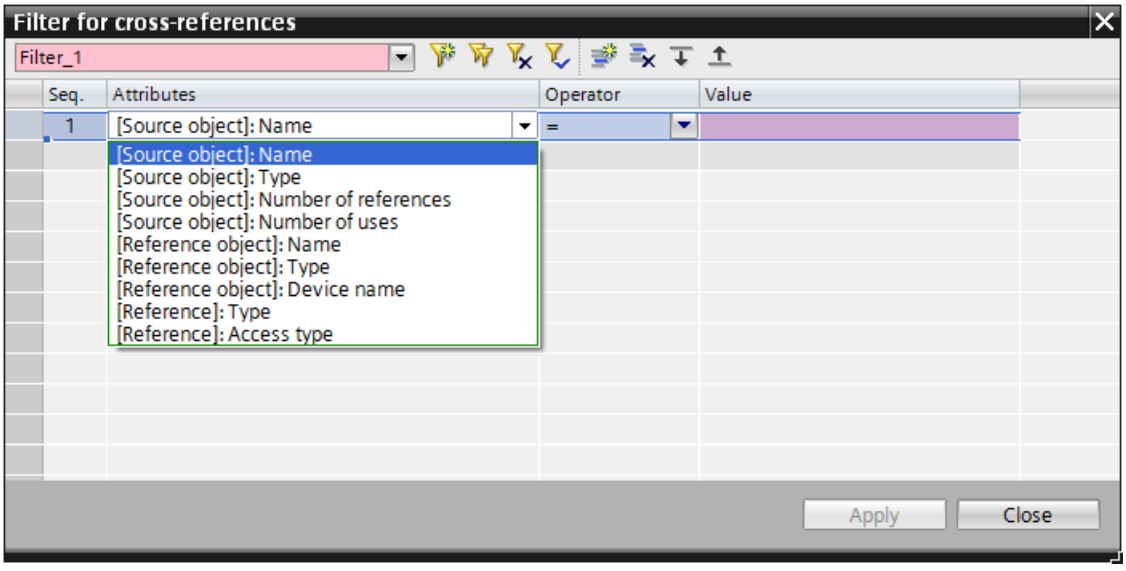

Rules for criteria in the filter dialog

The rules specified below have to be observed when creating a new user-defined filter:

- Filter criteria can be defined for source objects, for reference objects and for references.
- Source objects are not affected when a criterion only applies to one reference object or one reference.
- The filter criteria are applied in the specified order.

  Note

  Note that the result of the first criterion is used as a specification for the second criterion, etc. Therefore the sequence of the criteria is also decisive.

  You can move the existing criteria "up" or "down" by using the corresponding symbols in the toolbar. To do so, select the criterion and then click the desired icon.

- Individual criteria are added in sequence, depending on the corresponding sequence number.
  Example: The first criteria has the sequence number "1", the second criteria "2" etc.
- The permissible operators for the respective attribute can be selected from the drop-down list in the "Operator" column. The drop-down list is displayed as soon as you click the corresponding line in the filter dialog.
- If a filter is invalid because no valid value has been entered as a criterion yet, for example, the "Value" column is displayed with a red background. Such an invalid filter can be stored but cannot be applied. In this case, you can only exit the filter dialog by using "Close".
- When a criterion applies to a structured source object, this source object is displayed with all the parents and child objects and their references.
- The parent objects of the structured source object are, if they are not affected, only displayed because of the hierarchy and thus without references and in gray.

- When a criterion applies to a structured reference object, this reference object is displayed with all its lower-level reference locations.
- When a criterion only applies to elements of a structured reference object, only the respective lower-level reference location is displayed.

22.6.4 Creating user-defined filters for cross-references

Introduction

You can also create user-defined filters to display the cross-references.

After user-defined filters have been created and stored, they can also be selected via the drop-down list in the toolbar.

Newly created user-defined filters are available immediately in all currently open cross-reference editors.

This allows you to reduce the number of objects displayed in the cross-reference list and to search specifically for certain objects.

Note

Creating user-defined filters

You can create user-defined filters in all cross-reference editors as well as in the Inspector window under "Info > Cross-references".

The procedure is identical in all cross-reference editors.

Requirement for the creation of a user-defined filter

You have created a project which contains objects with references.

Creating a user-defined filter

To create user-defined filters for the cross-reference list, follow these steps:

1. Select the program blocks folder and open the cross-reference list by using the "Cross-references" shortcut menu command.
2. Click the filter icon in the toolbar of the open cross-reference list.
   The filter dialog opens and shows the criteria that meet the selected system filter.
   Example: You have selected the system filter "Show objects with references" before opening the filter dialog in the drop-down list. You now see the Attributes, Operators and Values defined for this system filter in the filter dialog.
3. Click the "Create new filter" icon.
   The newly defined filter name "Filter_1" is displayed in the drop-down list. The icons for adding new filter criteria are activated.

4. Define the associated attributes, operators and values.
   Permissible attributes and operators can be selected from drop-down lists. The drop-down lists are displayed as soon as you click in the corresponding line in the filter dialog.
5. If you need several criteria, click the "Create new criterion" icon:

   Note

   Note that the result of the first criterion is used as a specification for the second criterion, etc. Therefore the sequence of the criteria is also decisive.

   You can move the existing criteria "up" or "down" by using the corresponding symbols in the toolbar. To do so, select the criterion and then click the desired icon.

6. Enter all criteria required for the new filter.
7. Assign a name for the new filter if you do not want to use the default name "Filter_1". You can enter the desired name directly in the drop-drop-down list.

   Note

   If you enter a name in the drop-down list that already exists as a filter name, this filter is applied in the cross-reference editor.

   The new filter is saved under the name suggested by the system (for example, "Filter_10") and can be selected via the drop-down list in the editor.

8. Click "Close" if you want to store the new filter settings, but do not want them displayed immediately in the lower-level cross-reference editor.
9. Click "Apply" to display the new filter settings immediately in all lower-level cross-reference editors.

Result

When the filter dialog is exited with "Close", the new filter is stored but not applied. The new filter is not used until you select it from the drop-down list in the cross-reference editor.

When the filter dialog is exited with "Apply", the new filter is stored and is applied in the open cross-reference editors.

22.6.5 Editing user-defined filters

Introduction

User-defined filters for the display of cross-references can be selected after they have been created via the drop-down list in the toolbar of the cross-reference editor.

In contrast to system filters, user-defined filters can be edited and renamed.

The toolbar contains the icons for creating and editing filters and filter criteria.

Note

Renaming filters in the cross-references

Only user-defined filters can be renamed; system filters cannot be renamed and cannot be deleted either.

Requirement for editing a user-defined filter

You have created a project which contains objects with references.

At least one user-defined filter has been created for the cross-references.

For additional details, see: Creating user-defined filters for cross-references (Page 7667)

Editing and renaming user-defined filters

To edit user-defined filters for the cross-reference list, follow these steps:

1. Select the program blocks folder and open the cross-reference list by using the "Cross-references" shortcut menu command.
2. Select the user-defined filter that you want to edit from the drop-down list.
3. Click the filter icon in the toolbar of the open cross-reference list.
   The filter dialog opens and shows the criteria that meet the selected filter.
4. Click the "Insert new criterion" icon in the toolbar.
5. Define the associated attributes, operators and values.
   Permissible attributes and operators can be selected from the drop-down lists. The drop-down lists are displayed as soon as you click in the corresponding line in the filter dialog.
6. Enter all criteria required for the new filter.

   Note

   Note that the result of the first criterion is used as a specification for the second criterion, etc. Therefore the sequence of the criteria is also decisive.

   You can move the existing criteria "up" or "down" by using the corresponding symbols in the toolbar. To do so, select the criterion and then click the desired icon.

7. Click "Delete criterion" if you want to delete a filter criterion.
8. If you wish, assign a name for the new filter. You can enter the desired name directly in the drop-drop-down list.

   Note

   If you enter a name in the drop-down list that already exists as a filter name, this filter is applied in the cross-reference editor.

   The new filter is saved under the name suggested by the system (for example, "Filter_10") and can be selected via the drop-down list in the editor.

9. Click "Close" if you want to store the new filter settings, but do not want them applied immediately in the lower-level cross-reference editor.
10. Click "Apply" to apply the new filter settings immediately in all lower-level cross-reference editors.

Result

When the filter dialog is exited with "Close", the modified filter is stored but not applied. The filter is not used until you click "Update" in the cross-reference editor.

When the filter dialog is exited with "Apply", the new filter is stored and is applied in the open cross-reference editors.

Invalid user-defined filters

If a user-defined filter becomes invalid through editing, for example, because a value has not been entered yet, the corresponding line in the filter dialog has a red background. This has the following effects:

- The filter cannot be used because the "Apply" button is not active.
- After the filter dialog has been closed, no changes are displayed in the editor.
- If you click "Update" in the cross-reference editor, the filter defined as the default setting is applied for the display of the cross-references.
- If no user-defined filter is defined as the default setting, the system filter "Show objects with references" is applied.

See also

Creating user-defined filters for cross-references (Page 7667)

22.6.6 Deleting user-defined filters

Introduction

User-defined filters for the display of cross-references can be selected after they have been created via the drop-down list in the toolbar of the cross-reference editor.

In contrast to system filters, user-defined filters can also be deleted.

The toolbar contains the icons for deleting filters and filter criteria.

Note

Renaming and deleting filters in the cross-references

Only user-defined filters can be deleted; system filters cannot be renamed and cannot be deleted either.

Requirement for deleting a user-defined filter

You have created a project which contains objects with references.

At least one user-defined filter has been created for the cross-references.

For more details, see: Creating user-defined filters for cross-references (Page 7667)

Deleting user-defined filters

To delete user-defined filters for the cross-reference list, follow these steps:

1. Select the program blocks folder and open the cross-reference list by using the "Cross-references" shortcut menu command.
2. Select the user-defined filter that you want to delete from the drop-down list.
3. Click the filter icon in the toolbar of the open cross-reference list.
   The filter dialog opens and shows the criteria that meet the selected filter.

   Note

   Keep in mind that the filter is deleted irrevocably without any further prompts as soon as the "Delete filter" icon is clicked.

4. Click the "Delete filter" icon in the toolbar if you want to delete the selected filter irrevocably.
5. Exit the dialog with "Close" or "Apply".

Result

The selected filter was deleted and is no longer shown in the drop-down list.

22.6.7 Exporting and importing user-defined filters

Introduction

After they have been created, user-defined filters for the display of cross-references can be exported and then imported again.

This has the advantage that you can also pass a filter on to other users when it has been created.

You use the functionality of the TIA Portal for the export and import of user-defined filters.

Requirement for exporting a user-defined filter

You have created a project which contains objects with references.

At least one user-defined filter has been created for the cross-references.

For more details, see: Creating user-defined filters for cross-references (Page 7667)

Exporting user-defined filters

To export user-defined filters, follow the steps below:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Select the "General" entry in the area navigation.
3. In the "Import/export settings" area, click on the "Export settings ..." button.
   The "Export settings" dialog opens.
4. Click "Deselect all" to select only the desired filter settings for the export.
5. Select the desired filter for the export under "Cross-references > User-defined filters".
   If you want to accept the settings selected from previous export operations, activate the check box "Use selection from a previous export file". Enter the file in the field.
6. Enter a file name for the initialization file in the "File name" box.
7. Click "Export" to export the selected filters.

Result

Settings are exported to a file with the extension ".tps14".

Importing user-defined filters

To import user-defined filters, follow the steps below:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Select the "General" entry in the area navigation.
3. Click the "Import settings ..." button in the "Import/export settings" area.
   The "Export settings" dialog opens.
4. Select the desired file with the extension ".tps14".

   Note

   If a filter with an identical name already exists, it is overwritten by the import and is no longer available.

5. Click "Open" to import the file.

Result

The selected file is imported and, after a restart of the TIA Portal, the user-defined settings are applied to the cross-references.

22.6.8 Setting a filter as the default setting

Introduction

You can use various system-defined filter settings (called system filters below), which can be selected via the drop-down list in the toolbar, to display the cross-references. In addition, you can also create user-defined filters in accordance with your own requirements.

You have the option of setting any filter as the default setting for the display of cross-references.

If a filter is set as the default setting, this filter is always applied when the cross-references are opened.

Rules for the default setting of filters

The following rules apply for the default setting of filters:

- Only one filter can be selected as the default setting.
- The default filter is identified with the corresponding icon.
- Invalid filters can also be set as the default setting, but they cannot be applied. Instead of the invalid filter, however, the system filter "Show objects with references" is applied automatically when the cross-reference list is opened.

Requirements

You have created a project which contains objects with references.

Setting a filter as the default setting

To set a filter as the default setting for the display of cross-references, follow these steps:

1. Select the program blocks folder and open the cross-reference list by using the "Cross-references" shortcut menu command.
2. Select the filter that you want to set as the default from the drop-down list.
3. Click the filter icon in the toolbar of the open cross-reference list.
   The filter dialog opens and shows the criteria that meet the selected filter.
4. Click the Apply filter as default setting icon in the toolbar.
5. Click "Close" if you want to store the new default setting, but do not want it applied immediately in the lower-level cross-reference editor.
6. Click "Apply" to apply the new filter settings immediately in all lower-level cross-reference editors.

Result

When the filter dialog is exited with "Close", the new default setting is stored but not applied. The new default setting is not used until you click "Update" in the cross-reference editor.

When the filter dialog is exited with "Apply", the new default setting is stored and is applied in the open cross-reference editors.

See also

Creating user-defined filters for cross-references (Page 7667)

22.6.9 Duplicating a filter

Introduction

You can use various system-defined filter settings (called system filters below), which can be selected via the drop-down list in the toolbar, to display the cross-references. In addition, you can also create user-defined filters in accordance with your own requirements.

You can duplicate an existing filter.

This facilitates the adding of further filter criteria.

Requirements

You have created a project which contains objects with references.

Duplicating a filter

Proceed as follows to duplicate a filter:

1. Select the program blocks folder and open the cross-reference list by using the "Cross-references" shortcut menu command.
2. Select the filter that you want to duplicate from the drop-down list.
3. Click the filter icon in the toolbar of the open cross-reference list.
   The filter dialog opens and shows the criteria that meet the selected filter.
4. On the toolbar, click the "Duplicate selected filter" icon.
   The filter is then duplicated and is displayed in the drop-down list under the new name "Filter_x".
5. Enter additional filter criteria as required or change the existing attributes, operators or values.

Result

When the filter dialog is exited, the duplicated filter is stored and is applied in the open cross-reference editors.

22.6.10 Filtering the cross-reference list

Introduction

You can use various system-defined filter settings (called system filters below), which can be selected via the drop-down list in the toolbar, to display the cross-references.

In addition to the filter options already offered, you can also create user-defined filters in accordance with your own requirements.

The system filters are always available in the drop-down list for filter selection and cannot be deleted.

User-defined filters are displayed after creation in the drop-down list for filter selection below the system filters and can be deleted.

See also: Creating user-defined filters for cross-references (Page 7667)

Requirements

You have created a project which contains objects with references.

Filtering the cross-reference list

To filter the cross-reference list, follow these steps:

1. Select the required source object and open the associated cross-reference list using the "Cross-references" shortcut menu command.
2. Alternatively, you can open the cross-references by selecting objects in the project tree for which cross-references exist, for example:
   - A device
   - The "Program blocks" folder
   - A block
3. Then click the "Cross-references" icon in the toolbar of the TIA Portal.
4. In the opened cross-reference list, select the required filter setting via the drop-down list.

Result

The selected filter setting is applied for the objects displayed in the lower-level cross-reference editors.

22.7 Printing the cross-reference list

Introduction

In the print preview, you can view and print out the project documentation contents shown in the cross-reference list.

The "Project" > "Print preview" and "Project" > "Print" commands are available for this purpose in the TIA Portal.

The following rules apply to printing out:

- The cross-reference list is always expanded and printed with all existing columns (including the hidden columns).
- The set filter criteria are applied at the same time.
- The column width is automatically optimized.
- Any overlapping accesses displayed are not printed.
- Regardless of which object is selected in the cross-reference list, the printout always starts with the displayed source object.

Note

Printing an empty cross-reference list

If the cross-reference list selected for printing contains no entries, an empty cross-reference list is printed out.

A cross-reference list which is printed out without contents contains the note "There are no entries in cross reference list".

Requirements

You have created a project which contains objects with references.

Printing the cross-reference list

To print the cross-reference list, follow these steps:

1. Select an object from the cross-reference list and start the print preview via the "Project" > "Print preview" command.
2. Make sure that you print out the required cross-reference list and select the required options.
3. Start the printout using the "Project" > "Print" command.
4. In the following dialog box, select the required print options and click "OK".

Result

The selected cross-reference list is printed out with the set print options.

22.8 Adding a new object to the cross-reference list

Introduction

Depending on the selected object, all cross-references belonging to this source object are displayed when the cross-reference list is opened.

You have the following options for inserting new source objects in an existing cross-reference list:

- By clicking the "Add new source object" button which is displayed at the end of the cross-reference list.

Requirement

You have created a project which contains objects with references.

Adding a new object to the cross-reference list

To add new objects to the cross-reference list, follow these steps:

1. Open an existing cross-reference list.
2. Click the "Add new source object" button and enter the required object. When an existing object is inserted, you are supported by the intelligent pre-selection.

Note

Multiple insertion of identical objects

If you insert an identical object multiple times in the cross-reference list, the duplicates are automatically removed after the next update.

Note

Functionality for newly added source objects

Note that the "Undo and redo operations" functionality is not available for newly added source objects.

Result

The newly added object is displayed as last object in the cross-reference list.

The current filter selection is applied to the newly added object only after the next update.

22.9 Show overlapping accesses in the cross-reference list

Introduction

The overlapping access to a selected object can be displayed in a second window in the lower part of the cross-reference list.

The display of the overlapping access cannot be saved and is no longer available after the cross-reference list is opened again.

Definition

Overlapping access for tags and structured tags is determined based on the absolute addresses.

We refer to "overlapping access" if the address for a tag or structured tag is partly or completely in the address space of the associated source object.

Note

Displaying "overlapping access"

Overlapping access is only displayed for a single selected source object or lower-level source object in the cross-reference list.

If several objects are selected in the cross-references shown, no overlapping access is displayed.

Example

The following graphic shows the cross-reference list with the display of the overlapping access for "Tag_1":

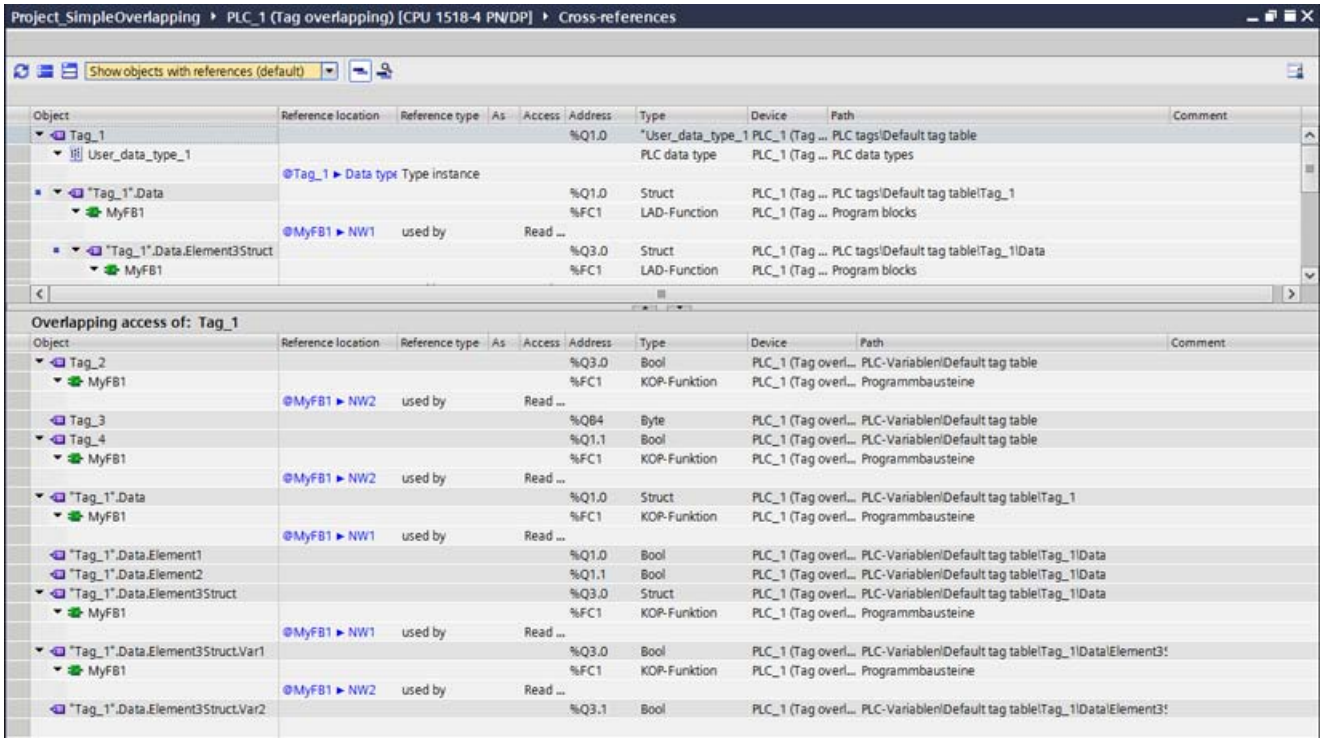

Requirement

A project has been created which contains objects with overlapping access.

Showing overlapping access in the cross-reference list

Proceed as follows to display overlapping access in the cross-reference list:

1. Open the cross-reference list and click the "Check overlapping accesses" button in the toolbar.
   Result: All objects that have overlapping access are marked with the corresponding overlapping access symbol in the "Object" column.
2. In the upper part of the cross-reference list, select a marked object and click the "Show overlapping accesses" button.
   Result: The overlapping access to the selected object is displayed in a second window with the header "Overlapping accesses of: <ObjectName>".

Note

Display of the overlapping access

The display of the overlapping access cannot be saved and is no longer shown after the cross-reference list is opened again.

To show the overlapping access again, perform the steps described above.

22.10 Restoring cross-references after project upgrade.

Introduction

After a project upgrade to a higher TIA Portal version, you have the option of recreating the associated cross-reference information for the updated project.

This happens automatically at the end of the upgrade process. The current project is opened after this and the cross-references are then available again.

You are notified if the cross-reference information is inconsistent or incomplete. A banner text with a link to additional information is displayed when the cross-reference list is opened.

In this case, you can reorganize the cross-reference information yourself.

Requirements

You have performed a project upgrade and received the message that the cross-reference information is inconsistent or incomplete.

Recreate the cross-reference information

To create new cross-references, follow these steps:

1. To restore the cross-references, click the link in the displayed message.
   Alternatively, you can also call and execute the command via "Options" > "Settings" > "General" > "Cross-references".
   The settings for the TIA Portal then open under "Cross-references".
2. Click the displayed button "Recreate the cross-reference information" to start the process.
3. When the restoration has been completed, check the message in the Inspector window to see if the process was successful.
   If the process was successful, you receive a message which you must confirm with "OK".
   If the process was not successful, you receive an error message.
4. In the cross-reference list, click the "Update" button to close the displayed banner again.

Result

The new cross-reference information for the selected project was created.

22.11 Displaying and editing cross-references in the Inspector window

Introduction

The Inspector window displays cross-reference information about an object you have selected in the "Info > Cross-references" tab.

If you select an object in any editor in the TIA Portal or in the "Instructions" or "Extended Instructions" task card, the associated references to the selected object are shown in this tab.

Note

Displaying instructions

- As of TIA Portal V15 instructions with version identification can also be displayed with the cross-references and in the Inspector window.
- Instructions without version identification are **not** displayed.

Structure

The Inspector window displays the cross-reference information in tabular format.

The cross-reference list has the following structure:

Column	Content/meaning
Object	Displays the name of the source object which was selected when the cross-reference list was opened, with all lower-level objects and the associated reference objects.
Reference location	Shows the reference location of the objects, e.g. a network.
Reference type	Shows the relationship between the source object and the referenced objects: • "Uses": The source object uses this object. • "Used by": The source object is used by this object. • "Type -> Instance": The source object is a type of the referenced object. • "Instance -> Type": The source object is an instance of the referenced object. • "Group -> Element": The source object is a group of the referenced object. • "Element -> Group": The source object is an element of the referenced object. • "Defines": The source object defines the referenced object. • "Defined by": The source object is defined by the referenced object.
As	Shows additional information about objects, for example that a tag is used by several devices.
Access	Shows the type of access, e.g. whether access to the operand is read access (R) and/or write access (W).
Address	Shows the address of the relevant object.

Column	Content/meaning
Type	Shows the type and the language used to create the object.
Device	Shows the associated device name, e.g. "CPU_1".
Path	Displays the path of the object in the project tree with specification of folders and groups.
Comment	Displays comments on individual objects, if available.

Displaying cross-references in the Inspector window

To display cross-reference information in the Inspector window, follow these steps:

1. Select the required object (e.g. in the project tree or in the program editor) and click "Show cross reference list" in the shortcut menu.
   Result: The references to the selected object are displayed in the Inspector window in the "Info" > "Cross-references" tab.

Note

Please note that only the cross-reference information for supported objects can be shown in the Inspector window

If, for example, you have selected a folder, you will not receive any cross-reference information for this.

Editing cross-references in the Inspector window

You also have the option of editing and filtering cross-reference information in the Inspector window.

The procedure for filtering cross-reference information is identical to the procedure in the cross-reference editor.

See also:

Creating user-defined filters for cross-references (Page 7667)

Deleting user-defined filters (Page 7670)

Setting a filter as the default setting (Page 7673)

Filtering the cross-reference list (Page 7675)

Testing the user program 23

23.1 Basics of testing the user program

Introduction

You have the option of testing the running of your user program on the device. You can then monitor signal states and values of tags and can assign values to tags to simulate certain situations in the running of the program.

Requirement

There must be an executable program loaded on the device.

Test options

The following test options are available:

- Testing with program status
  The program status allows you to monitor the running of the program. You can display the values of operands and the results of logic operations (RLO) allowing you to recognize and fix logical errors in your program. You can test either with program status or in single step mode within a S7-300/400 CPU. You cannot, however, use both test options at the same time.
- Testing with breakpoints
  You can test blocks you created in STL or SCL with breakpoints. You do this by setting breakpoints in the program code at which program execution stops. You can then continue to run the program one step at a time. You can test either with program status or in single step mode within a S7-300/400 CPU. You cannot, however, use both test options at the same time.
  Be sure to read the following before you start testing with breakpoints:
  Safety information for testing with breakpoints (Page 7718)
  Security information for testing with breakpoints (Page 7717)
- Testing with the watch table
  With the watch table, you can monitor and modify the current values of individual tags in the user program or on a CPU. You can assign values to individual tags for testing and run the program in a variety of different situations. You can also assign fixed values to the I/O outputs of a CPU in STOP mode, for example to check the wiring.
- Testing with the force table
  With the force table, you can monitor and force the current values of individual tags in the user program or on a CPU. When you force, you overwrite individual tags with specified values. This allows you to test your user program and run through various situations.
  When forcing, make sure that you keep to the necessary Safety precautions when forcing tags (Page 7786)!

See also

Introduction to testing with program status (Page 7687)
Introduction to testing with breakpoints (Page 7716)
Security information for testing with breakpoints (Page 7717)
Safety information for testing with breakpoints (Page 7718)
Introduction to testing with the watch table (Page 7739)
Introduction for testing with the force table (Page 7768)

23.2 Testing with program status

23.2.1 Introduction to testing with program status

Program status function

You can monitor the execution of the program by displaying the program status. This shows you the values of the individual operands and their results of logic operations. This allows you to check whether the components of the automation system are being correctly controlled.

The display of the program execution in the program status can differ slightly, depending on the CPU series used.

Testing with program status for S7-300/400

During testing with program status, the CPU cycle time can become extended because the recording of the data affects the duration of the programmed instructions.

During execution of the following test functions, an alarm is displayed once for each online session indicating that there is a risk of a timeout:

- During testing with call conditions
- During testing with breakpoints

You can only perform these test functions after you have acknowledged the alarm.

Note

With older CPUs from the S7-300/400 CPU series, you will need to change the operating response using the hardware configuration and then load the hardware configuration to the device. You have the option to set the "Process operation" or "Test mode" operating response.

Testing with program status for S7-1200/1500

When you run the "Testing with program status" function with a CPU from the S7-1500 series, the cycle time can increase significantly when monitoring loops due to the fast program execution. The CPU may go to STOP, especially when it exceeds the configured cycle clock for interrupt OBs or the configured maximum cycle time.

Note

To avoid a possible STOP in the CPU, ensure that no programmed loops are displayed in the active monitoring window during "Testing with program status". Alternatively, you can also increase the maximum permissible cycle time of the CPU.

Restrictions with the "Program status" function

The monitoring of loops can increase the cycle time significantly, depending in each case on the number of tags to be monitored and on the actual number of loops processed.

To ensure that the cycle time is influenced as little as possible, the "Program status" function is restricted as described below:

For CPUs from the S7-300/400 series:

- The status display of a programmed loop is stopped at the return point.

For CPUs from the S7-1200/1500 series:

- The monitoring of loops is deactivated by default. If required, monitoring can be activated via the shortcut menu.
- If the monitoring of loops is enabled with an S7-1200 CPU, the program status for the area is not displayed after a loop that is processed multiple times. In this case, move the visible editor window to the area after the loop so that the monitor values are displayed again.
- The first occurrence of an instruction is always observed for a CPU S7-1200 as of V4.2 and for a CPU S7-1500.

WARNING

Testing with program status

A test with the "Program status" function can cause serious damage to property or injury to persons if malfunctions or program errors occur.

Make sure that no dangerous situations can arise before you conduct a test with the "Program status" function.

Recommended procedure if there is a risk of a timeout

If there is a risk of a timeout during monitoring with program status, you are alerted by the system with a corresponding alarm.

To continue monitoring, you have to acknowledge the alarm with "Yes".

In this case, you also have the following option to avoid exceeding the maximum permissible cycle time during testing with program status:

- Reduce the size of the editor window so that fewer tags are displayed and thus fewer tags are monitored.
- Monitor the required tags within the watch table.

23.2.2 Setting the call environment

23.2.2.1 Basics of the call environment

Function

Depending on the CPU used, you can define the call environment for monitoring blocks and for breakpoints (not with S7-1500). With this, you specify the conditions for recording the program status of a block or for stopping program execution at a breakpoint.

Valid for S7-1500er CPUs as of firmware version V2.5:

Call environment for monitoring blocks:

- As soon as a breakpoint is enabled in a block, it is **not** possible to configure a call environment for monitoring.
- In the other direction, a breakpoint in a block **cannot** be enabled if a call environment has already been configured.
- If you attempt to use both functionalities, you are shown an error message with information about how to proceed.

Specification of the call environment for block monitoring

Alternatively, you can activate one of the following options to specify the call environment:

- No condition defined
  If no other option was selected this option is the default.
- Instance data block
  The program status of a function block will then only be recorded when you call the function block with the selected instance data block.
- Call path
  The program status of a block will then only be recorded when you call the block with a specific block or from a specific path.
- Manually adapted call path
  You can enter the desired call environment manually in this field. The "Transfer to 'adjusted manually'" button is used to transfer the content selected under "Call environment", which you can edit further if required.
  The program status of a block will then only be recorded when you call the block with a specific block or from a specific path.

Specification of the call environment for breakpoints

When working with a CPU S7-300/400, you can define a separate call condition for each breakpoint.

When working with a CPU S7-1500 as of V2.5, you cannot specify a call environment for breakpoints.

If you do not define a call environment, the program status of any block call is recorded within the call structure and the program execution is always interrupted at the given breakpoint. If possible, set a call environment to display the program status for a particular call.

See also

Setting the call environment for block monitoring (Page 7690)

23.2.2.2 Setting the call environment for block monitoring

By setting the call environment, you can specify when the program status of a block is recorded.

The section "Setting the call environment for breakpoints (Page 7722)" describes how to set the call environment for breakpoints.

Requirement

- The block is open.

Specifying the call environment

To set the call environment, follow these steps:

1. Open the "Testing" task card.
2. Click "Change" in the "Call environment" pane.
   The "Call environment of the block" dialog opens.
3. Select the condition you want to apply.
   See also: Basics of the call environment (Page 7689)
4. Confirm your selection with "OK".

Result

The selected call environment is displayed in the "Testing" task card within the "Call environment" pane. The program status is now carried out in accordance with the set call environment.

Changing the call environment

Proceed as follows to change the call environment:

1. Open the "Testing" task card.
   If a call environment is already set, this is displayed in the "Call environment" pane.
2. Click "Change" in the "Call environment" pane.
   The "Call environment of the block" dialog opens.
3. Select the condition you want to apply.
   See also: Basics of the call environment (Page 7689)
4. Confirm your selection with "OK".

Result

The selected call environment is displayed in the "Testing" task card within the "Call environment" pane. The program status is now carried out in accordance with the set call environment.

See also

Basics of the call environment (Page 7689)

Introduction to testing with program status (Page 7687)

23.2.3 Switching test with program status on/off

You can monitor all blocks by switching on the program status of the block. This function is available to you for all code blocks, regardless of the programming language used. For blocks that were programmed with LAD, FBD or SCL you can also enable the program status from a specific position or for a specific selection. You can switch on the program status for an open block directly, or open a block from the calling block and view the program status.

A tooltip shows you whether or not data about the program status is currently being received. To do his, place the mouse over the blue dashed "Indicator" in the upper right corner of the workspace.

Note

Note the following:

- The resources for testing with program status are limited. If there are not enough resources for an additional test, the tests that are already running are terminated.
- It is possible that another user can carry out a loading process on the selected CPU through joint parallel working on a CPU. It is therefore possible in the following cases that you can either not start the test with program status or that a running test is terminated:
  - Through the loading process the block for which you want to start or already have carried out the test with program status is loaded.
  - You use an instance data block as the call environment for the test with program status and the block changes structurally through the loading process, e.g. by renumbering.
  - You use a call path as the test condition for the test with program status and a block within the call path changes through the loading process.

  If a running test is terminated, a corresponding message is displayed in the Inspector window.

Requirement

- Code block: The code of the offline block is identical with the code of the online block. In this case the "Code time stamps" of the blocks are identical.
- Data block: The structure of the offline block is identical with the structure of the online block. In this case the "Interface time stamps" of the blocks are identical.

Switching the program status on or off directly in the block

To switch the program status for a block on or off directly in the block, follow these steps:

1. Open the block for which you wish to switch on the program status.
2. Click "Monitoring on/off" in the toolbar.
   If you have not already established an online connection, the "Go online" dialog opens. In this dialog, you can establish an online connection.
   See also: Establishing and terminating an online connection
3. Click "Monitoring on/off" in the toolbar again to switch the program status off.

Switching on or off program status starting at a specific point in a network

To start the program status for LAD and FBD at a specific point, follow these steps:

1. Open the block for which you wish to switch on the program status.
2. Click "Monitoring on/off" in the toolbar.
3. Right-click on the tag you want program status to start from.
4. Select "Modify > "Monitor from here" in the shortcut menu.
5. Click "Monitoring on/off" in the toolbar again to switch the program status off.

To start the program status in SCL at a specific point, follow these steps:

1. Open the block for which you wish to switch on the program status.
2. Click "Monitoring on/off" in the toolbar.
3. Select the tag from which you want to start the program status.
4. After this, click "Display program status from the selected line" in the toolbar.
5. Click "Monitoring on/off" in the toolbar again to switch the program status off.
6. To make matters clearer, you can use the shortcut menu commands "Open" and "Close" or "Open all" and "Close all" to expand or collapse individual or all closed code sections within the block.

Switching on or off program status for selected tags

To start the program status for LAD and FBD for selected tags, follow these steps:

1. Open the block for which you wish to switch on the program status.
2. Click "Monitoring on/off" in the toolbar.
3. Select the tags for which you want to start the program status.
4. Select "Modify > Monitor selection" in the shortcut menu.
5. Click "Monitoring on/off" in the toolbar again to switch the program status off.

Switching on program status from the calling block

To switch on the program status for a block from the calling block (e.g. OB1), follow these steps:

1. Open the calling block.
2. Right-click on the block call.
3. Select the command "Open and monitor" in the shortcut menu.
   The block will open in the program editor. An online connection is established and the program status is displayed. The calling block is set in the call path.

Result

If you enable the display of the program status, an online connection is established and the program status is displayed. When you turn off the display of the program status, you can terminate the online connection at the same time.

The call path of the block is shown under the block interface. If necessary, you can change the call environment in the "Call environment" section at the right-hand edge at "Test" and "Options". In the case of CPUs of the S7-1200/1500 series the "Call hierarchy" is displayed additionally on the right-hand edge. You can open the calling block by clicking the link.

See also

Setting the call environment (Page 7689)

23.2.4 Monitoring loops

Introduction to monitoring loops

If you display the program status, you can monitor the execution of the program with a block. This provides you with an overview of the values of the individual operands and the results of the logic operations and you can check whether the components of the automation system are correctly controlled.

If the values are shown in a lighter, less saturated font, these values do not come from the current cycle.

Note

The monitoring of loops can increase the cycle time of the CPU significantly, depending in each case on the number of tags to be monitored and on the actual number of loops processed.

Special rules apply to loop monitoring, depending on the specific CPU and the programming language that are used.

Rules for loop monitoring (S7-300/400)

To ensure that the cycle time is influenced as little as possible, the "Program status" function is restricted as follows during the monitoring of loops with CPU from the S7-300/400 family in all programming languages:

- The status display of a programmed loop is stopped at the return point.

Rules for loop monitoring in STL (S7-1500)

The display of the program status within programmed loops is disabled by default during the monitoring of loops in STL. To avoid any lengthening of the cycle time, no values are displayed for the tags within loops between the "Label" and the associated "Loop" instruction.

If you are monitoring an STL block with programmed loops, a tooltip informs you that the monitoring of loops is currently disabled and can be enabled if required via the shortcut menu.

Note

Please note that only those loops that are programmed within a network are recognized.

Enabling/disabling the monitoring of loops in STL (S7-1500)

To enable the monitoring of loops in STL, follow these steps:

1. Open the STL block that you want to monitor.
2. Click "Monitoring on/off" in the toolbar.
   If you have not already established an online connection, the "Go online" dialog opens. In this dialog, you can establish an online connection.
   See also: Establishing and terminating an online connection
3. Enable the monitoring within the program editor with the shortcut menu command "Monitoring" > "Monitor loops".
4. Confirm the next prompt with "Yes".
   Result: The monitoring of loops is enabled for the entire block and is active until monitoring of loops is once again disabled or the online connection to the CPU is terminated.
5. Click "Monitoring on/off" in the toolbar again to switch the program status off.

Rules for loop monitoring in SCL (S7-1200/1500)

When monitoring loops in SCL, the display of the program status within programmed loops is switched off by default.

This affects the instructions "FOR", "WHILE" and REPEAT-UNTIL": No values are displayed for the tags within these instructions to avoid affecting the cycle time.

If you are monitoring an SCL block with programmed loops, a tooltip informs you that the monitoring of loops is currently disabled and can be enabled if required via the shortcut menu.

Enabling/disabling the monitoring of loops in SCL (S7-1200/1500)

To enable the monitoring of loops in SCL, follow these steps:

1. Open the SCL block that you want to monitor.
2. Click "Monitoring on/off" in the toolbar.
   If you have not already established an online connection, the "Go online" dialog opens. In this dialog, you can establish an online connection.
   See also: Establishing and terminating an online connection
3. Right-click within the loop, in the column "value, for example, and enable monitoring with the shortcut menu command "Monitor loops".
4. Confirm the next prompt with "Yes".
   Result: The monitoring of loops is enabled for the entire block and is active until monitoring of loops is once again disabled or the online connection to the CPU is terminated.
5. Click "Monitoring on/off" in the toolbar again to switch the program status off.

23.2.5 Monitoring structures

Introduction to monitoring structures

You can monitor the execution of the program with a block. This provides you with an overview of the values of the individual operands and the results of the logic operations and you can check whether the components of the automation system are correctly controlled.

As of TIA Portal V14, it is also possible to monitor structured PLC tags.

There are special rules for monitoring structures, depending on the particular CPU and programming language in use, and the properties of the respective structures.

Rules for monitoring structures (S7-1200/1500)

During the monitoring of structures, the values of a structured PLC tag are displayed in the program editor and in the Inspector window, with the following exception:

- It is not possible to monitor structures whose elements have adjustable retentivity properties.

To display the monitor values for a structure, you must first enable monitoring via the shortcut menu.

Displaying the monitor values for structures in the Inspector window

If the monitoring of structures is enabled and a structure tag is selected in the program editor, the associated monitor values are displayed in the Inspector window in the tab "Diagnostics" > "Monitor values".

The Inspector window is structured as follows:

- "Name" column shows the selected structure tag with the elements under the first level opened. If you click the name of the structure tag, the program editor jumps to this tag's point of use.
- The "Data type" column shows the associated data type.
- The "Value" columns shows the associated monitor values. If dashes are shown here instead of values, the monitoring of loops is disabled. In this case, enable the monitoring of loops again by means of the shortcut menu in the program editor.
- Columns "Value - IN" and "Value - OUT": These columns with the corresponding values are only displayed if the selected structure is used as an IN/OUT parameter.
- If the monitor values are displayed in a font with only 50 % saturation, this means that these vales are not from the current cycle.
- The monitor values displayed in the Inspector window use the same call path that is set as default for the block.
- The table with monitor values displayed in the Inspector window can be copied to an Excel file.
- If it is not possible to display any monitor values in the Inspector window, you will receive information that these are not available.

Procedure for monitoring structures

To enable the monitoring of structures, follow these steps:

1. Open the block in the program editor.
2. Click "Monitoring on/off" in the toolbar.
   If you have not already established an online connection, the "Go online" dialog opens. In this dialog, you can establish an online connection.
   See also: Establishing and terminating an online connection
3. If the structure to be monitored is located in a loop, first enable the monitoring of loops.
   See also: Monitoring loops (Page 7693)
4. Right-click on a structure and enable the monitoring of structures by means of the shortcut menu command "Monitoring" > "Monitor values".
   Result: The selected structure tag is displayed in the Inspector window in the tab "Diagnostics" > "Monitor values" is open up to the first level with the current values displayed.
5. If needed, open the closed elements of the structure to display all the lower-level elements and their values.
6. Click "Monitoring on/off" in the toolbar again to switch the program status off.

Note

Monitoring a structure within an array

Click directly behind the closing bracket to select the relevant structure for monitoring.

See also

Monitoring loops (Page 7693)

23.2.6 Editing blocks during the program test

If you edit blocks while they are being monitored, monitoring will be interrupted and you will be able to edit the block offline. If the block is not available offline in the project, you will first have to load it from the device to the project. After editing the block, you will also have to compile and download it again.

Procedure

To edit blocks while these are being monitored, follow these steps:

1. Edit the block as necessary.
   Monitoring is interrupted and the block is switched offline mode assuming it exists offline.
2. If the block does not exist offline, load it to the project from the device.
3. Compile the block.
   See also: Compiling blocks (Page 7539)
4. Download the block to the device.
   See also: Downloading blocks for S7-1200/1500 (Page 7544)

Result

The block now contains your modifications both online and offline. The online connection is re-established and monitoring continues.

23.2.7 Modifying tags during monitoring

During monitoring, you have the option of modifying tags immediately and once in the program status.

For Boolean values you can switch between the values "0" and "1".

You can enter a modify value for tags that do not belong to the "BOOL" data type.

Note that you cannot modify peripheral inputs, for example, via TagName:P.

You can control tags during monitoring either by double-clicking the online value (not for data type BOOL) or after the tags have been clicked by using the shortcut menu.

Procedure

To modify tags during monitoring, follow these steps:

1. Open the desired block and start the monitoring of tags by clicking "Monitoring on/off" in the toolbar.
2. Double-click the online value (not for data type BOOL) that you wish to modify.
   Alternatively, you can click the desired tag and start modifying by using the shortcut menu.
3. In the subsequent dialog enter the value you require in the "Modify value" box and confirm with "OK".
   Or confirm the query in the subsequent dialog whether the Boolean online value is to be switched over.
   Example:
   – If the Boolean tag has the value "0", the value is switched over to the value "1" if the query has been confirmed with "Yes".
   – If the Boolean tag has the value "1", the value is switched over to the value "0" if the query has been confirmed with "Yes".
   – If you click "No" or exit the dialog by using " Close", the existing value remains.
4. Alternatively, select one of the following commands in the shortcut menu:
   – "Modify > Modify to 1"
     Modifies tags of the "BOOL" data type to the value "TRUE".
   – "Modify > Modify to 0"
     Modifies tags of the "BOOL" data type to the value "FALSE".
   – "Modify > Modify operand"
5. If you select "Modify operand", the "Modify operand" dialog opens.
6. In the subsequent dialog enter the value you require in the "Modify value" box and confirm with "OK".

See also

Switching test with program status on/off (Page 7691)

23.2.8 Switching display formats in the program status

Introduction

The display formats for tags depend on the respective tag type. During monitoring, you have the option of switching the current display format by means of the shortcut menu. The possible display formats for a tag are offered in the shortcut menu.

You can modify the display format for the selected tag, for all tags in the network or for all tags in the block.

This is useful, for example, when you need a hexadecimal display in order to search for a hexadecimal error code.

Procedure

To switch the display format, follow these steps:

1. Open the desired block in the programming editor.
2. Switch on the program status by clicking "Monitoring on/off" in the toolbar.
   If you have not already established an online connection, the "Go online" dialog opens. In this dialog, you can establish an online connection.
3. Select the tag you want to monitor and select the desired display format in the shortcut menu under "Display format".

Result

The display format is modified accordingly.

Note

Switching display format during monitoring

Please note that it is not possible to switch the display format for unconnected outputs, as no monitoring value is output in this case.

23.2.9 Special considerations when testing GRAPH programs (S7-300, S7-400, S7-1500)

23.2.9.1 Basics on testing GRAPH programs (S7-300, S7-400, S7-1500)

Introduction

You can display the program status of a GRAPH program to check the logic, consistency and function of a sequential control system.

In the process you can, for example, determine the following errors:

- Program errors, for example, shifts between the programmed steps and conditions and the actual process sequence
- Programming errors in the sequential control system, such as incorrectly defined monitoring times in supervisions

To be able to test a user program, an online connection to the CPU must be available.

WARNING

Danger when testing during ongoing plant operation

Testing while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors.

Make sure that no dangerous situations can arise before you conduct a test.

Note

It is possible that another user can carry out a loading process on the selected CPU through joint parallel working on a CPU. If this load operation results in changes to the instance data block of the GRAPH block, the test is ended with program status for the GRAPH block and you receive an alarm in the Inspector window.

Test functions of GRAPH

The following test functions are available for testing sequential control systems that were created in GRAPH:

- Program status for sequences
- Program status for conditions and actions
- Program status for interlocks and supervision
- Program status for permanent pre-instructions and post-instructions

During a test, you have the option to control the sequencer or to synchronize it with the current process status.

Operating modes for testing GRAPH programs

The following three operating modes are available for the testing GRAPH programs:

- Automatic mode
  In this operating mode, the sequence automatically advances to the next step as soon as the transition is fulfilled.
- Semiautomatic mode
  In this operating mode, the sequence advances to the next step if one of the following conditions is fulfilled:
  - Transition is satisfied.
  - There is a rising signal edge at the "T_PUSH" parameter.
  - You advance by switching manually with the "Ignore transition" button.
- Manual mode
  In this operating mode, you can either advance from one step to the next manually or specifically select a step.

System synchronization

If a process is brought to another state in manual mode, it may no longer be synchronized with the sequencer. To synchronize the process with the program once again, you can have the program look for synchronization points and subsequently perform the synchronization.

You can select from the following methods to find the synchronization points:

- Preceding transition satisfied
  All steps whose preceding transition has been fulfilled are selected.
- Interlock condition satisfied
  All steps whose interlock condition has been fulfilled are selected.

Only those steps for which the subsequent transition is not fulfilled are selected for both options. Next you can enable one or more of the proposed steps.

Learning mode (S7-1500 only)

By testing with program status, you have the option of having the maximum step activation time and warning time determined by the system. You can use these times for your supervision conditions.

Note

For GRAPH blocks with a version lower than 2.0, the learning mode for determining the values of "T_MAX" and "T_WARN" is not available. You can change the version in the properties of the GRAPH block under "General > Block".

See also

Basics of testing the user program (Page 7685)

Introduction to testing with program status (Page 7687)

Overview of the test settings for GRAPH (Page 7702)

Controlling a sequencer (Page 7703)

Acknowledging supervision errors (Page 7704)

Testing in various operating modes (Page 7704)

Performing a system synchronization (Page 7705)

23.2.9.2 Specifying the test settings (S7-300, S7-400, S7-1500)

Overview of the test settings for GRAPH (S7-300, S7-400, S7-1500)

Overview

The following table provides an overview of the settings for testing GRAPH programs:

Setting	Description
Track active step	The step of the sequencer which is active in each case is displayed in the single-step view or in the sequence view.
Skip steps	A step whose preceding or subsequent transition is valid will not be activated. The default setting of this option is adopted from the corresponding block settings. This option is not available when you use the "Create minimized DB" option for the GRAPH function block.
Acknowledge supervision alarms	Specifies that supervision errors must be acknowledged before the program continues. This option overwrites the sequence property "Acknowledge supervision alarms" in the settings of the GRAPH function block and applies until the next download of the GRAPH instance data block.
Stop sequence	The sequencer is stopped, even if the next transition is fulfilled.
Stop timers	If this option is enabled, all step activation timers are stopped. If this option is disabled, all timers continue to run.
Process all interlocks	All interlocks, even those in step that are not active, are processed. You can recognize the interlocks that are fulfilled in the sequence view.
Process all transitions	All transitions are always processed. You can recognize the transitions that are fulfilled in the sequence view.
Activate actions	Activates the actions in active steps. This option is enabled by default. When you disable this option, no additional actions are executed in active steps.
Activate supervisions	Activates the supervisions in active steps. This option is enabled by default. When you disable this option, the conditions of supervision are ignored in active steps.
Activate interlocks	Activates the interlocks in active steps. This option is enabled by default. When you disable this option, the conditions of the interlock are ignored in active steps.

See also

Basics on testing GRAPH programs (Page 7699)

Changing test settings (Page 7702)

Changing test settings (S7-300, S7-400, S7-1500)

WARNING

Changing the test settings

Keep in mind that changes to the test settings can have unintended effects on the process. Make sure that no dangerous states can occur before you change the test settings.

Requirement

A GRAPH function block is open.

Procedure

To change the test settings of GRAPH, follow these steps:

1. Start the test with program status for the GRAPH function block.
   See also: Switching test with program status on/off (Page 7691)
2. Open the "Testing" task card.
3. Change the settings in the "Test settings" pallet.

See also

Overview of the test settings for GRAPH (Page 7702)

23.2.9.3 Controlling a sequencer during testing (S7-300, S7-400, S7-1500)

Controlling a sequencer (S7-300, S7-400, S7-1500)

You can re-initialize the sequencer during testing with the program status. The sequencer then starts with the initial steps once again. You also have the option to disable all steps for testing.

Requirement

The program status is displayed.

Initialize sequencer

To initialize the sequencer, follow these steps:

1. Open the "Testing" task card.
2. Click the "Initialize" in the "Sequence control" pallet.
   The sequencer is executed starting with the initial steps.

Disable all steps

To disable all steps for testing of a sequencer, follow these steps:

1. Open the "Testing" task card.
2. Click "Disable all" in the "Sequence control" pallet.
   All steps of the sequencer are disabled.

See also

Acknowledging supervision errors (Page 7704)
Testing in various operating modes (Page 7704)
Performing a system synchronization (Page 7705)
Basics on testing GRAPH programs (Page 7699)
Introduction to testing with program status (Page 7687)
Switching test with program status on/off (Page 7691)

Acknowledging supervision errors (S7-300, S7-400, S7-1500)

You can specify in the test settings that supervision errors have to acknowledged during testing before the program continues.

Requirement

The program status is displayed.

Procedure

To acknowledge supervision errors, follow these steps:

1. Open the "Testing" task card.
2. Click "Acknowledge - (V) -" in the "Sequence control" pallet.

See also

Controlling a sequencer (Page 7703)
Testing in various operating modes (Page 7704)
Performing a system synchronization (Page 7705)
Basics on testing GRAPH programs (Page 7699)
Overview of the test settings for GRAPH (Page 7702)
Changing test settings (Page 7702)
Introduction to testing with program status (Page 7687)
Switching test with program status on/off (Page 7691)

Testing in various operating modes (S7-300, S7-400, S7-1500)

Requirement

The program status is displayed.

Testing in automatic mode

Proceed as follows to test a sequencer in automatic mode:

1. Open the "Testing" task card.
2. Select "Automatic" as operating mode in the "Sequence control" pallet.

Testing in semiautomatic mode

Proceed as follows to test a sequencer in semiautomatic mode:

1. Open the "Testing" task card.
2. Select "Semiautomatic mode" as operating mode in the "Sequence control" pallet.
3. When switching does not continue automatically by a fulfilled transition or via the parameter "T_PUSH", click the "Ignore transition" button.

Testing in manual mode

To test a sequencer in manual mode, follow these steps:

1. Open the "Testing" task card.
2. Select "Manual mode" as operating mode in the "Sequence control" pallet.
3. Advance to another step:
   - If you want to enable the next step, click the "Next" button.
   - If you want to enable or disable any given step, enter the step number in the "Select step manually" field or select a step in the sequencer and click the "Enable" or "Disable" button.

See also

Controlling a sequencer (Page 7703)

Acknowledging supervision errors (Page 7704)

Performing a system synchronization (Page 7705)

Switching test with program status on/off (Page 7691)

Basics on testing GRAPH programs (Page 7699)

Introduction to testing with program status (Page 7687)

Performing a system synchronization (S7-300, S7-400, S7-1500)

Requirement

- The program status is displayed.
- The test takes place in "Manual mode" operating mode.

Procedure

Proceed as follows to perform a system synchronization:

1. Open the "Testing" task card.
2. Select the "Enable synchronization" check box in the "Sequence control" pallet.
3. Select one of these two methods for locating synchronization points:
   - "Preceding transition satisfied"
     All steps are marked whose preceding transitions are satisfied and whose subsequent transitions are not satisfied.
   - "Interlock condition satisfied"
     All steps are marked whose interlocks are satisfied and whose subsequent transitions are not satisfied.
4. Select a step with which the system synchronization should start. You can also select multiple steps.
5. Click the "Activate" button.

Result

The process and the sequencers are synchronized with each other.

See also

Basics on testing GRAPH programs (Page 7699)

Controlling a sequencer (Page 7703)

Acknowledging supervision errors (Page 7704)

Testing in various operating modes (Page 7704)

Introduction to testing with program status (Page 7687)

Switching test with program status on/off (Page 7691)

Calculating times for T_MAX and T_WARN when testing (S7-1500)

With the learning mode, you can have the values for the maximum step activation time and warning time for the entire duration of a step determined by the system. To be able to do this, each step must be executed at least once. The longer the learning mode is active, the more often a step is executed and the more precise the learned values are. The maximum step activation time and the warning time for a step are made up of the learned time and an additive threshold. You specify the additive thresholds for the supervision and the warning yourself.

When learning mode is ended, the values are stored in the instance data block on the device. You can also save the values in the offline instance data block. The values are adopted as the start values for the static parameters "T_MAX" and "T_WARN". Following this, the learned values are used in the comparison instructions "CMP>T_MAX" and "CMP>T_WARN".

You can reset the times for "T_MAX" and "T_WARN" determined in the learning mode to their default values at any time. This resets both the online and the offline data block.

Note

For GRAPH blocks with a version lower than 2.0, the learning mode for determining the values of "T_MAX" and "T_WARN" is not available. You can change the version in the properties of the GRAPH block under "General > Block".

Requirement

- The identical GRAPH function block exists in the device.
- Step time monitoring is programmed for at least one step.

Determining times

To determine the maximum step activation time and the warning time using the learning mode, follow these steps:

1. Open the GRAPH function block.
2. Enable the test with program status for the GRAPH function block.
   See also: Switching test with program status on/off (Page 7691)
3. Open the "Testing" task card.
4. Enter the additive threshold for the supervision as a percentage.
5. Enter the additive threshold for warnings as a percentage.
6. Select the "Enable learning mode" check box.
7. Wait until each step has been active at least once.
8. Clear the "Enable learning mode" check box.
   The "Save learned times" dialog opens.
9. If you want to save the learned times in the offline instance data block, click "OK".

   Note

   Regardless of your selection, the learned times are entered in the online instance data block.

   The learned times are entered in the static parameters "T_MAX" and "T_WARN" of the steps that were active during learning mode.

Resetting times

To reset the times, follow these steps:

1. Open the GRAPH function block.
2. Enable the test with program status for the GRAPH function block.
   See also: Switching test with program status on/off (Page 7691)

3. Open the "Testing" task card.
4. Click the "Reset learned times" button.
   The values for "T_MAX" and "T_WARN" in the instance data block are reset to the default values.

See also

Using step time monitoring (Page 7465)

23.2.10 Examples of program status display

23.2.10.1 Program status display for LAD programs

Displays in program status

The display of the program status is updated cyclically.

The following figure shows an example of the program status display for LAD:

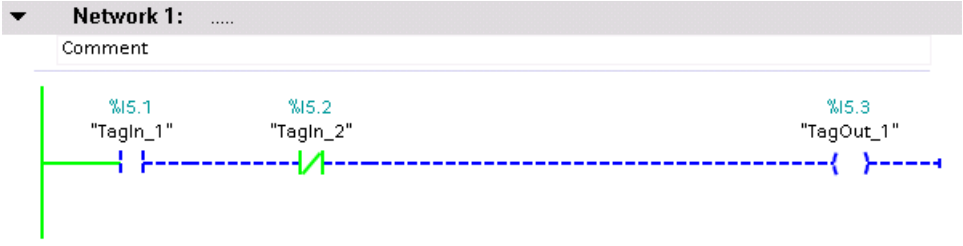

Representation of the program status

You can recognize the status of individual instructions and lines of a network quickly based on the color and type of lines and symbols. The following table shows the relationship between representation and status:

Representation	Status
Green solid	Satisfied
Blue dashed	Not satisfied
Gray solid	Unknown or not executed
Black	Not interconnected
Lines or parameters in a frame with a saturation of 100 %	Value is current
Lines or parameters in a frame with a saturation of 50 %	Value originates from an earlier cycle. The point in the program was not executed in the current cycle.

23.2.10.2 Program status display for FBD programs

Displays in program status

The display of the program status is updated cyclically.

The following figure shows an example of the program status display for FBD:

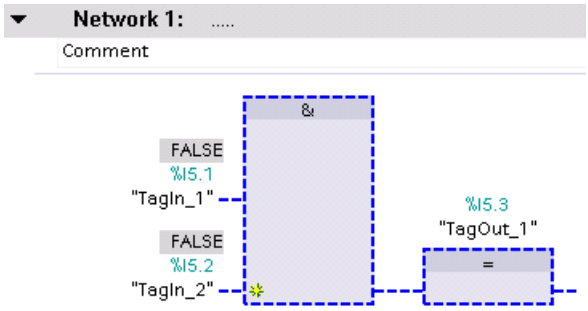

Representation of the program status

You can recognize the status of individual instructions and lines of a network quickly based on the color and type of lines and symbols. The following table shows the relationship between representation and status:

Representation	Status
Green solid	Satisfied
Blue dashed	Not satisfied
Gray solid	Unknown or not executed
Black	Not interconnected
Lines or parameters in a frame with a saturation of 100 %	Value is current
Lines or parameters in a frame with a saturation of 50 %	Value originates from an earlier cycle. The point in the program was not executed in the current cycle.

The values of the operands are displayed above the relevant operand name in a gray box.

Note

Program status display for outputs which are not interconnected

Please note that a monitor value cannot be displayed for outputs which are not interconnected.

23.2.10.3 Program status display for STL programs (S7-300, S7-400, S7-1500)

Displays in program status

The display of the program status is updated cyclically and shown in tables. The tables are shown directly next to the STL program. You can read the program status for each line of the program. The display depends on the CPU in use (S7-300, S7-400, or S7-1500).

The tables to be displayed in the program status always contain the following information:

- RLO
  The "RLO" column shows the result of logic operation for each line of program. You can recognize the value of the RLO based on the background color of the table cell. Here, green means an RLO of 1 and lilac an RLO of 0.
- Value
  The current value of the operand is shown in the "Value" column.
- Extra
  The "Extra" column shows additional information depending on the particular operation, for example, relevant status bits for mathematical instructions, time or count values for timers and counters or memory addresses for indirect addressing.

The following figure shows an example of the program status display of a CPU S7-300 under STL:

Network 1: .....

Comment

			RLO	Value	Extra
1	OPN "DB1"	%DB1	0	DB1	
2	L %DBW2	%DBW2	0	16#4D2	DB1
3	L %DBW4	%DBW4	0	16#3333	DB1
4	+I		0	14341	OS=0,OV=0,A0=0,A1=1
5	T "Tag_15"	%MW4	0	14341	
6					
7	L 3.0	3.0	0	3.0	
8	T "Tag_24"	%MD10	0	3.0	
9	L 5.0	5.0	0	5.0	
10	T "Tag_52"	%MD20	0	5.0	
11	L MD ["Tag_29"]	%MD26	0	16#40400000	P#10.0
12	LAR1 P#M16.0	P#M16.0	0	P#M16.0	
13	L MD [AR1 , P#4.0]	P#4.0	0	16#40A00000	P#M20.0
14	+R		0	8.0	OS=0,OV=0,A0=0,A1=1
15	T "Tag_17"	%MD30	0	8.0	
16					
17	OPN DB ["Tag_48"]	%MW14	0	DB1	
18	L DBW ["Tag_38"]	%MD60	0	16#3333	P#4.0
19	L W#16#7777	W#16#7777	0	16#7777	
20	XOW		0	16#4444	
21	T "Tag_47"	%MW70	0	16#4444	
22					
23	A "Tag_1"	%I0.0	1	1	
24	L S5T#55S	S5T#55S	1	S5T#55S	
25	SE "Tag_49"	%T5	1	S5T#47S600MS	
26	A "Tag_49"	%T5	1	1	S5T#47S600MS
27	= "Tag_3"	%Q0.0	1	1	
28					
29	A "Tag_55"	%I0.2	1	1	
30	CU "Tag_23"	%C1	1	C#556	
31	A "Tag_23"	%C1	1	1	C#556
32	= "Tag_3"	%Q0.0	1	1	
33					
34	A "Tag_31"	%M10.0	0	0	
35	AN M ["Tag_51"]	%MD64	0	0	P#5.1
36	O DBX ["Tag_38"]	%MD60	1	1	P#4.0
37	LAR2 P#M10.0	P#M10.0	1	P#M10.0	
38	X [AR2 , P#3.4]	P#3.4	1	0	P#M13.4
39	= Q ["Tag_51"]	%MD64	1	1	P#5.1
40					

23.2.10.4 Program status display for SCL programs

Displays in program status

The display of the program status is updated cyclically and shown in a table. The table is displayed immediately beside the SCL program and you can see the program status for each line of the program. The table contains the following information:

- Tag names
- Value

You can move the table to the left or right at any time.

The following figure shows an example of the program status display for SCL:

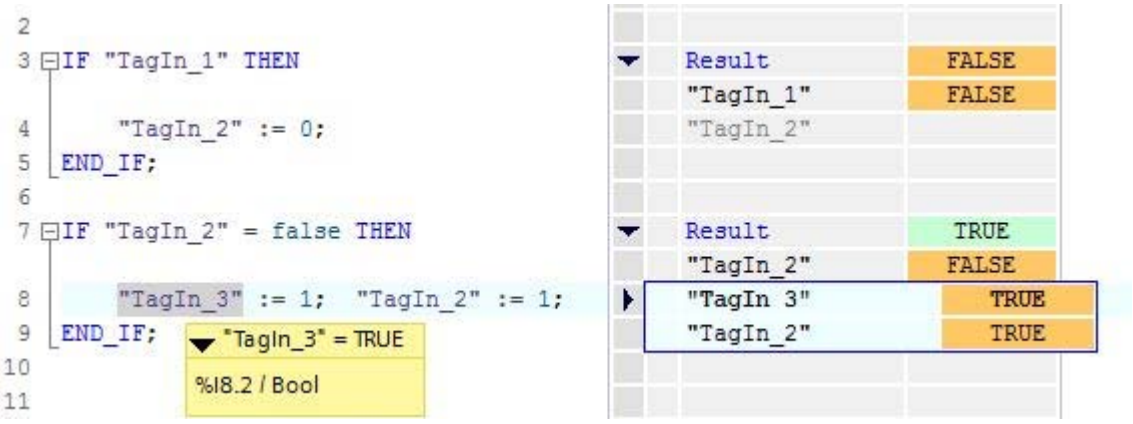

Explanation on monitoring in SCL

- If the code of the line is not executed, the tag name is displayed in the values table in gray text.
- The current values of the tags are displayed in the last column. If no values can be displayed for a tag, the corresponding cell has a yellow background and three question marks are shown. In this case, select the "Create extended status information" check box in the properties of the block and download the block to the device again. All values are then displayed.
- In the first column, you can see the name of the tag for which the current value is being displayed.
- If the line includes the "IF", "WHILE" or "REPEAT" instruction, the result of the instruction is displayed in the line as "True" or "False".
- If the line contains more than one tag, the value of the first tag is displayed. In both cases, all tags of these lines are displayed with their values in a separate list as soon as you select a line.
- If you place the cursor in a tag in the program code, this is shown in bold face in the list. You can also display the other tags of a line explicitly by clicking the arrow right located in front of lines containing more than one tag.

23.2.10.5 Program status display for GRAPH programs (S7-300, S7-400, S7-1500)

Displays in program status

For GRAPH programs you can have the program status displayed in sequence view and in single step view and for the permanent instructions. The program status display of permanent instructions corresponds to the program status display for LAB/FBD programs. The display of the program status is updated cyclically.

The following table shows the relationship between representation and status:

Representation	Area	Status
Green	Step, sequencer	There are no faults.
Light green	Step, sequencer	There are no faults. The values are from elapsed cycles.
	Condition	The transition is fulfilled.
Red	Step, sequencer	There is a supervision error.
Yellow	Step, sequencer	There is an interlock error.
Black	Condition	The transition is not fulfilled.

The following figures show examples of the display for the program status in the sequence view:

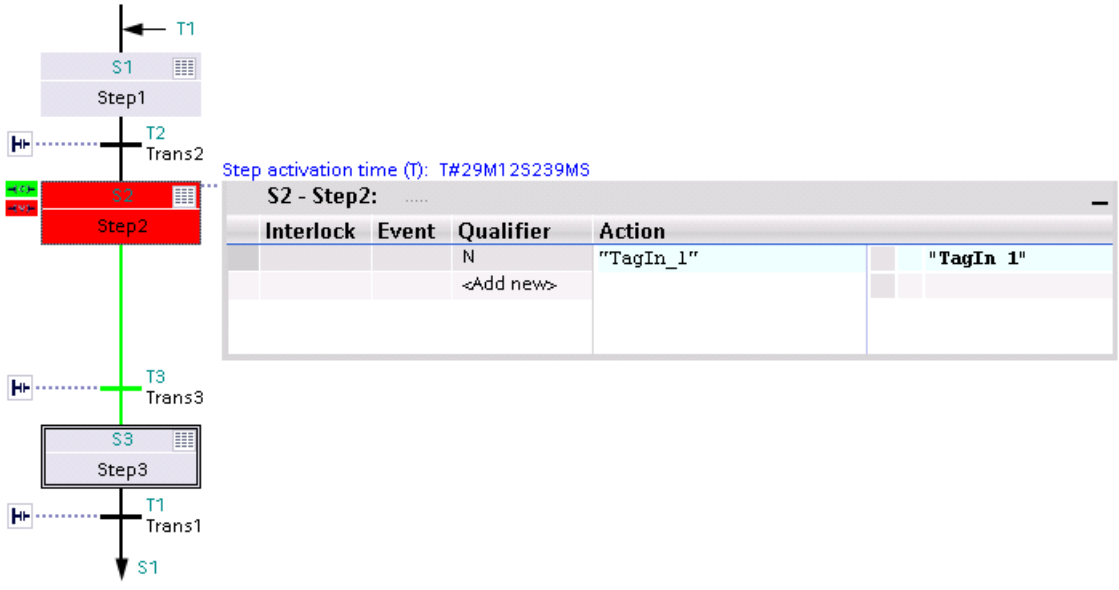

Step 2 contains a supervision error. The transition for switching from step 2 to step 3 is fulfilled.

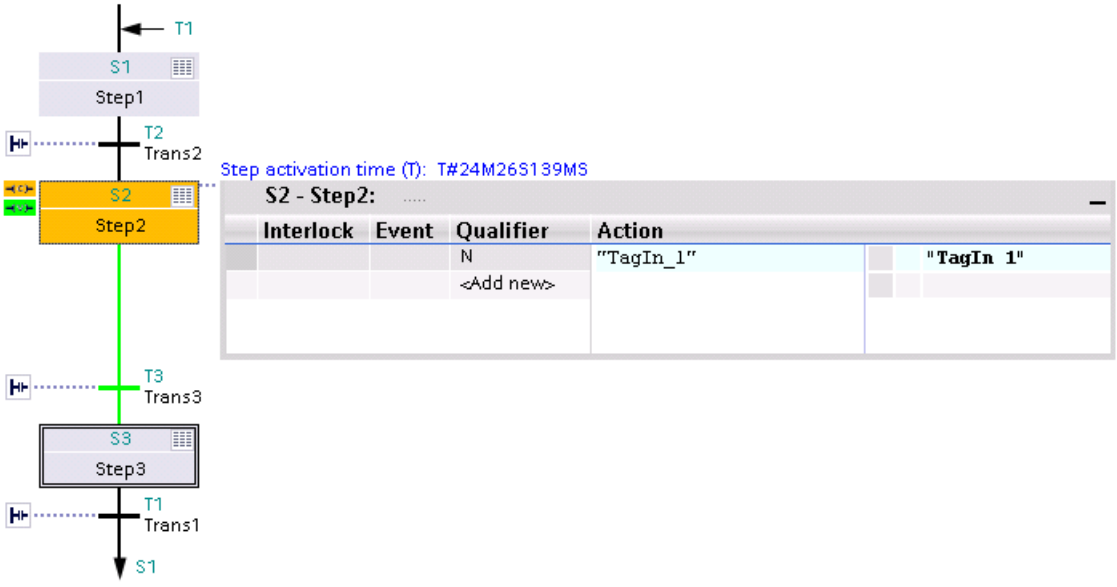

Step 2 contains an interlock error. The transition for switching to step 3 is fulfilled.

The figure below shows an example of the display for the program status in the single step view of a S7-300:

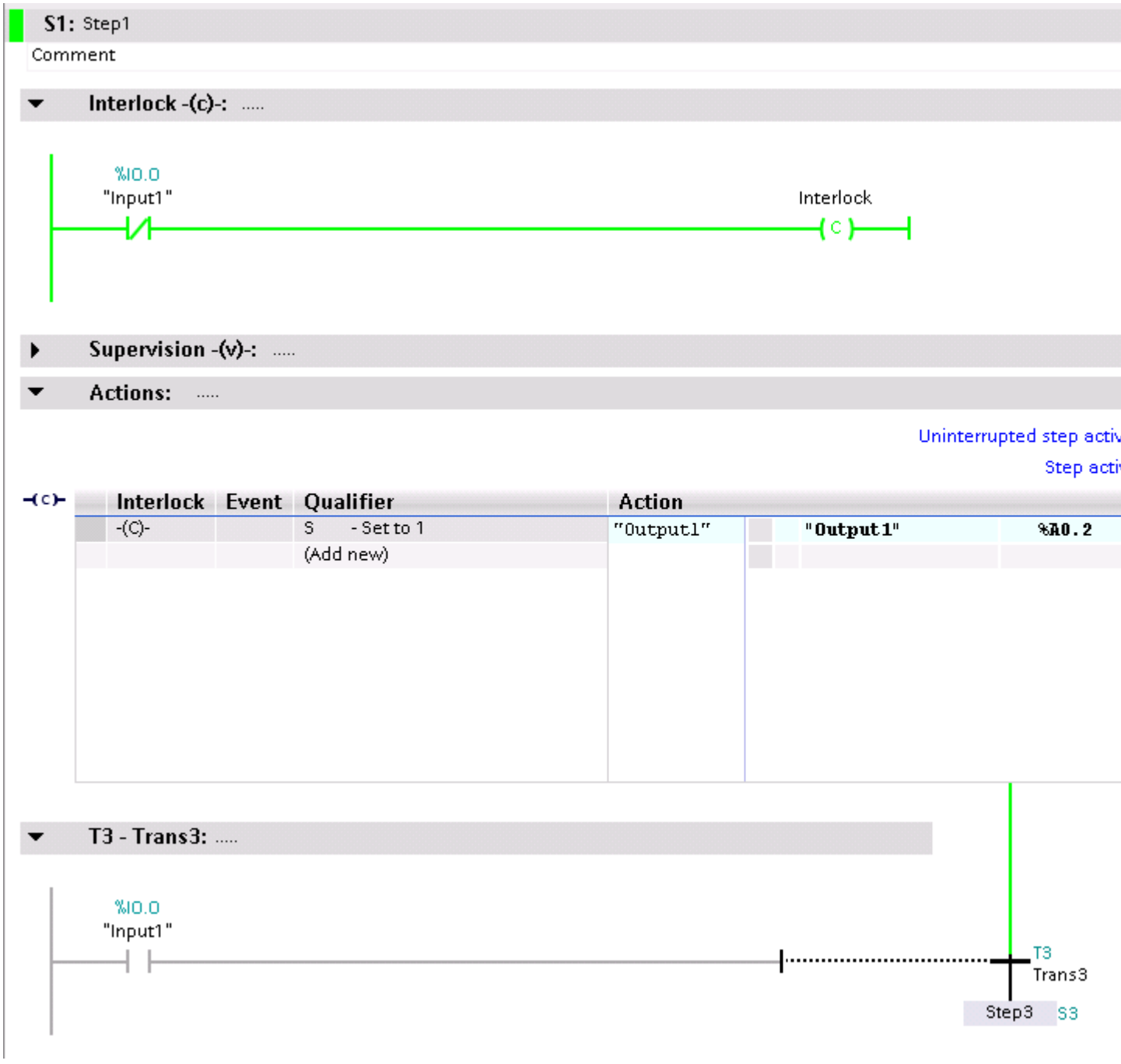

The step does not contain an interlock error. The subsequent transition is not satisfied.

23.3 Testing with breakpoints (S7-300, S7-400, S7-1500)

23.3.1 Introduction to testing with breakpoints (S7-300, S7-400, S7-1500)

Introduction

You can test blocks you created in the textual programming languages STL and SCL by having each instruction individually executed, step by step. To do this, you set breakpoints into the program code, establish an online connection and enable the breakpoints in the CPU.

The program code in STL and SCL can also be tested with breakpoints if this code is inserted into a LAD/FBD block.

As of TIA Portal V15, the following CPU families support the setting of breakpoints:

- S7-300 and S7-400
- S7-1500, as of firmware version V2.5

Note

Safety information for testing with breakpoints

Because you are interfering with the running process when testing with breakpoints, you should first be become familiar and follow the safety instructions for testing with breakpoints.

For more details, see:

Safety information for testing with breakpoints (Page 7718)

Security information for testing with breakpoints (Page 7717)

Note

Compatibility rules for breakpoints

- S7-1500 CPUs with older firmware versions (<2.5) do not support breakpoints.
- Breakpoints are not available for projects with older firmware versions that have been upgraded to the TIA Portal V15.
- Breakpoints are also not supported if a CPU S7-1500 with firmware version V2.5 is reset to an older firmware version. In this case, any breakpoints in the code are no longer displayed.
- Blocks that were created in older TIA Portal versions must be re-compiled on a CPU S7-1500 as of firmware V2.5 in order to use breakpoints.

Testing with breakpoints

If you have set valid breakpoints and have enabled them, you can run the program up to the first breakpoint in single-step testing and continue from there step-by-step using the following functions:

- Run
  Continue to the next enabled breakpoint.
- Run to cursor
  Continue to the cursor position.
- Step over (for S7-300/400 only)
  The currently selected instruction is executed.
- Step into (for S7-300/400 only)
  Jump into a lower-level block or call a lower-level block.
- Step out (for S7-300/400 only)
  If a block call is being processed and the program execution in the called block was interrupted at a breakpoint, you can use "Step out" to jump back to the called block.

You can set any number of breakpoints in the program code. The number of breakpoints that can be enabled in the CPU depends on the CPU. You can specify the time at which the program execution of individual breakpoints should be interrupted for a CPU S7-400 via the call environment.

Note

- If you terminate the online connection to the CPU, the breakpoints are disabled.
- The breakpoints are not saved with the project.
- If you close the project, the breakpoints are lost.

See also

Safety information for testing with breakpoints (Page 7718)

Rules for testing with breakpoints (Page 7720)

23.3.2 Security information for testing with breakpoints (S7-300, S7-400, S7-1500)

Disclaimer for the use of breakpoints with SIMATIC S7-1500

The SIMATIC S7-1500 supports the use of breakpoints for specifically stopping the program sequence for locating errors in the user program. When a configured breakpoint is reached by the user program, the processing cycle of the SIMATIC S7-1500 is stopped at it. The output values for the connected output modules are stopped in their respective states and (in contrast to the SIMATIC S7-300/S7-400) neither reset nor set to the configured substitute values. In addition, the SIMATIC S7-1500 can no longer react to input values of input groups in this state.

Keep this information in mind when operating your system and use breakpoints only if the above-described behavior in your application cannot result in damage to the environment, objects or injuries to persons.

Each customer is responsible for the proper and safe operation of the products and their installation within the applicable regulations.

Our liability for damages resulting from your use of breakpoints is excluded regardless of the legal basis unless required by law, e.g. according to the product liability act in cases of willful misconduct, gross negligence, personal injury or death, failure to achieve guaranteed characteristics, fraudulent concealment of a defect or in case of breach of fundamental contractual obligations.

The liability in case of a breach of fundamental contractual obligations, however, is limited to the foreseeable damages typical of the contract, unless intention or gross negligence are present or in case of mandatory liability due to injuries to life, body or health.

A change in the burden of proof to your disadvantage is not associated with this provision.

23.3.3 Safety information for testing with breakpoints (S7-300, S7-400, S7-1500)

Safety information for testing with breakpoints

You can test with breakpoints on all CPU series from the TIA Portal V15. However, the behavior is different when testing with breakpoints within the CPU series.

Since you interrupt the running process when testing with breakpoints, you should pay attention to the following safety instructions.

Testing with breakpoints on an S7-300, S7-400 and S7-1500 CPU

The following information applies when testing with breakpoints on an S7-300, S7-400 and S7-1500 CPU!

DANGER

Prevent personal injury and material damage!

Be aware that an incorrect action when executing the "Test with breakpoints" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Testing with breakpoints should **not** be used during plant operation!

Setting breakpoints in the standard user program results in errors in the safety program.

If you nevertheless want to use breakpoints for testing, you must disable safety mode.

The following errors will occur even when safety mode is disabled and breakpoints are used:

- Error in communication with the fail-safe I/O
- Errors during safety-related CPU-CPU communication

Testing with breakpoints on an S7-1500 CPU

The following information only applies when testing with breakpoints on an S7-1500 CPU, as of firmware V2.5!

DANGER

Prevent personal injury and material damage!

- If you enable the "Test with breakpoints" function, the outputs remain enabled. The outputs retain their last status and are no longer updated when the CPU goes to "HOLD" mode!
- When testing with breakpoints, enabled outputs are **not** disabled and are **not** reset to safe substitute values.
  This can, for example, cause motors to continue running and valves to remain open.
- Testing with breakpoints should **not** be used during plant operation!
- Especially where a Motion Control TO object is used in the CPU, it must **not** be tested with breakpoints. In this case, if a breakpoint is enabled and is reached, the CPU immediately goes to "STOP" after "HOLD".
  For example, this can cause connected motors and drives to be stopped, the loss of axis homing and the synchronous operation to be canceled in an uncontrolled manner.
- Testing with breakpoints can increase the cycle time of the CPU.
- Before you start the "Test with breakpoints" function, make sure that no one is running this function on the same CPU.
- Setting breakpoints in the standard user program results in errors in the safety program. When testing with breakpoints, enabled outputs of the S7-1500 are **not** disabled. This means that the F-modules automatically switch to the safe state after the configured monitoring time has elapsed.

Testing with breakpoints on an S7-1500 CPU

The following information only applies when testing with breakpoints on an S7-1500 CPU, as of firmware V2.5!

NOTICE

Incorrect display of tag values during monitoring

- Due to the optimization of the code, values that have not yet been reached can also be displayed as already assigned. Because these values could already be calculated, they are also displayed in black.
- When monitoring arrays within loops it may happen that incorrect values are displayed. In these cases only those values are displayed as correct that are within the loop and before a breakpoint. Displayed values that are located after the breakpoint within the loop and the displayed value at the breakpoint may not be correct. The code is correctly executed independent of the incorrect display.
- Loops can also be generated through specific instructions (for example, "Label" and "GOTO" in SCL and "LOOP" in STL). In these cases, too, only those values are displayed as correct that are within the loop and before a breakpoint. Displayed values that are located after the breakpoint within the loop and the displayed value at the breakpoint may not be correct. The code is correctly executed independent of the incorrect display.

See also

Introduction to testing with breakpoints (Page 7716)

Rules for testing with breakpoints (Page 7720)

23.3.4 Rules for testing with breakpoints (S7-300, S7-400, S7-1500)

Introduction

Depending on the CPU and programming language used, the rules for setting breakpoints listed below apply.

Note

Breakpoints must be enabled after they are set before you can execute them.

Breakpoints are **not** saved in the project. If you close the project, the breakpoints are lost.

Rules for setting breakpoints for S7-1500 CPUs in STL and SCL

Breakpoints can be set at the beginning of each line of code, but may be invalidated if this code is optimized during compilation.

If a breakpoint is set on a line of code optimized during compilation, this breakpoint is marked as "invalid" and cannot be enabled.

Therefore, it is helpful to set breakpoints at the beginning of logically related sections of code.

Valid breakpoints in STL

Valid breakpoints can be set in STL, for example, at the following positions:

- On the first instruction at the beginning of an STL network.
- Before an instruction with operand read access following an instruction with write operand access.
- On the first instruction after a conditional jump.
- Before and after block calls.

Valid breakpoints in SCL

Valid breakpoints can be always be set in all instructions in SCL.

Exceptions are:

- If more than one instruction is compiled by the compiler, only the first breakpoint before the start of the first instruction remains valid.
- No breakpoints are possible at the start or end of control structures (REPEAT, END_IF, END_FOR, END_WHILE, END_CASE).

23.3.5 Status display for breakpoints (S7-300, S7-400, S7-1500)

Introduction

The display for breakpoints is identical in the left column of the open program editor as well as under "Breakpoints" in the "Testing" task card.

Changes you make are immediately updated in the Program editor as well as in the "Testing" task card.

Breakpoints are visualized with different symbols depending on their respective status.

Displaying the status for breakpoints

The following table shows how breakpoints are displayed in the Program editor depending on their status in the program code:

Displaying the status for breakpoints in an S7-300/400 and S7-1500 CPU:

Icon	Meaning
	Breakpoint is inactive
	Breakpoint is inactive with condition, (for example, call environment for the breakpoint is defined)
	Breakpoint is inactive, program execution is at this point
	Breakpoint is inactive, has a condition (for example, call environment for the breakpoint is defined) and the program execution is at this point.
	Breakpoint is active
	Breakpoint is active with condition, (for example, call environment for the breakpoint is defined)
	Breakpoint is active and the program execution is at this point.
	Breakpoint is active, has a condition (for example, call environment for the breakpoint is defined) and the program execution is at this point.
	Program execution is at this point

The following symbols are also available in the S7-1500 CPU family as of firmware V2.5:

Icon	Meaning
	The breakpoint is disabled and cannot be enabled. Possible causes: • This breakpoint can no longer be reached after compiling because of code optimizations. • The breakpoint cannot be enabled because a call environment has been defined for monitoring the block. For more on this, see: Rules for testing with breakpoints (Page 7720)
	The breakpoint is inactive and cannot be enabled and the program execution is at this point. Possible causes: • The breakpoint cannot be enabled because a call environment has been defined for monitoring the block. For more on this, see: Rules for testing with breakpoints (Page 7720)

See also

Rules for testing with breakpoints (Page 7720)

23.3.6 Setting the call environment for breakpoints (S7-300, S7-400, S7-1500)

Introduction

By setting the call environment, you can specify when the program is interrupted at the breakpoints.

However, you can only set a call environment for S7-400 CPUs. If you are using a different CPU, it is **not** possible to set a call environment for a breakpoint.

The following rules apply to setting a call environment for monitoring blocks:

Valid for S7-1500er CPUs as of firmware version V2.5:

- As soon as a breakpoint is enabled in a block, it is **not** possible to configure a call environment for monitoring.
- In the other direction, a breakpoint in a block **cannot** be enabled if a call environment has already been configured.
- If you attempt to use both functionalities, you are shown an error message with information about how to proceed.

Requirement

- There are valid breakpoints in the program.
- The CPU supports the use of breakpoints and configuration of a call environment.

Procedure

To set the call environment for a breakpoint, follow these steps:

1. Open the "Testing" task card.
2. On the "Breakpoints" pane, click the "..." button after the breakpoint for which you want to set the call environment.
   The "Call environment of a block" dialog opens.
3. Select the condition you want to apply.
   See also: Basics of the call environment (Page 7689)
4. Confirm your selection with "OK".

Result

The call environment is specified for the breakpoint and the symbol for the breakpoint changes from a single gray circle to a gray circle with vertical lines before and after it.

See also

Introduction to testing with breakpoints (Page 7716)

Setting breakpoints (Page 7723)

Enabling breakpoints (Page 7725)

Testing with breakpoints in single step mode (Page 7731)

Deleting breakpoints (Page 7730)

23.3.7 Setting breakpoints (S7-300, S7-400, S7-1500)

Introduction

Breakpoints can be set in the program code beginning with TIA Portal V15 for STL and SCL with a CPU S7-300, S7-400 and S7-1500 as of firmware V2.5.

To use the breakpoint functionality, older projects must be re-compiled with the new version of the TIA Portal.

There are several ways to set breakpoints:

- From the "Online" menu > "Breakpoints" > "Set breakpoint".
  This menu command is only visible if the cursor is not placed in the block interface in an open block.
- Via the shortcut menu "Set breakpoint".
- Use the icons under the "Breakpoints" button in the "Testing" task card.
- Via the keyboard command "CTRL+Shift+F9"
- Click on the gray column before the line number in the opened block.

Newly set breakpoints are initially disabled by default and must first be enabled before execution.

See also:

Security information for testing with breakpoints (Page 7717)

Safety information for testing with breakpoints (Page 7718)

Rules for testing with breakpoints (Page 7720)

Requirement

The block was created in STL or SCL.

You are working on a CPU S7-300, S7-400 or S7-1500 as of firmware V2.5, which supports breakpoints.

Procedure

To set a breakpoint in the program code, follow these steps:

1. Open the block in which you want to set a breakpoint.
2. Right-click in the line in which you want to set the breakpoint and double-click in the gray column in front of the line number.
3. Select the "Set breakpoint" command in the shortcut menu.

Or:

1. Open the block in which you want to set a breakpoint.
2. On the line for which you want to set the breakpoint, click on the gray column in front of the line number.

Result

A breakpoint is set in the program code. The breakpoint is displayed in the "Breakpoints" pane of the "Testing" task card and the breakpoint is also displayed as a grey circle in front of the corresponding line in the program code. Before the program is interrupted at the newly set breakpoint, the breakpoint must first be enabled.

Note

Setting and enabling breakpoints during monitoring

As of TIA Portal V15, you can also set breakpoints while block monitoring is active.

You can also enable breakpoints with a CPU S7-1500 as of firmware V2.5 while block monitoring is active.

See also

Introduction to testing with breakpoints (Page 7716)
Setting the call environment for breakpoints (Page 7722)
Enabling breakpoints (Page 7725)
Testing with breakpoints in single step mode (Page 7731)
Deleting breakpoints (Page 7730)
Security information for testing with breakpoints (Page 7717)
Safety information for testing with breakpoints (Page 7718)
Rules for testing with breakpoints (Page 7720)

23.3.8 Enabling breakpoints (S7-300, S7-400, S7-1500)

Introduction

To be able to use breakpoints when testing, the set breakpoints must first be enabled.

Enabling breakpoints installs and activates them on the CPU.

You can enable breakpoints either individually or all together.

You have the option to edit breakpoints using the "Testing" task card or alternatively using the Online menu and the shortcut menu.

Valid for S7-1500er CPUs as of firmware version V2.5:

- As soon as a breakpoint is enabled in a block, it is **not** possible to configure a call environment for monitoring.
- In the other direction, a breakpoint in a block **cannot** be enabled if a call environment has already been configured.
- If you attempt to use both functionalities, you are shown an error message with information about how to proceed.

Note

The number of breakpoints that can be enabled on the CPU depends on the particular CPU.

Once you have reached the maximum number, you cannot enable any more breakpoints.

Because the single step functions "Step over", "Step into" and "Run to cursor" are also used as breakpoints, they also cannot be performed when the maximum number of breakpoints is reached. Before you can enable further breakpoints or use these single step functions, you first need to disable breakpoints already enabled.

Note

Enabling breakpoints during monitoring

As of TIA Portal V15, breakpoints can be enabled with CPU S7-1500 as of firmware V2.5 even when the block monitoring is active.

Requirement

- Valid breakpoints were set.
- There is an online connection to the CPU and this CPU supports the use of breakpoints.
- The maximum number of breakpoints that can be enabled for the CPU has not been reached.
- The offline and the online blocks on the CPU are identical.

Enabling individual breakpoints

To enable individual breakpoints, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Right-click on the breakpoint you want to enable.
3. Select the "Enable breakpoint" command in the shortcut menu.

NOTICE
Enabling breakpoints on a CPU S7-1500 as of firmware version V2.5 As soon as you enable a breakpoint on a CPU S7-1500 as of V2.5, you are informed that the output values for the connected output modules are stopped in their respective state and (in contrast to SIMATIC S7-300/S7-400) neither reset nor set to the configured substitute values. In addition, the SIMATIC S7-1500 can no longer react to input values of input groups in this state. This information is shown to you only once in each online session.

Result

The selected breakpoint is installed on the CPU and enabled and the program is interrupted at the breakpoint when it reaches that point in the program. The display of the breakpoint changes from a gray circle to a red circle.

Enabling all breakpoints

To enable all breakpoints, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Make sure that no breakpoint is selected.
3. Click "Enable all breakpoints" in the toolbar of the "Breakpoints" pane.

NOTICE
Enabling breakpoints on a CPU S7-1500 as of firmware version V2.5 As soon as you enable a breakpoint on a CPU S7-1500 as of firmware version V2.5, you are informed that the output values for the connected output modules are stopped in their respective state and (in contrast to SIMATIC S7-300/S7-400) neither reset nor set to the configured substitute values. In addition, the SIMATIC S7-1500 can no longer react to input values of input groups in this state. This information is shown to you only once in each online session.

Result

Depending on the capacity of the CPU, all breakpoints are installed and enabled on the CPU and the program is interrupted at all breakpoints that are reached when it executes.

- If you have set more breakpoints than allowed by the CPU, only the permitted number of breakpoints is enabled.
- The selection of which breakpoints are enabled can change in runtime.
- Repeatedly executing the function can produce different results.

The display of the enabled breakpoints changes from gray circles to red circles.

See also

Introduction to testing with breakpoints (Page 7716)

Setting breakpoints (Page 7723)

Setting the call environment for breakpoints (Page 7722)

Testing with breakpoints in single step mode (Page 7731)

Deleting breakpoints (Page 7730)

23.3.9 Disabling breakpoints (S7-300, S7-400, S7-1500)

Introduction

You can disable enabled breakpoints either individually or all together. After you have disabled breakpoints, they are ignored when the program executes.

You have the option to edit breakpoints using the "Testing" task card or alternatively using the Online menu and the shortcut menu.

Note

Important facts about breakpoints

- If you terminate the online connection to the CPU, the breakpoints are disabled.
- The breakpoints are not saved with the project.
- If you close the project, the breakpoints are lost.

Note

Disabling breakpoints during monitoring

As of TIA Portal V15, breakpoints can also be disabled while block monitoring is active.

Requirement

- Valid breakpoints were set and enabled.
- There is an online connection to the CPU and this CPU supports the use of breakpoints.

Disabling individual breakpoints

To disable individual breakpoints, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Right-click the breakpoint that you want to disable.
3. Select the "Disable breakpoint" command in the shortcut menu.

Result

The selected breakpoint is uninstalled on the CPU and disabled and the program is no longer interrupted at the breakpoint. The display of the breakpoint changes from a red circle to a gray circle.

Disabling all breakpoints

To disable all breakpoints, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Make sure that no breakpoint is selected.
3. Click "Disable all breakpoints" on the toolbar of the "Breakpoints" pane.

Result

All breakpoints are uninstalled on the CPU and disabled and the program is no longer interrupted at any breakpoint. The display of the breakpoints changes from red circles to gray circles.

23.3.10 Navigating to the breakpoints (S7-300, S7-400, S7-1500)

Introduction

To navigate to the breakpoints in your program, you can either jump from breakpoint to breakpoint or to the point in the program where a specific breakpoint is located.

Navigating from breakpoint to breakpoint

To navigate from breakpoint to breakpoint in your program, follow these steps:

1. Open a block and select the programming editor.
2. In the menu "Online > Breakpoints", select "Go to next breakpoint" to jump to the next breakpoint after the current cursor position. When you reach the last breakpoint, you jump to the first breakpoint.
3. In the menu "Online > Breakpoints", select "Go to previous breakpoint" to jump to the previous breakpoint before the current cursor position. When you reach first breakpoint, you jump to the last breakpoint.

Navigating to a specific breakpoint

To navigate to a specific breakpoint in your program, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Double-click on the breakpoint to which you want to navigate.
   The block with the breakpoint is displayed in the programming editor and the line with the breakpoint is highlighted.

Or:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Right-click on the breakpoint to which you want to navigate.
3. Select the "Go to breakpoint" command in the shortcut menu.
   The block with the breakpoint is displayed in the programming editor and the line with the breakpoint is highlighted.

23.3.11 Deleting breakpoints (S7-300, S7-400, S7-1500)

Introduction

You can delete breakpoints either individually or all together.

Alternatively, you can delete breakpoints as follows:

- From the menu "Online > Breakpoints > Set/delete breakpoints", or "Online > Breakpoints > Delete all".
- Via the corresponding shortcut menu commands.
- Using the corresponding keyboard commands (CTRL+Shift+F9).
- Using the commands in the "Testing" task card.

Note

- If you terminate the online connection to the CPU, the breakpoints are disabled.
- The breakpoints are not saved with the project.
- If you close the project, the breakpoints are lost.

Note

Deleting breakpoints during monitoring

As of TIA Portal V15, breakpoints can be deleted with CPU S7-1500 as of firmware V2.5 even when the block monitoring is active.

Deleting individual breakpoints

To delete an individual breakpoint from the program, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Select the breakpoint you want to delete.

3. Right-click on the selected breakpoint.
4. Select the "Delete breakpoint" command in the shortcut menu.

Or:

1. Open the block in which you want to delete a breakpoint.
2. Click in the line containing the breakpoint you want to delete and click in the gray column in front of the row number.

Result

The selected breakpoint is deleted.

Deleting all breakpoints

To delete all breakpoints, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Click "Delete Breakpoints" in the toolbar of the "Breakpoints" pane.

Result

All existing breakpoints are deleted.

See also

Introduction to testing with breakpoints (Page 7716)

Setting breakpoints (Page 7723)

Setting the call environment for breakpoints (Page 7722)

Enabling breakpoints (Page 7725)

Testing with breakpoints in single step mode (Page 7731)

23.3.12 Testing with breakpoints in single step mode (S7-300, S7-400, S7-1500)

Introduction

If the program has been interrupted at a breakpoint, you can continue the program step-by-step.

Requirement

- Breakpoints are set and enabled in the program code.
- The maximum number of breakpoints that can be enabled for the CPU has not been reached.
- A block is open.

Procedure

To run the program in single step mode, follow these steps:

1. Establish an online connection to the CPU.
2. Enabling breakpoints
   The program code executes as far as the enabled breakpoint and is then interrupted, in other words, the CPU changes to "HOLD" mode. The red circle for the breakpoint has a yellow arrow superimposed on it.
3. To continue running the program step by step, select one of the following commands from the toolbar:
   - Run
     The program is continued. If there are other breakpoints, the program is interrupted at the next breakpoint.
   - Run to cursor
     The program continues as far as the point at which the insert point is set. If there is a further breakpoint before the insert point, the program is interrupted first at this breakpoint.
   - Step over (for S7-300/400 only)
     The next instruction after the breakpoint is executed. If the instruction is a block call, the call is processed and execution jumps to the next instruction following the block call.
   - Step into (for S7-300/400 only)
     The next instruction after the breakpoint is executed. If the instruction is a block call, execution jumps to the called block. There, you can continue testing in single step mode or set further breakpoints. At the end of the block, execution jumps back to the next instruction following the block call.
   - Step out (for S7-300/400 only)
     If a block call is being processed and the program execution in the called block was interrupted at a breakpoint, you can use "Step out" to jump back to the called block. In the process the remaining instructions of the called block will be executed and the program execution is halted again in the calling block after the block call.

See also

Introduction to testing with breakpoints (Page 7716)

Setting breakpoints (Page 7723)

Setting the call environment for breakpoints (Page 7722)

Enabling breakpoints (Page 7725)

Deleting breakpoints (Page 7730)

Introduction to testing with program status (Page 7687)

23.3.13 "Run to cursor" function (S7-300, S7-400, S7-1500)

Introduction

The "Run to cursor" function sets a temporary breakpoint at the insertion point and deletes it again once it is reached.

You can perform this function using the corresponding icon in the "Testing" task card as well as using the shortcut menu and also with the keyboard combination "CTL + Shift + 8".

Note

The number of breakpoints that can be enabled on the CPU depends on the particular CPU.

Once you have reached the maximum number, you cannot enable any more breakpoints.

Because the single step functions "Step over", "Step into" and "Run to cursor" are also used as breakpoints, they also cannot be performed when the maximum number of breakpoints is reached. Before you can enable further breakpoints or use these single step functions, you first need to disable breakpoints already enabled.

Requirement

- There is an online connection to the CPU and this CPU supports the use of breakpoints.
- Valid breakpoints were set and enabled.
- The maximum number of breakpoints that can be enabled for the CPU has not been reached.
- The offline and the online blocks on the CPU are identical.

Run to cursor

To enable the "Run to cursor" function, proceed as follows:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Right-click on the breakpoint you want to enable.
3. Select the "Enable breakpoint" command in the shortcut menu.
4. Wait until the breakpoint is reached and the CPU at the defined breakpoint goes to the "HOLD" state.

5. Move the insertion point in the program code to another location that is valid for inserting another breakpoint.
6. Start the "Run to cursor" function with the icon in the "Testing" task card or via the shortcut menu.

NOTICE
Enabling breakpoints on a CPU S7-1500 as of firmware version V2.5 As soon as you enable a breakpoint on a CPU S7-1500 as of V2.5, you are informed that the output values for the connected output modules are stopped in their respective state and (in contrast to SIMATIC S7-300/S7-400) neither reset nor set to the configured substitute values. In addition, the SIMATIC S7-1500 can no longer react to input values of input groups in this state. This information is shown to you only once in each online session.

Result

The function is executed and a temporary breakpoint is set at the cursor. The CPU goes to "HOLD" at the location indicated by the insertion point.

The position at which the breakpoint was set is indicated as "reached" by the small yellow triangle, the temporary breakpoint has been deleted.

23.3.14 Monitoring tags when the breakpoint is reached (S7-1500)

Introduction

For a CPU S7-1500 as of firmware V2.5, you can also monitor tags in conjunction with breakpoints, for example, if the CPU is "HOLD" when a breakpoint is reached in the operating state.

This functionality is only available for global operands (inputs, outputs, bit memory, SIMATIC timers, SIMATIC counters and DBs) in elementary data types.

When you position the mouse pointer over the tags, the value of the operand is displayed in a tooltip.

Requirement

- There is an online connection to the CPU.
- Valid breakpoints were set and enabled.
- A breakpoint is shown as "reached" in both the "Testing" task card as well as in the Program editor.
- The CPU is in "HOLD" mode, which is shown in the "Testing" task card for both the operating panel as well as for the breakpoints.

Monitoring tags when the breakpoint reached (S7-1500)

To monitor tags in "HOLD", proceed as follows:

1. Open an STL or SCL block that contains valid breakpoints in the Program editor.
2. Open the "Testing" task card. In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
3. Right-click on the breakpoint you want to enable.
4. Select the "Enable breakpoint" command in the shortcut menu.
5. Wait until the breakpoint is reached during the program execution and the CPU goes to "HOLD".
6. Position the mouse pointer in the program code over a global tag.

Result

The monitor value for this global tag is displayed in a yellow tooltip.

Rules for monitoring tags with active breakpoints

In order to be able to monitor tags, this functionality must be enabled via the "Monitoring on/off" button in the toolbar of the programming editor.

Depending on the time you enable tag monitoring during program execution, you obtain different results in the program status display:

- If a breakpoint is reached before you have enabled monitoring, no values are displayed in the program status.
- If a breakpoint is reached during the first run after the monitoring is switched on, the program status only shows the values up to the breakpoint reached.
- If a breakpoint is reached only after the block has been called several times, the monitoring values are displayed with the program status as follows:
  - Values from previous cycles are displayed in gray.
  - Values from the current cycle in which the CPU reached the breakpoint are displayed in black.

Note

- The breakpoints are not saved with the project. If you close the project, the breakpoints are lost.
- If you terminate the online connection to the CPU, the breakpoints are disabled.

See also

Modifying tags during monitoring (Page 7697)

23.3.15 Modifying tags when the breakpoint reached (S7-300, S7-400, S7-1500)

Introduction

For a CPU S7-1500 from firmware V2.5, you can also control tags in conjunction with breakpoints, for example, if the CPU is "HOLD" when a breakpoint is reached in the operating state.

This functionality is only available for global operands (inputs, outputs, bit memory, SIMATIC timers, SIMATIC counters and DBs) in elementary data types.

The following control options are available in the shortcut menu:

- Modify to 0
- Modify to 1
- Modify operand: Enter the desired modify value in the following dialog.

The selected operand is modified immediately and once.

With a CPU S7-1500 as of firmware V2.5, you can also control global operands directly in the code during monitoring.

With a CPU S7-300/400, you can also control global operands in an SCL block directly in the code during monitoring.

If the CPU has reached a breakpoint and is in the "HOLD" state, you can also control global operands via the watch table.

Requirement

- There is an online connection to the CPU.
- Valid breakpoints were set and enabled.
- A breakpoint is shown as "reached" in both the "Testing" task card as well as in the Program editor.
- The CPU is in "HOLD" mode, which is shown in the "Testing" task card for both the operating panel as well as for the breakpoints.

Modifying tags when the breakpoint reached (S7-1500)

To modify tags in "HOLD", proceed as follows:

1. Open an STL or SCL block that contains valid breakpoints in the Program editor.
2. Start monitoring via the button in the "Monitoring on/off" toolbar.
3. Open the "Testing" task card. In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
4. Right-click on the breakpoint you want to enable.
5. Select the "Enable breakpoint" command in the shortcut menu.
6. Wait until the breakpoint is reached during the program execution and the CPU goes to "HOLD".

7. Modify the operand to the desired value by clicking on the corresponding command in the shortcut menu.
   - "Modify > Modify to 1"
     Modifies tags of the "BOOL" data type to the value "TRUE".
   - "Modify > Modify to 0"
     Modifies tags of the "BOOL" data type to the value "FALSE".
   - "Modify > Modify operand"
     If you select "Modify operand", the "Modify operand" dialog opens.
     In the subsequent dialog enter the value you require in the "Modify value" box and confirm with "OK".
8. You can also double-click the online value (not for data type BOOL) that you wish to modify.
9. In the subsequent dialog enter the value you require in the "Modify value" box and confirm with "OK".
   Or confirm the query in the subsequent dialog whether the Boolean online value is to be switched over.

- **Example**:
  - If the Boolean tag has the value "0", the value is switched over to the value "1" if the query has been confirmed with "Yes".
  - If the Boolean tag has the value "1", the value is switched over to the value "0" if the query has been confirmed with "Yes".
  - If you click "No" or exit the dialog by using " Close", the existing value remains.

  Note

  Modifying tags in operating mode "HOLD"

  Note that the displayed tags are no longer updated if the CPU is in the operating mode "HOLD".

  The tags displayed in "HOLD" are always modified.

Result

The selected operand is modified to the desired value.

See also

Modifying tags during monitoring (Page 7697)

23.3.16 Resuming program execution (S7-300, S7-400, S7-1500)

Introduction

When you have set valid breakpoints and enable them, you can run the program up to any breakpoint for testing. As soon as a valid breakpoint has been reached, the CPU is in the "HOLD" state.

Program execution on a CPU S7-1500 is not resumed until you reset the CPU to the "RUN" mode.

You can resume the program execution with or without breakpoints.

Requirement

- There is an online connection to the CPU.
- Valid breakpoints were set and enabled.
- A breakpoint is shown as "reached" in both the "Testing" task card as well as in the Program editor.
- The CPU is in "HOLD" mode, which is shown in the "Testing" task card for both the operating panel as well as for the breakpoints.

Resuming program execution

You have the following options for resuming program execution after reaching a breakpoint:

- Click the "RUN" button in the "Testing" task card under "Breakpoints".
- Select the "RUN" option from the shortcut menu in the opened Program editor.
- Use the keyboard command "Ctrl+Shift+F7" in the open Program editor.

Result

The CPU changes from "HOLD" to "RUN" and the normal program processing continues to the next breakpoint.

If it is not possible to resume program execution, you receive a corresponding alarm.

If no other breakpoints are enabled, the program execution resumes without any breakpoints.

23.4 Testing with the watch table

23.4.1 Introduction to testing with the watch table

Overview

The following functions are available in the watch table:

- **Monitoring tags**
  This allows the current values of the individual tags of a user program or a CPU to be displayed on the programming device or PC.
- **Monitoring tags**
  You can use this function to assign fixed values to the individual tags of a user program or CPU. Modifying is also possible with Test with program status (Page 7687).
- **"Enable peripheral outputs" and "Modify now"**
  These two functions enable you to assign fixed values to individual peripheral outputs of a CPU in the STOP mode. You can also use them to check your wiring.

Monitoring and modifying tags

The following tags can be monitored and modified:

- Inputs, outputs, and bit memories
- Contents of data blocks
- Content of UDTs
- I/O

Team engineering in the watch table

As of TIA Portal V13 SP1, within the framework of team engineering with an S7-1500 CPU with firmware version >= 1.7, several Engineering Systems can access the CPU online simultaneously, for example to monitor and modify tags and download blocks. If you use this function, make sure that you keep to the requirements and rules that apply to team engineering as explained in the information system in "Using Team Engineering" in the section "Shared commissioning of projects".

Possible applications

The advantage of the watch table is that a variety of test environments can be stored. This enables you to reproduce tests during commissioning or for service and maintenance purposes.

See also

Layout of the watch table (Page 7740)

Basic mode and expanded mode in the watch table (Page 7741)

Icons in the watch table (Page 7742)

Creating and editing watch tables (Page 7743)

23.4.2 Layout of the watch table

Introduction

A watch table contains the tags you defined for the entire CPU. A "Watch and force tables" folder is automatically generated for each CPU created in the project. You create a new watch table in this folder by selecting the "Add new Watch table" command.

Layout of the watch table

The columns displayed in the watch table depend on the mode you are working in: basic mode or expanded mode.

The following additional columns are shown in expanded mode:

- Monitor with trigger
- Modify with trigger

The names of the columns can also be changed dynamically based on the action.

Meaning of the columns

The following table shows the meaning of the individual columns in basic mode and expanded mode:

Mode	Column	Meaning
Basic mode	i	Identifier column
	Name	Name of the inserted tag
	Address	Address of the inserted tag
	Display format	Selected display format
	Monitor value	Values of the tags, depending on the selected display format.
	Modify value	Value with which the tag is modified.
		Select the tag to be modified by clicking the corresponding check box.
	Comment	Option to enter a comment to document the tags in the watch table.
	Tag comment	Display of the comment on the selected tag that was entered in other editors. This column cannot be edited.

Mode	Column	Meaning
The following additional columns are shown in expanded mode:	Monitor with trigger	Display of selected monitoring mode
	Modify with trigger	Display of selected modify mode

See also

Icons in the watch table (Page 7742)

23.4.3 Basic mode and expanded mode in the watch table

Difference between basic mode and expanded mode in the watch table

Depending on the mode specified, the watch table displays different columns and column headings that can be used to perform different actions.

You will find a detailed list of the columns in Layout of the watch table (Page 7740).

Switching between basic mode and expanded mode

You have the following options of toggling between the basic and expanded mode:

- Click the icon "Show/hide advanced setting columns". Click this icon again to return to the basic mode.
  Or:
- In the "Online" menu, select the "Expanded mode" check box. Deactivate this check box to return to the basic mode.

Functionality in expanded mode

The following functionality is only possible in expanded mode:

- Monitor with trigger
- Modify with trigger
- Enable peripheral outputs
- Monitor peripheral inputs
- Modify peripheral outputs

NOTICE
Danger of a time-out while monitoring peripheral inputs and controlling peripheral outputs Note that the monitoring of peripheral inputs and the controlling of peripheral outputs in the watch table can result in a time-out. The CPU assumes the "STOP" mode.

See also

Setting the monitoring and modify mode (Page 7755)

23.4.4 Icons in the watch table

Meaning of the icons

The following table shows the meaning of the icons in the watch table:

Icon	Meaning
	Identifies a table inside the project tree as a watch table.
	Shows information in the identifier column.
	Inserts a row before the selected row.
	Inserts a row after the selected row.
	Inserts a comment line above the selected row.
	Modifies the addresses of all selected tags immediately and once. This command is executed once and as quickly as possible without reference to a defined trigger point in the user program.
	Modifies the addresses of all selected tags with reference to a defined trigger point in the user program.
	Disables the command output disable of the peripheral outputs. You can then modify the peripheral outputs when the CPU is in STOP mode.
	Displays all columns of expanded mode. If you click this icon again, the columns of expanded mode will be hidden.
	Displays all modify columns. If you click this icon again, the modify columns will be hidden.
	Starts monitoring of the visible tags in the active watch table. The default setting for the monitoring mode in basic mode is "permanent". In expanded mode, you can set defined trigger points for the monitoring of tags.
	Starts monitoring of the visible tags in the active watch table. This command is executed immediately and the tags are monitored once.
	Displays the check box for the selection of tags to be modified.
	Indicates that the value of the selected tag has been modified to "1".
	Indicates that the value of the selected tag has been modified to "0".
	Indicates that the address is being used multiple times.
	Indicates that the substitute value is being used. Substitute values are values that are output to the process in case of signal output module faults or are used instead of a process value in the user program in case of signal input module faults. The substitute values can be assigned by the user (e.g., retain old value).
	Indicates that the address is blocked because it is already being modified.

Icon	Meaning
	Indicates that the address cannot be modified.
	Indicates that the address cannot be monitored.
	Indicates that an address is being forced.
	Indicates that an address is being partly forced.
	Indicates that an associated I/O address is being fully or partly forced.
	Indicates that an address cannot be fully forced. Example: It is indeed possible to force the address QW0:P, but it is not possible to force the address QD0:P since this address area is eventually not available on the CPU.
	Indicates that a syntax error occurred.
	Indicates that the address is selected but at the moment e.g. has not yet been modified.

See also

Layout of the watch table (Page 7740)

23.4.5 Creating and editing watch tables

23.4.5.1 Creating a watch table

Introduction

The watch table allows you to monitor and modify tags in the user program. Once you have created a watch table, you can save it, duplicate it, and print it and use it again and again to monitor and modify tags.

Requirement

A project is open.

Procedure

To create a watch table, follow these steps:

1. Click "Project view" in the status bar.
   The project view is displayed.
2. In the project tree, double-click the CPU for which you want to create a watch table.
3. Double-click the "Watch and force tables" folder and then the "Add new watch table" command.
   A new watch table is added.

4. In the "Name" column or in the "Address" column, enter the name or the absolute address for the tags that you want to monitor or modify.
5. You can select a display format from the drop-down list in the "Display format" column if you want to change this default setting.
6. Now decide whether you want to monitor or modify the entered tags and, if applicable, enter the desired values for modifying.

23.4.5.2 Opening a watch table

Requirement

A watch table has been created.

Procedure

To open a watch table, follow these steps:

1. Open the "Watch and force tables" folder below the desired CPU.
2. Double-click on the required watch table in the folder.

Result

The selected watch table opens.

23.4.5.3 Copying and pasting a watch table

Requirement

A watch table has been created.

Procedure

To copy a watch table, follow these steps:

1. Right-click the watch table that you want to copy.
2. In the context menu, select "Copy".
3. In the project tree, open the folder structure for the CPU in which you want to paste the copied watch table.
4. Right-click on the "Watch and force tables" folder.
5. In the context menu, select "Paste".
6. Alternatively, you can select the entire contents of the watch table and Drag & Drop it onto another watch table.

Result

A copy of the selected watch table is placed in the "Watch and force tables" folder of the relevant CPU.

23.4.5.4 Saving a watch table

Prerequisite

A watch table has been created.

Procedure

To save a watch table, follow these steps:

1. In the project tree select the watch table you want to save.
2. If you wish to change the preset name of the table, select the "Rename" command in the context menu and enter a new name for the table.
3. In the "Project" menu, select "Save". Note that this save operation will save the entire project.

Result

The contents of the watch table and the project are saved.

Note

You can reuse saved watch tables to monitor and modify tags when retesting your program.

23.4.6 Entering tags in the watch table

23.4.6.1 Basic information on entering tags in the watch table

Recommended procedure

- Select the tags whose values you want to monitor or modify, and enter them in the watch table.
- When entering tags into the watch table, please note that these tags must be previously defined in the PLC tag table.
- When entering tags, work from the outside to the inside. This means that you start by entering the tags for the inputs in the watch table. Then, you enter the tags that are affected by the inputs or that affect the outputs. Finally, you enter the tags for the outputs.

Example of filling out a watch table

- Enter the absolute address to be monitored or modified in the "Address" column.
- Enter the symbolic name for the tag in the "Name" column.
- Select the display format you require from the drop-down list in the "Display format" column, if you do not want to use the default setting.
- Now decide whether you want to monitor or modify the entered tags. Enter the desired values for modifying as well as a comment in the corresponding columns of the watch table.

Create comment row

If required, you can create a comment row by entering the string "//" in the "Name" column.

Syntax check

When you enter the tags in the watch table, the syntax of each cell is checked when you exit the cell. Incorrect entries are marked in red.

Note

When you place the mouse pointer in a cell marked in red, brief information is displayed with additional notes on the error.

See also

Permitted operands for the watch table (Page 7746)

Permissible modify values for the watch table (Page 7748)

23.4.6.2 Permitted operands for the watch table

Permissible operands for the watch table

The following table shows the operands that are permitted for the watch table:

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	BOOL	I1.0, Q1.7, M10.1 I0.0:P; Q0.0:P
Input/output/bit memory	BYTE	IB1/QB10/MB100 IB1:P; QB1:P
Input/output/bit memory	WORD	IW1; QW10; MW100 IW2:P; QW3:P
Input/output/bit memory	DWORD	ID4; QD10; MD100 ID2:P; QD1:P
Timers	TIMER	T1

Permitted operand	Example of data type	Example (International mnemonics)
Counters	COUNTER	C1
Data block	BOOL	DB1.DBX1.0
Data block	BYTE	DB1.DBB1
Data block	WORD	DB1.DBW1
Data block	DWORD	DB1.DBD1

Note

Please observe the following notes to work with the watch table.

- You cannot enter "DB0..." because it is used by the system!
- Peripheral outputs can be modified but not monitored.
- Peripheral inputs can be monitored but not modified.

Note

No access to peripheral bits for S7-300/400 CPUs

The watch table does not allow S7-300/400 CPUs to access peripheral bits, e.g. %I0.0:P; %Q0.0:P.

These bits cannot be watched or controlled.

Please use the bits in the process image instead, e.g. %I0.0; %Q0.0.

NOTICE

Danger of a time-out while monitoring peripheral inputs and controlling peripheral outputs

Note that the monitoring of peripheral inputs and the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

See also

Basic information on entering tags in the watch table (Page 7745)

Overview of the valid data types (Page 231)

23.4.6.3 Permissible modify values for the watch table

Entry of modify values in the watch table

The following table shows the operands that are permitted for the entry of modify values in the watch table:

Table 23-1 Bit operands

Possible bit operands	Example for permitted modify values
I1.0	True
M1.7	False
Q1.0	0
Q1.1:P	1
DB1.DBX1.1	2#0
M1.6	2#1

Table 23-2 Byte operands

Possible byte operands	Example for permitted modify values
IB1	2#0011_0011
MB12	B#16#1F
QB10	1F
QB11:P	'a'
DB1.DBB1	10

Table 23-3 Word operands

Possible word operands	Example for permitted modify values
IW1	2#0011_0011_0011_0011
MW12	W#16#ABCD
MW14	ABCD
QW10	B#(12, 34)
QW12:P	12345
DB1.DBW1	'ab'
MW16	S5T#9s_340ms
MW18	C#123
MW9	D#2006-12-31

Table 23-4 Double word operands

Possible double word operands	Example for permitted modify values
ID1	2#0011_0011_0011_0011_0011_0011_0011_0011
QD10	Dw#16#abcdef10

Possible double word operands	Example for permitted modify values
QD12:P	ABCDEF10
DB1.DBD2	b#(12,34,56,78)
MD8	L#-12
MD12	L#12
MD16	123456789
MD20	123456789
MD24	T#12s_345ms
MD28	Tod#1:2:34.567
MD32	P#e0.0

Table 23-5 Timers

Possible operands of the "Timer" type	Permitted control values	Explanation
T1	0 ms	Time value in milliseconds (ms)
T12	20 ms	Time value in milliseconds (ms)
T14	12400 ms	Time value in milliseconds (ms)
T16	S5t#12s300ms	Time value 12s 300 ms

Table 23-6 Counters

Possible operands of the "Counter" type	Permitted control values
C1	0
C14	20
C16	C#123

Notes on timers and counters

- Timers

  Note

  Modifying a timer influences only the value, not the status. Timer T1 can be modified to the value "0", but the result of logic operation for A T1 is not changed.

  The time sequences "s5t" and "s5time" can be written in both lower-case and upper-case characters.

- Counters

  Note

  Modifying a counter influences only the value, not the status. Counter C1 can be changed to the value "0", but the result of the logic operation for A C1 is not changed.

23.4.6.4 Overview of the display formats

Display formats in the watch table

The display format you select specifies the representation of a tag value.

When entering the address a display format is automatically preset. If you want to change this, you can select a display format from the drop-down list in the "Display formats" column. The drop-down list only offers the display formats which are valid for this data type. The display format that appears first in the list is the pre-selected format.

Example

The following table shows the 32-bit data types permitted for all CPU families in the watch table and their possible display formats:

Data type	Possible display formats
BOOL	Bool, Hex, BCD, Octal, Bin, Dec, Dec+/-
BYTE	Hex, BCD, Octal, Bin, Dec, Dec+/-, Character
WORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Unicode_Character, SIMATIC_Time, Date, Counter
DWORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Unicode_Character, Floating-point number, Time of day, Timer
SINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
INT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Time, Counter, Date
DINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer
USINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
UINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Time, Counter, Date
UDINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer
REAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character,Dec, Dec+/-, Dec_Sequence, Time of day, Timer
DATE	Date, Dec, Hex, Bin
TIME_OF_DAY	Time of day, Dec, Hex, Bin
TIME	Timer, Hex, Bin, Dec, Dec+/-
DATE_AND_TIME	Date and time,
TIMER	SIMATIC_Time, Hex, BCD, Bin
CHAR	Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
WCHAR	Unicode_Character, Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
STRING	Character string
WSTRING	Unicode_character string
POINTER	Pointer, Hex
COUNTER	Counter, Hex, BCD, Bin
S5TIME	SIMATIC_Time, Hex, BCD, Bin

For the S7-1200 CPU family, all 32-bit data types are permitted (see table above), as well as the 64-bit data type LREAL with the following possible display formats:

Data type	Possible display formats
LREAL	In a project created with TIA Portal < V12: Floating-point number Note: The display of LREAL is limited to 13 digits plus exponent.
LREAL	In a project created with TIA Portal >= V12: Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Time, Date and time Note: The display of LREAL is limited to 13 digits plus exponent.

For the S7-1500 CPU family, in addition to 32-bit data types, the 64-bit data types listed in the table are also permitted with the following possible display formats:

Data type	Possible display formats
LWORD	Hex, Octal, BCD, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Time, Date and time
LINT	Dec+/-, Dec, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Time, Date and time
ULINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Time, Date and time
LREAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Time, Date and time
LTIME	Timer, Dec+/-, Dec, Hex, Bin
LTOD	Time of day, Dec, Hex, Bin
LDT	Date and time, Dec, Hex, Bin

For more information, refer to the description of the valid data types (Page 231).

Note

Rounding of floating-point numbers

In the watch table, floating-point numbers are stored as binary numbers in IEEE format. Because not every floating point number (real, longreal) that can be displayed on the user interface can be mapped to the IEEE format, there is a possibility that floating-point numbers will be rounded. If a rounded floating-point number in the watch table is copied and, in turn, inserted in another input field, the rounding may cause a slight difference.

Note

Only symbolic addressing is possible

In the watch table, LongDataTypes such as LWORD or LREAL can only be addressed symbolically.

23.4.6.5 Selecting the display format for tags

Procedure

To select the display format of the tags, follow these steps:

1. Enter the desired address in the watch table.
2. Click the desired cell in the "Display format" column, and open the drop-down list.
   The permissible display formats are shown in the drop-down list.
3. Select the desired display format from the drop-down list.

   Note

   If the selected display format cannot be applied, then the last selected display format will be displayed automatically.

See also

Overview of the valid data types (Page 231)

23.4.6.6 Creating and editing comment lines

Basic principles of comment lines

In addition to the row related comments in the comment column, you can now also create complete comment lines to enhance the structure of the watch table.

The contents of the comment line are stored in the "Languages & Resources" folder in the "Project texts" tab and can be compiled in other project languages.

Alternatively, you have the following options to enter comment lines:

- Using the corresponding icon in the toolbar of the watch table.
- Using the menu command "Insert > Insert comment line".
- Using the shortcut menu command "Insert comment line".
- By entering the string "//" in the "Name" column of the watch table.

Inserting comment lines

To insert comment lines, follow these steps:

1. Open the watch table and enter the required addresses.
2. Insert a new comment line:
   - To do so, click the "Insert comment line" icon in the toolbar of the watch table.
   - Or click the menu command "Insert > Insert comment line".
   - Or click the shortcut menu command "Insert comment line".
     Result: A new comment line is inserted above the selected row in the watch table.
3. Enter the required comment in the comment line. The entered comment is shown in green.
4. To show all comments you entered, double-click "Project texts" in the project tree under "Languages & Resources".
5. If you are working in multi-lingual projects and want your comment to be translated into other languages, you can set the project languages required in addition to the editing language in the project tree under "Languages & Resources > Project languages".

Creating comment lines

To create comment lines, follow these steps:

1. Open the watch table and enter the required addresses.
2. Enter the character string "//" in a free row in the "Name" column. This turns the row into a comment line.
   No comment lines can be created in the other columns.
3. Enter the required comment in the comment line. The entered comment is shown in green.
4. To show all comments you entered, double-click "Project texts" in the project tree under "Languages & Resources".
5. If you are working in multi-lingual projects and want your comment to be translated into other languages, you can set the project languages required in addition to the editing language in the project tree under "Languages & Resources > Project languages".

Deleting comment lines

To delete comment lines, follow these steps:

1. Open a watch table containing comment lines.
2. Delete the entire comment with the introductory string "//", if you no longer required this.
3. Alternatively, delete only the introductory string "//". In this case the existing comment is retained and is displayed in the "Comment" column in the watch table.

Note

Deleting comment lines

When you delete comment lines the project languages and any existing translations for these comments are also deleted.

23.4.7 Monitoring tags in the watch table

23.4.7.1 Introduction to monitoring tags in the watch table

Introduction

The watch table allows you to monitor the tags of the configured input and output modules in the CPU, depending on the monitoring and modify mode (Page 7755) selected. To monitor tags, an online connection to the CPU must exist.

NOTICE
Danger of a time-out while monitoring peripheral inputs Note that the monitoring of peripheral inputs can result in a time-out. The CPU assumes the "STOP" mode.

Options for monitoring tags

The following options are available for monitoring tags:

- Monitor now
  This command starts the monitoring of the visible tags in the active watch table immediately and once only.
- Monitor all
  This command starts the monitoring of all visible tags in the active watch table, depending on the selected watch mode:
  - In basic mode, the monitoring mode is set to "permanent" by default.
  - In expanded mode, you can specify defined trigger points for the monitoring of tags.

    Note

    If the monitoring mode is changed while in expanded mode and then a switch is made to basic mode, the monitoring mode set before will also be applied in basic mode.

CPU-specific limitations when monitoring tags

The following CPU-specific differences exist:

- CPU S7-300/400:
  CPUs from this family can only monitor the first 30 characters of a string.
- CPU S7-1200/1500:
  CPUs from this family can monitor a string up to the total size of 254 characters.

23.4.7.2 Setting the monitoring and modify mode

Introduction

By selecting the monitoring and modify mode, you specify the trigger point and the duration of the tag monitoring in the watch table and the force table.

Possible monitoring and modify modes (duration of monitoring or modifying)

The following monitoring and modifying modes are available:

Trigger	Execution	CPU status	Duration
Permanent	Permanent During monitoring: The inputs are monitored at the end of the cycle and the outputs at the start of the cycle. During modify: The inputs are modified at the start of the cycle and the outputs at the end of the cycle.	RUN	Will be executed until the user stops the action or the online connection to the CPU is interrupted.
Permanently, at start of scan cycle	Permanently, at start of scan cycle	RUN	Will be executed until the user stops the action or the online connection to the CPU is interrupted.
Permanently, at end of scan cycle	Permanently, at end of scan cycle	RUN	Will be executed until the user stops the action or the online connection to the CPU is interrupted.
Permanently, at transition to STOP	Permanently, at transition from RUN to STOP	RUN > STOP	Will be executed until the user stops the action or the online connection to the CPU is interrupted.
Once only, at start of scan cycle	Once only, at start of scan cycle	RUN	Ends automatically after being executed once.
Once only, at end of scan cycle	Once only, at end of scan cycle	RUN	Ends automatically after being executed once.
Once only, at transition to STOP	Once only, at transition from RUN to STOP	STOP > RUN	Ends automatically after being executed once.

Special features when using "Permanent" mode

The "Permanent" mode is executed differently for the monitoring and modifying of tags.

- Monitoring: The inputs are monitored at the end of the cycle and the outputs at the start of the cycle.
- Modifying: The inputs are modified at the start of the cycle and the outputs at the end of the cycle.

Selecting the trigger point

The trigger points "Beginning of scan cycle", "End of scan cycle", and "Switch to stop" specify the time at which the tags are to be read from the CPU or updated in the CPU.

The following diagram shows the position of these trigger points:

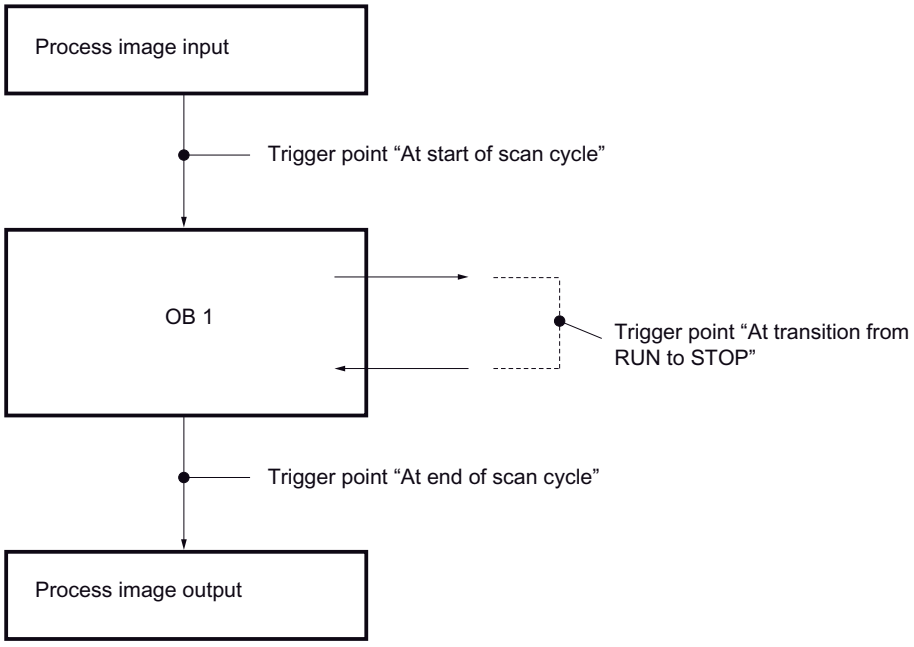

Position of the trigger points

From the position of the trigger points, it follows that:

- Modifying of inputs is only appropriate at the beginning of the scan cycle (corresponding to the beginning of the user program OB 1), because otherwise the process image input is updated again after modifying and is thus overwritten.
- Modifying of outputs is only appropriate at the end of the scan cycle (corresponding to the end of the user program OB 1), because otherwise the process image output can be overwritten by the user program.
- The modified value is displayed in the "Monitor value" column, provided that monitoring is active and the modified value is not overwritten by the user program.

Monitoring tags

When tags are being modified, the following applies to the trigger points:

- If you have specified the modify mode as "once only", you will receive an alarm if the selected tags cannot be modified.
- In "permanent" modify mode, you do not receive an alarm.

Note regarding the "Modify now" command

You can modify the values of selected tags immediately using the "Online > Modify >Modify now" command. This command is executed once only and as quickly as possible without reference to a defined position (trigger point) in the user program. This function is used mainly for modifying when the CPU is in STOP mode.

23.4.7.3 "Monitor all" command for tags

Introduction

The "Monitor all" command allows you to start monitoring the visible tags in the active watch table. The default setting for the monitoring mode in basic mode of the watch table is "permanent". In expanded mode, you can specify defined trigger points for the monitoring of tags. In this case, the tags are monitored with reference to the specified trigger points.

NOTICE
Danger of a time-out while monitoring peripheral inputs Note that the monitoring of peripheral inputs can result in a time-out. The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To execute the "Monitor all" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the watch table.
2. Switch to expanded mode by clicking
   the icon "Show/hide advanced setting columns" in the toolbar.
3. If you want to change the default monitoring mode for a tag, click the appropriate cell in the "Monitor with trigger" column and select the desired monitoring mode from the drop-down list.
4. Click the "Monitor all" icon in the toolbar.

Result

The tags of the active watch table are monitored using the monitoring mode selected.

See also

Icons in the watch table (Page 7742)
Basic mode and expanded mode in the watch table (Page 7741)
Entering tags in the watch table (Page 7745)

23.4.7.4 "Monitor now" command for tags

Introduction

The "Monitor now" command starts the monitoring of tags immediately without reference to defined trigger points. The tag values are read out once only and displayed in the watch table.

NOTICE
Danger of a time-out while monitoring peripheral inputs Note that the monitoring of peripheral inputs can result in a time-out. The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To execute the "Monitor now" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the watch table.
2. Click the "Monitor now" icon in the toolbar.

Result

The tags of the active watch table are monitored immediately and once only.

See also

Icons in the watch table (Page 7742)
Basic mode and expanded mode in the watch table (Page 7741)
Entering tags in the watch table (Page 7745)

23.4.8 Modifying tags in the watch table

23.4.8.1 Introduction to modifying tags

Introduction

The watch table allows you to modify the tags of the configured input and output modules in the CPU, depending on the monitoring and modify mode (Page 7755) selected.

To monitor the tags, an online connection to the CPU must exist.

DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

NOTICE

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

NOTICE

Danger due to modifying an identical operand in parallel with different modify values in more than one watch table

When working with more than one watch table, avoid modifying identical operands permanently multiple times with different modify values.

If an identical operand is modified permanently with different modify values at the same time in different watch tables, all watch tables will display the last modified value, because the modify value assigned last will be used in this case.

Options for modifying tags

The following options are available for modifying tags:

- Modify to "0"
  This command modifies the selected address to the modify value "0".
- Modify to "1"
  This command modifies the selected address to the modify value "1".
- Modify once only and immediately
  This command modifies all selected addresses in the active watch table "once only and immediately".
- Modify with trigger
  This command modifies all selected addresses in the active watch table using the monitoring and modify mode (Page 7755) selected.
  The "Modify with trigger" function is only available in expanded mode. You will not receive an alarm indicating whether or not the selected addresses were actually modified with the specified value. You should use the "Modify once only and immediately" function if you require such a confirmation.
- Enable peripheral outputs
  This command disables the command output disable.
  This function can only be executed in expanded mode, when the CPU is in STOP and the option Force (Page 7784) of tags is not enabled. If desired, deactivate this function in the force table.

Note

When modifying, note the following:

Modifying of tags **cannot** be undone.

23.4.8.2 Modify tags to "0"

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

NOTICE
Danger of a time-out while controlling peripheral outputs Note that the controlling of peripheral outputs in the watch table can result in a time-out. The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags to "0", follow these steps:

1. Enter the desired address in the watch table.
2. Select the "Online > Modify > Modify to 0" command in order to modify the selected address with the specified value.

Result

The selected address is modified to "0".

Note
When modifying, note the following:

Modifying can **not** be undone!

23.4.8.3 Modify tags to "1"

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

⚠ DANGER
Danger when modifying: Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors! Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

NOTICE
Danger of a time-out while controlling peripheral outputs Note that the controlling of peripheral outputs in the watch table can result in a time-out. The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags to "1", follow these steps:

1. Enter the desired address in the watch table.
2. Select the "Online > Modify > Modify to 1" command in order to modify the selected address with the specified value.

Result

The selected address is modified to "1".

Note

When modifying, note the following:

Modifying can **not** be undone!

23.4.8.4 "Modify now" command for tags

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them immediately. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

⚠ DANGER
Danger when modifying: Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors! Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

NOTICE
Danger of a time-out while controlling peripheral outputs Note that the controlling of peripheral outputs in the watch table can result in a time-out. The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags immediately, follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Select the "Online > Modify > Modify once and now" command in order to immediately modify the selected address once only with the specified value.

Result

The selected addresses are modified immediately and once only.

Note

When modifying, note the following:

Modifying can **not** be undone!

23.4.8.5 "Modify with trigger" command for tags

Introduction

You can assign values to addresses dependent on the defined monitoring and modify mode and modify them. The modify command is executed as specified in the monitoring and modify mode, with reference to the defined trigger position in the user program.

DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

NOTICE

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.
- The watch table has to be in expanded mode.

Procedure

To modify tags "with trigger", follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Switch to expanded mode using the icon "Show/hide advanced settings columns" in the toolbar or the "Online > Expanded mode" command.
   The "Monitor with trigger" and "Modify with trigger" columns are displayed.

4. In the "Modify with trigger" column, select the desired modify mode from the drop-down list box. The following options are available:
   - Permanent
   - Permanently, at start of scan cycle
   - Once only, at start of scan cycle
   - Permanently, at end of scan cycle
   - Once only, at end of scan cycle
   - Permanently, at transition to STOP
   - Once only, at transition to STOP
5. Start modifying using the "Online > Modify > Modify with trigger" command.
6. Confirm the prompt with "Yes" if you want to start modifying with trigger.

Result

The selected tags are modified using the selected monitoring and modify mode. The yellow triangle is no longer displayed.

Note

When modifying, note the following:

Modifying can **not** be undone!

23.4.8.6 Enable peripheral outputs

Introduction

The "Enable peripheral outputs" function deactivates the command output disable of the peripheral outputs. You can then modify the peripheral outputs when the CPU is in STOP mode.

This function is available in the watch table in "Expanded mode" only and only for CPUs of the S7-300/400 and S7-1200 family.

DANGER

Danger when enabling the peripheral outputs:

Attention, the enabling of the peripheral outputs can cause serious personal injury and material damage!
Make certain that dangerous conditions cannot occur before you execute the "Enable peripheral outputs" function.

Prerequisites

- A watch table has been created.
- An online connection to the CPU exists.
- The CPU is in STOP mode before you can enable the peripheral outputs.
- The watch table has to be in expanded mode.
- The option Force (Page 7784) of tags must not be enabled.

Note

"Enable peripheral outputs" function

- This function is possible only in STOP mode. The function is exited by an operating state change of the CPU and by the termination of the online connection.
- While the function is enabled, forcing is not possible.

Procedure

To enable the peripheral outputs in STOP mode, follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Switch to expanded mode using the icon "Show/hide advanced settings columns" in the toolbar or the "Online > Expanded mode" command.
   The "Monitor with trigger" and "Modify with trigger" columns are displayed.
4. Change the relevant CPU to STOP using the operator panel.
5. Right-click to open the shortcut menu and select "Enable peripheral outputs".
6. Confirm the prompt with "Yes" if you want to unlock the command output disable for the peripheral outputs.
7. Modify the peripheral outputs using the "Online > Modify > Modify now" command.

Result

The peripheral outputs are modified with the selected modify values. The yellow triangle is no longer displayed.

Enabling the peripheral outputs

The "Enable peripheral outputs" function remains active until:

- The "Enable peripheral outputs" command is deactivated again via the shortcut menu or via the "Online > Modify > Enable peripheral outputs" command.
- The CPU is no longer in STOP mode.
- The online connection is terminated.

Note

When modifying, note the following:

Modifying can **not** be undone!

23.5 Testing with the force table

23.5.1 Introduction for testing with the force table

Overview

You can use the force table to assign permanent values to individual tags

of the user program. This action is referred to as "forcing".

The following functions are available in the force table:

- Monitoring tags
  This allows the current values of the individual tags of a user program or a CPU to be displayed on the programming device or PC. Tags can be monitored with or without a trigger condition.
- Forcing tags
  This function lets you assign a fixed value to individual I/O tags of the user program.

Monitoring and forcing tags

The monitoring and forcing of tags is always dependent on the operand scope of the CPU used.

The following tags can be monitored:

- Inputs, outputs, and bit memories
- Contents of data blocks
- Peripheral inputs

The following tags can be forced:

- Peripheral inputs
- Peripheral outputs

Example

Independent of the CPU used, only I/O can be forced, such as: "Tag_1":P or "QW0:P" or "IW0:P". Note that "Tag_1":P must not be the symbolic name of a bit memory.

Possible applications

One advantage of the force table is that you can simulate different test environments and overwrite tags in the CPU with a permanent value. This enables you to intervene in the ongoing process for regulating purposes.

See also

Layout of the force table (Page 7770)
Basic mode and expanded mode in the force table (Page 7771)
Icons in the force table (Page 7771)
Open and edit force table (Page 7772)

23.5.2 Safety precautions when forcing tags

Safety precautions when forcing tags

Because the forcing function allows you to intervene permanently in the process, observance of the following notices is essential:

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

CAUTION

Prevent personal injury and material damage!

- Before you start the "Force" function, you should ensure that no one else is currently executing this function on the same CPU.
- Forcing can only be stopped by clicking the "Stop forcing" icon or using the "Online > Force > Stop forcing" command. Closing the active force table does **not** stop the forcing!
- Forcing can **not** be undone!
- Review the differences between " modifying tags" (Page 7759) and "forcing tags" (Page 7784).
- If a CPU does not support the "Force" function, the relevant icons cannot be selected.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is **not** possible on this CPU. If desired, deactivate this function in the watch table.

23.5.3 Layout of the force table

Introduction

In the force table, enter the CPU-wide tags that you have defined and selected and which are to be forced in the allocated CPU. Only peripheral inputs and peripheral outputs can be forced.

For each CPU created in the project, a force table will automatically be created in the "Watch and force tables" folder. Only one force table can be allocated to a CPU. This force table displays all the addresses forced in the allocated CPU.

Layout of the force table

The columns displayed in the force table depend on the mode you are working in: basic mode or expanded mode.

In expanded mode the "Monitor with trigger" column is also displayed.

Meaning of the columns

The following table shows the meaning of the individual columns in basic mode and expanded mode:

Mode	Column	Meaning
Basic mode	i	Identification column
	Name	Name of the inserted tag
	Address	Address of the inserted tag
	Display format	Selected display format
	Monitor value	Values of the tags, dependent on the selected display format.
	Force value	Value with which the tag is forced.
	F ("Force")	Select the tag to be forced by activating the corresponding check box.
	Comment	Option to enter a comment to document the tags in the force table.
	Tag comment	Display of the comment on the selected tag that was entered in other editors. This column cannot be edited.
The following additional column is shown in expanded mode:	Monitor with trigger	Display of selected monitoring mode

See also

Icons in the force table (Page 7771)

Basic mode and expanded mode in the force table (Page 7771)

23.5.4 Basic mode and expanded mode in the force table

Difference between basic mode and expanded mode in the force table

In expanded mode the "Monitor with trigger" column is also displayed in the force table.

You will find a detailed list of the columns under Layout of the force table (Page 7770).

Switching between basic mode and expanded mode

You have the following options of toggling between the basic and expanded mode:

- Click the icon "Show/hide advanced setting columns". Click this icon again to return to the basic mode.
  Or:
- In the "Online" menu, select the "Expanded mode" check box. Deactivate this check box to return to the basic mode.

Functionality in expanded mode

The following functionality is only possible in expanded mode:

- Monitor with trigger
- Monitor peripheral inputs

23.5.5 Icons in the force table

Meaning of the icons

The following table shows the meaning of the icons in the force table:

Icon	Meaning
	Identifies a table inside the project tree as a force table.
	Identification column
	Inserts a row before the selected row.
	Inserts a row after the selected row.
	Inserts a comment line above the selected row.
	Displays all columns of expanded mode. If you click this icon again, the columns of the expanded mode will be hidden.
	Updates all operands and values currently being forced on the CPU in the open force table.
	Starts forcing for all addresses of the selected tags. If forcing is already running, the previous action is replaced without interruption.

Icon	Meaning
	Stops forcing of addresses in the force table.
	Starts monitoring of the visible tags in the force table. The default setting for monitoring in basic mode is "permanent". In expanded mode an additional column is shown and you can set certain trigger points for monitoring tags.
	Starts monitoring of the visible tags in the force table. This command is executed immediately and the tags are monitored once.
	Displays the check box for the selection of tags to be forced.
	Indicates that an address cannot be fully forced. Example: It is possible to force the address QW0:P, but it is not possible to force the address QD0:P because this address area is potentially not available on the CPU.
	Indicates that an address cannot be monitored.
	Indicates that an address is being forced.
	Indicates that an address is being partly forced.
	Indicates that the associated peripheral address is being forced.
	Indicates that a syntax error occurred.
	Indicates that the address is selected but has not been forced yet.
	Indicates that the selected tag was monitored for the value "1".
	Indicates that the selected tag was monitored for the value "0".
	Indicates that the address is being used more than once.
	Indicates that a substitute value is being used. Substitute values are values that are output to the process in case of signal output module faults or are used instead of a process value in the user program in case of signal input module faults. The substitute values can be assigned by the user (e.g., retain old value).

See also

Layout of the force table (Page 7770)

23.5.6 Open and edit force table

23.5.6.1 Display force table

Introduction

You cannot create a new force table; one force table already exists for each CPU. It is permanently allocated to this CPU and cannot be copied or duplicated.

Requirements

A project with an allocated CPU has to be open.

Displaying a force table

The force table is always displayed below a CPU in the "Watch and force tables" folder.

23.5.6.2 Open force table

Requirements

A project with an allocated CPU must be created.

Procedure

Proceed as follows to open a force table:

1. Open the "Watch and force tables" folder below the desired CPU.
2. Double-click the "Force table" in this folder.

Result

The selected force table opens.

23.5.6.3 Save force table

Requirements

A project with an allocated CPU has been created.

Procedure

Proceed as follows to save a force table:

1. Enter the desired changes in the force table.
2. Select the "Save" command in the "Project" menu or click the "Save project" icon in the toolbar. Note that this save operation will save the entire project.

Result

The contents of the force table and the associated project are saved.

Note

You cannot rename a force table.

23.5.7 Entering tags in the force table

23.5.7.1 Basic principles for entering tags in the force table

Recommended procedure

Select the tags whose values you want to monitor or force, and enter them in the force table.

When entering tags in the force table, please note that these tags must be previously defined in the PLC tag table.

Example of filling out a force table

- You can enter the absolute address that is to be forced or monitored in the "Address" column or you can enter the symbolic name of the tag in the "Name" column.
- Select the display format you require from the drop-down list in the "Display format" column, if you do not want to use the default setting.
- Now you have to decide whether you want to monitor or force the entered tags. Enter the required force value and a comment in the appropriate columns of the force table.
- Note that only peripheral inputs and peripheral outputs can be forced and review the Safety precautions when forcing tags (Page 7786).

Create comment line

If required, you can create a comment row by entering the string "//" in the "Name" column.

Syntax check

When you enter tags in the force table, the syntax of each cell will be checked when you exit the cell. Incorrect entries are marked in red.

Note

When you place the mouse pointer in a cell marked in red, brief information is displayed with additional notes on the error.

23.5.7.2 Permitted operands for the force table

Permitted operands for the force table

The following table shows the operands that are permitted for forcing in the force table:

Permitted operand	Example of data type	Example (International mnemonics)
Peripheral input/peripheral output	BOOL	I0.0:P; Q0.0:P
Peripheral input/peripheral output	BYTE	IB1:P; QB1:P
Peripheral input/peripheral output	WORD	IW2:P; QW3:P
Peripheral input/peripheral output	DWORD	ID2:P; QD1:P

The following table shows the operands that are permitted for monitoring in the force table:

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	BOOL	I1.0, Q1.7, M10.1 I0.0:P
Input/output/bit memory	BYTE	IB1/QB10/MB100 IB1:P
Input/output/bit memory	WORD	IW1; QW10; MW100 IW2:P
Input/output/bit memory	DWORD	ID4; QD10; MD100 ID2:P
Timers	TIMER	T1
Counters	COUNTER	C1
Data block	BOOL	DB1.DBX1.0
Data block	BYTE	DB1.DBB1
Data block	WORD	DB1.DBW1
Data block	DWORD	DB1.DBD1

Note

You cannot enter "DB0..." because it is used by the system!

Note

No access to peripheral bits for S7-400 CPUs

The force table does not allow S7-400 CPUs to access peripheral bits, e.g. %I0.0:P; %Q0.0:P.

These bits cannot be forced.

23.5.7.3 Permitted force values for the force table

Entering force values in the force table

The following table shows the operands that are permitted for entering force values in the force table:

Table 23-7 Bit operands

Possible bit operands	Example for permitted force values
I1.0:P	True
I1.1:P	False
Q1.0P	0
Q1.1:P	1
I2.0:P	2#0
I2.1:P	2#1

Table 23-8 Byte operands

Possible byte operands	Example for permitted force values
IB1:P	2#00110011
IB2:P	B#16#1F
QB14:P	1F
QB10:P	'a'
IB3:P	10

Table 23-9 Word operands

Possible word operands	Example for permitted force values
IW0:P	2#0011001100110011
IW2:P	W#16#ABCD
QW10:P	ABCD
QW12:P	B#(12, 34)
IW4:P	'ab'
IW6:P	12300
IW8:P	S5T#9S_300ms
IW10:P	C#123
IW12:P	D#2006-12-31

Table 23-10 Double word operands

Possible double word operands	Example for permitted force values
ID0:P	2#00110011001100110011001100110011
ID4:P	1.2

Possible double word operands	Example for permitted force values
QD10:P	1.234.e4
QD14:P	Dw#16#abcdef10
ID8:P	16#ABCDEF10
ID12:P	b#(12,34,56,78)
ID16:P	L#-12
ID20:P	L#12
ID24:P	123456789
ID28:P	123456789
ID32:P	T#12s300ms
ID36:P	Tod#14:20:40.645
ID40:P	P#e0.0

23.5.7.4 Overview of the display formats

Display formats in the force table

The display format you select specifies the representation of a tag value.

When entering the address a display format is automatically preset. If you want to change this, you can select a display format from the drop-down list in the "Display formats" column. The drop-down list only offers the display formats which are valid for this data type. The display format that appears first in the list is the pre-selected format.

Example

The following table shows the 32-bit data types permitted for all CPU families in the force table and their possible display formats:

Data type	Possible display formats
BOOL	Bool, Hex, BCD, Octal, Bin, Dec, Dec+/-
BYTE	Hex, BCD, Octal, Bin, Dec, Dec+/-, Character
WORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, SIMATIC_Time, Date, Unicode_Character, Counter
DWORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Floating-point number, Time of day, Timer, Unicode_Character
SINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
INT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Time, Counter, Date
DINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer
USINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
UINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Time, Counter, Date
UDINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer

Data type	Possible display formats
REAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer
DATE	Date, Dec, Hex, Bin
TIME_OF_DAY	Time of day, Dec, Hex, Bin
TIME	Timer, Hex, Bin, Dec, Dec+/-
DATE_AND_TIME	Date and time,
TIMER	SIMATIC_Time, Hex, BCD, Bin
CHAR	Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
WCHAR	Unicode_Character, Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
STRING	Character string
WSTRING	Unicode_character string
POINTER	Pointer, Hex
COUNTER	Counter, Hex, BCD, Bin
S5TIME	SIMATIC_Time, Hex, BCD, Bin

For the S7-1200 CPU family, all 32-bit data types are permitted (see table above), as well as the 64-bit data type LREAL with the following possible display formats:

Data type	Possible display formats
LREAL	In a project created with TIA Portal < V12: Floating-point number Note: The display of LREAL is limited to 13 digits plus exponent.
LREAL	In a project created with TIA Portal >= V12: Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Time, Date and time Note: The display of LREAL is limited to 13 digits plus exponent.

For the S7-1500 CPU family, in addition to 32-bit data types, the 64-bit data types listed in the table are also permitted with the following possible display formats:

Data type	Possible display formats
LWORD	Hex, Octal, BCD, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Time, Date and time
LINT	Dec+/-, Dec, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Time, Date and time
ULINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Time, Date and time
LREAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Time, Date and time
LTIME	Timer, Dec+/-, Dec, Hex, Bin

Data type	Possible display formats
LTOD	Dec, Hex, Bin, Time of day
LDT	Dec, Hex, Date and time, Bin

For more information, refer to the description of the valid data types (Page 231).

Note

Rounding of floating-point numbers

In the force table, floating-point numbers are stored as binary numbers in IEEE format. Because not every floating point number (real, longreal) that can be displayed on the user interface can be mapped to the IEEE format, there is a possibility that floating-point numbers will be rounded. If a rounded floating-point number in the force table is copied and, in turn, inserted in another input field, the rounding may cause a slight difference.

Note

Only symbolic addressing is possible

In the force table, LongDataTypes such as LWORD or LREAL can only be addressed symbolically.

23.5.7.5 Selecting the display format for tags

Procedure

To select the display format of the tags, follow these steps:

1. Enter the desired address in the force table.
2. Click the desired cell in the "Display format" column, and open the drop-down list.
   The permitted display formats are shown in the drop-down list.
3. Select the desired display format from the drop-down list.

   Note

   If the selected display format cannot be applied, then the last selected display format will be displayed automatically.

23.5.7.6 Creating and editing comment lines

Basic principles of comment lines

In addition to the row-related comments in the comment column, you can now also create complete comment lines to enhance the structure of the force table.

The contents of the comment line are stored in the "Languages & Resources" folder in the "Project texts" tab and can be compiled in other project languages.

- Using the corresponding icon in the toolbar of the force table.
- Using the menu command "Insert > Insert comment line".
- Using the shortcut menu command "Insert comment line".
- By entering the string "//" in the "Name" column of the force table.

Inserting comment lines

To insert comment lines, follow these steps:

1. Open the force table and enter the required addresses.
2. Insert a new comment line:
   - To do so, click the "Insert comment line" icon in the toolbar of the force table.
   - Or click the menu command "Insert > Insert comment line".
   - Or click the shortcut menu command "Insert comment line".
     Result: A new comment line is inserted above the selected row in the force table.
3. Enter the required comment in the comment line. The entered comment is shown in green.
4. To show all comments you entered, double-click "Project texts" in the project tree under "Languages & Resources".
5. If you are working in multi-lingual projects and want your comment to be translated into other languages, you can set the project languages required in addition to the editing language in the project tree under "Languages & Resources > Project languages".

Creating comment lines

To create comment lines, follow these steps:

1. Open the force table and enter the required addresses.
2. Enter the character string "//" in a free row in the "Name" column. This turns the row into a comment line.
   No comment lines can be created in the other columns.
3. Enter the required comment in the comment line. The entered comment is shown in green.
4. To show all comments you entered, double-click "Project texts" in the project tree under "Languages & Resources".
5. If you are working in multi-lingual projects and want your comment to be translated into other languages, you can set the project languages required in addition to the editing language in the project tree under "Languages & Resources > Project languages".

Deleting comment lines

To delete comment lines, follow these steps:

1. Open a force table containing comment lines.
2. Delete the entire comment with the introductory string "//", if you no longer required this.
3. Alternatively, delete only the introductory string "//". In this case the existing comment is retained and is displayed in the "Comment" column in the force table.

Note

Deleting comment lines

When you delete comment lines the project languages and any existing translations for these comments are also deleted.

23.5.8 Monitoring tags in the force table

23.5.8.1 Introduction to monitoring tags in the force table

Introduction

Use the force table to monitor the tags of the configured input and output modules in the CPU, dependent on the monitoring mode (Page 7782) you have selected. An online connection to the CPU must exist to monitor tags.

Options for monitoring tags

The following options are available for monitoring tags:

- Monitor all
  This command starts the monitoring of all visible tags in the active force table, dependent on the selected monitoring mode:
  - In basic mode, the monitoring mode is set to "permanent" by default.
  - In expanded mode, you can specify defined trigger points for the monitoring of tags.

  Note

  If the monitoring mode is changed while in expanded mode and then a switch is made to basic mode, the monitoring mode set before will also be applied in basic mode.

- Monitor now
  This command starts the monitoring of the visible tags in the active force table immediately and once only.

CPU-specific limitations when monitoring tags

The following CPU-specific differences exist:

- CPU S7-300/400:
  CPUs from this family can only monitor the first 30 characters of a string.
- CPU S7-1200:
  CPUs from this family can monitor a string up to the total size of 254 characters.

23.5.8.2 Setting the monitoring mode in the force table

Introduction

By selecting the monitoring mode, you specify the trigger point and the duration of tag monitoring in the force table.

Possible monitoring mode (duration of monitoring)

The following selection options are available:

- Permanent
- Once only, at start of scan cycle
- Once only, at end of scan cycle
- Permanently, at start of scan cycle
- Permanently, at end of scan cycle
- Once only, at transition to STOP
- Permanently, at transition to STOP

Special features when using "Permanent" mode

The "Permanent" mode is executed as follows for the monitoring of tags: The inputs are monitored at the end of the cycle and the outputs at the start of the cycle.

Selecting the trigger point

The trigger points "Beginning of scan cycle", "End of scan cycle", and "Switch to stop" specify the time at which the tags are to be read from the CPU or updated in the CPU.

The following diagram shows the position of these trigger points:

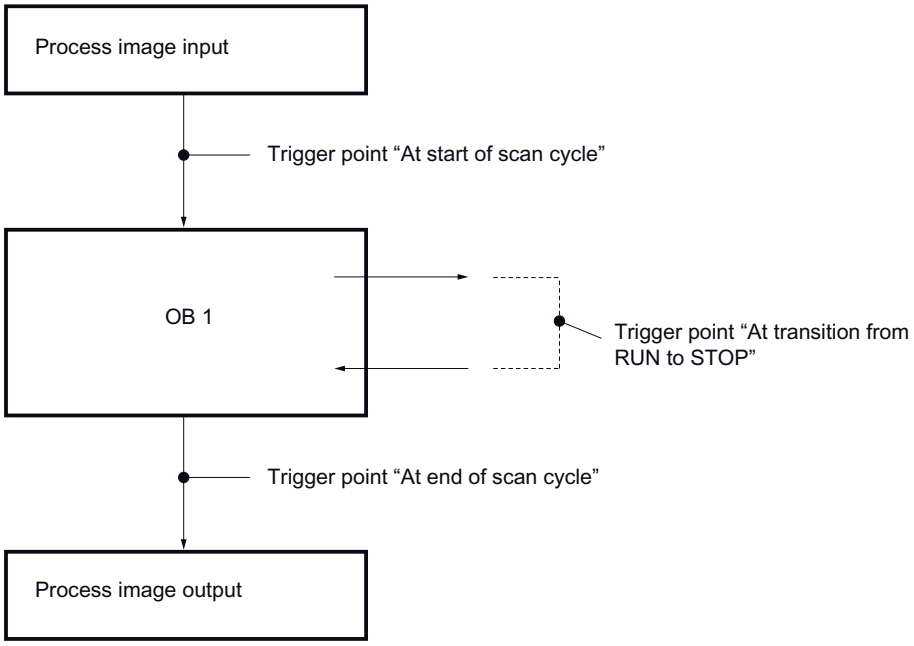

23.5.8.3 "Monitor all" command for tags

Introduction

Use the "Monitor all" command to start monitoring the visible tags in the active force table. In basic mode of the force table, the default setting for the monitoring mode is "permanent". In expanded mode, you can specify defined trigger points for the monitoring of tags. In this case, the tags are monitored with reference to the specified trigger points.

Requirements

An online connection to the CPU exists.

Procedure

To execute the "Monitor all" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the force table.
2. Switch to expanded mode by clicking
   the icon "Show/hide advanced setting columns" in the toolbar.
3. If you want to change the default monitoring mode for a tag, click the appropriate cell in the "Monitor with trigger" column and select the desired monitoring mode from the drop-down list.
4. Click the "Monitor all" icon in the toolbar.

Result

The tags of the active force table will be monitored using the set monitoring mode.

23.5.8.4 "Monitor now" command for tags

Introduction

The "Monitor now" command starts the monitoring of tags immediately without reference to defined trigger points. The tag values are read out only once and displayed in the force table.

Requirements

An online connection to the CPU exists.

Procedure

To execute the "Monitor now" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the force table.
2. Click the "Monitor now" icon in the toolbar.

Result

The tags of the active force table are monitored immediately and once only.

23.5.9 Forcing tags in the force table

23.5.9.1 Introduction to forcing tags

Introduction

You can use the force table to assign permanent values to individual tags of the user program. This action is referred to as forcing. Only peripheral inputs and peripheral outputs can be forced.

To use the forcing function, you must have an online connection to the CPU and the utilized CPU must support this functionality.

If you open a force table in the "Watch and force tables" folder below a CPU on which a force job is already running, you will first be prompted to update the display of the forced operand. Forcing or stopping forcing in the open force table is only possible following this command.

Possible applications

By permanently assigning defined values to tags, you can specify defined default settings for your user program and, thus, test the programmed functions. Forcing is possible in basic mode and in expanded mode (Page 7771).

Caution when forcing tags

Before forcing, make sure that you review the safety precautions for this procedure (Page 7786).

DANGER

Prevent personal injury and material damage!

Remember that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Options for forcing tags

The following options are available for forcing tags:

- Force to "0"
  This command forces the selected address in the CPU to the force value "0".
- Force to "1"
  This command forces the selected address in the CPU to the force value "1".
- Force all
  This command starts the forcing of enabled addresses in the active force table or replaces an existing force job without interruption.
- Stop forcing
  This command stops the forcing of all addresses in the active force table.

Constraints when forcing tags

Note the following constraints when forcing:

- Forcing is always dependent on the operand scope of the CPU used.
- In principle, only peripheral inputs and peripheral outputs can be forced.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is not possible. If desired, deactivate this function in the watch table.

Unique aspects when forcing tags

Note that forcing of tags will overwrite values in the CPU and will continue even after the online connection to the CPU is terminated.

- Stop forcing
  Terminating the online connection is not sufficient to stop the forcing operation! To stop forcing, you must select the "Online > Force > Stop forcing" command. Only then will the tags that are visible in the active force table no longer be forced.
- Stop forcing of individual tags
  The "Online > Force > Stop forcing" command always applies to all tags displayed in the force table. To stop forcing individual tags, you must clear the check mark for forcing of these tags in the force table and restart forcing using the "Online > Force > Force all" command.

23.5.9.2 Safety precautions when forcing tags

Safety precautions when forcing tags

Because the forcing function allows you to intervene permanently in the process, observance of the following notices is essential:

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

CAUTION

Prevent personal injury and material damage!

- Before you start the "Force" function, you should ensure that no one else is currently executing this function on the same CPU.
- Forcing can only be stopped by clicking the "Stop forcing" icon or using the "Online > Force > Stop forcing" command. Closing the active force table does **not** stop the forcing!
- Forcing can **not** be undone!
- Review the differences between " modifying tags" (Page 7759) and "forcing tags" (Page 7784).
- If a CPU does not support the "Force" function, the relevant icons cannot be selected.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is **not** possible on this CPU. If desired, deactivate this function in the watch table.

23.5.9.3 Updating forced operands

Introduction

If a force job is already running on a CPU, after opening the force table, you first need to make sure that the operands and values currently being forced on the CPU are displayed in the force table.

The command "Online" > "Force" > "Update forced operands" updates all operands and values currently being forced on the CPU in the open force table.

"Force" or "Stop forcing" in the open force table is only possible following this command.

Caution when forcing tags

Before forcing, make sure you are familiar with the safety precautions when forcing tags (Page 7786).

DANGER

Prevent personal injury and material damage!

Remember that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Requirement

- An online connection to the CPU is possible.
- A force job is currently running on the CPU being used.

Procedure

To update the forced operands and values, follow these steps:

1. Open a force table.
2. Establish an online connection to the CPU.
3. Confirm the "Update forced operands" dialog that follows with "Yes".

Result

All forced operands in the open force table are updated with the relevant values. A red "F" is displayed in the first column indicating which operands are being forced.

This enables the "Force all" and "Stop forcing" buttons and you can select these functions.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing function!
- To stop forcing, the forced address must be visible in the active force table.

23.5.9.4 Force tags to "0"

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 7786).

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags to "0", follow these steps:

1. Open the force table.
2. Enter the desired address in the force table.

3. Select the "Online > Force> Force to 0" command in order to force the selected address with the specified value.
4. Confirm the next dialog with "Yes".

Result

The selected address is forced to "0". The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected values is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

23.5.9.5 Force tags to "1"

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 7786).

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Endanger the life or health of personnel
- Cause damage to machinery or the entire plant.

Requirements

- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags to "1", follow these steps:

1. Open the force table.
2. Enter the desired address in the force table.
3. Select the "Online > Force> Force to 1" command in order to force the selected address with the specified value.
4. Confirm the next dialog with "Yes".

Result

The selected address is forced to "1". The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected values is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

23.5.9.6 "Force all" command for tags

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

If forcing is already active, this forcing operation is replaced without interruption by the "Online > Force > Force all" command. Any forced addresses that are not selected will no longer be forced.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 7786).

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags with the "Online > Force > Force all" command, follow these steps:

1. Open the force table.
2. Enter the desired addresses and force values in the force table.
3. Select the addresses to be forced by selecting the check boxes for forcing in the column after the "Force value".
   A yellow triangle appears behind the selected check box, indicating that the address is selected for forcing but is not being forced at the moment.
4. Select the "Online > Force> Force all" command in order to force the selected addresses with the specified values.
5. Confirm the next dialog with "Yes".

Result

The selected addresses are forced to the specified values. The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected addresses is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

23.5.10 Stop forcing tags

23.5.10.1 Stop forcing all tags

Introduction

Note the following before you stop forcing tags:

- Stopping forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 7786).

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when stopping the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- Tags are forced in a force table.
- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is not enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

Proceed as follows to stop **forcing all tags** :

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command in order to stop forcing the displayed addresses.
3. Confirm the "Stop forcing" dialog with "Yes".

Result

The forcing of all tags is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is flagged for forcing but is not being forced at the moment.

23.5.10.2 Stop forcing individual tags

Introduction

Note the following before you stop forcing tags:

- Stopping forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 7769).

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when stopping the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- Tags are forced in a force table.
- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is not enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

Proceed as follows to stop **forcing individual tags** :

1. Open the force table.
2. Deactivate the check boxes for the addresses that are no longer to be forced.
3. Reselect the "Online > Force" command.

Result

Forcing of the disabled addresses will be stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is flagged for forcing but is not being forced at the moment.

Supervision of machinery and plants with ProDiag (S7-1500)

24

24.1 System requirements for ProDiag (S7-1500)

24.1.1 Notes on licenses (S7-1500)

Description

The configuration of supervisions is completely integrated in STEP 7 Professional. It can be used on any STEP 7 engineering station and independent of the application. Only the number of configured supervisions is licensed, whereby the first 25 supervisions can be used free of charge.

The table below shows you how many licenses you need:

Number of supervisions in use	<= 25	<= 250	<= 500	<= 750	<= 1000	> 1000
Number of licenses required	None	1	2	3	4	5

For additional information on general allocation of licenses is available at: "Installation > System requirements for installation > Notes on licenses".

24.1.2 Setting ProDiag licenses (S7-1500)

Requirement

You have created an S7-1500 CPU.

Procedure

Proceed as follows to set the licenses for ProDiag:

1. Double-click the Device configuration of the S7-1500 CPU in the project tree.
   The Device configuration is opened in the Device view.
2. Click the subordinate tab "General" in the "Properties" Inspector window.
3. Click the arrow next to the "Runtime licenses" section.
   The available Runtime licenses are displayed.

4. Click the "ProDiag" entry.
   All the supervisions used within the S7-1500 CPU are displayed in the "Supervisions" section.
5. Select the number of licenses required in the "Runtime licenses" section in accordance with the number of supervisions used.

Result

The required licenses are set.

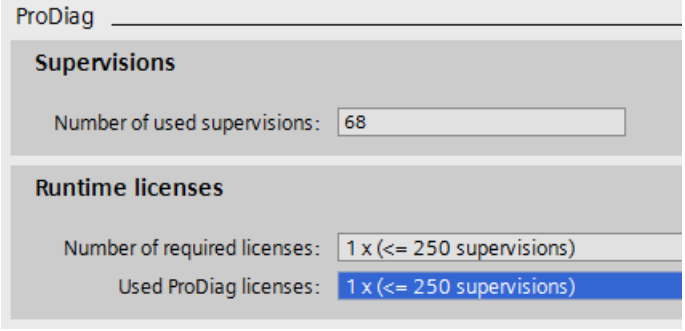

24.1.3 Software and hardware requirements (S7-1500)

Requirements for ProDiag

The table below shows the minimum software and hardware requirements that have to be met for use of ProDiag:

Software/Hardware	Requirement
TIA Portal	STEP 7 Professional
CPU of the S7-1500 series	>= Firmware V2.0

24.2 Technical specifications for ProDiag (S7-1500)

24.2.1 Technical specifications for ProDiag (S7-1500)

Description

The following table shows the maximum values possible:

Object	Number of objects
ProDiag function blocks	There is a maximum of 100 ProDiag FBs that can be used in a project.
ProDiag supervisions	ProDiag FB V1.0: A ProDiag FB can be assigned a maximum of 250 supervisions. ProDiag FB V2.0: A ProDiag FB can be assigned a maximum of 1000 supervisions.

24.3 Basics of supervision with ProDiag (S7-1500)

24.3.1 Introduction to ProDiag (S7-1500)

Description

Starting in the TIA Portal Version V14, you have the option of supervising your machine or plant using the ProDiag functionality and intervening in case of a fault. The supervision alarms that you can create for different kinds of faults give you specific information on the supervision type, on the location and cause of the fault. You can also output notes on the removal of the determined faults. This means you cannot only detect faults, but you can also identify a possible risk of error in advance and take appropriate countermeasures.

You can integrate simple supervisions in your program with just a few configuring steps and without having to change the program code. For example, you can create a supervision by monitoring a Boolean operand for its signal state. As soon as the operand has the set signal state, a ProDiag supervision alarm is output based on the configured ProDiag supervision settings. The configuration of the supervisions is independent of the programming languages of the TIA Portal because only individual operands are supervised and you do not need any additional programming sections.

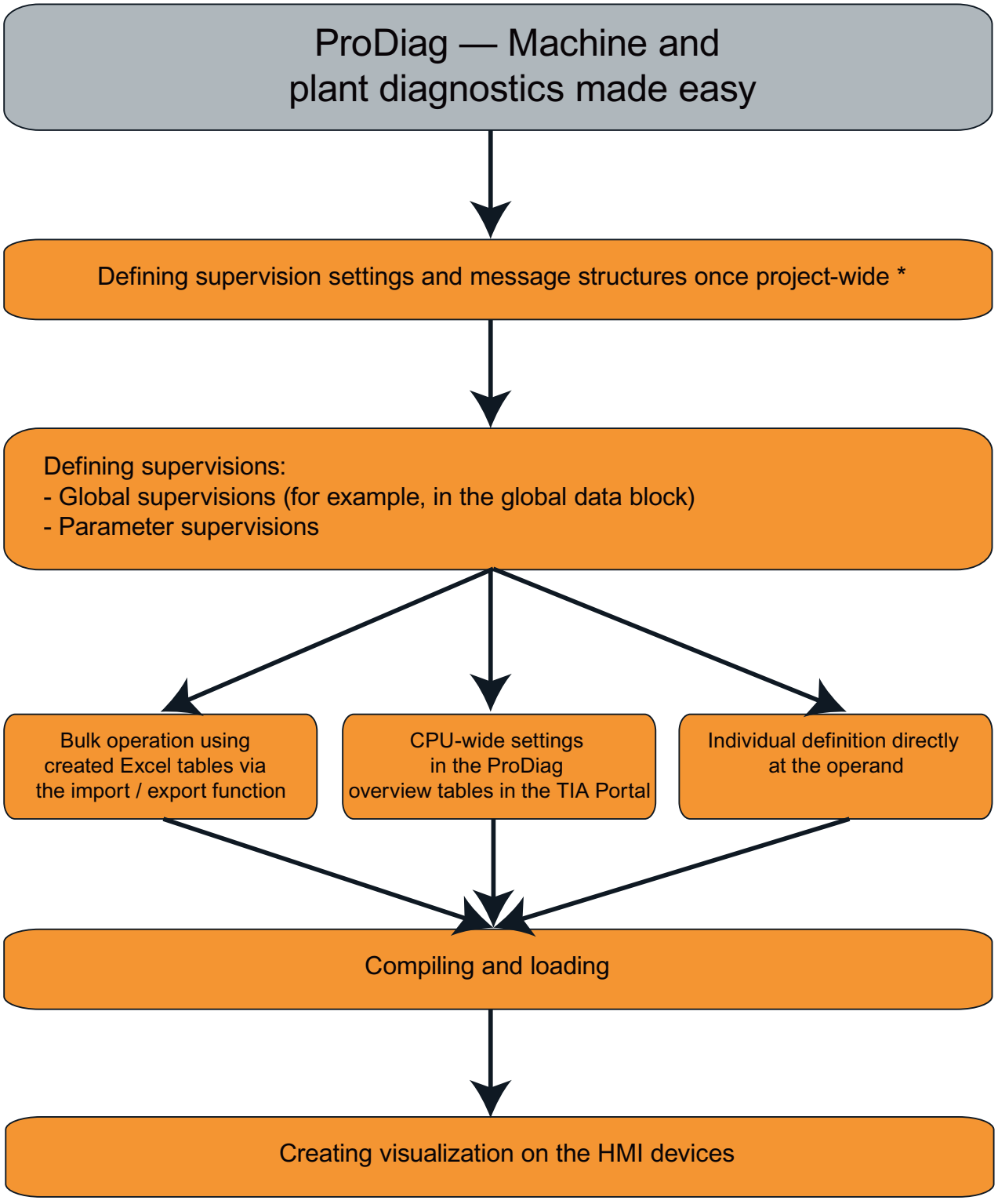

* The supervision settings are not available until you have created a GRAPH FB or a supervision.

For an introductory example of how to create supervisions with the help of ProDiag, go to: Example for creating a supervision based on an operand supervision (Page 7803)

Benefits

The supervision of your machine or plant with ProDiag has the following benefits:

- Your program code is not changed when you create supervisions and the ProDiag function blocks are automatically called in the ProDiag organization block, as long as you do not want to call them yourself in your program code.
- With the help of the ProDiag function blocks, you can create units and group all supervisions of a unit, such as a motor, according to technological aspects.
- You can reduce downtimes and production outages after a fault occurs or in the case of foreseeable events with the help of early warnings or information.
- Fault removal and possible maintenance work are made much easier because specific information about the cause of a fault is made available to the machine operator on site.
- Regardless of which S7-1500 CPU you are using, the configuration of supervisions is simple and easy to understand.
- You can also supervise both F blocks as well as know-how protected function blocks without problems with ProDiag since they are not changed and therefore do not need to be recompiled.
- Alarm texts are automatically derived from texts in the project, for example:
  - Category
  - CPU name
  - FB name
  - Instance name, etc.
- Changes in the CPU configuration are applied automatically by the HMI device. The alarm texts are managed directly in the CPU.
- Relatively low influence on the cycle time.
- The change management of the supervisions is completely managed in the TIA Portal, which means users do not have to focus on ensuring consistency. This means the error probability of incorrect alarm texts is largely eliminated.

24.3.2 Example for creating a supervision based on an operand supervision (S7-1500)

Description

The example below provides you with a brief overview of how to create a supervision based on an operand supervision: Detailed Information on the individual steps can be found in the following sections.

1. Create a ProDiag supervision group. To do this insert a new function block of the type PRODIAG with the name "Station_1":

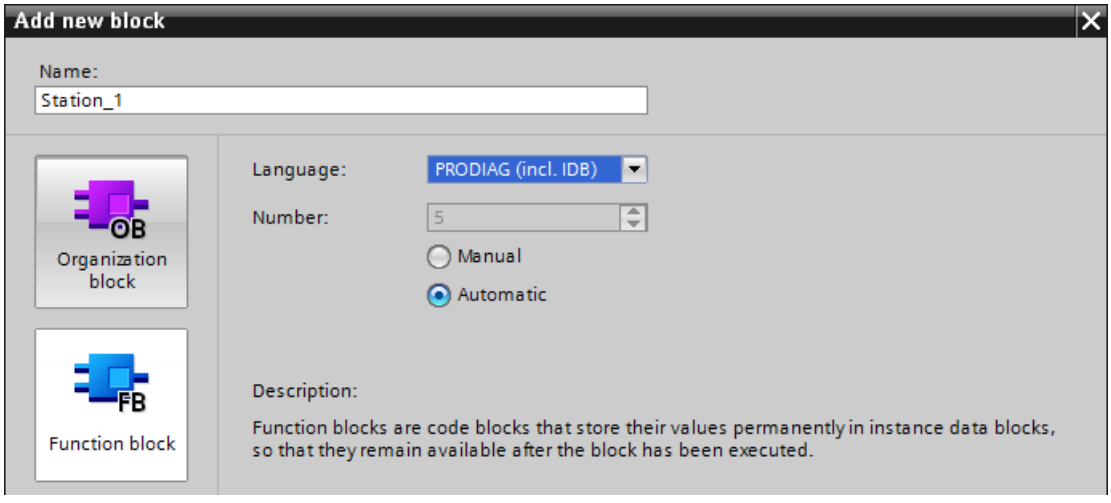

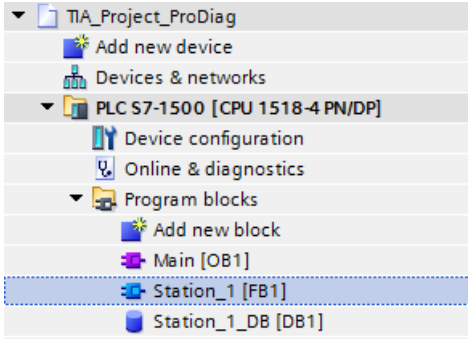

2. Call up the ProDiag supervision settings in the project tree in the "Common data" folder.

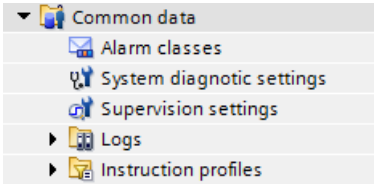

3. Specify the ProDiag supervision settings

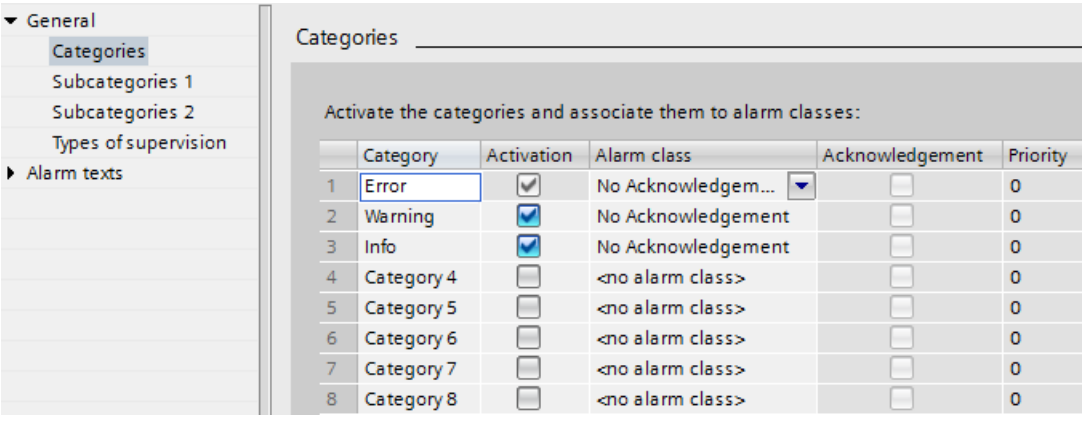

4. In the default tag table declare a Boolean tag and add a supervision.

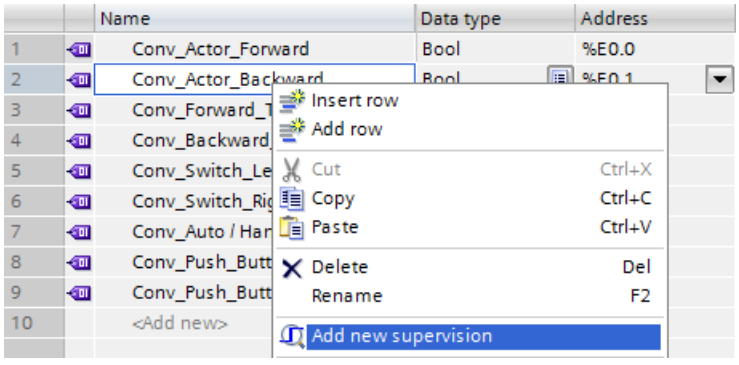

5. In the Inspector window define the properties of the operand supervision in "Properties > Supervisions".

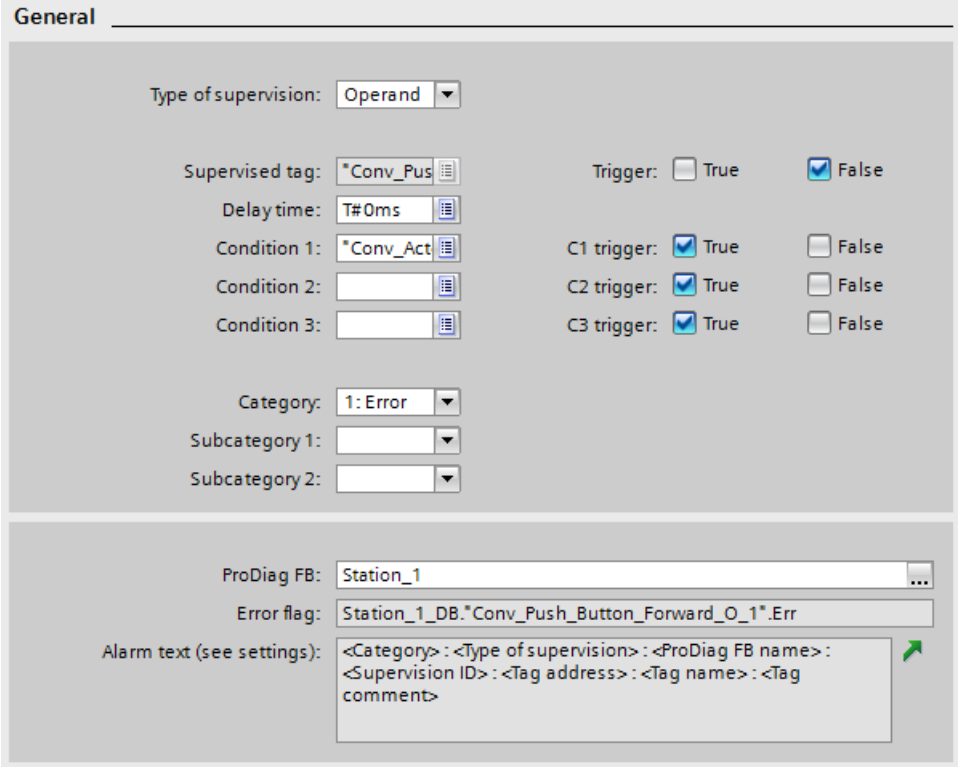

You can create either a global tag supervision or a local parameter supervision at the function block. To do so, you can either use an operand that is already included in the program code or you create a new operand and subsequently call it in the program code. When you create a parameter supervision, you must call the function block in the program code.

6. Compile and download the project.
   The ProDiag function block "Station_1" is automatically called in the ProDiag organization block.

See also

Creating ProDiag function blocks (Page 7838)

Defining the ProDiag supervision settings (Page 7869)

Creating ProDiag supervisions (Page 7889)

24.3.3 Types of supervision (S7-1500)

24.3.3.1 Overview of the types of supervision (S7-1500)

Description

With the ProDiag functionality you can define different alarm texts for the following types of supervision:

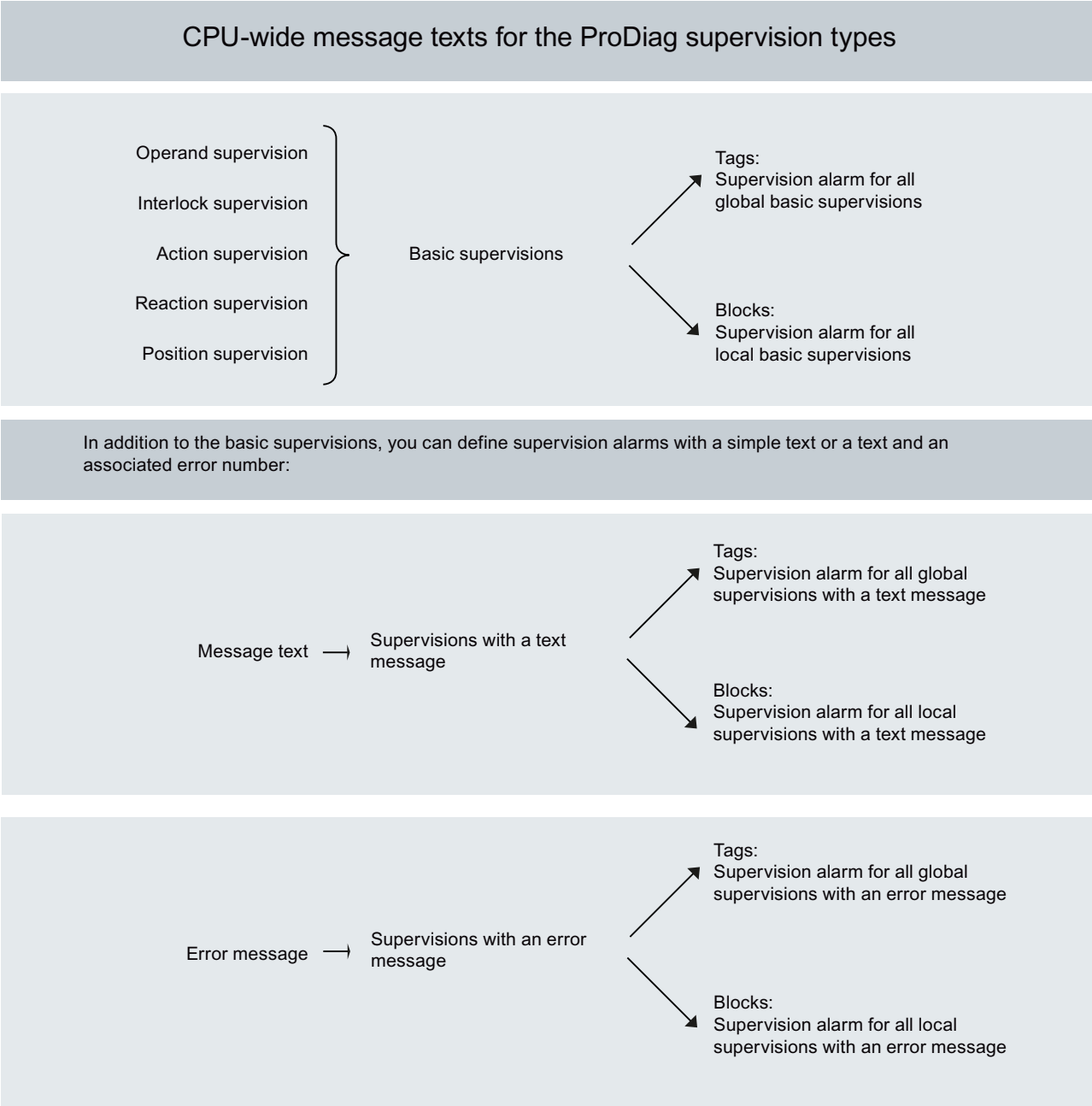

Additional information on the definition of the alarm texts is available here:Specifying central ProDiag alarm texts (Page 7879)

Note

You can find more detailed information on the types of supervision here:

Operand supervision (Page 7807)

Interlock supervision (Page 7809)

Action supervision (Page 7811)

Reaction supervision (Page 7812)

Position supervision (Page 7814)

Text message (Page 7816)

Error message (Page 7817)

Note

You will find examples of the types of supervision here:

Example of an operand supervision (Page 7820)

Example of an interlock supervision (Page 7822)

Example of an action supervision (Page 7825)

Example of a reaction supervision (Page 7828)

Example of a position supervision (Page 7831)

Example for a text message (Page 7834)

Example of an error message (Page 7835)

24.3.3.2 Operand supervision (S7-1500)

Description

Operand supervision monitors the signal state (TRUE or FALSE) of a Boolean operand and is mainly used for supervision of so-called "static signals", such as pressure, temperature, signal from the plant, etc.

You can also use operand supervision to monitor two interdependent Boolean operands.

The triggering of an operand supervision sets the error flag in the assigned ProDiag function block. As a result, you can visualize the fault/error on an HMI device with the "ProDiag

overview" object and query it in your program, for example, to shut down the machine in case of specific faults/errors.

Note

Additional information on the HMI object "ProDiag overview"

For additional information, refer to the information system of the TIA Portal. "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag > Visualizing supervisions > Configuring the ProDiag overview"

Configuration

The following table shows the configuration of an operand supervision:

Properties	Entry	Operand status	Description
Supervised tag / parameter *	Mandatory	TRUE or FALSE	Operand to be supervised
Delay time	Optional	Time value, e.g. T#2s or a tag of the data type TIME.	Time that is allowed to expire while the supervised signal state of the operand is queued. If the time, e.g. a bounce time, is exceeded, a fault is present and a supervision alarm is displayed.
Condition 1	Optional	TRUE or FALSE	If you are using multiple conditions, they are automatically linked with an AND logic operation.
Condition 2	Optional		
Condition 3	Optional		
Category	Mandatory	-	You can define up to 8 categories. The first three categories are preassigned by default. However, they can be changed at any time. 1. Error 2. Warning 3. Info You can specify which categories are available in the ProDiag supervision settings. You can find additional information here: Overview of the categories (Page 7870)
Subcategory 1	Optional	-	You can define up to 16 entries per subcategory. You can find additional information here: Overview of the subcategories (Page 7873)
Subcategory 2	Optional	-	
ProDiag FB (only for global supervisions)	Mandatory	-	Each supervision must be assigned to a ProDiag function block. You can find additional information here: Overview of the ProDiag function blocks (Page 7838)

Properties	Entry	Operand status	Description
Error flag (only for global supervisions)	Is entered by the system.	-	Indicates whether a fault has occurred: • TRUE = Supervision alarm was triggered • FALSE = No supervision alarm, which means the fault has not occurred The error flag is accessible from HMI and visible in HMI.
Alarm text (see settings)	Is entered by the system.	-	You define the content of the alarm text in the ProDiag supervision settings. You can find additional information here: Structure of the alarm text editor (Page 7879)
* Depending on whether you create a local or a global supervision either a supervised tag (global) or a supervised parameter (local) is displayed in the "Properties" tab.			

See also

Example of an operand supervision (Page 7820)

24.3.3.3 Interlock supervision (S7-1500)

Description

The interlock supervision monitors if the action is executed after triggering the action (trigger) and after expiration of the delay time (optional), and if the interlock conditions are met. If the interlock conditions are not met, a fault is present and the action is not executed.

The interlock supervision is often used for supervision of outputs that drive a motor.

The triggering of an interlock supervision sets the error flag in the assigned ProDiag function block. As a result, you can visualize the fault/error on an HMI device with the "ProDiag overview" object and query it in your program, for example, to shut down the machine in case of specific faults/errors.

Note

Additional information on the HMI object "ProDiag overview"

For additional information, refer to the information system of the TIA Portal. "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag > Visualizing supervisions > Configuring the ProDiag overview"

Configuration

The following table shows the configuration of an interlock supervision:

Properties	Entry	Operand status	Description
Supervised tag / parameter *	Mandatory	TRUE or FALSE	Operand to be supervised
Delay time	Optional	Time value, e.g. T#2s or a tag of the data type TIME.	Time that is allowed to expire while the supervised signal state of the operand is queued. If the time, e.g. a bounce time, is exceeded, a fault is present and a supervision alarm is displayed.
Actuator (Condition 1)	Mandatory	TRUE or FALSE	Condition 1 must be present. If you are using multiple conditions, they are automatically linked with an AND logic operation.
Condition 2	Optional		
Condition 3	Optional		
Category	Mandatory	-	You can define up to 8 categories. The first three categories are preassigned by default. However, they can be changed at any time. 1. Error 2. Warning 3. Info You can specify which categories are available in the ProDiag supervision settings. You can find additional information here: Overview of the categories (Page 7870)
Subcategory 1	Optional	-	You can define up to 16 entries per subcategory. You can find additional information here: Overview of the subcategories (Page 7873)
Subcategory 2	Optional	-	
ProDiag FB (only for global supervisions)	Mandatory	-	Each supervision must be assigned to a ProDiag function block. You can find additional information here: Overview of the ProDiag function blocks (Page 7838)
Error flag (only for global supervisions)	Is entered by the system.	-	Indicates whether a fault has occurred: • TRUE = Supervision alarm was triggered • FALSE = No supervision alarm, which means the fault has not occurred The error flag is accessible from HMI and visible in HMI.
Alarm text (see settings)	Is entered by the system.	-	You define the content of the alarm text in the ProDiag supervision settings. You can find additional information here: Structure of the alarm text editor (Page 7879)
* Depending on whether you create a local or a global supervision either a supervised tag (global) or a supervised parameter (local) is displayed in the "Properties" tab.			

See also

Example of an interlock supervision (Page 7822)

Possible uses of supervisions (Page 7892)

24.3.3.4 Action supervision (S7-1500)

Description

Action supervision monitors whether the production part on the conveyor belt has left the start position within the start-up time after the machine operation. For example, this is not the case if the production part is blocked in its movement. The use of action supervision is especially practical for slow processes, such as the filling of a silo.

The triggering of an action supervision sets the error flag in the assigned ProDiag function block. As a result, you can visualize the fault/error on an HMI device with the "ProDiag overview" object and query it in your program, for example, to shut down the machine in case of specific faults/errors.

Note

Additional information on the HMI object "ProDiag overview"

For additional information, refer to the information system of the TIA Portal. "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag > Visualizing supervisions > Configuring the ProDiag overview"

Configuration

The following table shows the configuration of an action supervision:

Settings	Entry	Operand status	Description
Supervised tag / parameter *	Mandatory	TRUE or FALSE	Operand to be supervised
Start-up time	Mandatory	Time value, min. T#150s or a tag of the data type TIME.	Time that may elapse until the start position must be left. The start-up time must be greater than the cycle time.
Action (Condition 1)	Mandatory	TRUE or FALSE	Condition 1 must be present. If you are using multiple conditions, they are automatically linked with an AND logic operation.
Condition 2	Optional		
Condition 3	Optional		

Settings	Entry	Operand status	Description
Category	Mandatory	-	You can define up to 8 categories. The first three categories are preassigned by default. However, they can be changed at any time. 1. Error 2. Warning 3. Info You can specify which categories are available in the ProDiag supervision settings. You can find additional information here: Overview of the categories (Page 7870)
Subcategory 1	Optional	-	You can define up to 16 entries per subcategory. You can find additional information here: Overview of the subcategories (Page 7873)
Subcategory 2	Optional	-	
ProDiag FB (only for global supervisions)	Mandatory	-	Each supervision must be assigned to a ProDiag function block. You can find additional information here: Overview of the ProDiag function blocks (Page 7838)
Error flag (only for global supervisions)	Is entered by the system.	-	Indicates whether a fault has occurred: • TRUE = Supervision alarm was triggered • FALSE = No supervision alarm, which means the fault has not occurred The error flag is accessible from HMI and visible in HMI.
Alarm text (see settings)	Is entered by the system.	-	You define the content of the alarm text in the ProDiag supervision settings. You can find additional information here: Structure of the alarm text editor (Page 7879)
* Depending on whether you create a local or a global supervision either a supervised tag (global) or a supervised parameter (local) is displayed in the "Properties" tab.			

See also

Example of an action supervision (Page 7825)

Possible uses of supervisions (Page 7892)

24.3.3.5 Reaction supervision (S7-1500)

Description

The reaction supervision monitors whether the production part on the conveyor belt has reached the desired end position within the reaction time after a machine operation. That cannot happen, for example, if the production part has fallen down or is jammed. The use of reaction supervision is especially practical for fast processes.

The triggering of a reaction supervision sets the error flag in the assigned ProDiag function block. As a result, you can visualize the fault/error on an HMI device with the "ProDiag overview" object and query it in your program, for example, to shut down the machine in case of specific faults/errors.

Note

Additional information on the HMI object "ProDiag overview"

For additional information, refer to the information system of the TIA Portal. "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag > Visualizing supervisions > Configuring the ProDiag overview"

Configuration

The following table shows the configuration of a reaction supervision:

Settings	Entry	Operand status	Description
Supervised tag / parameter *	Mandatory	TRUE or FALSE	Operand to be supervised
Reaction time	Mandatory	Time value, e.g. T#5000 ms or a tag of the data type TIME.	Maximum time that may elapse until the end position must be reached. If the time is exceeded, a fault is present and a supervision alarm is displayed.
Action (Condition 1)	Mandatory	TRUE or FALSE	Condition 1 must be present. If you are using multiple conditions, they are automatically linked with an AND logic operation.
Condition 2	Optional		
Condition 3	Optional		
Category	Mandatory	-	You can define up to 8 categories. The first three categories are preassigned by default. However, they can be changed at any time. 1. Error 2. Warning 3. Info You can specify which categories are available in the ProDiag supervision settings. You can find additional information here: Overview of the categories (Page 7870)
Subcategory 1	Optional	-	You can define up to 16 entries per subcategory. You can find additional information here: Overview of the subcategories (Page 7873)
Subcategory 2	Optional	-	
ProDiag FB (only for global supervisions)	Mandatory	-	Each supervision must be assigned to a ProDiag function block. You can find additional information here: Overview of the ProDiag function blocks (Page 7838)

Settings	Entry	Operand status	Description
Error flag (only for global supervisions)	Is entered by the system.	-	Indicates whether a fault has occurred: • TRUE = Supervision alarm was triggered • FALSE = No supervision alarm, which means the fault has not occurred The error flag is accessible from HMI and visible in HMI.
Alarm text (see settings)	Is entered by the system.	-	You define the content of the alarm text in the ProDiag supervision settings. You can find additional information here: Structure of the alarm text editor (Page 7879)
* Depending on whether you create a local or a global supervision either a supervised tag (global) or a supervised parameter (local) is displayed in the "Properties" tab.			

See also

Example of a reaction supervision (Page 7828)

Possible uses of supervisions (Page 7892)

24.3.3.6 Position supervision (S7-1500)

Description

Position supervision monitors whether the currently reached end position (start or target) of the production part is maintained in a stable manner after expiration of the delay time. The end position should not be left again without a trigger. For example, this is the case if the production part is knocked from a conveyor belt by a forklift truck.

The triggering of a position supervision sets the error flag in the assigned ProDiag function block. As a result, you can visualize the fault/error on an HMI device with the "ProDiag overview" object and query it in your program, for example, to shut down the machine in case of specific faults/errors.

Note

Additional information on the HMI object "ProDiag overview"

For additional information, refer to the information system of the TIA Portal. "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag > Visualizing supervisions > Configuring the ProDiag overview"

Configuration

The following table shows the configuration of a position supervision:

Settings	Entry	Operand status	Description
Supervised tag / parameter *	Mandatory	TRUE or FALSE	Operand to be supervised
Delay time	Optional	Time value, e.g. T#2s or a tag of the data type TIME.	Time that is allowed to expire while the supervised signal state of the operand is queued. If the time, e.g. a bounce time, is exceeded, a fault is present and a supervision alarm is displayed.
Action (Condition 1)	Mandatory	TRUE or FALSE	Condition 1 must be present. If you are using multiple conditions, they are automatically linked with an AND logic operation.
Condition 2	Optional		
Condition 3	Optional		
Category	Mandatory	-	You can define up to 8 categories. The first three categories are preassigned by default. However, they can be adjusted at any time. 1. Error 2. Warning 3. Info You can specify which categories are available in the ProDiag supervision settings. You can find additional information here: Overview of the categories (Page 7870)
Subcategory 1	Optional	-	You can define up to 16 entries per subcategory. You can find additional information here: Overview of the subcategories (Page 7873)
Subcategory 2	Optional	-	
ProDiag FB (only for global supervisions)	Mandatory	-	Each supervision must be assigned to a ProDiag function block. You can find additional information here: Overview of the ProDiag function blocks (Page 7838)
Error flag (only for global supervisions)	Is entered by the system.	-	Indicates whether a fault has occurred: • TRUE = Supervision alarm was triggered • FALSE = No supervision alarm, which means the fault has not occurred The error flag is accessible from HMI and visible in HMI.
Alarm text (see settings)	Is entered by the system.	-	You define the content of the alarm text in the ProDiag supervision settings. You can find additional information here: Structure of the alarm text editor (Page 7879)
* Depending on whether you create a local or a global supervision either a supervised tag (global) or a supervised parameter (local) is displayed in the "Properties" tab.			

See also

Example of a position supervision (Page 7831)

Possible uses of supervisions (Page 7892)

24.3.3.7 Text message (S7-1500)

Description

Through this additional type of supervision you have the possibility to output only one text that you define in the specific text field. Triggering a text message does not set the group error bit "All" of the "State" status tag in the ProDiag instance data block.

Configuration

The following table shows the configuration of a text message:

Settings	Entry	Operand status	Description
Supervised tag / parameter *	Mandatory	TRUE or FALSE	Operand to be supervised
Delay time	Optional	Time value, e.g. T#2s or a tag of the data type TIME.	Time that is allowed to expire while the supervised signal state of the operand is queued. If the time, e.g. a bounce time, is exceeded, a fault is present and a text message is displayed.
Condition 1	Optional	TRUE or FALSE	If you are using multiple conditions, they are automatically linked with an AND logic operation.
Condition 2	Optional		
Condition 3	Optional		
Category	Mandatory	-	You can define up to 8 categories. The first three categories are preassigned by default. However, they can be changed at any time. 1. Error 2. Warning 3. Info You can specify which categories are available in the ProDiag supervision settings. You can find additional information here: Overview of the categories (Page 7870)
Subcategory 1	Optional	-	You can define up to 16 entries per subcategory. You can find additional information here: Overview of the subcategories (Page 7873)
Subcategory 2	Optional	-	

Settings	Entry	Operand status	Description
ProDiag FB (only for global supervisions)	Mandatory	-	Each supervision must be assigned to a ProDiag function block. You can find additional information here: Overview of the ProDiag function blocks (Page 7838)
Error flag (only for global supervisions)	Is entered by the system.	-	Indicates whether a fault has occurred: • TRUE = Text message was triggered • FALSE = No text message, which means the fault has not occurred The error flag is accessible from HMI and visible in HMI.
Alarm text (see settings)	Is entered by the system.	-	You define the content of the alarm text in the ProDiag supervision settings. You can find additional information here: Structure of the alarm text editor (Page 7879)
* Depending on whether you create a local or a global supervision either a supervised tag (global) or a supervised parameter (local) is displayed in the "Properties" tab.			

See also

Example for a text message (Page 7834)

24.3.3.8 Error message (S7-1500)

Description

The "Error message" type of supervision outputs a simple text message plus an error number. Through this additional type of supervision you have the possibility to output only one text that you define in the specific text field and the error number.

The triggering of an error message sets the error flag in the assigned ProDiag function block. As a result, you can visualize the fault/error on an HMI device with the "ProDiag overview" object and query it in your program, for example, to shut down the machine in case of specific faults/errors.

Note

Additional information on the HMI object "ProDiag overview"

For additional information, refer to the information system of the TIA Portal. "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag > Visualizing supervisions > Configuring the ProDiag overview"

Configuration

The following table shows the configuration of an error message:

Settings	Entry	Operand status	Description
Supervised tag / parameter *	Mandatory	TRUE or FALSE	Operand to be supervised
Delay time	Optional	Time value, e.g. T#2s or a tag of the data type TIME.	Time that is allowed to expire while the supervised signal state of the operand is queued. If the time, e.g. a bounce time, is exceeded, a fault is present and an error message is displayed.
Condition 1	Optional	TRUE or FALSE	If you are using multiple conditions, they are automatically linked with an AND logic operation.
Condition 2	Optional		
Condition 3	Optional		
Category	Mandatory	-	You can define up to 8 categories. The first three categories are preassigned by default. However, they can be changed at any time. 1. Error 2. Warning 3. Info You can specify which categories are available in the ProDiag supervision settings. You can find additional information here: Overview of the categories (Page 7870)
Subcategory 1	Optional	-	You can define up to 16 entries per subcategory. You can find additional information here: Overview of the subcategories (Page 7873)
Subcategory 2	Optional	-	
ProDiag FB (only for global supervisions)	Mandatory	-	Each supervision must be explicitly assigned to a ProDiag function block. You can find additional information here: Overview of the ProDiag function blocks (Page 7838)
Error flag (only for global supervisions)	Is entered by the system.	-	Indicates whether a fault has occurred: • TRUE = Supervision alarm was triggered • FALSE = No supervision alarm, which means the fault has not occurred The error flag is accessible from HMI and visible in HMI.
Alarm text (see settings)	Is entered by the system.	-	You define the content of the alarm text in the ProDiag supervision settings. You can find additional information here: Structure of the alarm text editor (Page 7879)
* Depending on whether you create a local or a global supervision either a supervised tag (global) or a supervised parameter (local) is displayed in the "Properties" tab.			

24.3.4 Examples for individual types of supervision (S7-1500)

24.3.4.1 Introduction (S7-1500)

Description

Based on the following example, we will explain all types of supervision to you by gradually moving the production part from the left to the right on the conveyor belt. Different faults can occur during the process; for example, the production part does not leave its start position or the production part does not reach its end position in time. To supervise the entire process and identify all problems, you can implement different supervisions with the ProDiag functionality.

The figure below serves as template for all subsequent examples:

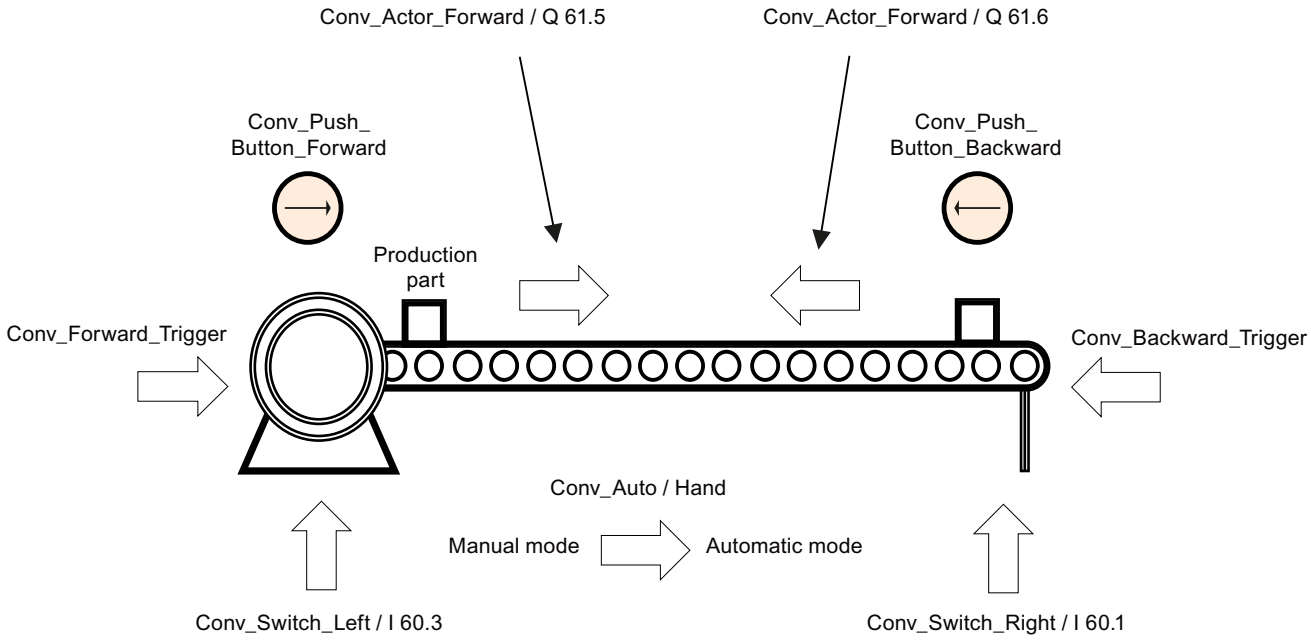

The operands that are required in the respective examples are highlighted in gray.

The following sections include examples for all possible types of supervision:

- Example of an operand supervision (Page 7820)
- Example of an interlock supervision (Page 7822)
- Example of an action supervision (Page 7825)
- Example of a reaction supervision (Page 7828)
- Example of a position supervision (Page 7831)

24.3.4.2 Example of an operand supervision (S7-1500)

Requirement

You have created a supervision for the "Conv_Switch_Left" operand and have selected "Operand" as the type of supervision.

Description

The figure below shows a conveyor belt at standstill on which the production part is located. The production part is to the left in the start position and is currently not triggered. The arrows visualize the movement of the production part from left to right and from right to left as soon as production starts:

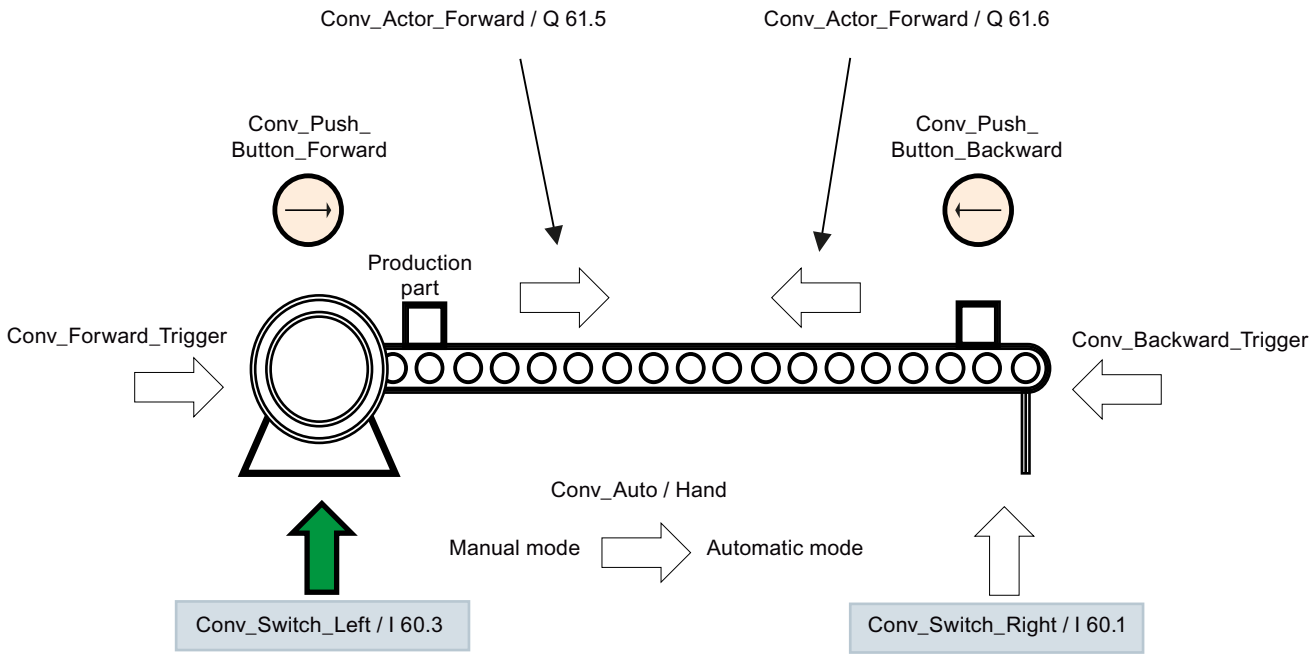

The operands required for this supervision are highlighted in gray.

Operand	Explanation
Conv_Actor_Forward Conv_Actor_Backward	These are the control signals (output of the motor) from the user program for movements to the left and right.
Conv_Forward_Trigger Conv_Backward_Trigger	These are the control signals from the user program (e.g. from a GRAPH function block) for triggering the movements to the left and right.
Conv_Switch_Left Conv_Switch_Right	This is the start and target position of the production part.
Conv_Auto / Hand	This is the switch for selecting the operating mode.
Conv_Push_Button_Forward Conv_Push_Button_Backward	These are the respective pushbuttons for the forward and backward motion.

Procedure

Enter the following values into the properties of the operand supervision:

Properties	Values	Operand status / Explanation
Type of supervision	Operand	-
Supervised tag	Conv_Switch_Left	TRUE As soon as the operand switches to the signal state TRUE and all conditions are met, the supervision alarm is triggered.
Delay time	T#0ms	This fault must be detected and reported immediately. Therefore, 0ms is specified as delay time.
Condition 1	Conv_Switch_Right	TRUE You can use Condition 1 to supervise the two interdependent Boolean operands.
Condition 2	-	The conditions are not required.
Condition 3	-	
Category	1: Error	Triggering a supervision alarm is classified as an error.
Subcategory 1	-	The subcategories are not required.
Subcategory 2	-	
ProDiag FB	Default_SupervisionFB	This is the associated ProDiag function block. As long as you are not using your own function block, the Default_SupervisionFB is used by default.
Error flag	Default_SupervisionDB.Conv_Switch_Left_O_1.Err	This error flag is set as soon as a fault occurs.
Specific text field	Both operands have the signal state TRUE	Text with explanation that can be output with the supervision alarm.

Result

The supervised operand "Conv_Switch_Left" and the condition 1 "Conv_Switch_Right" are both monitored for the signal state TRUE. This means there is no fault as long as only one of the two changes from the signal state FALSE to TRUE. As soon as both switch to the signal state TRUE, however, a fault is present and a supervision alarm is displayed.

Depending on the settings you have made for basic supervision in the ProDiag supervision settings, the supervision alarm may look as follows:

Supervision alarm for an operand error

```
Error: Operand: CPU_50: Default_SupervisionFB: I 60.3: Conv_Switch_Left: Both operands have
the signal state TRUE
```

See also

Operand supervision (Page 7807)

24.3.4.3 Example of an interlock supervision (S7-1500)

Requirement

You have created two supervisions for the "Conv_Actor_Forward" operand and have selected "Interlock" as the type of supervision.

- Supervision_ID_1 (Default_SupervisionFB)
- Supervision_ID_2 (Default_SupervisionFB)

You can implement OR operations in this way. For example, you can define "Automatic mode" as a condition for <Supervision_ID_1> and "Manual mode" as a condition for <Supervision_ID_2>.

Note

Possible applications

Ensure that the global tag "Conv_Actor_Forward" is not declared as an input.

You can find additional information on possible applications here: Possible uses of supervisions (Page 7892)

Description

The figure below shows a conveyor belt at standstill on which the production part is located. The production part is to the left in the start position and is currently not triggered. The arrows visualize the movement of the production part from left to right and from right to left as soon as production starts:

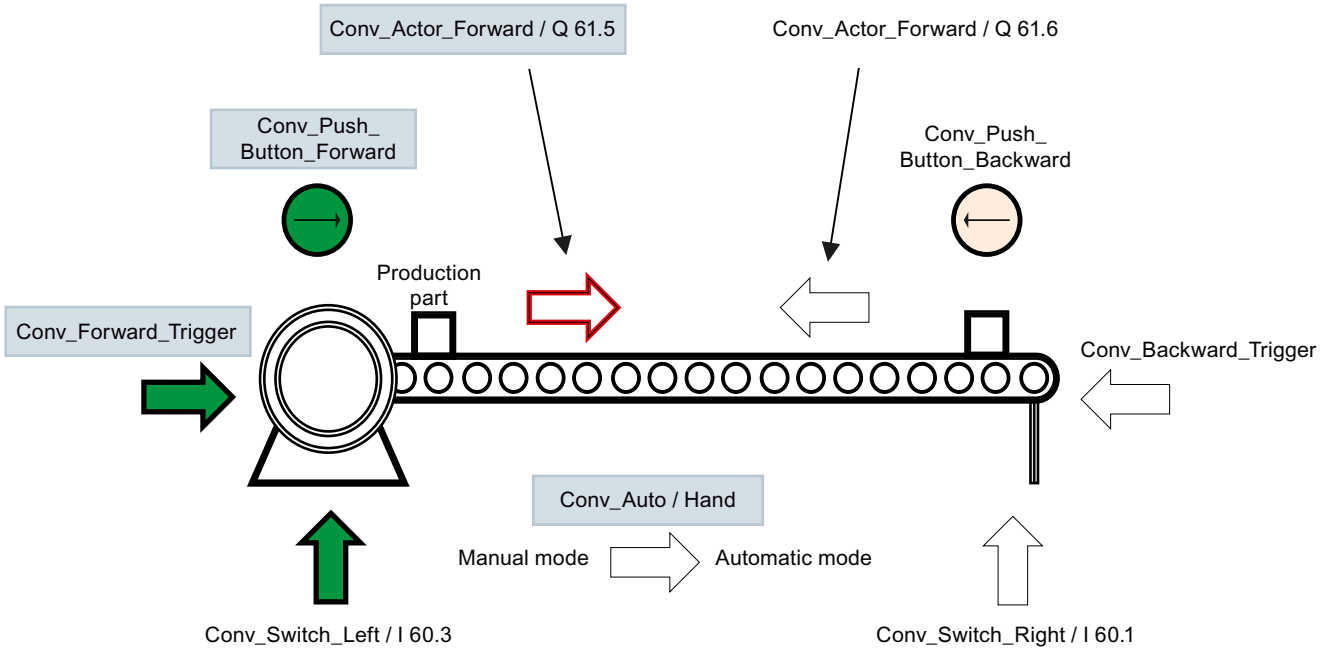

The operands required for this supervision are highlighted in gray.

Operand	Explanation
Conv_Actor_Forward Conv_Actor_Backward	These are the control signals (output of the motor) from the user program for movements to the left and right.
Conv_Forward_Trigger Conv_Backward_Trigger	These are the control signals from the user program (e.g. from a GRAPH function block) for triggering the movements to the left and right.
Conv_Switch_Left Conv_Switch_Right	This is the start and target position of the production part.
Conv_Auto / Hand	This is the switch for selecting the operating mode.
Conv_Push_Button_Forward Conv_Push_Button_Backward	These are the respective pushbuttons for the forward and backward motion.

Procedure for automatic mode

Enter the following values into the properties of the interlock supervision:

Properties	Values	Operand status / Explanation
Type of supervision	Interlock	-
Supervised tag	Conv_Actor_Forward	FALSE As soon as the operand switches to the signal state FALSE or even retains the signal state TRUE and all conditions are met, the supervision alarm is triggered.
Delay time	T#0ms	This fault must be detected and reported immediately. Therefore, 0ms is specified as delay time.
Actuator (Condition 1)	Conv_Forward_Trigger	TRUE You can use Condition 1 to supervise the two interdependent Boolean operands.
Condition 2	Conv_Auto / Manu	TRUE = Automatic mode
Condition 3	-	The condition is not required.
Category	1: Error	Triggering a supervision alarm is classified as an error.
Subcategory 1	-	The subcategories are not required.
Subcategory 2	-	
ProDiag FB	Default_SupervisionFB	This is the associated ProDiag function block. As long as you are not using your own function block, the Default_SupervisionFB is used by default.

Properties	Values	Operand status / Explanation
Error flag	Default_SupervisionDB.Conv_Actor_Forward_I_1.Err	This error flag is set as soon as a fault occurs.
Specific text field	Automatic mode	Text with explanation that can be output with the supervision alarm.

Procedure for manual mode

Enter the following values into the properties of the operand supervision:

Properties	Values	Operand status / Explanation
Type of supervision	Interlock	-
Supervised operand	Conv_Actor_Forward	FALSE As soon as the operand switches to the signal state FALSE and all conditions are met, the supervision alarm is triggered.
Delay time	T#0ms	This fault must be detected and reported immediately. Therefore, 0ms is specified as delay time.
Actuator (Condition 1)	Conv_Push_Button_Forward	TRUE You can use Condition 1 to supervise the two interdependent Boolean operands.
Condition 2	Conv_Auto / Manu	FALSE = Manual mode
Condition 3	-	The condition is not required.
Category	1: Error	Triggering a supervision alarm is classified as an error.
Subcategory 1	-	The subcategories are not required.
Subcategory 2	-	
ProDiag FB	Default_SupervisionFB	This is the associated ProDiag function block. As long as you are not using your own function block, the Default_SupervisionFB is used by default.
Error flag	Default_SupervisionDB.Conv_Actor_Forward_I_1.Err	This error bit is set as soon as a fault occurs.
Specific text field	Manual mode	Text with explanation that can be output with the supervision alarm.

Result

The movement of the production part from left to right was triggered either by the program with the operand "Conv_Forward_Trigger" or with the pushbutton and the operand "Conv_Push_Button_Forward". The production part has not left the start position on the left, however, and the operand "Conv_Actor_Forward" therefore does not change to the signal state TRUE. This may also happen, for example, in the middle of the motion. This means all requirements for triggering the supervision alarm have been met.

Depending on the settings you have made for basic supervision in the ProDiag supervision settings, the supervision alarms can look like this:

Supervision alarm for an interlock error in automatic mode

```
Error: Interlock: CPU_50: Default_SupervisionFB: I 61.5:
Conv_Actor_Forward: Automatic mode
```

Supervision alarm for an interlock error in manual mode

```
Error: Interlock: CPU_50: Default_SupervisionFB: I 61.5: Conv_Actor_Forward: Manual mode
```

See also

Interlock supervision (Page 7809)

24.3.4.4 Example of an action supervision (S7-1500)

Requirement

You have created a supervision for the "Conv_Switch_Left" operand and have selected "Action" as the type of supervision.

Note

Possible applications

Ensure that the global tag "Conv_Switch_Left" is not declared as an output.

You can find additional information on possible applications here: Possible uses of supervisions (Page 7892)

Description

The figure below shows a conveyor belt at standstill on which the production part is located. The production part is on the left in the start position. The arrows visualize the movement of the production part from left to right and from right to left as soon as production starts. The movement of the production part from left to right is currently triggered by the user program, but the production part does not leave the start position in time:

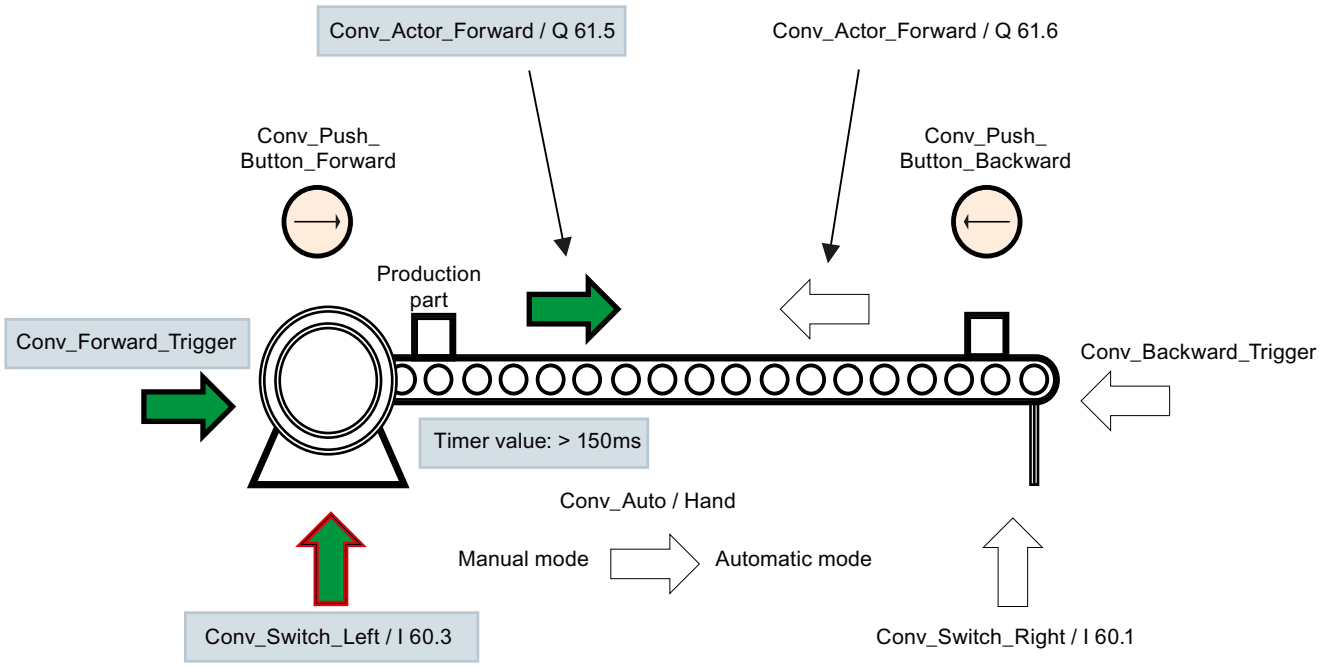

The operands required for this supervision are highlighted in gray.

Operand	Explanation
Conv_Actor_Forward Conv_Actor_Backward	These are the control signals (output of the motor) from the user program for movements to the left and right.
Conv_Forward_Trigger Conv_Backward_Trigger	These are the control signals from the user program (e.g. from a GRAPH function block) for triggering the movements to the left and right.
Conv_Switch_Left Conv_Switch_Right	This is the start and target position of the production part.
Conv_Auto / Hand	This is the switch for selecting the operating mode.
Conv_Push_Button_Forward Conv_Push_Button_Backward	These are the respective pushbuttons for the forward and backward motion.

Procedure

Enter the following values into the properties of the action supervision:

Properties	Values	Operand status / Explanation
Type of supervision	Action	-
Supervised tag	Conv_Switch_Left	TRUE When the production part does not leave its start position on the left of the conveyor belt, the operand "Conv_Switch_Left" remains in the signal state TRUE. When all conditions are met as well, the supervision alarm is triggered.
Start-up time	T#200ms	The production part must leave its start position within 200ms.
Action (Condition 1)	Conv_Actor_Forward	TRUE The control signal for the movement is present. You can use Condition 1 to supervise the two interdependent Boolean operands.
Condition 2	-	The conditions are not required.
Condition 3	-	
Category	1: Error	Triggering a supervision alarm is classified as an error.
Subcategory 1	-	The subcategories are not required.
Subcategory 2	-	
ProDiag FB	Default_SupervisionFB	This is the associated ProDiag function block. As long as you are not using your own function block, the Default_SupervisionFB is used by default.
Error flag	Default_SupervisionDB.Conv_Switch_Left_A_1.Err	This error flag is set as soon as a fault occurs.
Specific text field	-	Text with explanation that can be output with the supervision alarm.

Result

The movement of the production part from left to right was triggered either by the program with the operand "Conv_Forward_Trigger". The operand "Conv_Actor_Forward" changes to the signal state TRUE and the forward motion of the production part is to start from left to right. The production part does not leave the start position on the left in time, however, and the operand "Conv_Switch_Left" therefore remains in the signal state TRUE. This means all requirements for triggering the supervision alarm have been met.

Depending on the settings you have made for basic supervision in the ProDiag supervision settings, the supervision alarms can look like this:

Supervision alarm for an interlock error in automatic mode

```
Error: Action: CPU_50: Default_SupervisionFB: I 60.3: Conv_Switch_Left
```

See also

Action supervision (Page 7811)

24.3.4.5 Example of a reaction supervision (S7-1500)

Requirement

You have created a supervision for the "Conv_Switch_Right" operand and have selected "Reaction" as the type of supervision.

Note

Possible applications

Ensure that the global tag "Conv_Switch_Right" is not declared as an output.

You can find additional information on possible applications here: Possible uses of supervisions (Page 7892)

Description

The figure below shows a conveyor belt at standstill on which the production part is located. The production part is on the way from left (start position) to right (end position). The arrows visualize the movement of the production part from left to right and from right to left as soon as production starts. The movement of the production part from left to right was triggered by the user program, but the production part does not reach its end position in time:

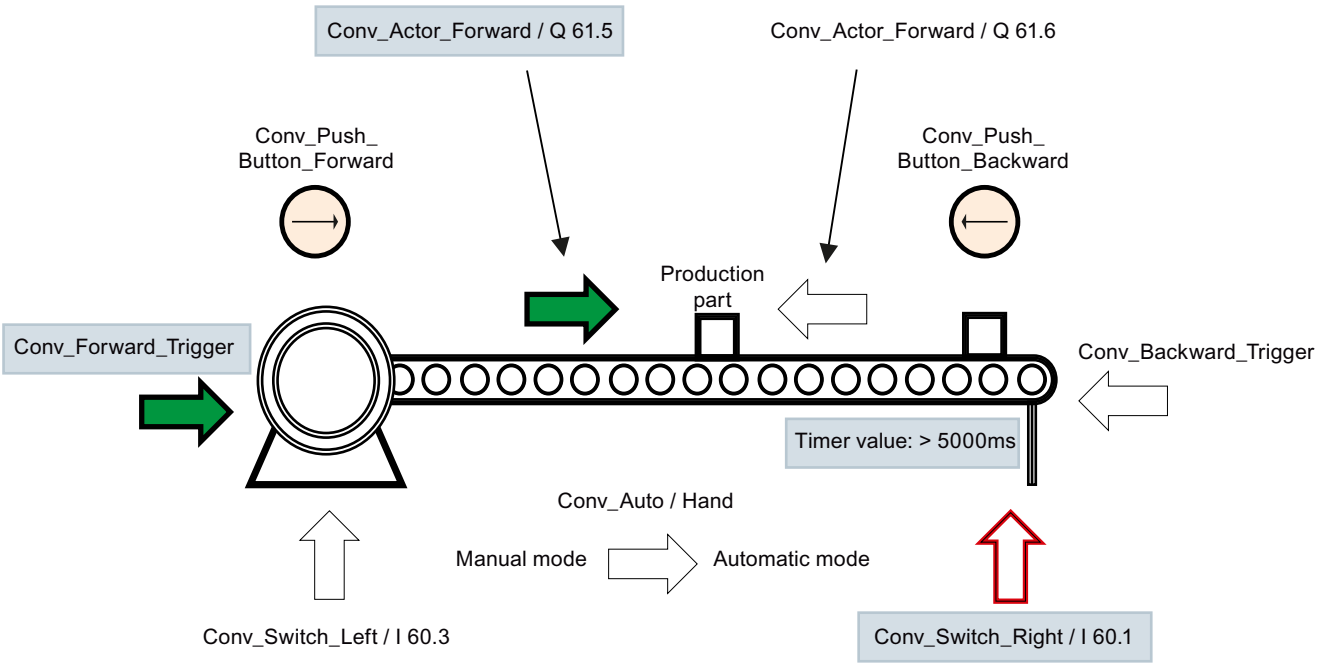

The operands required for this supervision are highlighted in gray.

Operand	Explanation
Conv_Actor_Forward Conv_Actor_Backward	These are the control signals (output of the motor) from the user program for movements to the left and right.
Conv_Forward_Trigger Conv_Backward_Trigger	These are the control signals from the user program (e.g. from a GRAPH function block) for triggering the movements to the left and right.
Conv_Switch_Left Conv_Switch_Right	This is the start and target position of the production part.
Conv_Auto / Hand	This is the switch for selecting the operating mode.
Conv_Push_Button_Forward Conv_Push_Button_Backward	These are the respective pushbuttons for the forward and backward motion.

Procedure

Enter the following values into the properties of the reaction supervision:

Properties	Values	Operand status / Explanation
Type of supervision	Reaction	-
Supervised tag	Conv_Switch_Right	FALSE As soon as the production part reaches its end position, the operand changes to the signal state TRUE. If this does not happen within the reaction time and all conditions are met, the supervision alarm is triggered.
Reaction time	T#5000ms	The movement of the production part from left (start position) to right (end position) must not take longer than 5 seconds.
Action (Condition 1)	Conv_Actor_Forward	TRUE The control signal for the movement is present. You can use Condition 1 to supervise the two interdependent Boolean operands.
Condition 2	-	The conditions are not required.
Condition 3	-	
Category	1: Error	Triggering a supervision alarm is classified as an error.
Subcategory 1	-	The subcategories are not required.
Subcategory 2	-	
ProDiag FB	Default_SupervisionFB	This is the associated ProDiag function block. As long as you are not using your own function block, the Default_SupervisionFB is used by default.
Error flag	Default_SupervisionDB.Conv_Switch_Right_R_1.Err	This error flag is set as soon as a fault occurs.
Specific text field	-	Text with explanation that can be output with the supervision alarm.

Result

The movement of the production part from left to right was triggered either by the program with the operand "Conv_Forward_Trigger". The operand "Conv_Actor_Forward" changes to the signal state TRUE and the forward motion of the production part starts from left to right. The production part does not reach its end position on the right within 5 seconds, however, and the operand "Conv_Switch_Right" remains in the signal state FALSE. This means all requirements for triggering the supervision alarm have been met.

Depending on the settings you have made for basic supervision in the ProDiag supervision settings, the supervision alarms can look like this:

Supervision alarm for a reaction error

```
Error: Reaction: CPU_50: Default_SupervisionFB: I 60.1: Conv_Switch_Right
```

See also

Reaction supervision (Page 7812)

24.3.4.6 Example of a position supervision (S7-1500)

Requirement

You have created a supervision for the "Conv_Switch_Left" operand and have selected "Position" as the type of supervision.

Note

Possible applications

Ensure that the global tag "Conv_Switch_Left" is not declared as an output.

You can find additional information on possible applications here: Possible uses of supervisions (Page 7892)

Description

The figure below shows a conveyor belt at standstill on which the production part is located. The production part is on the way from left (start position) to right (end position). The arrows visualize the movement of the production part from left to right and from right to left as soon as production starts. The movement of the production part from left to right was not triggered by the user program, however, and the production part has left its start position without permission:

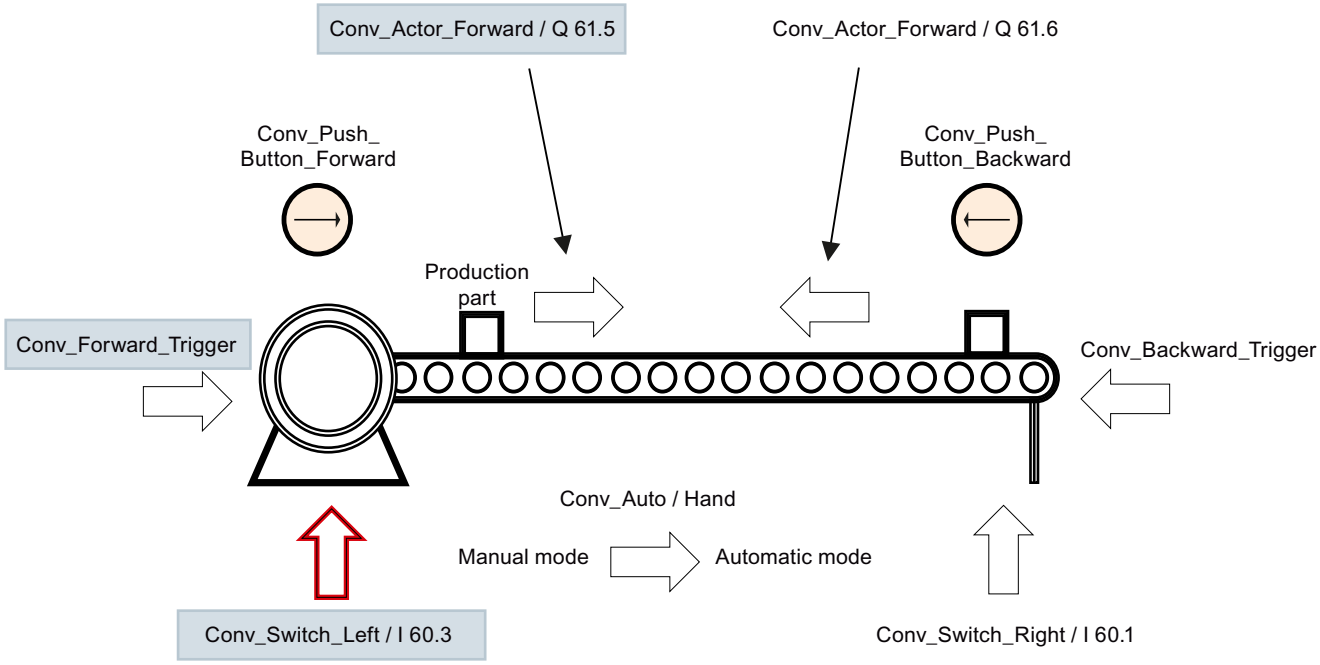

The operands required for this supervision are highlighted in gray.

Operand	Explanation
Conv_Actor_Forward Conv_Actor_Backward	These are the control signals (output of the motor) from the user program for movements to the left and right.
Conv_Forward_Trigger Conv_Backward_Trigger	These are the control signals from the user program (e.g. from a GRAPH function block) for triggering the movements to the left and right.
Conv_Switch_Left Conv_Switch_Right	This is the start and target position of the production part.
Conv_Auto / Hand	This is the switch for selecting the operating mode.
Conv_Push_Button_Forward Conv_Push_Button_Backward	These are the respective pushbuttons for the forward and backward motion.

Procedure

Enter the following values into the properties of the reaction supervision:

Properties	Values	Operand status / Explanation
Type of supervision	Position	-
Supervised tag	Conv_Switch_Left	FALSE As soon as the operand switches to the signal state FALSE without permission and all conditions are met, the supervision alarm is triggered.
Delay time	T#0ms	This fault must be detected and reported immediately. Therefore, 0ms is specified as delay time.
Action (Condition 1)	Conv_Actor_Forward	FALSE No control signal for the movement is present. You can use Condition 1 to supervise the two interdependent Boolean operands.
Condition 2	-	The conditions are not required.
Condition 3	-	
Category	2: Warning	Triggering a supervision alarm is classified as a warning.
Subcategory 1	-	The subcategories are not required.
Subcategory 2	-	
ProDiag FB	Default_SupervisionFB	This is the associated ProDiag function block. As long as you are not using your own function block, the Default_SupervisionFB is used by default.
Error flag	Default_SupervisionDB.Conv_Switch_Left_P_1.Err	This error flag is set as soon as a fault occurs.
Specific text field	Left start position without permission	Text with explanation that can be output with the supervision alarm.

Result

The movement of the production part from left to right was not triggered by the program with the operand "Conv_Forward_Trigger". The operand "Conv_Actor_Forward" remains in the signal state FALSE. The forward motion of the production part from left to right is not supposed to start. However, the production part leaves its start position without permission. This means all requirements for triggering the supervision alarm have been met.

Depending on the settings you have made for basic supervision in the ProDiag supervision settings, the supervision alarms can look like this:

Supervision alarm for an interlock error in automatic mode

```
Warning: Position: CPU_50: Default_SupervisionFB: I 60.3: Conv_Switch_Left: Left start
position without permission
```

See also

Position supervision (Page 7814)

24.3.4.7 Example for a text message (S7-1500)

Requirement

You have created a supervision for the "Trigger_Message" operand and have selected "Text message" as the type of supervision.

Description

You wish to send a text message from the central computer to all the HMI devices, for example, stating that the system will be shut down in an hour in order to carry out maintenance. You can use the "Text message" type of supervision to ensure that you send only this brief information in the supervision alarm, and nothing in addition, such as the CPU name, etc.

The desired text that is to be contained in the supervision alarm is entered into the specific text field.

The tag required for this supervision is as follows:

Operand	Explanation
Trigger_Message	A Boolean variable as actuator for the supervision alarm.

Procedure

Enter the following values into the properties of the text message:

Properties	Values	Operand status / Explanation
Type of supervision	Text message	-
Supervised tag	Trigger_Message	TRUE As soon as the tag switches to the signal state TRUE, the supervision alarm is triggered.
Delay time	T#0ms	The supervision alarm is to be output immediately.

Properties	Values	Operand status / Explanation
Condition 1	-	The conditions are not required.
Condition 2	-	
Condition 3	-	
Category	3: Info	Triggering a supervision alarm is classified as information.
Subcategory 1	-	The subcategories are not required.
Subcategory 2	-	
ProDiag FB	Default_SupervisionFB	This is the associated ProDiag function block. As long as you are not using your own function block, the Default_SupervisionFB is used by default.
Error flag	Default_SupervisionDB.Trigger_Message_O_1.Err	This error flag is set as soon as a fault occurs.
Specific text field	The system will be shut down in 1 hour.	Text with explanation that is output with the supervision alarm.

Result

As soon as the "Trigger_Message" tag assumes the signal state TRUE, the supervision alarm with the text from the specific text field is output.

The supervision alarm has the following form:

Text message

```
The system will be shut down in 1 hour.
```

See also

Text message (Page 7816)

24.3.4.8 Example of an error message (S7-1500)

Requirement

You have created a supervision for the tag "+01R01_Error" and have selected "Error message" as the type of supervision.

Description

You have received a robot from a supplier. In addition to the hardware you have also received a text list with error texts, specially for this robot, and a function block for the robot control. The function block communicates with your program via PROFINET. This means that the robot can signal errors autonomously through its control system. You can have these error texts output in a supervision alarm in the event of a fault.

The function block has, for example, the following form:

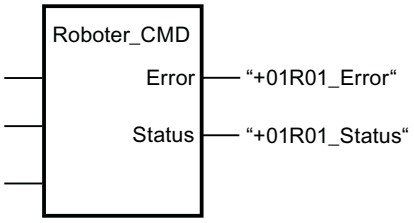

The text list with the error texts of the robot has, for example, the following form:

Text list number	Entry
1	Alarm / Problem 1111
2	Alarm / Problem 2222
3	Alarm / Problem 3333
4	Alarm / Problem 4444

The following tags are required to use this function block in your program code:

Tags	Explanation
+01R01_Error	A Boolean variable as actuator for the error text.
+01R01_Status	Corresponds to the error number

Procedure

Proceed as follows to realize outputting of the error texts of the robot:

1. Create the two required tags in the default tag table.

		Name	Data type	Address
1		+01R01_Error	Bool	%E0.0
2		+01R01_Status	UInt	%EW0

2. Call the supplied function block in your program code.
3. Enter the following values into the properties of the alarm:

Properties	Values	Operand status / Explanation
Type of supervision	Error message	-
Supervised tag	+01R01_Error	TRUE As soon as the function block of the robot outputs an error and sets the error flag, the tag is set to the signal state TRUE and the supervision alarm is triggered.
Delay time	T#0ms	The supervision alarm is to be output immediately.
Condition 1	-	The conditions are not required.
Condition 2	-	
Condition 3	-	
Category	1: Error	Triggering a supervision alarm is classified as an error.

Properties	Values	Operand status / Explanation
Subcategory 1	-	The subcategories are not required.
Subcategory 2	-	
ProDiag FB	Default_SupervisionFB	This is the associated ProDiag function block. As long as you are not using your own function block, the Default_SupervisionFB is used by default.
Error flag	Default_SupervisionDB.Trigger_Message_O_1.Err	This error flag is set as soon as a fault occurs.
Specific text field	Following associated values • @4%4d@ (error message) • @4%t#T_Robotor@ (text from text list)	Error text that is output in the supervision alarm.

You can find additional information on the associated values here: Use of the associated values (Page 7898)

Result

As soon as the "+01R01_Error" tag assumes the signal state TRUE, the supervision alarm with the text from the specific text field is output. The error number in three-digit decimal form and the error text from the text list are displayed.

Depending on the settings you have made for error text in the ProDiag supervision settings, an error message may look as follows:

Error message

```
Error: Err_Txt: CPU_50: Default_SupervisionFB : +01R01_Error : 3: Alarm / Problem 3333
```

24.3.5 Application examples for supervisions with ProDiag (S7-1500)

Description

You can find a detailed application example for supervisions with ProDiag in the Siemens Industry online support here: Application example for ProDiag (https://support.industry.siemens.com/cs/ww/en/view/109740151)

24.4 Creating ProDiag function blocks (S7-1500)

24.4.1 Overview of the ProDiag function blocks (S7-1500)

Description

To structure the program code according to the machine and plant units, e.g. Station_1, Station_2, etc., you can create a ProDiag function block for each unit and name it accordingly. A ProDiag function block is created in the PRODIAG programming language and can have a maximum of 250 supervisions in version 1.0 and 1000 supervisions in version 2.0.

As soon as you create a new ProDiag function block, a corresponding ProDiag instance data block is created.

A detailed view of the ProDiag blocks used in your program code and how they relate to one another can be found here:Displaying supervised user-defined blocks (Page 7927)

Advantages of user-defined ProDiag function blocks

The user-defined units offer a number of advantages:

- ProDiag units allow you to better structure your program and benefit from performance advantages.
- Each unit offers group error bits that you can use to define reactions in the user program. In addition, you can implement a simple display on the HMI device.
- You can use the "ProDiag overview" object to create a general overview of all group error bits of this unit on an HMI device.

See also

Creating ProDiag function blocks when creating global supervisions (Page 7839)

Structure of a ProDiag function block (Page 7839)

Overview of calling the ProDiag function blocks (Page 7916)

Assigning ProDiag function blocks (Page 7906)

Creating ProDiag function blocks (Page 7839)

Editing ProDiag function blocks (Page 7913)

24.4.2 Structure of a ProDiag function block (S7-1500)

Structure of a ProDiag function block

The following graphic shows the components of a ProDiag function block as an example:

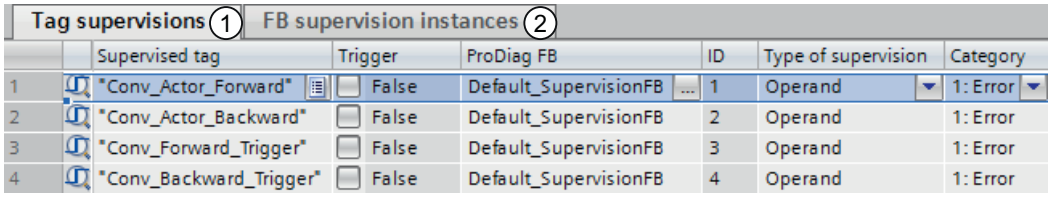

Numbering	Explanation	Additional information
1	Overview of all global supervisions	All supervisions that are assigned to this ProDiag function block.
2	Overview of all instantiated function block supervisions	

24.4.3 Saving the layout of the columns in the ProDiag FB (S7-1500)

You can adapt the display of the columns in the ProDiag function block to meet your requirements. For example, you can hide columns that you do not need. You cannot change the sorting order of columns. You can then save your customized view.

Procedure

Proceed as follows to save the layout of the columns:

1. Adapt the columns according to your requirements.
2. Click the "Save window settings" button in the ProDiag function block.

Result

The layout is saved. When you reopen the ProDiag function block, this layout will be used.

24.4.4 Creating ProDiag function blocks (S7-1500)

24.4.4.1 Creating ProDiag function blocks when creating global supervisions (S7-1500)

Requirement

You have created at least one global tag.

Procedure

To create a ProDiag function block directly when creating a global supervision, follow these steps:

1. Create a supervision for a global tag.
   The "Supervisions" tab opens in the "Properties" Inspector window and a supervision for the tag is created. When creating the first supervision, the ProDiag function block "Default_SupervisionFB" is created automatically.
2. Enter the desired properties.
3. Click the "…" selection field at ProDiag FB.
   A selection dialog opens.
4. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
5. To create a new ProDiag function block, click "Add new" in the right-hand column.
   The "Add new block" dialog opens.
6. Click the "Function block" button.
7. Enter a name for the new ProDiag function block.
8. Select in the "Language > PRODIAG (incl. IDB)" section.
9. To enter additional properties for the new ProDiag function block, click "Additional information".
   An area with additional entry fields is displayed.
10. Enter the desired properties.
11. Select the "Add new and open" check box if the ProDiag function block does not open as soon as it is created.
12. Confirm your entries with "OK".

Result

In addition to the new ProDiag function block, a corresponding ProDiag instance data block is also created. You can find the blocks in the project tree in the "Program blocks" folder.

24.4.4.2 Creating ProDiag function blocks during the block call (S7-1500)

Requirement

You have created a ProDiag function block and at least one local supervision.

Procedure

Proceed as follows to create a ProDiag function block directly when calling a function block:

1. Call the function block that contains the supervisions in the program code.
   The "Call options" dialog opens.
2. Click the "Single instance" button.

3. Click the "…" selection field at ProDiag FB.
   A selection dialog opens. If it exists, the ProDiag function block "Default_SupervisionFB" is displayed automatically. If this does not exist, the first user-defined ProDiag function block to be found is displayed.
4. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
5. To create a new ProDiag function block, click "Add new" in the right-hand column.
   The "Add new block" dialog opens.
6. Click the "Function block" button.
7. Enter a name for the new ProDiag function block.
8. Select in the "Language > PRODIAG (incl. IDB)" section.
9. To enter additional properties for the new ProDiag function block, click "Additional information".
   An area with additional entry fields is displayed.
10. Enter the desired properties.
11. Select the "Add new and open" check box if the ProDiag function block does not open as soon as it is created.
12. Confirm your entries with "OK".

Result

In addition to the new ProDiag function block, a corresponding ProDiag instance data block is also created. You can find the blocks in the project tree in the "Program blocks" folder. The supervisions of the function block are instantiated.

24.4.4.3 Creating ProDiag function blocks at the instance data block of a function block (S7-1500)

Requirement

You have created and compiled a function block with at least one local supervision.

Procedure

Proceed as follows to create a ProDiag function block in the properties of the instance data block:

1. Right-click the instance data block of a function block that contains local supervisions.
2. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the instance data block opens.
3. Click the "Attributes" section.
4. Click the "…" selection field at Assign ProDiag FB.
   A selection dialog opens.
5. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.

6. To create a new ProDiag function block, click "Add new" in the right-hand column.
   The "Add new block" dialog opens.
7. Click the "Function block" button.
8. Enter a name for the new ProDiag function block.
9. Select in the "Language > PRODIAG (incl. IDB)" section.
10. To enter additional properties for the new ProDiag function block, click "Additional information".
    An area with additional entry fields is displayed.
11. Enter the desired properties.
12. Select the "Add new and open" check box if the ProDiag function block does not open as soon as it is created.
13. Confirm your entries with "OK".

Result

In addition to the new ProDiag function block, a corresponding ProDiag instance data block is also created. You can find the blocks in the project tree in the "Program blocks" folder. The supervisions of the function block are instantiated.

24.4.4.4 Creating ProDiag function blocks by using the "Add new block" dialog. (S7-1500)

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a ProDiag function block using the "Add new block" dialog, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog opens.
2. Click the "Function block" button.
3. Enter a name for the new ProDiag function block.
4. Select in the "Language > PRODIAG (incl. IDB)" section.
5. To enter additional properties for the new ProDiag function block, click "Additional information".
   An area with additional entry fields is displayed.
6. Enter the desired properties.
7. Select the "Add new and open" check box if the ProDiag function block does not open as soon as it is created.
8. Confirm your entries with "OK".

Or:

Proceed as follows to assign a ProDiag function block to an existing user-defined function block that contains local supervisions by using the "Add new block" dialog:

1. Double-click the "Add new block" command.
   The "Add new block" dialog opens.
2. Click "Data block".
3. At "Type" select the existing function block which contains local supervisions from the drop-down list.
4. Click the "…" selection field at ProDiag FB.
   A selection dialog opens.
5. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
6. To create a new ProDiag function block, click "Add new" in the right-hand column.
   The "Add new block" dialog opens.
7. From this point, the procedure described above from item 2 applies.

Result

In addition to the new ProDiag function block, a corresponding ProDiag instance data block is also created. You can find the blocks in the project tree in the "Program blocks" folder.

24.4.4.5 Creating ProDiag function blocks in the ProDiag overview tables (S7-1500)

Requirement

You have created local and global supervisions.

Procedure

Proceed as follows to create a ProDiag function block in the ProDiag overview tables "Tag supervisions" and "FB supervision instances":

1. Click the "…" selection field in the "ProDiag FB" column.
   A selection dialog opens.
2. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
3. To create a new ProDiag function block, click "Add new" in the right-hand column.
   The "Add new block" dialog opens.
4. Click the "Function block" button.
5. Enter a name for the new ProDiag function block.
6. Select in the "Language > PRODIAG (incl. IDB)" section.
7. To enter additional properties for the new ProDiag function block, click "Additional information".
   An area with additional entry fields is displayed.

8. Enter the desired properties.
9. Select the "Add new and open" check box if the ProDiag function block does not open as soon as it is created.
10. Confirm your entries with "OK".

Result

In addition to the new ProDiag function block, a corresponding ProDiag instance data block is also created. You can find the blocks in the project tree in the "Program blocks" folder. The local supervisions are instantiated.

24.5 Using a central time stamp (S7-1500)

24.5.1 Using a central time stamp (S7-1500)

Description

As of ProDiag function block version 2.0, you have the option to specify a central time stamp tag in the ProDiag supervision settings and then use it in every CPU in the entire project.

The central time stamp tag is defined as a global tag or as a global data component in the relevant CPU.

The same system time is assigned to all ProDiag supervision alarms within a program cycle that are sent by ProDiag function blocks when the "Use central time stamps" option is enabled.

Requirement

The ProDiag function block has version 2.0.

Note

ProDiag function block version 1.0

The local block time stamp continues to be assigned to all ProDiag supervision alarms that are sent by a version 1.0 ProDiag function block.

Procedure

To use a central time stamp, follow these steps:

1. Ensure that the block has version 2.0 in the block properties of the ProDiag function block.
2. In the project tree under "Common data" > "Supervision settings" > "General", open the "Central time stamp" area.
3. Define a time stamp tag by entering a free text into the text field.

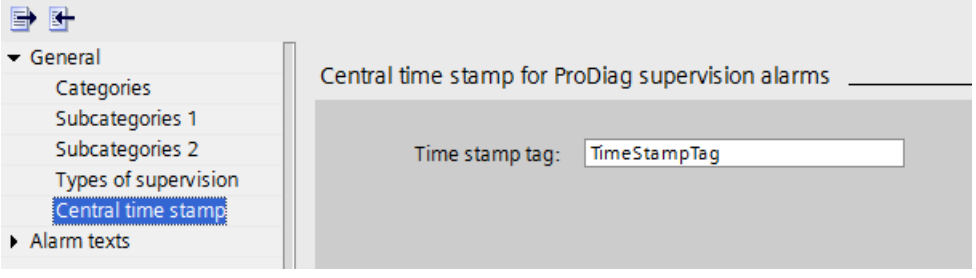

4. In each CPU in which you want to use this time stamp, define a global tag or global data block with the label from the free text field of the LDT data type.

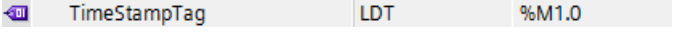

5. To copy the current time tamp "TimeStampTag" to the global time tamp tag, expand your program code (for example, Main[OB1]) in the corresponding CPUs with a time function, for example "RD_SYS_T":

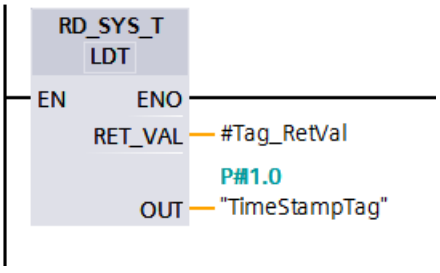

6. If you want to use the central time stamp in existing ProDiag function blocks, you need to select the "Use central time stamp" option in the block properties of the respective ProDiag function block:

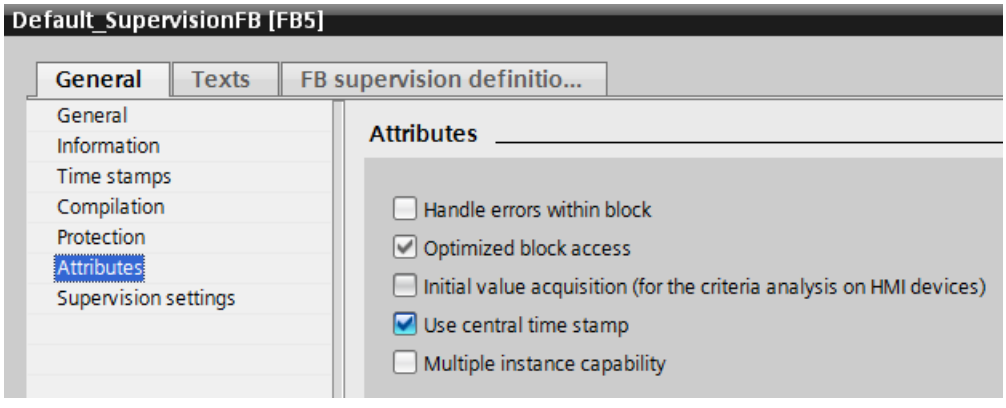

If you create the ProDiag function blocks only after defining the central time stamp tag, this option is activated automatically.

Result

All ProDiag supervision alarms whose function block has enabled the "Use central time stamp" option are given a uniform time stamp within a program cycle.

24.5.2 Multiple supervision alarms of a ProDiag FB within one cycle time (S7-1500)

Description

Only one supervision alarm can be displayed for each ProDiag function block per CPU cycle time, even if several faults have occurred simultaneously. All simultaneously triggered ProDiag supervision alarms of a ProDiag function block receive the same time stamp, which contains the time of the triggering fault. The remaining supervision alarms, i.e. one for each CPU cycle time, can be processed one after the other. This prevents jumps in the cycle time caused by alarm storms.

You can display the time stamps of the supervision alarms, for example, in the "Time" column in the alarm display.

Depending on whether or not you have enabled the "Use central time stamp" option in the ProDiag function block, either the current system time or the local block time is used as a time stamp.

See also

Using a central time stamp (Page 7845)

24.6 Using ProDiag instance data bases (S7-1500)

24.6.1 Structure of a ProDiag instance data block of version 1.0 (S7-1500)

Structure of a ProDiag instance data block of version 1.0

The following graphic shows the components of a ProDiag instance data block as an example:

Default_SupervisionDB (1)

	Name	Data type
1	Input	
2	Output	
3	InOut	
4	Static	
5	State (2)	SV_FB_State
6	Conv_Actor_Forward_O_2	SV_State
7	Conv_Actor_Backward_O_2	SV_State
8	Conv_Forward_Trigger_O_2	SV_State
9	Conv_Actor_Forward_O_1	SV_State
10	Conv_Actor_Backward_O_1	SV_State
11	Conv_Forward_Trigger_O_1 (3)	SV_State
12	Conv_Backward_Trigger_O_1	SV_State
13	Conv_Switch_Left_O_1	SV_State
14	Conv_Switch_Right_O_1	SV_State
15	Conv_Auto_Hand_O_1	SV_State
16	Conv_Push_Button_Forward_O_1	SV_State
17	Conv_Push_Button_Backward_O_1	SV_State

Num-bering	Explanation	Additional information
1	The block interface of the ProDiag instance data block	The block interface cannot be edited.
2	The "State" status tag in the "Static" section	The "State" status tag of the "SV_FB_State" PLC data type contains the group error bits of all the supervisions that are assigned to this ProDiag instance data block. There are two different types of group error bits: • Group error bit for the supervision types (All, O, I, R, A, P, Merr and Mtxt) • Group error bit for the categories (C1, C2, C3, C4, C5, C6, C7, and C8) You can visualize these group error bits either in your HMI device with the "ProDiag overview" object or you can program them in your program code as condition for further execution of the program code. You only have read access to the group error bits in the program code. As soon as a supervision alarm is triggered, the group error bits of the assigned ProDiag instance data block are set.
3	Status tags	These tags are displayed if supervisions have been created and are assigned to this ProDiag instance data block.

24.6.2 Content of a ProDiag instance data block of version 1.0 (S7-1500)

Block interface

As soon as you create a new ProDiag function block, a corresponding ProDiag instance data block is created.

The following elements are available in the "Static" section:

State

Name	Data type	Explanation
State	SV_FB_State	System-internal, know-how-protected PLC data type

Name	Data type	Explanation
Total_Number	UINT	Total number of all supervisions to which this ProDiag function block was assigned.
All	BOOL	Group error bit for all contained types of supervision, except text message: • TRUE => At least one supervision has been triggered in the current program cycle. • FALSE => No supervision has been triggered. Accessible from HMI and visible in HMI.
O (Operand)	BOOL	Group error bit for all operand supervisions: • TRUE => At least one operand supervision has been triggered in the current program cycle. • FALSE => No operand supervision has been triggered.
I (Interlock)	BOOL	Group error bit for all interlock supervisions: • TRUE => At least one interlock supervision has been triggered in the current program cycle. • FALSE => No interlock supervision has been triggered.
R (Reaction)	BOOL	Group error bit for all reaction supervisions: • TRUE => At least one reaction supervision has been triggered in the current program cycle. • FALSE => No reaction supervision has been triggered.
A (Action)	BOOL	Group error bit for all action supervisions: • TRUE => At least one action supervision has been triggered in the current program cycle. • FALSE => No action supervision has been triggered.
P (Position)	BOOL	Group error bit for all position supervisions: • TRUE => At least one position supervision has been triggered in the current program cycle. • FALSE => No position supervision has been triggered.
Merr (Message error)	BOOL	Group error bit for all error messages: • TRUE => At least one error message has been triggered in the current program cycle. • FALSE => No error message has been triggered.
Mtxt (Message text)	BOOL	Group error bit for all text messages: • TRUE => At least one text message was triggered in the current program cycle. • FALSE => No text message was triggered.
SV_Types (Types of supervision)	BYTE	Byte for all group error bits of the types of supervision A type of supervision is assigned to each individual bit. As a result, you can use this byte, for example, in the "ProDiag overview" object on the HMI device to create

Name		Data type	Explanation
			a display that shows at first glance when a supervision of a certain type of supervision has been triggered. • Bit 1 => Operand • Bit 2 => Interlock • etc. Accessible from HMI and visible in HMI.
	C1 (Category)	BOOL	First category (The first category = Error by default. This cannot be changed.) • TRUE => At least one supervision alarm of category C1 has been triggered in the current program cycle. • FALSE => No error has occurred.
	C2 (Category)	BOOL	Second category (The second category = Warning by default. You can change the setting at any time.) • TRUE => At least one supervision alarm of category C2 has been triggered in the current program cycle. • FALSE => There are no faults.
	C3 (Category)	BOOL	Third category (The third category = Info by default. You can change the setting at any time.) • TRUE => At least one supervision alarm of category C3 has been triggered in the current program cycle. • FALSE => There are no faults.
	C4 (Category)	BOOL	Fourth category • TRUE => At least one supervision alarm of category C4 has been triggered in the current program cycle. • FALSE => There are no faults.
	C5 (Category)	BOOL	Fifth category • TRUE => At least one supervision alarm of category C5 has been triggered in the current program cycle. • FALSE => There are no faults.
	C6 (Category)	BOOL	Sixth category • TRUE => At least one supervision alarm of category C6 has been triggered in the current program cycle. • FALSE =>There are no faults.
	C7 (Category)	BOOL	Seventh category • TRUE => At least one supervision alarm of category C7 has been triggered in the current program cycle. • FALSE => There are no faults.
	C8 (Category)	BOOL	Eighth category

Name		Data type	Explanation
			• TRUE => At least one supervision alarm of category C8 has been triggered in the current program cycle. • FALSE => There are no faults.
	Categories	BYTE	Group error byte for all categories A category is assigned to each individual bit. As a result, you can use this byte, for example, in the ProDiag overview object on the HMI device to create a display that shows at first glance when a supervision of a certain category has been triggered. • Bit 1 => Category 1 (C1) • Bit 2 => Category 2 (C2) • etc. Accessible from HMI and visible in HMI.
	Sub_Cat_1 (Subcategory 1)	DWORD	Group error bits for subcategories 1 (Bit 1 to 16)
	Sub_Cat_2 (Subcategory 2)	DWORD	Group error bits for subcategories 2 (Bit 1 to 16)
	Reserved	BYTE	Reserve byte

Note

Additional information on the HMI object "ProDiag overview"

For additional information, refer to the information system of the TIA Portal. "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag > Visualizing supervisions > Configuring the ProDiag overview"

Optional status tag as soon as a supervision has been created

As soon as you create a supervision, this section is displayed in the block interface in addition:

Name	Data type	Explanation
e.g. Conv_Push_Button_Forward_O_1 • Conv_Push_Button_Forward = Name of the supervised operand • O = "Operand" type of supervision • 1 = Total number of supervisions of the "Operand" type of supervision	SV_State	System-internal, know-how-protected PLC data type
Err (Error)	BOOL	Error that is also displayed in the properties of the supervision. Indicates whether a supervision alarm has been triggered. • TRUE = Supervision alarm was triggered • FALSE = No supervision alarm, which means the fault has not occurred Accessible from HMI and visible in HMI
Err_Neg_Flag (Error_Negativ_Flag)	BOOL	For the negative edge evaluation of the error flag
T_D_E_Going (Time_Date_Error_Going)	LDT	Time and day in the program cycle at which the fault went out
E_Flag (Error_Flag)	BOOL	Error flag: Only one supervision alarm is sent per ProDiag function block and per program cycle.
E_Pos_Flag (Error_Positive_Flag)	BOOL	For the positive edge evaluation of the error flag
T_D_E_Coming (Time_Date_Error_Coming)	LDT	Time and day in the program cycle at which the fault occurred.
S_Coming (Signal_Coming)	BOOL	Bit memory • TRUE = Supervision alarm was sent fault-free • FALSE = There is a fault/error.
Pos_Flag (Positive_Flag)	BOOL	This bit memory is only needed for the "Position" type of supervision.
Ti (Timer)	TON_TIME	This time is needed to generate the delay time.
PT (Preset Time)	TIME	Preset time value
ET (Elapsed Time)	TIME	Current time value
IN (Input)	TIME	Start input
Q (Output)	TIME	Output that is set after expiration of time PT

24.6.3 Structure of a ProDiag instance data block of version 2.0 (S7-1500)

Structure of a ProDiag instance data block of version 2.0

The following graphic shows the components of a ProDiag instance data block as an example:

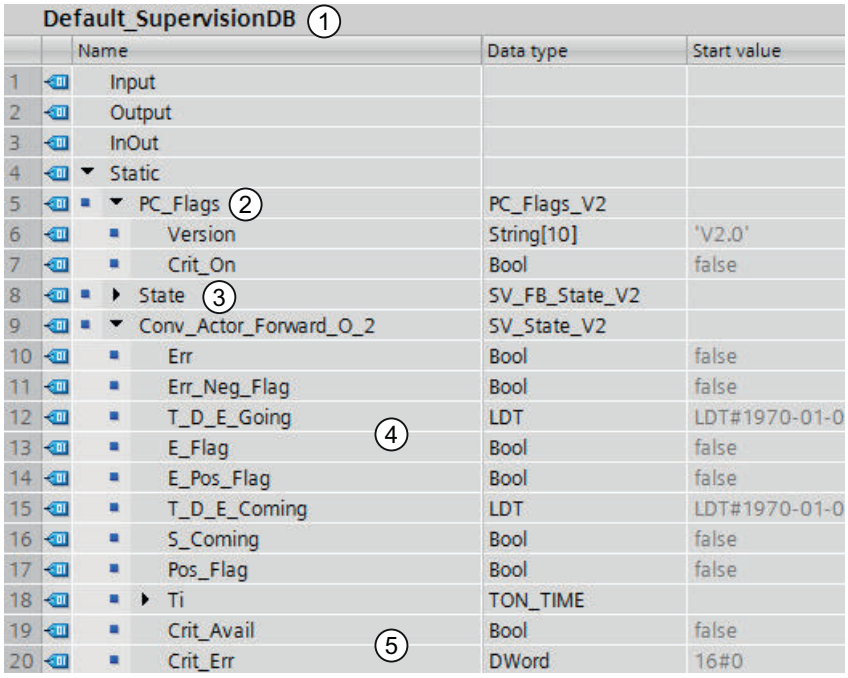

	Name	Data type	Start value
1	Input		
2	Output		
3	InOut		
4	Static		
5	PC_Flags (2)	PC_Flags_V2	
6	Version	String[10]	'V2.0'
7	Crit_On	Bool	false
8	State (3)	SV_FB_State_V2	
9	Conv_Actor_Forward_O_2	SV_State_V2	
10	Err	Bool	false
11	Err_Neg_Flag	Bool	false
12	T_D_E_Going (4)	LDT	LDT#1970-01-0
13	E_Flag	Bool	false
14	E_Pos_Flag	Bool	false
15	T_D_E_Coming	LDT	LDT#1970-01-0
16	S_Coming	Bool	false
17	Pos_Flag	Bool	false
18	Ti	TON_TIME	
19	Crit_Avail (5)	Bool	false
20	Crit_Err	DWord	16#0

Num-bering	Explanation	Additional information
1	The block interface of the ProDiag instance data block	The block interface cannot be edited.
2	The status tag "PC_Flags"	The "PC_Flags" status tag of the "PC_Flags_V2" PLC data type contains the version of the ProDiag function block (Version parameter) and the CRIT_ON parameter which indicates whether initial value acquisition is enabled.

Num-bering	Explanation	Additional information
3	The status tag "State"	The "State" status tag of the "SV_FB_State_V2" PLC data type contains the group error bits of all the supervisions that are assigned to this ProDiag instance data block. There are two different types of group error bits: • Group error bit for the supervision types (All, O, I, R, A, P, Merr and Mtxt) • Group error bit for the categories (C1, C2, C3, C4, C5, C6, C7, and C8) You can visualize these group error bits either in your HMI device with the "ProDiag overview" object or you can program them in your program code as condition for further execution of the program code. You have read and write access to the group error bits in the program code. As soon as a supervision alarm is triggered, the group error bits of the assigned ProDiag instance data block are set.
4	Status tags	These tags of the "SV_State_V2" PLC data type are displayed if supervisions have been created and are assigned to this ProDiag instance data block.
5	The parameters Crit_Avail and Crit_Err	• Crit_Avail: Signal state = TRUE: Initial values were recorded. • Crit_Err: Storing the 32 signal states per network of a LAD or FBD block.

24.6.4 Content of a ProDiag instance data block of version 2.0 (S7-1500)

Block interface

As soon as you create a new ProDiag function block, a corresponding ProDiag instance data block is created.

The following elements are available in the "Static" section:

State

Name		Data type	Explanation
PC_Flags		PC_Flags_V2	System-internal, know-how-protected PLC data type
	Version	String[10]	Version of the ProDiag function block

Name		Data type	Explanation
	Crit_On	BOOL	Indicates whether the initial value acquisition is enabled.
State		SV_FB_State_V2	System-internal, know-how-protected PLC data type

Name	Data type	Explanation
Total_Number	UINT	Total number of all supervisions to which this ProDiag function block was assigned.
All	BOOL	Group error bit for all contained types of supervision, except text message: • TRUE => At least one supervision has been triggered in the current program cycle. • FALSE => No supervision has been triggered. Accessible from HMI and visible in HMI.
O (Operand)	BOOL	Group error bit for all operand supervisions: • TRUE => At least one operand supervision has been triggered in the current program cycle. • FALSE => No operand supervision has been triggered.
I (Interlock)	BOOL	Group error bit for all interlock supervisions: • TRUE => At least one interlock supervision has been triggered in the current program cycle. • FALSE => No interlock supervision has been triggered.
R (Reaction)	BOOL	Group error bit for all reaction supervisions: • TRUE => At least one reaction supervision has been triggered in the current program cycle. • FALSE => No reaction supervision has been triggered.
A (Action)	BOOL	Group error bit for all action supervisions: • TRUE => At least one action supervision has been triggered in the current program cycle. • FALSE => No action supervision has been triggered.
P (Position)	BOOL	Group error bit for all position supervisions: • TRUE => At least one position supervision has been triggered in the current program cycle. • FALSE => No position supervision has been triggered.
Merr (Message error)	BOOL	Group error bit for all error messages: • TRUE => At least one error message has been triggered in the current program cycle. • FALSE => No error message has been triggered.
Mtxt (Message text)	BOOL	Group error bit for all text messages: • TRUE => At least one text message was triggered in the current program cycle. • FALSE => No text message was triggered.
SV_Types (Types of supervision)	BYTE	Byte for all group error bits of the types of supervision A type of supervision is assigned to each individual bit. As a result, you can use this byte, for example, in the "ProDiag overview" object on the HMI device to create

Name	Data type	Explanation
		a display that shows at first glance when a supervision of a certain type of supervision has been triggered. • Bit 1 => Operand • Bit 2 => Interlock • etc. Accessible from HMI and visible in HMI.
C1 (Category)	BOOL	First category (The first category = Error by default. This cannot be changed.) • TRUE => At least one supervision alarm of category C1 has been triggered in the current program cycle. • FALSE => No error has occurred.
C2 (Category)	BOOL	Second category (The second category = Warning by default. You can change the setting at any time.) • TRUE => At least one supervision alarm of category C2 has been triggered in the current program cycle. • FALSE => There are no faults.
C3 (Category)	BOOL	Third category (The third category = Info by default. You can change the setting at any time.) • TRUE => At least one supervision alarm of category C3 has been triggered in the current program cycle. • FALSE => There are no faults.
C4 (Category)	BOOL	Fourth category • TRUE => At least one supervision alarm of category C4 has been triggered in the current program cycle. • FALSE => There are no faults.
C5 (Category)	BOOL	Fifth category • TRUE => At least one supervision alarm of category C5 has been triggered in the current program cycle. • FALSE => There are no faults.
C6 (Category)	BOOL	Sixth category • TRUE => At least one supervision alarm of category C6 has been triggered in the current program cycle. • FALSE =>There are no faults.
C7 (Category)	BOOL	Seventh category • TRUE => At least one supervision alarm of category C7 has been triggered in the current program cycle. • FALSE => There are no faults.
C8 (Category)	BOOL	Eighth category

Name		Data type	Explanation
			• TRUE => At least one supervision alarm of category C8 has been triggered in the current program cycle. • FALSE => There are no faults.
	Categories	BYTE	Group error byte for all categories A category is assigned to each individual bit. As a result, you can use this byte, for example, in the ProDiag overview object on the HMI device to create a display that shows at first glance when a supervision of a certain category has been triggered. • Bit 1 => Category 1 (C1) • Bit 2 => Category 2 (C2) • etc. Accessible from HMI and visible in HMI.
	Sub_Cat_1 (Subcategory 1)	DWORD	Group error bits for subcategories 1 (Bit 1 to 16)
	Sub_Cat_2 (Subcategory 2)	DWORD	Group error bits for subcategories 2 (Bit 1 to 16)
	Reserved	BYTE	Reserve byte

Note

Additional information on the HMI object "ProDiag overview"

For additional information, refer to the information system of the TIA Portal. "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag > Visualizing supervisions > Configuring the ProDiag overview"

Optional status tag as soon as a supervision has been created

As soon as you create a supervision, this section is displayed in the block interface in addition:

Name	Data type	Explanation
e.g. Conv_Push_Button_Forward_O_1 • Conv_Push_Button_Forward = Name of the supervised operand • O = "Operand" type of supervision • 1 = Total number of supervisions of the "Operand" type of supervision	SV_State_V2	System-internal, know-how-protected PLC data type
Err (Error)	BOOL	Error that is also displayed in the properties of the supervision. Indicates whether a supervision alarm has been triggered. • TRUE = Supervision alarm was triggered • FALSE = No supervision alarm, which means the fault has not occurred Accessible from HMI and visible in HMI
Err_Neg_Flag (Error_Negativ_Flag)	BOOL	For the negative edge evaluation of the error flag
T_D_E_Going (Time_Date_Error_Going)	LDT	Time and day in the program cycle at which the fault went out
E_Flag (Error_Flag)	BOOL	Error flag: Only one supervision alarm is sent per ProDiag function block and per program cycle.
E_Pos_Flag (Error_Positive_Flag)	BOOL	For the positive edge evaluation of the error flag
T_D_E_Coming (Time_Date_Error_Coming)	LDT	Time and day in the program cycle at which the fault occurred.
S_Coming (Signal_Coming)	BOOL	Bit memory • TRUE = Supervision alarm was sent fault-free • FALSE = There is a fault/error.
Pos_Flag (Positive_Flag)	BOOL	This bit memory is only needed for the "Position" type of supervision.
Ti (Timer)	TON_TIME	This time is needed to generate the delay time.
PT (Preset Time)	TIME	Preset time value
ET (Elapsed Time)	TIME	Current time value
IN (Input)	TIME	Start input
Q (Output)	TIME	Output that is set after expiration of time PT
Crit_Avail	BOOL	Signal state = TRUE: Initial values were recorded.
Crit_Err	DWORD	Storing the 32 signal states per network of a LAD or FBD block.

24.7 Defining properties for a ProDiag function block (S7-1500)

24.7.1 Structure of supervision settings at a ProDiag FB (S7-1500)

Structure of supervision settings

The following figure shows the components of the supervision settings:

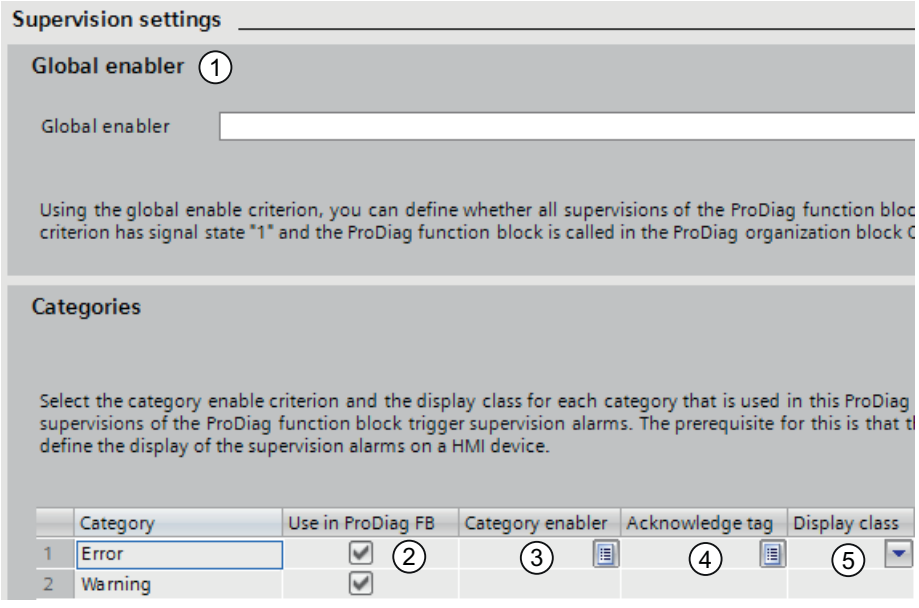

Numbering	Explanation	Additional information
1	Global enabler of a ProDiag function block	You can use the global enabler to control the call of a ProDiag function blocks in the ProDiag organization block. You can find additional information here: Using the global enabler (Page 7864)
2	The "Use in ProDiag FB" cannot be edited.	If a supervision of this category is assigned to the ProDiag function block, the category is activated.
3	Category enabler	You can use the category enabler to define an enabler (global Boolean tag) for the ProDiag function block and for each individual category contained therein and use this to activate or deactivate the supervisions of this category. You can find additional information here: Using the category enabler (Page 7865)

Numbering	Explanation	Additional information
4	Acknowledge tag	You can use the acknowledge tag to acknowledge a supervision alarm of a category within the STEP 7 project. This ensures that the supervision alarm only goes again when the problem has been eliminated. You can control the program sequence with the relevant error flag. You can find additional information here: Use acknowledge tag (Page 7866)
5	Display class	You can use the display class to define which supervisions are to be displayed on an HMI device. You can find additional information here: Setting the display class (Page 7867)

24.7.2 Creating the enablers for supervision alarms (S7-1500)

24.7.2.1 Overview of the enablers (S7-1500)

Description

You have the option of creating enablers (Boolean tags) in the settings of a ProDiag function block. Two different types of enablers are available for this:

- Global enabler
- Category enabler

Global enabler

You can control the call of the ProDiag function blocks in the ProDiag organization block using the global enabler. The ProDiag is created automatically as soon as you compile your project. For example, you can deactivate all supervisions of a ProDiag function block by defining a global enabler that is used automatically by the system as a call condition as soon as the ProDiag function block is called in the ProDiag organization block. If the call condition is not met (signal state = False), the ProDiag function block is not called and no supervision alarms are displayed. The global enabler must be a global Boolean tag.

However, when you do not call the ProDiag function block in ProDiag OB but in another block within your program, you must integrate the global enabler in the program code yourself if you wish to use it.

The global enabler offers you the following advantages:

- You can control which ProDiag function block is to be processed in which program cycle. All ProDiag function blocks that have not been processed do not display any supervision alarms even if a fault has occurred.
- If you assign the same tag as an enabler to all ProDiag function blocks, you can easily program, for example, that no undesired supervisions are to be triggered during commissioning.
- In the case of very critical cycle times, you can achieve performance improvements by assigning different global Boolean tags as enablers to the ProDiag function blocks and using this to selectively call the blocks in the program code.
- If a supervision alarm has already been triggered, it exists until the corresponding ProDiag function block is called again. As soon as the ProDiag function block is processed again, the supervision alarm is displayed.

Category enabler

You can use the category enabler to define an enabler (global Boolean tag) for each ProDiag function block and for each individual category contained therein and use this to activate or deactivate the supervisions of this category. In this way, you can influence all supervisions of the "Error" category of a ProDiag function block, for example. If you use the category enabler as a call condition within your program code and the call condition (signal state = False) is not met during runtime, none of the supervisions of this category are executed. This means that the group error bits in the block interface of the ProDiag function block are not changed and no supervision alarms are displayed.

The category enabler offers you the following advantages:

- You can selectively deactivate the supervision alarms of an individual category within a ProDiag function block.
- If a supervision alarm has already been triggered, it is revoked (identified as "Going"). As long as the global Boolean tag has the signal state = False, for example, during maintenance work, no unnecessary supervision alarms are displayed.

See also

Using the global enabler (Page 7864)

Using the category enabler (Page 7865)

24.7.2.2 Using the global enabler (S7-1500)

Requirement

You have created a ProDiag function block and a supervision.

Procedure

To use a global enabler, follow these steps:

1. Right-click the ProDiag function block and select "Properties..." from the shortcut menu. The properties dialog box of the ProDiag function block opens.
2. In the "General" tab, select the "Supervision settings" entry.
3. Specify a global Boolean tag as global enabler.
   For the global enabler to have an effect on the sequence of the program code, you must not call the ProDiag function block in a user-defined program block. This means that ProDiag function block is automatically called in the ProDiag organization block.

Result

Only those supervision alarms of the ProDiag function blocks whose global enabler has the signal state "1" are displayed.

24.7.2.3 Using the category enabler (S7-1500)

Requirement

You have created a ProDiag function block and a supervision.

Procedure

To use a category enabler, follow these steps:

1. Right-click the ProDiag function block and select "Properties..." from the shortcut menu. The properties dialog box of the ProDiag function block opens.
2. In the "General" tab, select the "Supervision settings" entry.
3. Use an enabler for the individual categories in the "Categories" area.
   For the enabler to also have an effect on the sequence of the program code, you must use it as step enabling condition in the program code.

Result

Only those supervision alarms of the categories of the ProDiag function block whose enabler has the signal state "1" are displayed.

24.7.3 Defining an acknowledge tag for individual categories (S7-1500)

24.7.3.1 Overview of the acknowledge tags (S7-1500)

Description

You have the option of using a global Boolean tag as an acknowledge tag for each individual category in each ProDiag function block.

By using the acknowledge tag you initially bring about that the error flag and the group error bits retain the signal status TRUE and that the supervision alarm does not go. So that, for example, the conveyor belt is stopped immediately you need to use these error bits explicitly in your program code. This provides you with the option that the conveyor belt, for example, does not start up again until the supervision error has been explicitly acknowledged by the machine operator even if the cause of the fault, such as a breached light barrier, has already been corrected. There is also the acknowledgment of the supervision alarm on an HMI device; however, it does not have an impact on the processing of the program code.

You can use a separate acknowledge tag for each of the 8 categories and incorporate this in your program code as a step enabling condition for the ongoing program sequence.

The error bits are retained in the ProDiag function block once the fault has been corrected until the signal state of the acknowledge tag changes from "0" to "1". The processing of the program code cannot continue until you confirm the acknowledge tag, which means the signal state changes from "0" to "1":

- The error flag is reset to the signal state "0" once again.
- The group error bits for HMI are also reset to "0".
- The supervision alarm on the HMI device disappears.

24.7.3.2 Use acknowledge tag (S7-1500)

Requirement

You have created a ProDiag function block and a supervision.

Procedure

To use an acknowledge tag, follow these steps:

1. Right-click the ProDiag function block and select "Properties..." from the shortcut menu. The properties dialog box of the ProDiag function block opens.
2. In the "General" tab, select the "Supervision settings" entry.
3. Use acknowledge tags for the individual categories in the "Categories" area.
   For the acknowledge tags to have an effect on the sequence of the program code, you must use the error flag as a step enabling condition in the program code.

Result

A fault has occurred and the supervision alarm was triggered. The program code is not processed further until the fault has been corrected and the acknowledge tag has been confirmed.

24.7.4 Setting display classes (S7-1500)

24.7.4.1 Overview of display classes (S7-1500)

Description

You have the option of defining a display class for each individual category in each ProDiag function block.

You can use the display classes to define which supervisions are to be displayed on an HMI device. This is useful, in particular, when multiple HMI devices are connected to a CPU. In order for the supervisions to be correctly displayed, you must activate the display classes that are to be displayed for each HMI device.

A maximum of 17 display classes are possible.

24.7.4.2 Setting the display class (S7-1500)

Requirement

You have created a ProDiag function block and a supervision.

Procedure

To set a display class, follow these steps:

1. Right-click the ProDiag function block and select "Properties..." from the shortcut menu. The properties dialog box of the ProDiag function block opens.
2. In the "General" tab, select the "Supervision settings" entry.
3. Set the display classes for the individual categories in the "Categories" area.

Result

If you have assigned display class "1" to the "Error" category, for example, and you activate this display class for an HMI device, only the supervision alarms that are assigned to the "Error" category are displayed on this HMI device.

24.7.5 Displaying the number of supervisions for a ProDiag FB (S7-1500)

Description

From the properties of the ProDiag function block you can read out how many ProDiag supervisions are assigned to this ProDiag FB.

Procedure

To read out the number of assigned ProDiag supervisions, follow these steps:

1. Open the properties of the ProDiag function block by either right-clicking on the ProDiag FB and selecting "Properties..." in the shortcut menu or double-click on the ProDiag FB. The properties dialog box of the ProDiag function block opens.
2. Click on the "Information" tab.

Result

The actual number of the possible ProDiag supervisions is shown.

24.8 Defining the ProDiag supervision settings (S7-1500)

24.8.1 Overview of the ProDiag supervision settings (S7-1500)

Description

In order to edit the ProDiag supervision settings, you must have created either a GRAPH function block or a supervision in your program beforehand.

You specify the settings that apply to all supervisions in the entire TIA Portal project in the ProDiag supervision settings. You can add and modify settings. You can find the ProDiag supervision settings in the project tree under "Common data".

See also

Defining general ProDiag supervision settings (Page 7869)

Specifying central ProDiag alarm texts (Page 7879)

24.8.2 Defining general ProDiag supervision settings (S7-1500)

24.8.2.1 Structure of the ProDiag supervision settings (S7-1500)

Structure of the ProDiag supervision settings

The following graphic shows the components of the ProDiag supervision settings as an example:

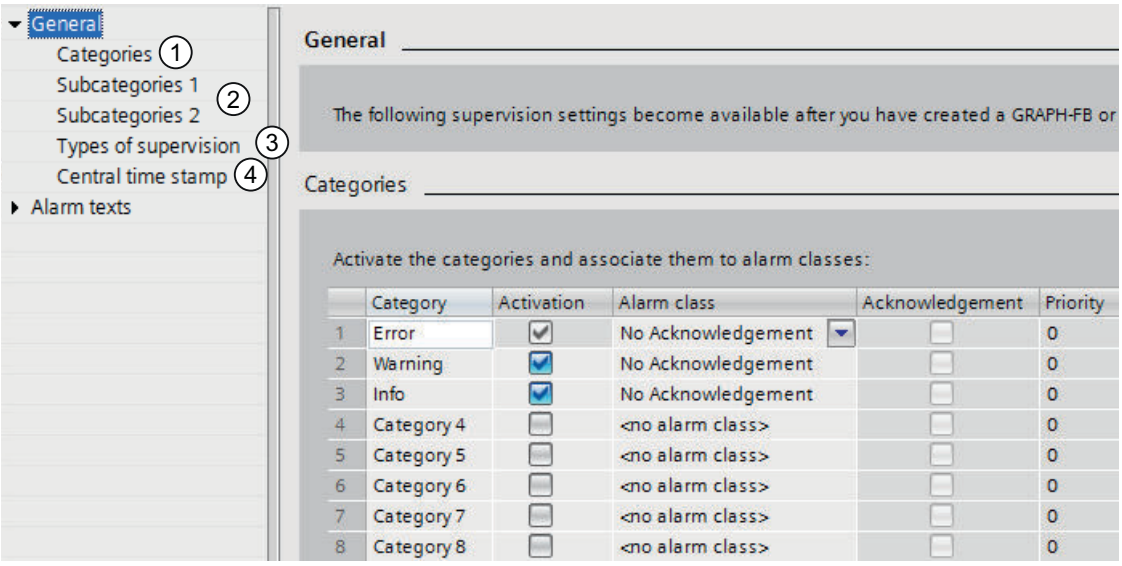

Numbering	Explanation	Additional information
1	8 categories can be defined.	You can use categories to divide faults into different priority levels such as errors, warnings, etc. You can find additional information on the categories here: Categories (Page 7870)
2	32 subcategories can be defined.	You can use the subcategories to specify the information within a supervision alarm in more detail. You can find additional information on the subcategories here: Subcategories (Page 7873)
3	There are 10 supervision types.	For each supervision type you can either use the pre-specified names or define your own ones. You can find additional information on the supervision types here: Types of supervision (Page 7876)
4	A central time stamp for all ProDiag function blocks in the project.	The time stamp tag is centrally assigned in the supervision settings for the entire project and can be used by every CPU in the project. You can find additional information on the central time stamp here: Using a central time stamp (Page 7845)

24.8.2.2 Categories (S7-1500)

Overview of the categories (S7-1500)

Description

You can use categories to divide faults into different priority levels. You can assign an alarm class to each category.

The categories can either be displayed as a component of a supervision alarm or they can be displayed on the HMI device as group error bit in the "ProDiag overview" object.

The figure below shows the "Categories" section in the ProDiag supervision settings editor with preset values:

Categories

Activate the categories and associate them to alarm classes:

	Category	Activation	Alarm class	Acknowledgement	Priority
1	Error	☑	No Acknowledgement	☐	0
2	Warning	☑	No Acknowledgement	☐	0
3	Info	☑	No Acknowledgement	☐	0
4	Category 4	☐	<no alarm class>	☐	0
5	Category 5	☐	<no alarm class>	☐	0
6	Category 6	☐	<no alarm class>	☐	0
7	Category 7	☐	<no alarm class>	☐	0
8	Category 8	☐	<no alarm class>	☐	0

Note

Additional information on the HMI object "ProDiag overview"

For additional information, refer to the information system of the TIA Portal. "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag > Visualizing supervisions > Configuring the ProDiag overview"

Alarm class

If you assign categories different alarm classes, you can create different areas in the alarm display on your HMI device based on the defined alarm classes.

If you additionally assign the alarm classes different priorities, you can use this to control the sequence of supervision alarms in "Alarm display " object. For each alarm class, you can also specify whether the supervision alarm is output with or without requiring acknowledgement.

See also

Creating alarm classes (Page 7528)

Defining categories (S7-1500)

You can change the settings of a category ("Category", "Activation" or "Alarm class") at any time, even if this category has already been assigned to a supervision. The changes are applied automatically to the supervision.

Requirement

You have opened the "Common data" folder in the project tree.

Procedure

To create a category, follow these steps:

1. Double-click the "Supervision settings" entry in the project tree.
   The ProDiag supervision settings are opened.
2. On the "General" tab, select the "Categories" section.
   The three categories "Error", "Warning" and "Info" are already available by default. In addition, there are five other placeholder categories. You can change and translate the names of the categories. You can find additional information on translating the names here: Displaying translations of the category names (Page 7872)
3. Activate or deactivate the category in the "Activation" column.

   Note

   "Error" category

   The pre-selected "Error" category cannot be deactivated. It remains activated at all times.

4. The system's own "No Acknowledgement" alarm class is set in the "Alarm class" column by default. Besides the system's own alarm class "No Acknowledgement" (= Does not require acknowledgement), the "Acknowledgement" (= Requires acknowledgement) alarm class or self-defined alarm classes are available. Change the default settings, if necessary.

   Note

   "Acknowledgement" alarm class

   If you select the "Acknowledgement" alarm class, a supervision alarm requiring acknowledgement appears on your HMI device. This acknowledgement has no effect on the progress of your program. If you want to influence the progress of your program after a fault has occurred, you must use the acknowledgement for a category with the acknowledge tag. You can set this acknowledgment in the block properties of the ProDiag function block.

   The "Acknowledgement" and "Priority" columns cannot be edited because these alarm class settings can only be changed in the alarm class editor. You can find additional information on creating alarm classes here: Creating alarm classes (Page 7528)
5. Save the project.

Displaying translations of the category names (S7-1500)

You can display the names of all categories in all project languages that have been activated in the setting options for the project languages and for which translations already exist. You can find additional information on activating project languages under AUTOHOTSPOT.

Requirement

You have opened the "Common data" folder in the project tree.

Procedure

To display the name of a category in all project languages, follow these steps:

1. Double-click the "Supervision settings" entry in the project tree.
   The ProDiag supervision settings are opened.
2. On the "General" tab, select the "Categories" section.
3. Select the line with the category whose translations you want to display.
   If you want to display the translations of multiple categories simultaneously, select the corresponding lines.
4. Open the "Properties" tab in the Inspector window.
   If the names of the categories appear only in the current user interface language, either you have not yet selected any project languages or translations are not yet available. You can enter the translations directly in the table on the "Texts" tab or you can enter the translations for the individual project languages in the global "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts".

You can find additional information on translation of texts under AUTOHOTSPOT.

Result

You see the selected lines in the desired project languages and, if available, the translations of the category names on the lower-level "Texts" tab.

Texts

German (Germany)	English (United States)	Chinese (Peopl..	French (France)	Italian (Italy)	Spanish (Spain)
Fehler	Error	错误	Erreur	Errore	Error
Warnung	Warning	警告	Avertissement	Avviso	Advertencia
Information	Info	Info	Information	Informazione	Información
Kategorie 4	Category 4	类别 4	Catégorie 4	Categoria 4	Categoría 4
Kategorie 5	Category 5	类别 5	Catégorie 5	Categoria 5	Categoría 5
Kategorie 6	Category 6	类别 6	Catégorie 6	Categoria 6	Categoría 6
Kategorie 7	Category 7	类别 7	Catégorie 7	Categoria 7	Categoría 7
Kategorie 8	Category 8	类别 8	Catégorie 8	Categoria 8	Categoría 8

24.8.2.3 Subcategories (S7-1500)

Overview of the subcategories (S7-1500)

Description

In addition to the 16 subcategories of "Subcategory 1", you can also activate 16 subcategories of "Subcategory 2". The categories and subcategories are independent of one another and can be freely combined with one another within a supervision.

With the help of subcategories, you have the option of specifying the information within a supervision alarm in more detail. For example, if you have already created the "Error", "Warning" and "Info" categories, you can expand these with subcategories "Technical" or "Organizational". In this way, you receive detailed information on a fault that has occurred.

The subcategories can either be displayed as a component of a supervision alarm or they can be displayed on the HMI device as group error bit in the "ProDiag overview" object.

The figure below shows the "Subcategories 1" section in the ProDiag supervision settings:

Define and activate subcategories and subcategory descriptions:

	Subcategory 1	Activation	Description
1	Subcategory 1	☑	
2	Subcategory 2	☑	
3	Subcategory 3	☐	

Note

Additional information on the HMI object "ProDiag overview"

For additional information, refer to the information system of the TIA Portal. "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag > Visualizing supervisions > Configuring the ProDiag overview"

Defining subcategories (S7-1500)

You can change the settings of a subcategory ("Subcategory x", "Activation" or "Description") even if this subcategory has already been assigned to a supervision. The changes are applied automatically to the supervision.

Requirement

You have opened the "Common data" folder in the project tree.

Procedure

To activate a subcategory, follow these steps:

1. Double-click the "Supervision settings" entry in the project tree.
   The ProDiag supervision settings are opened.
2. On the "General" tab, select the "Subcategories 1" or "Subcategories 2" section.
3. Enter the desired subcategory names.
   The names of the subcategories can be translated. You can find additional information on translating the names here: Displaying and editing translations of the subcategory names and description texts (Page 7875)
4. Activate or deactivate the desired subcategories in the "Activation" column.
5. Enter a text in the "Description" column, if necessary.
   You can use the "Description" column to forward information to the programmer creating the STEP 7 project. This information is available only within the TIA Portal and is not forwarded to Runtime Professional.
   In addition to the names of the subcategories, you also see the description texts in the lower-level "Texts" tab of the "Properties" Inspector window.
6. Save the project.

Result

A new subcategory 1 is created, which you can combine with a category when creating a supervision.

Define and activate subcategories and subcategory descriptions:

	Subcategory 1	Activation	Description
1	Technical	☑	Technical error
2	Organizational	☑	Organizational error

Displaying and editing translations of the subcategory names and description texts (S7-1500)

You can display the names of all categories in all project languages that have been activated in the setting options for the project languages and for which translations already exist. You can find additional information on activating project languages under AUTOHOTSPOT.

Requirement

You have opened the "Common data" folder in the project tree.

Procedure

To display the name of a subcategory in all project languages, follow these steps:

1. Double-click the "Supervision settings" entry in the project tree.
   The ProDiag supervision settings are opened.
2. On the "General" tab, select the "Subcategories x" section.
3. Select the line with the subcategory whose translations you want to display.
   If you want to display the translations of multiple subcategories simultaneously, select the corresponding lines.
4. Open the "Properties" tab in the Inspector window.
   If the names of the subcategories appear only in the current user interface language, either you have not yet selected any project languages or translations are not yet available. You can enter the translations directly in the table on the "Texts" tab or you can enter the translations for the individual project languages in the global "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts".

You can find additional information on translation of texts under AUTOHOTSPOT.

Result

You see the selected lines in the desired project languages and, if available, the translations of the subcategory names and their description texts in the lower-level "Texts" tab.

Texts

German (Germany)	English (United States)	French (France)	Italian (Italy)
Technisch	Technical	Technique	Tecnico
Technischer Fehler	Technical error	Erreur technique	Errore tecnico
Organisatorisch	Organizational	Organisation	Organizzativo
Organisatorischer Fehler	Organizational error	Erreur d'organisation	Errore organizzativo

24.8.2.4 Types of supervision (S7-1500)

Overview of the types of supervision (S7-1500)

Description

There are ten types of supervision with preset names. The names are write-protected and the translations are made available in the selected user interface languages by the system. In addition, you can define your own display name of the type of supervision in the alarm text for each type of supervision and translate it into all user interface languages. You can define default settings for each type of supervision within your project and thereby define the signal state for which an operand or condition is to be supervised. The defaults apply to newly created supervisions or if you change the type of supervision. In addition, you can preset a delay time for each type of supervision.

The figure below shows the "Types of supervision" section in the ProDiag supervision settings editor with preset values:

Types of supervision

Specify the default settings for each type of supervision within your project. These settings

If you select the check box, the triggering status is TRUE.

You will be able to adapt these default values for every supervision. Your changes regardi

	Type of supervision	Trigger	C1 Trigger	C2 Trigger	C3 Trigger	Delay time
1	Operand	☐	☑	☑	☑	T#0ms
2	Interlock	☐	☑	☑	☑	T#0ms
3	Reaction	☐	☑	☑	☑	T#5s
4	Action	☑	☑	☑	☑	T#200ms
5	Position	☐	☐	☑	☑	T#0ms
6	Error message	☑	☑	☑	☑	T#0ms
7	Text message	☑	☑	☑	☑	T#0ms
8	GRAPH-Interlock	☐	☐	☐	☐	T#0ms
9	GRAPH-Supervision	☐	☐	☐	☐	T#0ms
10	GRAPH-Warning	☐	☐	☐	☐	T#0ms

Activation

When you create a GRAPH function block or a supervision, all options, except for the GRAPH settings, are automatically activated by the operating system by default. You can change the default settings for the triggers and the delay time at any time by activating or deactivating the relevant columns. If you select the check box, the state of the operand is supervised for the TRUE signal state. If you clear the check box, the operand status is supervised for the FALSE signal state.

Displaying translations of the type of supervision names (S7-1500)

You can display the names of all types of supervision in all project languages that have been activated in the setting options for the project languages and for which translations already exist. You can find additional information on activating project languages under "Editing project data > Working with multi-language projects > Select project languages".

Requirement

You have opened the "Common data" folder in the project tree.

Procedure

To display the name of a type of supervision in all project languages, follow these steps:

1. Double-click the "Supervision settings" entry in the project tree.
   The ProDiag supervision settings are opened.
2. In the "General" tab, select the "Types of supervision" section.
3. Select the line with the type of supervision whose translations you want to display.
   If you want to display the translations of multiple types of supervision simultaneously, select the corresponding lines.
4. Open the "Properties" tab in the Inspector window.
   If you have not yet translated the names of the types of supervision in all selected project languages, you will see the texts in the current user interface language by default. If the translations are not yet available, you can enter them directly in the table on the "Texts" tab or you can enter the translations for the individual project languages in the global "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts".

You can find additional information on translating texts under "Editing project data > Working with multi-language projects > Project text basics".

Result

You see the desired project languages and, if available, the translations of the type of supervision names in the lower-level "Texts" tab.

Texts

German (Germany)	English (United States)	French (France)	Italian (Italy)	Spanish (Spain)
Operand	Operand	Opérande	Operando	Operando
Verriegelung	Interlock	Verrouillage	Interblocco	Enclavamiento
Reaktion	Reaction	Réaction	Reazione	Reacción
Aktion	Action	Action	Azione	Acción
Position	Position	Position	Posizione	Posición
Fehlermeldung	Error message	Message d'erreur	Messaggio di errore	Mensaje de error
Textmeldung	Text message	Message texte	Messaggio di testo	Texto de aviso
GRAPH-Interlock	GRAPH-Interlock	GRAPH-Interlock	GRAPH-Interlock	GRAPH-Interlock
GRAPH-Supervision	GRAPH-Supervision	GRAPH-Supervision	GRAPH-Supervision	GRAPH-Supervision
GRAPH-Warnung	GRAPH-Warning	GRAPH-Avertissement	GRAPH-Avviso	GRAPH-Advertencia

24.8.3 Specifying central ProDiag alarm texts (S7-1500)

24.8.3.1 Structure of the alarm text editor (S7-1500)

Structure of the alarm text editor

The following graphic shows an example of the components in the alarm text editor:

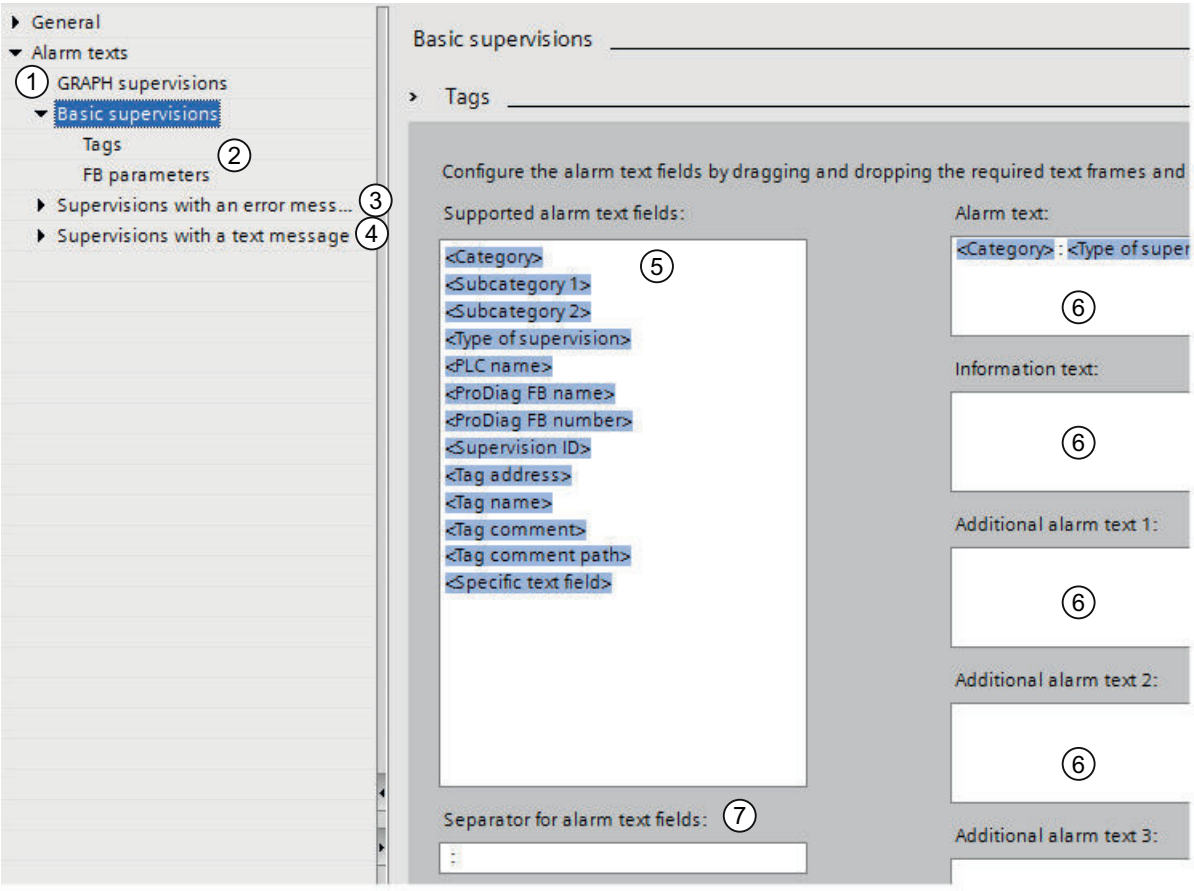

Number-ing	Explanation	Additional information
1	Creating alarm text structures for GRAPH supervisions (only in connection with the GRAPH programming language)	You can find additional information on the GRAPH supervision messages here: Structure of a GRAPH supervision alarm (Page 7881)
2	Creating alarm text structures for the basic supervisions: • Operand • Interlock • Action • Reaction • Position You can create a separate alarm text for global supervisions (tag) and local supervisions (function block-parameter) for each basic supervision.	You can find additional information on the global supervision messages here: Structure of a global supervision alarm (tag) (Page 7882) You can find additional information on the local supervision messages here: Structure of a local supervision alarm (parameter) (Page 7884)
3	Creating alarm text structures for Supervisions with an error message. You can create a separate alarm text structure for all global supervisions (tags) and local supervisions (FB parameters) for each supervision with an error message.	
4	Creating alarm text structures for Supervisions with a text message. You can create a separate alarm text structure for all global supervisions (tags) and local supervisions (FB parameters) for each supervision with a text message.	
5	Listing of all the supported alarm text fields that you can drag-and-drop into the individual text frame.	
6	Various text frames that you can fill with alarm text fields in order to configure the desired supervision alarms.	You can define up to 11 text frames (alarm text, information text and nine additional alarm texts). The nine additional alarm texts can only be displayed in WinCC Runtime Professional.
7	Separator for alarm text fields	A colon is used by default to separate the alarm text fields within a text frame. You can change or delete the separator at any time if you do not want to use any separator. The previously used separator, e.g. a colon, is then automatically replaced in all text frames by the new separator, e.g. a semicolon.

Description

A preset text field structure which you can change at any time is provided for the "Alarm text" text frame.

From the "Supported alarm text fields" area, you can drag or copy available alarm text fields into one of the text frames. Each alarm text field can be used only once within a text frame. The alarm text fields that are available depend on the area you clicked on below "Alarm texts" in the tree. You can change the order of the alarm text fields within a text frame using drag-and-drop.

When you use the "<Specific text field>" alarm text box, you can have a specific text displayed for this supervision in the supervision alarm. For the alarm text field to display the requested information, you must configure specific information for this supervision in the settings of the individual supervisions in the lower-level "Specific text field" tab.

See also

Creating a specific text field (Page 7898)

24.8.3.2 Structure of a GRAPH supervision alarm (S7-1500)

Description

This alarm text structure for a GRAPH supervision applies to all three GRAPH types of supervision:

- GRAPH-Interlock
- GRAPH-Supervision
- GRAPH-Warning (step-time supervision)

The table below shows you all components that can be included in a GRAPH supervision. You can, however, reduce the number of components at any time:

Components of a GRAPH supervision alarm	Description	Specified by default	Translation-related
<Category>	Name of the category	x	Yes
<Subcategory 1>	Name of the subcategory 1	-	
<Subcategory 2>	Name of the subcategory 2	-	
<Type of supervision>	Name of the type of supervision	x	

Components of a GRAPH supervision alarm	Description	Specified by default	Translation-related
<PLC name>	Name of the PLC	x	No
<FB name>	Name of the function block	x	
<FB number>	Number of the function block	-	
<Step name>	Step name	x	
<Step number>	Step number	x	
<Instance DB name>	Name of the instance data block	-	
<Instance DB number>	Number of the instance data block	-	
<Step specific field>	Specific interlock or supervision information that you have defined at each step.	x	

You can change the number of supervision alarms at any time by adding new components or removing those already included. All components can be used optionally.

See also

Creating alarm texts (Page 7885)

Editing alarms (Page 7485)

Enabling and disabling the alarm view (Page 7485)

24.8.3.3 Structure of a global supervision alarm (tag) (S7-1500)

Description

You can create global supervision alarms for supervisions of a tag that was defined in a tag table or in a data block.

The table below shows you all components that can be included in a global supervision alarm. You can, however, reduce the number of components at any time:

Components of a global supervision alarm	Description	Specified by default	Translation-related
<Category>	Name of the category	x	Yes
<Subcategory 1>	Name of the subcategory 1	-	
<Subcategory 2>	Name of the subcategory 2	-	No
<Type of supervision>	Name of the type of supervision	x	Yes

Components of a global supervision alarm	Description	Specified by default	Translation-related
<PLC name>	Name of the PLC	-	No
<ProDiag FB name>	Name of the ProDiag function block	x	
<ProDiag FB number>	Number of the ProDiag function block	-	
<Supervision ID>	ID of the supervision	x	
<Tag address>	• Tag address from the tag table • An address is not output for Boolean elements from a data block.	x	
<Tag name>	Name of the supervised tag If this is a structure element (PLC data type, ARRAY or STRUCT), all higher-level structure levels are displayed, e.g. "Motors".Motor_1.Element_BOOL	x	
<Tag comment>	• Comment of the tag from the tag table • Comment of the element from the data block	x	Yes
<Tag comment path>	Comments of the tags of all levels of a structured tag, separated by dots	-	
<Specific text field>	Specific information for a supervision (text with 3 tags) • <Associated value SD_5> • <Associated value SD_6> • <Associated value SD_7>	-	

You can change the number of supervision alarms at any time by adding new components or removing those already included. All components can be used optionally.

See also

Creating alarm texts (Page 7885)

24.8.3.4 Structure of a local supervision alarm (parameter) (S7-1500)

Description

You can create local supervision alarms for parameter supervisions that are defined in the block interface of a function block.

Note

Structured tags

No Boolean elements of a structured tag (PLC data type, ARRAY or STRUCT) can be monitored in the block interface.

The table below shows you all components that can be included in a local supervision alarm. You can, however, reduce the number of components at any time:

Components of a local supervision alarm	Description	Specified by default	Translation-related
<Category>	Name of the category	x	Yes
<Subcategory 1>	Name of the subcategory 1	-	
<Subcategory 2>	Name of the subcategory 2	-	No
<Type of supervision>	Name of the type of supervision	x	Yes
<PLC name>	Name of the CPU	-	No
<ProDiag FB name>	Name of the ProDiag function block	x	
<ProDiag FB number>	Number of the ProDiag function block	-	
<Supervision ID>	ID of the supervision	x	
<FB name>	Name of the function block	-	
<Instance path>	Name and path of the instance data block	-	
<Instance name>	Name of the instance data block (for example Block_9_DB)	x	
<Instance number>	Number of the instance data block	-	
<Instance comment>	• At a single instance: The title of the instance DB is output. • At a multiple instance: The comment of the multiple instance from the block interface is output.	-	Yes

Components of a local supervision alarm	Description	Specified by default	Translation-related
<Parameter name>	Name of the supervised parameter	x	No
<Parameter comment>	Parameter comment	-	Yes
<Tag address>	• Tag address from the tag table • An address is not output for Boolean elements from a data block.	x	No
<Tag name>	In this case, the parameter (<Parameter name>) is supervised and the tag (<Tag name>) is the preceding tag, if one exists.	x	
<Tag comment>	• Comment of the tag from the tag table • Comment of the element from the data block	x	Yes
<Tag comment path>	Comments of the tags of all levels of a structured tag, separated by dots	-	
<Specific text field>	Specific information for a supervision (text with 3 tags) • <Associated value SD_5> • <Associated value SD_6> • <Associated value SD_7>	-	

You can change the number of supervision alarms at any time by adding new components or removing those already included. All components can be used optionally.

See also

Creating alarm texts (Page 7885)

24.8.3.5 Creating alarm texts (S7-1500)

Requirement

You have opened the "Common data" folder in the project tree.

Procedure

To create an alarm text, follow these steps:

1. Double-click the "Supervision settings" entry in the project tree.
   The ProDiag supervision settings are opened.
2. In the "Alarm texts" tab, select the area for which you want to create an alarm text, e.g. Basic supervisions > Tags.
3. Use drag-and-drop to move the desired alarm text fields from the "Supported alarm text fields" field to the "Alarm text" text frame.
   You can change the order of the individual text fields at any time.
4. If you want to delete a text field again, right-click the corresponding text field and select "Delete" from the shortcut menu.
5. Select the desired delimiter for the text fields.

Result

The text fields serve as placeholders and are filled with current process values as soon as a supervision alarm is triggered.

24.8.3.6 Exporting ProDiag and GRAPH texts (S7-1500)

You can filter project texts from ProDiag and GRAPH and export them for translation. The texts are exported to an Office Open XML file with the extension ".xlsx". This can be edited in Microsoft Excel or a number of other spreadsheet programs.

You define the export of ProDiag and GRAPH texts using the following filter options:

Option	Contents of the export
ProDiag / GRAPH: Alarm texts	• User-defined text lists for supervisions • Display names for the steps and transitions in GRAPH • Comments and comment paths of the supervised tags • Comments of the supervised elements in the block interface • Title of the supervised instance data blocks and instance comments when a multi-instance is involved • Comments and comment paths of the tags that are series-connected with supervised elements of the block interface (if only one operand is series-connected) • Step-specific interlock and supervision alarm texts for GRAPH
ProDiag / GRAPH: Texts for HMI display	• Tag comments in transition and interlock networks • Tag comments in permanent networks supervised with ProDiag • Tag comments in networks supervised with ProDiag • Titles of networks supervised with ProDiag (including permanent networks)
ProDiag / GRAPH: All texts	• ProDiag / GRAPH alarm texts • ProDiag / GRAPH texts for HMI display

Note

These filter options are not available for CPU S7-300/400.

You can find additional information in the section "Exporting project texts".

24.8.4 Exporting ProDiag supervision settings (S7-1500)

You can export the entire ProDiag supervision settings to a .dat file and import them into a new project.

Requirement

You have defined the ProDiag supervision settings for a project.

Procedure

To export the ProDiag Monitor settings, follow these steps:

1. Open the "Common data" folder in the project navigation.
2. Double-click the "Supervision settings" entry.
3. Click "Export supervision settings". If the option is not selected, you have not yet created a ProDiag supervision or a GRAPH function block.
   The "Save as" dialog opens.
4. Select a storage location and assign a file name.
5. Click "Save".

Result

A .dat file is created at the selected memory location. This file cannot be edited.

24.8.5 Importing ProDiag supervision settings (S7-1500)

Requirement

You have exported the ProDiag supervision settings of a project to a .dat file.

You can import the ProDiag supervision settings into your project even if you have not yet created a ProDiag supervision function or a GRAPH function block.

Note

Upgrading project

If you have upgraded your project to TIA Portal version 15 and the "Repair" button is visible in the subcategories area of the ProDiag supervision settings, you must first press this button before you can import the ProDiag supervision settings from another project.

Procedure

To import the ProDiag supervision settings into a new project, follow these steps:

1. Open the "Supervision settings" in the "Common data" folder of the project tree.
2. Click "Import supervision settings".
   If "Import supervision settings" is grayed out and the "Repair" button is visible in the subcategories area, you must first press this button. This repairs the subcategories. You can then import the remaining ProDiag supervision settings.
3. The "Open" dialog opens.
   Select the path where the file to be imported is stored.
4. Select the required .dat file.
5. Confirm the dialog with "Open".

Result

All ProDiag supervision settings entered in the .dat file are transferred to the new project.

Note

Importing non-existent ProDiag supervision settings

For example, if you import alarm classes that do not yet exist in the new project, the system displays an error message and the alarm classes defined by default are assigned to the categories.

If you create the alarm classes after the import, you must then assign them manually to the desired categories.

24.9 Creating ProDiag supervisions (S7-1500)

24.9.1 Overview of the supervisions (S7-1500)

Description

You have the option to create global and local supervisions. Moreover, you can create more than one supervision for each operand.

For each supervision you create, a new entry is created in the "Supervisions" tab of the "Properties" Inspector window. You can define the settings of each individual supervision in the "Supervisions" tab. If you create multiple supervisions for an operand, these are then numbered in ascending order by adding the suffix "ID_<consecutive supervision ID> (assigned ProDiag FB)".

Note

Creating supervisions in an F-instance data block

You can create global ProDiag supervisions for the Boolean parameters in the block interface of a fails-safe instance data block.

As soon as you create the first global supervision within your project or instantiate a local supervision, the "Default_SupervisionFB" and "Default_SupervisionDB" ProDiag blocks are automatically created in the "Program blocks" folder of the project tree.

You can see at a glance how many ProDiag supervisions are assigned to a ProDiag function block. You can find additional information under: Displaying the number of supervisions for a ProDiag FB (Page 7868)

Global supervisions

Supervision of a Boolean variable within a tag table, a global data block or an ARRAY data block.

The figure below shows you which data types you can supervise:

GlobalDB

	Name	Data type	Start value	Retain	Acce...	Read..	Visi...	Set...	Supervision
1	Static								
2	Start_Motor	Bool	false						
3	Flash_Light	Array[0..1] of Bool							
4	Flash_Light[0]	Bool	false						
5	Flash_Light[1]	Bool	false						
6	Stop_Motor	Struct							
7	Input_1	Bool	false						

Local supervisions

Supervision of a Boolean parameter of a function block

The figure below shows you which parameters you can supervise:

LAD_FB_1500

		Name	Data type	Default value	Ret...	Acc...	Read-...	Visi...	S...	Supervision
1		Input								
2		Start_Motor	Bool	false	No...	☑	☐	☑	☐	
3		Stop_Motor	Bool	false	No...	☑	☐	☑	☐	
4		Output								
5		Flash_Light	Bool	false	No...	☑	☐	☑	☐	
6		InOut								
7		Static								
8		Signal_Storage	Bool	false	No...	☑	☐	☑	☐	
9		Temp								
10		Constant								

"Supervisions" tab in the "Properties" Inspector window

In this tab, you can specify the settings for each created supervision of an operand:

Group	Property	Description
General	Type of supervision	The following types of supervision are available: • Operand, e.g. for supervision of static signals • Interlock, e.g. for supervision of control signals • Action, e.g. for supervising whether the start position has been left • Reaction, e.g. for supervising whether the end position has been reached • Position, e.g. for supervising whether the end position has been left without permission • Text message (displays a simple text message) • Error message (displays an error number and the associated text) Some of the fields in the properties are mandatory fields and must be filled out, depending on which type of supervision you select. You can find out which fields are mandatory here: Types of supervision (Page 7806)
	Supervised tag / parameter	Indicates which operand is being supervised. The signal states TRUE and FALSE of the supervised operand indicate the signal state for which the operand is to be monitored.
	Delay time / reaction time / start-up time	A different time is displayed depending on which type of supervision you select: • Delay time: Operand, Interlock, Position, Text message, Error message • Reaction time Reaction • Start-up time: Action
	Conditions	You can only use Boolean tags and slice access operations of the BIT data type as conditions. The signal states TRUE and FALSE of the individual conditions indicate the signal state for which the operand is to be monitored.
	Error flag (only for global tags)	The error flag is written to the data block with the same name of the ProDiag function block.
	ProDiag FB (for global tags only)	If you do not assign a user-defined ProDiag function block to a supervision, it is automatically assigned to the ProDiag function block "Default_SupervisionFB". This ProDiag function block contains all global supervisions and already instantiated local supervisions which were assigned to this block. However, you can also create your own ProDiag function blocks in order to group supervisions into technological units. For instructions on how to create ProDiag function blocks, refer to: Creating ProDiag function blocks (Page 7838)
	Category	All 8 categories are displayed in the "Category" drop-down list.
	Subcategory 1 and 2	In order to have possible selections in the drop-down lists of the "Subcategory" fields, you must define them in the ProDiag supervision settings. You can find an example here: Defining the ProDiag supervision settings (Page 7869)
	Alarm text	The alarm text cannot be edited because it is also defined in the ProDiag supervision settings.
Specific text field	-	Here you can create a specific text field for each supervision with the help of associated values. You can find an example here: Creating a specific text field (Page 7900)

See also

Displaying supervisions of all called function blocks (Page 7921)
Display supervisions of all global tags (Page 7919)
Creating global supervision (Page 7893)
Possible uses of supervisions (Page 7892)
Multiple supervision alarms of a ProDiag FB within one cycle time (Page 7846)
Creating a local supervision (Page 7895)
Creating a specific text field (Page 7898)

24.9.2 Possible uses of supervisions (S7-1500)

Description

You can create supervisions for the following memory areas:

Memory area/Section of the block interface	Storage location	Supervision
Inputs	Tag table	Global supervisions (tag)
Outputs		
Bit memory		
Elements of a data block of data type BOOL	Global data block or an ARRAY data block of data type BOOL	
Input	Block interface	Local supervisions (parameter)
Output		
Static		

Note

Structured tags

Boolean elements of a structured tag (PLC data type, ARRAY or STRUCT) can be monitored in the block interface of a user block if this has been declared in the "Static" section.

A variety of possible uses of supervisions are available for each type of supervision:

	Global supervisions			Local supervisions		
Type of supervision	Inputs	Outputs	Bit memory/ elements of a data block of data type BOOL	Input	Output	Static
Operand	x	x	x	x	x	x
Interlock	-	x	x	x	-	-
Action	x	-	x	x	-	-

	Global supervisions			Local supervisions		
Type of supervision	Inputs	Outputs	Bit memory/ elements of a data block of data type BOOL	Input	Output	Static
Reaction	x	-	x	x	-	-
Position	x	-	x	x	-	-
Error message	x	x	x	x	x	x
Text message	x	x	x	x	x	x
x: The supervision can be used. -: The supervision cannot be used.						

24.9.3 Creating global supervision (S7-1500)

Requirement

You have created at least one Boolean tag either in a tag table or in a data block.

		Name	Data type	Address	Retain	Visibl...	Acces...	Read-...	Supervision
1		Conv_Actor_Forward	Bool	%I0.0		☑	☑		
2		Conv_Actor_Backward	Bool	%I0.1		☑	☑		
3		Conv_Forward_Trigger	Bool	%I0.2		☑	☑		
4		Conv_Backward_Trigger	Bool	%I0.3		☑	☑		
5		Conv_Switch_Left	Bool	%I0.4		☑	☑		
6		Conv_Switch_Right	Bool	%I0.5		☑	☑		
7		Conv_Auto / Hand	Bool	%I0.6		☑	☑		
8		Conv_Push_Button_Forward	Bool	%I0.7		☑	☑		
9		Conv_Push_Button_Backward	Bool	%I1.0		☑	☑		

Procedure

Proceed as follows to create global supervision:

1. Select one or more tags with the right mouse button and select "Add new supervision" from the shortcut menu. The shortcut menu entry is only available when all selected tags (Boolean tags) can be supervised.
   The "Supervisions" tab opens in the "Properties" Inspector window and a supervision for the tag is created. The supervisions are numbered in ascending order (Supervision_ID_<consecutive supervision ID>(assigned ProDiag FB)).
2. The "Operand" type of supervision is selected by default. Select a different type of supervision, if necessary.
   The type of supervision you have selected determines which settings must be filled out.
   You can find additional information on the types of supervision here: Types of supervision (Page 7806)

   Note

   Settings are overwritten

   If you change the type of supervision of a previously created supervision, your settings will be overwritten by the preset values of the new type of supervision.

3. Set a delay time, if applicable.
4. Select conditions under which a supervision alarm is to be triggered, if necessary.
5. If you do not wish to use the default ProDiag FB "Default_SupervisionFB", you can either select an existing ProDiag FB or create a new ProDiag FB.
6. Select a category and, if necessary, subcategories.
7. If you are not using the supervised tag in your program code yet, add it now.

Or:

1. Click the desired tag.
2. Double-click the entry "Add new supervision" in the "Supervisions" in the "Properties" Inspector window.
   A new supervision is created. The supervisions are numbered in ascending order (Supervision_ID_<consecutive supervision ID>(assigned ProDiag FB)).
3. From this point, the procedure described above from item 2 applies.

Result

The supervision of a global tag has been created and the tag is marked with the ProDiag symbol in the variable table. You can use the tags in all program blocks within your program.

	Name	Data type	Address	Retain	Visibl...	Acces...	Read-...	Supervision
1	Conv_Actor_Forward	Bool	%I0.0		☑	☑		
2	Conv_Actor_Backward	Bool	%I0.1		☑	☑		
3	Conv_Forward_Trigger	Bool	%I0.2		☑	☑		
4	Conv_Backward_Trigger	Bool	%I0.3		☑	☑		
5	Conv_Switch_Left	Bool	%I0.4		☑	☑		
6	Conv_Switch_Right	Bool	%I0.5		☑	☑		
7	Conv_Auto / Hand	Bool	%I0.6		☑	☑		
8	Conv_Push_Button_Forward	Bool	%I0.7		☑	☑		
9	Conv_Push_Button_Backward	Bool	%I1.0		☑	☑		

Indicates that the tag is being supervised.

Note

Copying tags

If you copy a global tag for which you have already created supervisions from one CPU to another CPU or from one project to another project, the supervisions are not included in the copy operation.

24.9.4 Creating a local supervision (S7-1500)

Requirement

You have created either a Boolean parameter in the block interface of a function block in one of the three sections:

LAD_FB_1500

	Name	Data type	Default value	Ret...	Acc...	Read-...	Visi...	S...	Supervision
1	Input								
2	Start_Motor	Bool	false	No...	☑		☑		
3	Stop_Motor	Bool	false	No...	☑		☑		
4	Output								
5	Flash_Light	Bool	false	No...	☑		☑		
6	InOut								
7	Static								
8	Signal_Storage	Bool	false	No...	☑		☑		
9	Temp								
10	Constant								

Note

Structured tags

No Boolean elements of a structured tag (PLC data type, ARRAY or STRUCT) can be monitored in the block interface.

Note

Properties of a local supervision

All properties of a local supervision (tags for the conditions, delay time or the specific text field) must also be declared locally in the block interface.

Procedure

Proceed as follows to create a local supervision:

1. Right-click the parameter and select "Add new supervision" from the shortcut menu.
   The "Supervisions" tab opens in the "Properties" Inspector window and a supervision for the parameter is created. The supervisions are numbered in ascending order (Supervision_ID_<consecutive supervision ID>(assigned ProDiag FB)).
2. The "Operand" type of supervision is selected by default. Select a different type of supervision, if necessary.
   The type of supervision you have selected determines which settings must be filled out.
   You can find additional information on the types of supervision here: Types of supervision (Page 7806)

   Note

   Settings are overwritten

   If you change the type of supervision of a previously created supervision, your settings will be overwritten by the preset values of the new type of supervision.

3. Set a delay time, if applicable.
4. Select conditions under which a supervision alarm is to be triggered, if necessary.
5. Select a category and, if necessary, subcategories.
6. Use the supervised parameter in the program code.
7. Call the function block in your program, and select an instance data block in the "Call options" dialog in the "Single instance" area.
   The supervisions are not activated in your project until you have called the function block and have created an instance data block for the function block.

Or:

1. Click the desired parameter.
2. Double-click the entry "Add new supervision" in the "Supervisions" in the "Properties" Inspector window.
   A new supervision is created. The supervisions are numbered in ascending order (Supervision_ID_<consecutive supervision ID>(assigned ProDiag FB)).
3. From this point, the procedure described above from item 2 applies.

Result

The supervisions of the parameters have been created and identified in the block interface with the ProDiag symbol.

LAD_FB_1500

		Name	Data type	Default value	Ret...	Acc...	Read-...	Visi...	S...	Supervision
1		Input								
2		Start_Motor	Bool	false	No...	☑	☐	☑	☐	
3		Stop_Motor	Bool	false	No...	☑	☐	☑	☐	
4		Output								
5		Flash_Light	Bool	false	No...	☑	☐	☑	☐	
6		InOut								
7		Static								
8		Signal_Storage	Bool	false	No...	☑	☐	☑	☐	
9		Temp								
10		Constant								

Indicates that the parameter is being supervised.

When you right-click the function block in the project tree, the properties dialog of the function block is displayed. You can see all supervisions you have created for this function block in the "FB supervision definitions" tab.

You can find additional information on the overview of the supervisions in a function block here: Displaying supervisions of a function block (Page 7925)

Note

Code Viewer

When supervising input parameters of a function block, you can display the series connection of the input parameters in the Code Viewer.

When supervising the output and/or static parameters of a function block, you can display the complete call of the function block in the Code Viewer.

24.9.5 Creating a specific text field (S7-1500)

24.9.5.1 Use of the associated values

The associated values SD_4, SD_5 and SD_6 can be referenced as follows:

@<Number of the associated value><Format>@

The character "@" initiates the use of an associated value and also closes it.

Number of the associated value

The number specifies which associated value or which tag is to be read:

Number	Explanation
4	Associated value SD_4 (tag 1)
5	Associated value SD_5 (tag 2)
6	Associated value SD_6 (tag 3)

Maximum permitted size of the associated values

The reason for the total size of the associated values to exceed the maximum permitted size is usually that one or more of the associated values are of the data type STRING or WSTRING and an actual length was not specified. If no actual length is specified for WSTRING, for example, the maximum permitted length is used, and the maximum permitted size is already exceeded with one associated value.

Always use brackets [] for the data types STRING and WSTRING to define the actual length.

Format

Determine the output format for the associated value on the display device. The format is preceded by the "%" sign. The following fixed formats apply to associated values:

Format	Description
%[i]X	Hexadecimal number with i digits
%[i]u	Decimal number without sign with i digits
%[i]d	Decimal number with sign with i digits
%[i]b	Binary number with i digits
%[i][.y]f	Floating-point number with sign with y digits after the decimal point and total number of digits i
%[i]s	String (ANSI string) with i digits Characters are printed up to the first 0 Byte (00Hex).
%t#<Name of text library>	Access to text library

If the number of digits [i] is too small, the value is nevertheless output in full.

If the number of digits [i] is too large, an appropriate number of fill characters is output before the value.

If the number of digits [i] is too large and if a sign is output (for decimal/floating-point numbers), the number of fill characters is reduced by one. The same applies to periods or commas within floating-point numbers.

Note

Please note that you can optionally enter "[i]" (without the square brackets).

Examples

The table below shows some examples:

Tag value	Associated value	Explanation	Output
123*	@4%6d@	The value of associated value SD_4 is shown as a decimal number with sign and a total number of 6 digits.	000123
1234.567	@4%8.3f@	The value of associated value SD_4 is shown as a floating-point number with sign with 3 digits after the decimal point and a total number of 8 digits. A fill character is not added because the decimal point represents the eighth character.	1234.567
5.4*	@5%6f@	The value of associated value SD_5 is shown as an integer with a total number of 6 digits. The number of places after the decimal point has not been defined.	0005.4
5.4	@5%2f@	The value of associated value SD_5 is shown as an integer with a total number of 3 digits. The value is not shortened.	5.4
Contents of the text library <TextLib1>	@6%t#TextLib1@	The associated value SD_6 of the data type WORD is the index which references the text library used "TextLib1".	90°

The type of the fill character (for example, zeros or blanks) depends on the output system.

24.9.5.2 Creating a specific text field (S7-1500)

With the help of specific text fields, you can display additional information for each supervision in a supervision alarm with the help of associated values. This information can be combined freely in the specific text field and consists of the following elements:

- Free text (for example, temperature, manual mode, automatic mode)
- Value of the three different tags (associated values SD_4 - SD_6) that are read out during runtime
- Text list entry

Note

Use the @ character in the specific text field

Please use the @ character only for declaring the associated values, for example @4%6d@. Also, if you use the @ character in the text-specific text field, you may not be able to resolve the associated values contained in the free text of the supervision alarm.

Requirement

You have created either a global or a local supervision and opened the "Supervisions" tab in the "Properties" Inspector window is open.

Procedure

To create the specific text field for a supervision, follow these steps:

1. Select the "Specific text field" group.
2. Select the required tags from the autocomplete of the three associated values SD_4 - SD_6. The tags must be of the data type BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, REAL, LREAL, CHAR, WCHAR, STRING or WSTRING.
3. Enter the desired text in the text field. It can, for example, consist of a free text and a reference to the tags which you have defined as associated values. (for example, @4 references to the tag of the associated value SD_4)
   For information on what the syntax of the reference to a tag must look like, refer to: Use of the associated values (Page 7898)

Note

Display of the specific text field

This information is only output in the supervision alarm when you have used the text field "<Specific text field>" when configuring the alarm texts in the ProDiag supervision settings. You can find additional information on the alarm texts in the ProDiag supervision settings here: Specifying central ProDiag alarm texts (Page 7879)

Result

When a supervision alarm is triggered by a fault/error, the information from the specific text field is displayed if you have used the alarm text field "<Specific text field>" in the alarm text structure.

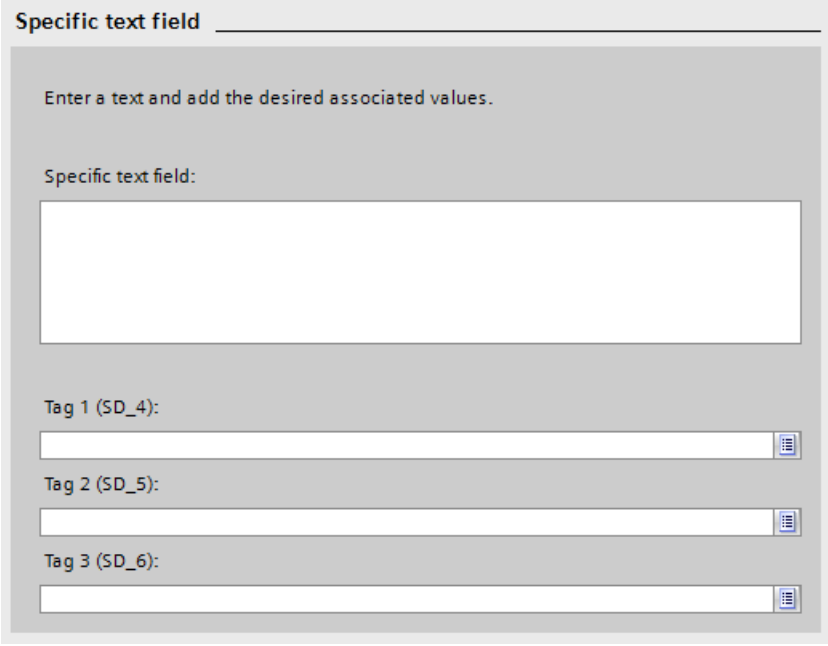

24.9.5.3 Displaying and editing translations of the specific text field (S7-1500)

You can display the specific text field in all project languages that have been activated in the setting options for the project languages and for which translations already exist. You can find additional information on activating project languages under AUTOHOTSPOT.

Requirement

You have created a specific text field for at least one ProDiag supervision.

Procedure

You have several options for displaying the translations of the specific text field:

- In the ProDiag overview editor
- In the block properties of the ProDiag function block
- In the ProDiag function block

To display the translations in the ProDiag overview editor, follow these steps:

1. Double-click the "PLC supervisions & alarms" entry in the project tree below the CPU.
   The alarm and supervision editor opens.
2. Click either the "Tag supervisions" tab or the "FB supervision definitions" tab.
3. Select either the entire row of the desired supervision or the cell that contains the text of the specific text field. If you want to display multiple texts in the "Texts" tab, select multiple cells with specific texts.
   The text of the specific text fields is displayed in the Inspector window in the "Texts" tab.

To display the translations in the block properties of the ProDiag function block, follow these steps:

1. Right-click the ProDiag function block and select "Properties..." from the shortcut menu.
   The properties dialog box of the ProDiag function block opens.
2. Click on the "Texts" tab.
   The texts of the specific text fields are displayed.

To display translations in the ProDiag function block, follow these steps:

1. Double-click the ProDiag function block.
   The ProDiag function block opens.
2. In the Inspector window, click on the "Texts" tab.
   The texts of the specific text fields are displayed.

If the names of the categories appear only in the current user interface language, either you have not yet selected any project languages or translations are not yet available. You can enter the translations directly in the table on the "Texts" tab or you can enter the translations for the individual project languages in the global "Project texts" table. You can find the table in the project tree under "Languages & Resources > Project texts".

You can find additional information on translation of texts under AUTOHOTSPOT.

24.9.6 Deleting a supervision (S7-1500)

Requirement

You have created at least one supervision.

Procedure

Proceed as follows to delete a supervision:

1. Click the tag or the parameter with the supervision to be deleted.
   The "Supervisions" tab opens in the "Properties" Inspector window.
2. Right-click the supervision and select "Delete supervision" from the shortcut menu.

Result

The supervision has been deleted.

Note

Monitoring is not deleted.

If you change the data type of a monitored operand or delete the monitored operand, the monitoring is not automatically deleted at the same time. If you really want to delete the operand including monitoring, then you must delete the monitoring separately.

24.10 Copying ProDiag function blocks (S7-1500)

24.10.1 Overview of copying ProDiag function blocks (S7-1500)

Description

Instead of creating the ProDiag function blocks, you can also copy them. The following options are available for copying:

- You can copy the ProDiag FB within a project from one S7-1500-CPU to another S7-1500-CPU.
- You can copy the ProDiag FB from one project to another.
- You can copy the ProDiag FB from a project to a project library.
- You can copy the ProDiag FB from a project to a global library.
- You can copy the ProDiag FB from a project library to a global library and vice versa.

All global supervisions included in the ProDiag function block are retained during copying. We therefore recommend to also copy the supervised global tags; otherwise you may encounter a compilation error.

The included local supervisions are not copied.

Note

Guaranteeing consistency of the ProDiag configuration

A ProDiag function block has to be compiled after copying in order to ensure the consistency of the ProDiag configuration. Any difference is always displayed in an offline comparison after copying.

See also

Copying ProDiag function block (Page 7904)

Pasting a ProDiag function block (Page 7905)

24.10.2 Copying ProDiag function block (S7-1500)

Requirement

You have selected the ProDiag function block.

Procedure

To copy the ProDiag function block, follow these steps:

1. Right-click the ProDiag function block.
2. Select the "Copy" command from the shortcut menu.

Result

A copy of the ProDiag function block is now on the clipboard and can be pasted to the desired location.

24.10.3 Pasting a ProDiag function block (S7-1500)

Requirement

You have copied a ProDiag function block.

Procedure

To paste a copied ProDiag function block and its global supervisions, follow these steps:

1. Open the desired location to which you want to paste the copied block.
2. Right-click and select the "Paste" command from the shortcut menu.
   - If you are pasting the ProDiag function block into the same S7-1500 CPU as the original block, the copy is pasted with the extension "_<consecutive number>".
   - If you are pasting the ProDiag function block into a different S7-1500 CPU where a ProDiag function block of the same name already exists, the "Paste" dialog box opens. Select the required option and confirm with "OK".

Result

The copied ProDiag function block has been pasted.

24.11 Assigning ProDiag function blocks (S7-1500)

24.11.1 Overview of the assignment to a ProDiag FB (S7-1500)

Description

Each ProDiag supervision must be assigned to a ProDiag function block so that it is activated in the user program.

When a global supervision is created, this is automatically assigned to a ProDiag function block. You can change this assignment either in the "Properties" Inspector window or in the ProDiag overview table "Tag supervisions". You can find additional information here: Assigning global supervisions to a ProDiag FB (Page 7906)

With local supervisions, you can assign the supervisions to a ProDiag function block either when a user-defined function block is called in the program code, in the ProDiag overview table "FB supervision instances", or in the properties of the user-defined instance data block. You can find additional information here: Assigning local supervisions to a ProDiag FB (Page 7907)

If you do not select a user-defined ProDiag function block, the supervisions are assigned by default either to the ProDiag function block "Default_SupervisionFB" or the first user-defined ProDiag function block found.

You can change the assignment of a supervision to a ProDiag function block at any time later on.

See also

Assigning an instance data block to a ProDiag FB (local supervisions) (Page 7909)

Changing the existing assignment of a ProDiag FB (Page 7910)

Removing the existing assignment of a ProDiag FB (Page 7911)

24.11.2 Assigning global supervisions to a ProDiag FB (S7-1500)

Requirement

You have created a global supervision.

Procedure

Proceed as follows to assign global supervisions to a ProDiag function block:

1. Click the "Supervisions" tab in the "Properties" Inspector window.
2. Click the desired supervision.

3. Click the "…" selection field at ProDiag FB.
   A selection dialog opens.
4. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
5. Select the corresponding ProDiag function block.

Or:

1. Double-click the "PLC supervisions & alarms" folder in the project tree.
   The alarm and supervision editor opens.
2. Click the "Supervisions > Tag supervisions" tab.
   All global supervisions are displayed.
3. Click the "…" selection field in the "ProDiag FB" column.
   A selection dialog opens.
4. From this point, the procedure described above from item 4 applies.

Result

The supervision has been assigned to a ProDiag function block.

See also

Creating global supervision (Page 7893)

24.11.3 Assigning local supervisions to a ProDiag FB (S7-1500)

Requirement

You have created at least one local supervision at a function block.

Procedure

Proceed as follows to assign local supervisionsof a function block to a ProDiag function block:

1. Call the function block in the program code.
   The "Call options" dialog opens.
2. Click the "Single instance" button.
3. Select the desired ProDiag function block from the selection box "...".
4. Confirm your entries with "OK".

Or:

1. Call the function block in the program code.
   The "Call options" dialog opens.
2. Click the "Multi instance" button.

3. Select the desired multi-instance.
4. Confirm your entries with "OK".
5. Call the function block that contains the multi-instance, also in the program code.
   The "Call options" dialog opens.
6. Click the "Single instance" button.
7. Select the desired ProDiag function block from the selection box "...".
8. Confirm your entries with "OK".

Or:

Proceed as follows to assign local supervisions to a ProDiag function block using the "Add new block" dialog:

1. Double-click the "Add new block" command.
   The "Add new block" dialog opens.
2. Click "Data block".
3. At "Type" select the existing function block which contains local supervisions from the drop-down list.
4. Select the desired ProDiag function block from the selection box "...".
5. Confirm your entries with "OK".

Or:

Proceed as follows to assign local supervisions to a ProDiag function block in the ProDiag overview editor using the "PLC supervisions & alarms" dialog:

1. Double-click the "PLC supervisions and alarms" folder in the project tree.
   The alarm and supervision editor opens.
2. Click the "Supervisions > FB supervision instances" tab.
   All the local supervisions are displayed.
3. Click the "…" selection field in the "ProDiag FB" column.
   A selection dialog opens.
4. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
5. Select the corresponding ProDiag function block.

Result

The ProDiag supervisions of the user-defined function block are instantiated.

24.11.4 Assigning an instance data block to a ProDiag FB (local supervisions) (S7-1500)

Description

You can assign an instance data block of a user-defined function block directly to a ProDiag function block.

You can make this assignment even if the user-defined function block does not yet contain any supervisions. In addition, the assignment to the ProDiag function block is retained when the user defined function block is deleted.

Procedure

Proceed as follows to assign an instance data block of a user-defined function block to a ProDiag function block:

1. Right-click the instance data block of a function block.
2. Select the "Properties" command in the shortcut menu.
   The properties dialog of the instance data block opens.
3. Click the "Attributes" section.
4. Click the "…" selection field at "Assign ProDiag FB" to select a ProDiag function block.
   A selection dialog opens.
5. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
6. Select the corresponding ProDiag function block.

Or:

1. Double-click the "PLC supervisions & alarms" folder in the project tree.
   The alarm and supervision editor opens.
2. Click the "Supervisions > FB supervision instances" tab.
   All the local supervisions are displayed.
3. Click the "…" selection field in the "ProDiag FB" column to select a ProDiag function block.
   A selection dialog opens.
4. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
5. Select the corresponding ProDiag function block.

Result

The ProDiag supervisions of the user-defined function block are instantiated.

24.11.5 Changing the existing assignment of a ProDiag FB (S7-1500)

Requirement

You have already created global or local supervisions and have assigned a ProDiag function block to them.

Procedure at global supervisions

Proceed as follows to change the existing assignment of global supervisions to ProDiag function blocks:

1. Double-click the "PLC supervisions & alarms" folder in the project tree.
   The alarm and supervision editor opens.
2. Click the "Supervisions > Tag supervisions" tab.
   All global supervisions are displayed.
3. Assign a different ProDiag function block to the corresponding supervisions in the "ProDiag FB" column.

or:

1. Click the supervised tag, for example, in the default tag table.
   The "Supervisions" tab opens in the "Properties" Inspector window and the properties of the supervision are displayed.
2. Click the "…" selection field at ProDiag FB.
   A selection dialog opens.
3. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
4. Select the corresponding ProDiag function block.

or:

1. Double-click the relevant ProDiag function block in the project tree.
   The ProDiag function block opens.
2. Click the "…" selection field in the "ProDiag FB" column.
   A selection dialog opens.
3. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
4. Select the corresponding ProDiag function block.

Procedure at local supervisions

Proceed as follows to assign an instance data block of a user-defined function block to a ProDiag function block:

1. Right-click the instance data block of a function block.
2. Select the "Properties" command in the shortcut menu.
   The properties dialog of the instance data block opens.

3. Click the "Attributes" section.
4. Click the "…" selection field at "Assign ProDiag FB" to select a ProDiag function block.
   A selection dialog opens.
5. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
6. Select the corresponding ProDiag function block.

Or:

1. Double-click the "PLC supervisions & alarms" folder in the project tree.
   The alarm and supervision editor opens.
2. Click the "Supervisions > FB supervision instances" tab.
   All the local supervisions are displayed.
3. Click the "…" selection field in the "ProDiag FB" column to select a ProDiag function block.
   A selection dialog opens.
4. Click the "Program blocks" folder in the left-hand column.
   All existing ProDiag function blocks are displayed in the right-hand column.
5. Select the corresponding ProDiag function block.

Result

The assignment of the ProDiag function block has been changed.

See also

Displaying supervisions of all called function blocks (Page 7921)

Display supervisions of all global tags (Page 7919)

24.11.6 Removing the existing assignment of a ProDiag FB (S7-1500)

Requirement

You have already created local supervisions and have assigned a ProDiag function block to them.

Procedure

To change the existing assignment of an instance data block to a ProDiag function block, follow these steps:

1. Double-click the "PLC supervisions & alarms" folder in the project tree.
   The alarm and supervision editor opens.
2. Click the "Supervisions > FB supervision instances" tab.
   All the local supervisions are displayed.

3. Click the "…" selection field in the "ProDiag FB" column.
   A selection dialog opens.
4. Click "None".

or:

1. Right-click the instance data block of a function block that contains local supervisions.
2. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the instance data block opens.
3. Click the "Attributes" section.
4. Click the "…" selection field at Assign ProDiag FB.
   A selection dialog opens.
5. Click "None".

Result

The assignment of the ProDiag function block has been removed.

24.12 Editing ProDiag function blocks (S7-1500)

24.12.1 Inserting new supervisions in the ProDiag FB (S7-1500)

Description

In addition to displaying the information for the created supervisions in the "Tag supervisions" tab, you can also edit supervisions and, in some cases, add new supervisions.

Requirement

You have assigned global supervisions to a ProDiag function block and have opened it.

Procedure

You have two options for adding a new supervision:

1. Right-click the last line of the table and select "Add new supervision" from the shortcut menu.
   A new supervision is created with the default values (type of supervision: "Operand", category: "1: Error", delay time: T#0ms).
2. Select a tag to be supervised.
3. Overwrite the default settings by changing the settings correspondingly.

or:

1. Enter a tag to be supervised in the last line in the "Supervised tag" column.
   A new supervision is created with the default values (type of supervision: "Operand", category: "1: Error", delay time: T#0ms).
2. If you specify a tag that you have not yet defined, the "Supervised tag" field turns red.
3. Define the tag in a tag table or data block.

Result

The new supervisions have been added.

24.12.2 Copying supervisions in the ProDiag FB (S7-1500)

Description

In addition to displaying the information for the created global supervisions in the "Tag supervisions" tab, you can also copy individual or several supervisions within the tab or into a different tab "Tag supervisions".

Requirement

You have created global supervisions.

Procedure

Proceed as follows to copy the complete line of a supervision:

1. Select the line you want to copy.
   You can also select several lines by clicking them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking the first and last row.
2. Select the "Copy" command from the shortcut menu.
3. Position the cursor at the end of the table.
4. Select "Paste" in the shortcut menu.

Result

- The supervision is copied to the destination.
- A new ID is assigned to the copied supervision.
- All other properties of the supervision remain unchanged.

24.12.3 Deleting supervisions in the ProDiag FB (S7-1500)

Description

In addition to displaying the information for the created supervisions in the "Tag supervisions" tab, you can also delete individual or several supervisions.

Requirement

You have created global supervisions.

Procedure

Proceed as follows to delete the complete line of a supervision:

1. Select the line with the supervision to be deleted.
   You can also select several lines by clicking them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking the first and last row.
2. Select the "Delete" command in the shortcut menu.

Result

The supervision is deleted.

24.12.4 Sorting supervisions in the ProDiag FB (S7-1500)

Description

In addition to displaying the information for the created global supervisions in the "Tag supervisions" tab, you can also sort the supervisions in the columns "Supervised tag" and "ProDiag FB".

Requirement

You have created global supervisions.

Procedure

Proceed as follows to sort the supervisions:

1. Select one of the two columns by which you want to sort.
2. Click the column header.
   The column is sorted in the order of increasing values.
   An up arrow shows the sort sequence.
3. In order to change the sort sequence, click the arrow.
   The column is sorted in the order of decreasing values. A down arrow shows the sort sequence.
4. To restore the original sequence, click a third time on the column header.

Result

The supervisions are sorted as desired.

24.13 Calling the ProDiag function blocks (S7-1500)

24.13.1 Overview of calling the ProDiag function blocks (S7-1500)

Description

Once you have created and configured the supervisions and compiled your project, the created ProDiag function blocks are automatically called in the ProDiag organization block.

This activates the supervisions in your project. You now have various options for visualizing the supervisions on your HMI device.

If you do not want to call the ProDiag function blocks in the ProDiag OB, you can also call them in other blocks (FCs, FBs or OBs). We recommend that you program the calls either directly in a cycle OB or in a block that is called in a cycle OB. In this case, the respective ProDiag function block is no longer called in the ProDiag OB.

See also

Displaying supervised user-defined blocks (Page 7927)

24.14 Load ProDiag blocks (S7-1500)

24.14.1 Load ProDiag blocks (S7-1500)

Description

During the loading operation, all information that is required for the reconstruction of the program, including symbolic information such as the names and comments for code and data blocks, is also loaded in the current project language.

The ProDiag blocks have no memory reserve and you can only load with re-initialization.

You can find additional information about loading blocks here: Downloading blocks for S7-1200/1500 (Page 7544)

24.15 Displaying ProDiag supervisions (S7-1500)

24.15.1 Overview of the display options (S7-1500)

Description

You have the option to have all the created supervisions displayed in different overviews. Depending on whether the supervision is local or global, the following display options are available:

Display option	Description
ProDiag overview editor	In the ProDiag overview editor you can have both the global and the local supervisions of a CPU displayed.
Properties of a function block	You can have all the local supervisions that you have created in the block interface of the function block displayed in the properties of a function block.
ProDiag function blocks	You can have all the global and instantiated local supervisions to which you have assigned this ProDiag function block displayed in the opened ProDiag function block.

See also

Displaying supervisions of a function block (Page 7925)

Displaying supervisions of all function blocks (Page 7920)

Displaying supervisions of all called function blocks (Page 7921)

Display supervisions of all global tags (Page 7919)

Inserting new tag supervisions in the ProDiag overview editor (Page 7921)

Displaying supervisions of a ProDiag function block (Page 7926)

Displaying supervisions in the ProDiag overview editor (Page 7919)

24.15.2 Displaying supervisions in the ProDiag overview editor (S7-1500)

24.15.2.1 Structure of the ProDiag overview editor (S7-1500)

Structure of the ProDiag overview editor

The following graphic shows an example of the components of the ProDiag overview editor:

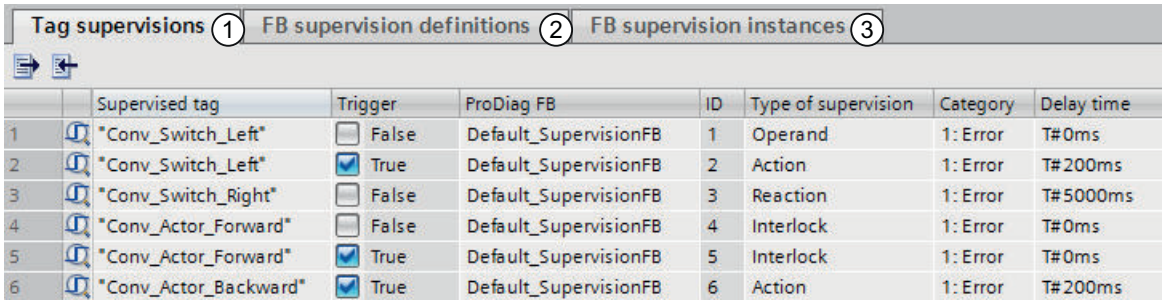

	Supervised tag	Trigger	ProDiag FB	ID	Type of supervision	Category	Delay time
1	"Conv_Switch_Left"	False	Default_SupervisionFB	1	Operand	1: Error	T#0ms
2	"Conv_Switch_Left"	True	Default_SupervisionFB	2	Action	1: Error	T#200ms
3	"Conv_Switch_Right"	False	Default_SupervisionFB	3	Reaction	1: Error	T#5000ms
4	"Conv_Actor_Forward"	False	Default_SupervisionFB	4	Interlock	1: Error	T#0ms
5	"Conv_Actor_Forward"	True	Default_SupervisionFB	5	Interlock	1: Error	T#0ms
6	"Conv_Actor_Backward"	True	Default_SupervisionFB	6	Action	1: Error	T#200ms

Numbering	Explanation	Additional information
1	The "Tag supervisions" tab shows all the global supervisions.	Additional information is available here: Display supervisions of all global tags (Page 7919)
2	The "FB supervision definitions" tab shows all the supervisions defined in the clock interface of user-defined function blocks even when this user-defined function block has not yet been called in the program code.	Additional information is available here: Displaying supervisions of all function blocks (Page 7920)
3	The "FB supervision instances" tab shows all the supervisions of the function blocks that have already been called in the program code.	Additional information is available here: Displaying supervisions of all called function blocks (Page 7921)
	Indicates that at least one supervision has been created for the tag.	

24.15.2.2 Display supervisions of all global tags (S7-1500)

Requirement

At least one tag is being supervised in a global data block or tag table.

Procedure

To display all global supervisions of a CPU, follow these steps:

1. Double-click the "PLC supervisions & alarms" folder in the project tree.
   The alarm and supervision editor opens.
2. Click the "Supervisions > Tag supervisions" tab.
   All global supervisions are displayed.

3. You can display and hide other columns in the table as needed. To do so, click in the column header and select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
4. To show a column, select the column's check box.
5. To hide a column, clear the column's check box.
6. To show or hide multiple columns, click "More" and select or clear the check boxes of the corresponding columns in the "Show/Hide" dialog.
   The sorting order of the columns cannot be changed.

Result

All global supervisions and the desired columns are displayed on the "Tag supervisions" tab.

24.15.2.3 Displaying supervisions of all function blocks (S7-1500)

Requirement

Supervisions have been created in the block interface of multiple function blocks

Procedure

To display the supervisions of all function blocks, follow these steps:

1. Double-click the "PLC supervisions & alarms" folder in the project tree.
   The alarm and supervision editor opens.
2. Click on the "Supervisions > FB supervision definitions" tab.
   All supervisions of all function blocks of the CPU are displayed even those that you have not instantiated yet.
3. You can display and hide other columns in the table as needed. To do so, click in the column header and select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
4. To show a column, select the column's check box.
5. To hide a column, clear the column's check box.
6. To show or hide multiple columns, click "More" and select or clear the check boxes of the corresponding columns in the "Show/Hide" dialog.
   The sorting order of the columns cannot be changed.

Result

All supervisions of all function blocks and the desired columns are displayed in the "FB supervision definitions" tab.

24.15.2.4 Displaying supervisions of all called function blocks (S7-1500)

Requirement

At least one function block that contains supervisions has been called in the program code.

Procedure

To display the supervisions of all called function blocks, follow these steps:

1. Double-click the "PLC supervisions & alarms" folder in the project tree.
   The alarm and supervision editor opens.
2. Click the "Supervisions > FB supervision instances" tab.
   All instance data blocks that contain supervisions of function blocks are displayed.
3. You can display and hide other columns in the table as needed. To do so, click in the column header and select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
4. To show a column, select the column's check box.
5. To hide a column, clear the column's check box.
6. To show or hide multiple columns, click "More" and select or clear the check boxes of the corresponding columns in the "Show/Hide" dialog.
   The sorting order of the columns cannot be changed.

Result

All instance data blocks that contain supervisions of function blocks and the desired columns are displayed in the "Instantiated supervisions at the FB" tab.

24.15.3 Working with the ProDiag overview editor (S7-1500)

24.15.3.1 Inserting new tag supervisions in the ProDiag overview editor (S7-1500)

Description

In addition to displaying the information for the created supervisions in the "Tag supervisions" tab, you can also edit supervisions and, in some cases, add new supervisions.

Procedure

You have three possibilities to add a new supervision:

1. Right-click the last line of the table and select "Add new supervision" from the shortcut menu. A new supervision is created with the default values (type of supervision: "Operand", category: "1: Error", delay time: T#0ms).
2. Select a tag to be supervised.
3. Overwrite the default settings by changing the settings correspondingly.

or:

1. Enter a tag to be supervised in the last line in the "Supervised tag" column. A new supervision is created with the default values (type of supervision: "Operand", category: "1: Error", delay time: T#0ms).
2. If you specify a tag that you have not yet defined, the "Supervised tag" field turns red.
3. Define the tag in a tag table or data block.

Result

The new supervisions have been added.

24.15.3.2 Copying tag supervisions in the ProDiag overview editor (S7-1500)

Description

In addition to displaying the information for the created global supervisions in the "Tag supervisions" tab, you can also copy individual or several supervisions within the tab or into a different tab "Tag supervisions".

Requirement

You have created global supervisions.

Procedure

Proceed as follows to copy the complete line of a supervision:

1. Select the line you want to copy. You can also select several lines by clicking them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking the first and last row.
2. Select the "Copy" command from the shortcut menu.
3. Position the cursor at the end of the table.
4. Select "Paste" in the shortcut menu.

Result

- The supervision is copied to the destination.
- A new ID is assigned to the copied supervision.
- All other properties of the supervision remain unchanged.

24.15.3.3 Deleting tag supervisions in the ProDiag overview editor (S7-1500)

Description

In addition to displaying the information for the created supervisions in the "Tag supervisions" tab, you can also delete individual or several supervisions.

Requirement

You have created global supervisions.

Procedure

Proceed as follows to delete the complete line of a supervision:

1. Select the line with the supervision to be deleted.
   You can also select several lines by clicking them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking the first and last row.
2. Select the "Delete" command in the shortcut menu.

Result

The supervision is deleted.

24.15.3.4 Sorting tag supervisions in the ProDiag overview editor (S7-1500)

Description

In addition to displaying the information for the created supervisions in the "Tag supervisions" tab, you can also sort them. The text columns are sorted alphabetically.

Requirement

You have created global supervisions.

Procedure

To sort the supervision by columns, follow these steps:

1. Select one of the columns by which you want to sort.
2. Click on the column header.

Result

The table is sorted by the selected column. A triangle with it pointing tip upwards appears in the column header.

Repeated clicking on the column header changes the sorting as follows:

- Symbol "▲": The column is sorted in the order of increasing values.
- Symbol "▼": The column is sorted in the order of decreasing values.
- No symbol: The sorting is canceled. The table assumes with default display.

24.15.3.5 Editing supervising definitions on the function block in the ProDiag overview editor (S7-1500)

Description

In the "FB supervision definitions" tab, you cannot only display the supervisions defined by user-defined function blocks in the block interface, but you can also edit them.

Requirement

You have defined supervisions for function blocks.

Procedure

To edit a supervision, proceed as follows:

1. Select the supervision field to be edited.
2. Change the supervision property to be edited.

Result

The supervision is edited.

24.15.3.6 Deleting supervision definitions on the function block in the ProDiag overview editor (S7-1500)

Description

In the "FB supervision definitions" tab, you cannot only display the supervisions defined by user-defined function blocks in the block interface, but you can also delete individual or multiple supervisions.

Requirement

You have defined supervisions for function blocks.

Procedure

To delete a complete supervision, proceed as follows:

1. Select the line with the supervision to be deleted.
   You can also select several lines by clicking them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking the first and last row.
2. In the shortcut menu, select the "Delete" command
   or
   press the "Del" key.

Result

The supervision is deleted.

24.15.4 Displaying supervisions of a function block (S7-1500)

Requirement

Supervisions have been created in the block interface of a user-defined function block.

Procedure

To display the supervisions of the function block, follow these steps:

1. Right-click the function block in the project tree and select "Properties..." from the shortcut menu.
   The properties dialog box of the function block opens.
2. Click on the "FB supervision definitions" tab.
3. You can display and hide other columns in the table as needed. To do so, click in the column header and select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
4. To show a column, select the column's check box.

5. To hide a column, clear the column's check box.
6. To show or hide multiple columns, click "More" and select or clear the check boxes of the corresponding columns in the "Show/Hide" dialog.
   The sorting order of the columns cannot be changed.

Or:

1. Double-click the function block in the project tree.
   As long as you do not select anything in the block interface, the supervisions are displayed in the Inspector window "Properties" in the tab "FB supervision definitions"
2. From this point, the procedure described above from item 3 applies.

Result

All supervisions of the user-defined function block and the desired columns are displayed in the "FB supervision definitions" tab.

24.15.5 Displaying supervisions of a ProDiag function block (S7-1500)

Requirement

At least one supervision has been assigned to the ProDiag function block.

Procedure

Proceed as follows to have all the global and local supervisions of a ProDiag function block displayed:

1. Double-click the desired ProDiag function block in the project tree.
   The ProDiag function block opens.
2. Click either the "Tag supervisions" tab or the "FB supervision instances" tab.
3. You can display and hide other columns in the table as needed. To do so, click in the column header and select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
4. To show a column, select the column's check box.
5. To hide a column, clear the column's check box.
6. To show or hide multiple columns, click "More" and select or clear the check boxes of the corresponding columns in the "Show/Hide" dialog.
   The sorting order of the columns cannot be changed.

Result

All the global supervisions are displayed in the "Tag supervisions" tab and all the local supervisions that are assigned to this ProDiag function block are displayed in the "FB supervision instances" tab.

24.16 Displaying supervised user-defined blocks (S7-1500)

24.16.1 Displaying supervised user-defined blocks (S7-1500)

You have the option of displaying all blocks of a CPU in which you have created supervisions in the overview of the project tree.

Requirements

You have created ProDiag supervisions and compiled your program.

Procedure

To display the overview, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. Click "Maximizes/minimizes the Overview".
3. Click in the column header and select the "Show/Hide" command in the shortcut menu.
4. The selection of available columns is displayed.
5. Select the check box in the "Supervisions" column to display the desired column.

Result

For all blocks that contain supervisions, the ProDiag symbol appears in the "Supervisions" column.

24.17 Exporting and importing ProDiag supervisions and properties of a ProDiag FB (S7-1500)

24.17.1 Overview of exporting and importing supervisions and properties of a ProDiag FB (S7-1500)

Description

It is possible to export the properties of one or more ProDiag function blocks in addition to the created supervisions from the ProDiag overview tables "Tag supervisions" and "FB supervision definitions".

You can export these supervisions and properties to a standard XLSX format (Excel Spreadsheet) and work with external table editors (for example, to translate). You can also import new supervisions, properties, and new ProDiag function blocks added with external table editors to the TIA Portal. The supervision, properties and ProDiag function blocks are created automatically by the system.

From the ProDiag overview table "FB supervision instances" you can export and import assignments of ProDiag function blocks to local supervisions.

Format of the export file

The names of the column titles are fixed by the system and must not be modified. All columns are exported even if you have hidden them in the ProDiag overview table. Each supervision has a separate line in the file.

The following table specifies the contents expected for the individual columns of the "ProDiag Supervisions" tab:

Column	Explanation
ProDiag FB	Name of the associated ProDiag function block
ID	Number of the supervision
Supervised tag / parameter	Supervised tag / parameter
Trigger	Signal state for which the operand is supervised: • TRUE • FALSE
Type of supervision	Type of supervision: • Operand • Interlock • Reaction • Action • Position • Text message • Error message

Column	Explanation
Category	Category • 1 (Error) • 2 (Warning) • 3 (Info)
Subcategory 1 and 2	Subcategories that you have created in the ProDiag supervision settings
Delay time	Depending on the type of supervision, either: • Delay time • Reaction time • Start-up time
C1, C2 or C3 trigger	Signal state for which the conditions are to be supervised: • TRUE • FALSE
Condition 1, 2 or 3	Operand used as the condition
Specific text field	Specific text field
Tag 1 (SD_4), Tag 2 (SD_5) or Tag 3 (SD_6)	Associated values of the text field that are read out during runtime

The following table specifies the contents expected for the individual columns of the "ProDiag Settings" tab:

Column	Explanation
ProDiag FB	Name of the associated ProDiag function block
Global enabler	Global enabler
Category	Category • 1 (Error) • 2 (Warning) • 3 (Info)
Category enabler	Enable criteria of the categories
Acknowledge tag	Acknowledgment tags for individual categories
Display class	Display class

Example

The figure below shows an excerpt from the export file as an example:

	A	B	C	D	E	F
1	ProDiag FB	ID	Supervised tag	Trigger	Type of supervision	Category
2	"Default_SupervisionFB"	3	"Conv_Actor_Forward"	False	O	1
3	"Default_SupervisionFB"	4	"Conv_Actor_Backward"	False	O	1
4	"Default_SupervisionFB"	5	"Conv_Forward_Trigger"	False	O	1
5	"Default_SupervisionFB"	6	"Conv_Backward_Trigger"	False	O	1
6	"Default_SupervisionFB"	7	"Conv_Switch_Left"	False	O	1
7	"Default_SupervisionFB"	8	"Conv_Switch_Right"	False	O	1
8	"Default_SupervisionFB"	9	"Conv_Push_Button_Forward"	False	O	1
9	"Default_SupervisionFB"	10	"Conv_Push_Button_Backward"	False	O	1

See also

Exporting supervisions and properties of a ProDiag FB (Page 7930)

Importing supervisions and properties of a ProDiag FB (Page 7932)

24.17.2 Exporting supervisions and properties of a ProDiag FB (S7-1500)

Requirement

You have created global supervision and defined properties for one or more ProDiag function blocks.

The supervision settings and the following settings are exported from the block properties:

- "General" area: The block version
- "Attributes" area: The "Initial value acquisition" and "Use central time stamp" options
- "Supervision settings" area: All settings

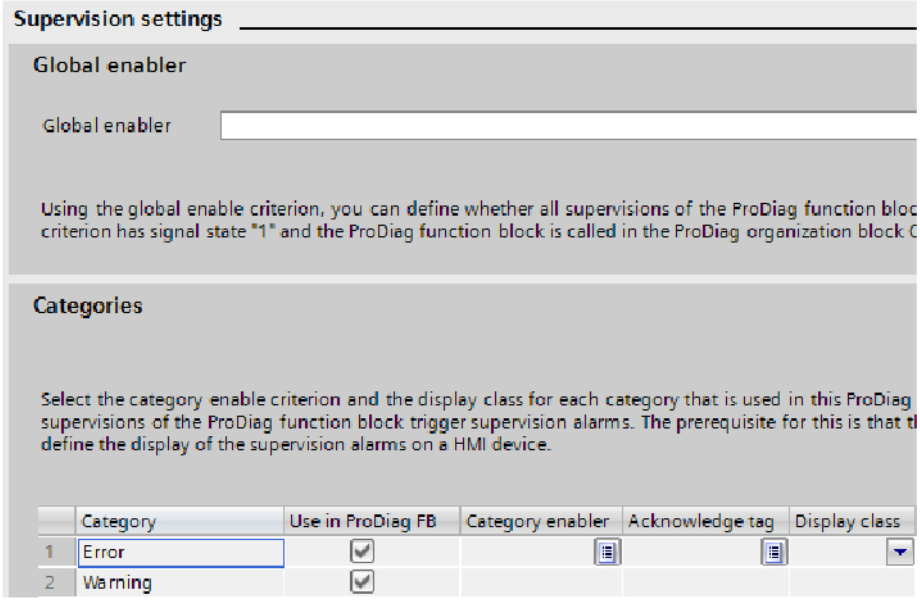

Note

Exporting the properties of a ProDiag function block

The properties of a ProDiag function block can only be exported from the "Tag supervisions" tab.

Procedure

To export the supervisions and properties of one or more ProDiag function blocks, follow these steps:

1. Open the "PLC supervisions & alarms" folder in the project tree.
2. Click on the "Supervisions" tab.
3. In the "Tag supervisions" tab, click "Exports tag supervisions" in the toolbar.
   The "Save as" dialog opens.
4. Select a storage location and assign a file name.
5. Click "Save".
   If the file was successfully saved, the "Export completed" dialog opens.
6. Click "OK" to confirm.

or:

1. Open the ProDiag function block with a double-click.
2. Click "Exports tag supervisions" in the toolbar. You can only export global supervisions from the ProDiag function block.
   The "Save as" dialog opens.
3. Select a storage location and assign a file name.
4. Click "Save".
   Once the file was successfully saved, the "Export completed" dialog opens.
5. Click "OK" to confirm.

Result

An .xlsx file is created at the selected storage location.

- You can find the exported supervisions in the "ProDiag Supervisions" tab.

	A	B	C	D	E	F
1	ProDiag FB	ID	Supervised tag	Trigger	Type of supervision	Category
2	"Default_SupervisionFB"	1	"Monday"	False	O	1
3	"Default_SupervisionFB"	2	"Tuesday"	False	R	1
4	"Station_1"	1	"Wednesday"	False	P	1

- You can find all ProDiag-specific properties that you have defined for the ProDiag function blocks in the "ProDiag Settings" tab.

	A	B	C	D	E
1	ProDiag FB	Version	Initial value acquisition	Use central time stamp	Global enabler
2	"Default_SupervisionFB"	V2.0	False	False	
3	"Station_1"	V2.0	False	True	"Wednesday"
4	ProDiag FB	Category	Category enabler	Acknowledge tag	Display class
5	"Station_1"	1	"Tuesday"		0

24.17.3 Importing supervisions and properties of a ProDiag FB (S7-1500)

Requirement

You have exported the supervisions and properties of a ProDiag function block to an .xlsx file.

Procedure

To import the supervisions and properties of one or more ProDiag function blocks, follow these steps:

1. Open the "PLC supervisions & alarms" folder in the project tree.
2. Click on the "Supervisions" tab.

3. In the "Tag supervisions" tab, click "Imports tag supervisions" in the toolbar.
   The "Imports tag supervisions" dialog is opened.
4. Select the path where the file to be imported is stored.
   The "Open" dialog opens.
5. Select the desired .xlsx file.
6. Confirm the dialog with "Open".
7. You have several options in the "Imports tag supervisions" dialog:
   - "Import supervisions"
     If you enable the options "Import supervisions" and "Delete existing supervisions", the existing supervisions of ProDiag function blocks for which there are also supervisions in the .xlsx import file are deleted in the overview table and the newly created supervisions are added from the .xlsx file. If there are no supervisions for A ProDiag function block in the .xlsx import file, this is also excluded from the deletion.
     If you disable the "Delete existing supervisions" option, the existing supervisions are kept and the newly created supervisions are added.
   - "Import and overwrite supervision settings"
     If you enable the option "Import and overwrite supervision settings" and the ProDiag function block already contains supervision settings, the old settings are overwritten with the new settings.
8. Click the "Import" button.
   The "Import tag supervisions" dialog is opened. The dialog displays the current status of the import.
9. If the import was successful, the "Import completed" dialog opens.
10. If an error occurred during the import, the "Import completed with warnings" dialog opens.
    The following errors can occur:
    - The .xlsx file contains incorrect entries.
    - The maximum number of supervisions for a ProDiag function block was exceeded.
    - The function block specified in the .xlsx file is not contained in your program. (Can only occur when supervisions are imported within the "FB supervision definitions" overview table.)
11. In this case, click on "Click here to view the log file".
12. Click "OK" to confirm.

Or:

1. Open the ProDiag function block with a double-click.
2. Click "Imports tag supervisions" in the toolbar.
   The "Imports tag supervisions" dialog is opened.
3. All other steps are the same as described above, starting with step 4.

Result

The ProDiag overview table now shows the imported supervisions and the block properties in the ProDiag function blocks correspond to the previously exported properties.

For global supervisions the following applies:

If you have added new supervisions in the .xlsx file, these are automatically assigned to the corresponding ProDiag function block after the import.

For local supervisions the following applies:

With local supervisions initially only the supervision definitions are imported and only through the import of the assignments are the supervisions assigned to concrete ProDiag function blocks.

24.18 Exporting and importing assignments of ProDiag FBs to Instance DBs (S7-1500)

24.18.1 Overview of exporting and importing assignments (S7-1500)

Description

It is possible to export the created assignments of ProDiag function blocks to the instance data blocks of user-defined function block from the ProDiag overview table "FB supervision instances".

You can export these assignments to a standardized XLSX format (Excel spreadsheet) and edit (e.g. translate) them with external spreadsheet programs. Similarly, you can import new assignments that were added with external spreadsheet programs into the TIA Portal.

Format of the export file

The names of the column titles are fixed by the system and must not be modified. Each assignment has a separate line in the file.

The following table specifies the contents expected in the individual columns:

Column	Explanation
Instance	Name of the associated instance data block of a user-defined function block
ProDiag FB	Name of the ProDiag function block

Example

The figure below shows an excerpt from the export file as an example:

1	Instance	ProDiag FB
2	"LAD_FB_1500_DB"	"Default_SupervisionFB"
3	"FBD_FB_1500_DB"	

See also

Exporting assignments (Page 7935)

Importing assignments (Page 7936)

24.18.2 Exporting assignments (S7-1500)

Requirement

You have created local supervisions and called the function block in the program code.

Procedure

Proceed as follows to export the assignment:

1. Open the "PLC supervisions & alarms" folder in the project tree.
2. Click "FB supervision instances" in the "Supervisions" tab.
3. Click "Exports assignments of ProDiag FBs to instance DBs" in the toolbar.
   The "Save as" dialog opens.
4. Select a storage location and assign a file name.
5. Click "Save".
   If the file was successfully saved, the "Export completed" dialog opens.
6. Click "OK" to confirm.

Result

An .xlsx file is created at the selected memory location.

24.18.3 Importing assignments (S7-1500)

Requirement

You have exported assignments into an .xlsx file.

Procedure

Proceed as follows to import the assignments:

1. Open the "PLC supervisions & alarms" folder in the project tree.
2. Click "FB supervision instances" in the "Supervisions" tab.
3. Click "Imports assignments of ProDiag FBs to instance DBs" in the toolbar.
   The "Imports assignments of ProDiag FBs to instance DBs" dialog is opened.
4. Select the path where the file to be imported is stored.
   The "Open" dialog opens.
5. Select the desired .xlsx file.
6. Confirm the dialog with "Open".
7. Click the "Import" button.
   The "Imports assignments of ProDiag FBs to instance DBs" dialog is opened. The dialog displays the current status of the import.
8. If the import was successful, the "Import completed" dialog opens.

9. If an error occurred during the import, the "Import completed with warnings" dialog opens. The following errors can occur:
   - The .xlsx file contains incorrect entries.
   - The .xlsx file contains missing assignments.
10. In this case, click on "Click here to view the log file".
11. Click "OK" to confirm.

Result

The imported assignments are displayed in the ProDiag overview table.

24.19 Displaying ProDiag cross-references (S7-1500)

24.19.1 Overview of cross-references (S7-1500)

Description

Within your user program the cross-reference list provides you with an overview of the points of use of supervised tags and for example tags used as a condition within a ProDiag supervison.

You can jump directly to the point of use of a supervised tag from the cross-reference list.

You can find additional information displaying cross-references here: General information about cross references (Page 7651)

24.19.2 Displaying cross-references (S7-1500)

Description

The cross-reference information for a selected supervised tag (global supervisions) is displayed in the Inspector window in the tab "About > Cross reference list in the programming window. For each selected supervised tag, it shows which ProDiag function block it is assigned to and at which points it is used. If the supervised tag is not only supervised but is for example used as a condition in another ProDiag supervision this use is also listed. You can recognize a supervised tag because the identifier "PRODIAG Supervision" is entered in the "Access" column. If the tag is used as a condition, this is indicated in the "Access" column by the identifier "Read-only"

You can display the cross-references at the following locations in the TIA Portal:

- Default tag table
- ProDiag overview editor
  - "Tag supervisions" tab
- ProDiag function blocks
  - "Tag supervisions" tab

Procedure

To display the cross references in the Inspector window , follow these steps:

1. Click the supervised tag, for example, in the default tag table.
   In the Inspector window in the tab "About > Cross reference list" you will see the supervised tag and the assigned ProDiag function block.
2. Click the arrow to the left of the ProDiag function block.
   All points of use of this tag are displayed (supervision IDs).
3. Click on one of the displayed points of use, to jump directly to it.
   The corresponding ProDiag function block is opened and the relevant cell marked.

Note

Marking of a cell

If no cell is marked the corresponding column containing the tag is not displayed.

24.20 Acquiring initial values (S7-1500)

24.20.1 Overview of initial value acquisition (S7-1500)

Requirement

In order to enable initial value acquisition, you need to set version 2.0 in the ProDiag function block instead of version 1.0.

Initial value acquisition for ProDiag is available in the LAD and FBD programming languages.

The following requirements must be met in the user program for the initial values to be acquired:

- A maximum of 32 signal states is recorded. If a LAD/FBD network has more than 32 elements, no recording takes place.
- For global supervisions: The user program must have only one write access to the supervised tag and this write access must be placed on the right power rail.
- For local supervisions: A separate instance supervision is created for each new instance. If, however, local tags are used as a prefix in the calling block, the recording of the initial value only works if the associated instance of the calling block has the name "FBName_DB" and only the Input, Output and Static sections have been used in the block interface.

You can find additional information here: Activating initial value acquisition (Page 7943)

Recording of the initial values

The initial values can be recorded for the following supervision types:

Types of supervision	Global supervisions			Local supervisions		
	Input	Output	Bit memory	Input	Output	Static
Operand	-	-	-	x	-	-
Interlock	-	x	x	x	-	-
Action	-	-	-	x	-	-
Reaction	-	-	-	x	-	-
Position	-	-	-	x	-	-
Text message	-	-	-	-	-	-
Error message	-	-	-	-	-	-

Description

The initial value acquisition gives you the option of analyzing why a supervision alarm was triggered. The initial values of the operands that are programmed in the user program prior to the supervised operand are recorded for this purpose.

You have the option of testing the execution of your user program on the HMI device. The data and values on the HMI device are continuously synchronized with the CPU and updated. You can always see the current program status for each LAD or FBD block on the HMI device.

For global supervisions the initial values of all Boolean operands in the network are recorded. If the network contains multiple individual power rails that are not connected to each other, only the initial values of the respective power rail are recorded.

For local supervisions only the initial values that are specified as conditions for the supervised parameter with the block call are recorded.

The initial values are recorded as soon as an event occurs that triggers a supervision alarm. The initial values are saved in a DWORD in the static CRIT_ERR parameter in the ProDiag instance data block; each individual signal state takes up one bit. The initial values are recorded in the cycle in which the event occurred. The supervision alarm is then sent in the next cycle. A maximum of 32 signal states per network can be recorded in an LAD or FBD block. The initial values are not changed until a new event occurs in the active network.

You can display the actual values with the help of the PLC code view. As soon as an event occurs, you can have the initial values displayed and the bad operands can be immediately recognized (criteria analysis). You can switch between display of the actual values and display of the initial values.

For criteria analysis on the HMI device, the operands are always read starting from the right at the coil against the circuit to the left.

You can find additional information on visualization on the HMI device here: "Visualize processes > Using diagnostics functions > Supervising machinery and plants with ProDiag"

Supported instructions

The following instructions are supported in LAD and FBD for initial value acquisition:

Instructions	Display on the HMI device (HMI)
Bit logic operations	
Normally open contact	Initial values and criteria analysis
Normally closed contact	
Invert RLO	The instruction is supported but it is not relevant for initial values or criteria analysis.
Assignment	
Negate assignment	
Reset output	Initial values
Set output	
Set/reset flip-flop	Initial values and criteria analysis up to and including the instruction box
Reset/set flip-flop	
AND (FBD)	Initial values and criteria analysis
OR (FBD)	
Comparator operations	
Equal	Initial values and criteria analysis
Not equal	
Greater or equal	
Less or equal	
Greater than	
Less than	
Timers	

Instructions	Display on the HMI device (HMI)
TP	Initial values and criteria analysis up to and including the instruction box
TON	Initial values and criteria analysis
TOF	Initial values and criteria analysis up to and including the instruction box
TONR	
Counters	
CTU	Initial values and criteria analysis up to and including the instruction box
CTD	
CTUD	

For bit logic operations, the status of the operand is recorded. For comparators, the result of the comparison is recorded.

For flip-flops, both inputs (R and S) are recorded if they are interconnected.

For timers and counters, the status of the operand at the output, and the inputs if they are interconnected, are recorded. (for example with CTUD: CU, CD, R, LD)

Evaluation order on the HMI device in the PLC code view

The recording of the signal status in the respective DWORD or the list of operands in the symbol table of the PLC code view is carried out in a defined order. In an LAD network, for example, the process always starts in the power rail at the top left and ends at the bottom right.

Note

Observe the evaluation order during programming

Also in the criteria analysis on the HMI device, this order is adhered to for the list of the bad operands in the symbol table. Observe this evaluation order for the programming of the networks.

Additional information on the order of the list is available here: Example of order of the list (Page 7943)

See also

Structure of a ProDiag instance data block of version 2.0 (Page 7855)

Content of a ProDiag instance data block of version 2.0 (Page 7856)

24.20.2 Activating initial value acquisition (S7-1500)

Procedure

Proceed as follows to activate initial value acquisition:

1. Right-click the ProDiag function block and select "Properties..." from the shortcut menu. The properties dialog box of the ProDiag function block opens.
2. Select version V2.0 in the "General > Block" tab.
3. Click on the "Attributes" tab.
4. Activate the "Initial value acquisition (for criteria analysis on HMI devices)" option.
5. Click "OK".
6. Compile and download the user program.

Result

Initial value acquisition is activated as soon as the user program has been compiled. This means that the corresponding signal states are recorded in case of error.

24.20.3 Example of order of the list (S7-1500)

The recording of the actual or initial values and the list of operands in the symbol table of the PLC code view are subject to the same defined order.

Using the following examples you can see how the recording of the signal states within an LAD network and the listing of operands on the HMI device work.

Recording the signal states in case of error

The signal states are written to the static parameter CRIT_ERR in the following order:

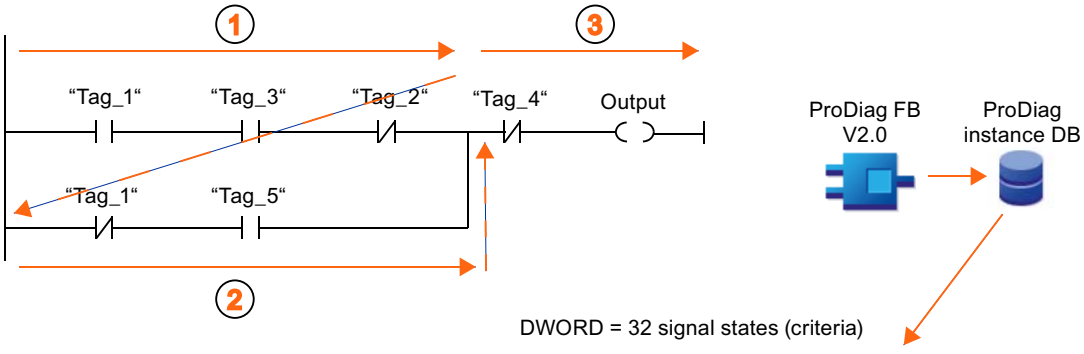

Bit 31	Bit ...	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	Tag_4	Tag_5	Tag_1	Tag_2	Tag_3	Tag_1

On the HMI device, all operands are listed in exactly this order in the PLC code view:

Symbol name	Operand	Comment
Tag_1	e.g. %M2xy	Automatic mode / manual mode
Tag_3		Switch on the motor
Tag_2		Interlock_1
Tag_1		Automatic mode / manual mode
Tag_5		Manual switch
Tag_4		Light barrier

List of operands in case of error

If errors occur in the network, the bad operands are marked in color and only the bad operands are listed in the symbol table:

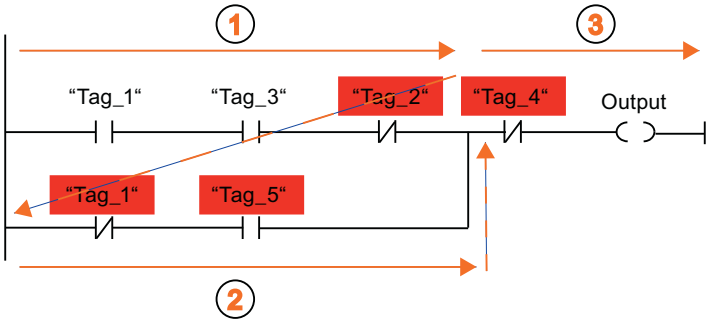

List of the bad operands in the symbol table of the PLC code view:

Symbol name	Operand	Comment
Tag_2	e.g. %M2xy	Interlock_1
Tag_1		Automatic mode / manual mode
Tag_5		Manual switch
Tag_4		Light barrier

Optimized programming of the network in case of error

In order that you can see at first glance that the light barrier is responsible for the error, in this example it is beneficial to program the operand "Tag_4: Light barrier" first:

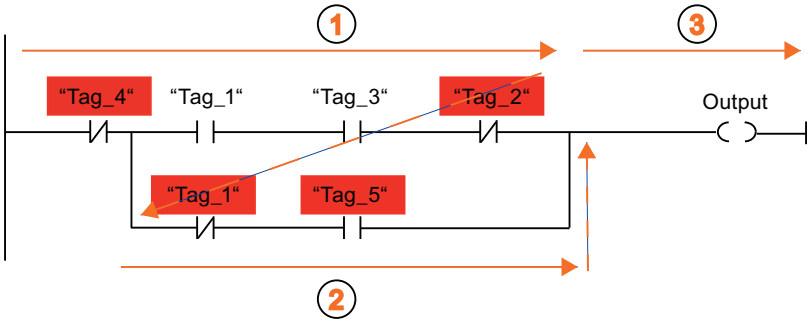

The information about the open light barrier is then displayed in the first line of the symbol table.

Symbol name	Operand	Comment
Tag_4	e.g. %M2xy	Light barrier
Tag_2		Interlock_1
Tag_1		Automatic mode / manual mode
Tag_5		Manual switch

Index

-

-, 7295
-D, 1537, 4767
-I, 1533, 4763
-R, 1544, 4775

#

#, 6977

&

&, 7303, 7304

(

(), 7304

*

*, 7295
**, 7295
*D, 1537, 4768
*I, 1534, 4764
*R, 1545, 4776

/

/, 7295
/D, 1538, 4768
/FC, 173
/I, 1535, 4764
/R, 1546, 4777

:

:=, 7304, 7305

+

+, 7295
+AR1, 1654, 4889
+AR2, 1656, 4890
+D, 1536, 4766
+I, 1532, 4762
+R, 1543, 4774

<

<, 7297
<=, 7297
<>, 7297, 7304

=

=, 7297
==, 7304

>

>, 7297
>=, 7297

A

A, 1482, 4708
A(, 1490, 4716
A_DEAD_B
    Input parameters, 5721
    Instruction, 5718
    Output parameters, 5722
AB, 7073
ABS, 739, 1087, 1546, 1712, 1927, 4261, 4472, 4778, 4932, 5063
ABS_CTRL, 5900
    Call, 5900
    Control signals, 5901
    Data blocks used, 5900
    Description, 5900
    Error response, 5902
    Function switches, 5918
    Job status, 5919
    Jobs, 5917
    Operating principle, 5900
    Return values, 5901
ABS_CTRL_451, 5904
    Call, 5904
    Control signals, 5905
    Data blocks used, 5904
    Description, 5904

Error response, 5906
Function switches, 5920
Jobs, 5920
Operating principle, 5904
Parameter, 5905
Return values, 5905
ABS_DIAG, 5902
Call, 5902
Commands, 5903
Data blocks used, 5902
Description, 5902
Diagnostics data, 5903
Error response, 5903
Operating principle, 5902
Return values, 5903
Startup, 5903
ABS_DIAG_451, 5906
Call, 5906
Data blocks used, 5906
Description, 5906
Diagnostics data, 5907
Error response, 5907
Jobs, 5907
Operating principle, 5906
Parameters, 5906
Return values, 5907
Startup, 5907
ABS_INIT, 5899
Call, 5899
Data blocks used, 5899
Description, 5899
Parameters, 5899
Return values, 5899
Absolute value, 739, 1087, 1546, 1712, 1927, 4261, 4472, 4778, 4932, 5063
Access coordination, 6515
Access to operands, 103, 104, 106, 108, 109, 110, 111, 270, 277, 281, 289
Accumulator, 1652, 1653, 4883, 4884, 4885, 4886, 4887
Accumulator instructions, 7245
Accuracy, 7308
ACOS, 757, 1105, 1553, 1726, 1939, 4277, 4490, 4785, 4943, 5075
ACT_TINT, 2457
Action
Basics, 7378
Block call, 7384
Changing, 7459
Copying, 7460
Counter operations, 7387
Cutting, 7460
Deleting, 7462
Event-dependent actions, 7381
Inserting, 7454, 7460
Introduction, 7377
Moving, 7461
Standard actions, 7380
Timers, 7385
AD, 1632, 4863, 7073
Add, 729, 1077, 1532, 1536, 1539, 1543, 4465, 4762, 4766, 4770, 4774
Adding timers with T_ADD, 5175
Adding
Adding timers with T_ADD, 2092
address conversion
Instructions for, 2738
Address register, 126, 127, 133, 134, 1654, 1656, 4889, 4890
Addresses
Determining a module with GEO_LOG, 2751, 5550
Read station address with GetStationInfo, 2600
Reading out the MAC address with GetStationInfo, 2600
Addressing operands, 103, 104, 106, 108, 109, 110, 111, 114, 120, 121, 125, 126, 127, 129, 130, 131, 132, 133, 134, 270, 277, 281, 289
ADM_ZONE
In/out parameters, 6078
Instruction, 6076
Output parameters, 6077
Static variables, 6078
AG_CNTEX, 6521
AG_CNTRL, 6536
AG_LOCK, 6517
AG_LRECV, 6653
AG_LSEND, 6646
AG_RECV / AG_LRECV / AG_SRECV, 6511
AG_SEND / AG_LSEND / AG_SSEND, 6506
AG_UNLOCK, 6519
AG-RECV, 6653
AG-SEND, 6646
Alarm
Alarm data type, 7499
Components, 7495
Edit GRAPH alarms, 7486
Enabling and disabling the GRAPH alarm view, 7485
Formal parameter, 7499
Time-stamped transferring with TIMESTMP, 5200
Alarm blocks, 7496

Alarm class
Copying, 7529
Creating, 7528
Editing, 7529
Alarm configuration
Additional text, 7520
Deleting an associate value, 7510
Entering text, 7520
Info text, 7520
Introduction, 7493
Locking an attribute, 7523
Locking text, 7522
Unlocking an attribute, 7524
Unlocking text, 7523
Alarm editor
Structure, 7501
Alarm number
Assigning, 7494
Alarm type, 7498
Editing, 7505, 7513
Alternative branch
Creating, 7448
Introduction, 7373
AN, 1483, 4709
AN(, 1491, 4717
ANALOG, 6097
Analog value, 852, 855, 1200, 1202, 1417, 1419, 1817, 1819, 2002, 2004
Analog value processing, 852, 855, 1200, 1202, 1417, 1419, 1817, 1819, 2002, 2004
AND, 892, 942, 944, 1239, 4316, 4397, 4398, 4529, 7303, 7304
ANY, 344
Application cycle, 3442
Arccosine, 757, 1105, 1553, 1726, 1939, 4277, 4490, 4785, 4943, 5075
Arcsine, 755, 1103, 1551, 1725, 1938, 4276, 4488, 4783, 4943, 5074
Arctangent, 1555, 1727, 4787, 4944
Arctangent value, 758, 1106, 1940, 4278, 4491, 5076
Arrangement of byte sequence, 1159, 1767
ARRAY, 121, 132, 200
Addressing, 109, 289
Declaration in global data blocks, 7030
Declaration in PLC data types, 7111
Declaration in the block interface, 6978
Example, 295
Format, 283
Reset bit,
Reset byte,
Reset in the I/O area,
see ARRAY,
Set bit,
Set byte array,
Set in the I/O area,
Textual block interface, 6998
ARRAY data block, 42, 44, 55, 106, 109, 812, 815, 817, 820, 1161, 1164, 1166, 1169, 1375, 1377, 1379, 1382, 1769, 1771, 1773, 1775, 1978, 1981, 1983, 1985, 6910, 7019, 7023, 7063, 7560
ARRAY DB, 44
ARRAY limits, 7317
ASI_CTRL, 2359, 5296
ASIN, 755, 1103, 1551, 1725, 1938, 4276, 4488, 4783, 4943, 5074
Assigning parameters
A module with PARM_MOD, 5360
Assignment, 603, 950, 1496, 4192, 4402, 4723, 7305
Negate, 604, 951
Assignment list
Defining filter,
Delete filter,
Displaying,
Enabling the display of retentive bit memories, 7632
Example for displaying bit memory, 7625
Example for displaying inputs and outputs, 7625
Filter options,
Filtering,
Introduction,
Meaning of symbols,
Setting view options,
Structure, 7625
Associated value
Example, 7517
Inserting, 7514
Settings, 7516
Asynchronous error
Delaying with DIS_AIRT, 5391
Disabling with DIS_IRT, 5388
Enabling with EN_AIRT, 5392
Enabling with EN_IRT, 5390
Instructions, 5386
Asynchronous error event
Delaying with DIS_AIRT, 2509
Disabling with DIS_IRT, 2501
Enabling with EN_AIRT, 2510
Enabling with EN_IRT, 2503
AT, 111
ATAN, 758, 1106, 1555, 1727, 1940, 4278, 4491, 4787, 4944, 5076
ATH, 2160, 5209

ATTACH, 2433
ATTR_DB, 2723
Autocompletion
Function, 6859
Insert tag, 6859, 6860
Inserting an instruction, 6860
AW, 1626, 4858, 7073
AX, 7073
Axis DB
Contents, 5867
Function, 5867

B

Base time
Calculate base time from local time with LT_BT, 5191
Basic mode, 7740
BCD, 1572, 1574, 4804, 4807
BCDCPL, 939, 1285, 1480, 1903, 2065, 4394, 4608, 4706, 5043, 5161
BE, 1620, 4851
BEC, 1621, 4852
BEU, 1622, 4853
Binary result, 173, 186
Bit array
Reset, 4564, 4660, 5001, 5121
Reset a bit array in the I/O area, 4352, 4565, 4661, 5002, 5122
Resetting, 4351
Set, 4559, 4655, 4997, 5117
Set in the I/O area, 4347, 4560, 4657, 4998, 5118
Setting, 4346
Bit field
Reset, 956
Resetting, 607
Set, 954
Setting, 606
Bit logic operation
AND, 942, 944, 1482, 1483, 1490, 1491, 4397, 4398, 4708, 4709, 4716, 4717
EXCLUSIVE OR, 948, 1487, 1488, 1494, 1495, 4400, 4713, 4714, 4720, 4721
Insert input, 949, 4401
OR, 944, 946, 1484, 1485, 1489, 1492, 1493, 4398, 4399, 4710, 4712, 4715, 4718, 4719
Bit mask, 930, 1276, 1467, 1895, 2054, 4377, 4590, 4687, 5027, 5146
Bit memories
Enabling the display of retentive bit memories, 7632
Bit string, 236, 237, 238
64-bit, 240
Bits
Count, 940, 1481, 1904, 2066, 4396, 4707, 5044, 5162
Counting, 1286, 4609
BITSUM, 940, 1286, 1481, 1904, 2066, 4396, 4609, 4707, 5044, 5162
BLD, 1657, 4897
BLKMOV, 835, 1183, 1408, 1801, 1988, 4282, 4495, 4629, 4945, 5079
Block,
Block, 1183, 4495
Call, 4337, 4549
Call conditionally, 1624, 4856
Changing passwords for know-how protected blocks, 6963
Closing, 6931
Comparing, 7590
Comparing code blocks, 7587
Comparison, 7585
Compiling, 7539
Compiling in the program editor, 7542
Compiling in the project tree, 7541
Consistency check, 7539, 7638, 7643, 7644
Consistency check in the call structure, 7638
Copying, 6913, 6916
Deleting offline, 6932
Deleting online, 6933
Disabling sequencer before loading a GRAPH DB, 7581
Displaying properties, 6928
Download to device, 7552
Downloading blocks to a memory card, 7579
Downloading to a memory card, 7558
Downloading to device, 7544, 7568, 7575
Downloading to device in RUN operating mode, 7548
Editing properties, 6928
Entering a comment, 6918
Entering a title, 6917
exporting to an external source file, 6938
Fill, 785, 841, 1134, 1189, 1353, 1413, 1745, 1806, 1956, 1993, 4288, 4501, 4635, 4951, 5085
Fill uninterruptible, 789, 1138, 1356, 1749, 1959
Find and open, 6929
Inserting, 6913
Know-how protection, 6943
Leave, 1844, 4975
Loading blocks from device, 7557
Loading to device in RUN operating state, 7571

Move, 1125, 1345, 1408, 1737, 1801, 1948, 1988, 4629, 4945, 5079
Move uninterruptible, 783, 838, 1131, 1186, 1351, 1411, 1743, 1804, 1954, 1991, 4285, 4498, 4633, 4949, 5083
Moving, 776, 835, 4282
Opening, 6929
Opening know-how protected blocks, 6960
Optimized access, 55, 57
Pasting, 6917
Printing know-how protected blocks, 6962
Properties, 6921
Remove write protection, 6956
Removing copy protection, 6954
Renaming, 6931
Reset bit, 4564, 4660, 5001
Reset byte, 4567, 4662, 5003
Reset in the I/O area, 4565, 4661, 5002
Saving, 6930
Set bit, 4559, 4655, 4997
Set byte array, 4562, 4658, 5000
Set in the I/O area, 4657, 4998
Setting up copy protection, 6954
Setting up write protection, 6956
Time stamp, 6925
Types, 39
Upload from device, 7544, 7568
Uploading blocks from a memory card, 7559, 7581
Uploading blocks from device, 7578
Using a library, 6911
Write to memory card, 7558, 7579
Block access
Data block, 7037
Block call
"Call by reference" or "Call by value", 80
Basics, 60
Calling as single instance or multi-instance, 63
Changing, 7146, 7209, 7484
Correcting the call type, 7146, 7209, 7274
GRAPH, 7384
Inserting, 7143, 7206, 7271, 7345, 7347, 7348, 7349, 7351, 7480, 7481
Multi-instance, 65
Nesting depth, 61
Pasting, 7270
Single instance, 63
Transfer parameter as copy or as pointer, 80
Updating, 7145, 7208, 7273, 7353, 7483
Block comment
Hiding, 6842
Showing, 6842
Block comparison
Basics, 7585
Comparing code blocks, 7587
Detailed comparison, 7592, 7594, 7596, 7600, 7603, 7606
Navigation, 7612
Synchronize scrolling, 7613
Updating comparison results, 7615
Block consistency
Checking, 7638
Checking in the dependency structure, 7644
Block end, 1620, 1621, 1622, 4851, 4852, 4853
Block folder, 6907
Block interface
Declare STRING, 6981
Declaring ARRAY, 6978
Declaring PLC data type, 6982
Declaring STRUCT, 6980
Declaring tags, 6975, 6977, 6983
GRAPH, 7390
Hiding, 6842
Importing and exporting tags, 6990
Multi-instance, 6983
Purpose of variable declaration, 6965
Retentivity, 6971
Return value, 6992
Showing, 6842
Structure, 6974, 6991
Tag properties, 7009, 7011
Textual block interface, 6991, 6992, 6993, 6994, 7001, 7002, 7003, 7004, 7006, 7007
Updating, 7015
Valid data types, 6968, 6970
Block parameters, 76, 78, 80, 83, 85, 86, 87
Block interface, 6974
Block property
Displaying, 6928
Editing, 6928
Function, 6920
Overview, 6921
BLOCK_DB_TO_WORD, 591
Blocking control loop calls, 5804
Bookmarks
Deleting, 7326
Function, 7324
Navigating, 7325
Setting, 7325
BOOL, 236, 357, 405, 492, 511, 558, 569
BOOL_TO_, 405, 511, 569
BR, 173, 186, 1501, 4338, 4550, 4729
Branch, 1495, 1833, 1834, 4722, 4963, 4965
Closing, 7165, 7449

Definition, 7163, 7226
Deleting, 7165, 7227, 7450
Inserting, 7164, 7227
Rules, 7163, 7226
BRCV, 3469
Breakpoint
Deleting, 7730
Disable, 7726, 7728, 7733
Enable, 7726, 7728, 7733
Navigating to the breakpoints, 7729
Resume program execution, 7732
Resume run to cursor, 7732
Setting, 7724
Setting the call environment, 7722
Step into, 7732
Step over, 7732
Testing in single step mode, 7716, 7734, 7736, 7738
BSEND, 3466
BTD, 1574, 4807
BTI, 1572, 4804
BUFFER parameter, Send_P2P, 3949, 6263
BUSY for asynchronous instructions, 4186
BY, 1836
BYTE, 236, 358, 407, 493, 512, 559, 570
BYTE_TO_, 407, 512, 570
Bytes
Swap, 811, 1159, 1767, 1976, 5077

C

CAD, 1583, 4816
CALC, 726, 1073
Calculate, 726, 1073, 7147, 7210
Call, 1623, 1624, 1625, 4337, 4549, 4855, 4856, 4857
ABS_CTRL, 5900
ABS_CTRL_451, 5904
ABS_DIAG, 5902
ABS_DIAG_451, 5906
ABS_INIT, 5899
CAM_CTRL, 5933
CAM_CTRL_452, 5936
CAM_DIAG, 5934
CAM_DIAG_452, 5938
CAM_INIT, 5932
CAM_MSRM_452, 5939
CNT_CTL1, 5872
CNT_CTL2, 5875
CNT_CTRL, 5870
CNT2_CTR, 5886
CNT2RDPN, 5886
CNT2WRPN, 5885
DIAG_INF, 5872
DIAG_RD, 5887
EncoderAbsSensorDP, 5840
EncoderCPU314C, 5847
EncoderET200S1Count, 5843
EncoderET200S1SSI, 5838
EncoderFM350, 5845
EncoderFM450, 5836
EncoderIM174, 5850
EncoderIM178, 5834
EncoderSINAMICS, 5852
EncoderSM338, 5841
EncoderUniversal, 5854
High_Speed_Counter, 2770
MC_Control, 5824
MC_GearIn, 5830
MC_Home, 5820
MC_Init, 5808
MC_MoveAbsolute, 5811
MC_MoveJog, 5816
MC_MoveRelative, 5813
MC_Simulation, 5825
MC_StopMotion, 5823
OutputCPU314C, 5858
OutputET200S2AO, 5857
OutputIM174, 5862
OutputIM178, 5856
OutputMM4_DP, 5861
OutputSINAMICS, 5866
OutputSM332, 5859
OutputSM432, 5857
OutputUniversal, 5860
SSI_Absolute_Encoder, 2780
Call data, 5718
Call environment
Basics, 7689
Setting for blocks, 7690
Setting for breakpoints, 7722
Call hierarchy, 61
Call structure, 7633
Displaying, 7636
Introduction, 7633
Meaning of symbols, 7634
Setting view options, 7637
Structure, 7635
CAM_CTRL, 5932
Call, 5933
Data blocks used, 5933
Description, 5932
Error response, 5934
Function switches, 5951

Job status, 5952
Jobs, 5951
Operating principle, 5933
Return values, 5933
CAM_CTRL_452, 5935
Call, 5936
Command status, 5953
Data blocks used, 5936
Description, 5935
Error response, 5937
Function switches, 5953
Jobs, 5952
Operating principle, 5936
Parameters, 5936
Return values, 5937
CAM_DIAG
Call, 5934
Commands, 5935
Data blocks used, 5934
Description, 5934
Error response, 5935
Operating principle, 5934
Return values, 5935
Startup, 5935
CAM_DIAG_452
Call, 5938
Data blocks used, 5938
Description, 5937
Error response, 5938
Jobs, 5938
Operating principle, 5937
Parameters, 5938
Return values, 5938
Startup, 5938
CAM_INIT
Call, 5932
Description, 5932
Parameters, 5932
Return values, 5932
CAM_MSRM_452, 5939
Call, 5939
Data blocks used, 5939
Description, 5939
Error response, 5940
Parameters, 5939
Return values, 5939
Startup, 5940
CAN_DINT, 2471
CAN_TINT, 2456
CAR, 1567, 4799
CASE, 1834, 4965
CAW, 1582, 4815
CC 0, 173, 178, 179, 180, 182, 186, 190
CC 1, 173, 178, 179, 180, 182, 186, 190
CD, 686, 1036, 1527, 4245, 4457, 4756
CDB, 1616, 4848
CEIL, 848, 1196, 1812, 1999, 4294, 4507, 4956, 5091
CH_DIAG
Block diagram, 5987
Input parameters, 5985
Instruction, 5984
Output parameters, 5985
CH_DIAG_455
Block diagram, 6066
Input parameters, 6064
Instruction, 6063
Output parameters, 6064
Channel data block, 5940
Channel DB
Configuration, 5908, 5940
Content, 5909, 5941
Purpose, 5908, 5940
Channel diagnostics
Diagnostic data, 5396
CHAR, 263, 399, 479, 507, 508, 550, 551, 568, 589
CHAR_TO_, 479, 550, 551, 589
Character, 263
Character string, 264, 267
Combine character strings with CONCAT, 5213
Combining character strings with CONCAT, 2167
Comparing string tags with S_COMP, 2126, 5205
Converting from a hexadecimal number with HTA, 2163, 5210
Converting number string with STRG_VAL, 2131
Converting number to character string with VAL_STRG, 2135
Converting to a hexadecimal number with ATH, 2160, 5209
Converting with S_CONV, 2128, 5206
Copying characters to character string with Chars_TO_Strg, 2143
Deleting characters with DELETE, 2173, 5215
Determine maximum length with MAX_LEN, 2146
Determining length with LEN, 2165, 5212
Finding a character with FIND, 2180
Finding characters with FIND, 5217
Inserting characters with INSERT, 2175, 5216
Move operation, 5206
Moving with S_MOVE, 2124
Reading out left character with LEFT, 2168
Reading out middle character with MID, 2171
Reading out right character with RIGHT, 2170
Reading the left character with LEFT, 5213

Reading the middle character with MID, 5214
Reading the right character with RIGHT, 5214
Replacing character with REPLACE, 2177
Replacing characters with REPLACE, 5216
Chars_TO_Strg, 2143
CHECKBACK_SIGNALS
CNT2_CTR, 5895
Checking the connection, 3603
Checksum
Reading out with GetChecksum, 2606
CiR, 4314, 4527, 4649, 4983, 5104
CiR process
Controlling, 4527, 4649, 4983
CJ_T_PAR
Input parameters, 5994
Instruction, 5994
Output parameters, 5995
CJ_T_PAR_455
Input parameters, 6072
Instruction, 6072
Output parameters, 6073
Close branch, 7374
CLR, 1500, 4728
CMP <, 705, 1054, 4252, 4463
CMP <=, 700, 1049, 4250, 4461
CMP <>, 693, 1042, 4247, 4459
CMP ==, 687, 1037, 4246, 4458
CMP >, 703, 1051, 4251, 4462, 5166, 5167, 5172
CMP >=, 698, 1047, 4248, 4460
CMP>T, 2072, 2083
CMP>T_MAX, 2075, 2086
CMP>T_WARN, 2076, 2087
CMP>U, 2073, 2084
CNT_CTL1, 5872
Call, 5872
Description, 5872
Error response, 5874
Jobs, 5873
Operating principle, 5872
Parameter, 5872
Startup characteristics, 5874
Value transfer, 5881
CNT_CTL2, 5874
Call, 5875
Description, 5874
Operating principle, 5874
CNT_CTRL, 5870
Call, 5870
Description, 5870
Error response, 5871
Jobs, 5871
Operating principle, 5870
Parameter, 5870
Startup characteristics, 5871
CNT2_CTR, 5886
Call, 5886
CHECKBACK_SIGNALS, 5895
CONTROL_SIGNALS, 5894
Description, 5886
Operating principle, 5886
Parameters, 5887
CNT2RDPN, 5885
Call, 5886
Description, 5885
Operating principle, 5886
Parameters, 5886
Read job, 5898
Status of the write job, 5898
CNT2WRPN, 5885
Call, 5885
Description, 5885
Operating principle, 5885
Parameters, 5885
Status of the write job, 5897
Write command, 5896
Coil, 603, 4192
Negate, 604
Command management, 5900
Command status
CAM_CTRL_452, 5953
Commands
ABS_DIAG, 5903
CAM_DIAG, 5935
Comment section, 7275
Comments
Inserting in SCL program, 7356
communication
Open User Communication,
Query architecture,
Status data, 5473
Communication interfaces
Programming, 3929, 6244
Communication module (CM)
Data reception, 3949, 6263
Programming, 3929, 6244
Communications module (CM), 4067
Comparator operations
Time tags, 2089, 5173
Compare
Bit mask, 930, 1276, 1467, 1895, 2054, 4377, 4590, 4687, 5027, 5146
Character strings with S_COMP, 2126, 5205
Equal, 687, 1037, 4246, 4458
Floating-point numbers, 1531, 4760

Greater or equal, 698, 1047, 4248, 4460
Greater than, 703, 1051, 2072, 2073, 2075, 2076, 2083, 2084, 2086, 2087, 4251, 4462, 5166, 5167, 5172
Integers (16 bit), 1528, 4757
Integers (32 bit), 1529, 4758
Less or equal, 700, 1049, 4250, 4461
Less than, 705, 1054, 4252, 4463
Not equal, 693, 1042, 4247, 4459
Compare scan matrix, 932, 1278, 1471, 1897, 2057, 4380, 4593, 4691, 5030, 5148
Compile
Block, 7541
Compiling
Blocks, 7539
Consistency check, 7539
Correcting compilation errors, 7543
COMPRESS, 4308, 4521, 4643, 4977, 5098
Compressing, 4308, 4643, 4977
CONCAT, 2167
Condition
Inserting a permanent instruction, 7477
Introduction, 7389
Programming a supervision, 7464
Programming an interlock, 7463
Programming transitions, 7469
CONF_DB see configuration data block, 6607
Configuration data block, 6607
Connection
canceling to partner outside of the local S7 station with X_ABORT, 6803
canceling to partner within the local S7 station with I_ABORT, 6787
Diagnosing with C_DIAG, 5510
Establishing with TCON, 6208
Querying status with C_CNTRL, 6157
Connection ID, 6612
Possible values, 6612
Connection types
Parameter fields for, 6612
Consistency check, 7638, 7643, 7644
Introduction, 7638, 7643
Constant
Data types, 100
Definition, 97
Global constant, 98, 100, 101, 7088, 7089, 7092, 7093
Local constant, 98, 100, 101, 6974, 6975, 7009
Non-typed constant, 97
Symbolic constant, 98
Typed constant, 97
Constants, 1539, 4770, 7313
CONT_C
Block diagram, 5563
Input parameters, 5564
Mode of operation, 5561
Output parameters, 5566
CONT_S
Block diagram, 5569
Input parameters, 5570
Instruction, 5566
Mode of operation, 5567
Output parameters, 5571
CONTINUE, 1841, 4972
Control signals
ABS_CTRL, 5901
ABS_CTRL_451, 5905
Writing, 5900, 5904
CONTROL_SIGNALS
CNT2_CTR, 5894
Controlled positioning
analog with ANALOG, 6097
Digital with DIGITAL, 6107
Controlling counter from user program with COUNT, 6116
Control-timer alarm, 923, 926, 1269, 1273, 1460, 1463, 1889, 1891, 2047, 2050, 4371, 4374, 4584, 4587, 4680, 4683, 5021, 5024, 5140, 5143
Conversion, 354, 489, 555
Explicit, 405, 407, 410, 413, 417, 421, 425, 428, 432, 435, 439, 442, 446, 449, 452, 455, 457, 460, 463, 466, 469, 472, 474, 476, 479, 481, 483, 486, 511, 512, 514, 517, 521, 524, 527, 530, 533, 536, 539, 542, 545, 547, 548, 549, 550, 551, 552, 554, 569, 570, 572, 574, 577, 580, 583, 584, 585, 587, 588, 589, 590, 591
Implicit, 357, 358, 361, 362, 364, 365, 367, 368, 370, 372, 374, 376, 378, 380, 381, 383, 385, 387, 389, 391, 393, 395, 397, 399, 400, 402, 403, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 558, 559, 561, 562, 563, 564, 565, 566, 567, 568
Conversion operations
Converting timers with T_CONV, 2091, 5174
Convert, 845, 1192, 1572, 1573, 1575, 1576, 1577, 1810, 1997, 4292, 4804, 4806, 4808, 4809, 4810, 4954, 5089
HW identifier to slot, 2738
IO address to HW identifier, 2738
Slot to HW identifier, 2738
Convert BCD to integer, 1574, 4807
Convert value, 4504

Converting
A character string to a hexadecimal number with ATH, 2160, 5209
Character string in array with Strg_TO_Chars, 2140
Character strings with S_CONV, 2128, 5206
Converting number string to number with STRG_VAL, 2131
Converting number to number string with VAL_STRG, 2135
Converting timers with T_CONV, 2091, 5174
Copying characters to numerical character string with Chars_TO_Strg, 2143
Hexadecimal number to character string with HTA, 2163, 5210
Copy protection, 6954
COS, 753, 1101, 1552, 1724, 1936, 4273, 4486, 4784, 4941, 5072
Cosine, 753, 1101, 1552, 1724, 1936, 4273, 4486, 4784, 4941, 5072
Count,
Down, 672, 680, 686, 1022, 1030, 1036, 1305, 1527, 1698, 1705, 1920, 4233, 4239, 4245, 4445, 4451, 4620, 4756, 4922, 4928, 5058
Up, 670, 678, 685, 1019, 1028, 1035, 1303, 1526, 1696, 1703, 1918, 4231, 4237, 4244, 4443, 4449, 4455, 4618, 4755, 4920, 4926, 5056
Up and down, 675, 682, 1025, 1308, 1700, 1706, 1923, 4235, 4241, 4446, 4622, 4924, 4930, 5060
COUNTER, 348, 682, 1706, 4241, 4930
Counter DB
Configuration, 5875
Contents, 5876, 5888
Function, 5875
Purpose, 5887
Counter operations
Enable, 1520, 4749
Reset, 1524, 4753
Set, 1525, 4754
Counters
Control high-speed counter (extended), 2767
Control high-speed counters, 2765
High-speed counters, 2765, 2767
Counting
Down, 4457
CountOfElements, 826, 1175, 1404, 1798
CP address, 6641
CPR_OUT
Input parameters, 5724
Instruction, 5723
Output parameters, 5724
CPU
Identifying with the system status list, 5461
Reading characteristics with the system status list, 5456
Reading date and time-of-day with RD_SYS_T, 2103
Reading the date and time-of-day with RD_SYS_T, 5179
Relationship of protocol variant and data length, 6199
Setting the time-of-day time-of-day with WR_SYS_T, 2101, 5178
Startup, 5901, 5905, 5934, 5937
CPU data block
Definition, 53
Deleting, 6933
CPU memory area
Displaying, 7649
Create bit pattern for seven-segment display, 1478, 2064
CREATE_DB, 2710
Creating a watch table, 7743
Crossing
Definition, 7166
Deleting, 7167
Inserting, 7166
rearranging, 7167
Cross-reference list, (See cross-references)
Adding new objects, 7677
Benefits, 7651
Displaying, 7658
Introduction, 7651
Overview, 7658
Settings, 7655
Sorting columns, 7653
Source object, 7653
Structure, 7653
Uses, 7651
Cross-references
Add new object, 7677
Displayed objects, 7651
Displaying, 7658
CRP_IN
Input parameters, 5723
Instruction, 5722
Output parameters, 5723
CTD, 672, 1022, 1305, 1698, 1920, 4233, 4445, 4620, 4922, 5058
CTRL_HSC, 2765
CTRL_HSC_EXT, 2767
CTRL_PTO, 2645
CTRL_PWM, 2643

CTU, 670, 1019, 1303, 1696, 1918, 4231, 4443, 4618, 4920, 5056
CTUD, 675, 1025, 1308, 1700, 1923, 4235, 4446, 4622, 4924, 5060
CU, 685, 1035, 1526, 1527, 4244, 4455, 4755, 4756
Cycle monitoring time, 877, 1224, 1436, 1853, 2016, 4310, 4522, 4645, 4978, 5100
Cyclic interrupt OB
    Querying parameters with QRY_CINT, 2445
    Setting parameters with SET_CINT, 2443

D

D_ACT_DP, 2310
Data block
    Adjusting data values during commissioning, 7061, 7063, 7065, 7066
    ARRAY data block, 42, 55, 106, 109, 6910, 7019, 7023, 7063, 7560
    Based on a PLC data type, 7033
    CONF_DB, 6229
    CPU data block, 53
    Creating, 6910, 7023
    Creating with CREA_DB, 5521
    Creating with CREA_DBL, 5523
    Creating with CREAT_DB, 5520
    Creating with CREATE_DB, 2710
    Declaration table, 7021
    Declaring ARRAY, 7030
    Declaring STRUCT, 7031
    Default value, 7035
    Deleting with DEL_DB, 5518
    Deleting with DELETE_DB, 2727
    Downloading changes without reinitialization, 7027
    Global data block, 42, 7019
    Importing and exporting tags, 7049
    Instance data block, 53, 7019
    Monitoring and modifying tags, 7053, 7054, 7055, 7056, 7058, 7059
    Open, 1614, 1615, 4306, 4307, 4519, 4520, 4847
    Optimized access, 55, 57
    Programming, 7019, 7029
    Reading attributes with ATTR_DB, 2723
    Reading from load memory with READ_DBL, 2714, 5514
    Retentive behavior, 7037, 7038
    Rule DB WS_RULES, 5204
    Snapshot, 7056, 7058, 7059
    Start value, 7035, 7036
    Tag properties, 7040, 7043, 7044
    Testing with TEST_DB, 5526
    Updating, 7025
    Using setpoints, 7061, 7063, 7065, 7066
    Writing to load memory with WRIT_DBL, 2718
    Writing to the load memory with WRIT_DBL, 5516
Data block area
    Fetching with FETCH_RK, 6775
    Sending with SEND_PTP, 6765
    Sending with SEND_RK, 6772
Data block register, 1616, 4848
Data blocks used
    ABS_CTRL, 5900
    ABS_CTRL_451, 5904
    ABS_DIAG, 5902
    ABS_DIAG_451, 5906
    ABS_INIT, 5899
    CAM_CTRL, 5933
    CAM_CTRL_452, 5936
    CAM_DIAG, 5934
    CAM_DIAG_452, 5938
    CAM_MSRM_452, 5939
Data consistency, 3442, 6133
Data exchange, 7245
Data log
    Closing with DataLogClose, 2685
    Creating with DataLogCreate, 2669
    Creating with DataLogNewFile, 2688
    Creating with DataLogTypedNewFile, 2690
    Deleting with DataLogDelete, 2687
    Empty with DataLogClear, 2681
    Opening with DataLogOpen, 2677
    Opening with DataLogTypedOpen, 2679
    Writing with DataLogWrite, 2683
Data record
    Asynchronously reading data record of a module with RD_DPARA, 2427, 5359
    Dynamically transferring parameters with WR_PARM, 5364
    Making available on I-device with PRVREC, 2344, 5270
    Reading from configured system data with RD_DPARM, 2429, 5363
    Reading of a module with RD_DPAR, 2421, 5357
    Reading with RD_REC, 2332, 5256
    Reading with RDREC, 2213, 5224
    Receiving on I-device with RCVREC, 2342, 5268
    Transferring with WRREC, 2232, 5226
    Writing and reading data records, 2420, 5355
    Writing predefined parameters with WR_DPARM, 2431, 5365
    Writing with WR_REC, 2337, 5260
Data transmission, trigger, 3946, 6260

Data type
ANY, 344
ARRAY, 283
BOOL, 236, 357, 405, 492, 511, 558, 569
BYTE, 236, 358, 407, 493, 512, 559, 570
CHAR, 263, 399, 479, 507, 508, 550, 551, 568, 589
Conversion, 354, 361, 489, 495, 504, 505, 506, 555, 561, 564, 566, 567
Convert explicit, 1997
DATE, 258, 393, 472, 505, 547, 566, 586
DINT, 245, 370, 435, 500, 533, 563, 580
DT, 259, 387, 463, 567, 588
DTL, 261, 391, 469, 507, 549
DWORD, 238, 361, 413, 495, 517, 561, 574
Explicit conversion, 405, 407, 410, 413, 417, 421, 425, 428, 432, 435, 439, 442, 446, 449, 452, 455, 457, 460, 463, 466, 469, 472, 474, 476, 479, 481, 483, 486, 511, 512, 514, 517, 521, 524, 527, 530, 533, 536, 539, 542, 545, 547, 548, 549, 550, 551, 552, 554, 569, 570, 572, 574, 577, 580, 583, 584, 585, 586, 587, 588, 589, 590, 845, 1192, 1810, 4954
GRAPH instruction, 7457
Implicit conversion, 357, 358, 359, 362, 364, 365, 367, 368, 370, 372, 374, 376, 378, 380, 381, 383, 385, 387, 389, 391, 393, 395, 397, 399, 400, 402, 403, 492, 493, 494, 496, 497, 498, 499, 500, 501, 502, 503, 507, 508, 509, 510, 558, 559, 560, 562, 563, 565, 567, 568
INT, 243, 367, 428, 498, 527, 562, 577
LDT, 261, 389, 466
LINT, 247, 374, 442
LREAL, 252, 380, 452, 503, 542
LTIME, 257, 385, 460
LTIME_OF_DAY, 259
LTOD, 397, 476
LWORD, 240, 362, 417
PLC data type, 273
POINTER, 341
REAL, 251, 378, 449, 502, 539, 563, 583
S5TIME, 255, 381, 455, 565, 585
SCL instruction, 7339, 7340
SINT, 241, 364, 421, 496, 521
STRING, 264, 402, 483, 509, 510, 552, 554, 568, 590
STRUCT, 278
TIME, 256, 383, 457, 504, 545, 564, 584
TIME_OF_DAY, 258
TOD, 395, 474, 506, 548, 567, 587
UDINT, 246, 372, 439, 501, 536
UINT, 244, 368, 432, 499, 530
ULINT, 248, 376, 446
USINT, 242, 365, 425, 497, 524
Validity, 231
VARIANT, 324
WCHAR, 263, 400, 481
WORD, 237, 359, 410, 494, 514, 560, 572
WSTRING, 267, 403, 486
Data types, 195
DataLogClear, 2681
DataLogClose, 2685
DataLogCreate, 2669
DataLogDelete, 2687
DataLogNewFile, 2688
DataLogOpen, 2677
DataLogTypedNewFile, 2690
DataLogTypedOpen, 2679
DataLogWrite, 2683
DATE, 258, 393, 472, 505, 547, 566, 586
DATE_AND_LTIME, 261
DATE_AND_TIME, 259
DATE_TO_, 472, 547, 586
Daylight Saving Time/Standard Time
Adjusting time-of-day status with SET_SW_S, 5197
Switching with SET_SW, 5194
DB, 6910, 7023
DB 101 to DB 104
In/out parameters, 6001
Input parameters, 5996
Output parameters, 5999
DB 101 to DB 116
In/out parameters, 6096
Input parameters, 6091
Output parameters, 6094
DB variable
Definition, 96
DB_ANY_TO_VARIANT, 1423, 1826
DCAT, 923, 1269, 1460, 1889, 2047, 4371, 4584, 4680, 5021, 5140
DEAD_T
Global data block, 5727
Input parameters, 5726
Instruction, 5725
Output parameters, 5726
DEADBAND
Input parameters, 5729
Instruction, 5728
Output parameters, 5729
DEC, 738, 1086, 1541, 4772
Decimal places, 759, 1107, 1728, 1941
DECO, 898, 1245, 1450, 1866, 2027, 4322, 4535, 4652, 4985, 5107

Decode, 898, 1245, 1450, 1866, 2027, 4322, 4535, 4652, 4985, 5107
Decrement, 738, 1086, 1541, 4772
Defining
For the assignment list, 7629
DELETE, 2173
DELETE_DB, 2727
Deleting
Filter in the assignment list, 7629
Deleting receive buffer with RES_RCVB, 6770
Demultiplex, 906, 1252, 1872, 2033
DEMUX, 906, 1252, 1872, 2033
Dependency structure, 7640
Displaying,
Introduction, 7640
Meaning of symbols, 7641
Setting view options, 7643
Structure, 7641
Description
ABS_CTRL, 5900
ABS_CTRL_451, 5904
ABS_DIAG, 5902
ABS_DIAG_451, 5906
ABS_INIT, 5899
CAM_CTRL, 5932
CAM_CTRL_452, 5935
CAM_DIAG, 5934
CAM_DIAG_452, 5937
CAM_INIT, 5932
CAM_MSRM_452, 5939
CNT_CTL1, 5872
CNT_CTL2, 5874
CNT_CTRL, 5870
CNT2_CTR, 5886
CNT2RDPN, 5885
CNT2WRPN, 5885
DIAG_INF, 5871
DIAG_RD, 5887
EncoderAbsSensorDP, 5839
EncoderCPU314C, 5847
EncoderET200S1Count, 5843
EncoderET200S1SSI, 5838
EncoderFM350, 5844
EncoderFM450, 5835
EncoderIM174, 5848
EncoderIM178, 5833
EncoderSINAMICS, 5851
EncoderSM338, 5841
EncoderUniversal, 5854
MC_Control, 5823
MC_GearIn, 5825
MC_Home, 5817
MC_Init, 5808
MC_MoveAbsolute, 5809
MC_MoveJog, 5814
MC_MoveRelative, 5812
MC_Simulation, 5824
MC_StopMotion, 5821
OutputCPU314C, 5858
OutputET200S2AO, 5857
OutputIM174, 5861
OutputIM178, 5856
OutputMM4_DP, 5860
OutputSINAMICS, 5863
OutputSM332, 5859
OutputSM432, 5857
OutputUniversal, 5859
Deserialize, 767, 1116, 1335, 1730, 1942
DETACH, 2437
Detailed comparison
Starting, 7592, 7594
Visualization of the comparison result for GRAPH, 7606
Visualization of the comparison result for LAD/FBD, 7596
Visualization of the comparison result for SCL, 7603
Visualization of the comparison result for STL, 7600
Device status
Querying a remote partner with STATUS, 6181
Receiving change with USTATUS, 6184
DeviceStates, 2619
DIAG_INF, 5871
Call, 5872
Description, 5871
Operating principle, 5871
Parameter, 5872
DIAG_RD, 5887
Call, 5887
Description, 5887
Operating principle, 5887
Parameters, 5887
Diagnostics
Connections with C_DIAG, 5510
Diagnostic data, 5394
Diagnostic data record, 5504
Diagnostic data, channel-specific, 5396
DP slave, reading diagnostic data with DPNRM_DG, 2353
DP slave, reading diagnostics data with DPNRM_DG, 5287
Module diagnostic information, 5505

Module, system status list, 5502
Writing a diagnostic event to the diagnostics buffer with WR_USMSG, 5428
Diagnostics buffer
System status list, 5501
Diagnostics data
ABS_DIAG, 5903
ABS_DIAG_451, 5907
Reading out with GET_DIAG, 2632
Diagnostics DB
Configuration, 5949
Content, 5950
Purpose, 5949
Diagnostics DB FM x51
Configuration, 5916
Content, 5916
Purpose, 5916
DIF
Input parameters, 5731
Output parameters, 5731
Differentiator, 5730
DIGITAL, 6107
Digital input
TIO_DI, 3429
Digital output
TIO_DQ, 3436
DINT, 245, 370, 435, 500, 533, 563, 580
DINT_TO_, 435, 533, 580
DIS_AIRT, 2509
DIS_IRT, 2501
Display
Communication instructions for non-configured S7 connections, 6127
Displaying
Assignment list,
Call structure, 7636
CPU memory area, 7649
Cross-references, 7658
Dependency structure,
Load memory, 7650
Maximum load memory available, 7650
Program information, 7623
Displaying a force table, 7773
Displaying monitor values for structures in the Inspector window, 7695
Displaying or hiding tag information, 7161, 7224, 7267, 7341, 7478
Displaying the monitor values for structures in the Inspector window, 7695
DIV, 733, 1081, 4257, 4468
Divide, 733, 1081, 1535, 1538, 1546, 4257, 4468, 4764, 4768, 4777
DMSK_FLT, 2486
DO, 1836, 1839, 4970
Double word, 238
Downloading
Blocks to the device, 7548
Downloading blocks to a memory card, 7557
Downloading blocks to device, 7553, 7554
in RUN operating mode, 7548
Uploading blocks from a memory card, 7559
DP master system
Determining topology with DP_TOPOL, 2356, 5290
Information, system status list, 5481
DP slave
Activating and deactivating with D_ACT_DP, 2310, 5251
Reading diagnostic data with DPNRM_DG, 2353
Reading diagnostics data with DPNRM_DG, 5287
Synchronizing groups with DP_SYC_FR, 2347, 5281
DP standard slaves
Reading a portion of the inputs with GETIO_PART, 2261, 5230
Reading all inputs with GETIO, 2252, 5228
Reading consistent data with DPRD_DAT, 2338, 5262
Writing a part of the outputs with SETIO_PART, 2263, 5231
Writing all outputs with SETIO, 2253, 5229
Writing consistent data with DPWR_DAT, 2340, 5265
DP_CTRL, 6676
DP_DIAG, 6668
DP_RECV, 6662
DP_SEND, 6659
DP_TOPOL, 2356
DPNRM_DG, 2353
DPRD_DAT, 2338
DPSYC_FR, 2347
DPWR_DAT, 2340
DRUM, 917, 1263, 1453, 1882, 2041, 4357, 4570, 4665, 5006, 5126
DRUM_X, 4363, 4576, 4672, 5013, 5132
DT, 259, 387, 463, 567, 588
DT_TO_, 463, 588
DTB, 1576, 4809
DTL, 261, 391, 469, 507, 549
DTL_TO_, 469, 549
DTR, 1577, 4810
Duration, 255
DWORD, 238, 361, 413, 495, 517, 561, 574

DWORD_TO_, 413, 517, 574
Dynamically allocated system resources
Releasing with DEL_SI, 5440

E

EB, 7073
ED, 7073
Edge
Negative, 613, 616, 618, 961, 964, 966, 1502, 1660, 4199, 4202, 4409, 4412, 4730
Positive, 611, 614, 617, 960, 963, 965, 1503, 1659, 4197, 4200, 4407, 4410, 4731
ELSE, 1833, 1834, 4963, 4965
ELSIF, 1833, 4963
E-mail
Transferring e-mail with TM_MAIL, 4178
Transferring e-mail with TMAIL_C, 3513
E-mail connection
Parameter field for, 6616
Empty box
Inserting a LAD element, 7138
Inserting an FBD element, 7201
EN/ENO mechanism
Basics, 141
FBD example, 150
LAD example, 146
SCL example, 158
STL example, 153
EN_AIRT, 2510
EN_IRT, 2503
Enable
Alarms with EN_MSG, 5434
ENCO, 899, 1246, 1451, 1867, 2028, 4323, 4536, 4653, 4986, 5108
Encode, 899, 1246, 1451, 1867, 2028, 4323, 4536, 4653, 4986, 5108
EncoderAbsSensorDP, 5839
Call, 5840
Description, 5839
Error handling, 5841
Parameter, 5841
Synchronization, 5841
EncoderCPU314C, 5847
Call, 5847
Description, 5847
Error handling, 5848
Parameter, 5847
Synchronization, 5847
EncoderET200S1Count, 5843
Call, 5843
Description, 5843
Error handling, 5844
Parameter, 5843
Synchronization, 5844
EncoderET200S1SSI, 5838
Call, 5838
Description, 5838
Error handling, 5839
Parameter, 5839
Synchronization, 5839
EncoderFM350, 5844
Call, 5845
Description, 5844
Error handling, 5846
Parameter, 5845
Synchronization, 5846
EncoderFM450, 5835
Call, 5836
Description, 5835
Error handling, 5837
Parameter, 5836
Synchronization, 5837
EncoderIM174, 5848
Call, 5850
Configuration of IM 174 (hardware configuration), 5849
Description, 5848
Error acknowledgment, 5851
Error handling, 5851
Parameter, 5850
Synchronization, 5850
EncoderIM178, 5833
Call, 5834
Configuration of IM 178-4 (hardware configuration), 5834
Description, 5833
Error acknowledgment, 5835
Error handling, 5835
Parameter, 5834
Synchronization, 5834
EncoderSINAMICS, 5851
Call, 5852
Configuration of SINAMICS (hardware configuration), 5851
Description, 5851
Error acknowledgment, 5854
Error handling, 5854
Parameter, 5852
Synchronization, 5853
EncoderSM338, 5841
Call, 5841
Description, 5841
Error handling, 5842

Parameter, 5842
Synchronization, 5842
EncoderUniversal, 5854
Call, 5854
Description, 5854
Error handling, 5855
Operating principle, 5854
Parameter, 5854
Synchronization, 5855
END_CASE, 1834, 4965
END_FOR, 1836
END_IF, 1833, 4963
END_REPEAT, 1840, 4971
END_WHILE, 1839, 4970
Endian, 1393, 1395, 1397, 1399, 1787, 1789, 1791, 1793
ENDIS_PW, 871, 1218, 1430, 1846, 2011
ENT, 4886
EQ_ElemType, 716, 1064, 1319
EQ_Type, 713, 1061, 1311, 1316
Equal, 719, 1067, 1322
EQUAL ARRAY, 722, 1070, 1325, 1710
ERR_MON
Input parameters, 5733
Instruction, 5731
Output parameters, 5733
Error analysis with RET_VAL, 162
Error codes
TIO_DI, 3430
TIO_DQ, 3440
TIO_IOLink_IN, 3427
TIO_IOLink_OUT, 3434
TIO_SYNC, 3423
Error handling
GetError, 879, 1226, 1438, 1854, 2017
GetErrorID, 883, 1230, 1441, 1857, 2020
Error information, 162, 879, 883, 1226, 1230, 1438, 1441, 1854, 1857, 2017, 2020
Error response
ABS_CTRL, 5902
ABS_CTRL_451, 5906
ABS_DIAG, 5903
ABS_DIAG_451, 5907
CAM_CTRL, 5934
CAM_CTRL_452, 5937
CAM_DIAG, 5935
CAM_DIAG_452, 5938
CAM_MSRM_452, 5940
CNT_CTL1, 5874
CNT_CTRL, 5871
High_Speed_Counter, 2774
SSI_Absolute_Encoder, 2783
ErrorID, 2780, 2786
Event
Assigning organization block (OB) with ATTACH, 2433
Canceling assignment to organization block (OB) with DETACH, 2437
EW, 7073
EX, 7073
Example
Calculating an equation, 7179, 7240, 7286
Controlling a conveyor belt, 7172, 7280, 7360
Controlling room temperature, 7181, 7242
Detecting the direction of a conveyor belt, 7174, 7234, 7282, 7362
Detecting the fill level of a storage area, 7176, 7236, 7284, 7364
for displaying bit memory in the assignment list, 7625
for displaying inputs and outputs in the assignment list, 7625
For entering force values in the force table, 7776
Heating an oven, 7288
Modify values in the watch table, 7748
Examples
Controlling a conveyor belt, 7232
EXCLUSIVE OR, 946, 948, 4399, 4400
Bit logic operation, 946, 4399
Exit,
User program, 878, 1225, 1437, 1853, 2017, 4310, 4523, 4645, 4979, 5100
EXP, 750, 1098, 1549, 1722, 1935, 4271, 4483, 4781, 4939, 5071
Expanded mode, 7740
Expanding and collapsing sections of code, 7323
Exponential value, 750, 1098, 1549, 1935, 4271, 4483, 4781, 5071
Exponentiate, 761, 1109
Expression
Arithmetic expression, 7295
Basics, 7294
Logical expression, 7303
Relational expressions, 7297
EXPT, 761, 1109
Extended status information, 7317
External source file
Basics, 6935
Editing, 6941
Exporting blocks, 6938
inserting, 6939
Linking file types to an editor, 6940
Opening, 6941
Rules for programming, 6937

F

F_TRIG, 620, 969, 1289, 1660
FALSE, 619, 620, 968, 969, 1288, 1289, 1659, 1660
Favorites
    Adding, 7141, 7204, 7268, 7342, 7431
    Hiding, 6842
    Removing, 7142, 7205, 7269, 7343, 7432
    Showing, 6842
    Using, 7141, 7204, 7268, 7343, 7432
FB, 41
FBD, 7185, 7232, 7234, 7236
    Detailed comparison, 7596
FBD element
    Copying, 7215
    Cutting, 7216
    Deleting, 7220
    Inserting, 7199, 7200
    Inserting an operand, 7222
    pasting from the clipboard, 7217
    Replacing, 7218
    Rules for inserting, 7198
    Selecting, 7215
FBD network
    Branch, 7226
    Deleting a branch, 7227
    Inserting a block call, 7143, 7206
    Inserting a branch, 7227
    Program status display, 7709
    Rules for branches, 7226
FC, 40
Feedback signals
    Read, 5904
    Reading, 5900
FETCH_RK, 6775
FieldRead, 831, 1179
FieldWrite, 833, 1181
FIFO, 336
FIFOQueue, 336
Fill,
    Block, 785, 841, 1134, 1189, 1353, 1413, 1745, 1806, 1956, 1993, 4288, 4501, 4635, 4951, 5085
    Block uninterruptible, 789, 1138, 1356, 1749, 1959
FILL_BLK, 785, 1134, 1353, 1745, 1956
Filter
    Defining for the assignment list, 7629
    Delete, 7629
    In the assignment list, 7628
    Selecting, 7630
Filters
    Assignment list,
FIND, 2180
First scan, 173
Flip-flop
    Reset/set, 610, 958, 4196, 4406
    Set/reset, 609, 957, 4195, 4405
Floating-point number, 251, 252
    Check invalidity, 712, 1060
    Check validity, 711, 1059
Floating-point numbers, 179, 253
    Invalid, 253
FLOOR, 850, 1197, 1814, 2000, 4296, 4508, 4957, 5092
FM 351
    Startup, 5901
FM 352
    Startup, 5934, 5937
FM 451
    Startup, 5905
FM 452
    Startup, 5934, 5937
FMT_CJ_T
    In/out parameters, 6041
    Input parameters, 6040
    Instruction, 6040
    Output parameters, 6041
FMT_DS1
    In/out parameters, 6028
    Input parameters, 6027
    Instruction, 6026
    Output parameters, 6027
FMT_PAR
    In/out parameters, 6024
    Input parameters, 6023
    Instruction, 6022
    Output parameters, 6023
FMT_PID
    Block diagram, 6013
    In/out parameters, 6007
    Input parameters, 6006
    Instruction, 6002
    Mode of operation, 6003
    Output parameters, 6006
    Static variables, 6007, 6009, 6011
FMT_PV
    In/out parameters, 6039
    Input parameters, 6038
    Instruction, 6034
    Output parameters, 6039
FMT_TUN
    In/out parameters, 6031

Input parameters, 6030
Instruction, 6029
Output parameters, 6030
Static variables, 6031
FN, 1502, 4730
FOR, 1836, 4967
Force
Force all, 7791
Stop forcing, 7793, 7794
Force table
"Monitor once and now" command for tags, 7784
Basic mode, 7771
Display, 7773
Expanded mode, 7771
Functionality, 7768
Layout, 7770
Meaning of the columns, 7770
Meaning of the icons, 7771
Monitoring and modifying modes, 7755
Opening, 7773
Overview of the display formats, 7777
Overview of the test options, 7768
Permitted operands, 7775
Permitted operands for force values, 7776
Saving, 7773
Switching between basic mode and expanded mode, 7771
Syntax check, 7774
Test options, 7768
Force value
Permitted operands, 7776
FORCE355
Input parameters, 5982
Instruction, 5980
Output parameters, 5982
FORCE455
Input parameters, 6061
Instruction, 6059
Output parameters, 6061
Forcing tags
Safety precautions, 7769, 7786
Form exponential value, 1722, 4939
Form square, 1932
Form square root, 1933
FP, 1503, 4731
FR, 1505, 1520, 4733, 4749
FRAC, 759, 1107, 1728, 1941
Frame configuration
Instructions, 3930, 6244
Free-form comments
Deleting, 7151, 7214
Editing, 7149, 7212
Inserting, 7149, 7212
Introduction, 7148, 7211
FREQUENC, 6120
Frequency measurement with FREQUENC, 6120
FTP, 3849, 6685
FTP connection
Parameter field for, 6617
FTP_CMD, 3849, 6619, 6685
FTPS, 3849, 6685
Function (FC)
Creating, 6909
Definition, 40
exporting to an external source file, 6938
Function block (FB)
Creating, 6909
Definition, 41
exporting to an external source file, 6938
Instance data block, 41
Function Block Diagram, 7185
Function switches
ABS_CTRL, 5918
ABS_CTRL_451, 5920
CAM_CTRL, 5951
CAM_CTRL_452, 5953
Functionality of the force table, 7768
FUZ_355
In/out parameters, 5980
Input parameters, 5979
Instruction, 5978
Output parameters, 5980
FUZ_455
In/out parameters, 6059
Input parameters, 6058
Instruction, 6057
Output parameters, 6059

G

GADR_LGC, 2756
GEN_DIAG, 2629
Gen_UsrMsg, 2533
Generate off-delay, 1914
Generate on-delay, 1912
Generate pulse, 1909
Generating diagnostics information, 2629
GEO_LOG, 2751
GEO2LOG, 2739
GET, 3447
Get_AlarmState, 2529
GET_DIAG, 2632
Get_IM_Data, 2590
GET_NAME, 2595

GetBlockName, 2195
GetChecksum, 2606
GetError, 879, 1226, 1438, 1854, 2017
GetErrorID, 883, 1230, 1441, 1857, 2020
GetInstanceName, 2187
GetInstancePath, 2190
GETIO, 2252
GETIO_PART, 2261
GetSMCinfo, 2609
GetStationInfo, 2600
GetSymbolName, 2181
GetSymbolPath, 2184
Global data block, 42
  Retentive behavior, 7038
Global library
  Overview of the USS protocol, 3971, 6285
Global tag, (See PLC tag)
GOTO, 1843, 4973
GRAPH, 160
  Acknowledging supervision errors, 7704
  Actions, 7378, 7380, 7381, 7384, 7385, 7387, 7454, 7459, 7460, 7461, 7462
  Alternative branch, 7373, 7448
  Block calls, 7480, 7481, 7483
  Block interface, 7390, 7393, 7397, 7400, 7403, 7404
  Changing the settings, 7414
  Close branch, 7374, 7449
  Conditions, 7389, 7463, 7464, 7469, 7477
  Data type basics, 7457
  Declaring internal parameters as retentive, 7420
  Declaring internal parameters as visible and accessible from HMI, 7420
  Deleting a branches, 7450
  Detailed comparison, 7606
  Disable all steps, 7703
  Disable the alarm view, 7485
  Editing alarms, 7486
  Enable the alarm view, 7485
  Initialize sequencer, 7703
  Interlock, 7389
  Jump to step, 7374, 7451, 7452, 7453
  Learning mode, 7465, 7467, 7468, 7707
  Operating modes, 7700, 7704
  Permanent instructions, 7388
  Permanent post-instructions, 7428
  Permanent pre-instructions, 7423
  Program status display, 7712
  Programming language, 7369
  Programming window, 7421
  Resetting registration requirements, 7392
  Sequence end, 7374
  Sequence view, 7424
  Sequencers, 7371, 7372, 7434, 7435, 7436, 7437, 7438
  Settings, 7412
  Simultaneous branch, 7372, 7448
  Single step view, 7426
  Specifying T_MAX and T_WARN manually, 7467
  Specifying the interface parameter set, 7417
  Specifying the memory space model, 7418
  Specifying the programming language, 7416
  Step, 7375, 7376, 7438, 7440, 7441, 7443, 7444, 7445, 7446
  Step time monitoring, 7465, 7466, 7467, 7468, 7707
  Supervision, 7389
  System synchronization, 7700, 7706
  T_MAX, 7707
  T_WARN, 7707
  Test settings, 7702, 7703
  Testing with program status, 7700
  Transitions, 7377, 7438, 7441, 7443, 7444, 7446
Greater than maximum step activation time, 2075, 2086
Greater than step activation time, 2072, 2083, 5166, 5172
Greater than uninterrupted step activation time, 2073, 2084, 5167, 5172
Greater than warning time, 2076, 2087

H

Hardware data types, 351
Hardware identifier, 2738, 3868, 6707, 7089
  Determining with LOG2MOD, 2744
Hidden parameters, 7160, 7223
High_Speed_Counter, 2770
  Call, 2770
  Description, 2770
  Error response, 2774
  Input parameters, 2775
  Operating principle, 2770
  Output parameters, 2777
  Static tags, 2778
Hot restart
  with RESUME, 6180
HTA, 2163, 5210

I

I&M data: Read with Get_IM_Data, 2590
I/Os, 104

I_ABORT, 6787
I_GET, 6784
I_PUT, 6786
IB, 7073
Icons in the force table, 7771
Icons in the watch table, 7742
ID, 7073
IEC Check,
    Setting, 357, 558
IEC timer, 256, 257
IF, 1833, 4963
IF_CONF for the configuration data (telecontrol), 3887, 6723
IM0_Data, 2590
IMC, 930, 1276, 1467, 1895, 2054, 4377, 4590, 4687, 5027, 5146
IN_RANGE, 708, 1056
INC, 737, 1085, 1541, 4771
Increment, 737, 1085, 1541, 4771
Indenting and outdenting
    Lines, 7321
Indirect addressing, 114, 120, 121, 125, 126, 129, 130, 131, 132, 133, 134, 196, 200
INIT_RD, 886, 1233, 1444, 1860, 2022
Initial step
    Canceling, 7440
    Specifying, 7440
Initialize all retain data, 886, 1233, 1444, 1860, 2022
Input
    Insert, 949, 4401
    Inserting, 7155
    remove, 7156, 7219
Input driver
    Call, 5832
    Error handling, 5833
    Function, 5832
    Initialization, 5832
    Simulation mode, 5833
Input parameter PID_ES, 5694
Input parameters
    High_Speed_Counter, 2775
    LP_SCHED_S, 5805
    SSI_Absolute_Encoder, 2783
Input parameters of GRAPH function block, 7393
INSERT, 2175
Insert a comment section, 1845, 4975
Insert input, 7218
Instance data block
    Creating, 53
    Definition, 53
    Modifying the data types of IEC timers and IEC counters, 7340
    Retentive behavior, 7037
Instance DB
    Creating, 7505, 7513
Instruction, 6641
    AG_SEND / AG_LSEND / AG_SSEND, 6506
    Copying, 7358
    Cutting, 7359
    Deleting, 7359
    Inserting, 7359
    Rules, 7328
    Search, 6846
    Specify data type, 7138, 7140, 7201, 7203, 7265
    Versions, 6850
Instruction profile
    Activating and deactivating, 6856
    Basics, 6853
    Creating, 6854
    Deleting, 6857
    Editing, 6855
    Opening, 6855
Instructions
    AG_LOCK / AG_UNLOCK, 6515
    AG_RECV / AG_LRECV / AG_SRECV, 6511
    for configured connections, 6503
    for DP mode, 6658
    For S5S5 connection, 6644
    General information, 6641
    P3964_Config (protocol configuration), 3944, 6258
    Port_Config (port configuration), 3934, 6249
    Receive_Config (receive configuration), 3939, 6253
    Receive_P2P (receive point-to-point data), 3949, 6263
    Receive_Reset (reset receiver), 3951, 6265
    S7-400 Automation System, 6425
    Send_Config (send configuration), 3937, 6252
    Send_P2P (send point-to-point data), 3946, 6260
    Signal_Get (get RS232 signals), 3952, 6266
    Signal_Get (set RS232 signals), 3954, 6268
    Symbolic name, 6641
    USS_Drive_Control / USS_Drive_Control_31, 3979, 6293
    USS_Port_Scan, 3976, 6290
    USS_Port_Scan_31, 3976, 6290
    USS_Read_Param / USS_Read_Param_31, 3983, 6297
    USS_Write_Param / USS_Write_Param_31, 3985, 6299

Instructions for access coordination with FETCH/WRITE, 6515
INT, 243, 367, 428, 498, 527, 562, 577
INT_TO_, 428, 527, 577
INTEG
    Input parameters, 5736
    Instruction, 5734
    Output parameters, 5736
Integer, 851, 1199, 1585, 1815, 2001, 4297, 4510, 4818, 4958, 5093
    16-bit, 243, 244
    32-bit, 245, 246
    64-bit, 247, 248
    8-bit, 241, 242
Integers, 178
Integrating texts from text lists in alarms, 7526
    Example, 7527
Interface, (See block interface)
Interface configuration
    Instructions, 3930, 6244
Interface parameter set, 7390, 7393, 7397, 7417
Interlock
    Introduction, 7376, 7389
    Programming, 7463
Internal parameters
    declaring as retentive, 7420
    Declaring as visible and accessible from HMI, 7420
Interrupt
    Delaying with DIS_AIRT, 5391
    Disabling with DIS_IRT, 5388
    Enabling with EN_AIRT, 5392
    Enabling with EN_IRT, 5390
Interrupt event
    Delaying with DIS_AIRT, 2509
    Disabling with DIS_IRT, 2501
    Enabling with EN_AIRT, 2510
    Enabling with EN_IRT, 2503
Interrupts
    Activating time-of-day interrupt with ACT_TINT, 2457, 5371
    Canceling time-delay interrupt with CAN_DINT, 2471, 5375
    Canceling time-of-day interrupt with CAN_TINT, 2456
    Canceling time-of-day interrupts with CAN_TINT, 5370
    Querying time-delay interrupt with QRY_DINT, 2471, 5376
    Querying time-of-day interrupt with QRY_TINT, 2459, 5371
    Reading interrupt status with the system status list, 5468
    Receiving from I/O module with RALRM, 2273, 5233
    Sending an I-slave with SALRM, 5272
    Setting time-of-day interrupt with SET_TINT, 2451, 5369
    Setting time-of-day interrupt with SET_TINTL, 2454
    Starting time-delay interrupt with SRT_DINT, 2469, 5374
    Triggering process interrupt with DP_PRAL, 5279
INV, 2026, 5106
INVD, 1578, 4811
Invert, 602, 897, 950, 1243, 4192, 4321, 4402, 4534
Inverting, 1499, 1578, 4726, 4811
INVI, 1578, 4811
IO address, 2738
IO2MOD, 2745
IO-Link, 3422, 3425, 3432
IP configuration
    Change parameters from the user program, 3612, 6226
IP_CONFIG, 6602
IS_ARRAY, 722, 1070, 1325, 1710
IS_NULL, 719, 1067, 1322
I-slave
    Sending interrupt with SALRM, 5272
    Triggering process interrupt with DP_PRAL, 5279
ISO-on-TCP connection
    Parameter field for, 6615
ITB, 1573, 4806
ITD, 1575, 4808
IW, 7073
IX, 7073

J

JBI, 1595, 4828
JC, 1590, 4823
JCB, 1592, 4825
JCN, 1591, 4824
JL, 1611, 4844
JM, 1604, 4837
JMP, 862, 1210, 4303, 4515
JMP_LIST, 865, 1213
JMPN, 863, 1211, 4304, 4516
JMZ, 1608, 4841
JN, 1601, 4834
JNB, 1594, 4827
JNBI, 1596, 4829
JO, 1597, 4830

Job buffer, 6644
Job execution
    Order, 5918, 5920
Job header, 6505, 6644, 6646
Job management, 5904
Job status, 5922
    ABS_CTRL, 5919
    CAM_CTRL, 5952
Jobs
    ABS_CTRL, 5917
    ABS_CTRL_451, 5920
    ABS_DIAG_451, 5907
    CAM_CTRL, 5951
    CAM_CTRL_452, 5952
    CAM_DIAG_452, 5938
    CNT_CTL1, 5873
    CNT_CTRL, 5871
JOIN, 2147
JOS, 1598, 4831
JP, 1603, 4836
JPZ, 1606, 1608, 4839, 4841
JU, 1589, 4822
Jump, 862, 863, 864, 865, 866, 869, 1210, 1211, 1212, 1213, 1214, 1217, 1588, 1611, 4303, 4304, 4305, 4306, 4515, 4516, 4517, 4518, 4821, 4844
    Ending sequencers, 7437
    if BR = 0, 1596, 4829
    if BR = 1, 1595, 4828
    if OS = 1, 1598, 4831
    if OV = 1, 1597, 4830
    if RLO = 0, 1591, 1594, 4824, 4827
    if RLO = 1, 1590, 1592, 4823, 4825
    If the result is greater than or equal to zero, 1606, 4839
    if the result is greater than zero, 1603, 4836
    if the result is invalid, 1609, 4842
    if the result is less than or equal to zero, 1608, 4841
    if the result is less than zero, 1604, 4837
    if the result is not zero, 1601, 4834
    if the result is zero, 1600, 4833
    Unconditional, 1589, 4822
Jump distributor, 866, 1214
Jump label, 864, 1212, 1588, 1843, 4305, 4517, 4821, 4973
Jump list, 865, 1213, 1611, 4844
Jump to step
    Changing the jump destination, 7452
    Deleting, 7453
    Inserting, 7451
    Introduction, 7374
    Navigating to the jump destination, 7452
JUO, 1609, 4842
JZ, 1600, 4833

K

Keyword highlighting, 7317
Keywords, 90
Know-how protection
    Changing a password, 6963
    Introduction, 6943
    Opening blocks, 6960
    Printing block, 6962
    remove, 6964
    Setting up, 6959

L

L, 1506, 1521, 1556, 4734, 4750, 4788
L DBLG, 1616, 4849
L DBNO, 1617, 4849
L DILG, 1618, 4850
L DINO, 1619, 4851
L STW, 1558, 4790
LABEL, 864, 1212, 4305, 4517
LAD, 7121, 7174
    Detailed comparison, 7596
LAD element
    Copying, 7153
    Cutting, 7153
    Deleting, 7157
    Inserting, 7136, 7137
    Inserting an operand, 7159
    pasting from the clipboard, 7154
    Replacing, 7155
    Rules for inserting, 7134
    Selecting, 7152
LAD network
    Branch, 7163
    Closing a branch, 7165
    Crossing, 7166
    Deleting a branch, 7165
    Deleting a crossing, 7167
    Insert crossing, 7166
    Inserting a block call, 7143, 7206
    Inserting a branch, 7164
    Program status display, 7708
    Prohibited interconnections, 7136
    Rearranging a crossing, 7167
    Rules for simultaneous branches, 7163
Ladder Logic, 7121

LAG1ST
Input parameters, 5739
Instruction, 5737
Output parameters, 5740
LAG2ND
Input parameters, 5743
Output parameters, 5743
LAR1, 1559, 4791
LAR1 <D>, 1560, 4792
LAR1 AR2, 1561, 4793
LAR2, 1562, 4794
LAR2 <D>, 1563, 4795
Layout of the force table, 7770
LC, 1508, 1523, 4736, 4752
LDT, 261, 389, 466
LDT_TO_, 466
Lead and lag algorithm, 934, 1280, 1475, 1899, 2061, 4382, 4595, 4695, 5032, 5150
LEAD_LAG, 934, 1280, 1475, 1899, 2061, 4382, 4595, 4695, 5032, 5150
Learning mode, 7465, 7466, 7467, 7468, 7707
LEAVE, 4887
LED, 2587
LED status
Reading out with LED, 2587
Reading the system status list, 5479
LEFT, 2168
LEN, 2165
LENGTH parameter, Send_P2P, 3949, 6263
LGC_GADR, 2757
Library
Adding and using a block, 6911
Library for the USS protocol
General information on drive setup, 3987, 6301
Overview, 3971, 6285
Requirements for use, 3973, 6287
USS_Drive_Control /
USS_Drive_Control_31, 3979, 6293
USS_Port_Scan, 3976, 6290
USS_Port_Scan_31, 3976, 6290
USS_Read_Param /
USS_Read_Param_31, 3983, 6297
USS_Write_Param /
USS_Write_Param_31, 3985, 6299
LIMALARM
Input parameters, 5745
Instruction, 5743
Output parameters, 5745
LIMIT, 744, 1092, 1331, 1717, 1930, 4265, 4477, 4627, 4935, 5067
Limit and enable password legitimation, 871, 1218, 1430, 1846, 2011
Limit value, 744, 1092, 1717, 4935
LIMITER
Input parameters, 5747
Instruction, 5745
Output parameters, 5747
Line comment, 7275
Linear programming, 37
LINT, 247, 374, 442
LINT_TO_, 442
LMNGEN_C
In/out parameters, 5751
Input parameters, 5750
Instruction, 5747
Output parameters, 5751
Static variables, 5751
LMNGEN_S
In/out parameters, 5758
Input parameters, 5757
Instruction, 5752
Output parameters, 5758
Static variables, 5759
LN, 749, 1097, 1550, 1721, 1934, 4270, 4482, 4781, 4938, 5070
Load
Loading block changes without reinitialization, 7562, 7565, 7566
Load memory, 7646
Displaying, 7650
Loading, 1556, 1558, 1559, 1560, 1561, 1562, 1563, 1616, 1617, 1618, 1619, 4788, 4790, 4791, 4792, 4793, 4794, 4795, 4849, 4850, 4851
Blocks in the device, 7571
Counter value, 1521, 1523, 4750, 4752
Downloading blocks to a memory card, 7579
Downloading blocks to device, 7575, 7576
In RUN operating state, 7571
Loading block changes without reinitialization, 7560, 7564
Switching off the sequencer prior to loading a GRAPH DB, 7581
Timer value, 1506, 1508, 4734, 4736
Uploading blocks from a memory card, 7581
Local time
Calculate local time from base time with BT_LT, 5190
Calculating with SET_TIMEZONE, 2110
Reading and calculating with LOC_TIME, 5189
Reading out with RD_LOC_T, 2105
Set time-of-day interrupt with S_LTINT, 5193
Writing with WR_LOC_T, 2107
Local variable, 6965

Log
Sending data with AR_SEND, 5432
LOG_GEO, 2753
LOG2GEO, 2742
LOG2MOD, 2744
Logarithm, 749, 1097, 1550, 1721, 1934, 4270, 4482, 4781, 4938, 5070
Logic operation
AD, 1632, 4863
AND, 892, 1239, 4316, 4529
AW, 1626, 4858
EXCLUSIVE OR, 895, 1242, 4319, 4532
OD, 1633, 4865
OR, 894, 1240, 4318, 4531
OW, 1628, 4860
XOD, 1635, 4866
XOW, 1630, 4861
Logic path
Deleting, 7230
Inserting, 7230
Use, 7229
Logical address
Determining a module with GADR_LGC, 2756, 5555
Determining a module with RD_LGADR, 2754, 5553
Determining associated slot with LGC_GADR, 2757, 5556
Determining associated slot with LOG_GEO, 2753, 5551
Long word, 240
Loop, 1612, 1833, 1836, 1839, 1840, 1841, 1842, 4845, 4963, 4967, 4970, 4971, 4972, 4973
Loop monitoring with S7-1200/1500, 7694
Loop monitoring with S7-300/400, 7694
LP_SCHED
Input parameters, 5716
LP_ZONE
In/out parameters, 6075
Input parameters, 6075
Instruction, 6073
Output parameters, 6075
Static variables, 6075
LREAL, 252, 380, 452, 503, 542, 7308
LREAL_TO_, 452, 542
LTIME, 257, 385, 460
LTIME_TO_, 460
LTOD, 259, 397, 476
LTOD_TO_, 476
LWORD, 240, 362, 417
LWORD_TO_, 417

M

Machine tailoring, (See Reconfiguring IO systems)
Master Control Relay, 4339, 4341, 4343, 4344, 4551, 4554, 4556, 4557, 4891, 4893, 4894, 4895
Master control relay (MCR)
Formal parameter access, 192
Functionality, 191
Notes on use, 192
Parameter transfer, 192
Math functions
CALCULATE, 7147, 7210
Mathematical functions
CALCULATE, 7179, 7240
MAX, 742, 1090, 1329, 1715, 1929, 4263, 4475, 4626, 4934, 5065
MAX_LEN, 2146
Maximum, 742, 1090, 1329, 1715, 1929, 4263, 4475, 4626, 4934, 5065
Maximum load memory available
Displaying, 7650
MB, 7073
MB_CLIENT, 4094, 4113
MB_COMM_LOAD, 4077
MB_MASTER, 4080
MB_RED_CLIENT, 4140
MB_SERVER, 4103, 4126
MB_SLAVE, 4088
MC_AbortMeasuringInput, 3149, 3301
MC_CamIn, 3173, 3180, 3325, 3331
MC_CamTrack, 3153, 3155, 3305, 3307
MC_ChangeDynamic
Instruction, 3034, 3068
Parameter, 3034, 3068
MC_CommandTable
Instruction, 3032, 3066
Parameter, 3033, 3067
MC_Control, 5823
Call, 5824
Control mode, 5823
Description, 5823
Follow-up mode, 5824
Manual mode, 5824
Operating principle, 5824
Parameter, 5824
MC_DefineKinematicsZone, 3238
MC_DefineTool, 3247
MC_DefineWorkspaceZone, 3236
MC_GearIn, 3156, 3159, 3308, 3311, 5825
Call, 5830
Description, 5825

Operating principle, 5825
Parameter, 5830
MC_GearInPos, 3160, 3164, 3312, 3316
MC_GetCamFollowingValue, 3185, 3336
MC_GetCamLeadingValue, 3187, 3337
MC_GroupContinue, 3207, 3208
MC_GroupInterrupt, 3206
MC_GroupStop, 3210, 3212
MC_Halt, 3116, 3119, 3267, 3270
Function chart, 3017, 3052
Instruction, 3015, 3050
Parameter, 3083
Parameters, 3016, 3051
MC_Home, 3111, 3262, 5817
Call, 5820
Description, 5817
Instruction, 3010, 3045
Operating principle, 5817
Parameter, 3012, 3047, 5821
Reference point approach, 5817
Set reference point, 5818
MC_Init, 5808
Call, 5808
Description, 5808
Operating principle, 5808
MC_InterpolateCam, 3184, 3335
MC_MeasuringInput, 3143, 3145, 3294, 3296
MC_MeasuringInputCyclic, 3145, 3147, 3296, 3299
MC_MotionInPosition, 3191, 3194
MC_MotionInVelocity, 3188, 3190
MC_MoveAbsolute, 3120, 3123, 3271, 3274, 5809
Call, 5811
Description, 5809
Function chart, 3021, 3055
Instruction, 3018, 3053
Operating principle, 5809
Parameter, 3019, 3054, 5811
MC_MoveCircularAbsolute, 3223, 3228
MC_MoveCircularRelative, 3229, 3234
MC_MoveJog, 3133, 3136, 3284, 3287, 5814
Call, 5816
Description, 5814
Function chart, 3031, 3065
Instruction, 3029, 3063
Operating principle, 5814
Parameter, 3030, 3064, 5816
MC_MoveLinearAbsolute, 3213, 3217
MC_MoveLinearRelative, 3218, 3222
MC_MoveRelative, 3124, 3127, 3275, 3278, 5812
Call, 5813
Description, 5812
Function chart, 3024, 3058
Instruction, 3022, 3056
Operating principle, 5812
Parameter, 3023, 3057, 5813
MC_MoveSuperimposed, 3137, 3140, 3288, 3291
MC_MoveVelocity, 3128, 3132, 3279, 3283
Function chart, 3028, 3062
Instruction, 3025, 3059
MC_OutputCam, 3150, 3152, 3302, 3304
MC_PhasingAbsolute, 3165, 3168, 3317, 3320
MC_PhasingRelative, 3169, 3172, 3321, 3324
MC_Power, 3103, 3108, 3255, 3259
Function chart, 3008, 3043
Instruction, 3004, 3039
Parameters, 3004, 3040
MC_ReadParam
Instruction, 3036, 3070
Parameter, 3037, 3071
MC_Reset, 3009, 3044, 3109, 3260
MC_SetKinematicsZoneActive, 3243
MC_SetKinematicsZoneInactive, 3245
MC_SetOcsFrame, 3250
MC_SetSensor, 3141
MC_SetTool, 3248
MC_SetWorkspaceZoneActive, 3240
MC_SetWorkspaceZoneInactive, 3242
MC_Simulation, 5824
Call, 5825
Description, 5824
Operating principle, 5825
MC_StopMotion, 5821
Call, 5823
Description, 5821
Operating principle, 5822
Parameter, 5823
Stopping a motion, 5822
MC_SynchronizedMotionSimulation, 3182, 3333
MC_TorqueAdditive, 3201, 3202
MC_TorqueLimiting, 3196, 3199, 3342
MC_TorqueRange, 3203, 3205
MC_WriteParam
Instruction, 3038, 3072
Parameter, 3038, 3072
MCAT, 926, 1273, 1463, 1891, 2050, 4374, 4587, 4683, 5024, 5143
MCR, 4339, 4341, 4343, 4344, 4551, 4554, 4556, 4557, 4891, 4893, 4894, 4895
MCRA, 4343, 4556, 4894
MCRD, 4344, 4557, 4895
MD, 7073
Meaning of the columns in the force table, 7770
Measure program runtime, 890, 1236, 1448, 1864

Memory
Compress, 4521, 5098
Compressing, 4643, 4977
Memory area
Reading the module with the system status list, 5458
Memory areas
System memory, reading with the system status list, 5459
Memory card
Reading out information with GetSMCinfo, 2609
Memory space model, 7390, 7418
MID, 2171
MIN, 741, 1089, 1328, 1713, 1928, 4262, 4473, 4624, 4933, 5064
Minimum, 741, 1089, 1328, 1713, 1928, 4262, 4473, 4624, 4933, 5064
MOD, 734, 1082, 1542, 4258, 4470, 4773, 7295
Modbus
Modbus_Comm_Load, 4005, 6319
Modbus_Slave, 4009, 4015, 6323, 6329
Modbus communication
as a slave with MB_SLAVE, 4088
as Modbus master with MB_MASTER, 4080
Configuring a port with MB_COMM_LOAD, 4077
Modbus master, 6440
Modbus_Comm_Load, 4005, 6319
Modbus_Slave, 4009, 4015, 6323, 6329
Modify
High-speed counters, 2765, 2767
Modify value
Permitted operands, 7748
Modifying mode, 7755
Module
Assigning parameters with PARM_MOD, 5360
Asynchronously reading data record with RD_DPARA, 2427, 5359
Determining logical address of the slot with LOG_GEO, 2753, 5551
Determining logical address with GADR_LGC, 2756, 5555
Determining logical addresses with RD_LGADR, 2754, 5553
Determining slot of a logical address with LGC_GADR, 2757, 5556
Determining start address with GEO_LOG, 2751, 5550
Diagnostic data, 5394
Diagnostics information, system status list, 5502
Identification via system status list, 5455
Module class, 5453
Module status information using the system status list, 5482
Reading data record with RD_DPAR, 2421, 5357
ModuleStates, 2624
Monitor GRAPH programs, 7712
Monitor SCL programs, 7711
Monitor STL programs, 7710
Monitoring
"Monitor all" command, 7757, 7783
"Monitor now" command, 7784
"Monitor once and now" command, 7758
Monitoring and modifying tags in the DB editor, 7053, 7054, 7055
Monitoring and modifying modes, 7755
Monitoring FBD programs, 7709
Monitoring GRAPH programs, 7712
Monitoring LAD programs, 7708
Monitoring loops, 7693
Monitoring mode, 7755
Monitoring SCL programs, 7711
Monitoring STL programs, 7710
Monitoring structures, 7695
Monitoring structures (S7-1200/1500), 7695
Move,
Block, 776, 835, 1125, 1183, 1345, 1408, 1737, 1801, 1948, 1988, 4282, 4495, 4629, 4945, 5079
Block uninterruptible, 783, 838, 1131, 1186, 1351, 1411, 1743, 1804, 1954, 1991, 4285, 4498, 4633, 4949, 5083
Character string in array with Strg_TO_Chars, 2140
Value, 762, 1110, 1333, 4280, 4493
Move block, 778, 1127, 1347, 1739, 1950
MOVE_BLK, 776, 1125, 1345, 1737, 1948
MOVE_BLK_VARIANT, 333, 778, 1127, 1347, 1739, 1950
MoveVelocity
Parameter, 3026, 3060
MSK_FLT, 2485
MUL, 732, 1079, 4256, 4467
Multicomputing interrupt
Triggering with MP_ALM, 5393
Multi-instance, 108
Declaring, 6983
Definition, 65
Multiple instance
Changing the call type, 7146, 7209, 7274
Multiplex, 903, 1249, 1869, 2031, 4988
Multiply, 732, 1079, 1534, 1537, 1545, 4256, 4467, 4764, 4768, 4776
MUX, 903, 1249, 1869, 2031, 4988
MW, 7073

MX, 7073

N

N, 613, 616, 961, 964, 4199, 4409
N_TRIG, 618, 966, 4202, 4412
NE_ElemType, 718, 1066, 1321
NE_Type, 714, 1062, 1318
NEG, 736, 1084, 1926, 4259, 4471, 5062
Negate, 736, 1084, 1926, 4259, 4471, 5062
Negating, 1579, 1580, 1581, 4812, 4813, 4814
NEGD, 1580, 4813
NEGI, 1579, 4812
NEGR, 1581, 4814
Nesting depth, 38, 61
Network
  Closing, 7130, 7194, 7254, 7474
  Copying, 7129, 7192, 7253, 7473
  Deleting, 7129, 7193, 7253, 7473
  Entering a comment, 7132, 7196, 7256
  Insert title, 7131, 7195, 7255, 7475
  Inserting, 7127, 7129, 7191, 7192, 7251, 7253, 7471, 7473
  Navigating blocks, 7133, 7197, 7257, 7476
  Opening, 7130, 7194, 7254, 7474
  Selecting, 7128, 7192, 7252, 7472
  Using, 7126, 7190, 7251, 7471
Network comments
  Hiding, 6842
  Showing, 6842
New data types, 195
NONLIN
  Input parameters, 5761
  Instruction, 5759
  Output parameters, 5761
non-typed, 7313
NOP 0, 1658, 4897
NOP 1, 1658, 4898
NORM
  Input parameters, 5764
  Instruction, 5763
  Output parameters, 5764
NORM_X, 855, 1202, 1419, 1819, 2004
Normalize, 855, 1202, 1419, 1819, 2004
Normally closed contact, 601, 4191
Normally open contact, 599, 4189
NOT, 602, 1499, 2026, 4192, 4726, 5106, 7303, 7304
Not equal, 721, 1069, 1324
NOT_NULL, 721, 1069, 1324
NOT_OK, 712, 1060
NULL, 341
Null instruction, 1657, 1658, 4897, 4898

O

O, 1484, 1489, 4710, 4715
O(, 1492, 4718
OD, 1633, 4865
OF, 1834, 4965
Off delay, 627, 639, 658, 668, 976, 1007, 1017, 1294, 1518, 1670, 1689, 1914, 4207, 4220, 4229, 4417, 4430, 4441, 4616, 4747, 4905, 4917, 5054
OK, 711, 1059
ON, 1485, 4712
On delay, 624, 636, 653, 664, 973, 1002, 1013, 1292, 1515, 1667, 1684, 1912, 4205, 4215, 4225, 4415, 4425, 4437, 4613, 4743, 4902, 4913, 5052
  Retentive, 655, 666, 1005, 1015, 1517, 1687, 4217, 4227, 4428, 4439, 4745, 4915
ON(, 1493, 4719
Ones complement, 897, 1243, 1578, 4321, 4534, 4811
OP, 4306
Open User Communication, 6186
  Changing IP configuration parameters with IP_CONF, 6226
  Changing IP configuration parameters with T_CONFIG, 3612
  Configuring connections with TCP and ISO on TCP, 6191
  Configuring the local communications access point with UDP, 6196
  Establishing a communication connection with TCON, 3538, 3541
  Establishing a connection and reading data with TRCV_C, 3492, 3497
  Establishing a connection and sending data with TSEND_C, 3478, 3483
  Form of the address information for the remote partner with UDP, 6198
  Functional description of instructions, 6187
  FW_IOT, 6236
  FW_TCP, 6232
  Receiving data via UDP with TURCV, 3583, 6223
  Receiving data with TRCV, 3557, 3561, 6216
  Sending data via UDP with TUSEND, 3579, 6220
  Sending data with TSEND, 3550, 3553, 6213
  TCON, 6208
  Terminating a communication connection with TDISCON, 3548
  Terminating a local communication access point with TDISCON, 6212
  via Industrial Ethernet, 6186

Opening a force table, 7773
Opening a watch table, 7744
Operand, 89, 90, 95, 97, 98, 100, 101, 103, 104, 106, 108, 109, 110, 111, 270, 277, 281, 289
  Inserting, 7159, 7222
  Pasting, 7266
Operating hours counters
  Handling with RTM, 2121, 5185
  Setting, starting, stopping and reading with RTM, 5185
Operating mode
  For testing GRAPH programs, 7700
  Testing in automatic mode, 7704
  Testing in manual mode, 7704
  Testing in semiautomatic mode, 7704
Operating principle
  ABS_CTRL, 5900
  ABS_CTRL_451, 5904
  ABS_DIAG, 5902
  ABS_DIAG_451, 5906
  CAM_CTRL, 5933
  CAM_CTRL_452, 5936
  CAM_DIAG, 5934
  CAM_DIAG_452, 5937
  CNT_CTL1, 5872
  CNT_CTL2, 5874
  CNT_CTRL, 5870
  CNT2_CTR, 5886
  CNT2RDPN, 5886
  CNT2WRPN, 5885
  DIAG_INF, 5871
  DIAG_RD, 5887
  High_Speed_Counter, 2770
  MC_Control, 5824
  MC_GearIn, 5825
  MC_Home, 5817
  MC_Init, 5808
  MC_MoveAbsolute, 5809
  MC_MoveJog, 5814
  MC_MoveRelative, 5812
  MC_Simulation, 5825
  MC_StopMotion, 5822
  SSI_Absolute_Encoder, 2781
Operating system, 35
OPN, 1614, 4519, 4846
OPNDI, 1615, 4847
OPNI, 4307, 4520
Optimized access, 55, 57
Optimized block access, 194
OR, 173, 894, 944, 946, 1240, 4318, 4398, 4399, 4531, 7304
Order
  Job execution, 5918, 5920
Organization block (OB)
  Activating time-of-day interrupt organization blocks from user program, 5371
  Assigning event with ATTACH, 2433
  Canceling event assignment with DETACH, 2437
  Creating, 6908
  Cyclic interrupt OB, querying parameters with QRY_CINT, 2445
  Cyclic interrupt OB, setting parameters with SET_CINT, 2443
  Determining program runtime with OB_RT, 5506
  exporting to an external source file, 6938
  Function, 39
  Measuring runtime with RT_INFO, 2582
  Reading start information with RD_SINFO, 2571, 5442
  Start information, 39
Organization blocks (OB)
  Activating time-delay interrupt organization blocks from user program, 5374
  Canceling time-delay interrupt organization blocks from user program, 5375
  Querying time-delay interrupt organization blocks from user programs, 5376
OS, 173, 178, 179, 180, 186
OUT_RANGE, 709, 1057
Output
  Inserting, 7155, 7218
  remove, 7156, 7219
Output driver
  Call, 5855
  Error handling, 5855
  Function, 5855
  Initialization, 5855
  Simulation mode, 5856
Output parameters
  High_Speed_Counter, 2777
  SSI_Absolute_Encoder, 2785
Output parameters of GRAPH function block, 7397
Output pulse sequence
  with preset frequency, 2645
OutputCPU314C, 5858
  Call, 5858
  Description, 5858
  Parameter, 5858
OutputET200S2AO, 5857
  Call, 5857
  Description, 5857
  Parameter, 5858

OutputIM174, 5861
Call, 5862
Configuration of IM 174 (hardware configuration), 5861
Description, 5861
Parameter, 5862
OutputIM178, 5856
Call, 5856
Description, 5856
Digital outputs, 5856
Parameter, 5856
OutputMM4_DP, 5860
Call, 5861
Description, 5860
Parameter, 5861
OutputSINAMICS, 5863
Call, 5866
Configuration of the SINAMICS in HWCN, 5863
Description, 5863
Parameter, 5866
OutputSM332, 5859
Call, 5859
Description, 5859
Parameter, 5859
OutputSM432, 5857
Call, 5857
Description, 5857
Parameter, 5857
OutputUniversal, 5859
Call, 5860
Description, 5859
Operating principle, 5859
Parameter, 5860
OV, 173, 178, 179, 180, 182, 186, 190
Overflow, 173, 186
Overlaying
Tag, 111
Overlaying tags, 6983
OVERRIDE
Input parameters, 5766
Instruction, 5764
Output parameters, 5766
Overview
using the display formats in the force table, 7777
via the display formats in the watch table, 7750
Overview of the cross-reference list, 7658
OW, 1628, 4860

P

P, 611, 614, 960, 963, 4197, 4407
P_RCV_RK, 6441
P_SND_RK, 6441
P_TRIG, 617, 965, 4200, 4410
P3964_Config (protocol configuration), 3944, 6258
PARA_CLT
Input parameters, 5767
Output parameters, 5768
PARA_CTL
Instruction, 5766
Parameter
ABS_CTRL_451, 5905
CNT_CTL1, 5872
CNT_CTRL, 5870
DIAG_INF, 5872
EncoderAbsSensorDP:, 5841
EncoderCPU314C, 5847
EncoderET200S1Count, 5843
EncoderET200S1SSI, 5839
EncoderFM350, 5845
EncoderFM450, 5836
EncoderIM174, 5850
EncoderIM178, 5834
EncoderSINAMICS, 5852
EncoderSM338, 5842
EncoderUniversal, 5854
ErrorID, 2780, 2786
Hidden parameters, 7160, 7223
MC_Control, 5824
MC_GearIn, 5830
MC_Home, 5821
MC_MoveAbsolute, 5811
MC_MoveJog, 5816
MC_MoveRelative, 5813
MC_StopMotion, 5823
OutputCPU314C, 5858
OutputET200S2AO, 5858
OutputIM174, 5862
OutputIM178, 5856
OutputMM4_DP, 5861
OutputSINAMICS, 5866
OutputSM332, 5859
OutputSM432, 5857
OutputUniversal, 5860
Predefined - writing with WR_DPARM, 2431, 5365
Parameter configuration
LENGH and BUFFER for Send_P2P, 3949, 6263
Parameter DB
Configuration, 5947
Content, 5947
Purpose, 5947
Parameter DB FM x51
Configuration, 5914

Content, 5914
Purpose, 5914
Parameter passing, 76
Activating via registers, 7249
Parameter transfer, 78, 80, 83, 85, 86, 87
Transfer parameter as copy or as pointer, 80
Parameter types, 348, 3912, 6748
Parameters, 6126
ABS_DIAG_451, 5906
ABS_INIT, 5899
CAM_CTRL_452, 5936
CAM_DIAG_452, 5938
CAM_INIT, 5932
CAM_MSRM_452, 5939
CNT2_CTR, 5887
CNT2RDPN, 5886
CNT2WRPN, 5885
DIAG_RD, 5887
Dynamically - transferring with WR_PARM, 5364
Internal - Changing settings, 7420
S7 basic communication, 6126
PARM_MOD, 5360
Password legitimation, 871, 1218, 1430, 1846, 2011
PE_CMD, 2373, 5310
PE_DS3_Write_ET200S, 2378, 5314
PE_End_RSP, 2409, 5344
PE_Error_RSP, 2407, 5342
PE_Get_Mode_RSP, 2412, 5347
PE_I_DEV, 2403, 5338
PE_List_Modes_RSP, 2411, 5345
PE_Measurement_List_RSP, 2417, 5352
PE_Measurement_Value_RSP, 2418, 5353
PE_Mode_ID, 6570, 6594
PE_PEM_Status_RSP, 2414, 5349
PE_Start_End, 2368, 5306
PE_Start_RSP, 2408, 5343
PE_WOL, 2379, 5315
PEEK, 1385, 1778
Read memory address, 1385, 1778
PEEK_BOOL, 1386, 1780
Read memory bit, 1386, 1780
Performance, 193
Permanent instruction
Inserting, 7477
Introduction, 7388
Permitted operands for the force table, 7775
Permitted operands for the watch table, 7746
PID
Input parameters, 5771
Instruction, 5768
Operating principle, 5769
Output parameters, 5772
PID_3Step
In/out parameters, 2856
Input parameters, 2852, 2882
Instruction, 2843, 2875
Output parameters, 2854, 2883
Static tags, 2885
PID_Compact
In/out parameters, 2799
Input parameters, 2796, 2827
Instruction, 2824
Output parameters, 2798, 2828
Static tags, 2829
PID_CP
Description, 5650
In/out parameters, 5669
Input parameters, 5666
Output parameters, 5668
PID_ES
Output parameters, 5696
PID_FM
Block diagrams, 5972
In/out parameters, 5968
Input parameters, 5965
Instruction, 5962
Operating principle, 5963
Output parameters, 5966
PID_FM_455
Block diagrams, 6051
In/out parameters, 6047
Input parameters, 6044
Instruction, 6041
Mode of operation, 6042
PID_PAR
Changeable parameters, 5991
Input parameters, 5990
Instruction, 5988
Output parameters, 5991
PID_PAR_455
Alterable parameters, 6069
Input parameters, 6069
Instruction, 6067
Output parameters, 6069
PID_Temp
ActivateRecoverMode tag, 2960
Cascade, 2919
ErrorBits parameter, 2958
In/out parameters, 2919
Input parameters, 2915
Mode, 2919
Operating principle, 2909
Output parameters, 2916
PID_Temp state and mode parameters, 2950

PwmPeriode, 2964
Tag Warning, 2963
PING command, 6522
PIP_Mode, 3442
PLC alarms with instructions
Acknowledgeable with ALARM_DQ, 5425
Always acknowledged with ALARM_D, 5423
Creating, 5398
Determining acknowledgement status with ALARM_SC, 5427
Disabling with DIS_MSG, 5436
Enabling with EN_MSG, 5434
Error behavior of instructions, 5415
Overview, 5416
Reading dynamically allocated system resources with READ_SI, 5438
Releasing dynamically allocated system resources with DEL_SI, 5440
Startup characteristics of instructions, 5415
System resources with ALARM_SQ and ALARM_S, 5418, 5421
With acknowledgement display with ALARM, 5412
With associated values for eight signals with ALARM_8P, 5406
Without acknowledgement display with NOTIFY, 5409
Without acknowledgement display with NOTIFY_8P, 5402
Without associated values for eight signals with ALARM_8, 5404
PLC data type, 220
Addressing, 277
Change the number, 7110
Comparing, 7619
Creating, 7108
Declaration in data blocks, 7033
Declaration in the block interface, 6982
Declaration table for PLC data types, 7105
Declaring, 7114
Declaring ARRAY, 7111
Declaring STRUCT, 7113
Definition, 273
Deleting, 7109
Programming the structure, 7111
Tag properties, 7115, 7116
PLC data types, 195
PLC tag
Automatically filling in cells, 7097
Comparing, 7616
Comparing a PLC tag table, 7588
Comparison, 7585
Copying, 7096
Declaring, 7079, 7080, 7084, 7086
Definition, 96
Importing and exporting, 7098, 7099, 7100, 7101
Monitor, 7095
Offline/offline comparison, 7588
Permissible addresses and data types, 7075
PLC tag table, 7069, 7070, 7076, 7077
Properties, 7091
Retentive behavior, 7081, 7082
Rules, 7075
Sorting rows, 7097
PNIO_ALARM instruction
Formal parameter, 6567
PNIO_RECV, 6554
PNIO_RECV instruction
Formal parameter, 6555
PNIO_RW_REC instruction
Formal parameter, 6563
PNIO_SEND, 6547
PNIO_SEND instruction
Formal parameter, 6548
Pointer,
ANY, 344
POINTER, 341
VARIANT, 324
Point-to-point connection
Configuring communication parameters with PORT_CFG, 4044
Configuring receive parameters with RCV_CFG, 4049
Configuring serial transmission parameters with SEND_CFG, 4047
Deleting receive buffer with RCV_RST, 4061
Enable receiving with RCV_PTP, 4059
Initiating data transfer with SEND_PTP, 4057
Querying the signal state with SGN_GET, 4062
Setting output signals with SGN_SET, 4063
Point-to-point programming, 3929, 6244
POKE, 1388, 1781
Write memory address, 1388, 1781
POKE_BLK, 1391, 1785
Write memory area, 1391, 1785
POKE_BOOL, 1390, 1783
Write memory bit, 1390, 1783
POP, 1653, 4885
PORT_CFG, 4044
Port_Config (port configuration), 3934, 6249
Pragmas in SCL, 7355
Procedure for monitoring structures, 7696

Process data
AA_Data, 3432
SA_Data, 3425
Process image
Reading inputs with GETIO, 2252, 5228
Reading the process image area with GETIO_PART, 2261, 5230
Synchronizing the inputs with SYNC_PI, 2203, 5221
Synchronizing the outputs with SYNC_PO, 2205, 5222
Transferring the process image area with SETIO_PART, 2263, 5231
Updating inputs with UPDAT_PI, 2197, 5218
Updating the outputs with UPDAT_PO, 2200, 5219
Writing outputs with SETIO, 2253, 5229
Process interrupt
Triggering for DP master with DP_PRAL, 5279
PROFIenergy
Control PROFIenergy commands in the I-Device, 2403, 5338
Description, 2366, 5303
Generate answer to command at end of pause, 2409, 5344
Generate answer to command at start of pause, 2408, 5343
Generate list of supported measured values as answer, 2417, 5352
Generate negative answer to command, 2407, 5342
Generate PEM status as answer, 2414, 5349
Generate queried energy data as answer, 2412, 5347
Generate queried energy savings modes as answer, 2411, 5345
Generate queried measured values as answer, 2418, 5353
Generate supported PROFIenergy commands as answer, 2415, 5350
PE_CMD, 2373, 5310
PE_DS3_Write_ET200S, 2378, 5314
PE_End_RSP, 2409, 5344
PE_Error_RSP, 2407, 5342
PE_Get_Mode_RSP, 2412, 5347
PE_I_DEV, 2403, 5338
PE_Identify_RSP, 2415, 5350
PE_List_Modes_RSP, 2411, 5345
PE_Measurement_List_RSP, 2417, 5352
PE_Measurement_Value_RSP, 2418, 5353
PE_PEM_Status_RSP, 2414, 5349
PE_Start_End, 2368, 5306
PE_Start_RSP, 2408, 5343
Reading out status information, 2373, 5310
Set power module switching behavior, 2378, 5314
Start and exit energy-saving mode, 2368, 2373, 5310
Starting and stopping energy-saving mode, 5306
Wake on LAN, 2379, 5315
PROFIenergy controller, 6569
PROFIenergy device, 6569
PROFINET CBA
Updating inputs of the program interface with PN_IN, 6791
Updating interconnections with PN_DP, 6793
Updating outputs of the PROFINET interface with PN_OUT, 6792
PROFINET IO devices
Reading a portion of the inputs with GETIO_PART, 2261, 5230
Reading consistent data with DPRD_DAT, 2338, 5262
Writing a part of the outputs with SETIO_PART, 2263, 5231
Writing all outputs with SETIO, 2253, 5229
Writing consistent data with DPWR_DAT, 2340, 5265
PROFINET CBA component
Updating with instructions, 6789
Program
Determining runtime with OB_RT, 5506
Program alarm
Creating, 7503, 7511
Deleting, 7504, 7513
Editing, 7503, 7504, 7512
generating with Program_Alarm, 2522
Program control operations, 878, 1225, 1437, 1853, 2017, 4310, 4523, 4645, 4979, 5100
Program display, 1657, 4897
Program editor
"Instructions" task card, 6833
Block interface, 6833
Enlarging the programming window, 6841
Favorites, 6833
Function, 6831
General settings, 6861
Programming window, 6833
Structure, 6832
Testing task card, 6834
Toolbar, 6833
Using the keyboard, 6836
Program information
Displaying, 7623

In the assignment list, 7624
In the call structure, 7633
In the dependency structure, 7640
in the tab, 7646
Views, 7623
Program loop, 1612, 4845
Program resources, 6907
Program status
Call environment, 7689, 7690
Disable all steps, 7703
Editing blocks, 7697
FBD, 7709
Function, 7687
GRAPH, 7700, 7703, 7704, 7712
Initialize sequencer, 7703
LAD, 7708
Modify tag, 7698
Operating modes, 7700, 7704
SCL, 7711
STL, 7710
Switching off, 7692
Switching on, 7692
System synchronization, 7700, 7706
Program_Alarm, 2522
Programming
Linear, 37
PtP instructions, 3929, 6244
Structured, 37
Programming language
Change over, 7416
Changeover, 6867
FBD, 7185
GRAPH, 7369
LAD, 7121
Rules for changing, 6867
Programming recommendations, 197
Programming window
Customizing, 7320
GRAPH, 7421
Permanent post-instructions, 7428
Permanent pre-instructions, 7423
SCL, 7319
Sequence view, 7424
Single step view, 7426
Project tree
Hiding, 6841
Showing, 6841
PROTECT, 4312, 4524, 4647, 4980, 5101
Protection level
Changing, 4312, 4524, 4647, 4980, 5101
PRVREC, 2344
PT, 994
PtP communication
Programming, 3929, 6244
PtP error classes, 3934, 6248
PtP instructions return values, 3933, 6247
Pulse, 622, 633, 647, 660, 970, 996, 1010, 1290, 1511, 1664, 1679, 1909, 4203, 4209, 4222, 4413, 4419, 4433, 4611, 4739, 4899, 4908, 5049
Extended, 650, 662, 999, 1012, 1513, 1682, 4212, 4224, 4422, 4435, 4741, 4910
Pulse generator
Enabling/disabling with CTRL_PWM, 2643
Pulse width modulation with PULSE, 6123
PULSEGEN
Input parameters, 5580
Instruction, 5571
Mode of operation, 5573
Output parameters, 5581
PULSEGEN_M
Input parameters, 5779
Instruction, 5772
Operating principle, 5773
Output parameters, 5780
PUSH, 1653, 4884
PUT, 3452

Q

QB, 7073
QD, 7073
QRY_CINT, 2445
QRY_DINT, 2471
QRY_TINT, 2459
Query architecture, 3931, 6246
Query architecture master, 3932, 6246
Query architecture slave, 3932, 6247
QW, 7073
QX, 7073

R

R, 604, 952, 1497, 1510, 1524, 4193, 4403, 4724, 4738, 4753
R_TRIG, 619, 968, 1288, 1659
RALRM, 2273
Range
Set in the I/O area, 4347
Value outside range, 709, 1057
Value within range, 708, 1056
RCV_CFG, 4049
RCV_PTP, 4059, 6768
RCV_RST, 4061

RCVREC, 2342
RD_ADDR, 2748
RD_DPAR, 2421
RD_DPARA, 2427
RD_DPARM, 2429
RD_LGADR, 2754
RD_LOC_T, 2105
RD_REC, 2332
RD_SINFO, 2571
RD_SYS_T, 2103
RDREC, 2213
RE_TRIGR, 877, 1224, 1436, 1853, 2016, 4310, 4522, 4645, 4978, 5100
Read
    Feedback signals, 5900, 5904
Read field, 831, 1179
Read job
    CNT2RDPN, 5898
Read jobs, 5919, 5921
Read memory address, 1385, 1778
Read memory bit, 1386, 1780
Read system time with TIME_TCK, 2120, 5180
READ_355
    Input parameters, 5983
    Instruction, 5982
    Output parameters, 5984
READ_455
    Input parameters, 6062
    Instruction, 6062
    Output parameters, 6063
READ_BIG, 1397, 1791
READ_DBL, 2714
READ_ERR, 2487
READ_LITTLE, 1393, 1787
ReadFromArrayDB, 812, 1161, 1375, 1769, 1978
ReadFromArrayDBL, 817, 1166, 1379, 1773, 1983
Reading
    A data record with RD_REC, 2332, 5256
    Asynchronously- data record of a module with RD_DPARA, 2427, 5359
    Data record of a module with RD_DPAR, 2421, 5357
    Dynamically allocated system resources with READ_SI, 5438
    from a data block in load memory with READ_DBL, 2714, 5514
    Reading the local time with LOC_TIME, 5189
    Reading the runtime meter with READ_RTM, 5188
    SZL partial list with RDSYSST, 5444
Reading data
    From DP standard slaves/PROFINET IO devices with DPRD_DAT, 2338, 5262
    From remote CPU with GET, 3447
    outside of the local S7 station with X_GET, 6799
    within the local S7 station with I_GET, 6784
Reading out
    LED status with LED, 2587
    Local time with RD_LOC_T, 2105
REAL, 251, 378, 449, 502, 539, 563, 583, 7308
REAL_TO_, 449, 539, 583
Receive_Config (receive configuration), 3939, 6253
Receive_P2P (receive point-to-point data), 3949, 6263
Receive_Reset (reset receiver), 3951, 6265
Receiving data
    and storing with RCV_PTP, 6768
    and supplying with SERVE_RK, 6780
    from outside of the local S7 station with X_RCV, 6795
    on I-device with RCVREC, 2342, 5268
Receiving SMS messages (TC_RECV), 3877, 6713
Recipe
    Exporting to CSV file, 2649
    Importing into data block, 2651
RecipeExport, 2649
RecipeImport, 2651
ReconfigIOSystem, 2315
Reconfiguring IO systems, 2315
References, 195
Regions in SCL, 7331, 7334, 7335, 7337, 7338
Remainder of division, 734, 1082, 1542, 4258, 4470, 4773
REPEAT, 1840, 4971
REPL_VAL, 4356, 4569, 4664, 5005, 5125
REPLACE, 2177
Replacing a module, 6501, 6642
RES_RCVB, 6770
Reset,
    Bit array, 4351, 4564, 4660, 5001, 5121
    Bit field, 607, 956
    Byte array, 4354, 4662, 5003, 5124
    Counter operations, 1524, 4753
    I/O area, 4352, 4565, 4661, 5002, 5122
    IEC timer, 644, 1299, 1676
    Operand, 604, 610, 952, 958, 1497, 4193, 4196, 4403, 4406, 4724
    RLO, 1500, 4728
    Set byte array, 4567
    Timer, 1510, 4738
Reset timer, 993
RESET_BF, 607, 956

RESETI, 4354, 4567, 4662, 5003, 5124
RESETP, 4352, 4565, 4661, 5002, 5122
Resetting the connection, 3597
Resources, 7646
    Code work memory, 7646
    Data work memory, 7646
    Introduction, 7646
    Load memory, 7646
    Retentive memory, 7646
    Retentive memory data, 7646
    Work memory, 7646
Resources tab
    Structure, 7648
Result of logic operation, 173, 186
RET, 869, 1217, 4306, 4518
RET_VAL for asynchronous instructions, 4186
Retain, 57
Retentive behavior
    Bit memory, timers, counters, 7082
Retentive bit memories
    Enabling the display, 7632
Retentive memory, 7646
Retentivity, 55
    Bit memory, timers, counters, 7081
    Block interface, 6971
    Data block, 7037, 7038
    PLC tag, 7081, 7082
RETURN, 1844, 4975
Return values
    ABS_CTRL, 5901
    ABS_CTRL_451, 5905
    ABS_DIAG, 5903
    ABS_DIAG_451, 5907
    ABS_INIT, 5899
    CAM_CTRL, 5933
    CAM_CTRL_452, 5937
    CAM_DIAG, 5935
    CAM_DIAG_452, 5938
    CAM_INIT, 5932
    CAM_MSRM_452, 5939
    PtP instructions, 3933, 6247
Return values receive runtime, 3949, 6263
RIGHT, 2170
Ring buffer, 336
RLD, 1647, 4878
RLDA, 1650, 4881
RLO, 173, 186
    Invert, 602, 950, 4192, 4402
    Inverting, 1499, 4726
    Reset to 0, 1500, 4728
    set to 1, 1500, 4727
RMP_SOAK
    Input parameters, 5786
    Instruction, 5780
    Output parameters, 5786
RND, 1584, 4817
RND-, 1587, 4820
RND+, 1586, 4819
ROC_LIM
    Input parameters, 5793
    Instruction, 5788
    Output parameters, 5794
ROL, 915, 1261, 1880, 2039, 4331, 4545, 4995, 5115
Root, square, 1720, 4937
ROR, 913, 1259, 1878, 2038, 4330, 4543, 4993, 5114
Rotate
    Left, 915, 1261, 1647, 1648, 1650, 1880, 2039, 4331, 4545, 4878, 4880, 4881, 4995, 5115
    Right, 913, 1259, 1651, 1878, 2038, 4330, 4543, 4882, 4993, 5114
Round, 847, 848, 850, 851, 1194, 1196, 1197, 1199, 1584, 1585, 1586, 1587, 1811, 1812, 1814, 1998, 1999, 2000, 2001, 4293, 4294, 4296, 4297, 4506, 4507, 4508, 4510, 4817, 4818, 4819, 4820, 4955, 4956, 4957, 5090, 5091, 5092, 5093
RRD, 1648, 4880
RRDA, 1651, 4882
RS, 610, 958, 4196, 4406
RT, 644, 993, 1299, 1676
RT_INFO, 2582
RTM, 351, 2121
Rules
    STL network, 7259
Rules for loop monitoring with S7-1200/1500, 7694
Rules for loop monitoring with S7-300/400, 7694
Rules for monitoring structures (S7-1200/1500), 7695
Rung
    Deleting, 7170
    Inserting, 7170
    Use, 7169
RUNTIME, 890, 1236, 1448, 1864
    Measure program runtime, 890, 1236, 1448, 1864
    Measuring with RT_INFO,
Runtime meters
    Properties, 5184
    Range of values, 5184
    Reading with READ_RTM, 5188
    Setting with SET_RTM, 5186
    Starting and stopping with CTRL_RTM, 5187

RW_REC, 6562

S

S, 605, 953, 1498, 1525, 4194, 4404, 4725, 4754
S_AVERZ, 658, 4220
S_CD, 680, 1030, 4239, 4451
S_COMP, 2126
S_CONV, 2128
S_CU, 678, 1028, 1703, 4237, 4449, 4926
S_CUD, 1032, 4452
S_EVERZ, 653, 4215
S_MOVE, 2124
S_ODT, 653, 1002, 1684, 4215, 4425, 4913
S_ODTS, 655, 1005, 1687, 4217, 4428, 4915
S_OFFDT, 658, 1007, 1689, 4220, 4430, 4917
S_PEXT, 650, 999, 1682, 4212, 4422, 4910
S_PULSE, 647, 996, 1679, 4209, 4419, 4908
S_SEVERZ, 655, 4217
S5TIME, 255, 381, 455, 565, 585
S5TIME_TO_, 455, 585
S7 basic communication
    Common parameters, 6126
    I_ABORT, 6787
    I_PUT, 6786
    X_ABORT, 6803
    X_GET, 6799
    X_PUT, 6801
    X_RCV, 6795
    X_SEND, 6794
S7 communication
    Change in operating mode, 6132
    Data exchange, 6132
    Determining the connection status with CONTROL, 6167
    Error behavior, 6140
    Overview of instructions, 6131
    Parameter, 3444
    Parameters, 6134
    Performing a hot restart with RESUME, 6180
    Performing a warm or cold restart with START, 6176
    Putting the device in the STOP operating mode with STOP, 6178
    Query connection, 6133
    Query operating mode, 6133
    Querying connection status with C_CNTRL, 6157
    Querying the device status with STATUS, 6181
    Read data from a remote CPU with GET_S, 6159
    Read data from remote CPU with GET, 6141
    Receive data from a remote partner instruction with BRCV, 3469, 6154
    Receive data from a remote partner instruction with URCV, 3460, 6149, 6165
    Receiving device status change with USTATUS, 6184
    Send and receive parameters, 3445, 6137
    Send data to a printer with PRINT, 6170
    Send data to a remote partner instruction with BSEND, 3466, 6151
    Send data to a remote partner instruction with USEND, 3458, 6146
    Send data to a remote partner instruction with USEND_S, 6163
    Startup characteristics, 6138
    User data size, 3446, 6137
    Write data to a remote CPU with PUT_S, 6161
    Write data to remote CPU with PUT, 6144
Safety precautions when forcing tags, 7769, 7786
SAVE, 1501, 4338, 4550, 4729
    RLO in BR bit,
Saving a force table, 7773
Saving a watch table, 7745
SC, 684, 1034, 4243, 4454
Scale, 852, 857, 1200, 1205, 1417, 1426, 1817, 1829, 2002, 2006, 4298, 4511, 4638, 4959, 5094
SCALE_M
    Input parameters, 5796
    Instruction, 5795
    Output parameters, 5796
SCALE_X, 852, 1200, 1417, 1817, 2002
Scan operand for negative signal edge, 613, 961, 4199, 4409
Scan operand for positive signal edge, 611, 960, 4197, 4407
Scan RLO for negative signal edge, 618, 966, 4202, 4412
Scan RLO for positive signal edge, 617, 965, 4200, 4410
SCL, 7293, 7360, 7362
    Copying and inserting regions, 7335
    Delete region, 7338
    Detailed comparison, 7603
    Insert comment, 1845, 4975
    Insert pragmas, 7355
    Inserting regions, 7334
    Navigating in regions, 7337
    Programming window, 7319
    Using regions, 7331
SCL instruction
    Changing the data type, 7340
    Data type basics, 7339
    Inserting, 7329, 7330

Rules, 7328
Selecting, 7358
SCL program
Inserting a block call, 7344, 7345, 7347, 7348, 7349, 7351
Inserting an instruction, 7329, 7330
Inserting comments, 7356
Program status display, 7711
Selecting an instruction, 7358
Showing and hiding the parameter list, 7354
SD, 662, 664, 1013, 1513, 1515, 4224, 4225, 4437, 4741, 4743
SE, 662, 1012, 1513, 4224, 4435, 4741
SEG, 937, 1283, 1478, 1901, 2064, 4393, 4607, 4705, 5041, 5160
SEL, 901, 1247, 1452, 1868, 2029, 4324, 4538, 4654, 4987, 5109
Select, 901, 1247, 1452, 1868, 2029, 4324, 4538, 4654, 4987, 5109
Send clock, 3442
SEND_CFG, 4047
Send_Config (send configuration), 3937, 6252
Send_P2P (send point-to-point data), 3946, 6260
LENGH and BUFFER parameters, 3949, 6263
SEND_PTP, 4057, 6765
SEND_RK, 6772
Sending data
outside of the local S7 station with X_SEND, 6794
With TSEND, 6213
Sending SMS messages (TC_SEND), 3873, 6709
Sequence end
Ending sequencers, 7437
Introduction, 7374
Sequence view, 7424
Sequencer, 917, 1263, 1453, 1882, 2041, 4357, 4363, 4570, 4576, 4665, 4672, 5006, 5013, 5126, 5132
Adding comments, 7435
Basics, 7371
Deleting, 7438
Elements, 7372
Ending, 7437
Initializing, 7703
Inserting, 7434
Inserting new elements, 7436
Renaming, 7434
Serialize, 771, 1120, 1341, 1733, 1945
SERVE_RK, 6780
Service_Request_ID, 6579
Set,
Bit array, 4346, 4559, 4655, 4997, 5117
Bit field, 606, 954
Byte array, 4349, 4658, 5000, 5120
Counter initial value, 4243
Counter operations, 1525, 4754
Counter start value, 684, 1034, 4454
I/O area, 4347, 4657, 4998, 5118
Operand, 605, 609, 953, 957, 1498, 4194, 4195, 4404, 4405, 4725
RLO, 1500, 4727
Set byte array, 4562
Set limit value, 744, 1092, 1331, 1717, 1930, 4265, 4477, 4627, 4935, 5067
Set operand on negative signal edge, 616, 964
Set operand on positive signal edge, 614, 963
SET_BF, 606, 954
SET_CINT, 2443
SET_CLKS, 5181
SET_TIMEZONE, 2110
SET_TINT, 2451
SET_TINTL, 2454
SETI, 4349, 4562, 4658, 5000, 5120
SETIO, 2253
SETIO_PART, 2263, 5231
SETP, 4347, 4560, 4657, 4998, 5118
Setting
View options for the assignment list, 7628
View options for the dependency structure, 7643
Setting ENO automatically, 7317
Setting the font, size and color, 7320
Setting the mnemonics, 6843
Setting the tab spacing, 7320
Setting time-of-day status with SET_CLKS, 5181
Settings
Changing, 6864, 7124, 7188, 7250, 7318, 7414
FBD, 7124, 7188
General, 6861
GRAPH, 7412
LAD, 7124, 7188
SCL, 7317
STL, 7250
Settings in the cross-reference list, 7655
Seven-segment display, 937, 1283, 1901, 4393, 4607, 4705, 5041, 5160
SF, 668, 1017, 1518, 4229, 4441, 4747
SGN_GET, 4062
SGN_SET, 4063
Shadow memory, 6789
Shared PtP parameter errors, 3934, 6248

Shift
Left, 911, 1257, 1640, 1643, 1876, 2036, 4328, 4541, 4871, 4875, 4992, 5112
Right, 909, 1255, 1637, 1639, 1642, 1645, 1875, 2035, 4326, 4539, 4868, 4870, 4873, 4876, 4990, 5111
Shift register, 4388, 4390, 4600, 4603, 4700, 4703, 5037, 5039, 5155, 5158
SHL, 911, 1257, 1876, 2036, 4328, 4541, 4992, 5112
Show line numbers, 7320
Showing and hiding absolute operands, 7320
Showing and hiding columns, 6989, 7049, 7098, 7120
SHR, 909, 1255, 1875, 2035, 4326, 4539, 4990, 5111
SHRB, 4390, 4603, 4703, 5039, 5158
Shut down (PC system), 876, 1223, 1435, 1852
Shut down target system, 876, 1223, 1435, 1852
SHUT_DOWN, 876, 1223, 1435, 1852
Sign, 739, 1087, 1546, 1927, 4261, 4472, 4778, 5063
Signal_Get (get RS232 signals), 3952, 6266
Signal_Get (set RS232 signals), 3954, 6268
Simultaneous branch
Creating, 7448
Introduction, 7372
SIN, 751, 1099, 1550, 1723, 1935, 4272, 4484, 4782, 4940, 5071
Sine, 751, 1099, 1550, 1723, 1935, 4272, 4484, 4782, 4940, 5071
Single instance
Correcting the call type, 7146, 7209, 7274
Definition, 63
Example, 63
Single step mode
Basics, 7716, 7734, 7736, 7738
Deleting breakpoints, 7730
Disable breakpoint, 7726, 7728, 7733
Enable breakpoint, 7733
Enable breakpoint., 7726, 7728
Navigating to the breakpoints, 7729
Resume program execution, 7732
Resume run to cursor, 7732
Setting breakpoints, 7724
Setting the call environment for breakpoints, 7722
Step into, 7732
Step over, 7732
Single step view, 7426
SINT, 241, 364, 421, 496, 521
SINT_TO_, 421, 521
SLD, 1643, 4875
Slice access, 110
Slicing, 196
Slot, 2738
Determining logical address with LGC_GADR, 2757, 5556
Determining logical address with LOG_GEO, 2753, 5551
SLW, 1640, 4871
SMC, 932, 1278, 1471, 1897, 2057, 4380, 4593, 4691, 5030, 5148
SMS text (TC_RECV), 3879, 6715
SMS text (TC_SEND), 3875, 6711
SMS text (TCON_PHONE), 3885, 6721
SNC_RTCB, 2119
SP, 660, 1010, 1511, 4222, 4433, 4739
SP_GEN
Input parameters, 5798
Instruction, 5796
Output parameters, 5798
SPLIT, 2152
SPLT_RAN
Input parameters, 5800
Instruction, 5798
Output parameters, 5800
SQR, 746, 1094, 1547, 1719, 1932, 4267, 4479, 4779, 4936, 5068
SQRT, 748, 1096, 1548, 1720, 1933, 4268, 4480, 4780, 4937, 5069
Square, 746, 1094, 1547, 1719, 4267, 4479, 4779, 4936, 5068
Square root, 748, 1096, 1548, 1720, 4268, 4480, 4780, 4937, 5069
SR, 609, 957, 4195, 4405
SRD, 1645, 4876
SRT_DINT, 2469
SRW, 1642, 4873
SS, 666, 1015, 1517, 4227, 4439, 4745
SSD, 1639, 4870
SSI, 1637, 4868
SSI_Absolute_Encoder, 2780
Call, 2780
Description, 2780
Error response, 2783
Input parameters, 2783
Operating principle, 2781
Output parameters, 2785
Static tags, 2786
STA, 173
Start information, 39
Start off-delay timer, 987
Start on-delay timer, 985

Start pulse timer, 982
Startup
ABS_DIAG, 5903
ABS_DIAG_451, 5907
CAM_DIAG, 5935
CAM_DIAG_452, 5938
CAM_MSRM_452, 5940
CPU, 5901, 5905, 5934, 5937
FM 351, 5901
FM 352, 5934, 5937
FM 451, 5905
FM 452, 5934, 5937
Instructions for generating PLC alarms,
Startup characteristics
CNT_CTL1, 5874
CNT_CTRL, 5871
Static parameters of GRAPH function block, 7400, 7403, 7404
Static tags
High_Speed_Counter, 2778
SSI_Absolute_Encoder, 2786
Station
Read information with GetStationInfo, 2600
Status, 173
Status bit, 173, 178, 179, 180, 182, 186, 190
Query, 183, 187, 4548
Status bits
Query, 184, 189
Status of the write job
CNT2RDPN, 5898
CNT2WRPN, 5897
Status word, 173, 178, 179, 180, 182, 183, 186, 187, 190
Step
Actions, 7377
Adding comments, 7444
Basics, 7375
Copying and inserting, 7444
Deleting, 7446
Disable all steps, 7703
Elements, 7376
Inserting, 7438
Interlock, 7376
Renaming, 7443
Renumbering, 7441
Skipping, 7445
Static parameters, 7400
Supervision, 7377
Step time monitoring
Basics, 7465
Determining T_MAX and T_WARN, 7468, 7707
Programming, 7466
Specifying T_MAX and T_WARN manually, 7467
STL, 7245, 7282
Detailed comparison, 7600
Parameter passing, 7249
STL instruction
Copying, 7277
Cutting, 7278
Deleting, 7279
Entering, 7262
Inserting an operand, 7266
paste from Clipboard, 7279
Pasting, 7263
Select, 7277
STL program
Comment section, 7275
Entering an instruction, 7262
Inserting a block call, 7270, 7271
Inserting an instruction, 7263
Line comment, 7275
Program status display, 7710
Rules, 7259
Select instructions, 7277
STL source
Generating blocks, 6941
Stop forcing, 7793, 7794
STP, 878, 1225, 1437, 1853, 2017, 4310, 4523, 4645, 4979, 5100
Strg_TO_Chars, 2140
STRG_VAL, 2131
STRING, 125, 264, 402, 483, 509, 510, 552, 554, 568, 590
Addressing, 270
Comparing tags with S_COMP,
Declaration in the block interface, 6981
STRING/WSTRING
Textual block interface, 7000
STRING_TO_, 483, 552, 554, 590
STRUCT
Addressing, 281
Declaration in global data blocks, 7031
Declaration in the block interface, 6980
Declaring in a PLC data type, 7113
Structure, 278
Textual block interface, 6995
Structure
Call structure, 7635
of the cross-reference list, 7653
Of the dependency structure, 7641
Resources tab, 7648
STRUCT, 278
Structured programming, 37

SUB, 730, 1078, 4254, 4466
Subfield types, 6609
Substitute value, 4356, 4569, 4664, 5005, 5125
Subtract, 730, 1078, 1533, 1537, 1544, 4254, 4466, 4763, 4767, 4775
    Subtract timers with T_SUB, 5176
Subtracting
    Subtracting timers with T_SUB, 2095
Supervision
    Acknowledging supervision errors, 7704
    Introduction, 7377, 7389
    Programming, 7464
SWAP, 811, 1159, 1767, 1976, 5077
Swap bytes, 811, 1976, 5077
Swapping, 4799, 4815, 4816, 4848
SWITCH, 866, 1214
    Input parameters, 5801
    Instruction, 5800
    Output parameters, 5801
Switching, 1567, 1582, 1583, 1616
Switching between basic mode and expanded mode in the force table, 7771
Switching between basic mode and expanded mode in the watch table, 7741
Symbolic addressing, 198
Symbolic programming
    Displaying absolute addresses, 6844
Symbolism, 193
Symbols
    In the assignment list, 7626
    In the call structure, 7634
    In the dependency structure, 7641
SYNC_PI, 2203
SYNC_PO, 2205
Synchronization
    From DP slaves with DP_SYC_FR, 2347, 5281
    Slave clocks with SNC_RTCB, 2119, 5182
Synchronizing user pages, 3838
Synchronizing WWW user pages, 3838
Synchronous errors
    Masking with MSK_FLT, 2485, 5384
    Querying the error register with READ_ERR, 2487, 5385
    Unmasking with DMSK_FLT, 2486, 5385
Syntax errors
    Basics, 6865
    Finding errors, 6865
System block
    System blocks folder, 6907
System constant, 7089
System data
    Parameter field for, 6608
System status list
    Definition, 5451
    Partial list extract SZL ID W#16#0132 index W#16#0005, 5474
    Partial list extract SZL ID W#16#0132 index W#16#0008, 5475
    Partial list extract SZL ID W#16#0132 index W#16#000B, 5477
    Partial list extract SZL ID W#16#0132 index W#16#000C, 5477
    Partial list extract SZL ID W#16#0232 index W#16#0004, 5478
    Partial lists, overview, 5454
    Reading partial SZL excerpt with RDSYSST, 5444
    Structure of a partial list, 5452
    SZL ID W#16#xy13 - user memory areas, 5458
    SZL ID W#16#xy25 - assignment of process image partitions to OBs, 5471
    SZL ID, structure, 5453
    SZL-ID W#16#00B1 - module diagnostic information, 5502
    SZL-ID W#16#00B2 - diagnostic data record 1 with physical address, 5504
    SZL-ID W#16#00B3 - module diagnostic data with logical base address, 5505
    SZL-ID W#16#00B4 - diagnostic data of a DP slave, 5505
    SZL-ID W#16#0x94 - status information for rack/station, 5491
    SZL-ID W#16#xy11 - module Identification, 5455
    SZL-ID W#16#xy12 - CPU characteristics, 5456
    SZL-ID W#16#xy14 - system areas, 5459
    SZL-ID W#16#xy15 - block types, 5460
    SZL-ID W#16#xy1C - component identification, 5461
    SZL-ID W#16#xy22 - interrupt status, 5468
    SZL-ID W#16#xy32 - communication status data, 5473
    SZL-ID W#16#xy74 - status of the module LEDs, 5479
    SZL-ID W#16#xy90 - DP master system information, 5481
    SZL-ID W#16#xy91 - module status information, 5482
    SZL-ID W#16#xy92 - rack / station status information, 5488
    SZL-ID W#16#xy95 - extended DP master system information, 5493
    SZL-ID W#16#xy96 - PROFINET IO and PROFIBUS DP module status information, 5494

SZL-ID W#16#xy9C - Tool changeover information (PROFINET IO), 5497
SZL-ID W#16#xyA0 - diagnostics buffer, 5501
System synchronization
For testing GRAPH programs, 7700
Performing, 7706

T

T, 1564, 4797
T STW, 1566, 4799
T_ADD, 2092
T_COMBINE, 2099
T_COMP, 2089
T_CONFIG, 3612
T_CONV, 2091
T_DIAG, 3603
T_DIFF, 2097
T_RESET, 3597
T_SUB, 2095
Table
Adding values with ATT, 5527
Calculating the standard deviation with DEV, 5541
Collecting / distributing data with PACK, 5547
Comparing table values with CDT, 5543
Copying values with TBL_WRD, 5536
Executing operations with TBL, 5534
Finding values with TBL_FIND, 5530
Linking values logically and storing with WRD_TBL, 5538
Linking with TBL_TBL, 5545
Outputting the first value with FIFO, 5528
Outputting the last value with LIFO, 5532
TADDR_Param, 3587
TADDR_RCV_IP, 3589
TADDR_SEND_QDN, 3589
Tag
Displaying or hiding tag information, 7161, 7224, 7267, 7341, 7478
Negative, 620, 969, 1289
Overlaying, 111
Overview, 95
PLC tag, 7069
PLC tags and DB variables, 96
Positive, 619, 968, 1288
Tag declaration
Automatically filling in cells, 6989, 7048, 7097, 7119
Based on a PLC data type, 7033
Block interface, 6974
Declare STRING, 6981
Declaring ARRAY, 6978
Declaring PLC data type, 6982
Declaring STRUCT, 6980
Declaring tags, 6975, 6977, 6983
Deleting a tag, 6988, 7047, 7097, 7118
Importing and exporting tags, 6990, 7049
Inserting a table row, 6988, 7047, 7096, 7118
Inserting table rows at the end, 6988, 7047, 7118
Multi-instance, 6983
Overlaying tags, 6983
Purpose of variable declaration, 6965
Reserved key words, 90
Retentivity, 6971
Showing and hiding columns, 6989, 7049, 7098, 7120
Sorting rows, 7097
Tag properties, 7009, 7011, 7040, 7043, 7044
Updating the block interface, 7015
Valid data types, 6968, 6970
Tags
Monitor all, 7757, 7783
Monitor now, 7784
Monitor once and now, 7758
TAK, 1652, 4883
Tan, 1725, 4942
Tangent, 754, 1102, 1554, 1725, 1937, 4274, 4487, 4786, 4942, 5073
TAR1, 1567, 4800
TAR1 <D>, 1568, 4801
TAR1 AR2, 1569, 4802
TAR2, 1570, 4803
TAR2 <D>, 1571, 4803
Task card
Find instructions, 6846
TCON, 3538, 3541
TCON_IP_RFC for telecontrol connections, 3883, 6718
TCON_IP_V4 for telecontrol connections, 3883, 6718
TCON_PHONE for telecontrol connections, 3883, 6718
TCON_WDC for telecontrol connections, 3883, 6718
TCONT_CP
In/out parameters, 5597
Input parameters, 5595
Instruction, 5581
Mode of operation, 5582
Output parameters, 5596
Static tags, 5597
TCONT_S
In/out parameters, 5610
Input parameters, 5609

Instruction, 5603
Operating principle, 5604
Output parameters, 5610
Static tags, 5610
TCP connection
Parameter field for, 6613
TDISCON, 3548
TeleService
AS_DIAL, 6808
AS_MAIL, 6820
Establish connection to AS, 6808
Establish connection to PG/PC, 6804
PG_DIAL, 6804
Send text (SMS) message, 6814
Sending e-mails, 6820
SMS_SEND, 6814
Transferring e-mail with TM_MAIL, 4178
Tens complement, 939, 1285, 1480, 1903, 2065, 4394, 4608, 4706, 5043, 5161
Test options in the force table, 7768
Test settings for GRAPH
Changing, 7703
Overview, 7702
Text lists for alarms, 7525
Textual block interface
ARRAY, 6998
Attributes for variables, 7002
Block declaration, 6992
Comments, 7003
Declaration and initialization of variables or constants, 6994
Declaration sections, 6993
Editing, 7004
Example, 7007
Instances, 7001
Restore default interface of an organization block, 7006
STRING/WSTRING, 7000
STRUCT, 6995
Structure, 6991
THEN, 1833, 4963
THIS, 106
Time, 255, 257
Switching from daylight saving time/standard time with SET_SW, 5194
Time accumulator, 630, 979, 990, 1297, 1673, 1916, 4385, 4598, 4698, 5035, 5153
Time delay, 887, 1234, 1445, 1861, 2024, 4311, 4523, 4646, 4979, 5101
Time duration, 645, 994, 1301, 1677
Time stamp
Transferring alarms with TIMESTMP, 5200
TIME_TCK, 2120
TIME_TO_, 457, 545, 584
Time-delay interrupt
Canceling with CAN_DINT, 2471, 5375
Instructions, 2467, 5373
Querying with QRY_DINT, 2471, 5376
Starting with SRT_DINT, 2469, 5374
Time-of-day, 258, 259, 261
Calculate local time with BT_LT, 5190
Calculating base time with LT_BT, 5191
Calculating local time with SET_TIMEZONE, 2110
Reading the system time of the CPU with TIME_TCK, 2120, 5180
Reading the time-of-day and date of the CPU with RD_SYS_T, 5179
Reading time-of-day and date of CPU with RD_SYS_T, 2103
Set time-of-day interrupt with S_LTINT, 5193
Setting for CPU with WR_SYS_T, 2101, 5178
Setting the time-of-day and time-of-day status with SET_CLKS, 5181
Synchronizing between slaves and master, 5182
Time-of-day interrupt
Activating with ACT_TINT, 2457, 5371
Canceling with CAN_TINT, 2456, 5370
Instructions, 5366
Properties of SET_TINT, CAN_TINT, ACT_TINT, QRY_TINT, 5367
Querying with QRY_TINT, 2459, 5371
Setting with SET_TINT, 2451, 2454, 5369
Time-of-day status
Adjusting with SET_SW_S, 5197
Timer, 256, 647, 996, 4209, 4419
Enable, 1505, 4733
Recording, 641
Reset, 1510, 4738
Timers
Adding with the instruction T_ADD, 2092, 5175
Combine date and time with T_COMBINE, 2099
Combining with T_COMBINE, 5177
Comparing time tags with T_COMP, 2089, 5173
Converting with T_CONV, 2091, 5174
Determining difference with T_DIFF, 2097, 5177
Subtracting with T_SUB, 2095, 5176
TIO_DI, 3429
TIO_DQ, 3436
TIO_IOLink_IN, 3425
TIO_IOLink_OUT, 3432
TIO_SYNC, 3422
TIO_Time
TIO_DI, 3429

TIO_DQ, 3436
TIO_IOLink_IN, 3425
TIO_SYNC, 3422
UDT "TIO_SYNC_Data", 3442
TM_MAIL, 4178
TMAIL_C, 3513
TMAIL_C V4.0, 3517
TO, 1836
TOD, 258, 395, 474, 506, 548, 567, 587
TOD_TO_, 474, 548, 587
TOF, 627, 639, 976, 987, 1294, 1670, 1914, 4207, 4417, 4616, 4905, 5054
TON, 624, 636, 973, 985, 1292, 1667, 1912, 4205, 4415, 4613, 4902, 5052
TONR, 630, 641, 979, 990, 1297, 1673, 1916, 5153
TONR_X, 4385, 4598, 4698, 5035
TP, 622, 633, 970, 982, 1290, 1664, 1909, 4203, 4413, 4611, 4899, 5049
Transfer, 1564, 1566, 1567, 1568, 1569, 1570, 1571, 4797, 4799, 4800, 4801, 4802, 4803
Transition
Copying and inserting, 7444
Deleting, 7446
Inserting, 7438
Introduction, 7377
Programming, 7469
Renaming, 7443
Renumbering, 7441
Static parameters, 7403
TRCV, 3557, 3561
TRCV_C, 3492, 3497
TRUE, 619, 620, 968, 969, 1288, 1289, 1659, 1660
TRUNC, 851, 1199, 1585, 1815, 2001, 4297, 4510, 4818, 4958, 5093
TSEND, 3550, 3553
TSEND_C, 3478, 3483
TUN_EC
In/out parameters, 5626
Input parameters, 5622
Instruction, 5613
Operating principle, 5617
Output parameters, 5624
TUN_ES
In/out parameters, 5644
Input parameters, 5640
Instruction, 5632
Operating principle, 5636
Output parameters, 5641
TURCV, 3583
TUSEND, 3579
Twos complement, 736, 1084, 1579, 1580, 1581, 1926, 4259, 4471, 4812, 4813, 4814, 5062
typed, 7313
TypeOf, 1708
TypeOfElements, 1709

U

UBLKMOV, 838, 1186, 1411, 1804, 1991, 4285, 4498, 4633, 4949, 5083
UC, 1625, 4857
UDINT, 246, 372, 439, 501, 536
UDINT_TO_, 439, 536
UDP connection
Parameter field for, 6614
UDT, 195, 220
UDT "TIO_SYNC_Data", 3442
UFILL_BLK, 789, 1138, 1356, 1749, 1959
UINT, 244, 368, 432, 499, 530
UINT_TO_, 432, 530
ULINT, 248, 376, 446
ULINT_TO_, 446
UMOVE_BLK, 783, 1131, 1351, 1743, 1954
Universal Serial Interface Protocol, (See USS protocol)
Universal symbolism, 193
UNSCALE, 859, 1207, 1428, 1831, 2008, 4301, 4513, 4641, 4961, 5096
UNTIL, 1840, 4971
UPDAT_PI, 2197
UPDAT_PO, 2200
Update
DP interconnections, 6789
PROFINET CBA components, 6789
Updating DP interconnections with PN_DP, 6793
URCV, 3460
USEND, 3458
User diagnostic alarm
Editing, 7506
User diagnostics alarm
Creating, 7506
Deleting, 7507
User program
Finding errors, 6865
Function, 35
Testing, 7685
Using the keyboard
Editing texts, 6836
FBD, 6836
GRAPH, 6836
LAD, 6836
Program editor, 6836
SCL, 6836
STL, 6836

Using UDTs, 220
USINT, 242, 365, 425, 497, 524
USINT_TO_, 425, 524
USS communication
    Controlling a transmission to a drive with USS_PORT, 4069
    Data exchange with drives via USS_DRIVE, 4070
    Modifying parameters in the drive with USS_WPM, 4074
    Reading parameters from the drive with USS_RPM, 4073
USS protocol
    Instructions, 4065
USS status codes, 4076
USS_DRIVE, 4070
USS_Drive_Control, 3972, 6286
USS_Drive_Control / USS_Drive_Control_31, 3979, 6293
USS_PORT, 4069
USS_Port_Scan, 3972, 3976, 6286, 6290
USS_Port_Scan_31, 3976, 6290
USS_Read_Param, 3972, 6286
USS_Read_Param / USS_Read_Param_31, 3983, 6297
USS_RPM, 4073
USS_WPM, 4074
USS_Write_Param, 3972, 6286
USS_Write_Param / USS_Write_Param_31, 3985, 6299

V

VAL_STRG, 2135
Value, 739, 1087, 1546, 4261, 4472, 4778
Value assignment, 7305
Value transfer
    CNT_CTL1, 5881
    DB parameters for count modes, 5882
    DB parameters for measuring modes, 5884
Variable index, 200
VARIANT, 324, 327, 328, 333, 336, 713, 714, 716, 718, 823, 825, 826, 1061, 1062, 1064, 1066, 1172, 1173, 1175, 1311, 1316, 1318, 1319, 1321, 1401, 1403, 1404, 1421, 1423, 1708, 1709, 1795, 1796, 1798, 1824, 1826
VARIANT_TO_DB_ANY, 1421, 1824
VariantGet, 823, 1172, 1401, 1795
VariantPut, 825, 1173, 1403, 1796
View options
    For the assignment list, 7628
    Setting the call structure, 7637
    Setting the dependency structure, 7643
VOID, 348

W

WAIT, 887, 1234, 1445, 1861, 2024, 4311, 4523, 4646, 4979, 5101
Wake on LAN, 2379, 5315
Watch table
    Basic mode, 7740
    Copying, 7744
    Creating, 7743
    Example of filling out a watch table, 7745
    Expanded mode, 7740
    Layout, 7740
    Meaning of the columns, 7740
    Meaning of the icons, 7742
    Monitoring and modifying modes, 7755
    Opening, 7744
    Overview of the display formats, 7750
    Overview of the test options, 7739
    Permitted operands, 7746
    Permitted operands for modify values, 7748
    Possible applications, 7739
    Saving, 7745
    Switching between basic mode and expanded mode, 7741
    Syntax check, 7745
    Testing wiring, 7739
WCHAR, 263, 400, 481
WCHAR_TO_, 481
WHILE, 1839, 4970
Wiring tests, 7739
WORD, 237, 359, 410, 494, 514, 560, 572
Word logic operations, 190
WORD_TO_, 410, 514, 572
WORD_TO_BLOCK_DB, 591
Work memory, 7646
WR_DPARM, 2431
WR_LOC_T, 2107
WR_REC, 2337
WR_SYS_T, 2101
WRIT_DBL, 2718
Write command
    CNT2WRPN, 5896
Write field, 833, 1181
Write jobs, 5918, 5920
Write memory address, 1388, 1781
Write memory area, 1391, 1785
Write memory bit, 1390, 1783
Write protection, 6956
WRITE_BIG, 1399, 1793
WRITE_LITTLE, 1395, 1789

WriteToArrayDB, 815, 1164, 1377, 1771, 1981
WriteToArrayDBL, 820, 1169, 1382, 1775, 1985
Writing
    A data record with WR_REC, 2337, 5260
    Control signals, 5900, 5904
Writing data
    in remote CPU with PUT, 3452, 6144
    outside of the local S7 station with X_PUT, 6801
    To DP standard slaves/PROFINET IO devices
    with DPWR_DAT, 2340, 5265
    within the local S7 station with I_PUT, 6786
WRREC, 2232
WSR, 4388, 4600, 4700, 5037, 5155
WSTRING, 125, 267, 403, 486
    Addressing, 270
WSTRING_TO_, 486

X

X, 1487, 4713
X(, 1494, 4720
XN, 1488, 4714
XN(, 1495, 4721
XOD, 1635, 4866
XOR, 895, 946, 948, 1242, 4319, 4399, 4400, 4532, 7303, 7304
XOW, 1630, 4861

Z

ZONE
    In/out parameters, 6081
    Input parameters, 6080
    Instruction, 6078
    Static variables, 6081, 6085

SIEMENS

WinCC
Visualizing processes

System Manual

Creating screens 1

Working with tags 2

Working with alarms 3

Working with logs 4

Access to archive data via WinCC OLE DB Provider (RT Professional) 5

Working with recipes 6

Working with reports (RT Professional, RT Advanced, Panels, Comfort Panels) 7

Configuring user administration 8

Working with system functions and Runtime scripting 9

Mobile Panels (Panels, Comfort Panels) 10

Planning tasks 11

Using the diagnostics functions 12

Communicating with controllers 13

Using global functions 14

Compiling and loading 15

Performance features 16

Options 17

Migrating to WinCC in the TIA Portal 18

Online help printout

10/2018
Online help printout

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Document order number: Online help printout
Ⓟ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Creating screens....37

1.1 Basics....37
1.1.1 Screen basics (RT Advanced, Basic Panels, Panels, Comfort Panels)....37
1.1.2 Screen basics (RT Professional)....39
1.1.3 Device-dependent functional scope of screens (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....40
1.1.4 Elements and basic settings....42
1.1.4.1 Task cards....42
1.1.4.2 Zooming the view....43
1.1.4.3 Moving the view....44
1.1.5 Working with screens....45
1.1.5.1 Steps (RT Advanced, Basic Panels, Panels, Comfort Panels)....45
1.1.5.2 Steps (RT Professional)....46
1.1.5.3 Creating a new screen (RT Advanced, Basic Panels, Panels, Comfort Panels)....46
1.1.5.4 Creating a new screen (RT Professional)....47
1.1.5.5 Managing screens (RT Advanced, Basic Panels, Panels, Comfort Panels)....48
1.1.5.6 Defining the start screen of the project (RT Advanced, Basic Panels, Panels, Comfort Panels)....50
1.1.5.7 Defining the start screen of the project (RT Professional)....51
1.1.5.8 Changing the screen resolution (RT Advanced)....52
1.1.6 Working with templates (RT Advanced, Basic Panels, Panels, Comfort Panels)....53
1.1.6.1 Basics on working with templates (RT Advanced, Basic Panels, Panels, Comfort Panels)....53
1.1.6.2 Global screen (RT Advanced, Basic Panels, Panels, Comfort Panels)....54
1.1.6.3 Creating a new template (RT Advanced, Basic Panels, Panels, Comfort Panels)....56
1.1.6.4 Configuring permanent areas in the template (RT Advanced, Panels, Comfort Panels)....57
1.1.6.5 Managing a template (RT Advanced, Basic Panels, Panels, Comfort Panels)....58
1.1.6.6 Using a template in the screen (RT Advanced, Basic Panels, Panels, Comfort Panels)....59
1.1.7 Working with pop-up screens (RT Advanced, Basic Panels, Panels, Comfort Panels)....60
1.1.7.1 Basics for working with pop-up screens (RT Advanced, Panels, Comfort Panels)....60
1.1.7.2 Creating a new pop-up screen (RT Advanced, Panels, Comfort Panels)....61
1.1.7.3 Managing pop-up screens (RT Advanced, Panels, Comfort Panels)....62
1.1.7.4 Using the pop-up screen (RT Advanced, Panels, Comfort Panels)....63
1.1.8 Working with slide-in screens (RT Advanced, Basic Panels, Panels, Comfort Panels)....64
1.1.8.1 Basics on working with slide-in screens (RT Advanced, Panels, Comfort Panels)....64
1.1.8.2 Configuring slide-in screens (RT Advanced, Panels, Comfort Panels)....65
1.1.8.3 Configuring slide-in screens for devices without multi-touch function (RT Advanced, Panels, Comfort Panels)....67
1.1.8.4 Using slide-in screens (RT Advanced, Panels, Comfort Panels)....69
1.1.9 Working with designs....70
1.1.9.1 Basics on working with designs (RT Professional)....70
1.1.9.2 Designs editor (RT Professional)....70
1.1.9.3 Creating a design (RT Professional)....72
1.1.9.4 Changing a design (RT Professional)....72
1.1.9.5 Deleting a design (RT Professional)....73
1.1.9.6 Applying a design (RT Professional)....74
1.1.9.7 Specifying the basic design user interface of an HMI device (RT Professional)....74

1.1.9.8 Applying design templates to screens (RT Professional)....75
1.1.9.9 Using design templates for basic objects and elements (RT Professional)....75
1.1.9.10 Applying design templates to controls (RT Professional)....76
1.1.9.11 Use default design....77
1.1.10 Working with styles and style sheets....78
1.1.10.1 Basics on working with styles (RT Advanced, Basic Panels, Panels, Comfort Panels)....78
1.1.10.2 Displaying predefined styles (RT Advanced, Basic Panels, Panels, Comfort Panels)....80
1.1.10.3 Defining a style (RT Advanced, Basic Panels, Panels, Comfort Panels)....80
1.1.10.4 Managing styles (RT Advanced, Basic Panels, Panels, Comfort Panels)....82
1.1.10.5 Adjusting the font size to the device size (RT Advanced, Basic Panels, Panels, Comfort Panels)....84
1.1.10.6 Using style items (RT Advanced, Basic Panels, Panels, Comfort Panels)....85
1.1.10.7 Basics on working with style sheets (RT Advanced, Basic Panels, Panels, Comfort Panels)....87
1.1.10.8 Creating a new style sheet (RT Advanced, Basic Panels, Panels, Comfort Panels)....88
1.1.10.9 Managing style sheets (RT Advanced, Basic Panels, Panels, Comfort Panels)....90
1.1.10.10 Using style sheets (RT Advanced, Basic Panels, Panels, Comfort Panels)....91

1.2 Working with objects and object groups....93
1.2.1 Working with objects....93
1.2.1.1 Overview of objects (RT Professional)....93
1.2.1.2 Overview of objects (RT Advanced, Panels, Comfort Panels)....97
1.2.1.3 Overview of objects (Basic Panels)....102
1.2.1.4 Options for editing objects....104
1.2.1.5 Inserting an object....105
1.2.1.6 Deleting an object....107
1.2.1.7 Positioning an object....108
1.2.1.8 Resizing objects....109
1.2.1.9 Selecting multiple objects....111
1.2.1.10 Aligning objects....113
1.2.1.11 Moving objects with keyboard shortcuts....114
1.2.1.12 Moving an object forwards or backwards....115
1.2.1.13 Showing objects outside the screen area....116
1.2.1.14 Rotating objects....117
1.2.1.15 Flipping objects....118
1.2.1.16 Designing an object (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....119
1.2.1.17 Designing an object (RT Professional)....119
1.2.1.18 Designing the fill pattern....120
1.2.1.19 Specifying scroll bars....122
1.2.1.20 Using predefined styles....124
1.2.1.21 Formatting text in an object....125
1.2.1.22 Connecting tags and text lists in the text (RT Advanced)....126
1.2.1.23 Formatting graphics in an object....128
1.2.1.24 Rotating an object in runtime (RT Professional)....129
1.2.1.25 Designing table-based objects....131
1.2.1.26 Designing a border....132
1.2.1.27 Defining color gradients....133
1.2.1.28 Configuring ranges....135
1.2.1.29 Flashing (RT Advanced, Panels, Comfort Panels)....137
1.2.1.30 Flashing (RT Professional)....138
1.2.1.31 Inserting multiple objects of the same type (stamping)....138
1.2.1.32 Repositioning and resizing multiple objects....140

1.2.1.33 Hiding or showing objects (RT Professional)....141
1.2.1.34 Linking user data types with objects (RT Professional)....141
1.2.1.35 Managing own controls (RT Professional, RT Advanced)....144
1.2.1.36 External graphics....147
1.2.1.37 Managing external graphics....148
1.2.1.38 Storing an external image in the graphics library (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....149
1.2.1.39 Individually adjusting graphics in controls (RT Professional)....152
1.2.2 Working with object groups....157
1.2.2.1 Basics on groups (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....157
1.2.2.2 Grouping objects (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....158
1.2.2.3 Ungrouping (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....160
1.2.2.4 Adding objects to the group (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....161
1.2.2.5 Removing objects from the group (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....162
1.2.2.6 Editing an object in a group (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....163
1.2.3 Configuring the keyboard access....165
1.2.3.1 Overview of keyboard access (RT Advanced, Basic Panels, Panels, Comfort Panels). 165
1.2.3.2 Overview of keyboard access (RT Professional)....166
1.2.3.3 Defining cursor mode (RT Professional)....167
1.2.3.4 Defining the operator authorization and operator control enable for an object (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....167
1.2.3.5 Defining object types for tab cursor mode (RT Professional)....168
1.2.3.6 Setting the tab sequence order (RT Advanced, Basic Panels, Panels, Comfort Panels) 169
1.2.3.7 Setting the tab sequence order (RT Professional)....171
1.2.4 Examples....172
1.2.4.1 Example: Inserting and configuring a rectangle....172
1.2.4.2 Example: Inserting a rectangle....173
1.2.4.3 Example: Configuring a rectangle....173

1.3 Working with text lists and graphics lists....176
1.3.1 Working with text lists....176
1.3.1.1 Basics on text lists (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....176
1.3.1.2 Creating a text list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....178
1.3.1.3 Assigning texts and values to an area text list (RT Advanced, Basic Panels, Panels, Comfort Panels)....179
1.3.1.4 Assigning texts and values to an area text list (RT Professional)....181
1.3.1.5 Assigning texts and values to a bit text list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....182
1.3.1.6 Assigning texts and values to a bit number text list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....184
1.3.1.7 Notes for bit number text list....185
1.3.1.8 Configuring objects with a text list (RT Advanced, Basic Panels, Panels, Comfort Panels)....186
1.3.1.9 Configuring objects with a text list (RT Professional)....188
1.3.2 Working with graphics lists....189

1.3.2.1 Basic principles of graphics lists (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....189
1.3.2.2 Creating a graphics list (RT Advanced, Basic Panels, Panels, Comfort Panels)....190
1.3.2.3 Creating a graphics list (RT Professional)....192
1.3.2.4 Assigning a graphic and values to an area graphics list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....193
1.3.2.5 Assigning graphics and values to an area graphics list (RT Professional)....195
1.3.2.6 Assigning graphics and values to a bit graphics list (RT Advanced, Basic Panels, Panels, Comfort Panels)....197
1.3.2.7 Assigning graphics and values to a bit number graphics list (RT Advanced, Basic Panels, Panels, Comfort Panels)....198
1.3.2.8 Notes for bit number graphics list....200
1.3.2.9 Configuring objects with a graphics list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....201
1.3.3 Working with C text lists (RT Professional)....203
1.3.3.1 Basics on C text lists (RT Professional)....203
1.3.3.2 Creating entries in the C text list (RT Professional)....203

1.4 Dynamizing screens....205
1.4.1 Basics on dynamizing screens....205
1.4.1.1 Basics on dynamizing screens (Basic Panels)....205
1.4.1.2 Basics on dynamizing screens (RT Professional, RT Advanced, Panels, Comfort Panels)....205
1.4.1.3 Dynamization in the inspector window (Basic Panels)....206
1.4.1.4 Dynamization in the inspector window (RT Professional, RT Advanced, Panels, Comfort Panels)....208
1.4.1.5 Show overview of the dynamizations....211
1.4.2 Dynamization with animations....212
1.4.2.1 Configuring a new animation (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....212
1.4.2.2 Dynamizing the appearance of an object (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....213
1.4.2.3 Configuring movements (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....215
1.4.2.4 Configuring direct movement (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....217
1.4.2.5 Dynamizing control enable of an operating element (RT Professional, RT Advanced, Panels, Comfort Panels)....218
1.4.2.6 Dynamizing the visibility of an object (RT Professional, RT Advanced, Panels, Comfort Panels)....219
1.4.2.7 Configuring animation with tag connection (RT Professional, RT Advanced, Panels, Comfort Panels)....221
1.4.2.8 Animations of object groups (Basic Panels)....222
1.4.2.9 Animations of object groups (RT Professional, RT Advanced, Panels, Comfort Panels)....223
1.4.2.10 Animations in multiple selection (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....224
1.4.3 Dynamization with property animation (RT Professional)....225
1.4.3.1 Dynamization with property animations (RT Professional)....225
1.4.3.2 Configuring property animation of the Range type (RT Professional)....227
1.4.3.3 Configuring property animation of the Direct type (RT Professional)....228
1.4.3.4 Configuring property animation of the Bool type (RT Professional)....229
1.4.4 Dynamization in the property list (RT Professional)....232
1.4.4.1 Basics on dynamization in the property list (RT Professional)....232
1.4.4.2 Dynamizing an object property with a tag (RT Professional)....235

1.4.4.3 Programming a dynamization method for a tag change (RT Professional)....237
1.4.4.4 Dynamizing an object property with local scripts (RT Professional)....238
1.4.5 Dynamizing with system functions....239
1.4.5.1 Basic on events....239
1.4.5.2 Configure system function on the "Click" event (Basic Panels)....239
1.4.5.3 Example: Configuring a button for language switching....240

1.5 Working with function keys (RT Advanced, Basic Panels, Panels, Comfort Panels)....241
1.5.1 Working with function keys (RT Advanced, Basic Panels, Panels, Comfort Panels)....241
1.5.2 Assigning function keys globally (RT Advanced, Basic Panels, Panels, Comfort Panels)....243
1.5.3 Local assignment of function keys (RT Advanced, Basic Panels, Panels, Comfort Panels)....245
1.5.4 Assigning a function to a function key (RT Advanced, Basic Panels, Panels, Comfort Panels)....246
1.5.5 Assigning operator authorization for a function key (RT Advanced, Basic Panels, Panels, Comfort Panels)....248
1.5.6 Assigning a graphic to a function key (RT Advanced, Basic Panels, Panels, Comfort Panels)....250
1.5.7 Configuring LED tags (RT Advanced, Basic Panels, Panels, Comfort Panels)....252
1.5.8 Example: Using function keys for screen navigation (RT Advanced, Basic Panels, Panels, Comfort Panels)....254

1.6 Working with layers....256
1.6.1 Basics on working with layers....256
1.6.2 Moving objects on layers....257
1.6.3 Setting the active layer....258
1.6.4 Showing and hiding layers (RT Advanced, Basic Panels, Panels, Comfort Panels)....259
1.6.5 Showing and hiding layers (RT Professional)....261
1.6.6 Hiding layers and objects in Runtime (RT Professional)....262
1.6.7 Renaming layers....263

1.7 Working with faceplates....265
1.7.1 Basics on faceplates....265
1.7.1.1 Basics on faceplates (RT Professional, RT Advanced, Panels, Comfort Panels)....265
1.7.1.2 Device dependency of faceplates (RT Professional, RT Advanced, Panels, Comfort Panels)....266
1.7.1.3 Faceplate editor (RT Professional, RT Advanced, Panels, Comfort Panels)....267
1.7.2 Creating and managing faceplate types....270
1.7.2.1 Creating a faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....270
1.7.2.2 Configuring a faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....272
1.7.2.3 Editing the category and property of a faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....276
1.7.2.4 Configuring a tag in the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....277
1.7.2.5 Configuring scripts in the faceplate type (RT Advanced, Panels, Comfort Panels)....278
1.7.2.6 Configuring an event in the faceplate type (RT Advanced, Panels, Comfort Panels)....279
1.7.2.7 Configuring an event in the faceplate type (RT Professional)....281
1.7.2.8 Configuring a text list in the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....282
1.7.2.9 Configuring a graphics list in the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....283
1.7.2.10 Translating texts directly in the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....284

1.7.2.11 Exporting and importing texts in the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....286
1.7.2.12 Editing a faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....287
1.7.2.13 Resizing a faceplate (RT Advanced, Panels, Comfort Panels)....288
1.7.2.14 Resizing a faceplate (RT Professional)....289
1.7.2.15 Creating an instance of the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....290
1.7.2.16 Removing faceplate from faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)....291
1.7.3 Dynamizing faceplates....292
1.7.3.1 Basics of dynamizing faceplates (RT Professional, RT Advanced, Panels, Comfort Panels)....292
1.7.3.2 Dynamizing faceplates (RT Professional, RT Advanced, Panels, Comfort Panels)....293
1.7.4 Example: Creating and dynamizing faceplates....294
1.7.4.1 Example: Configuring a faceplate....294
1.7.4.2 Example: Creating a faceplate type....294
1.7.4.3 Example: Configuring a faceplate type....295
1.7.4.4 Example: Creating a script in the faceplate type (RT Advanced, Panels, Comfort Panels)....296
1.7.4.5 Example: Creating an instance of the faceplate type....298
1.7.4.6 Example: Configuring included objects....300

1.8 Configuring screen navigation....301
1.8.1 Screen navigation basics....301
1.8.1.1 Basics on screen navigation (RT Advanced, Basic Panels, Panels, Comfort Panels)....301
1.8.1.2 Basics on screen navigation (RT Professional)....301
1.8.1.3 Assigning screen change to button (RT Professional, RT Advanced, Panels, Comfort Panels)....302
1.8.1.4 Assigning screen change to function key (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....304
1.8.2 Menus and toolbars for Runtime Professional (RT Professional)....305
1.8.2.1 Basics (RT Professional)....305
1.8.2.2 Working with menus and toolbars (RT Professional)....309
1.8.2.3 Example: Configuring a screen change with toolbars (RT Professional)....320

1.9 Display and operating elements....326
1.9.1 Device dependency of the objects....326
1.9.1.1 Objects for Basic Panels (Basic Panels)....326
1.9.1.2 Objects for Comfort Panels (Panels, Comfort Panels)....327
1.9.1.3 Objects for Mobile Panels (Panels, Comfort Panels)....329
1.9.1.4 Objects for WinCC Runtime Advanced (RT Advanced)....331
1.9.1.5 Objects for WinCC Runtime Professional (RT Professional)....333
1.9.2 Objects....335
1.9.2.1 Bar (RT Advanced, Basic Panels, Panels, Comfort Panels)....335
1.9.2.2 Bar (RT Professional)....337
1.9.2.3 User view (as of V13)....339
1.9.2.4 User view, simple (RT Advanced, Basic Panels, Panels, Comfort Panels)....341
1.9.2.5 User view, advanced....342
1.9.2.6 Watch table (RT Advanced, Panels, Comfort Panels)....344
1.9.2.7 Screen window (RT Professional)....346
1.9.2.8 Scroll bar (RT Professional)....349
1.9.2.9 Date/time field (RT Advanced, Basic Panels, Panels, Comfort Panels)....350
1.9.2.10 Double T-piece (RT Professional)....351
1.9.2.11 Print job/script diagnostics (RT Professional)....352

1.9.2.12 I/O field....355
1.9.2.13 I/O field: Behavior during operation (RT Professional)....357
1.9.2.14 I/O field: Format pattern....358
1.9.2.15 I/O field: Format (RT Professional)....360
1.9.2.16 Editable text field (RT Professional)....363
1.9.2.17 Ellipse (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....363
1.9.2.18 Elliptical arc (RT Professional)....364
1.9.2.19 Ellipse segment (RT Professional)....366
1.9.2.20 f(t) trend view (RT Professional)....367
1.9.2.21 f(x) trend view (RT Advanced, Panels, Comfort Panels)....371
1.9.2.22 f(x) trend view (RT Professional)....374
1.9.2.23 Graphic view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....377
1.9.2.24 Graphic I/O field (RT Advanced, Basic Panels, Panels, Comfort Panels)....379
1.9.2.25 Graphic I/O field (RT Professional)....381
1.9.2.26 GRAPH overview (RT Professional, RT Advanced, Panels, Comfort Panels)....382
1.9.2.27 Handwheel (Panels, Comfort Panels)....386
1.9.2.28 Help indicator (Basic Panels)....387
1.9.2.29 HTML browser (RT Advanced, Basic Panels, Panels, Comfort Panels)....388
1.9.2.30 HTML browser (RT Professional)....393
1.9.2.31 Camera view (RT Advanced, Panels, Comfort Panels)....395
1.9.2.32 Channel diagnostics display (RT Professional)....399
1.9.2.33 Combo box (RT Professional)....400
1.9.2.34 Check box (RT Professional)....401
1.9.2.35 Circle (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....402
1.9.2.36 Circular arc (RT Professional)....403
1.9.2.37 Circle segment (RT Professional)....404
1.9.2.38 Criteria analysis view (RT Professional, RT Advanced, Panels, Comfort Panels)....405
1.9.2.39 Trend view (RT Advanced, Basic Panels, Panels, Comfort Panels)....408
1.9.2.40 Charge condition (Panels, Comfort Panels)....410
1.9.2.41 Illuminated pushbutton (Panels, Comfort Panels)....411
1.9.2.42 Line....415
1.9.2.43 List box (RT Professional)....416
1.9.2.44 Media Player (RT Professional, Panels, Comfort Panels)....417
1.9.2.45 Alarm view (RT Professional)....420
1.9.2.46 Alarm view (as of V13) (RT Advanced, Basic Panels, Panels, Comfort Panels)....425
1.9.2.47 Alarm view, simple (RT Advanced, Basic Panels, Panels, Comfort Panels)....429
1.9.2.48 Alarm view, advanced (RT Advanced, Panels, Comfort Panels)....431
1.9.2.49 Alarm window (RT Advanced, Basic Panels, Panels, Comfort Panels)....435
1.9.2.50 Alarm indicator (RT Advanced, Basic Panels, Panels, Comfort Panels)....438
1.9.2.51 Option buttons (RT Professional)....439
1.9.2.52 PDF view (RT Advanced, Panels, Comfort Panels)....440
1.9.2.53 PLC code view (RT Advanced, Panels, Comfort Panels)....446
1.9.2.54 PLC code view (RT Professional)....450
1.9.2.55 Polygon (RT Professional, RT Advanced, Panels, Comfort Panels)....453
1.9.2.56 Polyline (RT Professional, RT Advanced, Panels, Comfort Panels)....454
1.9.2.57 ProDiag overview (RT Professional, RT Advanced, Panels, Comfort Panels)....456
1.9.2.58 Rectangle....459
1.9.2.59 Recipe view (RT Professional)....460
1.9.2.60 Recipe view (as of V13) (RT Advanced, Basic Panels, Panels, Comfort Panels)....463
1.9.2.61 Recipe view, simple (RT Advanced, Basic Panels, Panels, Comfort Panels)....465
1.9.2.62 Recipe view, advanced (RT Advanced, Basic Panels, Panels, Comfort Panels)....466

1.9.2.63 Round button (RT Professional)....468
1.9.2.64 Pipe (RT Professional)....469
1.9.2.65 Pipe elbow (RT Professional)....470
1.9.2.66 Switch (RT Advanced, Basic Panels, Panels, Comfort Panels)....471
1.9.2.67 Button (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....472
1.9.2.68 Slider....474
1.9.2.69 Key switch (Panels, Comfort Panels)....476
1.9.2.70 Sm@rtClient view (RT Advanced, Panels, Comfort Panels)....479
1.9.2.71 Memory space view (RT Professional)....482
1.9.2.72 Symbol library (RT Professional, RT Advanced, Panels, Comfort Panels)....483
1.9.2.73 Symbolic I/O field (RT Advanced, Basic Panels, Panels, Comfort Panels)....485
1.9.2.74 Symbolic I/O field (RT Professional)....487
1.9.2.75 System diagnostics view (Basic Panels)....488
1.9.2.76 System diagnostics view (RT Advanced, Panels, Comfort Panels)....490
1.9.2.77 System diagnostics view (RT Professional)....492
1.9.2.78 System diagnostics window (RT Advanced, Panels, Comfort Panels)....495
1.9.2.79 Table view (RT Professional)....497
1.9.2.80 Text field (RT Professional, RT Advanced, Panels, Comfort Panels)....501
1.9.2.81 T-piece (RT Professional)....502
1.9.2.82 Clock....503
1.9.2.83 Connector (RT Professional)....504
1.9.2.84 Value table (RT Professional)....505
1.9.2.85 Effective range name (Panels, Comfort Panels)....507
1.9.2.86 Effective range name (RFID) (Panels, Comfort Panels)....509
1.9.2.87 Effective range signal (Panels, Comfort Panels)....511
1.9.2.88 WLAN reception (Panels, Comfort Panels)....512
1.9.2.89 Gauge....514
1.9.2.90 Zone name (Panels, Comfort Panels)....516
1.9.2.91 Zone signal (Panels, Comfort Panels)....517

2 Working with tags....519

2.1 Basics....519
2.1.1 Basics of tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....519
2.1.2 Overview of HMI tag tables (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....520
2.1.3 External tags (RT Advanced, Basic Panels, Panels, Comfort Panels)....522
2.1.4 External tags (RT Professional)....524
2.1.5 Addressing external tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....526
2.1.6 Internal Tags (RT Advanced, Basic Panels, Panels, Comfort Panels)....528
2.1.7 Internal Tags (RT Professional)....529

2.2 Working with Tags....531
2.2.1 Creating tags....531
2.2.1.1 Creating external tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....531
2.2.1.2 Creating internal tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....533
2.2.1.3 Creating multiple tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....534
2.2.1.4 Adapting the data type of a tag (RT Professional)....536

2.2.2 Editing Tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....538
2.2.2.1 Manage tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....538
2.2.2.2 Edit tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....540
2.2.2.3 Configuring multiple tags simultaneously (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....541
2.2.2.4 Using multiple tags simultaneously in a screen (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....542
2.2.2.5 Settings for synchronization (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....544
2.2.2.6 Synchronizing tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....546
2.2.3 Configuring Tags....547
2.2.3.1 Tag Configuration Basics (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....547
2.2.3.2 Special configuration options (RT Advanced, Basic Panels, Panels, Comfort Panels)....555
2.2.3.3 Special configuration options (RT Professional)....569

2.3 Working with Arrays....575
2.3.1 Basics on arrays....575
2.3.2 Creating array tags....578
2.3.3 Examples of arrays....579

2.4 Working with Arrays (RT Professional)....581
2.4.1 Basics on arrays (RT Professional)....581
2.4.2 Creating array tags (RT Professional)....583

2.5 Working with user data types (RT Professional, RT Advanced, Panels, Comfort Panels)....585
2.5.1 Basics on user data types (RT Professional, RT Advanced, Panels, Comfort Panels)....585
2.5.2 Creating a user data type (RT Professional, RT Advanced, Panels, Comfort Panels)....586
2.5.3 Creating user data type elements (RT Professional, RT Advanced, Panels, Comfort Panels)....588
2.5.4 Managing versions of user data types (RT Professional, RT Advanced, Panels, Comfort Panels)....589
2.5.5 Creating tags with a user data type data type (RT Professional, RT Advanced, Panels, Comfort Panels)....590

2.6 Working with cycles (RT Advanced, Basic Panels, Panels, Comfort Panels)....593
2.6.1 Cycle basics (RT Advanced, Basic Panels, Panels, Comfort Panels)....593
2.6.2 Defining cycles (RT Advanced, Panels, Comfort Panels)....594

2.7 Working with cycles (RT Professional)....596
2.7.1 Cycle basics (RT Professional)....596
2.7.2 Defining cycles (RT Professional)....598
2.7.3 Define starting point (RT Professional)....598

2.8 Reference (RT Professional)....600
2.8.1 Mapping and coding of data types (RT Professional)....600
2.8.2 Coding and value range of data types (RT Professional)....602
2.8.3 Data types (RT Professional)....608
2.8.3.1 Data types for SIMATIC S7-300/400 (RT Professional)....608
2.8.3.2 Data types for SIMATIC S7-1200 and S7-1500 (RT Professional)....609
2.8.3.3 Data types for OPC DA (RT Professional)....612

2.8.3.4 Data types for OPC XML DA (RT Professional)....613
2.8.3.5 Data types for Allen Bradley Ethernet IP (RT Professional)....614
2.8.3.6 Data types for Modicon Modbus TCP/IP (RT Professional)....614
2.8.3.7 Data types for Mitsubishi MC (RT Professional)....615

2.9 Archive tags....616
2.9.1 Basic principles for data logging (RT Professional, RT Advanced, Panels, Comfort Panels)....616
2.9.2 Data logging (RT Advanced, Basic Panels, Panels, Comfort Panels)....616
2.9.2.1 Basics (RT Advanced, Basic Panels, Panels, Comfort Panels)....616
2.9.2.2 Working with data logs (RT Advanced, Basic Panels, Panels, Comfort Panels)....618
2.9.3 Data logging (RT Professional)....633
2.9.3.1 Data logging (RT Professional)....633
2.9.3.2 Basics (RT Professional)....634
2.9.3.3 Working with data logs (RT Professional)....651

2.10 Displaying Tags....676
2.10.1 Displaying tags with Basic Panels (Basic Panels)....676
2.10.1.1 Outputting tag values in screens (Basic) (Basic Panels)....676
2.10.1.2 Configuring trend display for values from the PLC (Basic) (Basic Panels)....676
2.10.2 Displaying tags with Runtime Advanced and Panels (RT Advanced, Panels, Comfort Panels)....678
2.10.2.1 Trends (RT Advanced, Panels, Comfort Panels)....678
2.10.2.2 Outputting tag values in screens (RT Advanced, Panels, Comfort Panels)....680
2.10.2.3 Configuring trend displays for values from the PLC (RT Advanced, Panels, Comfort Panels)....681
2.10.2.4 Configuring a trend view for a data log (RT Advanced, Panels, Comfort Panels)....683
2.10.2.5 The structure of a *.CSV file with tags (RT Advanced, Panels, Comfort Panels)....684
2.10.2.6 Direct access to the ODBC log database (RT Advanced, Panels, Comfort Panels)....685
2.10.2.7 Representing path trends in the trend view (RT Advanced, Panels, Comfort Panels)....686
2.10.3 Displaying tags with Runtime Professional (RT Professional)....689
2.10.3.1 Basics (RT Professional)....689
2.10.3.2 Output of tag values in the form of a trend (RT Professional)....699
2.10.3.3 Output of tag values in tabular format (RT Professional)....711
2.10.3.4 Configuring the value table (RT Professional)....723
2.10.3.5 Using controls in Runtime (RT Professional)....731

3 Working with alarms....753

3.1 Principles....753
3.1.1 Alarm Logging in WinCC (Basic Panels)....753
3.1.2 Alarm Logging in WinCC (RT Advanced, Panels, Comfort Panels)....754
3.1.3 Alarm Logging in WinCC (RT Professional)....756
3.1.4 Alarm Procedures....758
3.1.4.1 Overview of the alarm types (Basic Panels)....758
3.1.4.2 Overview of the alarm types (RT Advanced, Panels, Comfort Panels)....759
3.1.4.3 Overview of the alarm types (RT Professional)....760
3.1.4.4 Custom alarm classes....762
3.1.4.5 System-defined alarms....764
3.1.5 Alarm states (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....766
3.1.6 Alarm classes....767
3.1.6.1 Basics on alarm classes (Basic Panels)....767
3.1.6.2 Basics on alarm classes (RT Professional, RT Advanced, Panels, Comfort Panels)....768

3.1.6.3 Predefined alarm classes (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....770
3.1.7 Acknowledgement....771
3.1.7.1 Acknowledging alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....771
3.1.7.2 Acknowledgment model (RT Advanced, Basic Panels, Panels, Comfort Panels)....773
3.1.7.3 Acknowledgment model (RT Professional)....773
3.1.8 Alarm groups (RT Advanced, Basic Panels, Panels, Comfort Panels)....775
3.1.9 Alarm groups (RT Professional)....775
3.1.9.1 Alarm Group Basics (RT Professional)....775
3.1.9.2 Hierarchical alarm groups (RT Professional)....777
3.1.9.3 Tags of an alarm group (RT Professional)....778
3.1.10 Alarm number (RT Advanced, Basic Panels, Panels, Comfort Panels)....779
3.1.11 Alarm number (RT Professional)....780

3.2 Working with Alarms....781
3.2.1 Alarm components and properties (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....781
3.2.2 Configuring Alarms....783
3.2.2.1 Overview of alarm configuration tasks (Basic Panels)....783
3.2.2.2 Overview of alarm configuration tasks (RT Professional, RT Advanced, Panels, Comfort Panels)....784
3.2.2.3 Configuring Alarm Classes....785
3.2.2.4 Configuring Alarm Groups....791
3.2.2.5 Configure alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)....794
3.2.2.6 Configure alarms (RT Professional)....806
3.2.2.7 Configuring loop-in alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....829
3.2.2.8 Alarms in the "HMI tags" Editor....831
3.2.3 Configuring the Outputting of Alarms....834
3.2.3.1 Overview of configuring alarm output (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....834
3.2.3.2 Configuring the alarm view....835
3.2.3.3 Configuring the alarm view (RT Professional)....850
3.2.3.4 Reporting Alarms (RT Professional, RT Advanced, Panels, Comfort Panels)....873
3.2.4 Configuring acknowledgment (RT Advanced, Basic Panels, Panels, Comfort Panels)....876
3.2.4.1 Configuring alarm acknowledgment by means of alarm class (RT Advanced, Basic Panels, Panels, Comfort Panels)....876
3.2.4.2 Configuring trigger for alarm acknowledgment (RT Advanced, Basic Panels, Panels, Comfort Panels)....877
3.2.4.3 Sending alarm acknowledgments to the PLC (RT Advanced, Basic Panels, Panels, Comfort Panels)....879
3.2.5 Configuring acknowledgment (RT Professional)....879
3.2.5.1 Configuring alarm acknowledgment (RT Professional)....879
3.2.5.2 Configuring alarm group acknowledgment (RT Professional)....881
3.2.5.3 Configuring acknowledgment of the alarm annunciator (RT Professional)....882
3.2.5.4 Configuring the Group Acknowledgement of Alarms (RT Professional)....883
3.2.5.5 Configuring the Resetting of the Acknowledgment Bit for Acknowledgment Tags (RT Professional)....883

3.3 Alarm Logging (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....885
3.3.1 Configuring alarm logging (RT Advanced, Panels, Comfort Panels)....885

3.3.1.1 Basics on alarm logging (RT Advanced, Panels, Comfort Panels)....885
3.3.1.2 Create an alarm log (RT Advanced, Panels, Comfort Panels)....887
3.3.1.3 Logging alarms (RT Advanced, Panels, Comfort Panels)....889
3.3.1.4 Configuring an alarm view for logged alarms (RT Advanced, Panels, Comfort Panels)....890
3.3.1.5 Configuring alarm buffer overflow (RT Advanced, Panels, Comfort Panels)....891
3.3.1.6 Direct access to the ODBC log database (RT Advanced, Panels, Comfort Panels)....891
3.3.1.7 Setting up the ODBC data source (RT Advanced, Panels, Comfort Panels)....892
3.3.1.8 Configuring a checksum for a log (RT Advanced, Panels, Comfort Panels)....893
3.3.1.9 Evaluating the checksum of log data (RT Advanced, Panels, Comfort Panels)....894
3.3.1.10 Checksums for logs in regulated projects (RT Advanced, Panels, Comfort Panels)....896
3.3.1.11 Log response to language switching in runtime (RT Advanced, Panels, Comfort Panels)....896
3.3.1.12 Managing logging behavior when Runtime starts (RT Advanced, Panels, Comfort Panels)....897
3.3.1.13 Controlling the Logging in relation to the Fill Level (RT Advanced, Panels, Comfort Panels)....898
3.3.2 Configuring alarm logging (RT Professional)....900
3.3.2.1 Basics on alarm logging (RT Professional)....900
3.3.2.2 Logging alarms (RT Professional)....901
3.3.2.3 Configuring the Alarm Log Backup (RT Professional)....904
3.3.2.4 Connecting and Disconnecting the Alarm Log Backup (RT Professional)....908
3.3.2.5 Accessing the Log Database Directly with C-API/ODK (RT Professional)....909
3.3.2.6 Accessing the Log Database Directly with ADO/OLE DB (RT Professional)....910
3.3.2.7 Configuring the Reloading of Alarms after a Power Failure (RT Professional)....910

3.4 Using Alarms in Runtime (RT Advanced, Basic Panels, Panels, Comfort Panels)....912
3.4.1 Alarms in Runtime (Basic Panels)....912
3.4.2 Alarms in Runtime (RT Advanced, Panels, Comfort Panels)....914
3.4.3 Using the Alarm Window or Alarm View (RT Advanced, Basic Panels, Panels, Comfort Panels)....916
3.4.3.1 Alarm window, alarm view in Runtime (RT Advanced, Basic Panels, Panels, Comfort Panels)....916
3.4.3.2 Using the Alarm Window, Alarm View (RT Advanced, Panels, Comfort Panels)....918
3.4.4 Using the Simple Alarm Window, Alarm View (RT Advanced, Basic Panels, Panels, Comfort Panels)....920
3.4.4.1 Basic alarm view, alarm window in Runtime (RT Advanced, Basic Panels, Panels, Comfort Panels)....920
3.4.4.2 Basic alarm view, using the alarm window (RT Advanced, Basic Panels, Panels, Comfort Panels)....922
3.4.5 Using the Alarm Indicator (RT Advanced, Basic Panels, Panels, Comfort Panels)....923
3.4.5.1 Alarm indicator in Runtime (RT Advanced, Basic Panels, Panels, Comfort Panels)....923
3.4.5.2 Operating the alarm indicator using the mouse (RT Advanced, Panels, Comfort Panels)....923
3.4.6 Acknowledging alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)....924
3.4.7 Filtering alarms (RT Advanced, Panels, Comfort Panels)....925

3.5 Using Alarms in Runtime (RT Professional)....926
3.5.1 Alarms in Runtime 926
3.5.2 Alarm view (RT Professional)....929
3.5.3 Alarm View Buttons (RT Professional)....931
3.5.4 Filtering Alarms in Runtime (RT Professional)....932
3.5.5 Sorting Alarms in Runtime (RT Professional)....933
3.5.6 Toggling the Alarm View in Runtime (RT Professional)....935
3.5.7 Displaying logged alarms in Runtime (RT Professional)....936
3.5.8 Printing Alarms in Runtime (RT Professional)....936

3.5.9 Locking Alarms in Runtime (RT Professional)....938
3.5.9.1 Locking Single Alarms (RT Professional)....938
3.5.9.2 Locking alarms by means of alarm number (RT Professional)....939
3.5.9.3 Locking Alarms using the Alarm Class or Alarm Group (RT Professional)....940
3.5.10 Acknowledgement of Alarms (RT Professional)....941
3.5.10.1 Acknowledging alarms (RT Professional)....941
3.5.10.2 Group Alarm Acknowledgement (RT Professional)....942
3.5.10.3 Emergency acknowledgment of alarms (RT Professional)....943
3.5.10.4 Acknowledging an alarm annunciator (RT Professional)....944

3.6 Displaying security events....946
3.6.1 Displaying security events on the HMI device (RT Professional, RT Advanced, Panels, Comfort Panels)....946
3.6.2 Configuring the display of security events (RT Professional, RT Advanced, Panels, Comfort Panels)....947

3.7 Sending a complete alarm from the controller to the HMI device....948
3.7.1 Sending a complete alarm from the controller to the HMI device and automatic update (RT Advanced, Panels, Comfort Panels)....948
3.7.2 Sending a complete alarm from the controller to the HMI device and automatic update (RT Professional)....950
3.7.3 Configuring automatic update of controller alarms on the HMI device (RT Professional, RT Advanced, Panels, Comfort Panels)....951

3.8 Reference....953
3.8.1 System functions for alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....953
3.8.2 Alarm events (RT Advanced, Basic Panels, Panels, Comfort Panels)....954
3.8.3 Alarm events (RT Professional)....955
3.8.4 System functions for logs (RT Professional, RT Advanced, Panels, Comfort Panels)....955
3.8.5 Structure of *.csv files that contain alarms (RT Professional, RT Advanced, Panels, Comfort Panels)....956
3.8.6 Description of the System Blocks (RT Professional)....957
3.8.7 SQL Statements for Filtering the Alarm View (RT Professional)....960
3.8.8 System events....961
3.8.8.1 Basics on system events (RT Advanced, Basic Panels, Panels, Comfort Panels)....961
3.8.8.2 9999 - Global alarm (RT Advanced, Panels, Comfort Panels)....962
3.8.8.3 10000 - Printer alarms (RT Advanced, Panels, Comfort Panels)....963
3.8.8.4 20000 - Global script alarms (RT Advanced, Panels, Comfort Panels)....963
3.8.8.5 30000 - Alarms errors when using system functions (RT Advanced, Basic Panels, Panels, Comfort Panels)....964
3.8.8.6 40000 - Linear scaling alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)....965
3.8.8.7 50000 - Data server alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)....965
3.8.8.8 60000 - Windows function alarms (RT Advanced, Panels, Comfort Panels)....966
3.8.8.9 70000 - Windows function alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)....966
3.8.8.10 80000 - Log alarms (RT Advanced, Panels, Comfort Panels)....969
3.8.8.11 90000 - FDA alarms (RT Advanced, Panels, Comfort Panels)....972
3.8.8.12 110000 - Offline function alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)....973
3.8.8.13 120000 - Trend alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)....974
3.8.8.14 130000 - System information alarms (RT Advanced, Panels, Comfort Panels)....974
3.8.8.15 140000 - Connection alarms: Connection + device (RT Advanced, Basic Panels, Panels, Comfort Panels)....975
3.8.8.16 160000 - Connection alarms: OPC: Connection (RT Advanced, Panels, Comfort Panels)....978
3.8.8.17 170000 - S7 dialog alarms (RT Advanced, Panels, Comfort Panels)....979

3.8.8.18 180000 - General alarms (RT Advanced, Basic Panels, Panels, Comfort Panels).....979
3.8.8.19 190000 - Tag alarms (RT Advanced, Basic Panels, Panels, Comfort Panels).....980
3.8.8.20 190100 - Area pointer alarms (RT Advanced, Basic Panels, Panels, Comfort Panels).....981
3.8.8.21 200000 - PLC coordination alarms (RT Advanced, Basic Panels, Panels, Comfort Panels).....982
3.8.8.22 200100 - Area pointer alarms project ID (RT Advanced, Basic Panels, Panels, Comfort Panels).....982
3.8.8.23 210000 - PLC job alarms (RT Advanced, Basic Panels, Panels, Comfort Panels).....983
3.8.8.24 220000 - WinCC communication driver alarms (RT Advanced, Basic Panels, Panels, Comfort Panels).....984
3.8.8.25 230000 - Screen object alarms (RT Advanced, Basic Panels, Panels, Comfort Panels) 985
3.8.8.26 240000 - Authorization alarms (RT Advanced, Panels, Comfort Panels).....987
3.8.8.27 250000 - S7 Status/Force alarms (RT Advanced, Panels, Comfort Panels).....988
3.8.8.28 260000 - Password system alarms (RT Advanced, Basic Panels, Panels, Comfort Panels).....988
3.8.8.29 270000 - System alarms (RT Advanced, Basic Panels, Panels, Comfort Panels).....991
3.8.8.30 280000 - Alarms SIMATIC 505 DP Communication Driver (CSP).....992
3.8.8.31 290000 - Recipe system alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)....992
3.8.8.32 300000 - Alarm_S alarms (RT Advanced, Panels, Comfort Panels).....997
3.8.8.33 310000 - Reporting alarms (RT Advanced, Panels, Comfort Panels).....997
3.8.8.34 330000 - Interaction alarms (RT Advanced, Panels, Comfort Panels).....997
3.8.8.35 340000 - Alarms info text.....998
3.8.8.36 350000 - PROFIsafe alarms (RT Advanced, Panels, Comfort Panels).....998
3.8.8.37 380000 - Alarms Sm@rtServer (RT Professional, RT Advanced, Panels, Comfort Panels).....999
3.8.8.38 390000 alarms password encryption and certificates.....999
3.8.9 System events (RT Professional).....1000
3.8.9.1 System events (RT Professional).....1000
3.8.9.2 1000000 - Alarms Runtime.....1000
3.8.9.3 1000800 - Alarms WinCC client.....1001
3.8.9.4 1000900 - Alarms Performance Monitor.....1002
3.8.9.5 1001000 - Alarms Screens in Runtime.....1002
3.8.9.6 1002000 - Alarms Tag logging.....1003
3.8.9.7 1003018 - Alarms Alarm system.....1004
3.8.9.8 1004000 - Alarms Print jobs.....1007
3.8.9.9 1005000 - Alarms Text library.....1007
3.8.9.10 1006000 - Alarms Global scripts.....1008
3.8.9.11 1008000 - Alarms User administration.....1009
3.8.9.12 1009000 - Alarms Lifebeat Monitoring.....1010
3.8.9.13 1010000 - Alarms STRRT.....1010
3.8.9.14 1011000 - Alarms Group display.....1010
3.8.9.15 1011101 - Alarms Picture Tree Manager.....1011
3.8.9.16 1011201 - Alarms Split Screen Manager.....1011
3.8.9.17 1012001 - Alarms Time synchronization.....1011
3.8.9.18 1012200 - Alarms Redundancy.....1012
3.8.9.19 1012400 - Alarms WebNavigatorClient.....1015
3.8.9.20 1012500 - Alarms Process Historian.....1015
3.8.9.21 1012700 - Alarms Redundancy self-diagnostics.....1017
3.8.9.22 1016000 - Alarms IndustrialDataBridge.....1017
3.8.9.23 12508141 - Alarms 1018
3.8.10 Tag Bit Assignments (RT Professional).....1018
3.8.10.1 Tags for Alarm Groups and Alarms (RT Professional).....1018

3.8.10.2 Tags for Alarm Groups (RT Professional)....1023

4 Working with logs....1027

4.1 Working with Logs (RT Advanced, Basic Panels, Panels, Comfort Panels)....1027
4.1.1 Log Basics (RT Advanced, Basic Panels, Panels, Comfort Panels)....1027
4.1.2 Properties of Logs (RT Advanced, Basic Panels, Panels, Comfort Panels)....1027
4.1.3 Storage locations of logs (RT Advanced, Basic Panels, Panels, Comfort Panels)....1029

4.2 Working with Logs (RT Professional)....1033
4.2.1 Log basics (RT Professional)....1033
4.2.2 Properties of logs (RT Professional)....1033

5 Access to archive data via WinCC OLE DB Provider (RT Professional)....1035

5.1 WinCC Archive Connector (RT Professional)....1035
5.1.1 WinCC Archive Connector (RT Professional)....1035
5.1.2 The "Configuration" tab (RT Professional)....1037
5.1.3 The "Connect/Disconnect Archive" tab (RT Professional)....1038

5.2 Access to archive data via OLE DB Provider (RT Professional)....1040
5.2.1 Options for accessing archive data (RT Professional)....1040
5.2.2 Configuring access to archive data (RT Professional)....1040
5.2.3 Establishing the Connection to the Archive Database (RT Professional)....1041

5.3 Application cases (RT Professional)....1044
5.3.1 Use Case 1: Local Access to WinCC RT Databases (RT Professional)....1044
5.3.2 Use Case 2: Remote Access to WinCC RT Databases (RT Professional)....1044
5.3.3 Use Case 3: Local Access to WinCC Archive Databases (RT Professional)....1045
5.3.4 Use Case 4: Remote Access to WinCC Archive Databases (RT Professional)....1046
5.3.5 Use Case 5: Local Access to WinCC User Archive (RT Professional)....1047
5.3.6 Use Case 6: Remote Access to WinCC User Archives (RT Professional)....1048

5.4 Querying the archive data (RT Professional)....1049
5.4.1 Displaying Process Value Archives (RT Professional)....1049
5.4.2 Querying Process Value Archives (RT Professional)....1049
5.4.3 Querying Alarm Message Archives (RT Professional)....1055
5.4.4 Displaying Alarm Message Archives (RT Professional)....1057
5.4.5 Query for User Archives (RT Professional)....1059
5.4.6 Configure Access via the Wizard "SQL Server Import/Export" (RT Professional)....1061
5.4.7 Meeting prerequisites for using the Reporting Services (RT Professional)....1062
5.4.8 Querying the Archive Data (RT Professional)....1064
5.4.9 Displaying User Archives (RT Professional)....1065

5.5 Analysis functions for messages and process values (RT Professional)....1066
5.5.1 Analysis Functions for Messages and Process Values (RT Professional)....1066
5.5.2 Display of Message Archives for Analysis Queries (RT Professional)....1069

5.6 Bases of OLE DB (RT Professional)....1071

5.7 WinCC OLE DB Provider (RT Professional)....1073

5.8 Installation (RT Professional)....1075

6 Working with recipes....1077

6.1 Working with recipes....1077
6.1.1 Basics....1077
6.1.1.1 Definition and applications....1077

6.1.1.2 Examples for using recipes....1079
6.1.1.3 Recipe structure....1079
6.1.1.4 Display of recipes....1081
6.1.1.5 Transferring recipe data records....1083
6.1.1.6 Configuration of recipes....1086
6.1.1.7 Special features of some devices....1087
6.1.1.8 Synchronization of recipe data records with the PLC....1089
6.1.1.9 "Recipes" editor....1090
6.1.2 Recipes in runtime....1093
6.1.2.1 Recipe screen and recipe view....1093
6.1.2.2 Simple recipe view....1097
6.1.2.3 Advanced recipe view (prior to V13)....1108
6.1.2.4 Advanced recipe view (as of V13)....1119
6.1.2.5 Exporting and importing recipe data....1131
6.1.3 Configuring Recipes....1135
6.1.3.1 Creating and Editing Recipes....1135
6.1.3.2 Configuring the display of recipes....1146
6.1.4 Examples....1151
6.1.4.1 Example of creating a recipe (RT Advanced, Basic Panels, Panels, Comfort Panels)....1151
6.1.4.2 Example of configuring a recipe screen (RT Advanced, Panels, Comfort Panels)....1153
6.1.4.3 Scenario for Entering Recipe Data Records in Runtime (RT Advanced, Panels, Comfort Panels)....1155
6.1.4.4 Scenario for a manual production sequence (RT Advanced, Panels, Comfort Panels)....1156
6.1.4.5 Scenario for an Automatic Production Sequence (RT Advanced, Panels, Comfort Panels)....1158

6.2 Working with recipes (RT Professional)....1160
6.2.1 Principles (RT Professional)....1160
6.2.1.1 Definition and applications (RT Professional)....1160
6.2.1.2 Recipe structure (RT Professional)....1161
6.2.1.3 Displaying Recipes in Runtime (RT Professional)....1164
6.2.1.4 Communication with the PLC (RT Professional)....1165
6.2.1.5 "Recipes" Editor (RT Professional)....1165
6.2.1.6 SQL language (RT Professional)....1168
6.2.1.7 SQL keywords (RT Professional)....1169
6.2.2 Configuring Recipes (RT Professional)....1170
6.2.2.1 General configuration procedure (RT Professional)....1170
6.2.2.2 Creating a New Recipe (RT Professional)....1171
6.2.2.3 Creating Recipe Elements and Data Records (RT Professional)....1173
6.2.2.4 Importing recipes into the configuration and exporting them again (RT Professional)....1175
6.2.2.5 Procedure for changing the recipe structure (RT Professional)....1177
6.2.2.6 Creating recipe queries (RT Professional)....1178
6.2.3 Configuring the visualization of recipes (RT Professional)....1181
6.2.3.1 Displaying recipes (RT Professional)....1181
6.2.3.2 Properties of the recipe view (RT Professional)....1182
6.2.3.3 Configuring a recipe view (RT Professional)....1184
6.2.3.4 Exporting recipe data in Runtime (RT Professional)....1186
6.2.3.5 Configuring persistence (RT Professional)....1187
6.2.4 Data transfer to the PLC (RT Professional)....1188
6.2.4.1 Data transfer basics (RT Professional)....1188
6.2.4.2 Raw data tag (RT Professional)....1193
6.2.5 Operation in Runtime (RT Professional)....1202
6.2.5.1 Recipe view (RT Professional)....1202

6.2.5.2 Using the recipe screen (RT Professional)....1203
6.2.5.3 Mouse and keyboard operation (RT Professional)....1205
6.2.5.4 Editing recipe data (RT Professional)....1207
6.2.5.5 Selecting and configuring data (RT Professional)....1209
6.2.5.6 Select data (RT Professional)....1212
6.2.5.7 Sorting data (RT Professional)....1214
6.2.6 Examples (RT Professional)....1216
6.2.6.1 Example: Creating recipes (RT Professional)....1216
6.2.6.2 Example: Creating recipe queries (RT Professional)....1219
6.2.6.3 Example: Configuring a recipe view (RT Professional)....1220
6.2.6.4 Using control tags (RT Professional)....1221

7 Working with reports (RT Professional, RT Advanced, Panels, Comfort Panels)....1225

7.1 Basics (RT Professional, RT Advanced, Panels, Comfort Panels)....1225
7.1.1 Reports (RT Professional)....1225
7.1.2 Structure of reports (RT Advanced, Panels, Comfort Panels)....1226
7.1.3 Structure of reports (RT Professional)....1228

7.2 Working with reports (RT Professional, RT Advanced, Panels, Comfort Panels)....1231
7.2.1 Creating reports (RT Professional, RT Advanced, Panels, Comfort Panels)....1231
7.2.1.1 Principles for preparation of reports (RT Advanced, Panels, Comfort Panels)....1231
7.2.1.2 Principles for preparation of reports (RT Professional)....1234
7.2.1.3 Create a report (RT Professional, RT Advanced, Panels, Comfort Panels)....1238
7.2.1.4 Report design tools (RT Professional)....1240
7.2.1.5 Continuous alarm reporting (RT Professional)....1243
7.2.1.6 CSV file for trend output (RT Professional)....1244
7.2.1.7 CSV file for output of a table (RT Professional)....1247
7.2.1.8 Administration of detail pages (RT Professional, RT Advanced, Panels, Comfort Panels)..1250
7.2.2 Report output (RT Professional, RT Advanced, Panels, Comfort Panels)....1251
7.2.2.1 Report output in Runtime Advanced (RT Professional, RT Advanced, Panels, Comfort Panels)....1251
7.2.2.2 Report output in Runtime Professional (RT Professional)....1252
7.2.2.3 System reports and system print jobs (RT Professional)....1259
7.2.3 Working with objects (RT Professional, RT Advanced, Panels, Comfort Panels)....1263
7.2.3.1 Inserting an object (RT Professional, RT Advanced, Panels, Comfort Panels)....1263
7.2.3.2 Deleting an object (RT Professional, RT Advanced, Panels, Comfort Panels)....1264
7.2.3.3 Inserting several objects of the same type (RT Professional, RT Advanced, Panels, Comfort Panels)....1265
7.2.3.4 Positioning objects (RT Professional, RT Advanced, Panels, Comfort Panels)....1266
7.2.3.5 Moving an object forward or backward (Basic, Advanced, Professional) (RT Professional, RT Advanced, Panels, Comfort Panels)....1268
7.2.3.6 Resizing objects (RT Professional, RT Advanced, Panels, Comfort Panels)....1269
7.2.3.7 Selecting multiple objects (RT Professional, RT Advanced, Panels, Comfort Panels)....1270
7.2.3.8 Repositioning and resizing multiple objects (RT Professional, RT Advanced, Panels, Comfort Panels)....1272
7.2.3.9 Aligning objects (RT Professional, RT Advanced, Panels, Comfort Panels)....1273
7.2.3.10 Rotating objects (RT Professional, RT Advanced, Panels, Comfort Panels)....1274
7.2.3.11 Flipping objects (RT Professional, RT Advanced, Panels, Comfort Panels)....1276
7.2.4 Operation in Runtime (RT Professional, RT Advanced, Panels, Comfort Panels)....1277
7.2.4.1 Printing reports (RT Advanced, Panels, Comfort Panels)....1277
7.2.4.2 Printing reports (RT Professional)....1277

7.3 Objects in reports (RT Professional, RT Advanced, Panels, Comfort Panels)....1279

7.3.1 Overview (RT Professional, RT Advanced, Panels, Comfort Panels)....1279
7.3.2 Audit report (RT Advanced, Panels, Comfort Panels)....1279
7.3.3 Control Data (RT Professional)....1280
7.3.4 Control hard copy (RT Professional)....1281
7.3.5 Control Information (RT Professional)....1282
7.3.6 CSV provider trends (RT Professional)....1283
7.3.7 CSV provider tables (RT Professional)....1284
7.3.8 Date/time field (RT Advanced, Panels, Comfort Panels)....1285
7.3.9 Date/time field (RT Professional)....1286
7.3.10 I/O field (RT Professional, RT Advanced, Panels, Comfort Panels)....1288
7.3.11 f(t) trend view in reports (RT Professional)....1289
7.3.12 f(x) trend view in reports (RT Professional)....1290
7.3.13 Graphic view (RT Professional, RT Advanced, Panels, Comfort Panels)....1291
7.3.14 Graphic I/O field (RT Advanced, Panels, Comfort Panels)....1292
7.3.15 Hardcopy (RT Professional)....1292
7.3.16 Alarm view in reports (RT Professional)....1293
7.3.17 Alarm sequence report (RT Professional)....1294
7.3.18 Alarm report (RT Advanced, Panels, Comfort Panels)....1295
7.3.19 ODBC database field (RT Professional)....1297
7.3.20 ODBC database table (RT Professional)....1298
7.3.21 Project name (RT Professional)....1300
7.3.22 Report name (RT Professional)....1301
7.3.23 Recipe view in reports (RT Professional)....1302
7.3.24 Recipe report (RT Advanced, Panels, Comfort Panels)....1303
7.3.25 Page number (RT Advanced, Panels, Comfort Panels)....1304
7.3.26 Page number (RT Professional)....1305
7.3.27 Symbolic I/O field (RT Advanced, Panels, Comfort Panels)....1306
7.3.28 Table view in reports (RT Professional)....1306
7.3.29 Text field (RT Professional, RT Advanced, Panels, Comfort Panels)....1307

8 Configuring user administration....1309

8.1 Basics....1309
8.1.1 Field of application of the user administration (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1309
8.1.2 Form of the user administration (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1310
8.1.3 User (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1311
8.1.4 Users work area (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1312
8.1.5 User groups (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1313
8.1.6 User groups work area (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1314
8.1.7 Settings for the user administration (Basic Panels)....1316
8.1.8 Settings for the user administration (RT Advanced, Panels, Comfort Panels)....1319
8.1.9 Settings for the user administration (RT Professional)....1322
8.1.10 Target groups in user administration (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1323

8.2 Setting up the user administration....1324
8.2.1 Administering users for Runtime (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1324

8.2.1.1 Creating an authorization (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1324
8.2.1.2 Creating a user group (RT Advanced, Basic Panels, Panels, Comfort Panels)....1324
8.2.1.3 Creating a user group (RT Professional)....1325
8.2.1.4 Assigning an authorization (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1326
8.2.1.5 Creating users (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1326
8.2.1.6 Assigning a user to a user group (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1328
8.2.1.7 Managing users (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1328
8.2.1.8 Managing user groups (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1329
8.2.1.9 Changing the number of the user group (RT Advanced, Basic Panels, Panels, Comfort Panels)....1331
8.2.1.10 Logging on users dynamically (RT Professional)....1332
8.2.2 Managing users on the server....1333
8.2.2.1 Central user administration using SIMATIC Logon (RT Professional, RT Advanced, Panels, Comfort Panels)....1333
8.2.2.2 Authentication and certificate management in SIMATIC Logon....1335
8.2.2.3 Making settings on the server (RT Professional, RT Advanced, Panels, Comfort Panels)..1336
8.2.2.4 Logging on using SIMATIC Logon (RT Advanced, Panels, Comfort Panels)....1336
8.2.2.5 Logging on using SIMATIC Logon (RT Professional)....1338
8.2.3 Administering users for Web Client (RT Professional)....1338
8.2.3.1 Administering user groups for Web Client (RT Professional)....1338
8.2.3.2 Managing users for the Web client (RT Professional)....1340
8.2.4 Administering users in Runtime....1341
8.2.4.1 Users in runtime (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1341
8.2.4.2 User view (RT Professional, RT Advanced, Panels, Comfort Panels)....1342
8.2.4.3 Configuring a user view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1343
8.2.4.4 Creating users (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, Multipanels, Mobile Panels)....1344
8.2.4.5 Managing users in the complex user view (RT Professional, RT Advanced, Panels, Comfort Panels)....1346
8.2.4.6 Unlock locked out users (RT Advanced, Basic Panels, Panels, Comfort Panels)....1347
8.2.4.7 Logging on as a user (RT Advanced, Basic Panels, Panels, Comfort Panels)....1348
8.2.4.8 Logging on as a user (RT Professional)....1350
8.2.4.9 Exporting the user administration (RT Professional, RT Advanced, Panels, Comfort Panels)....1352
8.2.4.10 Importing the user administration (RT Professional, RT Advanced, Panels, Comfort Panels)....1353
8.2.5 Configuring access protection....1354
8.2.5.1 Access protection (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1354
8.2.5.2 Configuring operating authorizations (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1354

8.3 Reference....1356
8.3.1 Objects with access protection (Basic Panels)....1356
8.3.2 Objects with access protection (RT Advanced, Panels, Comfort Panels)....1356

8.3.3 Objects with access protection (RT Professional)....1357
8.3.4 Default user groups and authorizations (RT Advanced, Basic Panels, Panels, Comfort Panels)....1359
8.3.5 Default user groups and authorizations (RT Professional)....1359

8.4 Examples....1360
8.4.1 Example: Configuring a button with logon dialog box (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1360
8.4.2 Example: Logging the logon and logoff events (RT Advanced, Basic Panels, Panels, Comfort Panels)....1361
8.4.3 Example: Structure of user administration (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1362
8.4.4 Example: Creating and configuring authorizations (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1364
8.4.5 Example: Configuring a button with access protection (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1365
8.4.6 Example: Creating user groups and assigning authorizations: (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1366
8.4.7 Example: Creating users and assigning them to a user group (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1368

9 Working with system functions and Runtime scripting....1371

9.1 Principles....1371
9.1.1 Runtime scripting (RT Professional, RT Advanced, Panels, Comfort Panels)....1371
9.1.2 System functions (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1373
9.1.3 User-defined functions (RT Professional, RT Advanced, Panels, Comfort Panels)....1375
9.1.4 Local scripts (RT Professional)....1377

9.2 Working with Function Lists....1379
9.2.1 Basics of the function list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1379
9.2.2 Properties of a function list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1380
9.2.3 Configuring a function list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1380
9.2.4 Editing a function list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1382

9.3 Working with customized functions....1384
9.3.1 "Scripts" editor (RT Professional, RT Advanced, Panels, Comfort Panels)....1384
9.3.2 Calling system functions (RT Professional, RT Advanced, Panels, Comfort Panels)....1387
9.3.3 Calling customized functions (RT Professional, RT Advanced, Panels, Comfort Panels)..1388
9.3.4 Access to HMI tags (RT Professional, RT Advanced, Panels, Comfort Panels)....1390
9.3.5 Testing the syntax of customized functions (RT Professional, RT Advanced, Panels, Comfort Panels)....1393

9.4 Creating customized VB functions....1395
9.4.1 Accessing objects with VBS (RT Professional, RT Advanced, Panels, Comfort Panels) ..1395
9.4.2 Use of VB tags (RT Professional)....1395
9.4.3 Create customized VB functions (RT Professional, RT Advanced, Panels, Comfort Panels)....1398
9.4.4 Renaming customized VB functions (RT Professional, RT Advanced, Panels, Comfort Panels)....1400

9.4.5 Executing customized VB functions (RT Professional, RT Advanced, Panels, Comfort Panels)....1400
9.4.6 Transfer and return of values in VBS (RT Professional, RT Advanced, Panels, Comfort Panels)....1402

9.5 Creating user-defined C functions (RT Professional)....1404
9.5.1 Accessing objects with C (RT Professional)....1404
9.5.2 Creating a C-Header (RT Professional)....1404
9.5.3 Use of Global C-Tags (RT Professional)....1406
9.5.4 Calling functions from DLLs in a user-defined C function (RT Professional)....1407
9.5.5 Set up code page for C function (RT Professional)....1408
9.5.6 Create customized C functions (RT Professional)....1409
9.5.7 Renaming customized C functions (RT Professional)....1412
9.5.8 Executing customized C functions (RT Professional)....1413
9.5.9 Transfer and return of values in C (RT Professional)....1414

9.6 Protecting user-defined functions....1416
9.6.1 Setting up know-how protection (RT Professional, RT Advanced, Panels, Comfort Panels)....1416
9.6.2 Opening user-defined functions with know-how protection....1416
9.6.3 Removing know-how protection....1417
9.6.4 Changing a password....1417

9.7 Creating local scripts (RT Professional)....1419
9.7.1 Local scripts (RT Professional)....1419
9.7.2 Editor for local scripts (RT Professional)....1420
9.7.3 Converting a function list to a local script (RT Professional)....1421
9.7.4 Dynamizing an object property with a local script (RT Professional)....1423

9.8 Debugging....1424
9.8.1 Debugging of user-defined VB functions....1424
9.8.2 Principles of Debugging....1425
9.8.3 Activating the debugger....1427
9.8.4 Structure of VBScript files (RT Advanced, Panels, Comfort Panels)....1428
9.8.5 Structure of VBScript files (RT Professional)....1429
9.8.6 Names in the debugger (RT Professional)....1430
9.8.7 Breakpoints....1431
9.8.8 Working step-by-step....1432
9.8.9 Executing commands....1433
9.8.10 Determining tag and property values....1434
9.8.11 Diagnostics (RT Professional)....1435
9.8.11.1 Diagnostics (RT Professional)....1435
9.8.11.2 GSC diagnostics (RT Professional)....1436
9.8.11.3 Inserting a GSC diagnostics window in a screen (RT Professional)....1436

9.9 Runtime behavior in Runtime....1438
9.9.1 Executing user-defined functions in Runtime (RT Professional, RT Advanced, Panels, Comfort Panels)....1438
9.9.2 Executing a function list in Runtime (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1439
9.9.3 Execution of functions and local scripts in Runtime (RT Professional)....1439
9.9.4 Processing sequence for user-defined functions and system functions (RT Professional, RT Advanced, Panels, Comfort Panels)....1440
9.9.5 Making object properties dynamic in Runtime (RT Professional, RT Advanced, Panels, Comfort Panels)....1442

9.10 Examples....1443
9.10.1 Examples of VBS (RT Advanced, Panels, Comfort Panels)....1443
9.10.1.1 Example: Converting Fahrenheit into degrees Celsius (RT Advanced, Panels, Comfort Panels)....1443
9.10.1.2 Example: Converting inches into meters (RT Advanced, Panels, Comfort Panels)....1445
9.10.1.3 Example: Changing the operating mode on the HMI device with the current display (RT Advanced, Panels, Comfort Panels)....1447
9.10.2 Examples of VBS....1451
9.10.2.1 Overview (RT Professional)....1451
9.10.2.2 Example of accessing objects in the Screens editor (RT Professional)....1451
9.10.2.3 Example of defining object colors (RT Professional)....1452
9.10.2.4 Example of configuring a language switch (RT Professional)....1452
9.10.2.5 Example of deactivating Runtime (RT Professional)....1453
9.10.2.6 Example of configuring a screen change via a property (RT Professional)....1453
9.10.2.7 Example of configuring a diagnostics output via a trace (RT Professional)....1453
9.10.2.8 Example of writing tag values (RT Professional)....1455
9.10.2.9 Example of reading tag values (RT Professional)....1457
9.10.2.10 Example of writing object properties (RT Professional)....1459
9.10.2.11 Example of configuring the recipe view (RT Professional)....1462
9.10.2.12 Example of configuring the alarm view (RT Professional)....1463
9.10.2.13 Example of statistically evaluating tag values (RT Professional)....1465
9.10.2.14 Example of the start of a script at the server (Logging object) (RT Professional)....1466
9.10.2.15 Example for opening the PLC code view from the GRAPH overview (RT Professional)....1467
9.10.3 Examples of C (RT Professional)....1468
9.10.3.1 Overview (RT Professional)....1468
9.10.3.2 Example of accessing objects in the Screens editor (RT Professional)....1469
9.10.3.3 Example of defining object colors (RT Professional)....1469
9.10.3.4 Example of configuring a language switch (RT Professional)....1470
9.10.3.5 Example of deactivating Runtime (RT Professional)....1470
9.10.3.6 Example of configuring a screen change via a property (RT Professional)....1471
9.10.3.7 Example of configuring a diagnostics output via a trace (RT Professional)....1471
9.10.3.8 Example of writing tag values (RT Professional)....1472
9.10.3.9 Example of reading tag values (RT Professional)....1472
9.10.3.10 Example of writing object properties (RT Professional)....1473

9.11 Reference....1474
9.11.1 Function list....1474
9.11.1.1 Device dependency (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1474
9.11.1.2 System functions (RT Advanced, Basic Panels, Panels, Comfort Panels)....1505
9.11.1.3 System functions (RT Professional)....1656
9.11.2 Events (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1703
9.11.2.1 Overview (Basic Panels)....1703
9.11.2.2 Overview (RT Advanced, Panels, Comfort Panels)....1708
9.11.2.3 Overview (RT Professional)....1715
9.11.2.4 Events (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1726
9.11.3 WinCC Engineering V15.1 - Programming reference....1756

10 Mobile Panels (Panels, Comfort Panels)....1757

10.1 Basics (Panels, Comfort Panels)....1757
10.1.1 Field of application of the Mobile Panel Wireless (Panels, Comfort Panels)....1757
10.1.2 How does the transponder system work? (Panels, Comfort Panels)....1758

10.1.3 How does the RFID system work? (Panels, Comfort Panels)....1760
10.1.4 Zones (Panels, Comfort Panels)....1763
10.1.5 Zones work area at transponders (Panels, Comfort Panels)....1764
10.1.6 Zones work area on connection boxes (Panels, Comfort Panels)....1765
10.1.7 Effective ranges (Panels, Comfort Panels)....1766
10.1.8 Effective ranges working area (Panels, Comfort Panels)....1767
10.1.9 Effective ranges (RFID) (Panels, Comfort Panels)....1768
10.1.10 Work area for effective ranges (RFID) (Panels, Comfort Panels)....1769

10.2 Configuring Mobile Panels V2 for fail-safe operation (from V12 SP1) (Panels, Comfort Panels)....1772
10.2.1 Configuration overview (Panels, Comfort Panels)....1772
10.2.2 Creating PLC for fail-safe operation (Panels, Comfort Panels)....1772
10.2.3 Creating Mobile Panel (Panels, Comfort Panels)....1773
10.2.4 Creating a connection between the Mobile Panel and PLC (Panels, Comfort Panels)....1774

10.3 Configuring KTP Mobile Panels for fail-safe operation (Panels, Comfort Panels)....1775
10.3.1 Creating KTP Mobile Panel for fail-safe operation (Panels, Comfort Panels)....1775
10.3.2 Configuring the termination of the PROFIsafe connection (Panels, Comfort Panels)....1776

10.4 Working with zones on transponders (Panels, Comfort Panels)....1777
10.4.1 Configuring a zone in the transponder system (Panels, Comfort Panels)....1777
10.4.2 Displaying the screen on entering a zone (Panels, Comfort Panels)....1778
10.4.3 Displaying an object in relation to the zone (Panels, Comfort Panels)....1779

10.5 Working with zones on connection boxes (Panels, Comfort Panels)....1781
10.5.1 Configuring the zone of a connection box (Panels, Comfort Panels)....1781
10.5.2 Calling system functions when connected to the connection box. (Panels, Comfort Panels)....1782

10.6 Working with effective ranges (transponders) (Panels, Comfort Panels)....1784
10.6.1 Overview (Panels, Comfort Panels)....1784
10.6.2 Configuring an effective range (transponder) (Panels, Comfort Panels)....1785
10.6.3 Configuring the effective range name (Panels, Comfort Panels)....1786
10.6.4 Configuring additional Mobile Wireless objects (Panels, Comfort Panels)....1787

10.7 Working with effective ranges (RFID) (Panels, Comfort Panels)....1788
10.7.1 Overview (Panels, Comfort Panels)....1788
10.7.2 Configuring effective range (RFID) (Panels, Comfort Panels)....1789
10.7.3 Configuring the effective range name (RFID) (Panels, Comfort Panels)....1790

10.8 Reference (Panels, Comfort Panels)....1791
10.8.1 PROFIsafe address (Panels, Comfort Panels)....1791
10.8.2 Power Management (Panels, Comfort Panels)....1791
10.8.3 Zone ID / connection point ID (Panels, Comfort Panels)....1792

11 Planning tasks....1793

11.1 Basics....1793
11.1.1 Field of application of the Scheduler (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1793
11.1.2 Working with tasks and triggers (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1794
11.1.3 Work area of the "Scheduler" editor (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1794
11.1.4 Triggers (Basic Panels)....1795
11.1.5 Triggers (RT Professional, RT Advanced, Panels, Comfort Panels)....1796

11.1.6 Timers for cyclic triggers (RT Advanced, Panels, Comfort Panels)....1797
11.1.7 Tag triggers (RT Professional)....1798
11.1.8 Function list (Basic Panels)....1799
11.1.9 Function list (RT Professional, RT Advanced, Panels, Comfort Panels)....1799
11.1.10 Print task (RT Professional)....1799
11.1.11 Integrating local VB scripts (RT Professional)....1801
11.1.12 Integrating local C scripts (RT Professional)....1802

11.2 Planning tasks....1803
11.2.1 Planning tasks with acyclic triggers (RT Professional, RT Advanced, Panels, Comfort Panels)....1803
11.2.2 Planning tasks with cyclic triggers (RT Professional)....1804
11.2.3 Planning tasks with event triggers (RT Advanced, Basic Panels, Panels, Comfort Panels)..1806
11.2.4 Administer task (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1807

11.3 Examples for Runtime Professional (RT Professional)....1809
11.3.1 Example: Activate the start screen daily (RT Professional)....1809
11.3.2 Example: Printing a report once (RT Professional)....1810
11.3.3 Example: Reading the current user name (RT Professional)....1812
11.3.4 Example: Increment a tag (RT Professional)....1813

11.4 Examples for Runtime Advanced and Panels....1814
11.4.1 Example: Update user following change of user (RT Advanced, Basic Panels, Panels, Comfort Panels)....1814
11.4.2 Example: Weekly printout of a report (RT Advanced, Panels, Comfort Panels)....1815
11.4.3 Example: Changing the starting point of a task in Runtime (RT Advanced, Panels, Comfort Panels)....1816
11.4.4 Example: 1. Configuring a tag for Runtime (RT Advanced, Panels, Comfort Panels)....1817
11.4.5 Example: 2. Configuring a Date/time field (RT Advanced, Panels, Comfort Panels)....1818
11.4.6 Example: 3. Configuring archiving with a timer variable (RT Advanced, Panels, Comfort Panels)....1818

12 Using the diagnostics functions....1821

12.1 Configuring system diagnostics....1821
12.1.1 System diagnostics (RT Advanced, Basic Panels, Panels, Comfort Panels)....1821
12.1.1.1 Basic Panel basics (RT Advanced, Basic Panels, Panels, Comfort Panels)....1821
12.1.1.2 Panels and Runtime Advanced basics (RT Advanced, Basic Panels, Panels, Comfort Panels)....1824
12.1.1.3 Configuring system diagnostics objects (RT Advanced, Basic Panels, Panels, Comfort Panels)....1828
12.1.2 System diagnostics (RT Professional)....1837
12.1.2.1 Runtime Professional basics (RT Professional)....1837
12.1.2.2 Configuring system diagnostics objects (RT Professional)....1840
12.1.2.3 Example: System diagnostics with all objects (RT Professional)....1844

12.2 Monitoring functions for SIMATIC IPCs....1850

12.3 Supervising machinery and plants with ProDiag (RT Professional, RT Advanced, Panels, Comfort Panels)....1852
12.3.1 Basics of supervision with ProDiag (RT Professional, RT Advanced, Panels, Comfort Panels)....1852
12.3.2 Basics of supervising operands (RT Professional, RT Advanced, Panels, Comfort Panels)....1852
12.3.3 Requirements and licensing (RT Professional, RT Advanced, Panels, Comfort Panels) ..1854

12.3.4 Visualizing supervisions (RT Professional, RT Advanced, Panels, Comfort Panels)....1855
12.3.4.1 Objects for the supervision and diagnostics of plants (RT Professional, RT Advanced, Panels, Comfort Panels)....1855
12.3.4.2 Interaction of the objects for the supervision and diagnostics (RT Advanced, Panels, Comfort Panels)....1856
12.3.4.3 Interaction of the objects for the supervision and diagnostics (RT Professional)....1859
12.3.4.4 Configuring the ProDiag overview (RT Professional, RT Advanced, Panels, Comfort Panels)....1862
12.3.4.5 Configuring a GRAPH overview (RT Professional, RT Advanced, Panels, Comfort Panels)....1866
12.3.4.6 Configuring the PLC code view (RT Advanced, Panels, Comfort Panels)....1873
12.3.4.7 Configuring the PLC code view (RT Professional)....1886
12.3.4.8 Correcting display problems in the PLC code view (RT Professional, RT Advanced, Panels, Comfort Panels)....1900
12.3.5 Example: Visualization of the ProDiag functionality (RT Professional, RT Advanced, Panels, Comfort Panels)....1902
12.3.5.1 Procedures overview (RT Professional, RT Advanced, Panels, Comfort Panels)....1902
12.3.5.2 Requirements (RT Professional, RT Advanced, Panels, Comfort Panels)....1904
12.3.5.3 Creating screens (RT Professional, RT Advanced, Panels, Comfort Panels)....1904
12.3.5.4 Inserting diagnostic objects (RT Professional, RT Advanced, Panels, Comfort Panels)....1905
12.3.5.5 Configuring diagnostic objects (RT Professional, RT Advanced, Panels, Comfort Panels)....1906
12.3.5.6 Configuring the alarm view for ProDiag (RT Professional, RT Advanced, Panels, Comfort Panels)....1908
12.3.5.7 Configuring the jump from the GRAPH overview (RT Professional, RT Advanced, Panels, Comfort Panels)....1910
12.3.5.8 Configuring the jump from the ProDiag overview (RT Professional, RT Advanced, Panels, Comfort Panels)....1913
12.3.6 Initial value acquisition and criteria analysis (RT Professional, RT Advanced, Panels, Comfort Panels)....1915
12.3.6.1 Overview of initial value acquisition and criteria analysis (RT Professional, RT Advanced, Panels, Comfort Panels)....1915
12.3.6.2 Supported instructions (RT Professional, RT Advanced, Panels, Comfort Panels)....1917
12.3.6.3 Example of order of the criteria analysis (RT Professional, RT Advanced, Panels, Comfort Panels)....1919
12.3.6.4 Configuring the criteria analysis view (RT Advanced, Panels, Comfort Panels)....1921
12.3.6.5 Configuring the criteria analysis view (RT Professional)....1922
12.3.6.6 Outputting alarms with criteria (RT Professional, RT Advanced, Panels, Comfort Panels)....1923

13 Communicating with controllers....1927

13.1 WinCC Engineering V15.1 - Communication....1927

14 Using global functions....1929

14.1 Working with the HMI Device Wizard....1929
14.1.1 HMI device wizard basics (Basic Panels, Panels, Comfort Panels)....1929
14.1.2 HMI device wizard basics (RT Professional, RT Advanced)....1932

14.2 Working with libraries....1934
14.2.1 Basics on libraries....1934
14.2.2 Overview of the library view....1936
14.2.3 Master copies and types....1938
14.2.4 Libraries in WinCC....1939
14.2.5 Managing libraries....1941
14.2.5.1 Overview of the library management....1941

14.2.5.2 Opening library management....1942
14.2.5.3 Filtering types in the library management....1943
14.2.5.4 Creating a global library....1944
14.2.5.5 Saving a global library....1945
14.2.5.6 Opening a global library....1946
14.2.5.7 Displaying logs of global libraries....1947
14.2.5.8 Updating a project with the contents of a library....1947
14.2.5.9 Updating a library with the contents of another library....1948
14.2.5.10 Upgrading types of a library....1949
14.2.5.11 Exporting and importing library texts....1950
14.2.6 Managing objects in a library....1952
14.2.6.1 Displaying library objects....1952
14.2.6.2 Storing an object in a library....1956
14.2.6.3 Inserting a library object....1957
14.2.7 Using types and their versions....1958
14.2.7.1 State of type versions....1958
14.2.7.2 Generating a script as a type....1959
14.2.7.3 Generating a screen as a type....1959
14.2.7.4 Generating a style as a type....1960
14.2.7.5 Creating a style sheet as a type....1960
14.2.7.6 Create a new version of a type....1961

14.3 Importing and exporting project data....1963
14.3.1 Data exchange between projects and external applications....1963
14.3.2 Importing and exporting recipe data records....1964
14.3.2.1 Exporting recipe data records (RT Advanced, Basic Panels, Panels, Comfort Panels). ...1964
14.3.2.2 Importing recipe data records (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1966
14.3.2.3 Exporting recipe data records (RT Professional)....1967
14.3.2.4 Importing recipe data records (RT Professional)....1968
14.3.2.5 Editing exported recipe data records....1970
14.3.2.6 Format of data from recipe data records (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1970
14.3.3 Importing and exporting alarms....1971
14.3.3.1 Exporting alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1971
14.3.3.2 Importing alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)....1972
14.3.3.3 Importing alarms (RT Professional)....1974
14.3.3.4 Format of the analog alarm data (RT Advanced, Basic Panels, Panels, Comfort Panels)..1975
14.3.3.5 Format of the discrete alarm data (RT Advanced, Basic Panels, Panels, Comfort Panels)..1978
14.3.3.6 Format of the analog alarm data (RT Professional)....1980
14.3.3.7 Format of the discrete alarm data (RT Professional)....1983
14.3.4 Importing and exporting tags....1986
14.3.4.1 Exporting tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1986
14.3.4.2 Importing tags (WinCC V13 SP1 or higher) (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1987
14.3.4.3 Importing tags (prior to WinCC V13 SP1)....1989
14.3.4.4 Format of the tag data (RT Advanced, Basic Panels, Panels, Comfort Panels)....1990
14.3.4.5 Format of the tag data (RT Professional)....1993
14.3.5 Importing and exporting text lists....1995
14.3.5.1 Exporting text lists (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1995

14.3.5.2 Importing text lists (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1996
14.3.5.3 Format of text list data (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....1997
14.3.6 Importing and exporting project texts....2000
14.3.6.1 Importing project texts....2000
14.3.6.2 Exporting project texts....2000

14.4 Using cross-references....2003
14.4.1 General information about cross references (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2003
14.4.2 Special features when working with cross-reference lists in WinCC (RT Professional, RT Advanced, Panels, Comfort Panels)....2003
14.4.3 Textual cross-references (RT Professional, RT Advanced, Panels, Comfort Panels)....2004
14.4.4 Invalid cross-references (RT Professional, RT Advanced, Panels, Comfort Panels)....2006
14.4.5 Rewiring tags in the screens (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2008

14.5 Managing languages....2012
14.5.1 Languages in WinCC....2012
14.5.2 Language settings in the operating system....2013
14.5.3 Operating system settings for Asian languages....2014
14.5.4 Setting project languages....2015
14.5.4.1 Selecting the user interface language....2015
14.5.4.2 Enable project languages....2015
14.5.4.3 Selecting the reference language and editing language....2016
14.5.5 Creating one project in multiple languages....2018
14.5.5.1 Working with multiple languages....2018
14.5.5.2 Project text basics....2019
14.5.5.3 Translating texts directly....2020
14.5.5.4 Translating texts using reference texts....2021
14.5.5.5 Exporting project texts....2022
14.5.5.6 Importing project texts....2024
14.5.6 Using language-specific graphics....2025
14.5.6.1 "Project graphics" editor....2025
14.5.6.2 Storing an image in the graphics library....2026
14.5.6.3 Storing an external image in the graphics library....2027
14.5.7 Languages in Runtime....2029
14.5.7.1 Languages and fonts in Runtime....2029
14.5.7.2 Methods for language switching....2030
14.5.7.3 Enabling the runtime language....2031
14.5.7.4 Setting the runtime language order for language switching....2032
14.5.7.5 Setting the default font for a runtime language....2034
14.5.7.6 Standardizing font for all languages....2035
14.5.7.7 Selecting the log language....2035
14.5.7.8 Specific features of Asian and Eastern languages in Runtime (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)....2036
14.5.8 Example of multilingual configuration....2037
14.5.8.1 Example: Configuring a button for language switching....2037
14.5.8.2 Example: Configuring a button in multiple languages....2038
14.5.8.3 Example: Configuring a button for language switching for each runtime language....2039

14.6 Replacing devices....2041
14.6.1 Basics (RT Advanced, Basic Panels, Panels, Comfort Panels)....2041

14.6.2 Customizing the configuration of connections....2042
14.6.3 Device-specific functions....2043
14.6.3.1 Device-specific functions (RT Advanced, Basic Panels, Panels, Comfort Panels)....2043
14.6.3.2 Key assignment when replacing devices (Basic Panels)....2043
14.6.4 Adjusting screens to the new device....2046
14.6.4.1 Screen adjustment options (RT Advanced, Basic Panels, Panels, Comfort Panels)....2046
14.6.4.2 Fit objects to contents (RT Advanced, Basic Panels, Panels, Comfort Panels)....2048
14.6.4.3 Specifying the position of screen objects (RT Advanced, Basic Panels, Panels, Comfort Panels)....2050
14.6.5 Example: Replacing devices....2051
14.6.5.1 Example: Procedures overview (RT Advanced, Panels, Comfort Panels)....2051
14.6.5.2 Example: Adjusting screens (RT Advanced, Panels, Comfort Panels)....2051
14.6.5.3 Example: Replacing devices (RT Advanced, Panels, Comfort Panels)....2052

14.7 Copying between devices and editors....2055
14.7.1 Basics....2055
14.7.1.1 Basics (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2055
14.7.1.2 Copy and paste options (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2056
14.7.1.3 Unsupported objects and functionalities (Professional) (RT Professional)....2057
14.7.1.4 Unsupported objects and functionalities (RT Advanced, Panels, Comfort Panels)....2057
14.7.2 Copying and pasting....2059
14.7.2.1 Copying objects with linked objects (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2059
14.7.2.2 Linked objects copied automatically (Basic Panels)....2059
14.7.2.3 Linked objects copied automatically (RT Advanced, Panels, Comfort Panels)....2060
14.7.2.4 Copying HMI devices with HMI connections....2060
14.7.2.5 Copying linked screen objects with user data types....2061
14.7.2.6 Drag & drop from the details view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2061
14.7.2.7 Copying screens (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2062
14.7.2.8 Copying recipes within an HMI device (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2063
14.7.2.9 Differences between Runtime Advanced and Panels and Runtime Professional (RT Professional, RT Advanced, Panels, Comfort Panels)....2063
14.7.3 Copying and pasting between Runtime Advanced and Panels and Runtime Professional....2064
14.7.3.1 Basics....2064
14.7.3.2 System functions....2065
14.7.3.3 Limitations during copying....2066
14.7.3.4 Objects in screens....2067
14.7.3.5 Screen objects....2067
14.7.3.6 Connections....2069
14.7.3.7 Copying tags....2070
14.7.3.8 Tag data types and properties....2071
14.7.3.9 Structures....2072
14.7.3.10 Alarms....2075
14.7.3.11 Logs....2080
14.7.3.12 Copying VB scripts....2083
14.7.3.13 Copying recipes between Runtime Advanced and Panels and Runtime Professional....2084
14.7.3.14 Copying cycles....2085
14.7.3.15 Reports....2086
14.7.3.16 Text and graphics lists....2088

14.7.3.17 User management....2089
14.7.3.18 Scheduled tasks....2090
14.7.4 Copying between different RT and ES versions (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2091

14.8 Using WinCC version compatibility....2094
14.8.1 Basics on version compatibility (RT Advanced, Basic Panels, Panels, Comfort Panels)....2094
14.8.2 Basic information on version compatibility (RT Professional)....2096
14.8.3 Upgrading projects (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2097
14.8.4 Upgrading a global library (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2098
14.8.5 Editing projects of a previous WinCC version (RT Advanced, Basic Panels, Panels, Comfort Panels)....2100
14.8.6 Changing between device versions (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2101
14.8.7 Changing the device version (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2102

14.9 Managing colors centrally....2104
14.9.1 Basic principles for color management....2104
14.9.2 Finding and replacing colors....2104

14.10 Load files using the "HMI files" editor....2107

14.11 Exchanging controller data from other projects....2110

14.12 Recognizing the project status by the SIMATIC WinCC RT symbol....2111

15 Compiling and loading....2113

15.1 Establishing a connection to the HMI device....2113

15.2 Basic Panels....2115
15.2.1 Runtime settings....2115
15.2.1.1 Runtime settings....2115
15.2.1.2 Configuring display in Runtime (Comfort Panels)....2117
15.2.1.3 Configuring operation in Runtime....2118
15.2.1.4 Compatibility check with the controller (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)....2120
15.2.1.5 Password protection in Runtime....2120
15.2.2 Overview of compiling and loading projects (RT Advanced, Panels, Comfort Panels)....2122
15.2.3 Compiling a project (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2124
15.2.4 Simulating projects....2125
15.2.4.1 Simulation basics....2125
15.2.4.2 Simulating a project (RT Advanced, Panels, Comfort Panels)....2126
15.2.4.3 Working with the tag simulator (RT Advanced, Basic Panels, Panels, Comfort Panels)....2128
15.2.4.4 Simulation restrictions....2130
15.2.5 Loading projects....2130
15.2.5.1 Overview for loading of projects....2130
15.2.5.2 Loading a project....2132
15.2.5.3 Loading project from external storage medium (Basic Panels)....2135
15.2.6 Compiling and loading with Multiuser Engineering....2136
15.2.6.1 Compiling and loading with multiuser engineering (overview) (RT Advanced, Basic Panels, Panels, Comfort Panels)....2136

15.2.6.2 Compiling in the server project view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2138
15.2.6.3 Compiling in the local session (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2139
15.2.7 Starting Runtime....2140
15.2.7.1 Starting Runtime on the HMI device (Basic Panels)....2140
15.2.8 Error messages during loading of projects (Basic)....2141
15.2.9 Adapting the project for another HMI device....2142
15.2.10 Reducing the project size (RT Advanced, Basic Panels, Panels, Comfort Panels)....2143
15.2.11 Basics of operating in Runtime....2145
15.2.11.1 Overview (Basic Panels)....2145
15.2.11.2 Operation with the touch screen....2146
15.2.11.3 Operation with keys....2150
15.2.11.4 Navigating in the display (BS)....2153
15.2.11.5 Triggering an action....2154
15.2.11.6 Entering a value....2154
15.2.11.7 Moving operator controls....2155
15.2.11.8 Displaying infotext....2156
15.2.11.9 Changing Runtime language....2157
15.2.12 Entering bar codes with optical handheld readers....2157
15.2.13 Security on the HMI device....2159
15.2.13.1 Signature basics (RT Advanced, Basic Panels, Panels, Comfort Panels)....2159
15.2.13.2 Switching off the signature check (RT Advanced, Basic Panels, Panels, Comfort Panels)..2160

15.3 Runtime Advanced and Panels....2161
15.3.1 Runtime settings....2161
15.3.1.1 Runtime settings....2161
15.3.1.2 Configuring display in Runtime (Comfort Panels)....2163
15.3.1.3 Configuring operation in Runtime....2165
15.3.1.4 Compatibility check with the controller (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)....2167
15.3.1.5 Setting time base (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)..2167
15.3.1.6 Printing in Runtime (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)..2168
15.3.1.7 Password protection in Runtime....2168
15.3.2 Overview of compiling and loading projects (RT Advanced, Panels, Comfort Panels)....2170
15.3.3 Compiling a project (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2172
15.3.4 Simulating projects....2173
15.3.4.1 Simulation basics (RT Professional, RT Advanced, Panels, Comfort Panels)....2173
15.3.4.2 WinCC Runtime Advanced simulation (RT Advanced, Panels, Comfort Panels)....2174
15.3.4.3 Simulating a project (RT Advanced, Panels, Comfort Panels)....2175
15.3.4.4 Simulating a screen (RT Professional, RT Advanced, Panels, Comfort Panels)....2177
15.3.4.5 Working with the tag simulator (RT Advanced, Basic Panels, Panels, Comfort Panels)....2177
15.3.4.6 Simulation restrictions....2179
15.3.5 Loading projects....2179
15.3.5.1 Overview for loading of projects (Panels, Comfort Panels)....2179
15.3.5.2 Overview for loading of projects (RT Advanced)....2182
15.3.5.3 Loading a project (RT Advanced, Basic Panels, Panels, Comfort Panels)....2184
15.3.5.4 Generating a "Pack&Go" file (Basic Panels, Panels, Comfort Panels)....2188
15.3.5.5 Downloading projects from a "Pack&Go" file to the HMI device (Basic Panels, Panels, Comfort Panels)....2189
15.3.5.6 Loading via USB interface (RT Advanced, Panels, Comfort Panels)....2190
15.3.5.7 Load project from external storage medium (RT Advanced, Panels, Comfort Panels)....2191

15.3.5.8 Installing a USB driver in Windows 7 (RT Advanced, Panels, Comfort Panels).....2194
15.3.6 Compiling and loading with Multiuser Engineering.....2195
15.3.6.1 Compiling and loading with multiuser engineering (overview) (RT Advanced, Basic Panels, Panels, Comfort Panels).....2195
15.3.6.2 Compiling in the server project view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni).....2197
15.3.6.3 Compiling in the local session (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni).....2198
15.3.7 Starting Runtime.....2199
15.3.7.1 Starting Runtime on the configuration PC (RT Advanced, Basic Panels, Panels, Comfort Panels).....2199
15.3.7.2 Starting Runtime Advanced and Panel Runtime (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni).....2200
15.3.8 Error messages during loading of projects.....2201
15.3.9 Adapting the project for another HMI device.....2203
15.3.10 Reducing the project size (RT Advanced, Basic Panels, Panels, Comfort Panels).....2204
15.3.11 Supported network cameras and settings (RT Professional, RT Advanced, Panels, Comfort Panels).....2205
15.3.12 Viewing memory card data.....2209
15.3.12.1 Basics (Basic Panels, Panels, Comfort Panels).....2209
15.3.12.2 Working with backups.....2209
15.3.13 Working with HMI device images.....2212
15.3.13.1 Viewing HMI device images.....2212
15.3.13.2 Deleting HMI device images.....2213
15.3.13.3 Creating HMI device images on memory card.....2214
15.3.13.4 HMI device images V15 and higher.....2216
15.3.14 Basics of operating in Runtime.....2218
15.3.14.1 Overview (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels).....2218
15.3.14.2 Operation with the touch screen.....2218
15.3.14.3 Operation with keys.....2228
15.3.14.4 Navigating in the display (BS).....2232
15.3.14.5 Triggering an action.....2233
15.3.14.6 Entering a value.....2234
15.3.14.7 Moving operator controls.....2234
15.3.14.8 Displaying infotext.....2235
15.3.14.9 Changing Runtime language.....2236
15.3.14.10 Web browser of WebKit engine: Overview (Panels, Comfort Panels).....2236
15.3.15 Entering bar codes with optical handheld readers.....2242
15.3.16 Security on the HMI device.....2243
15.3.16.1 Signature basics (RT Advanced, Basic Panels, Panels, Comfort Panels).....2243
15.3.16.2 Download of add-ons from third parties (RT Advanced, Panels, Comfort Panels).....2244
15.3.16.3 Switching off the signature check (RT Advanced, Basic Panels, Panels, Comfort Panels)..2244

15.4 Runtime Professional.....2246
15.4.1 Runtime settings.....2246
15.4.1.1 Settings in the Runtime software.....2246
15.4.1.2 Configuring display in Runtime (Professional).....2248
15.4.1.3 Configuring operation in Runtime.....2250
15.4.1.4 Setting up the start sequence.....2258
15.4.1.5 Activating time synchronization (RT Professional).....2259
15.4.1.6 Setting the time base.....2264
15.4.1.7 Configuring persistence.....2264
15.4.1.8 Working with service mode.....2265

15.4.1.9 Password protection in Runtime....2269
15.4.2 Overview of compiling and loading projects (RT Professional)....2272
15.4.3 Compiling a project (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2274
15.4.4 Simulating projects....2275
15.4.4.1 Simulation basics (RT Professional, RT Advanced, Panels, Comfort Panels)....2275
15.4.4.2 Runtime Professional simulation (RT Professional)....2275
15.4.4.3 Simulating a project (RT Professional)....2276
15.4.4.4 Simulating a screen (RT Professional, RT Advanced, Panels, Comfort Panels)....2278
15.4.4.5 Working with the tag simulator (RT Professional)....2278
15.4.4.6 Simulation restrictions....2279
15.4.5 Loading projects....2280
15.4.5.1 Overview for loading of projects (Panels, Comfort Panels)....2280
15.4.5.2 Overview for loading of projects (RT Advanced)....2283
15.4.5.3 Overview for loading of projects (RT Professional)....2285
15.4.5.4 Downloading the project to an HMI device (RT Professional)....2287
15.4.5.5 Downloading the project to a file system....2290
15.4.5.6 Online delta transfer for Runtime Professional....2291
15.4.5.7 Updating a runtime project on an HMI device to V13 SP1 (RT Professional)....2294
15.4.6 Compiling and loading with Multiuser Engineering....2295
15.4.6.1 Compiling and loading with Multiuser Engineering (overview) (RT Professional)....2295
15.4.6.2 Compiling in the server project view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2297
15.4.6.3 Compiling in the local session (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)....2298
15.4.6.4 Rules on online delta loading....2298
15.4.6.5 Online delta loading from the server project view (RT Professional)....2300
15.4.6.6 Online delta loading from the local session (RT Professional)....2301
15.4.7 Starting Runtime....2303
15.4.7.1 Starting Runtime on the configuration PC (RT Professional)....2303
15.4.7.2 Starting Runtime Professional (RT Professional)....2304
15.4.8 Basics of operating in Runtime....2307
15.4.8.1 Overview of operating in Runtime (Professional)....2307
15.4.8.2 Operation with the touch screen....2308
15.4.8.3 Operation with keys....2316
15.4.8.4 Navigating in the display (BS)....2320
15.4.8.5 Triggering an action....2321
15.4.8.6 Entering a value....2321
15.4.8.7 Moving operator controls....2322
15.4.8.8 Displaying infotext (Professional)....2323
15.4.8.9 Changing Runtime language....2323
15.4.8.10 Runtime data....2324

15.5 Jumping to the configuration....2325
15.5.1 Jumping to the configuration....2325
15.5.2 Read mode in the TIA Portal....2326

15.6 Jumping to the configuration (RT Professional)....2328
15.6.1 Jumping to the configuration (RT Professional)....2328
15.6.2 Read mode in the TIA Portal (RT Professional)....2329
15.6.3 Configuring read mode (RT Professional)....2331
15.6.4 Example: Configuring read mode using the API function (RT Professional)....2332

15.7 Servicing the HMI device....2334

15.7.1 Overview of HMI device maintenance (Basic Panels)....2334
15.7.2 Overview of HMI device maintenance tasks (Basic Panels)....2335
15.7.3 Maintenance overview of the HMI device (RT Adv, RT Prof)....2336
15.7.4 ProSave....2337
15.7.5 Backup of HMI data....2337
15.7.6 Backing up and restoring data of the HMI device....2339
15.7.7 Updating the operating system....2340
15.7.8 Updating the operating system on the HMI device....2342
15.7.9 Updating the operating system of the HMI device from a data carrier....2343
15.7.10 Transferring license keys....2345
15.7.11 Managing licenses....2345
15.7.12 Installing and uninstalling an option....2347
15.7.13 Loading Asian fonts to the HMI device (RT Advanced, Panels, Comfort Panels)....2348

15.8 Printer driver (Comfort Panels)....2349
15.8.1 Validity (Comfort Panels)....2349
15.8.2 Supported HMI devices (Comfort Panels)....2349
15.8.3 Setting up the printer driver (Comfort Panels)....2350
15.8.3.1 Postscript (Comfort Panels)....2350
15.8.3.2 HTML (Comfort Panels)....2351
15.8.3.3 PDF (Comfort Panels)....2352
15.8.4 Installation (Comfort Panels)....2353
15.8.5 Transferring the Options (Comfort Panels)....2353

16 Performance features....2355

16.1 General technical specifications....2355
16.1.1 Permitted characters....2355
16.1.2 Recommended printers....2356
16.1.3 Printing via print server....2357
16.1.4 Memory requirement of recipes....2358
16.1.5 Memory requirements of recipes for Basic Panels....2359

16.2 Performance features of the devices....2361
16.2.1 Basic Panel (Basic Panels)....2361
16.2.2 Basic Panel 2nd Generation (Basic Panels)....2364
16.2.3 Mobile Panel (Panels, Comfort Panels)....2368
16.2.4 Comfort Panel (Panels, Comfort Panels)....2373
16.2.5 WinCC Runtime Advanced (RT Advanced)....2379
16.2.6 WinCC Runtime Professional (RT Professional)....2384
16.2.7 Orientation help for performance in Runtime (WinCC Runtime Professional)....2388
16.2.8 Orientation help for communication in Runtime (WinCC Runtime Professional)....2390

17 Options....2393

17.1 WinCC Engineering V15.1 - Options....2393

18 Migrating to WinCC in the TIA Portal....2395

18.1 Overview....2395

18.2 WinCC V7 (RT Professional)....2396
18.2.1 Actions (RT Professional)....2396
18.2.2 User objects (RT Professional)....2397
18.2.3 User administration (RT Professional)....2399
18.2.4 Direct connection (RT Professional)....2402
18.2.5 Dynamic Wizard (RT Professional)....2403

18.2.6 Dynamics dialog (RT Professional)....2405
18.2.7 Working efficiently (RT Professional)....2407
18.2.8 Structures and structure tags (RT Professional)....2409

18.3 WinCC flexible (RT Advanced, Basic Panels, Panels, Comfort Panels)....2411
18.3.1 Libraries (RT Advanced, Basic Panels, Panels, Comfort Panels)....2411
18.3.2 Screens and templates (RT Advanced, Basic Panels, Panels, Comfort Panels)....2413
18.3.3 Scripts in faceplates (RT Advanced, Basic Panels, Panels, Comfort Panels)....2415
18.3.4 Synchronization of recipes (RT Advanced, Basic Panels, Panels, Comfort Panels)....2416
18.3.5 Special considerations for converting (RT Advanced, Basic Panels, Panels, Comfort Panels)....2417

Index....2419

Creating screens

1

1.1 Basics

1.1.1 Screen basics (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

In WinCC you create screens that an operator can use to control and monitor machines and plants. When you create your screens, the object templates included support you in visualizing processes, creating images of your plant, and defining process values.

Application example

The figure shows a screen that was created in WinCC. With the help of this screen, the operator can operate and monitor the mixing station of a fruit juice manufacturing system. Fruit juice base is supplied from various tanks to a mixing unit. The screen indicates the fill level of the tanks.

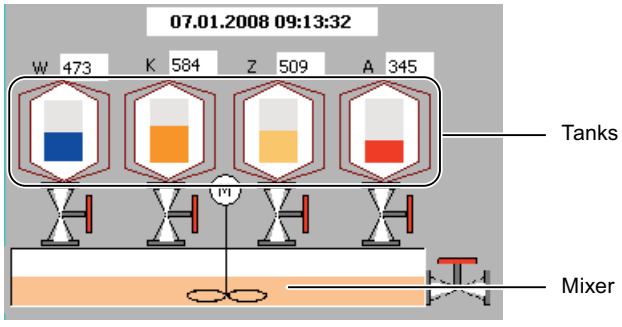

Screen design

Insert an object you need to represent a process into your screen. Configure the object to correspond to the requirements of your process.

A screen may consist of static and dynamic elements.

- Static elements such as text or graphic objects do not change their status in runtime. The tank labels (W, K, Z, A) shown in this example of a mixing unit are static elements.
- Dynamic elements change their status based on the process. Visual current process values as follows:
  - From the memory of the PLC
  - From the memory of the HMI device in the form of alphanumeric displays, trends and bars.

  Input fields on the HMI device are also considered dynamic objects. The fill level values of the tanks in our example of a mixing plant are dynamic objects.

Process values and operator inputs are exchanged between the controller and the HMI device via tags.

Screen properties

The screen layout is determined by the features of the HMI device you are configuring. It corresponds with the layout of the user interface of this device. If the set HMI device has function keys, the screen shows these function keys. Other properties such as the screen resolution, fonts and colors are also determined by the characteristics of the selected HMI.

Function keys

A function key is a key on the HMI device to which you can assign one or several functions in WinCC. The functions are triggered as soon as the operator presses the key on the HMI device.

A function key can be assigned global or local functions:

Global function keys always trigger the same action, regardless of the currently displayed screen.

Local function keys trigger different actions depending on the currently displayed screen on the HMI device. This assignment applies only to the screen in which you have defined the function key.

Navigation

You configure a screen navigation to enable the operator to call a screen on the HMI device in Runtime.

- The "Screen" editor is used to configure buttons for calling other screens.
- You use the "Global Screen" editor to configure globally assigned function keys.

See also

Basic principles of graphics lists (Page 189)

Screen basics (Page 39)

Device-dependent functional scope of screens (Page 40)

Task cards (Page 42)
Steps (Page 45)
Basics on working with templates (Page 53)
Basics on working with layers (Page 256)
Overview of objects (Page 93)
Basics on text lists (Page 176)
Basics on dynamizing screens (Page 205)
Basics on libraries (Page 1934)
Basics on faceplates (Page 265)
Print job/script diagnostics (Page 352)
Basics on screen navigation (Page 301)

1.1.2 Screen basics (RT Professional)

Introduction

In WinCC you create screens that an operator can use to control and monitor machines and plants. When you create your screens, the object templates included support you in visualizing processes, creating images of your plant, and defining process values.

Application example

The figure shows a screen that was created in WinCC. With the help of this screen, the operator can operate and monitor the mixing station of a fruit juice manufacturing system. Fruit juice base is supplied from various tanks to a mixing unit. The fill levels of the tanks are indicated.

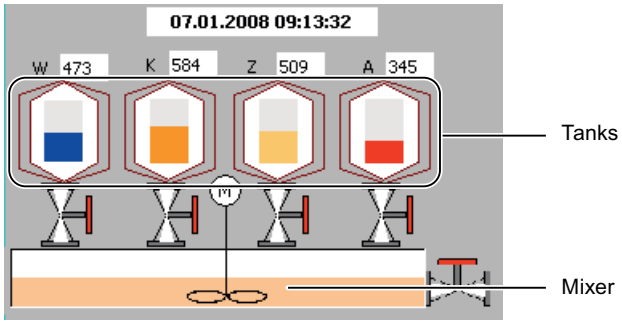

Screen design

Insert an object you need to represent a process into your screen. Configure the object to correspond to the requirements of your process.

A screen may consist of static and dynamic elements:

- Static elements such as text or graphic objects in the screen above do not change their status in Runtime. The tank labels (W, K, Z, A) shown in this example of a mixing unit are static elements.
- Dynamic elements change their status based on the process. Visual current process values as follows:
  - From the memory of the PLC
  - From the memory of the HMI device in the form of alphanumeric displays, trends and bars.

  Input fields on the HMI device are also considered dynamic objects. The fill level values of the tanks in our example of a mixing plant are dynamic objects.

Process values and operator inputs are exchanged between the controller and the HMI device via tags.

Navigation

You configure a screen navigation to enable the operator to call a screen on the HMI device in Runtime.

- You use the "Menus & Toolbars" editor to configure the navigation between the screens.
- The "Screen" editor is used to configure buttons for calling other screens.

See also

Screen basics (Page 37)

Device-dependent functional scope of screens (Page 40)

1.1.3 Device-dependent functional scope of screens (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The functions of an HMI device determine the display of the device in WinCC and the scope of functions of the editors.

The following screen properties are determined by the functions of the selected HMI device:

- Device layout
- Screen resolution
- Number of colors
- Fonts
- Objects available

Device layout

The device layout of a screen forms the image of the HMI device in your configuration. The device layout of the screen shows all the function keys available on the HMI device, for example.

Screen resolution

The screen resolution is determined by the different display dimensions of the various operator panels. You can only change the screen resolution if you configure the "WinCC Runtime Advanced" or "WinCC Runtime Professional" HMI device.

Number of colors

You can assign colors to the screen objects. The number of possible colors is determined by the color depth and specific colors supported on the selected HMI device.

Fonts

You can customize the appearance of the texts in all the screen objects that contain static or dynamic text. How to highlight individual texts in a screen. Select the font, font style and size, and set additional effects such as underscoring, for example.

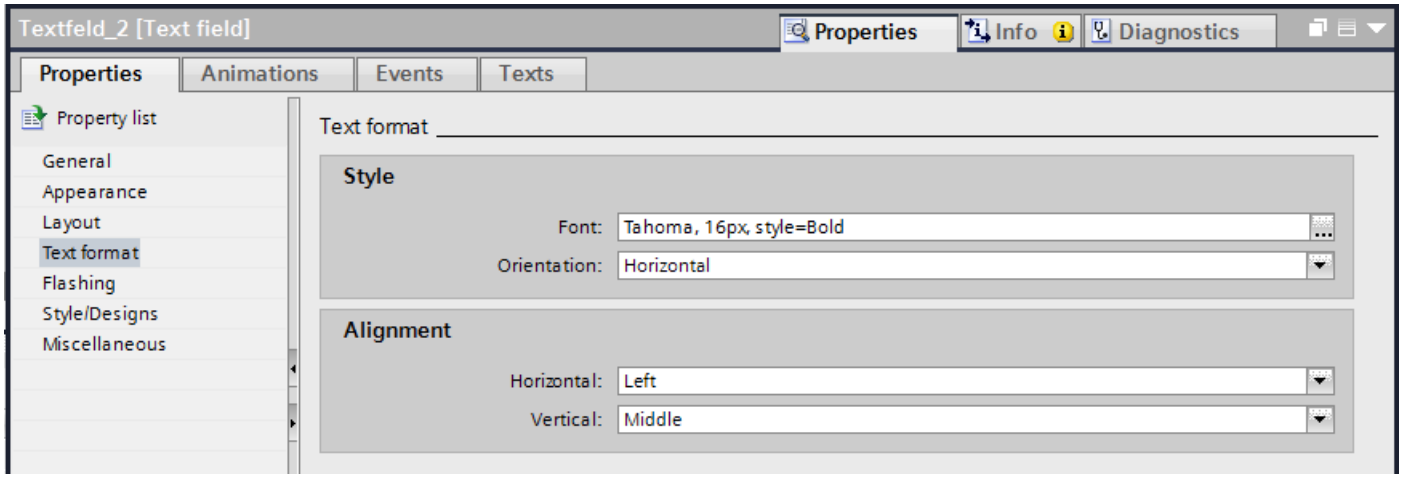

The settings for the text markups such as font style and effects always refer to the entire text of a screen object. That is, you can display the complete title in bold format, but not its individual characters or words, for example.

Objects available

Some of the screen objects can not be configured globally for all HMI devices. These screen objects are not displayed in the "Tools" task card. For a KTP 1000 touch panel unit you can not configure a slider, for example.

See also

Screen basics (Page 39)

1.1.4 Elements and basic settings

1.1.4.1 Task cards

Introduction

The following task cards are available in the "Screens" editor:

- Toolbox: Display and operating objects
- Animations: Templates for dynamic configuration
- Layout: Aid for customizing the display
- Libraries: Administration of the project library and of the global libraries

Note
WinCC Basic

The "Animations" task card is not available in WinCC Basic.

Toolbox

The "Toolbox" task card contains objects in different panes:

- Basic objects
- Elements
- Controls
- User controls (optional)
- Graphics

You paste objects from the palettes into your screens by drag&drop or a double click. The objects available for selection are determined by the features of the HMI device you are configuring. The following icons are used to change the display mode:

Icon	Meaning
	Displays the objects as a list.
	Displays the objects as a graphic.

Animations

The "Animations" task card contains the possible dynamizations of a screen object in the palettes. You paste the animations to a screen object by drag&drop or a double click from the "Movements", "Display" and "Tag Binding" palettes.

Layout

The "Layout" task card contains the following panes for displaying objects and elements:

- Layers: Serves to manage screen object layers The layers are displayed in a tree view and contain information about the active layer and the visibility of all layers.
- Grid: You specify whether you want to align the objects to a grid or to other objects and set the grid size for a grid.
- Objects out of range: Objects that lie outside the visible area are displayed with name, position and type.

Libraries

The "Libraries" task card show the following libraries in separate panes:

- Project library: The project library is stored together with the project.
- Global library: The global library is stored in a separate file in the specified path on your configuration PC.

See also

Zooming the view (Page 43)

Moving the view (Page 44)

Screen basics (Page 37)

Steps (Page 45)

Basics on working with templates (Page 53)

1.1.4.2 Zooming the view

Introduction

To view a small screen section in closer detail, use the zoom tool to magnify the screen in the working area. The maximum zoom amounts to 800%.

You can zoom with the toolbar in the work area or with the "Layout > Zoom" task card.

There are different ways to increase the screen, for example, with the zoom factor or by adapting the work area to the height of the screen.

Requirement

The screen is opened.

Procedure

Proceed as follows to zoom a view with the selection frame:

1. Click the toolbar button.
2. Use the mouse to draw a selection frame in the screen.

After you have released the mouse button, the section enclosed by the selection frame is zoomed to fit the complete work area.

Alternatively, use the slider in the lower right-hand corner of the screen.

Result

The selected screen section is magnified.

See also

Task cards (Page 42)

1.1.4.3 Moving the view

Introduction

To display only a section of the entire screen in the work area, use the icon of the "Screens" editor:

Requirement

- A screen is open.
- The view shows only a screen section.

Procedure

To move the view:

1. Click the icon at the bottom right corner of the work area and press the left mouse button. A miniature view of the full screen is shown. An orange frame shows the currently selected area.
2. Hold down the mouse button and drag the frame to the desired area.

Note

The screen is scrolled when you drag a screen object from the visible to a currently hidden section.

See also

Task cards (Page 42)

1.1.5 Working with screens

1.1.5.1 Steps (RT Advanced, Basic Panels, Panels, Comfort Panels)

Steps

To create screens, you need to take the following initial steps:

- Plan the structure of the process visualization: Number of screens and their layout
  Example: Subprocesses are visualized in separate screens, and merged in a master screen.
- Define your screen navigation control strategies.
- Adapt the templates and the global screen.
  You define objects centrally and assign function keys for example.
- Create the screens. Use the following options of efficient screen creation:
  - Working with libraries
  - Working with layers
  - Working with faceplates

See also

Steps (Page 46)

Creating a new screen (Page 47)

Managing screens (Page 48)

Defining the start screen of the project (Page 51)

Screen basics (Page 37)

Basics on working with templates (Page 53)

Task cards (Page 42)

1.1.5.2 Steps (RT Professional)

Steps

To create screens, you need to take the following initial steps:

- Plan the structure of the process visualization: Number of screens and their layout Example: Subprocesses are visualized in separate screens, and merged in a master screen.
- Define your screen navigation control strategies.
  Create the menus and toolbars to navigate between the screens.
- Create the screens. Use the following options to achieve efficient screen configuration:
  - Working with libraries
  - Working with faceplates
  - Working with layers
  - Working with designs

See also

Steps (Page 45)

Basics on working with designs (Page 70)

1.1.5.3 Creating a new screen (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Create screens to display processes in your system.

Requirement

- The project has been created.
- The Inspector window is open.

Procedure

1. Double-click "Screens > Add New Screen" in the project navigation.
   The screen is generated in the project and appears in your view. The screen properties are shown in the Inspector window.
2. Enter a meaningful name for the screen.
3. Configure the screen properties in the Inspector window:
   - Specify whether and on which template the screen is based.
   - Set the "Background Color" and the "Screen Number."
   - Specify a documenting text under "Tooltip".
   - Specify the layers to be displayed under "Layers" in the engineering system.
   - Select dynamic screen update under "Animations."
   - Select "Events" to define which functions you want to execute in Runtime when you call and exit the screen or at other events.

     Note

     Not all HMI devices support the "Visibility" animation.

Result

You created the screen in your project. You can now paste objects and control elements from the "Tools" task card and assign function keys in further work steps.

See also

Steps (Page 45)

1.1.5.4 Creating a new screen (RT Professional)

Introduction

Create screens to display processes in your system.

Requirement

- The project has been created.
- The Inspector window is open.

Procedure

To create a new screen:

1. Double-click "Screens > Add New Screen" in the project tree.
   The screen is generated in the project and appears in your view. The screen properties are shown in the Inspector window.
2. Enter a meaningful name for the screen, such as "Overview_Diagram."
   Do not use the special characters ?, . , ", /, \, *, <, >
   This name is language-neutral.
3. If required, enter a descriptive display name. The name is language-specific and can be translated for the required languages. You can output the display name in a text field with the SetPropertyByProperty system function, for example.
4. Configure the screen in the Inspector window:
   - Set the background color and the grid color under "General."
   - Specify a tooltip providing additional information.
   - Specify the size of the screen under "Layout."
   - Define the pattern and the color of a fill pattern under "Appearance."
   - Specify the layers to be displayed under "Layers" in the engineering system.
   - Specify the tab sequence for activating the screen objects and selecting the objects to be activated under "Miscellaneous."
   - Under "Hide/Show" specify whether a size to be defined is used to determine the display for layers and/or objects.
   - Dynamize the display and operating objects under "Animations."
   - Select "Events" to define which functions you want to execute in runtime when you call and exit the screen or at other events.

Result

You created the screen in your project. You can now paste objects and control elements from the "Toolbox" task card in further work steps.

See also

Steps (Page 45)

1.1.5.5 Managing screens (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You can move screens within a project to other groups, or copy, rename, and delete them.

Moving screens in a group

1. Select the "Screens" folder in the project tree.
2. Select the "Add group" command from the shortcut menu.
   A folder called "Group_x" is inserted.
3. Select the screen in the project tree.
4. Drag-and-drop the screen to the required group.
   The screen is moved into this group.

Copy screen

1. Select the screen in the project tree.
2. Select the "Copy" command in the shortcut menu to copy the screen to the clipboard.
3. In the project tree, select the screen insert position.
4. Select "Paste" from the shortcut menu to insert the screen.
   A copy of the screen is inserted. A consecutive number is appended to the name of the original in the copy.

Alternatively, press <Ctrl> while you drag the screen to the required position.

Note

If you copy a screen with interconnected template for several devices and projects, the template will also be copied. Any existing matching template is not used. This holds particularly true when you copy the screens with drag-and-drop.

Rename screen

1. Select the screen in the project tree.
2. Select "Rename" from the shortcut menu.
3. Type in a new name.
4. Press <Enter>.

As an option, use the <F2> function key to rename the screen.

Delete screen

1. Select the screen in the project tree.
2. Select "Delete" from the shortcut menu.
   The screen and all its objects are deleted from the current project.

See also

Steps (Page 45)

1.1.5.6 Defining the start screen of the project (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

The start screen is the initial screen opened at the start of project in Runtime. You can define a different start screen for each one of the HMI devices. Beginning at this start screen, the operator calls the other screens.

Requirement

The project contains the screen you want to use as the start screen.

Procedure

1. Double-click "Runtime settings > General" in the project tree.

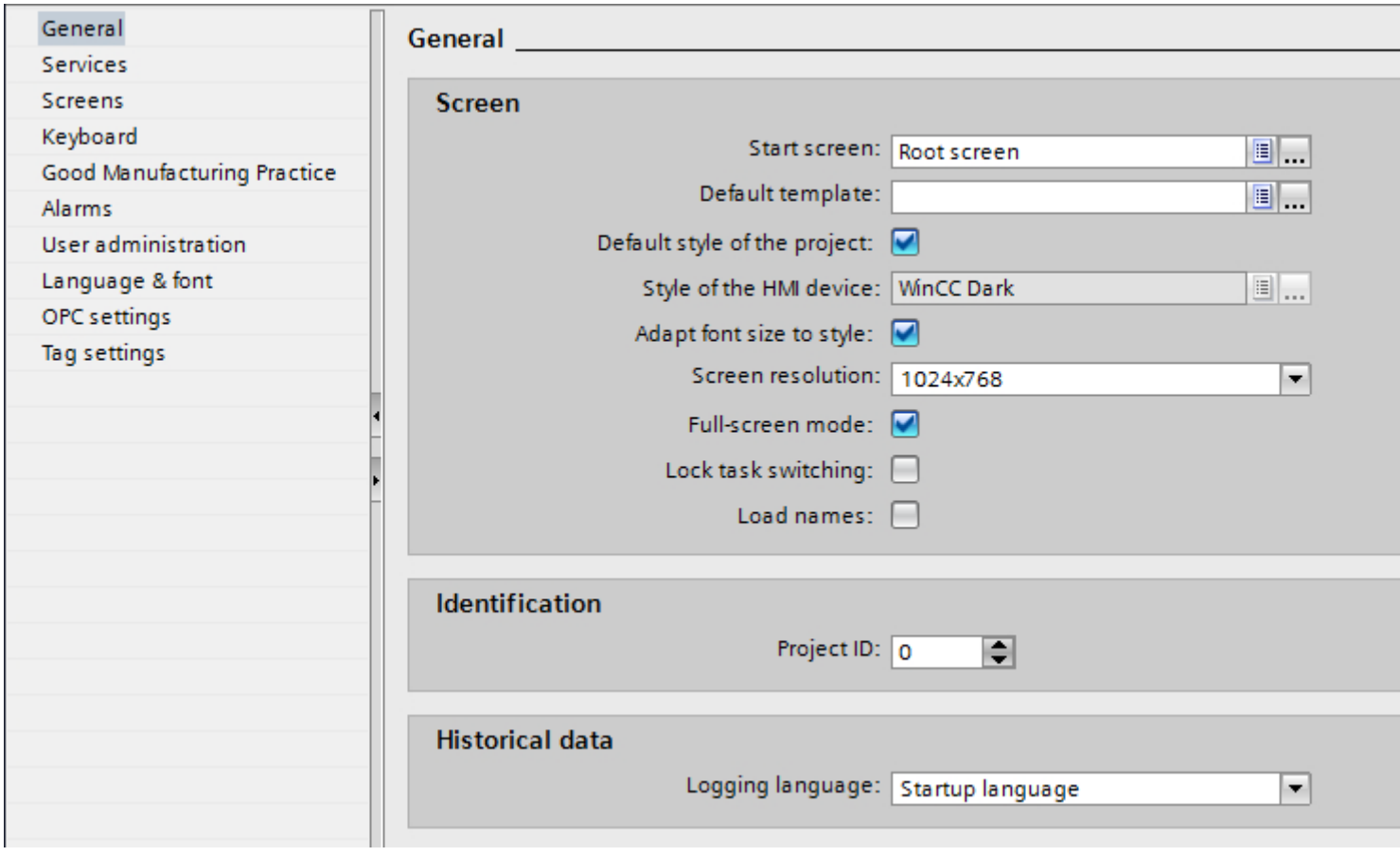

2. Select the desired screen as "Start screen".

Alternatively select a screen in the project tree and select "Define as start screen" in the shortcut menu.

Result

The start screen opens on the HMI at the start of Runtime.

See also

Steps (Page 45)

1.1.5.7 Defining the start screen of the project (RT Professional)

Introduction

The start screen is the initial screen opened at the start of project in Runtime. You can define a different start screen for each one of the HMI devices. Beginning at this start screen, the operator calls the other screens.

Requirement

The project contains the screen you want to use as the start screen.

Procedure

1. Double-click "Runtime settings > General" in the project tree.

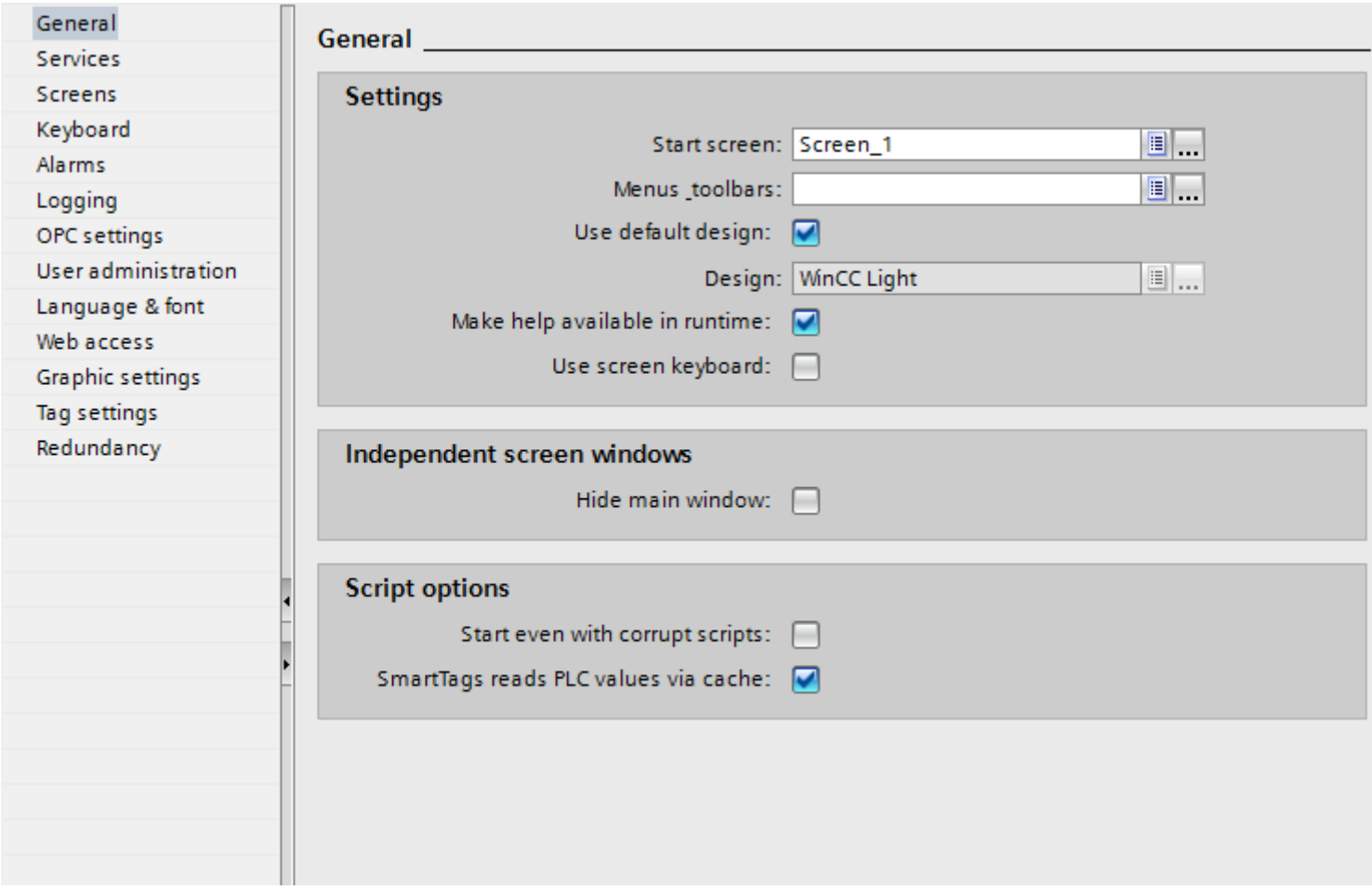

2. Select the desired screen as "Start screen".

   Alternatively select a screen in the project tree and select "Define as start screen" in the shortcut menu.

Result

The start screen opens on the HMI at the start of Runtime.

1.1.5.8 Changing the screen resolution (RT Advanced)

You change the screen resolution in the Runtime settings of the HMI device.

The screen resolution can be configured for the following HMI devices:

- RT Advanced

Configuring a fixed screen resolution

1. Open the Runtime settings of the HMI device.
2. Select the screen resolution under "General > Screen > Screen resolution".

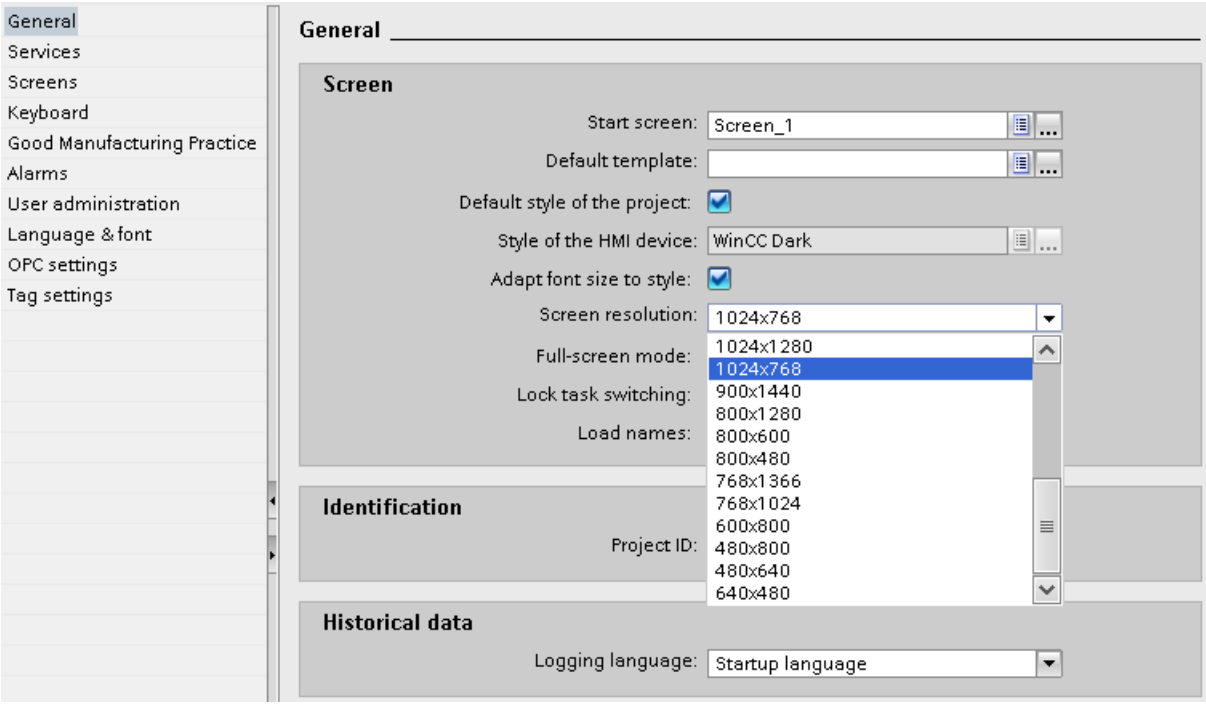

1.1.6 Working with templates (RT Advanced, Basic Panels, Panels, Comfort Panels)

1.1.6.1 Basics on working with templates (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Configure the objects in a template which are to be displayed in all screens based on this template.

Note the following rules:

- A screen must not be based on a template.
- A screen is only based on one template.
- You can create several templates for one device.
- A template cannot be based on another template.

Objects for a template

You determine functions and objects in the template which are to apply for all screens based on this template:

- Assignment of function keys: You also assign the function keys in the template for HMI devices with function keys. This assignment overwrites a possible global assignment.
- Permanent areas: Some devices support a permanent area for all screens in the top area of the screen. In contrast to the template, the permanent area occupies an area of the screen for itself alone.
- Operator controls: You can paste all screen objects which you also use for a screen into a template.

Application examples

- You want to assign the ActivateScreen" function to a function key in the template. The operator uses this key to switch to another screen in runtime. This configuration applies to all screens that are based on this template.
- A graphic with your company logo can be added to the template. The logo appears on all screens that are based on this template.

Note

If an object from the template has the same position as an object in the screen, the template object is covered.

See also

Screen basics (Page 37)

Steps (Page 45)

Task cards (Page 42)

Creating a new template (Page 56)

Managing a template (Page 58)

Using a template in the screen (Page 59)

Global screen (Page 54)

Configuring permanent areas in the template (Page 57)

1.1.6.2 Global screen (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You define global elements for all screens of an HMI device independently of the used template.

Function keys

For HMI devices with function keys you assign the function keys globally in the "Global Screen" editor. This global assignment applies for all screens of the HMI device.

Proceed as follows to assign function keys locally in screens or templates:

1. Click the function key in your screens or templates.
2. Deactivate "Properties > Properties > General > Use Global Assignment" in the inspection window.

Indicator and control objects for alarms

The "Alarm window" and "Alarm indicator" objects that are available as global objects are configured within the "Global screen" editor.

The "Alarm window" and "Alarm indicator" are always shown in the foreground.

For Comfort Panels you also have the possibility of configuring a "System diagnostic view" in the global screen.

Note

If you have configured a permanent area in a template, do not position the alarm window and alarm indicator in the area of the permanent area. Otherwise, the alarm window and the alarm indicator will not be displayed in Runtime.

However, the permanent area is not visible in the "Global screen" editor.

Order of configuration of screens

The following order applies for the configuration:

- The global screen comes before screens and templates
- Screens come before templates

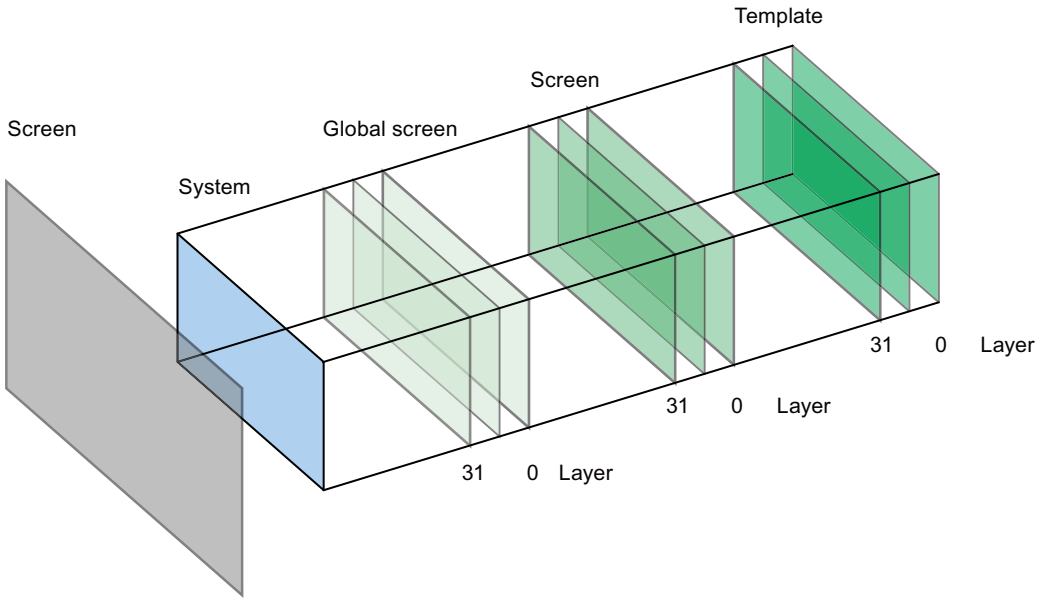

The system layer is not configurable. This contains

- input dialog boxes
- alarms from the operating system
- the direct keys for touch panels

See also

Basics on working with templates (Page 53)

1.1.6.3 Creating a new template (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

In a template, you can centrally modify objects and function keys. Changes to an object or of a function key assignment in the template are applied to the object in all the screens which are based on this template.

Note

HMI device dependency

Function keys are not available on all HMI devices.

Requirement

- The project has been created.
- The Inspector window is open.

Procedure

1. Select "Screen management > Templates" in the project tree and then double-click "Add new template".
   The template is created in the project, and appears in your view.
   The properties of the template are displayed in the Inspector window.
2. Define the name of the template under "Properties > Properties > General" in the Inspector window.
3. Specify the layers in the engineering system that are displayed under "Properties >Properties >Layers" in the Inspector window.
4. Add the necessary objects from the "Tools" task card.
5. Configure the function keys.

Result

The template is created in your project.

See also

Basics on working with templates (Page 53)

1.1.6.4 Configuring permanent areas in the template (RT Advanced, Panels, Comfort Panels)

Introduction

In the permanent area, you configure objects which are visible in all screens. In contrast to the template, the permanent area occupies an area of the screen for itself alone.

You can configure permanent areas in a template or in a screen. In the project tree, the permanent area is available under "Screen management > Permanent area".

Configuring permanent areas

To configure a permanent area, follow these steps:

1. Open a template for a HMI device with a permanent area.
2. Using the mouse (cursor form:), drag the top edge of the editable screen area downwards.

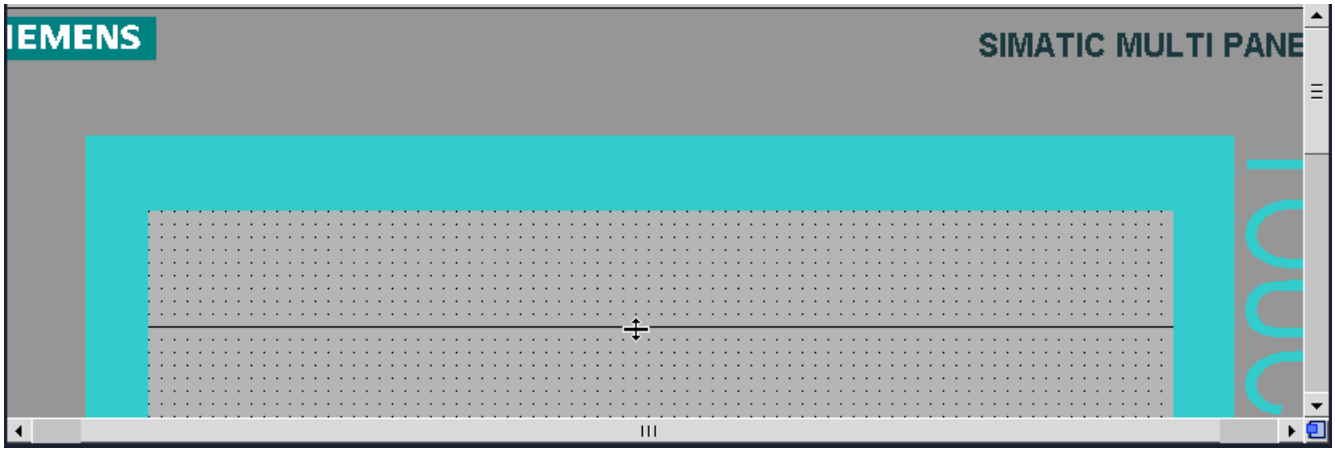

3. Configure the desired objects within the permanent area.

The default height of the permanent area is "0". The maximum height is the maximum height of the screen minus 10 pixels.

Result

The content of the permanent area is shown on every screen and in the template. The area above this line is now used as a permanent area in all the screens of the HMI device. Objects already configured in the screens are moved down by the height of the permanent area.

See also

Basics on working with templates (Page 53)

1.1.6.5 Managing a template (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You can move, copy, rename, and delete templates within a project in the Project window.

Moving a template into a group

1. Select the templates in the project navigation "Screen management > Templates".
2. Select "Add group" in the shortcut menu.
   A folder called "Group_x" is inserted.

3. Select the template in the project navigation.
4. Drag-and-drop the template to the required group.
   The template is moved to this group.

Copying templates

1. Select the template in the project navigation.
2. Select "Copy" in the shortcut menu.
3. Select the position in the project navigation where you want to paste the template.
4. Select "Paste" from the shortcut menu to insert the template.
   A unique name is assigned automatically to the copy.

Alternatively, you can hold down the <Ctrl> key, and drag the template into position.

Deleting a template

1. In the project navigation, select the template to be deleted.
2. Select "Delete" in the shortcut menu.
   The template, and all its objects are deleted from the current project.

Assigning a template to a screen

1. In the project navigation, select the screen to which you want to assign the template.
2. In the Inspector window, select "Properties > Properties > General".
3. Select the desired template under "Template."
   The selected template and all the objects contained in it are assigned to the screen.

See also

Basics on working with templates (Page 53)

1.1.6.6 Using a template in the screen (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You use a template in the screen. All the objects configured in the template are also available in the screen.

Requirement

A template has been created.

A screen has been created.

Procedure

Proceed as follows to use a template in a screen:

1. Double click a screen in the project tree. The screen opens in the work area.
2. Open "Properties > Properties > General" in the inspector window.
3. Select a template that is to be applied to the screen under "Template".

Show template in screen

When you edit a screen, you can show an existing template in the screen.

Proceed as follows to show a template in the screen:

1. Activate "Extras > Settings > Visualization > Show template in screens" in the menu.

Result

The screen is based on the selected template. All objects which you have configured in the template are available in the screen. The template is displayed in the screen.

See also

Basics on working with templates (Page 53)

1.1.7 Working with pop-up screens (RT Advanced, Basic Panels, Panels, Comfort Panels)

1.1.7.1 Basics for working with pop-up screens (RT Advanced, Panels, Comfort Panels)

Introduction

You use the pop-up screen to configure a screen for additional content, for example, object settings. The pop-up screen is displayed on top of the current screen as soon as you call a system function.

Only one pop-up screen at a time is displayed in a screen.

Note

Device dependency of the "Pop-up screen" object

The "Pop-up screen" object is only available for KTP Mobile Panels, Comfort Panels and RT Advanced.

Use

You configure a button in the screen and assign the "ShowPopupScreen" or "ShowPopupScreenSizable" function in order to display a pop-up screen with a defined size.

When operating the button, a pop-up screen is opened or closed. In the pop-up screen, you make the settings.

Size

You have the option to configure pop-up screens that are larger than the screen size:

- Horizontal, up to maximum double the width
- Vertical, up to maximum six times the height

Objects in the pop-up screen

You configure the following objects in the pop-up screen:

- Basic objects, such as lines and polygon-based objects
- Elements, such as buttons
- Controls, such as trend view or alarm view
- Faceplates

Note

You cannot configure alarm windows, system diagnostics windows and alarm indicators in the pop-up screen.

Softkeys and hotkeys are not supported in the "Pop-up screen" object.

Note

Objects in the pop-up screen as well as the pop-up screen itself do not support access with VB scripting.

1.1.7.2 Creating a new pop-up screen (RT Advanced, Panels, Comfort Panels)

Introduction

A new pop-up screen is created in Project tree > Screen management. The size of a pop-up screen is defined in the properties.

Requirements

- The project has been created.
- The Inspector window is open.

Creating a new pop-up screen

1. Double-click "Screen management > Pop-up screens > Add new pop-up screen" in the project tree.
   The pop-up screen is generated in the project and is displayed in the work area.
   The pop-up screen properties are displayed in the Inspector window.
2. Define the name of the pop-up screen under "Properties > Properties > General" in the Inspector window.
3. Specify the layers in the engineering system that are displayed under "Properties >Properties >Layers" in the Inspector window.
4. Define the height and width of the pop-up screen under "Properties > Properties > Layout" in the Inspector window.
5. Define the color for the scroll arrows and the sizing handle in the scroll bar in "Properties > Properties > Scroll bar" in the Inspector window.
6. Add the necessary objects from the "Tools" task card.
7. Configure the function keys.

   Note

   You cannot address the "Pop-up screen" object and the objects configured in it with VB scripts.

   Note

   Objects and controls can be dynamized in the pop-up screen. The pop-up screen itself does not support dynamization.

Result

You have created a new pop-up screen in which you have configured objects and functions.

1.1.7.3 Managing pop-up screens (RT Advanced, Panels, Comfort Panels)

Introduction

You can move, copy, rename, and delete pop-up screens within a project in the Project window.

Moving pop-up screens in a group

1. Select "Screen management > Pop-up screens" in the project tree.
2. Select "Add group" in the shortcut menu.
   A folder called "Group_x" is inserted.
3. Select the pop-up screen in the project tree.
4. Drag-and-drop the pop-up screen to the required group.
   The pop-up screen is moved into this group.

Copying pop-up screen

1. Select the pop-up screen in the project tree.
2. Select "Copy" from the shortcut menu.
3. In the project tree, select the pop-up screen insert position.
4. Select "Paste" in the shortcut menu.
   A unique name is assigned automatically to the copy.

Deleting pop-up screen

1. Select the pop-up screen you want to delete in the project tree.
2. Select "Delete" in the shortcut menu.
   The pop-up screen and all its objects are deleted from the current project.

1.1.7.4 Using the pop-up screen (RT Advanced, Panels, Comfort Panels)

Introduction

You configure the system function to open a pop-up screen at the event of an operating object. The linked system function is executed when the event occurs in runtime.

Note

You configure a button to open a pop-up screen in a slide-in screen.

If the configured position of the pop-up screen is outside the slide-in screen, the slide-in screen is closed when the pop-up screen closes.

Requirements

- A pop-up screen has been configured.
- A screen is open.
- The Inspector window is open.

Procedure

1. Configure a button.
2. Click "Properties > Events" in the Inspector window.
3. Select an event, for example, "Click".
4. Click on "Add function" in the table.
5. Select the "ShowPopupScreen" system function.

Result

The pop-up screen opens when the operator clicks the button in runtime.

1.1.8 Working with slide-in screens (RT Advanced, Basic Panels, Panels, Comfort Panels)

1.1.8.1 Basics on working with slide-in screens (RT Advanced, Panels, Comfort Panels)

Introduction

The "Slide-in screen" object provides quick navigation between the start screen and the slide-in screens containing additional configured contents stored outside the start screen. Navigation is supported by the multi-touch functions. The configurable handles appearing at the edges of the start screen provide quick access to the stored slide-in screens.

The size of the slide-in screens depends on the HMI device used.

Note

Device dependency of the "Slide-in screen" object

The "Slide-in screen" object is only available for KTP Mobile Panels, Comfort Panels and RT Advanced.

Application

The "Slide-in screen" is used to configure up to four slide-in screens for additional contents and operator controls, for example, a virtual keyboard or a system dialog.

By default, the following slide-in screens are set up for each device:

- "Slide-in screen top": appears at the top of the start screen in Runtime
- "Slide-in screen bottom": appears at the bottom of the start screen in Runtime
- "Slide-in screen left": appears at the left of the start screen in Runtime
- "Slide-in screen right": appears at the right of the start screen in Runtime

It is possible to configure objects in each of these slide-in screens.

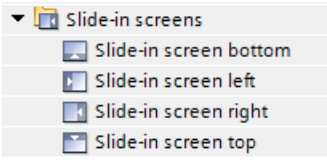

Note

It is not possible to delete the default slide-in screens or to set up your own slide-in screens.

Objects in the slide-in screen

You configure the following objects in the slide-in screen:

- Basic objects, such as lines and polygon-based objects
- Elements, such as buttons
- Controls, such as trend view or alarm view
- Faceplates

Note

Objects in the slide-in screen as well as the slide-in screen itself do not support access with VB scripting.

Note

Release buttons and locked operator controls cannot be configured in the slide-in screen.

Softkeys and hotkeys are not supported in the slide-in screen.

See also

Configuring slide-in screens (Page 65)

Configuring slide-in screens for devices without multi-touch function (Page 67)

Using slide-in screens (Page 69)

1.1.8.2 Configuring slide-in screens (RT Advanced, Panels, Comfort Panels)

Introduction

By default, four slide-in screens are preset for each device in the project tree under "Screen management > Slide-in screens". For each of these slide-in screens, you configure the size, the layout and the handle for operation in Runtime using the properties.

Requirement

- The project has been created.
- An HMI device has been created.
- The Inspector window is open.

Configuring properties for slide-in screens

1. Select the slide-in screen you wish to configure in the project tree under "Screen management > Slide-in screens.
   The slide-in screen properties are displayed in the Inspector window.

   Note

   The names of all slide-in screens are write-protected and cannot be changed.

2. Define the background color of the slide-in screen in the "Background color" selection list of the "Properties > Properties > General" Inspector window.
3. Define the grid color of the slide-in screen in the "Grid color" selection list.
4. Enable the "Activate" check box.
5. Specify the layers in the engineering system that are displayed under "Properties >Properties >Layers" in the Inspector window.
6. Define the height and width of the slide-in screen in "Properties > Properties > Layout" in the Inspector window:
   - Define the height only for "Slide-in screen top" and "Slide-in screen bottom".
   - Define the width only for "Slide-in screen left" and "Slide-in screen right".

Note

If you do not need all four slide-in screens, enable the slide-in screens relevant to you only.

If you do not need any slide-in screens, leave the "Activate" check box empty for all four slide-in screens.

Note

Slide-in screens cannot be deleted nor copied to a different HMI device.

However, it is possible to copy the objects configured in a slide-in screen to another slide-in screen.

Configuring handles for slide-in screens

1. Define the color of the first line of the handle in the "Lines" area of the "Color" selection list under "Properties > Properties > Handle" in the Inspector window.
2. Define the color of the second line of the handle in the "Alternative color" selection list.

3. Define the color for the operable area of the handle in the "Color" selection list under "Operable area".
4. Select one of the following options from the "Visibility" area:
   – "Hide handle automatically" (activated by default)
   – "Always show handle"
   – "Never show handle"

Note

When selecting the "Hide handle automatically" option, the handle only becomes visible once the user clicks the operable area.

You select the "Never display handle" option when the slide-in screen is configured using the "ShowSlideInScreen" system function.

Note

You are notified by a warning if a handle which is configured as visible covers a configured object in a screen, such as a button, completely or partially.

Note

The following dimensions are defined for the handle:

- The width of a handle for the "Slide-in screen top" and "Slide-in screen bottom" objects is one-third of the screen length.
- The height of a handle for the "Slide-in screen left" and "Slide-in screen right" objects is one-third of the screen width.

Result

You have configured the properties of the slide-in screens and the handle for operation in Runtime.

See also

Basics on working with slide-in screens (Page 64)

1.1.8.3 Configuring slide-in screens for devices without multi-touch function (RT Advanced, Panels, Comfort Panels)

Introduction

You use the "ShowSlideInScreen" system function to make the slide-in screens available on devices without multi-touch functions, for example Key Panels. You link this system function to buttons. Buttons are used to navigate between the start screen and the slide-in screens.

Requirement

- The project has been created.
- An HMI device has been created.
- A screen is open.
- The Inspector window is open.

Procedure

1. Drag the "Button" object to the screen from the toolbox.

   Note

   Navigation in Runtime is much easier if the buttons for operating the slide-in screens are created directly in the start screen.

2. Enter a suitable text of any length for the button.
3. In the Inspector window, select "Properties > Events > Click".
   The "Function list" dialog box is opened.
4. Select the "ShowSlideInScreen" system function from the "Keyboard operation for screen objects" group.
5. Assign the ID of the slide-in screen appearing in runtime when the button is operated to the "ShowSlideInScreen" function:
   - "Slide-in screen left"
   - "Slide-in screen top"
   - "Slide-in screen right"
   - "Slide-in screen bottom"
6. Assign the appropriate mode to the "ShowSlideInScreen" function:
   - "Toggle"
   - "Off"
   - "On"

Result

You have configured the slide-in screens for operation on devices without multi-touch function. When clicking on the respective button in the screen in Runtime, the slide-in screen behaves in accordance with the configuration.

See also

Basics on working with slide-in screens (Page 64)

1.1.8.4 Using slide-in screens (RT Advanced, Panels, Comfort Panels)

Introduction

In slide-in screens, you configure additional contents and operator controls, for example, navigation buttons, an alarm view or a trend view.

Requirements

- The project has been created.
- An HMI device has been created.
- A slide-in screen is open.
- The Inspector window is open.

Procedure

1. Add the necessary objects from the "Tools" task card to the slide-in screen.
2. Configure the function keys.

Note

You cannot address the "Slide-in screen" object and the objects configured in it with VB scripts.

Note

Objects and controls can be dynamized in the slide-in screens. The slide-in screens as such do not support dynamization.

Result

You have configured objects and functions in the slide-in screen.

See also

Basics on working with slide-in screens (Page 64)

1.1.9 Working with designs

1.1.9.1 Basics on working with designs (RT Professional)

What are designs?

Designs are templates for displaying screens and the objects configured there. A design contains, for example, color specifications for foreground, border line, text, etc..

The design is applied to the following objects:

- Screens
- Menus and toolbars
- Screen objects

Application

To design the user interfaces of several HMI devices in a uniform layout, assign the same design to each HMI device.

You can modify the layout of screens and objects by customizing the current design to suit your requirements. The new design is activated automatically for all screens and objects of the HMI device.

Managing designs

You manage designs in the "Designs" editor.

WinCC features a number of predefined designs such as "WinCC Classic" for user interfaces in the classic WinCC layout and "WinCC Glass" for user interfaces in the style of Windows Vista. Predefined designs cannot be deleted or changed.

See also

Use default design (Page 77)

1.1.9.2 Designs editor (RT Professional)

Introduction

Designs are project-specific. The "Designs" editor is available in the project tree under "Common data".

Structure of the "Designs" editor

The "Designs" area lists all designs. The "Preview" area shows how images and objects are displayed with the selected design.

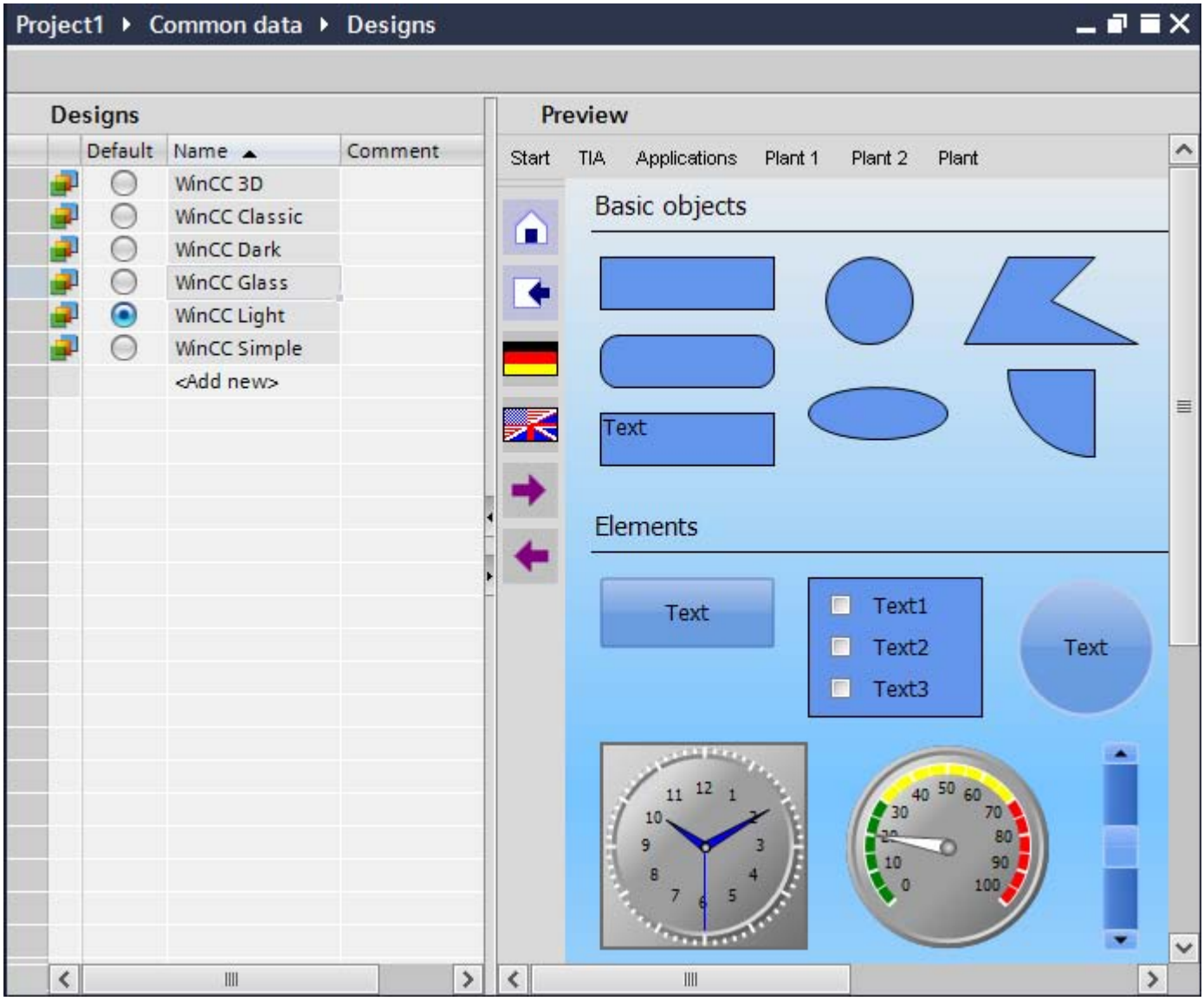

Predefined designs have a gray background and cannot be deleted or modified. The default design is selected in the "Default" column.

Click the desired design in the "Designs" list for a preview of another design. The "Preview" in the right pane is refreshed:

- The first line offers a preview of the menu layout.
- The icons on the left offer a preview of the layout of the toolbar.
- The objects of the "Tools" task card are grouped by basic objects, elements and controls.

You can find detailed information on which layout properties are specified by the design in the Inspector window.

See also

Basics on working with designs (Page 70)
Use default design (Page 77)
Creating a design (Page 72)
Changing a design (Page 72)
Deleting a design (Page 73)

1.1.9.3 Creating a design (RT Professional)

Requirement

The "Designs" editor is open.

Procedure

Proceed as follows:

1. Click "Add" in the list of designs.
   A new design is created. The WinCC default properties are set for screens, menus, toolbars and objects.
2. Overwrite the suggested name with a meaningful name.
3. Enter a description of the design, e.g. the intended use in the "Comment" column.
4. Configure the design as required.

Result

You have created and configured a new design.

See also

Designs editor (Page 70)
Changing a design (Page 72)
Deleting a design (Page 73)
Basics on working with designs (Page 70)

1.1.9.4 Changing a design (RT Professional)

Requirement

The "Designs" editor is open.

Procedure

Proceed as follows:

1. Select the design you wish to change from the list of designs.

   Note

   You cannot manipulate the predefined designs of WinCC.

2. Make the required changes in the Inspector window.
3. Check the result in the preview.

The changes are effective immediately.

See also

Designs editor (Page 70)
Creating a design (Page 72)
Deleting a design (Page 73)
Basics on working with designs (Page 70)

1.1.9.5 Deleting a design (RT Professional)

Requirement

The "Designs" editor is open.

Procedure

Proceed as follows:

- Select "Delete" from the shortcut menu of the desired design.

  Note

  You cannot manipulate the predefined designs of WinCC.

Result

The design is deleted. WinCC assigns the predefined "WinCC Classic" design as the new default design to HMI devices to which the deleted design was assigned as default in the Runtime settings.

See also

Designs editor (Page 70)
Creating a design (Page 72)

Changing a design (Page 72)

Basics on working with designs (Page 70)

1.1.9.6 Applying a design (RT Professional)

Core message

In WinCC, every HMI device is assigned a design as a basic design. The following basic design specifications are applied to all screens of the HMI device:

- Settings relating to the display of the shadow of an object (if "Shadow" is activated for the object)
- Settings relating to the display of elements when the mouse pointer is moved over them (hover effect)
- Settings relating to the layout of menus and toolbars

For the individual screens, elements and controls, you also specify in each case whether the specifications of the basic design or the individual settings are applied in the Inspector window.

See also

Basics on working with designs (Page 70)

Designs editor (Page 70)

Use default design (Page 77)

Specifying the basic design user interface of an HMI device (Page 74)

Applying design templates to screens (Page 75)

Using design templates for basic objects and elements (Page 75)

Applying design templates to controls (Page 76)

1.1.9.7 Specifying the basic design user interface of an HMI device (RT Professional)

Procedure

To change the basic design of the user interface of an HMI device, follow the steps below:

1. Open the "Runtime settings" of the HMI device.
2. Under "General > Settings", specify whether the default design or a different design is used as the basic design.

See also

Basics on working with designs (Page 70)

Applying a design (Page 74)

Applying design templates to screens (Page 75)
Using design templates for basic objects and elements (Page 75)
Applying design templates to controls (Page 76)

1.1.9.8 Applying design templates to screens (RT Professional)

Procedure

Proceed as follows to use design templates for the representation of images on a screen:

1. Open the desired screen in the "Screens" editor.
2. Activate "Properties > Properties > Design > Settings > Use color scheme" in the Inspector window.

Result

The settings of the design are applied to the screen.

All the settings that are not predefined by the design are activated and can be changed.

See also

Basics on working with designs (Page 70)
Specifying the basic design user interface of an HMI device (Page 74)
Applying a design (Page 74)
Using design templates for basic objects and elements (Page 75)
Applying design templates to controls (Page 76)

1.1.9.9 Using design templates for basic objects and elements (RT Professional)

Procedure

To apply the design specifications for the layout of basic elements and elements, proceed as follows:

1. Open the screen in the "Screens" editor.
2. Click the required object in the work area.
3. Select the group settings "Properties > Properties > Designs" in the Inspector window.
4. Activate "Shadow" to visualize the object with a shadow.
5. Activate "Use color scheme" to to apply the default colors of the design.

Result

The design settings are applied to the object. In the Inspector window, the fields for setting the background color, frame etc. are deactivated. All the settings that are not predefined by the design are activated and can be changed.

If the "Shadow" option is deactivated in the design, the object is displayed without a shadow in the screen - even if a "shadow" is activated in the object properties.

See also

Basics on working with designs (Page 70)

Applying a design (Page 74)

Specifying the basic design user interface of an HMI device (Page 74)

Applying design templates to screens (Page 75)

Applying design templates to controls (Page 76)

1.1.9.10 Applying design templates to controls (RT Professional)

Procedure

Proceed as follows to apply design templates for visualization of controls:

1. Open the screen in the "Screens" editor.
2. Click the required object in the work area.
3. Deactivate "Properties > Properties > Appearance > Style > Use project settings for design".
4. Select the design from the selection list.

The control is displayed in the style specified in the design.

See also

Basics on working with designs (Page 70)

Applying a design (Page 74)

Specifying the basic design user interface of an HMI device (Page 74)

Applying design templates to screens (Page 75)

Using design templates for basic objects and elements (Page 75)

1.1.9.11 Use default design

Introduction

A design is specified as a default design. If you want to design the user interfaces of several HMI devices in a uniform layout, specify the desired design as the default design, and assign it to the HMI devices as the basic design in the Runtime settings "Default design".

If you want to change the layout of the screens and objects, adapt the default design to your requirements or configure a different design as the default design. The latter is the actual advantage of working with the default design: You can practically change the layout for the screens and objects of all the HMI devices at the click of a mouse.

Configuring a default design

Proceed as follows:

1. Open the "Designs" editor.
2. Select the desired design in the list of designs in the "Default" column in the work area.
   Or:
   Display a preview of the desired desire in the work area and select "Properties > Properties > General > Design > Default" in the Inspector window.
3. If required, adapt the design to your requirements.

Use default design

The default design settings are effective for the objects which satisfy the following conditions:

- "Runtime settings > General > Settings > Use default design" is activated for the HMI device.
- For screens, basic elements and elements: "Properties > Properties> Design > Settings > Use color scheme" is selected in the Inspector window.
- For controls: "Properties > Properties > Appearance > Style > Accept project settings" is selected in the Inspector window.

See also

Basics on working with designs (Page 70)

Designs editor (Page 70)

Applying a design (Page 74)

1.1.10 Working with styles and style sheets

1.1.10.1 Basics on working with styles (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

The style editor is a global editor. You define a uniform appearance for display and operating elements in the style editor. In this way, you harmonize the display of objects in Runtime. The style editor offers predefined styles. Additional predefined styles are available for download at Siemens Industry Online Support (http://support.automation.siemens.com/WW/view/en/91174767).

You can choose one of the predefined styles or create your own style. You use the style editor independently of devices and projects.

Views in the style editor

Objects are grouped in the style editor according to their main visual features.

The following groups are available:

- Buttons, lines, and polygon-based objects
- Text-based objects
- Trend-based objects, such as trend view,
- graphic or value-based objects, such as value table.

Style items

You use a style item to define the appearance of a screen object within a style. You can design several objects of the same type differently in one style. You define, for example, different style items for simple buttons and navigation buttons.

You create and manage style items in the style editor.

Style sheets

You use style sheets to configure the same properties for a group of objects. In a style sheet you can, for example, define the width and background color of table headers for all table-based objects. Style sheets are divided into predefined types depending on the included properties, for example, buttons, limits, table headers.

You use style sheets to define the design of shared object properties. Some special object properties are not included in style sheets. You use styles to design all objects of a project with visual consistency.

Managing styles and style sheets

You create and edit styles and style sheets in the project library. To get a better overview of types in expansive libraries, you save styles and style sheets in folders with meaningful names. You should also assign meaningful names to the styles and style sheets you create yourself.

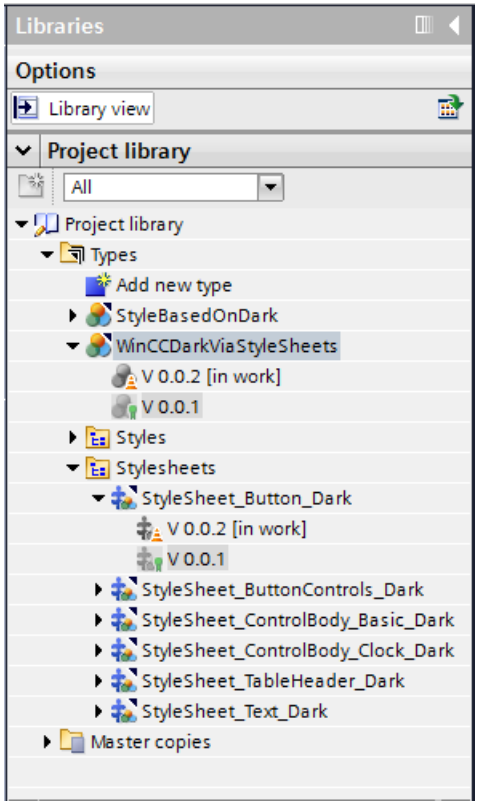

Support of styles in faceplates

Styles and style items can be assigned to faceplate instances.

However, there is no direct connection between the faceplate types and styles, so selection and preview of a style item in the faceplate editor is not possible. To assign a style item to a faceplate, enter the name of the style item under "Style item appearance".

The configured style item is used in the instance of a faceplate, provided the name of the style item exists in the style used. Ensure that the name is written correctly.

See also

Displaying predefined styles (Page 80)

Defining a style (Page 80)

Managing styles (Page 82)

Using style items (Page 85)

1.1.10.2 Displaying predefined styles (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

In the "Style" editor, you see predefined styles and create your own styles for representing display and operating objects in Runtime.

Open "Styles" editor

1. Double-click "Common data" in the project navigation.
2. Click the "Styles" editor. The editor opens.

Note

Style editor in the project navigation

The style editor is only visible in the project navigation if a device that supports styles, e.g., Comfort Panels, is created in the project.

3. The standard styles are displayed in the work area of the "Style" editor:

- WinCC Light
- WinCC Dark
- WinCC Fresh
- WinCC Wireframe.

Result

The settings for the groups and individual objects are displayed in the work area.

You will assign predefined styles or define your own styles in the next steps.

Predefined styles are write-protected and cannot be changed.

See also

Basics on working with styles (Page 78)

1.1.10.3 Defining a style (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

In the "Styles" editor, you see predefined styles. Predefined styles are write-protected and cannot be changed.

In addition, you have the option of defining your own styles and applying them to individual objects, groups, and even projects.

Defining a style

1. Open the "Common data" folder in the project navigation.
2. Open the "Style" editor.
3. Click "Add" in the "Styles" area.
   A new style is created.
   The library view opens. The version of the type is displayed in the "Types" folder. The version 0.0.1 has the status "in progress".
4. In the Inspector window, select the colors, borders and other settings for individual objects.

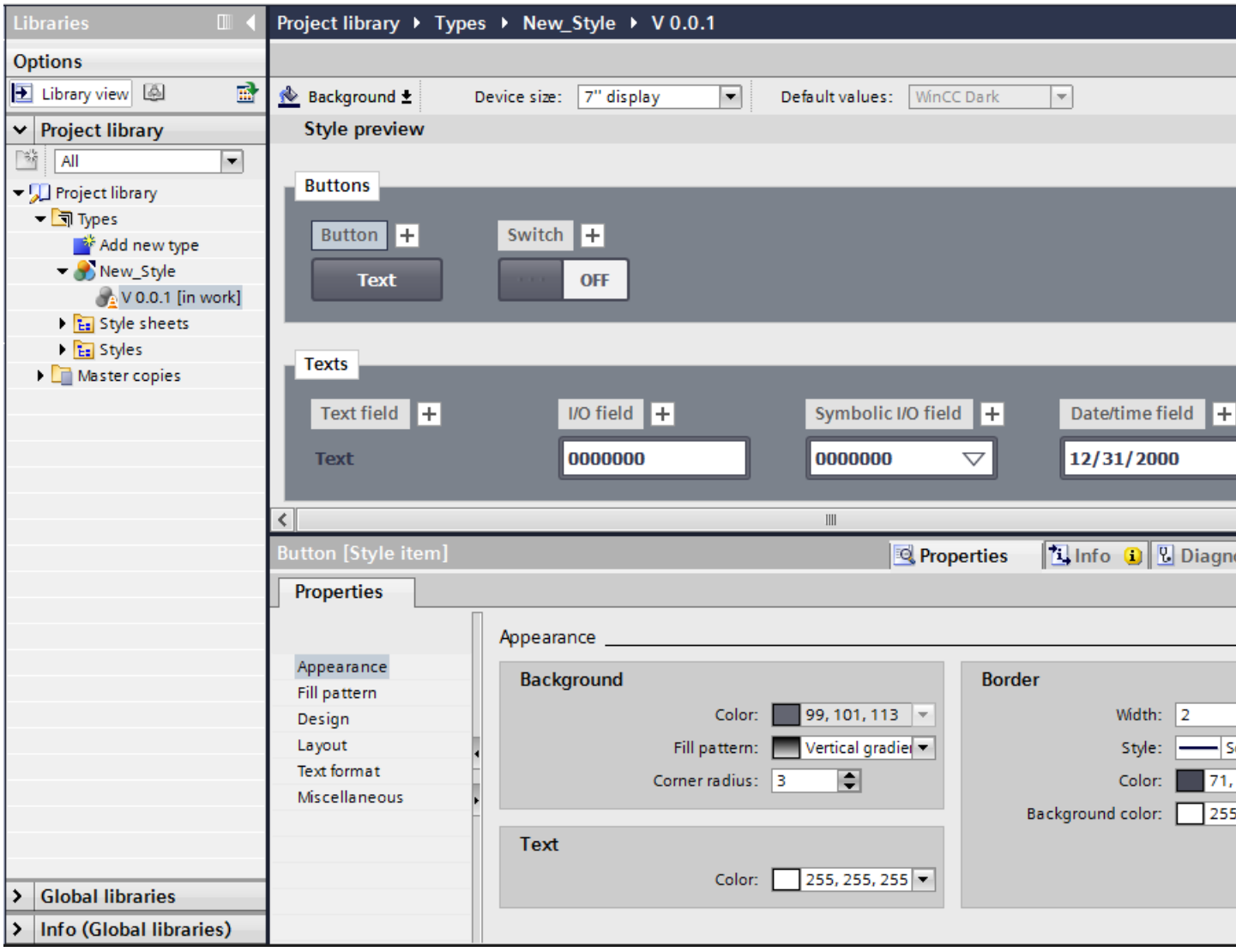

5. In order to activate the new style, select the current version "in progress" and select "Release version" in the shortcut menu.
   The new style appears in the "Styles" editor.

Result

You have added a new style which can be applied to objects or projects.

Defining a style based on a predefined style

1. Open the "Common data" folder in the project navigation.
2. Open the "Style" editor.
3. Select a predefined style in the work area.
4. Select the "Duplicate type" command in the shortcut menu.
   The "Duplicate type" dialog opens.
5. Overwrite the suggested name with a meaningful name.
6. In the "Comment" column, enter a description of the style, e.g. the intended use.
7. Click "OK" to confirm your input.
8. Select the created type in the project library.
9. Select "Edit new type" in the shortcut menu.
   The library view opens. The version of the type is displayed in the "Types" folder. The version 0.0.2 has the status "in progress".
10. Configure the type as required.
11. Select the "Release version" command from the shortcut menu of the edited version.

Result

They have configured a new style based on the predefined style of WinCC.

See also

Basics on working with styles (Page 78)

State of type versions (Page 1958)

1.1.10.4 Managing styles (RT Advanced, Basic Panels, Panels, Comfort Panels)

Deleting a style

You can delete styles that you have defined yourself.

1. Open the "Common data" folder in the project navigation.
2. Open the "Style" editor.

3. Select the style in the work area that you want to delete.

   Note

   You cannot delete the predefined styles of WinCC.

4. Select the "Delete" command in the shortcut menu.

The style is deleted.

Editing a style

You can edit styles that you have defined yourself.

1. Open the project library.
2. Select the released version of the style that you want to change.
3. Select "Edit type" in the shortcut menu.
   A new version "in progress" is created.
4. Make the required changes in the Inspector window.
5. Select the "Release version" command from the shortcut menu of the edited version to put the changes into effect.
   The changes made take effect.

Determining a standard style for a project

A project can be assigned a standard style to achieve a visual conformity of all display objects in the project.

1. Open the "Common data" folder in the project navigation.
2. Open the "Style" editor.
3. Activate a style in the work area.

The activated style is used as standard style in the project. When you add a device to the project, this device uses the standard style.

Changing the style for a specific device

Within a project, you change the style for a specific device.

1. Open the "Runtime settings" editor of the HMI device.
2. Select a device style under "General".

The selected style is applied to the device.

If the device style is not assigned, the device uses the standard style of the project.

See also

Basics on working with styles (Page 78)

1.1.10.5 Adjusting the font size to the device size (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You define the size of the device to whose objects you want to apply a style in the style editor. This means you specify device-dependent font sizes in a style. All font sizes are automatically adapted to the device size by selecting a specific reference size.

The table below shows the device sizes and the reference font sizes in pixels:

Device size	Font size in pixels
4" display	15 px
7" display	17 px
9" display	15 px
12" display	19 px
15" display	19 px
19" display	17 px
22" display	21 px
WinCC RT Advanced	17 px

Activating automatic adjustment of font sizes

1. Double-click the "Runtime settings" editor in the project window.
2. Click "General".
3. Activate the option "Adjust font size in style" under "General > Screen".

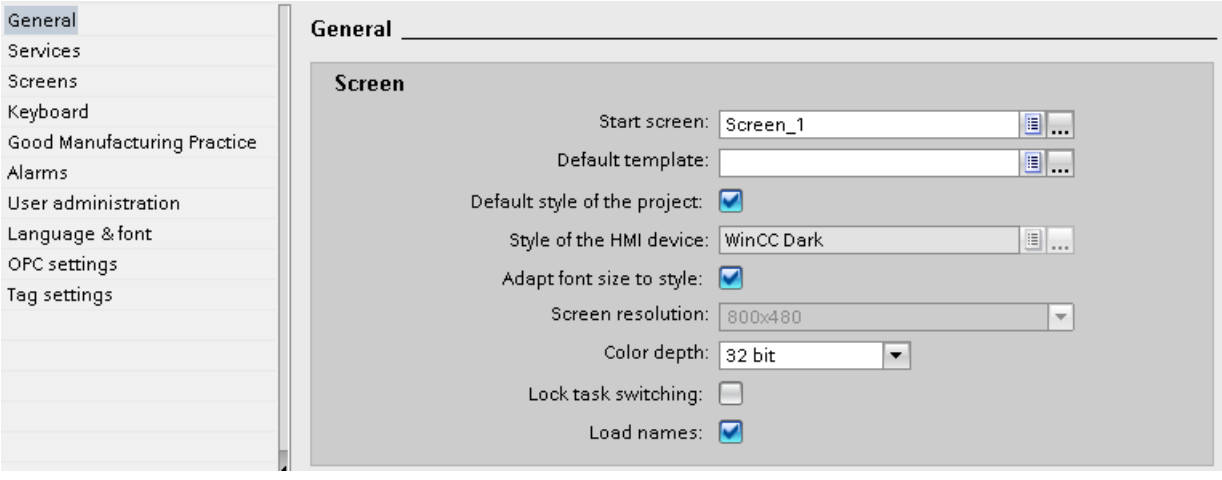

Automatic adjustment of font sizes to the device size was activated in a style.

Adjusting font size in a device style

1. Open the corresponding style for editing in the project library.
2. Select the reference size of the HMI device in the work area.

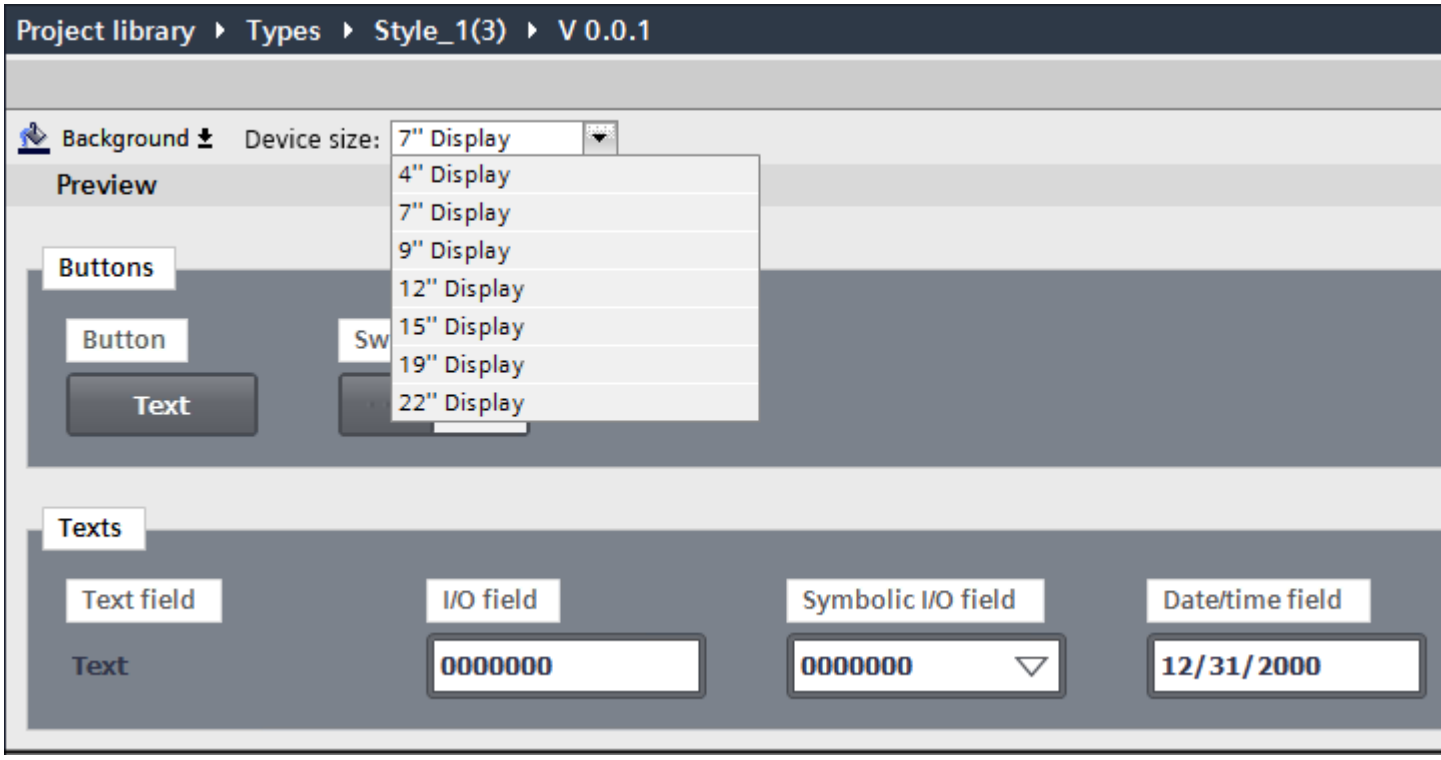

All font sizes in the style are automatically adjusted to the selected device size.

1.1.10.6 Using style items (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You use style items to change the style of a screen object without changing the entire device style. You can, for example, define types of buttons with the help of different style items that are to receive the same appearance in one or several devices.

You define style items in the style editor.

Adding a new style item

1. Open the project library.
2. Select the released version of the style that you want to change.
3. Select "Edit type" in the shortcut menu.
   A new version "in progress" is created.
4. Select the object that you want to use in a different style.

5. Click the plus sign next to the object name.

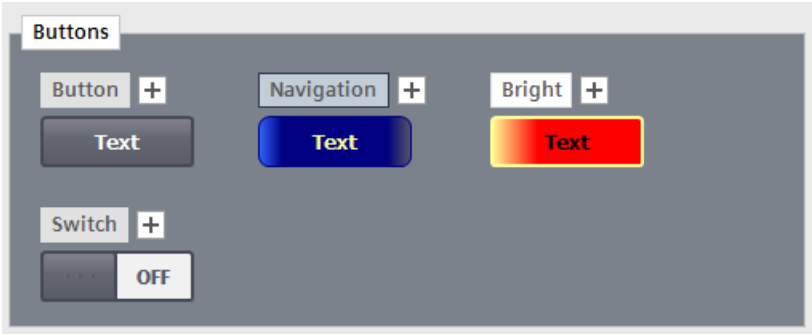

A new style item has been created.
You can also select "Duplicate style item" in the shortcut menu of the object.

6. Make the required changes in the inspector window.
7. Assign a meaningful name to the new style item under "Properties > Miscellaneous > Object > Name".
8. Release the version of the style.

Deleting a style item

You can delete style items that you have defined yourself.

1. Open the project library.
2. Select the style that includes the style item.
3. Open the "in progress" version.
4. Select the style item that you want to delete in the work area.
5. Select the "Delete" command in the shortcut menu.

Note

You cannot rename or delete predefined WinCC style items.

The style item is deleted.

Specifying a style item for an object

If you have configured additional style items for an object in the device style, you can use the predefined style item for a screen object.

1. Open "Properties > Styles/Designs > Settings".
2. Enable the option "Use style/design".
3. Select the predefined style item under "Style item appearance".

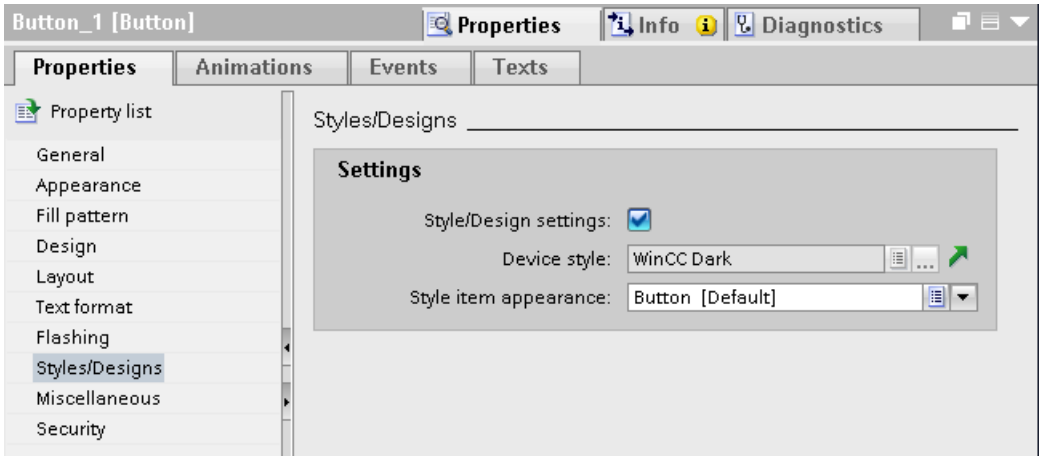

The object is displayed as selected style item.

1.1.10.7 Basics on working with style sheets (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Some screen objects have the same properties which can be configured similarly. These general properties are grouped in style sheets. You can configure several properties for several objects simultaneously and consistently by using style sheets.

Types of style sheets

Style sheets are divided into several predefined categories. Each category contains a specific number of common properties shared by different objects. Select the required category when you create a style sheet. A style sheet can only be applied to objects for which the corresponding category is defined.

The table below shows the available categories and objects that can be configured with them.

Category/Objects	Polylines/ Lines	Text-based objects	Buttons	Value-based objects	Table-based objects	Diagrams
Button	-	-	X	-	X	X
Object body	X	-	-	X	X	X
Diagram body	-	-	-	-	-	X

Category/Objects	Polylines/ Lines	Text-based objects	Buttons	Value-based objects	Table-based objects	Diagrams
Focus and selection	-	X	X	-	X	X
Limits	-	X	-	X	-	-
Labels	-	-	-	X	-	-
Scale	-	-	-	X	-	-
Table body	-	-	-	-	X	X
Table header	-	-	-	-	X	X
Text field	-	X	-	-	-	-

Use of style sheets

You apply a style sheet to a style item when editing a style. The properties configured in the style sheet are transferred to the style item.

To apply a style sheet to a style item, drag the required style sheet from the task card to the style item in the work area using drag-and-drop.

Managing style sheets

Style sheets are library types. You manage style sheets in the project library. Style sheet categories are available in the project library in the "Style sheets > Categories" task card during style editing.

The style sheets that are available for a style sheet type are available in the "Style sheets > Visual attributes" task card. The task card includes predefined style sheets as well as style sheets you have created yourself and released.

1.1.10.8 Creating a new style sheet (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You configure the properties for several objects of a similar type in a style sheet.

Create new style sheet

1. Open the project library.
2. Click the "Add new type" button.
   The "Add new type" dialog opens.
3. Select the option "HMI style sheet" in the dialog.

4. Select the type of the style sheet.

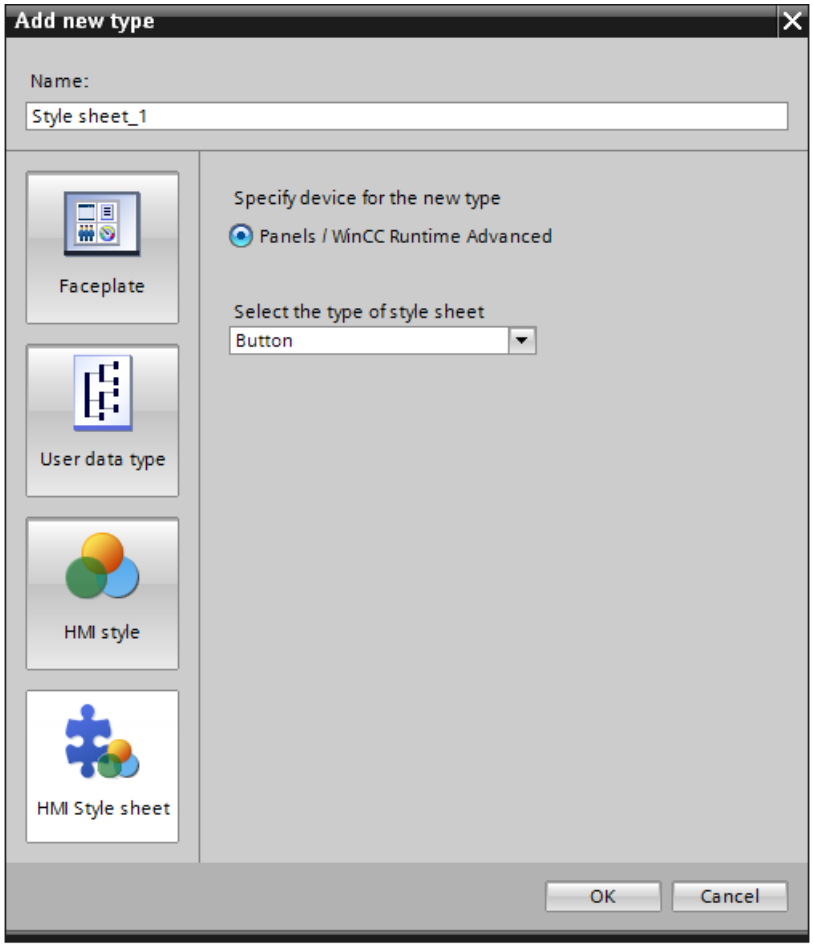

5. Assign a meaningful name to the style sheet.
6. Click "OK" to confirm your input.
   The new style sheet was created.

Create new style sheet based on a style

To automatically configure the properties for similar screen objects when editing a style, create a new style sheet based on properties of a style item that have already been configured.

1. Select the style item you want to use to create a style sheet.
2. Select "Create new style sheet" in the shortcut menu.
3. Select the category for the style sheet in the submenu.
   The "Add type" dialog opens.

4. Assign a meaningful name to the style sheet.
5. Click "OK" to confirm.
   The new style sheet based on a style was created and has been added to the "Style sheets > Visual attributes" task card.
   The properties in the style sheet were applied from the selected style item. The properties of the style sheet not included in the selected style have been assigned default values.

1.1.10.9 Managing style sheets (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You edit, copy and delete the style sheets in the project library.

You can also open, copy, create and delete style sheets with the commands in the shortcut menu of the "Style sheets > Categories" task card.

Edit style sheet

You can edit style sheets that you have created yourself.

1. Open the project library.
2. Select the released version of the style sheet that you want to change.
3. Select "Edit type" in the shortcut menu.
   A new version "in progress" is created.
4. Make the required changes in the Inspector window.
5. Select the "Release version" command from the shortcut menu of the edited version to put the changes into effect.
   The changes made take effect.

Delete style sheet

You can delete style sheets that you have defined yourself.

1. Open the project library.
2. Select the style sheet that you want to delete in the project library.

   Note

   You cannot delete the predefined style sheets of WinCC.

   Note

   You cannot delete the style sheet which includes an "in progress" type. To delete the style sheet, start by releasing all versions of the style sheet.

3. Select the "Delete" command in the shortcut menu.

The style sheet is deleted.

Copy style sheet

1. Open the project library.
2. Select the style sheet that you would like to copy.
3. Select "Duplicate type" in the shortcut menu.
4. Overwrite the suggested name with a meaningful name.
5. In the "Comment" column, enter a description of the style sheet, for example, the intended use.
6. Click "OK" to confirm your input.
   The required style sheet was copied and stored in the same folder.

1.1.10.10 Using style sheets (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You are using a style sheet while editing a style in the style editor. Style sheets are object-specific and are only applied to style items with which they are compatible.

Only a released version of a style sheet can be applied to a style item.

You have the following options to use a style sheet:

- To design the selected style item, drag the style sheet to a style item in the style editor.
- To design several style items of a group, drag the style sheet to a group of style items.
- To design all style items of the style which are compatible with the style sheet, drag the style sheet to an empty area in the style editor.

Requirement

- You have created a style.
- A version of the style is opened "in progress".

Apply style sheet to style item

1. Select the style sheet category in the "Style sheets > Categories" task card.
2. Select the required style sheet in the "Style sheets > Visual attributes" task card.
3. Drag the selected style sheet to a style item with drag&drop.

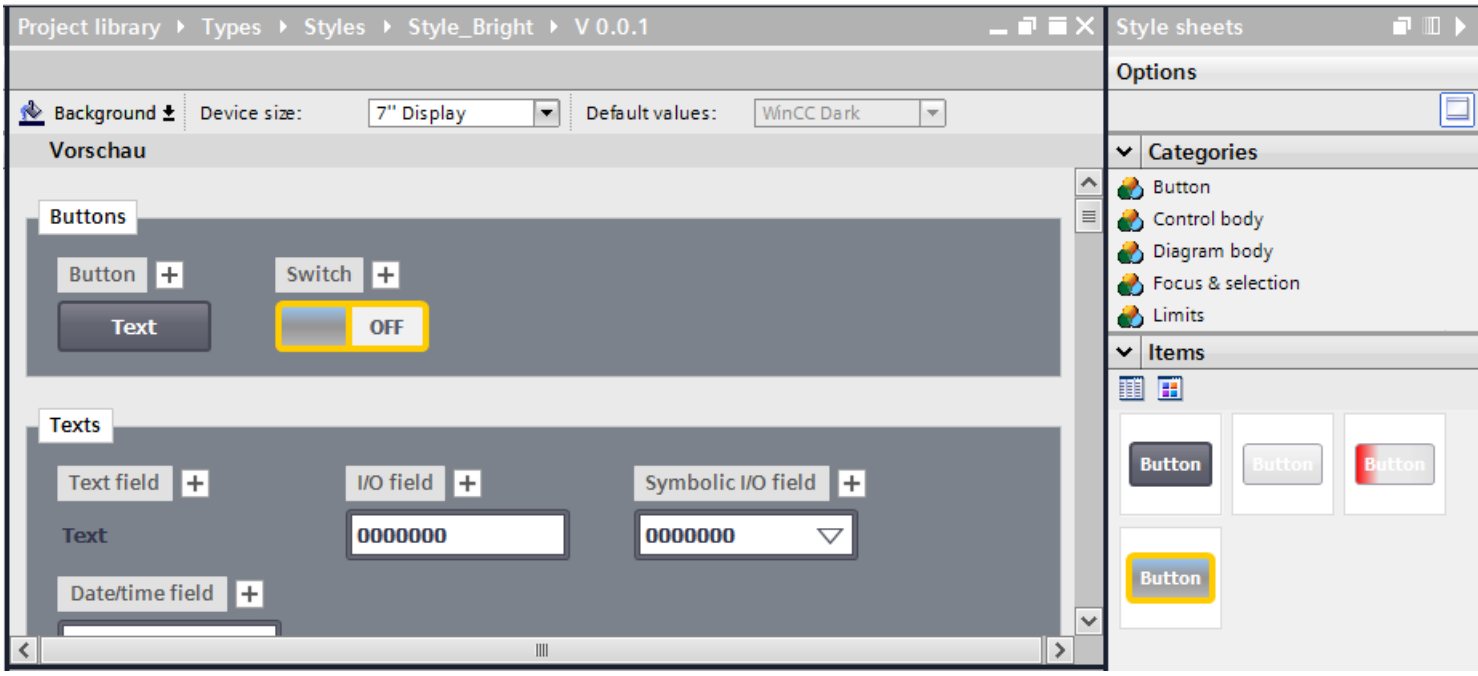

When you drop the style sheet on the style item, the style item applies the settings from the style sheet.

Note

There is no connection between the style item and the used style sheet. When you change the style sheet, the changes are not automatically applied in the style item. Apply the style sheet again to make the changes to the style sheet effective in the style item.

Note

Style sheets are not linked with screen objects. Unused or outdated style sheet versions are deleted by cleaning up or versioning the library.

Result

You have applied a style sheet to a style item in a style.

1.2 Working with objects and object groups

1.2.1 Working with objects

1.2.1.1 Overview of objects (RT Professional)

Introduction

Objects are graphics elements which you use to design the screens of your project.

The "Toolbox" task card contains all objects that can be used for the HMI device.

Display the tool window with menu command "View" by activating the "Task Card" option.

The toolbox contains various palettes, depending on the currently active editor. If the "Screens" editor is open, the toolbox contains the following palettes:

- "Basic objects"
  The basic objects include basic graphic objects such as "Line", "Circle", "Text field" or "Graphic view".
- "Elements"
  The elements include basic control elements, e.g. "I/O field", "Button" or "Gauge".
- "Controls"
  The controls provide advanced functions. They also represent process operations dynamically, for example "Trend view" and "Recipe view".
- "My controls"
  In this palette, you can add ActiveX controls and simple .Net controls to the Toolbox window. From this palette you incorporate ActiveX controls into your project. The following restrictions apply:
  - Only the ActiveX controls that are registered in the operating system of the configuration computer are available.
  - Only the simple .Net controls are supported by the TIA Portal. XAML controls, for example, WPF (Windows Presentation Foundation) controls, are not supported.

- "Graphics"
  Graphics are broken down into subjects in the form of a directory tree structure. The various folders contain the following graphic illustrations:
  - Machine and plant areas
  - Measuring equipment
  - Operator controls

  You can create links to your own graphic folders. The external graphics are located in these folders and subfolders. They are displayed in the toolbox and incorporated into the project using links.
- "Libraries" task card
  In addition to the display and operating objects, the library objects are available. They are located within the palettes of the "Libraries" task card. A library contains preconfigured objects such as graphics of pipes, pumps, or preconfigured buttons. You can also integrate multiple instances of library objects into your project without having to reconfigure them. The WinCC software package includes libraries, e.g. "HMI Buttons & Switches".
  You may also store customized objects and faceplates in user libraries. Faceplates are objects that you create from existing screen objects, and for which you define the configurable properties.

Basic objects

Icon	Object	Instructions
	"Line"	-
	"Polyline"	The polyline is an open object. Even if the start and end points have the same coordinates, the area they enclose cannot be filled in. If you want to fill a polygon, select the "Polygon" object.
	"Polygon"	-
	"Ellipsis"	-
	"Circle"	-
	"Rectangle"	-
	"Circular arc"	The circular arc is an open object. Even if the start and end points have the same coordinates, the area they enclose cannot be filled in. If you want to fill a circle, select the "Circle" or "Circle segment" object.
	"Ellipse arc"	An ellipse arc is an open object. Even if the start and end points have the same coordinates, the area they enclose cannot be filled in. If you want to fill an ellipse, select the "Ellipse" or "Ellipse segment" object.
	"Circle segment"	-
	"Ellipse segment"	-
	"Connector"	Connects objects with lines.
A	"Text field"	One or more lines of text. The font and layout are adjustable.

Icon	Object	Instructions
	"Graphic view"	Displays graphics from external graphic programs, and inserts OLE objects. The following graphic elements can be used: "*.emf", "*.wmf", "*.dib", "*.bmp", "*.jpg", "*.jpeg", "*.gif", "*.tif" and "*.svg".
	"Pipe"	Represents a continuous pipe with several bends in a pipe system.
	"Double T-piece"	Represents a pipe junction in a pipe system.
	"T piece"	Represents a T-shaped pipe connection.
	"Pipe bends"	Represents a bent section of pipe.

Elements

Icon	Object	Instructions
	"I/O field"	Outputs the values of a tag, and/or writes values to a tag. You can define limits for the tag values shown in the I/O field. To hide the operator input in runtime, configure "Hidden input".
	"Editable text field"	Displays several lines of text in a rectangle on the screen.
	"List box"	Displays a selection of several list entries.
	"Combo box"	Displays a selection of several entries in a drop-down list.
	"Button"	Executes a list of functions, or a script as configured.
	"Round button"	Operates processes, such as acknowledge alarms.
	"Symbolic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A text from a text list is displayed in relation to the tag value.
	"Graphic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A graphic from a graphics list is displayed in relation to the tag value.
	"Bar"	The bar represents a value from the PLC in the form of a scaled bar graph.
	"Symbol library"	Used to add screen objects based on controls of the same name.
	"Slider"	Used to display a current value or to enter a numeric value.
	"Scroll bar"	Enters a numerical value at the PLC. The value can be adjusted smoothly.
	"Check box"	Selects options.
	"Option button"	Selects an option.
	"Gauge"	Displays numeric values. The appearance of the gauge is adjustable.
	"Clock"	Displays the system time in analog or digital format.
	"Memory space view"	Shows the existing memory size.

Controls

Icon	Object	Description
	"Screen window"	Displays other screens of the object.
	"f(t) trend view"	Represents multiple curves with values from the PLC, or from a log.
	"f(x) trend view"	Represents the values of a tag as a function of another tag.
	"Table view"	Represents current or archived process data in a table.
	"Value table"	Represents evaluated data and statistics in a table.
	"System diagnostics view"	Provides an overview of all diagnostics capable devices. Displays errors in the plant.
	"User view"	Allows an administrator to administer users on the HMI device. It also allows an operator without administrator rights to change his password.
	"HTML browser"	Displays HTML pages.
	"Print job/Script diagnostics"	Displays data of the Global Script application and of the log system for editing.
	"Recipe view"	Used to display and modify recipes.
	Media Player	Enables video and audio files to be played.
	"Channel diagnostics display"	Allows users to control the active connections.
	"Alarm view"	Shows currently pending alarms or alarm events from the alarm buffer or alarm log.
	"PLC code view"[1]	Displays the PLC programs which have been programmed in the LAD or FBD programming languages.
	"ProDiag overview"[2]	Indicates the current state of operand monitoring.
	"GRAPH overview"[2]	Shows the current status of the program in a sequence and single step view.
	"Criteria analysis view"[3]	Shows the operands with error in the program that were determined by the criteria analysis.

[1] Available as of device version V14.

[2] Available as of device version V14.

[3] Available as of device version V15.

Note

Some of the toolbox objects are either available with restricted functionality, or not at all. This depends on the HMI device you are configuring. Unavailable properties of an object are displayed as deactivated, and cannot be selected.

Note

Support for SVG graphics as of TIA Portal version V15

You can use the SVG graphics in the following screen objects:

Pictures (as background image)

Round buttons

Graphic view

Graphic I/O field

Graphic list entry

See also

Overview of objects (Page 97)

Designing an object (Page 119)

Designing an object (Page 119)

Example: Inserting and configuring a rectangle (Page 172)

1.2.1.2 Overview of objects (RT Advanced, Panels, Comfort Panels)

Introduction

Objects are graphics elements which you use to design the screens of your project.

The "Toolbox" task card contains all objects that can be used for the HMI device. Display the Task Card with menu command "View" by activating the "Task Card" option.

The toolbox contains various palettes, depending on the currently active editor. If the "Screens" editor is open, the toolbox contains the following palettes:

- "Basic objects"
  The basic objects include basic graphic objects such as "Line", "Circle", "Text field" or "Graphic view".
- "Elements"
  The elements include basic control elements, e.g. "I/O field", "Button" or "Gauge".
- "Controls"
  The controls provide advanced functions. They also represent process operations dynamically, for example Trend view and Recipe view.

- "Graphics"
  Graphics are broken down into subjects in the form of a directory tree structure. The various folders contain the following graphic illustrations:
  - Machine and plant areas
  - Measuring equipment
  - Operator controls
  - Flags
  - Buildings

  You can create links to your own graphic folders. The external graphics are located in these folders and subfolders. They are displayed in the toolbox and incorporated into the project using links.
- "Libraries" task card
  In addition to the display and operating objects, the library objects are available. They are located within the palettes of the "Libraries" task card. A library contains preconfigured objects such as graphics of pipes, pumps, or preconfigured buttons. You can also integrate multiple instances of library objects into your project without having to reconfigure them.
  The WinCC software package includes libraries, e.g. "HMI Buttons & Switches".
  You can also store customized objects, and faceplates in user libraries.
  Faceplates are objects that you create from existing screen objects, and for which you define the configurable properties.

Note

HMI device dependency

Some of the toolbox objects are either available with restricted functionality, or not at all. This depends on the HMI device you are configuring. Unavailable properties of an object are displayed as deactivated, and cannot be selected.

Basic objects

Icon	Object	Instructions
	"Line"	-
	"Polyline"	The polyline is an open object. Even if the start and end points have the same coordinates, the area they enclose cannot be filled in. If you want to fill a polygon, select the "Polygon" object.
	"Polygon"	-
	"Ellipsis"	-
	"Circle"	-
	"Rectangle"	-
	"Text field"	One or more lines of text. The font and layout are adjustable.
	"Graphic view"	Displays graphics from external graphic programs, and inserts OLE objects. The following graphic formats can be used: "*.emf", "*.wmf", "*.dib", "*.bmp", "*.jpg", "*.jpeg", "*.gif" and "*.tif".

Elements

Icon	Object	Instructions
	"I/O field"	Outputs the values of a tag, and/or writes values to a tag. You can define limits for the tag values shown in the I/O field. To hide the operator input in runtime, configure "Hidden input".
	"Button"	Executes a list of functions, or a script as configured.
	"Symbolic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A text from a text list is displayed in relation to the tag value.
	"Graphic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A graphic from a graphics list is displayed in relation to the tag value.
	"Date/time field"	Outputs the system date and time, or the time and date from a tag. This allows the operator to enter new values. The display format is adjustable.
	"Bar"	The bar represents a value from the PLC in the form of a scaled bar graph.
	"Charge condition"	Shows the charge condition of the battery for Mobile Wireless
	"Switch"	Toggles between two defined states. You can label a switch with text, or a graphic.
	"Slider"	Displays a current value from the PLC, or sends a numeric value to the PLC.
	"Symbol library"	Used to add screen objects based on controls of the same name.
	"Effective range name"	Shows the name of the effective range.
	"Effective range name (RFID)"	Displays the name of the effective range (RFID).
	"Effective range signal"	Indicates how accurately the Mobile Panel Wireless is positioned within an effective range.
	"WLAN reception"	Indicates how good the WLAN radio connection is.
	"Gauge"	Displays numeric values. The appearance of the gauge is adjustable.
	"Zone name"	Displays the name of the zone.

Icon	Object	Instructions
	"Zone signal"	Indicates how accurately the Mobile Panel Wireless is positioned within a zone.
	"Clock"	Displays the system time in analog or digital format.

Controls

Icon	Object	Description
	"Alarm view"	Shows currently pending alarms or alarm events from the alarm buffer or alarm log.
	"Trend view"	Represents multiple curves with values from the PLC, or from a log.
	"User view"	Allows an administrator to administer users on the HMI device. It also allows an operator without administrator rights to change his password.
	"HTML browser"	Displays HTML pages.
	"Camera view"[1]	Displays pictures from a connected network camera.
	"PDF view"	Displays PDF documents.
	"Watch table"	This function gives the operator direct read / write access from the HMI device to individual address areas in the connected SIMATIC S7.
	"Sm@rtClient view"	Allows an operator to monitor and maintain a connected device remotely.
	"Recipe view"	Displays data records, and allows them to be edited.
	"f(x) trend view"	Represents the values of a tag as a function of another tag.
	"System diagnostics view"	Provides an overview of all diagnostics capable devices. Displays errors in the plant.
	"PLC code view"[2]	Displays the PLC programs which have been programmed in the LAD or FBD programming languages.
	"GRAPH overview"[3]	Shows the current program status for executed steps of the PLC sequencer.
	"ProDiag overview"[3]	Indicates the current state of operand monitoring.
	"Criteria analysis view"[4]	Shows the operands with error in the program that were determined by the criteria analysis.
	"Media Player"	Media files are shown in the media player.

[1] Available for Comfort Panels and KTP Mobile Panels (as of device version V13) and RT Advanced (as of device version V14 SP1)

[2] Available for WinCC Runtime Advanced (as of device version V14) and Comfort Panels and KTP Mobile Panels (as of device version V14 with a display size > 4").

[3] Available for WinCC Runtime Advanced, Comfort Panels and KTP Mobile Panels (as of device version V14)

[4] Available as of device version V15.

See also

Overview of objects (Page 93)

Options for editing objects (Page 104)

Inserting an object (Page 105)

Deleting an object (Page 107)

Positioning an object (Page 108)

Resizing objects (Page 109)

Selecting multiple objects (Page 111)

Aligning objects (Page 113)

Moving an object forwards or backwards (Page 115)

Showing objects outside the screen area (Page 116)

Rotating objects (Page 117)

Flipping objects (Page 118)

Designing an object (Page 119)

Designing an object (Page 119)

Hiding or showing objects (Page 141)

Managing own controls (Page 144)

External graphics (Page 147)

Managing external graphics (Page 148)

Storing an external image in the graphics library (Page 149)

Basics on groups (Page 157)

Overview of keyboard access (Page 165)

Example: Inserting and configuring a rectangle (Page 172)

Flashing (Page 138)

Inserting multiple objects of the same type (stamping) (Page 138)

Screen basics (Page 37)

Flashing (Page 137)

Overview of objects (Page 102)

Rotating an object in runtime (Page 129)

Repositioning and resizing multiple objects (Page 140)

1.2.1.3 Overview of objects (Basic Panels)

Introduction

Objects are graphics elements which you use to design the screens of your project.

The "Tools" task card contains all objects that can be used for the HMI device. Display the Task Card with menu command "View" by activating the "Task Card" option.

The toolbox contains various palettes, depending on the currently active editor. If the "Screens" editor is open, the toolbox contains the following palettes:

- "Basic objects"
  The basic objects include basic graphic objects such as "Line", "Circle", "Text field" or "Graphic view".
- "Elements"
  The elements include basic operator controls, e.g. "I/O field", "Button" or "Gauge".
- "Controls"
  The controls provide advanced functions. They also represent process operations dynamically, for example Trend view and Recipe view.
- "Graphics"
  Graphics are broken down into subjects in the form of a directory tree structure. The various folders contain the following graphic illustrations:
  - Machine and plant areas
  - Measuring equipment
  - Operator controls
  - Flags
  - Buildings

  You can create links to your own graphic folders. The external graphics are located in these folders and subfolders. They are displayed in the toolbox and incorporated into the project using links.
- "Libraries" task card
  In addition to the display and operating elements, the library objects are available. They are located within the palettes of the "Libraries" task card. A library contains preconfigured objects such as graphics of pipes, pumps, or preconfigured buttons. You can also integrate multiple instances of library objects into your project without having to reconfigure them. The WinCC software package includes libraries, e.g. "HMI Buttons & Switches".
  You can also store customized objects, and faceplates in user libraries.
  Faceplates are objects that you create from existing screen objects, and for which you define the configurable properties.

Note

HMI device dependency

Some of the toolbox objects are either available with restricted functionality, or not at all. This depends on the HMI device you are configuring. Unavailable properties of an object are displayed as deactivated, and cannot be selected.

Basic objects

Symbol	Object	Instructions
	"Line"	-
	"Ellipse"	-
	"Circle"	-
	"Rectangle"	-
	"Text field"	One or more lines of text. The font and layout are adjustable.
	"Graphic view"	Displays graphics from external graphic programs, and inserts OLE objects. The following graphic formats can be used: "*.emf", "*.wmf", "*.dib", "*.bmp", "*.jpg", "*.jpeg", "*.gif" and "*.tif".

Elements

Symbol	Object	Instructions
	"I/O field"	Outputs the values of a tag, and/or writes values to a tag. You can define limits for the tag values shown in the I/O field. To hide the operator input in runtime, configure "Hidden input".
	"Button"	Executes a function list, or a script as configured.
	"Symbolic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A text from a text list is displayed in relation to the tag value.
	"Graphic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A graphic from a graphics list is displayed in relation to the tag value.
	"Date/time field"	Outputs the system date and time, or the time and date from a tag. This allows the operator to enter new values. The display format is adjustable.
	"Bar"	The bar represents a value from the PLC in the form of a scaled bar graph.
	"Switch"	Toggles between two defined states. You can label a switch with text, or a graphic.

Controls

Symbol	Object	Description
	"Alarm view"	Shows currently pending alarms or alarm events from the alarm buffer or alarm log.
	"Trend view"	Represents multiple curves with values from the PLC, or from a log.
	"User view"	Allows an administrator to administer users on the HMI device. It also allows an operator without administrator rights to change his password.
	"HTML Browser"[1)]	Displays HTML pages.

Symbol	Object	Description
	"Recipe view"	Displays data records, and allows them to be edited.
	"System diagnostics view"	Provides an overview of all devices capable of diagnostics. Displays errors in the plant.

[1)]Available for Basic Panels 2nd Generation.

See also

Overview of objects (Page 97)

1.2.1.4 Options for editing objects

Introduction

Objects are graphics elements which you use to design the screens of your project.

You have the following options for editing objects:

- Copying, pasting or deleting objects using the shortcut menu If you copy an object in a screen and the screen already includes an object of the same name, the name of the object is changed.
- Maintaining the standard size of the objects you are inserting or customizing their size on insertion.
- Moving an object in front of or behind other objects
- Rotating objects
- Mirroring objects
- Defining the tab sequence for objects
- Stamping: Inserting several objects of the same type
- Selecting several objects simultaneously
- Repositioning and resizing multiple objects
- You can assign external graphics to objects, e.g. in the Graphic View.
  You can view only the images you have previously stored in the graphic browser of your WinCC project.
  You can save graphics in the graphic browser:
  - Via drag & drop from the "Graphics" object group to the working area
  - As a graphic file in the following formats: *.bmp, *.dib, *.ico, *.emf, *.wmf, *.gif, *.tif, *.svg, *.jpeg or *.jpg
  - As an OLE object
    You either create a new OLE object or save an existing graphic file as an OLE object.
    To save an OLE object, an OLE-compatible graphics program must be installed on the configuration computer.

See also

Inserting an object (Page 105)
Deleting an object (Page 107)
Positioning an object (Page 108)
Resizing objects (Page 109)
Selecting multiple objects (Page 111)
Aligning objects (Page 113)
Moving an object forwards or backwards (Page 115)
Showing objects outside the screen area (Page 116)
Rotating objects (Page 117)
Flipping objects (Page 118)
Inserting multiple objects of the same type (stamping) (Page 138)
Hiding or showing objects (Page 141)
Managing own controls (Page 144)
External graphics (Page 147)
Managing external graphics (Page 148)
Storing an external image in the graphics library (Page 149)
Example: Inserting and configuring a rectangle (Page 172)
Overview of objects (Page 97)
Repositioning and resizing multiple objects (Page 140)

1.2.1.5 Inserting an object

Introduction

In the "Screens" or "Reports" editor, insert the objects to the "Toolbox" task card. Use the mouse to drag the objects into the work area. You either keep the objects in their original size, or scale them up or down when you paste them.

In addition, you can copy or move objects via the clipboard from one editor to another, for example to transfer a screen object to a report. Alternatively, you can also use the mouse instead of the clipboard for copying and moving:

- Copy: <Ctrl + Drag&Drop>
- Moving: Drag&drop

Note
Basic Panels

The "Reports" editor is not available for Basic Panels.

Requirement

The "Tools" task card is open.

Inserting objects in their standard size

1. In the "Toolbox" task card, select the desired graphic object or the desired graphic in the WinCC graphics folder.
   When you move the cursor across the work area, it turns into a crosshair with an appended object icon.
2. Click the location in the work area where you want to insert the object or graphic.
   The object is inserted with its standard size at the desired position in the work area.

Alternatively, double-click the object in the "Toolbox" task card.

Copying an object

1. Select the desired object.
2. Select "Copy" in the shortcut menu.
3. Click the desired location and select "Paste" in the shortcut menu.

WinCC inserts a copy of the object at the desired location. You can only change the properties that are appropriate for the relevant context.

Example: In the "Screens" editor, you can set the mode for input and output for I/O fields. In the "Reports" editor, the mode is set to "Output".

Original and copy are not interconnected and are configured independently from one another.

Inserting lines

1. Select the desired graphic object in the "Tools" task card.
2. Click on a location in the work area. A line in the standard size is inserted.

Inserting a polygon or polyline

1. Select the desired object "Polyline" or "Polygon" in the "Tools" task card.
2. Click on a location in the work area. This fixes the starting point of the object.
3. Click another location in the work area. A corner point is set.
4. For every additional corner point, click on the corresponding location in the work area.
5. Double-click on a location in the work area. The last corner point is set.
   This fixes all the points of the polygon or polyline.

Note

Basic Panels

The "Polyline" and "Polygon" objects are not available for Basic Panels.

Note

If you want to insert several objects of the same type, use the "Stamp" function. This avoids having to reselect the object in the "Tools" task card every time before inserting it. To do so, select the icon in the toolbar of the "Tools" task card.

See also

Moving an object forwards or backwards (Page 115)

Overview of objects (Page 97)

1.2.1.6 Deleting an object

Introduction

You can delete objects individually or with a multiple selection.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to delete.
   To delete multiple objects, keep the <Shift> key pressed and select the objects to be deleted one after the other. Alternatively, drag and maximize an area around the desired objects with the mouse.
2. Select "Delete" from the shortcut menu.

Result

The selected objects are deleted.

See also

Overview of objects (Page 97)

1.2.1.7 Positioning an object

Introduction

When you select an object, it is enclosed by a rectangle with resizing handles. This rectangle is the selection rectangle. The position of an object is defined by the coordinates of the top left corner of the rectangle surrounding the object.

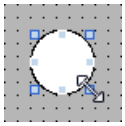

Note

If the position is outside the work area the object is not displayed in Runtime.

Position and align

You have the possibility of having a grid displayed in the work area. You have three options for easier positioning of objects:

- "Snap to grid" When you resposition objects, they are automatically snapped and pasted to the grid. If you hold down the <Alt> key, the object is no longer snapped to the grid.
- "Snap to objects" When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.
- "None": You position the objects in any position.

You activate and deactivate the grid and options as follows:

- In the "Options > Settings > Visualization > Screens" menu
- In the "Layout > Grid" task card

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object you want to move.
   The selected object is framed by a rectangle with resizing handles.

   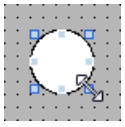

2. Left-click the object and keep the mouse button pressed.

3. Move the mouse pointer onto the new position.
   The contour of the object moves with the mouse and displays the new position for the object.

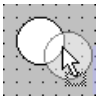

   The object initially remains at its original position.
4. Now release the mouse button.
   The object is moved into the position indicated by the contour of the selection rectangle.

Alternative procedure

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the X and Y values for the position under "Position & Size".

Result

The object appears at its new position.

See also

Overview of objects (Page 97)

1.2.1.8 Resizing objects

Introduction

When you select an object, it is enclosed by a rectangle with handles. You have the following options of resizing an object:

- Drag the handles using the mouse.
- Modify the "Size" property in the Inspector window.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object you want to resize.
   The selection rectangle appears. The following figure shows a selected object:

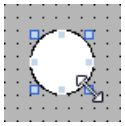

2. Drag a resizing contact of the rectangle to a new position.
   The size of the object changes.
   – The size of the object is aligned to the grid pattern, provided the "Snap to grid" function is set.
   – Press <ALT> to disable this function while you drag the object.
     In order to scale the object proportionally, keep the <Shift> key pressed while changing the size with the mouse.

Alternative procedure

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the size of the object under "Position & Size".

Harmonizing the object size

1. Select the objects.
2. Now, click one of the following buttons: or or
   The size of the selected objects is matched to each other.

The following screen shows how the selected objects are adapted to the height of the reference object:

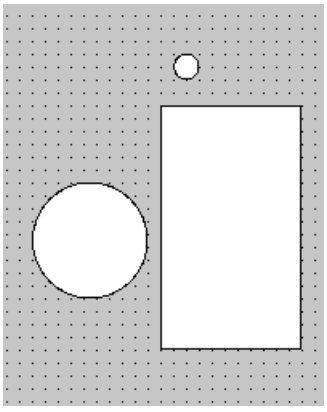

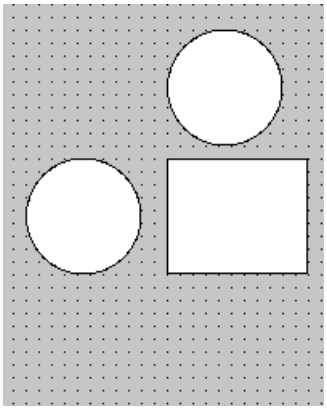

Icon	Description
	Aligns the selected objects to the width of the reference object.
	Aligns the selected objects to the height of the reference object.
	Aligns the selected objects to the width and height of the reference object.

Result

The object now appears with its new size.

See also

Overview of objects (Page 97)

1.2.1.9 Selecting multiple objects

Introduction

Select all objects you want to align with each other or to change global properties. This procedure is called "multiple selection."

The Inspector window shows all the properties of the selected objects.

You now have several options of selecting multiple objects:

- Draw a selection frame around the objects.
- Hold down the <Shift> key, and click the required objects.

Selection frame of a multiple selection

The selection frame surrounds all objects of a multiple selection. The selection frame is comparable with the rectangle that surrounds an individual object.

The selection frame is not visible. When you have made your multiple selection, the following frame is displayed:

- The reference object is indicated by the rectangle around it.
- The other selected objects are indicated by a dashed-line frame.

Specifying a reference object

The reference object is the object upon which the other objects are oriented. The reference object is framed by a rectangle with handles. The following figure shows a reference object with two other selected objects:

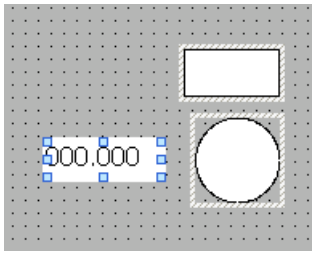

You have the following options to specify the reference object:

- Select the objects via multiple selection. The object selected first is then the reference object.
- Draw a selection frame around the objects. The reference object compiled automatically. If you wish to specify a different object within the selection as the reference object, click on the desired object. This action does not cancel your multiple selection.

Requirement

You have opened the work area containing at least two objects.

Selecting multiple objects with a selection frame

1. Position the mouse pointer in the work area close to one of the objects to be selected.
2. Hold down the mouse button, and draw a selection frame around the objects to be selected.

Or:

1. Hold down the <Shift> key.
2. Click the relevant objects, working in succession.
   All the selected objects are identified by frames.
   The object selected first is identified as reference object.

Note

To remove an object from the multiple selection, press <SHIFT>, hold it down and then click the relevant object once again.

Result

Multiple objects are selected. One of those is identified as the reference object. You can now perform the following steps:

- Changing the object properties of all the objects
- Resizing all the objects by the same ratio, by dragging the selection frame to increase or reduce the size

- Moving all the objects in one group
- Aligning the objects to the reference object

See also

Overview of objects (Page 97)

1.2.1.10 Aligning objects

Procedure

1. Select the objects via multiple selection.
2. Specify an object as the reference object.
3. Select the desired command in the toolbar or the shortcut menu - see table below.
   The selected objects will be aligned.

Aligning objects flush

The selected objects will be aligned flush to the reference object.

Icon	Description
	Aligns the selected objects to the left edge of the reference object.
	Aligns the selected objects to the vertical center axis of the reference object.
	Aligns the selected objects to the right edge of the reference object.
	Aligns the selected objects to the upper edge of the reference object.
	Aligns the selected objects to the horizontal center axis of the reference object.
	Aligns the selected objects to the lower edge of the reference object.
	Centers the selected objects to the center points of the reference object.
	Centers the selected objects vertically in the screen.

Snap to object

When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.

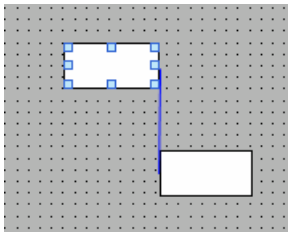

If you are working with the keyboard, press the Alt key. When you move the selected object with the arrow keys, the next anchor point is displayed.

Distributing objects evenly

You need at least three selected objects. A reference object is not required.

1. Select the objects.
2. Click one of the buttons "Distribute horizontally equal" or "Distribute vertically equal". The selected objects are distributed at equal distances.

The following screen shows how you align the vertical spacing of the selected objects:

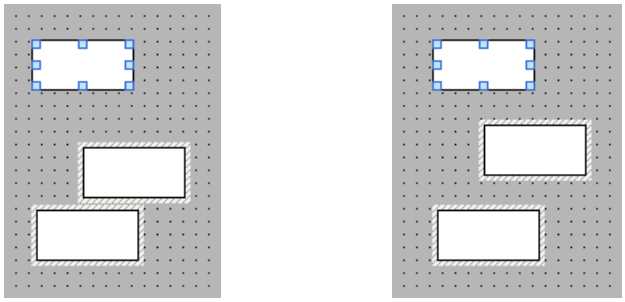

Icon	Description
	Aligns the horizontal distance between the objects. The position of the objects on the extreme left and right side remains unchanged. All other objects are distributed evenly between them.
	Aligns the vertical distance between the objects. The position of the objects at the extreme top and bottom (right and left) remains unchanged. All other objects are distributed evenly between them.

See also

Overview of objects (Page 97)

1.2.1.11 Moving objects with keyboard shortcuts

Introduction

You can move the objects in the screen by using shortcut keys or select them by type.

Controlling objects with the keyboard

Function	Keyboard shortcut
Moves the selected screen object to the left to the first available vertical alignment position.	<Alt+Shift+Left>
Moves the selected screen object to the right to the first available vertical alignment position.	<Alt+Shift+Right>
Moves the selected screen object up to the first available horizontal alignment position.	<Alt+Shift+Up>

Function	Keyboard shortcut
Moves the selected screen object down to the first available horizontal alignment position.	<Alt+Shift+Down>
Selects all screen objects of this type.	<Ctrl+Shift+A>
Selects all objects that support the use styles.	<Ctrl+Shift+D>

1.2.1.12 Moving an object forwards or backwards

Introduction

You can use the "Order" functions in the shortcut menu of a selected object or in the toolbar to move a selected object in front of or behind other objects within an object layer.

Note

ActiveX controls are always positioned in front of an object layer (.NET property).

Requirement

You have opened a screen which contains a layer with multiple objects.

Procedure

1. Select the object you want to move forward or backward.
2. Select the "Sort" command and one of the following commands from the shortcut menu:

Icon	Description
	Moves the selected object before all the other objects of the same layer
	Moves the selected object to the lowest position in the same layer
	Moves the selected object up by one position
	Moves the selected object down by one position

Alternative procedure

1. Open the "Layers" palette of the "Layout" task card.
2. Navigate to the required object.
3. Hold down the mouse button, and drag the object in the tree topology to the required position in the layer.
4. Now release the mouse button.

Result

The object is moved up or down.

See also

Inserting an object (Page 105)

Overview of objects (Page 97)

1.2.1.13 Showing objects outside the screen area

Introduction

If you assign objects to positions that are outside the configurable area, these objects will be hidden. The functions of the "Objects outside the visible area" palette in the "Layout" task card are used to move these objects back into the screen.

Requirement

- You have opened a screen which contains objects that are outside the configurable area.
- The "Layout" task card is open.

Procedure

1. Open the "Layout > Objects outside the area" task card.
   This displays a list of objects that are outside the configurable area.
2. Select the the object which you want to move back into the screen from the list.
3. Select "Move to screen" in the object shortcut menu.

Alternatively open the "Layout > Layer" task card. Objects outside the area are indicated by the icon. If you click this icon, the object is moved back into the screen.

Result

The objects are moved to the configurable area.

See also

Overview of objects (Page 97)

1.2.1.14 Rotating objects

Introduction

You can rotate a suitable object clockwise or counterclockwise around its center axis in steps of 90°.

Note

Not all the objects can be rotated. Some objects that can be rotated in screens cannot be rotated in reports.

You can also rotate multiple objects using the multiple selection function. Certain WinCC objects such as buttons cannot be rotated.

The alignment of elements in an object will change in a rotated object. The following figure shows how a rectangle and an ellipse behave under the different commands for rotating an object:

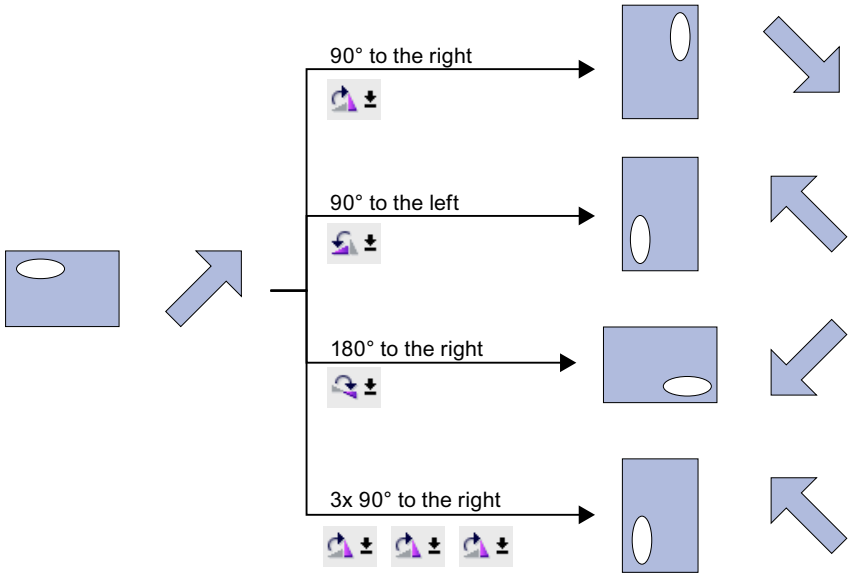

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to rotate.
2. Click one of the following toolbar icons:
   , to rotate the object clockwise around its center point. The angle of rotation is 90°.
   , to rotate the object counterclockwise around its center point. The angle of rotation is 90°.
   , to rotate the object clockwise by 180°.

Result

The object is shown at its new angle.

See also

Overview of objects (Page 97)

1.2.1.15 Flipping objects

Introduction

You can flip an object along its vertical or horizontal center axis. The alignment of elements in an object will change when you flip an object. The following figure shows how a rectangle and an ellipse behave under the different commands for flipping an object.

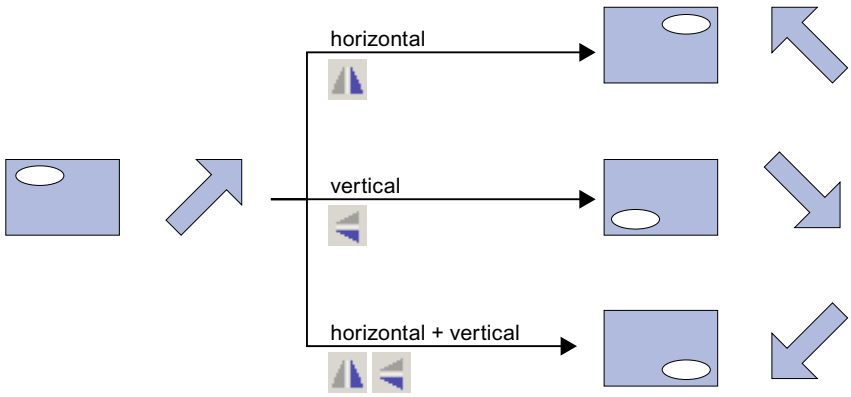

Requirement

You have opened a screen which contains at least one object.

Procedure

1. Select the object that you want to flip.
2. Click the "Flip" command in the shortcut menu and select one of the options displayed:
   – , to flip the selected object along its vertical center axis.
   – , to flip the selected object along its horizontal center axis.

Result

The object is shown at its flipped position.

See also

Overview of objects (Page 97)

1.2.1.16 Designing an object (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You design the border and background of an object.

Requirements

A line has been created in a screen.

Procedure

1. Select the line on your screen.
2. In the Inspector window, select "Properties > Properties > Appearance":
3. Select "Dash" as the style.
4. To display the dashed line in two colors, select the line width "1".
5. Select the setting "Arrow" in the "Line ends" area.

Result

The line is displayed in two colors as a dashed line. The end of the line is an arrow.

See also

Overview of objects (Page 93)

1.2.1.17 Designing an object (RT Professional)

Introduction

You design the border and background of an object.

Requirement

A circle has been created in a screen.

Procedure

1. Select the circle in the screen.
2. In the Inspector window, select "Properties > Properties > Appearance":
3. Select "Solid" as the background fill pattern.
4. Activate "Fill level".

5. Enter "50" for the fill level.
6. Select "Solid" as the style.

Result

The object is displayed as a semicircle without a border.

See also

Overview of objects (Page 93)

1.2.1.18 Designing the fill pattern

Introduction

WinCC lets you design the background color and the fill pattern of an object. The design options change in the Inspector window depending on object for which you are making the filling pattern.

For certain objects, you can not only define the color, but also a transparent background or a background with a color gradient.

Note

Device-specific nature of the properties

The fill style that is available depends on the object and the HMI device used.

Requirement

The object has been created and selected.

Designing the background color of an object

1. In the Inspector window, select "Properties > Properties > Appearance".
2. Select a color for the background of the object, for example, yellow.
   The object is filled with the selected color.

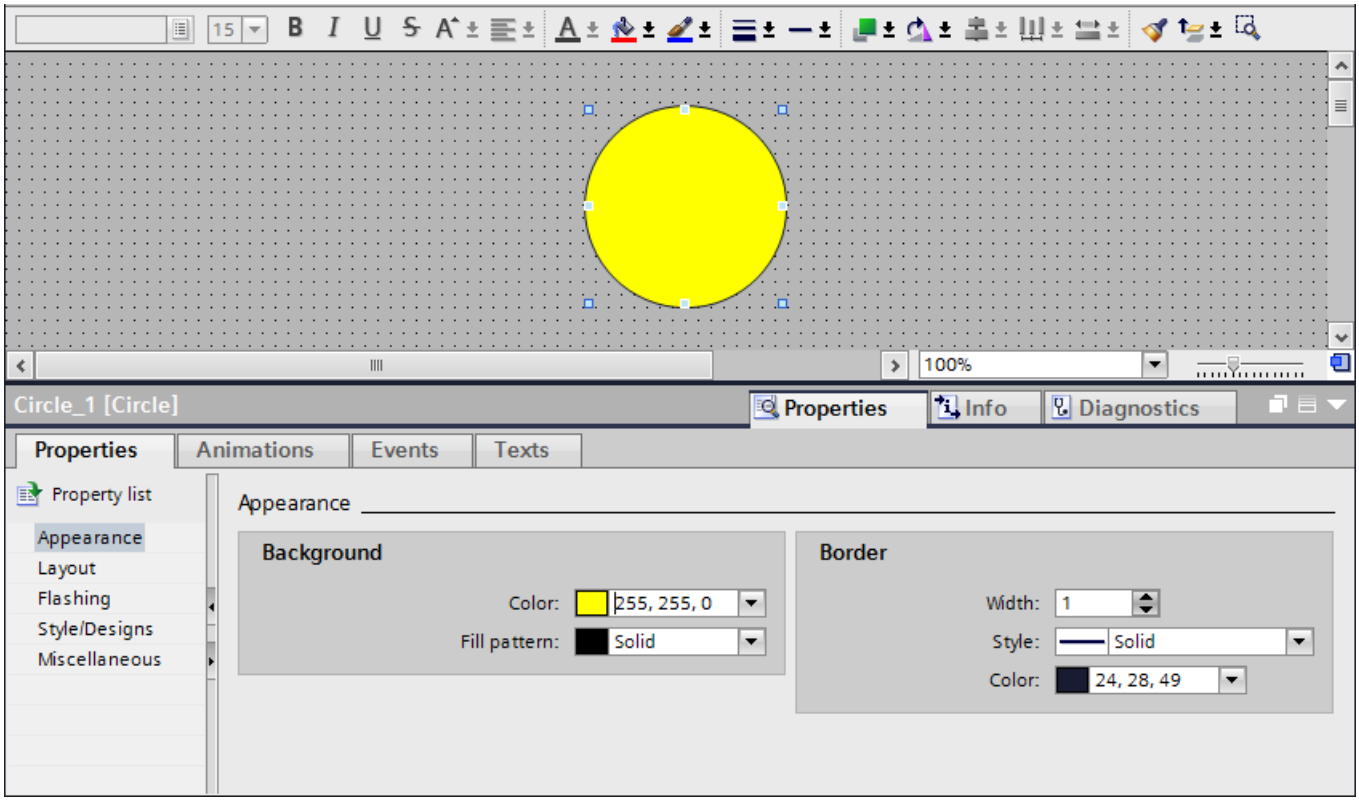

Designing the fill pattern of an object

1. In the Inspector window, select "Properties > Properties > Appearance".
2. To define a transparent background for the object, select transparent.
   You can find information about designing a fill pattern with a color gradient in the section Defining color gradients (Page 133).
   The object is shown as transparent.

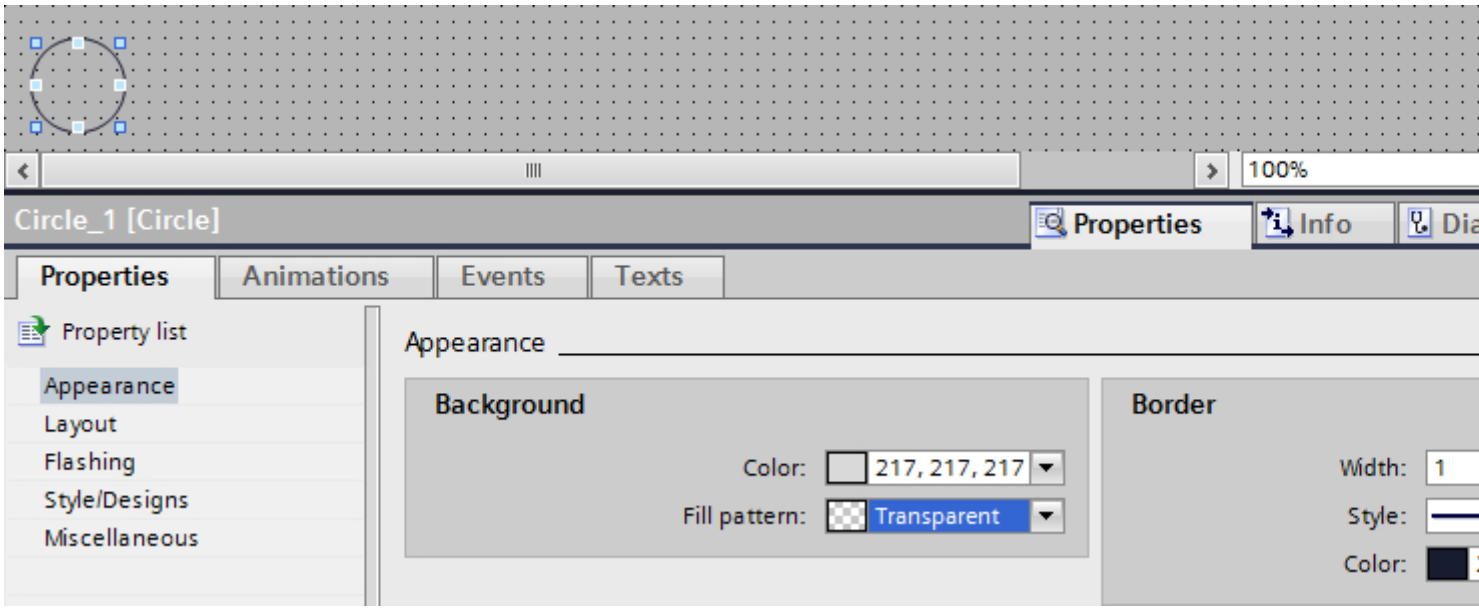

You can optionally set the fill pattern in the Inspector window under "Properties > Properties > Fill Pattern".

See also

Defining color gradients (Page 133)

1.2.1.19 Specifying scroll bars

You can use the scroll bars for navigation in the table-based objects:

- Use the scroll arrows to scroll line by line either up or down or from left to right in the view.
- Use the scroll box in the scroll bar to scroll slowly to the required position in an object.
- Click on the scroll box to navigate quickly in the object through a large volume of data.

Note

On HMI devices with keys, you can move the scroll bar with arrow keys.

In pop-up screens, use <ALT>+<arrow key> to scroll.

In WinCC version V14 and higher, you can configure different colors for the scroll bars.

Scrolling in controls

Two display options for scrolling are available under "Runtime settings > Screens > Scrolling in controls":

- Scroll bar with scroll arrows for devices with device version V14 or higher
  The scroll bar is displayed in the screen object with scroll arrows and scroll box.
- Scroll indicator
  A scroll indicator is shown in the screen object

In the Engineering System, you can see the scroll preview in all controls with the exception of HTML browser, PDF display and NC partial program display.

Note

The display of scroll indicators is not supported in the PLC code display or in the pop-up screens.

Showing all data

The scroll bars are used to display content in limited space on the screen. Please remember in configuration that the scroll bars themselves take up space in the control. The width and height of the scroll bars vary depending on the screen resolution.

If you configure a control to display just a few lines, the scroll bars may cover the content in runtime. To avoid hiding the data, take account of the scroll bars in configuration.

Note

When configuring the user display, avoid the setting "Fit object to contents". This setting calculates the object size in runtime automatically and cannot be changed.

Configuring scroll bar colors

You can configure the color of the scroll bar in conjunction with other colors for certain objects.

- The scroll bar arrows and scroll box are shown in the text color of the table
- The scroll bar background color is the alternative color of the table

Note

In the objects graphic I/O field, HTML browser, PDF display and file browser, the scroll bar colors cannot be changed.

Example

This example shows the configuration of scroll bar colors for an alarm view.

1. Open "Properties > Properties > Appearance".
2. Select a color for the table foreground, for example "blue" for the scroll arrows and the scroll box in the scroll bar.
3. Select the alternative color, for example "yellow" for the even rows in the table and the background of the scroll bar.
   The scroll bar is displayed in the configured colors.

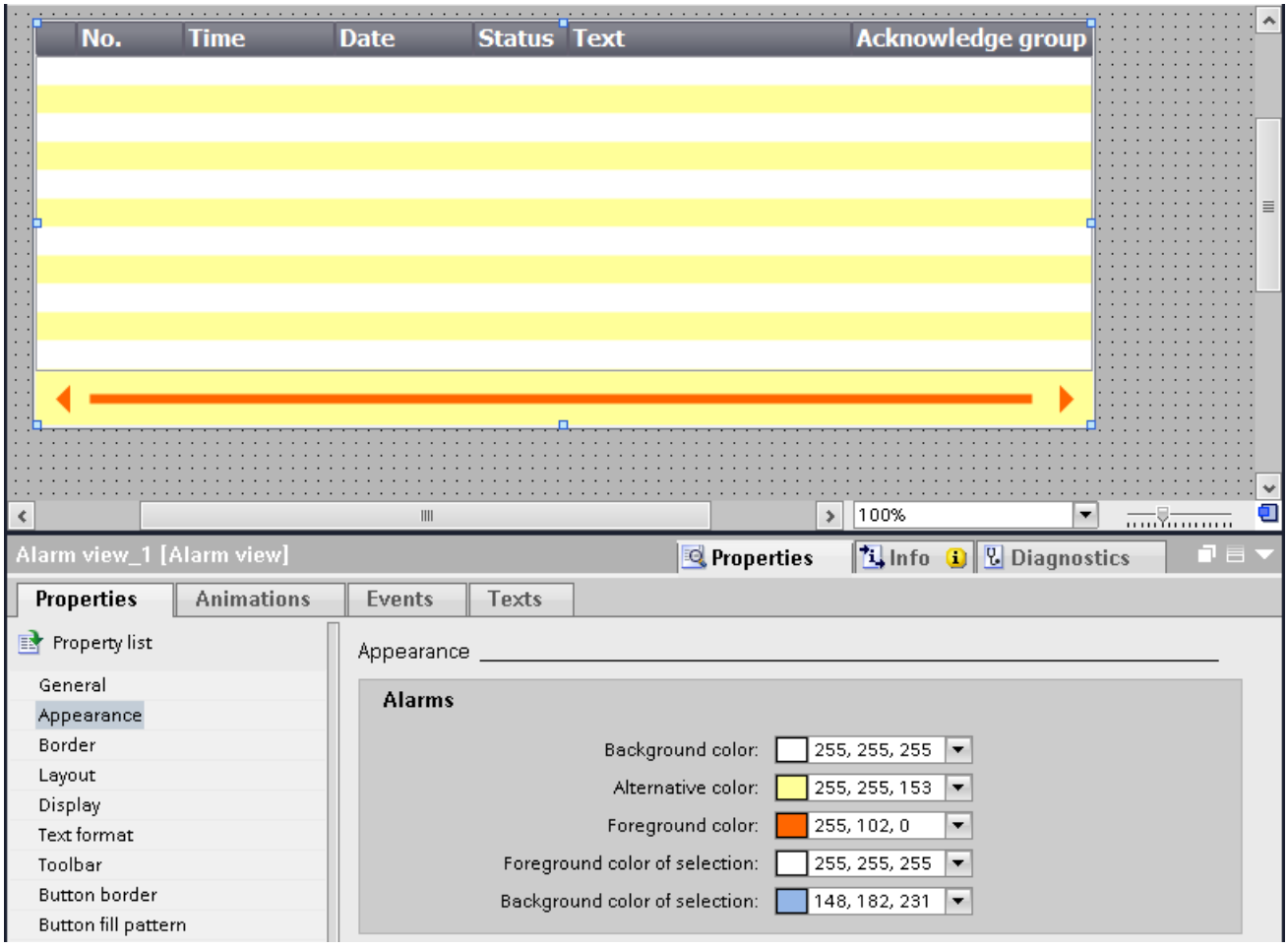

1.2.1.20 Using predefined styles

Introduction

You can assign predefined styles to objects and faceplates in runtime. You can change the background color of the display and operating elements using predefined styles. In this way, you harmonize the display in Runtime.

Requirement

The "Toolbox" task card is open.

Setting predefined styles

1. Select the object that you want to insert in the "Toolbox" task card.
2. Select one of the following options in the "Toolbox" task card:
   – "Default" to use the default style
   – "Use device design" to use the color settings from the current device design
3. Insert the required object in the work area.
   The object is displayed in the selected style.
4. The objects are created in the selected style as long as the style remains activated in the toolbar.
   To return to the default style, select the "Default" option in the toolbar.

See also

Working with styles and style sheets (Page 78)

1.2.1.21 Formatting text in an object

Introduction

Some objects, for example, the I/O field, support text within the object.

You have several options to align text.

Configuring the distance from the object border

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the value "5" for the left border, for example.

The text is aligned five pixels from the left border of the object.

Aligning the text position

1. In the Inspector window, select "Properties > Properties > Text format".
2. Select the horizontal alignment, for example, centered.
3. Select the vertical alignment, for example, top.
   The text is shown at the top center in the object.

Defining the text orientation

1. In the Inspector window, select "Properties > Properties > Text format".
2. Select the orientation of the text, for example, vertical, right.
   The text flow is shown from bottom to top.

1.2.1.22 Connecting tags and text lists in the text (RT Advanced)

Introduction

Comfort Panels and WinCC Runtime Advanced give you the option to add output fields for tags or entries to a text list for the following objects:

- Symbolic I/O field
- Text field

Note

The symbolic I/O field supports the connection of tags and text list entries in the "Two states" mode.

Requirement

- The object has been created and selected

Connecting tags in a text field

1. Open "Properties > Properties > General" in the inspector window.
2. Open the shortcut menu of the input field.

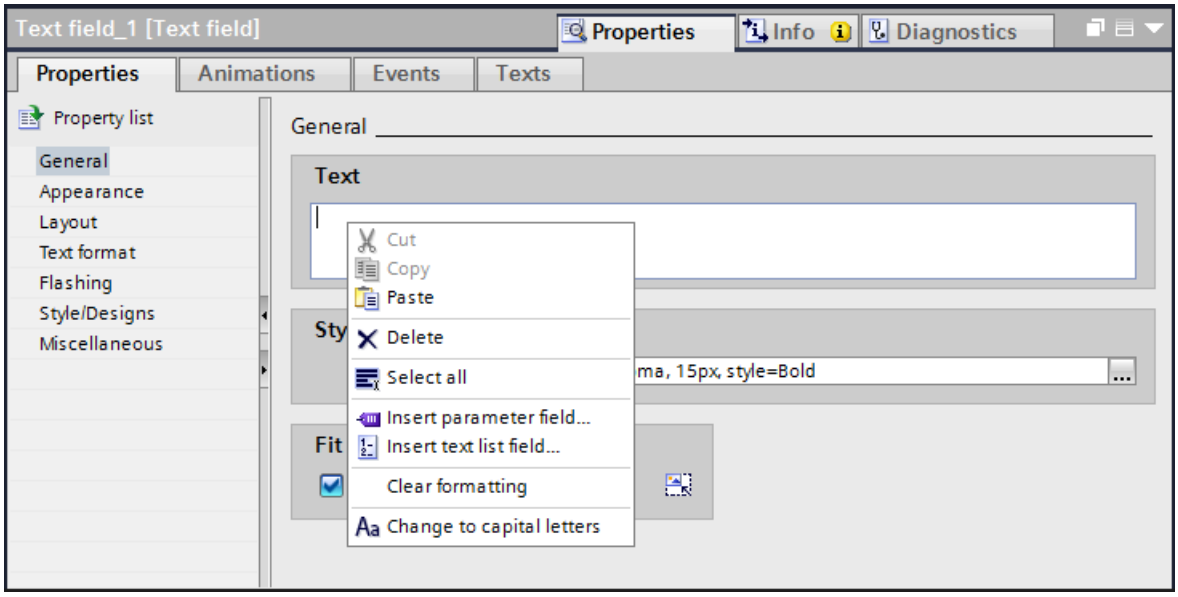

3. Select the command "Insert parameter field ...".
   A dialog opens.
4. Select the tag and the format in which you want to display the tag.
5. Confirm your entries.

The name of the connected tag is displayed in the text field of the object.

Connecting the text list

1. Open "Properties > Properties > General" in the inspector window.
2. Open the shortcut menu of the input field.
3. Select the command "Insert text list field ...".
   A dialog opens.
4. Select the text list from which a text entry is displayed.
5. Confirm your entries.

The reference to the text list entry is displayed in the text field of the object.

Note

The number of references to text list entries that in turn include references to text list entries or tags is limited to three layers.

1.2.1.23 Formatting graphics in an object

Introduction

WinCC lets you insert graphics into some objects and format them. You can change the size, orientation and distances of a graphic to the object border. You change the properties of the graphic in the Inspector window of the relevant object.

Requirement

- The selected object is in graphic mode.
- The selected object contains at least one graphic.

Resizing a graphic

1. In the Inspector window, select "Properties > Properties > Layout".
2. Select "No stretching of picture" or "Stretch picture".
   The contained graphic is displayed in its original size or stretched to the size of the object.

Note

To adjust the object size to the contained graphic, select "Fit object to contents".

Aligning a graphic horizontally and vertically

1. In the Inspector window, select "Properties > Properties > Layout".
2. To determine the horizontal position of the graphic, select "Right", for example.
3. To determine the vertical position of the graphic, select "Top", for example.
   The graphic is then shown at the top right of the object.

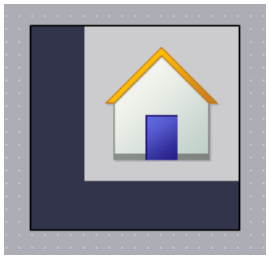

Defining the distance from the object border

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the value for the distance to the object border, for example, "20" for the top border.
   The graphic is shown with a distance of 20 pixels to the top border of the object.

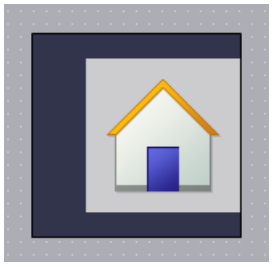

Icon	Description
	Sets the distance from the left border of the object.
	Sets the distance from the right border of the object.
	Sets the distance from the top border of the object.
	Sets the distance from the bottom border of the object.

1.2.1.24 Rotating an object in runtime (RT Professional)

Introduction

You define the rotation of an object around a reference point. You define the reference point with "X", "Y" in the Inspector window in the group "Properties > Properties > Layout".

The rotation of the object is visible in Runtime only.

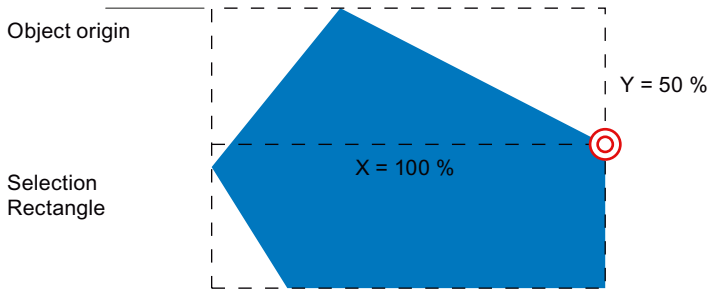

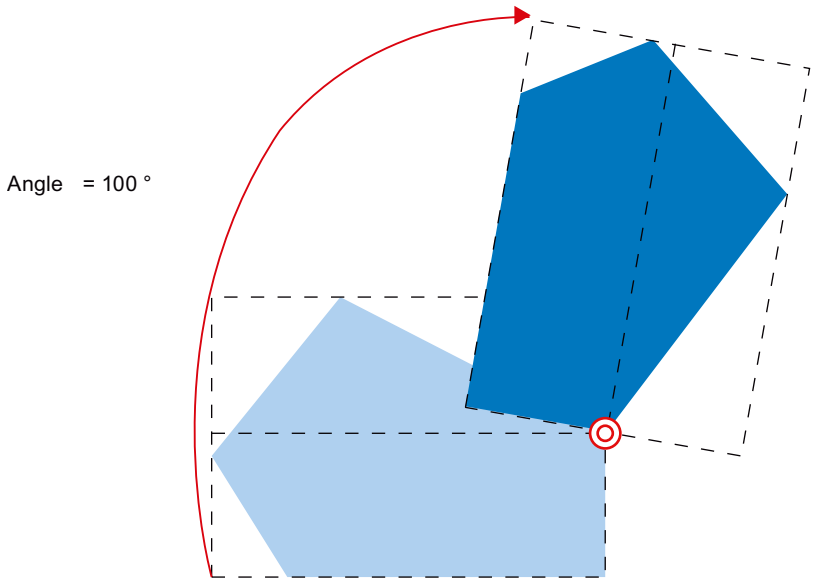

X, Y

The attributes "X" and "Y" define the horizontal and vertical distance of the reference point from the object origin.

The values are specified as a percentage. The object width or object height correspond to a value of 100%. The reference point value can be outside the selection rectangle. This means that both negative values and positive values higher than 100% are possible.

Angle

The angle defines the rotation of an object around the reference point. The angle is specified in degrees. The configured start point corresponds to a value of 0°. The position of an object deviates from its configured start position by the value of the angle.

See also

Polygon (Page 453)
Polyline (Page 454)
Overview of objects (Page 97)

1.2.1.25 Designing table-based objects

Introduction

WinCC offers various properties for designing display and operating objects.

You change the properties of table-based objects in the Inspector window of the relevant object.

Designing colors

Select the colors for font, selections and areas under "Properties > Properties > Appearance".

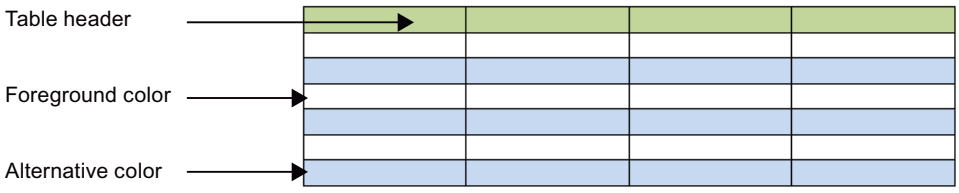

Figure 1-1 Selecting table properties

Designing the border of the table header

1. In the Inspector window, open "Properties > Properties > Table header border".
2. Select "Solid", for example, as the style.
3. Select 5, for example, as the "Width".
4. Select red, for example, as a foreground color.

The border of the table header is shown with a red border and width of 5 pixels. With "Solid" border style, only the foreground color is visible.

Colors and color gradient in the table header

1. Select the table-based object in the screen.
2. In the Inspector window, open "Properties > Properties > Table header fill pattern".
3. Select the fill pattern, for example "Horizontal gradient".
4. Select a background color, for example, blue, under "Gradient".
5. Activate "Gradient 1".

6. Select a color for "Gradient 1", for example, white.
7. Select a "Width" for the color gradient, for example, 12.
8. Activate Gradient 2.
9. Select a color for Gradient 2, for example, yellow.
10. Select a "Width" for the color gradient, for example, 10

The table header is displayed with a color gradient.

1.2.1.26 Designing a border

Introduction

WinCC offers various properties for designing display and operating objects.

The I/O field example below shows the possible settings for designing borders.

You change the appearance in the Inspector window of the relevant object.

Designing a border

1. Open "Properties > Properties > Appearance".
2. Select the width of the border, e.g. "5".
3. Select the style of the border, e.g. double line.
4. Select the foreground color.
5. Select the background color.

Figure 1-2 "Double line" border style

Border style

The figure below shows a border in "3D style".

The figure below shows a border in "Solid" style.

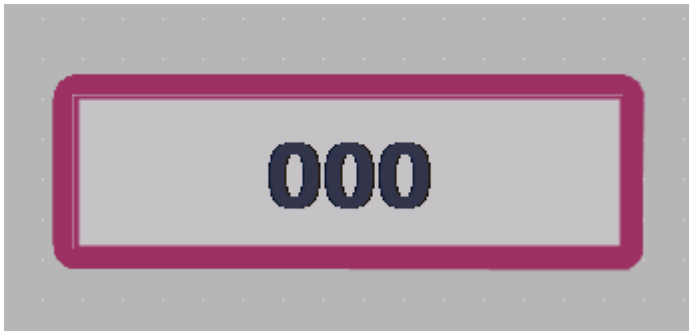

Note

Device-specific nature of the properties

The border style that is available depends on the object and the HMI device used.

1.2.1.27 Defining color gradients

Introduction

For the objects in WinCC, color gradients can be specified as background for various surfaces.

Change the category in the Inspector window, depending on which surface you fill with a color gradient. The procedure remains the same.

The following section describes how to configure the color gradient of buttons.

Horizontal color gradient with two colors

1. Select an object with buttons, for example, a button.
2. In the Inspector window, select "Properties > Properties > Fill Pattern".

3. Select "Fill Pattern > Horizontal gradient".
4. Select a background color for the vertical color gradient, for example, orange.
5. Activate "Gradient 1".
6. Select a "Color" for Gradient 1, for example, red.
7. Select a "Width" for Gradient 1, for example, "10".

The background of the button is displayed in orange.

For the horizontal color gradient, Gradient 1 is shown from the left border. Gradient 1 is 10 pixels wide.

Vertical color gradient with three colors

1. Select a button in the screen.
2. In the Inspector window, select "Properties > Properties > Fill Pattern".
3. Select the "Vertical gradient" background.
4. Select a background color under "Gradient", for example, orange.
5. Activate "Gradient 1".
6. Select a color for "Gradient 1", for example, red.
7. Select a "Width" for the color gradient, for example, 8.
8. Activate Gradient 2.
9. Select a color for Gradient 2, for example, yellow.
10. Select a "Width" for the color gradient, for example, 10.

	← Color Gradient 1
	← Background color
	← Color Gradient 2

For the vertical color gradient, Gradient 1 is shown from top to bottom. Gradient 1 is 8 pixels wide.

The background of the button is displayed in orange.

Gradient 2 is shown at the bottom border. Gradient 2 is 10 pixels wide.

1.2.1.28 Configuring ranges

Introduction

As of WinCC V14, you have the option of defining up to five value ranges for the "Gauge", "Bar" and "Slider" objects and configuring them centrally in a process tag. This means you can, for example, distinguish between different operating states with different colors. To display the ranges defined in the tag in the screen object, activate the option "Display ranges of tags" in the Inspector window under "Properties > Limits/Ranges".

This setting is not available for display with inner scale, for example, the option "Layout with inner bar" must be cleared for the bar.

You define the following five ranges:

Limit/range	Use
Upper 2	Displays the high danger range.
Upper 1	Displays the high warning range.
Normal	Displays the normal range.
Lower 1	Displays the low warning range.
Lower 2	Displays the low danger range.

Limits/ranges

You specify limits for numerical values to restrict the value range. This range is defined by a high limit and a low limit. You specify the use and color of ranges as well as the maximum and minimum values at the screen object.

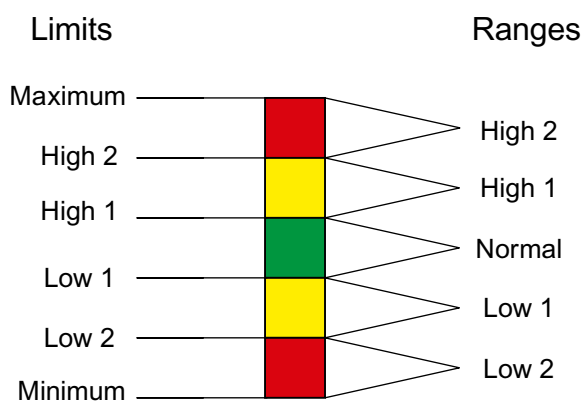

You can find additional information on limits under "Defining limits for a tag (Page 548)".

Requirements

- You have created at least one tag.
- You have defined four ranges at the tag.
- You have created the screen object.

Configuring ranges

1. Specify the tag that defines the ranges for the screen object under "Properties > General > Process tag".
2. If necessary, specify the values for maximum and minimum.
3. Select the option "Display ranges of tags" under "Properties > Limits/Ranges".

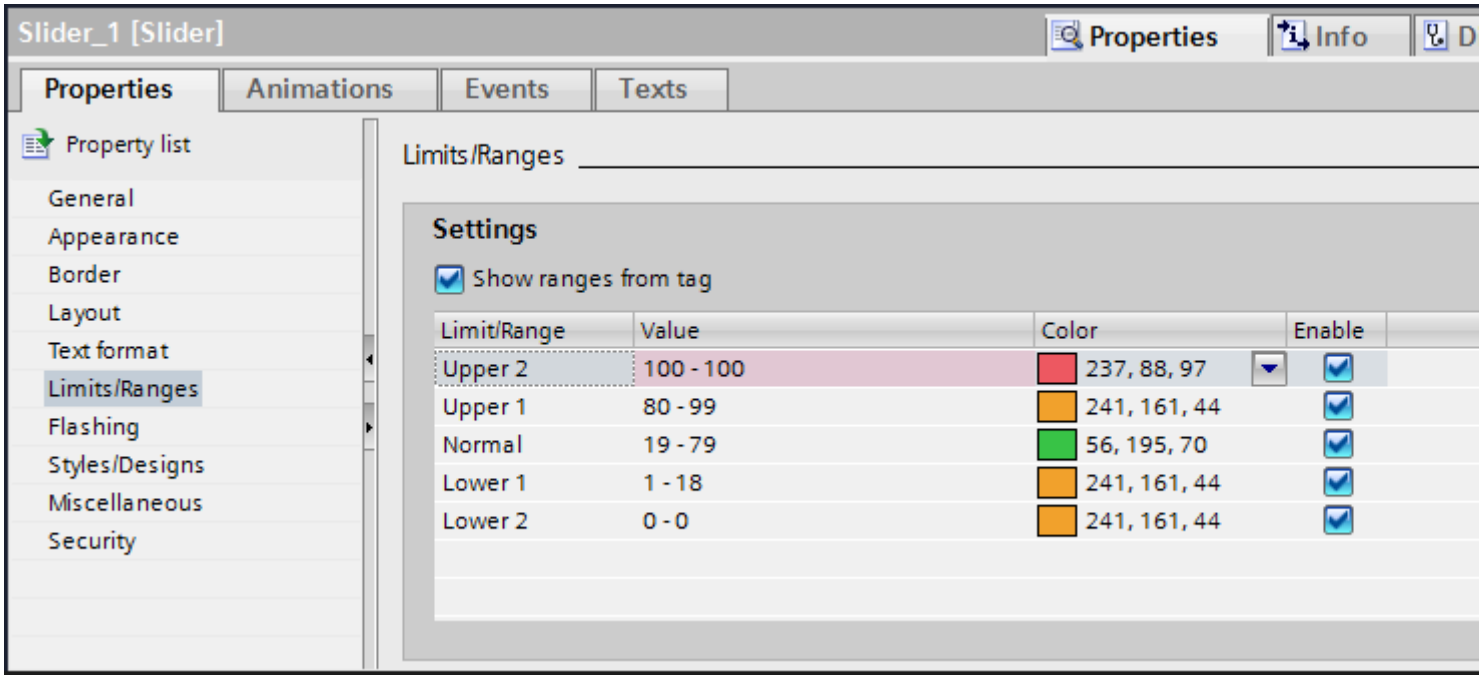

4. Specify which ranges are displayed in the object.
5. You can change the default colors for the ranges, if required.

Result

You have configured ranges in a screen object. The ranges are displayed in different configured colors in Runtime.

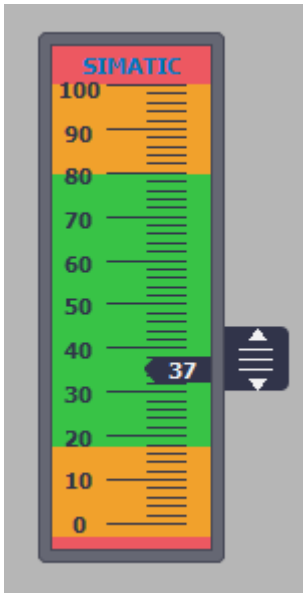

See also

Defining Limits for a Tag (Page 548)

1.2.1.29 Flashing (RT Advanced, Panels, Comfort Panels)

Introduction

You want an object to flash in Runtime.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to flash.
2. In the Inspector window, select the type "Standard enabled" under "Properties > Properties > Flashing".

Result

The object flashes in runtime.

See also

Overview of objects (Page 97)

1.2.1.30 Flashing (RT Professional)

Introduction

You display objects flashing in runtime. You select the flash rate and colors for the object.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to flash.
2. In the Inspector window, select the type "Advanced" under ""Properties > Properties > Flashing".
3. Activate "Line" to make the line flash in runtime.
4. Activate "Text" to make the text flash in runtime.
5. Activate "Background" to make the background flash in runtime.
6. Select the flash rate.
7. Select a color for the "OFF" state.
8. Select a color for the "ON" state.

Note

Flashing is only visible in runtime when there is a difference between the two colors.

Result

In runtime, the object flashes in the selected colors and flash rate.

See also

Overview of objects (Page 97)

1.2.1.31 Inserting multiple objects of the same type (stamping)

Introduction

WinCC offers the possibility to "stamp" several objects of the same type directly one after the other, i.e. paste without having to reselect the object every time. In addition you have the possibility of multiplying an object that has already been inserted.

Requirement

The "Tools" task card is open.

Inserting several objects of one type

1. Select the object that you want to insert in the "Tools" task card.
2. Click the icon in the toolbar of the "Tools" task card.
   The "Stamp" function is activated.
3. To insert the object with its standard size, click the relevant insertion position in the work area.
   To insert the object with another size, position the mouse pointer at the desired location in the work area. Press the left mouse button and drag the object to the required size.
   The object is inserted in the work area as soon as you release the mouse button.
4. Repeat step 3 to insert further objects of the same type.
5. Click the icon again.
   The "Stamp" function is deactivated.

Note

You can copy existing objects using the drag-and-drop +<CTRL> function. The existing object is not moved in this case. You paste a copy of this object into the new position instead.

Inserting and multiplying an object

1. Insert the desired object from the "Tools" task card.
2. Press the <Ctrl> key and position the cursor on one of the handles displayed in the figure shown below.

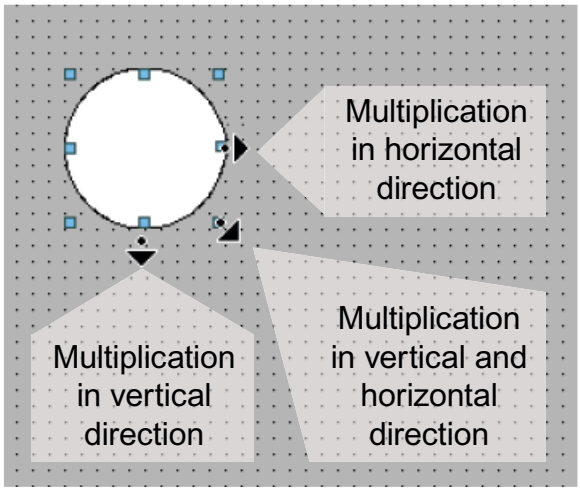

3. Drag the handles to the right and/or down while keeping the left mouse button pressed.
4. The object is multiplied depending on available space if you keep moving the cursor.

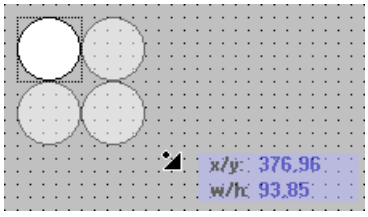

Result

You have pasted and stamped an object in a screen.

See also

Overview of objects (Page 97)

1.2.1.32 Repositioning and resizing multiple objects

Possible modifications

After you have selected multiple objects, you edit them:

- Shift using the mouse
  - To change the absolute position of the marked objects, position the mouse pointer over an object, and shift the multiple selection with the mouse button pressed.
  - To resize all the objects by the same ratio, grab the resizing handles of the reference object.
- Move over the work area with the icons of the toolbar
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects
- Moving with the shortcut menu commands of the work area
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects

See also

Overview of objects (Page 97)

1.2.1.33 Hiding or showing objects (RT Professional)

Introduction

You can hide and show objects from or up to a defined size. This means, for example, that objects are only shown if a sufficient size allows the required detail to be displayed.

Requirement

A screen is open.

Procedure

Proceed as follows to hide specific objects in a screen:

1. Click in an area of the screen that does not contain an object.
2. Activate the "Activate for objects" check box under "Hide/Show" in the Inspector window.
3. Specify the minimum and maximum percentage within which the objects in question should be displayed. The specified limit values are also still shown.

Result

Those objects are shown whose size lies within the specified range when the object layer is visible. The object continues to be displayed as long as either the height or the width lies within the specified range.

See also

Overview of objects (Page 97)

1.2.1.34 Linking user data types with objects (RT Professional)

Introduction

If you work with user data types, only create a user type once and then use it multiple times for objects, animations, feature lists and scripts. If, for example, a component is used more than once in your system, you can link the component with a suitable user type that contains structured subelements for each lower-level component.

You assign a user data type defined in the project library to a screen and link the elements of this user data type to objects. The elements of a user data type can be used for screens and screen objects instead of HMI tags or PLC tags. You use them for animations, function lists and scripts, for example.

Procedure

1. Select the "Use user data type" check box under "Properties > Properties > User data type" in the Inspector window.
2. In the "User data type" selection list, select a HMI user data type or PLC user data type from the project library that you want to assign to the screen.

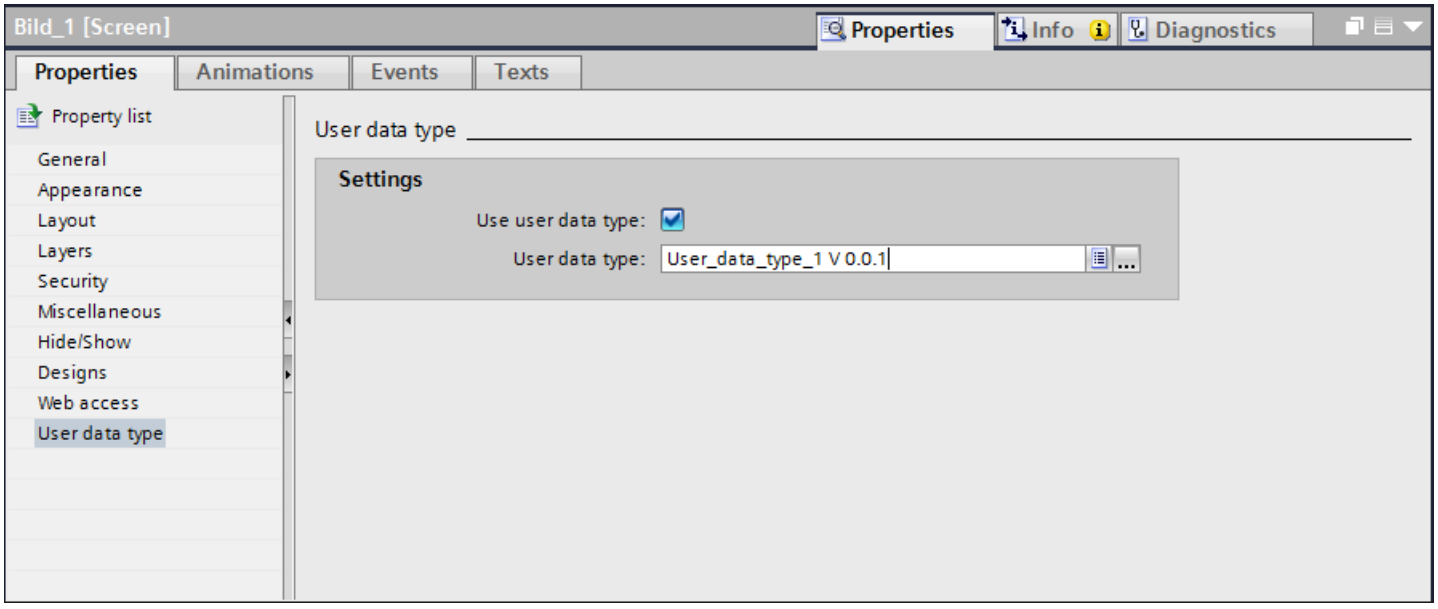

The user data type is assigned to the screen.

Linking elements of a user data type with objects

1. Insert in the screen an object that you want to link to an element of a user data type.
2. In the Inspector window under "Properties > Properties > General", select the element from the "Tag" selection list that you want to link to the object.

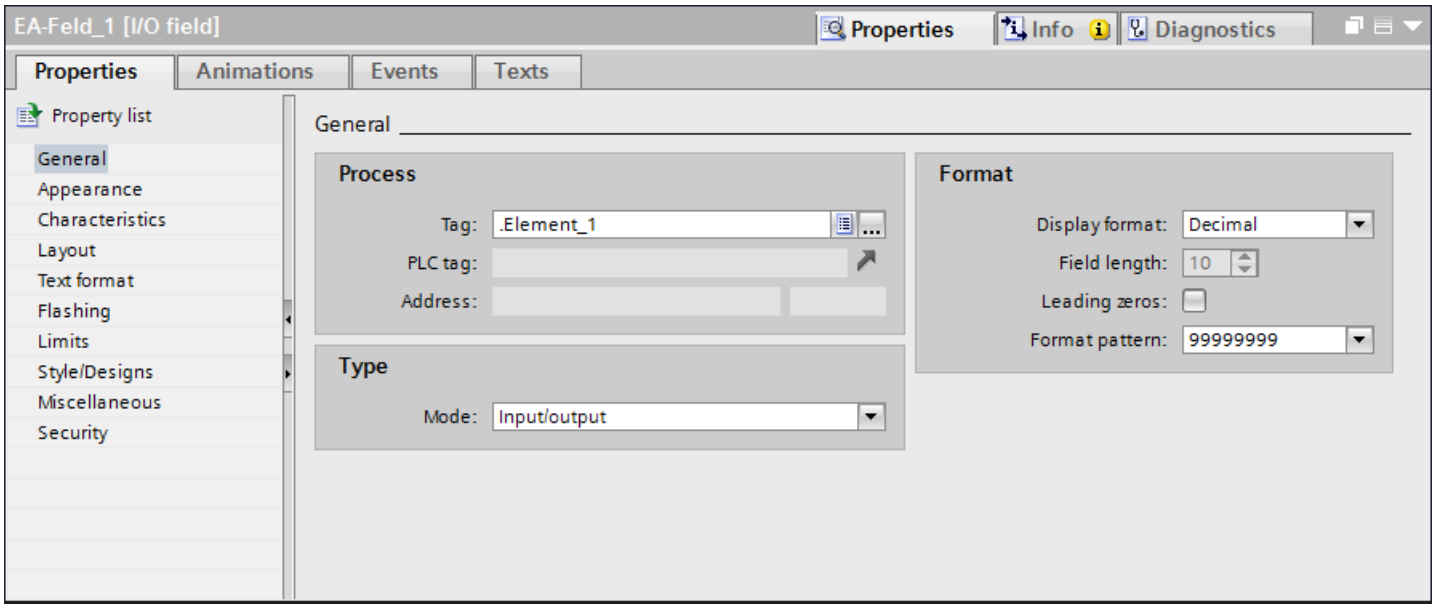

Alternatively, you can link the element using autocomplete.

Note

You can recognize by the point prefixed to the element name that an element of a user data type in the "Tag" selection list has been used.

The "@NOP" prefix indicates that an HMI tag or a PLC tag in the "Tag" selection list was used.

Comments

- If you disable the "Use user data type" check box under "Properties > Properties > User data type", the link between the element and the object is removed.
- If you assign the screen a different user data type, the link between the element and the object is removed. However, if an identically named element is contained in the new user data type on the same structural level, the link between the element and the object is retained.
- If you rename the element, the link between the element and the object is removed.
- If you change the data type of the element, the link between the element and the object is retained.

See also

Configuring a faceplate type (Page 272)

Working with user data types (Page 585)

1.2.1.35 Managing own controls (RT Professional, RT Advanced)

Introduction

You can use your own controls in the WinCC editions "WinCC Advanced" and "WinCC Professional". Add standardized or user-defined controls in the "Toolbox > My Controls" task card. All the ActiveX controls registered in the operating system and the .Net controls on your system are available for use in WinCC.

- ActiveX controls
  ActiveX controls are operator controls from any providers, which can be used by other programs over a defined OLE-based interface.
- .Net controls
  .Net controls are operator controls from any providers based on the Microsoft .Net Framework.
- Custom .Net controls
  .Net controls are operator controls from any providers based on the Microsoft .Net Framework.

Note

Only the simple .Net controls are supported. XAML controls, for example, WPF (Windows Presentation Foundation) controls, are not supported in the TIA Portal.

If the name of an event of a control contains the character string "Changed", this event is not displayed in WinCC.

If you use any controls in your project that are not included in the product package, you must

- check whether these can be run on the HMI device.
- first add these to the "My controls" palette before you copy the project to different PCs.

Note

If you use any controls that are not included in the product package (for example, from third parties), errors may occur, for example, reduced performance or system crashes.

The user of such control software is alone responsible for any problems caused by its use. We therefore advise you to closely examine such elements prior to their use.

Requirements

- The required ActiveX control is installed in the PC's operating system.
- The desired .Net control is available on the PC in an assembly.
- A screen is open.
- The "Toolbox" task card is open.

Adding a new control to the "My controls" group

1. Open "Toolbox > My controls" in the "Toolbox" task card.
2. Select "Select objects" from the shortcut menu.
   A dialog opens. This contains all the controls available on the system. Controls that already exist in the "Toolbox" task card are identified by a check mark.

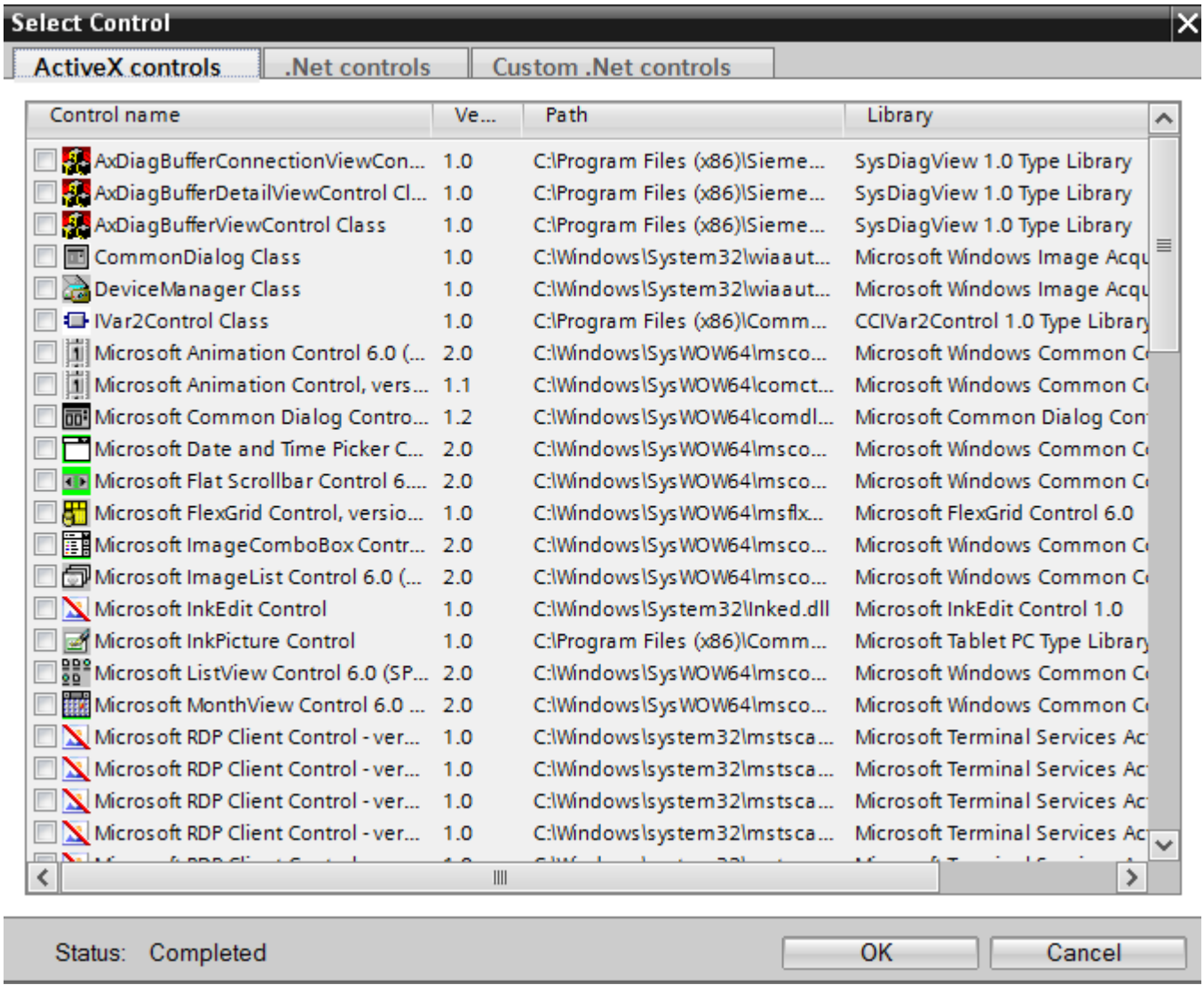

3. Select the desired tab, e.g. ActiveX controls.

4. Activate the required control.
5. Confirm your selection with "OK".

Inserting a custom .Net control in the "My controls" group

1. Select the "Custom .Net controls" tab. The list is empty the first time you select the tab.
2. Click the "…" button.
3. Select the folder in which the assembly is contained.
4. Select a file.
5. Click "Open." All the available controls are displayed in the "Custom .Net controls" tab.
6. Activate the required .Net control.
7. Confirm your selection with "OK".

Note

.Net controls in Runtime Advanced

If you have incorporated a .Net control in your project as "Custom .Net control", you have to copy the files belonging to these controls to the installation directory of WinCC Runtime, e.g. "C:\ProgramFiles\Siemens\Automation\WinCC RT Advanced". Otherwise, the control cannot be loaded in Runtime.

Result

The control now appears in the "Toolbox > My controls" task card and can be configured in a screen.

Removing a control from the "My controls" group

1. Select "Delete" from the shortcut menu for the control concerned.
   The control is removed from the "My controls" group.

Synchronization of the ActiveX controls

Changes to properties of ActiveX controls are only saved if the control synchronizes the OCXState property correctly in the case of changes.

For controls such as "Microsoft Date and Time Picker Control" that do not synchronize the OCXState property correctly, the changes are discarded after the screen is closed, during compilation or after the migration.

See also

Overview of objects (Page 97)

1.2.1.36 External graphics

Introduction

You can use graphics created with an external graphic program in WinCC. To use these graphics you store them in the graphic browser of the WinCC project.

You can save graphics in the graphic browser:

- When you drag-and-drop graphics objects from the "Graphics" pane into the work area, these are stored automatically in the graphic browser. The graphic names are numbered in the order of their creation, for example, "Graphic_1." Use the <F2> function key to rename the graphic.
- As a graphic file with the following formats:
  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tif, *.png, *.svg, *.jpeg or *.jpg
- As an OLE object that is embedded in WinCC and is linked to an external graphic editor. In the case of an OLE link, you open the external graphic editor from WinCC. The linked object is edited using the graphic editor. An OLE link only works if the external graphic editor is installed on your PC, and supports OLE.

Use of graphics from the graphic browser

Graphics from the graphic browser are used in your screens:

- In a Graphic view
- In a graphics list
- As labeling for a button/function key

Transparent graphics

In WinCC you also use graphics with a transparent background. When a graphic with a transparent background is inserted into a graphic object of WinCC, the transparency is replaced by the background color specified for the graphic object. The selected background color is linked firmly with the graphic. If you use the graphic in another graphic object of WinCC, this object is displayed with the same background color as the graphic object that was configured first. If you want to use the graphic with different background colors, include this graphic in the graphic browser again under a different name. The additional background color is configured when the graphic is used at the corresponding graphic object of WinCC.

Note

Transparent graphics in WinCC Professional

Transparent graphics and graphics with the format *.svg are not supported in WinCC Classic Design.

Managing graphics

An extensive collection of graphics, icons and symbols is installed with WinCC, for example:

In the Toolbox window of the "Graphic" pane the graphic objects are structured by topic in the "WinCC graphics folder." The link to the WinCC graphics folder cannot be removed, edited or renamed.

The "Graphics" pane is also used to manage the external graphics. The following possibilities are available:

- Creating links to graphics folders
  The external graphic objects in this folder, and in the subfolders, are displayed in the toolbox and are thus integrated in the project.
- Editing folder links
- You open the program required for editing of the external graphic in WinCC.

See also

Overview of objects (Page 97)

1.2.1.37 Managing external graphics

Introduction

External graphics that you want to use in WinCC are managed in the "Screens" editor by using the "Tools" task card in the "Graphics" pane.

Requirement

- The "Screens" editor is open.
- The "Tools" task card is open.
- The graphics are available.
- The graphics have the following formats:
  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tif, *.svg, *.jpeg or *.jpg

Creating a folder link

1. Click "My graphics folder."
2. Select "Link" in the shortcut menu.
   The "Create link to folder" dialog is opened. The dialog suggests a name for the folder link.
3. Edit the name as required. Select the path containing the graphic objects.
4. Click "OK" to confirm your input.
   The new folder link is added to the "Graphics" object group. The external graphics that are located in the target folder and in sub-folders are displayed in the toolbox.

Editing folder links

1. Select the folder link to edit.
2. Select the "Edit link..." command from the shortcut menu.
   The "Create link to folder" dialog is opened.
3. Edit the name and path of the folder link as required.
4. Click "OK" to confirm your input.

Renaming the folder link

1. Select the folder link to rename.
2. Select "Rename" from the shortcut menu.
3. Assign a name to the new folder link.

Removing a folder link

1. Select the folder link you want to delete.
2. Select "Remove" in the shortcut menu.

Edit external graphics

1. Select the graphic you want to edit.
2. Select the "Edit graphic" command from the shortcut menu.
   This opens the screen editor associated with the graphic object file.

Editing graphics folders from WinCC

1. Select the graphic you want to edit.
2. Select the "Open parent folder" command from the shortcut menu.
   The Windows Explorer opens.

See also

Overview of objects (Page 97)

1.2.1.38 Storing an external image in the graphics library (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

To display graphics that have been created in an external graphics program in your screens, you will first have to store these graphics in the graphics browser of the WinCC project.

Requirement

- A screen has been created.
- A Graphic View is inserted in the screen.
- The Inspector window of the Graphic view is open.

To store an external graphic in the image browser:

- A graphic is available.

To store an OLE object in the browser:

- An OLE-compatible graphics program is installed on your configuration computer.

Save graphics file

1. Open the Windows Explorer.
2. Select the graphic that you want to store.
3. Drag-and-drop the graphic into the graphic browser.

Creating and saving a new graphic as an OLE object

1. Select the Graphic view on your screen.
2. In the Inspector window, select "Properties > Properties > General":
3. Open the graphic selection list.
4. Click .
5. The "Insert object" dialog box opens.

   Note

   In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

6. From the "Insert object" dialog box, select "New" and an object type. The settings in "Settings > "OLE settings" determine which object types are shown.
7. Click "OK." The associated graphic program is opened.
   When you are finished creating graphics, end the graphic programming software with "File > Close" or "File > Close & return to WinCC."
   The graphic will be stored in the graphic programming software standard format and added to the graphic browser.

Inserting created graphics in WinCC

Note

A new graphic object created as OLE object may not be directly accepted in WinCC when you save it to an external graphics program.

1. Reopen the dialog for inserting a graphic.
2. From the "Insert object" dialog box, select "Create from file."
3. Click the "Browse" button.
4. Navigate to the created graphic and select it.

Saving an existing graphic object as an OLE object

1. In the Inspector window, select "Properties > Properties > General":
2. Open the graphic selection list.
3. Click .
4. The "Insert object" dialog box opens.

   Note

   In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

5. From the "Insert object" dialog box, select "Create from file."
6. Click the "Browse" button.
7. Use the dialog to help you navigate to the folder in which the graphic file is saved.

   Note

   To import graphics files, note the following size restrictions:

   *.bmp, *.tif, *.emf, *.wmf ≤4 MB

   *.jpg, *.jpeg, *.ico, *.gif "*≤1 MB

Result

The image file is now stored in your image browser. It is shown in a screen with a Graphic view , or is added as a list element in an image list.

You can double-click OLE objects in your library to open them for editing in the corresponding graphic editor. When you have finished editing graphics, end the graphic programming software with "File > Close" or "File > Close & return to WinCC." The changes are applied to WinCC.

See also

Overview of objects (Page 93)

1.2.1.39 Individually adjusting graphics in controls (RT Professional)

Introduction

In the standard configuration, neither the graphics for control buttons nor the size of those graphics can be changed. However, WinCC Professional provides the option to customize the appearance of controls to your own requirements. For example, you can enlarge the buttons of the controls for touch operation.

Controls are customized by creating and editing the graphics that are used by the software. To show a change made during runtime, select the new style from the configuration dialog of the control.

You have the following options for the customization of controls:

- Change the size and design of the buttons
- Adjust symbols in the table elements
- Changing the design of the scroll bar

Further information, a sample project and sample graphics are available on the Internet under Individualizing WinCC Runtime Professional Controls. (http://support.automation.siemens.com/WW/view/en/76327375)

Basics

It is possible to individualize the following controls:

- Recipe view
- Alarm view
- f(x) trend view
- f(t) trend view
- Table view
- Value table

After the installation of TIA Portal, the "CCAxControlSkins" folder is created under "C:\Program Files(x86)\Common Files\Siemens\bin\" for the design of the controls.

To use modified designs, you need to create subfolders within the "CCAxControlSkins" folder. The number and the name of the subfolder are determined by the elements in the controls you want to customize.

The design of a control can then be selected as a "Style" property in the "Style" selection list on the "General" page of the control's configuration dialog.

To make the symbols created for table elements of a control visible, the "Content as symbol" option must be selected for the corresponding column in the "Appearance" area.

Note

When creating a new design, you do not need to create all the files. The default settings of the controls are used for all files that are not available.

Settings in the engineering system

Some adjustments in the engineering system are required to enable you to customize controls.

1. Select the control that you want to customize in the screen.
   The properties of the control are displayed in the Inspector window.
2. Under "Properties > Properties > Columns", select one of the entries in the "Title as" column for the display of the respective column title:
   - "Both" (text and symbol)
   - "None"
   - "Symbol"
   - "Text"

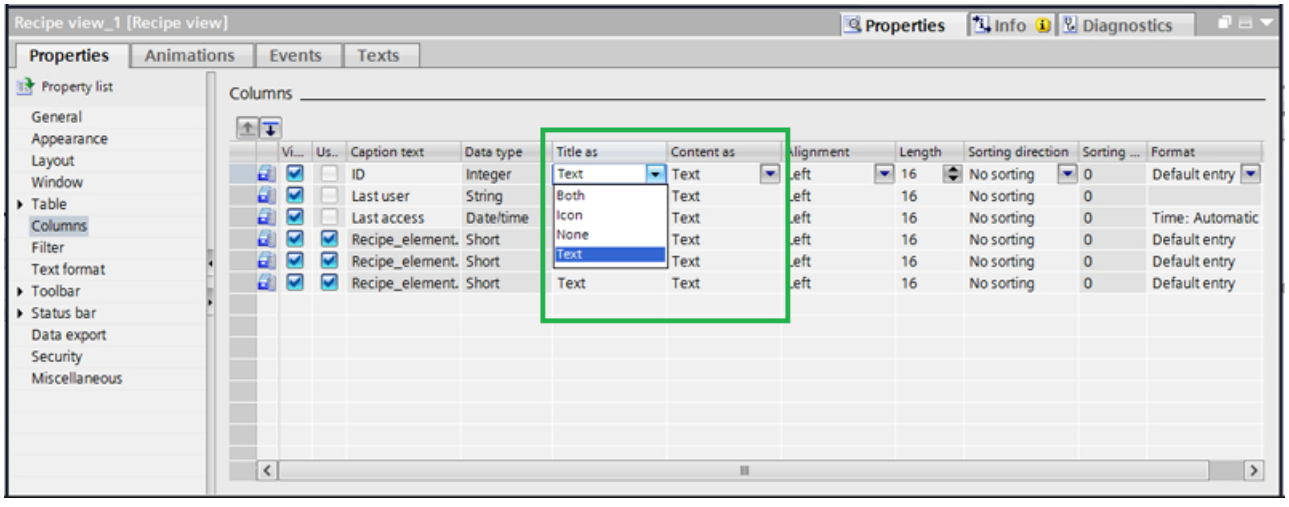

Select the desired style you need to configure in Runtime.
To perform the customization directly in the engineering system, use dynamics properties with tags or local C and VB scripts.

Creating a folder structure for your own designs

To enable new graphics to be used, a folder structure must be generated locally following specific rules.

1. Under "C:\Program Files(x86)\Common Files\Siemens\bin\CCAxControlsSkins", create a new folder from which graphics can be loaded to the control, e.g. "NewControlStyle".
2. In the new folder, generate a subfolder with the name of the control whose appearance you want to change, e.g. "AlarmControl" for the alarm view.

3. Create additional subfolders in the new folder:
   - "Toolbar" folder for the toolbar button graphics
   - "GridIcons" folder for graphics in the table cells

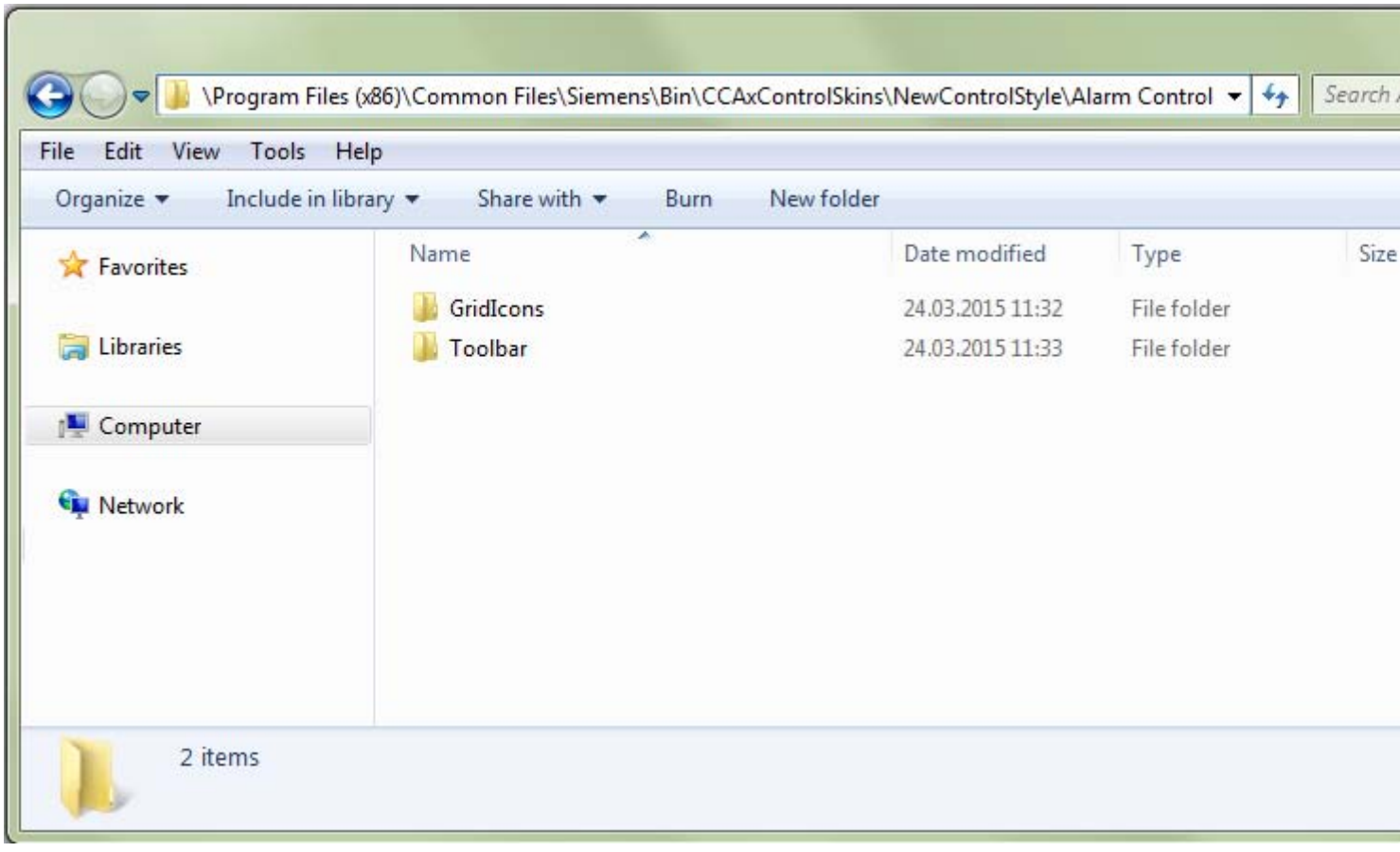

4. If you want to change the scroll bar display, create a "Scrollbar" folder for the scroll bar graphics under "C:\Program Files(x86)\Common Files\Siemens\bin\CCAxControlsSkins".

Note

Only those graphics that are stored in the folders are used for changes. The default settings of the controls are used for files that are not located in the folders.

Note

Please make sure that the file and folder names are spelled correctly so that the software can access the graphics.
Permissible image formats are PNG, GIF, JPG, and ICO.

Note

Please note, in particular when using multi-user systems or the "WebNavigator" option, that the "CCAxControlSkins" folder must be stored locally, i.e. on the WinCC Client.

Adjusting buttons on the toolbar

1. In the "Toolbar" folder of the relevant control, create an "IconsNormal.png" file.
2. Using image processing software, insert individual button graphics in the file one after another.
3. To use new graphics, edit the "IconsNormal.png" and "IconsDisabled.png" files.
4. To change the size of the toolbar graphics, enlarge or reduce the "IconsBackground.png" graphic.
   This file defines the size of all toolbar buttons together.
5. To adjust the space around the hyphen, save the "Space.png" file in the "Toolbar" folder.
6. Generate the file "Left.png", which is used to define the width of the margin to the left of the button.
7. To show a change during runtime, select the newly generated style from the configuration dialog of the control.

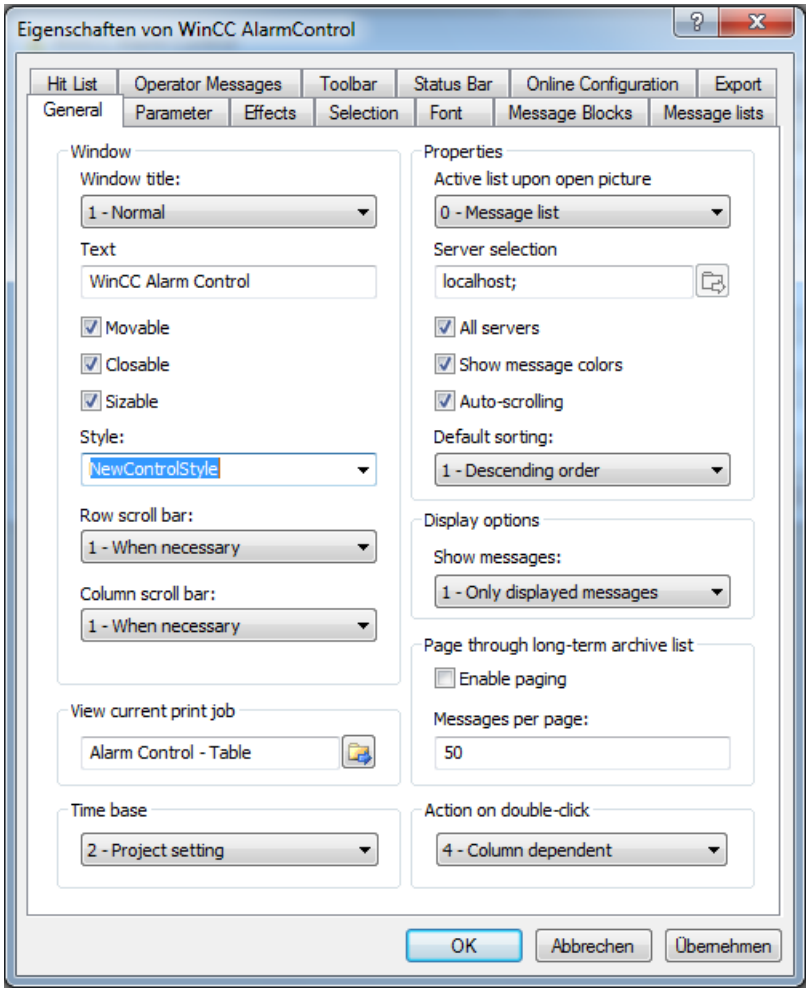

Note

The "Skin" parameter can be set only via the dynamics of the "Style" property with VB scripting, for example, HMIRuntime.Screens("Alarm").ScreenItems("Alarm view_1").SkinName = "MyOwnSkin".

Adjusting table elements

Proceed as follows to show different symbols depending on values in the table cells:

1. In the "..\[ControlName]\GridIcons" folder, generate the subfolders for each column in the table in which the graphics are displayed.
   Rename the subfolder to the name of the object property and save the state name in English, for example, "State" for the "Status" column.

   Note

   No symbols can be displayed for the "Date" or "Time" columns.

2. In the "State" folder, you need to save the graphics with the respective state names in English, for example, "ComeQuit".
   For the state for which you have saved a graphic, the new symbol appears in the table cell when the state occurs.
3. To assign symbols for message numbers, assign a graphic for each numerical value.
   If you want to specify symbols for specific intervals, define the range in the file name with an underscore, e.g. "50_100.png" for the interval between 50 to 100. The limits are contained in the interval.
4. To display only symbols instead of text for an alarm text block/column, you must specify a graphic file for each occurring text.
5. Configure the display options for the content during runtime in the configuration dialog on the "Alarm blocks" tab.

Adjusting the scroll bar

The scroll bar display consists of the various different graphics, and is created from the individual image files during runtime. Follow these steps to adjust the scroll bar:

1. Create the "Scrollbar" folder for the scroll bar graphics under "C:\Program Files(x86)\Common Files\Siemens\bin\CCAxControlsSkins".
2. Create the subfolders "horizontal" and "vertical" in the "Scrollbar" folder.
3. In this folder you place the individual graphics of the scroll bar, from which the scroll bar is assembled at runtime.
4. Select the style you have generated from the configuration dialog of the control during runtime.
   The scroll bars will be displayed in the new style.

Result

If you select the new style for the control during runtime, the updated graphics are displayed in the control.

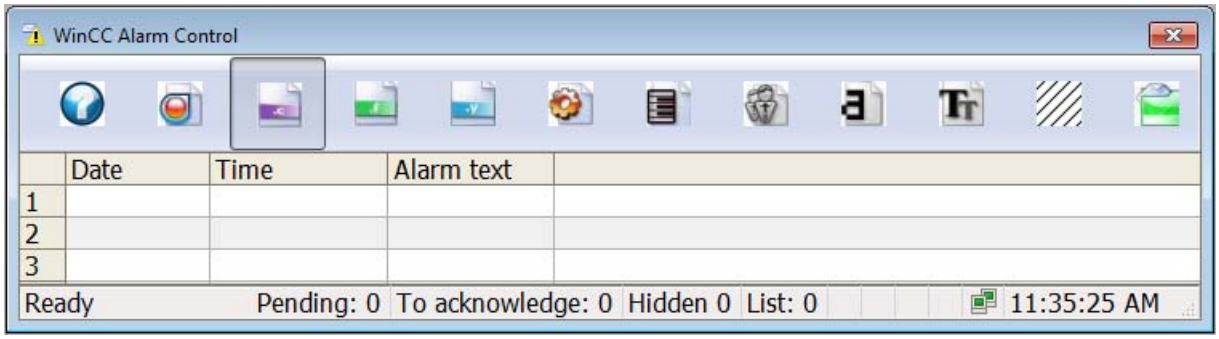

See also

Dynamization in the inspector window (Page 208)

1.2.2 Working with object groups

1.2.2.1 Basics on groups (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Groups are several objects that are grouped together with the "Group" function. You edit a group in the same way as any other object.

Overview

WinCC offers the following methods for editing multiple objects:

- Multiple selection (Page 111)
- Grouping objects (Page 158)

Editing mode

To edit an individual object in a group, select the object in the "Layout > Layers" task card.

Alternatively select "Group > Edit group" in the object group's shortcut menu.

Hierarchical groups

You add further objects or groups to extend a group. The group is enlarged by the new objects and is structured hierarchically in main and sub-groups. Such hierarchical groups must be broken up in stages. You also break up the group in the same order in which you grouped the objects or groups. It takes exactly the same number of steps to break up these hierarchical groups as it did to create them.

Rectangle surrounding the object

Only one surrounding rectangle is now displayed for the whole group. The surrounding rectangles of all objects are displayed for a multiple selection on the other hand.

Layers

All objects of a group are located in the same layer.

Properties of a group

A group has its own properties, such as authorization. If you set a property at a group, all objects of the group inherit this setting. If the same property is configured differently at an object in the group, the value of the property at the object and not the value in the group applies in Runtime.

See also

Ungrouping (Page 160)

Adding objects to the group (Page 161)

Removing objects from the group (Page 162)

Editing an object in a group (Page 163)

Overview of objects (Page 97)

Example: Inserting and configuring a rectangle (Page 172)

1.2.2.2 Grouping objects (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The "Group" command combines multiple objects to form a group.

You can change the size and position of the group. The following rules apply:

- The system automatically adapts the position coordinates of the grouped objects when you reposition the group. The relative position of the grouped objects to the group is not affected.
- The system automatically adapts the height and width of the grouped objects in proportion to a change of the group size.
- To change the size of the group proportionately, hold down the <Shift> key and drag the rectangle around the object until has the required size.

Note

To create a hierarchical group, organize the individual groups like objects.

Requirement

- You have opened a screen which contains at least two objects.

Creating object groups

1. Select all the objects you want to organize in a group.
2. Select the command "Group > Group" from the shortcut menu.

The objects of the group are displayed with a rectangle around the objects.

Grouping objects within a group

1. Select the group you want to edit.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group that you are editing is highlighted by a red frame.
3. Select the objects of a group that you want to combine into a subgroup.
4. Select the command "Group > Group" from the shortcut menu.

A subgroup with the objects is created.

Adding objects into an existing group

1. Select the group to which you want to add objects.
2. Press the <Shift> key and select the object you want to add to the group.
3. Select the "Group > Add to group" command from the shortcut menu.

The object is added to this group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

Result

The selected objects are combined in a group. The multiple selection rectangle becomes the rectangle surroundings the objects in the group. The handles are shown only for the group. The group in in the active layer.

See also

Basics on groups (Page 157)

Ungrouping (Page 160)

Adding objects to the group (Page 161)

Removing objects from the group (Page 162)

Editing an object in a group (Page 163)

Example: Inserting and configuring a rectangle (Page 172)

1.2.2.3 Ungrouping (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Select the "Ungroup" command to split a group into the original object fractions.

Requirement

- You have opened a screen that contains a group.

Ungroup

1. Select the group.
2. Select the "Group > Ungroup" command from the shortcut menu.

Ungrouping a group within a group

1. Select the higher-level group.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group that you are editing is highlighted by a red frame.
3. Select the lower-level group.
4. Select the "Group > Ungroup" command from the shortcut menu.

Result

The lower-level group is ungrouped. The objects are assigned to the next higher group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

See also

Basics on groups (Page 157)

Grouping objects (Page 158)

Adding objects to the group (Page 161)

Removing objects from the group (Page 162)

Editing an object in a group (Page 163)

Example: Inserting and configuring a rectangle (Page 172)

1.2.2.4 Adding objects to the group (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The "Add objects to group" command is used to add objects to a group, without ungrouping it first.

Requirements

A screen with one group and at least one other object is opened.

Procedure

1. Select the group.
2. Press the <Shift> key and select the object you want to add to the group.
3. Select the "Group > Add to group" command from the shortcut menu.

Result

The group consists of the original objects, and the newly-added objects. The added objects are arranged at the front of the group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag&drop you can also easily edit hierarchical groups in the "Layers" palette.

See also

Basics on groups (Page 157)

Example: Inserting and configuring a rectangle (Page 172)

1.2.2.5 Removing objects from the group (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You use the "Remove objects from group" command to remove individual objects from a group, without ungrouping it first.

You do not have to remove the object from the group to edit an object in a group. You can edit the objects of a group individually.

Requirement

- You have opened a screen that contains a group.

Removing objects from a group

To remove an object from a group:

1. Select the group.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group you want to edit is highlighted by a red frame.
3. Select all objects in the group that you want to remove from the group.
4. Select the "Group > Remove from group" command from the shortcut menu.

The objects are removed from the group.

Note

If there are only two objects in the group, the menu command "Remove from group" is not available.

Deleting objects from a group

To remove an object from the group, and from the screen:

1. Select the group.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group you want to edit is highlighted by a red frame.

3. Select the objects in the group that you want to delete.
4. Select "Delete" from the shortcut menu.

Note

If there are only two objects in the group, the menu command "Delete" is not available.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

See also

Basics on groups (Page 157)

Example: Inserting and configuring a rectangle (Page 172)

1.2.2.6 Editing an object in a group (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can edit the objects of a group individually.

Requirement

You have opened a screen that contains a group.

Editing grouped objects

1. Select the group.

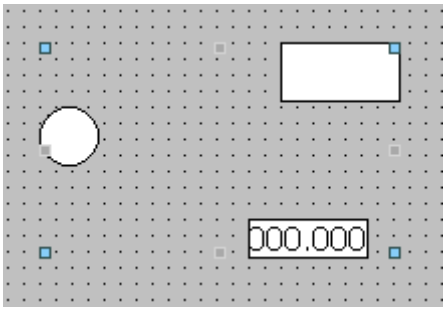

The properties of the group are displayed in the Inspector window.

2. Change the position and size of the grouped objects in "Properties > Properties >Layout."
3. Change the name of the group in "Properties > Properties > Miscellaneous."

Modifying the properties of an object within a group

1. Select the group.
2. Select the object whose properties you want to change in the Inspector window.

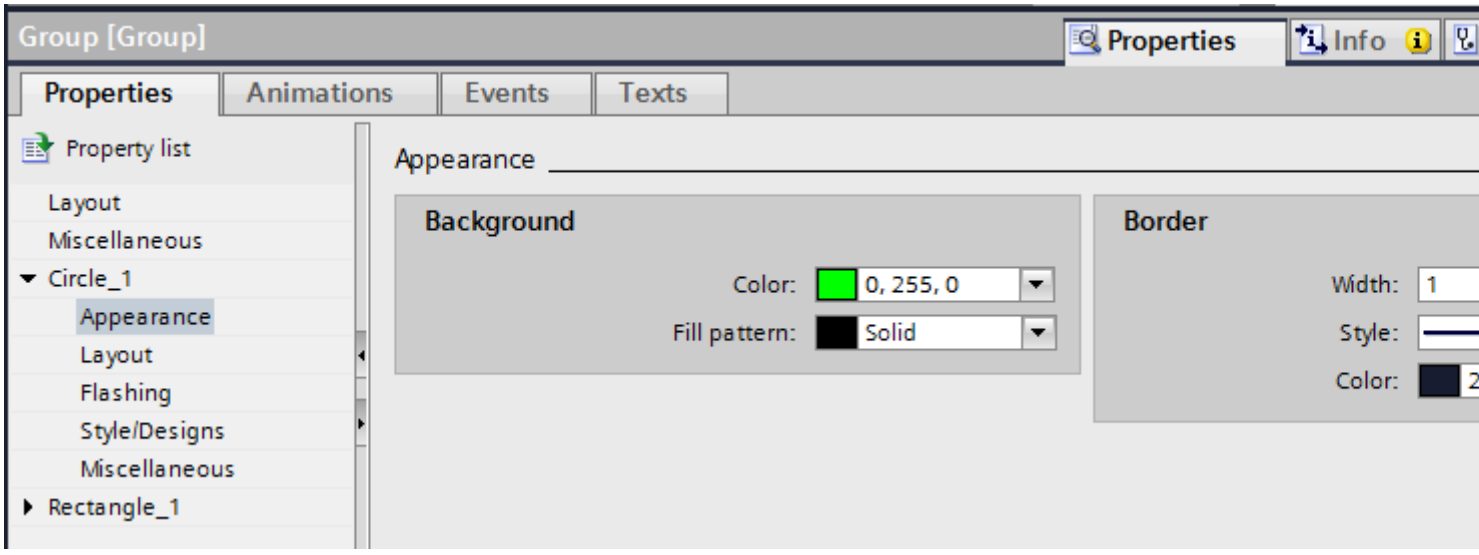

The properties of the object are displayed.

3. Change the object properties.

Note

The dynamization of properties for all objects of the group which have these properties is not possible in a group. You can only dynamize the properties of the objects associated with a group for each object itself.

Result

Although you have edited the object, it is still an element of the group. These changes do not affect the other objects of the group.

See also

Basics on groups (Page 157)

Example: Inserting and configuring a rectangle (Page 172)

1.2.3 Configuring the keyboard access

1.2.3.1 Overview of keyboard access (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

For keyboard units without a mouse, the operator activates control objects using the <Tab> key. You can set up keyboard input to make the process easier to run, and to make sure that the operator enters all the necessary values. If you are using the keyboard, use the <Tab> key to activate the objects in a certain order, and to enter the necessary values.

For HMI devices without key, you can simulate the <Tab> key by configuring the "SimulateSystemKey" system function to a function key.

Operator authorizations and operator control enables

If you configure an object for operator input with the <Tab> key, the object must have both an operator authorization, and an operator control enable.

Editing the tab sequence

The tab sequence is determined automatically when the control objects are created. The numbers of the tab sequence are assigned in the sequence in which the libraries are created.

It makes sense to change the tab sequence in the following cases:

- The operator changes directly to a specific control object.
- The screen requires a specific sequence

Change to the tab sequence mode to change the tab sequence. In this mode, the tab sequence number is displayed at the top left of the control objects. The tab sequence numbers of hidden objects are also shown. The distribution of these numbers is edited using the mouse.

Note

Other functions are not available in the tab sequence mode.

See also

Overview of keyboard access (Page 166)

Defining cursor mode (Page 167)

Defining the operator authorization and operator control enable for an object (Page 167)

Defining object types for tab cursor mode (Page 168)

Setting the tab sequence order (Page 169)

Setting the tab sequence order (Page 171)

Example: Inserting and configuring a rectangle (Page 172)

Overview of objects (Page 97)

1.2.3.2 Overview of keyboard access (RT Professional)

Introduction

To influence the process, the operator edits operable objects such as I/O fields or buttons in Runtime. You set up keyboard input to make the process easier to run, and to make sure that the operator enters all necessary values. When the operator makes entries, he uses the <Tab> key to activate the objects in a specified order and to enter the necessary values.

Keyboard access

There are two methods of mouseless operator input:

- Alpha cursor: The operator uses the <Tab> key in Runtime to activate objects of the "I/O field", and "text list" type only.
- Tab cursor: Depending on the settings, the operator uses the <Tab> key in Runtime to activate all objects that permit an input.

You define the cursor mode when Runtime starts. You set the cursor mode in the Inspector window of the screen.

Combining the Alpha cursor and Tab cursor modes

When you have defined a hotkey, the operator can toggle the cursor mode between the two modes during runtime if necessary. In this case you specify the tab sequence for both cursor types.

Operation in Runtime

You use the tab sequence to define in which order the operator can activate objects. The default setting is operation using the <Tab> key. If required, define operation using a hotkey or the mouse.

See also

Overview of keyboard access (Page 165)

Example: Inserting and configuring a rectangle (Page 172)

1.2.3.3 Defining cursor mode (RT Professional)

Introduction

You define the cursor mode when Runtime starts. The operator can change the cursor mode in Runtime as required. You set the cursor mode in the object properties of the screen.

Requirement

You have opened a screen which contains multiple objects.

Procedure

1. Click on a empty surface on the screen.
2. In the Inspector window, click "Properties > Properties > Miscellaneous".
3. Select the desired cursor mode under "Tab sequence".

Result

The required cursor mode is active when Runtime starts.

See also

Overview of keyboard access (Page 165)

Example: Inserting and configuring a rectangle (Page 172)

1.2.3.4 Defining the operator authorization and operator control enable for an object (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

If you configure an object for operator input with the <Tab> key, the object must have both an operator authorization, and an operator control enable.

Requirement

You have opened a screen which contains at least one object.

Procedure

1. Select the object.
2. Select "Properties > Properties > Security" in the Inspector window.
3. Select the operator authorization under "Authorization."
4. Activate the authorization to operate.

Result

The operator can use the <Tab> key in Runtime to select the object.

See also

Overview of keyboard access (Page 165)

Example: Inserting and configuring a rectangle (Page 172)

1.2.3.5 Defining object types for tab cursor mode (RT Professional)

Introduction

In the Inspector window, you can define the object types that the operator activates using the <Tab> key in "Tab cursor" mode.

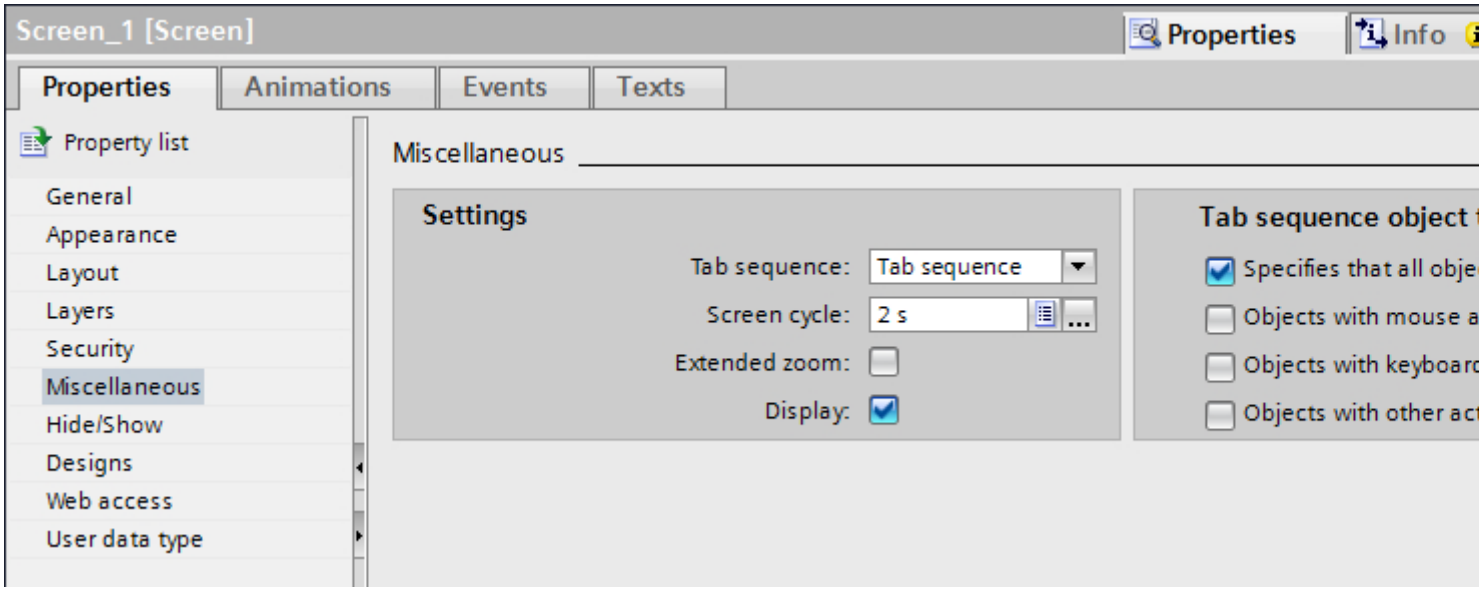

Requirement

- You have opened a screen which contains at least one object.
- The objects have been enabled for use in Runtime, and have operator authorization.
- "Tab cursor" mode is specified.

Procedure

1. Click on a empty surface on the screen.
2. Select the object types to be used in Runtime using the <Tab> key in the inspector window under "Properties > Properties > Miscellaneous > Tab sequence object types".

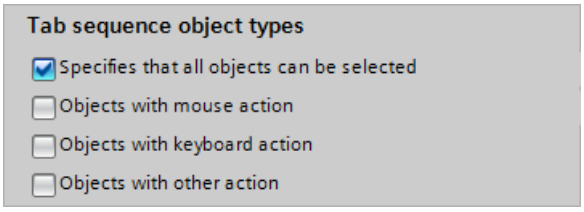

Result

The user can activate all the defined objects in Runtime using the <Tab> key in "Tab cursor" mode.

See also

Overview of keyboard access (Page 165)

Example: Inserting and configuring a rectangle (Page 172)

1.2.3.6 Setting the tab sequence order (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

All operable objects can be reached in runtime with the <Tab> key. You use the "Tab sequence" command to define the order in which the operator can activate objects in runtime.

Note

You cannot reach objects with the "Output" or "Two states" mode in runtime with the <Tab> key.

You can operate the screen in runtime:

- Using the <Tab> key
- Using the mouse
- Using a configured hotkey

Requirement

- The active screen contains operable objects.
- No object is selected.
- The objects have been enabled for use in runtime, and have operator authorization.

Procedure

1. Select "Edit tab sequence" in the "Edit" menu.
   Tab sequence mode is activated. The tab sequence number is displayed for all objects that can be used. The tab sequence number is also displayed for hidden objects.
2. Use edit the tab sequence mode, click the accessible objects in the order in which you want them to be activated using <Tab> in runtime.
   The following figure shows how the tab sequence is defined in the screen. In runtime, the <Tab> key first activates the alarm view (number 1), then the I/O field (number 2), and then the button (number 3):

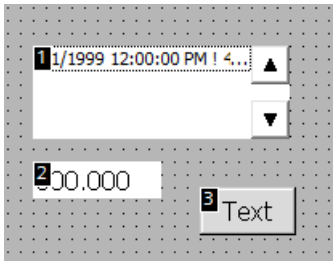

3. To exclude a screen object from the tab sequence, press the key combination <Shift+Ctrl> and click on the desired object.
   The tab sequence number is no longer displayed in the screen object. The screen object is now excluded from the tab sequence. The remaining tab sequence numbers are automatically decreased by 1.
4. To reenter an excluded screen object in the tab sequence, repeat step 3.
   The screen object entered as the first object in the tab sequence.

Result

The operator selects the objects in the specified order in Runtime with the <Tab> key.

Note

Changes after upgrades to a new software version

If you have configured a tab sequence in screens with faceplates in an older WinCC version, you should check the tab sequence of these screens after an upgrade to a newer WinCC version. The tab sequence may have been changed in both the screen and the faceplate.

See also

Overview of keyboard access (Page 165)

Example: Inserting and configuring a rectangle (Page 172)

1.2.3.7 Setting the tab sequence order (RT Professional)

Introduction

Every object can be accessed in Runtime in the mode "Tab cursor". You use the "Tab sequence" command to define the order in which the operator can activate objects in Runtime. You can operate the process screen in runtime:

- Using the <Tab> key
- Using the mouse
- Using a configured hotkey

Requirements

- You have opened a screen which contains at least one object.
- The objects have been enabled for use in Runtime, and have operator authorization.
- "Tab cursor" or "Alpha cursor" mode is activated.

Procedure

1. Select "Edit > Tab sequence > Edit tab cursor" or "Edit alpha cursor".
   Tab sequence mode is activated. The tab sequence number is displayed for all objects that can be used. The tab sequence number is also displayed for hidden objects.
2. Use edit the tab sequence mode, click the accessible objects in the order in which you want them to be activated using <Tab> in Runtime.
   The following figure shows how the tab sequence is defined in the screen. In Runtime, the <Tab> key first activates the I/O field (number 2), and then the button (number 3).

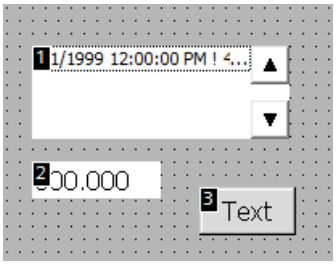

Result

The operator selects the objects in the specified order in Runtime with the <Tab> key.

Note

Changes after upgrading to a new software version

If you have configured a tab sequence in screens with faceplates in an older WinCC version, check the tab sequence of these screens after upgrading to a newer WinCC version. The tab sequence may have been changed in both the screen and the faceplate.

See also

Overview of keyboard access (Page 165)

Example: Inserting and configuring a rectangle (Page 172)

1.2.4 Examples

1.2.4.1 Example: Inserting and configuring a rectangle

Task

In this example, you insert a rectangle in a screen. You can configure the following properties:

- Name = "MyRectangle"
- Position = (20, 20)
- Size = (100,100)
- Color = red
- Black frame 2 pixels wide

Principle

The rectangle is a closed object which can be filled with a color or pattern. The height and width of a rectangle can be adjusted to allow its horizontal and vertical alignment.

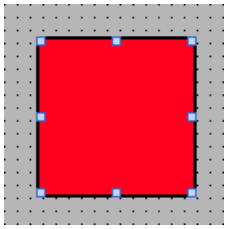

Overview

Carry out the following steps in order to create a rectangle:

- Inserting a rectangle
- Configuring a rectangle

See also

Basics on groups (Page 157)

Example: Inserting a rectangle (Page 173)

Example: Configuring a rectangle (Page 173)

Overview of objects (Page 97)

1.2.4.2 Example: Inserting a rectangle

Task

In this example, you insert and rename a rectangle. Do not use the special characters ?, ", /, \, *, <, > for the name.

Requirement

- A screen is open.
- The Inspector window is open.
- The "Tools" task card is open.

Procedure

1. Click the "Basic objects" palette in the "Tools" task card.
2. Drag the "Rectangle" object into the screen.
3. In the Inspector window, select "Properties > Properties > Miscellaneous".
4. Type in the new name "MyRectangle".

Result

The rectangle is now inserted and named "MyRectangle". The rectangle has the default properties of the "rectangle" object.

See also

Example: Inserting and configuring a rectangle (Page 172)

1.2.4.3 Example: Configuring a rectangle

Task

In this example you configure a rectangle:

- Color = red
- Black frame 2 pixels wide
- Position = (20, 20)
- Size = (100,100)

Changing the color of the rectangle

To change the color of the rectangle:

1. Select the rectangle.
2. Define the background color in "Properties > Properties > Appearance > Background > Color" in the Inspector window.
3. Select "Solid" as the fill pattern.
4. Define the color for the border in "Properties > Properties > Appearance > Border > Color" in the Inspector window.
5. Enter the value "2" for "width".
6. Select "Solid" as the "Style".

Interim result

The rectangle is red and has a black frame with a width of two pixels.

Repositioning and resizing the rectangle

To change the position and size of the rectangle:

1. Select the rectangle.
2. In the Inspector window, select "Properties > Properties > Layout".

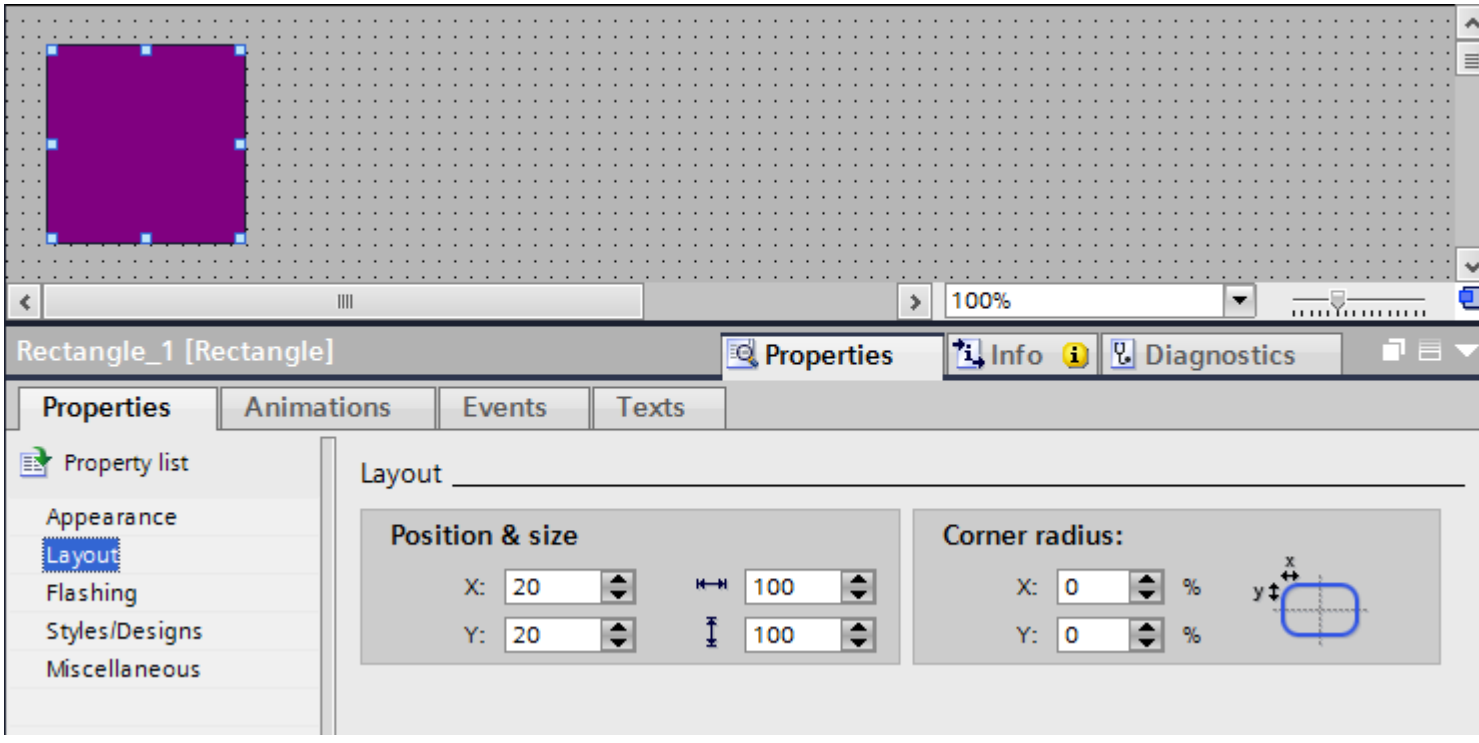

3. Set "20" for the both the X and Y coordinates under "Position & Size".
4. Set "100" for the height and for the width.

Result

The rectangle is positioned at the coordinates (20, 20), and has a width and height of 100 pixels.

See also

Example: Inserting and configuring a rectangle (Page 172)

1.3 Working with text lists and graphics lists

1.3.1 Working with text lists

1.3.1.1 Basics on text lists (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Texts are assigned to the values of a tag in a text list. Assign the text list to a symbolic I/O field for example in the configuration. This supplies the text to be displayed to the object. The text lists are created in the ""Text List" editor. You configure the interface between the text list and a tag at the object that uses the text list.

The selection of objects that can have a text list assigned depends on the Runtime.

Application

The text list is used, for example, to display a drop-down list in a symbolic I/O field.

If the symbolic I/O field is a display field, the associated texts will differ according to the value of the configured tags. If the symbolic I/O field is an input field, the configured tag assumes the associated value when the operator selects the corresponding text in Runtime.

Note

Display of tag values without text

The display of tag values to which no text has been assigned depends on the Runtime:

- The display and operating element remains empty.
- Three asterisks *** are displayed.

Ranges for the text list

Three types are available for the text lists:

- Value/Range
  This setting assigns text entries from the text list to integer values or value ranges of a tag. You can select the number of text entries as needed. The maximum number of entries depends on the HMI device you are using.
  You specify a default value which is shown if the value of the tag lies outside the defined range.
- Bit (0, 1)
  This setting assigns text entries from the text list to two states of a binary tag. You can create a text entry for each state of the binary tag.
- Bit number (0 - 31)
  This setting assigns a text entry from the text list to each bit of a tag. The maximum number of text entries is 32. This form of text list can be used, for example, in a sequence control when processing a sequence in which only one bit of the used tag may be set. You influence the behavior of the bit number (0 - 31) with the set bit of the least significance and a default value.

Multilingual texts

You can configure multiple languages for the texts in a text list. The texts will then be displayed in the set language in Runtime. To this purpose you set the languages in the Project window under "Language support > Project languages."

Configuration steps

The following steps are necessary to display texts in a symbolic I/O field for example:

1. Creating the text list
2. Assignment of the texts to values or value ranges of a text list
3. Assignment of a text list in the display object, for example, to the symbolic I/O field

See also

Creating a text list (Page 178)

Assigning texts and values to an area text list (Page 181)

Assigning texts and values to a bit text list (Page 182)

Assigning texts and values to a bit number text list (Page 184)

Notes for bit number text list (Page 185)

Configuring objects with a text list (Page 188)

Screen basics (Page 37)

Output of texts from a text list in an alarm (Page 801)

Configuring a text list in the faceplate type (Page 282)

1.3.1.2 Creating a text list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The text list allows you to assign specific texts to values and to output these in Runtime, for example in a symbolic I/O field. The type of symbolic I/O field can be specified, for example as a pure input field.

The following types of list are available:

- Value/Range
- Bit
- Bit Number

Procedure

1. Double-click "Text and graphics lists" in the project window.
2. Open the "Text lists" tab.

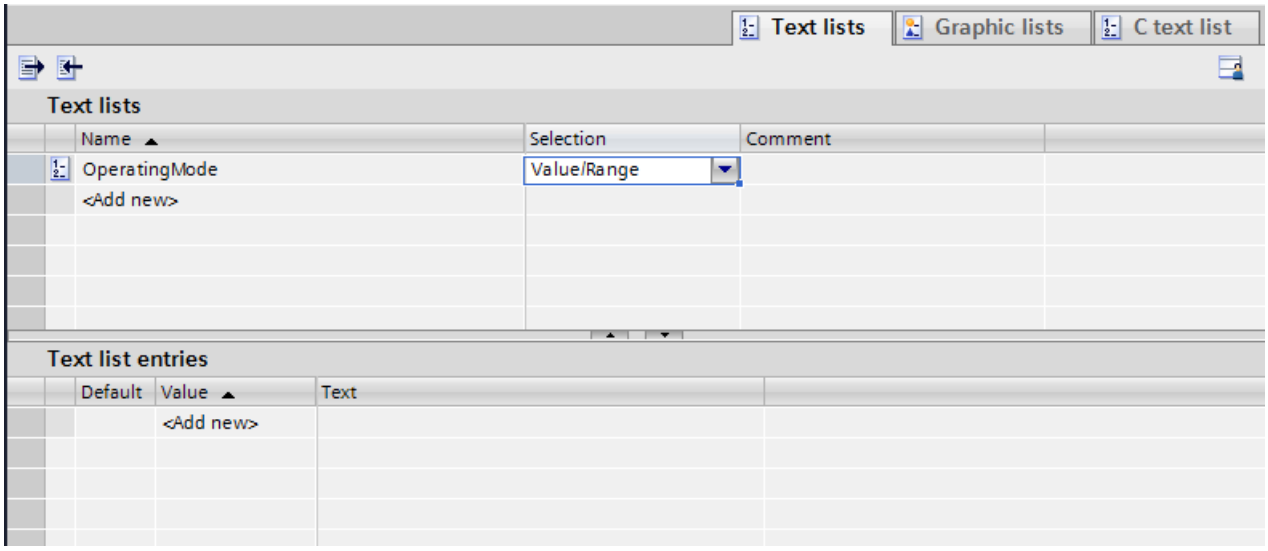

3. Click "Add" in the "Text lists" table.
   The Inspector window of the text list is open.
4. Assign a name to the text list that indicates its function.

5. Select the text list type under "Selection":
   - Value/Range: Text from the text list is displayed when the tag has a value that lies within the specified range.
   - Bit (0,1): A text from the text list is displayed when the tag has the value 0. A different text from the text list is displayed when the tag has the value 1.
   - Bit number (0-31): Text from the text list is displayed when the tag has the value of the assigned bit number.
6. Enter a comment for the text list.

Note

You should not use semicolons in the texts in a text list in WinCC Runtime Professional. The semicolon is a control character and is automatically deleted from a text.

Result

A text list is created.

See also

Basics on text lists (Page 176)

Configuring a text list in the faceplate type (Page 282)

1.3.1.3 Assigning texts and values to an area text list (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

For each area text list you specify which texts are displayed at which value range.

Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- An area text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.
   The Inspector window for this list entry opens.

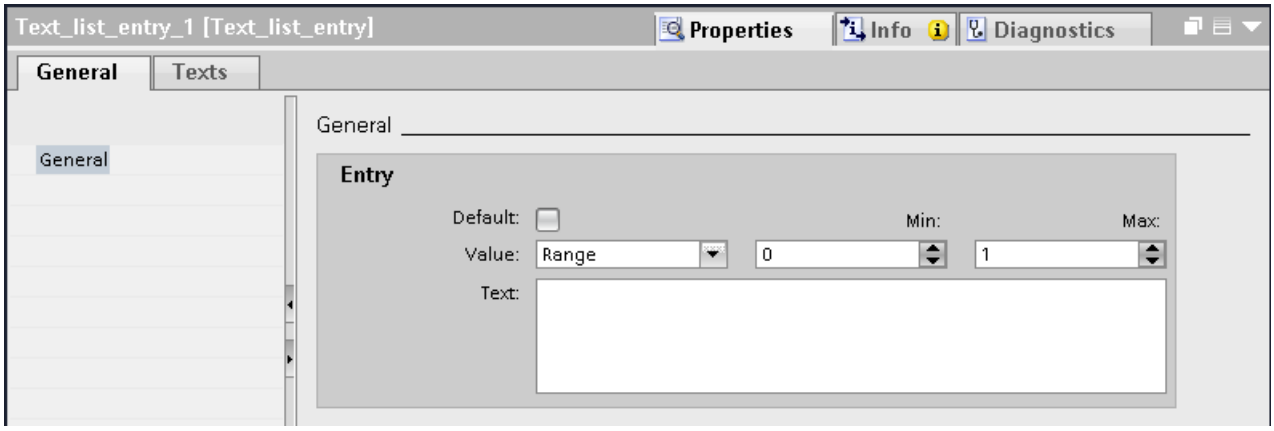

2. Select the setting "Range" in "Properties > Properties > General > Value" in the Inspector window.

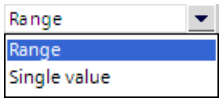

   – Enter the value "1" for "Min" for example.
   – Enter the value "20" for "Max" for example.
   – For "Text", enter the text that is displayed in Runtime if the tag is within the specified value range.

   Note

   Use a maximum of 320 characters for the text.

3. Click "Add" in the "Text list entries" table. A second list entry is created.
4. Select the setting "Range" in "Properties > Properties > General > Value" in the Inspector window.
   – Enter the value "21" for "Min" for example.
   – Enter the value "40" for "Max" for example.
   – For "Text", enter the text that is displayed in Runtime when the tag is within the specified value range.
5. If required, activate the "default entry".
   The entered text is always displayed when the tag has an undefined value. Only one default entry is possible per list.

Result

An area text list is created. Texts have been assigned to the possible value ranges.

See also

Basics on text lists (Page 176)

1.3.1.4 Assigning texts and values to an area text list (RT Professional)

Introduction

For each area text list you specify which texts are displayed at which value range.

Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- An area text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.
   The Inspector window for this list entry opens.

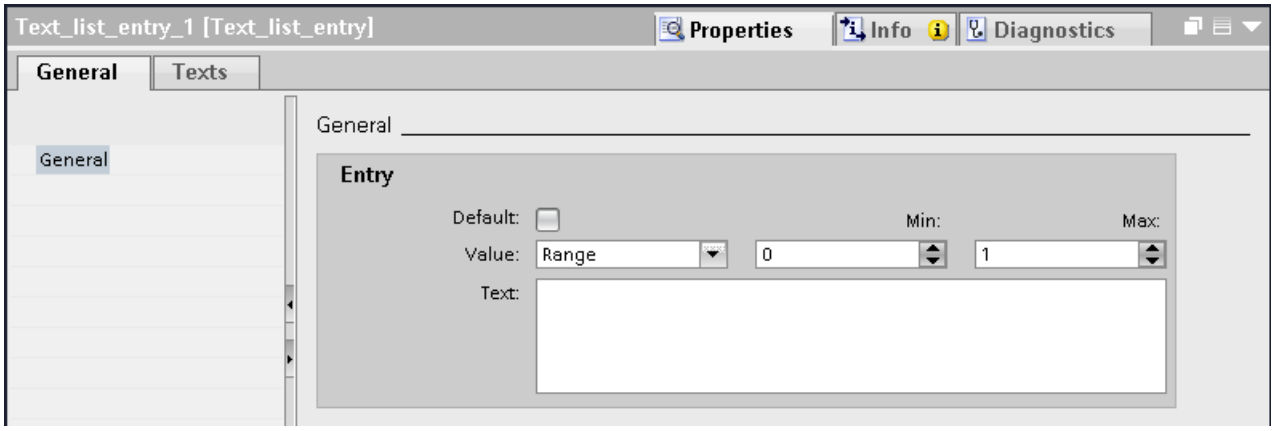

2. Select one of the following settings in the Inspector window under "Properties > Properties > General > Value".

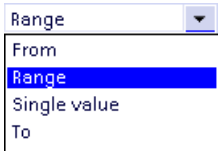

   – "Range": Minimum to maximum tag value, for example 1 ≦ Value ≦ 21
   – "To": Maximum tag value, e.g. value ≦ 13
   – "Individual value": Exactly one tag value, for example Value = 21
   – "From": Minimum tag value, e.g. value ≧ 2

3. Enter the text that is displayed in Runtime when the tag has the specified value or lies within the specified range of values under "Text."

   Note

   Use a maximum of 255 characters and no semicolons for the text.

4. If required, activate the "default entry".
   The entered text is always displayed when the tag has an undefined value. Only one default entry is possible per list.
5. Create further corresponding list entries for additional value ranges of the same text list.

Result

An area text list is created. Texts that appear in Runtime are assigned to the possible values.

See also

Basics on text lists (Page 176)

1.3.1.5 Assigning texts and values to a bit text list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

For each text list, you specify which text is displayed at which bit value.

Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- A bit text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.
   The Inspector window for this list entry opens.

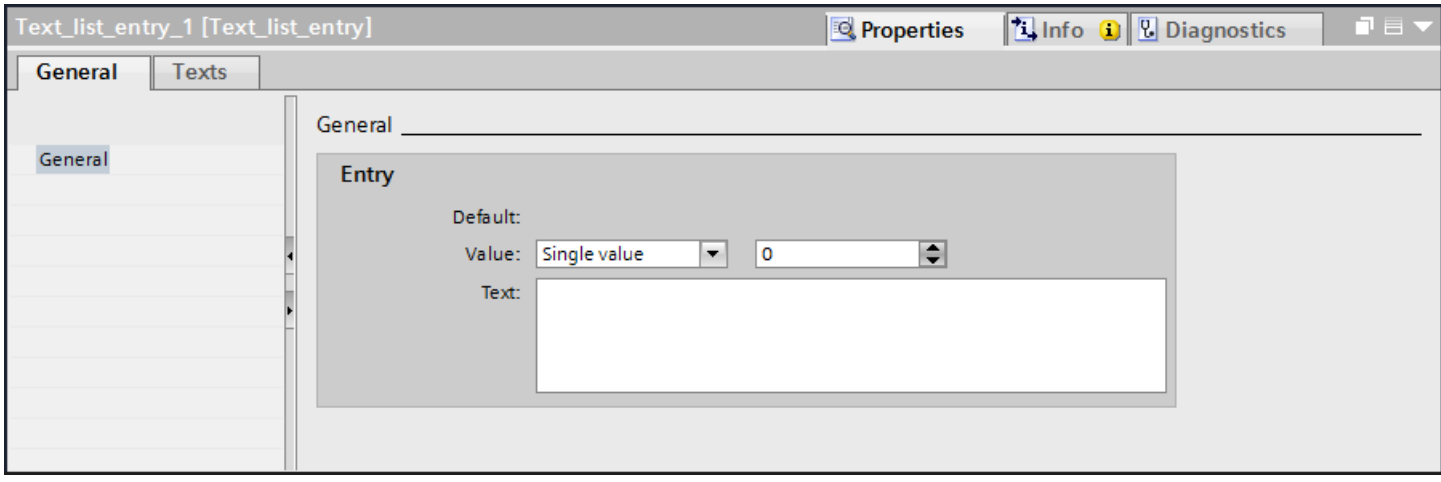

2. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "0" for "Value."
   - Enter the text which is displayed in Runtime under "Text" if the bit tag is set to "0".
3. Click "Add" in the "Text list entries" table. A second list entry is created.
4. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "1" under "Value."
   - Enter the text which is to be displayed in Runtime under "Text" if the bit tag is set to "1".

Note

Use a maximum of 320 characters for the text.

Use a maximum of 255 characters and no semicolons for the text in WinCC Runtime Professional.

Result

A bit text list is created. Texts that appear in Runtime are assigned to the possible values "0" and "1".

See also

Basics on text lists (Page 176)

1.3.1.6 Assigning texts and values to a bit number text list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

For each bit number text list you specify which texts are displayed at which bit number.

Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- A bit number text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.
   The Inspector window for this list entry opens.

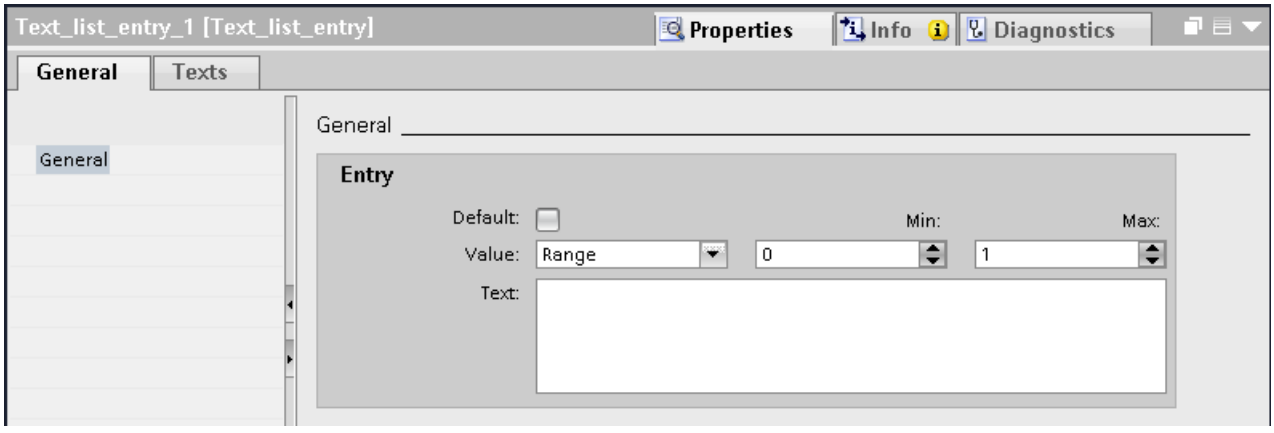

2. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   – Enter "10", for example, for "Value".
   – Under "Text", enter the text that is displayed in Runtime when the tag has the value "10".
3. If required, activate the "default entry".
   The entered text is always displayed when the tag has an undefined value. Only one default entry is possible per list.
4. Create further list entries for additional bit numbers of the same text list.

Note

Use a maximum of 320 characters for the text.

Use a maximum of 255 characters and no semicolons for the text in WinCC Runtime Professional.

Note

Bit selection for text lists

The output of the text list depends on the "Bit selection for text and graphics lists" option in the Runtime settings. Use this option to specify that the bit selection for text lists should be used on your HMI device. When you enable the option, the text displayed is the one configured for the set bit with the least significance. When you disable this option, only the text that has been configured for the set bit is displayed .

Result

A bit number text list is created. Texts that appear in Runtime are assigned to the specified bit numbers.

See also

Basics on text lists (Page 176)

1.3.1.7 Notes for bit number text list

Introduction

The bit number (0 - 31) range assigns a text entry from the list to each bit of a tag. If the option "Bit selection for text and graphic lists" is disabled in the Runtime settings and no default value is set, the following standard behavior applies:

- If only 1 bit is configured of all set bits, the stored text is displayed for the configured bit. In the following example, only the set bit with significance "4" is configured. Text 2 is displayed.

Significance	7	6	5	4	3	2	1	0
Set bits	0	0	1	1	0	1	0	0
Configured	-	Text 3	-	Text 2	Text 1	-	-	-

- If no bit is set or when several configured bits are set, no text is displayed.

Default value

Define a default value to prevent an empty display. A configured default value is displayed in the following cases:

- The option "Bit selection for text and graphic lists" is disabled. No bit or several configured bits are set in the tag.
- The option "Bit selection for text and graphic lists" is enabled and no bit is set or a text is not configured for the set bit with the least significance.

Displaying the default value

1. Enable the text for the default entry in the "Default" column of the "Text list entries" table. The value "Default entry" appears in the "Value" column of the text entry.
2. You can also select the "Default" option under "Properties > General" in the inspector window.

Set bit with the least significance

If no text is configured for the set bit with the least significance and if no

default value is set, nothing is displayed. If a default value is configured, the default value is displayed.

If no text is configured for the set bit with the least significance and if no default value is set, nothing is displayed. If a default value is configured, the default value is displayed.

To only display the text for the set bit with the least significance, enable the "Bit selection for text and graphic lists" option in the "Runtime settings" editor.

This setting is deselected by default to maintain downward compatibility. The setting is valid for all text lists of the HMI device.

Multiline text list entries

Use the <SHIFT>+<RETURN> shortcut to enter a line break in the text entry. Line breaks are represented by the "¶" paragraph mark.

Multiline text list entries are only output in symbolic output fields as well as on buttons with multiple lines. In all other cases, multiline texts are displayed with the paragraph mark.

1.3.1.8 Configuring objects with a text list (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

The output value and value application for text lists are specified in the display and operating object that displays the texts of the text list in Runtime. The properties of these objects are configured as required.

Requirement

- A text list is created.
- You have created a tag.
- The "Screens" editor is open.
- A screen with a symbolic I/O field is open. The object is edited.

Procedure

1. Select the text list which you want to have displayed in Runtime in "Properties > Properties > General > Text list" in the Inspection window.
2. Select the setting "Output" as the "Mode".

   Note
   Runtime dependency
   Different field types are available for a symbolic I/O field depending on the Runtime.

3. Select the tag the value of which determines the display in the symbolic I/O field as "Tag".

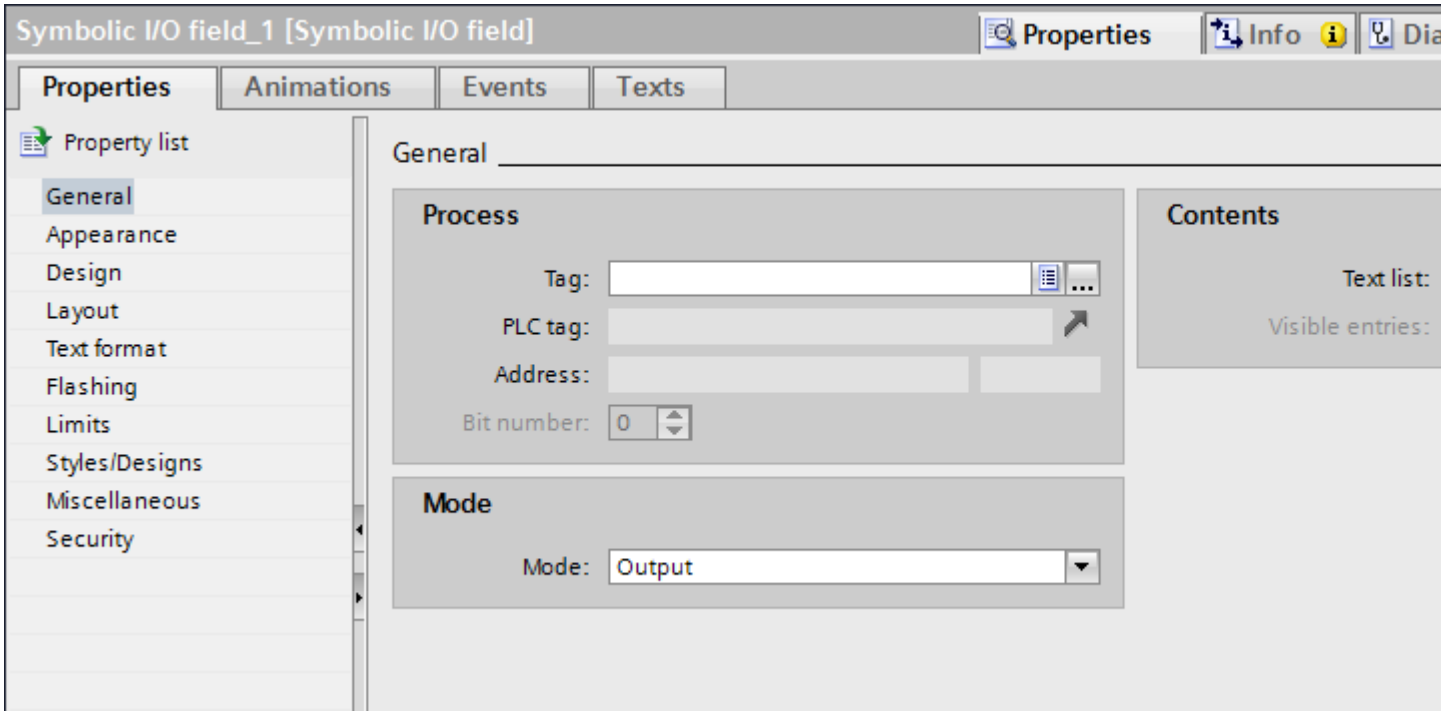

Result

The defined texts of the text list are displayed in the symbolic I/O field in Runtime when the tag has the specified value.

See also

Basics on text lists (Page 176)

1.3.1.9 Configuring objects with a text list (RT Professional)

Introduction

The output value and value application for text lists are specified in the display and operating object that displays the texts of the text list in Runtime. The properties of these objects are configured as required.

Requirement

- A text list is created.
- You have created a tag.
- The "Screens" editor is open.
- A screen with a symbolic I/O field is open. The object is edited.

Procedure

1. Select the text list which you want to have displayed in Runtime in "Properties > Properties > General > Text list" in the Inspection window.
2. Select the setting "Output" as the "Mode".

   Note
   Runtime dependency
   Different field types are available for a symbolic I/O field depending on the Runtime.

3. Select the tag the value of which determines the display in the symbolic I/O field as "Tag".

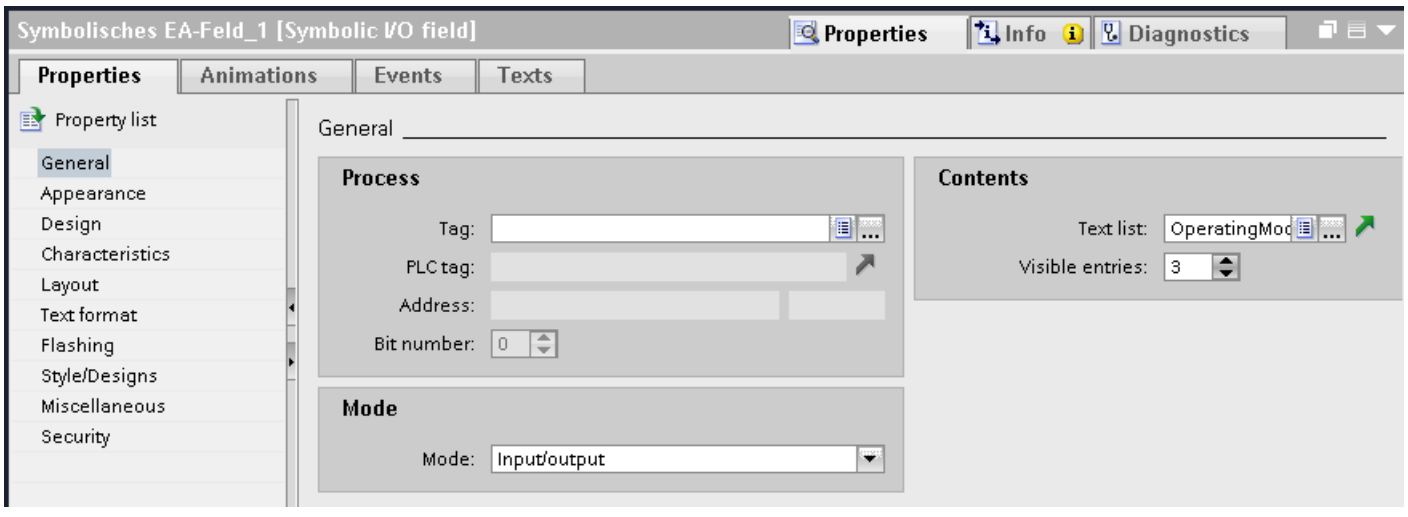

Result

The defined texts of the text list are displayed in the symbolic I/O field in Runtime when the tag has the specified value.

See also

Basics on text lists (Page 176)

1.3.2 Working with graphics lists

1.3.2.1 Basic principles of graphics lists (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The possible values of a tag are assigned to specific graphics in a graphics list. During configuration, you can create a graphics list for a button or a graphic I/O field. This supplies the graphics to be displayed to the object.

The graphics lists are created with the "Text and graphics list" editor. You configure the interface between the graphics list and a tag at the object that uses the graphics list. The availability of the graphics list is determined by the HMI device used.

Application

You can configure the graphics list for the following situations:

- Drop-down list with a graphic I/O field
- State-specific graphic for a button

The graphics in a graphics list can be configured as multilingual. The graphics will then be displayed in the set language in Runtime.

Graphic sources

Graphics can be added to the graphics list from the following sources:

- By selecting from a graphic browser
- Selection of an existing file
  You can use the following file types:
  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tiff, *.png, *.svg, *.jpeg and *.jpg.
- By creating a new file

Function

If the graphic I/O field is a display field, the associated graphics will differ according to the value of the configured tags. If the graphic I/O field is an input field, the configured tag assumes the associated value when the operator selects a graphic in Runtime.

Ranges for the graphics list

Three types are available for the graphics lists:

- Value/Range
  This setting assigns graphic entries from the graphics list to integer values or value ranges of a tag. You can select the number of graphic entries as needed. The maximum number of entries depends on the HMI device you are using.
  You specify a default value which is shown if the value of the tag lies outside the defined range.
- Bit (0, 1)
  This setting assigns graphic entries from the graphics list to two states of a binary tag. You can create a graphic entry for each state of the binary tag.
- Bit number (0 - 31)
  This setting assigns a graphic entry from the graphics list to each bit of a tag. The maximum number of graphic entries is 32. This form of graphics list can be used, for example, in a sequential control chart when processing a sequencer in which only one bit of the used tag may be set. You influence the behavior of the bit number (0 - 31) with the set bit of the least significance and a default value.

Configuration steps

The following tasks are required to display graphics, for example, in a graphic I/O field:

1. Creating the graphics list
2. Assignment of the graphics to values or value ranges of a graphics list
3. Assignment of a graphics list in the display object, for example, the graphic I/O field

See also

Creating a graphics list (Page 192)

Assigning a graphic and values to an area graphics list (Page 193)

Assigning graphics and values to a bit graphics list (Page 197)

Assigning graphics and values to a bit number graphics list (Page 198)

Notes for bit number graphics list (Page 200)

Configuring objects with a graphics list (Page 201)

Configuring a graphics list in the faceplate type (Page 283)

Screen basics (Page 37)

1.3.2.2 Creating a graphics list (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

The graphics list allows you to assign specific graphics to variable values and to output these in a graphic IO field in Runtime. You can specify the type of graphic I/O field, for example as a pure output field.

Procedure

1. Double-click "Text and graphics lists" in the project window.
2. Open the "Graphics lists" tab.

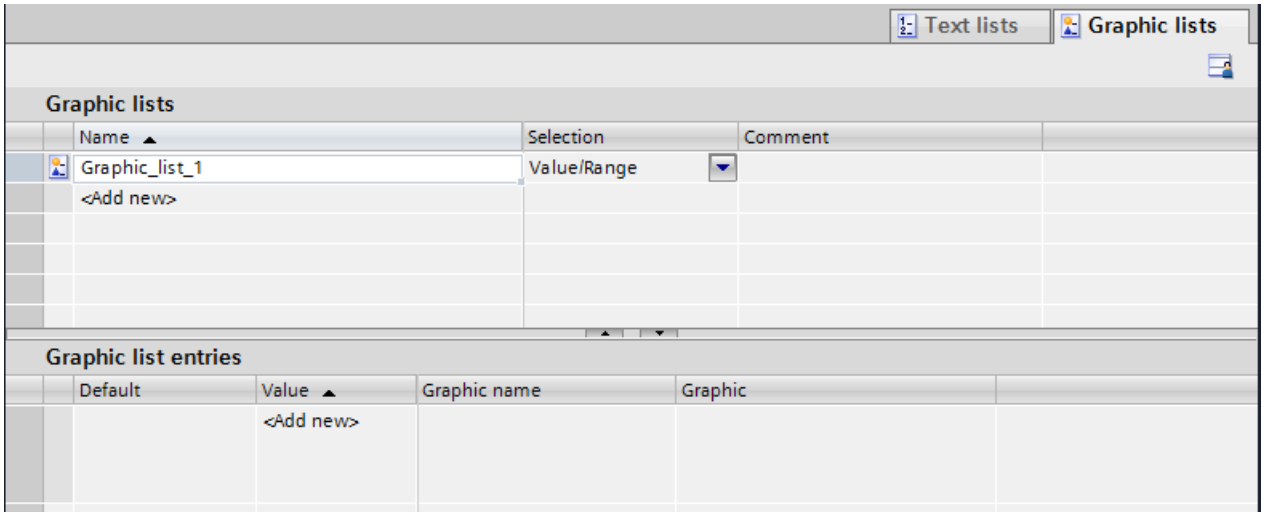

3. Click "Add" in the "Graphics lists" table. The Inspector window of the graphics list will open up.

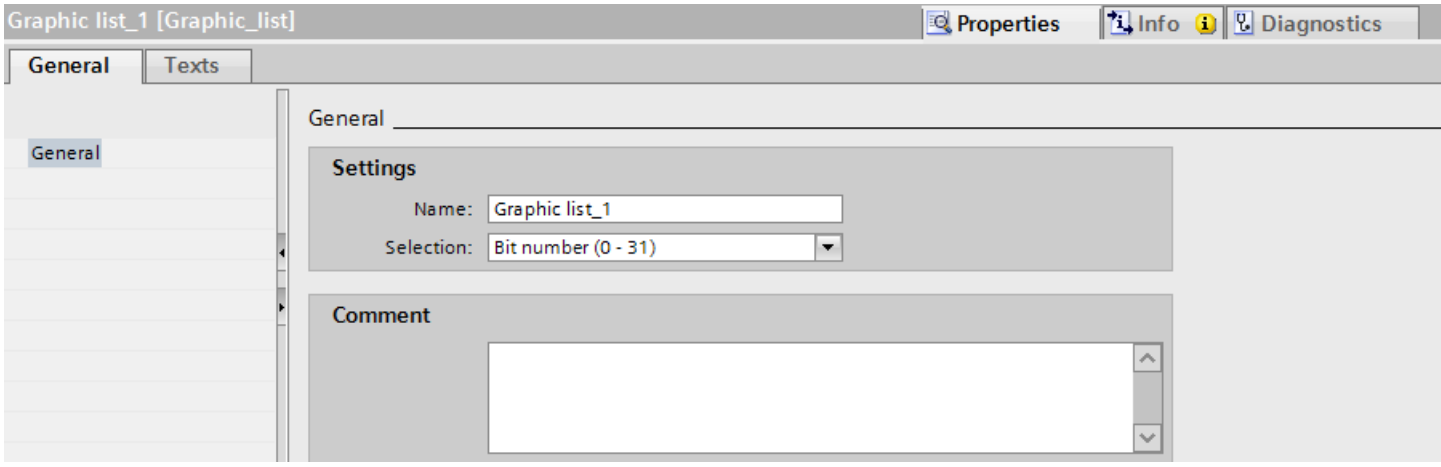

4. Assign a name to the graphics list that indicates its function.
5. Select the graphics list type "Bit number (0 - 31)" for example under "Select".
6. Enter a comment for the graphics list.

Result

A graphics list of the type "Range (0 - 31)" is created.

See also

Basic principles of graphics lists (Page 189)

1.3.2.3 Creating a graphics list (RT Professional)

Introduction

The graphics list allows you to assign specific graphics to variable values and to output these in a graphic IO field in Runtime. You can specify the type of graphic IO field, for example as a pure output field.

Procedure

1. Double-click "Text and graphics lists" in the project navigation.
2. Open the "Graphics lists" tab.
3. Click "Add" in the "Graphics lists" table.
   The Inspector window of the graphics list will open up.

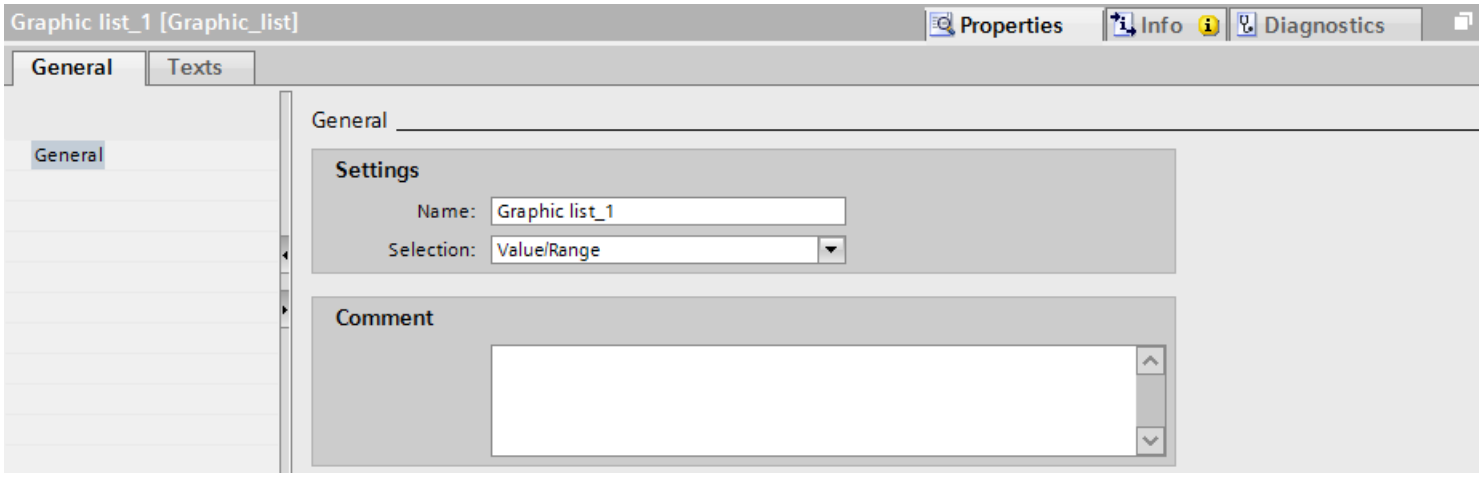

4. Assign a name to the graphics list that indicates its function.
5. Select the "Value/Range" graphics list type under "Selection."
6. Enter a comment for the graphics list.

Result

An area graphics list is created.

See also

Creating a graphics list (Page 190)
Configuring a graphics list in the faceplate type (Page 283)

1.3.2.4 Assigning a graphic and values to an area graphics list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

For each area graphics list you specify which graphics are displayed at which value range.

Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is open.
- An area graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.
   The Inspector window for this list entry opens.

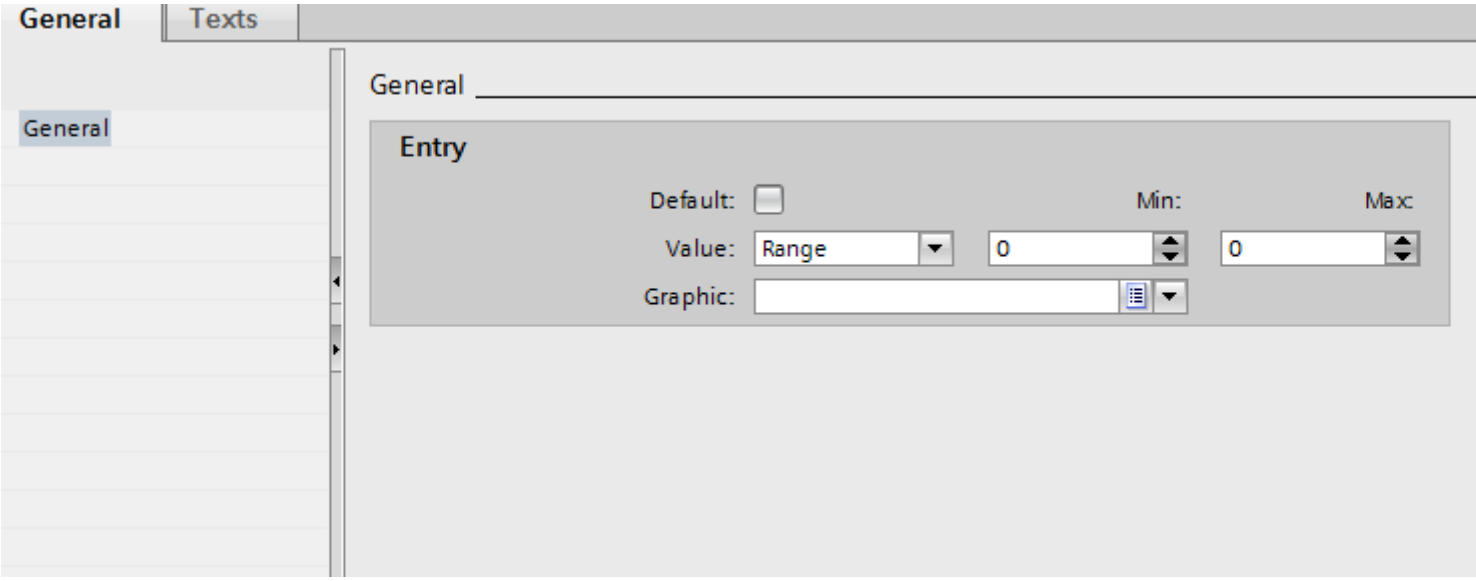

2. Select the settings "Range" in "Properties > Properties > General > Value" in the Inspector window:
   - Enter the value "1" for "Min" for example.
   - Enter the value "20" for "Max" for example.
   - Select a graphic that is displayed in Runtime when the tag is within the specified value range.

Note

As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:

1. Select a graphic in the library or in your file system.
2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. Click "Add" in the "Graphics list entries" table. A further list entry is created.

4. Select the settings "Single value" in "Properties > Properties > General > Value" in the Inspector window:
   – Enter the value "21" for example.
   – Select a graphic which is displayed in Runtime if the bit "21" is set in the tag.

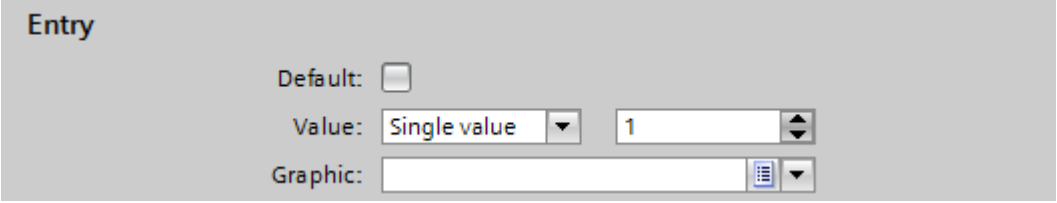

5. If required, activate the "default entry".
   The graphic is always displayed when the tag has an undefined value. Only one default entry is possible per list.

Result

An area graphics list is created. Graphics that appear in Runtime are assigned to the possible values.

See also

Basic principles of graphics lists (Page 189)

1.3.2.5 Assigning graphics and values to an area graphics list (RT Professional)

Introduction

For each area graphics list you specify which graphics are displayed at which value range.

Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is open.
- An area graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.
   The Inspector window for this list entry opens.

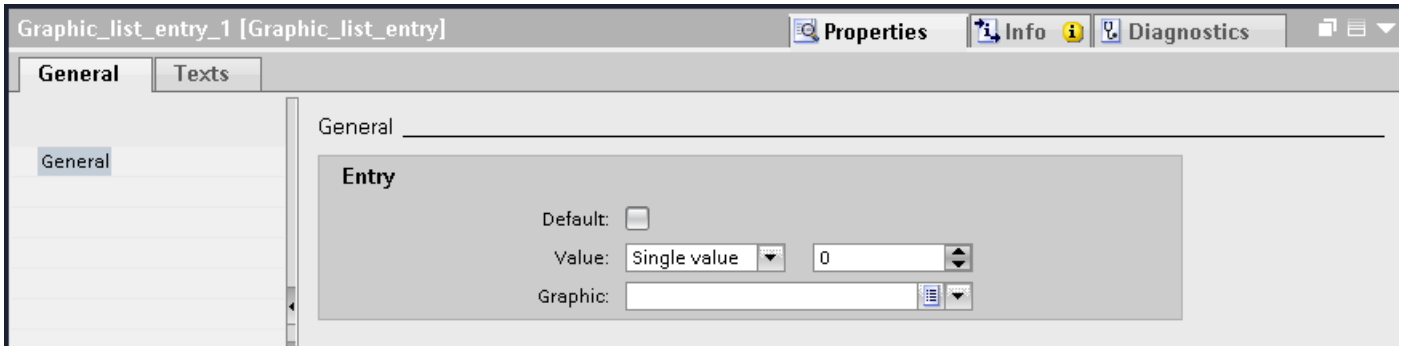

2. Select the settings "Single value" in "Properties > Properties > General > Value" in the Inspector window:
   - Enter the value "0" for example.
   - Select a graphic which is displayed in Runtime if the bit "0" is set.

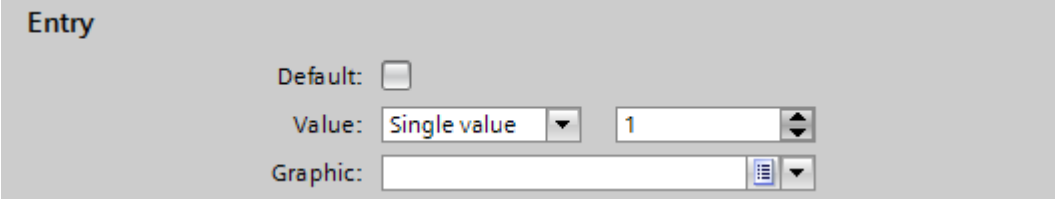

Note

As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:

1. Select a graphic in the library or in your file system.
2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. If required, activate the "default entry".
   The graphic is always displayed when the tag has an undefined value. Only one default entry is possible per list.
4. Activate "Properties > Properties > General > Use transparent color" in the inspector window and select a color.
5. Activate "Properties > Properties > Flash > Use flash" in the inspector window.
6. Select "Flash frequency > Slow" for example.
7. Create further list entries for additional bit numbers of the same graphics list.

Result

An area graphics list is created. Graphics that appear in Runtime are assigned to the specified bit numbers. The graphic flashes when the tag has a defined value.

1.3.2.6 Assigning graphics and values to a bit graphics list (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

For each graphics list you specify which graphic is displayed at which bit value.

Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is opened.
- A bit graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.
   The Inspector window for this list entry opens.

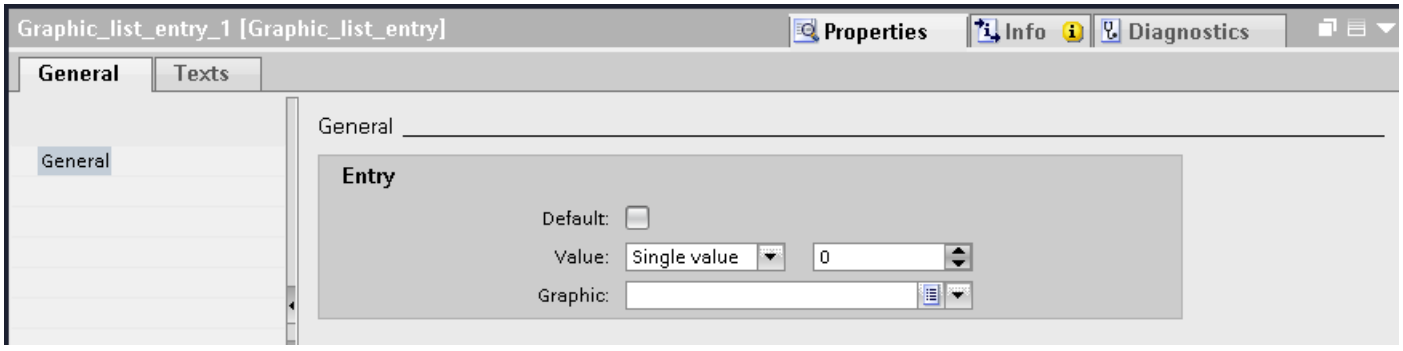

2. Select the settings "Single value" in the inspector window "Properties > Properties > General > Value":
   - Enter "0" as the value.
   - Select a graphic which is displayed in Runtime if the bit "0" is set in the tag.

   Note

   As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:
   1. Select a graphic in the library or in your file system.
   2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. Click "Add" in the "Graphics list entries" table. A new list entry is created.
4. Select "Properties > Properties > General > Value > Single value": in the Inspector window.
   - Enter "1" as the value.
   - Select a graphic which is displayed in Runtime if the bit "1" is set in the tag.

Result

A bit graphics list is created. Graphics that appear in Runtime are assigned to the values "0" and "1".

See also

Basic principles of graphics lists (Page 189)

1.3.2.7 Assigning graphics and values to a bit number graphics list (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

For each bit number graphics list you specify which graphics are displayed at which bit number.

Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is open.
- A bit number graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.
   The Inspector window for this list entry opens.

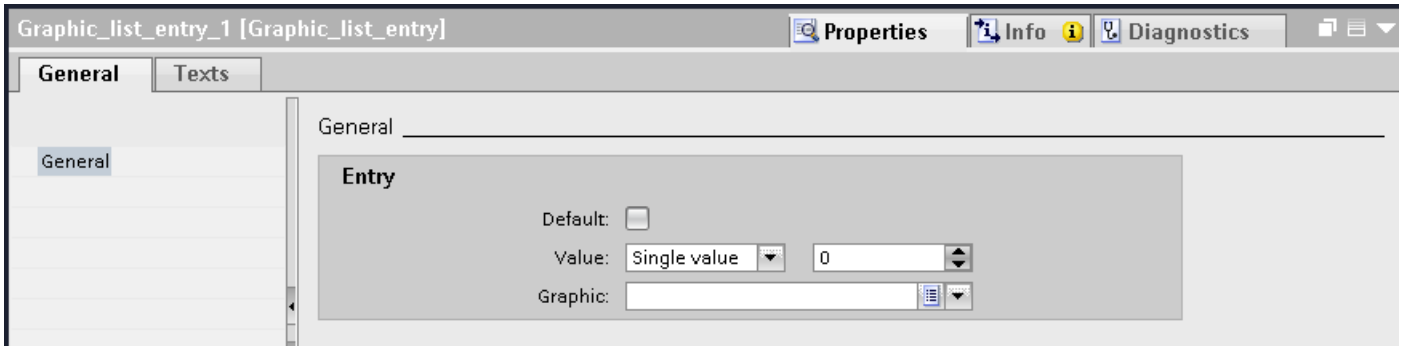

2. Select the settings "Single value" in the Inspector window "Properties > Properties > General > Value":
   - Enter the value "1" for example.
   - Select a graphic which is displayed in Runtime if the bit "0" is set in the tag.

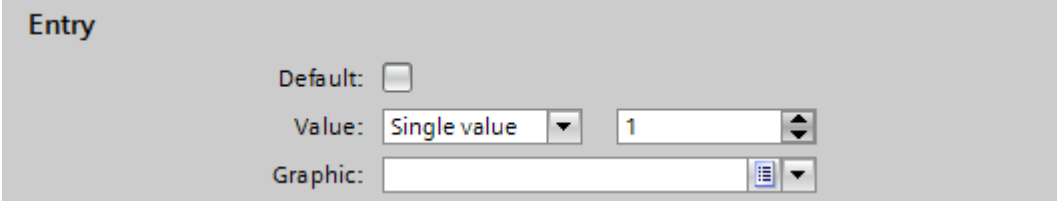

Note

As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:

1. Select a graphic in the library or in your file system.
2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. If required, activate the "default entry".
   The graphic is always displayed when the tag has an undefined value. Only one default entry is possible per list.
4. Create further list entries for additional bit numbers of the same graphics list.

Note

Bit selection for graphic lists

The output of the graphic list depends on the "Bit selection for text and graphics lists" option in the Runtime settings. Use this option to specify that the bit selection for graphic lists should be used on your HMI device. When you enable the option, the graphic displayed is the one configured for the set bit with the least significance. When you disable this option, the graphic that has been configured only for the set bit is displayed.

Result

A bit number graphics list is created. Graphics that appear in Runtime are assigned to the specified bit numbers.

See also

Basic principles of graphics lists (Page 189)

1.3.2.8 Notes for bit number graphics list

Introduction

The bit number (0 - 31) range assigns a graphic entry from the list to each bit of a tag. If the option "Bit selection for text and graphic lists" is disabled in the Runtime settings and no default value is set, the following standard behavior applies:

- If only 1 bit is configured of all set bits, the stored graphic is displayed for the configured bit. In the following example, only the set bit with significance "4" is configured. Graphic 2 is displayed.

Significance	7	6	5	4	3	2	1	0
Set bits	0	0	1	1	0	1	0	0
Configured	-	Graphic 3	-	Graphic 2	Graphic 1	-	-	-

- If no bit is set or when several bits are set that are also configured, only the cactus graphic is displayed.

Default value

Define a default value to prevent an empty display. A configured default value is displayed in the following cases:

- The option "Bit selection for text and graphic lists" is disabled. No bit or several configured bits are set in the tag.
- The option "Bit selection for text and graphic lists" is enabled and no bit is set or a graphic is not configured for the set bit with the least significance.

Displaying the default value

1. Enable the graphic for the default entry in the "Default" column of the "Graphics list entries" table.
   The value "Default entry" appears in the "Value" column of the entry.
2. You can also select the "Default" option under "Properties > General" in the inspector window.

Set bit with the least significance

When "Bit selection for text and graphic lists" is enabled, the graphic displayed is the one configured for the set bit with the least significance.

If no graphic is configured for the set bit with the least significance and if no default value is set, the cactus graphic is displayed. If a default value is configured, the graphic configured for the default value is displayed.

To only display the graphic for the set bit with the least significance, enable the "Bit selection for text and graphic lists" option in the "Runtime settings" editor.

This setting is deselected by default to maintain downward compatibility. The setting is valid for all graphic lists of the HMI device.

1.3.2.9 Configuring objects with a graphics list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The output value and value application for graphics list are specified in the display and operating object that displays the graphics of the graphics list in Runtime. The properties of these objects are configured as required.

Requirement

- A graphics list is created. The values have been defined. Graphics have been assigned to the values.
- You have created a tag.
- The "Screens" editor is open.
- A screen with a graphics I/O field is open. The object is edited.

Procedure

1. Select the graphics list the graphics of which you want to have displayed in Runtime in "Properties > Properties > General > Graphics list" in the Inspector window.
2. Select the "Input/output" setting for "Mode".

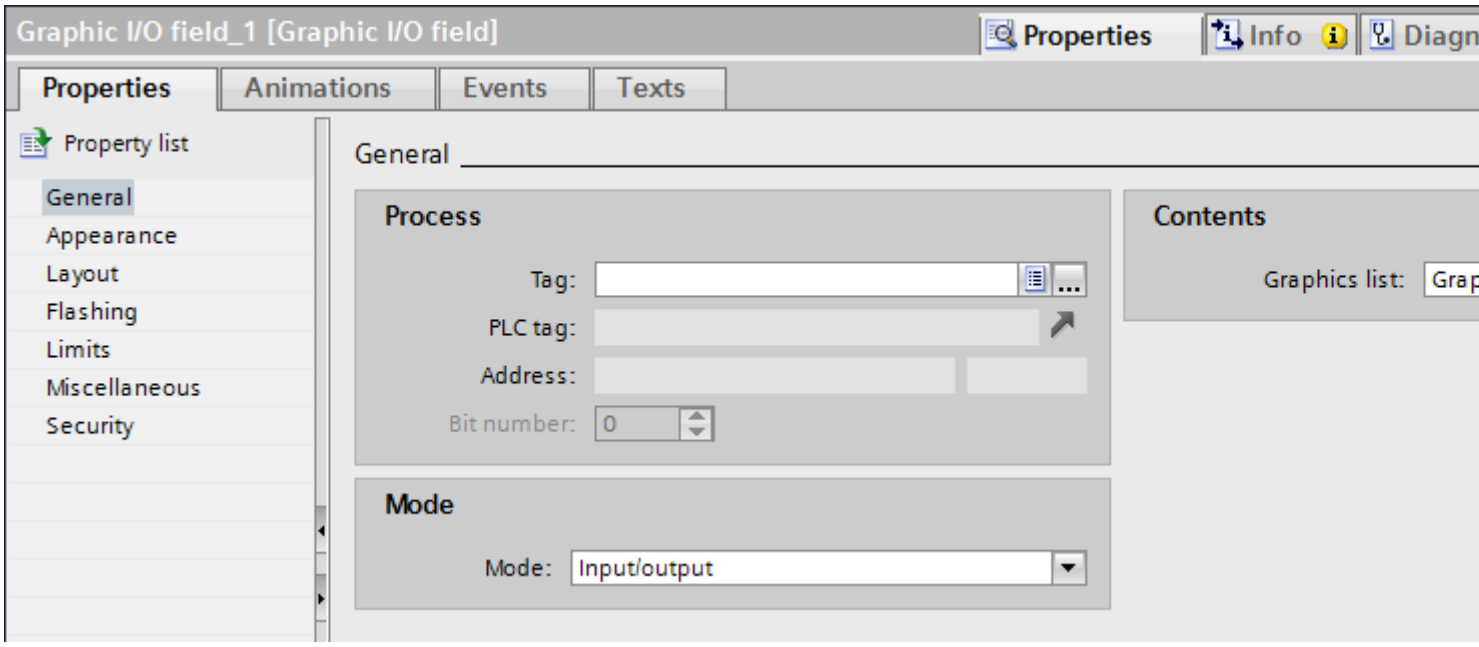

> **Note**
> **Runtime dependency**
>
> Different field types are available for a graphic I/O field depending on the Runtime.

3. As "Tag", select the tag whose values are defined by the display in the graphic I/O field.

Result

The defined graphics of the graphics list are displayed in the graphic I/O field in Runtime when the tag has the specified value.

See also

Basic principles of graphics lists (Page 189)

1.3.3 Working with C text lists (RT Professional)

1.3.3.1 Basics on C text lists (RT Professional)

Introduction

Each HMI device is assigned to a C text list. In this list you create multilingual texts, which you then reference via the text ID in a C script. You use the C text list exclusively with functions of the API.

Use

You use the C text list, for example, to display texts in a screen object.

Multilingual texts

You can configure multiple languages for the texts in a C text list. The texts will then be displayed in the set language in Runtime. To this purpose you set the languages in the Project window under "Language support > Project languages".

Configuration steps

The following steps are necessary to display texts in a screen object, for example:

1. Create entries in the C text list
2. Create a screen object
3. Link C script, for example to an event of the screen object
4. Reference entries in the C script through the corresponding functions

See also

Creating entries in the C text list (Page 203)

1.3.3.2 Creating entries in the C text list (RT Professional)

Introduction

In the C text list you create multilingual texts, which you then reference via the text ID in a C script. You use the C text list exclusively with functions of the API.

Procedure

To create an entry in the C text list, follow these steps:

1. Double-click "Text and graphics lists" in the project tree.
2. Open the "C text list" tab.

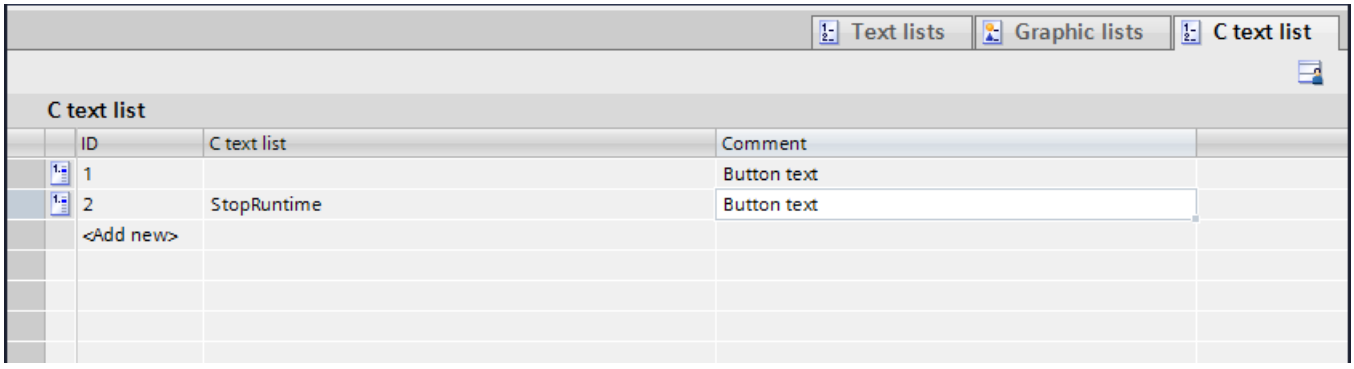

3. Click "Add" in the table.
   The Inspector window of the C text list opens.
4. Enter the text and your comment, if needed.
   The text ID is created automatically.
5. Change the ID as needed to identify texts that belong together.
   The text ID must be within the number range 1 to 2147483647 and it must be unique. The uniqueness is checked by the system.

   Note

   If you change the text ID of an entry already referenced, this change is ignored by the syntax check in the script editor.

Result

An entry has been created in the C text list.

See also

Basics on C text lists (Page 203)

1.4 Dynamizing screens

1.4.1 Basics on dynamizing screens

1.4.1.1 Basics on dynamizing screens (Basic Panels)

Dynamizing objects

In WinCC you dynamize objects to map your system and show processes on HMI devices.

You implement dynamizations by

- Animations
- Tags
- System functions

One example is the mapping of a tank, the liquid level of which rises or falls in relation to a process value.

The options for dynamization depend on the object involved. When you copy an object, its dynamization functions are included.

See also

Basics on dynamizing screens (Page 205)

Dynamization in the inspector window (Page 208)

Configuring a new animation (Page 212)

Show overview of the dynamizations (Page 211)

Basics on dynamization in the property list (Page 232)

Screen basics (Page 37)

Basic on events (Page 239)

1.4.1.2 Basics on dynamizing screens (RT Professional, RT Advanced, Panels, Comfort Panels)

Dynamizing objects

In WinCC you dynamize objects to map your system and show processes on HMI devices.

You implement dynamizations by

- Animations
- System functions

- Tags
- Local scripts

One example is the mapping of a tank, the liquid level of which rises or falls in relation to a process value.

The options for dynamization depend on the object involved. When you copy an object, its dynamization functions are included.

Customized menus and toolbars

Use the "Menus and Toolbars" editor to configure customized menus and toolbars. These menus and toolbars are displayed in all screens of your project or in the screen windows. The menu entries and icons are connected by means of tags or VB scripts.

Use the customized menus and toolbars, for example, to change from a screen in your project to any other screen.

See also

Basics on dynamizing screens (Page 205)

Dynamization in the inspector window (Page 208)

Cycle basics (Page 596)

1.4.1.3 Dynamization in the inspector window (Basic Panels)

Introduction

Basically, you can dynamize all the screen objects which you have configured in a screen. Which dynamization possibilities and which events are available depends on the device and the selected object.

Animations

WinCC helps you to implement dynamization using predefined animations. If you want to animate an object, first configure the desired animation in the object's inspector window. Then adapt the animation to the requirements of your project.

The selection of the supported animations depends on the HMI device and the selected object. You choose between the following types of animation:

- Layout: Appearance, visibility
- Movements: direct, diagonal, horizontal and vertical movement
- Variable binding

You can configure the "Variable binding" type of animation several times for one object.

You configure animations in the "Properties > Animations" inspector window.

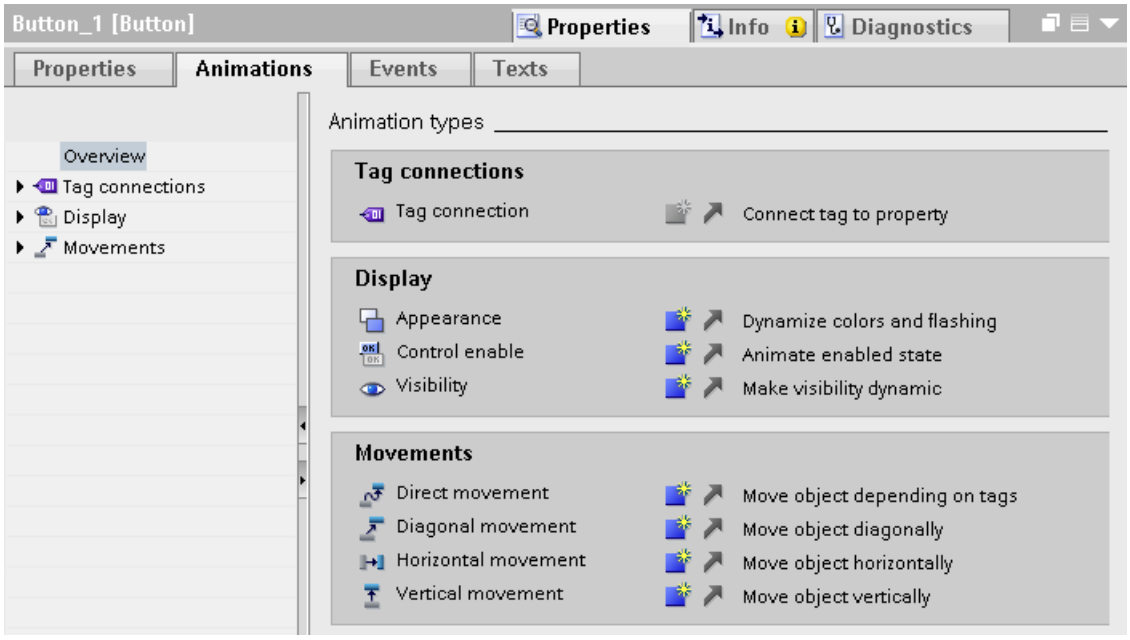

Events

Operable objects also react to events, such as a mouse click.

You configure a function list with system functions on an event. The system functions are processed as a reaction to the triggered event.

You configure events in the "Properties > Events" inspector window.

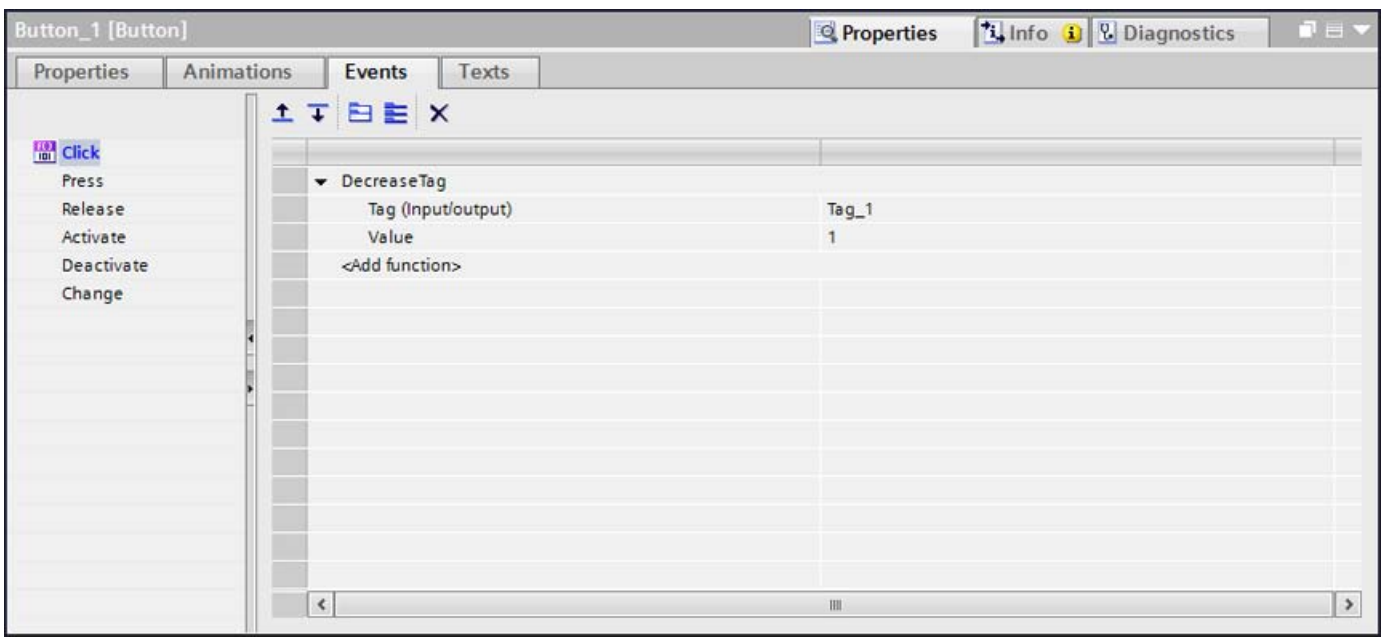

You can find more detailed information on this under "Working with Function Lists (Page 1379)".

See also

Basics on dynamizing screens (Page 205)

Working with Function Lists (Page 1379)

1.4.1.4 Dynamization in the inspector window (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Basically, you can dynamize all the screen objects which you have configured in a screen. Which dynamization possibilities and which events are available depends on the device and the selected object.

The inspector window offers you the possibilities for dynamization of screen objects in three tabs.

Property list

With the button you go to the properties list in the "Properties > Properties" inspector window.

In the property list, you set any dynamization for the individual object properties using tags or local C and VB scripts. You can, for example, dynamically change the background color of an object.

In the "Dynamic Value" column you can dynamize the object property of a tag, a C-script or a VB-script.

Whether an object property can be dynamized is indicated in the Property list by means of the background color:

- Light gray: Can be dynamized
- Dark gray: Cannot be dynamized

Any animations you have already configured will also appear in the object property list, where you can edit the animations.

The following table shows you how to sort the property list:

	Sort by alphabet
	Sort by category

With the button you return to the property pages in the inspector window.

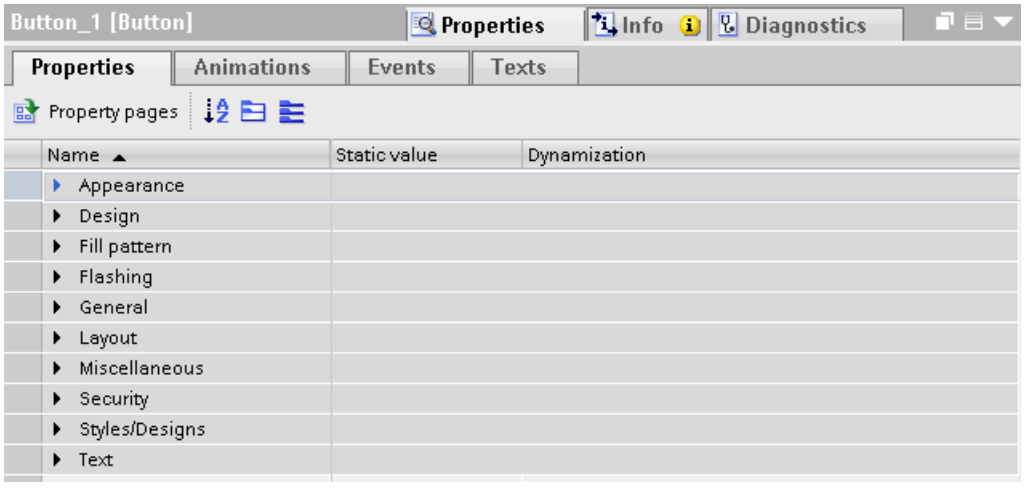

Animations

You have two different options for animating objects.

You configure animations

- in the inspector window under "Properties > Animations",
- in the "Animations" task card.

WinCC helps you to implement dynamization using predefined animations. If you want to animate an object, first configure the desired animation with the tools of the toolbox or in the object's inspector window. Then customize the animation in the Inspector window to suit the requirements of your project.

The selection of the supported animations depends on the device and the selected object. You choose between the following types of animation:

- Tag binding
- Layout: Appearance, control enable, visibility
- Movements: direct, diagonal, horizontal and vertical movement
- Property animation

Note

HMI device dependency

The "Control enable" animation is not available for every HMI device.

You can configure the "Tag binding" and "Property animation" types of animation several times for one object.

You configure animations in the "Properties > Animations" inspector window.

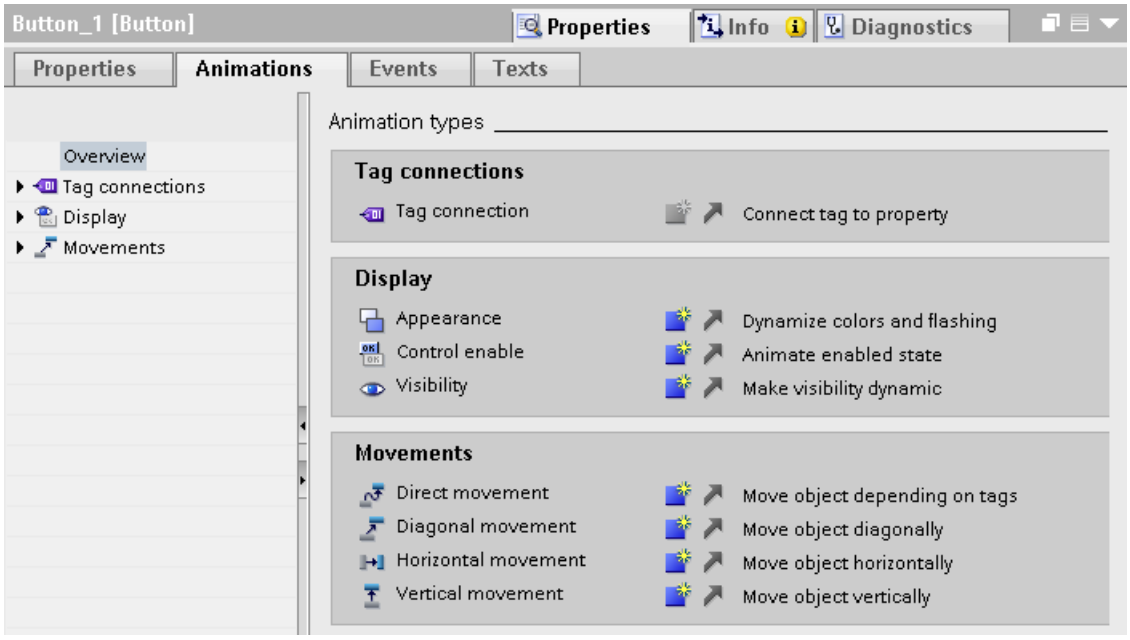

Events

Operator control enabled objects also react to events, such as a mouse click.

You configure a function list with system functions or local scripts on an event. The system functions or the local script are processed as a reaction to the triggered event.

You configure events in the "Properties > Events" inspector window.

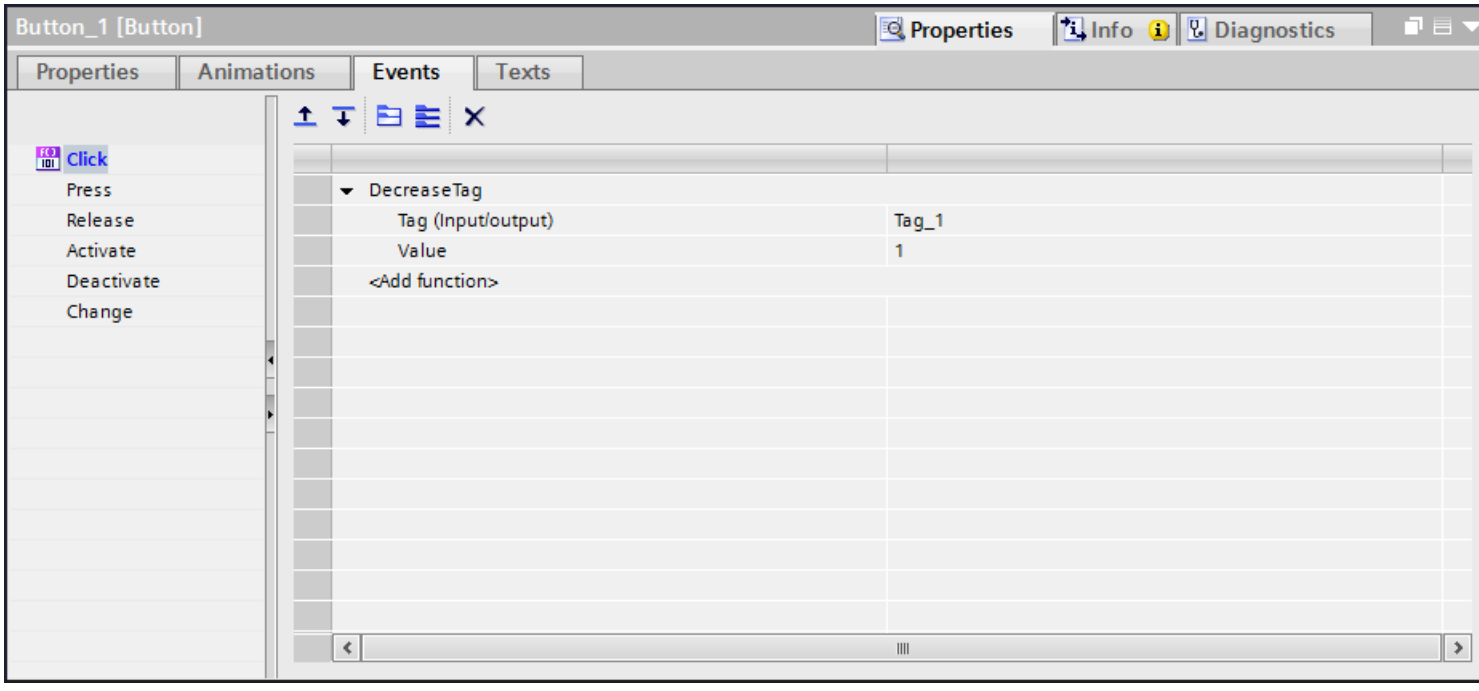

You can find more detailed information on this under "Working with Function Lists (Page 1379)".

See also

Local scripts (Page 1377)

Working with Function Lists (Page 1379)

1.4.1.5 Show overview of the dynamizations

Introduction

You can set the update cycle separately for every single dynamization. The number of different update cycles is limited to five per project.

Show overview of existing dynamizations

1. Select a screen in the project tree.
2. Select "Overview dynamizations" in the shortcut menu.

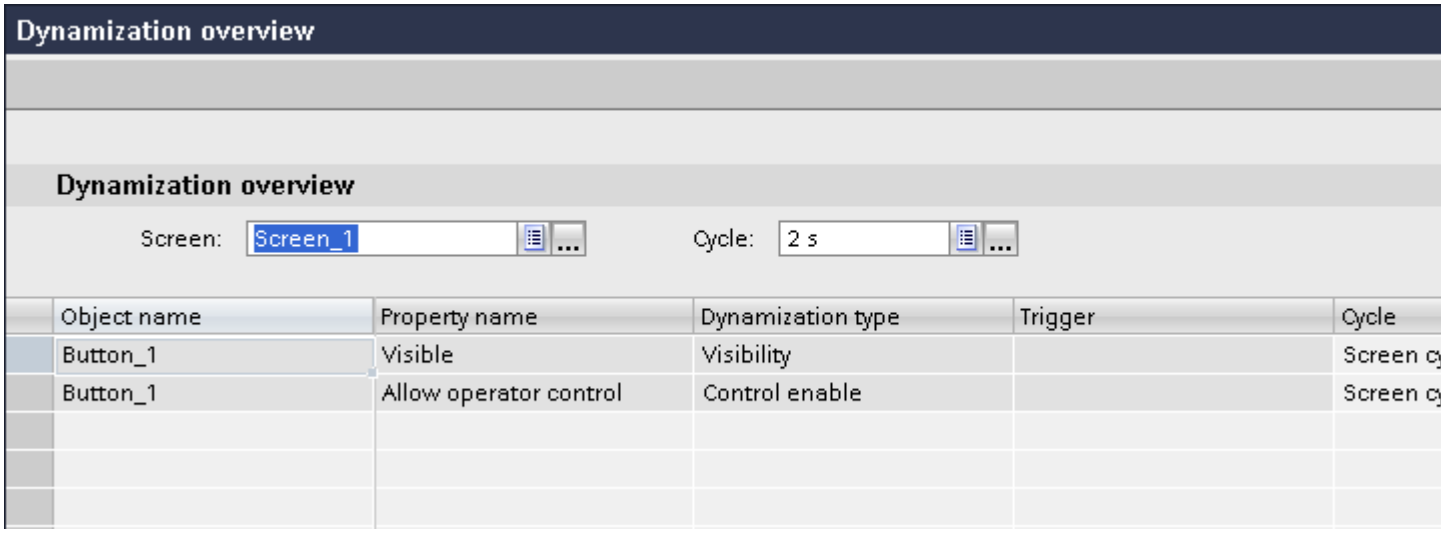

Result

All objects and their dynamizations are shown in the work area in the screen. Under "Cycle" you select another update cycle if necessary.

See also

Basics on dynamization in the property list (Page 232)

Basics on dynamizing screens (Page 205)

1.4.2 Dynamization with animations

1.4.2.1 Configuring a new animation (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Use predefined animations to dynamize screen objects.

Requirement

- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure in the inspector window

1. In the Inspector window, select "Properties > Animations".
2. Select the animation you want to use.
3. Click the button.

Procedure in the "Animations" task card.

1. Open the object group containing the required animation in the "Animations" task card.
2. Drag the animation onto the object that you want to make dynamic.

Alternatively you select the object in the screen and double click the desired animation in the "Animation" task card.

Result

The animation appears in the Inspector window of the object. You configure the animation in the following steps.

In the animations overview the green arrow indicates which animation is already configured. The configured animation opens in the inspector window when you click the green arrow.

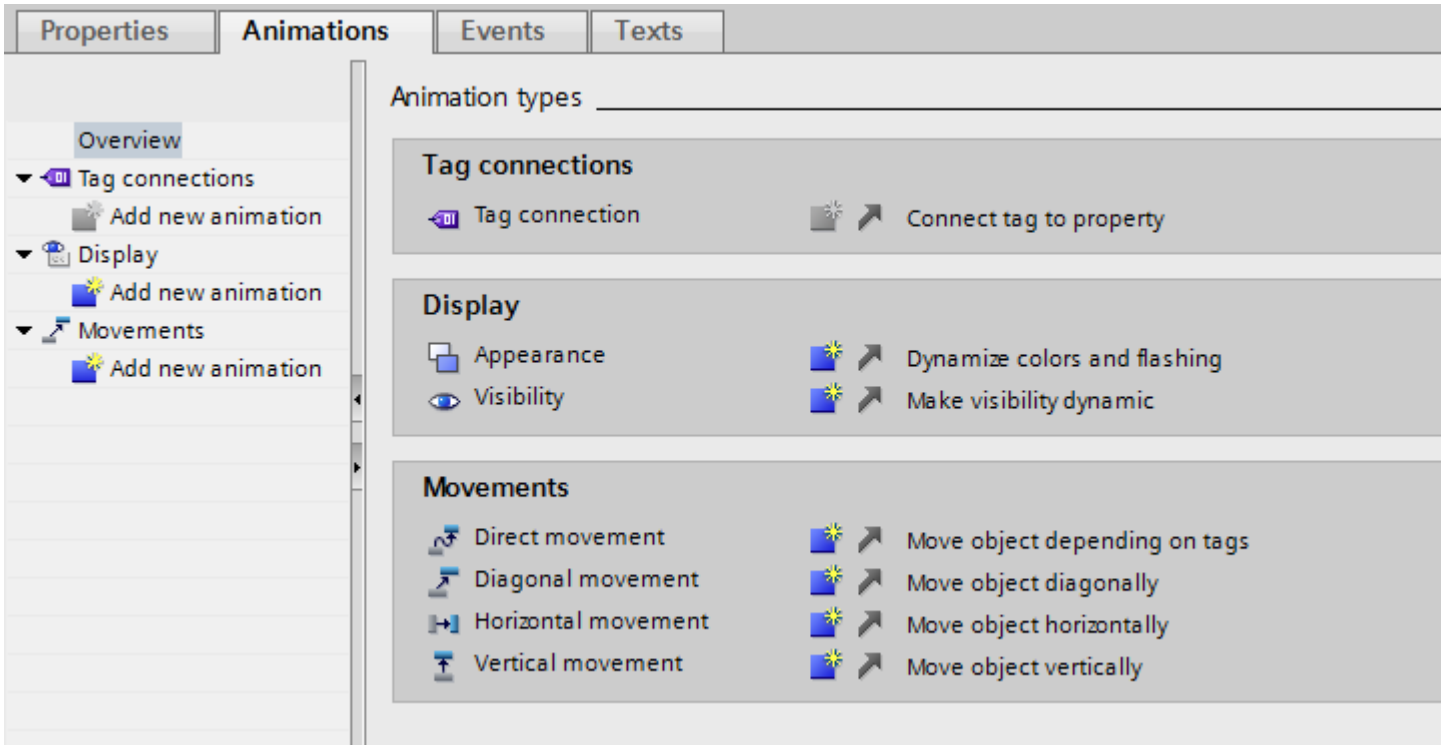

See also

Dynamizing the appearance of an object (Page 213)
Configuring movements (Page 215)
Configuring direct movement (Page 217)
Dynamizing control enable of an operating element (Page 218)
Dynamizing the visibility of an object (Page 219)
Configuring animation with tag connection (Page 221)
Animations in multiple selection (Page 224)
Basics on dynamizing screens (Page 205)
Animations of object groups (Page 223)
Dynamization with property animations (Page 225)

1.4.2.2 Dynamizing the appearance of an object (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The appearance of a screen object is controlled by changing the value of a tag in runtime. When the tag adopts a certain value, the screen object changes its color or flashing characteristics according to the configuration.

Type

Areas or single values of the tag are observed in Runtime depending on the selection. The appearance of the object changes according to the configuration.

Note

"Multiple bits" selection

If you use the "Multiple bits" option, also select "Bit selection for appearance" option in the runtime settings under "Screens > General".

Requirement

- A screen is open.
- A dynamic object is contained and selected in the screen.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure

1. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.
2. Select the animation "Appearance" and click the button.
   The parameters of the animation are displayed.
3. Select a tag in "Tag > Name".
4. Select "Type > Range" for example.
5. Click "Add" in the table.
6. Enter the tag interval "0 - 20" in the "Range" column for example.
7. For "Foreground color" and "Background color", select the color the object is to acquire in Runtime when the tag reaches the interval.

8. Select a flashing mode for the object from the "Flashing" list.
9. Repeat steps 5 to 8 to create another tag interval, e.g. "21 - 60".

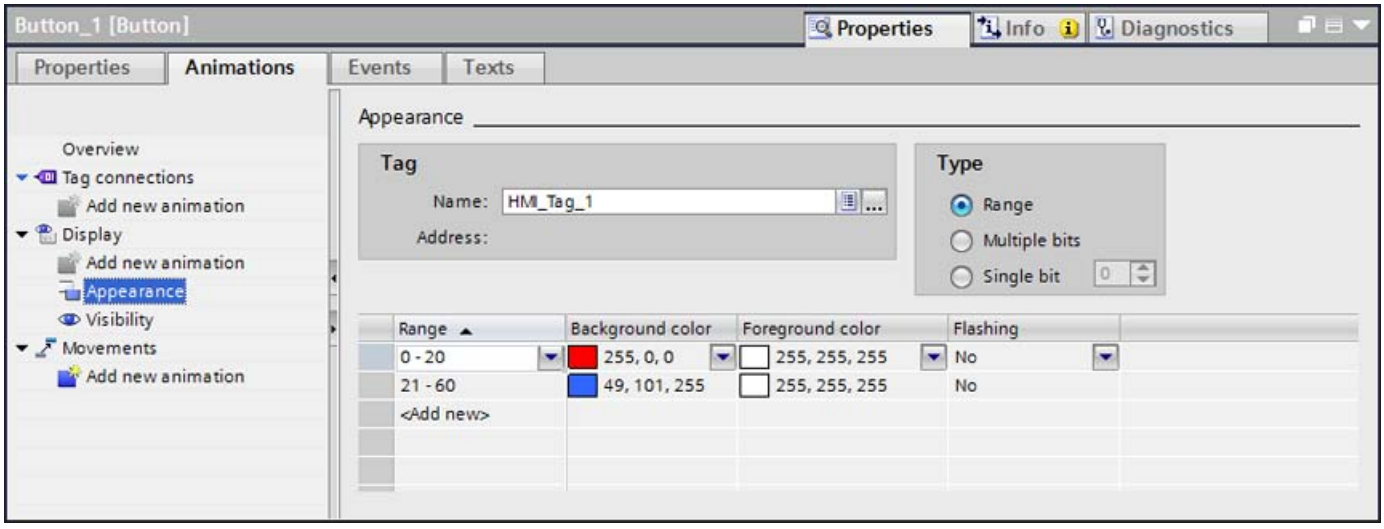

Result

In Runtime, the flashing response, and color of the object change dynamically according to the process value returned in the tag.

See also

Configuring a new animation (Page 212)

1.4.2.3 Configuring movements (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can configure dynamic objects in such a way that they move on a certain track. The movement is controlled via tags. The object moves every time the tag is updated.

You can only program one type of movement for each object.

Requirement

- You have created a tag.
- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure

1. Select the screen object you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.
3. Select "Horizontal movement" and click the button.
   The parameters of the animation are displayed.
   A transparent copy of the object is shown in the work area, which is connected to the source object by means of an arrow.
4. Select a tag for control of movement.
5. Move the object copy to the relevant destination. The system automatically enters the pixel values of the final position in the Inspector window.
6. Customize the range of values for the tag as required.

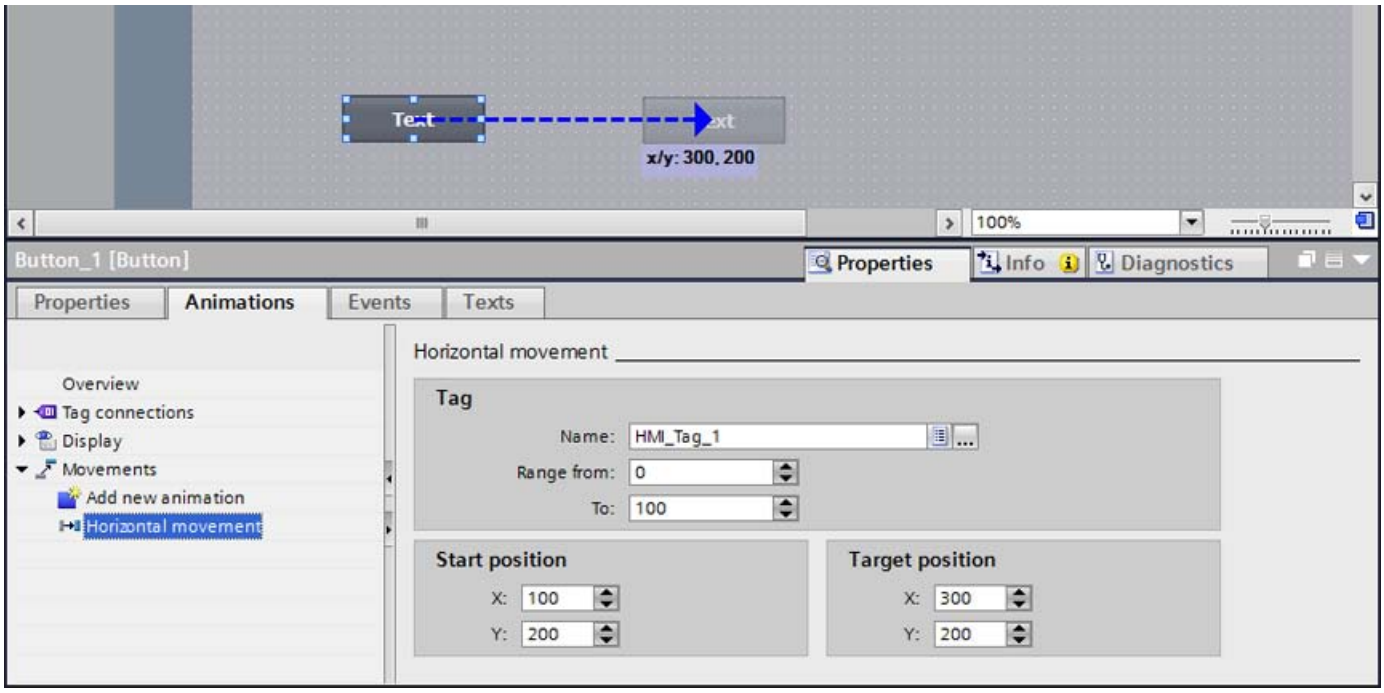

Result

In Runtime, the object moves in response to every change in value of the tag that is used to control the movement. The direction of movement corresponds to the configured type of movement "horizontal".

Note

You configure vertical and diagonal movements similar to horizontal movements

See also

Configuring a new animation (Page 212)

1.4.2.4 Configuring direct movement (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The object is moved respectively in x direction and y direction with "Direct movement". Two tags define the number of pixels by which the object moves from its original static start position.

Requirement

- Two tags are set up.
- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Configuring "Direct movement"

1. Select the screen object you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
3. Select "Direct movement" and click the button.
   The parameters of the animation are displayed.
4. Select a tag for the X position with which the movement in x direction is controlled.
5. Select a tag for the Y position with which the movement in y direction is controlled.

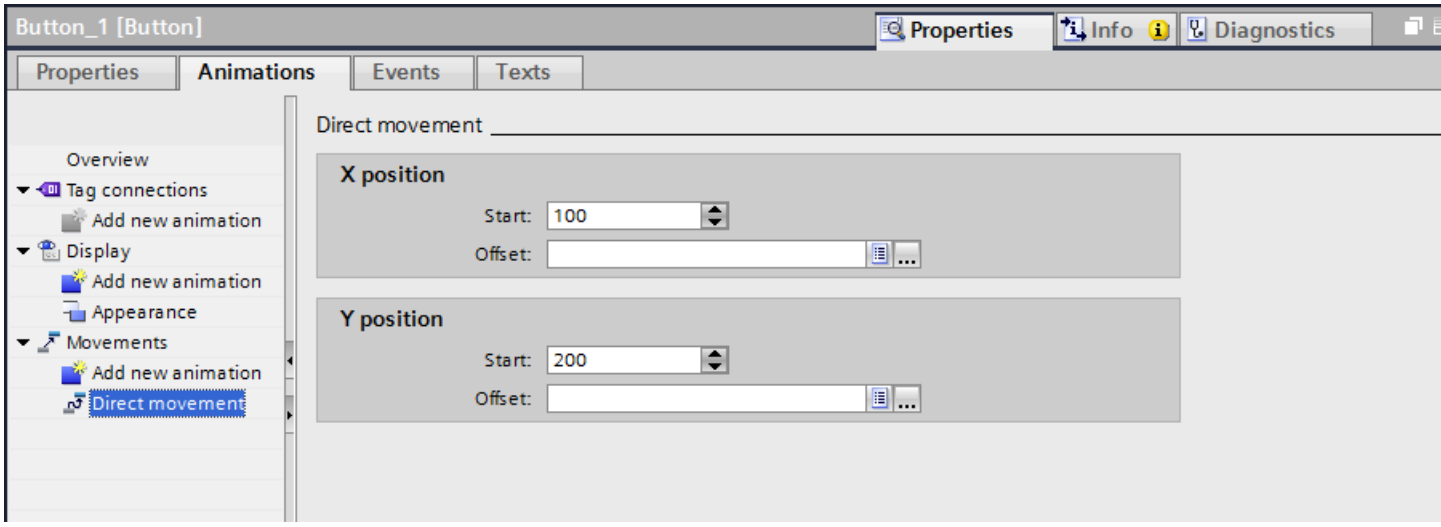

Result

In Runtime, the object moves in response to every change in value of the tag that is used to control the movement.

See also

Configuring a new animation (Page 212)

1.4.2.5 Dynamizing control enable of an operating element (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can dynamically change the usability of an operating element. The object either can, or cannot be used in Runtime, depending on the value of a tag.

You can use this dynamic control to restrict operator access to an operating element to specific situations, such as for maintenance.

Requirement

- You have created a tag.
- A screen with an operating element is open.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure

1. Select the operating element in the screen that you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.

3. Select "Control enable" and click the button.
   The parameters of the animation are displayed.

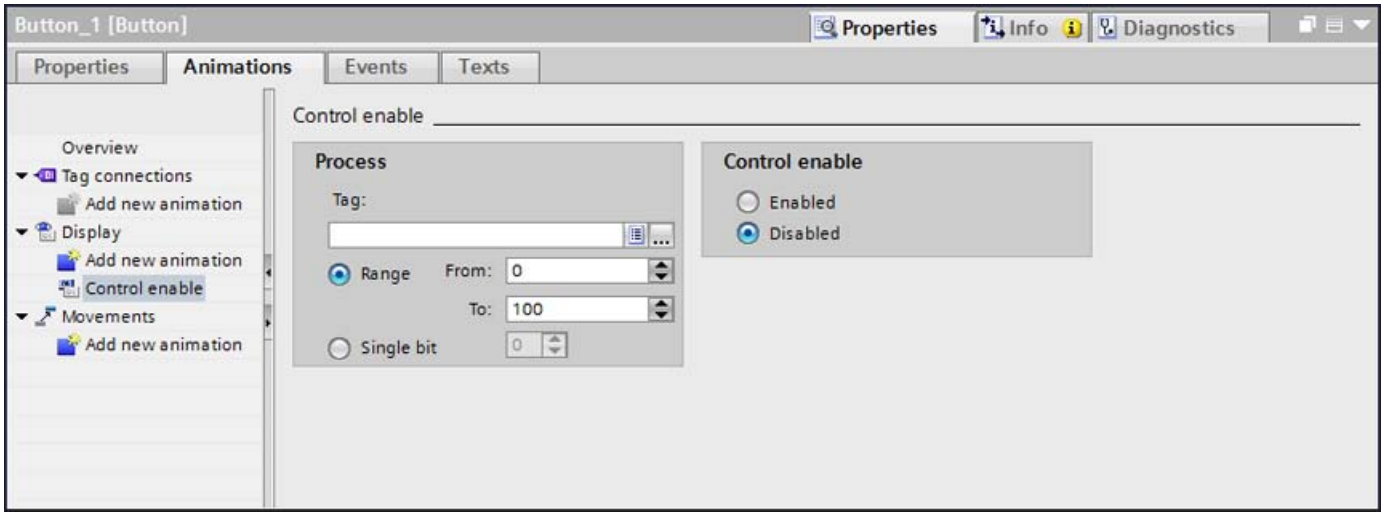

4. Select a tag.
5. Activate "Range".
   – For "From:" enter e.g. the value "0"
   – For "To:" enter e.g. the value "100"
6. Activate "Control enable > Activated".

Result

The screen object is operator control enabled in Runtime depending on the tag value:

- The object becomes operator control enabled when the tag value is between 0 and 100.
- The object is not operator control enabled if the tag value is outside the range.

See also

Configuring a new animation (Page 212)

1.4.2.6 Dynamizing the visibility of an object (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Dynamization of the "Visibility" property allows you to output an alarm to your screen, which is triggered when the tag value exceeds a critical range, for example. The alarm is cleared when the tag value returns to the non-critical range.

The "Simple recipe view" and "Simple alarm view" objects are always visible.

Requirement

- You have created a tag.
- You have opened a screen containing an object that you want to show or hide in Runtime.
- The Inspector window is open.

Procedure

1. Select the screen object you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.
3. Select "Visibility" and click the button.
   The parameters of the animation are displayed.
4. Select a tag.
5. Activate "Range".
6. Select, for example, "from 20" and "to 40"
7. Activate "Visible".

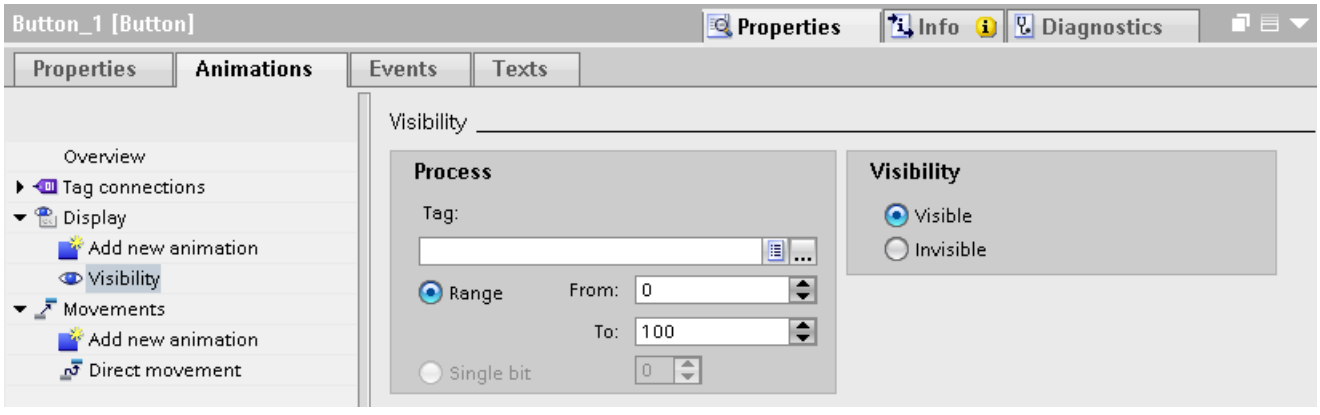

Result

The screen object is shown or hidden in Runtime depending on the tag value.

- If the tag value matches the configured range between 20 and 40, the screen object is shown.
- If the tag value is outside the configured range, the screen object is hidden.

See also

Configuring a new animation (Page 212)

1.4.2.7 Configuring animation with tag connection (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

When you object property an object property with a tag, the object property is changed in Runtime depending on the tag value.

The following example shows the tag binding in the "Properties > Animations" inspector window. If you have configured tag binding it is also visible in the property list or in the property pages in the inspector window.

Requirement

- An object with the property to be dynamized is selected.
- You have created a tag.
- The Inspector window is open.

Procedure

1. Select the screen object whose properties you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animation".
   The animations available for the selected object are displayed.
3. Select "Tag connection" and click the button.
   The parameters of the animation are displayed.
4. Select the object property that you want to dynamize in "Properties > Animation > Tag binding > Property > Name", e.g. "Background color".
5. Select a tag.

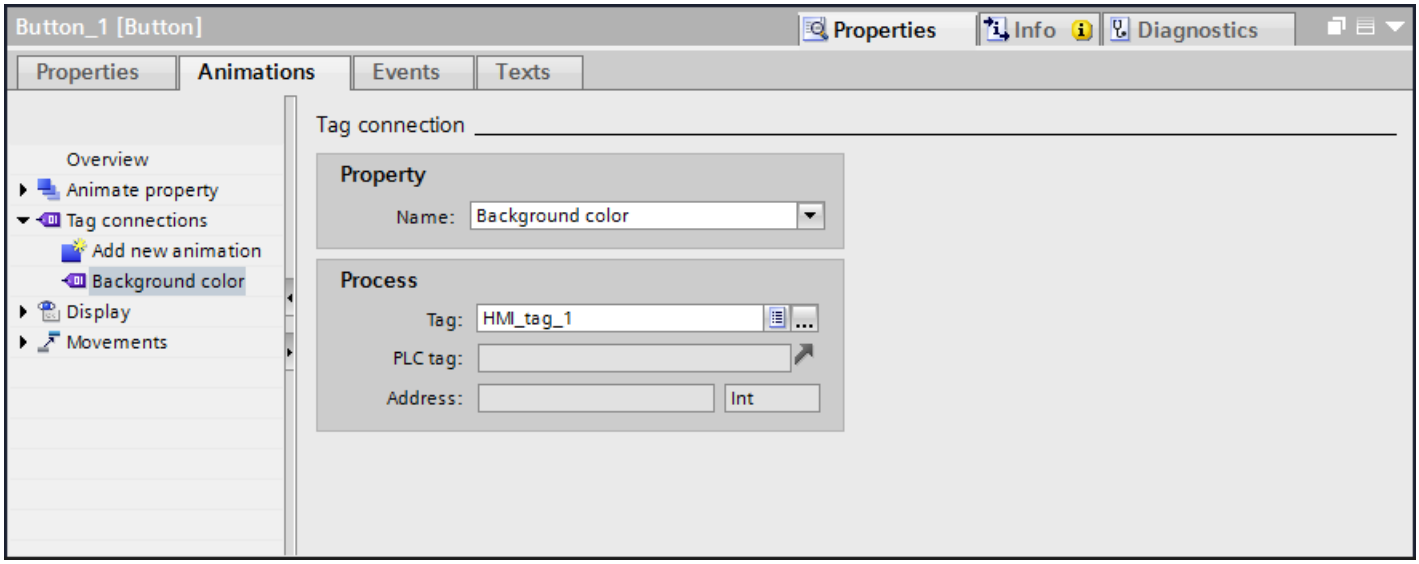

Result

You have dynamized the "Background color" property with a tag. The background color of the screen object changes in Runtime depending on which value the tag adopts.

See also

Configuring a new animation (Page 212)

1.4.2.8 Animations of object groups (Basic Panels)

Applying animations to object groups

The inspector window shows all objects of a group and their possible animations. The animation types which are supported by all objects in the group are also listed separately.

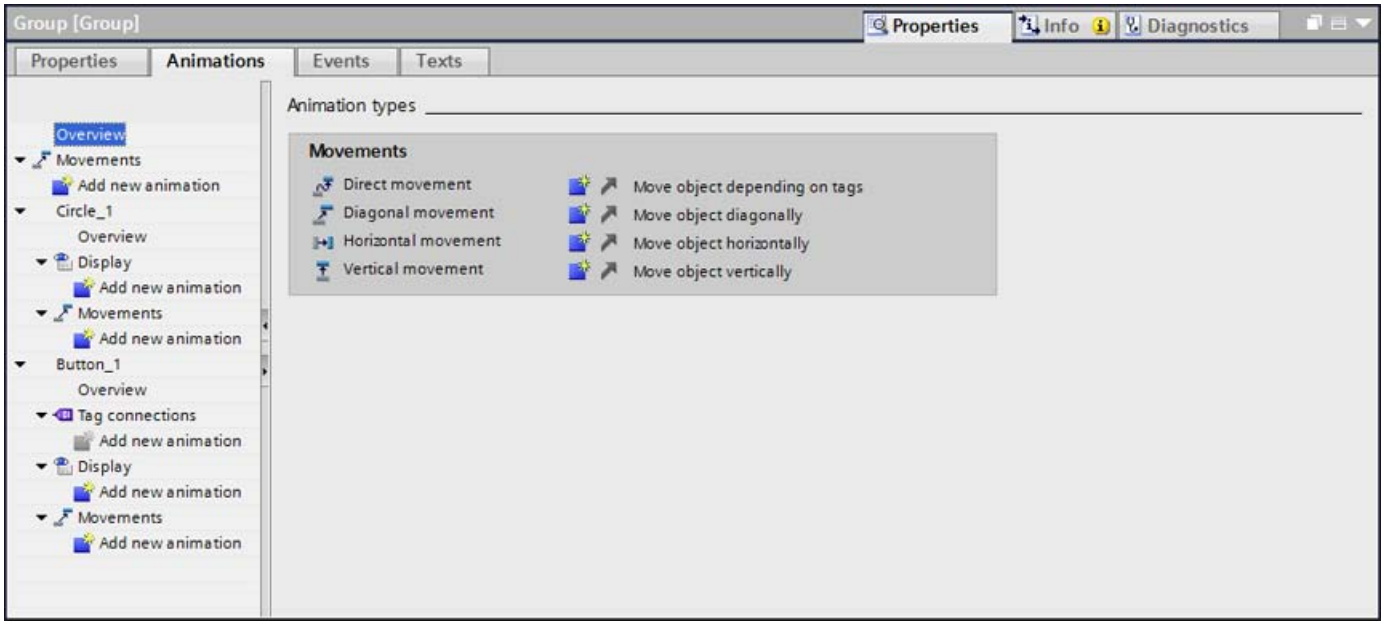

If you configure an animation for an object group, this animation will apply to all individual objects that support this animation.

The animation of nested groups is not supported.

Application example

The "horizontal movement" animation is configured for the object of an object group. The "direct movement" animation is configured for the whole object group. Only the animation of the object group i.e. "direct movement" is executed in Runtime. This also applies for object groups within object groups. Only the animation of the group on the top layer is listed.

See also

Configuring a new animation (Page 212)

1.4.2.9 Animations of object groups (RT Professional, RT Advanced, Panels, Comfort Panels)

Changing animations of object groups

The inspector window shows all objects of a group and their possible animations. The animation types which are supported by all objects in the group are also listed separately.

If you configure an animation for an object group, this animation will apply to all individual objects that support this animation.

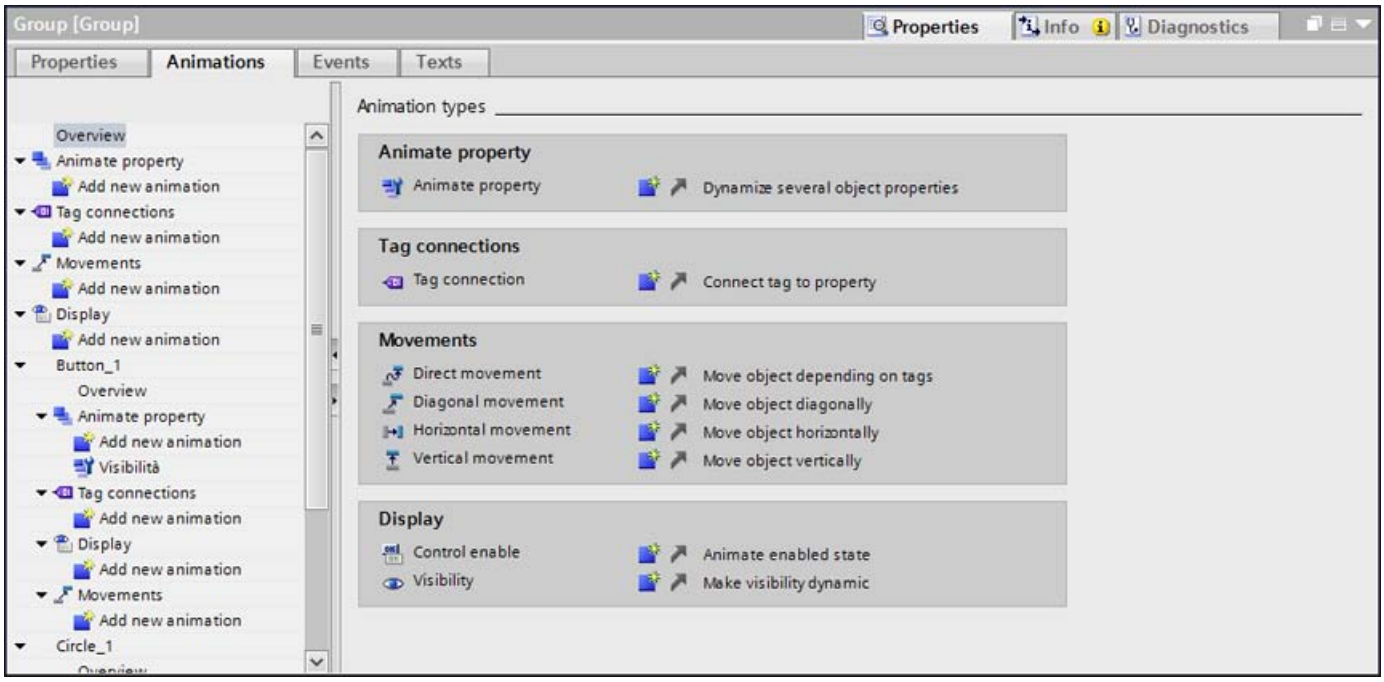

Application example

The "horizontal movement" animation is configured for the object of an object group. The "direct movement" animation is configured for the whole object group. Both animation types are executed in runtime. This also applies for object groups within object groups.

See also

Configuring a new animation (Page 212)

1.4.2.10 Animations in multiple selection (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Changing animations for multiple objects

For a multiple selection, the animations that are configured for the reference object appear in the Inspector window. You can change the animations as usual. The changes will apply to all the objects in the multiple selection that support the configured animation. This means that even objects for which you have not yet configured an animation will have the reference object's animation.

Application example

Select a button, and a circle at the same time. The button is the reference object. The "Appearance" animation is already configured for the button, so it appears in the Inspector window of the multiple selection. When you activate "Properties > Animations >Appearance > Flashing" in the inspector window, the settings of the "Appearance" animation apply for the button and for the circle.

Configuring new animations for multiple objects

If you configure a new animation for the objects of a multiple selection, this animation will apply to all selected objects that support the configured animation. If the new animation replaces an existing animation, a security prompt appears.

Application example

You select a circle, and a rectangle. The "Diagonal movement" animation is already configured for the circle. You configure the "Horizontal movement" animation for the multiple selection. This animation applies to the rectangle since no animation of the Movement type is yet configured for it. For the circle, you are asked to confirm that you want to replace the existing "Diagonal movement" animation with the new "Horizontal movement" animation.

See also

Configuring a new animation (Page 212)

1.4.3 Dynamization with property animation (RT Professional)

1.4.3.1 Dynamization with property animations (RT Professional)

Introduction

With the property animation you dynamize the properties of screens and screen objects depending on certain value ranges or tags. You select value ranges and determine how the linked object property changes when the tag adopts a certain value in Runtime. You have the option of dynamizing several object properties simultaneously, e.g. the position and the color of a screen object. You can also configure several property animations on one object.

Note

Dynamizing several object properties at the same time

You can dynamize multiple object properties with animation. The order of processing in runtime is not necessarily the same as the order of property configuration in the Inspector window under "Properties > Animation".

Note

If you use the operands "=", ">", "<", "!=" in the expression, note the following:
The value for "true" = -1
The value for "false" = 0

Please note that the runtime value of a Boolean tag "true" = 1 and = "false" = 0.

Type

The following types are available to define the value range:

- Range (integer)
  One or more value ranges of a tag are viewed. You configure different value ranges. You define the appropriate values of the object property for every value range. If the value is within a configured value range in Runtime, the linked object property is dynamized.
  Only define value ranges as integers. A non-integer tag value or expression is rounded up or down accordingly.
- Range (floating point)
  You configure different value ranges with floating point numbers. Enter each value range in square brackets and specify whether the range limit is included in this value range. If the value is within a configured value range in Runtime, the linked object property is dynamized.
- Multiple bits
  One or more bit numbers of a tag are viewed. You configure different bit numbers. The linked object property is only dynamized if the tag value in Runtime exactly matches one of the configured bit numbers. If the variable value does not match, the static value of the object property is displayed.

- Single bit
  You select the bit of tab which is to be monitored. If the bit is to be set, you configure the value "0". If the bit is not to be set, you configure the value "1".
- Bool (true, false)
  The Boolean tag or an expression which contains a logical comparison is viewed. If the tag or return value of the expression returns one of the two values - true or false - the value is assigned to the dynamized property.
- Direct
  Unlike the other types, the tag value or expression in the "Direct" type is written directly into the object property.

Expression

The "Expression" option is only available for the "Range", "Boolean" and "Direct" types.

You can formulate the expression by using tags, VB-scripts and arithmetic operators. The value of the expression is determined in Runtime and compared with the configured value ranges. The following selection of VBS keywords and constants is available within an expression:

```
"mod", "not", "and" "or","xor","eqv", "imp", "vbTrue", "vbFalse"
```

Evaluation "Quality Code"

Use "Quality Code" to check the status and quality of a tag. The quality of the entire value transmission and value processing of the respective tag is combined in the displayed "Quality Code". The monitoring of the "Quality Code" enables you to draw conclusions about the quality of the tags in the process.

The evaluation of the "Quality Code" has priority over the defined value ranges.

If you use several tags, the "Quality Code" of the tags is evaluated in the order in which they appear from left to right in the expression. The configured values are only viewed if the "Quality Code" of the tag is correct.

The figure below shows a selection of the offered "Quality Codes". The "Quality Codes" are sorted in descending order of priority in this list.

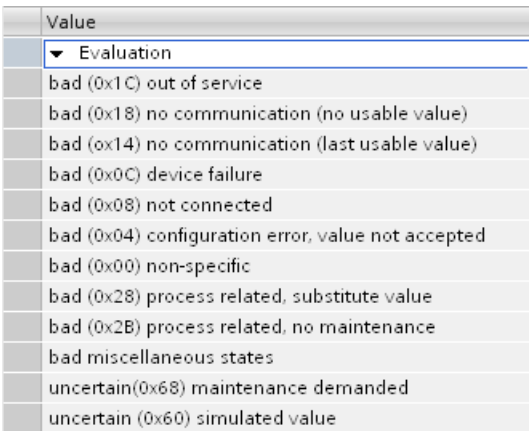

See also

Configuring a new animation (Page 212)
Configuring property animation of the Range type (Page 227)
Configuring property animation of the Direct type (Page 228)
Configuring property animation of the Bool type (Page 229)

1.4.3.2 Configuring property animation of the Range type (RT Professional)

Introduction

When you dynamize an object property with a tag, the object property is changed in Runtime depending on the tag value. The following example shows the dynamization of a circle. The circle changes color and position depending on a tag.

Requirement

- You have created a "Tag1" tag.
- You have created a screen.
- You have inserted and selected a circle in the screen.
- The Inspector window is open.

Select animation type, tag and type

1. Click "Properties > Animation" in the Inspector window.
   The animations available for the selected object are displayed.
2. Select "Property animation" and click the button.
   The parameters of the animation are displayed.
3. Enter the name "Light" under "Settings > Name".
4. Select "Range" as type".
5. Activate "Process > Tag".
6. Select "Tag1" under "Process > Tag".

Select object properties and ranges

1. Click "Add property". A drop-down list opens.
2. Select the property "Y position". A column "Y position" is added to the table.
3. Click "Add property". A drop-down list opens.
4. Select the property "Color background". A column "Color background" is added to the table.
5. Click "Add" in the table.
6. Define the first range from "0 - 49".

7. Click "Add" in the table.
8. Define the second range from "50 - 99".
9. Click "Add" in the table.
10. Define the second range from "100 - 149".

Assign the background color and position to a range

Define the following settings for the ranges:

- Range 0 - 49: Y position "200"; color "green"
- Range 50 - 99: Y position"220"; color "yellow"
- Range 100 - 149: Y position "240"; color "red"

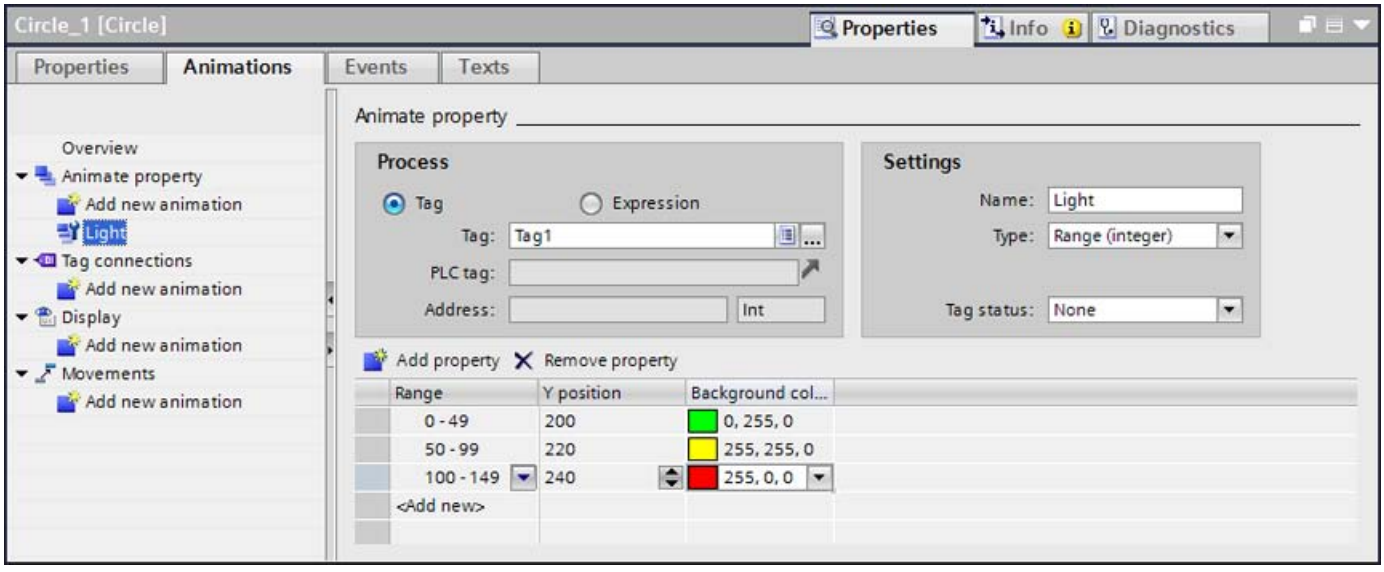

Result

Depending on the value range in which the tag is located in Runtime, the circle changes its position and background color.

See also

Dynamization with property animations (Page 225)

1.4.3.3 Configuring property animation of the Direct type (RT Professional)

Introduction

When you dynamize an object property with a tag, the object property is changed in Runtime depending on the tag value. The following example shows the dynamization of an I/O field. The I/O field changes its output value depending on a tag.

Requirement

- You have created a "Tag1" tag.
- You have created a screen.
- You have inserted and selected an I/O field with the "Output" mode in the screen.
- The inspector window is open.

Select animation type, tag and type

1. Click "Properties > Animation" in the inspector window.
   The animations available for the selected object are displayed.
2. Select "Property animation" and click the button.
   The parameters of the animation are displayed.
3. Enter the name "Fill level" under "Settings > Name".
4. Select "Direct" as type".
5. Activate "Process > Tag".
6. Select "Tag1" under "Process > Tag".

Result

The tag value in Runtime is written directly into the I/O field.

See also

Dynamization with property animations (Page 225)

1.4.3.4 Configuring property animation of the Bool type (RT Professional)

Introduction

When you dynamize an object property with a tag, the object property is changed in Runtime depending on the tag value. In the following example, a button is shown and hidden depending on the value of an expression.

Note

Boolean tags for the "Bool" type are implicitly converted to the VB script values "true" = -1 and "wrong" = 0 using the VB script function, CBool.

Use the constants "`vbTrue`" and "`vbFalse`" for a unique comparison in the expression.

Requirement

- A "TCelsius" tag of the "Uint" type is created.
- You have created a screen.
- You have inserted and selected a button in the screen.
- The Inspector window is open.

Select animation type, expression and type

1. Click "Properties > Animation" in the Inspector window.
   The animations available for the selected object are displayed.
2. Select "Property animation" and click the button.
   The parameters of the animation are displayed.
3. Enter the name "Visibility" under "Settings > Name".
4. Select "Boolean" as type".
5. Activate "Process > Expression".
6. Enter the following expression in the field: "(('TCelsius' × 9) / 5) + 32 < 100".

   Note

   Make sure to write the tag name in single inverted commas in the direct input.

   Note

   Boolean tags can be evaluated simplified:
   "'Booltag1' AND NOT 'Booltag2'".

Alternatively, enter the expression with the buttons above the field. For example, click the button and select a tag from the selection list.

Select object property

1. Click "Add property". A drop-down list opens.
2. Select the "Visibility" property. A "Visibility" column is added to the table.
3. Activate the "Visibility" property for the "True" value in the table.

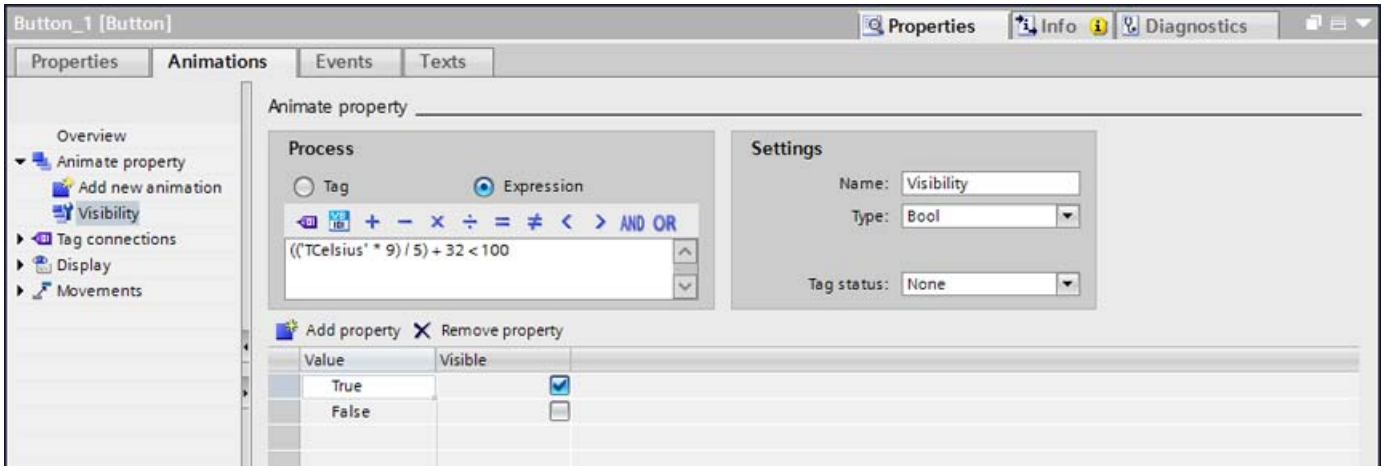

Result

The value of the expression is determined in runtime. The button is displayed if the value matches the configured "True" value.

See also

Dynamization with property animations (Page 225)

1.4.4 Dynamization in the property list (RT Professional)

1.4.4.1 Basics on dynamization in the property list (RT Professional)

Basic principle of dynamization

In the case of an object property that can be dynamized you first specify the type of dynamization. The following figure shows the types of dynamization available:

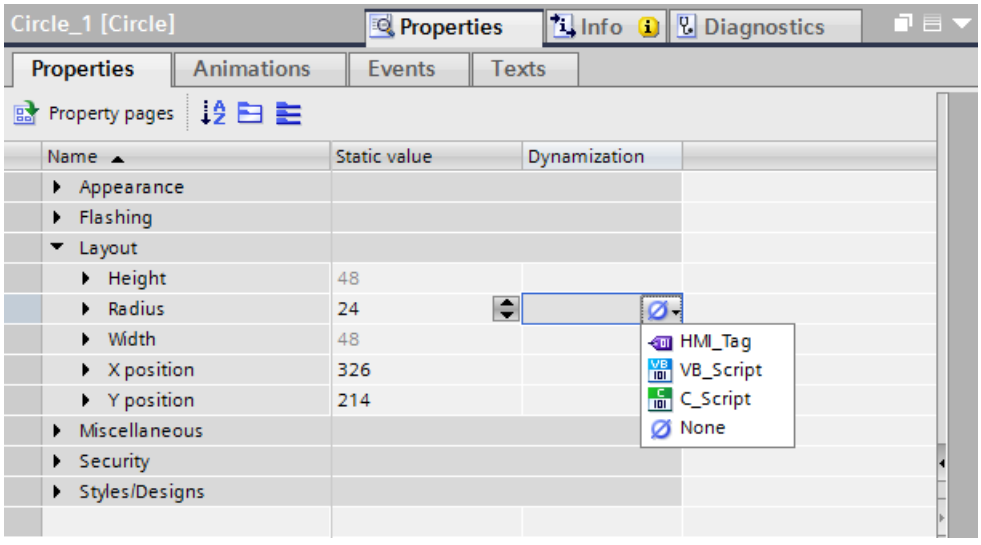

Depending on the type of dynamization you either select a tag using the object list or you create a local script.

Dynamization with tags

If you dynamize an object property with a tag, you can configure a function list additionally on the following events:

- Changed
- Tag status changed
- Quality code changed

The tags are updated in the update cycle set for the screen.

The figure below shows the "Radius" object property dynamized with a tag. When the value of the tag changes, a function list is executed additionally:

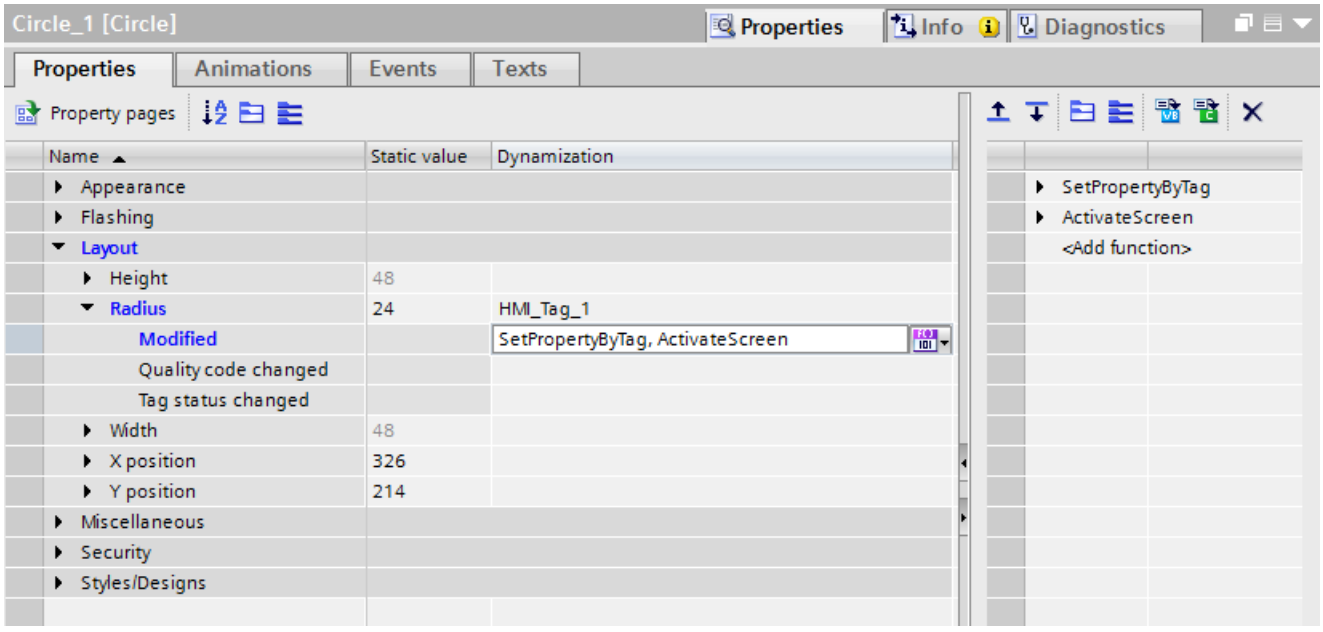

You can also convert the function list into a local script.

Quality Code

Use "Quality Code" to check the status and quality of a tag. The quality of the entire value transmission and value processing of the respective tag is combined in the displayed "Quality Code". The monitoring of the "Quality Code" enables you to draw conclusions about the quality of the tags in the process.

Dynamization with local scripts

With a local script you dynamize an object property in Runtime by means of the return value of the local script. The script is created directly at the instance where it is used. The following figure shows the implementation of a status indicator by dynamizing the background color using a VB script:

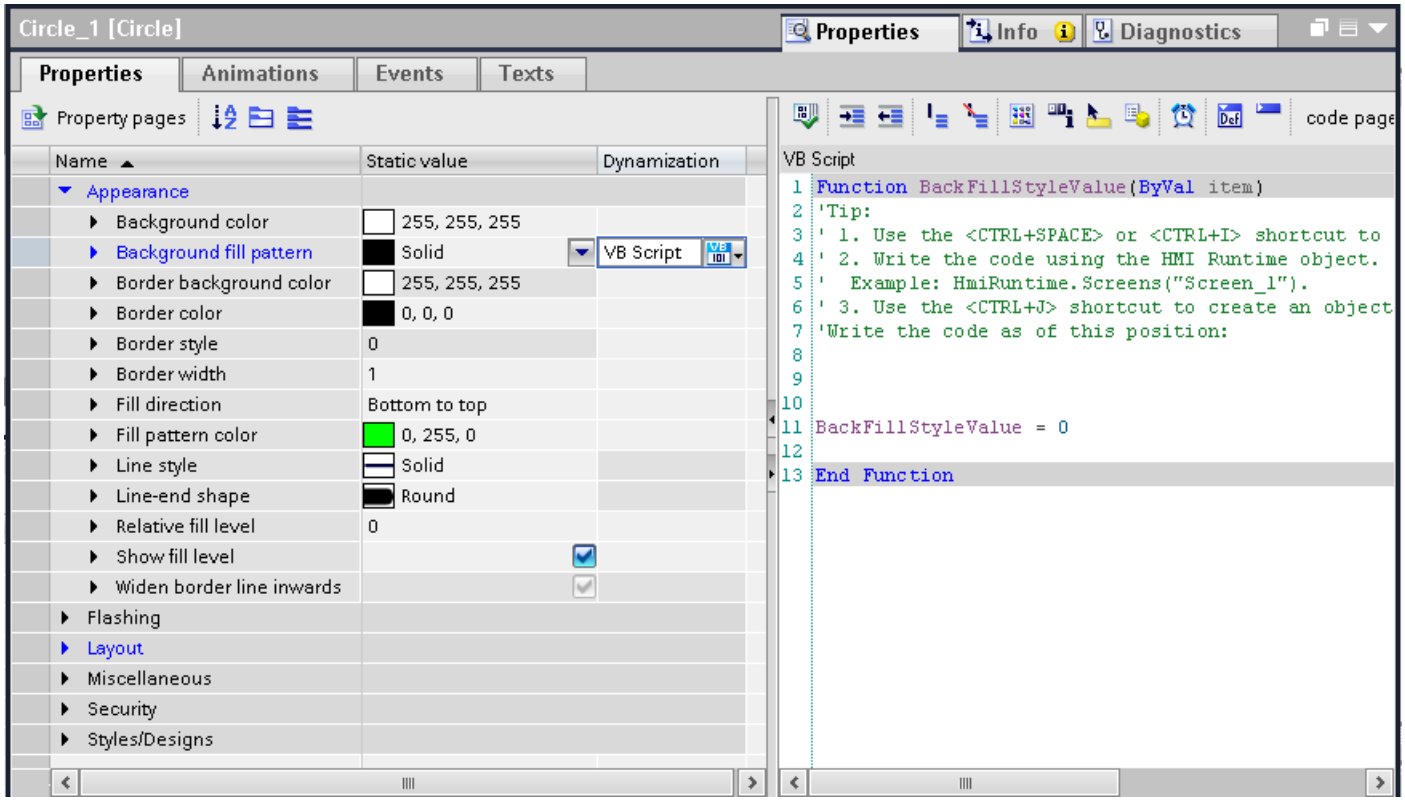

See also

Dynamizing an object property with a tag (Page 235)

Programming a dynamization method for a tag change (Page 237)

Dynamizing an object property with local scripts (Page 238)

Show overview of the dynamizations (Page 211)

Basics on dynamizing screens (Page 205)

Local scripts (Page 1419)

1.4.4.2 Dynamizing an object property with a tag (RT Professional)

Introduction

When you dynamize an object property with a tag, the object property is changed in Runtime depending on the tag value. When you indirectly dynamize the object property, you connect the property to a tag that sets the property value in Runtime.

Requirement

- An object with the property to be dynamized is selected.
- You have created a tag.
- The Inspector window is open.

Procedure

1. Select the screen object whose properties you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. Click the button in the Inspector window.
   The property list is displayed with the properties of the selected object.
3. Select the property to be dynamized.
4. In the "Dynamization" column, select the entry "HMI_Tag".
   A dialog opens.

5. Select a tag.
6. Activate "Use indirect addressing" to dynamize the object property.

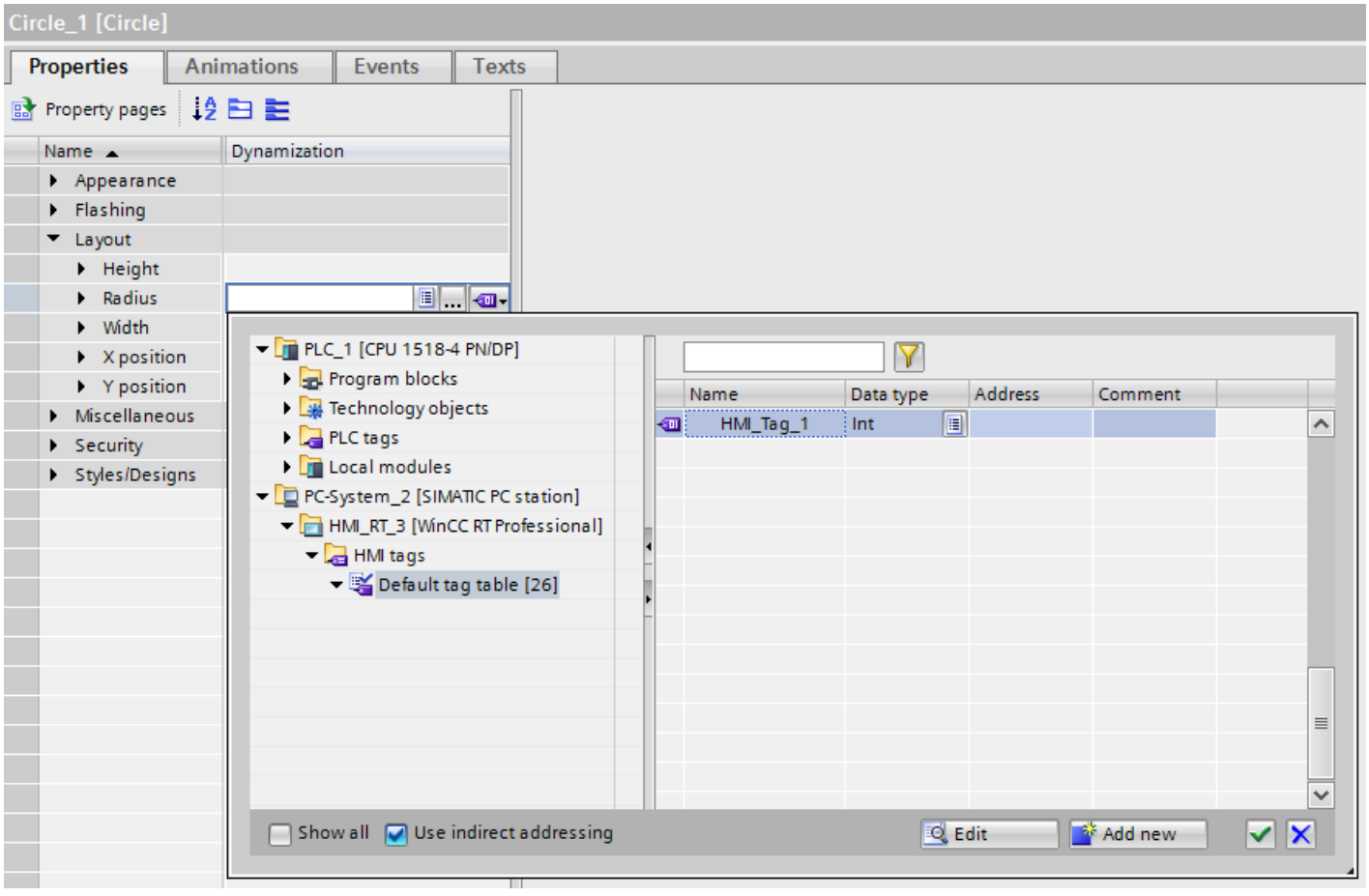

Result

The object property changes in Runtime according to how the property list is configured.

Representation of a multiple selection in the property list

If you selected multiple objects at the same time, the "Object type" column is added to the property list. The objects that support the property displayed in the row are displayed in this column. Changes to, or dynamization applied to a property affect all the objects displayed in the "Object type" column.

See also

Basics on dynamization in the property list (Page 232)

Basics on dynamizing screens (Page 205)

1.4.4.3 Programming a dynamization method for a tag change (RT Professional)

Introduction

If you have entered a tag in the "Dynamization" column, you can configure a function list additionally on different events.

Requirement

- An object with the property to be dynamized is selected.
- You have created a tag.
- The property list is open in the inspector window.
- An object property is connected with the tag.

Procedure

1. Extend the property list by clicking next to the property which you have dynamized with a tag.
2. Three other lines with the following events are displayed below the property.
   - Changed
   - Tag status changed
   - Quality code changed
3. Configure a function list for the required event.
   Alternatively you can convert the function list into a local script.

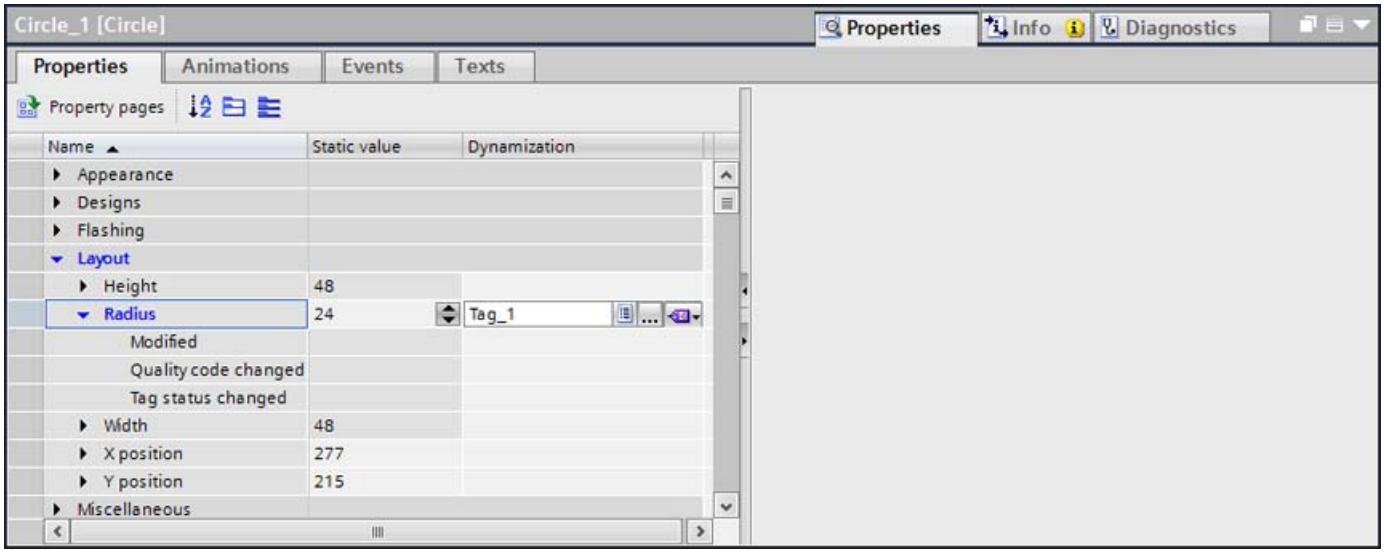

Result

The object property changes in Runtime according to how the property list is configured.

See also

Basics on dynamization in the property list (Page 232)

1.4.4.4 Dynamizing an object property with local scripts (RT Professional)

Introduction

With a local script you dynamize an object property in Runtime by the return value of the script.

Requirement

- The object with the property to be dynamized is selected.
- The Inspector window is open.

Procedure

1. Click in the Inspector window.
2. Select the object property to be dynamized.
3. Select the "VB script" or "C script" entry in the "Dynamization" column.
   A dialog opens.
4. Write the program code.
5. Click " " in the toolbar to change the trigger.

Result

The object property changes in Runtime according to how the property list is configured.

Deleting the dynamization

1. Select the dynamized object property in the property list.
2. Select the "None" entry in the "Dynamization" column.

See also

Basics on dynamization in the property list (Page 232)

Basics on dynamizing screens (Page 205)

Creating local scripts (Page 1419)

1.4.5 Dynamizing with system functions

1.4.5.1 Basic on events

Introduction

Screen objects react to events. You configure a function list with system functions on the events of an object.

Events

Which events and system functions are available depends on the object used.

If the operator activates a screen object for example, the configured system function is executed.

You can find additional information at "Basics on Function Lists (Page 1379)".

See also

Basics on dynamizing screens (Page 205)

Configure system function on the "Click" event (Page 239)

Example: Configuring a button for language switching (Page 240)

Working with Function Lists (Page 1379)

1.4.5.2 Configure system function on the "Click" event (Basic Panels)

Introduction

You configure a function list on an object event. The linked system function is executed when the event occurs in runtime.

Requirements

A screen is open.

A button has been created in the screen.

The inspector window is open.

Procedure

1. Select the button.
2. Click "Properties> Events" in the Inspector window.
3. Select the "Click" event.

4. Click "Add function" in the table.
5. Select the "ShowAlarmWindow" system function.

Result

The alarm window opens in the screen if the operator clicks the button in runtime.

See also

Basic on events (Page 239)
Basics of the function list (Page 1379)

1.4.5.3 Example: Configuring a button for language switching

Introduction

In this example, you configure a button that can be used to toggle between multiple runtime languages during runtime.

Requirements

- You have completed the "Configuring a button in multiple languages" example.
- The "Screen_1" screen is open.
- The button on the screen has been selected.

Procedure

1. In the Inspector window, select "Properties > Events > Press".
2. Click on "Add function" in the table.
3. Select the "SetLanguage" system function and the "Switch" setting.

Result

You have assigned the button the function "SetLanguage". Pressing the button during runtime will switch the runtime language. The runtime languages are switched in the order specified by the number sequence in the "Languages and fonts" editor.

See also

Basic on events (Page 239)

1.5 Working with function keys (RT Advanced, Basic Panels, Panels, Comfort Panels)

1.5.1 Working with function keys (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

The function key is a physical key on your HMI device and its functions can be configured. A function list can be configured for the events "Key pressed" and "Release key".

A function key can be assigned global or local functions.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Global function keys

Global function keys always trigger the same action, regardless of the displayed screen.

Global function keys are configured in the "Global Screen" editor. The global assignment applies for all the screens of the set HMI device.

Global function keys reduce programming considerably, because there is no need to assign these global keys to each individual screen.

Local function keys in screens

Local function keys in screens can trigger a different action in each screen. This assignment applies only to the screen in which you have defined the function key.

Within a screen, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Note

If a screen with local function keys is overlapped by an alarm view or an alarm window, then the function keys are still active in runtime. This may happen in particular on HMI devices with a small display.

Local function keys in templates

Local functions keys that are assigned in templates are valid for all the screens based on this template. They can trigger a different action in every screen. Function keys for templates are assigned in the template of the "Screens" editor. Within a template, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Hotkey assignment

You can assign hotkeys, such as buttons, to control objects. The available hotkeys depend on the HMI device.

Note

The function key has a local or global action assigned to it. If the function key also has a hotkey assigned to it, then the hotkey function will be executed in Runtime.

Graphics

When a function key is placed directly next to the display, you can assign a graphic to it to make the function of the function key more clear.

Display of assignment

Table 1-1 The following table shows which symbols display the assignment of the function keys:

Function key	Description
	Not assigned
	Global assignment
	Assigned locally in the template
	Local assignment
	Local assignment (local assignment of the template overwrites global assignment)
	Local assignment (local assignment overwrites global assignment)
	Local assignment (local assignment overwrites local assignment of the template)

Function key	Description
	Local assignment (local assignment overwrites local assignment of the template, which already overwrites global assignment)
	Assigning buttons with screen navigation

Note

Basic Panels

The "Screen Navigation" editor is not available for Basic Panels.

See also

Assigning function keys globally (Page 243)

Local assignment of function keys (Page 245)

Assigning a function to a function key (Page 246)

Assigning operator authorization for a function key (Page 248)

Assigning a graphic to a function key (Page 250)

Example: Using function keys for screen navigation (Page 254)

1.5.2 Assigning function keys globally (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You define the global assignment of a function key in the "Global Screen" editor. The global assignment applies to all screens of the set HMI device.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

- You have opened the project.
- The Inspector window is open.

Procedure

Proceed as follows to assign a screen-based function to a function key:

1. To open the "Global Screen" editor, double-click "Global Screen" in the "Screen management" group of the Project window.
2. Select the desired function key.
   The properties of the function key are shown in the Inspector window.

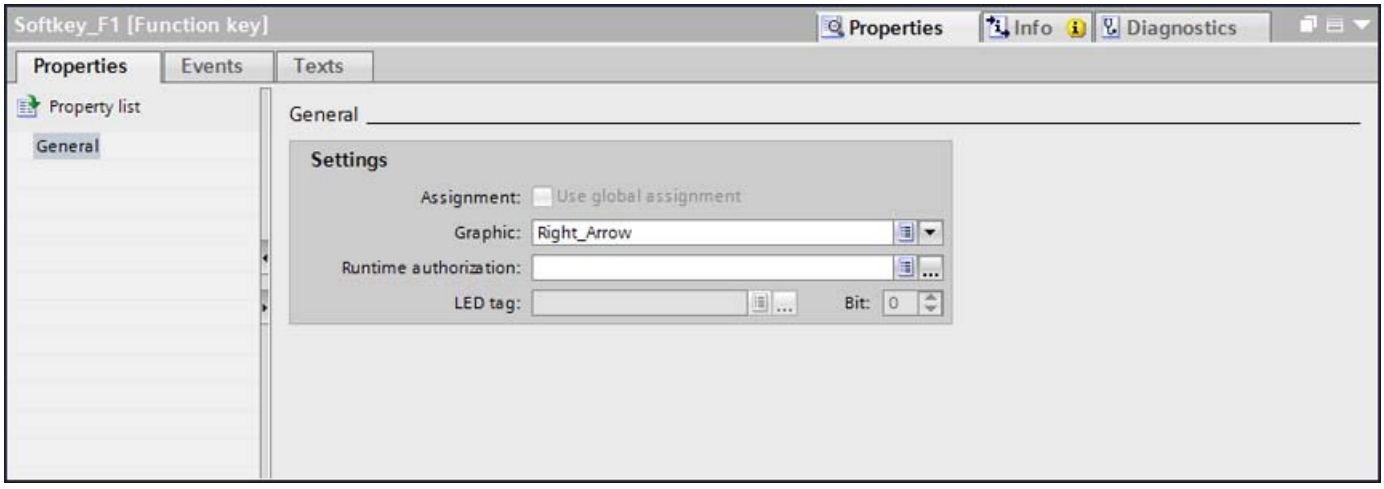

3. Under "General", configure a graphic and an operator authorization for the function key.
4. Configure a function list for the required event under "Events".

Result

If a local assignment does not overwrite the global assignment, the assignment of the function key changes in all the screens of the set HMI device in accordance with your entry.

See also

Working with function keys (Page 241)

1.5.3 Local assignment of function keys (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Function keys are assigned globally and locally. A local assignment of the function keys is only valid for the screen or the template in which it was defined. The following local function keys are available:

- Local function keys of a screen
  Different functions are assigned to the function key for each screen. This assignment applies only to the screen in which you have defined the function key.
- Local function keys of a template
  You assign the function keys in a template. The assignment applies to all the screens that are based on this template and are not overwritten by a local assignment in a screen.

A local assignment overwrites the global assignment of a function key.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Using existing assignments

The option for using existing assignments is referred to as follows in the Inspector window:

- In a template: "Use global assignment"
- In a screen:
  - Screen based on a template: "Use local template"
  - Screen not based on a template: "Use global assignment"

The "Use local template" option includes the use of the local assignment in the template and the global assignment.

Requirement

- You have opened the screen or the template in which you want to assign a function key locally.
- The Inspector window is open.

Procedure

Proceed as follows:

1. Select the desired function key in the screen or in the template.
   The properties of the function key are shown in the Inspector window.
2. Click "General" in the Inspector window.

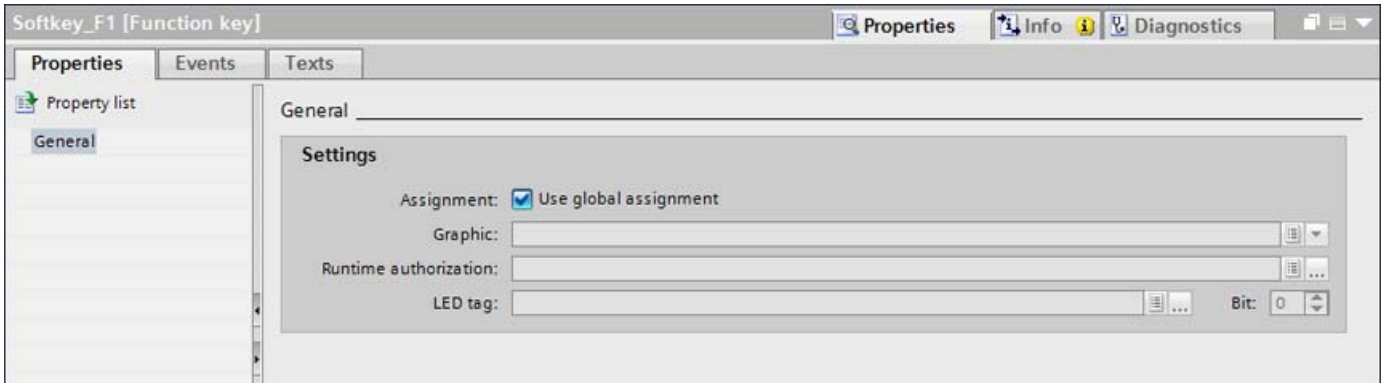

3. Disable the "Use local template" or "Use global assignment" option.
4. Under "General", configure a graphic and an operator authorization for the function key.
5. Configure a function list for the required event under "Events".

Result

The function key is assigned the configured function in the screen or in the template.

See also

Working with function keys (Page 241)

1.5.4 Assigning a function to a function key (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

A function key can have two states:

- Pressed: Defined by the "Key pressed" event.
- Released: Defined by the "Release key" event.

Both of these events are configured in the Inspector window of the function key. You can assign any event a function list which contains system functions or scripts. Execution of this function list is event-driven in runtime.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Note

Basic Panels

Scripts are not available for Basic Panels.

Requirement

To assign the function key a global function:

- The "Global Screen" editor is open.

To assign the function key a local function:

- The screen in which you want to assign a function key is open.

If you want to assign a function key locally in a template:

- The template in which you want to assign a function key is open.
- The Inspector window is open.

Procedure

Proceed as follows:

1. Select the function key you want to define.
   The properties of the function key are shown in the Inspector window.
2. Configure the function list for the desired result in the Inspector window under "Properties" in the "General" group.

Result

The function list is executed in runtime when the operator presses or releases the function key.

See also

Working with function keys (Page 241)

1.5.5 Assigning operator authorization for a function key (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

In WinCC you can assign an operator authorization for a function key in runtime. This allows you to restrict access to the function keys to specific persons or operator groups when you configure your project. Only authorized personnel can then change important parameters and settings in runtime.

You configure access protection in order to avoid operator errors and increase plant or machine security.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

- The user groups have been defined.

To protect a global function key:

- The "Global Screen" editor is open.

If you want to protect a local function key of a screen or of a template:

- The screen or the template which contains the function key is open.
- The Inspector window is open.

Procedure

Proceed as follows:

1. Select the relevant function key.
   The properties of the function key are shown in the Inspector window.
2. Click "General" in the Inspector window.

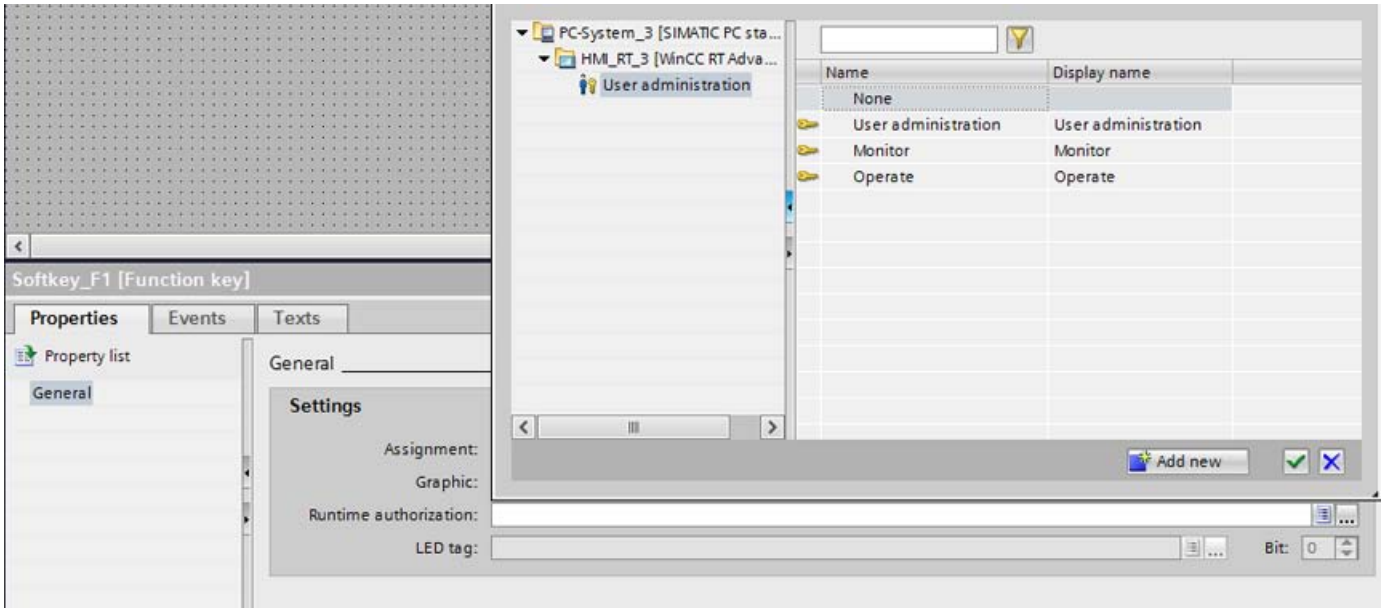

3. In the "Authorization" list, select the user group you want to allow runtime access to the function key.

Result

The operator authorization is configured.

See also

Working with function keys (Page 241)

1.5.6 Assigning a graphic to a function key (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

In order to make the function of a key more clear, you can insert a graphic in the screen alongside the function key. Graphics can only be assigned to function keys that border the screen margin of the HMI device.

Note

HMI device dependency

Function keys are not available on all HMI devices.

Note

Assigning graphics to a function key

A graphic can only be assigned to a function key if the bottom edge of the permanent window does not conceal the area of the function key graphic.

Requirement

To assign a graphic to a global function key:

- The "Global Screen" editor is open.

If you want to assign a graphic to a local function key in a screen or template:

- The screen or the template that contains the corresponding function key is open.
- The Inspector window is open.
- You have created the graphic for the function key.

Procedure

Follow these steps:

1. Select the relevant function key.
   The properties of the function key are shown in the Inspector window.
2. Click "General" in the inspector window.

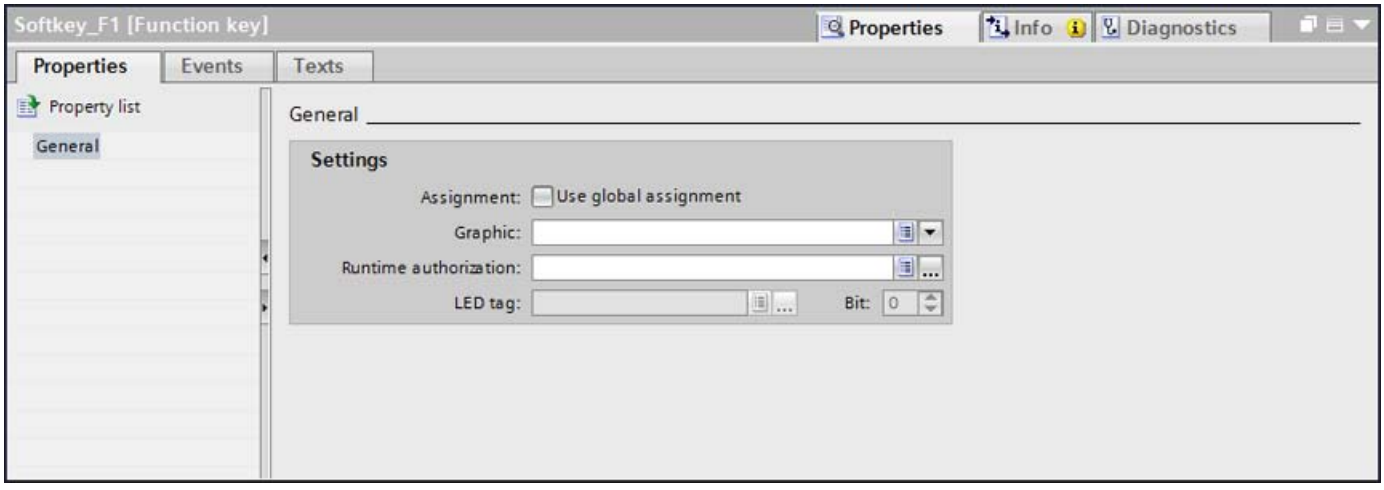

3. Click in the "Graphic" area of the list.
   The graphic browser of your WinCC project appears. The pane on the left side shows the external graphics which are stored in the browser. The pane on the right side shows you a preview of the graphic you have selected in the browser.

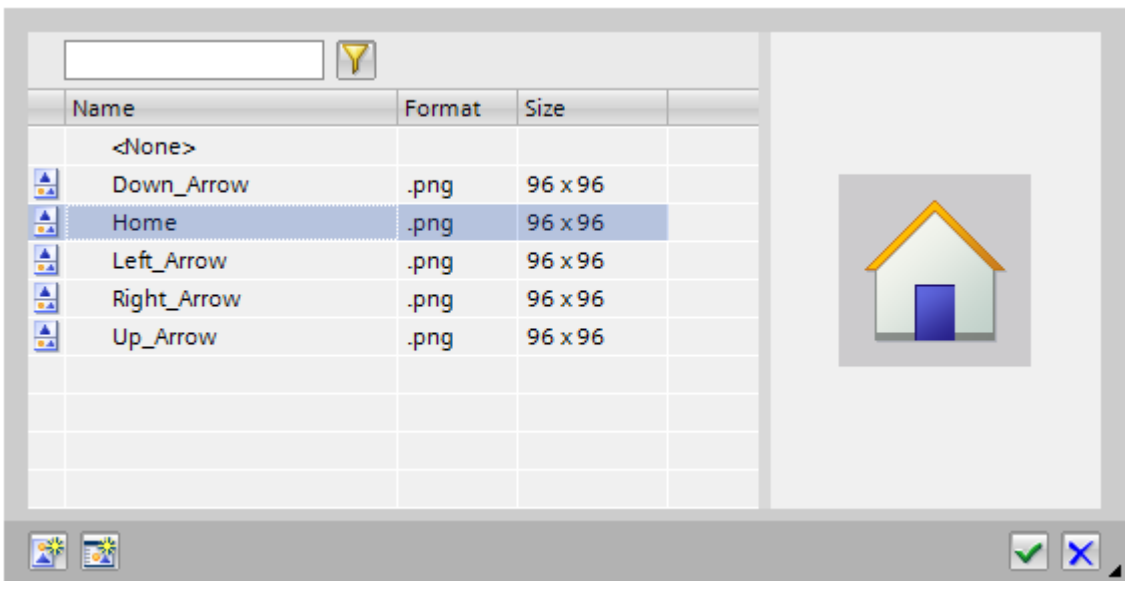

Using the and icons, you can display the collection either in form of thumbnails or as a list.
In order to open and edit OLE objects with the associated graphics program, double-click on the object.

4. In the browser click the desired graphic or store the relevant graphic in the graphic browser. The graphic preview is shown in the right pane.
5. Click "Select" to add the graphic to the screen.
   Click "Clear" to remove the graphic from the screen.

Result

The graphic is displayed next to the function key.

See also

Working with function keys (Page 241)

1.5.7 Configuring LED tags (RT Advanced, Basic Panels, Panels, Comfort Panels)

Requirements

- An HMI device with key operation has been created.
- You have created an LED tag.
- The global screen is open.

Procedure

1. Click an F-key of the HMI device.
2. In the Inspector window, click "Properties > Properties > General".

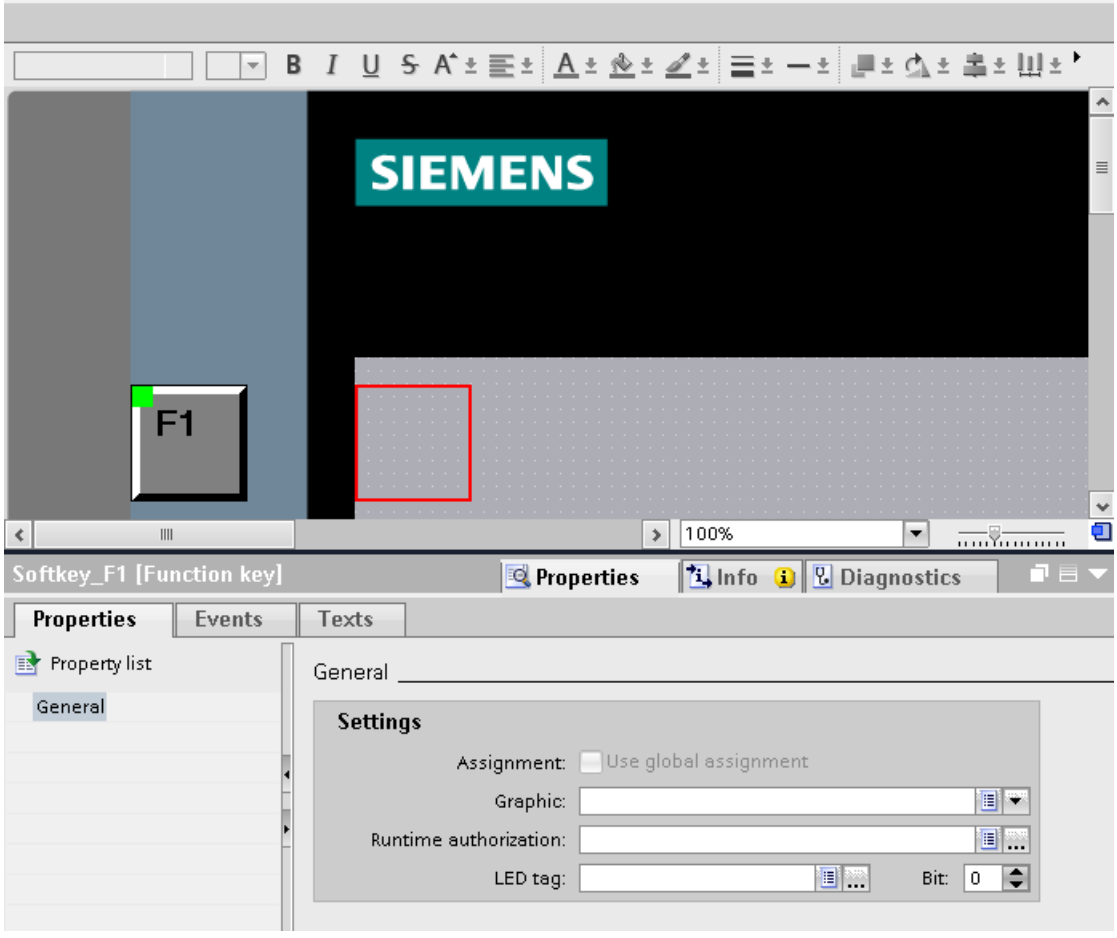

3. Select a tag under "LED tag" in the "General > Settings" area.
4. Under "Bit" enter the correct bit number.
   The correct bit number depends on the HMI device and the input and output assignments on the HMI device.

1.5.8 Example: Using function keys for screen navigation (RT Advanced, Basic Panels, Panels, Comfort Panels)

Task

In this example you create a local function key in a screen. When the operator presses this function key, a screen change to a predefined screen is triggered, for example "Boiler 2".

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

- The screen in which you want to assign the function key is open.
- You have created the "Boiler 2" screen.
- The Inspector window is open.

Procedure

Proceed as follows to use the "ActivateScreen" function:

1. Select the desired function key.
   The properties of the function key are shown in the Inspector window.
2. Click "General."
3. To overwrite a global assignment, disable the "Use local template" option.
4. Click "Key pressed" under "Events".

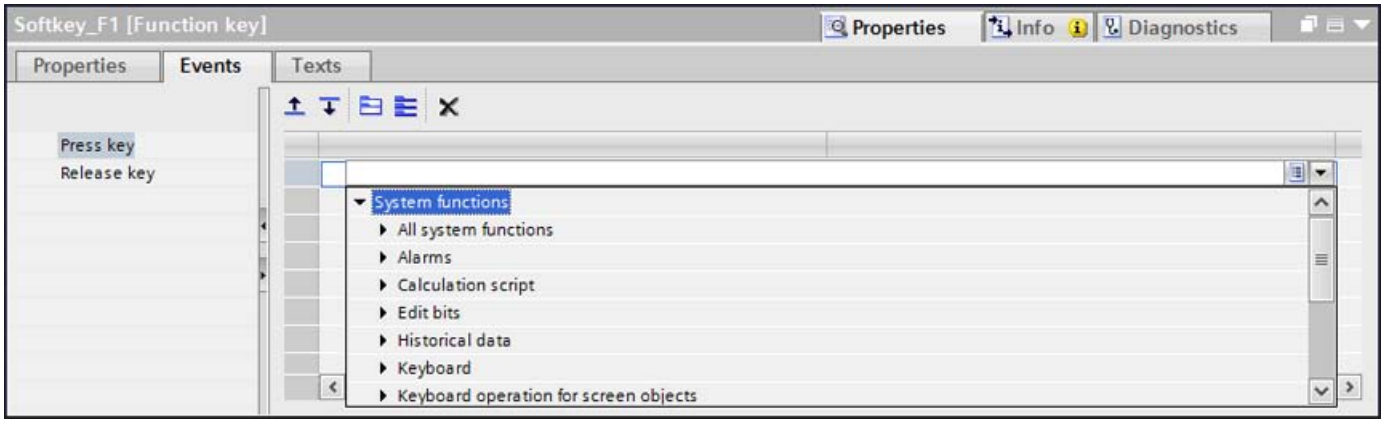

5. Select the "ActivateScreen" system function from the list.
   The "ActivateScreen" function appears in the "Function list" dialog box, including the "Screen name" and "Object number" parameters.

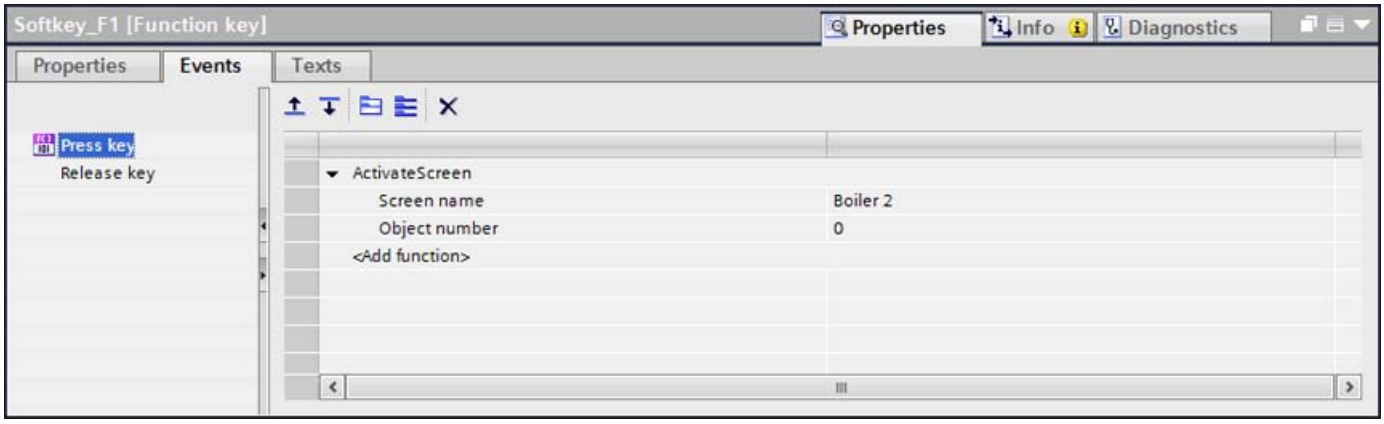

6. Select the "Boiler 2" screen from the "Screen name" list.

Result

The operator changes to the "Boiler 2" screen in runtime by pressing the selected function key.

See also

Working with function keys (Page 241)

1.6 Working with layers

1.6.1 Basics on working with layers

Layers

Use layers in order to achieve differentiated editing of the objects in a screen. A screen consists of 32 layers that you can give any names. If you assign objects to the layers, you thereby define the screen depth. Objects of layer 0 are located at the screen background, while objects of layer 31 are located in the foreground.

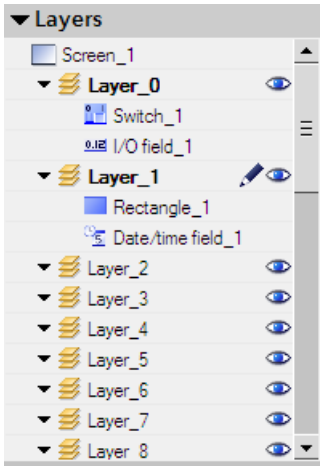

The objects of a single layer are also arranged hierarchically. When you create a screen, the object inserted first is located at the rear within the layer. Each further object is placed one position towards the front. You can shift objects forwards and backwards within a layer.

Principle of the layer technique

Always one layer of the 32 layers is active. New objects you add to the screen are always assigned to the active layer. The number of the active level is displayed in the inspector window of the screen and in the "Layout > Layers" task card.

When you open a screen, all 32 layers of the screen are displayed. You can hide all the layers except for the active layer in the inspector window of the screen and in the "Layout > Layers" task card. You then explicitly edit objects of the active layer.

In the tree view of the "Layers" palette in the "Layout" task card, you administer layers and objects with drag-and-drop and the context menu.

Application examples

Use layers, for example, in the following cases:

- To hide the labeling of objects when editing,
- To hide objects, e.g. alarm windows, while configuring other objects

Restrictions on the objects in RT Professional

Some controls in WinCC RT Professional appear in a separate window in Runtime.

The following controls are displayed in Runtime regardless of the levels:

- Table view
- F(x) trend view
- F(t) trend view
- Value table
- Recipe view
- Alarm view
- .Net control
- HTML browser
- Screen window

The assignment of these controls to certain levels has no effect in Runtime.

See also

Screen basics (Page 37)
Moving objects on layers (Page 257)
Setting the active layer (Page 258)
Showing and hiding layers (Page 259)
Showing and hiding layers (Page 261)
Hiding layers and objects in Runtime (Page 262)
Renaming layers (Page 263)

1.6.2 Moving objects on layers

Introduction

By default, a new object is inserted on the active layer. You can, however, assign an object to another layer at a later time.

Requirement

- A screen with an object is open.
- The Inspector window is open.

Procedure

1. Select the object in the screen.
   The object properties are displayed in the Inspector window.
2. Enter the layer to which you want to move the object in "Properties > Properties > Miscellaneous > Layer" in the Inspector window.

Alternatively, select the object from the "Layout" task card and drag it to the required layer.

Changing the order of objects

1. Select the object in the screen.
   The object properties are displayed in the Inspector window.
2. To move the object to the front or back, select the "Order" > "Move backward" or "Move forward" command from the shortcut menu.
   Alternatively, use the or button in the toolbar.

Result

The object is assigned to the selected layer, and positioned at the top of the layer.

See also

Basics on working with layers (Page 256)

1.6.3 Setting the active layer

Introduction

The screen objects are always assigned to one of the 32 layers. There is always an active layer in the screen. New objects you add to the screen are always assigned to the active layer.

The number of the active layer is indicated in the "Layer" toolbar. The active layer is indicated by the icon in the "Layout > Layers" task card.

Layer 0 is the active layer when you start programming. You can activate a different layer during configuration, if necessary.

Requirement

- You have opened a screen which contains at least one object.
- The Inspector window of the active screen is open.

Procedure

1. Click "Properties > Properties > Layers" in the Inspector window of the current screen.
2. Enter the layer number in "Settings > Active layer".

Alternative procedure

1. Select "Layout > Layers" in the "Layout" task card.
2. Select the "Set to active" command from the shortcut menu of a layer.

Result

The layer with the specified number is now active.

See also

Basics on working with layers (Page 256)

1.6.4 Showing and hiding layers (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You can show or hide the layers of a screen as required. You specify the layers that are shown in the Engineering System. When you open a screen, all the layers are always shown.

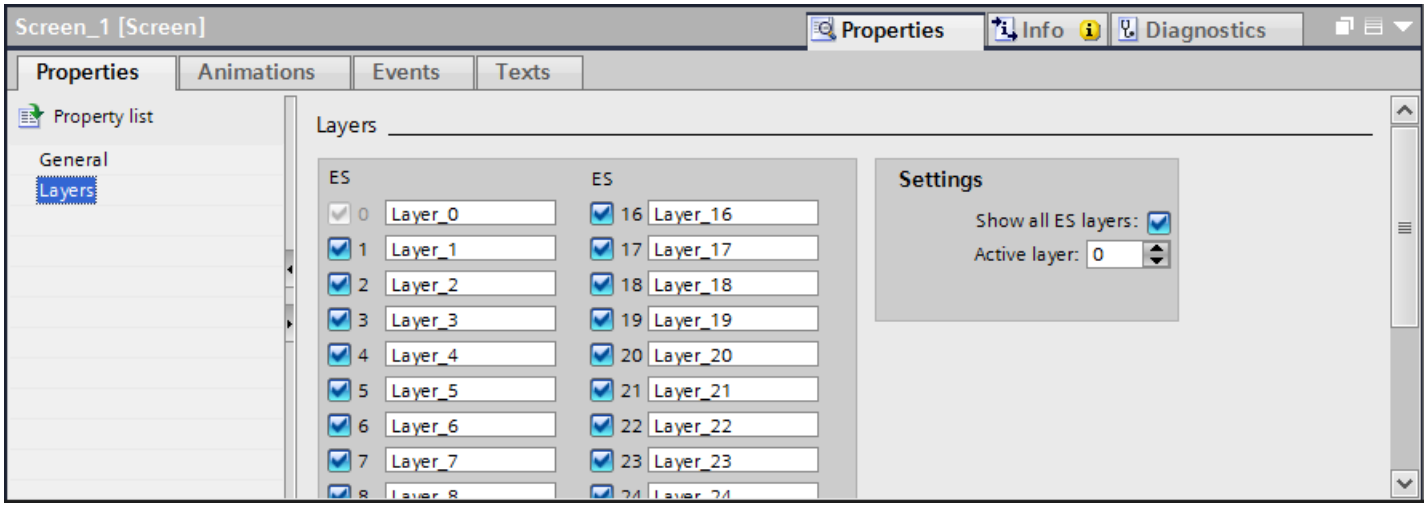

Requirement

- The screen is opened.
- The "Layout" task card is open.

Procedure

1. Select the layer that you want to hide or show in the "Layout > Layers" task card.
2. Click one of the icons next to the corresponding layer:
   - A shown layer is hidden
   - A hidden layer is shown

   Note

   The active layer cannot be hidden.

Alternative procedure

1. Click in an area of the screen that does not contain an object.
   The screen properties are shown in the Inspector window.
2. In the Inspector window, select "Properties > Properties > Layers":

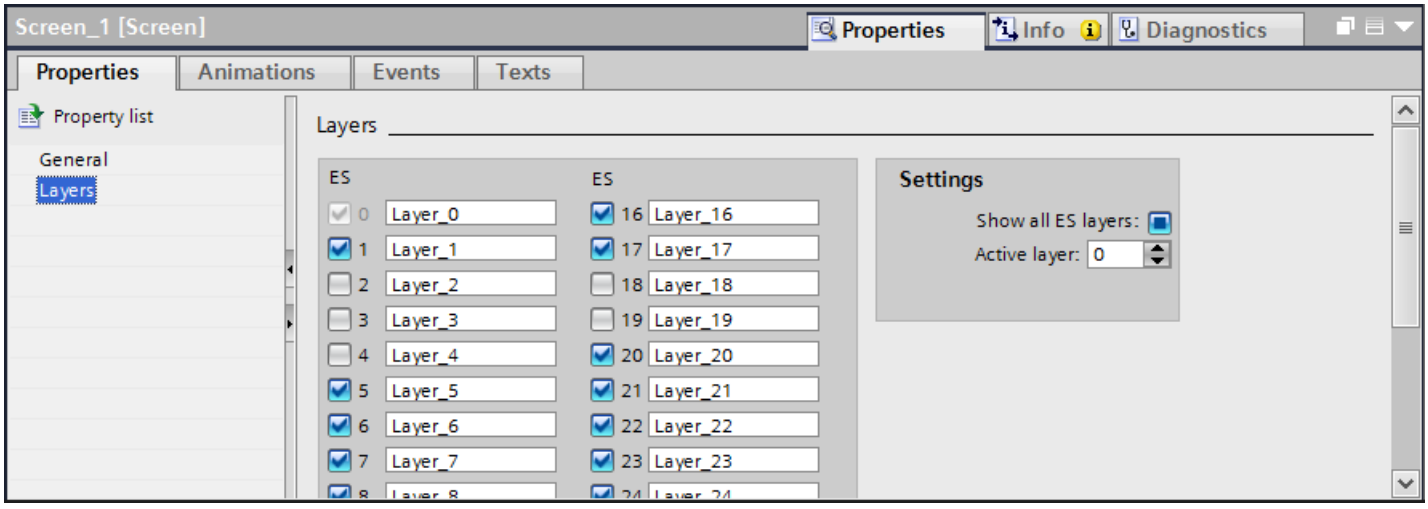

3. In the list, disable the levels you wish to hide.
   If you activate "All ES layers" for a layer, the objects in this layer will be shown in the Engineering System.

Result

The layers are shown according to your settings.

See also

Basics on working with layers (Page 256)

1.6.5 Showing and hiding layers (RT Professional)

Introduction

You can show or hide the layers of a screen as required. You define whether the layers are displayed in the engineering system, and/or in Runtime. When you open a screen, all the layers are always shown.

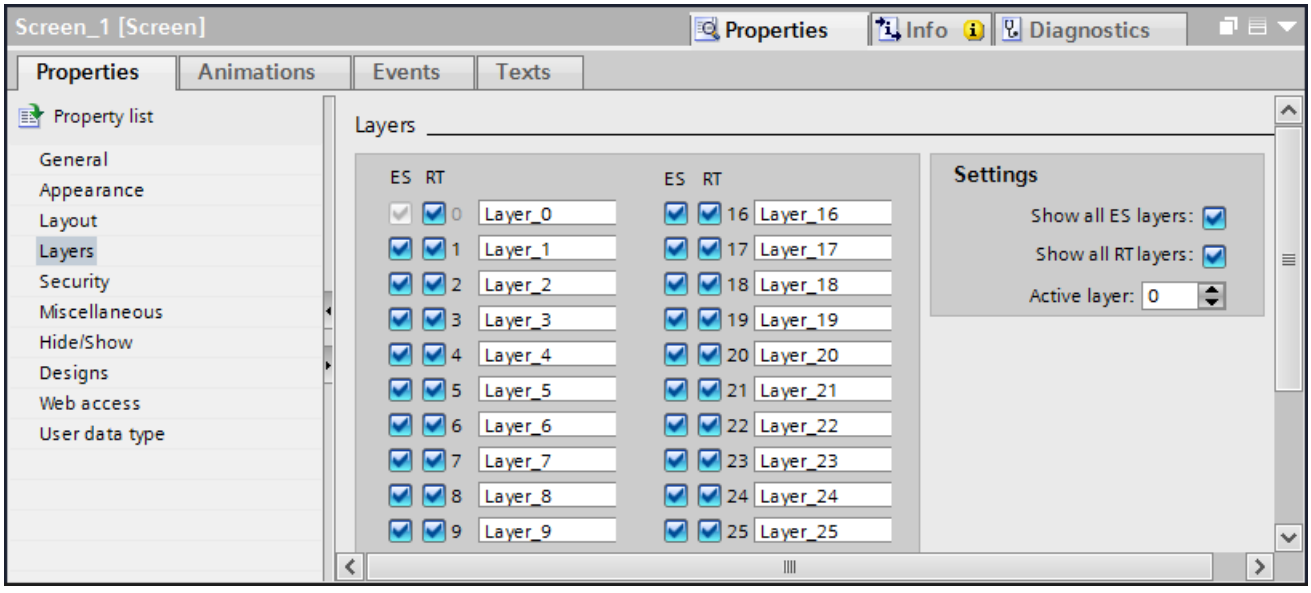

Requirement

- The screen is opened.

Procedure

1. In the Inspector window, select "Properties > Properties > Layers":
2. Deactivate the layers that are not to be displayed in the Engineering System.
3. Deactivate the layers that are not to be displayed in runtime.

Alternatively you can also activate and deactivate the layers in the "Layer" task card.

Result

The activated layers are displayed in ES and in Runtime.

See also

Basics on working with layers (Page 256)

1.6.6 Hiding layers and objects in Runtime (RT Professional)

Introduction

Layers and the objects stored in them can be shown and hidden in Runtime system-dependently depending on the zoom factor. This means, for example, that a layer is only shown if a sufficient zoom factor allows the required detail to also be displayed.

You can hide or show objects depending on the defined pixel size.

Requirement

A screen is open.

Hiding layers

1. Click in an area of the screen that does not contain an object.
2. Activate "Properties > Properties > Hide/Show > Activate for layers" in the Inspector window.

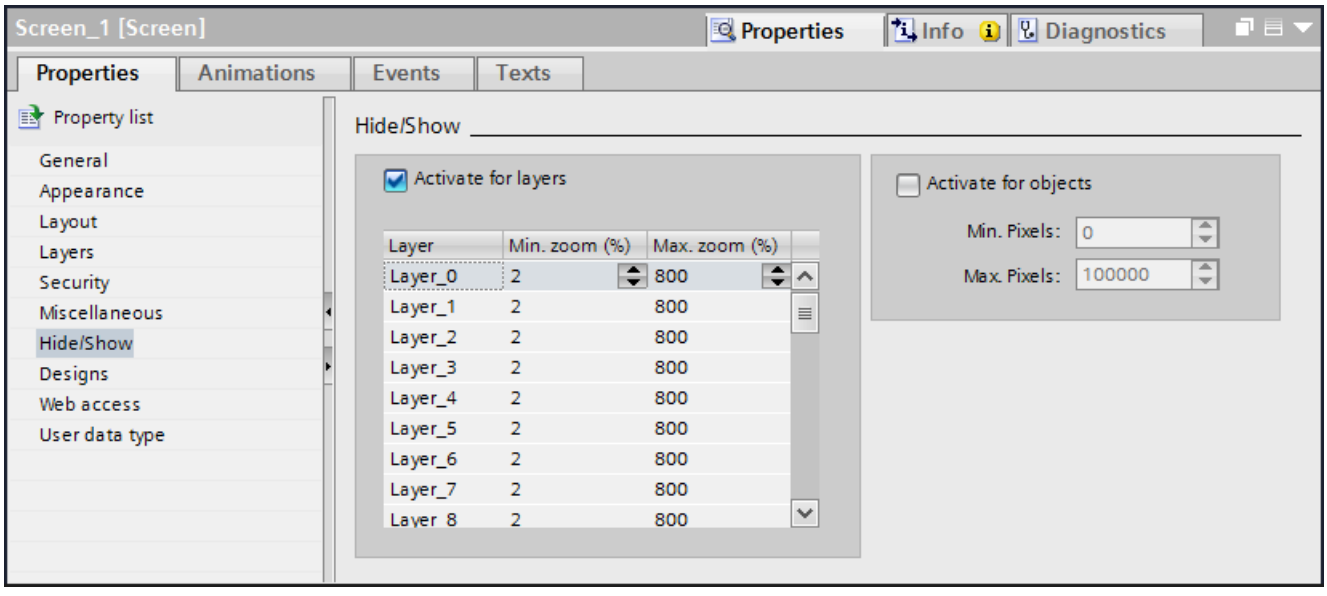

3. In the table enter the minimum and the maximum value for the zoom factor within which the respective layer is displayed.

Hiding objects

1. Click in an area of the screen that does not contain an object.
2. Activate "Properties > Properties > Hide/Show > Activate for layers" in the Inspector window.

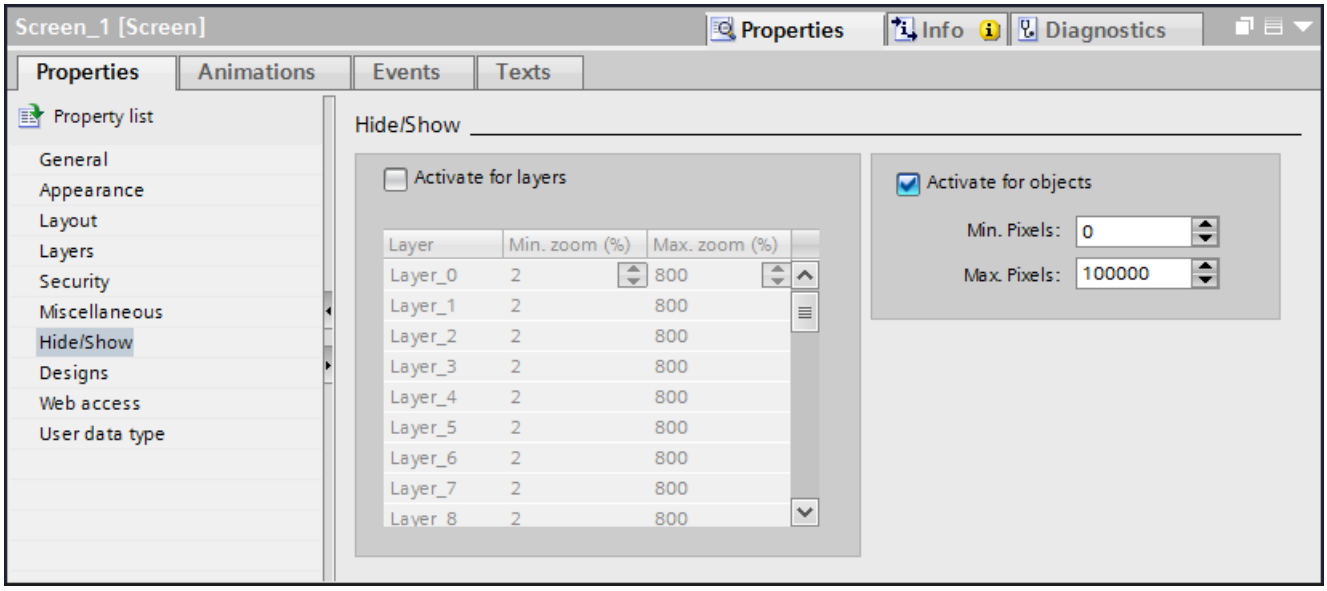

3. Enter the minimum size in which the objects are displayed under "Min. pixels."
4. Enter the maximum size in which the objects are displayed under "Max. pixels."

Result

Levels which are outside the zoom factor are not displayed in Runtime. Objects which are outside the specified size are also not displayed.

See also

Basics on working with layers (Page 256)

Supported gestures (Page 2311)

1.6.7 Renaming layers

Introduction

When you create a screen, the 32 layers are numbered consecutively by default. To improve clarity, you can rename the layers to suit your requirements.

Requirement

- The screen is opened.

Procedure

1. Click in an area of the screen that does not contain an object.
   The screen properties are shown in the Inspector window.
2. In the Inspector window, select "Properties > Properties > Layers".

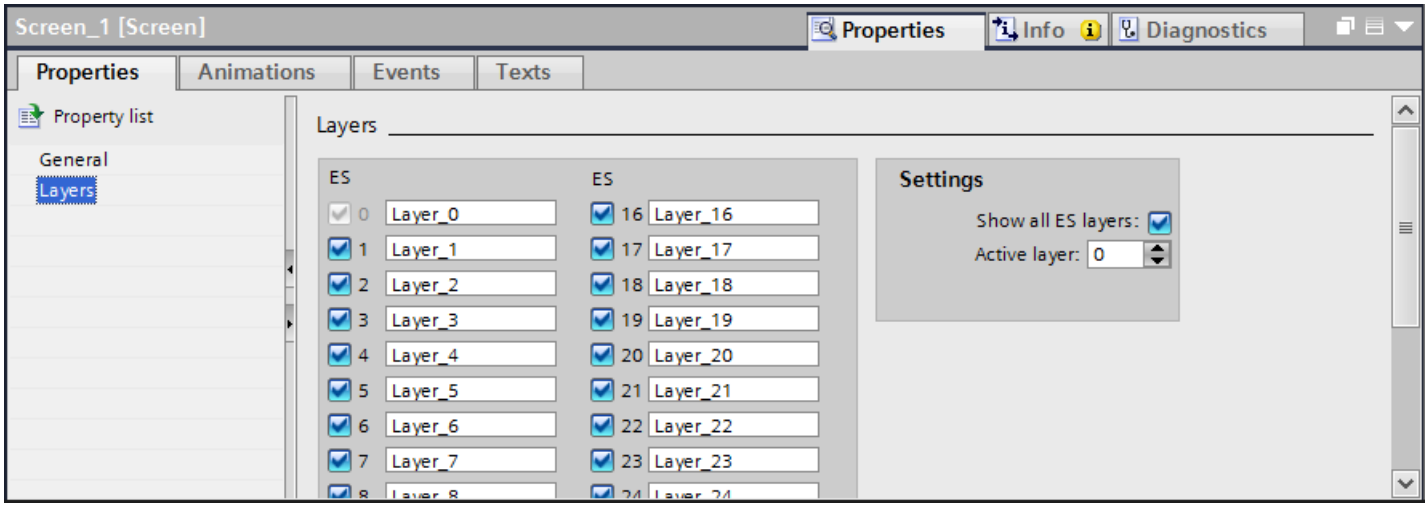

3. Enter the new layer name.

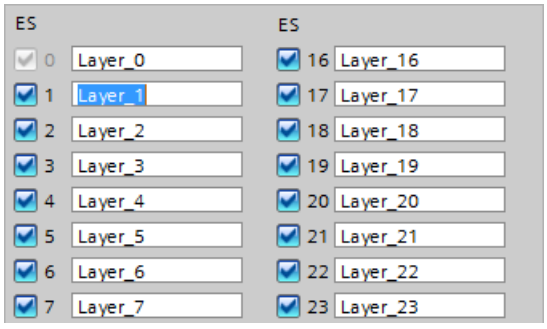

Result

The layer is displayed with the new name.

See also

Basics on working with layers (Page 256)

1.7 Working with faceplates

1.7.1 Basics on faceplates

1.7.1.1 Basics on faceplates (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Faceplates are a configured group of display and operating objects that you manage and change centrally in a library. You can use a faceplate in several projects as required. The faceplates are stored in the project library.

Use

You use faceplates to create individually configured display and operating objects. You can use faceplates several times in the project or in different projects. All instances of a faceplate in the project are changed centrally. This reduces the configuration effort.

Types and instances

Faceplates are based on a type-instance model to support the central changeability. You create central properties for an object in types. The instances represent local points of use of the types.

- Faceplate type
  You create a display and operating object according to your requirements and store it in the project library. The instances are bound to the respective faceplate type. If you change a property of a faceplate type, the property is saved centrally and also changed in all instances. In the faceplate type, you define the properties that can be changed on the faceplate. You edit a faceplate type in the "Faceplates" editor.
- Faceplate
  The faceplate is an instance of the faceplate type. You use a faceplate in screens as a display and operating object. You configure the variable properties of the faceplate type on the instance. You assign the tags of your project to the faceplate, for example. If you configure properties on the faceplate, you overwrite the properties of the faceplate type. The changes on the faceplate are saved at the point of use and have no effect on the faceplate type. To reset the variable properties at the faceplate to the properties of the faceplate type, select "Reset instance" in the shortcut menu.

The figure below shows the relationships between the faceplate type and the faceplate. The background color of the I/O field is configured in the faceplate type in such a way that every instance can change the property. Blue is used as the background color in the faceplate type. You use yellow as the background color in the faceplate.

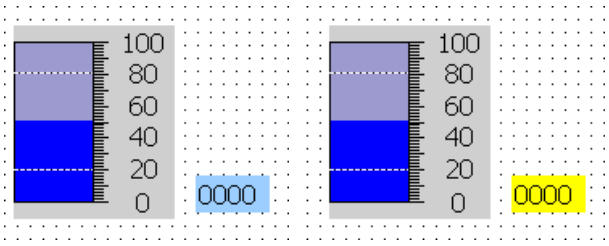

When you change the background color in the faceplate type, this has no effect on the faceplate because the "Background color" property is assigned on an instance-specific basis.

See also

Creating a faceplate type (Page 270)

Faceplate editor (Page 267)

Basics of dynamizing faceplates (Page 292)

Example: Configuring a faceplate (Page 294)

Screen basics (Page 37)

1.7.1.2 Device dependency of faceplates (RT Professional, RT Advanced, Panels, Comfort Panels)

Device dependency of faceplates

Not all HMI devices support every display and operating object. The screen objects that are not available in the respective HMI device are not displayed when using the faceplate.

Independent of the device, all properties are offered for the configuration during the generation of faceplates. When using a faceplate in a screen, only the properties supported by the configured device are available.

The following devices support faceplates:

Runtimes

- WinCC Runtime Advanced
- WinCC Runtime Professional

Comfort Panels

- KTP 400
- KP 400
- TP 700
- KP 700
- TP 900

- KP 900
- TP 1200
- KP 1200
- TP 1500
- KP 1500
- TP 1900
- TP 2200

Mobile Panels

- Mobile Panel 277
- Mobile Panel 277 IWLAN V2
- Mobile Panel 277F IWLAN V2
- Mobile Panel 277F IWLAN (RFID Tag)
- Mobile Panel KTP 400F
- Mobile Panel KTP 700
- Mobile Panels KTP 700F
- Mobile Panel KTP 900
- Mobile Panel KTP 900F

1.7.1.3 Faceplate editor (RT Professional, RT Advanced, Panels, Comfort Panels)

Open

Create faceplate types and edit them in the library view.

Layout

The library view for faceplate types consists of several editors, e.g. "Screens" and "Tags" which are arranged in the work area and the configuration area.

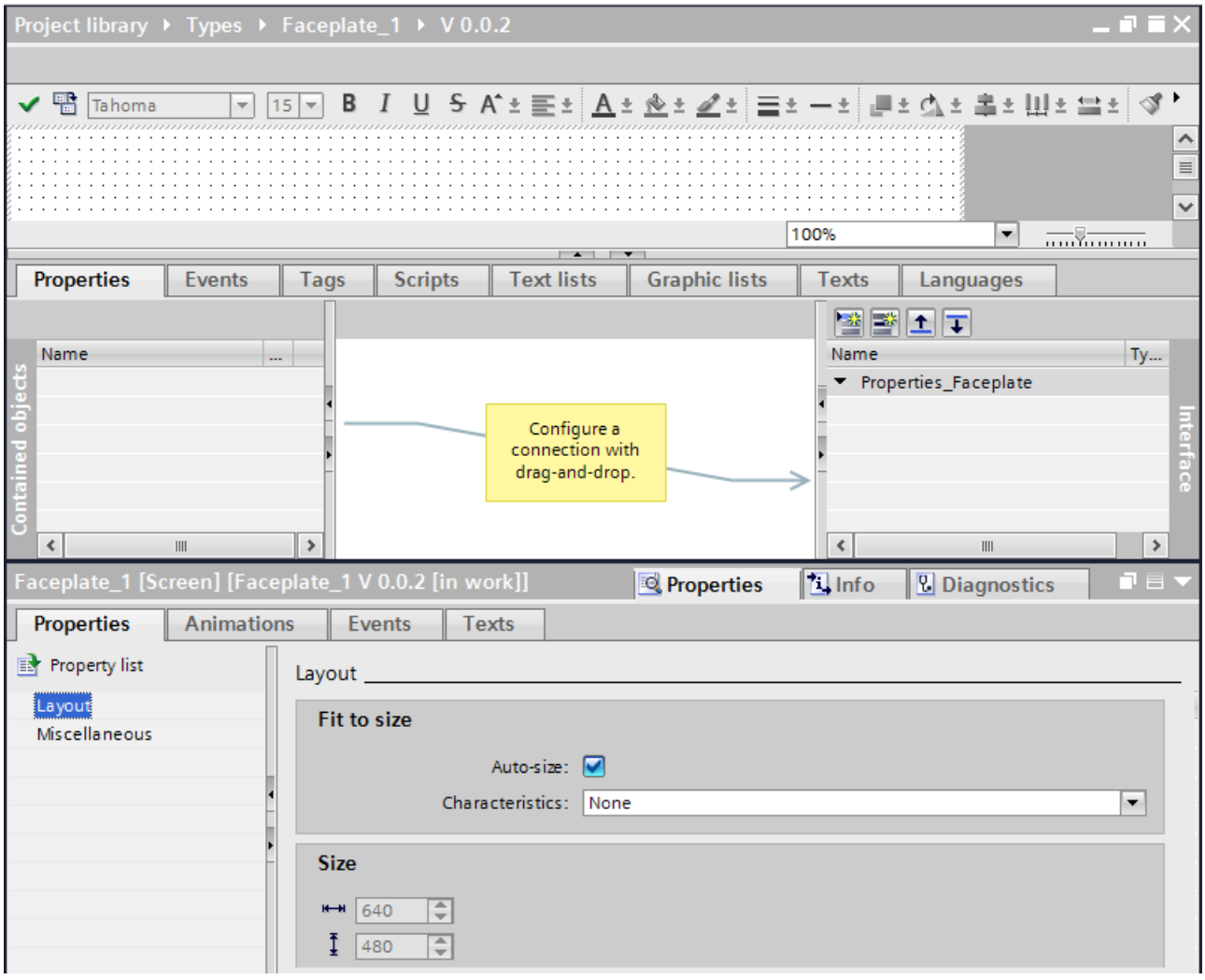

Work area

In the work area, place the objects contained in the faceplate type as you usually do in the "Screens" editor. You can remove objects, or use the "Toolbox" task card to add new objects.

Note

Faceplates cannot be rotated or mirrored.

Configuration section

- Properties
  The static and dynamic properties are identified as follows:
  - The symbol identifies a dynamic property. Dynamic implementation of this property, using tags or text and graphic lists.
  - The symbol identifies a static property. You change only the values of a static property.

  Note

  Device dependency of static properties

  Static properties are only available for panels and RT Advanced.

  You define the faceplate type properties in the "Properties" tab.
  You use the following two lists for this:
  - The "Contained objects" list
    This list includes the properties of the objects contained in the faceplate type. These properties can only be configured in the faceplate type in the "Faceplate" editor.
  - The "Interface" list contains the properties and tags of the faceplate type. The "Interface" list consists of the pre-defined category "Dynamic properties" and user-defined categories.
- Events
  You define the faceplate type events in the "Events" tab. You use the following two lists for this:
  - The "Contained objects" list contains the events of the objects embedded in the faceplate type.
  - The "Interface" list contains the events of the faceplate.
    You create links between the two lists using a drag and drop operation.
- Tags
  You can also create faceplate tags on the "Tags" tab. The tags are only available within the faceplate type. You can interconnect the faceplate type tags directly with an object contained in the faceplate type.
- Scripts
  You configure scripts for the faceplate type in the "Scripts" tab. In the script, you can call system functions or program new functions to convert values, for example. The scripts are only available within the faceplate type. They work with VB Script code.

  Note

  The "Scripts" tab is not available for Runtime Professional.

- Text lists
  You can also create and edit texts lists for the faceplate type in the "Text lists" tab. These text lists are only available within the faceplate type. You can interconnect the text lists of the faceplate type directly with an object contained in the faceplate type, such as a symbolic I/O field.

- Graphic lists
  If necessary, you can also create graphics lists for the faceplate type in the "Graphics lists" tab. These graphic lists are only available within the faceplate type. You can interconnect the graphic lists of the faceplate type directly with an object contained in the faceplate type, e.g. a graphic I/O field.
- User texts
  The "User text" tab shows the contained user text of the opened faceplate type, for example entries from text lists. You can enter the compilations for the individual project languages directly in the "User text" tab.

See also

Basics on faceplates (Page 265)

Example: Configuring a faceplate (Page 294)

1.7.2 Creating and managing faceplate types

1.7.2.1 Creating a faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Faceplate types are display and operating objects which are made up of several objects, e.g. controller modules.

Requirements

You have opened a screen which contains multiple objects.

Procedure in the project view

1. Select all the objects that you need for the faceplate type.
2. Select the "Create faceplate type" command in the shortcut menu of the multiple selection. A dialog opens.
3. Enter a name for the faceplate.
4. If necessary, add a comment or change the version number.

Result

The library view opens. Two versions of the type are displayed in the "types" folder. Version 0.0.1 has been released and contains the currently selected objects.

The 0.0.2 version has the status "in progress". Edit the faceplate type in version 0.0.2 as required.

Procedure in the library view

1. The library view is open.
2. Select the "Add new type" command in the shortcut menu of the project library. A dialog opens.
3. Select the runtime for which the faceplate type is to be available. An empty faceplate type is created.
4. Drag-and-drop the objects from the "Toolbox" task card to the work area of the faceplate type.

Result

The new faceplate type is created and displayed under the selected name in the project library.

The faceplate type is assigned the status "in progress" and the version 0.0.1.

See also

Basics on faceplates (Page 265)

Configuring a faceplate type (Page 272)

Editing the category and property of a faceplate type (Page 276)

Configuring a tag in the faceplate type (Page 277)

Configuring scripts in the faceplate type (Page 278)

Configuring an event in the faceplate type (Page 281)

Configuring a text list in the faceplate type (Page 282)

Configuring a graphics list in the faceplate type (Page 283)

Translating texts directly in the faceplate type (Page 284)

Editing a faceplate type (Page 287)

Resizing a faceplate (Page 288)

Creating an instance of the faceplate type (Page 290)

Removing faceplate from faceplate type (Page 291)

Master copies and types (Page 1938)

Example: Configuring a faceplate (Page 294)

1.7.2.2 Configuring a faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In the configuration area of the "Faceplates" editor you define which properties and process values of the objects contained in the faceplate can be configured in the "Properties" tab.

Note

For tooltip texts configured for objects in the faceplate to be displayed in Runtime, insert the property "Tooltip text" as faceplate property.

Note

Independent of the device, all properties are offered for the configuration during the generation of faceplates. When using a faceplate in a screen, only the properties supported by the configured device are available.

Note

The use of user-defined PLC data types (UDTs) of PLCs S7-300 and S7-400 is not supported. These PLC data types cannot be selected as a faceplate property.

Requirements

- The faceplate is generated and is displayed in the "Faceplate" editor.
- The "Properties" tab is open in the configuration area.

Procedure

1. To create a new property, click the "Add property" button in the interface list.
   A new property is displayed in the "Interface" list.
2. To create a new category, click the "Add category" button in the interface list.
   A new category including a new property is displayed in the "Interface" list.

3. Click the name of the property and assign a name, for example, "Color".
4. Select the data type.

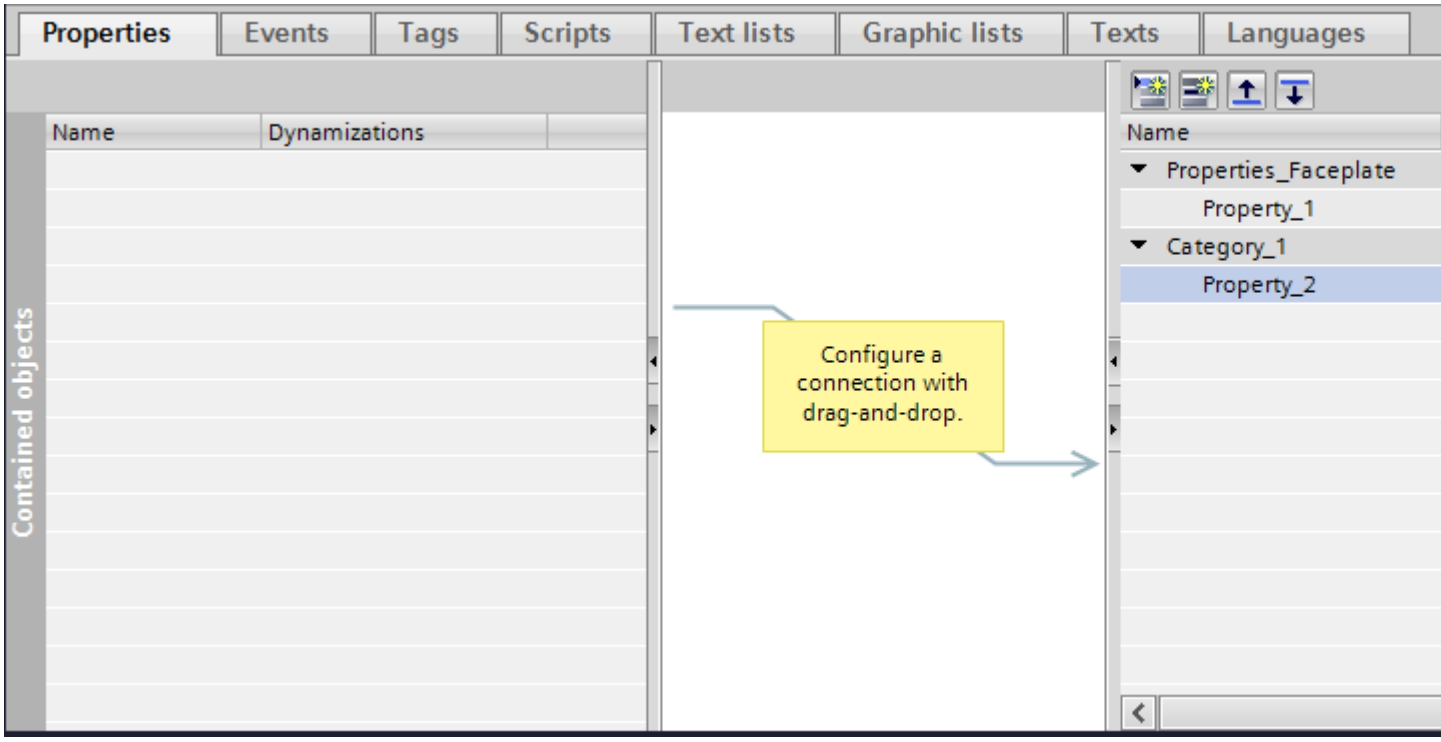

Note

Property names of the faceplates in RT Professional

Note the following rules when entering property names of the faceplates:

- The name must start with a letter.
- The name can contain alphanumeric characters and the underscore.
- The name may not contain more than 255 characters.
- Do not use any UNICODE characters (for example, Chinese characters).

Note

Using user data types (UDTs) in faceplates in RT Professional

If an element of a faceplate is assigned as data type of the user data type (UDT), the following rule applies:

The length of the name of a property of the faceplate together with the length of the name of the lower-level element of the "User Data Type" (UDT) data type must not exceed 126 characters.

Interim result

You have created a new property.

1. Click an object in the "Contained objects" list.
2. Select a property, e.g. "Appearance > Background color".
3. Move the property from the "Contained objects" list to the desired property in the "Interface" list by drag&drop.
   The connection is displayed as a colored line.
   The system assigns the same data type to the property in the "Interface" list as that of the property in the "Contained objects" list.
   To change the data type, select the data type from the data types of the connected property.
4. Repeat step 3 to connect other properties of the contained properties with the interface.

Note

The property from the "Contained objects" list must be in concordance with the data type of the already connected property of the "Interface" list.

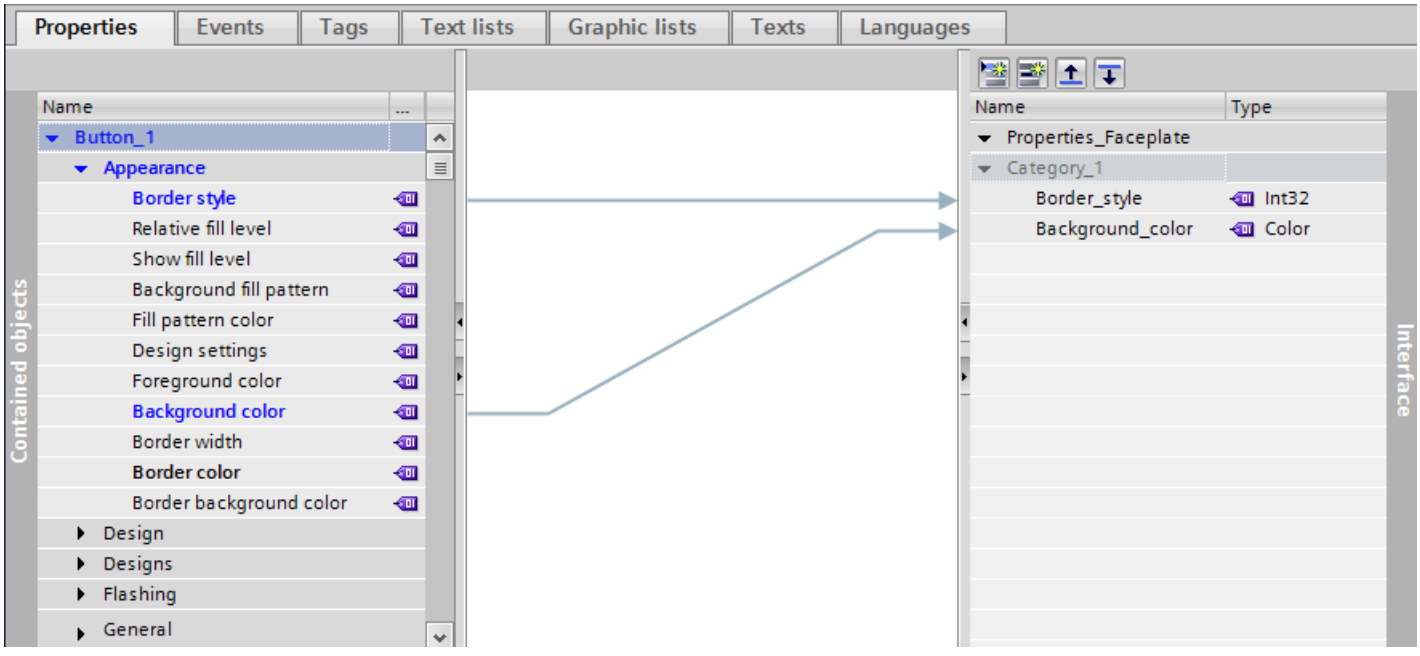

Deleting a connection

1. Click the connection that you wish to delete.
2. In the shortcut menu select the "Delete" command or use the <DEL> key.
   The connection is deleted.

Interconnecting user data types with faceplates

As of V13 SP1, you can use the HMI user data types and the PLC user data types as properties of the faceplates.

You interconnect the HMI user data types and the PLC user data types with the faceplates using interface properties by connecting the faceplates with the structural elements of the interface properties.

To set the HMI user data types and the PLC user data types as properties of the faceplates, follow these steps:

1. Select the user data type in the selection list under "Type".

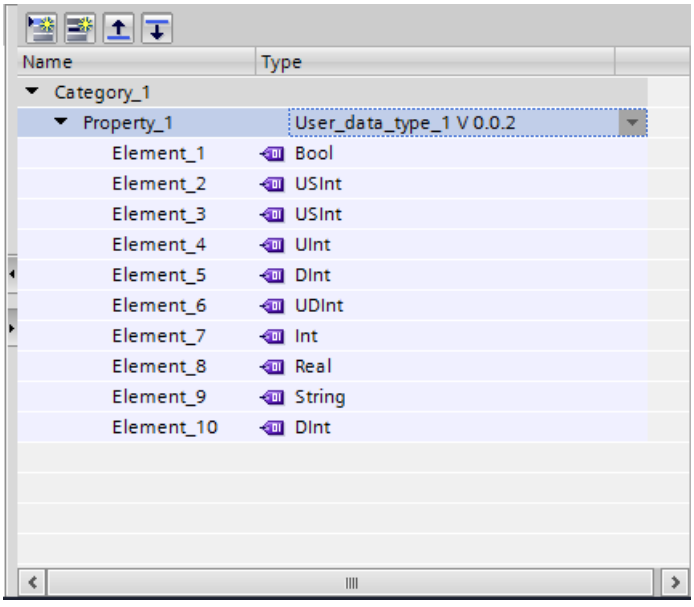

2. In the Inspector window under "Properties > Properties > General", select the interface property of the faceplate in the "Tag" selection list.

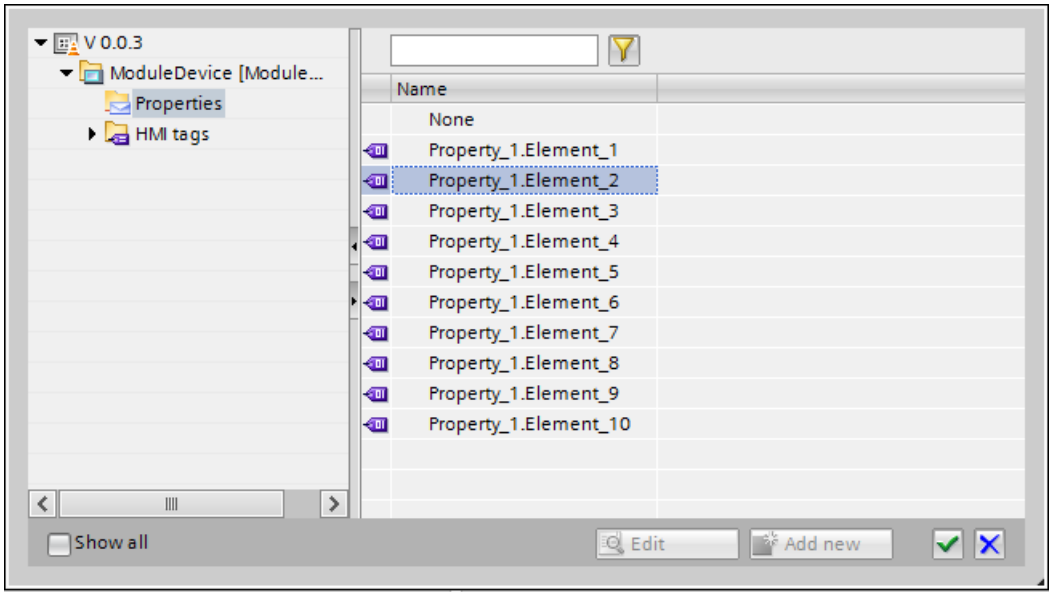

The faceplate is interconnected with the user data type.

Result

You have created a new property or a new category with a property. You have connected the property of the interface with a property of the contained objects. If you use an instance of the faceplate type in a screen, you configure the properties of the faceplate which you have created in the "Interface" list.

Note

Device dependency of static properties

Static properties are only available for panels and RT Advanced. You work with dynamic properties in RT Professional.

See also

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

Linking user data types with objects (Page 141)

1.7.2.3 Editing the category and property of a faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Deleting a property in the "Interface" list

1. Select "Edit faceplate" in the shortcut menu. The "Faceplates" editor is open.
2. Select the property which you want to delete in the "Interface" list.
3. Select "Delete" in the shortcut menu.

The property is deleted from the "Interface" list. Existing connections to the contained objects are deleted. All linked faceplates lose their instance-specific property.

Deleting a category in the "Interface" list

1. Select "Edit faceplate" in the shortcut menu. The "Faceplates" editor is open.
2. Select the category which you want to delete in the "Interface" list.
3. Select "Delete" in the shortcut menu.

The category including the properties is deleted from the "Interface" list. Existing connections to the contained objects are deleted. All linked faceplates lose their instance-specific properties.

Moving a property to another category

1. Select a property in the "Interface" list.
2. Move the property to a new category by "drag&drop".

You have moved the property to a new category.

Changing the data type of a property

1. Select a property in the "Interface" list.
2. Click the second column of the drop-down list.
3. Select a data type.

You have changed the data type of the property in the "Interface" list.

See also

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

1.7.2.4 Configuring a tag in the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You connect a tag in the faceplate type directly with the properties of the objects contained in the faceplate type. The tags in the faceplate type are a central means of defining dynamic properties in the faceplate type.

The tags of a faceplate type have limited functionality. The "Tags" tab of the configuration area has the same structure as the "Tags" editor.

Note

Use of the "." or "@" character in names of tags in faceplate types is not permitted. Do not use the special characters in the tag names in faceplates.

Array elements for faceplates

In the following situations, you may not assign an array element or a multiplex tag that is interconnected with a faceplate to a property of a screen object:

- The property is configured in a faceplate as parameter of a system function.
- System functions or scripts are assigned in the faceplate to events of the property.

Requirements

- The library view is open.
- The faceplate type is created and has the "in progress" status.
- The "Tags" tab is open in the configuration area.

Procedure

1. Click "Add" in the "HMI tags" table. A new tag is created in the faceplate type.
2. Open "Properties > Events" in the inspector window if necessary. You configure the "Value change" event on the tag for example.
3. In the work area, select the object to which you want to assign this tag, e.g. an I/O field.
4. Select the tag in the inspector window of the I/O field "Properties > Properties > General > Process > Tag".

Note

Only tags of the faceplate type are displayed in the object list in the "Faceplates" editor.

Result

You have created a tag in the faceplate type. You are using the tag in the faceplate for scripting, for example.

See also

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

1.7.2.5 Configuring scripts in the faceplate type (RT Advanced, Panels, Comfort Panels)

Introduction

In the configuration area of the "Faceplates" editor you create scripts which you only use within a faceplate type. You can only refer to tags of the faceplate type or properties of the contained objects within the script. The script is used as a copy in the instance of the faceplate type.

Requirement

- The library view is open.
- The faceplate type is created and has the "in progress" status.
- Faceplate tags or dynamic properties have been created.

Procedure

1. Click the "Scripts" tab in the configuration area of the faceplate.
2. Double click "Faceplate scripts > Add VB script".
3. Write the program code.
4. Press the shortcut keys <CTRL+J> to Include tags of the faceplate type in the script. The object list opens.

5. Click "HMI tags".
   All tags of the faceplate type are displayed in the object list.
6. Select a tag and confirm the selection.
7. Press the shortcut keys <CTRL+J> to use properties of the faceplate type in the script. The object list opens.
8. Click "Properties".
   All dynamic properties of the "Interface" list for the faceplate type are displayed in the object list.

Result

You have created a script in the faceplate type. The instance of the faceplate type uses a copy of the script. The script is executed in runtime depending on the configuration.

See also

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

1.7.2.6 Configuring an event in the faceplate type (RT Advanced, Panels, Comfort Panels)

Introduction

In the configuration area of the "Faceplates" editor you define which events of the objects contained in the faceplate can be configured in the "Events" tab. You configure a function list on the events of the faceplate in the "Screens" editor.

Requirement

- The library view is open.
- The faceplate type is created and has the "in progress" status.
- The "Events" tab of the configuration area is open.

Procedure

1. Click an object in the "Contained objects" list.
2. Select an event, e.g. "Activate".

3. Move the event from the "Contained objects" list to a new category in the "Interface" list.
   The new event is displayed in the "Interface" list.
   The connection appears as a colored line.

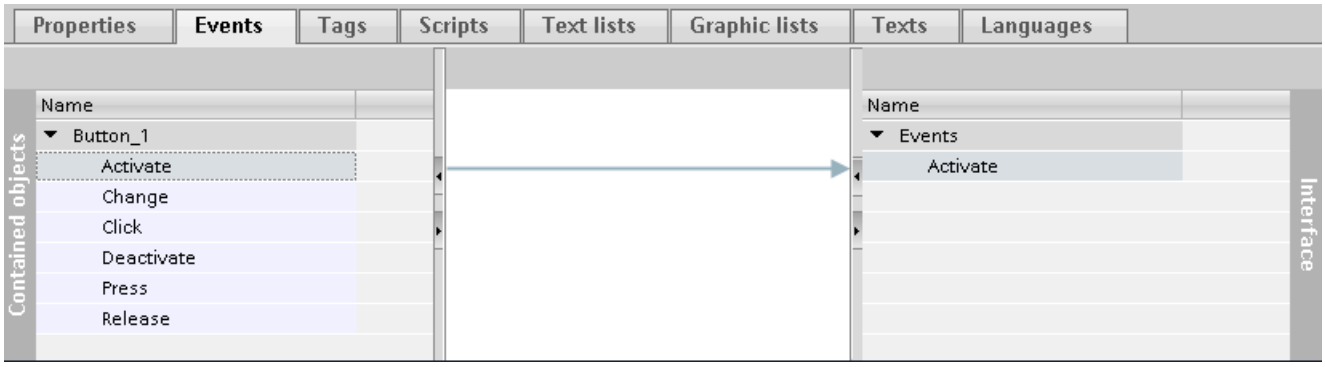

4. Double-click the event and, if required, change the name of the event.

Result

You have created an event in the faceplate type. if you use an instance of the faceplate type, you will only be offered the configured events in the inspector window. You configure a function list on the event in the faceplate.

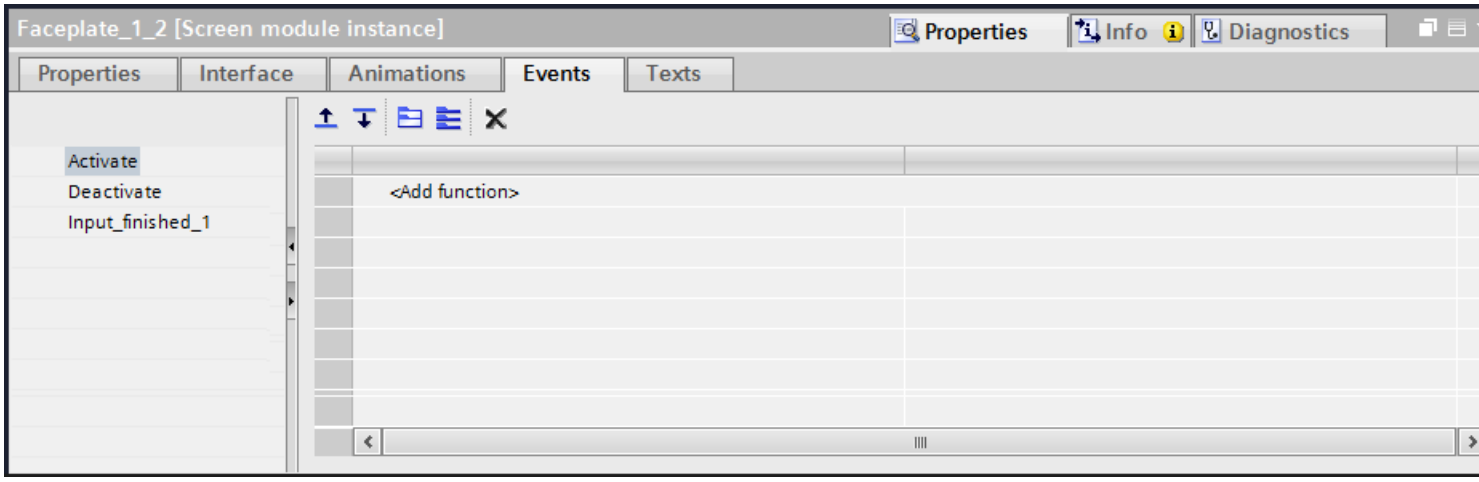

See also

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

1.7.2.7 Configuring an event in the faceplate type (RT Professional)

Introduction

In the configuration area of the "Faceplates" editor you define which events of the objects contained in the faceplate can be configured in the "Events" tab. You configure a function list on the events of the faceplate in the "Screens" editor.

Requirement

- The faceplate type is created and is displayed in the "Faceplates" editor.
- The "Events" tab of the configuration area is open.

Procedure

1. Click an object in the "Contained objects" list.
2. Select an event, e.g. "Activate".
3. Move the event from the "Contained objects" list to a new category in the "Interface" list.
   The new event is displayed in the "Interface" list.
   The connection appears as a colored line.

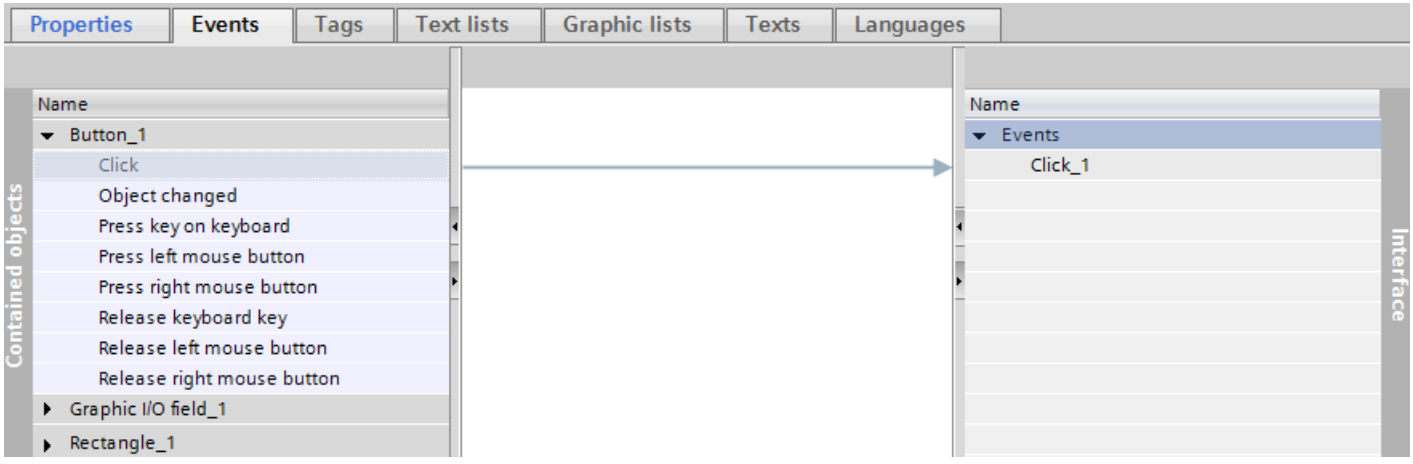

4. Double-click the event and, if required, change the name of the event.

Result

You have created an event in the faceplate type. If you use an instance of the faceplate type, you will only be offered all default settings for events in the inspector window. In the faceplate you configure a function list or a VB-script or C-script on the event.

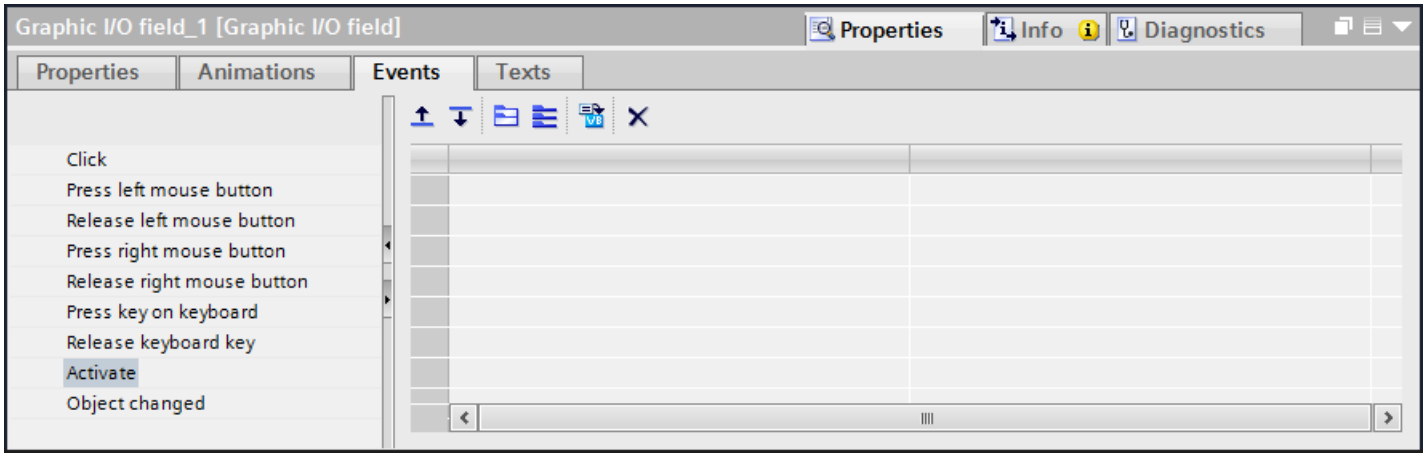

See also

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

1.7.2.8 Configuring a text list in the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In a faceplate type, you connect a text list with an object included in the faceplate type. Text is assigned to the values of a tag in a text list. The text list is used, for example, to display a selection list in a symbolic I/O field contained in the faceplate type. The interface between the text list and a tag of the faceplate type is configured at the object that uses the text list.

Requirements

- The library view is open.
- The faceplate type is created and has the "in progress" status.
- The "Text list" tab is open in the configuration range.
- A tag has been created in the faceplate type.

Procedure

1. Click "Add" in the "Text lists" table. A new text list is created in the faceplate type.
2. Assign a name to the text list that indicates its function.

3. Select the text list type, e.g. "Value/Range" under "Select".
4. Click "Add" in the "Text list entries" table.
5. Define the values or range of values for the text list.
6. Enter a text for every value range which is displayed in Runtime when the tag is within the specified value range.
7. Select an object, e.g. a symbolic I/O field, in the work area of the faceplate type.
8. In the Inspector window of the object, enable "Properties > Properties > General > Label > Text list".
9. Select the "Text list" from the selection list.
10. Select a tag of the faceplate type under "Process > Tag".

Result

You have created a text list in the faceplate type. This text list is linked to an object contained in the faceplate type. You have configured a faceplate type tag at the object.

See also

Basics on text lists (Page 176)

Creating a text list (Page 178)

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

1.7.2.9 Configuring a graphics list in the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In a graphics list, specific graphics are assigned to possible values of a tag. In a faceplate type, you connect a graphics list with an object included in the faceplate type. Graphics are assigned to the values of a tag in a graphics list. For example, the graphics list is used to display a selection list in a graphic I/O field contained in the faceplate type. The interface between the graphics list and a tag of the faceplate type is configured at the object that uses the graphics list.

Requirements

- The library view is open.
- The faceplate type is created and has the "in progress" status.
- The "Graphics list" tab is open in the configuration range.
- A tag has been created in the faceplate type.

Procedure

1. Click "Add" in the "Graphics lists" table. A new graphics list is created in the faceplate type.
2. Assign a functional name to the graphics list.
3. Select a graphics list type, e.g. "Range (... - ...)", under "Selection".
4. Click "Add" in the "Graphics list entries" table. A new entry is created in the list.
5. Define the values or range of values for the graphics list.
6. Select a graphic for every value or value range which is displayed in runtime when the tag is within the specified value range.
7. Select an object, e.g. a graphic I/O field, in the work area of the faceplate type.
8. Activate "Properties > Properties > General > Contents > Graphics list" in the object's Inspector window.
9. Select the "Graphics list" from the selection list.
10. Select a tag of the faceplate type in "Process > Tag".

Result

You have created a graphics list in the faceplate type. This graphics list is linked to an object contained in the faceplate type. You have configured a faceplate type tag at the object.

See also

Basic principles of graphics lists (Page 189)

Creating a graphics list (Page 192)

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

1.7.2.10 Translating texts directly in the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Translating texts

If you use several languages in your project, you can translate individual texts directly. As soon as you change the language of the software user interface, the translated texts are available in the selected language.

Note

When you create a faceplate in the "Text" tab, alarm class texts and alarm texts are automatically created in the tab. These texts are required for the WinCC alarm logging.

Requirements

- A project is open.
- You have opened the "Faceplates" editor.
- You have created a faceplate type.
- The faceplate type contains at least one object with text, e.g. a text field.

Procedure

Proceed as follows to translate individual texts:

1. Open the "Tasks" task card.
2. Click the button under "Languages & Resources > Editing language". The "Project languages" editor opens.
3. Activate at least two project languages.
4. Open the "Texts" tab in the "Faceplates" editor.
   A list with the texts in the project is displayed in the work area. There is a separate column for each project language.

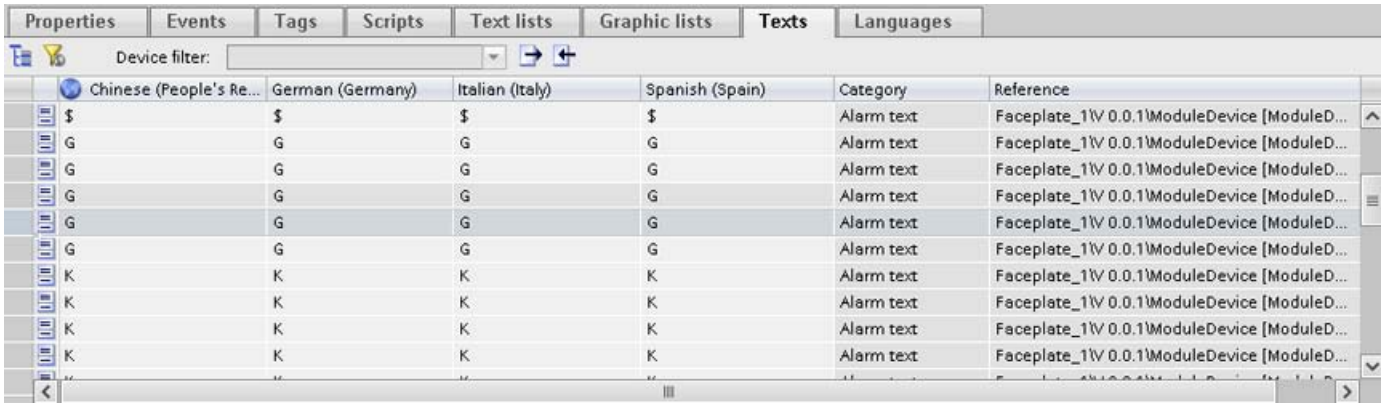

5. Click on the button in the toolbar to group identical texts together.
6. To hide texts that do not have a translation, click on the button in the toolbar.
7. Click on an empty column and enter the translation.

Result

You have translated individual texts in the "Texts" tab of the faceplate type. The texts will then be displayed in the runtime language.

See also

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

1.7.2.11 Exporting and importing texts in the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Exporting and importing texts

You have the option to export the texts of the faceplates for translation and then to import them back again. An OpenOffice XML file is created for the export that can be edited with Microsoft Excel and other spreadsheet programs. You set the languages available for the export and import in the "Languages" tab. You export and import the text of a faceplate type using the buttons "Export project texts" and "Import project texts" in the toolbar.

Alternatively, you can export and import the texts of the faceplates using the commands "Export texts" and "Import texts" in the "Library" task card. In this case, the project languages defined for this project are used for the export and import.

Requirements

- A project is open.
- You have opened the "Faceplates" editor.
- You have created a faceplate type.
- The faceplate type contains at least one object with text, e.g. a text field.

Exporting texts

To export the texts, proceed as follows:

1. Open the faceplate type in the project library.
2. Open the "Texts" tab in the "Faceplates" editor.
   A list with the texts in the project is displayed in the work area. There is a separate column for each project language.

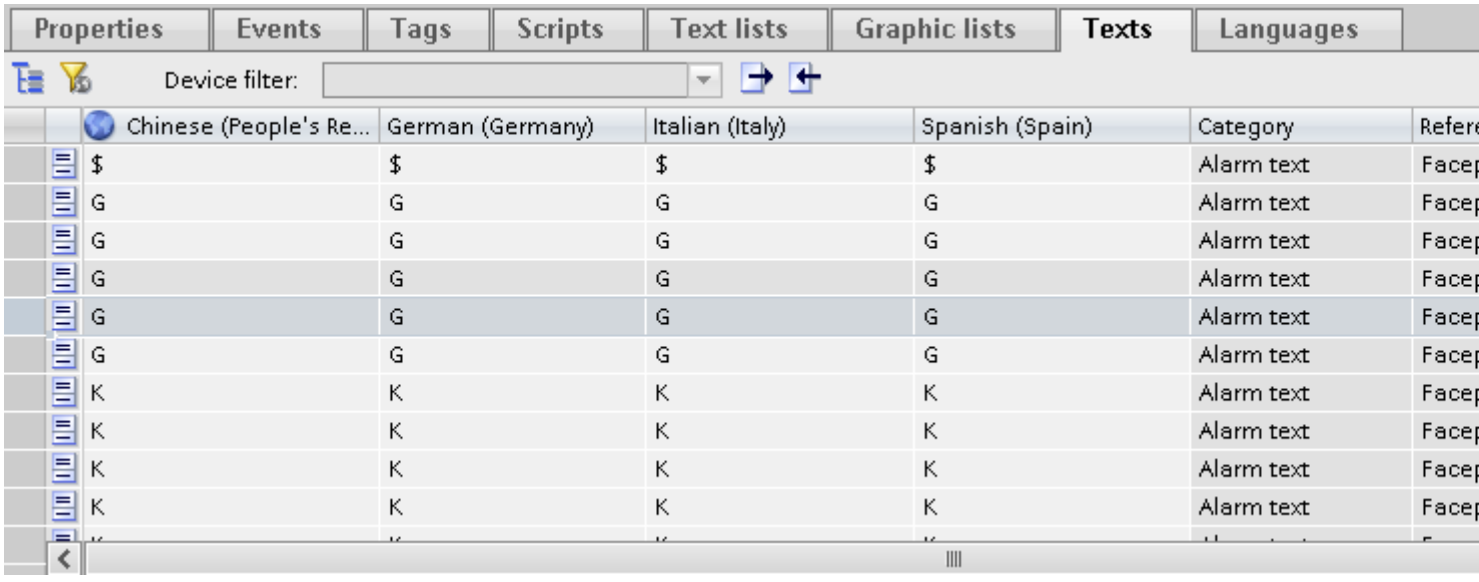

3. To export the text to an .xslx file, click "Export project texts" .
   The "Export" dialog box opens.
4. Select the source language and the target language in the dialog.
5. Select either the content selected in the current window for export or select the user texts from categories.
6. Enter the name and path for the export file.
7. Click "Export".
   Once the export is successfully concluded, the export file is stored under the specified path.

Importing texts

To import the text back into the TIA Portal after editing or translation, follow these steps:

1. Open the faceplate type in the project library.
2. Open the "Texts" tab in the "Faceplates" editor.
3. Click "Import project texts" in the toolbar.
   The "Import" dialog box opens.
4. Select the path and name of the import file in the "Select file for import" box.
   If you have made changes to the source language in the export file and would like to overwrite the entries in the project with the changes, select the "Import source language" check box.
5. Click on "Import".

Result

You have exported the text of a faceplate for translation and imported it back into the TIA Portal after translation.

1.7.2.12 Editing a faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

All faceplate types in the project library have a status. You create a faceplate type in the "in progress" status. You can edit the faceplate types as required in this status. When editing is complete, release the faceplate type.

Requirement

- A faceplate type has been created.
- The faceplate type has the version 0.0.1. and the status "in progress".
- The "Library" task card or the library view is open.

Enable faceplate type

1. Select version 0.01 of the faceplate type in the project library.
2. Select "Release version" in the shortcut menu.

You have released version 0.0.1 of the faceplate type.

Edit faceplate type

1. Select, for example, the released version 0.0.1 of a faceplate type in the project library.
2. Select "Edit faceplate type" in the shortcut menu.

The library view opens. The new version 0.0.2 of the faceplate type has been created.

The faceplate type has the "In progress" status.

Restoring the last version of the faceplate type

The last released version of the faceplate type is version 0.0.2.

Edit the faceplate type. A new version 0.0.3 is created and the "in progress" status is assigned to it.

1. Select the faceplate type from the project library.
2. Select "Discard changes and restore type" in the shortcut menu.

All changes to the faceplate type since the last enabling operation are rejected. The faceplate type is enabled again and has the version 0.0.2.

See also

Creating a faceplate type (Page 270)

1.7.2.13 Resizing a faceplate (RT Advanced, Panels, Comfort Panels)

Introduction

In the faceplate type, specify the response of the faceplate when it is resized.

Requirements

- A faceplate type has been created.
- You have opened the "Faceplates" editor.
- The Inspector window is open.

Specifying the response to resizing

1. Activate "Properties > Properties > Layout > Fit to size > Auto-size".
2. Select the "Fixed aspect ratio" setting, for example.
   This setting retains the aspect ratio when you resize the faceplate.
3. Click the faceplate in the "Libraries" task card.
4. Select "Release version" in the shortcut menu.

Result

You have specified how the faceplate responds when its size in a screen is changed. You can change the response for a particular faceplate.

Select a faceplate in the screen. Select the required setting in "Properties > Properties > Layout > Characteristics" in the Inspector window.

1.7.2.14 Resizing a faceplate (RT Professional)

Introduction

In the faceplate type, specify the response of the faceplate when it is resized.

Requirements

- A faceplate type has been created.
- You have opened the "Faceplates" editor.
- The Inspector window is open.

Specifying the response to resizing

1. Activate "Properties > Properties > Layout > Fit to size > Auto-size."
2. Select one of the following options for "Response to resizing":
   - Fixed size
     Specifies that the size of the faceplate is unchangeable.
   - Proportional
     The faceplate is scaled proportionally. The rectangle surrounding the object is not scaled.
   - None
     No restriction on resizing of the faceplate.
   - Original size
     Specifies that the size of the faceplate is unchangeable, but the size of the rectangle surrounding the object can be adapted.
   - Keep aspect ratio
     The size of the faceplate is changeable. The aspect ratio is kept during resizing.

3. Click the faceplate in the "Libraries" task card.
4. Select "Enable faceplate type" in the shortcut menu.

Result

You have specified how the faceplate responds when its size in a screen is changed. You can change the response for a particular faceplate.

Select a faceplate in the screen. Select the required setting in "Properties > Properties > Layout > Characteristics" in the Inspector window.

1.7.2.15 Creating an instance of the faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

The faceplate type is stored in the project library. If you use the faceplate type in a screen, you create an instance of the faceplate type.

Note

Note that a faceplate is always configured for a particular class of HMI devices. For example, you cannot use a faceplate type that is configured for "RT Professional" in a screen on an "RT Advanced" HMI device.

Note

The number of faceplate instances per screen is not limited. However, you should be aware of the fact that the number of faceplate instances used, or the use of scripts in the faceplate instances, will have an impact on performance.

Note

If you would like to copy a faceplate instance from one project to another project, first copy the faceplate type to the target project, followed by the faceplate instance.

Requirement

- A screen is open.
- The "Libraries" task card is opened.

- The project library is open and it contains at least one faceplate type.

  Note

  Faceplate types are also stored in global libraries. When you add a faceplate type from the global library to the screen, the system automatically saves a copy of it to the project library.

- The Inspector window is open.

Procedure

1. Move the desired faceplate type from a library to the screen by drag&drop.

Result

You have created an instance of the faceplate type. The instance is given the version number of the most recently released type.

Configure the properties of the faceplate in the Inspector window as required. To dynamize the faceplate, open the interface in the Inspector window.

See also

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

1.7.2.16 Removing faceplate from faceplate type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Cancel application, to remove specified instances from the update of the faceplate type.

Requirement

- A faceplate type has been created.
- The faceplate must be used in a minimum of one screen.

Remove faceplate object from faceplate type.

1. Select a faceplate on the screen.
2. In the shortcut menu, select "Remove faceplate object from faceplate type".

Result

The faceplate is converted into a group that is independent of the faceplate type. Modifications of the faceplate type have no effect on this group.

See also

Creating a faceplate type (Page 270)

Example: Configuring a faceplate (Page 294)

1.7.3 Dynamizing faceplates

1.7.3.1 Basics of dynamizing faceplates (RT Professional, RT Advanced, Panels, Comfort Panels)

Use

You can dynamically control events and properties of faceplates in two ways:

- Dynamically controlling faceplates
  You configure the events or dynamic properties individually for the application point on the faceplate. To do this, define in the "Faceplates" editor that these properties and events are configurable in the faceplate. To dynamize the faceplates in the "Screens" editor, use the scripts and tags that are created in the project.
- Dynamizing the faceplate type
  You dynamize the objects which are contained in the faceplate type in the "Faceplates" editor. You configure the individual objects as in the "Screens" editor. Tags and scripts are available in the "Faceplates" editor to dynamize properties and events. You do not have access to the tags and scripts of the project within the faceplate.
  Every faceplate created with the faceplate type has the same preconfigured dynamization. You can edit this dynamic control only in the "Faceplates" editor.

Note

Dynamization of instances of a faceplate type in a group

You are using the instance of a faceplate type in an object group. The properties of the instance are also displayed as properties of the group. Any dynamization with tags, scripts or animations of the group is not displayed in Runtime.

Animation

WinCC provides preconfigured dynamic control in the "Animations" task card. You use animations to dynamize faceplates and faceplate types.

See also

Dynamizing faceplates (Page 293)

Example: Configuring a faceplate (Page 294)

Basics on faceplates (Page 265)

1.7.3.2 Dynamizing faceplates (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You dynamize a faceplate in exactly the same way as you dynamize an object from the "Tools" task card.

You connect the dynamic properties of the faceplate in the "Screens" editior with a tag or a script which supplies values to the property in runtime.

You Include tags and scripts which you have created in the project for the faceplate.

Requirements

- A faceplate is inserted in the screen.

Procedure

1. Select the faceplate.
2. In the Inspector window, open "Properties > Animation".
   The animations available for the selected object are displayed.
3. Select "Horizontal movement" and click the button.
   The parameters of the animation are displayed.
   A transparent copy of the object is shown in the work area, which is connected to the source object by means of an arrow.
4. Select a tag for control of movement.
5. Move the object copy to the relevant destination. The system automatically enters the pixel values of the final position in the Inspector window.
6. Customize the range of values for the tag as required.

Result

In Runtime, the object moves in response to every change in value of the tag that is used to control the movement. The direction of movement corresponds to the configured type of movement "horizontal".

See also

Basics of dynamizing faceplates (Page 292)

Example: Configuring a faceplate (Page 294)

1.7.4 Example: Creating and dynamizing faceplates

1.7.4.1 Example: Configuring a faceplate

Introduction

In this example you create a faceplate in which you can convert the value of a length measuring system from kilometers into meters. The length in meters is displayed in an output field.

Procedures overview

The example is divided into the following steps:

1. Creating a faceplate type
2. Configuring a faceplate type
3. Creating a script in the faceplate type
4. Connecting a script with the contained objects of the faceplate type
5. Creating a faceplate and connecting with a tag

See also

Basics on faceplates (Page 265)
Example: Creating a faceplate type (Page 294)
Example: Configuring a faceplate type (Page 295)
Example: Creating a script in the faceplate type (Page 296)
Example: Creating an instance of the faceplate type (Page 298)
Example: Configuring included objects (Page 300)

1.7.4.2 Example: Creating a faceplate type

Task

You create a faceplate type.

Requirement

- A screen is open.
- The toolbox window is displayed.

Settings

For the example you require objects with the following settings:

Object	Object name	Properties
Text field	Label_Meter	Text: Meter
I/O field	Output_Meter	Mode: Output
Button	KM_to_Meter	Text OFF: Convert

Procedure

1. In the toolbox, click the individual objects and drag&drop the objects into the screen.
2. Set the properties as shown above.
3. Select all objects.
4. Select the "Create faceplate type" command in the shortcut menu of the multiple selection. The "Add type" window opens.
5. Enter "KMtoMeter" as type name.

Result

The faceplate type appears in the project library under the name "KMtoMeter". The faceplate type is assigned the "in progress" status.

See also

Example: Configuring a faceplate (Page 294)

1.7.4.3 Example: Configuring a faceplate type

Job

You create the dynamic property and tag of the faceplate type.

Creating a new property

1. Click "Properties" in the configuration area of the faceplate.
2. In the "Interface" list, click the icon "Add property".
   A new property is added to the "Interface" list.

3. Double click the name of the property and enter "KMToMeter".
4. Click the selection list and select the data type "Int".

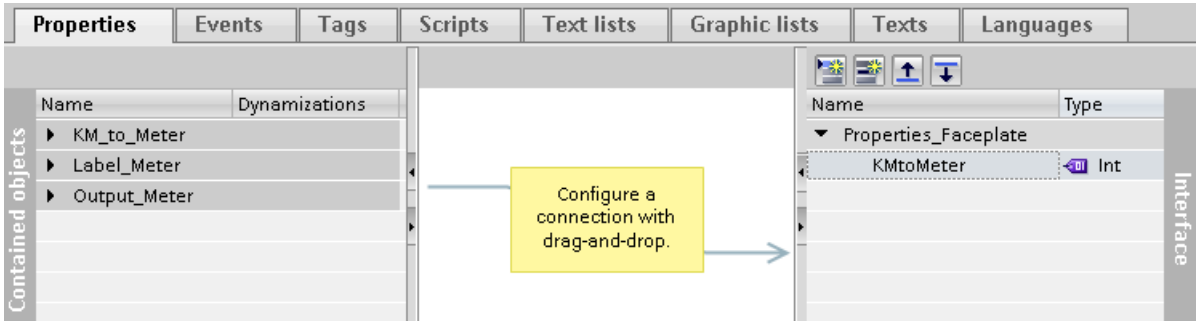

Interim result

The "KMToMeter" property is displayed in the "Interface" list. The symbol identifies the dynamic property. The property will be linked to a tag in the next step.

Creating a tag in the faceplate type

1. Click "Tags" in the configuration area.
2. Click "Add" in the "HMI tags" table. A new tag is added to the table. The inspector window of the tag is opened.
3. In the Inspector window, select "Properties > Properties > General":
4. Enter "BB_Tag" as the name. Select the "Int" data type.

Result

The "KMToMeter" property and the "BB_Tag" of the faceplate type are created in the configuration area of the "Faceplates" editor.

See also

Example: Configuring a faceplate (Page 294)

1.7.4.4 Example: Creating a script in the faceplate type (RT Advanced, Panels, Comfort Panels)

Task

You create a script that converts the value of a length measuring system from kilometers to meters.

Procedure

1. Click "Scripts > Faceplate scripts" in the configuration area.
2. Double click "Add VB script".
3. Press the shortcut keys <CTRL+J>.
   The object list opens.
4. Click "HMI tags".
   The faceplate tag "BB_Tag" is displayed in the object list.

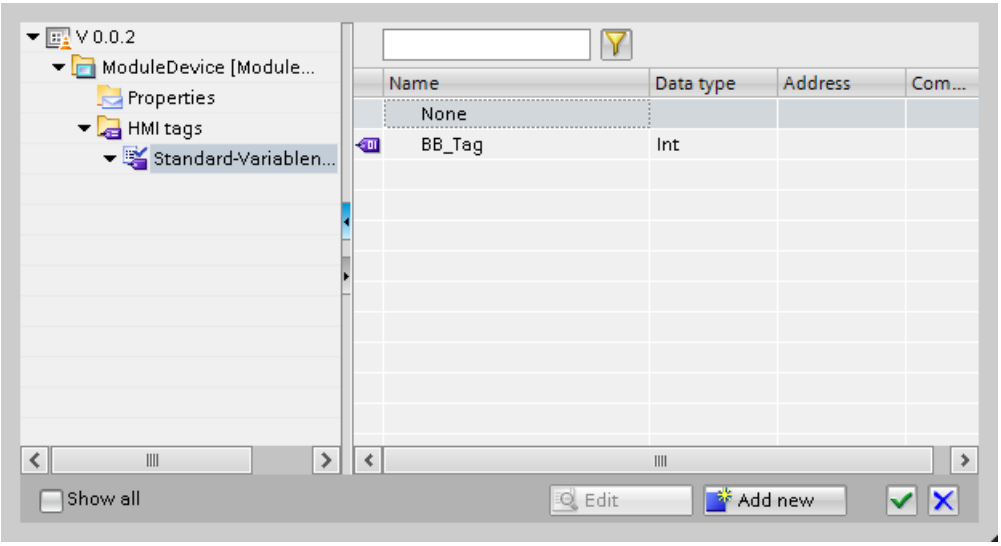

5. Select the faceplate type tag "BB_Tag". Confirm the selection.
6. Insert an "=" sign.
7. Press the shortcut keys <CTRL+J>. The object list is opened.
8. Click "Properties."
   The dynamic property "KMtoMeter" is displayed in the object list.

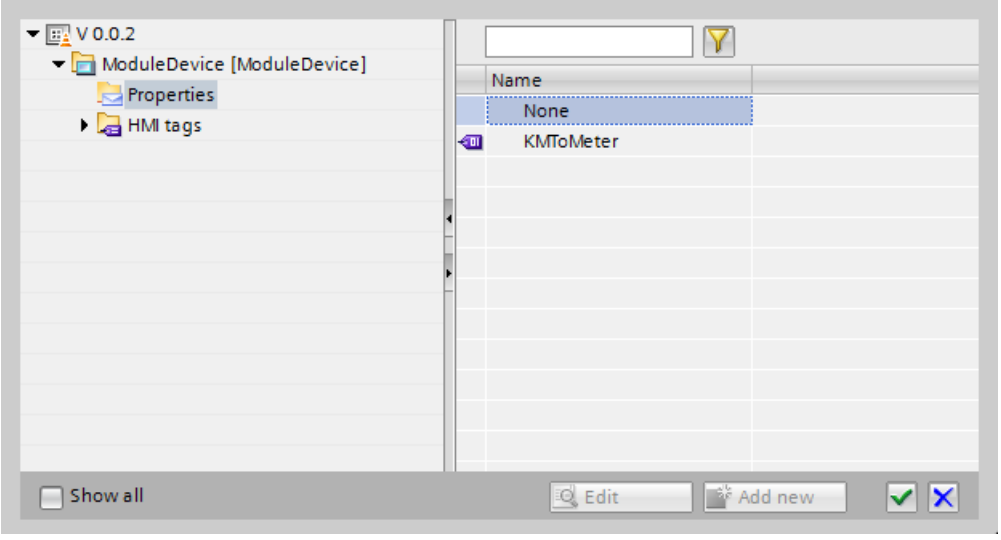

9. Select the "KMToMeter" dynamic property. Confirm the selection.

10. Enter "*1000" at the end of the code. This corresponds to the conversion factor from kilometers to meters.
11. Enter the name "Script_1" under "Properties > Properties > General >Setting > Name" in the Inspector window.

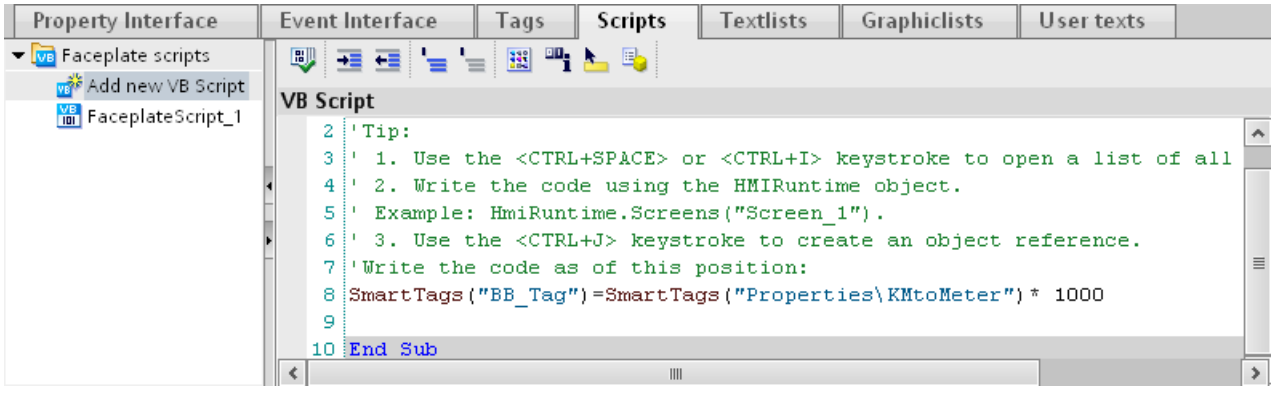

Result

The script has been created in the faceplate type. If you use the "KMtoMeter" faceplate type in a screen, you assign a tag to the "KMtoMeter" property. The value of this tag is multiplied by factor 1000 and assigned to the tag of the faceplate type "BB_Tag" as a new value. In this way you supply values to the tags of the faceplate type.

See also

Example: Configuring a faceplate (Page 294)

1.7.4.5 Example: Creating an instance of the faceplate type

Task

You insert the faceplate type "KMtoMeter" in a screen and assign a tag to the dynamic property "KMtoMeter".

Requirement

- The "Screens" editor is open.
- A new screen has been created.

Settings

For the example you require a tag with the following settings:

Name	PLC connection	Type
Kilometer	No	ULong

Procedure

1. Create the HMI tag "Kilometer" with the settings named above.
2. Open the "project library" in the "Libraries" task card.
3. Move the "KMtoMeter" faceplate type into the screen by drag&drop.
4. Click "Properties > Properties > Interface" in the Inspector window.
   The table shows all the properties which you can dynamize in the faceplate.

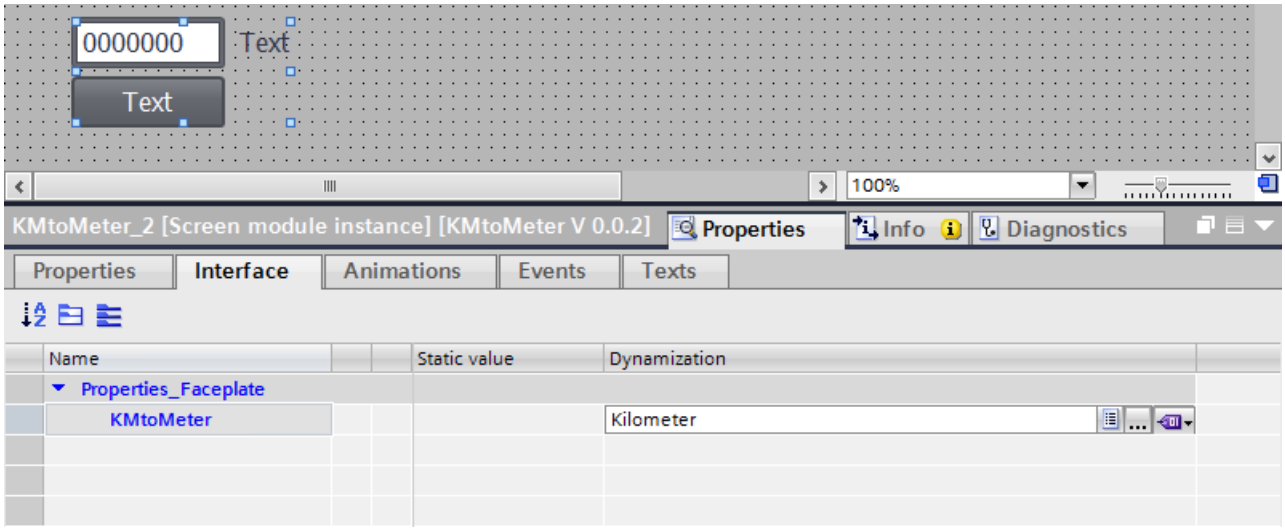

5. Click the "Dynamization" column in the "KMtoMeter" line.
6. Select "HMI tag".
7. Click the "..." button.
8. Select the tag "Kilometer" from the object list.
9. Confirm the selection.

Result

The dynamic property "KMtoMeter" is linked with the "Kilometer" tag. During runtime the dynamic property is supplied with values of the "Kilometer" tag. The values are multiplied by factor 1000 in the script of the faceplate type and assigned to the tag in the faceplate as a new value.

See also

Example: Configuring a faceplate (Page 294)

1.7.4.6 Example: Configuring included objects

Task

You connect the included objects in the faceplate "KMtoMeter" to a faceplate tag and a faceplate script.

You connect the I/O field with the "BB_Var" tag of the faceplate type. You connect the "Click" event for a button to the script "Script_1".

Requirement

The faceplate is displayed in the "Faceplates" editor.

Interconnecting I/O field with a tag

1. Select the "Output_Meter" I/O field in the faceplate.
2. Connect the I/O field with the faceplate tag "BB_Tag" in the "General > Process > Tag" inspector window.

Connecting an event to a faceplate script

1. Select the "KM_to_Meter" button in the faceplate.
2. Select "Script_1" in the inspector window under "Events > Click".

Result

The I/O field is connected with the tag of the faceplate type. The button is connected with the script of the faceplate type.

If you click the button of the faceplate during runtime, the script is executed. The value of the tag of the faceplate is output in the I/O field.

See also

Example: Configuring a faceplate (Page 294)

1.8 Configuring screen navigation

1.8.1 Screen navigation basics

1.8.1.1 Basics on screen navigation (RT Advanced, Basic Panels, Panels, Comfort Panels)

Types of navigation for the screen change

For a production process consisting of multiple subprocesses, you will configure multiple screens. You have the following options to enable the operator to switch from one screen to the next in Runtime:

- Assign buttons to screen changes
- Configuring screen changes at local function keys

Procedure

Before you create a screen change, define the plant structure and derive from it the screen changes that you want to configure.

Create the start screen under "Runtime Settings > General > Start screen".

See also

Basics on screen navigation (Page 301)

Assigning screen change to button (Page 302)

Assigning screen change to function key (Page 304)

Overview of working with menus and toolbars (Page 305)

Creating a menu (Page 309)

Screen basics (Page 37)

1.8.1.2 Basics on screen navigation (RT Professional)

Types of navigation for the screen change

For a production process consisting of multiple subprocesses, you will configure multiple screens. You have the following options to enable the operator to switch from one screen to the next in Runtime:

- Assign buttons to screen changes

- Configuring a screen change at a menu and toolbars

Note

For some HMI devices, you can alternatively create buttons for the screen change in the permanent area. In this case, the button is incorporated into every screen.

Procedure

Before you create a screen change, define the screen hierarchy based on the plant structure and derive from it the screen changes that you want to configure.

Create the start screen under "Runtime Settings > General > Start screen".

See also

Basics on screen navigation (Page 301)

Example: Configuring toolbars and buttons (Page 320)

1.8.1.3 Assigning screen change to button (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Configure a button in the screen to switch between the screens on the HMI device during operation.

Note

If you have set the "Visibility" of animations to "Hidden" in the Inspector window of a screen, this screen cannot be called up in Runtime.

Requirements

- You have created the project.
- You have created the "Screen_2" screen.
- "Screen_1" is created.

Procedure

1. Double-click "Screen_1" in the project navigation. The screen is displayed in the work area.
2. Move "Screen_2" from the project tree to the open screen by drag&drop.
   A button with the name "Screen_1" is inserted.

3. In the Inspector window, select "Properties > Events > Click".
   The "ActivateScreen" system function is displayed in the "Function list".

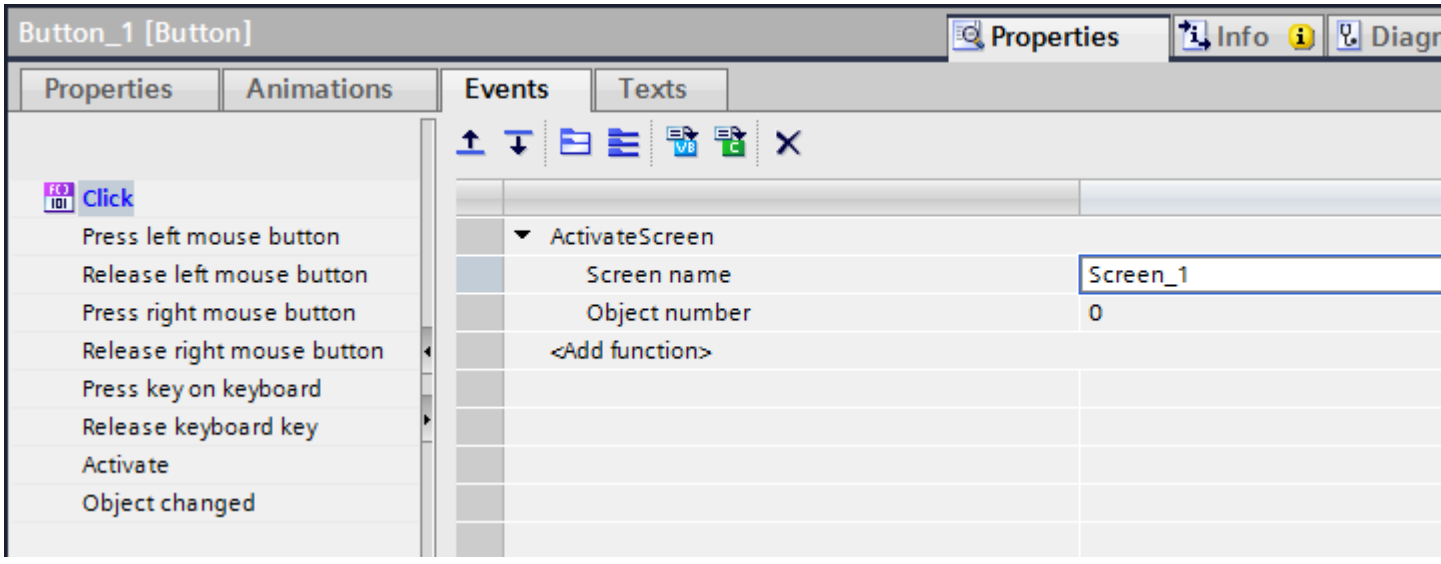

4. At the "Object number" attribute, define, if required, the tab sequence number of the object on which the focus is to be set after a screen change. You can also specify a tag that contains the object number.

Alternative procedure

1. Move a button from the "Tools" task card to "Screen2" by drag&drop.
2. In the Inspector window, select "Properties > Events > Click".
3. Select the "ActivateScreen" system function.
4. Select "Screen_2" for the "Screen number".

Result

The operator goes to "Screen_1 with the button in Runtime. If you have specified an object number, the object with this object number has the focus following a screen change.

See also

Basics on screen navigation (Page 301)

1.8.1.4 Assigning screen change to function key (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Configure a screen change function key in the screen to switch between the screens on the HMI device during operation.

Note

If you have set the "Visibility" of animations to "Hidden" in the inspector window of a screen, this screen cannot be called up in Runtime.

Requirements

- You have created a project.
- You have created the "Screen_2" screen.
- You have created the "Screen_1" screen.

Procedure

1. Double click "Screen_1" in the project tree. The screen is displayed in the work area.
2. Move "Screen_2" from the project tree to a function key, e.g. "F2".
   The configured function key displays a yellow triangle.
3. Click "Properties > Events > Press key" in the inspector window.
   The "ActivateScreen" system function is displayed.

Result

The operator goes to the specified "Screen_2" with function key "F2" in Runtime.

See also

Basics on screen navigation (Page 301)

1.8.2 Menus and toolbars for Runtime Professional (RT Professional)

1.8.2.1 Basics (RT Professional)

Overview of working with menus and toolbars (RT Professional)

Introduction

Use the "Menus and Toolbars" editor to configure customized menus and toolbars. The customized menus and toolbars are displayed in all screens of a project and in the screen windows. You connect the menu commands and icons with the local scripts.

Application

You use customized menus and toolbars, for example to implement a screen navigation. The customized menus and toolbars are displayed in each screen of a project or in a screen window. You can thus move from any screen to all the screens for which you have configured screen changes at menu entries or icons.

See also

Menu basics (Page 305)

Configuration (Page 307)

"Menus and Toolbars" editor (Page 307)

Basics on screen navigation (Page 301)

Example: Configuring toolbars and buttons (Page 320)

Menu basics (RT Professional)

Introduction

A menu is placed at the top margin of each screen beginning on the left. In a configuration file you specify which menu and toolbars are created with this file in screens or screen windows. You can configure one menu line for each configuration file.

Menu structure

The configuration options for a menu element depend on the position where the menu is located in the menu structure.

You create the following components for a menu:

- Main menu command
  - Is displayed in the screen.
  - Opens a submenu.
- Menu item
  - Is displayed in the screen when the main menu opens.
  - Executes a defined procedure when clicked.
  - You can enter a parameter (for example a screen name) that is to be passed to the procedure under "Properties > Events."
  - Opens a submenu, if it exists.
- Submenu command
  - Is displayed in the screen when a higher-level menu item is open.
- Separator lines between the menu entries

The following figure shows a typical menu structure with different menu elements:

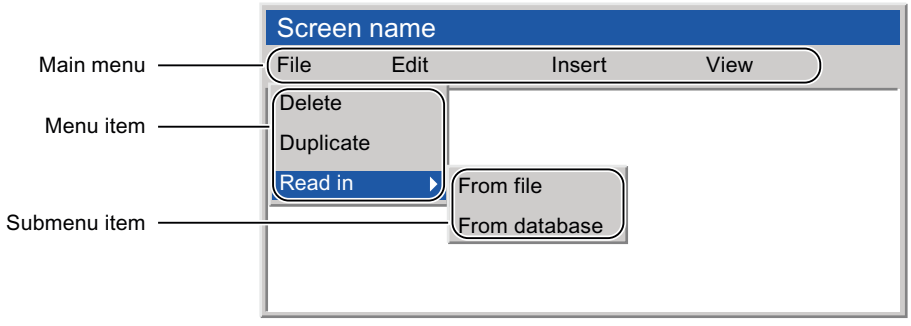

Editing options

You have the following options for configuring a menu:

- Adding a new main menu
- Adding a new menu command / submenu command
- Adding a separator line
- Moving a selected menu command to the left
- Moving a selected menu command to the right
- Changing settings in the Inspector window

See also

Overview of working with menus and toolbars (Page 305)

Example: Configuring toolbars and buttons (Page 320)

Configuration (RT Professional)

Introduction

The menus and toolbars that you have configured are combined in the configuration files. Each configuration file contains a maximum of one menu, but can contain several toolbars. You can create any number of configuration files. A configuration file is selected for the following reasons:

- To add a menu and a toolbar to a screen window
- To add a menu and a toolbar to all the screens of your project
- To exchange the menus and toolbars in Runtime dynamically

Configuring a configuration file

You have the following options for configuration a configuration file:

- Assign authorizations
  The elements are automatically disabled if a logged in user does not have the required authorization.
- Hiding or disabling menu entries and icons
  You can also exchange the configuration file, for e.g. in case of user change during runtime, if you save the modified functional scope in a new configuration file.

See also

Overview of working with menus and toolbars (Page 305)

Example: Configuring toolbars and buttons (Page 320)

"Menus and Toolbars" editor (RT Professional)

Introduction

Use the "Menus and Toolbars" editor to create, configure and edit customized menus and toolbars. The customized menus and toolbars are combined in the configuration files and are then displayed in all the screens of a project as well as in the screen windows. You connect the menu commands and icons with the local scripts.

Layout

The "Menus and Toolbars" editor has three tabs:

- Toolbars
- Menus
- Configurations

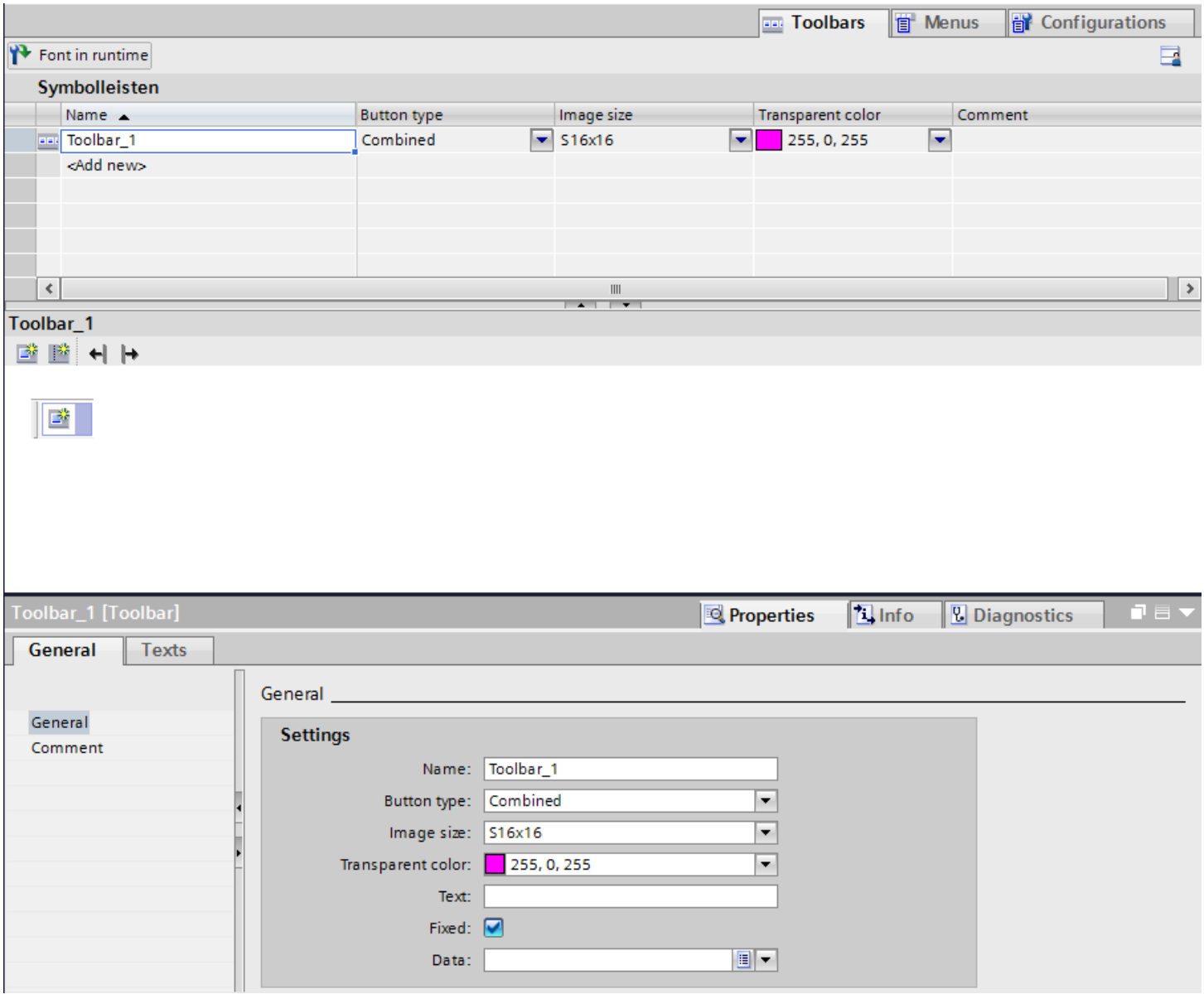

You open the editor in the project navigation by double-clicking "Screen management > Menus and Toolbars".

Work area

The work area is divided into two sections in the "Toolbars" and "Menus" tabs:

- In the table you define the complete toolbar and/or complete menu.
- The lower section offers you a preview of your toolbars or menus. You create and configure the individual icons and/or menu commands and specify the structure, labelling texts and symbolizing graphics.

The "Configurations" tab has an undivided work area that contains the table editor for selecting the corresponding toolbars and the corresponding menus.

Inspector window

You configure the menus and toolbars in the Inspector window under "Properties > Properties."

See also

Overview of working with menus and toolbars (Page 305)

Example: Configuring toolbars and buttons (Page 320)

1.8.2.2 Working with menus and toolbars (RT Professional)

Creating a menu (RT Professional)

Introduction

In the "Menus and Toolbars" editor you create a menu with main menu commands and the corresponding menu commands with up to six hierarchy levels. The menu is shown as it is displayed later in the screens or a screen window.

Requirement

- The "Menus and Toolbars" editor is open.
- The "Menus" tab is open.

Creating a menu

1. Double-click "Add" in the "Menus" table.
   The first main menu command is created in the preview of the menu commands.
2. Change the name of the menu and enter a comment, if appropriate.

Creating main menu commands

1. Click in the preview of the menu on the already created main menu command.
   Four yellow stars are displayed around the selection rectangle.

2. Click the yellow star to the right of the standard text.
   A further main menu command is created.

3. Enter a name under "Properties > Properties > General >Name" in the Inspector window.
4. Enter a label for the main menu command "Properties > Properties > General >Text" in the Inspector window.

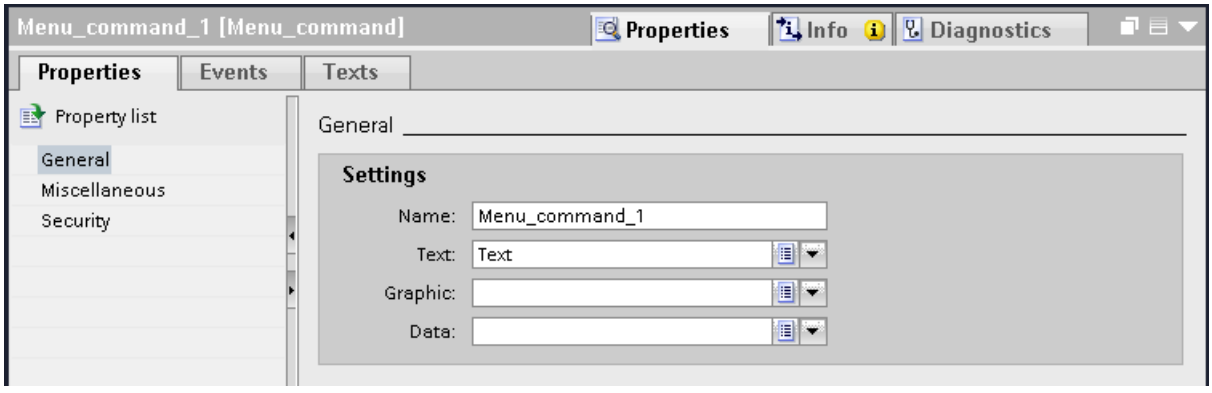

1. To display the main menu command as visible in Runtime, activate "Properties > Properties > Miscellaneous > Visibility".
2. To define an operator authorization, select an authorization under "Properties > Properties > Security".

Creating menu commands

1. Click in the preview on a main menu command.
   Yellow stars are displayed around the selection rectangle.

2. Click the yellow star below the selection rectangle.
   A menu command is created under the main menu command.

3. Configure the menu command in the Inspector window like a main menu command.

Note

The authorization, visibility and activation for Runtime can be configured both in the menu as well as in each individual menu command. In the case of differing configurations the following applies: Menu overwrites menu command.

Creating submenu commands

1. Click in the preview on a menu command.
   Yellow stars are displayed around the selection rectangle.

2. Click the yellow star to the right of the menu command.
   A submenu command is created.

3. Configure the submenu command in the Inspector window like a main menu command.

Result

The menu is created and can be selected as the menu for a specific configuration file in the "Configurations" tab.

See also

Creating a toolbar (Page 312)
Creating a configuration file (Page 315)
Editing menus and toolbars (Page 317)
Configuring menus and toolbars for all the screens (Page 318)
Configuring menus and toolbars in a screen window (Page 318)
Basics on screen navigation (Page 301)
Example: Configuring toolbars and buttons (Page 320)
Adapting menus and toolbars in runtime (Page 319)
Connecting a menu command or icon to a procedure (Page 314)

Creating a toolbar (RT Professional)

Introduction

Use the "Menus and Toolbars" editor to create a toolbar with buttons. The toolbar is shown in the preview as it is displayed later in the screens or a screen window.

You can configure any number of toolbars for each configuration. For each toolbar you specify the placement position of the toolbar in the screen. Further, you can also define whether the user should have the option of freely placing the toolbar.

Note

Toolbars are device-specific due to the connected scripts. The functionalities are lost when they are used for a different device.

Requirement

- The "Menus and Toolbars" editor is open.
- The tab "Toolbars" is displayed.

Creating a toolbar

1. Double-click "Add" in the "Toolbars" table.
   A toolbar is created and has a pre-assigned name given to it.
   A button is created in the preview of the toolbar.
2. Enter a name to the toolbar in the "Name" column.

3. Select one of the following options as "Button type":
   - Text and graphics combined
   - Graphics only
   - Text only
4. Specify the size of the buttons in the "Screen size" column.

Adjusting buttons

1. Select the toolbar for which you want to create buttons.
2. Click the already existing button in the preview of the toolbar.
   The button contains a default text and/or a default graphic depending on the type.

The properties are displayed in the Inspector window.

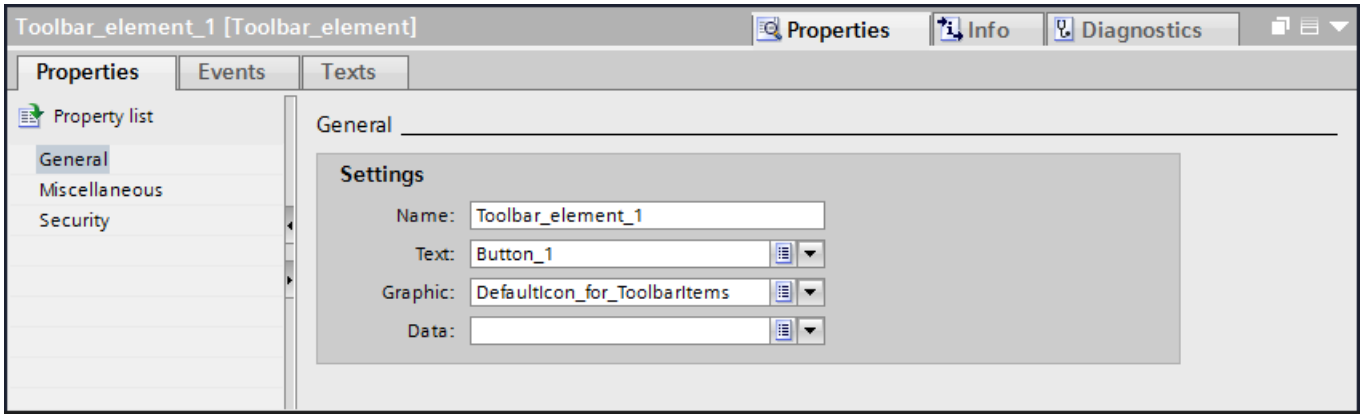

3. Enter a name under "Properties > Properties > General" in the Inspector window.
4. Depending on the button type, select a text and/or a graphic for the button design.
5. To display the button as visible in Runtime, activate "Properties > Properties > Miscellaneous > Visibility".
6. To display a tooltip in Runtime, enter a text under "Properties > Properties > Miscellaneous > Tooltip".

7. To define an operator authorization, select an authorization under "Properties > Properties > Security".
8. To insert further buttons, click in the preview of the toolbar on the symbol.

Note

The authorization, visibility and activation for Runtime can be configured both in the toolbar as well as in each individual button. The following rule is valid for different configurations: Toolbar overwrites button.

Result

A toolbar with the define toolbars has been created and can be selected as the menu for a specific configuration file in the "Configurations" tab.

See also

Creating a menu (Page 309)

Example: Configuring toolbars and buttons (Page 320)

Connecting a menu command or icon to a procedure (RT Professional)

Introduction

Use the "Menus and Toolbars" editor to configure customized menus and toolbars. The customized menus and toolbars are displayed in all the screens of a project and/or also in a screen window. The individual menu commands and buttons are connected with a function list. Only "Click" is available as an event.

Requirement

- A "Screen_1" screen has been created.
- A "Screen_2" screen has been created.
- The "Menus and Toolbars" editor is open.
- A button or menu command is created.

Configuring an event with a system function to a button or menu command.

1. Select the button or menu command.
2. In the Inspector window, select "Properties > Events > Click".

3. Select the "ActivateScreen" system function from the function list.
4. Select "Screen_2" as the "Screen name".

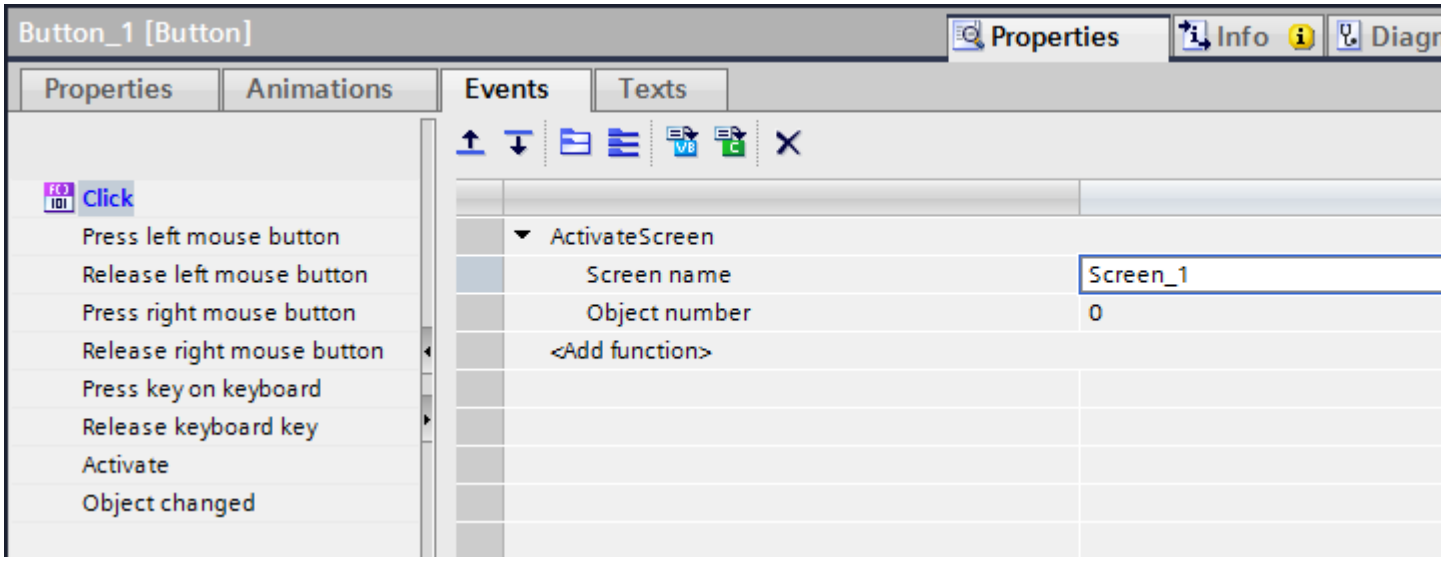

Configuring an event to a button or menu command via scripts.

1. Select the button or menu command.
2. In the Inspector window, select "Properties > Events > Click".
3. Click the icon "VB script" in the toolbar of the Inspector window.
4. Write the required script.
   The "Item" is the object that triggers the event, in this case the button or the menu command. It is specified using the "Name" property in the script.

Result

When the operator clicks the button or the menu command, the configured function is executed in Runtime.

See also

Creating a menu (Page 309)

Example: Configuring toolbars and buttons (Page 320)

Creating a configuration file (RT Professional)

Introduction

Use a configuration file to display menus and/or toolbars in all screens of your project or in a screen window. A configuration file consists of one/no menu and/or any number of toolbars.

Requirement

- The "Menus and Toolbars" editor is open.
- The "Configurations" tab is displayed.
- Menus and toolbars are required.

Creating a configuration file

1. Double-click "Add" in the "Configurations" table. A configuration file is created.
2. Change the name of the configuration file and enter a comment, if appropriate.
3. Select a name under "Properties > Properties > Menu >Name" in the Inspector window.

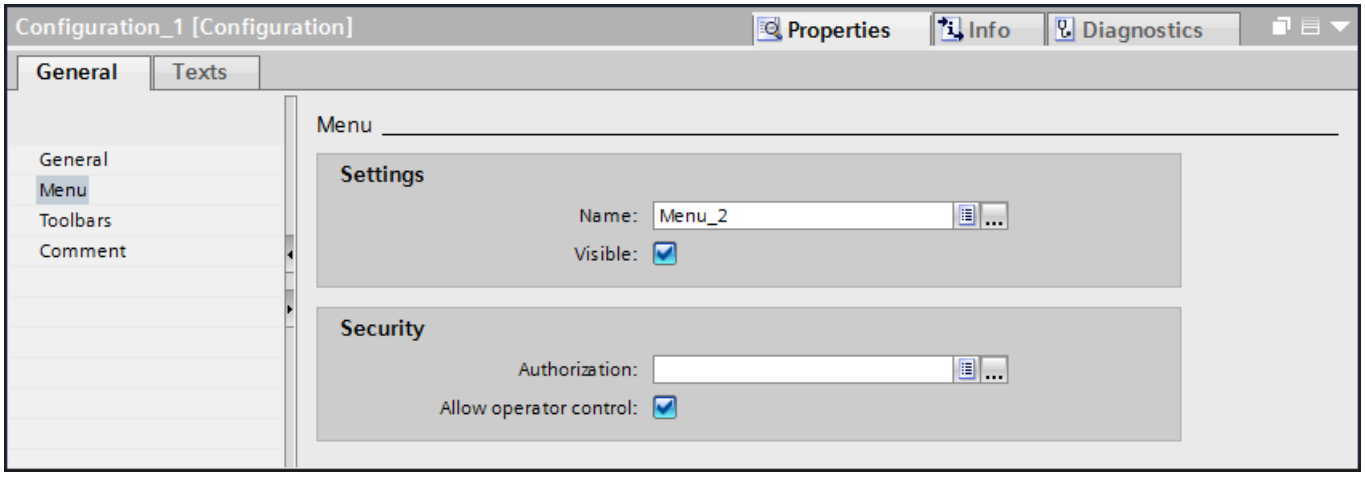

4. Specify whether the menu is to be activated and visible in Runtime and which authorization is required for operation.
5. Select the toolbars you want to add to the configuration file under "Properties > Properties > Toolbars" in the Inspector window.

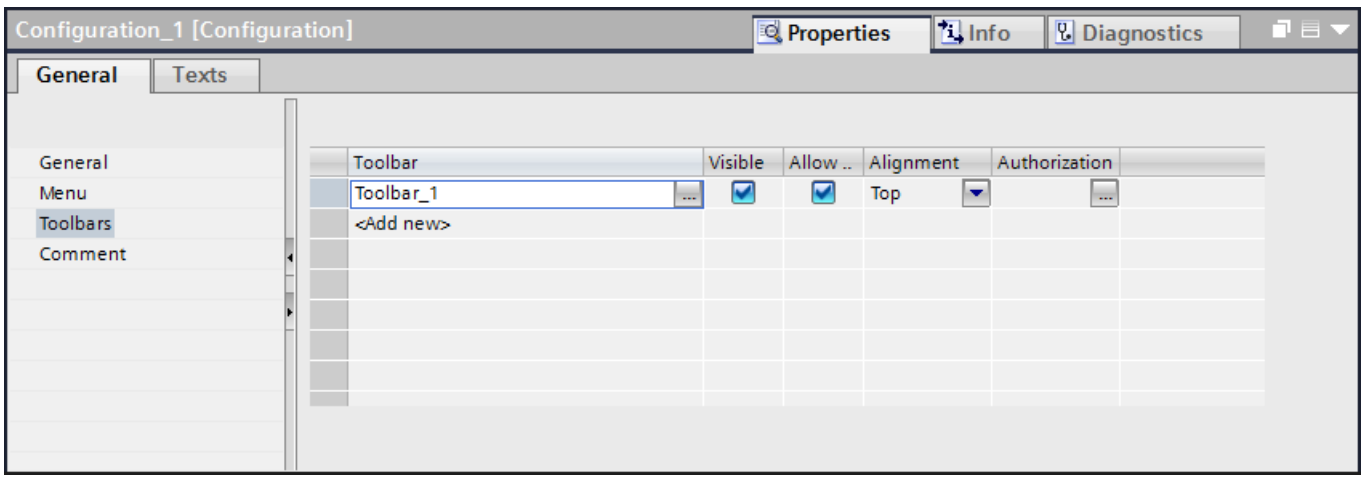

6. Specify for each selected toolbar whether or not the toolbar is to be activated and visible in Runtime and which authorization is required for operation.
7. You define the alignment for each toolbar in "Properties > Properties > Toolbars" in the Inspector window.

Result

The configuration file can be selected in the project for all screens or in a screen window.

See also

Creating a menu (Page 309)

Example: Configuring toolbars and buttons (Page 320)

Editing menus and toolbars (RT Professional)

Introduction

You edit and change already created menus and toolbars in the work are of the "Menus and Toolbars" editor".

Inserting a separation line

1. Open the "Menus" tab or Toolbars" tab.
2. Select the menu command or button in front of which you want to insert a separation line.
3. Click .
   A separation line is shown before the selected menu item.

You can alternatively insert separation lines by means of the shortcut menu and the command "Add separator."

Moving a menu command / button

1. Open the "Menus" tab or Toolbars" tab.
2. Select the menu command or the button you wish to move.
3. Click one of the symbols or .
   The menu command or button is moved one place to the left or to the right.

Alternatively, you can move a selected element by dragging and dropping it onto the desired position. A marking displays the selected target position.

Showing and hiding menus

1. Open the "Menus" tab.
2. To display all the menu commands, click .
3. To display only the main menu commands, click the symbol.
   Only the main menu is displayed.

See also

Creating a menu (Page 309)

Example: Configuring toolbars and buttons (Page 320)

Configuring menus and toolbars for all the screens (RT Professional)

Introduction

In the Runtime settings you can define a configuration file with a user-defined menu and toolbars as standard. The customized menu and toolbars are displayed in each screen at the configured position.

You can use VB Script to load other configuration files in Runtime. For e.g. when another user logs in, you can load a configuration file with modified scope of customized menus and toolbars.

Configuring menus and toolbars for all the screens

1. Open the "Runtime settings" editor.
2. Select the configuration file whose menu and toolbars are to be displayed in all the screens of your project under "General > Menus and toolbars."

Result

When you activate a project, all customized menus and toolbars will be displayed in all the screens of the project.

See also

Creating a menu (Page 309)

Example: Configuring toolbars and buttons (Page 320)

Configuring menus and toolbars in a screen window (RT Professional)

Introduction

Independent of the start configuration, you can load, in a screen window, an additional configuration file with customized menus and toolbars.

The customized menus and toolbars of the screen window and the start configuration can thus be displayed in the screen window.

Requirement

- You have created a configuration file.
- A screen is open.

How to configure menus and toolbars in a screen window

1. Drag-and-drop the "Screen window" object into the screen.
2. In the Inspector window, select the configuration file in "Properties > Properties > General".

Result

The customized menus and toolbars are displayed in the screen window during runtime.

See also

Creating a menu (Page 309)

Example: Configuring toolbars and buttons (Page 320)

Adapting menus and toolbars in runtime (RT Professional)

Introduction

Use the "Menus and Toolbars" editor to configure customized menus and toolbars. The customized menus and toolbars are displayed in all screens of a project and in the screen windows. You connect the menu entries and icons with procedures from the local scripts.

Example

You can dynamize the "MT configuration" property in a screen window. It is also possible to change the configuration of the basic screen and the screen window by means of VB script.

The following example shows a procedure in which the configuration file is transferred as a parameter.

```
Sub ChangeMenuToolbarConfigFile (ByVal strMTConfigFile)
HMIRuntime.MenuToolBarConfig = strMTConfigFile
End Sub
```

Note

One menu and several toolbars can be configured in every configuration.

You can make individual elements of menus or buttons inactive or invisible. This allows display of an adapted functional scope in various screens by editing the configuration and saving it under a different name.

See also

Creating a menu (Page 309)

Example: Configuring toolbars and buttons (Page 320)

1.8.2.3 Example: Configuring a screen change with toolbars (RT Professional)

Example: Configuring toolbars and buttons (RT Professional)

Introduction

Your process is split into six process image partitions. One screen is the start screen that does not have a higher-level screen.

Requirement

- The screens of the project have been created.
- The "Menus and Toolbars" editor is open.
- The "Toolbars" tab is opened.

Creating a toolbar for a start screen

1. Click "Add" in the "Toolbars" table. An toolbar is created.
2. Enter the name "Screen_Change_Start_Screen" in the column.
3. Select the setting "Combined" as the "Button type".

4. Select the screen size "S48x48".

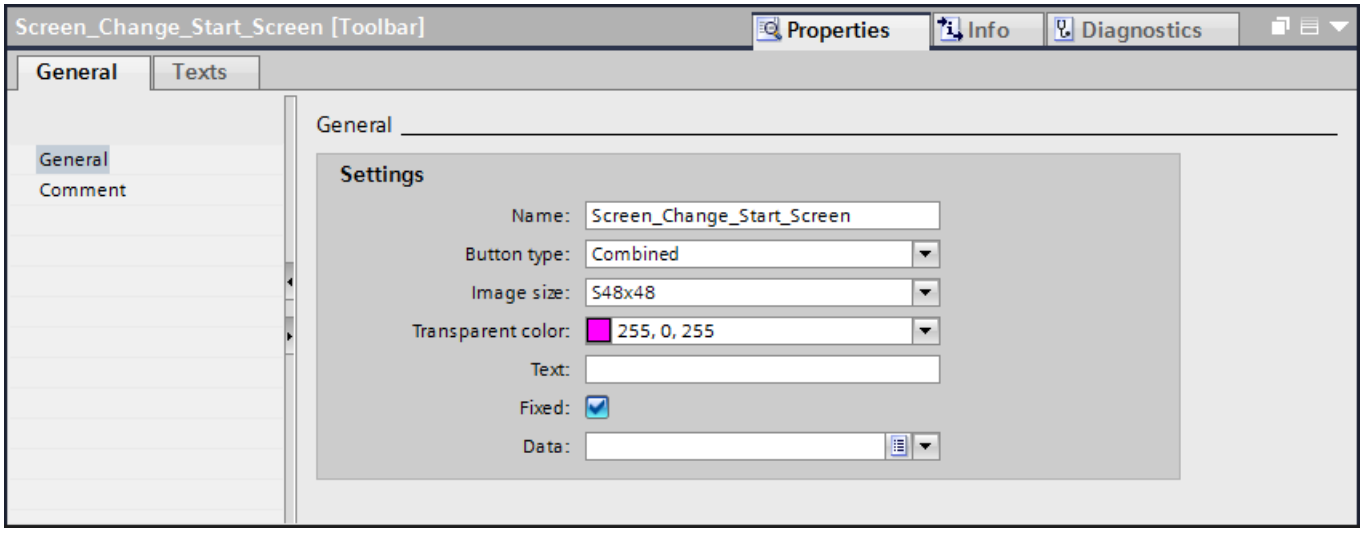

5. Repeat the procedure: Create a toolbar with a corresponding name for each screen.

Creating buttons for a start screen

1. Select the first button and select "Properties > Properties > General" in the Inspector window.

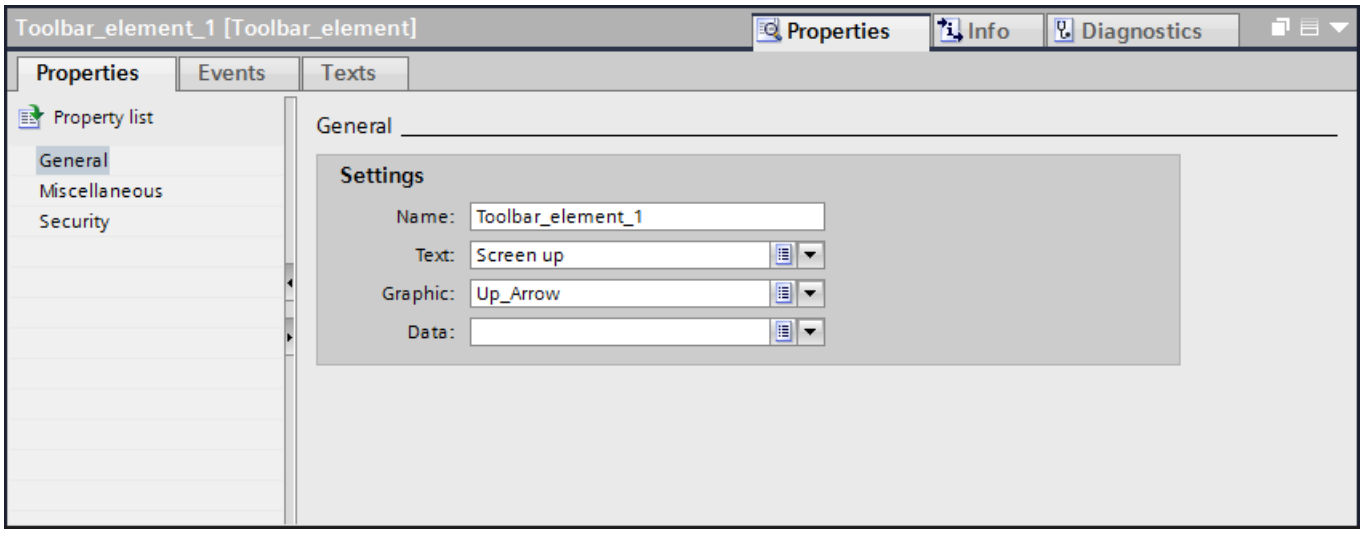

2. Enter the text "Screen up" under "Text".
3. Select the graphic "Up_Arrow" under "Graphic".
4. To display the button as visible in Runtime, activate "Properties > Properties > Miscellaneous > Visibility".
5. Enter the text "Changes to parent screen" under "Properties > Properties > Miscellaneous > Tooltip".

6. Click in the preview of the toolbar.

A button is created.

7. Repeat the procedure for all the buttons with the following properties:
   - Button_2: Text: "Screen down"; Graphic: "Down_Arrow".
   - Button_3: Text: "Screen left"; Graphic: "Left_Arrow".
   - Button_4: Text: "Screen right"; Graphic: "Right_Arrow".
   - Button_5: Text: "Start screen"; Graphic: "Home".

Result

A specific toolbar is now available for each screen.

Further steps

The following steps are still necessary to use the toolbars for the screen change:

- Assigning system functions to buttons
- Inserting toolbars into a configuration file
- Configuring in the screen window

See also

Example: Configuring functions at buttons (Page 322)

Example: Creating a configuration file (Page 324)

Example: Configuring a configuration file in the screen window (Page 325)

Basics on screen navigation (Page 301)

Example: Configuring functions at buttons (RT Professional)

Introduction

In order for each button to execute the desired function during a screen change, configure the system functions using the function lists.

Requirement

- The toolbars for all the screens have been created.
- Buttons for the corresponding screens have been created for each toolbar.
- The "Menus and Toolbars" editor is open.
- The "Toolbars" tab is opened.

Configuring system functions at a button of the toolbar.

1. Open the toolbar for the start screen.
2. Select the first button.
3. In the Inspector window, select "Properties > Events > Click".
4. Select the "ActivateScreen" system function from the function list.
5. Enter the name of the screen to which the change is to take place under "Screen name."
6. Enter "2" under "Object number" as the screen object in the screen to which the change is to take place. The focus is set to the screen object with the tab sequence number "2."

   Note

   The object number is defined during the configuration of the tab sequence in the screen. The sequence can be displayed in the screen by selecting the menu command "Edit > Tab sequence > Edit tab sequence".

7. Repeat the procedure for all the buttons of this toolbar:
   Always select the screen to be activated with the corresponding button as "Screen name".
8. Repeat the procedure for all the toolbars of the other screens:

Result

The buttons of all the toolbars are configured with the desired system functions for changing the screen.

Further steps

The following steps are still necessary to use the toolbars for the screen change:

- Creating configuration files with toolbars
- Configuring configuration files in the screen window

See also

Example: Configuring toolbars and buttons (Page 320)

Example: Creating a configuration file (RT Professional)

Introduction

In a configuration you specify which menu and toolbars are created with this file in screens or screen windows.

Requirement

- The toolbars for all the screens have been created.
- Buttons are created for every toolbar.
- The "Menus and Toolbars" editor is open.
- The "Configurations" tab is opened.

Defining a configuration

1. Click "Add" in the "Configurations" table.
   A new configuration is created.
2. Enter the name "Configuration_Start_Screen" in the "Name" column.
3. Select the toolbar "Toolbar_start_screen" in the "Toolbars" table.
4. Select the setting "Down" under "Alignment".
   The toolbar is displayed in the screen window at the lower margin of the display section.
5. To protect the toolbar with an operator authorization in Runtime, select an authorization.
6. Repeat the procedure for each screen you wish to create:

Result

The configuration files for displaying the toolbars of all screen windows have been created.

Further steps

The following steps are still necessary to use the toolbars for the screen change:

- Configuring configuration files in the screen window

See also

Example: Configuring toolbars and buttons (Page 320)

Example: Configuring a configuration file in the screen window (RT Professional)

Introduction

To change from any screen to additional screens, create as many screen windows as you require to represent your complete process in an overview screen. Both the corresponding screen and the associated configuration are configured in each screen window.

Requirement

- Screens have been created.
- A configuration file is created for each screen.

Configuring a configuration file in a screen window

1. Double-click "Screens > Add Screen" in the project window. A new screen is created.
2. Assign the name "Overview_Screen_Project" to this screen.
3. Drag-and-drop the "Screen window" object into the screen.
4. Enter a name for the title bar under "Properties > Properties > General >Title" in the Inspector window.
5. Select a screen that is displayed in the screen window in the Inspector window under "Properties > Properties > General > Name".
6. Define the configuration under "Properties > Properties > General > Configurations" in the Inspector window.
7. Repeat the procedure for all the screens and configurations.

Result

An overview screen with screen windows is created. A screen and a configuration are defined for each screen window.

See also

Example: Configuring toolbars and buttons (Page 320)

1.9 Display and operating elements

1.9.1 Device dependency of the objects

1.9.1.1 Objects for Basic Panels (Basic Panels)

Availability of display and operating objects for Basic Panels

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and operating objects for the Basic Panels.

On devices with the device version earlier than V13, configure simple display objects. On devices with the device version V13 or later, configure table-based display objects.

Overview

	KP300 Basic KP400 Basic	KTP400 Basic KTP600 Basic KTP1000 Basic TP1500 Basic	KTP700 Basic PN KTP900 Basic	KTP700 Basic DP KTP1200 Basic DP	KTP1200 Basic PN
Bar	Yes	Yes	Yes	Yes	Yes
User view	Yes	Yes	Yes [1]	Yes [1]	Yes [1]
Date/time field	Yes	Yes	Yes	Yes	Yes
I/O field	Yes	Yes	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes	Yes	Yes
Graphic view	Yes	Yes	Yes	Yes	Yes
Graphic I/O field	Yes	Yes	Yes	Yes	Yes
Help indicator	Yes	No	No	No	No
HTML browser	No	No	Yes	No	Yes
Circle	Yes	Yes	Yes	Yes	Yes
Trend view	Yes	Yes	Yes [1]	Yes [1]	Yes [1]
Line	Yes	Yes	Yes	Yes	Yes
Alarm view Alarm window	Yes	Yes	Yes [1]	Yes [1]	Yes [1]
Alarm indicator	Yes	Yes	Yes	Yes	Yes
Rectangle	Yes	Yes	Yes	Yes	Yes
Recipe view	Yes	Yes	Yes [1]	Yes [1]	Yes [1]
Button	Yes	Yes	Yes	Yes	Yes
Switch	Yes	Yes	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes	Yes	Yes

	KP300 Basic KP400 Basic	KTP400 Basic KTP600 Basic KTP1000 Basic TP1500 Basic	KTP700 Basic PN KTP900 Basic	KTP700 Basic DP KTP1200 Basic DP	KTP1200 Basic PN
System diagnostic view	Yes	Yes	Yes [1]	Yes [1]	Yes [1]
Text field	Yes	Yes	Yes	Yes	Yes

1) The objects are displayed on the panels as table-based objects.

See also

Objects for Comfort Panels (Page 327)
Objects for Mobile Panels (Page 329)
Objects for WinCC Runtime Advanced (Page 331)
Objects for WinCC Runtime Professional (Page 333)

1.9.1.2 Objects for Comfort Panels (Panels, Comfort Panels)

Availability of display and operating elements for Comfort Panels

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and control objects on Comfort Panels.

Overview

	KP400 Comfort KTP400 Comfort	KP700 Comfort TP700 Comfort	KP900 Comfort TP900 Comfort	KP1200 Comfort TP1200 Comfort	KP1500 Comfort TP1500 Comfort	TP1900 Comfort TP2200 Comfort
Bar	Yes	Yes	Yes	Yes	Yes	Yes
User view	Yes	Yes	Yes	Yes	Yes	Yes
Watch table	Yes	Yes	Yes	Yes	Yes	Yes
Date/time field	Yes	Yes	Yes	Yes	Yes	Yes
I/O field	Yes	Yes	Yes	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes	Yes	Yes	Yes
f(x) trend view	Yes	Yes	Yes	Yes	Yes	Yes
Function key	only on keyboard units					
Graphic view	Yes	Yes	Yes	Yes	Yes	Yes
Graphic I/O field	Yes	Yes	Yes	Yes	Yes	Yes
GRAPH overview	Yes	Yes	Yes	Yes	Yes	Yes
Handwheel	No	No	No	No	No	No
Help indicator	No	No	No	No	No	No

	KP400 Comfort KTP400 Comfort	KP700 Comfort TP700 Comfort	KP900 Comfort TP900 Comfort	KP1200 Comfort TP1200 Comfort	KP1500 Comfort TP1500 Comfort	TP1900 Comfort TP2200 Comfort
HTML browser	No	Yes	Yes	Yes	Yes	Yes
Camera view	Yes	Yes	Yes	Yes	Yes	Yes
Circle	Yes	Yes	Yes	Yes	Yes	Yes
Criteria analysis view	Yes	Yes	Yes	Yes	Yes	Yes
Trend view	Yes	Yes	Yes	Yes	Yes	Yes
Charge condition	No	No	No	No	No	No
Illuminated pushbutton	No	No	No	No	No	No
Line	Yes	Yes	Yes	Yes	Yes	Yes
Media Player	Yes	Yes	Yes	Yes	Yes	Yes
Alarm view Alarm window	Yes	Yes	Yes	Yes	Yes	Yes
Alarm indicator	Yes	Yes	Yes	Yes	Yes	Yes
PDF view	Yes	Yes	Yes	Yes	Yes	Yes
PLC code view	No	Yes	Yes	Yes	Yes	Yes
Polygon	Yes	Yes	Yes	Yes	Yes	Yes
Polyline	Yes	Yes	Yes	Yes	Yes	Yes
ProDiag overview	Yes	Yes	Yes	Yes	Yes	Yes
Rectangle	Yes	Yes	Yes	Yes	Yes	Yes
Recipe view	Yes	Yes	Yes	Yes	Yes	Yes
Button	Yes	Yes	Yes	Yes	Yes	Yes
Switch	Yes	Yes	Yes	Yes	Yes	Yes
Slider	Yes	Yes	Yes	Yes	Yes	Yes
Key switch	No	No	No	No	No	No
Sm@rtClient view	Yes	Yes	Yes	Yes	Yes	Yes
Symbol library	Yes	Yes	Yes	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes	Yes	Yes	Yes
System diagnostic view	Yes	Yes	Yes	Yes	Yes	Yes
System diagnostics window	Yes	Yes	Yes	Yes	Yes	Yes
Text field	Yes	Yes	Yes	Yes	Yes	Yes
Clock	Yes	Yes	Yes	Yes	Yes	Yes
Effective range signal	No	No	No	No	No	No
Effective range name	No	No	No	No	No	No

	KP400 Comfort KTP400 Comfort	KP700 Comfort TP700 Comfort	KP900 Comfort TP900 Comfort	KP1200 Comfort TP1200 Comfort	KP1500 Comfort TP1500 Comfort	TP1900 Comfort TP2200 Comfort
Effective range name (RFID)	No	No	No	No	No	No
WLAN reception	No	No	No	No	No	No
Gauge	Yes	Yes	Yes	Yes	Yes	Yes
Zone name	No	No	No	No	No	No
Zone signal	No	No	No	No	No	No

See also

Objects for Basic Panels (Page 326)

1.9.1.3 Objects for Mobile Panels (Panels, Comfort Panels)

Availability of display and operating elements for Mobile Panels

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and control objects for the Mobile Panels.

Overview

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP700 Mobile KTP700F Mobile KTP900 Mobile KTP900F Mobile	KTP400F Mobile
Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
User view	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Watch table	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time field	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
I/O field	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
f(x) trend view	No	No	No	No	No	No	Yes	Yes
Function key	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Graphic view	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP700 Mobile KTP700F Mobile KTP900 Mobile KTP900F Mobile	KTP400F Mobile
Graphic I/O field	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GRAPH overview	No	No	No	No	No	No	Yes	Yes
Handwheel	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	No	No
Help indicator	No	No	No	No	No	No	No	No
HTML browser	No	No	No	No	No	No	Yes	Yes
Camera view	No	No	No	No	No	No	Yes	Yes
Circle	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Criteria analysis view	No	No	No	No	No	No	Yes	Yes
Trend view	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Charge condition	No	No	No	Yes	Yes	Yes	No	No
Illuminated push-button	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]
Line	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Media Player	No	No	No	No	No	No	No	No
Alarm view Alarm window	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Alarm indicator	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PDF view	No	No	No	No	No	No	Yes	Yes
PLC code view	No	No	No	No	No	No	Yes	No
Polygon	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Polyline	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ProDiag overview	No	No	No	No	No	No	Yes	Yes
Rectangle	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Recipe view	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Button	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Switch	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Slider	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key switch	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]	Yes [1]
Sm@rtClient view	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Symbol library	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
System diagnostic view	No	No	No	No	No	No	Yes	Yes
System diagnostics window	No	No	No	No	No	No	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP700 Mobile KTP700F Mobile KTP900 Mobile KTP900F Mobile	KTP400F Mobile
Text field	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Clock	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Effective range signal	No	No	No	No	Yes	No	No	No
Effective range name	No	No	No	No	Yes	No	No	No
Effective range name (RFID)	No	No	No	No	No	Yes	No	No
WLAN reception	No	No	No	Yes	Yes	Yes	No	No
Gauge	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Zone name	No	No	No	Yes	Yes	No	No	No
Zone signal	No	No	No	Yes	Yes	No	No	No
1) optional operator control								

See also

Objects for Basic Panels (Page 326)

1.9.1.4 Objects for WinCC Runtime Advanced (RT Advanced)

Availability of display and operating elements for WinCC Runtime Advanced

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and control objects for WinCC Runtime Advanced.

Overview

	WinCC Runtime Advanced
Bar	Yes
User view	Yes
Watch table	Yes
Date/time field	Yes
I/O field	Yes

	WinCC Runtime Advanced
Ellipse	Yes
f(x) trend view	Yes
Function key	No
Graphic view	Yes
Graphic I/O field	Yes
GRAPH overview	Yes
Handwheel	No
Help indicator	No
HTML browser	Yes
Camera view	Yes
Circle	Yes
Criteria analysis view	Yes
Trend view	Yes
Charge condition	No
Illuminated pushbutton	No
Line	Yes
Media Player	No
Alarm view Alarm window	Yes
Alarm indicator	Yes
PDF view	Yes
PLC code view	Yes
Polygon	Yes
Polyline	Yes
ProDiag overview	Yes
Rectangle	Yes
Recipe view	Yes
Switch	Yes
Button	Yes
Slider	Yes
Key switch	No
Sm@rtClient view	Yes
Symbol library	Yes
Symbolic I/O field	Yes
System diagnostic view	Yes
System diagnostics window	Yes
Text field	Yes
Clock	Yes
Effective range signal	No
Effective range name	No
Effective range name (RFID)	No

	WinCC Runtime Advanced
WLAN reception	No
Gauge	Yes
Zone name	No
Zone signal	No

See also

Objects for Basic Panels (Page 326)
Camera view (Page 395)
PDF view (Page 440)

1.9.1.5 Objects for WinCC Runtime Professional (RT Professional)

Availability of display and operating objects

Only the objects which can be used for the device you are configuring will be shown in the object window. The following tables show the availability of indicator and operating objects for WinCC Runtime Professional.

Overview

Table 1-2 WinCC Runtime Professional

	WinCC Runtime Professional
Bar	Yes
User view	Yes
Watch table	No
Screen window	Yes
Scroll bar	Yes
Date/time field	No
Double T-piece	Yes
Print job/Script diagnostics	Yes
I/O field	Yes
Editable text field	Yes
Ellipse	Yes
Ellipse arc	Yes
Ellipse segment	Yes
Function key	No
f(x) trend view	Yes
f(t) trend view	Yes
Graphic view	Yes

	WinCC Runtime Professional
Graphic I/O field	Yes
GRAPH overview	Yes
HTML browser	Yes
Camera view	No
Channel diagnostics display	Yes
Combo box	Yes
Check box	Yes
Circle	Yes
Circular arc	Yes
Circle segment	Yes
Criteria analysis view	Yes
Trend view	No
Charge condition	No
Illuminated pushbutton	No
Line	Yes
List box	Yes
Media Player	Yes
Alarm view	Yes
Alarm indicator	No
Option button	Yes
PLC code view	Yes
Polygon	Yes
Polyline	Yes
ProDiag overview	Yes
Rectangle	Yes
Recipe view	Yes
Round button	Yes
Pipe	Yes
Pipe bends	Yes
Switch	No
Button	Yes
Slider	Yes
Sm@rtClient view	No
Disk space view	Yes
Symbol library	Yes
Symbolic I/O field	Yes
System diagnostic view	Yes
System diagnostics window	No
Table view	Yes
Text field	Yes
T piece	Yes

	WinCC Runtime Professional
Clock	Yes
Connector	Yes
Value table	Yes
Effective range name	No
Effective range name (RFID)	No
Effective range signal	No
WLAN reception	No
Gauge	Yes
Zone name	No
Zone signal	No

See also

Objects for Basic Panels (Page 326)

1.9.2 Objects

1.9.2.1 Bar (RT Advanced, Basic Panels, Panels, Comfort Panels)

Use

The tags are displayed graphically using the "Bar" object. The bar graph can be labeled with a scale of values.

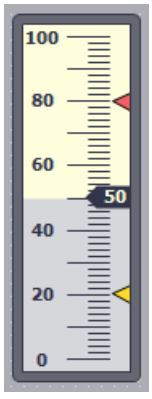

Layout

In the Inspector window, you customize the settings for the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Color transition: Specifies the change in color display when limit values are exceeded.
- Displaying the limit lines / limit markers: Shows the configured limit as a line or marker.
- Define bar segments: Defines the gradations on the bar scale.
- Define scale gradation: Defines the subdivisions, scale markings and intervals of a bar scale.

Color transition

You define how the color change is represented in "Properties > Properties > Appearance" in the Inspector window.

Color transition	Description
"Segmented"	If a particular limit was reached, the bar changes color segment by segment. With segment by segment representation, you visualize, for example, which limits are exceeded by the displayed value.
"Entire bar"	If a particular limit was reached, the entire bar changes color.

Displaying limit lines and limit markers

You display the configured limit in the bar as a line or marking in Runtime using the "Lines" and "Markings" property:

1. In the Inspector window, select "Properties > Properties > Appearance":
2. Activate "Lines" and "Markings".

Define bar segments

Use the "Subdivisions" property to define the number of segments into which the bar is divided by the main gradations on the scale.

Use the "Interval" property to divide the distance between the main gradations. The value appears as the difference in value between two adjacent main gradations:

1. In the Inspector window, select "Properties > Properties > Scales":
2. Activate "Show scale."
3. Select the corresponding value for "Settings > Subdivisions".
4. Select the corresponding value for "Settings > Marks label".
5. Select the corresponding value for "Large interval > Interval".

Configuring limits/ranges

You can represent limits and ranges in different colors. Define the colors in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Limits/Ranges".
2. Activate the limits/ranges that are to be displayed in Runtime
3. If necessary, change the default colors for the limits/ranges.

Note

If you select the option "Show ranges of tags", you can display up to five ranges in a slider with values that can be specified via a process tag. You define the values for the ranges with a process tag, which you interconnect with the screen object.

The "Show ranges of tags" option is available for Comfort Panels, KTP Mobile Panels and RT Advanced.

Note

If the "Show ranges of tags" option is enabled, the setting "Color gradient Segmented" under "Properties > Appearance > Bar" has no effect in Runtime.

See also

Device dependency of the objects (Page 326)

1.9.2.2 Bar (RT Professional)

Application

The tags are displayed graphically using the "Bar" object. The bar graph can be labeled with a scale of values.

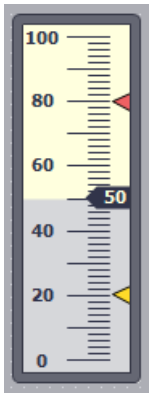

Layout

In the Inspector window, you customize the settings for the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Color transition: Specifies the change in color display when limit values are exceeded.
- Displaying the limit markers: Displays the configured limit value as an arrow.
- Define bar segments: Defines the gradations on the bar scale.
- Define scale gradation: Defines the position of the zero point on a bar scale.

Color transition

You define how the color change is represented in "Properties > Properties > Appearance" in the Inspector window.

Color transition	Description
"Segmented"	If a particular limit was reached, the bar changes color segment by segment. With segment by segment representation, you visualize, for example, which limits are exceeded by the displayed value.
"Entire bar"	If a particular limit was reached, the entire bar changes color.

Displaying the limit markers

You display the configured limit in the bar as an arrow in Runtime using the "Selections" property:

1. In the Inspector window, select "Properties > Properties > Appearance":
2. Activate "Selections".

Define bar segments

Use the "Scales" property to define the subdivisions into which the bar is divided by the main gradations on the scale.

Use the "Interval" property to divide the distance between the main gradations. The value appears as the difference in value between two adjacent main gradations:

1. In the Inspector window, select "Properties > Properties > Scales":
2. Activate "Show scale."
3. Enter the corresponding value for "Large interval > Interval".

Note

Intervals can only be changed if the "Auto-scale" option is disabled.

Define scale gradation

Use the "Zero point" property to define the zero point on the bar graph scale. Enter the value as a percentage. The percentage relates to the distance between the end values on the scale. For a value of 0 %, for example, the zero point is represented at the height of the lowest value on the scale. The zero point can also be outside the shown area.

The "Zero point" property is only evaluated if the "Bar scaling" property is set to "Auto-scale":

1. In the Inspector window, open "Properties > Properties > Scales"
2. Activate "Auto-size".
3. Enter the required value for "Start values .> Zero point".

See also

Device dependency of the objects (Page 326)

1.9.2.3 User view (as of V13)

Use

The "User view" object is used to set up and administer users and authorizations. The user view is used, for example, to create new users in Runtime and assign them to a user group.

On HMI devices with device version V13 or later, the table-based user view is available for managing users and authorizations.

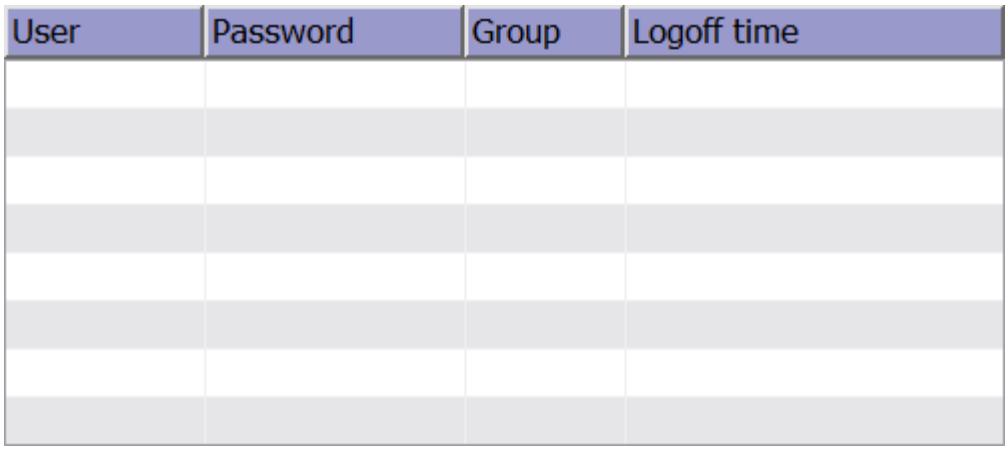

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Number of lines: Specifies the maximum number of visible entries.
- Columns moveable: Specifies whether the operator can change the sequence of columns in Runtime.

In addition, you set the borders, the fill pattern and the colors of the table header in the Inspector window.

Number of lines

The number of lines in the user view displayed in Runtime is specified in the Inspector window. The setting for the number of lines is only effective if "Fit object to contents" is active.

1. Click "Properties > Properties > View" in the Inspector window.
2. Enter an integer value under "Number of lines".
3. Click "Properties > Properties > Layout" in the Inspector window.
4. Select "Fit object to contents".

Columns moveable

The operator can change the sequence of columns in Runtime with the "Columns moveable" property.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Activate "Columns moveable".

Column width

In the Inspector window, you can change the width of columns displayed in Runtime.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Enter integer values for the column width under "Columns".

Note

The "Password" column is a dynamic column which adapts to the outer dimensions of the object so that the object content occupies the entire width. The column width you assign is applied as the minimum size. In a dynamic column, the minimum size is changed; however, the width is displayed dynamically to fill the display.

1.9.2.4 User view, simple (RT Advanced, Basic Panels, Panels, Comfort Panels)

Application

The "User view" object is used to set up and administer users and authorizations.

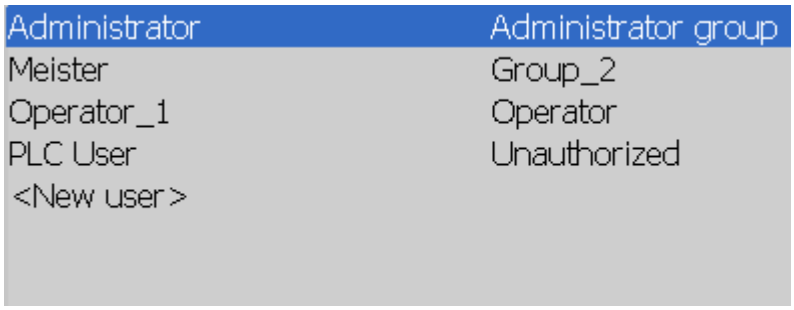

Note

Do not use the user view in a group.

Note

The "Simple user view" object cannot be operated dynamically with a script.

Note

Configure the "Simple user view" object on devices with device version 12.0.0.0 or older.

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Number of lines: Specifies the maximum number of visible entries.

Number of lines

The number of lines in the user view displayed in Runtime is specified in the Inspector window. The setting for the number of lines is only effective if the property "Fit object to contents" is active.

1. In the Inspector window, select "Properties > Properties > View".
2. Enter an integer value under "Number of lines".
3. In the Inspector window, activate "Properties > Properties > Layout".
4. Select "Fit object to contents".

Display in Runtime

The appearance depends on the authorizations:

- All users on the HMI device are displayed in the User view to the administrator or to a user with administrator authorizations.
- For a user without administrator rights, only the user's own entry is shown.

Operation

Depending on the configuration you can:

- Manage users, e.g. create, delete.
- Change existing user data.
- Export or import user data.

Note

The number is limited to 100 users and one PLC user on an HMI device. This restriction does not apply to PCs. On a PC, the maximum number of users is restricted by the physical memory.

1.9.2.5 User view, advanced

Application

The "User view" object is used to set up and administer users and authorizations. The user view is used, for example, to create new users in Runtime and assign them to a user group.

Advanced user view

Depending on the HMI devices, the extended or simple user view is available for administration of users and authorizations.

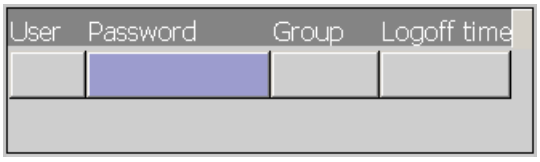

Changing the view type

On HMI devices with a display size larger than 6 ", you can configure both the advanced user view and a simplified user view. The view type is only available for configuration on devices up to device version V13.

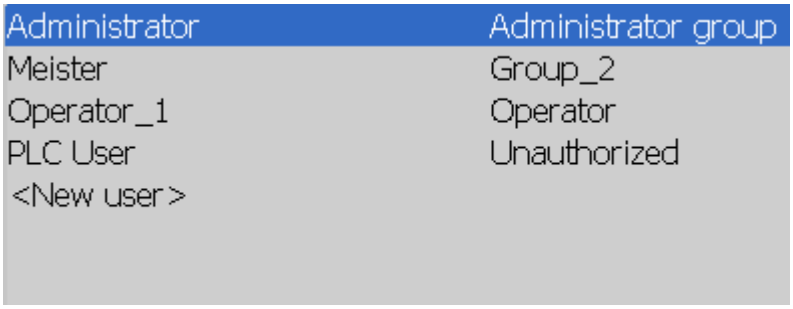

Proceed as follows to configure the appropriate user view:

1. Click "Properties > Properties > Layout > Mode" in the Inspector window.
2. Select "Simplified" or "Advanced".

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Number of lines: Specifies the maximum number of visible entries.
- Columns moveable: Specifies whether the operator can change the sequence of columns in Runtime.

Number of lines

The number of lines in the user view displayed in Runtime is specified in the Inspector window. The setting for the number of lines is only effective if "Fit object to contents" is active.

1. Click "Properties > Properties > View" in the Inspector window.
2. Enter an integer value under "Number of lines".
3. Click "Properties > Properties > Layout" in the Inspector window.
4. Select "Fit object to contents".

Columns moveable

The operator can change the sequence of columns in Runtime with the "Columns movable" property.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Activate "Columns moveable".

This option is only available at the complex user view.

See also

User view (Page 1342)

Creating users (Page 1344)

Managing users in the complex user view (Page 1346)

Device dependency of the objects (Page 326)

1.9.2.6 Watch table (RT Advanced, Panels, Comfort Panels)

Use

The "Watch table" object is used to configure an editor that will allow you to process single address areas of a SIMATIC S7 in runtime.

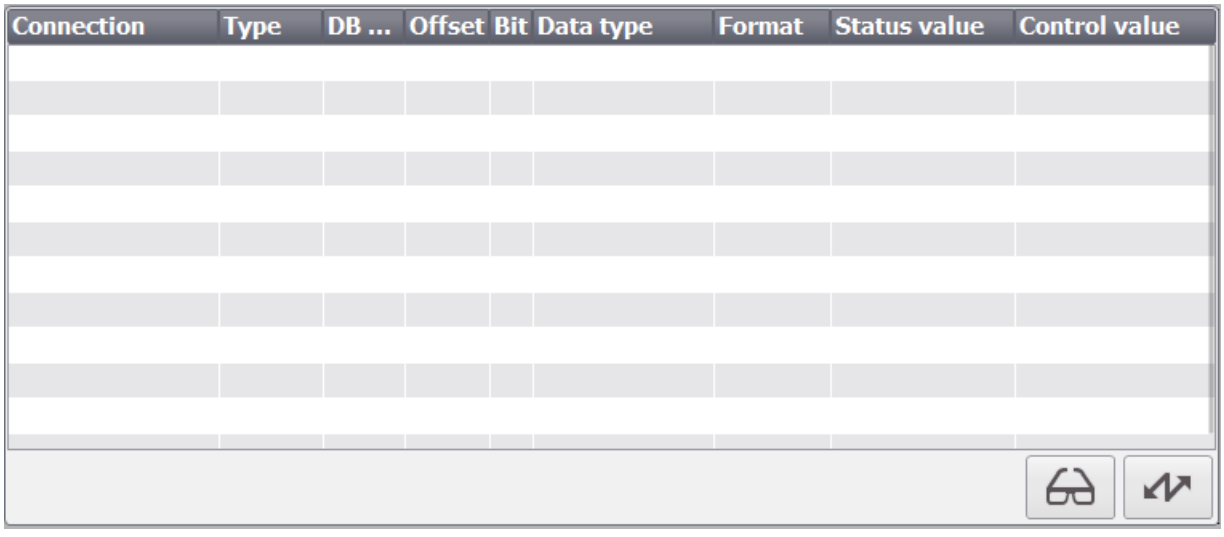

Note

The "Watch table" object is enabled for the following PLCs:

- SIMATIC S7-300
- SIMATIC S7-400

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- PLC type: Specifies the PLC type in the object for which the address area is output in runtime.
- Operator controls: Specifies the controls of the object.

- Visible columns: Specifies the displayed columns in Runtime.
- Columns moveable: Specifies whether the operator can change the sequence of columns in Runtime.

Operator controls

Set the control elements that you can use to control the "Watch table" object in runtime under "Properties > Properties > Display" in the "Settings" area of the inspector window.

	Function
	This button refreshes the display in the status value column.
	Use this button to accept the new value in the control value column. The control value is then written to the PLC.

Visible columns

The columns that are displayed in runtime are specified in the Inspector window in the "General" group in the "Visible columns" area.

Column	Description
"Connection"	The PLC of which the address areas are displayed.
"Type" "DB number" "Offset" "Bit"	The address area of the operand.
"Data type" "Format"	The data type of the operand.
"Status value"	The value that was read from the specified address of the operand.
"Control value"	The value written to the specified address of the operand.

Sequence of columns

The operator can use the "Column ordering" property to change the sequence of columns in runtime.

1. Select "Properties > Properties > Columns" in the Inspector window.
2. Activate "Properties of columns > Sequence of columns".

System functions

Two system functions are available for updating the values in the watch table and transferring the values to the PLC:

- StatusSteuernLeseWerte
- StatusSteuernSchreibeWerte

See also

Device dependency of the objects (Page 326)

1.9.2.7 Screen window (RT Professional)

Application

You use the "Screen window" object to represent other screens from the project in the current screen. You can make the object dynamic to constantly update the content of a screen window, for example. Customized menus and toolbars are used to add specific buttons to the screen window.

You can also use independent screen windows independently of the screen in question. With appropriate hardware equipment and support by the operating system you can also control multiple monitors and map processes in a more comprehensive and differentiated manner.

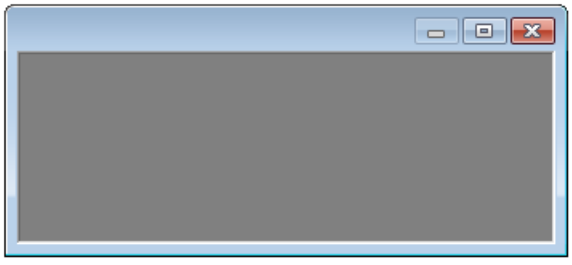

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Zoom factor: Defines the size of the embedded screen.
- Screen section: Defines the section of the embedded screen that is displayed in the screen window. If the embedded screen is larger than the screen window, you configure scroll bars for the screen window.
- Independent screen windows: Specified that the screen windows are displayed independently from the screen in which they were configured.

Note

Cascading screen windows

Screen windows can also display screens which, in turn, contain screen windows. The ability to display cascaded screen windows in runtime depends on the operating system. In the 32-bit version, 22 cascaded screen windows can be displayed. In the 64-bit version, 14 cascaded screen windows can be displayed.

Matching the size of the embedded screen and screen window

You can match the size of the embedded screen to the size of the screen window in the following ways:

- You want the embedded screen to appear smaller.
  Enter the required zoom factor under "Properties > Properties > General" in the Inspector window.
- You want to scroll to a section of the embedded screen.
  In the Inspector window, activate "Properties > Properties > Layout > Scroll bar".
  The user can scroll to details of the embedded screen in Runtime.
  You can move the position of the scroll bar horizontally or vertically under "Properties > Properties > Layout > Scroll bar position" in the Inspector window.
- You can adapt the embedded screen to the size of the screen window, or vice versa.
  Select either "Fit window to screen", or "Fit screen to window" in "Properties > Properties >" in the Inspector window.
  If you select the "Fit window to screen" setting, the size of the screen window is adapted to the embedded screen in Runtime and can no longer be changed in the engineering system.
  Choose between two options in scaling mode for the "Fit screen to window" setting:
  - Uniform: The full embedded screen is displayed in the screen window. If the aspect ratios differ, the screen does not fill the screen window.
  - Fill: The screen fills the screen window. If the aspect ratios differ, parts of the screen are trimmed.

  The aspect ratio is retained with both options.

Define screen section

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Enter the values for the horizontal and vertical displacement of the zero point of the screen display under "Offset."
   The desired section is displayed in the screen window.
3. If you want to scroll from the defined section to other areas of the screen, activate "Properties > Properties > Appearance > Scrollbar".

Defining independent screen windows

1. Insert a screen window in the start screen.
2. Select the screen that is displayed in the screen window under "Properties > Properties > General > Name".
3. Select "Properties > Properties > General > Independent screen windows".
4. Select "Position and mode":
   - As configured: The screen window is displayed in the configured size and at the configured position.
   - Centered: The screen window is displayed in the configured size in a central position.
   - Maximized: The screen window is adjusted to the monitor size.

5. If you are using multiple monitors select the desired monitor for the screen window under "Monitor number".

   Note

   The monitor numbers refer to the number of monitors set in Microsoft Windows: "Control Panel > Display > Settings > Display"

6. Repeat steps 1 to 5 for all further independent screen windows.

Define tag prefix

Define a tag prefix to access structure instances in Runtime. The tag prefix is freely definable but must match the name of the structure instance.

Note

The dynamic interconnection of tags in runtime

To interconnect tags dynamically in runtime, use the tag prefix or indirectly addressed tags.

Avoid using both methods at the same time.

1. Click "Properties > Properties > General > Contents" in the Inspector window.
2. Enter one of the created structure instances with a period after the name under "Tag prefix", e.g. "Motor1.".

The "Temperature" tag is requested on an object in the screen window. If a tag prefix "Motor1." is assigned to the screen window, the "Motor1.Temperature" tag is requested.

If the tag prefix for a screen window is set, the tag prefix is prefixed to all tags contained in the screen to be displayed. This also applies when the request is made in a function. If a tag or a tag prefix is to be read, you must prefix "@NOTP::" to the tag name.

Note

Availability of tag prefixes in indicator and control objects

The tag prefix property is only available for "slider" and "pointer instrument" objects.

The objects of the "Controls" palette do not support the tag prefix.

Note

If the screen is assigned to a user data type, the "Tag prefix" cannot be edited.

Note

Do not specify any additional tag prefix in the lower-level screen window

If a screen window is configured in a referenced screen of a screen window, the tag prefix of the higher-level screen window is used in the lower-level screen window. Do not specify any additional tag prefix in the lower-level screen window, because bundled structure tags are not supported in WinCC. The interpretation "TagPrefix1.TagPrefix2.TagName" cannot be used for dynamizations.

Defining a structured process tag

As an alternative to the tag prefix, you can assign the "screen window" object to a structured process tag. In order for a structured process tag to be selected, the user data type must be assigned to the screen in the screen window. This data type must be saved in the project library.

1. Once the screen of the HMI user data type is assigned, you can select a process tag of the HMI user data type from the "Process tag" list.
   Or:
   Once the screen of the PLC user data type is assigned, you can select a process tag that is connected to a PLC tag or a PLC data block.

The selected process tag appears under "Tag prefix" and "Process tag".

Note

The "Process tag" property cannot be made dynamic. Use the "Tag prefix" property to make it dynamic.

The "Process tag" property cannot be addressed by using VB or C scripting. Use the "Tag prefix" property for VB or C scripting.

Note

Cascading screen windows

For cascading screen windows, the screen in which you configure the screen window must be assigned a user type. In this case, both a process tag as well as a structural element of the assigned user data type can be selected under "Process tag". In order for the user data type to be selected, it must be available in the project library.

Special feature of visibility animation

If you want to animate the visibility of a screen window and the screen window is not to be visible at first, disable the "Visibility" setting under "Properties > Miscellaneous".

See also

Device dependency of the objects (Page 326)

1.9.2.8 Scroll bar (RT Professional)

Application

The "Scroll bar" object is used to control processes. The scroll bar allows you to make stepless changes to values, for example. To incorporate a scroll bar into the process, dynamize the corresponding attribute.

Layout

In the Inspector window, you customize the settings for the object position, style and color. You can, in particular, adapt the following properties as required:

- Maximum value and minimum value: Specifies the top and bottom values of the scale.
- Alignment: Defines the horizontal, or vertical alignment.

Maximum value and minimum value

You define the top and bottom values of the scale in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General":
2. Enter a number under "Maximum static" and "Minimum static". If you select a tag as the end value of the scale, the number will be no longer available.

Define alignment

Define whether the scroll bar is aligned vertically or horizontally in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Select the required alignment under "Style."

See also

Configure operator input alarms (Page 827)

Device dependency of the objects (Page 326)

1.9.2.9 Date/time field (RT Advanced, Basic Panels, Panels, Comfort Panels)

Use

The "Date/time field" object shows the system time and the system date. The appearance of the "Date/time field" depends on the language set on the HMI device.

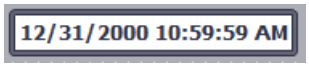

Layout

In the Inspector window, you customize the position, style, colors and font types of the object. You can adapt the following properties in particular:

- Display system time: Specifies that the system time is displayed.
- Include tag: Specifies that the time of the connected tag is displayed.
- Long date/time format: This setting defines the format displayed for the data and time.

Display system time

The time displayed in the "Date/time field" on the HMI device is specified in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Select "Format > System time".

Using tags

The time of the interconnected tag is displayed in the date/time field.

1. In the Inspector window, select "Properties > Properties > General".
2. In the "Format" area, select a tag with the "DateTime" data type, e.g. an internal tag.

Long date/time format

Visualization of the date and time is specified in the "Format" area under "General" in the Inspector window.

Option	Description
"Enabled"	Date and time are displayed in full, e.g. "Sunday, December 31, 2000 10:59:59 AM"
"Disabled"	Date and time are displayed in short form, e.g. "12/31/2000 10:59:59 AM"

Behavior during operation

If the operator ignores the syntax when entering values or enters invalid values in Runtime, these entries will not be applied. Instead, the original value (plus the time that has elapsed in the meantime) appears in the date/time field and a system event is displayed on the HMI device.

See also

Device dependency of the objects (Page 326)

1.9.2.10 Double T-piece (RT Professional)

Application

You use the "Double T-piece" to reproduce a pipe junction. To reproduce a pipe system, connect the "Double T-piece" to other objects, e.g. pipe bend or T piece.

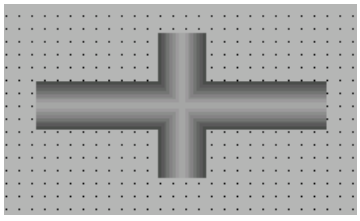

Layout

Customize the object position, color and width in the Inspector view.

See also

Pipe (Page 469)

Pipe elbow (Page 470)

T-piece (Page 502)

Device dependency of the objects (Page 326)

1.9.2.11 Print job/script diagnostics (RT Professional)

Application

You use the "Print job/Script diagnostics" object to show content from the "Print jobs" and "Scripts" editors. Information is transferred to the "Print job/Script diagnostics" object in runtime and to enable an operation. Depending on which windows content and which template you use, you can display a preview of the print job, for example.

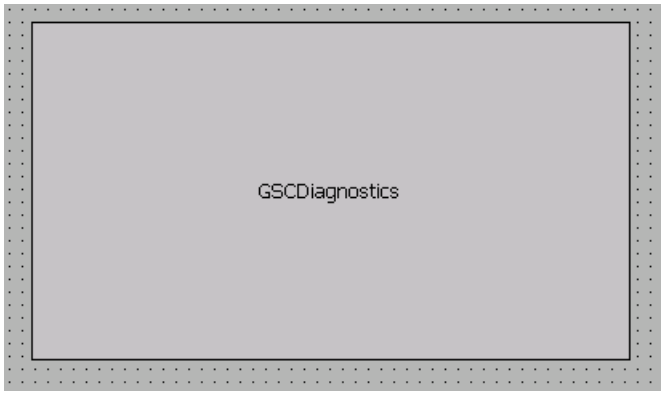

Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Window content: Defines the application that is displayed in the "Print job/Script diagnostics" object.
- Template: Defines the template in which the window content is displayed in the "Print job/Script diagnostics" object.

Templates for using functions

The following templates are available for functions:

- GSC diagnostics
  The "Print job/Script diagnostics" object is supplied by applications of the functions. It displays the results from the diagnostic system.

Templates for using print jobs

The following templates are available for the "Print jobs" window content:

- All print jobs
  The "Print job/Script diagnostics" object is supplied by the print jobs. The available print jobs are shown as a list.
- Shortcut menu for all jobs
  The "Print job/Script diagnostics" object is supplied by the print jobs. The available print jobs are shown as a list. A shortcut menu offers the following actions:
  - Print preview
  - Print report
- Jobs detailed view
  The "Print job/Script diagnostics" object is supplied by the print jobs. There is a menu for selecting the available print jobs. Detailed information is displayed for the selected print job.
- Shortcut menu for selected jobs.
  The "Print job/Script diagnostics" object is supplied by the print jobs. The available print jobs are shown as a list.
  To display print jobs in the list, select "Properties > Properties > General > Display print job/script diagnostics in Control" in the Print Job editor. A shortcut menu offers the following actions:
  - Print preview
  - Print report

Configure window content

You define the window content of the object in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Select the window content, e.g. "Print jobs".

Configure template

You define the template for the window content of the object in the Inspector window:

1. In the Inspector window, select "Properties > Properties > General".
2. Select a template

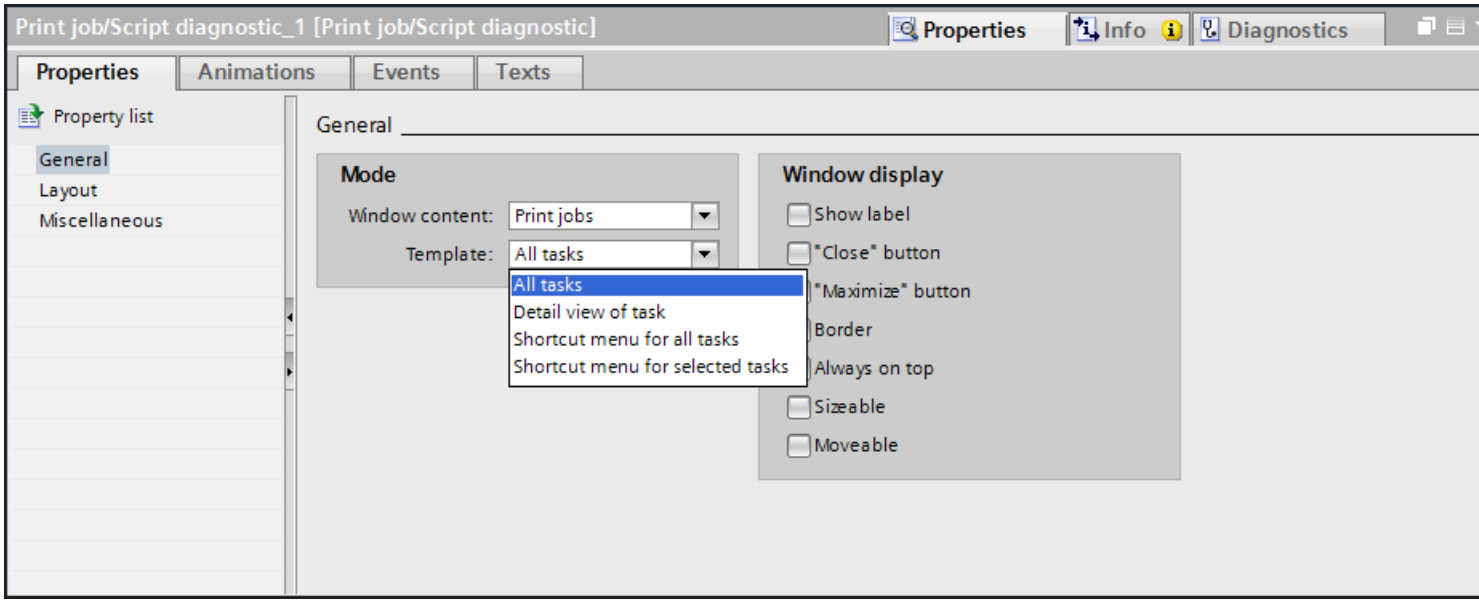

Window layout in Runtime

In runtime, the "Print Job/Script Diagnostics" object appears as a standalone window within a process picture. Use the attributes under "Window display" to customize the window layout features.

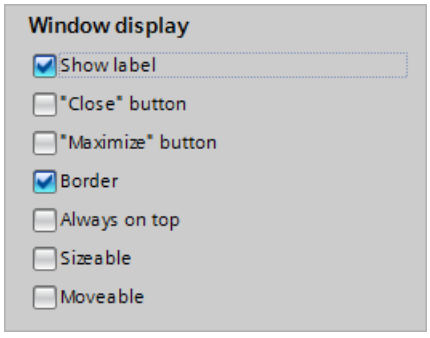

See also

Device dependency of the objects (Page 326)

1.9.2.12 I/O field

Application

The "I/O field" object is used to enter and display process values.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Display format: Specifies the display format in the I/O field for input and output of values.
- Hidden input: Specifies whether the input value is displayed normally or encrypted during input.

Note

Reports

In reports, I/O fields only output data. "Output" mode is preset. Properties for configuring input are not available, e.g. "hidden input".

Mode

The response of the I/O field is specified in the Inspector window in "Properties > Properties > General > Type".

Mode	Description
"Input"	Values can only be input into the I/O field in Runtime.
"Input/output"	Values can be input and output in the I/O field in Runtime.
"Output"	The I/O field is used for the output of values only.

Display format

The display format for the input and output of values is specified in "Properties > Properties > General > Format" in the Inspector window.

Display format	
"Binary"	Input and output of values in binary form
"Date"	Input and output of date information. The format depends on the language setting on the HMI device.
"Date/time"	Input and output of date and time information. The format depends on the language setting on the HMI device.
"Decimal"	Input and output of values in decimal form

Display format	
"Hexadecimal"	Input and output of values in hexadecimal form
"Time"	Input and output of times. The format depends on the language setting on the HMI device.
"String"	Input and output of character strings.

Note

Data formats

Not all data formats are available for selection for Runtime Professional.

Hidden input

In Runtime the input can be displayed normally or encrypted, for example for hidden input of a password. A "*" is displayed for every character during hidden input. The data format of the value entered cannot be recognized.

1. In the Inspector window, select "Properties > Properties > Characteristics":
2. Select "Hidden input".

Avoid overlaps in output fields

If several I/O fields are configured as output fields with a transparent background in a screen, these I/O fields may overlap. The transparent part of the one field covers the digits of the other field. This may cause display problems in Runtime. In order to avoid such overlaps, set the borders of the I/O fields to zero in the object properties under "Properties > Properties > Appearance". Select "Properties > Properties > Layout > Fit object to contents."

Limits

Under "Properties > Properties > Limits" in the Inspector window, set the colors for the values that violate the high or low limits.
You define the limits via the properties of a tag. In Runtime Professional you can specify if the colors should be used.

When there is a limit violation during Runtime, the background color of the I/O field changes according to your configuration, even if the I/O field is in "Input" mode.

In Runtime Professional, you can also define the limit range for the entry in the I/O field via "Properties > Properties > Limits".

If you enter a numeric value outside this limit, it is not applied; for example, 80 with a limit of 78. In this case, the HMI device generates a system event, if an alarm window is configured. The original value is displayed again.

Decimal places for numerical values

The configuration engineer can define the number of decimal places for a numerical input field. The number of decimal places is checked when you enter a value in this type of I/O field. Decimal places in excess of the limit are ignored. Empty decimal places are filled with "0".

Behavior when switching between input fields

When you change to another input field within the same screen, and the screen keyboard appears, the "Exit field" event is not executed for the previous field unless you close the screen keyboard.

See also

Configure operator input alarms (Page 827)

I/O field: Format pattern (Page 358)

Device dependency of the objects (Page 326)

1.9.2.13 I/O field: Behavior during operation (RT Professional)

Introduction

In WinCC Runtime Professional, you can configure the window display of the "I/O field" object via the properties in the Inspector window.

Behavior during operation

Apply value on completed entry

The "Apply value on completed entry" attribute specifies when an input value is to be applied. If the attribute has the value "No", then the input value is applied when the input is confirmed with <ENTER> . Otherwise the input value is automatically applied as soon as the preset number of characters has been entered.

Apply value on exit

By means of the "Apply value on exit" property, the application of a value can also be enabled for the case where the I/O field is exited without prior confirmation or reaching the required number of characters.

Clear on invalid input

The "Clear on invalid input" property can be used to prevent application of an incorrect input value when the field is exited. For example, an input value that does not correspond to the predefined data format of the input field is incorrect.

Clear on focus

The "Clear on focus" property specifies whether the field contents are to be cleared when the input field is selected.

Cursor control

Specifies a value at which the alpha cursor automatically jumps to the next field in the TAB sequence after an input is made.

Edit on enable

The "Edit on enable" attribute specifies for input fields whether or not an immediate change into input mode occurs when the object takes focus.

Hidden input

The "Hidden input" attribute specifies whether the input value is displayed during input as normal or encrypted. If this attribute has the value "Yes", every character input is replaced with the "*" character. The value entered and the data format of the value cannot be recognized.

1.9.2.14 I/O field: Format pattern

Introduction

Four different display formats are available for the input and output of values in an I/O field. Numerical values can be edited in binary, decimal or hexadecimal format. The "String" display format must be specified for the I/O field to display text.

Various format patterns can be selected or freely defined based on the display format.
The definition for a format can be rewritten as a sequence of formatting codes. The formatting codes act as placeholders for a specific group of characters. For example, if a formatting code for which only the display of the numbers 0-9 is preset for a specific position in the display of an I/O field, only letters can be input at this position.

In addition to the numerical values in an I/O field, you can also output character strings and date/time. To output a character string, use a tag of the data type "String" with a maximum field length of 320 characters. To display date/time formats, use tags with the "Date/time" data type. The display of date/time depends on the language setting on the HMI device in runtime.

Formatting codes - "Binary" data format

1 Placeholder for the binary values 0 and 1. The number of formatting code "1" specifies the number of digits allowed for the display of a binary value.

Table 1-3 Example: The eight-digit binary value 10011101 can be displayed as follows:

Format	Permissible number of digits	Display
1111111111	8	10011101

Additional formatting options depend on the data types of the tags used, e.g. in the data type Int, the formatting is "11111111 11111111".

Formatting codes - "Decimal" data format

9 Placeholder for the numbers 0 to 9. The number of formatting code "9" specifies the number of digits allowed for the display of a decimal value and can also be used for negative values with the prefix "-". If the actual number of decimal places exceeds the number specified in the display format, the displayed value is rounded.

, A period defines the location of the decimal point. The formatting code "." may be used at any point in the format, but it can only be used once. Decimals are only supported for floating-point data types. If you want to display a tag value of fixed-point data types, select the option "Decimal places" for the implicit conversion of the values.

s Positive decimal numbers are displayed with a sign. The formatting code "s" should be in the first place of the output format and may only be used once.

e The decimal number is displayed in exponent notation. The formatting code "e" should be in the last place of the output format and may only be used once.

Table 1-4 Example: The six-digit floating-point number 123.456 can be displayed as follows:

Format	Permissible number of digits	Display
999	3	123
999.9	4	123.5
s999.9	4 + sign	+123.5
999,999	6	123.456
9999.9999	8[1)]	0123.4560
s9999.9999	8 + sign[1)]	+0123.4560
9.99999e	6	1.23456e+002[2)]

[1)] To do this, activate the option "Leading zeros"

[2)] This display is only possible in an I/O field in output mode

I/O field with "decimal" display format and format pattern without "s" prefix

You have linked a process tag to an I/O field. The I/O field is assigned the "decimal" display format. You can select a format without a sign as a format pattern.

Formatting codes - "Hexadecimal" data format

The six-digit fixed-point number 123456 can be displayed as follows:

Format	Number of digits	Display
FFFFFFFF	8	0001E240

Additional formatting options depend on the data types of the tags used, e.g. in the data type Int, the formatting is "FFFF".

See also

I/O field (Page 355)

1.9.2.15 I/O field: Format (RT Professional)

Introduction

Four different display formats are available in WinCC Runtime Professional for the input and output of values in an I/O field. Numerical values can be edited in binary, decimal or hexadecimal format. The "String" display format must be specified for the I/O field to display text.

Various format patterns can be selected or freely defined based on the display format.
The definition for a format can be rewritten as a sequence of formatting codes. The formatting codes act as placeholders for a specific group of characters. For example, if a formatting code for which only the display of the numbers 0-9 is preset for a specific position in the display of an I/O field, only letters can be input at this position.

Note

If the value you want to display does not correspond exactly to the definition of the format, only three asterisks are displayed. This applies for the length of the entire character string and also for the type and position of the individual characters.

Formatting codes - "Binary" data format

1 Placeholder for the binary values 0 and 1. The number of formatting code "1" specifies the number of digits allowed for the display of a binary value.

0 If needed, the binary value is preceded by a leading zero. The format can also begin with the format code "0", but it can only include it once.

Table 1-5 Example: The eight-digit binary value 10011101 can be displayed as follows:

Format	Permissible number of digits	Display
11	2	01
011	2 + leading zero	001
1111	4	1101
01111	4 + leading zero	01101
1111111	7	0011101
01111111	7 + leading zero	00011101
1111111111	10	10011101
01111111111	10 + leading zero	010011101

Formatting codes - "Decimal" data format

9 Placeholder for the numbers 0 to 9. The number of formatting code "1" specifies the number of digits allowed for the display of a decimal value. If the actual number of decimal places exceeds the number specified in the display format, the displayed value is rounded.

, A period defines the location of the decimal point. The formatting code "." may be used at any point in the format, but it can only be used once.

s Positive decimal numbers are displayed with a sign. The formatting code "s" should be in the first place of the output format and may only be used once.

0 Leading and trailing zeros are displayed, when the actual number of integer or decimal places is less than the specified number in the display format. The formatting code "0" must be before the first "9" and may only be used once.

e The decimal number is displayed in exponent notation. The formatting code "e" should be in the last place of the output format and may only be used once.

Table 1-6 Example: The six-digit decimal value 123.456 can be displayed as follows:

Format	Permissible number of digits	Display
999	3	124
999.9	4	123.5
s999.9	4 + sign	+123.5
999,999	6	123.456
09999.9999	8 + zeros	0123.4560
s09999.9999	8 + sign + zeros	+0123.4560
1111111111	10	10011101
9.99999e	6	1.23456e+002

Note

When storing a floating-point number in IEEE format of S5, a format for display in an IO field should be used that provides for the sign and exponent (e.g. s0999.999e).

I/O field with "decimal" display format and format pattern without "s" prefix

You have linked a process tag to an I/O field. The I/O field is assigned the "decimal" display format. You may select a signed or an unsigned display format.

A "Format pattern" setting without "s", e.g., "999" has the following effects:

1. You cannot set negative values using the I/O field in Runtime.
2. If the tag assumes a negative value, the I/O field generates a two's complement and a corrupted positive value is output.

Formatting codes - "Hexadecimal" data format

f Placeholder for the letters A to F or a to f and numbers 0 to 9, which are used to represent hexadecimal numbers. The permitted number of characters is defined by the number of the formatting codes "f" in the output format.

0 Leading zeros of the hexadecimal value are displayed if the output format begins with the format code "0". The formatting code "0" may only be included once.

Table 1-7 Example: The eight-digit binary value ABCDEF01 can be displayed as follows:

Format	Permissible number of digits	Display
ff	2	01
0ff	2 + leading zero	001
ffff	4	EF01
0ffff	4 + leading zero	0EF01
fffffffff	10	ABCDEF01
0ffffffffff	7 + leading zero	0ABCDEF01

Formatting codes - "String" data format

The allowed length of a character string is defined by the number of formatting codes used (exception "*").

* Input of a character string of any length

? Input of any character string

a Lower-case letters, upper-case letters and digits are allowed
no separators or similar.

A Upper-case letters and digits are allowed
no lower-case letters, separators or similar.

b Lower-case letters and upper-case letters are allowed
no digits, separators or similar.

B Only upper-case letters are allowed
no lower-case letters, digits, separators or similar.

1-9 The formatting codes "1", "2", ..., "9" are used as placeholders for digits.
The selected formatting code also defines the actual digits allowed: For example, if the "2" is specified, only the digits 0, 1 or 2 can be displayed. The formatting code "8" allows all digits except for the 9.

h Only the digits 0 to 9 and the letters A to F or a to f are allowed.
The formatting code "h" allows only characters that are required to display hexadecimal numbers.

t The formatting code "t" forces input of a separator at the specified position. These separators are valid: Slash, colon, comma, period and space.

1.9.2.16 Editable text field (RT Professional)

Application

The "Editable text field" object enables you to display several lines of text in a rectangle on the screen. If the text is larger than the rectangle, WinCC automatically adds a scroll bar to the right margin.

If you allow operation, the operator can edit the text in runtime. You can use the multiple-line text for the input or output of text by connecting tags.

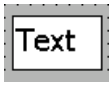

Layout

In the Inspector window, you customize the position, style, colors and font types of the object. You can adapt the following properties in particular:

- Text: Defines the text for the "Editable text field".

Text

The text for the "Editable text field" is defined in the inspector window.

1. Click "Properties > Properties > General" in the inspector window.
2. Enter a text of any length.
   For texts over several lines you can set a line break by pressing <Enter>, with the key combination <Shift + Enter> or the key combination <Ctrl + Enter>.

See also

Device dependency of the objects (Page 326)

1.9.2.17 Ellipse (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Application

The "Ellipse" is an enclosed object that can be filled with a color or pattern.

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Horizontal radius: Specifies the horizontal radius of the elliptical object.
- Vertical radius: Specifies the vertical radius of the elliptical object.

Horizontal radius

The horizontal radius of the "Ellipse" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 under "Horizontal."

Vertical radius

The vertical radius of the "Ellipse" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 at "Vertical."

See also

Elliptical arc (Page 364)

Ellipse segment (Page 366)

Rotating objects (Page 117)

Flipping objects (Page 118)

Device dependency of the objects (Page 326)

1.9.2.18 Elliptical arc (RT Professional)

Application

The "Elliptical arc" is an open object. Use the "Ellipse segment" object if you want to fill the object with color. By default, an elliptical arc is a quarter ellipse. It can be customized as required.

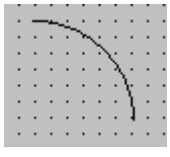

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Horizontal and vertical radii: Specifies the horizontal and vertical radius of the elliptical object.
- Start and end angle: Specify where the start and end angle lie on a virtual circle of 360°.

Defining radii

Define the horizontal and vertical radius of the "Elliptical arc" object in the Inspector window. Enter the values using Pixels as the unit.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value for "Horizontal radius" and "Vertical radius".

Define start and end angle

Determine the length of the elliptical arc using the "Start Angle" and "End Angle" attributes. Enter the values using Degrees as the unit.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value for "Angle start" and "Angle end".

Alternative procedure

You can also change the start and end points of the elliptical arc using the mouse.

1. Place the cursor on a start or end point of the elliptical arc. The start and end points are indicated by small blue squares.
   The mouse cursor changes to a cross.
2. Hold down the mouse button, and drag the start or end point to the desired position.

See also

Ellipse (Page 363)

Device dependency of the objects (Page 326)

1.9.2.19 Ellipse segment (RT Professional)

Application

The "Ellipse segment" is a closed object that you can fill with a color or pattern. By default, an ellipse segment is a quarter ellipse. It can be customized as required.

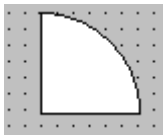

Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Horizontal and vertical radii: Specifies the horizontal and vertical radius of the elliptical object.
- Start and end angle: Specify where the start and end point lie on a virtual circle of 360°.

Defining radii

Define the horizontal and vertical radius of the "Ellipse segment" object in the Inspector window. Enter the values using Pixels as the unit:

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value for "Horizontal radius" and "Vertical radius" in the "Arc" area.

Define start and end angle

Determine the size of the ellipse segment using the "Start Angle" and "End Angle" attributes. Enter the values using Degrees as the unit:

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value for "Start angle" and "End angle" in the "Arc" area.

Alternative procedure

You can also change the start and end points of the ellipse segment using the mouse.

1. Place the cursor on a start or end point of the ellipse segment. The start and end points are indicated by small blue squares.
   The mouse cursor changes to a cross.
2. Hold down the mouse button, and drag the start or end point to the desired position.

See also

Ellipse (Page 363)

Device dependency of the objects (Page 326)

1.9.2.20 f(t) trend view (RT Professional)

Use

You use the "f(t) trend view" object to display tag values from the current process, or from the log in the form of trends as a function of the time.

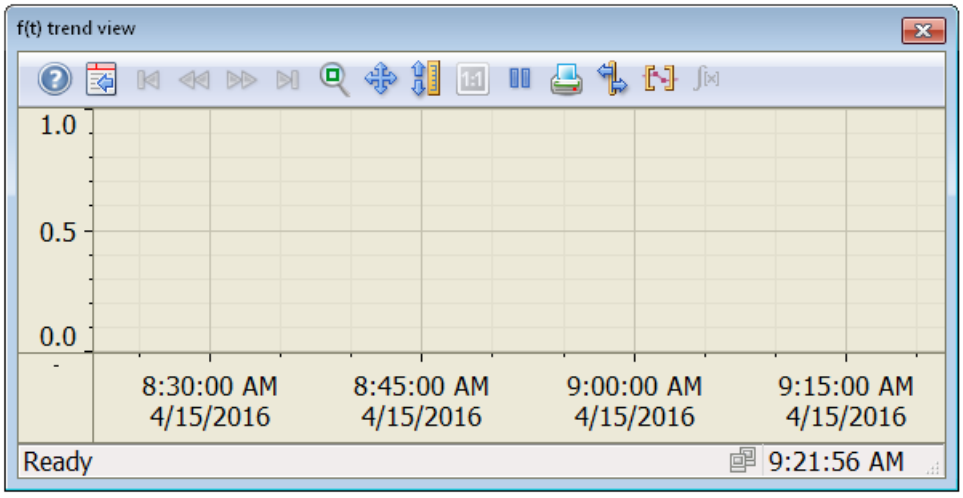

Layout

In the Inspector window, you customize the position, geometry, style, colors and font types of the object. You can adapt the following properties in particular:

- Adding and configuring trends
- Configuring a diagram
- Window settings
- Persistence
- Toolbar

Configuring trends

1. Select the data supply for the trend under "Properties > Properties > Trends > Data source".
   - "Log tags": The trend view is supplied with values from a data log.
   - "User-defined": The trend view is supplied user-defined in Runtime by means of scripts.
   - "Tags": The trend view is supplied with values of a tag.
2. Under "Data range", define whether the trend displays the values statically or if the values are refreshed.
   - "Automatic": The displayed value range is automatically adapted to the current values.
   - "Initial value / End value": You define the minimum value and maximum value for the value range.
3. Under "Limits", configure the colored marking of specific values, .e.g.. high and low limit.
4. Select "Properties > Properties > Time axis".
5. Configure the "Time range" of the trend display.
   - "Time span": You define the time range using a starting time and a following time span.
   - "End time": You define the time range using a starting time and an end time.
   - "Measuring points": You define the time range using a starting time and a number of measuring points.

Configuring a diagram

Configure the display of several trends under "Diagram":

- Common or separate diagrams
- Common or separate axes
- Writing direction of all trends

Window settings

Under "Layout > Window", define whether the user can change the size of the window in runtime and can close the window.

Persistence

Select "Properties > Properties > Security > Persistence" to specify whether or not a user may edit trend view configuration in Runtime and the retention period of such changes.

- "No persistence"
  The changes to the configuration are not saved. The changes are discarded at the next screen change and when Runtime is closed.
- "Persistence"
  The changes to the configuration are saved. The changes are retained even after Runtime was closed. You may also select "Response at screen change" to discard changes at the screen change.
- "Persistence in Runtime"
  The changes to the configuration are only saved during Runtime. The changes are discarded as soon as you close Runtime. You may also select "Response at screen change" to discard changes at the screen change.

Toolbar

You define the operator controls of the f(t) trend view in Runtime under "Properties > Properties > Toolbar" in the Inspector window. The following operator controls are available for the f(t) trend view:

Button	Name	Function
	Online help	Opens the online help.
	Open configuration	Opens the configuration dialog.
	First data record	Scroll in the log: Shows the trend curve starting with the first logged value.
	Previous data record	Scroll in the log: Shows the trend curve of the previous time interval.
	Next data record	Scroll in the log: Shows the trend curve of the next time interval.
	Last data record	Scroll in the log: Shows the trend curve up to the last logged value.
	Show/ hide ruler	Determines the coordinates of a point of the trend.
	Zoom area	Increases the size of any section of the trend window.
	Original view	Switches from the magnified trend view back to the normal view
	Tag selection	Opens the dialog for selecting the log and tags.
	Trend selection	Opens the dialog for setting the visibility of trends.

Button	Name	Function
	Time range selection	Opens the dialog for setting the time range displayed in the trend window.
	Previous trend	Displays the previous trend in the foreground.
	Next trend	Displays the next trend in the foreground.
	Start/end update	Stops and starts the trend update. The values are buffered and updated as soon as you start trend update again.
	Print log	Starts printing the trends shown in the trend window. You define the print job under "General > Print > Print job".
	Selection of the statistics area	Enables you to define a time range for which statistical values are determined. Vertical lines which you use to set the time range are displayed in the trend window.
	Statistics window	Opens a statistics window to display the minimum, maximum, means, and standard deviation for the selected time range and the selected trend.
	Exporting process data in runtime as a CSV file	Opens the dialog for saving the trend data in CSV format.

The order of the buttons is fixed. Under "Hotkey" you can set up individual key assignments, and shortcuts for every button on the toolbar.

See also

f(x) trend view (Page 374)

Table view (Page 497)

Configure trend view (Page 699)

Value table (Page 505)

Configuring persistence (Page 2264)

Device dependency of the objects (Page 326)

1.9.2.21 f(x) trend view (RT Advanced, Panels, Comfort Panels)

Application

You use the "f(x) trend view" object to represent the values of a tag as a function of another tag. This means that you can present temperature trends as a function of the pressure, for example.

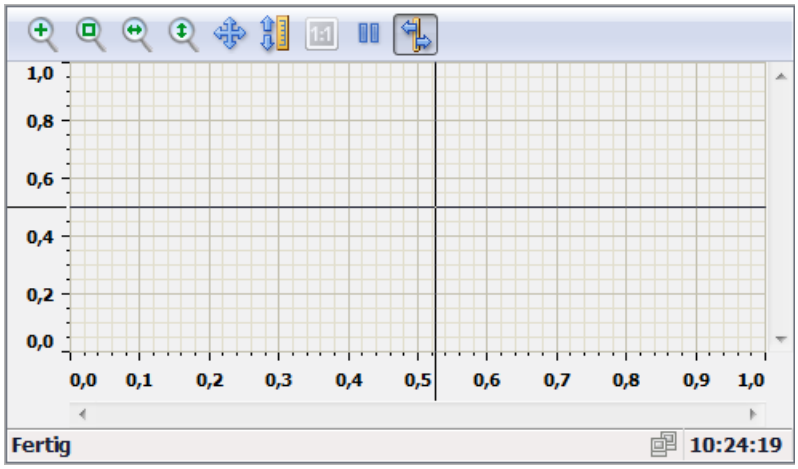

Layout

In the Inspector window, you customize the position, geometry, style, colors and font types of the object. You can adapt the following properties in particular:

- Adding and configuring trends
- Configuring a diagram
- Window settings
- Persistence
- Toolbar

Configuring trends

You create and configure trends in the Inspector window "Properties > Properties > Properties > Trend".

1. Select the data supply for the trend in the "Data source" column.
   - "Logging tag": The trend view is supplied with values from a data log.
   - "Tags": The trend view is supplied with current values from the PLC. The values are constantly updated.
2. Configure the area of the trend display under "Data area".
   - "Time span": You define the time range using a starting time and a following time span.
   - "End time": You define the time range using a starting time and an end time.
   - "Measuring points": You define the time range using a starting time and a number of measuring points.
3. Configure the value range of the trend display under "X axis" and "Y axis".
4. Under "Limits", configure the colored marking of specific values, .e.g. high and low limit, uncertain status.

Configuring a diagram

Configure the display of several trends under "Properties > Properties > Appearance":

- Common diagram or separate diagrams
- Common or separate axes
- Writing direction of all trends

Toolbar

In Runtime, the operator controls of the f(x) trend view are defined in the "Properties > Properties > Toolbar" Inspector window. The following operator controls are available for the f(x) trend view:

Button	Name	Function
	Zoom +/-	Increases or decreases the size of the trends in the trend window. Increase the size of the trends by left-clicking with the mouse. Hold down <Shift> and left-click to decrease the size of the trends.
	Zoom area	Increases the size of any section of the trend window. Specify an area in the trend window by using the mouse. This area of the trend window is enlarged.
	Zoom X axis	Increases or decreases the size of the X axis in the trend window. Enlarge the X axis by left-clicking with the mouse. Hold down <Shift> and left-click to decrease the size of the X axis.
	Zoom Y axis	Increases or decreases the size of the Y axis in the trend window. Enlarge the Y axis by left-clicking with the mouse. Hold down <Shift> and left-click to decrease the size of the Y axis.
	Trend area	Shift the trends in the trend window along the X axis and Y axis by using this button.

Button	Name	Function
	Axis area	Shift the trends in the trend window along the value axis by using this button.
	Original view	Switches from the magnified trend view back to the normal view.
	Start/stop	Stops and starts the trend update. The values are buffered and updated as soon as you start trend update again.
	Ruler	Determines the coordination of a point of the trend.

The order of the buttons is fixed. The operator can set up individual key assignments, and shortcuts for every button on the toolbar.

Notes on performance

The display of tags with short acquisition cycles can cause the data displayed in the f(X) trend view to be updated slowly.

To avoid delayed display in the control, configure the acquisition cycle of the tags greater than or equal to 1 s.

See also

Configure trend view (Page 699)

Device dependency of the objects (Page 326)

1.9.2.22 f(x) trend view (RT Professional)

Use

You use the "f(x) trend view" object to represent the values of a tag as a function of another tag. This means that you can present temperature trends as a function of the pressure, for example. You can also compare the trend to a setpoint trend.

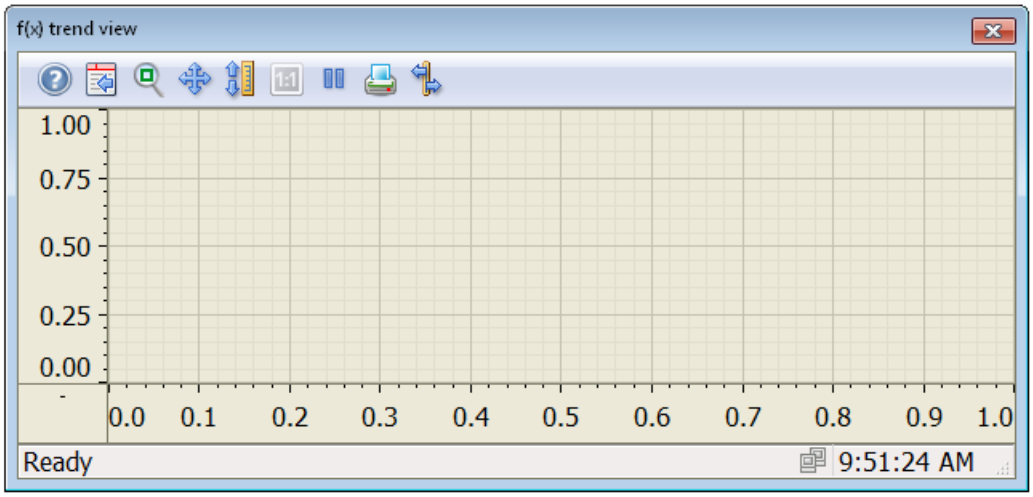

Layout

In the Inspector window, you customize the position, geometry, style, colors and font types of the object. You can adapt the following properties in particular:

- Adding and configuring trends
- Configuring a diagram
- Window settings
- Persistence
- Toolbar

Configuring trends

1. Select the data supply for the trend under "Properties > Properties > Trends > Data source".
   - "Log tags": The trend view is supplied with values from a data log.
   - "User-defined": The trend view is supplied user-defined in Runtime by means of scripts.
   - "Recipe" The trend view is supplied with values from a recipe.
   - "Tags": The trend view is supplied with values of a tag.
2. Configure the area of the trend view under "Data area".
   - "Time span": You define the time range using a starting time and a following time span.
   - "End time": You define the time range using a starting time and an end time.
   - "Measuring points": You define the time range using a starting time and a number of measuring points.
   - "Start ID" and "Number of data records": With data supply via recipes, you define the range by the ID of the first data record and the number of subsequent data records.
3. Configure the value range of the trend display under "X axis" and "Y axis".
   - "Automatic": The displayed value range is automatically adapted to the current values.
   - "Initial value / End value": You define the minimum value and maximum value for the value range.
4. Under "Limits", configure the colored marking of specific values, .e.g.. high and low limit, uncertain status.

Configuring a diagram

Configure the display of several trends under "Layout":

- Common diagram or separate diagrams
- Common or separate axes
- Writing direction of all trends

Window settings

Under "Layout > Window", define whether the user can change the size of the window in runtime and can close the window.

Persistence

Under "Security > Persistence" define how long a user's changes to the trend view in runtime are retained.

- "Allow persistence"
  Users can change settings. Changes are retained until the next screen change.
- "Persistence until logoff"
  User group changes are retained until logoff.
- "Persistence until next loading"
  User group changes are retained until the project is updated on the HMI device.

Toolbar

You define the operator controls of the f(x) trend view in Runtime under "Properties > Toolbar" in the Inspector window. The following operator controls are available for the f(x) trend view:

Button	Name	Function
	Online help	Opens the online help.
	Configuration dialog box	Opens the configuration dialog.
	Zoom +/-	Enlarges and/or shrinks the trends in the trend window. You can enlarge the trend with the left mouse button. You can reduce the size of the trend by holding down <Shift> and left-clicking.
	Zoom area	Increases the size of any section of the trend window. Use the mouse to specify an area in the trend window. This area of the trend window is enlarged.
	Zoom X axis	Enlarges and/or reduces the X axis in the trend window. You can enlarge the X axis with the left mouse button. You can reduce the size of the X axis by holding down <Shift> and left-clicking.
	Zoom Y axis	Enlarges and/or reduces the Y axis in the trend window. You can enlarge the Y axis by left-clicking. You can reduce the size of Y axis by holding down <Shift> and left-clicking.
	Move trend range	You can move the trends in the trend window along the X axis and the Y axis using the button.
	Move axis range	You can move the trends in the trend window along the value axis using the button.
	Original view	Switches from the magnified trend view back to the normal view
	Select data connection	Opens a dialog for selecting logs and tags.
	Select trends	Opens a dialog for setting the visibility of trends.
	Select time range	Opens a dialog for setting the displayed time range in the trend window.
	Previous trend	Displays the previous trend in the foreground.

Button	Name	Function
	Next trend	Displays the next trend in the foreground.
	Start/stop	Stops and starts the trend update. The values are buffered and updated as soon as you start trend update again.
	Print	Click this button to print the trend shown in the trend window. The print job used during printing is defined in the configuration dialog in the "General" tab.
	Export data	This button is used for exporting all or the selected runtime data to a csv file. If the "Display dialog" option is activated, a dialog opens in which you can view the settings for export and start the export. You may also select the file and the directory for the export with the corresponding authorizations. If no dialog is displayed, the export of data to the default file begins immediately.
	Ruler	Determines the coordinates of a point of the trend.
	Connect backup	Opens a dialog in which selected logs are connected to WinCC Runtime.
	Disconnect backup	Opens a dialog in which selected logs are disconnected from WinCC Runtime.

The order of the buttons is fixed. The operator can set up individual key assignments, and shortcuts for every button on the toolbar.

See also

f(t) trend view (Page 367)

Table view (Page 497)

Configure trend view (Page 699)

Value table (Page 505)

Configuring persistence (Page 2264)

Device dependency of the objects (Page 326)

1.9.2.23 Graphic view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Use

The "Graphic view" object is used to display graphics.

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Graphic: Specifies the graphic file that is displayed in the object.
- Adjust graphic: Specifies the automatic size stretching for objects with graphics.
- Transparent color: Specify whether or not the transparent color is used for the graphic.

Inserting graphics

Use the following graphic formats in the "Graphic view" object: *.bmp, *.tif, *.png, *.ico, *.emf, *.wmf, *.gif, *.svg, *.jpg or *.jpeg. You may also use graphics as OLE objects in the Graphic view .

1. In the Inspector window, select "Properties > Properties > General".
2. Select the graphic that you wish to insert.
   The graphic preview is shown in the right pane.
   You have the option to create the graphic from a file using the button, or to create a new graphic from an OLE object using the button.
3. Click "Apply" to insert the graphic in the Graphic view .

Adjust graphic

Whether a graphic displayed in a Graphic view is stretched to the size of the Graphic view in runtime is specified in the Inspector window.

Option	Description
"No auto-sizing"	The size is not automatically adapted in Runtime.
"Fit graphic to object size"	The graphic is automatically adapted to the object size in Runtime.
"Fit object size to graphic"	Graphic display is adapted to the largest used graphic and other graphics are stretched.
"Adjust object size to graphic"	Graphic display is adapted to the largest used graphic but the other graphics retain their original size.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select the required size adjustment for the graphic.

Transparent color

This property defines whether the transparent color is used for the graphic to be displayed.

1. In the Inspector window, select "Properties > Properties > Appearance":
2. Select "Background > Transparent".
3. Select a transparent color.

Note

When using bitmaps in WinCC screens the "Transparent color" setting demands a high character performance in the layout on the panel. Visualization performance is enhanced by disabling the "Transparent" setting in the properties of the relevant display object. This restriction applies in particular when bitmaps are used as background image.

Note

Basic Panels

The "Transparent" property is not available for Basic Panels.

Note

If the selected transparent color of the graphic is not shown properly although the "Transparent" property is activated, the color in the graphic is not entirely consistent with the selected color. For example, the selected transparent color [R=255;G=255;B=255] looks almost exactly like the color in the graphic that is expected to be transparent [R=255;G=251;B=255]. In this case, the graphics used in the object must be reedited. Under "Project Navigation > Languages & Resources > Graphics", select the graphic that you want to display transparently. From the shortcut menu of the default graphic, select the "Edit" command and fill in the background of the graph with the desired color that appears transparent in the screen. Save the graphic.

See also

Storing an external image in the graphics library (Page 149)

Options for editing objects (Page 104)

Overview of objects (Page 93)

Device dependency of the objects (Page 326)

1.9.2.24 Graphic I/O field (RT Advanced, Basic Panels, Panels, Comfort Panels)

Application

The "Graphic I/O field" object can be used to configure a list for display and selection of graphic files.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Scroll bar type: Specifies the graphic layout of the scroll bar.
- Mode: Specifies the response of the object in Runtime.

Note

Scroll bar

The "Scroll bar > Type" property is available for Panels and Runtime Advanced up to device version V12.

Note

Reports

Graphic I/O fields output exclusively graphics in reports. "Output" mode is preset. Properties for configuring the selection of graphics are not available, e.g. "scroll bar".

Note

Border

You can configure the width and the style of the border of a graphic I/O field in the "Two states" mode.

The dynamization of the border colors in "Two states" mode has no effect during runtime.

Mode

The response of the "Graphic I/O field" object is specified under "Properties > Properties > General > Type > Mode" in the Inspector window.

Mode	Description
"Input"	The "Graphic I/O field" object is only used to select graphics.
"Input/output"	The "Graphic I/O field" object is used to select and display graphics.
"Output"	The "Graphic I/O field" object is used to display graphics only.
"Two states"	The "Graphic I/O field" object is only used to display graphics and can have a maximum of two states. You use no graphics list but insert one graphic each for the "ON" and "OFF" state.

Adjust graphic

Whether a graphic displayed in a graphic I/O field is stretched to the size of the view in Runtime is specified in the Inspector window.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select the required size adjustment for the graphic.

Scroll bar type

The response for the graphic representation of the scroll bar is specified under "Properties > Properties > Appearance > Scroll Bar > Type" in the Inspector window.

Type	Description
"Permanent"	The scroll bar is always visible.
"No scrollbar"	The scroll bar is not visible.
"Visible after clicking"	The scroll bar is made visible by a mouse click.

Behavior during operation

If the graphic I/O field displays a cactus image, a graphic to be output for a specific value has not been defined in the project. The contents of the graphic I/O field change color to show that it is now activated. The frame in 3D is only shown graphically in an output field.

See also

Symbolic I/O field (Page 485)

Print job/script diagnostics (Page 352)

Device dependency of the objects (Page 326)

1.9.2.25 Graphic I/O field (RT Professional)

Use

With the "Graphic I/O field" object you configure a graphics list which is used to display and select graphic files.

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in runtime.
- Fit to size: Specifies the ratio of the display window and the embedded graphic.

Mode

The response of the "Graphic I/O field" object is specified in the Inspector window in "Properties > Properties > General > Mode".

Mode	Description
"Output"	The "Graphic I/O field" object is used to display graphics only.
"Two states"	The "Graphic I/O field" object is only used to display graphics and can have a maximum of two states. You use no graphics list but insert one graphic each for the "ON" and "OFF" state.

Fit to size

The ratio for the graphic representation in the display window is specified under "Properties > Properties > Layout > Fit to size" in the Inspector window.

Sizing type	Description
"Fit object size to content"	The display window is adapted automatically to the size of the graphic.
"Fit content to object size"	Adjust the graphic to the display window size.

See also

Symbolic I/O field (Page 487)

Device dependency of the objects (Page 326)

1.9.2.26 GRAPH overview (RT Professional, RT Advanced, Panels, Comfort Panels)

Use

The "GRAPH Overview" object is used to display the current program status for executed steps of the GRAPH sequencer. Errors during execution of a program are displayed directly at the corresponding step.

The following information is displayed in the "GRAPH Overview" object:

- Name and status of the function block
- Status of initial and simultaneous steps
- Number and name of the first step currently executed step
- Operating mode for running the GRAPH sequencer

WinCC supports the display of step names for the GRAPH blocks in multiple languages starting from Version 5.0. The step names will then be displayed in the selected Runtime language following a language changeover in Runtime. If the selected language is not available in the GRAPH block, the names are displayed in the default language (English).

Note

The "GRAPH Overview" object cannot be used in faceplates or in the global screen.

Note

Device dependency of the "GRAPH Overview" object

The "GRAPH Overview" object is available for RT Advanced, RT Professional, Comfort Panels and KTP Mobile Panels.

Note

Requirement for display in GRAPH overview

For the display of the program status of an S7 GRAPH instance data block in the "GRAPH overview" object to be possible, the instance-specific properties of the block must be set as "Visible in HMI" and "Accessible from HMI".

Layout

In the Inspector window, you customize the position, style, colors and font types of the object. You can adapt the following properties in particular:

- Assigned GRAPH DB tag
- Buttons on the toolbar

Operating mode

Four operating modes are available for running the GRAPH sequence:

- Auto (default setting) - Automatically switches to the next step when the transition is fulfilled.
- TAP - Automatically switches to the next step when the transition is fulfilled and there is an edge change from "0" to "1" at the T_PUSH parameter.
- TOP - Automatically switches to the next step when the transition is fulfilled or there is an edge change from "0" to "1" at the T_PUSH parameter.
- Manual - The next step is not automatically enabled when the transition is fulfilled. The steps can be selected and deselected manually.

Note

You set the operating mode by modifying the interface parameters of the GRAPH block in your control program.

In WinCC Runtime Professional, you have the option of editing the name of the operating mode that is displayed in the GRAPH overview.

Configuring three-line mode with criteria analysis

In three-line mode you see additional lines with the previous and next steps in the GRAPH overview as well as the first operand with error of the criteria analysis.

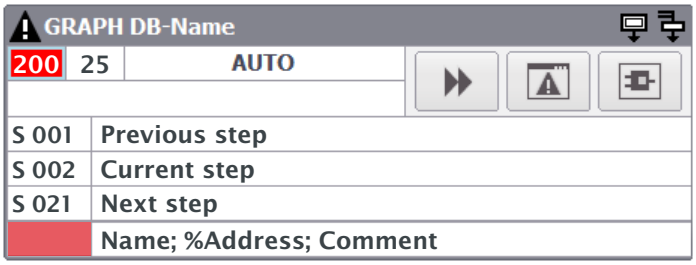

- To display the previous and next steps, select the "Previous and next step" option under "Properties > Appearance > Options".
- To display the first operand with error of the criteria analysis, select the "Criteria analysis" option under "Properties > Appearance > Options".

Note

The criteria analysis will not output any information in the GRAPH overview in the following cases:

- If the version of the GRAPH block is older than V4.0
- If the initial value acquisition was not activated or if the initial values were not acquired
- If no error is present in the current step

Configuring a compact view

You can also configure a compact GRAPH overview without toolbar buttons and operating mode display, as was previously possible in WinCC Professional versions before V14.

To display the GRAPH overview compactly in single-line compatibility mode, select the "Single-line mode" option under "Properties > Appearance > Options".

Symbols

You specify the symbols that are displayed in the GRAPH Overview in the Inspector window under "Properties > Appearance > Options". The following symbols are available for GRAPH Overview:

Symbol	Name	Function
	Error	Indicates that an error has occurred during the execution of a step.
	Initial step	Indicates that the currently executing step is the first step in the GRAPH block.
	Simultaneous step	Shows that there are other simultaneous steps in the GRAPH block in addition to the current one.

Buttons

You specify the buttons that are displayed in the GRAPH Overview under "Properties > Toolbar > General".

Button	Name	Function
	Next step	Jumps to the next step, if the GRAPH block contains more than four steps.
	Last step	Jumps to the first step of the sequencer when the final step has been reached.
	Alarm view	Opens the configured alarm view with the error message in Panels and RT Advanced. Opens the configured alarm view with the error message in RT Professional.
	PLC code view	Opens the configured PLC code view.

Display the current program status

A PLC including a GRAPH instance data block has been created and contains at least one PLC tag which is visible in and accessible from the HMI.

1. Move the GRAPH overview out of the toolbox window using drag-and-drop.
2. In the Inspector window, select "Properties > Properties > General".
3. Open the selection button under "Process > Tag".
   The "Add new object" dialog opens.
4. Select the corresponding PLC in the "Program blocks" folder.
5. Select the corresponding PLC tag of the GRAPH instance data block.
6. You can dynamize the properties in the "GRAPH Overview" object using tags and color-code the states.
   You can find a detailed description of dynamizing a property in the section Dynamizing screens (Page 205).

Note

If no connection exists between the HMI device and the selected PLC, a connection is set up automatically.

In addition, an HMI tag is created which is connected to the PLC tag.

See also

Initial value acquisition and criteria analysis (Page 1915)

Supervising machinery and plants with ProDiag (Page 1852)

Configuring a GRAPH overview (Page 1870)

Displaying the status of the GRAPH sequencer (Page 1866)

Configuring property animation of the Range type (Page 227)

Objects for WinCC Runtime Professional (Page 333)

Dynamizing screens (Page 205)

1.9.2.27 Handwheel (Panels, Comfort Panels)

Use

The object "Handwheel" is a control element for some SIMATIC Mobile Panels. You can enter incremental values in Runtime using the "HandWheel" object.

Layout

You can set the following property in the Inspector window:

- Tag: Specifies the tags to which the "HandWheel" object is linked.

Global assignment

The "Tag" property is used to specify the global assignment of the "HandWheel" object in the global screen. There is only one global screen per HMI device.

1. In the Inspector window, select "Properties > Properties > General".
2. Select a tag from the selection list under "Settings > Tag".

Local assignment

The "HandWheel" object is assigned to another tag. To do so you configure a function that assigns the tag to the handwheel in Runtime.

Example for configuration to a button:

1. Open the screen in which the "HandWheel" object is assigned to another tag.
2. Drag the "Button" object from the toolbox to the screen. Select the button on your screen.
3. Click "Properties > Events > Click" in the Inspector window.
4. The "Function list" dialog box opens. Click the first line of the function list. The list of system functions and scripts available in the project appears.
5. Select the system function "SetTagToHandWheel" from the "Other functions" group. Select a tag from the combo box at "Value".

Note

The Mobile Panel may ignore the change of the value in the PLC made at another location while the handwheel is being operated. The value is the PLC is overwritten again.

See also

Illuminated pushbutton (Page 411)

Device dependency of the objects (Page 326)

1.9.2.28 Help indicator (Basic Panels)

Application

The object "help indicator" is available for the HMI device KP300 Basic. If a a tooltip exists for the selected object, the help indicator is displayed during runtime. If a tooltip was configured for the opened screen, the help indicator always remains visible.

You configure the object "help indicator" exclusively in the global screen.

Layout

You can adapt the following properties in the Inspector window:

- Position: Determines the position of the object "Help indicator."

Position

You can use this property to set the position of the object "Help indicator."

1. Select the object "Help indicator" in the template.
2. In the Inspector window, select "Properties > Properties > Layout".
3. Enter a value for X and Y. You can also use the cursor keys to position the selected object.

If you have configured a screen object at this position, the visible help indicator covers the screen object. The help indicator is covered only by incoming system alarms and dialogs.

See also

Device dependency of the objects (Page 326)

1.9.2.29 HTML browser (RT Advanced, Basic Panels, Panels, Comfort Panels)

Use

The "HTML Browser" object is designed for the visualization of simple HTML pages. This function allows you to draw up machine-specific descriptions which are stored centrally and which can then be displayed from different HMI devices.

Note

Switching the functionality of the HTML Browser to that of a file explorer by one of the following methods, for example, is not authorized in the context of WinCC:

- Entering a folder or drive, for example "\" or "C:", or
- Connecting to an FTP server, for example "ftp://"

One reason is to prevent inadvertent changes to files, their deletion or execution.

When configuring, ensure that the operator can only enter valid Internet addresses, for example, by using symbolic I/O fields. Configure a password-protected input for service purposes.

Note

Only one instance of the HTML Browser is supported per screen for Panels to preserve memory space.

If you move quickly from a slide-in-screen to a pop-up screen and an HTML browser is configured in both screens, the second HTML browser will not open. After closing the slide-in screen, wait two seconds until the handle is no longer visible and then open the pop-up screen.

Note

Device-based dependency of the "HTML browser" object

The "HTML browser" object is available for Basic Panels, Comfort Panels and KTP Mobile Panels starting with a 7" display size and RT Advanced.

Layout

Customize the object position and size in the Inspector window. In particular, you can customize the following property:

- URL: Specifies which Internet address is opened in the HTML Browser.
- Tag for URL: Specifies the tag which sets the Internet address.

Address

The Internet address is specified in "Properties > Properties > General >URL" in the Inspector window.

Note

For Basic Panels 2nd Generation:

If you want to load a website from a USB flash drive to your HMI device, the address in the "URL" field must begin with the prefix "\media\usb\".

Example: You want to load the html5.htm web page from the USB device USB_XY. Enter "\media\usb\USB_XY\html5.htm" in the Inspector window.

Note

The DNS service for Internet access with the "HTML browser" object is not supported for Basic Panels 2nd Generation.
This means only access via the IP address is supported.

Browser type (RT Advanced as of V13 SP1)

You can select the browser type in Runtime Advanced as of version V13 SP1.

Specify the type of the Web browser under "Properties > Properties > General > HTML browser type" in the Inspector window.

Two options are available:

- Web browser based on WebKit engine without Active X support

  Note

  Dialogs from websites are not supported in the WebKit browser. "Yes" or "OK" is always set as answer instead.

- Web browser based on Internet Explorer with Active X support

  Note

  If Internet Explorer does not load the website, use the WebKit browser without Active X support.

Basic procedure

1. Create an internal tag of the "String" data type, for example, InternetAddress, in the "Tag" editor.
2. Insert the "HTML Browser" object in the screen in the "Screens" editor.
3. In the Inspector window, select "Properties > Properties > General".
4. At "Tag for the URL", select the "InternetAddress" tag from the selection list.
5. Insert an I/O field in the screen in the "Screens" editor.
6. In the Inspector window of the IO field "Properties > Properties > General > Tag for URL", select the "InternetAddress" tag from the selection list.

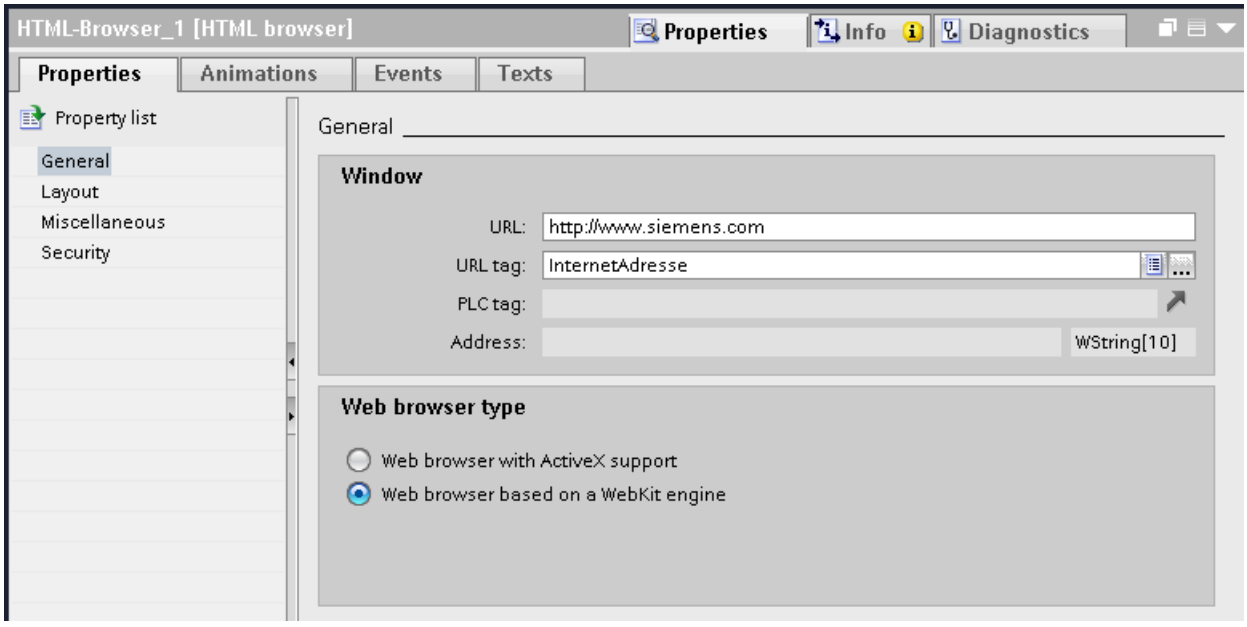

The HTML Browser opens the relevant page after the operator has entered an address in the I/O field.

Operator controls

Operator controls are configured buttons in the process screen or softkeys on the HMI device.

The "HTML Browser" object does not have its own operator controls. Assign the system functions used to control the "HTML Browser" object to your configured operator controls, such as buttons.

1. For example, drag the "Button" object to the screen from the toolbox.
2. Input a text of any length or select a graphic, for example Update.
3. In the Inspector window, select "Properties > Events > Click".
   The "Function list" dialog opens.
4. The system functions for controlling the "HTML Browser" object can be found in the list of system functions in the "Keyboard operation for screen objects" group.
   Select a function from the list, for example, HTMLBrowserRefresh.
5. Select the object name of the HTML Browser in which the command will be executed from the "Screen object" list.

Defining the website selection

To define the website selection that is provided in the HTML browser, you can define a number of allowed URL addresses.

You define a String or Wstring type tag or for the "Tag for URL" property of the HTML browser that can be filled with the predefined URL addresses in the following ways:

- Using the "SetTag" system function, which you configure on a button
- Using an I/O field

  Note

  Any website can be accessed by entering the address in an I/O field.

  To restrict the selection of permitted websites, use the provision of the pre-defined URL addresses via a system function or a script.

- Via a script (Panels and RT Advanced only)
- Via the PLC

Displaying SIMOTION PLC websites in the HTML browser

If the SIMOTION PLC websites are not displayed correctly in the HTML browser, add "/basic" after the website URL to display the websites in basic mode.

Comments

The functional scope of the HTML Browser is limited in comparison to the Internet Explorer:

- The HTML Browser will only show pure HTML pages. VBScript, Java, JavaScript, Flash and ActiveX controls are not supported. Set up the HTML pages for display in the HTML Browser by using a text editor or using a simple HTML editor.
- Links to embedded files, for example *.pdf or *.xls, are not supported.
- Queries and dialogs that are conducted during the access of, for example protected pages are not supported. When accessing protected pages, enter your user name and password in the URL: <http://username:password@servername> (e.g. "http://Administrator:Admin123@192.168.0.199").

Note

The object "HTML Browser" supports dynamization.

If you select the HTML browser type "Web browser, based on a WebKit engine", only the tag connection is available for dynamization under "Address".

Note

For Basic Panels 2nd Generation and Comfort Panels:

The object "HTML Browser" does not support proxy settings. This means the "HTML Browser" object cannot access URLs that require a change of the proxy settings.

Note

HTML browser in "Scroll indicator" mode

If you have specified the "Scroll indicator" scroll mode in "Runtime settings > Screens > Scrolling in controls", it may occur that some websites are not correctly displayed.

Note

HTML browser in "Web browser, based on WebKit Engine" mode with screen scroll bars

You configure the "HTML browser" object in the "Web browser based on WebKit Engine" mode and activate the scrolling mode for the display. The right and the lower bars are activated for scrolling in Runtime.

If the bars of the displayed HTML page and the HTML browser object overlap in Runtime, operate the scroll bars of the HTML browser with a mouse click and double-click on the scroll bars of the website. If you have already configured the object in the HMI screen, try to avoid the double scroll bars by adjusting the object to the size of the website to be displayed.

See also

Web browser of WebKit engine: Overview (Page 2236)

Device dependency of the objects (Page 326)

1.9.2.30 HTML browser (RT Professional)

Application

The "HTML Browser" object is designed for the visualization of simple HTML pages. This function allows you to draw up machine-specific descriptions which are stored centrally and which can then be displayed from different HMI devices.

Note

Changing the functionality of the HTML Browser to that of a file explorer by one of the following methods, for example, is not authorized in the context of WinCC:

- Entering a folder or drive, for example "\" or "C:" or
- Connecting to an FTP server, for example "ftp://"

One reason this function is not implemented is to prevent inadvertent changes to files, their deletion or execution.

When configuring, ensure that the operator can only enter valid Internet addresses, for example, by using symbolic I/O fields. Configure a password-protected input for service purposes.

Layout

Customize the object position and size in the Inspector window. In particular, you can customize the following property:

- Address: Specifies which Internet address is opened in the HTML Browser.
- Toolbar: Specifies whether a navigation toolbar is shown.

Address

The Internet address is specified in the Inspector window under "Properties > Properties > General > Window > URL".

Basic procedure

1. Create an internal tag of the "String" data type, for example InternetAddress, in the "Tag" editor.
2. Insert the "HTML Browser" object in the screen in the "Screens" editor.
3. In the Inspector window, select "Properties > Properties > General".
4. For "Tag for the URL", select the "InternetAddress" tag from the selection list.
5. Insert an I/O field in the screen in the "Screens" editor.
6. Select the "InternetAddress" tag at "URL tag" from the selection list under "Properties > Properties > General" in the Inspector window.

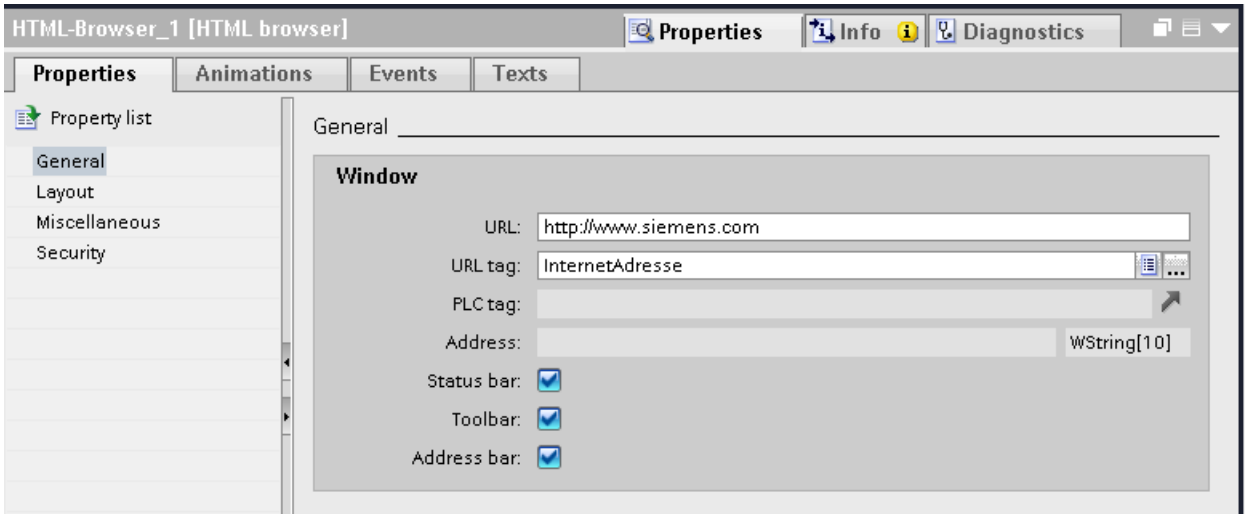

When the operator enters or activates an address, the HTML Browser opens the relevant page.

Comments

The functional scope of the HTML Browser is limited in comparison to the Internet Explorer:

- The HTML Browser will only show pure HTML pages. VBScript, Java, JavaScript, Flash and ActiveX controls are not supported. Set up the HTML pages for display in the HTML Browser by using a text editor or using a simple HTML editor.
- Links to embedded files, for example *.pdf or *.xls, are not supported.
- Queries and dialogs that are conducted during the access of, for example protected pages are not supported. When accessing protected pages enter your user name and password in the URL: <http://User_Name:Password@Server_Name> (for example, "http://Administrator:Admin123@192.168.0.199").

See also

Device dependency of the objects (Page 326)

1.9.2.31 Camera view (RT Advanced, Panels, Comfort Panels)

Use

You use the "Camera view" object to configure the camera view which outputs screens for the operator in runtime from a connected network camera.

Monitoring with the help of the network camera lends itself for situations in which the operator has limited visibility of the plant or cannot get close to the machines. The operator can then respond to any problem much faster to remedy its cause.

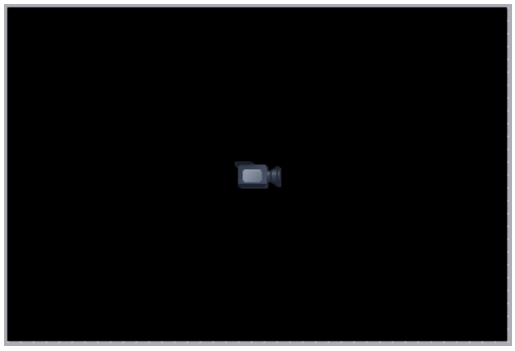

The camera view can be configured in the following screens:

- In a screen
- In a template
- In a slide-in screen
- In a pop-up screen

Note

Device dependency of the "Camera view" object

The "Camera view" object is available for Comfort Panels, KTP Mobile Panels (as of device version V13) and RT Advanced.

Note

Only one camera view may be in operation in runtime at the time.

Note

Use the "Camera view" object only with network cameras.
Make sure that the connected network camera is located in a secure network with known devices and IP addresses.

For more information on security, go to:

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

Layout

You enter the information on the source of the video stream as well as position and size of the camera view in the Inspector window. You can adapt the following properties in particular:

- URL for the source of the video stream:
  - Camera URL: Specifies a static URL which is only entered during configuration.
  - Camera URL tag: Specifies a dynamic URL which the operator can change later in runtime.
- Transmission protocol: Specifies the transmission protocol by which the data exchange between the panel and the network camera takes place.
- Position and size of the camera view: Specifies position and size of the camera view as well as the aspect ratio and size of the displayed video stream.

Supported network cameras and recommended settings

Use the "Camera view" object with the network cameras which support the streaming protocol RTP/RTSP and MJPEG, H.264 and MPEG4 codecs.

You can find information on the supported network cameras and recommended settings under Supported network cameras and settings. (Page 2205)

Specifying URL for the source of the video stream

1. Insert the camera view in the screen in the "Screens" editor. The camera view is found in the task card "Tools" > "Controls".
2. Click "Properties > Properties > General > Media" in the Inspector window.

3. Enter the static camera URL in the "Camera URL" field, for example, "rtsp://user:password@192.168.56.208" for a Siemens camera.
   The address of the network camera is available in the network camera manual.

   Note

   The camera URL always starts with "rtsp://" and must only include permitted characters. If you enter characters that are not permitted or if the URL does not start with "rtsp://", the camera cannot be operated in runtime. The operator will receive an error message in the "Camera view" to indicate the problem.

   The following characters of the US ASCII character set are permitted:

   0123456789

   A...Z a...z

   !#$%&()*+,'-_./:;<=>?@ [\]_{|}~^(")`

4. Assign a tag to the camera view under "Camera URL tag" in the Inspector window. The tag must be of the data type "String" for external tags and "WString" for internal tags.
   When you assign a tag to the camera view, the user can change the source of the video stream in runtime.

Note

If you leave the "Camera URL tag" field blank, the static URL is used in runtime.

If you enter information in the "Camera URL" and "Camera URL tag" field, the dynamic URL is used in runtime.

Specifying the transmission protocol

1. Click "Properties > Properties > General > Media" in the Inspector window.
2. Specify the transmission protocol with the "Use UDP instead of TCP" check box.
   The selection of the protocol depends on the protocol types which your network camera supports. Information on supported protocols is available in the network camera manual.
   In most cases, a TCP connection has reliable synchronization with the camera view.

Specifying position and size of the camera view

1. Click "Properties > Properties > General > Layout" in the Inspector window.
2. Enter the size of the object under "Camera view" in the "Position & Size" area. To achieve the best picture quality of the video stream, enter the physical resolution of the network camera as size of the camera view.
   The recommended setting "Keep video size" is selected by default in the "Fit to size" area. The "Keep aspect ratio" check box is disabled and cannot be selected.
3. To keep the aspect ratio of the video stream, disable "Keep video size".
   The "Keep aspect ratio" check box is automatically selected.

Note

The screen is not scaled when you keep the size of the video stream. The screen size of the network camera is output 1:1 by the camera view. If the camera view is smaller than the selected resolution of the network camera, only a corresponding section of the camera screen is output by the camera view. If the camera view is greater than the selection solution of the network camera, any blank spaces are displayed as black horizontal or vertical margins. For optimal performance on panels, we recommend using the "Keep video size" option and configuring the camera view to match the resolution of the camera in order to avoid black borders in the camera view.

Any blank spaces are output as black horizontal or vertical margins by the camera view when you keep the aspect ratio of the video stream.

Note

When you disable the "Keep video size" and "Keep aspect ratio" check boxes, the camera screen may be distorted.

Comments

Note

You cannot configure system functions for the camera view.

Note

When you test a project with the simulator, a black screen with a camera symbol is output.

See also

Objects for WinCC Runtime Advanced (Page 331)

Supported network cameras and settings (Page 2205)

1.9.2.32 Channel diagnostics display (RT Professional)

Application

You use the "Channel diagnostics display" object to monitor the active connections in Runtime. The "Channel Channel diagnostics display" object allows you to output the following status and diagnostic information for the channel units:

- Output status/statistical information about the connections in Runtime
- Text output to a logbook file for fault analysis, and troubleshooting
- Text output to a trace file to help the Hotline in pinpointing the cause of connection problems

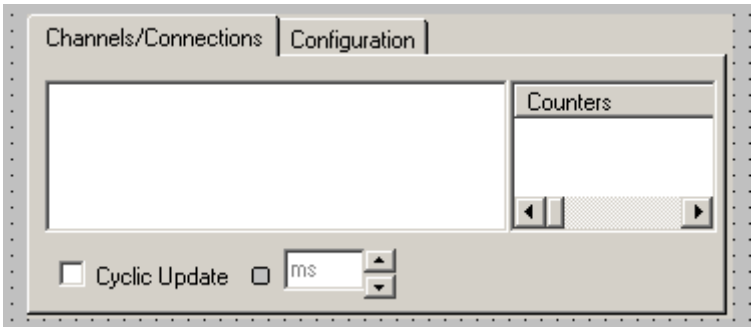

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color.

Check channel and connection

To do this, activate your project. Status information on all configured connections is displayed in the object "Channel diagnostics view" in the Channels/Connections tab.

1. Open a screen.
2. Insert the "Channel diagnostics view" from the "Controls" object pane into the screen.
3. Save the screen.
4. Start Runtime.
5. Select the screen in which you have configured the Channel diagnostics view.
6. The configured connections are displayed in the "Channels/Connection" tab.
7. Check the icons in front of the connection name. If the status of a channel, and connection are OK, a green check mark is displayed in front the associated entry.
8. If there is no green check mark in front of the name of channel and connection, click a connection. The status information for this connection is displayed in the area on the right.
9. Switch to the "Configuration" tab. Select one of the displayed channels and configure which error displays are to be entered in the associated log file.

10. Check the channel-specific log file. Open the log file from the "Siemens\Automation \SCADA_2007\WinCC\Diagnose" folder in a text editor.
11. Check the latest entries with the "ERROR" flag.
12. Activate the trace function.
    Contact Customer Support if you are unable to locate the error, even after checking the log file.

Overview of the status messages

The following statuses are displayed in the "Channel diagnostics display" object:

- Channel / connection ready without restriction
- Channel / connection ready with some restrictions
- No information available about connection status
- Channel / connection failed

See also

Device dependency of the objects (Page 326)

1.9.2.33 Combo box (RT Professional)

Application

You use the "Combo box" object to present and select entries in a drop-down menu. You activate an entry by default so that the operator only changes the preset value if necessary. To incorporate combo boxes into the process, dynamize the corresponding properties.

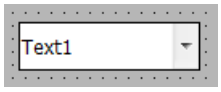

Layout

In the Inspector window, you customize the position, style, colors and font type settings of the object. You can adapt the following properties in particular:

- Number of entries: Defines the number of combo boxes.
- Selection of entries: Defines which entry is displayed as activated by default.

Defining the number of entries

1. In the Inspector window, select "Properties > Properties > General".
2. Define the desired number of list boxes under "Label > Entries".

At the same time, the value of the "Entries" attribute specifies the high limit for the "Index" property under "Label." Changing the value can have the following effects:

Changing the number

New fields are added under the field with the highest value for the "Index" property if you increase the number of entries.

All fields for which the value of the "Index" property is higher than the new number will be deleted if you reduce the number of entries.

See also

List box (Page 416)

Configure operator input alarms (Page 827)

Device dependency of the objects (Page 326)

1.9.2.34 Check box (RT Professional)

Application

You use the "Check box" object to present and select multiple options. You activate check boxes by default so that the operator only changes the preset value if necessary. The operator can select several options in Runtime. To incorporate a check box into the process, dynamize the corresponding properties.

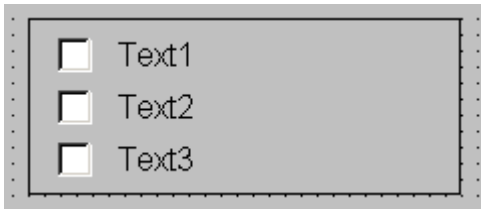

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Number of check boxes: Defines the number of options.
- Check box selection: Defines which options are displayed as activated by default.

Define number of check boxes

1. In the Inspector window, select "Properties > Properties > General".
2. Define the desired number of check boxes under "Label > Entries".

At the same time, the value of the attribute specifies the upper limit for the "Index" property under "Label." Changing the value can have the following effects:

- Increases the number
  New fields are added under the field with the highest value of the "Index" property. You can change the default label for the new field using the "Text" property under "Label."
- Reduces the number
  All fields for which the value of the "Index" property is greater than the new number are deleted.

Define the default setting of the check boxes

In the table under "Properties > Properties > General > Label > Presetting enabled", you define which options are shown activated in a check box list. You can activate multiple options.

Each option is represented by a bit in a 32-bit word. To activate a field, the corresponding bit must have the value "1". The 32-bit word contains the information for all options of the check box list. The value of the "Presetting enabled" property is specified in hexadecimal format.

See also

Option buttons (Page 439)

Device dependency of the objects (Page 326)

1.9.2.35 Circle (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Application

The "Circle" object is a closed object which can be filled with a color or pattern.

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Radius: Specifies the size of the circle.

Radius

The radius of the "Circle" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 in the "Radius" area.

See also

Circular arc (Page 403)
Circle segment (Page 404)
Device dependency of the objects (Page 326)

1.9.2.36 Circular arc (RT Professional)

Application

The "Circular arc" is an open object. Use the "Circle segment" object if you want to fill the object with color. By default, a circular arc is a quarter circle. It can be customized as required.

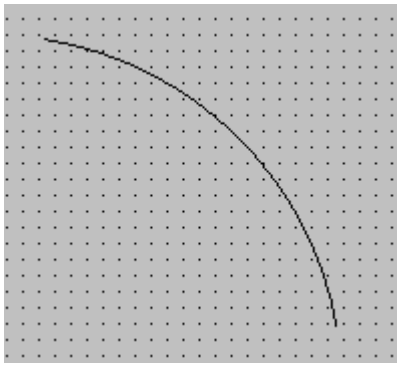

Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Radius: Define the size of the circular arc.
- Start and end angle: Specify where the start and end angle lie on a virtual circle of 360°.

Defining the radius

Define the horizontal, and the vertical radius of the "Circular arc" object in the Inspector window. Enter the values using Pixels as the unit.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value for "Radius" in the "Arc" area.

Define start and end angle

Determine the length of the circular arc using the "Start Angle" and "End Angle" attributes. Enter the values using Degrees as the unit.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value for "Start angle" and "End angle" in the "Radii" area.

Alternative procedure

You can also change the start and end points of the circular arc using the mouse.

1. Place the cursor on a start or end point of the circular arc. The start and end points are indicated by small blue squares.
   The mouse cursor changes to a cross.
2. Hold down the mouse button, and drag the start or end point to the desired position.

See also

Circle (Page 402)

Circle segment (Page 404)

Device dependency of the objects (Page 326)

1.9.2.37 Circle segment (RT Professional)

Application

The "Circle segment" is a closed object that you can fill with a color or pattern. By default, a circle segment is a quarter circle. It can be customized as required.

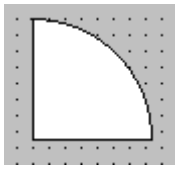

Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Radius: Define the size of the circle segment.
- Start and end angle: Specify where the start and end angle lie on a virtual circle of 360°.

Radius

You define the radius of the "Circle segment" object in the Inspector window. Enter the value using Pixels as the unit.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value for "Radius" in the "Arc" area.

Define start and end angle

Determine the size of the circle segment using the "Start Angle" and "End Angle" attributes. Enter the values using Degrees as the unit.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value for "Start angle" and "End angle" in the "Radii" area.

Alternative procedure

You can also change the start and end points of the circle segment using the mouse.

1. Place the cursor on a start or end point of the circle segment. The start and end points are indicated by small blue squares.
   The mouse cursor changes to a cross.
2. Hold down the mouse button, and drag the start or end point to the desired position.

See also

Circle (Page 402)

Circular arc (Page 403)

Device dependency of the objects (Page 326)

1.9.2.38 Criteria analysis view (RT Professional, RT Advanced, Panels, Comfort Panels)

Use

The "Criteria analysis view" object shows you the operands with error in the user program that have triggered a selected ProDiag or GRAPH alarm. As a result, you have the option of seeing the list of operands with error in addition to the alarm in the same screen.

To see the evaluation of the criteria analysis in the "Criteria analysis view" object in Runtime, select the initial value acquisition in the settings of the function blocks in the user program. Initial value acquisition is available for GRAPH function blocks Version 4.0 and higher and ProDiag function blocks V2.0 and higher.

To enable the link to the corresponding error message, configure a reference to a previously configured alarm view. If you select a GRAPH alarm or a ProDiag alarm in the alarm view in Runtime, then the name, address, comment and value of the operand that caused this error is displayed in the criteria analysis view.

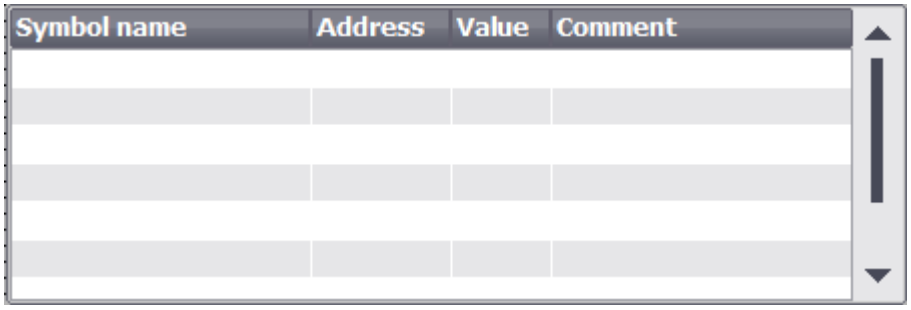

You see the incoming alarms and operands with error at a glance in Runtime if you configure the criteria analysis view and its linked alarm view in the same screen.

Note

Criteria analysis is only available for the user programs for which initial value acquisition has been activated.

Activate initial value acquisition in the properties of the following blocks:

- ProDiag function blocks with version greater than or equal to V2.0
- GRAPH function blocks with version greater than or equal to V4.0

Note

The "Criteria analysis view" object is not available in the global screen.

Layout

You change the settings for the position, style, colors, and fonts of the object in the Inspector window.

Columns

The following columns are displayed in the criteria analysis view in Runtime.

Column	Description
Symbol name	Symbolic name of the operand in the user program.
Address	Absolute address of the operand.
Comment	Additional comments from the user program in the language that is loaded into the controller.
Value	The value of the operand at the time of the error.

Configuring the criteria analysis view

1. Move the criteria analysis view from the toolbox window using drag-and-drop.
2. Select "Properties > Properties > General" in the Inspector window.

3. Open the selection button under the "Tag" property.
4. Select the status tag of the corresponding alarm view.
   To enable linking to alarms, enter the tag for the criteria analysis for the configured alarm view under "Properties > Properties > Display".
5. In the Inspector window, select "Properties > Properties > Columns".
6. Select the columns that you require in the device view for Runtime.
7. Customize the headers and the width of the columns if necessary.

Configuring the criteria analysis view (RT Professional)

1. Move the criteria analysis view from the toolbox window using drag-and-drop.
2. Select "Properties > Properties > General" in the Inspector window.
3. Enter the name of the configured alarm view under "Data source".

Using the keyboard to operate the criteria analysis view

To operate the PLC code view with the keyboard, proceed as follows:

1. Press the <Tab> key repeatedly until the focus is on the desired button.
   Use <Shift + Tab> to navigate back to the previous button.
2. Press <Enter>.
   The command is executed.

See also

Overview of initial value acquisition and criteria analysis (Page 1915)

Example of order of the criteria analysis (Page 1919)

Initial value acquisition and criteria analysis (Page 1915)

1.9.2.39 Trend view (RT Advanced, Basic Panels, Panels, Comfort Panels)

Application

The trend view is meant for the graphical representation of tag values from the current process or from a log in form of trends.

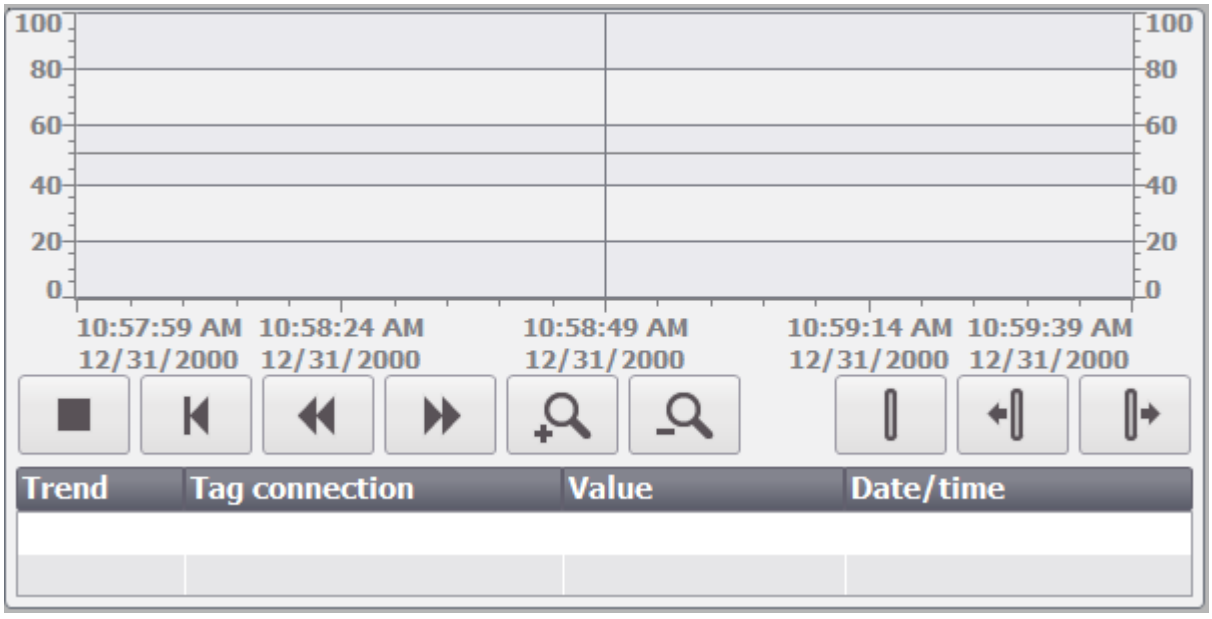

Layout

In the Inspector window, you customize the position, geometry, style, color, and font types of the object. You can adapt the following properties in particular:

- Display value table, ruler and grid: Specifies whether a value table, a ruler or a grid is displayed in addition to the coordinate system to improve legibility.
- Toolbars: Defines the display of the operator controls.

Display value table, ruler and grid

For improved legibility a value table, a ruler and a grid can be displayed in Runtime.

1. Activate "Properties > Properties > Appearance > Show ruler".
2. Activate "Properties > Properties > Table > Show table".
3. Activate "Properties > Properties > Table > Show grid".

Toolbars

The layout of the operator controls is defined in the "Properties > Properties > Toolbar" inspector window.

Note

Basic Panels

Because archiving with Basic Panels is only possible for devices as of the 2nd Generation, the operator controls are not available on all Basic Panels.

Toolbar button	Brief description	Description
	"Go to start"	Scroll to the current value of the trend.
	"Zoom in"	Reduces the displayed time slice.
	"Zoom out"	Zooms into the displayed time section.
	"Ruler backward"	Moves the ruler back.
	"Ruler forward"	Moves the ruler forward.
	"Backward"	Scrolls back one display width.
	"Forward"	Scrolls forward one display width.
	"Ruler"	Shows or hides the ruler. The ruler displays the X-value associated with a Y-value.
	"Start/stop"	Stops trend recording or continues trend recording.

Operation on Basic Panels and devices with small display size

The trend view buttons are not displayed on Basic Panels and HMI devices with a display size of less than 6". You can operate the trend view using the function keys of the HMI device that are assigned corresponding system functions.

Displaying column headers

The layout of the table in the trend view depends on the view settings in the Control Panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Appearance" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

Consistency test

If warnings or errors are displayed in the output window during a consistency check in connection with trend views, clicking "Go to Error/Tag" on the shortcut menu will not always take you to the exact error position. In some cases only the trend view is shown as cause of error.

Adding, configuring, and removing trends

The trends of the trend view are managed in the Inspector window under "Properties > Properties > Trend". You can copy trends between different trend views.

See also

Trends (Page 678)

Configuring trend displays for values from the PLC (Page 681)

Configure trend view (Page 699)

Creating trends for the trend window. (Page 703)

Device dependency of the objects (Page 326)

1.9.2.40 Charge condition (Panels, Comfort Panels)

Application

The object "Charging condition" is a control element for some SIMATIC Mobile Panels. The "Charging condition" object indicates the charging status of the battery. Charge the battery in good time. Alternatively, you can replace the battery with a spare battery.

Layout

Symbol	Description	Charging condition
	Battery is charged	>20 %
	Battery must be charged	<20 % and >6 %
	Battery is weak	< 6 %

Operation

The object is for display only and cannot be controlled by the operator.

See also

Effective range name (Page 507)

Effective range name (RFID) (Page 509)

Effective range signal (Page 511)

WLAN reception (Page 512)

Zone name (Page 516)

Zone signal (Page 517)

Device dependency of the objects (Page 326)

1.9.2.41 Illuminated pushbutton (Panels, Comfort Panels)

Application

The object "Illuminated pushbutton" is a control element for some SIMATIC Mobile Panels. The LEDs can be activated by the PLC. The illuminated LED signals the operator in Runtime that the illuminated pushbutton has to be pressed. The illuminated pushbutton is configured in the global screen.

Layout

You can adapt the following properties in the Inspector window:

- Tag: Specifies the tag in which the status of the illuminated pushbutton is written.
- LED assignment: Specifies the assignment of the LED to the bit in the LED tag.

Tag

The "Tag" property is used to specify the global assignment of the "Illuminated pushbutton" object in the global screen.

Note

Notes on configuring

The current status of the illuminated pushbutton is written to the tag when Runtime is started and whenever the button is pressed. This may cause inconsistencies between the status of the illuminated pushbutton and the illuminated pushbutton tags.

1. If the tag has a connection to the PLC, the current value is transferred from the PLC to the tag after establishment of communication. The tag that contains the current status of the key is overwritten by this process. The illuminated pushbutton tag may therefore no longer reflect the current value of the illuminated pushbutton, for example if the Mobile Panel is switched off while the illuminated pushbutton is pressed.
2. If the illuminated pushbutton is pushed before communication to the PLC could be established (for example after a device start-up), it may not be possible to send the value of the illuminated pushbutton tag to the PLC. In this case the value of the tag in the PLC is different from the current status of the key.

Carry out the following configuration steps so that the illuminated pushbutton always reflects the actual state.

Basic procedure

1. Specify the connection to the PLC in the "Connections" editor. Activate the "Coordination" area pointer to ensure that the life bit is available to the PLC side.

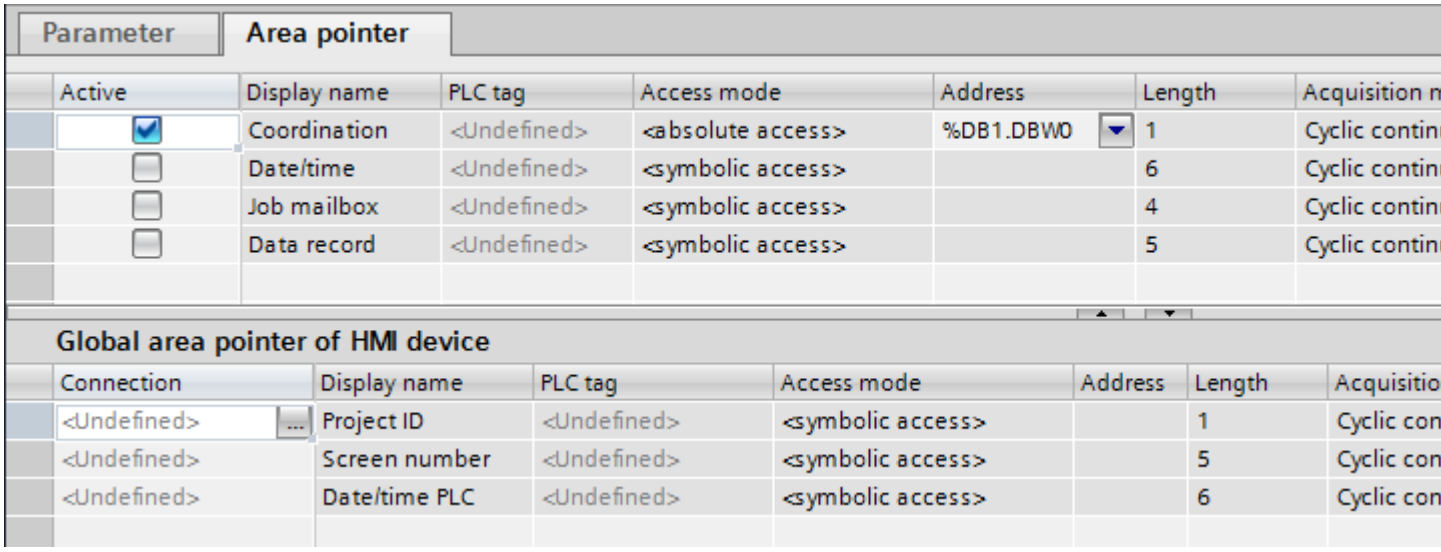

2. Open the "HMI tags" editor and create three tags.
   - Internal tag: Pushbutton_State
   - External tag: BufferTag
   - External tag: Pushbutton_PLC
3. Open the Inspector window of the "Pushbutton_State" tag.
4. In the Inspector window, select "Properties > Events > Value change". Click the first line of the function list. The list opens, showing the system functions available in the project.
5. Select the "SetTag" system function.
   - Select the "Pushbutton_PLC" tag at "Tag (output)".
   - Select the "Pushbutton_State" tag at "Value".

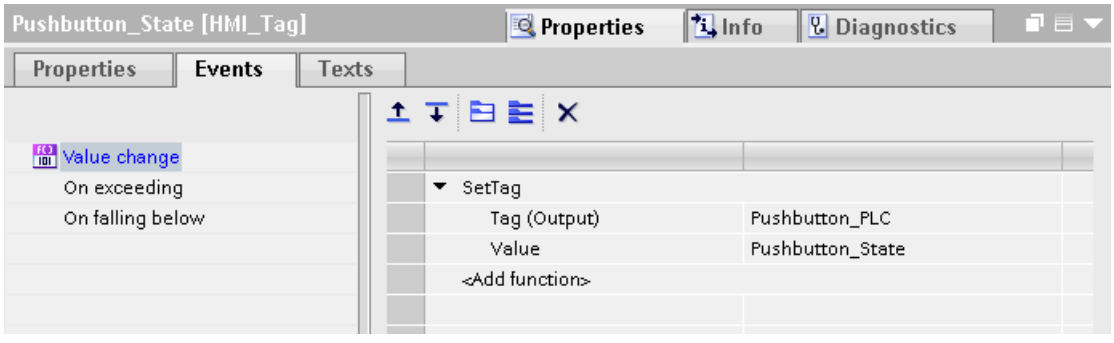

The value of the "Pushbutton_PLC" tag is written to the PLC with the "Pushbutton_PLC" tag. A program in the PLC evaluates the life bit. If communication is established, the current value is written to the "Pushbutton_PLC" tag from the PLC. The "BufferTag" is required to transfer the current position of the pushbutton to the PLC.

6. Open the Inspector window of the "BufferTag" tag.
7. In the Inspector window, select "Properties > Events > Value change".
8. Click the first line of the function list. The list opens, showing the system functions available in the project.
9. Select the "SetTag" system function.
   - Select the "Pushbutton_PLC" tag at "Tag (output)".
   - Select the "Pushbutton_State" tag at "Value".

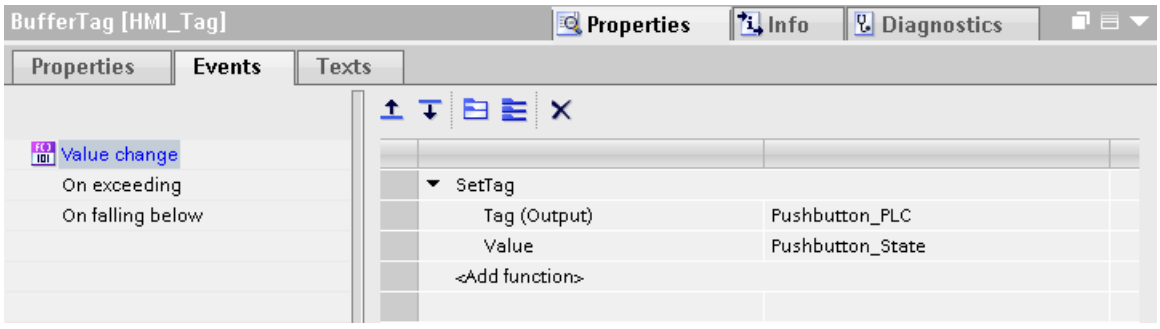

After establishing communication the current value is written to the "BufferTag" from the PLC. The "SetTag" system function is executed by the value change in the auxiliary tag. The system function re-allocates the value of the "Pushbutton_State" tag to the "Pushbutton_PLC" tag.

10. Open the global screen in the "Screen navigation" editor.
11. Select the illuminated key on the global screen.
12. Select the tag "Pushbutton_State" in "Properties > Properties > General > Settings >Tag" in the Inspector window.
    If you press the illuminated pushbutton, the value is written to the "Pushbutton_PLC" tag.

LED assignment

To actuate the LED an LED tag or an array tag must be set up in the PLC and specified as an LED tag during configuration. The LED tag is specified at "LED tag" in the "Settings" area in the "General" group. With "LED bit" the allocated bit is given. When the bit is set the following states of the LED can be realized in runtime.

Bit n+1	Bit n	LED function
0	0	Off
0	1	Fast flash
1	0	Slow flash
1	1	On permanently

See also

Handwheel (Page 386)

Device dependency of the objects (Page 326)

1.9.2.42 Line

Application

The "Line" object is an open object. The line length and gradient slope are defined by the height and width of the rectangle enclosing the object.

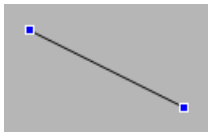

Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Line style
- Line start and end

Line style

The representation of the line is specified under "Properties > Properties > Appearance" in the Inspector window. The line is shown without interruption if you select "Solid", for example.

Note

The line styles available depend on the selected HMI device.

Line start and end

The start and end points of the line are specified under "Properties > Properties > Appearance > Line ends" in the Inspector window.

Use arrow point, for example, as start and end point. The available start and end points depend on the device.

See also

Device dependency of the objects (Page 326)

1.9.2.43 List box (RT Professional)

Application

You use the "List box" object to present and select multiple list entries. You activate list entries by default so that the operator only changes the preset entry if necessary. If the list box is larger than the selection rectangle, WinCC automatically adds a scroll bar to the right margin.

To incorporate list fields into the process, dynamize the corresponding properties.

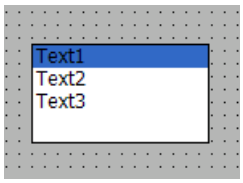

Layout

In the Inspector window, you customize the position, style, colors and font type settings of the object. You can adapt the following properties in particular:

- Number of entries: Defines the number of list entries.
- Selection of entries: Defines which entry is displayed as activated by default.

Defining the number of entries

1. In the Inspector window, select "Properties > Properties > General".
2. Define the desired number of list entries under "Entries".

At the same time, the value of the "Entries" attribute specifies the upper limit for the "Index" property under "Label." Changing the value can have the following effects:

Changing the number

New fields are added under the field with the highest value for the "Index" property if you increase the number of entries. You can change the default label for the new field using the "Text" property under "Label."

All fields for which the value of the "Index" property is higher than the new number will be deleted if you reduce the number of entries.

Specifying the default setting of the list boxes

Specify which list entry is displayed as active with "Properties > Properties > General > Presetting enabled".

Each option is represented by a bit in a 32-bit word. To activate a field, the corresponding bit must have the value "1". The 32-bit word contains the information for all texts of the list of list boxes. The value of the "Selected fields" property is given in hexadecimal notation.

See also

Combo box (Page 400)
Configure operator input alarms (Page 827)
Device dependency of the objects (Page 326)

1.9.2.44 Media Player (RT Professional, Panels, Comfort Panels)

Use

In Runtime, the Media Player is used to play multimedia files.

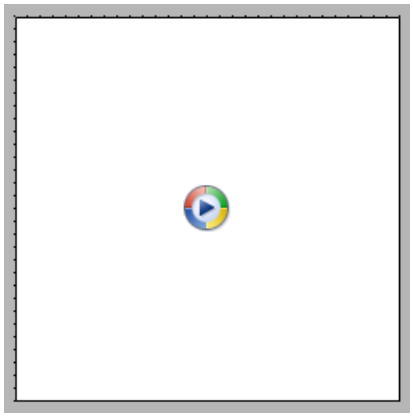

The Media Player is only available for the following devices:

- Comfort Panels
- RT Professional

Note

Playing videos on Comfort Panels

You can play video sequences on Comfort Panels in the "Media Player" screen object. You can find more detailed information on playing videos on the Internet under: http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/62101921) (entry ID 62101921)

Layout

You can set the following properties in the Inspector window:

- Display status bar: Determines whether to display the status bar.
- Display operator controls: specifies the operator controls in Runtime.
- Show tracker: establishes, whether a slider is available for the operation.

Operator controls

The operator controls that can be used to control the Media Player in Runtime are specified in the Inspector window under "Properties > Properties > Appearance > Elements".

Number of replays (Comfort Panels)

To play the file multiple times, select number of playbacks under "Properties > Properties > General > Number of replays". Select a value from 0 to 2,147,483,647.

Supported file formats (Comfort Panels)

The following audio formats are supported by Media Player:

- Moving Picture Experts Group standard 1, Layer 1, 2, 3 (.mpa, .mp2, .mp3)
- Waveform Audio (.wav)

The following video formats are supported by Media Player:

- Moving Picture Experts Group standard 1 (.mpg, .mpeg, .mpv, .mpe)
- Advanced Streaming Format (.asf)
- Windows Media Video Format (.wmv)
- Audio-Video Interleaved (.avi)

Note

Media Player does not support the creation and management of favorites. If you create favorites before closing the Media Player, they will not reappear when you open it again.

Note

The Media Player cannot be operated using the keyboard.

You cannot simulate the Media Player. A static screen appears in the simulation window instead of the Media Player.

Supported file formats (RT Professional)

The following file formats are supported by Media Player:

- graphics file formats (.gif, .bmp, .jpg, .jpeg, .png)
- Moving Picture Experts Group standard 1 (.mpg, .mpeg, .mp4)
- QuickTime Movie (.mov, .qt)
- Advanced Streaming Format (.asf)
- Windows Media Video Format (.wmv)
- Audio Video Interleave (.avi)

Note

Requirements for video files

To play video files in the Windows Server 2008 R2 SP1 and 2012 R2, install the Microsoft feature "Desktop Experience". You will find more detailed information on this topic on the Internet in the Microsoft documentation.

Note

Playing back multimedia files in the control depends on the video and audio codecs installed on the PC, as well as on the file format.

Note

Data loss when copying the project

If you copy the project to another PC, keep the following in mind:

Files indicated in the WinCC Media Control are not copied along with the other files if they are dynamically linked and no UNC path is specified. You have to load the files into the project again.

See also

Device dependency of the objects (Page 326)

1.9.2.45 Alarm view (RT Professional)

Use

The "Alarm view" object displays alarms that occur during the process in a plant. You also use the alarm view to visualize alarms in list format. WinCC offers various views, such as "Current alarms" or "Historical alarm list" (short-term).

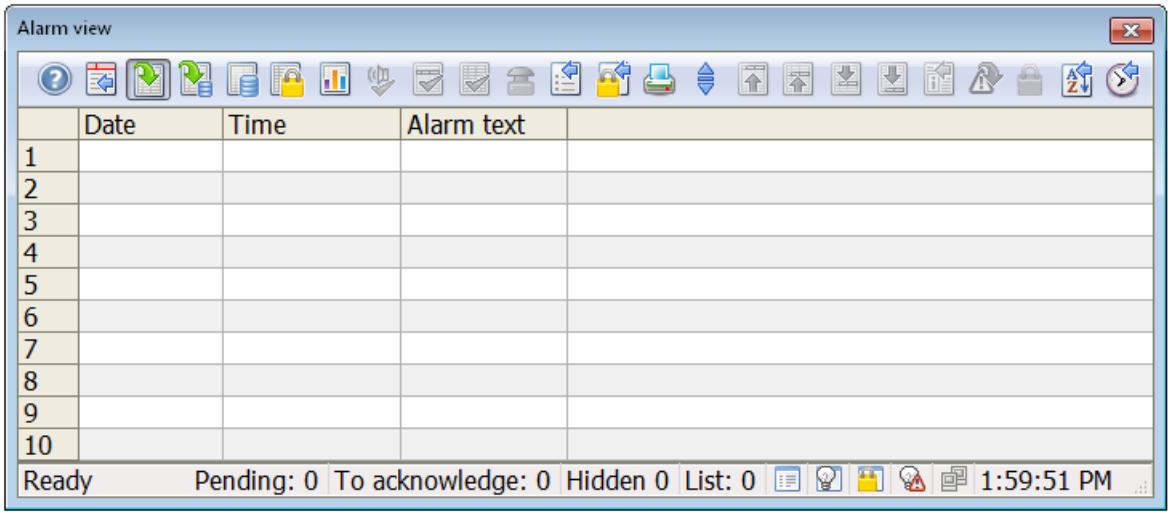

Layout

You change the settings for the position, geometry, style, color, and font of the object in the Inspector window. You can adapt the following properties in particular:

- Operator controls: Defines the operator controls of the alarm view.
- Status bar: Defines the elements in the status bar.
- Time base: Defines the time basis used to output the alarms.
- AutoScroll: Defines whether it is always the latest alarm that is selected in the alarm view.
- Text format: Defines the font and font size.
- Sorting: Defines whether and how the alarms can be sorted in Runtime.
- Table: Defines the properties of the table in the alarm view.
- Loop-In-Alarm: Specifies whether double-clicking an alarm triggers the configured screen change.

Note

The "alarm view" object cannot be grouped.

You only configure a script at the "Incoming" event

Configuring output of alarms

- Selection of displays and filters: Defines which alarms are displayed in this alarm view.
- Alarm statistics: Activates the display of a statistical overview for the display duration and frequency of the alarms.
- Print: This setting defines which print job is used for the alarm view.

Operator controls

You define the control elements of the alarm view in Runtime, and their operator authorizations, under "Properties > Properties > Toolbar" in the inspector window. The following control elements are available for the alarm view:

Button		Function
	Display "Current alarms"	Shows the currently active alarms.
	Display "Historical alarm list" (short-term)	Shows the logged and currently active alarms.
	Display "Historical alarm list" (long-term)	Shows the logged alarms.
	Display "Locked alarms"	Shows all locked alarms in the system.
	Display "Alarm statistics"	Displays a statistical overview for the display duration and frequency of the alarms.
	"Alarm annunciator acknowledgment"	Acknowledges a visual, or audible signaling device.
	"Single acknowledgment"	Acknowledges an individual alarm.
	"Group acknowledgment"	Acknowledges all active visible alarms in the alarm view that require acknowledgment, unless they are subject to single acknowledgment.
	"Autoscroll"	Defines whether it is always the latest alarm that is selected in the alarm view. The visible area of the alarm view moves, if necessary. You can only select individual alarms if "Autoscroll" is not active.
	"Filter"	Opens a dialog for filtering alarms in the alarm view. Even alarms that are not displayed in the filtered view will be logged.
	"Display Options" dialog	Opens a dialog for defining which alarms are displayed in the alarm view. If the "All alarms" option is selected, the alarm view displays both hidden, and visible alarms. If the "Visible alarms" option is activated, the alarm view contains only visible alarms. If the "Hidden alarms" option is activated, the alarm view contains only hidden alarms.

Button		Function
	"Lock dialog"	Opens a dialog for locking alarms. Any alarms that meet these criteria are neither displayed, nor logged.
	"Print alarm report"	Prints the alarm report that is configured for the alarm view.
	"Print current view"	Starts printing the alarms displayed in the alarm view.
	"Emergency acknowledg-ment"	Applies emergency acknowledgment to an alarm that requires acknowledgment. The acknowledgment bit for an alarm is sent directly to the AS, even if the alarm is not active.
	"First alarm"	Selects the first of the active alarms. The visible area of the alarm view moves, if necessary. This button can only be used if the "Autoscroll" function is disabled.
	"Last alarm"	Selects the last of the active alarms. The visible area of the alarm view moves, if necessary. This button can only be used if the "Autoscroll" function is disabled.
	"Next alarm"	Selects the next alarm in relation to the currently selected alarm. The visible area of the alarm view moves, if necessary. This button can only be used if the "Autoscroll" function is disabled.
	"Previous alarm"	Selects the previous alarm in relation to the currently selected alarm. The visible area of the alarm view moves, if necessary. This button can only be used if the "Autoscroll" function is disabled.
	"Infotext dialog"	Opens a dialog to display an infotext.
	"Comment dialog"	Opens a text editor where you can enter comments. This button is not available in the alarm view unless the "Historical alarm list (long-term)" display is displayed.
	"Loop-In-Alarm"	If a screen change is configured, the display switches to a screen containing information about the alarm.
	"Unlock alarm"	Unlocks an alarm in the display "Locked alarms."
	"Lock alarm"	Locks an alarm in the current alarm list, and in the alarm log lists. The alarm is added to the display "Locked alarms."
	"Sort dialog"	Opens a dialog for setting user-defined sort criteria for the displayed alarms.
	"Time base dialog"	Opens a dialog for selecting the time base for the time data displayed in alarms.
	"List of hidden alarms"	Opens the display "Hidden alarms."
	"Hide alarms"	Suppresses the display of an alarm in the displays "Current alarms", "Historical alarm list (short-term)" or "Historical alarm list (long-term)." The alarm is added to the display "Hidden alarms."
	'Show alarm"	Shows in the display "Hidden alarms" again an alarm in the displays "Current alarms", "Historical alarm list (short-term)" or "Historical alarm list (long-term)." The alarm is removed from to the display "Hidden alarms."

Status bar

You define which of the elements of the status bar are displayed using "Properties > Properties > Status bar" from the Inspector window. The following display elements are available for the alarm view:

Button	Function
Date	System date
Time of day	System time
	Lock is set.
	A filter is activated.
List:	Number of current alarms
Window:	Number of alarms in the window
Ackn:	Number of active alarms requiring acknowledgment

Access protection in Runtime

Configure access protection under "Properties > Properties > Security" in the Inspector window. If a logged-on user has the required authorization, he can acknowledge, and edit alarms using the control elements in the alarm view.

The logon dialog box is displayed in the following cases:

- The logged-on user attempts to process the alarm but does not have the required authorization.
- A user attempts to process an alarm while no user is logged on.

Set timebase

1. In the Inspector window, select "Properties > Properties > General".
2. Select the required timebase under "Properties > Properties > Miscellaneous > Time base."
3. Under "Time-based standard sort", define whether the alarms are sorted in ascending or descending order in Runtime.

Enable Autoscroll

1. In the Inspector window, activate "Properties > Properties > Layout > Window > Autoscroll".

Define text format

To format the alarm text:

1. In the Inspector window, select "Properties > Properties > Text format".
2. Click the "..." button in the "Font" area.
   A dialog opens.
3. Select the font and size in the dialog.

Set up sort order

1. In the Inspector window, select "Properties > Properties > Columns".

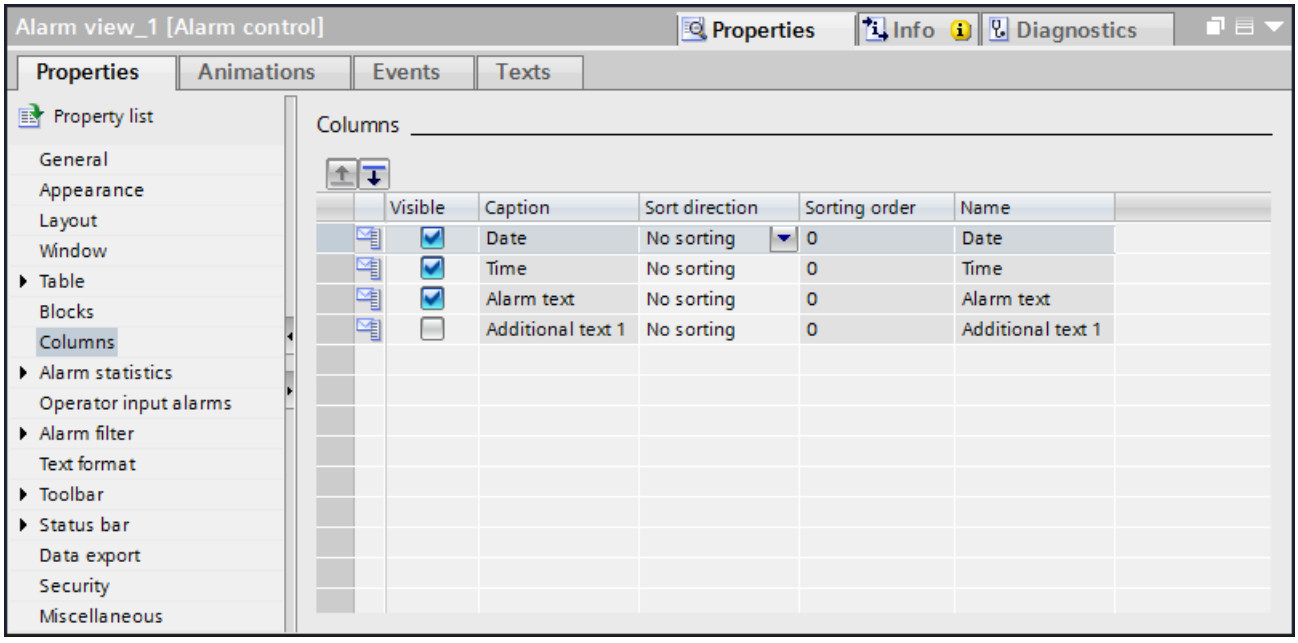

2. Select sort criteria and sorting order.

Set to a table in the alarm view

1. Click "Properties > Properties > Table font" in the Inspector window.
2. Define settings for the rows and cells in the "Table body" area.
   - "Optimum line height" activated: The font is aligned to the height of the line.
   - "Truncate cell contents": If the text is longer than the cell, the text is displayed truncated.
3. In the "Line numbering" area, define whether each row has a header.
4. Select the column header layout from the "Column headers" area.

Set Loop-In-Alarm

1. In the Inspector window, click "Properties > Properties > Miscellaneous".
2. Activate the "Loop-In-Alarm with double-click" option.

Displaying column headers

The layout of the alarm view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is available in the Control Panel, "Display" > "Appearance" tab. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

See also

Configuring an alarm view for active alarms (Page 837)

Configuring persistence (Page 2264)

Device dependency of the objects (Page 326)

1.9.2.46 Alarm view (as of V13) (RT Advanced, Basic Panels, Panels, Comfort Panels)

Use

Advanced alarm view is used to display alarms on the HMI device.

On HMI devices with device version V13 or later, the advanced table-based alarm view is available for managing alarms.

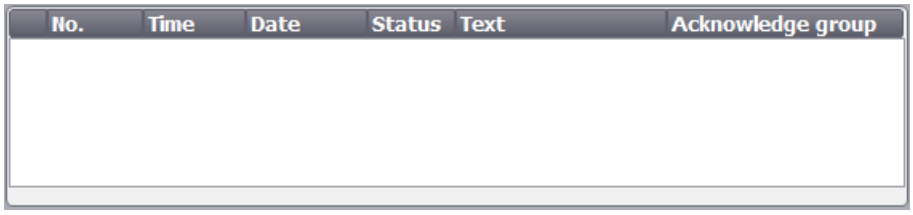

Alarm line

The alarm line allows display of the most current pending alarm in Runtime. The following figure shows an alarm line:

How to configure the alarm line:

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select "Mode > Alarm line".

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object.

Note

The fonts available for selection depend on the fonts you have configured in the Runtime settings under "Language & font" and the fonts supported by your HMI device.

You can adapt the following properties in particular:

- Toolbar: Defines the operator controls of the alarm view.
- Alarm classes: This setting defines which alarm classes are displayed in the alarm view.

- Columns: Specifies the displayed columns in Runtime.
- Columns moveable: Specifies whether the sequence of columns can be changed in Runtime.
- Identification of alarm classes: To be able to distinguish various alarm classes, they are identified in the first column of the alarm view.
- Filter: Defines that only alarms which contain a specific character string in the alarm text will be displayed.
- Define display area: This setting specifies the date from which alarms are displayed in the alarm view.
- Sorting by date/time possible: Specifies whether the alarms can be sorted in Runtime by date and time.

Note

If you have different alarm classes output, these will be initially sorted into alarm classes in Runtime, and then by the time at which the alarm occurred.

Operator controls

The operator controls that can be used to control the alarm view in Runtime are specified in the Inspector window under "Properties > Properties > Toolbar". The following table shows the operator controls in the alarm view and their function:

Button	Name	Function
	"Info text"	Displays a tooltip for an alarm.
	"Acknowledge"	Acknowledges an alarm.
	"Loop-in-alarm"	Executes the configured function (for example: "ActivateScreen") with each keystroke. It is hereby possible to call a screen containing your information about the selected alarm. Also acknowledges an alarm requiring acknowledgment. [1]

[1] Only if a function is configured for the selected alarm for the "loop-in-alarm" event.

Select alarm classes

1. Click "Properties > Properties > General" in the Inspector window.
2. Under "Alarm classes", activate the alarm classes to be displayed in the alarm view in Runtime.

Access protection in Runtime

You can configure access protection in the "Properties > Security" group in the properties of the alarm view. A logged-on user with the required authorization can acknowledge and edit alarms using the operator controls in the alarm view. If the logged-on user does not have the required authorization, or if no user is logged in, pressing the "Acknowledge" or "Edit" buttons or double-clicking an alarm line opens the logon dialog.

Define columns

Define the columns to be displayed in the alarm view in Runtime in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Activate the columns that are to be displayed in Runtime under "Visible columns".

Sequence of columns

If this property is enabled, the column sequence in the alarm view can be changed in Runtime.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Column properties > Columns moveable".

Sorting

If this property is enabled, the alarms displayed in the alarm view in Runtime can be sorted by date and time.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Column properties > Sorting by date/time possible".

Filter alarms

This property is used to define that only alarms that contain a configured string in the alarm text will be displayed in Runtime for the extended alarm view.

1. In the Inspector window, select "Properties > Properties > Filter".
2. In the "Filter string" field, enter the desired term for filtering.
   Alternatively, you can configure a filter tag using the "Filter tag" field. The contents of the filter tag serve as the filter criterion in Runtime.

Identify alarm classes

An icon is displayed in the first column of the alarm view. This symbol allows the alarm to be allocated to an alarm class.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Visible columns > Alarm class name".

3. Open the "HMI alarms" editor and click on the "Alarm classes" tab.
4. Specify an icon for use to identify the alarms of this alarm class for an alarm class in the "Display name" column.

Note

The "alarm view" object cannot be grouped.

Set colors for alarm classes

You have the option of having alarms of selected alarm classes shown in different colors in Runtime. You set the background colors of the alarms in the "HMI alarms" editor.

1. Open the "HMI alarms" editor and click on the "Alarm classes" tab.
2. Set the colors for alarms under "Properties > Properties > Colors".
3. Depending on the HMI device, also change the flashing characteristics if required.
4. To show alarms in the alarm view in the defined colors, activate the option "Alarm class colors" under "Runtime settings > Alarms > General".

Define display area

You select a tag which defines the time from which alarms should be displayed. The following data types are permitted:

- External tags: Date, Date_and_Time, Time_of_Day
- Internal tags: DateTime

To define the display area, proceed as follows:

1. Click "Properties > Properties > Display" in the Inspector window.
2. Define the tag in which the time is specified under "Control tag for display area".

Switch to PLC code view

You configure the "ActivatePLCCodeView" system function on a button to enable the jump from a monitoring alarm in the display to the affected program code in the PLC code view. In the alarm view, you define a control tag that you use to give the button dynamic properties. Control tag is used to evaluate whether the entry point of the selected monitoring signal is possible in the PLC code view. To define the control tag, proceed as follows:

1. Click "Properties > Properties > Display" in the Inspector window.
2. Specify a Boolean tag under "Properties > Display > Control tag for PLC code view" that will control the jump from the last active ProDiag alarm or GRAPH alarm to the PLC code view.

You have the option make the button dynamic using animations, for example, so that the button changes color with an incoming alarm.

Displaying column headers

The layout of the alarm view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Layout tab" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

1.9.2.47 Alarm view, simple (RT Advanced, Basic Panels, Panels, Comfort Panels)

Use

Alarms are indicated in the Alarm view or in the Alarm window of the HMI device.

The following screen contains a simple alarm view:

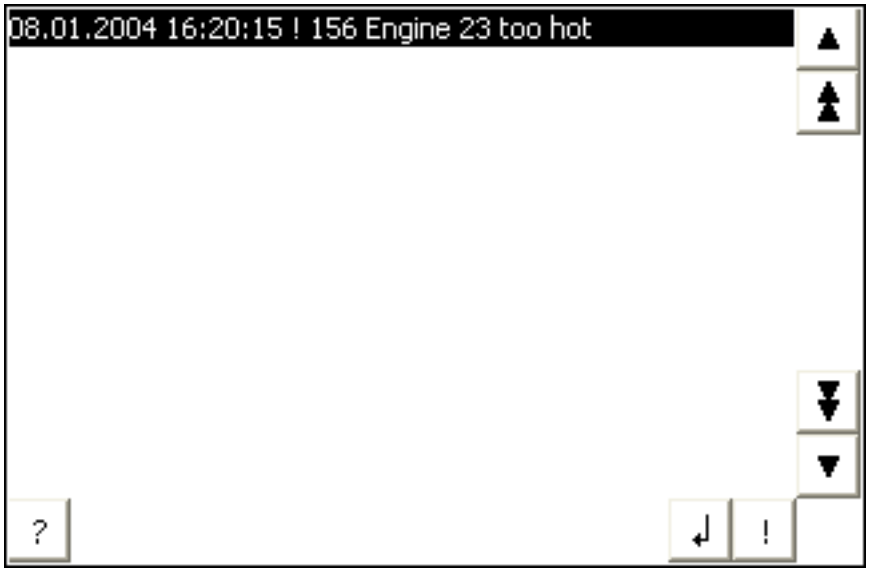

Note

The "Single alarm view" object cannot be operated dynamically with a script.

Note

Configure the "Simple alarm view" object on devices with device version 12.0.0.0 or older.

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object.

Note

The fonts available for selection depend on the fonts you have configured in the Runtime settings under "Language & font" and the fonts supported by your HMI device.

You can adapt the following properties in particular:

- Operator controls: Defines the operator controls of the alarm view.
- Alarm classes: This setting defines which alarm classes are displayed in the alarm view.
- Columns: Specifies the displayed columns in Runtime.

  Note

  If you have different alarm classes output, these will be initially sorted into alarm classes in Runtime, and then by the time at which the alarm occurred.

Operator controls

The operator controls that can be used to control the alarm display in Runtime are specified in the Inspector window under "Display > Settings". The following table shows the operator controls in the alarm view, and what they do:

Button		Function
"Info text"	?	Displays info text for an alarm.
"Acknowledge"	!	Acknowledges an alarm.
"Loop-In-Alarm"	↲	Executes the configured function (for example: "ActivateScreen") with each keystroke. It is hereby possible to call a screen containing your information about the selected alarm. Also acknowledges an alarm requiring acknowledgment. [1]

[1] Only if a function is configured for the selected alarm for the "loop-in-alarm" event.

Select alarm classes

1. Click "Properties" in the Inspector window.
2. Under "Alarm classes", activate the alarm classes to be displayed in the alarm view in Runtime.

Define columns

Define the columns to be displayed in the alarm view in Runtime in the Inspector window.

1. In the Inspector window, click "Properties > Columns".
2. Activate the columns that are to be displayed in Runtime under "Columns".

Displaying column headers

The layout of the alarm view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Layout tab" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

Note

In the engineering system, you can dynamically control the visibility of an object, for example, in the "Animations" group of the Inspector window. In Runtime, the "Simple alarm view" does not support animations. If you have configured an animation and, for example, wish to perform a consistency check of the project, then an error alarm is issued in the Output window.

1.9.2.48 Alarm view, advanced (RT Advanced, Panels, Comfort Panels)

Alarm view

Alarms are indicated in the Alarm view or in the Alarm window of the HMI device. The following screen contains an alarm view with no content:

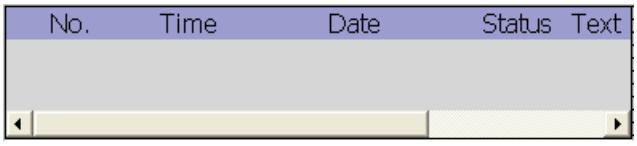

Alarm line

The alarm line allows display of the most current pending alarm in Runtime. The following figure shows an alarm line:

How to configure the alarm line:

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select "Mode > Alarm line".

Changing the view type

On HMI devices with a display size larger than 6 ", you can configure both the advanced alarm view and a simplified alarm view. The view type is only available for configuration on devices up to device version V13.

Proceed as follows to configure the simplified alarm view:

1. Click "Properties > Properties > Layout > Mode" in the Inspector window.
2. Select "Simplified".

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object.

Note

The fonts available for selection depend on the "Language and fonts" you have configured in the device settings and the fonts your HMI device supports.

You can adapt the following properties in particular:

- Operator controls: Defines the operator controls of the alarm view.
- Alarm classes: This setting defines which alarm classes are displayed in the alarm view.
- Columns: Specifies the displayed columns in Runtime.
- Sequence of columns: Specifies whether the sequence of columns can be changed in Runtime.
- Identification of alarm classes: To be able to distinguish various alarm classes, they are identified in the first column of the alarm view.
- Filter: Defines that only alarms that contain a specific character string in the alarm text will be displayed.

- Enable sorting: Specifies whether the alarms can be sorted in Runtime by date and time.
- Define display area: This setting specifies the date from which alarms are displayed in the alarm view.

Note

If you have different alarm classes output, these will be initially sorted into alarm classes in Runtime, and then by the time at which the alarm occurred.

Operator controls

The operator controls that can be used to control the alarm view in Runtime are specified in the Inspector window under "Properties > Properties > Display > Settings". The following table shows the operator controls in the alarm view, and what they do:

Button		Function
"Tooltip"	?	Displays a tooltip for an alarm.
"Acknowledge"	!	Acknowledges an alarm.
"Loop-In-Alarm"	↵	Executes the configured function (e.g.: "ActivateScreen") whenever a key is pressed. This enables a screen that contains your information on the selected alarm to be called. Also acknowledges an alarm that requires acknowledgment. [1]

[1] Only if a function is configured for the selected alarm for the event "Loop-In-Alarm".

Select alarm classes

1. Click "Properties > Properties > General" in the Inspector window.
2. Under "Alarm classes", activate the alarm classes to be displayed in the alarm view in Runtime.

Access protection in Runtime

Configure access protection in the "Properties > Security" group in the properties of the alarm view. If a logged-on user has the required authorization, he can acknowledge and edit alarms using the operator controls in the alarm view. If the logged-on user does not have the required authorization, or if no user is logged in, pressing the "Acknowledge" or "Edit" buttons or double-clicking an alarm line opens the logon dialog.

Define columns

Define the columns to be displayed in the alarm view in Runtime in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Activate the columns that are to be displayed in Runtime under "Columns".

Sequence of columns

If this property is enabled, the column sequence in the alarm view can be changed in Runtime.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Column properties > Column order".

Enable sorting

If this property is enabled, the alarms displayed in the alarm view in Runtime can be sorted by date and time.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Column properties > Sorting by date/time possible".

Filter alarms

This property is used to define that only alarms that contain a configured string in the alarm text will be displayed in Runtime for the extended alarm view.

1. In the Inspector window, select "Properties > Properties > Filter".
2. Enter the required term for filtering in the field "Filter string".
   Alternatively, you configure a filter tag using the field "Filter tag". The contents of the filter tag serve as the filter criterion in Runtime.

Identify alarm classes

An icon is displayed in the first column of the alarm view. This symbol allows the alarm to be allocated to an alarm class.

1. In the Inspector window, select "Properties > Properties > Columns".
2. Select "Columns > Alarm class".
3. Open the "Alarms" editor, and click the "Alarm classes" tab.
4. Specify an icon for use to identify the alarms of this alarm class for an alarm class in the "Display name" column.

   Note

   The "alarm view" object cannot be grouped.

Specifying colors for alarm classes

You have the option to display alarms of the selected alarm classes in different colors in Runtime. You specify the background colors of the alarms in the "HMI alarms" editor.

1. Open the "HMI alarms" editor, and click the "Alarm classes" tab.
2. Define the colors for alarms under "Properties > Properties > Colors".
3. To show alarms in the alarm view in the defined colors, activate the option "Alarm class colors" under "Runtime settings > Alarms > General".

Define display area

You select a tag which defines the time from which alarms should be displayed. The following data types are permitted:

- External tags: Date, Date_and_Time, Time_of_Day
- Internal tags: DateTime

To define the display area, proceed as follows:

1. Click "Properties > Properties > Display" in the Inspector window.
2. Define the tag in which the time is specified under "Control tag for the display area".

Configuration behavior

Displaying column headers

The layout of the alarm view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Layout tab" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

Displaying the buttons for operation

The display of the buttons for using the alarm view depends on the configured size. You should check on the HMI device whether all necessary buttons are available.

See also

Alarm window (Page 435)

Alarm indicator (Page 438)

Configuring an alarm view for active alarms (Page 837)

Device dependency of the objects (Page 326)

1.9.2.49 Alarm window (RT Advanced, Basic Panels, Panels, Comfort Panels)

Use

Alarms are indicated in the Alarm view or in the Alarm window of the HMI device. The layout and operation of the Alarm window are similar to that of the Alarm view. The Alarm window has the following characteristics that are the same as in the Alarm view:

- Alarm window
- Alarm line: The alarm line is not available on Basic Panels.

The Alarm window is configured in the "Global screen" editor.

The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active. Based on the configuration, it is not closed until the alarm is acknowledged.

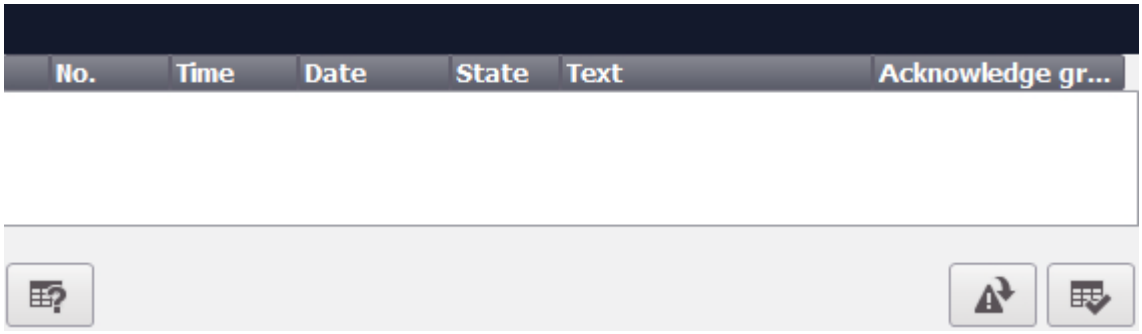

Note

In the engineering system you dynamize, for example, the visibility of an object in "Properties > Animations" in the inspector window. In Runtime, the "Simple alarm window" object does not support animations. If you have configured an animation and, for example, wish to perform a consistency check of the project, then an error alarm is issued in the Output window.

Simple alarm window

The simple alarm window is available on Basic Panels with a device version prior to V13 to display the alarms.

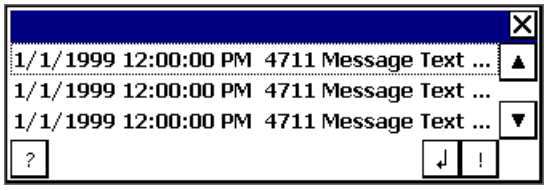

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You configure the Alarm window in the same way as the Alarm view. In addition you adapt the following properties:

- Fixed alarm window: Specifies that the Alarm window retains the focus after a screen change.
- Window: You define the operator input and response of the Alarm window in Runtime.

  Note

  If you have different alarm classes output, these will be initially sorted into alarm classes in Runtime, and then by the time at which the alarm occurred.

Operator controls

The operator controls that can be used to control the alarm view in Runtime are specified in the inspector window under "Properties >Display > Settings". The following table shows the operator controls in the Alarm window, and what they do:

Button		Function
"Tooltip"	?	Displays a tooltip for an alarm.
"Acknowledge"	!	Acknowledges an alarm.
"Loop-In-Alarm"	↵	Executes the configured function (for example: "ActivateScreen") with each keystroke. It is hereby possible to call a screen containing your information about the selected alarm. Also acknowledges an alarm requiring acknowledgment. [1]

[1] Only if a function is configured for the selected alarm for the "loop-in-alarm" event.

Access protection in Runtime

Configure access protection under "Properties > Properties > Security" in the inspector window of the alarm view. If a logged-on user has the required authorization, he can acknowledge and edit alarms using the operator controls in the alarm view. If the logged-in user does not have the required authorization, or if no user is logged in, clicking the "Acknowledge" or "Edit" buttons or double-clicking an alarm line opens the login dialog box.

Note

Basic Panels

Access protection is not available for Basic Panels.

Activating the focus of the Alarm window

Select the following option so that the Alarm window does not lose the focus after a screen change:

1. In the Inspector window, select "Properties > Properties > Mode".
2. Enable "Label".

Window

Define the response of the Alarm window under "Properties > Properties > Mode > Window" in the inspector window. The following table shows the possible properties:

Option	Function
Automatic display	The Alarm window is automatically displayed when a system event occurs, for example.
Closable	The window closes again after a set time has elapsed. You define the display duration in the alarm settings.

Option	Function
Modal	The alarm window is linked to a confirmation, e.g. the alarm must be acknowledged. If the modal alarm window has the focus, the buttons in the screen behind it cannot be used. The functions configured on a function key are carried out.
Sizeable	You can change the size of the Alarm window in Runtime.

See also

Alarm view, advanced (Page 431)

Alarm indicator (Page 438)

Configuring an alarm window (Page 844)

Device dependency of the objects (Page 326)

1.9.2.50 Alarm indicator (RT Advanced, Basic Panels, Panels, Comfort Panels)

Application

The alarm indicator is a graphic symbol that shows current errors or errors that need to be acknowledged, depending on the configuration. The alarm indicator is configured in the "Global screen" editor. The following figure shows an alarm indicator:

Alarm classes

You define which alarm classes are shown with an alarm indicator in "General > Alarm classes" in the Inspector window. Alarm classes, such as "Warnings" or "Errors".

Define operator control in Runtime

1. Select the alarm indicator in the screen.
2. Click "Events > Click" or "Click when flashing" in the Inspector window.
3. The "function list" opens. Click the first line of the function list. The list of system functions, and scripts available in the project opens.
4. Select the "ShowAlarmWindow". system function under "Alarms."
5. Under "object name" select the name of the Alarm window from the selection list. Under "Layout", define whether the Alarm window should be visible, hidden, or should toggle between the two states.

See also

Alarm view, advanced (Page 431)

Alarm window (Page 435)

Configuring an alarm indicator (Page 845)

Device dependency of the objects (Page 326)

1.9.2.51 Option buttons (RT Professional)

Application

You use the "Option button" object to display, and select various options. Only one of these options can be selected respectively by the operator. Enable one of the options by default so that the operator only changes the default value if necessary. To incorporate an option button into the process, dynamize the corresponding attribute.

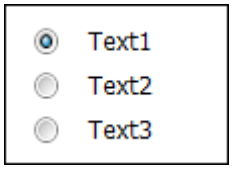

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Number of fields
- Selection of the fields: Specifies which fields are displayed as activated.

Specifying the number of fields

You specify the number of fields in the Inspector window:

1. Click "Properties > Properties > General" in the inspector window.
2. Define the desired number of fields in the area "Label > Number of fields".

The value of the "Number of fields" property also indicates the upper limit value for the "Index" property. Changing the value can have the following effects:

- Increasing the number of fields
  New fields are added under the field with the highest value of the "Index" property. The standard label of the new field is changed with "Font > Text".
- Decreasing the number of fields
  All fields for which the value of the "Index" property is greater than the new number are removed.

Specifying the default setting of the option buttons

The "Default" property is used to specify which of the fields of the option buttons are to be displayed as active. If the default setting for the field is that it is to be activated, activate the "Default" field under "General" in the table row. Only one field may be activated at any time.

See also

Check box (Page 401)

Device dependency of the objects (Page 326)

1.9.2.52 PDF view (RT Advanced, Panels, Comfort Panels)

Use

Using the object "PDF view" you project a PDF view on which the operator can open and display documents in PDF format in runtime. The documents in PDF format are located in the internal or external memory of the respective HMI device in this case.

The PDF view supports PDF versions 1.0 to 1.7. The search function is supported for PDF versions 1.4 to 1.7.

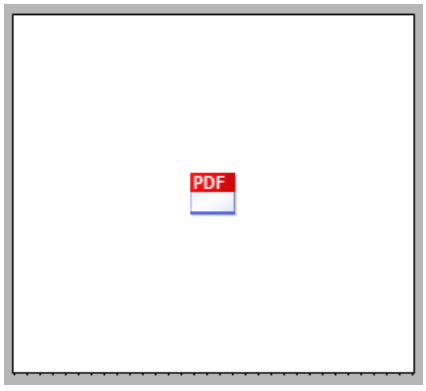

You can configure the PDF view in the following:

- In a screen
- In a template
- In a slide-in screen
- In a pop-up screen

Note

The "PDF view" object may be configured multiple times in a project. However, a maximum of one PDF view may be active at the same time.

Note

Device dependency of the "PDF view" object

The "PDF view" object is only available for KTP Mobile Panels, Comfort Panels and RT Advanced.

Note

Only one "PDF view" object may be inserted per screen.

Layout

You provide information about the position and size of the PDF view in the Inspector window. You can adapt the following properties in particular:

- File name: Specifies the PDF file that opens by default in the PDF view.
- File name tag: Specifies the tag that opens any PDF file.
- Toolbar: Specifies the buttons of the toolbar. If all buttons are disabled, the toolbar is not displayed in Runtime.

Basic procedure

1. Create an internal tag of data type "String" in the "Tag" editor, for example, "PDFFile".
2. Insert the PDF view in the screen of the "Screens" editor. The PDF view is available in the "Tools > Controls" task card.
3. In the Inspector window, select "Properties > Properties > General".
4. Enter the file name of the PDF file that opens by default in runtime in "File name". You can also leave the "File name" field blank.
5. For "File name tag", select the "PDFFile" tag from the selection list.
6. Insert an I/O field in the screen in the "Screens" editor.
7. In the Inspector window of the I/O field "Properties > Properties > General > Tag" select the "PDFFile" tag from the selection list.
8. Assign the PDF view the "PDFFile" tag under "File name tag" in the Inspector window.
9. Create a text field for the label in the configured I/O field, for example, "PDF path".

If the operator enters the file name along with the path in the I/O field, the PDF view opens the file in question.

Toolbar (RT Advanced)

In Runtime Advanced, you can configure a toolbar for the PDF view with which the PDF view is operated in Runtime.

You configure the operator controls that are only visible in the PDF view in Runtime in the engineering system in the Inspector window under "Properties > Properties > Toolbar":

- Search: Specifies whether the search bar and "forward" and "backward" buttons are displayed.
  To search in the displayed document, enter the search term in the search field. The searched text may not contain more than 255 characters. The found text is visually highlighted in the PDF view. You navigate between the search results in the text using the "Forward" and "Backward" buttons. If the found text is located outside of the visible area, the PDF display automatically scrolls to the search result.
  If the sought-after term cannot be found in the document, the search box is highlighted in yellow and the navigation buttons are grayed out. The searched text may not contain more than 255 characters. Line breaks and other formatting characters are ignored in the search.

  Note

  The search for a text that contains special characters or umlauts is not supported in PDF documents with version 1.3 or older.

- Copy: Specifies that the "Select"/"Move" and "Copy" buttons are displayed.
  To select the required section of text, switch on selection mode in the PDF view with the help of the "Select" symbol. Click in front of the first character and drag the cursor over the text to select the text section. The desired section is highlighted in blue. When you have selected the text, click "Copy" to copy the selected text. You can insert the text which is saved in the clipboard in a different object.
  Even if the selected section contains images and other objects, only the text is copied from the selected area to the clipboard.
  Only the text that is currently displayed in the PDF viewer is copied to the Clipboard. Make sure that the desired section of text is fully visible when selecting it.
- Zoom: Specifies that the "Zoom In", "Zoom Out", "Fit to Page Width" and "Fit to Page Height" buttons are shown.
  You adapt the view of the PDF view using the buttons in the zoom bar. The document is opened with the "Adapt to page width" setting by default. If you adapt the view to the page height, the entire page is displayed in the PDF view.

Operator control	Designation	Function
	"Browse"	Searches for a character string within the document. You use the arrows to navigate forwards and backwards between the search results.
	"Select"	Selects the text.
	"Copy"	Copies the selected text.
	"Move"	Moves the view horizontally and vertically.
	"Fit to Page Width"	Adapts the display to the page width.

Operator control	Designation	Function
	"Fit to Page Height"	Adapts the display to the page height.
	"Zoom in"	Enlarges the display.
	"Zoom out"	Reduces the display.

Configuring operator controls

Operator controls are configured buttons in the process picture or softkeys on the HMI device for operating the object.

Configure operator controls yourself, for example, buttons, and link the operator controls with system functions.

1. For example, drag the "Button" object to the screen from the toolbox.
2. Enter a text of any length.
3. In the Inspector window, select "Properties > Events > Click".
   The "Function list" dialog box is opened.
4. The system functions for operation of the PDF view are available in the list of system functions in the group "Keyboard operation for screen objects".
   From the list select a function, for example, "PDFZoomIn".
5. From the "Screen object" list, select the object name of the PDF view in which the command is executed.

System functions

The following system functions are available for the "PDF view" object:

System function	Description
"PDFZoomIn"	Zooms in one zoom level
"PDFZoomOut"	Zooms out one zoom level
"PDFGotoNextPage"	Moves to the next page
"PDFGotoPrevious-Page"	Moves to the previous page
"PDFGotoFirstPage"	Moves to the first page
"PDFGotoLastPage"	Moves to the last page
"PDFFitToWidth"	Fits the display to fit the width of the PDF view window
"PDFFitToHeight"	Fits the display to the height of the PDF view window
"PDFZoomOriginal"	Changes the display to the original size
"PDFScrollUp"	Scrolls up
"PDFScrollDown"	Scrolls down
"PDFScrollLeft"	Scrolls to the left

System function	Description
"PDFScrollRight"	Scrolls to the right
"PDFGotoPage"	Moves to the specified page

Note

Scrolling on all HMI devices also works with multi-touch operation.

The figure below shows a PDF view with the configured operator controls:

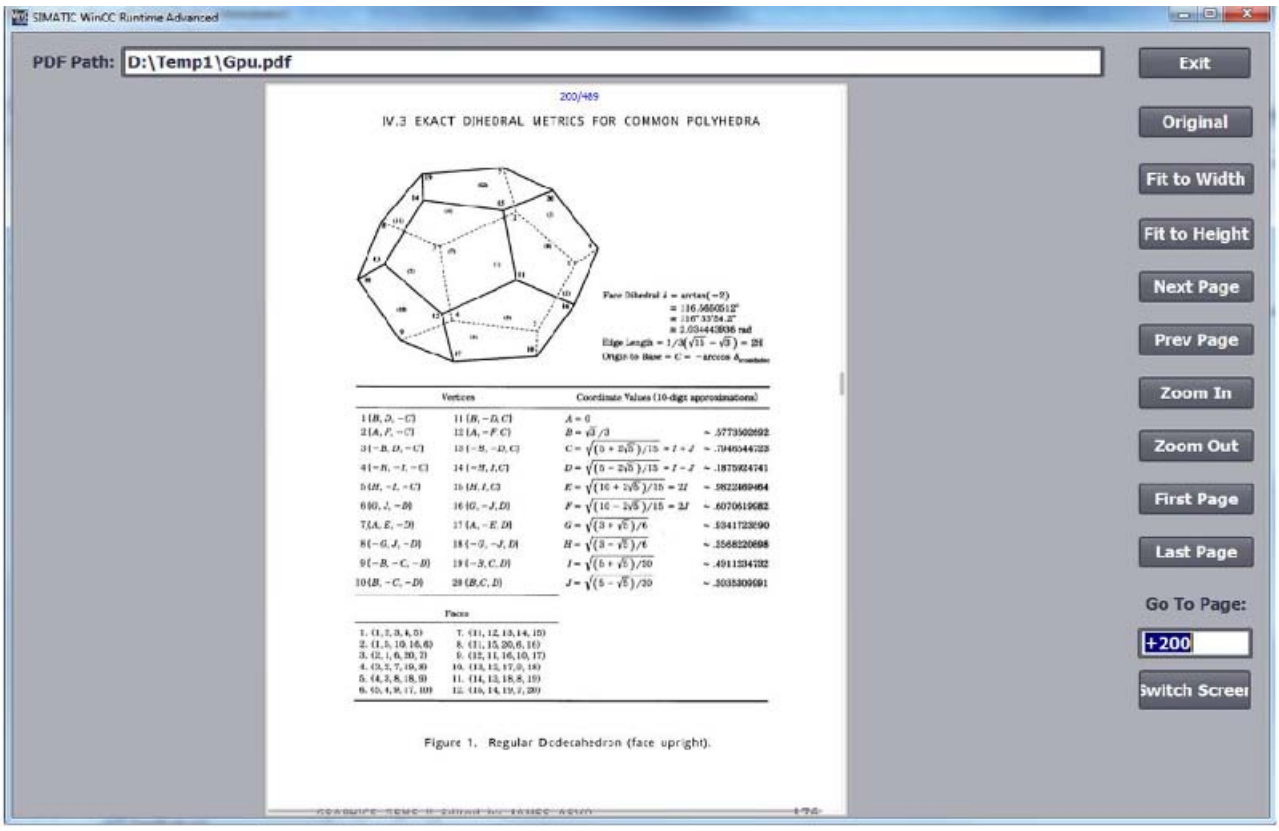

Comments

Compared to Adobe Acrobat, the PDF view has limited functionality:

- The PDF view does not support links.
- The PDF view supports the following zoom levels:
  - Zoom levels 25% to 200% for KTP Mobile Panels, Comfort Panels
  - Zoom levels 25% to 400% for Runtime Advanced

Note

In Runtime Advanced you can scale the view using multi-touch gestures. You zoom into/out from the view using two finger zoom.

Note

The "PDF view" object cannot be addressed by using VB scripting.

Note

A lot of RAM may be needed to display and browse through large PDF files. In such cases, the system event 130000 "Not enough RAM" is displayed.

See also

Objects for WinCC Runtime Advanced (Page 331)

1.9.2.53 PLC code view (RT Advanced, Panels, Comfort Panels)

Use

The "PLC code view" object is used to display the current program status of user programs and security programs (F blocks) which have been programmed in the LAD, FBD or GRAPH programming languages. Errors during execution of a GRAPH sequencer are displayed directly at the corresponding step.

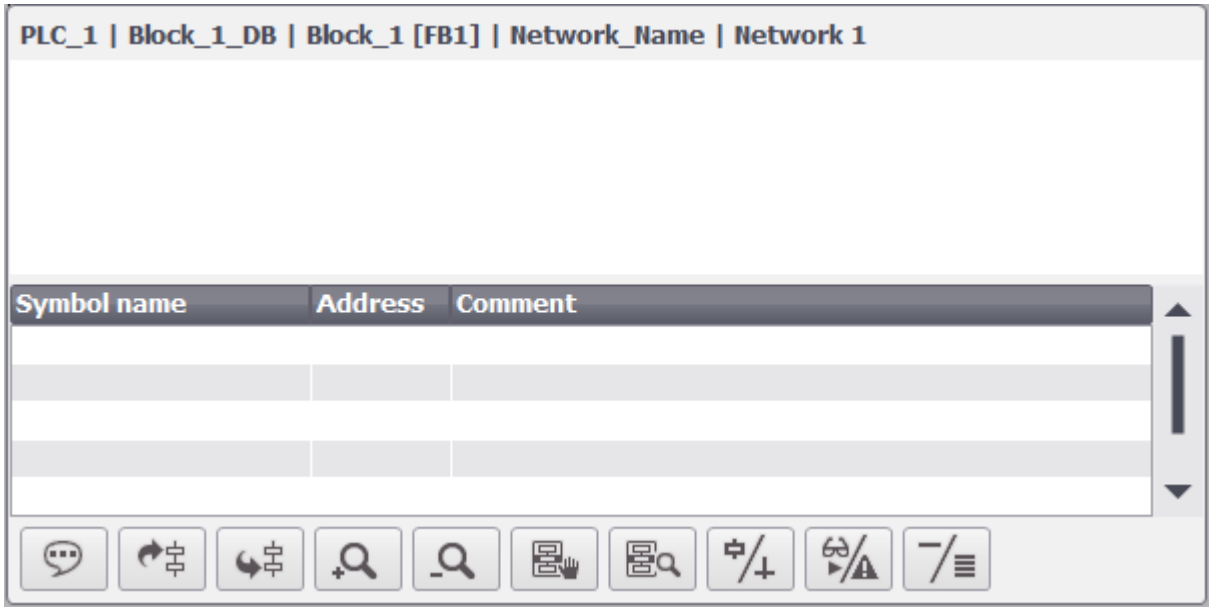

The PLC code view is divided into the following areas:

- Header
  The header shows the PLC name, the FB name, as well as the number and name of the network in Runtime.
- Information area
  The PLC network is graphically represented in the information area in Runtime.
- Symbol table
  The symbol table shows the symbolic names of the operands, absolute addresses of the operands and corresponding comments.
- Toolbar
  The toolbar shows the buttons available in the PLC code view in Runtime.

Note

The "PLC Code View" object cannot be used in faceplates, in the global screen, in pop-up screens and slide-in screens.

Note

Device dependency of the "PLC code view" object

The "PLC code view" object is available for RT Advanced, Comfort Panels and KTP Mobile Panels with 7" display size or more.

Layout

You change the settings for the position, style, colors, and fonts of the object in the Inspector window. You also, importantly, adjust the toolbar buttons here.

- You specify the display of path information in the header of the PLC code view under "Display > Show header (path)".
- You specify the number of lines that are displayed in the symbol table in Runtime under "Symbol table lines > Number of visible lines".
- You specify the width of the detail view relative to the information area under "GRAPH detail area > GRAPH detail area width (%)".
- You specify the two-line display of GRAPH step names in Runtime under "GRAPH view > Two-line view".
  To enable the two-line view, the "Show two lines" option must be selected under "Options > Settings > PLC programming > GRAPH > Length of step and transition names".

Toolbar

The operator controls with which the PLC code view is operated in Runtime are configured under "Properties > Properties > Toolbar > Buttons" in the Inspector window.

Operator control	Designation	Function
	"Symbol area"	Shows the symbol area.
	"Previous network"	Navigates to the previous network
	"Next network"	Navigates to the next network
	"Zoom in"	Enlarges the information area.
	"Zoom out"	Reduces the information area.
	"Detail"	Shows the detail view.
	"Step mode"	Switches between manual and automatic step selection for the active step.

Operator control	Designation	Function
	"Transition or Interlock"	Switches between the transition and interlock network views.
	"Actual values or initial values"	Switches between the views of actual values and initial values.
	"Single network or entire function block"	Switches between the view from a single network with the input logic and the view of the entire function block.

Configuring the PLC code view

1. Drag-and-drop the PLC Code View from the toolbox view.
2. Click "Properties > Properties > Toolbar" in the Inspector window.
3. Select the buttons that you require in Runtime, for example: Next network, Previous network, Step mode.
4. If required, configure the button display, e.g. background color or fill pattern.
5. In the Inspector window, select "Properties > Properties > Columns".
6. Select the columns that you require in the device view for Runtime.
7. Customize the column headers or order if necessary.
8. Select an authorization for operation in "Properties > Properties > Security".

Controlling the PLC code view with the keyboard

To control the PLC code view with the keyboard, proceed as follows:

1. Press the <Tab> key repeatedly until the focus is on the desired button.
   You can use the <Shift + Tab> shortcut keys to navigate back to the previous button.
2. Press <Enter>.
   The command is executed.

Operation with the touch screen

Scrolling by dragging in the symbol table of the PLC code view is not supported.
To scroll horizontally and vertically in the symbol table, use the scroll bars and the scroll arrows.

Multilingual GRAPH names

WinCC supports the display of step names and transition names for the GRAPH blocks in multiple languages starting from Version 5.0. The step names and transition names will then be displayed in the selected Runtime language following a language changeover in Runtime. If the selected language is not available in the GRAPH block, the names are displayed in the default language (English).

Mnemonics in the PLC code view

The PLC code viewer shows the data in the mnemonics you have set in TIA Portal. The international mnemonics are set by default. The international mnemonics provide e.g. the displayed addresses in all configured languages in runtime with the ID "IO", independent of the configured runtime languages.

Properties of a PLC in Runtime

If a PLC is connected to the PLC code view in WinCC and Runtime is active, changing the name of the PLC will trigger errors. Therefore, do not change the PLC name, the IP address, or other properties of the HMI connection.

Addressing tags in the "PLC code view" object

The "PLC code view" object only supports symbolic addressing of tags. If the operand is not addressed symbolically, the network with the operand cannot be displayed and an error message is generated.

Displaying time-value constants

When time-value constants are displayed, the time is shown in the format "Hours/Minutes/Seconds/" in the PLC code view, depending on the input configured in the Engineering System. For example, a constant "150 s" specified in the Engineering System appears in Runtime as "2 min 30 s".

See also

Supervising machinery and plants with ProDiag (Page 1852)

Supported data types (Page 1884)

Supported instructions (Page 1879)

Configuring the PLC code view (Page 1873)

Configuring the PLC code view (Page 1873)

Device dependency of the objects (Page 326)

1.9.2.54 PLC code view (RT Professional)

Use

The "PLC Code View" object is available for the HMI device, WinCC Runtime Professional. This object is used to display the current program status of PLC programs which have been programmed in the LAD, FBD or GRAPH programming languages. Errors during execution of a GRAPH step sequencer are displayed directly at the corresponding step.

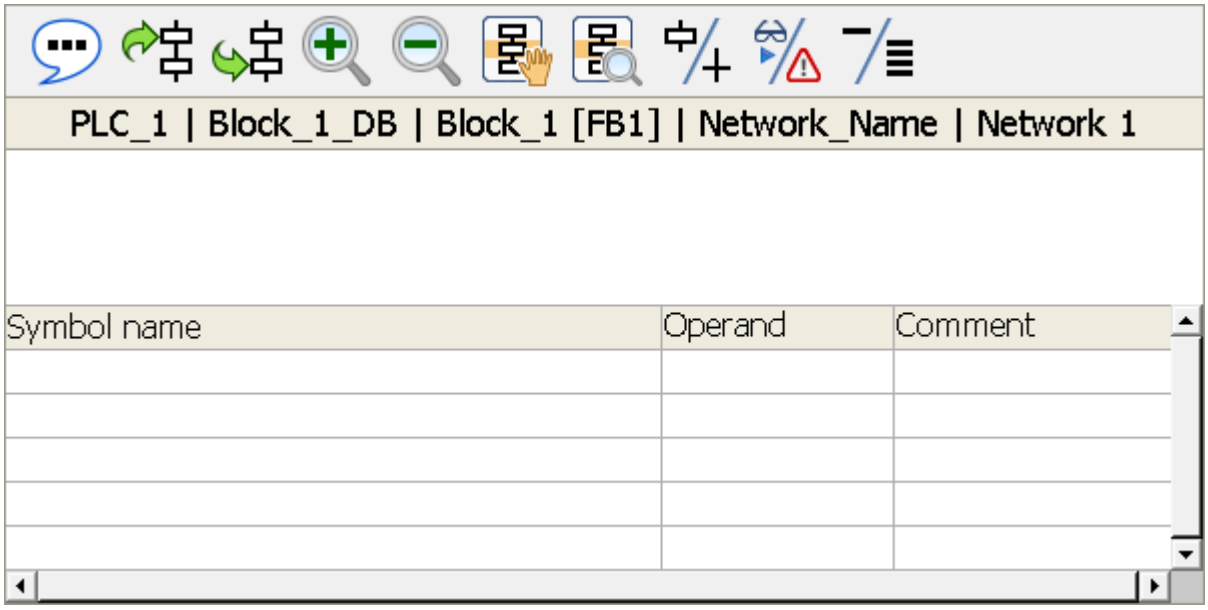

The PLC code view is divided into the following areas:

- Toolbar
  The toolbar shows the buttons available in the PLC code view in Runtime.
- Header
  The header shows the PLC name, the FB name, as well as the number and name of the network in Runtime.
- Network area
  The PLC network is graphically represented in the network area in Runtime.
- Symbol table
  The symbol table shows the symbolic names of the operands, absolute addresses of the operands and corresponding comments.

Note

The "PLC Code View" object cannot be used in groups or faceplates.

Layout

In the Inspector window, you customize the position, style, colors and font types of the object. You can adapt the following properties in particular:

- Buttons of the toolbar under "Toolbar > Toolbar buttons"
- Number of lines that are displayed in the symbol table in Runtime under "Symbol table lines > Number of visible lines"
- Two-line display of GRAPH step names in Runtime under "GRAPH view > Two-line view" To enable the two-line view, the "Show two lines" option must be selected under "Options > Settings > PLC programming > GRAPH > Length of step and transition names".

Displaying the network

The network to be displayed in the PLC code view is determined by user-defined functions. Depending on the programming language of the PLC program, one of the following API functions is called from within the user-defined function:

- OpenViewerIECPLByCall
- OpenViewerIECPLByFCCall
- OpenViewerIECPLByAssignment
- OpenViewerS7GraphByBlock

You can find detailed information and examples of how to set up a user-defined function in the Description of API functions.

Toolbar

The operator controls with which the PLC code view is operated in Runtime are configured under "Properties > Properties > Toolbar > Buttons" in the Inspector window.

Operator control	Designation	Function
	"Symbol area"	Shows the symbol area.
	"Previous network"	Navigates to the previous network
	"Next network"	Navigates to the next network
	"Zoom in"	Enlarges the information area.
	"Zoom out"	Reduces the information area.
	"Detail"	Shows the detail view.

Operator control	Designation	Function
	"Step mode"	Switches between manual and automatic step selection for the active step.
	"Transition or Interlock"	Switches between transition view and interlock view.
	"Actual or initial values"	Switches between the views of actual values and initial values.
	"Single network or entire function block"	Switches between the view from a single network with the input logic and the view of the entire function block.

Multilingual GRAPH names

WinCC supports the display of step names and transition names for the GRAPH blocks in multiple languages starting from Version 5.0. The step names and transition names will then be displayed in the selected Runtime language following a language changeover in Runtime. If the selected language is not available in the GRAPH block, the names are displayed in the default language (English).

Properties of a PLC in Runtime

If a PLC is connected to the PLC code view in WinCC and Runtime is active, changing the name of the PLC will trigger errors. Therefore, do not change the PLC name, the IP address, or other properties of the HMI connection.

Error message when PLC is loaded with a newer version of the TIA Portal

An error message is output if the PLC code view from an older version of WinCC accesses a program on the PLC that is loaded with a new version. Make sure that the PLC and the HMI are loaded with the same version of the TIA Portal.

Addressing tags in the "PLC code view" object

The "PLC code view" object only supports symbolic addressing of tags. If the operand is not addressed symbolically, the network with the operand cannot be displayed and an error message is generated.

See also

Configuring the PLC code view (Page 1873)

Configuring the PLC code view (Page 1873)

Supervising machinery and plants with ProDiag (Page 1852)

Supported instructions (Page 1892)

Supported data types (Page 1884)

1.9.2.55 Polygon (RT Professional, RT Advanced, Panels, Comfort Panels)

Application

The "Polygon" is a closed object which you can fill with a background color.

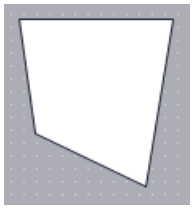

Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. In particular, you can customize the following property:

Geometry: Modifies, deletes or adds corners.

Geometry

The corner points are numbered in the order of their creation. You can change, delete, or add more corner points:

1. In the Inspector window, select "Properties > Properties > Layout".
2. Select the required corner point in the "Geometry" area. Input a value at "X position " and "Y position ".
3. To add a corner point, click the button.
4. To delete a corner point, click the button.

Alternative procedure

You can also change, delete, or add new corner points using the mouse:

1. Select the object. Position the mouse cursor on the desired corner. The mouse cursor changes to a crosshair.
2. Drag the corner to the desired position while holding down the mouse button.
3. Right-click the position at which you want to insert the corner point. Select "Add point" from the shortcut menu.

Configuring Rotation in runtime

You configure the "Polygon" object so that it rotates about a reference point in runtime. The rotation in runtime is only possible for devices with Runtime Professional.

Enter the values for the angle of rotation using Degrees as the unit.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter values for the following attributes in the "Rotation" area.
   - X
   - Y
   - Angle

See also

Polyline (Page 454)

Rotating an object in runtime (Page 129)

1.9.2.56 Polyline (RT Professional, RT Advanced, Panels, Comfort Panels)

Application

The "Polyline" is an open object. Use the "Polygon" object if you want to fill the object with color.

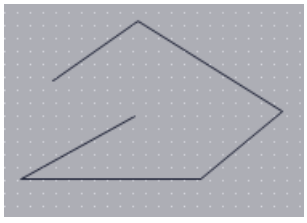

Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Line start and end: Specifies the type of line start and line end.
- Geometry: Modifies, deletes or adds corners.

Line start and end

You define the start point and the end point of the line under "Properties > Properties >"Appearance" in the Inspector window. Use arrow point, for example, as start and end point. The available start and end points depend on the device.

Geometry

The corner points are numbered in the order of their creation. You can change, delete, or add more corner points:

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select the required corner point in the "Geometry" area. Input a value at "X position" and "Y position".
3. To add a corner point, click the button.
4. To delete a corner point, click the button.

Alternative procedure

You can also change, delete, or add new corner points using the mouse:

1. Select the object. Position the mouse cursor on the desired corner. The mouse cursor changes to a crosshair.
2. Drag the corner to the desired position while holding down the mouse button.

Configuring rotation in Runtime

You configure the "Polyline" object so that it rotates about a reference point in Runtime. The rotation in Runtime is only possible for devices with Runtime Professional.

Enter the values for the angle of rotation using Degrees as the unit.

1. Click "Layout" in the Inspector window.
2. Enter values for the following attributes in the "Rotation" area.
   - X
   - Y
   - Angle

See also

Polygon (Page 453)

1.9.2.57 ProDiag overview (RT Professional, RT Advanced, Panels, Comfort Panels)

Use

The "ProDiag overview" object provides an overview of the current status of the configured monitoring in Runtime. When an error occurs, the type of error and the error category are determined in the ProDiag overview. You can navigate directly to the error message in an alarm view and have the affected program code shown in the PLC code view.

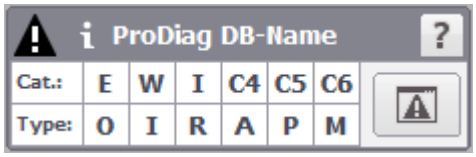

Note

The "ProDiag overview" object cannot be used in faceplates or in the global screen.

Note

Device dependency of the "ProDiag overview" object

The "ProDiag overview" object is available for RT Advanced, RT Professional, Comfort Panels and KTP Mobile Panels.

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Displayed buttons
- Names and colors for categories
- Names and colors for monitoring types

Configuring the ProDiag overview

A CPU and a ProDiag FB are created.

1. In the Inspector window, select "Properties > Properties > General".
2. Open the selection button under the "Tag" property.
3. Select the status tag of ProDiag FB.
   Alternatively, drag the status tag from the detail view and drop it at the desired point of use.
4. Under "Output > Categories", define the names, colors and flashing behavior for the categories.

5. Under "Output > Supervision types", define the names, colors and flashing behavior for the supervision types.
6. Select an authorization for operation under "Properties > Properties > Security".

Configuring a compact view

You can configure the ProDiag overview in a compact view.

Disable the display of categories and monitoring types under "Properties > Appearance > Options". Disable all buttons under "Properties > Toolbar > General".

Monitoring types and categories

You can display a maximum of eight categories and six types of monitoring in the "ProDiag overview" object. The pre-defined categories and monitor types are available:

Designation	Categories
E	Error
W	Warning
I	Info

Rename the categories C4 to C8, which are created by default, according to your requirements.

Designation	Monitoring type
O	Operand error
I	Interlock error
R	Reaction error
A	Action error
P	Position error
M	Alarm

You can change the display symbols of monitoring types and categories at any time under "Output" in the Inspector window.

Symbols

You specify the symbols that are displayed in the ProDiag overview in the Inspector window under "Properties > Appearance > Options". The following symbols are available for ProDiag overview:

Icon	Name	Function
?	Help	Opens the user-defined tooltip.
i	Info	Displays the information for monitoring.
!	Error	Indicates that an error has occurred.

"Alarm view" button

To display the "Alarm view" in the ProDiag overview, select the "Show alarm window button" property under "Properties > Toolbar > General".

Button	Name	Function
	Alarm view	Opens the configured alarm view with the error message.

Deactivated display

If there is a faulty connection to the controller during runtime, the object "ProDiag overview" is displayed grayed-out (unavailable). This deactivated display can be due to the following:

- The controller is deactivated
- The configured ProDiag program block was removed from the control program
- The controller is in stop mode

As soon as the cause of the error is removed and the connection reestablished, the ProDiag overview shows the current online status of the monitoring during runtime.

See also

Basics of supervision with ProDiag (Page 1852)

Basics of supervising operands (Page 1852)

Configuring the ProDiag overview (Page 1862)

Supervising machinery and plants with ProDiag (Page 1852)

1.9.2.58 Rectangle

Application

The "Rectangle" is a closed object which you can fill with a color.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Corner radius: Specifies the horizontal and vertical distance between the corner of the rectangle and the start point of a rounded corner.

Corner radius

The corners of the "Rectangle" object can be rounded to suit your requirements. When the properties "X" and "Y" are set to the 100 % value, the rectangle is displayed as an ellipse. As soon as one of the properties has the value 0%, a normal rectangle without a rounded corner is shown.

1. Click "Properties > Properties > Layout" in the inspector window.
2. Enter a value for "X" in the "Corner radius" area.
   The input value is the percentage proportion of half the width of the rectangle.
3. Enter a value for "Y" in the "Corner radius" area.
   The input value is the percentage proportion of half the height of the rectangle.

See also

Device dependency of the objects (Page 326)

1.9.2.59 Recipe view (RT Professional)

Application

The "Recipe view" object is used to display and modify recipes.

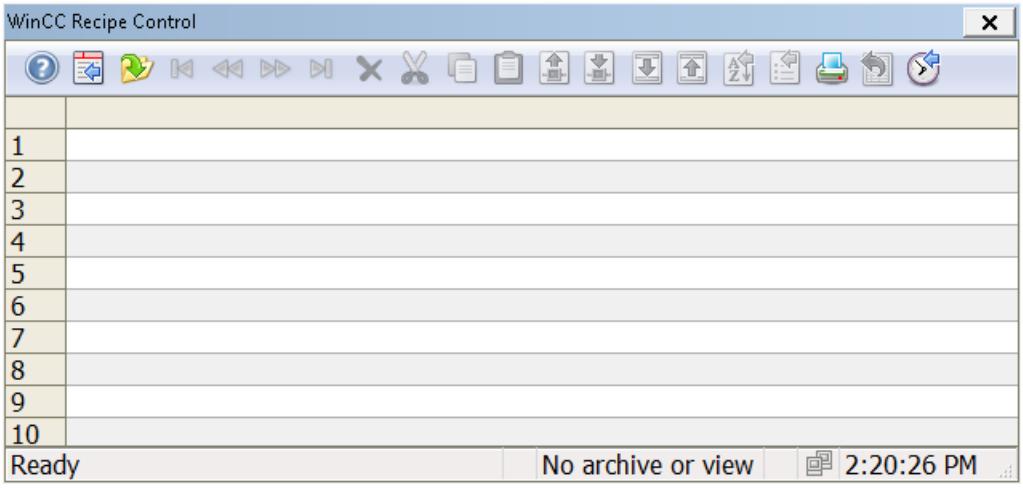

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Log/View: Defining a recipe or recipe query
- Toolbar: Defines the operator controls of the recipe view.
- Columns: Specifies the columns that are displayed in the recipe data record.

Log/View

Define the recipe or recipe query to be displayed in the recipe view under "Properties > Properties > General > Log/View" in the Inspector window.

Toolbar

The control elements with which the recipe view is operated in Runtime are configured under "Properties > Properties > Buttons" in the Inspector window. In the simple recipe view the operating elements are based on the functions of the menu.

Operator control		Description
	"Tooltip"	Calls the info text for the recipe display.
	"Configuration dialog"	Opens the configuration dialog box in which you change the properties of the recipe display.

Operator control		Description
	"Select data connection"	Opens a dialog for selecting a recipe or recipe query. The recipe data are then shown in the table of the recipe view.
	"First row"	The first data record is displayed in the table by the button.
	"Previous row"	The previous data record is displayed in the table by the button.
	"Next row"	The next data record is displayed in the table by the button.
	"Last row"	The last data record is displayed in the table by the button.
	"Delete row"	The content of the marked lines are deleted.
	"Cut lines"	The content of the marked lines are cut out.
	"Copy lines"	The content of the marked lines are copied.
	"Insert lines"	The content of the copied or cut-out lines is inserted starting from the marked line.
	"Read tags"	The contents of the connected WinCC tags are read and written into the recipe elements by the button. The "Tags" communication type must be activated in the displayed recipe to be able to use the button. The recipe elements must be connected with tags.
	"Write tags"	The contents of the recipe elements are written into the connected WinCC tags. The "Tags" communication type must be activated in the displayed recipe to be able to use the button. The recipe elements must be connected with tags.
	"Import Archive"	Using a button, a CSV file is imported from the "ua" directory of the project folder into the table of the recipe view.
	"Export archive"	The original contents of the table are exported with table headers by the button when loading. The recipe is stored as a CSV file in the "ua" directory of the project folder.
	"Sort dialog"	Opens a dialog box for setting a customized sorting of the displayed columns.
	"Print"	Starts the printout of the displayed values. The print job used for printing is defined in the configuration dialog in the "General" tab.
	"Set time base"	Opens a dialog box for setting the time base for the used times.

Operator control		Description
	"Export data"	Using a button, you can export the recipe or recipe query to a CSV file as configured but with the current runtime data. If the "Display dialog" option is active, a dialog opens in which you can view the settings for exporting and can start the export. With the appropriate operator authorizations you may also select the file and the directory for export. If no dialog is displayed, export of data to the default file is started immediately.
	"User-defined"	Displays the first button created by the user. The function of the button is customized.

Columns

In the Inspector window under "Properties > Properties > Columns > Column properties", you define which columns are displayed in the recipe view.

Configuration behavior

Displaying column headers

The layout of the recipe view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Appearance" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

See also

Displaying recipes (Page 1181)

Properties of the recipe view (Page 1182)

Editing recipe data (Page 1207)

Configuring persistence (Page 2264)

Device dependency of the objects (Page 326)

1.9.2.60 Recipe view (as of V13) (RT Advanced, Basic Panels, Panels, Comfort Panels)

Use

The "Recipe view" object is used to display recipes on the HMI device.

Note

Device dependency of the "Advanced recipe view" object

The "Advanced recipe view" object is available on the HMI devices Basic Panels 2nd Generation, Comfort Panels, Mobile Panels with the device version V13.

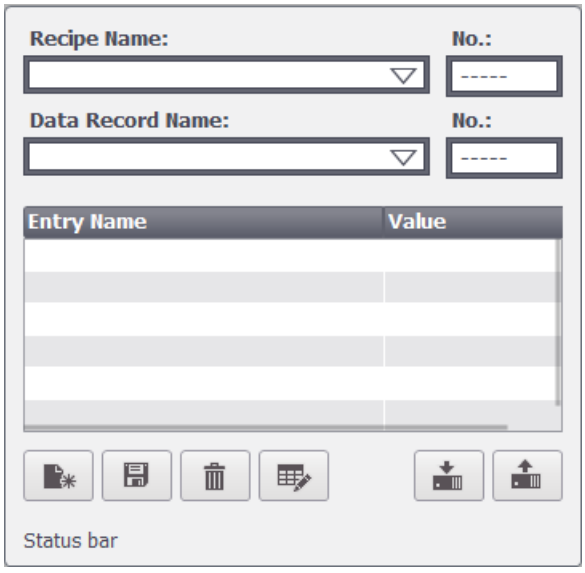

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Toolbar: Defines the operator controls of the recipe view.
- Display number: Defines whether the number of the recipe and the number of the recipe data record is displayed.
- Label: Sets the label for the name of the recipe and the recipe data record.

Operator controls

The operator controls with which the recipe view is operated in Runtime are configured under "Properties > Properties > Buttons" in the Inspector window. In the simple recipe view the operating elements are based on the functions of the menu.

Operator control		Description
	"Info text"	Calls up the configured tooltip for the selected recipe.
	"Add data record"	Creates a new recipe data record in the recipe.
	"Delete data record"	Deletes the selected data record.
	"Rename"	Changes the name of selected data record.
	"Save"	Saves the modified record with its current name.
	"Save as..."	Saves the modified record with a new name.
	"Write to PLC"	Sends the current value to the PLC.
	"Read from PLC"	Reads the current value from the PLC.
	"Synchronize recipe tags"	Compares the values of the selected data record with the values on the PLC.

Display number

Specifies whether the number of the recipe and the number of the recipe data record are displayed in Runtime. The number of the recipe uniquely identifies the recipe in the project.

1. Click "Properties > Properties > View" in the Inspector window.
2. Select "Display > Display numbers".

Display label

Use this property to specify the name to be shown for recipes and data records in the recipe view.

1. Click "Properties > Properties > Label" in the Inspector window.
2. Select "Display labels".
3. Enter the labels for the recipe and the recipe data record.

Behavior during configuration

Displaying column headers

The layout of the recipe view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display >

Appearance" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

Scrolling in the recipe view

A recipe view contains an element that is shown as a text list in the recipe table. You can only scroll between the elements in the table of the recipe view, if you have previously selected an element that does not contain a text list.

1.9.2.61 Recipe view, simple (RT Advanced, Basic Panels, Panels, Comfort Panels)

Use

The "Recipe view" object is used to display and modify recipes.

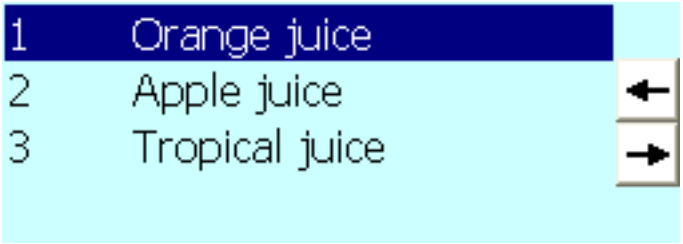

Note

Configure the "Simple recipe view" object on devices with device version 12.0.0.0 or older.

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Toolbar: Specifies the menu commands of the recipe view.

Toolbar

The menu items with which the recipe view is operated in Runtime are configured under "Properties > Properties > Toolbar" in the Inspector window.

Menu command	Description
"Tooltip"	Calls up the configured tooltip for the selected recipe.
"Add data record"	Creates a new recipe data record in the recipe.
"Delete data record"	Deletes the selected data record.
"Save"	Saves the modified record with its current name.

Menu command	Description
"Save as"	Saves the modified record with a new name.
"Write to PLC"	Sends the current value to the PLC.
"Read from PLC"	Reads the current value from the PLC.
"Rename"	Specifies that the "Rename" button is shown.
"Forward"	Specifies that the menu buttons are visible.
"Back"	Specifies that the "Back" button is shown.

1.9.2.62 Recipe view, advanced (RT Advanced, Basic Panels, Panels, Comfort Panels)

Advanced recipe view

The "Recipe view" object is used to display and modify recipes. The advanced or the simple user view is available depending on the HMI device

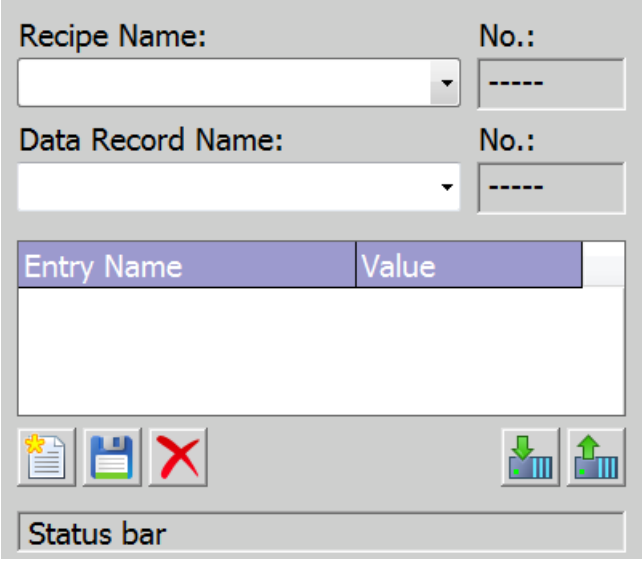

Changing the view type

On HMI devices with a display larger than 6", both the advanced and the simplified recipe view can be used to manage and process recipes. The view type is only available for configuration on devices up to device version V13.

Note

Do not use the simple recipe view in a group.

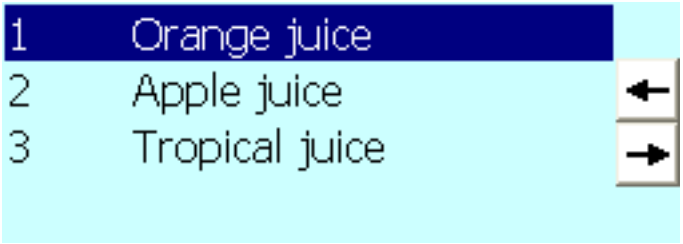

Configuring the simple recipe view:

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select "Mode > Display type > Simplified".

Note

Behavior for Runtime Professional, Runtime Advanced and Panels

In the Engineering System you can dynamically control the visibility of an object, for example, in the "Animations" tab of the Inspector window. In Runtime, the "Simple recipe view" does not support animations. If you have configured an animation and, for example, want to check the consistency of the project, an error alarm is displayed in the Output window.

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Operator controls: Defines the operator controls of the recipe view.
- Display number: Defines whether the number of the recipe and the number of the recipe data record is displayed.

Operator controls

The operator controls with which the recipe view is operated in Runtime are configured under "Properties > Properties > Buttons" in the Inspector window. In the simple recipe view the operating elements are based on the functions of the menu.

Operator control		Description
	"Tooltip"	Calls up the configured tooltip for the selected recipe.
	"Add data record"	Creates a new recipe data record in the recipe.
	"Delete data record"	Deletes the selected data record.
	"Save"	Saves the modified record with its current name.
	"Save as"	Saves the modified record with a new name.
	"Synchronize recipe tags"	Compares the values of the selected data record with the values on the PLC.
	"Write to PLC"	Sends the current value to the PLC.
	"Read from PLC"	Reads the current value from the PLC.

Display number

Specifies whether the number of the recipe and the number of the recipe data record are displayed in Runtime. The number of the recipe uniquely identifies the recipe in the project.

1. Click "Properties > Properties > View" in the Inspector window.
2. Select "Display > Display numbers".

Configuration behavior

Displaying column headers

The layout of the recipe view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Appearance" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

See also

Recipe view, simple (Page 465)

Description of the advanced recipe view (prior to V13) (Page 1108)

Device dependency of the objects (Page 326)

1.9.2.63 Round button (RT Professional)

Application

The "Round button" object is used to control processes. A round button is used, for example, to confirm alarms. To incorporate a round button into the process, dynamize the corresponding attribute.

Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. In particular, you can customize the following property:

Mode: Specifies whether the round button is shown pressed or as a toggle switch.

Type: The round button can be shown, for example, with text and graphic.

Show round button pressed

1. In the Inspector window, select "Properties > Properties > General".
2. Activate "Mode > Pressed".
3. To show the round button as two-way switch, activate "Mode > Toggle".

Specifying the round button type

1. In the Inspector window, select "Properties > Properties > General".
2. Select, for example, "Graphics and text" as "Type".

See also

Device dependency of the objects (Page 326)

1.9.2.64 Pipe (RT Professional)

Application

You reproduce a continuous pipe with several bends with the object "Pipe". To reproduce a pipe system, connect the "Pipe" object to other objects, e.g. pipe bend or T piece.

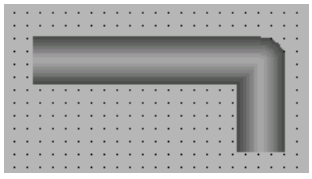

Layout

Customize the object position and color in the Inspector window. You can adapt the following properties in particular:

- Geometry: You add additional points and extend the object.

A pipe may have any number of corners. The corner points are numbered in the order of their creation. You can move them individually.

See also

Pipe elbow (Page 470)

T-piece (Page 502)

Double T-piece (Page 351)

Device dependency of the objects (Page 326)

1.9.2.65 Pipe elbow (RT Professional)

Application

A pipe bend is displayed using the "Pipe bend" object. To reproduce a pipe system, connect the "Pipe bend" object to other objects, e.g. pipe bend or double T piece.

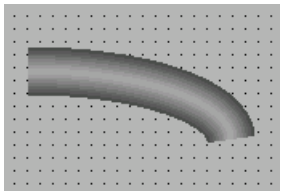

Layout

Customize the object position and color in the Inspector window. You can adapt the following properties in particular:

- Angle: You specify the start and end angle of the pipe bend.
- Radius: You specify the horizontal and vertical radius of the pipe bend.

The attributes are interdependent.

- When you change "Start angle" and "End angle", the width and height are also automatically changed.
  "Horizontal radius " and "Vertical radius" are retained.
- When you change "Horizontal radius" and "Vertical radius", the width and height are also automatically changed.
  "Start angle" and "End angle" are retained.
- When you change the width and the height, the "Horizontal radius" and "Vertical radius" are also automatically changed.
  "Start angle" and "End angle" are retained.

See also

Pipe (Page 469)

T-piece (Page 502)

Double T-piece (Page 351)

Device dependency of the objects (Page 326)

1.9.2.66 Switch (RT Advanced, Basic Panels, Panels, Comfort Panels)

Application

The "Switch" object is used to configure a switch that is used to switch between two predefined states in runtime. The current state of the "Switch" object can be visualized with either a label or a graphic.

The following figure shows a "Switch" type switch.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. In particular, you can customize the following property:

- Type: Defines the graphic representation of the object.

Type

Button visualization is specified at "Properties > Properties > General >Type" in the Inspector window.

Type	Description
"Switch"	The two states of the "Switch" are displayed in the form of a switch. The position of the switch indicates the current state. The state is changed in runtime by sliding the switch. You specify the direction of movement of the switch in "Switch orientation" with this type.
"Switch with text"	The switch is shown as a button. The current state is visualized with a label. In runtime click on the button to actuate the switch.
"Switch with graphic"	The switch is shown as a button. The current state is visualized with a graphic. In runtime click on the button to actuate the switch.

Note

Basic Panels

The "Switch" type is not available for Basic Panels.

See also

Overview of objects (Page 97)

Device dependency of the objects (Page 326)

1.9.2.67 Button (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Application

The "Button" object allows you to configure an object that the operator can use in Runtime to execute any configurable function.

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Defines the graphic representation of the object.
- Text / Graphic: Defines whether the Graphic view is static or dynamic.
- Define hotkey: Defines a key, or shortcut that the operator can use to actuate the button.

Note

You can only define a hotkey for HMI devices with keys.

Mode

The button display is defined in "Properties > Properties > General >Mode" in the Inspector window.

Mode	Description
"Invisible"	The button is not visible in Runtime.
"Text"	The button is displayed with text. This text explains the function of the button.
"Graphic"	The button is displayed with a graphic. This graphics represents the function of the button.
"Graphics or text"	The button is displayed with text or graphics. If the graphics cannot be displayed, the corresponding text is displayed.
"Graphics and text"	The button is displayed with text and graphics.

Different options are available depending on the device.

Text / Graphic

The "Mode" property settings are used to define whether the display is static or dynamic. The display is defined in "Properties > Properties > General >Text" or "Graphic" in the Inspector window.

You can, for example, select the following options for the "Graphic" or "Text" type.

Type	Option	Description
"Graphic"	"Graphic"	Use "Graphic when button is "not pressed"" to specify a graphic displayed in the button for the "OFF" state. Use "Graphic when button is "pressed"" to specify a graphic for the "ON" state.
	"Graphics list"	The graphic in the button depends on the state. The corresponding entry from the graphics list is displayed depending on the state.
"Text"	"Text"	Use "Text when button is "not pressed"" to specify the text displayed in the button for the "OFF" state. Use "Text when button is "pressed"" to specify text for the "ON" state.
	"Text list"	The text in the button depends on the state. The entry from the text list corresponding to the state is displayed.

Note

You cannot use text lists and graphics lists in buttons in WinCC Runtime Professional.

Define hotkey

In the Inspector window, a key or key combination is defined that the operator can use to control the button in Runtime.

1. In the Inspector window, select "Properties > Properties > General".
2. Select a key or key combination from the selection list in the "Hotkey" area.

See also

Example: Configuring a button for language switching (Page 240)

Example: Configuring a button with access protection (Page 1365)

Device dependency of the objects (Page 326)

1.9.2.68 Slider

Use

Process values are monitored and adapted within a defined range with the "Slider" object. The monitored range is visualized in the form of a slider. By adjusting the slider, you intervene in the process and correct the displayed process value.

Note

Do not configure a system function at the "Change" event for a regulated project if it is used to execute a GMP-relevant action.

Instead configure the system functions at the "Value change" event of the tags in order to reduce the number of user actions.

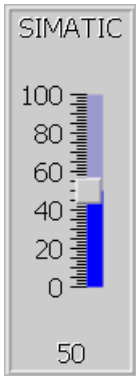

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can, in particular, adapt the following properties as required:

- Maximum Value and Minimum Value: Specifies the top and bottom values of the scale.
- Display current value: Specifies whether the current position of the controller appears below the slider.
- Display of bars: The sliders above and below the bar can be hidden.

Maximum Value and Minimum Value

The top and bottom end values of the scale are specified in the Inspector window.

1. In the Inspector window, activate "Properties > Properties > General".
2. Enter a number at "Maximum scale value" and "Minimum scale value". If you select a tag as the end value of the scale, the number will be no longer available.

Configuring limits/ranges

You can display limits and ranges in different colors. You define the colors in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Limits/Ranges".
2. Activate the limits/ranges that are to be displayed in runtime.
3. You can change the default colors for the ranges, if required.

Note

When you select the option "Show ranges from tag", you can display up to five ranges in the slider whose values are specified by means of a process tag. You define the values for the ranges with a process tag, which you interconnect with the screen object.

The option "Show ranges from tag" is available for Comfort Panels, KTP Mobile Panels and RT Advanced.

Display current value

Specify that the value of the current position is displayed below the slider in the Inspector window.

1. In the Inspector window, activate "Properties > Properties > Layout".
2. Select "Show current value."

Display bar

The bar can be hidden.

1. In the Inspector window, activate "Properties > Properties > Layout".
2. Deactivate "Display bar".

Behavior during operation

The displayed value on the slider control may deviate from the actual value of the associated tag in the following circumstances:

- The value range (minimum and maximum value) configured for the slider control does not correspond to the configured limits for the slider control tag.
- An invalid password has been entered for a password-protected slider control.

See also

Device dependency of the objects (Page 326)

1.9.2.69 Key switch (Panels, Comfort Panels)

Use

The object "KeySwitch" is an optional control element for the SIMATIC Mobile Panel. In runtime, the key switch is used to lock functions that can be triggered from the Mobile Panel.

You configure the object "Key switch" object in the "global screen".

Tag

The "Tag" property is used to specify the "Key switch" object assignment in the global screen.

Note

Notes on configuring

When starting runtime and every time the key is pressed the current position of the key switch is written to the tag. In doing so it can cause inconsistencies between the position of the key switch and the key switch tags.

1. If the tag has a process connection, the current value is transferred from the PLC to the tag after establishment of communication. The tag that contains the current position of the key is overwritten by this process. It can be that the key switch tag no longer reflects the current value of the key switch, e.g. if the Mobile Panel is switched off while the key is being turned.
2. If the key switch is pressed before communication with the PLC has been established (e.g. after a device start-up), it may not be possible to send the value of the key switch tag to the PLC. In this case the value of the tag in the PLC is different from the current position of the key.

A specific configuration is required to make sure that the key switch tag reflects the actual position of the key switch at all times. See the instructions below for the basic procedure.

Basic procedure

1. Specify the connection to the PLC in the "Connections" editor. Activate the "Coordination" area pointer to ensure that the life bit is available to the PLC side.

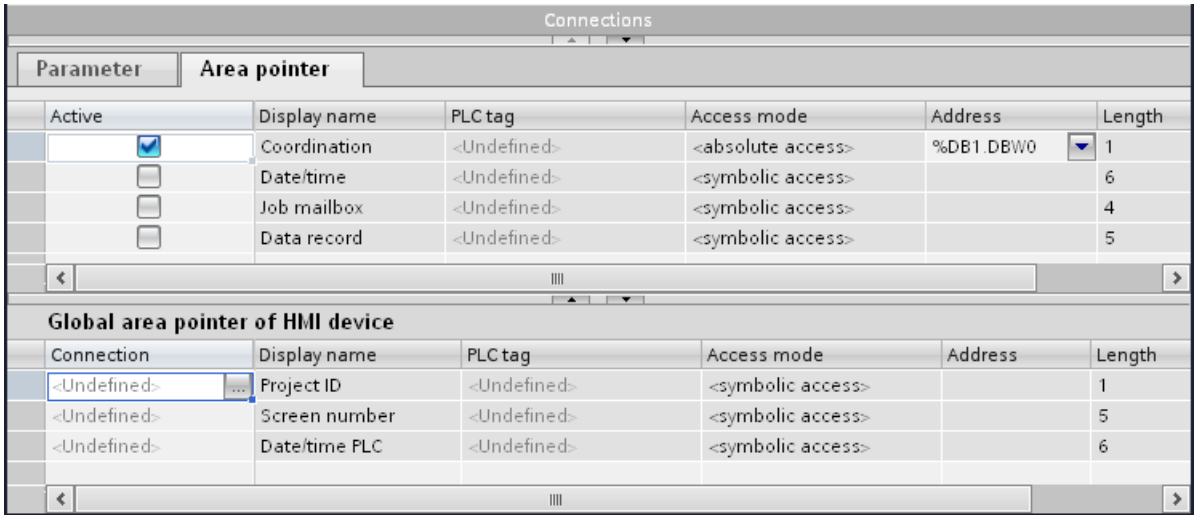

2. Create three tags in the "Tag" editor.
   - Internal tag: Position_Keyswitch
   - External tag: Auxiliary tag
   - External tag: Keyswitch_PLC
3. Open the Inspector window of the "Position_Keyswitch" tag.
4. In the Inspector window, select "Properties > Events > Value change".
5. Click the first line of the function list. The list opens, showing the system functions available in the project.

6. Select the "SetTag" system function.
   - Select the "Keyswitch_PLC" tag at "Tag (output)".
   - Select the "Position_Keyswitch" tag at "Value".

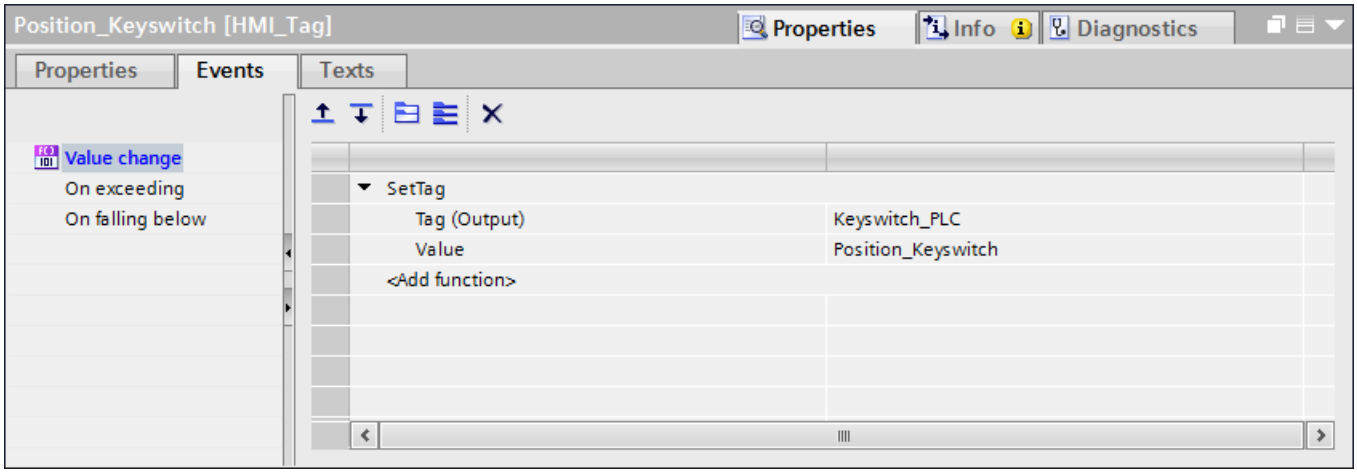

The value of the "Keyswitch_PLC" tag is written to the PLC with the "Position_Keyswitch" tag. A program in the PLC evaluates the life bit. If communication is established, the current value is written to the "Keyswitch_PLC" tag from the PLC.

7. Open the Inspector window of the "Auxiliary tag" tag.
8. In the Inspector window, select "Properties > Events > Value change".
9. Click the first line of the function list. The list opens, showing the system functions available in the project.
10. Select the "SetTag" system function.
    - Select the "Keyswitch_PLC" tag at "Tag (output)".
    - Select the "Position_Keyswitch" tag at "Value".

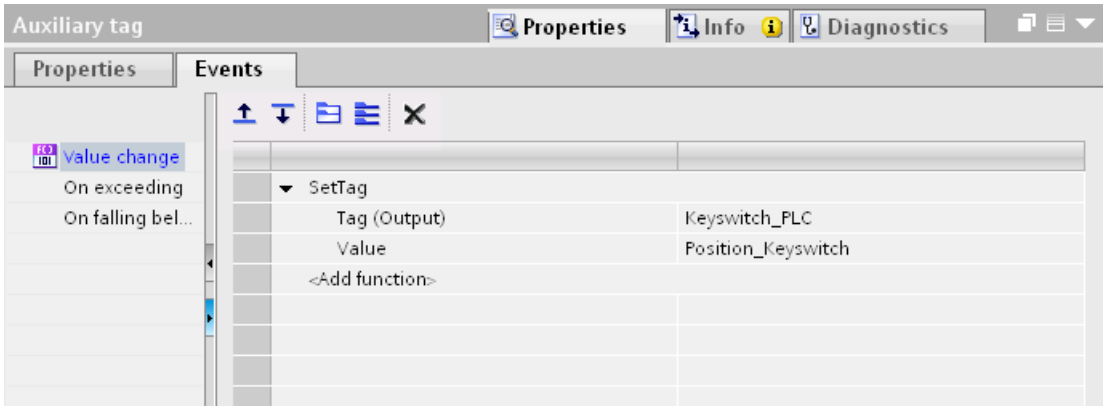

After establishing communication the current value is written to the "Auxiliary Tag" from the PLC. The "SetTag" system function is executed by the value change in the auxiliary tag. The system function re-allocates the value of the "Position_Keyswitch" tag to the "Keyswitch_PLC".

11. Open the global screen in the "Screen management" editor.

12. Select the key switch in the screen.
13. Select the "Position_KeySwitch" tag in the Inspector window under "Properties > Properties > General > Settings > Tag".
    If you actuate the key switch, the value is written to the "Position_KeySwitch" tag.

1.9.2.70 Sm@rtClient view (RT Advanced, Panels, Comfort Panels)

Application

The "Sm@rtClient View" object is used to configure a network connection to the remote monitoring and remote maintenance of a connected unit.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Connect on start: Establishes whether when opening the Sm@rtClient view the connection to the HMI device is automatically created which should be remote-controlled.
- Shared use_ Specifies that an HMI device is shared by several Sm@rtClient Views.
- Local cursor: Specifies whether the cursor data is transferred separately in order to increase the performance.
- Use cursor key scroll: For keyboard devices places the control of the horizontal and vertical scroll with the cursor keys.
- Show controls: Establishes whether additional fields exist for entering the address and password.
- Allow menu: Specifies whether the shortcut menu is enabled to control the Sm@rtClient view.
- Connection type: Establishes the expected transfer speed for the remote monitoring.
- Scaling: Specifies whether the Sm@rtClient view can be zoomed in or out.

Connect on start

Establishes whether when opening the Sm@rtClient view the connection to the HMI device is automatically created which is remote-controlled.

1. In the Inspector window, select "Properties > Properties > General".
2. Select "View > Connect on start".

Shared use

Define whether the HMI device can be used by several Sm@rtClient views under "Properties > Properties > General > View > Shared".

Shared use	
Enabled	HMI devices on which remote control is activated can access the remote HMI device and control the process. Only one HMI device can be active at a time in this case. A different HMI device can only assume control when there has been no activity on the active HMI device for a specified period of time.
Disabled	Only one HMI device can use the remote control at any given time. The activities can be followed on the other HMI devices. Depending on the settings on the Sm@rtServer the following options are available if another HMI device logs on: • The HMI device is rejected. • The active HMI device is disconnected and the new HMI device is connected.

Local cursor

If this property is set, then the cursor data is transferred separately.

1. In the Inspector window, select "Properties > Properties > General".
2. Select "View > Local cursor".

Display extended objects

If this property is set then there are additional fields for inputting the address and password. For the fields to be shown the "Connect on start" option is not activated.

1. In the Inspector window, select "Properties > Properties > General".
2. Select "View > Show enhanced objects".

Allow menu

If this property is set, then a shortcut menu to control the Sm@rtClient view is opened in Runtime.

When operating using the mouse, the left mouse key must remain depressed for a few seconds for the shortcut menu to open.

1. In the Inspector window, select "Properties > Properties > General".
2. Select "View > Allow Menu".

Connection type

The "Connection type" property establishes the expected transfer speed for the remote monitoring.

Set the connection type at "Connection type" under "Properties > Properties > General > Connection type" in the Inspector window.

Connection type	
LAN	Connection via network
Slow	Connection with a low transfer speed
Medium speed	Connection with a medium transfer speed
Modem	Connection via an analog modem
Very slow	Connection with a very low transfer speed

Scaling

If this property is set, then via "Client scale factor" and "Server scale" you can reduce or enlarge the Sm@rtClient view on the HMI device.

1. In the Inspector window, select "Properties > Properties > Scaling".
2. Select "Scaling > Scaling".
3. Enter the values for the scaling under "Client scale factor" and "Server scale factor".

Behavior during operation

If the program cannot resolve the dynamic address of a Sm@rtClient view after a screen change, the system connects to the configured server using the static address of the server. If a tag is now set using the dynamic address, you can make a second connection to this server. As a remedy preset the tags of the dynamic address for example by means of a script.

See also

Device dependency of the objects (Page 326)

1.9.2.71 Memory space view (RT Professional)

Application

The "Memory space view" object is used to display the memory space view. You can create one memory space display for each drive.

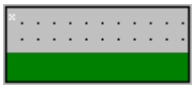

Layout

In the Inspector window, you customize the settings for the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Drive: Defines the drive whose memory assignment is displayed in the memory space display.
- Monitoring limits: Defines the values for the monitoring limits.

Define drive

1. In the Inspector window, activate "Properties > Properties > General".
2. Enter the relevant letter in the "Drive letter" area.

Change monitoring limits

Enter the monitoring limits for Alarm, Warning, and Tolerance in the Inspector view. The values are specified as a percentage. The percentage of the occupied memory space is monitored in relation to the total capacity of the specified drive:

1. In the Inspector window, activate "Properties > Properties > General".
2. Enter the value for each monitoring limit as a percentage in the "Limits" area.

See also

Device dependency of the objects (Page 326)

1.9.2.72 Symbol library (RT Professional, RT Advanced, Panels, Comfort Panels)

Application

The "Symbol library" object contains a large collection of ready-to-use icons. These icons are used to represent systems and plant areas in screens.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Select icon: Specifies which library object will be used.
- Fill style: Specifies the graphic design of the object.
- Flip: Specifies the flip type of the library object.
- Rotate: Specifies the angle around which the library object is rotated.
- Fixed aspect ratio Defines whether the aspect ratio is maintained if the size is changed.

Select symbol

Select the icon at "Properties > Properties > General" in the Inspector window.

Fill style

Specify the design of the library object at "Properties > Properties > Appearance > Style" in the Inspector window.

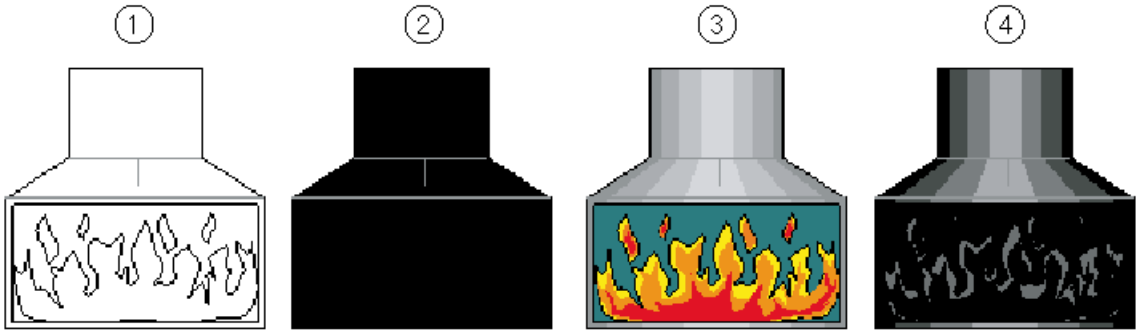

1	"Hollow"	The library object is displayed in outline.
2	"Solid"	Black lines remain as outlines. Elements of the icon with a different color are displayed as a main color.
3	"Original"	The screen object will not be changed.
4	"Shaded"	Black lines remain as outlines. Elements of the symbol in other colors are displayed as brightness levels of a main color.

If you change the setting from "hollow" to "original" or "shaded" then the color of the display on the configuration computer will deviate from that on the HMI device. This deviation is caused by the different color resolutions. You may also use a graphic object from the "Graphics" object group. The graphic objects are structured by topic and by depth of color in the "WinCC graphics folder."

Flip and rotate

The content of the screen flips around the horizontal or vertical central axis of the icon. The icon can be flipped horizontally, vertically or simultaneously horizontally and vertically.

The screen content rotates around the central point of the icon. The symbol is rotated clockwise in increments of 90, 180, or 270 degrees.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select the mirroring mode at "Orientation > Mirror image".
3. Select the angle of rotation at "Orientation > Rotate".

Fixed aspect ratio

The smaller page of the library object defines the maximum size of the symbol. If you change the size of the library object unproportionally, the symbol is still scaled proportionally. Proceed as follows to configure a fixed aspect ratio:

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select "Form > Fixed aspect ratio".

Behavior during operation

The possibility of operating the mouse is indicated in Runtime by the cursor symbol changing, depending on the configuration. Operation feedback, e.g. by means of a color change, does not take place.

See also

Device dependency of the objects (Page 326)

1.9.2.73 Symbolic I/O field (RT Advanced, Basic Panels, Panels, Comfort Panels)

Application

The "Symbolic I/O field" object can be used to configure a selection list for input and output of texts in Runtime.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Text list: Specifies the text list that is linked to the object.
- Button for selection list: Specifies that the object has a button to open the selection list.

Note

Reports

In reports, symbolic I/O fields only output data. "Output" mode is preset. Properties for configuring the selection of graphics are not available, e.g. "button for selection list".

Mode

The response of the symbolic I/O field is specified in the Inspector window in "Properties > Properties > General > Type".

Mode	Description
"Output"	The symbolic I/O field is used to output values.
"Input"	The symbolic I/O field is used to input values.
"Input/output"	The symbolic I/O field is used for the input and output of values.
"Two states"	The symbolic I/O field is used only to output values and has a maximum of two states. The field switches between two predefined texts. This is used, for example, to visualize the two states of a valve: closed or open.

Note

The behavior possible for the symbolic I/O field depends on the Runtime.

Text list

In the Inspector window, you specify which text list is linked to the symbolic I/O field.

1. In the Inspector window, select "Properties > Properties > General".
2. Under "Contents" open the selection list for "Text list".
3. Select a text list.

Button for selection list

The "Button for selection list" property is used to display a button for opening the selection list.

1. Select "Properties > Properties > Layout" in the Inspector window.
2. Select "Response > Button for selection list".

Note

Basic Panels

The "Button for selection list" option is not available for Basic Panels.

Behavior during operation

The selection list of the symbolic I/O field displays an empty text line if a corresponding entry was not defined in the project. The active state is indicated on the HMI device by changing the color of contents of the symbolic I/O field.

Special features

- The "corner radius" property is not available for a symbolic I/O field on Comfort Panels or RT Advanced. The corner radius that you specify for the "Symbolic I/O field" object in the style editor has no effect in the screen editor for these target systems.

See also

Connecting tags and text lists in the text (Page 126)

Graphic I/O field (Page 379)

Device dependency of the objects (Page 326)

1.9.2.74 Symbolic I/O field (RT Professional)

Application

The "Symbolic I/O field" object can be used to configure a selection list for input and output of texts in Runtime.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can customize the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Text list: Specifies the text list that is linked to the object.

Mode

The response of the symbolic I/O field is specified in the Inspector window in "Properties > Properties > General > Type".

Mode	Description
"Output"	The symbolic I/O field is used to output values.
"Input"	The symbolic I/O field is used to input values.
"Input/output"	The symbolic I/O field is used for the input and output of values.
"Two states"	The symbolic I/O field is used only to output values and has a maximum of two states. The field switches between two predefined texts. This is used, for example, to visualize the two states of a valve: closed or open.

Text list

In the Inspector window you specify which text list is linked to the symbolic I/O field.

1. In the Inspector window, select "Properties > Properties > General":
2. Open the selection list for "Text list".
3. Select a text list.

See also

Graphic I/O field (Page 381)

Configure operator input alarms (Page 827)

Device dependency of the objects (Page 326)

1.9.2.75 System diagnostics view (Basic Panels)

Introduction

The system diagnostics view offers you an overview of all the available devices in your plant. You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

Use

The system diagnostics view enables you to achieve the maximum level of detail of the diagnostics data. A precise diagnosis is possible, as all the available data is displayed. You have the system status of the entire plant at one glance.

Views in the system diagnostics view

A simple system diagnostics view is available on Basic Panels with a device version prior to V13.

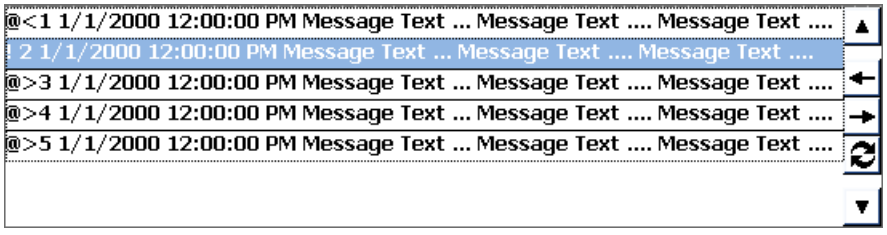

The advanced system diagnostics view is available on Basic Panels with a device version of V13 or later.

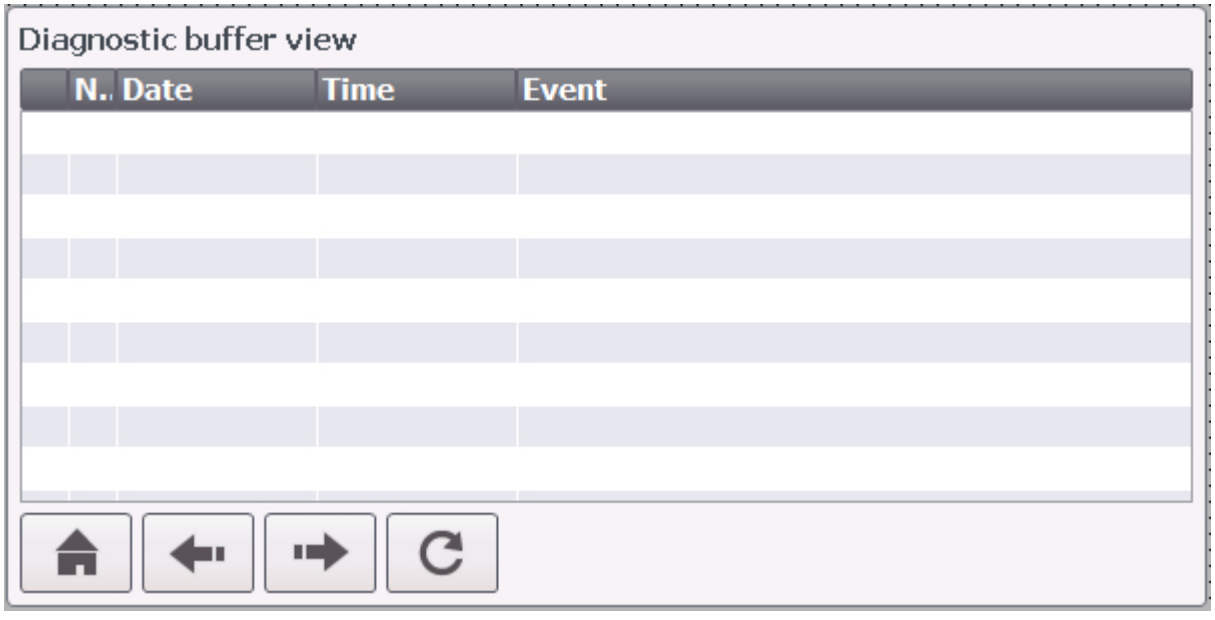

Three different views are available in the system diagnostics view.

- Device view
- Diagnostic buffer view
- Detail view

Device view

The device view of the system diagnostics view shows all the available connections in tabular form. Double-clicking a connection opens the detail view. The device view is only displayed if more than one connection has been created in the "Devices & Networks" editor.

Diagnostic buffer view

In the diagnostic buffer view, the current data from the diagnostic buffer is displayed.

Detail view

The detail view gives detailed information about the selected connection. You cannot sort error texts in the detail view. The detail view is only available if there is an integrated connection to an S7-1200 or S7-1500.

Layout

You change the settings for the position, geometry, style, color, and font of the object in the Inspector window. You can adapt the following properties in particular:

- Lines per entry: Specifies the number of lines that are shown for an entry.

Configuring the system diagnostics view

1. Drag-and-drop the system diagnostics view from the toolbox.
2. In the Inspector window, select "Properties > Layout".
3. Enter a number under "Lines per entry", i.e. 5.
4. Select an authorization for operation in "Properties > Properties > Security".

See also

Basic Panel basics (Page 1821)

Configuring system diagnostics objects (Page 1828)

1.9.2.76 System diagnostics view (RT Advanced, Panels, Comfort Panels)

Introduction

The system diagnostics view offers you an overview of all the available devices in your plant. You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

Application

The system diagnostics view enables you to achieve the maximum level of detail of the diagnostics data. A precise diagnosis is possible, as all the available data is displayed. You have the system status of the entire plant at one glance.

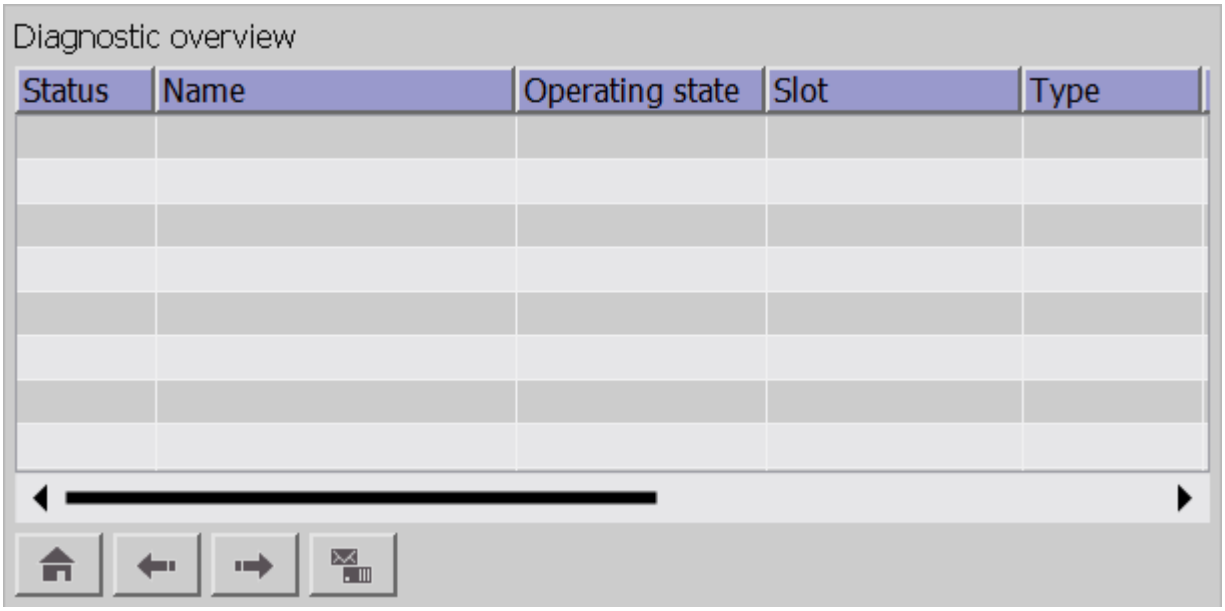

Four different views are available in the system diagnostics view.

- Device view
- Detail view
- Diagnostic buffer view
- Distributed I/O view

Note

The "System diagnostics view" object cannot be used in faceplates.

Layout

You change the settings for the position, geometry, style, color, and font of the object in the Inspector window. You can adapt the following properties in particular:

- Columns: Defines the elements of the device view and detail view.
- Show split view: Specifies that the device view and the detail view are shown simultaneously.

Configuring the system diagnostics view

1. Drag-and-drop the system diagnostics view from the toolbox.
2. You define the start view for the system diagnostics view under "Properties > Properties > General > Mode".
3. In the Inspector window, select "Properties > Properties > Columns".
4. Select the columns that you require in the device view for Runtime, for example, Operating mode, Name, Slot.
5. Select the columns which you need in the detail view for runtime, e.g., Operating mode, Name, Address.
6. Select the columns that you require in the diagnostic buffer view, e.g., Status, Name, Rack.
7. Customize the column headers, if necessary.
8. Enter "IP" for "Address", for example.
   The column header "IP" is displayed in the device view in Runtime.
9. Rename other elements if necessary.
10. Select an authorization for operation in "Properties > Properties > Security".

Showing device view and detail view in a single display

You can split the system diagnostics view and allow the device and detail view to be shown in the same window.

1. In the Inspector window, click "Properties > Properties > Layout".
2. Activate "Show split view".

Symbols in the system diagnostics view

Icon	Function
	Device in operation
	Device cannot be accessed
	Errors in device
	Input/output data are not available
	Device deactivated

Icon	Function
	Maintenance required
	Maintenance recommended
	Superimposed symbol Shows the subordinate status
	Current configuration
	Stop e.g. Update, Boot, Own initialization
	Stop

See also

System diagnostics window (Page 495)

Device dependency of the objects (Page 326)

Panels and Runtime Advanced basics (Page 1824)

Configuring system diagnostics objects (Page 1828)

1.9.2.77 System diagnostics view (RT Professional)

Introduction

The system diagnostics view offers you an overview of all the available devices in your plant. You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

Use

The system diagnostics view enables you to achieve the maximum level of detail of the diagnostics data. A precise diagnosis is possible, as all the available data is displayed. You have the system status of the entire plant at one glance.

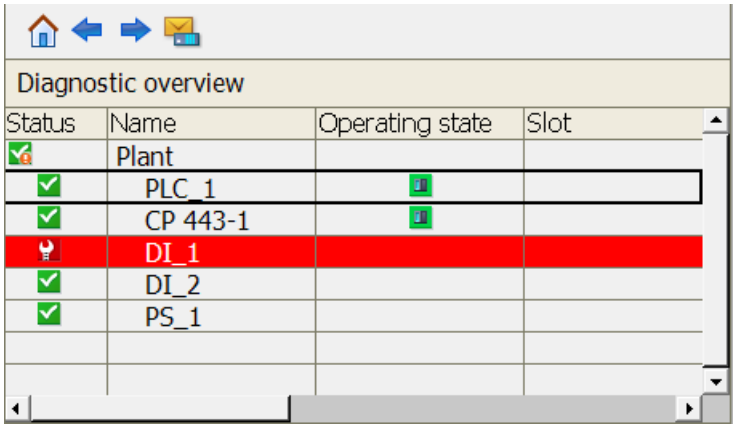

Three different views are available in the system diagnostics view.

- Device view
- Detail view
- Diagnostic buffer view

Note

The "System diagnostics view" object cannot be used in faceplates.

Layout

You change the settings for the position, geometry, style, color, and font of the object in the Inspector window. You can adapt the following properties in particular:

- Columns: Defines the elements of the device view and detail view.
- Show split view: Specifies that the device view and the detail view are shown simultaneously.

Configuring the system diagnostics view

1. Drag-and-drop the system diagnostics view from the toolbox.
2. In the Inspector window, select "Properties > Properties > Columns".
3. Select the columns that you require in the device view for Runtime, for example, status, name, slot.
4. Select the columns which you need in the detail view for runtime, e.g., status, name, address.
5. Customize the column headers, if necessary.

6. Enter "IP" for "Address", for example.
   The column header "IP" is displayed in the device view in runtime.
7. Rename other elements if necessary.
8. Select an authorization for operation in "Properties > Properties > Security".

Showing device view and detail view in a single display

You can split the system diagnostics view and allow the device and detail view to be shown in the same window.

1. In the Inspector window, click "Properties > Properties > Layout".
2. Activate "Show split view".

Single display

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select "Mode > Display type > Single".

Symbols in the system diagnostics view

Icon	Function
	Device in operation
	Device cannot be accessed
	Errors in device
	Input/output data are not available
	Device deactivated
	Maintenance required
	Maintenance recommended
	Superimposed symbol Shows the subordinate status
	Current configuration
	Stop e.g. Update, Boot, Own initialization
	Stop

See also

Runtime Professional basics (Page 1837)

Configuring system diagnostics objects (Page 1840)

Example: System diagnostics with all objects (Page 1844)

1.9.2.78 System diagnostics window (RT Advanced, Panels, Comfort Panels)

Introduction

The system diagnostics window offers you an overview of all the available devices in your plant. You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

The system diagnostics window is only available in the global screen.

Application

The system diagnostics window enables you to achieve the maximum level of detail of the diagnostics data. A precise diagnosis is possible, as all the available data is displayed. You have the system status of the entire plant at one glance.

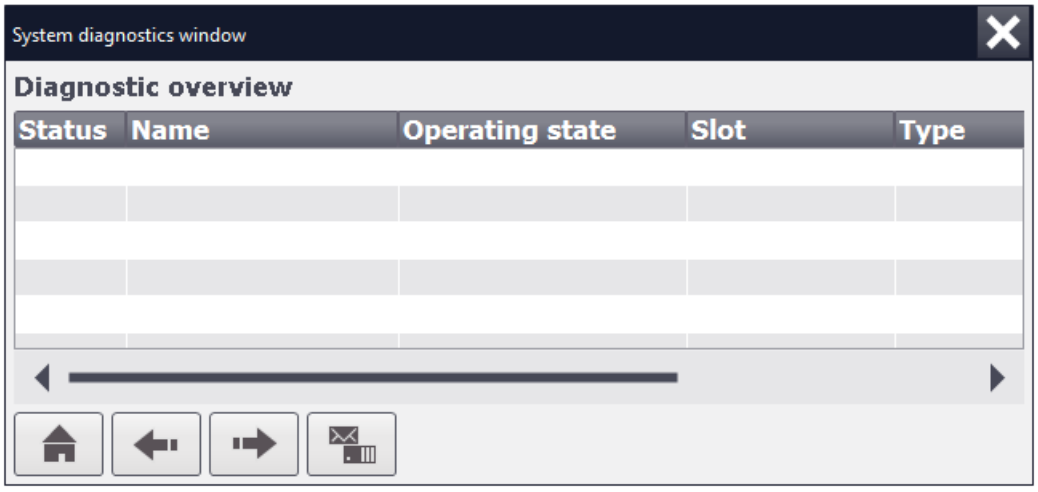

Four different views are available in the system diagnostics window.

- Device view
- Detail view
- Diagnostic buffer view
- Distributed I/O view

Note

The "System diagnostics window" object cannot be used in faceplates.

Layout

You change the settings for the position, geometry, style, color, and font of the object in the Inspector window. You can adapt the following properties in particular:

- Columns: Defines the elements of the device view and detail view.
- Table headers: Specifies the table headers of the view.
- Window: Specifies whether the system diagnostics window can be closed, for example.

Configuring the system diagnostics window

1. Drag and drop the system diagnostics view from the toolbox into the global screen.
2. You define the start view for the system diagnostics window under "Properties > Properties > General > Mode".
3. In the Inspector window, select "Properties > Properties > Columns".
4. Select the columns that you require in the device view for Runtime, e.g., Operating mode, Name, Slot.
5. Select the columns that you require in the detail view for Runtime, e.g., Operating mode, Name, Plant designation.
6. To change a column header, click in the field and (for example) enter "IP" for "Address". The column header "IP" is displayed in the device view in Runtime.
7. Rename other elements if necessary.
8. To be able to close the system diagnostics window in Runtime, select "Properties > Properties > Window > Closable" in the Inspector window.

Icons in the system diagnostics window

Button	Function
	Device in operation
	Device cannot be accessed
	Errors in device
	Device deactivated
	Input/output data are not available
	Maintenance required
	Maintenance recommended
	Superimposed symbol Shows the subordinate status
	Current configuration

Button	Function
	Stop e.g. Update, Boot, Own initialization
	Stop

System diagnostics window: Show wire break error

You have activated wire break diagnostics for a module. The error text is displayed in the system diagnostics window when a wire break error occurs. You can see the error text and possible causes of the error in the module detail view. Output of the channel number of the affected channel is not supported.

See also

System diagnostics view (Page 490)

Device dependency of the objects (Page 326)

Panels and Runtime Advanced basics (Page 1824)

Configuring system diagnostics objects (Page 1828)

1.9.2.79 Table view (RT Professional)

Use

You use the "Table view" object to display the tag values in a table. You can display current, or logged values in the table.

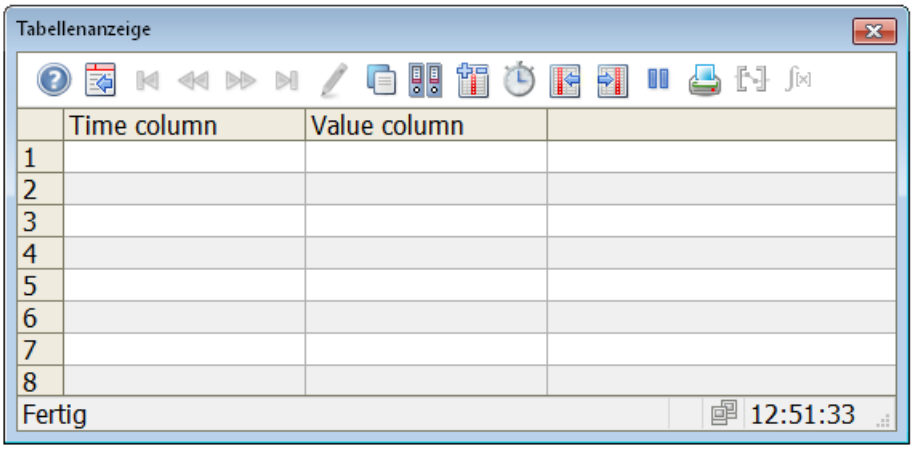

Layout

In the Inspector window, you customize the position, geometry, style, colors and font types of the object. You can adapt the following properties in particular:

- Configuring data source and columns
- Configuring a table
- Window settings
- Persistence
- Toolbar

Configuring data source and columns

1. Open "Properties > Properties > Value column".
2. Select the log tag or compressed log tag that supplies the column with values under "Data source".
3. Under "Time column", select the column of the table view in which the data source is to be displayed.
4. Select the time range of the table under "Properties > Properties > Time column > Time range":
   – "Time span": You define the time range using a starting time and a following time span.
   – "End time": You define the time range using a starting time and an end time.
   – "Measuring points": You define the time range using a starting time and a number of measuring points.
5. Use "Format > Update" to specify whether the table displays the values statically or the values are refreshed.
6. Under "Style" specify the format for displaying the time and the alignment within the column.

Configuring a table

Under "Properties > Properties > Table", you specify whether the columns of the table have a title and whether the grid lines of the table are displayed.

Window settings

Under "Properties > Properties > Layout > Window" you define whether the user can change the size of the window and close the window in Runtime.

Persistence

Under "Properties > Properties > Security > Persistence", define how long a user's changes to the trend view in runtime are retained.

- "Allow persistence"
  Users can change settings. Changes are retained until the next screen change.
- "Persistence until logoff"
  User group changes are retained until logoff.
- "Persistence until next loading"
  User group changes are retained until the project is updated on the HMI device.

Toolbar

You define the operator controls of the table view in Runtime in the "Properties > Properties > Toolbar" inspector window. The following operator controls are available for the table view:

Button	Name	Function
	Online help	Opens the online help.
	Open configuration	Opens the configuration dialog.
	First data record	Browse the log: Shows the tag values starting with the first logged value.
	Previous data record	Browse the log: Shows the tag values in the previous time interval.
	Next data record	Browse the log: Shows the tag values in the subsequent time interval.
	Last data record	Browse the log: Shows the tag values up to the last logged value.
	Edit data	Allows the editing of data in any table field that is opened when the user double-clicks it.
	Tag selection	Opens the dialog for selecting the log and tags.
	Select column	Opens the dialog for setting the visibility of columns.
	Time range selection	Opens the dialog for setting the time range displayed in the table view.
	Previous column	Displays the previous column in the foreground
	Next column	Displays the next column in the foreground
	Start/end update	Stops and starts the column update. The values are buffered and updated as soon as you start column update again.

Button	Name	Function
	Print log	Starts printing the columns displayed in the table view. You define the print job under "General > Print > Print job".
	Selection of the statistics area	Enables you to define a time range for which statistical values are determined.
	Statistics window	Opens a statistics window to display the minimum, maximum, means, and standard deviation for the selected time range and the selected column.

Dynamizing the value column in the table view (RT Professional)

Note the following information about dynamizing the properties in the value column of a table view:

- The "ValueColumnName" property specifies the name of the selected value column. The "ValueColumnName" attribute can be dynamized with the "ValueColumnRename" attribute. The data type is STRING.
- The "ValueColumnRename" property changes the name of the value column referenced with the "ValueColumnIndex" attribute. The attribute can be dynamized with the "ValueColumnRename" name. With "ValueColumnRename" you also dynamize the "ValueColumnName" attribute. The data type is STRING.
- The "TimeColumnName" property specifies the name of the selected time column. The "TimeColumnName" attribute can be dynamized with the "TimeColumnRename" attribute. The data type is STRING.
- The "TimeColumnRename" property changes the name of the time column referenced with the "TimeColumnIndex" attribute. The attribute can be dynamized with the "TimeColumnRename" name. With "TimeColumnRename" you also dynamize the "TimeColumnName" attribute. The data type is STRING.

See also

f(t) trend view (Page 367)

f(x) trend view (Page 374)

Configuring the Table View (Page 711)

Value table (Page 505)

Configuring persistence (Page 2264)

Device dependency of the objects (Page 326)

Operating table view in Runtime (Page 744)

1.9.2.80 Text field (RT Professional, RT Advanced, Panels, Comfort Panels)

Application

The "Text field" is a closed object which you can fill with a color.

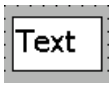

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Text: Specifies the text for the text field.
- Size of text field: Defines whether the size of the object is adapted to the space required by the largest list entry.

Text

Specify the text for the text field in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Enter a text.
   For texts over several lines you can set a line break by pressing the key combination <Shift + Enter>.

Size of text field

In the Inspector window, you can define whether the size of the object is adapted to the space required by the largest list entry.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Activate "Resize > Fit to contents".

See also

Connecting tags and text lists in the text (Page 126)

Device dependency of the objects (Page 326)

1.9.2.81 T-piece (RT Professional)

Application

You use the "T piece" object to reproduce a T-shaped pipe junction. To reproduce a pipe system, connect the "T piece" object to other objects, e.g. pipe or pipe bend.

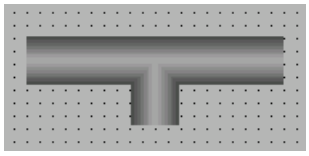

Layout

Customize the object position and color in the Inspector window. You can adapt the following properties in particular:

Rotation: Specifies the angle of rotation in degrees of the T piece.

Rotation

You specify by how many degrees the "T piece" is rotated under "Properties > Properties > Layout > Rotation" in the Inspector window. The value entered is automatically rounded to a multiple of 90.

- 0: the "leg" of the T piece is pointing downward
- 90: the "leg" of the T piece is pointing to the left
- 180: the "leg" of the T piece is pointing upward
- 270: the "leg" of the T piece is pointing to the right

See also

Pipe (Page 469)

Pipe elbow (Page 470)

Double T-piece (Page 351)

Device dependency of the objects (Page 326)

1.9.2.82 Clock

Application

The "Clock" object displays the date and time.

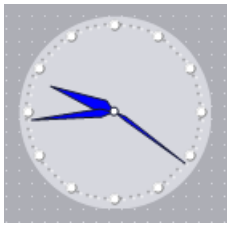

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Analog display: Specifies whether the clock is shown as an analog clock or digital clock.
- Display clock dial: Specifies whether hour marks of the analog clock will be displayed.
- Width and length of hands: Specifies the width and length of the hands.

Analog display

In the Inspector window you can specify whether the clock is displayed as an analog or digital clock. The digital clock shows both the time and the date. The display format depends on the language set on the HMI device.

1. In the Inspector window, select "Properties > Properties > General":
2. Activate "Analog".

Display dial

In the Inspector window you can specify whether hour marks will be displayed.

1. In the Inspector window, select "Properties > Properties > General":
2. Activate "Analog".
3. Activate "Show dial".

Width and length of hands

The width of the second hand, minute hand and hour hand is set for the analog display.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter values for "width" and "length".
   The values for the length are interpreted as a percentage of the size of the clock. The value for the length of the second hand influences the radius of the dial at the same time.

See also

Device dependency of the objects (Page 326)

1.9.2.83 Connector (RT Professional)

Application

You use the "Connector" object to interconnect several objects using a line. You can also attach multiple connectors to one another.

There are two types of connector:

- Automatic: The connector consists of horizontal and vertical parts.

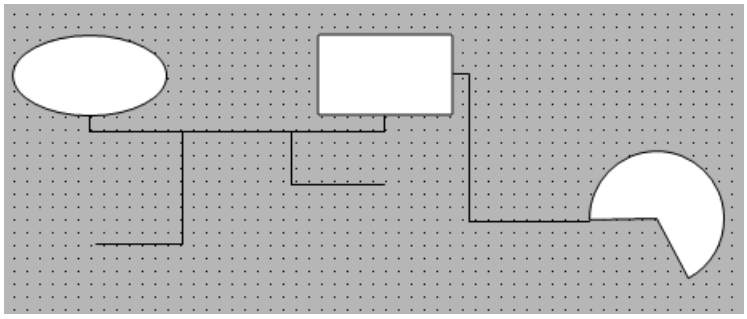

- Single: The connecting points are connected by a straight line.

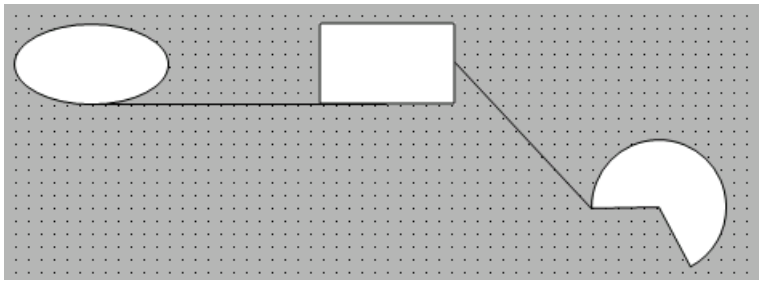

Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Line style
- Line start and end

Connecting objects

To connect objects, join the ends of the connectors to the connecting points on the objects.

1. Click the "Connector" object in the "Tools" task card.
2. Click on an object in the screen. The connector is connected to the first object.
3. Click on another object in the screen.
   Both objects are connected to each other.

See also

Device dependency of the objects (Page 326)

1.9.2.84 Value table (RT Professional)

Application

You use the "Value table" object to show evaluated data and statistics in a table.

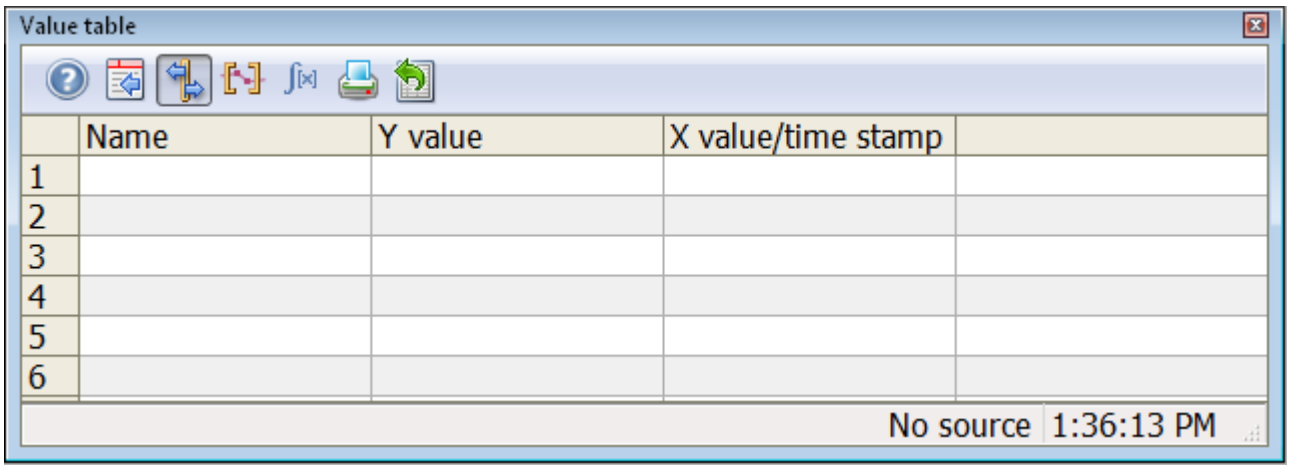

- f(t) trend view
- f(x) trend view
- Table view

Layout

In the Inspector window, you customize the position, geometry, style, colors and font types of the object. You can adapt the following properties in particular:

- Data source for displaying the values
- Mode
- Toolbar

Data source for displaying the values

You define which values are displayed in the value table in the "Properties > Properties > General > Data source" inspector window.

Select among the following objects:

- f(t) trend view
- f(x) trend view
- Table view

By default, the configuration for the display format, the format is taken from the connected control. The size, value range and zoom factor of the control are taken into account to display the optimum number of decimal places. You can configure the display formats for individual values in the Inspector window of the value table yourself, for example, to show a precise number of decimal places.

Mode

You define the mode in the "Properties > Properties > General > Mode" inspector window. You have a choice of three different types depending on the data source.

- The ruler window shows the coordinate values of the trends on the ruler or values of a selected line in the table.
- The statistics area window shows the values of the lower limit and upper limit of the trends between two rulers or the selected area in the table. The statistics area window is not provided for the object f(x) trend view.
- The statistics window shows the statistical evaluation of the trends between two rulers or the selected values in the table. The statistics window is not provided for the object "f(x) trend view".

Toolbar

You define the control elements of the value table in Runtime in the "Properties > Properties > Toolbar - Toolbar - Buttons" inspector window. The following control elements are available for the recipe data:

Button	Brief description	Description
	Tooltip	Calls help for the WinCC value table.
	Configuration dialog box	Opens the configuration dialog box in which you change the properties of the value table.
	Ruler window	You query the coordinate points of a trend with the button. The trend data are displayed in the ruler window.
	Define statistics area	Enables you to define a time range for which statistics values are determined.
	Calculating statistics	The button shows the statistical values in the statistics window. The displayed values refer to a selected trend with the configured calculation time period. The button can only be pressed if a statistics window is connected with an f(t) trend view.

	Printing	Start the print-out of the values shown in the table. You define the print job to be used in the "Properties > Properties > General > Print job" inspector window.
	Exporting data	This button is used for exporting all or the selected runtime data into a "CSV" file. If the "Properties > Properties > Data export > Show dialog" inspector window is activated, a dialog box opens in runtime. The dialog box offers you settings for the export and starting the export. With the appropriate authorizations you can also select the file and the directory for the export. If no dialog box is displayed, the data export to the preset file starts immediately.
	Customized 1	Shows the first key function created by the user. The function of the button is user-defined.

See also

Table view (Page 497)

f(t) trend view (Page 367)

f(x) trend view (Page 374)

Value table basics (Page 723)

Configuring the value table (Page 724)

Configuring persistence (Page 2264)

Device dependency of the objects (Page 326)

1.9.2.85 Effective range name (Panels, Comfort Panels)

Use

The "Effective range name" object is an operator control for some SIMATIC Mobile Panels. The "Effective range name" object shows the names of the effective range in which the HMI device is located and also the logon status.

Layout

Symbol	Meaning	Description	Logon status
MixingAxisControl	Acknowledgement button has no effect	Within the "MixingAxisControl" effective range	It is not possible to log on to the effective range
MixingAxisControl	Acknowledgement button functions	Within the "MixingAxisControl" effective range	Logged on to effective range

Symbol	Meaning	Description	Logon status
MixingAxisControl	Acknowledgement button has no effect	Within the "MixingAxisControl" effective range	Logon to the effective range is rejected because a different HMI device is already logged on. Note: When using the "Override" mode: Although no other HMI device is still logged on to the effective range, logon is rejected because the override switch is still set.
	Acknowledgement button has no effect	Outside of each effective range	Note: You can only log on within the effective range

In the Inspector window "Properties > Properties > Security", assign an authorization to log on to a machine in Runtime.

Logon requirements

- The HMI device is located within the effective range.
- No HMI device is logged on to this effective range.

Logon procedure

1. Click on the "Effective range name" object.
   The "Effective range logon" dialog box opens.
2. Read the ID from the machine or plant to be operated.
3. Enter the ID you read.
4. Confirm your entries by clicking "OK".

Result

If the ID and effective range match, the HMI device is logged on to the effective range. Successful logon is signaled by an icon on green background. No other HMI device can log on to this effective range.

Requirement for logging off

The HMI device is logged on to this effective range.

Logoff procedure

Before you leave the effective range or runtime, you must log off from the effective range first. If you leave the effective range without logging off, a warning appears.

1. Click on the "Effective range name" object. The "Effective range logoff" dialog box opens.
2. Confirm logoff with "Yes".

Result

The HMI device is no longer logged into the effective range. Other HMI devices can now log on to this effective range.

Note

Do not deactivate the "Override" mode again unless you logon a different HMI device.

See also

Configuring the effective range name (Page 1786)

Effective range signal (Page 511)

WLAN reception (Page 512)

Zone name (Page 516)

Zone signal (Page 517)

Charge condition (Page 410)

Device dependency of the objects (Page 326)

1.9.2.86 Effective range name (RFID) (Panels, Comfort Panels)

Use

The "Effective range name (RFID)" object is an operator control for some SIMATIC Mobile Panels. The "Effective range name (RFID)" object shows the name of the effective range (RFID) in which the HMI device is located and also the logon status.

Note

RFID tags in the plant area

RFID tags may only be attached in demarcated areas which are secured by additional protective measures, e.g. protective fencing.

Layout

Symbol	Meaning	Description	Logon status
Scan	Acknowledgement button has no effect	Within the effective range (RFID) "MixingAxisControl"	It is not possible to log on to the effective range (RFID)
Scanning...	Acknowledgement button functions	Within the effective range (RFID) "MixingAxisControl"	Logon to effective range (RFID) running
MixingAxisControl	Acknowledgement button functions	Within the effective range (RFID) "MixingAxisControl"	Logged on to the effective range (RFID)

In the "Properties > Properties > Security" category of the Inspector window, you can assign an authorization for the user group that logs on to the effective range (RFID) during runtime.

Logon requirements

- The HMI device is located within the effective range (RFID).
- The machine is demarcated by additional protective measures.
- No HMI device is logged on to this effective range (RFID).

Logon procedure

1. Click on the "Effective range name (RFID)" object.
   The "Effective range logon" dialog box opens.
2. Read the ID from the machine or plant to be operated.
3. Enter the ID you read.
4. Confirm your entries by clicking "OK".

Result

If the ID and effective range (RFID) match, the HMI device is logged on to the machine. Successful logon is signaled by an icon on green background. No other HMI device can log on to this effective range (RFID) or the machine.

Logoff requirements

The HMI device is logged on to this effective range (RFID).

Logoff procedure

Before you exit the effective range (RFID) or runtime, you must first log off from the effective range (RFID). If you exit the effective range (RFID) without logging off, a local rampdown is executed.

1. Click on the "Effective range name (RFID)" object. The "Effective range (RFID) - logoff" dialog box opens.
2. Confirm logoff with "Yes".

Result

The HMI device is no longer logged into the effective range (RFID) or machine. Other HMI devices can now log on to this effective range (RFID).

See also

WLAN reception (Page 512)

Charge condition (Page 410)

Configuring the effective range name (RFID) (Page 1790)

Device dependency of the objects (Page 326)

1.9.2.87 Effective range signal (Panels, Comfort Panels)

Application

The "Effective range signal" object is a control element for some SIMATIC Mobile Panels. The "Effective range signal" object indicates how accurately the Mobile Panel Wireless is positioned within an effective range.

Layout

Symbol	Meaning
	Within an effective range
	When leaving an effective range When entering an effective range
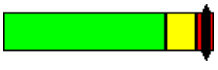	Outside of each effective range
	Override function is activated: Within a special security range of the effective range

The "Effective range name" object indicates which effective range this concerns.

Override function

An effective range, e.g. "robots" is formed outwardly through transponders. However, within a special protection zone, e.g. "robot cells", the effective range is not formed by transponders, but by alternative restrictions e.g. by railings and a door.

The operator initially logs on to the effective range. An operator who enters the restricted and secure zone "robot cells" must press an additional switch to activate the override function:

- The effective range is allocated and can no longer be established by the transponder.
- Therefore, the quality does not play a role and is no longer displayed.
- The HMI device remains logged into the effective range.
- Other HMI devices can no longer logon to this effective range.

Calculating the signal quality

The quality within an effective range depends, as follows, on the measured distance:

- In the center of the effective range the quality is 100%.
- On the transponder and on the border of the effective range, the quality is 0%.

Operation

The object is for display only and cannot be controlled by the operator.

See also

Effective range name (Page 507)

WLAN reception (Page 512)

Zone name (Page 516)

Zone signal (Page 517)

Charge condition (Page 410)

Configuring additional Mobile Wireless objects (Page 1787)

Device dependency of the objects (Page 326)

1.9.2.88 WLAN reception (Panels, Comfort Panels)

Application

The "WLAN reception" object is a control element for some SIMATIC Mobile Panels. The "WLAN reception" object displays the signal strength of the WLAN wireless connection. In addition, the signal strength is measured and displayed by means of 5 bars.

Layout

Symbol	Meaning	Signal strength
	No wireless connection	No signal
	Very poor wireless connection	=<20 %
	Poor wireless connection	>20 % =<40 %
	Wireless connection OK	>40 % =<60 %
	Good wireless connection	>60 % =<80 %
	Very good wireless connection	>80 %

Operation

The object is for display only and cannot be controlled by the operator.

See also

Effective range name (Page 507)

Effective range name (RFID) (Page 509)

Effective range signal (Page 511)

Zone name (Page 516)

Zone signal (Page 517)

Charge condition (Page 410)

Configuring additional Mobile Wireless objects (Page 1787)

Device dependency of the objects (Page 326)

1.9.2.89 Gauge

Use

The "Gauge" object shows numeric values in the form of an analog gauge. For example, a glance in Runtime is enough to note that the boiler pressure is in the normal range.

Note

The gauge is for display only and cannot be controlled by the operator.

Configure system functions that are performed at value changes in the tag properties at the "Value change" event and not at the "Change" event of the gauge. If your project is to comply with Good Manufacturing Practices, and the system function performs a GMP-relevant action. e.g. increasing a GMP-relevant tag, every value change of the tag used at the gauge is interpreted as a user action.

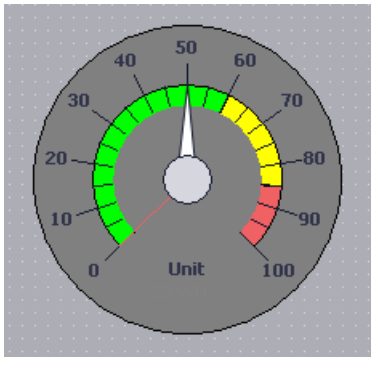

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Display peak value pointer: Specifies whether the actual measurement range is indicated with a slave pointer.
- Maximum Value and Minimum Value: Specifies the top and bottom values of the scale.
- Start value of the danger range and start value of the warning range: Specifies the scale value from which the danger range and the warning range start.
- Display normal range: Specifies whether the normal range is shown in color on the scale.
- Color of individual ranges: Different operating modes, such as normal range, warning range and danger range, are shown in different colors so that the operator can distinguish them easily.

Note

The use of many differently sized "Gauge" objects can reduce the performance in Runtime. With "Gauge", avoid minimally different heights and widths, for example, 48 pixels, 49 pixels, 51 pixels, etc. Use the same sizes instead.

Display peak value

The "Display peak value" property can be used to activate a marker function for the maximum and minimum pointer movement in Runtime. The actual measurement range is shown with a slave pointer.

1. In the Inspector window, activate "Properties > Properties > Appearance > Show peak values".

Maximum Value and Minimum Value

You can set the top and bottom end values of the scale in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Enter a number under "Process > Maximum scale value" and for "Minimum scale value". If you select a tag as the end value of the scale, the number will be no longer available.

Note

Ensure that the minimum value is always less than the maximum value, especially if you make the minimum value or maximum value dynamic through a tag. If the minimum value is greater than the maximum value, no adjustment is made to the scale end values.

Start value of the danger range and start value of the warning range

Define the scale value at which the danger range and the warning range start in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Limits/Ranges".
2. Enter the start value for the danger range in "Upper 2" and the start value for the warning range in "Upper 1".
3. Select "start value of danger range" and "start value of warning range" to display the ranges on the scale.

Normal range visible

In the Inspector window, define whether to show the normal range in color.

1. In the Inspector window, select "Properties > Properties > Limits/Ranges".
2. Select "Normal".

Color of individual ranges

You can display the normal range, the danger range and the warning range in different colors. Define the color in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Limits/Ranges".
2. Select a color from the color palette for the three areas.

Note

The following restrictions apply for HMI devices with Windows CE up to version 5.0:

- The three ranges (normal / warning / danger) are not visualized in different colors in Runtime.

Note

When you select the option "Show ranges from tag", you can display up to five ranges in the gauge whose values are specified by means of a process tag. You define the values for the areas with a process tag, which you interconnect with the screen object.

The option "Show ranges from tag" is available for Comfort Panels, KTP Mobile Panels and RT Advanced.

See also

Device dependency of the objects (Page 326)

1.9.2.90 Zone name (Panels, Comfort Panels)

Application

The "Zone name" object is a control element for some SIMATIC Mobile Panels. The "Zone name" object shows the names of the zone in which the HMI device is located.

Layout

Symbol	Meaning	Description
	The "On entry" and "On exit" events of the zone and the zone ID can be used in the configuration.	Within the "MixingPlant" zone
	–	Outside of each zone

Procedure for "On entry" event

1. In the "Zones" or "Zones and effective ranges" editor, configure the function "ActivateScreen" on the "On entry" event.

Procedure with zone ID

1. Select an internal tag as the "Zone ID / Connection point ID" in the "Runtime settings > General" editor.
2. Animate the visibility of an object, e.g. "I/O field" via the internal tag.

Operation

The object is for display only and cannot be operated.

See also

Effective range name (Page 507)

Effective range signal (Page 511)

WLAN reception (Page 512)

Zone signal (Page 517)

Charge condition (Page 410)

Device dependency of the objects (Page 326)

1.9.2.91 Zone signal (Panels, Comfort Panels)

Application

The "Zone signal" object is a control element for some SIMATIC Mobile Panels. The "Zone signal" object indicates how well the Mobile Panel is still in a zone. Compared to the "WLAN reception" object, it is not the signal strength that is measured, it is calculated from the distance.

Layout

Symbol	Meaning	Quality
	Within one zone	>=15%
	When leaving a zone When entering a zone	<15% and >0%
	Outside of each zone	= 0%

The "Zone name" object indicates which zone this concerns.

Calculating the signal quality

The quality of the signal within a zone depends, as follows, on the measured distance:

- In the centre of the zone the quality is 100%.
- To the rear towards the transponder and to the front away from the transponder, the quality starts to diminish linearly.
- On the transponder and on the border of the zone, the quality is 0%.

Operation

The object is for display only and cannot be operated.

See also

Effective range name (Page 507)

Effective range signal (Page 511)

WLAN reception (Page 512)

Zone name (Page 516)

Charge condition (Page 410)

Device dependency of the objects (Page 326)

Working with tags 2

2.1 Basics

2.1.1 Basics of tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Process values are forwarded in runtime using tags. Process values are data which is stored in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. Define external tags for processing the process values in WinCC.

WinCC works with two types of tag:

- External tags
- Internal tags

The external tags form the link between WinCC and the automation systems. The values of external tags correspond to the process values from the memory of an automation system. The value of an external tag is determined by reading the process value from the memory of the automation system. It is also possible to rewrite a process value in the memory of the automation system.

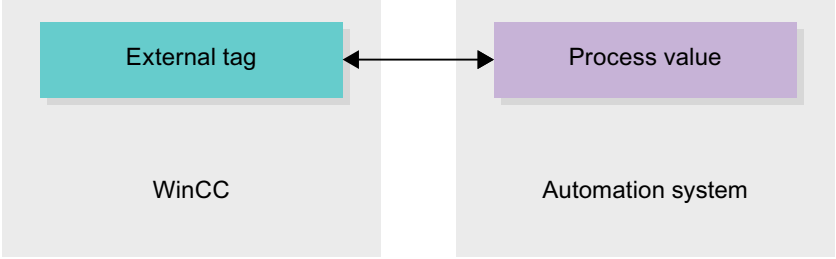

Internal tags do not have a process link and only convey values within the WinCC. The tag values are only available as long as runtime is running.

Tags in WinCC

For external tags, the properties of the tag are used to define the connection that the WinCC uses to communicate with the automation system and form of data exchange.

Tags that are not supplied with values by the process - the internal tags - are not connected to the automation system. In the tag's "Connection" property, this is identified by the "Internal tag" entry.

You can create tags in different tag tables for greater clarity. You then directly access the individual tag tables in the "HMI tags" node in the project tree. The tags from all tag tables can be displayed with the help of the table "Show all tags".

With structures you bundle a number of different tags that form one logical unit. Structures are project-associated data and are available for all HMI devices of the project. You use the "Types" editor in the project library to create and edit a structure.

See also

Internal Tags (Page 528)

External tags (Page 524)

Cycle basics (Page 593)

Basics on user data types (Page 585)

Basics on arrays (Page 575)

Overview of HMI tag tables (Page 520)

Addressing external tags (Page 526)

Creating external tags (Page 531)

Cycle basics (Page 596)

Basic principles for data logging (Page 616)

External tags (Page 522)

Internal Tags (Page 529)

Outputting tag values in screens (Basic) (Page 676)

2.1.2 Overview of HMI tag tables (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

HMI tag tables contain the definitions of the HMI tags that apply across all devices. A tag table is created automatically for each HMI device created in the project.

In the project tree there is an "HMI tags" folder for each HMI device. The following tables can be contained in this folder:

- Default tag table
- User-defined tag tables
- Table of all tags

In the project tree you can create additional tag tables in the HMI tags folder and use these to sort and group tags and constants. You can move tags to a different tag table using a drag-and-drop operation or with the help of the "Tag table" field. Activate the "Tags table" field using the shortcut menu of the column headings.

Default tag table

There is one default tag table for each HMI device of the project. It cannot be deleted or moved. The default tag table contains HMI tags and, depending on the HMI device, also system tags. You can declare all HMI tags in the standard tags table or, as necessary, additional user-defined tables of tags.

User-defined tag tables

You can create multiple user-defined tag tables for each HMI device in order to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. To group tag tables, create additional subfolders in the HMI tags folder.

All tags

The "All tags" table shows an overview of all HMI tags and system tags of the HMI device in question. This table cannot be deleted, renamed or moved. This table also contains the "Tags table" column, which indicates the tag table of where a tag is included. Using the "Tags table" field, the assignment of a tag to a tags table can be changed.

With devices for Runtime Professional, the table "All tags" contain an additional tab "System tags". The system tags are created by the system and used for internal management of the project. The names of the system tags begin with the "@" character. System tags cannot be deleted or renamed. You can evaluate the value of a system tag, but cannot modify it.

Additional tables

The following tables are also available in an HMI tag table:

- Discrete alarms
- Analog alarms
- Logging tags

With the help of these tables you configure alarms and logging tags for the currently selected HMI tag.

Discrete alarms table

In the "Discrete alarms" table, you configure discrete alarms to the HMI tag selected in the HMI tag table. When you configure a discrete alarm, multiple selection in the HMI tag table is not possible. You configure the discrete alarms for each HMI tag separately.

Analog alarms table

In the "Analog alarms" table, you configure analog alarms to the HMI tag selected in the HMI tag table. When you configure an analog alarm, multiple selection in the HMI tag table is not possible. You configure the analog alarms for each HMI tag separately.

Logging tags table

In the "Logging tags" table, you configure logging tags to the HMI tag selected in the HMI tag table. When you configure a logging tag, multiple selection in the HMI tag table is not possible. You configure the logging tags for each HMI tag separately. The "Logging tags" table is only available if the HMI device used supports logging.
If WinCC Runtime Professional is used, you can also assign several log tags to a tag. With the other HMI devices, you can only assign one log tag to a tag.

See also

Basics of tags (Page 519)

2.1.3 External tags (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

External tags allow communication (data exchange) between the components of an automation system, for example, between an HMI device and a PLC.

Principle

An external tag is the image of a defined memory location in the PLC. You have read and write access to this storage location from both the HMI device and from the PLC.

Since external tags are the image of a storage location in the PLC, the applicable data types depend on the PLC which is connected to the HMI device.

If you write a PLC control program in STEP 7, the PLC tags created in the control program will be added to the PLC tag table. If you want to connect an external tag to a PLC tag, access the PLC tags directly via the PLC tag table and connect them to the external tag.

Data types

All the data types which are available at the connected PLC are available at an external tag in WinCC. Information about data types which are available for connection to other PLCs can be found in the documentation about the respective communication drivers.

See "AUTOHOTSPOT" for additional information.

Note

As well as external tags, area pointers are also available for communication between the HMI device and PLC. You can set up and enable the area indicators in the "Connections" editor.

Central tag management in STEP 7

You can connect also connect DB instances of user-defined PLC data types (UDT) to the HMI tags.

The PLC data type and the corresponding DB instances are created and updated centrally in STEP 7. In WinCC, you can use the following sources as PLC tag (DB instances):

- Data block elements that use a UDT as data type
- Data block instances of a UDT

The data type is taken from STEP 7 and is not converted to an HMI data type. The access type is always "Symbolic access". Depending on the release for WinCC in STEP 7, elements and structured elements of the PLC data type are applied to WinCC. Elements of a structured UDT are applied and displayed in the PLC tag table if the instance-specific properties "Visible in HMI" and "Accessible from HMI" have been set.

Note

Accessing PLC data types

Access to PLC data types is only available in conjunction with SIMATIC S7-1200 or S7-1500.

Synchronization with PLC tags

A variety of options for synchronizing external tags with the PLC tags are available in the Runtime settings under "Settings for tags".

When you perform synchronization, you have the option of automatically applying the tag names of the PLC to external tags and reconnecting the existing tags.

The generated tag name is derived from the position of the data value in the hierarchical structure of the data block.

Update of tag values

For external tags, the current tag values are transmitted in Runtime via the communication connection between WinCC and the connected automation systems and then saved in the runtime memory. Next, the tag value will be updated to the set cycle time. For use in the runtime project, WinCC accesses tag values in the runtime memory that were read from the PLC at the previous cycle time. As a result, the value in the PLC can already change while the value from the runtime memory is being processed.

Note

PLC array elements in conjunction with S7-1200 or S7-1500

The index of the PLC array elements can begin with any number. In WinCC, indexing always starts with 0.

A PLC tag "Array [1..3] of Int", for example, is mapped to "Array [0..2] of Int" in WinCC.

When you access an array in a script, pay attention to the correct indexing.

See also

Basics of tags (Page 519)

2.1.4 External tags (RT Professional)

Introduction

External tags allow communication (data exchange) between the components of an automation system, for example, between an HMI device and a PLC.

Principle

An external tag is the image of a defined memory location in the PLC. You have read and write access to this storage location from both the HMI device and from the PLC.

Since external tags are the image of a storage location in the PLC, the applicable data types depend on the PLC which is connected to the HMI device.

If you write a PLC control program in STEP 7, the PLC tags created in the control program will be added to the PLC tag table. If you want to connect an external tag to a PLC tag, access the PLC tags directly via the PLC tag table and connect them to the external tag.

Data types

All the data types which are available at the connected PLC are available at an external tag in WinCC. If the data type of the PLC tag is not available in WinCC, a compatible data type is automatically used at the HMI tag. If you use PLC data types, the data type is adopted by WinCC. You can change the data type at the HMI tag, if necessary.

Central tag management in STEP 7

You can connect also connect DB instances of user-defined PLC data types (UDT) to the HMI tags.

The PLC data type and the corresponding DB instances are created and updated centrally in STEP 7. In WinCC, you can use the following sources as PLC tag (DB instances):

- Data block elements that use a UDT as data type
- Data block instances of a UDT

The data type is taken from STEP 7 and is not converted to an HMI data type. The access type is always "Symbolic access". Depending on the release for WinCC in STEP 7, elements and structured

elements of the PLC data type are also transferred to WinCC.

Note

Accessing PLC data types

Access to PLC data types is only available in conjunction with SIMATIC S7-1200 or S7-1500.

Synchronization with PLC tags

A variety of options for synchronizing external tags with the PLC tags are available in the Runtime settings under "Settings for tags".

When you perform synchronization, you have the option of automatically applying the tag names of the PLC to external tags and reconnecting the existing tags.

The generated tag name is derived from the position of the data value in the hierarchical structure of the data block.

Update of tag values

For external tags, the current tag values are transmitted in Runtime via the communication connection between WinCC and the connected automation systems and then saved in the runtime memory. Next, the tag value will be updated to the set cycle time. For use in the runtime project, WinCC accesses tag values in the runtime memory that were read from the PLC at the previous cycle time. As a result, the value in the PLC can already change while the value from the runtime memory is being processed.

You can read out the process values directly from the PLC memory, if necessary. Use the GetTag functions for this purpose. For more detailed information, refer to the documentation for C functions under "AUTOHOTSPOT".

Note

PLC array elements in conjunction with S7-1200 or S7-1500

The index of the PLC array elements can begin with any number. In WinCC, indexing always starts with 0.

A PLC tag "Array [1..3] of Int", for example, is mapped to "Array [0..2] of Int" in WinCC.

When you access an array in a script, pay attention to the correct indexing.

Displaying the PLC tag comment

If you want to see the comments for the connected PLC tag, select the column "Source comment" in the HMI tag table. The source comment is not available in runtime.

See also

Basics of tags (Page 519)

Adapting the data type of a tag (Page 536)

Synchronizing tags (Page 546)

Settings for synchronization (Page 544)

User-defined PLC data types (UDT) (Page 611)

2.1.5 Addressing external tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The options for addressing external tags depend on the type of connection between WinCC and the PLC in question. A distinction must be made between the following connection types:

- Integrated connection
  Connections of devices which are within a project and were created with the "Devices & Networks" editor are referred to as integrated connections.
- Non-integrated connection
  Connections of devices which were created with the "Connections" editor are referred to as non-integrated connections. It is not necessary that all of the devices be within a single project.

The connection type can also be recognized by its icon.

	Integrated connection
	Non-integrated connection

You can find additional information in the section "AUTOHOTSPOT".

Addressing with integrated connections

An integrated connection offers the advantage that you can address a tag both symbolically and absolutely.

For symbolic addressing, you select the PLC tag via its name and connect it to the HMI tag. The valid data type for the HMI tag is automatically selected by the system. You have to distinguish between the following cases when you address elements in data blocks:

Symbolic addressing of data blocks with optimized access and standard access:

During the symbolic addressing of a data block with optimized access and standard access, the address of an element in the data block is dynamically assigned and is automatically adopted in the HMI tag in the event of a change. You do not need to compile the connected data block or the WinCC project for this step.
For data blocks with optimized access, only symbolic addressing is available.

For symbolic addressing of elements in a data block, you only need to recompile and reload the WinCC project in case of the following changes:

- If the name or the data type of the linked data block element or global PLC tag has changed.
- If the name or the data type of the higher level structure node of a linked element in the data block element or global PLC tag has changed.
- If the name of the connected data block has changed.

Symbolic addressing is currently available with the following PLCs:

- SIMATIC S7-1200
- SIMATIC S7-1500

- SIMATIC ET 200 CPU
- SIMATIC S7-1500 software controller

Symbolic addressing is also available if you have an integrated link.

You can also use absolute addressing with an integrated connection. You have to use absolute addressing for PLC tags from a SIMATIC S7-300/400 PLC. If you have connected an HMI tag with a PLC tag and the address of the PLC tag changes, you only have to recompile the control program to update the new address in WinCC. Then you recompile the WinCC project and load it onto the HMI device.

In WinCC, symbolic addressing is the default method. To change the default setting, select the menu command "Options > Settings". Select "Visualization > Tags" in the "Settings" dialog. If required, disable the "Symbolic access" option.

The availability of an integrated connection depends on the PLC used. The following table shows the availability:

PLC	Integrated connection	Comments
S7-300/400	Yes	The linking of tags is not checked in Runtime. If the tag address changes in the PLC and the HMI device is not compiled again and loaded, the change is not registered in runtime.
S7-1200	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the behavior described for the S7-300/400 applies.
S7-1500	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the behavior described for the S7-300/400 applies.
SIMATIC ET 200 CPU	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the behavior described for the S7-300/400 applies.
SIMATIC S7-1500 software controller	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the behavior described for the S7-300/400 applies.

Create an integrated connection in the "Devices & Networks" editor. If the PLC is contained in the project and integrated connections are supported, you can then also have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.

Addressing with non-integrated connections

In the case of a project with a non-integrated connection, you always configure a tag connection with absolute addressing. Select the valid data type yourself. If the address of a PLC tag changes in a project with a non-integrated connection during the course of the project, you also have to make the change in WinCC. The tag connection cannot be checked for validity in Runtime, an error message is not issued.

A non-integrated connection is available for all supported PLCs.

Symbolic addressing is not available in a non-integrated connection.

With a non-integrated connection, the control program does not need to be part of the WinCC project. You can perform the configuration of the PLC and the WinCC project independently of each other. For configuration in WinCC, only the addresses used in the PLC and their function have to be known.

See also

Basics of tags (Page 519)

2.1.6 Internal Tags (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Internal tags do not have any connection to the PLC. Internal tags transport values within the HMI device. The tag values are only available as long as runtime is running.

Principle

Internal tags are stored in the memory of the HMI device. Therefore, only this HMI device has read and write access to the internal tags. You can create internal tags to perform local calculations, for example.

You can use the HMI data types for internal tags. Availability depends on the HMI device being used.

The following HMI data types are available:

HMI data type	Data format
Array	One-dimensional array
Bool	Binary tag
DateTime	Date/time format
DInt	Signed 32-bit value
Int	Signed 16-bit value
LReal	Floating-point number 64-bit IEEE 754
Real	Floating-point number 32-bit IEEE 754
SInt	Signed 8-bit value
UDnt	Unsigned 32-bit value

HMI data type	Data format
UInt	Unsigned 16-bit value
USInt	Unsigned 8-bit value
WString	Text tag, 16-bit character set

See also

Basics of tags (Page 519)

2.1.7 Internal Tags (RT Professional)

Introduction

Internal tags do not have any connection to the PLC.

Principle

Internal tags are stored in the memory of the HMI device. Therefore, only this HMI device has read and write access to the internal tags. You can create internal tags to perform local calculations, for example.

You can use the HMI data types for internal tags. Availability depends on the HMI device being used.

The following HMI data types are available:

HMI data type	Data format
Array	One-dimensional array
Bool	Binary tag
DateTime	Date/time format
DInt	Signed 32-bit value
Int	Signed 16-bit value
LReal	Floating-point number 64-bit IEEE 754
Raw	Raw data
Real	Floating-point number 32-bit IEEE 754
SInt	Signed 8-bit value
String	Text tag, 8-bit character set
TextRef	Text reference
UDInt	Unsigned 32-bit value
UInt	Unsigned 16-bit value
USInt	Unsigned 8-bit value
WString	Text tag, 16-bit character set

System tags

System tags are required for internal management of the project. The names of these tags always start with the "@" character. You may not delete or rename these tags. You cannot change the value of a tag.

Exceptions are the tags that are created by the "Redundancy" option. These can be set via scripts, for example:

- @RM_MASTER
- @RM_MASTER_NAME

Note

You must not create any tags whose name starts with a @.

See also

Basics of tags (Page 519)

Adapting the data type of a tag (Page 536)

2.2 Working with Tags

2.2.1 Creating tags

2.2.1.1 Creating external tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can access an address in the PLC via a PLC tag using an external tag. The following options are available for addressing:

- Symbolic addressing
- Absolute Addressing

You can find further details on symbolic addressing in section "Addressing external tags (Page 526)". If possible, use symbolic addressing when configuring a tag. You create tags either in the standard tag table or in a tag table you defined yourself.

Requirement

- You have opened the project.
- A connection to the PLC is configured.
- The Inspector window is open.

Procedure

To create an external tag, proceed as follows:

1. Open the "HMI tags" folder in the project tree and double-click the standard tag table. The tag table open.
   Alternatively, create a new tag table and then open it.
2. In the "Name" column, double-click "Add" in the tag table. A new tag is created.
3. Select the "Properties > Properties >General" category in the Inspector window and, if required, enter a unique tag name in the "Name" field. The tag name must be unique throughout the device.
4. If required, select the "Display name" field to enter a name to be displayed in Runtime. The name to be displayed is language-specific and can be translated for the required Runtime languages. The display name is available for Basic Panels, Panels and Runtime Advanced.

5. Select the connection to the required PLC in the "Connection" box. If the connection you require is not displayed, you must first create the connection to the PLC. You create the connection to a SIMATIC S7 PLC in the "Devices & Networks" editor. You create the connection to external PLCs in the "Connections" editor.
   If the project contains the PLC and supports integrated connections, you can also have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.
6. If you are working with an integrated connection, click the ... button in the "PLC tag" field and select an already created PLC tag in the object list. Click the ✓ button to confirm the selection.

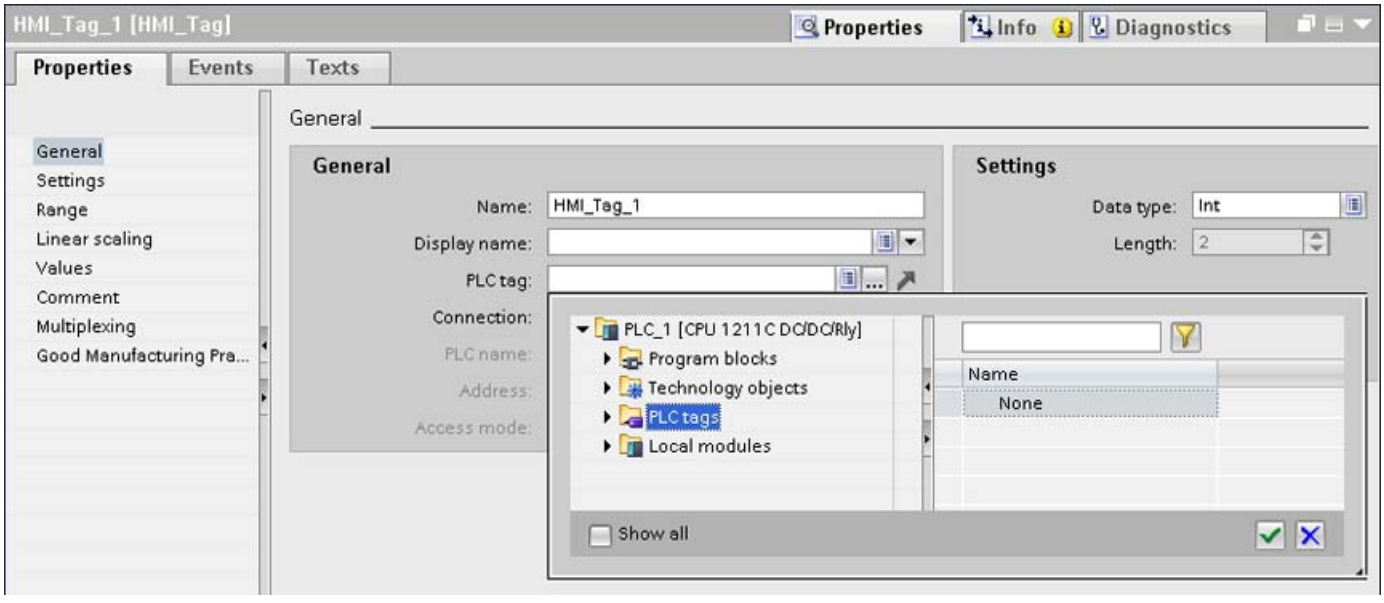

   Alternatively, use the autocomplete to select a PLC tag for an integrated connection.
   If you select a PLC tag from the autocomplete list, it is entered in the input path. The elements of the PLC tags are displayed in the autocomplete list. If you have selected a PLC tag that can be connected to the HMI tags, the PLC tag is connected to the HMI tags.
7. If you are working with a non-integrated connection, enter the address from the PLC in the "Address" field. The "PLC tag" field remains empty.
8. Configure the other properties of the tag in the inspector window.

You can also configure all tag properties directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

You can also create [illegible] tively directly at the application point, e.g. on an I/O field. To do this, click the Add new button in the object list. You then configure the new tag in the Inspector window.

Result

An external tag has been created and linked to a PLC tag or an address in the PLC.

Alternative procedure

You can also create external HMI tags by dragging and dropping data block elements or global PLC tags in an HMI tag table.

See also

Creating internal tags (Page 533)

Creating multiple tags (Page 534)

Addressing external tags (Page 526)

Adapting the data type of a tag (Page 536)

Manage tags (Page 538)

Tag Limits (Page 547)

Basics of tags (Page 519)

2.2.1.2 Creating internal tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You must at least set the name and data type for internal tags. Select the "Internal tag" item, rather than a connection to a PLC.

For documentation purposes, it is a good idea to enter a comment for every tag.

Requirement

You have opened the project.

Procedure

1. Open the "HMI tags" folder in the project tree and double-click the entry "Standard tag table". The tag table opens.
   Alternatively, create and then open a new tag table.
2. Double-click "Add" in the "Name" column of the tag table. A new tag is created.
3. If the Inspector window is not open, select the "Inspector window" option in the "View" menu.
4. Select the "Properties > Properties >General" category in the Inspector window and, if required, enter a unique tag name in the "Name" field. This tag name must be unique throughout the project.
5. If required, select the "Display name" field to enter a name to be displayed in Runtime. The name to be displayed is language-specific and can be translated for the required Runtime languages. The display name is available for Basic Panels, Panels and Runtime Advanced.
6. Select "Internal tag" as the connection in the "Connection" field.
7. Select the required data type in the "Data type" field.

8. In the "Length" field, you must specify the maximum number of characters to be stored in the tag according to the selected data type. The length is automatically defined by the data type for numerical tags.
9. As an option, you can enter a comment regarding the use of the tag. To do so, click "Properties > Properties > Comment" in the Inspector window and enter a text.

You can also configure all tag properties directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

You can a[illegible]ags alternatively directly at the application point, e.g. on an I/O field. Click the 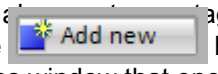

button in the object list. You can then configure the new tag in the Properties window that opens.

Result

An internal tag is created. You can now use this in your project.

In additional steps you can configure the tag, for example, by setting the start value and limits.

See also

Creating external tags (Page 531)

2.2.1.3 Creating multiple tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In a tag table, you create additional identical tags by automatically filling the rows of the table below a tag.

The tag names are incremented automatically when filling in automatically.

You can also use the auto fill function to fill table cells below a tag with a single tag property and thus modify the corresponding tags.

If you apply the automatic filling in to already filled cells of a tab table, you will be asked whether you want to overwrite the cells or insert new tags.

If you do not want to overwrite already configured tags, activate insert mode. Activate insert mode by keeping the <Ctrl> key pressed during insertion. Already existing entries in the tag table are moved down if insert mode is activated.

Requirement

- You have opened the project.
- A tag table is open.
- The tag which is to serve as a template for other tags is configured.

Procedure

1. If you want to create new tags, mark in the "Name" column the tag that should be used as a template for the new tags.
   If you want to copy a property of a tag to the tags below it, select the cell which contains this property.
   The selected cell will be highlighted in color and a small blue square will appear in its bottom right corner. If you move the mouse over this square, the cursor will change to a black cross.

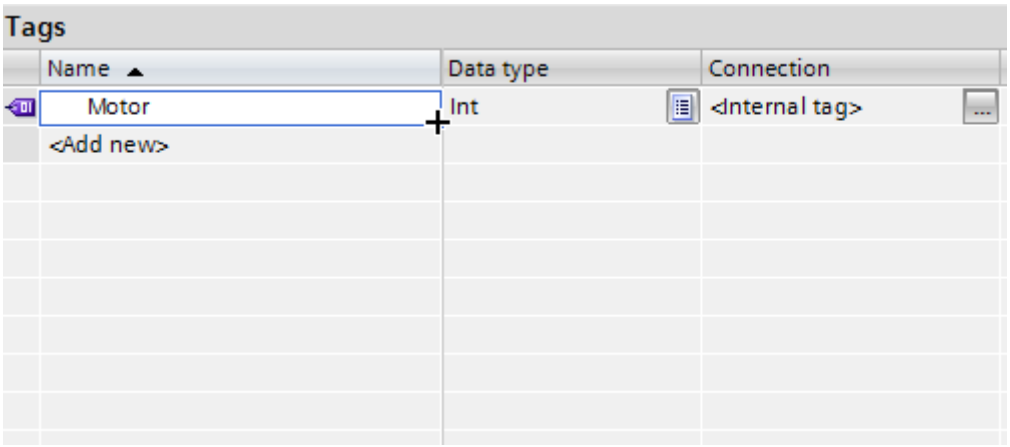

2. Hold down the mouse button and drag this square over the cells below that you wish to fill automatically.
   The marking will be extended to cover this area.

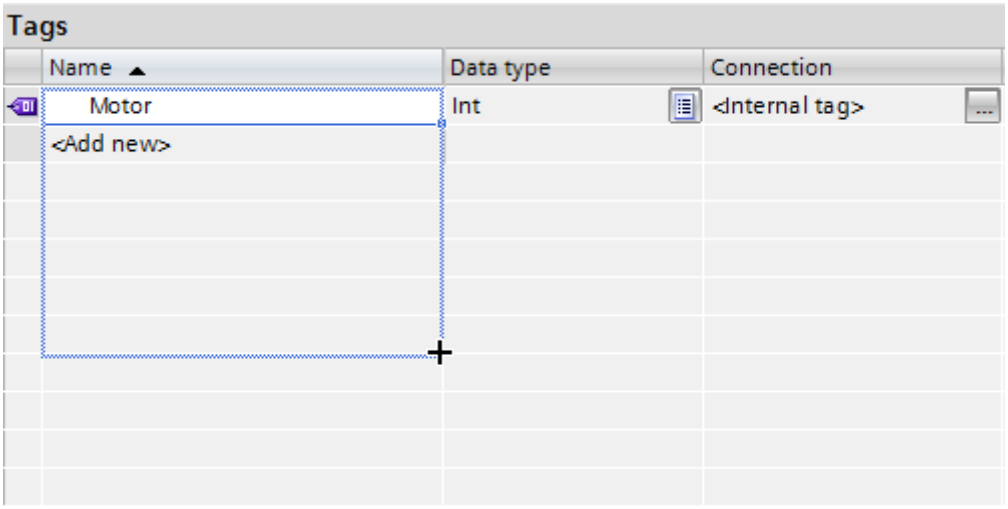

3. Now release the mouse button. All of the marked cells will be filled automatically.
   New tags will be created in all empty cells in the marked area.

Tag

Name	Data type	Connection
Motor	Int	<Internal tag>
Motor_1	Int	<Internal tag>
Motor_2	Int	<Internal tag>
Motor_3	Int	<Internal tag>
Motor_4	Int	<Internal tag>
Motor_5	Int	<Internal tag>
Motor_6	Int	<Internal tag>

Result

Depending on which cells were selected, the function may automatically fill individual properties or create new tags.

See also

Creating external tags (Page 531)

2.2.1.4 Adapting the data type of a tag (RT Professional)

Introduction

When you create a tag, you assign one of the possible data types to the tag. This data type depends on the type of data for which you would like to use the tag.

The data types available depend on the connected communication partner, such as a PLC.

Note

If you modify the data type of an existing, external tag, the previously defined tag address is identified as invalid. This reason for this is that the PLC address changes when the data type is modified.

Format adjustment

WinCC sets the data type of an external tag according to the data type of the connected PLC tag. If the data type of the PLC tag is not available in WinCC, a compatible data type is automatically used at the HMI tag. As required, you can specify that WinCC uses a different data type and converts the format of the data type of the PLC tag and the data type of the HMI tag.

In WinCC, you have access to the following data types:

HMI data type	Description	Value range
Array	Tag with contained elements[2)3)]	
DateTime	Date/time format[2)]	

HMI data type	Description	Value range
SInt	Signed 8-bit value	-128 ... +127
USInt	Unsigned 8-bit value	0 ... 255
Int	Signed 16-bit value	-32768 ... +32767
UInt	Unsigned 16-bit value	0 ... 65535
DInt	Signed 32-bit value	-2147483648 ... +2147483647
UDInt	Unsigned 32-bit value	0 ... 4294967295
Real	Floating-point number 32-bit IEEE 754	+-3.402823e+38
LReal	Floating-point number 64-bit IEEE 754	+-1.79769313486231e+308
Bool	Binary tag	0 ... 1
String	Text tag, 8-bit character set	
WString	Text tag, 16-bit character set	
TextRef	Text reference[1]	
Raw	Raw data type	

[1]: The "TextRef" data type is only available for internal tags. With tags of the "TextRef" data type, you reference an entry in the project texts. If you would like to display a text on an object that changes the configured language when the language of a text is changed, use the "TextRef" data type. You assign the tag to the project text.

[2]: Only available in conjunction with S7-1200 or S7-1500

[3]: To avoid structural conflicts in runtime, synchronize the tag name and replace the delimiters "[" and "]".

For format adaptation, select the desired PLC data type at the respective external tag. The suitable standard data type is then selected automatically in the "HMI data type" field for use in WinCC. Change the format adaptation as required. You can find more detailed information about this in section

Mapping and coding of data types (Page 600)

Data types without format adaptation

The data types are shown 1:1 without format adaptation.

SIMATIC S7-300/400 data types without format adjustment

PLC data type	Description
Bool	No format adaptation
String	No format adaptation

OPC-DA data types without format adaptation

OPC-DA data type	Description
VT_BSTR	No format adaptation
VT_BOOL	No format adaptation

OPC-DA XML data types without format adaptation

OPC-DA data type	Description
String	No format adaptation
boolean	No format adaptation
Base64Binary	No format adaptation

See also

Creating external tags (Page 531)

Mapping and coding of data types (Page 600)

Settings for synchronization (Page 544)

Data types (Page 608)

2.2.2 Editing Tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

2.2.2.1 Manage tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can always rename, copy or delete tags.

When a tag is renamed, the new name must be unique for the whole device.

If you use the "Copy" command to copy a tag to the clipboard, the objects and references linked to the tag are copied as well.

If you use the "Insert" command to add a tag to another device, the tag will be added without the connected references. Only the object name of the reference will be inserted. If a reference of the same name and valid properties exists in the target system, the existing reference will then be connected to the copied tag.

If you copy a tag, some of the objects linked to the tag are copied as well. The following objects are copied:

- Logging tags
- Cycles
- Alarms

If you add the copied tag to another device, the tag is added together with the linked objects.

Requirements

- The tag which you wish to rename, copy or delete must exist.
- The tag table containing the tag is open.

Renaming tags

Note

The connection to the tags can be interrupted in Runtime under the following conditions during renaming:

- HMI tag is used in a type, for example, to dynamize an object property via a script.
- One or more instances of the type are used in the project.
- Project is in Runtime.
- After the renaming, execute the command "Compile > Software (only changes)".

Solution: Exit Runtime and rename the tag. Execute the "Compile > Software (rebuild all)" command.

1. In the "Name" field, select the tag in the tag table.
2. Select "Rename" from the shortcut menu.
3. Type in a new name.
   The tag appears under its new name.

Copying tags

1. Select one or more tags in the tag table.
2. Select "Copy" from the shortcut menu.
3. Click on the point at which you want to insert the tag. For example, click another tag table in the same device or the tag table in a second device.
4. Select "Paste" from the shortcut menu. The tag is inserted as described above.

Deleting a tag

1. Select one or more tags in the tag table.
2. Select the "Cross-reference" command from the "Tools" menu. In the "Cross-reference" editor, check to see where the tags are used. In this manner, you can see what impact the deletion of the tag will have on your project.
3. Select "Delete" in the pop-up menu of the tag.
   All marked tags will be deleted.

Note

If you want a warning to be output during compiling when HMI tags are deleted in screens, clear the "Do not check deleted HMI tags in screens" check box under "Runtime settings > General > Compiler options". If this option is not activated, the affected field with the deleted tag is marked as invalid.

Export and import of tags

WinCC gives you the option to export and import tags. With Export and Import, you have the option to export tags from one project and import them into another project. There is also the option to create larger numbers of tags outside of WinCC, edit them and subsequently import into any WinCC project.

See also

Edit tags (Page 540)

Configuring multiple tags simultaneously (Page 541)

Using multiple tags simultaneously in a screen (Page 542)

Synchronizing tags (Page 546)

Creating external tags (Page 531)

Importing and exporting tags (Page 1986)

2.2.2.2 Edit tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can modify tags at any time to adapt them to changed requirements in the project.

Edit tags

You have the following options to change the configuration of a tag:

- You open the tag table in which the tag is contained.
- Open the tag table "Show all tags".
- You open the Inspector window of a tag with the "Edit" button in the object selection on the display and operating object.

In the tag tables, you can perform such tasks as comparing and adjusting the properties of multiple tags or sorting the tags by their properties.

You can change the properties either directly in the table or in the Inspector window.

If you change a tag property and this change causes a conflict with another property, it will be highlighted in color to draw your attention to this fact. If you link the tag with another controller, for example, which does not support the set data type, this property is highlighted in color.

See also

Manage tags (Page 538)

2.2.2.3 Configuring multiple tags simultaneously (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In WinCC, you can assign the same properties to multiple tags in a single operation. This facilitates efficient programming.

Requirement

- You created the tags you want to configure.
- The tag table is open.
- The Inspector window is open.

Procedure

1. In the tag table, select all the tags that you want to configure at the same time.
   If the selected property is identical for all the tags, the setting for this property will appear in the Inspector window. The associated field will remain blank otherwise.
2. You can define the shared properties in the Inspector window or directly in the tag table.

if you change a property commonly on several tags, only this one property is changed. The other properties of the tag remain unchanged.

Result

All marked tags will be reconfigured.

To edit tag properties which differ from one tag to the other, simply clear the multiple selection.

See also

Manage tags (Page 538)

2.2.2.4 Using multiple tags simultaneously in a screen (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In WinCC, you can create multiple I/O fields that are linked with tags in one screen in a single operation. This facilitates efficient programming.

Requirement

- Several tags are set up.
- A screen is open.

Procedure

1. In the project tree, select the required tag table under "HMI tags".

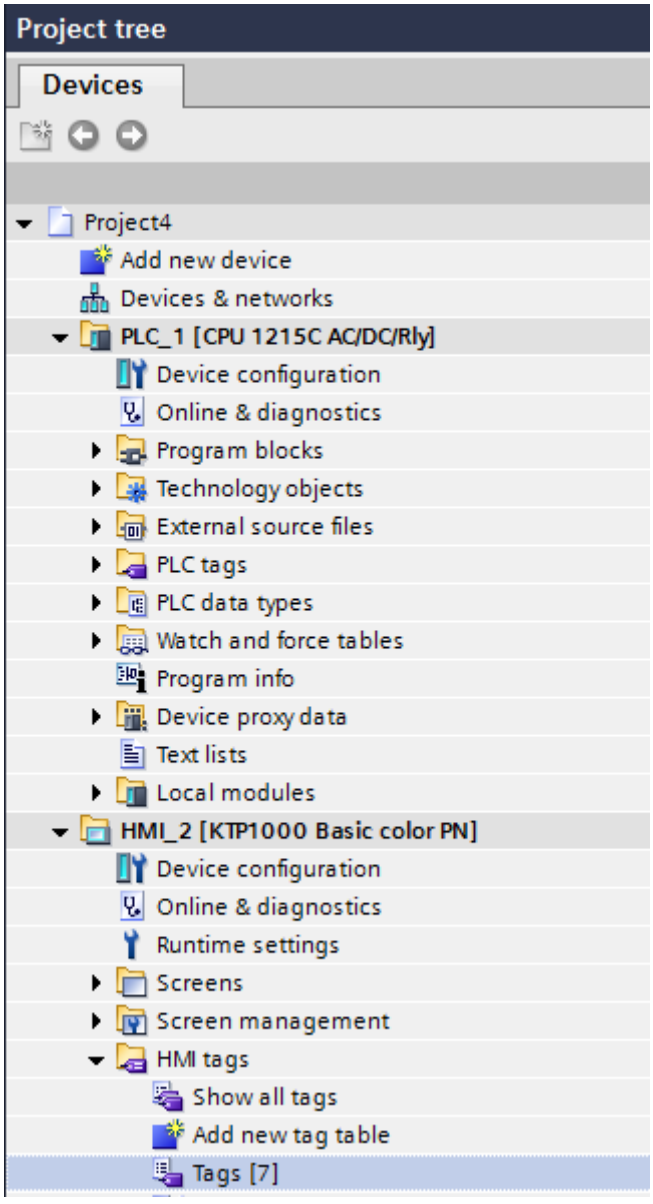

2. Select the detail view at the bottom of the project tree. The detail view shows the tags that exist in the selected tag group.

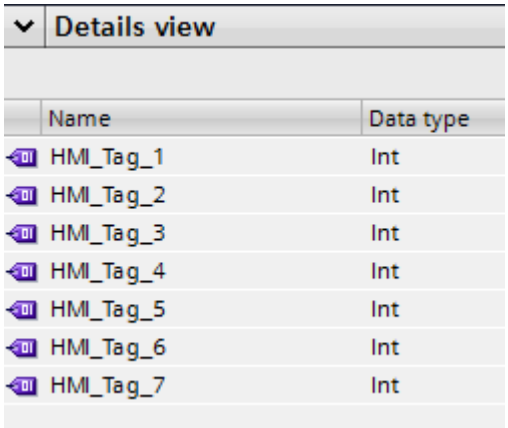

3. Mark the tags in the detail window.
4. Drag the tags to the screen. For each tag, this creates an I/O field that is connected to the tag.

Note

When you move a PLC tag from the detail window to the work area by drag&drop, a network and a connection are created additionally in the "Devices & Networks" editor.

See also

Manage tags (Page 538)

2.2.2.5 Settings for synchronization (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In WinCC, you can map the location of a data value in the structure of a data block in the WinCC tag name. To do this, you synchronize the WinCC tags with the connected PLC tags. If necessary, the PLC name is set as a prefix.

You configure the name matching based on the requirements of your project set under "Runtime Settings > Settings for Tags". WinCC offers you several options for this.

You specify whether the names are matched and the conditions for this to occur during synchronization.

Settings for tags

To avoid conflicts within complex tag types that you configure under "Settings for tags", the delimiters of the path statement from STEP 7 are replaced in WinCC similar to name matching:

- Compatibility mode
  In compatibility mode, an underscore is set between the names of the data block and the element name of the first hierarchy level. The delimiters in the lower hierarchy levels are retained. When you enable this option, all other options are disabled.
- Replace delimiter
  Depending on your selection, the delimiters of all hierarchy levels are replaced during synchronization.
- PLC prefix
  The PLC name is set as a prefix to the WinCC tag name. You set the option for each HMI connection.

Note

Duplicate tag names

If the newly generated tag name is already in use, a number is added in parentheses, for example, Datablock_1_Static_2{1}(1).

Example

The "PLC1" controller contains the structured data block "DB1". The "Db1.a[1].b.c[3]" data block element is used in a picture. Depending on your settings, the WinCC tag name is generated as follows:

WinCC tag name	Selected option	Comment
Db1_a[1].b.c[3]	Compatibility mode	
Plc1.Db1.a[1].b.c[3]	PLC prefix	
Db1;a(1);b;c(3)	Dot and parenthesis replaced with ; ( )	
Plc1_Db1_a{10}_b_c{3}	Dot and parenthesis replaced with _ { } PLC prefix	
Db1.a[1].b.c[3]	Delimiter replaced without character selection	The tag name is enclosed in quotes at the point of use in the picture: "Db1.a[1].b.c[3]"

See also

Synchronizing tags (Page 546)

2.2.2.6 Synchronizing tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

To synchronize the PLC and HMI tags, WinCC offers the following options:

- Synchronizing tags with or without name matching between PLC and WinCC
  The following options are available for this:
- Link tags with addresses in the PLC
  This procedure is suitable, for example, if changes were made to the connection between the HMI device and the PLC and the tag connections were lost. The function can also be used if you have configured the control program and HMI project separately.

Requirement

- External HMI tags have been created.
- PLC tags have been created.
- An HMI connection to a PLC in the project has been established.

Procedure

To synchronize HMI tags with PLC tags, follow these steps:

1. In the project tree, select the directory that contains the tags in question.
2. Select "Connect HMI tags to matching PLC tags" from the shortcut menu.
   A dialog opens.

3. Select the option you want to use.
   If you want to synchronize the tags without name matching, disable "Replace WinCC tag name with PLC tag name".
   If you want to connect HMI tags new with absolute access, select "Data type and absolute address match".
4. Confirm with "Synchronize".
   The system searches for a suitable PLC tag according to the selected option.

Result

The external HMI tags are synchronized with the PLC tags.

If you have selected the option "Data type and absolute address match", the tag connection is established as soon as a suitable PLC tag is found.

If you have selected a different option, the WinCC tags are updated accordingly and the PLC tag names are applied in WinCC.

See also

Manage tags (Page 538)

Settings for synchronization (Page 544)

2.2.3 Configuring Tags

2.2.3.1 Tag Configuration Basics (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Tag Limits (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can restrict the value ranges with limits for numerical tags.

Principle

You can specify a value range defined by a high limit and a low limit for numerical tags.

You configure four limit values that limit the value range. Using the limits Upper 2 and Lower 2 , you specify the maximum and minimum value for the value range. The limits Upper 1 and Lower 1 specify the threshold values at which the normal range is exceeded or undershot.

Limit	Use
Upper 2	Specifies the maximum value.
Upper 1	Specifies the value at which the normal range is exceeded.

Limit	Use
Lower 1	Specifies the value at which the normal range is undershot.
Lower 2	Specifies the minimum value.

If the operator enters a value for the tag that is outside the configured value range, the input is rejected. The value is not accepted. You configure a function list when configuring for Runtime Advanced and Panels. When the tag value leaves the value range, the function list is processed.

Note

If you want to output an analog alarm when a limit is violated, configure the respective tag in the "Analog alarms" tab. You can also configure the analog alarm in the "HMI alarms" editor. The values for output of an analog alarm depend on the configured tag limits.

Application example

Use the limits to warn the operator when the value of a tag enters a critical range, for example.

See also

Defining Limits for a Tag (Page 548)

Start value of a tag (Page 549)

Defining the start value of a tag (Page 550)

Connecting a tag to another PLC (Page 551)

Linear scaling of a tag (Page 551)

Applying linear scaling to a tag (Page 553)

Defining Limits for a Tag (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

For numerical tags, you can specify a value range by defining a low and high limit as well as the threshold values.

For Runtime Advanced and Panels you can configure the system to process a function list whenever a tag value drops below or exceeds its configured value range.

Requirements

- The tag for which you want to set the limits is created.
- The Inspector window with the properties for this tag is open.

Procedure

To define limits for a tag, proceed as follows:

1. In the Inspector window, select "Properties > Properties > Range". If you want to define one of the limits as a constant value, select "Constant" using the button. Enter a number in the relevant field.
   If you want to define one of the limits as a tag value, select "HMI tag" using the button. Use the object list to define the tag for the limit.
2. To set additional limits for the tag, repeat step 1 with the appropriate settings.

Alternative procedure

You can also configure the limits directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Configuring a function list for Runtime Advanced and Panels

When you configure Runtime Advanced and Panels, you can have a function list called up if a high or low limit is violated.

1. If you want to start a function list when the value drops below the value range, click "Properties > Events > On falling below" in the Inspector window. Create a function list in this dialog.
2. If you want to start a function list when the value exceeds the value range, click "Properties > Events > On exceeding" in the Inspector window. Create a function list in this dialog.

Result

You have set a value range defined by the limits for the selected tag. If the value range is exceeded or undershot, a function list may be carried out.

See also

Tag Limits (Page 547)

Start value of a tag (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Value of a tag at start of Runtime

You can configure a start value for numeric tags and tags for date/time values. The tag will be preset to this value when Runtime starts. In this way, you can ensure that the tag has a defined status when Runtime starts.

For external tags, the start value will be displayed on the HMI device until it is overwritten by the PLC or by input.

If no start value was configured, the tag contains the value "0" when starting Runtime.

In WinCC Runtime Professional you can enter a tag value in place of the start value on a tag with the "String" data type. The tag value is saved in the "Project texts" editor and is multilingual.

After the text has been translated, it is displayed in Runtime as a language-dependent start value.

Application example

You can assign a default value to an I/O field. Enter the desired default value as start value for the tag that is linked to the I/O field.

See also

Tag Limits (Page 547)

Defining the start value of a tag (Page 550)

Defining the start value of a tag (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In WinCC you can configure a start value for a numeric tag and a tag for date/time values which this adopts at Runtime start.

Requirement

- You have created the tag for which you want to define a start value.
- The Inspector window with the tag properties is open.

Procedure

To configure a start value, proceed as follows:

1. In the Inspector window select "Properties > Properties > Values."
2. Enter the desired "Start value."

Alternative procedure

You can also configure the start value directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Result

The start value you selected for the tag is transferred to the project.

See also

Tag Limits (Page 547)

Start value of a tag (Page 549)

Connecting a tag to another PLC (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In WinCC, you can change the PLC connection of a tag at any time. This is needed when you change the configuration of your plant, for example.

Depending on the PLC selected, you may need to modify the configuration of the tag. The tag properties which must be changed will be highlighted in color.

Requirement

- The external tag, whose connection you wish to change, must already exist.
- The connection to the PLC must already exist.
- The Properties window for this tag is open.

Procedure

To change the PLC connection of a tag, proceed as follows:

1. In the Inspector window select "Properties > Properties > General."
2. Select the new connection in the "Connection" field.
   The tag properties that you must change will be highlighted in color in the tag table and in the Inspector window.
3. Change all highlighted properties of the tag to suit the requirements of the new PLC.

Result

The external tag is connected to the new PLC.

See also

Tag Limits (Page 547)

Linear scaling of a tag (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Numeric data types can be processed with linear scaling. The process values in the PLC for an external tag can be mapped onto a specific value range in the project.

Principle

To apply linear scaling to a tag, you must specify one value range on the HMI device and one on the PLC. The value ranges will be mapped to each other linearly.

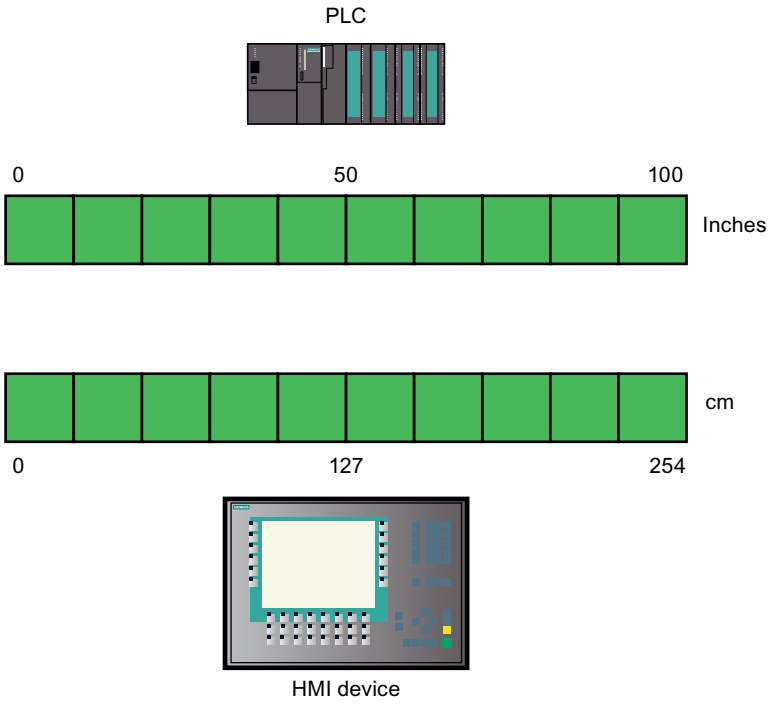

As soon as data from the HMI device is written to an external tag, it will be automatically mapped to the value range of the PLC. As soon as data from the HMI device is read from the external tag, a corresponding transformation will be performed in the other direction.

Note

You can also use the system functions "LinearScaling" and "InvertLinearScaling" to automatically convert process values.

Application example

The user enters length dimensions in centimeters but the PLC is expecting inches. The entered values are automatically converted before they are forwarded to the controller. Using linear scaling, the value range [0 to 100] on the PLC can be mapped onto the value range [0 to 254] on the HMI device.

See also

Applying linear scaling to a tag (Page 553)

Tag Limits (Page 547)

Applying linear scaling to a tag (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

To apply linear scaling to a tag, you must specify one value range on the HMI device and one on the PLC. The value ranges will be mapped to each other linearly.

Requirement

- The external tag to which linear scaling is to be applied must exist.
- The Inspector window with the properties for this tag is open.

Procedure

To apply linear scaling to a tag, follow these steps:

1. In the Inspector window select "Properties > Properties > Linear scaling."
2. Click on "Enable" to switch on linear scaling.
   Using this option, you can temporarily switch off linear scaling for testing purposes, for example. Settings which were made earlier for linear scaling remain unchanged.
3. In the "PLC" area, enter the start and end values of the value range to be applied to the process values on the PLC.
4. In the "HMI device" area, enter the end and start values of the value range to be applied to the process values on the HMI device.

Result

In Runtime the data will be automatically mapped from one value range to the other.

Note

You can also use the "LinearScaling" and "InvertLinearScaling" system functions to automatically convert process values.

See also

Linear scaling of a tag (Page 551)

Tag Limits (Page 547)

Making characters available for HMI tags of the type WString/WChar (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In tags of the type WString or WChar, you specify the strings that can be displayed with specific fonts.

Only those characters that are used in texts in the configuration are transferred to the Basic Panels and Comfort Panels via the configuration. Therefore, it is possible that characters that are displayed via tags of the WString or WChar type are not displayed.

The Tahoma and Courier New fonts are fully pre-installed on the Comfort Panels. The Tahoma font is fully pre-installed on the Basic Panels. The characters of most languages can be displayed with the Tahoma font. The characters of the Chinese, Taiwanese, Japanese and Korean languages cannot be displayed with Tahoma.

WinCC Runtime Advanced supports all fonts that are installed on the configuration PC and the Runtime PC. If you want to use a different font, install it on the configuration PC and the Runtime PC.

For Comfort Panels, it is possible to load the complete font to the HMI device via ProSave.

For Basic Panels, it is not possible to install fonts via ProSave. Therefore, the characters must always be contained in the configured texts.

Basics

For Comfort Panels, you can download the following fonts to the HMI device with the additional ProSave options:

- SimSun for the input of strings in Chinese (PRC), Chinese (Singapore), English, German, Italian, French and Spanish.
- Gulim for the input of strings in Korean.
- MS PGothic for the input of strings in Japanese.

You can find additional information on the topic of "Loading Asian fonts via ProSave" under Loading Asian fonts to the HMI device (Page 2348).

Note

"Chinese Traditional" (Taiwan, Hong Kong, Macao)

You need the "MingLiU" font to display traditional Chinese characters for Taiwan, Hong Kong and Macao. It is not possible to load this font to the Comfort Panels via the ProSave options. The SimSun and Tahoma fonts cannot be used to display traditional Chinese characters.

Therefore, the traditional Chinese characters that occur in WStrings/WChars must be contained in the configured texts on Comfort Panels.

Using characters as configured texts in screens

To use fonts that are not installed on an HMI device in Runtime, provide the characters as configured texts in a screen:

1. Create a new screen.
2. Insert a text box.
3. In the properties of the text box, set a font that contains characters that you wish to use later.
4. Insert the characters of the font that you wish to use later into the text box.

After the project is compiled, you can use the characters configured in the text in HMI tags of the type WString/WChar.

See also

Loading Asian fonts to the HMI device (Page 2348)

2.2.3.2 Special configuration options (RT Advanced, Basic Panels, Panels, Comfort Panels)

Address multiplexing (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Using address multiplexing, you can use a single tag to access a multitude of memory locations within the PLC's address range. You read and write to the addresses without defining a tag for each individual address.

The square brackets indicate address multiplexing.

Multiplexing with absolute addressing

When using multiplexing with absolute addressing, you configure tags as placeholders for the address in the PLC to be addressed.

If you want to access, for example, an address of the format "%DBx.DBWy", the expression for multiplexing is as follows:

"%DB[HMITag1].DBW[HMITag2]"

In Runtime, you supply the tag "HMITag1" with the required value for the data block you want to address.

In Runtime, you supply the tag "HMITag2" with the required address from the data block.

Tags are supplied with values, for example, with the help of values from the PLC or via a script.

Multiplexing with absolute addresses is supported for the following PLCs and communication drivers.

- SIMATIC S7-300/400
- SIMATIC S7-1200
- SIMATIC S7-1500

Multiplexing with absolute addresses is not available for data blocks with optimized access.

Multiplexing with symbolic addressing

When multiplexing with symbolic addressing, you access various array elements of an array tag in a data block of the connected controller by means of a multiplex tag and an index tag.

First, connect an element of the desired array tags in a data block to the multiplex tags in the HMI device. Then replace the array index (leftmost in the symbolic address) with the index tag.

If you want to access, for example, the various elements of the array tag "Arraytag_1" in the data block "Datablock_1", the expression for symbolic addressing is as follows:

"Datablock_1.Arraytag_1["HMITag_1"]

You control the access to the index of the array elements with the HMI tag "HMITag_1". In Runtime, you supply the tag with the index of the array element that you want to access.

Multiplexing with symbolic addressing is only available if the following components support symbolic addressing:

- HMI device
- PLC
- Communication driver

Symbolic addressing is supported by the following communication drivers:

- SIMATIC S7-1200
- SIMATIC S7-1500

Structured HMI tags with user-defined PLC data types (UDT) and arrays can be used for multiplexing with symbolic addressing.

Multiplexing is possible under the following conditions:

- The path of the connected PLC tag includes at least one array index.
- The array, which is left in the path of the PLC tags, is one-dimensional.
- If there are multiple arrays in the path of the PLC tags, only the array at the far left is available for multiplexing.
- The index tag must be of the "USInt", "UInt", "UDInt", "SInt", "Int" or "DInt" data type.

The following figure shows multiplexing with symbolic addressing using a structured HMI tag with custom PLC data types (UDT). The data block element "db1.udt3Arr[0]" is linked to the HMI tag "db1_udt3Arr{0}". Symbolic addressing of the HMI tag "db1_udt3Arr{0}" is multiplexed by the "index" HMI tag:

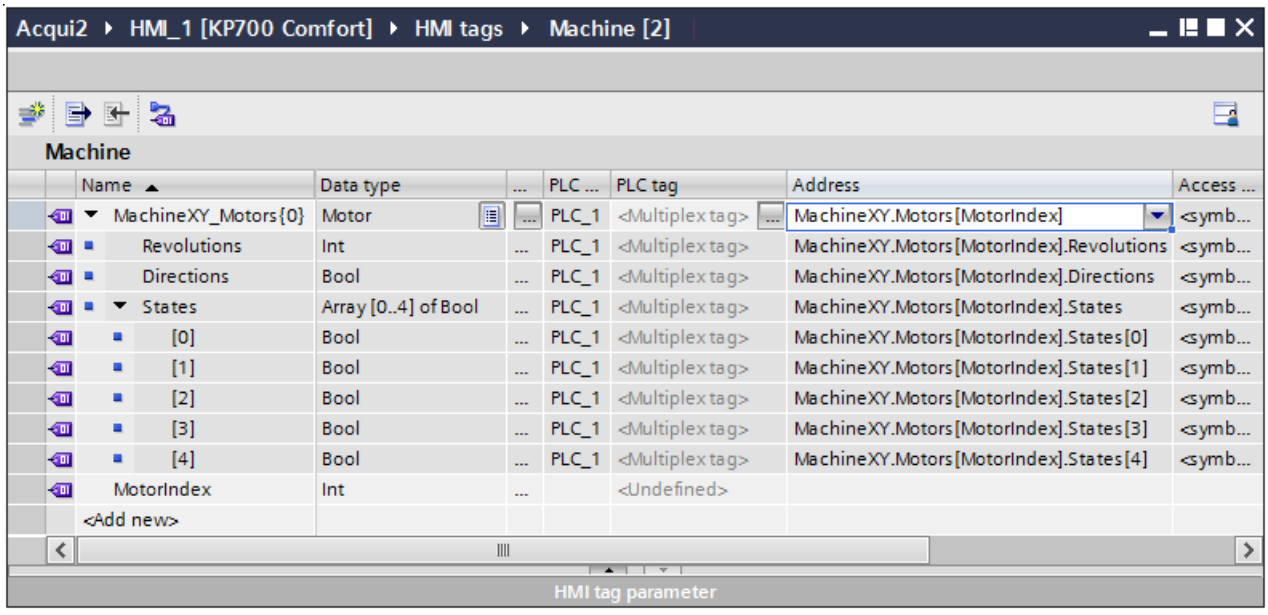

Name	Data type	...	PLC ...	PLC tag	Address	Access ...
MachineXY_Motors{0}	Motor		PLC_1	<Multiplex tag>	MachineXY.Motors[MotorIndex]	<symb...
Revolutions	Int	...	PLC_1	<Multiplex tag>	MachineXY.Motors[MotorIndex].Revolutions	<symb...
Directions	Bool	...	PLC_1	<Multiplex tag>	MachineXY.Motors[MotorIndex].Directions	<symb...
States	Array [0..4] of Bool	...	PLC_1	<Multiplex tag>	MachineXY.Motors[MotorIndex].States	<symb...
[0]	Bool	...	PLC_1	<Multiplex tag>	MachineXY.Motors[MotorIndex].States[0]	<symb...
[1]	Bool	...	PLC_1	<Multiplex tag>	MachineXY.Motors[MotorIndex].States[1]	<symb...
[2]	Bool	...	PLC_1	<Multiplex tag>	MachineXY.Motors[MotorIndex].States[2]	<symb...
[3]	Bool	...	PLC_1	<Multiplex tag>	MachineXY.Motors[MotorIndex].States[3]	<symb...
[4]	Bool	...	PLC_1	<Multiplex tag>	MachineXY.Motors[MotorIndex].States[4]	<symb...
MotorIndex	Int	...		<Undefined>		
<Add new>						

The following tags can not be used as index tags for multiplexing with symbolic addressing:

- HMI tags that are not of the "USInt", "UInt", "UDInt", "SInt", "Int" or "DInt" data type
- HMI tags that are already multiplexed
- Array elements
- HMI tags that are elements of structured HMI tags

Note

In Runtime, the first array element is always addressed with the index value "0". Have your array element in the data blocks also start with the index "0".

Note

You can export the HMI tags that are used for multiplexing with symbolic addressing to MS Excel. You can also import such HMI tags from MS Excel into an HMI tag table that contains all HMI tags of the HMI device.

See also

Configuring address multiplexing with absolute addressing (Page 558)

Configuring address multiplexing with symbolic addressing (Page 561)

Updating the Tag Value in Runtime (Page 567)

Indirect addressing of tags (Page 564)

Addressing tags indirectly (Page 565)

Using tags to trigger functions (Page 566)

Defining the acquisition cycle for a tag (Page 568)

Configuring address multiplexing with absolute addressing (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

When using address multiplexing, you can efficiently access different addresses in the PLC with the help of a small number of tags. Instead of the absolute address in the PLC, you use tags in order to be able to change the address in Runtime.

Requirement

- The tag for address multiplexing is created and connected to the PLC.
- The Properties window for this tag is open.

Procedure

1. Select the tag for address multiplexing in the tag table, and select "Properties > Properties > General" in the Inspector window. The general properties of the tag are displayed.

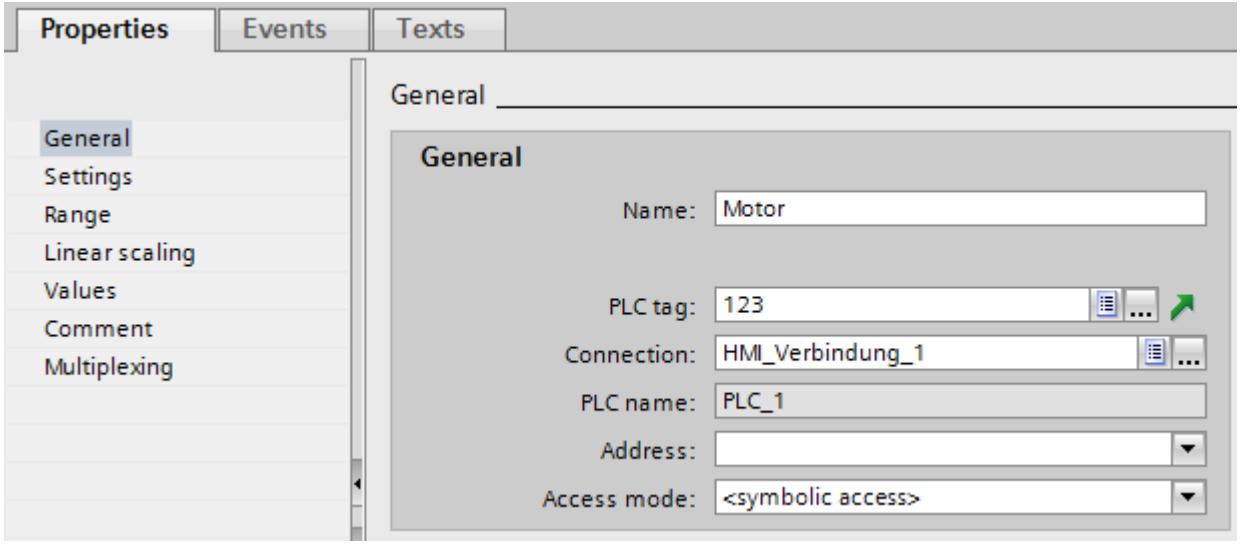

2. Select the "Int" data type for this example.
3. Select the access type "Absolute addressing".

4. Click the selection button in the "Address" field. The address dialog opens.

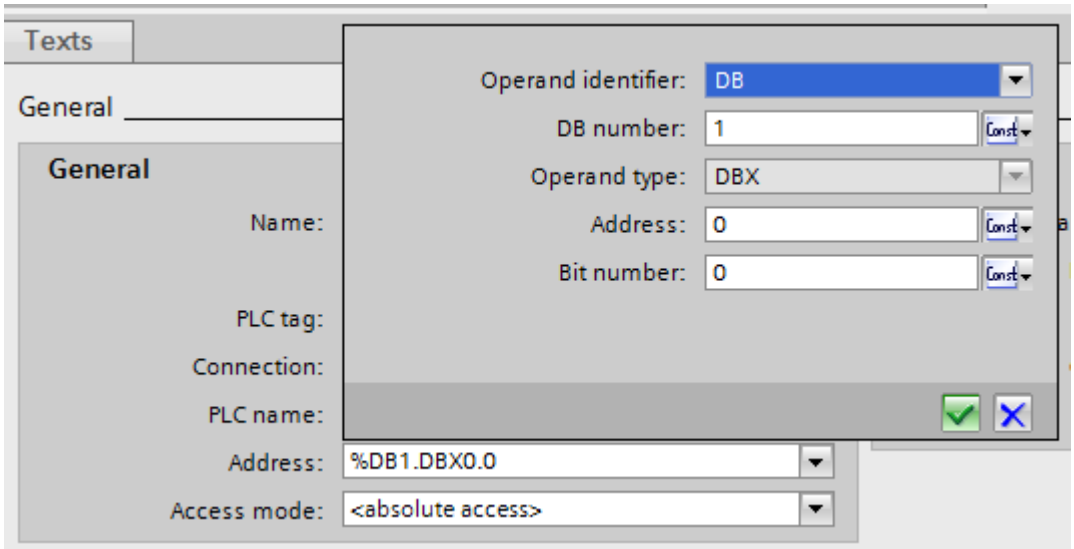

5. Click the selection button in the "DB number" field and select the entry "HMI tag".

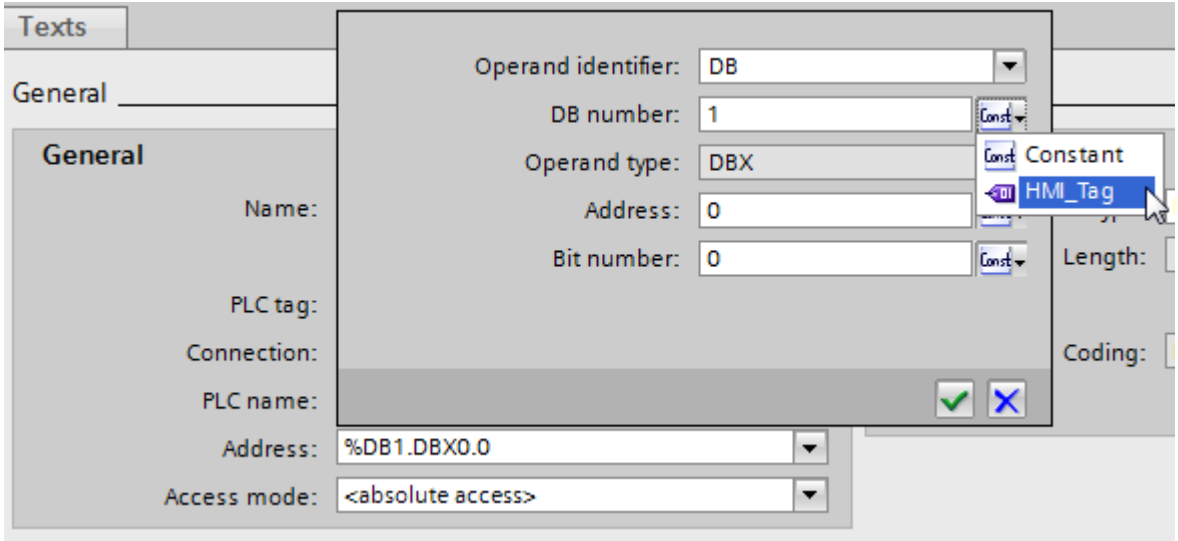

6. In the "DB number" field, click the button and select a tag for the DB number in the object list. Or create a new tag with the help of the object list. Accept the tag by clicking the button.

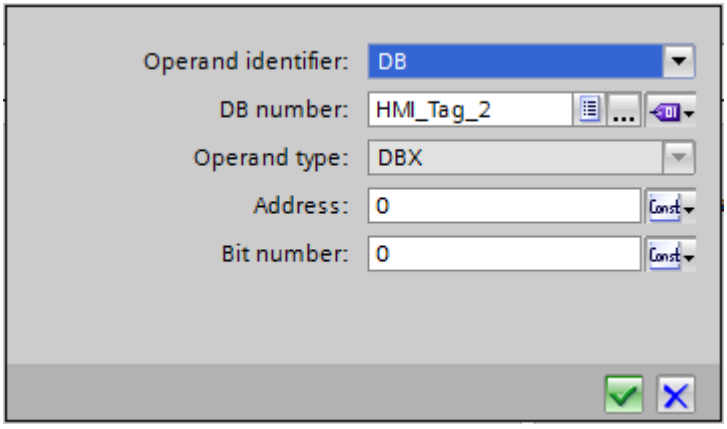

7. Repeat steps 3 and 4 for the "Address" field and configure a further tag for calling the address area in the data block.

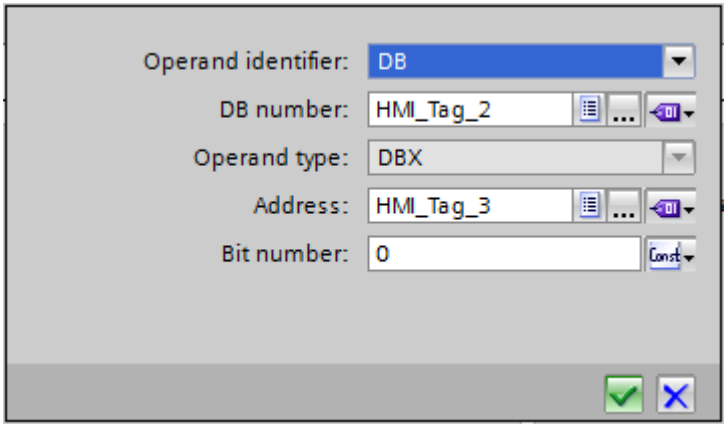

The selection options in the Address dialog depend on the selected data type of the multiplex tag. The Address dialog offers only address settings that can be configured with the selected data type.

Result

In runtime, the multiplex tag is used to access the memory location corresponding to the address currently found in the tag. You control access to the data block with the tag in the DB number field. You control access to the address in the selected data block with the tag in the "Address" field.

Note

The value in the memory location will only be read at the next update cycle for the addressed tag.

If, for example, you use a multiplex tag in a script, do not attempt to access contents of the memory location directly after changing it.

See also

Address multiplexing (Page 555)

Configuring address multiplexing with symbolic addressing (Page 561)

Configuring address multiplexing with symbolic addressing (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

When using address multiplexing, you can efficiently access different addresses in the PLC with the help of a small number of tags. Instead of the symbolic address in the PLC, you use tags in order to be able to change the address in Runtime.

Requirement

- The tag for address multiplexing is created.
- The Properties window for this tag is open.
- A data block with an array tag is created in the connected PLC.
- The data block was compiled.

Procedure

1. Select the tag for address multiplexing in the tag table, and select "Properties > Properties > General" in the Inspector window. The general properties of the tag are displayed.

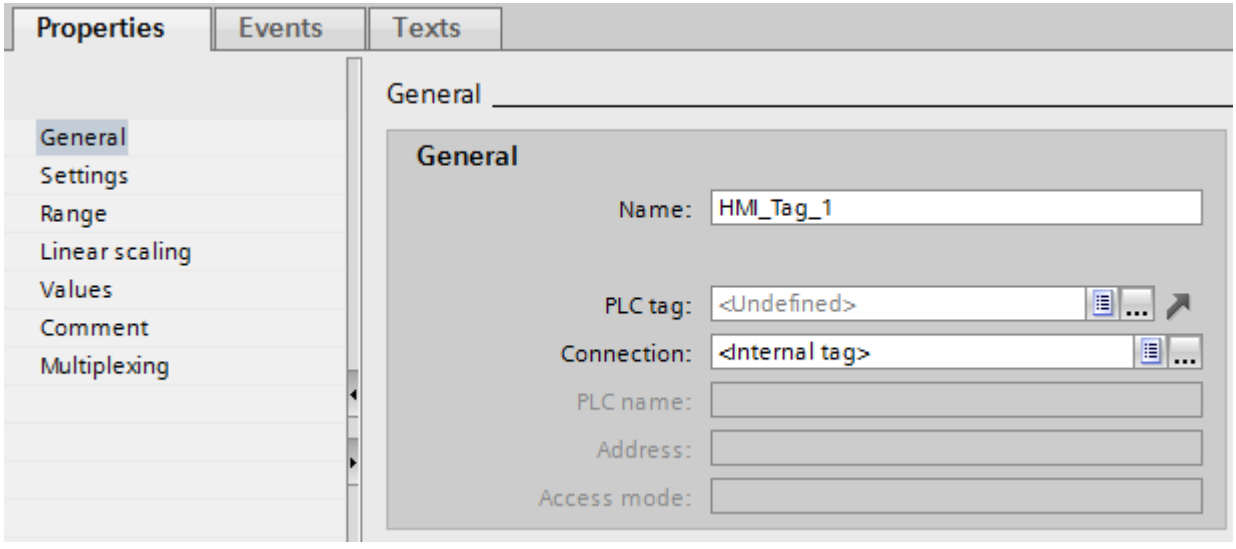

2. Select the connection to the PLC via the "Connection" field.

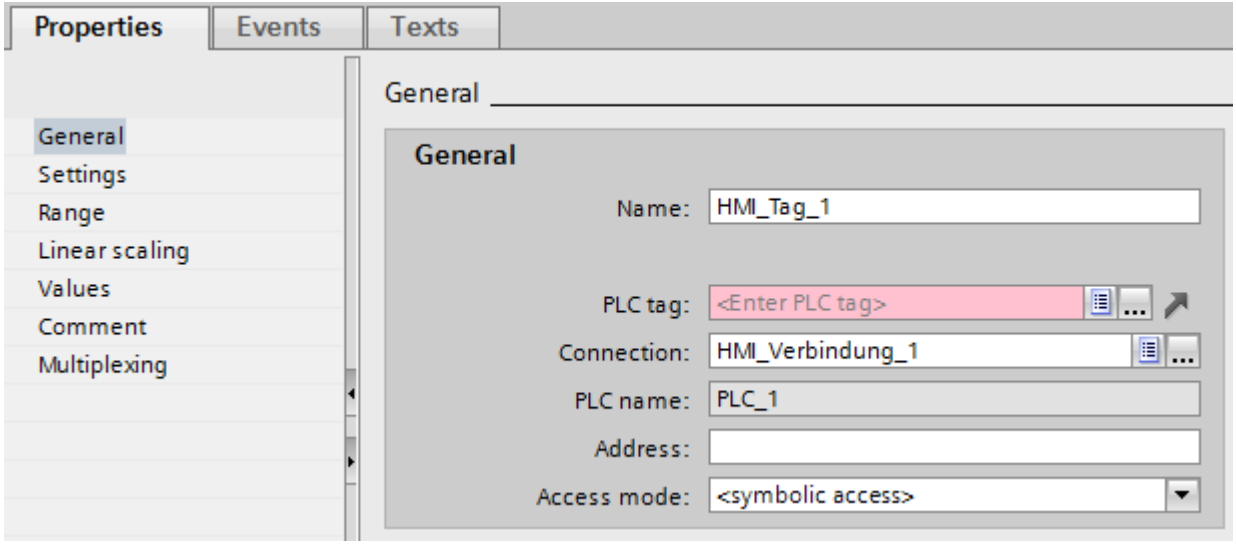

3. Select the access type "Symbolic access".

4. Navigate to the data block of the PLC via the "PLC tag" field and select an array element of the array tag.

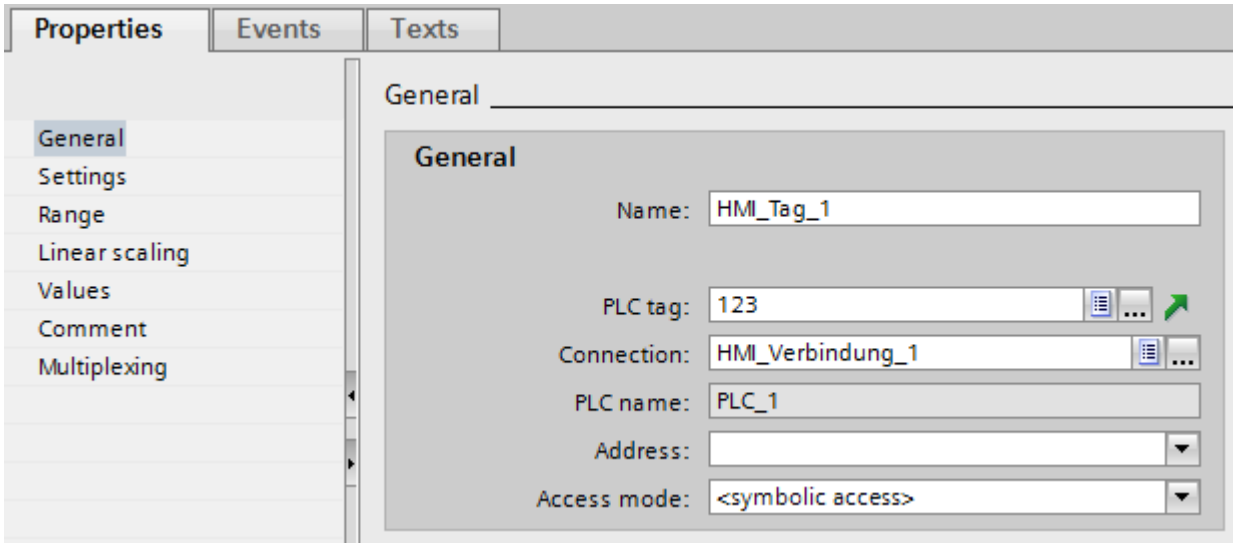

5. Click the selection button in the "Address" field. The address dialog opens.
6. Click the selection button in the "Index tag" field and select the entry "HMI tag".

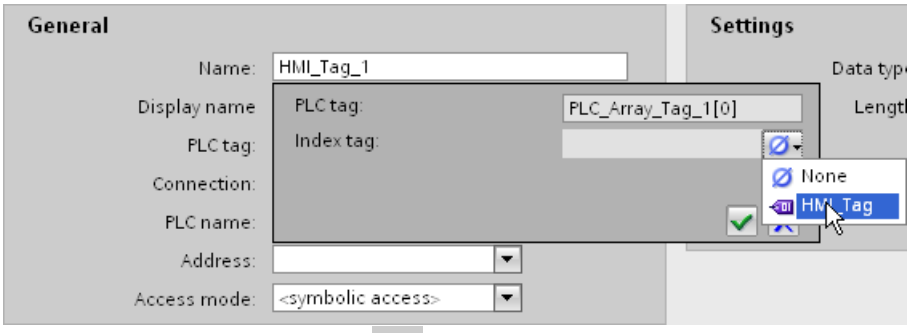

7. In the "Index tag" field, click the button and select a tag for the array index in the object list. Or create a new tag with the help of the object list. Accept the tag by clicking the button.

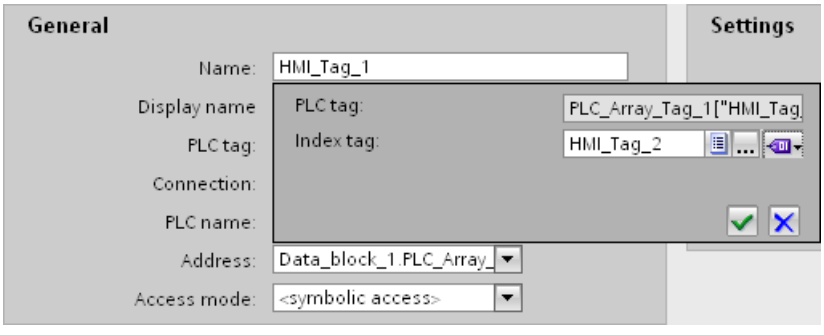

Result

In Runtime, the array element whose index value is contained in the index tag is accessed.

See also

Address multiplexing (Page 555)

Configuring address multiplexing with absolute addressing (Page 558)

Indirect addressing of tags (RT Advanced, Panels, Comfort Panels)

Principle

In multiplexes, a type of indirect addressing, the tag used is first determined at runtime. A list of tags is defined for the multiplex tag. The relevant tag is selected from the list of tags in runtime. The selection of the tag depends on the value of the index tag.

In Runtime, the system first reads the value of the index tag. Then the tag which is specified in the corresponding place in the tag list is accessed.

Application example

Using indirect addressing, you could configure the following scenario:

The operator selects one of several machines from a selection list. Depending on the operator's selection, data from the selected machine will be displayed in an output field.

To configure such a scenario, configure the index tag for a symbolic I/O field. Configure the multiplex tag at an I/O field. Configure the tag list of the multiplex tag to reflect the structure of the selection list.

If the operator selects another machine, the value of the index tag will change. The selection field then displays the content of the tag that is pointed to in the tag list in the multiplex tag by the new index value.

See also

Addressing tags indirectly (Page 565)

Address multiplexing (Page 555)

Addressing tags indirectly (RT Advanced, Panels, Comfort Panels)

Introduction

With indirect addressing, the tag used is first determined at runtime. Instead of a single tag, a list of tags is defined. The list entries consist of an index value and the name of the tag to be used. You can control which entry in the tag list is accessed using an index tag.

Note

Data type of the index tags

If you use tags in the tag list with different data types, the properties of the largest data type are used. If you use tags of the INT and REAL data types, for example, the values are shown in floating point numbers.

Requirement

- The tag which you wish to address indirectly must already exist.
- The index tag must exist.
- The tags which will be contained in the tag list must already exist.
- The Inspector window with the tag properties is open.

Procedure

To address tags indirectly, proceed as follows:

1. In the Inspector window select "Properties > Properties > Multiplexing".
2. Select the "Multiplexing" option to activate indirect addressing.
   Using this option, you can temporarily switch off indirect addressing for testing purposes, for example. Settings which were made earlier for indirect addressing remain unchanged.
3. Select an "Index tag" or define a new tag using the object list.
4. Click the first entry in the "Tags" column in the tag list.
5. Select a tag as a list entry or define a new tag using the object list.
   The entry in the "Index" column will be generated automatically.
6. Repeat step 5 for all tags that you wish to add to the tag list.
7. If necessary, you can use drag-and-drop to change the order of the entries in the list.

Result

In runtime, the system will dynamically access the tag in the tag list which has the same index value as the value currently in the index tag.

See also

Indirect addressing of tags (Page 564)

Address multiplexing (Page 555)

Using tags to trigger functions (RT Advanced, Panels, Comfort Panels)

Introduction

You can use the values of variables as the triggering event for an action in runtime. To start an action in Runtime, configure a function list for a tag. Include one or more system functions or VB scripts in the function list. The function list is processed when the configured event occurs.

The following events are available for a tag:

- Change in value of the tag
  Function list processing is triggered by each change in the value of the variable.
  When the tag is defined as an array tag, the function list is processed whenever an array element changes.
- Violation of the tag's high limit
  The function list is processed when the high limit is violated.
- Violation of the tag's low limit
  The function list is processed when the low limit is violated.

Requirement

- The tag whose value you wish to use as an event already exists.
- The Inspector window with the properties for this tag is open.

Procedure

To configure a tag with a function list, proceed as follows:

1. Under "Properties > Events >" in the Inspector window, select the event for which you want to create a function list.
   The function list associated with the selected event is shown.
2. Click "<Add function>". The second table column contains a selection button.
3. Click the selection button and select a system function.
4. Define the parameter values.

Alternatively, select a script to run in the function list, rather than a system function. The "VB scripts" entry is only displayed if there is a script available in the project.

Result

If the configured event occurs in Runtime, the function list or the script is processed.

See also

Address multiplexing (Page 555)

Updating the Tag Value in Runtime (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Tags contain process values which change while Runtime is running. Value changes are handled differently at internal and external tags.

Principle

When Runtime starts, the value of a tag is equal to its start value. Tag values change in Runtime.

In Runtime, you have the following options for changing the value of a tag:

- A value change in an external tag in the PLC.
- By input, for example, in an I/O field.
- By running a system function, such as "SetValue."
- A value assignment in a script.

Updating the Value of External Tags

The value of an external tag is updated as follows:

- Cyclic in operation
  If you select the "Cyclic in operation" acquisition mode, the tag is updated in Runtime while it is displayed in a screen or is logged. The acquisition cycle determines the update cycle for tag value updates on the HMI device. You can either choose a default acquisition cycle or define a user-specific cycle.
- Cyclic continuous
  If you select the "Cyclic continuous" acquisition mode, the tag will be updated continuously in Runtime, even if it is not in the currently-open screen. This setting is activated for tags that are configured to trigger a function list when their value changes, for example.
  Only use the "Cyclic continuous" setting for tags that must truly be updated. Frequent read operations increase communication load.
- On demand
  If you select the "On demand" acquisition mode, the tag is not updated cyclically. It will only be updated on demand using the "UpdateTag" system function, for example, or by a script.

See also

Address multiplexing (Page 555)

Defining the acquisition cycle for a tag (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

The value of an external tag can be changed in Runtime by the PLC to which the tag is linked. To ensure that the HMI device is informed of any changes in tag values by the PLC, the values must be updated on the HMI. The value is updated at regular intervals while the tag is displayed in the process screen or is logged. The interval for regular updates is set with the acquisition cycle. The update can also be made continuous.

Requirement

- You have created the tag for which you want to define an acquisition cycle.
- The Inspector window with the tag properties is open.

Procedure

To configure an acquisition cycle for a tag, follow these steps:

1. In the Inspector window select "Properties > Properties > General."
2. If you want to update the tag at regular intervals as long as it is being displayed on the screen or logged, select "Cyclic in operation" as the acquisition mode.
   Or:
   If you want to update the tag at regular intervals even though it is not being displayed on the screen or logged, select "Cyclic continuous" as the acquisition mode.
   The "Cyclic continuous" setting is selected for a tag, for example, that has a function list configured for a change of its value and that is not directly visible in a screen.
3. Select the required cycle time in the "Acquisition cycle" field or define a new acquisition cycle using the object list.

Alternatively, you can configure the acquisition cycle directly in the work area of the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Note

Only use the "Cyclic continuous" acquisition mode for tags that really have to be continuously updated. Frequent read operations generate a heavy communication load.

Note

For structured HMI tags, the acquisition mode can be selected for the respective structured HMI tag as well as their individual subordinate elements. The acquisition mode of the elements of the array tag cannot be changed. The elements of the array tag is automatically assigned to the acquisition mode of the higher-level array tag.

When the acquisition mode of structured HMI tags is changed, it is applied to all subordinate elements. This means changing the acquisition mode may overwrite subordinate elements.

Result

The configured tag is updated in Runtime with the selected acquisition cycle.

See also

Address multiplexing (Page 555)

2.2.3.3 Special configuration options (RT Professional)

Defining a substitute value (RT Professional)

Introduction

You can define a specific value as a substitute value for a tag.

In the "Use substitute value" area you define when WinCC should use this substitute value. The current process value is then not accepted from the automation system.

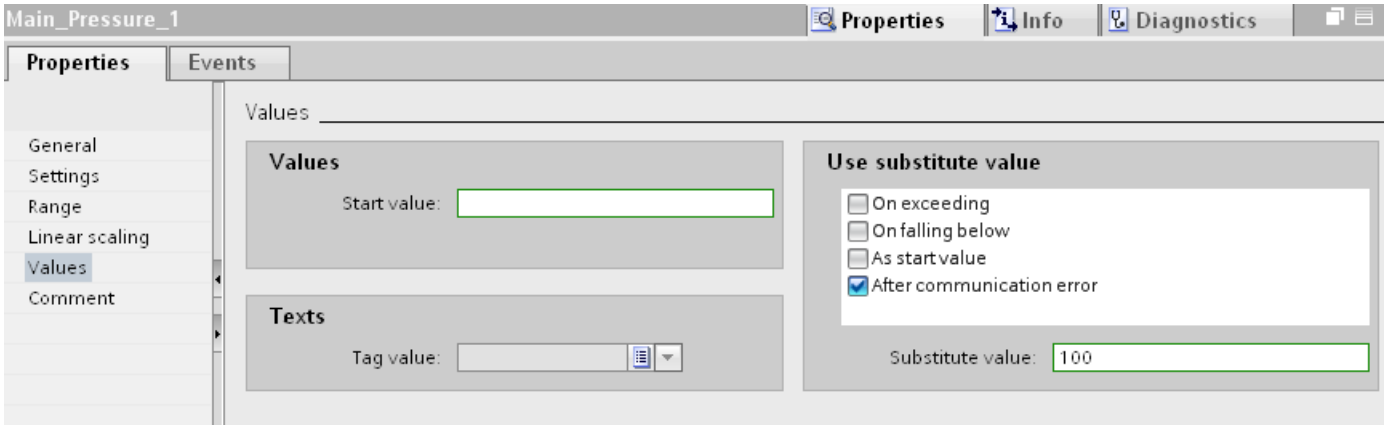

You can define a substitute value for the following cases:

- As start value
  WinCC sets the substitute value when the project is activated and there is no current process value.
- After communication error
  WinCC sets the substitute value when the connection to the automation system is disturbed and there is no valid process value.
- Maximum exceeded
  If you have set an upper limit for the tag, WinCC sets the substitute value as soon as the process value exceeds this upper limit.
- Minimum undershot
  If you have set a lower limit for the tag, WinCC sets the substitute value as soon as the process value falls below this lower limit.

Requirement

- The tag table is open.
- The Inspector window with the tag properties is open.

Procedure

To configure a substitute value, follow these steps:

1. Select the desired tag in the tag table, and select "Properties > Properties > Values" in the Inspector window.
2. In the "Use substitute value" segment, select when you want WinCC to use this substitute value in the tag.
3. Enter the required substitute value in the "Substitute value" field.

Result

The configured tag is populated with the substitute value in Runtime once the selected condition is fulfilled.

Note

If you have set a high or low limit in an I/O field, you cannot enter any value outside these limits.

WinCC ignores incorrect entries and therefore does not set a substitute value. The substitute value is only set by WinCC when an incorrect process value is read.

See also

Defining the acquisition cycle for a tag (Page 570)

Tag Limits (Page 547)

Indirect addressing of tags (Page 572)

Addressing tags indirectly (Page 572)

Defining the acquisition cycle for a tag (RT Professional)

Introduction

If you are using tags, you define the acquisition cycle for Runtime Professional at the object on which the tag is to be used.

If you are logging a tag, define the acquisition cycle at the connected logging tag. The connected tag is updated with this acquisition cycle.

If you use a tag on a graphic object, the tag is updated with the acquisition cycle of the graphic object. The graphic objects in a screen are usually updated with the update cycle of the screen. You configure the update cycles of the screen and the objects in it in the "Dynamization overview" editor. All the objects used in a screen that require a trigger are shown in the

"Dynamization overview" editor. Select the required acquisition cycle for each object. For a high-performance configuration, you should only configure an acquisition cycle for individual objects that deviates from the screen cycle if there truly is a need for it.

Requirement

- You have configured a screen.
- The screen contains an object on which a tag is configured.

Procedure

To define the acquisition cycle, follow these steps:

1. Select the screen from the project tree and select "Dynamization overview" from the shortcut menu.
   The "Dynamization overview" editor opens.

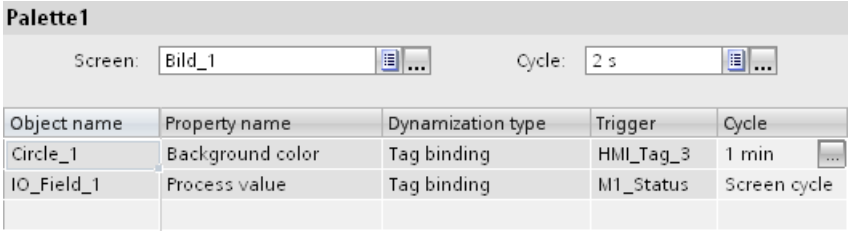

2. Configure the update cycle for the screen in the upper area of the "Dynamization overview" dialog. Select the desired update cycle for the screen in the "Cycle" field or define your own cycle using the object list. The set cycle time has an effect on all objects whose cycle is set to the value "Screen cycle".
3. If you want to configure a different cycle time for an individual object, select the desired object in the table of the editor and select the acquisition cycle in the "Cycle" field or define your own cycle using the object list.

If you want to change the acquisition cycle for an individual object only, select this object in the screen and select "Dynamization overview" from the shortcut menu.

Result

The configured object is updated in Runtime with the selected cycle.

See also

Defining a substitute value (Page 569)

Indirect addressing of tags (RT Professional)

Introduction

With indirect addressing, the tag used is first determined at runtime. Use a string tag for indirect addressing in WinCC Runtime Professional. You also need a series of tags whose values are to be displayed in Runtime. In Runtime you enter the name of the tag in the string tag whose value you want to display and process.

Application example

The operator enters the name of one or several machines in the input field. Depending on the operator input, a process value of the machine is displayed in an output field.

To configure such a scenario, configure the string tag with activation of indirect addressing for an I/O field. Use this I/O field to output the values. Configure the same tag to another I/O field without activation of indirect addressing. Use this I/O field to enter the machine names. Also configure a series of tags whose values you want to output.

When the operator enters the name of a different machine, the value of the string tag for indirect addressing changes. The output field shows the content of the tag whose name is included in the string tag.

You can use other mechanisms in WinCC to supply the string tag for indirect addressing, for example, a script.

See also

Addressing tags indirectly (Page 572)

Defining a substitute value (Page 569)

Addressing tags indirectly (RT Professional)

Introduction

With indirect addressing, the tag used is first determined during runtime. Instead of individual tags, you define a series of tags. In Runtime, the tag name is used to access individual tags.

Note

The dynamic interconnection of tags on the screen window

To interconnect tags dynamically on the screen window during runtime, use the tag prefix on the screen window or indirectly addressed tags. Avoid using both methods at the same time.

Requirements

- The "Temperature_Motor" tag that you want to use for indirect addressing does exist.
- A series of tags is created, for example, the "Motor1", "Motor2" and "Motor3" tags.

- The Inspector window with the tag properties is open.
- You have created a screen and the "Screens" editor is open.

Procedure

To address tags indirectly, proceed as follows:

1. Add an I/O field to the screen and select the following settings for this example in the information window under "Properties > Properties > General":
   - Mode: Output
   - Display format: Decimal
2. Open the object list in the "Tag" field using the selection button, select the "Temperature_Motor" tag and activate the "Use indirect addressing" option.

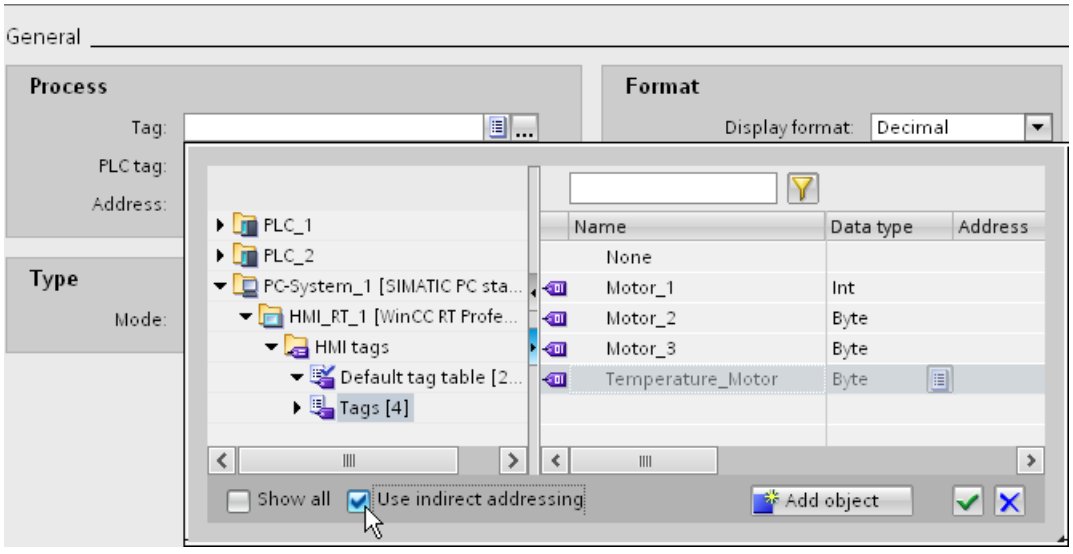

3. The "Temperature_Motor" tag is configured for indirect addressing. The additional text ("indirect") is displayed in front of the tag name in the "Tag" field.

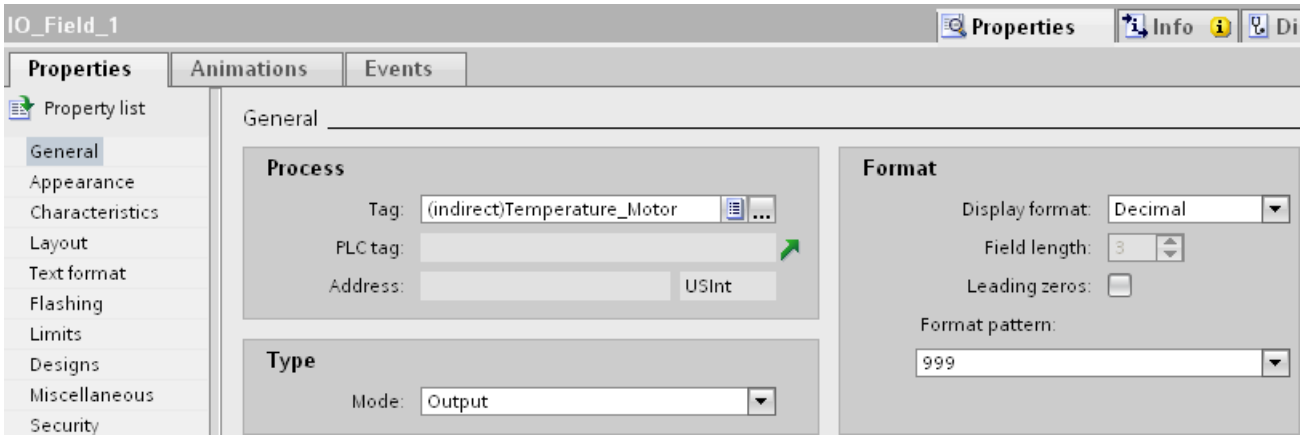

4. Add another I/O field to the screen and select the following settings for this example in the information window under "Properties > Properties > General":
   - Mode: Entry
   - Display format: Character string
5. Open the object list in the "Tag" field using the selection button, select the "Temperature_Motor" tag. Do not activate the option "Use indirect addressing".
6. Save the project.

Result

If you enter the name of one of the tags "Motor1", "Motor2" or "Motor3" in the input field in Runtime, the value of the respective tag is displayed in the output field.

See also

Indirect addressing of tags (Page 572)

Defining a substitute value (Page 569)

2.3 Working with Arrays

2.3.1 Basics on arrays

Definition

An array is a data list with data elements of a uniform data type. This data list is referred to as array tag or array for short. The data elements are referred to as array elements.

The individual array elements can be addressed by one or multiple integer indices. The properties of each array element are mainly identical and can be configured centrally at the array tag.

Default tag table

Name	Tag table	Data type	Connection
HMI_Tag_1	Default tag table	Array [0..4] of Int	<Internal tag>
[0]	Default tag table	Int	<Internal tag>
[1]	Default tag table	Int	<Internal tag>
[2]	Default tag table	Int	<Internal tag>
[3]	Default tag table	Int	<Internal tag>
[4]	Default tag table	Int	<Internal tag>

Advantages

Many similar array elements can be combined in a structured manner with only one array tag. You can specify the shared properties of similar array elements centrally at the array tag. You can then generally use each individual array element as individual tag with the same data type in the configuration.

Restrictions

When using arrays on HMI devices, RT Advanced and RT Professional, the following restrictions apply:

- An array can only contain one dimension. Multi-dimensional arrays of the PLCs are mapped to individual tags on the HMI systems.
- Only elementary data types without additional substructure can be used as data type of an array. This means that only scalable arrays are supported.
- The lower index of an array must begin with "0". The indices are moved accordingly for other low limits on the PLCs.
- Unlike individual tags, scalable arrays do not support ranges, linear scaling or tag multiplexing.

Application examples

Array tags can be used in the following situations:

- To group process values in profile trends: You map process values to trends which are acquired at different points in time, for example.
- To access specific values which are grouped in trends: For example, display all recorded values of the profile trend by gradually increasing the index tag.
- To configure discrete alarms with successive bit number.
- To save the complete machine data records in a recipe.

Tag count with arrays

- Scalable arrays on HMI devices and RT Advanced are counted as one tag.
- In RT Professional, each array element contained is counted separately.
- Multi-dimensional arrays of the PLCs are mapped to individual tags of the same data type and are counted as such.
- Structured arrays of the PLC are mapped to individual tags with the corresponding data type and are counted as such.

Special features

On HMI devices as well as in RT Advanced, all array elements are always read or written at once by the PLC when an element of a scalable array is accessed (except arrays of String/WString).

In RT Professional, the array elements are always read and written individually to and from the PLC.

Example for HMI devices and RT Advanced:

- An array tag which consists of 100 array elements of data type "Real" was configured.
- If an array element with a length of 4 bytes changes, 100 x 4 bytes are written to the PLC.

WARNING

Increased system load and performance losses

Due to the simultaneous transmission of all array elements of a scalable array, writing to the PLC, in particular, takes longer when accessing an element of a larger array than writing a tag with the same data type. Take this into consideration when designing the communication.

Also note that, with scalable arrays, reading entire array tags by the PLC is usually faster than the sum of read accesses to the same number of elementary tags of the same data type.

CAUTION

Data inconsistency at array tags

If the value of a single element must be changed in a scalable array tag, the entire array is read, changed and rewritten as a complete array. Changes to other array elements carried out in the meantime in the PLC are overwritten during rewriting.

You should always prevent the HMI device and the PLC from concurrently writing values to the same array tag. Use synchronous transfer of recipe data records, for example, to synchronize an array tag with the PLC.

Note

Array tags as list entry of multiplex tags

You can use the array tags of the Char or WChar data type just like the tags of the String data type.

The use of an array tag of the Char or WChar data type as list entry of a multiplex tag in the "HMI tags" editor is not supported.

Use in scripts

To maintain performance, always use internal arrays in scripts to change scalable arrays.

1. Copy the PLC array to the internal array at the start of the script.
2. No load is placed on communication to the PLC while the script processes the internal array.

CAUTION

Data inconsistency at array tags

If the value of a single element must be changed in an array tag, the whole array is read, changed and rewritten as a complete array. Changes to other array elements carried out in the meantime in the PLC are overwritten during rewriting.

You should always prevent the HMI device and the PLC from concurrently writing values to the same array tag. Use synchronous transfer of recipe data records, for example, to synchronize an array tag with the PLC.

See also

Creating array tags (Page 578)

Examples of arrays (Page 579)

Basics of tags (Page 519)

WinCC Runtime Advanced (Page 2379)

Basic Panel (Page 2361)

Mobile Panel (Page 2368)

Comfort Panel (Page 2373)

Basic Panel 2nd Generation (Page 2364)

2.3.2 Creating array tags

Introduction

Create an array tag to configure a large number of tags of the same data type. The array elements are saved to a consecutive address space.

You can create an array tag as an internal tag or as an external tag.

Note

Connecting HMI tags to PLC array tags

If you want to create an array tag as an external tag, first configure an array tag in a data block of the connected PLC. You then connect the array tag to an HMI tag.

Note

HMI array tags of the following data types are only supported with PLC array tags:

- Char
- WChar
- String
- WString

Creating HMI array tags

To create an array tag, follow these steps:

1. Open the HMI tag table.
2. Double click <Add> in the "Name" column of the HMI tag table.
   A new HMI tag is created.

Note

You can delete individual elements of the array tags via the shortcut menu.

Configuring HMI array tags

1. Click in the Data type column of the HMI tag table and select the "Array" data type.
2. Click ... in the data type column.
   The dialog box for configuring the array is opened.

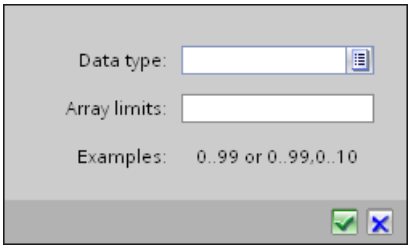

3. Select the desired data type for the array tag in the "Data type" field.

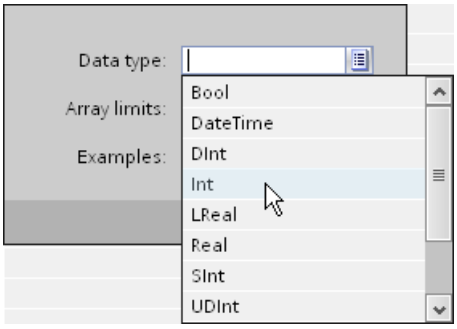

4. Define the number of array elements in the "Array limits" field. The lower limit must begin with "0".
5. Click . The settings for the array are saved.
6. As with tags with elementary data types, you still need to assign the address of the array tag in the PLC.
7. Save the project.

Result

An array tag is created. The properties of the array elements are inherited from the parent array tag.

See also

Basics on arrays (Page 575)

2.3.3 Examples of arrays

Introduction

Array tags combine a large number of tags, e.g. 100 array elements.

The common properties of array elements of the same kind can be defined centrally in the array tag. In the configuration, you then use each individual array element as an individual tag with the same data type. However, array elements cannot be used as parameters in function lists.

You use array tags as complete arrays in the following places:

- In the "Alarms" editor
- In the "Recipes" editor
- For address multiplexing
- In function lists

Examples

You can drag a PLC array tag straight from the PLC tag table to the HMI tag table.

Alternatively, you can configure an array tag separately with the corresponding number of array elements to handle multiple tags of the same data type.

- The individual array elements can be accessed indirectly by means of an index tag, for example.
- Use these index tags to operate and monitor the array elements.

See also

Basics on arrays (Page 575)

2.4 Working with Arrays (RT Professional)

2.4.1 Basics on arrays (RT Professional)

Definition

Individual array elements can be addressed via one or more integer indices. The properties of each array element are mainly identical and can be configured centrally at the array tag.

An array is a data list with data elements of a uniform data type. This data list is referred to as array tag or array for short. The data elements are referred to as array elements.

Default tag table

Name	Tag table	Data type	Connection
HMI_Tag_1	Default tag table	Array [0..4] of Int	<Internal tag>
[0]	Default tag table	Int	<Internal tag>
[1]	Default tag table	Int	<Internal tag>
[2]	Default tag table	Int	<Internal tag>
[3]	Default tag table	Int	<Internal tag>
[4]	Default tag table	Int	<Internal tag>

Advantages

You can group many array elements of the same type into structures using just one array tag. You can specify the common properties of the array elements of the same type centrally at the array tag. In the configuration, you then use each individual array element as individual tags with the same data type.

Restrictions

The following restrictions apply to the use of arrays:

- An array can contain only one dimension. Multi-dimensional arrays of controllers are mapped to individual tags on the HMI systems.
- The lower index of an array must begin with "0". The indices are shifted accordingly for other low limits on the controllers.
- Only elementary data types that are not substructured can be used as data type. Thus, only scalable arrays are supported.

- Unlike individual tags, scalable arrays do not support areas, linear scaling or tag multiplexing.
- External array tags are only possible with the S7-1200 and S7-1500 PLCs.

Note

Synchronization

To avoid structural conflicts in runtime, synchronize the tag names and replace the delimiters "[" and "]" for external array tags.

Configure "Settings for tags" for this in the runtime settings.

Tag counting with arrays

- Scalable arrays on HMI devices and RT Advanced are counted as one tag.
- In RT Professional, each contained array element is counted separately.
- Multi-dimensional arrays of controllers are mapped to individual tags of the same data type and counted like these.
- Structured arrays of the PLC are mapped to individual tags with the corresponding data type and counted like these.

Special features

On HMI devices and in RT Advanced, all array elements are read from or written to the PLC at the same time during access to an element of a scalable array (except arrays of String/WString).

In RT Professional, the array elements are always read and written individually to and from the PLC.

Multidimensional arrays are also possible in conjunction with PLC data types (UDT) of the S7-1200 and S7-1500 PLCs.

Note

Array tags as list entry of multiplex tags

You can use the array tags of the Char or WChar data type just like the tags of the String data type.

The use of an array tag of the Char or WChar data type as list entry of a multiplex tag in the "HMI tags" editor is not supported, however.

Example for writing an array element in the PLC

- An array tag which consists of 100 array elements of data type "Real" was configured.
- If an array element with a length of 4 bytes changes, 4 bytes are written to the PLC.

See also

Settings for synchronization (Page 544)

2.4.2 Creating array tags (RT Professional)

Introduction

Create an array tag to configure a large number of tags of the same data type. The array elements are saved to a consecutive address space.

You can create an array tag as an internal tag or as an external tag.

Note

Connecting HMI tags to PLC array tags

If you want to create an array tag as an external tag, first configure an array tag in a data block of the connected PLC. You then connect the array tag to an HMI tag.

Note

HMI array tags of the following data types are only supported with PLC array tags:

- Char
- WChar
- String
- WString

Creating HMI array tags

To create an array tag, follow these steps:

1. Open the HMI tag table.
2. Double click <Add> in the "Name" column of the HMI tag table.
   A new HMI tag is created.

Note

You can delete individual elements of the array tags via the shortcut menu.

Configuring HMI array tags

1. Click in the Data type column of the HMI tag table and select the "Array" data type.
2. Click ... in the data type column.
   The dialog box for configuring the array is opened.

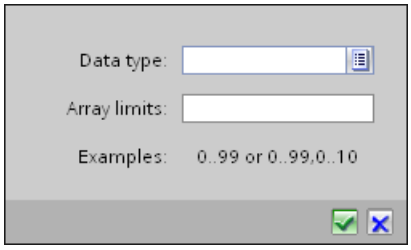

3. Select the desired data type for the array tag in the "Data type" field.

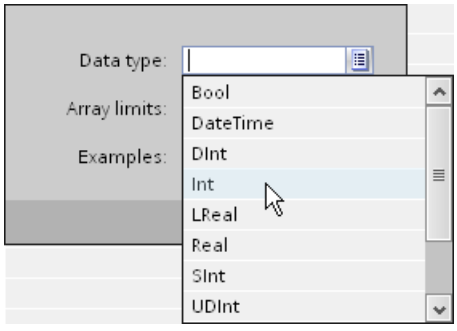

4. Define the number of array elements in the "Array limits" field. The lower limit must begin with "0".
5. Click . The settings for the array are saved.
6. As with tags with elementary data types, you still need to assign the address of the array tag in the PLC.
7. Save the project.

Result

An array tag is created. The properties of the array elements are inherited from the parent array tag.

2.5 Working with user data types (RT Professional, RT Advanced, Panels, Comfort Panels)

2.5.1 Basics on user data types (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

With user data types you bundle a number of different tags that form one logical unit. You create a user data type as a type and use instances of this type in the project. User data types are project-associated data and are available for all HMI devices of the project.

User data types differ in their association to a family of devices. User data types are available for the following families of devices:

- WinCC Runtime Advanced and Panels
- WinCC Runtime Professional

WinCC also supports the connection of PLC data types (UDTs) and user data types to HMITags.

User data types also differ in their applicability with a specific communication driver. User data types are available for the following communication drivers:

- SIMATIC S7-300/400
- SIMATIC S7-1200
- SIMATIC S7-1500

Create user data types and user data type elements in the project library.

Principle

For example, the different conditions of a motor can be described using 6 unique Boolean tags.

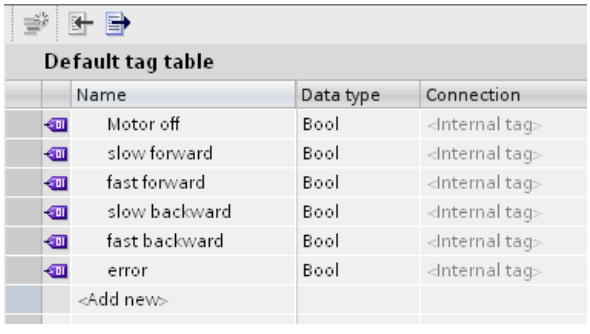

Default tag table

Name	Data type	Connection
Motor off	Bool	<Internal tag>
slow forward	Bool	<Internal tag>
fast forward	Bool	<Internal tag>
slow backward	Bool	<Internal tag>
fast backward	Bool	<Internal tag>
error	Bool	<Internal tag>
<Add new>		

If the overall condition should be described with a single tag, this tag can be created based on a user data type. For each of the individual Boolean tags you create a user data type element in the user data type.

This user data type can then be assigned complete to a faceplate for the motor. The created and released version of user data type is displayed at the tag in the "Data type" selection field.

The configured properties of a user data type are used in the instances of the user data type. If required, you change individual properties directly at the point of use, e.g. at a tag. Changing a property at the tag does not affect the user data type created.

NOTICE
Availability of user data types The connection of user data types to faceplates is only supported in WinCC Runtime Advanced and Panels.

License regulation in Runtime

If you use external tags of the "User data type" data type in a faceplate instance, each user data type element is counted as a tag in Runtime.

Example

You have created two screens in the "Screens" editor: Screen_1 and Screen_2.

3 instances of a faceplate type are inserted in Screen_1. 4 instances of a faceplate type are inserted in Screen_2.

Each faceplate instance is connected to an external tag of the "User data type" data type. The user data type includes 10 user data type elements.

Screen 1: 3 faceplate instances * 10 user data type elements corresponds to 30 external tags = 30 PowerTags.

Screen 2: 4 faceplate instances * 10 user data type elements corresponds to 40 external tags = 40 PowerTags.

In Runtime, 70 PowerTags are counted together for both screens. This also applies to the user data type elements that are not required.

See also

Creating tags with a user data type data type (Page 590)

Basics of tags (Page 519)

Creating user data type elements (Page 588)

2.5.2 Creating a user data type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You create a user data type in the project library.

Requirement

- A project is open.
- The "Libraries" task card is displayed or the library view is open.

Procedure

To create a user data type, follow these steps:

1. Click the "Libraries" task card.
2. Double-click the "Project library" item. The folder structure of the project library opens.
3. Click "Add new type".
   A dialog opens.

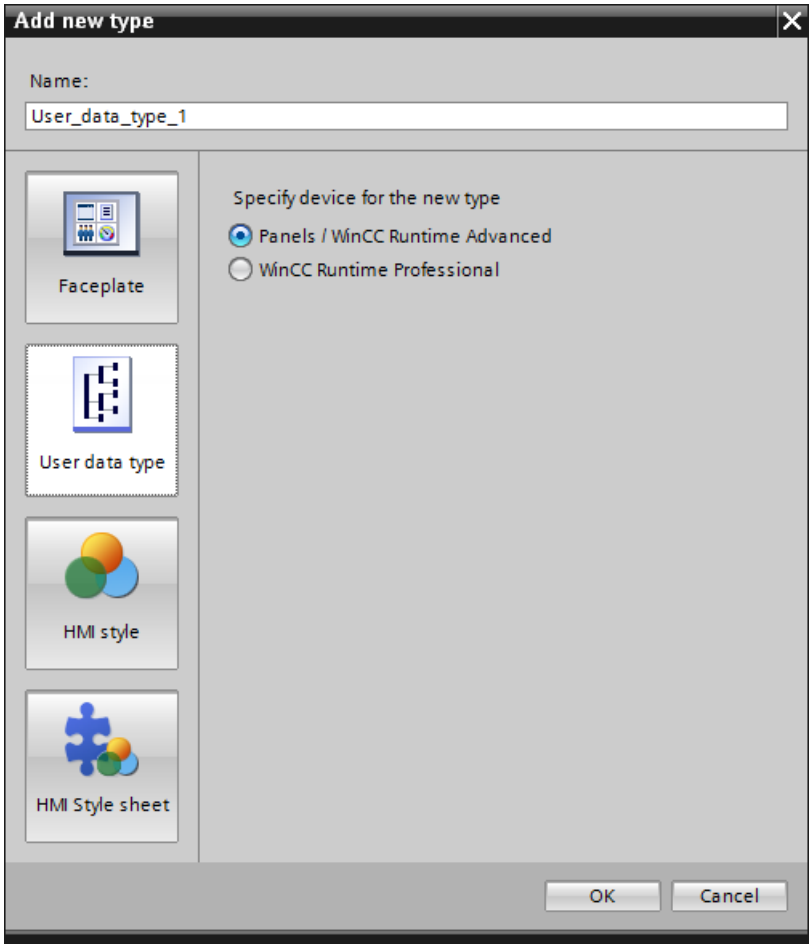

4. Click the "HMI UDT" button in the dialog.
5. In the "Specify device for the new type" area select the HMI device on which the user data type is used.
6. Enter a descriptive name in the "Name" field.
7. Click "OK" to apply your settings. The user data type is created. The library view opens.

Result

A user data type with the configured properties is created. Version 0.0.1 of the user data type is created and receives the status "In work".

Create the required user data type elements in the next step.

Before you use the version of user data types, for example at a tag, the version must be released.

2.5.3 Creating user data type elements (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You define user data type elements in the library view. You add or delete elements in the "HMI user data types" table in the work area.

Requirement

- The library view is open.
- A user data type is created and opened in the editor.

Procedure

1. Select a communication driver for the user data type.

- If you select the <Internal communication> entry, you can only assign the user data type to the internal tags as the data type.
- If a connection to a PLC is selected as the communication driver, the user data type can only be assigned to tags with a connection to this PLC as the data type.
- The communication type set applies to all user data type elements of a user data type. In a user data type for the"WinCC Runtime Professional" family of devices, you can define for each user data type element whether the configured driver is used for control or internal communication.

1. Double-click "Add" in the "Name" column of the table. A new user data type element is added to the user data type.
2. Assign a name.
3. Select the required data type in the "Data Type" field.
4. Create as many user data type elements as you need.
5. You configure the user data type elements in the "Properties" group in the Inspector window.

Alternatively, you can configure the properties of the user data type elements directly in the table. To view hidden columns, activate the column titles using the shortcut menu.

Result

You have added elements to version 0.0.1 of the user data type. The version 0.0.1 has the status "in progress".

To use the version in the project, release the version in the next step.

See also

Basics on user data types (Page 585)

2.5.4 Managing versions of user data types (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

All user data types have at least one version. When a user data type is created, a version is created at the same time and this version has the status "In work". You can edit the user data type in this status as required. When the editing is complete, you release the version of the user data type.

Requirement

- You have created a user data type.
- The user data type has the version 0.0.1. and the status "In work".
- The "Libraries" task card or the library view is open.

Releasing a version

1. Select the version 0.0.1 of the user data type in the project library.
2. Select "Release version" in the shortcut menu.
   A dialog opens.
3. If necessary, change the properties of the version:
   – Enter a name for the type in the "Name" field.
   – In the "Version" field, define a main and an intermediate version number for the version to be released.
   – To clean up version management of the type, enable "Delete unused type versions from the library".

You have released version 0.0.1 of the user data type.

Editing a version

1. Select, for example, the released version 0.0.1 of a user data type in the project library.
2. Select "Edit type" in the shortcut menu.

The library view opens. The new version 0.0.2 of the user data type is created.

The version has the status "in work".

Restoring the last version of a user data type

The last released version of the user data type is version 0.0.2.

You edit the user data type. A new version of the user data type, 0.0.3, is generated and receives the status "In work".

1. Select the version of the user data type in the project library.
2. Select "Discard changes and delete version" in the shortcut menu.

Alternatively

1. If you have opened a version for editing, click "Discard changes and delete version" in the toolbar.
   The version is deleted.

All changes to the user data type since the last release operation are discarded. The user data type is released again and has version 0.0.2.

Deleting user data type

If you delete a user data type, all instances and master copies of this user data type are deleted as well. The same is true for HMI tags that use an HMI user data type.

To delete a user data type, follow these steps:

1. In the project library, under "Types", select the user data type you want to delete.
2. Select "Delete" from the shortcut menu.

2.5.5 Creating tags with a user data type data type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

When a tag is created, you assign the version of user data type as a data type. In the "Tag" editor you can create internal tags or tags with a link to a PLC. A tag has access to all user data type versions that use the same communication driver as the tag itself.
In connection with a PLC, a user data type can only be used if the absolute addressing is selected.

Requirement

- A user data type with a user data type element is created.
- The user data type is enabled.

- The tag table is open.
- The Inspector window with the tag properties is open.

Procedure

To create a tag of the "User data type" data type, follow these steps:

1. In the "Name" column, double-click "Add" in the tag table. A new tag is created.
2. In the Inspector window select "Properties > Properties > General".
3. Enter a unique tag name in the "Name" field.
4. Select the connection to the required PLC in the "Connection" box.
5. Select the desired version of the user data type in the "Data type" field.
   The selected connection determines which user data types will be displayed.
   For internal tags, only user data type versions of the <Internal user data type> type are available.

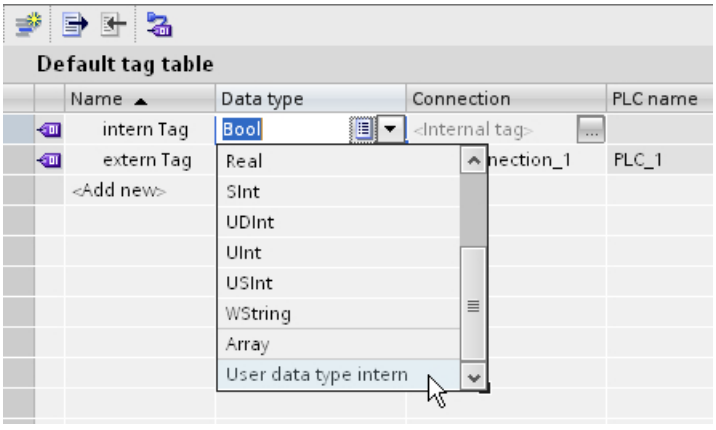

   For tags with a connection to a PLC, only those user data types that have a link to a PLC can be accessed.

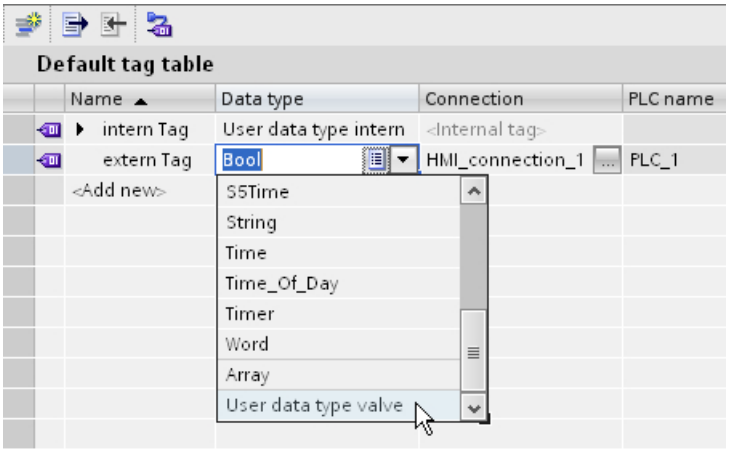

6. Enter the address of the PLC that you want to access with the external tags in the "Address" field of the "Settings" area. The specified address must always point to the start data bit, for example, <DB1.DBX0.0>.

Result

You have created a tag of the "User data type" data type. In additional steps you can configure the tag, for example, by setting the start value and limits.

With tags of the "User data type" data type, you can connect the dynamic properties of a faceplate instance, which then supplies these properties with values in Runtime.

If you wish to change the properties of a user data type tag, you must change the properties of the user data type element.

Properties such as "Start value", "Use substitute value" or " Linear scaling" can be changed in the employed instances of the user data type.

See also

Basics on user data types (Page 585)

2.6 Working with cycles (RT Advanced, Basic Panels, Panels, Comfort Panels)

2.6.1 Cycle basics (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Cycles are used to control actions that regularly occur in runtime. Common applications are the acquisition cycle, the logging cycle and the update cycle. You can also define your own cycles in addition to those already provided in WinCC.

Principle

In Runtime, actions that are performed regularly are controlled by cycles. Typical applications for cycles:

- Acquisition of external tags
  The acquisition cycle determines when the HMI device will read the process value of an external tag from the PLC. Set the acquisition cycle to suit the rate of change of the process values. The temperature of an oven, for example, changes much more slowly than the speed of an electrical drive.
  Do not set the acquisition cycle too low, since this will unnecessarily increase the communication load of the process.
- Logging process values
  The logging cycle determines when a process value is saved in the logging database. The logging cycle is always an integer multiple of the acquisition cycle.

The smallest possible value for the cycle depends on the HMI device that will be used in your project. For most HMIs, this value is 100 ms. The values of all other cycles are always an integer multiple of the smallest value.

If the cycles provided in WinCC do not meet the needs of your project, you can define your own cycles using the "Cycles" editor. User-defined cycles must always be an integer multiple of the smallest value.

Note

Device dependency

You cannot configure self-defined cycle times for Basic Panels.

Application example

You can use cycles for the following tasks:

- To regularly log a process.
- To draw attention to maintenance intervals.

Updating external tags in Runtime

If you select the "Cyclic in operation" acquisition mode to update the external tags in Runtime for Comfort Panels and the external tag is not being used in the current screen, the tag will not always contain the current value. In the following cases, it is therefore necessary to select the "Cyclic continuous" acquisition type:

- In a tooltip that displays limits when editing the recipe element. This means that the tags of these limits must be set to "Cyclic continuous" if the limits of recipe elements (recipe tags) are dynamically assigned ("Upper 2" and "Lower 2" with tags).
- The trigger tags of HMI alarms (discrete alarms and analog alarms) are automatically changed to "Cyclic continuous". Resetting this is not allowed.
- When using chained text lists for dynamic parameters of HMI alarms (discrete alarms and analog alarms), the tags of the second and all other subordinate levels must be changed to "Cyclic continuous".
- For "bit-triggered" trends, the control tags Trend request, Trend transfer 1, and Trend transfer 2 must be set to "Cyclic continuous".
- In VB scripts, if the tag name is formed dynamically by linking strings.

See also

Defining cycles (Page 594)

Basics of tags (Page 519)

2.6.2 Defining cycles (RT Advanced, Panels, Comfort Panels)

Introduction

Use cycles to control actions that are run at regular intervals in Runtime. You can also define your own cycles in addition to those already provided in WinCC.

Requirement

You have opened the project.

Procedure

1. Double-click the "Cycles" entry in the project navigation.
   The "Cycles" editor opens.
2. In the "Name" column of the "Cycles" editor, double-click "Add".
   A new cycle time is created.
3. Enter a unique name in the "Name" field.
4. Select the desired cycle unit.

5. Select the desired value for the cycle time.
   The available selection of values varies depending on the cycle unit selected. The smallest possible value for the cycle depends on the HMI device that will be used in your project. For most HMIs, this value is 100 ms. The available values are always an integer multiple of this value.
6. As an option, you can enter a comment regarding the use of the cycle.

Result

The cycle you configured is created and beside the default cycles in WinCC for use during configuration.

See also

Cycle basics (Page 593)

2.7 Working with cycles (RT Professional)

2.7.1 Cycle basics (RT Professional)

Introduction

Cycles are used to control actions that regularly occur in Runtime. Common applications are the acquisition cycle, the logging cycle and the screen cycle. You can also define your own cycles in addition to those already provided in WinCC.

Principle

In Runtime, actions that are performed regularly are controlled by cycles. Typical applications for cycles:

- Refreshing screens
  The screen cycle determines how often a screen will be refreshed. The screen cycle gives you the option to trigger all dynamizations in a screen at the same time. Use the predefined cycles in WinCC or self-defined cycles as cycle times. The standard setting for the screen cycle is 2 seconds.
- Updating objects
  With the default setting, all dynamic properties in a screen use the screen cycle. You can use other cycles to update individual objects, if necessary. See Basics on dynamizing screens (Page 205) for additional information.
- Triggering scheduled tasks
  In scheduled tasks you have the option to configure a task with a cyclical trigger. Use the cycle time to determine when the scheduled task is executed. See Triggers (Page 1796) for additional information
- Logging process values
  The logging cycle determines when a process value is saved in the logging database. The logging cycle is always an integer multiple of the acquisition cycle. See Cycles and Events (Page 640) for additional information.

The smallest values for a cycle in Runtime Professional are 100 and 125 ms. The next possible values are 250 and 300 ms. You can configure all other values with an increment of 100 ms.

Note

Cycle value 500 ms

The cycle value of 500 ms is available only for logging of process values.

Note

Cycle value 125 ms

The cycle value of 125 ms is only supported in Runtime scripting by the "ActivateDynamic" method. You define all other cycles in the "Cycles" editor.

Note

Defining your own cycles

If the cycles predefined in WinCC do not meet the requirements of your project, you can define your own cycles using the "Cycles" editor.

You can create an unlimited number of cycles and use these for configuring the logging. Keep in mind that you can only monitor up to 15 different update cycles in Runtime.

Note

Using cycles in screens

Use the 15 system-defined cycles to dynamize the object and screen properties and as screen cycle.

Starting point

You can define a starting point for every cycle in WinCC. This starting point will also apply to the cycles provided with the system. The following starting points are possible:

- Trigger cycle at start of Runtime
- Trigger cycle at system shutdown
- Trigger cycle at user-defined starting point

You can use cycle starting points for the following tasks:

- To synchronize different tasks within the system
- To relieve the load on the communication channels by postponing tasks with a high communication load.

Application example

You can use cycles for the following tasks:

- To regularly log a process.
- To draw attention to maintenance intervals.

See also

Basics of tags (Page 519)

Define starting point (Page 598)

Basics on dynamizing screens (Page 205)

Triggers (Page 1796)

Cycles and Events (Page 640)

2.7.2 Defining cycles (RT Professional)

Introduction

Use cycles to control actions that are run at regular intervals in Runtime. You can also define your own cycles in addition to those already provided in WinCC.

Requirement

The project is open.

Procedure

1. Double-click the "Cycles" entry in the project navigation.
   The "Cycles" editor opens.
2. In the "Name" column of the "Cycles" editor, double-click "Add".
   A new cycle time is created.
3. Enter a unique name in the "Name" field.
4. Select the desired cycle unit.
5. Select the desired value for the cycle time.
   The available selection of values varies depending on the cycle unit selected.
6. As an option, you can enter a comment regarding the use of the cycle.

Result

The cycle you configured is created and beside the default cycles in WinCC for use during configuration.

See also

Cycle basics (Page 596)

2.7.3 Define starting point (RT Professional)

Introduction

For cycles, define the starting points to start different cycles at different times. Staggering the start of the update processes will also stagger the communication load on the connection to the PLC.

Requirement

- The "Cycles" editor is open.
- The Inspector window with the cycle properties is open.

Procedure

To define the starting point for a cycle, follow these steps:

1. Select the cycle from the table in the "Cycles" editor.
2. Click "Properties > Properties > Starting point" in the Inspector window.
   The properties for the starting point of the cycle are displayed.
3. Select the settings for the starting point.
4. Save the project.

Result

The selected cycle is started in Runtime at the configured starting point.

See also

Cycle basics (Page 596)

2.8 Reference (RT Professional)

2.8.1 Mapping and coding of data types (RT Professional)

Introduction

WinCC sets the data type of an external tag according to the data type of the connected PLC tag. If the data type of the PLC tag is not available in WinCC, a compatible data type is automatically used at the HMI tag. As required, you can specify that WinCC uses a different data type and adapts the format of the data type of the PLC tag and the data type of the HMI tag. The following sections include the data types that are typically used for mapping. You will also find optional mapping versions that you can set yourself.

Coding of data types

In addition to mapping of data types, you can also change the coding of many mapping versions. Coding is used to map and process the binary information of PLC tag according to the requirements. The values of the PLC tags can be mapped with the following coding:

BCD

The BCD code is a binary decimal code that maps each individual digit of a number in binary form. The value of the digits is according to the number sequence 8-4-2-1. The result is the following value table:

Decimal digit	BCD code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Aiken

The Aiken code is a complementary BCD code. The value of the digits is according to the number sequence 2-4-2-1. The result is the following value table:

Decimal digit	Aiken code
0	0000
1	0001
2	0010
3	0011
4	0100

Decimal digit	Aiken code
5	1011
6	1100
7	1101
8	1110
9	1111

Excess

Excess-3 code is used for Excess coding. The Excess-3 code is a complementary BCD code. The value of the digits is according to the number sequence 8-4-(-2)-(-1). The result is the following value table:

Decimal digit	Excess code
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

ExtBCD

The "ExtBCD" coding used the BCD code described above for coding. But with "ExtBCD" the Most Significant Bit is interpreted as sign.

MSB

The "MSB" coding uses the binary code standard for coding. But with "MSB" the Most Significant Bit is interpreted as sign.

NanoToMilliseconds

The PLC tags with the data types "LTime" and "LTime_Of_Day" contain a number of nanoseconds. However, since HMI tags do not process nanoseconds, these are converted into milliseconds by the coding "NanoToMilliseconds" in the HMI runtimes:

Value of the PLC tag with PLC data type "LTime" or "LTime_Of_Day"	Converted value in the HMI tags with the HMI data type "DInt" or "UDInt"
1000000	1
2000000	2
10000000	10
100000000	100
...	...

See also

Coding and value range of data types (Page 602)
Adapting the data type of a tag (Page 536)

2.8.2 Coding and value range of data types (RT Professional)

Coding and value range

Depending on the selected coding, the value ranges listed in the tables are available for the different data types.

PLC data type Byte

Conversion	Value range
Byte to UnsignedByte	0...255
Byte to UnsignedWord	0...255
Byte to UnsignedDword	0...255
Byte to SignedByte	0...127
Byte to SignedWord	0...255
Byte to SignedDword	0...255
Byte to BCDByte	0...99
Byte to BCDWord	0...255
Byte to BCDDword	0...255
Byte to AikenByte	0...99
Byte to AikenWord	0...255
Byte to AikenDword	0...255
Byte to ExcessByte	0...99
Byte to ExcessWord	0...255
Byte to ExcessDword	0...255

PLC data type Char

Conversion	Value range
Char to UnsignedByte	0...127
Char to UnsignedWord	0...127
Char to UnsignedDword	0...127
Char to SignedByte	-128...+127
Char to SignedWord	-128...+127
Char to SignedDword	-128...+127
Char to MSBByte	-128...+127
Char to MSBWord	-128...+127
Char to MSBDword	-128...+127
Char to BCDByte	0...99
Char to BCDWord	0...127

Conversion	Value range
Char to BCDDword	0...127
Char to SignedBCDByte	-9...+9
Char to SignedBCDWord	-128...+127
Char to SignedBCDDword	-128...+127
Char to ExtSignedBCDByte	-79...+79
Char to ExtSignedBCDWord	-128...+127
Char to ExtSignedBCDDword	-128...+127
Char to AikenByte	0...99
Char to AikenWord	0...127
Char to AikenDword	0...127
Char to SignedAikenByte	-9...+9
Char to SignedAikenWord	-128...+127
Char to SignedAikenDword	-128...+127
Char to ExcessByte	0...99
Char to ExcessWord	0...127
Char to ExcessDword	0...127
Char to SignedExcessByte	-9...+9
Char to SignedExcessWord	-128...+127
Char to SignedExcessDword	-128...+127
Char to UnsignedByte	0...127
Char to UnsignedWord	0...127
Char to UnsignedDword	0...127

PLC data type Real

Conversion	Value range
Real to Real	+-1.79769313486231e+308
Real to UnsignedByte	0...255
Real to UnsignedWord	0...65535
Real to UnsignedDword	0...4294967295
Real to SignedByte	-128...+127
Real to SignedWord	-32768...+32767
Real to SignedDword	-2147483647...+2147483647
Real to Float	+-3.402823e+38
Real to MSBByte	-127...+127
Real to MSBWord	-32767...+32767
Real to MSBDword	-2147483647...+2147483647
Real to BCDByte	0...99
Real to BCDWord	0...9999
Real to BCDDword	0...99999999
Real to SignedBCDByte	-9...+9
Real to SignedBCDWord	-999...+999
Real to SignedBCDDword	-9999999...+9999999

Conversion	Value range
Real to ExtSignedBCDByte	-79...+79
Real to ExtSignedBCDWord	-7999...+7999
Real to ExtSignedBCDDword	-79999999...+79999999
Real to AikenByte	0...99
Real to AikenWord	0...9999
Real to AikenDword	0...99999999
Real to SignedAikenByte	-9...+9
Real to SignedAikenWord	-999...+999
Real to SignedAikenDword	-9999999...+9999999
Real to ExcessByte	0...99
Real to ExcessWord	0...9999
Real to ExcessDword	0...99999999
Real to SignedExcessByte	-9...+9
Real to SignedExcessWord	-999...+999
Real to SignedExcessDword	-9999999...+9999999
Real to S5Timer	10...9990000
Real to S5Float	+-1.701411e+38

PLC data type DWord

Conversion	Value range
DWord to UnsignedDword	0...4294967295
DWord to UnsignedByte	0...255
DWord to UnsignedWord	0...65535
DWord to SignedByte	0...127
DWord to SignedWord	0...32767
DWord to SignedDword	0...2147483647
DWord to BCDByte	0...99
DWord to BCDWord	0...9999
DWord to BCDDword	0...99999999
DWord to AikenByte	0...99
DWord to AikenWord	0...9999
DWord to AikenDword	0...99999999
DWord to ExcessByte	0...99
DWord to ExcessWord	0...9999
DWord to ExcessDword	0...99999999
DWord to SimaticTimer	10...9990000
DWord to SimaticBCDTimer	10...9990000
DWord to UnsignedDword	0...4294967295
DWord to UnsignedByte	0...255
DWord to UnsignedWord	0...65535

PLC data type Float

Conversion	Value range
Float to Float	+-3.402823e+38 (no conversion)
Float to UnsignedByte	0...255
Float to UnsignedWord	0...65535
Float to UnsignedDword	0 to 4.294967e+09
Float to SignedByte	-128...+127
Float to SignedWord	-32768...+32767
Float to SignedDword	-2.147483e+09 to +2.147483e+09
Float to Real	+-3.402823e+38
Float to MSBByte	-127...+127
Float to MSBWord	-32767...+32767
Float to MSBDword	-2.147483e+09 to +2.147483e+09
Float to BCDByte	0...99
Float to BCDWord	0...9999
Float to BCDDword	0...9.999999e+07
Float to SignedBCDByte	-9...+9
Float to SignedBCDWord	-999...+999
Float to SignedBCDDword	-9999999...+9999999
Float to ExtSignedBCDByte	-79...+79
Float to ExtSignedBCDWord	-7999...+7999
Float to ExtSignedBCDDword	-7.999999e+07 to +7.999999e+07
Float to AikenByte	0...99
Float to AikenWord	0...9999
Float to AikenDword	0...9.999999e+07
Float to SignedAikenByte	-9...+9
Float to SignedAikenWord	-999...+999
Float to SignedAikenDword	-9999999...+9999999
Float to ExcessByte	0...99
Float to ExcessWord	0...9999
Float to ExcessDword	0...9.999999e+07
Float to SignedExcessByte	-9...+9
Float to SignedExcessWord	-999...+999
Float to SignedExcessDword	-9999999...+9999999
Float to S5Timer	10...9990000
Float to S5Float	+-1.701411e+38

PLC data type Long

Conversion	Value range
Long to SignedDword	-2147483647...+2147483647
Long to UnsignedByte	0...255
Long to UnsignedWord	0...65535
Long to UnsignedDword	0...2147483647

Conversion	Value range
Long to SignedByte	-128...+127
Long to SignedWord	-32768...+32767
Long to MSBByte	-127...+127
Long to MSBWord	-32767...+32767
Long to MSBDword	-2147483647...+2147483647
Long to BCDByte	0...99
Long to BCDWord	0...9999
Long to BCDDword	0...99999999
Long to SignedBCDByte	-9...+9
Long to SignedBCDWord	-999...+999
Long to SignedBCDDword	-9999999...+9999999
Long to ExtSignedBCDByte	-79..+79
Long to ExtSignedBCDWord	-7999...+7999
Long to ExtSignedBCDDword	-79999999...+79999999
Long to AikenByte	0...99
Long to AikenWord	0...9999
Long to AikenDword	0...99999999
Long to SignedAikenByte	-9...+9
Long to SignedAikenWord	-999...+999
Long to SignedAikenDword	-9999999...+9999999
Long to ExcessByte	0...99
Long to ExcessWord	0...9999
Long to ExcessDword	0...99999999
Long to SignedExcessByte	-9...+9
Long to SignedExcessWord	-999...+999
Long to SignedExcessDword	-9999999...+9999999
Long to SimaticTimer	10...9990000
Long to SimaticBCDTimer	10...9990000

PLC data type Short

Conversion	Value range
Short to UnsignedByte	0...255
Short to UnsignedWord	0...32767
Short to UnsignedDword	0...32767
Short to SignedByte	-128...+127
Short to SignedWord	-32768...+32767
Short to SignedDword	-32768...+32767
Short to MSBByte	-127...+127
Short to MSBWord	-32767...+32767
Short to MSBDword	-32768...+32767
Short to BCDByte	0...99
Short to BCDWord	0...9999

Conversion	Value range
Short to BCDDword	0...32767
Short to SignedBCDByte	-9...+9
Short to SignedBCDWord	-999...+999
Short to SignedBCDDword	-32768...+32767
Short to ExtSignedBCDByte	-79...+79
Short to ExtSignedBCDWord	-7999...+7999
Short to ExtSignedBCDDword	-32768...+32767
Short to AikenByte	0...99
Short to AikenWord	0...9999
Short to AikenDword	0...32767
Short to SignedAikenByte	-9...+9
Short to SignedAikenWord	-999...+999
Short to SignedAikenDword	-32768...+32767
Short to ExcessByte	0...99
Short to ExcessWord	0...9999
Short to ExcessDword	0...32767
Short to SignedExcessByte	-9...+9
Short to SignedExcessWord	-999...+999
Short to SignedExcessDword	-32768...+32767
Short to UnsignedByte	0...255
Short to UnsignedWord	0...32767
Short to UnsignedDword	0...32767

PLC data type Word

Conversion	Value range
Word to UnsignedWord	0...65535
Word to UnsignedByte	0...255
Word to UnsignedDword	0...65535
Word to SignedByte	0...127
Word to SignedWord	0...32767
Word to SignedDword	0...65535
Word to BCDByte	0...99
Word to BCDWord	0...9999
Word to BCDDword	0...65535
Word to AikenByte	0...99
Word to AikenWord	0...9999
Word to AikenDword	0...65535
Word to ExcessByte	0...99
Word to ExcessWord	0...9999
Word to ExcessDword	0...65535
Word to SimaticCounter	0...999
Word to SimaticBCDCounter	0...999

Conversion	Value range
Word to UnsignedWord	0...65535
Word to UnsignedByte	0...255
Word to UnsignedDword	0...65535

See also

Mapping and coding of data types (Page 600)

2.8.3 Data types (RT Professional)

2.8.3.1 Data types for SIMATIC S7-300/400 (RT Professional)

Overview

The following table shows the data types for SIMATIC S7-300/400 with the corresponding HMI data types and value ranges in WinCC.

PLC data type	HMI data type	Value range
Bool	Bool	0 (FALSE), 1 (TRUE)
Byte	USInt	0 ... 255
Word	UInt	0 ... 65 535
DWord	UDInt	0 ... 4294967295
Char	USInt	0 ... 255
Int	Int	−32768 … 32767
DInt	DInt	−2147483648 … +2147483647
Real	Real	±1.17549E-38 to ±3.40282E+38 and 0.0
Time	DInt	−2147483648 … +2147483647
Date	DateTime/UInt	0 … 65535
Time_of_Day, TOD	DateTime/UDInt	0 … 86399999
S5Time	UDInt	0 … 9990000
Counter	UInt	0 ... 999
Timer	UDInt	0 … 9990000
String	String	-

2.8.3.2 Data types for SIMATIC S7-1200 and S7-1500 (RT Professional)

Data types for SIMATIC S7-1200 - V2 (RT Professional)

Overview

The following table shows the data types for SIMATIC S7-1200 - V2 with the corresponding HMI data types and value ranges in WinCC.

PLC data type	HMI data type	Value range
Bool	Bool	0 (FALSE), 1 (TRUE)
SInt	SInt	-128 ... +127
Int	Int	-32768 ... +32767
DInt	DInt	−2147483648 … +2147483647
USInt	USInt	0 ... 255
UInt	UInt	0 … 65535
UDInt	UDInt	0 ... 4294967295
Real	Real	±1.17549E-38 to ±-3.40282E +38 and 0.0
LReal	LReal	±1.79769313486231E+308 to ±2.22507385850720E-308 and 0.0
Time	DInt	−2147483648 … +2147483647
Date	DateTime/UInt	0 … 65535
DTL	DateTime	-
Time_of_Day, TOD	DateTime/UDInt	0 … 86399999
String	String	-
Char	USInt	0 ... 255
Array of Char	String	-
Byte	USInt	0 ... 255
Word	UInt	0 … 65535
DWord	UDInt	0 ... 4294967295
PlcUDT	PlcUDT	

Data types for SIMATIC S7-1500 (RT Professional)

Overview

The following table shows the data types for SIMATIC S7-1500 with the corresponding HMI data types and value ranges in WinCC.

PLC data type	HMI data type	Value range
Bool	Bool	0 (FALSE), 1 (TRUE)
SInt	SInt	-128 ... +127

PLC data type	HMI data type	Value range
Int	Int	-32768 ... +32767
DInt	DInt	−2147483648 … +2147483647
LInt	LReal	-4,503,599,627,370,495 … +4,503,599,627,370,495
USInt	USInt	0 ... 255
UInt	UInt	0 … 65535
UDInt	UDInt	0 ... 4294967295
ULInt	LReal	0 … 4,503,599,627,370,495
Real	Real	±1.17549E-38 to ±-3.40282E +38 and 0.0
LReal	LReal	±1.79769313486231E+308 to ±2.22507385850720E-308 and 0.0
S5Time	UDInt	0 … 9990000
Time	DInt	−2147483648 … +2147483647
LTime	DInt	−2147483648 … +2147483647
Date	DateTime/UInt	0 … 65535
Time_of_Day, TOD	DateTime/UDInt	0 … 86399999
LTime_of_Day, LTOD	DateTime/UDInt	0 … 86399999
Date_And_Time, DT	DateTime	-
LDT	DateTime	-
DTL	DateTime	-
String	String	-
WString	WString	-
Char	USInt	0 ... 255
WChar	WString	-
Byte	USInt	0 ... 255
Word	UInt	0 … 65535
DWord	UDInt	0 ... 4294967295
Timer	UDInt	0 … 9990000
Counter	UInt	0 ... 999
PLCUDT	PLCUDT	

Note

The 64-bit PLC data types LINT, UINT and LWORD are mapped to the HMI data type LREAL in the HMI channel. A loss of accuracy can be expected when the value exceeds the size 2^50.

User-defined PLC data types (UDT) (RT Professional)

Overview

You can connect with the HMI tags and DB instances of user-defined PLC data types (UDT).

The PLC data type and the corresponding DB instances are created and updated centrally in STEP 7. In WinCC, you can use the following sources as PLC tag (DB instances):

- Data block elements that use a UDT as data type
- Instance data blocks of a UDT

The data type is taken from STEP 7 and is not converted into an HMI data type. The access type is always "Symbolic access".

Elements of a PLC data type

You have access to the following elements in WinCC with a structured PLC data type:

- Elements that have been released for WinCC in STEP 7.
- Elements whose data types are supported in WinCC.

Note

Invalid elements of a PLC data type in WinCC

Invalid elements generate an error in WinCC.

If you disable the "Accessible from HMI" option for the corresponding elements of the associated PLC data type in STEP 7, these elements are excluded in WinCC.

Naming conventions

The following characters are invalid in the name of the PLC data type and generate an error in WinCC:

- Period: "."
- Brackets: "(" and ")"

Properties

The properties of the PLC data type and its elements are adopted in WinCC. Depending on the data type used, the properties are read-only or can be written to in WinCC.

If you change the connection of the PLC data type in WinCC, all elements of the PLC data type are deleted and the properties of the newly connected PLC tag are used.

In WinCC, you have access to STEP 7 comments on elements of the PLC data type.

You have limited access to properties in WinCC for the following elements of PLC data types:

- Elements of the data type "Struct"
- PLC data type

- Multidimensional arrays
- Array of complex data types except "DTL"

Mapping of the data type "DTL"

If a PLC data type contains elements of the data type "DTL", these elements are mapped in WinCC without lower-level elements. The data type "DTL" turns into "DateTime" in WinCC.

Tags with elements of the "DTL" data type

Tags that use the "DTL" data type element by element can only be used as read-only with symbolic addressing, e.g. with SIMATIC S7 1200 and SIMATIC S7 1500. With absolute addressing, write access is also possible.

2.8.3.3 Data types for OPC DA (RT Professional)

Overview

The following table shows the data types for OPC DA with the corresponding HMI data types and value ranges in WinCC.

PLC data type	HMI data type	Value range
VT_EMPTY	DInt	4
VT_BOOL	Bool	1
VT_I1	SInt	1
VT_UI1	USInt	1
VT_I2	Int	2
VT_UI2	UInt	2
VT_I4	DInt	4
VT_UI4	UDInt	4
VT_R4	Real	4
VT_R8	LReal	8
VT_BSTR	WString	Number of characters
VT_DATE | VT_ARRAY (Array [lo..hi] of VT_DATE)	Raw / Array [lo..hi] of DateTime	-
VT_I1 | VT_ARRAY (Array [lo..hi] of VT_I1)	Raw / Array [lo..hi] of SInt	-
VT_UI1 | VT_ARRAY (Array [lo..hi] of VT_UI1)	Raw / Array [lo..hi] of USInt	-
VT_I2 | VT_ARRAY (Array [lo..hi] of VT_I2)	Raw / Array [lo..hi] of Int	-
VT_UI2 | VT_ARRAY (Array [lo..hi] of VT_UI2)	Raw / Array [lo..hi] of UInt	-
VT_I4 | VT_ARRAY (Array [lo..hi] of VT_I4)	Raw / Array [lo..hi] of DInt	-

PLC data type	HMI data type	Value range
VT_UI4 | VT_ARRAY (Array [lo..hi] of VT_UI4)	Raw / Array [lo..hi] of UDInt	-
VT_R4 | VT_ARRAY (Array [lo..hi] of VT_R4)	Raw / Array [lo..hi] of Real	-
VT_R8 | VT_ARRAY (Array [lo..hi] of VT_R8)	Raw / Array [lo..hi] of LReal	-

2.8.3.4 Data types for OPC XML DA (RT Professional)

Overview

The following table shows the data types for OPC XML DA with the corresponding HMI data types and value ranges in WinCC.

PLC data type	HMI data type	Value range
Boolean	Bool	0 (FALSE), 1 (TRUE)
Byte	SInt	-128 … +127
UnsignedByte	USInt	0 … 255
Short	Int	-32768 … +32767
UnsignedShort	UInt	0 … 65535
Int	DInt	−2147483648…+2147483647
UnsignedInt	UDInt	0 … 4294967295
Float	Real	±1.17549E-38 to ±-3.40282E +38 and 0.0
Double	LReal	±1.79769313486231E+308 to ±2.22507385850720E-308 and 0.0
String	WString	-
Base64Binary	Raw	-
ArrayOfByte (Array [lo … hi] of byte)	Raw	-
ArrayOfShort (Array [lo … hi] of short)	Raw	-
ArrayOfUnsignedShort (Array [lo … hi] of unsignedShort)	Raw	-
ArrayOfInt (Array [lo … hi] of int)	Raw	-
ArrayOfUnsignedInt (Array [lo … hi] of unsignedInt)	Raw	-
ArrayOfLong (Array [lo … hi] of long)	Raw	-
ArrayOfUnsignedLong (Array [lo … hi] of unsignedLong)	Raw	-
ArrayOfFloat (Array [lo … hi] of float)	Raw	-
ArrayOfDecimal (Array [lo … hi] of decimal)	Raw	-

PLC data type	HMI data type	Value range
ArrayOfDouble (Array [lo … hi] of double)	Raw	-
ArrayOfBoolean (Array [lo … hi] of boolean)	Raw	-
ArrayOfString (Array [lo … hi] of string)	Raw	-
ArrayOfDateTime (Array [lo … hi] of dateTime)	Raw	-

2.8.3.5 Data types for Allen Bradley Ethernet IP (RT Professional)

Overview

The following table shows the data types for Allen Bradley Ethernet IP with the corresponding HMI data types and value ranges in WinCC.

PLC data type	HMI data type	Value range
Bool	Bool	0 (FALSE), 1 (TRUE)
SInt	SInt	-128 … +127
USInt	USInt	0 … 255
Int	Int	-32768..+32767
UInt	UInt	0 … 65535
DInt	DInt	−2147483648 … +2147483647
UDInt	UDInt	0 … 4294967295
Real	Real	±1.17549E-38 to ±-3.40282E +38 and 0.0
String	String	Cannot be used

2.8.3.6 Data types for Modicon Modbus TCP/IP (RT Professional)

Overview

The following table shows the data types for Modicon Modbus TCP/IP with the corresponding HMI data types and value ranges in WinCC.

PLC data type	HMI data type	Value range
Bit	Bool	0 (FALSE), 1 (TRUE)
+/- Int	Int	-32768 … +32767
Int	UInt	0 … 65535
16 bit group	UInt	0 … 65535
+/- Double	DInt	−2147483648 … +2147483647
Double	UDInt	0 … 4294967295

PLC data type	HMI data type	Value range
Float	Real	±1.17549E-38 to ±-3.40282E +38 and 0.0
ASCII	String	Cannot be used

2.8.3.7 Data types for Mitsubishi MC (RT Professional)

Overview

The following table shows the data types for Mitsubishi MC with the corresponding HMI data types and value ranges in WinCC.

PLC data type	HMI data type	Value range
Bool	Bool	0 (FALSE), 1 (TRUE)
4-bit block	USInt	0 … 255
8-bit block	USInt	0 … 255
12-bit block	UInt	0 … 65535
16-bit block	UInt	0 … 65535
Int	Int	-32768 … +32767
Word	UInt	0 … 65535
20-bit block	UDInt	0 … 4294967295
24-bit block	UDInt	0 … 4294967295
28-bit block	UDInt	0 … 4294967295
32-bit block	UDInt	0 … 4294967295
DInt	DInt	−2147483648 … +2147483647
DWord	UDInt	0 … 4294967295
Real	Real	±1.17549E-38 to ±-3.40282E +38 and 0.0
String	String	Cannot be used

2.9 Archive tags

2.9.1 Basic principles for data logging (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Data logging is used to collect, process and log process data from an industrial system.

When you analyze the logged process data, you can extract important business and technical information regarding the operational state of the system.

Application of the data logging

You can use data logging to analyze error statuses and to document the process. By analyzing data logs, you can extract the information necessary to allow you to optimize maintenance cycles, increase product quality and ensure that quality standards are met.

See also

Basics of tags (Page 519)

Logging Tags (Page 618)

Data logging (Page 633)

2.9.2 Data logging (RT Advanced, Basic Panels, Panels, Comfort Panels)

2.9.2.1 Basics (RT Advanced, Basic Panels, Panels, Comfort Panels)

Data logging (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Data is information that is collected during the process and saved in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. You define tags for acquiring and editing the process values in WinCC.

External tags are used in WinCC to collect process values and to access a memory address in the connected automation system. Internal tags are not linked to any process and are only available at the associated HMI device.

Principle

The values of external and internal tags can be saved in data logs. Create a logging tag for every tag and specify the log in which it is saved.

Data logging is controlled via cycles and events. Logging cycles are used to ensure continuous acquisition and storage of the tag values. You can also trigger data logging in response to events, such as the change in value of a tag. Define these settings independently for each logging tag.

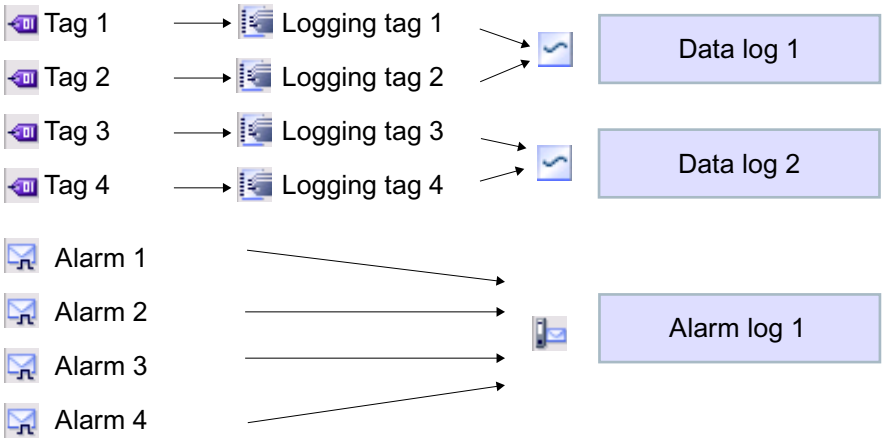

The tag values to be logged are compiled, processed and saved in the data log in Runtime. The HMI device you are using determines where the data log is saved. You can further process the saved data in other programs for analysis purposes, for example.

Logging methods

WinCC supports the following logging methods:

- Circular log
- Segmented circular log
- Circular log which sends a system alarm when it is full
- Circular log which executes system functions when it is full.

Outputting logged data

In runtime, you can output the logged tag values as trends in the process screens.

See also

Logging Tags (Page 618)

2.9.2.2 Working with data logs (RT Advanced, Basic Panels, Panels, Comfort Panels)

Logging Tags (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

In Runtime, you store tag values in logs. You can then analyze the logged data at a later date. You define the following conditions for logging a tag:

- The log tag, through which the values of the connected tag are logged.
- The data log in which the tag is stored.
- The cycle or the event with which the tag is stored.
- The value range within which the tag is stored.

Note

You use data logging mainly to log the values of external tags. You can also log the values of internal tags.

Principle

Several steps come together during data logging:

- Creating and configuring the data log
  When you create a data log, you define the following settings:
  - General settings, such as name, size and storage location
  - Runtime start characteristics
  - Behavior when the log is full
- Configuring the logging of tags
  For every log tag, specify a data log in which the values of the connected tags and other information are logged, such as the time of logging.
  When and how often the values of a log tag are logged are also defined. You have the following options:
  - "On demand":
    The tag values are logged by calling the "LogTag" system function.
  - "On change":
    The tag values are logged as soon as the HMI device detects a change in the value of the tag.
  - "Cyclic":
    The tag values are logged at regular intervals. You can supplement the default cycles in WinCC with your own cycles based on the default cycles. The smallest value that can be set is 1 s. All other values are integer multiples of this value.
    You can also restrict the logging of values within or outside a tolerance band. In this way, the logging of values within a relevant range of values is restricted.

  Note the following points if you want to log a tag on demand:
  - Do not log this type of tag in a segmented circular log in which tags are logged in a continuous cycle or in response to changes.

  Background:
  - If logging takes place on demand only rarely, the log segment will be filled by cyclically-logged values, for example, and the next log segment will be created. If you then attempt to access the tag that was logged on demand, it will not be possible to display the tag since it is the current log segment that is accessed in Runtime. To remedy this, you should create a separate data log for tags that are logged only rarely.
- Further processing logged tag values
  You can analyze the logged tag values directly in your project, such as in a trend view, or in another user program, such as Excel.

See also

Basic principles for data logging (Page 616)

Data logging (Page 616)

Configuring a checksum for a log (Page 627)

Evaluating the checksum of log data (Page 628)

Log response to language switching in runtime (Page 630)

Creating a data log (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Data logs are used to store values from external and internal tags in runtime. When you create a log, you must give it a name and define its size and storage location. In addition you can enter comments for each log.

Requirement

- A project is open.
- The Inspector window is open.

Procedure

To create a data log, proceed as follows:

1. Double-click on the "Historical Data" entry in the project tree.
   The editor for data logs and alarm logs opens.
2. Open the "Data logs" tab and double-click "Add" in the "Name" column of the "Data logs" editor.
   A new data log is created.
3. In the Inspector window select "Properties > Properties > General."
4. Enter a unique name for the log in the "Name" field.
5. Define the number of data records to be logged in each log in the "Number of data records per log" field.
   The size of a log is calculated as follows: The number of items * the length of each tag value to be logged.
   In the Inspector window, the maximum size that the log can reach if the currently selected number of data records is observed is displayed beneath the "Number of data records" input field.
6. In the "Storage location" field you select where the log entries are saved.
7. Depending on the selected "Storage location", you either select the "Path" or the "Name of the data source".
8. If necessary, enter a descriptive text in the "Comment" category to document your configuration.

Alternatively you can configure log properties directly in the "Data log" editor. To view hidden columns, activate the column titles using the shortcut menu.

Result

The data log is created.

You can now configure tags so that their values are stored in this log.

To continue the log configuration, perform the following steps:

- Define the restart characteristics of the log when Runtime is started.
- Define how the log should behave when it is full.
- Configure a function list for the "Overflow" event.

Managing logging behavior when Runtime starts (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

When you configure a log, you define the restart characteristics of the log when Runtime is started. You define whether the logging should start when Runtime starts in the log properties. You can also define whether an existing log will be continued or overwritten.

You define the restart characteristics separately for each log.

Requirement

- You have created a log.
- The "Historical Data" editor is open.
- The Inspector window with the log properties is open.

Procedure

To configure the restart characteristics of a data log, proceed as follows:

1. Select the log for which you want to define the restart characteristics in the "Historical Data" editor.
2. In the Inspector window select "Properties > Properties > Restart behavior".
3. If you want logging to start when Runtime starts, enable the "Enable logging at runtime start" option in the "Logging" area.
   You can also start logging in Runtime using the "StartLogging" system function, for example.
4. Select the restart behavior of the log in the "Log handling at restart" area.
   - The logged values are deleted and logging is started again with the option "Reset log".
   - The option "Append data to existing log" is used to append the values to be logged to the existing log.

Alternatively you can configure the restart characteristics of a log directly in the "Historical Data" editor. To view hidden columns, activate the column titles using the shortcut menu.

Result

Logging will start in runtime according to your settings.

Controlling the Logging in relation to the Fill Level (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

The size of a log is determined by the number of entries. You use the logging method to determine how the log responds when it is full.

Logging methods

The following logging methods are available:

- Circular log
  When the configured log size has been reached, the oldest entries are deleted. When the configured log size has been reached, approximately 20% of the oldest entries are deleted. It is therefore not possible to display all the configured entries. During configuration, select an appropriate size for the circulation log. Alternatively, configure a segmented circular log.
- Segmented circular log
  In a segmented circular log, multiple log segments of the same size are filled in succession. When all logs are completely full, the oldest log is overwritten.
- Log that sends a system alarm when it is full
  When a defined level is reached, such as 90 %, a system alarm is triggered. When the log is 100% full, new tag values are not logged.
- Log with level-dependent triggering of an event.
  When the log is completely full, the "Overflow" event is triggered. Configure a function list for the event that will be carried out when the "Overflow" event occurs. When the configured size of the log is reached, new tag values are not logged.
  The following system functions are available for further processing of full logs:

Requirement

- You have created a log.
- The "Historical Data" editor is open.
- The Inspector window with the log properties is open.

Procedure

1. Select the log for which you want to define the logging method in the "Historical Data" editor.
2. Select "Properties > Properties > Logging method" in the Inspector window and select the required logging method.
3. If you have selected the "Segmented circular log" type, enter the number of log segments. The system creates an additional log segment for the main log. This results in the total number of log files created from the number of configured segments as well as the automatically created log.
   If you selected a log with the "Display system alarm on" setting, specify the level as a percentage at which a system alarm is to be triggered.
   If you selected the "Trigger event" setting, configure the function list in the "Events" group.

Alternatively you can configure the logging method directly in the "Historical Data" editor table. To view hidden columns, activate the column titles using the shortcut menu.

The "Overflow" event is not available in the editor table. You must therefore configure the function list in the Inspector window.

Result

The selected log responds according to the settings in Runtime.

See also

System functions for logs (Page 626)

Logging process values (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You can save the process values of a tag in a data log in Runtime. You define the following conditions for logging a tag:

- The log tag, through which the values of the connected tag are logged.
- In which log the values are stored
- The conditions under which the values are stored
- If only process values for a certain range of values are stored

To log the tag values, assign a logging tag to an HMI tag. The logging tag is stored in the data log and logs the values of the connected HMI tag. You can configure logging tags directly in the "HMI tags" editor. The " HMI tags" editor contains the "Logging tags" editing table.

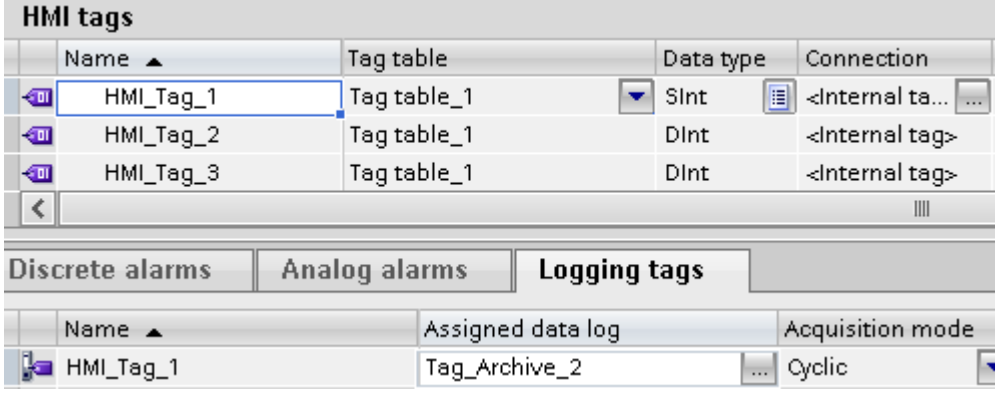

If the view of the "Logging tags" table is minimized, click the arrow button below the tag table.

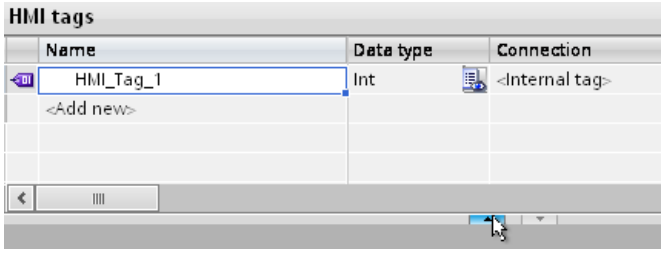

Figure 2-1 The "Logging tags" table is displayed.

Requirement

- The data log has been created.
- The tag for which you wish to configure the logging must already exist.
- The "Tags" editor is open.
- The "Logging tags" table is displayed.
- The Inspector window with the tag properties is open.

Procedure

To log process values in a tag, proceed as follows:

1. Select a tag in the tag table.
2. Double-click "Add" in the "Name" field in the "Logging tags" table.
   A new logging tag is created; it is given the same name as the associated HMI tag.
3. Select the data log in which the values of the tags are to be logged under "Properties > Properties > General" in the Inspector window.
4. Select "Properties > Properties > Logging type" in the Inspector window and select the log type for logging.
   - "Cyclic": The tag values are logged in accordance with the set logging cycle.
   - "On change": The tag values are logged, as soon as the operator device detects a change in value.
   - "On demand": The tag values are logged by calling the "LogTag" system function.
5. If you want to log tag values cyclically, select a cycle time in the "Logging cycle" area. Alternatively, you can define your own cycle using the object list. The smallest value that can be set is 1 s. All other values are integer multiples of this value.

6. If you only want to log tag values outside or inside a defined value range, select "Properties > Properties > Deadband for logging" in the Inspector window. Define the values for the high and low limits.
   If you want to configure a dynamic limit, select "HMI tag" using the selection button. In the second field, select the tag that contains the limit.
   If you want to configure a fixed limit, select "Constant." Enter the limit value in the second field.
   If you want to leave a limit undefined, select "None".
7. Under "Scope", specify whether tag values are to be logged only if they are within the defined limits or only if they are outside the defined limits.

Alternatively, you can configure the logging of a tag directly in the "Logging tags" editor table. To view hidden columns, activate the column titles using the shortcut menu.

You can also fully configure a logging tag in the "Historical Data" editor.

Result

The process values of the configured tag are logged in Runtime according to the selected settings.

Note

To have the tag values actually logged in runtime, you must ensure that the data log is started. Logging can be started automatically at start up or by means of a system function. To automatically start the log, use the setting in the log Properties window.

Triggering a system function when log is full (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You can select a logging method that executes a function list as soon as the log is full.

Application example

You can use system functions, for example, to swap the data from a full log file out to another log before the full log is overwritten. You can then continue to process the transferred data in another program. Assign the "CopyLog" system function to the "Overflow" event accordingly.

Requirement

- A log with the "Trigger event" logging method is created.
- The "Data log" editor is open.
- The Inspector window with the data log properties is open.

Procedure

To configure a system function for the "Overflow" event, proceed as follows:

1. Select the required log in the table in the "Data Log" editor.
2. In the Inspector window select "Properties > Events > Overflow."
   The function list will open.
3. Double-click "Add function" and select the desired system function.
4. Configure the required parameters of the selected system function.

Result

During runtime, the function list will be executed as soon as the log is full.

System functions for logs (RT Advanced, Basic Panels, Panels, Comfort Panels)

System functions

The following system functions are available for logging:

Function name	How it works
ArchiveLogFile	This function moves or copies a log to another storage location for long-term archiving. Use the system function if you want to move the Audit Trail, for example, from a local storage medium to the server. With Audit Trails, always use the "Move log (hmiMove)" mode, otherwise you will be violating the FDA guideline by duplicating the data stored.
LogTag	Saves the value of the given tags in the given data log. Use the system function to log a process values at a specific time.
StartLogging	Starts the logging process in the specified log. You can interrupt logging in Runtime by calling the "StopLogging" system function.
StopLogging	Stops the logging process in the specified log. To resume logging in Runtime, select the "StartLogging" system function.
ClearLog	Deletes all entries in the specified log.
StartNextLog	Stops the logging process in the specified log. Logging is continued in the log segment that has been configured for the given log.
CloseAllLogs	Closes all logs. The connection between WinCC and the log files or log database is terminated. You can use this system function, for example, to enable hot-swapping of the storage medium on the HMI device without exiting the Runtime software.
OpenAllLogs	Opens all logs to resume logging. The connection between WinCC and the log files or log database is restored.
CopyLog	Copies the contents of the specified log to another log.

Configuring a checksum for a log (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

In a regulated project, you have the option of assigning a checksum to the log data of an data log or alarm log. This checksum can be used during plant operation to determine if the data of this log has subsequently changed.

Note

Device dependency

The "Checksum" option is only available for display and HMI devices which support "Configuration conforms to GMP".

Requirements

- GMP compliant configuration is enabled.
- A data log or alarm log has been created.

Procedure

Proceed as follows to configure a data log or alarm log for the use of a checksum:

1. Open the data log or alarm log in the corresponding log editor.
2. In the "Storage location" box, select "File - CSV (ASCII)" or "File - TXT (Unicode)".

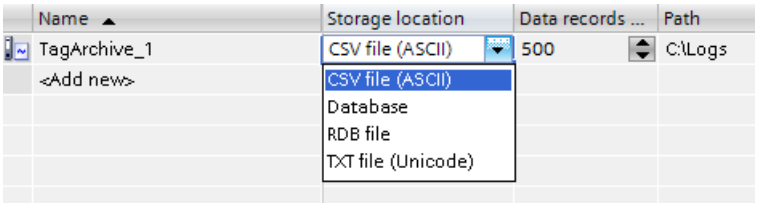

3. Under "Properties > Properties > Logging method" in the Inspector window, select the option "Display system event at" or "Trigger event".

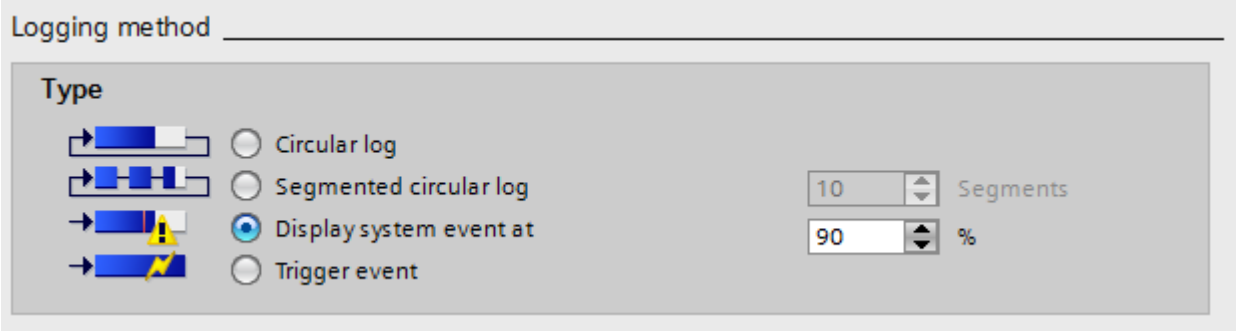

4. In the editor table, activate the option "Enable checksum".

5. In the editor table, activate the option "Enable logging at runtime start".
   Columns that are not displayed are activated with the shortcut menu of the column title.

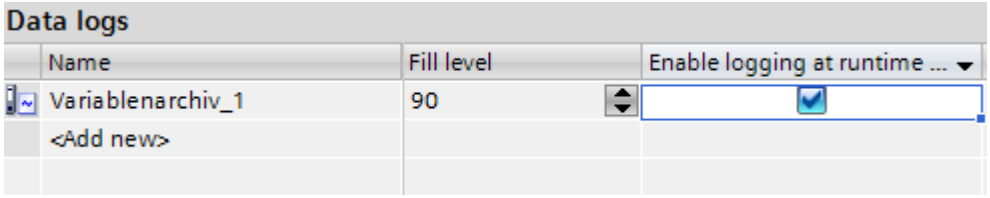

6. Save the project.

Result

The log data of the log is assigned a checksum in runtime.

See also

Logging Tags (Page 618)

Evaluating the checksum of log data (Page 628)

Evaluating the checksum of log data (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

If you have configured a data log or alarm log with generation of a checksum, you can check if the log data has subsequently changed.

Note

Device dependency

The "Checksum" option is only available for display and HMI devices which support "Configuration conforms to GMP".

The DOS program "HmiCheckLogIntegrity" is available for checking the integrity of the log data.

The "HmiCheckLogIntegrity" program supports verification of the following files:

- Log files of alarm logs, data logs, and Audit in CSV format
- Log files of alarm logs, data logs, and Audit in TXT format
- Recipe data records in CSV format
- Recipe data records in TXT format

You can find the "HmiCheckLogIntegrity.exe" program in the installation directory of WinCC under the folder "WinCC Runtime Advanced", for example <C:\Program Files\Siemens \Automation WinCC Runtime Advanced>.

Note

Audit Trail and Log with Checksum

Before you update WinCC with a Service Pack or a new version, exit and save the Audit Trail or the logs using checksum. After WinCC is updated, the audit trail or logs will be continued with new files using checksum.

Make sure that the logs are started at a defined state with the new version.

Procedure

1. Copy the file to be checked from the HMI device to your configuration computer.
2. Open a command line prompt with "Start > Programs > Accessories > Command Prompt".
3. Enter the path to "HmiCheckLogIntegrity.exe" followed by a space in the command line prompt. After the space, enter the storage location of the file to be checked within quotation marks.
4. Press <Enter>.
5. The check is performed.
   When the checked data are consistent, the "Consistency check succeeded" message appears.

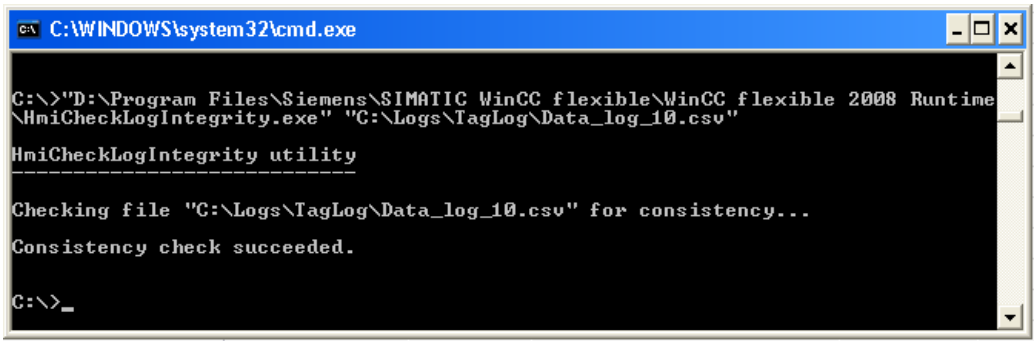

   When the checked data are inconsistent, the "Consistency check failed" message appears. Information about the first inconsistent line in the file is also displayed.

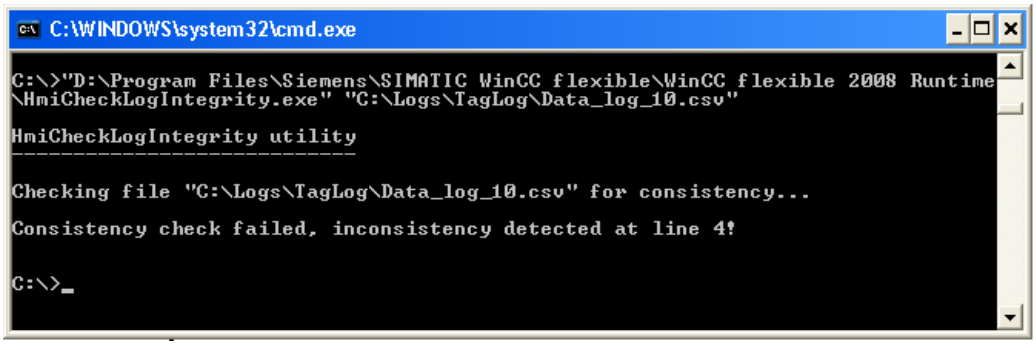

Note

If there are spaces in the path to the "HmiCheckLogIntegrity.exe" program, you need to specify the path in quotation marks.

You can also check the integrity of the log data with the AuditViewer.

See also

Logging Tags (Page 618)

Configuring a checksum for a log (Page 627)

Log response to language switching in runtime (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

In the Runtime settings of your HMI device, select the language to be used for writing to logs in runtime.

Requirement

- The languages used in your project are activated in the "Project languages" editor, for example, "German (Germany)" and "English (USA)" .

Procedure

To determine the startup language, follow these steps:

1. Select "Runtime settings > Language and fonts" in the project navigation.
2. Activate the runtime language, for example, "German (Germany)" and "English (USA)".
3. Set the "Language switch order". Use 0 to determine the startup language, for example:
   - German 0.
   - English 1
     With "0", German is specified as the "Startup language".
4. Select "Runtime settings > General" in the project navigation.
5. Select the "Logs > Logging language > Startup language".

Result

The project starts after it is transferred. German is specified as the "Startup language" with "Language switch order". The logs are now written in German. During runtime, the operator switches the runtime language to English. The logs will still to be written in German.

The operator closes runtime. Due to the previously performed language switch, the next time the system starts the "startup language" is English. Since English is the startup language, the logs are now written in English.

The logging language remains English even when the language is switched again in runtime until runtime is closed once again.

If you use another option instead of "Startup language" to select the language, the logs are always written in the same language. This is true regardless of the language selected by the operator in runtime.

See also

Logging Tags (Page 618)

Logging tags in the "Historical Data" editor (RT Advanced, Basic Panels, Panels, Comfort Panels)

Configuring logging tags (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You can also create and edit logging tags in the "Historical Data" editor in WinCC. You also directly edit the properties of logging tags in the "Historical Data" editor.

Note

If you delete, move or copy in the "Historical Data" editor, the changes also take effect in the tag table.

Requirement

- The "Data logs" tab is open in the "Historical Data" editor.
- A data log has been created.

Procedure

Proceed as follows to configure a logging tag in the "Historical Data" editor:

1. Select an existing data log in the "Data logs" table of the editor.
   Alternatively, double-click on "Add..." in the "Name" column to create a new data log.

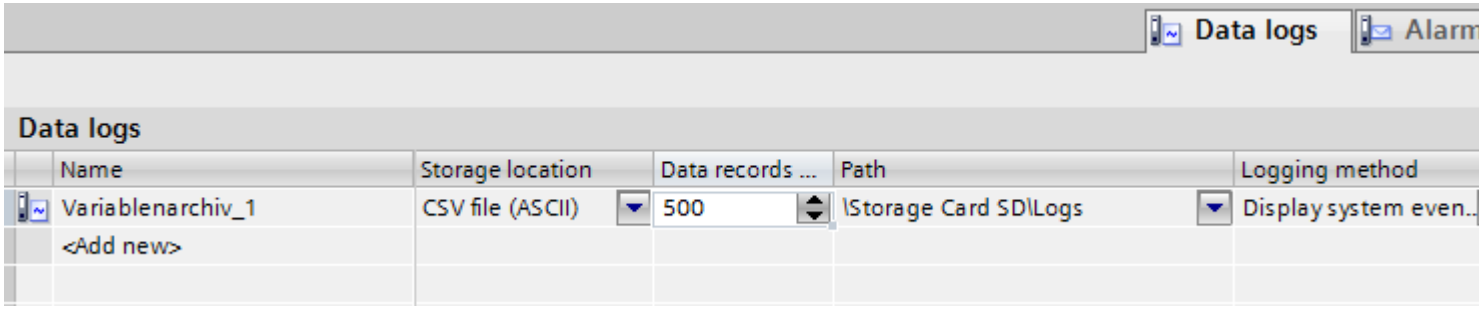

2. Double-click "Add ..." in the "Name" column of the editor "Logging tags" table.

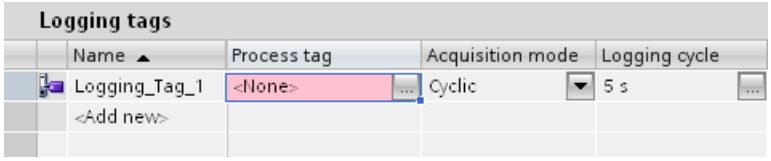

3. Enter a unique name for the logging tag in the "Name" field.
4. In the "Process tag" field, click on the selection button and select the process tag for logging in the object list.

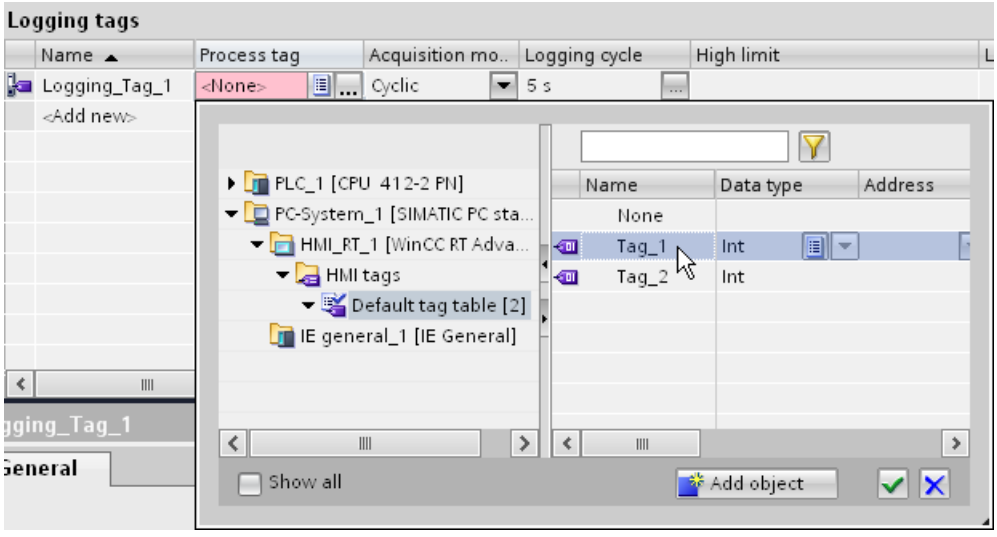

5. Select the desired trigger mode in the "Log type" field:
   - "Cyclic": The tag values are logged in accordance with the set logging cycle.
   - "On change": The tag values are logged, as soon as the operator device detects a change in value.
   - "On demand": The tag values are logged by calling the "LogTag" system function.

6. If you want to log tag values cyclically, select the desired cycle time in the "Logging cycle" area. Alternatively, you can define your own cycle using the object list. The smallest value that can be set is 1 s. All other values are integer multiples of this value.
7. Configure additional parameters for logging in the table of the editor or in the Inspector window.

Result

The configured logging tag is created in the "Historical Data" editor and is also displayed in the tag table.

2.9.3 Data logging (RT Professional)

2.9.3.1 Data logging (RT Professional)

Introduction

The logging system is used for data logging in Runtime. The logging system processes the process values cached in the Runtime database and writes them to the logging database.

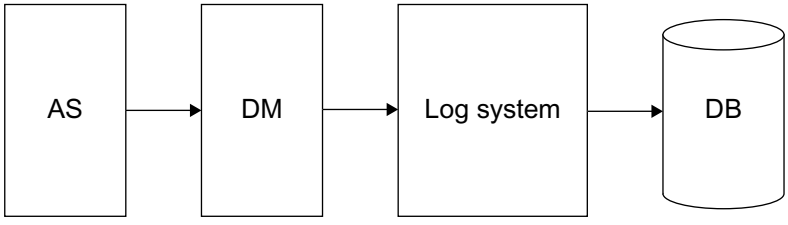

The following WinCC subsystems are involved in the data logging:

- Automation system (AS): Saves the process values that are sent to WinCC via communication drivers.
- Data manager (DM): Processes the process values and returns them to the logging system via process tags.
- Log system: Processes the acquired process values (by calculating the average, for example). The method of processing depends on the configuration of the log.
- Runtime database (DB): Saves the process values that are to be logged.

Term definitions

The question of whether and when process values are acquired and logged is dependent on a variety of parameters. The parameters to be configured depend on the applied logging method:

- Acquisition cycle: Determines when the value of a process tag is read out in the automation system. You can, for example, configure an acquisition cycle for cyclic data logging.
- Logging cycle: Determines when the processed process value is saved in the log database. You can, for example, configure a logging cycle for cyclic data logging.
- Start/stop tag: Starts data logging when the assigned binary tag has the value "1". Logging stops as soon as the start condition is no longer in effect. You can configure a start/stop tag for cyclic-selective data logging.
- Logging on demand: Process values are logged only upon demand. The request is controlled by a binary tag. Use the "Upon demand" trigger mode for the configuration.
- Data logging on change: Process values are logged only when they have changed. Use the "On change" trigger mode for the configuration.

See also

Basic principles for data logging (Page 616)

2.9.3.2 Basics (RT Professional)

Basic principles for data logging (RT Professional)

Introduction

Data logging is used to collect, process and log process data from an industrial system. When you analyze the logged process data, you can extract important business and technical information regarding the operational state of the system.

How it works

The process values to be logged are compiled, processed and saved in the log database in Runtime. Current or previously logged process values can be output in Runtime as a table or trend. In addition, it is possible to print out logged process values as a report.

Configuration

You configure the logging of process values in the "Historical Data" editor. You create data logs and compressed logs. You define the acquisition cycles, logging cycles, and select the process values to be logged.

You configure trend views and table views for displaying process data in Runtime in the "Screens" editor. These views allow you to output the process data in the form of trends and tables.

You configure the output of logged process data in report format in the "Reports" editor. You output the process data in the report as a table or a trend.

Application

You can use data logging for the following tasks:

- Early detection of danger and fault conditions
- Increase of productivity
- Increase of product quality
- Optimization of maintenance cycles
- Documentation of process value trends

See also

Telegram Tags (Page 636)

Structure of a Telegram with Raw Data Tags (Page 636)

Logging tags (RT Professional)

Introduction

The values of external and internal tags are stored in logging tags in a data log.

Logging tags

There must be one archive tag created in the data log for every tag whose values are to be logged. There are suitable types of logging tags for the various tag data types. We distinguish between the following types of logging tag:

- An analog archive tag stores numerical process values, such as the fill level of a tank.
- A binary archive tag stores binary process values, such as whether a motor was switched on or off.

Archived process values can be compressed. This compression is achieved through the application of mathematical functions, for example, averaging. Compressed process values of this type are stored in compressed logging tags in a compressed log.

Telegram Tags (RT Professional)

Introduction

Telegram tags are needed to acquire quickly changing process values or to combine several measuring points from one plant.

Note

Telegram tags are of the "raw data type" in WinCC and are therefore also referred to a "raw data tags".

Principle

In the automation system the process values are written to a binary file, and are sent as a telegram to WinCC where they are stored in a raw data tag.

Archiving of Telegrams

If you want the acquired process values belonging to a raw data tag to be archived, you need to configure a process-controlled tag in the data log. To enable the archive system to process the telegram in the process-controlled tag, select a format DLL. The format DLL is supplied with the automation system that you are using and dismantles the telegram, for example, to ascertain the process values. The process values are then written to the archive database.

A format DLL for the SIMATIC S7 is included as standard in the scope of delivery of WinCC.

If you want to log a raw data tag, you have to activate the "Archive data link" option in the properties of the raw data tag. In the Inspector window, activate the option under "Properties > Properties > Raw data". When the option "Archive data link" is activated, you can select and connect the raw data tag at the logging tag under "Properties > General" in the "Process tag" field.

See also

Basic principles for data logging (Page 634)

Structure of a Telegram with Raw Data Tags (Page 636)

Structure of a Telegram with Raw Data Tags (RT Professional)

Introduction

A telegram for the transfer of raw data tags consists of two parts: a header and a body.

Telegram Header	Telegram Footer with Block Status Word, Time Information, Measurement Area, Measured Values, ...

Telegram header

The header contains general data, i.e. length of the telegram. The high byte of data word 0 is not used by the system and might thus be assigned by the user as required.

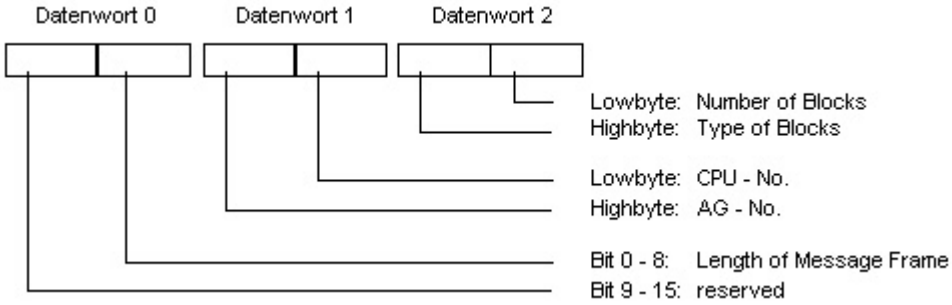

Telegram body

In the block status word, the format of the measurement values and the format of the measurement areas, among other things, are defined. Bit 10 is reserved and will be used in future versions for switching daylight saving and standard times (daylight saving time = 1).

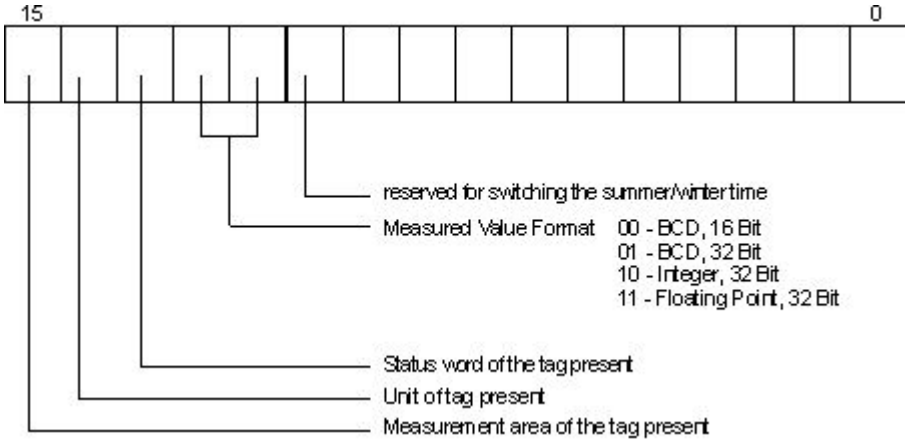

Time and date are structured in accordance with the definition of the sequential time indication.

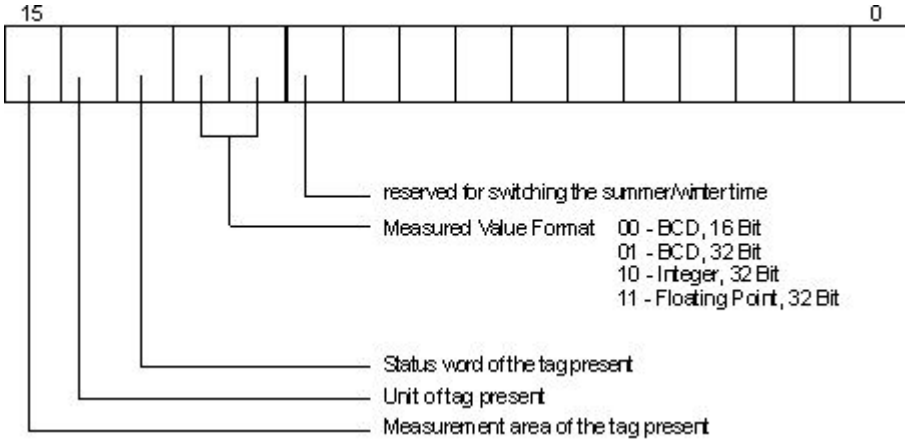

If a measurement range is also to be transferred, 8 data words are needed in which the upper and lower limits of the tag and the archive are specified.

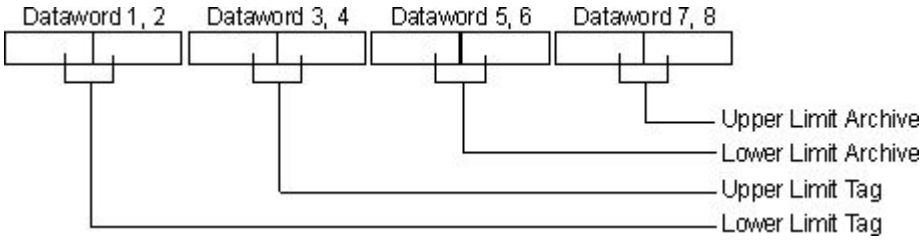

The following section contains a number of sample telegram types.

Type 1

Measured value of a process tag + date and time

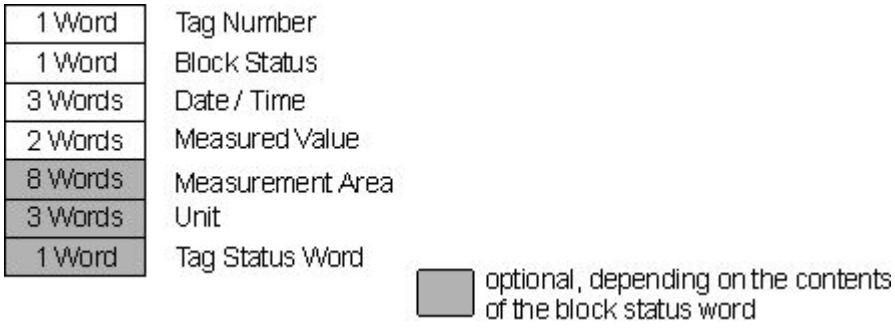

Type 2

n measured values of a process tag + date and time of each measured value

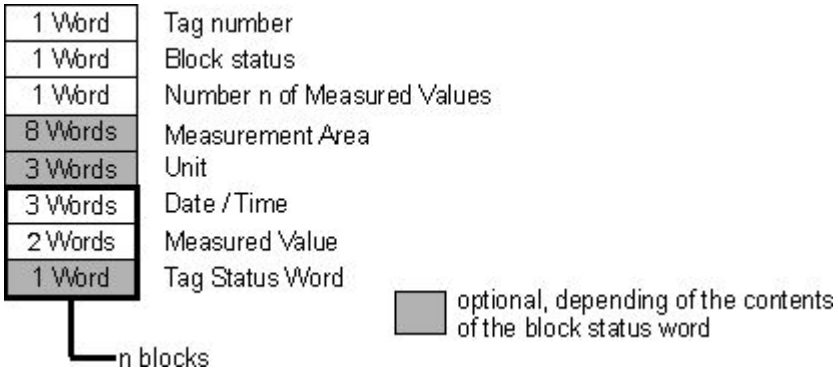

Type 3

n measured values of a process tag with date and time, and sampling cycle

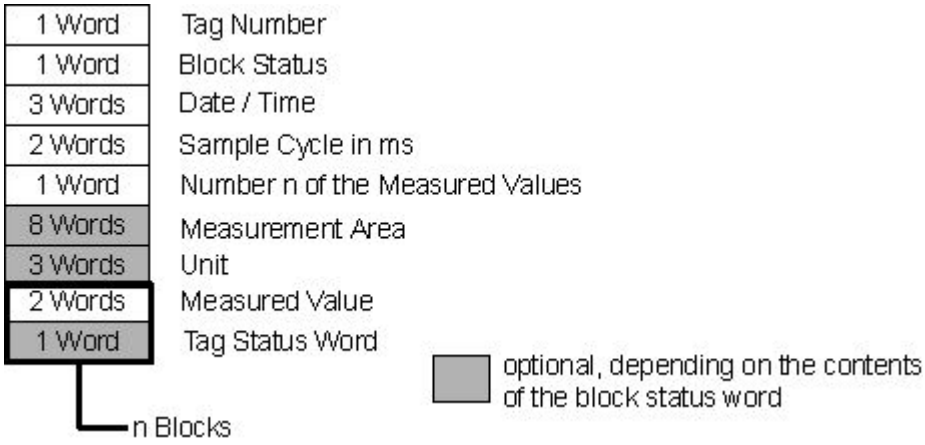

Type 4

n measured values of various process tags with time and date

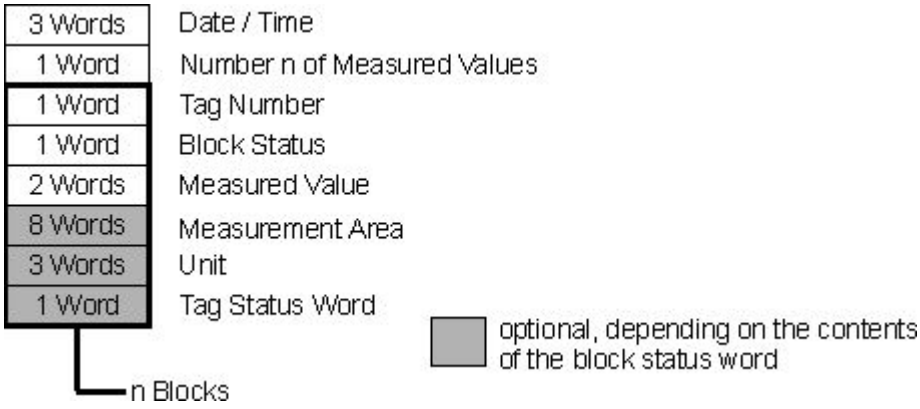

Type 5

n measured values of various process tags without time and date

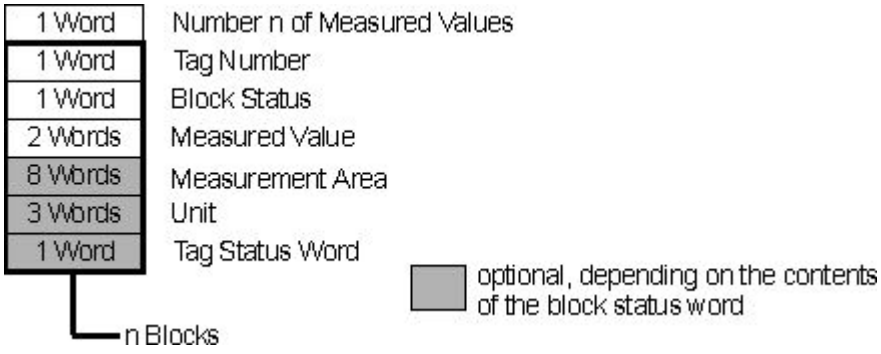

See also

Basic principles for data logging (Page 634)

Telegram Tags (Page 636)

Types of Logging (RT Professional)

Trigger mode (RT Professional)

Introduction

You can use different logging methods to log process values. This will allow you to monitor an individual process value at certain times, for example. You make the monitoring dependent on certain events. You can log quickly-changing process values without increasing the system load as a result. You can also compress existing logged process values to reduce the volume of data.

Logging methods

The logging method is determined by the selected trigger mode. The following logging methods are available in Runtime:

- Cyclic data logging: Continuous data logging to monitor a process value, for example.
- Cyclic-selective data logging: Event-controlled, continuous data logging to monitor a process value within a specific time period, for example.
- Data logging on demand: Process values are logged only on demand from a trigger tag.
- Data logging on change: Process values are logged only when they have changed.
- Compressed log: Compression of individual logging tags or of entire data logs, such as the hourly averaging of process values that were logged every minute.

Cycles and Events (RT Professional)

Introduction

Data logging is controlled via cycles and events. The acquisition and logging cycles allow you to continuously acquire and store process values. You can also trigger or end data logging by demand or by events. Cycles and events may be combined. For example, a process value can be acquired regularly, but the logging will only be triggered by a binary event.

Acquisition cycle

The acquisition cycle determines the interval at which the process value of a process tag is read. The shortest possible value is 500 ms. All other values are integer multiples of this value. The start time of an acquisition cycle is determined by the start time of WinCC Runtime.

Note

A short acquisition cycle may lead to a high system load. Make sure that the acquisition cycle you select is only as short as you actually need.

Logging cycle

The logging cycle determines when a process value is saved in the logging database. The logging cycle is always an integer multiple of the acquisition cycle. The start time of the logging cycle depends on either the start time of WinCC Runtime or the selected starting point.

When you set the start time, the values will be logged after a delay, which will share out the logging load. Examples:

- Process values are logged in three cycles: every minute, every two minutes, and every three minutes. This causes a high logging load every six minutes. You should therefore assign a different start time to each of the three cycles.

Logging cycle	Selected start time
1 min	At the 15th second
2 min	At the 30th second
3 min	At the 45th second

- Many process values must be logged every ten seconds. To distribute the logging load, configure two 10-second cycles with different starting points. Logging is performed at both of the selected starting points.
  You can configure the starting point for a cycle in the cycle properties in the "Cycles" editor.

The logging load will be distributed by the different starting points.

Note

If the same cycle is used for acquisition and logging, it does not necessarily mean that acquisition and logging are started simultaneously.

There might be a system delay of up to the length of one acquisition cycle between the time of acquisition and the time of logging.

Processing method

All process values read from the process tags during the time period between acquisition and logging will be processed by the logging function. There are different processing methods available for binary and analog values.

In a data log, use one of the following processing methods for analog values:

- Current: Saves the last acquired process value.
- Total: Saves the sum of all acquired process values.
- Maximum: Saves the highest of all acquired process values.
- Minimum: Saves the lowest of all acquired process values.
- Average value: Saves the mean of all acquired process values.
- Calculation: The value to be logged is calculated from the most recently acquired process value. The calculation is made using a function created in the "Scripts" editor.

In a data log, use one of the following processing methods for binary values:

- 0 to 1: The logging takes place whenever the value changes from 0 to 1.
- 1 to 0: The logging takes place whenever the value changes from 1 to 0.
- On each change: The logging takes place whenever the value changes from 0 to 1 or from 1 to 0.
- Always: The logging depends on the selected cycle and does not depend on the change in value.

Events

Start events and end data logging. Conditions that trigger an event can be linked to a tag. Events are differentiated as follows in WinCC:

- Binary action: Responds to changes in a Boolean tag.
  Example: Switching on a motor starts the data logging.
- Deadband criterion: Defines an absolute or percentage value by which the newly acquired tag value has to deviate from the old tag value so that the newly acquired tag value will be logged. The value change can be absolute or relative.
  Example: Logging is triggered in response to temperature fluctuations greater than 2 %.
- Time-controlled event: Responds at a fixed time or when a defined period of time from the start of data logging has elapsed.
  Example: A log is output after every shift change.

Cyclic data logging (RT Professional)

Introduction

Cyclic data logging begins when Runtime starts. The process values are acquired at fixed cycles and stored in the log database. You can also specify limits. The process value is then only logged if it is within the defined limits. A comparison of the process value with the limit values takes place after acquisition of the process value.

Cyclic data logging ends when Runtime closes.

How it works

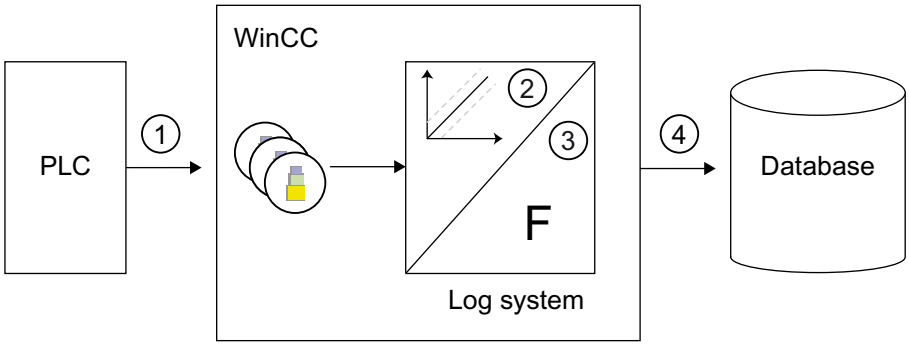

The external tags in WinCC correspond to a certain process value in the memory of one of the connected PLCs. The acquisition cycle (1) determines when the process value is read from the memory of the connected PLC.

The Runtime component of the logging system processes the process value:

- Whether the process value is logged at all will depend on the way you have configured the system. For example, the process value may have to change by a certain amount or percentage (2).
- The logging function (3) defines how the acquired process values will be processed (by averaging, for example).

The logging cycle (4) determines when the processed process value is written to the log database.

Cyclic-selective data logging (RT Professional)

Introduction

The start of cyclic-selective data logging is controlled by a "Bool" type tag. Logging is started as soon as the tag takes the value "1" and the logging cycle has been triggered in Runtime (by default when you start Runtime). The process values are subsequently acquired at fixed cycles and stored in the log database. You can also specify limits. The process value is then only logged if it is within the defined limits. A comparison of the process value with the limit values takes place after acquisition of the process value.

Cyclic data logging is ended by the following events:

- When the tag for starting the logging has the value "0".
- When you exit Runtime.

When Runtime stops, the most recently acquired process value is also logged.

How it works

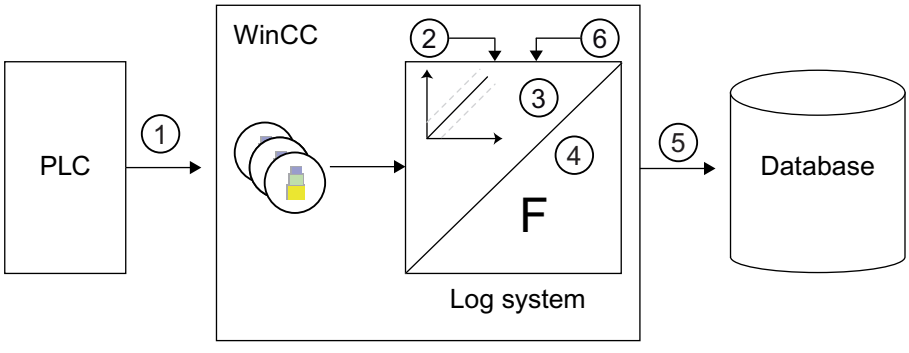

The external tags in WinCC correspond to a certain process value in the memory of one of the connected PLCs. Data logging starts as soon as the tag for the start of logging has the value "1" in Runtime (2). The acquisition cycle (1) determines when the process value is read from the memory of the connected PLC.

The Runtime component of the logging system processes the process value:

- Whether the process value is logged at all will depend on the way you have configured the system. For example, the process value may have to change by a certain amount or percentage (3).
- The logging function (4) defines how the acquired process values will be processed (by averaging, for example).

When logging stops (6), the logging cycle (5) determines when the processed process value is written to the log database.

Data logging on demand (RT Professional)

Introduction

Data logging on demand saves the current process value to the logging database only on demand in Runtime. The demand is controlled by a trigger tag. The trigger tag is of the data type "Bool". Logging takes place every time the status of the trigger tags changes. You can also specify limits. The process value is then only logged if it is within the defined limits. A comparison of the process value with the limit values takes place after acquisition of the process value.

Data logging on demand ends when Runtime closes.

How it works

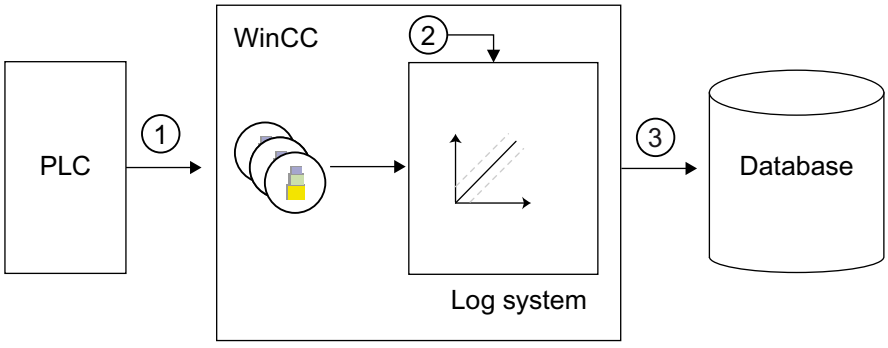

The external tags in WinCC correspond to a certain process value in the memory of one of the connected PLCs. The process value is read from the PLC (1) when a request from the trigger tag (2) is received. The Runtime component of the logging system processes the process value. The current process value is then written to the logging database (3).

The process value is read anew from the connected PLC (1) with each new request from the trigger tag (2), it is processed and logged (3).

Data logging on change (RT Professional)

Introduction

Data logging on change saves the current process value to the log database in Runtime when the process value changes. You can also specify limits. The process value is then only logged if it is within the defined limits. A comparison of the process value with the limit values takes place after acquisition of the process value.

Data logging on change ends when Runtime closes.

How it works

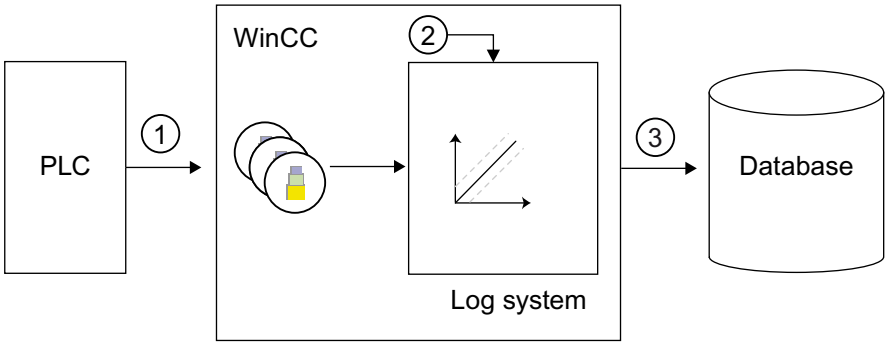

The external tags in WinCC correspond to a certain process value in the memory of one of the connected PLCs. The process value is read cyclically from the memory of the connected PLC (1) and monitored for change. The process value is logged when a change (2) occurs of the external tag.

The Runtime component of the logging system processes the process value.

The current process value is then written to the logging database (3).

Compressed log (RT Professional)

Introduction

To reduce the volume of data in the logging database, the logging tags for a specified period can be compressed. To do this, create a compressed log that will store each logged tag in a compressed log tag. The logging tags are retained, but they can also be copied, moved or deleted. The compressed log is stored in the log database in the same way as the data log.

How it works

This compression is achieved through the application of mathematical functions. To achieve this, one of the following functions is applied to the logged process values from a specified period:

- Maximum value: The highest process value is saved in the compressed log tag.
- Minimum value: The lowest process value is saved in the compressed log tag.

- Total: The total of all the process values is saved in the compressed log tag.
- Average value: The average process value is saved in the compressed log tag.
- Average weighted value: The average weighted process value is saved in the compressed log tag. The time period in which a tag value has not changed is included in the calculation. For tags which are acquired "on change", this function returns more meaningful results than the "Average value" function.

What happens to the old, logged process values after compression depends on which method of compression is used:

- Calculate: The process values of the logging tags of the specified period are read out and compressed. The process values of the logging tags are retained.
- Calculate and copy: The process values of the logging tags of the specified period are read out, compressed and copied to the compressed log.
- Calculate and delete: The process values of the logging tags of the specified period are read out, compressed and then deleted.
- Calculate, copy and delete: The process values of the logging tags of the specified period are read out, compressed, copied to the compressed log and then deleted.

Example

The following example illustrates the way that the compressed log works:

A process value is archived once every minute and returns 60 values in one hour. The compression takes place over the course of one hour by averaging, for example. Accordingly, every hour, the average of the 60 values is calculated and is stored in the compressed log tag. What happens to the 60 values depends on the compression method described above.

See also

Creating a Compressed Log (Page 653)

Creating a Compressed Log Tag (Page 658)

Storing Process Values (RT Professional)

Introduction

Process values can be stored either in the log database on the hard disk or in the main memory of the Runtime logging system.

Storing in Archive Database

The process values to be logged are stored in two separate cyclic archives in the log database.

- Fast data log (A)
- Slow data log (B)

Each circular log consists of a configurable number of segments. You define a size in MB and a period (such as one day) for the segments.

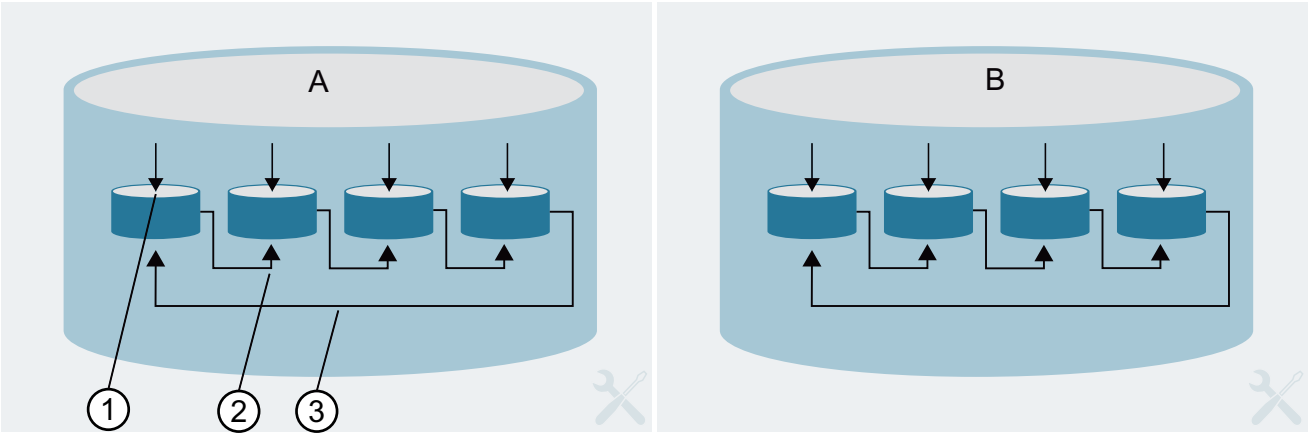

The process values are written continuously to the first segment (1). When the configured size of the segment is reached or the time span is exceeded, the system switches to the next segment (2). When all the segments are full, the process data in the first segment is overwritten (3). In order that process data is not destroyed by the overwritten process, it can be swapped (exported).

The "Fast data log" stores the process values whose acquisition cycle is less than or equal to one minute. These process values are initially saved and compressed in a binary file. When the binary file has reached a specific size, it is stored in the cyclic archive.

The "Slow data log" stores the process values whose acquisition cycle is longer than one minute. The "Slow data log" also stores the compressed logs. The data is written immediately to the circular log and is not compressed.

Note

When starting Runtime, the system tests whether the configured size of a data buffer has been calculated to a sufficient size. If the configured size is too small, the system automatically adapts to the minimum size.

Saving in Main Memory

In contrast to storage in the log database, process values archived in the main memory are only available while Runtime is active. Storing in the main memory has the advantage, however, that the values can be written and read very quickly. The process values stored in the main memory cannot be swapped out.

Note

Compressed logs cannot be saved in the main memory.

Optimized configuration

Use the available log types and the different storage methods to match logging to the dynamics of the changing process values.

Select the logging type depending on the frequency of the value changes of a tag and the necessity for logging.

Combing the logging type with the selection of the matching circular log. Use shorter logging types, for example, only for tags whose values change quickly.

Swapping Out Process Values (RT Professional)

Introduction

You can swap out process values from the log database as a backup. All process values contained in a log segment are swapped out. A log segment is always swapped out when it is full and a new segment is started. A log segment is also swapped out when the time set for a segment change is reached and a new segment is started.

Principle

You configure the swapping out of process values in the log settings. In the settings, you configure the size of the log and the log segmentsn as well as the time for the segment change. You should also state whether a backup is made in the settings.

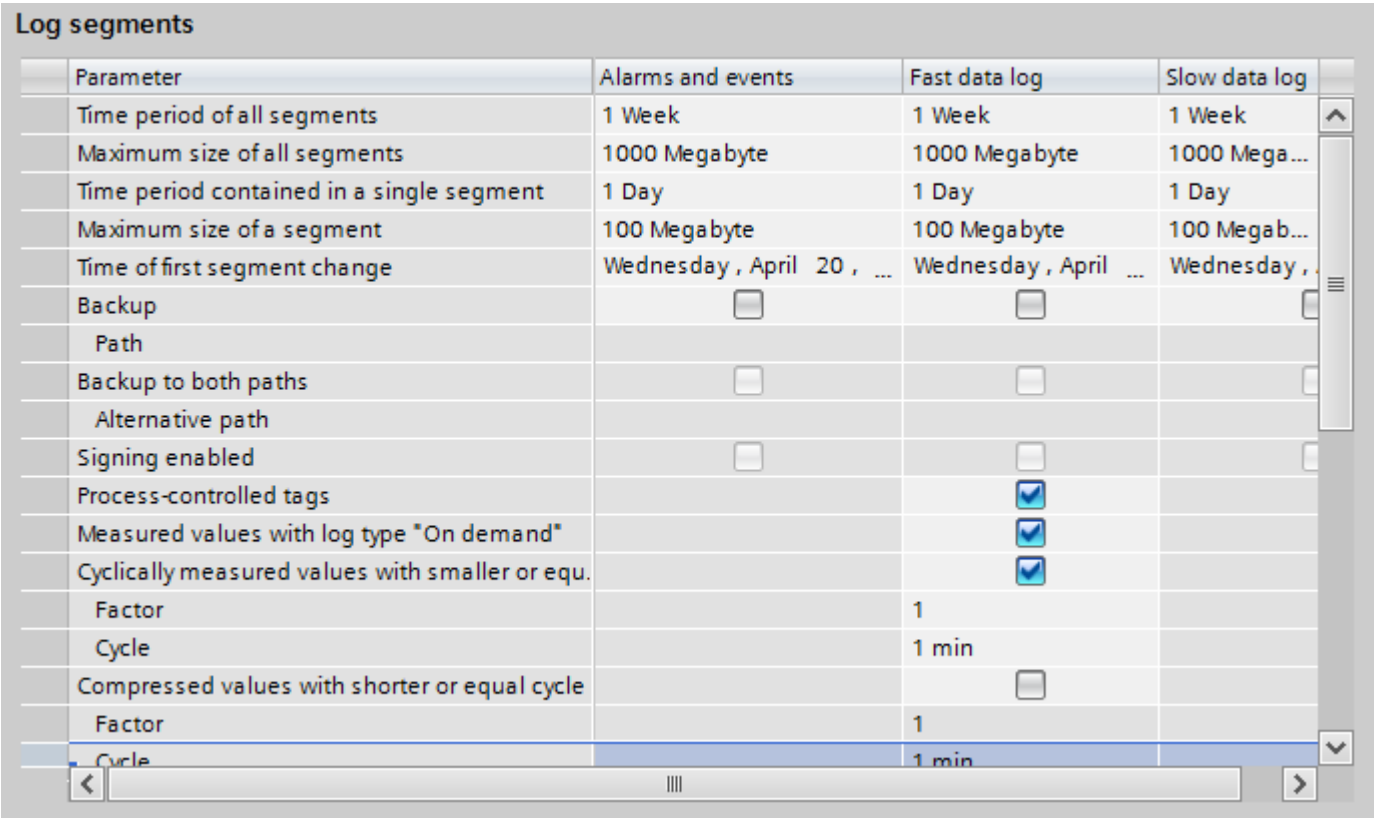

Log segments

Parameter	Alarms and events	Fast data log	Slow data log
Time period of all segments	1 Week	1 Week	1 Week
Maximum size of all segments	1000 Megabyte	1000 Megabyte	1000 Mega...
Time period contained in a single segment	1 Day	1 Day	1 Day
Maximum size of a segment	100 Megabyte	100 Megabyte	100 Megab...
Time of first segment change	Wednesday , April 20 , ...	Wednesday , April ...	Wednesday ,
Backup	☐	☐	☐
Path			
Backup to both paths	☐	☐	☐
Alternative path			
Signing enabled	☐	☐	☐
Process-controlled tags		☑	
Measured values with log type "On demand"		☑	
Cyclically measured values with smaller or equ.		☑	
Factor		1	
Cycle		1 min	
Compressed values with shorter or equal cycle		☐	
Factor		1	
Cycle		1 min	

In the lines of the backup configuration you define where the backup is created and whether the backup file is signed.

Note

You can change a displayed process value in Runtime using the table view.

If the process value has already been swapped out of the data log, the modified value will not be transferred to the swapped-out log. The change only occurs in the local log segment.

If the log segment has not yet been swapped, the changed value is accepted permanently.

See also

Logging Settings (Page 660)

Significance of Archive Value Flags (RT Professional)

The data logging sets flags for every value that is written to the log. The flags indicate the state of the tags.

The flags are represented by a 2-word value. This value is decimal-coded and is shown in the third column of the database log. The flags must be converted to hexadecimal format for analysis purposes.

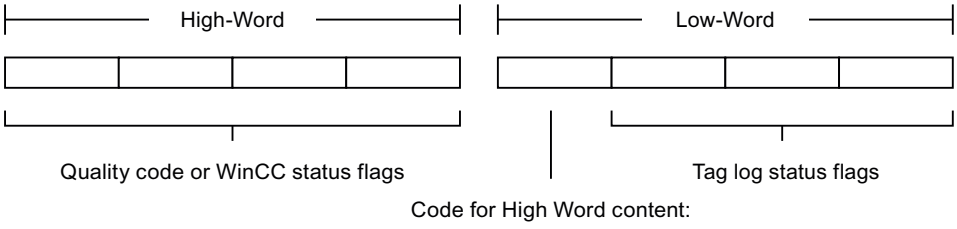

Structure of the 2-word value:

The High word contains the WinCC status flags or the quality code. The Low word contains the status flags for the data logging and an identifier for the content of the High word.

Code for High Word Content:

	Meaning
0x0	High word contains WinCC status flags
0x1	High word contains quality code

Tag logging status flags:

Name of flag	Value	Meaning
PDE_RT_DAYLIGHT	0x001	Summer time
PDE_RT_SUBSTITUTION	0x002	Substitute value
PDE_RT_TIME_BEVOR_JUMP	0x004	Value prior to time jump
PDE_RT_TIME_BEHIND_JUMP	0x008	Value after time jump
PDE_RT_TIME_OVERLAPPED	0x010	Values during time overlap
PDE_RT_LOAD_SYSTEM	0x020	Value that is initially archived when the log is created
PDE_RT_RELOAD_SYSTEM	0x040	Initial value after archiving of RT
PDE_RT_CMPCOPY	0x080	Compressed value
PDE_RT_TIME_CHANGED	0x100	Time change took place
PDE_RT_HAND	0x200	Manual tag supply

Examples

Value in database	16842753
Hexadecimal representation	0101 0001
Coding for high word	0: High word contains WinCC status flag
Data logging status flag	001: Summer time
WinCC status flag	0101: Link to partner not established; tag initialization value

Value in database	266242
Hexadecimal representation	0004 1002
Coding for high word	1: High word contains quality code
Data logging status flag	002: Substitute value
Quality codes	0004: Configuration error, value not accepted

2.9.3.3 Working with data logs (RT Professional)

Creating a data log (RT Professional)

Logging tags (RT Professional)

Principle

For the configuration of logs, the system distinguishes between the following log types:

- The data log stores process values in logging tags. When you configure the data log, select the process tags that are to be logged and the storage location.
- The compressed log compresses logging tags from data logs. The compressed log can further compress other compressed logs. When you configure the compressed log, select a calculation method and the compression time period.

Data buffer

For a data log, define whether the data buffer should be created on the hard disk or in the main memory.

In contrast to storage in the log database, process values logged in the main memory are only available while Runtime is active. Storing in the main memory has the advantage, however, that the values can be written and read very quickly. The process values stored in the main memory cannot be swapped out.

Note

Compressed logs can only be saved on the hard disk.

See also

Logging Settings (Page 660)

Configuring logging tags (Page 671)

Creating a data log (RT Professional)

Introduction

The configuration of a data log consists of the following steps:

- Create a data log.
- Configure the data log, for example, by selecting the storage location.
- Select the tags for logging.

Requirement

- A project is open.
- The Inspector window is open.

Procedure

To create a data log, proceed as follows:

1. Double-click on the "Historical Data" entry in the project tree.
   The editor for data logs and compressed logs opens.
2. Open the "Data logs" tab and double-click "Add" in the "Name" column of the "Data logs" editor.
   A new data log is created.
3. In the Inspector window select "Properties > Properties > General".
4. Enter a unique name for the log in the "Name" field.
5. In the "Storage location" field, decide whether the process values should be saved in the main memory or in a database.
6. If you select "Memory" as the storage location, set the number of data records to be logged per logging tag in the "Number of data records" field.
7. In the "Status" area, use the "Locked" option to determine whether the logging process is locked at the start of Runtime.
   If you want to start logging in Runtime, you will need to deactivate the option again.
8. If the status of the logging lock changes, you can be informed of this via a C function or configure subsequent steps. To do this, select an available C function under "Properties > Properties > Events" in the Inspector window. The C function is called in Runtime from the log server when the "Locked" status changes. You must first write the C function in the "Scripts" editor. See "Function for status check of the log lock (Page 672)" for additional information. Only one C function can be run by the "Changed lock" event.
9. If necessary, enter a descriptive text under "Comment" to document your project.

Alternatively you can configure log properties directly in the "Data log" editor. To view hidden columns, activate the column titles using the shortcut menu.

Result

The data log is created.

The next steps are to configure the higher-level logging parameters.
See "Specifying log settings (Page 660)" for additional information.

Note

The log name can be up to 32 characters long.

Do not use the following characters: '\\','\", '.', ',', ';', ':', '!', '?', '"', '´', '`', '^', '~', '-', '+', '=', '/', '*', '#', '%', '&', '§', '°', '(', ')', '[', ']','{', '}', '<', '>', '|', ' ', 'ä', 'ö', 'ü', 'Ä', 'Ö', 'Ü', 'ß'

Capitalization is ignored.

The characters _ @ # $ cannot be used as the first character of a name.

The following terms may not be used when assigning a name: "times", "userarchives", "project", "archives", "defaultarchives", "defaultvariables", "templates", "tabletemplateitems", "curvetemplateitems".

The names of logs must be unique in a project. Log names must also be unique if different storage locations are selected for different logs. You must also assign different names for alarm logs and data logs.

See also

Function for status check of the log lock (Page 672)

Specifying log settings (Page 660)

Creating a Compressed Log (RT Professional)

Introduction

The configuration of a compressed log consists of the following steps:

- Create the new compressed log.
- Configure the compressed log, for example, by selecting the calculation method and the time period for the compression.
- Select the logging tags that you want to add to the compressed log.

Requirement

- A project is open.
- The Inspector window is open.

Procedure

To create a compressed log, proceed as follows:

1. Double-click on the "Historical Data" entry in the project tree.
   The editor for data logs and compressed logs opens.
2. Open the "Compressed logs" tab and double-click "Add" in the "Name" column of the "Compressed logs" editor.
   A new compressed log will be created.
3. In the Inspector window select "Properties > Properties > General".
4. Enter a unique name for the log in the "Name" field.
5. Select the processing method and compression period in the "Compression" area.
   All times created in the "Cycles" editor that are >= 1 minute are available as the compression time period. If the compression cycle you want is unavailable, configure a new cycle time. Then select the new cycle as the compression time period.
6. Use the "Locked" option to determine whether the logging process is locked at the start of Runtime.
   If you want to start logging in Runtime, you will need to deactivate the option again.
7. If the status of the logging lock changes, you can be informed of this via a C function or configure subsequent steps. To do this, select an available C function under "Properties > Properties > Events" in the Inspector window. The C function is called in Runtime from the log server when the "Locked" status changes. You must first write the C function in the "Scripts" editor. See "Function for status check of the log lock (Page 672)" for additional information.
8. If necessary, enter a descriptive text under "Comment" to document your project.

Alternatively you can configure log properties directly in the "Compressed log" editor. To view hidden columns, activate the column titles using the shortcut menu.

Result

The compressed log is created.

Note

The log name can be up to 32 characters long.

Do not use the following characters: '\\','\", '.', ',', ';', ':', '!', '?', '"', '´', '`', '^', '~', '-', '+', '=', '/', '*', '#', '%', '&', '§', '°', '(', ')', '[', ']','{', '}', '<', '>', '|', ' ', 'ä', 'ö', 'ü', 'Ä', 'Ö', 'Ü', 'ß'

Capitalization is ignored.

The characters _ @ # $ cannot be used as the first character of a name.

The following terms may not be used when assigning a name: "times", "userarchives", "project", "archives", "defaultarchives", "defaultvariables", "templates", "tabletemplateitems", "curvetemplateitems".

The names of logs must be unique in a project. You must also assign different names for compressed logs and data logs.

See also

Function for status check of the log lock (Page 672)

Compressed log (Page 645)

Creating a Logging Tag (RT Professional)

Using Logging Tags (RT Professional)

Principle

Logging tags are used to save the process values to be logged. Use process tags to specify which process data should be logged. The data type is automatically defined by the process tag selected. Use the trigger mode to configure whether a tag is to be logged in relation to the time or specific events.

In a compressed log, each compressed logging tag is stored in a separate compressed logging tag.

Basic procedure

For binary and analog logging tags, specify the trigger mode (cyclic, etc.) as well as the acquisition and logging cycles. Select the parameters to start or end logging according to the trigger mode. Specify the display limits and parameters for processing the process value according to the type of logging tag.

When configuring a compressed logging tag, select the processing method for forming the logging tags to be compressed, such as the mean value.

Note

If you delete a logging tag, save the project and then create a new tag with the same name as the deleted tag; you can no longer access the previously saved values of the deleted tag for display or logging. Reason: The newly created logging tag is assigned a new ID. The ID of the deleted logging tag is no longer accessible.

Creating binary logging tag (RT Professional)

Introduction

To configure a binary logging tag, first create a logging tag and then connect it to a binary process tag. The data type of the logging tag will then be set automatically.

Then define the properties for the logging tag.

Requirement

- A data log has been created.
- The binary tag for which you want to configure the logging must already exist.
- The "Historical Data" editor is open.
- The Inspector window with the logging tag properties is open.

Procedure

To create a binary logging tag, proceed as follows:

1. Select the data log in which you want to create the logging tag from the table in the "Data log" editor.
2. In the "Name" column of the "Logging tags" table, double-click "Add".
   An logging tag is created.
3. Select "Properties > Properties > General" in the Inspector window and enter a unique name for the logging tag in the "Name" field.
4. In the "Process tags" field, select the binary tags to be linked to the logging tag.
   Use the "Locked" option to define whether logging should be enabled or disabled when Runtime starts. You will have to deactivate the option once again if you want to start the logging in Runtime.
5. To reuse the logged values for other purposes, you can also write the value of the logging tag to an internal tag. The update process is determined by the cycle settings for the logging tag. Select the required tag from the "Copy logged value to tag" field.
6. Select "Properties > Properties > Logging type" in the Inspector window and select the required log type.
   See "Trigger mode (Page 640)" for additional information.
7. Select "Properties > Properties > Processing method" in the Inspector window and select the required processing method.
   You can find more detailed information on this in the section "Cycles and Events (Page 640)".
8. If necessary, enter a descriptive text under "Comment" to document your project.

You can configure the properties of a logging tag directly in the "Data log" editor. To view hidden columns, activate the column titles using the shortcut menu.

You can also configure logging tags in the tag table in the "Logging tags" tab.

Result

The process values of the configured tag are logged in Runtime according to the selected settings.

See also

Trigger mode (Page 640)

Cycles and Events (Page 640)

Creating an analog logging tag (RT Professional)

Introduction

To configure an analog logging tag, first create a logging tag and then connect it to an analog process tag. The data type of the logging tag will then be set automatically.

Then define the properties for the logging tag.

Requirement

- The data log has been created.
- The analog tag for which you wish to configure the logging must already exist.
- The "Historical Data" editor is open.
- The Inspector window with the logging tag properties is open.

Procedure

To create an analog logging tag, proceed as follows:

1. Select the data log in which you want to create the logging tag from the table in the "Data log" editor.
2. In the "Name" column of the "Logging tags" table, double-click "Add".
   An logging tag is created.
3. Select "Properties > Properties > General" in the Inspector window and enter a unique name for the logging tag in the "Name" field.
4. In the "Process tags" field, select the analog tags to be linked to the logging tag.
   Use the "Locked" option to define whether logging should be enabled or disabled when Runtime starts. You will have to deactivate the option once again if you want to start the logging in Runtime.
5. To reuse the logged values for other purposes, you can also write the value of the logging tag to an internal tag. The update process is determined by the cycle settings for the logging tag. Select the required tag from the "Copy logged value to tag" field.
6. Select "Properties > Properties > Logging type" in the Inspector window and select the required log type.
   See "Trigger mode (Page 640)" for additional information.
7. Select "Properties > Properties > Deadband for logging" in the Inspector window and enter the required limits. Open the selection box and select the "Constant" entry to enable the input. Then enter the required limit value. Process values outside the limits will not be logged.

8. Select "Properties > Properties > Processing method" in the Inspector window and select the required processing method.
   For additional information see "Cycles and Events (Page 640)".
   You can specify a C function for converting tag values if you use the "Calculation" processing method. You must first write the C function in the "Scripts" editor. For additional information see "Converting values from logging tags (Page 673)".
9. If necessary, enter a descriptive text under "Comment" to document your project.

Alternatively you can configure log properties directly in the "Data log" editor. To view hidden columns, activate the column titles using the shortcut menu.

Result

The process values of the configured tag are logged in Runtime according to the selected settings.

See also

Trigger mode (Page 640)

Cycles and Events (Page 640)

Converting values from logging tags (Page 673)

Creating a Compressed Log Tag (RT Professional)

Introduction

You can save the process values of a logging tag or a compressed tag from another compressed log in compressed form in a compressed logging tag. Select the mathematical function with which you want to compress the process values it contains.

The following compression functions are available:

- Weighted mean
- Maximum
- Minimum
- Mean value
- Sum

You create a compressed logging tag in the "Compressed log" editor.

Requirement

- You have created a compressed variable.
- You have created the logging tag you want to configure.
- The "Historical Data" editor is open.
- The Inspector window with the logging tag properties is open.

Procedure

To create a compressed logging tag, proceed as follows:

1. Click the "Compressed logs" tab in the "Historical Data" editor.
   The "Compressed logs" editor opens.
2. Select the compressed log in which you want to create the compressed logging tag from the table in the "Compressed log" editor.
3. Click on the vertical "Logging tag" tab on the right side of the window. The "Logging tags" task card opens. The task card shows all objects you can add to the selected compressed log.

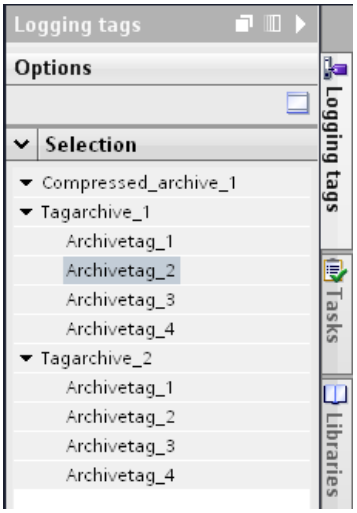

4. In the task card, select the objects you want to add to the compressed log. You can select the following objects:
   - Single or multiple logging tags
   - A complete data log
   - One or several compressed logging tags
   - A complete compressed log
5. Open the shortcut menu of the selected object and select the menu command "Add." A compressed logging tag is created for each object selected and linked to the respective object. If you have selected a complete data log or a complete compressed log, a compressed logging tag is created for each tag contained in this log.
   Alternatively, you can drag-and-drop the selected objects into the compressed log selected in step 2.
6. Select one or several compressed logging tags in the table of compressed logging tags. Specify more properties for the selected tags in the Inspector window.
   Use the "Locked" option to define whether logging should be enabled or disabled when Runtime starts. If you want to start logging in Runtime, you will need to disable the option again.
   Select the processing method for the compression in the "Compression" field.
7. If necessary, enter a descriptive text under "Comment" to document your project.

You can configure the properties of a compressed logging tag directly in the "Compressed log" editor. To view hidden columns, select these from the shortcut menu for the column titles.

Result

The compressed logging tags are connected to the selected logging tags. The values from the logging tag are compressed using the set parameters in Runtime.

If you have selected the compressed logging tags from another compressed log, they will be compressed even further.

See also

Compressed log (Page 645)

Specifying log settings (RT Professional)

Logging Settings (RT Professional)

Introduction

You configure the logical organization of data logs and compressed logs in the "Historical Data" editor. You can configure the basic properties of logs in Runtime Professional, such as the segment size, in the "Runtime settings > Logging" dialog.

You make the settings for data logs separately in the dialog for the "Fast data log" and "Slow data log" types.

The following conditions apply if you are using the default settings:

The Fast data log is used to store all values with a logging cycle <= 1 minute.

The Slow data log is used to store all values with a logging cycle > 1 minute.

You can change the cycle limit for allocation to the log type in the "Cycle" field in the "Runtime settings > Logging" dialog.

Overview of procedure

You configure the higher-level log configuration in the "Runtime settings > Logging" dialog. The following instructions contain a description of the configuration procedure:

Step	Description
1	Defining the Log Size and Segmentation (Page 661)
2	Configuring the log backup (Page 665)
3	Assigning Logging Tags to the Log Types (Page 667)

See also

Defining the Log Size and Segmentation (Page 661)

Configuring the log backup (Page 665)

Connecting the log backup (Page 669)

Disconnecting a log backup (Page 670)

Assigning Logging Tags to the Log Types (Page 667)

Logging tags (Page 651)

Swapping Out Process Values (Page 648)

Defining the Log Size and Segmentation (RT Professional)

Introduction

In the "Runtime settings > Logging" dialog, you define the sizes of the "Fast data log" and "Slow data log" types in the database. You can also subdivide the two types of log into segments. The size of the individual segments thus also defines the number of individual segments.

The individual segments are filled one after the other in Runtime. Once a segment is totally full, it switches to the next segment. You can also configure the segment change at specific times. If you define a time for the segment change, the next log segment is filled when the time is reached.

Note

Ensure that the log size does not exceed the free memory space available. The system does not validate the selected settings. A high number of linked log segments can lead to longer waiting periods in the system when starting and ending Runtime.

Requirement

A project is open.

Procedure

To edit the higher-level log settings, proceed as follows:

1. In the project tree, open the folder containing Runtime Professional and double-click the "Runtime Settings" entry.
   The Runtime settings dialog opens.

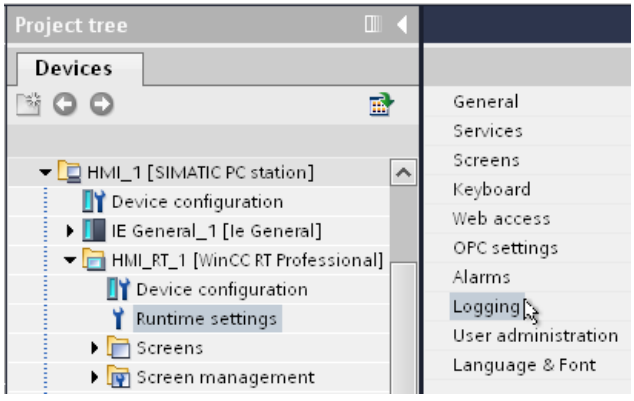

2. Click the "Logging" entry in the dialog.
   The "Logging" dialog opens.

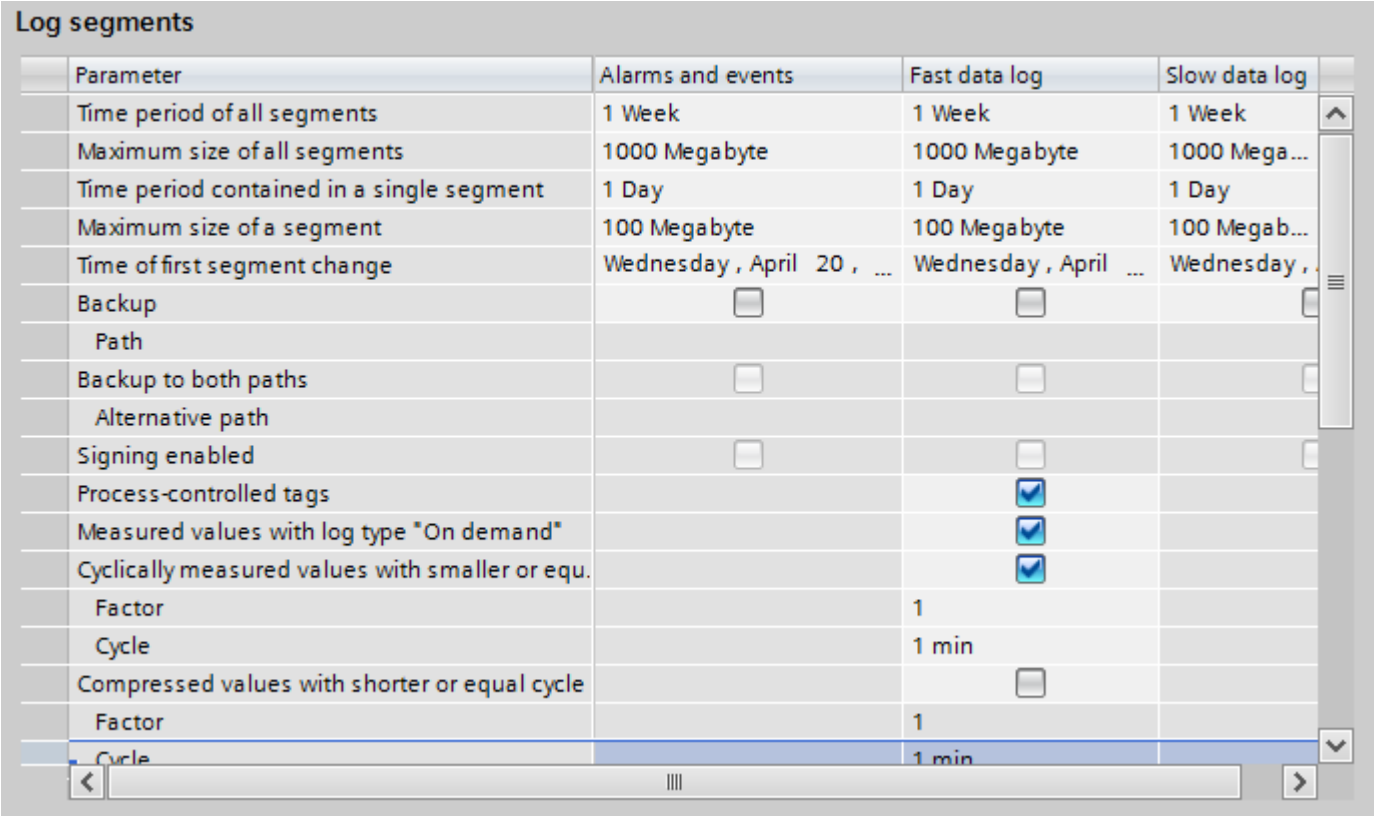

Log segments

Parameter	Alarms and events	Fast data log	Slow data log
Time period of all segments	1 Week	1 Week	1 Week
Maximum size of all segments	1000 Megabyte	1000 Megabyte	1000 Mega...
Time period contained in a single segment	1 Day	1 Day	1 Day
Maximum size of a segment	100 Megabyte	100 Megabyte	100 Megab...
Time of first segment change	Wednesday , April 20 , ...	Wednesday , April ...	Wednesday ,
Backup	☐	☐	☐
Path			
Backup to both paths	☐	☐	☐
Alternative path			
Signing enabled	☐	☐	☐
Process-controlled tags		☑	
Measured values with log type "On demand"		☑	
Cyclically measured values with smaller or equ.		☑	
Factor		1	
Cycle		1 min	
Compressed values with shorter or equal cycle		☐	
Factor		1	
Cycle		1 min	

3. Enter the required values in the "Time period of all segments" and "Maximum size of all segments" fields.
   – When you enter the time period across all the segments and the maximum size, you also define the size of the logging database. If either of these criteria is violated, a new segment is started. When all the segments are filled, the oldest segment is overwritten.
4. Enter the required values in the "Time period contained in a single segment" and "Maximum size of a segment" fields.
   – When you enter the time period for the single segments and the maximum size, you also define the size and number of single segments. If either of these criteria is violated, a new individual segment is started. When the criterion for "Time period of all segments" is exceeded, the oldest individual segment is also deleted.
5. Enter the start date and start time for the first segment change in the "Time of first segment change" field.
6. Continue the configuration from step 2 of the procedural overview and configure the log backup. See the "AUTOHOTSPOT" section for additional details.

Note

If you change the log size or the time period in Runtime, the change will not take effect until the next segment change.

Example

The input was configured:

Property	Value
Time period of all segments	1 week
Maximum size of all segments	700 MB
Time period contained in one single segment	1 day
Maximum size of a segment	100 MB
Time of first segment change	Friday, 23 November 2010 18:30

With the configuration shown in the table, the started segment will be changed for the first time at 18.30 on 23 November 2010. The next time-controlled segment change will take place cyclically, after periods of 1 day since the configured time. The segment will also change if the configured size of 100 MB is exceeded in the course of one day. The oldest single segment will be deleted if the maximum log size of 700 MB is exceeded.

Changing the log type

If the user does not change the relevant settings, the logging tags will be logged with a cycle time of <= 1 min in the "Fast data log". Logging tags with a cycle time > 1 min will be logged in the"Slow data log". The cycle time of a logging tag must not be changed in Runtime beyond the above limits. It is not possible to change of a logging tag from "Fast" to "Slow" log type and vice-versa in Runtime.

After a cycle change or reconfiguration, if tags are to be stored in a different log type, these tags will be read from the currently valid log. Previous log values of this tag are not accessible in Runtime.

If you reconfigure the logging tags so that their values are stored in a log other than the previous log type, more memory will be required in the log concerned. You should therefore adjust the log size after making major changes.

Note

Runtime data in the logs will be deleted if you reset the log configuration for data logging. Only previously swapped-out databases remain intact.

Resetting the log database

Use the "Extended loading" dialog to reset the log database settings. Select the HMI device in the project tree for this purpose and use the menu command "Online > Extended download to device". Select the interface for your HMI device. Click on "Download". The "Load preview" dialog opens. Make the settings for the reset in the "Load preview" dialog.

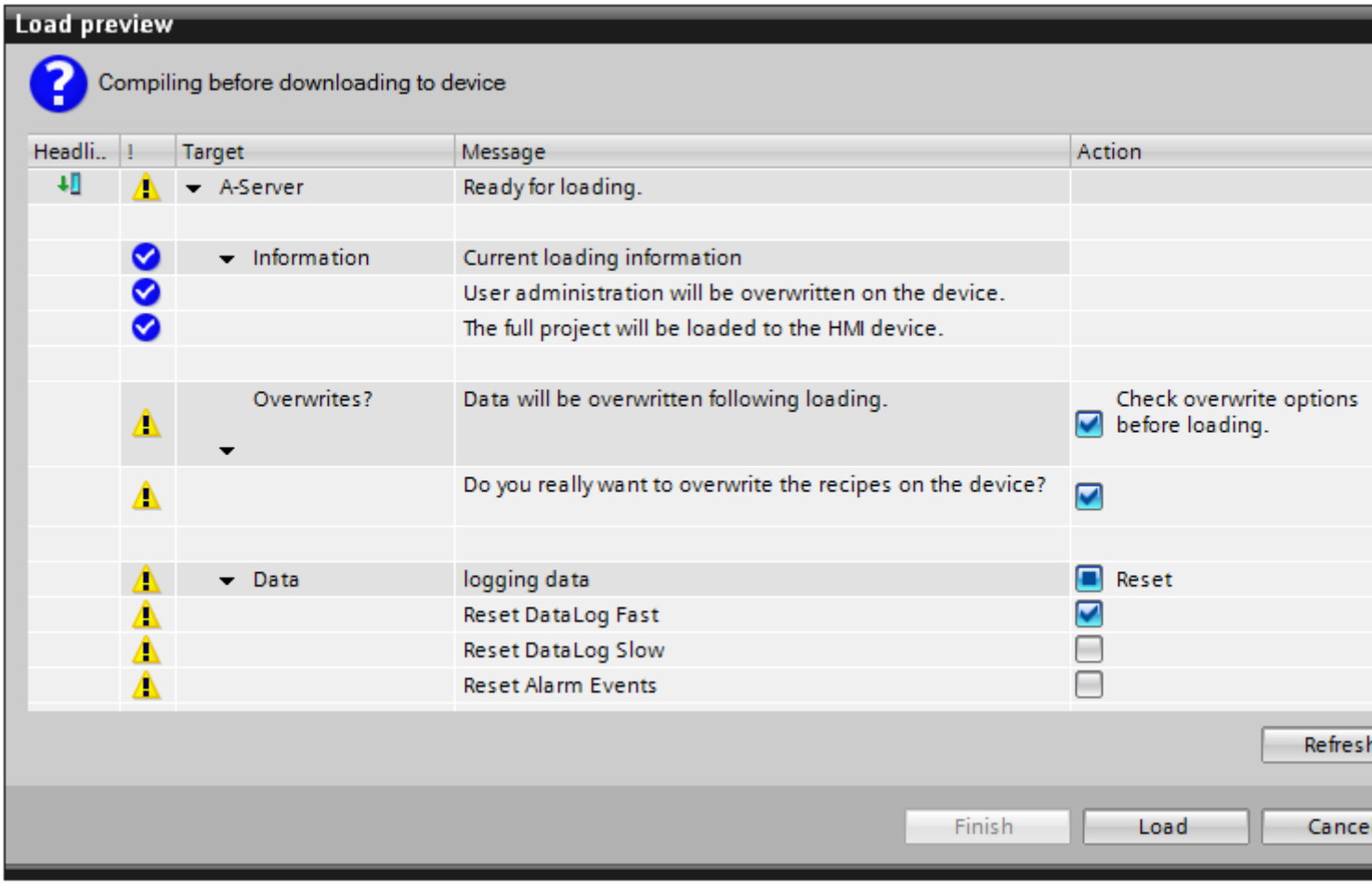

The reset will be executed when you start the download process.

Configuring the log backup (RT Professional)

Introduction

Make regular backups of your log data to ensure complete documentation of your process.

Note

The backup normally starts 15 minutes after the first time-related segment change. If the start of backup and start of segment should be synchronous with the start of Runtime, define the start time for the segment change prior to the start of Runtime.

You can change a displayed process value in Runtime using the table view. If the location of the log segment where the process value is stored has already been changed, then the modified value is not accepted in the shifted log. The change only occurs in the local log segment.

If the log segment has not yet been swapped, the changed value is accepted permanently.

Requirement

- Step 1 of the procedural overview is complete - Defining the Log Size and Segmentation (Page 661)
- The "Runtime settings > Logging" dialog is open.

Procedure

Proceed as follows to create a log backup:

1. Activate the "Backup" option in the column containing the type of log for which you want to configure the backup, such as the "Fast data log" column.
2. In the same column, enter the storage location for the backup in the "Path" field or navigate to the desired storage location in the file system using the dialog. A network drive is also possible as the storage location.
3. If you want to create a redundant backup, select the "Backup to both paths" option.
4. Enter an alternative storage location for the backup in the "Alternative path" field, or navigate to the desired storage location in the file system using the dialog. A network drive is also possible as the storage location. The alternative storage location is used in the following situations:
   - If the storage space on a backup medium is full.
   - If the original storage location is unavailable due to a network failure, for example.

   After the corresponding system alarms have been configured, the alarms are output, if the specified storage location is not available.

5. If the log backup file should be signed, select the "Enable signing" option. On reconnecting to WinCC, the signature will allow the system to detect whether any changes were made to a log backup file after it was swapped out.
6. Continue the configuration from step 3 of the procedural overview and assign the logging tags to the log types. See the "Assigning Logging Tags to the Log Types (Page 667)" section for more detailed information.

Structure of the Log Backup File

A log backup consists of two files with the extensions "*.LDF" and "*.MDF". To transfer a log backup to another computer, for example, copy the corresponding LDF and MDF files. The file name is composed as follows: "<Computer name>_<Project name>_<Type>_<Period_from>_<Period_until>". The type is defined by the log type:

- TLG_F: Data log with a logging cycle of one minute or less.
- TLG_S: Data log with a logging cycle of more than one minute.

The time period is specified in the following format: yyyymmddhhmm, e.g. 201012021118 for December 2, 2010, 11:18. Underscores "_" in the project name are displayed as "#".

Example:

FLEX-LOG2_WINCC#10#11#29#14#37#40_TLG_F_201011291414_201012030502.ldf

FLEX-LOG2_WINCC#10#11#29#14#37#40_TLG_F_201011291414_201012030502.mdf

Signing of the log backup files

If signing and backup are enabled, each log backup file is signed when it is swapped out. When the file is connected to WinCC once more, the signature is used to determine whether the file was changed after being swapped out.

If you use log signing, the maximum size of a single segment must not exceed 200 MB.

To verify the data, the "Signing enabled" option must be checked. Connecting "Slow" logs will take longer due to the signing.

The following values apply to the logging of signed data:

Logging signed data in Runtime	Value/seconds
Fast data log	1,000
Slow data log	500 [1]

[1] With "Slow" type data logs, expect longer screen selection times for a given amount of data as those for "Fast" type data logs.

Note

If you disable signing off to establish a fast connection to the backup files, for example, you should avoid segment changes while it is disabled. After the connection has been established, signing off should be reactivated in order to enable signing off for logged data.

Activating and deactivating signing

You activate and deactivate signing in the Runtime settings of the HMI device.

Activate or deactivate signing. If Runtime is running on the engineering station, recompile the project. The change becomes effective after compilation. If Runtime is running on a different PC than WinCC ES, you accept the change with online delta transfer. See Overview of online delta transfers (Page 2291) for additional information.

See also

Defining the Log Size and Segmentation (Page 661)

Assigning Logging Tags to the Log Types (Page 667)

Overview of online delta transfers (Page 2291)

Connecting the log backup (Page 669)

Disconnecting a log backup (Page 670)

Assigning Logging Tags to the Log Types (RT Professional)

Introduction

Data logging uses two different types of log to store data:

- Fast data log
- Slow data log

The logging tags are automatically allocated to the correct type of log. You can change this allocation in the log settings.

In Runtime, a changed assignment will only be accepted when the project has been deactivated and Runtime is restarted.

Requirement

- Step 2 of the procedural overview is complete - Configuring the log backup (Page 665)
- The "Runtime settings > Logging" dialog is open.

Procedure

To allocate the logging tags to the log types, follow these steps:

1. If you want to log acquired values to the fast data log on demand, activate the "Measured values with acquisition mode "On demand"" option in the "Fast data log" column.

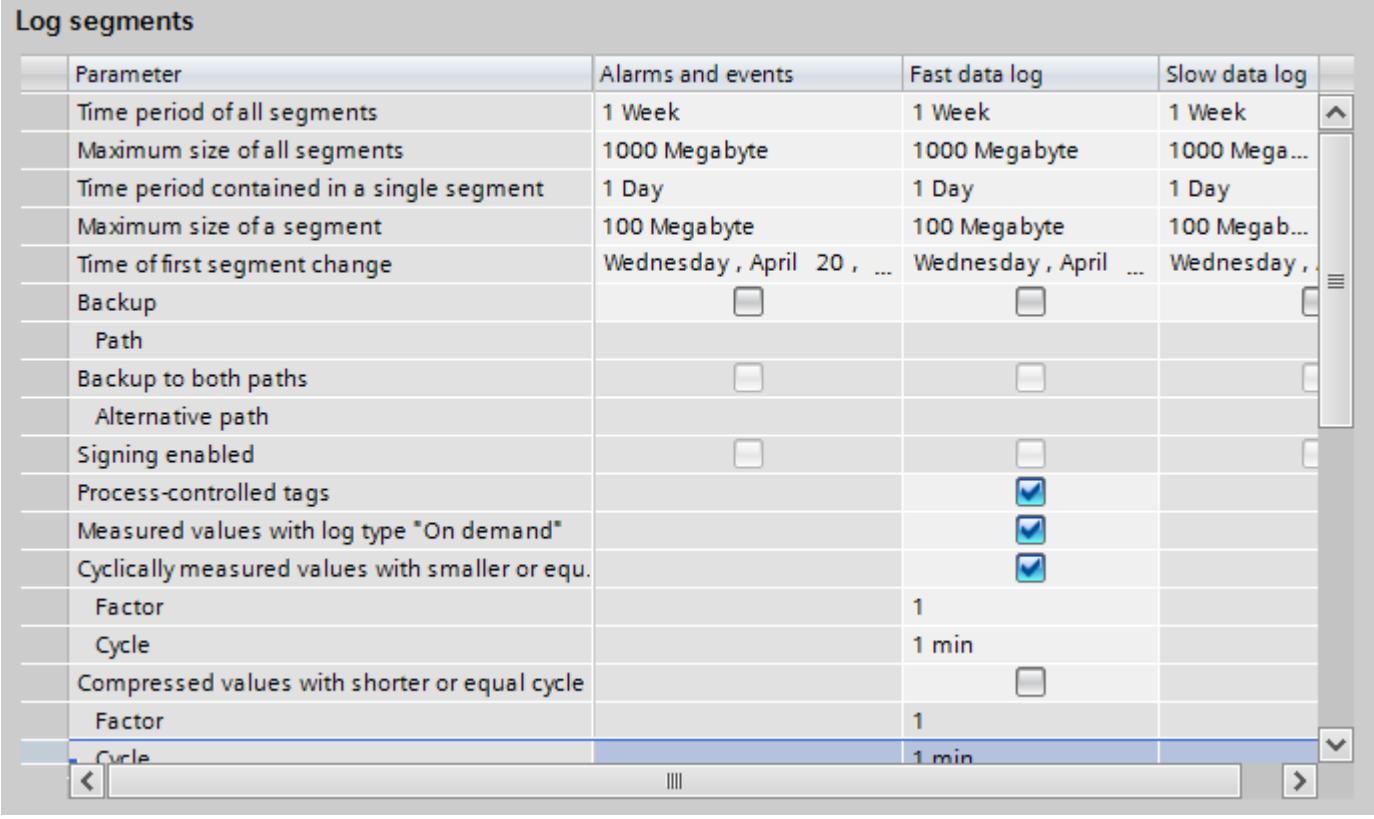

Log segments

Parameter	Alarms and events	Fast data log	Slow data log
Time period of all segments	1 Week	1 Week	1 Week
Maximum size of all segments	1000 Megabyte	1000 Megabyte	1000 Mega...
Time period contained in a single segment	1 Day	1 Day	1 Day
Maximum size of a segment	100 Megabyte	100 Megabyte	100 Megab...
Time of first segment change	Wednesday , April 20 , ...	Wednesday , April ...	Wednesday ,
Backup	☐	☐	☐
Path			
Backup to both paths	☐	☐	☐
Alternative path			
Signing enabled	☐	☐	☐
Process-controlled tags		☑	
Measured values with log type "On demand"		☑	
Cyclically measured values with smaller or equ.		☑	
Factor		1	
Cycle		1 min	
Compressed values with shorter or equal cycle		☐	
Factor		1	
Cycle		1 min	

2. Activate the "Cyclically measured values with cycles less or equal" option if you want to log logging tags to the fast data log with the "cyclic" trigger mode.
3. Select a cycle time as the upper limit of the logging cycle in the "Cycle" field. If necessary, select a factor for multiplying the selected cycle time in the "Factor" field.
4. Enable the "Compressed values with shorter or equal cycle" option if you want to log all compressed values with a logging cycle that is less than or equal to the limit allocated to the Fast data log.
5. Select a cycle time as the upper limit of the logging cycle in the "Cycle" field. If necessary, select a factor for multiplying the selected cycle time in the "Factor" field.
6. Save the project.

Result

All logging tags to which these settings apply are logged in the Fast data log. All logging tags to which these settings do not apply are logged in the Slow data log.

Connecting the log backup (RT Professional)

Introduction

In order to access data of a log backup in Runtime, connect the log backup to the project again. You can establish the connection using a VB function or have it established automatically. You can connect all backup segments that were created with each project. A requirement is that you did not change the log or the logging tags.

Requirement

- The corresponding LDF and MDF files for the log backup are located in a local folder on the configuration computer, such as the hard disk, MOD or CD.

Procedure

Proceed as follows to connect a log backup using a VB function:

1. Configure a button in a process screen and label the button "Connect log backup", for example.
2. Write a VB function using the "Restore" method based on this example:
   HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Restore("D:\Backup","","2010-11-14",-1) & vbNewLine
   You can find more detailed information about this in "AUTOHOTSPOT".
3. Configure a list of functions for the "Click" event for the configured button.
4. Select the VB function thus created from the list of functions.
5. Save the project.

Result

The VB function is executed in Runtime when the button is clicked. The log backup is connected using the parameters defined in the function.

The logged process values from the log backup are sorted in time-stamp order in the configured display.

Connecting "Tag Logging Slow" logs takes longer if signing is enabled.

Automatic linking to a log

1. Insert the log backup files into the "CommonArchiving" folder.
   The "CommonArchiving" folder is located in the project folder of your WinCC project.
2. In Runtime, the data log is automatically connected to the project.

If signing is enabled, signed log backup files that are changed will not be connected automatically. A WinCC system message is generated and an entry is added to the Windows event log in the "Application" section.

See also

Configuring the log backup (Page 665)

Disconnecting a log backup (Page 670)

Disconnecting a log backup (RT Professional)

Introduction

If you have connected a log backup in Runtime, you can also disconnect the connection once more. You can disconnect the connection using a VB function or manually remove the log backup files from the "CommonArchiving" folder.

Requirement

- The corresponding LDF file and MDF file of the log backup are located in the "CommonArchiving" folder in the project folder of the WinCC project.

Procedure

Proceed as follows to disconnect a log backup using a VB function:

1. Configure a button in a process screen and label the button "Disconnect log backup", for example.
2. Write a VB function using the "Remove" VBS method based on this example:
   HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Remove("","2010-11-14",-1) & vbNewLine
   You can find more detailed information about this in "AUTOHOTSPOT".
3. Configure a list of functions for the "Click" event for the configured button.
4. Select the VB function thus created from the list of functions.
5. Save the project.

Result

The VB function is executed in Runtime when the button is clicked. The log backup is disconnected from the log database. You will no longer have access to the data from the backup file in Runtime.

Alternatively, you can manually remove the backup files from the "CommonArchiving" folder. The "CommonArchiving" folder is found in the project folder of your WinCC project.

See also

Configuring the log backup (Page 665)

Connecting the log backup (Page 669)

Logging tags in the "Tags" editor (RT Professional)

Configuring logging tags (RT Professional)

Introduction

You can also create and edit logging tags in the "HMI tags" editor in WinCC. You also directly edit the properties of logging tags in the "Tags" editor.

Note

If you delete, move or copy in the "HMI tags" editor, the changes also take effect in the "Logs" editor.

Requirement

- The "Logging tags" tab is open in the "HMI tags" editor.
- You have created a tag.

Procedure

Proceed as follows to configure a logging tag in the "HMI tags" editor:

1. Select an existing tag in the tag table.
   Alternatively, double-click "Add" in the "Name" column to create a new tag.
2. Click on "Logging tags" in the lower editor. The editor for logging tags is brought to the foreground.

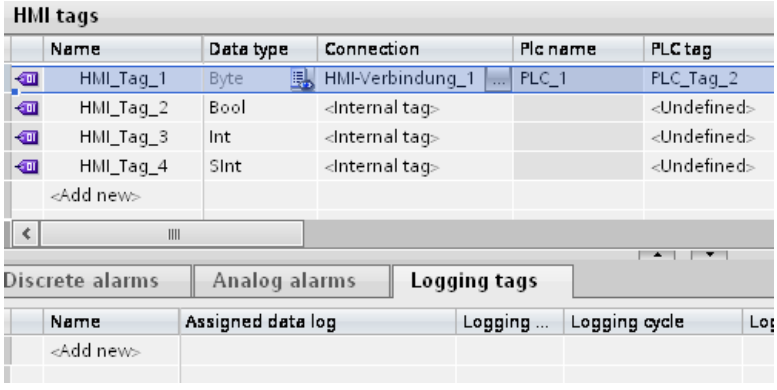

3. Double-click "Add ..." in the "Name" column of the editor "Logging tag" table. A new logging tag is created. The logging tag is linked to the tag selected in the first step. The data type of the logging tag is taken automatically from the linked tag.

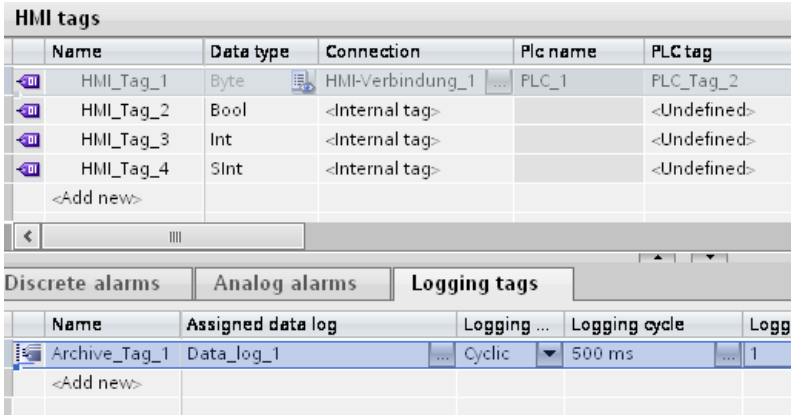

4. Select the desired logging tag from the "Data log" field. You can also create a new log using the object list.
5. Configure additional parameters for logging in the table of the editor or in the Inspector window.

Result

The configured logging tag is created in the "Tags" editor and is displayed in various editors.

See also

Logging tags (Page 651)

Functions for data logging (RT Professional)

Function for status check of the log lock (RT Professional)

Introduction

You can monitor the "Locked" state of a data log using a C function. You need to create the C function in the "Scripts" editor with the correct syntax.

Description

This function is called in Runtime from the log server when triggered by a "Locked" status change. You can assign this function to the event "Lock state change" in the "Events" group in the "Data logs" or "Compressed logs" editor. The C function is then only displayed under the "Lock changed" event when the syntax below is correctly written in the function.

Syntax

```
void function name (BOOL fFlag);
```

Parameters

fFlag

TRUE - log is locked

FALSE - log is released

Example application

You can use this function to receive information about the lock status of a log, for example.

Converting values from logging tags (RT Professional)

Description

You can use this C function to convert tag values before logging them. This C function is assigned in the "Logging tags" editor in the properties of the logging tag under "Properties > Processing method". The C function is then only offered for selection in the "Analog processing method" field when the syntax below is correctly written in the function.

Syntax

```
double function name (double dLastValue,
 double dCurrentValue,
 DWORD dwCount,
 BOOL fArchivingCycle);
```

Parameters

dLastValue: Return value of the function that was triggered in the last acquisition cycle

dCurrentValue: currently acquired value

dwCount: current acquisition cycle since the most recent log cycle (for example, log cycle is 4 x 1 seconds of acquisition cycle; the count is then 1 to 4, until the count starts again).

fArchivingCycle: TRUE = cycle is log cycle (default); FALSE = cycle is acquisition cycle

Example application

You can use this function to implement your own mean value calculation. The calculated mean value is used instead of the current tag value.

The following function determines the geometric mean value of the acquired process values:

```
double CalculateGeometricMean (double dLastValue,double
dCurrentValue,DWORD dwCount,BOOL fArchivingCycle)
{

static double dCurrentProduct = 1.0;

    double dReturnValue = 0.0;

if (!fArchivingCycle)

{

// 1. continuously calculate the product of all values with every
scan cycle

dCurrentProduct *= dCurrentValue;

printf ("current value=%f current product=%f archiving cycle flag=%d
\r\n",dCurrentValue,dCurrentProduct ,fArchivingCycle);

dReturnValue = dCurrentProduct ; // optinonally - the return value
is actually not of interest in a scan cycle

}

else

{

// 2. if this is the archiving cycle, use the product of all scan
values to calculate

// the geometric mean value

dReturnValue = pow (dCurrentProduct ,(double) 1 / dwCount);

      printf ("archiving cycle: geometric mean value=%f\r
\n",dReturnValue);

         printf ("\r\n");

         dCurrentProduct = 1.0; // reset current product

// the finally calculated value will be returned - this value will
be archived

// since this is the archiving cycle

}

return dReturnValue;
```

}

2.10 Displaying Tags

2.10.1 Displaying tags with Basic Panels (Basic Panels)

2.10.1.1 Outputting tag values in screens (Basic) (Basic Panels)

Introduction

In runtime you can output tag values in the screens of the operator device in the form of a trend. A trend is a graphic representation of the values that a tag takes during runtime. Use the "Trend display" graphic object to represent it. Process values for the trend display are loaded by the PLC from the ongoing process.

The values to be displayed are determined individually within a fixed, configurable cycle. Cyclically-triggered trends are suitable for representing continuous curves, such as the changes in the operating temperature of a motor.

Displayed values

You will need to configure a trend view in a screen so that tag values are displayed on the HMI device. When configuring the trend view, specify which tag values are to be displayed.

You can control the updating of the trend display by defining the cycle time.

See also

Configuring trend display for values from the PLC (Basic) (Page 676)

Basics of tags (Page 519)

2.10.1.2 Configuring trend display for values from the PLC (Basic) (Basic Panels)

Introduction

You use a trend view to graphically represent values that a tag assumes during the process.

Requirement

- A screen is open.
- The Inspector window with the trend view properties is open.

Procedure

To configure a trend view, follow these steps:

1. Add the "Trend view" object from the toolbox in the "Control" group to the screen.

2. Select the "Trend" category from the "Properties" group in the Inspector window and double-click "<Add>" in the "Name" column.

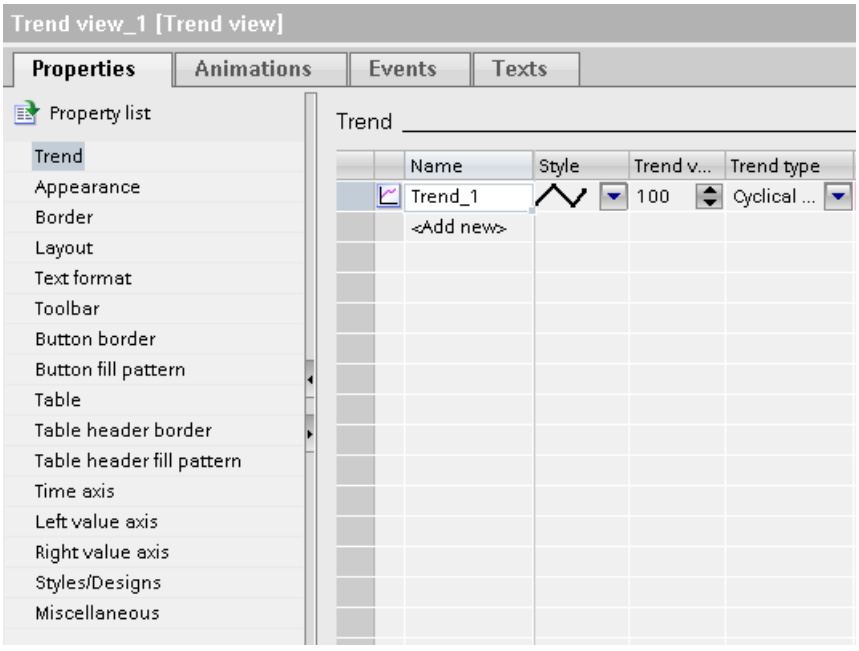

3. Assign a name to the trend in the "Name" column.
4. In the "Style" column, use the selection button to open the "Style" dialog and select the style of the line.
5. Select the number of trend values in the "Trend values" column.

6. In the "Settings" column, use the selection button to open the "Data source" dialog and select the tags to supply the trend with values.
   Specify the cycle for reading the tags from the PLC.

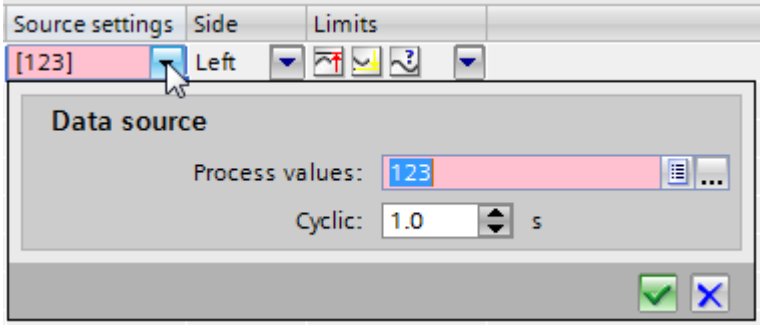

7. You can make other settings in the dialogs of the Inspector window. For example, you can select the "Display table" option in the "Table" category to display a value table beneath the trend view.

Note

If you hold down the <CTRL> key and double-click the trend view, the trend view is activated. You set the column width and the position of the columns in the table header of the values table in active mode. In order to activate the trend view the zoom factor has to be set to 100 %.

Result

In runtime, the values of the selected tags are displayed in the configured trend view.

See also

Outputting tag values in screens (Basic) (Page 676)

2.10.2 Displaying tags with Runtime Advanced and Panels (RT Advanced, Panels, Comfort Panels)

2.10.2.1 Trends (RT Advanced, Panels, Comfort Panels)

Introduction

A trend is a graphic representation of the values that a tag takes during runtime. In order to display trends, configure a trend view in a screen of your project.

To configure the trend view, specify a trend type for the values to be displayed.

- Log: For displaying the logged values of a tag
- Cyclical real time: For time-triggered display of values

- Bit-triggered real time: For event-triggered display of values
- Bit-triggered buffer: For event-triggered display with buffered data acquisition

Displaying logged values

In Runtime, the trend view displays the values of a tag from a data log. The trend shows the logged values in a particular window in time. The operator can move the time window in Runtime to view the desired information from the log.

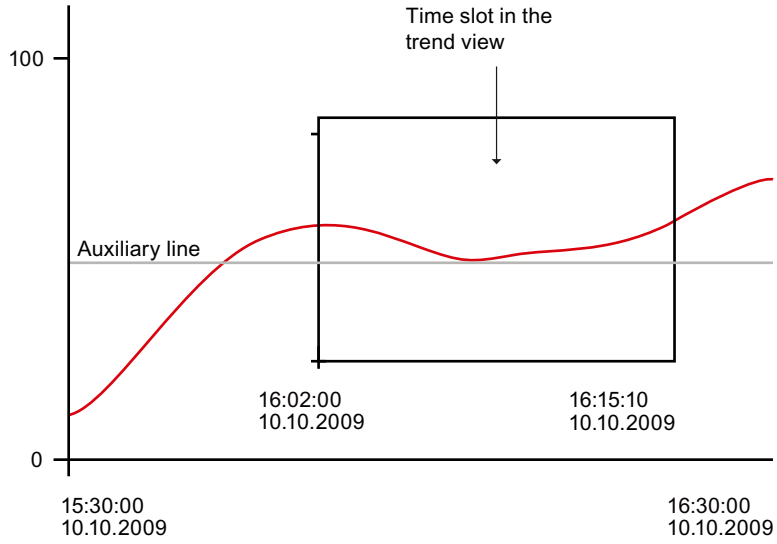

Pulse-triggered trends

The values to be displayed are determined individually with a definable time pattern. Cyclically-triggered trends are suitable for representing continuous curves, such as the changes in the operating temperature of a motor.

Bit-triggered trends

The values to be displayed are event-driven by setting a defined bit in "Trend transfer" tags. The bit is reset after reading has been completed. Bit-triggered trends are useful for displaying fast-changing values, such as the injection pressure for producing plastic parts.

Bit-triggered trends with buffered data acquisition

When you set buffered data acquisition, the values to be displayed are buffered in the PLC and read in bit-triggered as a block. These trends are suitable for displaying rapid changes when the course of the trend as a whole is interesting and not so much the individual values.

You configure a switch buffer in the PLC so it can continue to write the new values while the trend buffer is being read. The switch buffer ensures that the PLC does not overwrite values while the operator devices reads the values for the trend.

The switch between the trend buffer and the switch buffer functions as follows:

Whenever the bit assigned to the trend is set in the "Trend transfer 1" tag, all the values are read simultaneously from the trend buffer and are displayed as a trend on the HMI device. The bit in "Trend Transfer 1" is reset after reading has been completed.

While the operator device is reading the tag values from the trend buffer, the PLC writes the new tag values into the switch buffer. When the bit which is assigned to the trend is set in the "Trend transfer 2" tag , all the trend values are read from the switch buffer and displayed at the operator device. While the operating device is reading the switch buffer, the PLC writes again to the trend buffer.

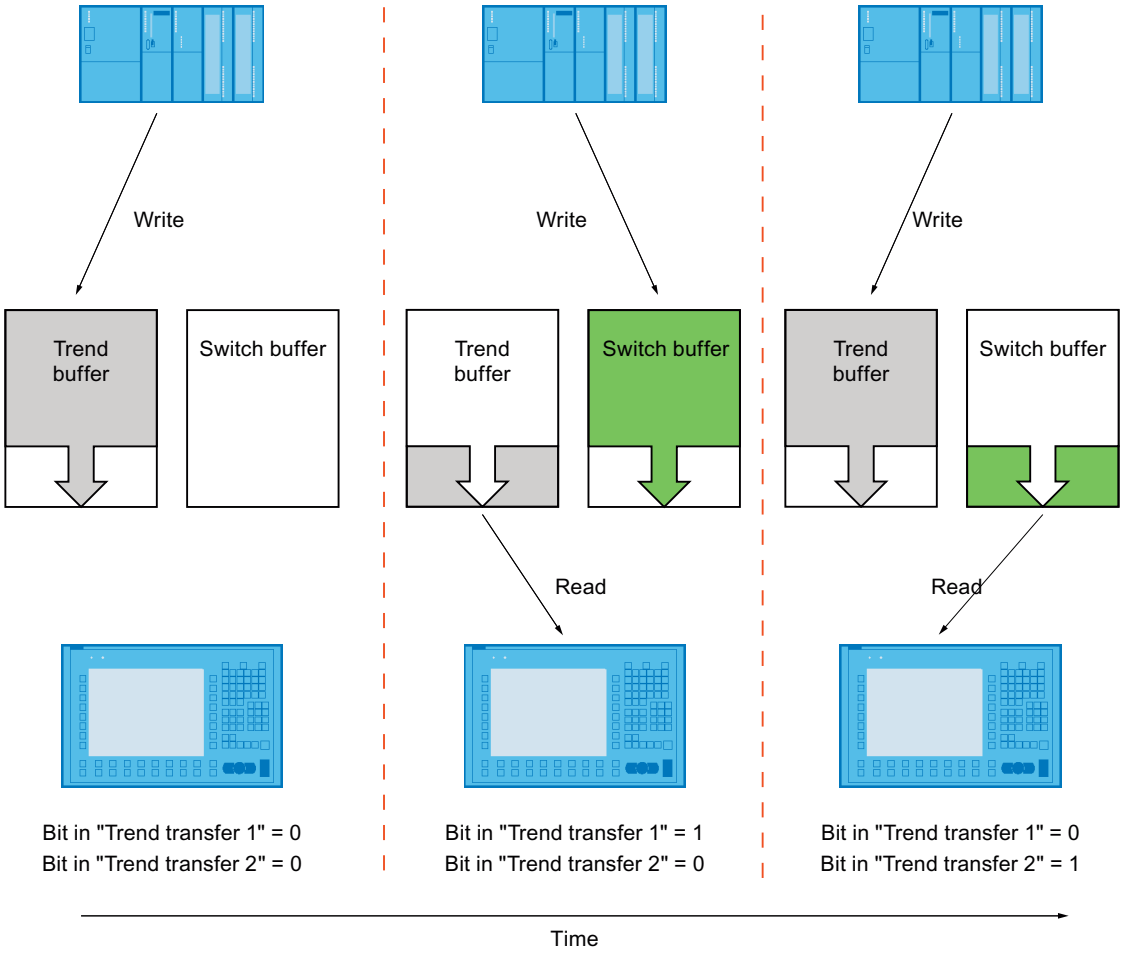

2.10.2.2 Outputting tag values in screens (RT Advanced, Panels, Comfort Panels)

Introduction

In runtime you can output tag values in the screens of the operator device in the form of a trend. The process values are loaded by the PLC from the current process or from a log database.

Displayed values

You will need to configure a trend view in a screen so that tag values are displayed on the HMI device. When configuring the trend view, specify which tag values are to be displayed.

The following options are available:

- Current values from the PLC
  Writing of the trend is continued with individual values from the PLC. The following trigger types are available for display in real time:
  - Bit-triggered real time: The time of the update can be controlled by setting a bit. Bit-triggered trends are normally used to visualize rapidly changing values. One example might be the injection pressure in the production of plastic parts.
  - Cyclical real time: You control the update time with a cycle. Cyclically-triggered trends are suitable for representing continuous curves, such as the changes in the operating temperature of a motor.
  - Bit-triggered buffer: The trend contains values that were stored in a buffer in the period between two reads from the PLC. Use this format pattern if the course of the trend as a whole is more interesting than the individual values.
- Logged tag values
  In runtime, the trend view displays the values of a tag from a data log. The following trigger type is available to display logged tag values:
  - Data log: The trend shows the logged values in a particular window in time. To obtain the requested information from the archive, the operator uses the corresponding operator controls in the screen or in the trend view.

2.10.2.3 Configuring trend displays for values from the PLC (RT Advanced, Panels, Comfort Panels)

Introduction

You use a trend view to graphically represent values that a tag assumes during the process.

Requirement

- A screen is open.

Procedure

To configure a trend view, follow these steps:

1. Add the "Trend view" object from the toolbox in the "Controls" group to the screen.
2. Select "Properties . Properties > Trend" in the Inspector window and double-click "Add" in the "Name" field. A new trend is created.
3. If required, enter a unique name for the trend in the "Name" field.
4. In the "Style" column, use the selection button to open the "Style" dialog and select the style of the line.

5. Select the number of trend values in the "Trend values" column.
6. In the "Trend type" column, select the required trend type. The following entries are available:
   – "Bit-triggered real time"
   – "Cyclical real time"
   – "Bit-triggered buffer"
   – "Data log"
7. In the "Settings source" column, use the selection button to open the "Data source" dialog and select the tags to supply the trend with values.
   If you selected the "Cyclical real time" trend type, specify a cycle for reading the tags from the PLC.
   Only for bit-triggered trends:
   – Under "Bit", specify which bit of the "Trend request" tag and of the "Trend transfer" tags are to be assigned to the trend. Select a different value for each trend.
   – Enter a tag under "Trend request" which contains at least as many bits as trends are to be displayed in the trend view. The status of the bits of these tags specifies which trend is currently being displayed at the operator device.

   Only for the trend type "Bit-triggered real time":
   – Enter a tag under "Trend transfer" which contains at least as many bits as trends are to be displayed in the trend view. The status of the bits of these tags specifies for which trend values are to be read from the PLC.

   Only for the trend type "Bit-triggered buffer":
   – Enter a buffer tag for the switch buffer. The tags have to be of the same type and of the same length as the trend tag.
   – Enter a tag each under "Trend transfer 1" and "Trend transfer 2" which contains at least as many bits as trends are to be displayed in the trend view. The status of these tag bits specifies the trend for which values will be read from the PLC. "Trend transfer 1" controls reading from the trend buffer; "Trend transfer 2" controls reading from the switch buffer.
8. You can make other settings in the dialogs of the Inspector window. For example, you can select the "Display table" option in the "Table" category to display a value table beneath the trend view.

Note

If you hold down the <CTRL> key and double-click the trend view, the trend view is activated. You can set the column width and the position of the columns in active mode in the values table. In order to activate the trend view the zoom factor has to be set to 100 %.

Result

In runtime, the values of the selected tags are displayed in the configured trend view.

2.10.2.4 Configuring a trend view for a data log (RT Advanced, Panels, Comfort Panels)

Introduction

You can use a trend view to display logged tag values in Runtime. The trend view shows the logged tag values in a particular window in time. To obtain the requested information from the archive, the operator uses the corresponding operator controls in the screen or in the trend view.

Application example

At the end of a shift, for example, an operator wants information about the process carried out during the shift.

Requirement

- A data log has been created.
- A screen is open.
- The Inspector window with the trend view properties is open.

Procedure

To configure a trend view to display values from a data log, follow these steps:

1. Add the "Trend view" object from the toolbox in the "Controls" group to the screen.
2. Select "Properties . Properties > Trend" in the Inspector window and double-click "Add" in the "Name" field. A new trend is created.
3. If required, enter a unique name for the trend in the "Name" field.
4. In the "Style" column, use the selection button to open the "Style" dialog and select the style of the line.
5. In the "Trend type" column, select the trend type "Data log".
6. In the "Settings" column, use the selection button to open the "Data source" dialog; select the data log and the tags to supply the trend with values.
7. You can make other settings in the dialogs of the Inspector window. For example, you can select the "Display table" option in the "Table" category to display a value table beneath the trend view.

   Note

   If you hold down the <CTRL> key and double-click the trend view, the trend view is activated. You can set the column width and the position of the columns in active mode in the values table. In order to activate the trend view the zoom factor has to be set to 100 %.

Result

In runtime, the logged values of the selected tags are displayed in the configured trend view.

See also

Representing path trends in the trend view (Page 686)

2.10.2.5 The structure of a *.CSV file with tags (RT Advanced, Panels, Comfort Panels)

Introduction

In the *.csv (comma-separated value) file format, the "Name" and "Value" table columns of the entry are separated by semicolons. Each table row ends with a line break.

Example of a *.CSV file

The example shows a file with logged tag values:

"VarName";"TimeString";"VarValue";"Validity";"Time_ms"

"Var_107";"01.04.98 11:02:52";66,00;1;35886460322,81

"Var_108";"01.04.98 11:02:55 AM";60.00;1;35886460358.73

"Var_109";"01.04.98 11:02:57 AM";59.00;1;35886460381.22

Structure of a log file in *.csv format

The following values are entered in the individual columns of a log file:

Parameters	Description
VarName	Name of the tag from WinCC
Time string	Time stamp as a STRING, e.g. readable data format
VarValue	Value of the tag
Validity	Validity: 1 = value is valid 0 = an error occurred (e.g. interrupted connection)
Time_ms	Specify a time stamp as a decimal value (see below for conversion). Only needed to display the tag values in a trend.

Conversion of the time stamp decimal value

If the value needs to be processed using a different program, proceed as follows:

1. Divide Time_ms by 1,000,000.
   Example: : 36343476928:1 000 000 = 36343,476928
2. The whole number portion (36344) is the date calculated from 31.12.1899.
   Example: 36343 results in 02.07.1999
   You can now convert the time stamp value to days in Excel by assigning a corresponding format from the "Date" group to the cells, which contain the time stamp.
   Result: 37986 results in 31.12.2003
3. The value after the comma (0,476928) indicates the time:
   - Multiply the value (0.476928) by 24 to obtain the hours (11.446272).
   - Multiply the remainder (0.446272) by 60 results in the minutes (26.77632).
   - Multiply the remainder (0.77632) by 60 results in the seconds (46.5792).

   Result 11:26:46.579
   This conversion is supported by Microsoft Excel, for example.

See also

Representing path trends in the trend view (Page 686)

2.10.2.6 Direct access to the ODBC log database (RT Advanced, Panels, Comfort Panels)

Introduction

The storage location of a log can be a database or a file.

The database is addressed by means of its "Data source name" (DSN). Select the database you want to use in WinCC from the Windows Start menu under "Settings > Control panel > ODBC Data Sources".

To store log data, specify the DSN, rather than a directory name during configuration. With the DSN, you are referencing the database and the storage location.

Application

The entire functional scope of the database is available for additional processing and evaluation of log data.

Principle

The data source establishes the connection to the database. You create the data source on the same computer that contains the Runtime software. Then enter the DSN configured here when you create a log in WinCC.

The ODBC interface allows you to use other programs, such as MS Access or MS SQL Server, to access the database directly.

With the "StartProgram" system function, you can also configure a program call (to MS Access, for example) on the HMI device. This does not interrupt Runtime.

2.10.2.7 Representing path trends in the trend view (RT Advanced, Panels, Comfort Panels)

Introduction

To visualize two-dimensional traverse paths of a machine prior to execution, for example, you can use the f(x) trend view.

Implementation

The implementation is based on a data log, in which the coordinates are saved with an identical time stamp.

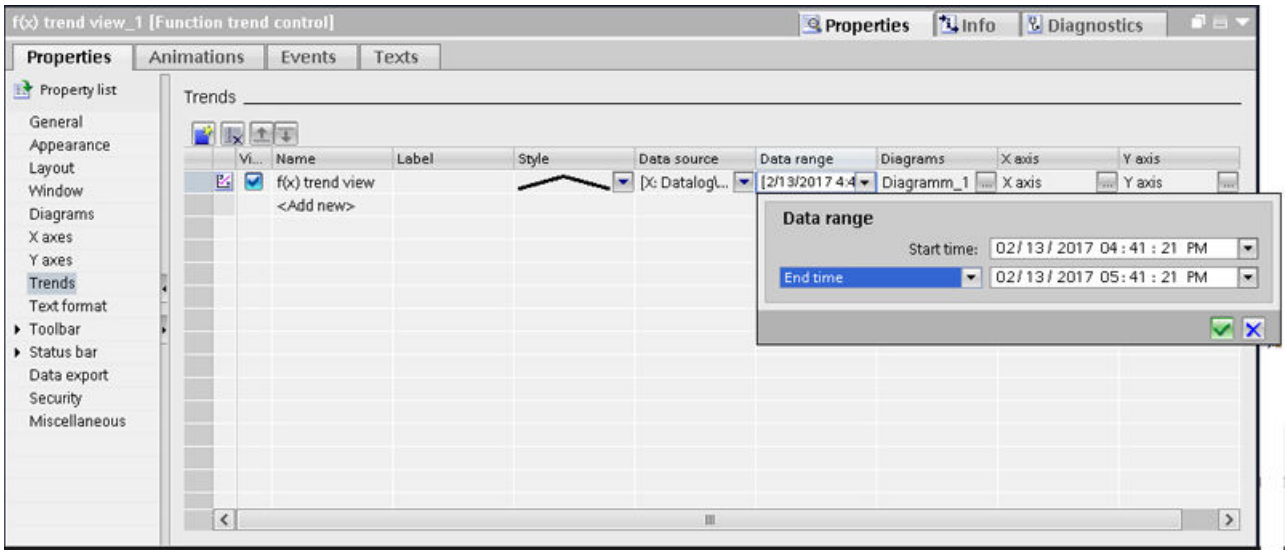

The representation of value pairs in the f(x) trend view is based on a data log that reads the calculated value pairs from a CSV file. The time stamps of each value pair are identical, for example, "01/01/2010 00:00:00".

The data records in the CSV file contain the logging tags with the calculated values. You can calculate the values in the PLC, for example, and store them in two data blocks. Then you can use a VB script, for example, to create the CSV file.

The data source for the trend is the data log.

The following settings are relevant for the configuration of the data area:

- The time range must exactly match the time stamps from the CSV file
- The number of "measuring points" must exactly match the number of data records in the CSV file.

VB script for creating a CSV file

The following VB script generates "Input_X" and "Input_Y" from the values of the HMI tags. The HMI tags have the "Array (0..10) of Real" data type.

```
'##########################################################################
' Location of date log and name of data log which are used in f(x) control
Dim ObjFileName, ObjFilePath

' Name of data log
ObjFileName = "TrendControl_f(x)0.csv"

' File location for data log
ObjFilePath = "\Storage Card SD\"

'##########################################################################
' create objects to edit files and use file system
Dim ObjFile, ObjFileSystem
Set ObjFile = CreateObject("FileCtl.File")
Set ObjFileSystem = CreateObject("FileCtl.Filesystem")

'##########################################################################
' Write header into the new data log
ObjFile.LinePrint(Chr(34)&"VarName"&Chr(34)&";"&Chr(34)&"TimeString"&Chr(34)&";"&Chr(34)&"
VarValue"&Chr(34)&";"&Chr(34)&"Validity"&Chr(34)&";"&Chr(34)&"Time_ms"&Chr(34))
If Err.Number <> 0 Then
    ShowSystemAlarm "Error: Cannot write to data log. " & Err.Number & " " & Err.Description
    Err.Clear
    OpenAllLogs
    TrendControl = 3
    Exit Function
End If
'##########################################################################
' Write entries into data log
' Adapt loop counter to length of archive and array of data tag
Dim ObjLoopCount
ObjLoopCount = 10 ' last element of array

Dim ObjIndex, ObjTime, ObjTimeStamp

For ObjIndex = 0 To ObjLoopCount
    ' Calculate time stamp
    ' 40179000000 = 01.01.2010 00:00:00
    ObjTime = ObjIndex/(60*60*24)*1000
    ObjTimeStamp = 40179000000 + ObjTime
    ' Write x element to log
    Dim Objx
    Objx = SmartTags("Input_X")(ObjIndex)
    ObjFile.LinePrint(Chr(34)&"Temp_x"&Chr(34)&";"&Chr(34)&"01.01.2010
00:00:00"&Chr(34)&";"&Objx&";1;"&ObjTimeStamp)
   ' Write y element to log
    Dim Objy
    Objy = SmartTags("Input_Y")(ObjIndex)
    ObjFile.LinePrint(Chr(34)&"Temp_y"&Chr(34)&";"&Chr(34)&"01.01.2010
00:00:00"&Chr(34)&";"&Objy&";1;"&ObjTimeStamp)
Next
' Write line counter to log
Dim ObjLineCount
ObjLineCount = 41
ObjFile.LinePrint(Chr(34)&"$RT_COUNT$"&Chr(34)&";"&ObjLineCount&";;;;")
```

```
'##########################################################################
' Close new log
ObjFile.close
Set ObjFile = Nothing
Set ObjFileSystem = Nothing
```

See also

The structure of a *.CSV file with tags (Page 684)

Configuring a trend view for a data log (Page 683)

2.10.3 Displaying tags with Runtime Professional (RT Professional)

2.10.3.1 Basics (RT Professional)

Outputting the tag values (RT Professional)

Overview

In Runtime you can output tag values as table or as trend. You can use either process values or logged values as source for the tag values.

Use a table, for example, for comparing tag values. In the table you can, for example, compare fill levels of supply tanks.

Use a trend, for example, for graphically displaying tag values. Using a trend you can, for example, display the development of an engine temperature.

Controls for displaying tag values

To display tag values in a table, use the table view.

To display tag values as a trend, use the trend view. The trend view is available in two versions:

- With the "f(t) trend view" you display a tag value against the time, e.g. a temperature trend.
- With the "f(x) trend view" you display a tag value against a second tag value, e.g. the engine speed against the heat produced.

In addition, the value table is a control with which you can create statistics from the displayed values. Furthermore, you can use the value table as reading assistance for the trend view.

Configuration

In the table and trend view, you configure the following properties:

- Data connection
- Display mode
- Evaluation of data
- Configuration in Runtime
- Persistence of changed properties in Runtime

See also

Table view (Page 690)

Properties of the trend view (Page 691)

f(t) trend view (Page 692)

f(x) trend view (Page 692)

Table view (RT Professional)

Overview

The table view in Runtime shows currently pending process values or logged values in a table. You can arrange the display of the table as you wish.

In Runtime, you create statistics from selected data. You also export the data for further use.

The following image shows a table view in which the fill levels of three supply tanks are shown:

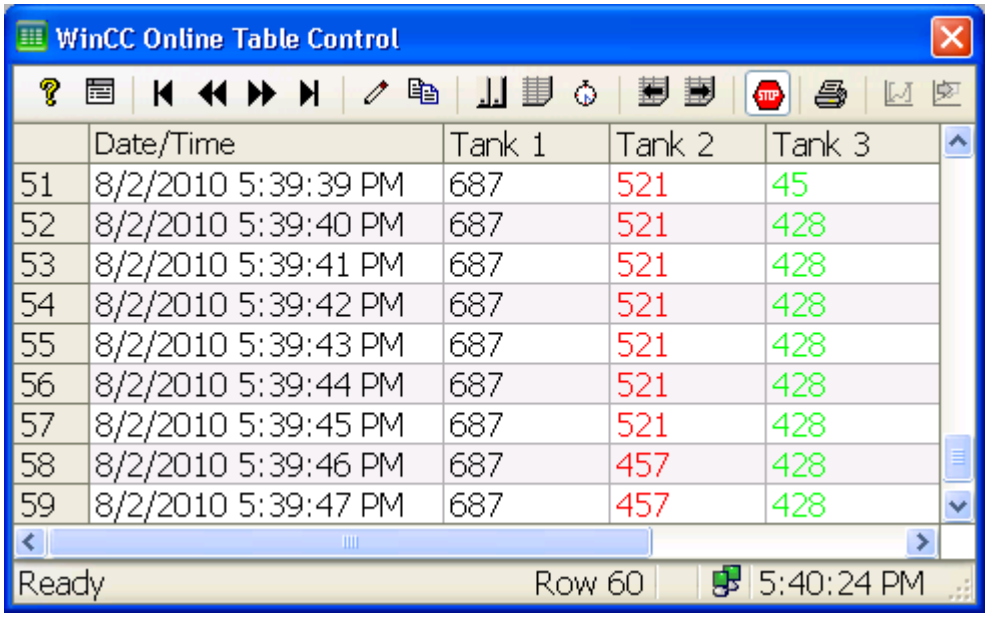

See also

Outputting the tag values (Page 689)

Properties of the trend view (RT Professional)

Number of trends

Both in the "f(x) trend view" and the "f(t) trend view" you can display an unlimited number of trends. For reasons of clarity, you should not, however, configure more than 8 trends at a time.

Resolution of Trend Display

The number of trend values that can be displayed on the screen is limited by the screen resolution and configured size of the trend view. Therefore it is possible that fewer values are displayed in the trend view than actually exist in the period to be represented.

If, for example, in an area of 100 pixels 200 measured values are archived, each pixel represents 2 measured values. The value shown on the screen is that of the most recent data (most recent time stamp).

The figure below shows a trend characteristic where each pixel represents a value pair with 2 measured values:

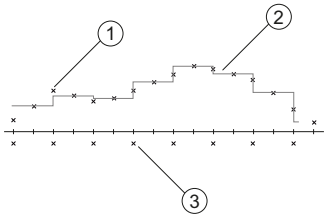

① Real measured values
② Determined trend
③ Pixels

See also

f(t) trend view (Page 692)

f(x) trend view (Page 692)

Outputting the tag values (Page 689)

f(t) trend view (RT Professional)

Overview

With the f(t) trend view you can display currently pending process values or archived values in Runtime as a trend against the time. You can create the trend display according to your wishes.

The following figure shows an f(t) trend view displaying two trends simultaneously:

See also

Properties of the trend view (Page 691)

Outputting the tag values (Page 689)

f(x) trend view (RT Professional)

Overview

With the f(x) trend view you can display currently pending process values or archived values in Runtime as a trend against another tag. You can create the trend display according to your wishes.

The following figure shows an f(x) trend view displaying two trends simultaneously:

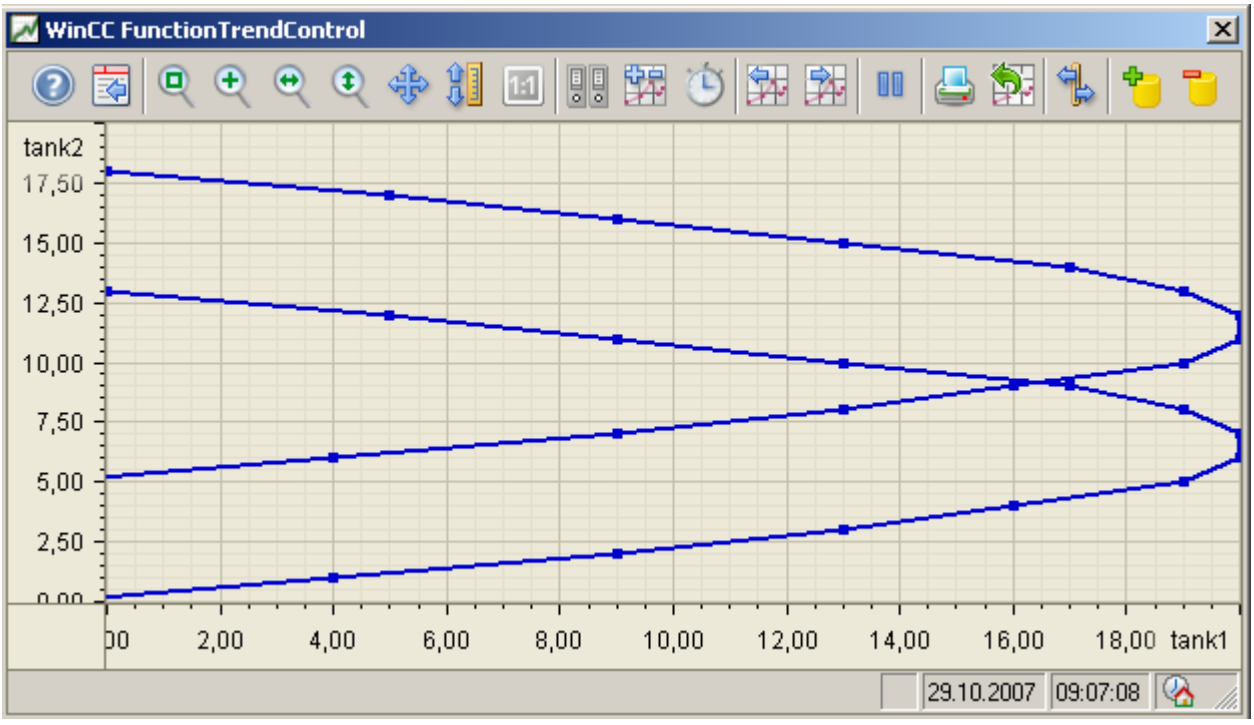

Rules for the data source

The following rules must be observed when displaying trends in the f(x) trend view:

- A trend can represent a maximum of 10000 value pairs
- The following sources are allowed to supply a trend with data:
  - Online tags
  - Logging tags
  - Recipe elements
  - User-defined: Data supply of value pairs using a script
- The value pairs are displayed in the f(x) trend view. For the value pairs to be displayed at the same time, update the online tags or logging tags of a trend at the same time in Runtime. Use the same update cycle for each tag, for example.
- The logging tags of a trend must have the same update cycle and must be recorded in a continuous cycle

See also

Properties of the trend view (Page 691)

Outputting the tag values (Page 689)

Representation Trend Lines (RT Professional)

Introduction

In a trend view, you display a trend curve with one of the following display types:

- Dots
- Interpolated
- Stepped

Select "Properties > Properties > Trend > Style" to configure the properties of the trend display in the Inspector window.

Dots

Values are shown as dots. The display of the points can be configured as you wish.

The following image shows the trend curve with the display form "Dots":

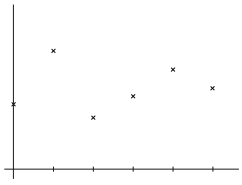

Interpolated

The trend curve is interpolated on a linear basis from the values. The display of the lines and points can be configured as you wish.

The following image shows the trend curve with the display form "Interpolated":

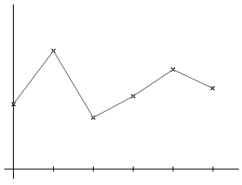

Stepped

The trend curve is interpolated as a stepped curve from the values. The display of the lines and points can be configured as you wish.

The following image shows the trend curve with the display form "Stepped":

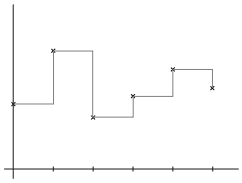

Trend curve

With the "Trend curve" option, you can specify where the current values for all trend windows are to be entered. Normally, the current values are written in the trend window from the right.

You configure the "trend curve" in the Inspector window under "Properties > Properties > General > Display".

See also

Creating trends for the trend window. (Page 703)

Time range basics (RT Professional)

Time range

The time range is the range from which the values at the HMI device are shown. The time range is determined by the start time and the end time. The time range is always in the past. If the end time is later than the current system time, the current system time is used as a temporary end time.

A distinction is made between the following time ranges:

- Static time range
- Dynamic time range

Static time range

The static time range is determined by fixed start and end times. The values are displayed within this time range.

Dynamic time range

The dynamic time range is determined by a period of time beginning with a fixed start time. The end time thus corresponds to the conclusion of the time period.

You set the time period as follows:

- Duration, e.g. 30 minutes
- The number of measurement points multiplied by the update cycle also produces a duration.

Configuring time range

Configure the time range for all controls. Configure the time range in the time column or in the time axis for the table view and the f(t) trend view. Configure the time range directly at the trend for the f(x) table view.

See also

Configuring the time axes of the the trend view (Page 700)

Displaying logged values (Page 750)

Representation Using Common Axes (RT Professional)

Introduction

If you display several trends simultaneously in the trend view, assign each trend its own value and time axis. Alternatively, you can assign a shared time and/or value axis to several trends.

You configure the axes of a trend view in the Inspector window under "Properties > Properties > Time axis / Value axis".

The axes are assigned to the configured trends in the Inspector window under "Properties > Properties > Trends".

Representation Using Different Axes

If the values to be displayed in a trend view differ greatly, a common value axis makes no sense. If you assign each trend its own value axis, they should also display different scales. Individual axes can be hidden if required.

The following figure shows two trends with different value axes using an f(t) trend display as example:

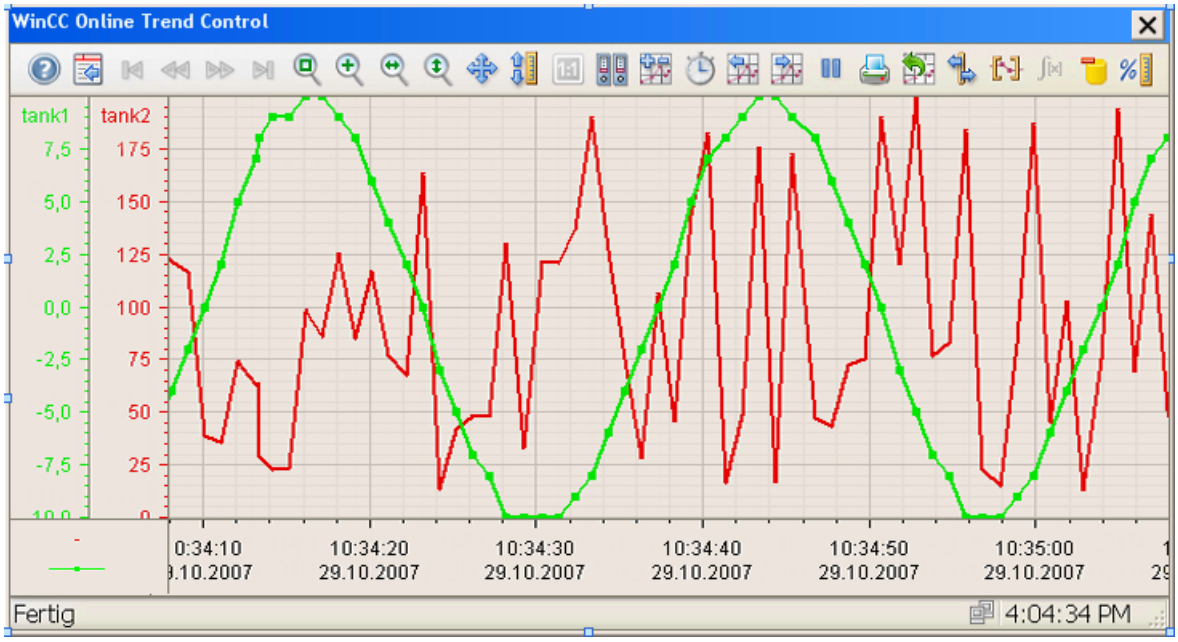

Representation Using Common Axes

If the comparability of the trend lines is important, common axes in a trend view is sensible. Connected trend views can have a common time axis.

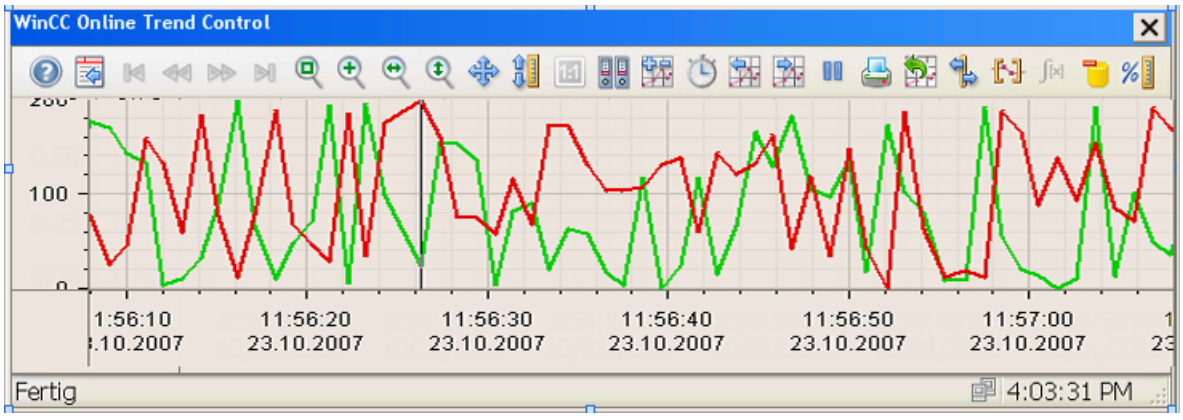

If you configure trends with a shared time axis, use tags with the same update cycle for the data supply.

In the case of different updating cycles, the length of the time axis is not identical for all tags. Since the trends are updated at different times due to the different updating cycles, a minimal different in the end time for the time axis occurs on each change. As a result, the trends displayed skip slightly to and fro on each change.

Making buttons in the toolbar dynamic (RT Professional)

Introduction

If you do not wish to operate the control via the toolbar, configure any operator control such as a script on an event. Each control button has a unique ID.

If you want to use a script to trigger a control button, assign the idea of the button to the "ToolbarButtonClick" property.

If you wish to determine which button of a control has been pressed, configure a VB script or a C script at the event "Click toolbar button". The ID of the button is passed on to the script as the parameter "toolbarbuttonID".

Example: Opening the configuration dialog for the control using VBS

To open the "Control_1" configuration dialog of the control using VBS, you have the following options:

- Call in a script of an operator control event, e.g. a button:
  `ScreenItems("Control_1").ToolbarButtonClick = 2`
- As an alternative to the property "ToolbarButtonClick", there are also methods in VBS for operating the toolbar:
  ```
  Dim obj
  Set obj = ScreenItems("Control_1")
  obj.ShowPropertyDialog
  ```

Example: Opening the configuration dialog for the control using C script

To open the "Control_1" configuration dialog of the control in a C script, you have the following options:

- `SetPropertyByConstant ("Screen_1", "Control_1", "ToolbarButtonClick", "2");`

See also

Using the trend and table view in Runtime (Page 731)

2.10.3.2 Output of tag values in the form of a trend (RT Professional)

Configure trend view (RT Professional)

Introduction

To display tag values graphically in Runtime, add a trend view to a screen. In the "Tools" task card, two trend views are contained under "Controls":

- f(x) trend view
- f(t) trend view

The axis designations are different for the two trend views:

- The f(x) trend view has an "X axis" and a "Y axis"
- The f(t) trend view has a "time axis" and a "value axis"

Requirement

- The screen is opened
- The Inspector window is open

Procedure

To configure a trend view, follow these steps:

1. Add the required trend view to the screen from the "Tools" task card.
2. Configure the appearance of the trend view in the Inspector window:
   – Configure the basic properties of the trend view, e.g. "Time base" or "Trend characteristic", under "Properties > Properties > General".
   – Configure the appearance of the trend view under "Properties > Properties > Appearance".
   – Configure the properties of the trend view in Runtime under "Properties > Properties > Window".
3. Configure additional diagrams, if required.
4. Configure the axes of the trend view in the Inspector window.
   – Configure the properties of the axes.
   – Assign each axis to a diagram of the trend view.
5. Enter the trends in the Inspector window that are shown in the trend view:
   – Define the "data source" for each trend.
   – Assign each trend the "diagram" in which it will be displayed.
   – Assign the axes to each trend.
   – Configure the view of each trend.

6. Configure the toolbar and status bar of the trend view.
7. If required, configure the data export from the trend view.
8. If required, configure the security settings of the trend view.

Result

The trend view is configured. Define an additional value display if you want to evaluate the data of the trend view in Runtime. You can also configure the value display as a "Ruler".

See also

Configuring the time axes of the the trend view (Page 700)

Configuring the value axis of the the trend view (Page 701)

Creating diagrams in the trend view (Page 702)

Creating trends for the trend window. (Page 703)

Defining the data source (Page 704)

Configuring toolbar and status bar (Page 706)

Configuring the export of Runtime data (Page 707)

Determining the effect of the online configuration (Page 708)

Online configuration of the f(t) trend view (Page 737)

Configuring the time axes of the the trend view (RT Professional)

Introduction

The time range for trend display is configured with time axes. In a trend view you can create several time axes, which you can assign to one or more diagrams.

If you assign several time axes to a trend view, the sequence of the time axes in the Inspector window corresponds to the sequence in the trend view. If several time axes are aligned to the same side of a trend display, the first time axis in the list takes the lower left position. The last time axis takes the upper right position.

The time axis is available only with the f(t) trend display.

Requirement

- The trend view is selected in the screen
- The Inspector window is open

Procedure

To configure the time axis of a trend view, follow these steps:

1. Double-click "Add" in the Inspector window under "Properties > Properties > Time axis" to add a new time axis.
2. Enter the "Name" and "Display name" of the time axis.
3. Configure the "axis style".
4. Select the "diagram" to which the time axis will be assigned.
5. Configure the "time range" and "format" of the time display of the time axis.
6. To always update the trends assigned to the time axis in the trend view, activate "Online".

   Note

   Printing a fixed time range

   If you want to print a defined time range, deactivate "Online".

   If you want to compare a current trend display with an earlier trend display, for example, supply the comparison trend from a logging tag.

Result

The time axis is configured. Assign the time axis to one or several trends.

See also

Time range basics (Page 695)

Structure of the time format specification (Page 709)

Structure of the date format specification (Page 710)

Configuring the value axis of the the trend view (RT Professional)

Introduction

In a trend view you can create several value axes, which you can assign to one or more diagrams.

The following properties are set by default with a value axis:

- The value range is based on the current values of the assigned trend
- The value axis scale is linear to the value range

Alternatively, you can configure a logarithmic scale:

- In logarithmic scaling, only positive values are displayed.
- In negative logarithmic scaling, only negative values are displayed.

If you configure a value axis for the f(t) trend view, you can also set up axis sections. A value range and a display name are assigned to an axis section.

In the f(x) trend view, the value axis corresponds to both the "X axis" and the "Y axis".

Requirement

- The trend view is selected in the screen
- The Inspector window is open

Procedure

To configure the value axis of a trend view, follow these steps:

1. Double-click "Add" in the Inspector window under "Properties > Properties > Value axis" to add a new value axis.
2. Enter the "Name" and "Display name" of the value axis.
3. Configure the "axis style".
4. Select the "diagram" to which the value axis will be assigned.
5. If required, configure the "value range" of the value axis.
6. If required, configure the "format" of the value axis labels.
7. If required, configure the "scaling" of the value axis.
8. If you configure an f(t) trend view, you can define "axis sections" if necessary:
   - Activate "User-defined axis sections".
   - To add an axis section, double-click on "Add".
   - Enter the "value range" of the axis section.
   - Configure the "display range" of the axis section.
   - Enter the "display name" of the axis section.

Result

The value axis is configured. Assign the value axis to one or several trends.

Creating diagrams in the trend view (RT Professional)

Introduction

A trend view is assigned a display range as standard, in which you can configure one or more trends.

Alternatively, you can subdivide the display range into several "diagrams". Each "diagram" functions like a standalone trend view. In this way, for example, temperature changes can be displayed clearly without overlapping trends. You can set the height of a diagram using "Range proportion". The "range proportion" establishes how much room is made available to a diagram in the trend view.

Calculation of range proportion

The amount of each range proportion can be calculated from the total number of range components. If, for example, you have configured a total of three diagrams, a range proportion of "1" each will result in three diagrams of equal size. To increase the size of range components in relation to each other, increase the range proportion of one or more diagrams. Changes to the range proportions immediately affect the trend view.

Requirement

- Trend view is selected
- The Inspector window is open

Procedure

Proceed as follows to subdivide a trend view into several diagrams:

1. Double-click on "Add" under "Properties > Properties > Diagrams" in the Inspector window. The diagram is added.
2. Enter a meaningful "Name" for the diagram.
3. Select the "range proportion" for the diagram in the trend view.
4. Configure the diagram "grid".
5. Configure the diagram "ruler".

Result

The diagram has been created. You can assign one or more trends to each diagram.

Creating trends for the trend window. (RT Professional)

Introduction

A trend allows you to represent values in the trend view. You assign a time axis and value axis to each trend. The value axis assigned to the trend determines the trend view in which the trend will be displayed.

By default, the data area is based on the current values of the assigned trend.

You can also configure the visualization of limits and values with "uncertain status" for a trend. If a trend exceeds or falls below a configured limit, the trend is colored.

If you assign several trends to a trend view, the sequence of the trends in the Inspector window corresponds to the sequence in the trend view.

Requirement

- The trend view is selected in the screen
- Time axes and value axes are configured

- Tags for the data source are configured
- The Inspector window is open

Procedure

To add a trend, follow these steps:

1. Double-click on "Add" under "Properties > Properties > Trends" in the Inspector window.
2. Enter a "label" for the trend.
3. If required, configure the "style" of the trend.
   The changes are shown in a preview.
4. Configure the data source.
5. If required, configure the "data area" of the trend.
6. Select the "diagram" in which the trend will be displayed.
7. Assign a "value axis" and "time axis" to the trend.
8. If required, configure the "limits" of the trend.

Result

The trend is configured.

See also

Representation Trend Lines (Page 694)

Defining the data source (Page 704)

Defining the data source (RT Professional)

Introduction

Using the data source, you define the sources from which the values are displayed on the HMI device in Runtime. The following sources are available:

- Current process values from tags
- Archived values from logging tags
- Recipe elements (only f(x) trend view)
- Data supply by means of script in Runtime

Requirements

- An online tag or logging tag is configured
- Value column or trend has been created
- The Inspector window is open

Displaying current process values

Proceed as follows to display current process values:

1. Click "Properties > Properties > Value columns" in the Inspector window to define the data source for a table view.
2. Click "Properties > Properties > Trends" in the Inspector window to define the data source for a trend view.
3. Configure the "data source":
   - Select the entry "Tags" as "Source".
   - If you configure the trend of an f(x) trend view, select one tag each from "X values" and "Y values".
   - If you configure the trend of an f(t) trend view or a value column, select a "tag".
   - Select the "update cycle".

Displaying values from a log

Proceed as follows to display values from a log:

1. Click "Properties > Properties > Value columns" in the Inspector window to define the data source for a table view.
2. Click "Properties > Properties > Trends" in the Inspector window to define the data source for a trend view.
3. Configure the "data source":
   - Select the entry "Logging tags" as "Source".
   - If you configure the trend of an f(x) trend view, select one tag each from "X values" and "Y values".
   - If you configure the trend of an f(t) trend view or a value column, select a "tag".

Displaying values from recipes

Output values from recipes in a f(x) trend view.

Proceed as follows to display values from a recipe:

1. In the Inspector window, configure the "data source" under "Properties > Properties > Trends":
   - Select the entry "Recipe" as "Source".
   - Under "X values" and "Y values" select one recipe element each.

Setting up user-defined data supply in Runtime

If you want to supply the trend values with data by means of scripts in Runtime, you require a script via the API interface.

To create a connection using scripts in Runtime, proceed as follows:

1. In the Inspector window, configure the "data source" under "Properties > Properties > Trends":
   – Select the entry "User-defined" as "Source".
   – Connect the script via the API interface to the corresponding trend.

Result

The data supply is configured.

See also

Basics of tags (Page 519)

Basic principles for data logging (Page 616)

Configure trend view (Page 699)

Creating trends for the trend window. (Page 703)

Configuring toolbar and status bar (RT Professional)

Introduction

You operate the controls in Runtime using the buttons in the toolbar. The status bar displays status messages from the control. During configuration, set the content of the toolbar and status bar.

Requirement

- The table view is selected in the screen.
- The Inspector window is open

Configuring the toolbar

Proceed as follows to configure the toolbar:

1. In the Inspector window, configure the general properties of the toolbar, such as orientation and background color, under "Properties > Properties > Toolbar > Toolbar - General".
2. In the Inspector window, activate the buttons you need in Runtime under "Properties > Properties > Toolbar > Toolbar - Buttons".
   For information on the functions of the individual buttons, refer to the description of the corresponding WinCC control at "Operation in Runtime".
3. To set the order of the buttons, select the buttons in the list. You can then move the buttons to the desired position.
4. If needed, you can define a "hotkey" for the buttons on the toolbar.

5. If necessary, select the authorization needed to operate the buttons in Runtime.
6. If a button should not be operated in Runtime, unselect "Activate".
   You can reactivate a deactivated a button using a script in Runtime, for example.

Configuring the status bar

Proceed as follows to configure the status bar:

1. In the Inspector window, configure the general properties of the status bar, such as orientation and background color, under "Properties > Properties > Status bar > Status bar - General".
2. In the Inspector window, activate the elements you need in Runtime under "Properties > Properties > Status bar > Status bar - Elements".
   For further information on status bar elements, refer to the descriptions of the corresponding WinCC control at "Operation in runtime".
3. To set the order of the elements, select the element in the list. You can then move the element to the desired position.
4. To adjust the width of an element in the status bar, deactivate "Auto-size". You can then enter the width in pixels.

See also

Using the trend and table view in Runtime (Page 731)

Configure trend view (Page 699)

Elements of the status bar (Page 751)

Configuring the export of Runtime data (RT Professional)

Introduction

In Runtime, you can export the displayed data of a control in CSV format. Depending on the control used and the configuration, the user exports all data or only the selected data.

The filename and location of the CSV file are set by the operator on export. Alternatively, you can generate the filename automatically from the following wildcards:

- @OBJECTNAME: Object name of the control
- @CURRENTDATE: Current date
- @CURRENTTIME: Current time

Requirement

- The control is selected in the screen
- The Inspector window is open

Procedure

To configure the export of Runtime data, proceed as follows:

1. In the Inspector window, open "Properties > Properties > Data export".
   The data export settings are already configured.
2. If necessary, enter your own filename and a path for the export file.
3. Select the "scope" of the data export.
4. Configure the "Operation in Runtime".
5. So that the operator can start the data export, activate "Properties > Properties > Toolbar > Toolbar - Buttons > Export Data" in the Inspector window.

Result

In Runtime, the operator can export the data of the control via the toolbar.

See also

Configure trend view (Page 699)

Determining the effect of the online configuration (RT Professional)

Introduction

In WinCC you configure the appearance and function of a control in Runtime. In addition, you can allow the operator to configure a control in Runtime. You can set whether changes to configuration in Runtime are retained, such as after a screen change, in the control under "Properties > Properties > Security > Persistence".

- No persistence
  The changes to the configuration are not saved. The changes are discarded on the next screen change and when the project is rejected.
- Persistence
  The changes to the configuration are saved. The changes are retained even after the project is deactivated and activated. You may also select "Response at screen change" to discard changes at the screen change.
- Persistence in Runtime
  The changes to the configuration are only saved during Runtime. If you deactivate the project, the changes are discarded. You may also select "Response at screen change" to discard changes at the screen change.

Note

If you change a configured control and load it onto the HMI device, the changes are overwritten in Runtime regardless of the settings.

Requirement

- A configured control
- The Inspector window is open

Procedure

To configure the persistence of data in Runtime, proceed as follows:

1. In the Inspector window, open "Properties > Properties > Security".
2. Select "Online configuration".
3. If needed, select the authorization.
4. Select the "Behavior on screen change".

Result

The behavior in online configuration is now set. In Runtime, depending on the settings, the user can adjust the configuration of the control.

If several operators are working on the HMI device, the individual configurations are accepted on a change of operator only following a screen change.

See also

Configure trend view (Page 699)

Online configuration of the f(t) trend view (Page 737)

Structure of the time format specification (RT Professional)

Time format

You can overwrite the default time formats of the selection list.

If you want to define your own time formats, note the following:

- Spaces that separate values in the format specification are set at the same position in the time display.
- Note that the entry is case-sensitive.
- Characters that are set in single quotation marks are set unchanged at the same position in the time display.

To set up a format specification, use the following values:

Value	Description
h	Hours, 12-hour clock, for single-digit hours without leading zero
hh	Hours, 12-hour clock, for single-digit hours with leading zero
H	Hours, 24-hour clock, for single-digit hours without leading zero
HH	Hours, 24-hour clock, for single-digit hours with leading zero

Value	Description
m	Minutes, without leading zero for single-digit minutes
mm	Minutes, with leading zero for single-digit minutes
s	Seconds, without leading zero for single-digit seconds
ss	Seconds, with leading zero for single-digit seconds
t	Single-digit AM/PM time identifier: A or P
tt	Two-digit AM/PM time identifier: AM or PM

Example

To set up the time display "11:29:40 PM", use the following format specification:

"hh:mm:ss tt"

Structure of the date format specification (RT Professional)

Time format

The default date formats of the selection list can be overwritten.

When you specify your own date formats, note the following:

- Spaces that separate values in the format specification are set in the same position in the date display.
- Note that the entry is case-sensitive.
- Characters that are set in single quotation marks are set unchanged at the same position in the date display.

To set up a format specification, use the following values:

Value	Description
D	Day of the month as number, for single-digit days without leading zero
Dd	Day of the month as number, for single-digit days with leading zero
Ddd	Weekday shortened with three letters
Dddd	Weekday in full form
M	Month as number, for single-digit month without leading zero
MM	Month as number, for single-digit month with leading zero
MMM	Month shortened with three letters
MMMM	Month in full form
y	Year, only the last two numbers, for years under 10 with leading zero, same as "YY" format
yy	Year, only the last two numbers, for years under 10 with leading zero
yyyy	Complete, four-digit year specification

Example

Use the following format specification to set up the date display "Wed, Aug 31 13":

"Ddd, MMM dd yy"

2.10.3.3 Output of tag values in tabular format (RT Professional)

Configuring the Table View (RT Professional)

Introduction

To display tag values in tables in Runtime, add a table view to a screen. A time stamp is displayed for each value. The values are displayed in value columns, and the time stamps in time columns. Assign the time column to one or several value columns.

Procedure

Proceed as follows to configure a table view:

1. Add the required table view to the screen from the "Tools" task card.
2. Configure the appearance of the table view in the Inspector window:
   - Configure the basic properties of the table view, e.g. "Time base", under "Properties > Properties > General".
   - Configure the appearance of the table view under "Properties > Properties > Appearance".
   - Configure the properties of the table view in Runtime under "Properties > Properties > "Window".
3. Configure the columns of the table view in the Inspector window.
   - Configure the properties of the columns.
   - Configure one or more time columns with the time ranges for the table.
   - Configure one or more value columns.
   - Define the data connection for each value column.
   - Assign time columns to the value columns.
4. Configure the appearance of each trend:
   - Configure the "Headers", the "Structure" and the "Grid".
   - If needed, configure the settings for "selecting" table entries.
   - If needed, configure the settings for sorting table entries.
5. Configure the toolbar and status bar of the table view.
6. If required, configure the data export from the table view.
7. If required, configure the security settings of the table view.

Result

The table view is configured. Define an additional value table if you want to evaluate the data of the table view in Runtime. You can also configure the value table as "Ruler".

See also

Configuring the time columns of the table view (Page 712)
Configuring the value columns of the table view (Page 713)
Configuring the appearance of the table elements (Page 714)
Configuration of the marking of selected cells and rows (Page 715)
Configuring table column sorting (Page 716)
Making buttons in the toolbar dynamic (Page 698)
Defining the data source (Page 717)
Configuring toolbar and status bar (Page 719)
Configuring the export of Runtime data (Page 720)
Determining the effect of the online configuration (Page 721)
Online configuration of the table view (Page 744)

Configuring the time columns of the table view (RT Professional)

Introduction

The time range for the tabular display is configured via time columns. A table can have separate time columns for several value columns or a common time column.

If you assign several time columns to a table view, the sequence of the time columns in the Inspector window corresponds to the sequence in the table view.

Requirement

- The table view is selected in the screen.
- The Inspector window is open

Procedure

To configure the time column of a table view, follow these steps:

1. Click "Add" in the Inspector window under "Properties > Properties > Time columns" to add time columns.
2. Enter the "Name" and "Label" of the time column.
3. Configure the "Style" of the time column.
4. Configure the "time range" and "format" of the time display of the time column.

5. Configure the appearance of the "Header" and "Cells".
6. If the values of the time column should be updated automatically, activate "Update".

Result

The time column is configured. Assign the time column to one or several value columns.

See also

Configuring the value columns of the table view (Page 713)

Structure of the time format specification (Page 722)

Structure of the date format specification (Page 722)

Configuring the value columns of the table view (RT Professional)

Introduction

You configure the values of the table view via value columns. You can display several value columns in a table, e.g. to compare the fill levels of several containers. Every value column is connected with a time column. The value columns can have a common time column.

If you assign several value columns to a table view, the sequence of the value columns in the Inspector window corresponds to the sequence in the table view. If you assign several value columns to the same time column, the value columns in the list are automatically grouped according to assigned time column.

Requirement

- The table view is selected in the screen.
- The Inspector window is open
- The time column is created

Procedure

Proceed as follows to configure a value column:

1. Double-click "Add" in the Inspector window under "Properties > Properties > Value columns" to add a new value column.
2. Enter the "Name" and "Label" of the value column.
3. Configure the data source.
4. Assign a time column to the value column.
5. If required, configure the appearance of the "Header" and "Cells".
6. If required, configure the "Sorting" of the value column.

Result

The value column is configured.

See also

Defining the data source (Page 704)
Configuring the time columns of the table view (Page 712)
Defining the data source (Page 717)

Configuring the appearance of the table elements (RT Professional)

Introduction

You can adjust the appearance of the table elements to your needs.

Requirement

- The table view or value table is selected in the screen.
- The Inspector window is open

Procedure

Proceed as follows to configure the colors of the table elements:

1. Define the following properties in the Inspector window under "Properties > Properties > Table > Table - Header":
   - Visibility of the column headers and row headers
   - Alignment of the column headers and row headers
   - Color design of column headers
2. Define the following properties in the Inspector window under "Properties > Properties > Table > Table - Structure":
   - Color design of table content, such as different colors for rows for better differentiation
   - Visibility of empty columns and rows
3. Define the following properties in the Inspector window under "Properties > Properties > Table > Table - Grid":
   - Design of grid lines
   - Character spacing in the table cells

Result

The appearance of the table elements is configured. You can also define further properties:

- Behavior and color design when selecting a table entry
- Table entry sorting

See also

Configuration of the marking of selected cells and rows (Page 715)

Configuring table column sorting (Page 716)

Configuration of the marking of selected cells and rows (RT Professional)

Introduction

If, for example, an operator wants to export values from a row, he must select the row. During configuration, define the following:

- Selection area
  For the selection area, configure the following properties:
  - Number of table elements that the operator can select
  - Design of selection area
- Color design of selection area
  Select the colors individually or use a default system-defined color.

Requirement

- The table view is selected in the screen.
- The Inspector window is open

Procedure

To configure the selection area, follow these steps:

1. In the Inspector window, select the "type" of selection area under "Properties > Properties > Table > Table - Selection > Selection area".
2. If needed, select the "frame" to be displayed around the selection area.
3. Choose the colors for the "cells", "rows", and "frame" as needed.

Result

The selection area of the table view is configured. To export the data from a selected area, configure the "data export". To evaluate the data from a selected area, configure a "value display".

See also

Defining the data source (Page 717)

Configuring table column sorting (RT Professional)

Introduction

If the operator is to be able to sort individual table columns in Runtime, configure the sorting within the table.

Depending on the configuration, the operator can sort a table column as follows:

- Clicking or double-clicking on the column header
- Clicking on a sorting button
  The sorting button is used to sort the selected column in the configured order. For the sorting button to appear, you must configure a vertical scroll bar.

Requirement

- The table view or value table is selected in the screen.
- The Inspector window is open

Procedure

To configure the sorting within a table, follow these steps:

1. In the Inspector window, select "Properties > Properties > Table > Table - Sorting > Sort by column header", "Sort with double-click", or "Sort with click".
2. Select the "Sorting order" option.
3. If needed, activate "Display sorting icon" and "Display sorting index".
4. If needed, activate "Display sorting button".
5. If you use the sorting button, select "Properties > Properties > Window > Scroll > Vertical scrollbar > Always" in the Inspector window.

Result

The sorting of the table columns is configured.

Defining the data source (RT Professional)

Introduction

Using the data source, you define the sources from which the values are displayed on the HMI device in Runtime. The following sources are available:

- Current process values from tags
- Archived values from logging tags
- Recipe elements (only f(x) trend view)
- Data supply by means of script in Runtime

Requirements

- An online tag or logging tag is configured
- Value column or trend has been created
- The Inspector window is open

Displaying current process values

Proceed as follows to display current process values:

1. Click "Properties > Properties > Value columns" in the Inspector window to define the data source for a table view.
2. Click "Properties > Properties > Trends" in the Inspector window to define the data source for a trend view.
3. Configure the "data source":
   - Select the entry "Tags" as "Source".
   - If you configure the trend of an f(x) trend view, select one tag each from "X values" and "Y values".
   - If you configure the trend of an f(t) trend view or a value column, select a "tag".
   - Select the "update cycle".

Displaying values from a log

Proceed as follows to display values from a log:

1. Click "Properties > Properties > Value columns" in the Inspector window to define the data source for a table view.
2. Click "Properties > Properties > Trends" in the Inspector window to define the data source for a trend view.
3. Configure the "data source":
   - Select the entry "Logging tags" as "Source".
   - If you configure the trend of an f(x) trend view, select one tag each from "X values" and "Y values".
   - If you configure the trend of an f(t) trend view or a value column, select a "tag".

Displaying values from recipes

Output values from recipes in a f(x) trend view.

Proceed as follows to display values from a recipe:

1. In the Inspector window, configure the "data source" under "Properties > Properties > Trends":
   - Select the entry "Recipe" as "Source".
   - Under "X values" and "Y values" select one recipe element each.

Setting up user-defined data supply in Runtime

If you want to supply the trend values with data by means of scripts in Runtime, you require a script via the API interface.

To create a connection using scripts in Runtime, proceed as follows:

1. In the Inspector window, configure the "data source" under "Properties > Properties > Trends":
   - Select the entry "User-defined" as "Source".
   - Connect the script via the API interface to the corresponding trend.

Result

The data supply is configured.

See also

Basics of tags (Page 519)

Basic principles for data logging (Page 616)

Configuring the Table View (Page 711)

Configuring the value columns of the table view (Page 713)

Configuring toolbar and status bar (RT Professional)

Introduction

You operate the controls in Runtime using the buttons in the toolbar. The status bar displays status messages from the control. During configuration, set the content of the toolbar and status bar.

Requirement

- The table view is selected in the screen.
- The Inspector window is open

Configuring the toolbar

Proceed as follows to configure the toolbar:

1. In the Inspector window, configure the general properties of the toolbar, such as orientation and background color, under "Properties > Properties > Toolbar > Toolbar - General".
2. In the Inspector window, activate the buttons you need in Runtime under "Properties > Properties > Toolbar > Toolbar - Buttons".
   For information on the functions of the individual buttons, refer to the description of the corresponding WinCC control at "Operation in Runtime".
3. To set the order of the buttons, select the buttons in the list. You can then move the buttons to the desired position.
4. If needed, you can define a "hotkey" for the buttons on the toolbar.
5. If necessary, select the authorization needed to operate the buttons in Runtime.
6. If a button should not be operated in Runtime, unselect "Activate".
   You can reactivate a deactivated a button using a script in Runtime, for example.

Configuring the status bar

Proceed as follows to configure the status bar:

1. In the Inspector window, configure the general properties of the status bar, such as orientation and background color, under "Properties > Properties > Status bar > Status bar - General".
2. In the Inspector window, activate the elements you need in Runtime under "Properties > Properties > Status bar > Status bar - Elements".
   For further information on status bar elements, refer to the descriptions of the corresponding WinCC control at "Operation in runtime".
3. To set the order of the elements, select the element in the list. You can then move the element to the desired position.
4. To adjust the width of an element in the status bar, deactivate "Auto-size". You can then enter the width in pixels.

See also

Using the trend and table view in Runtime (Page 731)

Configuring the Table View (Page 711)

Elements of the status bar (Page 751)

Configuring the export of Runtime data (RT Professional)

Introduction

In Runtime, you can export the displayed data of a control in CSV format. Depending on the control used and the configuration, the user exports all data or only the selected data.

The filename and location of the CSV file are set by the operator on export. Alternatively, you can generate the filename automatically from the following wildcards:

- @OBJECTNAME: Object name of the control
- @CURRENTDATE: Current date
- @CURRENTTIME: Current time

Requirement

- The control is selected in the screen
- The Inspector window is open

Procedure

To configure the export of Runtime data, proceed as follows:

1. In the Inspector window, open "Properties > Properties > Data export".
   The data export settings are already configured.
2. If necessary, enter your own filename and a path for the export file.
3. Select the "scope" of the data export.
4. Configure the "Operation in Runtime".
5. So that the operator can start the data export, activate "Properties > Properties > Toolbar > Toolbar - Buttons > Export Data" in the Inspector window.

Result

In Runtime, the operator can export the data of the control via the toolbar.

See also

Configuring the Table View (Page 711)

Determining the effect of the online configuration (RT Professional)

Introduction

In WinCC you configure the appearance and function of a control in Runtime. In addition, you can allow the operator to configure a control in Runtime. You can set whether changes to configuration in Runtime are retained, such as after a screen change, in the control under "Properties > Properties > Security > Persistence".

- No persistence
  The changes to the configuration are not saved. The changes are discarded on the next screen change and when the project is rejected.
- Persistence
  The changes to the configuration are saved. The changes are retained even after the project is deactivated and activated. You may also select "Response at screen change" to discard changes at the screen change.
- Persistence in Runtime
  The changes to the configuration are only saved during Runtime. If you deactivate the project, the changes are discarded. You may also select "Response at screen change" to discard changes at the screen change.

Note

If you change a configured control and load it onto the HMI device, the changes are overwritten in Runtime regardless of the settings.

Requirement

- A configured control
- The Inspector window is open

Procedure

To configure the persistence of data in Runtime, proceed as follows:

1. In the Inspector window, open "Properties > Properties > Security".
2. Select "Online configuration".
3. If needed, select the authorization.
4. Select the "Behavior on screen change".

Result

The behavior in online configuration is now set. In Runtime, depending on the settings, the user can adjust the configuration of the control.

If several operators are working on the HMI device, the individual configurations are accepted on a change of operator only following a screen change.

See also

Configuring the Table View (Page 711)

Online configuration of the table view (Page 744)

Structure of the time format specification (RT Professional)

Time format

You can overwrite the default time formats of the selection list.

If you want to define your own time formats, note the following:

- Spaces that separate values in the format specification are set at the same position in the time display.
- Note that the entry is case-sensitive.
- Characters that are set in single quotation marks are set unchanged at the same position in the time display.

To set up a format specification, use the following values:

Value	Description
h	Hours, 12-hour clock, for single-digit hours without leading zero
hh	Hours, 12-hour clock, for single-digit hours with leading zero
H	Hours, 24-hour clock, for single-digit hours without leading zero
HH	Hours, 24-hour clock, for single-digit hours with leading zero
m	Minutes, without leading zero for single-digit minutes
mm	Minutes, with leading zero for single-digit minutes
s	Seconds, without leading zero for single-digit seconds
ss	Seconds, with leading zero for single-digit seconds
t	Single-digit AM/PM time identifier: A or P
tt	Two-digit AM/PM time identifier: AM or PM

Example

To set up the time display "11:29:40 PM", use the following format specification:

"hh:mm:ss tt"

Structure of the date format specification (RT Professional)

Time format

The default date formats of the selection list can be overwritten.

When you specify your own date formats, note the following:

- Spaces that separate values in the format specification are set in the same position in the date display.
- Note that the entry is case-sensitive.
- Characters that are set in single quotation marks are set unchanged at the same position in the date display.

To set up a format specification, use the following values:

Value	Description
D	Day of the month as number, for single-digit days without leading zero
Dd	Day of the month as number, for single-digit days with leading zero
Ddd	Weekday shortened with three letters
Dddd	Weekday in full form
M	Month as number, for single-digit month without leading zero
MM	Month as number, for single-digit month with leading zero
MMM	Month shortened with three letters
MMMM	Month in full form
y	Year, only the last two numbers, for years under 10 with leading zero, same as "YY" format
yy	Year, only the last two numbers, for years under 10 with leading zero
yyyy	Complete, four-digit year specification

Example

Use the following format specification to set up the date display "Wed, Aug 31 13":

"Ddd, MMM dd yy"

2.10.3.4 Configuring the value table (RT Professional)

Value table basics (RT Professional)

Function

The value table displays values or statistics from a control. You establish the content of the value table when you configure it.

Overview of the value table

You connect the value table with one of the following controls:

- Table view
- f(x) trend view
- f(t) trend view

A "display mode" is set in the value table. The display mode determines which data are shown in the value table.

Display modes

Three different display modes are available in the value table.

- Ruler window
  The ruler window shows the coordinate values of the trends on a ruler or values of a selected line in the table.
- Statistics area window
  The statistics area window shows the values of the lower limit and upper limit of the trends between two rulers or the selected area in the table. You can only connect the statistics area window with the f(t) trend display or the table view.
- Statistics window
  The statistics window displays the statistical evaluation of the trends. Among other things, the statistics include:
  - Minimum
  - Maximum
  - Average
  - Standard deviation
  - Integral

All windows can also display additional information on the connected trends or columns, such as time stamps.

See also

Configuring the value table (Page 724)

Creating statistics for Runtime data (Page 748)

Determining the coordinates of a point (Page 742)

Configuring the value table (RT Professional)

Introduction

The contents of the value table are shown in columns. The available columns depend on the connected control. A block is assigned to each column. You define the alignment and appearance of the column headers using the blocks. By default, the format of the connected control is used for the display format, such as the time display.

Requirements

- The table view or trend view is selected in the screen
- The Inspector window is open

Procedure

Proceed as follows to configure a value table:

1. Add the required table view to the screen from the "Tools" task card.
2. Click "Properties > Properties > General > Data source" in the Inspector window to connect the value table to the selected control.
3. Select the "mode" of the value table under "Properties > Properties > General".
4. Configure the properties of the value table columns under "Properties > Properties > Blocks".
5. Configure the "Properties > Properties > Columns":
   - Change the "Visibility", if required
   - Change the "Label", if required
   - If required, change the sequence of columns using "Up" and "Down"
6. Configure the view of the value table in the Inspector window:
   - Configure the view of the value table under "Properties > Properties > Appearance".
   - Configure the properties of the value table in Runtime under "Properties > Properties > "Window".

Result

The value table is configured.

See also

Value table basics (Page 723)

Configuring toolbar and status bar (Page 725)

Configuring the appearance of the table elements (Page 727)

Configuring table column sorting (Page 728)

Configuring the export of Runtime data (Page 728)

Determining the effect of the online configuration (Page 730)

Configuring toolbar and status bar (RT Professional)

Introduction

You operate the controls in Runtime using the buttons in the toolbar. The status bar displays status messages from the control. During configuration, set the content of the toolbar and status bar.

Requirement

- The table view is selected in the screen.
- The Inspector window is open

Configuring the toolbar

Proceed as follows to configure the toolbar:

1. In the Inspector window, configure the general properties of the toolbar, such as orientation and background color, under "Properties > Properties > Toolbar > Toolbar - General".
2. In the Inspector window, activate the buttons you need in Runtime under "Properties > Properties > Toolbar > Toolbar - Buttons".
   For information on the functions of the individual buttons, refer to the description of the corresponding WinCC control at "Operation in Runtime".
3. To set the order of the buttons, select the buttons in the list. You can then move the buttons to the desired position.
4. If needed, you can define a "hotkey" for the buttons on the toolbar.
5. If necessary, select the authorization needed to operate the buttons in Runtime.
6. If a button should not be operated in Runtime, unselect "Activate".
   You can reactivate a deactivated a button using a script in Runtime, for example.

Configuring the status bar

Proceed as follows to configure the status bar:

1. In the Inspector window, configure the general properties of the status bar, such as orientation and background color, under "Properties > Properties > Status bar > Status bar - General".
2. In the Inspector window, activate the elements you need in Runtime under "Properties > Properties > Status bar > Status bar - Elements".
   For further information on status bar elements, refer to the descriptions of the corresponding WinCC control at "Operation in runtime".
3. To set the order of the elements, select the element in the list. You can then move the element to the desired position.
4. To adjust the width of an element in the status bar, deactivate "Auto-size". You can then enter the width in pixels.

See also

Configuring the value table (Page 724)

Elements of the status bar (Page 751)

Configuring the appearance of the table elements (RT Professional)

Introduction

You can adjust the appearance of the table elements to your needs.

Requirement

- The table view or value table is selected in the screen.
- The Inspector window is open

Procedure

Proceed as follows to configure the colors of the table elements:

1. Define the following properties in the Inspector window under "Properties > Properties > Table > Table - Header":
   - Visibility of the column headers and row headers
   - Alignment of the column headers and row headers
   - Color design of column headers
2. Define the following properties in the Inspector window under "Properties > Properties > Table > Table - Structure":
   - Color design of table content, such as different colors for rows for better differentiation
   - Visibility of empty columns and rows
3. Define the following properties in the Inspector window under "Properties > Properties > Table > Table - Grid":
   - Design of grid lines
   - Character spacing in the table cells

Result

The appearance of the table elements is configured. You can also define further properties:

- Behavior and color design when selecting a table entry
- Table entry sorting

See also

Configuring the value table (Page 724)

Configuring table column sorting (Page 728)

Configuring table column sorting (RT Professional)

Introduction

If the operator is to be able to sort individual table columns in Runtime, configure the sorting within the table.

Depending on the configuration, the operator can sort a table column as follows:

- Clicking or double-clicking on the column header
- Clicking on a sorting button
  The sorting button is used to sort the selected column in the configured order. For the sorting button to appear, you must configure a vertical scroll bar.

Requirement

- The table view or value table is selected in the screen.
- The Inspector window is open

Procedure

To configure the sorting within a table, follow these steps:

1. In the Inspector window, select "Properties > Properties > Table > Table - Sorting > Sort by column header", "Sort with double-click", or "Sort with click".
2. Select the "Sorting order" option.
3. If needed, activate "Display sorting icon" and "Display sorting index".
4. If needed, activate "Display sorting button".
5. If you use the sorting button, select "Properties > Properties > Window > Scroll > Vertical scrollbar > Always" in the Inspector window.

Result

The sorting of the table columns is configured.

See also

Configuring the value table (Page 724)

Configuring the export of Runtime data (RT Professional)

Introduction

In Runtime, you can export the displayed data of a control in CSV format. Depending on the control used and the configuration, the user exports all data or only the selected data.

The filename and location of the CSV file are set by the operator on export. Alternatively, you can generate the filename automatically from the following wildcards:

- @OBJECTNAME: Object name of the control
- @CURRENTDATE: Current date
- @CURRENTTIME: Current time

Requirement

- The control is selected in the screen
- The Inspector window is open

Procedure

To configure the export of Runtime data, proceed as follows:

1. In the Inspector window, open "Properties > Properties > Data export".
   The data export settings are already configured.
2. If necessary, enter your own filename and a path for the export file.
3. Select the "scope" of the data export.
4. Configure the "Operation in Runtime".
5. So that the operator can start the data export, activate "Properties > Properties > Toolbar > Toolbar - Buttons > Export Data" in the Inspector window.

Result

In Runtime, the operator can export the data of the control via the toolbar.

See also

Configuring the value table (Page 724)

Determining the effect of the online configuration (RT Professional)

Introduction

In WinCC you configure the appearance and function of a control in Runtime. In addition, you can allow the operator to configure a control in Runtime. You can set whether changes to configuration in Runtime are retained, such as after a screen change, in the control under "Properties > Properties > Security > Persistence".

- No persistence
  The changes to the configuration are not saved. The changes are discarded on the next screen change and when the project is rejected.
- Persistence
  The changes to the configuration are saved. The changes are retained even after the project is deactivated and activated. You may also select "Response at screen change" to discard changes at the screen change.
- Persistence in Runtime
  The changes to the configuration are only saved during Runtime. If you deactivate the project, the changes are discarded. You may also select "Response at screen change" to discard changes at the screen change.

Note

If you change a configured control and load it onto the HMI device, the changes are overwritten in Runtime regardless of the settings.

Requirement

- A configured control
- The Inspector window is open

Procedure

To configure the persistence of data in Runtime, proceed as follows:

1. In the Inspector window, open "Properties > Properties > Security".
2. Select "Online configuration".
3. If needed, select the authorization.
4. Select the "Behavior on screen change".

Result

The behavior in online configuration is now set. In Runtime, depending on the settings, the user can adjust the configuration of the control.

If several operators are working on the HMI device, the individual configurations are accepted on a change of operator only following a screen change.

See also

Configuring the value table (Page 724)

2.10.3.5 Using controls in Runtime (RT Professional)

Using the trend and table view in Runtime (RT Professional)

Introduction

You operate the trend and table view using the toolbar buttons.

To operate a button via a script, transfer the "ID" of the button to the script. Use the symbols from the symbol library to label the button. The symbols are available in the "Tools > Graphics" task card under "WinCC graphic folder > Runtime control icons > Trend View".

Buttons of the f(x) trend view

The table below shows the buttons that are available in the f(x) trend view:

Symbol	Name	Function	ID
	"Tooltip"	Calls up the control help.	1
	"Configuration dialog"	Opens the configuration dialog in which you can change the properties of the control	2
	"Ruler"	Displays a ruler that shows the coordinates of the intersection point with a trend in the value table. Requirement: The value table is configured with "Ruler window" display mode.	3
	"Zoom area"	Zooms the trend view area. You define the area by dragging with the mouse. Use the "Original view" button to return to the original view.	4
	"Zoom +/-"	Zooms in or out of the trends in the trend view. Left-click: Zoom in <Shift + Left-click>: Zoom out Use the "Original view" button to return to the original view.	5
	"Zoom X axis + /-"	Zooms in or out on the time axis in the trend view. Left-click: Zoom in <Shift + Left-click>: Zoom out Use the "Original view" button to return to the original view.	6
	"Zoom Y axis +/-"	Zooms in or out on the value axis in the trend view. Left-click: Zoom in <Shift + Left-click>: Zoom out Use the "Original view" button to return to the original view.	7
	"Move trend area"	Moves the trend along the value axis and the time axis	8

Symbol	Name	Function	ID
	"Move axis range"	Moves the trend along the value axis	9
	"Original view"	Returns to the original view from the zoomed display	10
	"Select data connection"	Opens a dialog in which you select the data source: • Process value archive • Tag • Recipe (only f(x) trend view)	11
	"Select trends"	Opens a dialog in which you can hide and show trends. You can also select which trend is to be placed in the foreground.	12
	"Select time range"	Opens a dialog in which you configure the time range	13
	"Previous trend"	Displays the previous trend in the foreground.	14
	"Next trend"	Displays the next trend in the foreground.	15
	"Stop"	Stops the updated display. The data is buffered.	16
	"Start"	Continues the updated display. The buffered values are entered.	16
	"Print"	Prints the selected values. The print job is defined in the configuration dialog on the "General" tab.	17
	"Connect backup"	Opens a dialog in which you connect selected archives from WinCC Runtime	18
	"Disconnect backup"	Opens a dialog in which you disconnect selected archives from WinCC Runtime	19
	"Export data"	Exports all or selected data to a *.CSV file. Depending on the configuration and the privileges, the following options can be available: • Display export settings and start export • Select file name and directory	20
	"User-defined 1"	Displays the first button created by the user. The function of the button is user-defined	1001

Buttons of f(t) trend view

The table below shows the buttons that are available in the f(t) trend view:

Symbol	Name	Function	ID
	"Tooltip"	Calls up the control help.	1
	"Configuration dialog"	Opens the configuration dialog in which you can change the properties of the control	2
	"First data record"	Displays the course of a tag within a specified period starting with the first archived value. Requirement: Values come from a process value archive.	3
	"Previous data record"	Displays the course of a tag within the previous time interval, based on the currently displayed time interval. Requirement: Values come from a process value archive.	4
	"Next data record"	Displays the course of a tag within the next time interval, based on the currently displayed time interval. Requirement: Values come from a process value archive.	5
	"Last data record"	Displays the course of a tag within a specified period ending with the last archived value. Requirement: Values come from a process value archive.	6
	"Ruler"	Displays a ruler that shows the coordinates of the intersection point with a trend in the value table. Requirement: The value table is configured with "Ruler window" display mode.	7
	"Zoom area"	Zooms the trend view area. You define the area by dragging with the mouse. Use the "Original view" button to return to the original view.	8
	"Zoom +/-"	Zooms in or out of the trends in the trend view. Left-click: Zoom in <Shift + Left-click>: Zoom out Use the "Original view" button to return to the original view.	9
	"Zoom time axis +/-"	Zooms in or out on the time axis in the trend view. Left-click: Zoom in <Shift + Left-click>: Zoom out Use the "Original view" button to return to the original view.	10
	"Zoom value axis +/-"	Zooms in or out on the value axis in the trend view. Left-click: Zoom in <Shift + Left-click>: Zoom out Use the "Original view" button to return to the original view.	11
	"Move trend area"	Moves the trend along the value axis and the time axis	12
	"Move axis range"	Moves the trend along the value axis	13
	"Original view"	Returns to the original view from the zoomed display	14

Symbol	Name	Function	ID
	"Select data connection"	Opens a dialog in which you select the data source: • Process value archive • Tag • Recipe (only f(x) trend view)	15
	"Select trends"	Opens a dialog in which you can hide and show trends. You can also select which trend is to be placed in the foreground.	16
	"Select time range"	Opens a dialog in which you configure the time range	17
	"Previous trend"	Displays the previous trend in the foreground.	18
	"Next trend"	Displays the next trend in the foreground.	19
	"Stop"	Stops the updated display. The data is buffered.	20
	"Start"	Continues the updated display. The buffered values are entered.	20
	"Print"	Prints the selected values. The print job is defined in the configuration dialog on the "General" tab.	21
	"Statistics area"	Defines the time frame used to calculate the statistics.	22
	"Statistical analysis"	Displays the statistical values from a fixed "statistics area" of the trend or table view in the statistics window of the value table. Only available with a configured value table	23
	"Connect backup"	Opens a dialog in which you connect selected archives from WinCC Runtime	24
	"Disconnect backup"	Opens a dialog in which you disconnect selected archives from WinCC Runtime	25
	"Export data"	Exports all or selected data to a *.CSV file. Depending on the configuration and the privileges, the following options can be available: • Display export settings and start export • Select file name and directory	26
	"Relative axis"	Switches from displaying the absolute values to the percentage display of value axis. The high and low limits for the trend correspond with a range of 0 to 100%.	27
	"User-defined 1"	Displays the first button created by the user. The function of the button is user-defined	1001

Table view buttons

The following table shows the buttons that are available in the table view:

Symbol	Name	Function	ID
	"Tooltip"	Calls up the control help.	1
	"Configuration dialog"	Opens the configuration dialog in which you can change the properties of the control	2
	"First data record"	Displays the course of a tag within a specified period starting with the first archived value. Requirement: Values come from a process value archive.	3
	"Previous data record"	Displays the course of a tag within the previous time interval, based on the currently displayed time interval. Requirement: Values come from a process value archive.	4
	"Next data record"	Displays the course of a tag within the next time interval, based on the currently displayed time interval. Requirement: Values come from a process value archive.	5
	"Last data record"	Displays the course of a tag within a specified period ending with the last archived value. Requirement: Values come from a process value archive.	6
	"Edit"	Activates editing of table entries. To edit a value, double-click on the desired table cell. Requirement: Updated display is stopped.	7
	"Copy lines"	Copies the content of the selected lines to the clipboard. Requirement: Updated display is stopped.	8
	"Select data connection"	Opens a dialog in which you select the data source: • Process value archive • Tag • Recipe (only f(x) trend view)	9
	"Select columns"	Opens a dialog in which you can show or hide columns and change the column sequence.	10
	"Select time range"	Opens a dialog in which you configure the time range	11
	"Previous column"	Moves a value column ahead of the previous value column. The function refers to the value columns that are assigned to a time axis.	12
	"Next column"	Moves a value column behind the next value column. The function refers to the value columns that are assigned to a time axis.	13
	"Stop"	Stops the updated display. The data is buffered.	14
	"Start"	Continues the updated display. The buffered values are entered.	14

Symbol	Name	Function	ID
	"Print"	Prints the selected values. The print job is defined in the configuration dialog on the "General" tab.	15
	"Statistics area"	Defines the time frame used to calculate the statistics.	16
	"Statistical analysis"	Displays the statistical values from a fixed "statistics area" of the trend or table view in the statistics window of the value table. Only available with a configured value table	17
	"Connect backup"	Opens a dialog in which you connect selected archives from WinCC Runtime	18
	"Disconnect backup"	Opens a dialog in which you disconnect selected archives from WinCC Runtime	19
	"Export data"	Exports all or selected data to a *.CSV file. Depending on the configuration and the privileges, the following options can be available: • Display export settings and start export • Select file name and directory	20
	"User-defined 1"	Displays the first button created by the user. The function of the button is user-defined	1001

Buttons of the value table

The table below shows the buttons that are available in the value table:

Symbol	Name	Function	ID
	"Tooltip"	Calls up the control help.	1
	"Configuration dialog"	Opens the configuration dialog in which you can change the properties of the control	2
	"Ruler"	Displays a ruler that shows the coordinates of the intersection point with a trend in the value table. Requirement: The value table is configured with "Ruler window" display mode.	3
	"Statistics area"	Defines the time frame used to calculate the statistics.	4
	"Statistical analysis"	Displays the statistical values from a fixed "statistics area" of the trend or table view in the statistics window of the value table. Only available with a configured value table	5
	"Print"	Prints the selected values. The print job is defined in the configuration dialog on the "General" tab.	6

Symbol	Name	Function	ID
	"Export data"	Exports all or selected data to a *.CSV file. Depending on the configuration and the privileges, the following options can be available: • Display export settings and start export • Select file name and directory	7
	"User-defined 1"	Displays the first button created by the user. The function of the button is user-defined	1001

Using trend view in Runtime (RT Professional)

Online configuration of the f(t) trend view (RT Professional)

Introduction

In Runtime you configure online and thus change the appearance of the trend view. The table view configuration specifies whether online configurations are retained or discarded on a screen change or after Runtime is ended.

Overview

The following buttons make online configuration possible in the trend view:

- "Configuration dialog" button
- "Select data connection" button
- "Select trends" button
- "Select Time Range" button

Opening the configuration dialog

To open the configuration dialog, proceed as follows:

1. Click in the toolbar.

The configuration dialog is opened. You can change the table appearance, for example, in the configuration dialog.

Note

Printing a fixed time range

If you want to print a defined time range, make sure that the following options in the configuration dialog are deactivated:

- "Time axes > Update"
- "General > Open screen > Launch update".

Changing the data connection

The following table shows the configuration options for data connection:

Field	Description
Trend	Choose one of the configured trends.
Data Source	Define whether the selected trend is supplied with a logging tag or online tag.
Tag name	Select the tag name for the data connection.

Proceed as follows to change the data connection:

1. Click in the toolbar.
   The "Archive/tag selection" dialog is opened.
2. Choose the "Value column" for which you want to change the data connection.
3. Select the "Data supply" and the "Tag name".

Changing a time range

The following table shows the configuration options for the time range:

Field	Description
Time axis	Select the configured time axis for which you want to define a time range.
Time range	Set the time range: • If you want to define a fixed time interval, select setting "Start to end time". Enter the date and time for each. • If you want to define a time period, select the setting "Time range". Define the date and time for the start time. The length of the time interval to be displayed is determined by multiplying the "Factor" by the "Time unit". • If you want to display a certain number of values, select the setting "Number of measurement points". Define the date and time for the start time. Enter the desired number of measurement points in the input field.

To configure the time range, follow these steps:

1. Click in the toolbar of the trend view.
   The "Time - Selection" dialog opens.
2. Choose the "Time axis" for which you want to adapt the time range.
   If the trends in a trend view are to be displayed with a common time axis, the specified time range applies to all trends.
3. Configure the time range.
   The entry format of the date and time depends on the Runtime language used.

See also

Trend to Front (Page 741)

Using the zoom functions in trend windows (Page 739)

Starting and Stopping Update (Page 747)

Creating statistics for Runtime data (Page 748)

Displaying logged values (Page 750)

Elements of the status bar (Page 751)

Configure trend view (Page 699)

Using the zoom functions in trend windows (RT Professional)

Introduction

Key functions can be used for zooming in on, zooming out of and returning to the original view for trends, axes and various zoom areas of the trend window.

Overview

The following zoom functions are available in the trend window:

- "Zoom area"
- "Original view"
- "Zoom +/-"
- "Zoom time axis +/-"
- "Zoom value axis +/-"
- "Move trend area"

Requirement

- The trend view is open
- Buttons with zoom functions are configured
- Runtime is activated

Zooming in on a trend view area

Proceed as follows to zoom in on an area of the trend view:

1. Click on .
   The updated display is stopped.
2. Drag with the mouse to draw a box around the area to be zoomed.
   If there are at least two measured values within this area, the area of the trend is zoomed.
3. To return to the original view of the trend, click .
4. Click on to start the update again.

The default values are used for the axis.

Zooming in or out on trends

If you zoom in or out on a trend, the 50% value of the trend is always in the middle of the value axes.

Proceed as follows to zoom in or out on a trend:

1. Click on .
   The updated display is stopped.
2. To zoom in on a trend, click on the trend with the left mouse button.
3. To zoom out on a trend, hold down the <Shift> key and click on the trend with the left mouse button.
4. To return to the original view of the trend, click .
5. Click on to start the update again.

The default values are used for the axis.

Note

If you change the value area of a value axis on the "Value Axis" tab in the configuration dialog while zooming, the visible zoom area is set to the new value area.

Zooming in on the time axis or value axis

While zooming with time or value axes, the 50% value of the trend is always in the middle of the axes.

Proceed as follows to zoom the time axis or value axis:

1. Click to zoom the time axis.
   The updated display is stopped.
2. Click to zoom the value axis.
   The updated display is stopped.
3. To zoom in on an axis, click on the trend view with the left mouse button.
4. To zoom out on an axis, hold down the <Shift> key and click on the trend view with the left mouse button.
5. To return to the original view of the trend, click .
6. Click on to start the update again.

The default values are used for the axis.

Moving a trend area

Proceed as follows to move the trend area:

1. Click on .
   The updated display is stopped.
2. While holding the left mouse button pressed, move the cursor in the desired direction in the trend view.
   The displayed area in the trend window is adapted on the time axis and on the value axis.
3. Click on again to return to the original view of the trend window.

See also

Online configuration of the f(t) trend view (Page 737)

Trend to Front (RT Professional)

Introduction

If more than one trend are to be displayed in a trend window, you can use key functions to define which trends will be displayed in the foreground.

Requirement

- The following buttons are configured:
  - "Select trends"
  - "Previous trend"
  - "Next trend"
- The project is activated

Procedure

Proceed as follows to display a trend in the foreground:

1. Click on .
   The dialog for displaying or hiding trends is opened. In this dialog you can also define which trend is in the foreground.

   Note

   The first trend of a trend window cannot be hidden.

2. In order to display the next trend in the foreground, click on
3. In order to display the previous trend in the foreground, click on .

See also

Online configuration of the f(t) trend view (Page 737)

Determining the coordinates of a point (RT Professional)

Introduction

The "Ruler" button is used to determine the coordinates of a point on the trend by means of a ruler. You can zoom in on an area of the trend to make coordinate finding easier. If you display the ruler in the trend view, you can move the ruler at any time.

If you click on the trend with the mouse, several trend parameters are shown in the tooltip for the trend view.

Requirement

- A trend view is configured
- The value table is configured and connected with the trend view
- The "Ruler window" display mode is activated in the value table
- Runtime is activated

Procedure

Proceed as follows to determine the coordinates of a point:

1. Click on in the trend view.
   The ruler is shown.
2. Move the ruler to the desired position with the mouse.
3. If you want to zoom in on an area, click on .
   - Move the ruler to the desired position with the mouse.
   - Click to return to the original view.

Result

In the ruler window of the value table, besides the X value/time stamp and the Y value, the data that you have configured in the value table is shown in the columns.

In the value table, the indices "i" and "u" can be displayed in addition to the values:

- "i.": The displayed value is an interpolated value.
- "u.": The displayed value has an uncertain status:
  - The initial value after Runtime activation is unknown
  - A substitute value is used

Note

You can also identity the "uncertain" status of a value in the displayed trend curve. You must activate the "Value with uncertain status" option on the "Trends" tab under "Limit values".

The following image shows the coordinates of a point in the ruler window of the value table:

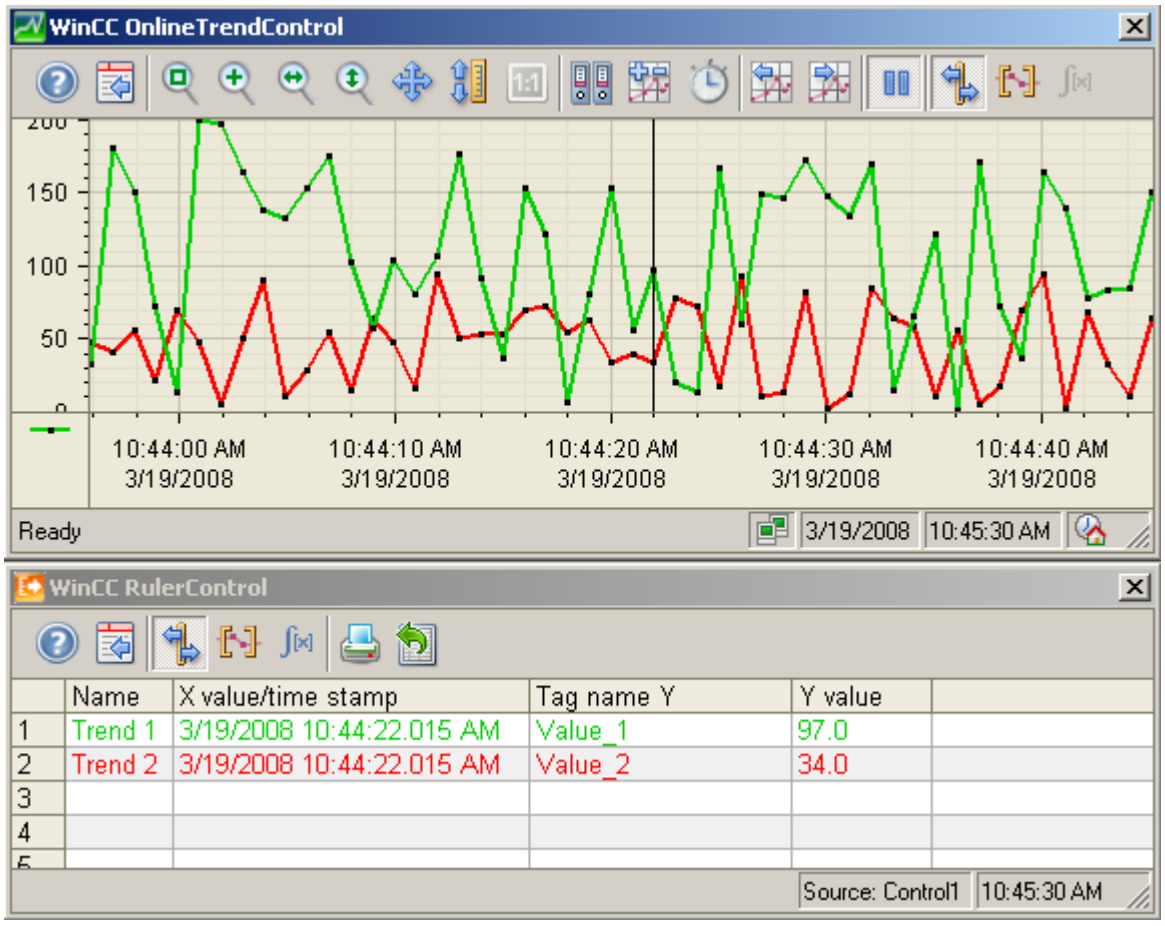

Alternative procedure

Alternatively, you can also connect the value table to the table display. In the "ruler window" display mode, the values of the selected line are displayed in the value table.

See also

Value table basics (Page 723)

Operating table view in Runtime (RT Professional)

Online configuration of the table view (RT Professional)

Introduction

In Runtime you configure online and thus change the appearance of the table view. The table view configuration specifies whether online configurations are retained or discarded on a screen change or after Runtime is ended.

Overview

The following buttons make online configuration possible in table view:

- "Configuration dialog" button
- "Select data connection" button
- "Select columns" button
- "Select Time Range" button

Opening the configuration dialog

To open the configuration dialog, proceed as follows:

Click in the toolbar.

The configuration dialog is opened. You can change the table appearance, for example, in the configuration dialog.

Changing the data connection

The following table shows the configuration options for data connection:

Field	Description
Value column	Choose the configured value column for which you want to change the data connection.
Data Source	Define whether the selected value column is supplied with a logging tag or online tag.
Tag name	Select the tag name for the data connection.

Proceed as follows to change the data connection:

1. Click in the toolbar.
   The "Archive/tag selection" dialog is opened.
2. Choose the "Value column" for which you want to change the data connection.
3. Select the "Data supply" and the "Tag name".

Showing or hiding columns

To show or hide columns, follow these steps:

1. Click in the toolbar.
   The dialog for showing or hiding columns is opened.
2. If required, change the sequence of the value columns that are assigned to a time column.

The value columns can only be moved in reference to the anchored time column.

Note

The first column of a table view cannot be hidden.

Changing a time range

The following table shows the configuration options for the time range:

Field	Description
Time column	Select the configured time column for which you want to define a time range.
time range	Set the time range: • If you want to define a fixed time interval, select setting "Start to end time". Enter the date and time for each. • If you want to define a time period, select the setting "Time range". Define the date and time for the start time. The length of the time interval to be displayed is determined by multiplying the "Factor" by the "Time unit". • If you want to display a certain number of values, select the setting "Number of measurement points". Define the date and time for the start time. Enter the desired number of measurement points in the input field.

To configure the time range, follow these steps:

1. Click in the toolbar of the table view.
   The "Time - Selection" dialog opens.
2. Choose the "Time column" for which you want to adapt the time range.
   If the columns of a table view are to be displayed with a common time axis, the specified time range applies to all columns.
3. Configure the time range.
   The entry format of the date and time depends on the Runtime language used.

See also

Editing the table field in Runtime (Page 746)
Moving columns in the table (Page 747)
Starting and Stopping Update (Page 747)
Creating statistics for Runtime data (Page 748)
Displaying logged values (Page 750)
Elements of the status bar (Page 751)
Configuring the Table View (Page 711)

Editing the table field in Runtime (RT Professional)

Introduction

You change the values displayed in the table view manually using the "Edit" button.

Requirement

- The table view is configured
- The "Edit" button is configured
- Runtime is activated

Procedure

Proceed as follows to edit a table field in Runtime:

1. Click in the table view.
   The updated display is stopped, the process data continues being archived.
2. Click on .
3. Double click on the desired table field of a value column.
4. Enter the new value.
   The changed value is archived.
5. In order to continue with the display of Runtime data in the table view, click on .

See also

Online configuration of the table view (Page 744)

Moving columns in the table (RT Professional)

Introduction

The time column is always shown in the first column in the table. The value columns that are assigned to this time column are displayed next. If there are more than one time columns configured, the second time column follows with the assigned value columns.

You can change the sequence of the value columns that are assigned with a time column in Runtime. The value columns can only be moved in reference to the anchored time column. The sequence of the time columns with the assigned value columns are defined on the "Time axes" tab.

Requirement

The following buttons are configured:

- "Select columns"
- "Previous column"
- "Next column".

Procedure

To change the sequence of the columns, follow these steps:

1. Click on .
   The dialog for changing the sequence of the columns is opened.
2. To hide a column, deactivate the option before the column name.
3. To move a value column, select the value value column:
   The function refers to the value columns that are assigned to a time column.
   – Click to move the value column behind the next value column.
   – Click to move the value column in front of the previous value column.

See also

Online configuration of the table view (Page 744)

Starting and Stopping Update (RT Professional)

Introduction

You can continue the update of the data contained in the control with the "Start/Stop" button.

Some buttons stop the update automatically, e.g. "Define statistics area"

The appearance of the button indicates whether the update is stopped or not:

- : Update has been stopped. To continue the update, click on the button.
- : The update has been started. To stop the update, click on the button

See also

Online configuration of the f(t) trend view (Page 737)

Online configuration of the table view (Page 744)

Creating statistics for Runtime data (RT Professional)

Introduction

You can generate an analysis of the process data for the Runtime data in the trend view or table view. You can display the evaluated data in the value table.

Requirement

- A trend view or table view is configured
- A value table is configured and connected with the trend view or table view
- Runtime is activated

Displaying data in a statistics area window

Requirement: The "Statistics area window" display mode is activated in the value table.

To display data in the statistics area window of the value table, proceed as follows:

1. In the trend view or table view, click on .
   The updated display is stopped, the process data continues being archived.
2. If you wish to evaluate data outside the displayed time range:
   – Click
   The "Time - Selection" dialog opens.
   – Enter the required time range.
   The data for the defined time range is displayed.

3. If you are using a trend view:
   - Click
     In the trend view, two vertical lines are displayed on the right and left edges.
   - To define the statistics area, move the two lines to the desired position.
4. If you are using a table view:
   - Use the mouse to select the lines for the desired time range in the table.
     For different columns with different time frames you can select different time ranges for the calculation of statistics.
   - Click in the toolbar

The evaluated data is displayed in the columns that you have configured in the statistics area window. The following figure shows the display in the statistics area window, using the trend view as an example:

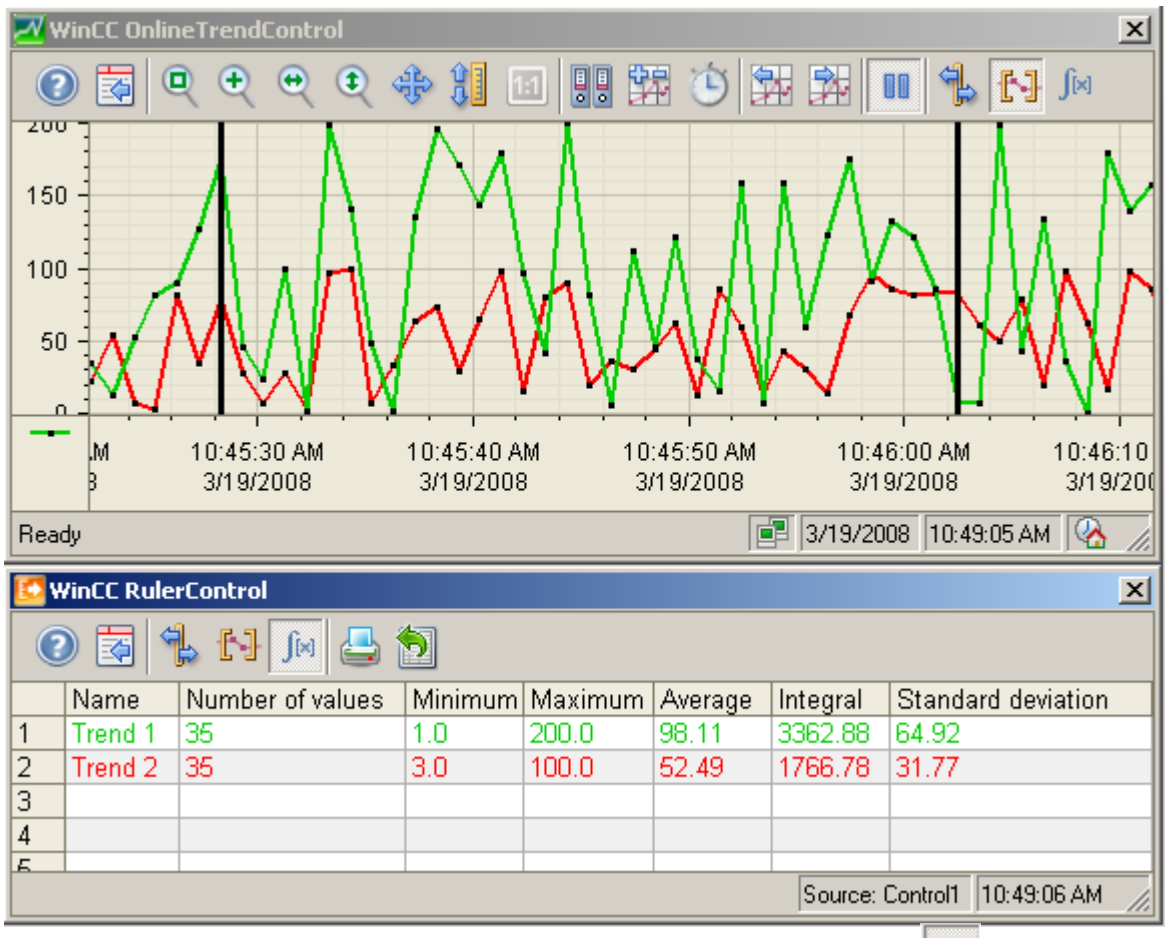

In order to continue with the display of Runtime data, click .

Note

For additional statistical analysis of process data and archiving of results you can write the scripts yourself.

See also

Value table basics (Page 723)

Online configuration of the f(t) trend view (Page 737)

Online configuration of the table view (Page 744)

Displaying logged values (RT Professional)

Introduction

Scroll through the displayed data of an archive using the buttons in the toolbar of a trend view or table view. If key combinations are configured, you can also use these for scrolling.

The buttons for browsing in archive are available only if data is supplied through archive tags.

The archived values of a tag are displayed within a time range in the trend or table view.

Requirement

- Time range is configured

Buttons for Archived Values

To scroll in archived values, proceed as follows:

1. To display the first data record of the time range, click on
2. To display the previous data record of the time range, click on .
3. To display the next data record of the time range, click on .
4. To display the last data record of the time range, click on .

See also

Time range basics (Page 695)

Online configuration of the f(t) trend view (Page 737)

Online configuration of the table view (Page 744)

Elements of the status bar (RT Professional)

Elements of the status bar

The status bar of the trend view or table view can contain the following elements:

Icon	Name	Description
	Connection status[1]	No faulty data connections
		Faulty data connections exist
		All data connections are faulty
"Line 1"[2]	Selected line	Shows the number of the selected line.
"Column 2"[2]	Selected column	Shows the number of the selected column.
"23.02.2010"	Date	Shows the system date.
"23:59:59"	Time	Shows the system time.
	Time base	Shows the time base used in the display of times.
[1]: If you double-click on the "Connection status" icon, the "Status of the data connections" window opens. The following properties of each data connection are listed in the window: • Name • Status • Tag name		
[2]: Only in the table view		

See also

Online configuration of the f(t) trend view (Page 737)

Online configuration of the table view (Page 744)

Configuring toolbar and status bar (Page 725)

Working with alarms 3

3.1 Principles

3.1.1 Alarm Logging in WinCC (Basic Panels)

Introduction

The alarm system allows you to display and record operating states and faults on the HMI device that are present or occur in a plant.

An alarm may have the following content:

No.	Time	Date	Alarm text	Status	Alarm class
5	12:50:24:590	24.02.2007	Boiler pressure above high limit.	Incoming Outgoing	Warning: Color Red

Alarm system in WinCC

The alarm system processes a variety of alarm types. The alarm procedures can be broken down into system-defined alarms and user-defined alarms:

- User-defined alarms are used to monitor the plant process.
- System-defined alarms monitor the HMI device.

The detected alarm events are displayed on the HMI device. Targeted access to the alarms combined with supplementary information about individual alarms ensures that faults are localized and cleared quickly. This reduces stoppages or even prevents them altogether.

The following figure shows the alarm system structure:

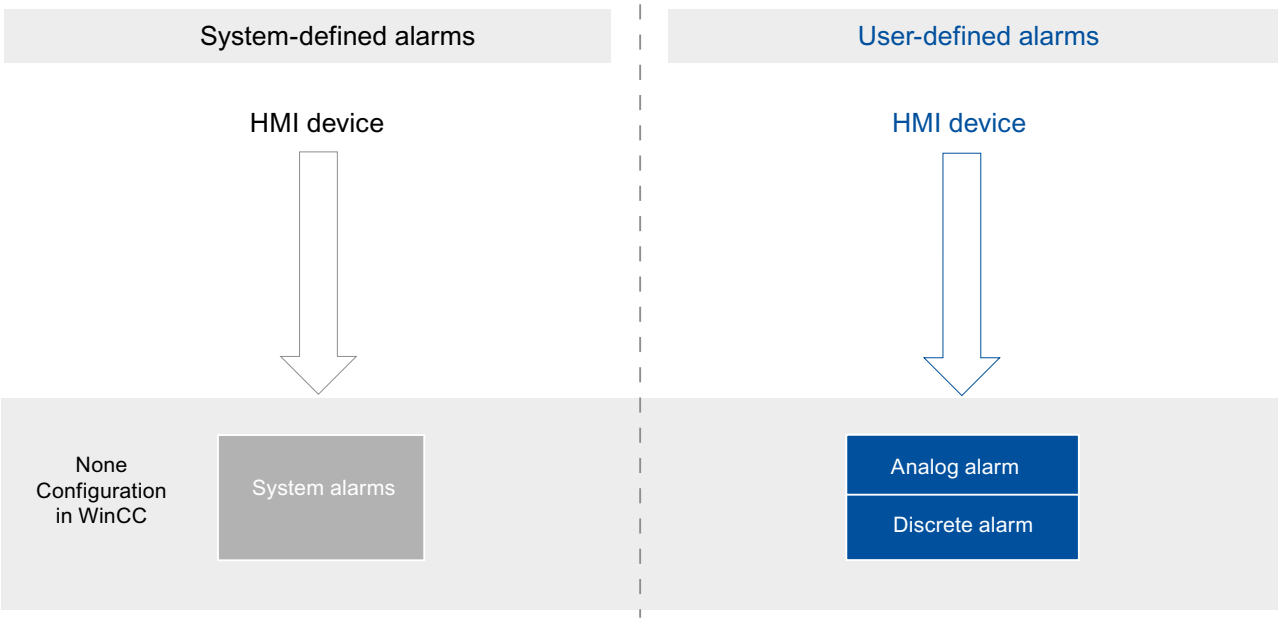

See also

Basics on alarm classes (Page 767)

3.1.2 Alarm Logging in WinCC (RT Advanced, Panels, Comfort Panels)

Introduction

Alarm logging allows you to display and record operating states and faults on the HMI device that are present or occur in a plant.

An alarm may have the following content:

No.	Time of day	Date	Alarm text	Status	Alarm class
5	12:50:24 :590	24.02. 2007	Boiler pressure above high limit.	Incoming Outgoing	Warning: Color Red

Alarm Logging in WinCC

Alarm logging processes various alarm procedures used by the PLC and the HMI device. The alarm procedures can be broken down into system-defined alarms and user-defined alarms:

- User-defined alarms serve to monitor the plant.
- System-defined alarms are used to monitor the HMI device or the PLC.

The detected alarm events are displayed on the HMI device. You can use the alarm logging system to log alarms from the ongoing process. Targeted access to the alarms combined with supplementary information about individual alarms ensures that faults are localized and cleared quickly. This reduces stoppages or even prevents them altogether.

The types of system-defined controller alarms depend on the controller used.

The following figure shows the structure of the alarm logging system for communication with the SIMATIC S7-300/400 controllers:

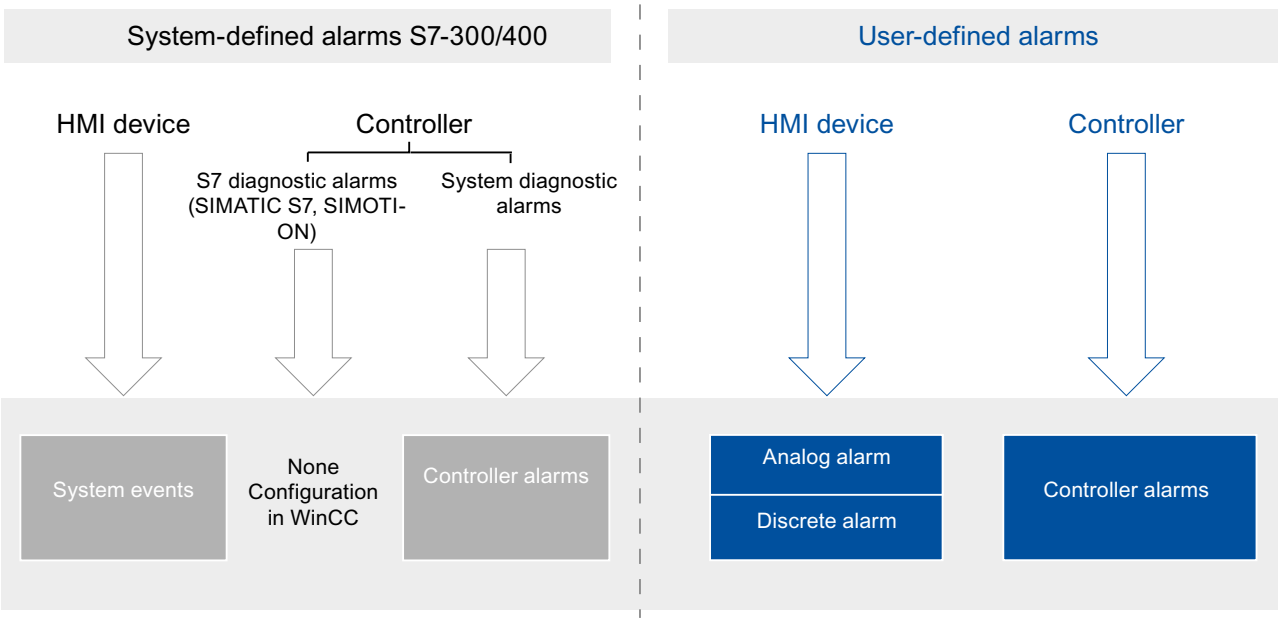

The following figure shows the structure of the alarm logging system for communication with the SIMATIC S7-1500 controllers:

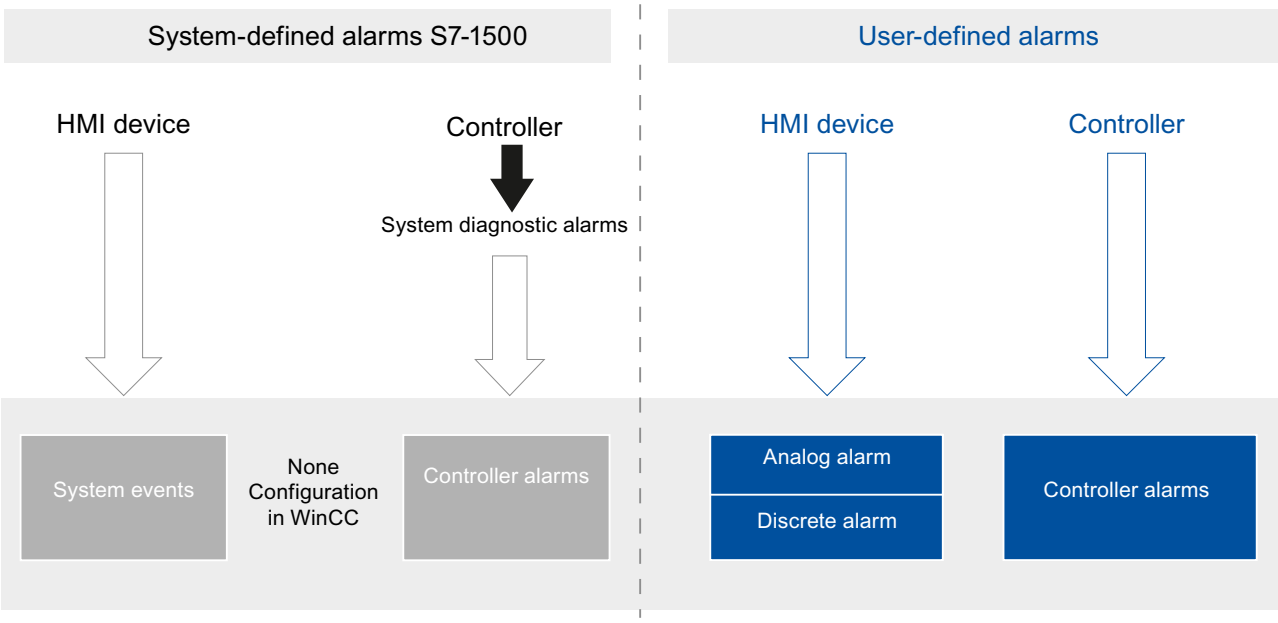

3.1.3 Alarm Logging in WinCC (RT Professional)

Introduction

Alarm logging allows you to display and record operating states and faults on the HMI device that are present or occur in a plant.

An alarm may have the following content:

No.	Time of day	Date	Alarm text	Status	Alarm class
5	12:50:24 :590	24.02. 2007	Boiler pressure above high limit.	Incoming Outgoing	Warning: Color Red

Alarm Logging in WinCC

Alarm logging processes various alarm procedures used by the PLC and the HMI device. The alarm procedures can be broken down into system-defined alarms and user-defined alarms:

- User-defined alarms serve to monitor the plant.
- System-defined alarms are used to monitor the HMI device or the PLC.

The detected alarm events are displayed on the HMI device. You can use the alarm logging system to log alarms from the ongoing process. Targeted access to the alarms combined with supplementary information about individual alarms ensures that faults are localized and cleared quickly. This reduces stoppages or even prevents them altogether.

The types of system-defined controller alarms on the controller box used.

The following figure shows the alarm logging structure for communication with the SIMATIC S7-300/400 controllers:

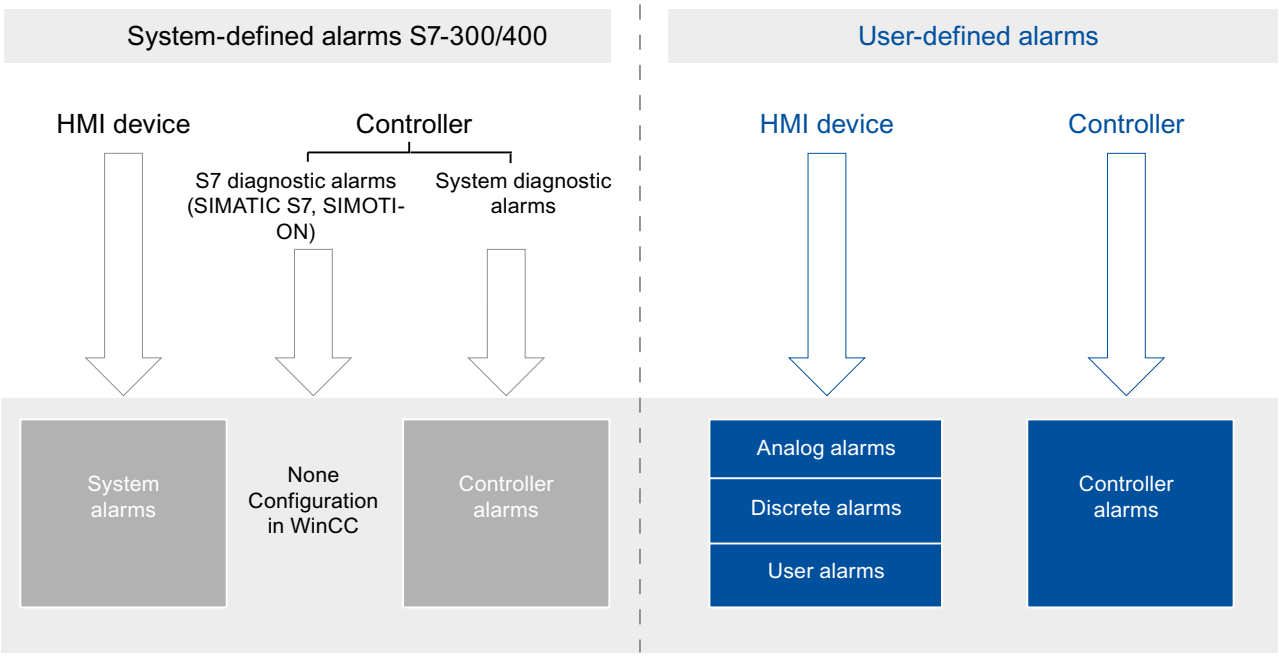

The following figure shows the alarm logging structure for communication with the SIMATIC S7-1500 controllers:

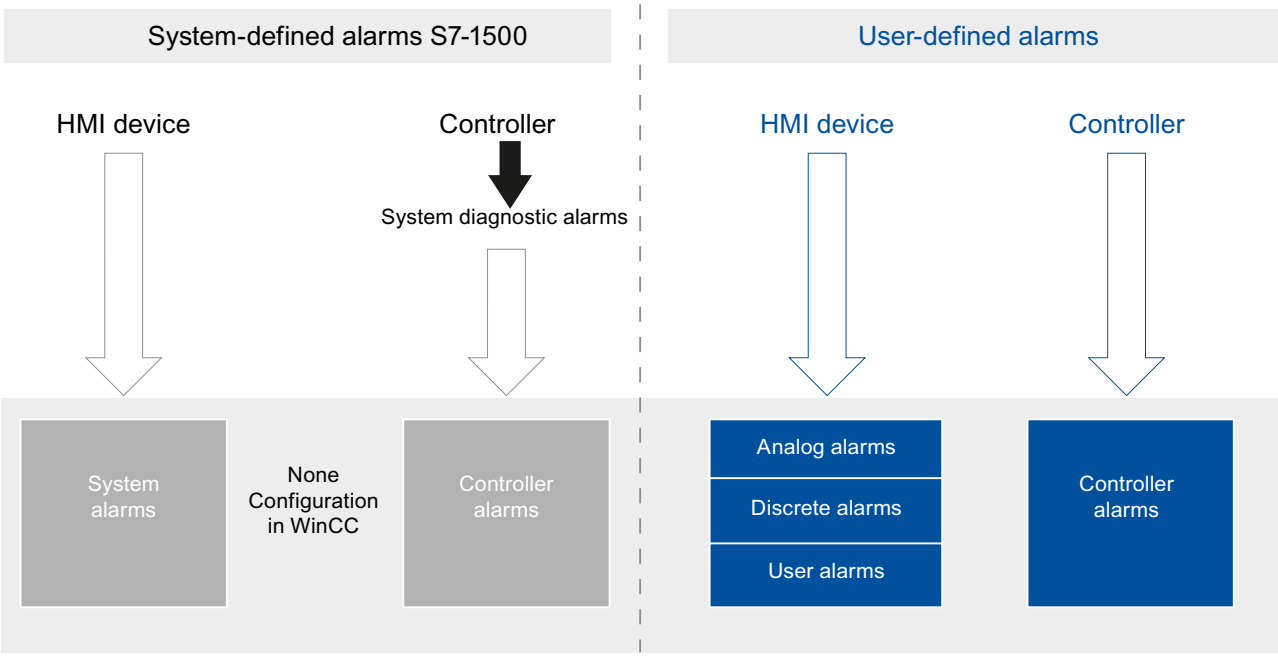

3.1.4 Alarm Procedures

3.1.4.1 Overview of the alarm types (Basic Panels)

Introduction

The alarm types serve various purposes for monitoring the plant. The alarms from the individual alarm types are configured and triggered in different ways.

Select the relevant tab in the "HMI alarms" editor to configure alarms based on the individual alarm types.

Alarm types in WinCC

WinCC supports the following alarm types:

User-defined alarms

- Analog alarms
  - Analog alarms are used to monitor limit violations.
- Discrete alarms
  - Discrete alarms are used to monitor states.

System-defined alarms

- System events
  - System events belong to the HMI device and are imported into the project.
  - System events monitor the HMI device.

See also

Analog alarms (Page 762)

3.1.4.2 Overview of the alarm types (RT Advanced, Panels, Comfort Panels)

Introduction

The alarm types serve various purposes for monitoring the plant. The alarms from the individual alarm types are configured and triggered in different ways.

Select the relevant tab in the "HMI alarms" editor to configure alarms based on the individual alarm types.

Alarm types in WinCC

WinCC supports the following alarm types:

User-defined alarms

- Analog alarms
  - Analog alarms are used to monitor limit violations.
- Discrete alarms
  - Discrete alarms are used to monitor states.
- Controller alarms
  - You configure controller alarms in STEP 7.
  - You continue to process the controller alarms in WinCC.

  Note
  Device dependency
  The controller alarm is not supported on all HMI devices.

System-defined alarms

- System-defined controller alarms
  - System-defined controller alarms consist of diagnostic alarms (SIMATIC S7) and system errors (SFM)
  - System-defined controller alarms are used to monitor the PLC.

  Note
  Device dependency
  System-defined controller alarms are not supported on all HMI devices.

- System events
  - System events belong to the HMI device and are imported into the project.
  - System events monitor the HMI device.

See also

Analog alarms (Page 762)
Controller alarms (Page 762)
System-defined controller alarms (Page 765)

3.1.4.3 Overview of the alarm types (RT Professional)

Introduction

The alarm types serve various purposes for monitoring the plant. The alarms from the individual alarm types are configured and triggered in different ways.

Select the relevant tab in the "HMI alarms" editor to configure alarms based on the individual alarm types.

Alarm types in WinCC

WinCC supports the following alarm types:

User-defined alarms

- Analog alarms
  - Analog alarms are used to monitor limit violations.
- Discrete alarms
  - Discrete alarms are used to monitor states.
- User alarms
  - User alarms serve to monitor operator input.
  - User alarms are triggered by means of alarm number and can be also used in scripts in Runtime.
- Controller alarms
  - You configure controller alarms in STEP 7.
  - You continue to process the controller alarms in WinCC.

  Note
  Device dependency
  Controller alarms and user alarms are not supported on all HMI devices.

System-defined alarms

- System-defined controller alarms
  - System-defined controller alarms are used to monitor the PLC.
  - Diagnostics alarms (SIMATIC S7) and system alarms (SFM) also belong to the category of system-defined controller alarms.

  Note
  Device dependency
  System-defined controller alarms are not supported on all HMI devices.

- System events
  - System events belong to the HMI device and are imported into the project.
  - System events monitor the HMI device.

3.1.4.4 Custom alarm classes

Analog alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Analog alarms signal limit violations during the process.

Example

The speed of the mixer in a fruit juice mixing plant must not be too high or too low. You can configure analog alarms to monitor the speed of the mixer. If the high or low limit for the speed of the mixer is violated, an alarm is output on the HMI device containing the following alarm text, for example: "Mixer speed is too low".

See also

Configuring Analog Alarms (Page 794)

Discrete alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Discrete alarms indicate a status in the current process.

Example

A fruit juice mixing plant consists of several tanks containing the ingredients. To ensure the correct mixing ratio of water, fruit concentrate, sugar, and flavoring, the valves in the intakes open and close at the right moment. This operation should be monitored.

You configure a suitable discrete alarm for all the valve states. If a valve on one of the four tanks opens or closes, an alarm is displayed, such as "Water valve closed".

The operator can thus monitor whether the plant is producing correctly.

See also

Configuring discrete alarms (Page 796)

Controller alarms (RT Professional, RT Advanced, Panels, Comfort Panels)

Example of an alarm

CPU operating mode switch to Stop

Description

The configuration engineer creates a custom controller alarm in STEP 7. The PLC status values, such as time stamp and process values, are mapped in the controller alarm. If controller alarms are configured in STEP 7, accept them into the integrated WinCC operation as soon as a connection is established to the PLC.

Note

Automatic update of new or modified controller alarms on the HMI device

When controller alarms are configured in STEP 7 and a connection to an S7-1500 controller with firmware version 2.0 (or later) is established, the controller alarms are sent to the HMI devices of device version V14. After changes of the controller alarms, the HMI device configuration must no longer be transferred.

In STEP 7, the controller alarm is assigned to an alarm class. You can import this alarm class with the controller alarm as a common alarm class.

Types of controller alarms

SIMATIC S7 supports different types of controller alarms. WinCC supports different types of controller alarms according to which Runtime is used.

Controller alarms for multiple HMI devices

If a PLC is connected to multiple HMI devices, the project engineer assigns display classes to the controller alarms in STEP7. The display classes determine the allocation to the HMI device. You can activate the display classes for your HMI device that are to be displayed on it. In this case, only the controller alarms from this display class will be displayed on the HMI device. Up to 17 display classes are possible.

See also

Sending a complete alarm from the controller to the HMI device and automatic update (Page 950)

Editing controller alarms (Page 803)

User alarms (RT Professional)

Description

User alarms indicate operator actions in Runtime. You can also configure user alarms in Runtime scripts for user-defined applications.

Example

An alarm of alarm class "Errors" is displayed on the HMI device of a plant. The operator eliminates this error in the plant and acknowledges the alarm in the alarm view of the HMI device. To monitor by whom and when the error was cleared, assign a user alarm to the appropriate button of the alarm view. The alarm contains the following information, for example:

- Type and content of the acknowledged alarm
- Time of acknowledgment
- Operator
- Date

3.1.4.5 System-defined alarms

System events (Basic Panels)

Examples for alarms

- "An online connection to the PLC is established."

Description

A system alarm indicates the status of the system, plus communication errors between the HMI device and system.

Under "Runtime settings > Alarms" you specify how long a system alarm is shown on the HMI device.

Support

The reference contains a list of the possible system events, along with the cause and possible remedies. If you contact online support because of a system alarm on the HMI device, you will need the alarm number and tags used in the system alarm.

System events (RT Professional, RT Advanced, Panels, Comfort Panels)

Examples of alarms

- "Overflow of the buffer for printing lines in text mode"
- "No printer is installed or a default printer has not been set up."

Description

A system event indicates the system status, including errors in the communication between the HMI device and system.

The system events depend on the HMI device.

You can load system events in the "HMI alarms" editor from a an *.xml file.

Support

The reference contains a list of the possible system events, along with the cause and possible remedies. When contacting Online Support because of a system event on the HMI device, you need the event number and tags used in the system event.

See also

System events (Page 1000)

Editing system events (Page 802)

System-defined controller alarms (RT Advanced, Panels, Comfort Panels)

Example of an alarm

Valve open due to overpressure

Description

System-defined controller alarms are installed with STEP 7 and are only available if WinCC is operated in the STEP 7 environment.

WinCC processes system-defined controller alarms of the type "SIMATIC S7 diagnostic alarm".

S7 diagnostic alarms show states and events in the SIMATIC S7 controllers. You only have read access to S7 diagnostic alarms.

- The system assigns the alarms to the "Diagnostics" alarm class.
- The available properties of the alarm, for example, the number of the alarm group, are accepted without changes.
- S7 diagnostic alarms are not acknowledged or reported. They are for signaling purposes only.

See also

Enabling system-defined controller alarms (Page 805)

3.1.5 Alarm states (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

An alarm assumes various alarm states in Runtime. The user analyzes and reports on the process execution with reference to the alarm states.

Note

Device dependency

Reporting and logging are not available for all HMI devices.

Description

Every alarm has an alarm status. The alarm states are made up of the following events:

- **Incoming**
  The condition for triggering an alarm is satisfied. The alarm is displayed, such as "Boiler pressure too high".
- **Outgoing**
  The condition for triggering an alarm is no longer satisfied. The alarm is no longer displayed as the boiler was vented.
- **Acknowledge**
  The operator acknowledges the alarm.

Alarms without acknowledgment

The following table shows the alarm states for alarms that do not have to be acknowledged:

Status	Description
Incoming	The condition for an alarm is satisfied.
Outgoing	The condition for an alarm is no longer satisfied.

Alarms with acknowledgment

The following table shows the alarm states for alarms that have to be acknowledged:

Status	Description
Incoming	The condition for an alarm is satisfied
Outgoing, not acknowledged	The condition for an alarm is no longer satisfied. The operator has not acknowledged the alarm.
Outgoing, and subsequently acknowledged	The condition for an alarm is no longer satisfied. The operator has acknowledged the alarm after this time.

Status	Description
Incoming, acknowledged	The condition for an alarm is satisfied. The operator has acknowledged the alarm.
Outgoing, but acknowledged first	The condition for an alarm is no longer satisfied. The operator acknowledged the alarm while the condition was still satisfied.

Each occurrence of these states can be displayed and logged on the HMI device and a protocol printed.

Note

The display text for the states of an alarm is language-specific and configuration-specific.

3.1.6 Alarm classes

3.1.6.1 Basics on alarm classes (Basic Panels)

Introduction

Many alarms occur in a plant. These are all of different importance. You can assign the alarms of your project to alarm classes to clearly show the user which of the alarms are most important.

Description

The alarm class defines how an alarm is displayed. The alarm class specifies if and how the user has to acknowledge alarms of this alarm class.

A new alarm class with mandatory acknowledgment is generated in WinCC.

Note

The choice of display modes for alarm classes depends on the options on your HMI device.

Examples of how to use alarm classes

- The alarm "Speed of fan 1 in upper tolerance range" has alarm class "Warnings". The alarm is displayed with a white background. The alarm does not have to be acknowledged.
- The alarm "Speed of fan 2 has exceeded upper warning range" is assigned to the "Errors" alarm class. The alarm is displayed with a red background and flashes at high frequency in runtime. The alarm is displayed until the user acknowledges it.

Using alarm classes

Use the following alarm classes to define the state machines and display of alarms for your project:

- Predefined alarm classes
  You cannot delete predefined alarm classes and edit them only to a limited extent. Predefined alarm classes have been created for each HMI device under "HMI alarms > Alarm classes".
- Custom alarm classes
  You can create new alarm classes under "HMI alarms > Alarm classes", configure how you want the alarms to be displayed, and define an acknowledgment model for alarms of this alarm class. The possible number of user-defined alarm classes depends on the Runtime used in your project.

Note

Alarm classes from STEP 7

If an HMI device is connected to the controller, alarm classes from STEP 7 will also be displayed in WinCC.

Display of system-defined controller alarms (configured in STEP 7) is not supported by Basic Panels. You can find additional information on the topic of "Controller alarms" and "Alarm logging" under Alarm Logging in WinCC (Page 753).

See also

Creating alarm classes (Page 785)

Creating alarm classes (Page 786)

Predefined alarm classes (Page 770)

Using common alarm classes (Page 788)

Alarm Logging in WinCC (Page 753)

3.1.6.2 Basics on alarm classes (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Many alarms occur in a plant. These are all of different importance. You can assign the alarms of your project to alarm classes to clearly show the user which of the alarms are most important.

Description

The alarm class defines how an alarm is displayed. The alarm class specifies if and how the user has to acknowledge alarms of this alarm class.

A new alarm class with mandatory acknowledgment is generated in WinCC.

Note

The choice of display modes for alarm classes depends on the options on your HMI device.

Examples of how to use alarm classes

- The alarm "Speed of fan 1 in upper tolerance range" has alarm class "Warnings". The alarm is displayed with a white background. The alarm does not have to be acknowledged.
- The alarm "Speed of fan 2 has exceeded upper warning range" is assigned to the "Errors" alarm class. The alarm is displayed with a red background and flashes at high frequency in runtime. The alarm is displayed until the user acknowledges it.

Using alarm classes

Use the following alarm classes to define the state machines and display of alarms for your project:

- Predefined alarm classes
  You cannot delete predefined alarm classes and edit them only to a limited extent. Predefined alarm classes have been created for each HMI device under "HMI alarms > Alarm classes".
- Custom alarm classes
  You can create new alarm classes under "HMI alarms > Alarm classes", configure how you want the alarms to be displayed, and define an acknowledgment model for alarms of this alarm class. The possible number of custom alarm classes depends on which runtime is used in your project.
- Common alarm classes
  Common alarm classes are displayed under "Shared data > Alarm classes" in the project tree and can be used for the alarms of an HMI device. Common alarm classes originate in the alarm configuration of STEP 7. If needed, you can create additional common alarm classes in WinCC.

See also

Creating alarm classes (Page 785)

Creating alarm classes (Page 786)

Predefined alarm classes (Page 770)

Using common alarm classes (Page 788)

3.1.6.3 Predefined alarm classes (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Predefined alarm classes

The following alarm classes already created in WinCC for every HMI device:

Alarm classes for user-defined alarms

- "Warnings"
  The "Warnings" alarm class is designed to show irregular statuses and routines in the process. Users do not acknowledge alarms from this alarm class.
- "Errors"
  The "Errors" alarm class is intended to show critical or dangerous states or limit violations in the process. The user must acknowledge alarms from this alarm class.

Alarm classes for system-defined alarms

- "System"
  The "System" alarm class contains alarms that display states of the HMI device and the PLCs. Alarms of the "System" alarm class belong to the system alarms.
- "Diagnosis Events"
  The "Diagnosis Events" alarm class contains alarms that display states and events in SIMATIC S7 controllers. Users do not acknowledge alarms from this alarm class.

  Note
  Device dependency

  The "Diagnosis Events" alarm class is not available for all HMI devices.

- "Safety Warnings"
  The "Safety Warnings" alarm class contains alarms for fail-safe operation. Users do not acknowledge alarms from this alarm class. Alarms of the "Safety Warnings" alarm class belong to the system alarms.

  Note
  Device dependency

  The "Safety Warnings" alarm class is not available for all HMI devices.

Priority of alarm classes on Comfort Panels and in RT Advanced

The following table shows the priority of alarm classes:

Alarm class	Priority
Errors	12
Warnings	4
System	16

Alarm class	Priority
Diagnosis Events	12
Alarm with single acknowledgment	12
Alarm without acknowledgment	4

If you filter the alarm view by priority, the alarm with priority "0" will appear at the top.

Note

If you configure the priority for the user-defined alarm classes, the priority is only adopted for controller alarms. For the other alarms, the priority is overwritten by the engineering system.

See also

Basics on alarm classes (Page 768)

Creating alarm classes (Page 786)

Creating alarm classes (Page 785)

Using common alarm classes (Page 788)

3.1.7 Acknowledgement

3.1.7.1 Acknowledging alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

To make sure that an alarm was registered by the plant operator, configure this alarm so that it is displayed until acknowledged by the operator. Alarms that display critical or hazardous states in the process have to be acknowledged.

Description

The acknowledgment of an alarm is an event that is logged and reported. Acknowledging an alarm changes the alarm status from "Incoming" to "Acknowledged". When the operator acknowledges an alarm, the operator confirms that he or she has processed the status that triggered the alarm.

Note

HMI device dependency

Reporting and logging are not available for all HMI devices.

Triggering acknowledgment of an alarm

In Runtime, you trigger alarm acknowledgments in various ways:

- Acknowledgement by the authorized user at the HMI device
- Automatic acknowledgment by the system without operator action, e.g. by means of
  - Tags
  - PLC
  - System functions in function lists or scripts

Note
HMI device dependency

Scripts are not available for all HMI devices.

Acknowledging alarms that belong together

To make the alarm system clearer and easier to use in Runtime, you can configure an alarm group. You can acknowledge all alarms belonging to this alarm group in a single pass.

Acknowledgment by the PLC

Discrete alarms will be automatically acknowledged by the PLC, if necessary. The acknowledgment is triggered by a bit in the "Acknowledgment tag PLC". You define the bit and tag at the configuration stage.

Acknowledgment of an alarm on the HMI device

In Runtime, the user acknowledges an alarm in one of the following ways, depending on the configuration:

- Using the acknowledgment button <ACK> on the HMI device
- Using the button in the alarm view
- Using configured function keys or buttons in screens

Note
Acknowledgment button <ACK> on the HMI device

To ensure that critical alarms are processed only by authorized users, protect the "ACK" button on the HMI devices, including the operating controls and display objects of the alarms. Use the appropriate operator authorization for this.

Note
HMI device dependency

The acknowledgement key <ACK> is not available on all HMI devices.

Reaction to the acknowledgment of alarms of an alarm group

- Alarm acknowledgment by the operator
  You acknowledge the alarm of an alarm group, for example, using the <ACK> acknowledgment key or a function key. All alarms of the alarm group are acknowledged.
- Alarm acknowledgment by the controller
  The controller sets a tag bit to acknowledge the alarm of an alarm group.
  The respective alarm is acknowledged.

3.1.7.2 Acknowledgment model (RT Advanced, Basic Panels, Panels, Comfort Panels)

Overview

You define the state machines for an alarm class. Alarms that are assigned to this alarm class will be acknowledged on the basis of these state machines. The following state machines are used in WinCC:

- Alarm without acknowledgment
  This alarm comes and goes without having to be acknowledged. There is no visible response from the system.
- Alarm with simple acknowledgment
  This alarm must be acknowledged as soon as the event that triggers the alarm occurs. The alarm remains pending until it is acknowledged.

3.1.7.3 Acknowledgment model (RT Professional)

Overview

You define the state machines for an alarm class. Alarms that are assigned to this alarm class will be acknowledged on the basis of these state machines. The following state machines are used in WinCC:

- Alarm without acknowledgment
  This alarm comes and goes without having to be acknowledged. There is no visible response from the system.
- Alarm with simple acknowledgment
  This alarm must be acknowledged as soon as the event that triggers the alarm occurs. The alarm remains pending until it is acknowledged.
- Alarm with dual-mode acknowledgment
  The alarm requires one acknowledgment as soon as the event that triggers the alarm has occurred, and a second acknowledgment when the event that triggers the alarm is no longer present. The alarm remains pending until it is acknowledged.
- Alarm without "outgoing" status with acknowledgment
  This alarm is displayed in the alarm view until it is acknowledged. It then disappears from the alarm view. The alarm is only logged.

- Alarm without "outgoing" status without acknowledgment
  This alarm is displayed, and goes out when the event that triggered the alarm is no longer present. The alarm is not added to the alarm view. The alarm is only logged.
- First value alarm with flashing and single acknowledgment
  The first value alarm is a form of alarm processing in which the first alarm in a list of alarms to undergo a status change since the last acknowledgment is highlighted. Only the first alarm in this alarm class is displayed flashing in the alarm window.
- New value alarm with flashing and single acknowledgment
  The new value alarm is a form of alarm processing in which any alarms from a list of alarms that have undergone a state change since the last acknowledgment are highlighted.
- New value alarm with flashing and dual-mode acknowledgment
  The alarm flashes if the value being monitored has changed. You acknowledge this alarm incoming and outgoing.

Overview of the state machines for multiple alarms

- Group Acknowledgment/Single Acknowledgment of Ala n the Alarm View
  The alarm view has a "Group acknowledgment" button . This button triggers the acknowledgment of all visible alarms that require acknowledgment and are pending in the alarm view. If the "Single acknowledgment" option is enabled for an alarm, you cannot use roup acknowledgment function and must acknowledge the alarm separately using the button.
- Acknowledging Alarms in Alarm Groups
  If an alarm group is assigned an acknowledgment tag, use this tag in Runtime for a group acknowledgment of all alarms of this alarm group. You can configure suitable buttons or function keys for using this tag.

Central alarm state machine

You can configure the acknowledgment of the alarm annunciator in the following ways:

- You acknowledge the alarm annunciator alongside with the alarm that triggered it.
- To acknowledge the alarm annunciator, use a separate button in the alarm view.
- A specified tag is used to control the alarm annunciator.

Emergency acknowledgment

In exceptional circumstances, you can acknowledge an alarm with reference to its alarm number. The acknowledgment bit is sent to the AS even if the alarm is currently not pending.

For this reason, the function should only be used in extreme emergency situations.

3.1.8 Alarm groups (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Many alarms from different areas and processes occur in a plant. You can compile associated alarms into alarm groups.

Alarm groups

You can use the alarm groups to monitor the parts of the plant and to acknowledge the associated alarms together as required.

Alarm groups can contain alarms from different alarm classes. You only assign alarms that require acknowledgment to alarm groups.

Using alarm groups

It is a good idea to compile alarm groups for alarms such as the following:

- Alarms that are caused by the same fault.
- Alarms of the same type
- Alarms from a machine unit, such as "Fault in drive XY"
- Alarms from an associated part of the process, such as "Fault in cooling water supply"

Display in Runtime

In Runtime, the "Alarm group" column displays the number of the alarm group to which the alarm belongs.

See also

Configuring alarm groups (Page 791)

3.1.9 Alarm groups (RT Professional)

3.1.9.1 Alarm Group Basics (RT Professional)

Introduction

Many alarms from different areas and processes occur in a plant. You use alarm groups to clearly represent the states of the plant areas on the HMI device, and to make the alarm system for your project easy to operate.

Description

An alarm group is a compilation of single alarms. An alarm group has several tags that address a specific property of the it alarms contains. You define the tags and their bits for a group. To display an alarm group on the HMI device, configure its visualization individually via the alarm group's status tag.

Application

You can use alarm groups for the following purposes, for example:

- To compile alarms that were caused by the same fault,
  such as "power failure"
- To compile alarms of the same type,
  such as "fuse blown"
- To sort and monitor alarms from a machine unit
  such as "fault in drive XY"
- Alarms from an associated part of the process
  such as "fault in cooling water supply"
- To process a group of all alarms from a plant area, such as:
  - Acknowledging alarms
  - Locking alarms
  - Suppressing alarm displays
- To visualize the status of plant areas
- To connect plant processes with the acknowledgment of alarms

Alarm Groups in WinCC

In WinCC, you work with two types of alarm group:

- Custom alarm groups
  You create custom alarm groups as required. These contain alarms that must be acknowledged, as well as sublevel alarm groups. You can create a hierarchy of custom alarm groups that contains up to five subgroups.
- Alarm Groups from Alarm Classes
  An alarm group is already created for each predefined alarm class. When you create a new alarm class, an alarm group is created with the same name. All the alarms from this alarm class will also be contained in the alarm group from the alarm class. Any changes to the alarm class configuration will be transferred to the "Alarm Groups" tab. The configuration for the corresponding alarm group will be updated. You only edit the properties of the alarm group for alarm groups from alarm classes.

Note

An alarm is always contained in an alarm group from alarm classes. An alarm that requires acknowledgment can also belong to a custom alarm group.

See also

Configuring alarm groups (Page 792)

3.1.9.2 Hierarchical alarm groups (RT Professional)

Introduction

WinCC allows you to compile alarms and custom alarm groups hierarchically. In this way, you construct a hierarchy of alarms that allows you to clearly and quickly detect and process the states of complex plants.

You can only create this hierarchy from alarms and alarm groups with custom alarm groups.

Structure of Hierarchical Alarm Groups

You compile custom alarm groups and single alarms hierarchically as required. You can also subsequently add child alarm groups.

The following figure shows a hierarchical alarm group in WinCC:

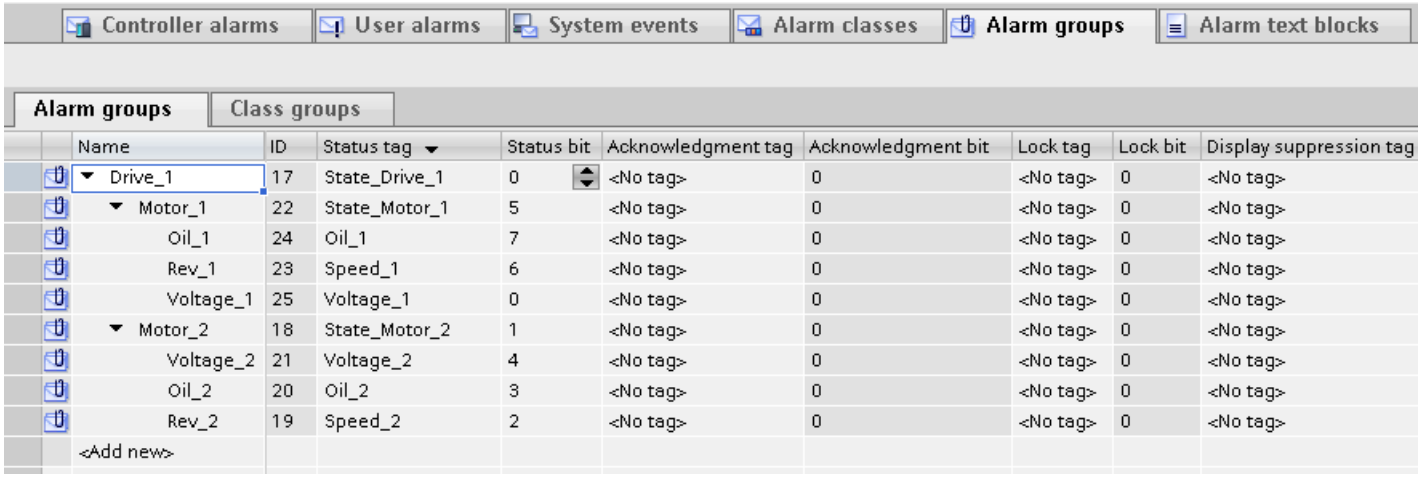

Name	ID	Status tag	Status bit	Acknowledgment tag	Acknowledgment bit	Lock tag	Lock bit	Display suppression tag
Drive_1	17	State_Drive_1	0	<No tag>	0	<No tag>	0	<No tag>
Motor_1	22	State_Motor_1	5	<No tag>	0	<No tag>	0	<No tag>
Oil_1	24	Oil_1	7	<No tag>	0	<No tag>	0	<No tag>
Rev_1	23	Speed_1	6	<No tag>	0	<No tag>	0	<No tag>
Voltage_1	25	Voltage_1	0	<No tag>	0	<No tag>	0	<No tag>
Motor_2	18	State_Motor_2	1	<No tag>	0	<No tag>	0	<No tag>
Voltage_2	21	Voltage_2	4	<No tag>	0	<No tag>	0	<No tag>
Oil_2	20	Oil_2	3	<No tag>	0	<No tag>	0	<No tag>
Rev_2	19	Speed_2	2	<No tag>	0	<No tag>	0	<No tag>
<Add new>								

Note

If you subsequently add an alarm group, the numbering will no longer be consecutive.

NOTICE
When you create a hierarchy of alarm groups, make sure that you do not use them recursively.

Note

If you delete an alarm group that contains other alarm groups, all lower-level alarm groups will be deleted as well.

Components of parent alarm groups

A parent alarm group contains the following elements as required:

- Single alarms
- Child alarm groups

Display suppression tag

You only assign the display suppression tag to alarm groups that contain no alarm subgroups and to customized alarm groups.

See also

Configuring alarm groups (Page 792)

3.1.9.3 Tags of an alarm group (RT Professional)

Tags of an alarm group

You can assign the following tags to an alarm group:

Status tag

The status tag stores the alarm states of an alarm group.

The status tag stores the acknowledgment request. The acknowledgment request is pending in an alarm group when at least one of the alarms in the group is queued and this alarm requires acknowledgement.

The status tag is used to visualize the alarm states and the acknowledgment request of an alarm group.

The status tag can be checked by other WinCC components.

Lock tag

The lock tag stores the locked status of an alarm group.

The lock tag is used to analyze the locked status of an alarm group.

Acknowledgment tag

The acknowledgment tag stores the "acknowledged" status of an alarm group.

The acknowledgment tag is used to acknowledge an alarm group.

The acknowledgment tag can be polled by other WinCC components.

Note

The acknowledgment bit of an acknowledgment tag is not automatically reset. You can configure a button or function key to do this. Use this button to reset the acknowledgment bit of the acknowledgment tag.

The acknowledgment tag of an alarm group does not display the acknowledgment via the alarm view.

Display suppression tag

The display suppression tag automatically suppresses the display of all the alarms for an alarm group in response to a specific plant status.

The display suppression tag returns the current value of the plant status as a value between 1 and 32. All alarms of the alarm group for which the corresponding bit of the number between 1 and 32 is set in the display suppression mask are suppressed.

See also

Configuring alarm groups (Page 792)

Configuring Alarm Display Suppression (Page 860)

Bit assignment for lock tags (Page 1023)

Bit assignment for display suppression tags (Page 1023)

Bit assignment of status tag (Page 1018)

Bit assignment for acknowledgment tags (Page 1021)

3.1.10 Alarm number (RT Advanced, Basic Panels, Panels, Comfort Panels)

Assigning alarm numbers

The system assigns unique alarm numbers within an alarm type.

Note

When adapting alarm numbers, observe the uniqueness of the alarm number within an alarm type.

3.1.11 Alarm number (RT Professional)

Assigning alarm numbers

The system assigns unique alarm numbers for the project.

Note

When adapting alarm numbers, observe the inter-project uniqueness of the alarm number.

The system event number overrides a custom alarm number. If you are using the alarm number of a system event for a user-defined alarm, change the alarm number of the user-defined alarm.

3.2 Working with Alarms

3.2.1 Alarm components and properties (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Overview

You configure the components of alarms in WinCC. The following table shows the basic components of alarms:

Alarm class	Alarm number	Time of day	Date	Alarm status	Alarm text	Alarm group	Tooltip	Trigger tag	Limit
Warn-ing	1	11:09:14	06.08.2007	IO	Maximum speed reached	2	This alarm is ...	speed_1	27
Sys-tem	110001	11:25:58	06.08.2007	I	Switch to "Online" mode	0	This alarm is ...	PLC-Varia-ble_1	–

Alarm class

Alarm classes, such as "Warnings" or "Errors". The alarm class defines the following for an alarm:

- State machine
- Appearance in Runtime (e.g. color)
- Logging

Note

Device dependency

Logging is not available for all HMI devices.

Alarm number

An alarm is identified by a unique alarm number. The alarm number is assigned by the system. You can change the alarm number to a sequential alarm number, if necessary, to identify alarms associated in your project.

Time and date

Every alarm has a time stamp that shows the time and date at which the alarm was triggered.

Alarm status

An alarm has the events "Incoming," "Outgoing," "Acknowledge." For each event, a new alarm is output with the current status of the alarm.

Alarm text

The alarm text describes the cause of the alarm.

The alarm text can contain output fields for current values. The values you can insert depend on the Runtime in use. The value is retained at the time at which the alarm status changes.

Alarm group

The alarm group bundles individual alarms.

Tooltip

You can configure a separate tooltip for each alarm; the user can display this tooltip in Runtime.

Trigger tag

Each alarm is assigned a tag as trigger. The alarm is output when this trigger tag meets the defined condition, e.g. when its state changes or it exceeds a limit.

Limit

Analog alarms indicate limit violations. Depending on the configuration, WinCC outputs the analog alarm as soon as the trigger tag exceeds or undershoots the limit value.

See also

Basics on alarm classes (Page 768)

Alarm states (Page 766)

Alarm groups (Page 775)

Overview of the alarm types (Page 759)

Overview of the alarm types (Page 760)

3.2.2 Configuring Alarms

3.2.2.1 Overview of alarm configuration tasks (Basic Panels)

Steps to configure alarms

Configuring alarms in WinCC involves the following steps:

1. Edit and create alarm classes
   You use the alarm class to define how an alarm will be displayed in runtime and to define the state machines for it.
2. Creating tags in the "HMI tags" editor
   - Configure the tags for your project.
   - You create range values for the tags.
3. Creating tags in the "HMI alarms " editor
   - Create custom alarms and assign these the tag to be monitored, alarm classes, alarm groups, and other properties.
   - You can also assign system functions or scripts to the alarm events.
4. Output of configured alarms
   To output configured alarms, configure an alarm view or an alarm window in the "Screens" editor.

Additional configuration tasks

Additional tasks may be necessary for configuring alarms, depending on the requirements of your project:

1. Creating alarm groups
   You assign the alarms of your project to alarm groups according to their association, such as by the cause of the problem (power failure) or source of the error (Motor 1).
2. Configuring Loop-In-Alarm
   A Loop-In-Alarm is configured in order to change to a screen containing relevant information on an alarm received.

3.2.2.2 Overview of alarm configuration tasks (RT Professional, RT Advanced, Panels, Comfort Panels)

Steps to configure alarms

Configuring alarms in WinCC involves the following steps:

1. Edit and create alarm classes
   You use the alarm class to define how an alarm will be displayed in runtime and to define the state machines for it.
2. Creating tags in the "HMI tags" editor
   - Configure the tags for your project.
   - You create range values for the tags.
1. Creating tags in the "HMI alarms " editor
   - Create custom alarms and assign these the tag to be monitored, alarm classes, alarm groups, and other properties.
   - You can also assign system functions or scripts to the alarm events.
2. Output of configured alarms
   To output configured alarms, configure an alarm view or an alarm window in the "Screens" editor.

Additional configuration tasks

Additional tasks may be necessary for configuring alarms, depending on the requirements of your project:

- Activating and editing system events
  You can import system events when you initially open the "System events" tab in the "HMI alarms" editor. On completion of the import, you can edit the system events.
- Activating and editing controller alarms
  For integrated operation of a project in STEP 7, specify the controller alarms to be displayed on your HMI device in the alarm settings.
- Creating alarm groups
  Assign the alarms of your project to alarm groups based on their relation, e.g. by error cause (e.g. "power failure"), or by error source (e.g. "Motor 1").
- Configuring Loop-In-Alarm
  Configure a Loop-In-Alarm in order to, after receipt of an alarm, change to a screen that contains your information about the selected alarm.

3.2.2.3 Configuring Alarm Classes

Creating alarm classes (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Create alarm classes in the "Alarm classes" tab of the "HMI alarms" editor. Some default alarm classes are already created for every project. You can create additional custom alarm classes. You can create up to 32 alarm classes.

Requirement

- The "HMI alarms" editor is open.
- The Inspector window is open.

Procedure

To create an alarm class, proceed as follows:

1. Click the "Alarm classes" tab.
   The predefined and existing custom alarm classes are displayed. The following table lists the predefined alarm classes:

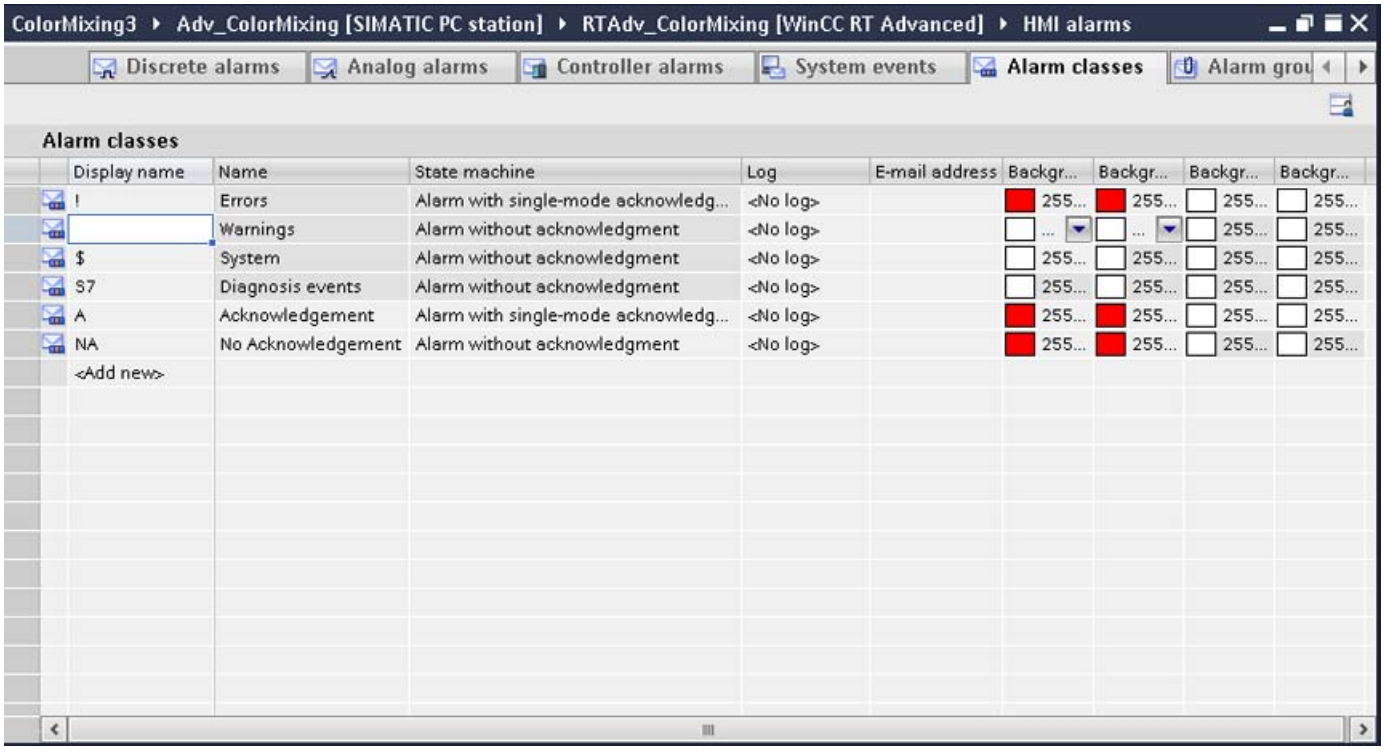

1. Double-click "<Add>" in the table.
   A new alarm class is created. Each new alarm class is automatically assigned a static ID. The properties of the new alarm class are shown in the Inspector window.
2. Configure the alarm class under "Properties > Properties >General" in the Inspector window.
   – Enter a "Name" and the "Display name".
   – Depending on the HMI device, you can also activate logging, or automatic sending of e-mails.
3. Define the state machines for the alarm class under "Properties > Properties > Acknowledgment" in the Inspector window.
4. Change the default text under "Properties > Properties > Status texts" in the Inspector window.
   This text indicates the status of an alarm in Runtime.
5. Change the default colors under "Properties > Properties > Colors" in the Inspector window. Depending on the HMI device, also change the flashing characteristics.

These settings define how alarms from this alarm class are displayed in Runtime.

Note

To display the alarm classes in color in Runtime, the "Use alarm class colors" option must be activated. In the project navigation, enable "Runtime settings > Alarms > General > Use alarm class colors" accordingly.

See also

Basics on alarm classes (Page 768)

Predefined alarm classes (Page 770)

Creating alarm classes (RT Professional)

Introduction

Create alarm classes in the "Alarm classes" tab of the "HMI alarms" editor. Some default alarm classes are already created for every project. You can create additional custom alarm classes. You can configure up to 256 alarm classes per project.

Requirement

- The "HMI alarms" editor is open.
- The Inspector window is open.

Procedure

To create an alarm class, proceed as follows:

1. Click the "Alarm classes" tab.
   A table of the pre-defined alarm classes is shown below:

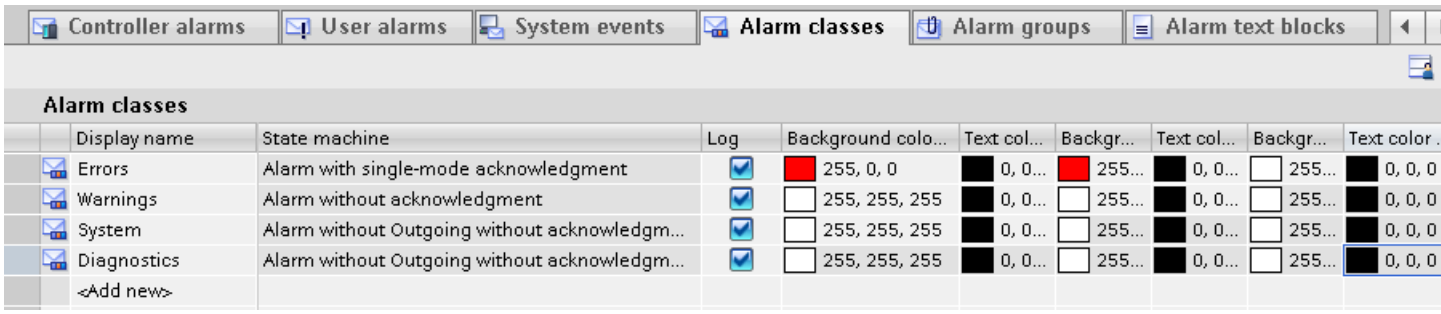

1. Double-click "<Add>" in the table.
   A new alarm class is created. Each new alarm class is automatically assigned a static ID.
   The properties of the new alarm class are shown in the Inspector window.
2. Configure the alarm class under "Properties > Properties >General" in the Inspector window:
   – Enter a "Name" and the "Display name".
   – Activate the"Log" check box if you want to log the alarms from this alarm class.
3. Select "Properties > Properties > Acknowledgment" in the Inspector window to define the state machines for the alarm class and alarm annunciator.
4. You can edit the default text under "Properties > Properties > Status" in the Inspector window.
   This text indicates the status of an alarm in Runtime.
5. You can also change the default background and foreground colors under "Properties > Properties > Colors" in the Inspector window.
   These settings define how alarms from this alarm class are displayed in Runtime.

Additional settings for alarm classes

Comments in Runtime

The user enters a comment for an incoming alarm in Runtime, as required. Using the buttons of the alarm view, users can show the comments in the "Historical alarm list" display. You can configure this function in the alarm class. Enter the user in the "User name" system block.

Assign comments to defined user

To assign the comments to the user logged on, proceed as follows:

1. Select the alarm class.
2. Enable "Properties > Properties > Acknowledgment > Settings > Assign comments to defined user" in the Inspector window.
   Any user can enter the first comment if a comment has yet to be entered.
   When the first comment has been entered, all other users have read-only access to it.

Use comment of the incoming alarm

Whenever the status of an alarm changes, for example, from "incoming" to "acknowledged" status, the alarm is output once again with another status. The user only assigns comments to one incoming alarm. To include comments for alarm output with the new status, proceed as follows:

1. Select the alarm class.
2. Enable "Properties > Properties > Acknowledgment > Settings > Use comment of the incoming alarm" in the Inspector window.

See also

Configuring alarm acknowledgment (Page 879)

Configuring acknowledgment of the alarm annunciator (Page 882)

Basics on alarm classes (Page 768)

Predefined alarm classes (Page 770)

Using common alarm classes (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Project-wide alarm classes are displayed under "Shared data > Alarm classes" in the project navigation. Create additional common alarm classes in WinCC.

Controller alarms are assigned to common alarm classes. If you work with controller alarms, common alarm classes are automatically loaded in WinCC.

Requirements

- You have created a project.

Creating project-wide alarm class

To create a project-wide alarm class, proceed as follows:

1. Double-click "Shared data > Alarm classes" in the project navigation.
   The "Project-wide message classes" editor opens in the work area.
2. To create a project-wide message class, double-click in the first empty line of the table editor.
3. Specify the name and display name of the alarm class.
4. Select "With acknowledgment" if the alarm class requires acknowledgment.

Assign alarms to a common alarm class

Proceed as follows to assign an analog or discrete alarm to a common alarm class:

1. In the "HMI alarms" editor, select the alarm that you want to assign to the common alarm class.
2. Click "General" in the Inspector window.
3. Click "Shared data > Alarm classes" in the project navigation.
4. Select the common alarm class in the detail view.
5. Drag-and-drop the alarm class to the "Alarm class" selection box or "Alarm class" column in the work area of the Inspector window of the alarm.

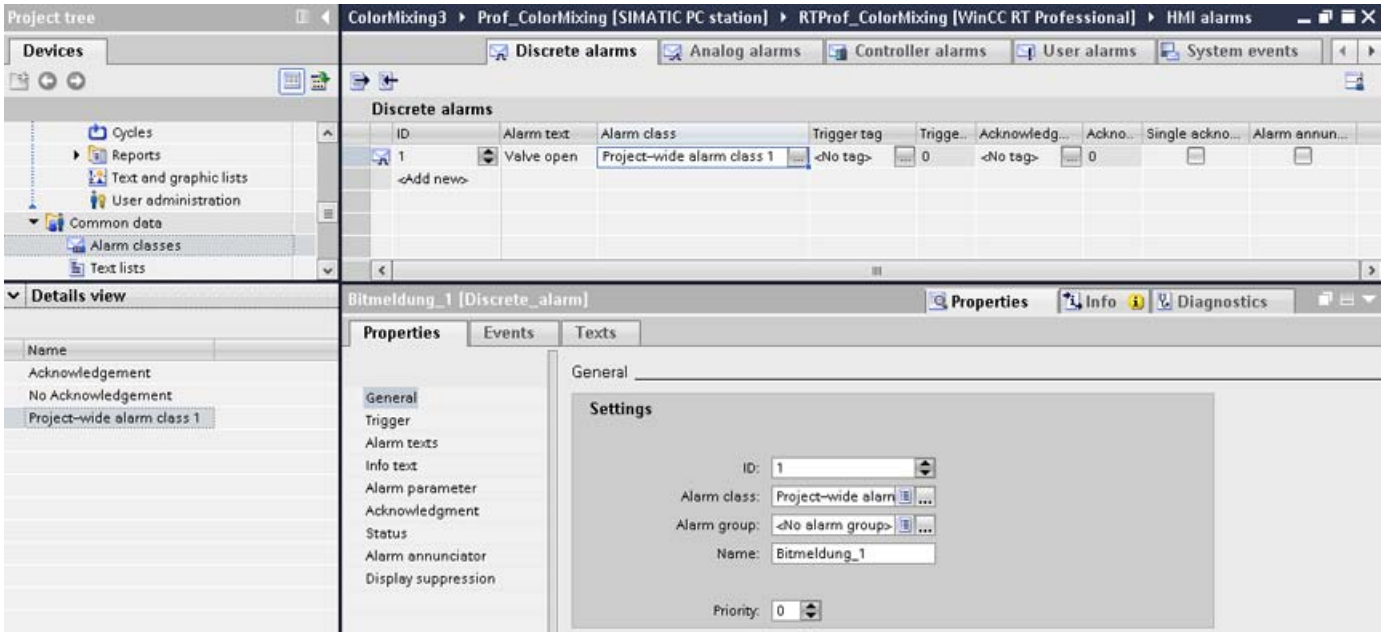

The system creates a new alarm class that is linked to the common alarm class. The new name of the common alarm class is displayed in the "Alarm class" area of the Inspector window of the alarm. The display name of the common alarm class is activated for the alarm class in your project.

6. Click the "Alarm Class" tab.
7. Check to see whether the display name of the common alarm class is displayed in WinCC.
8. You can change the object name of the alarm class.

Alternative procedure

To assign an alarm to a common alarm class, proceed as follows:

1. Click the "Alarm Classes" tab.
   The alarm classes are displayed.
2. Click "Shared data > Alarm classes" in the project navigation. The following table shows the common alarm classes:

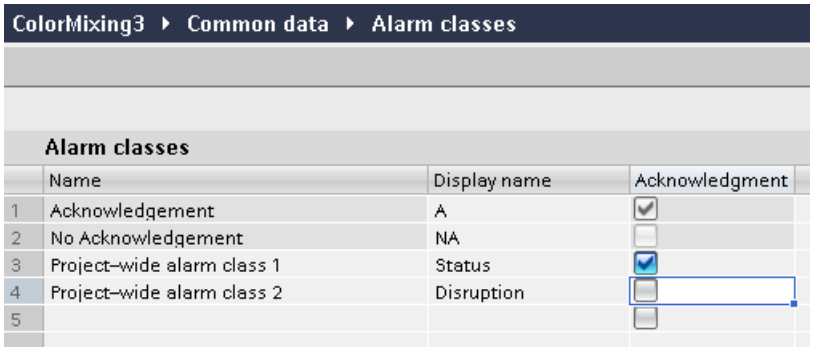

	Name	Display name	Acknowledgment
1	Acknowledgement	A	☑
2	No Acknowledgement	NA	☐
3	Project-wide alarm class 1	Status	☑
4	Project-wide alarm class 2	Disruption	☐
5			☐

3. Select the common alarm class in the detail view.
4. Drag-and-drop the alarm class to the "Alarm classes" tab of the "HMI alarms" editor
   The system creates a new alarm class that is linked to the common alarm class. The new name of the new alarm class in your project is displayed in the "Alarm class" tab.
5. Open the tab with the alarm procedure for the required alarm.
6. Under "General" in the Inspector window, select the alarm and the alarm class linked to the common alarm class.

Note

Deleting a common alarm class

If you delete a common alarm class, the linked alarm class is maintained. If necessary, link the alarm class with another common alarm class or use it as WinCC alarm class.

See also

Basics on alarm classes (Page 768)

3.2.2.4 Configuring Alarm Groups

Configuring alarm groups (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Create alarm groups on the "Alarm Groups" tab in the "HMI alarms" editor. An alarm group is a compilation of single alarms. You assign alarms in an alarm group by association, such as cause of the problem or source of the error. If you acknowledge an alarm from this alarm group in Runtime, all other alarms in the alarm group are acknowledged automatically.

Requirement

- You have created a project.
- The "HMI alarms" editor is open.
- The Inspector window is open.

Creating a new alarm group

1. Click the "Alarm Groups" tab.
   The existing alarm groups are displayed.

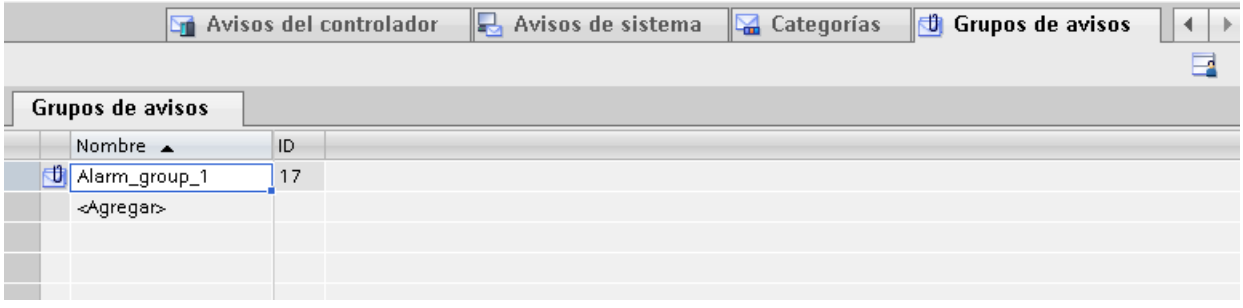

2. In the work area of the table, double-click "<Add>" in the first free row.
   A new alarm group is created.
3. You can overwrite the proposed "Name".

Result

An alarm group is created. For the group acknowledgment of alarms in Runtime, assign the associated alarms that require acknowledgment to an alarm group.

See also

Alarm groups (Page 775)

Configuring alarm groups (RT Professional)

Introduction

Create custom alarm groups on the "Alarm Groups" tab of the "HMI alarms" editor. The alarm groups from alarm classes are already created on a separate tab. You acknowledge alarms from an alarm group together in Runtime. Configure a corresponding group acknowledgment button for the alarm group, including an additional button for resetting the acknowledgment bit of the acknowledgment tag.

Requirements

- You have created a project.
- The "HMI alarms" editor is open.
- The Inspector window is open.

Creating a new alarm group

To create an alarm group, proceed as follows:

1. Click the "Alarm Groups" tab.
   The existing alarm groups are displayed.

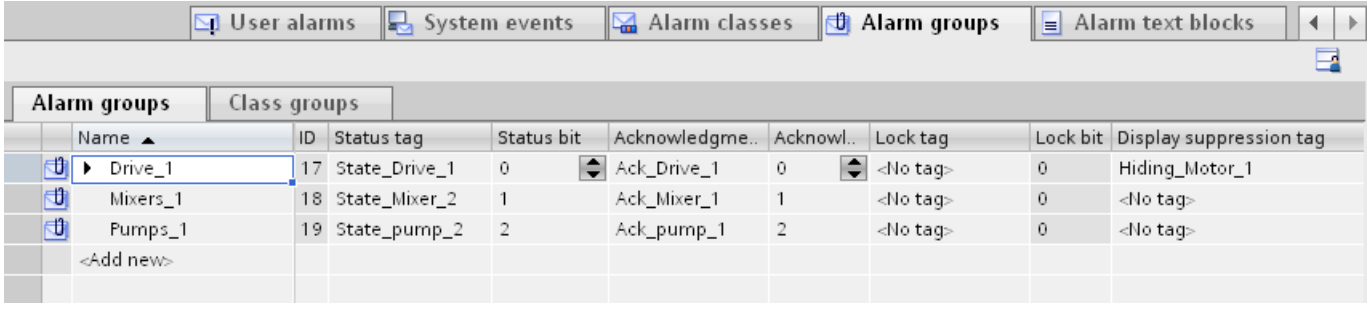

2. Double-click "<Add>" in the table.
   A new alarm group is created.
   The properties of the new alarm group are shown in the Inspector window.
3. In the Inspector window, click "Properties > Properties > General" and enter a name for the alarm group.
4. In the Properties window, click "Properties > Properties > Tags" to specify the following tags and bits:
   - Status tag for visualizing the alarm states of the alarm group
   - Acknowledgment tag for acknowledging an alarm group
   - Lock tag for visualizing the locked status of an alarm group
   - Display suppression tag for filtering alarms in Runtime by plant status

Creating a child alarm group

To create a child alarm group, proceed as follows:

1. Click the "Alarm Groups" tab.
2. In the work area or in the detail view, select the alarm groups you want to insert as a child group.
3. Drag and drop the alarm groups onto another alarm group.
   The selected alarm groups become a child group in the hierarchy of this alarm group.

If you want to create a parent group for a hierarchy of alarm groups, drag and drop the entire hierarchy to this alarm group.

Alternative procedure

1. Click the "Alarm Groups" tab.
2. Select the alarm groups for which you want to create a subgroup.
3. In the cell on the left of the selected row, open the shortcut menu and select "Insert child element".
   A new child alarm group is created.

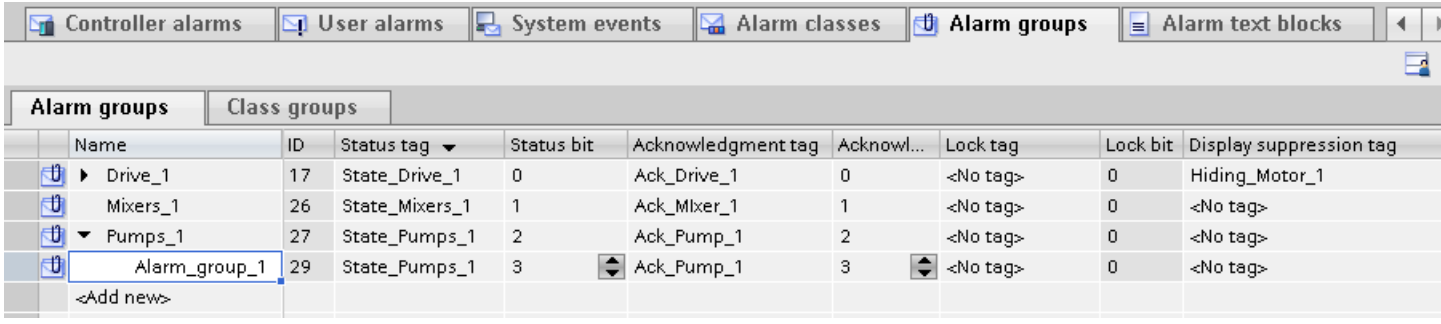

See also

Alarm Group Basics (Page 775)

Hierarchical alarm groups (Page 777)

Tags of an alarm group (Page 778)

Configuring Alarm Display Suppression (Page 860)

Tag Bit Assignments (Page 1018)

3.2.2.5 Configure alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Configuring Analog Alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Analog alarms indicate limit violations. For example, if the speed of a motor drops below a certain value, an analog alarm is triggered.

Requirements

- The "HMI alarms" editor is open.
- The Inspector window is open.
- You have created the required alarm classes and alarm groups.

Procedure

To configure an analog alarm, proceed as follows:

1. Click the "Analog alarms" tab.
2. To create a new analog alarm, double-click in the table on "<Add>".
   A new analog alarm is created.
3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   – Enter an alarm text as event text.
     Format the text character-for-character using the shortcut menu.
     Using the shortcut menu, you can insert output fields for HMI tags, or text from text lists.
   – You can renumber the alarm.
   – Select the alarm class and the alarm group, if necessary.

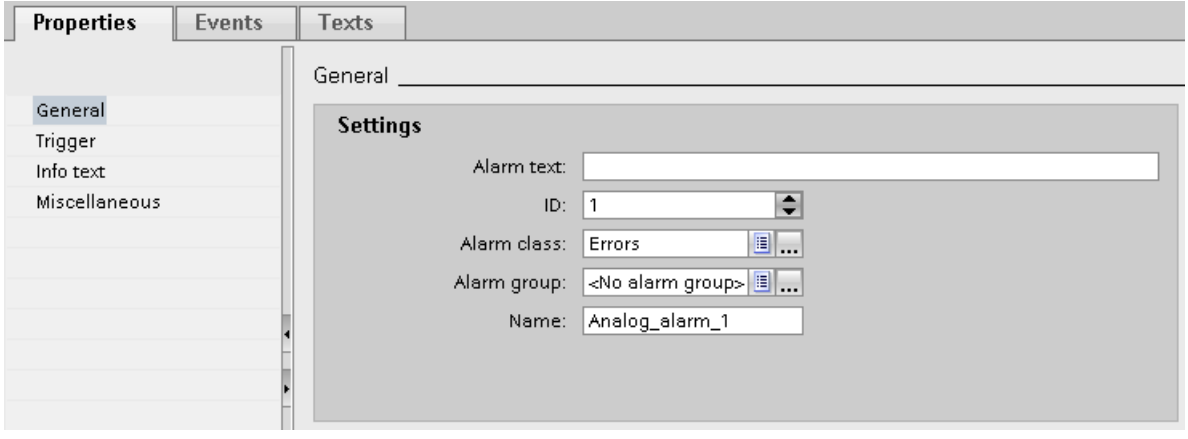

4. Configure the tag that triggers the alarm under "Properties > Properties > Trigger > Settings".
   Do not use trigger tags for anything else.

Configure limit values for an analog alarm

1. In the Inspector window, click the button under "Properties > Properties > Trigger > Limit > Value".
   - To use a constant as limit value, select "Constant".
     Enter the required limit value.
   - To use a tag as limit value, select "HMI tag".
     The button appears. Use this button to select the tag you want to use.

   Note

   If the required tag does not exist in the selection list yet, create it in the object list and change its properties later.

2. Select the mode:
   - "Higher": The alarm is triggered when the limit is exceeded.
   - "Lower": The alarm is triggered when the limit is undershot.

Optional settings for analog alarms

Setting the delay time

To set the delay time, proceed as follows:

- Enter a time period in the Inspector window under "Properties > Properties> Trigger > Settings > Delay".
  The alarm is only triggered when the trigger condition is still present after the delay time has elapsed.

Setting the deadband

Note

If a process value fluctuates around the limit, the alarm associated with this fault may be triggered multiple times. To prevent this from happening, configure a deadband or delay time.

To enter the deadband, follow these steps:

1. Under "Properties > Properties> Trigger > Deadband > Mode", select the change in alarm status for which the deadband is to be taken into account.
2. Enter a constant value under "Value".
3. To define the deadband value as a percentage of the limit, set the "in %" check box.

Automatic reporting

To print the alarms continuously:

- Select the "Activate" option under "Properties > Properties > Miscellaneous > Report."

Note

Device dependency

Reporting is not available for all HMI devices.

Creating a tooltip

To configure a tooltip for the alarm, follow these steps:

- Select "Properties > Properties > Tooltip" in the Inspector window and enter your text.

Configuring event-driven tasks

To configure event-driven tasks, such as a loop-in alarm, follow these steps:

1. Select the analog alarm.
2. In the Inspector window, select "Properties > Events > Loop-In-Alarm.
3. Select, for example, the system function "ActivateScreen" or a different system function that is be called for Loop-In-Alarm.
4. Select the parameters required for the selected system function.
   With "ActivateScreen“, the screen is to be called.

See also

Analog alarms (Page 762)

Output of a tag value in the alarm text (Page 800)

Configuring trigger for alarm acknowledgment (Page 877)

Configuring reporting of alarms (Page 873)

Formatting alarm text (Page 798)

Output of texts from a text list in an alarm (Page 801)

Configuring discrete alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Discrete alarms triggered by the PLC indicate status changes in a plant. They indicate the opened or closed state of a valve, for example.

The following sections describes the configuration procedures in the "HMI alarms" editor. You can also configure discrete alarms in the "HMI tags" editor.

Requirements

- The "HMI alarms" editor is open.
- The Inspector window is open.
- You have created the required alarm classes and alarm groups.

Procedure

To configure a discrete alarm, proceed as follows:

1. Open the "Discrete alarms" tab.
2. To create a new discrete alarm, double-click in the work area on "<Add>".
   A new discrete alarm is created.
3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   – Enter an alarm text as event text.
     Use the functions of the shortcut menu to format the text on a character-by-character basis, or to insert output fields for HMI tags, or texts from the text lists.
   – You can renumber the alarm.
   – Select the alarm class and the alarm group, if necessary.
4. In the Inspector window, select the tag and the bit that triggers the alarm under "Properties > Properties > Trigger". Note the following information:
   – Use the data types "Int" or "UInt" to select an HMI tag.
   – When you select a PLC tag, use the data types "Int" or "Word". The input and output area of a PLC tag is unsuitable as trigger.
   – Use trigger tag bits only for alarms.
   – Do not use trigger tags for anything else.
   – If you want to acknowledge the alarm via the PLC, use this tag also as PLC acknowledgment tag.

Note

Note the method used to count bits in the utilized PLC when specifying the bit. For more information, refer to the "Communication" section in the PLC Online Help.

Note

If the object does not yet exist in the selection list, create it directly in the object list and change its properties later.

Status-dependent alarm texts

To display a different text independent of the alarm status, link a text list to the alarm text. You control the text list with a tag.

Additional settings for discrete alarms

Creating a tooltip

To configure a tooltip for the alarm, follow these steps:

- Enter your text under "Properties > Properties > Tooltip".

Configuring event-driven tasks

To configure event-driven tasks, such as a loop-in alarm, follow these steps:

1. Select the discrete alarm.
2. Select "Properties > Events" in the Inspector window and configure a new function list for the relevant event.

See also

Discrete alarms (Page 762)

Configuring trigger for alarm acknowledgment (Page 877)

Configuring reporting of alarms (Page 873)

Output of texts from a text list in an alarm (Page 801)

Formatting alarm text (Page 798)

Formatting alarm text (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

The alarm texts can be formatted or special characters inserted in the alarm texts to emphasize specific events or processes.

Requirement

- The "HMI alarms" editor is open.
- An alarm has been created.

Formatting alarm texts

To format an alarm text, proceed as follows:

1. Select the alarm to edit.
2. In the Inspector window, select the characters to format under "Properties > Properties > General > Alarm text".
3. Select the formatting from the shortcut menu, e.g. "Underscored" or "Uppercase".
   The selected characters are displayed in Runtime with the selected formatting.

Note

It is not possible to underline tags and text list entries.

Note

Displaying special characters in alarm texts

When configuring alarm texts, a fixed character set is used in the Engineering System. This character set allows you to use numerous special characters in alarm texts.

Language-dependent fonts are used in runtime to display the texts, for example MS PGothic, SimSun. The fonts used in runtime do not support all special characters. As a result, some special characters are not displayed in runtime.

Removing format settings

To remove all text formats, proceed as follows:

1. In the Inspector window, select the characters whose formatting you want to remove in the alarm text.
2. Select "Delete formatting characters" from the shortcut menu.
   The selected characters are displayed in unformatted notation in Runtime.

See also

Overview of alarm configuration tasks (Page 783)

Configuring Analog Alarms (Page 794)

Configuring discrete alarms (Page 796)

Output of a tag value in the alarm text (Page 800)

Output of texts from a text list in an alarm (Page 801)

Output of a tag value in the alarm text (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

In WinCC, you can insert output fields into the alarm text which display the content of tags.

Requirements

- The "HMI alarms" editor is open.
- The alarm is selected.

Output of a tag value in the alarm text

To insert an output field for a tag value in the alarm text, proceed as follows:

1. Place the cursor onto the required position in the event text.
2. Select "Insert tag output field" in the shortcut menu.
3. Open the object list under "Tag" and select a tag.
   You can also create the tag in the object list.
4. Under "Format", specify the length of the output field and the format for tag value output in the alarm text.
   Configure an output field of sufficient size. Otherwise, the tag content is not output to the full extent in the alarm.
5. Click to save your entries.

WinCC inserts a placeholder for the output field into the alarm text: "<tag: n, [tag name]>" whereby n = text string length.

Editing output field properties

To edit the properties of an output field, proceed as follows:

- Double-click on the output field in the alarm text and edit the settings.

Deleting an output field from the alarm text

To delete an output field from the alarm text, proceed as follows:

- Select the output field in the alarm text and then select the "Delete" command from the shortcut menu.

Note

The sequence of the tag output fields in the alarm text depends on the language. The sequence of the Runtime language is used for logging alarms to a CSV log file.

Changing the tag of an output field in one language causes the modified output field to appear at the end of the alarm text in all other languages. This changes the sequence of the output fields in the log.

Device dependency

Logging is not available for all HMI devices.

See also

Configuring Analog Alarms (Page 794)

Formatting alarm text (Page 798)

Output of texts from a text list in an alarm (Page 801)

Output of texts from a text list in an alarm (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Text lists are used to create dynamic alarm texts. For example, you can configure the output of both states of a discrete alarm in an alarm text.

Insert a corresponding output field in the alarm text and specify the tag that returns the search key for the text list.

Requirements

- A text list is created.
- The "HMI alarms" editor is open
- The alarm is selected

Output of a text list value in an alarm text

To insert an output field for a value from a text list in the alarm text, proceed as follows:

1. Place the cursor at the required position in the alarm text.
2. Select "Insert text list output field" command from the shortcut menu.
3. Select the text list under "Text list".
4. Open the object list under "Tag" and select the tag that returns the search key for the text list.
   You can also create the tag in the object list.

5. Define the length of the output field under "Format".
   Configure an output field of sufficient size so that the complete text from the text list is displayed in the alarm.
6. Click on the icon to save your entries.
   WinCC inserts a placeholder for the output field into the alarm text:
   ""<textlist: n, [textlist name]>" whereby n = text string length.

Editing output field properties

To edit the properties of an output field, proceed as follows:

- Double-click on the output field in the alarm text and edit the settings.

Deleting text list from the alarm text

To delete an output field from the alarm text, proceed as follows:

- Select the output field in the alarm text and then select the "Delete" command from the shortcut menu.

See also

Formatting alarm text (Page 798)

Configuring Analog Alarms (Page 794)

Configuring discrete alarms (Page 796)

Output of a tag value in the alarm text (Page 800)

Editing system events (RT Advanced, Panels, Comfort Panels)

Editing system events

When you open the "System events" tab in the "HMI alarms" editor, WinCC prompts you to import or update the system events. Import the system events and specify the display period. You cannot delete or create new system events. You can only edit the alarm text with system events.

Requirements

- You have imported the system events.

Specifying the display period

To determine the display period, proceed as follows:

1. Double-click "Runtime settings" in the project navigation.
2. Enter a value under "Alarms > System events > Display period".
   This value specifies a time in seconds during which the system events are displayed on the HMI device. If you want to display system events permanently, enter "0".

Changing the alarm text

To change the alarm text of a system event, follow these steps:

1. Open the "HMI alarms" editor and then click the "System events" tab.
2. Select the system event that you want to edit.
1. Change the alarm text under "Properties > Properties > Properties > General" in the Inspector window.

Note

If you change an alarm text for a system event that contains a placeholder, leave the number of placeholders unchanged. A placeholder may be a "%1".

Configuring event-driven tasks

The "Incoming" event is available for certain system events, depending on the HMI device. To configure event-driven tasks, proceed as follows:

1. Select the system event.

2. Select "Properties > Properties > Incoming" in the inspector window and configure a function list.

See also

Overview of alarm configuration tasks (Page 784)

System events (Page 764)

Editing controller alarms (RT Advanced, Panels, Comfort Panels)

Introduction

Controller alarms are configured in STEP 7. Controller alarms are available in WinCC running in a STEP 7 environment.

In WinCC you only edit the properties of controller alarms that are specific to an HMI device. Different properties of a controller alarm can be edited, depending on the HMI device and PLC.

You can filter the displayed controller alarms for your project using the display classes that were configured on the PLC for controller alarms.

Note

When "Automatic update" is set for controller alarms for the connection to the S7-1500 controller, you cannot edit the properties of the respective controller alarms. The controller alarms are not displayed in the "HMI alarms" editor in this case.

Requirements

- The connection was established to the PLC.
- Alarms were configured in STEP 7.

Editing controller alarms

Proceed as follows to edit controller alarms:

1. Double-click "HMI alarms" in the project navigation.
   The "HMI alarms" editor is opened.
2. Open the "Controller alarms" tab.
3. Select the alarm that you want to change in the work area.
4. Make your changes. Depending on the HMI device, it may not be possible to display or enable some of the controller alarm properties.

Editing SIMATIC S7 alarms

You can edit these controller alarms in the STEP 7 alarm configuration system. Proceed as follows:

1. Select the alarm that you want to change in the work area.
2. Select "Open in alarm editor of the PLC" from the shortcut menu.
   The STEP 7 alarm configuration system is opened.
3. Make your changes.

Filtering the controller alarm using display classes

To filter controller alarms by display classes, proceed as follows:

1. Click "Runtime settings > Alarms" in the project navigation under your HMI device.
   One or several connections to a PLC are shown in "Contoller alarms".
2. Select the display classes whose controller alarms you want to display for the connection.

Note

Controller alarms after a basic panel device exchange

For the controller alarms (PLC monitoring alarms, program alarms or system alarms) to be displayed after a device replacement ("Device view > Change device") from a Basic Panel to a Comfort Panel, Mobile Panel or WinCC RT Advanced, you need to activate the "Display classes" setting for the respective connections under "Runtime settings > Alarms > Controller alarms".

Result

Only those controller alarms whose display class you have enabled for your HMI device will be displayed on the "Controller alarms" tab of the "HMI alarms" editor.

See also

Controller alarms (Page 762)

Enabling system-defined controller alarms (RT Advanced, Panels, Comfort Panels)

Enabling system-defined controller alarms

To activate system-defined controller alarms of the type "SIMATIC S7 diagnostic alarm" for your HMI device, follow these steps:

1. Select "Runtime settings > Alarms" in the project navigation.
   The alarm settings open.
2. Under "System events" select the check box "S7 diagnostic alarms (number)".
3. To display the system events with alarm text, activate "Show text".
4. To display these alarms in Runtime, configure an alarm view or an alarm window that displays alarms from the "Diagnosis Events" alarm class.

   Note

   For HMI devices with small display units, specify to display only the alarm number for system-defined controller alarms.

   Note

   The procedure described above can only be applied to connections with SIMATIC S7-300/400 controllers for system-defined controller alarms of the type "SIMATIC S7 diagnostic alarm".

See also

System-defined controller alarms (Page 765)

3.2.2.6 Configure alarms (RT Professional)

Configuring alarm blocks (RT Professional)

Alarm Text Block Basics (RT Professional)

Introduction

An alarm display is made up of alarm text blocks. Each alarm text block corresponds to one column in the table in the alarm view.

There are three groups of alarm text blocks:

- System blocks
  System blocks contain system data, such as date, time, alarm number and status.
- User text blocks
  User text blocks contain the alarm text with the description of the cause of a fault and additional text with information, for example, the location of a fault.
- Parameter blocks
  Parameter blocks are used to link the alarms to process values, such as current fill levels, temperatures or speeds. You can configure up to 10 parameter blocks per alarm.

Configuring Alarm Text Blocks

You enable and edit the alarm text blocks on the "Alarm text blocks" tab. To display alarm text blocks in an alarm view, select the alarm text blocks to be displayed and logged from the enabled alarm text blocks in the "Screens" editor.

Text length

The maximum text length of a user text block is 255 characters.

The text length of a parameter block can be configured as follows:

- Controller alarms: up to 32 characters
  Set the length of the parameter block to 32 characters to ensure that all characters will be displayed.
- Other alarm procedures: up to 255 characters

Note

"Status" system block

The status texts of the alarms are displayed in the "Status" system block. This system block must be configured so that it is long enough to display the status texts in full.

Flashing

An alarm to be acknowledged or single alarm text blocks can be displayed as flashing in Runtime. They have to meet the following requirements:

- The "Flashing" property must be enabled in the alarm class of the alarm. You configure this property on the "Alarm classes" tab.
- The "Flashing" option is enabled in the "Alarm text block" area under "General" in the Inspector window for alarm text blocks that should flash.

Alignment

On the "Alarm text blocks" tab, you define whether the alarm text block should be right-justified, left-justified or centered in an alarm view column.

Format

For certain alarm text blocks, you can choose between different display formats.

See also

Setting up flashing alarm text blocks (Page 808)

Configuring Alarm Text Blocks (RT Professional)

Introduction

An alarm is made up of configured alarm text blocks. You enable the alarm text blocks for an HMI device on the "Alarm text blocks" tab. Then you select, from the enabled alarm text blocks for each alarm view, which enabled alarm text blocks are to be displayed and logged.

Requirements

- The "HMI alarms" editor is open.
- The "Alarm text blocks" tab is selected.

Activating alarm text blocks

1. Select the "System blocks", "User text blocks" or "Parameter blocks" tab for the alarm text blocks.
2. Select the alarm text block to be displayed on your HMI device.
3. In the Inspector window, select "Properties > Properties > General > Blocks > Enable".

4. In the Inspector window under "Properties > Properties > General > Settings", define the display name of the alarm text block.

   Note

   The write-protected names of the alarm text blocks stored in the system are displayed in the "Name in the Alarms editor" column under "Properties > Properties > General > Settings". This column is available for the "User text blocks" and "Parameter blocks" tags.

   Based on the "Name in the Alarms editor" column, you recognize the relationship between the display names of the alarm blocks in Runtime and the designations in the "Alarms" editor, even if the display names of the alarm text blocks were changed.

5. Under "Properties > Properties > General > Blocks" in the Inspector window, define the text length and alignment of the alarm text block in the alarm view column.
6. In the Inspector window under "Properties > Properties > Settings > Format", select the format of the alarm text block. This option is not available for every alarm text block.

Result

The alarm text blocks are enabled and displayed in the Inspector window of the alarm view in the "Screens" editor.

Disabling alarm text blocks

1. Select the tabs for the alarm text blocks that you want to disable.
2. Disable the alarm text blocks that you no longer want to use for alarm view and alarm logging.
3. Repeat the process for every alarm text block that you no longer want to display.

See also

Description of the System Blocks (Page 957)

Setting up flashing alarm text blocks (Page 808)

Setting up flashing alarm text blocks (RT Professional)

Introduction

Whether or not an alarm text block flashes in Runtime depends on the alarm class of the alarm. The alarm class determines if the alarms it contains flash when acknowledgment is pending.

Requirements

- The alarms blocks are enabled for the HMI device.
- The alarms blocks are enabled for the respective alarm.

- The alarm is assigned to an alarm class with a state machine, which specifies whether or not the contained alarms are to flash.
- The alarm text blocks are activated in the alarm view.
- The "Alarm text blocks" tab is selected.

Procedure

To configure flashing for alarm text blocks in Runtime, proceed as follows:

1. Select the alarm text block to flash in Runtime from the subordinate tabs on the "Alarm text blocks" tab.
2. In the Inspector window, select "Properties > Properties > General > Blocks > Flash".
3. Repeat this step for all the alarm text blocks that you want to flash in Runtime.

Result

These alarm text blocks flash in Runtime when the alarm is queued for acknowledgment.

See also

Alarm Text Block Basics (Page 806)

Configuring Alarm Text Blocks (Page 807)

Configuring analog alarms (RT Professional)

Configuring analog alarms (RT Professional)

Introduction

In the "HMI alarms" editor, you can configure analog alarms and define their properties. You can also configure analog alarms in the "HMI tags" editor.

Requirement

- The "HMI alarms" editor is open.
- The Inspector window is open.

Procedure

To configure an analog alarm, proceed as follows:

1. Click the "Analog Alarms" tab.
2. To create a new analog alarm, double-click in the table on "<Add>".
   A new analog alarm is created.

3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   - You can always change the object name of the alarm.
   - Select the alarm class and the alarm group, if necessary.
4. To display the CPU number or PLC that sends the alarm, enter this under "Connection to the PLC". This option is only active if you have enabled the "CPU/PLC number" alarm text block on the "Alarm text blocks > System blocks" tab.
5. In the Inspector window, select the tag that triggers the alarm under "Properties > Properties > Trigger". Do not use trigger tags for anything else.
6. To configure the alarm text, select "Properties > Properties > Alarm texts".
   - To enter the alarm text, select "Alarm text".
   - To create additional alarm texts, select "Additional texts".
   - Use the functions of the shortcut menu to format text on a character-by-character basis and to insert output fields for system or alarm parameters.

   Note

   You can only enter additional text in the rows that you have enabled as user text blocks on the "Alarm text blocks" tab.

Defining limits

1. Select the analog alarm to which you want to assign the limits.
2. At the bottom of the work area, click <Add> under "Limits for analog alarm".

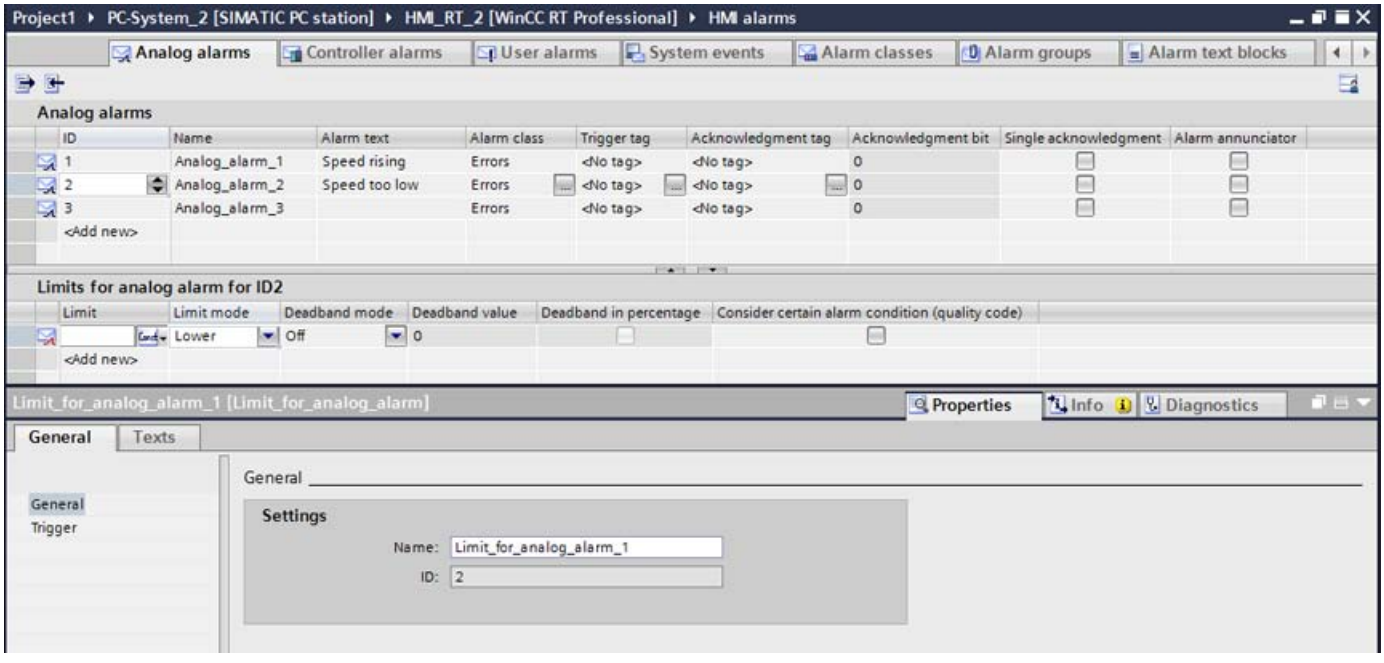

3. To use a **constant** as limit:
   - In the Inspector window, click the button under "Properties > Properties > Trigger > Value" and select "Constant".
   - Enter the required limit value.
4. To use a **tag** as limit:
   - In the Inspector window, click the button under "Properties > Properties > Trigger > Value" and select "HMI tag".

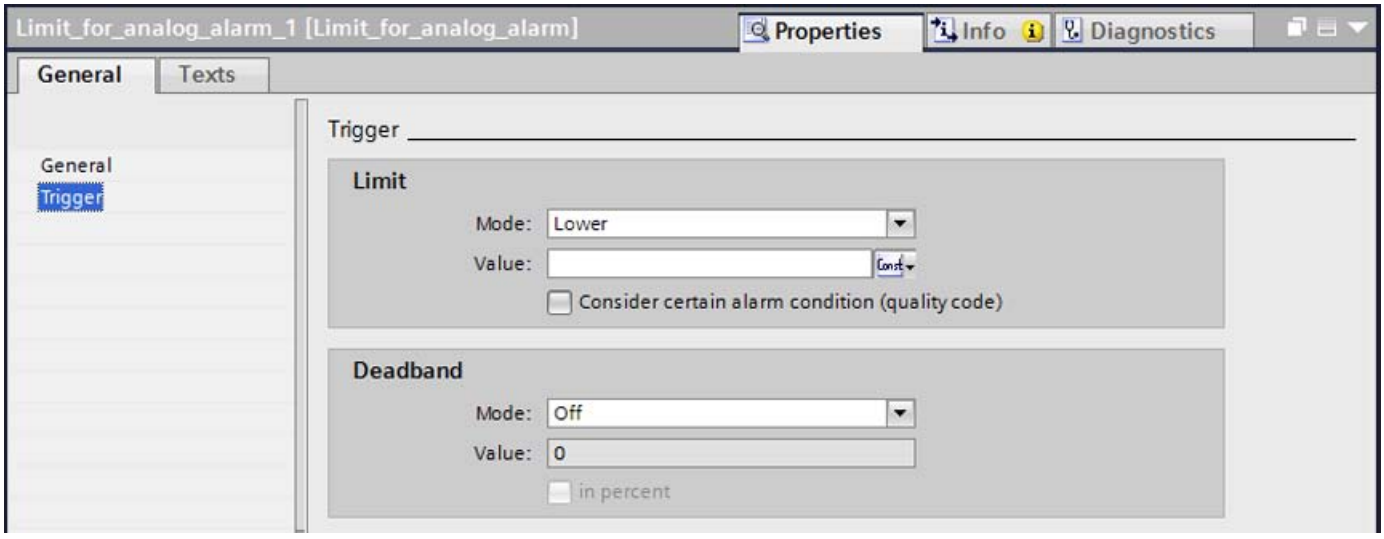

   - Click the button. The object list opens.
   - Select the tag.
5. Select the trigger mode in the "Mode" field.
   - "Equal to limit": The alarm is triggered when the limit is reached.
   - "Not equal to limit": The alarm is triggered when the limit is no longer reached.
   - "High limit violation": The alarm is triggered if the limit is exceeded.
   - "Low limit violation": The alarm is triggered if the limit is undershot.
6. You can create additional limits for the alarm, if necessary.

Note

If the object included in the selection does not yet exist, create it in the object list and change its properties later.

Note

If a process value fluctuates around the limit, the alarm associated with this fault may be triggered multiple times. In this case, configure a deadband or delay time.

Optional settings for analog alarms

Setting the delay time

To display an alarm after a delay, proceed as follows:

1. Select the analog alarm.
2. In the Inspector window, select "Properties > Properties > Trigger".
3. Enter a time period under "Delay". The trigger condition must be fulfilled within this time for the alarm to be triggered.

Setting the deadband

Proceed as follows to enter a tolerance range for the arrival or departure of an alarm:

1. Select the limit to which the deadband is to be assigned.
2. In the Inspector window, select "Properties > Properties > Trigger".
3. Under "Deadband > Mode", select the change to the alarm status to include in the evaluation of the deadband.
4. Enter a constant under "Deadband > Value".
5. To define the deadband value as a percentage of the limit, activate the "%" check box.

Saving the status of the alarm

To save the alarm states to a tag, proceed as follows:

1. Select the analog alarm.
2. Under "Properties > Properties > Status" in the Inspector window, select the tag and the bit that triggers saving of the incoming/outgoing alarm states and the acknowledgment request.

Setting the priority

To set the priority, proceed as follows:

1. Select the analog alarm.
2. Select a value between 0 and 16 under "Properties > Properties > General > Priority" in the Inspector window, .
   If you filter the alarm view by priority, the alarm with priority 0 will appear at the top.

Creating an infotext

1. Select the analog alarm.
2. Select "Properties > Properties > Tooltip" in the Inspector window and enter a text.

Configuring event-driven tasks

To configure event-driven tasks, such as a loop-in alarm, follow these steps:

1. Select the analog alarm.
2. Select "Properties > Events" in the Inspector window and configure a new function list for the relevant event.

See also

Configuring acknowledgment of the alarm annunciator (Page 882)

Configuring reporting of alarms (Page 875)

Configuring alarm acknowledgment (Page 879)
Parameter output in the analog alarm (Page 813)
Configuring loop-in alarms (Page 829)
Alarms in the "HMI tags" Editor (Page 831)
Configuring Alarm Display Suppression (Page 860)
Overview of alarm configuration tasks (Page 784)

Parameter output in the analog alarm (RT Professional)

Introduction

To display system or alarm parameters, insert an appropriate output field in the analog alarm.

- "Limit", "Deadband" or "Current Value" can be selected as alarm parameters.
- The following system blocks can be selected as system parameters:
  - User Name
  - Computer name
  - Comments
  - Application Name

Requirement

- The "HMI alarms" editor is open.
- The analog alarm is selected.

Procedure

To output parameters in alarm texts, proceed as follows:

1. Place the cursor at the required position in the alarm text.
2. To output an alarm parameter, select "Insert parameter output field in analog alarm" from the shortcut menu.
   -OR-
   Enable the output of system parameters by selecting "Insert system block output field" from the shortcut menu.
3. Specify the length, number of decimal places, tolerance, and alignment of the output field for alarm parameters.
   To display leading zeros in the output field, enable "Leading zeros".
4. Click to save your entries.

WinCC inserts a placeholder for the output field into the alarm text:

- For alarm parameters: "<tag: n, [###]>" whereby n = text length in characters.

- For system parameters; "<name of the system parameter>"

Note

If you test a project using the simulator, the output field values are not output in the alarm texts.

Deleting parameter from the alarm text

To delete an output field from the alarm text, proceed as follows:

- Select the output field in the alarm text and then select the "Delete" command from the shortcut menu.

Note

The sequence of output fields in the alarm text depends on the language. The sequence of the Runtime language is used for logging alarms to a CSV log file.

Changing the tag of an output field in one language causes the modified output field to appear at the end of the alarm text in all other languages. This changes the sequence of the output fields in the log.

Configuring discrete alarms (RT Professional)

Configuring discrete alarms (RT Professional)

Introduction

Discrete alarms triggered by the PLC indicate status changes in a plant. They indicate the opened or closed state of a valve, for example.

The following sections describes the configuration procedures in the "HMI alarms" editor. You can also configure discrete alarms in the "HMI tags" editor.

Requirements

- The "HMI alarms" editor is open.
- The Inspector window is open.

Procedure

To configure a discrete alarm, proceed as follows:

1. Click the "Discrete alarms" tab.
2. To create a new discrete alarm, double-click on "<Add>" in the table.
   A new discrete alarm is created.

3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   - You can always change the object name of the alarm.
   - Select the alarm class and the alarm group, if necessary.
   - To display the CPU number or PLC that sends the alarm, enter these in the "Connection" area. This option is only active if you have enabled the "CPU/PLC number" alarm text block on the "Alarm text blocks" tab.
4. Select "Properties > Properties > Trigger" in the Inspector window to select the tag and the bit that triggers the alarm. Note the following information:
   - Use data types "Bool" "USInt", "UInt", or "UDInt".
   - Use trigger tag bits only for alarms.
   - Do not use trigger tags for anything else.
   - The input and output area of a PLC tag is unsuitable as trigger.

Note

If the object does not yet exist in the selection list, create it directly in the object list and change its properties later.

Note

The trigger bit is not incremented automatically if you generate a discrete alarm by copying an alarm text to a new table row. Avoid any redundant use of the trigger bit.

Note

Note the method used to count bits in the utilized PLC when specifying the bit. For more information, refer to the "Communication" section in the PLC Online Help.

1. Select "Mode" to specify whether to trigger the alarm at a rising or falling edge.
2. To configure the alarm text, select "Properties > Properties > Alarm texts".
   - To enter the alarm text, select "Alarm text".
   - To create additional alarm texts, select "Additional texts".
   - Use the functions of the shortcut menu to format text on a character-by-character basis and to insert parameter output fields.

     Note

     You can only enter additional text in the rows that you have enabled as user text blocks on the "Alarm text blocks" tab.

Result

The discrete alarm is created.

Additional settings for discrete alarms

Creating an infotext

To configure a tooltip for the alarm, follow these steps:

1. Select the discrete alarm.
2. Select "Properties > Properties > Infotext" in the Inspector window and enter the required tip.

Enabling discrete alarm parameters

To output process values to an output field in the alarm text, assign corresponding tags to the parameter blocks. Proceed as follows:

1. Select the alarm.
2. In the Inspector window, click "Properties > Properties > Alarm parameters".
3. Enter a value for the alarm parameter, or select a tag.
4. You can enter several alarm parameters.
   Insert the activated process values as selection box into an alarm text, or show them as parameter block in the alarm view.

   Note

   To show the process values as parameter blocks, the parameter blocks must be enabled in the alarm view.

   You only edit parameters that you have enabled under "Parameter blocks" on the "Alarm text blocks" tab.

Saving the alarm status

To save the alarm states to a tag, proceed as follows:

1. Select the discrete alarm.
2. Under "Properties > Properties > Status" in the Inspector window, select the tag and the bit that triggers saving of the incoming/outgoing alarm states and the acknowledgment request.

Configuring event-driven tasks

To configure event-driven tasks, such as a loop-in alarm, follow these steps:

1. Select the discrete alarm.
2. Select "Properties > Events" in the Inspector window and configure a new function list for the relevant event.

See also

Configuring alarm acknowledgment (Page 879)

Configuring acknowledgment of the alarm annunciator (Page 882)

Configuring loop-in alarms (Page 829)

Alarms in the "HMI tags" Editor (Page 831)

Configuring Alarm Display Suppression (Page 860)
Configuring reporting of alarms (Page 875)
Overview of alarm configuration tasks (Page 784)

Parameter output in discrete alarms (RT Professional)

Introduction

To display system or alarm parameters, insert an appropriate output field in the analog alarm.

- You can select the parameters configured in "Properties > Properties > Alarm parameters" for use as alarm parameters.
- The following system blocks can be selected as system parameters:
  - User Name
  - Computer name
  - Comment
  - Application Name

Requirement

- The "HMI alarms" editor is open.
- The discrete alarm is selected.

Procedure

To output parameters in alarm texts, proceed as follows:

1. Place the cursor at the required position in the alarm text.
2. To enable the output an alarm parameter, select "Insert parameter output field" from the shortcut menu.
   -OR-
   Enable the output of system parameters by selecting "Insert system block output field" from the shortcut menu.
3. Select the parameter.

4. Moreover, you can specify the following data for alarm parameters:
   - The tag that provides the parameter values.
     Enter the tag you configured for the parameter in "Properties > Properties > Alarm parameters". If you select a different tag, WinCC updates the parameter configuration in "Properties > Properties > Alarm parameters" accordingly.
   - The length, number of decimal places, precision, and alignment of the output field
   - To display leading zeros in the output field, enable "Leading zeros".
5. Click to save your entries.

Note

If you test a project using the simulator, the output field values are not output in the alarm texts.

Note

The sequence of the tag output fields in the alarm text depends on the language. The sequence of the Runtime language is used for logging alarms to a CSV log file.

Changing the tag of an output field in one language causes the modified output field to appear at the end of the alarm text in all other languages. This changes the sequence of the output fields in the log.

See also

Configuring discrete alarms (Page 814)

Configuring discrete alarms (Page 831)

Configuring user alarms (RT Professional)

Configuring user alarms (RT Professional)

Introduction

User alarms serve to to indicate operator actions in Runtime. User alarms are triggered by means of alarm number. For other custom applications, configure user alarms that are triggered in Runtime by means of scripts.

Requirement

- The "HMI alarms" editor is open.
- The Inspector window is open.

Procedure

To configure user alarms, proceed as follows:

1. Click the "User alarms" tab.
2. To create a new user alarm, double-click "<Add>" in the table.
   A new user alarm is created.
3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   – You can always change the object name of the alarm.
   – Select the alarm class and the alarm group, if necessary.
   – To display the CPU number or PLC that sends the alarm, enter these in the "Connection" area. This option is only active if you have enabled the "CPU/PLC number" alarm text block on the "Alarm text blocks" tab.
4. Click "Alarm texts" in the Inspector window and configure the alarm text.
   – Enter the alarm text.
   – To create additional alarm texts, select "Additional texts".
   – You can insert output fields or or additional text into the alarm text.

   Note

   You can only enter additional text in the rows that you have enabled as user text blocks on the "Alarm text blocks" tab.

Result

The user alarm is created.

Additional settings for user alarms

Creating a tooltip

To configure a tooltip for the alarm, follow these steps:

1. Select the user alarm.
2. Select "Properties > Properties > Tooltip" in the Inspector window and enter your text.

Saving the alarm status

To save the alarm states to a tag, proceed as follows:

1. Select the user alarm.
2. Select "Properties > Properties > Status" in the Inspector window and select the tag and the bit that triggers saving of the incoming/outgoing alarm states and the acknowledgment request.

Configuring event-driven tasks

To configure event-driven tasks, proceed as follows:

1. Select the user alarm.
2. Select "Properties > Events" in the Inspector window and configure a new function list for the relevant event.

See also

Configuring alarm acknowledgment (Page 879)

Configuring acknowledgment of the alarm annunciator (Page 882)

Configuring acknowledgment (Page 879)

Configuring alarm blocks (Page 806)

Configuring Alarm Display Suppression (Page 860)

Parameter output in user alarms (RT Professional)

Introduction

To display system or alarm parameters, insert an appropriate output field in the user alarm.

- The following system blocks can be selected as system parameters:
  - User Name
  - Computer name
  - Comment
  - Application
- Alarm parameters are configured in the script that you use to trigger a user alarm in runtime.

Requirement

- The "HMI alarms" editor is open.
- The user alarm is selected.

Procedure

To output parameters in alarm texts, proceed as follows:

1. Place the cursor at the required position in the alarm text.
2. To output an alarm parameter, select "Insert parameter field" from the shortcut menu.
   -OR-
   To output a system parameter, select "Insert system parameter" from the shortcut menu.

3. Select the parameter.
4. Click to save your entries.

Note

If you test a project using the simulator, the output field values are not output in the alarm texts.

Note

The sequence of the tag output fields in the alarm text depends on the language. The sequence of the Runtime language is used for logging alarms to a CSV log file.

Changing the tag of an output field in one language causes the modified output field to appear at the end of the alarm text in all other languages. This changes the sequence of the output fields in the log.

See also

Example: Defining alarm parameters for a user alarm (Page 822)

Example: Configuring a trigger for a user alarm (RT Professional)

Task

In the following example, you configure a user alarm that is triggered by a button in runtime.

To do this, you configure a script at the "Click" event of this button. This script calls up an alarm number and sets the associated alarm to the status "Incoming".

Requirements

- The "HMI alarms" editor is open.
- The Inspector window is open.
- A screen with an alarm view and a button is open.

Procedure

1. You configure a user alarm in the "HMI alarms" editor.

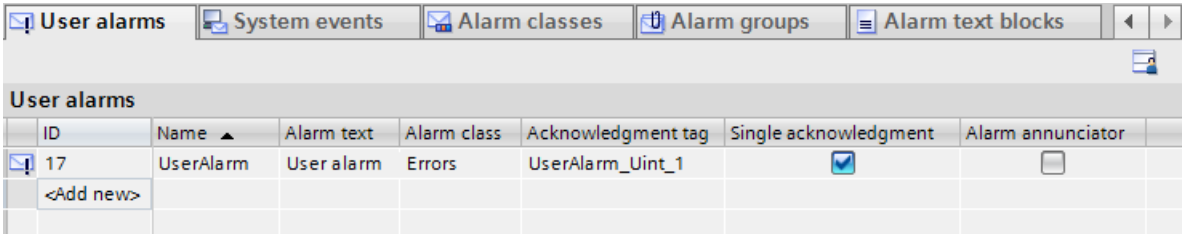

   The user alarm has the alarm number "17" in this case.

2. You configure the following script in the screen at the "Click" event of the button:

```
Sub onClick(byVal item)
    Dim UserAlarm
    Dim ID_UA
    Dim State_UserAlarm

    ID_UA = "17"
    State_UserAlarm = "5"

    Set UserAlarm = HMIRuntime.Alarms(ID_UA)
    UserAlarm.State = state_UserAlarm
    UserAlarm.Comment = "empty Comment"
    UserAlarm.UserName = "Max Mustermann"
    UserAlarm.Create "MyApplication"
End Sub
```

Result

If the user clicks on this button in runtime, the configured user alarm is displayed in the alarm view.

See also

Example: Defining alarm parameters for a user alarm (Page 822)

Example: Defining alarm parameters for a user alarm (RT Professional)

Task

In the following example, you configure a user alarm that is triggered by a button in runtime. For the user alarm, you define alarm parameters to be output in the alarm text.

To do this, configure the C system function "TriggerOperatorEvent" at the "Click" event of this button. This system function calls an alarm number and transfers the following parameters:

- Tag name as parameter 1 in the alarm text
- Start value of the tag as parameter 2 in the alarm text
- End value of the tag as parameter 3 in the alarm text

Note

Alarm parameters in the script

WinCC has a range of functions for defining alarm parameters, for example the "MSRTCreateMsgInstanceWithCommentPlus" function of the runtime API interface.

Requirements

- A screen with an alarm view and a button is open.
- The user alarm with alarm number 0x001 has been configured and uses alarm parameters 1 - 3 in the alarm text.
- Tag "HMI_Tag_1" of the type "Integer" has been configured and connected.

Procedure

1. You configure the following script in the screen at the "Click" event of the button:

```
Void SCFunction_1()
(
int a, b;
a = GetTagDWord ("HMI_Tag_1"); // old value
SetTagDWord("HMI_Tag_1", 100);
b = GetTagDWord ("HMI_Tag_1"); // new value
TriggerOperatorEvent(0x001, 1, "HMI_Tag_1", 0, a, b, "myComment");
}
```

Result

If the user clicks on this button in runtime, the configured user alarm is displayed in the alarm view. The tag name is output as parameter 1, and the start and end values of the tags as parameters 2 and 3.

See also

Example: Configuring a trigger for a user alarm (Page 821)

Parameter output in user alarms (Page 820)

Configuring alarm text (RT Professional)

Introduction

The alarm text for an alarm is made up of the alarm text, additional texts and output fields. The alarm text and additional texts are user text blocks. You can use up to 10 user text blocks for an alarm.

Each alarm text contains up to 255 characters and any number of output fields.

Requirements

- The "HMI alarms" editor is open.
- You have enabled user text blocks on the "Alarm text blocks" tab.
- An alarm has been created.

Procedure

Proceed as follows to assign additional user text blocks to the alarm text for an alarm:

1. Select the alarm that you want to edit on the relevant tab.
2. To enter the alarm text, select "Properties > Properties > Alarm text" in the Inspector window.
3. You can also add output fields to the alarm text.
4. To enter additional alarm texts, select "Properties > Properties > Alarm text > Additional text" in the Inspector window.
   Enter more additional texts according to which user text blocks you enabled under "User text blocks" on the "Alarm text blocks" tab.

   Note

   Use the scroll buttons to view text that is not fully visible in the text box. The additional text lines are displayed.

Result

You have configured the alarm text with additional texts for this alarm. To display it in Runtime, enable these user text blocks in an alarm view.

See also

Configuring analog alarms (Page 809)

Configuring discrete alarms (Page 814)

Editing system events (RT Professional)

Introduction

System events can only be changed to a limited extent. You cannot delete or create new system events.

Requirements

- You have imported the system events.
- The "HMI alarms" editor is open.

Procedure

To configure system events, proceed as follows:

1. Click the "System events" tab.
2. Select the system event that you want to edit.
3. Change the editable properties in the Inspector window.
4. Select the system event that you want to edit.
5. Change the editable properties in the Inspector window.

Configuring event-driven tasks

To configure event-driven tasks, proceed as follows:

- To configure a function list for the relevant event, select "Properties > Events".

Editing controller alarms (RT Professional)

Introduction

Controller alarms are configured in STEP 7. Controller alarms are available in WinCC running in a STEP 7 environment.

In WinCC you only edit the properties of controller alarms that are specific to an HMI device. Different properties of a controller alarm can be edited, depending on the HMI device and PLC.

You can filter the displayed controller alarms for your project using the display classes that were configured on the PLC for controller alarms.

Note

When "Automatic update" is set for controller alarms for the connection to the S7-1500 controller, you cannot edit the properties of the respective controller alarms. The controller alarms are not displayed in the "HMI alarms" editor in this case.

Requirements

- The connection was established to the PLC.
- Alarms were configured in STEP 7.

Editing controller alarms

Proceed as follows to edit controller alarms:

1. Double-click "HMI alarms" in the project navigation.
   The "HMI alarms" editor is opened.
2. Open the "Controller alarms" tab.
3. Select the alarm that you want to change in the work area.
4. Make your changes. Depending on the HMI device, it may not be possible to display or enable some of the controller alarm properties.

Editing SIMATIC S7 alarms

You can edit these controller alarms in the STEP 7 alarm configuration system. Proceed as follows:

1. Select the alarm that you want to change in the work area.
2. Select "Open in alarm editor of the PLC" from the shortcut menu.
   The STEP 7 alarm configuration system is opened.
3. Make your changes.

Filtering the controller alarm using display classes

To filter controller alarms by display classes, proceed as follows:

1. Click "Runtime settings > Alarms" in the project navigation under your HMI device.
   One or several connections to a PLC are shown in "Contoller alarms".
2. Select the display classes whose controller alarms you want to display for the connection.

Note

Controller alarms after a basic panel device exchange

For the controller alarms (PLC monitoring alarms, program alarms or system alarms) to be displayed after a device replacement ("Device view > Change device") from a Basic Panel to a Comfort Panel, Mobile Panel or WinCC RT Advanced, you need to activate the "Display classes" setting for the respective connections under "Runtime settings > Alarms > Controller alarms".

Result

Only those controller alarms whose display class you have enabled for your HMI device will be displayed on the "Controller alarms" tab of the "HMI alarms" editor.

See also

Controller alarms (Page 762)

Sending a complete alarm from the controller to the HMI device and automatic update (Page 950)

Configure operator input alarms (RT Professional)

Introduction

You have the option of triggering an operator input alarm for the operation in the process. The configuration of an operator input alarm depends on the operated object and on the type of operation.

For the operation of some objects, for example the input of a value in an I/O field or an list field, the property "Operator input log" can be activated at the objects. The layout of the alarm cannot be edited in the operator input log. The Operator Actions Report contains the value before the operation in the process value block 2, and the changed value process value block 3.

The property "Request motive" specifies whether the motif will be logged by operator for an operation. By enabling the option the operator can enter the motive (reason) of the operation as alarm after an operation in a dialog. The Operator Actions Report is logged in the alarm system. The motive as displayed as comment to the operator input alarm in the historical alarm list (long-term). In the comment dialog, the old value and the new are only displayed for the "12508141".

Requirement

- The "Screens" editor is open.
- An I/O field is created.
- Each object is associated with a tag.

Procedure

1. Select the I/O field.
2. Click "Properties > Properties > Security" in the Inspector window.
3. To use operator input for the object, select the option "Log > Operator Actions Report".
4. If you want to log the operation motive, enable the property "Request motive".

Result

The configured operator input alarm is triggered when the user performs an action on the I/O field in Runtime. When you operate the object, a dialog opens in Runtime in which the operator can enter a text.

See also

List box (Page 416)

Symbolic I/O field (Page 487)

Combo box (Page 400)

Scroll bar (Page 349)

Configuring Operator Input Alarms for an Alarm View (RT Professional)

Introduction

Operator input alarms for the alarm view are preconfigured system alarms. Manage system alarms using the "HMI alarms" editor. In the "Screens" editor, configure the trigger conditions for operator input alarms for the alarm view.

Requirement

- You have imported the system alarms from the HMI device.
- The "Screens" editor is open.
- An "Alarm view" object has been created.
- Alarm text blocks are activated in the "HMI alarms" editor.

Procedure

To configure operator input alarms for an alarm view, proceed as follows:

1. Select the alarm view.
2. In the Inspector window, click "Properties > Properties > Operator input alarms".
3. Activate the operator action at the alarm view for which an operator input alarm is to be output.
   The alarm number of the preconfigured operator input alarm for this operator action is displayed in the "Alarm number" output field.
4. To output another alarm for this operator action, enter the alarm number of this alarm.
5. To specify the columns to display in the operator input alarm, click the button in "Alarm text blocks".
   A dialog opens.

6. To enable the alarm text blocks to be output with the operator input alarm, select "Alarm text blocks".
7. Under "Alarm group of operator input alarm", specify the alarm text blocks to be output with the system alarm. Under "Output as", define whether the content of the alarm text block should be output as text or as a value.

Note

You will find the system alarm and its alarm number on the "System alarms" tab in the "HMI alarms" editor.

Result

The configured operator input alarm is displayed when the user performs an action on the alarm view in Runtime.

3.2.2.7 Configuring loop-in alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

A Loop-In-Alarm is configured to change to your screen that contains the relevant information for the error that has occurred.

Note

You configure the screen that contains information about the error that occurred. No standard screens are available that are automatically called depending the selected error alarm.

Requirement

- The screen called by the Loop-In-Alarm has been created.
- The "HMI alarms" editor is open.

Procedure

To configure a Loop-In-Alarm for an alarm, proceed as follows:

1. Click the tab that contains the alarm for which you want to configure the Loop-In-Alarm.
2. Select the alarm.
3. In the Inspector window, select "Properties > Events > Loop-In-Alarm".

4. Select the "ActivateScreen" system function.
5. Select the screen to be called by the Loop-In-Alarm as its parameter.

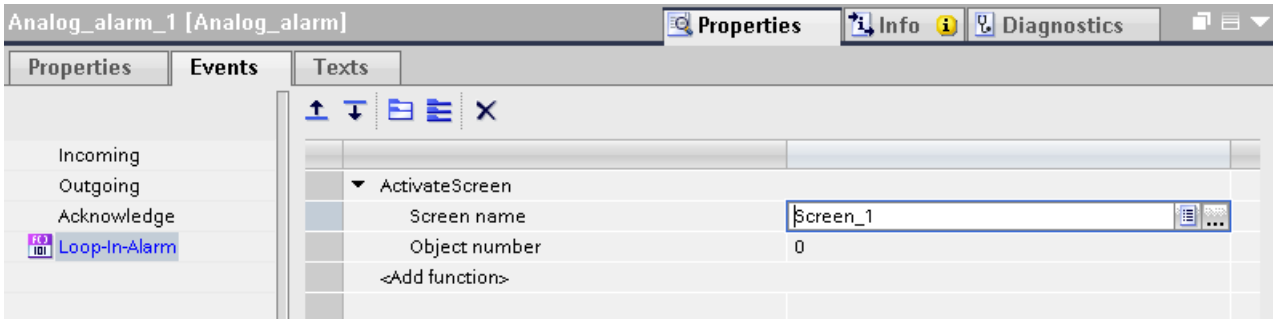

Note

To configure the Loop-In-Alarm for an alarm view with an "alarm line" format, use a button with the following system functions:

- "EditAlarm" for HMI devices with keys
- "AlarmViewEditAlarm" for HMI devices without keys

The system functions trigger the "Loop-In-Alarm" event. The alarm line has no buttons.

Result

If you select an alarm in the alarm view in Runtime and click on the "Loop-In-Alarm" button, a screen is opened with information on the selected alarm.

Even if multiple alarms are selected, exactly one alarm requiring acknowledgment is acknowledged (provided it requires acknowledgment) when the "Loop-In-Alarm" button is pressed for the first time. The system functions stored for the "Loop-In-Alarm" alarm event are always executed, regardless of whether the alarm requires acknowledgment or not.

The next time the "Loop-In-Alarm" button is pressed, only the system functions stored for the "Loop-In-Alarm" alarm event are executed.

Note

If no Loop-In-Alarm is configured for the selected alarm, pressing the "Loop-In-Alarm" button has no effect.

See also

Using the Alarm Window, Alarm View (Page 918)

Basic alarm view, using the alarm window (Page 922)

Alarm view (Page 929)

Configuring analog alarms (Page 809)

Configuring discrete alarms (Page 814)

3.2.2.8 Alarms in the "HMI tags" Editor

Configuring discrete alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In WinCC, you can create and edit discrete and analog alarms, including the trigger tags, in the "HMI tags" editor.

Note

If you delete, move or copy objects in the "HMI tags" editor, the changes also take effect in the "HMI alarms" editor.

Requirements

The "HMI tags" editor is open.

Procedure

To configure a discrete alarm, proceed as follows:

1. To create a tag, click on "<Add>" in the table at the top of the work area.
   A new tag is created.
2. Configure an internal or external tag as required.
   - Use the data types "Int" or "UInt" to select an HMI tag.
   - When you select a PLC tag, use the data types "Int" or "Word". The input and output area of a PLC tag is unsuitable as trigger.
3. Select the tag at the top of the work area.
4. Click on "<Add>" in the table on the "Discrete alarms" tab at the bottom of the work area.
   A new discrete alarm is created for the tag. If you have selected the incorrect data type, the tag will be marked in the discrete alarm.

5. Configure the discrete alarm in the Inspector window:
   - Enter the alarm text under "Properties > Properties > General > Alarm text".
     You can also insert output fields into the alarm text.
   - Select an alarm class.
   - Select the trigger bit of the tag that triggers the discrete alarm under "Properties > Trigger".
6. You can create additional discrete alarms to monitor the tags.

Note

A tag is monitored using only one alarm type. You should therefore create either analog alarms **or** discrete alarms for a tag.

Result

The configured discrete alarms are created in the "HMI tags" editor and displayed in the "HMI alarms" and "HMI tags" editors.

See also

Configuring discrete alarms (Page 796)

Parameter output in the analog alarm (Page 813)

Working with Tags (Page 531)

Configuring analog alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In WinCC, create the discrete and analog alarms, including the trigger tags, in the "HMI tags" editor. You can also edit the alarms as in the "HMI alarms" editor. You can create up to two range values for a tag. You monitor these limits with analog alarms.

Requirements

The "HMI tags" editor is open.

Procedure

To configure an analog alarm in the "HMI tags" editor, proceed as follows:

1. To create a tag, click on "<Add>" in the table at the top of the work area.
   A new tag is created.
2. Configure an internal or external tag as required.
3. In the Inspector window, configure the range values of the tags under "Properties > Properties > Range":
   – Select whether to use a "Constant" or an "HMI tag" as limit value for your range values. The object list opens when you select "HMI tag". Select the tag you want to use.

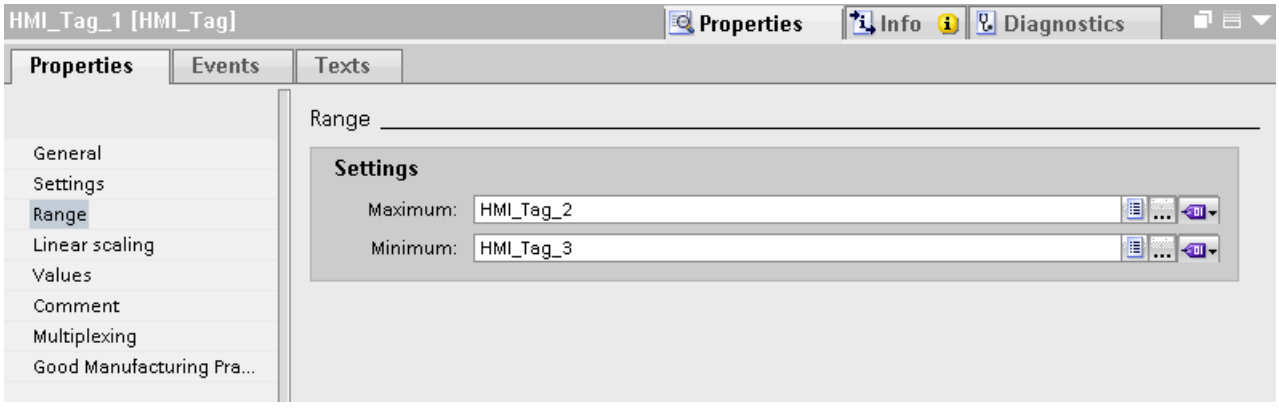

1. Click the "Analog Alarms" tab at the bottom of the work area.
   Create an analog alarm for both range values.
2. Select an analog alarm and configure it in the Inspector window:
   – Enter the alarm text under "Properties > Properties > General > Alarm text".
   – You can also insert output fields into the alarm text.
   – You can change the default alarm class.
3. Configure the analog alarms as in the "HMI alarms" editor.
4. Complete the configuration of all analog alarms.

Note

A tag is monitored using only one alarm type. You should therefore create either analog alarms **or** discrete alarms for a tag.

Result

The configured analog alarms are created in the "HMI tags" editor and displayed in the "HMI alarms" and "HMI tags" editors.

See also

Parameter output in the analog alarm (Page 813)
Configuring analog alarms (Page 809)
Working with Tags (Page 531)

3.2.3 Configuring the Outputting of Alarms

3.2.3.1 Overview of configuring alarm output (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Steps to complete when configuring alarm output

You configure the alarm output in WinCC in the following steps:

1. Create alarm view
   Use the display and control objects in the "Screens" editor to display alarms in Runtime. You can also configure an alarm view for logged alarms.
2. Configure acknowledgment
   In the "Screens" editor, you can set the operator action that will trigger the acknowledgment.
3. Configure reporting
   You can create reports in the "Reports" editor to print alarms in Runtime. In the "Scheduler", "Screens", "HMI alarms" or "HMI tags" editors, you define when and how the printing of a report is triggered.

Note

Device dependency

Reporting and logging are not available for all HMI devices.

Note

Note that only ASCII characters are supported when printing alarms directly in Runtime.

Additional configuration tasks

Additional tasks may be necessary for configuring alarm views, depending on the requirements of your project:

1. Setting up authorizations
   To make sure only authorized operators process the alarms, assign authorizations for the alarm view and the function keys of the HMI device.
2. Configuring the filtering of the alarm view
   You configure the filtering of the alarms in Runtime in the "Screens" editor. You can also configure alarm views that only display selected alarms.
3. Configure operator input alarms
   Configure the operator input alarms on the operator controls of the HMI device in the "Screens" editor. A preconfigured operator input alarm is output for an operator action. An operator input is, e.g. the acknowledgment of an alarm.

See also

Configuring the alarm view (Page 835)

Configuring the alarm view (Page 850)

Reporting Alarms (Page 873)

3.2.3.2 Configuring the alarm view

Displaying Alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Options for displaying alarms on the HMI device

WinCC offers the following options for displaying alarms on the HMI device:

- Alarm view
  The alarm view is configured in a screen. More than one alarm can be displayed simultaneously, depending on the configured size. You can configure multiple alarm views with different contents.
- Alarm window
  The Alarm window is configured in the "Global screen" editor. The alarm window can display multiple alarms at the same time, depending on the configured size. An event can trigger closing and reopening of the alarm window. To hide it during configuration, create an alarm window on its own level.

Additional signals

- Alarm indicator
  The alarm indicator is a configurable, graphical icon. When an alarm comes in, the alarm indicator is displayed on the HMI device. You configure the alarm indicator in the "Global screen" editor.
  The alarm indicator has two states:
  - Flashing: At least one alarm that requires acknowledgment is pending.
  - Static: The alarms are acknowledged but at least one of them has not gone out yet. The alarm indicator also displays the number of pending alarms according to the HMI device.
- E-mail notification
  To inform someone other than the operator, such as an engineer, when an alarm with a specific alarm class arrives, assign the alarm class to an e-mail address.

  Note

  Device dependency

  E-mail notification is not available for all HMI devices.

- System functions
  You can configure a list of functions for the event associated with an alarm. These functions must be executed in Runtime when the event occurs.
  Use system functions for alarms in WinCC to control the alarm view or the alarm window other than via the toolbar.

Displaying the predefined alarm classes in Runtime

The following table shows the symbols used to display the predefined alarm classes in the alarm view:

Alarm class	Displayed icon
"Diagnosis Events"	S7
"Errors"	!
"System"	$
"Warnings"	<No symbol>

See also

System functions for alarms (Page 953)

Configuring an alarm view

Configuring an alarm view for active alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Current alarms are displayed in Runtime in an alarm view or alarm window.

Requirement

- A screen is open in the "Screen" editor.
- The "Tools" task card is open.

Configuring alarms for the alarm view

To specify the alarms that will be shown in the alarm view, proceed as follows:

1. Insert an "Alarm view" object from the "Tools" task card into the screen.
2. Select the alarm view.
   – In the Inspector window, select "Properties > Properties > General > View > Current alarm states".
     Set whether to display alarms with and/or without mandatory acknowledgment.

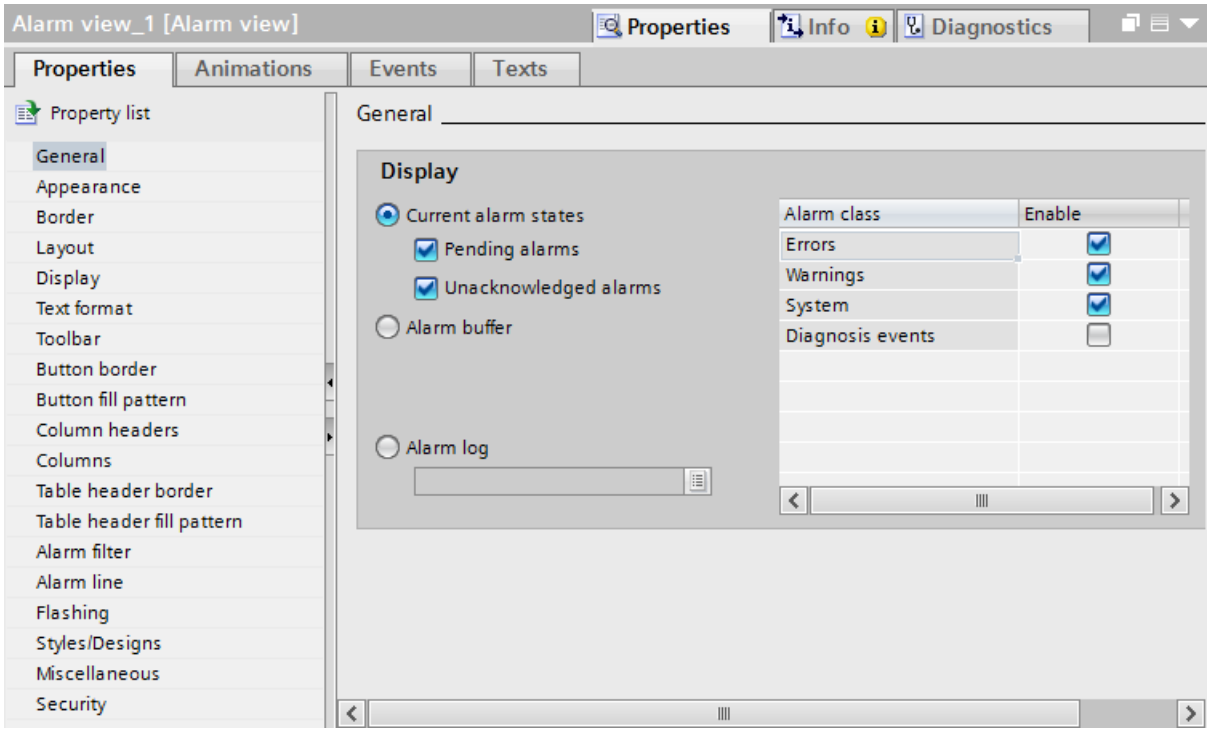

– To display all alarms in the alarm buffer, enable "Alarm buffer".

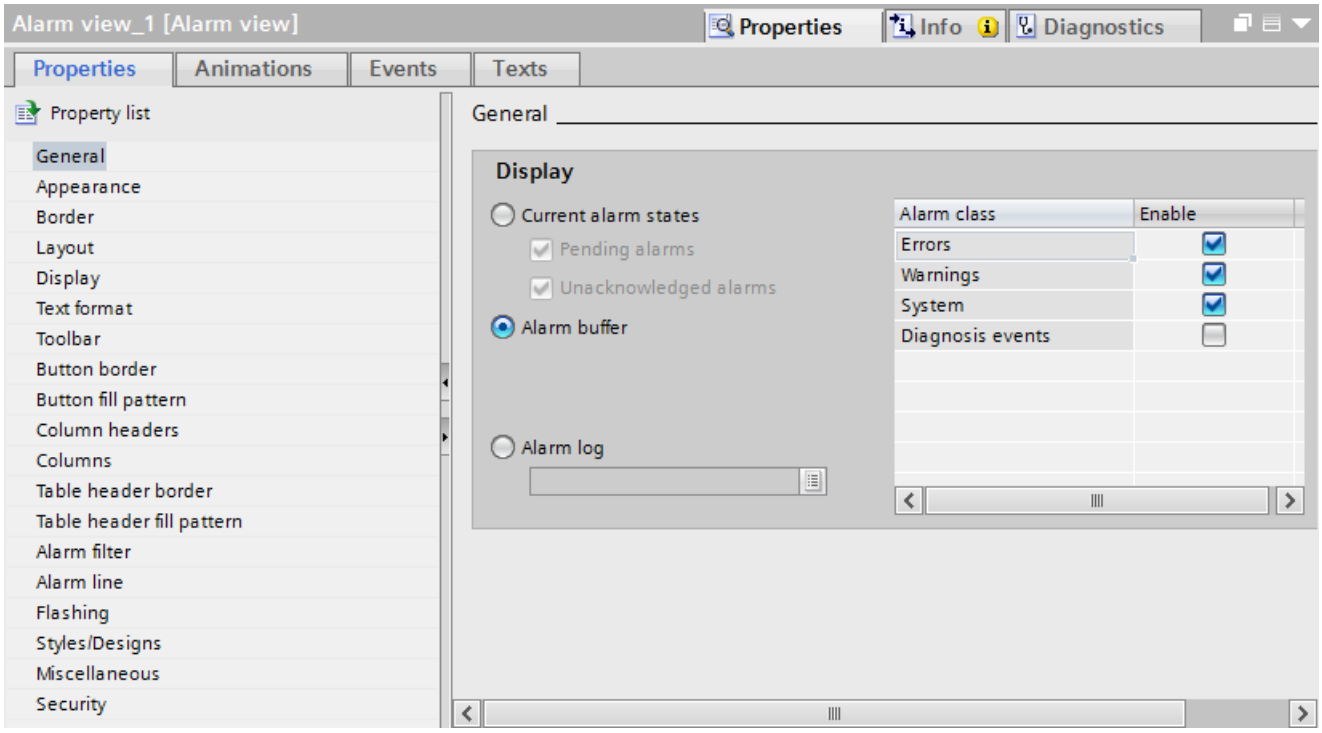

3. In the table, activate the alarm classes to be displayed in the alarm view.
4. Under ""Properties > Properties > View > Control tag for display area", define the tag that returns the date as of which the alarms will be displayed.

Note

Device dependency

The "Control tag for display area" property is not available for all HMI devices.

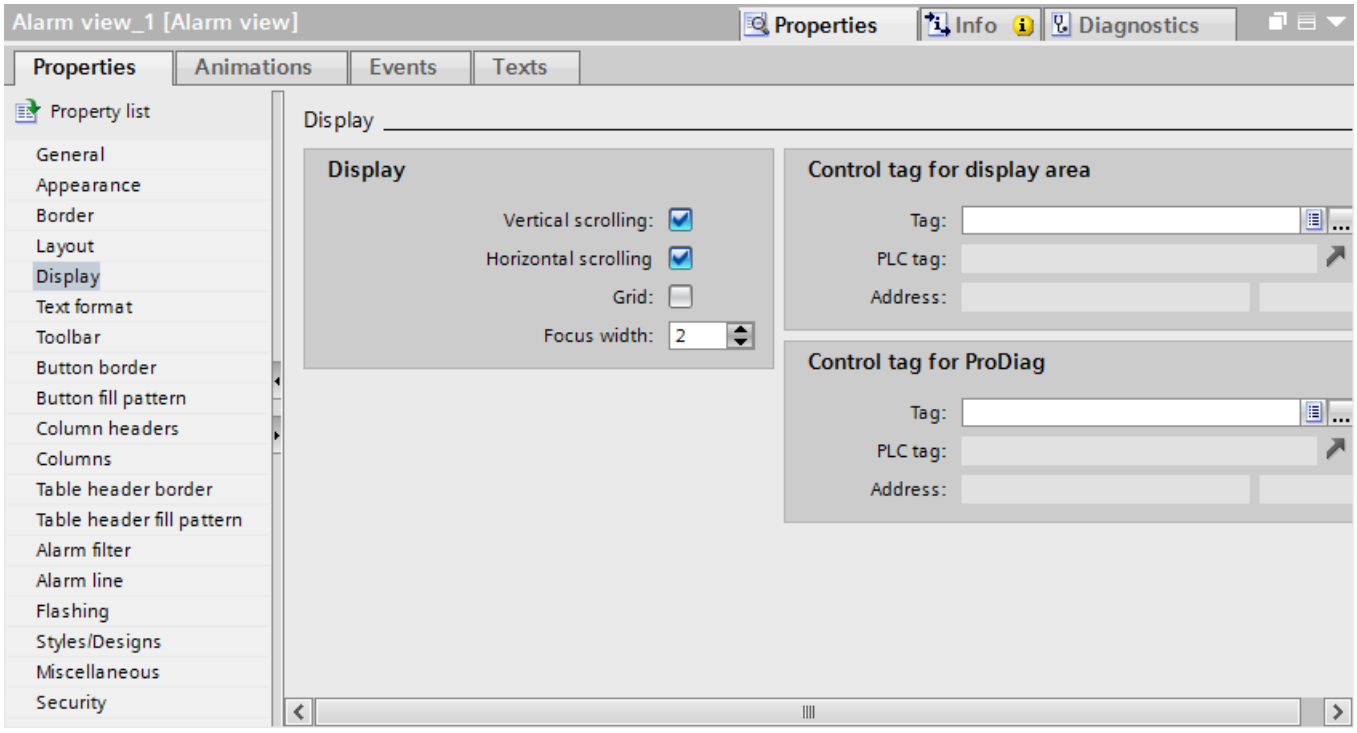

Configuring the layout of the alarm view

To specify how the alarms are shown in the alarm view, proceed as follows.

1. Under "Properties > Properties > Layout > Settings > Lines per alarm" in the Inspection window, specify the number of lines to display for each alarm.
2. In "Properties > Properties > View", select the control elements that are available on the HMI device.
3. Configure the columns under "Properties > Properties > Columns":
   - Under "Visible columns" select the columns to be output in the alarm view.
   - Under "Properties Column", define the properties of the columns.
   - Under "Sort", select the sorting order of the alarms.

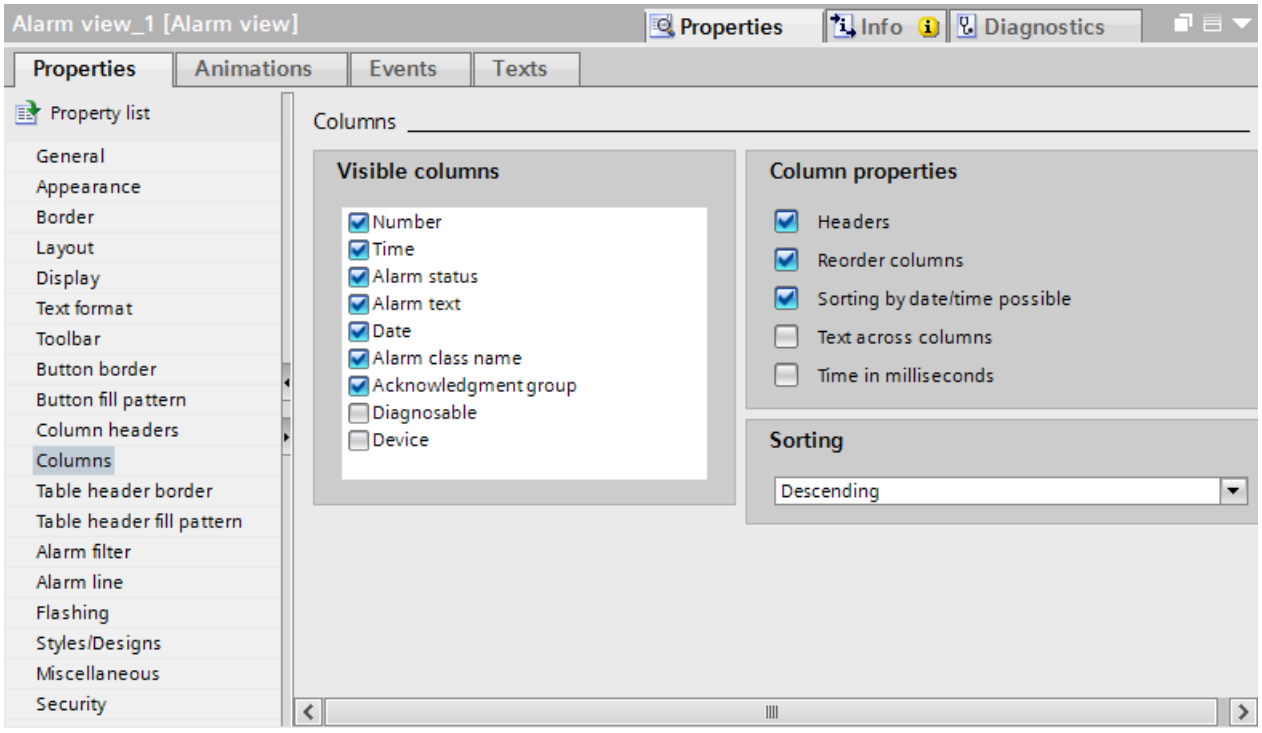

Note

Select the "Edit" command in the shortcut menu for the alarm view to activate the alarm view. You can set the column width and position in active mode. To enable the alarm view, set the zoom factor to 100 %.

The alarm view is deactivated as soon as you deselect the object in the screen.

Result

Alarms of various alarm classes are output in the alarm view during runtime.

See also

Creating alarm classes (Page 785)
Configuring an alarm window (Page 844)
Displaying Logged Alarms (Page 849)

Configuring an alarm view for the S7 diagnostic alarms (RT Advanced, Panels, Comfort Panels)

Introduction

To display S7 diagnostic alarms, configure an alarm view or alarm window for the predefined "Diagnosis Events alarm class. Activate the diagnostic alarms accordingly in the Runtime settings of your HMI device.

Note

Device dependency

- The "Diagnosis Events" alarm class is not available for all HMI devices.
- System-defined controller alarms of the type "SIMATIC S7 diagnostic alarm" can be activated for your HMI device only for connections with a SIMATIC S7-300/400 controller.

Note

Using system diagnostics view and/or system diagnostics window

As soon as a system diagnostics view or a system diagnostics window in available in the HMI project, the alarm texts for the screen objects are downloaded for system diagnostics and used there. Therefore, the option "Runtime settings > Alarms > System alarms > Show alarm text" has no effect when system diagnostics view or system diagnostics window and is always considered enabled. In this case the size of the Runtime file cannot be reduced by enabling the option.

Requirements

- An alarm view or alarm window is configured in the "Screens" editor
- A connection to a SIMATIC S7-300/400 controller has been established.

Configuring the display of S7 diagnostic alarms

To configure the display of S7 diagnostic alarms, follow these steps:

1. Select "Runtime settings > Alarms" in the project navigation.
2. Under "System events" select the check box "S7 diagnostic alarms (number)".
3. To display the S7 diagnostic alarms with text, activate "With event text".
4. Open the screen with the alarm view and select the alarm view.

5. In the Inspector window, select "Properties > Properties > General > Display > Alarm class "Diagnosis Events".

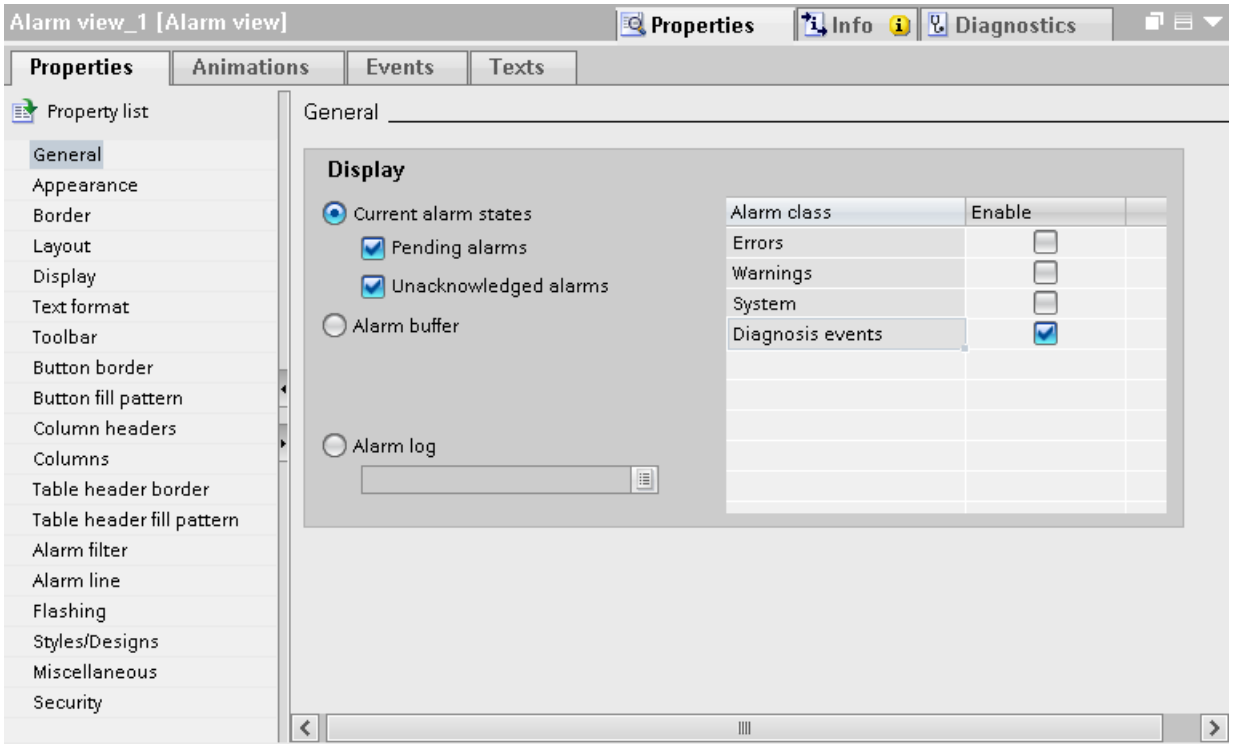

6. Continue to configure the alarm view as for the display of current alarms.

Result

In Runtime the alarm view displays the S7 diagnostic alarms of the controller.

Configuring an alarm view for logged alarms

Introduction

In Runtime, logged alarms are displayed in an alarm view or alarm window.

Requirements

- An alarm view or alarm window is configured in the "Screens" editor
- An alarm log was created in the "Log" editor.
- The alarms are assigned the "loggable" attribute in the "HMI alarms" editor.

Configuring an alarm view for logged alarms

To configure an alarm view for logged alarms, proceed as follows:

1. Open the screen with the alarm view and select the alarm view.
2. In the Inspector window, select "Properties > Properties > General > Alarm log".
3. Click the "...." button and select the alarm log.
4. Continue to configure the alarm view as for the display of current alarms.

Result

In Runtime, the logged alarms are output in the alarm view.

Configuring an alarm window (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

The alarm window displays current alarms. The alarm window is configured in the "Global screen" editor. The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active. Depending on the configuration, it is not closed until the alarm is acknowledged. The HMI device can still be used, even if alarms are pending and displayed. An alarm window is displayed and configured like an alarm view.

To hide an alarm window during configuration, create it on its own level.

Requirement

- The "Global Screen" editor is open.
- The "Tools" task card is displayed.
- The Inspector window is open.

Procedure

Proceed as follows to configure an alarm window:

1. Insert an "Alarm window" object from the "Tools" task card into the global screen.
2. Configure the alarm window like an alarm view.
3. Under "Properties > Properties > Mode > Window" in the Inspector window, select how the alarm window reacts and is operated in Runtime.
   – Activate "Modal" if the alarm window is to retain the focus in Runtime after a screen change.
     This option is important, as switching back and forth between the screen and different windows with <Ctrl+TAB> is not supported.

Result

During runtime, the alarms of the selected alarm class are displayed in the alarm window.

See also

Configuring an alarm view for active alarms (Page 837)

Creating alarm classes (Page 785)

Configuring an alarm indicator (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

The alarm indicator uses a warning triangle to indicate that alarms are pending or require acknowledgment. If an alarm of the configured alarm class occurs, the alarm indicator is displayed.

The alarm indicator has two states:

- Flashing: At least one alarm that requires acknowledgment is pending.
- Static: At least one of the acknowledged alarms has not gone out yet.

During configuration, specify whether Runtime has to open an alarm window when you operate the alarm indicator.

Requirement

- The "Global Screen" editor is open.
- The "Tools" task card is open.
- The Inspector window is open.

Procedure

Proceed as follows to configure the alarm indicator:

1. Insert the "Alarm view" object from the "Tools" task card into the work area.
2. Select the alarm indicator.

3. Under "Properties > Properties > General" in the Inspector window, select the alarm classes to be displayed by the alarm view.
   Specify whether to display pending and/or acknowledged alarms in the alarm view.

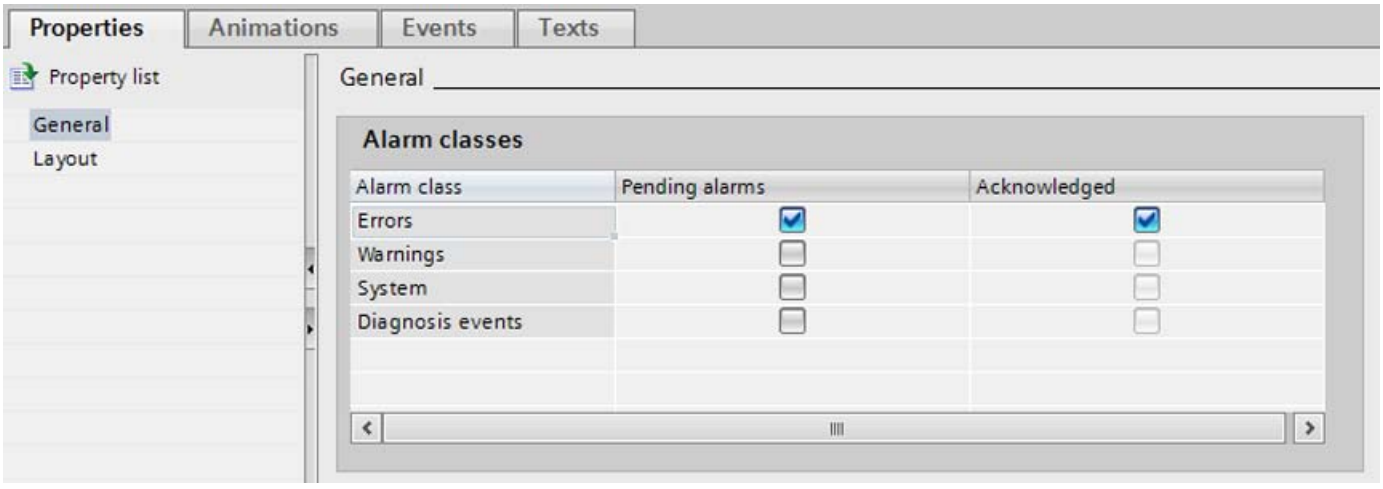

4. Under "Properties > Event", assign the system function "ShowAlarmWindow" to an event of the alarm view.

Note

If you have configured a permanent area in the screen or template, do not position the alarm window and alarm view in the vicinity of the permanent area. Otherwise, the alarm window and the alarm view will not be displayed in Runtime. However, the permanent area is not visible in the "Global screen" editor.

Result

The alarm view is displayed if alarms from the selected alarm class are pending or need to be acknowledged in Runtime. The alarm window opens when the user operates the alarm view.

Configuring alarm filtering (RT Advanced, Panels, Comfort Panels)

Introduction

You can filter the display of alarms in the enhanced alarm view. The criterion is always a character string. There are two ways to filter the alarm view:

- A fixed criterion is configured in WinCC.
  In Runtime, all alarms that contain the complete character string in the alarm text are displayed.
- You filter the alarm view in Runtime.
  In Runtime, a filter tag sets the relevant character string in the alarm view by means of an I/O field, e.g. as described below.
  The filter affects the display in the alarm view. All alarms in the alarm buffer are retained.

Requirement

- The alarm view has been configured.
- The screen with the alarm view is open.
- The Inspector window is open.

Configuring filters for a fixed character string

1. Select the alarm view.
2. In the Inspector window, select "Properties > Properties > Alarm filter".
3. Enter a filter string in "Filter string".

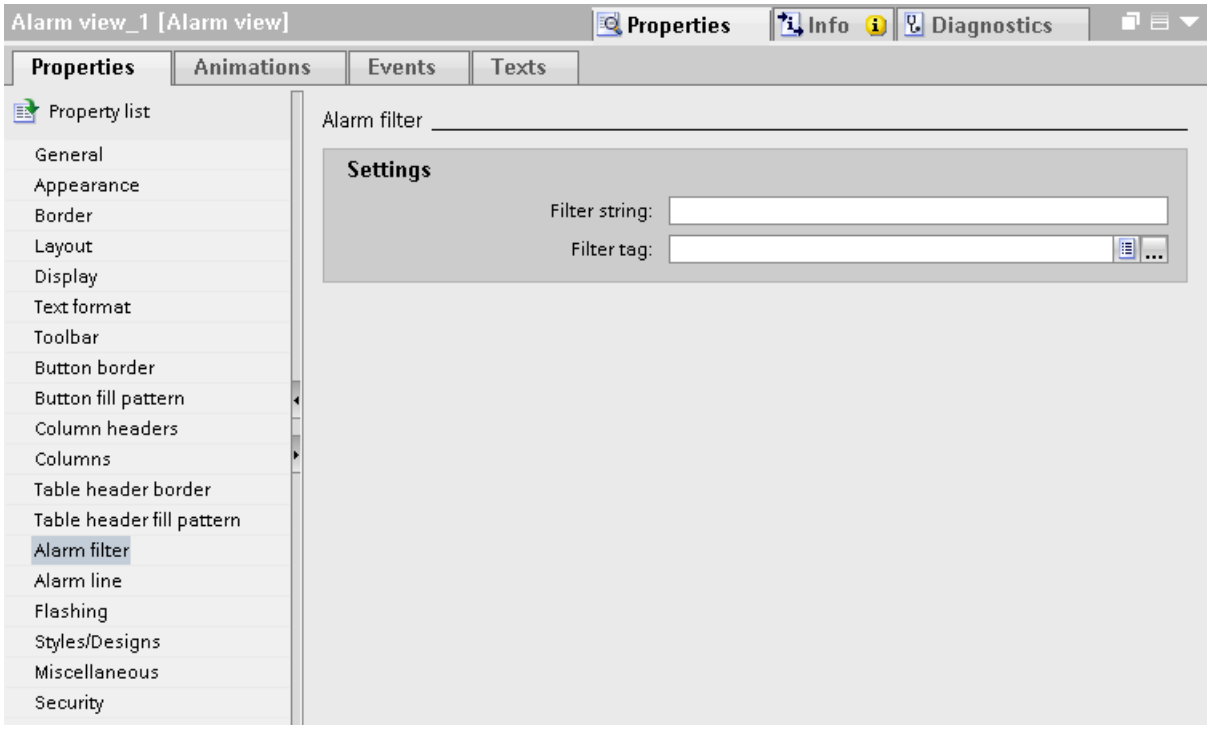

Result

Only those alarms are displayed in Runtime that contain the complete character string from the filter in the alarm text.

Configuring filters for a variable character string

1. Select the alarm view.
2. In the Inspector window, select "Properties > Properties > Filter".

3. Select a tag from "Filter tag". The tag must be of the "String" type.
4. In the screen, configure an I/O field for entering the filter string in Runtime.
   - In the Inspector window, select "Properties > Properties > General > Format > Display format > String".
   - To link the I/O field to the alarm view, select "Process" to set the filter tag you selected for the alarm view.

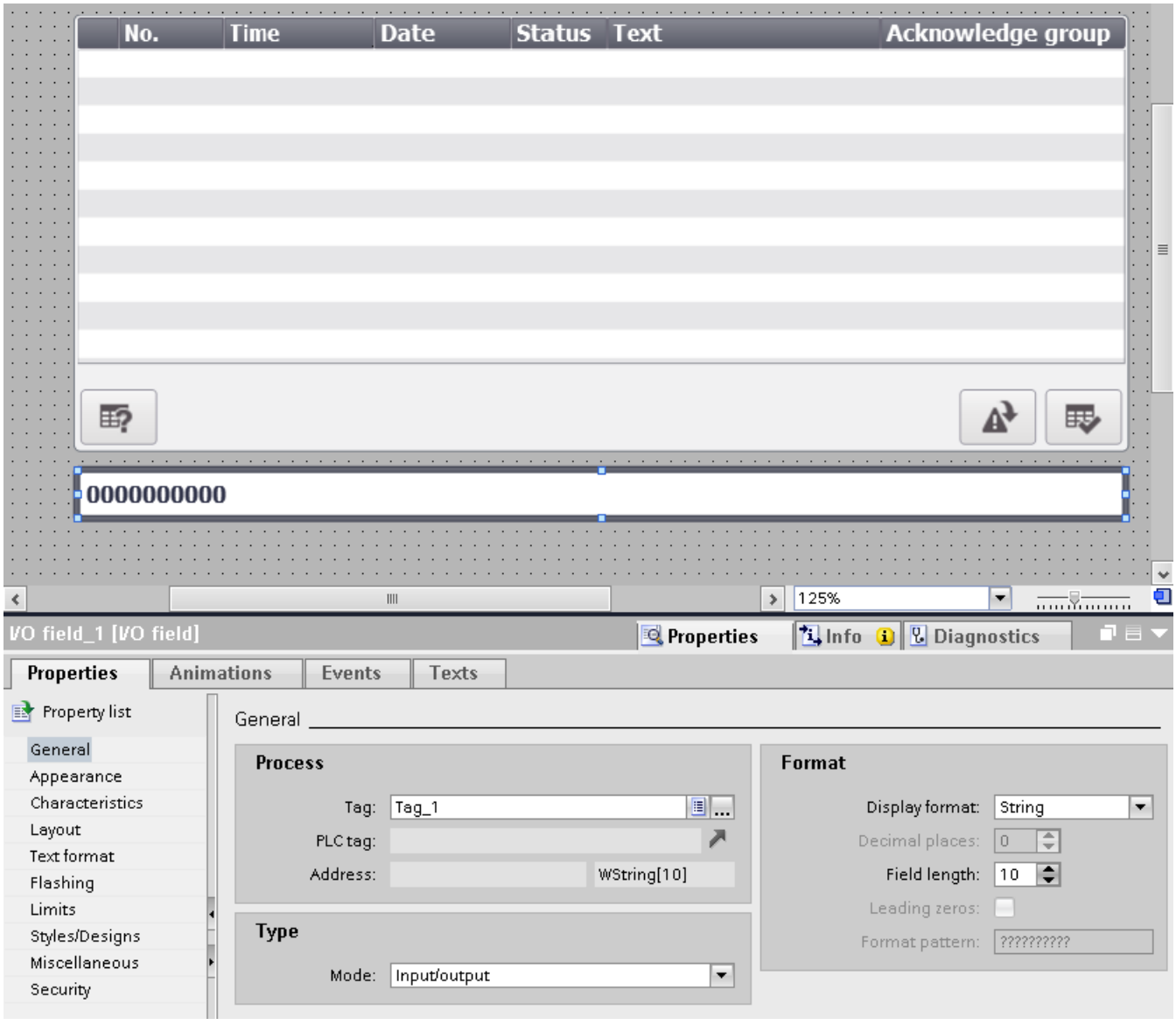

Result

If you enter a string in the I/O field in Runtime, the alarm view only displays alarms that contain the complete string in their alarm text.

If a permanently configured character string and a filter tag were configured, the alarms will be filtered in Runtime according to the content of the filter tag. If the filter tag is blank, the permanently configured character string is used as the filter.

Note

Filtering is not possible for the following alarm views:

- Simple alarm view
- Alarm line
- Alarm window
- Alarm view for logged alarms

See also

Working with Tags (Page 531)

Displaying Logged Alarms (RT Advanced, Panels, Comfort Panels)

Introduction

You display logged alarms in Runtime in an alarm view. The alarm view shows the content of an alarm log.

Application

For example, you want to view information about the process at the end of a shift.

Requirements

- An alarm log has been created.
- A screen is open.
- The Inspector window is open.

Procedure

Proceed as follows to configure an alarm view to output an alarm log:

1. Configure an alarm view in a screen.
2. Under "Properties > Properties > General > Display > Alarm log"" in the Inspector window, select the required alarm log.
   You can always create a new alarm log.

Result

During runtime, the alarms logged in the selected alarm log appear in the Alarm view.

Note

In runtime, the alarm texts logged are not displayed but rather the alarm texts in the current project. The logged alarm texts are only intended for external evaluation of the log file.

If the alarm log displayed was configured differently, the alarm texts displayed may not correspond with those logged.

See also

Configuring an alarm view for active alarms (Page 837)

3.2.3.3 Configuring the alarm view (RT Professional)

Displaying alarms (RT Professional)

Options for displaying alarms on the HMI device

WinCC offers the following options for displaying alarms on the HMI device:

- Alarm view for pending and outgoing alarms
  The alarm view is configured for a screen. More than one alarm can be displayed simultaneously, depending on the configured size. You can also configure multiple alarm views with different contents and in different screens. Use the "Historical alarm list (short-term)" and "Historical alarm list (long-term)" views to display outgoing alarms.
- Alarm view for statistical values
  The "Alarm statistics" view of the alarm view displays statistical information, such as the frequency and display duration of alarms.
- Status tag of an alarm group
  You can visualize the changing alarm states of an alarm group using the status tag of that alarm group. You can customize the display in the "Screens" editor.

Additional signals

- Alarm annunciator
  In addition to the alarm view, WinCC displays an alarm in the form of a visual or audible alarm in the plant. The alarm annunciator may be a hooter, or a warning lamp.
- System functions and scripts
  You can configure a list of functions for the event associated with an alarm. These functions must be executed in Runtime when the event occurs.
  To be able to control the alarm view other than via the toolbar, use system functions for alarms in WinCC.

See also

Bit assignment of status tag (Page 1018)
Tags of an alarm group (Page 778)
System functions for alarms (Page 953)

Configuring an alarm view (RT Professional)

Introduction

Current, and logged alarms are displayed in the alarm view. Alarms from all alarm classes are displayed in an alarm view, if necessary. Configure the criteria for alarm filtering.

Requirements

- A screen is open.
- The "Tools" task card is open.

Procedure

To configure an alarm view, proceed as follows:

1. Insert an "Alarm view" object from the "Tools" task card into the screen.
2. Select the alarm view and click "Properties > Properties > General" in the Inspector window.
   - Under "Display > Type", select the type of alarms you want to display as default in the alarm view.
   - Under "Display > Display" select which alarms are displayed. This shows or hides the alarms to be suppressed in the display.
   - Double-click of the operator on an alarm in Runtime starts a specific action. Specify this action under "Display > Double-click". If you want to trigger the Loop-In-Alarm not only using the "Loop-In-Alarm" button, select "Loop-In-Alarm".

– Select the required time base under "Time base".

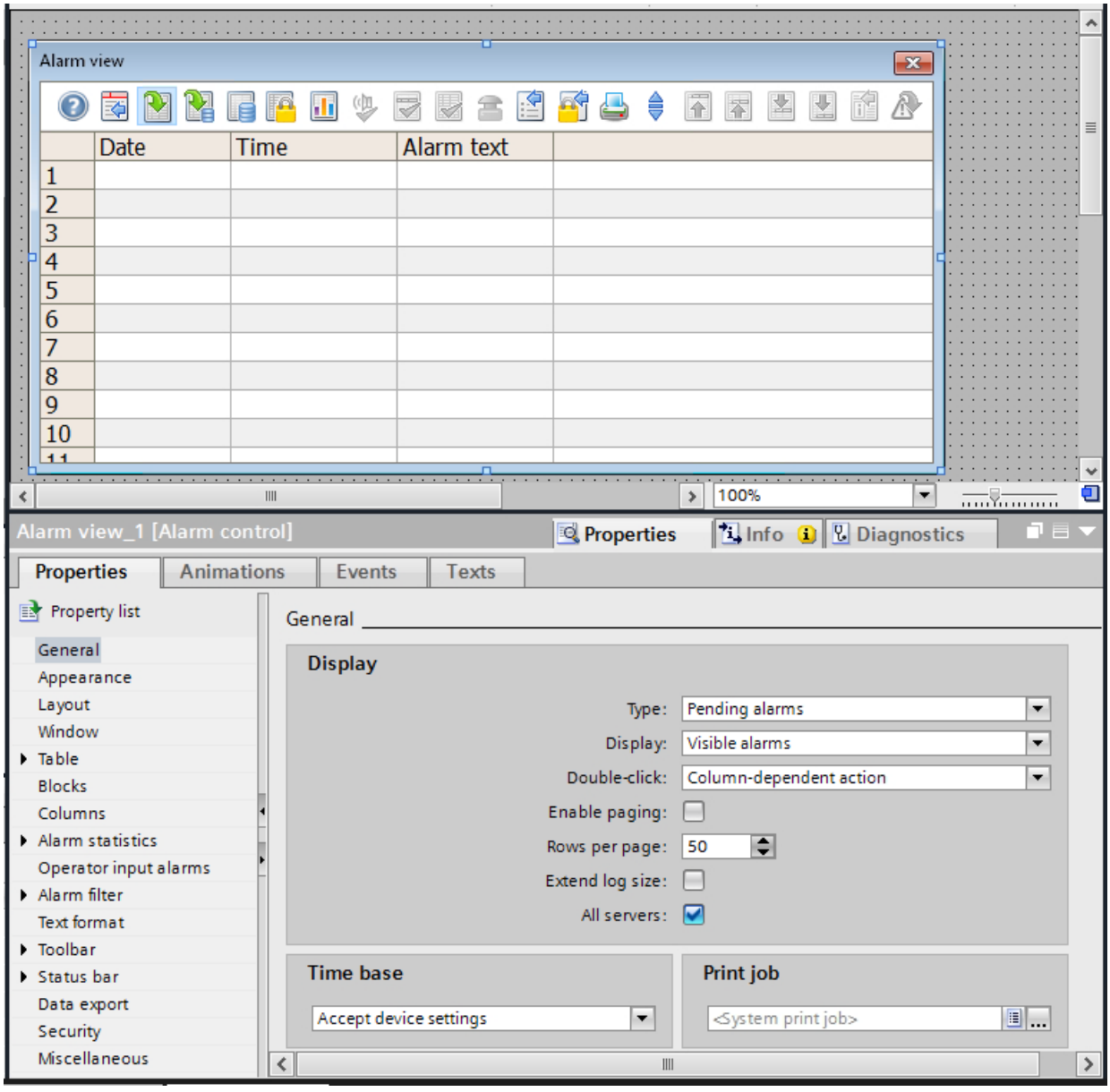

3. To specify the appearance of the alarm view, select "Properties > Properties > Appearance" in the Inspector window.
4. To specify the appearance of the headers, content, and table grid in the alarm view, select "Properties > Properties > Table" .
   – To specify the visualization of selected alarms in Runtime, select "Properties > Properties > Table > Table - Select".
   – To specify the sorting method for alarms in Runtime, select "Properties > Properties > Table > Table - Sorting".
5. To specify the sort order of alarms in Runtime, select "Properties > Properties > Alarm list" .

Enabling alarm text blocks for the alarm view

1. In the Inspector window under "Properties > Properties > Blocks > Inherit project settings for alarm text blocks", specify whether to apply the settings from the "Alarm text blocks" tab in the "HMI alarms" editor to this alarm view.
2. Select "Properties > Properties > Blocks" to enable the alarm text blocks to be visualized in the alarm view. You can specify the following properties for each alarm text block:
   - Name
   - Layout of the header
   - Layout of content
   - Alignment
   - Length
   - Format
   - Flashing

The contents can be represented either as text or as symbol. Representation as symbol is not available for all alarm text blocks.

Configuring the toolbar and status bar in the alarm view

1. To specify the the toolbar layout, select "Properties > Properties > Toolbar > Toolbar - General".
2. Select "Properties > Properties > Toolbar > Toolbar - Operator input elements", to enable the operator input elements to be included in the alarm view toolbar.
3. To specify the the status bar layout, select "Properties > Properties > Status bar > Status bar - General".
4. Select "Properties > Properties > Toolbar > Toolbar - Operator input elements", to enable the operator input elements to be included in the status bar of the alarm view.

Result

Alarms of various alarm classes are output in the alarm view during runtime. To change the view in Runtime, click the configured buttons on the alarm view toolbar.

See also

Configuring filters in the alarm view (Page 855)

Configuring Alarm Statistics (Page 856)

Configuring filters in the alarm view (RT Professional)

Introduction

WinCC supports the configuration of different filters for the alarm view:

- Standard filters
  The operator cannot access standard filters in Runtime. You can copy configured standard filter for use in other alarm views.
- Custom filters
  The operator can change the custom filter in Runtime. You are using custom filter only in a single alarm view.

Requirement

- The screen with the configured alarm view is open in the "Screens" editor.
- The Inspector window is open.

Procedure

1. Select the alarm view and click "Properties > Properties > Filter > Alarm filter" in the Inspector window.
2. To configure a custom filter, click on the "Custom" tab.
3. Define one or more filters. For each filter, specify one or more filter criteria under "Expression" with condition and value.

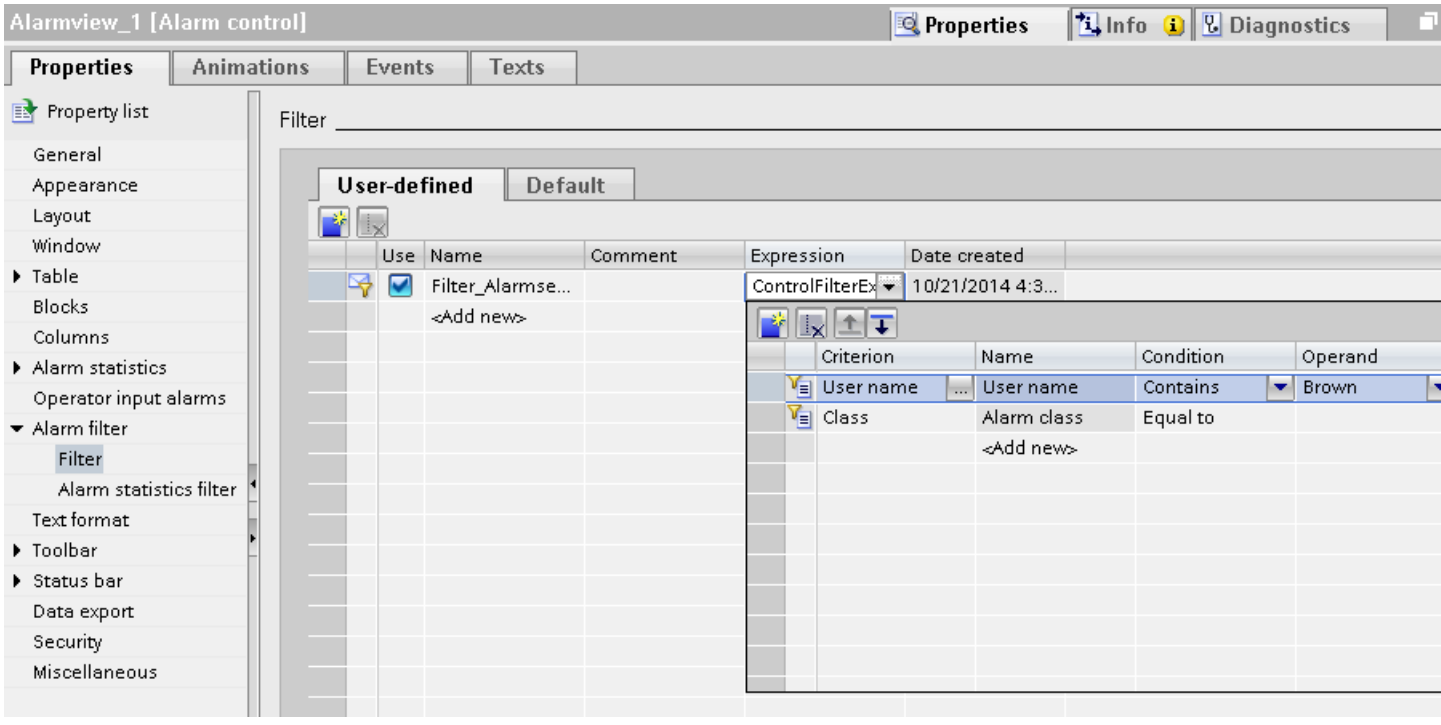

4. To configure a standard filter, select the "Standard" tab.
   You can configure several standard filters and activate these for the respective alarm view.
5. Enter the name for the standard filter and a comment.
6. To configure the filter, select the criteria, conditions, and the value for the standard filter in the "Filter" dialog.

Filter by process values

You cannot enter text as the criterion for a process value. To filter by the text of a parameter block, or by the process tag represented, proceed as follows:

1. Insert this process value as the alarm text for a user text block in an alarm.
2. In Runtime, filter the alarms by the text in the "Alarm text" alarm text block using the button.

In multi-user systems, make sure that contents displayed in the "Specify selection" dialog on a client have the same names on all servers.

Filter by time

When filtering by time, the start and stop values are not adjusted automatically when the time base of the alarm view is changed.

Example

At a PC location with time zone "UTC + 1h", the alarm view has the "Local time zone" time base. You should filter by the time 10:00 to 11:00. Convert the time base to "UTC". To display the same alarms as before, change the start or stop value of the filter to 9:00 and 10:00.

See also

SQL Statements for Filtering the Alarm View (Page 960)

Configuring Alarm Statistics (RT Professional)

Introduction

The alarm statistics contain statistical calculations for the logged alarms in the alarm view. The alarm statistics also contain a selection of the configured alarm text blocks. If the contents are dynamic, the alarm text blocks display the data for the last incoming alarm. You can compile the alarm statistics columns individually.

The alarm statistics contain the following statistical calculations:

- Alarm rate
  The system counts how often an alarm with the status "Incoming" is included in the log. If the alarm number does not occur, this alarm number is not included in the statistics.
- Total display time of an alarm in seconds.
  The following periods between alarm states are available as calculation variables:
  - "Incoming/Incoming"(sum +/+),
  - "Incoming/Outgoing"(sum +/-)
  - "Incoming/Initial acknowledgment"(sum +/*1)
  - "Incoming/second acknowledgment"(sum +/*2)

  Each type of calculation can be represented in a column in the alarm statistics.
- Average display time of an alarm in seconds
  The following periods between alarm states are available as calculation variables:
  - "Incoming/Incoming"(mean value +/+),
  - "Incoming/Outgoing"(mean value +/-)
  - "Incoming/Initial acknowledgment"((mean value +/*1)
  - "Incoming/second acknowledgment"((mean value +/*2)

  There is one column in the alarm statistics for each type of calculation.

The calculation of the time of acknowledgment includes the "Acknowledged" alarm status. This status includes both an emergency acknowledgment and an acknowledgment by the PLC.

Requirement

- The "Screens" editor is open.
- You have created a screen with an alarm view.

Configuring the alarm statistics functions

Proceed as follows to configure the functions of the alarm statistics:

1. Select the alarm view and click "Properties > Properties > Alarm statistics - General" in the Inspector window.

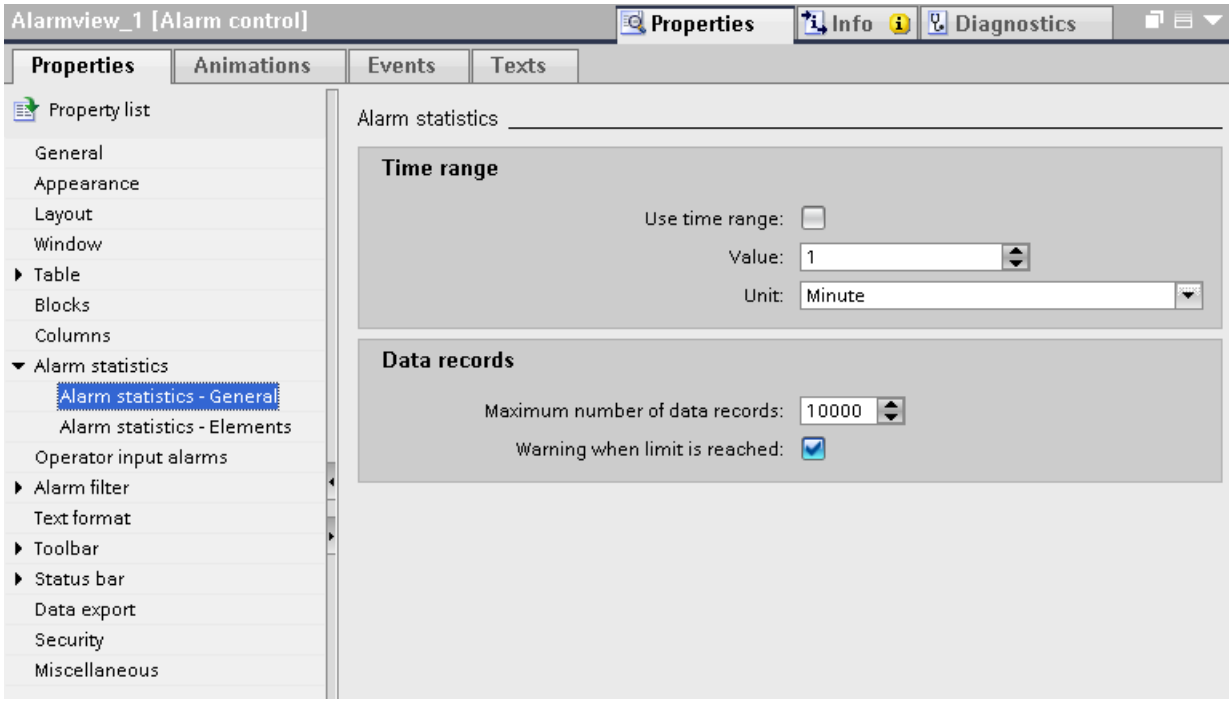

2. To limit the calculation period, activate the "Use time range" option under "Time range". Specify the frequency.
3. To define the maximum number of alarms to be processed in the alarm statistics, enter a value at "Data records > Maximum number of data records".
   This setting requires the following conditions to be met:
   – The number of alarms in the calculation period exceeeds the configured "Maximum number of data records which can be read".
   – The calculation period is not configured.
4. To output a warning when this number is exceeded, activate the "Warning when limit is reached" option.

Setting up the alarm statistics

Proceed as follows to set the columns of the alarm statistics:

1. Select the alarm view and click "Properties > Properties > Alarm statistics - Elements" in the Inspector window.

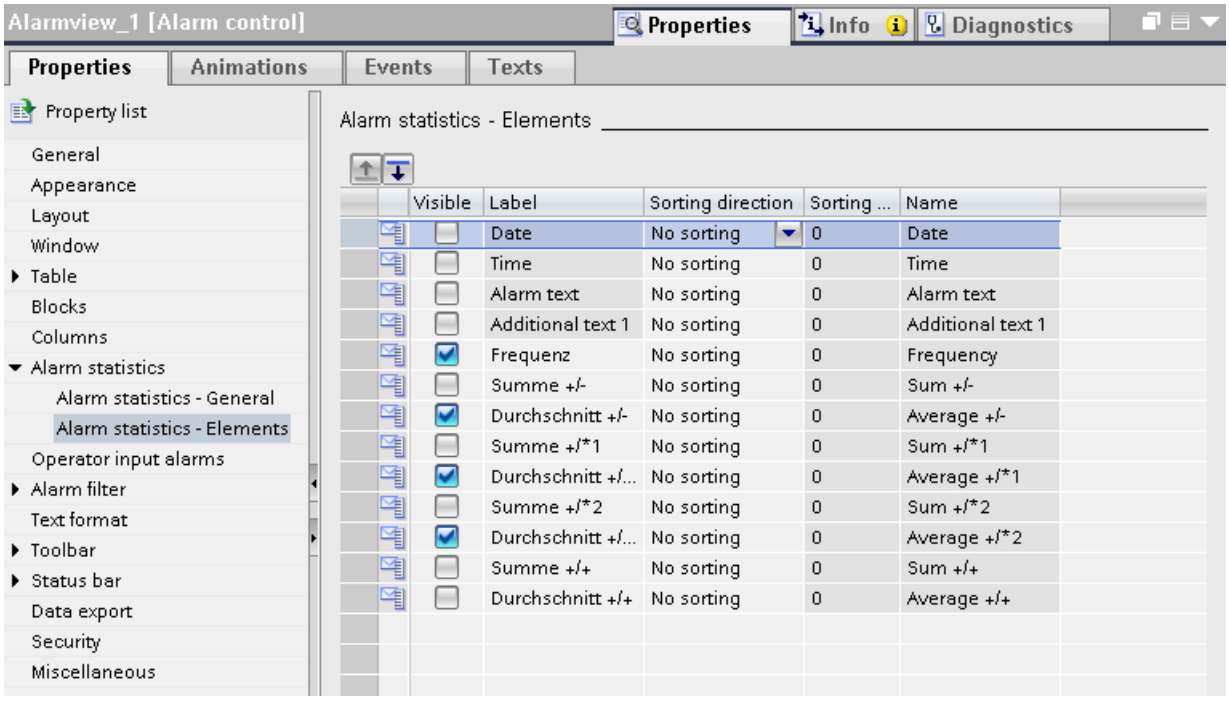

2. In the table, enable the alarm statistics columns that are to be displayed in Runtime.
3. To specify the arrangement of columns in the alarm view in Runtime, select a row in the Inspector window. Move the row to the selected location using the ⬆ and ⬇ buttons. The columns of the alarm statistics are displayed in Runtime in the same order as the rows are arranged in the Inspector window.
4. Specify the sort direction for each row in "Sort by".
5. Specify the sort order in "Sort order" by assigning values from 1 to 4.

Note

The sort function is only active if the alarm statistics view contains no more than four columns.

Result

In Runtime, the operator clicks the button in the alarm view to view the "Alarm statistics".

Note

Alarms that have the "acknowledged" and "outgoing" states are only used for the calculation if a suitable alarm with "incoming" state was found in the amount of the result .

If you acknowledge an alarm with single acknowledgment or dual acknowledgment only once, the "Incoming/First Acknowledgement" and "Outgoing/Second Acknowledgement" alarm states are used for the statistical calculation.

If there is an alarm pending at the PLC and Runtime is disabled and enabled several times, the alarm is entered several times in the log with "incoming" status. The alarm will also be included multiple times in the evaluation.

Configuring Alarm Display Suppression (RT Professional)

Overview of Alarm Display Suppression (RT Professional)

Introduction

Suppress the display of alarms to avoid an excessive burden of information for the plant operator. If only selected alarms are shown, the operator will concentrate his attention on essential alarms.

Requirement

The "Show alarm" and "Hide alarm" buttons are configured in the alarm view.

Methods for suppressing alarm displays

There are two options for suppressing the display of incoming alarms:

- **Automatic display suppression**
  The alarms are displayed in line with the specific plant status. You can enter these plant states as display suppression masks in each alarm. The design engineer specifies the values for the plant states.
  To transfer the current plant status to the alarm system, create an alarm group with a display suppression tag. This display suppression tag returns the value of the plant states.
  Assign this alarm group the alarms that the system does not display in Runtime at specific plant states.
  Example:
  In the "Shutdown" plant status you want to suppress the display of alarms that normally occur in this status. The "Online connection to the PLC was dropped" alarm will only not be displayed during "Shutdown".

  Note

  If you have specified a display suppression mask for an alarm that is not assigned to an alarm group, this alarm will not be suppressed in Runtime.

  Alarms with display suppression masks that are not allocated to any alarm group will not be suppressed in Runtime. To suppress the display of specific alarms in Runtime, you always have to assign these to an alarm group.

  Note

  You only use display suppression tags only for custom alarm groups.

- **Manual display suppression**
  When necessary, the user suppresses the display of an alarm in Runtime using the "Hide alarm" button in the alarm view.
  The user displays the alarm again when necessary using the "Show alarm" button.
  The system displays the alarms again after a configurable time.
  To enable the reporting of manual display suppression, configure a corresponding user alarm that is triggered when the display is suppressed manually. Configure this user alarm in the alarm view as an operator input alarm for the "Suppress" operator action.

Properties of alarms whose display is suppressed

The following applies to alarms whose display is suppressed:

- In the following alarm view display modes, you can use the following buttons to show the alarms again:
  - Display "Current alarms"
  - Display "Historical alarm list" (short-term)
  - Display "Historical alarm list" (long-term)
- The alarms are recorded and displayed under the "Hidden alarms" display.
- The alarms are also logged.
- If an alarm annunciator is set up for these alarms, it will not be triggered.

Manual display suppression

The following properties apply only to alarms whose display is suppressed **manually**.

- The alarms are shown again after a set time.
- An alarm that requires acknowledgment is acknowledged by the manual suppression of its display.
- To enable the reporting of manual display suppression, configure a corresponding user alarm that is triggered when the display is suppressed manually.

Automatic display suppression

The following properties apply only to alarms whose display is suppressed **automatically**.

- The display suppression does not trigger an operator input alarm.
- The PLC acknowledges alarms that require acknowledgment.
  In the time while alarms requiring acknowledgment are suppressed, the PLC acknowledges the outgoing alarms. If an alarm does not have the "Outgoing" status, the PLC acknowledges it immediately.

NOTICE
Forfeiture of acknowledgment requirement With automatic display suppression, alarms requiring acknowledgment no longer have to be acknowledged. You should therefore only use automatic display suppression for alarms that do not require operator actions during the configured plant states.

Interaction

The two methods of suppressing alarm displays interact as follows:

- When an alarm is automatically suppressed, use the "Hide alarm" and "Show alarm" buttons to show and hide the alarm as required.
- You have manually suppressed the display of an alarm using the button "Hide alarm". When the plant status changes and the conditions for automatic suppression of this alarm display are not fulfilled, the alarm is displayed once more.
  The time defined as the display suppression duration must not have elapsed.
- You have suppressed the alarm manually with the "Hide alarm" button. The status that was defined for automatic display suppression using the display suppression tags occurs. After the specified display suppression timeout was triggered, the alarm nevertheless remains suppressed.

Note

Hidden system diagnostic alarms

For hidden system diagnostic alarms, the alarm texts are not displayed in the list of hidden alarms.

The list of hidden alarms merely displays the configuration data of the alarm and not a live view.

To display active hidden system diagnostic alarms, you must adjust the filter for the online alarm list.

See also

Tags of an alarm group (Page 778)
Alarm view (Page 420)

Setting the Display Suppression Timeout (RT Professional)

Introduction

You can set the time after which manually-suppressed alarms should be automatically displayed once more in the alarm settings.

Requirements

- The project navigator is open.

Procedure

To set the alarm display suppression timeout, follow these steps:

1. Double-click "Runtime settings > Alarms" in the project navigation.
   The alarm settings open.
2. Select "General > Display suppression timeout" to configure the period during which a manually suppressed alarm remains hidden.
   30 minutes are preset.

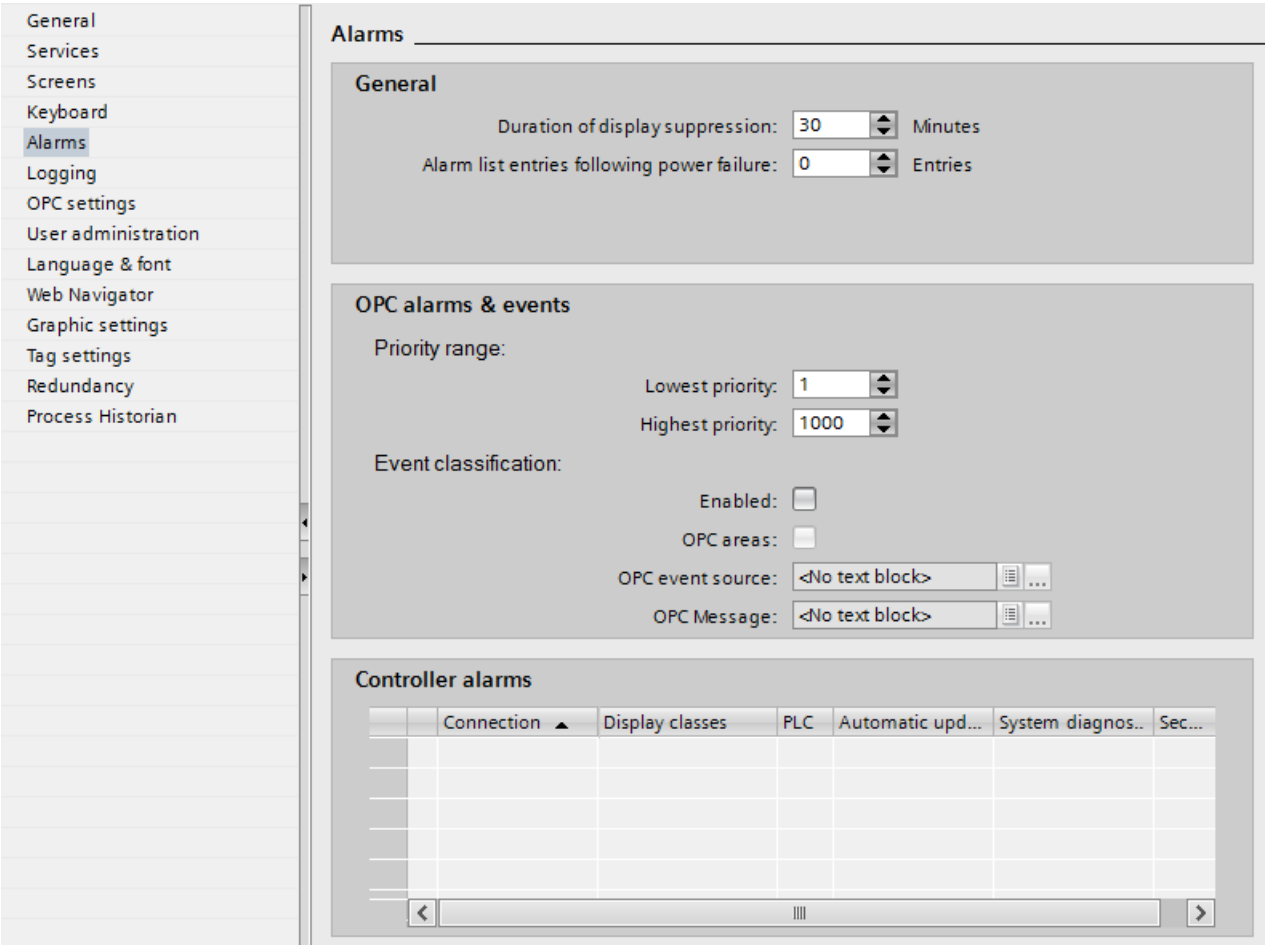

See also

Configuring Automatic Display Suppression (Page 865)

Configuring Automatic Display Suppression (RT Professional)

Introduction

With automatic display suppression, the alarm is displayed in relation to the status of the PLC. You define the plant status that causes an alarm to be hidden. Configure automatic display suppression using the "HMI alarms" editor. You will need to create an alarm group with a displace suppression tag. In the alarm, you assign the plant statuses using the display suppression mask. The display suppression tag must assume the value of the display suppression mask for an alarm so that it is not displayed in Runtime.

Requirement

The "Show alarm" button is configured in the alarm view.

Requirements

- The "HMI alarms" editor is open.
- The "Alarm groups" tab is selected.
- Alarms are set up.
- The display suppression tag must exist.

Creasing an alarm group for display suppression

1. Double-click "Add" in the table.
   A new alarm group is created.
2. Enter a name for the new alarm group.
3. Select the display suppression tag under "Tags" in the Inspector window.

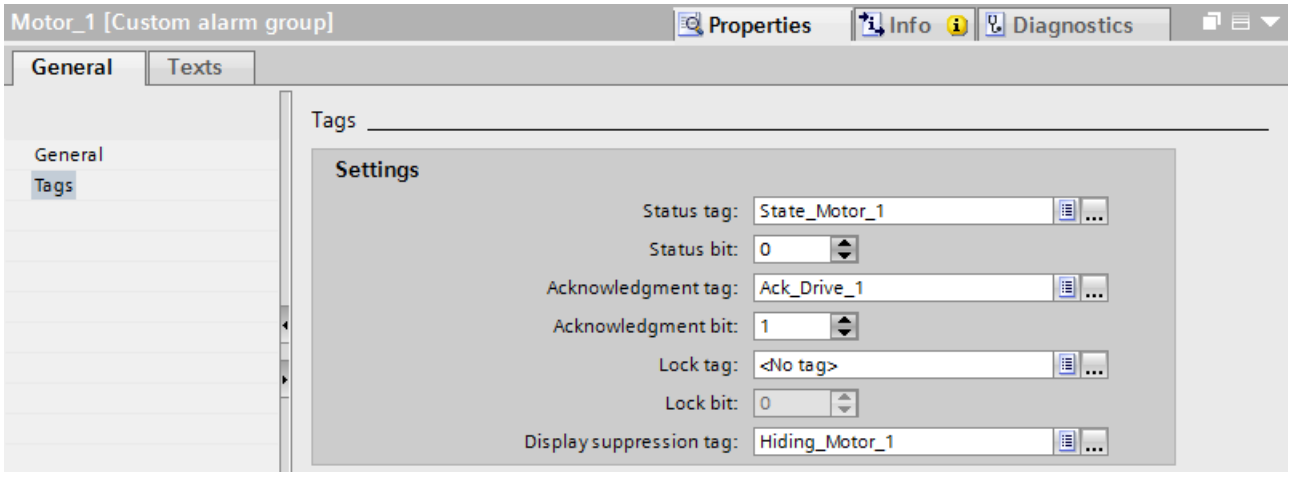

Creating a display suppression mask for the alarm

1. Select the alarm.
2. To select the alarm group for which you created the alarm suppression tag, select "Properties > Properties > General" in the Inspector window.

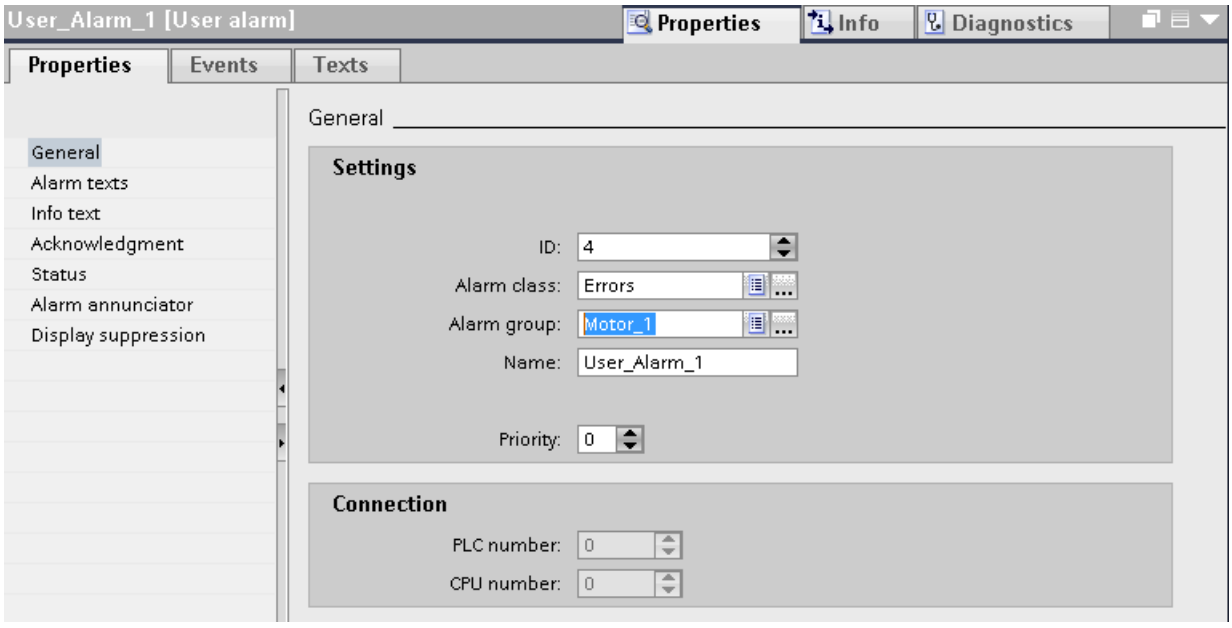

3. To enter the display suppression mask, select "Properties > Properties > Display suppression" in the Inspector window.

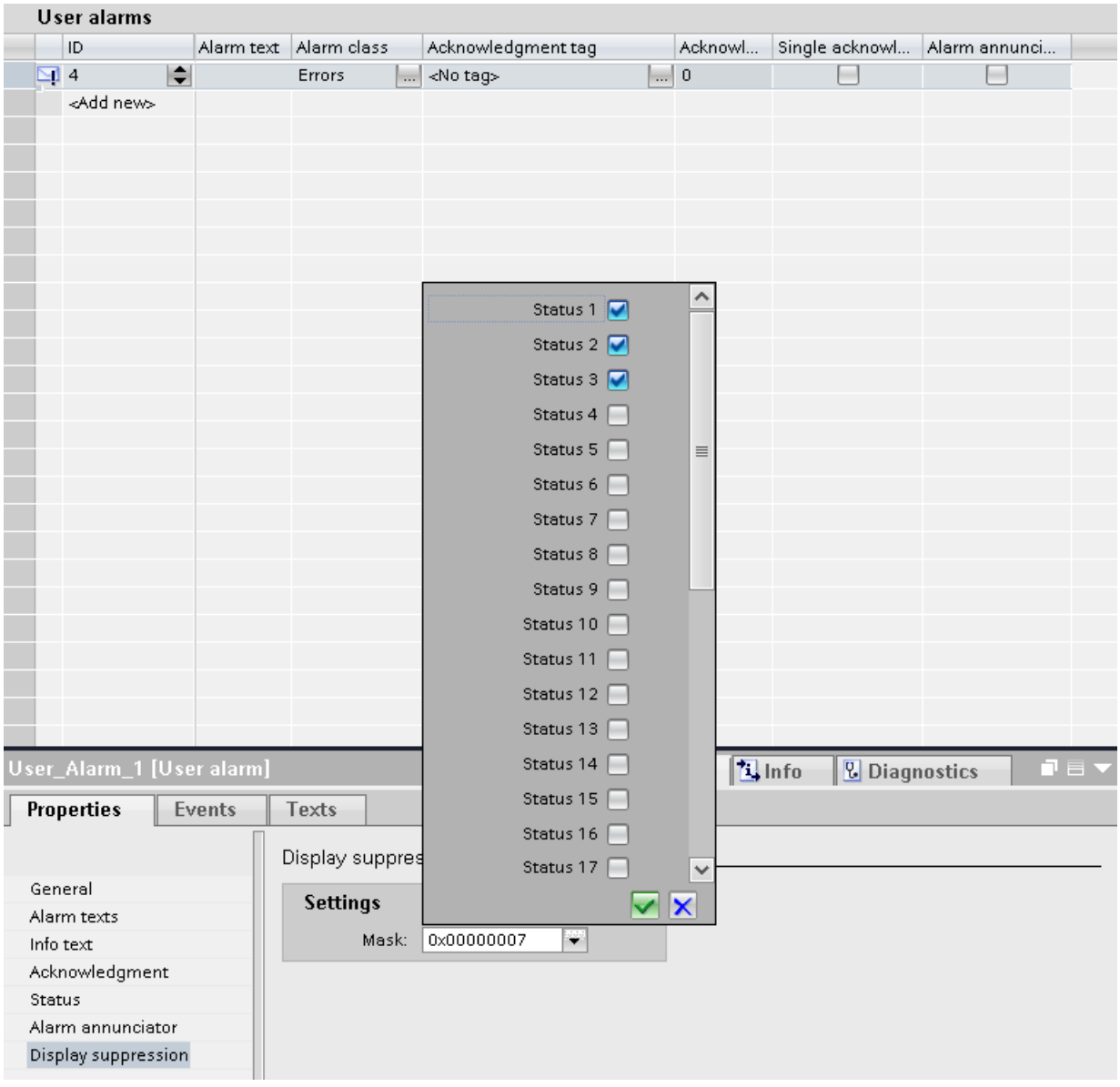
User alarms
ID
Alarm text
Alarm class
Acknowledgment tag
Acknowl...
Single acknowl...
Alarm annunci...
4
Errors
<No tag>
0
<Add new>
Status 1
Status 2
Status 3
Status 4
Status 5
Status 6
Status 7
Status 8
Status 9
Status 10
Status 11
Status 12
Status 13
Status 14
Status 15
Status 16
Status 17
User_Alarm_1 [User alarm]
Info
Diagnostics
Properties
Events
Texts
General
Alarm texts
Info text
Acknowledgment
Status
Alarm annunciator
Display suppression
Display suppres
Settings
Mask:
0x00000007

Result

The alarm is hidden when the value set in the display suppression mask is activated at the display suppression tag. This means that the alarm is hidden when the plant is in the specified status.

Note

Alarms with display suppression masks that are not allocated to any alarm group will not be suppressed in Runtime. You should therefore always assign alarms to an alarm group if you want to suppress them.

If an alarm is hidden, it can be shown once again in Runtime by clicking the button "Show alarm".

See also

Setting the Display Suppression Timeout (Page 863)

Alarm view (Page 420)

Configuring Alarm Alerts (RT Professional)

Introduction

In addition to the alarm view, WinCC also supports alarm indication in the plant by means of visual or audible signals. The alarm annunciator can be a horn or a warning lamp, for example. You can also display and acknowledge the alarm annunciator in a screen.

Requirements

- The "Alarm annunciator acknowledgment" button is configured in the alarm view.
- The "HMI alarms" editor is open.
- Alarms are set up.
- The Inspector window is open.
- Your plant is equipped with an alarm annunciator which is connected to a tag.

Configuring alarms for the alarm annunciator

1. Select the alarm that you want to display using an alarm annunciator.
2. In the Inspector window, select "Properties > Alarm annunciator".
3. In the "Alarm classes" tab, select the alarm class of the alarms for the alarm annunciator.
   - To acknowledge the alarm annunciator via the "Alarm annunciator acknowledgment" button of the alarm view, activate "Acknowledge > Alarm annunciator > By means of "Acknowledge central annunciator" button" in the Inspector window.
     To switch off the alarm annunciator, use the "Alarm annunciator acknowledgment" button. The alarm is still pending in the alarm view.
   - To acknowledge the alarm via the alarm annunciator, select the tag to which the alarm annunciator is connected.

Configuring alarm annunciator in the screen

1. Open the screen in which you display the alarm annunciator.
2. For example, configure a circle and an associated button with the label "Alarm annunciator".
3. At the event "Click", configure the button that runs a VBS script with the method "QuitHorn".
4. Dynamize the background color of the circle with the tag to which the alarm annunciator is connected.

Result

If an alarm is displayed with the alarm annunciator in Runtime, the user acknowledges the alarm as follows depending on its configuration:

- Using an "Alarm annunciator acknowledgment" button in the alarm view
- Using the configured "Alarm annunciator" button

The user can recognize the status of the alarm annunciator from the background color of the circle.

See also

Configuring acknowledgment of the alarm annunciator (Page 882)

Alarm view (Page 420)

Displaying logged alarms (RT Professional)

Introduction

An alarm view displays logged alarms in Runtime according to the configuration. The alarm view displays alarms from the log in the "Historical alarm list" (short-term) or "Historical alarm list" (long-term) display. The "Historical alarm list" (short-term) also displays the current alarms.

Requirement

- The "Historical alarm list (short-term)" and "Historical alarm list (long-term)" buttons are configured in the alarm view.
- The logged alarms are located on the local log server.
- The screen with the configured alarm view is open.
- The Inspector window is open.

Procedure

Proceed as follows to configure an "Historical alarm list" (short-term) or "Historical alarm list" (long-term) display:

1. Select the alarm view.
2. Select "Properties > Properties > General > Active list > "Historical alarm list" (short-term) or "Historical alarm list" (long-term) in the Inspector window.
3. Configure the alarm view similar to an alarm view for active alarms.

Result

In Runtime, you can view logged alarms using the "Historical alarm list (short-term)" or "Historical alarm list (long-term)" buttons. To visualize only alarms of the "Errors" alarm class or alarms at certain plant areas, you can filter or sort the logged alarms.

Note

In the "Historical alarm list" (short-term) display, the new incoming alarms are updated immediately. In the "Historical alarm list" (long-term), on the other hand, they are not. You cannot enter or view comments in the "Historical alarm list" (short-term). Comments can only be entered in the "Historical alarm list" (long-term).

See also

Alarm view (Page 420)

Configuring alarm export (RT Professional)

Introduction

To export alarms to a "*.csv" file in Runtime, the operator clicks on the button in the alarm view.You can configure this function in the alarm view in the "Screens" editor.

Requirements

- The screen with the configured alarm view is open.
- The Inspector window is open.

Procedure

To configure the export of alarms, proceed as follows:

1. Select the alarm view and configure data export by selecting "Properties > Properties > Data export" in the Inspector window.
   - You can specify a custom file name and folder.
   - At "Scope", specify the data to be exported from the alarm view.
   - at "CSV format", specify the separator to use in the ".csv" file.
2. Select "Operator input in Runtime" to configure the function of the button in Runtime:
   - Select "Display dialog" to open the "Default settings data export" dialog.
   - To export the data directly to the specified export file, disable "Display dialog".
3. Additionally, select "Operator input in Runtime" to specify whether or not the operator is allowed to rename the file and/or change the folder in Runtime.

Configuring alarm evaluation in Runtime (RT Professional)

Introduction

To enable evaluation of the alarms in Runtime, assign a custom C function to the "Status changed" event of an alarm. Edit the function to suit your requirements for evaluation in Runtime.

Requirement

- The "HMI alarms" editor is open.
- A custom function for reading the data has been created.
  This function must contain a pointer to a string whose data is mapped by scanf to the "MSG_RTDATA_STRUCT" structure.

Procedure

To configure the evaluation of an alarm in Runtime, proceed as follows:

1. Select the alarm.
2. In the Inspector window, select "Properties > Events > Status changed".

3. Select the custom function for reading the data.

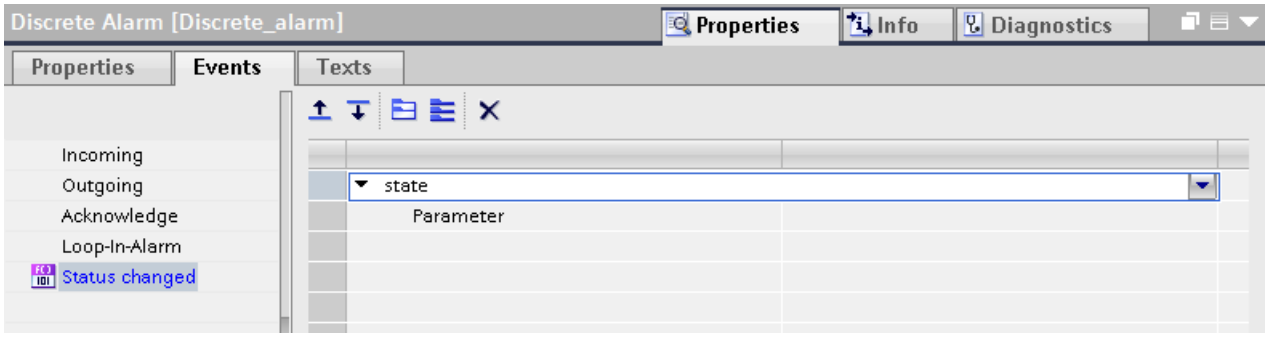

The alarm data is read in Runtime. Alarm data includes,for example, the alarm number, time stamp, and alarm state.

4. In the "Scripts" editor, configure evaluation of the alarm data within the function.

Result

When the alarm state changes, the alarm data is read in Runtime and evaluated according to your configuration.

Example

Use the following, user-defined function "state" to read the status of alarms, for example:

```
Void state (MSG_RTDATA_STRUCT Parameter)
DWORD state;
state = Parameter.dwMsgState;
printf("Status: %d\r\n", state);
```

3.2.3.4 Reporting Alarms (RT Professional, RT Advanced, Panels, Comfort Panels)

Configuring reporting of alarms (RT Advanced, Panels, Comfort Panels)

Overview

WinCC can report on all alarms that occur in the system. The following options are available:

- Output an alarm sequence report
  In Runtime, the standard printer of the HMI device will continuously print each alarm and any changes in its status.
- Output of an alarm report
  You configure an alarm report in the "Reports" editor and specify when it is output in Runtime:
  - For event-driven output, configure an object that is assigned the "PrintReport" system function. The object can be a button or tag.
  - For time-driven output, create a "Print job" in the scheduler. Assign an alarm report to the job.

  Note

  In the alarm report, specify whether to output current or logged alarms.

Requirements

- A printer was configured on the HMI device.

Activating continuous alarm reporting

To activate continuous reporting of alarms, follow these steps:

1. Double-click "Runtime settings" in the project navigation.
2. Select "Alarms > General > Report".

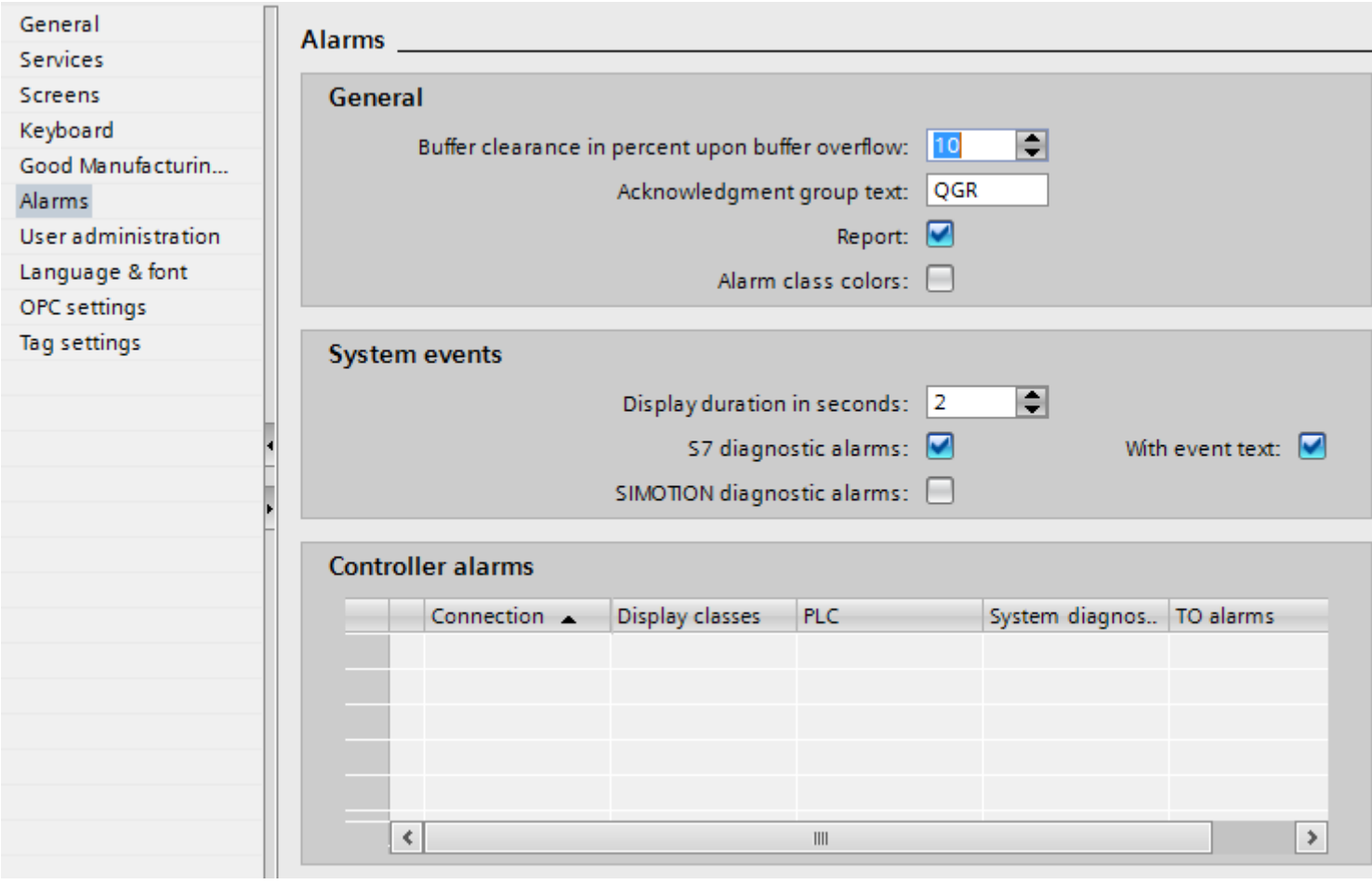

Result

The latest alarms are output at the printer of the HMI device.

Excluding an alarm from reporting

To exclude specific alarms from reporting, proceed as follows:

1. Open the "HMI alarms" editor.
2. Select the alarm to exclude from reporting on the tab of the selected alarm type.
3. In the Inspector window, disable "Properties > Properties > Miscellaneous > Report".

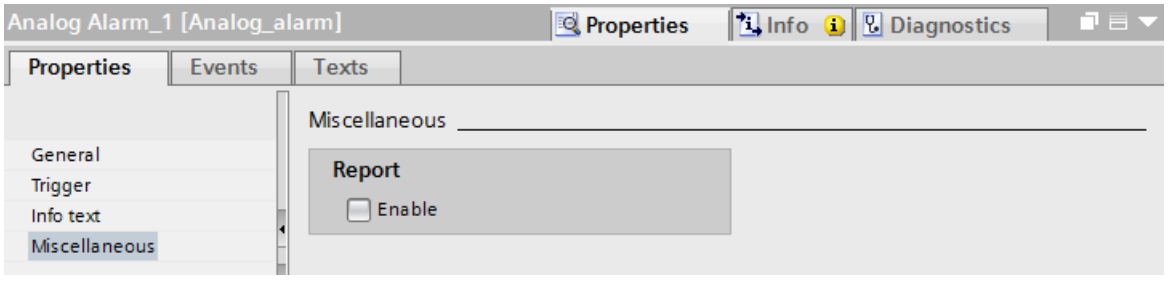

Result

WinCC does not output these alarms currently on the connected printer.

See also

Principles for preparation of reports (Page 1231)

Create a report (Page 1238)

Report output in Runtime Advanced (Page 1251)

Alarm report (Page 1295)

Working with reports (Page 1231)

Configuring reporting of alarms (RT Professional)

Overview

WinCC logs all system alarms as required.The following options are available:

- Output the current display in an alarm view
  You can print out a system report. You can also create a custom report, if necessary.In the print job, specify the print scope, page area, printer, or target file.
- Output an alarm sequence report
  In Runtime, every alarm is continuously output at a printer.

Requirements

- The "Screens" editor is open.
- Alarms are configured.
- You configured an alarm view with "Print current view" button.

Procedure

To configure the outputting of the current display in an alarm view using your own report, proceed as follows:

1. Select the required alarm view.
2. To select the print job, select "Properties > Properties > General > Print > Print job" in the Inspector window.
   The default print job is the system print job. You can also select a customized print job, or a modified system print job.

Result

If the user clicks the button in Runtime, the selected report with the alarms currently displayed in the alarm view is output to the printer.

Note

Printer settings

Configure the printer in the HMI device's Control Panel. The settings will depend on which operating system is used on the HMI device.

For more detailed information, refer to the operating instructions for your HMI device.

See also

Working with reports (Page 1231)

3.2.4 Configuring acknowledgment (RT Advanced, Basic Panels, Panels, Comfort Panels)

3.2.4.1 Configuring alarm acknowledgment by means of alarm class (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

To configure an alarm with alarm acknowledgment, assign it to an alarm class with the "Alarm with single acknowledgment" state machine.

Requirement

- The "HMI alarms" editor is open.
- The required alarm class has been created.
- The required alarm has been created.

Selecting the state machines for an alarm class

The state machine for a predefined alarm class has already been set. You can only set the state machines for user-defined alarm classes. Proceed as follows:

1. In the "HMI alarms" editor, click the "Alarm class" tab and select the alarm class.
2. Select the required acknowledgment model under "Properties > Properties > Acknowledgment" in the Inspector window.

Assign alarms to an alarm class requiring acknowledgment

Proceed as follows to assign an alarm to an alarm class requiring acknowledgment.

1. In the "HMI alarms" editor, click the tab for the alarm type and select the alarm.
2. Under "Properties > Properties > General" in the Inspector window, select the alarm class of the alarm.

Result

The alarm will not disappear in Runtime until it is acknowledged by the operator.

3.2.4.2 Configuring trigger for alarm acknowledgment (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You always specify the acknowledgment requirement for an alarm using the alarm class. Then the operator acknowledges the alarm using the "ACK" function key of the HMI device or the "Acknowledgment" button of the alarm view.

The following options are also available to trigger acknowledgment:

- Configuring a button to acknowledge an alarm
- Acknowledgment of a Discrete Alarm by the PLC

Requirements

- The "HMI alarms" editor is open.
- The required alarm class has been created.
- The required alarm has been created.
- An alarm view and a button are created in the "Screens" editor.

Configuring a button to acknowledge an alarm

To configure a button for acknowledging an alarm, proceed as follows:

1. Select the button in the "Screens" editor.
2. Under "Properties > Events" in the Inspector window, assign the "AlarmViewAcknowledgeAlarm" system function to the "Click" event.
3. Select the alarm view as parameter.

Acknowledgment of a Discrete Alarm by the PLC

1. In the "HMI alarms" editor, click the "Discrete alarm" tab and select the discrete alarm.
2. In the Inspector window, select the tag and the bit that acknowledges the PLC alarm under "Properties > Properties > Acknowledgment > PLC".

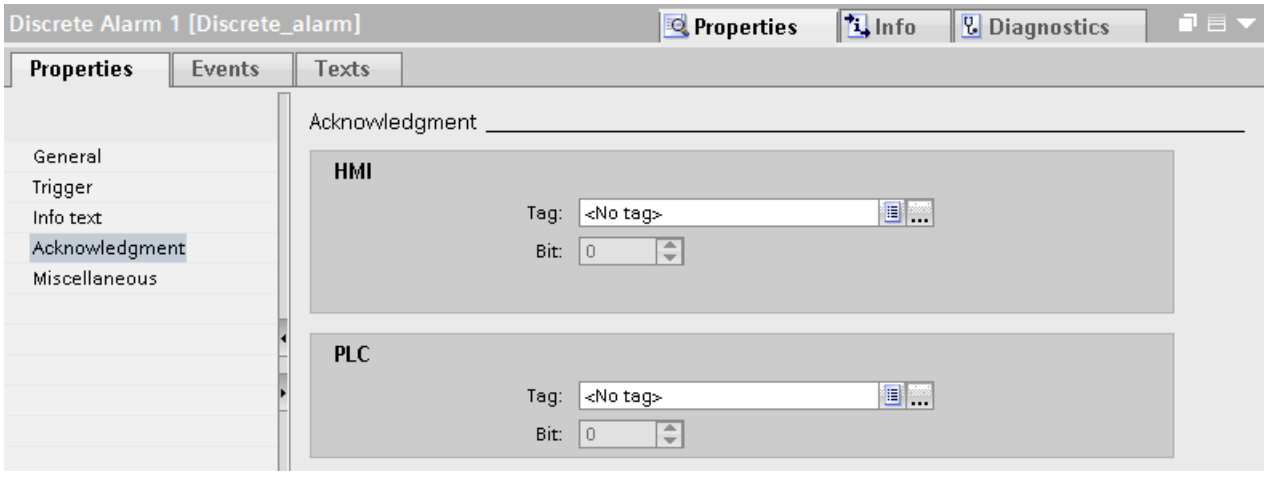

Acknowledging alarms in an alarm group

The various acknowledgment options have the following effect in alarm groups:

- Operator acknowledges the alarm
  You acknowledge the alarm of an alarm group, for example, using the <ACK> acknowledgment key or a function key. All alarms of the alarm group are acknowledged.
- The PLC acknowledges the alarm
  The alarm of an alarm group is acknowledged by setting a bit in the tag.
  Only the alarm to be acknowledged is acknowledged.

3.2.4.3 Sending alarm acknowledgments to the PLC (RT Advanced, Basic Panels, Panels, Comfort Panels)

Requirement

- The "HMI alarms" editor is open.
- The required alarm has been created and assigned to an alarm class requiring acknowledgment.

Note

You cannot send the acknowledgment of analog alarms to the PLC.

Sending alarm acknowledgments to the PLC

To configure that acknowledgment of an alarm is sent to the PLC, follow these steps:

1. In the "HMI alarms" editor, click the "Discrete alarm" tab and select the discrete alarm.
2. In the Inspector window, select "Properties > Properties > Acknowledgment".
3. Under "HMI", select the tag and the bit set by the alarm acknowledgment function.

Note

The HMI device and PLC only have read access to the acknowledgment tag memory area.

Result

If the operator acknowledges the alarm in Runtime, the operating step is forwarded to the PLC.

3.2.5 Configuring acknowledgment (RT Professional)

3.2.5.1 Configuring alarm acknowledgment (RT Professional)

Introduction

The alarm classes define how the alarms from an alarm class are to be acknowledged. When you assign an alarm to an alarm class, you define the state machine for that alarm.

Requirements

- The "HMI alarms" editor is open.
- The required alarm class has been created.
- The required alarm has been created.

Procedure

To configure the acknowledgement of an alarm, follow these steps:

1. In the "HMI alarms" editor, click the "Alarm class" tab and select the alarm class.
2. To select the state machine, select "Properties > Properties > Acknowledgment > Settings" in the Inspector window.
   - If you enable "Assign comment to defined user", this Runtime user writes a comment on the alarm acknowledgment that is written to the log file alongside with the alarm.
   - If you enable "Use comment of the incoming alarm", the same comment is attached to each status change of this alarm as the one assigned for the "incoming" alarm event.

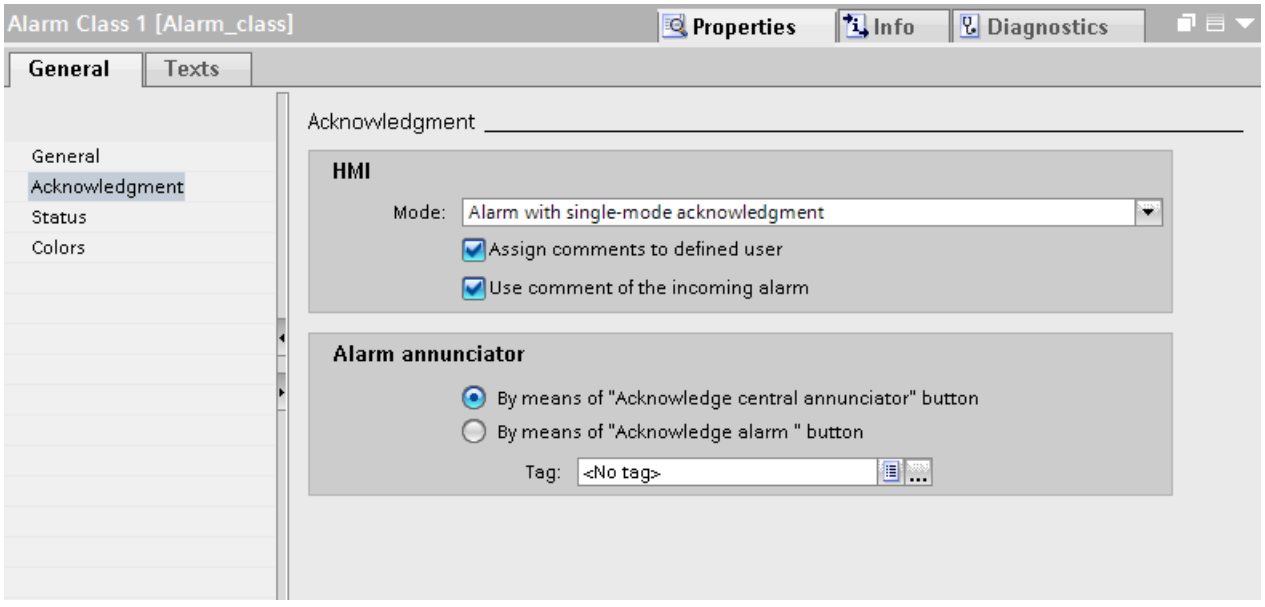

3. In the "HMI alarms" editor, select the alarm.
4. Select "Properties > Properties > General" in the Inspector window to select the alarm class of the alarm.
5. Select "Properties > Properties > Acknowledgement > Settings" in the Inspector window to specify the acknowledgment tag to which the alarm acknowledgment is saved, including the bit set for this purpose.
6. To exclude the alarm from the group acknowledgment of the alarm view, enable "Single acknowledgment".

Acknowledging an alarm via a system function

1. In the "HMI alarms" editor, select the alarm.
2. In the Inspector window, click "Properties > Events > Incoming".
   The function list opens.
3. In the "Function list" dialog, assign the "SetBit" system function to the "Incoming" event.
4. Enter the acknowledgment tag of the "BOOL" data type as the "Tag (In/Out)" parameter.

3.2.5.2 Configuring alarm group acknowledgment (RT Professional)

Introduction

You acknowledge an alarm group using the acknowledgment tag. To set and reset the acknowledgment bit, configure a display and operating element.

Requirements

- The "HMI alarms" editor is open.
- You have created the alarm groups.
- The required alarm has been created.

Procedure

To configure the acknowledgement of an alarm group, follow these steps:

1. In the "HMI alarms" editor, select the alarm.
2. Under "Properties > Properties > General" in the Inspector window, select the alarm group to which the alarm belongs.
3. Click the "Alarm groups" tab in the "HMI alarms" editor.
4. Select the alarm group.
5. Under "Properties > Properties > Tags > Settings" in the Inspector window, select the acknowledgment tag and acknowledgment bit.
6. Open the "Screens" editor and configure a button that sets the acknowledgment bit in Runtime for the acknowledgment tag.

Result

The user monitors the alarms for a given function together in Runtime. If a user identifies the cause of a fault, then he or she can acknowledge all alarms for this function using the configured operator action.

3.2.5.3 Configuring acknowledgment of the alarm annunciator (RT Professional)

Introduction

In addition to the alarm view, WinCC also supports alarm indication in the plant by means of visual or audible signals. The alarm annunciator can be a horn or a warning lamp, for example. You can acknowledge the alarm annunciator in the following ways:

- Using the button in the alarm view
  This "Acknowledge alarm annunciator" button must be configured in the alarm view. The button is always active, eve nother option for acknowledging the alarm annunciator has been enabled. Use the button to switch the alarm annunciator off. The alarm is still pending in the alarm view.
- Using a single acknowledgment
  You acknowledge the alarm annunciator with the alarm that triggered the ala annunciator. The operator acknowledges an individual alarm by clicking the "Single acknowledgment" button in the alarm view.
- Using a tag
  The specified tag is used to control the alarm annunciator.
  To acknowledge the alarm annunciator, e.g. with a button, use this option.

Requirements

- The "HMI alarms" editor is open.
- The required alarm class has been created.
- The required alarm has been created, and assigned to the alarm class.

Procedure

To configure the acknowledgment of an alarm annunciator, proceed as follows:

1. Click the "Alarm classes" tab in the "HMI alarms" editor.
2. Select the required alarm class.
3. In the Inspector window, select "Properties > Properties > Acknowledgment > Alarm annunciator" to specify how to acknowledge the alarm annunciator.

Result

The operator acknowledges the alarm annunciator in Runtime as specified for the alarm class of this alarm.

3.2.5.4 Configuring the Group Acknowledgement of Alarms (RT Professional)

Introduction

You can use the button in the alarm view to acknowledge all pending, and visible alarms that must be acknowledged in the alarm view. You can exclude individual alarms from group acknowledgment.After group acknowledgment was triggered, the excluded alarms are still pending.

Requirement

- A screen is open.
- An alarm view has been created.

Procedure

To set up a group acknowledgement, follow these steps:

1. Open the "Screens" editor, and select the relevant alarm view.
2. In the Inspector window, select "Properties Properties > Toolbar - Buttons > Group ackr dgment."
   The button appears in the alarm view. All alarms of an alarm view that are not excluded from group acknowledgment are acknowledged when the operator actuates this button.

Configuring a single acknowledgement

1. Open the "HMI alarms" editor and select the tab that contains the relevant alarm type.
2. Select the alarm that you want to exclude from the group acknowledgement.
3. Enable "Properties > Properties > Acknowledgment > Single acknowledgment" in the Inspector window.
   When you acknowledge all the alarms in an alarm view using the button, any alarms for which the "Single acknowledgement" option is enabled will be excluded. The operator acknowledges alarms with "single acknowledgment" option individually.

3.2.5.5 Configuring the Resetting of the Acknowledgment Bit for Acknowledgment Tags (RT Professional)

Introduction

The user or controller resets the acknowledgment bit of the acknowledgment tag. The acknowledgment bit is reset directly in the PLC or in WinCC Runtime via a script, for example. The acknowledgment bit is not automatically reset.

Resetting of the acknowledgment bit by the PLC

You can use an external tag as an acknowledgment tag to allow the PLC to reset the acknowledgment bit.

If the alarm is acknowledged in Runtime in WinCC, the defined acknowledgment bit is set in the PLC. In the PLC, the acknowledgment bit is processed as required before it is reset once more.

In the PLC, an alarm is triggered by a reusable function block or by functions. In this block the alarm can also be acknowledged or the acknowledgment bit reset.

Evaluation and resetting of the acknowledgment bit by the PLC has the following advantage:

It is easy to cause the acknowledgement with external signals, such as pushbutton units in the field.

You can acknowledge externally and via the alarm view. Both methods use the same bit in the automation system for acknowledgement.

Note

To ensure that the alarm view in WinCC reliably acquires the external acknowledgement in time, configure the resetting by the automation system with a time delay.

Resetting of the acknowledgment bit by the WinCC

If you want to acknowledge an alarm via the acknowledgment tag only, reset the acknowledgment bit immediately as soon as it is set.

If you acknowledge an alarm via the alarm view, configure the resetting of the acknowledgment tags separately.

You can reset the acknowledgment bit in the following ways in WinCC:

- Button
  You can reset the acknowledgment bit using a button.
- System function
  Use the "ResetBitInTag" system function. If the acknowledgment bit is set, it will be immediately reset.

  Note

  If the acknowledgment tag is of the "BOOL" data type, use the "ResetBit" system function.

  Note

  If lots of alarms are processed using scripts, this places a heavy load on the work memory. The acknowledgment bit should therefore ideally be reset by the PLC.

3.3 Alarm Logging (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

3.3.1 Configuring alarm logging (RT Advanced, Panels, Comfort Panels)

3.3.1.1 Basics on alarm logging (RT Advanced, Panels, Comfort Panels)

Introduction

An alarm log is used to record project alarms.

Note

Alarm logging is not available for every HMI device.

Configuration steps

To log an alarm, follow these configuration steps:

1. Creating an alarm log
   You define the following properties for the alarm log:
   - Logging method
     Behavior of the log when reaching a specific fill level
   - Storage location and file format
   - Log size
   - Runtime start characteristics
2. Assigning an alarm log to an alarm class
   You can log the alarms from multiple alarm classes in an alarm log.
3. Assigning an alarm to a loggable alarm class
4. Configure display of logged alarms in an alarm view

Content of the alarm log

All states are logged for configured alarms. The following three entries, for example, are stored in the log for an alarm that requires acknowledgment:

- 04.08.2007 10:00:25:520, analog alarm, ID5, K, error, fill level exceeded by 10%
- 04.08.2007 10:01:20:442, analog alarm, ID5, Q, error, fill level exceeded by 10%
- 04.08.2007 10:01:30:112, analog alarm, ID5, G, error, fill level exceeded by 10%

In the example, the alarm states are identified by the following letters:

K = Incoming

Q = Acknowledged

G = Outgoing

All data belonging to an alarm is stored in the alarm log, including configuration data such as the alarm class, time stamp, and alarm text.

The potential number of logged alarms depends on the data medium used. You can further process the logged alarms in other programs for analysis purposes, for example.

Note

Alarm text and point of error are only logged if you have configured such a setting in the log properties.

Logged alarms that contain alarm text and the point of error exceed the estimated size of the configured alarms. Check to see whether the specified storage location still has sufficient space.

Note

The time stamp of a logged alarm is always specified in standard UTC format (Universal Time Coordinated).

Logging methods

The logging method determines how the alarm log responds when the configured size is reached. WinCC supports the following logging methods:

- Circular log
  When the configured log size has been reached, the oldest entries are deleted. When the configured log size has been reached, approximately 20% of the oldest entries are deleted. It is therefore not possible to display all the configured entries. During configuration, select an appropriate size for the circulation log. Alternatively, configure a segmented circular log.
- Segmented circular log
  In a segmented circular log, multiple single logs of the same size are filled in succession. When all logs are completely full, the oldest log is overwritten.
- Log with level-dependent system alarm
  When a defined level is reached, such as 90 %, a system alarm is triggered. When the configured size of the log has been reached, new alarm events are not logged.
- Log with level-dependent execution of system functions
  When the log is completely full, the "Overflow" event is triggered. You configure a function list for the event.
  When the configured size of the log has been reached, new alarm events are not logged.

Displaying logged data

On the HMI device, the logged data is displayed in an alarm view that is configured for this purpose.

See also

Working with Logs (Page 1027)

3.3.1.2 Create an alarm log (RT Advanced, Panels, Comfort Panels)

Introduction

In Runtime, you save alarms to the alarm logs. Specify the alarm log in the alarm class. An alarm log contains the alarms of several alarm classes. Create the alarm log in the "Logs" editor. When creating an alarm log, define the following parameters:

- Name
- Size
- Storage location
- Runtime start characteristics
- Log type
- Advanced content
- Checksum

You can also enter comments for each log.

Requirement

- The "Alarm log" tab is open in the "Log" editor.
- The Inspector window is open.

Procedure

To create an alarm log, proceed as follows:

1. Double-click "<Add>" in the table.
   A new alarm log is created.
2. Under "Properties > Properties > General" in the Inspector window, enter a unique name for the alarm log.

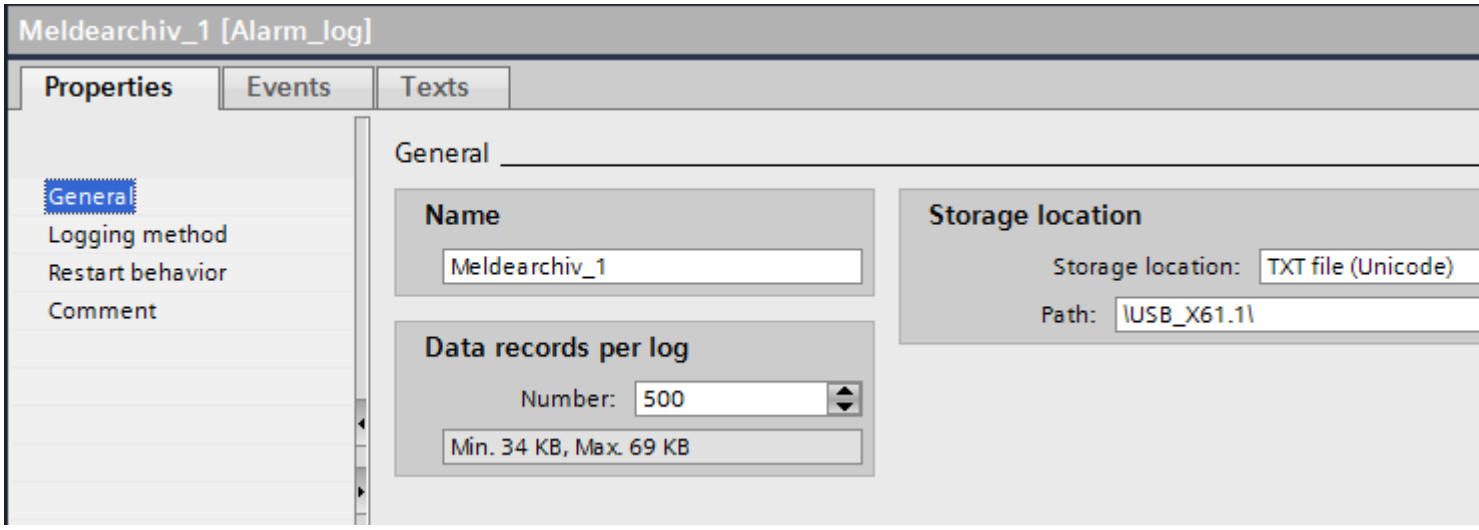

3. Under "Number of data records per log", define the number of alarms to be saved to a log file.
   The approximate space required on the storage medium is displayed. Memory space requirements will increase accordingly if you log alarm texts with tag values.
4. In the "Storage location" field you select where the log entries are saved.
5. Depending on the selected "Storage location", you either select the "Path" or the "Name of the data source".
6. To determine if audit trail data were changed at a later time, activate "Properties > Properties > General > Checksum".

   Note

   Device dependency

   The "Checksum" option is only available for display and HMI devices which support "Configuration conforms to GMP".

7. If necessary, enter a descriptive text in the "Comment" category to document your configuration.

Result

The alarm log is created. You can assign one or several alarm classes to this alarm log.

See also

Logging alarms (Page 889)
Working with Logs (Page 1027)

3.3.1.3 Logging alarms (RT Advanced, Panels, Comfort Panels)

Overview

To log alarms, follow these steps:

- Create an alarm log.
- Assign the created alarm log to an alarm class.
- Assign the alarm class with the created alarm log to an alarm.
- In the Runtime settings specify the language in which you want to write the logs.
- You evaluate logged alarms.
  You can analyze the logged alarms directly in your WinCC project, such as in an alarm view, or in another user program, such as Microsoft Excel.

Note

Tag fields in the alarm text

The sequence of tag fields in the alarm text is language dependent. The sequence of the Runtime language is used for logging alarms to a "*.csv" log file.

Changing the tag of an output field in one language causes the modified output field to appear at the end of the alarm text in all other languages. This means the sequence of the output fields in the log file can change.

Note

Text lists in alarm text

If you have inserted a text list into the alarm text and the alarm has been logged, the text list entries are displayed in the alarm log.

Requirements

- You have created an alarm log.
- The "HMI alarms" editor is open.

Assigning an alarm log to an alarm class

To assign the alarm log to an alarm class, proceed as follows:

1. Open the "Alarm classes" tab in the "HMI alarms" editor.
2. Select the required alarm class.
3. Under "Properties > Properties > General > Log" in the Inspector window, select the alarm log.

Assign alarm to an alarm class

To assign an alarm to the alarm class, proceed as follows:

1. In the "HMI alarms" editor, open the "Analog alarms" tab, or the "Discrete alarms" tab.
2. Select the required alarm.
3. Under "Properties > Properties > General > Alarm class" in the Inspection window, select the alarm class for which the alarm log was configured.

Result

The alarm is saved to the configured alarm log.

See also

Create an alarm log (Page 887)
Working with Logs (Page 1027)

3.3.1.4 Configuring an alarm view for logged alarms (RT Advanced, Panels, Comfort Panels)

Introduction

In Runtime, logged alarms are displayed in an alarm view or alarm window.

Requirements

- An alarm view or alarm window is configured in the "Screens" editor
- An alarm log was created in the "Log" editor.
- The alarms are assigned the "loggable" attribute in the "HMI alarms" editor.

Configuring an alarm view for logged alarms

To configure an alarm view for logged alarms, proceed as follows:

1. Open the screen with the alarm view and select the alarm view.
2. In the Inspector window, select "Properties > Properties > General > Alarm log".
3. Click the "...." button and select the alarm log.
4. Continue to configure the alarm view as for the display of current alarms.

Result

In Runtime, the logged alarms are output in the alarm view.

3.3.1.5 Configuring alarm buffer overflow (RT Advanced, Panels, Comfort Panels)

Introduction

The size of the alarm buffer depends on the type of HMI device. In WinCC, specify the percentage of the alarm buffer to be deleted on alarm buffer overflow.

Requirements

- You have created a project.

Procedure

To configure the response to an alarm buffer overflow, follow these steps:

1. Double-click "Runtime settings" in the project navigation under your HMI device.
2. Under "Alarms > General > Buffer clearance in percent upon buffer overflow", enter a value between 1 and 100.
   This value specifies the percentage of the alarm buffer that is deleted on alarm buffer overflow.

Result

If data in the alarm buffer exceeds the resources of alarm buffer memory, the configured percentage of data will be deleted from the oldest alarms in the alarm buffer.

Note

You cannot check the alarm buffer overflow separately by alarm method. You can use the "ClearAlarmBuffer" system function to delete specific alarms of specific alarm classes from the alarm buffer.

See also

Working with Logs (Page 1027)

3.3.1.6 Direct access to the ODBC log database (RT Advanced, Panels, Comfort Panels)

Overview

The storage location of a log can be a database or a file.

The database is addressed by means of its "Data source name" (DSN). Select the database to be used in WinCC as follows:

Select the database from the Windows Start menu under "Settings > Control panel > ODBC Data Sources".

In your configuration, specify the "Data source name" (DSN) instead of a directory name as storage path for log data. With the DSN, you are referencing the database and the storage location.

The entire functional scope of the database is available for additional processing and evaluation of log data.

Application

The data source sets up the connection to the database. Create the data source on the same PC on which the Runtime software is stored. Then enter the DSN configured on this PC when you create a log in WinCC.

The ODBC interface allows you to use other programs, such as MS Access or MS SQL Server, to access the database directly.

In addition, you can use the "StartProgram" system function to configure program calls, e.g. MS Access, on the HMI device. Runtime is not interrupted while you configure such calls.

3.3.1.7 Setting up the ODBC data source (RT Advanced, Panels, Comfort Panels)

Introduction

To store process values or alarms in the log database in Runtime, set up the ODBC data source.

Requirement

- A log database has been created.
  You create the log database using the SQL Enterprise Manager. You can find more detailed information on how to do this from Microsoft.

Procedure

To set up an ODBC data source, follow these steps:

1. In Control Panel, open "Administrative Tools" and select "Data Sources (ODBC)".
   The "ODBC Data Source Administrator" dialog is opened.
2. Click "User DSN > Add".
3. Select the "SQL-Server and click on "Finish".
4. In the dialog that follows, enter a name for the User DSN and the SQL-Server and click on "Next".
5. In the dialog that follows, define the logon procedure for the SQL database and click on "Next".
6. Activate "Change the default database to".
7. Select the database you have created and click on "Next".
8. In the dialog that follows, click "Finish".

3.3.1.8 Configuring a checksum for a log (RT Advanced, Panels, Comfort Panels)

Introduction

In a regulated project, you have the option of assigning a checksum to the log data of an data log or alarm log. This checksum can be used during plant operation to determine if the data of this log has subsequently changed.

Note

Device dependency

The "Checksum" option is only available for display and HMI devices which support "Configuration conforms to GMP".

Requirements

- GMP compliant configuration is enabled.
- A data log or alarm log has been created.

Procedure

Proceed as follows to configure a data log or alarm log for the use of a checksum:

1. Open the data log or alarm log in the corresponding log editor.
2. In the "Storage location" box, select "File - CSV (ASCII)" or "File - TXT (Unicode)".

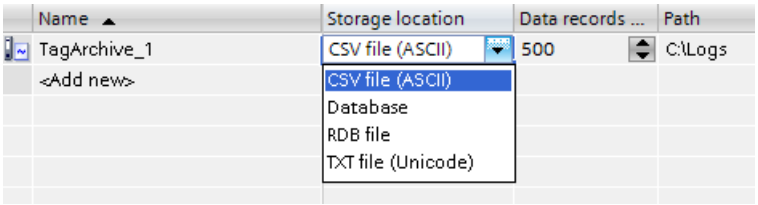

3. Under "Properties > Properties > Logging method" in the Inspector window, select the option "Display system event at" or "Trigger event".

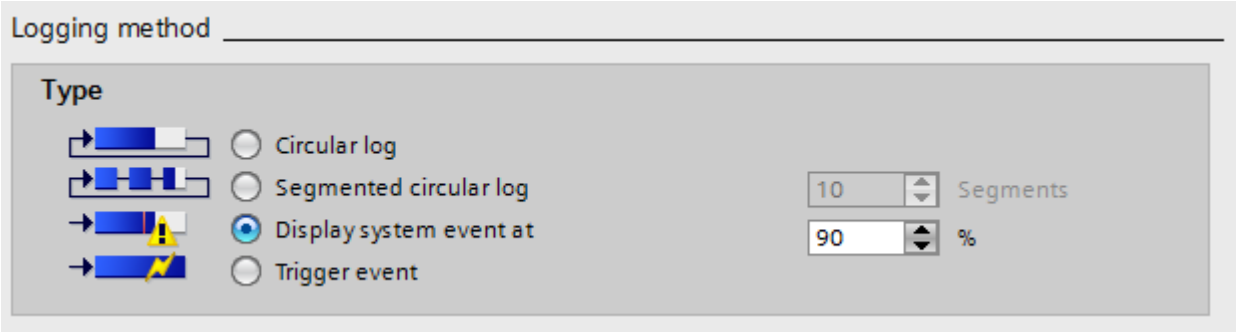

4. In the editor table, activate the option "Enable checksum".

5. In the editor table, activate the option "Enable logging at runtime start".
   Columns that are not displayed are activated with the shortcut menu of the column title.

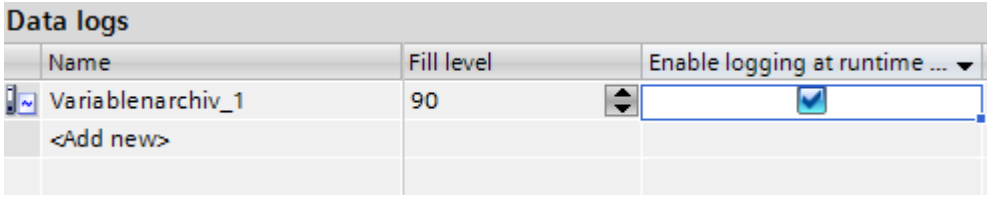

6. Save the project.

Result

The log data of the log is assigned a checksum in runtime.

3.3.1.9 Evaluating the checksum of log data (RT Advanced, Panels, Comfort Panels)

Introduction

If you have configured a data log or alarm log with generation of a checksum, you can check if the log data has subsequently changed.

Note

Device dependency

The "Checksum" option is only available for display and HMI devices which support "Configuration conforms to GMP".

The DOS program "HmiCheckLogIntegrity" is available for checking the integrity of the log data.

The "HmiCheckLogIntegrity" program supports verification of the following files:

- Log files of alarm logs, data logs, and Audit in CSV format
- Log files of alarm logs, data logs, and Audit in TXT format
- Recipe data records in CSV format
- Recipe data records in TXT format

You can find the "HmiCheckLogIntegrity.exe" program in the installation directory of WinCC under the folder "WinCC Runtime Advanced", for example <C:\Program Files\Siemens \Automation WinCC Runtime Advanced>.

Note

Audit Trail and Log with Checksum

Before you update WinCC with a Service Pack or a new version, exit and save the Audit Trail or the logs using checksum. After WinCC is updated, the audit trail or logs will be continued with new files using checksum.

Make sure that the logs are started at a defined state with the new version.

Procedure

1. Copy the file to be checked from the HMI device to your configuration computer.
2. Open a command line prompt with "Start > Programs > Accessories > Command Prompt".
3. Enter the path to "HmiCheckLogIntegrity.exe" followed by a space in the command line prompt. After the space, enter the storage location of the file to be checked within quotation marks.
4. Press <Enter>.
5. The check is performed.
   When the checked data are consistent, the "Consistency check succeeded" message appears.

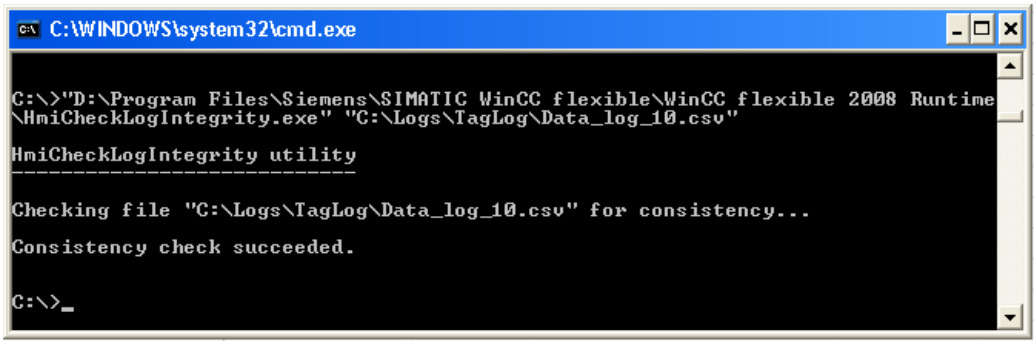

   When the checked data are inconsistent, the "Consistency check failed" message appears. Information about the first inconsistent line in the file is also displayed.

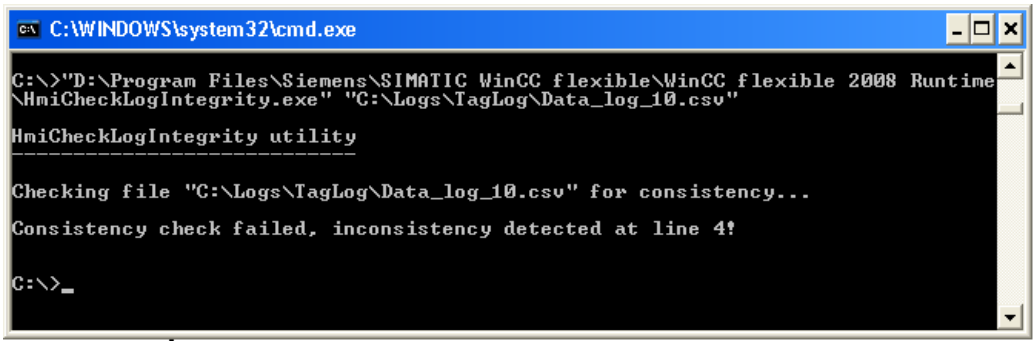

Note

If there are spaces in the path to the "HmiCheckLogIntegrity.exe" program, you need to specify the path in quotation marks.

You can also check the integrity of the log data with the AuditViewer.

3.3.1.10 Checksums for logs in regulated projects (RT Advanced, Panels, Comfort Panels)

Checksums in regulated projects

With WinCC you configure regulated projects as required by FDA guidelines, if necessary. Use the "Audit" option to create regulated projects.

In regulated projects you supply a checksum for the logged data of an alarm log. The operator will use this checksum during operation of the plant to see if the data of an alarm log have been changed.

For the checksum to be clear, it starts with the first line of an alarm log and integrates the previous lines when continuously generating the checksum. This means a checksum can only be generated for the following logs:

- Log that sends a system alarm when it is full
- Log with execution of system functions when log is full.

Generation of a checksum is also available for alarm logs that are saved as files of the "*.csv" or "*.txt" (Unicode) format.

If you want to continue an existing log without checksum as one with checksum, a backup copy of the existing log is created in the "*.keep" file format. A new log with checksums is created.

To maintain the validity of checksums, the following limitations apply when using the system function "CopyLog":

- You cannot copy a log without checksums into a log with checksums.
- You cannot copy a log with checksums into a log without checksums.

3.3.1.11 Log response to language switching in runtime (RT Advanced, Panels, Comfort Panels)

Introduction

In the Runtime settings of your HMI device, select the language to be used for writing to logs in runtime.

Requirement

- The languages used in your project are activated in the "Project languages" editor, for example, "German (Germany)" and "English (USA)" .

Procedure

To determine the startup language, follow these steps:

1. Select "Runtime settings > Language and fonts" in the project navigation.
2. Activate the runtime language, for example, "German (Germany)" and "English (USA)".
3. Set the "Language switch order". Use 0 to determine the startup language, for example:
   – German 0.
   – English 1
     With "0", German is specified as the "Startup language".
4. Select "Runtime settings > General" in the project navigation.
5. Select the "Logs > Logging language > Startup language".

Result

The project starts after it is transferred. German is specified as the "Startup language" with "Language switch order". The logs are now written in German. During runtime, the operator switches the runtime language to English. The logs will still to be written in German.

The operator closes runtime. Due to the previously performed language switch, the next time the system starts the "startup language" is English. Since English is the startup language, the logs are now written in English.

The logging language remains English even when the language is switched again in runtime until runtime is closed once again.

If you use another option instead of "Startup language" to select the language, the logs are always written in the same language. This is true regardless of the language selected by the operator in runtime.

See also

Sending a complete alarm from the controller to the HMI device and automatic update (Page 948)

3.3.1.12 Managing logging behavior when Runtime starts (RT Advanced, Panels, Comfort Panels)

Introduction

When you configure a log, you define the restart characteristics of the log when Runtime is started. You define whether the logging should start when Runtime starts in the log properties. You can also define whether an existing log will be continued or overwritten.

You define the restart characteristics separately for each log.

Requirement

- You have created a log.
- The "Historical Data" editor is open.
- The Inspector window with the log properties is open.

Procedure

To configure the restart characteristics of a data log, proceed as follows:

1. Select the log for which you want to define the restart characteristics in the "Historical Data" editor.
2. In the Inspector window select "Properties > Properties > Restart behavior".
3. If you want logging to start when Runtime starts, enable the "Enable logging at runtime start" option in the "Logging" area.
   You can also start logging in Runtime using the "StartLogging" system function, for example.
4. Select the restart behavior of the log in the "Log handling at restart" area.
   – The logged values are deleted and logging is started again with the option "Reset log".
   – The option "Append data to existing log" is used to append the values to be logged to the existing log.

Alternatively you can configure the restart characteristics of a log directly in the "Historical Data" editor. To view hidden columns, activate the column titles using the shortcut menu.

Result

Logging will start in runtime according to your settings.

3.3.1.13 Controlling the Logging in relation to the Fill Level (RT Advanced, Panels, Comfort Panels)

Introduction

The size of a log is determined by the number of entries. You use the logging method to determine how the log responds when it is full.

Logging methods

The following logging methods are available:

- Circular log
  When the configured log size has been reached, the oldest entries are deleted. When the configured log size has been reached, approximately 20% of the oldest entries are deleted. It is therefore not possible to display all the configured entries. During configuration, select an appropriate size for the circulation log. Alternatively, configure a segmented circular log.
- Segmented circular log
  In a segmented circular log, multiple log segments of the same size are filled in succession. When all logs are completely full, the oldest log is overwritten.
- Log that sends a system alarm when it is full
  When a defined level is reached, such as 90 %, a system alarm is triggered. When the log is 100% full, new tag values are not logged.
- Log with level-dependent triggering of an event.
  When the log is completely full, the "Overflow" event is triggered. Configure a function list for the event that will be carried out when the "Overflow" event occurs. When the configured size of the log is reached, new tag values are not logged.
  The following system functions are available for further processing of full logs:

Requirement

- You have created a log.
- The "Historical Data" editor is open.
- The Inspector window with the log properties is open.

Procedure

1. Select the log for which you want to define the logging method in the "Historical Data" editor.
2. Select "Properties > Properties > Logging method" in the Inspector window and select the required logging method.
3. If you have selected the "Segmented circular log" type, enter the number of log segments. The system creates an additional log segment for the main log. This results in the total number of log files created from the number of configured segments as well as the automatically created log.
   If you selected a log with the "Display system alarm on" setting, specify the level as a percentage at which a system alarm is to be triggered.
   If you selected the "Trigger event" setting, configure the function list in the "Events" group.

Alternatively you can configure the logging method directly in the "Historical Data" editor table. To view hidden columns, activate the column titles using the shortcut menu.

The "Overflow" event is not available in the editor table. You must therefore configure the function list in the Inspector window.

Result

The selected log responds according to the settings in Runtime.

3.3.2 Configuring alarm logging (RT Professional)

3.3.2.1 Basics on alarm logging (RT Professional)

Introduction

An alarm log is used to record project alarms.Alarm logs are created by the system. For example, after an error has occurred, or a limit was exceeded in Runtime, the corresponding alarm you configured in the "HMI alarms" editor is output in Runtime.Each alarm event is logged, e.g., the transition of the alarm from "incoming" to "acknowledged" status.

You can store the alarm events in a log database and/or log them to hardcopy in an alarm report. You can output the alarms stored in the database in Runtime, for example, in an alarm view.

The logged alarms are stored in a circular log that consists of multiple single segments. The size of all the segments and of an individual segment is defined under "Alarm log" in the log settings.

You should configure backups to save the data at regular intervals.

Configuration steps

To log an alarm, follow these configuration steps:

1. Configuring "Log settings"
   You define the settings for the alarm logging in the log settings.
2. Enable alarm logging
   You specify that alarms of particular alarm classes will be logged.
3. Assigning an alarm to a loggable alarm class

Content of the alarm log

The alarm logs are used to store all alarm data, including configuration data. You can read all alarm properties from the log files, e.g. the alarm class, the timestamp, and the alarm texts.A new log segment with the new configuration data is generated whenever you edit configuration data of an alarm. This function prevents any change from having an influence on alarms logged previously.

The possible number of logged alarms depends on the server used.

Note

The time stamp of a logged alarm is always specified in standard UTC format (Universal Time Coordinated).

As alarm configuration is language-specific, the logs contain a configuration data table for each language configured.

Storage media and location

Log data are stored in a database. You can further process the saved data in other programs for analysis purposes, for example.

Backup for log segments

Take backups of your log segments to ensure complete documentation of your process. You define the settings under "Log settings".

Displaying logged data

You can view the logged data on the HMI device.Configure a corresponding alarm view that displays the log data, or use buttons to show the "Historical alarm list" (short-term), or the "Historical alarm list" (long-term) in Runtime.

See also

Working with Logs (Page 1033)

3.3.2.2 Logging alarms (RT Professional)

Introduction

Specify the size and time range for the alarm log segments in the log settings.By specifying their size, you also define the number of segments.

The segments are filled successively with data in Runtime.The next segment is used on overflow of a segment.You can also configure a time-dependent segment change.If you specify a time for the segment change, the next log segment is filled after this time was reached.

Note

Ensure that the log size does not exceed memory resources. The log manager does not validate the selected settings.A large number of linked log segments might lead to longer waiting periods in the system during the startup and shutdown of Runtime.

Overview

To log alarms, follow these configuration steps:

- Configure the alarm log in the log settings.
  Define the size and segmentation.
- Configure a loggable alarm class.
- Assign the alarm to a loggable alarm class.

Requirements

- An alarm class has been created.

Configuring an alarm log

To configure the alarm log, proceed as follows:

1. Double-click "Runtime settings" in the project navigation.
2. Double-click "Logging" in the area navigator.
   The alarm settings open.

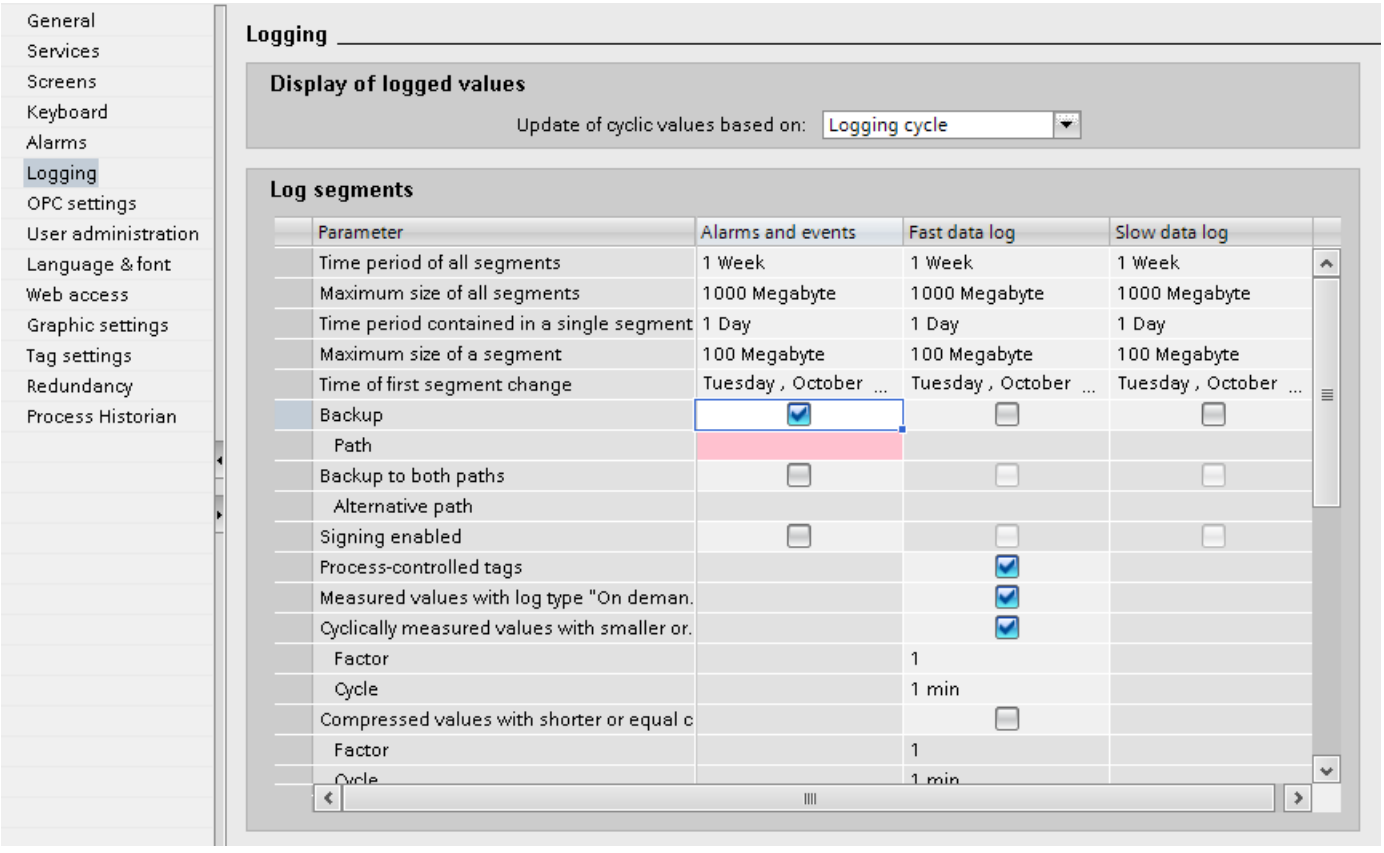

3. Click "Properties > General" in the Inspector window and specify the following:
   - Enter the values for "Time period of all segments" and "Maximum size of all segments". Define the size of the log database by entering the accumulated time period for all segments and the maximum segment size.A new segment is started if either one of these criteria is exceeded. The oldest segment is overwritten after all segments were filled with data.
   - Enter your values at "Time period covered by a single segment" and "Maximum size of a segment".
     You include the size and number of single segments when entering the time period for single segments and their maximum size. A new single segment is started if either one of those conditions is exceeded. In addition, the oldest single segment is deleted after the criterion for "Time period of all segments" is exceeded.
   - Select the date and time for the start of the first segment change at "Time of first segment change".
     The segment is changed even if the configured size is exceeded after Runtime starts. In this case, the oldest individual segment is also deleted.

Create alarm class with logging

To create an alarm class with logging, proceed as follows:

1. Open the "HMI alarms" editor.
2. Click the "Alarm Classes" tab.
3. Select the alarm class whose alarms are logged.
4. In the Inspector window, select "Properties > Properties > General > Settings > Alarm log".
5. Assign the alarm to be logged to this alarm class.

Example for the timing of the first segment change

A segment changes for the first time at 23:59 on January 17. The next time, the segment changes at the configured time in the cycle defined in "Time period contained in a single segment". If the cycle setting is "1 day", then the change will take place at 23:59 every day. If the cycle is set to "1 month", the segment will change at 23.59 on the 17th day of the next month.

See also

Logging Settings (Page 660)

Defining the Log Size and Segmentation (Page 661)

Working with Logs (Page 1033)

3.3.2.3 Configuring the Alarm Log Backup (RT Professional)

Principle

Make regular backups of your log data to ensure complete documentation of your process.

Note

The backup normally starts 15 minutes after the first time-related segment change. To start the backup and the segment synchronously with Runtime, define the time for the segment change before you start Runtime.

You can enter comments for alarms displayed in the alarm view in Runtime. If the log segment in which the alarm is stored has already been swapped out, the comment will not be accepted in the swapped-out log. The comment will only be in the local log segment.

If the archive segment has not yet been moved out, then the changed comment is accepted permanently.

Requirement

- The settings for logging are open.

Procedure

To create a backup of the log segments, proceed as follows:

1. Double-click "Logging" in the area navigator.
2. Enable "Alarm log > Backup".

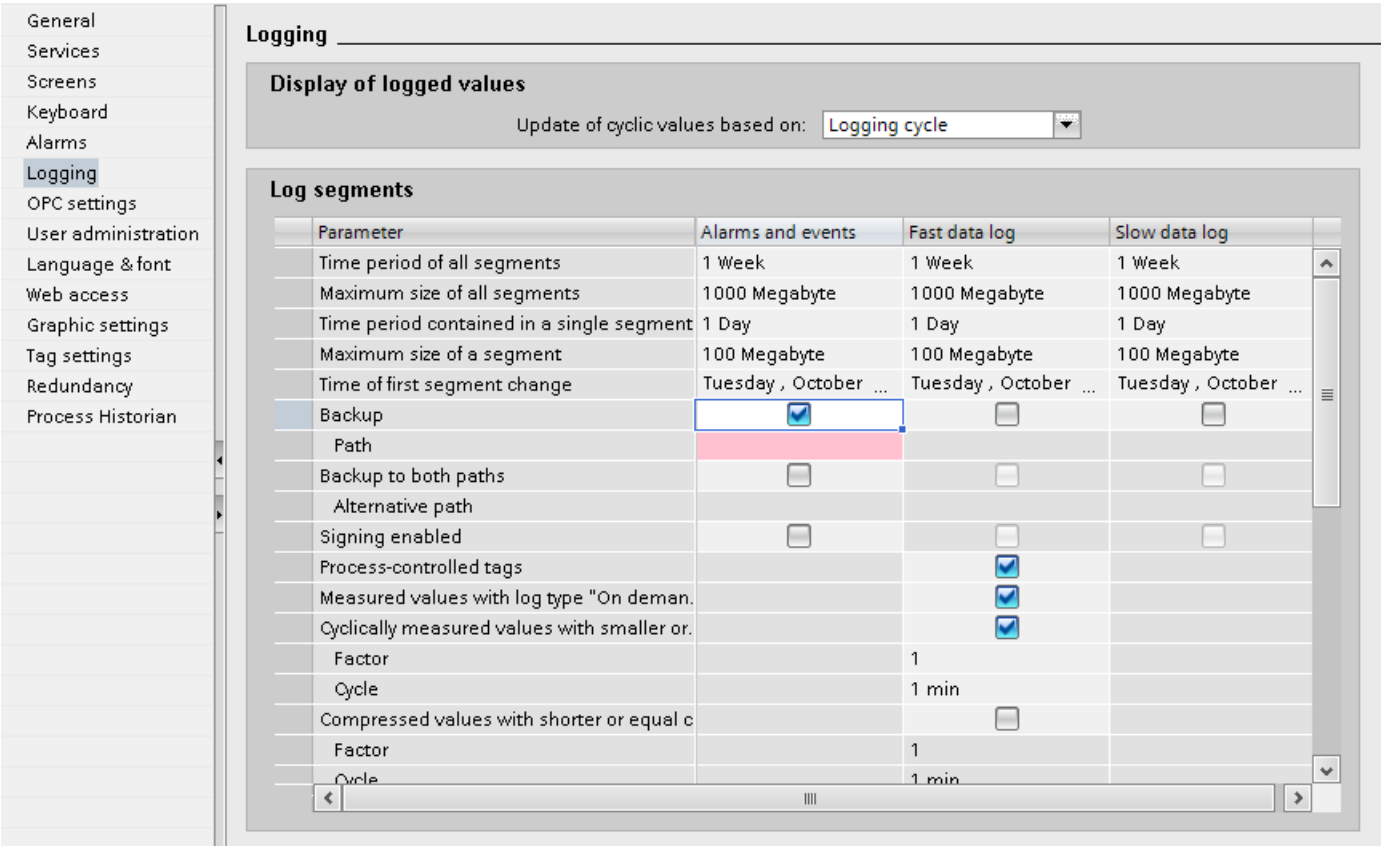

3. In the Inspector window, click "Properties > General > Settings Backup > Path".

4. Click ... and look for an existing folder in which to store the backup file. Use "Make New Folder" to create a new folder, if necessary. You can also use a network path as the storage location.

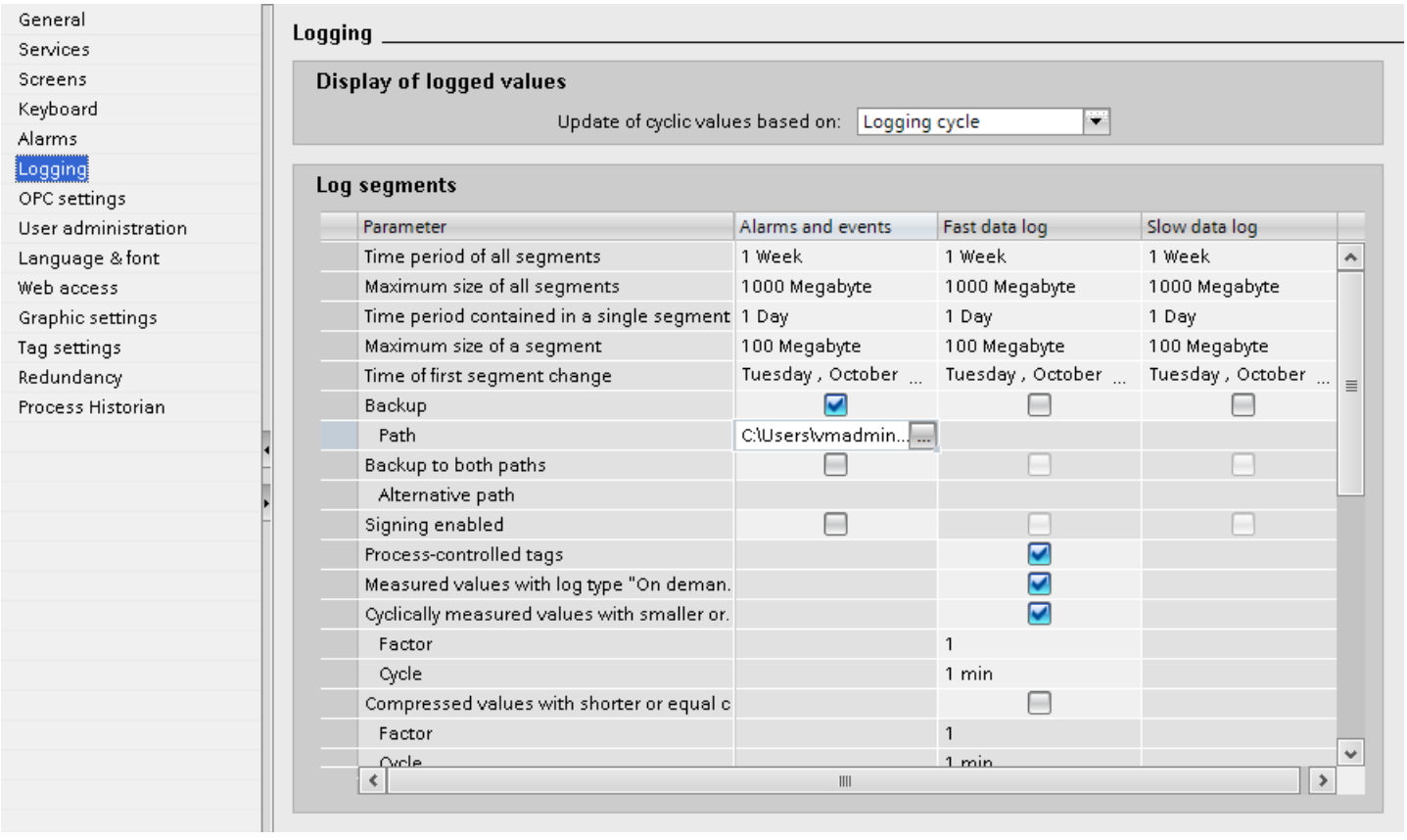

5. Activate "Backup" to generate a backup copy of logged data.
   Also activate "Backup to both paths" if a backup copy of logged data is to be saved to the "Path" and to the "Alternative path" folders.
   You can also use a network path as the storage location.
   If the "Backup to both paths" option is not enabled, the second backup file is used in the following situations:
   - If the storage space on a backup medium is full.
   - The original storage location is unavailable due to network failure, for example.

   An alarm is output if a storage location was not found, provided you configured corresponding system alarms.
6. Activate the "Sign" check box if the backup file must be saved with signature.
   When the connection to WinCC is reestablished, the signature is used to detect whether any changes have been made to a backup file since it was swapped out.

Note

The "Sign" option puts a load on the RAM. If you use log signing, the maximum size of a single segment must not exceed 200 MB.

Result

The log segment backup is stored in the specified location.

Structure of the Backup File

A backup file for an alarm log has the following structure:

- Extension
  A log segment backup consists of two files in "*.LDF" and "*.MDF" format. To transfer a backup to another PC, simply copy both files.
- File name
  The file name is in the following form:
  "<Computer_name>_<Project_name>_<Type>_<Time_from>_<Time_to>".

Component	Explanation	Example
Computer name		FLEX-LOG2
Project ID	The project ID is assigned by the system. Underscores are replaced with "#".	HMI#4ZQV
Type	The type identification of the alarm log is "ALG" (Archive Log).	
Time_from	The time period is specified in yyyymmddhhmm format.	200812021118 for December 2, 2008, 11:18.
Time_to		
Example of the complete file name of a backup file: FLEX-LOG2_HMI#4ZQV_ALG_200811291414_200812030502.ldf		

Note

Assignment to project

The project name cannot be seen in the file name of the generated project ID.

To be able to allocate the backup files to the project, assign a path name to the storage location that contains the project name.

Signing the Backup File

If signing and backup are enabled, each log backup file is signed when swapped out. When the file is reconnected to WinCC, it is thus determined whether the file was changed after being swapped out.

If you use log signing, the maximum size of a single segment must not exceed 200 MB.

The "Enable signing" check box must be enabled for data verification. When signing is enabled, it takes longer to link the log backup file to WinCC. Up to 500 values are logged per second.

Note

If you disable signing off to establish a fast connection to the backup files, for example, you should avoid segment changes while it is disabled. After having closed the connection, re-enable signing so that the data to be logged is assigned a signature.

See also

Working with Logs (Page 1033)

3.3.2.4 Connecting and Disconnecting the Alarm Log Backup (RT Professional)

Introduction

To access the data in a swapped-out alarm log in Runtime, connect the log backup to the project. You can configure an automatic connection or connect the alarm log to the project via a script. The archived alarms are displayed in Runtime in the alarm view.

If you no longer want to access the backup of a log segment in Runtime, disconnect the log backup from the project.

Requirement

- The relevant backup files in "*. ldf" and "*.mdf" format are stored locally.
- The project is in Runtime.

Display Time Range

The alarms are only displayed in Runtime if the time window in the alarm view has been configured accordingly.

Example

You have configured a time window to display only the alarms of the past 24 hours. If you connect to a log backup containing alarms that are older than 24 hours, these alarms are not included in the alarm view.

Connecting to the alarm log automatically

To automatically connect to the alarm log backup, proceed as follows:

1. Insert the log backup files in the "RuntimeProjectPath\ProjectName\CommonArchiving" folder.
2. In Runtime, the alarm log is automatically connected to the project.

If signing is enabled, signed log backup files that are changed will not be connected automatically. A WinCC system alarm is generated and an entry is added to the Windows event log in the "Application" section.

Connecting to the alarm log using a script

Using the "AlarmLogs" VBS object, you can link the log backup files to the project using a script. The log segments are then copied with the "Restore" VBS method to the Common Archiving folder of the Runtime project. For additional information, see the description of the "AlarmLogs" VBS object and the "Remove" VBS method.

Automatically disconnecting the alarm log

To automatically disconnect the alarm log backup from the project, proceed as follows:

1. Go to the folder "RuntimeProjectPath\ProjectName\CommonArchiving".
2. Remove the log backup files from the folder.

Disconnecting the alarm log using a script

Using the "AlarmLogs" VBS object, you can disconnect the log backup files from the project using a script. The log segments are then removed with the "Remove" VBS method from the Common Archiving folder of the Runtime project. For additional information, see the description of the "AlarmLogs" VBS object and the "Remove" VBS method.

3.3.2.5 Accessing the Log Database Directly with C-API/ODK (RT Professional)

Introduction

Various providers offer interfaces that you can use to access databases. These interfaces also allow you to directly access the WinCC log databases. Using direct access you can, for example, read process values to process them further in spreadsheet programs.

Accessing the log database with C-API/ODK

You can use the "WinCC Open Development Kit" option to access WinCC data and functions via open application programmer interfaces.

Additional information

You will find further information on this topic in the documentation for the WinCC Open Development Kit.

3.3.2.6 Accessing the Log Database Directly with ADO/OLE DB (RT Professional)

Introduction

Various providers offer interfaces that you can use to access databases. These interfaces also allow you to directly access the WinCC log databases. Using direct access you can, for example, read process values to process them further in spreadsheet programs.

Accessing the log databases with ADO/OLE DB

Some process values are stored in the log databases in compressed format. Use the WinCC OLE DB Provider to access these compressed process values. For non-compressed data and data from the alarm log, you can also use the Microsoft ADO/OLE DB interfaces, for example. You can use Visual Basic, Visual C++, etc., as the programming language.

Note

If you access the log database directly using ADO/OLE DB, remember that the log structure may be different in a new version of WinCC.

Additional information

You will find more detailed information on this topic on the Internet at:

msdn.microsoft.com (http://msdn.microsoft.com/en-us/library/default.aspx)

3.3.2.7 Configuring the Reloading of Alarms after a Power Failure (RT Professional)

Introduction

Configure the reloading of alarms so that the most recent alarms are displayed in Runtime once more after a power failure. In the event of a power failure, the number of alarms that you configured is read from the alarm log. This allows the most recent process image to be reconstructed.

Procedure

To define the number of alarms that are reloaded after a power failure, follow these steps:

1. Double-click "Runtime settings" in the project navigation under your HMI device.
2. Double-click "Alarms" in the area navigator.
3. Select "Properties > General > Number of alarm list entries after power failure" in the Inspector window to enter the number of alarms to be reloaded after power failure.

Result

After power failure, the number of alarms you configured is reloaded to the alarm view and displayed in Runtime.

3.4 Using Alarms in Runtime (RT Advanced, Basic Panels, Panels, Comfort Panels)

3.4.1 Alarms in Runtime (Basic Panels)

Alarms

Alarms indicate events and states on the HMI device which have occurred in the system, in the process or on the HMI device itself. A status is reported when it is received.

An alarm could trigger one of the following alarm events:

- Incoming
- Outgoing
- Acknowledge
- Loop-in-alarm

The configuration engineer defines which alarms must be acknowledged by the user.

An alarm may contain the following information:

- Date
- Time
- Alarm text
- Location of fault
- Status
- Alarm class
- Alarm number
- Alarm group

Alarm classes

Alarms are assigned to various alarm classes.

- "Warnings"
  Alarms of this class usually indicate states of a plant such as "Motor switched on". Alarms in this class do not require acknowledgement.
- "Errors"
  Alarms in this class must always be acknowledged. Error alarms normally indicate critical errors within the plant such as "Motor temperature too high".

- "System"
  System alarms indicate states or events which occur on the HMI device.
  System alarms provide information on occurrences such as operator errors or communication faults.
- Custom alarm classes
  The properties of this alarm class must be defined in the configuration.

Alarm buffer

Alarm events are saved to an internal buffer. The size of this alarm buffer depends on the HMI device type.

The "Persistent alarm buffer" setting is used to switch the alarm buffer on and off in Runtime. For storage space reasons it can be necessary to switch off the alarm buffer.

The setting "Persistent alarm buffer" is activated under "Runtime Settings > Alarms". In this setting the alarm view displays both current alarms and also alarms from the alarm buffer

If you switch off the alarm buffer you need to restart Runtime in order that the alarm view no longer displays the alarms from the alarm buffer.

Alarm view

The alarm view shows selected alarms or alarm events from the alarm buffer. Whether alarm events have to be acknowledged or not is specified in your configuration.

Alarm window

The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active.

You can configure the order in which the alarms are displayed. You can choose to display the alarms in ascending or descending order of their occurrence. The alarm window can also be set to indicate the exact location of the fault, including the date and time of the alarm event. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string will be shown.

Alarm indicator

The alarm indicator is a graphic symbol that is displayed on the screen when an alarm of the specified alarm class is activated.

The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated. The number indicates the number of queued alarms.

3.4.2 Alarms in Runtime (RT Advanced, Panels, Comfort Panels)

Alarms

Alarms indicate events and states on the HMI device which have occurred in the system, in the process or on the HMI device itself. A status is reported when it is received.

An alarm could trigger one of the following alarm events:

- Incoming
- Outgoing
- Acknowledge

The configuration engineer defines which alarms must be acknowledged by the user.

An alarm may contain the following information:

- Date
- Time
- Alarm text
- Location of fault
- Status
- Alarm class
- Alarm number
- Alarm group

Alarm classes

Alarms are assigned to various alarm classes.

- "Warnings"
  Alarms of this class usually indicate states of a plant such as "Motor switched on". Alarms in this alarm class do not require acknowledgment.
- "Errors"
  Alarms in this class must always be acknowledged. Error alarms normally indicate critical errors within the plant such as "Motor temperature too high".
- "System"
  System alarms indicate states or events which occur on the HMI device.
  System alarms provide information on occurrences such as operator errors or communication faults.
- "Diagnosis Events"
  SIMATIC diagnostic alarms show states and events in the SIMATIC S7 controller.
- "Safety Warnings"
  The "Safety Warnings" alarm class shows alarms for fail-safe operation. Users do not acknowledge alarms from this alarm class.

- STEP 7 alarm classes
  The alarm classes configured in STEP 7 are also available to the HMI device.
- Custom alarm classes
  The properties of this alarm class are defined in the configuration.

Alarm buffer

Alarm events are saved to an internal buffer. The size of this alarm buffer depends on the HMI device type.

Note

Valid for Panels and RT Advanced

The "Persistent alarm buffer" setting is used to switch the alarm buffer on and off in Runtime. For storage space reasons it can be necessary to switch off the alarm buffer.

The setting "Persistent alarm buffer" is activated under "Runtime Settings > Alarms". In this setting the alarm view displays both current alarms and also alarms from the alarm buffer

If you switch off the alarm buffer you need to restart Runtime in order that the alarm view no longer displays the alarms from the alarm buffer.

Alarm report

When alarm logging is enabled in the project, alarms are output directly to the printer.

You can set the logging function separately for each alarm. Printing of such an alarm is initiated when the "incoming" and "outgoing" alarm events are generated.

The output of alarms of the "System" alarm class to a printer must be initiated by means of the corresponding alarm buffer. This outputs the content of the alarm buffer to the printer. To be able to initiate this print function, you need to configure an operating element in the project.

Note

Valid for Panels and RT Advanced

When alarm report is enabled in the project, alarms are only printed after Runtime has been closed.

Alarm log

Alarm events are stored in an alarm log, provided this log file is configured. The capacity of the log file is limited by the storage medium and system limits.

Alarm view

The alarm view shows selected alarms or events from the alarm buffer or alarm log. Whether alarm events have to be acknowledged or not is specified in your configuration. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string in their alarm text are shown.

Alarm window

The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active. Depending on the configuration, it is not closed until the alarm is acknowledged.

You can configure the order in which the alarms are displayed. At the first position, the current, or the oldest alarm will be displayed. The alarm window can also be set to indicate the exact location of the fault, including the date and time of the alarm event. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string in their alarm text are shown.

Alarm indicator

The alarm indicator is a graphic symbol that is displayed on the screen when an alarm of the specified alarm class is activated.

The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated. The number indicates the number of queued alarms.

3.4.3 Using the Alarm Window or Alarm View (RT Advanced, Basic Panels, Panels, Comfort Panels)

3.4.3.1 Alarm window, alarm view in Runtime (RT Advanced, Basic Panels, Panels, Comfort Panels)

Application

Alarms are indicated in the alarm view or in the alarm window on the HMI device. The layout and operation of the alarm window correspond to that of the alarm view.

The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active. Depending on the configuration, it is not closed until the alarm is acknowledged.

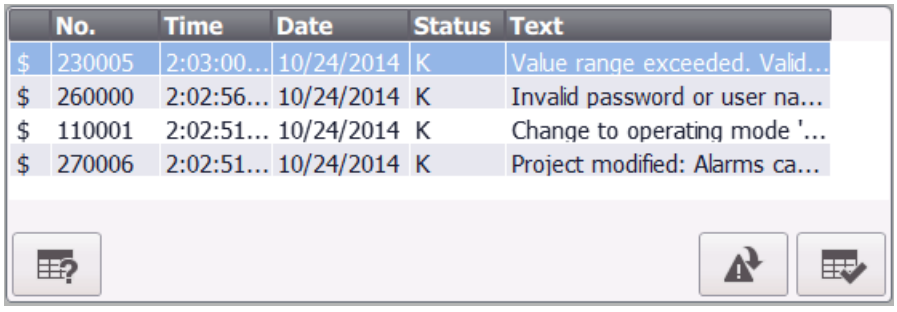

Layout

Depending on the configuration, in the alarm view different columns with information regarding an alarm or an alarm event are displayed. If a filter is configured, only alarms that contain a specific string in the alarm text will be displayed.

To differentiate between the different alarm classes, the first column in the alarm view contains an icon:

Symbol	Alarm class
!	"Errors"
Empty	"Warnings"
depends on the configuration	Custom alarm classes
$	"System"
S7	"Diagnosis Event"
!!	"Safety Warnings"

Operation

You use the alarm view as follows, depending on how it is configured:

- Change the order of the columns.
- Change the order in which the alarms are displayed.
- Acknowledging alarms
- Editing alarms

Operator controls

The buttons have the following functions:

Button	Function
	Displaying a tooltip for an alarm
	Loop-In-Alarm Executes the configured function (for example: "ActivateScreen") with each key-stroke. It is hereby possible to call a screen containing your information about the selected alarm. Also acknowledges an alarm requiring acknowledgment. [1]
	Acknowledge alarm

[1] Only if a function is configured for the selected alarm for the "loop-in-alarm" event.

Operation behavior

Linked alarm window for touch panels

When configuring the alarm window for keyboard units, enable the "Connected" (modal) property under "Properties > Mode". This ensures that the alarm window does not defocus during screen changes. Switching back and forth between the screen and different windows with <Ctrl+TAB> is not supported. If the linked alarm window has the focus, then the buttons in the screen behind it cannot be operated. The functions configured on a function key are carried out.

Changing the order of the displayed alarms

When you click on the column, first of all alarms requiring acknowledgment are sorted according to date and time. Then those alarms that do not require acknowledgment are sorted according to date and time.

3.4.3.2 Using the Alarm Window, Alarm View (RT Advanced, Panels, Comfort Panels)

Introduction

As an alternative to using the mouse, you can operate the alarm view and the alarm window using the <TAB> key on your HMI device. This allows you to select the button and the most recently selected alarm in the alarm view. Depending on the configuration, you can also operate the alarm view via the function keys.

Operation using the mouse

1. Select the alarm to be edited.
2. Click the button whose function you want to run.

Operation using the keyboard

1. Press the <Tab> key until the list of displayed alarms is selected in the alarm view.
2. Click on the alarm to be edited. Use the <Up> and <Down> keys accordingly.
3. Press the <Tab> key until the button whose function you wish to use is selected.
4. Press <Enter>.

Change the order of the columns

1. Select the column header, such as the "Date" column header.
2. While holding down the mouse button, drag the column header to the column header "Time".
   The "Date" column is in front of the "Time" column.

Change the sorting

You can sort the list according to date or time.

1. Click on the corresponding column header.
   The list is sorted in descending or ascending order according to this criterion.
2. Click the same column header again to reverse the sort order.

Acknowledge alarm

1. Click on the alarm to be edited.
2. Click the button.

Loop-In-Alarm trigger

1. Click on the alarm to be edited.
2. Click the button.
   The screen containing information about the alarm is displayed.

   Note

   If you trigger a Loop-In-Alarm during an unacknowledged alarm, it is acknowledged automatically.

Displaying configured tooltip

1. Click on the alarm concerned.
2. Click the button.
   The tooltip configured for the alarm is displayed.

3.4.4 Using the Simple Alarm Window, Alarm View (RT Advanced, Basic Panels, Panels, Comfort Panels)

3.4.4.1 Basic alarm view, alarm window in Runtime (RT Advanced, Basic Panels, Panels, Comfort Panels)

Application

The simple alarm view shows selected alarms or alarm events from the alarm buffer. The layout and operation of the simple alarm window correspond to that of the simple alarm view.

Note

The "Single alarm view" object is not assigned dynamic functions by means of scripting.

In the Engineering System, for example, dynamize the visibility of an object in the "Animations" tab of the Inspector window. In Runtime, the "Simple alarm view" does not support animations. If you configured an animation and, for example, run a consistency check on the project, an error alarm is displayed in the output window.

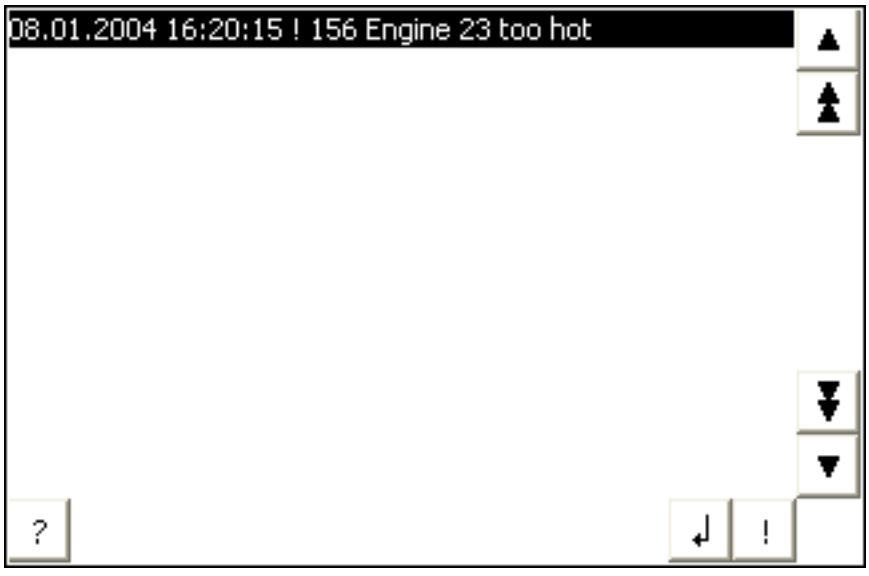

Layout

Depending on the configuration, in the alarm view different columns with information regarding an alarm or an alarm event are displayed.

To differentiate between the different alarm classes, the first column in the alarm view contains an icon:

Icon	Alarm class
!	"Errors"
empty	"Warnings"
depends on the configuration	Custom alarm classes
$	"System"

Operation

You use the alarm view as follows, depending on how it is configured:

- Acknowledging alarms
- Editing alarms

Control elements

The buttons have the following functions:

Button	Function
!	Acknowledge alarm
↲	Loop-In-Alarm Executes the configured function (e.g.: "ActivateScreen") whenever a key is pressed. This enables a screen that contains your information on the selected alarm to be called. Also acknowledges an alarm that requires acknowledgment. [1]
?	Displaying a tooltip for an alarm
▶	Displays the full text of the selected alarm in a separate window, namely the alarm text window In the alarm text window, you can view alarm texts that exceed the space available in the Alarm view. Close the alarm text window with the ✕ button.
▲	Scrolls one alarm up.
⇞	Scrolls one page up in the alarm view.
⇟	Scrolls one page down in the alarm view.
▼	Scrolls one alarm down.

[1] Only if a function is configured for the selected alarm for the event "Loop-In-Alarm".

Format of the control elements

The display of the buttons for using the simple alarm view depends on the configured size. You should therefore check on the HMI device whether all the required buttons are available.

3.4.4.2 Basic alarm view, using the alarm window (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

As an alternative to using the mouse, you can operate the simple alarm view using the <TAB> key on your HMI device. This allows you to select the button and the most recently selected alarm in the alarm view. Depending on the configuration, you can also operate the alarm view via the function keys.

Operation using the mouse

1. Select the alarm to be edited.
2. Click the button whose function you want to run.

Operation using the keyboard

1. Press the <Tab> key until the list of displayed alarms is selected in the alarm view.
2. Click on the alarm to be edited. Use the <Up> and <Down> keys accordingly.
3. Press the <Tab> key until the button whose function you wish to use is selected.
4. Press <Enter>.

Acknowledge alarm

1. Select the alarm to be acknowledged.
2. Click the button.

Loop-In-Alarm trigger

1. Select the alarm to be edited.
2. Click the button.
   The screen containing information about the alarm is displayed.

Calling the infotext

1. Select the alarm to be edited.
2. Click the button.
3. To close the window for displaying the operator note, press the button or use the key combination <Alt+F4>.

3.4.5 Using the Alarm Indicator (RT Advanced, Basic Panels, Panels, Comfort Panels)

3.4.5.1 Alarm indicator in Runtime (RT Advanced, Basic Panels, Panels, Comfort Panels)

Application

The alarm indicator is displayed if alarms of the specified alarm class are pending or require acknowledgment.

Layout

The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated. The number indicates the number of queued alarms.

Operation

Depending on the configuration, when operating the alarm indicator an alarm window is opened. The icons from the symbol library can only be operated with a mouse or touch screen.

Note

Device dependency

Operation with mouse is not available for all HMI devices.

3.4.5.2 Operating the alarm indicator using the mouse (RT Advanced, Panels, Comfort Panels)

Operation using the mouse

1. Click on the alarm indicator with the mouse pointer. Depending on the configuration, the Alarm window is open.
2. Click ✕ to close the alarm window. The Alarm window can be opened again by clicking on the alarm indicator.

3.4.6 Acknowledging alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You can acknowledge alarms in Runtime according to your project configuration settings. You can acknowledge alarms as follows:

- Using the display and control object buttons
- Using the "ACK" key on your HMI device
- Using individually-configured function keys or buttons

If an operator authorization is configured for an individual control, the alarms can only be acknowledged by authorized users.

To automatically acknowledge alarms in Runtime, use the system functions and scripts, plus the "Acknowledgment by the PLC" option.

Note

Device dependency

Scripts are not available for all HMI devices.

Acknowledgment variants

You acknowledge individual alarms or multiple alarms together in Runtime. They are distinguished as follows:

- Single acknowledgment
  Acknowledgment of an alarm using a button or a function key.
- Acknowledge alarm groups
  Acknowledgment of all the alarms of an alarm group using a button or a function key.

Requirement

- An alarm is displayed on the HMI device.

Procedure

To acknowledge an alarm, proceed as follows:

1. Select the alarm.
2. If you are using an alarm view or an alarm window, click the button.

3. If you are using a simple alarm view or a simple alarm window, click the ! button.
4. Press the "ACK" button on your HMI device to acknowledge an alarm in an alarm line.

Note

Device dependency

The extended alarm view or alarm window and the alarm line are not available for all HMI devices.

Result

The alarm status changes to "Acknowledged". If the condition for triggering an alarm no longer applies, the alarm status also changes to "Outgoing", and it is no longer displayed on the HMI device.

3.4.7 Filtering alarms (RT Advanced, Panels, Comfort Panels)

Introduction

The alarm view in Runtime does not contain a control element for entering filter criteria. To provide this functionality, configure an I/O field using the "Screens" editor.

Requirements

- A filter has been configured for the alarm view.
- An I/O field for entering the criterion is displayed.
- Alarms are displayed in Runtime.

Enable filter

1. Enter the filter string in the I/O field.
2. Confirm your input.
   The alarm view displays only those alarms that are contained in the specified character string. Thereby this distinguishes between upper and lower case letters.

Disable filter

1. Delete the string from the I/O field that is used to filter the alarm view.
2. Confirm your input.
   All alarms are displayed in the alarm view once more.

3.5 Using Alarms in Runtime (RT Professional)

3.5.1 Alarms in Runtime

Alarms

Alarms indicate events and states on the HMI device which have occurred in the system, in the process or on the HMI device itself. A status is reported when it is received.

An alarm could trigger one of the following alarm events:

- Incoming
- Outgoing
- Acknowledge

The configuration engineer defines which alarms must be acknowledged by the user.

An alarm may contain the following information:

- Date
- Time
- Alarm text
- Location of fault
- Status
- Alarm class
- Alarm number
- Alarm group

Alarm classes

Alarms are assigned to various alarm classes.

- "Warnings"
  Alarms of this class usually indicate states of a plant such as "Motor switched on". Alarms in this alarm class do not require acknowledgment.
- "Errors"
  Alarms in this class must always be acknowledged. Error alarms normally indicate critical errors within the plant such as "Motor temperature too high".
- "System"
  System alarms indicate states or events which occur on the HMI device.
  System alarms provide information on occurrences such as operator errors or communication faults.
- "Diagnosis Events"
  SIMATIC diagnostic alarms show states and events in the SIMATIC S7 controller.

- "Safety Warnings"
  The "Safety Warnings" alarm class shows alarms for fail-safe operation. Users do not acknowledge alarms from this alarm class.
- STEP 7 alarm classes
  The alarm classes configured in STEP 7 are also available to the HMI device.
- Custom alarm classes
  The properties of this alarm class are defined in the configuration.

Alarm buffer

Alarm events are saved to an internal buffer. The size of this alarm buffer depends on the HMI device type.

Note

Valid for Panels and RT Advanced

The "Persistent alarm buffer" setting is used to switch the alarm buffer on and off in Runtime. For storage space reasons it can be necessary to switch off the alarm buffer.

The setting "Persistent alarm buffer" is activated under "Runtime Settings > Alarms". In this setting the alarm view displays both current alarms and also alarms from the alarm buffer

If you switch off the alarm buffer you need to restart Runtime in order that the alarm view no longer displays the alarms from the alarm buffer.

Alarm report

When alarm logging is enabled in the project, alarms are output directly to the printer.

You can set the logging function separately for each alarm. Printing of such an alarm is initiated when the "incoming" and "outgoing" alarm events are generated.

The output of alarms of the "System" alarm class to a printer must be initiated by means of the corresponding alarm buffer. This outputs the content of the alarm buffer to the printer. To be able to initiate this print function, you need to configure an operating element in the project.

Note

Valid for Panels and RT Advanced

When alarm report is enabled in the project, alarms are only printed after Runtime has been closed.

Alarm log

Alarm events are stored in an alarm log, provided this log file is configured. The capacity of the log file is limited by the storage medium and system limits.

Alarm view

The alarm view shows selected alarms or events from the alarm buffer or alarm log. Whether alarm events have to be acknowledged or not is specified in your configuration. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string in their alarm text are shown.

Alarm window

The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active. Depending on the configuration, it is not closed until the alarm is acknowledged.

You can configure the order in which the alarms are displayed. At the first position, the current, or the oldest alarm will be displayed. The alarm window can also be set to indicate the exact location of the fault, including the date and time of the alarm event. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string in their alarm text are shown.

Alarm indicator

The alarm indicator is a graphic symbol that is displayed on the screen when an alarm of the specified alarm class is activated.

The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated. The number indicates the number of queued alarms.

3.5.2 Alarm view (RT Professional)

Introduction

The "Alarm view" object displays alarms that occur during the process in a plant. You also use the alarm view to visualize alarms in list format. WinCC offers various views, such as "Current alarms" or "Historical alarm list" (short-term).

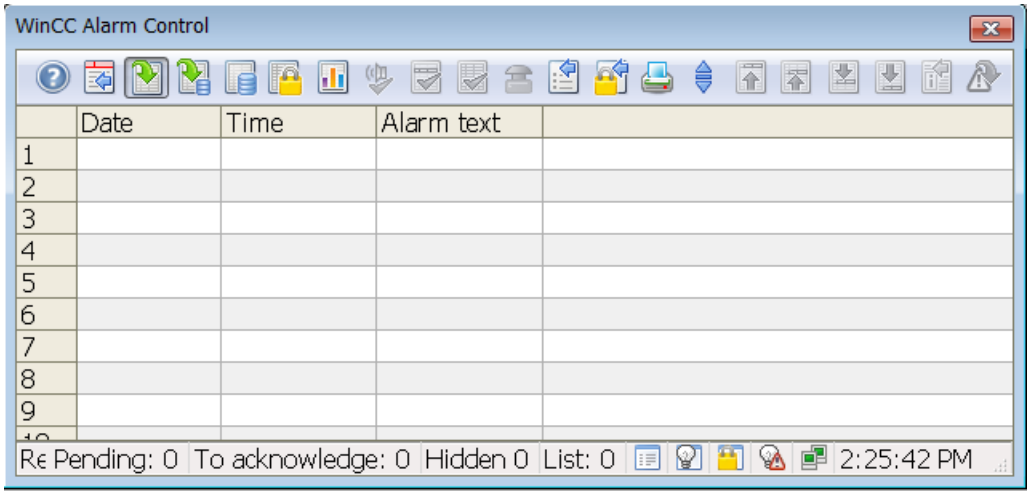

In Runtime, you can use the <TAB> key to jump to different objects in succession to enter values, or click buttons. Alternatively you can use the alarm view with the mouse.

Note

An alarm appears in the alarm view with the date and time stamp crossed out in the following situations:

- A locked alarm is unlocked.
- An alarm is reloaded after a power failure. This applies only to chronological messaging.
- The AS is restarted. This applies only to chronological messaging.

Requirement

- The "Loop-In-Alarm" button is configured in the alarm view.
- Cursor mode is set to "Tab cursor" for the screen.
- The objects are enabled for operation.
- The operator authorization is assigned.

Operation using the mouse

1. Click on the alarm to be edited.
2. Click on the operator control whose function you wish to use.

Operation using the keyboard

1. Press the <Tab> key until the list of displayed alarms is selected in the alarm view.
2. Click on the alarm to be edited. Use the <Up> and <Down> keys accordingly.
3. Press the <Tab> key until the operator control element you wish to use is selected.
4. Press <Enter>.

Alternative operation

Depending on the configuration, you can also operate the alarm view via the function keys.

Example: Change the order of the columns

The function can be configured. It is only available in Runtime if you have activated the alarm view "Font table > Header column > Sort" in the Inspector window.

1. Select the column header, such as the "Date" column header.
2. While holding down the mouse button, drag the column header to the column header "Time".

Example: Change the order of the columns

The function can be configured. It is only available in Runtime if you have activated the alarm view "Font table > Header column > Sort by column headers" in the Inspector window.

1. Click on the column header.
2. Click the same column header again to reverse the sort order.

Example: Loop-In-Alarm

Loop-In-Alarm triggers the configured change of screen to the screen containing information about the alarm.

1. Click on the alarm to be edited.
2. Click the "Loop-In-Alarm" button

Operation behavior

When Loop-In-Alarm is triggered for an unacknowledged alarm, it is acknowledged automatically.

See also

Alarm view (Page 420)

3.5.3 Alarm View Buttons (RT Professional)

Introduction

The alarm view displays specific alarm lists in order to provide a better overview of the active alarms. These lists filter, and sort alarms by certain properties. You can individually define these properties, and how the lists are presented when you configure the alarm view.

Lists in the alarm view

You can display six different lists in the alarm window. To display the alarm lists in the alarm view, click the associated button on the alarm view's toolbar.

	List	Description
	Alarm list	Shows the pending alarms.
	Short-term archive list	Shows the logged alarms. The display is updated immediately when new incoming alarms occur.
	Long-term archive list	Shows the logged alarms. The display is not updated immediately when new incoming alarms occur.
	Lock list	Shows the locked alarms.
	Alarm statistics	Shows statistical information.
	List of Hidden Alarms	Shows the hidden alarms. Hidden alarms are also logged.

Note

Hidden system diagnostic alarms

For hidden system diagnostic alarms, the alarm texts are not displayed in the list of hidden alarms.

The list of hidden alarms merely displays the configuration data of the alarm and not a live view.

To display active hidden system diagnostic alarms, you must adjust the filter for the online alarm list.

Filtered alarm list

You use the filter of the alarms view to configure alarm views in which you display specific alarm lists. You can configure a filter for any alarm list. Configure a separate alarm view for each filtered alarm list.

Note

An alarm appears in the alarm list with the date and time stamp crossed out in the following situations:

- A locked alarm is unlocked.
- An alarm is reloaded after a power failure; for chronological alarms only.
- The AS is restarted; for chronological alarms only.

See also

Filtering Alarms in Runtime (Page 932)

3.5.4 Filtering Alarms in Runtime (RT Professional)

Introduction

In Runtime, you can set specific criteria that define the alarms to be displayed in the alarm view. In the example below, only alarms that contain the alarm text "Motor on" are displayed.

Requirement

The "Filter alarms" button is configured in the alarm view.

Procedure

To filter alarms in the alarm view, proceed as follows:

1. Click the "Filter alarms" button in Runtime.
   The "Specify selection" dialog opens.
2. Double-click the "Text Blocks" folder in the tree structure, and click "Alarm Text". In the right pane, select the "Search string" option, and double-click "Search string".
3. The "Text input" dialog opens. Enter the search string "Motor on" and click "OK" to confirm.
4. In the "Specify selection" dialog, click "OK" to confirm your entries.
   The alarm view only shows the selected alarms.

In the "Specify selection" dialog, specify the start and end times, or search strings, for alarm text blocks such as the "Date" and "Time" system blocks. Your input must be in the format required in the dialog.

The following settings are available in the "Specify selection" dialog:

Field	Description
Text filter by exact match	If the "Text filter by exact match" option is not enabled, all text blocks that contain the specified string of characters are selected. If the "Text filter by exact match" option is enabled, all text blocks that correspond exactly to the specified string of characters are selected.
Persistence in RT	If the "Persistence in RT" option is enabled, changes to criteria are retained even after a screen change.
Persistence in CS and RT	If the "Persistence in RT and CS" option is enabled, the modified settings are also applied to the Engineering System. To do this, you must open the screen in the "Screens" editor, and save it again. The changed settings will also be used when you reactivate the project.
Delete filter	Use this button to delete all configured criteria.

Filter by process values

You cannot enter text as the criterion for a process value.

To filter by the text of a process value block, or the process tags it represents, proceed as follows:

1. Insert this process value as the alarm text for a user text block in an alarm.
2. In Runtime, filter the alarms by the text in the "Alarm text" text block.

Note

In multi-user systems, make sure that contents displayed in the "Specify selection" dialog on a client have the same names on all servers.

When filtering by time, the start and stop values are not adjusted automatically when the timebase of the alarm view is changed.

Example: At the PC location with time zone "UTC + 1h", the alarm view has the "Local time zone" time base. You should filter by the time 10:00 to 11:00. Change the timebase from "Local time zone" to "UTC". If you want to display the same alarms, change the filter to 9:00 to 10:00.

See also

Alarm view (Page 420)

3.5.5 Sorting Alarms in Runtime (RT Professional)

Introduction

In Runtime, you can sort the alarms in the alarm view by column header.

Examples for sorting alarms:

- In descending order by date, time, and alarm number. The most recent alarm is displayed at the top.
- By priority
  You must have defined the priority of the alarms in the "HMI alarms" editor and configured the "Priority" alarm text block in the alarm view. As a result, in a single-line alarm view, only the top-priority alarm appears in the alarm window. A lower-priority alarm will not be displayed, even if it is more recent. The alarms are displayed in chronological order.
- The "Alarm status" alarm text block is sorted by the type of state and not by the configured status texts. For an ascending sort order, the following order is used:
  - Incoming
  - Outgoing
  - Acknowledged
  - Locked
  - Enabled
  - Automatic acknowledgment
  - Emergency acknowledgment
  - Incoming/outgoing

  Note

  Specifying sorting criteria in the ES for the alarm view

  You can set the sort criteria to configure the alarm view as required. In the Inspector window of the alarm view, select "Properties > Properties > View > Columns".

Sorting alarms

To sort the alarm view by column, define the sort order over up to four columns. An arrow and a number are shown on the right in the column header. The arrow indicates the sort order (ascending or descending). The number beside the arrow indicates the sort order of the column headers.

Requirement

- "Sort by click" or "Sort by double-click" is selected in the alarm view.
  Click on "Properties > Properties > Table > Table - Sorting > Settings".
- The "Autoscroll" button is activated in the alarm view.
  Click "Properties > Properties > Toolbar > Toolbar buttons".

Procedure

To sort alarms in the alarm view by column, proceed as follows:

1. Click or double-click the column header by which you want to sort the alarms first.
   The number "1" is displayed, and the arrow points upwards for ascending sort order.
   - If sort order "Up and down" is enabled in the alarm view, each click or double-click in the column header toggles the sorting between the ascending and descending mode.
   - If sort order "Up, down, none" is enabled, then the sort order is cancelled after the 3rd click.
2. If you want to sort by several columns, click the column header in the required order.

Note

Sorting is not affected by events or changes to the alarm text block configuration.

An alarm text block corresponds to a field in an alarm. If you define an alarm text block as sorting criterion in the sort dialog and then remove this block from the alarm line under "Properties> Properties > Columns", the defined sort order remains unchanged. In the sort dialog, a blank field is displayed instead of the deleted alarm text block.

If you do not define another sort order order and reactivate the removed alarm text block in the alarm text block configuration, the alarms will be sorted in the original sort order once more.

3.5.6 Toggling the Alarm View in Runtime (RT Professional)

Introduction

You filter the alarms to be displayed in the alarm view using preconfigured buttons.

Requirements

- The six buttons needed to display a certain selection of alarms are configured in the alarm view.

Procedure

To display a certain selection of alarms in the alarm view, follow these steps:

- on one of the following buttons in the alarm view:
  Alarm list, to view alarms pending
  Historical alarm list (short-term), to view logged alarms
  display is updated immediately when new incoming alarms occur.
  Historical alarm list (long-term), to view logged alarms
  Lock list, to view the locked alarms
  Alarm statistics, to view statistical information
  List of hidden alarms, to view alarms which are logged but not visible.

3.5.7 Displaying logged alarms in Runtime (RT Professional)

Introduction

In Runtime, the alarm view displays alarms from the log, in addition to the current alarms.

Requirements

- All the archived data that you intend to display in Runtime must be stored locally on the archive server. The SQL server does not allow access to backup files held elsewhere, such as another computer on the network.
- The "Historical alarm list (short-term)" and "Historical alarm list (long-term)" buttons are configured in the alarm view.

Procedure

To logged alarms in Runtime, follow these steps:

1. In the alarm view, click on the "Historical alarm list (short-term)" button to display logged and current alarms. Any new incoming alarms will be immediately updated in the historical alarm list (short-term).

-OR-

1. In the alarm view, click on the "Historical alarm list (long-term)" button to display only logged alarms. The historical alarm list (long-term) only displays logged alarms. You can enter or view comments in the historical alarm list (long-term).

3.5.8 Printing Alarms in Runtime (RT Professional)

Introduction

You have the following options for printing alarms in Runtime, depending on the settings made in the "HMI alarms" and "Reports" editors:

Print alarm sequence report

If the alarm sequence report is activated, all alarm status transitions in Runtime are output continuously to a printer. No operator input is needed in Runtime.

Printing report

The operator input in Runtime depends on your configuration. Your options for printing an alarm report in Runtime are as follows:

- Time-driven output
  - One-off, time-driven output, e.g. at the start of a shift
  - Repeated output at specific intervals, such as every two hours
- Event-driven output
  - Change in value of a tag
  - By clicking a configured button in a screen
  - Click the button in the alarm view
  - Overflow of a log
  - Script

Note

Please note that only ASCII characters are supported when printing alarms directly in Runtime.

Requirement

- Several alarms are displayed on the HMI device.

Procedure

To print the current view in the alarm window in Runtime, follow these steps:

1. Filter the alarm view using the alarm view controls, if necessary.
2. Click the button in the alarm view.

Result

The alarms displayed in the alarm window are output to the printer that you configured in the "Reports" editor.

Printer settings

Configure the printer in the HMI device's Control Panel. The settings will depend on which operating system is used on the HMI device. For detailed information, refer to the operating instructions of your HMI device and to the "Reporting alarms (Page 873)" chapter.

See also

Working with reports (Page 1231)

3.5.9 Locking Alarms in Runtime (RT Professional)

3.5.9.1 Locking Single Alarms (RT Professional)

Introduction

If you lock an alarm, this alarm will not be displayed, and will not be logged. You can lock single alarms, alarm classes or alarm groups as required.

Note

Locked alarm:

Locked alarms are automatically unlocked when WinCC Runtime restarts. Only alarms that are locked directly in the AS by means of data blocks will continue to be locked (locking at source).

Locked alarm classes / alarm groups:

The locked state of alarm classes and alarm groups is not affected by a restart of WinCC Runtime.

Requirement

- The alarm view has been configured.
- The alarm view is displayed in Runtime.
- The "Lock alarm/Unlock alarm" / and "Locked alarms" buttons are configured.
- You are authorized to lock, and unlock alarms.

  Note

  The "Lock alarm", and "Unlock alarm" authorizations must be configured directly one above the other. This is necessary because the authorization level used automatically for the "Unlock alarms" authorization is directly below the "Lock alarms" authorization.

Procedure

To lock alarms and prevent them being displayed using the button, follow these steps:

1. Select the alarm that you want to lock from the alarm view.
2. Click the "Lock alarm" button.
   The alarm is removed from the alarm list.
3. To display all locked alarms, click the "Locked alarms" button.
   The lock list is displayed.

4. Select the alarm that you want to unlock.
5. Click the "Unlock alarm" button.
   The alarm is displayed again and removed from the lock list.

Note

Locking/unlocking an Alarm_8(P) alarm by means of S7PMC always causes all 8 alarms of this block to be locked, or unlocked.

See also

Alarm view (Page 420)

3.5.9.2 Locking alarms by means of alarm number (RT Professional)

Requirements

- The "Lock dialog" button is configured in the alarm view.

Procedure

Proceed as follows to lock alarms and prevent them being displayed using the alarm number:

1. Click the button.
   The "Configure Lock List" dialog opens.

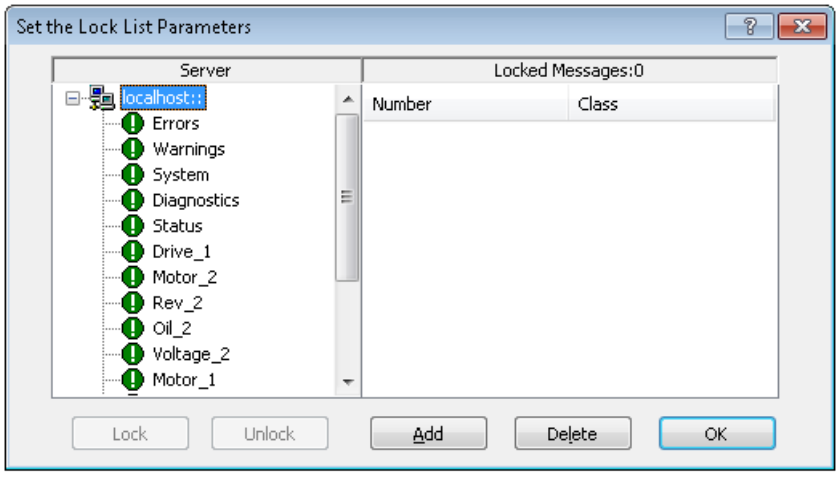

2. Click a server in the server list, or click the local PC for single-user projects.
3. Click the "Add" button. In the subsequent dialog, enter the alarm number that you want to lock.

4. You can lock multiple alarms at the same time. To do this, enter the alarm numbers, separated by commas. Enter a range of numbers in the format "5-10". Only contiguous number ranges are locked. If there are gaps in the specified range, an "Invalid range" message is output.
5. To unlock a locked alarm, select the alarm from the list of locked alarms, and click the "Delete" button.

Note

You cannot lock, or unlock more than 50 alarms at the same time.

3.5.9.3 Locking Alarms using the Alarm Class or Alarm Group (RT Professional)

Introduction

You can also lock alarms to prevent them from being displayed by locking the entire alarm class, or the alarm group to which the alarm belongs. None of the alarms from these locked alarm groups and alarm classes will be displayed.

Requirements

- The "Lock dialog" button is configured in the alarm view.

Procedure

To lock, or unlock alarms via the alarm class, or the alarm group, proceed as follows:

1. In the alarm view, click the button.
   The "Configure Lock List" dialog opens.

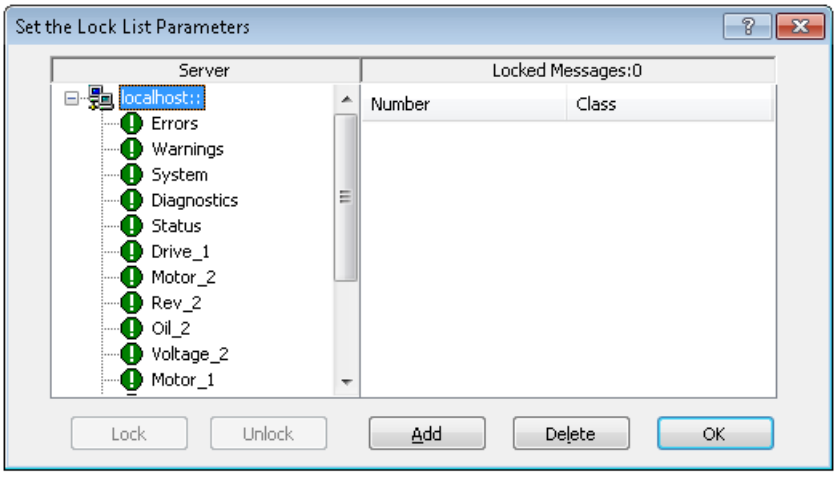

2. Click a server in the server list, or click the local PC for single-user projects.

3. In the tree structure, select an alarm class or user-defined alarm group that you want to lock.
4. Click the "Lock" button.
   All alarms from this alarm class or user-defined alarm group are locked.

To unlock the alarms, select the alarm class or user-defined alarm group and click on the "Unlock" button.

Operator input alarm when an alarm is locked

You can define in the alarm view, whether an operator input alarm is generated when an alarm is locked or unlocked. You should configure the operator input alarm yourself if needed. The operator input alarm contains the following objects by default:

- A time stamp
- The user logged on

The time stamp for the alarm originates as follows:

- From the alarm source (such as an AS) for active locking.

Actively-locked alarms are visible, and can be used on all HMI devices.

3.5.10 Acknowledgement of Alarms (RT Professional)

3.5.10.1 Acknowledging alarms (RT Professional)

Introduction

You can acknowledge alarms in Runtime according to your project configuration settings. You can acknowledge alarms as follows:

- Using the display and control object buttons
- With customized buttons

If an operator authorization is configured for an individual control, the alarms can only be acknowledged by authorized users.

To bypass the acknowledgement in Runtime, WinCC contains system functions and scripts that allow you to acknowledge alarms automatically.

Acknowledgement variants

You acknowledge individual alarms or multiple alarms together in Runtime. The following options are possible:

- Single acknowledgment
  Acknowledgment of an alarm using a button or a function key.
- Acknowledge alarm groups
  Acknowledgement of all the alarms of an alarm group using a button, or a function key.

- Group acknowledgement
  Acknowledgment of all pending alarms that require acknowledgment in the alarm view using the "Group acknowledgment" button of the alarm view.
- Dual-mode acknowledgment
  If an alarm requires dual-mode acknowledgment, you acknowledge both the incoming, and the outgoing alarm.

Requirement

- An alarm is displayed on the HMI device.
- The "Group acknowledgment" and "Single acknowledgment" buttons are configured in the alarm view.

Procedure

To acknowledge an alarm, follow these steps:

- Select the alarm.
- Click "Single acknowledgment" in the alarm view.

Result

The alarm status is set to "Acknowledged". If the trigger condition for an alarm no longer applies, the alarm status is also set to "outgoing" and no longer displayed on the HMI device.

See also

Alarm view (Page 420)

3.5.10.2 Group Alarm Acknowledgement (RT Professional)

Introduction

The acknowledgement of all pending, and visible alarms in the alarm window that need to be acknowledged is known as a group acknowledgement. If the "Single alarm" option is not selected for an alarm, you can trigger a group acknowledgment.

Requirement

There are several alarms that require acknowledgement in the alarm view.

Procedure

To group-acknowledge an alarm, follow these steps:

1. Read the alarm texts of the queued alarms and initiate remedial action, if necessary.
2. In the alarm view, click the button.

Result

All queued alarms with the following properties are acknowledged:

- Requires acknowledgement
- Does not require acknowledgement
- Visible

3.5.10.3 Emergency acknowledgment of alarms (RT Professional)

Introduction

In exceptional situations, the operator acknowledges an alarm based on its alarm number. The acknowledgment bit is sent to the AS even if the alarm is currently not pending. This function is only intended for use in imminent emergency situations.

Requirement

The button is configured in the alarm view.

Procedure

To acknowledge an alarm in Runtime using an emergency acknowledgement, follow these steps:

1. Click the button in the alarm view.
   The "Emergency Acknowledgment for Alarms" dialog opens.

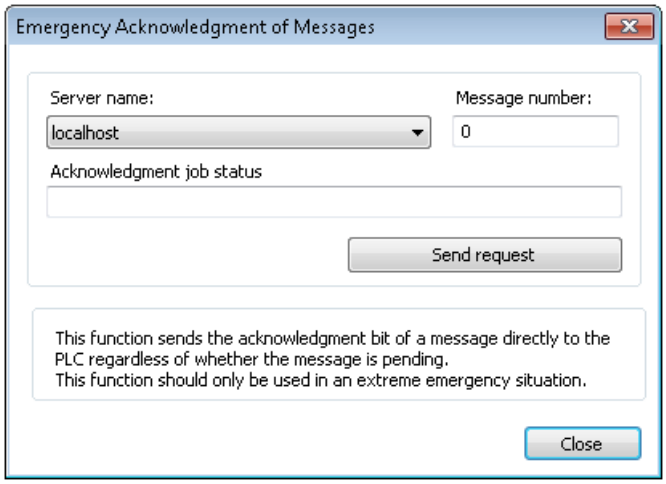

2. Select a server.

3. To enter the alarm number, select "Alarm number".
4. Click the "Send acknowledge job" button.
   The alarm then appears in the alarm view in the color that you defined for an acknowledged alarm.

Note

In the dialog for selecting the servers, only server projects whose "Packages" are loaded onto the PC are shown.

3.5.10.4 Acknowledging an alarm annunciator (RT Professional)

Introduction

The way you acknowledge an alarm annunciator in Runtime will depend on how you configured its acknowledgement.

Requirements

- The alarm annunciator and its acknowledgment are configured.
- The alarm annunciator is triggered.
- The "Alarm annunciator acknowledgment" and "Single acknowledgment" buttons are configured in the alarm view.

Acknowledgment using a separate key

- To acknowledge the alarm annunciator, click the "Alarm annunciator acknowledgment" button in the alarm view.
  You only acknowledge the alarm annunciator. The alarm that triggered the alarm annunciator is still set to "incoming" status.

Single acknowledgment

- To acknowledge the alarm annunciator, select the current alarm, and click the "Single acknowledgment" button in the alarm view.
  You thereby acknowledge the alarm annunciator and the alarm that triggered the alarm annunciator.

Acknowledgment tag

If the alarm annunciator is acknowledged using a tag, acknowledge the alarm annunciator using a customized button or function key, etc.

Result

This sets the acknowledgment bit of the alarm annunciator.

See also

Alarm view (Page 420)

3.6 Displaying security events

3.6.1 Displaying security events on the HMI device (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can also view security events on the HMI device for existing alarms from WinCC.

Security events are, for example, an attack on a device over the network or a change of the protection level for communication between the controller and the HMI device.

Security events are detected by the controller and forwarded to the HMI devices. The security events are displayed in the alarm log on the HMI devices.

A configuration or activation of the security events functionality is not necessary within the controller. Security events are automatically detected by the controller.

Restrictions

You can display security events from the following controllers on the HMI device:

- SIMATIC S7-1200
- SIMATIC S7-1500

Note

Device dependency

The security events of controllers are not displayed on Basic Panels and Basic Panels 2nd Generation.

Configuring the display of security events

The following steps are required to display security events on the HMI device:

- Selection of controller alarms
- Creation of an alarm log for controller alarms

You can find more detailed information on configuration here:Configuring the display of security events (Page 947)

See also

Configuring the display of security events (Page 947)

3.6.2 Configuring the display of security events (RT Professional, RT Advanced, Panels, Comfort Panels)

Requirements

A HMI connection is established between the HMI device and PLC.

Procedure

1. Open the "Runtime settings" of the HMI device.
2. Enable the "Security events" function under "Alarms".

   Note

   The "Security events" function is deactivated by default and must be activated for each HMI connection.

3. Create a new alarm log in the "Log" editor under "Alarm logs".
4. Open an HMI screen.
5. Create an alarm view.

Result

The security events are displayed in the alarm log.

3.7 Sending a complete alarm from the controller to the HMI device

3.7.1 Sending a complete alarm from the controller to the HMI device and automatic update (RT Advanced, Panels, Comfort Panels)

Basics

In addition to alarms in WinCC, you can configure controller alarms in STEP 7 and show them on your HMI device.

When controller alarms are configured in STEP 7, a connection to an S7-1500 controller is established, and controller alarms are currently pending, the controller alarms are automatically updated and sent to the HMI devices after alarm changes (e.g. change of the alarm text). This will save you time because you do not have to load configuration changes of the alarms to the HMI device separately. The HMI device does not need to exit Runtime operation when the alarms are changed.

The following controller alarms can be sent to the HMI device:

- System diagnostics
- ProDiag alarms
- Program alarms

The controller alarms can be sent completely to the HMI devices with device version V14 if the corresponding settings are configured in the controller and on the HMI device. The option "Automatic update" must be activated on the HMI device for the respective connection under "Runtime settings Alarms > Controller alarms". You can find additional information on the settings under Configuring automatic update of controller alarms on the HMI device (Page 951).

Device dependency

The controller alarms of the following controller are sent automatically and completely to the HMI device when they occur:

- SIMATIC S7-1500 (firmware version 2.0 and higher)

Basic Panels and Basic Panels 2nd Generation do not support controller alarms. For Basic Panels and Basic Panels 2nd Generation as well as HMI devices with a device version before V14, configuration changes of the alarms must be loaded to the controller and the HMI device.

When "Automatic update" is deactivated in the HMI device, configuration changes of the alarms must be loaded to the controller and the HMI device.

Note

When upgrading a project to the TIA Portal version V14, the transfer of controller alarms to the HMI device is deactivated under the "Runtime settings" of the HMI device because the connection between the HMI device and the respective controller was created in an older TIA Portal version.

Language settings

For alarms to be displayed in the correct language on the HMI device, the same three languages or fewer must be configured for the alarms in the controller and on the HMI device. You may have to coordinate the language selection with the configuration engineer.

If different languages are configured on the HMI device and in the controller, the HMI device shows the text "###Text missing###" rather than the controller alarms during operation.

Note

If you want to use more than three languages simultaneously on the HMI device, you must deactivate the option for automatic update of the controller alarms. Otherwise, "##Text missing##" is displayed instead of the controller alarms for the fourth and any additional language.

Special considerations for Asian fonts

When using Asian languages, only the Asian characters that were used in configuring the HMI device are loaded to the HMI device during the transfer. If the texts are sent directly from the controller to the HMI device, you may encounter display problems. You can load the complete Asian font to the HMI device in these cases. You can find additional information at Loading Asian fonts to the HMI device (Page 2348).

Notes and rules

- Modified texts that you load to the controller are displayed immediately on the HMI device when the controller alarm has already been triggered.
- The following applies to alarms that are displayed in the alarm log:
  - The alarm log includes alarms in the set language. This language is set during configuration of the HMI device under "Runtime settings > General > Logs > Logging language". Regardless of whether you set a specific language for logging of the controller alarms in the alarm log or select the default language, the respective language must also be activated in the controller. When you specify startup language, this can lead to mixed languages in the alarm log in conjunction with the "Automatic update" option. You can find additional information on language selection of the log in Runtime under Log response to language switching in runtime (Page 896).
  - The check box "Log event text and error location" must be selected for the respective alarm log under "Properties > Properties > Logging method" so that the logged controller alarms are displayed in the alarm view or in the alarm window.

See also

Loading Asian fonts to the HMI device (Page 2348)

Configuring automatic update of controller alarms on the HMI device (Page 951)

Log response to language switching in runtime (Page 896)

3.7.2 Sending a complete alarm from the controller to the HMI device and automatic update (RT Professional)

Basics

In addition to alarms in WinCC, you can configure controller alarms in STEP 7 and show them on your HMI device.

When controller alarms are configured in STEP 7, a connection to an S7-1500 controller is established and controller alarms are currently pending, the controller alarms are automatically sent to the HMI devices and updated in case of changes (e.g. change to alarm text). This enables you to save time as you do not have to load the configuration changes of the alarms to the HMI device separately. The HMI device does not need to exit Runtime operation when the alarms are changed.

The following controller alarms can be sent to the HMI device:

- System diagnostics
- Process diagnostics
- Program alarms

The controller alarms can be sent in their entirety to the HMI devices with device version V14 if the corresponding settings are configured in the controller and on the HMI device. On the HMI device, the "Automatic update" option must be activated under "Runtime Settings > Alarms > Controller alarms" for the relevant connection. You can find additional information on the settings under Configuring automatic update of controller alarms on the HMI device (Page 951).

Device dependency

When they occur, the controller alarms of the following controller are automatically sent in their entirety to the HMI device:

- SIMATIC S7-1500 (as of firmware version 2.0.)

Basic Panels and Basic Panels 2nd Generation do not support controller alarms. For Basic Panels and Basic Panels 2nd Generation as well as HMI devices with a device version lower than V14, the configuration changes of the alarms must be loaded to the controller and the HMI device.

If "Automatic update" is deactivated in the HMI device, the configuration changes of the alarms must be loaded to the controller and the HMI device.

Note

When upgrading a project to the TIA Portal version V14, the transfer of the controller alarms to the HMI device under "Runtime Settings" of the HMI device is deactivated because the connection between the HMI device and the respective controller was created in an older TIA Portal version.

Language settings

For alarms to be displayed in the correct language on the HMI device, the same three languages (or fewer) must be configured for the alarms in the controller and on the HMI device. You may have to coordinate the language selection with the configuration engineer.

If different languages are configured on the HMI device and in the controller, the HMI device in operation shows the text "###Text missing###" instead of the control alarms.

Note

If you want to use more than three languages simultaneously on the HMI device, you must deactivate the option for automatic update of the controller alarms. Otherwise, "##Text missing##" is displayed instead of the controller alarms for the fourth and any additional language.

See also

Editing controller alarms (Page 825)

Controller alarms (Page 762)

Configuring automatic update of controller alarms on the HMI device (Page 951)

3.7.3 Configuring automatic update of controller alarms on the HMI device (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

The option "Automatic update" is selected by default for all connections between the S7-1500 controllers (firmware version 2.0) and the HMI devices with the device version V14.

If the connection between an HMI device and an S7-1500 controller was created in a TIA Portal version before V14, the option "Automatic update" is deactivated even after upgrading the project to version V14. You can configure automatic updating of the controller alarms later for this connection.

Requirements

- There is an HMI connection between the HMI device and an S7-1500 controller (firmware version 2.0 or later) or a compatible device (for example, WinAC 1500).
- Controller and HMI device have at least version 14 or later.
- "Central alarm management" is activated in the controller (automatic updating was activated for the controller).
- Controller alarms were configured in STEP 7.
- An alarm view or an alarm window was configured on the HMI device.

- The alarm classes associated with the controller alarms are selected in the alarm view or in the alarm window.
- The same three languages (or fewer) are configured in the controller and on the HMI device for alarms.

Procedure

1. Open the "Runtime settings" of the HMI device.
   One or more connections to controllers are displayed under "Alarms > Controller alarms".
2. Activate the "Automatic update" option for the respective connection for which you want to display the controller alarms.
   The "Automatic update" function must be activated separately for each connection.

Note

If the languages configured for the controller differ from those for the HMI device, the alarms cannot be displayed in Runtime. Instead, the alarm "##Text missing##" is displayed.

Note

If you want to use more than three languages simultaneously on the HMI device, you must deactivate the "Automatic update" option. Otherwise, the alarm "##Text missing##" is displayed instead of the controller alarms for the fourth and any additional language.

Result

In Runtime, the controller alarms are displayed in the alarm view.

See also

Sending a complete alarm from the controller to the HMI device and automatic update (Page 950)

Sending a complete alarm from the controller to the HMI device and automatic update (Page 948)

3.8 Reference

3.8.1 System functions for alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

System functions

System functions are predefined functions you can use to implement many tasks in runtime, even with no programming knowledge. You use system functions in a function list or in a script.

Note

Device dependency

Scripts are not available for all HMI devices.

The table shows all the system functions available for displaying and editing alarms.

System function	Effect
EditAlarm	Triggers the "Loop-In-Alarm" event for the selected alarm or for the last selected alarm in case of multiple selection. If the alarm to be edited has not yet been acknowledged, the selected alarm is also acknowledged.
ClearAlarmBuffer	Deletes alarms from the alarm buffer on the HMI device.
ClearAlarmBufferProtoolLegacy	Function such as "ClearAlarmBuffer". This system function has been retained to ensure compatibility and uses the old ProTool numbering.
AlarmViewUpdate	Updates the alarm view.
‣AlarmViewLoopInAlarm	Triggers the "Loop-In-Alarm" event for the selected alarm or for the last selected alarm in case of multiple selection. If the alarm to be edited has not yet been acknowledged, the selected alarm is also acknowledged.
AlarmViewAcknowledgeAlarm	Acknowledges the alarms that are selected in the specified alarm view.
AlarmViewShowOperatorNotes	Displays the configured tooltip for the alarm selected in the specified alarm view.
AcknowledgeAlarm	Acknowledges all selected alarms.
SetAlarmReportMode	Switches the automatic reporting of alarms on the printer on or off.
ShowAlarmWindow	Hides or shows the alarm window on the HMI device.
ShowSystemEvent	Displays the value of the delivered parameter as a system event on the HMI device.

Note

Device dependency

The following system functions are not available for all HMI devices:

- SetAlarmReportMode
- AlarmViewUpdate
- ShowSystemEvent

3.8.2 Alarm events (RT Advanced, Basic Panels, Panels, Comfort Panels)

Alarm events and their display and control objects

In Runtime, the following alarm events are triggered by their display and control objects. You can configure a function list for every event.

Object	Configurable events
Alarms	Incoming Outgoing Acknowledge Loop-In-Alarm
Alarm view	Enabling Disable Selection changed
Alarm indicator	Click Click when flashing

Note

Device dependency

The following events in the alarm view are not available for all HMI devices:

- Enabling
- Disable
- Selection changed
- Click
- Click when flashing

3.8.3 Alarm events (RT Professional)

Alarm events and their display and control objects

In Runtime, the following alarm events are triggered by their display and control objects. You can configure a function list for every event.

Object	Configurable events
Alarms	Incoming Outgoing Acknowledge Loop-In-Alarm Status changed
Alarm view	Activate Loaded Click toolbar button Property changed Change toolbar button property Property status bar element change Block property modified Property column changed Property alarm statistics column changed Property operator input alarm changed

3.8.4 System functions for logs (RT Professional, RT Advanced, Panels, Comfort Panels)

System functions

The following system functions are available for logging:

Function name	Function method
ArchiveLogFile	This function moves or copies a log to another storage location for long-term archiving. Use the system function, for example, to move the audit trail from a local storage medium to the server. When handling audit trails, always use the "Move log (hmiMove)" mode in order to avoid noncompliance with FDA guidelines as a result of redundant data storage.
LogTag	Saves the value of the given tags in the given data log. Use the system function to log a process values at a specific time.
StartLogging	Starts the logging process in the specified log. You can interrupt logging in Runtime by calling the "StopLogging" system function.
StopLogging	Stops the logging process in the specified log. To resume logging in Runtime, select the "StartLogging" system function.
ClearLog	Deletes all entries in the specified log.

Function name	Function method
StartNextLog	Stops the logging process in the specified log. Logging is continued in the sequential log that has been configured for the specified log file.
CloseAllLogs	Closes all logs. The connection between WinCC and the log files or log database is terminated. You can use this system function, for example, to enable hot-swapping of the storage medium on the HMI device without exiting the Runtime software.
OpenAllLogs	Opens all logs to resume logging. The connection between WinCC and the log files or log database is recovered.
CopyLog	Copies the contents of the specified log to another log.

3.8.5 Structure of *.csv files that contain alarms (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can save an alarm log as a file in "*.csv" format. CSV stands for **C**omma **S**eparated **V**alue. In this format, the table columns that contain the names and the value of the entry are separated by semicolons. Each table row terminates with a line break.

Example of a log file in *.csv format

This example shows a file with logged alarms:

"Time_ms";"MsgProc";"StateAfter";"MsgClass";"MsgNumber";"Var1";...;"Var8";"TimeString";"MsgText";"PLC"37986550590,27;1;1;3;110001;"";...;"";"30.06.99 13:12:51";"Change to operating mode 'online'";37986550682,87;1;1;3;140010;"";...;"";"30.06.99 13:12:59";"Connection established: PLC_1, Station 2, Rack 0, Position 2";

Structure of a log file in *.csv format

The following values are entered in the log file columns:

Parameters	Description
Time_ms	Specify a time stamp as a decimal value (see below for conversion)
Msg_Proc	Alarm procedures: 0 = Unknown alarm procedure 1 = System event 2 = Alarm bit procedure (operating alarms) 3 = Alarm number procedure ALARM_S 4 = Diagnostic event 7 = Analog alarm procedure 100 = Alarm bit procedure (fault alarms)

Parameters	Description
State after	Alarm event: 0 = Incoming/Outgoing 1 = Incoming 2 = Incoming/Acknowledged/Outgoing 3 = Incoming/Acknowledged 4 = All alarms in the PLC were deleted with SFC106 or after PLC STOP/RUN an alarm is pending 6 = Incoming/Outgoing/Acknowledged
Msg_Class	Alarm class: 0 = No alarm class 1 = "Errors" 2 = "Warnings" 3 = "System" 4 = "Diagnostic Events" 64 ... = user-configured alarm classes
Msg Number	Alarm number
Var1 to Var8	Value of the trigger tag as a STRING
Time string	Time stamp, as STRING in legible data format
Msg text	Alarm in a readable STRING
PLC	Alarm localization (relevant PLC)

Conversion of the time stamp decimal value

To further process the time stamp value using a different program, proceed as follows:

1. Divide Time_ms by 1,000,000.
   Example: 37986476928 : 1,000,000 = 37986.476928
2. The whole number portion (37986) is the date calculated from 12/31/1899.
   In Excel you can now convert the time stamp into days. To do this, assign to the cell that contains the time stamp, a respective format form the "Date" group.
   Result: 37986 results in 12/31/2003
3. The value after the point (0.476928) indicates the time:
   - Multiply the value (0.476928) by 24 to obtain the hours (11.446272).
   - Multiply the remainder (0.446272) by 60 results in the minutes (26.77632).
   - Multiply the remainder (0.77632) by 60 results in the seconds (46.5792).

   Result 11:26:46.579
   This conversion is supported by Microsoft Excel, for example.

3.8.6 Description of the System Blocks (RT Professional)

Introduction

System blocks allow you to specify predefined information that is not freely utilizable, such as date, time and duration.

The value of the alarm text block (such as the time of day) is displayed in the alarm. The description of the system blocks provides information on the individual system blocks.

Overview

System block	Description	Default length
Date	Date of the "incoming", "outgoing", and "acknowledged" alarm statuses.	8
Time	Time of the "incoming", "outgoing", and "acknowledged" alarm statuses. Accuracy of a WinCC time stamp: 1 s. Display accuracy: 10 ms.	9
Duration	Time period between the "incoming", "outgoing", and "acknowledged" statuses. There is only one column for duration: • The column remains blank for alarms with "incoming" status. • For alarms with "outgoing" status, the time that elapses between "incoming" and "outgoing" is displayed. • When alarms are acknowledged, the time between "incoming" and "outgoing" is displayed.	8
Daylight Saving Time/ Standard Time	Indicates whether daylight saving time is set.	1
Status	Indicates the alarm status, such as "Incoming" or "Outgoing". The letters displayed in this system block to indicate the alarm status (such as "Q") are assigned in the alarm class.	1
Acknowledgment status	Indicates whether an alarm has been acknowledged. The letters that are displayed in this system block to indicate the acknowledgment status are configured in the alarm class.	1
ID	Alarm number	3
Alarm class	Alarm class The text is user-defined.	8
Controller/CPU number	Number of the CPU and AS in which the alarm was triggered. In Runtime, the value of this system block is not taken from the AS but rather is taken from the configured data for the alarm. The value has no function in communication with the AS.	2
Tag	Tag name for the operator input alarm from I/O field (and comparable object which can output operator input alarms)	1
Log	Flag indicating whether the alarm is to be logged.	1
Comment	Indicates whether there is a comment for this alarm. A comment is the users input about an alarm, such as "This alarm occurred today because...".	1

System block	Description	Default length
Info text	Indicates whether there is an infotext (tooltip) for this alarm. Tooltips contain information about the alarm, limited to 255 characters, such as "This alarm may occur if ...". Tooltip cannot be edited in Runtime.	1
Loop-In-Alarm	Indicates whether the "Loop-In-Alarm" function is enabled.	1
Computer Name	Indicates the name of the PC: • If the user has entered a comment in the "Historical alarm list (long-term)" and then changed the view, the name of the PC used appears in the "Historical alarm list (short-term)" and "Historical alarm list (long-term)". • If an alarm was acknowledged on a PC, the name of the PC appears in the "Historical alarm lists" (short and long-term) in the operator input alarm. The operator input alarm must be enabled under "Operator input alarm" in the Inspector window for the alarm view.	10
User Name	Indicates the name of the user (login name): • If you enter a comment in the "Historical alarm list" (long-term) and then exit the view, your user name is displayed in the "Historical alarm list" (short-term) and "Historical alarm list" (long-term). • If an alarm was acknowledged by a user logged on to WinCC, the user name is displayed in the "Historical alarm lists" (short and long-term) in the operator input alarm. The operator input alarm must be enabled under "Operator input alarm" in the Inspector window for the alarm view.	10
Priority	Defines the priority of an alarm. You can sort the alarms in the alarm view by priority. By sorting by priority, you can ensure that, in a single-line alarm view, the most important alarm (high priority) is shown in the display area. A lower-priority alarm will not be displayed, even if it is more recent. Value range: 0 - 16 In WinCC, the value corresponding to the highest priority is not specified.	3

Note

If you enabled the "Change ISO8601 format on all components" option in the shortcut menu of WinCC RT Professional, "Properties > Time settings > Central date and time formatting", the date and time are also displayed in the system blocks of the alarms in ISO8601 format.

3.8.7 SQL Statements for Filtering the Alarm View (RT Professional)

Introduction

In the alarm view, you can only use SQL statements that can be generated using the "Specify selection" dialog. The following conditions apply to WinCC:

- Structure: "Field", "Address", and "Value"
  Separate individual parameters with spaces.
  Example:
  DATETIME >= '2006-12-21 00:00:00.000' AND MSGNR >= 100
  All alarms from 21.12.2006 with an alarm number greater than or equal to 100
- The following information is entered in single quotes:
  - Character strings
  - Date
  - Time of day
- In the DATETIME argument, the date and time of day are separated by a blank. Regardless of the time base setting in the object properties, the output of DATETIME is based on the time base Local Time. The exception to this is when "UTC" is set as the time base. In this case, output is based on the time base "UTC".

Permissible Arguments

Name	SQL name	Type	Data	Example
Date/time	DATETIME	Date	'YYYY-MM-DD hh:mm:ss.msmsms'	DATETIME >= '2007-05-03 16:00:00.000' Outputs alarms as of 05/03/2007, 16:00 hours.
Number	MSGNR	Integer	Alarm number	MSGNR >= 10 AND MSGNR <= 12 Outputs the alarms with alarm numbers 10 - 12.
Class	CLASS IN	Integer	Alarm class ID 1-256 and "System" alarm classes 17 + 18	CLASS IN (1) Outputs alarms of alarm class 1.
Status	STATE	Integer	Value "ALARM_STATE_xx" Only the operands "=" and "IN(...)" are permitted. ALARM_STATE_1 ALARM_STATE_2 ALARM_STATE_3 ALARM_STATE_4 ALARM_STATE_10 ALARM_STATE_11 ALARM_STATE_16 ALARM_STATE_17	STATE IN(1,2,3) Outputs all alarms that came in, went out and were acknowledged. Possible values: 1 = incoming alarms 2 = outgoing alarms 3 = acknowledged alarms 4 = blocked alarms 10 = hidden alarms 11 = displayed alarms 16 = alarms acknowledged by the system 17 = Emergency-acknowledged alarms

Name	SQL name	Type	Data	Example
Priority	PRIORITY	Integer	Alarm priority 0 - 16	PRIORITY >= 1 AND PRIORITY =< 5 Outputs alarms that have a priority between 1 and 5.
Controller number	AG_NR	Integer	Controller number	AG_NR >= 2 AND AG_NR <= 2 Outputs alarms with the Controller number 2.
CPU number	CPU_NR	Integer	AG sub-number	CPU_NR >= 5 AND CPU_NR <= 5 Outputs alarms with AG sub-number 5.
Instance	INSTANCE	Text	Instance	-
Block: 1 ... Block: 10	TEXTxx	Text	Search string for Text1 - Text10	TEXT2 = "Error" Outputs the alarms whose Text2 corresponds with "Error". TEXT2 IN ('Error','Fault') Outputs the alarms whose Text2 corresponds with the text "Error" or "Fault". TEXT2 LIKE 'Error' Outputs the alarms whose Text2 includes the text "Error".
Process value: 1 ... Process value: 10	PVALUExx	Double	Search value for PVALUE1-PVALUE10	PVALUE1 >= 0 AND PVALUE1 <= 50 Outputs process value 1 with start value 0 and stop value 50.

Permissible Operands

- **>= , <= , = , > , <**
- **IN(...)** : Several values as an array, separated by commas, e.g.: CLASS IN(1 ,2 ,3).
- **LIKE** : Text must contain string, e.g.: TEXT1 LIKE 'Error' relays alarm where Text1 contains the search string "Error". The operand LIKE is only allowed for "text" type arguments.

Illegal arguments and operands

Only the arguments indicated in the table and operands from the list are permitted.

Grouping arguments, with brackets, for example, is not permitted.

3.8.8 System events

3.8.8.1 Basics on system events (RT Advanced, Basic Panels, Panels, Comfort Panels)

System events

System events on the HMI device provide information about internal states of the HMI device and PLC.

The following overview illustrates when a system event occurs and how to eliminate the cause of error.

Note

HMI device dependency

Some of the system events described in this section apply to the individual HMI devices based on their scope of functions.

Note

System events are output in an alarm view. System events are output in the language currently set on your HMI device.

The time format (AM/PM or 24-hour format) is based on the selected language. If there is no translation of the alarm texts in this language, English or Russian is selected as a replacement language and displays the appropriate time format.

System event parameters

System events may contain encrypted parameters. The parameters are of relevance when troubleshooting because they provide a reference to the source code of the Runtime software. These parameters are output after the "Error code:"

3.8.8.2 9999 - Global alarm (RT Advanced, Panels, Comfort Panels)

Meaning of the system event

The most important system events are listed below.

Table 3-1 9999 - Global alarm

Number	Effect/cause	Remedy
9999	The system start was interrupted. Check the following: • Line voltage • Cable connections • System settings • Configurations • Conversions	Check the power supply and cable connections. Check the system settings. Check the configurations and conversions. If the error occurs repeatedly, contact the hotline!

3.8.8.3 10000 - Printer alarms (RT Advanced, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-2 10000 - Printer alarms

Number	Effect/cause	Solution
10000	The print job could not be started or was canceled due to an unknown error. Faulty printer setup. Or: No authorization is available for accessing the network printer. The power supply was interrupted during data transfer.	Check the printer settings, cable connections and the power supply. Set up the printer once again. Obtain a network printer authorization. If the error persists, contact the hotline!
10001	No printer is installed or a default printer has not been set up.	Install a printer and/or select it as the default printer.
10002	The buffer for printing graphics is full. Up to two graphics are buffered.	Allow more time between print jobs.
10003	Graphics can now be buffered again.	--
10004	The buffer for printing lines in text mode (e.g. alarms) is full. Up to 1000 lines are buffered.	Allow more time between print jobs.
10005	Text lines can now be buffered again.	--
10006	The Windows printing system reports an error. Refer to the output text and the error ID to determine the possible causes. Nothing is printed or the printing is incorrect.	If necessary, repeat the action.

3.8.8.4 20000 - Global script alarms (RT Advanced, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below. The system events are divided into different ranges.

Table 3-3 20000 - Global script alarms

Number	Effect/causes	Solution
20010	An error has occurred in the specified script line. Execution of the script was therefore aborted. Note the system event that may have occurred prior to this.	Select the specified script line in the configuration. Ensure that the tags used are of the allowed types. For system functions, verify the correct number of parameters and the parameter types.
20011	An error has occurred in a script that was called by the specified script. Execution of the script was therefore aborted in the called script. Note the system event that may have occurred prior to this.	In the configuration, select the script that has been called directly or indirectly by the specified script. Ensure that the tags used are of the allowed types. For system functions, verify the correct number of parameters and the parameter types.

Number	Effect/causes	Solution
20012	Inconsistent configuration data. The script could therefore not be generated.	Recompile the configuration data.
20013	Incorrect installation of the scripting component of WinCC Runtime. Therefore, no scripts can be executed.	Reinstall WinCC Runtime on your PC. Recompile your project using the "Compile > Software (rebuild all)" command from the context-sensitive menu and then download the project to the HMI device.
20014	The system function returns a value that is not written in any return tag.	Select the specified script in the configuration. Check whether the script name has been assigned a value.
20015	Too many successive scripts have been triggered in short intervals. When more than 20 scripts are queued for processing, any subsequent scripts are rejected. In this case, the script indicated in the alarm is not executed.	Check what is triggering the scripts. Extend the times, e.g. the acquisition cycle of the tag initiating the script.

3.8.8.5 30000 - Alarms errors when using system functions (RT Advanced, Basic Panels, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below. The system events are divided into different ranges.

Table 3-4 30000 - Alarms errors when using system functions

Number	Effect/causes	Remedy
30010	The tag could not accept the function result, e.g. when it has exceeded the value range.	Check the tag types of the system function parameters.
30011	A system function could not be executed because the function was assigned an invalid value or type in the parameter.	Check the parameter value and tag type of the invalid parameter. If a tag is used as a parameter, check its value.
30012	A system function could not be executed because the function was assigned an invalid value or type in the parameter.	Check the parameter value and tag type of the invalid parameter. If a tag is used as a parameter, check its value.

3.8.8.6 40000 - Linear scaling alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below. The system events are divided into different ranges.

Table 3-5 40000 - Linear scaling alarms

Number	Effect/causes	Remedy
40010	The system function could not be executed since the parameters could not be converted to a common tag type.	Check the parameter types in the configuration.
40011	The system function could not be executed since the parameters could not be converted to a common tag type.	Check the parameter types in the configuration.

3.8.8.7 50000 - Data server alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below. The system events are divided into different ranges.

Table 3-6 50000 - Data server alarms

Number	Effect/causes	Remedy
50000	The HMI device is receiving data faster than it is capable of processing. Therefore, no further data is accepted until all current data have been processed. Data exchange then resumes.	--
50001	Data exchange has been resumed.	--

3.8.8.8 60000 - Windows function alarms (RT Advanced, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-7 60000 - Win32 function alarms

Number	Effect/causes	Solution
60000	This alarm is generated by the "ShowSystemEvent" system function. The text to be displayed is transferred to the function as a parameter.	--
60010	The file could not be copied in the direction defined because one of the two files is currently open or the source/target path is not available. It is possible that the Windows user has no access rights to one of the two files.	Restart the system function or check the paths of the source/target files. Under Windows: Users executing WinCC Runtime must be granted access to the files.
60011	An attempt was made to copy a file to itself. It is possible that the Windows user has no access rights to one of the two files.	Check the path of the source/target file. In Windows with NTFS: Users executing WinCC Runtime must be granted access to the files.

3.8.8.9 70000 - Windows function alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-8 70000 - Win32 function alarms

Number	Effect/causes	Remedy
70010	The application could not be started because it could not be found in the path specified or there is insufficient memory space.	Check whether the application exists in the specified path or close other applications.
70011	The system time could not be modified. The error message only appears in connection with area pointer "Date/time PLC". Possible causes: • An invalid time was transferred in the job mailbox. • The Windows user has no right to modify the system time. If the first parameter in the system event is displayed with the value 13, the second parameter indicates the byte containing the incorrect value.	Check the time that is to be set. Under Windows: Users running WinCC Runtime must be granted the right to set the system time of the operating system.
70012	Error when executing the system function "StopRuntime" with the "Runtime and operating system" option. Windows and WinCC Runtime are not closed. The error may have been generated because other programs cannot be closed.	Close all programs currently running. Now close Windows.

Number	Effect/causes	Remedy
70013	The system time could not be modified because an invalid value was entered. An incorrect delimiter may have been used.	Check the time which is to be set.
70014	The system time could not be modified. Possible causes: • An invalid time was transferred. • The Windows user has no right to modify the system time. Windows rejects the setting request.	Check the time that is to be set. Under Windows: Users running WinCC Runtime must be granted the right to set the system time of the operating system.
70015	The system time could not be read because Windows rejects the reading function.	--
70016	An attempt was made to select a screen by means of a system function or job. This is not possible because the configured screen number does not exist. Or: A screen could not be generated due to insufficient system memory. Or: The screen is blocked. Or: Screen call has not been executed correctly.	Check the screen number in the system function or job against the screen numbers configured. Assign the number to a screen if necessary. Check the details for the screen call and whether the screen is blocked for specific users.
70017	Date/time is not read from the area pointer because the address set in the PLC is either not available or has not been set up.	Change the address or set up the address in the PLC.
70018	Acknowledgment that the password list has been successfully imported.	--
70019	Acknowledgment that the password list has been successfully exported.	--
70020	Acknowledgment for activation of alarm reporting.	--
70021	Acknowledgment for deactivation of alarm reporting.	--
70022	Acknowledgment to starting the Import Password List action.	--
70023	Acknowledgment to starting the Export Password List action.	--
70024	The tag value range was exceeded during system function execution. No calculation of the system function.	Check and correct the calculation.
70025	The tag value range was exceeded during system function execution. No calculation of the system function.	Check and correct the calculation.
70026	No other screens are stored in the internal screen memory. No other screens can be selected.	--
70027	The backup of the RAM file system has been started.	--
70028	RAM file system backup is complete. The files from the RAM have been copied to the Flash memory. Following a restart, these saved files are copied back to the RAM file system.	--
70029	Backup of the RAM file system has failed. No backup copy of the RAM file system has been made.	Check the settings in the "Control Panel > OP" dialog and save the RAM file system using the "Save Files" button in the "Persistent Storage" tab.

Number	Effect/causes	Remedy
70030	The parameters configured for the system function are faulty. The connection to the new PLC was not established.	Compare the parameters configured for the system function with the parameters configured for the PLCs and correct them as necessary.
70031	The PLC configured in the system function is not an S7 PLC. The connection to the new PLC has not been established.	Compare the S7 PLC name parameter configured for the system function with the parameters configured for the PLC and correct them as necessary.
70032	The object configured with this number in the tab sequence is not available in the selected screen. The screen changes but the focus is set to the first object.	Check the number of the tab sequence and correct it if necessary.
70033	An e-mail could not be sent because a TCP/IP connection to the SMTP server no longer exists. This system event is only generated at the first failed attempt. All subsequent unsuccessful attempts to send an e-mail will no longer generate a system event. The event is only generated again if an e-mail has been successfully sent in the meantime. The central e-mail component in WinCC Runtime attempts to connect to the SMTP server at regular intervals (1 minute) in order to transmit the remaining e-mails.	Check the network connection to the SMTP server and re-establish it if necessary.
70034	Following a disruption, the TCP/IP connection to the SMTP server was successfully re-established. The queued e-mails are then sent.	--
70036	No SMTP server for sending e-mails is configured. An attempt to connect to an SMTP server therefore fails and it is not possible to send e-mails. WinCC Runtime generates the system event at the first attempt to send an e-mail.	Configure an SMTP server: In the WinCC Engineering System using "Device settings > Device settings" In the Windows CE operating system using "Control Panel > Internet Settings > E-mail > SMTP Server"
70037	An e-mail could not be sent for unknown reasons. The contents of the e-mail are discarded.	Check the e-mail parameters (recipient etc.).
70038	The SMTP server has rejected the sending or forwarding of an e-mail because the domain of the recipient is unknown to the server or because the SMTP server requires authentication. The contents of the e-mail are discarded.	Check the domain of the recipient address or disable the authentication on the SMTP server if possible. SMTP authentication is currently not used in WinCC Runtime.
70039	The syntax of the e-mail address is incorrect or contains invalid characters. The contents of the e-mail are discarded.	Check the e-mail address of the recipient.
70040	The syntax of the e-mail address is incorrect or contains illegal characters.	--
70041	The import of the user management has been aborted due to an error. Nothing has been imported.	Check your user administration or download it again to the panel.
70042	The range of values of the tag was exceeded while executing the system function. The system function was not calculated.	Check and correct the calculation.

Number	Effect/causes	Remedy
70043	The range of values of the tag was exceeded while executing the system function. The system function was not calculated.	Check and correct the calculation.
70044	An error occurred while sending the e-mails. The e-mails were not sent.	Check the SMTP settings and the error message in the system event.
70045	Cannot load a file required for encrypting the e-mail.	Update the operating system and Runtime.
70046	The server does not support encryption.	Select an SMTP server that supports encryption.
70047	The SSL versions of the HMI device and SMTP server may not be compatible.	Contact your network administrator or the operator of the SMTP server.

3.8.8.10 80000 - Log alarms (RT Advanced, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-9 80000 - Log alarms

Number	Effect/causes	Remedy
80001	The log specified is filled to the size defined (in percent) and must be stored elsewhere.	You swap out the file or table by executing a ‘move’ or ‘copy’ function.
80002	A line is missing in the specified log.	--
80003	The copying process for logging was not successful. In this case, it is advisable to check any subsequent system events, too.	--
80006	Since logging is not possible, this causes a permanent loss of the functionality.	In the case of databases, check whether the corresponding data source exists and start up the system again.
80009	A copying action has been completed successfully.	--
80010	An incorrect storage location was entered in WinCC and causes permanent loss of functionality.	Configure the storage location for the respective log again and restart the system when the full functionality is required.
80012	Log entries are stored in a buffer. If the values are read to the buffer faster than they can be physically written (using a hard disk, for example), overloading may occur and recording is then stopped.	Archive fewer values. Or: Increase the logging cycle.
80013	The overload status no longer exists. Archiving resumes the recording of all values.	--
80014	The same action was triggered twice in quick succession. Since the process is already in operation, the action is only carried out once.	--
80015	This system event is used to report DOS or database errors to the user.	--
80016	The logs are separated by system function "CloseAllLogs" and the incoming entries exceed the defined buffer size. All buffer entries are deleted.	Reconnect the logs.

Number	Effect/causes	Remedy
80017	The number of incoming entries cause a buffer overflow. This can be caused, for example, by several copying actions being activated at the same time. All copy jobs in the buffer are deleted.	Stop the copy action.
80019	The connection between WinCC and all log files was shut down, for example, after execution of the "CloseAllLogs" system function. Entries are written to the buffer and written to the logs when the connection is recovered. There is no connection to the storage location and the storage medium can be changed.	--
80020	The maximum number of simultaneously copy operations has been exceeded. Copying is not executed.	Wait until the current copying actions have been completed, then restart the last copy action.
80021	An attempt was made to delete a log which is still busy with a copy action. Deletion has not been executed.	Wait until the current copying actions have been completed, then restart the last action.
80022	An attempt was made to use system function "StartSequenceLog" to start a sequence log for a log which is not configured as sequence log. No sequence log file is created.	In the project, check • whether the "StartSequenceLog" system function was properly configured. • if the tag parameters are properly provided with data on the HMI device.
80023	An attempt was made to copy a log to itself. The log is not copied.	In the project, check • whether the "CopyLog" system function was properly configured. • if the tag parameters are properly provided with data on the HMI device.
80024	In your configuration, you specified that the "CopyLog" system function should not allow copying if the destination log already contains data ("Mode" parameter). The log is not copied.	Edit the "CopyLog" system function in your configuration, if necessary. Before you initiate the system function, delete the destination log file.
80025	You have canceled the copy operation. Data written up to this point are retained. The target log file (if configured) was not deleted. The cancellation is reported by an error entry RT_ERR at the end of the target log.	--
80026	This alarm is output after all logs are initialized. Values are written to the logs from then on. No entries are written to the logs before this time has expired, irrespective of the active state of WinCC Runtime.	--
80027	The internal Flash memory has been specified as the storage location for a log. This is not permissible. No values are written to this log and the log file is not created.	Configure "Storage Card" or network path as the storage location.
80028	The alarm serves as status report indicating that the logs are currently being initialized. No values are logged until the alarm 80026 is output.	--
80029	The number of logs specified in the alarm could not be initialized. Initialization of logs was finished. The faulty log files are not available for logging jobs.	Evaluate the additional system events related to this alarm. Check the configuration, the ODBC (Open Database Connectivity), and the specified drive.

Number	Effect/causes	Remedy
80030	The structure of the existing log file does not match the expected structure. Logging is stopped for this log.	Delete the existing log data manually, in advance.
80031	The log in CSV format is corrupted. The log cannot be used.	Delete the faulty file.
80032	Logs can be configured with events. These are triggered as soon as the log is full. Assuming WinCC Runtime is started and the log is already full, the event would never be triggered. The specified log is full and no longer accepts data.	Close WinCC Runtime, delete the log, and restart WinCC Runtime. Or: Configure a button which contains the same actions as the event and press it.
80033	"System Defined" is set in the data log file as the data source name. This causes an error. No data is written to the database logs, whereas the logging to the CSV logs works.	Reinstall SQL Sever 2005 Express.
80034	An error has occurred in the initialization of the logs. An attempt has been made to create the tables as a backup. This action was successful. A backup copy of the tables of the corrupted log file was generated and the cleared log restarted.	No action is necessary. However, it is recommended to save the backup files or delete them in order to make the space available again.
80035	An error has occurred in the initialization of the logs. An attempt has been made to create backups of the tables and this has failed. No logging or backup has been performed.	It is recommended to save the backups or to delete them in order to release memory.
80044	The export of a log was interrupted because Runtime was closed or due to a power failure. At the restart of Runtime, it was detected that the export must be resumed.	The export resumes automatically.
80045	The export of a log was interrupted due to an error in the connection to the server or at the server itself.	The export is repeated automatically. Check: • The connection to the server. • If the server is running. • If there is enough free space on the server.
80046	The destination file or the associated directory could not be created on the server.	Check whether there is enough space on the server and it you have permission to create the log file.
80047	The log could not be read while exporting it.	Check whether the storage medium is correctly inserted.
80049	The log could not be renamed while preparing to export it. The job can not be completed."	Check whether the storage medium is correctly inserted and if there is sufficient space on the medium.
80050	The log which shall be exported is not closed. The job can not be completed.	Make sure to call the "CloseAllLogs" system function before using the "ExportLog" system function. Change the configuration as required.
80051	The log to be copied contains an invalid checksum. The log was not copied.	Select a log with a valid checksum. The selected log may have been manipulated.
80052	The log cannot be read.	Check the log and the specified path.
80053	The closed log cannot be read.	Open the log.
80054	The storage medium is not available	Check whether the storage medium is correctly inserted.

3.8.8.11 90000 - FDA alarms (RT Advanced, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-10 90000 - FDA alarms

Number	Effect/causes	Solution
90024	No operator actions can be logged due to lack of space on the storage medium for log. The operator action will therefore not be executed.	Provide more storage space by inserting a blank storage medium, or backup the log files to the server using the "ExportLog" function.
90025	No user actions can be logged because of error state of the archive. Therefore the user action will not be executed.	Check whether the storage medium is correctly inserted.
90026	No operator actions can be logged because the log is closed. The operator action will therefore not be executed.	Before further operator actions are carried out, the log must be reopened with the help of system function "OpenAllLogs". Change the configuration as required.
90028	The password you entered is incorrect.	Enter the correct password.
90029	Runtime was closed during ongoing operation (perhaps due to a power failure) or a storage medium in use is incompatible with Audit Trail. An Audit Trail is not suitable if it belongs to another project or has already been logged. It could also be that archiving was not stopped before automatic execution of the "CloseAllArchives" function was started (e.g. by the task scheduler).	Ensure that you are using the correct storage medium.
90030	Runtime was closed during ongoing operation (perhaps due to a power failure).	--
90031	Runtime was closed during ongoing operation (perhaps due to a power failure).	--
90032	Running out of space on the storage medium for log.	Provide more storage space by inserting a blank storage medium, or backup the log files to the server using the "ExportLog" function.
90033	No more space on the storage medium for log. As of now, no more operator actions requiring logging will be executed.	Provide more storage space by inserting a blank storage medium, or backup the log files to the server using the "ExportLog" function.
90039	You do not have the necessary authorization to perform this action.	Adapt or upgrade your authorizations.
90040	Audit Trail is switched off because of a forced user action.	Reactivate the "Audit Trail" with the help of system function "StartLog".
90041	A user action which has to be logged has been executed without a logged on user.	A user action requiring logging should only be possible with permission. Change the configuration by setting a required authorization for the input object.
90044	A user action which has to be confirmed was blocked, because there is another user action pending.	Repeat the user action if necessary.
90048	The Audit Trail cannot be printed while data relevant to the audit is being logged.	Stop logging by calling system function "StopLogging".
90049	Access to required file is not possible.	Check the network connection or the storage medium.

Number	Effect/causes	Solution
90056	The recipe was not imported because the file contains no checksum.	Select a file with a checksum. You can also disable checksum verification by calling system function "ImportDataRecords".
90057	The recipe was not imported because the file contains an invalid checksum. The selected file may have been manipulated.	Select a file with a valid checksum.

3.8.8.12 110000 - Offline function alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-11 110000 - Offline function alarms

Number	Effect/causes	Remedy
110000	The operating mode was changed. "Offline" mode is now set.	--
110001	The operating mode was changed. "Online" mode is now set.	--
110002	The operating mode was not changed.	Check the connection to the PLCs. Check whether the address range for the “Coordination" area pointer is present in the PLC.
110003	The operating mode of the specified PLC was changed by the "SetConnectionMode" system function. The "offline" operating mode is now set.	--
110004	The operating mode of the specified PLC was changed by the "SetConnectionMode" system function. The "online" operating mode is now set.	--
110005	An attempt was made to use the "SetConnectionMode" system function to set the specified PLC to "online" mode, although the entire system is in "offline" mode. This changeover is not allowed. The PLC remains in "offline" mode.	Switch the complete system to "online" mode and repeat execution of the system function.
110006	The content of the "project ID" area pointer does not match the project ID configured in WinCC. The WinCC Runtime is therefore terminated.	Check: • the project ID entered on the PLC. • the project ID entered in WinCC.

3.8.8.13 120000 - Trend alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-12 120000 - Trend alarms

Number	Effect/causes	Remedy
120000	The trend is not displayed because you configured an incorrect axis to the trend or an incorrect trend.	Change the configuration.
120001	The trend is not displayed because you configured an incorrect axis to the trend or an incorrect trend.	Change the configuration.
120002	The trend is not displayed because the tag assigned attempts to access an invalid PLC address.	Check whether the data area for the tag is available on the PLC, whether the configured address is correct, or whether the value range for the tag is correct.

3.8.8.14 130000 - System information alarms (RT Advanced, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-13 130000 - System information alarms

Number	Effect/causes	Solution
130000	The action was not executed.	Close all other programs. Delete files no longer required from the hard disk.
130001	The action was not executed.	Delete files no longer required from the hard disk.
130002	The action was not executed.	Close all other programs. Delete files no longer required from the hard disk.
130003	No data medium found. The operation is canceled.	Check the following conditions, for example: • Access to the correct data carrier? • Is a data carrier inserted?
130004	The data carrier is write-protected. The operation is canceled.	Check whether the correct data medium is being accessed. Remove the write-protection if necessary.
130005	The file is read only. The operation is canceled.	Check whether the correct file is being accessed. Change the file attributes.
130006	No access to the file. The operation is canceled.	Check the following conditions, for example: • Is the correct file being accessed? • Does the file exist? • Is another action preventing concurrent access to the file?
130007	The network connection is interrupted. Records cannot be saved or read via the network connection.	Check the network connection and eliminate the cause of error.

Number	Effect/causes	Solution
130008	No storage card available. The specified records cannot be saved to or read from Storage Card.	Insert the storage card.
130009	The specified folder does not exist on the storage card. The files stored in this directory are not backed up after switching off the HMI device.	Insert the storage card.
130010	The maximum nesting depth can be exhausted when, for example, value changes in a script initiate the call of an infinite number of further scripts. The configured functionality is not provided.	Check the configuration.
130013	No storage card available. The specified records cannot be saved to or read from Storage Card.	Insert the storage card.

3.8.8.15 140000 - Connection alarms: Connection + device (RT Advanced, Basic Panels, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-14 140000 - Connection alarms: Connection + device

Number	Effect/causes	Remedy
140000	An online connection to the PLC is established.	--
140001	The online connection to the PLC was shut down.	--
140003	No tag update or write operations are executed.	Check whether the connection is up and the PLC is switched on. In the Control Panel, check the set parameters using the "Set PG/PC interface" function. Restart the system.
140004	No tag update or write operations are executed due to an incorrect access point, or incorrect module configuration.	Check the connection and whether the PLC is switched on. Check the access point or the module configuration (MPI, PPI, PROFIBUS) in the Control Panel with "Set PG/PC interface". Restart the system.
140005	No tag update or write operations are executed due to an incorrect HMI device address (possibly too high).	Use a different address for the HMI device. Check the connection and whether the PLC is switched on. Check the parameters set in the "Set PG/PC interface" dialog of the Control Panel. Restart the system.
140006	No tag update or write operations are executed due to an incorrect baud rate setting.	Select a different baud rate in WinCC (according to module, profile, communication peer, etc.).

Number	Effect/causes	Remedy
140007	An incorrect bus profile prevents tag updates or write operations (see %1). The following parameters cannot be written to the registry database: 1: Tslot 2: Tqui 3: Tset 4: MinTsdr 5: MaxTsdr 6: Trdy 7: Tid1 8: Tid2 9: Gap factor 10: Retry limit	Check the custom bus profile. Check the connection and whether the PLC is switched on. Check the parameters set in the "Set PG/PC interface" dialog of the Control Panel. Restart the system.
140008	An incorrect baud rate prevents tag updates or write operations. The following parameters cannot be entered in the registry database. 0: general error 1: incorrect version 2: Profile cannot be entered in the registry database. 3: Subnet type cannot be entered in the registry database. 4: Target rotation time cannot be entered in the registry database. 5: Highest address (HSA) is incorrect.	Check whether the connection is up and the PLC is switched on. In the Control Panel, check the set parameters using the "Set PG/PC interface" function. Restart the system.
140009	No tag updates or write operations because the S7 communication module was not found.	To reinstall the module, open the Control Panel and select "Set PG/PC interface".
140010	No S7 communication partner found because the PLC is shut down. DP/T: The option "PG/PC is the only master" is not set in the Control Panel under "Set PG/PC interface".	Switch on the PLC. DP/T: If only one master is connected to the network, disable "PG/PC is the only master" in "Set PG/PC interface". If several masters are connected to the network, enable these. Do not change any settings, for this will cause bus errors.
140011	No tag updates or write operations because communication is down.	Check the connection and whether the communication partner is switched on.
140012	Initialization error (e.g. if WinCC Runtime was closed in Task Manager). Or: Another application (e.g. STEP 7) with different bus parameters is active and the driver cannot be started with the new bus parameters (baud rate, for example).	Restart the HMI device. Or: Start WinCC Runtime and then start your other applications.
140013	The MPI cable is disconnected and, therefore, there is no power supply.	Check the connections.
140014	The configured bus address is in use by another application.	Change the HMI device address in the PLC configuration.
140015	Incorrect baud rate Or: Incorrect bus parameters (e.g. HSA) Or: OP address > HSA or: Incorrect interrupt vector (interrupt not registered by the driver)	Correct the parameters.

Number	Effect/causes	Remedy
140016	The hardware does not support the configured interrupt.	Change the interrupt number.
140017	The set interrupt is in use by another driver.	Change the interrupt number.
140018	SIMOTION Scout disabled the consistency check. Only a corresponding note appears.	In SIMOTION Scout, reactivate the consistency check and once again download the project to the PLC.
140019	SIMOTION Scout is downloading a new project to the PLC. Connection to the PLC is canceled.	Wait until the end of the reconfiguration.
140020	Mismatch of the PLC and project (FWX file) versions. The connection to the PLC is canceled.	The following remedies are available: Download the current version to the PLC using SIMOTION Scout. Recompile the project using WinCC ES, close WinCC Runtime, and restart with the new configuration.
140021	Setup of the connection to the PLC failed. Incorrect configuration of the "Access password" for the connection to the PLC.	Select the "Access password" area in the "Connections" editor to check the password entered for the connection to the PLC. Assign the correct password. The "Password" for the connection to the PLC is assigned in the "Security" area of the PLC properties.
140022	Setup of the connection to the PLC failed. Incorrect configuration of the access password for the connection to the PLC.	Select the "Access password" area in the "Connections" editor to check the password entered for the connection to the PLC. The "Password" for the connection to the PLC is assigned in the "Security" area of the PLC properties.
140023	Error during time synchronization: Unable to read system time of PLC %1.	Check whether the PLC is switched on. Select the "Access password" area in the "Connections" editor to check the password entered for the connection to the PLC.
140025	Setup of the connection to the PLC failed. The access password of the PLC is disabled in the PLC display.	Enable the access password in the PLC display.

3.8.8.16 160000 - Connection alarms: OPC: Connection (RT Advanced, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-15 160000 - Connection alarms: OPC: Connection

Number	Effect/causes	Solution
160000	No more data is read or written. Possible causes: • The cable connection is interrupted. • The PLC does not respond or has failed, etc. • The wrong port is used for the connection. • System overload	Check whether the cable is plugged in and the PLC is OK, and whether the correct port is used. Restart the system if the system event persists.
160001	The connection is recovered because the cause of the interruption has been eliminated.	--
160010	No connection to the server because the server ID (CLS-ID) cannot be determined. Values cannot be read or written.	Check access rights.
160011	No connection to the server because the server ID (CLS-ID) cannot be determined. Values cannot be read or written.	Check the following conditions, for example: • Correct server name? • Correct station name? • Is the server registered?
160012	No connection to the server because the server ID (CLS-ID) cannot be determined. Values cannot be read or written.	Check the following conditions, for example: • Correct server name? • Correct station name? • Is the server registered? Note for advanced users: Interpret the value from HRESULT.
160013	The specified server was started as InProc server. This has not been released and may lead to undefined behavior because the server is running in the same process area as WinCC Runtime.	Configure the server as OutProc Server or Local Server.
160014	Only one OPC server project can be started on a PC. An alarm is output after an attempt was made to start a second project. The second project has no OPC server functionality and cannot be located as an OPC server by external sources.	Do not start a second project with OPC server functionality on the computer.

3.8.8.17 170000 - S7 dialog alarms (RT Advanced, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-16 170000 - S7 dialog alarms

Number	Effect/causes	Solution
170000	S7 diagnostics events are not displayed because it is not possible to log on to the S7 diagnostics at this device. The service is not supported.	--
170001	The S7 diagnostics buffer cannot be viewed because communication with the PLC is shut down.	Set the PLC to online mode.
170002	The S7 diagnostics buffer cannot be viewed because reading of the diagnostics buffer (SSL) was canceled with an error.	--
170003	An S7 diagnostic alarm cannot be visualized. The system returns internal error %2.	--
170004	An S7 diagnostic alarm cannot be visualized. The system returns an internal error of error class %2, error number %3.	--
170007	It is not possible to read the S7 diagnostics buffer (SSL) because this operation was canceled with an internal error of class %2 and error code %3.	--

3.8.8.18 180000 - General alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-17 180000 - General alarms

Number	Effect/causes	Solution
180000	A component/OCX received configuration data with a version ID which is not supported.	Install a newer component.
180001	System overload because too many actions are running in parallel. Certain actions can be executed, while others are discarded.	Several remedies are available: • Generate alarms at a slower rate (polling). • Initiate scripts and functions at greater intervals. If the alarm appears more frequently: Restart the HMI device.
180002	The screen keyboard could not be activated. Possible causes: "TouchInputPC.exe" was not registered due to faulty Setup.	Reinstall WinCC Runtime.

Note

Alarms without additional information

Alarm 180003 is not described since the alarm text contains all the necessary information.

3.8.8.19 190000 - Tag alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Meaning of the system events

The most important system events are listed below.

Table 3-18 190000 - Tag alarms

Number	Effect/causes	Remedy
190000	It is possible that the tag is not updated.	--
190001	The tag is updated after the cause of the last error state has been eliminated (recovery of normal operation).	--
190002	The tag is not updated because communication with the PLC is down.	Select system function "SetOnline" to enable communication.
190004	The tag is not updated because the address configured for this tag does not exist.	Check the configuration.
190005	The tag is not updated because the configured PLC type does not exist for this tag.	Check the configuration.
190006	The tag is not updated because it is not possible to map the PLC type in the data type of the tag.	Check the configuration.
190007	The tag value is not modified because the connection to the PLC is interrupted or the tag is offline.	Set online mode or reconnect to the PLC.
190008	The configured tag limits were violated due to one of the following events: • Value input • System function • Script	Observe the configured or current tag limits.
190009	An attempt was made to assign this tag a value that is outside the valid range of values for this data type. For example, input of the value 260 for a byte tag, or input of the value -3 for an unsigned word tag.	Observe the range of values for the data type of the tags.
190010	The rate at which values are written to the tag is too high (e.g. initiated in a script loop). Values will be lost because buffer capacity is limited to 100 operations.	The following remedies are available: • Extend the interval between multiple write actions. • Do not use an array tag longer than 6 words when configuring an acknowledgment on the HMI device using "HMI acknowledgment tag".

Number	Effect/causes	Remedy
190011	Possible cause 1: The value entered could not be written to the configured PLC tag because the high or low limit was exceeded. The system discarded the entry and restored the original value. Possible cause 2: The connection to the PLC was interrupted.	Note that the value entered must be within the range of values of the control tag. Check the connection to the PLC.
190012	It is not possible to convert a value from a source format to a target format, for example: An attempt is being made to write a counter value that is outside the valid, PLC-specific range of values. A tag of the type Integer should be assigned a value of the type string.	Check the range of values, or the data type of the tag.
190013	You entered a string that exceeds the tag length. The string is truncated automatically to a valid length.	Always enter strings that do not exceed the valid tag length.

3.8.8.20 190100 - Area pointer alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

190100 - Area pointer alarms

The most important system events are listed below.

Number	Effect/causes	Remedy
190100	The area pointer is not updated because the address configured for this pointer does not exist. Type 1 Warnings 2 Errors 3 PLC acknowledgment 4 HMI device acknowledgment 5 LED image 6 Trend request 7 Trend transfer 1 8 Trend transfer 2 No.: Consecutive number displayed in WinCC ES.	Check the configuration.
190101	The area pointer is not updated because it is not possible to map the PLC type to the area pointer type. Parameter type and no.: see alarm 190100	--
190102	The area pointer is updated after the cause of the last error state has been eliminated (recovery of normal operation). Parameter type and no.: See alarm 190100.	--

3.8.8.21 200000 - PLC coordination alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

200000 - PLC coordination alarms

The most important system events are listed below.

Number	Effect/causes	Remedy
200000	Coordination is not executed because the address configured in the PLC does not exist/is not set.	Change the address or set up the address in the PLC.
200001	Coordination is canceled because the write access to the address configured in the PLC is not possible.	Change the address or set the address in the PLC at an area which allows write access.
200002	Coordination is not carried out at the moment because the address format of the area pointer does not match the internal storage format.	Internal error
200003	Coordination can be executed again because the last error state is eliminated (return to normal operation).	--
200004	The coordination may not be executed.	--
200005	No more data is read or written. Possible causes: • The cable connection is interrupted. • The PLC does not respond or has failed, etc. • System overload	Ensure that the cable is plugged in and the PLC is operational. Restart the system if the system event persists.

3.8.8.22 200100 - Area pointer alarms project ID (RT Advanced, Basic Panels, Panels, Comfort Panels)

200100 - Area pointer alarms project ID

The most important system events are listed below.

Number	Effect/causes	Remedy
200101	The project ID of the area pointer cannot be described. No entry possible.	Check the configuration.
200102	Error in the type conversion.	Check the type conversion.
200103	The project ID of the area pointer is updated as the cause of the last error state has been eliminated (recovery of normal operation).	--

Number	Effect/causes	Remedy
200104	Unknown error	Check the • Configuration • HMI tags • HMI alarms • System alarms • Controller alarms If the error occurs repeatedly, contact the hotline!
200105	The online connection to the PLC was interrupted. The connection is interrupted.	Check the following conditions, for example • Are the access correct? • Correct server name? • Correct station name? • Is the server registered? • Is the configuration correct?

3.8.8.23 210000 - PLC job alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

210000 - PLC job alarms

The most important system events are listed below.

Number	Effect/causes	Remedy
210000	Jobs are not processed because the configured address does not exist/has not been set up in the PLC.	Change the address or set up the address in the PLC.
210001	Jobs are not processed because read/write access to the configured address is not possible in the PLC.	Change the address, or set up the address in a PLC area at which read/write access is possible.
210002	Jobs are not executed because the address format of the area pointer does not match the internal storage format.	Internal error
210003	The job buffer is processed again because the last error status has been eliminated (recovery of normal operation).	--
210004	The job buffer is possibly not going to be processed.	--
210005	A job mailbox with invalid number was initiated.	Check the PLC program.
210006	An error occurred while executing the job mailbox. As a result, the control job is not executed. Observe the next/previous system event.	Check the parameters of the control job. Recompile the configuration data.

3.8.8.24 220000 - WinCC communication driver alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

220000 - WinCC communication driver alarms

The most important system events are listed below.

Number	Effect/causes	Remedy
220001	The tag is not downloaded because write access to data type Bool/Bit is not supported by the sublevel communication driver/HMI device.	Change the configuration.
220002	The tag is not downloaded because write access to data type Byte is not supported by the sublevel communication driver/HMI device.	Change the configuration.
220003	The communications driver could not be loaded. The driver might not be installed.	Install the driver by reinstalling WinCC Runtime.
220004	Communication is down and no update data is transferred because the cable is not connected or defective etc.	Check the connection.
220005	Communication is up.	--
220006	The connection between the specified PLC and the specified port is active.	--
220007	The connection to the specified PLC is interrupted at the specified port.	Check the following: • Is the cable plugged in? • Is the PLC OK? • Is the right port being used? • Is your configuration OK (port parameters, protocol settings, PLC address)? Restart the system if the system event persists.
220008	The communication driver cannot access or open the specified port. This port might be in use by another application, or a port that does not exist on the target device is being used. No communication with the PLC.	Close all applications that access the port and restart the computer. Use another port that exists in the system.
220009	The communication path is either empty, incomplete or has an invalid format.	Enter the communication path in the correct format.

3.8.8.25 230000 - Screen object alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

230000 - Screen object alarms

The most important system events are listed below.

Number	Effect/causes	Solution
230000	The value entered could not be used. The system discards this entry and restores the previous value. Possible causes: • The range of values is exceeded. • You entered invalid characters • The valid maximum number of users has been exceeded.	Enter a practical value, or delete a user that is no longer required.
230002	The user currently logged on does not have the necessary authorization, so the system discards the entry and restores the previous value.	Log on as user with appropriate authorization.
230003	Change to the specified screen failed because this screen is not available/configured. The current screen remains selected.	Configure the screen and check the selection function.
230005	The range of values of the tag has been exceeded in the I/O field. The original tag value is retained.	Observe the range of values for the tag when entering a value.
230100	During navigation in the Web browser, the system returned a message which may be of interest to the user. The Web browser continues to run but may not (fully) show the new page.	Navigate to another page.
230200	The HTTP channel connection was interrupted due to an error. This error is explained in detail by another system event. Data is no longer exchanged.	Check the network connection. Check the server configuration.
230201	The HTTP channel connection is set up. Data is exchanged.	--

Number	Effect/causes	Solution
230202	WININET.DLL has detected an error. Usually, this error occurs if it is not possible to connect to the server, or the server denies a connection because the client lacks proper authorization. An unknown server certificate may also be the cause if the connection is encrypted by means of SSL. The alarm text provides details. This text is always in the language of the Windows installation because it is returned by the Windows OS. Process values are no longer exchanged. The part of the alarm returned by the Windows OS might not be displayed, e.g. "An error has occurred". WININET.DLL returns the following error: Number: 12055 Text:HTTP: <no error text available>."	Depending on the cause: When an attempt to connect fails, or a timeout occurs: • Check the network connection and the network. • Check the server address. • Check whether the WebServer is actually running on the target station. Incorrect authorization: • The configured user name and/or password do not match the entries on the server. Set consistent data. If the server certificate is rejected: Certificate signed by an unknown CA ( ): • Ignore this point, or install a certificate that has been signed with one of the root certificates known to the client station. The date of the certificate is invalid: • Ignore this point, or install a certificate with valid date on the server. Invalid CN (Common Name or Computer Name): • Ignore this point, or install a certificate with a name that corresponds to the server address.
230203	Although a connection can be made to the server, the HTTP server refused to connect. Possible causes: • WinCC Runtime does not run on the server • The HTTP channel is not supported (503 Service unavailable). Other errors can only occur if the Webserver does not support the HTTP channel. The language of the alarm text depends on the Webserver. Data is not exchanged.	On 503 Service unavailable error: Check whether WinCC Runtime is running on the server and whether the HTTP channel is supported.
230301	Internal error. An English text explains the error in more detail. The error may be caused by insufficient memory. OCX does not work.	--
230302	The name of the remote server cannot be resolved. An attempt to connect has failed.	Check the configured server address. Check whether the DNS service is available on the network.
230303	The remote server is not running on the addressed computer. incorrect server address. An attempt to connect has failed.	Check the configured server address. Check whether the remote server is running on the target computer.
230304	The remote server on the addressed computer is incompatible with VNCOCX. An attempt to connect failed.	Use a compatible remote server.

Number	Effect/causes	Solution
230305	Authentication has failed due to incorrect password. An attempt to connect failed.	Configure the correct password.
230306	Error in the connection to the remote server. This may occur as a result of network problems. An attempt to connect failed.	Check whether the network cable is plugged in, or whether there are network problems.
230307	The connection to the remote server was shut down. Possible causes: • The remote server was shut down • The user instructed the server to close all connections. The connection is cancelled.	--
230308	This alarm provides information on the connection status. An attempt is made to connect.	--

Note

Alarms without additional information

Alarms 230400-230409 and 230500-230501 are not described since the alarm text contains all the necessary information.

3.8.8.26 240000 - Authorization alarms (RT Advanced, Panels, Comfort Panels)

240000 - Authorization alarms

The most important system events are listed below.

Number	Effect/causes	Solution
240000	WinCC Runtime is running in demo mode. You have no authorization, or your authorization is corrupted.	Install the authorization.
240001	WinCC Runtime is running in demo mode. You configured too many tags for the installed version.	Load an adequate authorization / powerpack.
240002	WinCC Runtime is running with time-limited emergency authorization.	Restore the full authorization.

Number	Effect/causes	Solution
240004	Error when reading the emergency authorization. WinCC Runtime is running in demo mode.	Restart WinCC Runtime. Install or repair the authorization (see Commissioning Instructions for Software Protection).
240005	Automation License Manager has detected an internal system error. Possible causes: • Corrupted file • Faulty installation • No free memory space for Automation License Manager, or similar.	Restart the HMI device/PC. If this procedure does not solve the problem, remove and then reinstall Automation License Manager.

3.8.8.27 250000 - S7 Status/Force alarms (RT Advanced, Panels, Comfort Panels)

250000 - S7 Status/Force alarms

The most important system events are listed below.

Number	Effect/causes	Solution
250000	The tag in the specified line in "Status Force" is not updated because the address configured for this tag is not available.	Check the set address and then verify that the address is set up in the PLC.
250001	The tag in the specified line in "Status Force" is not updated because the PLC type configured for this tag does not exist.	Check the set address.
250002	The tag in the specified line in "Status Force" is not updated because it is not possible to map the PLC type in the tag type.	Check the set address.
250003	An attempt to connect to the PLC failed. The tags are not updated.	Check the connection to the PLC. Check that the PLC is switched on and is online.

3.8.8.28 260000 - Password system alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

260000 - Password system alarms

The most important system events are listed below.

Number	Effect/causes	Solution
260000	An unknown user or an unknown password has been entered in the system. The current user is logged off from the system.	Log on to the system as a user with a valid password.
260001	The logged in user does not have sufficient authorization to execute the protected functions on the system.	Log on to the system as a user with sufficient authorization.
260002	This alarm output when the "TrackUserChange" system function is triggered.	--
260003	The user has logged off from the system.	--

Number	Effect/causes	Solution
260004	The user name entered into the user view already exists in the user management.	Select another user name because user names have to be unique in the user management.
260005	The entry is discarded.	Enter a shorter user name.
260006	The entry is discarded.	Use a shorter or longer password.
260007	The logoff time entered is outside the valid range from 0 to 60 minutes. The new value entered is discarded and the original value is retained.	Enter a logon timeout value between 0 and 60 minutes.
260008	An attempt was made in WinCC to read a PTProRun.pwl file created with ProTool V 6.0. Reading of the file was canceled due to incompatibility of the format.	--
260009	You have attempted to delete the user "Administrator" or "PLC User". These users are fixed components of the user management and cannot be deleted.	If you need to delete a user, because perhaps you have exceeded the maximum number permitted, delete another user.
260012	The password entries in the "Change Password" dialog and in the confirmation field do not match. The password is not changed. User will be logged off.	You have to log on to the system again. Then enter the identical password twice to be able to change the password.
260013	The password entered in the "Change Password" dialog is invalid because it is already in use. The password is not changed. User will be logged off.	You have to log on to the system again. Then enter a new password that has not been used before.
260014	You have tried to log on with an incorrect password three times in a row. You will be locked out and assigned to group no. 0.	You can log on to the system with your correct password. Only an administrator can change the assignment to a group.
260024	The password you entered does not meet the necessary security guidelines.	Enter a password that contains at least one number.
260025	The password you entered does not meet the necessary security guidelines.	Enter a password that comprises at least three characters.
260026	The password you entered does not meet the necessary security guidelines.	Enter a password that contains at least one special character.
260028	Upon system start-up, an attempt to log on, or when trying to change the password of a SIMATIC log-on user, the system attempts to access the SIMATIC Logon Server. If attempting to log on, the new user is not logged in. If a different user was logged on before, then this user is logged off.	Check the connection to the SIMATIC Logon Server and its configuration; for example: 1. Port number 2. IP address 3. Server name 4. Functional transfer cable. Or use a local user.
260030	The SIMATIC Logon user could not change his password on the SIMATIC Logon Server. The new password is possibly noncompliant with password rules set on the server, or the user is not authorized to change his password. The old password remains and the user is logged off.	Log in again and choose a different password. Check the password rules on the SIMATIC Logon Server.

Number	Effect/causes	Solution
260033	The action change password or log on user could not be carried out.	Check the connection to the SIMATIC Logon Server and its configuration; for example: 1. Port number 2. IP address 3. Server name 4. Functional transfer cable Or use a local user.
260034	The last logon operation has not yet ended. A user action or a logon dialog can therefore not be called. The logon dialog is not opened. The user action is not executed.	Wait until the logon operation is complete.
260035	The last attempt to change the password was not completed. A user action or a logon dialog can therefore not be called. The logon dialog is not opened. The user action is not executed.	Wait until the procedure is complete.
260036	There are insufficient licenses on the SIMATIC Logon Sever. The logon is not authorized.	Check the licensing on the SIMATIC Logon Server.
260037	There is no license on the SIMATIC Logon Sever. A logon is not possible. It is not possible to log on via the SIMATIC Logon Server, only via a local user.	Check the licensing on the SIMATIC Logon Server.
260040	The system attempts to access the SIMATIC Logon Server upon system start-up or when trying to change the password. If attempting to log on, the new user is not logged in. If a different user was logged on before, then this user is logged off.	Check connection to the domain and its configuration in the Runtime security settings editor. Or use a local user.
260043	It was not possible to log the user on to the SIMATIC Logon Server. The user name or the password could be incorrect or the user does not have sufficient rights to log on. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Try again. If necessary, check the password data on the SIMATIC Logon Server.
260044	It was not possible to log the user on to the SIMATIC Logon Server as his account is blocked. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Check the user data on the SIMATIC Logon Server.
260045	The SIMATIC Logon user is not associated to any or several groups. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Check user data on the SIMATIC Logon Server and the configuration in your WinCC project. A user may only be assigned to one group.

Note

Alarms without additional information

Alarms 260010, 260011, 260015-260023, 260029, 260031, 260032, 260038, 260039, 260041, 260042 and 260048 are not described since the alarm text contains all the necessary information.

3.8.8.29 270000 - System alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

270000 - System Alarms

The most important system events are listed below.

Number	Effect/causes	Remedy
270000	The alarm does not indicate the tag because it is accessing an invalid address in the PLC.	Check whether the data area for the tag exists on the PLC, whether the configured address is correct, and whether the value range for the tag is correct.
270001	There is a device-specific limit as to how many alarms may be queued for viewing (see the operating instructions). This limit has been exceeded. The view no longer contains all the alarms. However, all alarms are written to the alarm buffer.	--
270002	The view shows alarms of a log for which there is no data in the current project. Placeholders are output for the alarms.	Delete old log data, if necessary.
270003	The service cannot be set up because too many devices want to access this service. A maximum of four devices can execute this action.	Reduce the number of HMI devices which want to use the service.
270004	Access to the persistent alarm buffer is not possible. Alarms cannot be restored or backed up.	If the problems persist at the next restart, contact Customer Support (delete Flash).
270005	Persistent alarm buffer corrupted: Alarms cannot be restored.	If the problems persist at the next restart, contact Customer Support (delete Flash).
270006	Project modified: Alarms cannot be restored from the persistent alarm buffer.	The project was compiled and downloaded again to the HMI device. The error should no longer occur at the next restart of the HMI device.
270007	A configuration problem is preventing you from the restoring the data (e.g. a DLL was deleted, or unknown directory).	Update the operating system and download your project again to the HMI device.

3.8.8.30 280000 - Alarms SIMATIC 505 DP Communication Driver (CSP)

Alarms SIMATIC 505 DP Communication Driver (CSP)

The most important system events are listed below.

Number	Effect/causes	Solution
280000	The connection between the controller and port is active.	--
280001	The connection to the PLC at the port is interrupted.	Check the following: • Is the cable plugged in? • Is the controller active? • Is the correct port being used? • Is the configuration OK?
280002	The parameters of the controller are correct, the XSUB and .REC files are available.	--
280003	• The parameters of the controller are correct, the XSUB and .REC files are not available. • The controller is in STOP mode.	• Download the XSUB and .REC files to the controller. • Verify that the controller is in STOP mode.
280004	• The communication cable between HMI device and controller is disconnected. • The controller is in STOP mode.	Check the following: • The communication cable between operating unit and controller • Mode of the controller (Run/STOP)

3.8.8.31 290000 - Recipe system alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

290000 - Recipe system alarms

The most important system events are listed below.

Number	Effect/causes	Solution
290000	The recipe tag could not be read or written. It is assigned the start value. The alarm can be entered in the alarm buffer for up to four more faulty tags. After that, alarm 290003 is output.	Check the configuration to see whether the address has been set up in the PLC.
290001	An attempt was made to assign the recipe tag a value that is outside the valid range of values for this type. The alarm can be entered in the alarm buffer for up to four more faulty tags if necessary. After that, alarm 290004 is output.	Observe the range of values for the tag type.

Number	Effect/causes	Solution
290002	It is not possible to convert a value from a source format to a target format. The alarm can be entered in the alarm buffer for up to four more faulty recipe tags if necessary. After that, alarm 290005 is output.	Check the range of values or type of the tag.
290003	This alarm is output after alarm 290000 was triggered more than five times. In this case, no further separate alarms are generated.	Check the configuration to see whether the tag addresses have been set up in the PLC.
290004	This alarm is output after alarm 290001 was triggered more than five times. In this case, no further separate alarms are generated.	Observe the range of values for the tag type.
290005	This alarm is output after alarm 290002 was triggered more than five times. In this case, no further separate alarms are generated.	Check the range of values or type of the tag.
290006	The limits configured for the tag have been exceeded by the values entered.	Observe the configured or current tag limits.
290007	There is a difference between the source and target structure in the recipe currently being processed. The source structure contains an additional recipe tag that is not available in the target structure and cannot be assigned. The value is discarded.	Remove the specified recipe tag in the specified recipe from your configuration.
290008	There is a difference between the source and target structure in the recipe currently being processed. The target structure contains an additional recipe tag that is not available in the source structure. The recipe tag specified is assigned its start value.	Insert the specified recipe tag into the source structure.
290010	The storage location configured for the recipe is invalid. Possible causes: Invalid characters, write protection, data carrier out of space or not available.	Check the configured storage location.
290011	A data record of the specified number does not exist.	Check the source for the number (constant or tag value).
290012	A recipe of the specified number does not exist.	Check the source for the number (constant or tag value).
290013	An attempt was made to save a data record under a data record number that already exists. The operation is not executed.	The following remedies are available: • Check the source for the number (constant or tag value). • First, delete the data record. • Modify the "Overwrite" function parameter.
290014	The specified import file was not found.	Check the following: • The file name • Ensure that the file is in the specified directory.

Number	Effect/causes	Solution
290020	Check back to verify that the download of data records from the HMI device to the PLC has started.	--
290021	Check back to verify that the download of records from the HMI device to the PLC was completed.	--
290022	Check back to indicate that the download of data records from the HMI device to the PLC was canceled due to an error.	Check the following conditions in the configuration: • Are the tag addresses configured in the PLC? • Does the recipe number exist? • Does the data record number exist? • Is the "Overwrite" function parameter set?
290023	Check back to verify that the download of data records from the PLC to the HMI device has started.	--
290024	Check back to verify that the download of data records from the PLC to the HMI device was completed.	---
290025	Check back to indicate that the download of data records from the PLC to the HMI device was canceled due to an error.	Check the following conditions in the configuration: • Are the tag addresses configured in the PLC? • Does the recipe number exist? • Does the data record number exist? • Is the "Overwrite" function parameter set?
290026	An attempt was made to read/write a data record that is not free at present. This error can occur if recipes were configured for download with synchronization.	Set the mailbox status to zero.
290027	Unable to connect to the PLC at present. As a result, the data record cannot be read or written. Possible causes: No hardware connection to the PLC (no cable plugged in, cable is defect), or the PLC is switched off.	Check the connection to the PLC.
290030	This alarm is output after you selected screen which contains a recipe view in which a data record is already selected.	Reload the data record from the storage location, or retain the current values.
290031	While saving, it was detected that a data record with the specified number already exists.	Overwrite the data record, or cancel the action.
290032	During the export of data records, a file with the specified name was found.	Overwrite the file, or cancel the process.
290033	Confirmation prompt before deleting data records.	--
290040	A data record error with error code %1 that cannot be described in more detail occurred. The action is canceled. It is possible that the mailbox was not installed correctly on the PLC.	Check the storage location, the data record, the "Data record" area pointer, and the connection to the PLC. Restart the action after a short waiting time. If the error persists, contact Customer Support. Forward the relevant error code to Customer Support.
290041	A data record or file cannot be saved because the storage location is out of sufficient space.	Delete files no longer required.

Number	Effect/causes	Solution
290042	An attempt was made to execute several recipe actions simultaneously. The last action is not executed.	Retrigger the action after a short waiting time.
290043	Confirmation prompt before saving data records.	--
290044	The database for the recipe was corrupted and will be deleted.	--
290050	A check back indicates that the export of data records was started.	--
290051	A check back indicates successful completion of the export of data records.	--
290052	A check back indicates that the export of data records was canceled due to an error.	Ensure that the structure of the data records at the storage location and the current recipe structure on the HMI device are identical.
290053	A check back indicates that the import of records was started.	--
290054	A check back indicates successful completion of the import of data records.	--
290055	A check back indicates that the import of data records was canceled due to an error.	Ensure that the structure of the data records at the storage location and the current recipe structure on the HMI device are identical.
290056	Error when reading/writing the value in the specified row/column. The action was canceled.	Check the specified row/column.
290057	The tags of the recipe specified were toggled from "offline" to "online" mode. Each change of a tag in this recipe is now immediately transferred to the PLC.	--
290058	The tags of the specified recipe were toggled from "online" to "offline" mode. Modifications to tags in this recipe are no longer immediately transferred to the PLC but must be transferred there explicitly by downloading a data record.	--
290059	A check back indicates that the specified record was successfully saved.	--
290060	A check back indicates that the specified data record memory was cleared.	--
290061	A check back indicates that deletion of the data record memory was canceled due to an error.	--
290062	The data record number exceeds the maximum of 65536. This data record cannot be created.	Select another number.
290063	This occurs when you execute system function "ExportDataRecords" while the "Overwrite" parameter is set to "No". An attempt was made to save a recipe under a file name that already exists. The export is canceled.	Check the parameters of the "ExportDataRecords" system function.
290064	A check back indicates that the deletion of data records was started.	--

Number	Effect/causes	Solution
290065	A check back indicates successful completion of the deletion of data records.	--
290066	Confirmation prompt before deleting data records.	--
290068	Confirmation prompt for deletion of the selected recipe data records.	--
290069	Confirmation prompt for deletion of all recipe data records.	--
290070	The data record specified was not found in the import file.	Check the source of the data record number, or the data record name (constant, or tag value).
290071	When the editing data record values, you entered a value that is below the low limit of the recipe tag. The entry is discarded.	Enter a value within the recipe tag limits.
290072	When editing data record values, you entered a value that exceeds the high limit of the recipe tag. The entry is discarded.	Enter a value within the recipe tag limits.
290073	An action (e.g. saving a record) failed for an unknown reason. The error corresponds to status alarm IDS_OUT_CMD_EXE_ERR in the large recipe view.	--
290074	While saving, a data record with the specified number but with different name was found.	Overwrite the record, change the record number or cancel the action.
290075	A data record of this name already exists. The data record is not saved.	Select a different data record name.
290110	The default values could not be set due to an error.	--
290111	The recipes subsystem cannot be used. Recipe views have no content and recipe-specific functions will not be executed. Possible causes: • Error when loading the recipes. • The recipe structure was changed in the ES. The recipes were not included in the latest project download. This means that the new configuration data no longer matches the old recipes on the device.	Download the project to the device again, including the recipes (the corresponding check box in the download dialog must be check marked).

Note

Alarms without additional information

Alarms 290067 and 290076-290109 are not described since the alarm text contains all the necessary information.

3.8.8.32 300000 - Alarm_S alarms (RT Advanced, Panels, Comfort Panels)

300000 - Alarm_S alarms

The most important system events are listed below.

Number	Effect/causes	Solution
300000	Faulty configuration of process monitoring (e.g. using PDiag or S7-Graph): More alarms are queued than specified in the specifications of the CPU. No further ALARM_S alarms can be managed by the PLC and reported to the HMI devices.	Change the PLC configuration.
300001	ALARM_S is not registered on this PLC.	Select a controller that supports the ALARM_S service.

Note

Alarms without additional information

Alarms 300100-300102 are not described since the alarm text contains all the necessary information.

3.8.8.33 310000 - Reporting alarms (RT Advanced, Panels, Comfort Panels)

310000 - Reporting alarms

The most important system events are listed below.

Number	Effect/causes	Solution
310000	An attempt is being made to print too many reports in parallel. Only one log file can be output to the printer at a given time; the print job is therefore rejected.	Wait until the previous active log was printed. Repeat the print job if necessary.
310001	An error occurred on triggering the printer. The report is either not printed or printed with errors.	Evaluate the additional system events related to this alarm. Repeat the print job if necessary.

3.8.8.34 330000 - Interaction alarms (RT Advanced, Panels, Comfort Panels)

330000 - Interaction alarms

The most important system events are listed below.

Number	Effect/causes	Solution
330022	Too many open dialogs on the HMI device.	Close all dialogs you do not require on the HMI device.

Note

Alarms without additional information

Alarms 330001-330021, 330023-330070 are not described since the alarm text contains all the necessary information.

3.8.8.35 340000 - Alarms info text

340000 - Alarms info text

Note

Alarms without additional information

Alarms 340001-340005 are not described since the alarm text contains all the necessary information.

3.8.8.36 350000 - PROFIsafe alarms (RT Advanced, Panels, Comfort Panels)

350000 - PROFIsafe alarms

The most important system events are listed below.

Number	Effect/causes	Solution
350000	PROFIsafe packets were not received within the specified time. There is a communication problem with the F-CPU. RT is terminated.	Check the WLAN connection.
350001	PROFIsafe packets were not received within the specified time. There is a communication problem with the F-CPU. The PROFIsafe connection is set up again.	Check the WLAN connection.
350002	Internal error. Runtime is terminated.	Internal error
350003	Check back for indicating the connection setup to the F-CPU. The Emergency-Off buttons are active immediately.	--
350004	PROFIsafe communication was set up and the connection was shut down. Runtime can be terminated. The Emergency-Off buttons are deactivated immediately.	--

Number	Effect/causes	Solution
350005	Incorrect address configured for the F-Device. A PROFIsafe connection cannot be set up.	Check and change the address of the F device in the WinCC ES.
350006	The project was started. At the start of the project, check the proper function of the ACK buttons.	Press the two ACK buttons successively in the "Acknowledge" and "Panic" positions.
350008	You configured an incorrect number of failsafe buttons. A PROFIsafe connection cannot be set up.	Modify the number of failsafe buttons in the project.
350009	The device is in Override mode. It may no longer be possible to detect the location because transponder detection fails.	Exit the override mode.
350010	Internal error: The device has no failsafe buttons.	Return the device to Siemens. Worldwide contact partners

3.8.8.37 380000 - Alarms Sm@rtServer (RT Professional, RT Advanced, Panels, Comfort Panels)

380000 - Alarms Sm@rtServer

Note

Alarms without additional information

Alarms 380000-380100 are not described since the alarm text contains all the necessary information.

3.8.8.38 390000 alarms password encryption and certificates

390000 alarms password encryption and certificates

Number	Effect/causes	Remedy
390000	The Runtime project was downloaded by a different Windows user than the user that created the certificate for the password encryption.	A Runtime project with encrypted communication connections to S7 controllers must be downloaded for each Windows user that wants to use these connections in Runtime.

Note

Alarms without additional information

Alarms 390001-390003 are not described because the alarm text contains all necessary information.

3.8.9 System events (RT Professional)

3.8.9.1 System events (RT Professional)

Introduction

The system events generated by the various WinCC modules are described below. You can integrate these events into your alarm system using the "WinCC - System Events" menu command.

Note

You can obtain additional information on the system events in the "Comment" block of the active alarm.

Note

System events are output in an alarm view. System events are output in the language currently set on your HMI device.

The time format (AM/PM or 24-hour format) is based on the selected language. If no translation of the alarm texts exists in this language, English is used as replacement and the corresponding time format is displayed.

3.8.9.2 1000000 - Alarms Runtime

Meaning of the system events

The most important system alarms are listed below.

Table 3-19 1000000 - Alarms WinCC Runtime

Number	Effect/causes	Solution
1000000	Error	
1000001	Error while loading the object engine	
1000002	Channel could not be loaded	
1000003	Tag - Low limit violation	
1000004	Tag - High limit violation	
1000005	Error formatting a tag	
1000006	Error scaling a tag	
1000100	Driver error	
1000200	Status	
1000201	Object engine was loaded	
1000202	Runtime was activated	
1000203	Runtime was deactivated	
1000204	Connection not established	

Number	Effect/causes	Solution
1000205	Connection established	
1000206	Client connection established	
1000207	Client connection terminated	
1000208	Client connection terminated	
1000209	Connection deleted	
1000210	Connection changed	
1000211	Connection reestablished	
1000300	Driver status	
1000301	The legitimization of the connection @1%s@ has failed. The password is incorrect.	
1000302	The PLC is protected. A password must be configured for the connection @1%s@.	
1000303	The legitimization of the connection @1%s@ has failed. The password is locked. Unlock locally (e.g. on the display).	

3.8.9.3 1000800 - Alarms WinCC client

Meaning of the system events

The most important system alarms are listed below.

Table 3-20 1000800 - Alarms WinCC client

Number	Effect/causes	Solution
1000800	Import	
1000801	Export	
1000802	Delete	
1000803	New	
1000804	Reload	
1000805	Default server	
1000806	Implicit update	
1000807	Update	

3.8.9.4 1000900 - Alarms Performance Monitor

Meaning of the system events

The most important system alarms are listed below.

Table 3-21 1000900 - Alarms Performance Monitor

Number	Effect/causes	Solution
1000900	@1%s@:@7%s@ is low	
1000901	@1%s@:@7%s@ is OK	
1000902	@1%s@:@7%s@ is high	
1000903	@1%s@:@7%s@ is OK	
1000904	@1%s@:@7%s@ is low	
1000905	@1%s@:@7%s@ is OK	
1000906	@1%s@:@7%s@ is high	
1000907	@1%s@:@7%s@ is OK	
1000908	@1%s@:@7%s@ is low	
1000909	@1%s@:@7%s@ is OK	
1000910	@1%s@:@7%s@ is high	
1000911	@1%s@:@7%s@ is OK	
1000912	@1%s@:Redundancy loss of terminal adapter @2%s@	
1000913	@1%s@:Redundancy of terminal adapter @2%s@ restored	
1000914	@1%s@:Terminal adapter @2%s@ connected	
1000915	@1%s@:Terminal adapter @2%s@ disconnected	

3.8.9.5 1001000 - Alarms Screens in Runtime

Meaning of the system events

The most important system alarms are listed below.

Table 3-22 1001000 - Alarms Screens in Runtime

Number	Effect/causes	Solution
1001000	General error	
1001001	Operator action not active	
1001002	Wrong picture format	
1001003	Picture not found	
1001004	No dynamic active in picture	

Number	Effect/causes	Solution
1001005	Tag could not be written	
1001006	Dynamic actions in picture are not active	

3.8.9.6 1002000 - Alarms Tag logging

Meaning of the system events

The most important system alarms are listed below.

Table 3-23 1002000 - Alarms Tag logging

Number	Effect/causes	Solution
1002000	General error	
1002001	Error during initialization	
1002002	Error loading the runtime data	
1002003	Error changing languages	
1002004	Error accessing the database	
1002005	Error creating runtime objects	
1002006	Error in online configuration	
1002007	Error in client/server environment	
1002008	Error in memory management	
1002009	Error compiling measured values	
1002010	Error processing measured values	
1002011	Error logging measured values	
1002012	Error in Normalization DLL	
1002013	Error in user archive option	
1002014	Error in process-controlled logging	
1002015	Error in API	
1002016	Error in application window	
1002017	System fault	
1002018	Error overflow in database queue	
1002019	Error overflow in notification queue	
1002020	Error overflow of the Normalization DLL queue - loss of data!	
1002021	Problems with the connection to the central archive server.	
1002022	The problem in the connection to the central archive server has been eliminated.	
1002023	Archive value edited - Archive tag=@10%s@ Time stamp=@1%s@ New value=@2%s@ Old value=@3%s@	
1002024	Archive value created - Archive tag=@10%s@ Time stamp=@1%s@ Value=@2%s@	

3.8.9.7 1003018 - Alarms Alarm system

Meaning of the system events

The most important system alarms are listed below.

Table 3-24 1003018 - Alarms Alarm system

Number	Effect/causes	Solution
1003018	Alarm does not require acknowledgment An attempt was made to acknowledge a alarm that does not require acknowledgment.	
1003019	Alarm has already been acknowledged An attempt was made to acknowledge a alarm that has already been acknowledged.	
1003020	Alarm class for the alarm not found The alarm class associated with the alarm does not exist.	
1003021	Alarm status cannot be processed The alarm status (Incoming, Outgoing, Acknowledged, ...) could not be interpreted.	
1003022	Alarm is locked This alarm occurs if you attempt to trigger a locked alarm via the API.	
1003023	Alarm date/time stamp invalid The date/time stamp from the AS could not be interpreted.	
1003032	Alarm window template name unknown This error occurs when an alarm window that is already configured in the "Screens" editor is subsequently deleted. In the case of client/server projects, this alarm could also be triggered by a communication error in the network.	
1003033	Alarm window could not be created	
1003034	Alarm window - data invalid Incorrect project data.	
1003048	Bit is outside the tag range For example, an attempt was made to trigger the 18th bit of a 16-bit tag.	
1003049	No tag change	
1003050	Event tag (bit) already occupied by an alarm The bit of the alarm tag has already been interconnected with an alarm.	
1003051	Acknowledgment tag (bit) already occupied by an alarm The acknowledgment tag bit has already been interconnected with an alarm.	
1003052	Status tag already occupied by an alarm The bit of the status tag has already been interconnected with an alarm.	
1003053	Error during conversion of Variant data type	

Number	Effect/causes	Solution
1003054	Event tag (bit) already occupied-->different type	
1003055	Event tag or tag type invalid The data type of the alarm tag is invalid. You may have used a tag with sign.	
1003056	Acknowledgment tag or tag type invalid	
1003057	Status tag or tag type invalid The data type of the status tag is invalid. You may have used a tag with sign.	
1003058	Handle invalid This error message can occur during access via the API.	
1003059	Selection criteria invalid This error message can occur during access via the API.	
1003060	Resources DLL for language not found One of the language-dependent files is missing.	
1003061	Error while creating the memory mapped file This internal error indicates a problem involving the memory.	
1003062	Error while creating the synchronization mechanism This error occurs when the operating system is overloaded.	
1003063	Incorrect parameter The error message can occur during access via the API.	
1003064	Transfer buffer too small The error message can occur during access via the API.	
1003065	Function is not available at this time The error message can occur during access via the API.	
1003066	Format DLL send data cannot be evaluated	
1003067	Format DLL Do not execute function	
1003068	No alarm text blocks in the report	Check the alarm sequence report.
1003069	Invalid protocol identifier	
1003070	Report printout is already active An attempt was made to restart an active report.	
1003071	Alarm system runtime <server> not initialized The server reports that the project was activated without the runtime component of Alarm Logging.	
1003072	Printout of the alarm sequence report has not been started.	
1003073	Printout of the short-term archive report has not been started.	
1003074	Printout of the sequence archive report has not been started.	
1003075	Maximum number of alarms that can be configured online The maximum number of alarms that can be configured online (default setting is 600) has been exceeded.	
1003076	Parameter error in S7 frame A parameter error has occurred on the interface to S7.	
1003077	Parameter error for ALGRT A parameter error has occurred on the interface to ALGRT.	

Number	Effect/causes	Solution
1003078	Parameter error for TLGRT A parameter error has occurred on the interface to TLGRT.	
1003079	Parameter error in additional data The additional data of the alarms are faulty.	
1003080	Parameter error in AR_SEND structure The structure of the AR-SEND user data is faulty.	
1003081	General error An internal error of unknown cause has occurred.	
1003082	Alarm loss on the automation system	
1003083	Connecting and updating the automation system is underway	
1003084	Connecting and updating the automation system is finished	
1003085	System modification in RUN (CiR) active	
1003086	System modification in RUN (CiR) is finished	
1003087	Error while logging on to receive alarm.	
1003098	Alarm logging overflow - alarms are lost	
1003099	[Computer name]:Alarm [alarm number] locked [alarm text of locked alarm] This alarm occurs when a alarm is locked.	
1003100	[Computer name]:Alarm [alarm number] unlocked [alarm text of unlocked alarm] This alarm occurs when a alarm is unlocked.	
1003101	Acknowledgment request for alarm [alarm number] was issued This alarm occurs when a alarm is acknowledged.	
1003102	[Computer name]:Alarm group [alarm group number] locked This alarm occurs when a alarm group is locked.	
1003103	[Computer name]:Alarm group [alarm group number] unlocked This alarm occurs when a alarm group is unlocked.	
1003104	[Computer name]:Alarm logging overflow ended - no more alarms will be lost	
1003105	[Computer name]:Incoming alarm queue has reached critical value	
1003106	[Computer name]:Incoming alarm queue has reached normal value	
1003107	[Computer name]:Alarm [alarm number] hidden:@1%s@	
1003108	[Computer name]:Alarm [alarm number] shown:@1%s@	
1003109	Connection to master is down.	
1003110	Found incomplete configuration data for alarm [PC name]: . Status: @1%d@.	
1003300	Edited configuration data for alarm found.	
1003301	Error while logging on to receive alarm.	
1003302	Not all alarms could be acknowledged due to temporary resource bottleneck. Acknowledge again.	

3.8.9.8 1004000 - Alarms Print jobs

Meaning of the system events

The most important system alarms are listed below.

Table 3-25 100400 - Alarms Print jobs

Number	Effect/causes	Solution
1004000	General error	
1004001	PRT_OUT directory full	
1004002	Spool directory full	
1004003	Report was not printed. PRT_OUT directory full	
1004004	Report was not printed. Spool directory full	
1004005	Alarm sequence report is reprinted	
1004006	Spool directory full	
1004007	Hardcopy was not printed. Spool directory full	

3.8.9.9 1005000 - Alarms Text library

Meaning of the system events

The most important system alarms are listed below.

Table 3-26 100500 - Alarms Text library

Number	Effect/causes	Solution
1005000	General error	
1005001	Error while logging off the Runtime applications.	
1005002	Error while logging on the Runtime applications.	
1005003	Error during initialization of the MMF. A memory error has occurred.	
1005004	Error while loading the MMF. Error while accessing the database.	
1005005	Error while opening the MMF. A memory error has occurred.	
1005006	Error while creating the service window.	
1005007	No language found.	
1005008	Text ID not found. The requested Text ID could not be found in the text library.	
1005009	Read access to the MMF denied.	
1005010	Language not found. The requested language has not been configured in the text library.	

Number	Effect/causes	Solution
1005011	Languages table could not be opened. Either the data are faulty or the table is locked in the database.	
1005012	Text table could not be opened. Either the data are faulty or the table is locked in the database.	
1005013	Invalid language specified. The Language ID specified is invalid.	
1005014	DBConnect error No connection to the database could be set up.	

3.8.9.10 1006000 - Alarms Global scripts

Meaning of the system events

The most important system alarms are listed below.

Table 3-27 100600 - Alarms Global scripts

Number	Effect/causes	Solution
1006000	Error	
1007000	Overflow Overload: either there are too many actions running in a cycle that is too small, or an action is suspended (endless loop, dialog output). All the other actions are in the queue and cannot be processed.	
1007001	Action error One of the following errors has occurred: • Exception in the action (specific cause unknown) • Exception when accessing the return result (char* associated memory invalid) • Stack overflow upon execution of the action • The action contains a division by 0 • The action contains an access to a non-existing symbol • The action contains an access violation	You can integrate the OnErrorExecute function in your script, which allows for a detailed analysis of the error.
1007002	Overflow Internal lists have overflowed.	
1007003	Connection error The connection to the server is broken.	
1007004	Action error 1 The called function is not known.	Check the notation of the function call and implementation of the function.

Number	Effect/causes	Solution
1007005	Action error 2 This error can have several causes: • The action does not contain a P code. Recompile the action. • The function could not be loaded, for example, due to incorrect function name. • The type of return value of the function is invalid.	You can integrate the OnErrorExecute function in your script, which allows for a detailed analysis of the error.
1007006	Tag error A requested tag was not supplied by WinCC Explorer within 10 seconds.	Check the spelling of the tag name. In the case of external tags, there may be a communication problem between the WinCC Explorer and the PLC. You have the option to incorporate the OnErrorExecute function into your script, which allows for a detailed analysis of the error.
1007007	Info Additional information under "Diagnostics of WinCC / Runtime Monitoring for Actions".	
1007009	Error in thread Additional information under "Diagnostics of WinCC / Runtime Monitoring for Actions".	

3.8.9.11 1008000 - Alarms User administration

Meaning of the system events

The most important system alarms are listed below.

Table 3-28 100800 - Alarms User administration

Number	Effect/causes	Solution
1008000	Connection to the smart card reader interrupted	
1008001	Incorrect login name/password	
1008002	Incorrect login name/password by smart card	
1008003	Manual login	
1008004	Login with smart card	
1008005	Manual logout	
1008006	Logout with smart card	
1008007	Automatic logout through timer	
1008008	Authorizations effective for service user/group '@102%s@'	

3.8.9.12 1009000 - Alarms Lifebeat Monitoring

Meaning of the system events

The most important system alarms are listed below.

Table 3-29 100900 - Alarms Lifebeat Monitoring

Number	Effect/causes	Solution
1009000	Error	
1009999	Tag @2%s@ does not exist	

3.8.9.13 1010000 - Alarms STRRT

Meaning of the system events

The most important system alarms are listed below.

Table 3-30 101000 - Alarms STRRT

Number	Effect/causes	Solution
1010000	Error	

3.8.9.14 1011000 - Alarms Group display

Meaning of the system events

The most important system alarms are listed below.

Table 3-31 1011000 - Alarms Group display

Number	Effect/causes	Solution
1011000	Group display error at start up	
1011001	Group display hierarchy not updated	
1011002	Group display: Connection fault	
1011003	Group display: Tag does not exist	

3.8.9.15 1011101 - Alarms Picture Tree Manager

Meaning of the system events

The most important system alarms are listed below.

Table 3-32 1011101 - Alarms Picture Tree Manager

Number	Effect/causes	Solution
1011101	Error during startup	

3.8.9.16 1011201 - Alarms Split Screen Manager

Meaning of the system events

The most important system alarms are listed below.

Table 3-33 1011201 - Alarms Split Screen Manager

Number	Effect/causes	Solution
1011201	Error during startup	
1011202	The fill level of the project drive is over 80%.	
1011203	The project was created with the wrong WinCC version.	

3.8.9.17 1012001 - Alarms Time synchronization

Meaning of the system events

The most important system alarms are listed below.

Table 3-34 1012001 - Alarms Time synchronization

Number	Effect/causes	Solution
1012001	Switched to master operation	
1012002	Cannot send time frame	
1012003	Time reception service: Poor or failed signal	
1012004	Cannot receive time frame	
1012005	Cannot receive any time frame on redundant bus	
1012006	No time frame. Switch to redundant device	
1012007	Can successfully send time frame	
1012008	Time reception service functions properly	
1012009	Can successfully receive time frames	
1012010	Can receive time frames properly on redundant bus	
1012011	Switched to slave operation	

Number	Effect/causes	Solution
1012012	Time synchronization deactivated	
1012013	Time synchronization activated	
1012014	DCF77 client service failure	
1012015	DCF77 client service functions properly	
1012016	@2%s@ switched to master mode @3%s@	
1012017	@2%s@ switched to slave mode @3%s@	
1012018	@2%s@ cannot send time frame	
1012019	@2%s@ can successfully send time frame	
1012020	@2%s@ sets the local time	
1012021	LAN time synchronization with PC @2%s@ faulty	
1012022	LAN time synchronization set on PC @2%s@	
1012023	LAN time synchronization with PC @2%s@ established	
1012024	Configured device names of time synchronization do not match the PC installation	
1012025	LAN - time cannot be accepted from connected WinCC server	
1012026	Time jump noted - switched to permanent slave mode	
1012027	Time jump noted - time synchronization permanently disabled	
1012028	Time reception service not started	
1012029	Time reception service running	
1012030	Time synchronization disabled	

3.8.9.18 1012200 - Alarms Redundancy

Meaning of the system events

The most important system alarms are listed below.

Table 3-35 1012200 - Alarms Redundancy

Number	Effect/causes	Solution
1012200	Partner station has failed	
1012201	Partner station restarted	
1012202	Projects are not functionally identical	
1012203	Log synchronization failed	
1012204	Internal redundancy error	
1012205	Connection to partner faulty	
1012206	Connection to partner restored	
1012207	Partner server - WinCC has not been started	
1012208	Log synchronization starts	
1012209	Log synchronization complete	

Number	Effect/causes	Solution
1012210	Tag Logging being synchronized	
1012211	Tag Logging synchronization complete	
1012212	Alarm Logging being synchronized	
1012213	Alarm Logging synchronization complete	
1012214	User archive is being synchronized	
1012215	User archive synchronization complete	
1012216	Synchronization was interrupted	
1012217	Partner server project not activated	
1012218	Client was switched automatically	
1012219	Client was switched manually	
1012220	Synchronization not ready for all user archives	
1012221	Synchronization ready for all user archives	
1012222	Main connection faulty	
1012223	Main connection operational	
1012224	Backup connection faulty	
1012225	Backup connection operational	
1012226	Partner server project activated	
1012227	Error: Partner computer is not a server	
1012228	Log synchronization starts '@2%s@'	
1012229	Log synchronization complete '@2%s@'	
1012240	RedundancyControl error triggers switchover	
1012241	RedundancyControl: Switchover to status	
1012242	Delta download has been started	
1012243	Delta download is complete	
1012244	Overload during online synchronization of Alarm Logging	
1012245	RedundancyControl: Loss of serial connection	
1012246	RedundancyControl: Serial connection reestablished	
1012247	OS server (standby) redundancy error	
1012248	OS server (standby) redundancy reestablished	
1012250	Possible inconsistency of the log databases	
1012251	Internal error	
1012252	Backup: Error while writing to backup path	
1012253	Backup: Insufficient disk space on target drive	
1012254	No connection to the WriteArchiveServer	
1012255	No connection to Microsoft Message Queue	
1012256	No connection to the WinCC project	
1012257	No connection to the database	
1012258	No connection to the text library	
1012259	Error while creating the TagLogging data	
1012260	Error while creating the AlarmLogging data	

Number	Effect/causes	Solution
1012261	Notice: Backup is delayed until the partner server restarts.	
1012265	Database verification failed	
1012301	No access to SQL server (risk of data loss)	
1012348	Insufficient free disk space on project drive	
1012349	Loss of connection via network adapter (MAC) address	
1012350	Connection reestablished via network adapter (MAC) address	
1012351	RedundancyControl: System blockage detected. Switch to Fault status.	
1012352	RedundancyControl: System blockage detected. Restart your computer as soon as possible.	
1012354	[Computer name]:RedundancyControl: Status changed to FAULT. However, server isolation is not activated.	
1012355	[Computer name]:RedundancyControl: Status changed to FAULT. However, server isolation is locked by @1%s@. Reason: @2%s@	
1012356	RedundancyControl: Status changed to FAULT => server is isolated	
1012357	[Computer name]:RedundancyControl: Status changed to FAULT. However, automatic restart is not activated.	
1012358	[Computer name]:RedundancyControl: Status changed to FAULT. However, automatic restart is locked. The network adapter is disconnected and DHCP is released.	
1012359	[Computer name]:RedundancyControl: Reboot of computer disabled by @1%s@. Reason: @2%s@	
1012360	[Computer name]:RedundancyControl: Restart of the computer canceled. The last restart took place less than @1%s @ s ago.	
1012361	[Computer name]:RedundancyControl: Restart of the computer canceled. After @1%s@ restarts, no more restarts are permitted for @2%s@ s.	
1012362	[Computer name]:RedundancyControl: Rebooting computer in @1%s@ s	
1012363	[Computer name]:RedundancyControl: ERROR status deferred until partner has reached stable status.	

3.8.9.19 1012400 - Alarms WebNavigatorClient

Meaning of the system events

The most important system alarms are listed below.

Table 3-36 1012400 - Alarms WebNavigatorClient

Number	Effect/causes	Solution
1012400	WebClient @1%s@ Connection established (User=@2%s@)	
1012401	WebClient @1%s@ Connection terminated (User=@2%s@)	

3.8.9.20 1012500 - Alarms Process Historian

Meaning of the system events

The most important system alarms are listed below.

Table 3-37 1012500 - Alarms Process Historian

Number	Effect/causes	Solution
1012500	[Computer name]:Start recovery for Process Historian	
1012501	[Computer name]:Recovery for Process Historian completed	
1012502	[Computer name]:Communication with Process Historian not possible	
1012503	[Computer name]:Communication with Process Historian faulty	
1012504	[Computer name]:Communication with Process Historian restored	
1012505	[Computer name]:Process Historian server offline since @1%s@	
1012506	[Computer name]:Buffer limit of channel @1%s@ exceeded	
1012507	[Computer name]:Buffer limit of channel @1%s@ normal	
1012508	[Computer name]:Less than @1%d@ GB of disk space available on data storage medium '@2%s@' for communication with the Process Historian.	
1012509	[Computer name]:Communication with the Process Historian was terminated. Less than @1%d@ GB of disk space available on data storage medium '@2%s@'.	

Number	Effect/causes	Solution
1012510	[Computer name]:Connection to the Process Historian could not be established (check configuration).	
1012600	[Computer name]:@1%d@% of data storage capacity used	
1012601	[Computer name]:System out of resources	
1012602	[Computer name]:Redundancy failed	
1012603	[Computer name]:Redundancy restored	
1012604	[Computer name]:License volume exceeded Shutdown in @1%d@ days	
1012605	[Computer name]:PH-Ready @1%s@ failed	
1012606	[Computer name]:An automatic redundancy switchover of the Process Historian servers has taken place.	
1012607	[Computer name]:Less than @1%d@ GB of disk space available for Process Historian database.	
1012608	[Computer name]:Less than @1%d@ GB of disk space available for the "tempdb" database.	
1012609	[Computer name]:Less than @1%d@ GB of disk space available for emergency restore operation.	
1012610	[Computer name]:New backup for emergency restore operation failed. Insufficient memory.	
1012611	[Computer name]:An unknown error occurred while creating a backup for the emergency restore operation.	
1012612	[Computer name]:Memory path @1%s@ not accessible for emergency restore operation.	
1012613	[Computer name]:The emergency threshold value for drive @1%s@ was reached. Process Historian is locked as a result.	
1012614	[Computer name]:None of the prepared segments exists. Please check why this is.	
1012615	[Computer name]:Not all prepared segments have been created.	

3.8.9.21 1012700 - Alarms Redundancy self-diagnostics

Meaning of the system events

The most important system alarms are listed below.

Table 3-38 1012700 - Alarms Redundancy self-diagnostics

Number	Effect/causes	Solution
1012700	[Computer name]: Value @7%s@ of device @10%s@ is invalid.	
1012701	[Computer name]: Value @7%s@ of device @10%s@ has violated the error limit HIGH.	
1012702	[Computer name]: Value @7%s@ of device @10%s@ has violated the error limit LOW.	
1012703	[Computer name]: Value @7%s@ of device @10%s@ has violated the warning limit HIGH.	
1012704	[Computer name]: Value @7%s@ of device @10%s@ has violated the warning limit LOW.	
1012705	[Computer name]: Value @7%s@ of device @10%s@ no longer violates the error limit.	
1012706	[Computer name]: Value @7%s@ of device @10%s@ is OK.	
1012707	[Computer name]: Device @10%s@ causes @2%s@.	
1012708	[Computer name]: Value @7%s@ of device @10%s@ is invalid.	

3.8.9.22 1016000 - Alarms IndustrialDataBridge

Meaning of the system events

The most important system alarms are listed below.

Table 3-39 1016000 - Alarms IndustrialDataBridge

Number	Effect/causes	Solution
1016000	@1%s@ provider initialization failed for connection: @2%s@.	
1016001	@1%s@ consumer initialization failed for connection: @2%s@.	
1016002	@1%s@ provider data transmission failed for connection: @2%s@.	
1016003	@1%s@ consumer data transmission failed for connection: @2%s@.	
1016004	Provider initialization database failed for connection: @2%s@.	

Number	Effect/causes	Solution
1016005	Consumer initialization database failed for connection: @2%s@.	
1016006	Provider data transmission database failed for connection: @2%s@.	
1016007	Consumer data transmission database failed for connection: @2%s@.	
1016008	Provider initialization dynamic database failed for connection: @2%s@.	
1016009	Consumer initialization dynamic database failed for connection: @2%s@.	
1016010	Provider data transmission dynamic database failed for connection: @2%s@.	
1016011	Consumer data transmission dynamic database failed for connection: @2%s@.	

3.8.9.23 12508141 - Alarms

Meaning of the system events

The most important system alarms are listed below.

Table 3-40 12508141 - Alarms

Number	Effect/causes	Solution
12508141	@10%s@: @7%s@ @102%s@ new=@3%g@ @8%s@ old=@2%g@ @8%s@	

3.8.10 Tag Bit Assignments (RT Professional)

3.8.10.1 Tags for Alarm Groups and Alarms (RT Professional)

Bit assignment of status tag (RT Professional)

Introduction

In WinCC you can define a status tag for an alarm or an alarm group. The status tag for am alarm group represents the states of the alarms and child alarm groups it contains.

Description

The following states of an alarm or alarm group are stored in a status tag:

- Incoming/outgoing alarm (status bit)
- Alarm queued for acknowledgment / acknowledged (acknowledgment bit)

Data type of a status tag

The status tag is an internal or external WinCC tag. You can use the following data types for status tags:

- Unsigned 8-bit value
  The data type represents the states of up to four alarms.
- Unsigned 16-bit value
  The data type represents the states of up to eight alarms.
- Unsigned 32-bit value
  The data type represents the states of up to 16 alarms.

Each alarm or alarm group occupies 2 bits in the status tag.

Note

A status bit of a status tag can only be assigned to one alarm.

Acknowledgment bit of the status tag of an alarm

When an incoming alarm is received and is not acknowledged, the acknowledgment bit has status "1".

If the alarm was acknowledged, the acknowledgment bit status changes to "0".

Alarm event	Acknowledgment bit
Incoming alarm was not acknowledged	1
Alarm acknowledged	0

Acknowledgment bit of the status tag of an alarm group

When at least one incoming alarm is received and is not acknowledged, the acknowledgment bit status in an alarm group changes to "1".

If all the alarms for the alarm group were acknowledged, the acknowledgment bit is reset to "0".

Status bit for the status tags of an alarm or alarm group

If an incoming alarm or alarm group is received and has not yet gone, the acknowledgment bit has status "1". If the alarm has gone, the acknowledgment bit status changes to "0".

Alarm event	Status bit
Incoming alarm or alarm group	1
Alarm gone	0

Note

Alarms whose alarm class has the "Alarm without status" property cannot be displayed in a status tag.

Status tag of a locked alarm

If you lock an alarm in Runtime, the bits in the status tags will be reset or not set, regardless of the alarm status. The "locked" status of an alarm is not indicated in the status tag.

Note

To visualize the locked status of an alarm group, use the locked tag for the alarm group.

Status tag for chronological alarms

You can use a structure tag to represent the status of a chronological alarm. In the structure tag, the "EventState" element, for example, represents the status of a chronological alarm.

Position of the Status and Acknowledgment Bit

You can define the position of the status bit within a status tag. The acknowledgment bit must be a certain distance from this bit. The following definitions apply to the data type:

Data type	Position of status bit	Distance	Position of acknowledgment bit
Unsigned 8-bit value	0 - 3	4	4 - 7
Unsigned 16-bit value	0 - 7	8	8 - 15
Unsigned 32-bit value	0 - 15	16	16 - 31

Examples

Status Tag of Data Type "32 Bit Unsigned"

Status bit = Position 9

Distance = 16 positions

Acknowledgment bit = Position 25

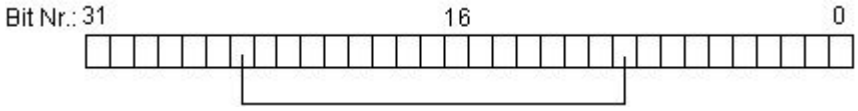

Status Tag of Data Type "16 Bit Unsigned"

Status bit = 6

Distance = 8 positions

Acknowledgment bit = Position 14

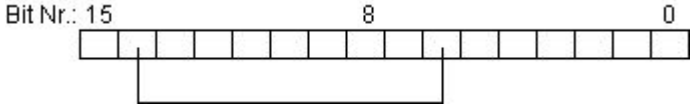

Status Tag of Data Type "8 Bit Unsigned"

Status bit = 2

Distance = 4 positions

Acknowledgment bit = Position 6

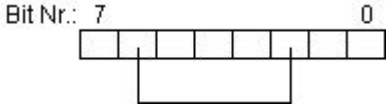

Configuring Function Lists for a Change in Value

You can configure a list of functions for the "Value change" event for the tag. You can hide any non-relevant bits of the tag to reduce the number of possible signal states for the tag. You can configure the hiding of the non-relevant bits using a C or VB script.

It is not necessary to hide the non-relevant bits if the following condition is fulfilled:

- You are using a separate status tag for each alarm.
- The non-relevant bits of these status tags always have the value "0".

See also

Tags of an alarm group (Page 778)

Bit assignment for acknowledgment tags (RT Professional)

Introduction

In WinCC you can define an acknowledgment tag for an alarm or alarm group.

Description

Use the acknowledgment tag to:

- Acknowledge an alarm
- Acknowledge an alarm group
  The acknowledgment tag of an alarm group acknowledges all the alarms and alarm subgroups it contains.
- Displaying the acknowledgment status of an alarm group
- Linking the acknowledgment to other WinCC components
  You can use the acknowledgment tag to link the acknowledgment to other WinCC components.

If the operator acknowledges an alarm or alarm group in Runtime, then the associated acknowledgment bit will be set.

Using acknowledgment tags

You have the following options available for configuring acknowledgment tags:

- You can configure a separate acknowledgment tag for each alarm or alarm group.
- You can compile several alarms or alarm groups in an acknowledgment tag. The acknowledgment bit allows you to distinguish between alarms and alarm groups.

Each alarm occupies one bit in the acknowledgment tag.

Data Type of an acknowledgment tag

The structure of the acknowledgment tag is not specified. The data type depends on the number of alarms that you want to represent in the acknowledgment tag.

Resetting the Acknowledgment Bit for acknowledgment tags

You configure the setting or resetting of the acknowledgment bit in WinCC using a button or via the PLC.

Note

When you acknowledge an alarm group in Runtime, the acknowledgment bit does not indicate this acknowledgment in the alarm view.

- See also "Resetting the acknowledgment bit"
- See also "Tags of an alarm group"

See also

Tags of an alarm group (Page 778)

3.8.10.2 Tags for Alarm Groups (RT Professional)

Bit assignment for lock tags (RT Professional)

Introduction

In WinCC you can define a lock tag for an alarm group.

Description

The lock tag of an alarm group is used to evaluate the locked status of that alarm group. You can define a lock bit in the lock tag for this purpose.

Using the Lock Tag

You have the following options for configuring lock tags:

- You can configure a separate lock tag for each alarm group.
- You can compile several alarm groups in a lock tag. The lock bit allows you to distinguish between alarm groups.

When you lock an alarm group in Runtime, the associated lock bit is set in the configured tag.

Data Type of a Lock Tag

The structure of the lock tag is not specified. The data type depends on the number of alarms that you want to represent in the lock tag.

See also

Tags of an alarm group (Page 778)

Bit assignment for display suppression tags (RT Professional)

Introduction

In WinCC you can define a display suppression tag for a custom alarm group. The display suppression tag saves the potential plant states. Only one plant state can be active at one time. The display suppression tag is therefore set by only one bit.

Description

In the display suppression mask, you can specify which alarm of an alarm group is to be suppressed for specific plant states in Runtime.

Data type of the display suppression tag

The display suppression tag is an external WinCC tag that stores the plant state. You can use the following data types for display suppression tags:

- Unsigned 8-bit value
- Unsigned 16-bit value
- Unsigned 32-bit value

Display suppression mask

In the Inspector window of an alarm, you can use the display suppression mask to define the plant states at which the alarm is not shown. The project engineer sets the bits that define specific plant states. A plant can have a maximum of 32 states.

You enter the selected plant states as hexadecimal value in the Inspector window of the alarm under "Display suppression > Mask".

The display suppression tag saves the plant state. When a plant state listed in the mask occurs, the alarm is no longer displayed. To view the alarms currently hidden due to a plant state, you can switch the alarm view in Runtime to "Alarms with display suppression".

Example

Five examples for the display suppression mask are offered below:

- Display suppression mask 0x0. The alarm is displayed.

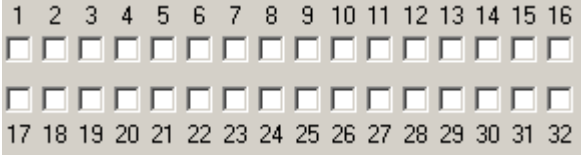

- Display suppression mask 0x1. If the "1" bit is set in the display suppression tag, the alarm is not displayed.

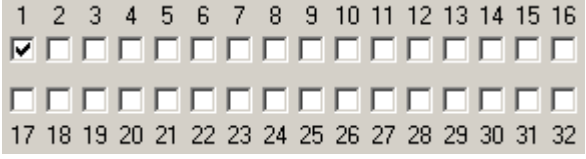

- Display suppression mask 0xD. If the "1", "3" or "4" bit is set in the display suppression tag, the alarm is not displayed.

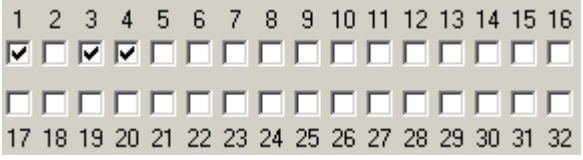

- Display suppression mask 0x80000008. If the "4" or "32" bit is set in the display suppression tag, the alarm is not displayed.

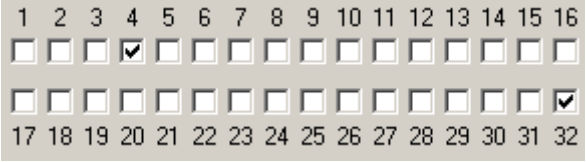

- Display suppression mask 0xFFFFFFFF. If any bit is set in the display suppression tags, the alarm is not displayed.

See also

Tags of an alarm group (Page 778)

Working with logs 4

4.1 Working with Logs (RT Advanced, Basic Panels, Panels, Comfort Panels)

4.1.1 Log Basics (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

WinCC provides the following types of log for logging process data for HMI Runtime:

- Data logs
- Alarm logs

A data log is used to log process data from an industrial plant.

An alarm log is used to log alarms that occur in the monitored process.

Principle

The two types of log both have roughly the same structure and largely work in the same way. They are very clear and easy to configure. With both types of log you define the same properties for the log. The same logging methods are also available for both types of log.

The following logging methods are available:

- Circular log
  If a circular log is totally full, the oldest entries are overwritten.
- Segmented circular log
  In a segmented circular log, multiple single logs of the same size are filled in succession. When all log segments are filled, the oldest log segment is overwritten.
- Log with level-dependent system alarm
  A system alarm is triggered when a defined level is reached.
- Log with level-dependent triggering of an event
  The "Overflow" event is triggered when the log is full. The "Overflow" event triggers a system function.

4.1.2 Properties of Logs (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You define the properties of a data log in the "Data logs" editor.

You define the properties of an alarm log in the "Alarm log" editor.

The properties of the data log and alarm log are configured in the same way. You configure the properties either directly in the table of the respective editor or in the log properties of the Inspector window.

General properties

- Name
  You can assign any name to the log. The name must contain at least one letter or one number.
- Storage location
  The storage location specifies where the log is saved. The HMI device determines which storage locations are available.
- Size
  The size of a log depends on the type of log and the selected settings.
  - Size of a data log
    The size of a data log is calculated as follows:
    The number of items * the length of each tag value to be logged.
    In the "Properties" window, the maximum size that the log accepts for retention of the currently selected number of data records is displayed in the input field "Number of data records". The maximum log size is limited by the volume of the storage medium.
  - Size of an alarm log
    The size of an alarm log is calculated from the number of data records and the approximate size of an entry. The size of an entry depends on whether the alarm text and the associated tag values are to be logged as well.
- Restart characteristics
  Under Restart characteristics you can specify that the logging starts when Runtime starts. Check the "Enable logging at runtime start" check box.
  You can also control the behavior when Runtime starts. Enable "Reset log" if you want to overwrite existing logged data with the new data. Select the "Append data to existing log" option if you want to retain the existing logged data. This setting adds the data to be logged to an existing log.

Note

You can use system functions to control the restarting of a log in Runtime.

Automatic log entries

In Runtime, the following log entries are created as standard:

Entry	File format	Log type	Meaning
RT_DIS	Any	Data log	Indicates that the connection to the log was interrupted at this point in time. (A bold line is shown in the trend view for this time period.)
RT_OFF	Any	Data log	Indicates that Runtime was shut down at this point in time. (No line is shown in the trend view for this time period.)
RT_ERR	Any	Data log Alarm log AuditTrail[1]	Indicates in the destination log that a copy operation was not successful or was interrupted. (The log copy was not fully created.)
$RT_COUNT	*.CSV *. TXT	Data log Alarm log AuditTrail[1]	This entry was created at the end of the log and serves to increase the system performance when Runtime starts.

[1] The "AuditTrail" logging method is not available for all HMI devices.

See also

Log Basics (Page 1027)

Basic principles for data logging (Page 616)

Basics on alarm logging (Page 885)

4.1.3 Storage locations of logs (RT Advanced, Basic Panels, Panels, Comfort Panels)

Storage location of a log

When you configure a log in WinCC, the available storage locations depend on the HMI device you are using.

HMI devices	Supported logs			Supported storage locations	Supported storage locations
	Alarms	Tags	Audit Trail		
Basic Panels[1]	No	No	No	-	-
Basic Panels 2nd Generation[2]	Yes	Yes	No	a TXT file (Unicode)	USB memory (at USB port)
Comfort Panels[3]	Yes	Yes	Yes	a CSV file (ASCII) RDB file a TXT file (Unicode)	Storage card (SD) Storage card (USB) Network drive

HMI devices	Supported logs			Supported storage locations	Supported storage locations
	Alarms	Tags	Audit Trail		
Mobile Panels[4]	Yes	Yes	Yes	a CSV file (ASCII) RDB file a TXT file (Unicode)	Storage card (MMC) Storage card (USB) Network drive
• Mobile Panels 2nd Generation[5]	Yes	Yes	Yes	a CSV file (ASCII) RDB file a TXT file (Unicode)	Storage card (SD) USB memory (at USB port) Network drive
PC systems with Runtime Advanced	Yes	Yes	Yes	a CSV file (ASCII) Database RDB file a TXT file (Unicode)	Local file system Network drive

1 KP 300, KP 400, KTP 1000, TP 1500

2 KTP 400, KTP 700, KTP 900, KTP 1200

3 All HMI devices from the device list

4 Mobile Panel 277 only

5 KTP 400F Mobile, KTP 700 Mobile, KTP 900 Mobile - including fail-safe versions

Note

Logging on network drives

Do not log alarms, tags and Audit Trail directly on a network drive. Power supply can be interrupted at any time. This means there is no guarantee for a reliable operation of logs and audit trails.

Save the logs on your local hard drive or a local storage card. Use the system function "ArchiveLogFile" to save the logs long-term on a network drive. This step ensures reliable operation.

Syntax examples for storage locations

Storage card storage location:

- <\Storage Card MMC\My_Archives\TagLogs>: Saves the archive on the MMC storage card to the subdirectory "My_Archives\TagLogs".

Local file system storage location:

- <C:\My_File_Folder\My_Archives\Machine_1>: Saves the log on the local hard disk drive C: in the subdirectory "My_File_Folder\My_Archives\Machine_1"

Network drive storage location:

- <\\ArchiveServer\My_File_Folder\My_Archives\Machine_1>: Saves the archive on the "ArchiveServer" server to the subdirectory "My_File_Folder\My_Archives\Machine_1".

Naming conventions

The log names must be unambiguous in a project. The name of a log must always be unique, regardless of whether different storage locations are selected for the log.

Note

The characters which can be used in the name of the data source depend on the storage location.

The \ / * ? : " < > | characters are not allowed at the following locations:

- File - RDB
- File - CSV (ASCII)
- File - TXT (Unicode)

If the "Database" storage location is used, the following characters may be used: a-z A-Z 0-9 _ @ # $

The characters _ @ # $ cannot be used as the first character of a name.

Note

Only applies to Basic Panels 2nd Generation

Unicode characters are not supported for the log names and the log paths.

File - CSV (ASCII)

Data is saved to a CSV file in standard ASCII format.

If you want to read or evaluate logged data without using WinCC Runtime, use the "CSV file" storage location.

Note

Double quotation marks or multiple characters are not permitted as list separators for the "CSV file" storage location. You can find the settings for list separators under "Start > Settings > Control Panel > Regional and Language Options".

Note

Logging tags of the "BOOL" data type

Boolean values are logged as digits:

- 0 (False) corresponds to the log entry 0
- 1 (True) corresponds to the log entry -1

File - TXT (Unicode)

Data is stored in Unicode.

This file format supports all characters that can be used in WinCC and WinCC Runtime. For editing, you will need software that can save files in Unicode, such as Notepad.

Note

Use "File - TXT (Unicode)" as the storage location to log Asian languages.

File - RDB

Data is saved with quick access in a proprietary database.

If you require maximum read performance in Runtime, use the "RDB file" storage location.

Database

Data is saved to a database which is set up for ODBC access by the PC administrator.

Log with checksum (Audit Trail)

The following files are generated under special circumstances:

***.keep**

1. If a log is started without checksum and will be continued with a checksum.
2. If you update WinCC with a service pack or a new version and the Audit Trail or the log is continued with the checksum.

The content of the keep file will remain the same when compared with the original csv file or txt file.

***.bak**

If WinCC Runtime has determined a serious, irregular problem in the file.

See also

Basics on alarm logging (Page 885)

Log Basics (Page 1027)

4.2 Working with Logs (RT Professional)

4.2.1 Log basics (RT Professional)

Introduction

WinCC provides the following types of logs for logging process data in Runtime Professional:

- Fast data log
- Slow data log
- Alarm logs

Data logs are used to log process data from an industrial plant.

Alarm logs are used to log alarms that occur in the monitored process.

Principle

The different types of logs all have roughly the same structure and largely work in the same way. They are very clear and easy to configure. With all three types of log you define the same properties for the log type.

All three log types can be broken down into individual segments. Segmenting can be made based on size or time.

There are two types of logs available for data logs:

- Fast data log
  This type of log is used to log process data that change rapidly with a cycle time <= 1 min.
- Slow data logs
  This type of log is used to log all process data not covered by the "Fast data log" type. In the configuration, you can specify several parameters to define the tag values to be saved in a certain type of log.

The alarm log settings are independent of the data log settings.

4.2.2 Properties of logs (RT Professional)

Introduction

You can configure the basic properties of logs in Runtime Professional, such as the segment size, in the "Runtime settings > Logging" dialog. The settings configured in this dialog apply to all the logs in the project.

You configure the special properties for the individual log types either directly in the table of the respective editor or in the log properties of the Inspector window.

You configure the logical organization of data logs and compressed logs in the "Historical Data" editor. See "Data logging (Page 633)" for additional information.

There are no comparable logical structures within the log for alarm logs. This means there is no separate "Alarm log" editor in Runtime Professional. You can only specify whether or not an alarm should be logged. You configure logging of alarms in the "HMI alarms" editor in the "Alarms classes" tab. See "Basics on alarm logging (Page 900)" for additional information.

Properties

The following properties are available for segmenting:

- Time period of all segments
- Maximum size of all segments
- Time period contained in one single segment
- Maximum size of a single segment
- Time of first segment change

You can also specify how and when backups are to be created.
Additional properties are available for the "Fast data" log type which you can use to divide the process data between the "Fast" and "Slow" log types.
See "Logging Settings (Page 660)" for more details.

See also

Data logging (Page 633)

Basics on alarm logging (Page 900)

Logging Settings (Page 660)

Log basics (Page 1033)

Data logging (Page 633)

Configuring alarm logging (Page 900)

Access to archive data via WinCC OLE DB Provider (RT Professional) 5

5.1 WinCC Archive Connector (RT Professional)

5.1.1 WinCC Archive Connector (RT Professional)

Functionality

The WinCC "Archive Connector" is used for configuring the access to the archive database.

The tool is an integral part of WinCC DataMonitor.

With the Archive Connector, already swapped out WinCC archives can be reconnected to an SQL Server.

The DataMonitor client or WinCC OLE DB Provider can then access the archives once again.

Functions of the WinCC Archive Connector:

- Manual Connection: Local databases may be selected and connected to the local SQL server.
- Manual Disconnection: Connected databases may be selected and disconnected from the SQL server.
- Automatic Connection: Local directories can be selected in which WinCC archives are swapped. All archives which were added to the selected directories from the moment change monitoring was activated are automatically linked to the SQL server.

The Archive Connector may only be operated using a local SQL server and a WinCC DataMonitor license.

Once the configuration has been completed, the Archive Connector may be closed.

Note

WinCC RT archives in the directory "<Project Directory> \ ArchiveManager" and the associated subdirectories must not be connected to or disconnected with the Archive Connector because their connection to the SQL server is managed by the WinCC RT Professional HMI device.

The path for the swapped out WinCC archives is set in WinCC with the Archive Configurator of tag logging, for example, not with the WinCC Archive Connector.

If access is to be made to swapped out archives which are on stored removable media, such as tape or MOD drives, pay attention that the connection to these archives on this medium is disconnected using the Archive Connector before changing the medium in the drive. After changing the medium, the user should check the Archive Connector whether or not the archives on the new medium are connected.

Configuration with the WinCC Archive Connector should be restricted to a limited group of people. This means access to the tool should be protected by using the Windows "Administrators" user right or other Windows protection mechanisms, such as storage in a protected directory.

Note

Use the Archive Connector or DataMonitor to access the connected archives.

The following objects do not give any access to the archives linked with the Archive Connector:

- Alarm view
- Table view
- f(t) trend view
- f(x) trend view

5.1.2 The "Configuration" tab (RT Professional)

The "Configuration" tab

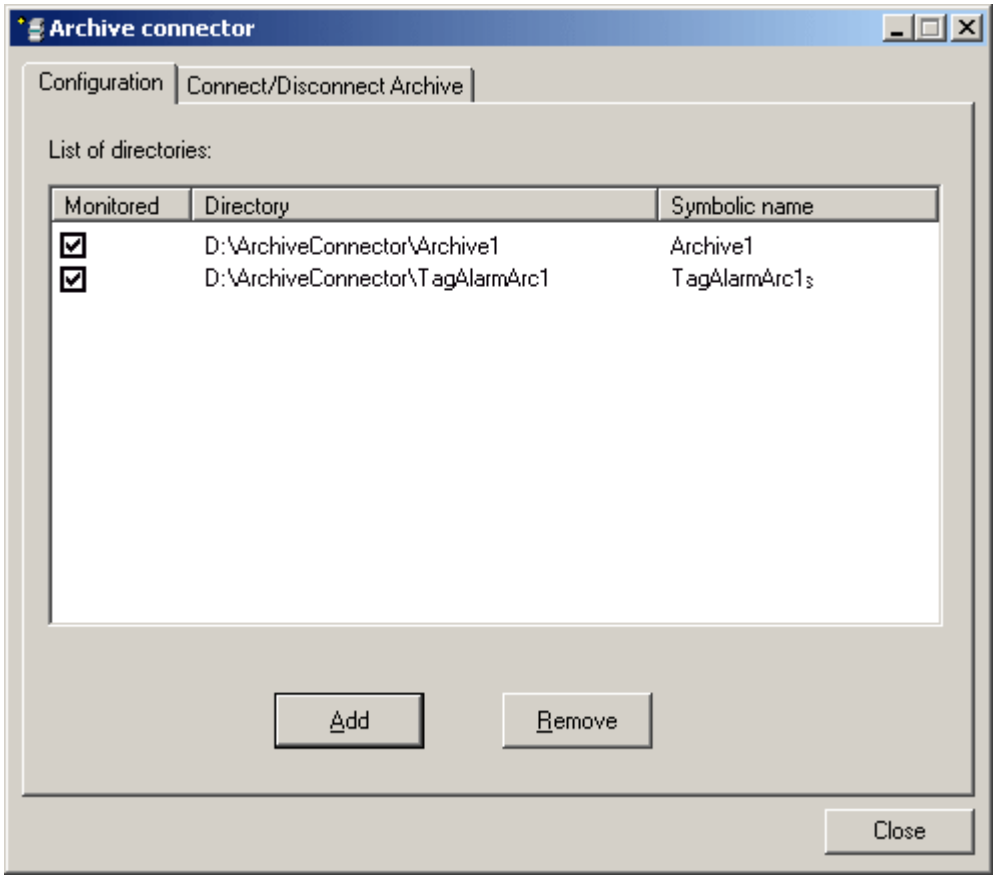

In the "Configuration" tab, archiving folders are displayed and managed that are to be accessed through the web or the WinCC OLE DB-Provider.

Archiving folders can be added or removed by using the buttons. A symbolic, unique name has to be assigned during configuration for each archiving folder.

The DataMonitor client or the WinCC OLE DB-Provider use the symbolic name to access the archive.

The name is also used for managing and connecting swapped out data from several computers or projects.

The symbolic names must only contain characters permitted in the SQL syntax.

By activating the corresponding check box, all archives added to the selected folder at the time of activation will automatically be connected to the SQL server.

If you activate or deactivate monitoring, the changes will not be activated until you close the Archive Connector.

5.1.3 The "Connect/Disconnect Archive" tab (RT Professional)

The "Connect/Disconnect Archive" tab

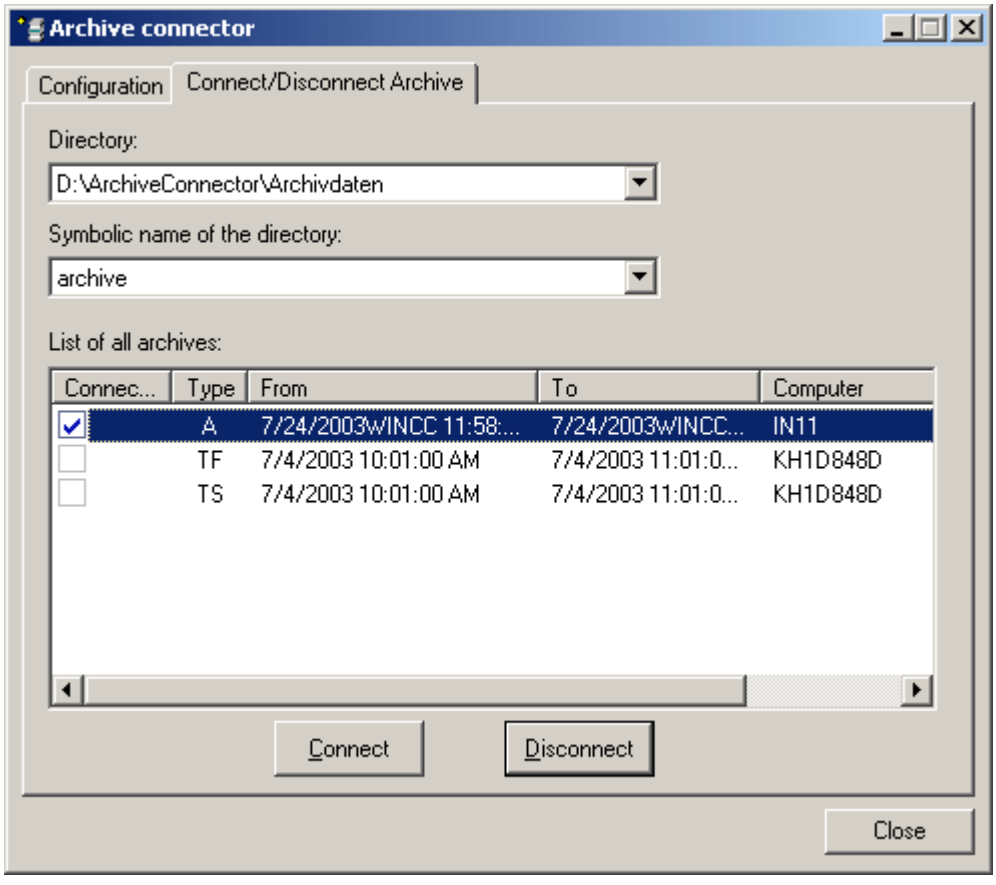

The "Connect/Disconnect Archive" tab lists all archives existing in the archiving directories.

The connection status of each archive is displayed. The connection to the archives can be established or terminated by using the buttons.

The archive type is shown in the "Type" field.

- "A" = Alarm Logging;
- "TF" = Tag Logging (Fast);
- "TS" = Tag Logging (Slow).

The columns "From" and "To" provide information on the local time zone.

Note

The connection of multiple, swapped-out archives to the SQL server may take several seconds.

It is not possible to connect a database file with the same name twice.

The WinCC Archive Connector connects finalized and backed up (swapped-out) archives to the SQL server. Archives not finalized are not supported.

The user interface language of the Archive Connector is based on the settings of the regional and language options in Windows.

Note

Archives created with SQL Server 2000 can only be connected if you remove the write protection of the archive. Archives that are linked can no longer be linked with the Archive Connector under SQL Server 2000.

In order to connect swapped out archives on CD or DVD, copy the archives to a media on which you can remove the write protection.

5.2 Access to archive data via OLE DB Provider (RT Professional)

5.2.1 Options for accessing archive data (RT Professional)

Introduction

Using OLE DB, you have the following options for accessing WinCC archive data and for displaying them using an external interface.

Access with WinCC OLE DB Provider

WinCC OLE DB provides access to all WinCC archive data.

Depending on the configuration, process data of WinCC are stored in compressed form. WinCC OLE DB Provider permits transparent access even to these data.

Use the "SQL Server Import / Export Wizard" to take advantage of standard SQL queries. You can save the unzipped files to an intermediate database using the wizard; you access the database with standard SQL queries.

Note

If WinCC closes a full archive and opens a new one, no data from the message and process value archives are read momentarily via the OLE DB Provider.

Access with Microsoft OLE DB

Microsoft OLE DB provides access to all WinCC user archives.

Note

Microsoft OLE DB is only tested and released for access to WinCC user archives but not to alarm and process value archives.
Use the WinCC OLE DB Provider to access message and process value archives.

5.2.2 Configuring access to archive data (RT Professional)

Configuration options

For access to databases with WinCC OLE DB, you may write your own applications.

For the communication with the WinCC OLE DB-Provider, ADO DB is used in applications created with, for example, Visual Basic, VBScript or VBA.

Note

Special characters in tag names

Please not that programming languages, such as Visual Basic, VBScript or VBA, only allow the following characters in the tag names: "A to Z", "a to z", "0 to 9" and "_".

If you use special characters, such as "," or ";", in the tag names in WinCC, the script will be aborted with an error message. In this case use the "Tag ID" to access a tag with special characters in the script name.

Requirements

- WinCC RT Professional or WinCC V12 must be installed on the accessing computer for access to archive data.
- For swapped out archives, establish the connection between the SQL database and the swapped out archives with the WinCC Archive Connector.

  Note

  The WinCC RT archives in the directory "<Project directory> \ ArchiveManager" and the associated subdirectories must not be connected or disconnected with the Archive Connector, because their connection to the SQL server is managed by WinCC RT Professional.

Procedure

1. Establish the connection to the database, for example, by using MS Excel or your own application.
2. Define the required selection criteria.
3. Read out the archive data.
   The query result may be displayed in MS Excel, for example, or exported as a CSV file.

5.2.3 Establishing the Connection to the Archive Database (RT Professional)

Introduction

For ActiveX data objects (ADO), the connection between the application and the archive database is established by the connection object. An important parameter here is the ConnectionString. The ConnectionString contains all necessary information for access to the database using OLE DB Provider.

Structure of the ConnectionString

```
"Provider = Name of the OLE DB Provider; Catalog=Datebase name;Data
Source=Server name;"
```

Parameter	Description
Provider	Name of the OLE DB Provider: e.g. WinCCOLEDBProvider
Catalog	Names of the WinCC database With WinCC RT databases, you will use database names that end in "R." <Datenbankname_R>. The database "CC_ExternalBrowsing" can also be used. If you have connected swapped out WinCC archives to the SQL Server via the WinCC Archive Connector, use their symbolic name. **Note** Enter the WinCC project name for "Catalog" for transparent access; for e.g.: "Catalog=WinCC_Project_Name". **Note** If you access message archives or swapped out archives via "CC_ExternalBrowsing", this access may take several minutes.
Data Source	Server name Local: ".\WinCC" or "<Computer Name>\WinCC" Remote: "<Computer name>\WinCC" **Note** Enter the transparent access to the Central Archive Server and in case of redundant servers enter the following via the OLE DB-Provider for "Data Source": <Symbolic Computer Name>::\WinCC. **Note** Use the archive tag name to directly access an archive tag on the central archive server CAS. The central archive server CAS returns the CAS-ID and not the archive tag ID as ID: <SYMBOLIC COMPUTER NAME>\\<Achive_Var_Name>

Example Process Value and Message Archive:

In the following example, a connection object is created with subsequent opening of the connection to the WinCC database (process value and message archive).

```
Set conn = CreateObject("ADODB.Connection")
```

```
conn.open "Provider=WinCCOLEDBProvider.
1;Catalog=CC_OpenArch_03_05_27_14_11_46R;Data Source=.\WinCC"
```

Example User Archive:

In the following example, a connection object is created with subsequent opening of the connection to the WinCC user archive.

```
Set conn = CreateObject("ADODB.Connection")
```

```
conn.open "Provider=SQLOLEDB.1; Integrated Security=SSPI; Persist
Security Info=false; Initial Catalog=CC_OpenArch_03_05_27_14_11_46R;
Data Source=.\WinCC"
```

Note

In order to improve performance during local access, enter "<Computer Name>\WinCC" as the data source instead of ".\WinCC".

5.3 Application cases (RT Professional)

5.3.1 Use Case 1: Local Access to WinCC RT Databases (RT Professional)

Requirements

The following licenses must be installed on the HMI device:

- WinCC RT Professional

Principles

An application uses the WinCC OLE DB-Provider to access the local WinCC Runtime database.

You may locally analyze the archive data and may, for example, calculate the standard deviation of a process value.

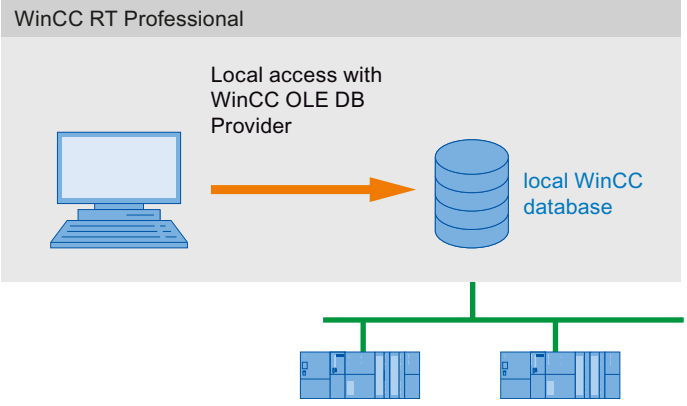

5.3.2 Use Case 2: Remote Access to WinCC RT Databases (RT Professional)

Requirements

The following licenses must be installed on the HMI device:

- WinCC RT Professional

The access may take place under various configurations of the Connectivity Pack Client.

- A WinCC software, such as WinCC V12, Web Navigator server or DataMonitor server, is installed on the client computer.
  The client does not have to be installed explicitly. Licensing is provided through WinCC licenses.

Principles

The client accesses the WinCC runtime database of an HMI device with WinCC RT Professional remotely. The client reads the data of the process value archives and alarm logs using the WinCC OLE DB Provider.

Since in this use case, the swapped-out WinCC archive is not accessed, the Archive Connector does not have to connect WinCC archives to an SQL server.

You can display, evaluate or further process the data on the client by exporting it to a CSV file, for example.

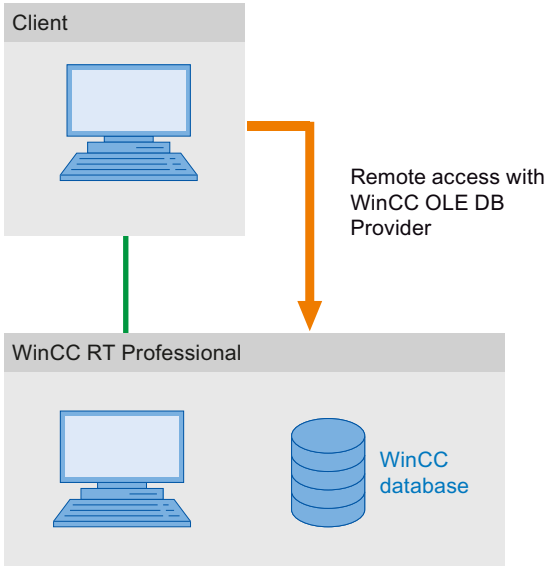

5.3.3 Use Case 3: Local Access to WinCC Archive Databases (RT Professional)

Requirements

The HMI device requires the following installations:

- WinCC V12
- WinCC RT Professional

Principles

An application accesses the local archive database using WinCC OLE DB Provider. The older archive data are copied from the WinCC RT database to a separate directory on the same computer.

With the Archive Connector, the swapped out WinCC archives are reconnected to an SQL Server. The archives are then available for access using WinCC OLE DB Provider.

Local archive data may be displayed, searched or analyzed, e.g. to search for process errors or to optimize processes.

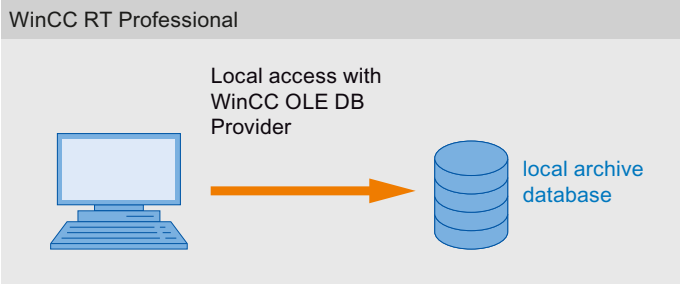

5.3.4 Use Case 4: Remote Access to WinCC Archive Databases (RT Professional)

Software Requirements

The long-term archive server requires the following to be installed:

- Connectivity Pack Server

Access may take place under various configurations of the client.

- A WinCC software, such as WinCC V12 RT Professional, WinCC V12, Web Navigator server or DataMonitor server, is installed on the client computer.
  The client does not have to be installed explicitly.
  Licensing is provided through WinCC licenses.

Principles

A long-term archive server is used to secure database files of process value and alarm logs, for example, in a monthly backup.

With the Archive Connector, already swapped out WinCC archives can be reconnected to an SQL server. The archives are then available for access using WinCC OLE DB Provider.

The client accesses the archives with the WinCC OLE DB-Provider . Using a VB application, for example, the archives may be analyzed, and process values of a specific day may be displayed.

Note

It may take several minutes if you access alarm logs or swapped out archives with "CC_ExternalBrowsing".

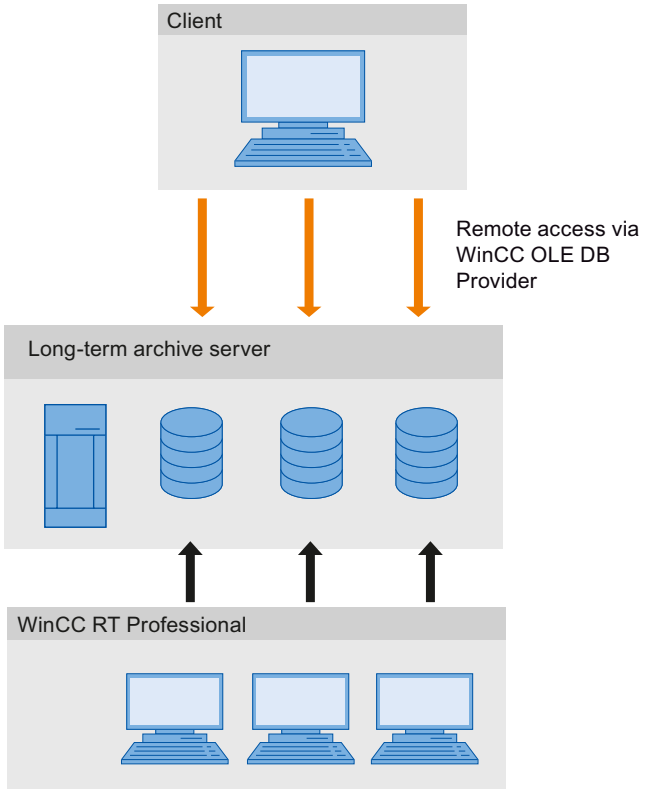

5.3.5 Use Case 5: Local Access to WinCC User Archive (RT Professional)

Requirements

The HMI device requires the following installations:

- WinCC RT Professional

Principles

An application accesses the local WinCC user archives with the Microsoft OLE DB Provider.

Using a VB application, for example, you may display, search and write back modified values to local archive data.

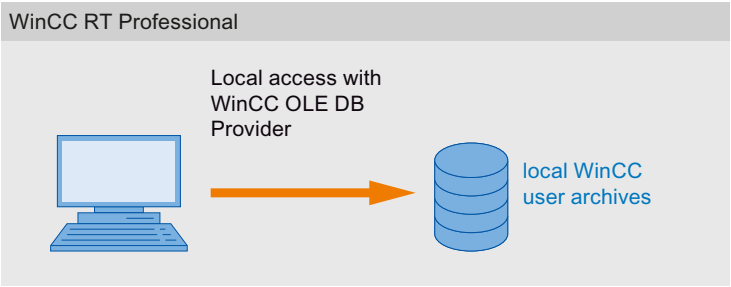

5.3.6 Use Case 6: Remote Access to WinCC User Archives (RT Professional)

Requirements

The HMI device requires the following installations:

- WinCC RT Professional

Access may take place under various configurations of the Connectivity Pack Client.

- A WinCC software, such as WinCC RT Professional, Web Navigator server or DataMonitor server, is installed on the client computer.
  The client does not have to be installed explicitly. Licensing is provided through WinCC licenses.

Principles

The client accesses the WinCC user archives with the MS OLE DB Provider.

Using a VB application, for example, you may display, search, and write back modified values for archive data.

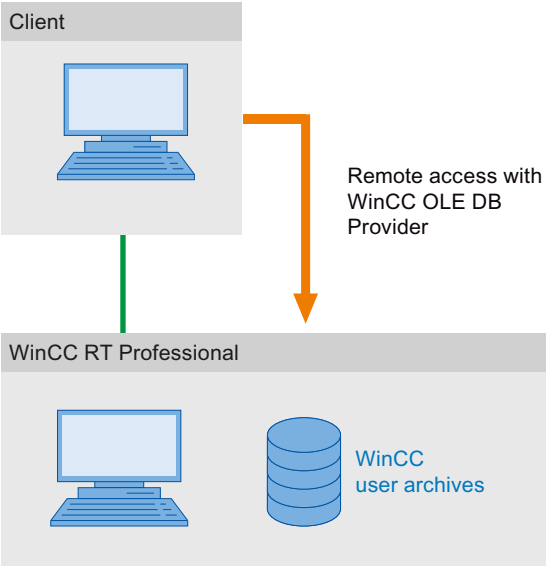

5.4 Querying the archive data (RT Professional)

5.4.1 Displaying Process Value Archives (RT Professional)

Introduction

The query result is returned as the Recordset. In this chapter, the structure of Recordset for process value archives is described.

Recordset Structure

Field name	Type	Comments
ValueID	Integer 4 Bytes or Integer 8 Bytes	Unique identification of value. The length depends on the type of query.
TimeStamp	DateTime	Time Stamp
RealValue	Real 8 Bytes	Tag value
Quality	Integer 4 Bytes	QualityCode of value (e.g. "good" or "bad").
Flags	Integer 4 Bytes	Internal Control Parameter

5.4.2 Querying Process Value Archives (RT Professional)

Principle

With the following query, a process value archive can be accessed. The data can be selected using filter criteria. The queries are forwarded to the database by the command object.

Note

The length of the ValueID can be different.

- For databases processed on a central archive server (CAS), the ValueID is 8 Bytes long and includes a server ID in the HI-DWORD area as well as the ValueID assigned by the respective server in the LO-DWORD area.
- For all other databases, the ValueID is 4 Bytes long and includes only the unique ValueID assigned by the WinCC server.

The 4-Bytes request via TAG:R is still available for compatibility. The 4-Byte ValueID returned is no longer unique in case of CAS databases.

Queries for process value archives are limited to a maximum of 20 tags, each with a maximum of 128 characters per tag.

Syntax

Note that the query may not contain any spaces.

Request of ValueIDs 8 Bytes long:

```
TAG_LLVID:R,<ValueID oder
ValueName>,<TimeBegin>,<TimeEnd>[,<SQL_clause>][,<TimeStep>]
```

Request of ValueIDs 4 Bytes long:

```
TAG:R,<ValueID oder ValueName>,<TimeBegin>,<TimeEnd>[,<SQL_clause>]
[,<TimeStep>]
```

Parameter

Parameter	Description
ValueID	Value ID from the database table Multiple namings are possible, e.g., "TAG:R,(ValueID_1;ValueID_2;ValueID_x),<TimeBegin>,<TimeEnd>"
ValueName	ValueName in format 'ArchiveName\Value_Name'. The <ValueName> parameter must be enclosed in single quotes. Multiple namings are possible, e.g., "TAG:R,('ValueName_1';'ValueName_2';'ValueName_x'), <TimeBegin>,<TimeEnd>" **Note** Please not that programming languages such as Visual Basic, VBScript or VBA only allow the following characters in the tag names: "A...Z", "a...z", "0...9" and "_". In WinCC if you use special characters such as "," or ";" in the tag names then the script will be aborted with an error message. In such a case use the "Tag-ID" to access a tag with special characters in the script name.
TimeBegin	Start time in 'YYYY-MM-DD hh:mm:ss.msc' format When <TimeStep> is used, <TimeBegin> must be specified as an absolute time. Relative time information or "0000-00-00 00:00:00.000" is not possible.
TimeEnd	End time in 'YYYY-MM-DD hh:mm:ss.msc' format
SQL_Clause	Filter criterion in SQL syntax: [WHERE search_condition] [ORDER BY {order_expression [ASC|DESC] }] The "ORDER BY" criterion can only be used when the specified sorting order is "{order_expression [ASC|DESC] }". Example: The following query supplies all values of the "ValueName_1" and "ValueName_2" tags that are less than 50 or greater than 100. "TAG:R,('ValueName_1';'ValueName_2'),<TimeBegin>,<TimeEnd>, 'WHERE RealValue > 100 OR RealValue < 50'"

Parameter	Description		
TimeStep	Values within the specified time interval are collected, beginning with the start time <TimeBegin> Format: 'TIMESTEP=x,y' x = Interval in seconds y = Aggregation type, defines the interval result The following values are possible for aggregation type:		
	Without interpolation	With interpolation	Description
	1 (FIRST) 2 (LAST) 3 (MIN) 4 (MAX) 5 (AVG) 6 (SUM) 7 (COUNT)	257 (FIRST_INTERPOLATED) 258 (LAST_INTERPOLATED) 259 (MIN_INTERPOLATED) 260 (MAX_INTERPOLATED) 261 (AVG_INTERPOLATED) 262 (SUM_INTERPOLATED) 263 (COUNT_INTERPOLATED)	First value Last value Minimum value Maximum value Mean value Sum Number of values
	Without interpolation means: If no values are present in the interval, no interval result will be returned. With interpolation means: If no values are present in the interval, the value will be derived by linear interpolation from the results of the neighboring intervals that are not empty. No extrapolation is done. Example: When TIMESTEP=60,257, for each interval of 60 seconds, the first value of this interval or - if there are no values in this interval - the linear, interpolated value from the first values of the neighboring intervals are returned. "TAG:R,1,'2013-07-09 09:03:00.000','0000-00-00 00:10:00.000','TIMESTEP=60,257'"		

Note

<TimeBegin> and <TimeEnd> must not both be "ZERO" = '0000-00-00 00:00:00.000'.

In order to improve performance, use the parameter "ValueID" instead of "ValueName" in the query. You can determine the "ValueID" from the "Archives" table.

Some applications cannot resolve the time of process values into 1 ms increments, which can lead to inaccuracies.
An example of resolving the milliseconds from the time stamp of process values can be found in the VB script "SplitDateTimeAndMs" in section "Example: Reading Process Value Archive Using WinCC OLE DB Provider". The script is also implemented in the demo project "OpConPack".

Selection of an Absolute Time Interval

Reads from start time <TimeBegin> to end time <TimeEnd>.

Example A1:

Reads the values of the ValueID 1 from start time 9:03 hours to end time 9:10 hours.

```
"TAG:R,1,'2013-07-09 09:03:00.000','2013-07-09 09:10:00.000'"
```

Selection of a Relative Time Interval

Reads from beginning of recording:

```
<TimeBegin> = '0000-00-00 00:00:00.000'
```

Reads until end of recording:

```
<TimeEnd> = '0000-00-00 00:00:00.000'
```

<TimeBegin> and <TimeEnd> must not both be "ZERO" = '0000-00-00 00:00:00.000'.

Note

Enter a relative period you want to query in a linked archive database using the following format:

- 0000-00-DD hh:mm:ss.msc

If you indicate the time frame in months, the content can be faulty, because a month can have 28 to 31 days.

Example B1:

Reads the absolute time from "TimeBegin" to end of recording, i.e. the last archived value.

```
<TimeBegin> = '2013-02-02 12:00:00.000', <TimeEnd> = '0000-00-00
00:00:00.000'
```

Example B2:

Reads the absolute time from "TimeBegin" for the next 10 seconds.

```
<TimeBegin> = '2013-02-02 12:00:00.000', <TimeEnd> = '0000-00-00
00:00:10.000'
```

Example B3:

Reads 10 seconds backward from the absolute time from "TimeEnd".

```
<TimeBegin> = '0000-00-00 00:00:10.000', <TimeEnd> = '2013-02-02
12:00:00.000'
```

Example B4:

Reads the values of the last hour starting from the time of the last archived value for multiple valueIDs (1;3;5;6).

```
"TAG:R,(1;3;5;6),'0000-00-00 01:00:00.000','0000-00-00
00:00:00.000'"
```

Example B5:

Reads the values of the last five minutes starting from the time of the last archived value for "TAG_2" from the "ArTags" archive.

```
"TAG:R,'ArTags\TAG_2','0000-00-00 00:05:00.000','0000-00-00
00:00:00.000'"
```

The following diagrams shows a possible result of this example. The query was implemented using the Connectivity Pack Demo Project.

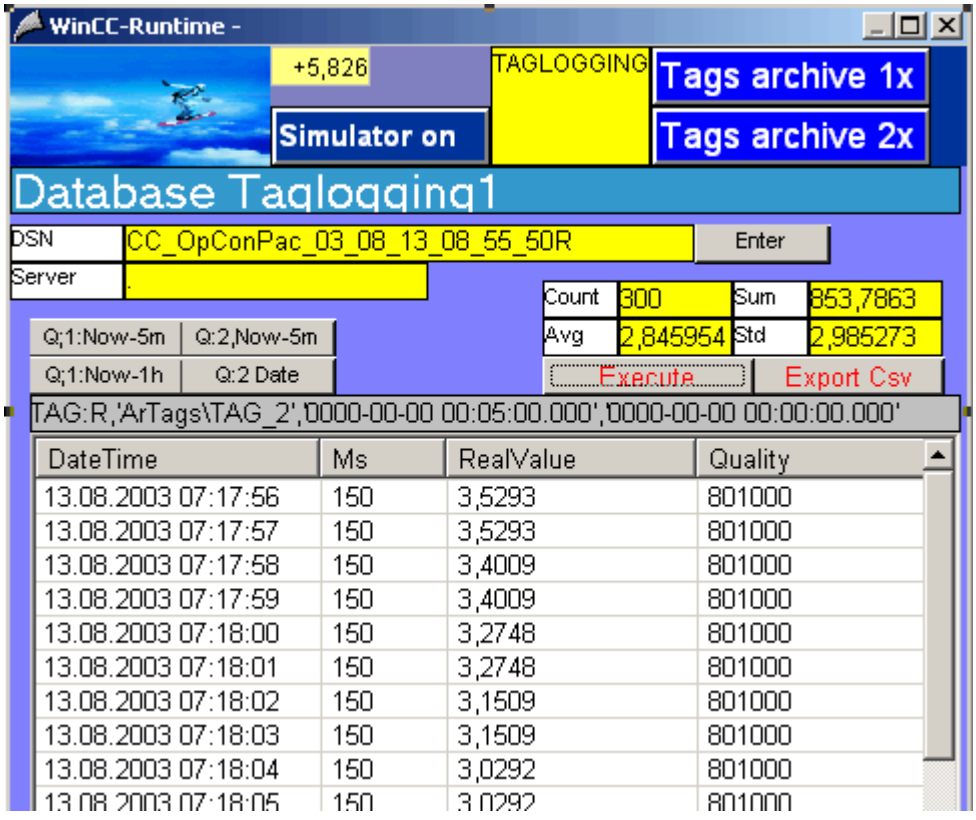

Multiple Return Values to a Query Using a Filter on Tag Value

Example C1:

The following query also uses the <SQL_Clause> parameter and returns all tag values that have the ValueID "3" and "6" and are below 50 or above 100.

```
"TAG:R,(3;6),<TimeBegin>,<TimeEnd>,'WHERE RealValue > 100 OR
RealValue < 50'"
```

Query with parameter <TimeStep>

Example C2:

The following query uses the <TimeStep> parameter and returns all values of ValueID "1" - starting from start time "TimeBegin" till 5 minutes later in intervals of "60" seconds with the aggregation type "5" = "Average value without Interpolation".

```
"TAG:R,1,'2004-10-13 17:00:00.000','0000-00-00 00:05:00.000',
'TIMESTEP=60,5'"
```

The following diagram shows the query result. The left table displays the archive data which were archived in an archiving cycle of 30 seconds. The right table displays the query result. It determines the average between two archive values at "0" seconds and "30" seconds, displayed with the first time stamp of the averaging interval, i.e. second "0".

TIMESTEP = 60,5

ValueID	Datetime	RealValue	Quality
1	2004-10-13 17:00:00.000	3.0	801000
1	2004-10-13 17:00:30.000	4.0	801000
1	2004-10-13 17:01:00.000	2.0	801000
1	2004-10-13 17:01:30.000	4.0	801000
1	2004-10-13 17:02:00.000	4.0	801000
1	2004-10-13 17:02:30.000	1.0	801000
1	2004-10-13 17:03:00.000	3.0	801000
1	2004-10-13 17:03:30.000	3.0	801000
1	2004-10-13 17:04:00.000	1.0	801000
1	2004-10-13 17:04:30.000	0.0	801000
1	2004-10-13 17:05:00.000	3.0	801000

ValueID	Datetime	RealValue	Quality
1	2004-10-13 17:00:00.000	3.5	801000
1	2004-10-13 17:01:00.000	3.0	801000
1	2004-10-13 17:02:00.000	2.5	801000
1	2004-10-13 17:03:00.000	3.0	801000
1	2004-10-13 17:04:00.000	0.5	801000
1	2004-10-13 17:05:00.000	3.0	801000

Example C3:

The following query uses the <TimeStep> parameter and returns all values of ValueID "1" and "2" - starting from start time "TimeBegin" till 2 minutes later in intervals of "15" seconds with the aggregation type "261" = "Average value with linear Interpolation".

```
"TAG:R,(1;2),'2004-10-13 17:00:00.000','0000-00-00 00:02:00.000',
'TIMESTEP=15,261'"
```

The following diagram shows the query result. The left table displays the archive data which were archived in an archiving cycle of 30 seconds. The right table displays the query result. The archive values at "0" and "30" seconds are displayed in the query result unchanged with their time stamp. For second "15," the linear, interpolated value is formed of archive values at seconds "0" and "30". For the "45" second, the linear, interpolated value is taken from the archive values of "30" second of the same minute and the "0" second of the next minute.

TIMESTEP = 15, 261

ValueID	Datetime	RealValue	Quality
1	2004-10-13 17:00:00.000	3,0	801000
2	2004-10-13 17:00:00.000	12,0	801000
1	2004-10-13 17:00:30.000	4,0	801000
2	2004-10-13 17:00:30.000	14,0	801000
1	2004-10-13 17:01:00.000	2,0	801000
2	2004-10-13 17:01:00.000	16,0	801000
1	2004-10-13 17:01:30.000	4,0	801000
2	2004-10-13 17:01:30.000	17,0	801000
1	2004-10-13 17:02:00.000	4,0	801000
2	2004-10-13 17:02:00.000	14,0	801000
1	2004-10-13 17:02:30.000	1,0	801000
2	2004-10-13 17:02:30.000	13,0	801000
1	2004-10-13 17:03:00.000	3,0	801000
2	2004-10-13 17:03:00.000	13,0	801000

ValueID	Datetime	RealValue	Quality
1	2004-10-13 17:00:00.000	3,0	801000
1	2004-10-13 17:00:15.000	3,5	801000
1	2004-10-13 17:00:30.000	4,0	801000
1	2004-10-13 17:00:45.000	3,0	801000
1	2004-10-13 17:01:00.000	2,0	801000
1	2004-10-13 17:01:15.000	3,0	801000
1	2004-10-13 17:01:30.000	4,0	801000
1	2004-10-13 17:01:45.000	4,0	801000
1	2004-10-13 17:02:00.000	4,0	801000
2	2004-10-13 17:00:00.000	12,0	801000
2	2004-10-13 17:00:15.000	13,0	801000
2	2004-10-13 17:00:30.000	14,0	801000
2	2004-10-13 17:00:45.000	15,0	801000
2	2004-10-13 17:01:00.000	16,0	801000
2	2004-10-13 17:01:15.000	16,5	801000
2	2004-10-13 17:01:30.000	17,0	801000
2	2004-10-13 17:01:45.000	15,5	801000
2	2004-10-13 17:02:00.000	14,0	801000

5.4.3 Querying Alarm Message Archives (RT Professional)

Introduction

With the following query, an alarm message archive can be accessed. The data can be selected using filter criteria. The queries are forwarded to the database by the command object.

You will find information about status of messages in the WinCC Information System under "Working with WinCC > ANSI-C Function for Creation of Functions and Actions > ANSI-C Function descriptions > Appendix > Structure Definitions > Structure Definition MSG_RTDATA_STRUCT".

When querying message archives, the result is summarized by archive, but without sorting the queried archive segments. The filter condition needs to be extended accordingly if the segments are to be sorted, e.g., for the chronological sorting "ORDER BY DateTime ASC, MS ASC".

Syntax

```
ALARMVIEW:SELECT * FROM <ViewName>[WHERE <Condition>...., optional]
```

Parameter

Parameter	Description
ViewName	Name of the database table. The table has to be specified in the desired language. The "ViewName" for the five European language is e.g.: ALGVIEWDEU: German alarm message archive data ALGVIEWENU: English alarm message archive data ALGVIEWESP: Spanish alarm message archive data ALGVIEWFRA: French alarm message archive data ALGVIEWITA: Italian alarm message archive data The "ViewName" for the Asian language is e.g.: ALGVIEWCHT: Chinese (simplified) alarm message archive data ALGVIEWCHT: Chinese (traditional) alarm message archive data ALGVIEWJPN: Japanese alarm message archive data ALGVIEWKOR: Korean message archive data Note The languages that are installed in the WinCC base system or that are configured in the WinCC Text Library are supported. Information concerning the possible query-languages or the respective "ViewName" can be found in the SQL-Server in the linked alarm archives under "Views". All languages that are supported in the corresponding archive are shown with their IDs e.g. "GENVIEWENU" here.
Condition	Filter criterion, e.g.: DateTime>'2003-06-01' AND DateTime<'2003-07-01' DateTime>'2003-06-01 17:30:00' MsgNr = 5 MsgNr in (4, 5) State = 2 With DateTime, only absolute time indications can be used.

Example 1:
Reads all entries of message no. 5 that were captured after July 5, 2003.

```
"ALARMVIEW:SELECT * FROM ALGVIEWENU WHERE MsgNr = 5 AND
DateTime>'2003-07-05'"
```

Example 2:
Reads all messages with the time stamp between July 3, 2003 and July 5, 2003.

```
"ALARMVIEW:SELECT * FROM ALGVIEWENU WHERE DateTime>'2003-07-03' AND
DateTime<'2003-07-05'"
```

The following picture shows a possible result of this example. The query was implemented using the Connectivity Pack Demo Project.

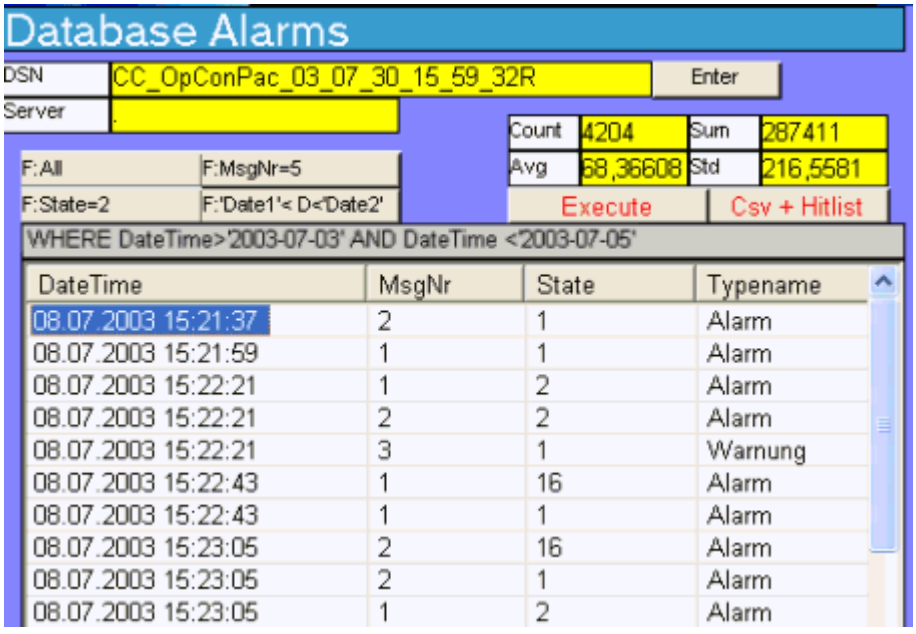

5.4.4 Displaying Alarm Message Archives (RT Professional)

Introduction

The query result is returned as the Recordset. In this chapter, the structure of the Recordset for alarm log archives is described.

You will find information about status of messages in the WinCC Information System under "Working with WinCC > ANSI-C Function for Creation of Functions and Actions > ANSI-C Function descriptions > Appendix > Structure Definitions > Structure Definition MSG_RTDATA_STRUCT".

Recordset Structure

Location	Field name	Type	Comments
1	MsgNo	Integer 4 Bytes	Message number
2	State	Small Integer 2 Bytes	Alarm Log Status
3	DateTime	DateTime 8 Bytes	Time stamp of the message (date/time without milliseconds)
4	Ms	Small Integer 2 Bytes	Time stamp of the message (milliseconds)
5	Instance	VarChar (255)	Instance Name of the Alarm Log
6	Flags1	Integer 4 Bytes	(only for internal use)
7	PValueUsed	Integer 4 Bytes	Process Values used

Location	Field name	Type	Comments
8 to 17	PValue1 to PVal-ue10	Real 8 Bytes	Numerical Process Value 1 to 10
18 to 27	PText1 to PText10	VarChar (255)	Process Value Text 1 to 10
28	ComputerName	VarChar (255)	Name of computer
29	Application	VarChar (255)	Application Name
30	Comment	VarChar (255)	Comments
31	UserName	VarChar (255)	User name
32	Counter	Integer 4 Bytes	Running Alarm Message Counter
33	TimeDiff	Integer 4 Bytes	Time difference to "Came in" status
34	ClassName	VarChar (255)	Name of the message class
35	Typename	VarChar (255)	Name of the message type
36	Class	Small Integer 2 Bytes	Message class ID
37	Type	Small Integer 2 Bytes	Message type ID
38 to 47	Text1 to Text10	VarChar (255)	Message Text 1 to 10
48	AG_NR	Small Integer 2 Bytes	Number of the PLC
49	CPU_NR	Small Integer 2 Bytes	Number of the CPU
50	CrComeFore	Integer 4 Bytes	Foreground Color for the "Came in" Status
51	CrComeBack	Integer 4 Bytes	Background Color for the "Came in" Status
52	CrGoFore	Integer 4 Bytes	Foreground Color for the "Went out" Status
53	CrGoBack	Integer 4 Bytes	Background Color for the "Went out" Status
54	CrAckFore	Integer 4 Bytes	Foreground Color for the "Acknowledged" Sta-tus
55	CrAckBack	Integer 4 Bytes	Background Color for the "Acknowledged" Sta-tus
56	LocalID	Integer 4 Bytes	Location of the Alarm
57	Priority	Integer 4 Bytes	Priority
58	AP_type	Integer 4 Bytes	Loop in Alarm
59	AP_name	VarChar (255)	Loop-in-Alarm Function Name
60	AP_PAR	VarChar (255)	Loop-in-Alarm Screen
61	InfoText	VarChar (255)	Infotext
62	TxtCame	VarChar (255)	Text came in
63	TxtWent	VarChar (255)	Text went out
64	TxtCameNWent	VarChar (255)	Text came in and went out
65	TxtAck	VarChar (255)	Text acknowledged
66	AlarmTag	Integer 4 Bytes	Message tag
67	AckType	Small Integer 2 Bytes	Acknowledgment Type
68	Params	Integer 4 Bytes	Parameter
69	Servername	VarChar (255)	Servername

5.4.5 Query for User Archives (RT Professional)

Introduction

With the following query, you may use MS OLE DB Provider to access WinCC user archives. Access may be read or write enabled in order to analyze the saved data and to modify and save same.

The data can be selected using filter criteria. The queries are forwarded to the database by the command object.

Note

Consider the following when accessing WinCC user archives via the MS OLE DB Provider:

- Ensure that the write access is not enabled simultaneously via the MS OLE DB Provider and WinCC. This prevents inconsistencies in the archives.
- Changes via MS OLE DB Provider will not be displayed in WinCC Runtime until the user archive table controls are selected by a picture change. The current data of the user archives are read again.
- User archives changed via MS OLE DB Provider are not synchronized in a redundant system.
- Note that WinCC updates can cause changes in the database scheme. The scheme can also be changed by the installation of hot fixes and service packs. In this case, you must adapt the read and write access accordingly.

Syntax

Reading of Values

```
SELECT * FROM UA#<ArchiveName>[WHERE <Condition>...., optional]
```

Writing of Values

```
UPDATE UA#<ArchiveName> SET UA#<ArchiveName>.<Column_n> = <Value>
[WHERE <Condition>...., optional]
```

Inserting a Data Set

```
INSERT INTO UA#<ArchiveName> (ID,<Column_1>,<Column_2>,<Column_n>)
VALUES (<ID_Value>, Value_1,Value_2,Value_n)
```

Deleting a Data Set

```
DELETE FROM UA#<ArchiveName> WHERE ID = <ID_Number>
```

Parameter

Parameter	Description
ArchiveName	Name of the user archive.
Condition	Filter criterion e.g.: LastAccess>'2004-06-01' AND LastAccess<'2004-07-01' DateTime>'2004-06-01 17:30:00' ID = 5 ID > 3

Example 1:
Reads all data in the user archive "Test".

```
SELECT * FROM UA#Test
```

Example 2:
Reads all data in the user archive "Test" that were changed between June 1, 2004 and July 1, 2004.

```
SELECT * FROM UA#Test WHERE LastAccess>'2004-06-01' AND
LastAccess<'2004-07-01'
```

Example 3:
Enters the value 'New_String' in the field F_STRING of the ID 3.

```
UPDATE UA#TEST SET F_STRING = 'New_String' WHERE ID = 3
```

Example 4:
Inserts a data set with the ID 100.

```
INSERT INTO UA#Test (ID,F_Integer,F_Float,F_Double,F_String) VALUES
(100.10,'10.0','AAAA')
```

Example 5:
Deletes the data set with the ID 100.

```
DELETE FROM UA#Test WHERE ID = 100
```

5.4.6 Configure Access via the Wizard "SQL Server Import/Export" (RT Professional)

Introduction

WinCC OLE DB Provider may be used to access WinCC databases while employing the Wizard "SQL Server Import/Export. You can save the unzipped files to an intermediate database using the wizard; you access the database with standard SQL queries.

- Using WinCC computers, access to runtime and archive databases may be established locally or remotely.
- In the case of long-term archive servers, local or remote access is only possible to the archive databases since they have no runtime databases.

Procedure

1. Start the "SQL Server Management Studio" and select the desired database.
2. In the shortcut menu of the database select "Tasks > Export Date...".
   The SQL Server Import/Export-Wizard opens.
3. Configure the data source.
   Click "Next". In the field Data source, select the entry WinCC OLE DB-Provider for Archives". Click the button "Properties...". The "Data Link Properties" dialog box opens.
4. Configure the correct Provider settings.
   In the field "Data Source" enter the following text as data source: ".\WinCC"The entry for "Location" remains empty.
   Under "Enter the initial catalog to use" either enter the desired Runtime database or the symbolic name, which was configured in "Archive Connector Tool". The correct spelling of the name may be found in "SQL Server Management Studio" in the "Databases" directory. Alternatively you can also enter the database "CC_ExternalBrowsing" for Runtime data and Archive data.
   Click the "Advanced" tab. For the Property "Connect timeout" select the desired time in seconds. In the property "Access permissions" only select the box "ReadWrite".
   Close the dialog by activating the "OK" button.

   Note

   In order to improve performance during local access, enter " <Computer Name>\WinCC" in the field "Data Source" instead of ".\WinCC".

5. Configure the data source.
   Click "Next". In the field "Destination", select the entry SQL Native Client" for example. The server name can be any SQL Server instance. The database name which you enter in the field "Database", can be any self-created target database.

   Note

   Leave the "Database" field empty. No target tables are then created.

6. Configure the query conditions.
   Click "Next". Select "Write a query to specify the data to transfer". Click "Next". Enter the desired query conditions. With the query "Tag:R,1,'0000-00-00 00:10:00.000','0000-00-00 00:00:00.000" of the ValueID "1", for example, the values of the last ten minutes of the archiving are read. Additional information on the syntax may be found in Chapter "Query for Process Value Archives".
7. End the Wizard and export the data.
   Click "Next" and on the last page of the Wizard click on "Finish". The Wizard executes the data export to the target database. If the data export was successful, the comprised data in the target database are saved in the newly created table"dbo.Query". You can change the table name. If you do not change the table name, data will be overwritten with a new export. New tables with names "Query1", "Query2" etc. are created by the wizard.

5.4.7 Meeting prerequisites for using the Reporting Services (RT Professional)

Introduction

With WinCC you can use the Reporting Services of SQL Server 2014. You can therefore make reports with log data created with Microsoft Visual Studio available in the network.

Requirements

The use of the Reporting Services with WinCC calls for the following additional software requirements:

- Internet Information Services
- Reporting Services of MS SQL Server 2014

Note

Perform the installation steps in exactly the order specified.

Installing the Information Services

1. Select "Settings > Control Panel > Software".
2. Click on the "Add/Remove Windows Components" button on the left.
3. Select the check box "Internet Information Services (IIS)" in the "Windows Components Wizard" dialog.

Installing Reporting Services of MS SQL Server 2014

You install the Reporting Services from the WinCC-DVD or its location in the file system.

1. Select "Settings > Control Panel > Programs and Features".
2. Select "Microsoft SQL Server 2014" and click on "Uninstall/Change".

3. Select "Add" and in WinCC setup the path "InstData > SQL > SQL2014STD > setup".
   The Microsoft SQL Server Installation Wizard is automatically opened. Follow the instructions.
4. Under "Installation Type" select the option "Add features to an existing instance of SQL Server 2014" and the instance "WINCC".
   The management tools and therefore also the SQL Server Data Tools are already installed.
5. Activate "Reporting Services - Native" and follow the instructions.
   The Reporting Services will be installed.
6. Start the Reporting Services Configuration Manager and configure the Reporting Services.

Configuring Internet Information Services

1. Open Computer Management.
2. Under "Services and Applications" select the "Internet Information services (IIS) Manager".
3. Open the feature "Authentication" and select the entry "Edit" in the shortcut menu of "Anonymous authentication".
4. Select the option "Specific user" and enter the user name and password.
   The user name has the following format: <domain or computer name>\<user>

Note

We recommend you to restart your computer following installation.

Result

The requirements for using the Reporting Services are now created. You can now create reports and make them available on the Internet.

5.4.8 Querying the Archive Data (RT Professional)

Introduction

The queries are forwarded to the database by the command object. An important parameter, aside from "ConnectionString", is CommandText. The CommandText transmits the query. The result is returned as the Recordset.

Note

Time range for archive inquiries for message and process values

If the query for message or process value archives specifies a time range for which no messages or other values exist within the archives, no information message or other status display occurs. If this status is to be displayed, error handling must be implemented by the user.

A simple version of this error handling routine is described in the sample script under the topic "Example: Reading message archive data via the WinCC OLE DB Provider".

In the following examples, a command object each is generated and the query transmitted as CommandText.

In the following structure examples, CommandText also includes the ConnectionString whose structure is described under "Establishing Connection to Archive Database".

Structure of CommandText

Process Value Archives:

```
Set oRs = CreateObject("ADODB.Recordset")
Set oCom = CreateObject("ADODB.Command")
oCom.CommandType = 1
Set oCom.ActiveConnection = conn
oCom.CommandText = "TAG:R,'PVArchive\Tag1','0000-00-00
00:10:00.000','0000-00-00 00:00:00.000'"
```

Alarm Message Archives:

```
Set oRs = CreateObject("ADODB.Recordset")
Set oCom = CreateObject("ADODB.Command")
oCom.CommandType = 1
Set oCom.ActiveConnection = conn
oCom.CommandText = "ALARMVIEW:Select * FROM AlgViewEnu"
```

User archives

```
Set oRs = CreateObject("ADODB.Recordset")
Set oCom = CreateObject("ADODB.Command")
```

```
oCom.CommandType = 1
Set oCom.ActiveConnection = conn
oCom.CommandText = "SELECT * FROM UA#Test"
```

Specifying the Recordset location

To specify the location of Recordset for query of the archive data, you need to set the value "3" for the "CursorLocation" property, for example, "conn.CursorLocation = 3". The Recordset is created on the client.

5.4.9 Displaying User Archives (RT Professional)

Introduction

Each user archive consists of data fields with editable properties. Each data field has properties such as name, alias name, type, lengths, value etc. The representation of the data fields and properties in the Editor User Archives is done in lines and columns. Therefore, we are talking of rows instead of data fields and of columns instead of properties.

In the following, the user archive "Test" is described as a structure example. This user archive is included in the Connectivity Pack Demo Project "OPConPack" in directory "\Samples \Connectivity Pack\DemoProject."

Structure of User Archive "Test"

Field Name	Type	Comment
ID	Integer	Unique identification of Value.
F_Integer	Integer	Example for Value
F_Float	Float	Example for Value
F_Double	Double	Example for Value
F_String	String	Sample character sequence

5.5 Analysis functions for messages and process values (RT Professional)

5.5.1 Analysis Functions for Messages and Process Values (RT Professional)

Introduction

Different analysis functions are available with WinCC for querying archived messages and process values.

The analysis is triggered by a query with parameters for different aggregate functions. Calculation of the aggregate function is performed on the Connectivity Pack server, and only the result is transferred to the client.

Additional information on CommandText and ConnectionString which are used in the following examples may be found under "Establishing Connection to Archive Database" and "Querying Archive Data".

Analysis Functions for Messages

The analysis query for alarm logs archives returns a specific recordset which contains configuration and runtime data for each message as well as results of the aggregate functions.

The returned recordset for analysis queries of alarm logs is not identical to the recordset of normal queries of message archives. Additional information may be found in Chapter "Display of Alarm Logs for Analysis Queries".

For each message, the following aggregate functions are calculated. The column descriptions of the result list are placed in parenthesis.

- Sum of message frequency ("FreqOfAlarm")
- Cumulative duration from "Message Incoming" until "Message Outgoing" ("CumDurationComeGo")
- Average duration from "Message Incoming" until "Message Outgoing" ("AvDurationComeGo")
- Cumulative duration from "Message Incoming" until initial acknowledgment ("CumDurationComeAckn1")
- Average duration from "Message Incoming" until initial acknowledgment ("AvDurationComeAckn1")
- Cumulative duration from "Message Incoming" until second acknowledgment ("CumDurationComeAckn2")
- Average duration from "Message Incoming" until second acknowledgment ("AvDurationComeAckn2")
- Cumulative duration from "Message Incoming" until "Message Incoming" ("CumDurationComeGo")
- Average duration from "Message Incoming" until "Message Incoming" ("AvDurationComeCome")

Syntax

For the calculation of aggregate functions for messages, the following command is issued to WinCC OLE DB Provider.

```
"AlarmHitView: SELECT * FROM <ViewName>[WHERE <Condition>]"
```

Here:
<ViewName> = Name of the database table in the desired language, e.g. ALGVIEWMENU for English.
[WHERE <Condition>] = optional filter criterion as WHERE condition in the SQL syntax.

Additional information on the syntax may be found in Chapter "Query for Alarm Logs".

Example

The example provides results of the aggregate functions for all messages for the time range between 7/15/2004 12:00 p.m. and 12:15 p.m. from the "ALGVIEWENU" database.

ConnectionString:

```
"Provider=WinCCOLEDBProvider.
1;Catalog=CC_OpenArch_03_05_27_14_11_46R;Data Source=.\WinCC"
```

CommandText:

```
"AlarmHitView: SELECT * FROM ALGVIEWENU WHERE DateTime>'2004-07-15
12:00:00' AND DateTime<'2004-07-15 12:15:00'"
```

Analysis Functions for Process Values

The analysis of process values returns the result of an aggregate function. Only one aggregate function can be calculated in a query.

The following aggregate functions are available for process values.

- MIN (minimum)
- MAX (maximum)
- AVG (average)
- SUM (sum of all values)
- COUNT (count of process values)
- COUNTER (number of entries with value "1", e.g., query of binary tags)
- STDEV (statistical standard deviation)
- VAR (statistical variance)

Syntax

For the calculation of aggregate functions for process values, a query is issued to the MS SQL OLE DB Provider and the procedure "cp_TagStatistic" from database "SQL Server Master" is executed.

Note

The analysis functions for process values in transparent access function only with Connectivity Station on a client with own project.

The following parameters are transferred to the "cp_TagStatistic" procedure.

```
cp_TagStatistic @P1,@P2,@P3[,@P4]
```

Where:
"@P1" = database name (e.g. WinCC Runtime database or symbolic name of the directory with the swapped out archives). For transparent access, use the WinCC project name instead of the database name.
"@P2" = WinCC OLE DB-Provider String for process values.
"@P3" = desired aggregate function.
"@P4" = <Symbolic computer name>::\WinCC (required only for transparent access).

Additional information on the syntax of parameters "@P1" and "@P2" may be found in Chapter "Query for Process Value Archives".

Note

Analysis functions for process values with Asian archive tag names

If you use archive tag names with Asian character sets to calculate an analysis function, the request for Unicode character sets has to be adapted.

Add the prefix "N'" in front of both parameters.

Example: cp_TagStatistic N'TestDB',N'TAG:R,17,"2004-05-17 12:00:00","2004-05-17 13:00:00'",'AVG'

Example

This query will return the average of process values in the time range between 5/17/2004 12:00 and 13:00 for ValueID "17" from database "TestDB".

ConnectionString:

```
"Provider=SQLNCLI11;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=master ;Data Source=.\WinCC"
```

CommandText:

```
"cp_TagStatistic 'TestDB','TAG:R,17,''2004-05-17
12:00:00'',''2004-05-17 13:00:00''','AVG'"
```

Example of transparent access

This query delivers the average process value in the time range between 14.09.2006 10:00 hrs and 11:00 hrs for the ValueID "7" from the "WinCCProj".

ConnectionString:

```
"Provider=SQLNCLI11;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=master ;Data Source=.\WinCC"
```

CommandText:

```
"cp_TagStatistic 'WinCCProj','TAG:R,7,''2006-09-14
10:00:00'',''2006-09-14 11:00:00''','AVG','Symb_WinCCProj::\WinCC'"
```

5.5.2 Display of Message Archives for Analysis Queries (RT Professional)

Introduction

The analysis query for message archives returns a specific recordset which contains configuration and runtime data for each message as well as results of the aggregate functions. This Recordset is not identical to the Recordset of normal queries of message archives.

Recordset Structure for Analysis of Message Archives

Upon query of message archives using the analysis function "AlarmHitView", the result is returned as recordset with the following structure.

Location	Field name	Type	Comments
1	MsgNo	Integer 4 Bytes	Message number
2	State	Small Integer 2 Bytes	Alarm Log Status
3	DateTime	DateTime 8 Bytes	Time stamp of the message (date/time without milliseconds)
4	Ms	Small Integer 2 Bytes	Time stamp of the message (milliseconds)
5	Instance	VarChar (255)	Instance Name of the Alarm Log
6	Flags1	Integer 4 Bytes	(only for internal use)
7	Counter	Integer 4 Bytes	Running Alarm Message Counter
8	TimeDiff	Integer 4 Bytes	Time difference to "Came in" status
9	ClassName	VarChar (255)	Name of the message class.
10	Typename	VarChar (255)	Name of the message type.
11	Class	Small Integer 2 Bytes	Message class ID
12	Type	Small Integer 2 Bytes	Message type ID
13 to 22	Text1 to Text10	VarChar (255)	Message Text 1 to 10
23	AG_NR	Small Integer 2 Bytes	Number of the PLC
24	CPU_NR	Small Integer 2 Bytes	Number of the CPU
25	CrComeFore	Integer 4 Bytes	Foreground Color for the "Came in" Status
26	CrComeBack	Integer 4 Bytes	Background Color for the "Came in" Status

Location	Field name	Type	Comments
27	CrGoFore	Integer 4 Bytes	Foreground Color for the "Went out" Status
28	CrGoBack	Integer 4 Bytes	Background Color for the "Went out" Status
29	CrAckFore	Integer 4 Bytes	Foreground Color for the "Acknowledged" Status
30	CrAckBack	Integer 4 Bytes	Background Color for the "Acknowledged" Status
31	Priority	Integer 4 Bytes	Priority
32	AP_type	Integer 4 Bytes	Loop in Alarm
33	AP_name	VarChar (255)	Loop-in-Alarm Function Name
34	AP_PAR	VarChar (255)	Loop-in-Alarm Screen
35	InfoText	VarChar (255)	Infotext
36	TxtCame	VarChar (255)	Text came in
37	TxtWent	VarChar (255)	Text went out
38	TxtCameNWent	VarChar (255)	Text came in and went out
39	TxtAck	VarChar (255)	Text acknowledged
40	AckType	Small Integer 2 Bytes	Acknowledgment Type
41	FreqOfAlarm	Integer 4 Bytes	Sum of message frequency
42	CumDurationComeGo	Integer 4 Bytes	Cumulative duration from "Message Came In" until "Message Went Out"
43	AvDurationComeGo	Real 8 Bytes	Average duration from "Message Came In" until "Message Went Out"
44	CumDurationCo-meAckn1	Integer 4 Bytes	Cumulative duration from "Message Came In" until initial acknowledgment
45	AvDurationComeAckn1	Real 8 Bytes	Average duration from "Message Came In" until initial acknowledgment
46	CumDurationCo-meAckn2	Integer 4 Bytes	Cumulative duration from "Message Came In" until second acknowledgment
47	AvDurationComeAckn2	Real 8 Bytes	Average duration from "Message Came In" until second acknowledgment
48	CumDuration Come-Come	Integer 4 Bytes	Cumulative duration from "Message Came In" until "Message Came In"
49	AvDurationComeCome	Real 8 Bytes	Average duration from "Message Came In" until "Message Came In"

5.6 Bases of OLE DB (RT Professional)

Introduction

Using the OLE DB interface and the associated database provider supplied by WinCC, you have access to process value and message archives.

OLE DB

OLE DB is an open standard for a fast access to different databases. It is irrelevant whether the database is relational or not.

The connection between the OLE DB level and the database is established through a database provider.

OLE DB interfaces and providers are offered from various manufacturers.

WinCC OLE DB Provider

Using WinCC OLE DB Provider, you may directly access WinCC archive data stored in the MS SQL server database. Depending on the configuration, process data of WinCC are stored in compressed form. WinCC OLE DB Provider permits transparent access even to these data.

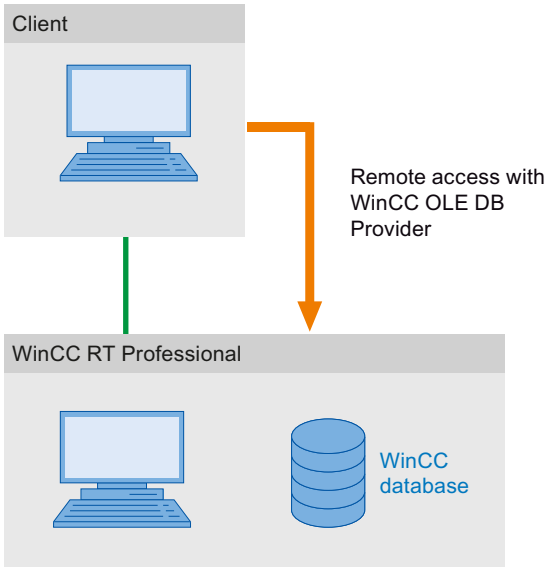

Microsoft OLE DB

Microsoft OLE DB only provides access to WinCC user archives.

Access to WinCC User Archives using Microsoft OLE DB requires a Connectivity Pack license on the computer where the WinCC user archives will be accessed. A WinCC Client Access License is required for remote access to the MS SQL Server database.

As protection from unauthorized access using MS OLE DB, the administrator of the databases can take appropriate actions. Additional information may be found in Chapter "Security Settings for Access to SQL Databases Using MS OLE DB".

Note

Microsoft OLE DB is only tested and released for access to WinCC user archives but not to alarm and process value archives.
Use the WinCC OLE DB Provider to access message and process value archives.

5.7 WinCC OLE DB Provider (RT Professional)

Introduction

The WinCC OLE DB Provider makes access to the process value and alarm archives possible.

The WinCC OLE DB Provider also provides analysis functions, such as minimum, maximum of archive tags.

Access to archive data

Access to archive data can take place under different constellations:

Simultaneous access to archive and Runtime databases

The function "ArchiveMonitor" combines the data of the Runtime and archive databases of the activated WinCC project in an SQL database.

The SQL database is called "CC_ExternalBrowsing" and creates the views "AMT" and "ARCHIVE".

The views provide all necessary information for the WinCC OLE DB Provider.

Because these views provide all the necessary information for the WinCC OLE DB Provider, the "CC_ExternalBrowsing" database can be used as a data source for the provider.

On deactivating WinCC runtime, views additionally created in "CC_ExternalBrowsing" are deleted once again.

Because a long-term archive server does not contain any runtime databases, access to the archive databases using the "CC_ExternalBrowsing" database is not possible.

WinCC OLE DB Provider as a "Linked Server"

Note

When configuring WinCC OLE DB Provider as a "Linked Server", the "WinCC OLE DB-Provider for Archives" entry must be selected in the "Provider Name" field of the ""Linked Server Properties" dialog.

Activate the "AllowInProcess" check box in the "Provider Options" dialog.
Additional information is available here: "SQL Server Books Online" under "Configuring OLE DB Providers for Distributed Queries".

The WinCC OLE DB Provider can be registered in the SQL server in the list of "Linked Servers". Regardless if the database is stored locally or remote .

Example:

WinCC OLE DB Provider as a Linked Server with the server name "WinCC":

```
select min(realvalue) from openquery(WINCC,'Tag:R,1,''0000-00-00
00:01:00.000'',''0000-00-00 00:00:00.000''')
select * from openquery(WINCC,'Tag:R,1,''0000-00-00
00:01:00.000'',''0000-00-00 00:00:00.000''')
```

5.8 Installation (RT Professional)

Installation of WinCC OLE DB-Provider

The WinCC OLE DB-Provider is automatically installed along with WinCC.

The WinCC OLE DB Provider is available for the following systems after installation of WinCC:

- Engineering station with WinCC
- HMI device with WinCC Runtime Professional

Working with recipes 6

6.1 Working with recipes

6.1.1 Basics

6.1.1.1 Definition and applications

Introduction

Recipes are collections of related data, for example, machine parameterizations or production data.

Examples:

- Machine parameter settings that are needed to convert production to a different product variant.
- Product components that result in different compositions for different end products.

A recipe has a fixed data structure. The structure of a recipe is defined during configuration. A recipe contains recipe data records. These differ in terms of their values, but not their structure.

Recipes are stored on the HMI device or on an external storage medium. Recipe data records are always transferred complete and in a single pass between the HMI device and the PLC. In addition, production data can be imported in runtime, in the form of a CSV file.

Note

Restrictions for Import / Export

It is not possible to export or import the recipes for the following HMI devices:

- Basic Panels

Complete recipe data, but not single recipe data records, can be exported and imported using ProSave in CSV format and transferred to the HMI device. Thereby, Runtime is interrupted.

Using recipes

Recipes can be used in the following situations:

- Manual production
  You select the required recipe data and display it on the HMI device. You modify the recipe data as required and save it on the HMI device. You transfer the recipe data to the PLC.
- Automatic production
  The control program starts transfer of the recipe data between the PLC and HMI device. You can also start the transfer from the HMI device. Production is then implemented automatically. It is not essential to display or modify the data.
- Teach-in mode
  You optimize production data that was optimized manually on the system, e.g. axis positions or filling volumes. The values thus determined are transferred to the HMI device and saved in a recipe data record. You can then transfer the saved recipe data back to the PLC at a later date.

Entering and modifying the recipe data

You enter the data in the individual recipe data records and modify it as required. The following options are available:

- Data entry during configuration
  If the production data exists already, you enter the data in the "Recipes" editor during recipe configuration.
- Entering the data in Runtime
  If you have to frequently modify production data, you can do this directly in Runtime as follows:
  - Enter the data directly on the HMI device.
  - Set the parameters directly on the machine. You then transfer the data from the PLC to the HMI device and save it in the recipe.

6.1.1.2 Examples for using recipes

Recipes are used in the manufacturing industry and mechanical engineering, for example. The following recipes show typical applications which you can implement with the recipe function of WinCC:

- Machine parameter assignment
  One field of application for recipes is the assignment of machine parameters in the manufacturing industry: A machine cuts wooden boards to a certain size and drills holes. The guide rails and drill have to be moved to new positions according to the board size. The required position data are stored as data records in a recipe. You reassign the machine parameters using "Teach in" mode if, for example, a new board size is to be processed. You transfer the new position data directly from the PLC to the HMI device and save it as a new data record.
- Batch production
  Batch production in the food processing industry represents another field of application for recipes: A filling station in a fruit juice plant produces juice, nectar, and fruit drinks in a variety of flavors. The ingredients are always the same, differing only in their mixing ratios. Each flavor corresponds to a recipe. Each mixing ratio corresponds to a data record. All of the required data for a mixing ratio can be transferred to the machine control at the touch of a button.

6.1.1.3 Recipe structure

Introduction

The basic structure of a recipe is illustrated with reference to the filling station in a fruit juice plant.

There may be several different recipes in an HMI device. A recipe can be compared to an index card box that contains several index cards. The index card box contains several variants for manufacturing a product family. All the data for each manufacturing variant is contained on a single index card.

Example:

In a soft drinks production plant, a recipe is needed for different flavors. Drink variants include fruit juice drink, juice and nectar.

Recipe

The recipe contains all the recipe data records for the different drink variants.

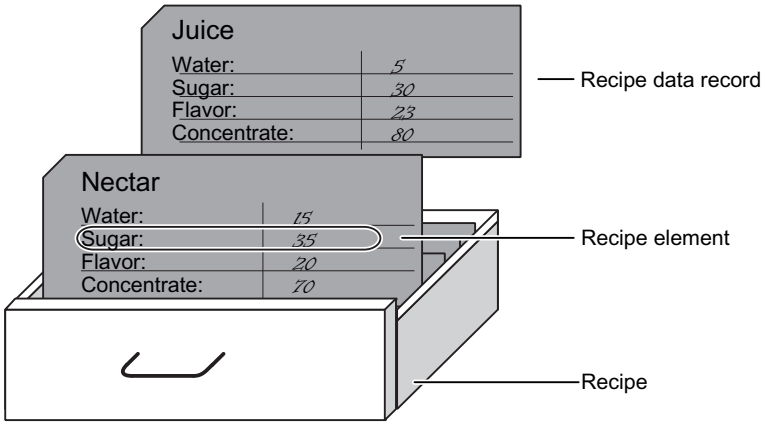

Recipe data records

Each index card represents a recipe data record needed to manufacture a product variant.

Recipe entries

Each index card in a drawer has the same structure. All the index cards contain fields for the different ingredients. Each field corresponds to a recipe entry. All the records of a recipe thus contain the same entries. The records differ, however, in the value of the individual entries.

Example:

All the drinks contain the following ingredients:

- Water
- Concentrate
- Sugar
- Flavoring

The records for juice drink, fruit juice or nectar differ, however, in the quantity of sugar used in production.

6.1.1.4 Display of recipes

Introduction

You can display recipes in the following ways:

- Recipe view
- Recipe screen

Input in the recipe view and recipe screen

You change the values of a recipe in the recipe screen or recipe view and thus modify the manufacturing process or a machine.

The recipe view and recipe screen can perform the same functions when using recipes. They differ in the following respects:

- Display options
- Operation
- Possibility of transferring data between the PLC and the HMI device

Recipe view

The recipe view is suitable for viewing simple recipes.

The recipe view is an off-the-shelf WinCC display and operator control object for managing recipe data records. The recipe view is always part of a screen. The recipe view shows recipe data records in tabular form. You adapt the appearance and the possible operations to suit your specific needs.

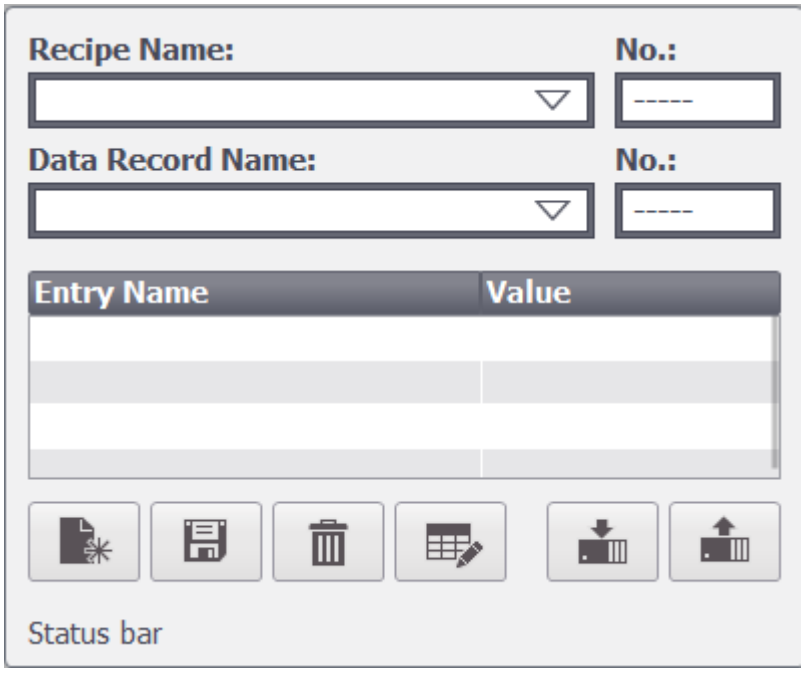

If you are editing recipes with a recipe view in your project, the values are saved in recipe data records. The values are not transferred between the HMI device and PLC until you use the relevant operator control object.

Recipe screen

The recipe screen is an individual screen that contains:

- Input fields for recipe variables
- Operator control objects for using the recipes, e.g. "SaveDataRecord"

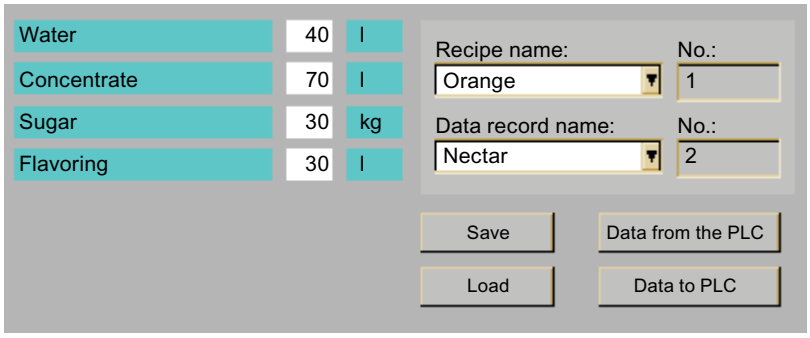

The recipe screen is suitable in the following situations:

- Large recipes
- Allocation of recipe fields to the graphic display of the relevant plant area
- Breakdown of the recipe data into several screens

Note

The values of recipe variables are transferred between the PLC and recipe screen at the following times depending on the configuration:

- Immediately after modification
- When a relevant operator control object is used

Synchronizing recipe view and recipe screen

There may be differences in Runtime between the values displayed in the recipe view and the values saved in the associated tags when you edit recipes in both a recipe view and a recipe screen. To prevent this, you must synchronize the recipe data record values with the values of the recipe tags.

A complete recipe data record will always be saved or synchronized.

Note

Recipe tags can only be synchronized in the advanced recipe view. Synchronization only takes place if the "Synchronize recipe view and recipe tags" setting is activated for the recipe.

6.1.1.5 Transferring recipe data records

Flow of data in recipes

The following picture illustrates the flow of data in recipes:

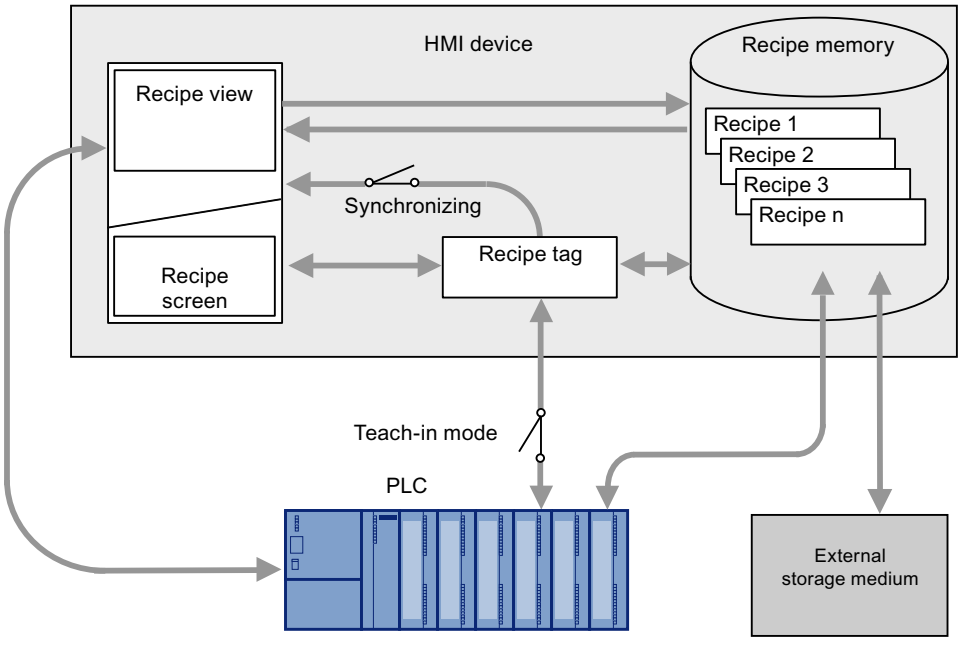

Interaction between the components

There is interaction between the following components at runtime:

- Recipe view / recipe screen
  On the HMI device, recipes are displayed and edited in the recipe view or in a recipe screen:
  - The recipe data records from the internal memory of the HMI device are displayed and edited in the recipe view.
  - The values of the recipe tags are displayed and edited in the recipe screen.

  You synchronize the values displayed in the recipe view with the values of recipe tags according to the configuration.
- HMI device recipe memory
  Recipes are saved in the form of recipe data records in the HMI device's recipe memory.
- Recipe tags
  The recipe tags contain recipe data. When you edit recipes in a recipe screen, the recipe values are stored in recipe tags. The point at which the values of the recipe tags are exchanged with the PLC depends on the configuration.

Note

You can synchronize the recipe tags with the recipe data records so that the same values are saved in both.

Loading and saving recipe data

Complete recipe data records are loaded from or saved to the recipe memory on the HMI device in the recipe view.

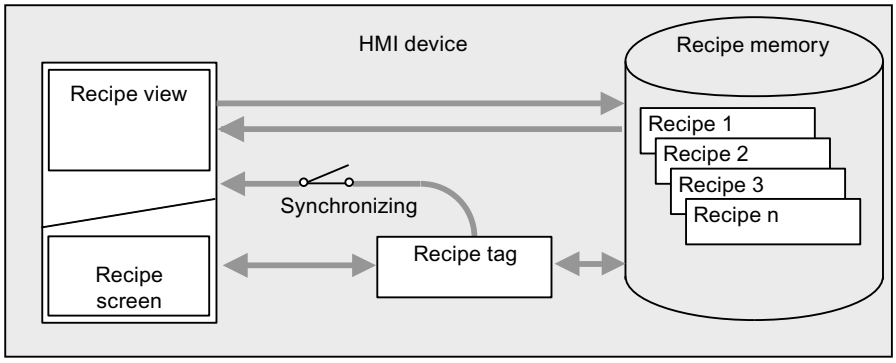

The values of the recipe data record are loaded from the recipe memory to the recipe tags in the recipe screen. When they are saved, the values of the recipe tags are saved to a recipe data record in the recipe memory.

Transferring the recipe values between the HMI device and the PLC

Complete recipe data records are transferred between the recipe view and PLC.

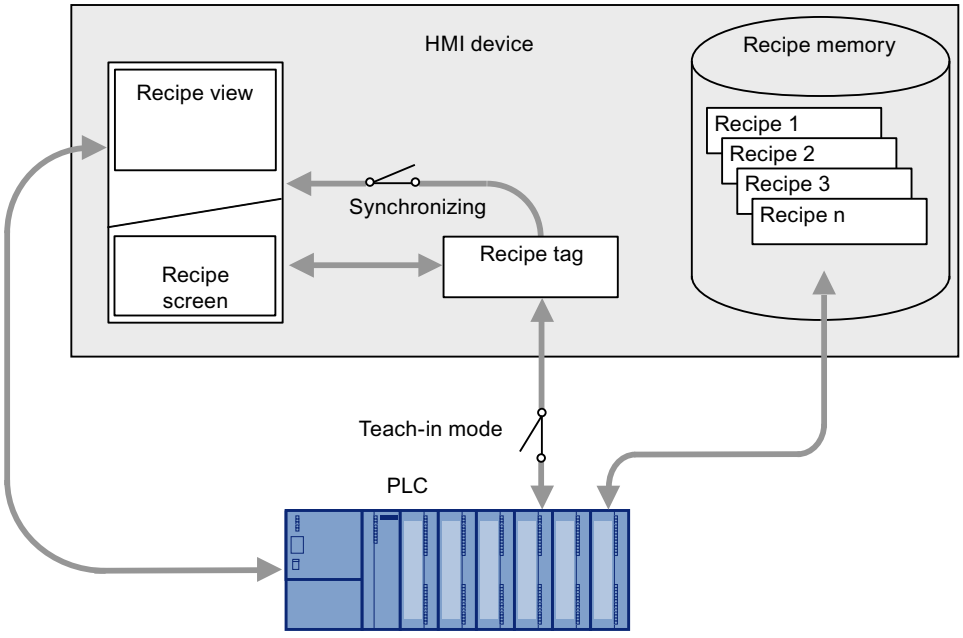

The following transfers are possible between the recipe screen and PLC, depending on the configuration:

- Transferring recipe data records between recipe view and recipe tags
- Immediate transfer of individual modified values between the recipe tags and PLC. The following settings are needed in the recipe in order to do this:
  - "Synchronizing recipe view and recipe tags" is activated.
  - "Manual transfer of individual modified values (teach-in mode)" is deactivated.

Recipe data records can be transferred directly between the HMI device and PLC. In these situations, the display on the HMI device is not essential.

Exporting and importing recipe data records

Recipe data records are exported from the HMI device recipe memory and are saved to a CSV file on the external storage medium. The records can be reimported from the storage medium to the recipe memory.

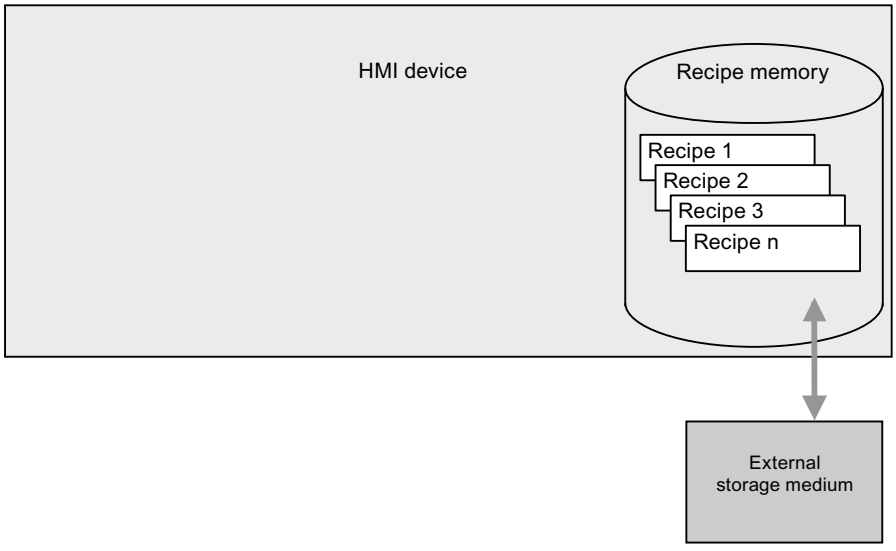

The following external storage media may be used, depending on the HMI device:

- No storage medium, for example, for basic panels
- Memory card
- USB stick
- Hard disk

6.1.1.6 Configuration of recipes

Introduction

Recipes are configured differently according to the intended use:

- If you are editing recipes with a recipe view in your project, the values are only saved in recipe data records.
- If you are editing recipes in a recipe screen in your project, the values are saved in recipe tags.

The following possible settings determine how the recipe data records, recipe tags and PLC all interact.

"Synchronizing recipe view and recipe tags" deactivated

The data in a data record is only displayed and can only be edited in the recipe view. If you use data outside the recipe view, they are not synchronized with the recipe view.

"Synchronizing recipe view and recipe tags" activated

There may be differences in Runtime between the values displayed in the recipe view and the values saved in the associated tags when you edit recipes in both a recipe view and a recipe screen. To prevent this, you must synchronize the recipe data record values with the values of the recipe tags.

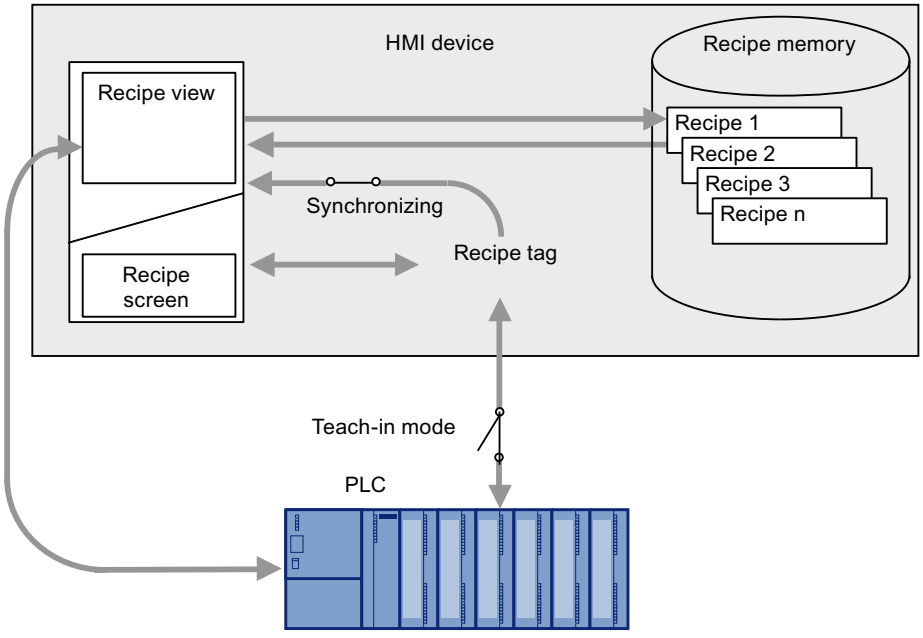

Note

Recipe tags can only be synchronized in the advanced recipe view.

The values of the recipe view and the associated recipe tags are not synchronized automatically. The recipe tags and the recipe view are not synchronized until you use the operating element with the "RecipeViewSynchronizeDataRecordWithTags" function.

"Synchronize recipe view and recipe tags" activated and "Manual transfer of individual modified values (teach-in mode)" activated

With this setting, modified recipe values are not synchronized immediately between the recipe tags in the recipe screen of the HMI device,and PLC.

There must be an operating element with the "SetDataRecordToPLC" and "GetDataRecordFromPLC" functions present in order to synchronize the values.

If recipe values are changed in the controller, the modified values are displayed immediately in the recipe screen if you use the operating element with the "GetDataRecordFromPLC" function.

"Synchronize recipe view and recipe tags" activated and "Manual transfer of individual modified values (teach-in mode)" deactivated

With this setting, modified recipe values are synchronized immediately between the recipe tags on the HMI device and PLC.

When you change recipe values in the recipe screen, these changes are applied immediately by the PLC and immediately influence the process.

If recipe values are changed in the PLC, the changed values are displayed immediately in the recipe screen.

6.1.1.7 Special features of some devices

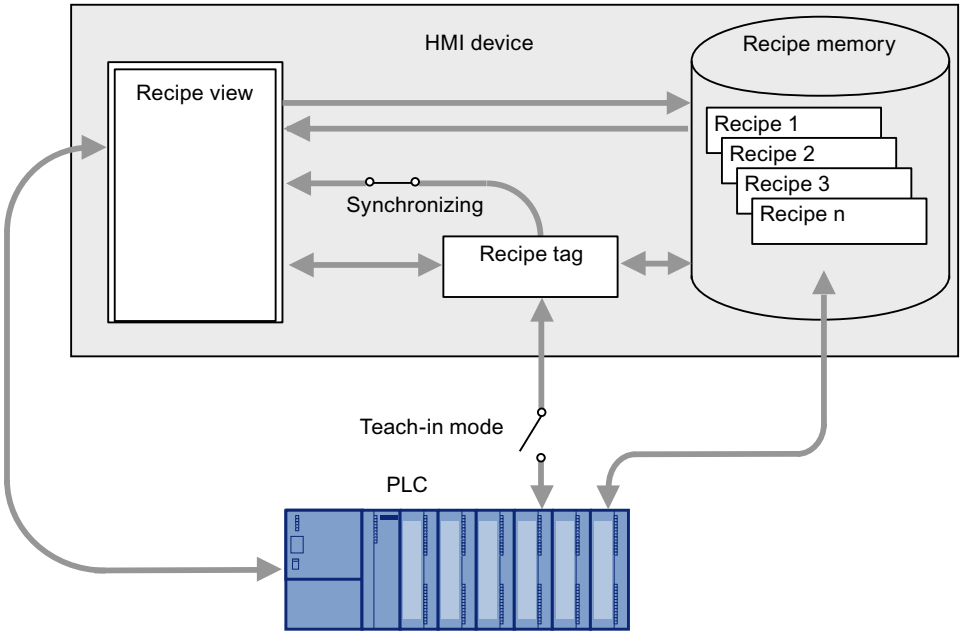

Basic Panels behave differently to other HMI devices in the following respects:

- Only the simple recipe view is supported.
- Recipe tags are always synchronized with the recipe view (see figure above): When the operator changes a value of the recipe element in recipe view, the value of the associated recipe tag will also change.
- Recipe tags are not automatically transferred between the PLC and HMI device when they are modified. Recipe tags are always transferred manually in teach-in mode.
  There must be an operator control object with the "SetDataRecordToPLC" and "GetDataRecordFromPLC" functions present in order to synchronize the values with the PLC.

Note

Restrictions for recipe tags

Generally, recipe tags cannot be used in Basic Panels, except in a recipe.

Note

Restrictions for Import / Export

It is not possible to export or import the recipes for the following HMI devices:

- Basic Panels

Complete recipe data, but not single recipe data records, can be exported and imported using ProSave in CSV format and transferred to the HMI device. Thereby, Runtime is interrupted.

NOTICE

Data loss with several recipe views in the screen

Only applies for Basic Panels: If two or more recipe views show the same recipe in a screen, you have a conflict when accessing the data.

The result is data loss and unpredictable status of recipe data.

Make sure the operators do not select and edit the same recipe in different recipe views.

- Display only one recipe in a recipe view.
- Display a different recipe in each recipe view.

Note

Saving recipes on Comfort Panels

Recipes should be preferably saved on the SD card rather than in the built-in flash memory.

Interaction between the components

There is interaction between the following components at runtime:

- Recipe view
  Recipes are displayed and edited in the recipe view on the HMI device.
  The recipe data records from the internal memory of the HMI device are displayed and edited in the recipe view.
- HMI device recipe memory
  Recipes are saved in the form of recipe data records in the HMI device's recipe memory.
- Recipe tags
  The recipe tags contain the values of recipe elements.

6.1.1.8 Synchronization of recipe data records with the PLC

Overview

When recipe data records are transferred between the HMI device and PLC, both communication peers access common communication areas on the other peer.

Recipe data records are always transferred directly. The values of the tags are written directly to or read directly from the configured addresses without being placed on the clipboard.

Data transfer types

There are two ways to transfer recipe data records between the HMI device and PLC:

- Transfer without coordination
- Coordinated transfer via the "Data record" area pointer.

Note

Coordinated transfer

Transfer with coordinated transfer is used to prevent the uncontrolled overwriting of data in either direction in your control program.

Requirements for coordinated transfer

The following requirements apply to coordinated transfer:

- The "Data record" area pointer must be set up for the required connection in the "Communication > Connections" editor.
- In the properties of the recipe "Coordinated transfer of data records" is activated.
- The connection to the PLC is specified in the properties of the recipe with which the HMI device coordinates the transfer.

Coordinated transfer

In the case of coordinated transfer, both the PLC and the HMI device set the status bits in the shared data compartment.

Coordinated transfer of recipe data records can be a useful solution in the following cases:

- The PLC is the "active partner" for the transfer of recipe data records.
- The PLC evaluates information about the recipe number and name, as well as the recipe data record number and name.
- The transfer of recipe data records is started by the following PLC jobs:
  - "Set_data_record_in_PLC"
  - "Get_data_record_from_PLC"

6.1.1.9 "Recipes" editor

Introduction

You can create, configure and edit recipes, recipe entries and recipe data records in the "Recipes" editor. The "Recipes" editor also allows you to enter values in recipe data records.

Structure of the "Recipes" editor

You create recipes in the top part of the table editor. You can also configure them there or in the Inspector window.

The bottom part of the table editor has the following tabs:

- Elements
  Define the recipe elements of the selected recipe using the table cells provided here. You can move recipe elements within the table with the shortcut menu commands, "Up" and "Down".
- Data records
  Define the values of the data records of the selected recipe using the table cells provided here.

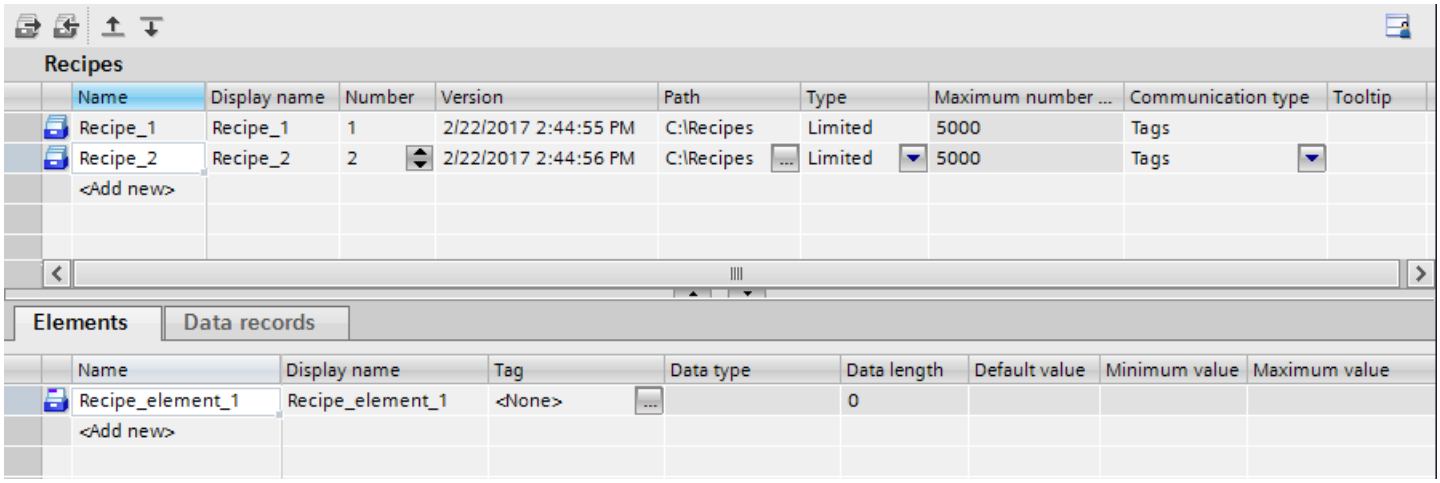

You can then configure the selected recipe, the recipe element or the recipe data record in the Inspector window. You will find further notes on configuring the components of a recipe under "Configuring Recipes".

Recipe settings

The following settings are available for recipes:

Setting	Description
Name of the recipe	This is a unique identification for the recipe within the HMI device.
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign descriptive names or designations which the operator can associate directly with a recipe, e.g. "fruit juice drink".
Recipe number	This is a unique identification for the recipe within the HMI device.
Version	Information about the recipe. The date and time of the last change to the recipe is set by default.
Path	Defines the storage location for recipes. The recipes are stored as a file.
Size type [fixed]	The recipe data records are limited to a predetermined number by default.
Number of data records [fixed]	Maximum number of data records in a recipe in Runtime. The number is limited by the recipe memory of the HMI device.
Communication type [fixed]	The recipe data records are written directly to the addresses of the recipe tags and read from there.
Tooltip	Tooltip for the recipe which is shown to the operator in Runtime.

Note

Path

The storage location depends on the storage media available on the HMI device.

Basic Panels

These HMI devices have no external memory. Recipes are always saved in the internal Flash memory. The "Path" setting is therefore not available.

Recipe element settings

You can make the following settings on the "Elements" tab:

Setting	Description
Name of the recipe element	Identifies a recipe element uniquely within the recipe. Enter meaningful names or labels that you can allocate uniquely, such as axis labels on a machine or ingredients such as "Flavoring".
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign meaningful names or designations which the operator can associate directly, e.g. "fruit juice flavoring".
Recipe tag	An assigned tag in Runtime stores the current value of the recipe element in the recipe data record.
Data type	Data type of the recipe tag.

Setting	Description
Data length [fixed]	Data length of the recipe tag, depending on the data type.
Text list	Text is assigned to a value or range of values in a text list. You can display this text in an output field, for example. The assigned recipe tag must have the data type of a number. The tag value must be within the range of values of the text list.
Default value	This is used as the default entry when you create a new recipe data record.
Minimum value [fixed]	The smallest representable value of a number-based recipe tag, depending on the type of data.
Maximum value [fixed]	The largest representable value of a number-based recipe tag, depending on the type of data.
Decimal places	Determines how many places a decimal number is rounded to, e.g. 3 decimal places and vice versa by what power of ten an integer value is multiplied, e.g. 1,000.
Tooltip	Tooltip about the recipe element which is shown to the operator in Runtime.

Recipe data record settings

You can make the following settings on the "Data records" tab:

Setting	Description
Name of the recipe data record	Identifies a recipe data record uniquely within the recipe.
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign meaningful names or product numbers which the operator can associate directly with a product, e.g. "yellow fruit juice E231".
Recipe data record number	Identifies a recipe data record uniquely within the recipe.
Recipe elements 1 to n	You can store various values for each recipe element even during configuration. Together with the values of the other recipe elements, a value always forms a recipe data record. You can store multiple recipe data records. If enabled in the transfer settings, the recipe data records are transferred to the HMI device when downloading the project and existing data records on the HMI device are overwritten.
Comment	Comment about the recipe data record

6.1.2 Recipes in runtime

6.1.2.1 Recipe screen and recipe view

Basics on the recipe screen

Introduction

The recipe screen contains an individual screen form for entering the recipes. The screen form contains I/O fields and other display and operator control objects. The recipe functionality is implemented with system functions, e.g. for saving recipe data records.

The picture below contains an example of a recipe screen:

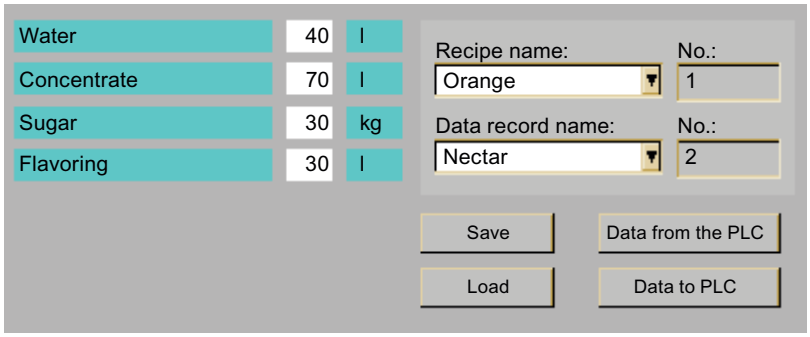

Note

Availability

Basic Panels and OP73A, OP77A, TP177A (Portrait) do not support any recipe screens.

Designing recipe screens

Configuring a recipe screen allows you to customize the display. You can spread large recipes over several screens according to topic and display them clearly using features such as graphical display and operator control objects.

- Spreading recipes over several screens according to topic
  - You can spread recipe data records containing lots of entries across several screens. For example, for each plant area you can configure a screen containing the associated screen forms for the recipe data records.

  Spreading recipes over several screens is useful for HMI devices with small displays as they avoid having to scroll through tables in Runtime.
- Visual machine simulation
  You can visually simulate your machine in a screen using graphical screen objects. This enables you to display parameter settings more clearly by positioning I/O fields immediately next to machine elements such as axes or guide rails. which produces a direct relationship between the values and the machine.

Synchronizing recipe values

To enter recipe data record values outside the advanced recipe view in the configured I/O fields, activate "Synchronize recipe view and recipe tags" under "Properties > Synchronization" in the recipe properties.

The "Synchronize recipe view and recipe tags" option is not available for the simple recipe view.

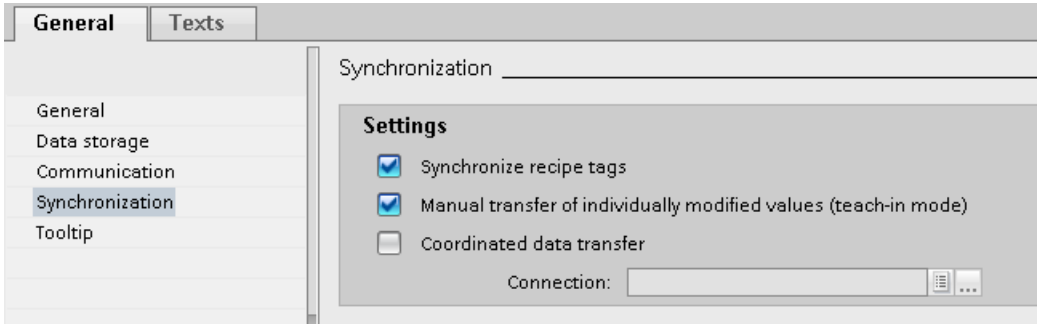

Figure 6-1 Recipes_Synchronization_Advanced

Transferring recipe values automatically

If the entered recipe values must be immediately transferred to the connected PLC in Runtime, deactivate "Manual transfer of individual modified values (teach-in mode)" under "Properties > Options".

Configure the "SetRecipeTags" system function if you want to enable and disable the immediate transfer of entered recipe values in Runtime.

System functions

The system functions for loading, saving and transferring recipe data records and recipes are located in the "Recipes" group.

The following system functions are available for operator control of a recipe screen:

- ImportDataRecords
- ExportDataRecords
- LoadDataRecord
- GetDataRecordFromPLC
- ClearDataRecord
- SaveDataRecord
- SetDataRecordToPLC
- SetDataRecordTagsToPLC
- SetRecipeTags
- GetDataRecordTagsFromPLC

The following system functions are available for operator control of the recipe view when it is being used in the recipe screen:

- RecipeViewSaveDataRecord
- RecipeViewSaveAsDataRecord
- RecipeViewSynchronizeDataRecordWithTags
- RecipeViewClearDataRecord
- RecipeViewNewDataRecord
- RecipeViewGetDataRecordFromPLC
- RecipeViewSetDataRecordToPLC
- RecipeViewRenameDataRecord
- RecipeViewShowOperatorNotes
- RecipeViewMenu (for simple recipe view only)
- RecipeViewOpen (for simple recipe view only)
- RecipeViewBack (for simple recipe view only)

See also

Recipe view basics (Page 1096)

Recipe view basics

Introduction

The recipe view is a WinCC display and operating object. The recipe view is used to display, edit and manage data records.

The layout of the recipe views is dependent on the device and the device version.

HMI devices with simple recipe view

HMI device	Type
Basic Panels	KP300 Basic mono PN KP400 Basic color PN
	KTP400 Basic mono PN KTP400 Basic color PN KTP400 Basic mono PN Portrait KTP400 Basic color PN Portrait
	KTP600 Basic
	KTP1000 Basic
	TP1500 Basic

[1] The simple recipe view display can be changed to the advanced recipe view

HMI devices with advanced recipe view prior to device version V13

HMI device	Type
Mobile Panels	Panel 177
	Panel 277 Panel 277 IWLAN V2 Panel 277F IWLAN V2 Panel 277F IWLAN (RFID Tag)

[1] The advanced recipe view display can be changed to the simple recipe view

HMI devices with advanced recipe view as of device version V13

HMI device	Type
Basic Panels	KTP400 Basic PN KTP400 Basic PN Portrait
	KTP700 Basic
	KTP900 Basic
	KTP1200 Basic

HMI device	Type
Comfort Panels	KP400 Comfort KTP400 Comfort
	KP700 Comfort TP700 Comfort
	KP900 Comfort TP900 Comfort
	KP1200 Comfort TP1200 Comfort
	KP1500 Comfort TP1500 Comfort
	TP1900 Comfort
	TP2200 Comfort
Mobile Panels	KTP400F Mobile
	KTP700 Mobile KTP700F Mobile
	KTP900 Mobile KTP900F Mobile

See also

Basics on the recipe screen (Page 1093)

Example of creating a recipe (Page 1151)

6.1.2.2 Simple recipe view

Description of the simple recipe view

Recipe view

The simple recipe view is a display and operating object that is used to manage recipe data records. The recipe view shows recipe data records in tabular form.

The buttons and information displayed in the columns are adjustable.

The values displayed or entered in the recipe view are saved in recipe data records. The recipe data record displayed can be written to the PLC and values read in from the PLC using buttons.

Layout of the display

The simple recipe view consists of the following display areas:

- Recipe list
- Data record list
- Element list

This application is illustrated below:

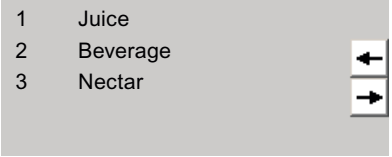

In the simple recipe view, each area is shown separately on the HMI device. You can use the shortcut menu to operate each of these display areas.

The simple recipe view always begins with the recipe list.

Operation

You have the following options for using the simple recipe view, according to the configuration:

- Create, change, copy or delete recipe data records
- Read recipe data records from the PLC or transfer to the PLC

Using the display area and shortcut menu

Toggle between the display areas and the shortcut menus to operate the simple recipe views.

Table 6-1 Operation of the display area

Button	Key	Function
	<Enter>	The next lowest display area is opened, i.e. the data record list or the element list.
	<Esc>	The previous display area opens.
	<INS>	Creates a new data record for the selected recipe if the list of recipes or recipe data records is displayed. Then changes to the list of recipe element. Requirement: "Properties >General > Processing mode" is activated. The button can be simulated with the "Key SimulateSystem-Key" function even on devices without keys.
	<DEL>	Deletes the selected recipe data record in the list of recipe data records. Requirement: "Properties >General > Processing mode" is activated.
	<Up>/<Down>	Selects the previous/next entry.
	<Pg Up>/<Pg Down>	Moves the display up or down one page.
	<Home>/<End>	Selects the first/last entry. The first/last entry is selected.

Table 6-2 Operation of the shortcut menu

Button	Key	Function
→	<Right>	The shortcut menu of the display area opens.
←	<Esc>	The menu is closed. The display area opens.
	Input of the number of the menu command	The menu command is executed.

Shortcut menus of the simple recipe view

You can click the → button in each display area to call up a selection of commands. The command selection lists those commands that are available in the current display area. A number is assigned to each command. The command is executed when you enter this number. Alternatively select the command and press the <Return> key.

Shortcut menus in the recipe list

Menu command	Function
New	A new recipe data record is created for the selected recipe. If a start value is configured, it is displayed in the input field.
Display tooltip	The tooltip configured for the recipe is displayed.
Open	The record list of the selected recipe opens.

Shortcut menus of the recipe data record list

Menu command	Function
New	Creates a new recipe data record. If a start value is configured, it is shown in the input field.
Delete	The displayed record is deleted.
Save as	The selected data record is saved under a different name. A dialog box opens where you can enter the name.
Rename	Renames the selected data record. A dialog box opens where you can enter the name.
Open	The element list of the selected data record opens.
Previous	The recipe list opens.

Shortcut menus of the recipe element list

Menu command	Function
Save	The selected data record with the recipe element is saved.
To PLC	The displayed values of the selected data record are transferred from the HMI device to the PLC.
From PLC	The recipe values from the PLC are displayed in the recipe view of the HMI device.

Menu command	Function
Save as	The data record is saved under a new name. A dialog box opens where you can enter the name.
Display tooltip	The tooltip configured for the recipe element is displayed.
Rename	The selected recipe element is renamed. A dialog box opens where you can enter the name.
Previous	The data record list opens.

Shortcut menus in the data record list

Note

HMI device dependency

The following menu commands are configured in Basic Panels.

Menu command	Function
To PLC	The displayed values of the selected data record are transferred from the HMI device to the PLC.
From PLC	The recipe values from the PLC are displayed in the recipe view of the HMI device.

Updating and displaying values

Note

Processed recipe data record is changed in the background

Only applies for Basic Panels: If an operator has changed a recipe data record and a PLC job wants to read or write any recipe data record of this recipe, the PLC job is stopped and a system alarm is output. On the other hand, the changed value is displayed immediately if only the PLC job and no operator has changed recipe data.

Does not apply for Basic Panels: If an operator has changed a recipe data record and a PLC job has changed values of the recipe data record concerned, the recipe view is not updated automatically. To update the recipe view, reselect the relevant recipe data record.

See also

Recipe view basics (Page 1096)

Configuration options of the simple recipe view (Page 1100)

Configuration options of the simple recipe view

The response of the simple recipe view can be defined in the recipe view inspector window. Some devices, for example basic panels, only support the simple recipe view.

Please note the following for devices with a simple recipe view:

1. In the Inspector window, under "Properties > Display > Mode", select "Simple view" as "View type".
2. The "Properties > Simple view" area contains additional properties only applicable to the simple recipe view.
3. All other properties are also applicable to the advanced recipe view.

Displaying recipe data record values only

To display the recipe data in a recipe view for checking only, proceed as follows:

1. Deselect "Edit mode" in the "General" group.

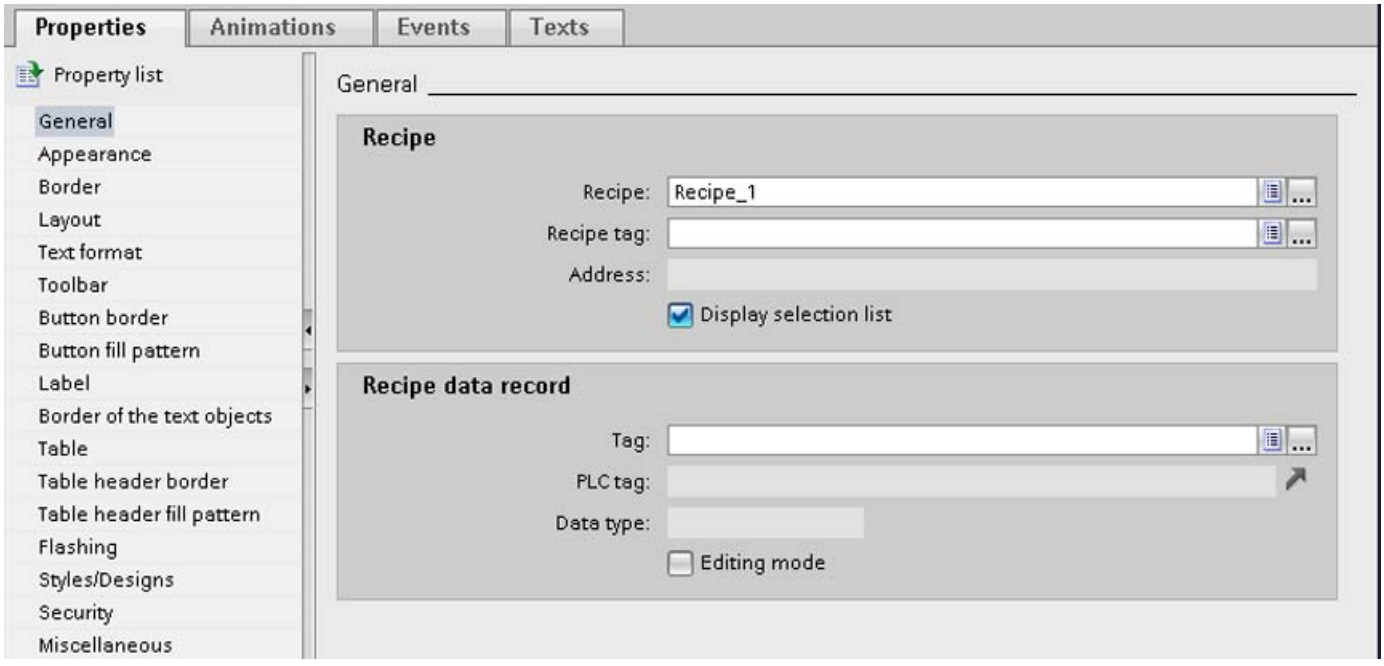

You cannot create, rename, edit or delete recipe data.

Writing a recipe data record number or name to a tag

A tag to each recipe data record can be configured in the advanced recipe view. Depending on the "String" or "Int" data type of the tag, the name or number of the recipe data record is stored in the tag. Conversely, the tag can also be used to select a recipe data record by entering the corresponding value. For example, the tag can be transferred as a parameter for a system function.

Follow these steps:

1. Enter a tag of the "Int" type in the field "Tag", under "Properties > General > Recipe data record".

The number of the recipe data record is stored in a tag.

Configuring an event on the recipe view

Note

Events and buttons

The "Events" register is faded out when a minimum of one button is enabled.

To configure an event on the recipe view, follow these steps:

1. Select the recipe view that was added to the screen in the "Screens" editor.
   The properties of the recipe view are displayed in the Inspector window.
2. Deactivate all buttons under "Properties > Toolbar" and "Properties > Simple view".
3. In the Inspector window, under "Properties > Events", click the event to be configured, e.g. "Enable".
4. Configure a function list for the event.

The function list is processed when the operator enables the recipe view.

Animation properties of the recipe view

To configure an animation of a recipe view, proceed as follows:

1. Select the recipe view that was added to the screen in the "Screens" editor.
   The properties of the recipe view are displayed in the Inspector window.
2. Click under "Properties > Animations" in the Inspector window.

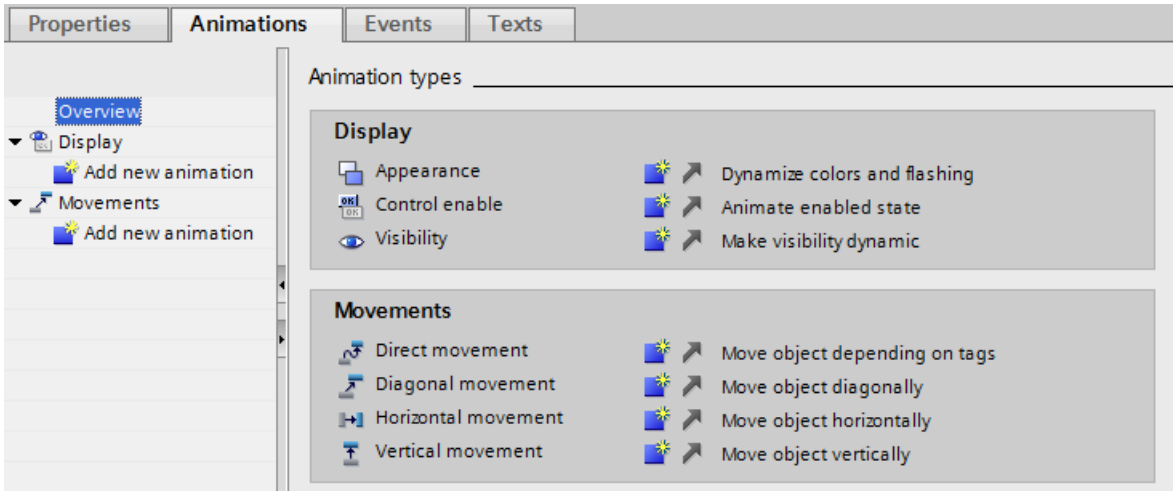

3. Link a tag to one or more of the following properties.
   - X-Position and y-Position
   - Design: Colors, flashing
   - Operability
   - Visibility
     In the "Animations> Overview" area, all animations are summarized in tabular form. Under "Animations> Tag links > Tag linkage", as well as visibility and position a tag can also be linked to height and width.

Note

Animations and buttons

If all buttons are not disabled, an error message appears whilst compiling the project for windows-CE HMI devices.

Constraints on the simple recipe view

The following functions are not possible in the simple recipe view:

- Synchronizing recipe view and recipe tags
- Writing a recipe number / name to a tag
- Display status bar
- Display data record number
- Display label
- Display table

See also

Recipe view basics (Page 1096)

Description of the simple recipe view (Page 1097)

Configuration of recipes (Page 1086)

Using the simple recipe view

Controlling the simple recipe view with mouse or touchpad

To control the simple recipe view with mouse or touchpad, proceed as follows:

1. Select the desired recipe from the recipe view.
2. Click the button.
   The shortcut menu is opened.
3. Select the desired menu command.
   The menu command is executed.

Controlling the simple recipe view with the keyboard

To control the simple recipe view with the keyboard, proceed as follows:

1. Press the <Tab> key until the simple recipe view is selected.
2. Select the desired recipe with the cursor keys.
3. Press <Right>.
   The shortcut menu is opened.
4. Press the <Down> key until the desired menu command is selected.
5. Press <Enter> to confirm the command.

Key shortcuts for the simple recipe view

The following key shortcuts are activated for the simple recipe view in Runtime if "Activate keyboard operation" is enabled in the ES.

Key shortcut	Effect	Menu command
<Insert>	Generates a new recipe data record	New
<Del>	Deletes the recipe data record displayed.	Delete

Operating the recipe view with function keys

The recipe view can be operated with function keys, for example if the HMI device does not have touch functionality. You can assign functions such as "SaveDataRecord" to the function keys on the HMI device.

See also

Recipe view basics (Page 1096)

Managing recipe data records

Administration of recipe data records

You have the following options for managing the simple recipe view, according to the configuration:

- Creating new recipe data records
- Copy recipe data records
- Edit recipe data records
- Delete recipe data records

Creating new recipe data records

To create a new recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to create a new recipe data record.
2. Select the "New" command from the shortcut menu for the recipe list.
   A new data record with the next available number will be created.
   The element list of the new recipe data record opens.
3. Enter values for the elements of the recipe data record.
   The configuration data may already contain default values for the recipe data record.
4. Select the "Save" command from the shortcut menu for the element list.
   The dialog "Save as" opens.
5. Enter the name and number of the recipe data record.
6. Click the "OK" button.
   The data record is saved under the new name.
   If the recipe data record already exists, a dialog is opened. In this dialog, specify whether the existing data record is to be overwritten.

Copying a recipe data record

To copy a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to copy an existing recipe data record.
2. On the HMI device, select the recipe data record of which you want to save a copy.
3. Select the "Save As" command from the shortcut menu for the data record list.
   The dialog "Save as" opens. The recipe data record is automatically given the next free recipe data record number.
4. Under name, enter the name of the record.
5. Click the "OK" button.

Modifying a recipe data record

To change a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to edit an existing recipe data record.
2. Select the recipe data record that you want to edit on the HMI device.
3. Select the recipe data record.
   The element list of the recipe data record is displayed.
4. Replace the old values with new ones.
5. Select the "Save" command from the shortcut menu for the element list.

Deleting a recipe data record

To delete a recipe data record, proceed as follows:

1. Select the recipe on the HMI device from which you want to delete an existing recipe data record.
2. Select the recipe data record that you want to delete on the HMI device.
3. Select the "Delete" command from the shortcut menu for the data record list.
4. Confirm this security prompt to delete the data record.

See also

Recipe view basics (Page 1096)

Read recipe data record from PLC

Introduction

In Runtime, you can change values directly in the plant that are also stored in recipes in the HMI device. This applies if a valve was opened further directly in the plant than was specified in the recipe. The values of the recipe data records saved in the HMI device possibly no longer match the values in the PLC.

You can read the values of the recipe tags from the PLC and write them to a recipe data record.

The read values are written to the recipe data record that is currently displayed on the HMI device.

Procedure

To read a recipe data record from the PLC, proceed as follows:

1. Open the recipe on the HMI device.
   The data record list opens.
2. Select the element list of the recipe data record to which you want to apply the values from the PLC.

3. Select the "From PLC" command from the shortcut menu for the element list.
   The values are read from the PLC and displayed in the current recipe data record.
4. If you want to save the values, select the "Save" or "Save As" command.

Result

The values are read from the PLC, visualized on the HMI device and saved to the recipe data record.

Note

Basic Panels

With Basic Panels, the "From PLC" menu command can also be configured for the data record list. In this case, you can also select the "From PLC" menu command in the data record list.

See also

Recipe view basics (Page 1096)

Transferring a recipe data record to the PLC

Introduction

For the values of a data record that was changed in the recipe view to take effect, you must transfer the values to the PLC.

The values displayed in the recipe view are always transferred to the PLC.

Procedure

To transfer a recipe data record to the PLC, proceed as follows:

1. Open the recipe you want to use.
   The data record list opens.
2. Select the element list of the recipe data record whose values you want to transfer to the PLC.
3. Select the "To PLC" command from the shortcut menu for the element list.

Result

The values of the recipe data record are transferred to the PLC.

Note

Basic Panels

With Basic Panels, the "From PLC" menu command can also be configured for the data record list. In this case, you can also select the "From PLC" menu command in the data record list.

See also

Recipe view basics (Page 1096)

Synchronization of recipe data records with the PLC (Page 1089)

Transferring recipe data records (Page 1083)

6.1.2.3 Advanced recipe view (prior to V13)

Description of the advanced recipe view (prior to V13) (RT Professional, RT Advanced, Panels, Comfort Panels)

Recipe view

The advanced recipe view is a display and operating object that is used to manage recipe data records. The recipe view shows recipe data records in tabular form.

The buttons, headers and information displayed in the columns can be set.

The values displayed or entered in the recipe view are saved in recipe data records.

The layout of the recipe views is dependent on the device and the device version.

Application

The recipe view is used to display, edit and manage data records.

Layout of the display

The figure below shows an example of the advanced recipe view on an HMI device prior to device version V13.

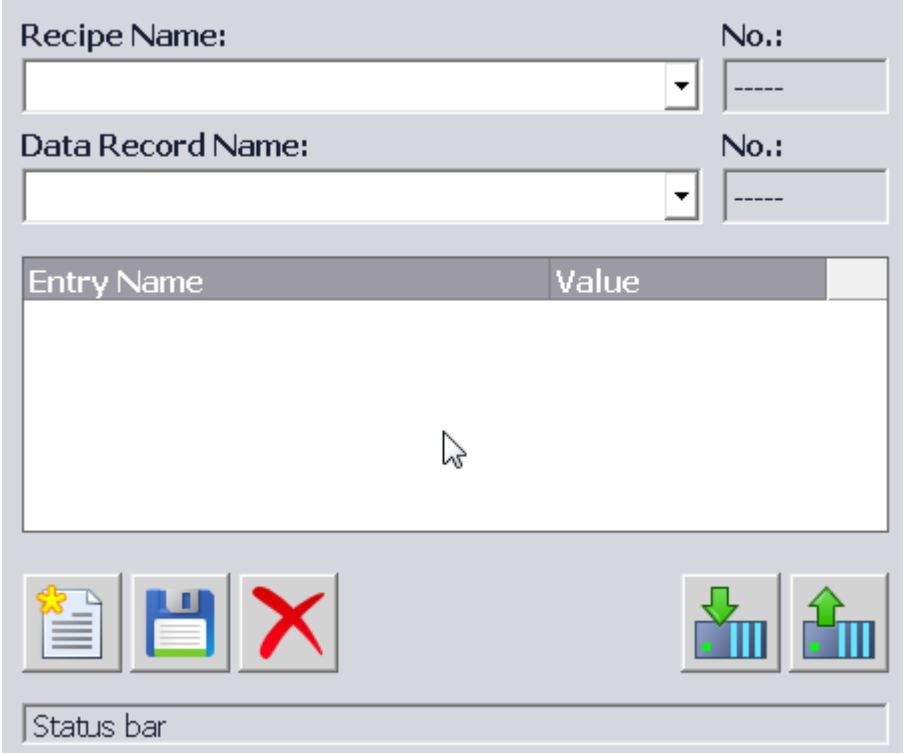

Operation

Depending on the configuration you can:

- Create, edit, copy or delete recipe data records
- Read recipe data records from or transfer to the PLC
- Synchronize recipe data records with the associated recipe tags

Operating elements

The following operating elements can be configured in the recipe view:

Operating object	Key combination	Function
?		The configured tooltip is displayed.
	<Ctrl+Spacebar>	Creates a new recipe data record. If a start value is configured, it is shown in the input field.

Operating object	Key combination	Function
	<Ctrl+Enter>	Saves the displayed values of the recipe data record. The storage location is predefined by the project.
	<Ctrl+*>	The recipe data record is saved under a different name regardless of the recipe view. A dialog box opens where you can enter the name.
	<Ctrl+Del>	Deletes the recipe data record displayed.
	<Ctrl+=>	The system always updates the current value of the recipe view with the up-to-date recipe tag value. When the value shown in the recipe view is more recent than the current recipe tag value, the system writes this value to the recipe tag. "Synchronize recipe view and recipe tags" must be activated in the recipe properties before this function can be used.
	<Ctrl+Down Arrow>	The values of the set recipe data record displayed in the recipe view are transferred to the PLC.
	<Ctrl+Up Arrow>	The recipe values from the PLC are displayed in the recipe view.

Updating and displaying values

If values of the corresponding recipe data record are changed by a job mailbox, the recipe view is not updated automatically. The values are changed in the background. To update the recipe view, reselect the relevant recipe data record.

See also

Recipe view basics (Page 1096)

Using the advanced recipe view (Page 1113)

Managing recipe data records (Page 1115)

Synchronizing recipe data record (Page 1116)

Reading recipe data records from the PLC (Page 1117)

Transferring a recipe data record to the PLC (Page 1118)

Configuration options of the advanced recipe view (Page 1110)

Configuration options of the advanced recipe view (RT Advanced, Panels, Comfort Panels)

You can specify the behavior of the advanced recipe view in the Inspector window of the recipe view.

The layout of the recipe views is dependent on the device and the device version.

Please note the following for devices with an advanced recipe view:

1. In the Inspector window under "Properties > Display > Mode", select "Advanced view" as the "View type".
2. The "Properties > Simple view" area contains properties only applicable to the simple recipe view.
3. All other properties apply to the advanced recipe view.

Displaying a recipe

To only allow access to the recipe data records of a specific recipe in a screen, follow these steps:

1. Enter the required recipe or select an existing recipe in the "Recipe" field under "Properties > General".
2. When a recipe is entered in the "Recipe" field, you will have to enable the "Selection list" if you want to display the recipe name in runtime.

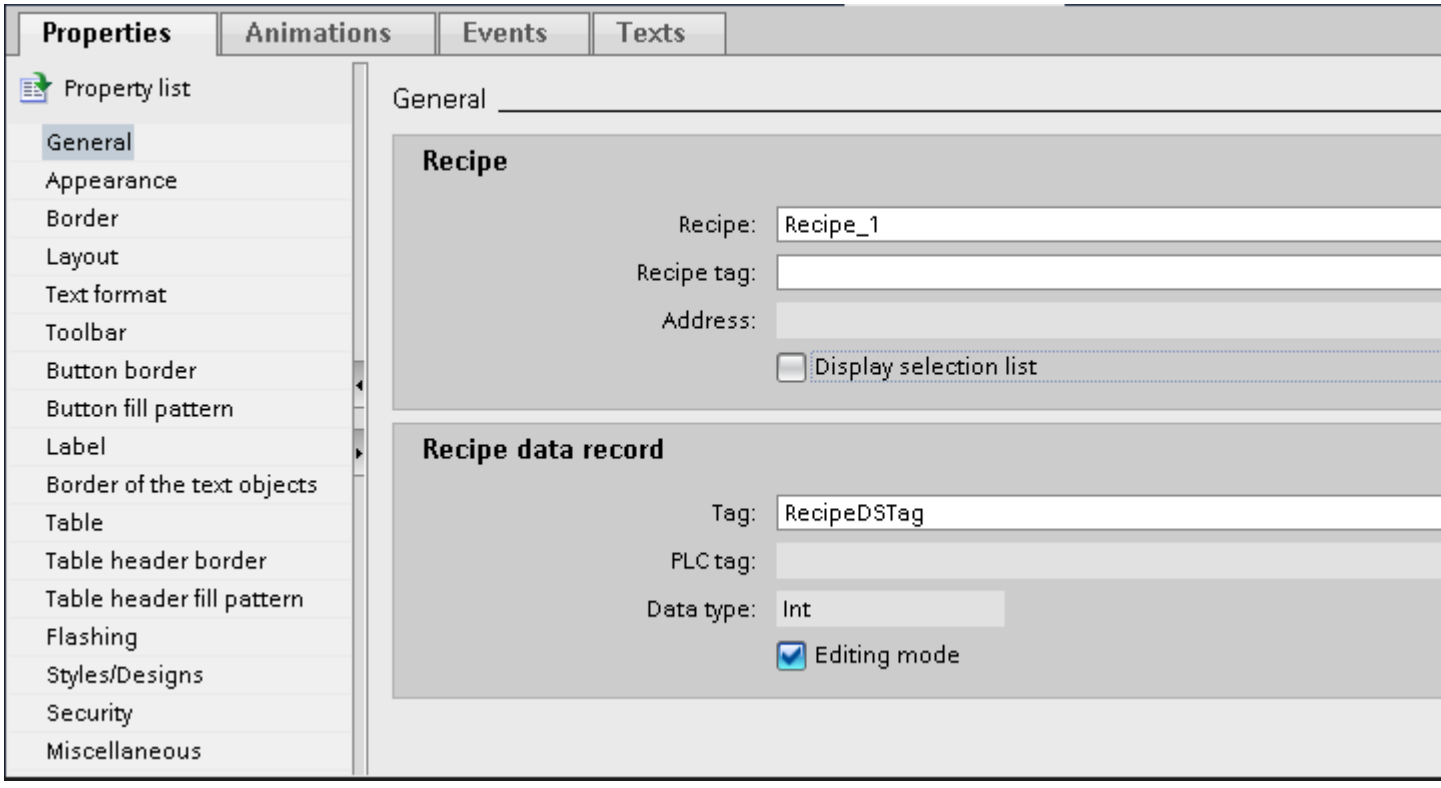

The selected recipe appears in the recipe view.

Writing a recipe number or name and recipe data record number or name to a tag

Both the recipe and the recipe data record can each be configured to a tag in the recipe view. Depending on the "String" or "Int" data type of the tag, the name or number of the recipe or recipe data record is stored in the tag. Conversely, the tag can be used to select the recipe or recipe data record by entering the corresponding value. For example, the tag can be transferred as a parameter for a system function.

Follow these steps:

1. Enter a tag of the "String" type in the field "Recipe tag", under "Properties > General > Recipe".
2. Enter a tag of the "Int" type in the field "Tag", under "Properties > General > Recipe data record".

The name of the recipe and the number of the recipe data record are each stored in a tag.

Using the recipe view as a drop-down list

To use the recipe view as a selection field for recipes and recipe data records in a recipe screen, follow these steps:

1. Select the tag for the recipe name under "General > Recipe > Recipe tag".
2. Select the tag for the recipe data record name under "General > Recipe data record >Tag".
3. Disable the "Editing mode". You cannot create, rename, edit or delete recipe data.
4. In order to select recipes, enable "Display selection list" and make sure that no recipe is selected under "Properties > General > Recipe".
5. Deactivate all buttons under "Properties > Toolbar".

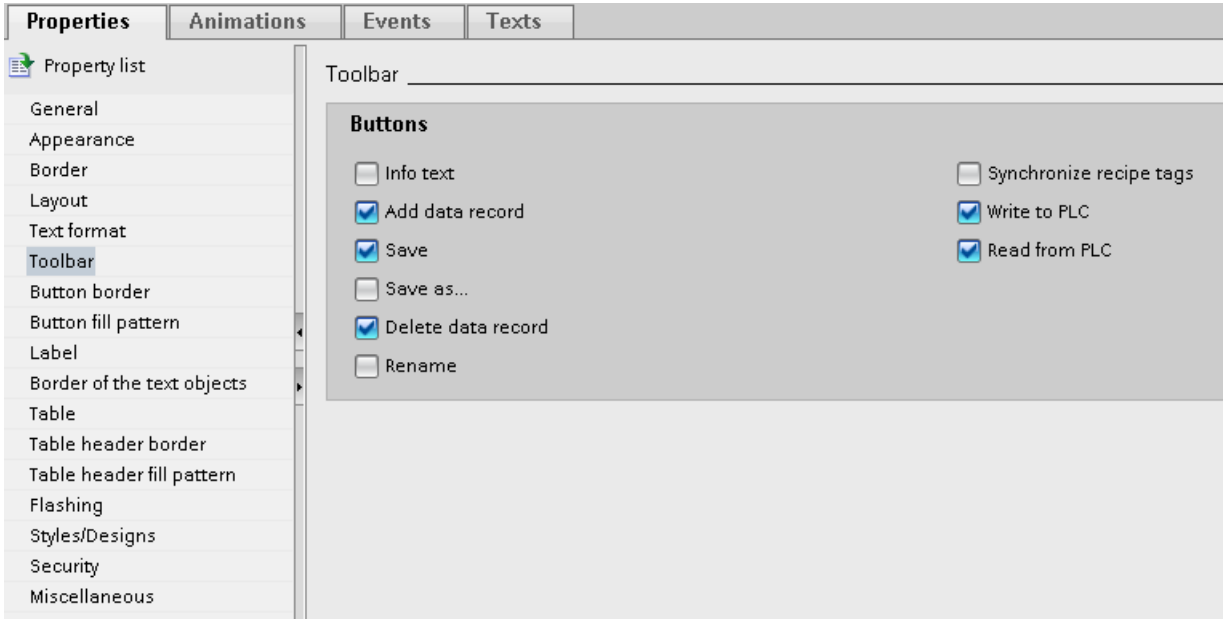

Configuring an event on the recipe view

To configure an event on the recipe view, follow these steps:

1. Select the recipe view that was added to the screen in the "Screens" editor.
   The properties of the recipe view are displayed in the Inspector window.
2. In the Inspector window, click the event you want to configure under "Properties > Events".
3. Configure a function list for the selected event.
   The function list is processed when the configured event occurs.

See also

Recipe view basics (Page 1096)

Description of the advanced recipe view (prior to V13) (Page 1108)

Configuration of recipes (Page 1086)

Using the advanced recipe view

Introduction

You can control the recipe view using both mouse or touchpad or with the keyboard.

The layout of the recipe views is dependent on the device and the device version.

Controlling the recipe view with mouse or touchpad

To control the recipe view with mouse or touchpad, proceed as follows:

1. Select the recipe you want to use.
   The records for the recipe are displayed.
2. Click on the data record you wish to edit.
3. Click the button whose function you want to run.

Controlling the recipe view with the keyboard

To control the recipe view with the keyboard, proceed as follows:

1. Press the <Tab> key until the cursor reaches the field for selecting the recipe.
2. Press <Enter>.
   The drop-down list box for the recipes opens.
3. Select a recipe. You navigate between the next or previous entry in the list by using the cursor keys <Left>, <Right>, <Up> and <Down>.
4. Select a data record.
5. Press the <Tab> key until the operator control object you wish to use is selected.

Alternatively you can control the recipe view using certain key combinations.

Operating the recipe view with function keys

The recipe view can be operated with function keys, for example if the HMI device does not have touch functionality. You can assign functions such as "SaveDataRecord" to the function keys on the HMI device.

Key shortcuts for the advanced recipe view

The following key shortcuts are activated for the advanced recipe view in Runtime if "Activate keyboard operation" is enabled in the ES.

Keys shortcut	Effect	Menu command	Button
<HELP>	Calls up the configured tooltip for the selected recipe.	Tooltip	
<Ctrl+Space-bar>	Generates a new recipe data record. Any configured start value is displayed in the input field.	Add data record	
<Ctrl+Enter>	Saves the modified record with its current name.	Save	
<Ctrl+*>	Saves the modified record with a new name.	Save as	
<Ctrl+Del>	Deletes the recipe data record displayed.	Delete data record	
<Ctrl+=>	Compares the values of the selected data record with the values on the PLC. Any value in the recipe view which is more recent compared to the current recipe tag value is written to the recipe tag.	Synchronize recipe tags	
<Ctrl+Down Arrow>	Sends the current value to the PLC.	Write to PLC	
<Ctrl+Up Arrow>	Reads the current value from the PLC.	Read from PLC	

Screen change

If you change to another screen and have not yet saved changes to the recipe data in the recipe view, you will be prompted to save the recipe data. The recipe name and the name of the recipe data record are displayed to show which recipe data have not been saved yet.

See also

Description of the advanced recipe view (prior to V13) (Page 1108)

Example of creating a recipe (Page 1151)

Managing recipe data records

Administration of recipe data records

You have the following options for managing recipe data records, according to the configuration:

- Creating new recipe data records
- Copy recipe data records
- Edit recipe data records
- Delete recipe data records

Creating new recipe data records

To create a new recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to create a new recipe data record.
2. Click or press <Ctrl+Spacebar>.
   A new recipe data record with the next available number is created.
   If you change the new data record number to an existing data record number, the existing data record is overwritten.
3. Enter values for the elements of the data record.
   The elements of the recipe data record can be assigned default values depending on the configuration.
4. Click or press <Ctrl+*>.
   The dialog "Save as" opens.
5. Enter a name for the data record.
6. Click "OK" to confirm your input.
   The data record is saved under the new name.
   If the recipe data record already exists, a dialog is opened. In this dialog, specify whether the existing data record is to be overwritten.

Copying a recipe data record

To copy a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to copy an existing recipe data record.
2. Select the recipe data record that you want to copy on the HMI device.
3. Click in the recipe view or press <Ctrl+*>.
   The dialog "Save as" opens.

4. Enter a name for the data record.
5. Click "OK" to confirm your input.

Modifying a recipe data record

To change a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to edit an existing recipe data record.
2. Select the recipe data record that you want to edit on the HMI device.
3. Replace the old values with new ones.
4. Click in the recipe view or press <Ctrl+Enter>.

Deleting a recipe data record

To delete a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to delete an existing recipe data record.
2. Select the recipe data record that you want to delete on the HMI device.
3. Click in the recipe view or press <Ctrl+Del>.

See also

Description of the advanced recipe view (prior to V13) (Page 1108)

Example of creating a recipe (Page 1151)

Synchronizing recipe data record

Introduction

Differences between the following values may occur during runtime:

- The values displayed in the recipe view
- The actual values of the recipe tags

Depending on the configuration, the values displayed in the recipe view are synchronized with the recipe tags. Synchronization encompasses all tags of a recipe data record.

Note

Changed tag name

The tag and the value of the recipe data record cannot be associated if you have renamed the tag you want to synchronize. The tags in question are not synchronized.

Note

Recipe tags can only be synchronized in the advanced recipe view.

Requirement

- A recipe data record is displayed in the recipe view.
- The values of recipe tags were modified by teaching, for example.

Procedure

To synchronize a recipe data record, proceed as follows:

- Click in the recipe view or press <Ctrl+=>.

The system always updates the current value of the recipe view with the up-to-date recipe tag value.

When the value shown in the recipe view is more recent than the current recipe tag value, the system writes this value to the recipe tag.

See also

Description of the advanced recipe view (prior to V13) (Page 1108)

Example of creating a recipe (Page 1151)

Synchronization of recipe data records with the PLC (Page 1089)

Reading recipe data records from the PLC

Introduction

In runtime, you can change values directly in the plant that are also stored in recipes in the HMI device. This applies for example if a valve was opened right at the plant wider than specified in the recipe. The values of the recipe data records saved in the HMI device may then possibly no longer match the values in the PLC.

You can read the values of the recipe tags from the PLC and write them to a recipe data record.

The values read are written to the recipe data record that is currently displayed on the HMI device.

Procedure

Proceed as follows to read a recipe data record from the PLC:

1. Select the recipe on the HMI device.
2. On the HMI device, select the recipe data record whose values you want to read from the PLC.
3. Click in the recipe view or press <Ctrl+Up Arrow>.

See also

Description of the advanced recipe view (prior to V13) (Page 1108)

Example of creating a recipe (Page 1151)

Transferring a recipe data record to the PLC

Introduction

For the values of a data record that was changed in the recipe view to take effect, you must transfer the values to the PLC.

The values displayed in the recipe view are always transferred to the PLC.

Procedure

Proceed as follows to transfer a recipe data record to the PLC:

1. Select the recipe on the HMI device.
2. On the HMI device, select the recipe data record whose values you want to transfer to the PLC.
3. Click in the recipe view or press <Ctrl+Down Arrow>.

See also

Description of the advanced recipe view (prior to V13) (Page 1108)

Example of creating a recipe (Page 1151)

Synchronization of recipe data records with the PLC (Page 1089)

Transferring recipe data records (Page 1083)

6.1.2.4 Advanced recipe view (as of V13)

Description of advanced recipe view (as of V13)

Introduction

The advanced recipe view is a display and operating object that is used to manage recipe data records. The recipe view shows recipe data records in tabular form.

The buttons, headers and information displayed in the columns can be set.

The values displayed or entered in the recipe view are saved in recipe data records.

The layout of the recipe views is dependent on the device and the device version.

Application

The recipe view is used to display, edit and manage data records.

Layout of the display

The figure below shows an example of the advanced recipe view on an HMI device, device version V13 or later.

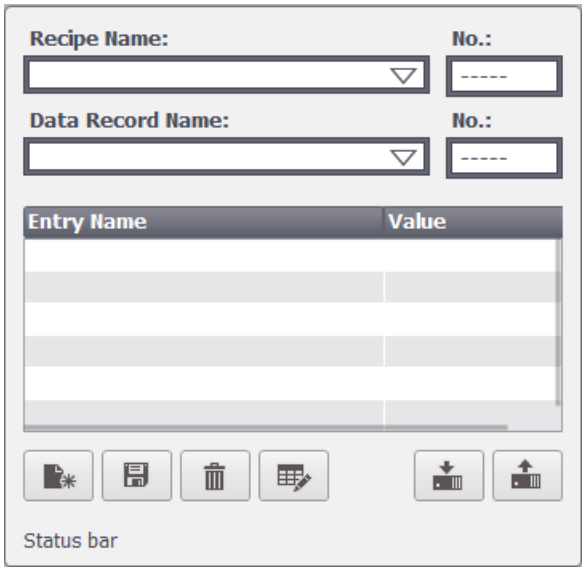

Operation

Depending on the configuration you can:

- Create, edit, copy or delete recipe data records
- Read recipe data records from or transfer to the PLC
- Synchronize recipe data records with the associated recipe tags

Operator controls

The following operating elements can be configured in the recipe view:

Operator controls	Keyboard shortcut	Function
	<HELP>	The configured tooltip is displayed.
	<Ctrl+Spacebar>	Creates a new recipe data record. If a start value is configured, it is shown in the input field.
	<Ctrl+Enter>	Saves the displayed values of the recipe data record. The storage location is predefined by the project.
	<Ctrl+*>	The recipe data record is saved under a different name regardless of the recipe view. A dialog box opens where you can enter the name.
	<Ctrl+Del>	Deletes the recipe data record displayed.
	<Ctrl+Shift+T>	The selected recipe data record is renamed. A dialog box opens where you can enter the name.

Operator controls	Keyboard shortcut	Function
	<Ctrl+=>	If tags on the HMI device are used in a recipe and at an additional point in the configuration (for example, as process tags in an I/O field) and the recipe-specific option "Manual transfer of individually modified values (teach-in mode)" is activated, the following applies: • If the value of a tag has a more recent value than in the recipe view, this value is transferred to the recipe view. • If the value displayed in the recipe view is more recent than the value of the tag, this value is transferred to the tag. Synchronization only takes place within the HMI device. In this case, the process tag named above as an example in the I/O field is not automatically read from the PLC. 1) If tags on the HMI device are used in a recipe and at an additional point in the configuration (for example, as process tags in an I/O field) and the recipe-specific option "Manual transfer of individually modified values (teach-in mode)" is deactivated, the following applies: • If the value of a tag has a more recent value than in the recipe view, this value is transferred to the recipe view. • If the value shown in the recipe view is more recent than the value of the tag, this value is transferred to the tag. Synchronization with the PLC also takes place. 2)
	<Ctrl+Down Arrow>	The values of the set recipe data record displayed in the recipe view are transferred to the PLC.
	<Ctrl+Up Arrow>	The recipe values from the PLC are displayed in the recipe view.

1) For the recipe, the options "Synchronize recipe tags" and "Manual transfer of individually modified values (teach-in mode)" are activated under "General > Synchronization > Settings".

2) For the recipe, the option "Synchronize recipe tags" is activated under "General > Synchronization > Settings" and the option "Manual transfer of individually modified values (teach-in mode)" is deactivated.

Updating and displaying values

If values of the corresponding recipe data record are changed by a job mailbox, the recipe view is not updated automatically. The values are changed in the background. To update the recipe view, reselect the relevant recipe data record.

See also

Example of creating a recipe (Page 1151)

Using the advanced recipe view (Page 1124)

Managing recipe data records (Page 1127)

Synchronizing recipe data record (Page 1129)

Read recipe data record from PLC (Page 1130)

Transferring recipe data records to the PLC (Page 1131)

Recipe view basics (Page 1096)

Configuration options of the advanced recipe view (V13 or higher) (Page 1122)

Configuration options of the advanced recipe view (V13 or higher) (RT Advanced, Basic Panels, Panels, Comfort Panels)

You can specify the behavior of the advanced recipe view in the Inspector window of the recipe view.

Displaying a recipe

To only allow access to the recipe data records of a specific recipe in a screen, follow these steps:

1. Enter the required recipe or select an existing recipe in the "Recipe" field under "Properties > General".
2. When a recipe is entered in the "Recipe" field, you will have to enable the "Selection list" if you want to display the recipe name in runtime.

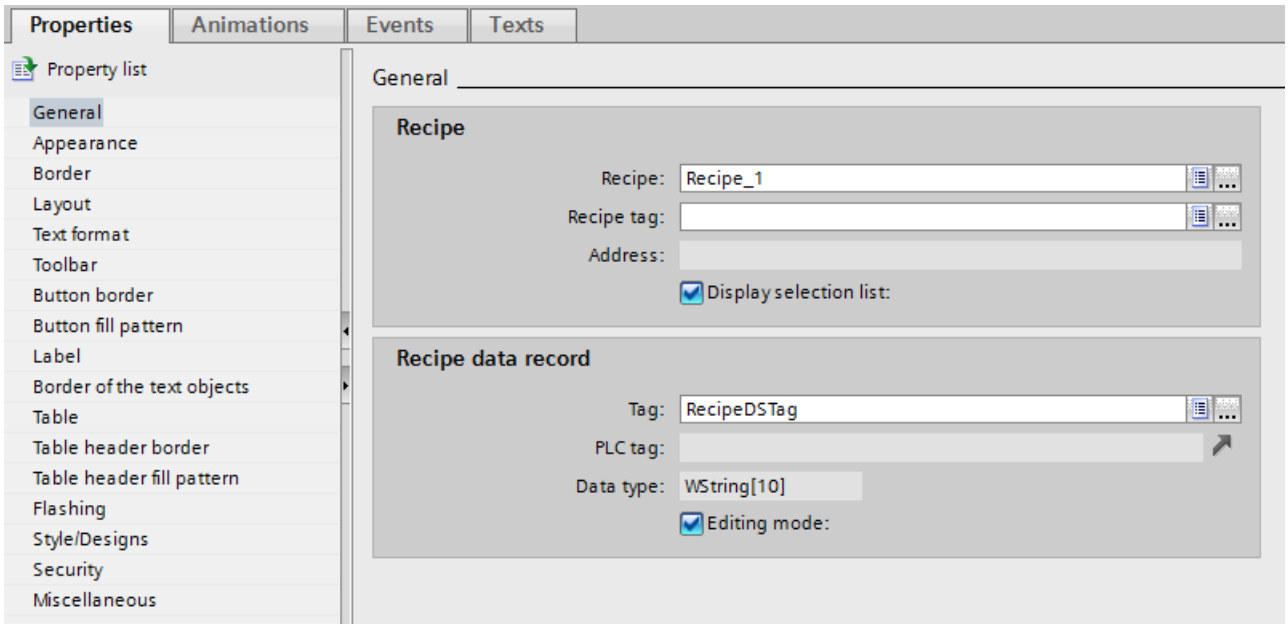

The selected recipe appears in the recipe view.

Writing a recipe number or name and recipe data record number or name to a tag

Both the recipe and the recipe data record can each be configured to a tag in the recipe view. Depending on the "String" or "Int" data type of the tag, the name or number of the recipe or recipe data record is stored in the tag. Conversely, the tag can be used to select the recipe or recipe data record by entering the corresponding value. For example, the tag can be transferred as a parameter for a system function.

Follow these steps:

1. Enter a tag of the "String" type in the field "Recipe tag", under "Properties > General > Recipe".
2. Enter a tag of the "Int" type in the field "Tag", under "Properties > General > Recipe data record".

The name of the recipe and the number of the recipe data record are each stored in a tag.

Using the recipe view as a drop-down list

To use the recipe view as a selection field for recipes and recipe data records in a recipe screen, follow these steps:

1. Select the tag for the recipe name under "General > Recipe > Recipe tag".
2. Select the tag for the recipe data record name under "General > Recipe data record >Tag".
3. Disable the "Editing mode". You cannot create, rename, edit or delete recipe data.
4. In order to select recipes, enable "Display selection list" and make sure that no recipe is selected under "Properties > General > Recipe".
5. Deactivate all buttons under "Properties > Toolbar".

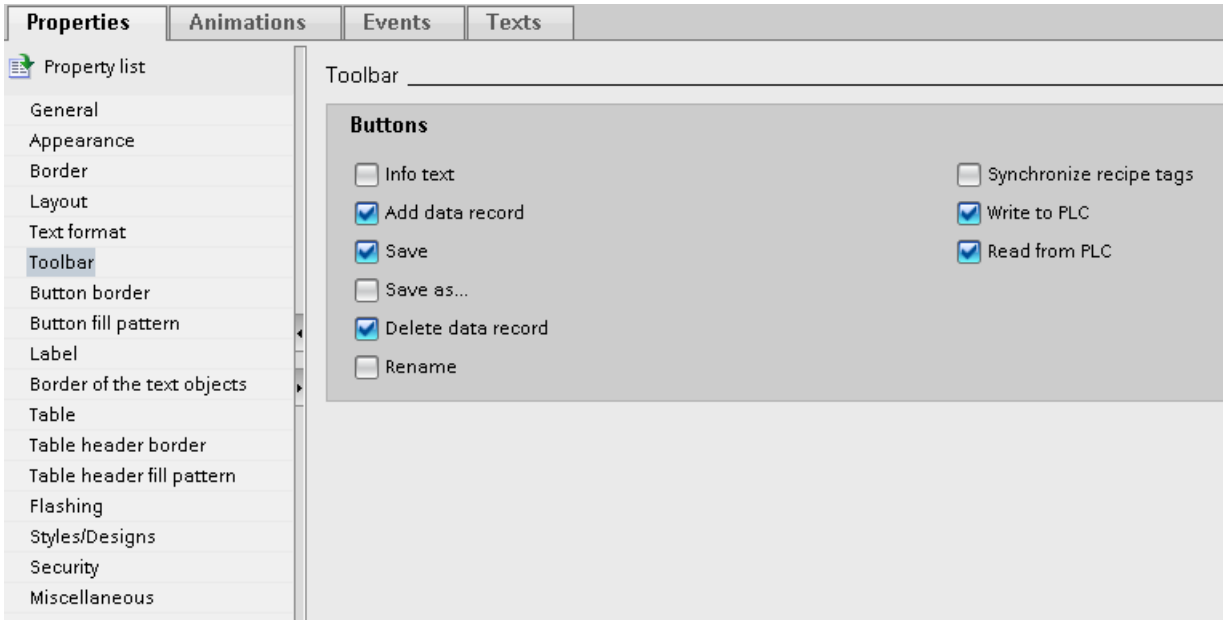

Configuring an event on the recipe view

To configure an event on the recipe view, follow these steps:

1. Select the recipe view that was added to the screen in the "Screens" editor.
   The properties of the recipe view are displayed in the Inspector window.
2. In the Inspector window, click the event you want to configure under "Properties > Events".
3. Configure a function list for the selected event.
   The function list is processed when the configured event occurs.

See also

Example of creating a recipe (Page 1151)

Recipe view basics (Page 1096)

Description of advanced recipe view (as of V13) (Page 1119)

Configuration of recipes (Page 1086)

Using the advanced recipe view

Introduction

You can control the recipe view using both mouse or touchpad or with the keyboard.

Controlling the recipe view with mouse or touchpad

To control the recipe view with mouse or touchpad, proceed as follows:

1. Select the recipe you want to use.
   The records for the recipe are displayed.
2. Click on the data record you wish to edit.
3. Click the button whose function you want to run.

Controlling the recipe view with the keyboard

To control the recipe view with the keyboard, proceed as follows:

1. Press the <Tab> key until the cursor reaches the field for selecting the recipe.
2. Press <Enter>.
   The drop-down list box for the recipes opens.
3. Select a recipe. You navigate between the next or previous entry in the list by using the cursor keys <Left>, <Right>, <Up> and <Down>.
4. Select a data record.
5. Press the <Tab> key until the operator control object you wish to use is selected.

Alternatively you can control the recipe view using certain key combinations.

Operating the recipe view with function keys

The recipe view can be operated with function keys, for example if the HMI device does not have touch functionality. You can assign functions such as "SaveDataRecord" to the function keys on the HMI device.

Key shortcuts for the advanced recipe view

The following key shortcuts are activated for the advanced recipe view in Runtime if "Activate keyboard operation" is enabled in the ES.

Keys shortcut	Effect	Menu command	Button
<HELP>	Calls up the configured tooltip for the selected recipe.	Tooltip	
<Ctrl+Space-bar>	Generates a new recipe data record. Any configured start value is displayed in the input field.	Add data record	
<Ctrl+Enter>	Saves the modified record with its current name.	Save	
<Ctrl+*>	Saves the modified record with a new name.	Save as	
<Ctrl+Del>	Deletes the recipe data record displayed.	Delete data record	
<Ctrl+Shift +T>	Changes the name of selected data record.	Rename data record	

Keys shortcut	Effect	Menu command	Button
<Ctrl+=>	If tags on the HMI device are used in a recipe and at an additional point in the configuration (for example, as process tags in an I/O field) and the recipe-specific option "Manual transfer of individually modified values (teach-in mode)" is activated, the following applies: • If the value of a tag has a more recent value than in the recipe view, this value is transferred to the recipe view. • If the value displayed in the recipe view is more recent than the value of the tag, this value is transferred to the tag. Synchronization only takes place within the HMI device. In this case, the process tag named above as an example in the I/O field is not automatically read from the PLC. 1) If tags on the HMI device are used in a recipe and at an additional point in the configuration (for example, as process tags in an I/O field) and the recipe-specific option "Manual transfer of individually modified values (teach-in mode)" is deactivated, the following applies: • If the value of a tag has a more recent value than in the recipe view, this value is transferred to the recipe view. • If the value shown in the recipe view is more recent than the value of the tag, this value is transferred to the tag. Synchronization with the PLC also takes place. 2)	Synchronizing recipe view and recipe tags	
<Ctrl+Down Arrow>	Sends the current value to the PLC.	Write to PLC	
<Ctrl+Up Arrow>	Reads the current value from the PLC.	Read from PLC	

1) For the recipe, the options "Synchronize recipe tags" and "Manual transfer of individually modified values (teach-in mode)" are activated under "General > Synchronization > Settings".

2) For the recipe, the option "Synchronize recipe tags" is activated under "General > Synchronization > Settings" and the option "Manual transfer of individually modified values (teach-in mode)" is deactivated.

Screen change

If you change to another screen and have not yet saved changes to the recipe data in the recipe view, you will be prompted to save the recipe data. The recipe name and the name of the recipe data record are displayed to show which recipe data have not been saved yet.

See also

Description of advanced recipe view (as of V13) (Page 1119)

Managing recipe data records

Administration of recipe data records

You have the following options for managing recipe data records, according to the configuration:

- Creating new recipe data records
- Copy recipe data records
- Edit recipe data records
- Delete recipe data records
- Rename recipe data records

Creating new recipe data records

To create a new recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to create a new recipe data record.
2. Click or press <Ctrl+Spacebar>.
   A new recipe data record with the next available number is created.
   If you change the new data record number to an existing data record number, the existing data record is overwritten.
3. Enter values for the elements of the data record.
   The elements of the recipe data record can be assigned default values depending on the configuration.
4. Enter a name for the data record.
5. Click on the button press the keyboard shortcut <Ctrl+Enter>.
   Alternatively, click on the button to save the data record or press the keyboard shortcut <Ctrl+*>.
   The "Save as" dialog opens. Enter the name for the data record and confirm your entries with "OK".
6. The data record is saved under the new name.
   If the recipe data record already exists, a dialog is opened. In this dialog, specify whether the existing data record is to be overwritten.

Copying a recipe data record

To copy a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to copy an existing recipe data record.
2. Select the recipe data record that you want to copy on the HMI device.
3. Click in the recipe view or press <Ctrl+*>.
   The dialog "Save as" opens.
4. Enter a name for the data record.
5. Click "OK" to confirm your input.

Modifying a recipe data record

To change a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to edit an existing recipe data record.
2. Select the recipe data record that you want to edit on the HMI device.
3. Replace the old values with new ones.
4. Click in the recipe view or press <Ctrl+Enter>.

Deleting a recipe data record

To delete a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to delete an existing recipe data record.
2. Select the recipe data record that you want to delete on the HMI device.
3. Click in the recipe view or press <Ctrl+Del>.

Renaming recipe data records

Proceed as follows to rename a recipe data record:

1. Select the recipe on the HMI device in which you want to rename an existing recipe data record.
2. Select the recipe data record that you want to rename on the HMI device.
3. Click in the recipe view or press <Ctrl+Shift+T>.
4. Enter a new name in the dialog window.

See also

Description of advanced recipe view (as of V13) (Page 1119)

Synchronizing recipe data record

Introduction

Differences between the following values may occur during runtime:

- The values displayed in the recipe view
- The actual values of the recipe tags

Depending on the configuration, the values displayed in the recipe view are synchronized with the recipe tags. Synchronization encompasses all tags of a recipe data record.

Note

Changed tag name

The tag and the value of the recipe data record cannot be associated if you have renamed the tag you want to synchronize. The tags in question are not synchronized.

Note

Recipe tags can only be synchronized in the advanced recipe view.

Requirement

- A recipe data record is displayed in the recipe view.
- The values of recipe tags were modified by teaching, for example.

Procedure

To synchronize a recipe data record, proceed as follows:

- Click in the recipe view or press <Ctrl+=>.

Result

The result depends on your settings:

- For the recipe, the options "Synchronize recipe tags" and "Manual transfer of individually modified values (teach-in mode)" are activated under "General > Synchronization > Settings".
  If tags on the HMI device are used in a recipe and at an additional point in the configuration (for example, as process tags in an I/O field) and the recipe-specific option "Manual transfer of individually modified values (teach-in mode)" is activated, the following applies:
  - If the value of a tag has a more recent value than in the recipe view, this value is transferred to the recipe view.
  - If the value displayed in the recipe view is more recent than the value of the tag, this value is transferred to the tag. Synchronization only takes place within the HMI device.

  In this case, the process tag named above as an example in the I/O field is not automatically read from the PLC.
- For the recipe, the option "Synchronize recipe tags" is activated under "General > Synchronization > Settings" and the option "Manual transfer of individually modified values (teach-in mode)" is deactivated.
  If tags on the HMI device are used in a recipe and at an additional point in the configuration (for example, as process tags in an I/O field) and the recipe-specific option "Manual transfer of individually modified values (teach-in mode)" is deactivated, the following applies:
  - If the value of a tag has a more recent value than in the recipe view, this value is transferred to the recipe view.
  - If the value shown in the recipe view is more recent than the value of the tag, this value is transferred to the tag.

  Synchronization with the PLC also takes place.

See also

Description of advanced recipe view (as of V13) (Page 1119)

Synchronization of recipe data records with the PLC (Page 1089)

Read recipe data record from PLC

Introduction

In runtime, you can change values directly in the plant that are also stored in recipes in the HMI device. This applies for example if a valve was opened right at the plant wider than specified in the recipe. The values of the recipe data records saved in the HMI device may then possibly no longer match the values in the PLC.

You can read the values of the recipe tags from the PLC and write them to a recipe data record.

The values read are written to the recipe data record that is currently displayed on the HMI device.

Procedure

Proceed as follows to read a recipe data record from the PLC:

1. Select the recipe on the HMI device.
2. On the HMI device, select the recipe data record whose values you want to read from the PLC.
3. Click in the recipe view or press <Ctrl+Up Arrow>.

See also

Description of advanced recipe view (as of V13) (Page 1119)

Transferring recipe data records to the PLC

Introduction

For the values of a data record that was changed in the recipe view to take effect, you must transfer the values to the PLC.

The values displayed in the recipe view are always transferred to the PLC.

Procedure

Proceed as follows to transfer a recipe data record to the PLC:

1. Select the recipe on the HMI device.
2. On the HMI device, select the recipe data record whose values you want to transfer to the PLC.
3. Click in the recipe view or press <Ctrl+Down Arrow>.

See also

Description of advanced recipe view (as of V13) (Page 1119)

Synchronization of recipe data records with the PLC (Page 1089)

Transferring recipe data records (Page 1083)

6.1.2.5 Exporting and importing recipe data

Exporting and importing recipe data records (RT Advanced, Panels, Comfort Panels)

Introduction

You can either export recipe data records to a CSV file or TXT file for editing in MS Excel, for example, or import them from a file.

Recipe export and import depends on the configuration of the device and the device version.

List separators

A list separator is used to separate the data records during importing and exporting. The list separator used as the default depends on the setting for formats and numbers in the operating system.

Select "Start > Settings > Control Panel > Regional and Language Options".

If you want to import or export recipe data records, do not use this list separator in the display name of the recipe data records.

The following operating elements can, for example, be configured for using the export/import function:

- Selection field for the recipe
- Selection field for the recipe data record
- Operating element with the "ExportDataRecords" functionality
- Operating element with the "ImportDataRecords" functionality

Exporting a recipe data record

Proceed as follows to export a recipe data record to a CSV file:

1. Select the relevant recipe and recipe data record from the selection boxes.
2. Click the control element with the "ExportDataRecords" functionality.

Importing recipe data records

To import a recipe data record, proceed as follows:

1. Select the relevant recipe and recipe data record from the selection boxes.
2. Click the control element with the "ImportDataRecords" functionality.

Display after import of recipes

If you open the recipe view during the import of recipe data, only the recipe data that is already completely imported will be displayed. The recipe view is not automatically updated following a data import. In order to have a complete view of all the recipe data, do not open the recipe view until the system reports that the recipe data has been imported successfully.

Alternatively, you can update the recipe view after successful completion of the import process.

Restrictions on import/export

On Basic Panels you have the possibility of exporting the recipes to a TXT file and importing them from a TXT file with the help of the system functions "ExportDataRecords" and “ImportDataRecords".

Note

Exporting and importing with ProSave to CSV format

Individual recipe data records cannot be exported or imported to CSV format with ProSave.

Only complete recipe data can be exported and imported to CSV format with ProSave and transferred to the HMI device.
Runtime is paused while this is done.

See also

Response of the recipe view in Runtime (Page 1134)

Reactions to modifications of the recipe structure

Introduction

Different recipe structures can occur in the following situations:

- In the event of changes during commissioning
- When work is carried out on the machine by the machine manufacturer (retrofit)
- When CSV files are imported, the structure of the CSV file can differ from the recipe structure.

Nevertheless, you can still use any recipe data records already created.

NOTICE
When a tag is renamed, the assignment is lost.

Effects

Handle any structural differences as follows:

- If the old recipe data record or the CSV file contains additional values, these values will be discarded.
- If the old recipe data record or the CSV file contains values of the wrong data type, the configured default value will be used in the recipe data record.
  Example: The recipe data record contains values that show the tank contents and were input as floating point numbers. However, the corresponding recipe tag expects an integer value. In this case, the system discards the transferred value and the configured default value is used.
- If the old recipe data record or the CSV file contains too few values, the configured default value will also be used in the recipe data record.

See also

Description of the advanced recipe view (prior to V13) (Page 1108)

Description of advanced recipe view (as of V13) (Page 1119)

Recipe view basics (Page 1096)

Response of the recipe view in Runtime

Screen change

If you change to another screen and have not yet saved changes to the recipe data in the recipe view, you will be prompted to save the recipe data. The recipe name and the name of the recipe data record are displayed to show which recipe data have not been saved yet.

Create, change, copy or delete recipe data records

If you attempt to create a new recipe data record and a recipe data record already exists, a system alarm will appear on screen.

Operating the recipe view with function keys

The Recipe view can be operated with function keys, e.g. if the HMI device does not have touch functionality. You can assign functions such as "SaveDataRecord" to the function keys on the HMI device.

Display after import of recipe data

Note

Availability

Recipe data import and export is not available for Basic Panels.

If you open the recipe view during the import of recipe data, only the recipe data that is already completely imported will be displayed. The recipe view is not automatically updated with a data import. In order to have a complete view of all the recipe data, do not open the recipe view until the system prompts you that the recipe data has been imported successfully. Alternatively, update the recipe view after successful completion of the import procedure.

Updating tag for recipes and recipe data records

Note

Availability

Tags for recipes and recipe data records are not available for Basic Panels.

The current recipe data record or its number can be saved to a tag, depending on the configuration. The tag will be updated under the following conditions:

- The recipe data record has been loaded.
- The screen with the recipe view was not exited during loading.

This operation may take some time.

See also

Exporting and importing recipe data records (Page 1131)

Description of the advanced recipe view (prior to V13) (Page 1108)

Description of advanced recipe view (as of V13) (Page 1119)

Recipe view basics (Page 1096)

6.1.3 Configuring Recipes

6.1.3.1 Creating and Editing Recipes

General configuration procedure

Carry out the following configuration steps when you create a new recipe:

Step	Description
1	Define the structure of the recipe.
2	Create tags according to the recipe structure. Assign process names to these tags.
3	Create the recipe.

Step	Description
4	Enter the required properties for the recipe: • Language-dependent view name of the recipe • "Coordinated transfer of data records" option Not for Basic Panels: • Recipe storage location • "Synchronize recipe view and recipe tags" option • "Manual transfer of individual modified values (teach-in mode)" option
5	Create the recipe elements and enter the required properties: • Language-dependent view name of the recipe elements • Tag binding of the recipe elements • Standard values and decimal places (power of ten) for the recipe elements
6	Create the recipe data records. Enter the language-specific display names for the recipe data records.
7	Configure a screen with recipe view or a recipe screen.

Note

Basic Panels and OP77A, TP177A (Portrait)

The selection of the storage location is not available for these devices. The recipes are always saved in the internal Flash memory.

Recipe tags cannot be used outside a recipe, e.g. not in I/O fields, not in alarms as trigger tags, not in systems functions as parameters, etc.

Note

Restrictions recipe view and recipe image

Only the simple recipe view is available in Basic Panels and OP77A, TP177A. Recipe images are not available in Basic Panels and OP73, OP77A, TP177A (Portrait).

Creating a new recipe

Introduction

To create a complete recipe, start by creating a new recipe, assign the corresponding recipe elements and then define the associated values in a recipe data record.

Note

Character strings that can be imported

Recipe data that contains semicolons or commas cannot be imported.

Only ever used character strings without semicolons or commas for recipes and recipe data records.

Requirement

- The tags for the recipe have been created.
- The "Recipes" editor is open.

Creating a recipe

Proceed as follows to create a recipe:

1. Click "Add" in the first free row of the table in the "Recipes" editor.
   The new recipe is created and displayed on a line.

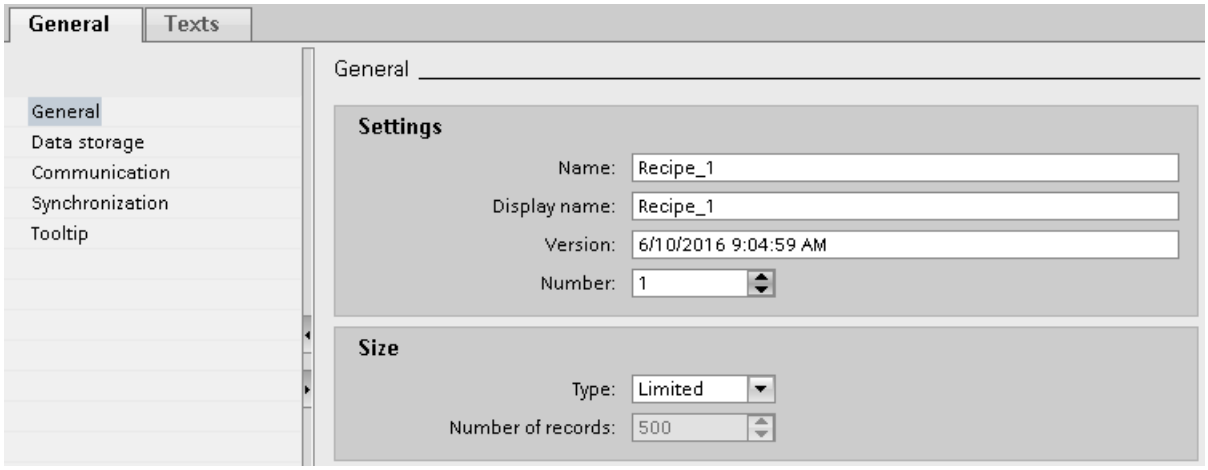

2. Enter a descriptive name for the recipe under "Name" in the "General" area.
   This name identifies the recipe unambiguously within the project.
3. Select "Display name" to enter the language-specific name to be displayed in runtime.
4. Select a recipe number in "Number".
   The number identifies the recipe unambiguously within the HMI device.
   The recipe is automatically assigned a version that indicates the date and time of the last change. As an alternative, you can enter specific information relating to the recipe.
5. Specify the storage location for recipe data records in "Data medium". The options offered depend on the specific HMI device used.

   Note

   Basic Panels

   The selection of the storage location is not available for these devices. The recipes are always saved in the internal Flash memory.

   Recipe tags cannot be used outside a recipe, e.g. not in I/O fields, not in alarms as trigger tags, not in systems functions as parameters, etc.

6. Enter a tooltip that is shown to the operator in runtime.

7. To compare recipe tags which are configured in I/O fields with the recipe view in Runtime, activate "Synchronize recipe view and recipe tags" in the Inspector window under "Properties > Synchronization".

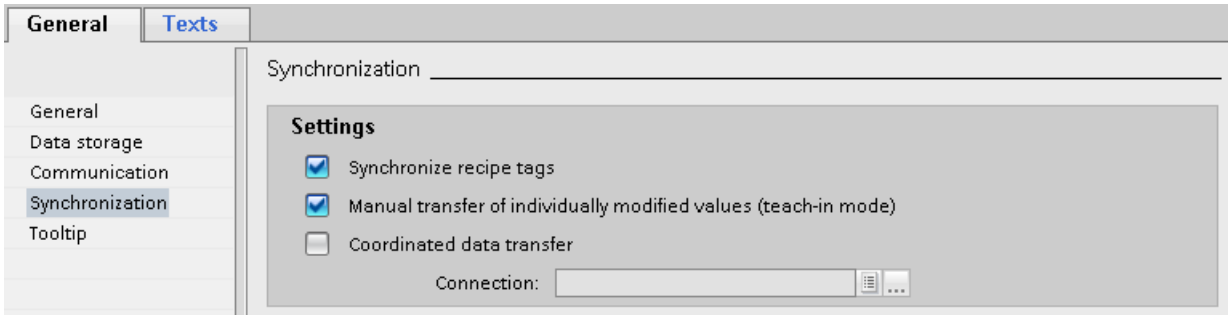

Note

Basic Panels

Because the recipe tags cannot be additionally used in I/O fields in screens for Basic Panels, the "Synchronize recipe view and recipe tags" is not available; you will also not be able to use the "Manual transfer of individual modified values (teach-in mode)" option.

8. Deactivate "Manual transfer of individual modified values (teach-in mode)" to specify that the recipe tags are automatically transferred to the PLC when editing the I/O fields.
9. Activate "Coordinated transfer of data records" to monitor the transfer of recipe data in Runtime using area pointers.
10. Select the appropriate connection to the PLC for coordinated transfer under "Synchronize with".

Create recipe element

To create recipe elements, proceed as follows:

1. Click the "Elements" tab.
2. Click "Add" in the first free line of the table editor.
   A new recipe element is created.
3. Enter a descriptive name for the element under "Name".
   The name identifies the element uniquely within the recipe.
4. Enter a language-specific display name for the element under "Display name".
   The display name appears in the recipe view, for example, in runtime.
5. Select the tag you want to link to the recipe element under "Tag".
   The value of the recipe data element is saved in Runtime in this tag, which is stored in a recipe data record.

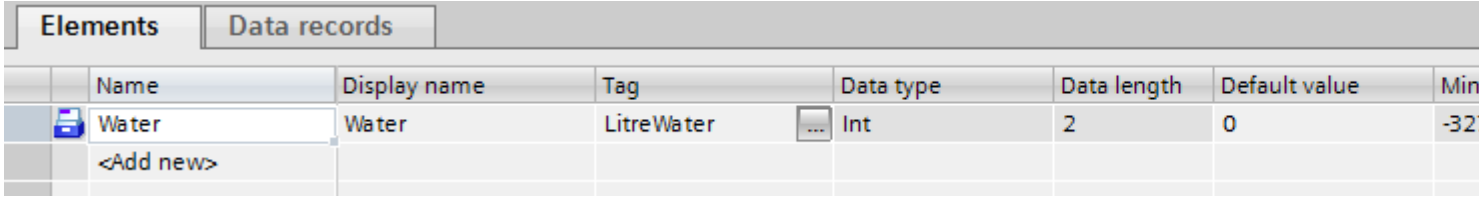

6. Enter a tooltip.
   The tooltip is shown to the operator in Runtime.
7. Under "Default value", enter the value that you want to use as the default entry when you create a new recipe data record.
8. To assign text to a value or range of values, select the relevant text list here. The assigned recipe tag must have the data type of a number. The tag value must be within the range of values of the text list.
   The text stored in the text list is displayed in an output field, for example, in Runtime.
9. Determine exactly how many places a decimal number is rounded to in the "Decimal places" column, e.g. 3 decimal places and vice versa by what power of ten an integer value is multiplied, e.g. 1,000.
   Examples for 3 decimal places: Entering "5" for a recipe element with the "Integer" data type gives the value "5000". Entering "5.6789" for a recipe element with the "Real" data type gives the value "5.679".
10. Create as many recipe entries as needed for the recipe. The maximum number of recipe entries possible depends on the HMI device being used.

Elements | Data records

Name	Display name	Tag	Data type	Data length	Default value
Water	Water	LitreWater	Int	2	0
Concentrat	Concentrat	AmountConcentrat	Int	2	0
Sugar	Sugar	KiloSugar	Int	2	0
Aroma	Aroma	GramAroma	Int	2	0
<Add new>					

Create recipe data record with known recipe values

To create recipe elements, proceed as follows:

1. Click the "Data records" tab.
2. Click "Add" in the first free line of the table editor.
   A new recipe data record is created. The recipe data record has a separate column for every recipe element created in the recipe.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Recipe_data_record_1	Recipe_data_record_1	1	0	0	0	0	
<Add new>							

3. Enter a descriptive name under "Name".
   The name identifies the data record uniquely within the recipe.
4. Enter a language-specific name under "Display name".
   The display name appears in the recipe view, for example, in runtime.
5. Enter a recipe data record number under "Number".
   The recipe data record number identifies the recipe data record uniquely within the recipe.

6. If you already know the recipe values at the configuration stage, you can enter the relevant value for each recipe element.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
<Add new>							

7. Create as many data records as you need for the recipe.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
Nectar	Nectar	2	50	50	10	300	
Juice	Juice	3	5	95	3	100	
<Add new>							

Enter the values in runtime

The following options are available for entering values in the recipe data records at runtime:

- Transfer data directly from the PLC (Teach-in mode)
- Import of values from a CSV file
- Input values on the HMI device

Note

Basic Panels

The import of values is not available for these devices.

Result

The complete recipe is configured.

Recipe data records with date or time stamp

If you use date or time data, make sure that the system setting for time and date on the configuring computer match those on the target system. Example: You load a recipe data record on the target system at 13:55 in which 14 h is stored as the processing time. If it is already 14:05 on the target computer, the recipe will not be processed. If an operator processes the recipe, change information will not written back correctly into the database.

After loading to the target system, check the recipes with date or time stamps on the target system.

Editing a recipe

Purpose

You want to modify, extend or delete parts of a recipe.

Requirement

- You have created at least one recipe.
- The "Recipes" editor is open.

Changing recipe settings

To change the recipe settings, proceed as follows:

1. Select the recipe that you want to change in the "Recipes" editor.
   The Inspector window opens.
2. Change the recipe configuration in the Inspector window.

You change recipe elements and recipe data records in the same way.

Change recipe values

To change recipe values, proceed as follows:

1. Select the recipe whose values you want to change.
2. Click the "Data records" tab.
3. Enter the new values in the value columns.

Adding a recipe element

To add more recipe elements to a recipe, proceed as follows:

1. Select the recipe to which you want to add more elements in the "Recipes" editor.
2. Click the "Elements" tab.
3. Click "Add" in the first free line.
   The recipe element is created.
4. Configure the recipe element.

You add recipe data records in the same way.

Managing recipes

Requirement

- You have created a recipe with recipe elements and recipe data record.
- The "Recipes" editor is open.

Renaming recipes

We distinguish between internal names and display names for recipes, recipe entries and recipe data records.

To rename recipe elements, proceed as follows:

1. Select the recipe that you want to rename.
   The Inspector window opens.
2. Select the "Rename" command from the shortcut menu.
3. Enter the new name.
   You rename recipe elements and recipe data records on the relevant tab in the same way.

Note

The view names in the "Recipes" editor can also be renamed under "Languages & Resources > Project Texts". This possibility is useful when you have already configured in several languages for example.

Copying and pasting recipes

To copy and paste recipes, proceed as follows:

1. Select the recipe that you want to copy.
2. Select the "Copy" command from the shortcut menu.
3. Select the "Paste" command from the shortcut menu in the first free table row.

The copied recipe is pasted into the table. The recipe elements and recipe data records are also copied in the appropriate tab with the recipe.

You also copy the recipe elements and recipe data records on the appropriate tab in the same way.

If a recipe data record of the same name already exists, the name of the copied recipe data record is extended by one digit. This ensures that the name is unique. Recipe data records can only be copied or pasted within the same recipe.

Deleting a recipe

To delete a recipe, proceed as follows:

1. Select the recipe that you want to delete.
2. Select the "Delete" command from the shortcut menu.
   The recipe is deleted.

You delete recipe elements and recipe data records on the relevant tab in the same way.

Note

When a recipe is deleted, the recipe data records contained in the recipe are also deleted.

Note

When you delete a recipe element, the associated values in the recipe data records are also deleted. The assigned tags are retained.

Importing recipes into the configuration and exporting them

Introduction

You can export recipes as a CSV file and import them again into the configuration.

Application case

Recipe data is exported for long-term storage and backup on a computer.

In order to unify and distribute recipe data, export the recipe data to an HMI device. Change the CSV file in Microsoft Excel and import the CSV file to all HMI devices that require the same recipe data.

You want to exchange recipe data between different projects. You change the recipe data in Runtime. To transfer the modified recipe data to WinCC ES, export the recipe data records in Runtime and copy the CSV file to the configuring computer. There you import the CSV file containing the recipe data records into the recipe. And in the opposite direction, you transfer the modified recipe data records to the HMI device in WinCC ES in Runtime.

Requirement

- You have created the recipe.
- The structures of the exported CSV file and the recipe in WinCC ES match; name, number and data type of the recipe elements are identical.
- The "Recipes" editor is open.
- Any semicolons or commas in the import file are used as separators only.

Importing CSV files

To import recipe data records into a recipe, proceed as follows:

1. Select the row which contains the desired recipe in the "Recipes" tab.
2. Select the "Import recipe data" command from the shortcut menu.
   The "Import" dialog box opens.

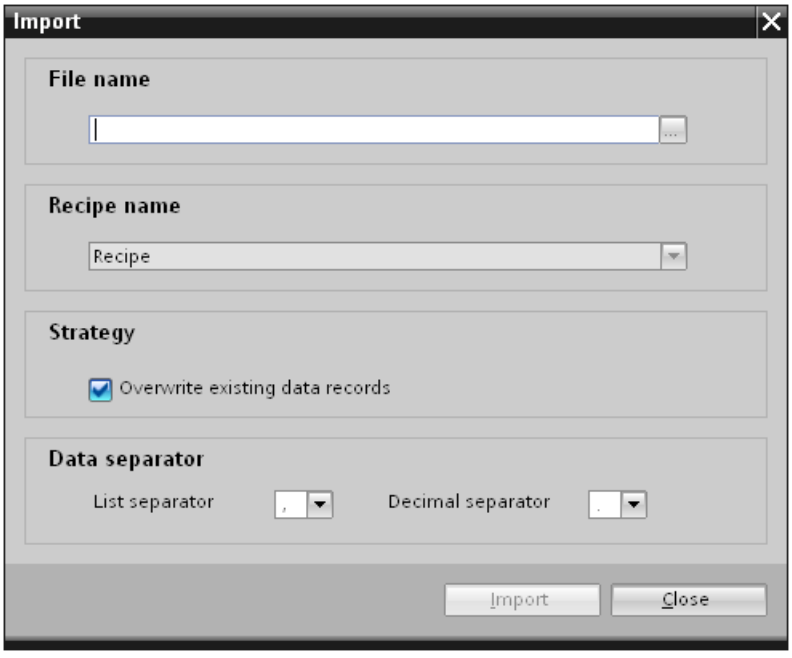

3. To select the desired CSV file, go to "File name".
4. Under "Strategy", specify whether a recipe data record with the same recipe number in WinCC ES should be overwritten.
5. Specify the list separator and decimal separator under "Data separation". The list separator separates individual recipe elements in the CSV file. The decimal separator separates integers and decimal places.

   Note

   Use the same list separator for import as in the CSV file for export.

6. Click "Import" to start the operation.

Result

The recipe will be amended by recipe data records from the CSV file that have not yet been entered. If the option is selected, existing recipe data records will be overwritten.

An error message is shown in the "Info" of the Inspector window if the structure of the recipe does not match the structure of the CSV file.

Exporting CSV files

1. Select the row which contains the recipe to export in the "Recipes" tab.
2. Select the "Export recipe data" command from the shortcut menu.
   The "Export" dialog box opens.

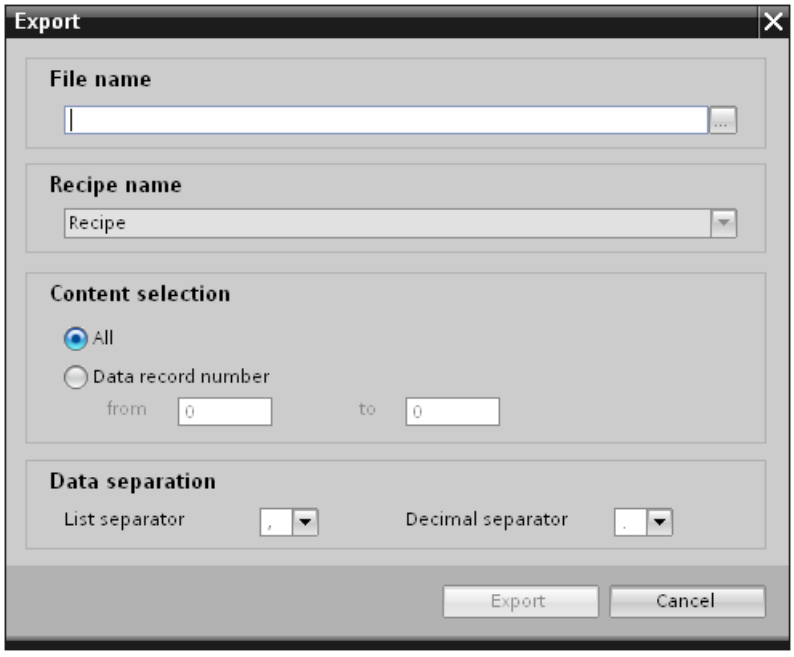

3. Under "File name", specify the path of the CSV file.
4. Select all recipe data records under "Selection content", or restrict the selection to specific record numbers of the recipe data.
5. Specify the list separator and decimal separator under "Data separation". The list separator separates individual recipe elements in the CSV file. The decimal separator separates integers and decimal places.
6. Click "Export" to start the operation.

Result

The configuration is exported as a CSV file.

6.1.3.2 Configuring the display of recipes

Configuring the simple recipe view

Requirement

- The recipe has been created.
- The "Screens" editor is open.
- The screen has been created and opened.

NOTICE
Data loss with several recipe views in the screen Applies only to Basic Panels, OP73, OP77A, TP177A and TP177A (Portrait): If two or more recipe views show the same recipe in a screen, you have a conflict when accessing the data. The result is data loss and unpredictable status of recipe data. Make sure the operators do not select and edit the same recipe in different recipe views. • Display only one recipe in a recipe view. • Display a different recipe in each recipe view.

Procedure

To configure a simple recipe view, proceed as follows:

1. Paste the recipe view into the screen. You will find the recipe view under "Controls" in the "Tools" task card.
2. Only in devices which also support the extended recipe view: Activate "Simple view" under "Properties > Display > Mode".

3. If you want to display only the recipe data records of a specific recipe in the recipe view, select the specific recipe under "Properties > General > Recipe".

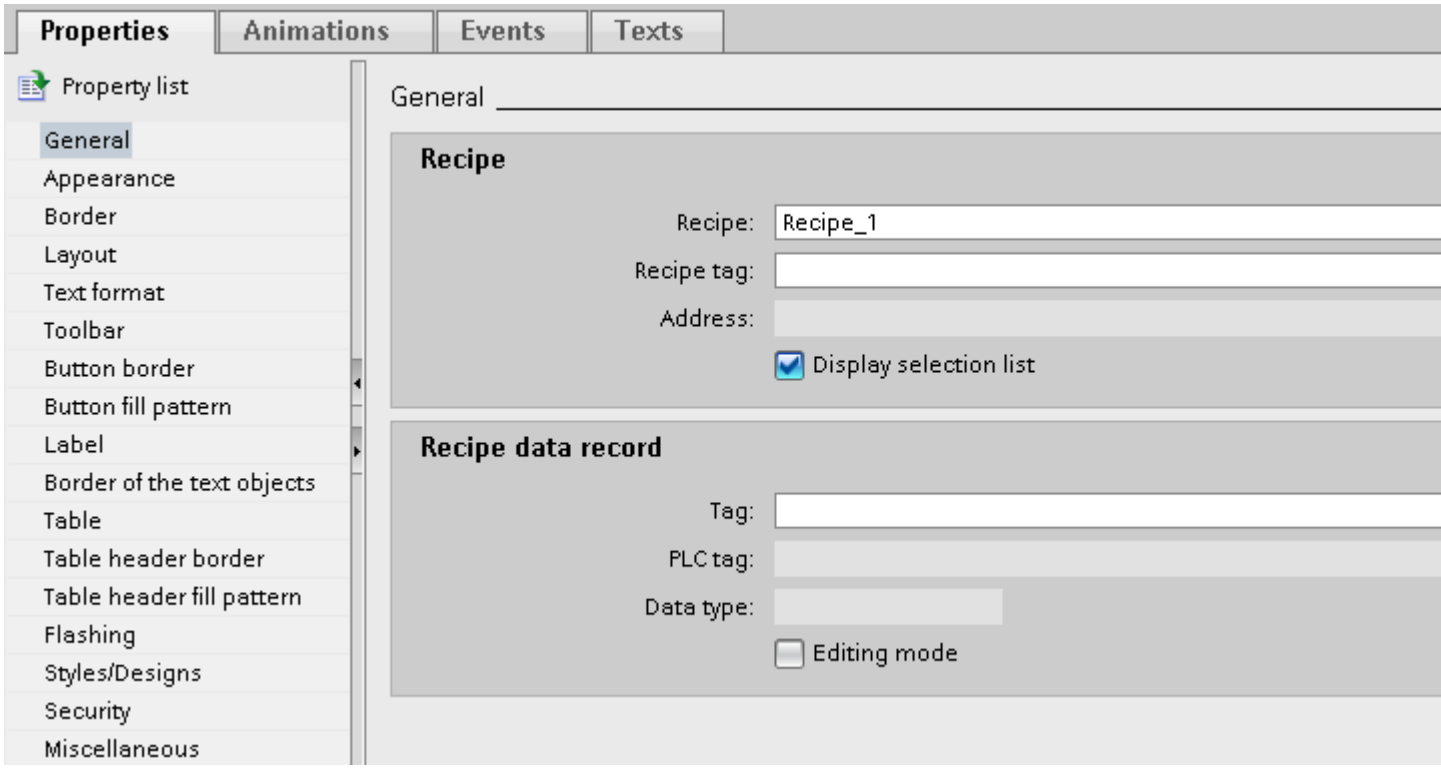

4. If you only want to display the recipe data in the recipe view, deactivate "Processing mode" in the "Recipe data record" area.
5. You can define additional options for the recipe view under "Properties > Appearance" and "Properties > Layout".
6. Under "Properties" > Toolbar" specify which menu commands are available in the recipe view in Runtime.

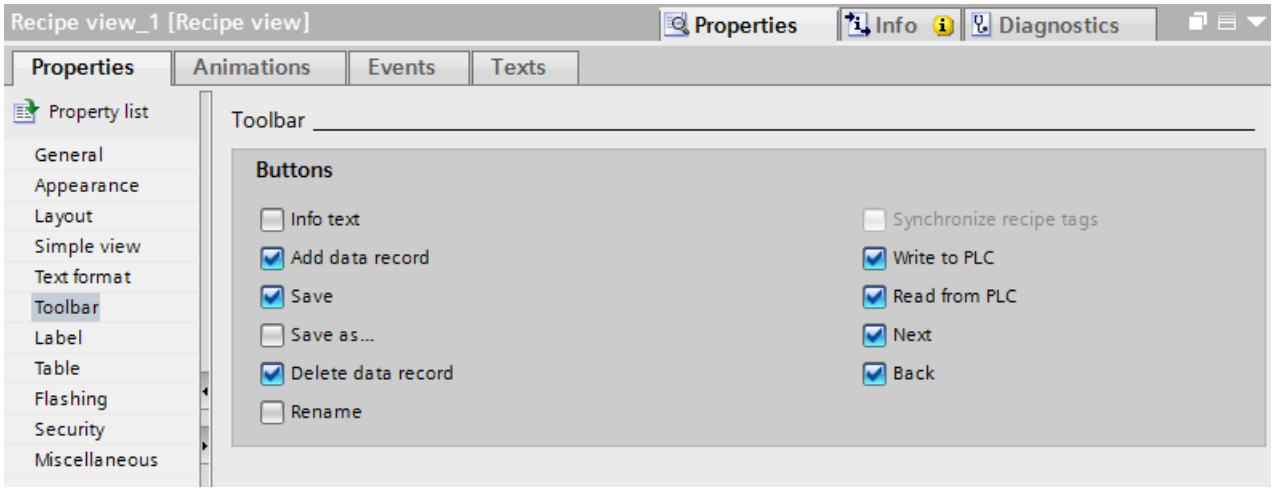

Result

The simple recipe view is configured. You can use the recipe view to display and edit recipe data during runtime.

Deactivation of the editing mode in "Properties > Properties > General" has no impact on the toolbar icons. The buttons activated under "Properties > Toolbar" can still be used even if editing mode is disabled.

Configuring the Advanced Recipe View (RT Advanced, Panels, Comfort Panels)

Requirement

- The recipe has been created.
- The "Screens" editor is open.
- The screen has been created and opened.

Procedure

To configure an advanced recipe view, proceed as follows:

1. Paste the recipe view into the screen. The recipe view is found in the task card "Tools" > "Controls".
2. Select "Properties > Display > Mode > Advanced view" in the Inspector window.

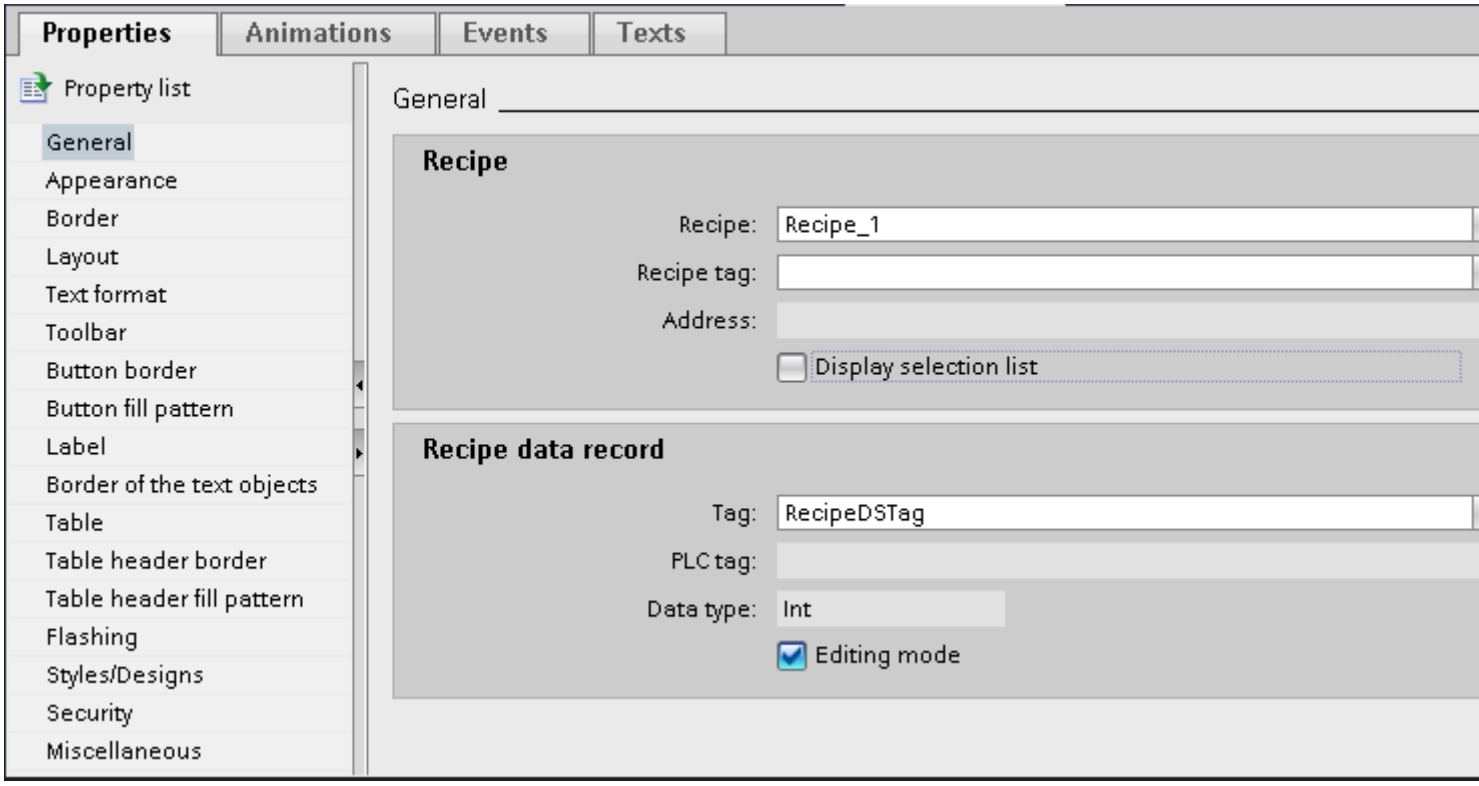

3. Select the required settings from the "General" group in the Inspector window.
   - If you want to display only the recipe data records for a particular recipe in the recipe view, select the recipe under "Recipe" in the "Recipes" area.
   - If you want to save the recipe name or the recipe number in a tag, select the tag under "Recipe tag" in the "Recipe" area.
   - If you want to save the recipe data record name or number in a tag, select the tag under "Tag" in the "Recipe data record" area.
   - If you only want to display the recipe data in the recipe view, deselect "Edit mode".
   - If you only want to use the recipe view to select recipes, disable "Display table" under "Properties > Table".
4. You can define additional options for the recipe view under "Properties > Appearance" and "Properties > Layout".
5. If you want to change the label in the recipe view, enter a suitable text under "Properties" > "Label".
6. Under "Properties" > Toolbar" specify which buttons are available in the recipe view in Runtime.

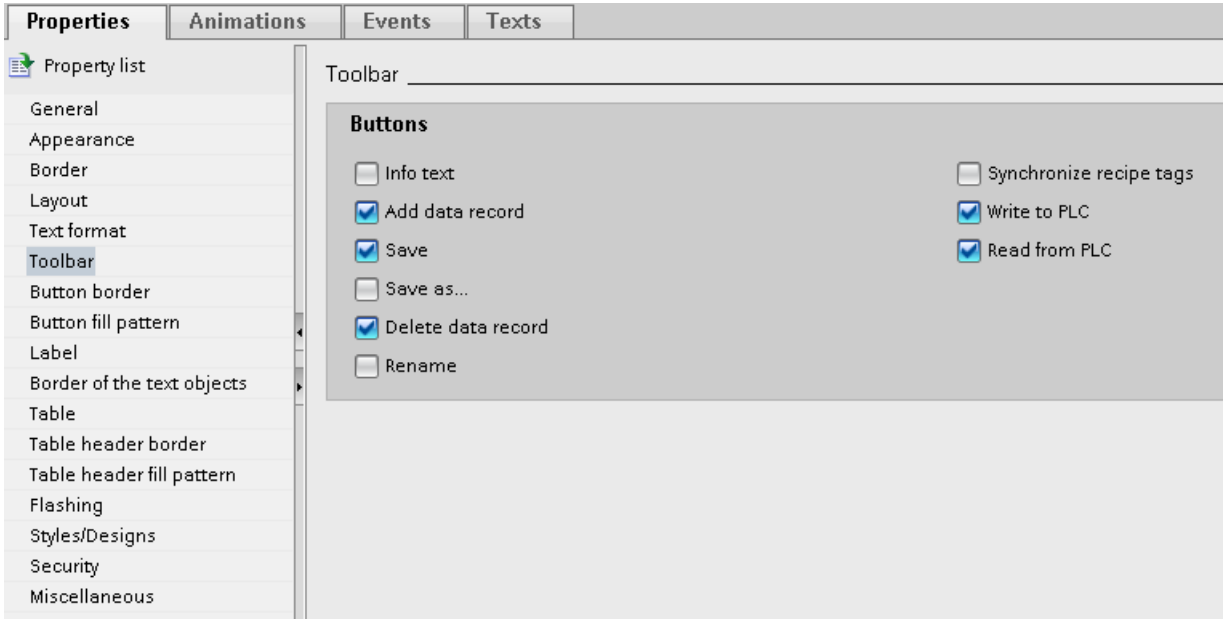

Note

If you select the "Edit" command in the context menu of the recipe view, the recipe view becomes active. To activate the recipe view, the zoom factor must be set to 100%.

You can set the column width and the position of the "Entry name" and "Value" columns in active mode.

Result

The recipe view is configured. You can use the recipe view to display and edit recipe data during runtime.

Deactivation of the editing mode in "Properties > Properties > General" has no impact on the toolbar icons. The buttons you activated in "Properties > Toolbar" can still be used even if editing mode is disabled.

Configuring a recipe screen (RT Advanced, Panels, Comfort Panels)

Introduction

The recipe screen is a screen in which you configure a customized screen form in the "Screens" editor. You create the screen form from input/output fields and other display and operator control objects. System functions are used to configure the recipe functionality, such as saving recipe data records.

Note

Availability

Recipe screens cannot be created for basic panels and OP73, OP77A and TP177A (Portrait).

Application

You can spread recipe data records containing lots of entries across several screens. For example, for each plant you can configure a screen containing the associated screen forms for the recipe data records.

You can visually simulate your machine in a screen using graphical screen objects. This enables you to display parameter settings more clearly by positioning I/O fields immediately next to machine elements, such as axes or guide rails. You can use this to produce a direct reference between the values and the machine.

Requirement

- You have created the recipe.
- The "Screens" editor is open.

Procedure

To configure a recipe screen, proceed as follows:

1. Configure the screen and create the I/O fields for the input mask of the recipe.
   You can create multiple screens to suit the size and complexity of the recipe.
2. Configure the I/O fields with the tags you have linked to the recipe element.
3. Configure I/O fields for selecting recipes and recipe data records.

Alternatively:

1. Configure a recipe view as a selection list for recipe data records and recipes.
2. Hide the buttons that are not required in the recipe view.
3. Configure the system functions for editing recipe data records on the configured control elements.
   Control elements are configured buttons in the screen or function keys on the HMI device. You will find the system functions for editing recipe data records under "Recipes" and "Keyboard operation for screen objects".

Result

The recipe screen is created.

6.1.4 Examples

6.1.4.1 Example of creating a recipe (RT Advanced, Basic Panels, Panels, Comfort Panels)

Task

In this example, you create three recipes for a fruit juice mixing machine. The fruit juice mixing machine produces drinks with "orange", "apple" and "tropical" flavors. You create a recipe for each flavor.

Each recipe contains a recipe data record for the following mixing ratios:

- Beverage
- Nectar
- Juice

Settings

The settings relate to an HMI device which is connected to a SIMATIC S7-300 or SIMATIC S7-400.

In this example, you will need the following tags, recipes, recipe entries and recipe data records:

Tags:

Name	PLC connection	Address	Type
Liter water	Yes	DB 120, DBW 0	Integer
Liter concentrate	Yes	DB 120, DBW 4	Integer
Kilo sugar	Yes	DB 120, DBW 8	Integer
Gram flavoring	Yes	DB 120, DBW 12	Integer

Recipes:

- Orange
- Apple
- Tropical

Recipe entries:

Recipe element	Associated tag
Liter water	Liter water
Liter concentrate	Liter concentrate
Kilo sugar	Kilo sugar
Gram flavoring	Gram flavoring

Recipe data records for drink, nectar and juice:

Data record name	Liter water	Liter concentrate	Kilo sugar	Gram flavoring
Beverage	30	70	45	600
Nectar	50	50	10	300
Juice	5	95	3	100

Procedure

To create a recipe, proceed as follows:

1. Create the following tags with the settings specified above: "LiterWater", "LiterConcentrate", "KiloSugar" and "GramFlavoring".
2. Create the "Orange", "Apple" and "Tropical" recipes with the settings indicated above. Create the recipe entries in each recipe.

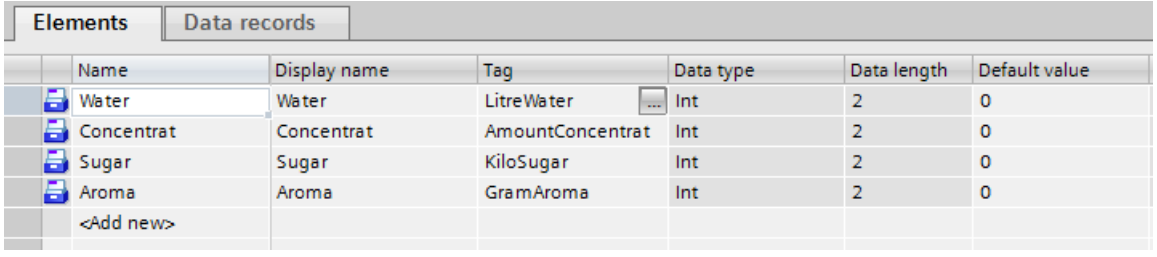

3. Not for Basic Panels: Configure each recipe so that you can synchronize the recipe data records between the recipe screen and recipe view. The values of the recipe tags should not be transferred automatically to the PLC.
   You will have to make the following settings in the Properties dialog for the recipe concerned: Under "Properties > Options":
   – Activate the "Synchronize recipe view and recipe tags" option.
   – Activate the "Manual transfer of individual modified values (teach-in mode)" option.
4. Create the data records indicated above in each recipe. Enter the values indicated above in each of the data records.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
Nectar	Nectar	2	50	50	10	300	
Juice	Juice	3	5	95	3	100	
<Add new>							

Result

The "Orange", "Apple" and "Tropical" recipes have been created.

See also

Configuration options of the advanced recipe view (V13 or higher) (Page 1122)

6.1.4.2 Example of configuring a recipe screen (RT Advanced, Panels, Comfort Panels)

Task

In this example, you create a recipe screen for the visualization of values of the fruit juice mixing machine. You use a recipe view to select the recipes and their associated recipe data records.

You should be able to use the following functions with the buttons:

- "Load" Button
  The selected recipe data record is loaded from the recipe memory on the HMI device and displayed in the recipe screen.
- "Save" Button
  The displayed recipe tags are saved in the recipe memory of the HMI device.

- "Data to PLC" button
  The displayed tags are transferred to the PLC.
- "Data from PLC" button
  The current recipe data record in the PLC is transferred to the recipe variables and displayed on the HMI device.

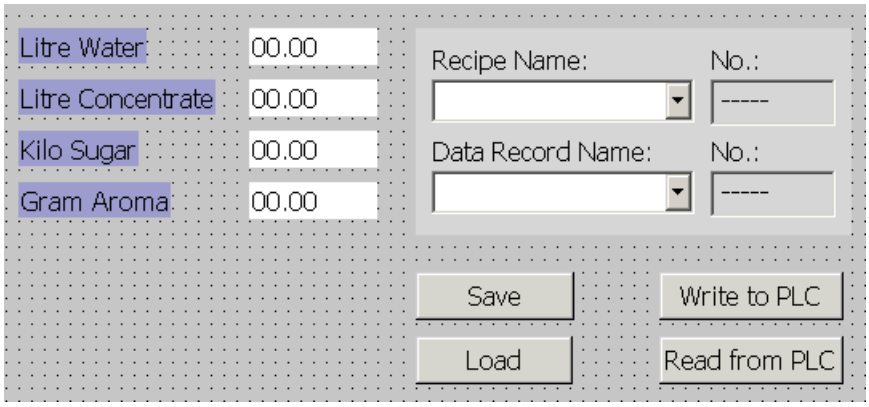

Requirement

- The "Creating a recipe" sample application has been carried out.
- You have created and opened the "Fruit juice mixing plant" screen.

Settings

In this example, you need the following tags and buttons with the indicated settings:

Tags:

Name	PLC connection	Type
RecipeNumber	No	Integer
Data record number	No	Integer

Buttons:

Labeling	Configured event	System function
Load	Click	LoadDataRecord
Save	Click	SaveDataRecord
Data to PLC	Click	SetDataRecordTagsToPLC
Data from PLC	Click	GetDataRecordTagsFromPLC

Procedure

To configure a recipe screen, proceed as follows:

1. Drag-and-drop the "Liter water", "Liter concentrate", "Kilo sugar" and "Gram flavoring" tags from the object view to the "Fruit juice mixing machine" process screen.
   Four IO fields are created and linked by the specified tags.
2. Add a recipe view containing selection fields for the recipe name and data record name only.
   Make the following settings in the Inspector window for the recipe view:
   – Select the "Advanced view" display type under "General".
   – Deselect "Edit mode" under "Properties > General" and "Display table" under "Properties > Table".
   – Connect the "Recipe tag" field to the "RecipeNumber" tag.
   – Connect the "Tag" field to the "DataRecordNumber" tag.
   – Disable all the buttons under "Properties > Buttons".
3. Assign the above settings to four buttons. Transfer each "Recipe number" and "Data record number" tag as a parameter for the recipe number and recipe data record number.

Result

You can select the recipe and the associated recipe data record from the recipe view and modify the recipe values at runtime. You can load, save and transfer the recipe data records using the buttons.

6.1.4.3 Scenario for Entering Recipe Data Records in Runtime (RT Advanced, Panels, Comfort Panels)

Objective

You want to enter production data on the HMI device without disturbing the process that is currently underway. Therefore, the production data should not be transferred to the PLC.

Requirement

- The recipe has been created.
- The recipe has the following settings:
  – "Synchronizing recipe view and recipe tags" is activated or deactivated.
  – If "Synchronize recipe view and recipe tags" is activated, "Manual transfer of individual modified values (teach-in mode)" has to be activated.
    This will prevent the recipe tags being transferred automatically between the HMI device and PLC.
- A recipe screen or a screen with recipe view is available.
- There is an operating element for saving the recipe data records.

Sequence

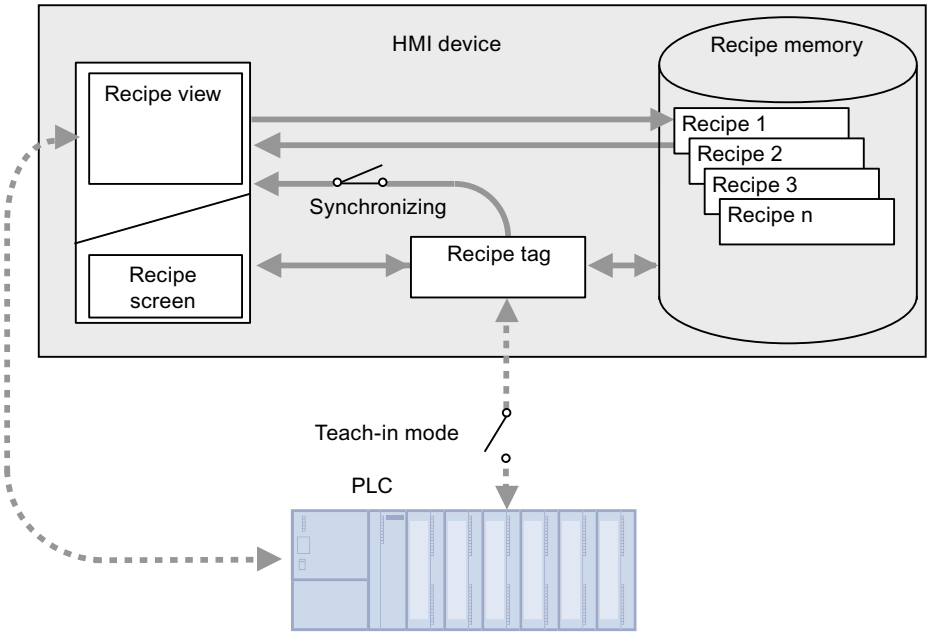

1. Enter the production data in the recipe view or recipe screen.
2. Save the modified recipe data record.

Transfer the recipe data to the PLC

The configuration may provide operating elements for transferring recipe data to the PLC.

6.1.4.4 Scenario for a manual production sequence (RT Advanced, Panels, Comfort Panels)

Objective

A reading device connected to the PLC reads a bar code on the work piece to be processed. The recipe data record names correspond to the respective bar code names. This will enable the PLC to load the necessary recipe data record from the storage medium of the HMI device. The recipe data record is displayed for inspection on screen.

You want to be able to correct the transferred production data online, if necessary.

Requirement

- You have created the recipe.
- The recipe has the following settings:
  - "Synchronizing recipe view and recipe tags" is activated.
  - "Manual transfer of individual modified values (teach-in mode)" is deactivated.

  Note

  The changes are immediately transferred to the PLC

- There is a recipe screen available.
  There may also be an operating element for saving the recipe data records in the recipe screen.

Sequence

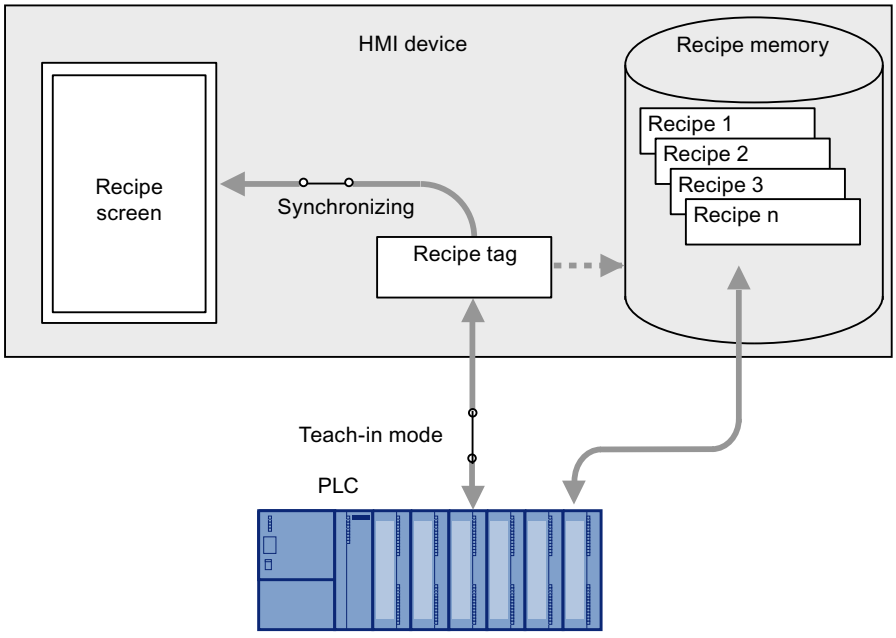

Behavior when the recipe view is used

If the recipe view is used, it is not possible to transfer changes immediately. You must use the operating element to transfer the recipe data record to the PLC.

6.1.4.5 Scenario for an Automatic Production Sequence (RT Advanced, Panels, Comfort Panels)

Objective

You want production to be executed automatically. The production data is to be transferred directly to the PLC, either from the recipe memory in the HMI device or from an external storage medium. The screen display is not necessary.

Requirement

- You have created the recipe.
- The recipe has the following settings:
  - "Coordinated transfer of data records" is activated.
    The production data is transferred to the PLC, so a coordinated transfer with the PLC in necessary to prevent the data from being accidentally overwritten.

Sequence

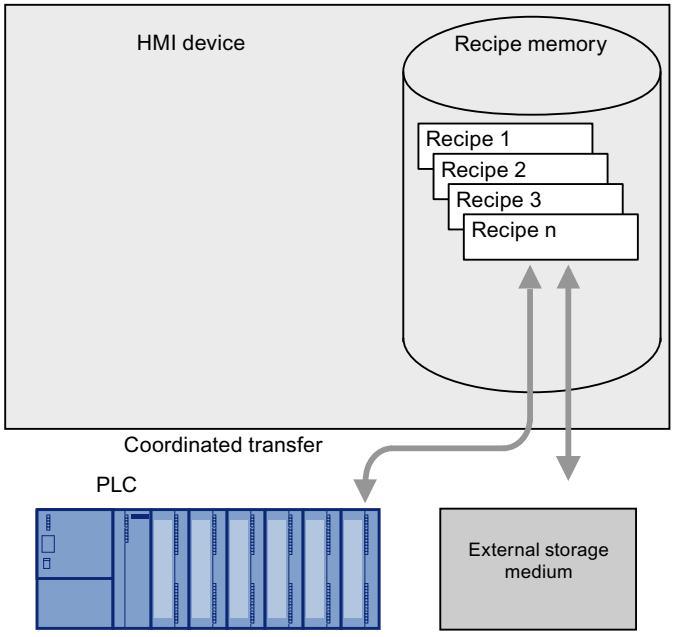

Implementation

You can control the flow of data in the following ways:

- The control program controls the automatic transfer via control jobs or via WinCC system functions, if necessary.
  The sequence is controlled via the status information in the mailbox and via return values from the functions used.
- One or more scripts control the automatic transfer via WinCC system functions.
  The sequence can be checked using the values returned by the functions used.

You can implement the automatic production sequence with available system functions:

- "ImportDataRecords"
  This function loads data records from a *.CSV file into the recipe memory of the HMI device.
- "SetDataRecordToPLC"
  This function transfers a data record from the HMI device's recipe memory to the PLC.

6.2 Working with recipes (RT Professional)

6.2.1 Principles (RT Professional)

6.2.1.1 Definition and applications (RT Professional)

Introduction

Recipes are collections of related data, for example, machine parameter settings or production data.

Examples:

- Machine parameter settings that are needed to convert production to a different product variant.
- Product components that result in different compositions for different end products.
- Data from technical processes that is continuously stored at longer intervals.

A recipe has a fixed data structure. The structure of a recipe is defined once at the configuration stage. A recipe contains recipe data records. These differ in terms of their values, but not their structure.

The project data is stored in an SQL database. Recipe data records are always transferred completely and in a single pass between the HMI device and the PLC. You can import the production data in Runtime additionally by a CSV file.

Using recipes

Recipes can be used in the following situations:

- Manual production
  You select the required recipe data and display it on the HMI device. You modify the recipe data as required and save it on the HMI device. You transfer the recipe data to the PLC.
- Automatic production
  The control program starts transfer of the recipe data between the PLC and HMI device. You can also start the transfer from the HMI device. Production is then implemented automatically. It is not essential to display or modify the data.
- Saving project data
  You are using a recipe which is not interconnected with the PLC to save project data. Project data saved to the recipe can be accessed using scripts in Runtime.

Views

Views relate data from existing recipes that have at least one common feature. They are used to compile data from the recipes.

Display recipes and views

The recipes and views can be displayed and edited on the HMI device in the following ways:

- A recipe view within a process screen
- A recipe screen

Entering and modifying the recipe data

You enter the data in the individual recipe data records and modify it as required. The following options are available:

- Enter the data during configuration
  If the production data exists already, enter it in the "Recipes" editor during recipe configuration.
- Entering the data in Runtime
  Frequent adaptations of production data should be handled directly in Runtime.
  - Enter the data directly on the HMI device.
  - Set the parameters directly on the machine. You then transfer the data from the PLC to the HMI device and save it in the recipe.
  - You import the recipe data from already swapped-out CSV files.

See also

Recipe view in reports (Page 1302)

Recipe report (Page 1303)

6.2.1.2 Recipe structure (RT Professional)

Introduction

The basic structure of a recipe is illustrated with reference to a filling station in a fruit juice plant.

There may be several different recipes in an HMI device. A recipe can be compared to an index card box that contains several index cards. The index card box contains several variants for manufacturing a product family. The complete data of a manufacturer variant are contained on one file card.

Example:

In a soft drinks production plant, a recipe is needed for different flavors. Drink variants include fruit juice drink, juice, and nectar.

Recipe

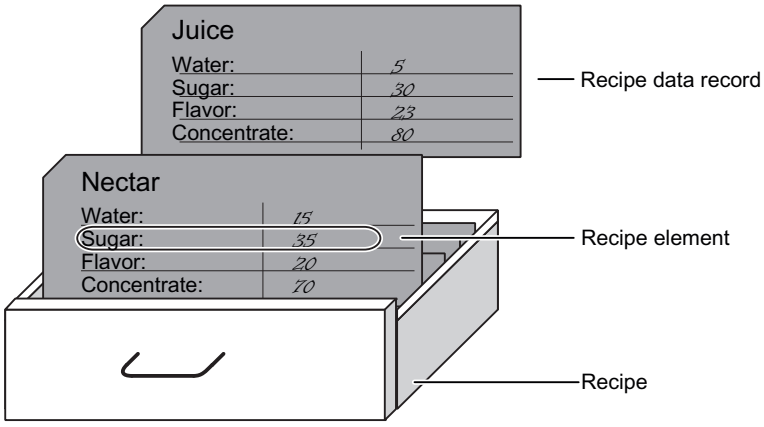

Recipe data records

Each index card represents a recipe data record needed to manufacture a product variant.

Recipe elements

Each index card in a drawer is printed identically. All the index cards contain fields for the different ingredients. Each field corresponds to a recipe element. Each recipe data record contains the same elements. The records differ, however, in the value of the individual elements.

Example:

All beverages contain the same elements:

- Water
- Concentrate
- Sugar
- Flavoring

The records for juice drink, fruit juice, or nectar differ, however, in the quantity of sugar used in production.

Recipe	Recipe data record	Recipe elements	Value
Drinks	1	Item	Juice
		Water	5
		Sugar	30
		Flavoring	23
		Concentrate	80
		Coloring	F30
	2	Item	Nectar
		Water	15
		Sugar	35
		Flavoring	20
		Concentrate	70
		Coloring	F35

Recipe queries

A recipe query is a recipe whose elements are made up of various recipes. Example:

```
Recipe query Element1 = Recipe1_Recipe element 3
Recipe query Element2 = Recipe3_Recipe element 7
Recipe query Element3 = Recipe2_Recipe element 10
```

The used recipes must already exist.

A recipe query is displayed in a recipe view. Instead of having its own data records, however, a recipe query combines all the recipe data records of the recipes used. Example: If there are 10 recipe data records for each recipe in the example above, the recipe query covers 10*10*10 = 1000 data records. You can define a relation in the recipe query to limit these data records. Example:

```
Recipe query Element2 = Recipe query Element3
```

The relation acts as a filter for the recipe query (see also example below).

Example

A "Label" recipe query is created from the "Drinks" and "Order" recipe for label printing.

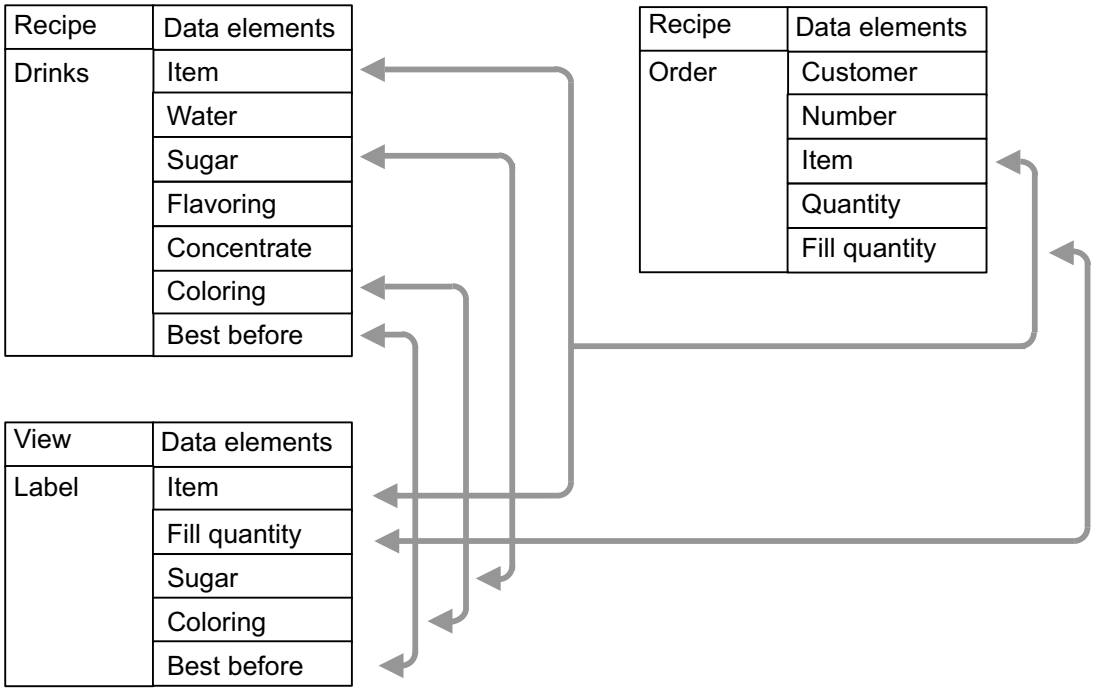

The figure below shows the "Label" recipe query. The necessary information for the label is contained in the "Drinks" and "Order" recipes. The recipe elements "Article", "Sugar", "Colorant" and "Best before" are transferred to the "Label" recipe query from the "Drinks" recipe, the recipe elements "Article" and "Net weight" from the "Order" recipe.

The "Article" recipe element is double:
`Drinks.Article, Order.Article, Drinks.Water, Order.Net weight...`

The recipe view first shows every combination of these data in its own line. However, the recipe data of the ordered article and not of another article must be on the label which is ensured by the following condition:
`Drinks.Article = Order.Article`

The recipe view is filtered and only the ordered articles with their recipe data displayed line by line.

6.2.1.3 Displaying Recipes in Runtime (RT Professional)

You have the following options of displaying recipe data in Runtime:

- In I/O fields at any position in plant screens
- In a recipe view in table format

You can connect a recipe view to a selected recipe and/or recipe query during configuration. The recipe view can only access this recipe and/or recipe query.

You can assign various authorizations for access.

6.2.1.4 Communication with the PLC (RT Professional)

Introduction

Data are transmitted in two ways between recipes and the PLC:

- By recipe tags with the aid of control tags
- By message frames with the aid of raw data tags

"Control tags" communication type

A complete recipe data record is transferred to the recipe tags and vice versa by control tags. The recipe tags have a communication connection to the PLC

"Raw data tag" communication type

The PLC and the HMI device exchange raw data tags as message frames with a fixed structure.

Raw data tags offer the following advantages:

- Several data records transferred between the PLC and the HMI device.
- Parts of a recipe data record transmitted.
- Use the same raw data tag for several recipes.

See also

Overview (Page 1188)

6.2.1.5 "Recipes" Editor (RT Professional)

Introduction

You can create, configure and edit the following objects in the "Recipes" editor.

- Recipes and the corresponding recipe elements and recipe data records.
- Recipe queries and the corresponding recipe query elements and relations

The "Recipes" editor also allows you to enter values in recipe data records.

Structure of the "Recipes" editor

The upper part of the working area in the "Recipes" editor contains the "Recipes" and "Recipe queries" tabs.

Create the recipes and/or recipe queries in the "Recipes" and "Recipe queries" tabs and configure these in the Inspector window.

The tab called in the upper area of the working area determines the tabs to display in the bottom area of the working area.

"Recipes" tab

- Recipe elements
  Define the recipe elements of the selected recipe using the table cells provided here. You can move recipe elements within the table with the shortcut menu commands, "Up" and "Down".
- Recipe data records
  Define the values of the data records of the selected recipe using the table cells provided here.

"Recipe queries" tab

- Recipe query elements
  Here, you define the recipe query elements of the selected recipe query via the table cells. You move recipe queries and recipe query elements within the table using the "Up" and "Down" shortcut menu commands.
  You define the relations between the recipe query elements in the "Conditions" field.

Inspector window

In the Inspector window, you configure each element that is selected in one of the tables, e.g. a recipe query or recipe data record.

Recipe settings

The following settings are available for recipes:

Setting	Description
Recipe name	This is a unique identification for the recipe within the HMI device.
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Enter meaningful names or designations which you can assign directly to a product, such as "Fruit_juice_Orange".
Recipe number	This is a unique identification for the recipe within the HMI device. For the "Raw data tag" communication type, the "PLC" field in the message frame must correspond to a valid recipe number.
Version	Information about the recipe. The date and time of the last change to the recipe is set by default.
Path	Defines the storage location for recipes. The recipes are not saved in a file but in a database.
Type	You can limit the recipe data records to a maximum number you select yourself.
Maximum number of data records	Maximum number of data records in a recipe. This limits the required recipe memory of the HMI device. The value "0" appears for the "Unlimited" size type.
Communication type	Define how the HMI device and PLC exchange recipe data as a rule: • No communication • Raw data tags • Tags
Raw data tag	Tag which transfers the recipe data as a frame between the PLC and the HMI device in the "raw data tag" communication type.

Setting	Description
Control tag task, ID, field, value	Four tags which control the data transmission between the recipe and recipe tags and therefore the PLC in the "tags" communication type.
For reading For writing	Name of the authorization which has read, write access to the recipe in runtime.
Last access Last user	Change information or access to a recipe is displayed in Runtime.
Tooltip	Infotext on recipe for the configuration engineer

Recipe element settings

The following settings are available for recipe elements:

Setting	Description
Recipe element name	Identifies a recipe element uniquely within the recipe. Enter meaningful names or labels that you can allocate uniquely, such as axis labels on a machine or ingredients such as "Flavoring".
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Enter meaningful names or designations which you can assign directly to a product, such as "Fruit_juice_Orange".
Recipe tag	An assigned tag in Runtime stores the current value of the recipe element in the recipe data record.
Data type	Data type of the recipe tag.
Data length	Data length of the recipe tag, depending on the data type.
Default value	This is used as the default entry when you create a new recipe data record.
Minimum value	The smallest representable value of a number-based recipe tag, depending on the data type.
Maximum value	The largest representable value of a number-based recipe tag, depending on the data type.
For reading For writing	Name of the authorization which has read, write access to the recipe in runtime.
Value required	The recipe element has a value other than zero.
Unique values	A value for the recipe element may only occur once in all recipe data records. Example recipe element "Order number".
Fast search	The value of the recipe element is indexed for faster searching.
Tooltip	Infotext on recipe element for the configuration engineer

Recipe data record settings

The following settings are available for recipe data records:

Setting	Description
Recipe data record name	Identifies a recipe data record uniquely within the recipe.
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Enter meaningful names or labels that you can assign directly to a product, such as "Product_numbers".

Setting	Description
Recipe data record number	Identifies a recipe data record uniquely within the recipe.
Recipe element name	Name of the created recipe element The values which the recipe element has in the different recipe data records (lines) are saved under this name. A value forms a recipe data record respectively with the values of the other recipe elements in a line. You can already save the values during the configuring.
Last access Last user	Change information or access to a recipe data record is displayed in Runtime.
Comment	Comment about the recipe data record

6.2.1.6 SQL language (RT Professional)

SQL (Structured Query Language) is an efficient and widely distributed database language. The SQL language is used for database queries in the functions of the WinCC script language. You will find further information in the relevant technical manuals.

In some standard functions and some functions in the "Recipes" editor, you must specify conditions in the database language SQL to specify the data records to be edited. A few examples of how provide an SQL instruction are given below:

Note

Name convention

Suffixes, extensions and the name of a recipe, recipe query or a recipe element may only consist of letters, numbers and underscores "_" and may not exceed a length of 20 characters. The first character must always be a letter.

- `FieldA > '1992-12-31 23:45:12.12'`
  The instruction selects all data records whose value in the "FieldA" column is greater than the specified value. FieldA is is of the data type DB_TYP_TIME.
- `FieldB like 'tank%'`
  Then the data records for which the "FieldB" column contains the value "tank1" or "tank4" or "tank12" etc. are selected for example. FieldB is of the data type DB_TYP_CHAR.
- `FieldC > 100`
  With a condition of this type, all data records are selected for which the "FieldC" column contains a value greater than 100. FieldC is of the data type DB_TYP_INTEGER.
- `BETWEEN FieldC = 20 AND FieldC = 200`
  The statement selects all data records for which the "FieldC" column contains a value between 20 and 200. FieldC is of the data type DB_TYP_INTEGER.
- `FieldD`
  Sorting then takes place by column "FieldD".
- `FieldE desc`
  Sorting then takes place by column "FieldE" in reverse alphabetical order (descending order).

See also

SQL keywords (Page 1169)

6.2.1.7 SQL keywords (RT Professional)

SQL keywords

Note

Do not use the following terms and keywords of the SQL language as names for a recipe, recipe query or recipe element:

- "Archive"
- "View"
- "Field"
- "ViewCol"

Keywords of the SQL language			
add	all	alter	and
any	as	asc	begin
between	binary	break	by
call	cascade	cast	char
char_convert	character	check	checkpoint
close	comment	commit	connect
constraint	continue	convert	create
cross	current	cursor	date
dba	dbspace	deallocate	dec
decimal	declare	default	delete
desc	distinct	do	double
drop	else	elseif	encrypted
end	endif	escape	exception
exec	execute	exists	fetch
first	float	for	foreign
from	full	goto	grant
group	having	holdlock	identified
if	in	index	inner
inout	insert	instead	int
integer	into	is	isolation
join	key	left	like
lock	long	match	membership
message	mode	modify	named
natural	noholdlock	not	null
numeric	of	off	on

Keywords of the SQL language			
open	option	options	or
order	others	out	outer
passthrough	precision	prepare	primary
print	privileges	proc	procedure
raiserror	readtext	real	reference
references	release	remote	rename
resource	restrict	return	revoke
right	rollback	save	savepoint
schedule	select	set	share
smallint	some	sqlcode	sqlstate
start	stop	subtrans	subtransaction
synchronize	syntax_error	table	temporary
then	time	tinyint	to
tran	trigger	truncate	tsequal
union	unique	unknown	update
user	using	validate	values
varbinary	varchar	variable	varying
view	when	where	while
with	work	writetext	

See also

SQL language (Page 1168)

6.2.2 Configuring Recipes (RT Professional)

6.2.2.1 General configuration procedure (RT Professional)

Configuring steps

Follow the steps outlined below to create a recipe:

Step	Task
1	Define the structure of the recipe.
2	Define the communication mode for the recipe.
3	Define the control tags or raw data tags according to the communication mode selected.
4	Create the recipe and define its properties.
5	Create the recipe elements in accordance with the structure planned.
6	Create the recipe data records.

Step	Task
7	Configure views of multiple recipes, if required.
8	Configure a screen which contains a recipe view and interconnect this recipe view with the recipe or view. Or: Configure a screen which contains I/O fields and buttons. Configure button functions for reading and writing the values of recipe data records.

Note

Runtime API

The Runtime API provides you with extensive uaArchive functions to edit recipes and recipe queries (see section "Interfaces > Runtime API", subsection "AUTOHOTSPOT").

6.2.2.2 Creating a New Recipe (RT Professional)

Introduction

Create the new recipes using the "Recipes" tab of the "Recipes" editor.

Naming the recipe

Do not use keywords or reserved words of the SQL database language for recipe names (see section "SQL keywords (Page 1169)").

Recipes with date or time stamp

If you use data or time data, make sure that the system setting for time and date on the configuring computer matches that on the target system. Example: You load a recipe data record on the target system at 13:55 in which the processing time 14 h is stored. If it already 14:05 on the target computer, the recipe will not be processed. If an operator processes the recipe, modification information is not written back correctly into the database.

After loading to the target system, check the recipes with date or time stamps on the target system.

Requirement

- You have created the control tags or the raw data tag.
- The "Recipes" editor is open.

Procedure

Create a recipe as follows:

1. Double-click the first empty table row in the "Recipes" tab of the "Recipes" editor.
2. Select "Properties > General > Settings > Name" in the Inspector window and enter a descriptive recipe name.
   The name identifies the recipe uniquely within the project.

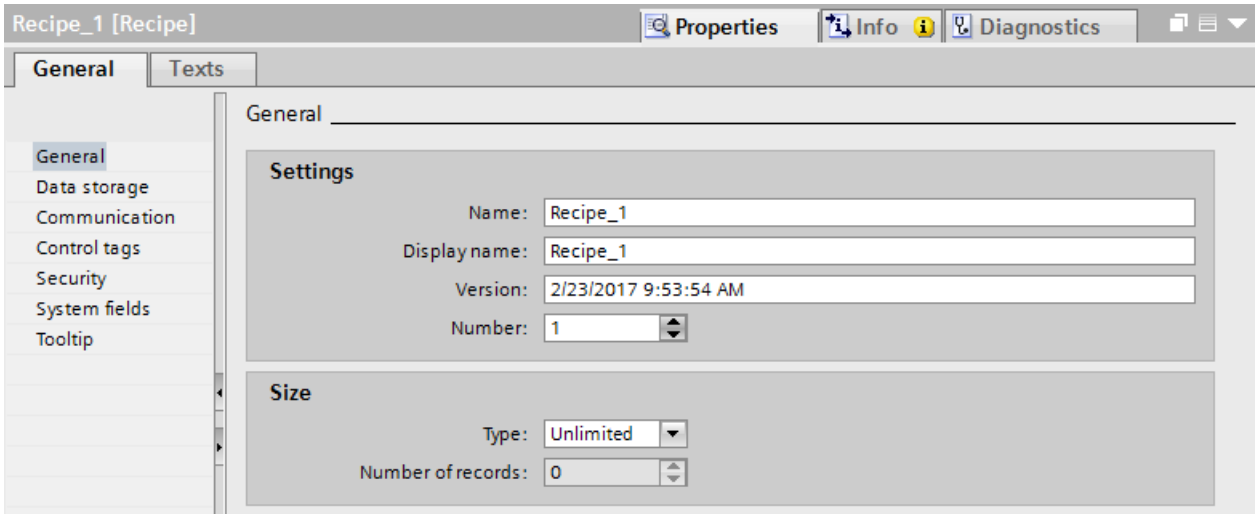

3. Select the "Display name" field to enter the language-specific name to be displayed in Runtime.
4. Define whether or not the recipe data records are limited to a specified maximum number in the area "Size" under "Type".
5. If the size type is limited, specify the maximum number of recipe data records of a recipe under "Number of data records".
6. Specify the "Communication type" under "Properties > Communication":
   - If you have selected "Raw data tag", create a raw data tag and select it under "Raw data tag".
   - Select "No communication" if you use the recipe only to save project data.

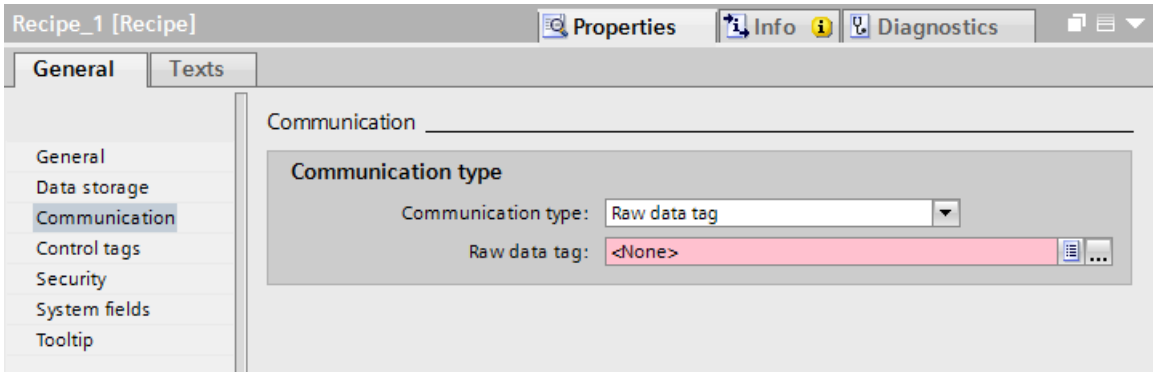

7. Select the control tags for access to the recipe under "Properties > Control tags" if you have selected the "Tags" communication type.

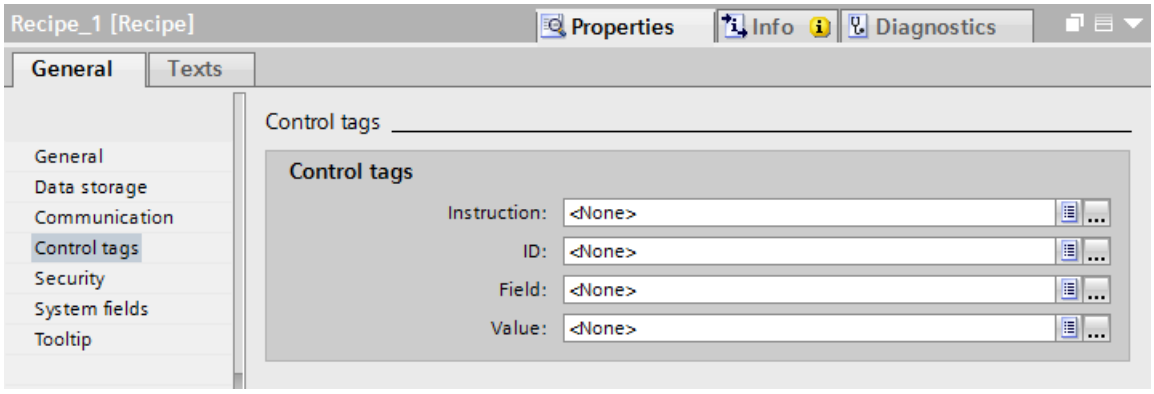

8. Define under "Properties > Security" which operator authorizations have read and write access to the recipe in Runtime.

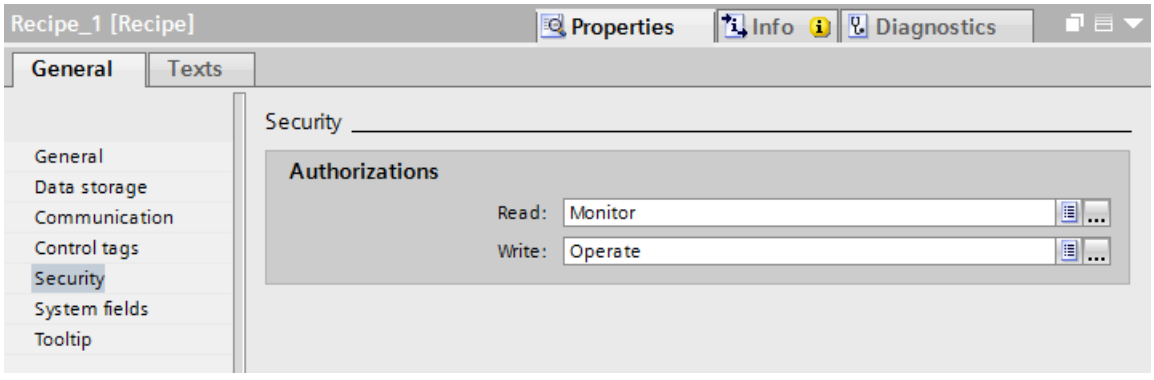

9. Under "System fields", specify whether to display the "Last access" and "Last user" entries in Runtime.

Result

You have created the recipe.

See also

SQL keywords (Page 1169)

6.2.2.3 Creating Recipe Elements and Data Records (RT Professional)

Introduction

Select the "Recipe elements" tab in the bottom area of the "Recipes" editor to set up the elements for the selected recipe. You can also enter the recipe element values in the "Recipe data records" tab.

Requirement

- The tags for the recipe have been created.
- You have created the recipe.
- The recipe is selected in the "Recipes" editor.

Procedure

Create a recipe element as follows:

1. Double-click the first empty table row in the "Recipe elements" tab.
2. Select "Properties > General > Settings > Name" in the Inspector window and enter a descriptive name for the recipe element. This name must be unique within a recipe.
3. Select the "Display name" to enter the language-specific name to be displayed in Runtime.

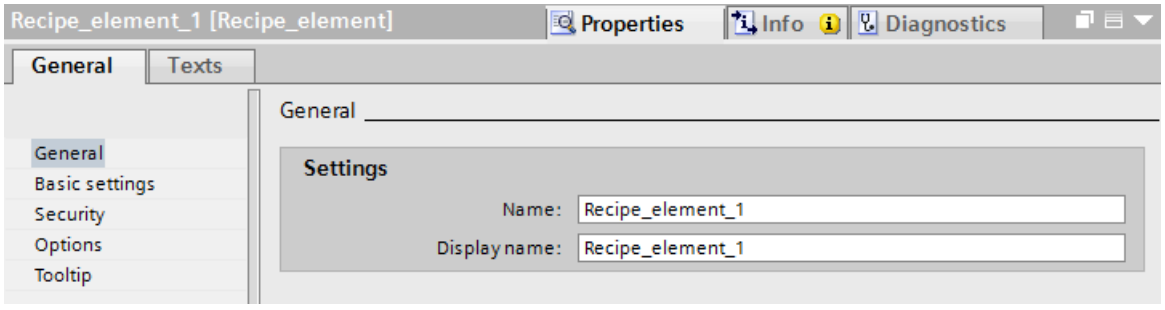

4. When the recipe has the "Tags" communication type:
   To link the recipe element to a tag, select a tag under "Properties > Basic settings > Settings". The data type will be displayed below.
5. For a "String" data type, also specify the maximum length of the string under "Data length". The "Settings" area on the right shows the value range for number-based data types, for which you can also specify a default value.

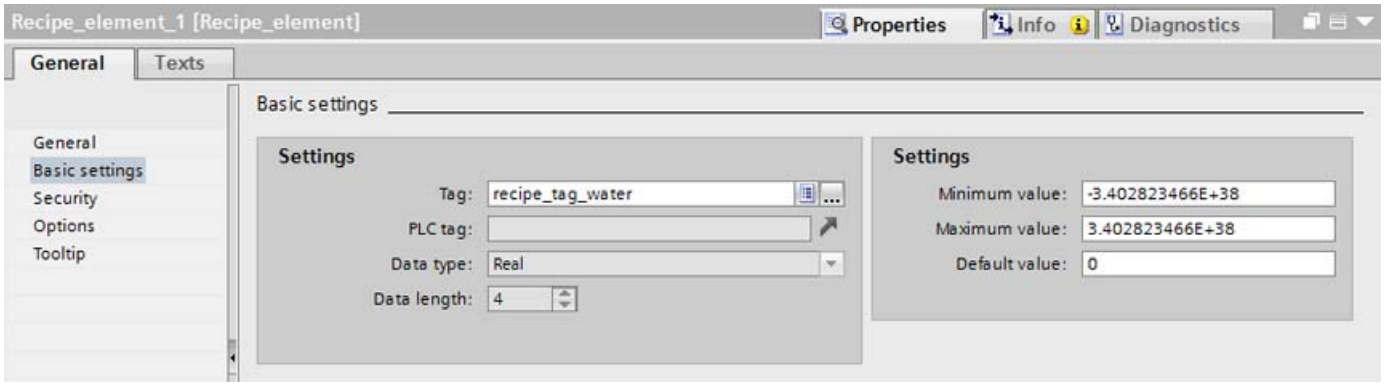

6. When the recipe has the "Raw data tag" communication type or "No communication": Select the "Data type" and also the "Data length" of the recipe element for the "String" data type.
7. Define under "Properties > Security" which operator authorizations have read and write access to the recipe element in Runtime.

8. Define in "Properties > Extras" which conditions the values of the recipe element must satisfy. For faster search, the values can be indexed:

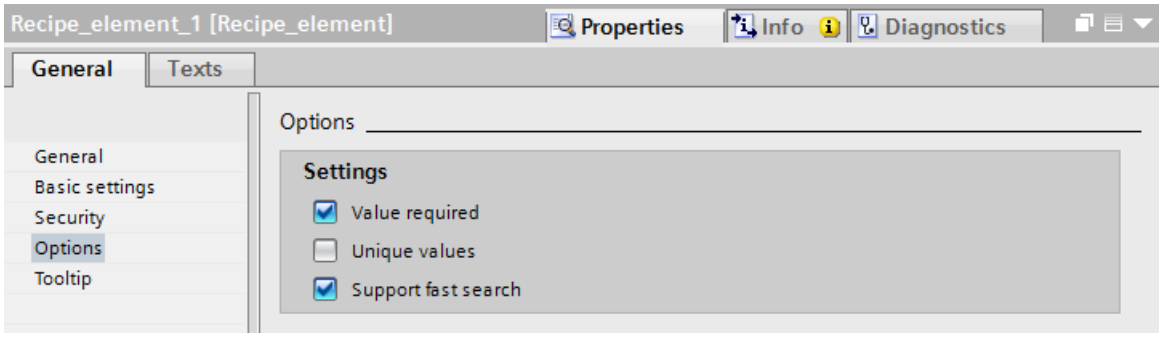

9. Double-click the first empty table row in the "Recipe data records" tab to set up a recipe data record.
10. Enter the desired value for the recipe element in the table editor.

Result

The recipe element and a corresponding recipe data record are created with one value.

6.2.2.4 Importing recipes into the configuration and exporting them again (RT Professional)

Introduction

You can export recipes as a CSV file and import them again into the configuration.

Note

You cannot import or export data records from recipe queries.

Application case

You export recipe data to store and back up recipe data long-term on a computer.

You export the recipe data to an HMI device to standardize and distribute recipe data. You change the CSV file Microsoft Excel and import the CSV file to all HMI devices which require the same recipe data.

You want to exchange recipe data between different projects. You change the recipe data in Runtime. To transfer the changed recipe data to WinCC ES, export the recipe data records to Runtime and copy the CSV file to the configuring computer. There you import the CSV file containing the recipe data records into the recipe. And in the opposite direction, you transfer the modified recipe data records to the HMI device in WinCC ES in Runtime.

Requirement

- You have created the recipe.
- The structures of the exported CSV file and the recipe in WinCC ES match; name, number and data type of the recipe elements are identical.
- The "Recipes" editor is open.

Importing CSV files

To import recipe data records into a recipe, proceed as follows:

1. Select the row which contains the desired recipe in the "Recipes" tab.
2. Select the "Import recipe data" command from the shortcut menu.
   The "Import" dialog box opens.

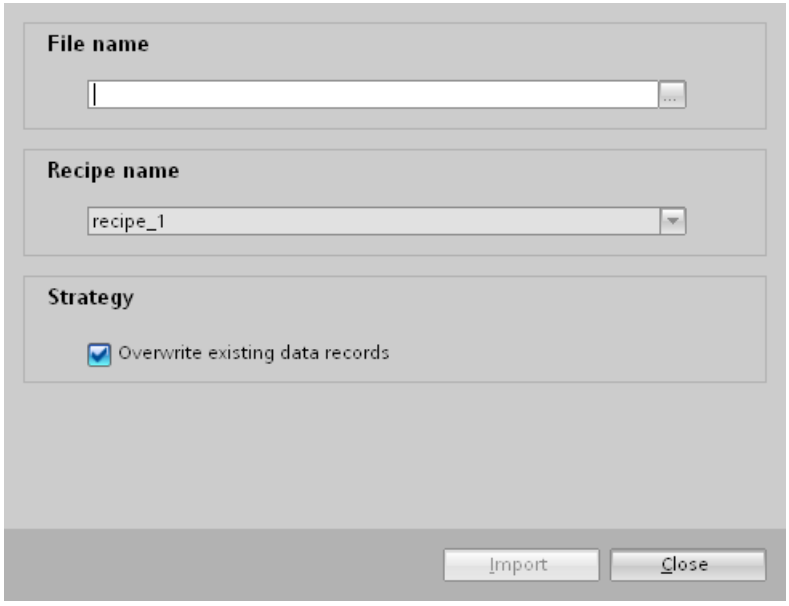

3. Select the desired CSV file with the recipe data under "File name".
4. Under "Strategy", define whether a recipe data record in the CSV file overwrites an existing recipe data record in WinCC with the same recipe data record number.
5. Click "Import" to start the operation.

Result

The recipe will be amended by recipe data records from the CSV file that have not yet been entered. If the option is selected, existing recipe data records will be overwritten.

An error message is shown in the "Info" of the Inspector window if the structure of the recipe does not match the structure of the CSV file.

Exporting CSV files

1. Select the row which contains the recipe to export in the "Recipes" tab.
2. Select the "Export recipe data" command from the shortcut menu.
   The "Export" dialog box opens.

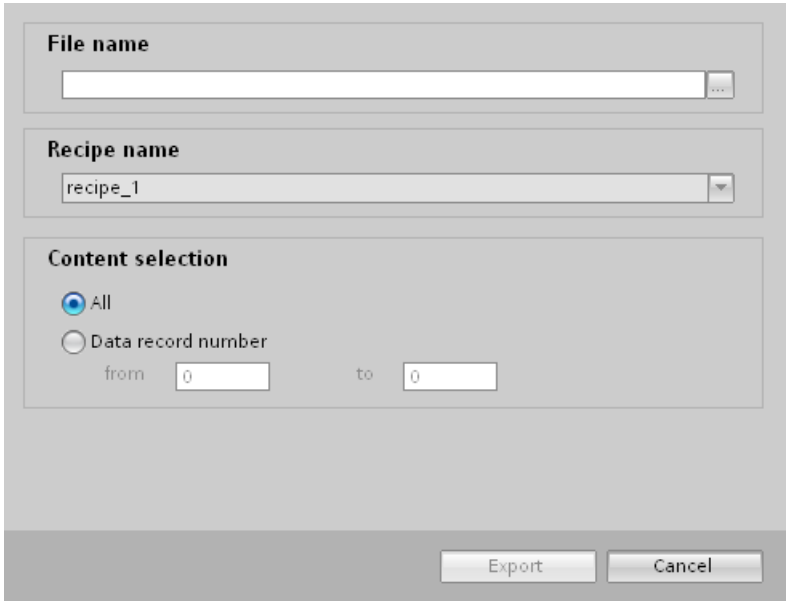

3. Select a path for the CSV file under "File name".
4. Select the desired recipe under "Recipe name".
5. Select all recipe data records or limit the selection to certain recipe data record numbers under "Select content".
6. Click "Export" to start the operation.

Result

The configuration is exported as a CSV file. The list separator is the semicolon "RecipeElement_1;RecipeElement_2". The decimal separator is a period, e.g. "3.14".

6.2.2.5 Procedure for changing the recipe structure (RT Professional)

Introduction

Different recipe structures can occur in the following situations:

- Due to changes during commissioning
- If a machine is upgraded by the machine manufacturer (retrofit)
- When CSV files are imported, the structure of the CSV file can differ from the recipe structure.

Nevertheless, you can still use any recipe data records already created.

NOTICE
When a tag is renamed, the assignment is lost.

Effects

If the structure of the CSV file differs from the structure of the recipe, deviations are handled as follows:

- If the old recipe data record or the CSV file contains additional values, these values will be discarded.
- If the old recipe data record or the CSV file contains values of the wrong data type, the configured default value will be used in the recipe data record.
  Example: The recipe data record contains values that show the tank contents and were input as floating point numbers. However, the corresponding recipe tag expects an integer value. In this case, the system discards the transferred value and the configured default value is used.
- If the old recipe data record or the CSV file contains too few values, the configured default value will also be used in the recipe data record.

6.2.2.6 Creating recipe queries (RT Professional)

Introduction

A recipe query is a set of recipe elements from various recipes. Example: Recipe query "Water consumption":

- Water from the "Fruit juice drink" recipe
- Water from the "Juice" recipe
- Water from the "Nectar" recipe

You can link recipe elements to form relations, such as "Nectar.Water > FruitJuiceDrink.Water". In the recipe query, only the data records that satisfy the relation are displayed in Runtime. In the example, these are the records in which the water requirement of the "Nectar" recipe is greater than that of "Fruit juice drink".

You can assemble recipe queries from recipe elements in the "Recipes" editor in the "Recipe queries" tab. Select the recipe elements from the recipes that are included in the recipe query and set their relation. The relation expresses a condition. Only those combinations of recipe elements which satisfy the condition (true) are displayed in the recipe view.

You can also edit the recipe query data in Runtime. The data you have edited are applied to the original recipe in Runtime.

The recipe query does not support the following actions:

- Creating new recipe data
- Deleting recipe data

Requirement

- You have created at least two recipes.
- The recipes are in the same database on the same HMI device.
- The recipes have at least one identical recipe element in the same data format.

Creating recipe queries

Create a recipe query as follows:

1. In the "Recipes" editor, double-click the first empty table row in the "Recipe queries" tab. A new recipe query is created.
2. In the inspection window, enter a name for the recipe query under "Properties > General > Name " to indicate its function.

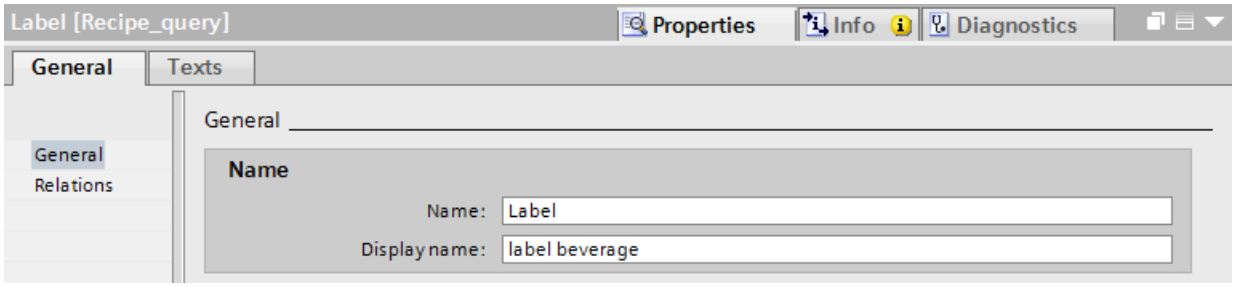

3. Select the "Display name" to enter the language-specific name to be displayed in Runtime.

Creating recipe query elements and relations

To create a new element for the recipe query and the relation, follow these steps:

1. Double-click on the first free table row in the "Recipe query elements" tab.
2. Under "Properties > General > Recipe element", select the recipe element to insert into the query.
3. Create other recipe query elements as required.
4. Select the recipe query in the "Recipe queries" index.

5. Select a recipe query element and the desired operator as a left and right operand in the Inspector window under "Properties > Relation > Relation".

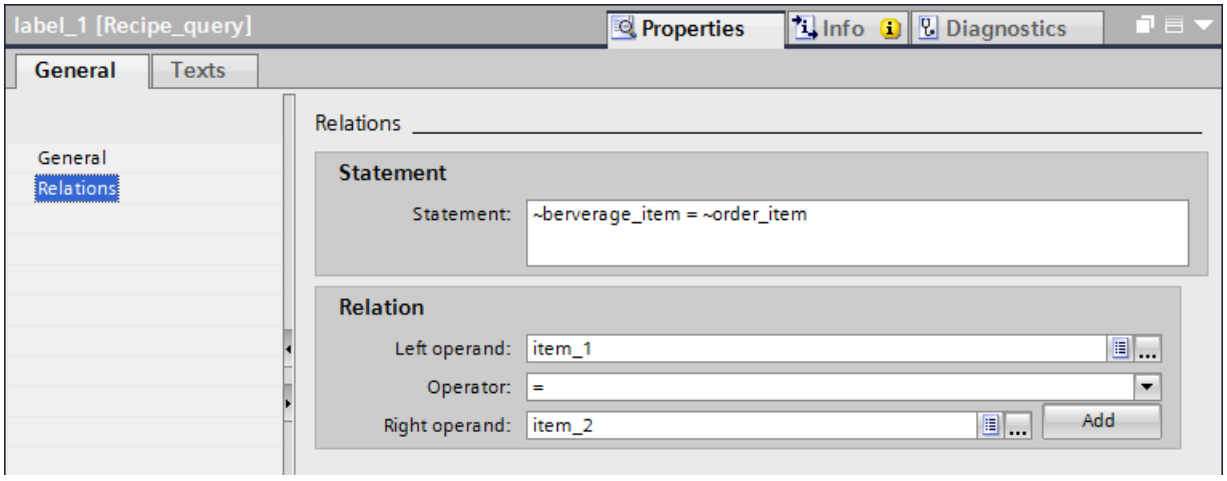

6. Click the "Add" button.
   The relation appears under "Condition". If a condition was already defined there, the new relation with the logical operator "AND" is added. You can edit the condition.
7. Change the order of the recipe query elements as required using drag-and-drop.

Result

The recipe query is created and a condition is defined. Only those combinations of recipe elements which satisfy the condition are displayed.

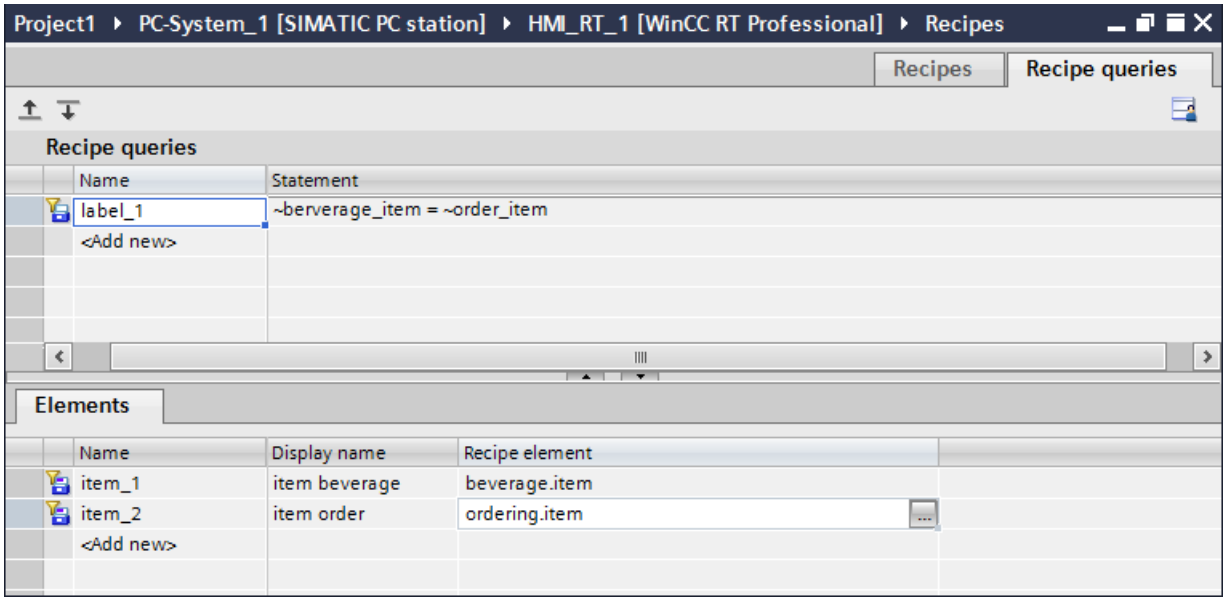

6.2.3 Configuring the visualization of recipes (RT Professional)

6.2.3.1 Displaying recipes (RT Professional)

Introduction

Recipes and recipe queries are displayed on the HMI device as follows:

- In a recipe view
- In individual I/O fields

Recipe view

The recipe view displays the recipes and queries in tabular format. Each recipe data record uses one line.

The picture below contains an example of a recipe view.

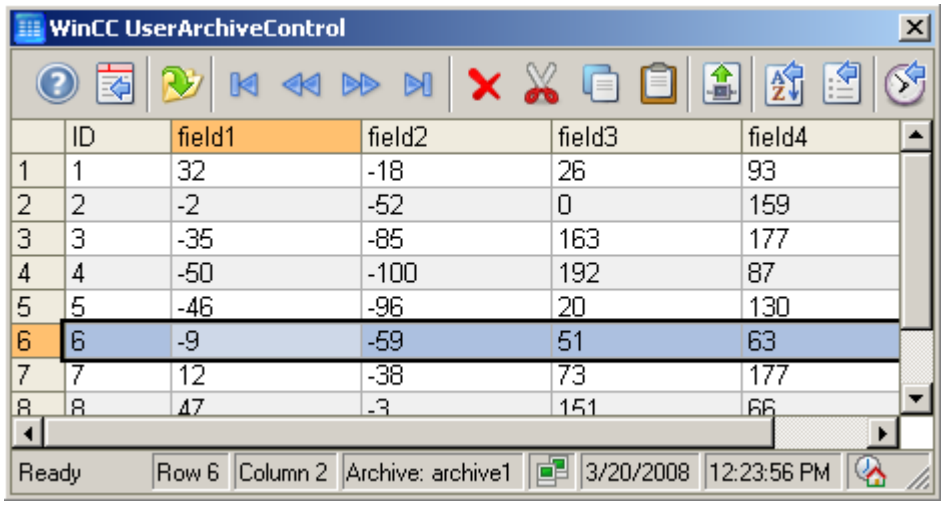

Operators use the recipe view in Runtime and, for example, create new data records or import recipes. You transfer a recipe data record to the recipe tags and vice versa with two buttons in the recipe view.

Note

You select a recipe or recipe query when configuring a recipe view. The operator can change the selection and other properties in Runtime.

Recipes in the sc

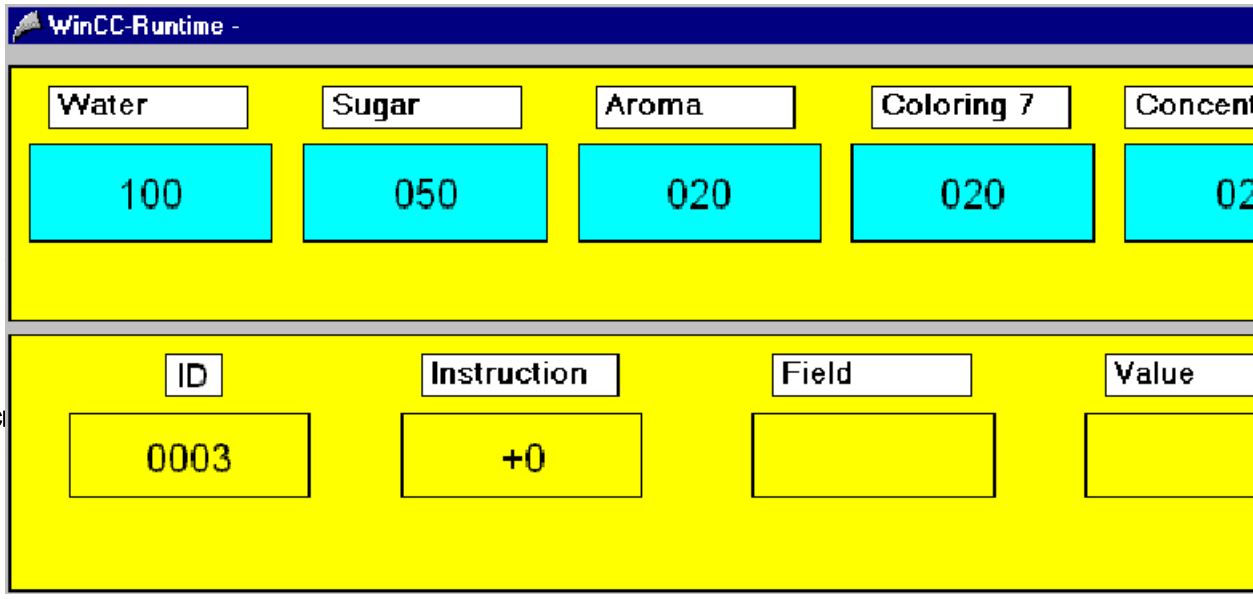

The blue fields at the top show the values of the recipe tags which have a communication connection with the PLC. The yellow fields at the bottom show the contents of the four control tags. In the example the blue recipe values were written into recipe data record number 3 (task = 6) and the number of errors = 0 returned.

See also

Control tags (Page 1190)

6.2.3.2 Properties of the recipe view (RT Professional)

Introduction

The recipe view visualizes recipe elements in table format on the screens. Operators can edit recipe data in the recipe view in Runtime.

Configuration of the recipe view

The following optional recipe view configurations are available:

- General > Display: You select a recipe or recipe query as a data source and define whether recipe data records can be edited, inserted or deleted.
- General > Print job
  The recipe view is automatically linked by logical operation with the system print job for recipe views. Create a logical link to the recipe view for any custom print jobs or system print jobs.
- General > Time
  Define the time base for the "Last access" display, for example the coordinated universal time (UTC). You can include the time base selected for output in Runtime.
- Appearance, layout, text format
  You define colors and design, size and position of the recipe display, font type and font size.

- Window
  You define the font size, window header of the recipe display and other window properties.
- Table
  You define the appearance of the column headers, table and boundary lines and the behavior for selecting and sorting data records.
- Table > Mark
  Determine whether lines or only cells can be marked with the mouse.
  Configure the properties of the selection rectangle which can be shown around the marked cells or lines of the table.
  Configure the marking colors for the markable cells and/or lines as required. The color defined by the system for the marking is used with the "Automatic coloring" property.
- Table > Sorting
  Determine whether sorting is to take place and if so, with click or double-click on the column header. Define the order of sorting which is to run cyclically when clicking or double-clicking the column header: either sorting ascending - descending or ascending - descending - no sorting
  Configure the sorting symbol and sorting index which are to be shown right-justified in the column header. You show sorting sequence and sorting order of the columns.
  If you activate the "Use sorting key" option, the sorting symbol is displayed as a sorting key above the vertical scrollbar. Click this sorting button to sort the selected column based on the configured sorting criteria. The sorting button is not displayed if the table does not contain a vertical scroll bar.
- Columns
  You determine which recipe elements are displayed in what order in Runtime. You apply write protection to certain elements and determine the width and formatting.
  Some columns can also show the content and header as symbols. Determine how these columns are displayed in the "Layout" field. The text and symbol can be displayed together.
- Filter
  As a filter criterion you define a condition for a recipe element and link several conditions. The data record is not displayed until all the filter criteria are satisfied.
- Toolbar
  In addition to the appearance of the toolbar you define which buttons are visible in what order in Runtime and with what operator authorization they can be operated.
  Activate the buttons in the list which you need for operating the WinCC-Control in Runtime. Information about the function of the individual buttons can be found in the subsection "Operation in Runtime"
  Define a hotkey for a button if necessary.
  If you assign operator authorizations to the individual buttons, the button is only released in Runtime for the users who have the appropriate operator authorization.
  If you deactivate the "Active" option for a button, the activated button is displayed in Runtime but cannot be operated.
- Status bar
  In addition to the appearance of the status line, you define what information is displayed in the status line in Runtime, e.g. the recipe data record number.

- Data export
  A standard file name and a standard directory are already entered by switch tags. However, you can assign your own file names and directory names.
  You can use the following switch tags.
  @OBJECTNAME: Object name of the recipe display, see "Properties > Miscellaneous"
  @PROJECTPATH: Project directory
  @CURRENTDATE: Current date
  @CURRENTTIME: Current time
- Security
  You determine whether and how changes to the properties of the recipe display can be saved in Runtime and after a screen change and which authorization is necessary for changing.
- Miscellaneous
  You define the visibility and layer for the recipe display.

6.2.3.3 Configuring a recipe view (RT Professional)

Requirement

- You have created a recipe or a recipe query.
- A screen is created and open in the "Screens" editor.

Procedure

Configure a recipe view as follows:

1. Drag the "Recipe display" object from the "Controls" palette of the "Tools" task card into the screen.
2. Select the desired recipe or recipe query in the Inspector window under "General > Display".

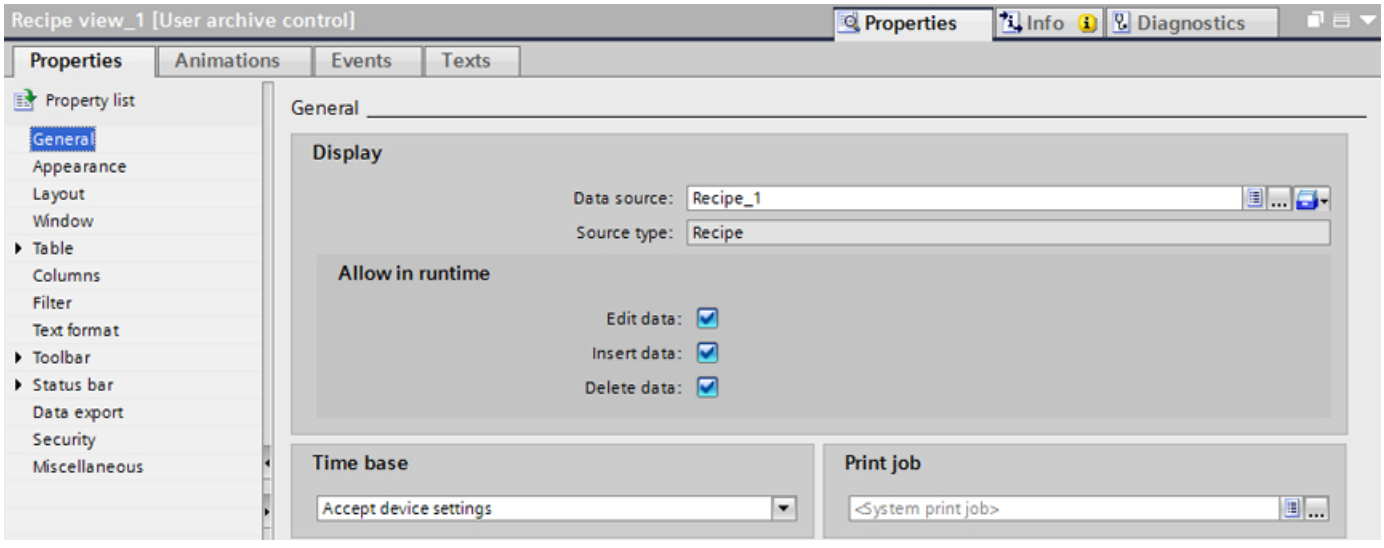

3. Define the write and read rights for the recipe display in Runtime under "General > Allow in Runtime".

4. Select "Appearance," "Layout," and "Text format" to define additional display options for the recipe view.
5. Select "Properties" > Columns" to define the recipe elements which will be visible and write-protected in Runtime.

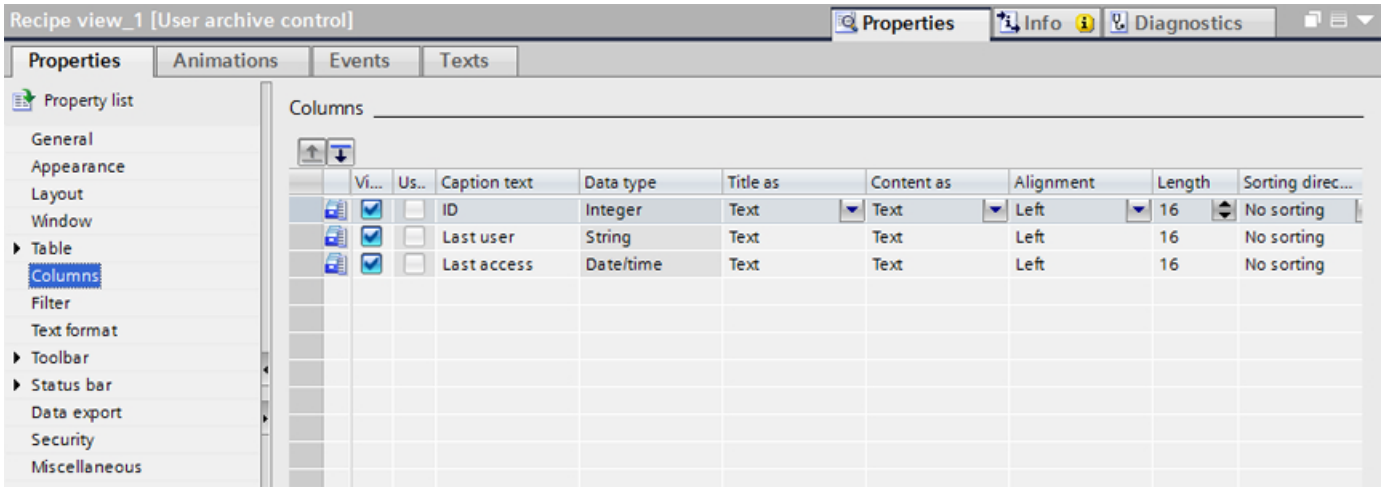

Note

Some columns can also show the content as symbol. In "Content as", determine how these columns are displayed. Text and symbol can also be displayed together.

6. Select "Toolbar" to define the recipe view buttons to be available in Runtime.

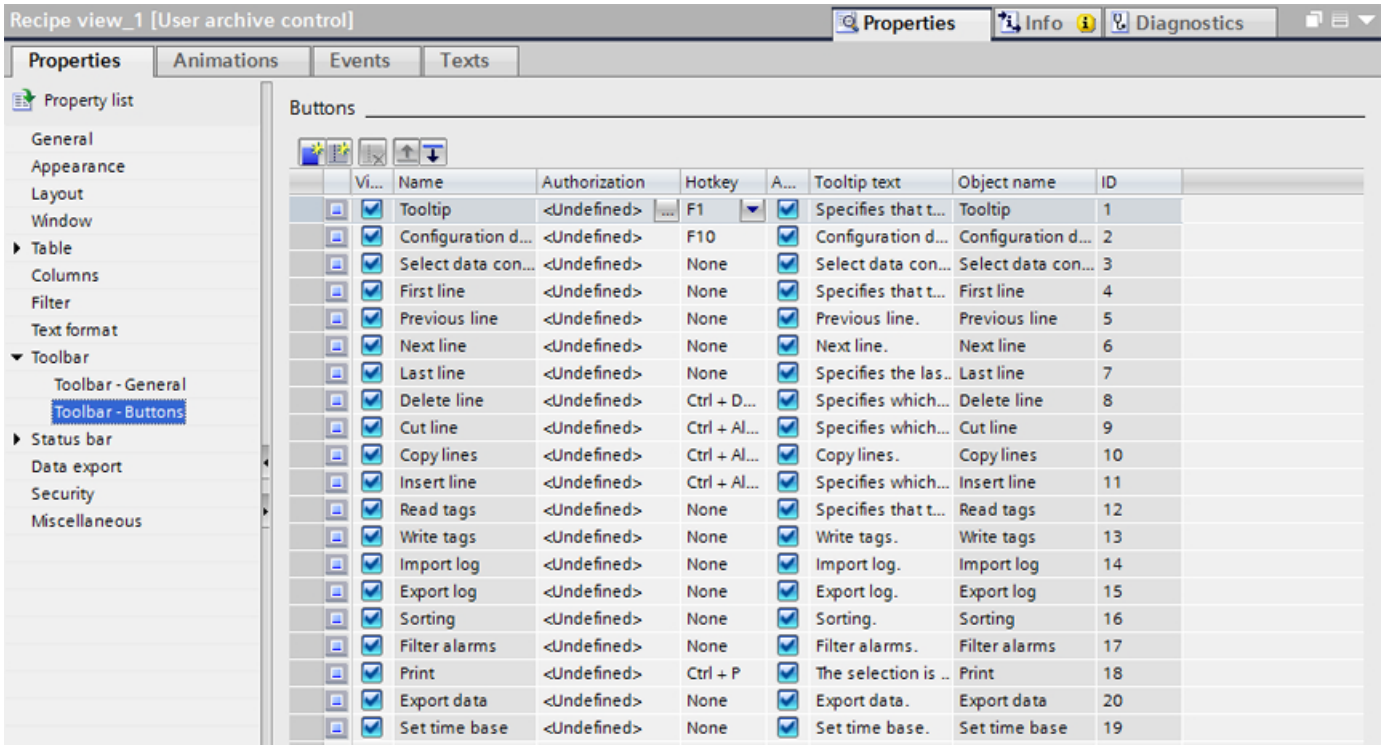

7. Select "Status bar" to define the scope of information to display in the status bar in Runtime.

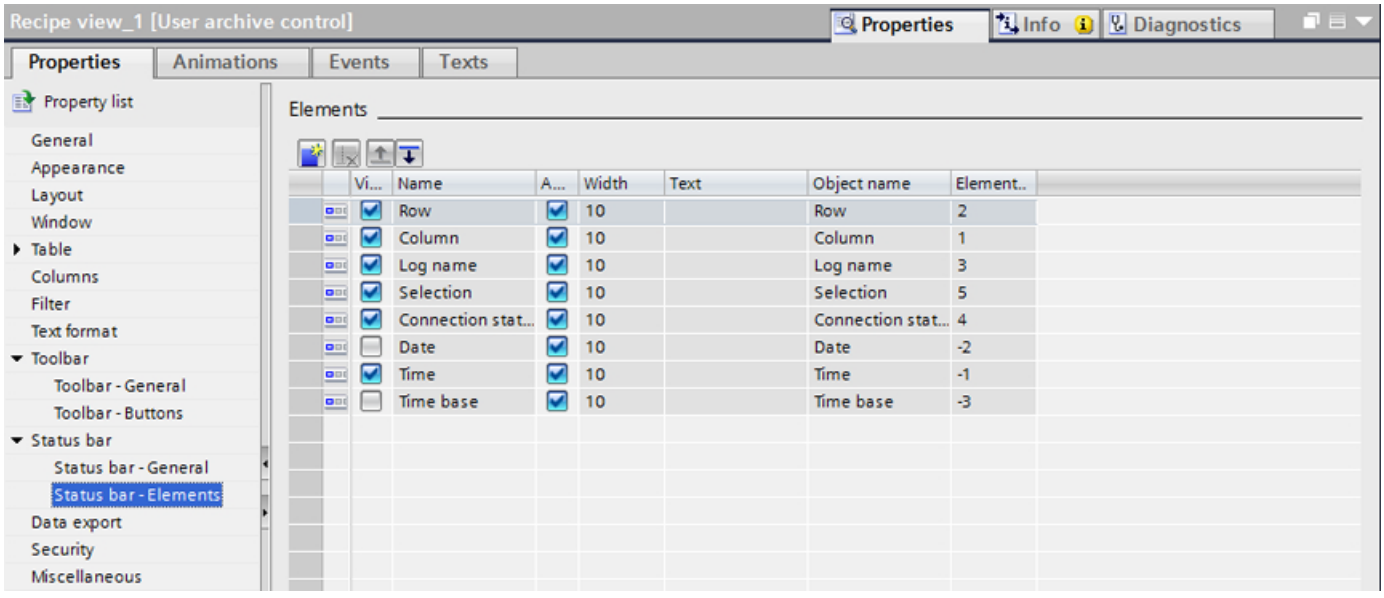

8. Select "Filter" to define the filter criteria for recipe elements in Runtime.
9. Select "Sort" to define the sorting criteria for recipe elements in Runtime.

Result

The recipe view is configured.

6.2.3.4 Exporting recipe data in Runtime (RT Professional)

Introduction

In Runtime you export the recipe data with a button in the recipe display. You configure operation of the data export in Runtime.

How to configure the operation of the data export

1. Define a file name and a directory for the export file under "Properties > Data export".
   The file name may be made up of names you select yourself and the following switch tags:
   @OBJECTNAME: Object name of the recipe display
   @PROJECTPATH: Project directory
   @CURRENTDATE: Current date
   @CURRENTTIME: Current time
1. Define the file extension, e.g. "CSV".
2. Define the scope of the data export:
   – All Runtime data are exported.
   – Selected Runtime data are exported.

3. Configure the operation of the data export during runtime. Define:
   - The "Data export defaults" are displayed in runtime.
   - The user may change the file name or the directory.
4. If "Show dialog" is deactivated, the data are exported immediately to the defined export file on pressing the "Export log" button.
5. Save the project.
6. Activate the "Export log" button under "Properties > Toolbar".

Results

In Runtime you can export all or selected recipe data to a defined file with the button.

6.2.3.5 Configuring persistence (RT Professional)

Introduction

A user can configure WinCC-Controls in Runtime. You define what effect the online configuration has in Runtime.

The changes configured in Runtime are saved separately from the screen in the engineering system. The screen is thus retained in its original configuration.

Procedure

1. Select "Properties > Safety > Persistence > Online configuration" which you can only operate in the engineering system. The online configuration can then be defined permanently in Runtime:
   - "No persistence" The online configurations are not retained in Runtime. This default leaves the user no option in Runtime. The online configurations are lost the next time the screen changes and with deactivate/activate.
   - "Persistence" This default leaves the user the options of "reject", "retain" or "reset". With the "retain" option the online configurations are retained the next time the screen changes and with deactivate/activate.
   - "Persistence in Runtime". This default leaves the user the options of "reject", "retain" or "reset". With the "retain" option the online configurations are retained the next time the screen changes but are lost when the project is deactivated/activated.

   Note

   In case of a user change, the different configurations are not saved until you have executed a screen change.

2. Define what operator authorization the user must have to carry out online configuration.

3. "Screen change characteristics" can be operated in the engineering system and in Runtime with the defaults "Persistence" and "Persistence during Runtime". The "reset" option can only be operated in Runtime because the original configuration is in the engineering system. Select one of the three effects of the online configuration in the next screen change:
   - Activate "Reject changes" so that the online configuration only exists for as long as the screen is selected. The changes are rejected at the next screen change. The recipe display appears as configured if you select the screen again.
   - Activate "Retain changes" so that the online configuration is not lost at the next screen change.
   - Activate "Reset" so that the configured screen is saved again in Runtime. All online configured changes are lost.

   Note

   The screen is also replaced in Runtime when saving the screen or in Delta compiling (change loading). All online configured changes are lost.

4. Save the configuration.

6.2.4 Data transfer to the PLC (RT Professional)

6.2.4.1 Data transfer basics (RT Professional)

Overview (RT Professional)

The options for transferring recipe values between the PLC and HMI device are as follows:

- No data transfer
- "Tags" communication type
- "Raw data tag" communication type

No data transfer

The recipe is not used for transferring recipe data to the PLC but as a data memory and user log: You store values which belong together in data records to which you have multiple access. In the simplest case an operator switches between the recipes in a recipe view. The respective recipe data are loaded into the recipe tags and processed further there.

"Tags" communication type

Recipe tags transfer the recipe values to the PLC. Vice versa the values from the PLC are written into a recipe data record by recipe tags. Control tags control the data transfer. The following tasks are started automatically by the desired entry into the corresponding control tags.

- Write data records into the recipe tags and therefore into the PLC
- Read data records out of the recipe tags and therefore out of the PLC
- Delete data records
- Create new data records

Note

A complete data record is always transferred in the "Tags" communication type. You cannot transfer recipe data from a recipe query.

"Raw data tag" communication type

The automation system sends the raw data tag as a message frame to the HMI device as an active partner. The message frame contains the following:

- Desired target recipe
- Desired target recipe data records
- Job e.g. "Read", "Write" or "Delete"
- New recipe values, e.g. for "Write" job

The HMI device returns recipe and acknowledgement data as a message frame.

The "BSEND/BRCV" function of the S7 communication is used for the data exchange. The consecutive number of the recipe serves to identify the recipe uniquely in the PLC (PLCID).

The PLC requires a user program which executes the following in Runtime:

- Collect data if necessary via I/O fields in the HMI device.
- Check data.
- Create raw data tag as a message frame.
- Send message frame.

If you want to start creation of the message frame from the HMI device, you configure functions in the screen which start the user program.

Note

The following is possible in the "Raw data tag" communication type:

- Transfer of an individual recipe element
- Transfer of an entire recipe data record
- Transfer of multiple recipe data records

Data transfer from the recipe view or by control tags is not possible.

Control tags (RT Professional)

Introduction

With the "Tags" communication type in the recipe, you transfer the recipe data between the PLC and the HMI device using recipe tags. The data transfer is controlled by four control tags which are assigned fixed to a recipe. Other control tags can be assigned to another recipe.

Data transfer is started automatically once you have entered the required value at the corresponding control tag.

Note

All four control tags must always be specified in a recipe to ensure that it works. Do not change the data type of the control tags.

Note

You must implement access protection for the control tags of a protected recipe separately using the object properties of the screen, I/O field or button, for example.

Control tags

Control tag	Required data type	Meaning	
ID	Long	Number of the recipe data record within the assigned recipe	
Job	Long	Read out of the recipe tags, write into the recipe memory:	6
		Read out of the recipe memory, write into the recipe tags:	7
		Delete in the recipe memory:	8
		After the job has been carried out, an error ID can be seen in the "Job" control tag:	
		No error	0
		Error	-1
Field	String	The name of the recipe element Make sure that you do not use the display name.	
Value	String	The value of the recipe element	

Requirement

- Communication type is set to "Tags" in the recipe.
- You have configured a process screen with all I/O fields required for process tags and control tags of the recipe.
  Example:

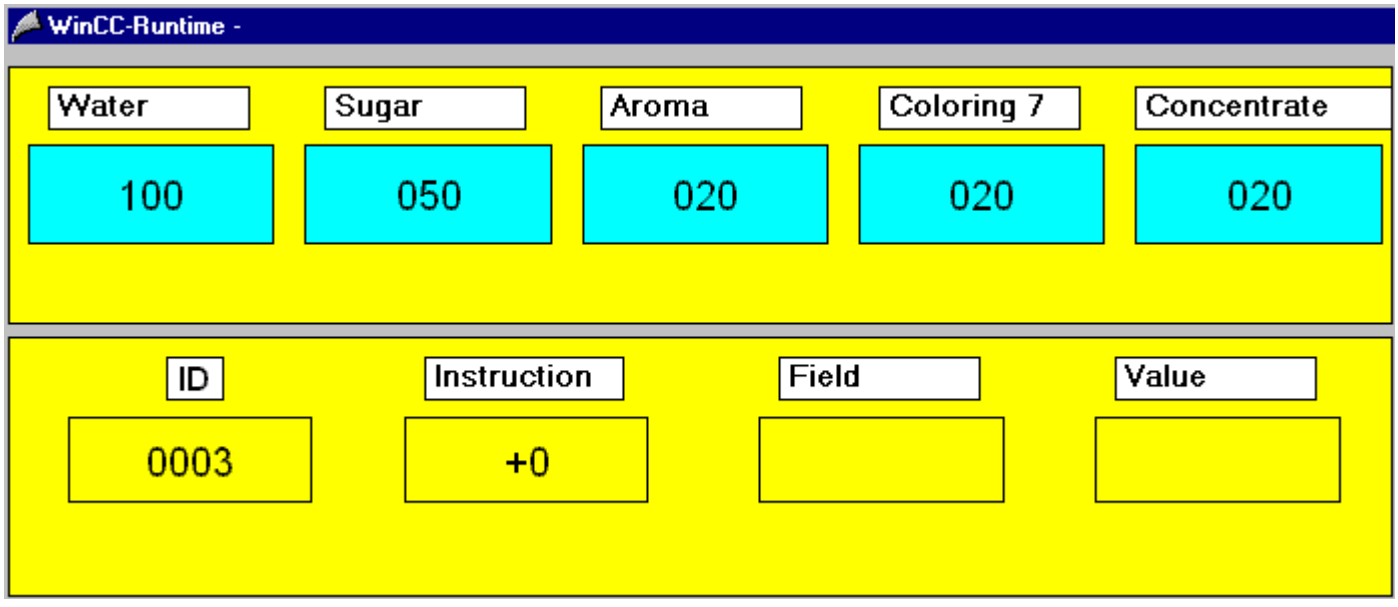

Inserting the control tag

The following control tag combinations are supported:

- "ID" and "Job" control tags
  The control tag "ID" indicates the number of the recipe data record which is transferred.
  The "Job" control tag indicates whether the selected recipe data record is written, read or deleted (6, 7, 8) in the recipe memory of the HMI device.
- "Job", "Field" and "Value" control tags
  The recipe data record is selected for the data transfer whose recipe element "Field" has the recipe value "Value". Example field = "Water", value = "5 liters".
  The "Job" control tag indicates whether the selected recipe data record is written, read or deleted (6, 7, 8) in the recipe memory of the HMI device.

Note

Different data records may not contain the same recipe value in the "Field" recipe element

You have to activate the "Unique values" checkbox under "Properties > Options" for the recipe element. Otherwise no clear assignment of the recipe data record by the value is possible and the first data record which satisfies the "Value in field" condition is used.

If the type "String" is assigned to the element, you must enclose the entry in the "Value" I/O field in inverted commas (such as 'Cola Light').

More combinations of values for the "ID" and "Job" control tags

ID	Job = 6	Job = 7	Job = 8
-1	Data record from the recipe tags is read and appended **to the end** in the recipe in the recipe memory.	-	Data record with the **lowest** ID is deleted in the recipe in the recipe memory.
-6	Data record from the recipe tags is read and written into the recipe data record with the **lowest** ID in the recipe.	Data record with the **lowest** ID is written in the recipe into the recipe tags.	Data record with the **lowest** ID is deleted in the recipe in the recipe memory.
-9	Data record from the recipe tags is read and written into the recipe data record with the **highest** ID in the recipe.	Data record with the **highest** ID is written in the recipe into the recipe tags.	Data record with the **highest** ID is deleted in the recipe in the recipe memory.

See also

Displaying recipes (Page 1181)

Using the recipe screen (Page 1203)

Differences between data formats (RT Professional)

Introduction

The data formats in WinCC conform to the Intel and Microsoft data formats. The least significant byte is saved first and the most significant byte is saved last. This data format is very common and is generally known as the "Intel format".

The data formats in WinCC differ fundamentally from the data formats in the PLCs.

Intel format

In the Intel format, the decimal number 300 is stored as follows:

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Binary	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

Hex	0	1	2	C

According to the Intel Format, the decimal number 300 corresponds to the hex-number 12C (1*256 + 2*16 + 12*1).

SIMATIC format

In SIMATIC format, the least significant byte is are stored in the most significant digits. In SIMATIC format, decimal number 300 is stored as follows:

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Binary	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1

Hex	2	C	0	1

According to the SIMATIC format, decimal number 300 corresponds to hex number 2C01.

If hex number 2C01 is interpreted according to Intel format, it corresponds to decimal number 11265.

There are function blocks that convert the data for PLCs from the SIMATIC S7 series. You must always call up the blocks before and after data transfers between the PLC and WinCC. The function blocks are available for download from the Siemens Customer Support website in the Internet. The compressed file ANSI_S5.EXE is then loaded. ANSI_S5.EXE contains the "IEEE:GP" function block.

Active sending is described in the reference manuals for the PLCs or CPs (communication processors).

6.2.4.2 Raw data tag (RT Professional)

Basics on data transfer via raw data tags (RT Professional)

Data Transfer via Raw Data Tags (RT Professional)

This section describes the transfer of recipe data between the HMI device, and PLC using raw data tags. The BSEND/BRCV function is used in the PLC. The message frame containing the raw data tag is actively sent by the PLC.

The message frames may contain one or more write or read jobs.

Data transfer procedure

1. The PLC sends the message frame to the HMI device.
2. The HMI device returns the following:
   - Data, if requested
   - A processing acknowledgement

The HMI device returns a separate acknowledgement for every job.

Note

The PLC is the active partner in this data transfer, so it is the PLC that must start the required function. On the HMI device, you can start the required function by means of a change in value of an external tag, for example, and the corresponding analysis in the PLC.

Do not use the "Job type" parameter to start the data transfer. The "Job type" parameter used in the job, or acknowledgement header only works with respect to recipes.

Job types and error codes (RT Professional)

Description of the job types

Type	Description
4	The existence of the recipe is checked.
5	All data records are deleted from the recipe.
6	The data record is requested from the HMI device and is written to the PLC.
7	The data record from the PLC is written to the required data record.
8	The data record is deleted from the recipe.
9	The element of the required data record is requested from the HMI device and is written to the PLC.
10	The element of the required data record is written to the required data record.

Description of the error codes

Group	No.	Description	Possible Fault Causes
General	0	The function has been executed.	--
Recipe	2	Data not available	No recipe configured with this PLCID.
Data record	101	Data not allowed	Recipe structure does not tally, such as number or type of fields.
			Failed data record insertion or update. Possible causes: • Recipe is of the "limited" type. • A min. or a max. value is configured for a field.
			Incorrect criterion
Data record	102	Data not available	For job type 6 only: • No data available • Criterion is incorrect.
Field	201	Data not allowed	For job type 10 only: Criterion is incorrect. Possible causes: • Field not found. • A min. or a max. value is configured for a field.
Field	202	Data not available	For job type 9 only: • Criterion is incorrect. • No field found corresponding to criterion.
General	254	Function not found	--
General	255	Undefined error	--

Message Frame Structure for Jobs (RT Professional)

General message frame structure (RT Professional)

Structure of a message frame for sending jobs:

	Length
Message frame header	16 bytes
Job header 1	12 bytes
Data for job 1	n bytes
Job header 2	12 bytes
Data for job 2	n bytes
Job header n	12 bytes
Data for job n	n bytes

Example

The total length of a message frame with a job, and 10 bytes of job data is 38 bytes.

16 bytes +12 bytes +10 bytes = 38 bytes

Structure of the message frame header (RT Professional)

16 bytes are needed for a message frame header.

Function of the field	Comment	Byte
Message frame length in bytes LSB [1])	Max. length 4091 bytes	1
Message frame length in bytes		2
Message frame length in bytes		3
Message frame length in bytes MSB [2])		4
Transfer type	"1" from the HMI device "2" from the PLC	5
Reserved		6
Number of jobs in the message frame LSB [1])		7
Number of jobs in the message frame MSB [2])		8
Number of the recipe (PLCID)	1. character, in ASCII	9
Number of the recipe (PLCID)	2. character, in ASCII	10
Number of the recipe (PLCID)	3. character, in ASCII	11
Number of the recipe (PLCID)	4. character, in ASCII	12

Function of the field	Comment	Byte
Number of the recipe (PLCID)	5. character, in ASCII	13
Number of the recipe (PLCID)	6. character, in ASCII	14
Number of the recipe (PLCID)	7. character, in ASCII	15
Number of the recipe (PLCID)	8. character, in ASCII	16

① Least significant byte
② Most significant byte

Note

The number of the recipe is defined automatically in the "Recipes" editor during configuration, but can later be changed as needed. Enter the number in ASCII format in the message frame.

Structure of a job header (RT Professional)

12 bytes are needed for a job header.

Function of the field	Comment	Byte
Job length in bytes LSB [1])		1
Job length in bytes MSB [2])		2
Job type	see "Description of job types and error codes"	3
Reserved		4
Field number LSB[1])		5
Field number MSB[2])		6
Data record number LSB[1])		7
Data record number		8
Data record number		9
Data record number MSB[2])		10
Selection criterion LSB[1])	Element value entered or element number as selection criterion.	11
Selection criterion MSB[2])	not if 0	12

① Least significant byte
② Most significant byte

The field number corresponds to the consecutive number of the recipe element.

Message Frame Format for Acknowledgements (RT Professional)

General Structure of an Acknowledgement Message Frame (RT Professional)

Structure of the acknowledgment message frame to the PLC:

	Length
Acknowledgement header	24 bytes
Acknowledgement data	

Structure of an acknowledgment header (RT Professional)

24 bytes are needed for an acknowledgement header

Function of the field	Comment	Byte
Message frame length in bytes LSB [1])	Max. length 4091 bytes	1
Message frame length in bytes		2
Message frame length in bytes		3
Message frame length in bytes MSB [2])		4
Transfer type	"1" from HMI device, "2" from the PLC	5
Reserved		6
Job type	See Description of "Job Types and Error Codes"	7
Error code	See Description of "Job Types and Error Codes"	8
Reserved		9
Reserved		10
Field number LSB[1])		11
Field number MSB[2])		12
Data record number LSB[1])		13
Record number		14
Record number		15
Data record number MSB[2])		16
Number of the recipe (PLCID)	1st character, in ASCII	17
Number of the recipe (PLCID)	2nd character, in ASCII	18
Number of the recipe (PLCID)	3rd character, in ASCII	19
Number of the recipe (PLCID)	4th character, in ASCII	20
Number of the recipe (PLCID)	5th character, in ASCII	21
Number of the recipe (PLCID)	6th character, in ASCII	22
Number of the recipe (PLCID)	7th character, in ASCII	23
Number of the recipe (PLCID)	8th character, in ASCII	24

① Least significant byte

② Most significant byte

Acknowledgement data (RT Professional)

The acknowledgement can contain the following:

- The required recipe data record
- The required content of the element
- No data

Configuring raw data tag (RT Professional)

Raw data tags of the "SIMATIC S7" channel (RT Professional)

Introduction

A tag of the type raw data type is a data telegram on a transport level. The contents of the raw data tag are not fixed and therefore only the sender and the receiver can interpret the transmitted data. There are no format changes in WinCC for this data type. Maximum length is 65535 Byte.

WinCC distinguishes between two types of raw data tags: Raw data tag for free application use and raw data tag for handling S7 functions.

Raw data tag for free application use

Raw data tags for free application use are used for transferring user data blocks between WinCC and PLC and handle only user data. It distinguishes between:

- Raw data tag as byte array
- Raw data tag for BSEND/BRCV functions

Raw data tag for handling S7 functions

These raw data tags do not have any channel-specific header and are normally used by the message system and for process data entry in WinCC.

No further description is needed here as these are tags and functions internal to the channel.

Creating raw data tag (RT Professional)

Procedure

1. Create a new connection, for example, "SIMATICS7".
2. In the "Parameters" tab, select "WinCC RT Prof connection > Send/receive raw data block".

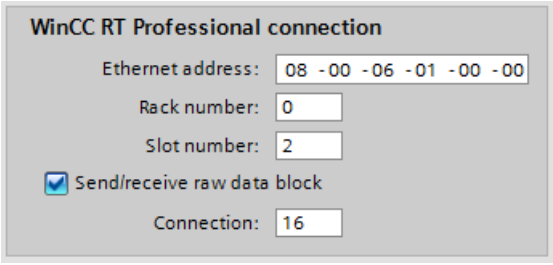

3. Under "Connection", specify the hexadecimal value of the connection ID for the connection resource that was assigned during the configuration of the controller.
4. Create a new tag "Raw data tag_1".

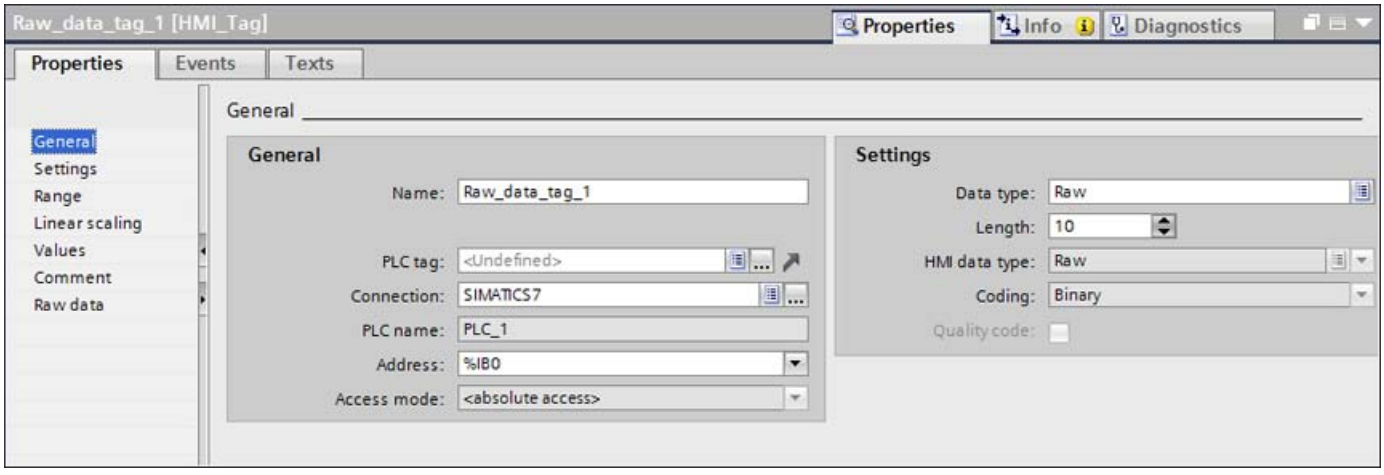

5. Select "SIMATICS7" as "Connection".
6. Select the "Raw" "Data type" and specify a sufficient "Length" in bytes.

7. Select "BSEND / BRCV" under "Raw data."

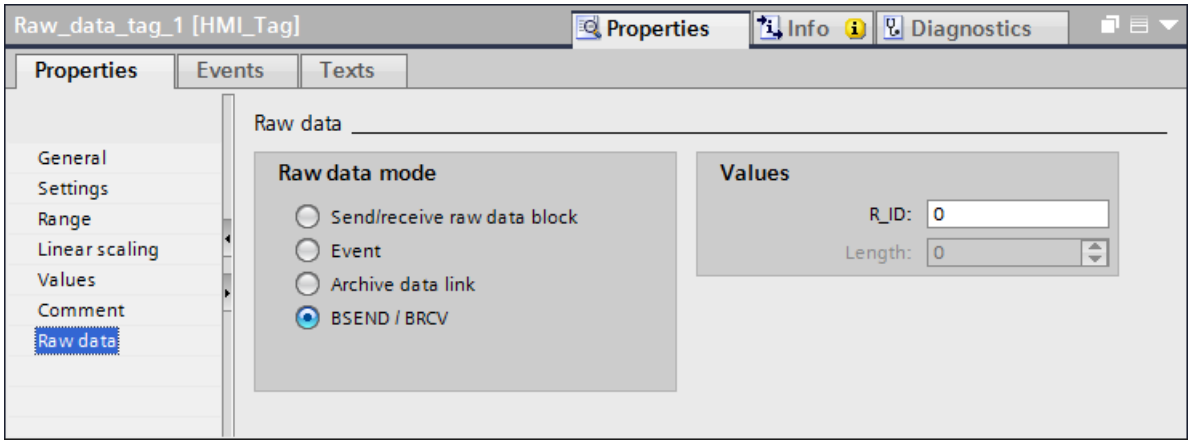

8. Specify the "R_ID" parameter, which was assigned in the configuration of the controller.
9. Create a new recipe.
10. In the Inspector window under "Properties > Communication" select "Raw data tag" as communication type and "Raw data tag_1" as raw data tag.
11. Create the recipe elements and recipe data records. Ensure that the recipe number matches the "PLC" field in the message frame.

Raw data tag for BSEND/BRCV functions of S7 communication (RT Professional)

Introduction

Raw data tags for "BSEND/BRCV" functions are used for transferring user data blocks between WinCC and AS and handle only user data.

This raw data type can be used to access the "BSEND/BRCV" functions of S7 communication. The functions are available in a S7-400 or a S7-300 using CP343. The initiative of data transfer always lies with the sending partner; hence "BSEND/BRCV" functions can also be used to implement event-controlled or sporadic data block transfers.

For resource reasons, it is advisable to keep the number of BSEND/BRCV raw data tags low.

Resource limitations when using S7 functions "AR_SEND" and "BSEND/BRCV" for communication with S7-400

The maximum data volume that can be sent simultaneously using AR_SEND and/or BSEND/BRCV functions from AS to WinCC is limited to 16 kByte.

Example:

- 1x BSEND with a max. of 16 Kbytes
- 1x AR_SEND with 8 kBytes + 1x BSEND with 8 kBytes
- 1x AR_SEND with 10 Kbytes + 1x AR_SEND with 2 Kbytes + 1x BSEND with 4 Kbytes

Note

If the data block of a write job is transferred to AS and has not yet been deleted or fully deleted from the receiving buffer, then the next write job will be rejected with an error message. During such an error display, write jobs with R_ID > 0x8000 0000 are written to a connection-specific queue and the system tries to repeat the write job for 6 seconds.

The responsibility for time co-ordination for transfer rests with the user and needs to be noted as shorter time intervals for write jobs.

Configuring a PBK Connection for Using "BSEND/BRCV" functions

"BSEND/BRCV" functions can only be used via a "hard-configured connection", a so-called PBK connection (programmed component communication). To configure a hard-configured connection, you must mention a connection resource (hex: 10 ... DF). This connection resource will be assigned by STEP 7 when the connection is configured within the PLC. The connection must be configured in AS as passive connection end-point.

A hard-configured connection can also be used to handle "normal" read and write jobs. If very large data areas are to be transferred via the connection, then the data blocks are transferred in multiple PDUs. For performance reasons, it would therefore be better to create a separate connection for "BSEND/BRCV" functions.

Raw data tag for "BSEND/BRCV" functions with the "R_ID" parameter

Raw data tags for transferring "BSEND/BRCV" data blocks are configured as raw data of type "BSEND/BRCV" with a "R_ID". The data length is derived implicitly from the sent or received data volume.

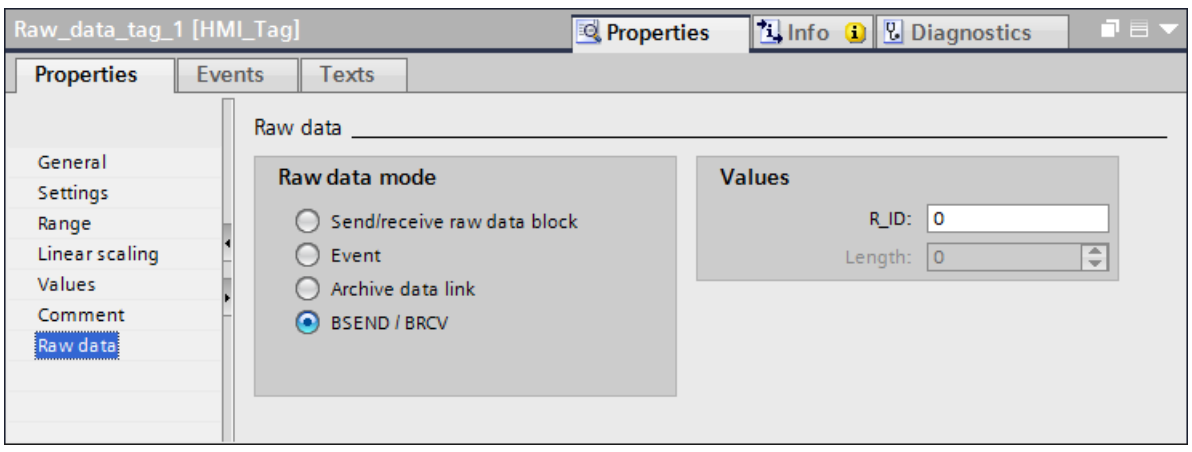

A decimal value of 1 to 999999 must be specified for "R_ID". The R_ID is assigned at the time of configuration in AS and is used for distinguishing multiple data block transfers over one connection. The send and receive calls are always notified with reference to this R_ID in the

underlying communication sub-system (SIMATIC Device Drivers). A raw data tag is thus assigned to one unique R_ID.

Sending a "BSEND/BRCV" raw data tag

Sending a "BSEND/BRCV" raw data tag takes place in the same way as writing a "normal" process tag. After sending the data block and receiving a positive acknowledgement from AS, the data block is transferred to the image of the Data Manager.

Receiving a "BSEND/BRCV" raw data tag

"BSEND/BRCV" raw data is sproradically sent to the channel on the initiative of the AS. Hence it is not possible to explicitly read S7 raw data tags.

The BSEND/BRCV mechanisms do not include any synchronization functions. If no user has logged in to receive the data during the start-up phase, the data blocks sent by AS will bounce on the receiver side. Hence, the user has to take care of the synchronization and, for example, release the sending direction on the AS by setting a flag with a data word.

6.2.5 Operation in Runtime (RT Professional)

6.2.5.1 Recipe view (RT Professional)

Functional scope

The recipe display offers access to user archives, the recipes and recipe queries.

In runtime, you can:

- Online configuration of recipe view
  The changes configured in runtime are saved separately from the screen in the configuration system on a user-specific basis. The screen is thus retained in its original configuration in the configuration system.
- Create, delete or modify new data records
- Scrolling in recipes
- Read and write tags for direct tag link
- Importing and exporting recipes

- Defining selection criteria to display only a certain part of the recipes
- Defining sorting conditions for the displayed recipe elements

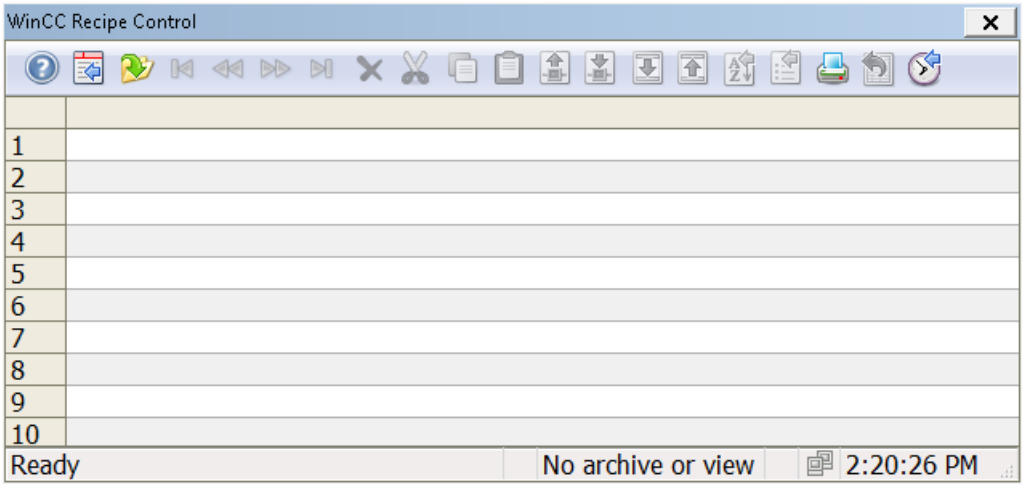

Properties

You connect a recipe view to a recipe or recipe query during the configuration. The online configuration and changing of selected recipe values in Runtime can be protected against unauthorized access respectively by own operator authorizations. If you cancel the access protection, you must reconnect the recipe view with the recipe in the configuration dialog box.

The access protection is queried when the recipe view screen opens.

- If the user has no authorization to read a recipe element, the corresponding column is not displayed in the table.
- If the user has no authorization to write a recipe element, the user cannot edit the data of the the corresponding column in the table.

Access protection for the control tags must be configured separately by the object properties, e.g. of a screen, an I/O field or a button.

6.2.5.2 Using the recipe screen (RT Professional)

Introduction

You use the recipe screen with the control objects you have configured. The section describes the operation of the recipe screen with control tags from the section "Control tags (Page 1190)".

Note

Runtime API

Runtime API provides you with extensive uaArchive functions to edit recipes and recipe queries in Runtime (see section "Interfaces > Runtime API", subsection "Control tags").

Requirement

- Communication type is set to "Tags" in the recipe.
- The "Unique values" checkbox is activated under "Options" in the properties of the recipe element "Coloring 7".
- Recipe tags and control tags are set up.
- A process picture with the required I/O fields is configured.
- The I/O fields are connected with the recipe tags and control tags.

Example:

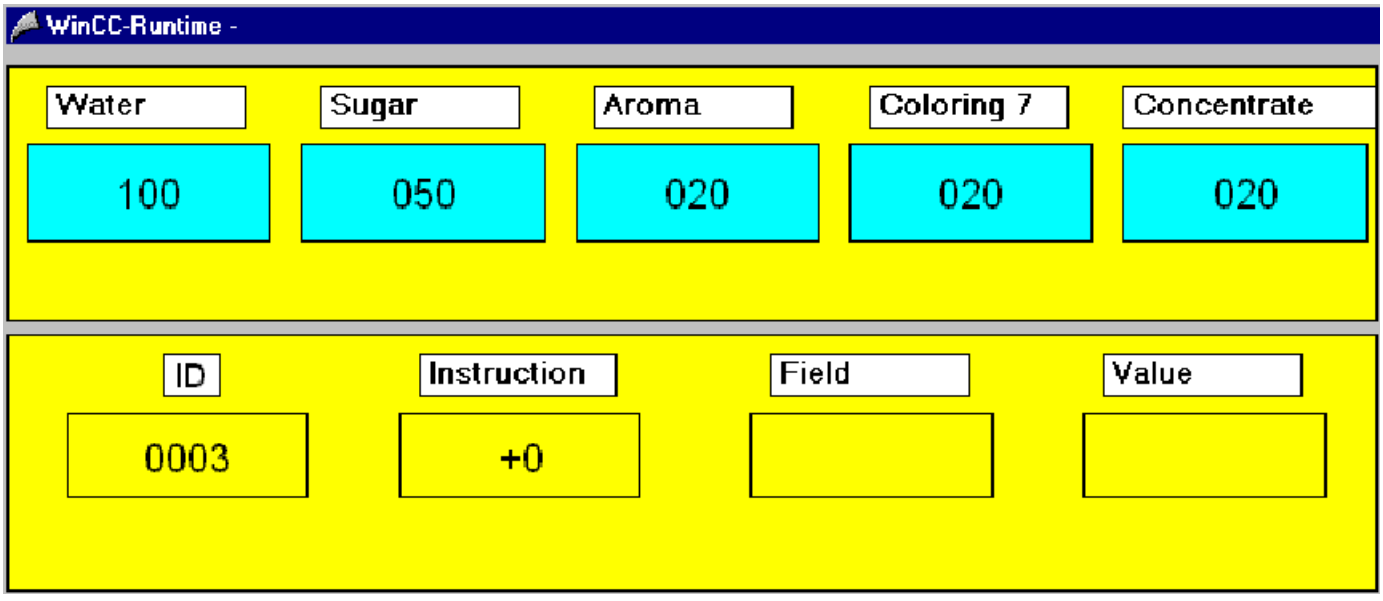

Procedure

To transfer recipe data to the recipe memory of the HMI device, follow these steps:

1. Enter the recipe values displayed in the blue fields "Water", "Sugar", "Aroma", "Coloring 7" and "Concentrate". The recipe values form the recipe data record you want to transfer.
2. Enter "3" as "ID" for the target recipe data record.
3. Enter "6" as job for reading in the "Instruction" field.

The entered recipe data are transferred as complete data record. The recipe data record number 3 is overwritten in the recipe memory. The job mailbox returns the value "0" (no error) in the "Job" control tag.

Alternative procedure

1. Enter new recipe values in the blue fields " Water", "Sugar", "Aroma", "Coloring 7" and "Concentrate".
2. Enter the recipe element "Coloring 7" as the "Field".

3. Enter the recipe element "020" as the value in the "Value" field.
4. Enter "6" as job for reading in the "Instruction" field.

The entered recipe data are transferred as complete data record. The recipe data record whose recipe element " Coloring 7" has the recipe value "020" - in the example the recipe data record number 3 in the recipe memory - will be overwritten. The job mailbox returns the value "0" (no error) in the "Job" control tag.

See also

Displaying recipes (Page 1181)

Control tags (Page 1190)

6.2.5.3 Mouse and keyboard operation (RT Professional)

Introduction

You operate the recipe view in Runtime by buttons in the toolbar. If you do not want to use the toolbar, you can write the "ID" of the required button using any type of dynamization in the object property "ToolbarButtonClick" .

Overview

Symbol	Description	ID
	"Tooltip" Calls the help for the recipe view.	1
	"Configuration dialog" Opens the configuration dialog box in which you change the properties of the recipe view.	2
	"Select data connection" Opens a dialog for selecting a recipe or recipe query. The recipe data are then shown in the table of the recipe view.	3
	"First line" The first data record is displayed in the table by the button.	4
	"Previous line" The previous data record is displayed in the table by the button.	5
	"Next line" The next data record is displayed in the table by the button.	6
	"Last line" The last data record is displayed in the table by the button.	7
	"Delete line" The content of the marked lines are deleted.	8
	"Cut line" The content of the marked lines are cut out.	9

	"Copy line" The content of the marked lines are copied.	10
	"Insert line" The content of the copied or cut-out lines is inserted starting from the marked line.	11
	"Read tags" The contents of the connected WinCC tags are read and written into the recipe elements by the button. The "Tags" communication type must be activated in the displayed recipe to be able to use the button. The recipe elements must be connected with tags.	12
	"Write tags" The contents of the recipe elements are written into the connected WinCC tags. The "Tags" communication type must be activated in the displayed recipe to be able to use the button. The recipe elements must be connected with tags.	13
	"Import Archive" Using a button, a CSV file is imported from the "ua" directory of the project folder into the table of the recipe view.	14
	"Export archive" The original contents of the table are exported with table headers by the button when loading. The recipe is stored as a CSV file in the "ua" directory of the project folder.	15
	"Sorting" Opens a dialog box for setting a customized sorting of the displayed columns.	16
	"Selection dialog" Specifies selection criteria for the columns of the recipe view to be displayed in the table.	17
	"Print" Starts the printout of the displayed values. The print job used during printing is defined in the configuration dialog in the "General" tab.	18
	"Set time base" Opens a dialog box for setting the time base for the used times.	19
	"Export data" You can use this button to export the current Runtime data of a recipe or recipe query to a CSV file. If the "Display dialog" option is activated, a dialog opens in which you can view the settings for export and start the export. With the appropriate operator authorizations you can also select the file and the directory for the export. If no dialog box is displayed, the data export to the preset file starts immediately.	20
	"User-defined 1" Displays the first button created by the user. The function of the button is customized.	1001

Possible elements of the status bar

The following elements may appear in the status bar of the recipe view:

Symbol	Name	Description
Archive: archive1	Archive name	Displays the name of the selected recipe or recipe query.
Row 36	Row	Shows the number of the marked line.
Column 1	column	Shows the number of the marked column.
08.01.2008	Date	Shows the system date.
09:02:20	Time	Shows the system time.
	Time Base	Shows the time base used in the display of times.

Navigation in the table

You can navigate in the table as follows:

- You enter the next cell with the "ENTER" key or with the "Right" cursor key.
- You enter the previous cell with "SHIFT+ENTER" key or with the "Left" cursor key.
- You enter the next line by clicking with the mouse in the line or with the "Down" cursor key.
- You enter the previous line by clicking with the mouse in the line or with the "Up" cursor key.

Note

If the "Error while connecting the data!" error message appears when Runtime starts, the recipe view has no connection to a recipe or recipe query. Check the following:

- The name is specified correctly.
- The configuration is not changed.
- The connected recipe or recipe query still exists.

6.2.5.4 Editing recipe data (RT Professional)

Introduction

You can edit data in the Data recipe view. The following options are available:

- Enter new data
- Change existing data
- Delete lines
- Cut-out, copy and insert lines

Requirement

- You have permitted editing in the configuration dialog in the "General" tab.
- You have deactivated the "Write-protected" property for the column to be edited in the configuration dialog in the "Columns" tab.
- The "ID" column cannot be edited.
- If the recipe view is connected to a recipe query, you cannot delete and cut out a row.

Entering new data in the table

1. Click on to move to the last line. The line is marked.
2. Double-click on the first cell of the marked line. You can also press on "F2", "Alt+Enter" or Ctrl+Enter" in the cell.
3. You enter the values in the cells one after the other and confirm each time by pressing Enter. After you have entered all values in the line and selected another line, the new data record is written into the recipe. You move to another line by clicking with the mouse, with the "ENTER" key or with the "Up" and "Down" cursor keys.
4. You can copy the data of a marked line with "CTRL+C" or "CTRL+X" into the clipboard. The copied data is inserted into a marked line with "CTRL+V".

Changing existing data in the table

1. Click on or to move to the desired line. You can also use the scrollbars to move to the desired line.
2. Double-click on the desired cell of the marked line. You can also press on "F2", "Alt+Enter" or Ctrl+Enter" in the cell.
3. You enter the values in the cells one after the other and confirm each time by pressing Enter. After you have entered all values in the line and selected another line, the changed data record is written into the recipe.

Deleting a line in the table

1. Click on or to move to the desired line. You can also use the scrollbars to move to the desired line.
2. Click on to delete the marked line.

Cutting, copying and inserting lines

1. Click on or to move to the desired line. You can also use the scrollbars to move to the desired line.
2. Click on or to cut or copy the data of the line. As an alternative, you can also use key combination "CTRL+ALT+X" or "CTRL+ALT+C".
3. Go to the desired line into which you want to copy the data. Click on to insert the cut-out or copied data. If you do not want to overwrite the data of the marked line, move into the last line to insert the data.

6.2.5.5 Selecting and configuring data (RT Professional)

Introduction

The recipe view shows the data of the connected recipe in a table. The displayed content of the table is determined by the selected columns and the selection of contents of the columns.

Requirement

- You have created one or more recipes and/or recipe queries.
- You have connected the recipe view to a recipe or recipe query.

Configuring the columns

1. Go to the "Columns" tab.

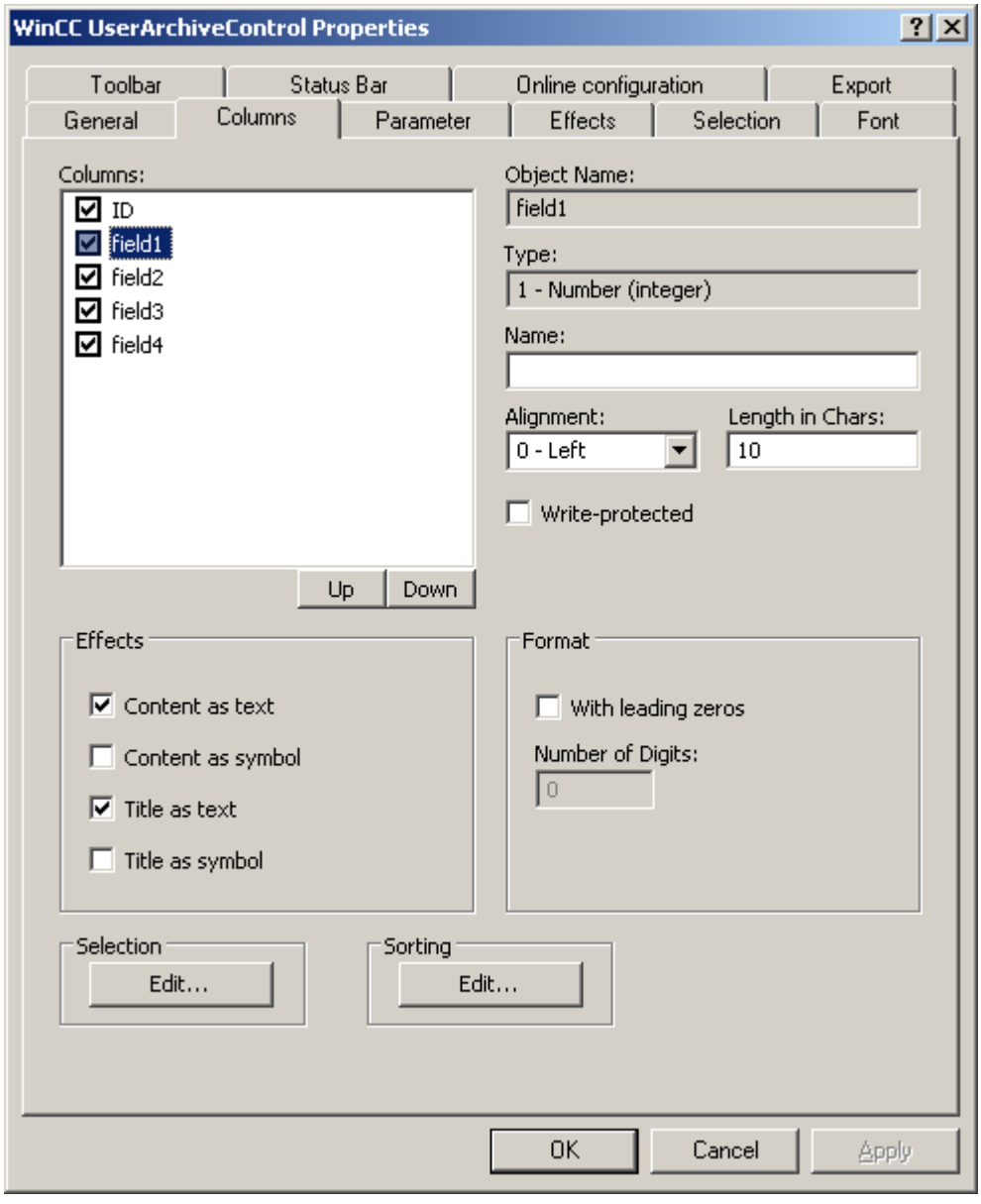

2. In the "Columns" list you see the fields of the connected recipe or recipe query. If you select the check box in front of the name of the column, the column is displayed in the table. Deactivate the check box if you do not want it to appear.
3. Determine the order of the columns in the table using the "Up" and "Down" buttons.
4. Select a column to configure the properties and the format.
5. If necessary, change the width of the column in the table. Enter a value in the "Length in chars" field.

6. Some columns can also show the content and the header as a symbol. Determine how these columns are displayed in the "Display" field. Text and symbols can be displayed at the same time.
7. Save the configuration.

Selection of column content that will be displayed in the table

Configure criteria used to display the content in the columns in the "Selection" area.

Procedure

1. Click "Edit...". The selection dialog is opened.

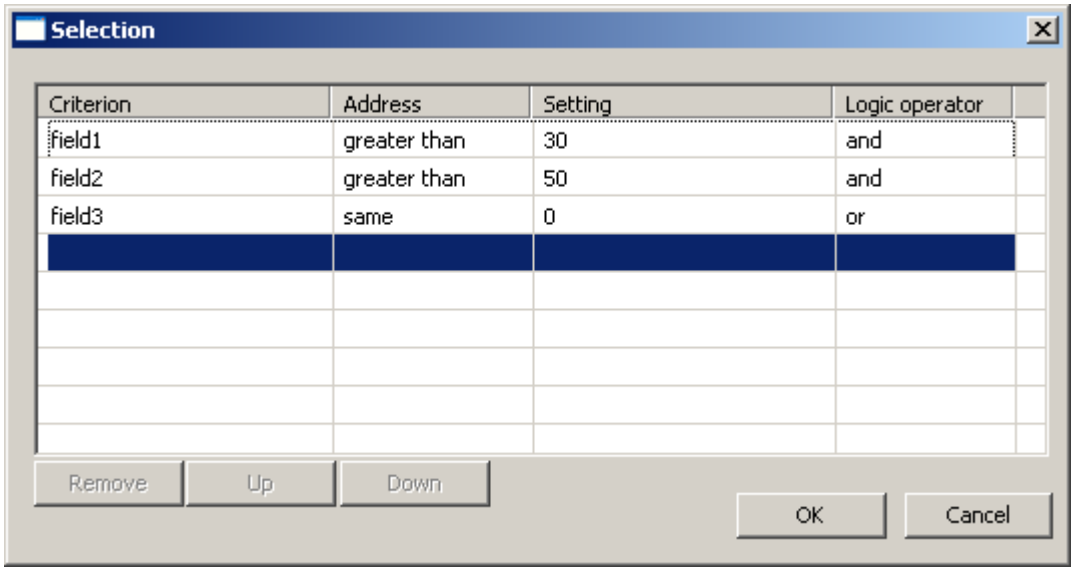

2. Specify the criteria for the display. You can find more detailed information on the selection of columns under "Select data (Page 1212)".
3. Click "OK" to close the selection dialog. The selection is considered at the start of Runtime in the table of the recipe view.

Configuring the Sorting of Columns

You configure the sorting of the columns in the table in the "Sorting" area. You can also specify the sorting criteria in Runtime using the key functions.

1. Click "Edit...". The sorting dialog opens.

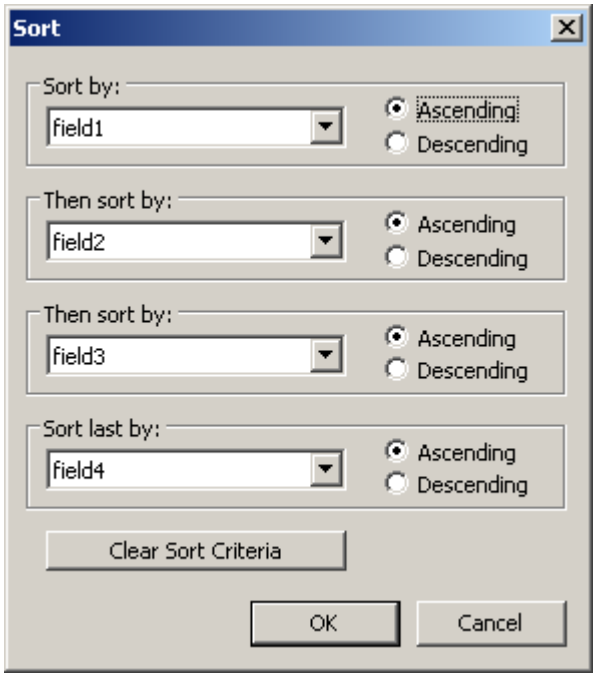

2. Set a sorting sequence.
3. Click "OK" to close the sorting dialog.
4. Save the configuration of the contents.

See also

Select data (Page 1212)

6.2.5.6 Select data (RT Professional)

Introduction

Which contents of the recipe you want to show or export in the recipe view table can be defined in Runtime by the selection dialog box. In the selection dialog box you defie selection criteria with regard to the displayed columns of the recipe.

Requirement

- The selection dialog is open (see section "Selecting and configuring data (Page 1209)").

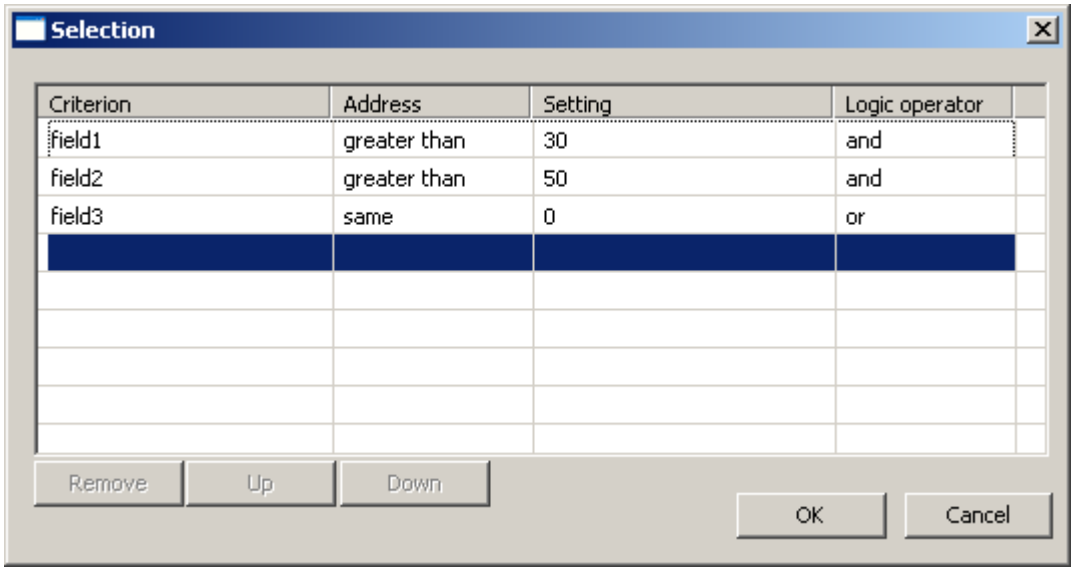

Procedure

Proceed as follows:

1. Double-click in the first empty line in the "Criteria" column. The list with the columns of the recipe view is shown. Select the desired columns, e.g. "field1".
2. Double-click in the "Operand" column to select an operand.
3. Double-click in the "Setting" column to enter a comparison value.
4. Double click in the "Logic operator" column to select an "AND" or "OR" function.
5. Repeat the procedure if you want to define further criteria.
6. Click "OK" to close the selection dialog. The selection is displayed in the table of the recipe view.

Note

Ensuring the display of column content

Make sure of proper use of the settings and connections of criteria.

Incorrect logical operations can lead to the data of the connected recipe not being displayed in the recipe view.

Therefore test every criterion separately before the logical operation and then the logically operated criteria respectively. Check that all expected contents are also displayed in combination.

This ensures that the selection is displayed completely in the recipe view.

See also

Selecting and configuring data (Page 1209)

6.2.5.7 Sorting data (RT Professional)

Introduction

The data in the recipe view can be sorted by columns in Runtime. You sort the columns either via the "Sort dialog" button function or directly via the column headers.

Note

You can define the sorting criteria already in the configuration in the recipe view by clicking the "Edit..." button under "Sorting" on the "Columns" tab.

How to sort with the Sort dialog

Requirement

- You have activated the "Sorting dialog" button on the "Toolbar" tab of the UserArchive Control.

Procedure

1. Click the "Sort Dialog" button .
2. Select the column of the connected recipe from the "Sort by" list which is to be used for sorting first. Select the relevant check box to specify sorting in ascending or descending order. If you want to sort by more than one column, select the other columns in the desired order in the "Then sort by" lists.

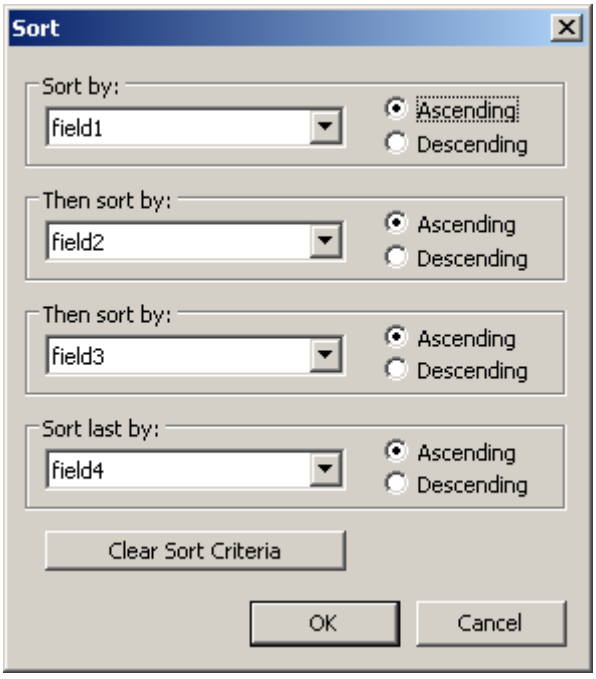

How to sort the column contents with the column heading

When sorting using the column header, you are able to specify the sort order for more than four columns. A sorting icon and sorting index, displayed with right-justification in the column heading, show sorting order and sorting sequence of the column contents.

Requirement

- You have allowed sorting in the list field "Sort by column header" by click or double click on the "Parameters" tab of the recipe view.
- You have activated the "Show sorting icon" and "Show sorting index" checkboxes.

Procedure

1. Click the column header of the column you want to sort as first column. The sorting index "1" is displayed, and the sorting icon points upwards for ascending sort order.
2. If you want to sort in descending order, click the column header again.

3. If the sorting order has been defined with "up/down/none", you can undo the sorting of the column with a third click.
4. If you want to sort several message blocks, click the respective header columns in the desired sequence.

6.2.6 Examples (RT Professional)

6.2.6.1 Example: Creating recipes (RT Professional)

Task

In this example, you will create two recipes.

- A "Beverage" recipe for drinks. The drinks mixing plant produces various drinks. You create a recipe data record for each item.
- An "Orders" recipe for orders. The orders contain the current order data.

Settings

The settings relate to an HMI device which is connected to a SIMATIC S7-300 or SIMATIC S7-400.

In this example, you will need the following tags, recipes, recipe entries and recipe data records:

Tags:

Name	PLC connection	Address	Type
article	Yes	DB 120, DBW 0	String
litrewater	Yes	DB 120, DBW 4	Integer
kilosugar	Yes	DB 120, DBW 8	Integer
gramaroma	Yes	DB 120, DBW 12	Integer
litreconcentrate	Yes	DB 120, DBW 16	Integer
coloring	Yes	DB 120, DBW 20	Integer
customer	Yes	DB 120, DBW 24	String
numbercustomer	Yes	DB 120, DBW 28	String
numberorder	Yes	DB 120, DBW 32	Integer
quantitiy	Yes	DB 120, DBW 36	Integer
filling	Yes	DB 120, DBW 40	Integer
dateorder	Yes	DB 120, DBW 44	Date
status	Yes	DB 120, DBW 48	Integer

"Beverage" recipe:

Recipe element	Associated tag
Article	article
Water	litrewater

Recipe element	Associated tag
Sugar	kilosugar
Aroma	gramaroma
Concentrate	litreconcentrate
Coloring	coloring

Data record ID	1	2	3
Article	Juice	Nectar	Cola Light
Water	5	50	72
Sugar	5	10	35
Aroma	100	300	240
Concentrate	95	50	35
Coloring	FS 0063	FS 0063	FS 1007

"Orders" recipe:

Recipe element	Associated tag
Customer	customer
Number	numbercustomer
Order	numberorder
Article	article
Quantitiy	quantitiy
Filling	filling
Status	status
Date	dateorder

Procedure

1. Create the tags listed above with the settings described above.
2. Create the "Beverage" and "Orders" recipes with the settings indicated above.
3. Select the following properties for the recipes:
   - Under "General > Size" the "Unlimited" type
   - Under "Communication > Communication type" the "Tags" communication type

4. Create the recipe entries in each recipe.

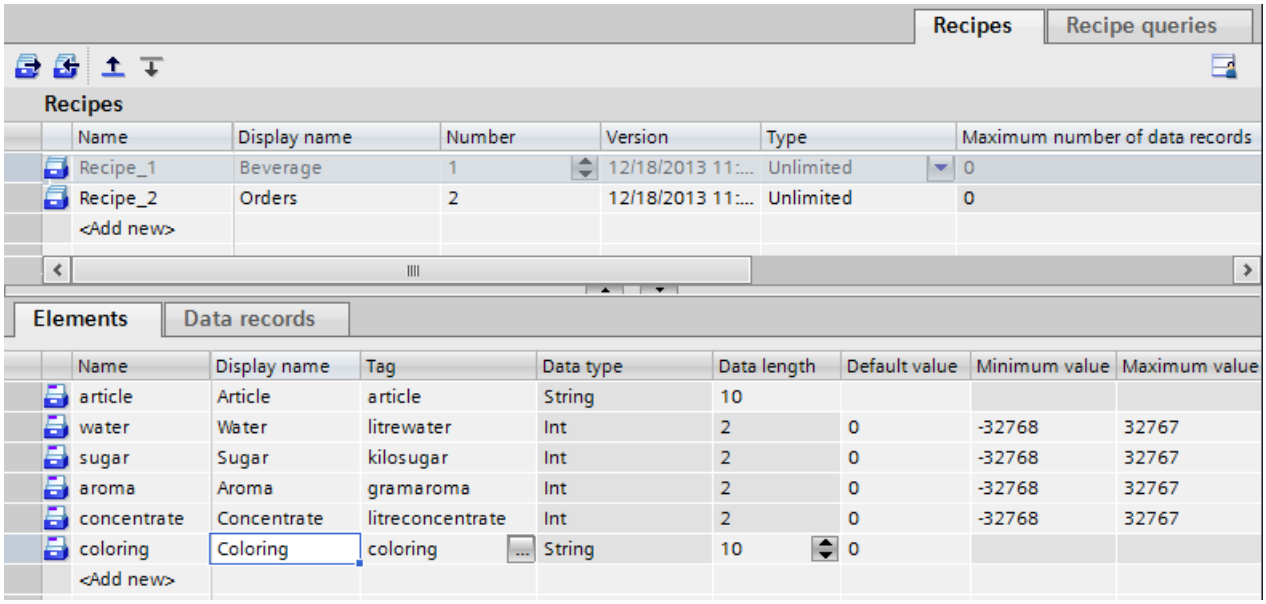

5. Activate the "Unique values" check box for the following elements in the property window under "Extras".
   - In the "Beverage" recipe for the "Article" recipe element
   - In the "Orders" recipe for the "Order" recipe element
6. Create the data records indicated above in the "Beverage" recipe. Enter the values indicated above in each of the data records.

Result

The "Beverage" and "Orders" recipes are created.

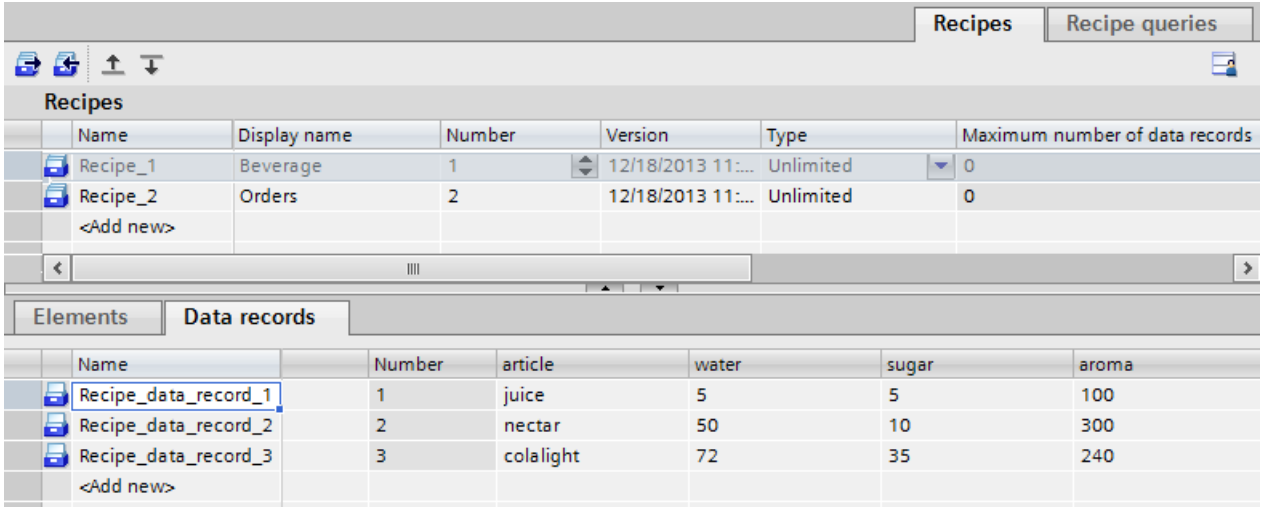

See also

Example: Configuring a recipe view (Page 1220)

6.2.6.2 Example: Creating recipe queries (RT Professional)

Task

In this example, you create the "Label" recipe query. The following recipe elements are displayed in the recipe query:

Recipe element	Recipe
Article	Beverage, Orders
Customer	Orders
Status	Orders
Order	Orders
Filling	Orders
Sugar	Beverage
Coloring	Beverage

The "Article" recipe element is contained as a common element of both recipes.

Requirements

The "Beverage" and "Orders" recipes are created.

Procedure

1. Create the "Label" recipe query. Create the elements listed above.

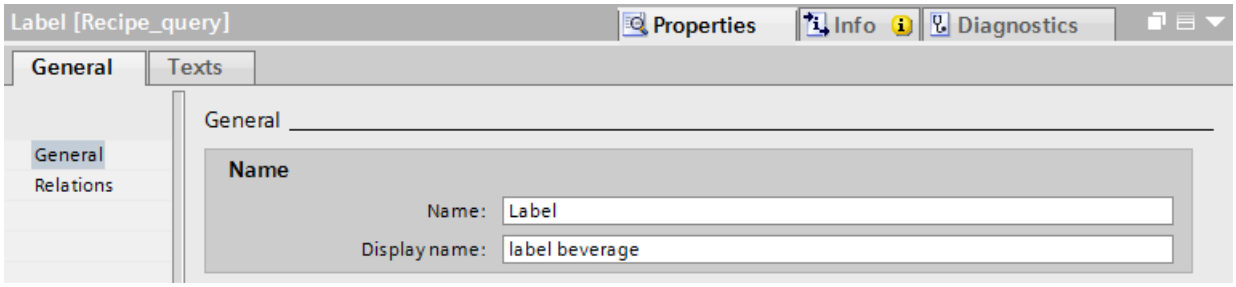

2. Enter the following conditions under "Properties > Relations":
   Beverage.Article = Orders.Article

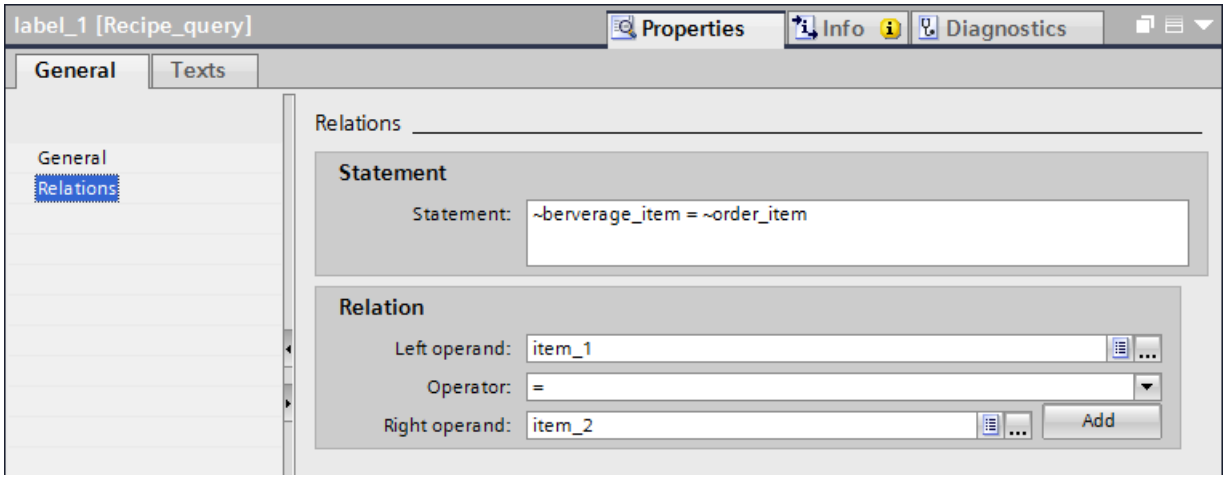

Result

The "Label" recipe query is created.

6.2.6.3 Example: Configuring a recipe view (RT Professional)

Task

In this example, you will add a plant screen with recipe view to the "Label" view.

Requirement

The "Create recipes" and "Create recipe query" examples are created.

You have created and opened the "Drinks mixing plant" process picture.

Procedure

1. Insert the recipe view in the process picture. You will find the recipe display in the toolbox under "Controls".
2. In the Inspector window, use the arrow to select the "Recipe query" type in the "Data source" input field of the "General > View" group.
3. Select the "Label" recipe query.
4. Define additional display options for the recipe view in the "Properties" > "Appearance" group in the Inspector window.
5. Go to the "Properties" > "Columns" group in the Inspector window.
   – Activate the check box in the "Visible" and "Protected" columns for all recipe elements.
6. In the Inspector window, switch to the "Properties" > "Toolbar" group. Activate the check box in the "Visible" column for the following buttons under "Toolbar - Buttons".
   – "First line"
   – "Previous line"
   – "Next line"
   – "Last line"
   – "Sort dialog"
   – "Selection dialog"
7. Go to the "Properties" > "Columns" group in the Inspector window. Enter the following sorting in the "Sort" column.
   – By job number
   – Ascending

Result

In Runtime,the recipe view is opened with the "Label" recipe query in the "Drinks mixing plant" process picture. The data records are displayed sorted by job number. You can use the buttons to switch between the data records.

Use the "Status" data field to display the current job or jobs that have been run.

You can adapt the recipe display to the data records as required with the "Sort dialog" and "Selection dialog" buttons.

See also

Example: Creating recipes (Page 1216)

6.2.6.4 Using control tags (RT Professional)

Example for the use of control tags:

You need to perform the following steps to work through the example with the control tags:

Requirement

You have created the "Drinks" recipe.

Properties of the "Cola" recipe	
Log type	"Unlimited"
Communication	"Tags"
Data type for the "ID" control tag	Long
Data type for the "Job" control tag	Long
Data type for the "Field" control tag	String (1 character)
Data type for the "Value" control tag	String (1 character)

Procedure

1. Create the control tag with the following properties:

Control tag	Data Format	Output format
ID	decimal	0999
Job	decimal	s9
Field	String	*
Value	String	*

1. Select the "Update for changes" object property for each control tag.
2. Connect the control tags to the "Drinks" recipe.
3. Create a new screen and add a recipe view.
   – Select the "Drinks" recipe.
   – Check the "Paste", "Change" and "Delete" check boxes under "General".
4. Create an I/O field for every configured recipe element.
5. Connect the I/O fields to the associated tags.
   For example, connect the I/O field for "Colorant" to the "Coloring" process tag.
   – Create a text field for the label of every configured I/O field.
     By doing so, you assign the I/O fields to the individual recipe elements in Runtime.
6. Create an I/O field for every control tag.
   – Create a text field for the label of every configured I/O field.
     By doing so, you assign the I/O fields to the individual control tags in Runtime.
7. Start Runtime or load the project into the HMI device and start the project.

Result

The configuration for the recipe is completed.

Accessing the recipe

You can now access the recipe in the following ways:

- In the table view of the recipe view

ID	Recipe	Water	Sugar	Coffein
1	Calif Cola	90	10	10
2	Cola	80	30	20
3	Cola Light	100	20	20
4	Standard	100	50	20

- Using the configured I/O fields

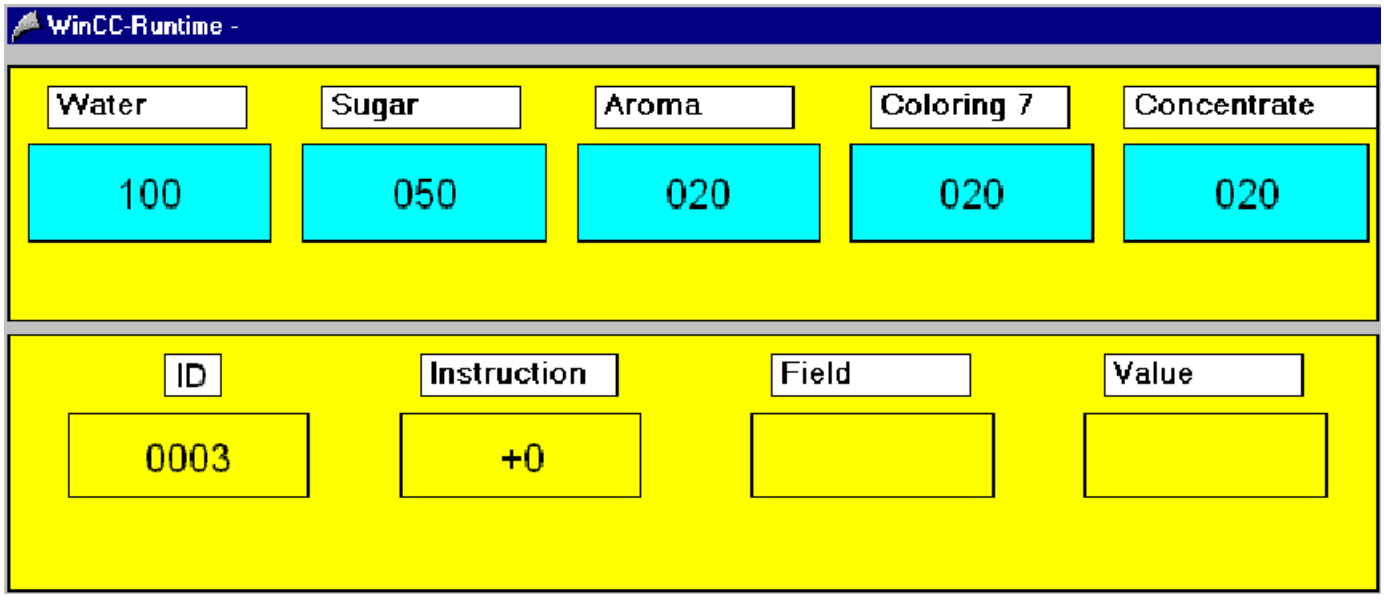

Write the data record to the process tags using the ID

- Enter the ID "3" in the "ID" I/O field and enter a "7" in the "Job" I/O field.
  The values of data record "3" will be written to the process tags in the I/O fields and displayed in the I/O fields.
  If the data record values were read without error from the recipe, error number "0" is displayed in the "Job" field.
  Error number "-1" is output if an error occurs.
  The "Field" and "Value" control tags are not required.

  Note

  The current contents of the process tags are read into the recipe by entering the ID "-1" and the job "6". The new values are appended at the end of the table. The data record IDs are continuously incremented.

Read the process tags via the ID

- Change the values in the I/O fields of the process tags and enter a "5" in the "ID" field. Enter "6" in the "Job" I/O field.
  The changed values of the process tag are written to data record "5" of the recipe. The values that were previously contained in this data record are overwritten.
  The "Field" and "Value" control tags are not required.

Write a data record to the process tag using the "Field" and "Value" control tags

Note

For an element that is referenced with the "Value" control tag, you must select the "Field must possess a unique value" check box under "Properties > Memory options". Otherwise it will not be possible to create a unique assignment between the recipe data record and the value in the element.

- In the "Field" I/O field, enter the word "Item" and write "Cola Light" to the "Value" I/O field. Enter "7" in the "Job" I/O field.
  The values of the "Cola Light" data record will be written to the process tags in the I/O fields and displayed in the I/O fields.
  The "ID" control tag is not needed, so it must be set to 0.

Working with reports (RT Professional, RT Advanced, Panels, Comfort Panels) 7

7.1 Basics (RT Professional, RT Advanced, Panels, Comfort Panels)

7.1.1 Reports (RT Professional)

Introduction

Reports are used to record process data and processed production cycles. You have the opportunity, for example, to create regular shift reports, output batch data, or record the production process for quality control (QC).

Creating reports

A report is created and edited in the "Reports" editor. In this editor, you configure the following report items:

- Formal appearance
  Specify the formal layout of the report in the Inspector window. In this window, for example, you specify the page format, page margins, title page, back page, headers, or footers for the report.
  Moreover, you can specify whether to output the different pages with or without watermark in the reports for Runtime Professional.
- Content
  In the work area, specify the content of the report, for example, the alarms of a shift. To this purpose, insert the corresponding objects into the detail page(s).
  You can also design the title page, back page, headers, and footers as you please. For watermarked pages, create additional pages to be included as "background image" in the output.

The modular structure lets you configure reports that suit all of your applications.

Note

The "Reports" editor is not available at HMI devices that do not support reporting.

Report output

In Runtime, report output is event-driven or time-driven.

- Time-driven: Automatic print at specific dates, times or intervals.
- Event-driven: Printing is initiated by specific events, e.g. click on a button, or a limit is exceeded.

The configuration of report output differs depending on the Runtime version.

In Runtime Advanced, the configured reports are output on the default printer of the HMI device.

For Runtime Professional, make the following decisions with the help of a print job:

- Output of selected pages, or of all pages of the report
- Output of all data, or only the data of a specific period
- Output of the report to a printer or file
- Whether the operator is allowed to select a printer or modify the scope of pages for report output
- Whether the report that is output is displayed in the print job list of an application window configured accordingly

Print function of screen objects

Certain screen objects of Runtime Professional, e.g. the alarm view, contain a default button function (printer icon) for the report output. WinCC employs predefined system reports for the output. You can can customize these to suit your requirements, or replace them with a report of your individual design.

See also

Structure of reports (Page 1226)

Create a report (Page 1238)

Structure of reports (Page 1228)

Principles for preparation of reports (Page 1234)

Principles for report output (Page 1253)

Basics for system traces and system print jobs (Page 1259)

7.1.2 Structure of reports (RT Advanced, Panels, Comfort Panels)

Introduction

A report in WinCC consists of several sections that can be enabled or disabled, as required.

Sections of a report

The following figure shows an example of the different sections of a report in the "Reports" editor.

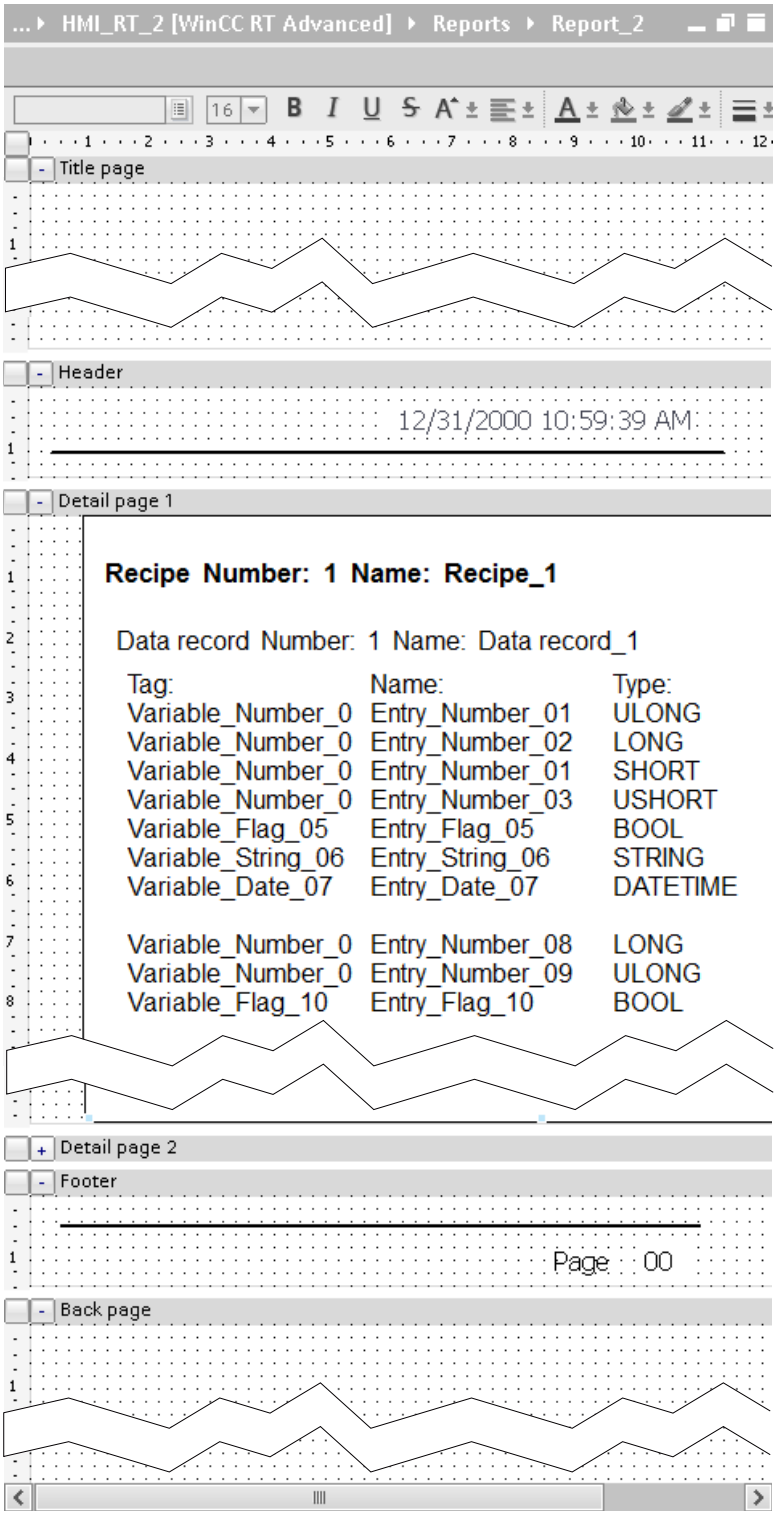

Title page and back page

The title page contains important information about the report content. The back page is used, for example, as Impressum, on which you provide contact information of shift managers or service technicians. The title page and back page are output separately on a single page. Each of them consist of exactly one page and page breaks are not used.

Detail page

Configure the output of runtime data such as recipe or alarm reports on the detail pages of the report.

Use the shortcut menu on the detail page to insert additional detail pages or change their order.

Header and footer

The header and footer are output on each detail page of the report. You typically insert the page numbers or the date in the header or footer.

See also

Create a report (Page 1238)

Reports (Page 1225)

Principles for preparation of reports (Page 1231)

7.1.3 Structure of reports (RT Professional)

Introduction

A report in WinCC consists of several sections that can be enabled or disabled, as required.

Sections of a report

The following figure shows an example of the different sections of a report in the "Reports" editor.

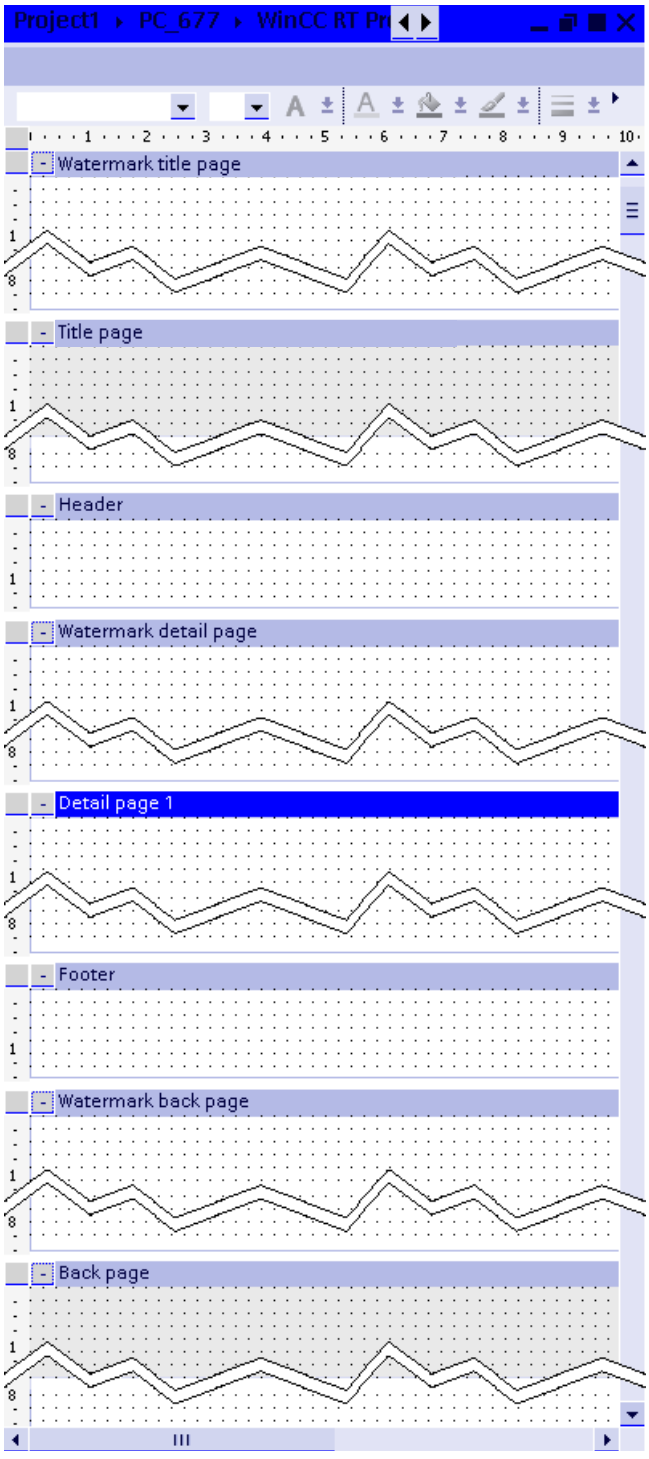

Title page and back page

The title page contains important information about the report content. The back page is used, for example, as Impressum, on which you provide contact information of shift managers or service technicians. The title page and back page are output separately on a single page. The page has a precise length and no page break, regardless of whether you configured a dynamic object to be output on that page.

Detail page

Configure the output of runtime data such as recipe or alarm reports on the detail pages of the report.

Use the shortcut menu on the detail page to insert additional detail pages or change their order.

Header and footer

The header and footer are output on each detail page of the report. You typically insert the project name, report name, page numbers or the date in the header or footer.

Watermarks

The title, detail and back pages can be output with a watermark. The page is printed with its superimposed watermark.

With the help of a watermark, output an object "A" next to an object of variable length "B" without displacing object "A". Example: You want to output an output field (A) next to a table (B) on all pages.

The title and back pages, as well as the detail pages have separate watermarks.

See also

Reports (Page 1225)

Principles for preparation of reports (Page 1234)

Create a report (Page 1238)

7.2 Working with reports (RT Professional, RT Advanced, Panels, Comfort Panels)

7.2.1 Creating reports (RT Professional, RT Advanced, Panels, Comfort Panels)

7.2.1.1 Principles for preparation of reports (RT Advanced, Panels, Comfort Panels)

Introduction

A report in WinCC consists of several sections that can be enabled or disabled, as required.

- Title page
- Back page
- Detail pages
- Headers and footers for the detail pages

Note

Up to 35 detail pages are available to you per report.

Procedure

The following figure shows the general procedure for creating a report:

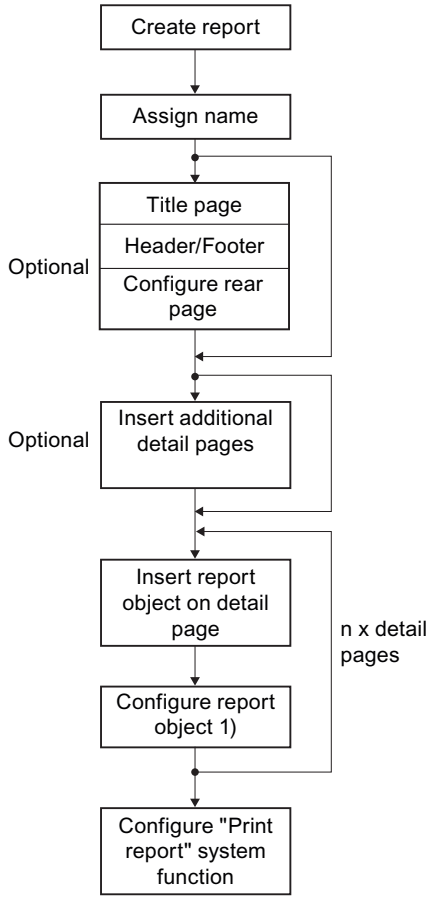

1) Settings depend on the report object used.

Tools for the design of reports

The tools that you can use to design reports are available in the "Tools" task card.

- For *graphic design* insert "basic objects", "elements" and "graphics" in the different sections of the report. For example, for a Logo in the header, separation lines and a page number in the footer.
- Using the "Controls" in the detail pages, configure the *output of runtime data*.

Note

The "Screens" editor provides many objects for designing a report – however, with restricted functionality. Data input properties are not available. The I/O fields in the reports, for example, serve only to output data.

Position and size of objects

Insert the objects at the position where you want them to be output in the report. The objects are output in their configured size and with the following special features:

On the detail pages, configure "recipe reports" or "alarm reports" that serve to output runtime data in tabular format.

- The *width* of the table is set up automatically to fit the width of the detail page. You cannot modify the width.
- WinCC automatically wraps the *height* of the table to fit the data content for output in the report. The table can be continued on the next page.
  In the detail page, WinCC automatically extends tables of dynamic length to the bottom margin.

The following figure shows an example of a detail page in the report and its output in Runtime:

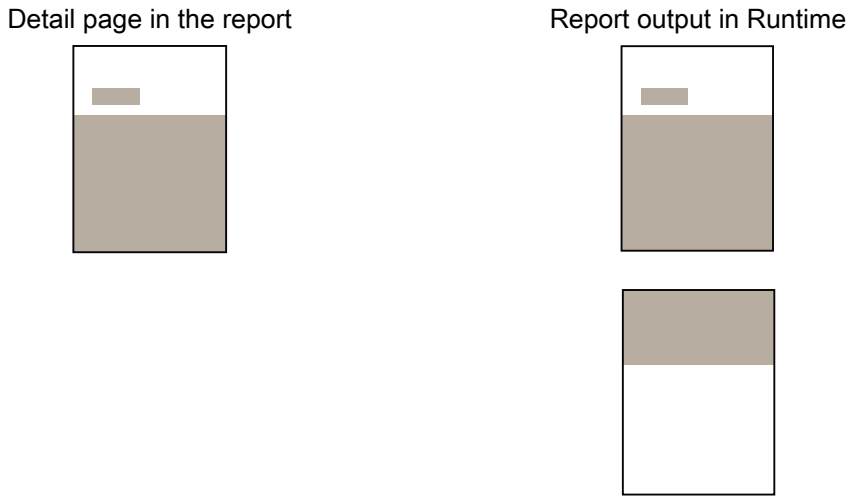

Additional detail pages

Use the shortcut menu on the detail page to insert additional detail pages or change their order.

See also

Configuring reporting of alarms (Page 873)

Structure of reports (Page 1226)

Create a report (Page 1238)

Administration of detail pages (Page 1250)

Objects in reports (Page 1279)

7.2.1.2 Principles for preparation of reports (RT Professional)

Introduction

A report in WinCC consists of several sections that can be enabled or disabled, as required.

- Title page
- Back page
- Detail pages
- Headers and footers for the detail pages
- Watermarks for title page, back page, detail page

Note

Up to 35 detail pages are available to you per report.

Procedure

The following figure shows the general procedure for creating a report:

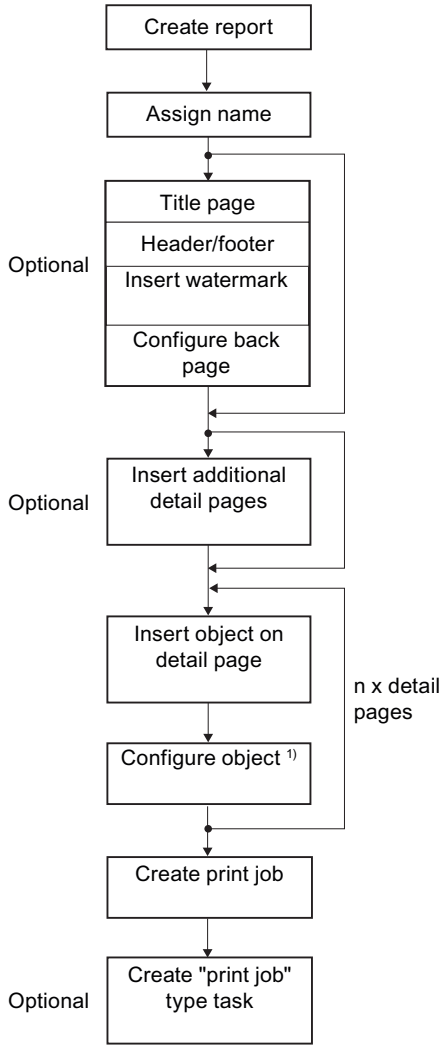

1) Settings depend on the report object used.

Tools for the design of reports

The objects that you can use to design the reports are available in the "Tools" task card.

Note

The "Screens" editor provides many objects for designing a report – however, with restricted functionality. Data input properties are not available. The I/O fields in the reports, for example, serve only to output data.

Positioning objects

Insert the objects at the position where you want them to be output in the report. However, you should be aware that WinCC ignores white spaces at the start of the *detail page* when the report is output. The top object marks the start of the detail page on the printed copy.

The following figure shows an example of a detail page in the report and its output in Runtime:

Detail page in the report

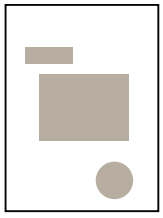

Report output in Runtime

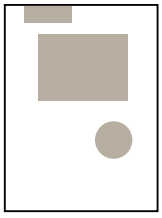

To include a white space in the output, extend the header accordingly. You can also place an "invisible" object on the detail page at the required start. For example a white line that is not visible on white paper.

Size of objects

Objects are output in their configured size.

On the detail pages, configure "controls" and "report objects" for the output of Runtime data. WinCC aligns the Runtime data to fit the object frame – with one special feature: If you configure an *object like a table with dynamic length*, WinCC automatically adjusts the height of the table to fit the amount of data to print. The table can be continued on the next page. WinCC adapts no more than the object height and places all following objects underneath the the object, as configured in the detail page.

The following figure shows an example of a detail page in the report and its output in Runtime:

Detail page in the report

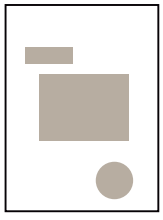

Report output in Runtime

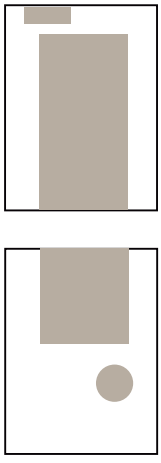

Usually, the "controls" and "report objects" are assigned a dynamic length and output Runtime data in tabular format. The following objects are included:

- Recipe report
- Alarm report
- Table view
- ODBC database table
- Alarm sequence report
- CSV provider tables
- Control Hardcopy
- Control Data
- Control information

Objects next to dynamic objects

In a detail page, do not place other objects next to objects that have a dynamic length. Given the fact that the position of the objects cannot be clearly defined, WinCC possibly outputs these with errors. Instead, configure the objects as watermark.

The following figure shows an example of a watermark and a detail page in the report and the output in Runtime:

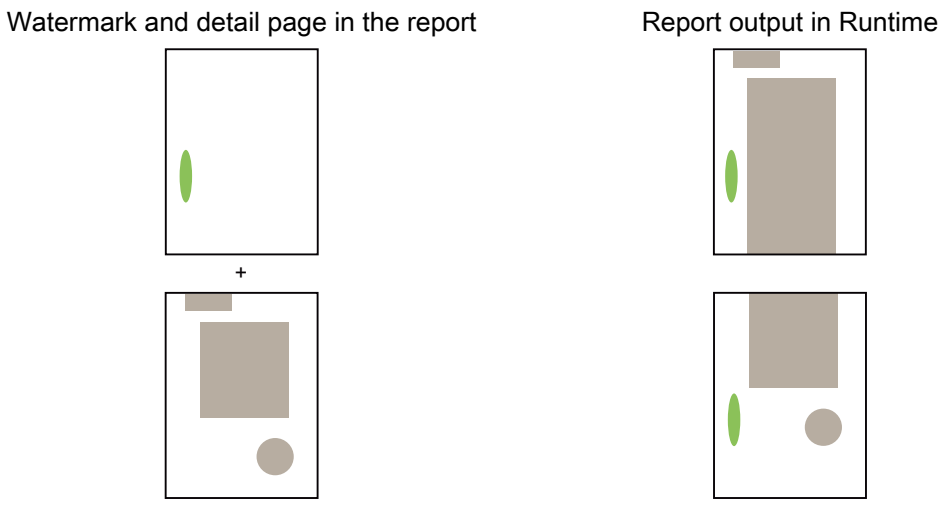

Title page and back page

The title page and back page are output separately on a *single* page. WinCC will not continue output on the next page if you configured an object with dynamic length for the title or back page. Runtime data of the object exceeding the output capacity of a single page is not output to the full extent in the report.

Additional detail pages

Use the shortcut menu on the detail page to insert additional detail pages or change their order.

See also

Report design tools (Page 1240)
CSV file for trend output (Page 1244)
CSV file for output of a table (Page 1247)
Continuous alarm reporting (Page 1243)
Reports (Page 1225)
Structure of reports (Page 1228)
Create a report (Page 1238)
Administration of detail pages (Page 1250)
Principles for report output (Page 1253)
Objects in reports (Page 1279)

7.2.1.3 Create a report (RT Professional, RT Advanced, Panels, Comfort Panels)

Requirement

A project with an HMI device is open.

Procedure

To create a report, proceed as follows:

1. Double-click "Add new report" under "Reports" in the project navigation.
   A new blank report is displayed in the "Reports" editor.
2. Select the "Report properties" command in the shortcut menu of the report.

3. In the "Properties > Properties > General" area of the Inspector window, specify whether you want to configure the "Title page", "Back page", "Header" and "Footer" in the report. The report sections are updated accordingly.

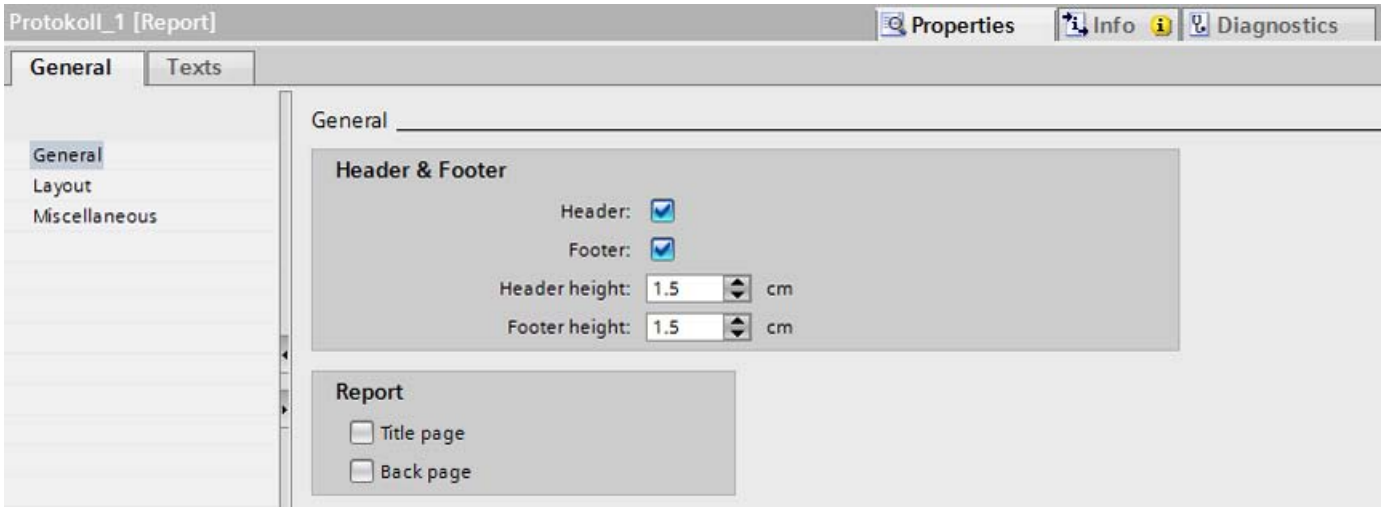

4. Configure the format, the page layout, and the page margins of the report under "Properties > Properties > Layout."

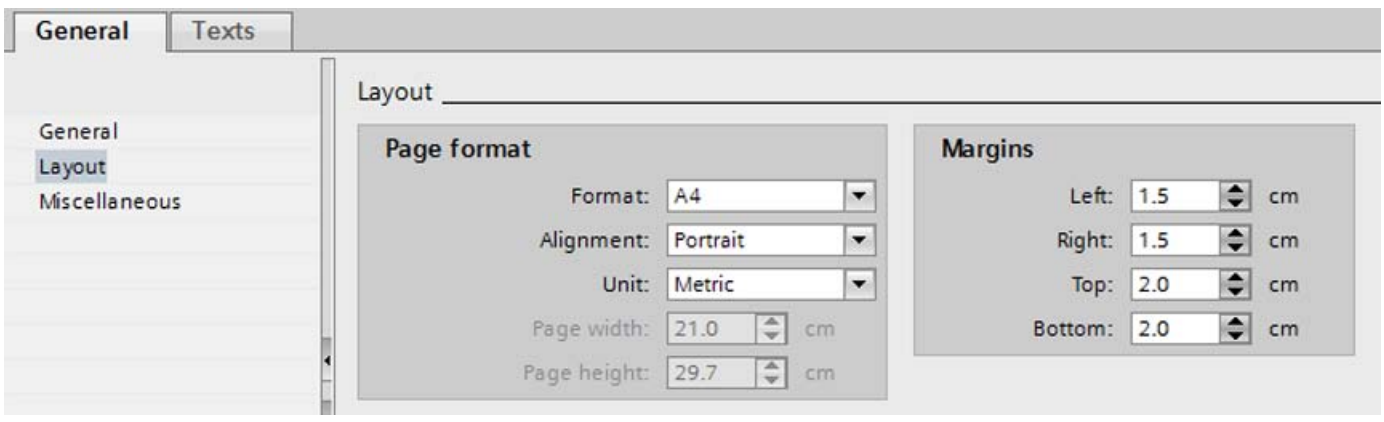

5. Enter a meaningful report name under "Properties > Properties > Miscellaneous."
6. Design the report sections as required.
   Drag and drop the necessary basic objects, elements, graphic images and controls from the "Tools" task card to the required position.
   Alternatively, you can also copy or move objects already configured from a screen to the report.
7. Configure the objects in the Inspector window:

See also

Report design tools (Page 1240)

CSV file for trend output (Page 1244)

CSV file for output of a table (Page 1247)

Continuous alarm reporting (Page 1243)

Configuring reporting of alarms (Page 873)
Reports (Page 1225)
Structure of reports (Page 1226)
Structure of reports (Page 1228)
Principles for preparation of reports (Page 1234)
Working with objects (Page 1263)
Objects in reports (Page 1279)

7.2.1.4 Report design tools (RT Professional)

Tools for the design of reports

In the "Reports" editor, use the objects in the "Tools" task card to design the reports:

- For *graphic design* insert "basic objects", "elements" and "graphics" in the different sections of the report. For example, for a Logo in the header, separation lines and a page number in the footer.
- Using the "Controls" in the detail pages, configure the *output of runtime data*.
- If you want to use your own reports instead of the system reports for the print function of a screen object, configure the output of Runtime data with the help of the "Report objects".

Note

The "Screens" Editor provides many objects, though with restricted functionality, that you can use to design a report. Data input properties are not available. The I/O fields in the reports, for example, serve only to output data.

Elements

The following table provides an overview of the objects of the "Elements" panel:

Object	Application
Date/time field	For output of the current data and time You can customize the output format.
I/O field	For the output of process values You can choose between different output formats.
Project name	For output of the project name The name can be output with or without path definition.
Report name	For output of the report name
Page number	For output of the page number and total number of pages You can customize the output format.

Output of data in tabular format

For a time-driven or event-driven output of data in table form, configure the following controls in reports:

Control	Use in report
Table view	For time-driven output of the values of one or several logging tags Each row of the table represents the status of the selected logging tag at a specific time.
ODBC database table	For the output of data that does not originate from WinCC The data is output in tabular format via the ODBC interface.
CSV provider tables	For the output of data of a CSV file The data of a CSV file must conform to a specific structure. You can select a CSV file, or configure a tag for dynamic selection of the CSV file.

A print function is already integrated in the "Table view" control in the "Screens" editor (button in Runtime). The "Table view" uses preconfigured system reports for printing. You can specify in the system reports if you want to output all data or only the current data displayed in the object (snapshot). You match system reports to your requirements or use individually designed reports instead of the system reports.

Data output in trends

For a time-driven or event-driven output of data in trend form, configure the following controls in reports:

Controls	Use in report
f(t) trend view in reports	For the output of logging tags as a function of time
f(x) trend view in reports	For the output of logging tags as a function of other logging tags, e.g. the temperature profile as a function of pressure You can also compare the trend to a setpoint trend.
CSV provider trends	For the output of data of a CSV file The data of a CSV file must conform to a specific structure. You can select a CSV file, or configure a tag for dynamic selection of the CSV file.

A print function is already integrated in the "f(t) trend view" and "f(x) trend view" controls in the "Screens" editor (button in Runtime). For printing, these controls use preconfigured system reports that you can customize to suit your requirements.

Output of recipes

For a time-driven or event-driven output of recipe data, configure the following controls in reports:

Control	Use in report
Recipe view	For the output of recipe elements

A print function is already integrated in the "Recipe view" control in the "Screens" editor (button in Runtime). This control uses preconfigured system reports for printing. You can specify in the system reports if you want to output all data or only the current data displayed in the object

(snapshot). You match system reports to your requirements or use individually designed reports instead of the system reports.

Output of alarms

WinCC offers several options for alarm reporting.

For a time-driven or event-driven output of alarms, configure the following control in reports:

Control	Use in report
Alarm view	For the output of alarms that occur during the process in a plant. WinCC outputs current alarms, suppressed alarms or alarms from the historical alarm list (short-term), depending on the configuration.

A print function is already integrated in the "Alarm view" control in the "Screens" editor (button in Runtime). This control uses preconfigured system reports for printing. You can specify in the system reports if you want to output all alarms or only the current alarms displayed in the object (snapshot). You match system reports to your requirements or use individually designed reports instead of the system reports.

WinCC lets you continuously output alarms after the start of Runtime, for example, to a line printer. All you have to do is activate this output. WinCC offers two preconfigured system reports that you can adapt to meet your requirements: one for output in the line layout and one for output in the page layout.

Output of a snapshot from Runtime

To output a snapshot of Runtime in a report, configure the following control in reports:

Control	Use in report
Hardcopy	For the output of a snapshot from Runtime WinCC outputs the entire content of the screen, of the active window, or of an individually selected screen area, depending on the configuration.

Data output via ODBC interface

To read data from external sources using the ODBC interface and output them in a report, configure the following controls in reports.

Control	Use in report
ODBC database field	For the output of data from a field of an external database table
ODBC database table	For the output of external database tables

Report objects

Certain screen objects, e.g. in the alarm view, contain a default button function (printer icon) for the report output.

For reports that you want to use for the printing a screen object instead of a preconfigured system report, configure the output of Runtime data with the help of the "Report objects".

The following table shows the report objects that you can use in the reports:

Object	Use in report
Control Hardcopy	For output of the screen object as it is displayed in Runtime (snapshot)
Control Data	For output of the current data of the screen object The data is output in tabular format.
Control Information	To output the data source, for example, window title, object name or screen name

See also

Create a report (Page 1238)
Principles for preparation of reports (Page 1234)
Control Data (Page 1280)
Control hard copy (Page 1281)
Control Information (Page 1282)
CSV provider trends (Page 1283)
CSV provider tables (Page 1284)
Date/time field (Page 1286)
f(t) trend view in reports (Page 1289)
f(x) trend view in reports (Page 1290)
Hardcopy (Page 1292)
Alarm view in reports (Page 1293)
ODBC database field (Page 1297)
ODBC database table (Page 1298)
Project name (Page 1300)
Report name (Page 1301)
Recipe view in reports (Page 1302)
Page number (Page 1305)
Table view in reports (Page 1306)
Continuous alarm reporting (Page 1243)

7.2.1.5 Continuous alarm reporting (RT Professional)

Alarm sequence report

An alarm sequence report outputs alarms that occur on the HMI device continuously in chronological order, for example, to a line printer.

To use the preconfigured alarm sequence report, you only need to activate it.

If you only want to output specific alarms, for example, adapt the report to match your needs.

Activation and configuration

The output of the pre-configured alarm sequence report is configured in the "Alarm sequence report" system print job.

To automatically run the system print job at the start of Runtime:

- In the Runtime settings of the HMI device activate "Services > Start sequence of WinCC Runtime > Alarm sequence report".

The alarm sequence report is output in the default line layout to a line printer.

To output the alarm report in the page layout:

- Open the "Alarm sequence report" system print job and deactivate the "Use line report" option under "Properties > Properties > General".

WinCC assigns the appropriate system report to the system print job.

Customize the system reports to suit your requirements:

- "@AlarmSequenceReportForPagePrinter" for output in page layout
- "@AlarmSequenceReportForLinePrinter" for output in line layout

See also

Principles for preparation of reports (Page 1234)

Report design tools (Page 1240)

Create a report (Page 1238)

Alarm sequence report (Page 1294)

Basics for system traces and system print jobs (Page 1259)

Alarm view in reports (Page 1293)

7.2.1.6 CSV file for trend output (RT Professional)

Requirements for a CSV file that is used for the output of trends

To enable output in a table, the data in the CSV file must have a defined structure.

In the report, the "CSV provider trends" object visualizes the CSV file data in trends. The data of a CSV file must conform to the structure described below.

The CSV file must contain a paragraph for the different parts of the graphic object. A paragraph, in turn, can contain several lines. The following paragraphs are mandatory:

- "Graph": Contains the number of trends and specifications concerning visualization of the graphic object
- "Trends": Contains information on the trend visualization
- "Trend values": Contains the values of the trends

The syntax is identical in all paragraphs. The first line marks the beginning of a paragraph and defines the interpretation of the paragraph data. The other lines of a paragraph contain the actual values of the graphic object or trends.

Use a semicolon to separate the values. Insert line breaks to mark the end of lines.

"Graph" paragraph:

The first line of the "Graph" paragraph must contain the following entry:

#Trend_T; Name; Curves; DateFrom; DateTo; Common Y-Axis; Font; Fontsize

The second line has to contain the values for the graphic object.

Name	Meaning	Comment
Name	Name of the trend view or file	
Curves	Number of trends	
DateFrom	Start of the time range	Notation: 2000-10-30 10:15:00.000
DateTo	End of the time range	Notation: 2000-10-30 10:15:00.000
Common Y-Axis	Common Y axis	
Font	Font	
Fontsize	Font size	

Example:

Line	Content
1	#Trend_T; Name; Curves; DateFrom; DateTo; Common Y-Axis; Font; Fontsize
2	"TrendControl1";3;"2001-10-10 16:30:00.000";"2001-10-10 16:40:00.000";0;"Arial";10

"Trends" paragraph:

The first line of the "Trend" paragraph must contain the following entry:

#Curve; Num; Name; Count; dMin; dMax; Color; Width; CurveType; Filling

For each trend that is visualized in the graphic object, the paragraph must contain a separate line that specifies the trend values.

Name	Meaning	Comment
Num	Trend number	beginning with 0
Name	Trend name	
Count	Number of values	
dMin	Low limit of the trend (for scaling)	
dMax	High limit of the trend (for scaling)	
Color	Color of the Y axis	in hexadecimal notation
Width	Line thickness in points	e.g. 1.5
CurveType	Trend type	LINE, DOTS, STEP
Filling	Fill color for fields	0=no, 1=yes

Example:

Line	Content
1	#Curve; Num; Name; Count; dMin; dMax; Color; Width; CurveType; Filling
2	0;"Temperature";3;10;30;0x00ff0000;1;STEP;0
3	1;"Pressure";5;0;50;0x0000ff00;2;DOTS;0
...	

Structure of the "Trend values" data record

The first line of the "Trend values" paragraph must contain the following entry:

#Data; Num; Date; Value; Flags; Color

For each value that is visualized in the trends, the paragraph must contain a separate line that specifies the coordinates.

Name	Meaning	Comment
Num	Trend number	beginning with 0
Date	Value on the X axis of the trend	Notation: 2000-10-30 10:15:00.000
Value	Value on the Y axis of the trend	
Flags	Limits / time overlap	
Color	Color of the trend section	in hexadecimal notation If you did not define a color for the trend, the color of the associated Y axis is applied to the trend.

Example:

Line	Content
1	#Data; Num; Date; Value; Flags; Color
2	0;"2001-10-10 16:30:00.000";22;0;0x000000FF
3	0;"2001-10-10 16:31:00.000";24;0;
...	

See also

Create a report (Page 1238)

Principles for preparation of reports (Page 1234)

CSV provider trends (Page 1283)

7.2.1.7 CSV file for output of a table (RT Professional)

Requirements for a CSV file to be output in tabular format

In the report, the "CSV provider table" object visualizes the CSV file data in tabular format in the report. The data of a CSV file must conform to the structure described below.

The CSV file must contain a paragraph for the different parts of the table. A paragraph, in turn, can contain several lines. The following paragraphs are mandatory:

- "Table": Specifies the number of columns, the font, and the font size
- Columns": Specifies the headers, widths, and alignment of the columns
- "Data" Contains the data

The syntax is identical in all paragraphs. The first line marks the beginning of a paragraph. The first line describes how the data of the paragraph are interpreted. The other lines of a paragraph contain the actual values of the graphic object or trends.

Use a semicolon to separate the values. Insert line breaks to mark the end of lines.

"Table" paragraph

The first line of the "Table" paragraph must contain the following entry:

#Table; Name; Columns; Font; Fontsize

The second line has to contain the values for the table.

Name	Meaning	Comment
Name	Name of the table view	
Columns	Number of columns	
Font	Font	
Fontsize	Font size	

Example:

Line	Content
1	#Table; Name; Columns; Font; Fontsize
2	"Tab_9";4;"Arial";14

"Columns" paragraph

The first line of the "Columns" paragraph must contain the following entry:

#Column; Num; Header; Width; Alignment

For each table column to be visualized in the graphic object, the paragraph must contain a separate line that specifies the column values.

Name	Meaning	Comment
Num	Column number	beginning with 0
Header	Column header	

Name	Meaning	Comment
Width	Ratio of the width between columns	Example: For a table with three columns, you specify the following widths: 2, 4, and 8. As a result, the third column is output with twice the width of the second and three times the width of the first column. Column width in characters = [table width in number of characters] x [Width] / [Total of Width all values of the columns]
Alignment	Alignment	Left/Central/Right

Example:

Line	Content
1	#Column; Num; Header; Width; Alignment
2	1;"Number";6;L
3	…
4	…

"Values" paragraph

The first line of the "Values" paragraph must contain the following entry:

#Data; Color; Col1; Col2; Col3; ...

For each line of the table, this paragraph must contain a separate line that specifies the relevant column values.

Name	Meaning	Comment
Color	Line color	Notation as in HTML as six-digit Hex number: 0xbbggrr (hex blue blue green green red red)
Col1	Data of column 1	
Col2	Data of column 2	
Col3	Data of column 3	

Example:

Line	Content
1	#Data; Color; Col1; Col2; Col3; Col4
2	0xFF00FF;"05/06/02";"15.55.53";86;"+/-"
3	0x32b400;"05/06/02";"15.55.54";87;"+/-"
4	0x32b400;"05/06/02";"16.00.00";82;"+/-"

Preparing data in the spreadsheet program

As demonstrated in the examples above, the first value in the first line of a paragraph represents the paragraph identifier, e.g. "#Data" as identifier of the "Values" paragraph. The next lines of the paragraph do not contain this identifier.

If you process the lines of a paragraph (of a CSV file) in a spreadsheet program such as Microsoft Excel, the column headers and values will not be congruent.

In order to avoid this effect, insert a semicolon ";" as wildcard at the beginning of the lines that contain the actual values:

- The column headers and values are now correctly aligned vertically in Excel.
- WinCC ignores the semicolon ";" at the beginning of the line and outputs the data correctly in a trend or table.

Example:

Line	Content
1	#Data; Color; Col1; Col2; Col3; Col4
2	;0xFF00FF;"05/06/02";"15.55.53";86;"+/-"
3	;0x32b400;"05/06/02";"15.55.54";87;"+/-"

Control characters for formatting values

The control characters are used to specify value attributes such as the color, font style, or the alignment. The control characters always lead the value. You can combine several control characters: "<B><U>output text", for example, returns the words "output text" in bold and underscored format. No distinction is made between upper and lower case text.

The following table provides an overview of the control characters:

<END>	Ends a sequence of control characters. The remaining text is applied as specified.
<COLOR=#rrggbb>	Font color; notation as six-digit Hex number with the following byte sequence: #rrggbb (red red green green blue blue)
<BGCOLOR=#rrggbb>	Background color on hexadecimal notation
<B>	Bold font style
<U>	Underlined font style
<I>	Italic font style
<STRIKE>	Strikethrough font style
<ALIGN=left>	Text aligned left
<ALIGN=center>	Text centered
<ALIGN=right>	Text aligned right

Note

Different notation for color definitions

Colors are defined by a six-digit hexadecimal number.

In the "Values" paragraph, the line color has the same notation as in HTML, with the following byte sequence:

0xbbggrr (blue blue green green red red).

To format a single value, the colors in the control character <Color> are defined by an inverted sequence:

#rrggbb (red red green green blue blue).

See also

Create a report (Page 1238)

Principles for preparation of reports (Page 1234)

CSV provider tables (Page 1284)

7.2.1.8 Administration of detail pages (RT Professional, RT Advanced, Panels, Comfort Panels)

Adding a detail page

1. Open the shortcut menu of a detail page.
2. Select the command "Pages > Insert page before" or "Pages > Insert page after."
   Depending on the selected command, the new detail page is inserted either before or after the existing detail page.

Deleting a detail page

1. Open the shortcut menu of the detail page that you want to remove.
2. Select the command "Pages > Delete detail page".
   The detail page is deleted.

Sorting detail pages

1. Open the shortcut menu of the detail page that you want to move.
2. Select the "Pages > Move one page up" or "Pages > Move one page down" command.
   Depending on the selected command, the detail page is moved either upward or downward.

Showing and hiding sections

1. To show or hide a specific section, click on the plus or minus sign in the title bar of the section in the working area.
2. If you want to show or hide all sections of a report, select the command "Show all pages" or "Hide all pages" in the report shortcut menu.

See also

Structure of reports (Page 1226)

Structure of reports (Page 1228)

Principles for preparation of reports (Page 1234)

Principles for preparation of reports (Page 1231)

7.2.2 Report output (RT Professional, RT Advanced, Panels, Comfort Panels)

7.2.2.1 Report output in Runtime Advanced (RT Professional, RT Advanced, Panels, Comfort Panels)

Report output in Runtime Advanced (RT Advanced, Panels, Comfort Panels)

Introduction

In Runtime Advanced, report output is event-driven or time-driven.

Configuration steps

Configuring event-driven output:

- An event is configured on a button or tag with the "PrintReport" system function.

Configuring time-driven output:

- In the scheduled tasks, configure a "Print job" task and assign it the desired report.
  In the task properties, specify the time and frequency of report output.

Output in Runtime

On the control panel of the HMI device, the report is output to the printer that is specified as the default printer.

The default printer enabled for a HMI device can be found in the "Printer list". For further information about the "Printer list", refer to the Internet page of Siemens Customer Support and the Article ID "11376409".

See also

Configuring reporting of alarms (Page 873)

Overview of report output (Page 1252)

7.2.2.2 Report output in Runtime Professional (RT Professional)

Overview of report output (RT Professional)

Introduction

In Runtime, report output is event-driven or time-driven.

You can also display the pending print jobs as a list in Runtime. Operators can use this print job list to decide which report they want to output.

Event-driven output

The report is output after an event was generated.

Examples:

- A tag value changes or exceeds a limit.
- Alarm is incoming, outgoing or acknowledged
- Action by the operator, for example, clicking a button

Time-driven output

The report is output automatically.

Examples:

- Once at a specified date, e.g. on December 31, 2010 at 6:30 AM
- At intervals, e.g. daily at 8 am, or on Mondays at 6 pm

Configuration steps

The configuration of a time-driven or event-driven report output differs depending on the Runtime version.

To display the print job list in Runtime, configure the "Application window" control in screens. The application window shows all or only select print jobs, depending on the configuration.

See also

Create a print job (Page 1254)

Configuring automatic start of a print job in the scheduled tasks (Page 1258)

Report output in Runtime Advanced (Page 1251)

Planning tasks (Page 1803)

System reports and system print jobs (Page 1259)

Report output in Runtime Professional (Page 1252)

Principles for report output (RT Professional)

Introduction

In Runtime Professional, report output is event-driven or time-driven.

Configuration steps

- Under Reports, create a print job and assign it the desired report.
  In the print job properties, specify the number of pages, the time range, and the output format (printer or file) of the report.
- To configure a time-driven output, create a task of the type "Print job" in the scheduled tasks and assign the desired print job to this task.
  In the task properties, specify the time and frequency of report output.
- To configure an event-driven output, configure an event to the selected object to run a C Script for calling the "ReportJob" function, e.g.
  - `ReportJob("<Name of print job>","PRINTJOB");` to output the print job
  - `ReportJob("<Name of print job>","PREVIEW");` to output a print preview

Note

Screen objects with preconfigured report output

Certain screen objects, e.g. in the alarm view, contain a default button function (printer icon) for the report output. You can choose to modify the default system reports, or use a report with individual design.

Output in Runtime

The report is written as follows depending on the configuration:

- On the printer that is specified as the default printer in the control panel of the HMI device. The default printer enabled for a HMI device can be found in the "Printer list". For additional information about the "Printer list", refer to the Internet page of Siemens Service & Support and the Article ID "11376409".
- On a network printer that is available for the HMI device.
- In a file in EMF format.

Output of reports

Reports are output to the printer in graphic mode. The use of a serial printer is not recommended because of the accumulated data volume.

For proper output, the printer must support the paper format and page layout of the report.

Note

The value of a tag in the report is read and output at the moment of printing. A substantial period of time may elapse between printing out the first and the last page of a report consisting of several pages. This may lead to the same tag being output with a different value on the last page than on the first page.

See also

Reports (Page 1225)

Create a print job (Page 1254)

Configuring automatic start of a print job in the scheduled tasks (Page 1258)

System reports and system print jobs (Page 1259)

Create a print job (RT Professional)

Introduction

You define the following in the print job:

- Scope of the print job: all pages, or specific pages only
- Time period during which report entries are output
- Printer on which the report is printed
- File to which the report is saved

Requirement

The "Print jobs" editor is open.

Procedure

Proceed as follows to create a new print job:

1. In the "Name" column, double-click in the "<Add>" cell.
   A new print job is inserted.

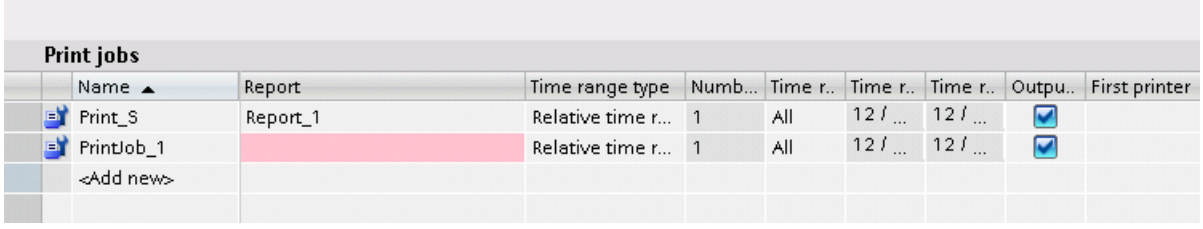

2. In the "Properties > Properties > General" dialog of the Inspector window, enter a "Name" for the print job and select the "Report".
3. Under "Dialog type", specify whether the operator is allowed to modify the report settings prior to output.

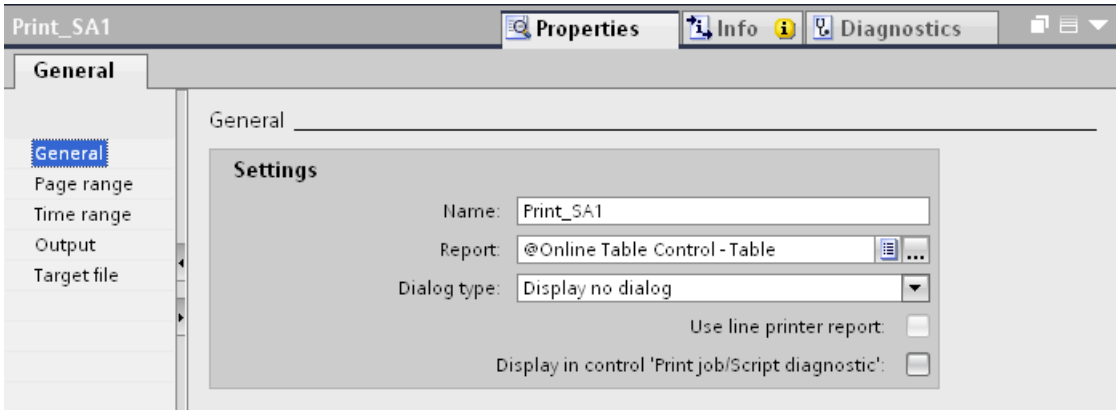

4. Under "Properties > Properties > Page range", specify whether the print job outputs all or only specific pages.

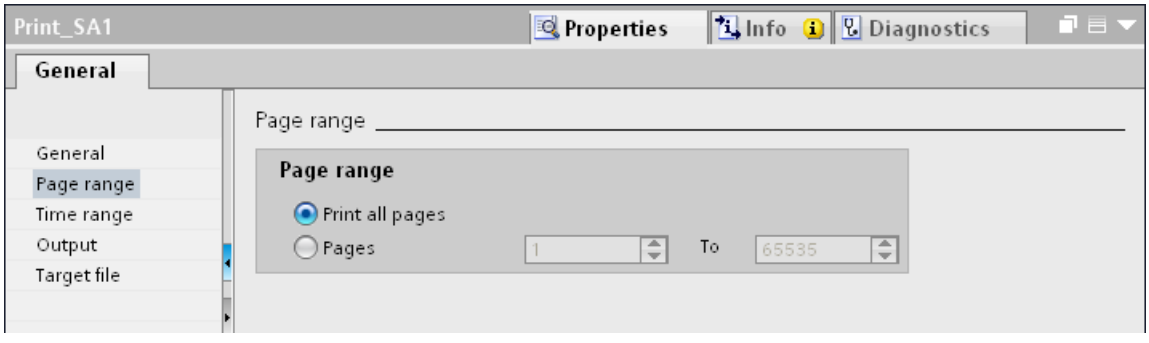

5. Under "Properties > Properties > Time range", specify the time period for the output of report entries.

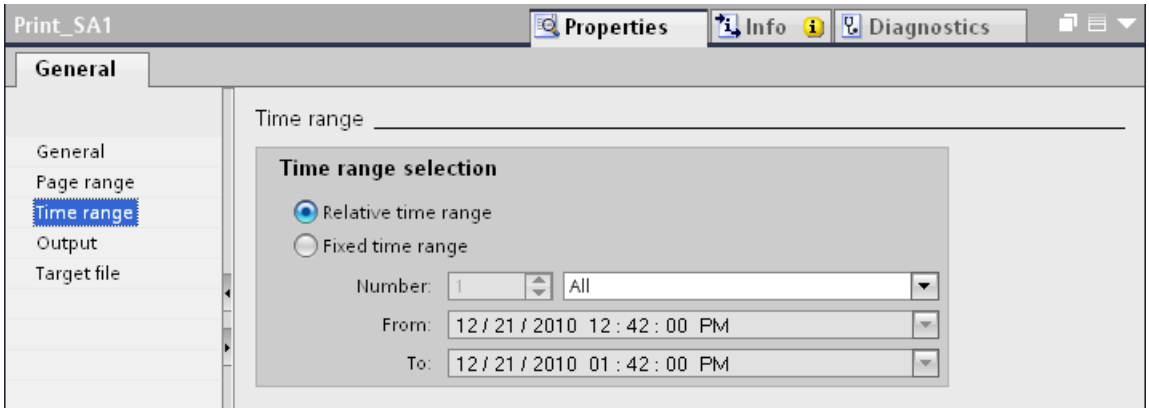

6. If you want to print the report, enable the "Printer" option under "Properties > Properties > Output > Print output".
   The report is output to the default printer of the HMI device.

7. You can select up to three printers if you want to output the report to a different printer.

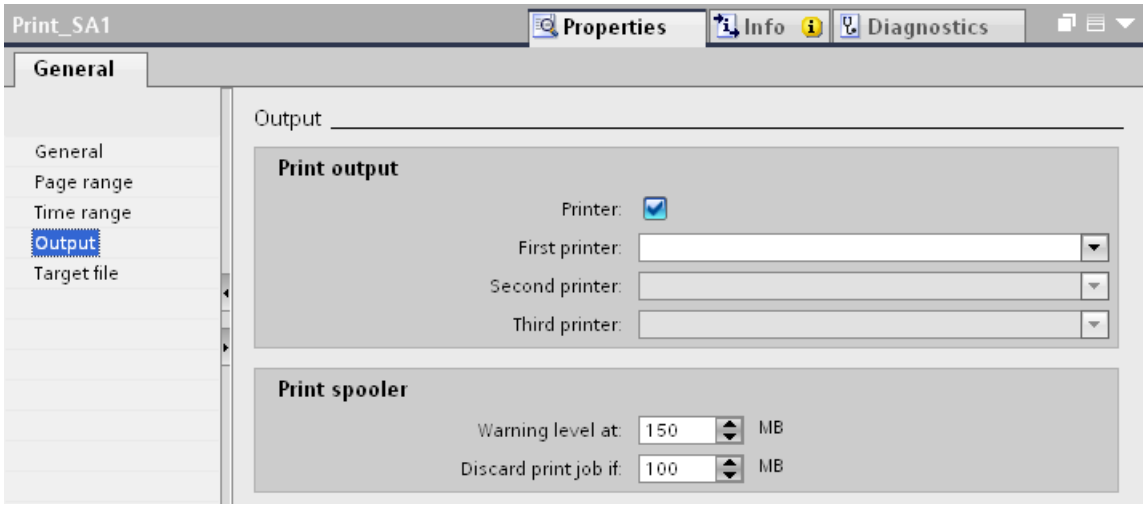

8. If the report should be exported to a file, select "Properties > Properties > Target file > Print to file > Print to file (*.emf)".
   A folder name is created using the information in "Target directory prefix" and the time stamp at the time of printing. The reports are created in the runtime project directory under "PRT_OUT" in a folder of this name.

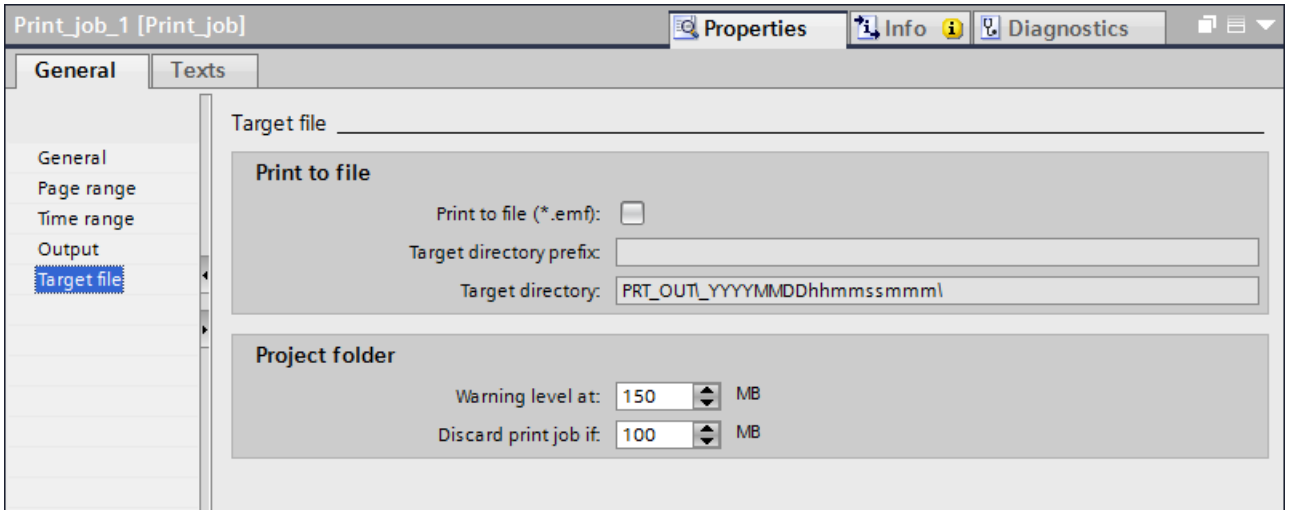

Result

The print job is created for the report.

- For time-driven output in Runtime, create a "Print job" task in the scheduled tasks. Assign the required print job to this task.
- For event-driven output, assign the object an event that runs a C script for calling the "PrintJob" function.

See also

Principles for report output (Page 1253)

Configuring automatic start of a print job in the scheduled tasks (Page 1258)

Configuring automatic start of a print job in the scheduled tasks (RT Professional)

Introduction

A task of the "print job" type specifies the time and frequency of automatic report output (time-driven output).

Requirements

- A print job has been created.

Procedure

To create a task for automatic start of a print job, follow these steps:

1. In the "Runtime settings" of the HMI device activate "Services > Start sequence of WinCC Runtime > Scheduled print jobs in Runtime".
2. Create a new task in the scheduled tasks.
3. Enter a "Name" for the task under "Properties > Properties > General".
4. Select the "Print job" setting in the "Type" field.
5. Select the required "print job".
6. Under "Start time", specify the time and frequency of report output, for example, daily at 5 PM:

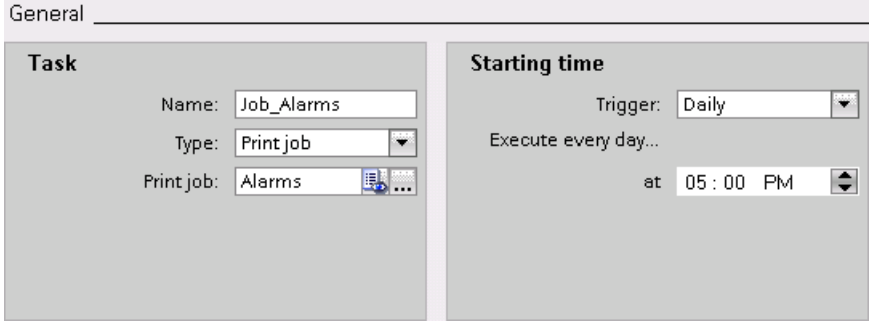

Result

The task for time-driven report output was created successfully.

See also

Principles for report output (Page 1253)
Create a print job (Page 1254)
Printing reports (Page 1277)

7.2.2.3 System reports and system print jobs (RT Professional)

Basics for system traces and system print jobs (RT Professional)

System reports and system print jobs for the print function of screen objects

Certain screen objects, e.g. in the alarm view, contain a default button function (printer icon) for the report output.

The "Print" button of the screen object is linked to a preconfigured system print job. The system print job, in turn, outputs a preconfigured system report.

System reports output a "snapshot", or "all data" of the screen object.

- Snapshot: The report outputs the data currently displayed in the screen object.
- All data: The report outputs all data content of the screen object.

The following table identifies the preconfigured system reports and system print jobs for the screen objects.

Screen object	Default	Print scope	System print job	System reports
f(t) trend view	X	Snapshot	Function Trend Control - Picture	@Function Trend Control - Picture
f(x) trend view	X	Snapshot	Online Trend Control - Picture	@Online Trend Control - Picture
Table view		Snapshot	Online Table Control – Picture	@Online Table Control – Picture
	X	All data	Online Table Control - Table	@Online Table Control - Table
Alarm view		Snapshot	Alarm Control – Picture	@Alarm Control – Picture
	X	All data	Alarm Control - Table	@Alarm Control - Table
Recipe view		Snapshot	Recipe Control - Picture	@Recipe Control - Picture
	X	All data	Recipe Control - Table	@Recipe Control - Table
Value table		Snapshot	Ruler Control - Picture	@Ruler Control - Picture
	X	All data	Ruler Control - Table	@Ruler Control - Table

Usually, there is no need to configure the report output. Customize the system print job or system report to suit your requirements: You can also choose to assign a screen object a different, preconfigured system print job, or a print job you created for the output of a report with individual design.

System print reports and system print job for alarm sequence reports

The output of an alarm sequence report is configured in a system print job. If you have activated "Properties > Properties > General > Use line report", WinCC uses the alarm sequence report with line layout. If you have not activated this option, WinCC uses the alarm sequence report with page layout.

Layout	System print job	System report
Page layout	Alarm sequence report	@AlarmSequenceReportForPagePrinter
Line layout	Alarm sequence report	@AlarmSequenceReportForLinePrinter

See also

Continuous alarm reporting (Page 1243)

Principles for report output (Page 1253)

Changing system reports and system print jobs (RT Professional)

Displaying system reports and system print jobs in the project navigation

The system reports and system print jobs are stored in the project template and not displayed automatically in the project navigation. If you want to modify a system report or system print job, display these in the project navigation.

To do this, follow these steps:

- In the project navigation select the command "Add system report" in the shortcut menu of the "Reports" folder.

The project navigation also shows the system reports under "Reports".

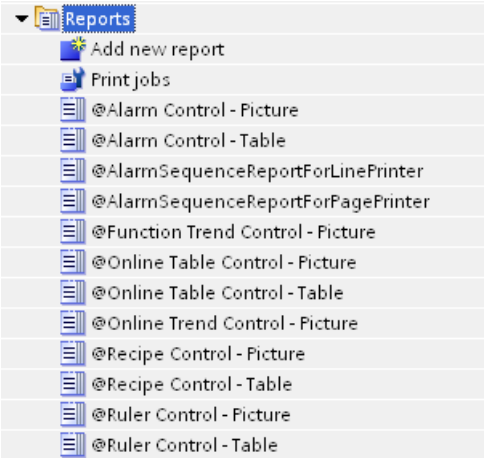

The list of "Print jobs" was extended to include system print jobs.

Print jobs

Name	Report	First printer	Comment
Ruler Control - Table	@Ruler Control - Table		
Ruler Control - Picture	@Ruler Control - Picture		
Recipe Control - Table	@Recipe Control - Table		
Recipe Control - Picture	@Recipe Control - Picture		
Online Trend Control - Picture	@Online Trend Control - Picture		
Online Table Control - Table	@Online Table Control - Table		
Online Table Control - Picture	@Online Table Control - Picture		
Function Trend Control - Picture	@Function Trend Control - Picture		
Alarm sequence report	@AlarmSequenceReportForLinePrinter	LPT1:	
Alarm Control - Table	@Alarm Control - Table		
Alarm Control - Picture	@Alarm Control - Picture		
<Add new>			

Changing system print jobs and system reports

You can modify system print jobs and system reports similar to "normal" print jobs and report, however, with the following restrictions:

- For system print jobs, you cannot rename the report or change its default assignment.
- You cannot add detail pages to system reports.

Validity of changes

In the project, any modification to system reports and system print jobs affect all objects assigned the system report or system print job.

Note

Cancel changes

To restore the preconfigured status of WinCC, for example, after the control for output of data was accidentally deleted in the report, first delete the system report or the system print job and then select the "Add system reports" command in the shortcut menu of the "Reports" folder in the project tree.

Deleting unnecessary system reports and system print jobs

To only show reports and print jobs used in the project (e. g. after migration), you can delete the system reports and system print jobs using the shortcut menu.

If the deleted report or the deleted print job is used in the project, the system report and the system print job preconfigured in WinCC are used.

To display all system reports and system print jobs once again, select the command "Add system reports" in the shortcut menu of the "Reports" folder in the project tree.

- Any deleted system reports and system print jobs will be shown once again. The preconfigured WinCC status is restored. The status at the time of deletion is not reconstructed.
- All other system reports and system print jobs remain the same.

See also

Basics for system traces and system print jobs (Page 1259)

Creating an individual layout for the report output of a screen object (RT Professional)

Introduction

If you do not want to use the default system print job (including the system report) that is linked to the screen object for report output, assign this screen object a different system print job, or a report with individual layout.

Using a different system print job

Proceed as follows to assign a different system print job to the screen object:

1. Select the screen object.
2. Select the system print job under "Properties > Properties > General > Print".

Using a report with individual layout

Proceed as follows to use a report with individual layout:

1. Create the report.
2. Define the content of the detail page.
   - To output a snapshot of the data currently *displayed in the screen object*, insert the "Control Hardcopy" report object.
   - To output all report data in table form that are *currently contained in the screen object*, insert the "Control Data" report object.
   - To include the data source in the report, e.g. window title, name of the object or screen, insert the "Control Screen" report object.
3. Create a print job and assign it to the report.
4. Select the screen object and then select the print job under "Properties > Properties > General > Print".

7.2.3 Working with objects (RT Professional, RT Advanced, Panels, Comfort Panels)

7.2.3.1 Inserting an object (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In the "Screens" or "Reports" editor, insert the objects to the "Toolbox" task card. Use the mouse to drag the objects into the work area. You either keep the objects in their original size, or scale them up or down when you paste them.

In addition, you can copy or move objects via the clipboard from one editor to another, for example to transfer a screen object to a report. Alternatively, you can also use the mouse instead of the clipboard for copying and moving:

- Copy: <Ctrl + Drag&Drop>
- Moving: Drag&drop

Note

Basic Panels

The "Reports" editor is not available for Basic Panels.

Requirement

The "Tools" task card is open.

Inserting objects in their standard size

1. In the "Toolbox" task card, select the desired graphic object or the desired graphic in the WinCC graphics folder.
   When you move the cursor across the work area, it turns into a crosshair with an appended object icon.
2. Click the location in the work area where you want to insert the object or graphic.
   The object is inserted with its standard size at the desired position in the work area.

Alternatively, double-click the object in the "Toolbox" task card.

Copying an object

1. Select the desired object.
2. Select "Copy" in the shortcut menu.
3. Click the desired location and select "Paste" in the shortcut menu.

WinCC inserts a copy of the object at the desired location. You can only change the properties that are appropriate for the relevant context.

Example: In the "Screens" editor, you can set the mode for input and output for I/O fields. In the "Reports" editor, the mode is set to "Output".

Original and copy are not interconnected and are configured independently from one another.

Inserting lines

1. Select the desired graphic object in the "Tools" task card.
2. Click on a location in the work area. A line in the standard size is inserted.

Inserting a polygon or polyline

1. Select the desired object "Polyline" or "Polygon" in the "Tools" task card.
2. Click on a location in the work area. This fixes the starting point of the object.
3. Click another location in the work area. A corner point is set.
4. For every additional corner point, click on the corresponding location in the work area.
5. Double-click on a location in the work area. The last corner point is set.
   This fixes all the points of the polygon or polyline.

   Note
   Basic Panels
   The "Polyline" and "Polygon" objects are not available for Basic Panels.

Note

If you want to insert several objects of the same type, use the "Stamp" function. This avoids having to reselect the object in the "Tools" task card every time before inserting it. To do so, select the icon in the toolbar of the "Tools" task card.

7.2.3.2 Deleting an object (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can delete objects individually or with a multiple selection.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to delete.
   To delete multiple objects, keep the <Shift> key pressed and select the objects to be deleted one after the other. Alternatively, drag and maximize an area around the desired objects with the mouse.
2. Select "Delete" from the shortcut menu.

Result

The selected objects are deleted.

7.2.3.3 Inserting several objects of the same type (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

WinCC offers the possibility to "stamp" several objects of the same type directly one after the other, i.e. paste without having to reselect the object every time. In addition you have the possibility of multiplying an object that has already been inserted.

Requirement

The "Tools" task card is open.

Inserting several objects of one type

1. Select the object that you want to insert in the "Tools" task card.
2. Click the icon in the toolbar of the "Tools" task card.
   The "Stamp" function is activated.
3. To insert the object with its standard size, click the relevant insertion position in the work area.
   To insert the object with another size, position the mouse pointer at the desired location in the work area. Press the left mouse button and drag the object to the required size.
   The object is inserted in the work area as soon as you release the mouse button.
4. Repeat step 3 to insert further objects of the same type.
5. Click the icon again.
   The "Stamp" function is deactivated.

Note

You can copy existing objects using the drag-and-drop +<CTRL> function. The existing object is not moved in this case. You paste a copy of this object into the new position instead.

Inserting and multiplying an object

1. Insert the desired object from the "Tools" task card.
2. Press the <Ctrl> key and position the cursor on one of the handles displayed in the figure shown below.

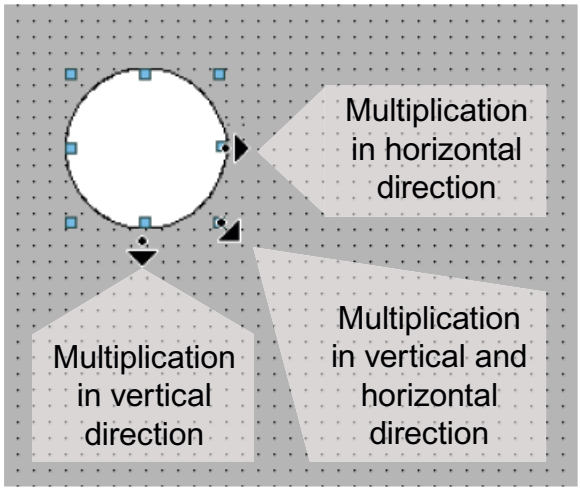

3. Drag the handles to the right and/or down while keeping the left mouse button pressed.
4. The object is multiplied depending on available space if you keep moving the cursor.

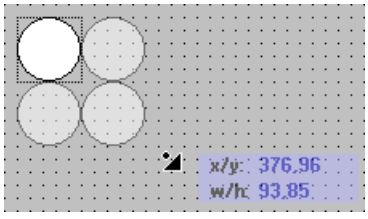

Result

You have pasted and stamped an object in a screen.

7.2.3.4 Positioning objects (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

When you select an object, it is enclosed by a rectangle with resizing handles. This rectangle is the selection rectangle. The position of an object is defined by the coordinates of the top left corner of the rectangle surrounding the object.

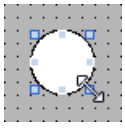

Note

If the position is outside the work area the object is not displayed in Runtime.

Position and align

You have the possibility of having a grid displayed in the work area. You have three options for easier positioning of objects:

- "Snap to grid" When you resposition objects, they are automatically snapped and pasted to the grid. If you hold down the <Alt> key, the object is no longer snapped to the grid.
- "Snap to objects" When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.
- "None": You position the objects in any position.

You activate and deactivate the grid and options as follows:

- In the "Options > Settings > Visualization > Screens" menu
- In the "Layout > Grid" task card

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object you want to move.
   The selected object is framed by a rectangle with resizing handles.

   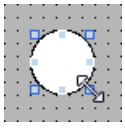

2. Left-click the object and keep the mouse button pressed.
3. Move the mouse pointer onto the new position.
   The contour of the object moves with the mouse and displays the new position for the object.

   The object initially remains at its original position.
4. Now release the mouse button.
   The object is moved into the position indicated by the contour of the selection rectangle.

Alternative procedure

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the X and Y values for the position under "Position & Size".

Result

The object appears at its new position.

7.2.3.5 Moving an object forward or backward (Basic, Advanced, Professional) (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can use the "Order" functions in the shortcut menu of a selected object or in the toolbar to move a selected object in front of or behind other objects within an object layer.

Note

ActiveX controls are always positioned in front of an object layer (.NET property).

Requirement

You have opened a screen which contains a layer with multiple objects.

Procedure

1. Select the object you want to move forward or backward.
2. Select the "Sort" command and one of the following commands from the shortcut menu:

Icon	Description
	Moves the selected object before all the other objects of the same layer
	Moves the selected object to the lowest position in the same layer
	Moves the selected object up by one position
	Moves the selected object down by one position

Alternative procedure

1. Open the "Layers" palette of the "Layout" task card.
2. Navigate to the required object.
3. Hold down the mouse button, and drag the object in the tree topology to the required position in the layer.
4. Now release the mouse button.

Result

The object is moved up or down.

7.2.3.6 Resizing objects (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

When you select an object, it is enclosed by a rectangle with handles. You have the following options of resizing an object:

- Drag the handles using the mouse.
- Modify the "Size" property in the Inspector window.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object you want to resize.
   The selection rectangle appears. The following figure shows a selected object:

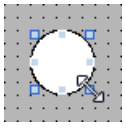

2. Drag a resizing contact of the rectangle to a new position.
   The size of the object changes.
   - The size of the object is aligned to the grid pattern, provided the "Snap to grid" function is set.
   - Press <ALT> to disable this function while you drag the object.
     In order to scale the object proportionally, keep the <Shift> key pressed while changing the size with the mouse.

Alternative procedure

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the size of the object under "Position & Size".

Harmonizing the object size

1. Select the objects.
2. Now, click one of the following buttons: or or
   The size of the selected objects is matched to each other.

The following screen shows how the selected objects are adapted to the height of the reference object:

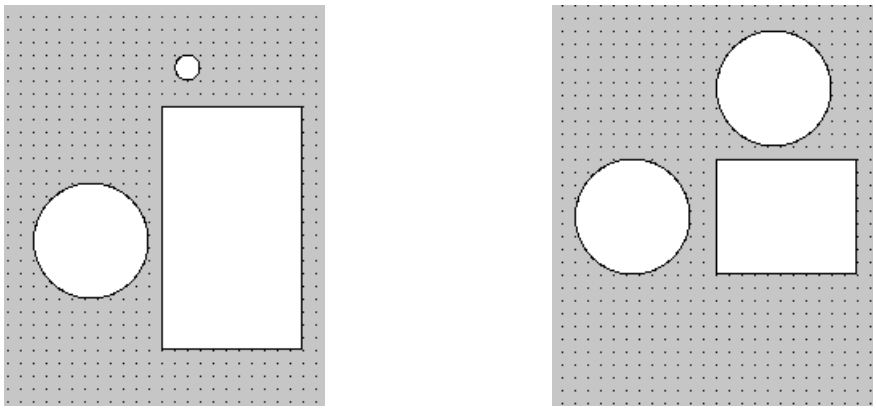

Icon	Description
	Aligns the selected objects to the width of the reference object.
	Aligns the selected objects to the height of the reference object.
	Aligns the selected objects to the width and height of the reference object.

Result

The object now appears with its new size.

7.2.3.7 Selecting multiple objects (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Select all objects you want to align with each other or to change global properties. This procedure is called "multiple selection."

The Inspector window shows all the properties of the selected objects.

You now have several options of selecting multiple objects:

- Draw a selection frame around the objects.
- Hold down the <Shift> key, and click the required objects.

Selection frame of a multiple selection

The selection frame surrounds all objects of a multiple selection. The selection frame is comparable with the rectangle that surrounds an individual object.

The selection frame is not visible. When you have made your multiple selection, the following frame is displayed:

- The reference object is indicated by the rectangle around it.
- The other selected objects are indicated by a dashed-line frame.

Specifying a reference object

The reference object is the object upon which the other objects are oriented. The reference object is framed by a rectangle with handles. The following figure shows a reference object with two other selected objects:

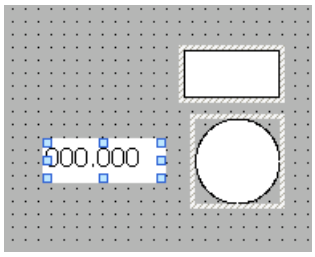

You have the following options to specify the reference object:

- Select the objects via multiple selection. The object selected first is then the reference object.
- Draw a selection frame around the objects. The reference object compiled automatically. If you wish to specify a different object within the selection as the reference object, click on the desired object. This action does not cancel your multiple selection.

Requirement

You have opened the work area containing at least two objects.

Selecting multiple objects with a selection frame

1. Position the mouse pointer in the work area close to one of the objects to be selected.
2. Hold down the mouse button, and draw a selection frame around the objects to be selected.

Or:

1. Hold down the <Shift> key.
2. Click the relevant objects, working in succession.
   All the selected objects are identified by frames.
   The object selected first is identified as reference object.

Note

To remove an object from the multiple selection, press <SHIFT>, hold it down and then click the relevant object once again.

Result

Multiple objects are selected. One of those is identified as the reference object. You can now perform the following steps:

- Changing the object properties of all the objects
- Resizing all the objects by the same ratio, by dragging the selection frame to increase or reduce the size
- Moving all the objects in one group
- Aligning the objects to the reference object

7.2.3.8 Repositioning and resizing multiple objects (RT Professional, RT Advanced, Panels, Comfort Panels)

Possible modifications

After you have selected multiple objects, you edit them:

- Shift using the mouse
  - To change the absolute position of the marked objects, position the mouse pointer over an object, and shift the multiple selection with the mouse button pressed.
  - To resize all the objects by the same ratio, grab the resizing handles of the reference object.
- Move over the work area with the icons of the toolbar
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects
- Moving with the shortcut menu commands of the work area
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects

7.2.3.9 Aligning objects (RT Professional, RT Advanced, Panels, Comfort Panels)

Procedure

1. Select the objects via multiple selection.
2. Specify an object as the reference object.
3. Select the desired command in the toolbar or the shortcut menu - see table below.
   The selected objects will be aligned.

Aligning objects flush

The selected objects will be aligned flush to the reference object.

Icon	Description
	Aligns the selected objects to the left edge of the reference object.
	Aligns the selected objects to the vertical center axis of the reference object.
	Aligns the selected objects to the right edge of the reference object.
	Aligns the selected objects to the upper edge of the reference object.
	Aligns the selected objects to the horizontal center axis of the reference object.
	Aligns the selected objects to the lower edge of the reference object.
	Centers the selected objects to the center points of the reference object.
	Centers the selected objects vertically in the screen.

Snap to object

When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.

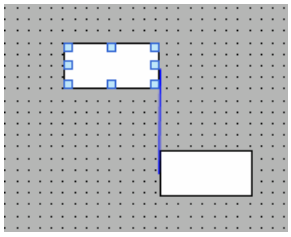

If you are working with the keyboard, press the Alt key. When you move the selected object with the arrow keys, the next anchor point is displayed.

Distributing objects evenly

You need at least three selected objects. A reference object is not required.

1. Select the objects.
2. Click one of the buttons "Distribute horizontally equal" or "Distribute vertically equal".
   The selected objects are distributed at equal distances.

The following screen shows how you align the vertical spacing of the selected objects:

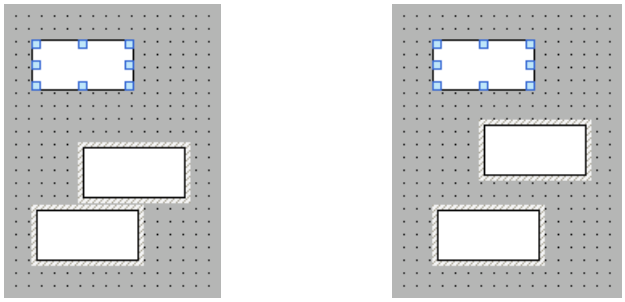

Icon	Description
	Aligns the horizontal distance between the objects. The position of the objects on the extreme left and right side remains unchanged. All other objects are distributed evenly between them.
	Aligns the vertical distance between the objects. The position of the objects at the extreme top and bottom (right and left) remains unchanged. All other objects are distributed evenly between them.

7.2.3.10 Rotating objects (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can rotate a suitable object clockwise or counterclockwise around its center axis in steps of 90°.

Note

Not all the objects can be rotated. Some objects that can be rotated in screens cannot be rotated in reports.

You can also rotate multiple objects using the multiple selection function. Certain WinCC objects such as buttons cannot be rotated.

The alignment of elements in an object will change in a rotated object. The following figure shows how a rectangle and an ellipse behave under the different commands for rotating an object:

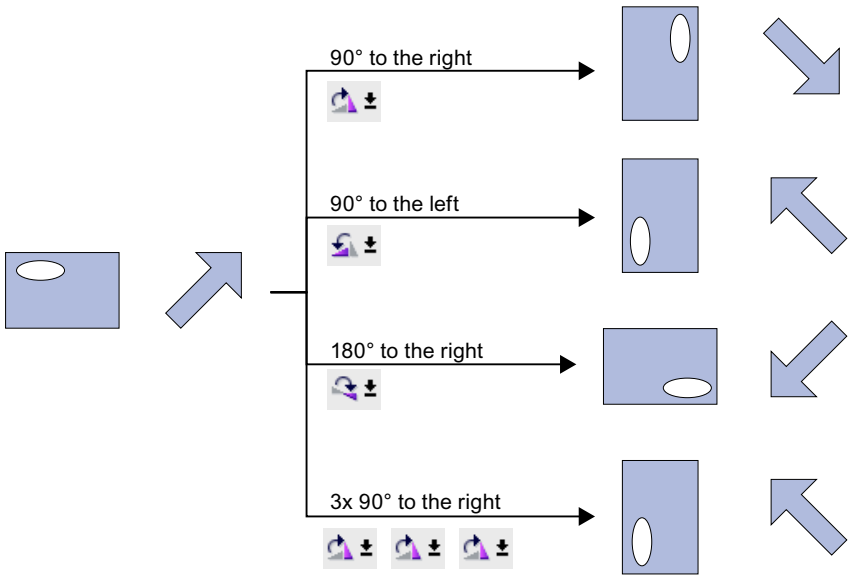

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to rotate.
2. Click one of the following toolbar icons:
   , to rotate the object clockwise around its center point. The angle of rotation is 90°.
   , to rotate the object counterclockwise around its center point. The angle of rotation is 90°.
   , to rotate the object clockwise by 180°.

Result

The object is shown at its new angle.

7.2.3.11 Flipping objects (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can flip an object along its vertical or horizontal center axis. The alignment of elements in an object will change when you flip an object. The following figure shows how a rectangle and an ellipse behave under the different commands for flipping an object.

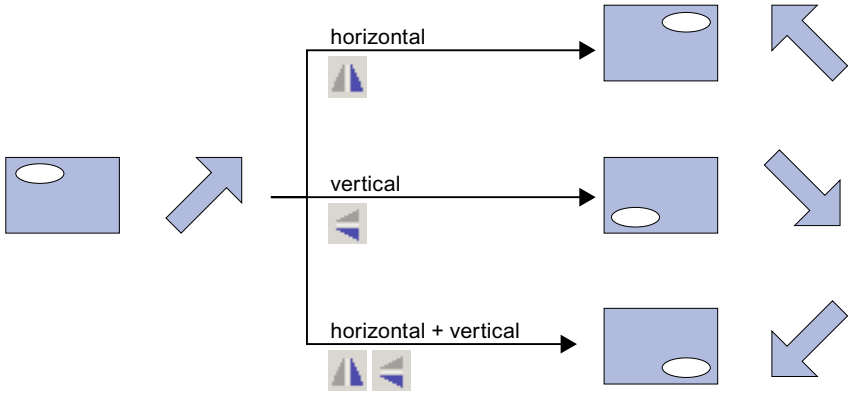

Requirement

You have opened a screen which contains at least one object.

Procedure

1. Select the object that you want to flip.
2. Click the "Flip" command in the shortcut menu and select one of the options displayed:
   – , to flip the selected object along its vertical center axis.
   – , to flip the selected object along its horizontal center axis.

Result

The object is shown at its flipped position.

7.2.4 Operation in Runtime (RT Professional, RT Advanced, Panels, Comfort Panels)

7.2.4.1 Printing reports (RT Advanced, Panels, Comfort Panels)

Printing a report in Runtime

If you have configured an object with the "PrintReport" system function, the operator can print out a report in Runtime. To do this, the operator uses the "Print report" button on a screen. The operator can set or change further settings for the report. The report is output to the default printer.

7.2.4.2 Printing reports (RT Professional)

Starting the report printout in Runtime by the operator

The operator starts printing of a report by using the "Print" button in the following screen objects in Runtime:

- f(x) trend view
- f(t) trend view
- Table view
- Alarm view
- Recipe view

If you have specified in the print job that a dialog is displayed for the selection of a printer or to specify properties, the operator changes these settings in Runtime.

If you have configured a button and linked it to an appropriate script, the operator can start the report printout using this button.

Automatic start of the report printout

The report printout is started automatically in Runtime in the following situations:

- You have linked a print job with a "Print job" type task.
- You have linked a global script for the report printout with a task.
- Under "Services" in the runtime settings of the HMI device, select the item "Scheduled print jobs in Runtime".

Print job/Script diagnostics

All or selected print jobs are shown in the "Print job/Script diagnostics" object depending on the configuration when they are queued in runtime. Depending on the configuration the operator can start the report printout, specify the print options and display a print preview in the "Print job/Script diagnostics" object.

See also

Principles for report output (Page 1253)

Basics for system traces and system print jobs (Page 1259)

7.3 Objects in reports (RT Professional, RT Advanced, Panels, Comfort Panels)

7.3.1 Overview (RT Professional, RT Advanced, Panels, Comfort Panels)

Objects in the "Reports" editor

Different objects are available in the "Reports" editor:

- Objects for formal design of a report,
  for example, "Page number", "Line" or "Circle"
- Objects for output of contents,
  for example, objects for alarm reports or output of recipes

The next section describes report-specific objects with special properties. Simple design objects are not further explained.

7.3.2 Audit report (RT Advanced, Panels, Comfort Panels)

Application

With the audit report, the content of the audit trail of a HMI device is output as a report.

RecordID	Timestamp	DeltaToUTC	UserID
0	18.11.2007 23:00 ObjectID: Application Description: This is an example description. It may not be cut and will be printed out completely.	-1:00	System

Requirements

The "audit report" object is available only if the following requirements are met:

- The HMI device supports a GMP-compliant configuration.
- GMP-compliant configuration is active in the Runtime settings of the HMI device.

Note that operator actions and system operations are only recorded if a corresponding log is configured.

Layout

In the Inspector window the position, shape, style, color and font types of the object are customized. In particular, the following can be adjusted:

- Visibility of the comment

7.3.3 Control Data (RT Professional)

Application

You need report objects if you want to configure a report with individual layout for the report output function of a screen object.

The "Control Data" report object outputs all data contained in the screen object in table form – including those data that are currently invisible in the screen object in Runtime.

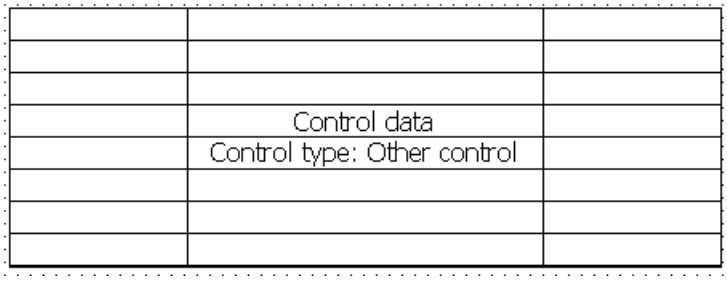

Layout

Customize the frame and color settings in the Inspector window. You can adapt the following properties in particular:

- Control type whose content is output
- Control objects of the screen object to be included in the visualization
- Position & size

Control type

Specify in the Inspector window under "Properties > Properties > General > Control type", if the value at the ruler position, the content of the object or the object is output.

- In the "f(x) trend view" and the "f(t) trend view" select either the "Ruler" or "Other control" option.
- For all other screen objects select "Other control".

Control objects of the screen object

In the Inspector window, select "Properties > Properties > General > Content" to specify whether to include the toolbar, the status bar and the scroll bar in the screen object.

Size & position

In the Inspector window, select "Properties > Properties > Layout" to specify the size and position of the report object.

The table is output in the report in accordance with the width of the report object. Therefore, configure a report object with suitable width.

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.4 Control hard copy (RT Professional)

Application

You need report objects if you want to configure a report with individual layout for the report output function of a screen object.

The "Control Hardcopy" object output a snapshot of the screen object. Only the data that is visible in the screen object in Runtime is output.

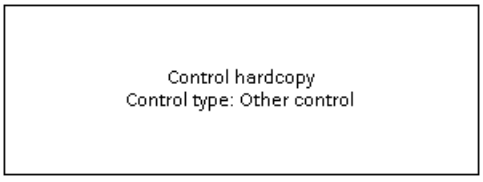

Layout

Customize the frame and color settings in the Inspector window.

You can adapt the following properties in particular:

- Control type whose content is output
- Specifies the control objects of the screen objects to include in the visualization.
- Size & position

Control type

Specify in the Inspector window under "Properties > Properties > General > Control type", if the value at the ruler position, the content of the object or the object is output.

- In the "f(x) trend view" and the "f(t) trend view" select either the "Ruler" or "Other control" option.
- For all other screen objects select "Other control".

Control objects of the screen object

In the Inspector window, select "Properties > Properties > Content" to specify whether to include the toolbar, the status bar and the scroll bar in the screen object.

Size & position

In the Inspector window, select "Properties > Properties > Layout" to specify the size and position of the report object.

In the report, the screen object is resized to fit the frame of the report object. Therefore, configure a report object of a suitable size.

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.5 Control Information (RT Professional)

Application

You need report objects if you want to configure a report with individual layout for the report output function of a screen object.

The "Control Information" report object outputs the window title, the screen object name or the screen name. This way you include, for example, the data source in the report.

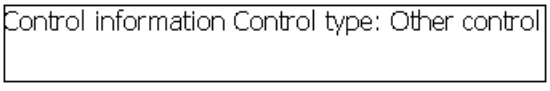

Layout

Customize the frame, color, font, and font size settings in the Inspector window. You can adapt the following properties in particular:

- Control type whose content is output
- Content

Control type

Specify in the Inspector window under "Properties > Properties > General > Control type", if the value at the ruler position, the content of the object or the object is output.

- In the "f(x) trend view" and the "f(t) trend view" select either the "Ruler" or "Other control" option.
- For all other screen objects select "Other control".

Content

In the Inspector window, select "Properties > Properties > General > Content" to specify the information to output with the report object:

- Window title
- Object name
- Screen name

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.6 CSV provider trends (RT Professional)

Application

The "CSV provider trends" object outputs data from a file in the format "*.csv" as trends in the report.

Page breaks are inserted automatically in the report depending on the amount of output data. The configured width is not changed in the report. Subsequent objects are shifted automatically.

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Data source

Data source

Specify the data source under "Properties > Properties > General > Provider". You can select a CSV file, or configure an internal tag for dynamic selection of the CSV file. This means that the data output is changed directly at the HMI device. The tag must be of data type "String" or "WString".

Note

For their visualization in trends, the data in the CSV file must have a defined structure.

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.7 CSV provider tables (RT Professional)

Application

The "CSV provider tables" object outputs data from a CSV file in tabular format in the report.

Page breaks are inserted automatically in the report, depending on the amount of output data. The configured width is not changed in the report. Subsequent objects are shifted automatically.

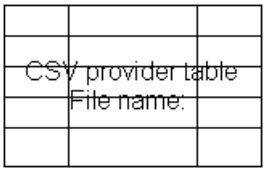

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Data source
- Applying fonts from the data source

Data source

Specify the data source under "Properties > Properties > General > Provider". You can select a CSV file, or configure an internal tag for dynamic selection of the CSV file. This means that the data output is changed directly at the HMI device. The tag must be of data type "String" or "WString".

Note

To enable output in a table, the data in the CSV file must have a defined structure.

Applying fonts from the data source

Under "Properties > Properties > General > Font", specify whether to apply the font and font size from the CSV file for the report output.

If you deactivate the "Font from data source" and "Font size from data source" options, the settings you specified under "Format Text > Style" are used.

See also

CSV file for output of a table (Page 1247)

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.8 Date/time field (RT Advanced, Panels, Comfort Panels)

Application

The "Date/time field" object shows the system time of the HMI device or the date and / or time of a connected tag. The layout depends on the language setting on the HMI device.

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. In the "Properties > Properties > General" dialog, you can adapt the following properties in particular:

- Long date/time format: This setting defines the format displayed for the data and / or time.
- System time: Specifies whether to use the system time of the HMI device, or the data and / or time of a connected tag.

Long date/time format

Option	Description
"Enabled"	Date and / or time is fully displayed, for example "Sunday, December 31, 2000 10:59:59 AM"
"Disabled"	Date and / or time is displayed in abbreviated form, e.g. "12/31/2000 10:59:59 AM"

System time

Option	Description
"Enabled"	The system time of the HMI device is displayed
"Disabled"	The date and / or time of the connected tag is displayed Select a tag of the "DateTime" data type, e.g. an internal tag. Information about data types which are available for connection to other PLCs can be found in the documentation about the respective communication drivers.

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.9 Date/time field (RT Professional)

Application

The "Date/time field" object outputs the date and / or time in an individually configured format in the report.

12/31/2000 10:59:59 AM

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Display format of date and / or time

Display format of date and time

Specify the date and / or time format under "Properties > Properties > General > Format". You can combine wildcards and any text.

The following table shows which wildcards you can use:

Wildcards	Implementation	Example
%a	Weekday abbreviated	Sun
%A	Weekday in full form	Sunday
%b	Month abbreviated	Dec
%B	Month in full form	December
%c	Date and time in standard format	12/31/2000 10:59:59 AM
%d	Day as decimal value (01 to 31)	31
%H	Hours in 24 hour format (00 to 23)	22
%I	Hours in 12 hour format (01 to 12)	10
%j	Day of the year as decimal value (000 of 366)	366
%m	Month as decimal value (01 to 12)	12
%M	Minute as decimal value (00 to 59)	59
%p	PM/AM notation for the 12 hour format	AM
%S	Second as decimal value (00 to 59)	59
%U	Week of year as decimal value (01 to 51)	50
%x	Date in standard format	12/31/2000
%X	Time in standard format	10:59 AM
%y	Year without century as decimal value (00 to 99)	00
%Y	Year with century as decimal value	2000
%z	Time zone as name	W. European Standard Time
%%	Percent character	%

Examples

The following table shows a few examples of the combination of wildcards and text:

Format	Output example
Date: %x time: %H:%M	Date: 12/31/2000 time: 22:59
Printout of %a %c	Printout of Sun 12/06/2010 10:59:59 AM

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.10 I/O field (RT Professional, RT Advanced, Panels, Comfort Panels)

Application

The "I/O field" object is used to enter process values.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Display format: Specifies the format in which the values in the I/O field are output.

Note

The "I/O field" object is also available in "Screens" editor. In reports, the object only outputs data. Accordingly only configure the output of data for application in reports.

Display format

The "display format" for the output of values is specified in "Properties > Properties > General > Format", in the Inspector window.

Layout	
"Binary"	Output of values in binary form
"Date" *	Output of date specifications. The format depends on the language setting on the HMI device.
"Date/time" *	Output of date and time specifications. The format depends on the language setting on the HMI device.
"Decimal" *	Output of values in decimal form.
"Hexadecimal"	Output of values in hexadecimal form.
"Time" *	Output of time specifications. The format depends on the language setting on the HMI device.
"Character string"	Output of strings.
*: not in Runtime Professional	

Avoid overlaps with output fields

If several I/O fields are configured as output fields with a transparent background in a report, these I/O fields may overlap. The transparent part of the one field covers the digits of the other field. This may cause display problems in the report. In order to avoid such overlaps, set the margins of the I/O fields to zero in the object properties under "Properties > Properties > Appearance". Activate "Properties > Properties > Layout > Fit object to contents."

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.11 f(t) trend view in reports (RT Professional)

Application

You use the "f(t) trend view" object to visualize logging tag values in the form of trends as a function of time. The report displays the status that the trend profile had at the time of the start of the print job.

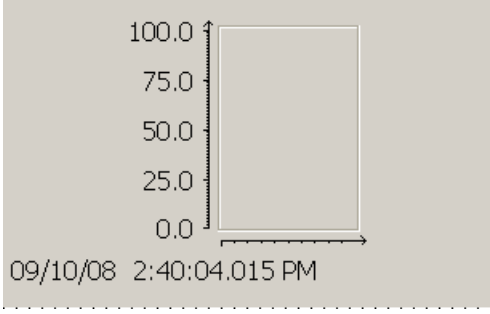

Note

Reduced functional scope of the output in reports

The configuration of the object in the "Reports" editor differs only slightly compared to its configuration in the "Screens" editor.

The functional scope of the object is adapted for the output in reports: In the "Reports" editor, for example, it is not possible to configure operating elements in Runtime.

Special features for the output in reports

In contrast to the configuration in the screen, you can dynamize the properties of each trend with tags, for example, the display and the data supply. This way you can, for example, use a report several times.

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.12 f(x) trend view in reports (RT Professional)

Application

You use the "f(x) trend view" object to represent the values of a tag as a function of another tag. This means that you can present temperature trends as a function of the pressure, for example. The report displays the status that the trend profile had at the time of the start of the print job.

You can also compare the trend to a setpoint trend.

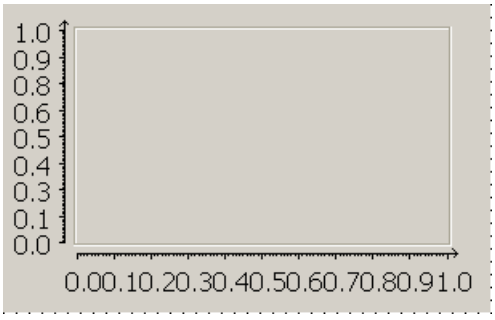

Note

Reduced functional scope for the output in reports

The configuration of the object in the "Reports" editor differs only slightly compared to its configuration in the "Screens" editor.

The functional scope of the object is adapted for the output in reports: In the "Reports" editor, for example, it is not possible to configure operating elements in Runtime.

Special features for the output in reports

In contrast to the configuration in the screen, you can dynamize the properties of each trend with tags, for example, the display and the data supply. This way you can, for example, use a report several times.

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.13 Graphic view (RT Professional, RT Advanced, Panels, Comfort Panels)

Application

The "Graphic view" object is used to display graphics.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Graphic: Specifies the graphic file that is displayed in the object.
- Adjust graphic: Specifies the automatic size stretching for objects with graphics.

Inserting graphics

Use the following graphic formats in the "Graphic view" object: *.bmp, *.tif, *.png, *.ico, *.emf, *.wmf, *.gif, *.svg, *.jpg or *.jpeg. You may also use graphics as OLE objects in the Graphic view .

1. Click "Properties > Properties > General" in the Inspector window.
2. Select the graphic that you wish to insert.
   The graphic preview is shown in the right pane.
3. Click "Apply" to insert the graphic in the Graphic view .

Adjust graphic

Whether a graphic displayed in a Graphic view is stretched to the size of the Graphic view in runtime is specified in the Inspector window.

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Select one of the following options from the "Fit to size" area:
   - Do not adapt automatically
   - Fit graphic to object size
   - Fit object size to graphic
   - Adjust object size to graphic

7.3.14 Graphic I/O field (RT Advanced, Panels, Comfort Panels)

Application

With the "Graphic I/O field" object, the graphics of a graphics list are displayed, depending on the tag value. In the Inspector window, the tag and graphics list are configured, under "Properties > Properties > General":

Note

The "Graphic I/O field" object is also available in "Screens" editor. In reports, the object only outputs data. Accordingly only configure the output of data for application in reports.

Layout

Customize settings for color, frame, position and object size in the inspector window. In particular, the "Fit to size" is defined by specifying whether the object is adjusted to the graphics during output in the report or the graphic is adjusted in the object.

See also

Create a report (Page 1238)

7.3.15 Hardcopy (RT Professional)

Application

Use the "Hardcopy" object to output a snapshot of the screen content in the report.

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Specifying the range

Specifying the range

Under "Properties > Properties > Range", specify the screen content to output in the report:

- "Current window (1)": Active window
- "Selected section (2)": Selected section
- "Entire screen (0)": Complete content

You can assign a tag of the "Number" type for dynamic range selection. The settings are made as follows:

- 0 = "Entire screen"
- 1 = "Current window"
- 2 = "Selected section"

Specify the following properties for a "Selected section (0)":

- X position of the upper left point of the area
- Y position of the upper left point of the area
- Width in pixels
- Height in pixels

You can assign a dynamic tag to each property.

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.16 Alarm view in reports (RT Professional)

Application

The "Alarm view" object displays alarms that occur during the process in a plant. You also use the alarm view to visualize alarms in list format. WinCC offers various views, such as "Current alarms" or "Historical alarm list" (short-term).

The report displays the status that the alarm view had at the time of the start of the print job.

Date	Time	Numbe	Event text
27/08/10	14:51:42.000	89	TEXT
27/08/10	14:51:43.000	90	TEXT
27/08/10	14:51:44.000	91	TEXT
27/08/10	14:51:45.000	92	TEXT
27/08/10	14:51:46.000	93	TEXT
27/08/10	14:51:47.000	94	TEXT
27/08/10	14:51:48.000	95	TEXT
27/08/10	14:51:49.000	96	TEXT
27/08/10	14:51:50.000	97	TEXT
27/08/10	14:51:51.000	98	TEXT
27/08/10	14:51:52.000	99	TEXT
27/08/10	14:51:53.000	100	TEXT

Note

Reduced functional scope of the output in reports

The configuration of the object in the "Reports" editor differs only slightly compared to its configuration in the "Screens" editor.

The functional scope of the object is adapted for the output in reports: In the "Reports" editor, for example, it is not possible to configure operating elements in Runtime.

See also

Report design tools (Page 1240)

Continuous alarm reporting (Page 1243)

Create a report (Page 1238)

7.3.17 Alarm sequence report (RT Professional)

Application

The "Alarm sequence report" object outputs all alarms from the HMI device as of the start of Runtime continuously in chronological order. It is only used in the "@AlarmSequenceReportForPage" system report that specified the output of the WinCC alarm sequence report in page layout.

You specify which alarms are output in the object properties.

Page breaks are inserted automatically in the report, depending on the amount of output data. The configured width is not changed in the report. Subsequent objects are shifted automatically.

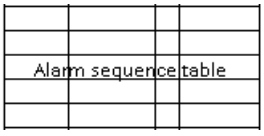

See also

Report design tools (Page 1240)

Continuous alarm reporting (Page 1243)

Create a report (Page 1238)

Changing system reports and system print jobs (Page 1260)

7.3.18 Alarm report (RT Advanced, Panels, Comfort Panels)

Application

Use the "Alarm report" object to output alarms of the selected alarm classes from the alarm buffer or alarm log to the report.

No.	Time	Status	Date		GR		PLC
0	12:00:00 PM normal kursiv normal	KGQ	1/1/1999	Class name	0	*	Device
1	12:00:00 PM normal blinken normal	KGQ	1/1/1999	Class name	1	*	Device
2	12:00:00 PM Message Text	KGQ	1/1/1999	Class name	2	*	Device
3	12:00:00 PM normal bold normal	KGQ	1/1/1999	Class name	3	*	Device
4	12:00:00 PM normal underline normal	KGQ	1/1/1999	Class name	4	*	Device

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Source
- Alarm classes
- Time range
- Additional settings for the display
- Visible columns

Specifying the source

Under "Properties > Properties > General > Settings > Source", specify whether to display alarms from the alarm buffer, or from the alarm log. Select the alarm log in the "Alarm log" dialog.

Note

The "Alarm log" source is only available if the HMI device supports logs.

Specifying alarm classes

Under "Properties > Properties > General > Alarm classes, enable the alarm classes for the alarms output to the report.

Specifying the time range

If you want to restrict alarm output to a specific time range, select the start and end tags for the time range in the "Properties > Properties > General > Time range" dialog. The tag must be of the "DateTime" type.

Additional settings for the display

Under "Properties > Properties > General > Settings, specify the following settings for the display in the alarm report:

- Under "Sorting", specify whether to start the display with the oldest or most recent alarm.
- Under "Lines per entry", specify the number of lines to be available for each alarm. The required number of lines depends on the following factors:
  - Number and width of the selected columns for the output
  - Font size used
  - Paper format of the printer
- Select "Visible heading" to enable the display of column headers.
- Select "Display milliseconds" to enable the display of milliseconds for time data.

Visible columns

Under ""Properties > Properties > Layout > Visible columns", enable the columns to be displayed in the alarm report.

See also

Configuring reporting of alarms (Page 873)

Create a report (Page 1238)

7.3.19 ODBC database field (RT Professional)

Application

Using the "ODBC database field" object, you can output data from a single table row in the report via the ODBC interface. The ODBC interface is used for the consistent output of data that does not originate from WinCC in the report.

Page breaks are inserted automatically in the report, depending on the amount of output data. The configured width is not changed in the report. Subsequent objects are shifted automatically.

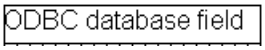

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Settings for the SQL connection
- Data selection for the SQL statement

Text format

Specify the data source under "Properties > Properties > Text format > Alignment". You have the options of displaying the text as horizontal text on the left or right or centered.

Note

The "Alignment > Vertical" setting has no effect in Runtime.

Connection settings

Under "Properties > Properties > SQL connection", specify the data source, user name, and password for the SQL connection. You can assign dynamic tag to all settings. This means that the data output is changed directly at the HMI device.

Data selection

Under "Properties > Properties > SQL statement", specify the data area to visualize in the report. Enter "Select-Statement".

Enter the required "Select-Statement" in the text field of the same name or select the "Tag" that supplies the Select-Statement. The tag must be of the "String" or "WString" type.

You can dynamize the Select-Statement in the input field of the same name with tags. To do so you select "Insert reportfield" at the required text passage in the shortcut menu and then select the required tag.

Note

Enter only "Select-Statements".

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.20 ODBC database table (RT Professional)

Application

Using the "ODBC database table" object, you can output data from a table in the report via the ODBC interface. The ODBC interface is used for the consistent output of data that does not originate from WinCC in the report.

Page breaks are inserted automatically in the report, depending on the amount of output data. The configured width is not changed in the report. Subsequent objects are shifted automatically.

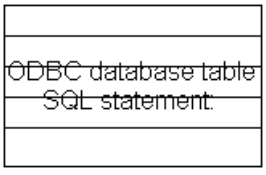

Layout

In the Inspector window, you customize the position, shape, style, colors and font types of the object. You can adapt the following properties in particular:

- Settings for the SQL connection
- Data selection for the SQL statement
- Table layout, e.g. number of columns and circular list

Connection settings

Under "Properties > Properties > SQL connection", specify the data source, user name, and password for the SQL connection. You can assign dynamic tag to all settings. This means that the data output is changed directly at the HMI device.

Data selection

Under "Properties > Properties > SQL statement", specify the data area to visualize in the report.

Enter the required "Select-Statement" in the text field of the same name or select the "Tag" that supplies the Select-Statement. The tag must be of the "String" or "WString" type.

You can dynamize the Select-Statement in the input field of the same name with tags. To do so you select "Insert reportfield" at the required text passage in the shortcut menu and then select the required tag.

Note

Enter only "Select-Statements".

Circular list

In the Inspector window, use the "Circular list" option under "Properties > Properties > Table" to specify the the way the data is distributed to the table columns.

- "Circular list" disabled
  The table rows are filled in descending order.
  A value is output in the first column of a row. The other columns will stay empty.
- "Circular list" is enabled
  The table rows are filled in descending order.
  All columns of a row are filled left to right.

  Note

  This option is only effective if the data are supplied in one column and if the configured table has several columns.

Example:

You configure a table with three columns. At the time of report output, the screen object contains the following data: A1, A2, A3, B1, B2, B3, C1, C3, D4

The following table shows the result with deactivated "Circular list" option:

Column 1	Column 2	Column 3
A1		
A2		
A3		
B1		
B2		
...		

The following table shows the result with activated "Circular list" option:

Column 1	Column 2	Column 3
A1	A2	A3
B1	B2	B3
C1	C3	D4

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.21 Project name (RT Professional)

Application

Use the "Project name" object to output the name of the current project in the report.

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Format of the project name

Format of the project name

In the "Format" field under "Properties > Text format > Style", specify the output format for the project name. You can use a placeholder for the project name and enter any text, if necessary.

The following table shows which wildcards you can use:

Wildcards	Value
"%R"	Path and project name
"%r"	Project name (without path information)

Examples

The following table shows a few examples of the combination of wildcards and text:

Format	Output example
Project: %R	Project: D:\WinCC\Projects\MP277_Mixing
Created for project "%r"	Created for project "MP277_Mixing"

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.22 Report name (RT Professional)

Application

Use the "Report name" object to output the report name in the report.

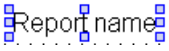

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Format of the report name

Format of the report name

In the "Format" field under "Properties > Text format > Style", specify the output format for the report name. You can use a placeholder for the report name and enter any text, if necessary.

The following table shows which wildcards you can use:

Wildcards	Value
"%L" or "%l"	Report name

Example

The following table shows a few examples of the combination of wildcards and text:

Format	Output example
Report: %L	Report: Alarms

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.23 Recipe view in reports (RT Professional)

Application

Use the "Recipe view" object to display recipe elements in tabular format. The report displays the status that the recipe view had at the time of the start of the print job.

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

Note

Reduced functional scope of the output in reports

The configuration of the object in the "Reports" editor differs only slightly compared to its configuration in the "Screens" editor.

The functional scope of the object is adapted for the output in reports: In the "Reports" editor, for example, it is not possible to configure operating elements in Runtime.

See also

Report design tools (Page 1240)

Definition and applications (Page 1160)

Create a report (Page 1238)

7.3.24 Recipe report (RT Advanced, Panels, Comfort Panels)

application

Use the "Recipe report" object to output the elements of recipe data records in the report.

Recipe Number: 1 Name: Recipe_1

Data record Number: 1 Name: Data record_1

Tag:	Name:	Type:	Value:
Variable_Number_0	Entry_Number_01	ULONG	689874030
Variable_Number_0	Entry_Number_02	LONG	-99253259
Variable_Number_0	Entry_Number_01	SHORT	-9
Variable_Number_0	Entry_Number_03	USHORT	8172
Variable_Flag_05	Entry_Flag_05	BOOL	false
Variable_String_06	Entry_String_06	STRING	WinCC RT
Variable_Date_07	Entry_Date_07	DATETIME	1/1/1999 12:00:00 AM
Variable_Number_0	Entry_Number_08	LONG	6968192
Variable_Number_0	Entry_Number_09	ULONG	9903155
Variable_Flag_10	Entry_Flag_10	BOOL	true

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Recipe
- Data record
- Format
- Visible entries

Selecting a recipe

Specify the recipes to be output in the recipe report under "Properties > Properties > General > Recipe". You can define the recipes based on their recipe name, or on a range of recipe numbers. Under "First recipe" and "Last recipe", enter a value or select a tag.

You can also choose to display all recipes.

Selecting a data record

Under "Properties > Properties > General > Data record", specify which data records of the selected recipes are to be output in the recipe report. You can define the data records based on the data record name, or on a range of data record numbers. Under "First recipe" and "Last recipe", enter a value or select a tag.

You can also choose to display all data records of the selected recipes.

Format

In the "Format" field of the "Properties > Properties > Layout > Settings" dialog, specify whether to output the data records as a table in column format or line report format.

WinCC updates the preview in the detail page accordingly.

Visible entries

Under ""Properties > Properties > Layout > Visible entries", enable the columns to be displayed in the recipe report. Select "Show headings" to enable the display of column headers.

See also

Definition and applications (Page 1160)

Create a report (Page 1238)

7.3.25 Page number (RT Advanced, Panels, Comfort Panels)

application

Use the "Page number" object to output the current page number in the report.

Layout

In the inspector window, change the settings for color, font, position and object size. In particular, specify "Size adaptation":

Fit to size

Under "Properties > Properties > Layout > Fit to size", the "Fit object to content" option is used to specify whether WinCC adjusts to the object size of the field content:

- Enabled "Fit object to contents" option:
  WinCC automatically adjusts the size of the object to the configured format. The font size and field length are defined under "Properties > Properties > General > Text".
  The object can be moved in the working area, however, the size cannot be changed. During report output, the entire field content is output at the time of report output.
- Disabled "Fit object to contents" option:
  Customize the field size by yourself. WinCC does not resize the field for report output. It can be possible that not all content of the field is output in the report. Therefore, configure a field of sufficient size.

See also

Create a report (Page 1238)

7.3.26 Page number (RT Professional)

Application

The "Page number" object outputs the current page number or the total number of pages in the report in the header or footer of the report.

Layout

In the Inspector window, change the settings for color, font, frame, alignment as well as position and object size. In particular you set the format of the page number.

Format

Under "Properties > General > Text > Format", specify the format of the page numbers to be displayed. You can use a wildcard for the page numbers and enter a user-specific text.

The following table shows which wildcards you can use:

Wildcards	Value
"%N", or "%n"	Current page number
"%T", or "%t"	Total number of pages in the report

Examples

The following table shows a few examples of the combination of wildcards and text:

Format	Output example
Page %n of %t	Page 2 of 10
%n/%t	2/10

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.27 Symbolic I/O field (RT Advanced, Panels, Comfort Panels)

application

With the "Symbolic I/O field" object, the content of a text list in reports is displayed, depending on the tag value. In the inspector window, the tag and text list is configured under "Properties > Properties > General":

Layout

In the inspector window, change the settings for color, frame, font, position and object size. In particular, the "Fit to size" is defined, by specifying whether the object size is adjusted to the text during output in the report or not.

Note

The "Symbolic I/O field" object is also available in "Screens" editor. In reports, the object only outputs data. Accordingly only configure the output of data for application in reports.

See also

Create a report (Page 1238)

7.3.28 Table view in reports (RT Professional)

Application

You use the "Table view" object to display the process data in a table. Each row of the table represents the status of the selected process tags at a specific time.

The report displays the status that the table had at the time of the start of the print job.

TimeColumn	ValueColumn

Note

Reduced functional scope of the output in reports

The configuration of the object in the "Reports" editor differs only slightly compared to its configuration in the "Screens" editor.

The functional scope of the object is adapted for the output in reports: In the "Reports" editor, for example, it is not possible to configure operating elements in Runtime.

Special features for the output in reports

By contrast to the configuration in the screen, you can assign tags with dynamic properties for the time and value columns. This way you can, for example, use a report several times.

See also

Report design tools (Page 1240)

Create a report (Page 1238)

7.3.29 Text field (RT Professional, RT Advanced, Panels, Comfort Panels)

Application

The "Text field" is a closed object which you can fill with a color.

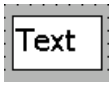

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Text: Specifies the text for the text field.
- Size of text field: Defines whether the size of the object is adapted to the space required by the largest list entry.

Text

Specify the text for the text field in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Enter a text.
   For texts over several lines you can set a line break by pressing the key combination <Shift + Enter>.

Size of text field

In the Inspector window, you can define whether the size of the object is adapted to the space required by the largest list entry.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Activate "Resize > Fit to contents".

Configuring user administration 8

8.1 Basics

8.1.1 Field of application of the user administration (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Principle

The access protection controls access to data and functions in Runtime. This feature protects your applications against unauthorized operation. Safety-related operations are already limited to specific user groups when a project is being created. To this purpose you set up users and user groups that you equip with characteristic access rights, so-called authorizations. You then configure the authorizations required for operation of safety-related objects. Operators only have access, for example, to specific operator controls. Commissioners, for example, have unlimited access in Runtime.

Definition

You administer users, user groups and authorizations centrally in the user administration of WinCC. You transfer users and user groups together with the project to the HMI device. The users and passwords are managed on the HMI device in the User view.

Application example

You configure the "Service" authorization so that only service technicians have access to the configuration parameters. You assign the authorization to the "Service technician" user group. This allows all members of this group to set the protected configuration parameters.

NOTICE
Access protection does not protect against incorrect operations. It is your job to ensure that only authorized personnel with appropriate training will design, commission, operate and maintain plants and machines. Access protection is not suitable for defining work routines and monitoring their observance.

8.1.2 Form of the user administration (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In case of a project in manufacturing engineering, the environment at the equipment manufacturer has to be differentiated from the environment at the end customer as plant operator.

The equipment manufacturer allows users, for example Mr. Foreman, a specific access within the application or HMI device. However, a user Foreman does not exist at the end customer. The machine manufacturer cannot know the end users and the tasks they have to perform for configuration. The final users are usually set after commissioning at the end customer.

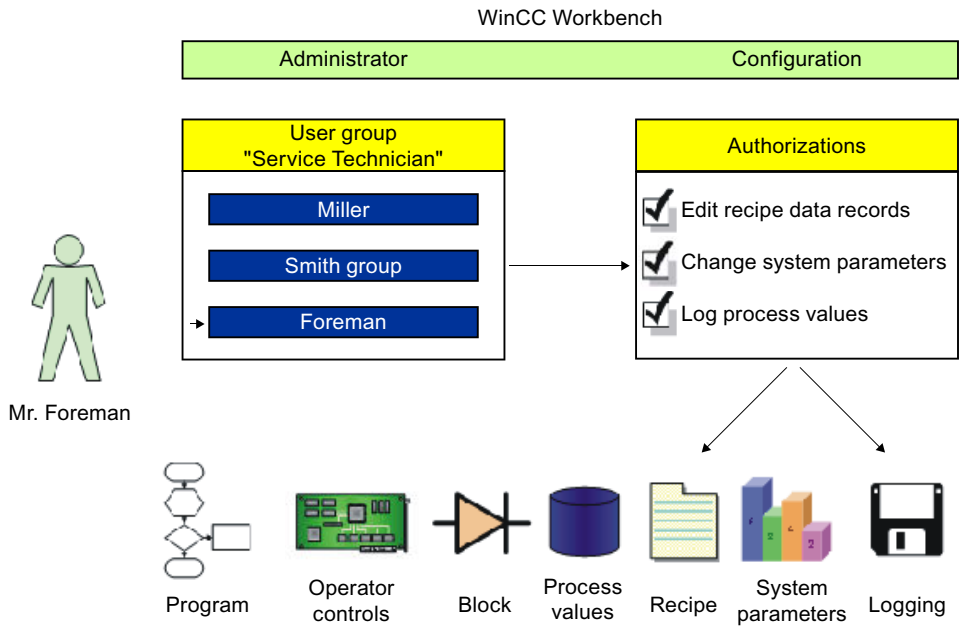

Principle

To minimize the work required for management, authorizations are assigned via user groups and not directly to individual users.

A user group assembles configured authorizations according to common jobs. For example, all permissions required for a service job are collected in a "Service technician" group. When you create a user who should be responsible for servicing, you simply assign him to the "Service technician" group.

The user view enables user administration in Runtime. Use user view to create, delete and assign an authorization to users in Runtime.

The user administration separates the administration of the users from the configuration of the authorizations. This ensures flexibility at the access protection.

Defaults can be set for the user administration during the configuration phase in the Engineering System.

8.1.3 User (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can create users in the "Users" tab of the "User administration" editor and assign them to user groups. The "Users" tab is part of the user administration in WinCC.

Open

To open the "Users" tab, double-click "User administration" in the project window.

Work area

The users are managed in the work area:

- You create or delete users.
- You assign users to user groups.

Note

You can assign a user to exactly one user group.

Note

On an HMI device the number is limited to 100 users and one PLC user. This restriction does not apply to PCs. On a PC, the maximum number of users is restricted by the physical memory.

Inspector window

When you select a user, you can change the password in the "General" group. Under "Automatic logoff" you can specify if the user is to be automatically logged off by the HMI device when there is no operator activity after the specified time.

Backup and restore

The user data is encrypted and saved on the HMI device to protect it from loss due to power failure.

The users, passwords, group assignments and logoff times set up on the HMI device can be backed up and restored. This prevents you having to enter all of the data again on another HMI device.

Note

The currently valid user data is overwritten in the following cases:

- Depending on settings, the next time the project is downloaded
- Upon restore of a backed-up project
- Upon import of the user administration via an operator control. The newly downloaded or restored user data and passwords take effect immediately.

Note

The user data from earlier Runtime versions can be backed up and restored in the V14.

The user data of the V14 cannot be restored to earlier versions.

See also

Settings for the user administration (Page 1319)

8.1.4 Users work area (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The "Users" work area lists the users and user groups in table form. You administrate the users and assign them to a user group.

Principle

The work area consists of the "Users" and "Groups" tables.

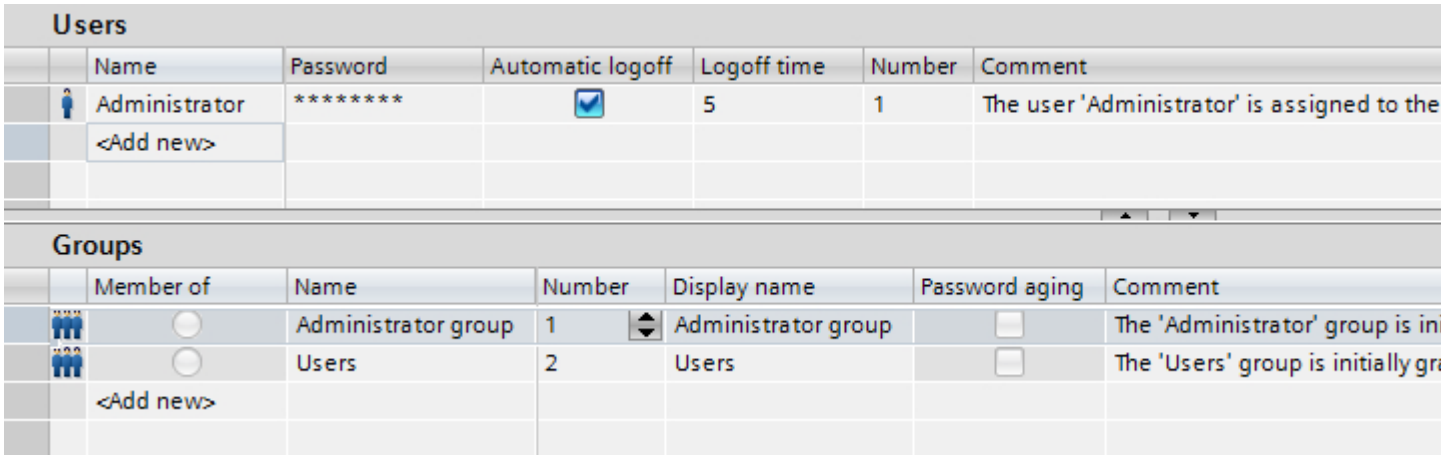

The "Users" table shows the existing users. When you select a user in this table, the "Groups" table shows the user group to which the user belongs.

Note

For the user "Administrator", the default password is "administrator". For security reasons, you should change the password of this user.

See also

Settings for the user administration (Page 1319)

User groups (Page 1313)

User groups work area (Page 1314)

Settings for the user administration (Page 1316)

Settings for the user administration (Page 1322)

8.1.5 User groups (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can create users and authorizations in the "User groups" tab of the "User Administration" editor. The "User groups" tab is part of the user administration in WinCC.

Open

Double-click the "User administration" in the project window. Open the "User groups" tab.

Work area

The user groups and authorizations are managed in the work area:

- You create new user groups and authorizations or delete them.
- You assign the authorizations to the user groups.

Inspector window

When a user group or an authorization is selected, you can edit the name in the "General" group. You can also enter a brief description in the "Comment" group.

See also

Settings for the user administration (Page 1319)

Users work area (Page 1312)

User groups work area (Page 1314)

Settings for the user administration (Page 1316)

Settings for the user administration (Page 1322)

8.1.6 User groups work area (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The "User groups" work area shows a table of the groups and their authorizations. You administer the user groups and assign authorizations to them.

Principle

The work area consists of the "Groups" and "Authorizations" tables.

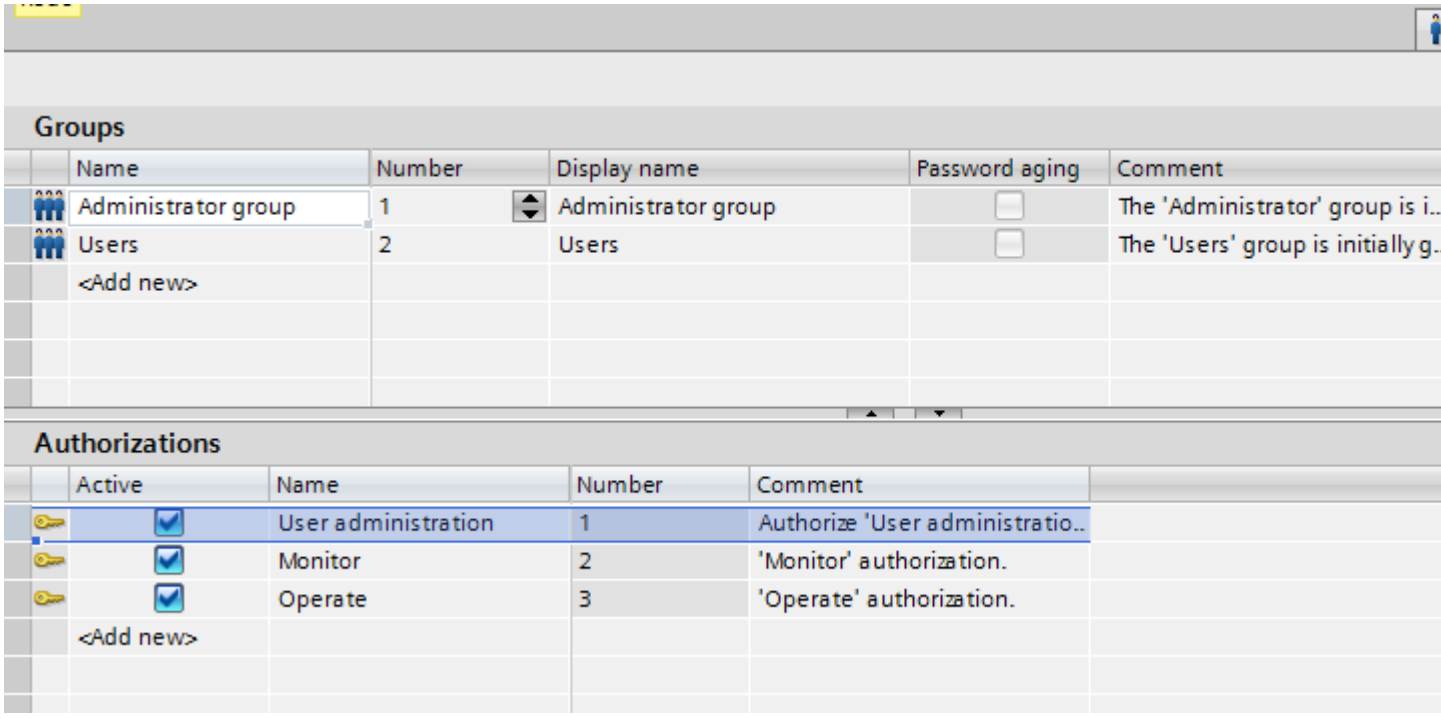

The "Groups" table shows the existing user groups. When you select a user group in this table, the "Active" column of the "Authorizations" table shows the authorizations which were assigned to the user group.

The number of the user group and of the authorization is assigned by the user administration. The designations and descriptions are assigned by you.

The number of predefined authorizations are fixed. Authorizations that you create can be freely edited. Ensure that the assigned numbers are unique.

See also

Settings for the user administration (Page 1319)

Users work area (Page 1312)

User groups (Page 1313)

Settings for the user administration (Page 1316)

Settings for the user administration (Page 1322)

8.1.7 Settings for the user administration (Basic Panels)

Introduction

In the "Runtime settings > User administration" editor, configure the security settings for users and their passwords in runtime.

Open

Double-click the "Runtime settings" editor in the project window. Click "User administration".

Work area

You carry out the settings for the validity of passwords in runtime in the work area. You determine the complexity of the password, for example.

Note

If the password from a previous project no longer corresponds to the policy, the user is prompted to enter a new password.

Effects in Runtime

The security settings have the following effects in runtime, depending on the configuration.

- "General" group
  - "Enable limit for logon attempts" check box selected
    The number entered in the "Number of incorrect logon attempts" box defines the number of logon attempts a user is allowed before being assigned to the "Unauthorized" group.
    "Enable limit for logon attempts" check box not selected
    The user has an unlimited number of logon attempts in runtime.
  - "Number of incorrect logon attempts" field
    If you enter "4" in the field, for example, and the fourth logon attempt fails, the user is automatically assigned to the "Unauthorized" group.
    You can specify 1 to 9 attempts.
  - "Logon only with password" check box
    When the check box is selected, the user will be authenticated by the password. The user name is not required.
    To match users to passwords, you cannot configure passwords more than once.

    Note

    If the "Logon only with password" option is enabled, the system sets the passwords for users that have already been created. It is therefore recommended to always enable the "Logon only with password" option before creating users.

    If you transfer user data from previous device versions in which the "Logon only with password" option was disabled, there may be multiple users with identical passwords. These users are listed in the user view. However, only users whose user name comes first by alphabetical order can log on in Runtime.

- "Hierarchy level" group
  - "Group-specific rights for user administration" check box
    When this check box is selected, administrators only manage users whose group number is less than or equal to their own.
    For example, an administrator whose group number is 5 can only manage users whose group number is less than or equal to 5. This means that the administrator can assign users only to groups with a group number less than or equal to 5.

- "Password" group
  - "Enable password aging" checkbox selected
    The password expires after the number of days set in the "Validity of the password (days)" field.
    The "Password aging" column is selected in the "User groups" editor. This enables you to specify group-by-group, if the passwords should expire or if the password generations should be saved. Passwords never expire for groups for which password aging is not enabled.
  - "Prewarning time (days)" field
    The user is informed that the password will expire the specified number of days before the password expires.
  - "Password generations" field
    If the user changes the password, the new password must be different from the specified number of previous passwords. The number of password generations ranges from 1 to 5.
- "Password complexity" group
  - "Must include special characters" check box selected
    The user must enter a password containing at least one special character at any position.
  - "Must include number" check box selected
    The user must enter a password containing at least one number at any position.
  - "Minimum password length" field
    The user must enter a password with a minimum length, as specified in the "Minimum password length" field.
    The minimum length of the password is 3 characters.

Note

If the "Must include special characters" option is enabled, the system sets the passwords for users that have already been created. It is therefore recommended to always enable the "Must include special characters" option before creating users.

The following special characters are permitted for passwords:

-+.,/*:\"?_()=^!€%~`§'{}[]<>#$;&|@

See also

Settings for the user administration (Page 1319)

Users work area (Page 1312)

User groups (Page 1313)

User groups work area (Page 1314)

8.1.8 Settings for the user administration (RT Advanced, Panels, Comfort Panels)

Introduction

In the "Runtime settings > User administration" editor, configure the security settings for users and their passwords in runtime.

Open

Double-click the "Runtime settings" editor in the project window. Click "User administration".

Work area

You carry out the settings for the validity of passwords in runtime in the work area. You determine the complexity of the password, for example.

Note

If the password from a previous project no longer corresponds to the guideline, the user is prompted to enter a new password.

Effects in Runtime

The security settings have the following effects in runtime, depending on the configuration.

- "General" group
  - "Change initial password" check box selected
    The user must change the administrator-assigned password when logging on for the first time.
  - "Change logoff time" checkbox selected
    Simple user rights are sufficient for changing the logoff time.
    The logoff time is the period after which the user administration automatically logs off a user when no operator activity is detected in the system.
    The logoff time for the SIMATIC Logon user corresponds to the logoff time of the default user "Administrator".
    Any user changes of the logoff time are logged in the Audit Trail.
  - "Enable limit for logon attempts" check box selected
    The number entered in the "Number of incorrect logon attempts" box defines the number of logon attempts a user is allowed before being assigned to the "Unauthorized" group.
    "Enable limit for logon attempts" check box not selected
    The user has an unlimited number of logon attempts in runtime.
  - "Number of incorrect logon attempts" field
    If you enter "4" in the field, for example, and the fourth logon attempt fails, the user is automatically assigned to the "Unauthorized" group.
    You can specify 1 to 9 attempts.
  - "Logon only with password" check box
    When the check box is selected, the user will be authenticated by the password. The user name is not required.
    To match users to passwords, you cannot configure passwords more than once.

    Note

    If the "Logon only with password" option is enabled, the system resets the passwords for users that have already been created. We therefore recommend that you always enable the "Logon only with password" option before creating users.

    If you transfer user data from previous device versions in which the "Logon only with password" option was disabled, there may be multiple users with identical passwords. These users are listed in the user view. However, only users whose user name comes first by alphabetical order can log on in Runtime.

- "Hierarchy level" group
  - "Group-specific rights for user administration" check box
    When this check box is selected, administrators only manage users whose group number is less than or equal to their own.
    For example, an administrator whose group number is 5 can only manage users whose group number is lower than or equal to 5. This means that the administrator can assign users only to groups with a group number lower than or equal to 5.

- "Password" group
  - "Enable password aging" checkbox selected
    The password expires after the number of days set in the "Validity of the password (days)" field.
    The "Password aging" column is selected in the "User groups" editor. This enables you to specify group-by-group, if the passwords should expire or if the password generations should be saved. Passwords never expire for groups for which password aging is not enabled.
  - "Prewarning time (days)" field
    The user is informed that the password will expire the specified number of days before the password expires.
  - "Password generations" field
    If the user changes the password, the new password must be different from the specified number of previous passwords. The number of password generations ranges from 1 to 5.
- "Password complexity" group
  - "Must include special characters" check box selected
    The user must enter a password containing at least one special character at any position.
  - "Must include number" check box selected
    The user must enter a password containing at least one number at any position.
  - "Minimum password length" field
    The user must enter a password with a minimum length, as specified in the "Minimum password length" field.
    The minimum length of the password is 3 characters.

    Note

    If the "Must include special characters" option is enabled, the system resets the passwords for users that have already been created. We therefore recommend that you always enable the "Must include special characters" option before creating users.

    The following special characters are permitted for passwords:

    -+.,/*:\"?_()=^!€%~`§'{}[]<>#$;&|@
- "SIMATIC Logon" group
  - "Activate SIMATIC Logon" check box selected
    A connection is established to the server. Authorization is performed via SIMATIC Logon.
  - "Encrypted transfer" checkbox selected
    The data is encrypted before it is transferred to the server.

See also

Central user administration using SIMATIC Logon (Page 1333)

User (Page 1311)

Users work area (Page 1312)

User groups (Page 1313)

User groups work area (Page 1314)

Settings for the user administration (Page 1316)

Settings for the user administration (Page 1322)

8.1.9 Settings for the user administration (RT Professional)

Introduction

In the "Runtime settings > User administration", editor you specify whether users log on dynamically to an HMI device in runtime. You can also activate a central user administration by means of SIMATIC Logon.

Open

Double-click the "Runtime settings" editor in the project window. Click "User administration".

Note

The view of the work area depends on the HMI device in use.

Work area

In the work area, you can specify how the operator logs on to the HMI device:

- Dynamic logon
  With dynamic logon, you assign a variable to each HMI device. A variable value is assigned to a user. The user then logs on in runtime by setting the variable value on an HMI device, for example, using a key switch.
- SIMATIC Logon
  When you use SIMATIC Logon, you specify the display name that is shown on the HMI device. SIMATIC Logon supports centralized user administration. As soon as you select the "SIMATIC Logon" option, the users created in the "User administration" editor will be ignored. The "Users" work area is no longer displayed.
  Only groups whose settings agree with the names of the Windows groups are important.

  Note

  If you use SIMATIC Logon , you cannot use the "Automatic Logon" option.

Effects in Runtime

With dynamic logon, the user is logged on to the HMI device as soon as the variable contains the correct value.

The set automatic logoff time does not affect users that log on via a logon tag.

When SIMATIC Logon is used, the operator logs on to the Windows operating system. The user settings, such as the password validity, are read by SIMATIC Logon from the user administration of the operating system.

See also

Settings for the user administration (Page 1319)

Users work area (Page 1312)

User groups (Page 1313)

User groups work area (Page 1314)

Logging on users dynamically (Page 1332)

Central user administration using SIMATIC Logon (Page 1333)

8.1.10 Target groups in user administration (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Principle

The target groups serve as examples for different groups of persons who use the user administration. We can distinguish between the following four groups in the user administration:

1. Administrator OEM
2. Administrator RT
3. Project engineer
4. Operator

As Administrator OEM, you create the user groups, users and authorizations for Runtime in the Engineering System of, for example, an equipment manufacturer.

As Administrator RT, you administer users in Runtime by means of the "User view."

As the project engineer, you assign the authorizations to the user groups in the Engineering System. You also configure authorizations for objects.

As Operator, you log on in Runtime. You can only access a protected object if you have the required authorization.

Note

The Administrator RT target group already exists in the Runtime user administration as the predefined user group "Administrator group." For ease of understanding, the predefined user groups and authorizations are not used below.

8.2 Setting up the user administration

8.2.1 Administering users for Runtime (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

8.2.1.1 Creating an authorization (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You create an authorization and assign it to one or more user groups.

Requirements

The "User groups" work area is open.

Procedure

1. Double-click "Add" in the "Authorizations" table.
2. Enter "Stop runtime" as the name of the authorization.
3. Enter a brief description as the "Comment".

See also

Creating a user group (Page 1324)
Managing user groups (Page 1329)
Changing the number of the user group (Page 1331)

8.2.1.2 Creating a user group (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

User groups are created so that you do not have to assign an authorization to every single user. You create a user group, assign authorizations and then assign users to it.

Note

The name of the user group has to be unique within the project. Otherwise the input is not accepted.

Note

Using SIMATIC Logon

Ensure that the names of the user groups in Windows and WinCC are identical.

Requirements

The "User groups" work area is open.

Procedure

1. Double-click "Add" in the "Groups" table.
2. Enter "Operators" as the "Name" of the user group.
3. Change the "Number" of the user group as required.
4. Enter "Display name" of the "Operators" user group.
5. Enter a brief description as the "Comment".

In runtime, the user view shows the display name of the user group. The display name of the user group depends on the language. You can specify the name in several languages and switch between languages in runtime.

See also

Creating an authorization (Page 1324)

8.2.1.3 Creating a user group (RT Professional)

Introduction

User groups are created so that you do not have to assign an authorization to every single user. You create a user group, assign authorizations and then assign users to it.

Note

Using SIMATIC Logon

Ensure that the names of the user groups in Windows and WinCC are identical.

Note

The name of the user group has to be unique within the project. Otherwise the input is not accepted.

Requirements

The "User groups" work area is open.

Procedure

1. Double-click "Add" in the "Groups" table.
2. Enter "Operators" as the name of the user group.
3. Change the "Number" of the user group as required.
4. Enter a brief description as the "Comment."

8.2.1.4 Assigning an authorization (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

When you allocate an authorization to a user group, all assigned users have this authorization.

Requirements

- A "Stop runtime" authorization has been created.
- An "Operators" user group has been created.
- The "User groups" work area is open.

Procedure

1. Click on the "Operators" user group in the "Groups" table. The "Authorizations" table shows all authorizations.
2. Enable the "Stop runtime" authorization in the "Authorizations" table.

Note

The "Stop runtime" authorization is only a designation and does not have any relation to the function "StopRuntime". You have to establish this relation yourself. To do so, configure the "StopRuntime" system functions at a button and select "Stop runtime" as the "Authorization".

8.2.1.5 Creating users (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You create a user so that users can log on with their user names in runtime after loading to the device.

As an alternative, you can create and change users by means of the "User view" in Runtime.

Note

Applies only to Runtime Professional:

New users who want to access through a Web client or a WebUX client on the system, can only be created in the engineering system.

When you create a user in Runtime with the user view, it is not possible to assign a Web startup screen to this user. This user is therefore not authorized to access the Web.

Neither can existing users be granted web access rights in Runtime.

In order for a created user to have authorizations you have to assign him to a user group and allocate authorizations to the user group.

The logon is successful when the user name entered during the logon matches a user in Runtime. In addition, the entered password must agree with the stored password of the user.

Note

Note that the entry is case-sensitive.

Requirements

The "Users" work area is open.

Procedure

1. Double-click "Add" in the "Users" table.
2. Enter "Foreman" as the user name.

   Note

   The user name must be unique within the project. Otherwise the input is not accepted.

   Note

   Do not use special characters such as / " § $ % ? ' & when you enter a user name and a password.

3. Click the button in the "Password" column. A dialog box for entering the password is displayed.
4. Enter the password of the user.
5. To confirm the password enter it a second time in the lower field.
6. Close the dialog box using the icon.
7. If a user is to be logged off after a specific time period, activate "Automatic logoff".

8. Click in the "Logoff time" column. The preset value for "Logoff time" is 5 minutes.
9. Enter a brief description as the "Comment".

8.2.1.6 Assigning a user to a user group (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

When you assign a user to a user group, the user has the authorizations of the user group.

Note

You have to assign a user to exactly one user group. The assignment is checked during the consistency check and generation of the project.

Requirements

- The user "Foreman" has been created.
- An "Operators" user group has been created.
- The "Users" work area is open.

Procedure

1. Click on the "Foreman" user in the "Users" table. The "Groups" table shows all user groups.
2. Activate the "Operators" user group in the "Groups" table.

8.2.1.7 Managing users (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In the work area, you can administer users and assign them to user groups.

Requirements

The "Users" work area is open.

Changing the user name

1. In the "Users" table, double-click the field in the "Name" column to change the user name.
2. Change the user name.
3. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the user name under "Properties > Properties > General" in the Inspector window.

Changing the password of the user

1. Click the button in the "Password" column of the "Users" table. A dialog for entering the password opens.
2. In the "Enter password" field, enter the new password.
3. Reenter the new password in the "Confirm password" field.
4. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the password under "Properties > Properties > General" in the Inspector window.

Displaying invisible columns

1. Position the mouse cursor on the title of the "Users" table.
2. Right-click to open the shortcut menu and enable the display of the "Logoff time" column, for example.

Changing the logoff time of the user

1. In the "Users" area, double-click on the field in the "Logoff time" column to change the logoff time.
2. Change the logoff time.
3. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the logoff time under "Properties > Properties > Automatic logoff" in the Inspector window.

Deleting a user

1. Select the line of the user to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

Note

Predefined users cannot be deleted.

8.2.1.8 Managing user groups (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In the workplace you administer user groups and assign authorizations for use in runtime.

Requirements

The "User groups" work area is open.

Changing the name of the user group

1. In the "Groups" table, double-click the field in the "Name" column to change the name of the user group.
2. Change the name of the user group.
3. Confirm your entry with <Return>.

Alternatively, select the user group in the work area. Change the name under "Properties > Properties > General" in the Inspector window.

Note

Predefined user groups cannot be deleted.

Displaying invisible columns

1. Position the mouse cursor on the title of the "Users" table.
2. Right-click to open the shortcut menu and enable the display of the "Display name" column, for example.

Changing the displayed name of the user group

1. In the "Groups" table, double-click the field in the "Display name" column to change the display name of the user group.
2. Change the displayed name of the user group.
3. Confirm your entry with <Return>.

Alternatively, select the user group in the work area. Change the display name under "Properties > Properties > General" in the Inspector window.

Deleting a user group

1. Mark the line of the user group to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

   Note

   Predefined user groups cannot be deleted.

Changing the name of the authorization

1. In the "Authorizations" table, double-click the field in the "Name" column to change the name of the authorization.
2. Change the name of the authorization.
3. Confirm your entry with <Return>.

Alternatively, select the authorization in the work area. Change the name under "Properties > Properties > General" in the Inspector window.

Deleting authorizations

1. Mark the line of the authorization to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

Note

Predefined authorizations cannot be deleted.

See also

Creating an authorization (Page 1324)

8.2.1.9 Changing the number of the user group (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

To assign group-specific rights, edit the number of a user group.

Requirements

- In the "Runtime settings> User administration" editor, "Group-specific rights for user administration" is selected.
- The "Operator" user group has been created.
- A "User administration" authorization has been assigned.
- The "User groups" work area is open.

Procedure

1. Click on the "Operator" user group in the "Groups" table.
2. Enter "6" in the "Number" column of the user group.
3. Enable the "User administration" authorization in the "Authorizations" table.

Result

The members of the "Operator" user group can only manage users whose group number is less than or equal to 6.

See also

Creating an authorization (Page 1324)

8.2.1.10 Logging on users dynamically (RT Professional)

Introduction

Two configuration steps are required to allow "Dynamic logon":

1. Assign an HMI device to a configured variable.
2. Assign a specific variable value to a user.

Note

If you have activated SIMATIC Logon, dynamic logon is not possible.

Requirements

- User administration exists.
- A "Login" variable of the "Byte" type is created.

Procedure

Assignment of the HMI device to the variable

1. Double-click "Runtime settings > User administration" in the project window.
2. Double-click "Add" in the "Dynamic logon" area of the table.
3. Click the button in the "HMI device" column. A dialog box for selecting the HMI device opens.
4. Select an HMI device.
5. Click the button in the "Variable" column. A dialog box for selecting the variable opens.
6. Select the "Login" variable.

Assignment of the user to the variable value

1. Open the "Users" work area.
2. Position the mouse cursor on the title of the "Users" table.
3. Open the shortcut menu with the right mouse button and activate the display of the "Dynamic logon" and "User identification" columns.
4. Double-click "Add" in the "Users" table.

5. Enter "Smith" as the user name.
6. Click the button in the "Password" column. A dialog box for entering the password is displayed.
7. Enter the password of the user.
8. To confirm the password enter it a second time in the lower field.
9. Close the dialog box by using the icon.
10. Activate the "Dynamic logon" column.

Result

When the variable equals 1, the user "Smith" will be logged on.

Note

The used variable must be of the type "binary" or 8, 16 or 32 bit.

8.2.2 Managing users on the server

8.2.2.1 Central user administration using SIMATIC Logon (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

To manage users and user groups for several applications or HMI devices, activate SIMATIC Logon.

Principle

SIMATIC Logon is a tool for system-wide user administration. If you use SIMATIC Logon, the users are centrally managed outside the application or HMI device.

You configure the user groups and their permissions as you usually do with the local user administration in WinCC. You assign identical names to the user groups on the server and in WinCC. Authorizations are assigned in runtime to a user group when the names are identical.

You do not have to create users in WinCC because they are taken dynamically from Windows user management during the logon process. The application or HMI device forwards each logon or password change to SIMATIC Logon for processing.

Note

SIMATIC Logon is a product requiring a license. For more information on SIMATIC Logon go to <http://support.automation.siemens.com>.

Enter the ID "34519648" in the search field and start searching. The "SIMATIC Logon - Electronic Signature" manual is available to download.

Log on process via SIMATIC Logon Service

The following diagram illustrates the process that runs automatically when a user logs onto Runtime.

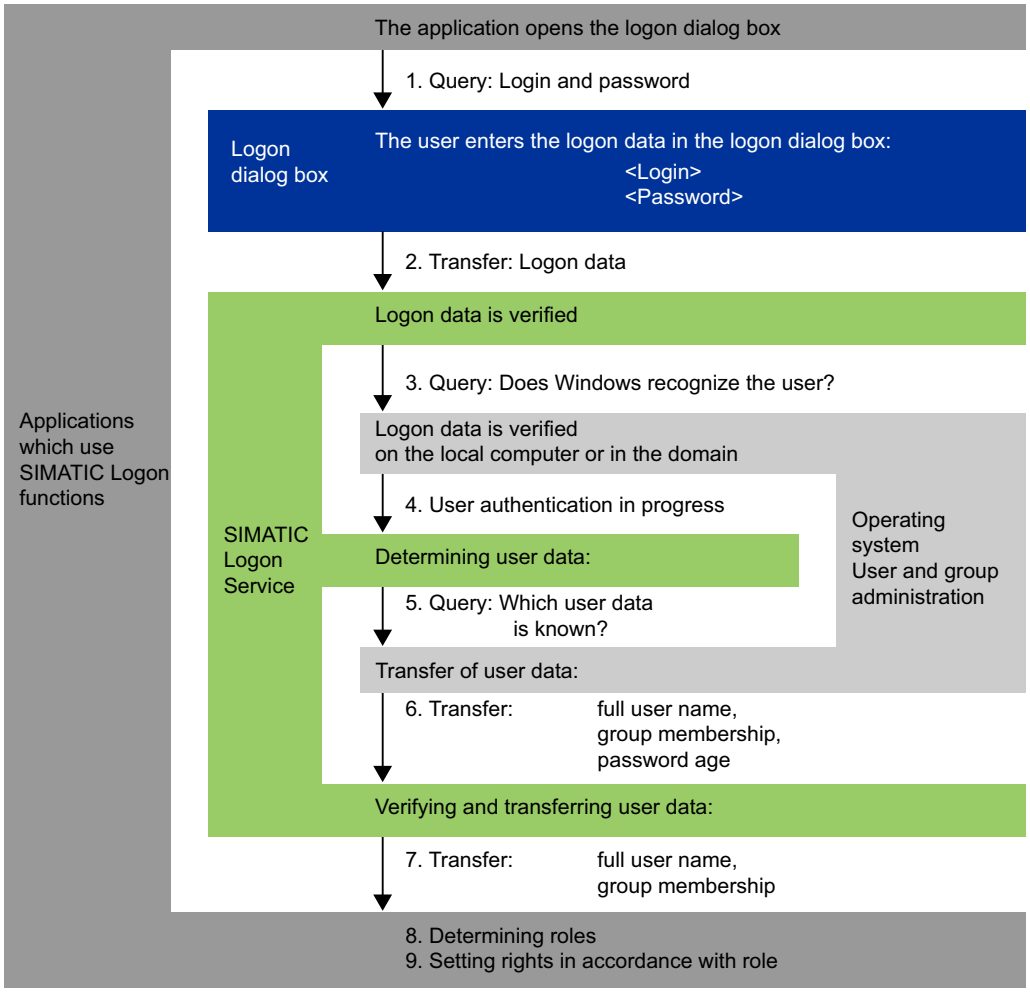

Results: The user can implement the role functions.

See also

Settings for the user administration (Page 1319)

Authentication and certificate management in SIMATIC Logon (Page 1335)

8.2.2.2 Authentication and certificate management in SIMATIC Logon

Introduction

SIMATIC Logon (V1.5 SP3 and later) supports secure encrypted communication connections to the following device families as of device version V13 SP1 and higher:

- Comfort Panels
- KTP Mobile Panels
- Runtime Advanced

Note

HMI devices with a device version V13 or earlier can only make a non-secure / non-encrypted connection to a SIMATIC Logon server.

Basics

The first time a connection is made the SIMATIC Logon certificate is compared with the local certificate of the HMI device. The compared certificate is stored under "SimaticLogon\rejected". If you trust the certificate of the SIMATIC Logon server, copy the stored certificate into the local certificate store.

The certificate store is located in the following path:

- On the PC usually under "C:\Program Files\Siemens\CoRtHmiRtm\SimaticLogon\certs"
- On HMI panels under "\flash\simatic\SimaticLogon\certs"

When the comparison has been successfully performed can a secure encrypted connection be successfully established.

You can find additional information about tested browsers on the Internet on the Customer Support pages, entry ID=109480490:

- https://support.industry.siemens.com/cs/document/109480490/ (https://support.industry.siemens.com/cs/de/de/view/109480490)

See also

Central user administration using SIMATIC Logon (Page 1333)

8.2.2.3 Making settings on the server (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Perform the following steps on your sever to ensure correct operation of SIMATIC Logon.

Procedure

1. Install SIMATIC Logon Services.
2. Create the user group in the user administration of the operating system.
3. Create the users in the user administration of the operating system.

   Note

   Users of SIMATIC Logon must be members of a user group of the operating system. Members of a subgroup cannot be logged on.

4. Assign the new users to a group.
5. Transfer the licenses for each HMI device on the server.

Note

You can find more detailed information in the documentation supplied with SIMATIC Logon.

Note

The password policies stored on the server are valid if users are authorized by means of SIMATIC Logon.

Note

You can configure the logoff time for the SIMATIC Logon user with "Configure SIMATIC Logon" on the server.

You set the automatic logoff time identical for all users, regardless of the panels.

8.2.2.4 Logging on using SIMATIC Logon (RT Advanced, Panels, Comfort Panels)

Requirements

- User administration exists.
- The groups created in WinCC agree with the Windows groups on the server.
- There is a server with SIMATIC Logon Service installed.

Note

A user may only be a member of one user group in WinCC. When a user is a member of several user groups on the server, only one of these user groups can be made known in WinCC.

"Administrators" and "Users" are predefined groups in both Windows and WinCC. Any new user created in Windows is automatically a member of the "Users" group. A user will be a member of two groups if assigned to a new group, for example, to the "SL user" group.

Remove this user from the "Users" group of the operating system to enable logon of this user using SIMATIC Logon.

A system message in runtime always confirms the successful logon of a user on the server.

Note

When you use SIMATIC Logon to manage access to a panel or a device with WinCC Runtime Advanced, you must note that the characters '/' and '\' cannot be used in the names of Windows user groups or Windows users.

Procedure

1. Open the "Runtime settings > User administration" editor in the Project window.
2. Select "Enable SIMATIC Logon".
3. Select "Windows domain" or "Windows computer", depending on where you administer your users.
4. Enter the name or IP address of the SIMATIC Logon server in the "Server name" text box.

   Note

   Make sure your SIMATIC Logon Server always has the same IP address if you use IP addresses.

5. Enter the port number for TCP/IP communication in the "Port number" text box. The valid range of values lies between 1024 and 49151. The preset port number is "16389" and used by SIMATIC Logon as the default.
6. Enter the Windows domain or the logon server with the user data in the "Windows domain" text box.
   Enter the name of the sever to access users on the logon server.
   Enter the name of the domain if the logon sever is in a domain and you wish to access users of the domain.

   Note

   Local Windows users are not accepted when the logon server is in a domain.

7. Activate the "Encrypted transfer" check box in order to encrypt the data before sending it to the server.

Emergency user

If the server cannot be accessed, all local users that were created in WinCC user administration can also be used as emergency users.

Emergency users have the rights of the user group to which they were assigned.

Note

For additional information, refer to the included SIMATIC Logon documentation.

Can be located under: [Installation directory SIMATIC Logon] \manuals

8.2.2.5 Logging on using SIMATIC Logon (RT Professional)

Requirements

- User administration exists.
- The groups created in WinCC agree with the Windows groups on the server.
- There is a server with SIMATIC Logon Service installed.

Note

When a user is a member of several user groups, the user is assigned the sum of authorizations for the groups. Logon is possible.

Procedure

1. Double-click the "Runtime settings" in the project window.
2. Double-click the "User administration" in the work area.
3. Activate "SIMATIC Logon."

8.2.3 Administering users for Web Client (RT Professional)

8.2.3.1 Administering user groups for Web Client (RT Professional)

Introduction

For each user group to be administered for web clients, you must configure the following:

- A start screen of your choice enabled for web access
- A language of your choice
- Number of reserved licenses (for WebUX)

This allows you to make various sections of a project immediately accessible or inaccessible for users.

Note

The name of the user group has to be unique within the project. Otherwise the input is not accepted.

Requirements

- The "Start" screen has been created and the web access is activate.
- The selected project language is activated.
- The "Web operators" user group has been created.

Procedure

1. Double-click the "Runtime settings" editor in the project window.
2. Click "User administration".
3. In the "User groups" tab, select the user group "Web operators".
4. In the Inspector window, select "Properties > Properties > Web options".
5. Under "WebNavigator settings", click the button in the "Start screen" selection list.
   A dialog box for selecting the start screen opens.
6. Select the "Start" screen for runtime.
7. Close the dialog box by using the icon.
8. Repeat steps 5 to 7 to set a start screen for WebUX.
9. To configure a language for the user group, click the button under "Options" in the "Web language" selection list.
   A dialog box for selecting the language opens.
10. Select the desired language as the runtime language of the user group.

Alternatively, you can make these settings in the shortcut menu by selecting the columns "WebNavigator start screen", "WebUX start screen" and "Web language" and making the respective configuration.

Note

The name of the user group is language-dependent. You can specify the name in several languages and switch between languages in runtime.

8.2.3.2 Managing users for the Web client (RT Professional)

Introduction

You must configure the following for each user to be logged on to the Web client (WebNavigator or WebUX):

- A start screen of your choice enabled for web access
- A language of your choice
- Number of reserved licenses (for WebUX)

This allows you to make various sections of a project immediately accessible or inaccessible for users.

Note

New users who want to access through a Web client or a WebUX client on the system, can only be created in the engineering system.

When you create a user in Runtime with the user view, it is not possible to assign a Web startup screen to this user. This user is therefore not authorized to access the Web.

Neither can existing users be granted web access rights in Runtime.

Requirements

- The "Start" screen has been created and the web access is activate.
- The selected project language is activated.

Procedure

1. Double-click the "Runtime settings" editor in the project window.
2. Click "User administration".
3. In the "Users" tab, select a user for whom you want to make the settings.
4. In the Inspector window, select "Properties > Properties > Web options".
5. Under "WebNavigator settings", click the ... button in the "Start screen" selection list.
   A dialog box for selecting the start screen opens.
6. Select the "Start" screen.
7. Close the dialog box by using the icon.
8. Repeat steps 5 to 7 to set a start screen for WebUX.

9. Under "Reserved number of licenses", you configure the number of reserved WebUX licenses for the selected user using the button.

Note

If more reserved licenses are configured than are available on the WebUX server, only the first users listed under the user administration are taken into consideration.

Note

If the "Use WebNavigator licenses for WebUX" option is selected, no reserved WebUX licenses can be guaranteed.

10. To configure a language for the user, click the button under "Options" in the "Web language" selection list.
A dialog box for selecting the language opens.
11. Select the desired language as the runtime language of the web user.

Alternatively, you can make these settings in the shortcut menu by selecting the columns "WebNavigator start screen", "WebUX start screen" and "Web language" and "Licenses reserved for WebUX" and making the respective configuration.

Note

The language setting is retained when the user becomes part of a new group. However, the start screen is overwritten.

Note

By default, the WebNavigator licenses are used for WebUX. If you want to use these licenses separately, clear the "Use WebNavigator licenses for WebUX" check box under "Runtime settings > Web access".

8.2.4 Administering users in Runtime

8.2.4.1 Users in runtime (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Principle

You create users and user groups and assign authorizations to them. You configure objects with authorizations. After download to the HMI device, all objects which were configured with an authorization are protected against unauthorized access in Runtime.

User view

When you configure a user view in the Engineering System, you administer users in this user view following download to the HMI device.

NOTICE
Changes in the user view are effective immediately in Runtime. Changes in runtime are not updated in the engineering system. When downloading the user administration to the HMI device, all changes in the user view are overwritten after a security prompt and based on the settings.

Users who have a "User administration" authorization have unlimited access to the user view. This allows them to administer all users. Any other user has only limited access to the user view for self administration.

See also

User view (Page 1342)

Managing users in the complex user view (Page 1346)

Logging on as a user (Page 1350)

8.2.4.2 User view (RT Professional, RT Advanced, Panels, Comfort Panels)

Purpose

You configure a user view in the engineering system to also administer the users in Runtime.

Structure

The user view shows the following in each line:

- The user
- The encrypted password
- The corresponding user group
- The logoff time

If no user is logged on, the user view is empty. The content of the individual fields is displayed after logon.

User view of an administrator

User	Password	Group	Logoff time
Administrator	********	Administrator gr...	5
Johnson	********	Users	5
Meister	********	Group 2	5
PLC User	********	Unauthorized	5

When an administrator is logged on, the user view shows all the users. The administrator changes the user name and the password. The administrator creates new users and assigns them to an existing user group.

User view of a user

User	Password	Group	Logoff time
Johnson	**********	Users	5

When no administrator is logged on, the user view shows only the logged-on user. Users can change their own passwords and logoff time.

See also

Users in runtime (Page 1341)

Logging on as a user (Page 1350)

8.2.4.3 Configuring a user view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You configure a user view in the Engineering System to also administer users in Runtime.

Requirements

A screen has been created.

Procedure

1. Select the "User view" object from the "Controls" category in the toolbox.
2. Drag-and-drop the "User view" object into the screen.
3. Click on "Properties > Properties" in the Inspector window.
4. Specify the appearance of the "User view".
5. You can, for example, select " "Display mode > Fit to size > Fit object to contents".

Result

You have created a user view in the screen.

See also

Logging on as a user (Page 1350)

8.2.4.4 Creating users (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, Multipanels, Mobile Panels)

Introduction

You create a user so that users can log on under their user name in runtime.

As an alternative, you can create users in the engineering system and download them to the HMI device.

Note

Applies only to Runtime Professional:

New users who want to access through a Web client or a WebUX client on the system, can only be created in the engineering system.

When you create a user in Runtime with the user view, it is not possible to assign a Web startup screen to this user. This user is therefore not authorized to access the Web.

Neither can existing users be granted web access rights in Runtime.

The logon is successful only when the user name entered during the logon matches a user in runtime. In addition, the password entered at logon has to match that of the user.

Note

Note that the entry is case-sensitive.

Note

Do not use special characters such as / " § $ % ? ' & when you enter a user name and a password.

You assign the user to a user group. The user then has the authorizations of the user group.

Note

Runtime users must be assigned to a user group. The user group is created in the engineering system. The designation of the user group is language-dependent.

Requirements

- The user view is open.
- A "Group 2" user group has been created.

Procedure

1. Click "<New User>" in the user view. A dialog opens.
2. Enter "Foreman" as the user name.
3. Press the <Return> button.
4. Click "Password."
5. Enter the password of the user.
6. Press the <Return> button. The password is hidden.
7. Click in the "Group" column.
8. Select "Group 2" as the "Group".

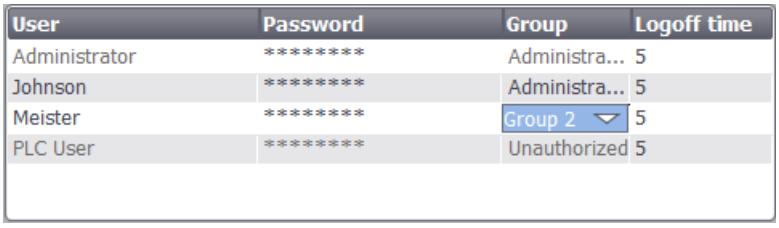

User	Password	Group	Logoff time
Administrator	********	Administra...	5
Johnson	********	Administra...	5
Meister	********	Group 2	5
PLC User	********	Unauthorized	5

9. Press the <Return> button.
10. Click in the "Logoff time" column.
11. Enter the time after which the user is logged off automatically.

See also

Logging on as a user (Page 1350)

8.2.4.5 Managing users in the complex user view (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

If you have configured a user view in the engineering system, the users and user groups can be managed in runtime.

NOTICE
Changes in the user view are effective immediately in runtime. Changes in runtime are not updated in the engineering system. When downloading the user administration to the HMI device, all changes in the user view are overwritten after a security prompt and based on the settings.

Requirements

- Runtime is enabled.
- A complex user view has been created.
- The screen with the user view is open.
- You have the default "User administration" authorization.

Note

If you do not have a "User administration" authorization, you can only change your own password and your logoff time.

Changing the user name

1. Enter a new user name in the "Users" column in the user display.
2. Confirm your entry with <Return>.

Note

The user can then no longer log on with his previous password in runtime. If you delete the name and press <Return>, the user is deleted.

Changing the password of the user

Availability of complex user display in selected devices only.

1. Enter a new password in the "Password" column in the user display.
2. Confirm your entry with <Return>.

Note

The user can then no longer log on with his previous password in runtime.

If you delete the password in the complex user view and press <Return>, the user will be deleted.

Changing the logoff time of the user

1. Enter a new logoff time in the "Logoff time" column in the user display.
2. Confirm your entry with <Return>.

Deleting a user

1. Click the name of the user to be deleted.
2. Delete the name.
3. Press the <Return> button.

Note

The user can no longer log on in runtime.

Assigning a user to a different user group

1. Activate the user group field for the corresponding user.
2. Select a user group.
3. Confirm your selection with <Return>.

See also

Users in runtime (Page 1341)

Logging on as a user (Page 1350)

8.2.4.6 Unlock locked out users (RT Advanced, Basic Panels, Panels, Comfort Panels)

Unlock locked out users

The check box "Activate limit for login attempts" is activated in the "Runtime settings > User administration".
The number 3 is entered in the field "Number of invalid login attempts".

If users have three failed attempts at login, e.g. by entering an incorrect password, they are assigned to the "Unauthorized" group. The user loses all authorizations. The user can still log on, but no longer has any authorizations. Only a user with administrator rights re-assigns the unauthorized user to a user group.

See also

Logging on as a user (Page 1350)

8.2.4.7 Logging on as a user (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

As a rule you log-on as a user by means of a special button. The logon dialog box is displayed to this purpose.

The logon dialog box is displayed by default during access to a protected object if

- No user is logged on in Runtime.
- The logged-on user does not have the required authorization.

Note

On Basic Panels, the system always opens the logon dialog when you press an access-protected button.

Requirements

- Under "Runtime settings > User administration" the
  - "Enable limit for logon attempts" check box has been selected.
  - The number 3 is entered in the field "Number of incorrect login attempts".
- The "ShowLogonDialog" system function is configured on a button called "Logon".

Procedure

1. Click the "Logon" button. The logon dialog box is displayed.

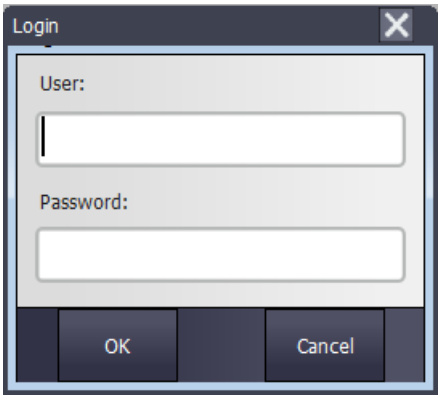

2. Enter your user name as it was specified in the user administration, for example "Foreman". If someone has logged on before you, the name of the user will be displayed.
3. Enter the corresponding password. The input is concealed.

   Note

   When entering the password, note that it is case sensitive.

4. Click "OK" to close the dialog box.

Note

When you change the HMI device, you do not need to re-enter the password if the permitted special characters are included in the password on all HMI devices.

The following special characters are permitted for passwords:

()€&@$%+#^[]|*-"<>!,.;:/_=\'? `~{}$

If you switch to V14 from an older device version and prohibited special characters were used for the password, the password is deleted. You are informed about this with an error message and have to save a new password.

Logon was successful

If you have entered the user name "Foreman" and the entered password corresponds with the stored password, you are logged on as the user "Foreman" in Runtime. You have the authorizations of the user "Foreman".

When the user "Foreman" accesses a protected object such as the "Logging" button, access to this protected object will only be authorized if the user "Foreman" has the required authorization. The programmed function is executed immediately.

If you do not have the required authorization after the successful log-on, a corresponding error message is displayed. However, you remain logged on in Runtime.

Logon was unsuccessful

An error message is displayed.

In order to maintain security, you or the user logged-on before you no longer has any authorizations. However, access to unprotected objects remains possible. The user view does not, however, show any entries. The user view and the authorizations change after the next successful log-on.

If the third login attempt has failed, the user will be assigned to the "Unauthorized" group. For this reason, do not configure a user group with this display name.

If the "Log off" function is called up or the logoff time of the user has expired, the user is logged off.

See also

Logging on as a user (Page 1350)

8.2.4.8 Logging on as a user (RT Professional)

Introduction

As a rule you log-on as a user by means of a special button. The logon dialog box is displayed to this purpose.

The logon dialog box is displayed by default during access to a protected object if

- No user is logged on in Runtime.
- The logged-on user does not have the required authorization.

Requirements

- The "ShowLogonDialog" system function is configured on a button called "Logon".

Procedure

1. Click the "Logon" button. The logon dialog box is displayed.

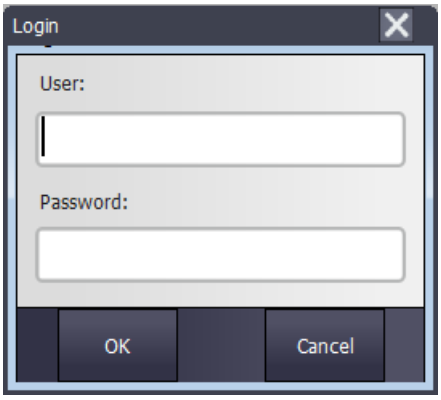

2. Enter your user name as it was specified in the user administration, for example "Foreman". If someone has logged on before you, the name of the user will be displayed.
3. Enter the corresponding password. The input is concealed.
4. Click "OK" to close the dialog box.

Logon was successful

If you have entered the user name "Foreman" and the entered password corresponds with the stored password, you are logged on as the user "Foreman" in Runtime. You have the authorizations of the user "Foreman".

When the user "Foreman" accesses a protected object such as the "Logging" button, access to this protected object will only be authorized if the user "Foreman" has the required authorization. The programmed function is executed immediately.

If you do not have the required authorization after the successful log-on, a corresponding error message is displayed. However, you remain logged on in Runtime.

Logon was unsuccessful

An error message is displayed.

In order to maintain security, you or the user logged-on before you no longer has any authorizations. However, access to unprotected objects remains possible. The user view does not, however, show any entries. The user view and the authorizations change after the next successful log-on.

If the "Log off" function is called up or the logoff time of the user has expired, the user is logged off.

See also

Users in runtime (Page 1341)

User view (Page 1342)

Configuring a user view (Page 1343)

Creating users (Page 1344)
Managing users in the complex user view (Page 1346)
Unlock locked out users (Page 1347)
Logging on as a user (Page 1348)
Exporting the user administration (Page 1352)
Importing the user administration (Page 1353)

8.2.4.9 Exporting the user administration (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

The users and user groups are transferred to the HMI device from the Engineering System. If you have configured a user view, you can administer the users at the HMI device via the user view.

If a user has access to several HMI devices, the same user and the same password must be configured on each HMI device. Create a standard for user administration of the different HMI devices. Export the users and passwords on an HMI device to a storage medium, such as a floppy, memory card or a network drive. You import the users and passwords from this storage medium at all the other HMI devices.

You program a function which exports the user data at a button.

The system function "ExportImportUserAdministration" is not available on all HMI devices.

NOTICE
When exporting, the user data are encrypted and written from the HMI device to the target file. The target file is overwritten

Requirements

- An HMI device has been created.
- A screen has been created.
- A button has been created in the screen.
- The Inspector window is open.

Procedure

1. Click the button in the screen.
2. Click "Properties > Events > Click" in the Inspector window.
3. Click the entry "Add function" in the function list.
4. Select the "ExportImportUserAdministration" system function.

5. Select "Export" as the "Direction". When exporting, the target file is overwritten with the user data.
6. Enter the full path of the destination file as the "File name", for example "a:\test \usersview.txt".

Note

Ensure that the file name is written correctly. If the directory, for example "\test\", does not exist, it will be created without a prompt.

See also

Logging on as a user (Page 1350)

8.2.4.10 Importing the user administration (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

If a user has access to several HMI devices, the same user and the same password must be configured on each HMI device. Create a standard for user administration of the different HMI devices. Export the users and passwords on an HMI device to a storage medium, such as a floppy, memory card or a network drive. You import the users and passwords from this storage medium at all the other HMI devices.

You configure a function which imports the user data at a command button.

The system function "ExportImportUserAdministration" is not available on all HMI devices.

NOTICE
When importing, the user data are read in from the source file and written into the HMI device. The users and passwords currently valid in the HMI device are overwritten. The imported users and passwords are valid immediately.

Requirements

- A screen has been created.
- A command button has been created in the screen.
- The Inspector window is open.

Procedure

1. Click the button in the screen.
2. Click "Properties > Events > Click" in the Inspector window.
3. Click the entry "Add function" in the function list.

4. Select the "ExportImportUserAdministration" system function.
5. Select "Import" as the "Direction." When importing, the user data in the HMI device are overwritten.
6. Enter the full path of the source file as the "File name", for example "a:\test\usersview.txt".

Note

Ensure that the file name is written correctly. If the folder, for example "\test\", or the file does not exist, an error message will be displayed.

See also

Logging on as a user (Page 1350)

8.2.5 Configuring access protection

8.2.5.1 Access protection (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You configure an authorization at an object in order to protect it against access. All logged-on users who have this authorization can then access the object. If a user does not have authorization to operate an object, the logon dialog is displayed automatically.

Note

Several system functions are available under "User administration" so that user, password, and user group can be edited, for example, in the PLC.

8.2.5.2 Configuring operating authorizations (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You configure the "Stop runtime" authorization on a button. Then only those users who have the appropriate authorization have access to this button, for example all the users of the "Operators" user group.

This ensures that access to the command button is protected. If a logged-on user who belongs to the "Operators" user group and has the required authorizations clicks the button, runtime is stopped.

An example describes in detail how to configure a command button with access protection.

Requirements

- The "Operators" user group has been created.
- The "Stop runtime" authorization has been created.
- A screen has been created and opened.
- The screen contains a button.

Procedure

1. Click the button in the screen.
2. Click "Properties > Properties > Security" in the Inspector window.
3. Select "Stop runtime" as the "Authorization".
4. In the Inspector window, select "Properties > Events > Click".
5. Select a system function from the function list, for example, "StopRuntime".

Note

The "Enable" and "Disable" events are only used to detect whether an object was selected or deselected. The events do not, however, trigger a password prompt.

Do not use the "Enable"or "Disable" event if you want to configure access protection at the function call of the object.

8.3 Reference

8.3.1 Objects with access protection (Basic Panels)

Introduction

The following objects can be configured with an authorization:

- I/O field
- Button
- Symbolic I/O field
- Graphic I/O field
- Date/time field
- Switch
- Alarm view
- User view
- Recipe view
- System diagnostic view

8.3.2 Objects with access protection (RT Advanced, Panels, Comfort Panels)

Introduction

The following objects can be configured with an authorization:

- I/O field
- Button
- Symbolic I/O field
- Graphic I/O field
- Date/time field
- Switch
- Symbol library
- Slider
- Effective range name
- Effective range name (RFID)
- Alarm view
- Alarm window

- User view
- HTML Browser
- Watch table
- Sm@rtClient view
- Recipe view
- f(x) trend view
- System diagnostic view
- System diagnostic window
- PLC code view
- GRAPH overview
- ProDiag overview

8.3.3 Objects with access protection (RT Professional)

Introduction

The following objects can be configured with an authorization:

- Connector
- Pipe
- Double T-piece
- T piece
- Pipe bends
- I/O field
- Editable text field
- List box
- Combo box
- Button
- Round button
- Symbolic I/O field
- Graphic I/O field
- Bar
- Slider
- Scroll bar
- Check box
- Option button

- Gauge
- Clock
- User view
- HTML Browser
- Recipe view
- Alarm view
- f(x) trend view
- f(t) trend view
- Table view
- Value table
- System diagnostics view
- Media Player
- Channel diagnostics view
- PLC code view
- GRAPH overview
- ProDiag overview

In order to configure the following objects with access control, you additionally link the system function "ShowLogonDialog" to an event, for example, Click.

- Line
- Polyline
- Polygon
- Ellipse
- Ellipse segment
- Circle segment
- Ellipse arc
- Circular arc
- Circle
- Rectangle
- Text field
- Graphic view
- Screen window
- Print job/Script diagnostics

8.3.4 Default user groups and authorizations (RT Advanced, Basic Panels, Panels, Comfort Panels)

Principle

The predefined user groups and authorizations have the following numbers:

User group	Number
"Administrator group"	1
"Users"	2

Authorization	Number
"User administration"	1
"Monitor"	2
"Operate"	3

8.3.5 Default user groups and authorizations (RT Professional)

Principle

The predefined user groups and authorizations have the following numbers:

User group:

- "Administrator group"
- "Users"

Authorization	Number
"User administration"	1
"Monitor"	2
"Operate"	3
"Enable remote control"	1000
"Web access - View only"	1002

Authorization 1000, "Enable remote control", is practical for client-server projects.

Authorization 1002, "Web access - view only" is significant in combination with the Web Navigator.

8.4 Examples

8.4.1 Example: Configuring a button with logon dialog box (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Task

In the following example, you configure the function "ShowLogonDialog" for a button. A different user can then log on in runtime when the shift changes, for example. In the process the user previously logged on is logged off.

Note

In runtime the logon dialog box is not displayed by default until you access a protected object. Either no user is logged on or the logged-on user does not have the required authorization.

Requirements

- A screen has been created.
- A button has been created in the screen.

Procedure

1. Click the button in the screen.
2. Click "Properties > Events > Release" in the Inspector window.

3. Click the entry "Add function" in the "Function list" table.
4. Select the system function "ShowLogonDialog" from the "User administration" group.

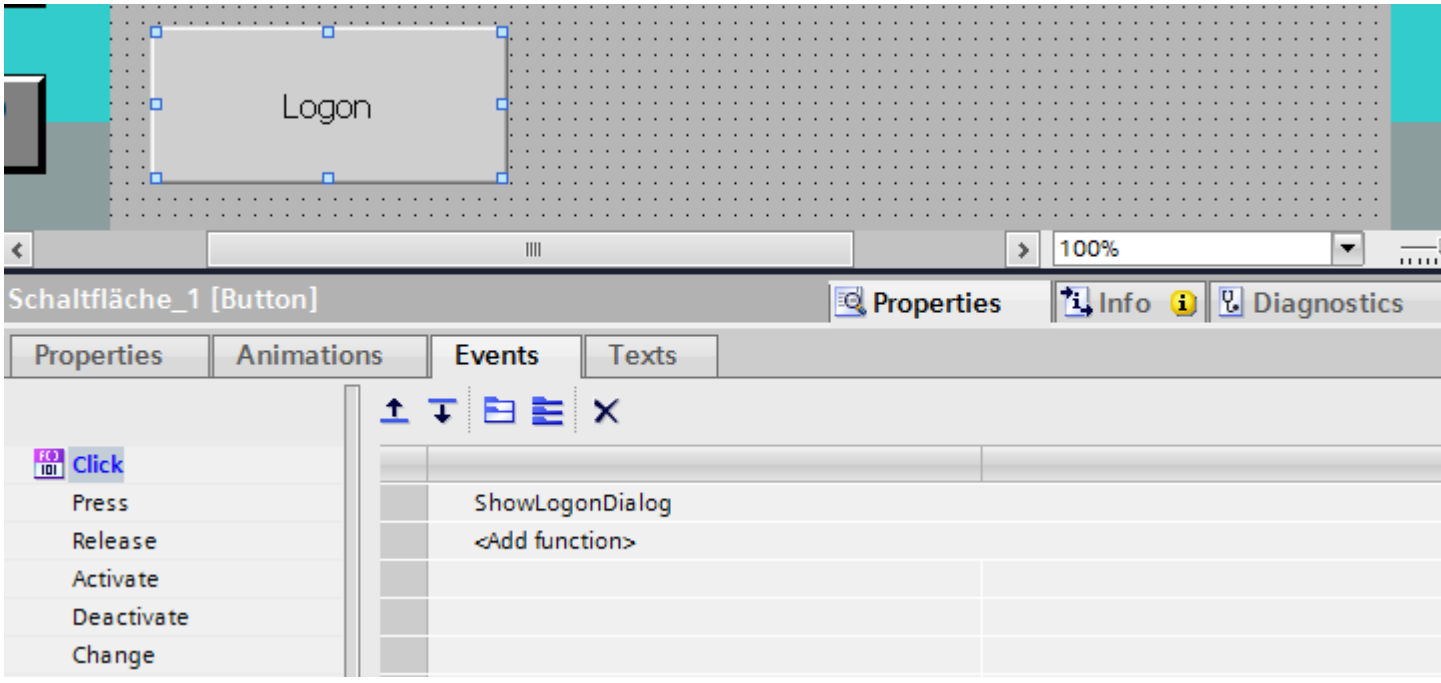

Result

If the user clicks on the button in runtime, the function "ShowLogonDialog" is called up. When the function "ShowLogonDialog" is called up, the logon dialog box is displayed. The user logs on with his user name and password.

8.4.2 Example: Logging the logon and logoff events (RT Advanced, Basic Panels, Panels, Comfort Panels)

Task

In the following example, you configure the function "TraceUserChange" to the event "User change".

Principle

The "TraceUserChange" function is called when a user logs on or off. When a function is called up, a system message with information about the corresponding user is output.

This system message can be archived. When archiving, the system message is provided with a date stamp and time stamp. This ensures that you can track which user was logged on at the HMI device at which time and for how long.

Requirements

- You have created an HMI device with Runtime Advanced.
- The Inspector window is open.

Procedure

1. Double-click the "Scheduler" in the Project view.
2. Double-click "Add" in the table of the tasks.
3. Enter "Logon-Protocol" as the "Name".
4. Select "User change" as the "Trigger".
5. Open "Properties > Events" in the Inspector window.
6. Click the entry "Add function" in the "Function list" table.
7. Select the "TraceUserChange" system function.

Result

A system message is output when a user logs on or logs off.

8.4.3 Example: Structure of user administration (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Task

In the following example you set up a user administration for different users and user groups. The example orientates itself to a typical requirement profile from manufacturing engineering.

Principle

Completely different groups of persons are involved in a plant or project. Each group of persons protects its respective data and functions against access by others. For this purpose, users are created and assigned to a user group.

You can reproduce different views through user groups.

Example:

- Organizational view: Commissioners, Operators, Shift I, Shift II
- Technological view: Axis control, Tool changers, Plant North, Plant South

The following example orientates itself to the organizational view.

Every user group has characteristic requirements regarding access protection: A user group has operating authorizations for specific application cases. A programmer changes, for example, recipe data records.

In the example the users Miller, Group Smith and Foreman are created and assigned to different user groups.

Ms. Miller works as a programmer in the engineering system. The Group Smith are commissioners. Mr. Foreman is an operator.

Requirements

- A new project has been created.
- The "User administration" editor is open.

Procedures overview

Working with user administration has the following procedure in the example:

1. Creating authorizations The planner specifies which authorizations are required for access protection.
2. Configuring authorizations: The planner specifies which objects may be operated and which functions may be executed.
3. Creating user groups and allocating authorizations: The administrator creates the user groups together with the planner. The planner uses the authorizations to specify who may operate objects and change parameters.
4. Creating users and assigning them to a user group: The administrator administers the users.

Result

The aim is the following structure of the user administration of users, user groups and authorizations:

Users			User groups	Authorizations			
Miller	Smith	Foreman	Roles	Changing recipe records	Changing system parameters	Changing process parameters	Managing
			Administrator group				x
X			Programmer	X			
	X		Commissioning engineers	X	X	X	
		X	Operators	x			

The user "Foreman" who belongs to the "Operators" user group has access to the configured "To Recipe view" button.

Note

Alternatively, you can create several users as operators with different operating authorizations, for example, Operator Level 1, Operator Level 2.

8.4.4 Example: Creating and configuring authorizations (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Task

The following example shows you how to create the authorizations

Procedure

1. Open the "User groups" work area.
2. Double-click "Add" in the "Authorizations" table.
3. Enter "Change recipe data records" as the "Name" of the authorization.
4. Repeat steps 2 and 3 to create additional authorizations: "Change system parameters", "Change process parameters".

Result

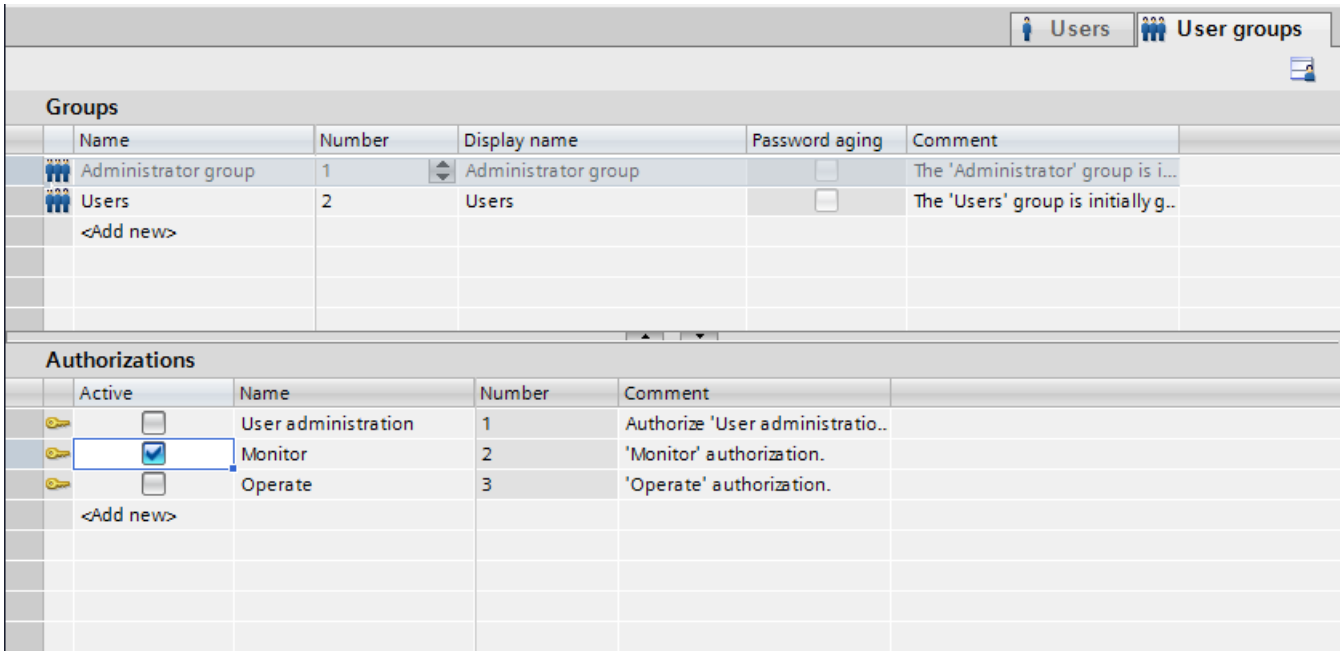

8.4.5 Example: Configuring a button with access protection (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Task

In the following example you use a system function to create a button for a screen change. You protect the "To Recipe view" button against unauthorized operation. To do so, you configure the "Change recipe data records" authorization at the "To Recipe view" button.

Requirements

- A "Change recipe data records" authorization has been created.
- A "Recipes" screen has been created.
- A "Start" screen has been created and opened.
- A button has been created and marked in the "Start" screen.

Procedure

1. Click "Properties > Properties > General" in the Inspector window.
2. Enter "To Recipe view" as the text.
3. Click "Properties > Events > Click" in the Inspector window.
4. Click the "Add function" entry in the first line of the "Function list" table.
5. Select the "ActivateScreen" system function in the "Screens" group.
6. Click the ... button in the "Screen name" field. A dialog box for selecting the screen opens.
7. Select the "Recipes" screen and use the button to close the dialog box.
8. Click "Properties > Properties > Security" in the Inspector window.
9. Select "Change recipe data records" as the "Authorization."

Result

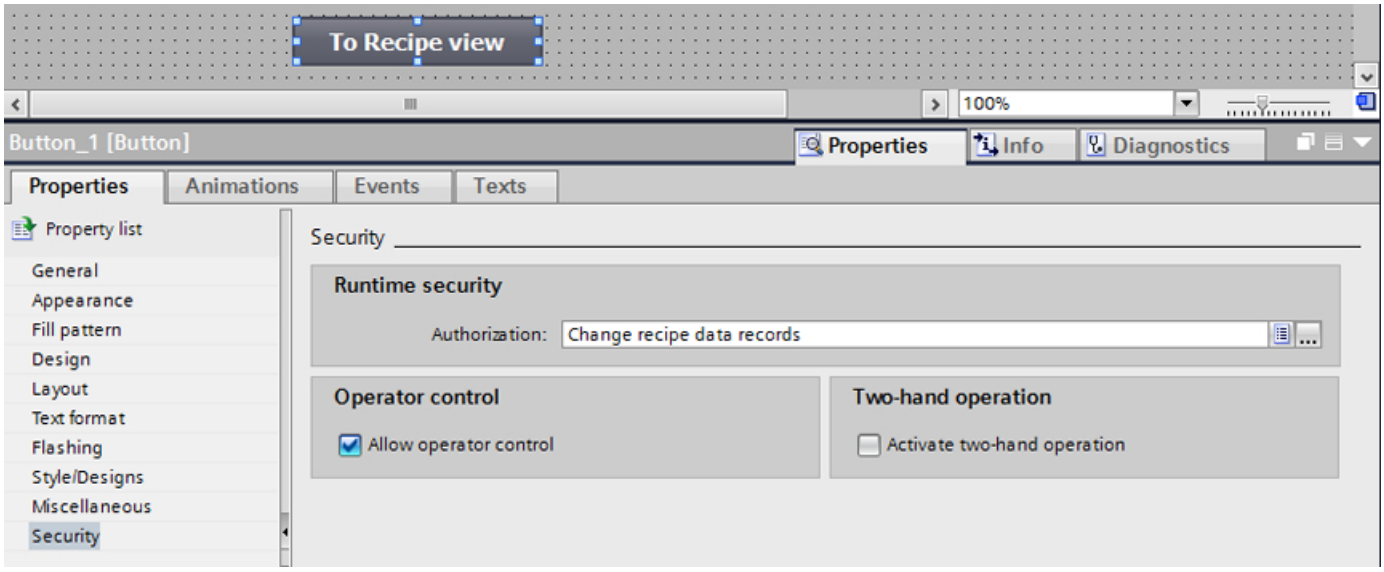

Access to the "To Recipe view" button is protected. If, for example, the user "Smith" clicks the button in Runtime, the function "Recipe view" screen is called up. Prerequisite is that the user "Smith" has logged on correctly and has the required authorization. The "Recipes" screen contains a recipe view and other screen objects.

Note

If the logged-on user does not have the required authorization or if no user is logged on, the "Logon dialog box" is displayed. An alarm indicating that no operator authorization is available for this user appears in Runtime Professional in this case.

8.4.6 Example: Creating user groups and assigning authorizations: (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Task

In the following example you create the user groups and assign authorizations to them.

Procedure

1. Open the "User groups" work area.
2. Double-click "Add" in the "Groups" table.
3. Enter "Programmer" as the "Name".
4. Repeat steps 2 and 3 to create the "Commissioner" and "Operator" user groups.

5. Click "Administrator" in the "Groups" table.
6. Activate the "Change system parameters" authorization in the "Authorizations" table.

Interim result

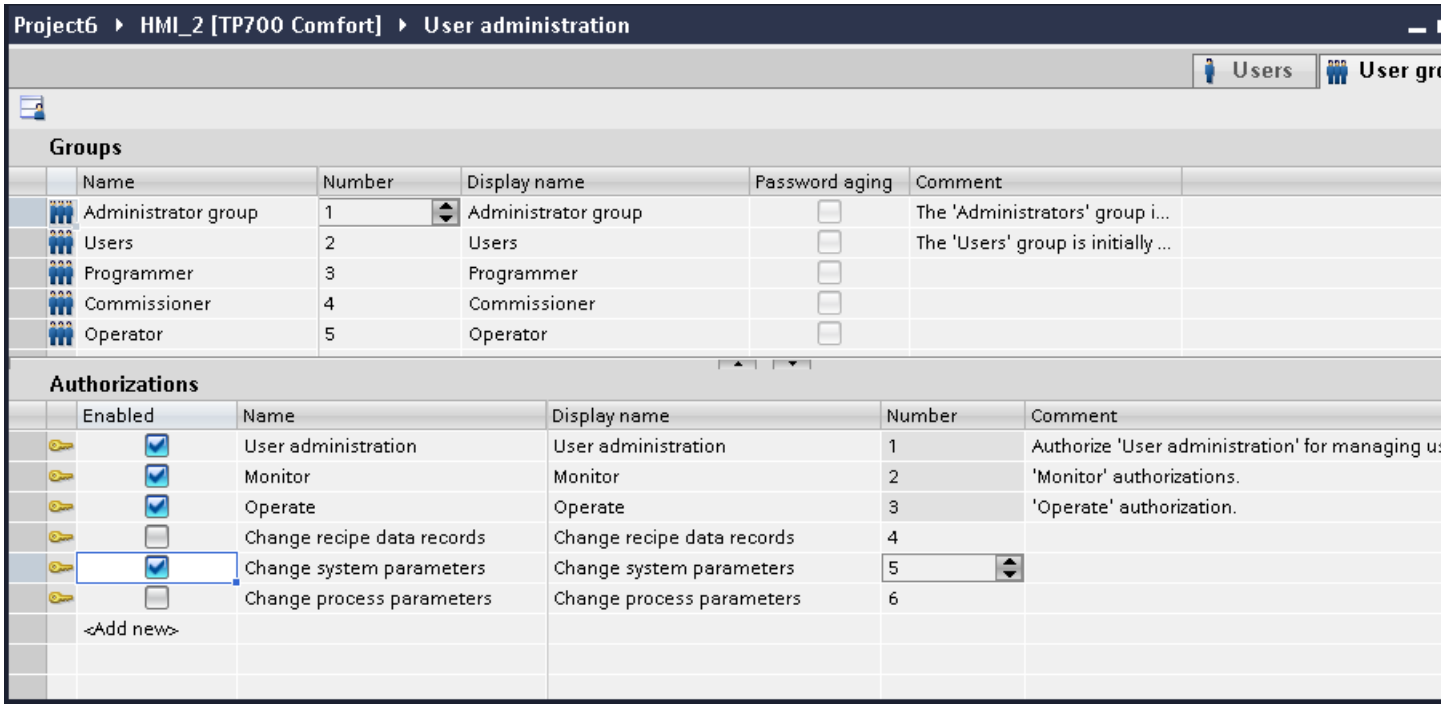

Procedure

1. Click "Operator" in the "Groups" table.
2. Activate the "Change recipe data records" authorization in the "Authorizations" table.
3. Click "Commissioner" in the "Groups" table.
4. Activate the authorizations "Change recipe data records", "Change system parameters" and "Change process parameters" in the "Authorizations" table.
5. Click "Programmer" in the "Groups" table.
6. Activate the "Change recipe data records" authorization in the "Authorizations" table.

Result

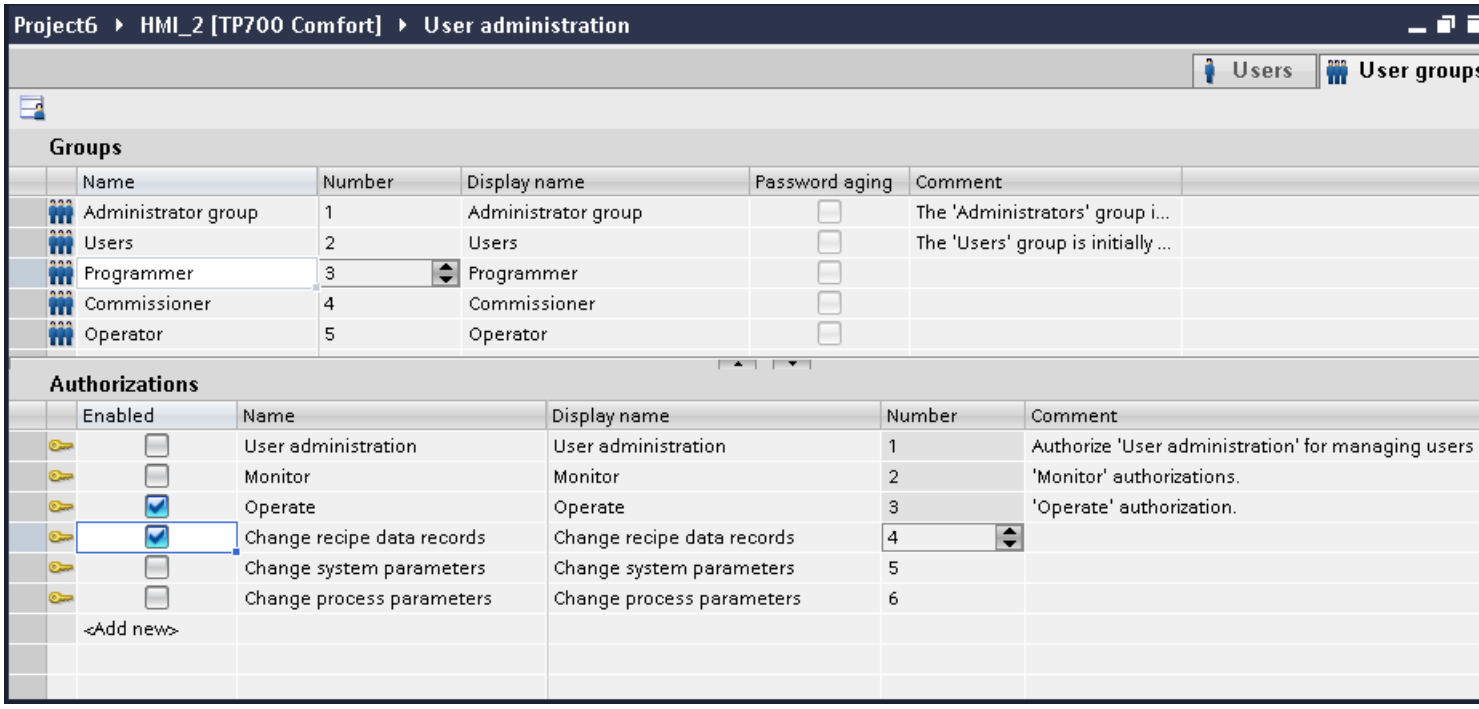

8.4.7 Example: Creating users and assigning them to a user group (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Task

In the following example you create the users and assign user groups to them. The users are sorted alphabetically immediately after the name has been entered.

Procedure

1. Open the "Users" work area.
2. Double-click "Add" in the "Users" table.
3. Enter "Miller" as the user name.
4. Click the button in the "Password" column. The dialog box for entering the password is displayed.
5. Enter "miller" as the password.
6. To confirm the password enter it a second time in the lower field.
7. Close the dialog box by using the icon.
8. Activate the "Programmer" user group in the "Groups" table.

Interim result

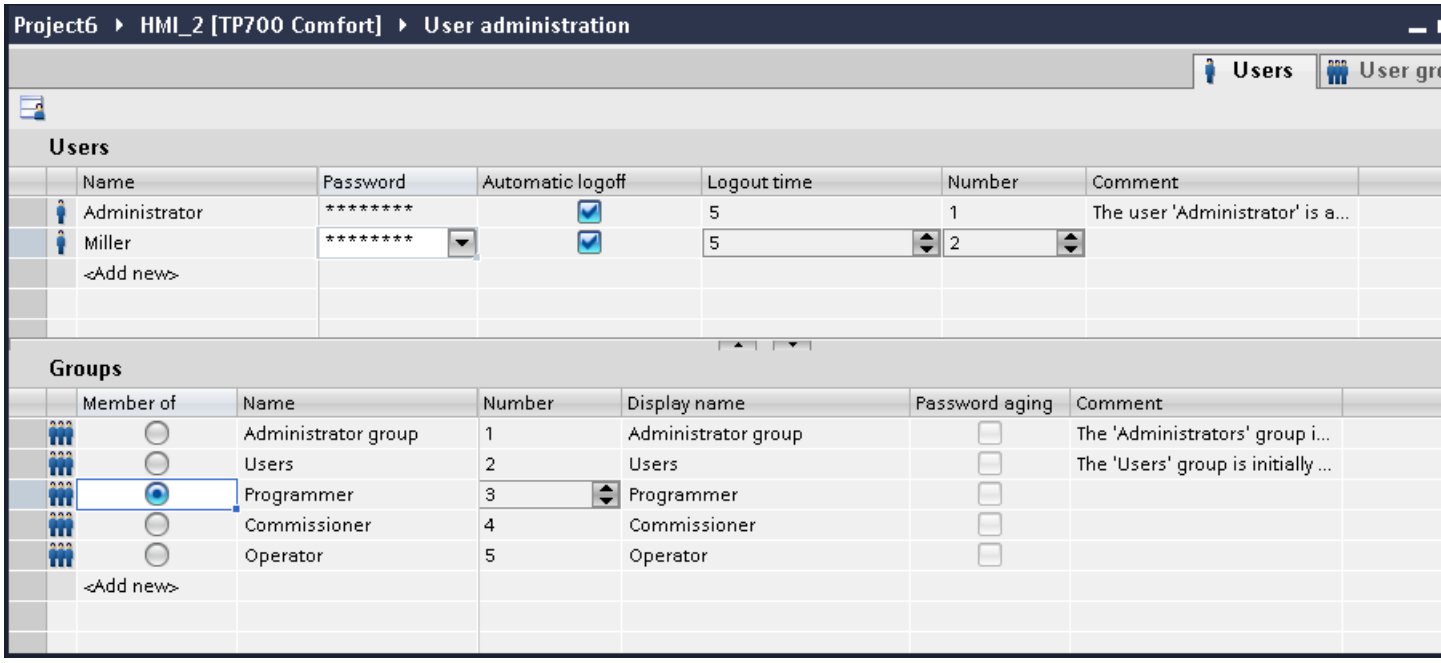

Procedure

1. Double-click "Add" in the "Users" table.
2. Enter "Smith" as the user name.
3. Click the button in the "Password" column. The dialog box for entering the password is displayed.
4. Enter "smith" as the password.
5. To confirm the password enter it a second time in the lower field.
6. Close the dialog box by using the icon.
7. Activate the "Commissioner" user group in the "Groups" table.
8. Repeat steps 2 to 6 for the user "Foreman."
9. Activate the "Operators" user group in the "Groups" table.

Result

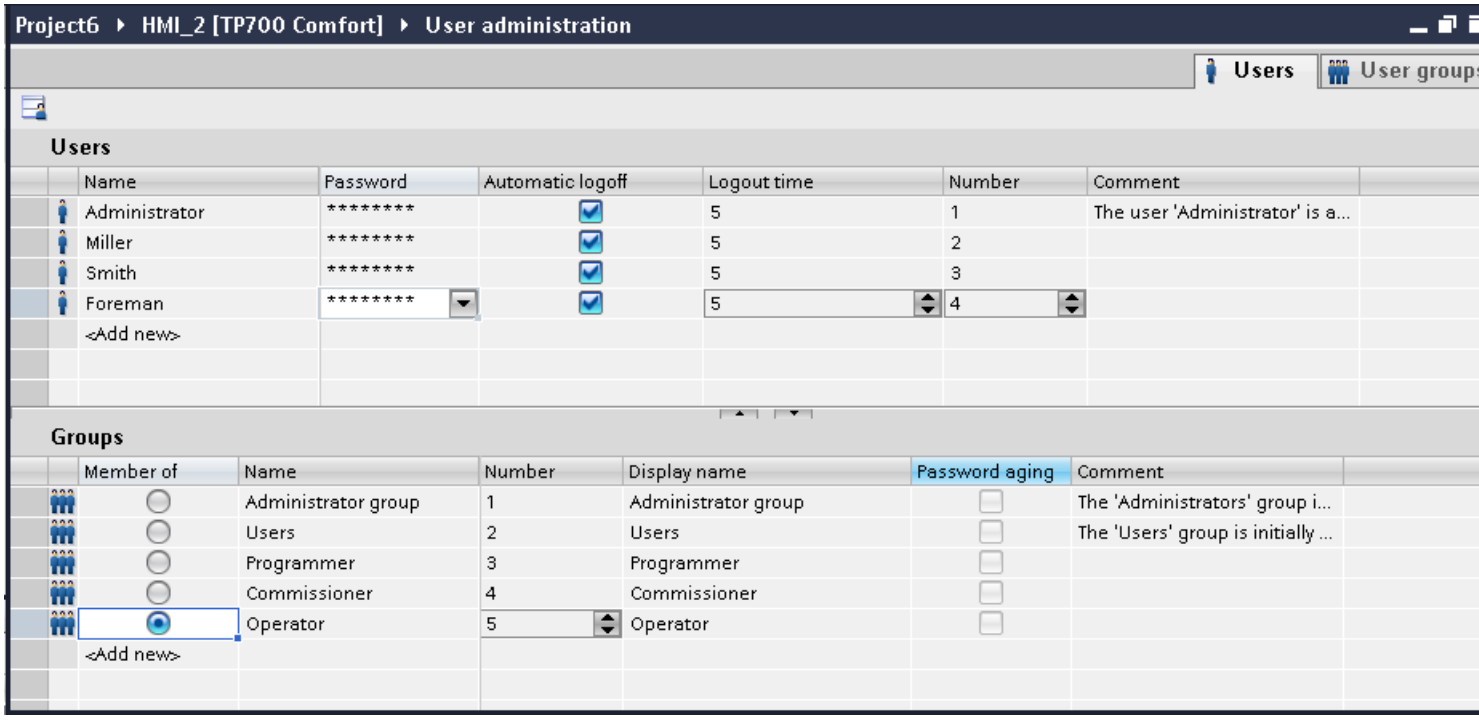
Project6 ▸ HMI_2 [TP700 Comfort] ▸ User administration
Users
User groups
Users
Name
Password
Automatic logoff
Logout time
Number
Comment
Administrator
****
5
1
The user 'Administrator' is a...
Miller
****
5
2
Smith
****
5
3
Foreman
****
5
4
<Add new>
Groups
Member of
Name
Number
Display name
Password aging
Comment
Administrator group
1
Administrator group
The 'Administrators' group i...
Users
2
Users
The 'Users' group is initially ...
Programmer
3
Programmer
Commissioner
4
Commissioner
Operator
5
Operator
<Add new>

Working with system functions and Runtime scripting 9

9.1 Principles

9.1.1 Runtime scripting (RT Professional, RT Advanced, Panels, Comfort Panels)

Term definition

The following terms are used in the documentation:

Term	
Runtime scripting	Generic term for all activities in user-defined functions and local scripts.
Function	Generic term for system functions and user-defined functions.
System functions	System functions are all functions supplied with WinCC. The system functions are used either in function lists, user-defined functions or local scripts.
User-defined functions	User-defined functions are functions written in the "Scripts" editor. For more precise specification, the term "User-defined VB function" and "User-defined C function" is used in the documentation.
Local script	A local script is created directly at the point of use, e.g. an object property, and has a cyclic or acyclic trigger. For more precise specification, the terms "local VB script" and "local C script" are used in the documentation.
VBS/VBScript	Abbreviation for Visual Basic Script

Availability

	WinCC RT Advanced and Panels	WinCC RT Professional
Customized VB functions	X	X
Customized C functions	--	X
Local VB scripts	--	X
Local C scripts	--	X

Applying Runtime scripting

The following figure serves as a decision guide for programming tasks at hand:

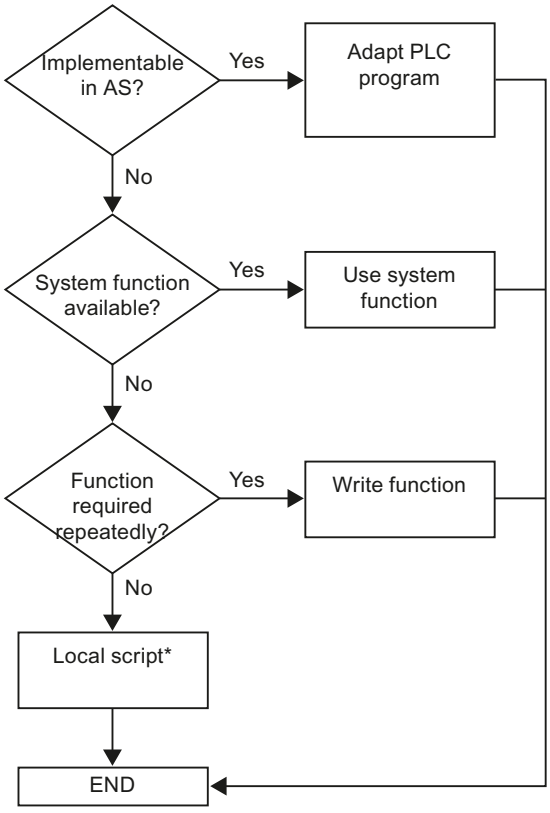

* Local scripts are not supported in WinCC RT Advanced and Panels.

Customized function or local script?

Whether you accomplish your programming task with a user-defined function or a local script depends on the application purpose and ultimately on the number of user-defined functions or local scripts.

Use local scripts for background activities, e.g. for daily printing or logs, for the monitoring of tags or for the execution of calculations.

Use customized functions to make codes available in several places in your project but only create them once. Instead of entering the code several times, you just call the appropriate customized function. Your code is clearer and easier to maintain.

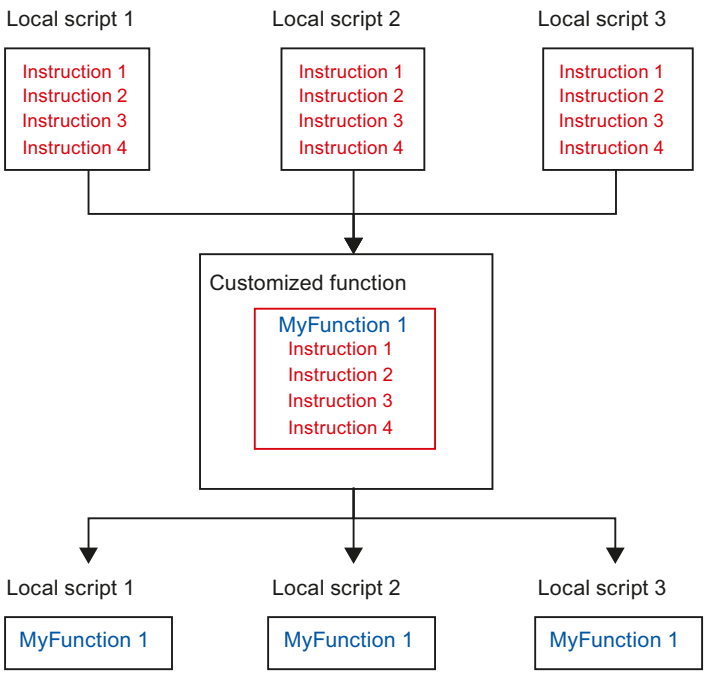

See also

User-defined functions (Page 1375)

Local scripts (Page 1377)

9.1.2 System functions (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Applications

- Function lists
  When the event occurs, the configured system functions can be performed with the function list.
- User-defined functions
  If the HMI device supports user-defined functions, you use system functions in combination with instructions and conditions in the code of the user-defined function. In this way you execute a customized function depending on a specific system state. In addition, you can evaluate return values of system functions, for example. Depending on the return value, you then perform test functions, for example, that in turn affect the function flow.

Application examples of system functions

- Calculations, increase tag value by a defined or variable amount.
- Log function, e.g. to start a process value log.

- Settings, e.g. to switch the PLC or set a bit in the PLC.
- Alarms, e.g after a different user logs on.

Application

You use system functions in a function list or in a user-defined function.

- Function list
  When configuring a function list, you select the system functions from a selection list that is sorted by categories:

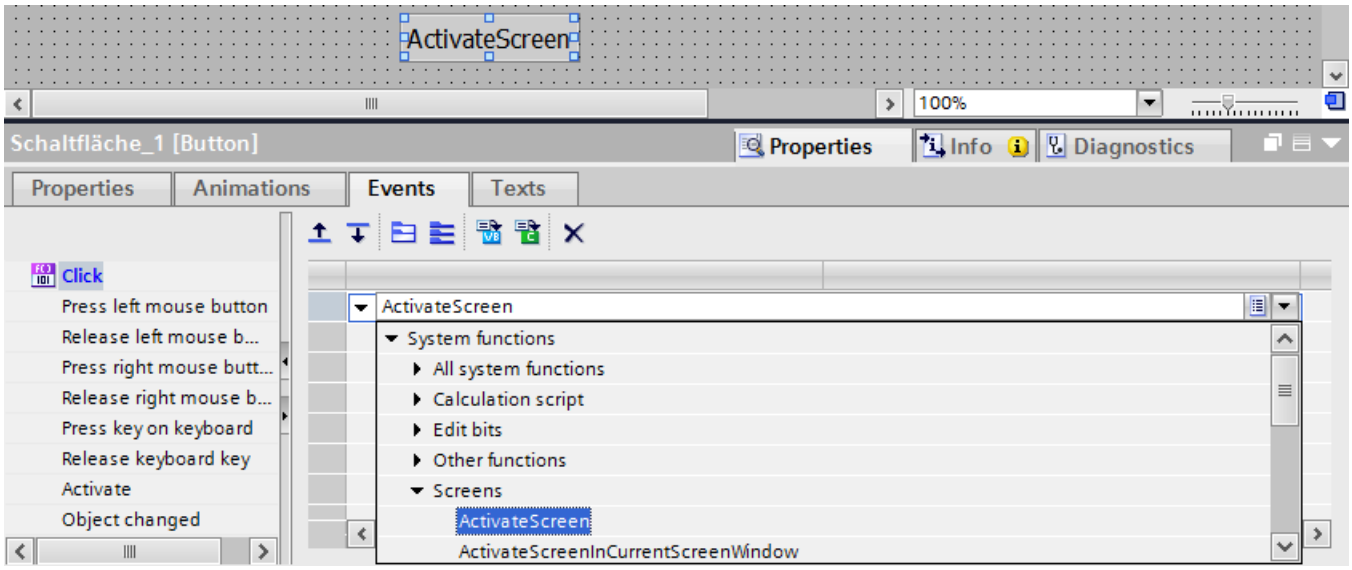

- User-defined functions
  If you are using a system function in a user-defined function, you choose it from a selection list. To open the selection list, use the key shortcut <Ctrl+Space> or click .

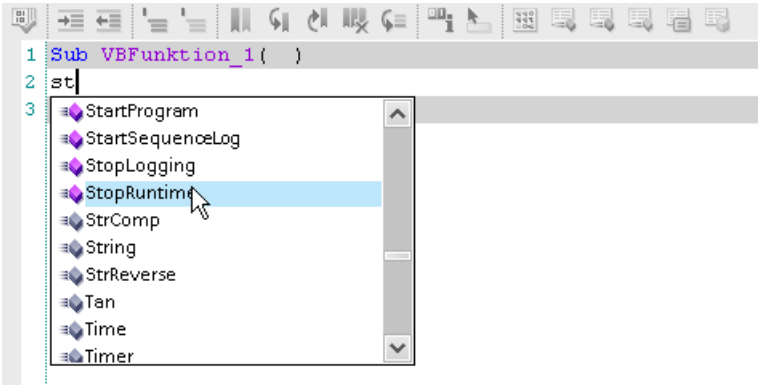

Language dependency

In the function list, the names of the system functions are dependent on the set project language. The functionality can then be recognized immediately by the project planner.

You use the English name of the system function in user-defined functions. You will find the English name of the system function in the reference.

Availability for WinCC Runtime Advanced and Panels

Which system functions are available depends on the selected HMI device.

HMI device changeover

If you use system functions in a function list that are not available on the set HMI device, these system functions are marked in color.

If you use a system function in a user-defined function which is not available on the set HMI device, a warning is issued. In addition, the respective system function will be underlined with a wavy blue line.

See also

System functions for logs (Page 626)

9.1.3 User-defined functions (RT Professional, RT Advanced, Panels, Comfort Panels)

Applications

You use customized functions for the following for example:

- Configuring a function list
  In a customized function you have the option to have customized functions and system functions executed depending on conditions or executed repeatedly. You then add the user-defined function to a function list, for example.
- Programming new functions
  Customized functions are only valid in the project in which they are defined. For user-defined functions you define transfer parameters and return values, for example, to convert values.

Note

You can only use user-defined functions of a programming language in a function list. Your selection of the first user-defined function determines whether VB functions or C functions can be selected in the function list.

Application examples of customized functions

- Convert values
  You use functions to convert values between different measurement units, e.g., temperatures.
- Evaluating tag values statistically
- Drawing a trend in a f(t) trend view

Properties of customized functions

A customized function has the following properties:

- Name
- Parameter (optional)
- Return value and type (optional)
- Can be modified centrally
- has no trigger. You have to call the user-defined function explicitly, e.g. in a function list.

Organization of customized functions

You create customized functions with the "Scripts" editor. For more information, refer to ""Scripts" editor (Page 1384)".

Customized functions are saved in the project. To protect user-defined functions, set up know-how protection. See Setting up know-how protection (Page 1416) for additional information.

- Project tree
  The user-defined functions are listed in the project tree under "VB scripts" or "C scripts".

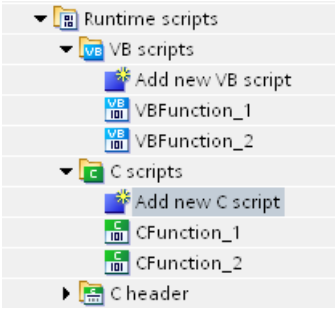

- Function list
  The user-defined functions are listed in a function list under "VB functions" or "C functions".
- User-defined function
  If you are using a system function in a user-defined function, you choose it from a selection list. To open the selection list, use the key shortcut <Ctrl+Space> or click .

See also

Runtime scripting (Page 1371)

Setting up know-how protection (Page 1416)

"Scripts" editor (Page 1384)

9.1.4 Local scripts (RT Professional)

Applications

You can use local scripts for the following:

- Dynamizing objects
  You use a local script to determine the value of an object property in Runtime. Use this approach, for example, to change the color depending on a tag value.
- Performing tasks at any point in time.
  You use a local script to plan jobs, e.g. daily printout of a log at the end of the shift.

Application examples of local scripts

- Configuring desired values for tags for operator control of a graphic object, e.g., in order to assign a value for the PLC via a mouse click.
- Configuring the Runtime language switch for operator control of a graphic object.
- Status monitoring of tags, e.g. to ensure that a value is valid.

Properties of local scripts

You create a local script directly at the point of use. A local script has the following properties:

- Predefined name
- Predefined parameters
- Predefined return value (optional)
- Can only be changed at the point of use
- has no trigger
- Calling customized functions, system functions

Triggers

Triggers are necessary to execute local scripts in Runtime. The trigger is either determined automatically when creating a local script or specified by the project manager.

The following triggers are supported by WinCC:

- Acyclic triggers, e.g. you plan a one-time stop of Runtime for maintenance work.
- Cyclic triggers, e.g. you plan a job which starts daily printing of a log.
- Events, e.g. when clicking the button.

The following figure shows different types of triggers:

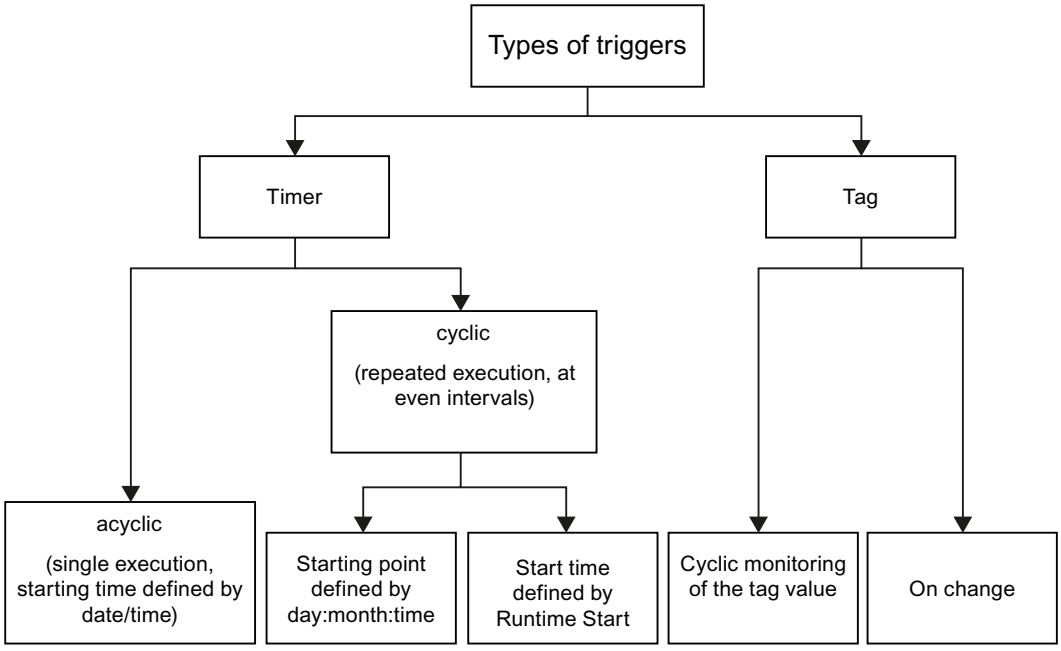

Triggers are used in the following editors:

- Scheduler
  First you need to configure the trigger in the Scheduler. Then connect the trigger to the local script or a function list.
- Screens
  To dynamize an object property, first configure a local script in the property list of the screen. The trigger is determined automatically when creating the local script. You can change the trigger, e.g. to tag trigger.

See also

Runtime scripting (Page 1371)

9.2 Working with Function Lists

9.2.1 Basics of the function list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

A function list performs one or more system functions and user-defined functions when the configured event occurs.

Principle

The function list is configured for an event of an object, for example, for a screen object or a task. You can configure a function list precisely on every event. The events which are available depend on the selected object and the HMI device.

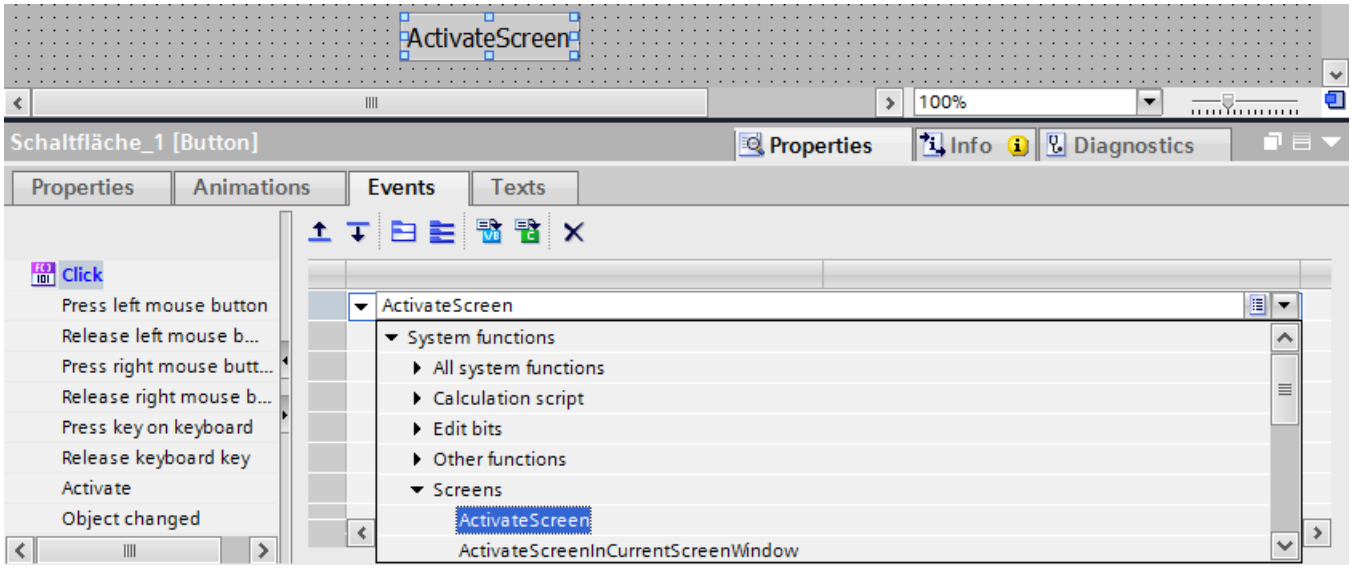

Events occur only when the project is in Runtime. Events include:

- Execution of a task
- Pressing of a button
- Acknowledging an alarm

Note

The choice of configurable system functions in a function list is dependent on the selected HMI device. For additional information, refer to "Availability for specific devices of system functions".

See also

Function list (Page 1799)

9.2.2 Properties of a function list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Status information

During configuration the project data is tested in the background.

With the following causes the function list is not executed in Runtime and the incorrect entries are marked red:

- At least one system function or one user-defined function is not completely supplied with parameters.
- At least one system function or one user-defined function is contained which is not supported by the selected HMI device, e.g. by changing the device type.

Processing of system functions and user-defined functions (WinCC Runtime Advanced and Panels)

A function list is executed from top to bottom in Runtime. If a function list includes system functions with longer runtime, these are processed asynchronously. You can find additional information about this in the sections "Calling system functions (Page 1387)" and "Calling user-defined functions (Page 1388)".

See also

Calling system functions (Page 1387)

Calling customized functions (Page 1388)

9.2.3 Configuring a function list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You configure a function list by selecting the system functions and customized functions from a drop-down list. The system functions are arranged in the selection list according to categories.

You only use customized functions of a programming language in a function list. Your selection of the first user-defined function determines whether user-defined VB functions or user-defined C functions can be selected in the function list. Which programming languages are available depends on the selected HMI device.

If you have created a user-defined VB function, for example, the user-defined function is available under the entry "VB functions".

Requirement

Object has at least one configurable event.

Procedure

Proceed as follows to configure a function list:

1. Open the editor in which the object is located.
2. Select the object.
3. Click "Properties > Events" in the Inspector window. Choose the event for which you want to configure the function list.
4. Mark the "<Add function>" entry in the inspector window in the drop-down list.
5. Select the desired system function or the desired user-defined function from the selection list. You can also enter the name of the system function or the user-defined function.

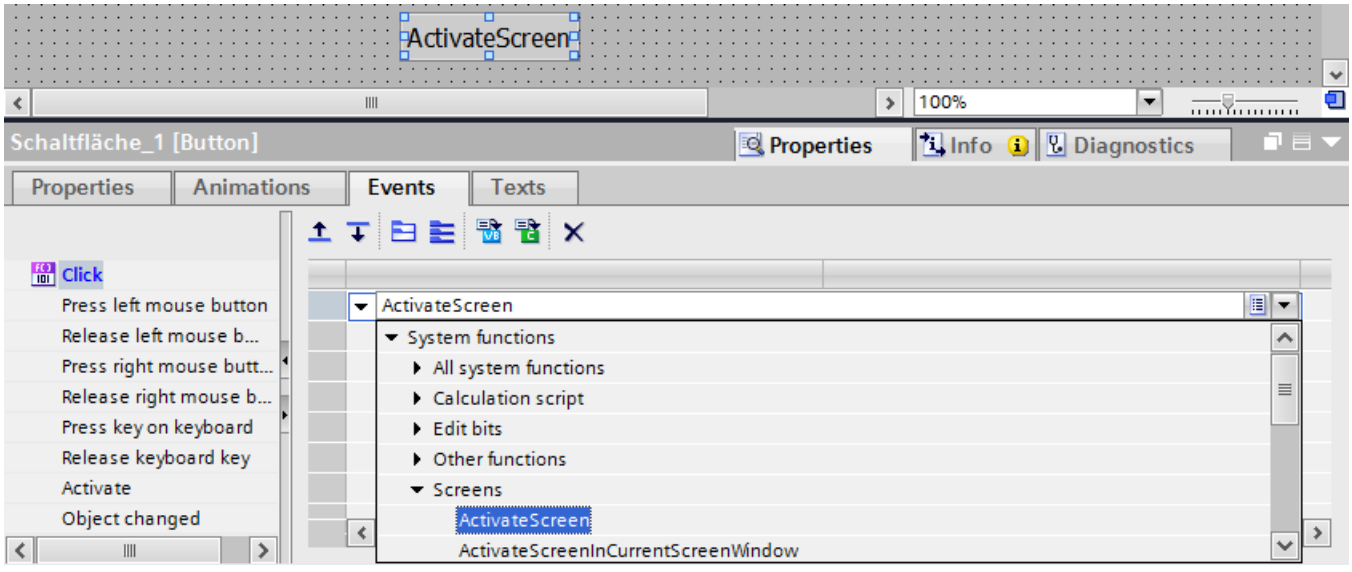

The system function or the customized function is entered in the function list.

6. If the system function or the customized function has parameters, select the appropriate values for the parameters.

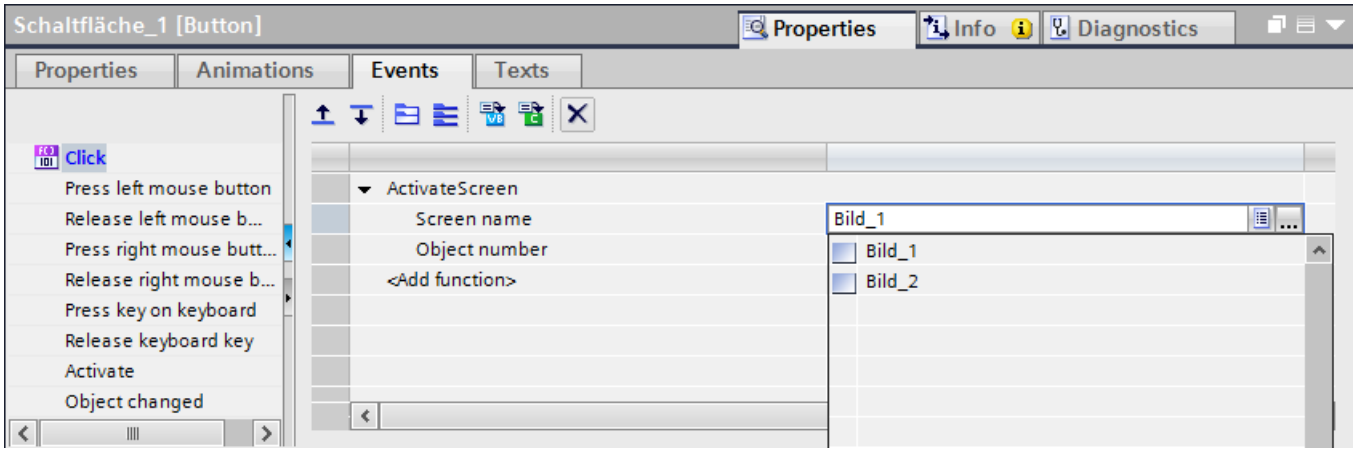

7. If you want to add other system functions or user-defined functions to the function list, repeat steps 4) to 6).

Result

The function list is configured. In addition, to the configured event, the status of the function list is displayed in the Inspector window. When the configured event occurs in Runtime, the function list is completed from top to bottom.

See also

Example: Changing the operating mode on the HMI device with the current display (Page 1447)

9.2.4 Editing a function list (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

A function list can be edited as follows:

- Change the order of system functions and user-defined functions.
- Remove system functions or user-defined functions.

Requirement

The function list is configured and opened.

Changing the order of a system function or a customized function

1. Select the desired system function or customized function in the function list.
2. Then click the appropriate direction arrow in the inspector window until the system function or customized function is in the desired position.

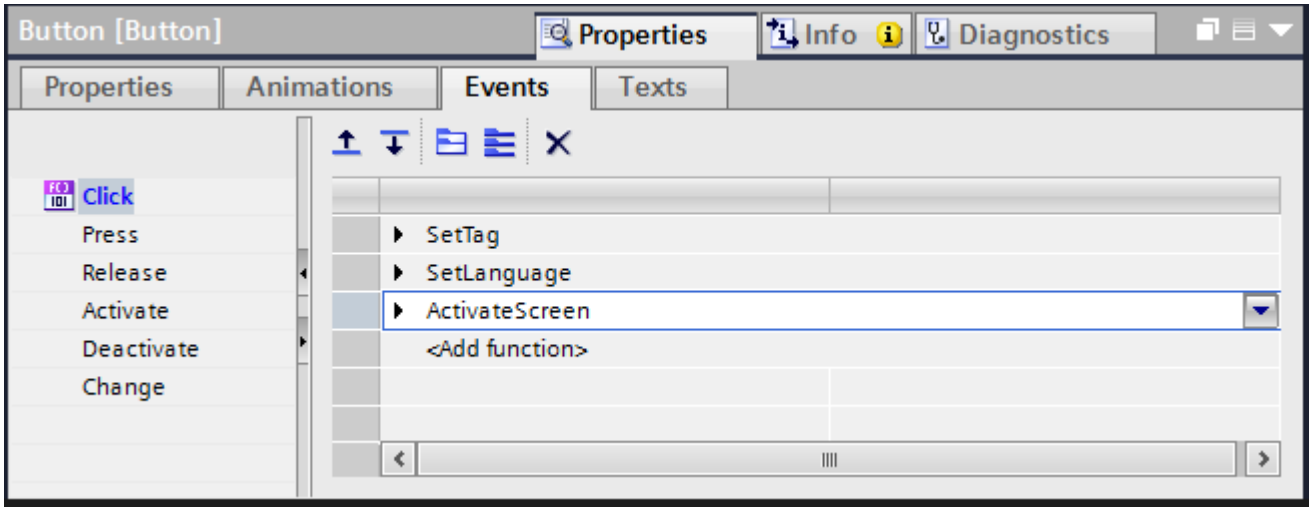

Changing the order of several system functions and customized functions

1. Hold down the <Shift> key.
2. Click desired system functions or customized functions one after another with the mouse.
3. Move the selection to the desired position by drag&drop.

Removing a system function or customized function

1. Select the desired system function or customized function in the function list.
2. Select "Delete" in the shortcut menu.

9.3 Working with customized functions

9.3.1 "Scripts" editor (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

The "Scripts" editor supports you in the creating of user-defined functions with functionalities such as auto-completion and syntax highlighting. You can insert code templates for frequently used instructions with the "Instructions" task card for example.

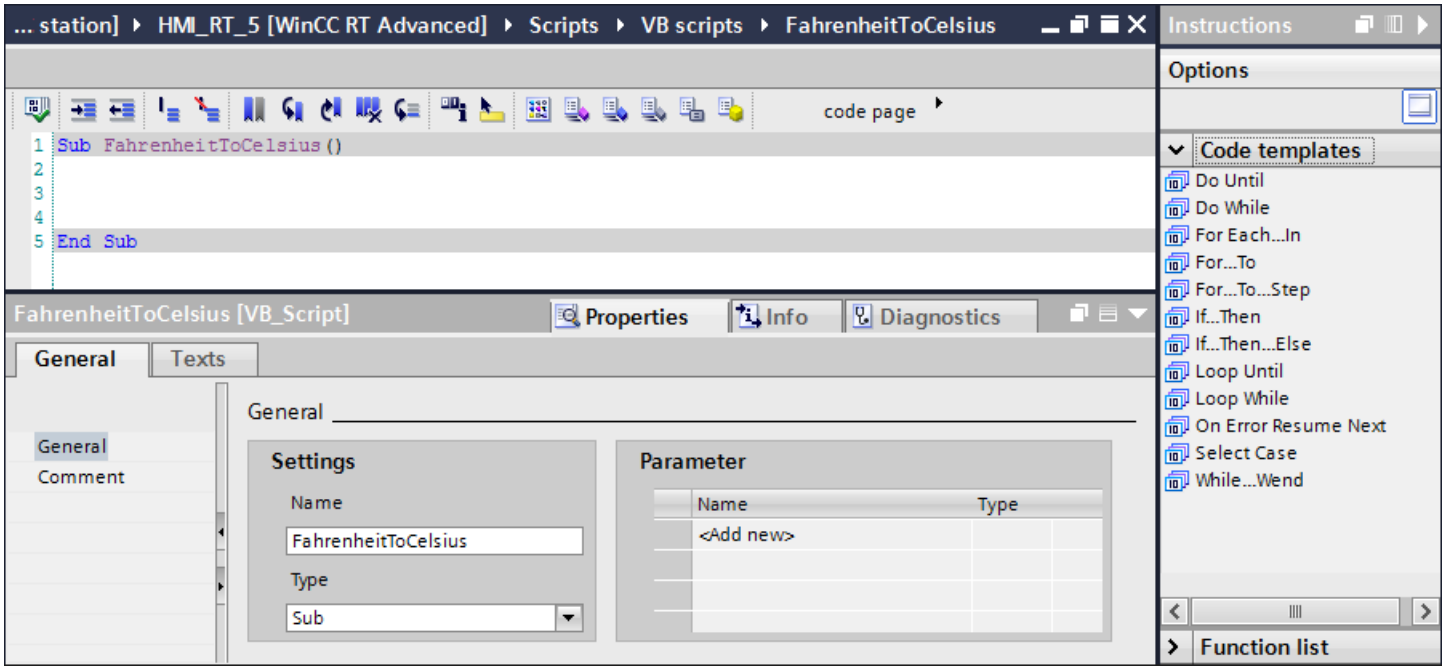

Autocompletion and parameter entry

The "Scripts" editor provides you context-dependent programming support with autocompletion parameter entry support.

When you use system functions, the parameters of the function are shown as a tooltip.

<Ctrl+J>

The object list can be called context-specifically by using the shortcut <Ctrl+J>. Use the object list to supply values to system functions, methods, and properties: For example, you select screens, screen objects, tags, or colors. You transfer a selected object from the object list into the code by double-clicking it.

Note

Tags and screens can also be inserted using a drag-and-drop operation. Use, for example, drag-and-drop to assign values to parameters.

<Ctrl+Space>

With the key combination <Ctrl+Space> you call a list with the following contents:

- Constants and functions of the programming language used in the "Scripts" editor.
- User-defined functions
- System functions

Integrating control characters in strings

If you want to use control characters such as tabs or line breaks in strings, use the function "Chr()". Transfer the decimal ASCII code of the required control character as parameter:

Example:

- Chr(9) returns "Tab".
- Chr(15) returns "SI" (Shift in).
- Chr(20) returns "DC4" (Device Control 4).

You will find more detailed information on this VBS function on the Internet (https://msdn.microsoft.com/en-us/library/ws6aa3sf(v=vs.84).aspx).

"Instructions" task card

The "Instructions" task card contains the "Code templates" and "Function list" palettes.

- Code templates
  To insert frequently used instructions, double click the desired instruction in the "Code templates" palette. Replace the space holders with the used expressions.
- Function list
  In the function list you select the system functions and user-defined functions from a drop-down list. Proceed as for configuring a function list. To transfer the function list to the code, click the "Transfer" button. The function list is automatically converted into the correct syntax.

  Note

  Not all system functions are available in the "Scripts" editor. For additional information, refer to the reference.

Edit functions

Use the buttons of the toolbar "Advanced Edit" to make the code more readable by means of indentation and the use of comments.

Bookmarks help you keep an overview, even of many lines of code.

Further information regarding each button can be obtained in the accompanying direct help.

Synchronization of tags and objects

If you change tag names or object names in the project which you use in a user-defined function, the names are adapted automatically in the user-defined function.

If you change the tag name or object name in the function, the tag names or object names are not adapted in the project. An appropriate error message is output when executing the user-defined function.

Syntax highlighting

Keywords in the "Scripts" editor are highlighted using different colors. Unknown words are underlined with a red wavy line.

The table shows the preset colors for the most important keywords:

Color	Meaning	Example
Blue	Keyword	If, Then
Dark violet	Function	FahrenheitToCelsius
Chocolate	System function	SetTag
Orange	HMI tag	Tag_1
Green	Comment	'This is a comment'

Adapting display properties

When you have opened a user-defined function, you can customize the display properties in the editor. Select the "Options > Settings" menu command. Click "General > Text editors" in the "Settings" editor. For example, you can change the colors for syntax highlighting or adapt the indents for code in the work area.

See also

Create customized VB functions (Page 1398)

Create customized C functions (Page 1409)

Local scripts (Page 1419)

Editor for local scripts (Page 1420)

9.3.2 Calling system functions (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can call system functions in a customized function. The syntax depends on the used programming language. Which programming language is supported depends on the selected HMI device.

You have the following options for calling a system function in a customized function:

- With <Ctrl+Space> or with
- Direct input
- With the "Function list" palette

With <Ctrl+Space> or with

Open the selection list with <Ctrl+Space> and select the desired system function.

Direct input

You enter the system functions directly into the code. You use the English name of the system function. You can find the English name of the system function in the system function reference under "Syntax". The editing language setting is not taken into consideration.

References to objects, e.g. screens, connections and logs are transferred in inverted commas:

- Calling system functions without return value in VBS

```
SetTag "Tag1",64
```

- Calling system functions with return value in VBS

```
InverseLinearScaling "Xvalue","Tag1", 0, 1
```

- Calling system functions without return value in C

```
SetTag ("Tag1",64);
```

- Calling system functions with return value in C

```
InverseLinearScaling ("Xvalue","Tag1", 0, 1);
```

With the "Function list" palette

Select the desired system function from a drop-down list in the "Function list" palette. Proceed as for configuring a function list. You will find further information about this under "Configuring a function list".

To transfer the list to the code, click the "Transfer" button. The list is automatically converted into the correct syntax.

HMI device replacement (Runtime Panels and Advanced)

The code of a customized function depends on the selected HMI device. If you use system functions in the customized function which are not supported by the selected HMI device, you receive an error message in the output window.

See also

Configuring a function list (Page 1380)

Create customized VB functions (Page 1398)

Executing customized VB functions (Page 1400)

Executing customized C functions (Page 1413)

9.3.3 Calling customized functions (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can insert user-defined functions of the same programming language in a user-defined function, e.g. only user-defined C functions in a user-defined C function.

The syntax depends on the used programming language. Which programming language is supported depends on the selected HMI device.

You have the following options for calling a customized function in a customized function:

- With <Ctrl+Space> or with
- Direct input
- With the "Function list" palette

With <Ctrl+Space> or with

Open the selection list with <Ctrl+Space> or with and select the desired user-defined function.

Direct input

You enter the customized function directly into the code. References to objects, e.g. screens, connections and logs are transferred in inverted commas.

Calling customized functions without return value in VBS

```
Average 4,10
```

Calling customized functions with return value in VBS

```
Dim ValueA
ValueA = Average (4,10)
```

If you do not want to evaluate the return value, use the call as for a customized function without return value.

Calling customized functions without return value in C

```
Average(4,10);
```

Calling customized functions with return value in C

```
SetTagDouble("AverageValue",Average (4, 10));
```

If you do not want to evaluate the return value, use the call as for a customized function without return value.

With the "Function list" palette

Select the desired customized function from a drop-down list in the "Function list" palette.

Proceed as for configuring a function list. For more information, refer to "Configuring a function list (Page 1380)".

To transfer the list to the code, click the "Transfer" button. The list is automatically converted into the correct syntax.

See also

Create customized VB functions (Page 1398)

Executing customized VB functions (Page 1400)

Executing customized C functions (Page 1413)

Configuring a function list (Page 1380)

9.3.4 Access to HMI tags (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

With VBS or C you can access the HMI tags which you have created in the project. Change or read the value of an HMI tag with a user-defined function in Runtime.

You can also create local tags as counters or buffers. Every local tag must be initialized to prevent errors by writing local tags incorrectly.

HMI tags

- WinCC Runtime Advanced and Panels
  The tag value which is created in the tag map is saved in the user-defined function. Next, the tag value will be updated to the set cycle time. The customized function first accesses the tag values which were read at the previous cycle time.
  If the tag name matches the VBS name conventions in the project, you can use the tag in the user-defined function as follows.
  ```
  'VBS_Example_03
   If BeltDriveOilTemperature > 100 Then [instruction]
  ```
  If the tag name in the project does not match the VBS name conventions, you must reference the tag with the "SmartTags" list. In the following example, the tag name contains the "&" character, which is not allowed according to the VBS naming conventions.
  ```
  'VBS_Example_04
  SmartTags("Test&Trial")= 2005
  ```
- WinCC Runtime Professional
  You have the same access possibilities as in the HMI devices with WinCC Runtime Advanced and Panels. You can also access the tag values asynchronously. The tag value in the PLC is accessed directly in asynchronous access. You access the tag value as follows in a local script or a user-defined C function:
  - SmartTags list (VBS)
  - Tags listing with Read-/Write method (VBS)
  - GetTag function (C)
  - SetTag function (C)

  The tag value is read once from the PLC when the method or user-defined function is called. The tag value is not updated regularly, in contrast to when it is read from the tag image.

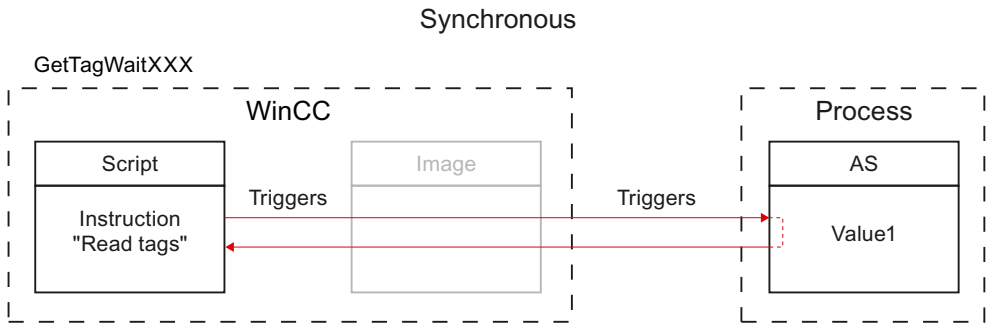

Figure 9-1 Synchronous: Tag value directly from the PLC

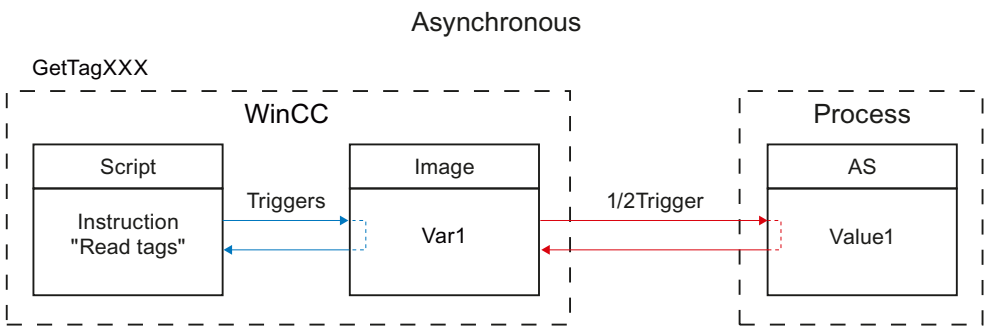

Figure 9-2 Asynchronous: Tag value from the tag image

Note

Runtime scripting

The index of the PLC array elements can begin with any number. In WinCC, indexing always starts with 0.

A PLC tag "Array [1..3] of Int", for example, is mapped to "Array [0..2] of Int" in WinCC.

When you access an array in a script, pay attention to the correct indexing.

Local tags

You can only use local tags within user-defined functions or local scripts.

You use local tags for the following:

- Buffers
- Counters, for example For instruction

Initialize every tag according to the programming language syntax.

Example: Dim objTagFillLevel

Initialize every tag with a value to avoid errors due to empty HMI tags. To use the value of the local tag outside, link it with an internal or external HMI tag.

Access to HMI tags with dynamic names (WinCC Runtime Advanced and Panels)

The user-defined VB function accesses the tag value using the tag name. You can specify the tag name in such a way that the tag name is composed during the runtime of the user-defined function.

If you only call the user-defined VB function in a screen in which the HMI tag is also used elsewhere, for example in an I/O field, you should for performance reasons configure the HMI tag with "Cyclic in operation" acquisition mode.

If the user-defined VB function is accessed and the HMI tag is not being used in the screen currently displayed, configure the HMI tag with "Cyclic continuous" acquisition mode. This ensures that the current value of the tag is always available.

9.3.5 Testing the syntax of customized functions (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

During programming, the code is tested in the background. Syntactical errors are marked with a wavy red line. The customized function is checked for correct syntax and valid object references.

Check the syntax in the "Scripts" editor to identify all the errors in the code and output the appropriate error messages. In this case, error messages generated by the script parser are output.

Use a separate debugger to check customized VB functions for logical programming errors.

Error types

The following error types are detected in the syntax test and output in the "output window":

- Unknown string (e.g., not a keyword)
- Assignment of value to a constant
- The name of a local tag created with the Dim instruction is already known, e.g.: as a parameter, object, or component of the object model
- Syntax error

Requirements

- Customized function is open.

Procedure

Proceed as follows to check the syntax:

1. Click to start the syntax check.
   The syntax is checked. The result of the syntax check is displayed in the Inspector window under "Info > Compilation." Syntax errors are output with the line number.
2. Correct the errors in the customized function if necessary.

Special consideration in the syntax checking of customized functions

Customized functions are not interpreted until in Runtime. When an HMI device is being compiled, the system checks its user-defined functions for correct syntax. However, runtime errors may occur in certain circumstances.

9.4 Creating customized VB functions

9.4.1 Accessing objects with VBS (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

The objects of the VBS object model with the appropriate properties and methods are available in user-defined VB functions.

The object properties can be read and changed in Runtime.

Referencing objects

In customized VB functions you reference the objects by the corresponding list. To identify the object, use its name or the position number within the list.

If you access the properties of an object more often, create an object reference. You can access object properties with and without object reference.

With the following instruction the first object is referenced in the "MainScreen":

```
'VBS_Example_01
Dim objObject
'Change to Screen "MainScreen"
HMIRuntime.BaseScreenName = "MainScreen"
Set objObject = HMIRuntime.Screens(1).ScreenItems(1)
```

With the following instruction an object is referenced by its name and an object property is changed. You must have created the object with this name in the screen.

```
'VBS_Example_02
Dim objCircle
HMIRuntime.BaseScreenName = "MainScreen"
Set objCircle = HMIRuntime.Screens(1).ScreenItems("Circle_01")
objCircle.BackColor = vbGreen
```

9.4.2 Use of VB tags (RT Professional)

Definition of VB tags

VB tags are declared by inserting definition lines in front of the function name of a user-defined VB function.

```
Dim objScreen
Dim objScrItem
```

The validity range of a VB tag depends on where it was declared.

User-defined VB functions

In the "Scripts" editor, lines 1 and 2 are by default the declaration range. You can add any number of lines.

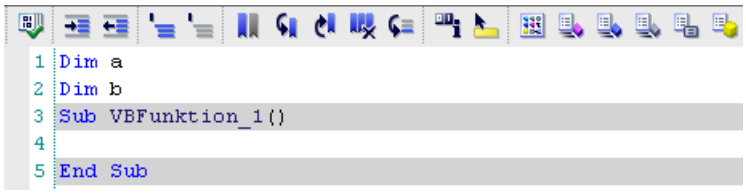

You can use a VB tag defined in this manner in user-defined VB functions. When you check the syntax in the "Scripts" editor, you receive a message that the VB tag is not known.

The VB tag is only known in Runtime. It is created as soon as Runtime is started, even if the user-defined VB function itself is not called.

To prevent inconsistencies, the VB tags must be unique within the user-defined VB functions.

Local VB scripts

The declaration area of a local script belongs to the screen. All scripts can access the tags declared in this area within the screen . Whenever a local script is used to copy a screen object to a different screen or to the library, the script but not the declaration area is copied also.

You can show and hide the declaration range in the editor for local VB scripts. For more information, refer to "Editor for local scripts (Page 1420)".

- Local VB scripts with acyclic trigger
  Every VB tag defined in this manner is automatically transferred to the declaration range of every local script with acyclic trigger. The declaration range remains empty for local scripts with cyclic trigger.
- Local VB scripts with cyclic trigger
  Every VB tag defined in this manner is automatically transferred to the declaration range of every local VB script with cyclic trigger. The declaration range remains empty for local scripts with acyclic trigger.

Example

A local VB script is configured for the property "Background color". The local VB script has a cyclic trigger.

The VB tags a and b are declared in the declaration range.

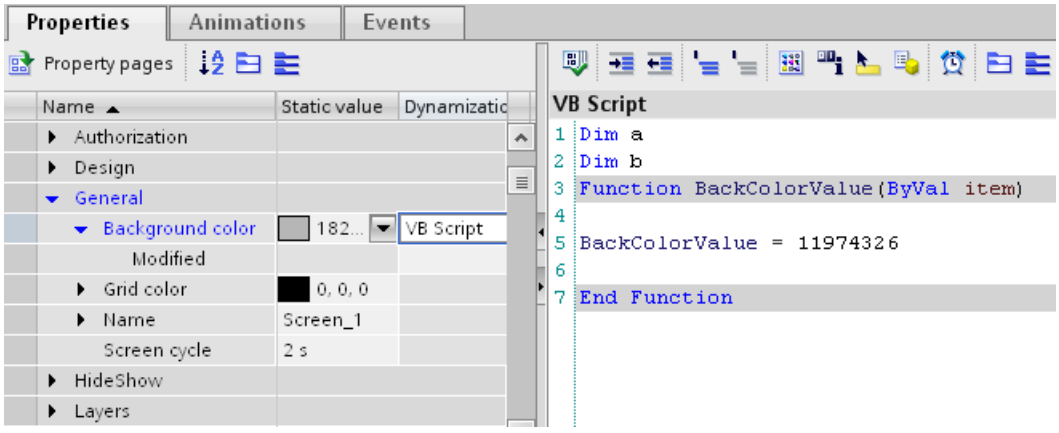

When you configure a local VB script for the "Changed" event, this VB script has an acyclic trigger. The declaration range is empty.

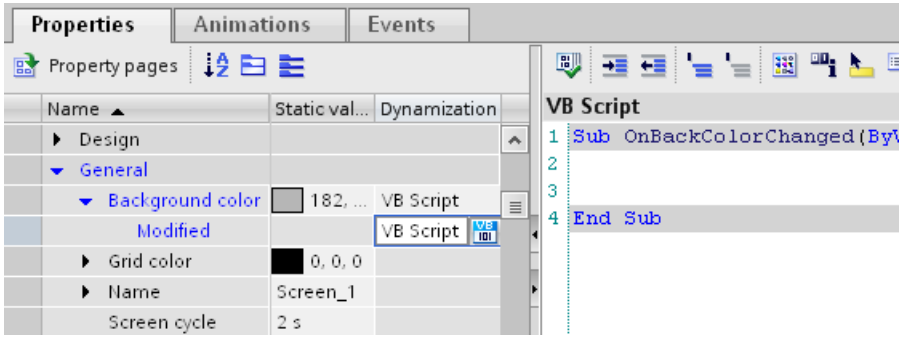

When you configure a local VB script with cyclic trigger for a different property, the VB tags a and b are already declared in the declaration range.

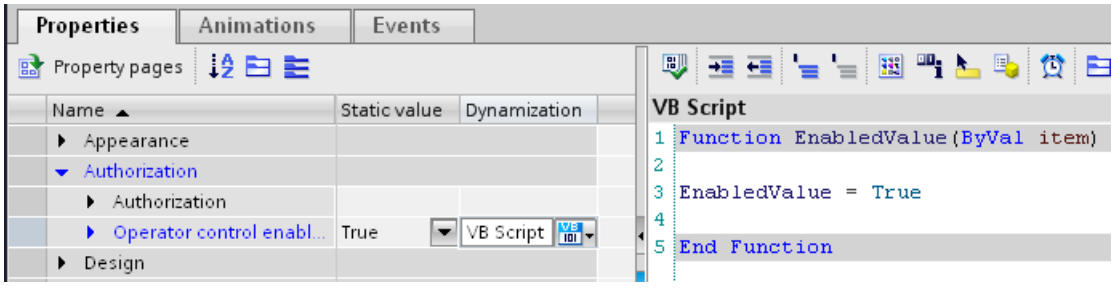

See also

Editor for local scripts (Page 1420)

9.4.3 Create customized VB functions (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

When you create a customized VB function, you define the following settings:

- The name with which you call the user-defined VB function.
- The type of user-defined VB function.
- The parameters which are transferred to the user-defined VB function in Runtime.

Use only the following characters for the name:

- "A" to "Z"
- "a" to "z"
- "0" to "9"
- "_" as a separator

Note

When you change the parameters or the function type, the change is color marked at the application point, e.g. in the function list.

Procedure

To create a user-defined VB function, follow these steps:

1. Open "Scripts" in the project tree.
2. Double click "Add new VB function" in the project tree under "VB scripts".

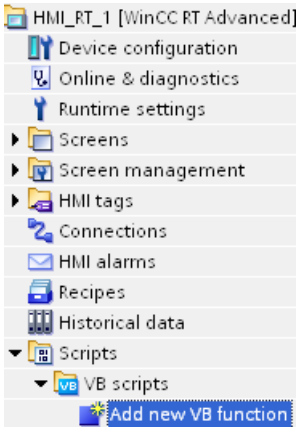

The user-defined VB function is created as a new tab in the work area. The input mask for the configuration settings of the user-defined VB function opens in the Inspector window.

3. Configure the user-defined VB function in the Inspector window "Properties > General > General".
4. Enter a descriptive "Name" for the user-defined VB function.

5. Select the "Type" of the user-defined VB function.
   - Function" have a return value.
   - Sub" have no return value.
6. Click "<Add>" in the "Parameters" list to add parameters. Enter the parameter name and specify the parameter type.
   Further information can be found in "Transfer and return of values in VBS (Page 1402)".
   - "ByVal" transfers the parameter value.
   - "ByRef" transfers the address of the parameter.
7. Enter its code in the work area.
   You will find further information about this in "Scripts" editor (Page 1384)".

Result

The user-defined VB function has been created. The type, name, and function parameters are displayed in the title bar.

The start and end are inserted automatically when the user-defined function is created.

If you have protected the user-defined VB functions by a password, it is saved encrypted in the project. For additional information, refer to "Protecting user-defined functions (Page 1416)".

The figure below shows a user-defined VB function which converts the temperature from "Fahrenheit" to "Degrees Celsius":

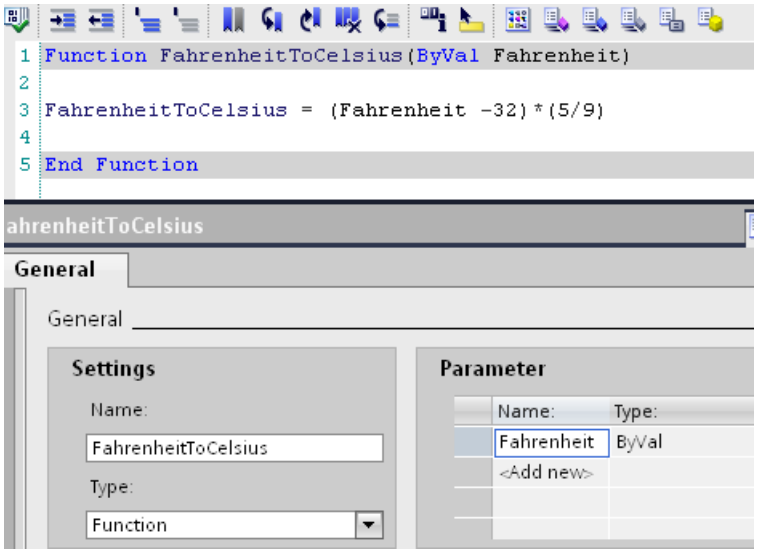

See also

Transfer and return of values in VBS (Page 1402)

"Scripts" editor (Page 1384)

Protecting user-defined functions (Page 1416)

Calling system functions (Page 1387)
Calling customized functions (Page 1388)

9.4.4 Renaming customized VB functions (RT Professional, RT Advanced, Panels, Comfort Panels)

Renaming a user-defined VB function in the project tree

Proceed as follows to rename a user-defined VB function in the project tree:

1. Select the desired user-defined VB function in the project tree.
2. Select the "Rename" (F2) command in the shortcut menu of the user-defined VB function.
3. Enter a new name for the user-defined VB function.

Renaming a user-defined VB function in the Inspector window

Proceed as follows to rename a user-defined VB function in the Inspector window:

1. Open the user-defined VB function in the "Scripts" editor.
2. In the Inspector window, click "Properties > Properties > General".
3. Enter the new function name.

Result

The user-defined VB function is renamed. The application points are adapted automatically.

9.4.5 Executing customized VB functions (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In WinCC you have the following options to execute a customized function:

- Function list
- Customized VB function
- Local VB script (only for WinCC Runtime Professional)

Calling a customized VB function in a function list

1. Add the user-defined VB function to a function list like a system function.
   You will find the customized VB functions in the drop-down list under "VB functions".
   You will find further information about this in "Configuring a function list".
2. If the user-defined VB function has parameters, supply the parameters with static values or HMI tags.
3. If the user-defined VB function has a return value, select an HMI tag. The value to be processed is transferred to the HMI tag.

Note

You can only use customized functions of the same programming language in a function list. You specify the programming language with the selection of the first customized function.

Calling a customized VB function in a customized VB function

1. Open the user-defined VB function in which you want to call the user-defined VB function.
2. Call the customized VB function in the syntax of the programming language. You will find further information about this in "Calling customized functions".
3. If the user-defined VB function has a return value, assign an expression, such as a local tag, to the VB function.

Note

You can only call user-defined functions of the same programming language in a user-defined VB function.

Calling a customized VB function in a local VB function

1. The function list is converted into a local VB script.
2. Call the customized VB function in the syntax of the programming language. You will find further information about this in "Calling customized functions".
3. If the user-defined VB function has a return value, assign an expression, such as a local tag, to the VB function.

Note

You can only call user-defined functions of the same programming language in a local VB script.

See also

Calling system functions (Page 1387)

Calling customized functions (Page 1388)

Configuring a function list (Page 1380)

9.4.6 Transfer and return of values in VBS (RT Professional, RT Advanced, Panels, Comfort Panels)

Transfer of a value

The following options are available for transferring parameters.

- "Call by Value" - ByVal
  ByVal transfers the parameter value. If you transfer a tag as a parameter, the tag value is transferred to the user-defined function when the user-defined function is executed.
- "Call by Reference" - ByRef
  ByRef transfers the address of the parameter. If you transfer a tag as a parameter, the tag address is transferred to the user-defined function when the user-defined function is executed.
  When user-defined functions and system functions are called in user-defined functions, the parameter is transferred as ByRef.

Return of a value

Return values can return the result of a calculation (e.g., average value of two numbers). But a return value can also give information about whether an instruction was executed correctly.

Therefore, the system functions that perform file operations such as "Delete" also have return values.

For a user-defined function to return a value, you must set the "Function" type. You assign the function name to the return value in the user-defined function:

```
1 Function Average( Value1 , Value2 )
2 'Check if Parameters are not numeric:
3 If IsNumeric (Value1) = False Then Value1 = 1
4 If IsNumeric (Value2) = False Then Value2 = 1
5 Average = (Value1+Value2)/2
6
7 End Function
```

To form an average value of two numbers, call the user-defined VB function Average and transfer the result to the HMI tag for example AverageValue.

- In a customized VB function

```
SmartTags("AverageValue") = Average ("4", "10")
```

- In the function list

Average	
Return value	AverageValue
Value1	10
Value2	4

You can output the value of the HMI tag AverageValue in an I/O field.

See also

Create customized VB functions (Page 1398)
Example: Converting Fahrenheit into degrees Celsius (Page 1443)
Example: Converting inches into meters (Page 1445)

9.5 Creating user-defined C functions (RT Professional)

9.5.1 Accessing objects with C (RT Professional)

Introduction

Use the following system functions to access the properties of objects:

- "GetProp": System functions which begin with "GetProp" get a value.
- "SetProp": System functions which begin with "SetProp" set a value.

There is a separate system function with GetProp and SetProp for every data type.

Addressing an object property

You can open the object list for assigning parameters with <Ctrl+J> and select the available objects and properties from the list. The data types of the transferred parameters are checked for plausibility in a syntax validation.

Writing an object property

The following example shows how a circle is moved to layer "4":

SetPropLong("Screen_1", "Circle_1", "Layer", 4);

See also

Example of accessing objects in the Screens editor (Page 1469)

9.5.2 Creating a C-Header (RT Professional)

Introduction

Use the "GlobalDefinitions.h" header file to define functions, tags and constants which you need in all local C scripts and customized C functions. In this way local C scripts and customized C functions remain slim and you avoid redundancy. Changes are carried out at a central point in the header file.

Customized C functions which you have defined in the project are automatically available. You do not need to define these C functions in the "GlobalDefinitions.h".

C libraries such as "Math" are reference by default.

You can also:

- create additional C header files.
- integrate external C header files.

Creating an additional C header file

1. Double click "Add new header" under "Scripts >C headers" in the project tree".
2. Enter a "Name" for the C header in the Inspector window under "Properties >General >General".
3. Insert the required declarations of tags and constants in the work area.

Using the C header file in a C function

1. To use the declarations of the C header file in a C function, insert the C header in the global area of the C function. C header is integrated with an #include instruction. C header files may not be read in several times, to avoid errors in the tag access. Multiple reading of header files is prevented for example by the following structure.
```
#ifndef _myHeader_h
#define _myHeader_h
#include "myheader.h"
#endif
```

Include path for external C header files

You must specify the Include path to use external C header files. Proceed as follows:

1. Select the required HMI device.
2. Select the "Options > Settings" menu command.
3. Select the path for the additional C header in the area navigation under "Visualization > Runtime Scripting".
4. Close WinCC and start it again.
5. Select the "Compile > Software (compile all)" command in the shortcut menu of the HMI device.

Note

Copying a project in which an external C header file is used

The path specification for the external header file is not saved in the project and is only valid on the computer on which it was set. If you copy this project to another computer, you must reassign the Include path for external C header files in the copied project.

9.5.3 Use of Global C-Tags (RT Professional)

Definition of global C tags

A global C tag is inserted by adding the definition line in front of the function name of a function:

```
int global_a; //The tag global_a is defined as integer
void myfunction()
{
//insert the code starting here
}
```

Use of global C tags

You use global C tags in user-defined C functions or local C scripts by declaring the C tags within the header file "GlobalDefinitions.h" as external:

GlobalDefinitions.h

```
extern int global_a;
```

Customized C function

```
// Insert the header starting here
#include "GlobalDefinitions.h"
void myFunction()
{
//insert the code starting here
printf ("C2: %i\r\n", ++ global_a);
}
```

This tells the compiler that it does not need to create the tag global_a, rather that it will be created elsewhere in Runtime.

If the value of the global_a tag is now changed, the changed value is available in all user-defined C functions and local C scripts.

You can also declare C tags directly within a function. However, for reasons of clarity and to avoid duplicate definitions, we recommend defining global C tags in only one location.

Note

A maximum of 64 Kbytes is available to a user-defined function and the global C tag defined with it.

Validity range

All tags in GlobalDefinitions.h declared as external are known to all user-defined C functions and local C scripts in Runtime. It is created as soon as Runtime is started, even if the user-defined function itself was not called.

Note

In WinCC service mode, global C tags cannot be exchanged between user-defined C functions in the Scheduler and user-defined functions in the Screens Editor.

If you want to address tags via the WebNavigator, you need to call up the C function via the globally declared C tag.

9.5.4 Calling functions from DLLs in a user-defined C function (RT Professional)

Introduction

You can use functions from DLLs in user-defined C functions. This lets you use the range of functions from existing DLLs and avoid redundancy.

Note

Release and debug DLLs

WinCC is supplied as a Release version and therefore uses the Release versions of system DLLs. If you want to use your own DLLs in the Debug version, you can also load the Release DLL. Always use the Release version of the DLL when using your own DLLs to avoid a system overload.

Requirement

DLL is in the "..\Common Files\Siemens\bin" folder or in a folder which is defined in the PATH system tag. This tag is specified by the system properties of the operating system.

Procedure

Proceed as follows to call functions from DLLs:

1. Open the user-defined C function in which you want to call DLL functions.
2. Add the following code to the beginning of the user-defined C function:

```
#pragma code("<Name>.dll")
//Declare functions stored in DLL:
<Return value> <Function name 1>(...);
<Return value> <Function name 2>(...);
...
<Return value> <Function name n>(...);
#pragma code
```

Result

The functions <Function name 1> ... <Function name n> from <Name.dll> are declared. You can call the functions in the user-defined C function.

Example

The following example shows how to integrate the "kernel32.dll" file and call the "GetLocalTime" function:

```
#pragma code("kernel32.dll")
VOID GetLocalTime(LPSYSTEMTIME lpSystemTime);
#pragma code()

SYSTEMTIME st;

GetLocalTime(&st);
```

9.5.5 Set up code page for C function (RT Professional)

Introduction

You can use your WinCC project in several languages. You define for each C script which code page is used to translate which language. The code page affects the display of all texts, which are output in runtime in the C script.

When you create a C function, you decide whether to select the settings of the HMI device or one of the configured runtime languages.

The following options are available in the runtime settings of the HMI device:

- Operating system language for non-Unicode programs
  The C function is translated into English in the ES.
  Texts are displayed in the code page of this language in runtime.
- Currently set WinCC Runtime language
  The C function is translated into English in the ES.
  Texts are displayed in the code page of this language in runtime.
- Configured runtime language
  The C function is translated into this language in the ES.

Requirement

- The required languages are enabled in the "Project Languages" editor.
- The required runtime languages are enabled in the "Language & Font" editor.

Select code page for the HMI device

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font".
3. Under "Language setting for script > Language of C scripts with 'Use device setting'", set the selection window using :
   - Selection of all runtime languages
   - Selection of all the other options
4. Select the required code page.

Result

If texts are output in runtime via C function, the code page set for the script is used.

9.5.6 Create customized C functions (RT Professional)

Introduction

When you create a customized C function, you define the following settings:

- The name with which you call the user-defined C function.
- The type of user-defined C function.
- The parameters which are transferred to the user-defined C function in Runtime.

  Note

  When you change the parameters or the function type, the change is color marked at the application point, e.g. in the function list.

- The code page for the C function.

Use only the following characters for the name:

- "A" to "Z"
- "a" to "z"

- "0" to "9"
- "_" as a separator

Note

Names are not case-sensitive. It is not possible to create two user-defined C functions with the same name but different notation.

Note

Invalid tag names

The following designations are already assigned to the "char" data type in Runtime and must not therefore be used as internal tags in C scripting:

- "Object"
- "Property"
- "Tag"
- "Picture"

Procedure

Proceed as follows to create a new customized C function:

1. Open "Scripts" in the project tree.
2. Double click "Add new C function" in the project tree under "C scripts".

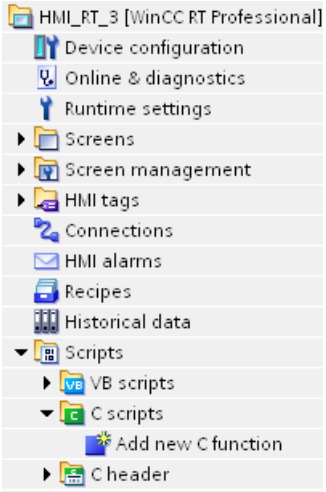

The user-defined C function is created as a new tab in the work area. The input mask for the configuration settings of the function opens in the Inspector window.

3. Select the required language setting under "Code page".

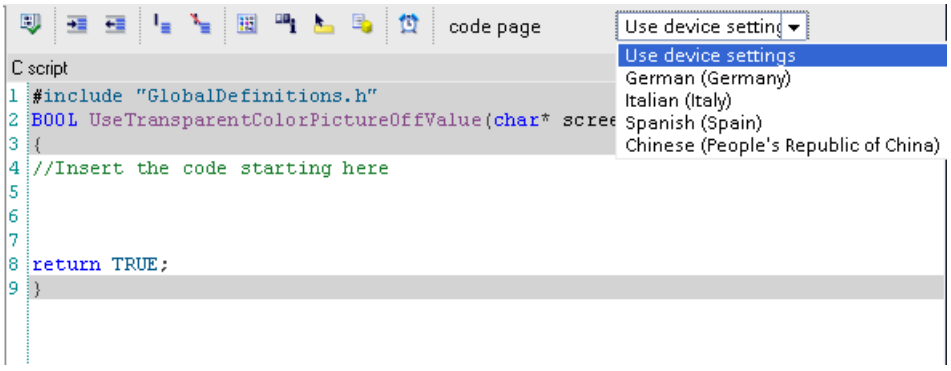

4. Configure the function in the Inspector window "Properties > General > General".
5. Enter a descriptive "Name" for the user-defined C function.
   Select the "Type" of the user-defined C function. Further information can be found in "Transfer and return of values in C".
   - Void: Function has no return value.
   - Tag: Return value is returned as a tag value.
   - Pointer: Return value points on the address of the tag.
6. Double click "<Add>" in the "Parameters" list to add parameters. Enter the parameter name. Further information can be found in "Transfer and return of values in C".
   - Tag transfers the parameter value.
   - Pointer transfers the address of the parameter.
7. Write the code in the work area.

Result

The user-defined C function is created. The type, name, and function parameters are displayed in the title bar.

The start and end are inserted automatically when the user-defined function is created.

If you have protected the user-defined C function by a password, it is saved encrypted in the project. For more information, refer to "Protecting user-defined functions (Page 1416)".

The figure below shows a user-defined C function which converts the temperature from "Fahrenheit" to "Degrees Celsius":

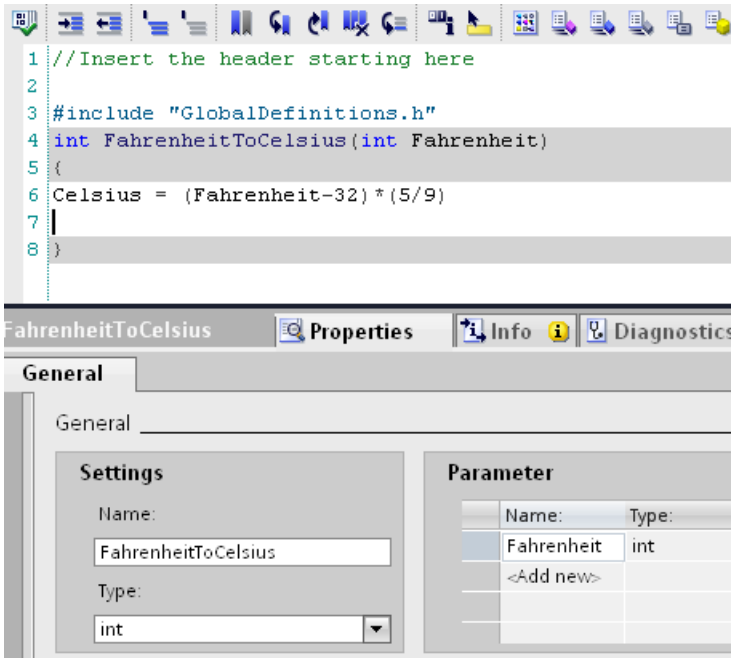

See also

"Scripts" editor (Page 1384)

Protecting user-defined functions (Page 1416)

9.5.7 Renaming customized C functions (RT Professional)

Renaming a user-defined C function in the project tree

Proceed as follows to rename a user-defined C function in the project tree:

1. Select the user-defined C function in the project tree.
2. Select the "Rename" (F2) command in the shortcut menu of the user-defined C function.
3. Enter a new name for the user-defined C function.

Renaming a user-defined C function in the Inspector window

Proceed as follows to rename a user-defined C function in the Inspector window:

1. Open the user-defined C function in the "Scripts" editor.
2. In the Inspector window, click "Properties > General > General".
3. Enter the new function name.

Result

The user-defined C function is renamed. The application points are adapted automatically.

9.5.8 Executing customized C functions (RT Professional)

Introduction

In WinCC you have the following options to execute a customized C function:

- Function list
- Customized C function
- Local C script

Calling a customized C function in a function list

1. Add the user-defined C function to a function list like a system function.
   You will find the customized C functions in the drop-down list under "C functions".
   You will find further information about this in "Configuring a function list".
2. Supply the parameters with static values or HMI tags.
3. If the user-defined C function has a return value, select an HMI tag. The value to be processed is transferred to the HMI tag.

Note

You can only use customized functions of the same programming language in a function list. You specify the programming language with the selection of the first customized function.

Calling a user-defined C function in a user-defined C function

1. Open the user-defined C function in which you want to call the user-defined C function.
2. Call the customized C function in the syntax of the programming language. You will find further information about this in "Calling customized functions".
3. If the user-defined C function has a return value, assign an expression, such as a local tag, to the function.

Note

You can only call user-defined C functions of the same programming language in a user-defined C function.

Calling a customized C function in a local C script

1. The function list is converted into a local C script.
2. Call the customized C function in the syntax of the programming language. You will find further information about this in "Calling customized functions".
3. If the user-defined C function has a return value, assign an expression, such as a local tag, to the C function.

Note

You can only use user-defined functions of the same programming language in a local C script.

See also

Calling system functions (Page 1387)

Calling customized functions (Page 1388)

Configuring a function list (Page 1380)

9.5.9 Transfer and return of values in C (RT Professional)

Transfer of a value

You have the following possibilities for transferring parameters:

- Tag (Call by value)
  The tag name and the data type are specified to save values in the tag. But this tag is not transferred directly but a copy made. When executing the user-defined C function, the tag value of the copy is transferred to the C function. If you change the tag value, the change is only valid within the user-defined C function.
- Pointer (Call by reference)
  A pointer is a tag which points on the address of a tag. Pointers are defined like tags. The * character is added to the data type, e.g. Double*. The user-defined C function accesses the tag to which the pointer is pointing. If you change the tag value, the changed tag value is written into the tag.

Return of a value

A user-defined C function can execute simple instructions and return no value. The return value is then a "void" data type.

If a return value is required, the value can be returned as a tag or a pointer.

But a return value can also give information about whether an instruction was executed correctly. Therefore, the system functions that perform file operations such as write to a file also have return values.

For a user-defined C function to return a value, you must set the "Double" type, for example. The result is transferred to the user-defined C function with the return instruction.

```
#include "GlobalDefinitions.h"
double Average(double Value1, double Value2)
{
double dbAverage;
dbAverage= (Value1+Value2)/2;
return dbAverage;
}
```

To form an average value from two numbers, call the user-defined C function Average and transfer the result to the HMI tag, for example, AverageValue.

- In a customized C function

```
SetTagDouble("AverageValue",Average (4, 5));
```

- In the function list

Average	
Return value	AverageValue
Value1	10
Value2	4

You can output the value of the HMI tag AverageValue in an I/O field.

9.6 Protecting user-defined functions

9.6.1 Setting up know-how protection (RT Professional, RT Advanced, Panels, Comfort Panels)

Procedure

To set up know-how protection for user-defined functions, follow these steps:

1. Select the user-defined function without know-how protection that you want to protect.
2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.
3. Click "Define".
   The "Define password" dialog box opens.
4. Enter a password in the "New" field.
5. Enter the same password in the "Confirm" field.
6. Confirm your entries with "OK".
7. Close the "Know-how protection" dialog by clicking on "OK".

Result

The user-defined function receives know-how protection. User-defined functions with know-how protection are marked with a lock in the project tree. The password entered is valid for all user-defined functions selected.

9.6.2 Opening user-defined functions with know-how protection

You can only open multiple user-defined functions with know-how protection at once if they are protected with the same password.

Procedure

To open a know-how-protected user-defined function, follow these steps:

1. Double-click the user-defined function you want to open.
   The "Access protection" dialog will open.
2. Enter the password for the user-defined function with know-how protection.
3. Confirm your entry with "OK".

Result

The user-defined function with know-how protection will open provided you have entered the correct password. However, the function will remain know-how protected.

Once you have opened the user-defined function, you can edit it for as long as the user-defined function or TIA portal is open. You must enter the password again the next time you open the user-defined function. If you close the "Access protection" dialog with "Cancel" or "Close", the user-defined function is opened but the code is not displayed. You cannot edit the user-defined function.

9.6.3 Removing know-how protection

Procedure

To remove the know-how protection of a user-defined function, follow these steps:

1. Select the user-defined function for which you want to remove know-how protection.

   Note

   To remove know-how protection for several user-defined functions at once, all selected user-defined functions must have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.
3. Deactivate the check box "Hide code (know-how protection)".
4. Enter the password.
5. Confirm your entries with "OK".

Result

Know-how protection will be disabled for the selected user-defined function.

9.6.4 Changing a password

Procedure

To change the password, follow these steps:

1. Select the user-defined function with know-how protection for which you want to change the password.

   Note

   To change the password for several user-defined functions at once, all selected user-defined functions must have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.
3. Click the "Change" button.

4. Enter the old password in the "Old" field.
5. Enter the new password in the "New" field.
6. Enter the new password again in the "Confirm" field.
7. Confirm your entries with "OK".
8. Close the "Know-how protection" dialog by clicking on "OK".

9.7 Creating local scripts (RT Professional)

9.7.1 Local scripts (RT Professional)

Definition

Local scripts are bound to a single object, such as an object property, and have a cyclic or acyclic trigger. A local script is created directly at the point of use:

- Scheduler
- Screens
  - Event (converted function list)
  - Object property (property list)
  - Property change (property list)
- Menus and toolbars
- Only C script: Reports
  - I/O field at the process value property

Local scripts are not displayed in the project navigation and are only available at the point of use.

Application

Local scripts are used for the following:

- Property list (screen object): Dynamic control of object properties
  A local script is used to dynamize an object property in Runtime by means of its return value. Example: Changing the object position in dependence of several parameters.
- Function list (screen object): Converting a function list to a local script
  Convert a function list to a local script in order to configure conditional instructions or loops at an event.
- Scheduler (task): Execute local script cyclically or acyclically
  Use the Scheduler to execute a script at a specified point in time.

Local scripts are not loaded into the memory of the HMI device until they are used, for example when loading a screen. Local scripts thus reduce memory requirements on the HMI device.

Programming

When a local script is created at an object property, a local script with return value is generated. The return value supplies the value for the object property.

When a local script is created at an event, a local script without return value is generated.

The parameters which are transferred to the local script depend on the used programming language and the application point.

Typical parameters of a local VB script

Item: Contains the object reference:

- Scheduler (task): No object is transferred.
- Menu & toolbars: The object that triggers the event is transferred.
- Screens (screen object):
  - Property list (screen object): In dynamizations, the screen object whose property you dynamize is transferred.
  - Converted function list (screen object): The screen object whose event you configure is transferred.

  To access the properties of the transferred screen object, enter "item.". The available properties are listed. Use ScreenItems to access another object of the object model.. Further information about parameters of events are available in the reference.

Typical parameters of a local C script

Char* screenName: Contains the name of the screen that contains the object whose property is being dynamized.

Char* objectName: contains the name of the object whose property you dynamize.

Char* propertyName: Contains the property that is being dynamized.

Further information about parameters of events are available in the reference.

Execution of local scripts

Local scripts require a trigger. The trigger specifies when the script will be executed. The point of use of the local script determines which trigger is used:

- Local scripts of a converted function list are executed when the event occurs, for example, printing.
- Local scripts in the property list get the trigger automatically. You can change the trigger, e.g. to tag trigger.
- Local scripts of a task are assigned a trigger by the project engineer.

See also

"Scripts" editor (Page 1384)

9.7.2 Editor for local scripts (RT Professional)

Properties of the editor for local scripts

The editor for local scripts has no own setting possibilities but "inherits" the settings from "Runtime Scripting".

The editor offers similar functions to the "Scripts" editor. You will find further information in ""Scripts" editor (Page 1384)".

The following functions are not available in the toolbar:

- Drop-down lists for properties and functions

The editor for local VB scripts contains the following additional icons:

- Shows declaration range.
- Hides declaration range.

You define local VB tags in the declaration range. For more information, refer to "Use of VB tags (Page 1395)".

Availability

The editor for local scripts can be called in the following places:

- Scheduler
  - Event: Execute
- Screens
  - Event
  - Object property (property list)
  - Property change (property list)
- Menus and toolbars
  - Event: Click
- Only C script: Reports
  - I/O field at the process value property

See also

"Scripts" editor (Page 1384)

Use of VB tags (Page 1395)

9.7.3 Converting a function list to a local script (RT Professional)

Introduction

When you convert a function list to a local script, you can configure conditional instructions or loops for an event. A function list is converted either to a local VB or C script.

If the function list does not contain a customized function, the programming language is determined by the function.

If a C function is contained, for example, the function list is converted into a local C script.

Note

Error message in the Global Script diagnostics window

If a tag of the type WString is used in a SetTagsystem function that was converted into a C script, an error message is output in the Global Script diagnostics window when executing the function. The message does not affect the execution of the function.

The following SetTagfunctions are affected:

- SetTagWithOperatorEvent
- SetTagIndirect
- SetTagByTagIndirect
- SetTagIndirectByTagIndirect
- SetTagByProperty
- SetTagIndirectByProperty

Requirement

The function list is opened.

Procedure

Proceed as follows to convert a function list to a local script:

1. Select the event whose function list you want to convert to a script.
2. Configure system functions at the event as required.
3. To convert the function list to a local script, click or .
4. Write the program code.

Alternatively you use the "Convert VB script" or "Convert C script" command in the shortcut menu.

Result

If the event occurs, the local script is processed automatically.

Deleting a local script

Proceed as follows to delete a local script:

1. Select the event with the local script.
2. Select "Function list" in the shortcut menu of the event.

The local script is deleted.

9.7.4 Dynamizing an object property with a local script (RT Professional)

Introduction

With a local script you dynamize an object property in Runtime by the return value of the script.

Requirement

- The object with the property to be dynamized is selected.
- The Inspector window is open.

Procedure

1. Click in the Inspector window.
2. Select the object property to be dynamized.
3. Select the "VB script" or "C script" entry in the "Dynamization" column.
   A dialog opens.
4. Write the program code.
5. Click " " in the toolbar to change the trigger.

Result

The object property changes in Runtime according to how the property list is configured.

Deleting the dynamization

1. Select the dynamized object property in the property list.
2. Select the "None" entry in the "Dynamization" column.

9.8 Debugging

9.8.1 Debugging of user-defined VB functions

Overview

To test a user-defined VB function in Runtime, you can use the debugger of "Microsoft Visual Studio 2008", for example.

The debugger is included in the WinCC Setup. If another Visual Studio version is already installed, Microsoft Visual Studio 2010 for example, use the debugger of this version.

Debugger for WinCC Runtime Professional

The Microsoft Visual Studio 2008 Shell (Integrated Mode) debugger is located on the DVD of WinCC Runtime Professional the Support directory.

Debugger for WinCC Runtime Advanced

You can download the Microsoft Visual Studio 2008 Shell (Integrated Mode) debugger for free from the Microsoft homepage. In doing so, make sure that you use the English version.

The latest versions from Microsoft are generally recommended for the purpose of debugging.

Installing the debugger for WinCC

Procedure

1. Start the Setup of Microsoft Visual Studio 2008 Shell (Integrated Mode)
2. Follow the instructions and accept the default settings.
   The following program is installed: Microsoft Visual Studio 2008 Shell (Integrated Mode) - ENU
3. When you start the debugger the first time, the "Visual Studio Just-In-Time Debugger" dialog opens.
   Select the entry "New instance of Visual Studio 2008".
   To specify "Visual Studio 2008" as the default debugger, select the "Set the currently selected debugger as the default." setting.

Microsoft Visual Studio 2008 Debugger

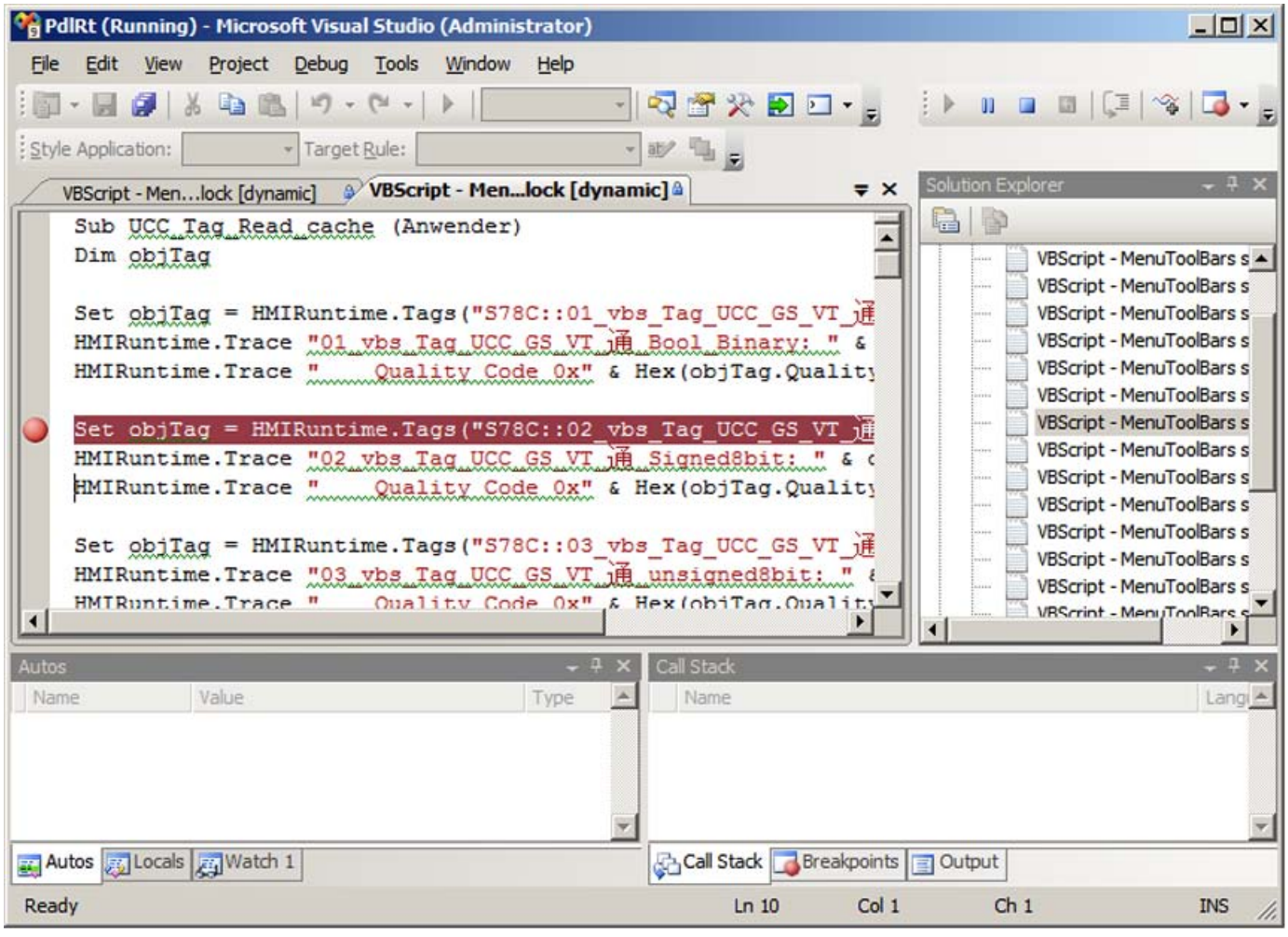

9.8.2 Principles of Debugging

Introduction

For example, you can use a debugger to test whether correct values are being transferred to tags and whether abort conditions are being correctly implemented. You can do the following in the debugger:

- Inspect the source code of the function
- Check the step-by-step execution of the functions
- View and change tags and property values
- See and check function sequence

The debugger is only intended for identifying error locations in the user-defined VB function. To do this, you use breakpoints, for example, or you execute the user-defined VB function step-by-step. For detailed information, refer to the documentation of the debugger you are using. In order to correct errors, you change to the "Scripts" editor in WinCC. After compiling and downloading again, you test the user-defined VB function again with the debugger.

Note

Please note that the code displayed in the debugger is write-protected. The code cannot be changed directly in the debugger but only test the necessary changes.

Difference with regard to VBScript for Windows and Windows CE

The debugger checks the syntax for VBScript for Windows. If the function contains a VBScript function for Windows CE, a corresponding error message is output. Some VBScript functions are different, e.g. CreateObject.

Error types

A distinction is made between the following types of error by the debug:

- Syntax error
  Syntax errors arise, for example, when you misspell a keyword or fail to close parentheses. If you use the syntax check of WinCC, you can eliminate syntax errors before testing your functions in Runtime.
  The syntax check in WinCC parses a function without executing it. The function is parsed again immediately before its execution in Runtime. All sections are parsed, including those that are not executed until a particular action occurs at a later time.
  If a function contains syntax errors, you cannot start Runtime.

  Note

  For Runtime Professional you can specify in "Runtime settings > General > Script options" that Runtime will be started even when a function contains syntax errors.

- Runtime error
  A runtime error occurs when an invalid/incorrect command is to be executed, e.g. because a tag is not defined. To intercept runtime errors, you can use the "On Error Resume Next" instruction in a function. This instruction causes the subsequent instruction to be executed after a runtime error. In the subsequent line, the error code can then be checked using the Err object. To switch off the handling of runtime errors in the script, use the "On Error Goto 0" instruction.
- Logical errors
  The debugger is particularly helpful in clearing up logical errors. A logical error occurs when an unexpected result is received because, for example, a condition was incorrectly checked. To eliminate logical errors, execute your function step-by-step in order to identify the section of the script that is not functioning properly.

Example of error handling

```
Sub OnClick(ByVal Item)
'VBS27
  Dim objScreenItem
'
  'Activation of errorhandling:
  On Error Resume Next
  For Each objScreenItem In ScreenItems
    If "HMIRectangle" = objScreenItem.Type Then
    '
      '=== Property "RoundCornerHeight" only available for RoundRectangle
      objScreenItem.RoundCornerHeight = objScreenItem.RoundCornerHeight * 2
      If 0 <> Err.Number Then
        HMIRuntime.Trace objScreenItem.Name & ": no RoundedRectangle" & vbCrLf
        '
        'Delete error message
        Err.Clear
      End If
    End If
  Next
  On Error Goto 0  'Deactivation of errorhandling
End Sub
```

Basic Procedure

When an error has occurred and the debugger is open, the script appears in a window, write-protected. It is possible to navigate through the script document, set breakpoints, execute the script again in Runtime and to process the script step-by-step.

The source codes of the scripts cannot be edited directly in the scripts. If you have found an error, you can correct the error in the function in WinCC, download the picture again and update the picture in the debugger.

Change Picture During Debug

If a picture change is executed during debugging, the script document of the "old" picture remains open but is no longer valid. If necessary, invalid errors are displayed because the objects called following the picture change are no longer available.

9.8.3 Activating the debugger

Principle

There are several ways of activating the debugger:

- Automatic activation of the debugger when an error occurs in Runtime.
- Opening an error box in Runtime via which the debugger can be activated.
- Starting the debugger from the Start menu and opening a running Runtime scripts.

Connecting Visual Studio to Runtime

1. Start Runtime.
2. Open "Microsoft Visual Studio 2008".
3. In the "File > New" menu, click on "File".
4. Select "Windows Script Host" and click on "Open".
5. In the "Debug" menu, click on "Attach to Process".
6. Select:
   HmiRTm.exe: for a panel or WinCC Runtime Advanced
   PdlRt.exe: for WinCC Runtime Professional
7. Double-click one of the script files in the "Solution Explorer".

Visual Studio as Just-In-Time debugger

1. Open "Microsoft Visual Studio 2008".
2. In the "Tools" menu, click on "Options".
3. In the tree, click on "Debugging" and "Just-In-Time".
4. Under "Just-In-Time Debugging", select the "Script" option.
5. Close Visual Studio.
6. Start the simulation using "Online > Simulation > With script debugger".

When a function with errors is executed in Runtime, a dialog opens from which you can open the debugger.

Stopping the debugger

It is possible to stop the debugger without exiting the WinCC Runtime.

9.8.4 Structure of VBScript files (RT Advanced, Panels, Comfort Panels)

Structure of VBScript files

All user-defined functions configured in a device are contained in a script file.

9.8.5 Structure of VBScript files (RT Professional)

Principle

To enable simultaneous execution of cyclic and event-driven VBScript files in Runtime Professional, event-driven VBScript files and cyclic/tag-triggered VBScript files are strictly separated during their execution. A cyclic local script cannot interfere with the execution of a user-defined function which is to be triggered by clicking a button, for example.

To ensure this, the event-driven local scripts and cyclic local scripts as well as the tag-triggered local scripts are stored in separate VBScript files. Any global screen element you have defined for VBScript files in the "Screens" editor is copied to both VBScript files.

Note

Because the two VBScript files are handled separately, they have no common data area. Therefore, there is no synchronization of global tags between the two VBScript files. Implement any synchronization function which may be required using the DataSet object or internal HMI tags.

Structure of VBScript files

When VBScript files are debugged with a debugger, the VBScript files always open the different Runtime systems.

In the case of the graphical Runtime system, this means that you receive two files per picture:

- <ScreenName>.pdl_events: Contains the event-driven scripts.
- <ScreenName>.pdl_triggers: Contains the cyclic and tag-driven VBScript files.

The following section describes the structure of script files within the graphical Runtime system:

- Content of the VBScript file <ScreenName>.pdl_events
  General code
  Event-driven scripts
- Content of the VBScript file <ScreenName>.pdl_triggers
  General code
  Cyclic or tag-triggered scripts

  Note

  Note that the local scripts and user-defined functions of the graphical Runtime system are not displayed by their names in the VBScript file to which they were saved in WinCC.

9.8.6 Names in the debugger (RT Professional)

Names in the debugger

The names of user-defined functions in the debugger differ from the names by which you saved your user-defined functions in WinCC.

The function and script names in the script files are compiled according to the following rules:

Action type	Name of the script file
Cyclic or tag-triggered scripts at a property	ObjectName_PropertyName_Trigger
Mouse events	ObjectName_OnClick ObjectName_OnLButtonDown ObjectName_OnLButtonUp ObjectName_OnRButtonDown ObjectName_OnRButtonUp
Keyboard events	ObjectName_OnKeyDown ObjectName_OnKeyUp
Object events	ObjectName_OnObjectChanged ObjectName_OnSetFocus
Events on properties	ObjectName_PropertyName_OnPropertyChanged ObjectName_PropertyName_OnPropertyState-Changed
Picture events	Document_OnOpen Document_OnClosed

Valid length of the script names

The name of scripts in the script files is limited to 255 characters. Each special character used in an object name is converted to five characters. The special character appended to the leading X is represented as four-digit hexadecimal code. If you configure a button action named "PushMe" with mouse click trigger, the "DrXFFFCckMe_OnClick" script is written to the script file.

If the composite object name is too long, the syntax check in WinCC returns an error message. Because of this restriction, you cannot configure graphic object names of unlimited length.

Note

To ascertain the name of an object in Runtime, click <Ctrl+Alt+Shift> and position the mouse over the corresponding object. The screen name and the object name are displayed in a tooltip.

9.8.7 Breakpoints

Introduction

To stop the execution of the VBScript file at certain points and to localize logical errors step-by-step, set breakpoints.

When a VBScript file is updated in the debugger, all the breakpoints are lost.

Note

When you set a breakpoint in script file "<Picturename>.pdl_trigger" in Runtime Professional, all trigger-driven procedures are stopped.

When you set a breakpoint in script file "<Picturename>.pdl_event" in Runtime Professional, all event-driven procedures are stopped.

Requirement

- Runtime is activated.
- The screen to be debugged is active.

Setting a breakpoint

1. Start the debugger and select the VBScript file.
   If automatic activation of the debuggers in WinCC has been selected, the debugger is called in as soon as an erroneous script is executed.
2. Place the cursor in the line in which you want to set a breakpoint.
3. In the "Debug" menu, select the "Toggle Breakpoint" command.
   The line is marked with a red dot.
4. Change to WinCC Runtime and execute the VBScript file to be debugged.
   The debugger stops at the first breakpoint. The current line is displayed on a yellow background.
   Simulate the program code step-by-step.

Deleting a breakpoint

1. Position the cursor in a line with a breakpoint.
2. In the "Debug" menu, select the "Toggle Breakpoint" command.

To delete all breakpoints in a file, select the "Delete All Breakpoints" command in the "Debug" menu.

9.8.8 Working step-by-step

Introduction

You use the debugger in order to execute your VBScript files (user-defined function or local script) step-by-step, for example, to systematically localize logical errors. Simulate the effect of each code line in Runtime.

Requirements

- Runtime is activated.
- The screen to be debugged is active.

The Procedure in Principle

1. Start the debugger and select the VBScript file.
   If you have selected automatic activation of the debugger in WinCC, the debugger is called as soon as a script with error is executed.
2. Set a breakpoint in the VBScript file. Breakpoints are normally set in front of code lines in which errors are suspected.
3. Change to WinCC Runtime and trigger an action that initiates execution of the VBScript files.
   The debugger stops at the first break point and marks the current line.
4. To continue working step-by-step, select one of the following menu commands:
   – "Debug > Step Over": goes to the next code line. If a user-defined function is called in this line, you skip the called user-defined function. The user-defined function is executed but the debugger does not guide you through the individual lines of the function. Instead, you jump to the first line after the function was executed.
   – "Debug > Step Into": goes to the next code line. If a user-defined function is called in this line, you jump to this function. Execute the called function step-by-step.
   – Select the "Debug > Step Out" menu command to cancel step-by-step execution of a called function. The function is executed completely. The debugger jumps to the first line after the function was executed.
5. Go through the document step-by-step until you reach the end, or select the "Debug > Run" menu command to execute the VBScript file to the end.

9.8.9 Executing commands

Introduction

Commands are entered and directly executed in the "Command Window". You can:

- Call methods
- Call procedures
- Change object properties

The "Command Window" is used to execute all commands which can also be executed in a VBScript.

Note

If you want to determine the name of a WinCC object in WinCC Runtime Professional, press <Ctrl+Alt+Shift> and move the mouse over the corresponding object. The picture name and the object name are then displayed in a tooltip.

Requirement

- The VBScript file runs in Runtime.
- The debugger is open.

Procedure

1. Set at least one breakpoint in the VBScript file.
2. Change to WinCC Runtime and trigger an action that initiates execution of the VBScript file. The debugger stops at the first breakpoint.
3. Select "View > Other Windows > Command Window". "Command Window" opens.
4. Enter the desired command and confirm with "<Enter>".

   Note

   An error message will not be generated in runtime if you enter an invalid command in the "Command Window". The "<Script Error>" message will appear in the Command Window instead.

Example

```
Dim i
For i = 1 to 10
    HMIRuntime.Trace i & vbnewline
Next
```

When this VBScript is executed and the debugger is in the loop at the breakpoint, you can use

- ?i
  to read out the value of variable i
- i = 5
  to set the value of variable i to 5.

9.8.10 Determining tag and property values

Introduction

You can determine and change the values of tags or properties in the "Command Window". You reset a process value for the user-defined function to zero without having to stop the process, for example.

Note

If you want to determine the name of a WinCC object in WinCC Runtime Professional, press <Ctrl+Alt+Shift> and move the mouse over the corresponding object. The screen name and the object name are displayed in a tooltip.

Requirement

- The VBScript file runs in Runtime.
- The debugger is open.

Procedure

1. Set at least one breakpoint in the VBScript file.
2. Change to WinCC Runtime and trigger an action that initiates execution of the VBScript file. The debugger stops at the first breakpoint.
3. Select "View > Other Window > Command Window". "Command Window" opens.
4. To determine the value of a tag or property, enter the following character string
   - ? myTag
   - ? myObject

   Note the space in front of the object name.
5. Press <Enter>.

9.8.11 Diagnostics (RT Professional)

9.8.11.1 Diagnostics (RT Professional)

Introduction

If you execute your local VB scripts and C scripts in Runtime, you output a quick analysis with the aid of the "Print job/Script diagnostics" object.

Diagnostics tools

WinCC provides you with tools to analyze the runtime behavior in local C scripts and VB scripts:

- The print job/script diagnostics "GSC diagnostics" object
- Using a debugger

Application

To use the GSC Diagnostics, you insert the "Print job/Script diagnostics" object in a screen. Use the "GSC diagnostics" template.

This can be a dedicated screen which is designed for diagnostics and which is called in Runtime.

GSC diagnosis of VBS

GSC diagnostics returns the trace methods contained in the local scripts in the order in which they were called. Trace instructions in user-defined VB functions, which are called in local scripts, are also output. Explicit implementation of trace instructions, e.g. for output of the tag values, enables the tracing of script execution and of the functions called within those scripts. You enter the trace instruction in the form "HMIRuntime.Trace(<Output>)". Error situations which result in calling of the OnErrorExecute function are also displayed in the diagnostic window.

GSC diagnosis of C

GSC diagnosis outputs the printf methods contained in the local scripts in chronological order of their being called. printf instructions in user-defined C functions, which are called in local scripts, are also output. By targeted use of printf instructions, e.g. for outputting tag values, the sequence of scripts and the functions called in them can be tracked.

See also

Example of configuring a diagnostics output via a trace (Page 1453)

9.8.11.2 GSC diagnostics (RT Professional)

Introduction

GSC Diagnostics outputs the trace instructions of the scripts to the diagnostics view in the chronological order of their call. The application also outputs trace instructions in functions which are called in the scripts. Explicit implementation of trace instructions, e.g. for output of the tag values, enables the tracing of script execution and of the functions called within those scripts.

Application

Insert the "Print job/Script diagnostics" object of the type GSC Diagnostics in a screen to enable the use of GSC Diagnostics. Control the appearance of the GSC Diagnostics window by means of the GSC Diagnostics properties.

In the case of a picture change, the content of the GSC Diagnostics window is deleted.

Note

Messages are also displayed in the "GSC Diagnostics" window when the debugger is activated.

9.8.11.3 Inserting a GSC diagnostics window in a screen (RT Professional)

Introduction

To use the GSC Diagnostics, you insert the "Print job/Script diagnostics" object in a screen. Use either any existing screen, or create a screen that is dedicated to diagnostics functions. In the Inspector window of the "Print job/Script diagnostics" object, you specify that the object is used for GSC Diagnostics.

Requirement

- The required screen is open.
- The "Tools" task card is open.

Procedure

Proceed as follows to insert the GSC diagnostics window:

1. Insert the "Print job/Script diagnostics" object from the "Controls" palette of the "Tools" task card into the screen.
2. Assign a name to the GSC diagnostics window at "Miscellaneous > Name".
3. Select the "Functions" option in the inspector window under "Properties > Properties > General > Content > Window content".

4. Select the "GSC diagnostics" option under "Template".
5. Specify the further settings for the GSC diagnostics window under "General", "Layout", and "Miscellaneous".

9.9 Runtime behavior in Runtime

9.9.1 Executing user-defined functions in Runtime (RT Professional, RT Advanced, Panels, Comfort Panels)

Principle

Only one user-defined function at a time is executed in Runtime. If several user-defined functions are waiting to be executed, they are lined up in a queue and executed one after the other.

Note

A loop in a user-defined function therefore blocks the execution of the other functions in the queue even if the functions were initiated asynchronously.

WinCC supports a maximum nesting depth of eight user-defined functions. Note that the nesting depth is not checked.

Note

If a user-defined function is configured for the "Runtime stop" event, the only system functions that may be used in this user-defined function are those which are available at the "Runtime stop" event.

Ensure that the ending of the Runtime is not interfered with by the execution of the user-defined function.

Note

Configuration of user-defined functions

During configuration make sure that not too many user-defined functions are activated at the same time. Avoid a continuous system load of 100%.

User-defined functions are processed at a lower priority so as not to interfere with the display of values and operability. If system utilization is extreme, the user-defined functions to be executed are therefore first only reserved for execution. The maximum size of the reservation list is dependent on the HMI device and is limited by the maximum permitted number of user-defined functions. For additional information, see the performance features. If more user-defined functions are activated at one time than can be reserved, excess calls are discarded and a system alarm displayed.

HMI device changeover

If you use system functions in a customized function which are not available on the set HMI device, you get a warning. In addition, the corresponding system function in the user-defined function will be underlined with a wavy blue line.

See also

User-defined functions (Page 1375)

9.9.2 Executing a function list in Runtime (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Principle

A function list is executed from top to bottom in Runtime. A distinction is made between synchronous and asynchronous execution, so that no waiting periods ensue during execution. The distinction is made by the system by evaluating the different runtimes of the system functions. User-defined functions are always executed synchronously independent of the runtime. If a system function returns an error status, the execution of the function list is cancelled.

Synchronous execution

During synchronous execution, the system functions in a function list are executed one after the other. The previous system function must be finished before the next system function can be executed.

Asynchronous execution

System functions that perform file operations such as saving and reading have a longer runtime than system functions that, for example, set a tag value.

Therefore, system functions with longer runtimes are executed asynchronously. For example, while a system function is writing a recipe data record to a storage medium, the next system function is already being executed. Due to the parallel execution of system functions, waiting periods at the HMI device are avoided.

9.9.3 Execution of functions and local scripts in Runtime (RT Professional)

Processing local scripts in Runtime

Two locale scripts of the same type cannot be executed simultaneously in Runtime. Event-triggered and cyclic/tag-triggered scripts in screens are executed separately to prevent any script from interfering with cyclic scripts.

Note

The script types in WinCC are synchronized exclusively by the DataSet object or internal HMI tags. Separate execution prevents the generation of any shared data areas for event-triggered and cyclic/tag-triggered scripts.

If high system load or other scripts interfere with cyclic execution of local scripts in the screens, the local script is executed once at the next opportunity. Cycles which are not executed are not retained in a queue but rejected.

After a screen change any active scripts are stopped automatically on expiration of 1 minute.

Local scripts which are still active at the end of Runtime are stopped after 5 seconds.

Executing functions in Runtime

Functions are executed successively in Runtime after being triggered. If a function is initiated while another is active, the second function is queued until it can be executed.

Note

The local scripts and functions are synchronized exclusively by the DataSet object or internal HMI tags. Local scripts and functions do not share any data areas.

See also

User-defined functions (Page 1375)

Local scripts (Page 1377)

9.9.4 Processing sequence for user-defined functions and system functions (RT Professional, RT Advanced, Panels, Comfort Panels)

WARNING

Adherence to the expected processing sequence for user-defined functions and system functions in Runtime cannot always be guaranteed. This fact may lead to unexpected operating states.

Please note the following description in which the correlations are explained. Instructions for a solution are available under "Remedy".

Use of user-defined functions for events

It is not always possible to adhere to the processing sequence for user-defined functions and system functions expected in the configuration, for example, when the processing of a user-defined function lasts beyond the trigger time of a subsequent event. In this case, system functions that are configured for the following event may be executed before the system functions that were configured with the user-defined function for the same event.

Background

User-defined functions are arranged in a low-priority queue. The queue is processed in sequence. If system functions and user-defined functions are configured for an event, all system functions and user-defined functions configured for this event are executed in this queue.

Example

On a button, you have configured:

	Event	Functions
	Press	User-defined function (1) SetBit(Tag) (2)
	Release	ResetBit(Tag) (3)

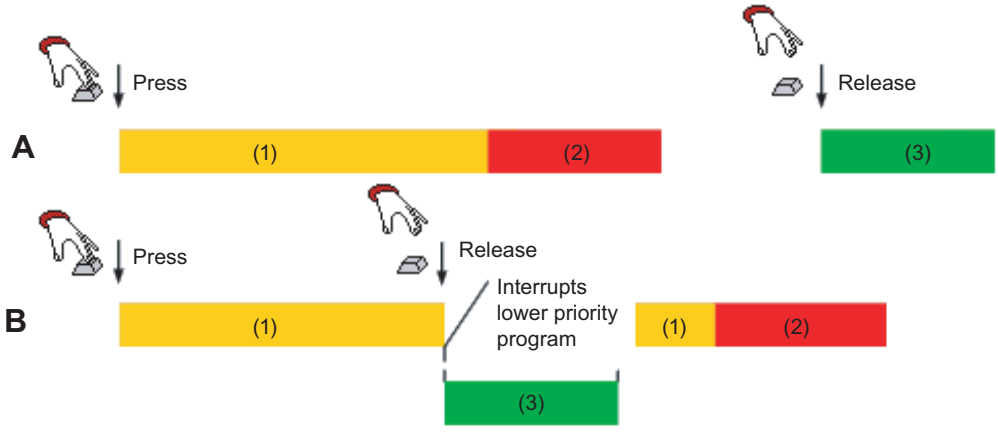

(1) Execution time of the user-defined function

(2) Execution time of system function SetBit(Tag)

(3) Execution time of system function ResetBit(Tag)

A: Expected execution of the user-defined function and system functions in the configuration.

When you press the button, user-defined function (1) and then system function SetBit(Tag) (2) are executed. When you release the key, system function ResetBit(Tag) (3) is then executed.

B: Processing of the user-defined function is not yet complete when you release the key.

When you press the key, the user-defined function (1) is then executed. The execution of the user-defined function (1) is not yet complete and you have already released the key. System function ResetBit(Tag) (3) interrupts the execution of the function. In this case, system function ResetBit(Tag) (3) is executed before system function SetBit(Tag) (2). As a result, the bit remains placed.

System function "SetBitWhileKeyPressed" shows the same behavior, if the system function is configured together with user-defined functions.

Remedy

The processing sequence of the previous Example A is adhered to, if you additionally configure a user-defined function for the "Release" event. The user-defined function does not require functionality for this purpose.

	Event	Functions
	Press	User-defined function (1) SetBit(Tag) (2)
	Release	User-defined function_2 (3) ResetBit(Tag) (4)

Note

Make sure that the user-defined function queue does not overflow; otherwise the system function for the following event may not be executed.

An overflow is recognizable by "Overload: Script <name> is rejected" or "Adherence to the expected processing order for user-defined functions and system functions in Runtime cannot always be ensured".

9.9.5 Making object properties dynamic in Runtime (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can use user-defined functions to access object properties of screen objects as well as tags in Runtime. If you use a user-defined function to change values of object properties, this will not affect the project data.

Changes to object properties

If you use a user-defined function in Runtime to change an object property of a screen element, this change will only remain effective while the screen is active. If you change to the screen, or reload the screen, the configured object properties are displayed once more.

Language switching

If you switch the language in Runtime, the foreign language labels are loaded from the project data. If you use a user-defined function to change texts, the texts changed by the user-defined function are overwritten again.

9.10 Examples

9.10.1 Examples of VBS (RT Advanced, Panels, Comfort Panels)

9.10.1.1 Example: Converting Fahrenheit into degrees Celsius (RT Advanced, Panels, Comfort Panels)

Scheduled task

In this example, create a user-defined VB function which converts the values of a temperature sensor from Fahrenheit to degrees Celsius. The temperature is displayed in an output field on the HMI device.

Settings

For the example you need two HMI tags and a user-defined VB function with the following settings:

HMI tags:

Name	PLC connection	Type
Fahrenheit	Yes	Real
Centigrade	No	Real

User-defined VB function:

Name	Type	Parameters
FahrenheitToCelsius	Function	Fahrenheit

Procedure

1. Create the two HMI-tags "Fahrenheit" and "Celsius" with the settings named above.

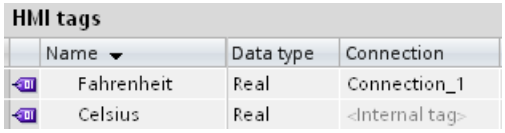

2. Create the user-defined VB function "FahrenheitToCelsius" with the settings given above.

3. Write the following VBS code:
   `FahrenheitToCelsius = (Fahrenheit-32) * (5/9)`

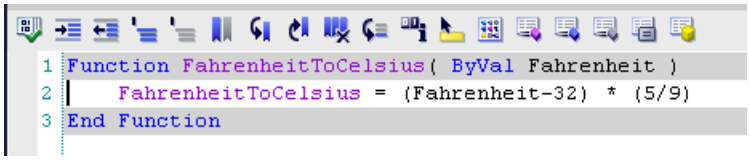

Interim result

The HMI tags and the VB user-defined function are created.

Procedure

1. Open the "HMI Tag"editor and click on the "Fahrenheit" tag.
2. Click the "Properties > Events" tab in the Inspector window. Select the "Value change" event.
3. Configure the user-defined VB function "FahrenheitToCelsius" to the "Value change" event.
4. Select the tag "Fahrenheit" for the parameter "Fahrenheit".
5. Select the HMI-tag "Celsius" for the return value "Fahrenheit to Celsius".
6. Configure an output field in a screen and connect it with the tag "Celsius".

Alternative procedure

Instead of using a user-defined VB function of the "Function" type, you can also use the "Sub" type. In this case, assign the converted value directly to the HMI-tag "Celsius".

```
SmartTags("Celsius") = (Fahrenheit-32) * (5/9)
```

In this case the return value of the user-defined VB function is not applicable.

Result

When the tag value of "Fahrenheit" changes in Runtime, the user-defined VB function "Fahrenheit to Celsius" is performed. The converted temperature value is returned to the HMI-tag "Celsius" and displayed in the output field.

9.10.1.2 Example: Converting inches into meters (RT Advanced, Panels, Comfort Panels)

Scheduled task

In this example you create a user-defined VB function that converts the value of a length measuring system from inches into meters. The length in meters is displayed on the HMI device in an output field.

Settings

For the example you need two HMI tags and a user-defined VB function with the following settings:

HMI tags:

Name	PLC connection	Type
VarInch	Yes	Real
VarMeter	No	Real

Customized VB function

Name	Type	Parameters
InchToMeter	Sub	Inch

Procedure

1. Create the two HMI tags "VarInch" and "VarMeter" with the above settings.

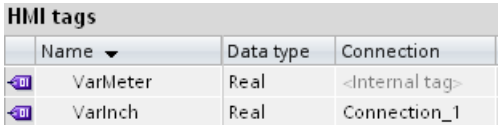

2. Create the user-defined VB function "InchToMeter" with the settings given above.

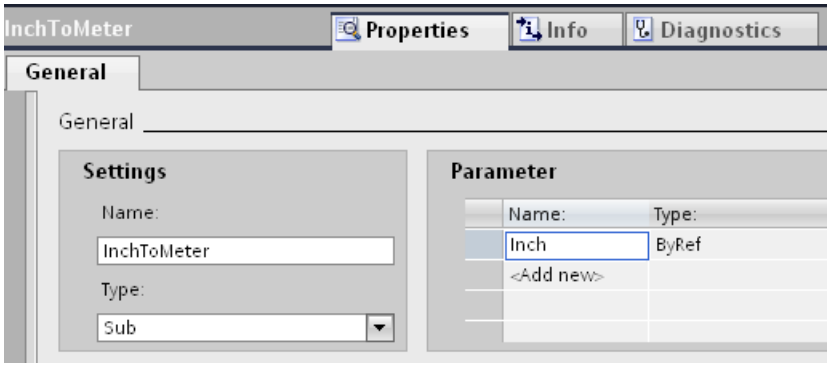

3. Write the following VBS code:
```
VarMeter = Inch*0.0254
```

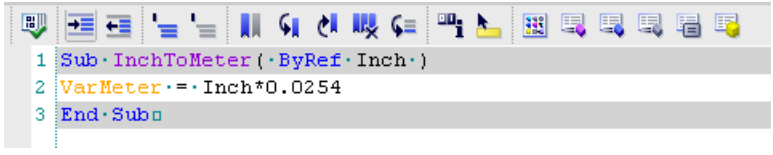

Interim result

The tags and the user-defined VB function are created.

Procedure

1. Open the "HMI tags" editor and click the "VarInch" tag.
2. Click "Properties> Events" in the Inspector window. Select the "Value change" event.
3. Configure the user-defined VB function "InchToMeter" to the "Value change" event.
4. Select the "VarInch" HMI tag for the "Inch" parameter.
5. Configure an output field in a screen and connect it with the "VarMeter" HMI tag.

Result

When the tag value of "VarInch" changes in Runtime, the "InchToMeter" user-defined VB function is performed. The calculated value is written into the "VarMeter" HMI tag and displayed in the output field.

9.10.1.3 Example: Changing the operating mode on the HMI device with the current display (RT Advanced, Panels, Comfort Panels)

Scheduled task

In this example, you use the "SetDeviceMode" system function to switch between the "online" and "offline" modes on the HMI device. You also display the current set operating mode on the HMI device.

Requirements

A screen has been created.

Settings

For this example you require a HMI-tag and a text list with the following settings:

HMI tag:

Name	PLC connection	Type
OperatingMode	No	Bool

Text list:

Name	Contains	Values
ShowOperatingMode	Bit (0/1)	1: Operating mode: "Online" 0: Operating mode: "Offline"

Procedure

1. Create the "OperatingMode" HMI-tag indicated above.

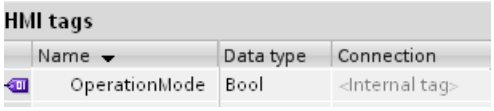

2. Create the "ShowOperatingMode" text list indicated above.

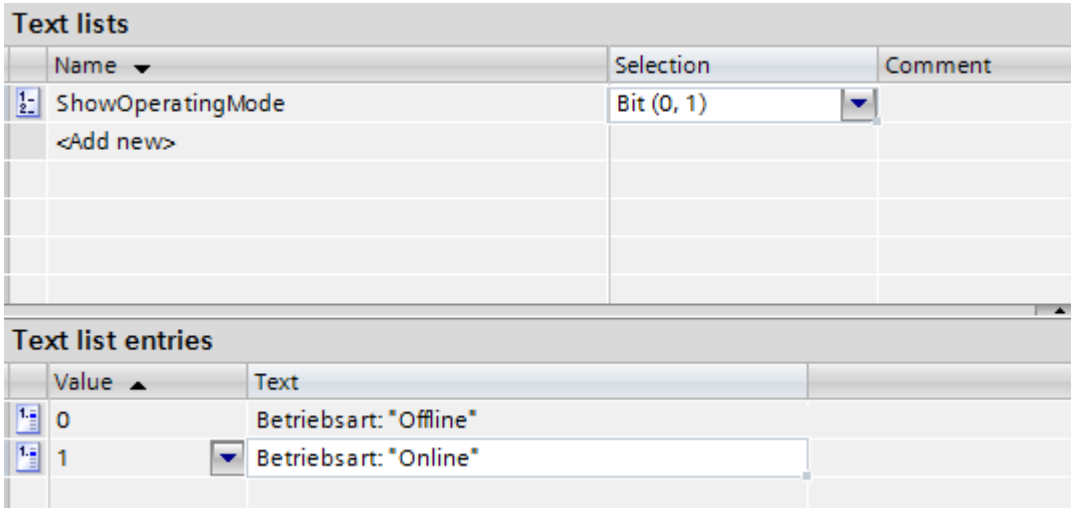

3. Open the screen and insert a button for which you configure the operating mode change to "online".
4. Click "Properties> Events" in the Inspector window. Select the "Press" event.
5. Configure the "SetDeviceMode" system function for the "Press" event. The system function is found in the selection list under "Settings".
6. For the "Mode" parameter, select the "Online" entry.
7. Configure the system function "SetBit" on the event "Press". The system function is found in the selection list under "Bit processing".
8. Select the HMI-tag "Operating mode" from the selection list for the parameter "Tag".

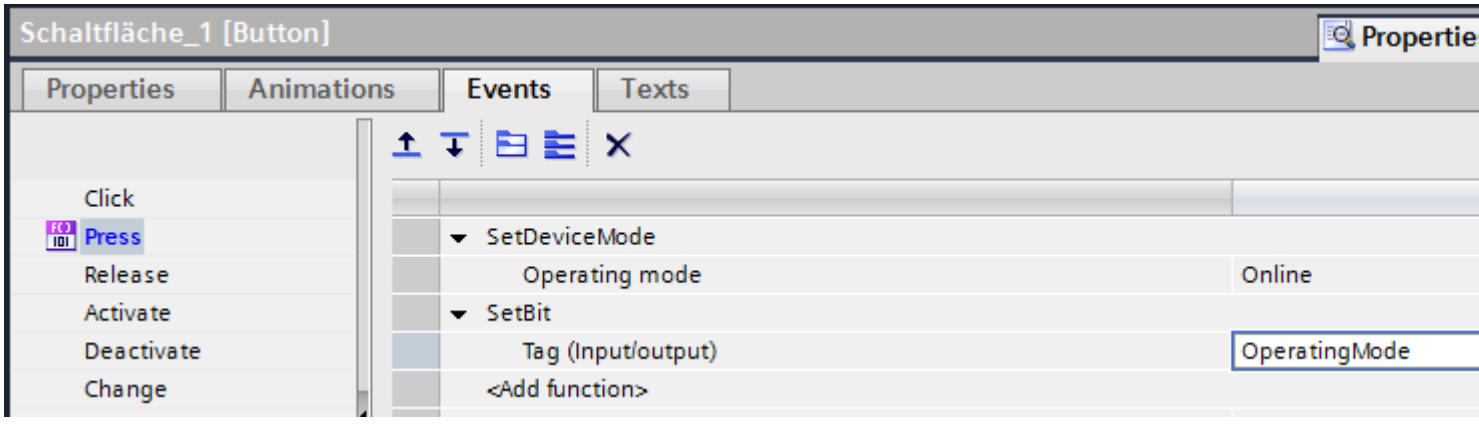

9. Add a button in the process screen for which you configure the operating mode change to "offline".
10. Repeat steps four to seven. For the "Mode" parameter, select the "Offline" entry. Configure the system function "ResetBit" in place of the system function "SetBit."

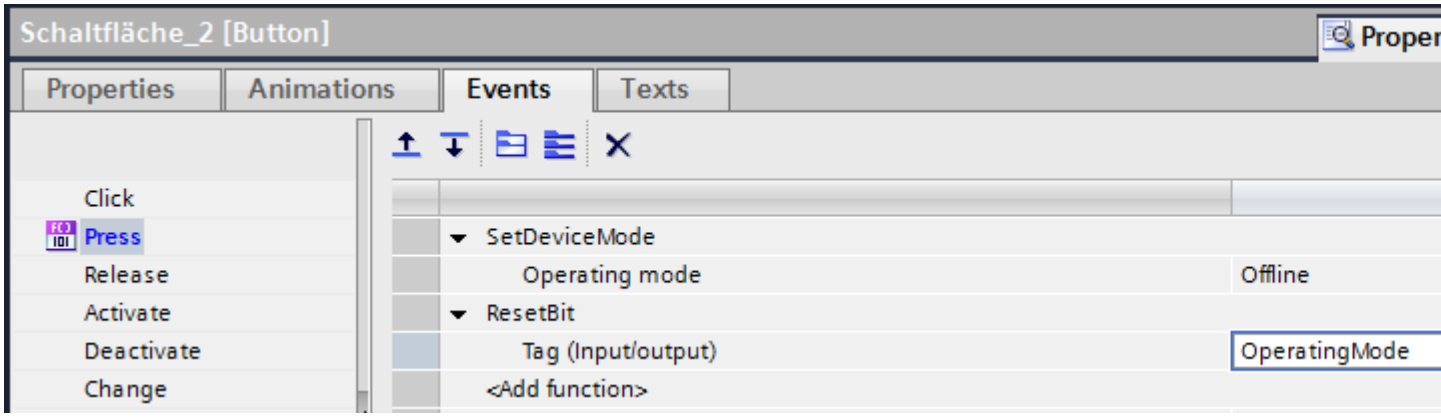

Interim result

You can toggle the operating mode of the HMI device with the two buttons in Runtime.

You want to display the current set operating mode in an output field on the HMI device.

Procedure

1. Create a "Symbolic I/O field" in the process image. Click "Properties > Properties" in the Inspector window.
2. Make the following settings in the "General" group:
   - Select "Output" as the "Mode".
   - Select the text list "Show operating mode" as "Text list".
   - Select "Operating mode" as "Tag".

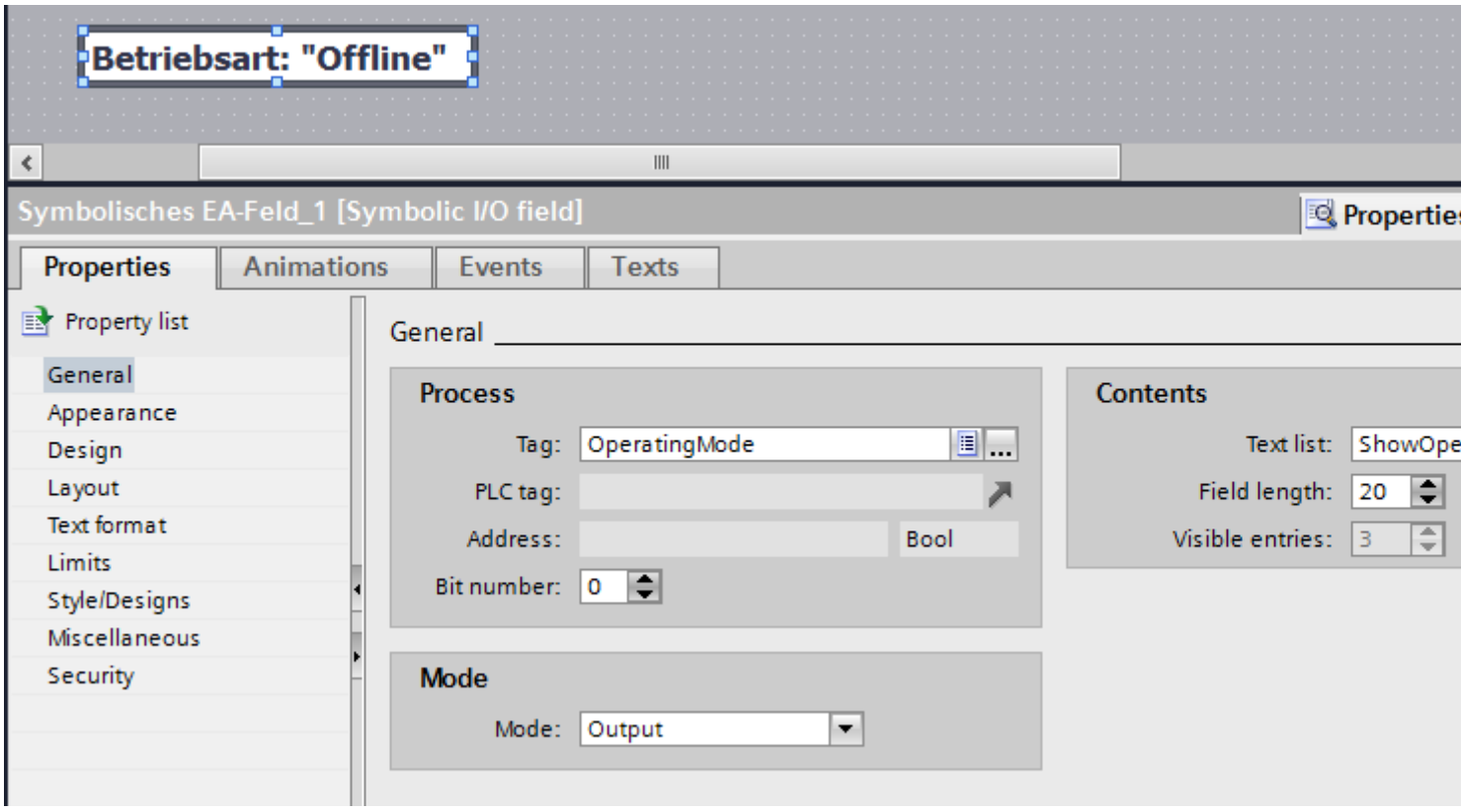

Result

When you change the operating mode with the buttons, the currently set operating mode on the HMI device is always shown.

See also

Configuring a function list (Page 1380)

9.10.2 Examples of VBS

9.10.2.1 Overview (RT Professional)

Introduction

You can find the following examples for VBS:

- Access to objects in the "Screens" editor (e.g., color or text change)
- Determining color of objects
- Configuring a language switch
- Deactivating Runtime
- Configuring a screen change via a property
- Configuring diagnostic output via a trace
- Writing tag values
- Reading tag values
- Writing object properties
- Configuring a recipe view
- Configuring an alarm view
- Drawing a trend
- Evaluating tag values statistically

See also

Example of configuring the recipe view (Page 1462)

Example of configuring the alarm view (Page 1463)

Example of statistically evaluating tag values (Page 1465)

9.10.2.2 Example of accessing objects in the Screens editor (RT Professional)

Introduction

To make the graphic Runtime environment dynamic, you can use VBS scripting to access all objects in the "Screens" editor in WinCC. In doing so, you can make graphic objects dynamic in response to an operator input, e.g., in response to a mouse click or depending on tag values.

Procedure

In the following example, the radius of a circle is set to 20 in Runtime per mouse click:

```
'VBS121
Dim objCircle
Set objCircle= ScreenItems("Circle1")
objCircle.Radius = 20
```

9.10.2.3 Example of defining object colors (RT Professional)

Introduction

The object colors are defined by means of RGB values (Red/Green/Blue). The color values for graphic objects can be set or read.

Procedure

The following example defines the fill color for "ScreenWindow1" to blue:

```
'VBS122
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
objScreen.FillColor = RGB(0, 0, 255)
```

9.10.2.4 Example of configuring a language switch (RT Professional)

Introduction

The Runtime language can be changed by means of VBS. Use buttons with corresponding country markings that you place on the start page of your project.

Using VBS, define the Runtime language by means of country code, e.g., 1031 for German - Default, 1033 for English - USA, etc. An overview of all country codes is available in the Basics of VBScript, "Regional Scheme ID (LCID) Diagram".

Procedure

Configure a local VBS script for a "Mouse click" event on a button and enter the following code to change the Runtime language to German:

```
'VBS123
HMIRuntime.Language = 1031
```

9.10.2.5 Example of deactivating Runtime (RT Professional)

Introduction

You can use VBS to exit Runtime, e.g. by means of mouse click, or depending on tag values, or on other events.

Procedure

The following example terminates WinCC Runtime:

```
'VBS124
HMIRuntime.Stop
```

9.10.2.6 Example of configuring a screen change via a property (RT Professional)

Introduction

If using split screens in your configuration, e.g. the title and operating bar for the user interface in a basic screen and an embedded screen window for the actual screen display, configure a screen change using the properties of the screen window.

The property of screen window "ScreenName" must be changed so that the other screen is displayed. The local script and the screen window must be configured in the same screen.

Procedure

In the following example you show the "test" screen in a "ScreenWindow" when executing the local VB script:

```
'VBS126
Dim objScrWindow
Set objScrWindow = ScreenItems("ScreenWindow")
objScrWindow.ScreenName = "test"
```

9.10.2.7 Example of configuring a diagnostics output via a trace (RT Professional)

Introduction

If you have inserted the Print job/Script diagnostics" object in your screen, you can use the trace command to display diagnostics outputs in the diagnostics window in Runtime.

Print job/Script diagnostics returns the trace methods contained in the user-defined functions in the order in which they were called. This also applies to trace instructions in functions that are called in user-defined functions. The targeted use of trace instructions allows you to trace

the flow of user-defined functions and the functions called therein. You enter the trace instruction in the form "HMIRuntime.Trace<Output>".

The "Print job/Script diagnostics" object displays trace outputs from C and VBS.

Outputting text

The following example writes a text in the diagnostics window:

```
'VBS127
HMIRuntime.Trace "Customized error message"
```

Outputting alarms

The following example writes the alarms of a "Control2" alarm view to the diagnostics window:

```
Sub TraceMsg
  Dim objAlarmControl
  Dim lIndex
  Dim lCellIndex

  'create reference to the alarm control
  Set objAlarmControl = ScreenItems("Control2")

  'enumerate and trace out row numbers
  For lIndex = 1 To objAlarmControl.GetRowCollection.Count
     HMIRuntime.Trace "Row: " & (objAlarmControl.GetRow(lIndex).RowNumber) & "  "

     'enumerate and trace out column titles and cell texts
     For lCellIndex = 1 To objAlarmControl.GetRow(lIndex).CellCount
       HMIRuntime.Trace objAlarmControl.GetMessageColumn(lCellIndex -1).Name & " "
       HMIRuntime.Trace objAlarmControl.GetRow(lIndex).CellText(lCellIndex) & "  "
     Next
     HMIRuntime.Trace vbNewLine
  Next
End Sub
```

Output information about HMI tags

The following example uses the "VarArch100" log tag to write information to the diagnostics window:

```
Sub ReadTagInfo
  Dim Tag
  Set Tag = HMIRuntime.Tags("TagLoggingfast\VarArch100")

  HMIRuntime.Trace "Last Error = " + CStr(Tag.LastError) + vbNewLine
  HMIRuntime.Trace "Name = " + Tag.Name + vbNewLine
  HMIRuntime.Trace "QualityCode = " + CStr(Tag.QualityCode) + vbNewLine
End Sub
```

See also

Diagnostics (Page 1435)

9.10.2.8 Example of writing tag values (RT Professional)

Introduction

You can use VBScript to write the tag value to the PLC, for example, by mouse click on a setpoint button or to set internal tag values for triggering other user-defined functions.

Various write variations are mentioned and explained below.

Simple writing

In the following example, a value is written to the "Tag1" tag:

```
'VBS128
HMIRuntime.Tags("Tag1").Write 6
```

This is the simplest form of writing since no object reference is generated.

Writing with object reference

In the following example, a local copy of the tag object is generated and a value is written to "Tag1":

```
'VBS129
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Write 7
```

Referencing offers the advantage of being able to work with the tag object before writing. You can read the tag value, perform calculations, and write the tag value again.

```
'VBS130
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
objTag.Value = objTag.Value + 1
objTag.Write
```

Synchronous writing

The value to be written is usually transferred to tag management and execution of the script is continued. In certain cases, however, it must be ensured that the value has actually been written before script execution continues.

This type of writing is realized by specifying the value 1 for the additional, optional parameters:

```
'VBS131
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Write 8.1
```

or

```
'VBS132
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Value = 8
objTag.Write ,1
```

Note

Note that the call takes longer in comparison to the standard call. The duration is also dependent on the channel and AS, among other things.

This type of writing corresponds to the SetTagXXXWait() call in C scripting.

Writing with status handling

In order to ensure that a value has been written successfully, you perform an error check or evaluate the tag status after the write operation was completed.

This is done by checking the value of the LastError property after writing. The tag status is checked if the check was completed successfully, i.e., if the job was successfully output.

In the case of a write job, the current status from the process is not determined. To determine this, it is necessary to read the tag. The value specified in the QualityCode property after the read process provides an indication of the tag status and, if necessary, makes reference to a failed AS connection.

In the following example, the "Tag1" tag is written. If an error occurs during writing, the error value and error description appear in the Global Script diagnostics window. Finally, the

QualityCode is checked. If the QualityCode is not OK (0x80), it will be displayed in the diagnostics window.

```
'VBS133
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Write 9
If 0 <> objTag.LastError Then
  HMIRuntime.Trace "Error: " & objTag.LastError & vbCrLf & "ErrorDescription: " &
objTag.ErrorDescription & vbCrLf
Else
  objTag.Read
  If &H80 <> objTag.QualityCode Then
     HMIRuntime.Trace "QualityCode: 0x" & Hex(objTag.QualityCode) & vbCrLf
  End If
End If
```

Note

After writing a tag, the QualityCode property of the local tag object is set to "BAD Out of Service" because it is not known which QualityCode the tag handles in the process. The new QualityCode can be determined by reading again.

The Quality Code cannot be written from VBS.

9.10.2.9 Example of reading tag values (RT Professional)

Introduction

VBS can be used to read and further process a tag value. This allows you to click a button, for example, to obtain information on the system status or to perform calculations.

Various read variations are mentioned and explained below.

Simple reading

In the following example, the value of "Tag1" is read and displayed in the Global Script diagnostics window:

```
'VBS134
HMIRuntime.Trace "Value: " & HMIRuntime.Tags("Tag1").Read & vbCrLf
```

This is the simplest form of reading since no object reference is generated.

Reading with object reference

In the following example, a local copy of the tag object is created, and the tag value is read and displayed in the "Print job/Script diagnostics" object:

```
'VBS135
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
HMIRuntime.Trace "Value: " & objTag.Read & vbCrLf
```

Referencing offers the advantage of being able to work with the tag object. You can read the tag value, perform calculations, and write the tag value again.

```
'VBS136
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
objTag.Value = objTag.Value + 1
objTag.Write
```

Using the Read method, process tags that have been read are added to the image, from this moment on they are cyclically requested from the automation system (AS). If the tag is already in the image, the value it contains is returned.

When the screen is exited, the tags are logged off.

Note

If a tag is requested in a function, it remains logged on as long as WinCC Runtime is running.

Direct reading

Normally, the tag values are read from the tag image. In certain situations, however, it may be necessary to read the value directly from the AS, e.g., to synchronize fast processes.

The tag is not logged on cyclically if you set the optional parameter to 1 when reading the value. Instead, the value is requested once from the AS.

```
'VBS137
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
HMIRuntime.Trace "Value: " & objTag.Read(1) & vbCrLf
```

Note

Note that the call takes longer in comparison to the standard call. The duration is also dependent on the channel and AS, among other things.

This type of call must be avoided in cyclic C scripts as this is the major cause of performance problems.

This read mode corresponds to the GetTagXXXWait() call used in C scripting.

Reading with status handling

Always validate the value after reading it. This occurs by checking the QualityCode.

In the following example, the "Tag1" tag is read and the QualityCode then checked. When the Quality Code does not correspond to OK (0x80) the LastError, ErrorDescription, and QualityCode properties are displayed in the Global Script diagnostics window.

```
'VBS138
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
If &H80 <> objTag.QualityCode
  Then
    HMIRuntime.Trace "Error: " & objTag.LastError & vbCrLf & "ErrorDescription: " &
objTag.ErrorDescription & vbCrLf & "QualityCode: 0x" & Hex(objTag.QualityCode) & vbCrLf
  Else
    HMIRuntime.Trace "Value: " & objTag.Value & vbCrLf
End If
```

Note

If an error occurs during reading, QualityCode is set to BAD Out of Service". Therefore, it is sufficient to check the QualityCode following reading.

9.10.2.10 Example of writing object properties (RT Professional)

Introduction

VBS provides access to the properties of all display and operating objects in the "Screens" editor. You can read out properties or change them during Runtime.

The following examples illustrate various forms of access.

Simple setting of a property

In the next example, the background color of the "Rectangle1" object in the screen is set to red:

```
'VBS139
ScreenItems("Rectangle1").BackColor = RGB(255,0,0)
```

This is the simplest form of writing since no object reference is generated.

Note

If you are working without an object reference, only the standard properties are provided by autocompletion.

The mode of expression used in this example can only be executed in the "Screens" editor. By analogy, you address the objects in the scheduler using the HMIRuntime object.

Setting a property with object reference

In the following example, a reference is created to the "Rectangle1" object contained in the picture and the background is set to red using the VBS standard function RGB():

```
'VBS140
Dim objRectangle
Set objRectangle = ScreenItems("Rectangle1")
objRectangle.BackColor = RGB(255,0,0)
```

Additional information on RGB is available in "VBScript for Windows".

Referencing is useful if you want to edit several object properties. In this way all of the properties of the object will be listed with the autocompletion feature.

Note

The mode of expression used in this example can only be executed in the "Screens" editor. By analogy, you address the objects in the scheduler using the HMIRuntime object.

Setting properties across screen windows

VBS in the "Screens" editor offers two options of global screen addressing:

- Using the screen object of a screen window with "ScreenItems"
- From the root screen with "HMIRuntime.Screens"

Referencing via the screen window

In the following example, the color of a rectangle is changed in an subordinate screen window. The user-defined VB function is executed in the "BaseScreen" screen in which "ScreenWindow1" is located. The screen window displays a screen, which contains an object of the type "Rectangle" with the name "Rectangle1".

```
'VBS199
Sub OnLButtonUp(ByVal Item, ByVal Flags, ByVal x, ByVal y)
  Dim objRectangle
  Set objRectangle = ScreenItems("ScreenWindow1").Screen.ScreenItems("Rectangle1")
  objRectangle.BackColor = RGB(255,0,0)
End Sub
```

Referencing from the root screen

You can reference the screen with the object to be modified via HMIRuntime.Screens. The specification of the screen is defined relative to the root screen via the following access code:

[<Root screen name>.]<Screen window name>[:<Screen name>]... .<Screen window name>[:<Screen name>]

In the following example, a reference is created to the "Screen2" object contained in the "Rectangle1" picture and the background color is set to red.

The screen "Screen2", in this case, is in "Screen1". "Screen1" is displayed in the "BaseScreen" root screen.

```
'VBS141
Dim objRectangle
Set objRectangle =
HMIRuntime.Screens("BaseScreen.ScreenWindow1:Screen1.ScreenWindow1:Screen2").ScreenItems("
Rectangle1")
objRectangle.BackColor = RGB(255,0,0)
```

It is not necessary to specify the screen name. It is possible to address a screen uniquely using the screen window name. Therefore, it is sufficient to specify the name of the screen window, as in the following example:

```
'VBS142
Dim objRectangle
Set objRectangle =
HMIRuntime.Screens("ScreenWindow1.ScreenWindow2").ScreenItems("Rectangle1")
objRectangle.BackColor = RGB(255,0,0)
```

This type of addressing enables objects in screen windows to be addressed in different pictures. This is a particularly interesting aspect for the faceplate system.

Making the property dynamic using the return value

In addition to initiating local scripts for properties by means of an event trigger or cyclic trigger, you can also make properties dynamic directly by means of a local script.

In the next example, the background color of an object made dynamic by means of return value. The value transferred can be derived from the evaluation of events in the PLC, for example, and can be used for graphic visualization of an operating state:

```
'VBS146
Function BackColor_Trigger(ByVal Item)
  BackColor_Trigger = RGB(125,0,0)
End Function
```

Note

If you use a user-defined function to make an object property dynamic by means of the return value of a user-defined function, the value of the object property is only written if it has changed compared to the last function session. It is not considered if the value had been changed from another location.

For this reason, properties that have been made dynamic by means of the return value with user-defined functions must not be changed from another location (e.g. other user-defined functions). Incorrect values could be displayed if you ignore this rule.

9.10.2.11 Example of configuring the recipe view (RT Professional)

Introduction

Recipe data records are output in the recipe view.

You can add, copy, edit or delete data records.

Loading and displaying a recipe

The following example loads the "Juice" recipe in the "Control1" recipe view. The recipe is displayed in the recipe view.

```
Sub loadUserArchive()
  Dim objUAC
  Set objUAC = ScreenItems("Control1")
  objUAC.BorderColor = RGB(255,0,0)
  objUAC.BorderWidth = 3
  objUAC.Closeable = False
  objUAC.Datasource = "Juice"
End Sub
```

Copying a recipe data record

The following example copies the recipe data record of line 3. The copy is inserted at the end of the table.

```
Sub CopyRow
  Dim objUAC
  Set objUAC = ScreenItems("Control1")

  Dim iRowCount

  objUAC.AutoCompleteRows = True

  objUAC.SelectRow(3)
  objUAC.CopyRows

  iRowCount = objUAC.GetRowCollection.Count
  objUAC.SelectRow(iRowCount + 1)
  objUAC.PasteRows
  objUAC.SelectRow(1)
  objUAC.UnselectAll
End Sub
```

See also

Overview (Page 1451)

9.10.2.12 Example of configuring the alarm view (RT Professional)

Introduction

The alarm view displays alarms. You can show, hide, sort, and filter alarms.

Filtering alarms

In this example, only the alarms of alarm number 2 are displayed in the "Control2" alarm view.

```
Sub Filter
  Dim objAlarmControl

  'create reference to AlarmControl
  Set objAlarmControl = ScreenItems("Control2")

  'set / reset the filter and create a trace
  If (objAlarmControl.MsgFilterSQL = "") Then
     objAlarmControl.MsgFilterSQL = "MSGNR = 2"
  Else
     objAlarmControl.MsgFilterSQL = ""
  End If
End Sub
```

Removing alarms

In this example, alarms 1-3 will be removed from the alarm view.

```
Sub ResetAlarm
  Dim objAlarm
  'reset alarm 1 - 3
  Set objAlarm = HMIRuntime.Alarms(1)
  objAlarm.State = 2 'hmiAlarmStateGone
  objAlarm.Create
  Set objAlarm = HMIRuntime.Alarms(2)
  objAlarm.State = 2 'hmiAlarmStateGone
  objAlarm.Create
  Set objAlarm = HMIRuntime.Alarms(3)
  objAlarm.State = 2 'hmiAlarmStateGone
  objAlarm.Create
End Sub
```

Adding alarms

In this example, alarms 1-3 will be added to the alarm view. Alarms 1-3 must be created.

```
Sub SetAlarm
  Dim objAlarm

  'create Alarm 1 - 3
  Set objAlarm = HMIRuntime.Alarms(1)
  objAlarm.State = 1 'hmiAlarmStateCome
  objAlarm.Create
  Set objAlarm = HMIRuntime.Alarms(2)
  objAlarm.State = 1 'hmiAlarmStateCome
  objAlarm.Create
  Set objAlarm = HMIRuntime.Alarms(3)
  objAlarm.State = 1 'hmiAlarmStateCome
  objAlarm.Create
End Sub
```

See also

Overview (Page 1451)

9.10.2.13 Example of statistically evaluating tag values (RT Professional)

Introduction

You can use the value table to display the coordinates of trends and the statistical evaluation of tag values in tabular form. You connect the value table to the corresponding object for this purpose. The evaluated data can be printed out and exported.

Procedure

In the following example, the tag values of the f(t) trend view "Control2" are statistically evaluated. To display the result in the statistics area window of the value table, you set the following in the value table properties:

- Data source: "Control2"
- Mode: "Statistic area" window"

Add a new button in f(t) trend view and configure the user-defined VB function for it.

```
Sub OnToolbarButtonClicked(ByVal Item,  ByVal lId)
  Dim objTrend
  Set objTrend = ScreenItems("Control2")
  objTrend.CalculateStatistic()
End Sub
```

See also

Overview (Page 1451)

9.10.2.14 Example of the start of a script at the server (Logging object) (RT Professional)

Introduction

In mulit-user projects, the Logging object currently works only on the server. The following examples shows how you start a script from the client and thus also store and delete log segments at the client.

The following example shows a global script that is started with a control tag. The content of the control tag determines whether the method "Restore" or the method "Remove" is called. The control tag is set to "0" at the end of the script. A query prevents the script from starting on client computers. Path and period are transferred by internal tags.

```
'VBS180
'StartLoggingOnServer
Dim StartLogging
Dim SourcePath
Dim TimeFrom
Dim TimeTo
Dim RetVal
'Exit when running on client
If (Left(HMIRuntime.ActiveProject.Path, 1) = "\") Then
Exit Function
End If
'read parameters
StartLogging = HMIRuntime.Tags("StartLogging").Read
SourcePath = HMIRuntime.Tags("SourcePath").Read(1)
TimeFrom = HMIRuntime.Tags("TimeFrom").Read(1)
TimeTo = HMIRuntime.Tags("TimeTo").Read(1)
'restore or remove depends on the parameter
If (StartLogging = 1) Then
RetVal = HMIRuntime.Logging.Restore(SourcePath, TimeFrom, TimeTo, -1)
HMIRuntime.Tags("RetVal").Write RetVal, 1
HMIRuntime.Tags("StartLogging").Write 0,1
Elseif (StartLogging = 2) Then
RetVal = HMIRuntime.Logging.Remove(TimeFrom, TimeTo, -1)
HMIRuntime.Tags("RetVal").Write RetVal, 1
HMIRuntime.Tags("StartLogging").Write 0,1
End If
```

A network enable can also be specified as path. The log segments to be stored do not, therefore, have to be located locally on the server. However, it must be ensure that the server can access the path correctly.

Note

The example shows a proposed solution and is to be adjusted if necessary.

Procedure

1. Create the following internal tags with project-wide update in the WinCC Explorer:
   - StartLogging(Unsigned 8-bit value)
   - SourcePath(Text tag, 8-bit character set)
   - TimeFrom(Text tag, 8-bit character set)
   - TimeTo (Text tag, 8-bit character set)
   - RetVal(Signed 32-bit value)
2. Create the script "StartLoggingOnServer".
3. Configure the tag "StartLogging" the script "StartLoggingOnServer" at the event "Value change".

You can start the script on the client, for example, with the following script:

```
'VBS181
'Prerequisite: set parameters
HMIRuntime.Tags("SourcePath").Write "\\client_pc\temp",1
HMIRuntime.Tags("TimeFrom").Write "2004",1
HMIRuntime.Tags("TimeTo").Write "2004",1
'start action
HMIRuntime.Tags("StartLogging").Write 1,1
```

Note

The tags are mainly written and read in "direct" mode. Thus, the processes are synchornized. As these are internal tags, this mode can be used without hesitation.

9.10.2.15 Example for opening the PLC code view from the GRAPH overview (RT Professional)

Introduction

With VBS, even when using screen windows, you have the option of switching from the GRAPH overview to the PLC code view.

For this purpose, you create a VB script at a graph overview, a screen with a screen window, and an additional screen with a PLC code view that shows the corresponding program code.

Opening PLC code view from a screen window

In the following example, the PLC code view "PLC code view_1" in the screen "CodeViewScreen" is called with an event from the GRAPH overview in the screen window "Screen window_1".

```
Sub OnPlcCodeViewButtonClick(ByVal item, ByVal codeViewerDiagnosticsContext)
  Dim objScreenWindow
  Set objScreenWindow = ScreenItems("Screen window_1")
  objScreenWindow.ScreenName = "CodeViewScreen"
  objScreenWindow.Screen.ScreenItems("PLC code view_1").NavigateTo =
codeViewerDiagnosticsContext
End Sub
```

9.10.3 Examples of C (RT Professional)

9.10.3.1 Overview (RT Professional)

Introduction

You can find examples for C functions:

- Access to objects in the "Screens"editor (e.g. color or text change)
- Determining color of objects
- Configuring a language switch
- Deactivating Runtime
- Configuring a screen change via a property
- Configuring diagnostic output via a trace
- Writing tag values
- Reading tag values
- Writing object properties

See also

Example of accessing objects in the Screens editor (Page 1469)
Example of defining object colors (Page 1469)
Example of configuring a language switch (Page 1470)
Example of deactivating Runtime (Page 1470)
Example of configuring a screen change via a property (Page 1471)
Example of configuring a diagnostics output via a trace (Page 1471)
Example of writing tag values (Page 1472)

Example of reading tag values (Page 1472)

Example of writing object properties (Page 1473)

9.10.3.2 Example of accessing objects in the Screens editor (RT Professional)

Introduction

To make the graphic Runtime environment dynamic, you can use C scripting to access all objects in the "Screens" editor in WinCC. In doing so, you can make graphic objects dynamic in response to an operator input, e.g., in response to a mouse click or depending on tag values.

Procedure

In the following example, the radius of a circle is set to 10 in Runtime per mouse click:

```
'C121
SetPropDouble(screenName,"Kreis_1","Radius",10);
```

See also

Overview (Page 1468)

9.10.3.3 Example of defining object colors (RT Professional)

Introduction

The object colors are defined by means of RGB values (Red/Green/Blue). The color values for graphic objects can be set or read.

Procedure

The following example defines the fill color for "Roundbutton1" to blue:

```
'C122
SetPropertyByConstant (screenName, " Roundbutton1", "FillPatternColor", "65035");
```

See also

Overview (Page 1468)

9.10.3.4 Example of configuring a language switch (RT Professional)

Introduction

The Runtime language can be changed by means of VBS. Use buttons with corresponding country markings that you place on the start page of your project.

Define the Runtime language by means of country code, for example, 1031 for German - Default, 1033 for English - USA etc. An overview of all country codes is available in the basics of VBScript, "Regional Scheme ID (LCID) Diagram".

Procedure

Create a user-defined C function for a button for the "Mouse click" event, and enter the following code to switch the Runtime language to German:

```
'C123
SetLanguageByLocaleID(1033);
```

See also

Overview (Page 1468)

9.10.3.5 Example of deactivating Runtime (RT Professional)

Introduction

You can use C functions to exit Runtime, for example, by means of mouse click, or depending on tag values, or on other events.

Procedure

The following example terminates WinCC Runtime:

```
'C124
StopRuntime(hmiStopRuntime);
```

See also

Overview (Page 1468)

9.10.3.6 Example of configuring a screen change via a property (RT Professional)

Introduction

If using split screens in your configuration, for example, the title and operating bar for the user interface in a basic screen and an embedded screen window for the actual screen display, configure a screen change using the properties of the screen window.

The property of screen window "ScreenName" must be changed so that the other screen is displayed. The C function and screen window must be configured in the same screen.

Procedure

In the next example, the "test" screen is displayed in a "ScreenWindow" when you execute the C function:

```
'C126
ActivateScreenInScreenWindow (screenName, "ScreenWindow", "test");
```

See also

Overview (Page 1468)

9.10.3.7 Example of configuring a diagnostics output via a trace (RT Professional)

Introduction

If you have inserted a GSC diagnostics window in your screen, you can use the trace command to display diagnostics outputs in the diagnostics window in Runtime.

GSC diagnostics returns the trace methods contained in the functions in the order in which they were called. This also applies to trace instructions in functions that are called in user-defined functions. The targeted use of trace instructions allows you to trace the execution of user-defined functions and the functions called therein. You enter the trace instruction in the form "printf(<Output>)".

The GSC diagnostics window displays trace outputs of C functions.

Outputting text

The following example writes a text in the diagnostics window:

```
'C127
printf("Customized C script error message\r\n");
```

See also

Overview (Page 1468)

9.10.3.8 Example of writing tag values (RT Professional)

Introduction

You can use C functions to write the tag value to the PLC, foe example, by mouse click on a setpoint button, or to set internal tag values for triggering other functions.

Simple writing

In the following example, a value is written to the "Tag1" tag:

```
'C128
int value;
value = 7;
SetTagSDWord("Tag1", value);
```

See also

Overview (Page 1468)

9.10.3.9 Example of reading tag values (RT Professional)

Introduction

C functions can be used to read and further process a tag value. This allows you to click a button, for example, to obtain information on the system status or to perform calculations.

Simple reading

In the following example, the value of "Tag1" is read and displayed in the Global Script diagnostics window:

```
'C134
int value;
value = GetTagGetTagSDWord("Tag1");
```

See also

Overview (Page 1468)

9.10.3.10 Example of writing object properties (RT Professional)

Introduction

C functions provide access to the properties of all display and operating objects in the "Screens" editor. You can read out properties or change them during Runtime.

The following example shows how to access a property.

Simple setting of a property

In the next example, the background color of the "Button1" object in the screen is set to red:

```
'C139
SetPropLong("Screen_1","Button1","ForeColor",65333);
```

See also

Overview (Page 1468)

9.11 Reference

9.11.1 Function list

9.11.1.1 Device dependency (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

System functions for Basic Panels (Basic Panels)

Availability of system functions

The following table shows the availability of the system functions on the Basic Panels.

Technical data subject to change.

Overview

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
User-defined functions	No	No	No	No	No
Logoff (Page 1547)	Yes	Yes	Yes	Yes	Yes
ActivateScreen (Page 1505)	Yes	Yes	Yes	Yes	Yes
ActivateScreenByNumber (Page 1506)	Yes	Yes	Yes	Yes	Yes
ActivateCleanScreen (Page 1648)	No	No	Yes	Yes	Yes
ActivatePreviousScreen (Page 1507)	Yes	Yes	Yes	Yes	Yes
UpdateTag (Page 1508)	Yes	Yes	Yes	Yes	Yes
AdjustContrast (Page 1582)	Yes	No	Yes [1)]	Yes	Yes
Logon (Page 1547)	Yes	Yes	Yes	Yes	Yes
ArchiveLogFile (Page 1538)	No	No	No	No	No
LogTag (Page 1540)	No	No	No	No	No
EditAlarm (Page 1617)	Yes	Yes	Yes	Yes	No
ScreenObjectCursorDown (Page 1651)	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorUp (Page 1652)	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorLeft (Page 1654)	No	No	No	No	No
ScreenObjectCursorRight (Page 1652)	No	No	No	No	No
ScreenObjectPageDown (Page 1653)	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageUp (Page 1653)	Yes	Yes	Yes	Yes	Yes
Encode (Page 1508)	No	No	No	No	No
EncodeEx (Page 1509)	No	No	No	No	No

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
Direct key (Page 1510)	No	No	No	No	No
DirectKeyScreenNumber (Page 1513)	No	No	No	No	No
PrintScreen (Page 1570)	No	No	No	No	No
PrintReport (Page 1571)	No	No	No	No	No
NotifyUserAction (Page 1513)	No	No	No	No	No
IncreaseFocusedValue (Page 1515)	Yes	Yes	Yes	Yes	No
IncreaseTag (Page 1552)	Yes	Yes	Yes	Yes	Yes
ExportDataRecords (Page 1623)	Yes	Yes	Yes	Yes	Yes
ExportDataRecordsWithChecksum (Page 1625)	No	No	No	No	No
ExportImportUserAdministration (Page 1548)	No	No	No	No	No
GoToHome (Page 1655)	Yes	Yes	Yes	Yes	No
GoToEnd (Page 1656)	Yes	Yes	Yes	Yes	No
SafelyRemoveHardware (Page 1515)	No	No	No	No	No
HTMLBrowserStop (Page 1586)	No	No	No	No	No
HTMLBrowserScrollDown (Page 1587)	No	No	No	No	No
HTMLBrowserRefresh (Page 1587)	No	No	No	No	No
HTMLBrowserScrollUp (Page 1588)	No	No	No	No	No
HTMLBrowserForward (Page 1591)	No	No	No	No	No
HTMLBrowserBack (Page 1592)	No	No	No	No	No
HTMLBrowserZoomIn (Page 1588)	No	No	No	No	No
HTMLBrowserZoomOut (Page 1589)	No	No	No	No	No
HTMLBrowserScrollLeft (Page 1589)	No	No	No	No	No
HTMLBrowserScrollRight (Page 1589)	No	No	No	No	No
HTMLBrowserPageUp (Page 1590)	No	No	No	No	No
HTMLBrowserPageDown (Page 1590)	No	No	No	No	No
HTMLBrowserHome (Page 1591)	No	No	No	No	No
ImportDataRecords (Page 1628)	Yes	Yes	Yes	Yes	Yes
ImportDataRecordsWithChecksum (Page 1629)	No	No	No	No	No
InvertBit (Page 1557)	Yes	Yes	Yes	Yes	Yes
InvertBitInTag (Page 1559)	Yes	Yes	Yes	Yes	Yes
InvertLinearScaling (Page 1553)	Yes	Yes	Yes	Yes	Yes
CalibrateTouchScreen (Page 1583)	No	Yes	Yes	Yes	Yes
CopyLog (Page 1541)	No	No	No	No	No
TrendViewScrollForward (Page 1592)	Yes	Yes	Yes	Yes	Yes
TrendViewScrollBack (Page 1593)	Yes	Yes	Yes	Yes	Yes
TrendViewExtend (Page 1593)	Yes	Yes	Yes	Yes	Yes
TrendViewCompress (Page 1593)	Yes	Yes	Yes	Yes	Yes

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
TrendViewRulerLeft (Page 1594)	Yes	Yes	Yes	Yes	Yes
TrendViewRulerRight (Page 1595)	Yes	Yes	Yes	Yes	Yes
TrendViewSetRulerMode (Page 1595)	Yes	Yes	Yes	Yes	Yes
TrendViewStartStop (Page 1596)	Yes	Yes	Yes	Yes	Yes
TrendViewBackToBeginning (Page 1596)	Yes	Yes	Yes	Yes	Yes
LoadDataRecord (Page 1631)	Yes	Yes	Yes	Yes	Yes
GetUserName (Page 1549)	Yes	Yes	Yes	Yes	Yes
GetDataRecordFromPLC (Page 1632)	Yes	Yes	Yes	Yes	Yes
GetDataRecordName (Page 1633)	No	No	No	No	No
GetDataRecordTagsFromPLC (Page 1635)	Yes	Yes	Yes	Yes	Yes
GetGroupNumber (Page 1550)	Yes	Yes	Yes	Yes	Yes
GetBrightness (Page 1586)	No	No	No	No	No
GetPassword (Page 1550)	Yes	Yes	Yes	Yes	Yes
GetPLCMode (Page 1572)	Yes	Yes	Yes	Yes	Yes
LinearScaling (Page 1554)	Yes	Yes	Yes	Yes	Yes
ClearLog (Page 1542)	No	No	No	No	No
ClearDataRecord (Page 1636)	Yes	Yes	Yes	Yes	Yes
ClearDataRecordMemory (Page 1637)	No	No	No	No	No
ClearAlarmBuffer (Page 1617)	Yes	Yes	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 1618)	Yes	Yes	Yes	Yes	Yes
AlarmViewUpdate (Page 1619)	No	No	No	No	No
AlarmViewLoopInAlarm (Page 1619)	Yes	Yes	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 1620)	Yes	Yes	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 1620)	Yes	Yes	Yes	Yes	Yes
OpenAllLogs (Page 1543)	No	No	No	No	No
OpenScreenKeyboard (Page 1650)	No	No	No	No	No
OpenControlPanelDialog (Page 1516)	Yes [2)]	Yes [2)]	Yes [2)]	Yes [2)]	Yes [2)]
OpenCommandPrompt (Page 1517)	No	No	No	No	No
OpenInternetExplorer (Page 1517)	No	No	No	No	No
OpenControlPanel (Page 1518)	No	No	No	No	No
OpenTaskManager (Page 1650)	No	No	No	No	No
AcknowledgeAlarm (Page 1621)	Yes	Yes	Yes	Yes	No
PDFScrollDown (Page 1597)	No	No	No	No	No
PDFScrollUp (Page 1597)	No	No	No	No	No
PDFFitToWidth (Page 1598)	No	No	No	No	No
PDFFitToHeight (Page 1598)	No	No	No	No	No
PDFGoToFirstPage (Page 1599)	No	No	No	No	No
PDFGoToLastPage (Page 1599)	No	No	No	No	No
PDFGoToNextPage (Page 1600)	No	No	No	No	No

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
PDFGoToPage (Page 1600)	No	No	No	No	No
PDFGoToPreviousPage (Page 1601)	No	No	No	No	No
PDFZoomIn (Page 1601)	No	No	No	No	No
PDFZoomOut (Page 1601)	No	No	No	No	No
PDFScrollLeft (Page 1602)	No	No	No	No	No
PDFScrollRight (Page 1602)	No	No	No	No	No
PDFZoomOriginal (Page 1603)	No	No	No	No	No
RecipeViewNewDataRecord (Page 1638)	Yes	Yes	Yes	Yes	Yes
RecipeViewGetDataRecordFromPLC (Page 1638)	Yes	Yes	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 1639)	Yes	Yes	Yes	Yes	Yes
RecipeViewMenu (Page 1639)	Yes	Yes	Yes	Yes	Yes
RecipeViewOpen (Page 1640)	Yes	Yes	Yes	Yes	Yes
RecipeViewSetDataRecordToPLC (Page 1640)	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 1641)	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 1641)	Yes	Yes	Yes	Yes	Yes
RecipeViewSynchronizeDataRecordWith-Tags (Page 1642)	Yes	Yes	Yes	Yes	Yes
RecipeViewRenameDataRecord (Page 1642)	Yes	Yes	Yes	Yes	Yes
RecipeViewShowOperatorNotes (Page 1643)	Yes	Yes	Yes	Yes	Yes
RecipeViewBack (Page 1643)	Yes	Yes	Yes	Yes	Yes
ResetBit (Page 1561)	Yes	Yes	Yes	Yes	Yes
ResetBitInTag (Page 1562)	Yes	Yes	Yes	Yes	Yes
PressButton (Page 1521)	Yes	Yes	Yes	Yes	No
ReleaseButton (Page 1522)	Yes	Yes	Yes	Yes	No
ShiftAndMask (Page 1564)	Yes	Yes	Yes	Yes	Yes
CloseAllLogs (Page 1543)	No	No	No	No	No
SetDataRecordToPLC (Page 1644)	Yes	Yes	Yes	Yes	Yes
SetDataRecordTagsToPLC (Page 1645)	Yes	Yes	Yes	Yes	Yes
PageDown (Page 1654)	Yes	Yes	Yes	Yes	No
PageUp (Page 1655)	Yes	Yes	Yes	Yes	No
SendEMail (Page 1519)	No	No	No	No	No
SetAcousticSignal (Page 1584)	No	No	No	No	No
SetDisplayMode (Page 1573)	No	No	No	No	No
SetDeviceMode (Page 1573)	Yes	Yes	Yes	Yes	Yes
SetBit (Page 1565)	Yes	Yes	Yes	Yes	Yes
SetBitInTag (Page 1567)	Yes	Yes	Yes	Yes	Yes

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
SetBitWhileKeyPressed (Page 1569)	Yes	Yes	Yes	Yes	Yes
SetBacklightColor (Page 1585)	Yes	Yes	No	No	No
SetBrightness (Page 1584)	No	Yes	Yes	No	No
SetScreenKeyboardMode (Page 1574)	No	No	No	No	No
SetAlarmReportMode (Page 1571)	No	No	No	No	No
SetRecipeTags (Page 1645)	Yes	Yes	Yes	Yes	Yes
SetDaylightSavingTime (Page 1520)	No	No	No	No	No
SetLanguage (Page 1576)	Yes	Yes	Yes	Yes	Yes
SetTag (Page 1555)	Yes	Yes	Yes	Yes	Yes
SetConnectionMode (Page 1577)	Yes	Yes	Yes	Yes	Yes
SetWebAccess (Page 1578)	No	No	No	No	No
BackupRAMFileSystem (Page 1522)	No	No	No	No	No
SimulateSystemKey (Page 1523)	Yes	Yes	Yes	Yes	No
SimulateTag (Page 1524)	Yes	Yes	Yes	Yes	Yes
SmartClientViewRefresh (Page 1610)	No	No	No	No	No
SmartClientViewReadOnlyOff (Page 1611)	No	No	No	No	No
SmartClientViewReadOnlyOn (Page 1611)	No	No	No	No	No
SmartClientViewDisconnect (Page 1612)	No	No	No	No	No
SmartClientViewConnect (Page 1612)	No	No	No	No	No
SmartClientViewLeave (Page 1613)	No	No	No	No	No
SaveDataRecord (Page 1647)	Yes	Yes	Yes	Yes	Yes
StartLogging (Page 1544)	No	No	No	No	No
StartNextLog (Page 1545)	No	No	No	No	No
StartProgram (Page 1525)	No	No	No	No	No
StatusForceGetValues (Page 1527)	No	No	No	No	No
StatusForceSetValues (Page 1527)	No	No	No	No	No
ControlSmartServer (Page 1528)	No	No	No	No	No
ControlWebServer (Page 1528)	No	No	No	No	No
StopLogging (Page 1546)	No	No	No	No	No
StopRuntime (Page 1529)	Yes	Yes	Yes	Yes	Yes
SystemDiagnosticViewUpdatePLCBuffer (Page 1614)	Yes	Yes	Yes	Yes	Yes
ActivateSystemDiagnosticsView (Page 1613)	No	No	No	No	No
SystemDiagnosticViewDetailView (Page 1614)	Yes	Yes	Yes	Yes	Yes
SystemDiagnosticsViewDiagnosticBuffer (Page 1615)	No	No	No	No	No
SystemDiagnosticViewDeviceView (Page 1615)	No	No	No	No	No
SystemDiagnosticsViewBack (Page 1616)	Yes	Yes	Yes	Yes	Yes

	KP300 Basic PN	KP400 Basic PN	KTP600 Basic PN / DP	KTP1000 Basic PN / DP	TP1500 Basic PN
LookupText (Page 1530)	No	No	No	No	No
ResetTagToHandWheel (Page 1531)	No	No	No	No	No
SetTagToHandWheel (Page 1531)	No	No	No	No	No
TraceUserChange (Page 1551)	Yes	Yes	Yes	Yes	Yes
DecreaseFocusedValue (Page 1532)	Yes	Yes	Yes	Yes	No
DecreaseTag (Page 1557)	Yes	Yes	Yes	Yes	Yes
ChangeConnection (Page 1579)	Yes	Yes	Yes	Yes	Yes
ChangeConnectionEIP (Page 1580)	No	No	No	No	No
ShowLogonDialog (Page 1551)	Yes	Yes	Yes	Yes	Yes
ShowOperatorNotes (Page 1532)	Yes	Yes	Yes	Yes	Yes
ShowAlarmWindow (Page 1621)	Yes	Yes	Yes	Yes	Yes
ShowPopupScreen (Page 1534)	No	No	No	No	No
ShowPopupScreenSizable (Page 1535)	No	No	No	No	No
ShowSlideInScreen (Page 1538)	No	No	No	No	No
ShowSoftwareVersion (Page 1533)	No	No	No	No	No
ShowSystemDiagnosticsWindow (Page 1616)	No	No	No	No	No
ShowSystemEvent (Page 1622)	No	No	No	No	No

1) For KTP600 Basic mono PN only
2) Only for editing the IP settings

System functions for Basic Panels 2nd Generation (Basic Panels)

Availability of system functions

The following table shows the availability of the system functions on the Basic Panels.

Technical data subject to change.

Overview

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic PN	KTP1200 Basic PN / DP
User-defined functions	No	No	No	No
Logoff (Page 1547)	Yes	Yes	Yes	Yes
ActivateScreen (Page 1505)	Yes	Yes	Yes	Yes
ActivateScreenByNumber (Page 1506)	Yes	Yes	Yes	Yes
ActivateCleanScreen (Page 1648)	Yes	Yes	Yes	Yes
ActivatePreviousScreen (Page 1507)	Yes	Yes	Yes	Yes

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic PN	KTP1200 Basic PN / DP
UpdateTag (Page 1508)	Yes	Yes	Yes	Yes
AdjustContrast (Page 1582)	No	No	No	No
Logon (Page 1547)	Yes	Yes	Yes	Yes
ArchiveLogFile (Page 1538)	No	No	No	No
LogTag (Page 1540)	Yes	Yes	Yes	Yes
EditAlarm (Page 1617)	Yes	Yes	Yes	Yes
ScreenObjectCursorDown (Page 1651)	Yes	Yes	Yes	Yes
ScreenObjectCursorUp (Page 1652)	Yes	Yes	Yes	Yes
ScreenObjectCursorLeft (Page 1654)	Yes	Yes	Yes	Yes
ScreenObjectCursorRight (Page 1652)	Yes	Yes	Yes	Yes
ScreenObjectPageDown (Page 1653)	Yes	Yes	Yes	Yes
ScreenObjectPageUp (Page 1653)	Yes	Yes	Yes	Yes
Encode (Page 1508)	No	No	No	No
EncodeEx (Page 1509)	No	No	No	No
Direct key (Page 1510)	No	No	No	No
DirectKeyScreenNumber (Page 1513)	No	No	No	No
PrintScreen (Page 1570)	No	No	No	No
PrintReport (Page 1571)	No	No	No	No
NotifyUserAction (Page 1513)	No	No	No	No
IncreaseFocusedValue (Page 1515)	Yes	Yes	Yes	Yes
IncreaseTag (Page 1552)	Yes	Yes	Yes	Yes
ExportDataRecords (Page 1623)	Yes	Yes	Yes	Yes
ExportDataRecordsWithChecksum (Page 1625)	No	No	No	No
ExportImportUserAdministration (Page 1548)	No	No	No	No
GoToHome (Page 1655)	Yes	Yes	Yes	Yes
GoToEnd (Page 1656)	Yes	Yes	Yes	Yes
SafelyRemoveHardware (Page 1515)	Yes	Yes	Yes	Yes
HTMLBrowserStop (Page 1586)	Yes	Yes	Yes	Yes
HTMLBrowserScrollDown (Page 1587)	Yes	Yes	Yes	Yes
HTMLBrowserRefresh (Page 1587)	Yes	Yes	Yes	Yes
HTMLBrowserScrollUp (Page 1588)	Yes	Yes	Yes	Yes
HTMLBrowserZoomIn (Page 1588)	Yes	Yes	Yes	Yes
HTMLBrowserZoomOut (Page 1589)	Yes	Yes	Yes	Yes
HTMLBrowserScrollLeft (Page 1589)	Yes	Yes	Yes	Yes
HTMLBrowserScrollRight (Page 1589)	Yes	Yes	Yes	Yes
HTMLBrowserPageDown (Page 1590)	Yes	Yes	Yes	Yes
HTMLBrowserPageUp (Page 1590)	Yes	Yes	Yes	Yes
HTMLBrowserHome (Page 1591)	Yes	Yes	Yes	Yes
HTMLBrowserForward (Page 1591)	Yes	Yes	Yes	Yes
HTMLBrowserBack (Page 1592)	Yes	Yes	Yes	Yes

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic PN	KTP1200 Basic PN / DP
ImportDataRecords (Page 1628)	Yes	Yes	Yes	Yes
ImportDataRecordsWithChecksum (Page 1629)	No	No	No	No
InvertBit (Page 1557)	Yes	Yes	Yes	Yes
InvertBitInTag (Page 1559)	Yes	Yes	Yes	Yes
InvertLinearScaling (Page 1553)	Yes	Yes	Yes	Yes
CalibrateTouchScreen (Page 1583)	Yes	Yes	Yes	Yes
CopyLog (Page 1541)	No	No	No	No
TrendViewScrollForward (Page 1592)	Yes	Yes	Yes	Yes
TrendViewScrollBack (Page 1593)	Yes	Yes	Yes	Yes
TrendViewExtend (Page 1593)	Yes	Yes	Yes	Yes
TrendViewCompress (Page 1593)	Yes	Yes	Yes	Yes
TrendViewRulerLeft (Page 1594)	Yes	Yes	Yes	Yes
TrendViewRulerRight (Page 1595)	Yes	Yes	Yes	Yes
TrendViewSetRulerMode (Page 1595)	Yes	Yes	Yes	Yes
TrendViewStartStop (Page 1596)	Yes	Yes	Yes	Yes
TrendViewBackToBeginning (Page 1596)	Yes	Yes	Yes	Yes
LoadDataRecord (Page 1631)	Yes	Yes	Yes	Yes
GetUserName (Page 1549)	Yes	Yes	Yes	Yes
GetDataRecordFromPLC (Page 1632)	Yes	Yes	Yes	Yes
GetDataRecordName (Page 1633)	No	No	No	No
GetDataRecordTagsFromPLC (Page 1635)	Yes	Yes	Yes	Yes
GetGroupNumber (Page 1550)	Yes	Yes	Yes	Yes
GetBrightness (Page 1586)	No	No	No	No
GetPassword (Page 1550)	No [1)]	No [1)]	No [1)]	No [1)]
GetPLCMode (Page 1572)	Yes	Yes	Yes	Yes
LinearScaling (Page 1554)	Yes	Yes	Yes	Yes
ClearLog (Page 1542)	Yes	Yes	Yes	Yes
ClearDataRecord (Page 1636)	Yes	Yes	Yes	Yes
ClearDataRecordMemory (Page 1637)	No	No	No	No
ClearAlarmBuffer (Page 1617)	Yes	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 1618)	Yes	Yes	Yes	Yes
AlarmViewUpdate (Page 1619)	No	No	No	No
AlarmViewLoopInAlarm (Page 1619)	Yes	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 1620)	Yes	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 1620)	Yes	Yes	Yes	Yes
OpenAllLogs (Page 1543)	Yes	Yes	Yes	Yes
OpenScreenKeyboard (Page 1650)	No	No	No	No
OpenControlPanelDialog (Page 1516)	Yes [2)]	Yes [2)]	Yes [2)]	Yes [2)]
OpenCommandPrompt (Page 1517)	No	No	No	No
OpenInternetExplorer (Page 1517)	No	No	No	No

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic PN	KTP1200 Basic PN / DP
OpenControlPanel (Page 1518)	No	No	No	No
OpenTaskManager (Page 1650)	No	No	No	No
PDFScrollDown (Page 1597)	No	No	No	No
PDFFitToWidth (Page 1598)	No	No	No	No
PDFFitToHeight (Page 1598)	No	No	No	No
PDFScrollUp (Page 1597)	No	No	No	No
PDFGoToFirstPage (Page 1599)	No	No	No	No
PDFGoToLastPage (Page 1599)	No	No	No	No
PDFGoToNextPage (Page 1600)	No	No	No	No
PDFGoToPreviousPage (Page 1601)	No	No	No	No
PDFGoToPage (Page 1600)	No	No	No	No
PDFZoomIn (Page 1601)	No	No	No	No
PDFZoomOut (Page 1601)	No	No	No	No
PDFScrollLeft (Page 1602)	No	No	No	No
PDFScrollRight (Page 1602)	No	No	No	No
PDFZoomOriginal (Page 1603)	No	No	No	No
AcknowledgeAlarm (Page 1621)	Yes	Yes	Yes	Yes
RecipeViewNewDataRecord (Page 1638)	Yes	Yes	Yes	Yes
RecipeViewGetDataRecordFromPLC (Page 1638)	Yes	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 1639)	Yes	Yes	Yes	Yes
RecipeViewMenu (Page 1639)	Yes	Yes	Yes	Yes
RecipeViewOpen (Page 1640)	Yes	Yes	Yes	Yes
RecipeViewSetDataRecordToPLC (Page 1640)	Yes	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 1641)	Yes	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 1641)	Yes	Yes	Yes	Yes
RecipeViewSynchronizeDataRecordWithTags (Page 1642)	Yes	Yes	Yes	Yes
RecipeViewRenameDataRecord (Page 1642)	Yes	Yes	Yes	Yes
RecipeViewShowOperatorNotes (Page 1643)	Yes	Yes	Yes	Yes
RecipeViewBack (Page 1643)	Yes	Yes	Yes	Yes
ResetBit (Page 1561)	Yes	Yes	Yes	Yes
ResetBitInTag (Page 1562)	Yes	Yes	Yes	Yes
PressButton (Page 1521)	Yes	Yes	Yes	Yes
ReleaseButton (Page 1522)	Yes	Yes	Yes	Yes
ShiftAndMask (Page 1564)	Yes	Yes	Yes	Yes
CloseAllLogs (Page 1543)	Yes	Yes	Yes	Yes
SetDataRecordToPLC (Page 1644)	Yes	Yes	Yes	Yes
SetDataRecordTagsToPLC (Page 1645)	Yes	Yes	Yes	Yes
PageDown (Page 1654)	Yes	Yes	Yes	Yes
PageUp (Page 1655)	Yes	Yes	Yes	Yes

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic PN	KTP1200 Basic PN / DP
SendEMail (Page 1519)	No	No	No	No
SetAcousticSignal (Page 1584)	No	No	No	No
SetDisplayMode (Page 1573)	No	No	No	No
SetDeviceMode (Page 1573)	Yes	Yes	Yes	Yes
SetBit (Page 1565)	Yes	Yes	Yes	Yes
SetBitInTag (Page 1567)	Yes	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 1569)	Yes	Yes	Yes	Yes
SetBacklightColor (Page 1585)	No	No	No	No
SetBrightness (Page 1584)	Yes	Yes	Yes	Yes
SetScreenKeyboardMode (Page 1574)	No	No	No	No
SetAlarmReportMode (Page 1571)	No	No	No	No
SetRecipeTags (Page 1645)	Yes	Yes	Yes	Yes
SetDaylightSavingTime (Page 1520)	No	No	No	No
SetLanguage (Page 1576)	Yes	Yes	Yes	Yes
SetTag (Page 1555)	Yes	Yes	Yes	Yes
SetConnectionMode (Page 1577)	Yes	Yes	Yes	Yes
SetWebAccess (Page 1578)	No	No	No	No
BackupRAMFileSystem (Page 1522)	No	No	No	No
SimulateSystemKey (Page 1523)	Yes	Yes	Yes	Yes
SimulateTag (Page 1524)	Yes	Yes	Yes	Yes
SmartClientViewRefresh (Page 1610)	No	No	No	No
SmartClientViewReadOnlyOff (Page 1611)	No	No	No	No
SmartClientViewReadOnlyOn (Page 1611)	No	No	No	No
SmartClientViewDisconnect (Page 1612)	No	No	No	No
SmartClientViewConnect (Page 1612)	No	No	No	No
SmartClientViewLeave (Page 1613)	No	No	No	No
SaveDataRecord (Page 1647)	Yes	Yes	Yes	Yes
StartLogging (Page 1544)	Yes	Yes	Yes	Yes
StartNextLog (Page 1545)	No	No	No	No
StartProgram (Page 1525)	No	No	No	No
StatusForceGetValues (Page 1527)	No	No	No	No
StatusForceSetValues (Page 1527)	No	No	No	No
ControlSmartServer (Page 1528)	No	No	No	No
ControlWebServer (Page 1528)	No	No	No	No
StopLogging (Page 1546)	Yes	Yes	Yes	Yes
StopRuntime (Page 1529)	Yes	Yes	Yes	Yes
ActivateSystemDiagnosticsView (Page 1613)	No	No	No	No
SystemDiagnosticViewUpdatePLCBuffer (Page 1614)	Yes	Yes	Yes	Yes
SystemDiagnosticsViewDetailView (Page 1614)	Yes	Yes	Yes	Yes

	KTP400 Basic PN	KTP700 Basic PN / DP	KTP900 Basic PN	KTP1200 Basic PN / DP
SystemDiagnosticsViewDiagnosticBuffer (Page 1615)	No	No	No	No
SystemDiagnosticViewDeviceView (Page 1615)	No	No	No	No
SystemDiagnosticsViewBack (Page 1616)	Yes	Yes	Yes	Yes
LookupText (Page 1530)	No	No	No	No
ResetTagToHandWheel (Page 1531)	No	No	No	No
SetTagToHandWheel (Page 1531)	No	No	No	No
TraceUserChange (Page 1551)	Yes	Yes	Yes	Yes
DecreaseFocusedValue (Page 1532)	Yes	Yes	Yes	Yes
DecreaseTag (Page 1557)	Yes	Yes	Yes	Yes
ChangeConnection (Page 1579)	Yes	Yes	Yes	Yes
ChangeConnectionEIP (Page 1580)	No	No	No	No
ShowLogonDialog (Page 1551)	Yes	Yes	Yes	Yes
ShowOperatorNotes (Page 1532)	Yes	Yes	Yes	Yes
ShowAlarmWindow (Page 1621)	Yes	Yes	Yes	Yes
ShowPopupScreen (Page 1534)	No	No	No	No
ShowPopupScreenSizable (Page 1535)	No	No	No	No
ShowSlideInScreen (Page 1538)	No	No	No	No
ShowSoftwareVersion (Page 1533)	No	No	No	No
ShowSystemDiagnosticsWindow (Page 1616)	No	No	No	No
ShowSystemEvent (Page 1622)	No	No	No	No

[1] Not available for devices as of device version V14.0

[2] Only for editing the IP settings

System functions for Comfort Panels (Panels, Comfort Panels)

Availability of system functions

The following table shows the availability of the system functions on the Comfort Panels.

Technical data subject to change.

Overview

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
User-defined functions	Yes	Yes	Yes
Logoff (Page 1547)	Yes	Yes	Yes
ActivateScreen (Page 1505)	Yes	Yes	Yes
ActivateScreenByNumber (Page 1506)	Yes	Yes	Yes
ActivatePLCCodeViewer (Page 1603)	No	Yes	Yes
ActivateCleanScreen (Page 1648)	Yes [1)]	Yes	No
ActivatePreviousScreen (Page 1507)	Yes	Yes	Yes
UpdateTag (Page 1508)	Yes	Yes	Yes
AdjustContrast (Page 1582)	No	No	No
Logon (Page 1547)	Yes	Yes	Yes
ArchiveLogFile (Page 1538)	Yes	Yes	Yes
LogTag (Page 1540)	Yes	Yes	Yes
EditAlarm (Page 1617)	Yes	No	Yes
ScreenObjectCursorDown (Page 1651)	Yes	Yes	Yes
ScreenObjectCursorUp (Page 1652)	Yes	Yes	Yes
ScreenObjectCursorLeft (Page 1654)	No	Yes	Yes
ScreenObjectCursorRight (Page 1652)	No	Yes	Yes
ScreenObjectPageDown (Page 1653)	Yes	Yes	Yes
ScreenObjectPageUp (Page 1653)	Yes	Yes	Yes
Encode (Page 1508)	Yes	Yes	Yes
EncodeEx (Page 1509)	Yes	Yes	Yes
DirectKey (Page 1510)	Yes [1)]	Yes	No
DirectKeyScreenNumber (Page 1513)	Yes [1)]	Yes	No
PrintScreen (Page 1570)	Yes	Yes	Yes
PrintReport (Page 1571)	Yes	Yes	Yes
NotifyUserAction (Page 1513)	Yes	Yes	Yes
IncreaseFocusedValue (Page 1515)	Yes	No	Yes
IncreaseTag (Page 1552)	Yes	Yes	Yes
ExportDataRecords (Page 1623)	Yes	Yes	Yes
ExportDataRecordsWithChecksum (Page 1625)	Yes	Yes	Yes
ExportImportUserAdministration (Page 1548)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
GoToHome (Page 1655)	Yes	No	Yes
GoToEnd (Page 1656)	Yes	No	Yes
SafelyRemoveHardware (Page 1515)	Yes	Yes	Yes
HTMLBrowserStop (Page 1586)	No	Yes	Yes
HTMLBrowserScrollDown (Page 1587)	No	Yes	Yes
HTMLBrowserRefresh (Page 1587)	No	Yes	Yes
HTMLBrowserScrollUp (Page 1588)	No	Yes	Yes
HTMLBrowserForward (Page 1591)	No	Yes	Yes
HTMLBrowserBack (Page 1592)	No	Yes	Yes
HTMLBrowserZoomIn (Page 1588)	No	Yes	Yes
HTMLBrowserZoomOut (Page 1589)	No	Yes	Yes
HTMLBrowserScrollLeft (Page 1589)	No	Yes	Yes
HTMLBrowserScrollRight (Page 1589)	No	Yes	Yes
HTMLBrowserPageDown (Page 1590)	No	Yes	Yes
HTMLBrowserPageUp (Page 1590)	No	Yes	Yes
HTMLBrowserHome (Page 1591)	No	Yes	Yes
AUTOHOTSPOT	No	Yes	Yes
ImportDataRecords (Page 1628)	Yes	Yes	Yes
ImportDataRecordsWithChecksum (Page 1629)	Yes	Yes	Yes
InvertBit (Page 1557)	Yes	Yes	Yes
InvertBitInTag (Page 1559)	Yes	Yes	Yes
InvertLinearScaling (Page 1553)	Yes	Yes	Yes
CalibrateTouchScreen (Page 1583)	Yes [1)]	Yes	No
CopyLog (Page 1541)	Yes	Yes	Yes
TrendViewScrollForward (Page 1592)	Yes	Yes	Yes
TrendViewScrollBack (Page 1593)	Yes	Yes	Yes
TrendViewExtend (Page 1593)	Yes	Yes	Yes
TrendViewCompress (Page 1593)	Yes	Yes	Yes
TrendViewRulerLeft (Page 1594)	Yes	Yes	Yes
TrendViewRulerRight (Page 1595)	Yes	Yes	Yes
TrendViewSetRulerMode (Page 1595)	Yes	Yes	Yes
TrendViewStartStop (Page 1596)	Yes	Yes	Yes
TrendViewBackToBeginning (Page 1596)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
LoadDataRecord (Page 1631)	Yes	Yes	Yes
GetUserName (Page 1549)	Yes	Yes	Yes
GetDataRecordFromPLC (Page 1632)	Yes	Yes	Yes
GetDataRecordName (Page 1633)	Yes	Yes	Yes
GetDataRecordTagsFromPLC (Page 1635)	Yes	Yes	Yes
GetGroupNumber (Page 1550)	Yes	Yes	Yes
GetBrightness (Page 1586)	Yes	Yes	Yes
GetPassword (Page 1550)	No [2)]	No [2)]	No [2)]
GetPLCMode (Page 1572)	Yes	Yes	Yes
LinearScaling (Page 1554)	Yes	Yes	Yes
ClearLog (Page 1542)	Yes	Yes	Yes
ClearDataRecord (Page 1636)	Yes	Yes	Yes
ClearDataRecordMemory (Page 1637)	Yes	Yes	Yes
ClearAlarmBuffer (Page 1617)	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 1618)	Yes	Yes	Yes
AlarmViewUpdate (Page 1619)	Yes	Yes	Yes
AlarmViewLoopInAlarm (Page 1619)	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 1620)	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 1620)	Yes	Yes	Yes
OpenAllLogs (Page 1543)	Yes	Yes	Yes
OpenScreenKeyboard (Page 1650)	Yes	Yes	Yes
OpenFileBrowser (Page 1649)	Yes	Yes	Yes
OpenControlPanelDialog (Page 1516)	Yes	Yes	Yes
OpenCommandPrompt (Page 1517)	Yes	Yes	Yes
OpenInternetExplorer (Page 1517)	Yes	Yes	Yes
OpenControlPanel (Page 1518)	Yes	Yes	Yes
OpenTaskManager (Page 1650)	Yes	Yes	Yes
AcknowledgeAlarm (Page 1621)	Yes	No	Yes
PDFScrollDown (Page 1597)	Yes	Yes	Yes
PDFScrollUp (Page 1597)	Yes	Yes	Yes
PDFFitToWidth (Page 1598)	Yes	Yes	Yes
PDFFitToHeight (Page 1598)	Yes	Yes	Yes
PDFGotoFirstPage (Page 1599)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
PDFGotoLastPage (Page 1599)	Yes	Yes	Yes
PDFGotoNextPage (Page 1600)	Yes	Yes	Yes
PDFGotoPage (Page 1600)	Yes	Yes	Yes
PDFGotoPreviousPage (Page 1601)	Yes	Yes	Yes
PDFZoomIn (Page 1601)	Yes	Yes	Yes
PDFZoomOut (Page 1601)	Yes	Yes	Yes
PDFScrollLeft (Page 1602)	Yes	Yes	Yes
PDFScrollRight (Page 1602)	Yes	Yes	Yes
PDFZoomOriginal (Page 1603)	Yes	Yes	Yes
PLCCodeViewDetails (Page 1606)	No	Yes	Yes
PLCCodeViewNextNetwork (Page 1606)	No	Yes	Yes
PLCCodeViewStepMode (Page 1607)	No	Yes	Yes
PLCCodeViewSymbols (Page 1607)	No	Yes	Yes
PLCCodeViewTransitionInterlock (Page 1608)	No	Yes	Yes
PLCCodeViewZoomIn (Page 1609)	No	Yes	Yes
PLCCodeViewZoomOut (Page 1609)	No	Yes	Yes
PLCCodeViewPreviousNetwork (Page 1610)	No	Yes	Yes
AUTOHOTSPOT	No	Yes	Yes
RecipeViewNewDataRecord (Page 1638)	Yes	Yes	Yes
RecipeViewGetDataRecordFromPLC (Page 1638)	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 1639)	Yes	Yes	Yes
RecipeViewMenu (Page 1639)	Yes	Yes	Yes
RecipeViewOpen (Page 1640)	Yes	Yes	Yes
RecipeViewSetDataRecordToPLC (Page 1640)	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 1641)	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 1641)	Yes	Yes	Yes
RecipeViewSynchronizeDataRecord-WithTags (Page 1642)	Yes	Yes	Yes
RecipeViewRenameDataRecord (Page 1642)	Yes	Yes	Yes
RecipeViewShowOperatorNotes (Page 1643)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
RecipeViewBack (Page 1643)	Yes	Yes	Yes
ResetBit (Page 1561)	Yes	Yes	Yes
ResetBitInTag (Page 1562)	Yes	Yes	Yes
PressButton (Page 1521)	Yes	Yes	Yes
ReleaseButton (Page 1522)	Yes	Yes	Yes
ShiftAndMask (Page 1564)	Yes	Yes	Yes
CloseAllLogs (Page 1543)	Yes	Yes	Yes
SetDataRecordToPLC (Page 1644)	Yes	Yes	Yes
SetDataRecordTagsToPLC (Page 1645)	Yes	Yes	Yes
PageDown (Page 1654)	Yes	No	Yes
PageUp (Page 1655)	Yes	No	Yes
SendEMail (Page 1519)	Yes	Yes	Yes
SetAcousticSignal (Page 1584)	Yes [1)]	Yes	No
SetDisplayMode (Page 1573)	No	No	No
SetDeviceMode (Page 1573)	Yes	Yes	Yes
SetBit (Page 1565)	Yes	Yes	Yes
SetBitInTag (Page 1567)	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 1569)	Yes	Yes	Yes
SetBacklightColor (Page 1585)	No	No	No
SetBrightness (Page 1584)	Yes	Yes	Yes
SetScreenKeyboardMode (Page 1574)	Yes	Yes	Yes
SetPLCMode (Page 1576)	Yes	Yes	Yes
SetAlarmReportMode (Page 1571)	Yes	Yes	Yes
SetRecipeTags (Page 1645)	Yes	Yes	Yes
SetDaylightSavingTime (Page 1520)	Yes	Yes	Yes
SetLanguage (Page 1576)	Yes	Yes	Yes
SetTag (Page 1555)	Yes	Yes	Yes
SetConnectionMode (Page 1577)	Yes	Yes	Yes
SetWebAccess (Page 1578)	Yes	Yes	Yes
BackupRAMFileSystem (Page 1522)	Yes	Yes	Yes
SimulateSystemKey (Page 1523)	Yes	No	Yes
SimulateTag (Page 1524)	Yes	Yes	Yes
SmartClientViewRefresh (Page 1610)	Yes	Yes	Yes
SmartClientViewReadOnlyOff (Page 1611)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
SmartClientViewReadOnlyOn (Page 1611)	Yes	Yes	Yes
SmartClientViewDisconnect (Page 1612)	Yes	Yes	Yes
SmartClientViewConnect (Page 1612)	Yes	Yes	Yes
SmartClientViewLeave (Page 1613)	Yes	Yes	Yes
SaveDataRecord (Page 1647)	Yes	Yes	Yes
StartLogging (Page 1544)	Yes	Yes	Yes
StartNextLog (Page 1545)	Yes	Yes	Yes
StartProgram (Page 1525)	Yes	Yes	Yes
StatusForceGetValues (Page 1527)	Yes	Yes	Yes
StatusForceSetValues (Page 1527)	Yes	Yes	Yes
ControlSmartServer (Page 1528)	Yes	Yes	Yes
ControlWebServer (Page 1528)	Yes	Yes	Yes
StopLogging (Page 1546)	Yes	Yes	Yes
StopRuntime (Page 1529)	Yes	Yes	Yes
ActivateSystemDiagnosticsView (Page 1613)	Yes	Yes	Yes
SystemDiagnosticViewUpdatePlc-Buffer (Page 1614)	Yes	Yes	Yes
SystemDiagnosticViewDetailView (Page 1614)	Yes	Yes	Yes
SystemDiagnosticViewDiagnostic-Buffer (Page 1615)	Yes	Yes	Yes
SystemDiagnosticViewDeviceView (Page 1615)	Yes	Yes	Yes
SystemDiagnosticViewBack (Page 1616)	Yes	Yes	Yes
LookupText (Page 1530)	Yes	Yes	Yes
ResetTagToHandWheel (Page 1531)	No	No	No
SetTagToHandWheel (Page 1531)	No	No	No
TraceUserChange (Page 1551)	Yes	Yes	Yes
DecreaseFocusedValue (Page 1532)	Yes	No	Yes
DecreaseTag (Page 1557)	Yes	Yes	Yes
ChangeConnection (Page 1579)	Yes	Yes	Yes
ChangeConnectionEIP (Page 1580)	Yes	Yes	Yes
ShowLogonDialog (Page 1551)	Yes	Yes	Yes
ShowOperatorNotes (Page 1532)	Yes	Yes	Yes
ShowAlarmWindow (Page 1621)	Yes	Yes	Yes

	KTP400 KP400	TP700 TP900 TP1200 TP1500 TP1900 TP2200	KP700 KP900 KP1200 KP1500
ShowPopupScreen (Page 1534)	Yes	Yes	Yes
ShowPopupScreenSizable (Page 1535)	Yes	Yes	Yes
ShowSlideInScreen (Page 1538)	Yes	Yes	Yes
ShowSoftwareVersion (Page 1533)	Yes	Yes	Yes
ShowSystemDiagnosticsWindow (Page 1616)	Yes	Yes	Yes
ShowSystemAlarm (Page 1622)	Yes	Yes	Yes

[1] Only on KTP 400 Comfort

[2] Not available for devices as of device version V14.0 and higher

System functions for Mobile Panels (Panels, Comfort Panels)

Availability of system functions

The following table shows the availability of the system functions on the mobile panels.

Technical data subject to change.

Overview

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
User-defined functions	No	No	Yes	Yes	Yes	Yes
Logoff (Page 1547)	Yes	Yes	Yes	Yes	Yes	Yes
ActivateScreen (Page 1505)	Yes	Yes	Yes	Yes	Yes	Yes
ActivateScreenByNumber (Page 1506)	Yes	Yes	Yes	Yes	Yes	Yes
ActivatePLCCodeViewer (Page 1603)	No	No	No	No	Yes	Yes
ActivateCleanScreen (Page 1648)	No	No	No	No	No	No
ActivatePreviousScreen (Page 1507)	Yes	Yes	Yes	Yes	Yes	Yes
UpdateTag (Page 1508)	Yes	Yes	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
AdjustContrast (Page 1582)	Yes	Yes	No	No	No	No
Logon (Page 1547)	Yes	Yes	Yes	Yes	Yes	Yes
ArchiveLogFile (Page 1538)	No	No	Yes	Yes	Yes	Yes
LogTag (Page 1540)	No	No	Yes	Yes	Yes	Yes
EditAlarm (Page 1617)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorDown (Page 1651)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorUp (Page 1652)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageDown (Page 1653)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageUp (Page 1653)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorLeft (Page 1654)	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorRight (Page 1652)	Yes	Yes	Yes	Yes	Yes	Yes
Encode (Page 1508)	Yes	Yes	Yes	Yes	Yes	Yes
EncodeEx (Page 1509)	Yes	Yes	Yes	Yes	Yes	Yes
DirectKey (Page 1510)	Yes	Yes	Yes	Yes	Yes	Yes
DirectKeyScreenNumber (Page 1513)	Yes	Yes	Yes	Yes	Yes	Yes
PrintScreen (Page 1570)	No	Yes	Yes	Yes	Yes	Yes
PrintReport (Page 1571)	No	Yes	Yes	Yes	Yes	Yes
NotifyUserAction (Page 1513)	No	No	Yes	Yes	Yes	Yes
IncreaseFocusedValue (Page 1515)	Yes	Yes	Yes	Yes	Yes	Yes
IncreaseTag (Page 1552)	Yes	Yes	Yes	Yes	Yes	Yes
ExportDataRecords (Page 1623)	Yes	Yes	Yes	Yes	Yes	Yes
ExportDataRecordsWithChecksum (Page 1625)	No	No	Yes	Yes	Yes	Yes
ExportImportUserAdministration (Page 1548)	Yes	Yes	Yes	Yes	Yes	Yes
GoToHome (Page 1655)	Yes	Yes	Yes	Yes	Yes	Yes
GoToEnd (Page 1656)	Yes	Yes	Yes	Yes	Yes	Yes
SafelyRemoveHardware (Page 1515)	No	No	Yes	Yes	Yes	Yes
HTMLBrowserStop (Page 1586)	No	No	No	Yes	Yes	Yes
HTMLBrowserScrollDown (Page 1587)	No	No	No	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
HTMLBrowserRefresh (Page 1587)	No	No	No	Yes	Yes	Yes
HTMLBrowserScrollUp (Page 1588)	No	No	No	Yes	Yes	Yes
HTMLBrowserZoomIn (Page 1588)	No	No	No	Yes	Yes	Yes
HTMLBrowserZoomOut (Page 1589)	No	No	No	Yes	Yes	Yes
HTMLBrowserScrollLeft (Page 1589)	No	No	No	Yes	Yes	Yes
HTMLBrowserScrollRight (Page 1589)	No	No	No	Yes	Yes	Yes
HTMLBrowserPageDown (Page 1590)	No	No	No	Yes	Yes	Yes
HTMLBrowserPageUp (Page 1590)	No	No	No	Yes	Yes	Yes
HTMLBrowserHome (Page 1591)	No	No	No	Yes	Yes	Yes
HTMLBrowserForward (Page 1591)	No	No	No	Yes	Yes	Yes
HTMLBrowserBack (Page 1592)	No	No	No	Yes	Yes	Yes
ImportDataRecords (Page 1628)	Yes	Yes	Yes	Yes	Yes	Yes
ImportDataRecordsWithChecksum (Page 1629)	No	No	Yes	Yes	Yes	Yes
InvertBit (Page 1557)	Yes	Yes	Yes	Yes	Yes	Yes
InvertBitInTag (Page 1559)	Yes	Yes	Yes	Yes	Yes	Yes
InvertLinearScaling (Page 1553)	Yes	Yes	Yes	Yes	Yes	Yes
CalibrateTouchScreen (Page 1583)	Yes	Yes	Yes [1)]	Yes	Yes	Yes
CopyLog (Page 1541)	No	No	Yes	Yes	Yes	Yes
TrendViewScrollForward (Page 1592)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewScrollBack (Page 1593)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewExtend (Page 1593)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewCompress (Page 1593)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewRulerLeft (Page 1594)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewRulerRight (Page 1595)	Yes	Yes	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
TrendViewSetRulerMode (Page 1595)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewStartStop (Page 1596)	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewBackToBeginning (Page 1596)	Yes	Yes	Yes	Yes	Yes	Yes
LoadDataRecord (Page 1631)	Yes	Yes	Yes	Yes	Yes	Yes
GetUserName (Page 1549)	Yes	Yes	Yes	Yes	Yes	Yes
GetDataRecordFromPLC (Page 1632)	Yes	Yes	Yes	Yes	Yes	Yes
GetDataRecordName (Page 1633)	Yes	Yes	Yes	Yes	Yes	Yes
GetDataRecordTagsFromPLC (Page 1635)	Yes	Yes	Yes	Yes	Yes	Yes
GetGroupNumber (Page 1550)	Yes	Yes	Yes	Yes	Yes	Yes
GetBrightness (Page 1586)	No	No	No	Yes	Yes	Yes
GetPassword (Page 1550)	Yes	Yes	Yes	No [2)]	No [2)]	No [2)]
ClearLog (Page 1542)	No	No	Yes	Yes	Yes	Yes
ClearDataRecord (Page 1636)	Yes	Yes	Yes	Yes	Yes	Yes
ClearDataRecordMemory (Page 1637)	Yes	Yes	Yes	Yes	Yes	Yes
ClearAlarmBuffer (Page 1617)	Yes	Yes	Yes	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 1618)	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewUpdate (Page 1619)	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewLoopInAlarm (Page 1619)	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 1620)	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 1620)	Yes	Yes	Yes	Yes	Yes	Yes
OpenAllLogs (Page 1543)	No	No	Yes	Yes	Yes	Yes
OpenFileBrowser (Page 1649)	No	No	No	Yes	Yes	Yes
OpenScreenKeyboard (Page 1650)	No	No	Yes	Yes	Yes	Yes
OpenCommandPrompt (Page 1517)	No	No	Yes	Yes	Yes	Yes
OpenControlPanelDialog (Page 1516)	No	No	No	Yes	Yes	Yes
OpenInternetExplorer (Page 1517)	No	Yes	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
OpenTaskManager (Page 1650)	No	No	Yes	Yes	Yes	Yes
OpenControlPanel (Page 1518)	No	No	Yes [1]	No	Yes/No [3]	Yes/No [3]
AcknowledgeAlarm (Page 1621)	Yes	Yes	Yes	Yes	Yes	Yes
PDFScrollDown (Page 1597)	No	No	No	Yes	Yes	Yes
PDFScrollUp (Page 1597)	No	No	No	Yes	Yes	Yes
PDFFitToWidth (Page 1598)	No	No	No	Yes	Yes	Yes
PDFFitToHeight (Page 1598)	No	No	No	Yes	Yes	Yes
PDFGotoFirstPage (Page 1599)	No	No	No	Yes	Yes	Yes
PDFGotoLastPage (Page 1599)	No	No	No	Yes	Yes	Yes
PDFGotoNextPage (Page 1600)	No	No	No	Yes	Yes	Yes
PDFGotoPage (Page 1600)	No	No	No	Yes	Yes	Yes
PDFGotoPreviousPage (Page 1601)	No	No	No	Yes	Yes	Yes
PDFZoomIn (Page 1601)	No	No	No	Yes	Yes	Yes
PDFZoomOut (Page 1601)	No	No	No	Yes	Yes	Yes
PDFScrollLeft (Page 1602)	No	No	No	Yes	Yes	Yes
PDFScrollRight (Page 1602)	No	No	No	Yes	Yes	Yes
PDFZoomOriginal (Page 1603)	No	No	No	Yes	Yes	Yes
PLCCodeViewDetails (Page 1606)	Yes	Yes	Yes	No	Yes	Yes
PLCCodeViewNextNetwork (Page 1606)	Yes	Yes	Yes	No	Yes	Yes
PLCCodeViewStepMode (Page 1607)	Yes	Yes	Yes	No	Yes	Yes
PLCCodeViewSymbols (Page 1607)	Yes	Yes	Yes	No	Yes	Yes
PLCCodeViewTransitionInterlock (Page 1608)	Yes	Yes	Yes	No	Yes	Yes
PLCCodeViewZoomIn (Page 1609)	Yes	Yes	Yes	No	Yes	Yes
PLCCodeViewZoomOut (Page 1609)	Yes	Yes	Yes	No	Yes	Yes
PLCCodeViewPreviousNetwork (Page 1610)	Yes	Yes	Yes	No	Yes	Yes
RecipeViewNewDataRecord (Page 1638)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewGetDataRecordFromPLC (Page 1638)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 1639)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewMenu (Page 1639)	Yes	Yes	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
RecipeViewOpen (Page 1640)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSetDataRecord-ToPLC (Page 1640)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 1641)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 1641)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSynchronizeDataRecordWithTags (Page 1642)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewRenameDataRecord (Page 1642)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewShowOperator-Notes (Page 1643)	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewBack (Page 1643)	Yes	Yes	Yes	Yes	Yes	Yes
ResetBit (Page 1561)	Yes	Yes	Yes	Yes	Yes	Yes
ResetBitInTag (Page 1562)	Yes	Yes	Yes	Yes	Yes	Yes
PressButton (Page 1521)	Yes	Yes	Yes	Yes	Yes	Yes
ReleaseButton (Page 1522)	Yes	Yes	Yes	Yes	Yes	Yes
ShiftAndMask (Page 1564)	Yes	Yes	Yes	Yes	Yes	Yes
CloseAllLogs (Page 1543)	No	No	Yes	Yes	Yes	Yes
SetDataRecordToPLC (Page 1644)	Yes	Yes	Yes	Yes	Yes	Yes
SetDataRecordTagsToPLC (Page 1645)	Yes	Yes	Yes	Yes	Yes	Yes
PageDown (Page 1654)	Yes	Yes	Yes	Yes	Yes	Yes
PageUp (Page 1655)	Yes	Yes	Yes	Yes	Yes	Yes
SendEMail (Page 1519)	No	Yes	Yes	Yes	Yes	Yes
SetAcousticSignal (Page 1584)	No	No	No	Yes	Yes	Yes
SetDisplayMode (Page 1573)	No	No	No	No	No	No
SetDeviceMode (Page 1573)	Yes	Yes	Yes	Yes	Yes	Yes
SetBit (Page 1565)	Yes	Yes	Yes	Yes	Yes	Yes
SetBitInTag (Page 1567)	Yes	Yes	Yes	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 1569)	Yes	Yes	Yes	Yes	Yes	Yes
SetBacklightColor (Page 1585)	No	No	No	No	No	No
SetBrightness (Page 1584)	No	No	No	Yes	Yes	Yes
SetScreenKeyboardMode (Page 1574)	No	No	No	Yes	Yes	Yes
SetAlarmReportMode (Page 1571)	No	No	Yes	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
SetRecipeTags (Page 1645)	Yes	Yes	Yes	Yes	Yes	Yes
SetDaylightSavingTime (Page 1520)	No	No	Yes	Yes	Yes	Yes
SetLanguage (Page 1576)	Yes	Yes	Yes	Yes	Yes	Yes
SetTag (Page 1555)	Yes	Yes	Yes	Yes	Yes	Yes
SetConnectionMode (Page 1577)	Yes	Yes	Yes	Yes	Yes	Yes
SetWebAccess (Page 1578)	No	Yes	Yes	Yes	Yes	Yes
BackupRAMFileSystem (Page 1522)	No	No	Yes	Yes	Yes	Yes
SimulateSystemKey (Page 1523)	Yes	Yes	Yes	Yes	Yes	Yes
SimulateTag (Page 1524)	Yes	Yes	Yes	Yes	Yes	Yes
SmartClientViewRefresh (Page 1610)	No	Yes	Yes	Yes	Yes	Yes
SmartClientViewReadOnlyOff (Page 1611)	No	Yes	Yes	Yes	Yes	Yes
SmartClientViewReadOnlyOn (Page 1611)	No	Yes	Yes	Yes	Yes	Yes
SmartClientViewDisconnect (Page 1612)	No	Yes	Yes	Yes	Yes	Yes
SmartClientViewConnect (Page 1612)	No	Yes	Yes	Yes	Yes	Yes
SmartClientViewLeave (Page 1613)	No	Yes	Yes	Yes	Yes	Yes
SaveDataRecord (Page 1647)	Yes	Yes	Yes	Yes	Yes	Yes
StartLogging (Page 1544)	No	No	Yes	Yes	Yes	Yes
StartNextLog (Page 1545)	No	No	Yes	Yes	Yes	Yes
StartProgram (Page 1525)	No	No	Yes	Yes	Yes	Yes
StatusForceGetValues (Page 1527)	Yes	Yes	Yes	Yes	Yes	Yes
StatusForceSetValues (Page 1527)	Yes	Yes	Yes	Yes	Yes	Yes
ControlSmartServer (Page 1528)	No	Yes	Yes	Yes	Yes	Yes
ControlWebServer (Page 1528)	No	Yes	Yes	Yes	Yes	Yes
StopLogging (Page 1546)	No	No	Yes	Yes	Yes	Yes
StopRuntime (Page 1529)	Yes	Yes	Yes	Yes	Yes	Yes
LookupText (Page 1530)	Yes	Yes	Yes	Yes	Yes	Yes
ActivateSystemDiagnosticsView (Page 1613)	No	No	No	Yes	Yes	Yes

	Mobile Panel 177 DP	Mobile Panel 177 PN	Mobile Panel 277 Mobile Panel 277 IWLAN Mobile Panel 277 F IWLAN	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
ResetTagToHandWheel (Page 1531)	Yes	Yes	Yes	No	No	No
SetTagToHandWheel (Page 1531)	Yes	Yes	Yes	No	No	No
TraceUserChange (Page 1551)	Yes	Yes	Yes	No	No	No
DecreaseFocusedValue (Page 1532)	Yes	Yes	Yes	Yes	Yes	Yes
DecreaseTag (Page 1557)	Yes	Yes	Yes	Yes	Yes	Yes
ChangeConnection (Page 1579)	Yes	Yes	Yes	Yes	Yes	Yes
ChangeConnectionEIP (Page 1580)	Yes	Yes	Yes	Yes	Yes	Yes
ShowLogonDialog (Page 1551)	Yes	Yes	Yes	Yes	Yes	Yes
ShowOperatorNotes (Page 1532)	Yes	Yes	Yes	Yes	Yes	Yes
ShowAlarmWindow (Page 1621)	Yes	Yes	Yes	Yes	Yes	Yes
ShowPopupScreen (Page 1534)	No	No	No	Yes	Yes	Yes
ShowPopupScreenSizable	No	No	No	Yes	Yes	Yes
ShowSlideInScreen (Page 1538)	No	No	No	Yes	Yes	Yes
ShowSoftwareVersion (Page 1533)	No	No	Yes	Yes	Yes	Yes
ShowSystemDiagnosticsWindow (Page 1616)	No	No	No	Yes	Yes	Yes
ShowSystemAlarm (Page 1622)	Yes	Yes	Yes	Yes	Yes	Yes

1) Only for Mobile Panel 277, Mobile Panel 277 IWLAN

2) Not available for devices as of device version V14.0 and higher

3) Not for F Panels

See also

LinearScaling (Page 1554)

TerminatePROFIsafe (Page 1519)

System functions for Runtime Advanced (RT Advanced)

Availability of system functions

The following table shows the availability of the system functions for WinCC RT Advanced.

Technical data subject to change.

Overview

	WinCC RT Advanced
User-defined functions	Yes
Logoff (Page 1547)	Yes
ActivateScreen (Page 1505)	Yes
ActivateScreenByNumber (Page 1506)	Yes
ActivatePLCCodeViewer (Page 1603)	Yes
ActivateCleanScreen (Page 1648)	Yes
ActivatePreviousScreen (Page 1507)	Yes
UpdateTag (Page 1508)	Yes
AdjustContrast (Page 1582)	No
Logon (Page 1547)	Yes
ArchiveLogFile (Page 1538)	Yes
LogTag (Page 1540)	Yes
EditAlarm (Page 1617)	Yes
ScreenObjectCursorDown (Page 1651)	Yes
ScreenObjectCursorUp (Page 1652)	Yes
ScreenObjectCursorLeft (Page 1654)	Yes
ScreenObjectCursorRight (Page 1652)	Yes
ScreenObjectPageDown (Page 1653)	Yes
ScreenObjectPageUp (Page 1653)	Yes
Encode (Page 1508)	Yes
EncodeEx (Page 1509)	Yes
DirectKey (Page 1510)	No
DirectKeyScreenNumber (Page 1513)	No
PrintScreen (Page 1570)	Yes
PrintReport (Page 1571)	Yes
NotifyUserAction (Page 1513)	Yes
IncreaseFocusedValue (Page 1515)	Yes
IncreaseTag (Page 1552)	Yes
ExportDataRecords (Page 1623)	Yes
ExportDataRecordsWithChecksum (Page 1625)	No
ExportImportUserAdministration (Page 1548)	Yes
GoToHome (Page 1655)	Yes
GoToEnd (Page 1656)	Yes
SafelyRemoveHardware (Page 1515)	No
HTMLBrowserStop (Page 1586)	Yes
HTMLBrowserScrollDown (Page 1587)	Yes
HTMLBrowserRefresh (Page 1587)	Yes
HTMLBrowserScrollUp (Page 1588)	Yes
HTMLBrowserZoomIn (Page 1588)	Yes
HTMLBrowserZoomOut (Page 1589)	Yes

	WinCC RT Advanced
HTMLBrowserScrollLeft (Page 1589)	Yes
HTMLBrowserScrollRight (Page 1589)	Yes
HTMLBrowserPageDown (Page 1590)	Yes
HTMLBrowserPageUp (Page 1590)	Yes
HTMLBrowserHome (Page 1591)	Yes
HTMLBrowserForward (Page 1591)	Yes
HTMLBrowserBack (Page 1592)	Yes
ImportDataRecords (Page 1628)	Yes
ImportDataRecordsWithChecksum (Page 1629)	Yes
InvertBit (Page 1557)	Yes
InvertBitInTag (Page 1559)	Yes
InvertLinearScaling (Page 1553)	Yes
CalibrateTouchScreen (Page 1583)	No
CopyLog (Page 1541)	Yes
TrendViewScrollForward (Page 1592)	Yes
TrendViewScrollBack (Page 1593)	Yes
TrendViewExtend (Page 1593)	Yes
TrendViewCompress (Page 1593)	Yes
TrendViewRulerLeft (Page 1594)	Yes
TrendViewRulerRight (Page 1595)	Yes
TrendViewSetRulerMode (Page 1595)	Yes
TrendViewStartStop (Page 1596)	Yes
TrendViewBackToBeginning (Page 1596)	Yes
LoadDataRecord (Page 1631)	Yes
GetUserName (Page 1549)	Yes
GetDataRecordFromPLC (Page 1632)	Yes
GetDataRecordName (Page 1633)	Yes
GetDataRecordTagsFromPLC (Page 1635)	Yes
GetGroupNumber (Page 1550)	Yes
GetBrightness (Page 1586)	No
GetPassword (Page 1550)	No [1)]
LinearScaling (Page 1554)	Yes
ClearLog (Page 1542)	Yes
ClearDataRecord (Page 1636)	Yes
ClearDataRecordMemory (Page 1637)	No
ClearAlarmBuffer (Page 1617)	Yes
ClearAlarmBufferProTool (Page 1618)	Yes
AlarmViewUpdate (Page 1619)	Yes
AlarmViewLoopInAlarm (Page 1619)	Yes
AlarmViewAcknowledgeAlarm (Page 1620)	Yes
AlarmViewShowOperatorNotes (Page 1620)	Yes

	WinCC RT Advanced
OpenAllLogs (Page 1543)	Yes
OpenScreenKeyboard (Page 1650)	Yes
OpenControlPanelDialog (Page 1516)	No
OpenCommandPrompt (Page 1517)	No
OpenInternetExplorer (Page 1517)	No
OpenControlPanel (Page 1518)	No
OpenTaskManager (Page 1650)	Yes
AcknowledgeAlarm (Page 1621)	Yes
PDFScrollDown (Page 1597)	Yes
PDFScrollUp (Page 1597)	Yes
PDFFitToWidth (Page 1598)	Yes
PDFFitToHeight (Page 1598)	Yes
PDFGotoFirstPage (Page 1599)	Yes
PDFGotoLastPage (Page 1599)	Yes
PDFGotoNextPage (Page 1600)	Yes
PDFGotoPage (Page 1600)	Yes
PDFGotoPreviousPage (Page 1601)	Yes
PDFZoomIn (Page 1601)	Yes
PDFZoomOut (Page 1601)	Yes
PDFScrollLeft (Page 1602)	Yes
PDFScrollRight (Page 1602)	Yes
PDFZoomOriginal (Page 1603)	Yes
PLCCodeViewDetails (Page 1606)	Yes
PLCCodeViewNextNetwork (Page 1606)	Yes
PLCCodeViewStepMode (Page 1607)	Yes
PLCCodeViewSymbols (Page 1607)	Yes
PLCCodeViewTransitionInterlock (Page 1608)	Yes
PLCCodeViewZoomIn (Page 1609)	Yes
PLCCodeViewZoomOut (Page 1609)	Yes
PLCCodeViewPreviousNetwork (Page 1610)	Yes
RecipeViewNewDataRecord (Page 1638)	Yes
RecipeViewGetDataRecordFromPLC (Page 1638)	Yes
RecipeViewClearDataRecord (Page 1639)	Yes
AUTOHOTSPOT	Yes
RecipeViewMenu (Page 1639)	Yes
RecipeViewOpen (Page 1640)	Yes
RecipeViewSetDataRecordToPLC (Page 1640)	Yes
RecipeViewSaveDataRecord (Page 1641)	Yes
RecipeViewSaveAsDataRecord (Page 1641)	Yes
RecipeViewSynchronizeDataRecordWithTags (Page 1642)	Yes
RecipeViewRenameDataRecord (Page 1642)	Yes

	WinCC RT Advanced
RecipeViewShowOperatorNotes (Page 1643)	Yes
RecipeViewBack (Page 1643)	Yes
ResetBit (Page 1561)	Yes
ResetBitInTag (Page 1562)	Yes
PressButton (Page 1521)	Yes
ReleaseButton (Page 1522)	Yes
ShiftAndMask (Page 1564)	Yes
CloseAllLogs (Page 1543)	Yes
SetDataRecordToPLC (Page 1644)	Yes
SetDataRecordTagsToPLC (Page 1645)	Yes
PageDown (Page 1654)	Yes
PageUp (Page 1655)	Yes
SendEMail (Page 1519)	Yes
SetAcousticSignal (Page 1584)	No
SetDisplayMode (Page 1573)	Yes
SetDeviceMode (Page 1573)	Yes
SetBit (Page 1565)	Yes
SetBitInTag (Page 1567)	Yes
SetBitWhileKeyPressed (Page 1569)	Yes
SetBacklightColor (Page 1585)	No
SetBrightness (Page 1584)	No
SetScreenKeyboardMode (Page 1574)	Yes
SetAlarmReportMode (Page 1571)	Yes
SetRecipeTags (Page 1645)	Yes
SetDaylightSavingTime (Page 1520)	Yes
SetLanguage (Page 1576)	Yes
SetTag (Page 1555)	Yes
SetConnectionMode (Page 1577)	Yes
SetWebAccess (Page 1578)	Yes
BackupRAMFileSystem (Page 1522)	No
SimulateSystemKey (Page 1523)	Yes
SimulateTag (Page 1524)	Yes
SmartClientViewRefresh (Page 1610)	Yes
SmartClientViewReadOnlyOff (Page 1611)	Yes
SmartClientViewReadOnlyOn (Page 1611)	Yes
SmartClientViewDisconnect (Page 1612)	Yes
SmartClientViewConnect (Page 1612)	Yes
SmartClientViewLeave (Page 1613)	Yes
SaveDataRecord (Page 1647)	Yes
StartLogging (Page 1544)	Yes
StartNextLog (Page 1545)	Yes

	WinCC RT Advanced
StartProgram (Page 1525)	Yes
StatusForceGetValues (Page 1527)	Yes
StatusForceSetValues (Page 1527)	Yes
ControlSmartServer (Page 1528)	Yes
ControlWebServer (Page 1528)	Yes
StopLogging (Page 1546)	Yes
StopRuntime (Page 1529)	Yes
ActivateSystemDiagnosticsView (Page 1613)	Yes
LookupText (Page 1530)	Yes
ResetTagToHandWheel (Page 1531)	No
SetTagToHandWheel (Page 1531)	No
TraceUserChange (Page 1551)	Yes
DecreaseFocusedValue (Page 1532)	Yes
DecreaseTag (Page 1557)	Yes
ChangeConnection (Page 1579)	Yes
ChangeConnectionEIP (Page 1580)	Yes
AUTOHOTSPOT	Yes
ShowLogonDialog (Page 1551)	Yes
ShowOperatorNotes (Page 1532)	Yes
ShowAlarmWindow (Page 1621)	Yes
ShowPopupScreen (Page 1534)	Yes
ShowPopupScreenSizable (Page 1535)	Yes
ShowSlideInScreen (Page 1538)	Yes
ShowSoftwareVersion (Page 1533)	Yes
ShowSystemDiagnosticsWindow (Page 1616)	Yes
ShowSystemAlarm (Page 1622)	Yes

1) Not available for devices as of device version V14.0 and higher

See also

TerminatePROFIsafe (Page 1519)

System functions for Runtime Professional (RT Professional)

Availability of system functions

The following table shows the availability of the system functions for WinCC Runtime Professional.

Technical data subject to change.

Overview

	WinCC RT Professional
User-defined functions	Yes
ActivateScreen (Page 1674)	Yes
ActivateScreenInCurrentScreenWindow (Page 1675)	Yes
ActivateScreenInScreenWindow (Page 1675)	Yes
IncreaseTag (Page 1660)	Yes
ExportImportUserAdministration (Page 1658)	Yes
InvertBit (Page 1688)	Yes
InvertBitInTag (Page 1690)	Yes
InvertLinearScaling (Page 1661)	Yes
LinearScaling (Page 1663)	Yes
ResetBit (Page 1691)	Yes
ResetBitInTag (Page 1693)	Yes
SetBit (Page 1695)	Yes
SetBitInTag (Page 1696)	Yes
SetPropertyCurrentWindow (Page 1687)	Yes
SetPropertyCurrentWindowByProperty (Page 1686)	Yes
SetPropertyCurrentWindowByTagIndirect (Page 1687)	Yes
SetPropertyByProperty (Page 1679)	Yes
SetPropertyByConstant (Page 1680)	Yes
SetPropertyByTag (Page 1682)	Yes
SetPropertyByTagIndirect (Page 1684)	Yes
SetTag (Page 1664)	Yes
SetTagByProperty (Page 1666)	Yes
SetTagByTagIndirect (Page 1670)	Yes
SetTagWithOperatorEvent (Page 1671)	Yes
SetTagIndirect (Page 1667)	Yes
SetTagIndirectByProperty (Page 1668)	Yes
SetTagIndirectByTagIndirect (Page 1669)	Yes
StopRuntime (Page 1656)	Yes
DecreaseTag (Page 1672)	Yes
ShowLogonDialog (Page 1659)	Yes
ShowPLCCodeViewFromAlarm (Page 1702)	Yes
ShowBlockInTiaPortalFromAlarm (Page 1700)	Yes

9.11.1.2 System functions (RT Advanced, Basic Panels, Panels, Comfort Panels)

Other functions (RT Advanced, Basic Panels, Panels, Comfort Panels)

ActivateScreen (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Performs a screen change to the given screen.

Use the "ActivateScreenByNumber" system function to change from the root screen to the permanent area or vice versa.

Use in the function list

ActivateScreen (Screen name, Object number)

Use in user-defined functions

ActivateScreen Screen_name, Object_number

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Name of the screen to which you change.

Object number

The operator control element which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent area when the system function is called up, the permanent area maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

Note

If the "Reach margin" event is assigned to the "ActivateScreen" system function, only the value "0" is valid for the "Object number" parameter. The active object is not defined by the object number, but rather by the X position it had prior to the screen change.

Example

The following program code activates "Screen_2" with the ActivateScreen function, for example, when you click a button.

```
Sub ActivateScreen_2()

'User-defined code
'' i. e. when pressing a button

ActivateScreen "Screen_2",0
```

See also

ActivateScreenByNumber (Page 1506)

ActivateScreenByNumber (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Performs a screen change to a screen depending on a tag value.

The screen is identified by its screen number.

Use in the function list

ActivateScreenByNumber (Screen number, Object number)

Use in user-defined functions

ActivateScreenByNumber Screen_number, Object_number

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Screen number

Tag which contains the screen number of the destination screen.

When a change from the root screen to the permanent area is desired, specify "0" or "-1":

0 = Change from root screen to permanent area.

-1 = Change from permanent area to root screen

Object number

The number of the screen object which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent area when the system function is called up, the permanent area maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

See also

ActivateScreen (Page 1505)

ActivatePreviousScreen (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Performs a screen change to the screen which was activated before the current screen. The screen change is not performed if no screen was activated beforehand.

The last 10 screens that were called up are saved. A system alarm is output when you change to a screen which is no longer saved.

Note

If you want to use the system function, the screen to which you change has to be used in the navigation structure.

Use in the function list

ActivatePreviousScreen

Use in user-defined functions

ActivatePreviousScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

UpdateTag (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Reads the current value of the tag with the specified Update ID from the PLC.

Use in the function list

UpdateTag (Update ID)

Use in user-defined functions

-

Parameters

Update ID

Update ID assigned to the tag that will be updated.

See also

Logoff (Page 1547)

Encode (RT Advanced, Panels, Comfort Panels)

Description

Adapts the "String" data type of a tag for transfer to the automation system (AS). The tag of WinCC data type "String" is converted into the AS data type "Array of byte". The result is written to a tag.

Use in the function list

Encode (Byte array, String, Coding)

Use in user-defined functions

Encode Byte_array, String, Encoding

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Byte array

The tag that contains the converted value.

Note

The Byte array must be twice the size of the string length + 2. The Byte array must contain 242 array elements if the string has a length of 120 characters.

Characters are either truncated or not converted if the size is insufficient.

String

The tag of data type "String" that is converted.

Encode

0 (hmiEncodeUTF16LE) - String is encoded in UTF16LE (Unicode 16 Little Endian).

EncodeEx (RT Advanced, Panels, Comfort Panels)

Description

Adapts the "String" data type of a tag for transfer to the automation system (AS). The tag of WinCC data type "String" is converted into the AS data type "Array of byte". The result is written to a tag.

By contrast to the Encode system function, you can define the Line break parameter. Using the Line break parameter you can delete line breaks or replace these with predefined characters.

Use in the function list

EncodeEx (Byte-Array, String, Encoding, Line break)

Use in user-defined functions

EncodeEx Byte_array, String, Encoding, Line_break

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Byte array

The tag that contains the converted value.

Note

The byte array must be twice the size of the string length +2. If the string has a length of 120 characters, the byte array must contain 242 array elements.

Characters are either truncated or not converted if the size is insufficient.

String

The tag of data type "String" that is converted.

Encoding

0 (hmiEncodeUTF16LE) - String is encoded in UTF16LE (Unicode 16 Little Endian).

Line break

All line breaks are either deleted or replaced with predefined characters. Do not replace line breaks if set as the default value.

0 (replace with "\r\n' (0x000D, 0x000A)) - the line break is replaced with "\r\n".

1 (replace with "\n' (0x000A)) - the line break is replaced with "\n".

2 (do not replace) - the line break is not replaced.

3 (remove line breaks) - the line breaks are removed.

DirectKey (RT Advanced, Panels, Comfort Panels)

Description

Allows rapid operation of keys on an HMI device without having communication caused delays occur.

Use this system function, for example. to set bits in the I/O area of a SIMATIC S7 directly from the HMI device.

Note

If you have integrated WinCC in SIMATIC STEP 7 and have configured a DP area in SIMATIC STEP 7: When the button on which the system function was configured is activated, a bit is set in the IO area of the CPU. Releasing the button resets the bit.

Notes on configuring the system function

When a button is configured with the "DirectKey" system function, a fixed area of the touch display is reserved for DirectKey operation.

Screen objects that are above the button with the "DirectKey" system function in this area in runtime will hide the button visually. However, the screen objects do not interfere with the triggering of the "DirectKey" system function.

Note

If an external application, such as Pocket Internet Explorer or Control Panel is started, it will run in the foreground and Runtime is placed in the background. The bit for the "DirectKeyScreenNumber" function is no longer set and the keys or buttons which are assigned the "DirectKey" function no longer to trigger the associated bit in the PLC.

WARNING

Triggering the "DirectKey" system function by mistake will endanger personnel or result in damage to the machine.

In order to avoid this danger, the following must be observed:

- When the process screen is configured, the button with the "DirectKey" system function must not be covered by a screen object.
- The dynamic positioning or display (release) of a screen object in relation to process values must not cover the button with the "DirectKey" system function in runtime.

  Note

  Please observe this guideline during configuration. Check also existing configurations and adjust them immediately.

In the following cases, the covering of a button with the "DirectKey" system function does not lead to the problems described above:

- Operation of the alarm window
- Operation of the Recipe view
- Cancellation of the screen saver
- Use of the screen keyboard
- Running the "ActivateCleanScreen" system function

Actions of DP direct-keys in offline operation

When the "ChangeConnection" system function is performed with the "Offline" parameter on the HMI device, the connection to the specified PLC is disconnected.

Note

Please note the following on the SIMATIC STEP 7 side as well as the WinCC side:

The DP DirectKeys are still active in this case. If you activate a button with the "DirectKey" system function in "Offline" operating mode or activate the DirectKey on a keyboard device, the corresponding bit is set in the PLC.

Use in the function list

DirectKey (Bit)

Use in user-defined functions

-

Parameters

Bit

Specifies the bit which is set. Depending on the HMI device, the bit numbers 0 to 31 or 0 to 39 are possible.

Application example

Objective

You want to set bits in the IO area of a SIMATIC S7 directly from the HMI device using a button.

Requirement

WinCC integrated in SIMATIC STEP 7 is installed.

During operation, the HMI device is coupled to a SIMATIC S7 through PROFIBUS-DP.

WinCC integrated was installed when you compiled your project.

The bit area for DirectKeys is determined in SIMATIC STEP 7. For configuration see the "Communication" user's manual.

Notes on configuring

Configure the button to be used as the DirectKey. Assign the "DirectKey" system function to the button. As a parameter, enter the number of the bit which is set when the key is pressed.

Procedure on HMI device

When the DirectKey is touched, the bit is set, and reset when the key is released or the screen is exited.

DirectKeyScreenNumber (RT Advanced, Panels, Comfort Panels)

Description

Sets the bit within the given bit area of a DirectKey and transfers it to the S7 controller to which the HMI device is connected. This ensures unambiguous allocation of a control bit to screen number at all times.

Without the use of the system function, the S7 controller must distinguish the respective functionality by means of the screen number. This delays the updating of the screen number after a screen change.

Note

If an external application, such as Pocket Internet Explorer or Control Panel is started, it will run in the foreground and Runtime is placed in the background. Nevertheless, the bit for the "DirectKeyScreenNumber" function is set and the keys or buttons with the configured "DirectKey" function continue to trigger the associated bit in the PLC.

Use in the function list

DirectKeyScreenNumber (Bit)

Use in user-defined functions

-

Parameters

Bit

Specifies the bit which is set. Depending on the HMI device, the bit numbers 0 to 31 or 0 to 39 are possible.

NotifyUserAction (RT Advanced, Panels, Comfort Panels)

Description

This system function is used to log user actions that are not automatically logged in the Audit Trail. You can also use this system function to require the user to enter an acknowledgment or an electronic signature and a comment for the operator action. A requirement for the use of the system function is that the GMP-compliant configuration is activated under "Runtime settings > GMP".

If you use the "NotifyUserAction" system function in a function, the debugger may open if you cancel your input by clicking "Cancel". To compensate for this reaction, you can use the "On Error Resume Next" statement in a function. With this instruction, the next instruction is executed after a runtime error. If you use the "On Error Resume Next" statement, output of system events is also suppressed.

Use in the function list

NotifyUserAction (Confirmation type, Mandatory commenting, Category, Object name, Description)

Use in user-defined functions

NotifyUserAction Confirmation_type, Mandatory_commenting, Category, Object_name, Description

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Confirmation type

Establishes how the action must be confirmed

0 = (None): No confirmation required, an entry is created in the Audit Trail

1 = (Acknowledgement): Acknowledgment, the user must acknowledge the action; an entry is created in the Audit Trail

2 = (Digital Signature): Electronic signature; a dialog window opens in which the user must enter the electronic signature - an entry is created in the Audit Trail

Mandatory commenting

Establishes if the user has to enter a comment. The comment is logged in the Audit Trail.

0 = (True): True; a dialog window opens in which the user must enter a comment

1 = (False): False; no comment required

Category

Category or class name of the modified object

Object name

Name of the modified object

Description

Text which describes the archiving user action.

IncreaseFocusedValue (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Adds the given value to the value of the tag which is connected to the input field (drop-down list, graphic selection list, slider bar) which has the current focus.

This system function can only be used for function keys.

Use in the function list

IncreaseFocusedValue (Value)

Use in user-defined functions

-

Parameters

Value

The value which is added to the tag value.

SafelyRemoveHardware (Basic Panels, Panels, Comfort Panels)

Description

Checks whether there is read or write access to the external storage medium. If there is no access, the external storage medium can be removed without data loss.

Use in the function list

SafelyRemoveHardware(Path, Result)

Use in user-defined functions

SafelyRemoveHardware Path, Result

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Result

The tag in which the result is entered.

TRUE: The storage medium can be removed safely. A corresponding system alarm will be output.

FALSE: The storage medium cannot be removed safely. A corresponding system alarm will be output.

Path

Path of storage medium, e.g. \Storage Card USB\

OpenControlPanelDialog (RT Advanced, Panels, Comfort Panels)

Description

Opens a dialog that you can use to edit selected Control Panel settings.

This system function lets you set the following on the HMI device:

- Properties and value of the IP address
- User identification on the network
- WinCC Internet settings

Note

Project security

The "OpenControlPanelDialog" system function is used to bypass the SecureMode on the HMI device. Take the corresponding precautions to ensure the security of your project.

Use in the function list

OpenControlPanelDialog (Dialog)

Use in user-defined functions

-

Parameters

Dialog

Sets the Control Panel dialog to be opened.

- PROFINET_X1: Setting of the IP address and Ethernet parameters
- PROFINET_X3: Setting of the IP address and Ethernet parameters; only for Comfort Panel KP 1500, TP 1500; TP1900, TP2200
- WinCC Internet settings: Setting for Web Server, e-mail notification, provided the HMI device supports these functions
- Network ID: Setting for identification on the network, provided the HMI device supports these functions

OpenCommandPrompt (RT Advanced, Panels, Comfort Panels)

Description

Opens a Windows system prompt.

This function is used, e.g., to copy files or to call up another application.

Use in the function list

OpenCommandPrompt

Use in user-defined functions

OpenCommandPrompt

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

OpenInternetExplorer (RT Advanced, Panels, Comfort Panels)

Description

Opens the Internet Explorer on the HMI device.

If the Internet Explorer is already open, it will be closed and reopened when you call the system function.

Note

The Internet Explorer saves data temporarily in the DRAM file system of the HMI device, e.g. the last web sites that were be called up.

This data can be saved using the "BackupRAMFileSystem" system function so that it is still available when the HMI device is restarted.

Use in the function list

OpenInternetExplorer (Start page)

Use in user-defined functions

OpenInternetExplorer Start_page

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Start page

The page which is loaded when Internet Explorer is started, e.g. "www.siemens.com".

OpenControlPanel (RT Advanced, Panels, Comfort Panels)

Description

Opens the window that displays the Windows CE control panel. You cannot use this system function on a PC.

This system function enables you to set the following on the Windows CE-based HMI device:

- Select printer
- Select transfer properties
- Configure screen saver
- Configure flash memory

Note

No backup or restore during Runtime

Backup and restore functions may only be executed if Runtime has been terminated. Otherwise, this could result in undesired effects, such as display errors.

Use in the function list

OpenControlPanel

Use in user-defined functions

OpenControlPanel

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

--

TerminatePROFIsafe (RT Advanced, Panels, Comfort Panels)

Description

Disconnects the PROFIsafe connection for fail-safe operation between a KTP Mobile Panel and the PLC.

After execution of the system function "TerminatePROFIsafe", the connector of the KTP Mobile Panel can be removed from the PLC without the system signaling an error.

Use in the function list

TerminatePROFIsafe

Use in user-defined functions

TerminatePROFIsafe

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

--

See also

System functions for Mobile Panels (Page 1491)

Device dependency (Page 1474)

SendEMail (RT Advanced, Panels, Comfort Panels)

Description

Sends an e-mail from the HMI device to the given addressee.

This system function is used, for example, when, in the case of service, the alarm is to be passed on directly to the service technician.

Note

To send alarms as e-mails, the HMI system must have an e-mail client at its disposal.

Use in the function list

SendEMail (Address, Subject, Text, Return address)

Use in user-defined functions

SendEMail Address, Subject, Text, Return_address

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Address

The e-mail address of the addressee.

Subject

The subject line of the e-mail.

Text

The text sent in the e-mail.

Return address

The e-mail address to which the addressee of this e-mail should send the reply.

SetDaylightSavingTime (RT Advanced, Panels, Comfort Panels)

Description

The system function "SetDaylightSavingTime" changes the setting of the HMI device from daylight saving to standard time and vice versa.

Time settings will take place immediately following system function.

Note

The "SetDaylightSavingTime" system function does not support time zones without daylight saving time.

Note

Windows 7

The system function "SetDaylightSavingTime" is not supported for PC-based HMI devices under Windows 7.

Use in the function list

SetDaylightSavingTime(DaylightSavingTime)

Use in user-defined functions

SetDaylightSavingTime Daylight_saving_time

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Daylight Saving Time

Determines whether Daylight Saving Time is set on the HMI device:

0 = Daylight Saving Time is not activated

1 = Daylight Saving Time is activated

PressButton (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

You configure the system function to the function keys of an HMI device. The system function "PressButton" triggers the function that is assigned to the event "PressKey" at the specified screen object.

Use this system function when you want to operate a button in a screen with a function key of the HMI device, for example.

Note

The "PressButton" and "ReleaseButton" system functions must always be configured together. If you configure the "PressButton" system function on the "Press key" event for a function key, the "ReleaseButton" system function is configured on the "Release" event for the same function key.

Use in the function list

PressButton (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object on which the event is triggered.

ReleaseButton (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

The system function can only be configured on the function keys of an HMI device and triggers the "Release button" event at the specified screen object.

Use this system function when you want to operate a button in a screen with a function key of the HMI device, for example.

Note

The "PressButton" and "ReleaseButton" system functions must always be configured together. If you configure the "PressButton" system function on the "Press key" event for a function key, then the "ReleaseButton" system function is configured on the "Release key" event for the same function key.

Use in the function list

ReleaseButton (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object on which the event is triggered.

BackupRAMFileSystem (RT Advanced, Panels, Comfort Panels)

Description

Backs up the RAM file system in the memory medium of the HMI device.

After restarting the HMI device, the data is automatically reloaded in the RAM file system.

Applications such as the Internet Explorer save data (e.g. the last web sites called up) temporarily to the DRAM file system of the HMI device.

Use in the function list

BackupRAMFileSystem

Use in user-defined functions

BackupRAMFileSystem

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

SimulateSystemKey (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Simulates the behavior of a system key. Use this system function if a system key, such as the "ACK" key, "Input" key or the number pad is not available on the HMI device.

Use in the function list

SimulateSystemKey (System key)

Use in user-defined functions

-

Parameters

System key

System key whose behavior is to be simulated.

System key "+/-"

With the SimulateSystemKey system function, the system key "+/-" is only supported for the following HMI devices:

- KP300 Basic
- KP400 Basic
- KTP400 Basic mono PN
- KTP400 Basic color PN
- KTP400 Basic color PN Portrait
- KTP600 Basic mono PN
- KTP600 Basic color PN
- KTP600 Basic color DP
- KTP700 Basic PN / DP
- KTP900 Basic PN
- KTP1000 Basic PN
- KTP1000 Basic DP
- KTP1200 Basic PN / DP
- Comfort Panels

Use the system keys "+" and "-" separately for all other HMI devices.

SimulateTag (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Simulates the behavior of tags and dynamic objects such as text lists, without having the HMI device connected to a PLC. You can, for example, configure the system function to the "Loaded" event of a screen.

This system function is used, for example, to demonstrate the functionality of a project.

Only tags of the data type Integer can be used for simulation.

Note

If you use the system function "SimulateTag" with a short cycle time on a Basic Panel, the HMI device may be overloaded.

Use in the function list

SimulateTag (Tag, Cycle, Maximum value, Minimum value, Value)

Use in user-defined functions

-

Parameter

Tag

The tag whose value is changed.

Cycle

The factor by which the basic cycle of 200 milliseconds is multiplied. The cycle defines when the tag value is changed by the specified value. Possible cycles between 1 and 32767.

Maximum value

The maximum value that the tag can assume during simulation. The maximum value must be greater than the minimum value but less than / equal to 32767.

Minimum value

The minimum value that the tag can assume during simulation. The minimum value must be lower than the maximum value but greater than / equal to -32768.

Value

The value by which the tag value is changed during each cycle. Possible values between -32768 and 32767.

- A positive value increases the tag value. When the maximum value is reached, the tag value is set to the minimum value after the next update cycle.
- A negative value reduces the tag value. When the minimum value is reached, the tag value is set to the maximum value after the next update cycle.

StartProgram (RT Advanced, Panels, Comfort Panels)

Description

Starts the specified program on the HMI device.

The runtime software continues running in the background. Alarms continue to be output and data continues to be updated.

When the given application is exited, the screen which was active during the performance of the system function is displayed on the HMI device.

This system function is used, for example, to edit recipe data records in MS Excel on the HMI device.

Note

If Windows CE is installed on the HMI device, during the configuration it must be checked whether the desired application can be started with this system function.

This system function allows all applications to be started which can be started in the "Execute" dialog of Windows CE.

The application to be started must be installed on the HMI device.

Note

If an application is not executed via the "StartProgram" function on a Mobile Panel or Comfort Panel, check the path specification of the system function. On Mobile Panels and Comfort Panels, some applications are located in the folder "\flash\addons".

Use in the function list

StartProgram (Program name, Program parameters, Display mode, Wait for program to end)

Use in user-defined functions

StartProgram Program_name, Program_parameters, Display_mode, Wait_for_program_to_end

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Program name

Name and path of the program which is started. Upper and lower case are taken into account in this parameter.

Note

If the path contains spaces, the program can only be started correctly if the path is specified in inverted commas, e.g. "C:\Program Files\START\start.exe".

Program parameters

The parameters you transfer at the start of the program, for example a file that is opened after the start of the program.

The description of the necessary parameters is found in the documentation of the program to be started.

Display mode

Determines how the program window is displayed on the HMI device:

0 (hmiShowNormal) = Normal

1 (hmiShowMinimized) = Minimized

2 (hmiShowMaximized) = Maximized

3 (hmiShowMinimizedAndInactive) = Minimized and inactive

Wait for program to end

Determines whether there is a change back to the project after the called up program has ended:

0 (hmiNo) = No: No change to project

1 (hmiYes) = Yes: Change to project

Note

The "Wait for program to end" parameter is only available for Runtime Advanced and Panels.

StatusForceGetValues (RT Advanced, Panels, Comfort Panels)

Description

Starts or stops the updating of values in the watch table. The values are read from the PLC connected to the HMI device until the update is stopped.

Note

As long as the values are updated, no entries are possible in the input fields of the watch table.

Use in the function list

StatusForceGetValues (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the watch table to which data from the PLC is written.

See also

Watch table (Page 344)

StatusForceSetValues (RT Advanced, Panels, Comfort Panels)

Description

Writes the values from the watch table to the PLC to which the HMI device is connected.

Use in the function list

StatusForceSetValues (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the watch table from which data is written to the PLC.

See also

Watch table (Page 344)

ControlSmartServer (RT Advanced, Panels, Comfort Panels)

Description

Starts or stops the Sm@rtServer.

Use in the function list

ControlSmartServer (Mode)

Use in user-defined functions

ControlSmartServer Mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Specifies whether the Sm@rtServer is started or stopped.

-1 (hmiToggle) = Toggle: Toggles between the two modes

0 (hmiStop) = Stop: Sm@rtServer is stopped

1 (hmiStart) = Start: Sm@rtServer is started

ControlWebServer (RT Advanced, Panels, Comfort Panels)

Description

Starts or stops the Web server.

Use in the function list

ControlWebServer (Mode)

Use in user-defined functions

ControlWebServer Mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Specifies whether the Web server is started or stopped.

-1 (hmiToggle) = Toggle: Toggles between the two modes

0 (hmiStop) = Stop: The Web server is stopped.

1 (hmiStart) = Start: The Web server is started.

StopRuntime (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Exits the runtime software and thereby the project running on the HMI device.

Use in the function list

StopRuntime (Mode)

Use in user-defined functions

StopRuntime Mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Mode

Determines whether the operating system is shut down after exiting runtime.

0 (hmiStopRuntime) = Runtime: Operating system is not shut down

1 (hmiStopRuntimeAndOperatingSystem) = Runtime and operating system: The operating system is shut down (not possible with WinCE)

Example

The following program code shuts down Runtime and the operating system.

```
{

//Stop runtime and shutdown
StopRuntime (hmiStopRuntimeAndOperationSystem);

}
```

The saved return value can be processed in the following code.

LookupText (RT Advanced, Panels, Comfort Panels)

Description

Identifies an entry from a text list. The result depends on the value and on the selected runtime language. The result is saved to a tag of data type "String".

Use in the function list

LookupText (Output text, Index, Language, Text list)

Use in user-defined functions

LookupText Output_text, Index, Language, Text_list

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Output text

The tag to which the result is written.

Index

The tag that defines the value of the list entry.

Language

Defines the runtime language in which the list entry is identified.

- Runtime language
  Language code as per VBScript reference, for example, "de-DE" for German (Germany) or "en-US" for English (United States of America). The selection depends on the activated runtime languages.
- Tag
  The tag that contains the language. Enter the runtime language as decimal value of the country ID, for example, 1031 for German - Standard, 1033 for English - USA. Details are available in the VBScript basics, "Locale identifier (LCID) diagram".
- Integer
  The number that corresponds to the order of runtime languages for language switching. The selection depends on the activated runtime languages, for example, "0" for the language that appears when you first start runtime. You can find more information under the topic of "Languages in Runtime".

Text list

Defines the text list. The list entry is read from the text list.

ResetTagToHandWheel (RT Advanced, Panels, Comfort Panels)

Description

Separates the tag that is connected with the operating element handwheel, and reconnects the handwheel with the global tags.

Use in the function list

ResetTagToHandWheel

Use in user-defined functions

-

Parameters

--

SetTagToHandWheel (RT Advanced, Panels, Comfort Panels)

Description

Connect the tag with the operating device handwheel If the handwheel is operated, the tag value changes. The connection can be reset using the "ResetTagToHandWheel" system function.

Use in the function list

SetTagToHandWheel (Value)

Use in user-defined functions

-

Parameters

Value

Tag names which are connected to the operating device handwheel.

DecreaseFocusedValue (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Subtracts the specified value from the value of the tag which is connected to the screen object and currently has the focus.

This system function can only be used for function keys.

Use in the function list

DecreaseFocusedValue (Value)

Use in user-defined functions

-

Parameters

Value

The value which is subtracted from the tag value.

ShowOperatorNotes (RT Advanced, Basic Panels, Panels, Comfort Panels)

Application

Displays the tooltip configured for the selected object.

If the system function is configured on a function key, the tooltip for the screen object that currently has the focus is displayed. If a tooltip is configured for the screen itself, you can switch to this text by pressing <Enter> or by double-clicking on the help window.

If the system function is configured on a button, only the tooltip for the current screen is displayed. If a tooltip is configured on the button itself, initially only the tooltip for the button is displayed. You can press <Enter> or double-click on the help window to switch to the tooltip for the current screen.

Note

No other screen object can be used while the help window is open. To use the screen object, close the help window.

Closing the help window

You can close the help window in the following ways:

Using the keys:

- By pressing the <HELP> key again
- By pressing the <ESC> key

Using the touch screen:

- By pressing the ❌ button

Use in the function list

ShowOperatorNotes (Layout)

Use in user-defined functions

ShowOperatorNotes Display_mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines whether the configured tooltip is hidden or shown:

0 (hmiOff) = Off: Configured tooltip is hidden

1 (hmiOn) = On: Configured tooltip is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

ShowSoftwareVersion (RT Advanced, Panels, Comfort Panels)

Description

Hides or shows the version number of the runtime software.

Use this system function if during servicing, for example, you required the version of the runtime software used.

Use in the function list

ShowSoftwareVersion (Display mode)

Use in user-defined functions

ShowSoftwareVersion Display_mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines whether the version number is shown:

0 (hmiOff) = Off: Version number is not shown

1 (hmiOn) = On: Version number is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

ShowPopupScreen (RT Advanced, Panels, Comfort Panels)

Description

Opens the pop-up-screen, for example, when a button is pressed.

You can enter a constant value or assign a tag as coordinates.
If the configured pop-up screen is not visible or only partially visible, the coordinates are set to 0.0.

Regardless of the size of the permanent area, the start position of the coordinates is always 0.0.

Use in the function list

ShowPopupScreen (Name of the screen, X coordinate, Y coordinate, Display mode, Animation, Animation speed)

Use in user-defined functions

ShowPopupScreen Screen_name, X_coordinate, Y_coordinate, Display_mode, Animation, Animation_speed

Parameters

Name of the screen

Specifies the name of the pop-up screen that appears in Runtime when a button is pressed.

X coordinate

Position of the pop-up screen in the current screen on the X axis

Y coordinate

Position of the pop-up screen in the current screen on the X axis

Display mode

Specifies the mode for the pop-up screen:

Switching

Off

On

Animation

Specifies the direction from which the pop-up screen is shown:

Off

Left

Top

Right

Bottom

Animation speed

Specifies the speed at which the pop-up screen is shown:

Slow

Medium

Fast

ShowPopupScreenSizable (RT Advanced, Panels, Comfort Panels)

Description

Opens the pop-up screen with a defined size. You can open a pop-up screen in a different size with this system function; the scroll bars are displayed for navigating in the content of the pop-up screen.

You can enter a constant value or assign a tag as coordinates. If the configured pop-up screen is not visible or only partially visible, the coordinates are set to 0.0. Regardless of the size of the permanent area, the start position of the coordinates is always 0.0.

You also define the displayed width and height of the pop-up screen, which can deviate from the sizes of the configured pop-up screen. The scroll bars are displayed in the pop-up screen in this case.

Note

Operation on HMI devices with keys

You scroll in pop-up screens using the keyboard shortcut <ALT>+<arrow key>.

Display only one scroll bar

If you specify only one of the sizes smaller than the size of the configured pop-up screen in the parameters of the system function, the two scroll bars are displayed in the pop-up screen. You have two options to configure only one scroll bar in the pop-up screen:

- Reduce the width/height of the configured pop-up screen by the size of the device-specific scroll bar.
- Specify a value that corresponds to the actual size of the pop-up screen for one of the size parameters and add the device-specific size of the scroll bar. You specify the value for the other size as needed.
  Example:
  You want to open a pop-up screen with the configured size of 500 x 420 in the size 500 x 300 with a vertical scroll bar. Specify the value 534 for the width parameter which corresponds to the actual width of the configured pop-up screen plus the size of the scroll bar of 34. Enter the value 300 for the height. The pop-up screen is opened in the size 500 x 300 in Runtime and only the vertical scroll bar is displayed.

Device type (version V14)	Width	Height
HMI devices with 4" display Comfort Panels with 19" display Runtime Advanced	25	25
HMI devices with 9" display Comfort Panels with 15" display Comfort Panels with 22" display	27	27
HMI devices with 7" display HMI devices with 12" display	34	34

Use in the function list

ShowPopupScreenSizable (Name of the screen, X coordinate, Y coordinate, Width, Height, Display mode, Animation, Animation speed)

Use in user-defined functions

ShowPopupScreenSizable Screen_name, X_coordinate, Y_coordinate, Width, Height, Display_mode, Animation, Animation_speed

Parameters

Name of the screen

Specifies the name of the pop-up screen that appears in Runtime when a button is pressed.

X coordinate

Position of the pop-up screen in the current screen on the X axis

Y coordinate

Position of the pop-up screen in the current screen on the X axis

Width

Specifies the width of the pop-up screen. The maximum width must not exceed the screen width of the configured HMI device.

Height

Specifies the height of the pop-up screen. The maximum height must not exceed the screen width of the configured HMI device.

Display mode

Specifies the mode for the pop-up screen:

Switching

Off

On

Animation

Specifies the direction from which the pop-up screen is shown:

Off

Left

Top

Right

Bottom

Animation speed

Specifies the speed at which the pop-up screen is shown:

Slow

Medium

Fast

See also

System functions for Runtime Advanced (Page 1498)

ShowSlideInScreen (RT Advanced, Panels, Comfort Panels)

Description

Calls the slide-in screen, for example, when operating a button.

Use in the function list

ShowSlideinScreen (screen name, mode)

Use in user-defined functions

ShowSlideInScreen SlideInScreen_name, Mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Specifies the slide-in screen that appears in Runtime when a button is pressed:

Top Slide-in screen

Bottom Slide-in screen

Left Slide-in screen

Right Slide-in screen

Mode

Specifies the mode for the slide-in screen:

Switching

Off

On

Logs (RT Advanced, Basic Panels, Panels, Comfort Panels)

ArchiveLogFile (RT Advanced, Panels, Comfort Panels)

Description

This system function moves or copies a log to another storage location for long-term logging.

With Audit Trails, always use the "Move" (hmiMove)" mode, otherwise you will be violating the FDA guideline by duplicating the data stored.

Prior to "ArchiveLogFile" always run the system function "CloseAllLogs".

After this system function, run the "OpenAllLogs" system function.

In "Copy log" mode, the logs are not reopened until the log has been successfully copied, or unless a timeout occurs during the copy. In "Move log" mode, the logs to be moved are renamed and the new logs are opened immediately. To move the renamed logs, a job is submitted which attempts to move them every 300 seconds if the server cannot be reached. The job is also retained after Runtime is restarted until it is executed.

Use in the function list

ArchiveLogFile (Log type, Log, Directory name, Mode)

Use in user-defined functions

ArchiveLogFile Log_type, Log, Directory_name, Mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Tag log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAuditrailArchive)= Audit trail log available for GMP compliant projects.. Additional information is available under "Activating GMP-compliant configuration".

Log

Name of the log being archived.

Directory name

Path for saving the log.

Mode

0 (hmiCopy) = Copy log

1 (hmiMove) = Move log

Application example

You want to move a log file from a local storage medium to the server in order to generate a backup copy of this file at cyclic intervals.

Notes on configuring

Set up a task in the scheduler which is executed at a specific time on a daily basis. Configure the following function list for the task:

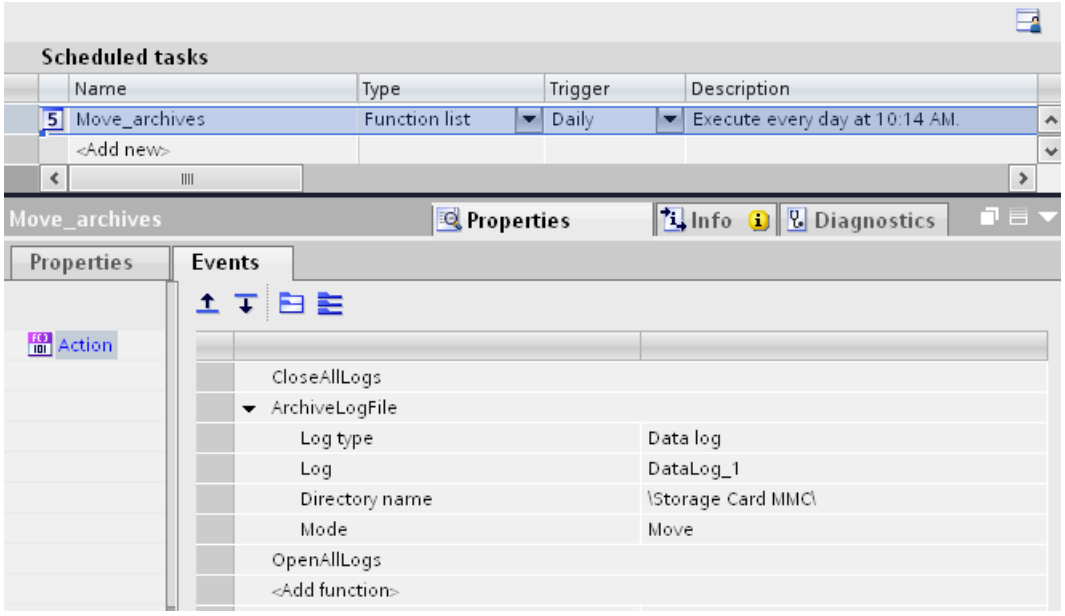

Procedure on HMI device

- All log files will be closed.
- The log file you specified is moved to the server.
- All closed log files will be opened again.

LogTag (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Saves the value of the given tags in the given data log.

This system function is used to archive a process value at a certain point in time.

Use in the function list

LogTag (Tag)

Use in user-defined functions

-

Parameters

Tag

The tag whose value is logged. The tag is stored in the log which is configured for the specified tag.

CopyLog (RT Advanced, Panels, Comfort Panels)

Description

Copies the contents of a log to another log. Tag values can only be copied to other data logs and alarms only to other alarm logs, however.

Note

If you copy a log using the "CopyLog" system function, it is possible that external applications will be unable to read certain country-specific special characters in the logged message text of the log copy. This does not apply to WinCC Runtime. WinCC Runtime can read the copied log files without errors.

Note

80% of the log entries are copied when copying the circular logs. 20% of entries are not copied because the space is reserved for buffer overflow by default.

Use in the function list

CopyLog (Log type, Destination log, Source log, Mode, Delete source log)

Use in user-defined functions

CopyLog Log_type, Destination log, Source_log, Mode, Delete_source_log

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Tag log

1 (hmiAlarmArchive) = Alarm log

Destination log

Name of the log into which the entries are copied (Destination log).

Source log

Name of the log from which the entries are copied (Source log).

Mode

Determines what is done with copied entries in the destination log:

0 (hmiOverwrite) = Overwrite: Existing entries are overwritten.

2 (hmiAppend) = Append: The entries are inserted at the end of the destination log. When the configured size of the log has been reached, the destination log is treated like a circular log.

Delete source log

Determines whether the source log is deleted after copying.

0 (hmiNo) = No: Is not deleted.

1 (hmiYes) = Yes: Is deleted.

ClearLog (RT Advanced, Panels, Comfort Panels)

Description

Deletes all data records in the given log.

Use in the function list

ClearLog (Log type, Log)

Use in user-defined functions

ClearLog Log_type, Log

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Tag log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAuditrailArchive) = Audit trail log. Available for GMP-compliant projects Additional information is available under "Activating GMP-compliant configuration".

Log

Name of the log from which all entries are deleted.

OpenAllLogs (RT Advanced, Panels, Comfort Panels)

Description

Reestablishes the connection between WinCC and the logs. Logging can be continued.

Note

Run the "StartLogging" system function in order to restart logging.

Use in the function list

OpenAllLogs

Use in user-defined functions

OpenAllLogs

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "StopLogging" and "CloseAllLogs" system functions on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When the button "Close Archive" is pressed, the specified log is stopped and all open logs are closed. The data medium can be changed. You open all logs with the "Open Archive" button. Logging will be continued in the specified log.

CloseAllLogs (RT Advanced, Panels, Comfort Panels)

Description

Disconnects the connection between WinCC and all logs.

Use in the function list

CloseAllLogs

Use in user-defined functions

CloseAllLogs

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "CloseAllLogs" system function on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When you press the "Close Archive" button, the opened logs are closed. The data medium can be changed. The logging continues during the change of the data medium. The process values to be logged are cached. You open all logs with the "Open Archive" button. The logging process will continue in the specified log. The buffered process values are later added to the log.

StartLogging (RT Advanced, Panels, Comfort Panels)

Description

Starts the logging of data or alarms in the specified log. The function can also be applied to audit trails.

You can interrupt logging at runtime using the "StopLogging" system function.

Use in the function list

StartLogging (Log type, Log)

Use in user-defined functions

StartLogging Log_type, Log

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Tag log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAuditrailArchive) = Audit trail

Log

Name of the log which is started.

See also

Managing logging behavior when Runtime starts (Page 621)

StartNextLog (RT Advanced, Panels, Comfort Panels)

Description

Stops the logging of data or alarms for the given log.

Logging is continued in the next log of the segmented circular log you configured for the specified log.

If you did not configure a segmented circular log for the specified log, the system function has no effect.

Use in the function list

StartNextLog (Log type, Log)

Use in user-defined functions

StartNextLog Log_type, Log

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Tag log

1 (hmiAlarmArchive) = Alarm log

Log

Name of the log for which the logging is stopped and continued in the next log.

StopLogging (RT Advanced, Panels, Comfort Panels)

Description

Stops the logging of process values or alarms in the specified log. The function can also be applied to audit trails.

The "StartLogging" system function is used to resume logging at runtime.

Note

When logging is stopped, a connection between WinCC and the log files or log database still exists. Use the "CloseAllLogs" system function to disconnect this connection.

Use in the function list

StopLogging (Log type, Log)

Use in user-defined functions

StopLogging Log_type, Log

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Tag log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAuditrailArchive) = Audit trail

Log

Name of the log that is stopped.

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "StopLogging" and "CloseAllLogs" system functions on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When the button "Close Archive" is pressed, the specified log is stopped and all open logs are closed. The data medium can be changed. The "Open Archive" button opens all logs and continues logging in the specified log.

User administration (RT Advanced, Basic Panels, Panels, Comfort Panels)

Logoff (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Logs off the current user on the HMI device.

Use in the function list

Logoff

Use in user-defined functions

Logoff

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Logon (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Logs on the current user on the HMI device.

Use in the function list

Logon (Password, User name)

Use in user-defined functions

Logon Password, User_name

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Password

The tag from which the password for the user logging on is read. If the user is logged on, the password in the tag is deleted.

User name

The tag from which the user name for the user logging on is read.

ExportImportUserAdministration (RT Advanced, Panels, Comfort Panels)

Description

Exports all users of the user administration of the currently active project to the given file or imports the users from the given file into the currently active project.

Users, their passwords and rights are saved in the user administration.

All users are overwritten when importing. The imported users are valid immediately.

Note

Exporting and importing of the user administration from V14 is not supported on HMI devices with a Runtime version older than V14.

Use in the function list

ExportImportUserAdministration (File name, Direction)

Use in user-defined functions

ExportImportUserAdministration File_name, Direction

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

File name

Name of the file which contains the passwords or to which the passwords are written. Enter the file location and the file extension (*.txt), for example, "C:\TEMP\Passwords.txt".

Note

If a storage card is used as file location, specify the file location as follows: "\StorageCard \<FileName".

Direction

Specifies whether passwords are exported or imported:

0 (hmiExport) = Export: Passwords are exported.

1 (hmiImport) = Import: Passwords are imported.

See also

Exporting the user administration (Page 1352)

Importing the user administration (Page 1353)

GetUserName (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Writes the user name of the user currently logged on to the HMI device in the given tag.

If the given tag has a control connection, the user name is also available in the PLC connection. This system function makes it possible, for example, to implement a user-dependent release of certain functionalities.

Use in the function list

GetUserName (Tag)

Use in user-defined functions

GetUserName Tag

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the user name is written.

GetGroupNumber (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Reads the number of the group to which the user logged on to the HMI device belongs, and writes it to the given tag.

Use in the function list

GetGroupNumber (Tag)

Use in user-defined functions

GetGroupNumber Tag

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the number of the group is written.

GetPassword (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Writes the password of the user currently logged on to the HMI device in the given tag.

Note

Make sure that the value of the given tag is not displayed in another place in the project.

Note

The passwords of SIMATIC Logon users cannot be read.

Note

The "GetPassword" system function is not available for devices as of device version 14.0.0.0.

If this system feature is used in a project older than version V14, it is removed after migration to V14. The event linked to the system function is thereby invalidated. The "GetPassword" system function appears to be faulty in the configuration. A warning is generated during compiling. The "GetPassword" system function is no longer included in the compiled Runtime project.

Use in the function list

GetPassword (Tag)

Use in user-defined functions

GetPassword Tag

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the password is written.

TraceUserChange (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Outputs a system event that shows which user is currently logged in on the HMI device.

This system function can only be used in the Scheduler.

Use in the function list

TraceUserChange

Use in user-defined functions

-

Parameters

--

ShowLogonDialog (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Opens a dialog on the HMI device with which the user can log on to the HMI device.

Use in the function list

ShowLogonDialog

Use in user-defined functions

-

Parameters

--

Calculation (RT Advanced, Basic Panels, Panels, Comfort Panels)

IncreaseTag (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Adds the given value to the value of the tags.

X = X + a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, help tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

IncreaseTag (Tag, Value)

Use in user-defined functions

IncreaseTag Tag, Value

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the given value is added.

Value

The value that is added.

See also

SetTag (Page 1555)

InvertLinearScaling (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Assigns a value to the tag X, which is calculated from the value of the given tag Y using the linear function X = (Y - b) / a.

The tags X and Y must not be identical. This system function is the inverse of the "LinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

InvertLinearScaling (X, Y, b, a)

Use in user-defined functions

InverseLinearScaling X, Y, b, a

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

X

The tag which is assigned the value calculated from the linear equation.

Y

The tag that contains the value used for calculation.

b

The value which is subtracted.

a

The value through which is divided.

Example

The following program code assigns a value to the varX tag by means of the InverseLinearScaling function.

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
 InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n, varX);
...
}
```

The saved return value can be processed in the following code.

LinearScaling (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Assigns a value to the tag Y, which is calculated from the value of the given tag X using the linear function Y= (a *X) + b.

The inverse of this function is the "InvertLinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

LinearScaling (Y, a, X, b)

Use in user-defined functions

LinearScaling Y, a, X, b

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Y

The tag which is assigned the value calculated from the linear equation.

a

The value with which is multiplied.

X

The tag that contains the value used for calculation.

b

The value that is added.

Example

The following program code uses the LinearScaling function to assign a value to the Yvar tag.

```
{
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
 LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n, Yvar);
...
}
```

The saved return value can be processed in the following code.

SetTag (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Assigns a new value to the given tag.

Note

This system function can be used to assign strings and numbers, depending on the type of tag.

Use in the function list

SetTag (Tag, Value)

Use in user-defined functions

SetTag Tag, Value

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag to which the given value is assigned.

Value

The value which the given tag is assigned.

Note

The "SetTag" system function is only executed after a connection has been established.

Example

The following program code uses the SetTag function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag.

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
bvalue = GetTagBit("gs_tag_bit");
  printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

The saved return value can be processed in the following code.

DecreaseTag (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Subtracts the given value from the tag value.

X = X - a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. The auxiliary tags are assigned to the tag value with the "SetTag" system function.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

DecreaseTag (Tag, Value)

Use in user-defined functions

DecreaseTag Tag, Value

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag from which the given value is subtracted.

Value

The value which is subtracted.

Bit processing (RT Advanced, Basic Panels, Panels, Comfort Panels)

InvertBit (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Inverts the value of the given tag of the "Bool" type:

- If the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

Use in the function list

InvertBit (Tag)

Use in user-defined functions

InvertBit Tag

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag whose bit is set.

Example

The following program code inverts the value of the Boolean tag bStatus and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit
InvertBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Invert variable
invertBit(bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
  ...
}
```

InvertBitInTag (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Inverts a bit in the given tag:

- If the bit in the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the bit in the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "InvertBit" system function instead.

Use in the function list

InvertBitInTag (Tag, Bit)

Use in user-defined functions

InvertBitInTag Tag, Bit

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which the given bit is set.

Bit

The number of the bit that is set.

When this system function is used in a user-defined function, the bits in a tag are counted from right to left. The counting begins with 0.

Example

The following program code inverts a bit at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit in position
bitposition=2
InvertBit(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BYTE bStatusWord;
BYTE bsaved = bStatusWord;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bStatusWord, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bsaved);
  ...
}
```

ResetBit (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Sets the value of a "Bool" type tag to 0 (FALSE).

Use in the function list

ResetBit (Tag)

Use in user-defined functions

ResetBit Tag

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 0 (FALSE).

Example

The following program code sets the value of the Boolean tag bStatus to 0 by means of the ResetBit function and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=1
myTag.Value=bValue

'Save current value
bSaved=bValue

'Reset Bit
ResetBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 1;
BOOL bSaved = bStatus;

//Reset bit
  ResetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

See also

ResetBitInTag (Page 1562)

ResetBitInTag (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Sets a bit in the specified tag to 0 (FALSE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "ResetBit" system function instead.

Use in the function list

ResetBitInTag (Tag, Bit)

Use in user-defined functions

ResetBitInTag Tag, Bit

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 0 (FALSE).

Bit

The number of the bit that is set to 0 (FALSE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Example

The following program code sets a bit to 0 at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Reset Bit
bitposition=2
ResetBit(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BYTE bStatusWord;
BYTE bSaved = bStatusWord;
BYTE bitposition = 2;

//Reset bit in bitposition
ResetBitInTag (bStatusWord, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bSaved);
  ...
}
```

See also

ResetBit (Page 1560)

ShiftAndMask (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

This system function converts the input bit pattern of the source tags into an output bit pattern of the target tags. This involves bit shifting and masking.

Note

If the source and target tag have a different number of bits, using the system function in the target tag can result in a violation of the value range.

Use in the function list

ShiftAndMask (Source tag, Target tag, Bits to shift, Bits to mask)

Use in user-defined functions

ShiftAndMask Source_tag, Target_tag, Bits_to_shift, Bits_to_mask

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Source tag

The tag includes the input bit pattern. Integer-type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The actual value 72 is set at the 16-bit integer source tag: 0000000001001000.

Target tag

The output bit pattern is saved in the tag. Integer type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The shifted input bit pattern is multiplied by the bit mask, with bit-by-bit logical AND operation: 0000000000001001. The resultant decimal value "8" is saved to the target tag.

Please note the following:

- The source and target tags have the same number of bits.
- The number of bits to shift is less than the number of bits in the source tag and target tag.
- Bits to mask does not have more bits than the source tag and the target tag.

Bits to shift

Number of bits by which the input bit pattern is shifted right. A negative value shifts the input bit pattern to the left.

Example: "Bits to shift" has the value "+3". The input bit pattern is shifted right by three bits when the system function is called: 0000000000001001.

Bits to the left are padded with "0". Three bits are truncated on the right. The new decimal value is "9".

Note

The left bit is "1" in a source tag of the data type with negative signed integer. This sign bit is padded with "0" when the bits are shifted right. The sign changes to "+".

Bits to mask

An integer serves as bit mask. The bit pattern is used to multiply the shifted input bit pattern. Example: Integer "2478" with the bit pattern "0000100110101110".

You can enter the bit mask in three different ways:

- Hexadecimal: First enter the prefix "0h" or "0H", followed by an optional space for better readability. Then group the bit pattern in blocks of four (0000)(1001)(1010)(1110) and set each block in hexadecimal code: (0)(9)(A)(E). Only the characters 0-9, A-F, a-f are allowed: "0h 09AE".
- Binary: First enter the prefix "0b" or "0B", followed by an optional space for better readability. Then group the binary bit pattern into blocks of four 0000 1001 1010 1110 with spaces in between as a check. Only the characters "0" or "1" are allowed: "0b 0000 1001 1010 1110".
- Decimal: Enter the value "2478" directly, without a prefix.

Note

If you change the device version of the target HMI device after configuration (e.g."13.1.0" to "13.0.0" or vice versa), you must check and test the parameters of this system function.

You can use the "Char" and "Word" data types for the "Source tag" and "Target tag" parameters as of device version 13.1.0. In the device versions before 13.1.0, these parameters must be assigned other data types:

- Use "SInt" instead of "Char"
- Use "Int" instead of "Word"

Otherwise, there may be unwanted effects, for example incorrect or unexpected behavior of the configured system functions.

SetBit (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Sets the value of a "Bool" type tag to 1 (TRUE).

Use in the function list

SetBit (Tag)

Use in user-defined functions

SetBit Tag

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 1 (TRUE).

Example

The following program code sets the value of the Boolean tag bStatus to 1 by means of the SetBit function and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=0
myTag.Value=bValue

'Save current value
bSaved=bValue

'Set Bit
SetBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Set bit
  SetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

SetBitInTag (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Sets a bit in the given tag to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitInTag (Tag, Bit)

Use in user-defined functions

SetBitInTag Tag, Bit

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 1 (TRUE).

Bit

The number of the bit that is set to 1 (TRUE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an I/O field or assign the system function to a screen object, such as a button.

If you have configured a short event such as the activation of an alarm for the system function you can only access the actual process values by setting the tag for continuous reading.

Example

The following program code sets a bit to 1 at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Set Bit in tag
bitposition=2
SetBitInTag(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BYTE bStatusWord;
BYTE bSaved = bStatusWord;
BYTE bitposition = 2;

//Reset bit in bitposition
SetBitInTag (bStatusWord, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bSaved);
  ...
}
```

See also

SetBit (Page 1564)

SetBitWhileKeyPressed (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Sets the bit of the given tag to 1 (TRUE) as long as the user keeps the configured key pressed.

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC. You should only access tags of the BOOL type with this system function to avoid problems with overlapping access to the same tag.

Note

All functions on the event "Release" are performed immediately by means of a screen change configured for a key, even if the key is kept pressed.

If the "SetBitWhileKeyPressed" system function is configured for a function key, the bit will be reset immediately following a screen change. This action is necessary since the key assignments change after the screen change.

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitWhileKeyPressed (Tag, Bit)

Use in user-defined functions

-

Parameters

Tag

The tag in which a bit is temporarily set to 1 (TRUE). Use only tags of the type BOOL, as far as allowed by the PLC.

Bit

The number of the bit that is temporarily set to 1 (TRUE).

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an IO field, or assign the function to a screen element such as a button.

If you configured a short event such as the activation of an alarm for the function you can only access the actual process values by setting the tag for continuous reading.

See also

SetBit (Page 1564)

Printing (RT Advanced, Panels, Comfort Panels)

PrintScreen (RT Advanced, Panels, Comfort Panels)

Description

Prints the screen currently being displayed on the HMI device from the printer which is connected to the HMI device.

Opened windows are also printed.

Note

The printer settings are taken over from the current settings in the Windows operation system.

Use in the function list

PrintScreen

Use in user-defined functions

PrintScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

PrintReport (RT Advanced, Panels, Comfort Panels)

Description

Prints the given report from the printer which is connected to the HMI device. The report is printed in the language which is set on the HMI device.

Note

If runtime is closed whilst log data are being printed using the system function, the report will cease to be supplied with data as soon as runtime stops.

Use in the function list

PrintReport (Report)

Use in user-defined functions

PrintReport Report

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Report

Name of the report to be printed.

Note

If you have set up via the "Function list" dialog box a new report for the "PrintReport" function, when compiling, the following warning appears: "The report "Report_1" has no printed pages."

In order to remedy the warning, open the "Report_1" via the project view and recompile the project.

SetAlarmReportMode (RT Advanced, Panels, Comfort Panels)

Description

Switches the automatic reporting of alarms on the printer on or off.

Use in the function list

SetAlarmReportMode (Mode)

Use in user-defined functions

SetAlarmReportMode Mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether alarms are reported automatically on the printer:

0 (hmiDisablePrinting) = Reporting off: Alarms are not printed automatically.

1 (hmiEnablePrinting) = Reporting on: Alarms are printed automatically.

-1 (hmiToggle) = Toggle: Toggles between the two modes.

Settings (RT Advanced, Basic Panels, Panels, Comfort Panels)

GetPLCMode (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Evaluates the current state of the connected PLC.

The system function "GetPLCMode" can only be configured for the following PLCs:

- SIMATIC S7-1200
- SIMATIC S7-1500

Use in the function list

GetPLCMode (Connection, Mode)

Use in user-defined functions

-

Parameters

Connection

Connection of PLC and HMI device.

Mode

Evaluates the state of the connected PLC.
8: RUN = The PLC program is being executed.
4: STOP = The PLC program has been interrupted.
Select a suitable user tag for the evaluation.

SetDisplayMode (RT Advanced, Panels, Comfort Panels)

Description

Changes the settings of the screen in which the runtime software runs.

The runtime software runs in full-screen mode as default. The Windows task switch is disabled.

Use in the function list

SetDisplayMode (Layout)

Use in user-defined functions

SetDisplayMode Display_mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines the settings for the screen in which the runtime software runs.

1 (hmiScreenFull): Full screen: Title bar of the screen is not visible

2 (hmiScreenMaximized): Maximized

3 (hmiScreenRestore): Restore: The last used screen setting is used. This layout can only be used when the window is displayed minimized or maximized.

4 (hmiScreenMinimized): Minimized

5 (hmiScreenAutoAdjust): Automatic: The size of the window is set so that all screen objects in it will be visible.

6 (hmiScreenOnTop): Foreground; either the window appears in the foreground or the program icon associated with the window flashes on the taskbar depending on the Windows setting. The setting can be changed in the Windows configuration and applies to all Windows applications.

SetDeviceMode (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Toggles the operating mode on the HMI device. The following types of operation are possible: "Online", "Offline" and "Loading".

Use in the function list

SetDeviceMode (Operating mode)

Use in user-defined functions

SetDeviceMode Operating_mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Operating mode

Specifies the operating mode of the HMI device. For devices with Windows CE, this parameter cannot be supplied via tag of data type BOOL.

0 (hmiOnline) = online: The connection to the PLC is established. The configured connection status is always set in this process. The states that were last used in Runtime are not considered.

1 (hmiOffline) = offline: The connection to the PLC is disconnected.

2 (hmiTransfer) = Load: A project can be transferred from the configuration computer to the HMI device.

Note

If you use a PC as an HMI device, the runtime software will be exited when you switch operating mode after "Load".

See also

SetConnectionMode (Page 1577)

SetScreenKeyboardMode (RT Advanced, Panels, Comfort Panels)

Description

Enables or disables the automatic display of the screen keyboard on the HMI device.

This system function is also used to prevent the display of the screen keyboard, e.g. because an external keyboard is connected to the HMI device.

Note

To enable the "SetScreenKeyboardMode" ("SetScreenKeyboardMode") system function on an HMI other than a Touch Panel device, set the "Use on-screen keyboard" check box in the "Runtime settings" dialog of the device settings.

Use in the function list

SetScreenKeyboardMode (Mode)

Use in user-defined functions

SetScreenKeyboardMode Mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether the screen keyboard is hidden or shown:

0 (hmiOff) = Off: Screen keyboard is hidden

1 (hmiOn) = On: Screen keyboard is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes.

SetPLCDateTime (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Changes the data and the time of the linked PLC

The system function "SetPLCDateTime" can only be configured for the following PLCs:

- SIMATIC S7-1200
- SIMATIC S7-1500

Use in the function list

SetPLCDateTime (connection, time)

Use in user-defined functions

-

Parameters

Connection

Connection of PLC and HMI device.

Time

Transfers the date and the time of the HMI device to the PLC. The PLC applies the date and the time of the HMI device.

SetPLCMode (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Switches the operating mode of the PLC to one of the following states:

- RUN
- STOP

The system function "SetPLCMode" can only be configured for the following PLCs:

- SIMATIC S7-1200
- SIMATIC S7-1500

Use in the function list

SetPLCMode (connection, mode)

Use in user-defined functions

-

Parameters

Connection

Connection of PLC and HMI device.

Mode

Specifies the operating mode of the PLC:

8: RUN = The controller is set to the RUN state. The control program is executed.

4: STOP = The controller is set to the STOP state. The PLC program is interrupted.

See also

SystemDiagnosticViewUpdatePlcBuffer (Page 1614)

SetLanguage (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Toggles the language on the HMI device. All configured text and system events are displayed on the HMI device in the newly set language.

Use in the function list

SetLanguage (Language)

Use in user-defined functions

SetLanguage Language

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Language

Determines which language is set on the HMI device. The following specifications are possible:

- -1 (hmiToggle) = Toggle: Changes to the next language. The sequence is determined during configuration in the "Project languages" editor.
- Number you have defined under "Languages and fonts" in the "Runtime Settings" editor. Changes to the language with the given number.
- Language you have defined under "Languages and fonts" in the "Runtime Settings" editor.
- Language abbreviation in accordance with the VBScript5 reference: This changes to the language corresponding to the specified language code, e.g. "de-DE" for German (Germany) or "en-US" for English (United States).
  An overview of the language abbreviations is available in the basic information of VBScript under the topic "Area diagram-ID (LCID) Diagram".

SetConnectionMode (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Connects or disconnects the given connection.

Note

A connection to the PLC cannot be established until the "Online" operating mode has been set on the HMI device. Use the "SetDeviceMode" system function for this purpose.

If the connection is loaded in "Offline" mode, the connection is closed again each time the mode switches to "Offline". To re-establish the connection after switching to "Online" mode, set the connection again to "Online" mode.

Use in the function list

SetConnectionMode (Mode, Connection)

Use in user-defined functions

SetConnectionMode Mode, Connection

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Specifies whether a connection to the PLC is established or disconnected. For devices with Windows CE, this parameter cannot be supplied via tag of data type BOOL.

0 (hmiOnline) = online: Connection is established.

1 (hmiOffline) = offline: Connection is disconnected.

Connection

The PLC to which the HMI device is connected. You specify the name of the PLC in the connection editor.

Multiple use of the system function in a user-defined function

If you use the "SetConnectionMode" system function for different connections, it may be possible that not all system functions are executed correctly. Proceed as follows to prevent this situation:

1. Create a tag with the start value "0".
2. Configure the "SetConnectionMode" system function on the "Value change" event of the HMI tags. If you want to disconnect three connections, for example, you must configure the system function three times.
3. In the user-defined function, apply the "InvertBit" system function to the HMI tag.

Application example

Two typical application examples for this system function are as follows:

- Test
  As long as no PLC is connected to the HMI device, no error messages will be output during the test on the HMI device. If the HMI device is connected to a PLC, the connection to the PLC can be established by pressing a key.
- Commissioning
  Several PLCs are to be configured for a system. At first, all PLCs except one are configured "Offline". After commissioning of the first PLC, the connection to each of the other PLCs is established by pressing a key. In this way, the other PLCs are started up one after another.

SetWebAccess (RT Advanced, Panels, Comfort Panels)

Description

Determines the access mode to the runtime application using the Internet.

Use in the function list

SetWebAccess (Access mode)

Use in user-defined functions

-

Parameters

Access mode

Determines the access mode to the runtime application:

-1 (hmiToggle) = Toggle: Toggles between the two modes.

0 (hmiReadOnly) = Read-only.

1 (hmiReadWrite) = Read-write.

ChangeConnection (RT Advanced, Panels, Comfort Panels)

Description

Disconnects the connection to the PLC currently in use and establishes a connection to a PLC with another address. The newly connected PLC must belong to the same device class (S7-300, S7-400, etc).

Note

When changing to another address, ensure that the address is not already being used by another HMI device.

The follows address types are supported:

- IP address
- MPI address

The follows PLC types are supported:

- SIMATIC S7 300/400
- SIMATIC S7 200
- SIMATIC S7 1200 (device version up to V11)
- SIMATIC S7 LOGO!
- SIMATIC S7-NC
- SIMOTION

Use in the function list

ChangeConnection (Connection, Address, Slot, Rack)

Use in user-defined functions

ChangeConnection Connection, Address, Slot, Rack

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Connection

Name of the connection that is disconnected. The name is set during configuration, for example, in the "Connections" editor.

Address

MPI/PROFIBUS or IP address of the PLC to which the connection will be established.

Note

Set the address by means of a tag. The objects list displays the tags of all data types. Select only tags of the following data types:

- Ethernet connection: "String" data type
- MPI connection: "Int" data type

Slot

Slot of the PLC to which the connection will be established.

Rack

Rack of the PLC to which the connection will be established.

Application example

You want to operate one HMI device on several machines. Configure a controller in the project. When changing the PLC, the connection to the PLC in use is disconnected. Then the connection to the new PLC with other address parameters is reestablished. All tags in this connection are now updated from the new controller.

The PLC which you have indicated when creating the project will be used as default.

1. Enter the name and address of the PLC in the "Connections" editor.
2. Configure a button in the process screen.
3. Configure the "ChangeConnection" system function on the "Press" event.
4. Enter the name of the connection and address of the PLC as parameters.

ChangeConnectionEIP (RT Advanced, Panels, Comfort Panels)

Description

Terminates the connection to the Allan Bradley controller currently in use and establishes a connection to another Allan Bradley controller in Runtime. "EIP" stands for "Ethernet/IP".

The newly connected controller must belong to the same device class (Allen Bradley).

For the connection to be established, the CPU type of the controller to be connected must match the CPU type of the controller already connected, for example, both controllers have the CPU type "SLC/Micrologix PLC".

Note

When changing to another address, ensure that the address is not already being used by another HMI device.

The follows address types are supported:

- IP address

Use in the function list

ChangeConnectionEIP (Connection, Address, Communication path)

Use in user-defined functions

ChangeConnectionEIP Connection, Address, Communication_path

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Connection

Name of the connection that is disconnected. The name is set during configuration, for example, in the "Connections" editor.

Address

IP address of the controller to which the connection is being established.

Note

Set the address by means of a tag. The objects list displays the tags of all data types. Select only tags of the following data type "String".

Note

The stored IP address must consist of 4 decimal numbers with the value range 0 to 255.

Communication path

CIP path from the Ethernet module to the controller to which the connection is being established.

You specify the communication path either with a String tag or an HMI tag of the "String" data type.

Application example

You want to operate one HMI device on several machines. To do so, you configure a PLC in the project. When changing the PLC, the connection to the PLC in use is disconnected. Then the connection to the new PLC with other address parameters is reestablished. All tags of this connection are now updated from the new PLC.

1. Enter the name and address of the PLC in the "Connections" editor.
2. Configure a button in the process screen.
3. Configure the "ChangeConnectionEIP" system function on the "Press" event.
4. Enter the name of the connection and address of the PLC as parameters.

Functions of the display (RT Advanced, Basic Panels, Panels, Comfort Panels)

AdjustContrast (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Changes the contrast of the display one level on the HMI device.

Use in the function list

AdjustContrast (Adjust)

Use in user-defined functions

-

Parameters

Adjust

Specifies how the contrast is changed:

0 (hmiDecrease) = Decrease: Decreases the contrast one level.

1 (hmiIncrease) = Increase: Increases the contrast one level.

Application example

Objective

One button each for increasing and decreasing the screen contrast is desired.

Notes on configuring

Configure two buttons and configure the "AdjustContrast" system function on the "Press" event. The parameters "Increase" and "Decrease" are allocated.

Procedure on HMI device

When one of the two buttons is pressed in runtime, the contrast is increased or decreased one level.

See also

Logoff (Page 1546)

CalibrateTouchScreen (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Calls a program for calibrating the touch screen.

During the calibration process, there is a prompt to touch five positions on the screen display. Touch the screen display within 30 seconds, to confirm the calibration process. If the calibration is not completed within this time span, the calibration settings are discarded. The user prompt is in English.

Use this system function the first time you start the HMI device.

Use in the function list

CalibrateTouchScreen

Use in user-defined functions

CalibrateTouchScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Note

The CalibrateTouchScreen system function cannot be simulated.

SetAcousticSignal (RT Advanced, Panels, Comfort Panels)

Description

Configures the acoustic feedback of touch screen operation on the HMI device.

Note

The configuration that was set at Switch off is reestablished when restarting the HMI device.

Use in the function list

SetAcousticSignal (Volume)

Use in user-defined functions

SetAcousticSignal Volume

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Volume

Determines whether and how loud an acoustic signal is emitted:

-1 (hmiToggle) = Toggle: Toggles the emission of the acoustic signal as follows: Muted > Quiet > Loud.

0 (hmiMuted) = Mute: no acoustic signal

1 (hmiQuiet) = Quiet: quiet acoustic signal

2 (hmiLoud) = Loud: loud acoustic signal

SetBrightness (RT Advanced, Panels, Comfort Panels)

Description

Determines the brightness of the display.

Note

The configuration that is set in the Control Panel / Start Center will be reestablished when you restart the HMI device.

For Basic Panels 2nd Generation, Mobile Panels and Comfort Panels:

The value for the system function "SetBrightness" can be set between 0% and 100%. The set value is transferred to the HMI device. The brightness settings on the HMI device can be viewed

and edited in "Start Center > Settings > Display". The HMI devices support a brightness setting between 10% and 100%.

If the system function "SetBrightness" is assigned a value of 0%, the display of the HMI device is switched off by default in Runtime. If the operator touches the display, the display switches to the previous brightness setting.

If the system function "SetBrightness" is assigned a value between 1% and 10% and the operator opens the display settings in the Start Center, brightness is reset to 10%.

Use in the function list

SetBrightness (value)

Use in user-defined functions

SetBrightness Value

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Value

New value for the brightness.

SetBacklightColor (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Defines the background lighting of the button.

Note

The configuration that was set at Switch off is reestablished when restarting the HMI device.

Use in the function list

SetBacklightColor (Value)

Use in user-defined functions

-

Parameters

Value

Defines the background lighting of the button:

0 (hmiWhite) = White: No color

1 (hmiGreen) = Green: Green color

2 (hmiYellow) = Yellow: Yellow color

3 (hmiRed) = Red: Red color

GetBrightness (RT Advanced, Panels, Comfort Panels)

Description

Reads the value of the brightness.

Use in the function list

GetBrightness (Brightness)

Use in user-defined functions

GetBrightness Brightness

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Brightness

The tag to which the value is written.

Functions of the HTML browser (RT Advanced, Basic Panels, Panels, Comfort Panels)

HTMLBrowserStop (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Performs the function "Stop" the HTML browser.

Use in the function list

HTMLBrowserStop (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is performed.

HTMLBrowserScrollDown (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Scrolls down in the HTML browser.

Use in the function list

HTMLBrowserScrollDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

HTMLBrowserRefresh (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Performs the function "Refresh" of the HTML browser.

Use in the function list

HTMLBrowserRefresh (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is performed.

HTMLBrowserScrollUp (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Scrolls up in the HTML browser.

Use in the function list

HTMLBrowserScrollUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

HTMLBrowserZoomIn (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Zooms in one zoom level in the HTML browser display.

Use in the function list

HTMLBrowserZooomIn (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

HTMLBrowserZoomOut (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Zooms out one zoom level in the HTML browser display.

Use in the function list

HTMLBrowserZoomOut (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

HTMLBrowserScrollLeft (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Scrolls left in the HTML browser.

Use in the function list

HTMLBrowserScrollLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

HTMLBrowserScrollRight (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Scrolls right in the HTML browser.

Use in the function list

HTMLBrowserScrollRight (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

HTMLBrowserPageUp (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Results in a page-by-page, upward cursor movement in the HTML browser.

Use in the function list

HTMLBrowserPageUp (screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

HTMLBrowserPageDown (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Results in a page-by-page, downward cursor movement in the HTML browser.

Use in the function list

HTMLBrowserPageDown (screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

HTMLBrowserHome (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Switches to the start page stored for the HTML browser.

Use in the function list

HTMLBrowserHome (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is executed.

HTMLBrowserForward (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Performs the function "Forward" of the HTML browser.

Use in the function list

HTMLBrowserForward (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is performed.

HTMLBrowserBack (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Performs the "Back" function of the HTML browser.

Use in the function list

HTMLBrowserBack (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Object name of the HTML browser in which the command is performed.

Functions of the trend view (RT Advanced, Basic Panels, Panels, Comfort Panels)

TrendViewScrollForward (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Scrolls forward one display width in the Trend view.

Use in the function list

TrendViewScrollForward (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which is scrolled forward.

TrendViewScrollBack (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Scrolls back one display width in the trend view.

Use in the function list

TrendViewScrollBack (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which is scrolled back.

TrendViewExtend (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Reduces the time period which is displayed in the trend view.

Use in the function list

TrendViewExtend (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the displayed time period is reduced.

TrendViewCompress (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Increases the time period which is displayed in the trend view.

Use in the function list

TrendViewCompress (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the displayed time period is increased.

TrendViewRulerLeft (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Moves the read-line backwards (to the left) in the trend view.

Note

In order to be able to move the read-line, the read-line must have been switched on. This is done using the "TrendViewSetRulerMode" system function.

Use in the function list

TrendViewRulerLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is moved backwards.

See also

TrendViewSetRulerMode (Page 1595)

TrendViewRulerRight (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Moves the read-line forwards (to the right) in the trend view.

Note

In order to be able to move the read-line, the read-line must have been switched on. This is done using the "TrendViewSetRulerMode" system function.

Use in the function list

TrendViewRulerRight (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is moved forward.

See also

TrendViewSetRulerMode (Page 1595)

TrendViewSetRulerMode (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Hides or shows the read-line in the trend view. The read-line displays the Y value belonging to the X value.

Note

To ensure that the ruler is displayed, you have to activate the setting "Show ruler" in the trend view properties.

Use in the function list

TrendViewSetRulerMode (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is hidden or shown.

See also

TrendViewRulerRight (Page 1594)
TrendViewRulerLeft (Page 1593)

TrendViewStartStop (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Stops the trend recording or continues the trend recording in the trend view.

Use in the function list

TrendViewStartStop (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the recording of the trend is started or stopped.

TrendViewBackToBeginning (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Scrolls back to the beginning of the display range in the trend view.

Use in the function list

TrendViewBackToBeginning (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which you scroll to the beginning of the display range.

See also

System functions for Runtime Advanced (Page 1498)

Functions of the PDF view (RT Advanced, Panels, Comfort Panels)

PDFScrollDown (RT Advanced, Panels, Comfort Panels)

Description

Scrolls down in the PDF file in the PDF view.

Use in the function list

PDFScrollDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFScrollUp (RT Advanced, Panels, Comfort Panels)

Description

Scrolls up in the PDF file in the PDF view.

Use in the function list

PDFScrollUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFFitToWidth (RT Advanced, Panels, Comfort Panels)

Description

Fits the display of the PDF file to the height of the PDF view window.

Use in the function list

PDFFitToWidth (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFFitToHeight (RT Advanced, Panels, Comfort Panels)

Description

Fits the display of the PDF file to the height of the PDF view window.

Use in the function list

PDFFitToHeight (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFGotoFirstPage (RT Advanced, Panels, Comfort Panels)

Description

Switches to the first page of the PDF file in the PDF view.

Use in the function list

PDFGotoFirstPage (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFGotoLastPage (RT Advanced, Panels, Comfort Panels)

Description

Switches to the last page of the PDF file in the PDF view.

Use in the function list

PDFGotoLastPage (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFGotoNextPage (RT Advanced, Panels, Comfort Panels)

Description

Switches to the next page of the PDF file in the PDF view.

Use in the function list

PDFGotoNextPage (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFGotoPage (RT Advanced, Panels, Comfort Panels)

Description

Switches to the entered page of the PDF file in the PDF view.

Use in the function list

PDFGotoPage (Screen object, Page)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

Page

Page in the PDF document that is displayed.

PDFGotoPreviousPage (RT Advanced, Panels, Comfort Panels)

Description

Switches to the previous page of the PDF file in the PDF view.

Use in the function list

PDFGotoPreviousPage (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFZoomIn (RT Advanced, Panels, Comfort Panels)

Description

Zooms in one zoom level in the display of the PDF file in the PDF view.

Use in the function list

PDFZoomIn (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFZoomOut (RT Advanced, Panels, Comfort Panels)

Description

Zooms out one zoom level in the display of the PDF file in the PDF view.

Use in the function list

PDFZoomOut (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFScrollLeft (RT Advanced, Panels, Comfort Panels)

Description

Scrolls to the left in the PDF file in the PDF view.

Use in the function list

PDFScrollLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFScrollRight (RT Advanced, Panels, Comfort Panels)

Description

Scrolls to the right in the PDF file in the PDF view.

Use in the function list

PDFScrollRight (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

PDFZoomOriginal (RT Advanced, Panels, Comfort Panels)

Description

Switches the display of the PDF file in the PDF view to the original size.

Use in the function list

PDFZoomOriginal (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PDF view in which the command is executed.

Functions of the PLC code view (RT Advanced, Basic Panels, Panels, Comfort Panels)

ActivatePLCCodeViewer (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Performs a screen change to the given screen with the PLC code view. The PLC code view shows the program code of the relevant network.

You configure the ActivatePLCCodeView system function at the "Click PLC code view button" event of a GRAPH overview or at an event of a button.

Screen change from the GRAPH overview

You configure the system function at the "Click PLC code view button" event in a GRAPH overview to display the GRAPH step sequencer in the PLC code view. If no error is pending, the sequence of the GRAPH step sequencer is displayed.

If an error occurs, the faulty step is displayed in the PLC code view after jump of the faulty step. If multiple errors have occurred, the first faulty step of the sequencer is displayed after the jump. If you have corrected the error, the next faulty step is automatically displayed in the view. In the detail view, the transition or the interlock is displayed depending on the type of error.

Screen change from an alarm

You configure the system function at an event of a button. When the button is pressed in Runtime, the system function checks whether the alarm selected last in the configured alarm view is a supervision alarm or a GRAPH alarm. If a jump is possible for this alarm, the configured PLC code view is opened with the corresponding program code.

The jump from an alarm in the alarm view to the PLC code view is made possible for the following supervision alarms:

- With global supervisions, only for interlock supervisions (Interlock)
- With local supervisions, for all basic supervisions at input parameters

You can find more information on supervisions in the section "Programming PLC > Supervising machinery and plants with ProDiag".

The jump to the PLC code view is possible for all GRAPH alarms. After the jump, the GRAPH sequencer and the faulty step are displayed in the PLC code view.

If a supervision alarm and an interlock alarm are pending simultaneously for the same step, the interlock network is always displayed first after triggering the system function in the PLC code view regardless of whether the supervision alarm or the interlock alarm was selected.

For the jump from a supervision alarm to the PLC code view, the instance name must conform to the following naming convention when using supported local operands in a function block: <FB-Name>_DB.

The jump to a function or an organization block is only possible if only global operands are used.

Note

The jump in connection with the alarm buffer or the alarm log is not supported.

Use in the function list

ActivatePLCCodeView(Screen name, Screen object)

Use in user-defined functions

ActivatePLCCodeView Screen_name, Object_name

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Note

The version of the GRAPH block must be at least V3.0 for the jump into the GRAPH sequencer via the "ActivatePLCCodeView" system function.

Note

Use of the system function "ActivatePLCCodeView" in a VB script has no effect at a GRAPH overview.

Parameters

Screen name

Name of the screen that contains the PLC code view.

Note

Screen change to a pop-up screen or a slide-in screen is not supported.

Screen object

Object name of the PLC code view.

PLCCodeViewMoveScrollBar (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Moves the scroll bar in the PLC code view. This system function provides you with the option of accessing up to six scroll bars in specific drawing areas.

Use in the function list

PLCCodeViewMoveScrollBar (Screen object, Area, Direction)

Use in user-defined functions

-

Parameters

Screen object

Name of the PLC code view.

Area

Area in which the scroll bar is moved:

Drawing area

Symbol table

Detail view

Direction

Direction in which the scroll bar is moved

Left

Right

Top

Bottom

PLCCodeViewDetails (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Shows the detail view in the selected PLC code view.

Use in the function list

PLCCodeViewDetails (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PLC code view.

PLCCodeViewNextNetwork (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Navigates to the next network in the PLC code view.

Use in the function list

PLCCodeViewNextNetwork (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PLC code view.

PLCCodeViewStepMode (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Switches between manual and automatic step selection of the active step in the PLC Code View.

Use in the function list

PLCCodeViewStepMode (screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PLC code view.

PLCCodeViewSymbols (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Displays the symbol area of the network in the PLC code view.

Use in the function list

PLCCodeViewSymbols (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PLC code view.

PLCCodeViewTransitionInterlock (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Toggles between the display of transition and interlock networks in the PLC code view.

Use in the function list

PLCCodeViewTransitionInterlock (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PLC code view.

PLCCodeViewActualOrTriggerValues (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Toggles between the display of actual values and initial values in the PLC code view.

Note

Requirements for initial value acquisition

Initial value acquisition is only possible for the GRAPH function blocks V4.0 and higher.

The "Actual value acquisition" check box must be selected in the properties of the GRAPH block for the initial value acquisition.

Use in the function list

PLCCodeViewActualOrTriggerValues (screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PLC code view.

PLCCodeViewZoomIn (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Zooms in the view of the network in the network area of the PLC Code View.

Use in the function list

PLCCodeViewZoomIn (screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PLC code view.

PLCCodeViewZoomOut (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Zooms out the view of the network in the network area of the PLC code view.

Use in the function list

PLCCodeViewZoomOut (screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PLC code viewer.

PLCCodeViewPreviousNetwork (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Navigates to the previous network in the PLC code view.

Use in the function list

PLCCodeViewPreviousNetwork (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the PLC code view.

Functions of the SmartClient display (RT Advanced, Panels, Comfort Panels)

SmartClientViewRefresh (RT Advanced, Panels, Comfort Panels)

Description

Updates the contents displayed in the Sm@rtClient view.

Use in the function list

SmartClientViewRefresh (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

SmartClientViewReadOnlyOff (RT Advanced, Panels, Comfort Panels)

Description

Sets read-only access to "Off" in the Sm@rtClient view.

This setting allows a distant HMI device to be operated. The "SmartClientViewReadOnlyOn" system function is used to switch read-only access on again.

Use in the function list

SmartClientViewReadOnlyOff (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

SmartClientViewReadOnlyOn (RT Advanced, Panels, Comfort Panels)

Description

Sets read-only access to "On" in the Sm@rtClient view.

This setting allows a distant HMI device to be monitored only. The "SmartClientViewReadOnlyOff" system function is used to switch read-only access off again.

Use in the function list

SmartClientViewReadOnlyOn (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

SmartClientViewDisconnect (RT Advanced, Panels, Comfort Panels)

Description

Executes the "Disconnect" command in the Sm@rtClient view.

This system function is used when the integrated button of the screen object should not be used.

Use in the function list

SmartClientViewDisconnect (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

SmartClientViewConnect (RT Advanced, Panels, Comfort Panels)

Description

Executes the "Connect" command in the Sm@rtClient view.

This system function is used when the integrated button of the screen object should not be used. The Sm@rtClient view establishes a connection with the configured HMI device.

Note

It is not possible to establish an encrypted connection between the Sm@rtClient and the Sm@rtServer using the system function "SmartClientViewConnect" if the server certificate has not been previously accepted by the Sm@rtClient. To establish an encrypted connection between the Sm@rtClient and the Sm@rtServer, use the "Connect" button configured in the Sm@rtClientView in runtime.

Use in the function list

SmartClientViewConnect (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

SmartClientViewLeave (RT Advanced, Panels, Comfort Panels)

Description

Exits the Sm@rtClient view and returns to operation of the HMI device.

The connection to the HMI device configured in the Sm@rtClient view is retained.

Use in the function list

SmartClientViewLeave (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the Sm@rtClient view in which the command is triggered.

Functions of the System Diagnostics view (RT Advanced, Panels, Comfort Panels)

ActivateSystemDiagnosticsView (RT Advanced, Panels, Comfort Panels)

Description

Activates the system diagnostics view. The system diagnostics view shows the detail view of the device concerned.

Use in the function list

ActivateSystemDiagnosticsView (Screen name, Screen object)

Use in user-defined functions

ActivateSystemDiagnosticsView Screen_name, Object_name

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Name of the screen contained in the system diagnostics view.

Screen object

Object name of the system diagnostics view.

SystemDiagnosticViewUpdatePlcBuffer (RT Advanced, Panels, Comfort Panels)

Description

Refreshes the data from the diagnostics buffer of the PLC in the system diagnostics view.

Use in the function list

SystemDiagnosticViewUpdatePlcBuffer (screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the system diagnostics view.

SystemDiagnosticViewDetailView (RT Advanced, Panels, Comfort Panels)

Description

The detail view returns detailed information about the selected connection and errors pending.

Use in the function list

SystemDiagnosticViewDetailView (screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the system diagnostics view that is switched to the detail view.

SystemDiagnosticViewDiagnosticBuffer (RT Advanced, Panels, Comfort Panels)

Description

Opens a view in a system diagnostics for displaying the current data of the diagnostics buffer.

Use in the function list

SystemDiagnosticViewDiagnosticBuffer (screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the system diagnostics view that is switched to the detail view.

SystemDiagnosticViewDeviceView (RT Advanced, Panels, Comfort Panels)

Description

Opens a view in a system diagnostics for displaying all available connections.

Use in the function list

SystemDiagnosticViewDeviceView (screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the system diagnostics view.

SystemDiagnosticViewBack (RT Advanced, Panels, Comfort Panels)

Description

Changes in a system diagnostics to the higher-level view.

- The function opens the device view if the diagnostics buffer view is set in the system diagnostics view.
- The function opens the diagnostics buffer view if the detail view is displayed in the system diagnostics view.

Use in the function list

SystemDiagnosticViewBack (screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the system diagnostics view.

ShowSystemDiagnosticsWindow (Panels, Comfort Panels)

Description

Hides or shows the system diagnostic window on the HMI device. The system diagnostic view is only available in the global screen for Comfort Panels and for WinCC Runtime Advanced.

Use in the function list

ShowSystemDiagnosticsWindow (Screen object)

Use in user-defined functions

ShowSystemDiagnosticsWindow Object_name

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen object

Name of the system diagnostic window which is hidden or shown.

Alarms (RT Advanced, Panels, Comfort Panels)

EditAlarm (RT Advanced, Panels, Comfort Panels)

Description

Triggers the "Edit" event for the selected alarm or for the last selected alarm in case of multiple selection. If the alarms to be edited have not yet been acknowledged, they are acknowledged automatically when this system function is called.

This system function can only be used for function keys.

Use in the function list

EditAlarm

Use in user-defined functions

EditAlarm

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

--

ClearAlarmBuffer (RT Advanced, Panels, Comfort Panels)

Description

Deletes alarms from the alarm buffer on the HMI device.

Note

Alarms which have not yet been acknowledged are also deleted.

Use in the function list

ClearAlarmBuffer (Alarm class number)

Use in user-defined functions

ClearAlarmBuffer Alarm_class_number

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Alarm class number

Determines which alarms are to be deleted from the alarm buffer:

0 (hmiAll) = All alarms

1 (hmiAlarms) = Alarms of alarm class "Error"

2 (hmiEvents) = Alarms of alarm class "Warning"

3 (hmiSystem) = Alarms of alarm class "System event"

4 (hmiS7Diagnosis) = Alarms of alarm class "S7 diagnostic alarm"

Note

Device dependency

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

ClearAlarmBufferProTool (RT Advanced, Panels, Comfort Panels)

Description

The system function exists to ensure compatibility.

It has the same functionality as the "ClearAlarmBuffer" system function, but uses the old ProTool numbering.

Use in the function list

ClearAlarmBufferProTool (Alarm class number)

Use in user-defined functions

ClearAlarmBufferProtoolLegacy Alarm_class_number

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Alarm class number

Alarm class number whose messages are to be deleted:

-1 (hmiAllProtoolLegacy) = All alarms

0 (hmiAlarmsProtoolLegacy) = Alarms of alarm class "Errors"

1 (hmiEventsProtoolLegacy) = Alarms of alarm class "Warnings"

2 (hmiSystemProtoolLegacy) = Alarms of alarm class "System"

3 (hmiS7DiagnosisProtoolLegacy) = Alarms of alarm class "Diagnosis Events"

Note

Device dependency

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

AlarmViewUpdate (RT Advanced, Panels, Comfort Panels)

Description

Updates the enhanced alarm view.

Use in the function list

AlarmViewUpdate (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view which is updated.

AlarmViewLoopInAlarm (RT Advanced, Panels, Comfort Panels)

Description

Triggers the Loop-In-Alarm event for the selected alarm or for the last selected alarm in case of multiple selection.

This system function is used when the integrated button of the ActiveX control should not be used.

You can configure a system function for the "Loop-in-Alarm" event in turn. For example, it is possible to change to the process screen in which the alarm appeared.

Note

If the alarms to be edited have not yet been acknowledged, the acknowledgment takes place automatically when this system function is called up.

Use in the function list

AlarmViewLoopInAlarm (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

AlarmViewAcknowledgeAlarm (RT Advanced, Panels, Comfort Panels)

Description

Acknowledges the alarms selected in the given alarm view.

This system function is used when the integrated button of the ActiveX control should not be used.

Use in the function list

AlarmViewAcknowledgeAlarm (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

AlarmViewShowOperatorNotes (RT Advanced, Panels, Comfort Panels)

Description

Displays the configured tooltip of the alarm selected in the given alarm view.

Use in the function list

AlarmViewShowOperatorNotes (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

See also

System functions for Runtime Advanced (Page 1498)

AcknowledgeAlarm (RT Advanced, Panels, Comfort Panels)

Description

Acknowledges all selected alarms.

This system function is used when the HMI device does not have an ACK key or when the integrated key of the alarm view should not be used.

This system function can only be used for function keys.

Use in the function list

AcknowledgeAlarm

Use in user-defined functions

AcknowledgeAlarm

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

ShowAlarmWindow (RT Advanced, Panels, Comfort Panels)

Description

Hides or shows the alarm window on the HMI device.

Use in the function list

ShowAlarmWindow (Object name, Display mode)

Use in user-defined functions

ShowAlarmWindow Object_name, Display_mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Object name

Name of the alarm view which is hidden or shown.

Display mode

Determines whether the alarm window is hidden or shown:

0 (hmiOff) = Off: Alarm view is hidden

1 (hmiOn) = On: Alarm view is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

ShowSystemAlarm (RT Advanced, Panels, Comfort Panels)

Description

Displays the value of the parameter passed as a system event to the HMI device.

Use in the function list

ShowSystemAlarm (Text/value)

Use in user-defined functions

ShowSystemAlarm Text_or_value

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Text/Value

The text or the value, which was output as a system alarm.

Recipes (RT Advanced, Basic Panels, Panels, Comfort Panels)

ExportDataRecords (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Exports one or all data records of a recipe to a CSV file or a TXT file.

One file is created per recipe.

Use in the function list

ExportDataRecords (Recipe number/name, Data record number/name, File name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ExportDataRecords Recipe_number/name, Data_record number/name, File_name, Overwrite, Output_status_message, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Recipe number/name

Number or name of the recipe from which the data records are exported. Specify "0" if recipe data records are to be exported from all available recipes.

Data record number/name

Number or name of the recipe data record to be exported. Specify "0" if all recipe data records are to be exported.

File name

Name of the file to which the recipe data records are exported. Enter the name, including the storage location and the file extension (*.csv or *.txt), for example "C:\TEMP\Orange.csv". If a folder does not exist, it is created during the export.

If you do not enter the file name fully, the response depends on the number of configured recipes:

- If multiple recipes are configured and
  you only specify one file name and no path, the file is saved in a system directory, for example, "C:\Documents and Settings\[User]".
  When only a path and no file name is specified, the file name is automatically created from the recipe name. It is a requirement that the folder "D:\Data\", for example, has been created in the specified folder. If the folder "D:\Temp" has not been created, the folder name will be used as the prefix of the file name, Temp_Recipe names.csv.
- If only one recipe is configured and only one path and no file name is specified,
  a file with the folder name is created if the folder does not exist. However, this has no file extension.
  the export is aborted with an error message if the folder exists.

If a memory card is used as storage location, specify the storage location as follows: "\StorageCard\<Name>".

For Basic Panel, enter the file name as follows: "\USB_X60.1\<Name>".

Overwrite

Determines whether an existing export file with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: Export file will not be overwritten. The export process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Export is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithPrompting) = With confirmation: Export file is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the export:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Export format of the recipe data records

If ".csv" is chosen as a file extension for the export file, then only valid characters from the ANSI character set will be supported. This also applies to separators in decimal numbers and list elements. The separators used are defined in the country/regional settings of the operating system of the exporting computer.

You may also set "Unicode text" (".txt") format for the export file. This file format supports the WinCC and WinCC Runtime character set. Again, the separators used are specified in the country/regional settings of the operating system of the exporting computer. This file format always uses tab separators in list elements.

The corresponding file import function also supports the ".csv" and ".txt" (Unicode) file formats.

Application example

You want to export all data records using a key.

Notes on configuring

Configure the "ExportDataRecords" system function on the "Press" event for the desired key. Transfer the following parameters:

- Recipe number/name = 1
- Data record number/name = 0
- File name = c:\temp\orange.csv (for Basic Panels "\USB_X60.1\orange.csv")
- Overwrite = 1
- Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data records are exported.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated. All data records of Recipe 1 are exported to the orange.csv file. If the file already exists, it will be overwritten.

After exporting the data records, a system alarm is output.

ExportDataRecordsWithChecksum (RT Advanced, Panels, Comfort Panels)

Description

Exports one or all data records of a recipe to a CSV file and generates a checksum for each line in the file.

One file is created per recipe.

Use in the function list

ExportDataRecordsWithChecksum (Recipe number/name, Data record number/name, File name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ExportDataRecordsWithChecksum Recipe_number_or_name, Data_record number_or_name, File_name, Overwrite, Output_status_message, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which the data records are exported. Specify "0" if recipe data records are to be exported from all available recipes.

Data record number/name

Number or name of the recipe data record to be exported. Specify "0" if all recipe data records are to be exported.

File name

Name of the CSV file to which the recipe data records are exported. Enter the path and the file extension e.g. "C:\TEMP\Orange.CSV".

If a storage card is used as storage location, specify the storage location as follows: "\StorageCard\<FileName>".

If you define only a file name without specifying a path, the file is saved to the directory from which Runtime was started. If write access to this directory is not enabled in the Windows 7 operating system, the file is saved to the "VirtualStore" folder of the user directory.

When only one path for the export is specified, the file name is automatically created from the respective recipe name. This requires, for example, that the directory "D:\Temp\" has been created. If the directory "D:\Temp" does not exist, the directory name is used as the prefix for the file name, Temp_Recipe names.csv.

Overwrite

Determines whether an existing CSV file with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: CSV file will not be overwritten. The export process will not be carried out.

1 (hmiOverwriteAlways) = Yes: CSV file is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithConfirmation) = With confirmation: CSV file is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the export:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Export format of the recipe data records

If ".csv" is chosen as a file extension for the export file, then only valid characters from the ANSI character set will be supported. This also applies to separators in decimal numbers and list elements. The separators used are defined in the country/regional settings of the operating system of the exporting computer.

You may also set "Unicode text" (".txt") format for the export file. This file format supports the WinCC and WinCC Runtime character set. Again, the separators used are specified in the country/regional settings of the operating system of the exporting computer. This file format always uses tab separators in list elements.

The corresponding file import function also supports the ".csv" and ".txt" (Unicode) file formats.

Application example

Using a key, you want to export all data records and assign a checksum.

Notes on configuring

Configure the "ExportDataRecordsWithChecksum" system function on the "Press" event for the desired key. Transfer the following parameters:

- Recipe number/name = 1
- Data record number/name = 0
- File name = c:\temp\orange.csv
- Overwrite = 1
- Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data records are exported.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated. All data records of Recipe 1 are exported to the orange.csv file and assigned checksums. If the file already exists, it will be overwritten.

After exporting the data records, a system message is output.

ImportDataRecords (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Imports one or all data records of a recipe from a CSV file or a TXT file.

When a path is specified, all records of the given directory are imported.

Use in the function list

ImportDataRecords (File name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ImportDataRecords File_name, Data_record_number_or_name, Overwrite, Output_status_message, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

File name

Name of the file from which the recipe data records are imported. Enter the storage location and the file extension (*.CSV or *.TXT), for example "C:\TEMP\Orange.csv".

For Basic Panel, enter the file name as follows: "\USB_X60.1\<Name>"

For additional devices: If a memory card is used as storage location, specify the storage location as follows: "\StorageCard\<Name>"

Data record number/name

Number or name of the recipe data record to be imported. Specify "0" if all recipe data records are to be imported.

Overwrite

Determines whether existing recipe data records are to be overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data records are not overwritten. The import process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data records are overwritten without prompting.

2 (hmiOverwriteWithPrompting) = With confirmation: Recipe data records are not overwritten until confirmed.

Output status message

Determines whether a status message is output after the import:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Configurable objects

Object	Event
Tag	Value change Upper limit exceeded Lower limit violated
Function key (global)	Release Press
Function key (local)	Release Press
Screen	Loaded Cleared
Screen object	Press Release Click Change (or toggle for switch) Switch on Switch off Enable Disable
Scheduler	Time expired

ImportDataRecordsWithChecksum (RT Advanced, Panels, Comfort Panels)

Description

Imports one or all data records of a recipe from a CSV file with a checksum and verifies the checksum.

Use in the function list

ImportDataRecordsWithChecksum (File name, Data record number/name, Overwrite, Output status message, Processing status, verify checksum)

Use in user-defined functions

ImportDataRecordsWithChecksum File_name, Data_record_number_or_name, Overwrite, Output_status_message, Processing_status, Verify_checksum

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

File name

Name of the CSV file from which the recipe data records are imported. Enter the path and the file extension, for example, "C:\TEMP\Orange.CSV".

If a storage card is used as storage medium, specify the storage location as follows: "\StorageCard\<FileName>".

If you specify a directory instead of an individual CSV file, all files in the specified directory will be imported.

Data record number/name

Number or name of the recipe data record to be imported. Specify "0" if all recipe data records are to be imported.

Overwrite

Determines whether existing recipe data records are to be overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data records are not overwritten. The import process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data records are overwritten without prompting.

2 (hmiOverwriteWithConfirmation) = With confirmation: Recipe data records are not overwritten until confirmed.

Output status message

Determines whether a status message is output after the import:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example to delay execution of other system functions, until this system function has been successfully completed.

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Verify checksum

Determines if the checksum should be verified during import:

0 (hmiFalse) = No: Checksum is not verified.

1 (hmiTrue) = Yes: Checksum is verified.

LoadDataRecord (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Loads the given recipe data record from the memory medium of the HMI device in the recipe tags. This system function is used, for example, to display a recipe data record in the recipe screen.

Activate the "Synchronize recipe view and recipe tags" option in the synchronization settings of the recipe. If this option is deactivated, the system function has no effect.

Use in the function list

LoadDataRecord (Recipe number/name, Data record number/name, Processing status)

Use in user-defined functions

LoadDataRecord Recipe_number_or_name, Data_record_number_or_name, Confirmation, Output_status_message, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which a recipe data record is loaded.

Data record number/name

Number or name of the recipe data record to be loaded.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

GetDataRecordFromPLC (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Transfers the selected recipe data record from the PLC to the storage medium of the HMI device.

Use in the function list

GetDataRecordFromPLC (Recipe number/name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

GetDataRecordFromPLC Recipe_number_or_name, Data_record_number_or_name, Overwrite, Output_status_message, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data records are transferred.

Data record number/name

Number or name of the recipe data record which is transferred from the PLC to the data medium of the HMI device.

Overwrite

Determines whether an existing recipe data record with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data record is not overwritten. The transfer process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data record is overwritten without prompting.

2 (hmiOverwriteWithPrompting) = With confirmation: Recipe data record is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the transfer:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Application example

You want to transfer a data record from the PLC to the data medium of the HMI device using a key.

Notes on configuring

Configure the "GetDataRecordFromPLC" system function on the "Press" event for the desired key. Transfer the following parameters:

Recipe number/name = 1

Data record number/name = 1

Overwrite = 1

Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data record is transferred from the PLC.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated and the first data record of Recipe 1 is transferred from the PLC to the data medium of the HMI device. If the recipe data record already exists, it will be overwritten.

A system message is output after the transfer.

GetDataRecordName (RT Advanced, Panels, Comfort Panels)

Description

Writes the name of the given recipe and recipe data record to the given tags.

Note

If the recipe or the recipe data record do not exist, wildcards ("###") are written to the tags.

Note

Only internal tags or external tags are supported as tags.

Use in the function list

GetDataRecordName (Recipe number, Data record number, Recipe name, Data record name, Processing status)

Use in user-defined functions

GetDataRecordName Recipe_number, Data_record_number, Recipe_name, Data_record_name, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number

Number of the recipe whose name is written to the given tag.

Data record number

Number of the recipe data record whose name is written to the given tag.

Recipe name

The tag to which the recipe name is written. The tag must be of the STRING type.

Data record name

The tag to which the name of the recipe data record is written. The tag must be of the STRING type.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Application example

You want to output the names of the displayed recipes and the name of the displayed recipe data record on the HMI device.

Configure the following tags:

- RecNumber" of type INTEGER
- RecDataNumber" of type INTEGER
- RecName" of type STRING
- RecDataName" of type STRING

Configure a recipe view with the tags "RecNumber" for the recipe number and "RecDataNumber" for the data record number.

Configure the "GetDataRecordName" system function on the "Press" event for a button and pass the following parameters:

- Recipe number: RecNumber
- Data record number: RecDataNumber
- Recipe name: RecName
- Data record name: RecDataName

Configure two output fields and connect these to the "RecName" and "RecDataName" tags.

Select a recipe and a data record number from the recipe view. As soon as the button is activated, the system function is triggered and the name of the recipe and the recipe data record are displayed in both output fields.

GetDataRecordTagsFromPLC (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Transfers the values of the recipe data record loaded in the PLC to the corresponding recipe tags.

This system function is used, for example, during teach-in mode on a machine.

Use in the function list

GetDataRecordTagsFromPLC (Recipe number/name, Processing status)

Use in user-defined functions

GetDataRecordTagsFromPLC Recipe_number_or_name, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe data record whose values are written from the PLC to the tags.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

ClearDataRecord (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Deletes a recipe data record.

Several data records can be deleted from one or more recipes.

Use in the function list

ClearDataRecord (Recipe number/name, Data record number/name, Confirmation, Output status message, Processing status)

Use in user-defined functions

ClearDataRecord

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Recipe number/name

Number or name of the recipe from which recipe data records are deleted. "0" is specified if you want to delete recipe data records from all available recipes.

Data record number/name

Number or name of the recipe data record to be deleted. "0" is specified if you want to delete all recipe data records.

Confirmation

Determines whether the operator should confirm the deletion:

0 (hmiOff) = Off: Deletion is begun without confirmation.

1 (hmiOn) = On: The starting of the deletion process must be confirmed.

Output status message

Determines whether a status message is output after the deletion:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

ClearDataRecordMemory (RT Advanced, Panels, Comfort Panels)

Description

Deletes all recipes data records from the specified storage medium.

Use in the function list

ClearDataRecordMemory (Storage location, After confirmation, Output status message, Processing status)

Use in user-defined functions

ClearDataRecordMemory Storage_location, Confirmation, Output_status_message, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Storage location

Determines the storage location:

0 (hmiFlashMemory) = Flash memory: Internal flash memory of the HMI device

1 (hmiStorageCard) = Storage card

2 (hmiStorageCard2) = Storage card

3 (hmiStorageCard3) = Storage card MultiMediaCard

4 (hmiStorageCard4) = Storage card

With confirmation

Determines whether the operator should confirm the deletion:

0 (hmiOff) = Off: Deletion is begun without confirmation.

1 (hmiOn) = On: The starting of the deletion process must be confirmed.

Output status message

Determines whether a status message is output after the deletion:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

RecipeViewNewDataRecord (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Creates a new data record in the given recipe view.

Use in the function list

RecipeViewNewDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the new recipe data record is created.

RecipeViewGetDataRecordFromPLC (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Transfers the data record that is currently loaded in the PLC to the HMI device and displays it in the recipe view.

Use in the function list

RecipeViewGetDataRecordFromPLC (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record from the PLC is displayed.

RecipeViewClearDataRecord (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Deletes the data record which is currently displayed in the recipe view.

Use in the function list

RecipeViewClearDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the displayed recipe data record is deleted.

RecipeViewMenu (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Opens the menu of the specified simple recipe view.

Only use this system function at a simple recipe view.

Use in the function list

RecipeViewMenu (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the menu is to be opened.

RecipeViewOpen (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Displays the data record values in the given recipe view or changes to the next selection field. The system function has no effect if the selection field for the recipe data record values is displayed on the HMI device.

Operation sequence of the selection lists in runtime:

- Recipe name
- Data record name
- RecipeDataRecordValues

This system function is used when a simple recipe view has been configured. In the simple recipe view, only one selection list is displayed at a time on the HMI device. Use the "RecipeViewBack" system function to display the previous selection list.

Use in the function list

RecipeViewOpen (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the command is triggered.

See also

RecipeViewBack (Page 1643)

RecipeViewSetDataRecordToPLC (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Transfers the recipe data record which is currently displayed in the recipe view to the PLC.

Use in the function list

RecipeViewSetDataRecordToPLC (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view from which the recipe data record is transferred to the connected PLC.

RecipeViewSaveDataRecord (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Saves the recipe data record which is currently displayed in the recipe view.

Use in the function list

RecipeViewSaveDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record is saved.

RecipeViewSaveAsDataRecord (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Saves the data record currently being displayed in the recipe view under a new name.

Use in the function list

RecipeViewSaveAsDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Saves the data record currently being displayed in the recipe view under a new name and/or new number.

RecipeViewSynchronizeDataRecordWithTags (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Synchronizes the values of the data record which is currently displayed in the recipe view with their recipe tags. Only use this system function at an advanced recipe view.

During synchronization, all values of the data record are written to their recipe tags.

Use in the function list

RecipeViewSynchronizeDataRecordWithTags (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the values are synchronized with their tags.

RecipeViewRenameDataRecord (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Renames the selected data record in the given recipe view.

Use in the function list

RecipeViewRenameDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record is renamed.

RecipeViewShowOperatorNotes (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Displays the configured tooltip of the specified recipe view.

Use in the function list

RecipeViewShowOperatorNotes (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view whose configured tooltip is displayed.

RecipeViewBack (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Returns to the previous selection list in the given recipe view.

The system function has no effect if the selection field for the recipe is displayed on the HMI device. Operation sequence of the selection lists in runtime:

- Recipe name
- Data record name
- RecipeDataRecordValues

This system function is used when a simple recipe view has been configured. In the simple recipe view, only one selection list is displayed at a time on the HMI device. Use the "RecipeViewOpen" system function to display the recipe data record values or the next selection field.

Use in the function list

RecipeViewBack (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the command is triggered.

SetDataRecordToPLC (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Transfers the given recipe data record directly from the data medium of the HMI device to the PLC with which the HMI device is connected.

Note

The values of the recipe data record don't need to be displayed on the HMI device.

Use in the function list

SetDataRecordToPLC (Recipe number/name, Data record number/name, Output status message, Processing status)

Use in user-defined functions

SetDataRecordToPLC Recipe_number_or_name, Data_record_number_or_name, Output_status_message, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data record is transferred to the PLC.

Data record number/name

Number or name of the recipe data record to be transferred to the PLC.

Output status message

Determines whether a status message is output after the transfer:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

SetDataRecordTagsToPLC (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Transfers the values of the recipe tags to the PLC. The recipe tags contain the values of the data record which is displayed on the HMI device.

Use in the function list

SetDataRecordTagsToPLC (Recipe number/name, Processing status)

Use in user-defined functions

SetDataRecordTagsToPLC Recipe_ number_or_name, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data record is transferred to the PLC.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

SetRecipeTags (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Changes the status of the recipe tags from "Online" to "Offline" and vice versa.

This system function is used, for example, when recipe data record values are fine tuned when starting up a machine.

Use in the function list

SetRecipeTags (Recipe number/name, Status, Output status message, Processing status)

Use in user-defined functions

SetRecipeTags Recipe_number_or_name, Status, Output_status_message, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe in which the recipe data record is saved.

Status

Determines the status of the recipe tags:

0 (hmiOnline) = online: Value changes of the recipe tags are transferred immediately to the PLC connected to the HMI device.

1 (hmiOffline) = offline: Value changes to the recipe tags are only transferred to the PLC connected to the HMI device when, for example, the "SetDataRecordTagsToPLC" system function is executed.

Output status message

Determines whether a status message is output after saving:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

SaveDataRecord (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Saves the current values of the recipe tags as data record to the memory medium of the HMI device.

This system function is used, for example, to save a recipe data record in the recipe screen.

Use in the function list

SaveDataRecord (Recipe number/name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

SaveDataRecord Recipe_number_or_name, Data_record_number_or_name, Overwrite, Output_status_message, Processing_status

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Recipe number/name

Number or name of the recipe to which a recipe data record is saved.

Data record number/name

Number or name of the recipe data record to be saved. A new data record will be created if no record of this name or number was found in the recipe, independent of the value at the "Overwrite" parameter.

Overwrite

Specifies whether an existing data record is overwritten:

0 (hmiOverwriteForbidden) = No: The recipe data record is not overwritten, the data record is not saved.

1 (hmiOverwriteAlways) = Yes: The recipe data record is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithConfirmation) = With confirmation: The recipe data record is overwritten only with confirmation by the user.

Output status message

Determines whether a status message is output after saving:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

System (RT Advanced, Basic Panels, Panels, Comfort Panels)

ActivateCleanScreen (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Activates the clean screen on the HMI device. The display of the HMI device is disabled for the given time period.

When the display of the HMI device is deactivated, it can be cleaned without triggering touch functions by mistake.

Use in the function list

ActivateCleanScreen (Time period)

Use in user-defined functions

--

Parameters

Time period

Time period for which the display is disabled. The time remaining is displayed as a progress bar.

Value range in seconds from 10 through 300.

Note

The system function ActivateCleanScreen cannot be simulated.

See also

Logoff (Page 1546)

OpenFileBrowser (RT Advanced, Panels, Comfort Panels)

Description

Opens a file browser dialog.

Depending on the mode the user can select a file or a folder.

Use in the function list

OpenFileBrowser (Mask, Output Folder, File, Navigation Back, Navigation Up, New folder, Path, Status)

Use in user-defined functions

-

Parameters

Mask

Sets a mask for filtering the displayed files. This parameter is only evaluated in "File" mode. The filter criteria must be delimited by ; semicolon. Example: *.htm;*.html

Start folder

Specifies the folder with which the file selection dialog starts.

File

Specifies the mode in which the dialog is opened.

0 = folder search mode

1 = file search mode

Navigation Back

Specifies whether or not the "Back" button is activated. This button offers simple navigation to the last folder you opened.

Navigation Up

Specifies whether or not the "Up" button is activated. This button offers simple navigation to the parent folder.

New folder

Specifies whether or not the button can be used to create a new folder.

Path

Contains the path name of the last folder opened after you close the dialog.

Status

Returns information about the dialog closing mode

-1 = dialog was canceled

0 = dialog was closed with "OK".

OpenTaskManager (RT Advanced, Panels, Comfort Panels)

Description

Shows the task manager.

The task manager allows changing to other open applications on the HMI device.

Note

The appearance of the task manager depends on the operating system installed.

Use in the function list

OpenTaskManager

Use in user-defined functions

OpenTaskManager

Can be used if the configured device supports user-defined scripts. For additional information, refer to "Device dependency".

Parameter

--

Keyboard (RT Advanced, Panels, Comfort Panels)

OpenScreenKeyboard (RT Advanced, Panels, Comfort Panels)

Description

Hides or shows the screen keyboard.

The screen keyboard remains open until the screen keyboard is explicitly closed. In this way, the screen keyboard can also be used in other applications.

Use in the function list

OpenScreenKeyboard (Display mode)

Use in user-defined functions

OpenScreenKeyboard Display_mode

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Specifies whether the window is opened minimized or maximized with the screen keyboard:

0 (hmiScreenKeyboardMinimized) = Minimized

1 (hmiScreenKeyboardMaximized) = Maximized

Keyboard operation for screen objects (RT Advanced, Basic Panels, Panels, Comfort Panels)

ScreenObjectCursorDown (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Results in a line-by-line, downward cursor movement in the table of the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view
- NC subprogram display
- PLC code view

Use in the function list

ScreenObjectCursorDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

ScreenObjectCursorUp (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Results in a line-by-line, upward cursor movement in the table of the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view
- NC subprogram display
- PLC code view

Use in the function list

ScreenObjectCursorUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

ScreenObjectCursorRight (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Results in a line-by-line, cursor movement to the right in the specified screen object.

Use in the function list

ScreenObjectCursorLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the command is executed.

ScreenObjectPageDown (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Results in a page-by-page, downward cursor movement in the table of the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view
- NC subprogram display

Use in the function list

ScreenObjectPageDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

ScreenObjectPageUp (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Results in a page-by-page, upward cursor movement in the table of the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view
- NC subprogram display

Use in the function list

ScreenObjectPageUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

ScreenObjectCursorLeft (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Results in a line-by-line, cursor movement to the left in the specified screen object.

Use in the function list

ScreenObjectCursorLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the command is executed.

PageDown (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Executes the key function <Pagedown> on the HMI device.

This system function can only be used for function keys.

Use in the function list

PageDown

Use in user-defined functions

PageDown

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

-

PageUp (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Executes the key function <PageUp> on the HMI device.

This system function can only be used for function keys and tasks with a time trigger.

Use in the function list

PageUp

Use in user-defined functions

PageUp

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

-

GoToHome (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Executes the key function <Home> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToHome

Use in user-defined functions

GoToHome

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

GoToEnd (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Executes the key function <End> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToEnd

Use in user-defined functions

GoToEnd

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

9.11.1.3 System functions (RT Professional)

Other functions (RT Professional)

StopRuntime (RT Professional)

Description

Exits the runtime software and thereby the project running on the HMI device.

Use in the function list

StopRuntime (Mode)

Use in user-defined functions

StopRuntime (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Mode

Determines whether the operating system is shut down after exiting runtime.

0 (hmiStopRuntime) = Runtime: Operating system is not shut down

1 (hmiStopRuntimeAndOperatingSystem) = Runtime and operating system: The operating system is shut down (not possible with WinCE)

Example

The following program code shuts down Runtime and the operating system.

```
{

//Stop runtime and shutdown
StopRuntime (hmiStopRuntimeAndOperationSystem);

}
```

The saved return value can be processed in the following code.

See also

System functions for Runtime Professional (Page 1503)

LookupText (RT Professional)

Description

Identifies an entry from a text list. The result depends on the value and on the selected runtime language. The result is saved to a tag of data type "String".

Use in the function list

LookupText (Output text, Index, Language, Text list)

Use in user-defined functions

LookupText (Output_text, Index, Language, Text_list)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Output text

The tag to which the result is written.

Index

The tag that defines the value of the list entry.

Language

Defines the runtime language in which the list entry is identified.

- Runtime language
  Language code as per VBScript reference, for example, "de-DE" for German (Germany) or "en-US" for English (United States of America). The selection depends on the activated runtime languages.
- Tag
  The tag that contains the language. Enter the runtime language as decimal value of the country ID, for example, 1031 for German - Standard, 1033 for English - USA. Details are available in the VBScript basics, "Locale identifier (LCID) diagram".
- Integer
  The number that corresponds to the order of runtime languages for language switching. The selection depends on the activated runtime languages, for example, "0" for the language that appears when you first start runtime. You can find more information under the topic of "Languages in Runtime".

Text list

Defines the text list. The list entry is read from the text list.

See also

Device dependency (Page 1474)

User administration (RT Professional)

ExportImportUserAdministration (RT Professional)

Description

Exports all users of the user administration of the currently active project to the given file or imports the users from the given file into the currently active project.

Users, their passwords and rights are saved in the user administration.

All users are overwritten when importing. The imported users are valid immediately.

Note

Exporting and importing of the user administration from V14 is not supported on HMI devices with a Runtime version older than V14.

Use in the function list

ExportImportUserAdministration (File name, Direction)

Use in user-defined functions

ExportImportUserAdministration (File_name, Direction)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

File name

Name of the file which contains the passwords or to which the passwords are written. Enter the file location and the file extension (*.txt), for example, "C:\TEMP\Passwords.txt".

Note

If a storage card is used as file location, specify the file location as follows: "\StorageCard \<FileName".

Direction

Specifies whether passwords are exported or imported:

0 (hmiExport) = Export: Passwords are exported.

1 (hmiImport) = Import: Passwords are imported.

ShowLogonDialog (RT Professional)

Description

Opens a dialog on the HMI device with which the user can log on to the HMI device.

Use in the function list

ShowLogonDialog

Use in user-defined functions

ShowLogonDialog

Can be used if the configured device supports user-defined functions. Additional information is available under "System functions for Runtime Professional (Page 1503)".

Parameters

--

See also

System functions for Runtime Professional (Page 1503)

Calculation (RT Professional)

IncreaseTag (RT Professional)

Description

Adds the given value to the value of the tags.

X = X + a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function for events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual tag value is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

IncreaseTag (Tag, Value)

Use in user-defined functions

IncreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

Parameters

Tag

The tag to which the given value is added.

Value

The value that is added.

Example

The following program code increments the value of the varX tag by the value in the value tag. The value entered is saved to the old_value tag and output along with the new varX value.

```
{
BYTE varX;
BYTE value;

//user input
...
BYTE old_value = varX;

//Increase tag
IncreaseTag(varX, value);

//print original value and function result
printf ("User input: %i\r\n, Result of function IncreaseTag: %i\r\n", old_value, varX);
...
}
```

See also

System functions for Runtime Professional (Page 1503)

InvertLinearScaling (RT Professional)

Description

Assigns a value to the tag X, which is calculated from the value of the given tag Y using the linear function X = (Y - b) / a.

The tags X and Y must not be identical. This system function is the inverse of the "LinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

InvertLinearScaling (X, Y, b, a)

Use in user-defined functions

InvertLinearScaling (X, Y, b, a)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

X

The tag which is assigned the value calculated from the linear equation.

Y

The tag that contains the value used for calculation.

b

The value which is subtracted.

a

The value through which is divided.

Example

The following program code assigns a value to the varX tag by means of the InverseLinearScaling function.

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
 InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n, varX);
...
}
```

The saved return value can be processed in the following code.

See also

System functions for Runtime Professional (Page 1503)

LinearScaling (RT Professional)

Description

Assigns a value to the tag Y, which is calculated from the value of the given tag X using the linear function Y= (a *X) + b.

The inverse of this function is the "InvertLinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

LinearScaling (Y, a, X, b)

Use in user-defined functions

LinearScaling (Y, a, X, b)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Y

The tag which is assigned the value calculated from the linear equation.

a

The value with which is multiplied.

X

The tag that contains the value used for calculation.

b

The value that is added.

Example

The following program code uses the LinearScaling function to assign a value to the Yvar tag.

```
{
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
 LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n, Yvar);
...
}
```

The saved return value can be processed in the following code.

See also

System functions for Runtime Professional (Page 1503)

SetTag (RT Professional)

Description

Assigns a new value to the given tag.

Note

This system function can be used to assign strings and numbers, depending on the type of tag.

Use in the function list

SetTag (Tag, Value)

Use in user-defined functions

SetTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag to which the given value is assigned.

Value

The value which the given tag is assigned.

Note

The "SetTag" system function is only executed after a connection has been established.

Example

The following program code uses the SetTag function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag.

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
bvalue = GetTagBit("gs_tag_bit");
  printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

The saved return value can be processed in the following code.

See also

System functions for Runtime Professional (Page 1503)

SetTagByProperty (RT Professional)

Description

Specifies a tag value with the value of an object property. The change is also logged in the alarm system.

Use in the function list

SetTagIndirectByProperty (tag name, screen name, screen object, name of the property, with or without operator event)

Use in user-defined functions

SetTagByProperty (Tag_name, Screen_name, Screen_object, Property_name, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

Parameters

Tag name

Name of the tag whose value is specified by the object property.

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property supplies the tag value.

Name of the property

Name of the property that supplies the tag value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator event

1 (hmiWithOperatorEvent) = With operator event

Example

The following program code returns the value of the selected text when you click in a combo box.

```
{
char* rt_value;

SetTagByProperty (rt_value, screenName, objectName, "SelectedText",
hmiWithoutOperatorEvent);

...
}
```

See also

System functions for Runtime Professional (Page 1503)

SetTagIndirect (RT Professional)

Description

Writes a value to an indirect addressed tag. The tag transferred as output parameter contains the name of a second tag the value of which is changed by the function. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirect (tag name, , value, with or without operator event)

Use in user-defined functions

SetTagIndirect (Tag_name, Value, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

Parameters

Tag name

Name of a string tag which contains the name of a tag whose value is to be changed.

Value

Value which is to be written.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator event

1 (hmiWithOperatorEvent) = With operator event

Example

The following program code writes the value "value" to the tag Tag3.

```
{
int value;

SetTag ("IndirectWrite", "Tag3");
SetTagIndirect ("IndirectWrite", "value", hmiWithoutOperatorEvent);
...
}
```

See also

System functions for Runtime Professional (Page 1503)

SetTagIndirectByProperty (RT Professional)

Description

Writes the value of an object property to an indirectly addressed tag. The tag transferred as output parameter contains the name of a second tag the value of which is changed by the function. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirectByProperty (Tag name, Screen name, Screen object, Name of the property, With or without operator event)

Use in user-defined functions

SetTagIndirectByProperty (Tag_name, Screen_name, Screen_object, Property_name, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

Parameters

Tag name

Name of a string tag which contains the name of a tag whose value is to be changed.

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property supplies the value.

Name of the property

Name of the property which supplies the value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without input alarm

1 (hmiWithOperatorEvent) = With operator input alarm

Example

The following program code writes the value of the object property "Background color" to the tag Tag2.

```
{
SetTag ("IndirectWrite", "Tag2");
SetTagIndirectByProperty ("IndirectWrite", screenName, objectName, "BackColor",
hmiWithoutOperatorEvent);
...
}
```

See also

System functions for Runtime Professional (Page 1503)

SetTagIndirectByTagIndirect (RT Professional)

Description

Writes the value of an indirectly addressed tag to an indirectly addressed tag. The tag transferred as output parameter contains the name of a second tag the value of which is changed by the function. The tag transferred as parameter contains the name of a second tag the value of which is read. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirectByTagIndirect (Tag name, Tag name, With or without operator input alarm)

Use in user-defined functions

SetTagIndirectByTagIndirect Tag_name, Source_tag_name, With_or_without_operator_event

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

Parameters

Tag name

Name of a string tag which contains the name of a tag whose value is to be changed.

Tag name

Name of the indirect tag that returns the tag value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator event

1 (hmiWithOperatorEvent) = With operator event

Example

The following program code writes the value of the tag "Tag4" to the tag "Tag2".

```
{
SetTag ("IndirectWrite", "Tag2");
SetTag ("IndirectRead", "Tag4");
SetTagIndirectByTagIndirect ("IndirectWrite", "IndirectRead");
...
}
```

See also

System functions for Runtime Professional (Page 1503)

SetTagByTagIndirect (RT Professional)

Description

Writes the value of an indirect addressed tag to a tag. The tag transferred as parameter contains the name of a second tag the value of which is read. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirectByTagIndirect (name of the tag, tag name, with or without operator input alarm)

Use in user-defined functions

SetTagByTagIndirect Tag_name, Source_tag_name, With_or_without_operator_event

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

Parameters

Name of the tag

Name of the tag whose value is to be set.

Tag name

Name of a string tag which contains the name of a tag which supplies the tag value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator input alarm

1 (hmiWithOperatorEvent) = With operator input alarm

Example

The following program code writes the value of the tag Tag4 to the tag Tag1.

```
{
SetTag ("IndirectRead", "Tag4");
SetTagByTagIndirect ("Tag1", "IndirectRead", hmiWithoutOperatorEvent);

...
}
```

See also

System functions for Runtime Professional (Page 1503)

SetTagWithOperatorEvent (RT Professional)

Description

Specifies the value for a tag. The change is also logged in the alarm system.

Use in the function list

SetTagWithOperatorEvent (name of the tag, value)

Use in user-defined functions

SetTagWithOperatorEvent Tag_name, Value

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

Parameters

Name of the tag

Name of the tag whose value is to be set.

Value

Value written to the tag.

Example

The following program code transfers the value from the "value" tag to the "result" tag when you click the corresponding button.

```
'Programming language: VBS
SetTagWithOperatorEvent result, value
...
```

```
//Programming language: C
{
SetTagWithOperatorEvent ("result", "value");
...
}
```

See also

System functions for Runtime Professional (Page 1503)

DecreaseTag (RT Professional)

Description

Subtracts the given value from the tag value.

X = X - a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function for events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual tag value is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

DecreaseTag (Tag, Value)

Use in user-defined functions

DecreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

Parameters

Tag

The tag from which the given value is subtracted.

Value

The value which is subtracted.

Example

The following program code decrements the value at the varX tag by the value at the value tag. The value entered is saved to the old_value tag and output along with the new varX value.

```
{
BYTE varX;
BYTE value;

//user input
...
BYTE old_value = varX;

//Decrease tag
DecreaseTag(varX, value);

//print original value and function result
printf ("User input: %i\r\n, Result of function DecreaseTag: %i\r\n", old_value, varX);
...
}
```

See also

System functions for Runtime Professional (Page 1503)

Screens (RT Professional)

ActivateScreen (RT Professional)

Description

Performs a screen change to the given screen.

Use the "ActivateScreenByNumber" system function to change from the root screen to the permanent area or vice versa.

Use in the function list

ActivateScreen (Screen name, Object number)

Use in user-defined functions

ActivateScreen (Screen_name, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Name of the screen to which you change.

Object number

The operator control element which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent area when the system function is called up, the permanent area maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

Note

If the "Reach margin" event is assigned to the "ActivateScreen" system function, only the value "0" is valid for the "Object number" parameter. The active object is not defined by the object number, but rather by the X position it had prior to the screen change.

Example

The following program code activates "Screen_2" with the ActivateScreen function, for example, when you click a button.

```
Sub ActivateScreen_2()

'User-defined code
'' i. e. when pressing a button

ActivateScreen "Screen_2",0
```

See also

System functions for Runtime Professional (Page 1503)

ActivateScreenInCurrentScreenWindow (RT Professional)

Description

Performs a screen change to the specified screen in the current screen window.

Use in the function list

ActivateScreenInCurrentScreenWindow (New screen name)

Use in user-defined functions

-

Parameters

New screen name

Name of the screen to be displayed in the screen window.

See also

System functions for Runtime Professional (Page 1503)

ActivateScreenInScreenWindow (RT Professional)

Description

Performs a screen change to the specified screen in a specified screen window.

Use in the function list

ActivateScreenInScreenWindow (Screen name, Screen window, New screen name)

Use in user-defined functions

ActivateScreenInScreenWindow (Screen_name, Screen_window, New_screen_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

Parameters

Screen name

Name of the screen to be displayed in the screen window.

Screen window

Name of the screen window in which the new screen is to be displayed.

New screen name

Name of the new screen to be displayed in the screen window.

Example

The ActivateScreenInScreenWindow function is used in the following program code to activate the "Screen_2" screen when you click a button.

```
{
// User defined code
// i.e. when pressing a button
ActivateScreenInScreenWindow (GetParentScreen(screenName),
GetParentScreenWindow(screenName), "Screen_2");
...
}
```

See also

System functions for Runtime Professional (Page 1503)

GetParentScreen (RT Professional)

Description

The function determines the name of the parent screen from a specified screen path.

Use in the function list

GetParentScreen (screen)

Use in user-defined functions

GetParentScreen Screen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen

The structure of the transferred "Screen" parameter must correspond to that formed by the graphics system for screen paths:

<Screen_name>.<Screen_window_name>:<Screen_name>.<Screen_window_name>:<Screen_name>...

If you use the object list in the function list to specify the parameter, the screen name is entered instead of the screen path.

Note

The "." character is used to distinguish between screen and screen object names. The ":" character is used for the hierarchy structuring. Therefore, use only the "-" or "_" as a delimiter in the name designations.

Return value

Name of the parent screen.

Principle

A "Screenwindow_1" screen window is in a "Screen_1" screen. In the screen window, the screen becomes "Screen_2" with the "Screenwindow_2" screen window, etc.

Screen path: Screen_1.Screenwindow_1:Screen_2.Screenwindow_2:Screen_3

GetParentScreen returns:

- "Screen_2", if the system function is called in the "Screen_3" screen.
- "Screen_1", if the system function is called in the "Screen_2" screen.

GetParentScreenwindow returns:

- "Screenwindow_2", if the system function is called in the "Screen_3" screen.
- "Screenwindow_1", if the system function is called in the "Screen_2" screen.

GetParentScreenWindow (RT Professional)

Description

The function determines the name of the parent screen window from a transferred screen path.

Use in the function list

GetParentScreenWindow (Screen)

Use in user-defined functions

GetParentScreenWindow Screen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen

The structure of the transferred "Screen" parameter must correspond to that formed by the graphics system for screen paths:

<Screen_name>.<Screen_window_name>:<Screen_name>.<Screen_window_name>:<Screen_name>...

If you use the object list in the function list to specify the parameter, the screen name is entered instead of the screen path.

Note

The "." character is used to distinguish between screen and screen object names. The ":" character is used for the hierarchy structuring. Therefore, use only the "-" or "_" as a delimiter in the name designations.

Return value

Name of the parent screen window.

Principle

A "Screenwindow_1" screen window is in a "Screen_1" screen. In the screen window, the screen becomes "Screen_2" with the "Screenwindow_2" screen window, etc.

Screen path: Screen_1.Screenwindow_1:Screen_2.Screenwindow_2:Screen_3

GetParentScreen returns:

- "Screen_2", if the system function is called in the "Screen_3" screen.
- "Screen_1", if the system function is called in the "Screen_2" screen.

GetParentScreenwindow returns:

- "Screenwindow_2", if the system function is called in the "Screen_3" screen.
- "Screenwindow_1", if the system function is called in the "Screen_2" screen.

SetPropertyByProperty (RT Professional)

Description

Specifies the value of an object property with another object property.

Use in the function list

SetPropertyByProperty (screen name, object, property name, target screen name, target screen object, target property name)

Use in user-defined functions

SetPropertyByProperty Screen_name, Screen_object, Property_name, Source_screen_name, Source_screen_object, Source_property_name

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

If you want to specify the property of a screen with another screen property, the parameters "Object" and "Destination object" must be empty. For this purpose, use the following syntax, for example:

SetPropertyByProperty ("<Screen name>","","<Property name>","<Destination_Screen name>","","<Destination_Property name>")

Parameters

Screen name

Name of the screen that contains the object.

Object

Name of the object whose property value is transferred to the target object.

Property name

Name of the property whose value is transferred to the target object.

Target screen name

Name of the screen that contains the target object.

Target screen object

Name of the target object for which a property is changed.

Target property name

Name of the property that will be changed.

Example

The following program code uses the SetPropertyByProperty function to set the property "ToolbarButtonClick" of the object "Control_1" in the original screen Trend_1": on the corresponding property in the destination screen "Trend_2".

```
'Programming language: VBS
'Name of source picture: Trend_1
'Name of target picture: Trend_2
'Name of the f(t) trend view control: Control_1

SetPropertyByProperty "Trend_1", "Control_1", "ToolbarButtonClick", "Trend_2",
"Control_2", "ToolbarButtonClick"

'User defined code
...
```

```
{
//Programming language: C
//Name of source picture: Trend_1
//Name of target picture: Trend_2
//Name of the f(t) trend view control: Control_1

SetPropertyByProperty ("Trend_1", "Control_1", "ToolbarButtonClick", "Trend_2",
"Control_2", "ToolbarButtonClick");

// User defined code
...
}
```

See also

System functions for Runtime Professional (Page 1503)

SetPropertyByConstant (RT Professional)

Description

Specifies the value of an object property as a string.

Use in the function list

SetPropertyByConstant (Screen name, screen object, name of the property, value)

Use in user-defined functions

SetPropertyByConstant

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

If you want to change the property of a screen, the parameter "Object" must be empty. For this purpose, use the following syntax, for example:

SetPropertyByConstant ("<Screen name>","","<Property name>","<Value>")

Parameters

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property is changed.

Name of the property

Name of the property that will be changed.

Value

The value assigned to the property.

Example

The SetPropertyByConstant function is used in the following program code to change an object property: In the "Trends" screen, the "ToolbarButtonClick" property of the "Control_1" object is set to the value 26.

```
'Programming language: VBS
'Name of the picture: Trends
'Name of the f(t) trend view control: Control_1

SetPropertyByConstant "Trends", "Control_1", "ToolbarButtonClick", "26"

'User defined code
...
```

```
{
//Programming language: C
//Name of the picture: Trends
//Name of the f(t) trend view control: Control_1

SetPropertyByConstant ("Trends", "Control_1", "ToolbarButtonClick", "26");

// User defined code
...
}
```

Example: Changing a screen property

The SetPropertyByConstant function is used in the following program code to change a screen property: In the "Trends" screen, the "Backcolor" property is set to the value 255.

```
'Programming language: VBS
'Name of the picture: Trends

SetPropertyByConstant "Trends", "Trends", "Backcolor", "255"

'User defined code
...
```

```
{
//Programming language: C
//Name of the picture: Trends

SetPropertyByConstant ("Trends", "Trends", "Backcolor", "255");

// User defined code
...
}
```

Alternatively, use the password ZERO or a space-string instead of the second parameter (object).

See also

System functions for Runtime Professional (Page 1503)

SetPropertyByTag (RT Professional)

Description

Specifies the value of an object property with a tag value.

Use in the function list

SetPropertyByTag (Screen name, screen object, name of the property, tag name)

Use in user-defined functions

SetPropertyByTag (Screen_name, Screen_object, Property_name, Tag_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

If you want to specify the property of a screen by means of a tag value, the "Object" parameter must be empty. For this purpose, use the following syntax, for example:

SetPropertyByTag ("<Screen name>","","<Property name>","<Tag name>")

Parameters

Screen name

Name of the screen that contains the object.

Screen object

Name of the object containing the property to be set with the tag value.

Name of the property

Name of the property that is set with the tag value.

Tag name

Name of the tag that contains the value of the property.

Example

The SetPropertyByTag function is used in the following program code to change an object property: When you click on the object, the object name and the screen in which the object is located are transferred. The CaptionText in the screen window contains the value of the HMI_value_1 tag.

```
'Programming language: VBS
SetPropertyByTag screenName, objectName, "CaptionText", "HMI_value_1"

'User defined code
...
```

```
{
//Programming language: C
SetPropertyByTag (screenName, objectName, "CaptionText", "HMI_value_1");

// User defined code
...
}
```

Example

The SetPropertyByTag function is used in the following program code to change an object property: In the "Trends" screen, the "ToolbarButtonClick" property of the "Control_1" object is set to the value 26.

```
'Programming language: VBS
'Name of the picture: Trends
'Name of the f(t) trend view control: Control_1

SetPropertyByConstant "Trends", "Control_1", "ToolbarButtonClick", "26"

'User defined code
...
```

```
{
//Programming language: C
//Name of the picture: Trends
//Name of the f(t) trend view control: Control_1

SetPropertyByConstant ("Trends", "Control_1", "ToolbarButtonClick", "26");

// User defined code
...
}
```

See also

System functions for Runtime Professional (Page 1503)

SetPropertyByTagIndirect (RT Professional)

Description

Writes the value of an indirect addressed tag to an object property. The tag transferred as parameter contains the name of a second tag the value of which is read.

Use in the function list

SetPropertyByTagIndirect (Screen name, screen object, name of the property, , tag name)

Use in user-defined functions

SetPropertyByTagIndirect (Screen_name, Screen_object, Property_name, Tag_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional (Page 1503)".

If you want to specify the property of a screen by means of a tag, the "Object" parameter must be empty. For this purpose, use the following syntax, for example:

SetPropertyByTagIndirect ("<Screen name>","","<Property name>","<Tag name>")

Parameters

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property you want to change.

Name of the property

Name of the property to be changed.

Tag name

Name of the tag which, in turn, contains the name of the tag whose value is read.

Example

The SetPropertyByTagIndirect function is used in the following program code to change an object property: .

```
'Programming language: VBS
SetPropertyByTagIndirect GetParentScreen(screenName), GetParentScreenWindow(screenName),
"ScreenName", "HMI_value_1"

'User defined code
...
```

```
{
//Programming language: C
SetPropertyByTagIndirect (GetParentScreen(screenName), GetParentScreenWindow(screenName),
"ScreenName", "HMI_value_1");

// User defined code
...
}
```

See also

System functions for Runtime Professional (Page 1503)

SetPropertyCurrentWindowByProperty (RT Professional)

Description

Specifies the value of an object property in the current window with another object property.

Use in the function list

SetPropertyCurrentWindowByProperty (Name of the property, Screen name, Screen object, Property name)

Use in user-defined functions

-

Parameters

Name of the property

Name of the property whose property is to be transferred to the destination object.

Screen name

Name of the screen that contains the screen object.

Screen object

Name of the screen object to which the property is transferred.

Name of the property

Name of the property that will be changed.

See also

System functions for Runtime Professional (Page 1503)

SetPropertyCurrentWindow (RT Professional)

Description

Specifies the value of an object property in the current window with a tag value.

Use in the function list

SetPropertyCurrentWindow (Name of the property, value)

Use in user-defined functions

-

Parameters

Name of the property

Name of the property that is set with the tag value.

Value

Name of the tag that contains the value of the property.

See also

System functions for Runtime Professional (Page 1503)

SetPropertyCurrentWindowByTagIndirect (RT Professional)

Description

Specifies the value of an object property in the current window with a tag. The tag contains the tag name that specifies the object property.

Use in the function list

SetPropertyCurrentWindowByTagIndirect (Name of target property, tag name)

Use in user-defined functions

-

Parameters

Name of the target property

Name of the property that is set with the tag value.

Tag name

Name of the tag that, in turn, contains the name of the tag that specifies the object property.

See also

System functions for Runtime Professional (Page 1503)

Bit processing (RT Professional)

InvertBit (RT Professional)

Description

Inverts the value of the given tag of the "Bool" type:

- If the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

Use in the function list

InvertBit (Tag)

Use in user-defined functions

InvertBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag whose bit is set.

Example

The following program code inverts the value of the Boolean tag bStatus and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit
InvertBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Invert variable
invertBit(bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
  ...
}
```

See also

System functions for Runtime Professional (Page 1503)

InvertBitInTag (RT Professional)

Description

Inverts a bit in the given tag:

- If the bit in the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the bit in the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "InvertBit" system function instead.

Use in the function list

InvertBitInTag (Tag, Bit)

Use in user-defined functions

InvertBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which the given bit is set.

Bit

The number of the bit that is set.

When this system function is used in a user-defined function, the bits in a tag are counted from right to left. The counting begins with 0.

Example

The following program code inverts a bit at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit in position
bitposition=2
InvertBit(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BYTE bStatusWord;
BYTE bsaved = bStatusWord;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bStatusWord, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bsaved);
  ...
}
```

See also

System functions for Runtime Professional (Page 1503)

ResetBit (RT Professional)

Description

Sets the value of a "Bool" type tag to 0 (FALSE).

Use in the function list

ResetBit (Tag)

Use in user-defined functions

ResetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 0 (FALSE).

Example

The following program code sets the value of the Boolean tag bStatus to 0 by means of the ResetBit function and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=1
myTag.Value=bValue

'Save current value
bSaved=bValue

'Reset Bit
ResetBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 1;
BOOL bSaved = bStatus;

//Reset bit
  ResetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

See also

System functions for Runtime Professional (Page 1503)

ResetBitInTag (RT Professional)

Description

Sets a bit in the specified tag to 0 (FALSE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "ResetBit" system function instead.

Use in the function list

ResetBitInTag (Tag, Bit)

Use in user-defined functions

ResetBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 0 (FALSE).

Bit

The number of the bit that is set to 0 (FALSE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Example

The following program code sets a bit to 0 at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Reset Bit
bitposition=2
ResetBit(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BYTE bStatusWord;
BYTE bSaved = bStatusWord;
BYTE bitposition = 2;

//Reset bit in bitposition
ResetBitInTag (bStatusWord, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bSaved);
  ...
}
```

See also

System functions for Runtime Professional (Page 1503)

SetBit (RT Professional)

Description

Sets the value of a "Bool" type tag to 1 (TRUE).

Use in the function list

SetBit (Tag)

Use in user-defined functions

SetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 1 (TRUE).

Example

The following program code sets the value of the Boolean tag bStatus to 1 by means of the SetBit function and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=0
myTag.Value=bValue

'Save current value
bSaved=bValue

'Set Bit
SetBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Set bit
  SetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

See also

System functions for Runtime Professional (Page 1503)

SetBitInTag (RT Professional)

Description

Sets a bit in the given tag to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitInTag (Tag, Bit)

Use in user-defined functions

SetBitInTag(Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 1 (TRUE).

Bit

The number of the bit that is set to 1 (TRUE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an I/O field or assign the system function to a screen object, such as a button.

If you have configured a short event such as the activation of an alarm for the system function you can only access the actual process values by setting the tag for continuous reading.

Example

The following program code sets a bit to 1 at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Set Bit in tag
bitposition=2
SetBitInTag(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BYTE bStatusWord;
BYTE bSaved = bStatusWord;
BYTE bitposition = 2;

//Reset bit in bitposition
SetBitInTag (bStatusWord, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bSaved);
  ...
}
```

See also

System functions for Runtime Professional (Page 1503)

Settings (RT Professional)

GetPLCMode

Description

Evaluates the current state of the connected PLC.

The system function "GetPLCMode" can only be configured for the following PLCs:

- SIMATIC S7-1200
- SIMATIC S7-1500

Use in the function list

GetPLCMode (Connection, Mode)

Use in user-defined functions

-

Parameters

Connection

Connection of PLC and HMI device.

Mode

Evaluates the state of the connected PLC.
8: RUN = The PLC program is being executed.
4: STOP = The PLC program has been interrupted.
Select a suitable user tag for the evaluation.

SetLanguage

Description

Toggles the language on the HMI device. All configured text and system events are displayed on the HMI device in the newly set language.

Use in the function list

SetLanguage (Language)

Use in user-defined functions

SetLanguage (Language)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Language

Determines which language is set on the HMI device. The following specifications are possible:

- -1 (hmiToggle) = Toggle: Changes to the next language. The sequence is determined during configuration in the "Project languages" editor.
- Number you have defined under "Languages and fonts" in the "Runtime Settings" editor. Changes to the language with the given number.
- Language you have defined under "Languages and fonts" in the "Runtime Settings" editor.
- Language abbreviation in accordance with the VBScript5 reference: This changes to the language corresponding to the specified language code, e.g. "de-DE" for German (Germany) or "en-US" for English (United States).
  An overview of the language abbreviations is available in the basic information of VBScript under the topic "Area diagram-ID (LCID) Diagram".

Keyboard (RT Professional)

SetScreenKeyboardMode (RT Professional)

Description

Enables or disables the automatic display of the screen keyboard on the HMI device.

This system function is also used to prevent the display of the screen keyboard, e.g. because an external keyboard is connected to the HMI device.

Note

To enable the "SetScreenKeyboardMode" ("SetScreenKeyboardMode") system function on an HMI other than a Touch Panel device, set the "Use on-screen keyboard" check box in the "Runtime settings" dialog of the device settings.

Use in the function list

SetScreenKeyboardMode (Mode)

Use in user-defined functions

SetScreenKeyboardMode (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether the screen keyboard is hidden or shown:

0 (hmiOff) = Off: Screen keyboard is hidden

1 (hmiOn) = On: Screen keyboard is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes.

Keyboard operation for screen objects (RT Professional)

ShowBlockInTiaPortalFromAlarm (RT Professional)

Description

Switches from the alarm view to the affected program code of the selected project in the engineering system. Changing from the alarm view to the project in the TIA Portal enables further analysis of the program code, for example, search for a missing contact in the interlock.

You configure this system function at a screen object, for example, a button or a button of the toolbar in an alarm view. When you click the button, the necessary context is stored by the system and the corresponding location is automatically displayed in the program code after switching to the engineering system. You use the Write protection parameter to specify which authorization is required to enable the switch to the engineering system.

If you have configured the system function to a button, the TIA Portal is started in the mode that was specified with the write protection parameter. To enable different levels of security when accessing the system function, configure the buttons with two different authorizations:

- For the first button, configure the authorization for unlimited access by setting the write protection parameter to FALSE.
- For the second button, configure the authorization so that data cannot be changed by setting the write protection parameter to TRUE.

Use in the function list

ShowBlockInTiaPortalFromAlarm (write protection, offline mode, TIA Portal project path, screen name of alarm view, name of alarm view, error tag)

Use in user-defined functions

ShowBlockInTIAPortalFromAlarm ReadOnly, Offline_mode, TiaPortalProject_path, AlarmScreen_name, AlarmView_name, Error_tag

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional" (Page 1503).

Parameters

Write protection

Specifies whether the project is opened in read-only mode.

TRUE (default): TIA Portal is opened in read mode, the user cannot make any changes.

FALSE: TIA Portal is opened with read and right authorization.

Offline mode

Specifies whether the project is opened in offline mode.

TRUE: The TIA Portal does not go to online mode after the block is opened.

FALSE (default): Online mode is started after the block is opened.

TIA Portal project path

Path and file name of the project in the TIA Portal to which you change from the alarm view.

Screen name of the alarm view

Name of the screen that contains the alarm view.

Alarm view name

Object name of the alarm view to which you change from the TIA Portal.

Error tag

If an error occurs, information about the error is provided by the error tag.

ShowPLCCodeViewFromAlarm (RT Professional)

Description

Switches from the alarm view with the selected ProDiag alarm or selected GRAPH alarm to the PLC code view with the respective network. The screen that contains the PLC code view is either opened in a root screen or in a screen window.

Use in the function list

ShowPLCCodeViewFromAlarm (alarm screen name, alarm view name, basic screen name, screen window name, screen name of the PLC code view, name of the PLC code view, error tag)

Use in user-defined functions

ShowPLCCodeViewFromAlarm (Screen, Alarmview, Screen, ScreenWindow, Screen, PLCCodeView, Error tag)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional" (Page 1503).

Parameters

Alarm screen name

Name of the screen that contains the alarm view.

Alarm view name

Object name of the alarm view from which the selected alarm is transferred.

Basic screen name

Name of the screen which includes a screen window in which the screen can be displayed.

Screen window name

Name of the screen window in which the screen with the PLC code view can be displayed.

Screen name of the PLC code view

Name of the screen that contains the PLC code view.

Name of the PLC code view

Object name of the PLC code view.

Error tag

If an error occurs, information about the error is provided by the error tag.

Error messages

The following errors can be reported back using the error tag:

KOPAPI_E_TPSF_ALGCTRL_NOTSELECTED	Exactly one alarm must be selected
KOPAPI_E_TPSF_ONLYONLINELISTALLOWED	Only select alarms from the alarm list of the alarm view
KOPAPI_E_TPSF_ONALARMWRONGPRODUCERID	No GRAPH/ProDiag alarm selected
KOPAPI_E_TPSF_ONALARMNOSD0JUMPFLAG	It is not possible to jump over this GRAPH/ ProDiag alarm. The jump from an alarm in the alarm view to the PLC code view is made possible for the following supervision alarms: • With global supervisions, only for interlock supervisions (Interlock) • With local supervisions, for all basic supervisions at input parameters

9.11.2 Events (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

9.11.2.1 Overview (Basic Panels)

Editors (Basic Panels)

Introduction

The following table shows which events occur in which editor.

Technical data subject to change.

Icon	Editor
	Screens
	HMI alarms
	HMI tags
	Scheduler

Cleared (Page 1726)	X	--	--	--
Activate (Page 1727)	--	--	--	--

Change (Page 1728)	--	--	--	--
Loaded (Page 1728)	X	--	--	--
Execute (Page 1728)	--	--	--	X
Selection changed (Page 1728)	--	--	--	--
On exceeding (Page 1728)	--	--	X	--
On falling below (Page 1729)	--	--	X	--
When dialog is opened (Page 1729)	--	--	--	X
When dialog is closed (Page 1730)	--	--	--	X
User change (Page 1730)	--	--	--	X
Screen change (Page 1730)	--	--	--	X
Deactivate (Page 1731)	--	--	--	X
Double-click (Page 1732)	--	--	--	--
Press (Page 1732)	--	--	--	--
On Finish Input (Page 1736)	--	--	--	--
Press ESC twice (Page 1737)	--	--	--	--
Outgoing (Page 1738)	--	X	--	--
Incoming (Page 1738)	--	X	--	--
Click (Page 1739)	--	--	--	--
Click when flashing (Page 1739)	--	--	--	--
Loop-in alarm (Page 1742)	--	X	--	--
Release (Page 1742)	--	--	--	--
Alarm buffer overflow (Page 1743)	--	--	--	X
Acknowledge (Page 1745)	--	X	--	--
Reach margin (Page 1745)	--	--	--	--
Runtime Stop (Page 1746)	--	--	--	X
Press key (Page 1750)	--	--	--	--
Release key (Page 1750)	--	--	--	--
Overflow (Page 1751)	--	--	--	--
Switch OFF (Page 1752)	--	--	--	--
Switch ON (Page 1753)	--	--	--	--
Low free storage space (Page 1754)	--	--	--	--
Free space critically low (Page 1755)	--	--	--	--
Value change (Page 1755)	--	--	X	--
Time expired (Page 1756)	--	--	--	--

Basic objects (Basic Panels)

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	Line
	Ellipse
	Circle
	Rectangle
A	Text field
	Graphic view

					A	
Cleared (Page 1726)	--	--	--	--	--	--
Activate (Page 1727)	--	--	--	--	--	--
Change (Page 1728)	--	--	--	--	--	--
Loaded (Page 1728)	--	--	--	--	--	--
Execute (Page 1728)	--	--	--	--	--	--
Selection changed (Page 1728)	--	--	--	--	--	--
On exceeding (Page 1728)	--	--	--	--	--	--
On falling below (Page 1729)	--	--	--	--	--	--
When dialog is opened (Page 1729)	--	--	--	--	--	--
When dialog is closed (Page 1730)	--	--	--	--	--	--
User change (Page 1730)	--	--	--	--	--	--
Screen change (Page 1730)	--	--	--	--	--	--
Deactivate (Page 1731)	--	--	--	--	--	--
Double-click (Page 1732)	--	--	--	--	--	--
Press (Page 1732)	--	--	--	--	--	--
On Finish Input (Page 1736)	--	--	--	--	--	--
Press ESC twice (Page 1737)	--	--	--	--	--	--
Outgoing (Page 1738)	--	--	--	--	--	--
Incoming (Page 1738)	--	--	--	--	--	--
Click (Page 1739)	--	--	--	--	--	--
Click when flashing (Page 1739)	--	--	--	--	--	--
Loop-in alarm (Page 1742)	--	--	--	--	--	--
Release (Page 1742)	--	--	--	--	--	--
Alarm buffer overflow (Page 1743)	--	--	--	--	--	--
Acknowledge (Page 1745)	--	--	--	--	--	--
Reach margin (Page 1745)	--	--	--	--	--	--
Runtime Stop (Page 1746)	--	--	--	--	--	--
Press key (Page 1750)	--	--	--	--	--	--
Release key (Page 1750)	--	--	--	--	--	--
Overflow (Page 1751)	--	--	--	--	--	--
Switch OFF (Page 1752)	--	--	--	--	--	--

Switch ON (Page 1753)	--	--	--	--	--	--
Low free storage space (Page 1754)	--	--	--	--	--	--
Free space critically low (Page 1755)	--	--	--	--	--	--
Value change (Page 1755)	--	--	--	--	--	--
Time expired (Page 1756)	--	--	--	--	--	--

Elements (Basic Panels)

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	IO field
	Button
	Symbolic IO field
	Graphic IO field
	Date/time field
	Bar
	Switch

Cleared (Page 1726)	--	--	--	--	--	--	--
Activate (Page 1727)	X	X	X	X	--	--	X
Change (Page 1728)	--	X	X	--	--	--	X
Loaded (Page 1728)	--	--	--	--	--	--	--
Execute (Page 1728)	--	--	--	--	--	--	--
Selection changed (Page 1728)	--	--	--	--	--	--	--
On exceeding (Page 1728)	--	--	--	--	--	--	--
On falling below (Page 1729)	--	--	--	--	--	--	--
When dialog is opened (Page 1729)	--	--	--	--	--	--	--
When dialog is closed (Page 1730)	--	--	--	--	--	--	--
User change (Page 1730)	--	--	--	--	--	--	--
Screen change (Page 1730)	--	--	--	--	--	--	--
Deactivate (Page 1731)	X	X	X	X	--	--	X
Double-click (Page 1732)	--	--	--	--	--	--	--
Press (Page 1732)	--	X	--	--	--	--	--
On Finish Input (Page 1736)	--	--	--	--	--	--	--
Press ESC twice (Page 1737)	--	--	--	--	--	--	--
Outgoing (Page 1738)	--	--	--	--	--	--	--
Incoming (Page 1738)	--	--	--	--	--	--	--

Click (Page 1739)	--	X	--	--	--	--	--
Click when flashing (Page 1739)	--	--	--	--	--	--	--
Loop-in alarm (Page 1742)	--	--	--	--	--	--	--
Release (Page 1742)	--	X	--	--	--	--	--
Alarm buffer overflow (Page 1743)	--	--	--	--	--	--	--
Acknowledge (Page 1745)	--	--	--	--	--	--	--
Reach margin (Page 1745)	--	--	--	--	--	--	--
Runtime Stop (Page 1746)	--	--	--	--	--	--	--
Press key (Page 1750)	--	--	--	--	--	--	--
Release key (Page 1750)	--	--	--	--	--	--	--
Overflow (Page 1751)	--	--	--	--	--	--	--
Switch OFF (Page 1752)	--	--	--	--	--	--	X
Switch ON (Page 1753)	--	--	--	--	--	--	X
Low free storage space (Page 1754)	--	--	--	--	--	--	--
Free space critically low (Page 1755)	--	--	--	--	--	--	--
Value change (Page 1755)	--	--	--	--	--	--	--
Time expired (Page 1756)	--	--	--	--	--	--	--

Controls (Basic Panels)

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	Alarm view/alarm window
	Alarm indicator
	Trend view
	User view
	HTML browser (Basic Panel 2nd Generation only)
	Recipe view
	System diagnostics view
	Help indicator

Cleared (Page 1726)	--	--	--	--	--	--	--	--
Activate (Page 1727)	X	--	X	X	X	--	--	X
Change (Page 1728)	--	--	--	--	--	--	--	--
Loaded (Page 1728)	--	--	--	--	--	--	--	--
Execute (Page 1728)	--	--	--	--	--	--	--	--

Selection changed (Page 1728)	--	--	--	--	--	--	--	--
On exceeding (Page 1728)	--	--	--	--	--	--	--	--
On falling below (Page 1729)	--	--	--	--	--	--	--	--
When dialog is opened (Page 1729)	--	--	--	--	--	--	--	--
When dialog is closed (Page 1730)	--	--	--	--	--	--	--	--
User change (Page 1730)	--	--	--	--	--	--	--	--
Screen change (Page 1730)	--	--	--	--	--	--	--	--
Deactivate (Page 1731)	X	--	X	X	X	--	--	X
Double-click (Page 1732)	--	--	--	--	--	--	--	--
Press (Page 1732)	--	--	--	--	--	--	--	--
On Finish Input (Page 1736)	--	--	--	--	--	--	--	--
Press ESC twice (Page 1737)	--	--	--	--	--	--	--	--
Outgoing (Page 1738)	--	--	--	--	--	--	--	--
Incoming (Page 1738)								
Click (Page 1739)	--	X	--	--	--	--	--	--
Click when flashing (Page 1739)	--	X	--	--	--	--	--	--
Loop-in alarm (Page 1742)	--	--	--	--	--	--	--	--
Release (Page 1742)	--	--	--	--	--	--	--	--
Alarm buffer overflow (Page 1743)	--	--	--	--	--	--	--	--
Acknowledge (Page 1745)	--	--	--	--	--	--	--	--
Reach margin (Page 1745)	--	--	--	--	--	--	--	--
Runtime Stop (Page 1746)	--	--	--	--	--	--	--	--
Press key (Page 1750)	--	--	--	--	--	--	--	--
Release key (Page 1750)	--	--	--	--	--	--	--	--
Overflow (Page 1751)	--	--	--	--	--	--	--	--
Switch OFF (Page 1752)	--	--	--	--	--	--	--	--
Switch ON (Page 1753)	--	--	--	--	--	--	--	--
Low free storage space (Page 1754)	--	--	--	--	--	--	--	--
Free space critically low (Page 1755)	--	--	--	--	--	--	--	--
Value change (Page 1755)	--	--	--	--	--	--	--	--
Time expired (Page 1756)	--	--	--	--	--	--	--	--

9.11.2.2 Overview (RT Advanced, Panels, Comfort Panels)

Editors (RT Advanced, Panels, Comfort Panels)

Introduction

The following table shows which events occur in which editor.

Technical data subject to change.

Icon	Editor
	Screens
	HMI alarms
	HMI tags
	Logs
	Scheduler
	Audit Trail

Cleared (Page 1726)	X	--	--	--	--	--
Activate (Page 1727)	--	--	--	--	--	--
Change (Page 1728)	--	--	--	--	--	--
Loaded (Page 1728)	X	--	--	--	--	--
Execute (Page 1728)	--	--	--	--	X	--
Selection changed (Page 1728)	--	--	--	--	--	--
On exceeding (Page 1728)	--	--	X	--	--	--
On falling below (Page 1729)	--	--	X	--	--	--
When dialog is opened (Page 1729)	--	--	--	--	X	--
When dialog is closed (Page 1730)	--	--	--	--	X	--
User change (Page 1730)	--	--	--	--	X	--
Screen change (Page 1730)	--	--	--	--	X	--
Deactivate (Page 1731)	--	--	--	--	--	--
Double-click (Page 1732)	--	--	--	--	--	--
Press (Page 1732)	--	--	--	--	--	--
On Finish Input (Page 1736)	--	--	--	--	--	--
Press ESC twice (Page 1737)	--	--	--	--	--	--
Outgoing (Page 1738)	--	X	--	--	--	--
Incoming (Page 1738)	--	X	--	--	--	--
Click (Page 1739)	--	--	--	--	--	--
Click when flashing (Page 1739)	--	--	--	--	--	--
Loop-in alarm (Page 1742)	--	X	--	--	--	--
Release (Page 1742)	--	--	--	--	--	--
Alarm buffer overflow (Page 1743)	--	--	--	--	X	--
Acknowledge (Page 1745)	--	X	--	--	--	--
Reach margin (Page 1745)	--	--	--	--	--	--
Runtime Stop (Page 1746)	--	--	--	--	X	--
Press key (Page 1750)	--	--	--	--	--	--
Release key (Page 1750)	--	--	--	--	--	--
Overflow (Page 1751)	--	--	--	X	--	--
Switching (Page 1752)	--	--	--	--	--	--
Switch OFF (Page 1752)	--	--	--	--	--	--

Switch ON (Page 1753)	--	--	--	--	--	--
Low free storage space (Page 1754)	--	--	--	X	--	X
Free space critically low (Page 1755)	--	--	--	X	--	X
Value change (Page 1755)	--	--	X	--	--	--
Time expired (Page 1756)	--	--	--	--	--	--

Basic objects (RT Advanced, Panels, Comfort Panels)

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	Line
	Polyline
	Polygon
	Ellipse
	Circle
	Rectangle
A	Text field
	Graphic view

Cleared (Page 1726)	--	--	--	--	--	--	--	--
Activate (Page 1727)	--	--	--	--	--	--	--	--
Change (Page 1728)	--	--	--	--	--	--	--	--
Loaded (Page 1728)	--	--	--	--	--	--	--	--
Execute (Page 1728)	--	--	--	--	--	--	--	--
Selection changed (Page 1728)	--	--	--	--	--	--	--	--
On exceeding (Page 1728)	--	--	--	--	--	--	--	--
On falling below (Page 1729)	--	--	--	--	--	--	--	--
When dialog is opened (Page 1729)	--	--	--	--	--	--	--	--
When dialog is closed (Page 1730)	--	--	--	--	--	--	--	--
User change (Page 1730)	--	--	--	--	--	--	--	--
Screen change (Page 1730)	--	--	--	--	--	--	--	--
Deactivate (Page 1731)	--	--	--	--	--	--	--	--
Double-click (Page 1732)	--	--	--	--	--	--	--	--
Press (Page 1732)	--	--	--	--	--	--	--	--
On Finish Input (Page 1736)	--	--	--	--	--	--	--	--

Press ESC twice (Page 1737)	--	--	--	--	--	--	--	--
Outgoing (Page 1738)	--	--	--	--	--	--	--	--
Incoming (Page 1738)	--	--	--	--	--	--	--	--
Click (Page 1739)	--	--	--	--	--	--	--	--
Click when flashing (Page 1739)	--	--	--	--	--	--	--	--
Loop-in alarm (Page 1742)	--	--	--	--	--	--	--	--
Release (Page 1742)	--	--	--	--	--	--	--	--
Alarm buffer overflow (Page 1743)	--	--	--	--	--	--	--	--
Acknowledge (Page 1745)	--	--	--	--	--	--	--	--
Reach margin (Page 1745)	--	--	--	--	--	--	--	--
Runtime Stop (Page 1746)	--	--	--	--	--	--	--	--
Press key (Page 1750)	--	--	--	--	--	--	--	--
Release key (Page 1750)	--	--	--	--	--	--	--	--
Overflow (Page 1751)	--	--	--	--	--	--	--	--
Switching (Page 1752)	--	--	--	--	--	--	--	--
Switch OFF (Page 1752)	--	--	--	--	--	--	--	--
Switch ON (Page 1753)	--	--	--	--	--	--	--	--
Low free storage space (Page 1754)	--	--	--	--	--	--	--	--
Free space critically low (Page 1755)	--	--	--	--	--	--	--	--
Value change (Page 1755)	--	--	--	--	--	--	--	--
Time expired (Page 1756)	--	--	--	--	--	--	--	--

Elements (RT Advanced, Panels, Comfort Panels)

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	IO field
	Button
	Symbolic IO field
	Graphic IO field
	Date/time field
	Bar
	Switch
	Symbol library
	Slider
	Gauge
	Clock

Cleared (Page 1726)	--	--	--	--	--	--	--	--	--	--	--
Activate (Page 1727)	X	X	X	X	--	--	X	X	X	--	--
Change (Page 1728)	--	X	X	--	--	--	X	--	X	X	--
Loaded (Page 1728)	--	--	--	--	--	--	--	--	--	--	--
Execute (Page 1728)	--	--	--	--	--	--	--	--	--	--	--
Selection changed (Page 1728)	--	--	--	--	--	--	--	--	--	--	--
On exceeding (Page 1728)	--	--	--	--	--	--	--	--	--	--	--
On falling below (Page 1729)	--	--	--	--	--	--	--	--	--	--	--
When dialog is opened (Page 1729)	--	--	--	--	--	--	--	--	--	--	--
When dialog is closed (Page 1730)	--	--	--	--	--	--	--	--	--	--	--
User change (Page 1730)	--	--	--	--	--	--	--	--	--	--	--
Screen change (Page 1730)	--	--	--	--	--	--	--	--	--	--	--
Deactivate (Page 1731)	X	X	X	X	--	--	X	X	X	--	--
Double-click (Page 1732)	--	--	--	--	--	--	--	X	--	--	--
Press (Page 1732)	--	X	--	--	--	--	--	X	--	--	--
On Finish Input (Page 1736)	X		X	X	--	--	--	--	--	--	--
Press ESC twice (Page 1737)	--	--	--	--	--	--	--	--	--	--	--
Outgoing (Page 1738)	--	--	--	--	--	--	--	--	--	--	--
Incoming (Page 1738)	--	--	--	--	--	--	--	--	--	--	--
Click (Page 1739)	--	X	--	--	--	--	--	X	--	--	--
Click when flashing (Page 1739)	--	--	--	--	--	--	--	--	--	--	--
Loop-in alarm (Page 1742)	--	--	--	--	--	--	--	--	--	--	--
Release (Page 1742)	--	X	--	--	--	--	--	X	--	--	--
Alarm buffer overflow (Page 1743)	--	--	--	--	--	--	--	--	--	--	--
Acknowledge (Page 1745)	--	--	--	--	--	--	--	--	--	--	--
Reach margin (Page 1745)	--	--	--	--	--	--	--	--	--	--	--
Runtime Stop (Page 1746)	--	--	--	--	--	--	--	--	--	--	--
Press key (Page 1750)	--	--	--	--	--	--	--	--	--	--	--
Release key (Page 1750)	--	--	--	--	--	--	--	--	--	--	--
Overflow (Page 1751)	--	--	--	--	--	--	--	--	--	--	--
Switching (Page 1752)	--	--	--	--	--	--	--	--	--	--	--
Switch OFF (Page 1752)	--	--	--	--	--	--	X	--	--	--	--
Switch ON (Page 1753)	--	--	--	--	--	--	X	--	--	--	--
Low free storage space (Page 1754)	--	--	--	--	--	--	--	--	--	--	X
Free space critically low (Page 1755)	--	--	--	--	--	--	--	--	--	--	X
Value change (Page 1755)	--	--	--	--	--	--	--	--	--	--	--
Time expired (Page 1756)	--	--	--	--	--	--	--	--	--	--	--

Controls (RT Advanced, Panels, Comfort Panels)

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	Alarm view/alarm window
	Alarm indicator
	Trend view
	User view
	HTML browser
	Watch table
	Sm@rtClient view
	Recipe view
	f(x) trend view
	System diagnostics view / system diagnostics window
	Media Player[1]
	PLC code view
	GRAPH overview
	ProDiag overview
	Criteria analysis view
	Camera view[1]
	PDF view[1]
	Function key

[1] Only with Comfort HMI devices

Cleared (Page 1726)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Activate (Page 1727)	X	--	X	X	X	X	X	X	--	X	--	X	X	X	X	X	X	--
Change (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Loaded (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Execute (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Selection changed (Page 1728)	X	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
On exceeding (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
On falling below (Page 1729)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
When dialog is opened (Page 1729)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
When dialog is closed (Page 1730)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

User change (Page 1730)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Screen change (Page 1730)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Deactivate (Page 1731)	X	--	X	X	X	X	X	X	--	--	--	X	X	X	X	X	X	--
Double-click (Page 1732)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Press (Page 1732)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
On Finish Input (Page 1736)	--	--	--	--	--	--	--	--		--	--	--	--	--	--	--	--	--
Press ESC twice (Page 1737)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Outgoing (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Incoming (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click (Page 1739)	--	X	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click when flashing (Page 1739)	--	X	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Loop-in alarm (Page 1742)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Release (Page 1742)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Alarm buffer overflow (Page 1743)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Acknowledge (Page 1745)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Reach margin (Page 1745)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Runtime Stop (Page 1746)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click the Alarm view but-ton (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	X	X	--	--	--	--
Click the PLC code view button (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Press key (Page 1750)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	X
Release key (Page 1750)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	X
Overflow (Page 1751)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Switching (Page 1752)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Switch OFF (Page 1752)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Switch ON (Page 1753)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Low free storage space (Page 1754)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Free space critically low (Page 1755)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Value change (Page 1755)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Time expired (Page 1756)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

9.11.2.3 Overview (RT Professional)

Editors (RT Professional)

Introduction

The following table shows which events occur in which editor.

Technical data subject to change.

Icon	Editor
	Screens
	HMI alarms
	Logs
	Scheduler

Play ended (Page 1727)	--	--	--	--
Activate (Page 1727)	X	--	--	--
Change (Page 1728)	--	--	--	--
Loaded (Page 1728)	X	--	--	--
Execute (Page 1728)	--	--	--	X
Selection changed (Page 1728)	--	--	--	--
Double-click (Page 1732)	--	--	--	--
Press (Page 1732)	--	--	--	--
Operator input alarm property changed (Page 1732)	--	--	--	--
Block property changed (Page 1732)	--	--	--	--
Property diagram changed (Page 1733)	--	--	--	--
Property changed (Page 1733)	--	--	--	--
Trend property changed (Page 1733)	--	--	--	--
Column property changed (Page 1733)	--	--	--	--
Alarm statistics column property changed (Page 1734)	--	--	--	--
Statistics area column name property changed (Page 1734)	--	--	--	--
Change property status bar element (Page 1734)	--	--	--	--
Change property toolbar button (Page 1734)	--	--	--	--
Change property value axis (Page 1735)	--	--	--	--
Change property value column (Page 1735)	--	--	--	--
X-axis property changed (Page 1735)	--	--	--	--
Y-axis property changed (Page 1735)	--	--	--	--
Change property time axis (Page 1736)	--	--	--	--
Change property time column (Page 1736)	--	--	--	--
Error (Page 1737)	--	--	--	--
Outgoing (Page 1738)	--	X	--	--

Incoming (Page 1738)	--	X	--	--
In the alarm range (Page 1738)	--	--	--	--
In the tolerance range (Page 1738)	--	--	--	--
In the warning range (Page 1739)	--	--	--	--
Click (Page 1739)	X	--	--	--
Shortcut menu (Page 1740)	--	--	--	--
Volume (Page 1740)	--	--	--	--
Change layout (Page 1741)	--	--	--	--
Change ruler (Page 1741)	--	--	--	--
Press left mouse button (Page 1741)	X	--	--	--
Release left mouse button (Page 1741)	X	--	--	--
Loop-in alarm (Page 1742)	--	X	--	--
Release (Page 1742)	--	--	--	--
Move mouse (Page 1743)	--	--	--	--
Medium opened (Page 1743)	--	--	--	--
Medium closed (Page 1744)	--	--	--	--
Object changed (Page 1744)	X	--	--	--
Pause (Page 1744)	--	--	--	--
Acknowledge (Page 1745)	X	X	--	--
Press right mouse button (Page 1746)	X	--	--	--
Release right mouse button (Page 1746)	X	--	--	--
Runtime Stop (Page 1746)	--	--	--	--
Click the Alarm view button (Page 1747)	--	--	--	--
Click the PLC code view button (Page 1747)	--	--	--	--
Click toolbar button (Page 1747)	--	--	--	--
Change slider (Page 1747)	--	--	--	--
StepForward (Page 1748)	--	--	--	--
StepBack (Page 1748)	--	--	--	--
Lock changed (Page 1748)	--	--	X	--
Status changed (Page 1749)	--	X	--	--
Stop (Page 1749)	--	--	--	--
Switch toolbar (Page 1749)	--	--	--	--
Press key on keyboard (Page 1750)	X	--	--	--
Release key on keyboard (Page 1751)	X	--	--	--
Change video format (Page 1753)	--	--	--	--
Full screen (Page 1754)	--	--	--	--
Exit full screen (Page 1754)	--	--	--	--
Replay (Page 1756)	--	--	--	--

Basic objects (RT Professional)

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	Line
	Polyline
	Polygon
	Ellipse
	Ellipse segment
	Circle segment
	Ellipse arc
	Circular arc
	Circle
	Rectangle
	Connector
A	Text field
	Graphic view
	Pipe
	Double T-piece
	T piece
	Pipe bends

Play ended (Page 1727)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Activate (Page 1727)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Change (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Loaded (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Execute (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Selection changed (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Double-click (Page 1732)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Press (Page 1732)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Operator input alarm property changed (Page 1732)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Block property changed (Page 1732)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Property diagram changed (Page 1733)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Property changed (Page 1733)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Trend property changed (Page 1733)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Column property changed (Page 1733)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Alarm statistics column property changed (Page 1734)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Statistics area column name property changed (Page 1734)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property status bar element (Page 1734)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property tool-bar button (Page 1734)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property value axis (Page 1735)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property value column (Page 1735)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
X-axis property changed (Page 1735)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Y-axis property changed (Page 1735)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property time axis (Page 1736)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property time column (Page 1736)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Error (Page 1737)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Outgoing (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Incoming (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
In the alarm range (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
In the tolerance range (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
In the warning range (Page 1739)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click (Page 1739)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Shortcut menu (Page 1740)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Volume (Page 1740)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change layout (Page 1741)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change ruler (Page 1741)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Press left mouse button (Page 1741)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Release left mouse button (Page 1741)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Loop-in alarm (Page 1742)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Release (Page 1742)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Move mouse (Page 1743)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Medium opened (Page 1743)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Medium closed (Page 1744)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Object changed (Page 1744)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Pause (Page 1744)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Acknowledge (Page 1745)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Press right mouse button (Page 1746)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Release right mouse button (Page 1746)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Runtime Stop (Page 1746)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click the Alarm view button (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click the PLC code view button (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click toolbar button (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change slider (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
StepForward (Page 1748)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
StepBack (Page 1748)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Lock changed (Page 1748)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Status changed (Page 1749)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Stop (Page 1749)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Switch toolbar (Page 1749)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Press key on keyboard (Page 1750)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Release key on keyboard (Page 1751)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Full screen (Page 1754)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Exit full screen (Page 1754)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change video format (Page 1753)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Replay (Page 1756)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Elements (RT Professional)

Introduction

The following table shows which events occur on which elements.

Technical data subject to change.

Icon	Object
	IO field
	Editable text field
	List box
	Combo box
	Button
	Round button
	Symbolic IO field
	Graphic IO field
	Bar
	Symbol library
	Slider
	Scroll bar
	Check box
	Option button
	Gauge
	Clock
	Disk space view

Play ended (Page 1727)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Activate (Page 1727)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Change (Page 1728)	--	--	--	--	--	--	--	--	--	--	X	--	--	--	X	--	--
Loaded (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Execute (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Selection changed (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Double-click (Page 1732)	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--
Press (Page 1732)	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--
Operator input alarm property changed (Page 1732)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Block property changed (Page 1732)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Property diagram changed (Page 1733)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Property changed (Page 1733)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Trend property changed (Page 1733)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Column property changed (Page 1733)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Alarm statistics column property changed (Page 1734)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Statistics area column name property changed (Page 1734)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property status bar element (Page 1734)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property tool-bar button (Page 1734)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property value axis (Page 1735)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property value column (Page 1735)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
X-axis property changed (Page 1735)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Y-axis property changed (Page 1735)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property time axis (Page 1736)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change property time column (Page 1736)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Error (Page 1737)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Outgoing (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Incoming (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
In the alarm range (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	X
In the tolerance range (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	X

In the warning range (Page 1739)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	X
Click (Page 1739)	X	X	X	X	X	X	X	X	X	X	--	X	X	X	--	--	--
Shortcut menu (Page 1740)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Volume (Page 1740)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change layout (Page 1741)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change ruler (Page 1741)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Press left mouse button (Page 1741)	X	X	X	X	X	X	X	X	X	--	--	X	X	X	--	--	--
Release left mouse button (Page 1741)	X	X	X	X	X	X	X	X	X	--	--	X	X	X	--	--	--
Loop-in alarm (Page 1742)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Release (Page 1742)	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--
Move mouse (Page 1743)	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--
Medium opened (Page 1743)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Medium closed (Page 1744)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Object changed (Page 1744)	X	X	X	X	X	X	X	X	X	--	--	X	X	X	--	--	--
Pause (Page 1744)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Acknowledge (Page 1745)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Press right mouse button (Page 1746)	X	X	X	X	X	X	X	X	X	--	--	X	X	X	--	--	--
Release right mouse button (Page 1746)	X	X	X	X	X	X	X	X	X	--	--	X	X	X	--	--	--
Runtime Stop (Page 1746)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click the Alarm view button (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click the PLC code view button (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click toolbar button (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change slider (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
StepForward (Page 1748)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
StepBack (Page 1748)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Lock changed (Page 1748)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Status changed (Page 1749)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Stop (Page 1749)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Switch toolbar (Page 1749)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Press key on keyboard (Page 1750)	X	X	X	X	X	X	X	X	X	--	--	X	X	X	--	--	--
Release key on key-board (Page 1751)	X	X	X	X	X	X	X	X	X	--	--	X	X	X	--	--	--
Change video format (Page 1753)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Full screen (Page 1754)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Exit full screen (Page 1754)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Replay (Page 1756)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Controls (RT Professional)

Introduction

The following table shows which events occur on which controls.

Technical data subject to change.

Icon	Object
	Screen window
	User view
	HTML browser
	Print job/Script diagnostics
	Recipe view
	Alarm view
	f(x) trend view
	f(t) trend view
	Table view
	Value table
	System diagnostics view
	Media Player
	Channel diagnostics display
	PLC Code View
	GRAPH overview

Icon	Object
	ProDiag overview
	Criteria analysis view

Play ended (Page 1727)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Activate (Page 1727)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Change (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Loaded (Page 1728)	--	--	--	--	X	X	X	X	X	X	--	--	--	--	--	--	--
Execute (Page 1728)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Selection changed (Page 1728)	--	--	--	--	X	--	--	--	--	--	--	--	--	--	--	--	--
Double-click (Page 1732)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Press (Page 1732)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Operator input alarm property changed (Page 1732)	--	--	--	--	--	X	--	--	--	--	--	--	--	--	--	--	--
Block property changed (Page 1732)	--	--	--	--	--	X	--	--	--	X	--	--	--	--	--	--	--
Property diagram changed (Page 1733)	--	--	--	--	--	--	X	X	--	--	--	--	--	--	--	--	--
Property changed (Page 1733)	--	--	--	--	X	X	X	X	X	X	--	--	--	--	--	--	--
Trend property changed (Page 1733)	--	--	--	--	--	--	X	X	--	--	--	--	--	--	--	--	--
Column property changed (Page 1733)	--	--	--	--	X	X	--	--	--	X	--	--	--	--	--	--	--
Alarm statistics column property changed (Page 1734)	--	--	--	--	--	X	--	--	--	--	--	--	--	--	--	--	--
Statistics area column name property changed (Page 1734)	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--
Change property status bar element (Page 1734)	--	--	--	--	X	X	X	X	X	X	--	--	--	--	--	--	--
Change property toolbar button (Page 1734)	--	--	--	--	X	X	X	X	X	X	--	--	--	--	--	--	--
Change property value axis (Page 1735)	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--	--	--
Change property value column (Page 1735)	--	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--	--
X-axis property changed (Page 1735)	--	--	--	--	--	--	X	--	--	--	--	--	--	--	--	--	--

Y-axis property changed (Page 1735)	--	--	--	--	--	--	X	--	--	--	--	--	--	--	--	--	--
Change property time axis (Page 1736)	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--	--	--
Change property time column (Page 1736)	--	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--	--
Error (Page 1737)	--	--	--	--	--	--	--	--	--	--	--	--	--	X	--	X	X
Outgoing (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Incoming (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
In the alarm range (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
In the tolerance range (Page 1738)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
In the warning range (Page 1739)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Click (Page 1739)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	X	--	--
Shortcut menu (Page 1740)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Volume (Page 1740)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Change layout (Page 1741)	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--	--	--
Change ruler (Page 1741)	--	--	--	--	--	--	--	X	--	--	--	--	--	--	--	--	--
Press left mouse but-ton (Page 1741)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Release left mouse button (Page 1741)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Loop-in alarm (Page 1742)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Release (Page 1742)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Move mouse (Page 1743)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Medium opened (Page 1743)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Medium closed (Page 1744)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Object changed (Page 1744)	X	--	--	X	--	--	--	--	--	--	--	--	--	--	--	--	--
Pause (Page 1744)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Acknowledge (Page 1745)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Press right mouse but-ton (Page 1746)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Release right mouse button (Page 1746)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Runtime Stop (Page 1746)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Click the Alarm view button (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	X	X	--
Click the PLC code view button (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	X	--	--
Click toolbar button (Page 1747)	--	--	--	--	X	X	X	X	X	X	--	--	--	--	--	--	--
Change slider (Page 1747)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
StepForward (Page 1748)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
StepBack (Page 1748)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Lock changed (Page 1748)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Status changed (Page 1749)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Stop (Page 1749)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Switch toolbar (Page 1749)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Press key on key-board (Page 1750)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Release key on key-board (Page 1751)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Change video format (Page 1753)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Full screen (Page 1754)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Exit full screen (Page 1754)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--
Replay (Page 1756)	--	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--	--

9.11.2.4 Events (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Cleared (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the active screen on the HMI device is cleared.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Play ended (RT Professional)

Description

Occurs when the video or audio file has been played.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Activate (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the user selects a display or operating object using the configured tab sequence.

Note

Note that the availability of the event depends on the HMI device and object type.

Note

If the user e.g. clicks a button with the mouse, the "Click" event is triggered. Users wishing to trigger the "Enable" event must select the button using the tab key.

The "Enable" event is only used to detect whether an object was selected. The event does not trigger a password prompt.

For this reason, do not use the "Enable" event if you want to configure access protection on the function call of the object.

Change (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs if the status of a display and operator control object changes.

The status of an object changes if, for example, the user presses the key.

Note

Note that the availability of the event depends on the HMI device and object type.

Loaded (RT Professional)

Description

Occurs if a display or operating object is completely loaded in Runtime.

Note

Note that the availability of the event depends on the HMI device and object type.

Execute (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the scheduled task has been executed.

Selection changed (RT Professional)

Description

Occurs when the selection is changed.

On exceeding (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the high limit of a tag is exceeded.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

On falling below (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the low limit of a tag is undershot.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

When dialog is opened (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

The event is triggered when a modal dialog opens.

Note

Please note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

When dialog is closed (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

The event is triggered when a modal dialog closes.

Note

Please note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

User change (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when a user logs off at an HMI device or another user logs on at the HMI device.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Screen change (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when all configured display and operating objects are loaded in the screen after a screen change.

Use the "Loaded" event if you want to perform other system functions during a screen change to a certain screen.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Editors (Page 1707)

Basic objects (Page 1709)

Elements (Page 1710)

Controls (Page 1712)

Deactivate (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the user takes the focus from a display and operating object.

A screen object can be disabled using the configured tab order or by performing another action with the mouse.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

System functions or user-defined functions on the "Deactivate" event of a screen are not executed when a screen is being closed.

The "Deactivate" event is only used to detect whether an object was deselected. The event does not trigger a password prompt.

For this reason, do not use the "Deactivate" event if you want to configure access protection on the function call of the object.

See also

Editors (Page 1707)

Basic objects (Page 1709)

Elements (Page 1710)

Controls (Page 1712)

Double-click (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the user double-clicks on an object from the symbol library.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Press (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the user clicks on a button with the left mouse button, presses <RETURN> or <SPACE>.

Also occurs when the user right-clicks on an object of the symbol library.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Operator input alarm property changed (RT Professional)

Description

Occurs when the property of an operator input alarm is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Block property changed (RT Professional)

Description

Occurs when the property of a block, e.g., alarm text block, is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Property diagram changed (RT Professional)

Description

Occurs when the property of a trend diagram is changed.

Note

Note that the availability of the event depends on the HMI device and object type.

Property changed (RT Professional)

Description

Occurs when a property is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Trend property changed (RT Professional)

Description

Occurs when the property of the trend is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Column property changed (RT Professional)

Description

Occurs when the property of a column is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Alarm statistics column property changed (RT Professional)

Description

Occurs when the property of the alarm statistics column is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Statistics area column name property changed (RT Professional)

Description

Occurs when the column name of the statistics area is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Change property status bar element (RT Professional)

Description

Occurs when the property of a status bar element is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Change property toolbar button (RT Professional)

Description

Occurs when the property of a toolbar button is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Change property value axis (RT Professional)

Description

Occurs when the property of the time axis changes.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Change property value column (RT Professional)

Description

Occurs when the property of the value column is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

X-axis property changed (RT Professional)

Description

Occurs when the property of the X-axis is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Y-axis property changed (RT Professional)

Description

Occurs when the property of the Y-axis is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Change property time axis (RT Professional)

Description

Occurs when the property of the time axis changes.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Change property time column (RT Professional)

Description

Occurs when the property of the time column is changed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

On Finish Input (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Triggered when you confirm input at an I/O field by pressing ENTER, or by mouse click, or by touch screen operation.

The "On Finish Input" event is also started if the value of a tag does not change, for example, if a value is exceeded, or if a user cancels the dialog to acknowledge a tag (Audit option package) that has to be acknowledged.

The event is not triggered, on the other hand, by user logon or by input fields configured with an authorization.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Press ESC twice (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the user presses the <ESC> key twice at the HMI device.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Error (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs in the event of an error when configuring the view of a program network in the PLC Code View.

Note

Note that the availability of the event depends on the HMI device and object type.

Outgoing (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when an alarm is deactivated.

Note

Please note that the availability of the event depends on the HMI device and object type.

Incoming (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when an alarm is triggered and displayed in the alarm view.

Note

Please note that the availability of the event depends on the HMI device and object type.

In the alarm range (RT Professional)

Description

Occurs if the available storage space has reached the configured monitoring limit "alarm". You configure this event in the display and operating object "storage space display".

Note

Note that the availability of the event depends on the HMI device and object type.

In the tolerance range (RT Professional)

Description

Occurs if the available storage space has reached the configured monitoring limit "tolerance". You configure this event in the display and operating object "storage space display".

Note

Note that the availability of the event depends on the HMI device and object type.

In the warning range (RT Professional)

Description

Occurs if the available storage space has reached the configured monitoring limit "warning". You configure this event in the display and operating object "storage space display".

Note

Note that the availability of the event depends on the HMI device and object type.

Click (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs if the user clicks a display and operating object with the mouse or touches the touch display with a finger.

In case you click the incorrect object, prevent processing of configured function list as follows:

- Move the mouse pointer away from the object while keeping the mouse button pressed. Release the mouse button as soon as the mouse pointer leaves the object. The function list will then not be processed.
- On touch displays, the display must be touched with the finger until a reaction occurs, e.g., a screen change.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Click when flashing (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the user clicks a flashing alarm indicator with the mouse or touches it with a finger.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Shortcut menu (RT Professional)

Description

Ocurs when the shortcut menu opens.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Volume (RT Professional)

Description

Occurs when the volume changes.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Change layout (RT Professional)

Description

Activated when a property is changed.

Note

Note that the availability of the event depends on the HMI device and object type.

Change ruler (RT Professional)

Description

Activated when a property is changed.

Note

Note that the availability of the event depends on the HMI device and object type.

Press left mouse button (RT Professional)

Description

Occurs when the operator left-clicks on a display and operating object.

Note

Note that the availability of the event depends on the HMI device and object type.

Release left mouse button (RT Professional)

Description

Occurs when the user releases the left mouse button after clicking.

This event does not occur as long as the left button remains pressed.

If the operator releases the button after clicking on a button, this event will also occur even if the mouse pointer is no longer pointing on the button.

Note

Note that the availability of the event depends on the HMI device and object type.

Loop-in alarm (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

The "Loop-In-Alarm" event occurs when the user selects an alarm in the alarm view and clicks on the "Loop-In-Alarm" button or double-clicks on the alarm.

The "Loop-In-Alarm" event is configurable for discrete alarms and for analog alarms. A system function can be configured at the "Loop-In-Alarm" event, for example, switching to a screen in which you provide more information on the alarm that occurred. If a function is configured for the Loop-in-Alarm event and the selected alarm requires acknowledgment, then this alarm is also acknowledged the first time the event occurs.

If no Loop-In-Alarm is configured for the selected alarm, pressing the "Loop-In-Alarm" button has no effect.

You cannot configure local scripts for the "Loop-In-Alarm" event in Runtime Professional.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Release (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the user releases a button.

This even does not occur, as long as the button remains pressed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Alarm buffer overflow (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the configured size of the alarm buffer is reached.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Move mouse (RT Professional)

Description

Occurs when the user moves the mouse.

Note

Note that the availability of the event depends on the HMI device and object type.

Medium opened (RT Professional)

Description

Occurs when the medium is opened.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Medium closed (RT Professional)

Description

Occurs when the medium is closed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Object changed (RT Professional)

Description

Occurs if the user changes a property of a display and operating object in Runtime.

Note

Every action triggered by this event is logged on individually and then logged off again when the screen is changed. This behavior puts a burden on the working memory. Use this event only for important changes to properties, e.g., for an I/O field.

Note

The availability of the event is dependent on the HMI device and object type.

Pause (RT Professional)

Description

Occurs when playing of the video or audio file is paused.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Acknowledge (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the user acknowledges an alarm.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Reach margin (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the user reaches the beginning or the end of a scrollable area.

Note

Note that the availability of the event depends on the HMI device and object type.

Note

A user-defined function must not be configured for the "Boundary reached" event.

Configurable objects

The event can only be configured on the <Up> and <Down> keys, or on the keys on which you have configured the "ScreenObjectPageUp" or "ScreenObjectPageDown" system functions.

See also

Editors (Page 1707)

Basic objects (Page 1709)

Elements (Page 1710)

Controls (Page 1712)

Press right mouse button (RT Professional)

Description

Occurs when the user right-clicks on a display and operating object.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Release right mouse button (RT Professional)

Description

Occurs when the user releases the right mouse button after clicking.

This event does not occur as long as the right mouse button remains pressed.

If the operator releases the button after clicking on a button, this event will also occur even if the mouse pointer is no longer pointing on the button.

Note

Note that the availability of the event depends on the HMI device and object type.

Runtime Stop (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the user exits the Runtime software on the HMI device.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

A user-defined function must not be configured for the "Runtime stop" event.

Click the Alarm view button (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the user clicks the "Alarm View" button.

Note

Note that the availability of the event depends on the HMI device and object type.

Click the PLC code view button (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the user clicks the "PLC Code View" button.

Note

Note that the availability of the event depends on the HMI device and object type.

Click toolbar button (RT Professional)

Description

Occurs when the user clicks a button in the toolbar.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Change slider (RT Professional)

Description

Occurs when the volume slider changes.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

StepForward (RT Professional)

Description

Occurs when the video or audio file goes to the next sequence.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Elements (Page 1719)

StepBack (RT Professional)

Description

Occurs when the video or audio file goes to the previous sequence.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Lock changed (RT Professional)

Description

Occurs when the write access for the log is changed, e.g., from locked to released.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Status changed (RT Professional)

Description

Occurs when the status of an alarm is changed, e.g., from incoming to outgoing.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Stop (RT Professional)

Description

Occurs when playing of the video or audio file is stopped.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Elements (Page 1719)

Switch toolbar (RT Professional)

Description

Occurs when the property of the time axis changes.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Elements (Page 1719)

Press key (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the user presses a function key.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Release key (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the user releases a function key.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Press key on keyboard (RT Professional)

Description

Occurs when the user presses a key on the keyboard.

The keys <F10> and <ALT+PRINT> are not used by the operator for process operations.

Note

Note that the availability of the event depends on the HMI device and object type.

Release key on keyboard (RT Professional)

Description

Occurs when the user releases a key on the keyboard.

The keys <F10> and <ALT+PRINT> are not used by the operator for process operations.

Note

Note that the availability of the event depends on the HMI device and object type.

Overflow (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the configured size of the log is reached. You use the log type "Trigger event".

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Switching (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Description

Occurs when the user toggles a display and operating object, e.g. a switch from "ON" to "OFF".

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Switch OFF (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the user moves the display and operating object "Switch" to the OFF position.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Switch ON (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the user moves the display and operating object "Switch" to the ON position.

Note

Note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

Connected (RT Professional)

Description

Occurs when the user of a Mobile Panel connects to a connection box.

Note

Please note that the availability of the event depends on the HMI device.

Change video format (RT Professional)

Description

Occurs when the video format changes.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Full screen (RT Professional)

Description

Occurs when the full screen mode is activated.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Exit full screen (RT Professional)

Description

Occurs when the full screen mode is exited.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Low free storage space (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

This event is triggered if the storage space available on the medium to which the Audit Trail is less than the configured minimum.

See also

Editors (Page 1707)

Basic objects (Page 1709)

Elements (Page 1710)

Controls (Page 1712)

Free space critically low (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

This event is triggered if the storage medium to which an Audit Trail is saved provides insufficient storage space due to hardware restrictions.

See also

Editors (Page 1707)

Basic objects (Page 1709)

Elements (Page 1710)

Controls (Page 1712)

Value change (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the value of a tag or the value of an array element changes.

The value change of a tag is triggered by the PLC or by the user, e.g. when a new value is entered. No event is output if a system function changes the value.

Note

Please note that the availability of the event depends on the HMI device and object type.

See also

Editors (Page 1707)

Basic objects (Page 1709)

Elements (Page 1710)

Controls (Page 1712)

Replay (RT Professional)

Description

Occurs when the video or audio file has been replayed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Elements (Page 1719)

Time expired (RT Advanced, Basic Panels, Panels, Comfort Panels)

Description

Occurs when the time configured in the scheduler expires.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

See also

Editors (Page 1707)
Basic objects (Page 1709)
Elements (Page 1710)
Controls (Page 1712)

9.11.3 WinCC Engineering V15.1 - Programming reference

WinCC Programming reference

You can find a detailed description of VB scripts, C scripts and Runtime API in the "WinCC Engineering V15.1 – Programming reference" manual.

Mobile Panels (Panels, Comfort Panels) 10

10.1 Basics (Panels, Comfort Panels)

10.1.1 Field of application of the Mobile Panel Wireless (Panels, Comfort Panels)

Introduction

The Mobile Panel Wireless enables you to move through the plant without restrictions and with or without security technology. You can operate and monitor moving machine parts or the entire plant depending on the situation. You can plan, configure, simulate and extend wireless networks quickly and reliably.

The Mobile Panels Wireless are based on standardized WLAN technology and are specially tailored to the requirements of automation.

WLAN area

A WLAN area is the area of the plant in which the HMI device communicates with the PLC over a wireless local area network. The wireless coverage of the network has a sufficient signal strength in this area.

The WLAN of a plant is formed by the radio area of one or more Access Points.

Access Point

The Access Point serves as a gateway between the wireless and hard-wired network. The HMI device communicates with an Access Point via WLAN. You can operate the different machines or plants without annoying cables. The HMI device is connected to a network in which it communicates with a PLC via the access point.

Effective range

Safety-related operator input is only possible in a limited part of a WLAN area upstream of a machine or plant known as the effective range. One exception to this rule is the emergency stop, which works in the entire WLAN area.

The configured effective ranges are stored in the HMI device. The HMI device can only log on to a machine within the effective range. This also requires there no other HMI device is logged on in the same effective range.

To ensure clear logon and machine operator inputs, one effective range must not overlap another effective range.

Tags

Effective ranges are formed by the spatial assignment of tags to a machine or plant. Tags create a geometrical space which is relevant for logon to a machine. After logon you operate the machine using acknowledgement buttons.

A distinction is made between two types of tags:

- Transponder
- RFID tag

Transponder

After successful logon to a machine, the spanned geometrical area will be monitored. The distance between the HMI device and the transponder can only be measured if the two devices are in the reception range of each other. When you leave the effective range, safety-related operator input of the machine with acknowledgment buttons is prevented.

Zones in the transponder system

Zones are configurable objects within a transponder system. A zone is defined by the maximum distance from one or more transponders. In a hard-wired Mobile Panel, the length of the connection cable defines a zone around a connection box.

Zones are operating areas in which safety-related operator input is not possible. To enter a zone, you have to configure a screen change to the correct process picture.

RFID tag

After successful logon to a machine, the spanned geometrical area is not monitored. Monitoring therefore calls for an additional organizational measure, for example: a light barrier or a protective fence. The protection area is separated from the plant by a security system. The protection area is the area in the plant in which you operate one or more machines in fail-safe operation. One or more RFID Tags are distributed within protection zones. The protection area is not a configurable object.

10.1.2 How does the transponder system work? (Panels, Comfort Panels)

Overview

To give a basic understanding of all this, the following section describes the interaction between:

- HMI device
- Zone
- Effective range
- Transponder

Note

You should always assign administrator rights or set up encryption to protect the project containing a Mobile Panel Wireless for fail-safe operation against unauthorized access.

Principle of operation in the transponder system

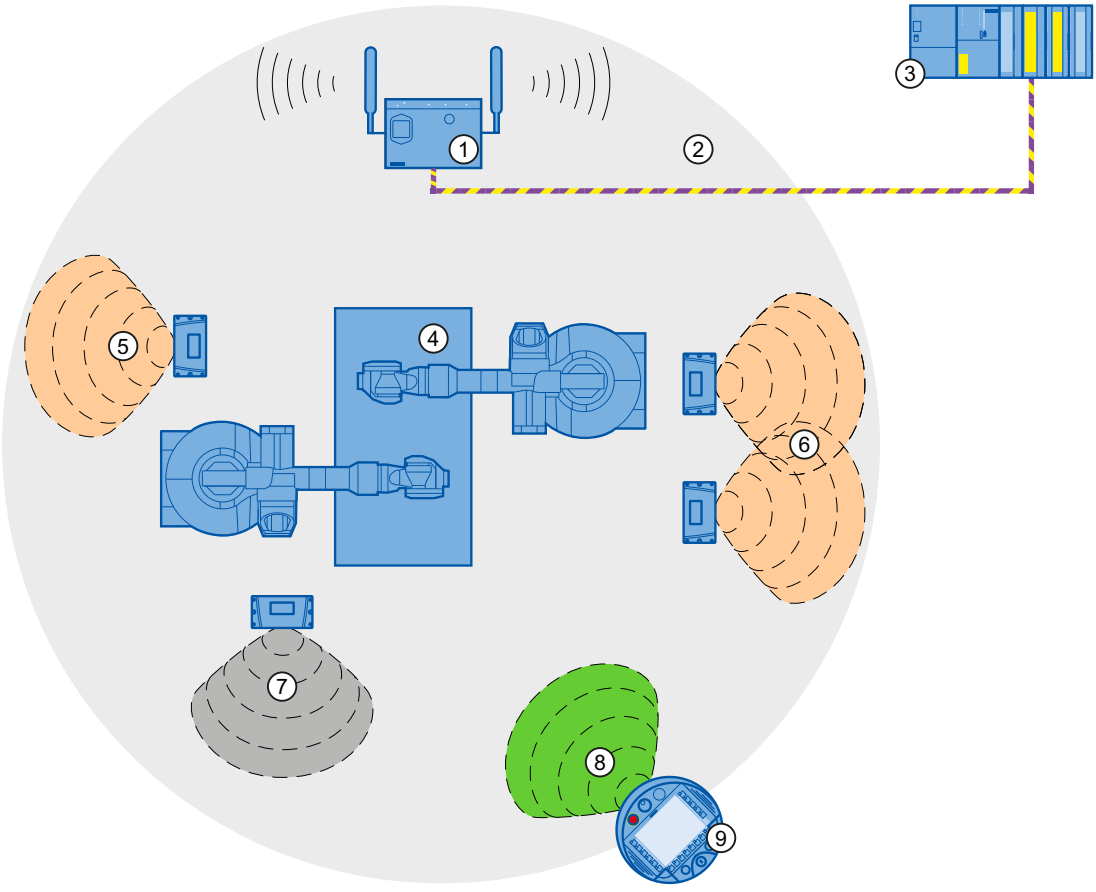

1 Access Point
2 WLAN of the Access Point
3 PLC
4 Plant
5 Effective range "Robot 1" with a transponder
6 Effective range "Robot 2" with two transponders
7 Zone "Zone 1" with a transponder
8 Radio area of the HMI device
9 HMI device

Example:

- A screen change is configured for "Zone 1".
- Transponder1 is assigned to the "Robot 2" effective range.
- The operator enters the radio cone of the transponder to receive the transponder ID.
- To log onto the machine "Robot 2", the operator presses the object "Effective range - Name".
- The "Effective range - Logon" dialog box opens. The operator enters the effective range ID and confirms with "OK". The dialog is closed.
- If the logon was successful, then the acknowledgement buttons will be active.
- The HMI device measures the distance from the transponder which is less than the configured distance of the effective range.

Result:

The operator is logged onto the machine "Robot 2". The operating element "Effective range - Name" is green and shows the name of the effective range "Robot 2".

If the HMI device is logged onto the effective range, then the following will apply:

- The operator may not leave the effective range without logging off. If the operator leaves the effective range without logging off for more than 30 seconds, a local rampdown will occur.
- No other HMI device can log onto the machine.

To log off from the machine, the operator presses the operating element "Effective range - Name". A logoff dialog is displayed. The operator confirms the dialog.

After successful logoff, another operator can log on to the machine "Robot2".

When the operator enters "Zone 1", a screen change will take place.

10.1.3 How does the RFID system work? (Panels, Comfort Panels)

Overview

RFID stands for radio frequency identification. RFID allows wireless identification of objects without contact or visual sighting.

For a basic understanding, the following section describes the interaction between:

- HMI device
- Effective range (RFID)
- RFID tag

Note

RFID Tags in the plant area

You may only affix RFID Tags in restricted plant areas that are protected by additional protective measures, for example, protective railings.

Principle of operation in the RFID system

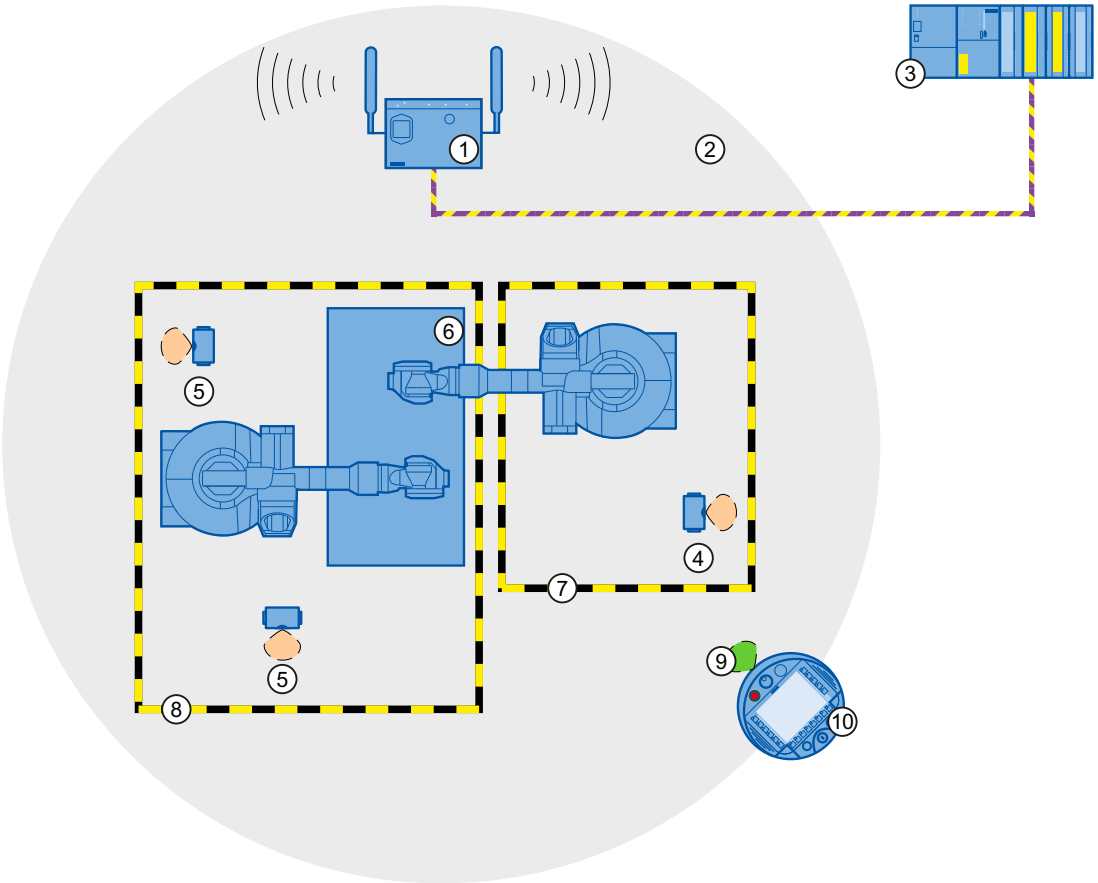

① Access Point
② WLAN of the Access Point
③ PLC
④ RFID tag with effective range (RFID) for logon to "Robot 1"
⑤ RFID tag with effective range (RFID) for logon to "Robot 2"
⑥ Plant
⑦ Protective area limit of the machine "Robot1"
⑧ Protective area limit of the machine "Robot2"
⑨ Radio area of the HMI device
⑩ HMI device

Example:

- "RFID Tag1" is assigned to the effective range (RFID) "Robot1".
- An operator enters the restricted plant area. The operator manually switches to the screen with the configured operating element "Effective range - Name (RFID)".

- At "RFID Tag1", he presses the "Scan" button.
  During the logon operation, the button is yellow and displays the text "Scanning…".
- The "Effective range - Logon" dialog box opens. The operator enters the effective range (RFID) ID and confirms it with "OK". The dialog is closed.
- The "Confirmation of logon" dialog box opens. The operator confirms logon with the acknowledgment button.
- If logon was successful, the acknowledgement buttons are active.

Result:

The operator is logged on to the machine "Robot 1" and operates the machine within the protection area in fail-safe operation. The operating element "Effective range - Name (RFID)" is green and shows the name of the effective range (RFID) "Robot1".

If the HMI device is logged on to the machine, the following applies:

- The user may not leave the protection area without logging off from the machine. If the operator leaves the protection area without logging off, a local rampdown occurs.
- No other HMI device can log on to this machine.

To log off from the machine, the operator presses the operating element "Effective range - Name (RFID)". A logoff dialog is displayed. The operator confirms the dialog.

After successful logoff, another operator can log on to the machine "Robot 1".

10.1.4 Zones (Panels, Comfort Panels)

Introduction

The section below applies to the following Mobile Panels:

- Mobile Panels Wireless, for example, Mobile Panel 277 IWLAN V2 and Mobile Panel 277F IWLAN V2
- KTP Mobile Panels, wired, e.g., Mobile Panel KTP700

A "Zones" work area is only visible for these HMI devices.

Note

For more information refer to the operating instructions for the HMI device:

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

Open

Depending on the HMI device selected, open the work area in the project window by double-clicking on "Zones" or "Zones and effective ranges".

Work area

For Mobile Panels Wireless, the "Zones" work area displays the zones that have been set up and the transponders assigned to them.

For wired Mobile Panels, the "Zones" work area shows the zones that have been set up and the associated box ID of the connection box.

Inspector window

When a zone is selected, edit the name and display name under "General".

For Mobile Panels Wireless, configure the limit of the zone around the transponder. The zone of a transponder has the events "Upon entry" and "Upon exit"

With wired Mobile Panels enter the box ID of the connection box. The zone of a connection box includes the event "Connected".

Configure the system function ActivateScreen" to the events.

10.1.5 Zones work area at transponders (Panels, Comfort Panels)

Introduction

The "Zones" work area displays the zones and the transponders assigned to them in table format. You create a list of transponders and assign certain transponders to a zone. The limit of the zone is defined by the maximum distance from the transponders.

Principle

The work area consists of the "Zones" and "Transponders" tables.

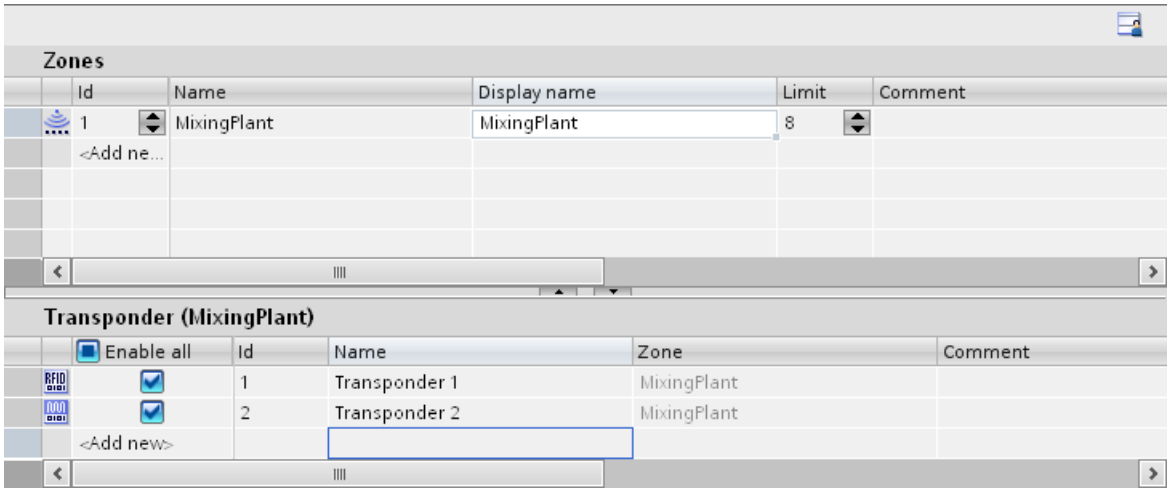

If you select a zone in the "Zones" table, the "Transponder" table displays the following:

- Transponder enabled: The transponder is assigned to the selected zone.
- Transponder deactivated: The transponder is not yet assigned to any zone.
- Transponder not available: The transponder has already been assigned to another zone. To undo the assignment, switch to the relevant zone and deactivate the transponder.

The zone and transponder IDs are initially assigned automatically. They can be changed, however.

Please note the following:

- A maximum of 254 zones may be configured.
- The zone ID must be unique and fall within the range from 1 - 254.
- You can initially configure transponders without assigning them to a zone.
- There may be no more than 255 transponders assigned to a zone.
- The transponder ID must be unique and fall within the range from 1 - 65534.

The transponder IDs are set on the transponder.

10.1.6 Zones work area on connection boxes (Panels, Comfort Panels)

Introduction

The "Zones" work area displays the box IDs of the connection boxes and their zones in table format.

You create a list of box IDs of the connection boxes and assign specific connection boxes to a zone. The limit of the zone is defined by the length of the connecting cable to your Mobile Panel.

Principle

The work area consists of the "Zones" table.

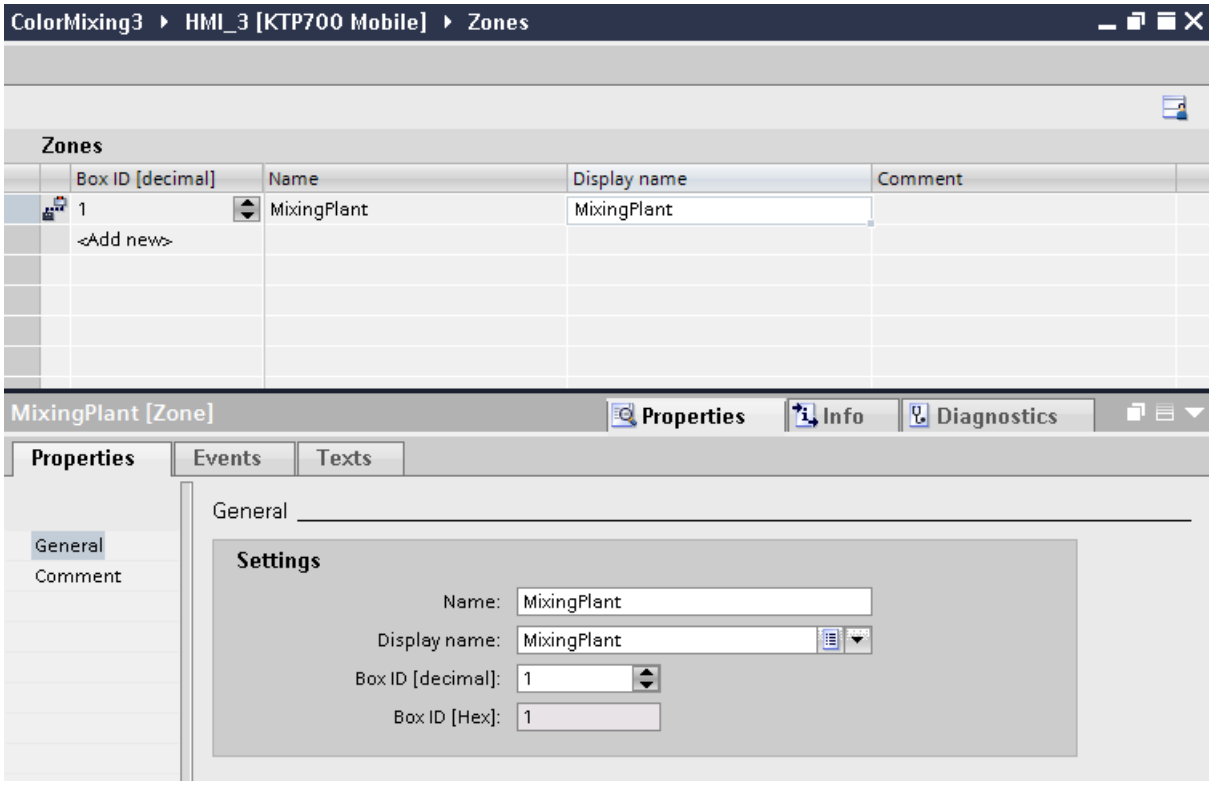

The box ID is set on the connection box and can be read out from the HMI device.

10.1.7 Effective ranges (Panels, Comfort Panels)

Introduction

The following section applies only to the Mobile Panels Wireless that support fail-safe operation, e.g. Mobile Panel 277F IWLAN V2. The "Effective ranges" work area is only visible for these HMI devices.

You set up effective ranges in order to control safety related operations. An effective range is defined by the maximum distance from one or more transponders.

Note

For more information refer to the operating instructions for the HMI device:

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

Open

Open the work area in the project window by double-clicking on "Zones & effective ranges". Click the "Effective ranges" tab.

Work area

The "Effective ranges" work area displays the effective ranges that have been set up and the transponders assigned to them.

Inspector window

When an effective range is selected, you can edit the names, display names and the limits of the effective range under the "General" category.

When you select a transponder, you can see both the effective range and the zone that are assigned to that transponder. You can only assign a transponder to one effective range so that the effective ranges do not overlap.

10.1.8 Effective ranges working area (Panels, Comfort Panels)

Introduction

The "Effective ranges" work area displays the effective ranges and their transponders in table format. You create a list of transponders and assign certain transponders to an effective range. The limit of the effective range is defined by the maximum distance from the transponders.

Principle

The work area consists of the "Effective ranges" and "Transponders" tables.

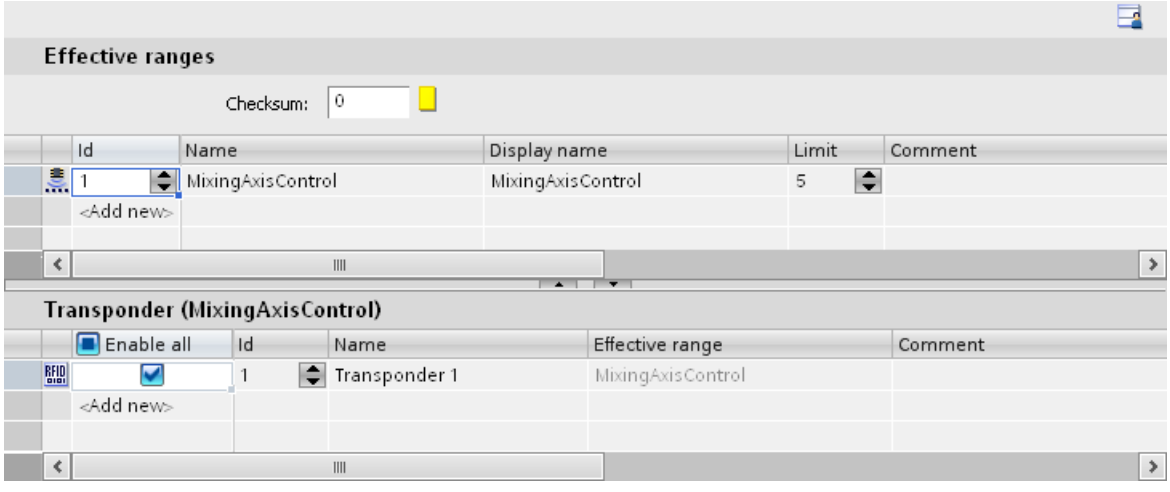

The HMI device calculates a checksum from the local data so that the configured effective ranges and transponders reliably match those locally on the machine. The project cannot be

started on the HMI device unless the local checksum agrees with the checksum stored in the "Effective ranges" editor.

When you select an effective range from the "Effective ranges" table, the "Transponder" table displays the following:

- Transponder enabled: The transponder is assigned to the selected effective range.
- Transponder deactivated: The transponder is not yet assigned to any effective range.
- Transponder not available: The transponder has already been assigned to another effective range. To undo the assignment, switch to the relevant effective range and deactivate the transponder.
- The zone assigned to the transponder is displayed in addition to the effective range.

The effective range and transponder IDs are initially assigned automatically. They can be changed, however.

Please note the following:

- A maximum of 127 effective ranges may be configured.
- The effective range ID must be unique and fall within the range from 1 - 127.
- The display name of an effective range must not be the same as its ID.
- You can initially configure transponders without assigning them to an effective range.
- There may be no more than 127 transponders assigned to an effective range.
- The transponder ID must be unique and fall within the range from 1 - 65534.

The transponder IDs are set on the transponder.

Effective range in Runtime

To log on to the effective range in Runtime, only the display name of the effective range appears to the operator in the Runtime language. The operator reads the effective range ID in the plant and enters it on the HMI device. This ensures that the right machine operated. The acknowledgment buttons are activated after successful logon.

10.1.9 Effective ranges (RFID) (Panels, Comfort Panels)

Introduction

The following section only applies to the Mobile Panel 277F IWLAN (RFID tag) which supports fail-safe operation with RFID. The "Effective ranges (RFID)" work area is only visible with these HMI devices.

To monitor safety-related operator inputs, you set up effective ranges (RFID).

Note

For more information, refer to the operating instructions of the HMI device:

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

Open

Open the work area in the project window by double-clicking on "Effective ranges (RFID)".

Work area

The "Effective ranges (RFID)" work area displays the effective ranges that have been set up and the RFID Tags assigned to them.

Inspector window

When an effective range (RFID) is selected, you can edit the name, display name and ID of the effective range under the "General" category.

10.1.10 Work area for effective ranges (RFID) (Panels, Comfort Panels)

Introduction

The "Effective ranges (RFID)" work area displays the effective ranges (RFID) and their RFID Tags in table format. You create a list of RFID Tags and assign specific RFID Tags to an effective range.

Principle

The work area consists of the "Effective ranges (RFID)" and "RFID Tags" tables.

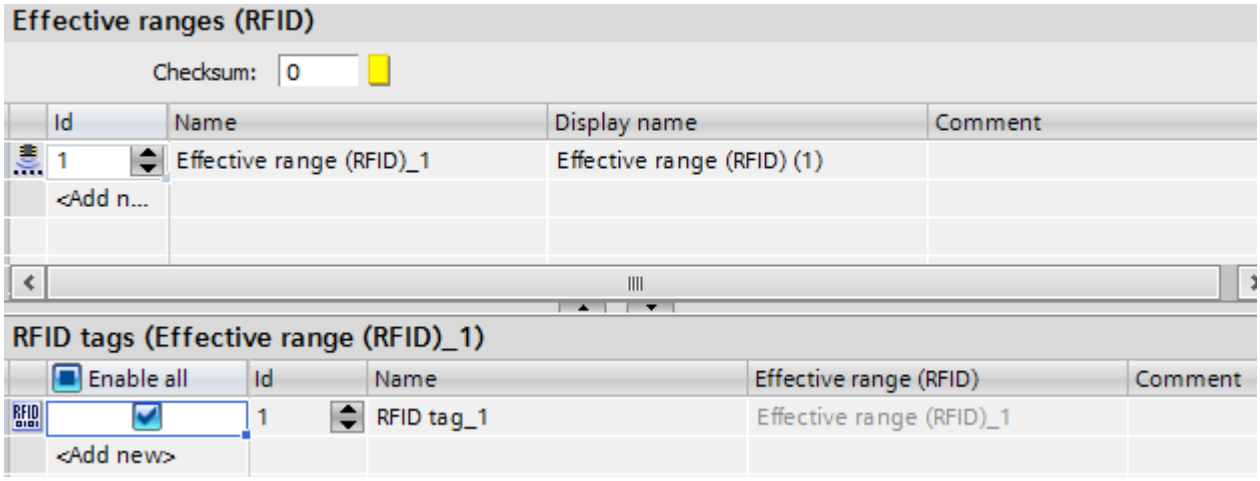

The HMI device calculates a checksum from the local data so that the configured effective ranges (RFID) and RFID Tags reliably match those locally on the machine. The project cannot be started on the HMI device unless the local checksum agrees with the checksum stored in the "Effective ranges (RFID)" editor.

When you select an effective range (RFID) in the "Effective ranges (RFID)" table, the "RFID Tags" table displays the following:

- RFID Tag enabled: The RFID Tag is assigned to the selected effective range (RFID).
- RFID Tag disabled: The RFID Tag is not yet assigned to any effective range (RFID).
- RFID Tag not available: The RFID Tag has already been assigned to another effective range (RFID). To undo the assignment, switch to the relevant effective range (RFID) and deactivate the RFID Tag.

The IDs of the effective ranges (RFID) and RFID Tags are initially assigned automatically. They can be changed, however.

Please note the following:

- A maximum of 127 effective ranges (RFID) may be configured.
- The effective range (RFID) ID must be unique and fall within the range from 1 - 127.
- The display name of an effective range (RFID) must not be the same as its ID.
- You can initially configure RFID Tags without assigning them to an effective range (RFID).
- A maximum of 127 RFID Tags is available to assign to an effective range (RFID).
- The RFID Tag ID must be unique and fall within the range 1-65534.

The RFID Tag ID is written to the RFID Tag by the HMI device.

Effective range in Runtime

To log on to the effective range (RFID) in Runtime, only the display name of the effective range (RFID) is displayed to the operator in the Runtime language. The operator reads the effective range (RFID) ID in the plant and enters it on the HMI device. This ensures that the right machine operated. The acknowledgment buttons are activated after successful logon.

10.2 Configuring Mobile Panels V2 for fail-safe operation (from V12 SP1) (Panels, Comfort Panels)

10.2.1 Configuration overview (Panels, Comfort Panels)

Introduction

Configure fail-safe operation in the "Devices & Networks" editor of the TIA Portal, version V12 SP1 or later, for the following Mobile Panels V2:

- Mobile Panel 277F IWLAN V2
- Mobile Panel 277F IWLAN (RFID Tag)

Procedures overview

1. Create a PLC for fail-safe operation.
2. Create a Mobile Panel V2 for fail-safe operation.
3. Create a connection between the Mobile Panel and PLC.

See also

Creating Mobile Panel (Page 1773)

Creating PLC for fail-safe operation (Page 1772)

Creating a connection between the Mobile Panel and PLC (Page 1774)

10.2.2 Creating PLC for fail-safe operation (Panels, Comfort Panels)

Requirement

A project is open

Procedure

1. Click "Add new device" in the project navigation.
2. Click on "Controller" in the "Add new device" dialog.
3. Select a PLC for fail-safe operation, for example CPU 315F-2 PN/DP.
   The "Devices & Networks" editor opens.
4. Select the device in the "Device overview".
5. Click "Properties > General > PROFINET interface [X2] >Ethernet addresses > Interface connected with > Add new subnet" in the inspector window.

6. Enter the IP address under "IP protocol > Set IP address in the project".
7. Enter a time for "Properties > PROFINET interface [X2] > F parameter > Default F monitoring time for F I/O of this interface".
8. Specify the properties for fail-safe operation in "Properties > Fail-safe":
   – "F-capability activated"
   – "Basis for PROFIsafe addresses"
9. Specify further properties for fail-safe operation as usual, for example, protection level.

Result

You have created a PLC in the project and made the settings for fail-safe operation.

See also

Configuration overview (Page 1772)

10.2.3 Creating Mobile Panel (Panels, Comfort Panels)

Requirement

- The Inspector window is open.
- A PLC for fail-safe operation is created in the project, for example, CPU-315F-2PN/DP.

Inserting Mobile Panel into the project.

1. Click "Add new device" in the project navigation.
2. Click on "HMI" in the "Add new device" dialog.
3. Select a Mobile Panel V2 for fail-safe operation, for example, Mobile Panel 277F IWLAN V2. The device is inserted into the project.

Defining properties

1. Open the "Devices and Networks" editor.
2. Select the Mobile Panel in the work area.
3. Open "Properties > General" in the inspector window.
4. To facilitate assignment in the project, enter the name of the associated module under "Name"
5. Enter the IP address under "Ethernet addresses > IP protocol > Set IP address in the project".

Result

You have created a Mobile Panel for fail-safe operation in the project.

See also

Configuration overview (Page 1772)

10.2.4 Creating a connection between the Mobile Panel and PLC (Panels, Comfort Panels)

Requirements

- A PLC for fail-safe operation is created in the project e.g. CPU-315F-2PN/DP.
- A device has been created.
- The "Devices & Networks" editor is open.

Procedure

1. Open the "Network view".
2. Select the Mobile Panel in the work area.
3. Select "Properties > General > Mode > IO device" in the inspector window.
4. Assign the fail-safe PLC under "Assigned IO controller".
   PROFINET is activated and the transfer areas of I-device communication are displayed.
5. Click "Connections" and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.
6. Click on the PLC interface and drag-and-drop a connection to the interface of the HMI device.
   An HMI connection is created.
7. Select the Mobile Panel in the work area.
8. Configure the F parameters under "Properties > General > F parameters".
   To set a different F-monitoring time from that specified by the PLC, select "Manual assignment of F-monitoring time".
   Enter an F-monitoring time suitable for your HMI device and plant.

Result

The HMI device is connected to the PLC and you have made the settings for fail-safe operation.

See also

Configuration overview (Page 1772)

10.3 Configuring KTP Mobile Panels for fail-safe operation (Panels, Comfort Panels)

10.3.1 Creating KTP Mobile Panel for fail-safe operation (Panels, Comfort Panels)

Introduction

Configure fail-safe operation in the "Devices & Networks" editor for the following KTP Mobile Panels:

- KTP400F Mobile Panel
- KTP700F Mobile Panel
- KTP900F Mobile Panel

Requirement

- The Inspector window is open.
- A PLC for fail-safe operation is created in the project, for example, CPU-1517F-2PN/DP.

Inserting a mobile panel into the project

1. Click "Add new device" in the project navigation.
2. Click on "HMI" in the "Add new device" dialog.
3. Select a KTP Mobile Panel for fail-safe operation, for example, Mobile Panel KTP700F.
   The device is inserted into the project.

Configuring fail-safe operation for the mobile panel

1. Open the network view in the "Devices & Networks" editor.
2. Select the Mobile Panel in the work area.
3. Select "Properties > General > Mode > IO device" in the inspector window.
4. Assign the fail-safe PLC under "Assigned IO controller".
   PROFINET is activated and the transfer areas of the I-device communication are displayed.
5. In the Inspector window, click "Activate PROFIsafe" under "Properties > General" > PROFIsafe".
   PROFIsafe is activated and the F-parameters are displayed under "Properties > General> PROFIsafe parameters > F-parameters".
6. To enter a new F-monitoring time, select "Manual assignment of F-monitoring time".
7. Enter an F-monitoring time suitable for your HMI device and plant.

Result

The fail-safe operation of the KTP Mobile Panel is configured in WinCC. Now create the corresponding F-blocks in STEP 7 and interconnect them in the controller. The PROFIsafe address must be set on the Mobile Panel.

You can find additional information on configuration and programming of fail-safe blocks in the manual "SIMATIC Safety - Configuration and Programming" of the STEP 7 documentation and in the device manual of your mobile panel.

10.3.2 Configuring the termination of the PROFIsafe connection (Panels, Comfort Panels)

Procedure

Before you disconnect your Mobile Panel from a connection box, you have to terminate the connection to PROFIsafe.

Use the "Disconnect PROFIsafe" system function to close the PROFIsafe connection. The system function forwards your call to the fail-safe PROFINET module.

Requirements

- A mobile panel has been created for fail-safe operation, for example, KTP700F Mobile Panel.
- A screen has been created and opened.

Procedure

To configure the PROFIsafe the connection and disconnection, follow these steps:

- Configure the "Disconnect PROFIsafe" system function on a button, for example, or use it in a script.

Result

If you release the configured trigger for terminating the PROFIsafe connection in runtime, a dialog for ending the PROFIsafe connection opens. The user can select "Yes" or "No".

10.4 Working with zones on transponders (Panels, Comfort Panels)

10.4.1 Configuring a zone in the transponder system (Panels, Comfort Panels)

Validity

The next section applies to all Mobile Panels Wireless, e.g. Mobile Panel 277 IWLAN.

Introduction

You will have to set up a zone in order to carry out plant-specific operator control and monitoring tasks.

Requirements

- A suitable editing language, e.g. English, must have been set in the "Project languages" editor.
- The "Zones" work area is open.

Procedure

1. Double-click "Add..." in the "Zones" table.
2. Enter the "MixingPlant" zone in the "Name" field.
3. Enter "MixingSystem" as the "Display name" for the zone at runtime. Enter zone "1" in the "ID" field.
4. Enter "8" meters as the zone "Limit".
5. Double-click "Add..." in the "Transponders" table.
6. Enter "Transponder 1" as the "Name" of the transponder.
7. Enter transponder "1" in the "ID" field.
   Then set transponder IDs on the transponder.
8. Enable the assignment of "Transponder 1".
   "Transponder 1" is assigned to the "MixingPlant" zone.
9. Repeat steps 6 to -9 accordingly for "Transponder 2".
   "Transponder 2" is assigned to the "MixingPlant" zone.

Result

The "MixingPlant" zone is configured as two conical areas arranged at a distance of 8 meters, regardless of the effective ranges: one area around "Transponder 1" and one around "Transponder 2".

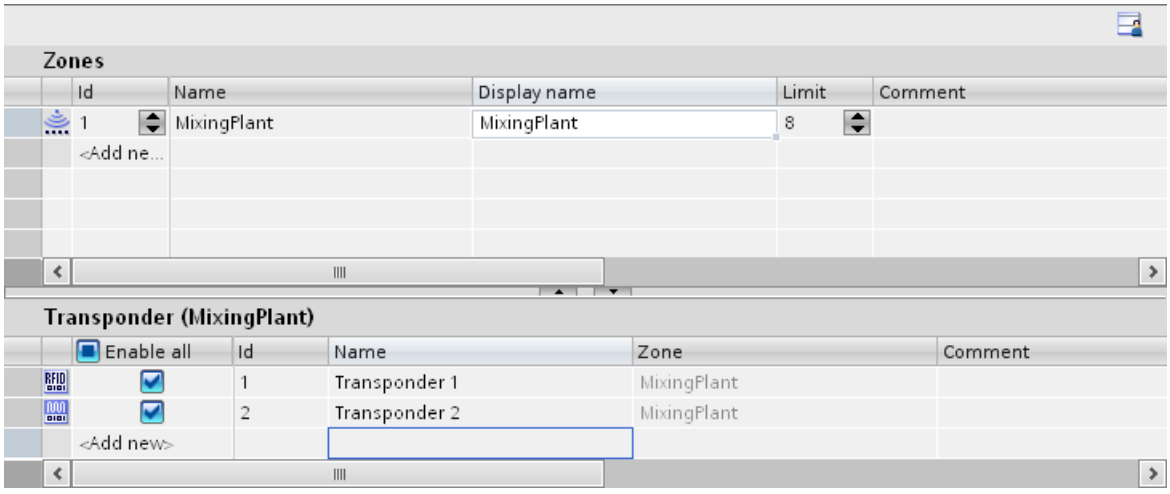

The display name designates the zone in the set editing language. An overview of the display names can be found in the "Project texts" editor.

10.4.2 Displaying the screen on entering a zone (Panels, Comfort Panels)

Introduction

Configure a screen change when entering and leaving the zone.

Requirements

- A Mobile Panel which supports zones has been created, for example, Mobile Panel 277 IWLAN.
- The "MixingPlant" zone is configured.
- The "Plant_1" screen has been created and opened.
- The "MixingPlant_1" screen has been created and opened.
- The "Zones" work area is open.

Procedure

1. Select the "MixingPlant" zone from the "Zones" table.
2. Click "Properties> Events" in the Inspector window.
3. Click the event "On entry".

4. Select the "ActivateScreen" system function in the "function list".
5. Select "MixingPlant_1" as the "Screen name".
6. Click the event "On exit".
7. Select the "ActivateScreen" system function in the function list.
8. Select "Plant_1" as the "Screen name".

Result

The "MixingPlant_1" screen is displayed on the HMI device when you enter the "MixingPlant" zone. When you exit the zone, the "Plant_1" screen appears once more.

10.4.3 Displaying an object in relation to the zone (Panels, Comfort Panels)

Validity

The following example is independent of the WLAN and applies to all Mobile Panels.

Introduction

You configure an I/O field that is only visible within a specific zone.

Requirements

- A Mobile Panel which supports zones has been created, for example, Mobile Panel 277 IWLAN.
- The "Zone_id" variable of the "Int" data type has been created.
- A "Plant_1_Overview" screen has been created.

Procedure

1. Open the "Runtime Settings > General" editor.
2. Select the "Zone_id" tag as the "ID Zone/ Effective range".
3. Open the screen "Plant_1_Overview".
4. Drag and drop the "I/O box" object from the toolbox into the screen.
5. Click "Properties > Animations" in the Inspector window.
6. Click "Layout > Visibility."
7. Select "Zone_id" as the "tag".
8. Select "from" 1 "to" 1 as the "Range".
9. Enable "Visible".

Result

The IO field is only visible at runtime under the following conditions:

- The hard-wired Mobile Panel is connected to a connection box with ID = 1.
- The Mobile Panel Wireless is located in a zone with ID = 1, for example, the "MixingPlant" zone.

10.5 Working with zones on connection boxes (Panels, Comfort Panels)

10.5.1 Configuring the zone of a connection box (Panels, Comfort Panels)

Validity

The following section applies to all KTP Mobile Panels, for example, Mobile Panel KTP700.

Introduction

You will have to set up a zone in order to carry out plant-specific operator control and monitoring tasks.

Requirements

- A suitable editing language, e.g. English, must have been set in the "Project languages" editor.
- The "Zones" work area is open.

Procedure

1. Double-click "Add..." in the "Zones" table.
2. Enter the box ID of your connection box for "ID".
3. Enter the "MixingPlant" zone in the "Name" field.

4. Enter "MixingSystem" as the "Display name" for the zone at runtime.

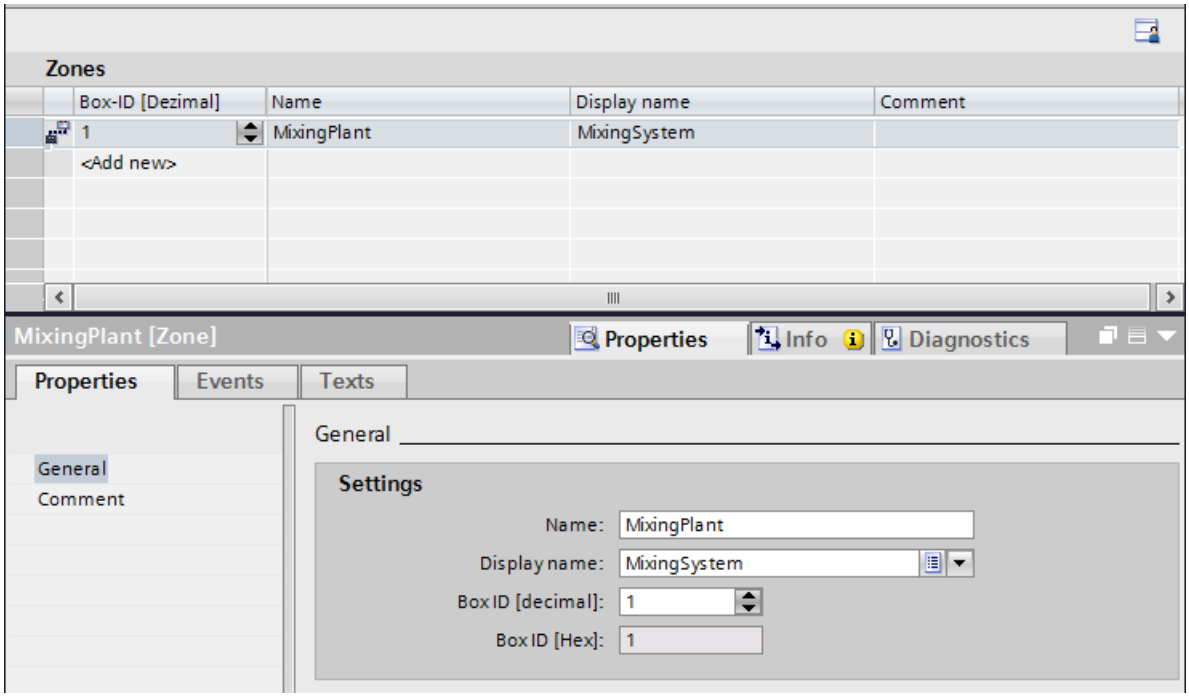

5. Repeat these steps for all required connection boxes in the project.

Result

The "MixingPlant" zone is configured. The range of the zone is defined by the length of the connecting cable of the Mobile Panel.

The display name designates the zone in the set editing language. An overview of the display names can be found in the "Project texts" editor.

10.5.2 Calling system functions when connected to the connection box. (Panels, Comfort Panels)

Introduction

When entering a zone you have the option of triggering various system functions, e.g. calling a configured pop-up screen or logging the current user in and out. In addition, you can use user-defined VB functions, which you can also select from the toolbar.

In the following example you configure the screen that is displayed when a Mobile Panel is connected to a connection box.

Requirements

- A Mobile Panel which supports zones has been created, for example, Mobile Panel KTP700.
- The "MixingPlant" zone is configured.
- The "Plant_1" screen has been created and opened.
- The "MixingPlant_1" screen has been created and opened.
- The "Zones" work area is open.

Procedure

1. Select the "MixingPlant" zone from the "Zones" table.
2. Click "Properties> Events" in the Inspector window.
3. Click on the "Connected" event.
4. Select the "ActivateScreen" system function in the "function list".
5. Select "MixingPlant_1" as "Screen name".

Result

When you connect your Mobile Panel to the connection box of the "MixingPlant" zone, the "MixingPlant_1" screen appears on the display of the HMI device.

10.6 Working with effective ranges (transponders) (Panels, Comfort Panels)

10.6.1 Overview (Panels, Comfort Panels)

Introduction

The following configuration guide describes the steps needed to set up an effective range for fail-safe operation on a Mobile Panel Wireless.

Procedures overview

1. Configuring the effective range:
   You configure the "MixingAxisControl" effective range as a conical area around "Transponder1 " at a distance of 5 meters.
2. Configuring the name of the effective range:
   To ensure logon to the effective range, configure the object "Effective range - Name".
3. Configuring additional effective range objects:
   Configure additional objects for displaying the position and signal strength within an effective range.
4. Set parameters for loading:
   - PROFIsafe communication
   - WLAN network
   - Power management
   - Interface
5. Configure the data channel
6. Configure network operation
7. Set the transponder
8. Commissioning effective ranges
9. Switching on and Testing the HMI device
10. Start loading manually
11. Acknowledging the effective range at the plant:
    You acknowledge the effective ranges and their transponders.
12. Determine the checksum

13. Load the project once again with checksum:
    Enter the checksum you have determined in the project and load the project once again.
14. Test effective range

Note

For detailed information on items 5 to 14, refer to the operating instructions of the HMI device.

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

10.6.2 Configuring an effective range (transponder) (Panels, Comfort Panels)

Introduction

You set up an effective range in order to control safety related operator input upstream of a machine or plant.

Requirements

- A suitable editing language, e.g. English, must have been set in the "Project languages" editor.
- The "Zones & effective ranges" work area is open.
- The "Effective ranges" tab is open.

Procedure

1. Double-click "Add..." in the "Effective Ranges" table.
2. Enter "MixingAxisControl" as the "Name" of the effective range.
3. Enter "Mixing_axis_PLC" as the "Display name" for the effective range at runtime.
4. Enter effective range "1" in the "ID" field.
   Make sure that the effective range ID matches the ID on the machine.
5. Enter "5" meters as the "Limit" of the effective range.
6. Double-click "Add..." in the "Transponders" table.
7. Enter "Transponder 1" as the "Name" of the transponder.
8. Enter transponder "1" in the "ID" field.
   Then set transponder IDs on the transponder.
9. Enable the assignment of "Transponder 1".
   "Transponder 1" is assigned to the "MixingAxisControl" effective range.

Result

You configure the "MixingAxisControl" effective range as a conical area around "Transponder 1" at a distance of 5 meters, regardless of the zones.

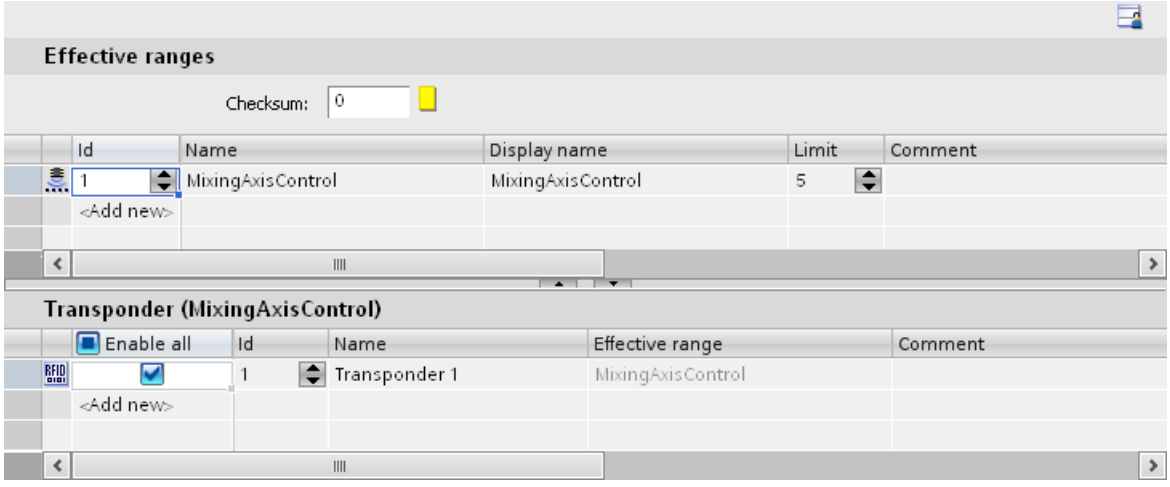

The display name designates the effective range in the set editing language. An overview of the display names can be found in the "Project texts" editor.

10.6.3 Configuring the effective range name (Panels, Comfort Panels)

Introduction

Configure at least the "Effective range - Name" object so that an operator can log on and off a machine using an effective range. If you do not configure an effective range object for a Mobile Panel Wireless for fail-safe operation, a warning appears during generation.

Requirements

A screen has been created and opened.

Procedure

1. Select the "Effective range name" object from the "Elements" category in the toolbox.
2. Drag and drop the "Effective range - Name" object from the toolbox into the screen.

Result

If the operator clicks an object during runtime, he initiates logon or logoff of the HMI device to a machine that is assigned to an effective range.

10.6.4 Configuring additional Mobile Wireless objects (Panels, Comfort Panels)

Introduction

Configure other objects such as the charge condition or WLAN reception in order to display further information about the effective range.

Requirements

- A screen has been created and opened.
- The effective range name is configured in the screen.

Procedure

1. Select the "Effective range - Signal" object from the "Elements" category in the toolbox.
2. Drag and drop the "Effective range - Signal" object from the toolbox into the screen.
3. Repeat steps 1 and 2 with the "WLAN - Reception" object.

Result

The "Effective range - Signal" object indicates how accurately the Mobile Panel Wireless is within an effective range. The "WLAN - Reception Quality" object displays the signal strength of the WLAN wireless connection.

See also

WLAN reception (Page 512)

Effective range signal (Page 511)

Effective range name (Page 507)

10.7 Working with effective ranges (RFID) (Panels, Comfort Panels)

10.7.1 Overview (Panels, Comfort Panels)

Validity

The next section applies only to Mobile Panels Wireless with RFID, for example, Mobile Panel 277F IWLAN (RFID Tag).

Introduction

The following configuration guide describes the steps required to set up an effective range (RFID) for fail-safe operation on a Mobile Panel Wireless with RFID.

Procedures overview

1. Configure effective range (RFID):
   You configure the effective range (RFID) "MixingAxisControl".
2. Configuring the name of the effective range (RFID):
   To ensure logon to the effective range, configure the object "Effective range - Name (RFID)".
3. Set parameters for loading
4. Configure the data channel
5. Configure network operation
6. Install RFID Tag in the plant
7. Turn on the HMI device and enable loading mode
8. Loading a project
9. Write the suitable IDs to the RFID Tags in the plant
10. Determine the checksum
11. Load the project once again with checksum:
    Enter the checksum you have determined in the project and load the project once again.
12. Test effective range (RFID)

Note

For detailed information on items 4 to 12, refer to the operating instructions of the HMI device.

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

10.7.2 Configuring effective range (RFID) (Panels, Comfort Panels)

Introduction

You set up an effective range (RFID) to control safety-related operator input upstream of a machine or plant.

Requirements

- A suitable editing language, e.g. English, must have been set in the "Project languages" editor.
- The "Effective ranges (RFID)" work area is open.

Procedure

1. Double-click "Add..." in the "Effective Ranges (RFID)" table.
2. Enter "MixingAxisControl" as the "Name" of the effective range (RFID).
3. Enter "MixingAxisControl" in the "Display name" field of the effective range (RFID).
4. Enter "1" in the "ID" field of the effective range (RFID).
   Make sure that the effective range (RFID)-ID matches the ID on the machine.
5. Double-click "Add..." in the "RFID Tag table.
6. Enter "RFID Tag 1" as the "Name" of the RFID Tag.
7. Enter "1" as the "ID" of the RFID Tag. Then set the RFID Tag-ID for the RFID Tag.
8. Enable the assignment of "RFID Tag 1".
   "RFID Tag 1" is assigned to the effective range (RFID) "MixingAxisControl".

Result

The effective range (RFID) "MixingAxisControl" is configured.

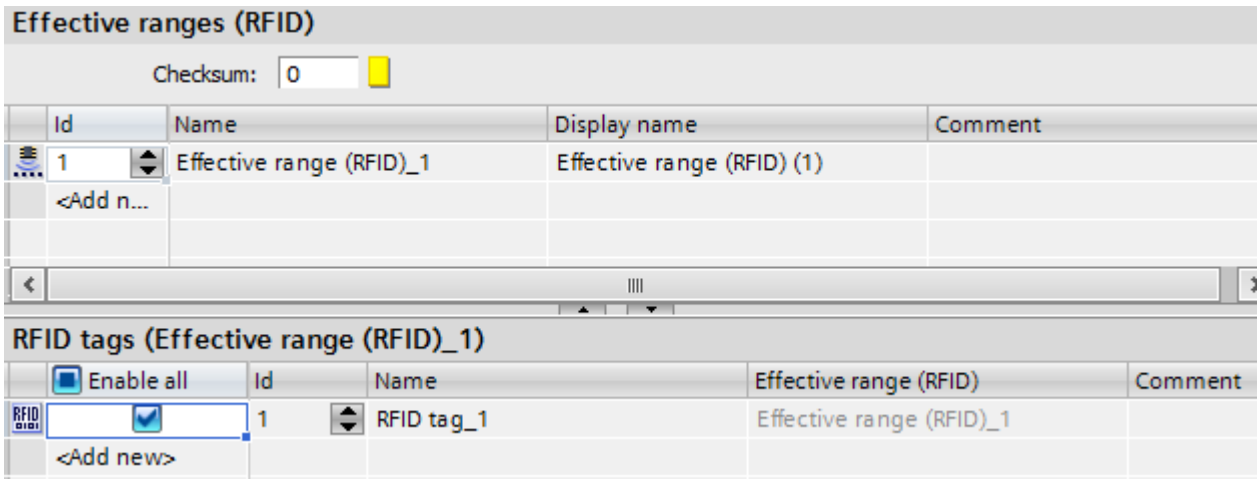

The display name describes the effective range (RFID) in the set editing language. An overview of the display names can be found in the "Project texts" editor.

10.7.3 Configuring the effective range name (RFID) (Panels, Comfort Panels)

Introduction

Configure the "Effective range - Name" object so that an operator can log on and off a machine using an effective range (RFID). If you do not configure the "Effective range-Name" object for a Mobile Panel Wireless RFID with (RFID), a warning will appear during generation.

Requirements

A screen has been created and opened.

Procedure

1. Select the "Effective range - Name (RFID)" object from "<Elements" in the toolbox.
2. Drag and drop the "Effective range - Name (RFID)" object from the toolbox into the screen.

Result

If the operator clicks an object during runtime, he will initiate logon or logoff of the HMI device to a machine which has been assigned to an effective range (RFID).

10.8 Reference (Panels, Comfort Panels)

10.8.1 PROFIsafe address (Panels, Comfort Panels)

Validity

The PROFIsafe address is available for Mobile Panels Wireless and KTP F-devices.

Note

Additional information on fail-safe operation can be found in the operating instructions for the HMI device:

Complete documentation for Mobile Wireless (http://support.automation.siemens.com/WW/view/en/26268960)

PROFIsafe address

The PROFIsafe address used for fail-safe I/Os in PROFIsafe communication is unique on the entire network and on all stations. On the Mobile Panel Wireless, you create a default value in the project and load it to the HMI device. You can also store a PROFIsafe address on the HMI device.

The PROFIsafe address fall within the range from 1 to 65534. The two PROFIsafe addresses (in the Engineering System and on the HMI device) are checked for formal validity.

The PROFIsafe address is loaded as follows at the start of Runtime:

- You programmed a valid PROFIsafe address in the Control Panel.
  The HMI loads the PROFIsafe address set in the Control Panel.

10.8.2 Power Management (Panels, Comfort Panels)

Validity

In the "Runtime Settings > Power Management" editor, configure the energy-saving mode of a Mobile Panel.

Power Management

The Mobile Panel Wireless can save energy in the following ways:

- Reduce brightness - some savings
- Switch off screen - high savings

You can also indicate after how long without operator input the power management function is enabled. The communication link is maintained. The time for "Switch off screen" is longer than the time for "Reduce brightness".

10.8.3 Zone ID / connection point ID (Panels, Comfort Panels)

Validity

In the "Runtime Settings > General" configure an internal tag which, depending on the value, shows specific information on the display of the HMI device.

Note

Mobile Panels Wireless with RFID

For Mobile Panels Wireless with RFID, the area "Zone ID / Connection point ID" is not available.

Zone ID / connection point ID

This field is used to configure an internal tag that contains the following value at runtime in relation to the type of HMI device:

- The ID of the connection box or effective range to which the hard-wired Mobile Panel is connected.
- The ID of the zone in which the Mobile Panel Wireless is located. If the value is set to 255, then the HMI device is not in any zone.

Planning tasks 11

11.1 Basics

11.1.1 Field of application of the Scheduler (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Definition

You can use the Scheduler to configure tasks to run independent of the screen in the background. You create tasks by linking system functions or scripts to a trigger. The linked functions will be called when the triggering event occurs.

Example of an application

The Scheduler is used to execute event-controlled tasks automatically. For example, you use a task to automate the following:

- Regular swap out of log data
- Printout of an alarm report when an alarm buffer overflow occurs
- Printout of a report at shift end
- Monitoring a tag
- Monitoring of a user change

Note

The availability of the listed examples is determined by the HMI device.

11.1.2 Working with tasks and triggers (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

A task consists of a trigger and a task type.

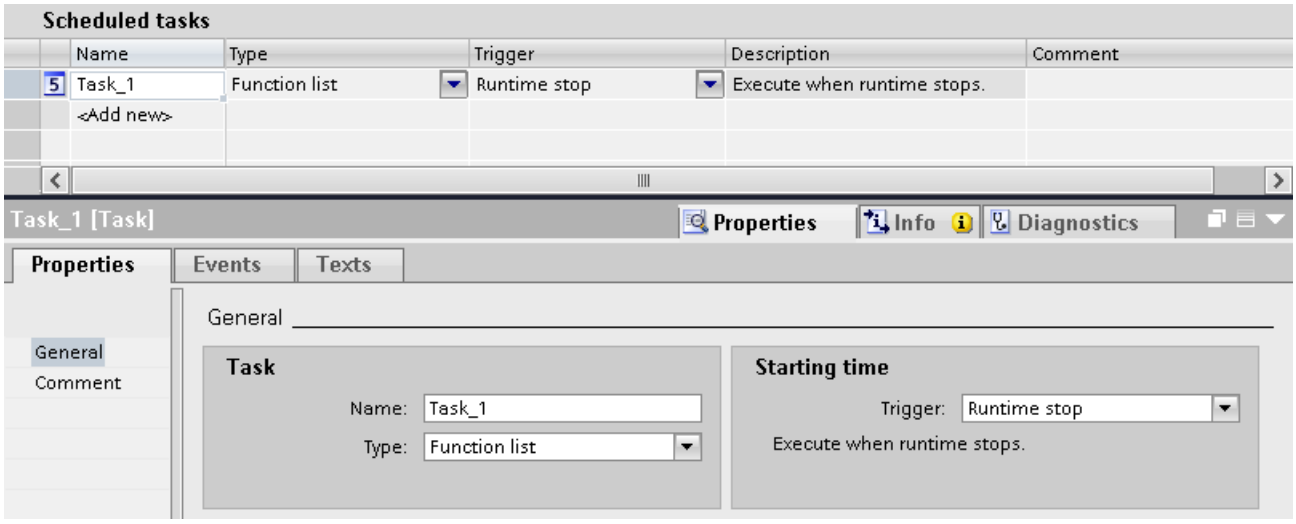

Starting a task

Controlled by a trigger, the Scheduler starts the task linked to the trigger.

11.1.3 Work area of the "Scheduler" editor (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Double-click on "Scheduler" to open it in the project view. The work area shows the scheduled tasks, which consist of the trigger and the task type, for example, the function list.

Structure

The work area consists of the table of jobs.

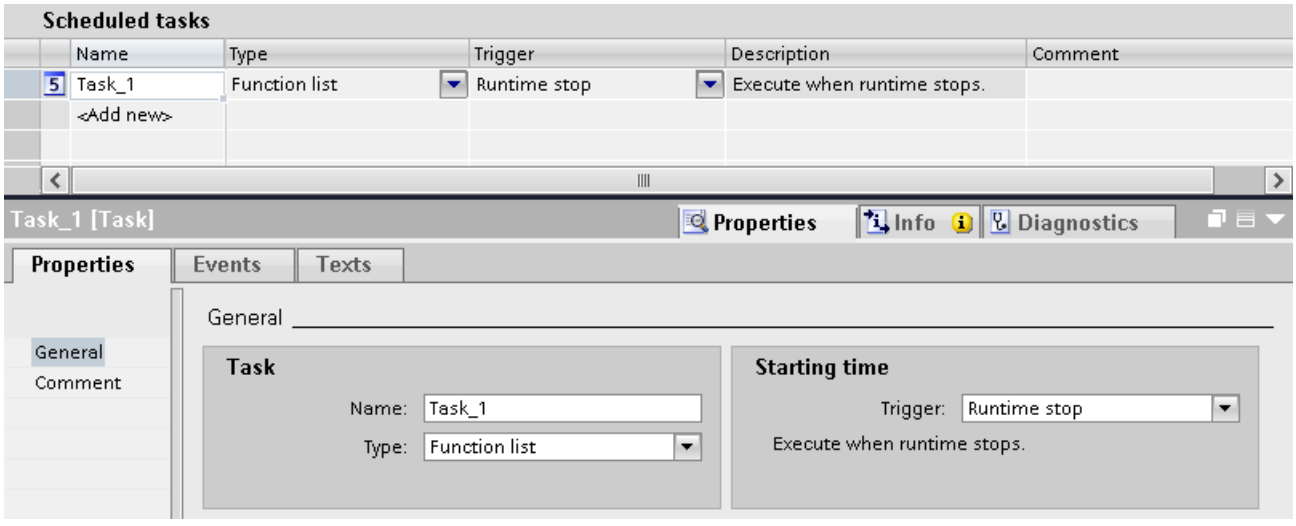

The table of tasks shows specified tasks with their properties, such as triggers. You select a task type and a trigger. You assign a name and a comment to the task. The description provides a written summary of the task including the timing for the task.

Inspector window

The "Properties" tab of the Inspector window is split into two parts.

The "Job" area lists the name of the job and the job type. The "Starting time" area shows the trigger. The area is different depending on the trigger selected.

In the "Events" tab use the function list with system functions that will be executed in the task.

Note

You can obtain more detailed information about the elements of the user interface using the tooltips. To do so, move the mouse pointer to the relevant object or press <F1> if the object has already been selected.

11.1.4 Triggers (Basic Panels)

Introduction

A trigger is linked to a task and forms the triggering event which will call this task. The task is executed when the trigger occurs.

Event trigger

When a task is linked to a system event, the task will be triggered by the event. System events include, for example, Runtime stop, screen change, user change, etc.

Each system event can only be configured once for each HMI device.

Deactivating job

If you do not need a certain job temporarily, deactivate the job in the Engineering System. You also use the trigger "Deactivated" to make a previously configured system event available once again.

Example: Task "A" is planned with the system event "Shutdown". This system event is then no longer available for another task "B". Select "Disabled" as the trigger for task "A" to make the "Runtime stop" system event available again.

Note

The available triggers depend on the HMI device.

11.1.5 Triggers (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

A trigger is linked to a task and forms the triggering event which will call this task. The task is executed when the trigger occurs.

Types of triggers

The Scheduler recognizes different types of triggers:

Acyclic triggers

They consist of a specification for the date and time of day. Tasks with an acyclic trigger are performed once by the Scheduler on the specified date and at the specified time.

Cyclic triggers

Tasks linked to a cyclic trigger occur regularly at a specific point in time.

- Standard cycle: With a standard cycle, the beginning of the first interval coincides with the start of Runtime. The length of the interval is determined by the cycle.
- Trigger with defined starting time: You specify a start time and a time interval. You can use cyclic triggers to perform tasks on a daily or weekly basis, for example. For example, you can select a "Weekly" trigger to specify a week as the interval. You specify the beginning of the interval with the day of the week and the time of day.

Note

User-defined cycles

You have defined your own cycles. You can only use cycles in the Scheduler whose maximum value amounts to one hour.

Event trigger

When a task is linked to a system event, the task will be triggered by the event. In contrast to cyclic triggers, system events usually occur acyclically. A Runtime stop is a system event, for example.

Each system event can only be configured once for each HMI device.

Deactivating job

If you do not need a certain job temporarily, deactivate the job in the Engineering System. You also use the trigger "Deactivated" to make a previously configured system event available once again.

Example: Task "A" is planned with the "User change" system event. This system event is then no longer available for another task "B". Select "Disabled" as the trigger for task "A" to make the "User change" system event available again.

Note

The available triggers depend on the HMI device.

11.1.6 Timers for cyclic triggers (RT Advanced, Panels, Comfort Panels)

Timers for cyclic and acyclic triggers

You have the option to change the start time of a task dynamically in Runtime. You can select an HMI variable as the timer. The value of the tag determines the start time for the task during Runtime.

Note

An HMI tag must be of the type "DateTime". A PLC tag must be of the type "Date and Time" or "DTL".

As long as the operator does not change the tag after Runtime starts, the start time configured in the task is valid. As soon as the operator enters a change to the tag, the current value of the tag is always the start time. To reset the configured time as the start time, the operator has to enter the configured time again.

11.1.7 Tag triggers (RT Professional)

Tag triggers

In addition to acyclic, cyclic and event triggers, WinCC offers tag triggers.

Tag triggers consist of the specification of one or more tags. A task linked to such a trigger is only performed when the value of the tag changes. This reduces the load on the system. The cycle in which the tag values are scanned can be set individually for each tag. Select a value ranging from "250 ms" to once an hour for the scanned cycles. You can also define your own user cycles.

Rules for selecting the trigger

Use the tag trigger instead of the cyclic trigger for tag processing. This reduces the load on the system.

- Tag triggers
  The task is performed only when the value of the tag changes.
  = task is executed.

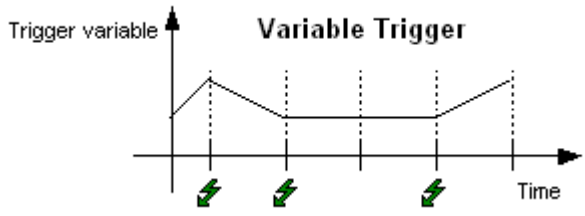

- Cyclic triggers
  You have selected "10 seconds" as the cyclic trigger. The task will be executed when the trigger occurs.
  = task is executed.

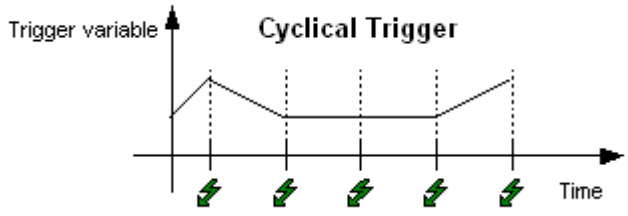

11.1.8 Function list (Basic Panels)

Function list

A trigger starts the function list. The function list is executed line by line. Each line contains a system function. You can configure exactly one function list for each task.

Note

The choice of configurable system functions in a function list depends on the selected trigger and the HMI device.

See also

Integrating local VB scripts (Page 1801)

Integrating local C scripts (Page 1802)

11.1.9 Function list (RT Professional, RT Advanced, Panels, Comfort Panels)

Function list

A trigger starts the function list. The function list is executed line by line. Each line contains a system function or a local script. You can configure exactly one function list for each task.

Note

The choice of configurable system functions in a function list depends on the selected trigger and the HMI device.

Processing of system functions and scripts

System functions and scripts in a function list are processed in Runtime sequentially from top to bottom.

Use a script with loops, conditional statements and abort conditions to program non-sequential and conditional procedures.

11.1.10 Print task (RT Professional)

Introduction

In addition to the "Function list", WinCC Runtime Professional also includes the "Print job" task type.

Print job

You print the report by creating a print job. When a cyclic or acyclic trigger occurs, the Scheduler starts the printout of a report. You define the scope of printing and the time range of the report beforehand in the "Print jobs" editor. In addition, you specify whether the report will be printed or written into a file.

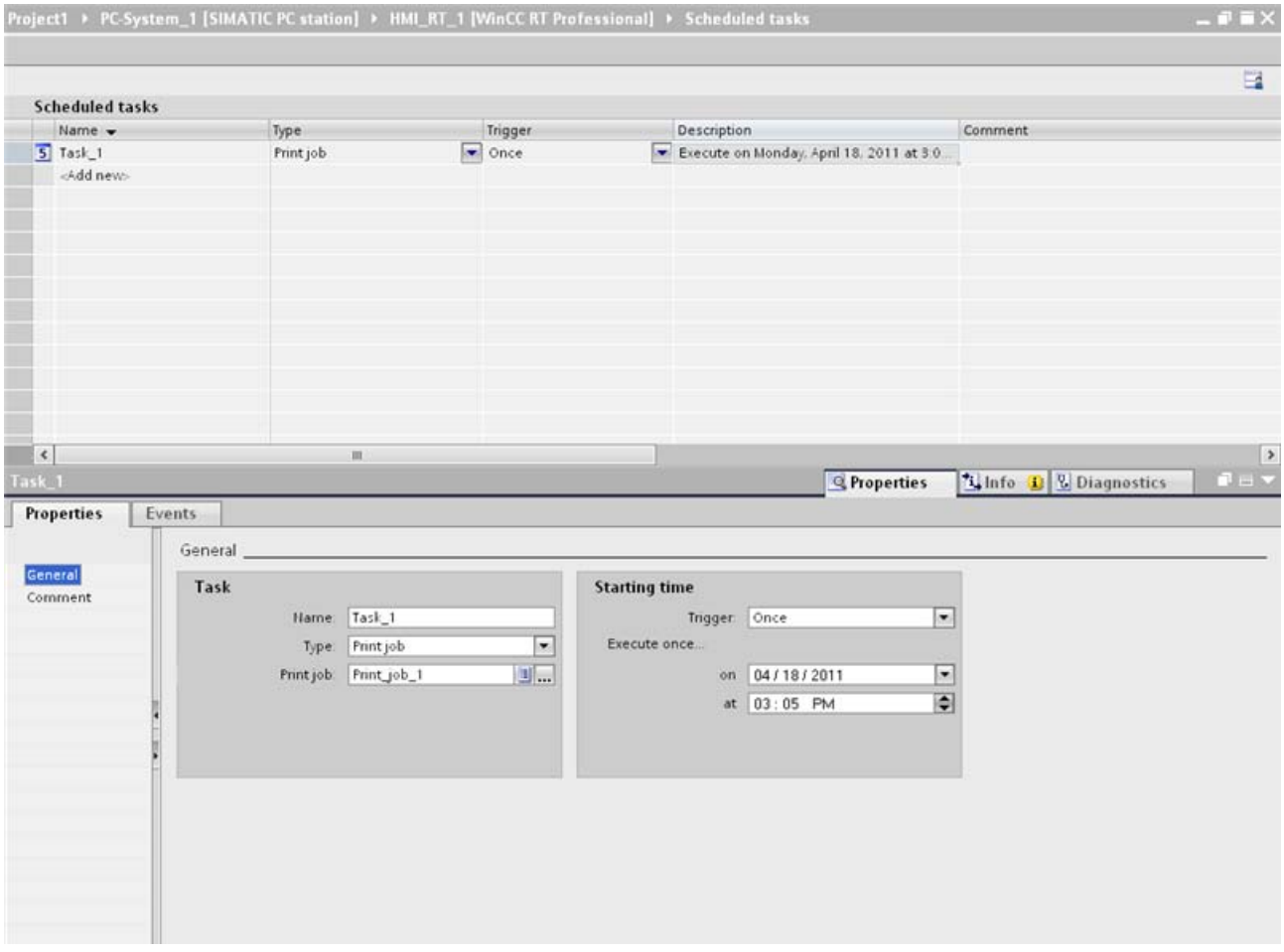

If you have selected "Print job" as the task type, an additional field will appear in the "Task" area. Select the predefined print job for the task in the "Print job" field.

Note

Activate "Scheduled print jobs in runtime" in the start sequence of WinCC Runtime to use the "Print job" job type.

Click on "Runtime settings > Services > Scheduled print jobs in runtime" in the editor.

11.1.11 Integrating local VB scripts (RT Professional)

Introduction

In addition to the "Function list" and "Print job" task types, Professional Runtime also allows you to integrate local VB scripts.

Integrating local VB scripts

You can create and edit local scripts. You can also convert the system functions offered in the function list in a local script and edit the code. The local VB script is stored with the task.

A trigger starts the script in Runtime. Once you have converted a function list to a VB script, you can no longer access the function list for this task. You will have to delete the VB script to access the function list once again.

The Scheduler allows you to start VB scripts independent of screens. The VB scripts give you access to tags and screens in Runtime.

There is no common data area between scripts in the Scheduler and screens. Scripts in the Scheduler and screens can be synchronized with the DataSet object or tags. The data type of VBS variables is always VARIANT.

The local VB script is not executed simultaneously. The script last triggered is put into a queue until it is processed.

Note

When you create a script, the "Option explicit" statement is entered automatically in the declaration section and cannot be deleted. The statement is necessary to avoid errors from incorrectly spelled tags without declaration.

The statement requires you to always define tags with the "Dim" statement in your code. Do not use the "Option explicit" statement in your code, as it can lead to Runtime errors.

Requirement

Enable "Tasks in Runtime" in the startup sequence of WinCC to use the function list and local scripts.

Click "Runtime settings > Services > Tasks in Runtime" in the editor.

Note

You cannot call C scripts in VB scripts and vice versa.

User cycles as triggers

If you use user cycles as triggers for VB scripts, the scripts are always executed in Runtime with a granularity of 250 ms.

See also

Function list (Page 1799)

11.1.12 Integrating local C scripts (RT Professional)

Introduction

In addition to the "Function list" and "Print job" task types, Runtime Professional also allows you to integrate local C scripts.

Integrating local C scripts

You can create and edit local scripts. You can also convert the system functions offered in the function list in a local C script and edit the code. The local C script is stored with the task.

A trigger starts the script in Runtime. Once you convert a function list to a C-script, you can no longer access the function list for this task. You will have to delete the C script to access the function list once again.

A trigger starts the local C script in Runtime.

The local C script is not executed simultaneously. The script last triggered is put into a queue until it is processed.

Requirement

Enable "Tasks in Runtime" in the startup sequence of WinCC to use the function list and local scripts.

Click "Runtime settings > Services > Tasks in Runtime" in the editor.

Note

You cannot call C scripts in VB scripts and vice versa.

User cycles as triggers

If you use user cycles as triggers for C scripts, the scripts are always executed in Runtime with a granularity of 250 ms.

See also

Function list (Page 1799)

11.2 Planning tasks

11.2.1 Planning tasks with acyclic triggers (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You are planning a one-time Runtime stop for maintenance work. The task starts a function which requires Runtime to stop.

Requirements

- The "Scheduler" editor is open.
- The Inspector window is open.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Maintenance at end of year" as the "Name."
3. Select "Once" as the "Trigger."
4. Select "31.12.2008" in the "Properties > Properties > General > on" field of the Inspector window.
5. Enter the time "18:00" in the "at" field.
6. Click "Properties > Events" in the Inspector window.

7. Select the "StopRuntime" system function in the function list.
8. Select "Runtime" as the "Mode".

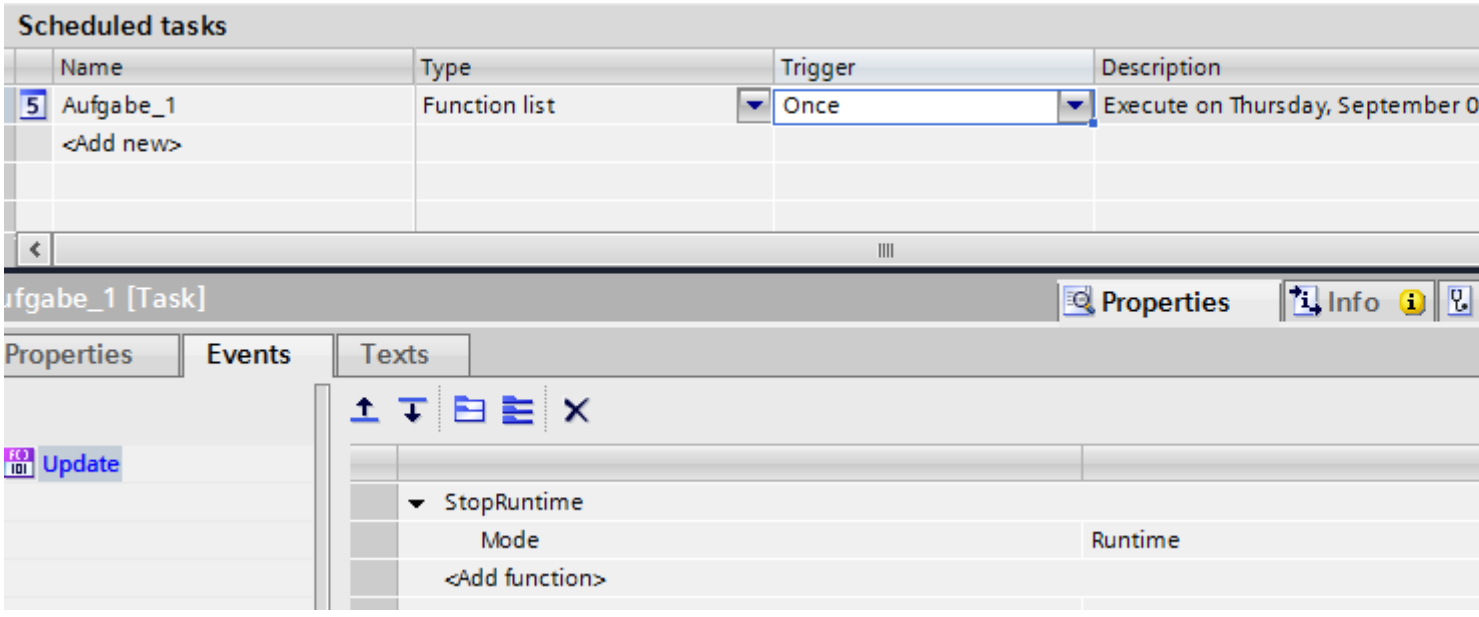

Result

The task is executed once. Runtime will be stopped at 18:00 on December 31, 2008.

Note

"StopRuntime" event on RT Professional

Please note that the "StopRuntime" event means runtime is in the process of ending and that some functionalities are therefore no longer available.

11.2.2 Planning tasks with cyclic triggers (RT Professional)

Introduction

You plan a task that starts the daily printout of a report.

Requirements

- You have created an HMI device with Runtime Professional.
- The "Daily alarm report" print task has been created.
- The "Scheduler" work area is open.

Procedure

1. Click "Add..." in the table of the task area.
2. Select "Print task" under "Type."
3. Enter "Daily alarm report printout" as the "Name."
4. Select "Daily" as the "Trigger."
5. In the Inspector window "Properties >Properties > General > at" enter the time "18:00".
6. Select the "Daily alarm report" print task in the "Task" area.

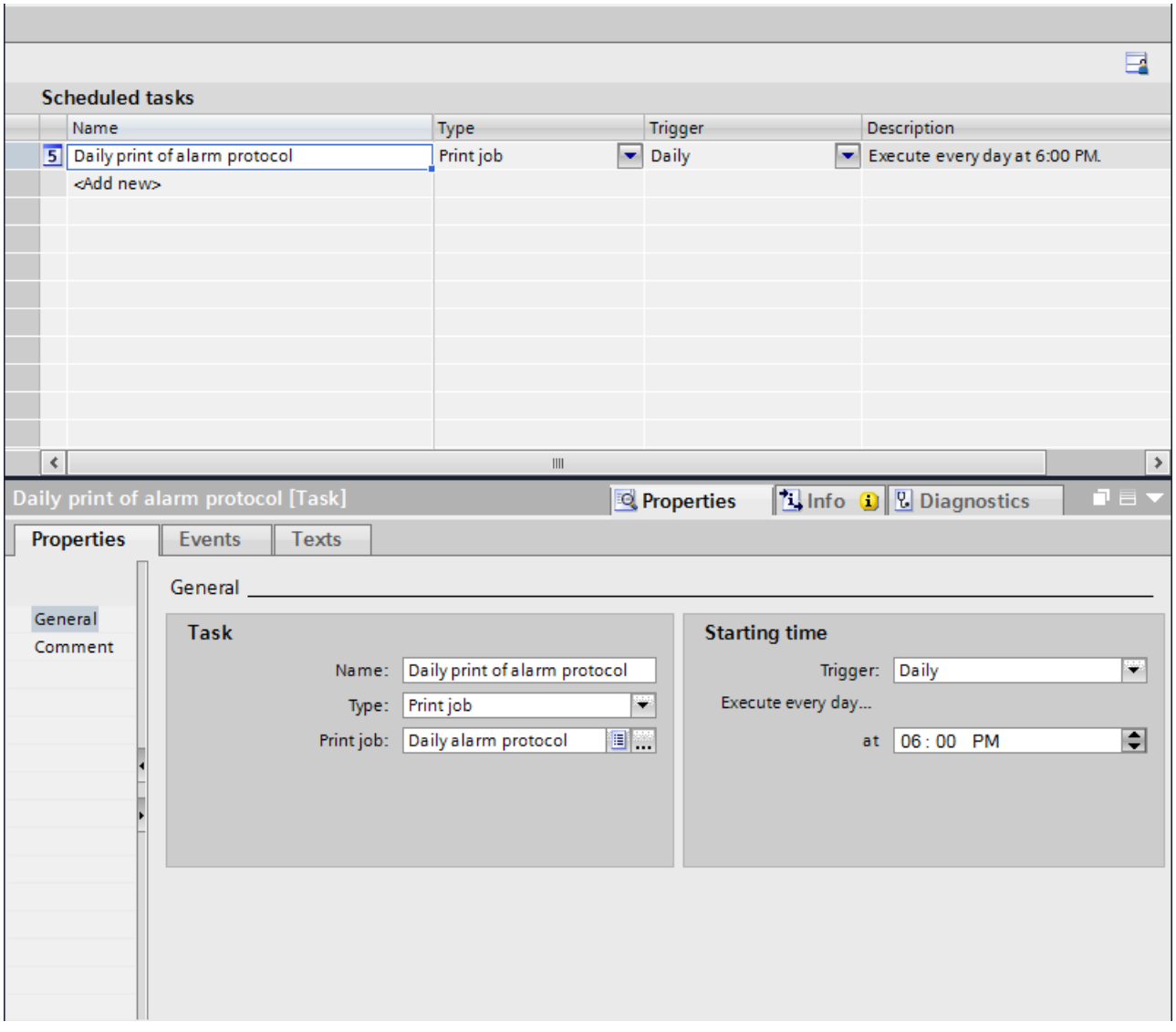

7. Open the "Runtime settings > Services" editor.
8. Enable "Start sequence of WinCC Runtime > Planned print jobs in Runtime".

Result

The report "Daily alarm report" will be printed every day at 18:00 hours.

11.2.3 Planning tasks with event triggers (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You plan a task that generates a screen change when the user changes.

Requirements

- The "Scheduler" work area is open.
- You have created the "Start" screen.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Screen change at user change" as the "Name."
3. Select "User change" as the "Trigger."
4. In the Inspector window, select "Properties > Events".

5. Select the "Screen/ActivateScreen" system function in the function list.
6. Select the "Start" screen in the screen name field.

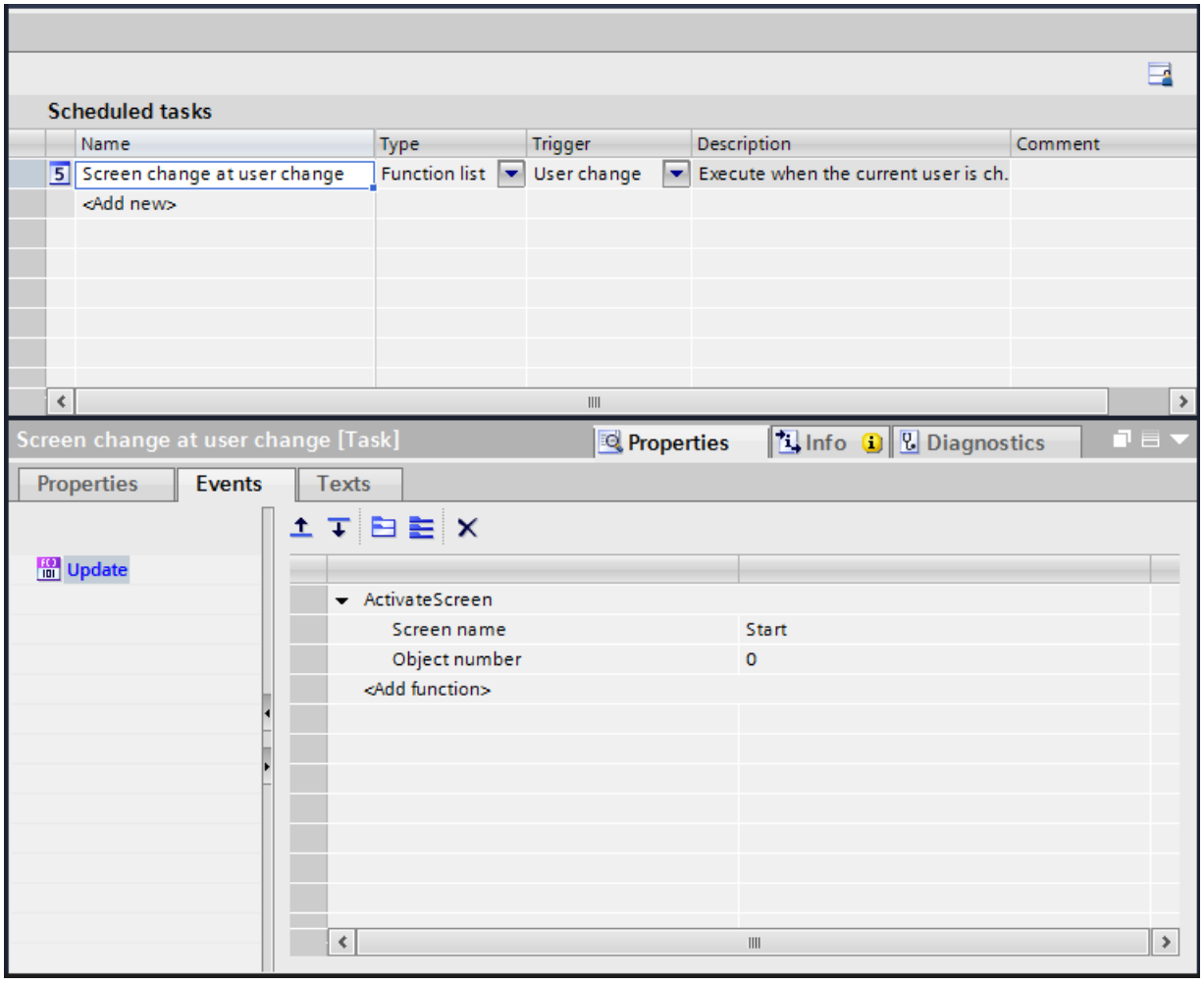

Result

The task is executed with the "User change" event. When a new user logs on successfully, the "Start" screen is called up.

11.2.4 Administer task (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Changing the designation

1. In the work area, double-click the field in the "Name" column.
2. Change the name of the task.
3. Confirm your entry with <Return>.

As an alternative, you can change the designation in the "Job > Name" box of the Inspector window.

Changing triggers

1. In the work area, mark the field in the "Trigger" column.
2. Open the drop-down list using the button and select the trigger you require.

You can alternatively change the trigger in the "Starting time" area of the Inspector window.

Deleting a task

1. Select one or more lines for the tasks to be deleted.
2. Open the shortcut menu with the right button and select the menu command "Delete".

As an alternative, delete one or more tasks by using the <Del> button.

11.3 Examples for Runtime Professional (RT Professional)

11.3.1 Example: Activate the start screen daily (RT Professional)

Introduction

You plan a task that activates the start screen daily with a logon dialog before the shift begins.

Requirements

- A "start screen" has been created.
- The "Scheduler" work area is open.

Procedure

1. Click "Add..." in the "Task" table.
2. Enter "Activate start screen" as the "Name".
3. Select "Function list" as the "Type".
4. Select "Daily" as the "Trigger."
5. In the Inspector window "Properties >Properties > General > at" enter the time "05:30".
6. In the Inspector window, open "Properties > Events".
7. Select the system function "ActivateScreen" from the "Screens" group in the function list.

8. Enter "Start screen" as the "Screen name".

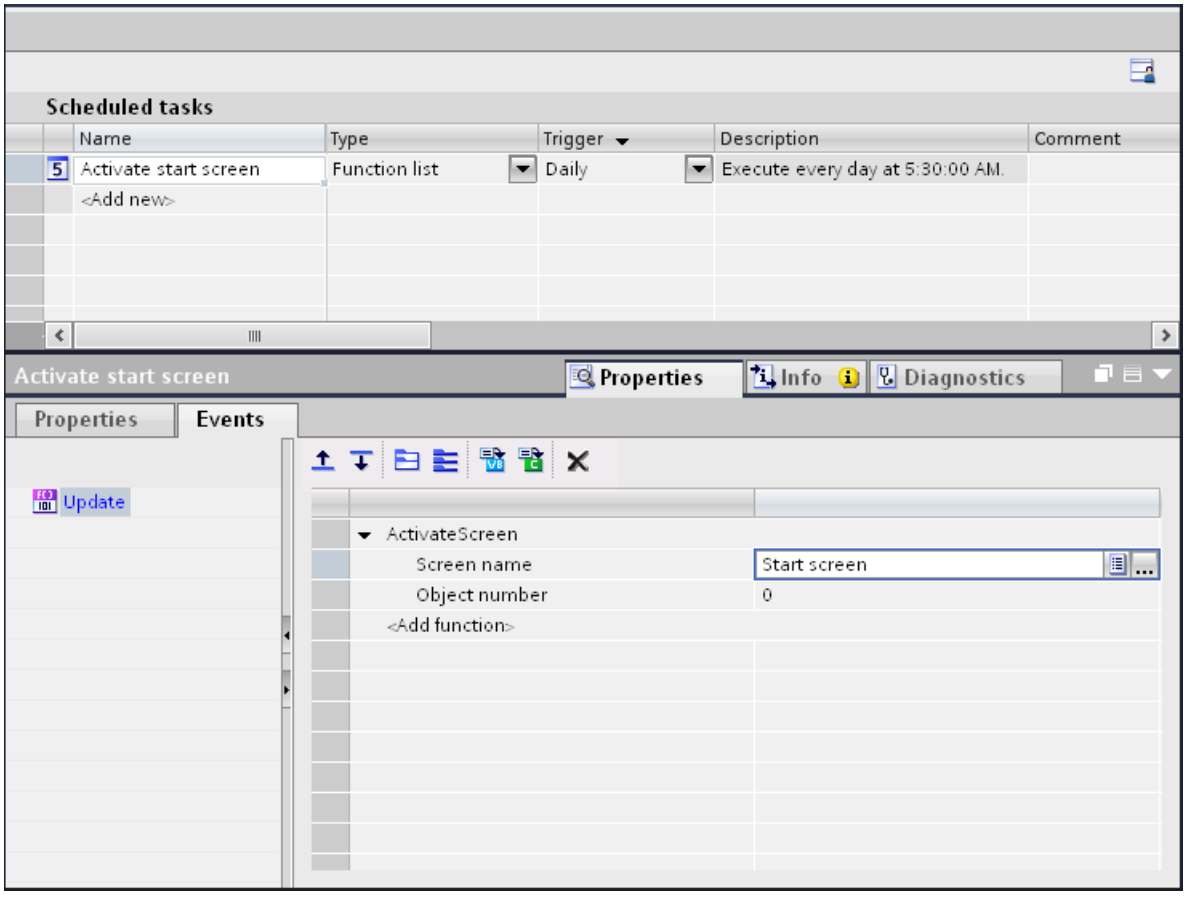

9. Open the "Runtime settings > Services" editor.
10. Activate "Scheduled tasks in Runtime".

Result

The "Start screen" is activated daily at 5:30 a.m.

11.3.2 Example: Printing a report once (RT Professional)

Task

You plan a task that prints out a report once at the end of the year.

Requirement

- The "Annual overview" print task is created.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Annual summary" as the "Name".
3. Select "Print task" as the "Type".
4. Select "Once" as the "Trigger."
5. Select "31.12.2010" in the "Properties > Properties > General > on" field of the Inspector window.
6. Enter the time "22:00" in the "at" field.
7. Select the "Annual overview" print task in the "Task" area.

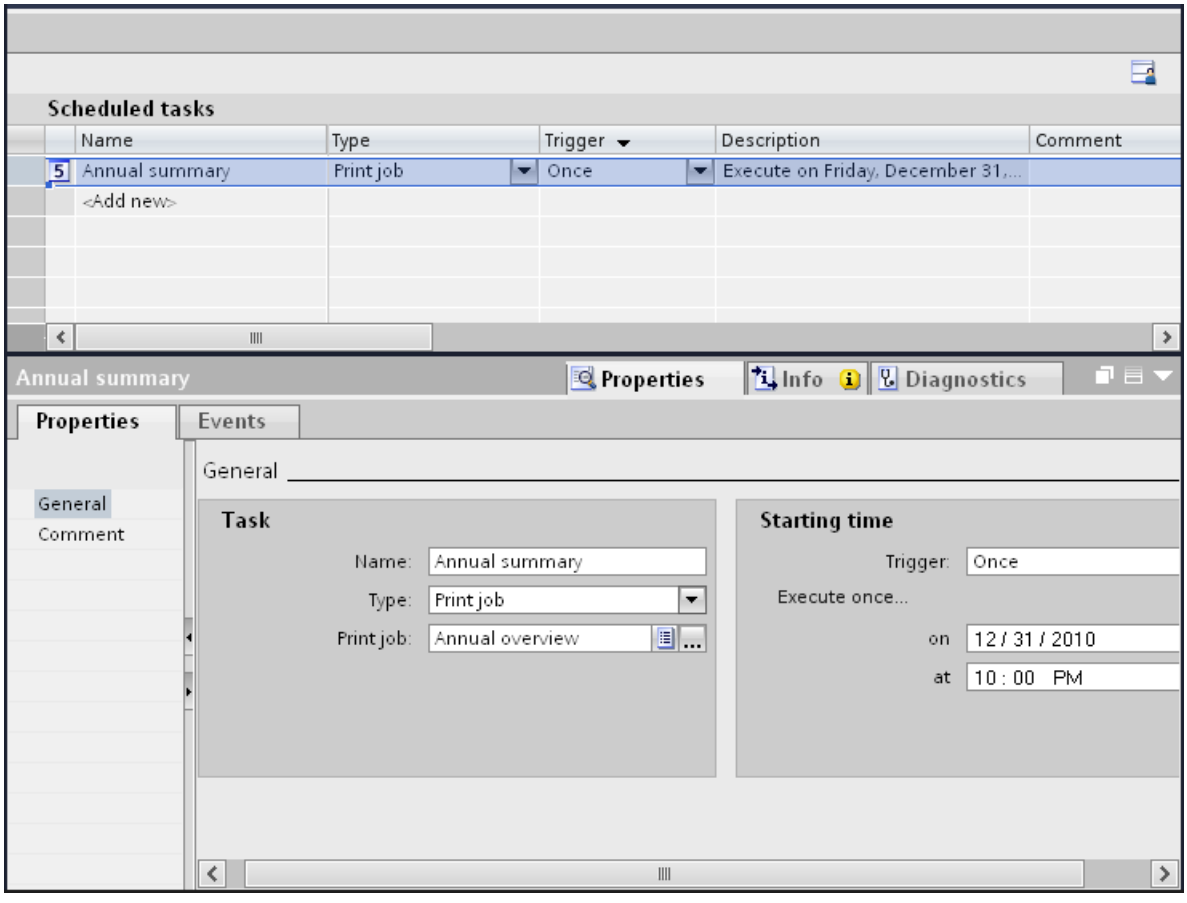

8. Open the "Runtime settings > Services" editor.
9. Enable "Start sequence of WinCC Runtime > Scheduled print jobs in Runtime".

Result

The "Annual overview" print job will be started on December 31, 2008 at 10:00 p.m.

11.3.3 Example: Reading the current user name (RT Professional)

Introduction

You plan a task that writes the current user into a text field.

Requirements

- A "Screen_1" screen has been created.
- A "Textfield_1" text field has been created in the "Screen_1" screen.
- The "Scheduler" work area is open.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Reading the user name" as the "Name".
3. Select "Tag trigger" as the "Trigger".
4. Click "Properties > Properties > General > Add trigger" in the Inspector window.
5. Select the "@CurrentUser" system variable.
6. Select "1 min" as the "Cycle".
7. In the Inspector window, open "Properties > Events".
8. Click in the "Function list" area.
9. Enter the following VB script:

```
Sub Action ()
    Dim user
    user = HMIRuntime.Tags("@CurrentUser").Read
    HMIRuntime.Screens("Screen_1").ScreenItems("Textfield_1").Text = user
End Sub
```

1. Open the "Runtime settings > Services" editor.
2. Activate "Scheduled tasks in Runtime".

Result

When the tag value changes, the current user is written into the "Textfield_1" text field.

11.3.4 Example: Increment a tag (RT Professional)

Introduction

You schedule a task that increments the value of a tag by one every hour.

Requirements

- You have created an HMI device with Runtime Professional.
- The "Scheduler" work area is open.
- The "Tag1" tag has been created.

Procedure

1. Click "Add..." in the table of the task area.
2. Select "Function list" as the "Type."
3. Enter "Count up tag" as the "Name."
4. Select "1 hour" as the "Trigger".
5. In the Inspector window, under "Properties > Properties > General > Starting time > at minute" enter the value "35:00".
6. In the Inspector window, open "Properties > Events".
7. Click in the "Function list" area.
8. Enter the following program code in the entry field:

```
Sub Action ()
    Dim objTag1
    Dim lngValue
    Set objTag1 = HMIRuntime.Tags("Tag1")
    lngValue = objTag1.Read
    objTag1.Write lngValue + 1
End Sub
```

1. Open the "Runtime settings > Services" editor.
2. Select "Scheduled tasks in Runtime".

Result

The tag is incremented cyclically.

11.4 Examples for Runtime Advanced and Panels

11.4.1 Example: Update user following change of user (RT Advanced, Basic Panels, Panels, Comfort Panels)

Task

Configure an I/O field which displays the logged on user. Configure a task which updates the I/O field when the logged on user changes.

Requirements

- A "CurrentUser" tag of the "String" type is created.
- A screen has been created and opened.
- An I/O field is created in the screen.

Procedure

1. Click on the "I/O field" object.
2. In the Inspector window, select "Properties > Properties > General":
   - Select "String" as the "Display format."
   - Select "CurrentUser" as the "Variable."
   - Select "Output" as the mode.
3. Change to the work area of the Scheduler.
4. Click "Add..." in the table of the task area.
5. Enter "CurrentUser" as the "Name".
6. Select "User change" as the "Trigger."
7. In the Inspector window, select "Properties > Events".

8. Select the system function "ReadUserName" from the "User Management" group in the function list.
9. Select "CurrentUser" as the "Variable."

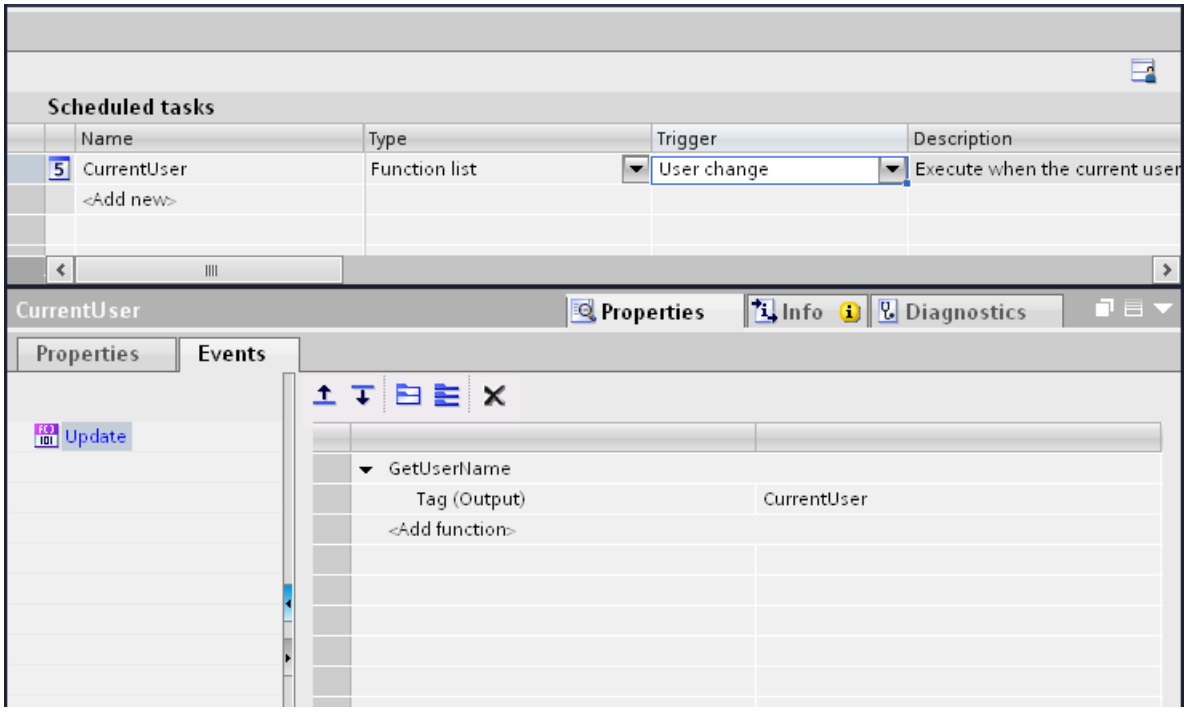

Result

When a new user logs on successfully, the "ReadUserName" function is called up. The "CurrentUser" tag is updated and displayed in the I/O field of the newly logged on user.

If a user does not log on successfully, the logged on user is logged off. The I/O field continues to display the user previously logged on until a new user logs on successfully.

11.4.2 Example: Weekly printout of a report (RT Advanced, Panels, Comfort Panels)

Introduction

You plan a task that starts the weekly printout of a report.

Requirements

- A "Weekly report" report has been created.
- The "Scheduler" work area is open.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Weekly report printout for end of shift" as the "Name."
3. Select "Weekly" as the "Trigger."
4. Select "Properties > Properties > General > on day" "Friday" in the Inspector window.
5. Enter the time "18:00" in the "at" field.

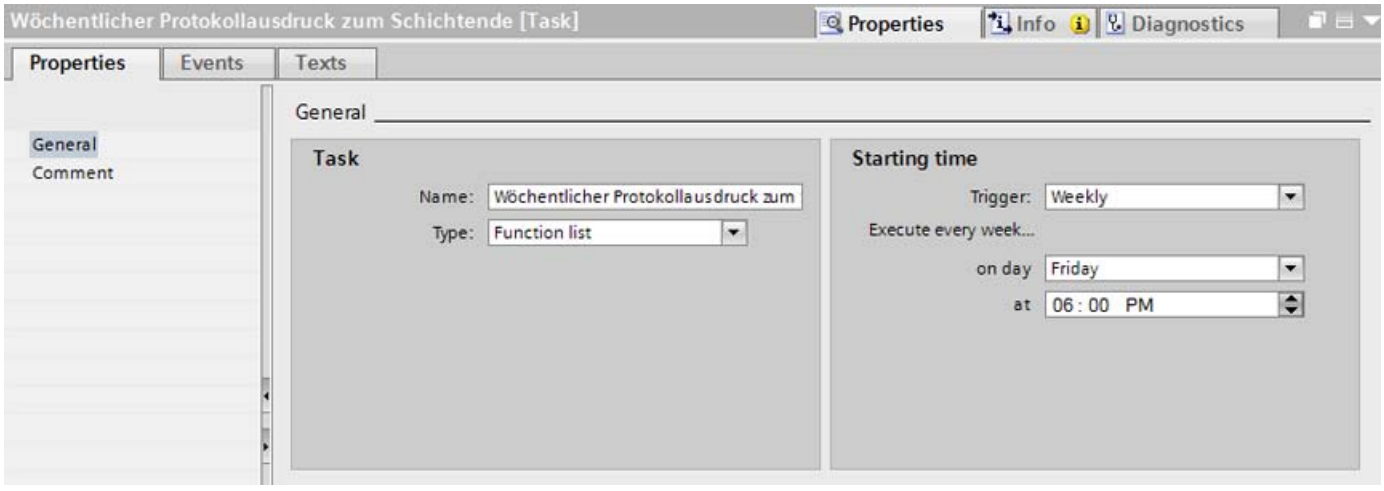

6. In the Inspector window, select "Properties > Events".
7. Select the "Print/PrintReport" system function in the function list.
8. Select "Weekly report" report.

Result

The weekly print task starts on Friday at 18:00.

11.4.3 Example: Changing the starting point of a task in Runtime (RT Advanced, Panels, Comfort Panels)

Introduction

A cyclic trigger, for example "Starting daily at 18:00," is configured in the Engineering System. The time "18:00" is fixed as a constant. To change the start time in Runtime, select a tag as the "Timer." The value of the tag determines the start time of the task during Runtime.

Task

You plan a task that swaps out log data daily and has the "LogTime" tag as the timer. You configure a "LogTime" tag in an input field.

Requirements

- The "DataLogSource" source log has been created as a data log.
- The "DataLogDestination" destination log has been created as a data log.
- A "ChangeLogTime" screen has been created.

Procedures overview

1. Creating a timer tag: You create a tag with which you change the starting point of the log data swap in Runtime.
2. Configuring a Date/time field: The operator changes the starting point of the task by using the Date/Time field in Runtime.
3. Configure the task with a timer tag: You create a task whose start time can be changed dynamically in Runtime.

Result

If nothing is entered in the "LogDataTimeField" input field, then the task starts daily at 18:00 by default. The log data are swapped out. If an operator enters 19:00 in the "LogDataTimeField" input field, then the log data are swapped out at 19:00. Provided the value of the tag is not changed again before the task starts.

11.4.4 Example: 1. Configuring a tag for Runtime (RT Advanced, Panels, Comfort Panels)

Task

In the following example you configure the "LogTime" variable.

Procedure

1. Open the "HMI tags" editor.
2. Click "Add..." in the table of the task area.
3. Enter "LogTime" as the "Name" of the variable.
4. Select "Internal variable" in the "Connection" column.
5. Select "DateTime" in the "Date type" column.

Result

A variable of the "DateTime" type has been created.

11.4.5 Example: 2. Configuring a Date/time field (RT Advanced, Panels, Comfort Panels)

Task

In the following example you configure a Date/time field. To change the starting point of the task in Runtime, link the "LogTime" variable to the Date/time field.

Requirements

- The "LogTime" variable of the "DateTime" type has been created.
- The "ChangeLogTime" screen is open.

Procedure

1. In the toolbox view, drag-and-drop a "Date/time field" from the "Basic objects" category to the screen.
2. In the Inspector window, click "Properties >Properties > General".
3. Disable the "System time."
4. Click "Variable" and select "LogTime."
5. Disable "Display date" and enable "Display time."
6. Select "Input/output" as the "Mode."
7. Click "Properties > Miscellaneous."
8. Enter "LogDataTimeField" as the "Name".

Result

The operator enters a time using the created date/time field.

11.4.6 Example: 3. Configuring archiving with a timer variable (RT Advanced, Panels, Comfort Panels)

Task

In the following example you configure a task with a timer tag with which you change the starting point of the task in Runtime.

Requirements

- The "LogTime" tag has been created.
- The "DataLogSource" source log has been created as a data log.

- The "DataLogDestination" destination log has been created as a data log.
- The Scheduler work area is open.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Swap out log daily at a tag time" as the "Name".
3. Select "Daily" as the "Trigger".
4. In the Inspector window "Properties >Properties > General > at" enter the time "18:00".
5. Select the "LogTime" tag as the "Standard timer".
6. In the Inspector window, open "Properties > Events".

7. Select the "System functions/Logs/CopyLog" system function in the function list.
8. Select these settings:
   - Select "Data log" as the "Log type".
   - Select "DataLogDestination" as the "Destination log".
   - Select "DataLogSource" as the "Source log".
   - Select "Overwrite" as the "Mode".
   - Select "Yes" for "Delete source log".

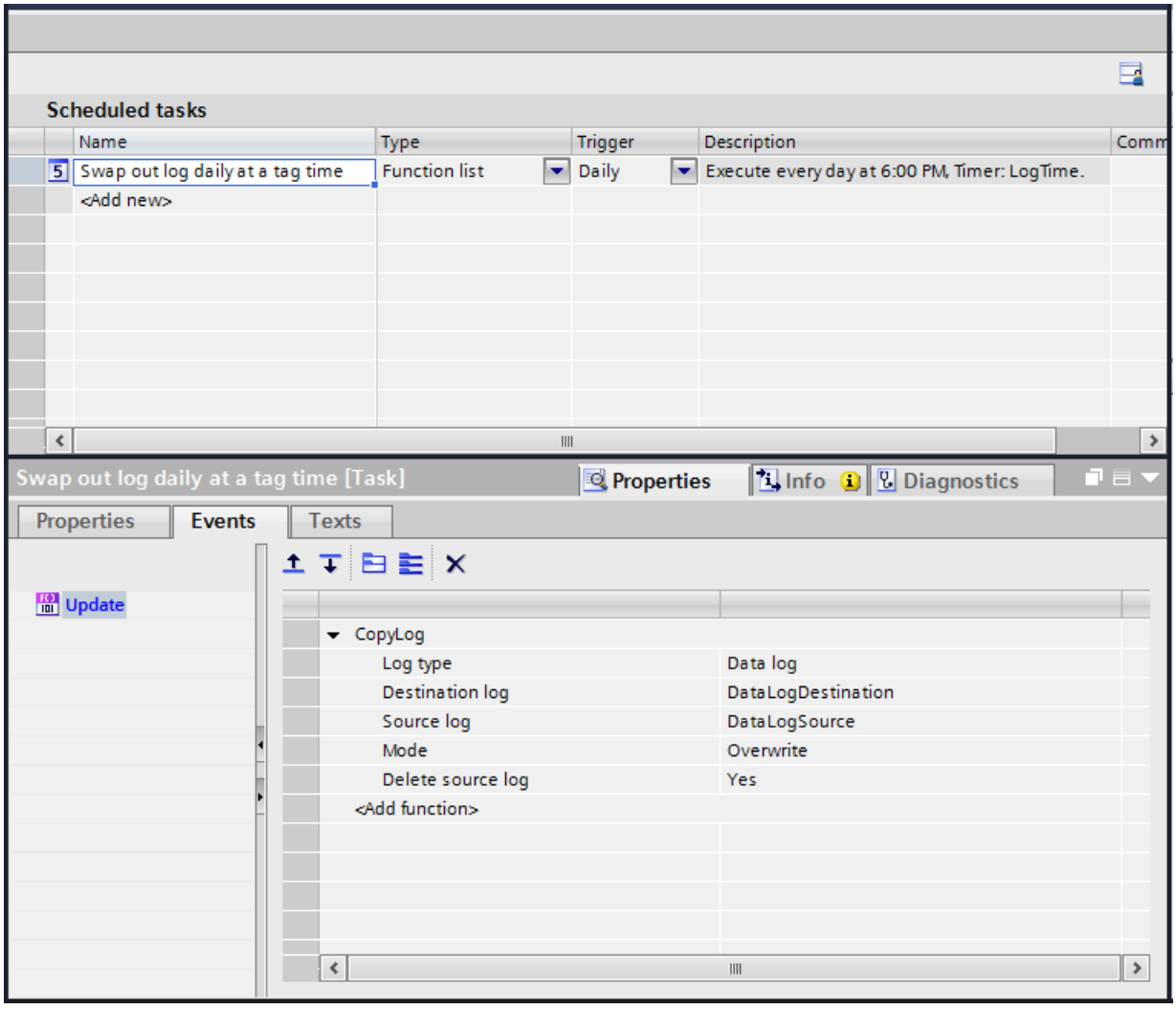

Result

If nothing is entered in the "LogDataTimeField" input field, then the task starts daily at 18:00 by default. The log data are swapped out. If an operator enters 19:00 in the "LogDataTimeField" input field, then the log data are swapped out at 19:00. Provided the value of the tag is not changed again before the task starts.

Using the diagnostics functions 12

12.1 Configuring system diagnostics

12.1.1 System diagnostics (RT Advanced, Basic Panels, Panels, Comfort Panels)

12.1.1.1 Basic Panel basics (RT Advanced, Basic Panels, Panels, Comfort Panels)

System diagnostics basics (Basic Panels)

Introduction

Using system diagnostics, you can display the messages from the diagnostic buffer of all integrated connections.

System diagnostics view

The system diagnostics display is an operating and display object that you use in a screen.

You navigate directly to the cause of the error and the associated connection. You have access to all integrated connections that you have configured in the "Devices & networks" editor.

System diagnostics views (Basic Panels)

Introduction

Three different views are available in the simple system diagnostic view.

- Device view
- Diagnostic buffer view
- Detail view

Device view

The device view is only displayed if more than one integrated connection has been configured.

The device view shows all available connections in a table. Double-clicking a connection opens the diagnostic buffer view.

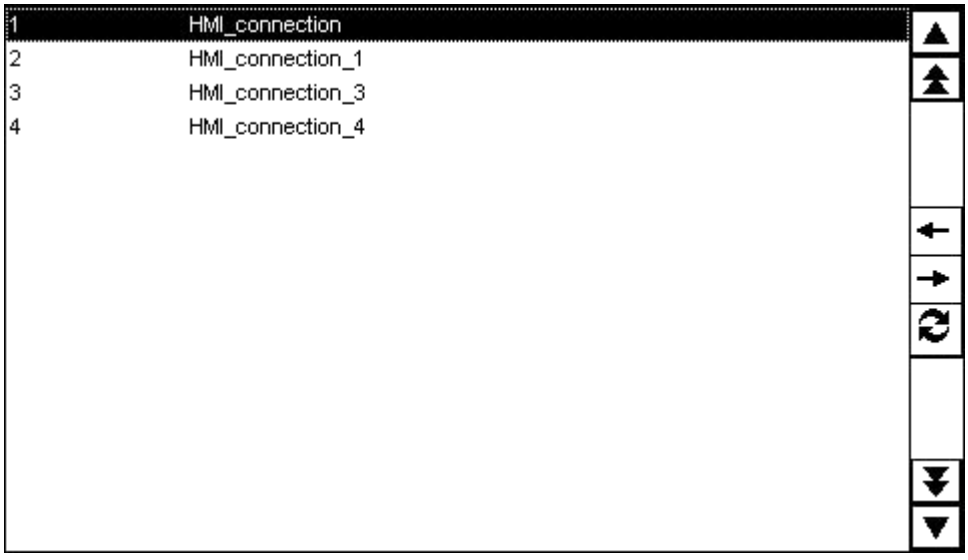

Diagnostic buffer view

The diagnostic buffer view shows the current data from the diagnostic buffer.

To update the diagnostic buffer view, you click .

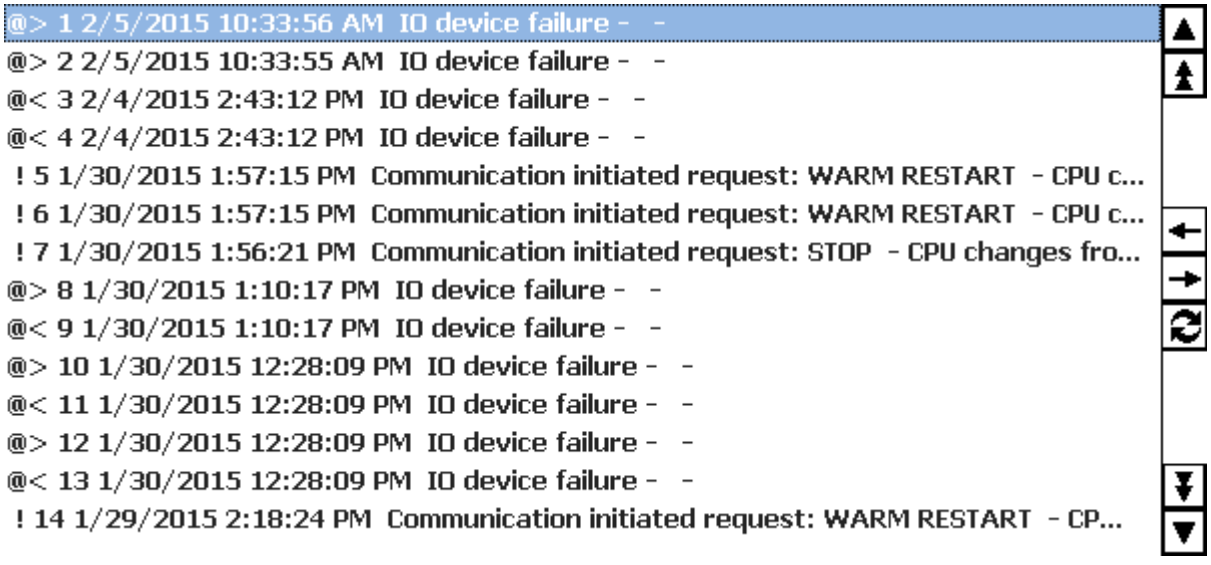

Detail view

The detail view gives detailed information about the selected connection and any pending errors. Check whether the data is correct in the detail view.

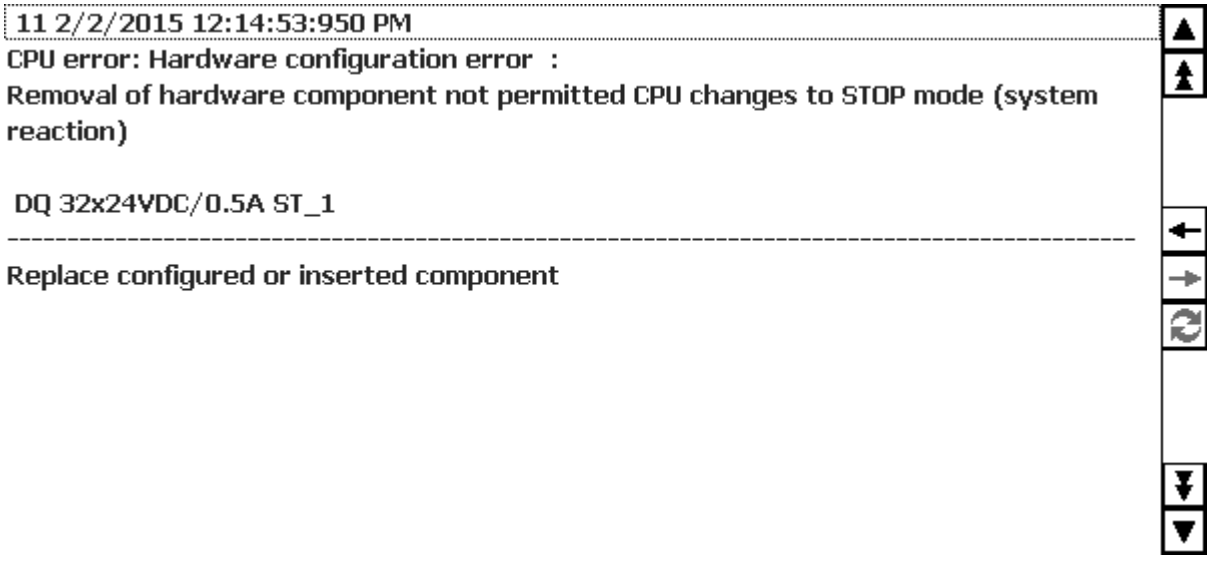

Note

Contents of the detail view

The contents of the detail view are only available for integrated connections with S7-1200 and S7-1500 PLCs.

Navigation buttons

Button	Key	Function
	Enter key	In the device view: Opens the diagnostic buffer view of the selected device. In the diagnostic buffer view: Opens the detail view.
	Esc key	In the diagnostic buffer view: Opens the device view. In the detail view: Opens the diagnostic buffer view.
	Configured function key, e.g. F1.	Updates the diagnostic buffer view.

12.1.1.2 Panels and Runtime Advanced basics (RT Advanced, Basic Panels, Panels, Comfort Panels)

System diagnostics basics (RT Advanced, Panels, Comfort Panels)

Introduction

You use system diagnostics to detect problems and errors in any part of your plant. WinCC has two display and operating elements for quick error localization.

For the controller to be able to transfer error messages to the system diagnostics view or the system diagnostics window, you must activate the system diagnostics in the settings for some controllers.

During communication with the SIMATIC S7-1500 and SIMATIC S7-1200 PLCs, system diagnostics is automatically activated.

The option "System diagnostics" is automatically activated for the respective HMI connection on the HMI device under "Runtime settings > Alarms > Controller alarms".

System diagnostics view

The alarm view shows the status of a PLC while the system diagnostics view gives you an overview of all devices available in your plant: You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

System diagnostics window

The system diagnostics window is an operating and display element that you can only use in the global screen.

The functions of the system diagnostics window are no different than those of the system diagnostics view. Because the system diagnostics window is configured in the global screen, you can, for example, also specify if the object is closable in Runtime.

Note

System diagnostics on Basic Panels

The "System diagnostics window" object is not available on Basic Panels.

System diagnostics views (RT Advanced, Panels, Comfort Panels)

Introduction

There are four different views available in the system diagnostics view and the system diagnostics window.

- Device view
- Diagnostic buffer view
- Detail view
- Matrix view (for master systems, PROFIBUS, PROFINET only)

Device view

The device view shows all the available devices of a layer in a table. Double-clicking on a device opens either the child devices or the detail view. Symbols in the first column provide information about the current status of the device.

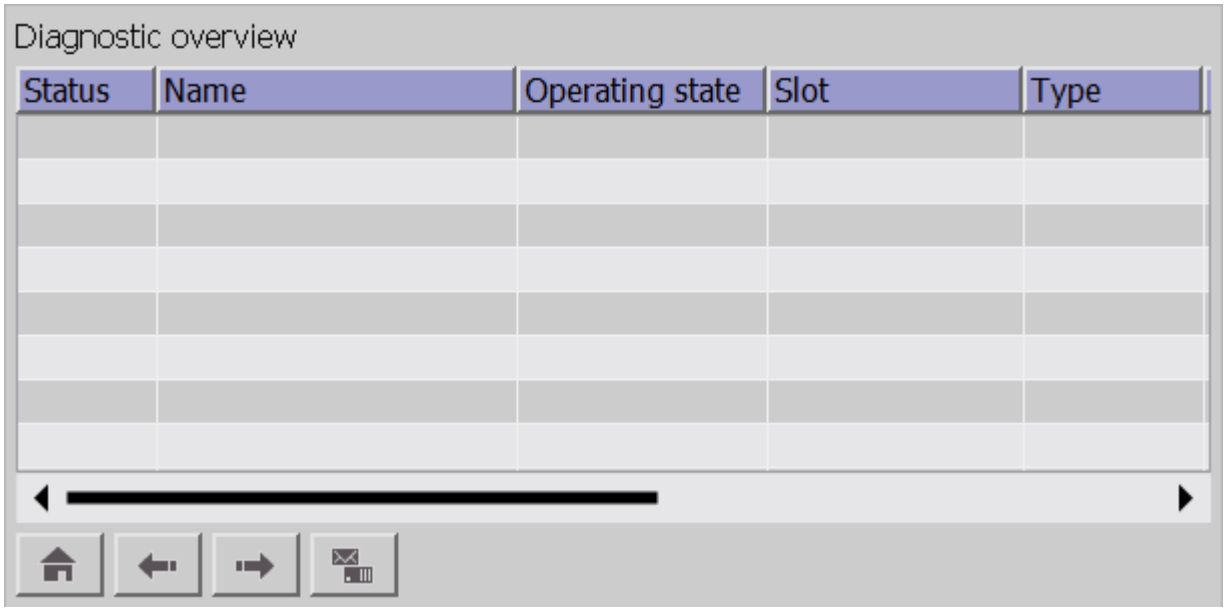

Diagnostic buffer view

In the diagnostic buffer view, the current data from the diagnostic buffer of the device is displayed. The diagnostic buffer view can only be called up in the device view.

To update the diagnostic buffer view, you click .

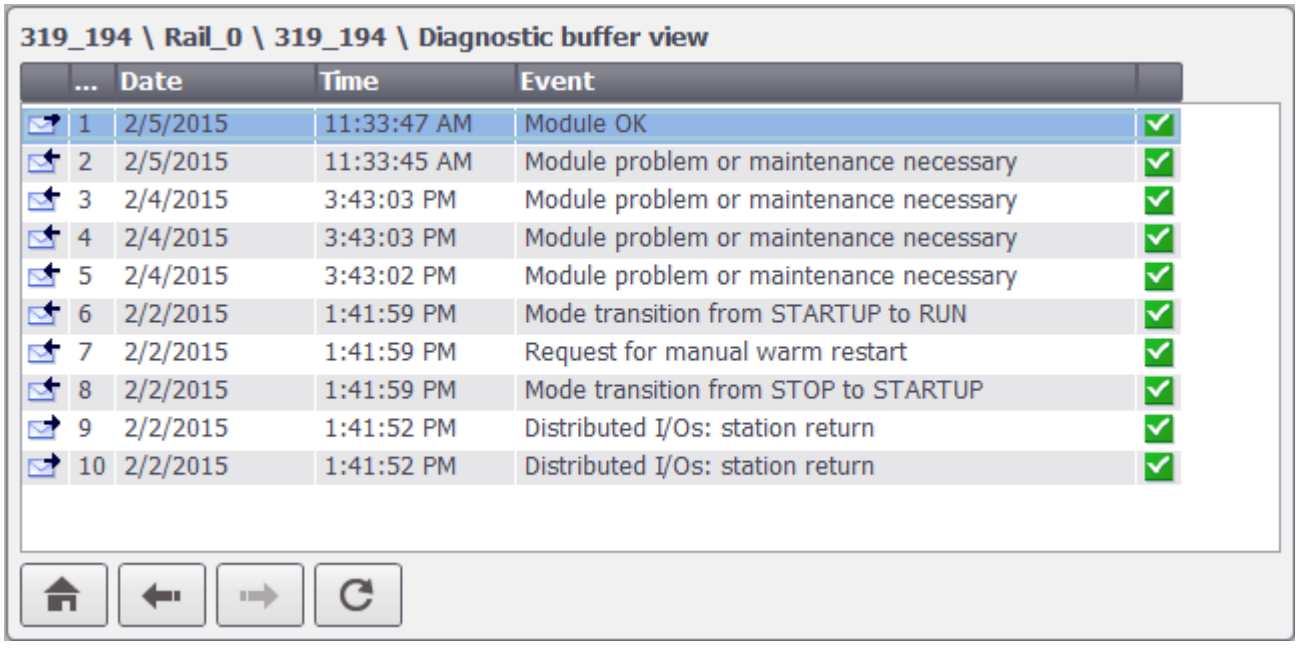

Detail view

The detail view gives detailed information about the selected device and the pending errors. Check whether the data is correct in the detail view. You cannot sort error texts in the detail view.

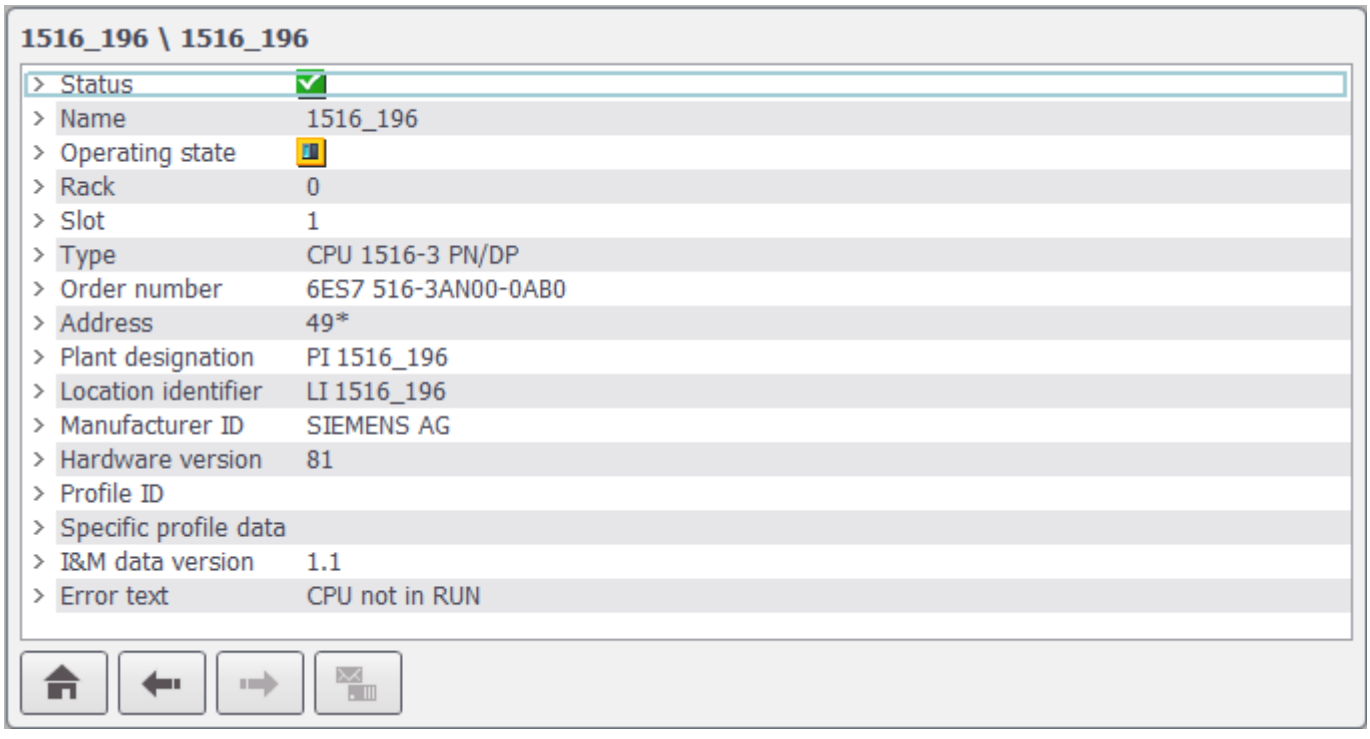

Note

Contents of the detail view

The contents of the detail view are only available for integrated connections with S7-1200 and S7-1500 PLCs.

Distributed I/O view

The distributed I/O view is available for master systems and IO systems. In this view, you can see the status of the subdevices of the system. Each element in the view shows the device name, the device type and the IP address or the PROFIBUS address.

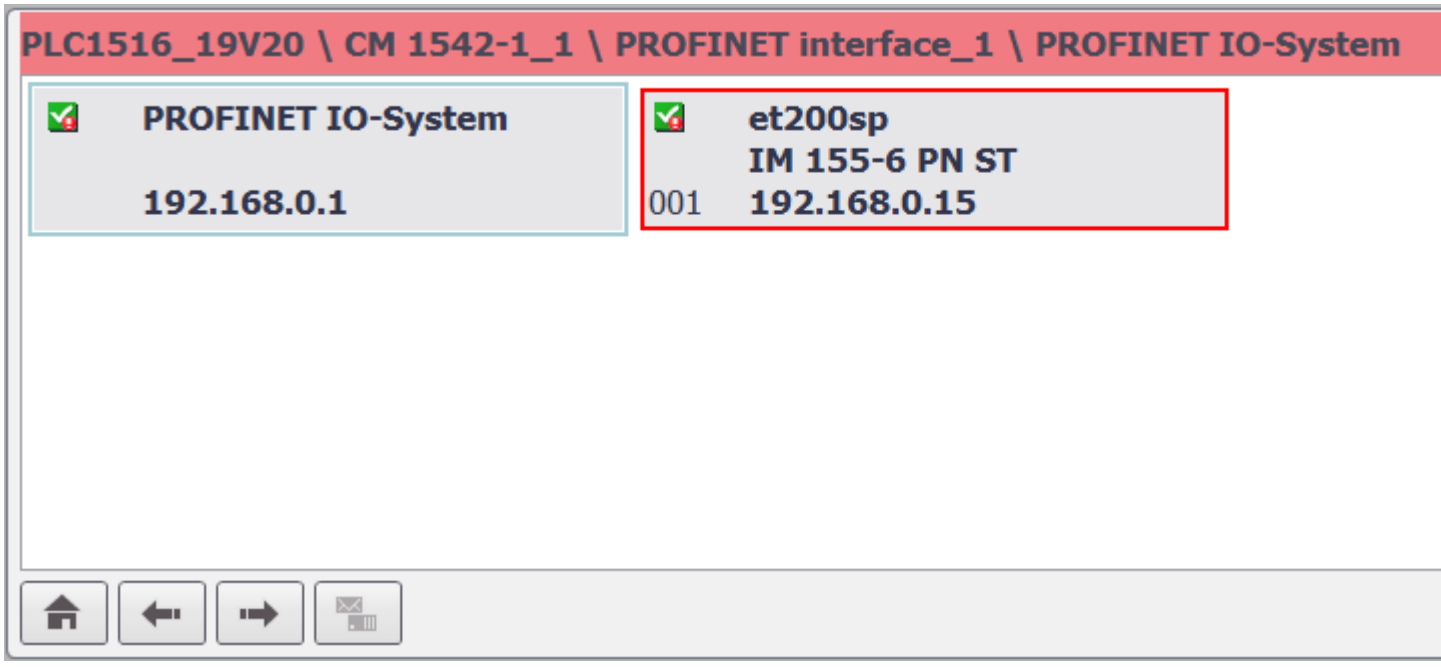

In the case of a fault, the affected element is visually highlighted:

- Red border indicates a fault in a lower-level element
- Red background indicates a fault in the element itself, for example, the device is not available.

Navigation buttons

Button	Key	Function
	Enter key	Opens the child devices or the detail view if there are no child devices.
	Esc key	Opens the parent device or the device view if there is no parent device.

Button	Key	Function
	Home key	Opens the device view.
	Configured function key, e.g. F1.	Opens the diagnostic buffer view. Only visible in the device view.
	Configured function key, e.g. F2.	Updates the diagnostic buffer view.

Note

Navigation in the matrix view of Comfort Panels with function keys

The Shift key is additionally used to select PNIO or PROFIBUS slaves in the matrix view.

Language switching in runtime

Switch over the language in runtime before you open the details view.

If you switch over the runtime language while the details view is opened, you change back to the diagnostic buffer view.

Open the details view again. The selected runtime language is now displayed correctly.

12.1.1.3 Configuring system diagnostics objects (RT Advanced, Basic Panels, Panels, Comfort Panels)

Enabling system diagnostics in the PLC

Introduction

During communication with the SIMATIC S7-1500 and SIMATIC S7-1200 PLCs, system diagnostics is automatically activated.

For other PLCs, the following settings must be implemented in the PLC to enable the PLC to transfer error messages to the system diagnostics view.

Requirements

- A PLC such as SIMATIC S7-300/400 or ET 200 has been created.
- The "Devices & Networks" editor is open in the "Device view".

Procedure

1. Select the PLC in the selection list.
2. Click on the PLC and select "Properties" in the shortcut menu. The properties of the PLC are displayed in the Inspector window.
3. Select "Properties > General > System diagnostics" in the Inspector window.
4. Enable "Activate system diagnostics for this CPU".
5. Select the PLC and then "Compile > Hardware configuration" in the shortcut menu.

Note

Special feature system diagnostic alarms SIMATIC S7-300/400

When you deactivate the display class 0 for the respective HMI connection under "Runtime settings > Alarms > Controller alarms > Display classes", the system diagnostic alarms are not transferred to the HMI device by the controller even when the "System diagnostics" option is activated in the controller and on the HMI device.

Result

The settings will be effective following the next compilation. The PLC can now transfer error messages to the system diagnostics display.

Adding a system diagnostics indicator (RT Advanced, Panels, Comfort Panels)

Introduction

The system diagnostics indicator is a predefined graphic symbol of the library which alerts you to errors in your system.

If an error occurs, the appearance of the object changes. The library object can display two states:

- No error

- Error

Requirement

- The "Libraries" task card is opened.
- The global library "Buttons and Switches > Master copies > DiagnosticsButtons" is open.
- A screen is open.
- A system diagnostics window has been created in the global screen.

Procedure

1. Select the "DiagnosticsIndicator" object in the library.

   Note
   System diagnostics indicator for different device types
   When selecting the object ensure that you use the correct object for the device type used in the project.

2. Drag-and-drop the library object to the position in the work area where you want to insert the object.
   The library object is inserted.

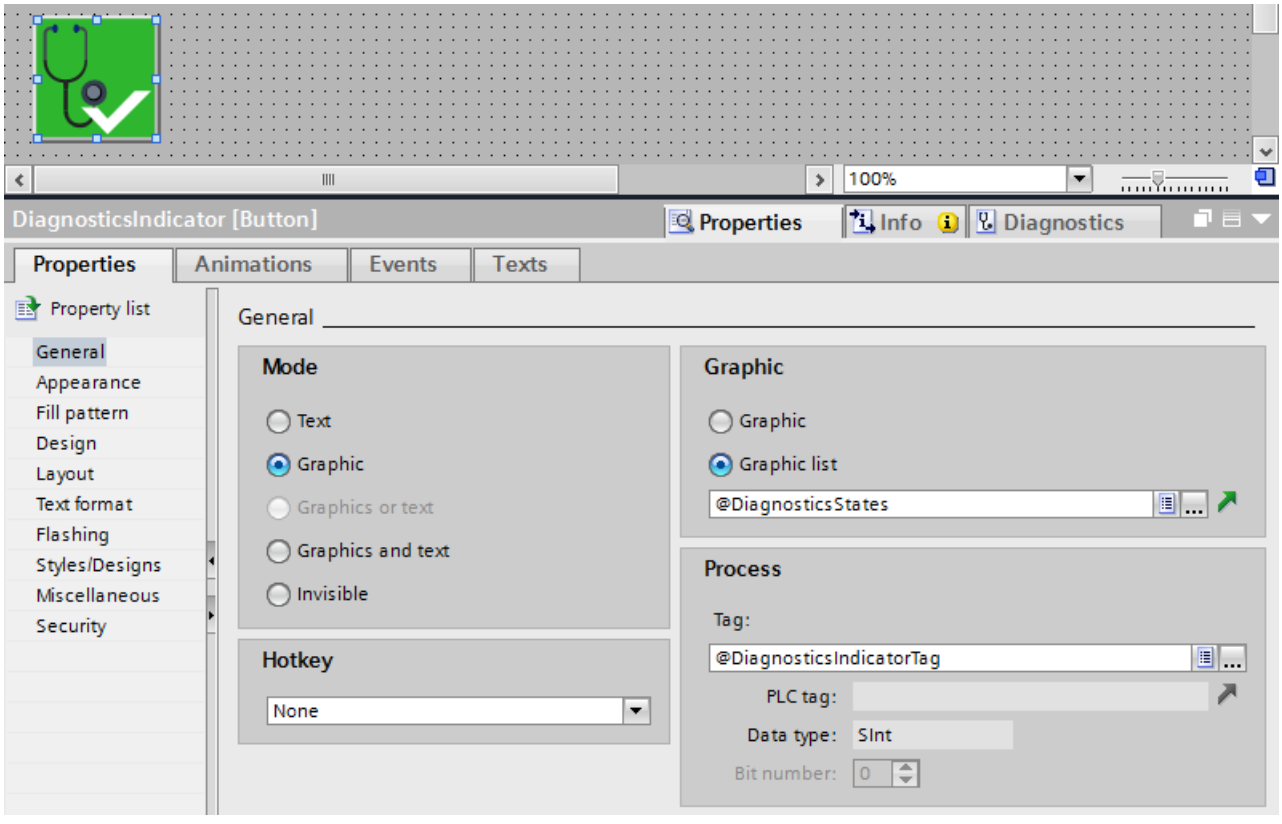

3. Select the library object.
4. Click "Properties > Events" in the Inspector window.
   The system function "ShowSystemDiagnosticsWindow" is preset for the event "Click".

Result

The status diagnostics indicator has been added to the project and connected with the system diagnostics window.

The system diagnostics indicator changes its appearance if an error message is output in Runtime. The system diagnostics window opens when you click on the system diagnostics indicator. The system diagnostics window shows the detail view of the affected device.

Configuring access protection for the system diagnostics window

Configure access protection for the system diagnostics indicator to prevent unauthorized access to the system diagnostics window.

1. Select the "DiagnosticsIndicator" object in the screen.
2. Select an authorization in "Properties > Properties > Security in Runtime" in the Inspector window.

A logon dialog opens when you click on the system diagnostics indicator in Runtime. The system diagnostics window does not open unless you have the required authorization.

Configuring a system diagnostics window in the global screen (RT Advanced, Panels, Comfort Panels)

Introduction

The system diagnostics window gives you an overview of all devices available in your plant. The system diagnostics window operates like the system diagnostics view, but it is only available in the global screen.

Requirements

- At least one PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on each PLC.
- A Comfort Panel or WinCC RT Advanced has been created.
- The global screen is open.
- The Inspector window is open.

Procedure

1. Double-click the "System diagnostics window" object in the "Tools" task card. The object is added to the global screen.
2. Select "Properties > Properties > Columns > Devices/Detail view" in the Inspector window.
3. Enable the columns that you require in the device view for Runtime, for example: status, name, address.
4. Enable the properties which you require in the detail view for Runtime, for example: status, name, plant designation.
5. Activate the columns that you require in the diagnostic buffer view, for example, status, name.
6. Customize the column headers, if necessary.
7. Enable "Properties >Properties > Layout > Column settings > Columns moveable" to move the columns in Runtime.
8. Enable "Properties > Properties > Window > Closable" to allow the system diagnostics window to be closed in Runtime.

Result

The system diagnostics window has been added to the global screen. The system diagnostics window will respond to error messages in the plant and display the device affected.

Configure a system diagnostics indicator in a screen to open the system diagnostics window.

Configuring button as system diagnostics indicator (RT Advanced, Panels, Comfort Panels)

Introduction

Instead of using the object "DiagnosticsIndicator" from the library, you can configure a button, for example, to indicate errors in your plant.

Requirements

- At least one PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on each PLC.
- The "Tools" task card is open.
- A bit graphics list has been created with two different graphics for the statuses.
- A screen is open.
- You have created a system diagnostics view.

Procedure

1. Double-click the "Button" object in the "Tools" task card. A button will be added to the screen.
2. In the Inspector window, select "Properties > Properties > General > Mode > Graphic".
3. Select the bit graphics list under "Graphic > Graphics list".
4. Select the "@DiagnosticsIndicatorTag" tag under "Properties > Properties > General > Tag" in the Inspector window.

Configuring system function to the button

1. Click "Properties > Events" in the Inspector window.
2. Select the "Click" event.
3. Click on "Add function" in the table.
4. Select the "ActivateSystemDiagnosticsView" system function.
5. Select the system diagnostics view.

Result

You have now configured a button which will respond to error messages from the PLC. The button changes its appearance if an error message is output in Runtime.

The button has two states:

- Error
  The system diagnostics view opens when you click the button. The system diagnostics view shows the detail view of the device concerned.
- No error
  The system diagnostics view opens when you click the button. The system diagnostics view shows the device view.

Configuring the system diagnostic view (RT Advanced, Panels, Comfort Panels)

Introduction

You add a system diagnostics view to your project to get an overview of all devices available in your plant.

Requirements

- At least one PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on each PLC.
- A Comfort Panel or WinCC RT Advanced has been created.
- You have created a screen.
- The Inspector window is open.

Procedure

1. Double-click the "System diagnostics view" object in the "Tools" task card. The object is added to the screen.
2. Select "Properties > Properties > Columns > Devices/Detail view" in the Inspector window.
3. Activate the columns that you require in the device view for Runtime, e.g. Status, Name, Slot.
4. Activate the columns that you require in the detail view for Runtime, e.g. Status, Name, Plant designation.
5. Activate the columns that you require in the diagnostic buffer view, for example, Status, Name.
6. Enable "Properties >Properties > Layout > Column settings > Columns moveable" to move the columns in Runtime.
7. You can change the column headers under "Properties > Properties > Column headers", if necessary.

Result

The system diagnostics view has been added to the screen. Error for the entire plant are now displayed in the system diagnostics view in Runtime.

Configuring the system diagnostic view (Basic Panels)

Introduction

For an overview of all integrated connections, you insert a system diagnostics view in your project.

Requirements

- A PLC has been created.
- A Basic Panel has been created.
- An integrated connection has been created in the "Devices & Networks" editor.
- You have created a screen.
- The Inspector window is open.

Procedure

1. Double-click the "System diagnostics view" object in the "Tools" task card. The object is added to the screen.
2. In the Inspector window, select "Properties > Layout".
3. Enter a number under "Lines per entry", i.e. 5.

Result

The system diagnostics view has been added to the screen.

To obtain the latest alarms, update the diagnostic buffer.

Splitting the view in system diagnostics view (RT Advanced, Panels, Comfort Panels)

Introduction

The system diagnostics view also offers a split view of the display. You have the option to see the devices and the associated details at a glance.

Requirements

- At least one PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on each PLC.
- An HMI device has been created, for example, a Comfort Panel.
- You have created a screen.
- The Inspector window is open.

Procedure

1. Double-click the "System diagnostics view" object in the "Tools" task card. The object is added to the screen.
2. In the Inspector window, click "Properties > Properties > Layout".
3. Activate "Show split view".
4. Activate "Sizing > Auto-size"
5. You may also change the number of lines for the "Device view" and the "Detail view" if required.

Result

The system diagnostics view has been added to the screen. The system diagnostics view is split into two areas.

The top area shows the device view. The bottom view shows the detail view.

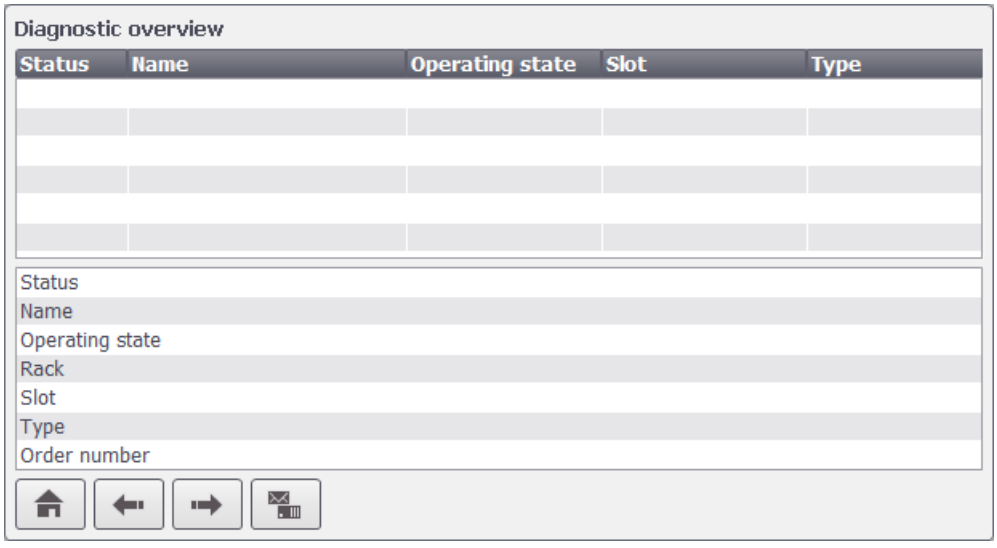

If you open the diagnostic buffer view, the top area displays an overview of the diagnostic buffer and the bottom area displays the details.

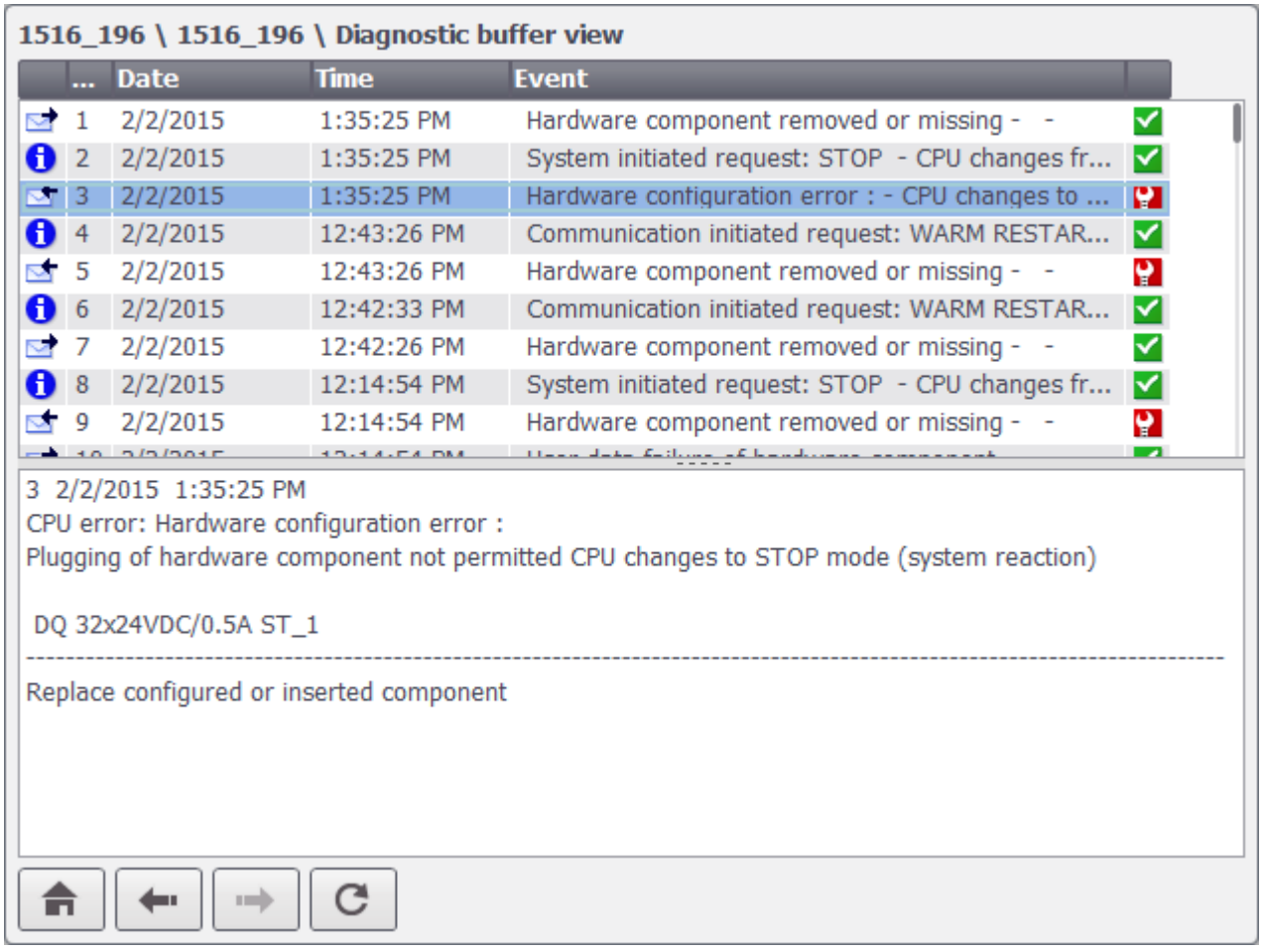

Note

Contents of the detail view

The contents of the detail view are only available for integrated connections with S7-1200 and S7-1500 PLCs.

12.1.2 System diagnostics (RT Professional)

12.1.2.1 Runtime Professional basics (RT Professional)

System diagnostics basics (RT Professional)

Introduction

You use system diagnostics to detect problems and errors in any part of your plant. WinCC offers two operating elements for quick error localization.

To ensure that the controller can transfer error messages to the system diagnostics view or the system diagnostics window, you have to activate system diagnostics for some PLCs in the settings.

During communication with the SIMATIC S7-1500 and SIMATIC S7-1200 PLCs, system diagnostics is automatically activated.

On the HMI device, the option "System diagnostics" is automatically activated under "Runtime Settings > Alarms > Controller alarms" for the relevant HMI connection.

System diagnostics view

The alarm view shows the status of a PLC while the system diagnostics view gives you an overview of all devices available in your plant: You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

System diagnostics indicator

The system diagnostics indicator is a predefined graphic symbol of the library which alerts you to errors in your system.

The system diagnostics indicator changes its appearance when an error occurs.

GoTo button

The GoTo button is a predefined graphic symbol in the library. When diagnostic alarms appear in the alarm view, you can use the GoTo function to jump to the error location in the system diagnostics view.

See also

Enabling system diagnostics in the PLC (Page 1840)

System diagnostics views (RT Professional)

Introduction

There are four different views available in the system diagnostics view and the system diagnostics window.

- Device view
- Diagnostic buffer view
- Detail view
- Matrix view (for master systems, PROFIBUS, PROFINET only)

Device view

The device view shows all the available devices of a layer in a table. Double-clicking on a device opens either the child devices or the detail view. Symbols in the first column provide information about the current status of the device.

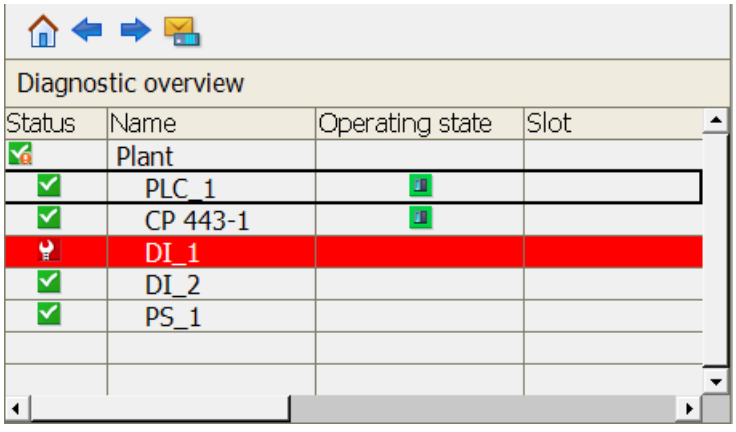

Diagnostic buffer view

In the diagnostic buffer view, the current data from the diagnostic buffer of the device is displayed. The diagnostic buffer view can only be called up in the device view.

To update the diagnostic buffer view, click "Update".

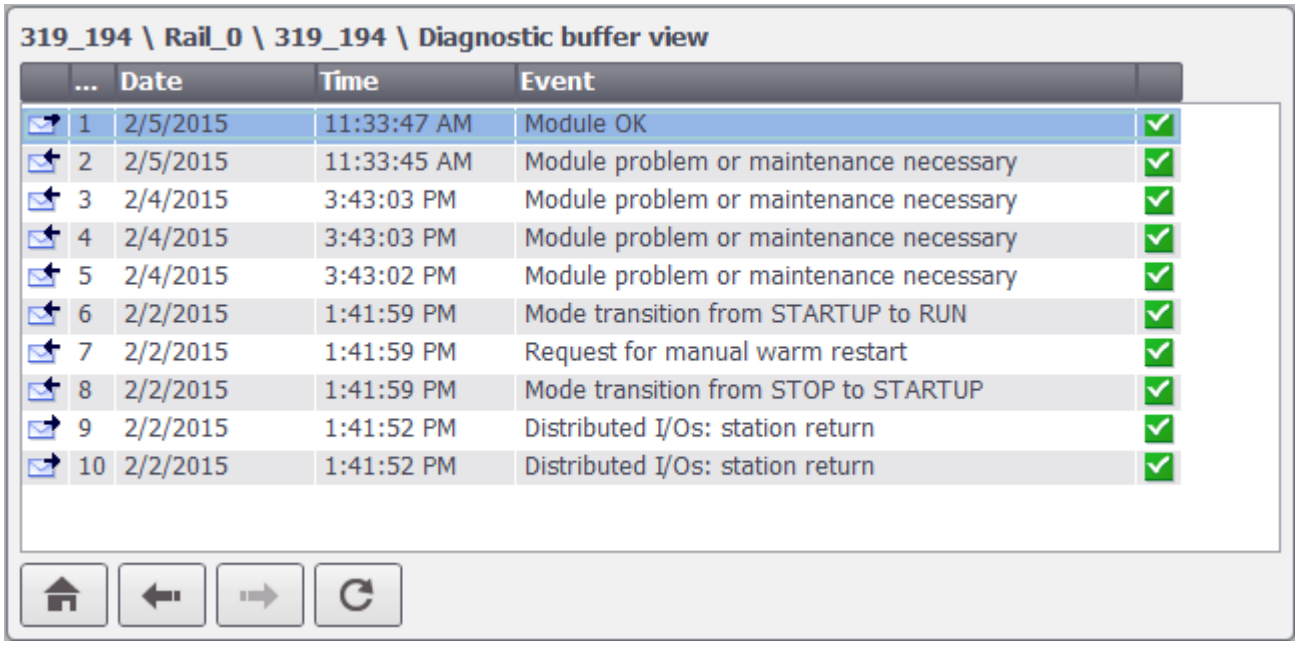

...	Date	Time	Event
1	2/5/2015	11:33:47 AM	Module OK
2	2/5/2015	11:33:45 AM	Module problem or maintenance necessary
3	2/4/2015	3:43:03 PM	Module problem or maintenance necessary
4	2/4/2015	3:43:03 PM	Module problem or maintenance necessary
5	2/4/2015	3:43:02 PM	Module problem or maintenance necessary
6	2/2/2015	1:41:59 PM	Mode transition from STARTUP to RUN
7	2/2/2015	1:41:59 PM	Request for manual warm restart
8	2/2/2015	1:41:59 PM	Mode transition from STOP to STARTUP
9	2/2/2015	1:41:52 PM	Distributed I/Os: station return
10	2/2/2015	1:41:52 PM	Distributed I/Os: station return

Detail view

The detail view gives detailed information about the selected device and the pending errors. Check whether the data is correct in the detail view. You cannot sort error texts in the detail view.

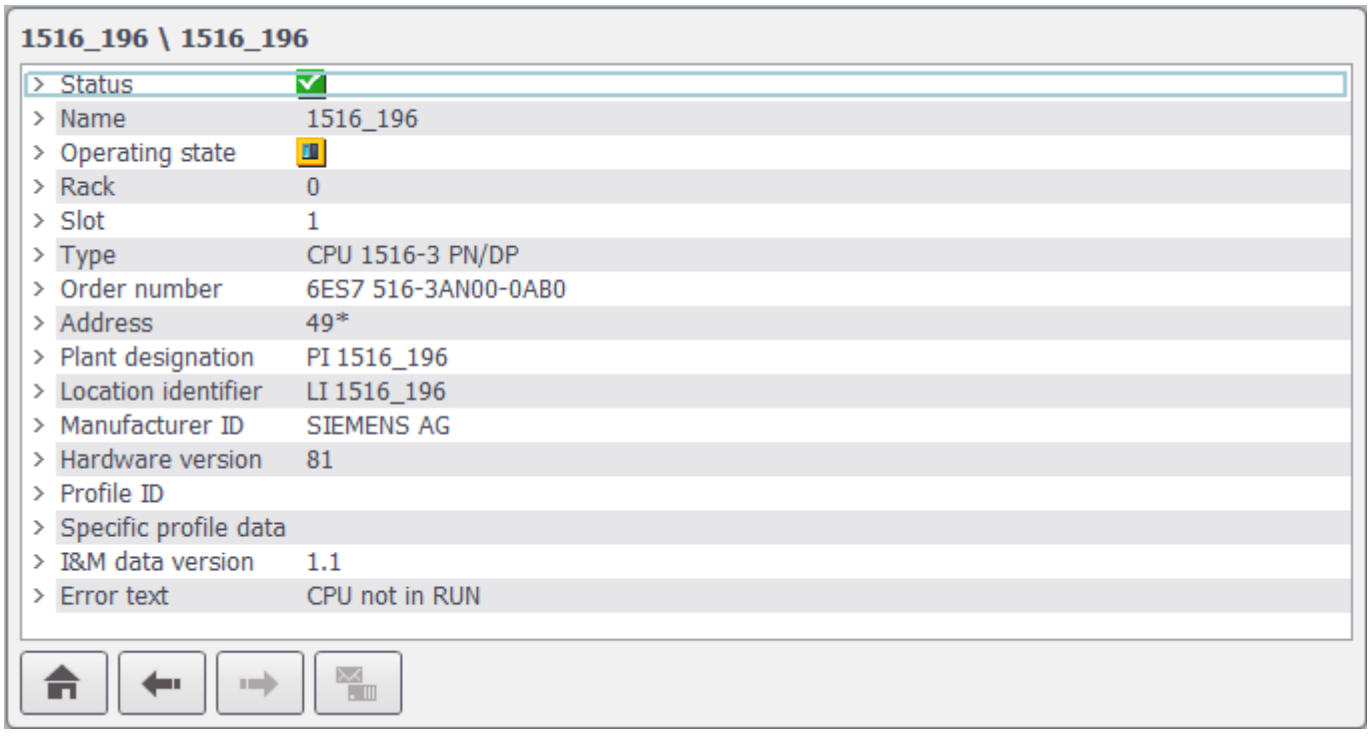

Status	
Name	1516_196
Operating state	
Rack	0
Slot	1
Type	CPU 1516-3 PN/DP
Order number	6ES7 516-3AN00-0AB0
Address	49*
Plant designation	PI 1516_196
Location identifier	LI 1516_196
Manufacturer ID	SIEMENS AG
Hardware version	81
Profile ID	
Specific profile data	
I&M data version	1.1
Error text	CPU not in RUN

Note

Contents of the detail view

The contents of the detail view are only available for integrated connections with S7-1200 and S7-1500 PLCs.

Navigation buttons

Button	Key	Function
	Enter key	Opens the child devices or the detail view if there are no child devices.
	Esc key	Opens the parent device or the device view if there is no parent device.
	Home key	Opens the device view.
	Function key F4.	Opens the diagnostic buffer view. Only visible in the device view.
	Function key F5.	Updates the diagnostic buffer view.

12.1.2.2 Configuring system diagnostics objects (RT Professional)

Enabling system diagnostics in the PLC (RT Professional)

Introduction

During communication with the SIMATIC S7-1500 and SIMATIC S7-1200 PLCs, system diagnostics is automatically activated.

For other PLCs, the following settings must be implemented in the PLC to enable the PLC to transfer error messages to the system diagnostics view.

Requirements

- A PLC such as SIMATIC S7-300/400 or ET 200 has been created.
- The "Devices & Networks" editor is open in the "Device view".

Procedure

1. Select the PLC in the selection list.
2. Click on the PLC and select "Properties" in the shortcut menu. The properties of the PLC are displayed in the Inspector window.

3. Select "Properties > General > System diagnostics" in the Inspector window.
4. Enable "Activate system diagnostics for this CPU".
5. Select the PLC and then "Compile > Hardware configuration" in the shortcut menu.

Note

Special feature system diagnostic alarms SIMATIC S7-300/400

When you deactivate the display class 0 for the respective HMI connection under "Runtime settings > Alarms > Controller alarms > Display classes", the system diagnostic alarms are not transferred to the HMI device by the controller even when the "System diagnostics" option is activated in the controller and on the HMI device.

Result

The settings will be effective following the next compilation. The PLC can now transfer error messages to the system diagnostics display.

See also

System diagnostics basics (Page 1837)

Configuring the system diagnostics view (RT Professional)

Introduction

You add a system diagnostics view to your project to get an overview of all devices available in your plant.

Requirements

- At least one PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on each PLC.
- You have created a PC station with RT Professional.
- An HMI connection has been established between the PLC and RT Professional.
- You have created a screen.
- The Inspector window is open.

Procedure

1. Double-click the "System diagnostics view" object in the "Tools" task card. The object is added to the screen.
2. Select "Properties > Properties > Columns > Devices/Detail view" in the Inspector window.

3. Activate the columns that you require in the device view for Runtime, e.g. Status, Name, Slot.
4. Activate the columns that you require in the detail view for Runtime, e.g. Status, Name, Plant designation.
5. Activate the columns that you require in the diagnostic buffer view, for example, status, name.
6. Select "Properties >Properties > Layout > Column settings > Columns movable" to enable move of the columns in Runtime.
7. You can change the column headers under "Properties > Properties > Column headers", if necessary.

Result

The system diagnostics view has been added to the screen. Error for the entire plant are now displayed in the system diagnostics view in Runtime.

Splitting the view in system diagnostics view (RT Professional)

Introduction

The system diagnostics view also offers a split view of the display. You have the option to see the devices and the associated details at a glance.

Requirements

- At least one PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on each PLC.
- You have created a PC station with RT Professional.
- You have created a screen.
- The Inspector window is open.

Procedure

1. Double-click the "System diagnostics view" object in the "Tools" task card. The object is added to the screen.
2. In the Inspector window, click "Properties > Properties > Layout".
3. Activate "Show split view".
4. You may also change the number of lines for the "Device view" and the "Detail view" if required.

Result

The system diagnostics view has been added to the screen. The system diagnostics view is split into two areas.

The top area shows the device view. The bottom view shows the detail view.

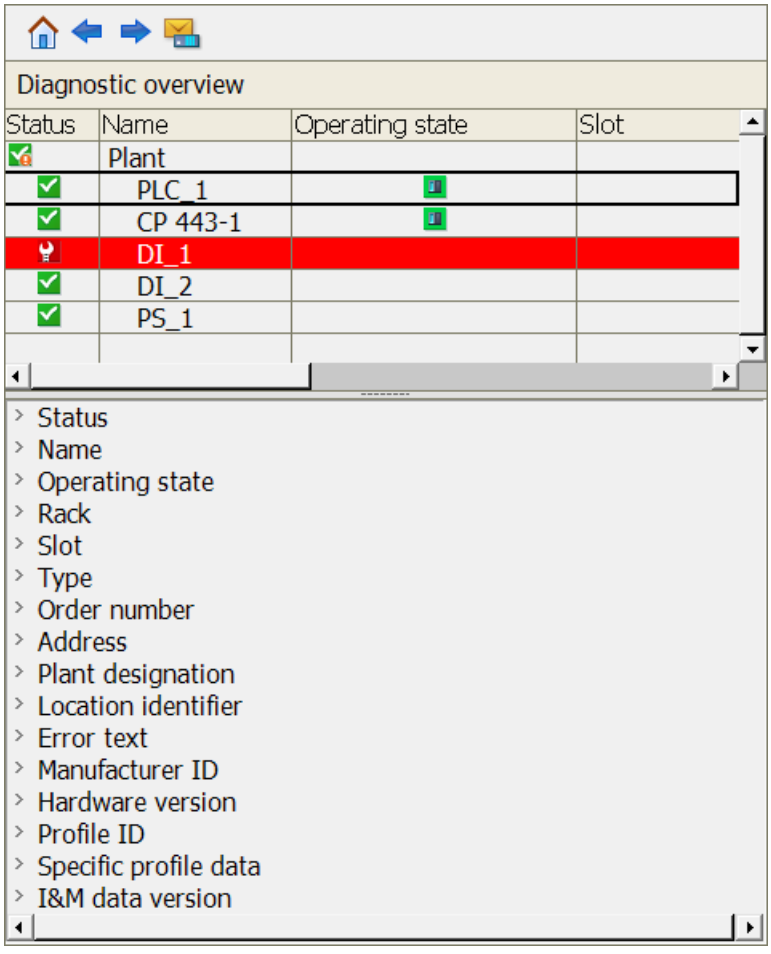

If you open the diagnostic buffer view, the top area displays an overview of the diagnostic buffer and the bottom area displays the details.

Note

Contents of the detail view

The contents of the detail view are only available for integrated connections with S7-1200 and S7-1500 PLCs.

12.1.2.3 Example: System diagnostics with all objects (RT Professional)

Example: Procedures overview (RT Professional)

Introduction

- A PLC has been created.
- The option "Activate system diagnostics for this CPU" is activated on the PLC.
- A connection has been established between the PLC and PC station.

Configuration steps

To get a quick view of errors, create an overview screen with the various objects for displaying diagnostic alarms.

The following example shows how to efficiently use the objects of the "Tools" or "Libraries" task cards in your project.

The example is divided into several steps:

- Creating screens
  The project engineer creates multiple screens for system diagnostics:
  - Overview screen with all objects for system diagnostics
  - Screen for the alarm view
  - Screen for the system diagnostics view
- Inserting objects in the screens
  The project engineer inserts various objects in the screens:
  - Alarm view in the "Alarm" screen
  - Screen window for displaying the alarm view and system diagnostics view
- Configuring objects
  The project engineer connects objects to enable targeted navigation to the cause of the error in runtime.
  - System diagnostic indicator with screen window of the system diagnostics view
  - Goto button with the screen window and the alarm view and system diagnostics view.

See also

Example: Creating screens (Page 1845)

Example: Inserting objects in the screens (Page 1845)

Example: Adding a system diagnostics indicator (Page 1846)

Example: Inserting a GoTo button (Page 1848)

Example: Creating screens (RT Professional)

Introduction

In the following, you will create three screens for displaying various objects.

Requirement

A project is open.

You have created a PC station with RT Professional.

Procedure

1. Open the project tree.
2. Click "Add new screen" in the Screens folder.
3. Open the shortcut menu of the screen and assign the name "Overview SysDiag".
4. Create another screen named "Alarm".
5. Open the "Libraries" task card.
6. Open the global library "Buttons and Switches > Kopiervorlagen > DiagnosticsButtons > RTProfessional".
7. Click the "DiagnosticScreen" object in the global library.
8. Drag-and-drop the object into the "Screens" folder in the project tree.

Result

You have created three screens in the project tree:

- Overview SysDiag
- Alarm
- SystemDiagnostics

A system diagnostics view has been created in the "SystemDiagnostics" screen.

See also

Example: Procedures overview (Page 1844)

Example: Inserting objects in the screens (RT Professional)

Introduction

In the following example, you will insert various objects from the "Tools" task card into the screens.

Requirement

The following three screens have been created:

- Overview SysDiag
- Alarm
- SystemDiagnostics

The "Tools" task card is open.

Procedure

1. Open the "Alarm" screen.
2. Drag-and-drop an "Alarm view" from the "Tools" task card into the screen.
3. Open the screen "Overview SysDiag".
4. Drag-and-drop a "Screen window" from the "Tools" task card into the screen.
5. Enter the name "Alarm" under "Properties > Properties > Miscellaneous > Name" in the Inspector window.
6. Select the "Alarm" screen under "Properties > Properties > General > Content > Displayed screen".
7. Create a second screen window named "SysDiag".
8. Select the "SystemDiagnostics" screen under "Properties > Properties > General > Content > Displayed screen".

Result

You have inserted an alarm view in the "Alarm" screen. The alarm view has the unchanged name "Alarm_view_1".

You have created two screen windows in the "Overview SysDiag" screen. The first screen window has the name "Alarm" and shows the alarm view. The second screen window has the name "SysDiag" and shows the system diagnostics view.

See also

Example: Procedures overview (Page 1844)

Example: Adding a system diagnostics indicator (RT Professional)

Introduction

The system diagnostics indicator is a predefined graphic symbol of the library which alerts you to errors in your system.

If an error occurs, the appearance of the object changes. The library object can display two states:

- No error

- Error

Requirement

- The "Libraries" task card is opened.
- The global library "Buttons and Switches > Master copies > DiagnosticsButtons > RT Professional" is open.
- The following three screens have been created:
  - Overview SysDiag
  - Alarm
  - SystemDiagnostics
- The "SysDiag" screen window has been created and linked with the "DiagnosticScreen" screen.

Procedure

1. Select the "DiagnosticsIndicator" object in the library.
2. Drag the library object to the screen "Overview SysDiag".
   The library object is inserted.
3. Select the library object.
4. Click "Properties > Events" in the Inspector window.
   Three system functions are preset for the "Click" event.

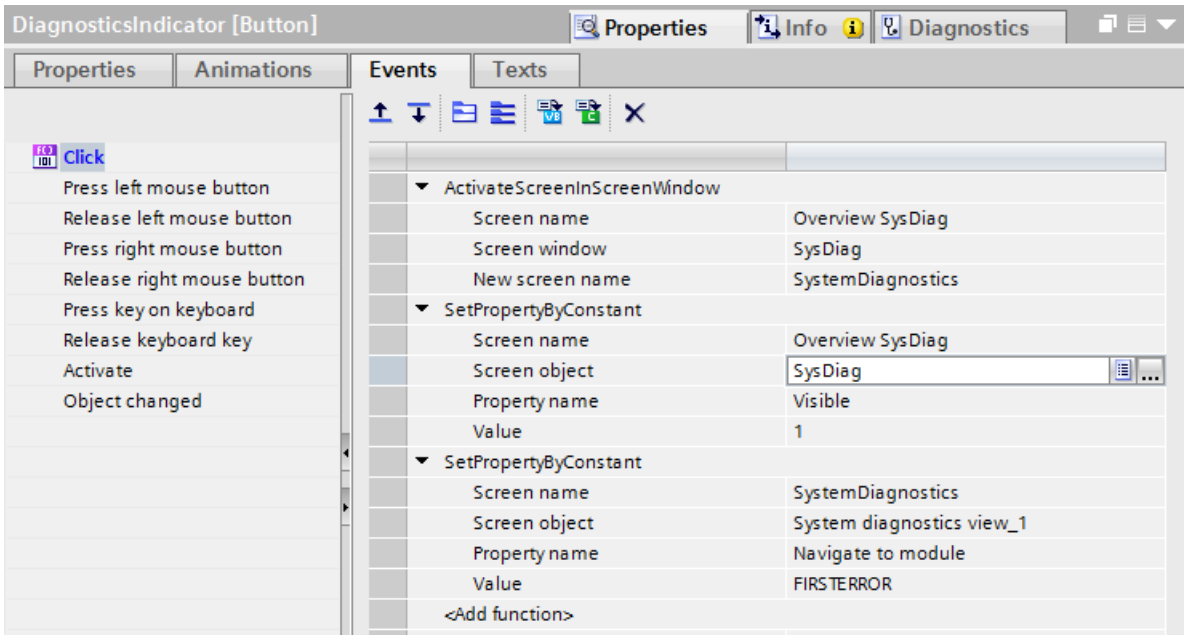

5. Select "SysDiag" for the "Screen name" parameter in the "ActivateScreenInScreenWindow" system function.
6. Select "SysDiag" for the "Screen object" parameter in the "SetPropertyByConstant" system function.

Result

The system diagnostics indicator has been added to the project and connected with the "SysDiag" screen window.

The system diagnostics indicator changes its appearance if an error message is output in Runtime. The SysDiag screen window in which system diagnostics view is configured opens when you click the system diagnostics indicator. The system diagnostics view shows the detail view of the device concerned.

See also

Example: Procedures overview (Page 1844)

Example: Inserting a GoTo button (RT Professional)

Requirements

- The "Libraries" task card is opened.
- The global library "Buttons and Switches > Master copies > DiagnosticsButtons > RT Professional" is open.
- The following three screens have been created:
  - Overview SysDiag
  - Alarm
  - SystemDiagnostics
- The "Alarm" screen window has been created and linked with the "Alarm" screen.
- The "SysDiag" screen window has been created and linked with the "DiagnosticScreen" screen.

Configuring a Goto button in the screen

1. Open the screen "SystemDiagnostic".
2. Open the global library "Buttons and Switches > Master copies > DiagnosticsButtons".
3. Select the "DiagnosticsGoto" object in the library.
4. Drag the library object to the screen "Overview SysDiag". The library object is inserted.
5. Select the library object.

6. Click "Properties > Events" in the Inspector window.
   Four system functions are preset for the event.

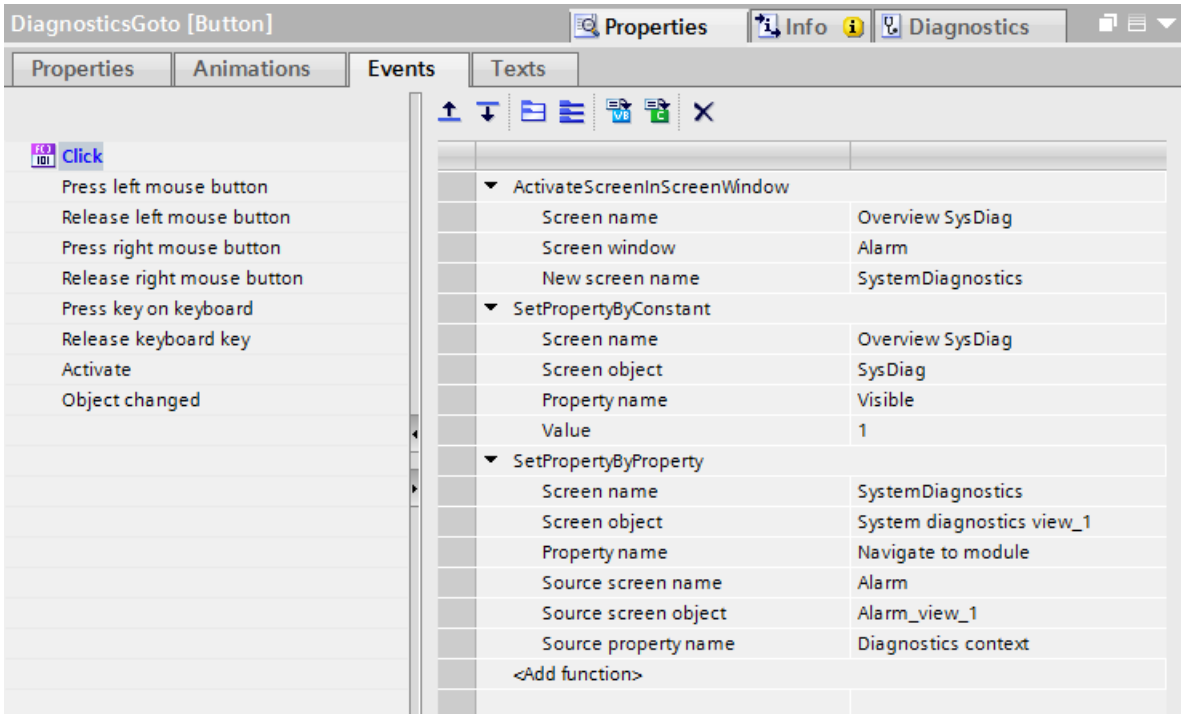

7. Select "SysDiag" for the "Screen name" parameter in the "ActivateScreenInScreenWindow" system function.
8. Select "SysDiag" for the "Screen object" parameter in the "SetPropertyByConstant" system function.
9. Select the following settings in the "SetPropertyByProperty" system function:

- Screen name: Alarm
- Object: Alarm_view_1

Result

You have configured the Goto button and linked it to the alarm view and the system diagnostics view.

Click on the GoTo button if the alarm view outputs an error message in the "Alarms" screen window in runtime.

A system diagnostics view has been added to the "SysDiag" screen window. The device with the associated error is opened in the detail view.

See also

Example: Procedures overview (Page 1844)

12.2 Monitoring functions for SIMATIC IPCs

Hardware monitoring for SIMATIC IPCs

All SIMATIC IPCs support monitoring functions. Some of the monitoring functions can be displayed in the TIA Portal if you connect to the IPC online. The diagnostics editor provides information on operating states of the following groups:

- Fans
- Temperatures
- Voltages
- Drives/Storage media

The following figure shows an overview of all groups in the diagnostics editor.

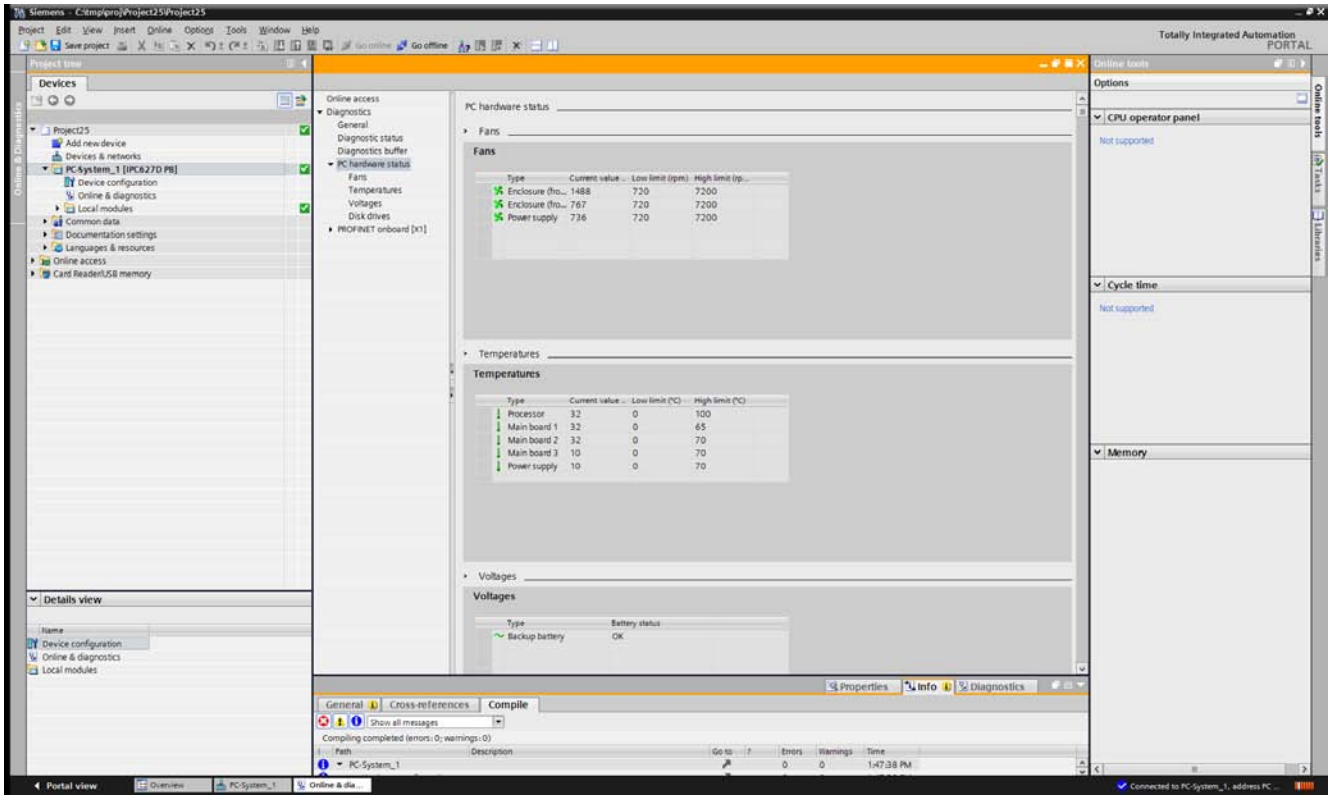

You can choose between the complete view and the single view in the diagnostics editor.

"Temperatures" group

The "Temperatures" group displays the temperatures of different components.

"Fans" group

The "Fans" group indicates the status of available fans (depending on the device). The speeds are given in revolutions per minute (rpm).

"Voltages" group

The "Voltages" group indicates the status of the available voltage sensors (depending on the device). The voltage of the CMOS battery is monitored. This sensor provides information about whether the voltage is OK, critical or faulty.

"Drives" group

The "Drives" group indicates the status of the available drives/storage media (depending on the device). The following storage media are supported here:

- ATA hard disks
- CompactFlash cards
- SCSI hard disks
- USB hard disks or USB sticks
- Hard disks of an INTEL RAID adapter

Information about whether each drive has S.M.A.R.T. capability and/or is a RAID drive is specified in the properties. Status information is displayed in this case.

Limits

High and low limits are defined for the two groups, Fans and Temperatures. The limits depend on the IPC and cannot be changed.

Color code

The icons next to each display object change color depending on the state of the object. The following applies to all groups:

- Green = OK
- Red = High or low limit violation or status of the object is critical.

Reaction to deviations

If there is a high or low limit violation or one of the groups has reached a critical state, an entry is triggered in the diagnostic buffer. At the same time, the icon color changes to red.

You can find additional information on the diagnostics functions of the IPCs in SIOS Portal (https://support.industry.siemens.com/cs/de/en/view/39129913).

12.3 Supervising machinery and plants with ProDiag (RT Professional, RT Advanced, Panels, Comfort Panels)

12.3.1 Basics of supervision with ProDiag (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

The TIA Portal ProDiag functionality is used for monitoring and identification of errors that occur in your plant or machine. You can use ProDiag to show the type of error, the cause of the error and the location of the error on the HMI device.

Use

You can use ProDiag functions to monitor your plant and to visualize it on an HMI device. The main objective of ProDiag is the reduction of downtime and loss of production after an error occurs, and the avoidance of errors using timely warnings. Diagnostic and display objects provide specific information for the user for troubleshooting and show the processes on an HMI device on site.

Principle

In STEP 7, you create operand supervisions and configure the settings according to your requirements. When an error occurs, a supervision alarm is generated based on the criteria you have configured. The configured supervision instances are stored in the preset ProDiag function block. You can use the automatically generated ProDiag FBs or create and configure your own ProDiag FBs.

You can use WinCC to visualize the ProDiag functions for display on an HMI device. You configure the screen objects which provide an overview of the current status of the plant and display the affected code or the step sequence when an error occurs.

12.3.2 Basics of supervising operands (RT Professional, RT Advanced, Panels, Comfort Panels)

Supervising operands

You can use a supervision to view the status of a Boolean tag. If an event occurs in your plant or machine, it triggers the configured supervision and the status of the operand is written to a ProDiag block in an error bit. The following types of supervision are available:

- Operand supervision
- Interlock monitoring
- Action supervision

- Reaction supervision
- Position supervision
- Text message
- Error message

Categories and subcategories

You can define categories and subcategories for each supervision instance. You can use eight categories to classify occurring events according to their urgency. Subcategories are used to further subdivide information. You can combine categories and subcategories in a supervision instance.

You can find categories, subcategories and alarm texts for the supervision in the project tree under "Common data > Supervision settings". You can find more information about supervisions and categories in the chapter "Programming PLC > Supervising machinery and plants with ProDiag".

Tracking a supervision in HMI

Track the status of the configured supervisions in WinCC using the "ProDiag overview" object.

When an event occurs in your plant, the corresponding symbols of the supervision category and the type of supervision light up in the ProDiag overview. You jump directly to the error message or to the display if the affected network in the PLC code view via the configured buttons.

You display the status of the GRAPH sequencer of a GRAPH data block in the object "GRAPH overview". If a GRAPH supervision error occurs in one of the steps, the error symbol in the GRAPH overview lights up and the faulty step is highlighted visually.

With the object "PLC code view", you show the user program in the LAD, FBD and GRAPH programming languages on your HMI device.

The initial value acquisition and criteria analysis functions enable you to record the values at the time of the error and to quickly identify the operands with error with the "Criteria analysis view" object.

Advantages

ProDiag enables you to configure supervisions and monitor your plant without changing the user program code.

You carry out plant diagnostics on your HMI device. The data is automatically synchronized in order to keep the display on your HMI device always up-to-date.

You can find additional information on the integrated diagnostics of your machines and plants with SIMATIC ProDiag on the Siemens YouTube channel:

Increased availability of your machines and plants with SIMATIC ProDiag (https://youtu.be/pri8m08g4Ak)

An application example for machine and plant diagnostics with ProDiag can be found in the Siemens Industry Online Support with Entry ID 109740151:

Machine and plant diagnostics with ProDiag (https://support.industry.siemens.com/cs/ww/en/view/109740151)

See also

Machine and plant diagnostics with ProDiag (https://support.industry.siemens.com/cs/ww/en/view/109740151)

12.3.3 Requirements and licensing (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You configure the ProDiag supervisions in the TIA Portal using STEP 7 Professional and create the screen objects for the supervision and diagnostics using WinCC. You need a license to use the ProDiag functionality and the corresponding screen objects.

Software requirements

To configure ProDiag supervisions, you need the following products:

- TIA Portal STEP 7 Professional
- TIA Portal WinCC
  - Comfort/Advanced V14 or higher or
  - Professional V14 or higher

Hardware requirements

ProDiag functionality is available for CPUs of the S7-1500 series with firmware version 2.0 or higher.

The objects for the supervision and diagnostics of plants are available for the following HMI devices:

- Comfort Panels und KTP Mobile Panels*
- RT Advanced
- RT Professional

* The "PLC code view" object is available on Comfort Panels and KTP Mobile Panels from display size 7"

Note

Objects for supervision and diagnostics of plants can be used under the "Full access" and "Read access" protection levels configured in the CPU.

ProDiag objects under the "HMI access" and "No access" protection levels cannot be used.

Licensing of ProDiag supervisions

The number of ProDiag supervisions that you configure using STEP 7 Professional is licensed. You do not need a license for the first 25 supervisions, licenses must be used for additional supervisions.

Number of supervisions	<= 25	<= 250	<= 500	<= 750	<= 1000	> 1000
Number of licenses	None	1	2	3	4	5

Licensing of ProjDiag objects

To use the objects for the diagnostics and supervision in conjunction with the ProDiag supervision in your program, you need a ProDiag license.

12.3.4 Visualizing supervisions (RT Professional, RT Advanced, Panels, Comfort Panels)

12.3.4.1 Objects for the supervision and diagnostics of plants (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

WinCC offers the following objects for displaying the current monitoring status and for fault diagnostics in the program code:

- ProDiag overview
- GRAPH overview
- PLC code view
- Criteria analysis view

ProDiag overview

The "ProDiag overview" object provides an overview of the current status of the configured supervisions.
The ProDiag overview is available on Comfort Panels, KTP Mobile Panels, RT Advanced and RT Professional.

GRAPH overview

The "GRAPH overview" object is used to display the current program status for executed steps of the GRAPH sequencer.
The GRAPH overview is available on Comfort Panels, KTP Mobile Panels, RT Advanced and RT Professional.

PLC code view

The "PLC code view" object is used to display the current program status of user programs that have been programmed in the LAD, FBD or GRAPH programming languages.

The PLC code view is available on Comfort Panels, KTP Mobile Panels (display size 7" or larger), RT Advanced and RT Professional.

Criteria analysis view

The "Criteria analysis view" object is used to display incorrect operands in the user program that were identified for a selected alarm by the criteria analysis.
The criteria analysis view is available on Comfort Panels, KTP Mobile Panels, RT Advanced and RT Professional.

12.3.4.2 Interaction of the objects for the supervision and diagnostics (RT Advanced, Panels, Comfort Panels)

Introduction

With the objects for the supervision and diagnostics of plants, you have the following options:

- On a HMI device, you show the status of the supervisions configured in the user program
- Evaluate the current alarms and display the network that causes the alarms,
- Display the program code on the HMI device

Depending on the network, you display the following information on the HMI device:

- With the object "ProDiag overview" you display the current status of the ProDiag supervisions.
- With the object "GRAPH overview" you display the current status of the GRAPH sequencer.
- With the "Criteria analysis view" object, you display the incorrect operands for a selected GRPAH or ProDiag alarm.

Interaction of ProDiag objects

The figure below shows the interaction of objects for visualizing the ProDiag supervisions:

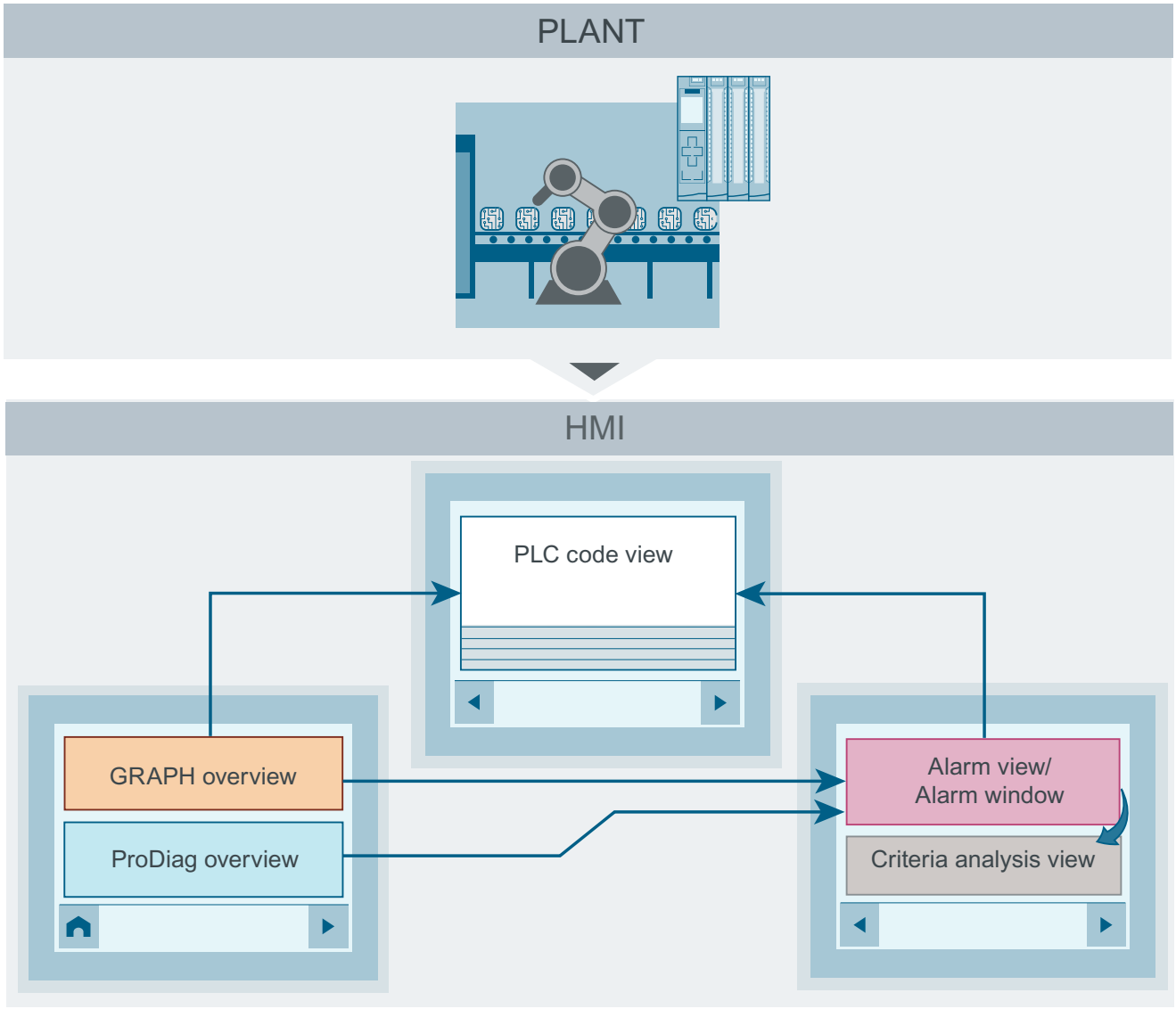

The visualization of the ProDiag functionality on the HMI device is comprised of the following actions:

1. Display program status
   - Display status of the supervisions with the ProDiag overview
     On the HMI device the connection is established to the PLC and the connected ProDiag FB and the current status of the supervisions is displayed. The corresponding symbol in the ProDiag overview lights up as soon as an event occurs.
     The "Alarm view" button allows you to jump to the alarm view or alarm window showing details on the event that occurred. To find the supervision-relevant alarms more easily in the alarm view/alarm window, it is recommended to use a filter to restrict the alarm shown.
   - Show sequence of the GRAPH sequencer using the GRAPH overview
     As soon as the connection is established to the PLC, the currently executed step of the GRAPH sequencer is shown in Runtime. Navigate to the next step or to the initial step of the sequencer using the operator buttons.
     The "Alarm view" button allows you to jump to the alarm view or alarm window showing details on the event that occurred.
     The "PLC code view" allows you to jump to the PLC code view, for example, to display an error in the execution of the GRAPH sequencer.
2. Display current alarms in the alarm view or in the alarm window
   Alarms that occur in Runtime are display in the alarm view or in the alarm window.

   Note

   Display of alarms of "Acknowledgement" and "No Acknowledgement" alarm classes

   To display the supervision alarms of the system alarm classes or self-defined alarm classes in your alarm view or in your alarm window, enable the display of the these alarm classes in the Inspector window under "Properties > Properties > General".

   In the alarm view/alarm window the alarm classes of the supervision alarms can only be enabled after the supervisions have been configured and the PLC compiled.

   The system function "ActivatePLCCodeView" allows you to jump from the last active or a selected alarm to the PLC code view with the affected program code.
   In addition, the initial value acquisition and criteria analysis functions allow you to acquire the values at the time of the error and to output the incorrect operands in the program in the "Criteria analysis view" object.
3. Display program code in the PLC code view
   As soon as the connection has been established to the PLC, the PLC code view displays the current status of the network depending on the configuration:
   - In Runtime, the LAD/FBD network and the program code is displayed. The symbol names, absolute addresses of the symbols and corresponding comments are show in the symbol table.
   - In Runtime, the GRAPH sequencer is shown. In the detail view, the transitions of the steps are displayed.

   You have the option of configuring a button that changes its appearance in case of an error and allows you to jump to the configured PLC code view. To do so, specify a button and configure the system function "ActivatePLCCodeView" to it. Dynamize the appearance (e.g. background color) of the button using the control tag for PLC code view that you previously

connected with the alarm view, so that the button changes its color, for example, in the event of an incoming ProDiag alarm.

Transfer of languages from CPU to HMI device

In the TIA Portal, the texts for the ProDiag functionality, for example, symbol names and comments in a PLC code view as well as the texts of the supervision alarms, are automatically loaded from the user program of the controller to the HMI device. In multi-lingual projects, make sure that the languages that are available in the PLC match the Runtime languages of the HMI device. The alarm texts on the HMI device can only be displayed in the desired languages if the languages match.

For more information on multilingual projects, refer to "Configuring devices and networks > Additional information on configurations > Functional description of S7-1500 CPUs > Setting the operating behavior > CPU properties > Multilingual capability".

12.3.4.3 Interaction of the objects for the supervision and diagnostics (RT Professional)

Introduction

With the objects for the supervision and diagnostics of plants, you have the following options:

- On a HMI device, you show the status of the supervisions configured in the user program
- Evaluate the current alarms and display the network that causes the alarms,
- Display the program code on the HMI device

Depending on the network, you display the following information on the HMI device:

- With the object "ProDiag overview" you display the current status of the ProDiag supervisions.
- With the object "GRAPH overview" you display the current status of the GRAPH sequencer.
- With the "Criteria analysis view" object, you display the incorrect operands for a selected GRPAH or ProDiag alarm.

Interaction of ProDiag objects

The figure below shows the interaction of objects for visualizing the ProDiag supervisions:

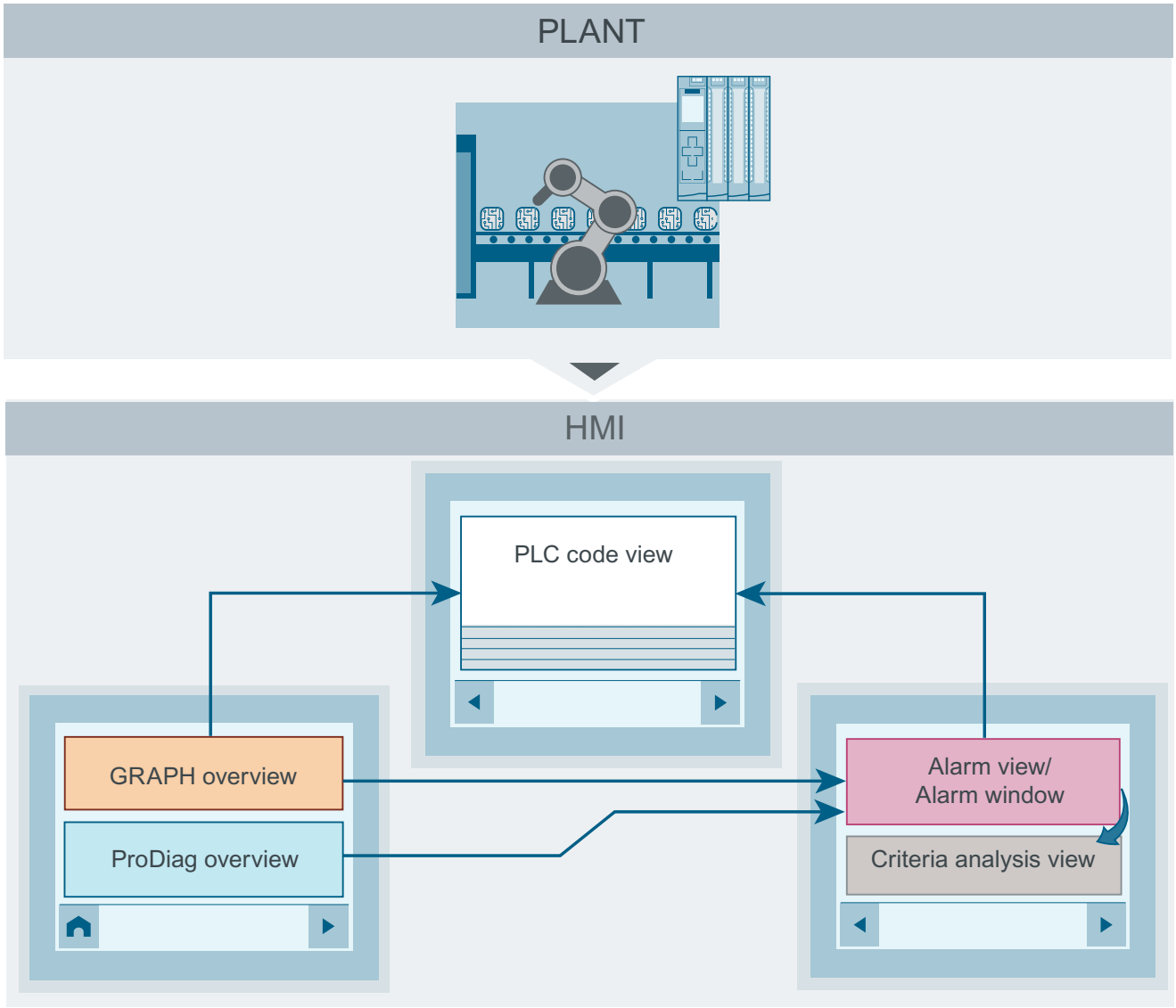
PLANT
HMI
PLC code view
GRAPH overview
ProDiag overview
Alarm view/
Alarm window
Criteria analysis view

The visualization of the ProDiag functionality on the HMI device is comprised of the following actions:

1. Display program status
   - Display status of the supervisions with the ProDiag overview
     On the HMI device the connection is established to the PLC and the connected ProDiag FB and the current status of the supervisions is displayed. The corresponding symbol in the ProDiag overview lights up as soon as an event occurs.
     The "Alarm view" button allows you to jump to the alarm view or alarm window showing details on the event that occurred. To find the supervision-relevant alarms more easily in the alarm view/alarm window, it is recommended to use a filter to restrict the alarm shown.
   - Show sequence of the GRAPH sequencer using the GRAPH overview
     As soon as the connection is established to the PLC, the currently executed step of the GRAPH sequencer is shown in Runtime. Navigate to the next step or to the initial step of the sequencer using the operator buttons.
     The "Alarm view" button allows you to jump to the alarm view or alarm window showing details on the event that occurred.
     The "PLC code view" allows you to jump to the PLC code view, for example, to display an error in the execution of the GRAPH sequencer.
2. Display current alarms in the alarm view or in the alarm window
   Alarms that occur in Runtime are display in the alarm view or in the alarm window.

   Note

   Display of alarms of "Acknowledgement" and "No Acknowledgement" alarm classes

   To display the supervision alarms of the system alarm classes or self-defined alarm classes in your alarm view or in your alarm window, enable the display of the these alarm classes in the Inspector window under "Properties > Properties > General".

   In the alarm view/alarm window the alarm classes of the supervision alarms can only be enabled after the supervisions have been configured and the PLC compiled.

   The system function "ShowPLCCodeViewFromAlarm" allows you to jump from the last active alarm or a selected alarm to the PLC code view with the affected program code. In addition, the initial value acquisition and criteria analysis functions allow you to acquire the values at the time of the error and to output the incorrect operands in the program in the "Criteria analysis view" object.
3. Display program code in the PLC code view
   As soon as the connection has been established to the PLC, the PLC code view displays the current status of the network depending on the configuration:
   - In Runtime, the LAD/FBD network and the program code is displayed. The symbol names, absolute addresses of the symbols and corresponding comments are show in the symbol table.
   - In Runtime, the GRAPH sequencer is shown. In the detail view, the transitions of the steps are displayed.

   You have the option of configuring a button that allows you to jump to the configured PLC code view in the case of an error. To do so, specify a button and configure the system function "ShowPLCCodeViewFromAlarm" to it.

Transfer of languages from CPU to HMI device

In TIA Portal V14, the texts for the ProDiag functionality, such as symbol names and comments in a PLC code view and supervision alarm texts, are automatically loaded from the PLC's user program to the HMI device. In multi-lingual projects, make sure that the languages that are available in the PLC match the Runtime languages of the HMI device. The alarm texts on the HMI device can only be displayed in the desired languages if the languages match.

For more information on multilingual projects, refer to "Configuring devices and networks > Additional information on configurations > Functional description of S7-1500 CPUs > Setting the operating behavior > CPU properties > Multilingual capability".

12.3.4.4 Configuring the ProDiag overview (RT Professional, RT Advanced, Panels, Comfort Panels)

Overview of the supervision configuration (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In TIA Portal, configure ProDiag supervision functions in your user program in STEP 7 and configure the visualization for the HMI devices in WinCC.

Requirements

- An S7-1500 controller has been created.
- There is at least one Boolean tag, either in a data block, tag table or in a block interface.
- An HMI device has been created
- A screen has been created

Steps for configuring the supervisions with ProDiag

You configure the ProDiag functionality in the TIA Portal in the following steps:

In STEP 7

1. Create ProDiag function block for supervision.
2. Define settings at the ProDiag function block.
3. Defining the ProDiag supervision settings
   - Specify the type of supervision
   - Set the delay time
   - You select the conditions by which a supervision signal is triggered
   - Define the categories and subcategories
4. Add supervision for a Boolean tag.
   You create either global or a local supervision for a Boolean tag.

5. Call the ProDiag function block.
6. Compile and download the project

In WinCC

1. Insert the "ProDiag overview" object in the HMI screen.
2. Select a status tag.
3. Configure the display and the representation of the object in the settings.
4. Configuring the jump to the alarm view.

In addition, you can have the program execution in the "GRAPH overview" object displayed, as well as the current program status in the "PLC code view" object.

Assignment of symbols in the ProDiag overview

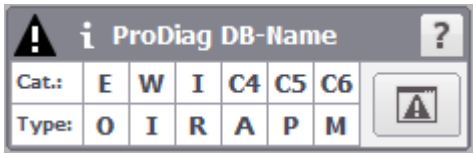

The status tag "State" from the PLC data type "SV_FB_State" of the ProDiag function block contains the group error bits for the supervision types and categories that you visualize with the help of the ProDiag overview.

You can visualize two types of group error bit in the "ProDiag overview" object:

- Group error bit for supervisions (All, O, I, R, A, P, Merr, Mtxt)
- Group error bit for categories (C1 to C6)

The table below shows the assignment of the group error bits of the status tag "State" to the symbols in the ProDiag overview.

ProDiag overview	Elements in ProDiag FB	Description
Symbols		
	All	Group error bit for all supervision types, excluding message text If value of All = TRUE, at least one supervision was triggered and the symbol is visually highlighted in the ProDiag overview.
	SV_Types	Byte for all group error bits of the supervision types
	Mtxt (message text)	Group error bit for all alarms
	-	Tooltip with information on this ProDiag overview
Categories		

ProDiag overview	Elements in ProDiag FB	Description
E (Error)	C1 (Category)	Error
W (Warning)	C2 (Category)	Warning
	Categories	Group error byte for all categories.
I (Info)	C3 (Category)	Information
C4 to C6	C4 to C6 (Category)	Additional categories
Supervision types		
O	O (Operand)	Group error bit for operand supervisions
I	I (Interlock)	Group error bit for interlock supervisions
A	A (Action)	Group error bit for action supervisions
R	R (Reaction)	Group error bit for reaction supervisions
P	P (Position)	Group error bit for position supervisions
M	Merr (Message error)	Group error bit for error texts

Configuring the ProDiag overview (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

ProDiag overview is used to monitor your machine or system at runtime and determination of diagnostics information in the event of a fault occurring.

Once you have set the status tag in the object and the connection to a ProDiag FB has been established, the status of the "State" status tag of the corresponding PLC data type is queried. In Runtime, the states of the monitored operands are represented as symbols in the ProDiag overview, similar to a traffic light colors.

In WinCC, you configure the display and the representation of categories and supervision types that are displayed in the "ProDiag overview" object independent of the supervision settings in STEP7.

Requirements

- At least one S7-1500 controller has been created.
- At least one supervision instance has been configured.
- A ProDiag FB and ProDiag DB are available.
- A PC station or an HMI device that supports the ProDiag functionality has been created.
- An HMI connection has been established between the controller and HMI device.
- An screen is created and the Inspector window is open.

Procedure

1. In the Inspector window, select "Properties > Properties > General".
2. Open the selection button under the "Tag" property.
3. Select the status tag of ProDiag FB.
   Alternatively, you can add the corresponding status tags from the detail view using drag-and-drop.

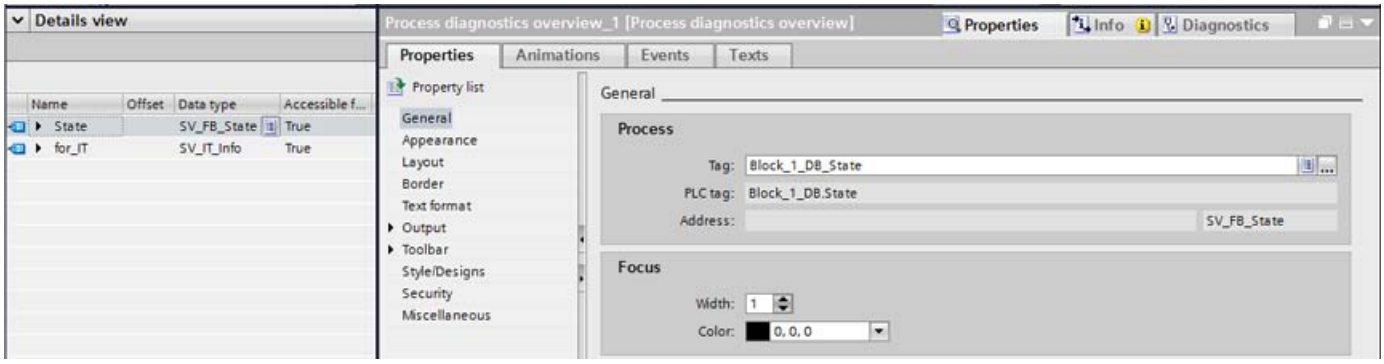

4. Under "Output > Categories", define the names and colors for the supervision categories.
5. Under "Output > Monitoring types", define the names and colors for the supervision categories.
6. Under "Layout", specify which icons and buttons are to be displayed in the object.
7. Select an authorization for operation under "Properties > Properties > Security".
8. You can configure a system function to the event "Alarm view button click" under "Properties > Events" to jump from the ProDiag overview to the alarm view in Runtime.

Result

The ProDiag overview is inserted in the screen. The current states for supervised events are displayed in Runtime.

Configuring jump to the alarm view (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

After an error has occurred, you have the option of jumping directly from the ProDiag overview to an alarm window or a screen with an alarm view.

Requirement

- You have created a screen.
- An alarm view or an alarm window has been created.
- The "ProDiag overview" object has been created.
- The Inspector window is open.

Configuring the jump to the alarm view

1. Click "Properties > Events" in the Inspector window.
2. Click on the event "Click alarm view button".
3. Select the "ShowAlarmWindow" system function under "<Add function".
4. Specify the object name of the alarm window.
5. Alternatively, open the screen with the configured alarm view using the "ActivateScreen" function.
   Specify the screen name of the screen that contains the configured alarm view.

Result

When you press the "Alarm view" button in the configured ProDiag overview, the screen with the corresponding alarm view or an alarm window opens automatically.

12.3.4.5 Configuring a GRAPH overview (RT Professional, RT Advanced, Panels, Comfort Panels)

Displaying the status of the GRAPH sequencer (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

With the help of the "GRAPH overview" object, you visualize the status of the GRAPH sequencer.

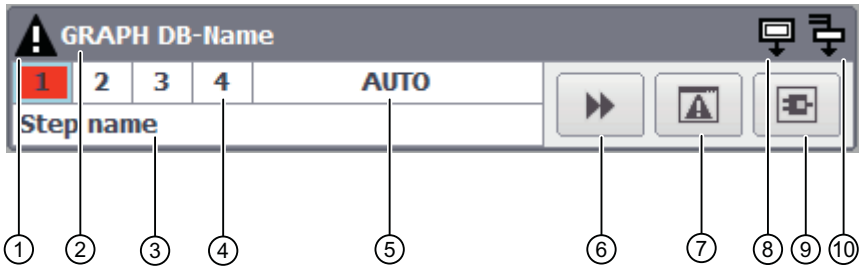

① Error symbol

② Name of the GRAPH data block

③ Name of the currently executed step

④ Active step of the sequencer

- If an error occurs in a step, the step number is highlighted in red.

⑤ Operating mode

⑥"Step" button

- If the sequencer is in the last step, the "Initial step" button is displayed.

⑦ "Alarm view" button

⑧ Initial step symbol

⑨ "PLC code view" button

⑩ Simultaneous step symbol

Displaying the status of the GRAPH sequencer

You connect the GRAPH overview to the PLC via the GRAPH DB tag. After the connection has been established to the PLC, the object shows the step numbers of the steps of the GRAPH sequencer. The name of the currently executed step is shown below the sequencer. If there are additional simultaneous steps besides the currently executed step, the simultaneous step symbol lights up in the GRAPH overview.

Depending on the application, you can switch between the four operating modes for executing the GRAPH sequencer. The currently selected mode is displayed in the GRAPH overview.

If an error occurs during the execution of a step, the faulty step and the error symbol are highlighted visually.

In the case of error you have the option of jumping to the alarm in the configured alarm view or of viewing the affected program in the PLC code view.

Single-line mode

You have the option of displaying the GRAPH overview in the standard view or in single line mode. You can, for example, configure multiple GRAPH overviews in a screen. This provides you with the overview of the current status of multiple production areas, whose supervisions are summarized in the corresponding GRAPH FBs.

Display of steps and criteria analysis

In extended mode you activate the additional lines for display of previous and next steps as well as the first operand of the criteria analysis with errors.

See also

GRAPH overview (Page 382)

Configuring a GRAPH overview (Page 1870)

Configuring the operating mode (Page 1872)

Interaction of the objects for the supervision and diagnostics (Page 1856)

Initial value acquisition and criteria analysis (Page 1915)

Show step history (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You have the option of displaying the previous and next steps of the GRAPH sequence in addition to the current step in the GRAPH overview. For this, the active steps are internally recorded after a screen with a GRAPH overview is called. If an error occurs in a step, you see the actions that were executed before and after the step with an error.

You specify the multilingual step names displayed in the GRAPH overview in a multilingual text list in the user program beforehand.

Marking of the steps

The GRAPH sequencer is evaluated from left to right. The step numbers in the GRAPH overview are additionally marked to indicate whether a simultaneous or alternative branch is present directly before or after this step. This allows you to see the following information on the program flow at a glance:

- Plus sign
  - Previous step: a simultaneous branch is present before the active step. Multiple steps were active simultaneously.
  - Next step: a simultaneous branch is present after the active step. More than one step will be activated.
- Question mark
  - Previous step: A different step than the first (left) step of the sequence could have been active before the active step.
  - Next step: an alternative branch is present after the active step. As a result, a different step than the first (left) step of the sequence may also become activated.
- No character: the previous step or next step is uniquely determined.

Note

Limitations affecting step determination

At the first call of the GRAPH overview, the previous and next steps are determined from the GRAPH block in the order of the programmed sequence.

At the first call of the GRAPH overview and after a screen change, no recording of the active steps exists, and a unique determination of the previous steps is thus not possible.

GRAPH overview updates the display at regular intervals. If the sequence is run through quickly or you manually switch between the steps frequently, a different step of the sequence than the step that was active may be determined.

Displaying previous and next steps

To display the previous and next steps in the GRAPH overview, select the "Previous and next step" option under "Properties > Appearance > Options".

Displaying the result of the criteria analysis

You have the option of displaying the result of the criteria analysis in the GRAPH overview when an error is present. The symbol name, the absolute address, the value of the first operand with error and the comment are displayed.

To display the first operand with error of the criteria analysis, select the "Criteria analysis" option under "Properties > Appearance > Options".

Note

No information will be output for the criteria analysis in the GRAPH overview in the following cases:

- The version of the GRAPH block is older than V4.0.
- The initial value acquisition was not activated in the block properties or the initial values were not acquired.
- There is no error in the current step.

Configuring a GRAPH overview (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can use the GRAPH overview to view the current program status for the executed steps of a GRAPH sequencer.

Requirement

- A PLC including a GRAPH instance data block has been created.
- GRAPH instance data block contains at least one tag which is visible in HMI and accessible from HMI.

  Note

  The process tag you are using for the GRAPH overview must be visible in HMI and accessible from HMI.

  To identify the tags of the GRAPH data block as visible and accessible for HMI, open the GRAPH function block, select the block in the work area, and select "Edit > Internal parameters visible/accessible from HMI" in the menu bar. Then compile the program blocks.

- An HMI device has been created.
- You have created a screen.
- The Inspector window is open.

Procedure

1. Drag the GRAPH overview from the toolbox view into the configured screen.
2. In the Inspector window, select "Properties > Properties > General".
3. Open the selection button under "Process > Tag".
   The "Add new object" dialog opens.
4. Select the corresponding PLC in the "Program blocks" folder.

5. Select the corresponding PLC tag of the GRAPH instance data block.

   Note

   If no connection was configured between the HMI device and the selected PLC, a connection is set up automatically.

   In addition, an HMI tag is created which is connected to the PLC tag.

6. Under "Properties > Properties > Layout > Options", specify the symbols that are displayed in the GRAPH overview .
   To represent the GRAPH overview in compatibility mode without toolbar buttons and operating mode display, enable the "Single-line mode" property.

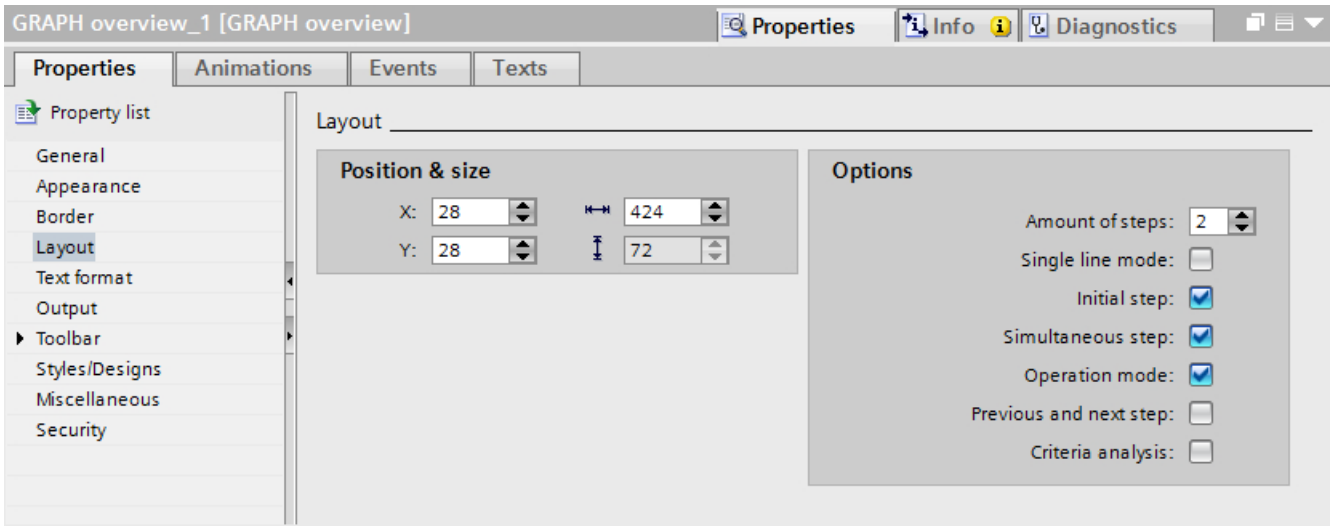

7. Under "Properties > Properties > Toolbar > General", specify the buttons that will be displayed in the object.
8. You can dynamize the properties in the "GRAPH overview" object using tags and color-code the states.
   You can find a detailed description of dynamizing a property in the section Dynamizing screens (Page 205)
9. Under "Properties > Events", you can assign system functions to the buttons in the GRAPH overview to jump to the alarm view and the PLC code view in Runtime.

Result

The GRAPH overview is inserted in the screen. The current status of the GRAPH step sequence is displayed in Runtime.

See also

Displaying the status of the GRAPH sequencer (Page 1866)

Dynamizing screens (Page 205)

Basics on dynamizing screens (Page 205)

Configuring the operating mode (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Four different operating modes are available for executing the GRAPH block:

- Auto (default setting)
  Automatically switches to the next step when the transition is fulfilled.
- TAP
  Automatically switches to the next step when the transition is fulfilled and there is an edge change from "0" to "1" at the T_PUSH parameter.
- TOP
  Automatically switches to the next step when the transition is fulfilled or there is an edge change from "0" to "1" at the T_PUSH parameter.
- Manual
  The next step is not automatically enabled when the transition is fulfilled. The steps can be selected and deselected manually.

Note

You set the operating mode by modifying the interface parameters of the GRAPH block in your control program.

Note

The name of the operating mode is language dependent and can be translated for the required Runtime languages.

In WinCC Runtime Professional you have the option of renaming the operating mode that is displayed in th GRAPH overview.

Configuring labels for the operating modes

A screen with the configured GRAPH overview is open.

1. In the Inspector window, select "Properties > Properties > Output".
2. Configure the labels for the four operating modes under "Output > Labeling".

12.3.4.6 Configuring the PLC code view (RT Advanced, Panels, Comfort Panels)

Configuring the PLC code view (RT Advanced, Panels, Comfort Panels)

Introduction

To display the PLC program networks in the LAD, FBD and GRAPH programming languages in Runtime, insert a PLC code view into your project.

Requirement

- At least one PLC has been created.
- An HMI device has been created.
- An HMI connection has been established between the controller and HMI device.
- You have created a screen.
- The Inspector window is open.

Procedure

1. Drag-and-drop the PLC code view from the toolbox view.
2. Click "Properties > Properties > Toolbar" in the Inspector window.
3. Select the buttons that you require in Runtime, for example: Next network, Previous network, Step mode.

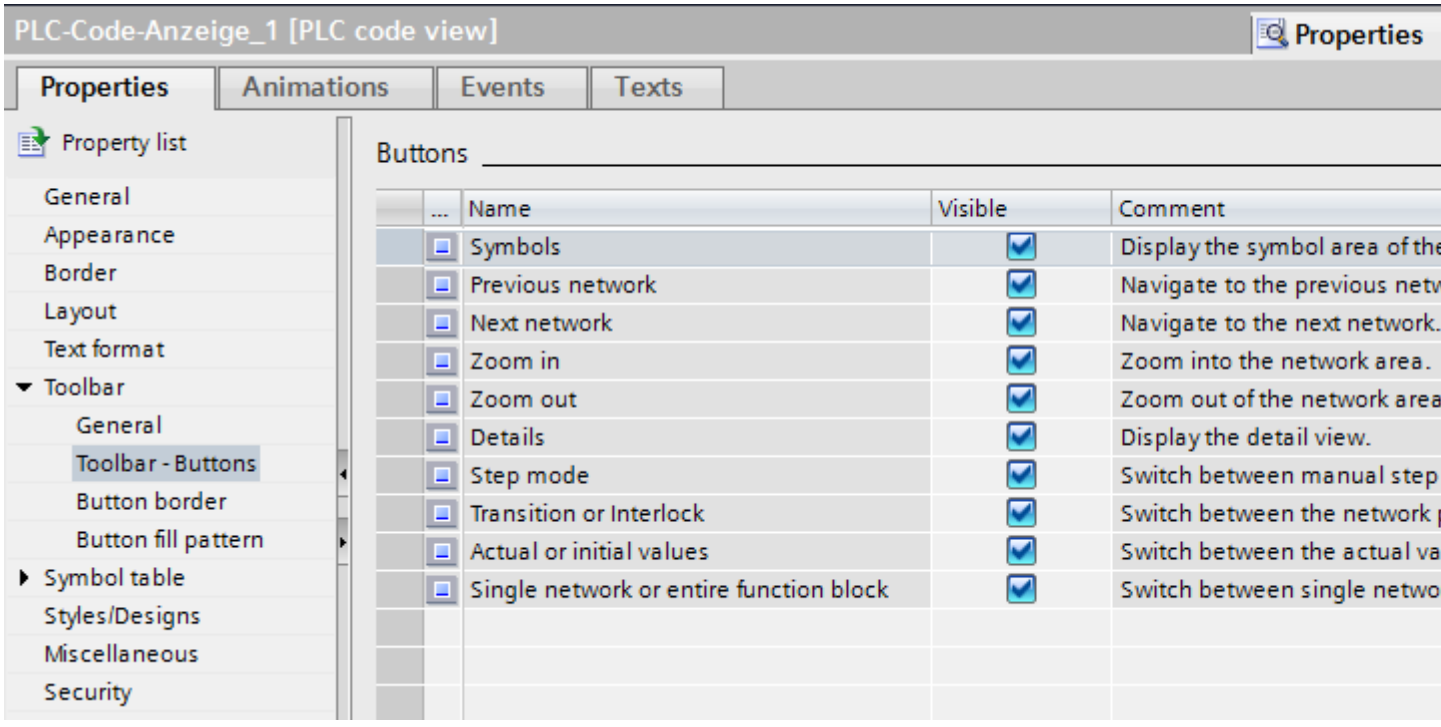

4. If required, configure the button display, e.g. background color and fill pattern.
5. In the Inspector window, select "Properties > Properties > Columns".
6. Adapt the column headers or change the order if necessary.
7. Select an authorization for operation in "Properties > Properties > Security".

Result

The PLC code view is inserted in the screen. In Runtime, PLC user programs that are programmed in the LAD, FBD or GRAPH programming languages can be displayed.

Views in the PLC code view (RT Advanced, Panels, Comfort Panels)

Introduction

In the PLC code view you display various items of information about the user program:

- Information area
- Symbol table
- Detail view
- "Transition/Interlock view"
- Initial value/actual value view

Information area

The information area displays the LAD/FBD program code of your user program, and/or the GRAPH sequencer of a GRAPH function block. After the connection to the PLC has been established, the network status is displayed in the information area highlighted via the program code.

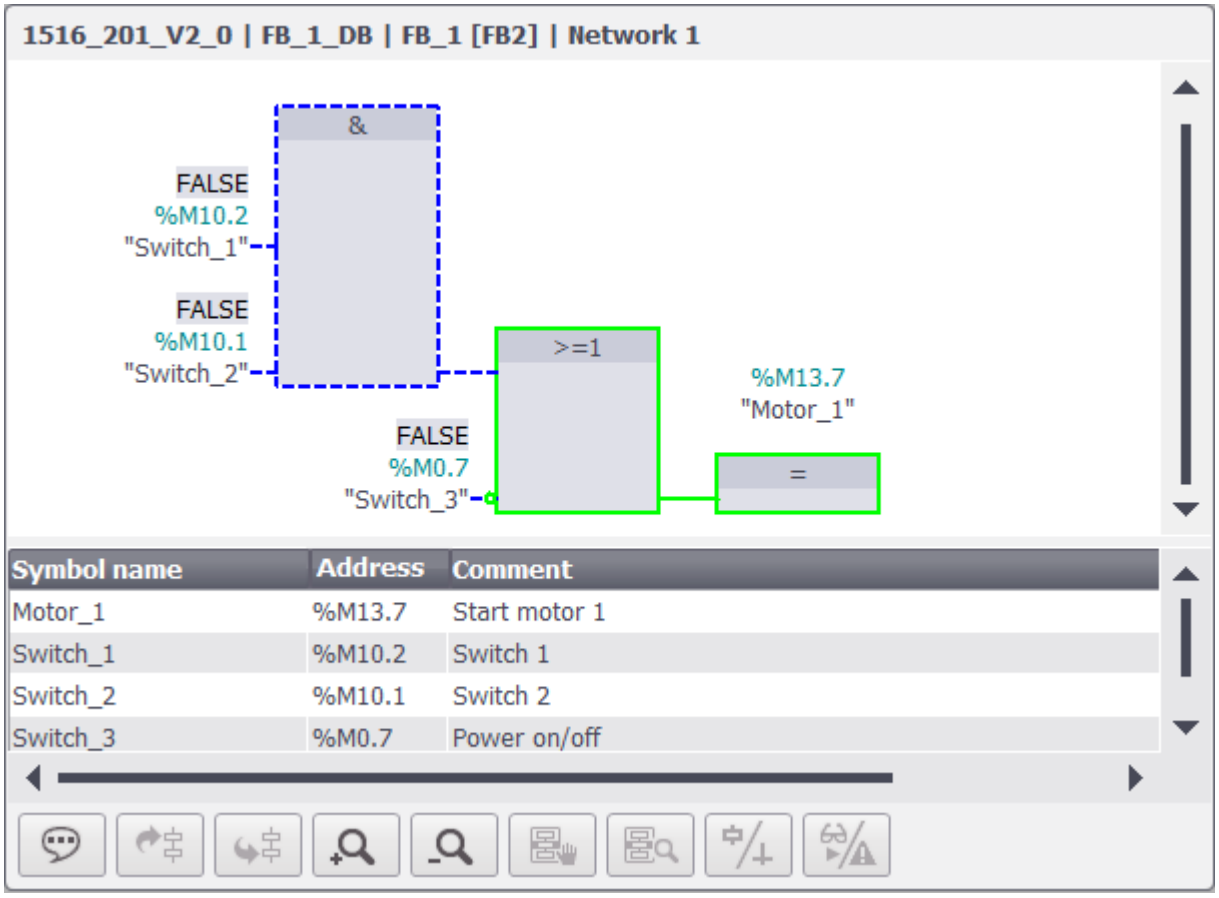

Symbol table

The symbol table displays the symbols that are used in the displayed network. The three columns of the symbol table display the symbol names of the tags, the absolute address and the comments. As the comments are loaded from the user program, these are displayed in the language that is stored in the PLC.

You can dynamically change the height and width of the symbol table in Runtime.

Detail view

The detail view is also available for displaying your GRAPH sequencer.

The detail view gives detailed information about the selected network and the pending errors. In the detail view you see, for example, the transition of the first active step of the GRAPH sequencer. Open the detail view in Runtime using the "Detail view" button.

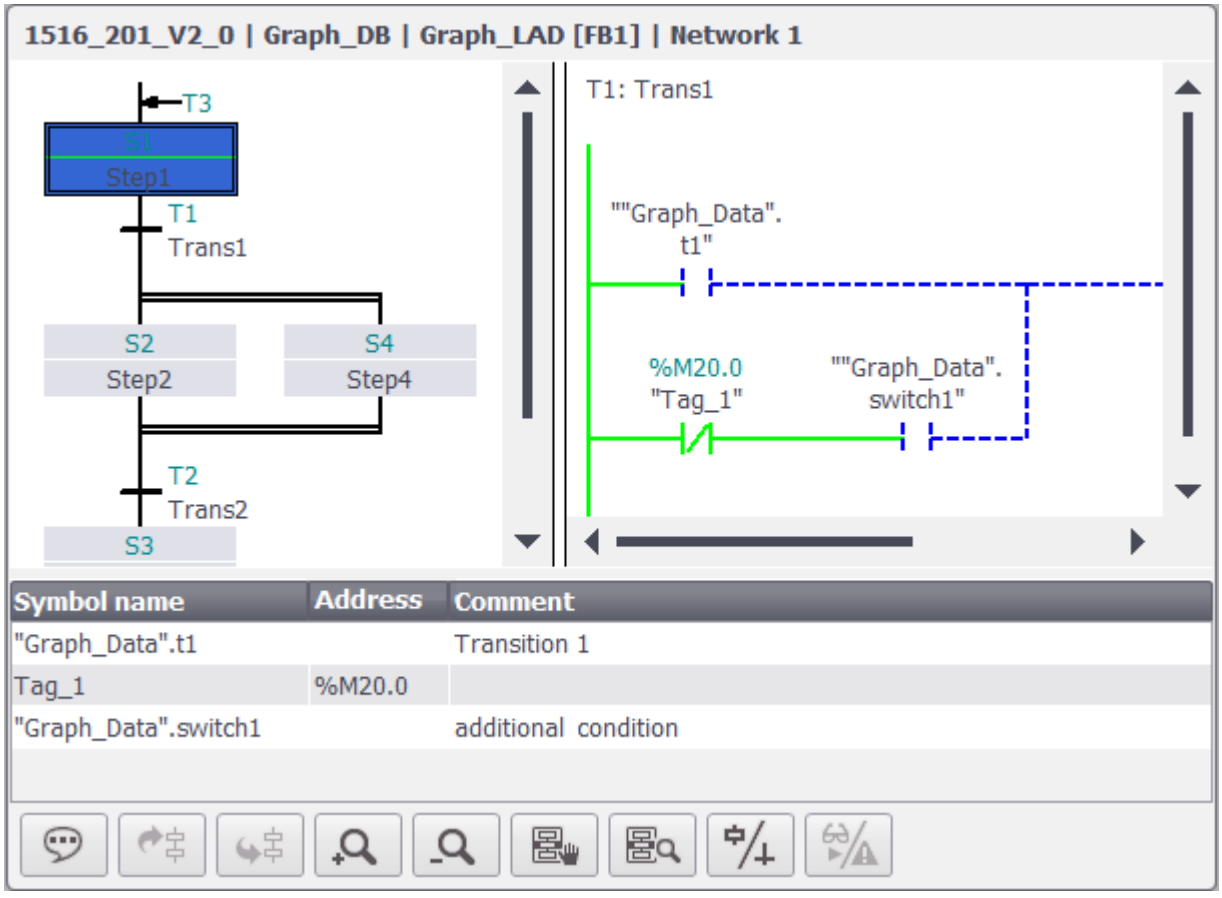

"Transition/Interlock view"

You have the option of switching between the transition and interlock views in the display of your user program. The transition/interlock view is available when the detail view is displayed.

To switch between the transition/interlock views in Runtime, use the "Transition/Interlock" button. When the button is enabled the interlock network is displayed, when disabled the transition network.

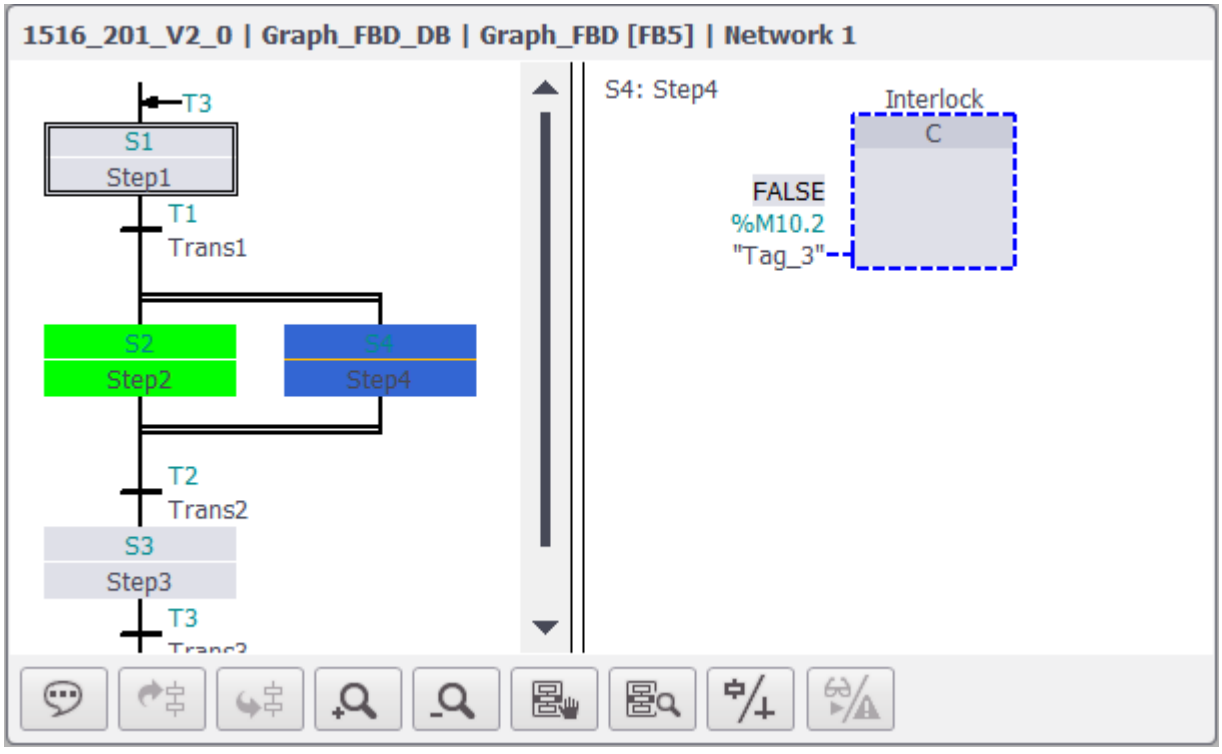

Initial value/actual value view

If you have activated actual value acquisition for the corresponding function block in the user program, you can switch between the initial value view and actual value view in the PLC code PLC code view. The actual value view uses the current values from the PLC and displays the current program status. The initial value view uses the values that were recorded at the time of the error and displays the operands with error and criteria. The operands with error are visually highlighted in the initial value view.

To switch between the actual values and initial values, use the "Initial values or actual values" button. When initial value acquisition is activated, the initial value view is displayed by default in the PLC code view after the jump.

In the initial value view, the operands with error are visually highlighted in Runtime in case of an error.

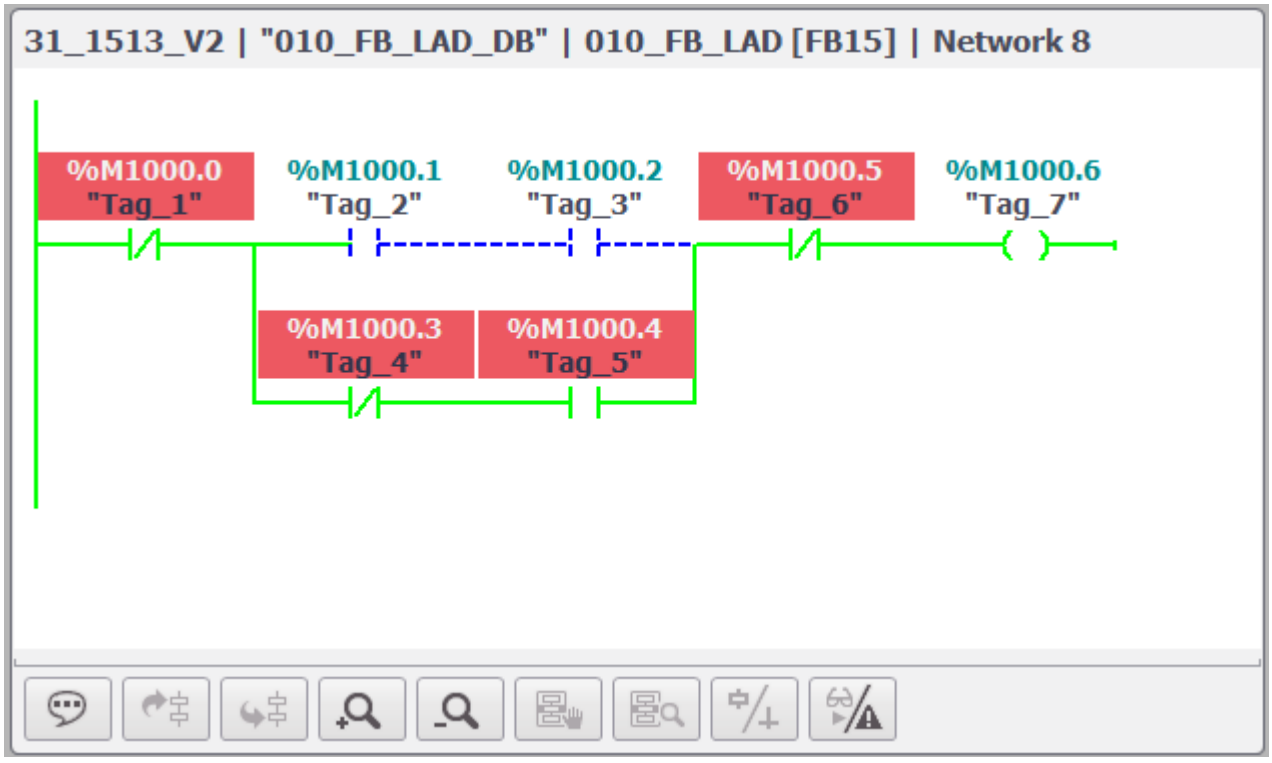

Buttons on the toolbar

The table below shows the buttons on the toolbar and their meaning.

Operator control	Designation	Function
	"Symbol area"	Shows the symbol area.
	"Previous network"	Navigates to the previous network
	"Next network"	Navigates to the next network
	"Zoom in"	Enlarges the information area.
	"Zoom out"	Reduces the information area.
	"Detail"	Shows the detail view.
	"Step mode"	Switches between manual and automatic step selection for the active step.

Operator control	Designation	Function
	"Transition or Interlock"	Switches between the transition and interlock network views.
	"Actual values or initial values"	Switches between actual value view and initial value view.

Configuring a view in the PLC code view (RT Advanced, Panels, Comfort Panels)

Introduction

The PLC code view also provides a split view of the display to show the GRAPH sequencer. You have the option of viewing the program and the associated details at a glance. Press "Detail view" to open the detail view in Runtime.

In WinCC, you can define the size ratio between the information area and the detail view.

Requirement

- You have created a screen.
- A PLC code view has been created in the screen.
- The Inspector window is open.

Define width of detail view

1. Click "Properties > Properties > Layout" in the Inspector window.
2. Specify the width of the detail view in relation to the information area using the option "GRAPH detail view width (%)".

Result

The PLC code view is divided in two areas to represent the GRAPH sequencer in Runtime. The left-hand area shows the GRAPH sequencer and the right-hand area shows the detail view with the transition of the active step.

Supported instructions (RT Advanced, Panels, Comfort Panels)

Introduction

The "PLC code view" object supports instructions in the FBD and LAD programming languages.

Instructions

The following instructions are supported in the PLC code view:

- Bit logic operations
- Comparator operations
- Timers
- Counters

Bit logic operations

Instruction	LAD	FBD
Normally closed contact		“Tag_1“
Normally open contact		“Tag_1“
Set output	“Tag_1“ (S)	“Tag_1“ S
Reset output	“Tag_1“ (R)	“Tag_1“ R
Assignment	“Tag_1“ ( )	“Tag_1“ =
NOT: Invert RLO	NOT	“Tag_1“ -
Negated contact	“Tag_1“ (/)	“Tag_1“ /=

Instruction	LAD	FBD
SR: Set/reset flip-flop	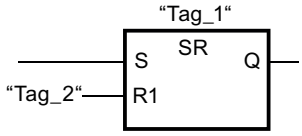	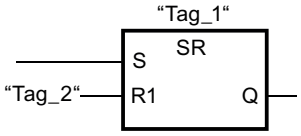
RS: Reset/set flip-flop	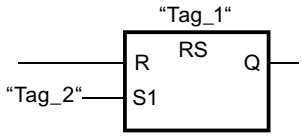	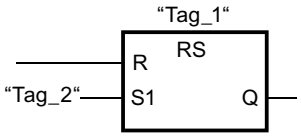
AND	Is implemented by the corresponding interconnection of normally closed contacts or normally open contacts.	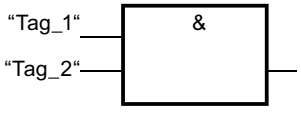
OR	Is implemented by the corresponding interconnection of normally closed contacts or normally open contacts.	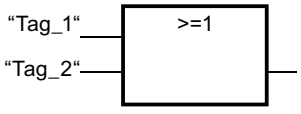
EXCLUSIVE OR	Is implemented by the corresponding interconnection of normally closed contacts or normally open contacts.	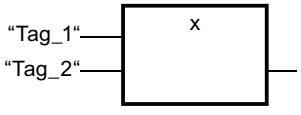

You can find additional information on this in the section "Programming PLC > Instructions".

Comparator operations

Instruction	LAD	FBD
CMP ==: Equal	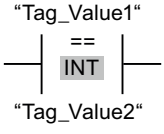	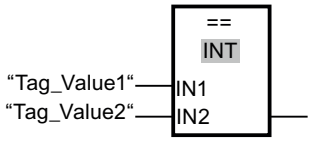
CMP <>: Not equal	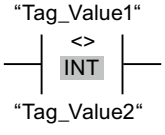	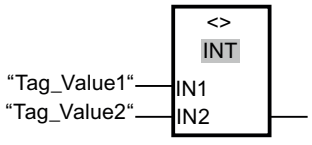
CMP >=: Greater or equal	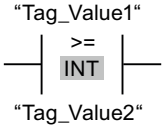	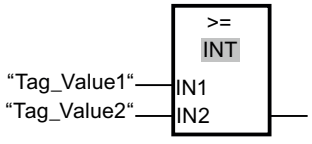

Instruction	LAD	FBD
CMP <=: Less or equal	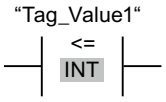	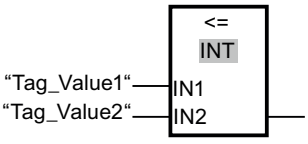
CMP >: Greater than	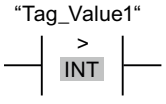	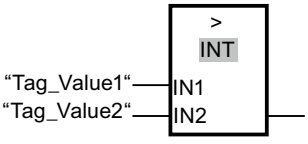
CMP <: Less than	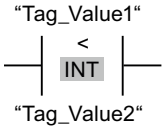	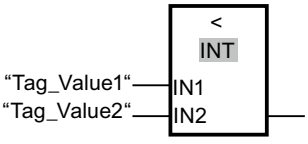

Note

PLC code view supports the following data types for displaying the comparator: Binary numbers, integers, floating-point numbers and timers with exception of S5TIME.

You can find additional information on this in the section "Programming PLC > Instructions".

Timers

Instruction	LAD	FBD
TP: Generate pulse	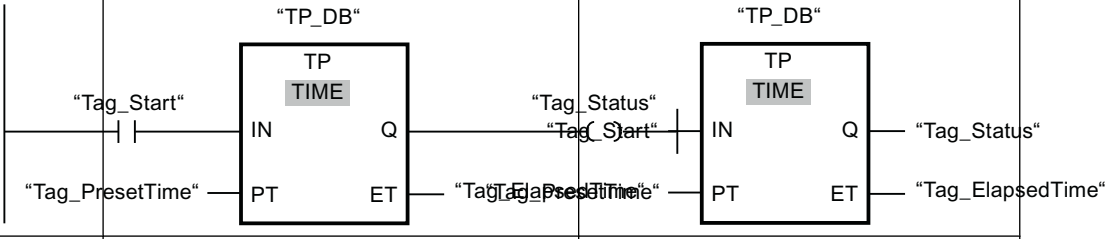	
TON: Generate on-delay	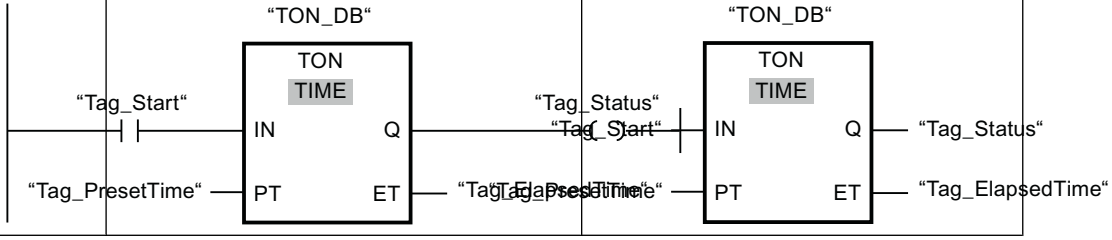	

Instruction	LAD	FBD
TOF: Generate off-delay	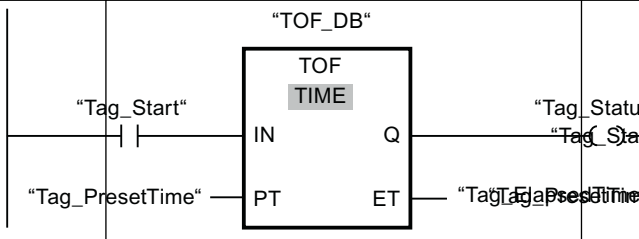	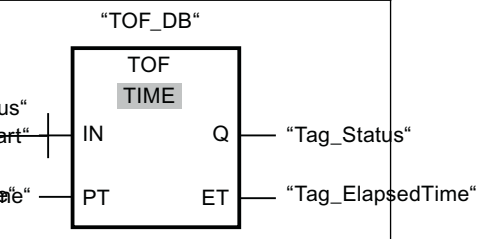
TONR: Time accumulator	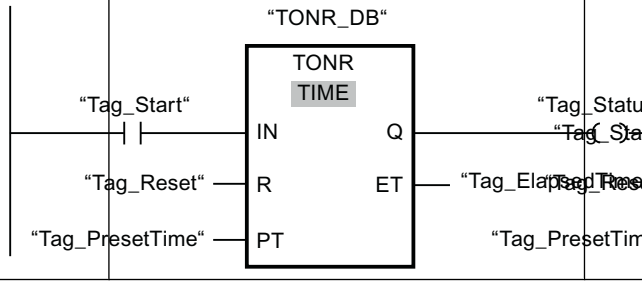	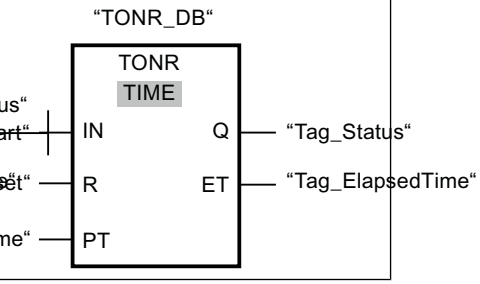

You can find additional information on this in the section "Programming PLC > Instructions".

Counters*

* Counters support the integer data types

Instruction	LAD	FBD
CTU: Count up	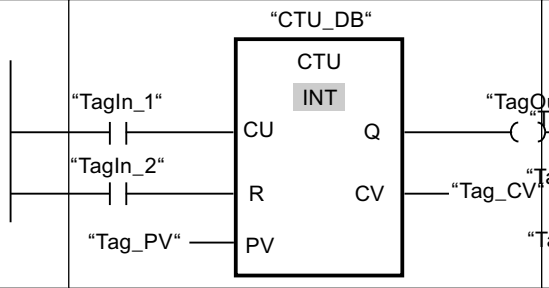	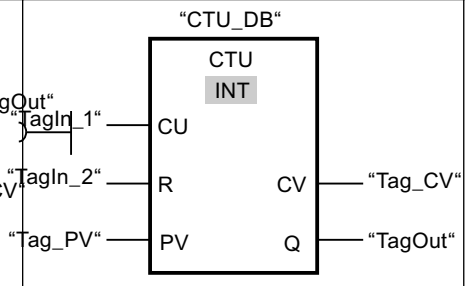
CTD: Count down	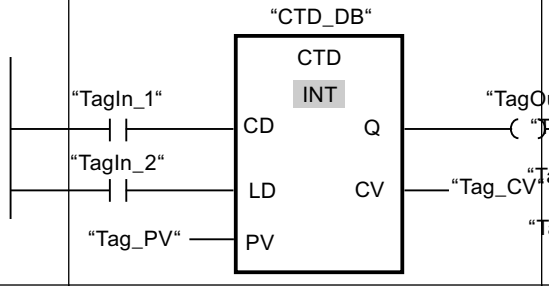	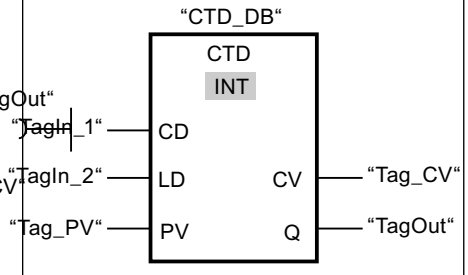
CTUD: Count up and down	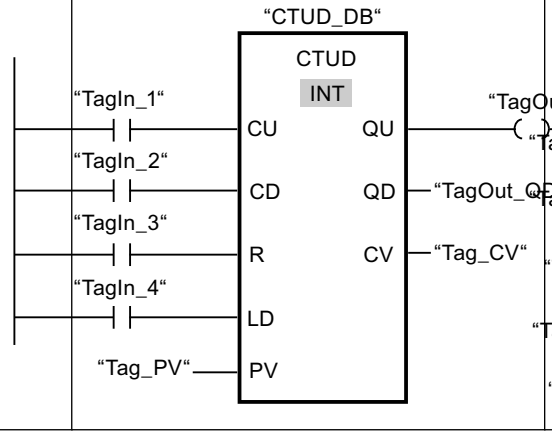	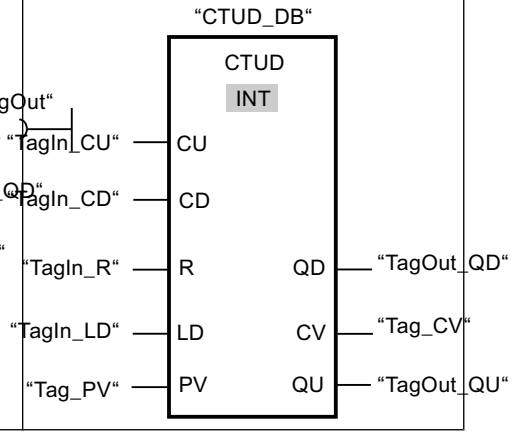

You can find additional information on this in the section "Programming PLC > Instructions".

Supported data types (RT Advanced, Panels, Comfort Panels)

Supported data types

The following table shows data types supported in the "PLC code view" object and their display formats:

Data type	Length	Format
Binary numbers		
BOOL	1 bit	Bool
BYTE	8-bit	Hex

Data type	Length	Format
WORD	16-bit	Hex
DWORD	32-bit	Hex
LWORD	64-bit	Hex
Integers		
SINT	8-bit	Dec
USINT	8-bit	Dec
INT	16-bit	Dec
UINT	16-bit	Dec
DINT	32-bit	Dec
UDINT	32-bit	Dec
LINT	64-bit	Dec
ULINT	64-bit	Dec
Floating-point numbers		
REAL	32-bit	Floating point number
LREAL	64-bit	Floating point number
Timers		
TIME	32-bit	Time
LTIME	64-bit	LTime
Date and time		
DTL**	12 bytes	Date and Time
* Only with Boolean operands ** Complete DTL structures are not supported. Only single elements of DTL structures are supported.		

Note

You can find more information on the data types under "PLC Programming > Data Types".

Note

The 64-bit PLC data types LINT, ULINT and LWORD are mapped to the HMI data type LREAL in the HMI channel. A loss of accuracy can be expected when the value exceeds the size 2^50.

Restrictions for the PLC code view (RT Advanced, Panels, Comfort Panels)

Operands and UDTs

Operands that are declared in the "#Temp" or "#InOut" area are generally not supported by the PLC code view. This applies both to elementary data types and to data types that are contained in UDTs. Data types of a UDT can be declared in the "#In" and "#Out" area and displayed in the PLC code view. The same limitations as for elementary data types apply to the data types of the UDT.

Restrictions to data types

The use of the data types STRING, WSTRING, CHAR, WCHAR, S5TIME results in the network not being able to be displayed in the PLC code view.

Use of 64-bit data types

The use of 64-bit data types may result in a slight loss of accuracy as these data types are mapped in the HMI channel to the data type Double. It is therefore possible that integer data types are displayed with decimal places.

Addressing of tags

Please note the following restrictions when addressing tags:

- The "PLC code view" object only supports symbolic addressing of tags. If the operand is not addressed symbolically, the network with the operand cannot be displayed and an error message is generated.
- The use of array elements with a tag as index is supported, e.g. #myArray[MyTag]. Please use numerical values when addressing array elements, for example #myArray[6].
- Slice access enables you to address specific areas within declared variables. With Slice access, PLC code view only supports the access width "Bit" to Boolean tags.

Jump to the PLC code view

For the jump from a supervision alarm to the PLC code view, the instance name must conform to the following naming convention when using supported local operands in a function block: <FB-Name>_DB.

The jump to a function or an organization block is only possible if only global operands are used.

12.3.4.7 Configuring the PLC code view (RT Professional)

Configuring the PLC code view (RT Professional)

Introduction

To display the PLC program networks in the LAD, FBD and GRAPH programming languages in Runtime, insert a PLC code view into your project.

Requirement

- At least one PLC has been created.
- A PC station has been created.
- An HMI connection has been established between the controller and HMI device.

- You have created a screen.
- The Inspector window is open.

Procedure

1. Drag-and-drop the PLC code view from the toolbox view.
2. Click "Properties > Properties > Layout" in the Inspector window.
3. Under "Layout > Symbol table lines", you adjust the number of lines that are displayed in the symbol table.
4. Click "Properties > Properties > Toolbar" in the Inspector window.
5. Assign a hotkey to the button that you require in Runtime, e.g. Step mode.

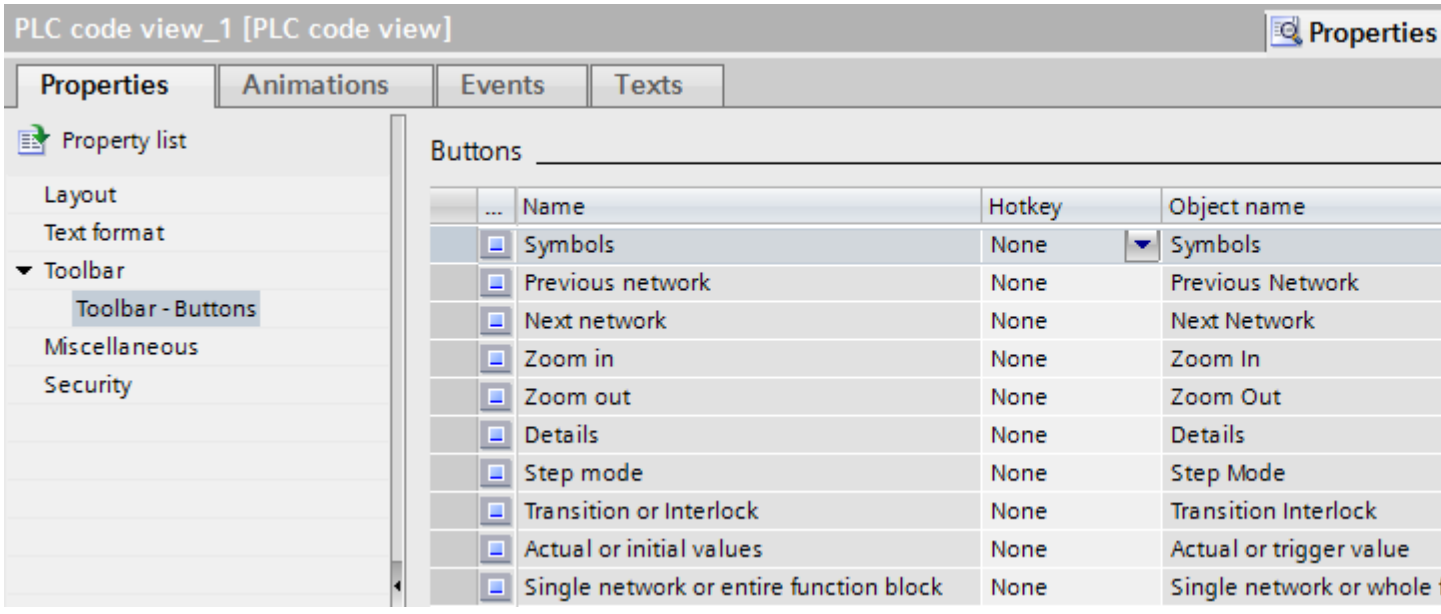

6. To display or handle errors in the configuration of the display of a network in Runtime, configure a corresponding script under Properties > Events > Errors".

Result

The PLC code view is inserted in the screen. In Runtime, PLC user programs that are programmed in the LAD, FBD or GRAPH programming languages can be displayed.

Views in the PLC code view (RT Professional)

Introduction

In the PLC code view you display various items of information about the user program:

- Information area
- Symbol table

- Detail view
- "Transition/Interlock view"
- Initial value/actual value view

Information area

The information area displays the LAD/FBD program code of your user program, and/or the GRAPH sequencer of a GRAPH data block. After the connection to the PLC has been established, the network status is displayed in the information area highlighted via the program code.

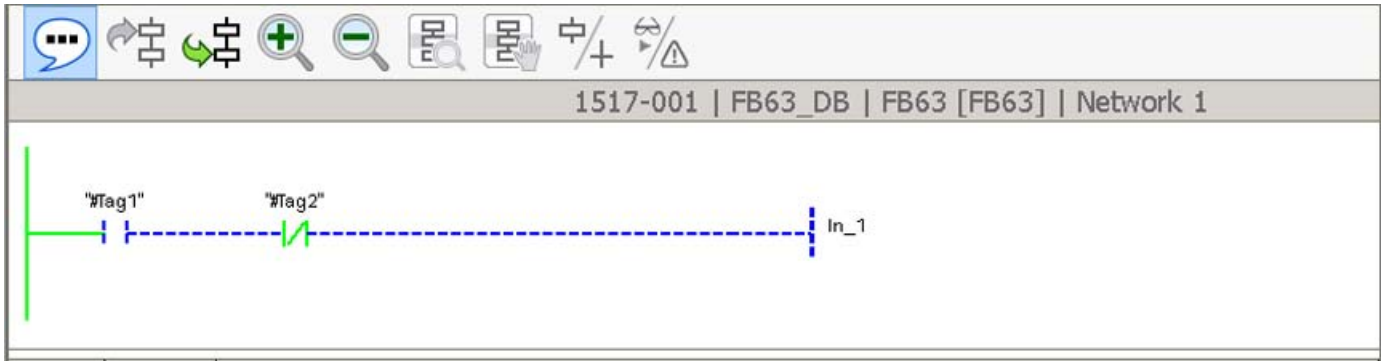

Name	Address	Comment
#Tag1		Tag 1
#Tag2		Tag 2
In_1		Input pin 1

Symbol table

The symbol table shows the symbols that are used in the displayed network. The three columns of the symbol table display the symbol names of the tags, the absolute address and the comments. As the comments are loaded from the user program, these are displayed in the language that is stored in the PLC.

You can dynamically change the height and width of the symbol table in Runtime.

Detail view

The detail view gives detailed information about the selected network and the pending errors. In the detail view you see, for example, the transition of the first active step of the GRAPH sequencer. Open the detail view in Runtime using the "Detail view" button.

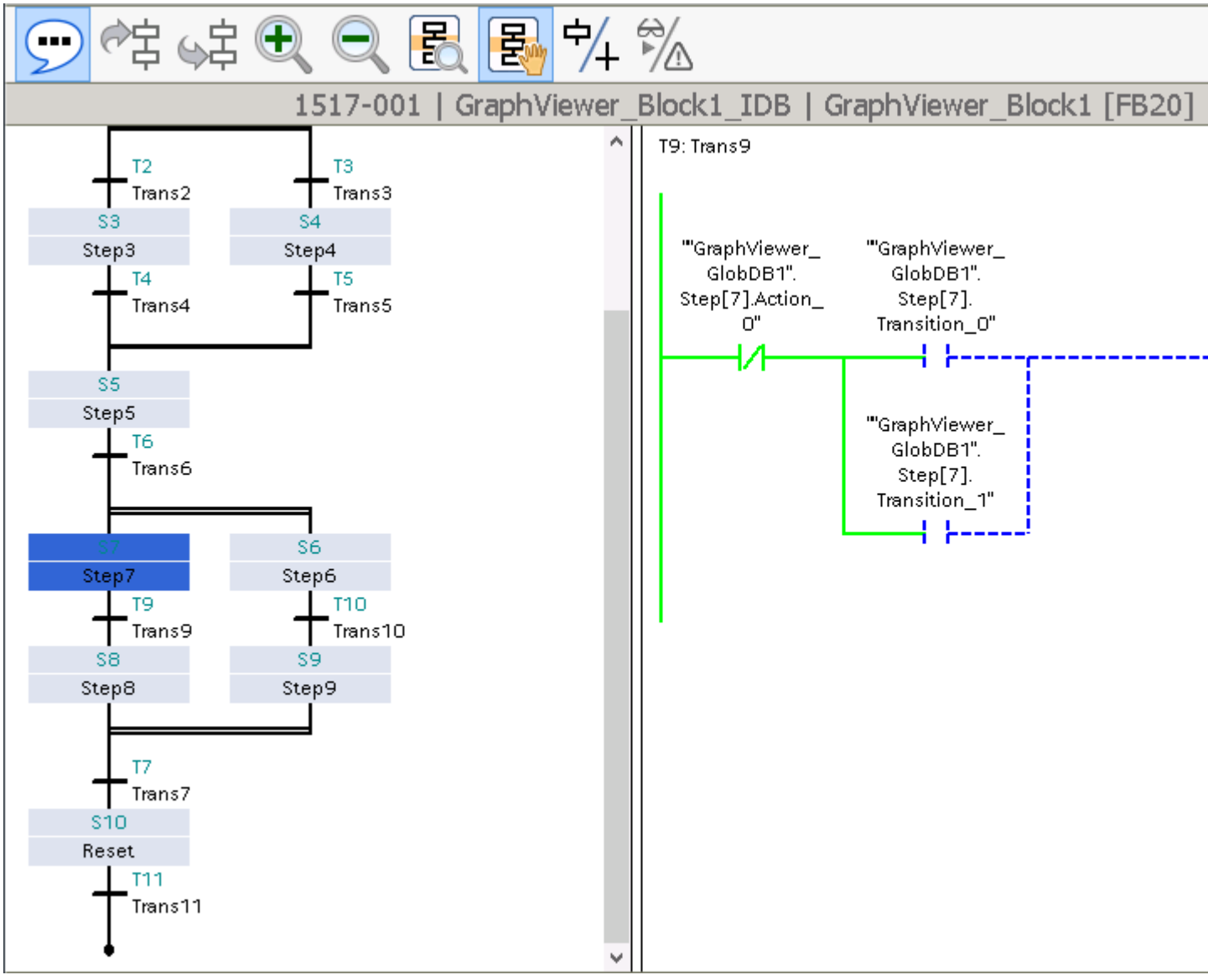

"Transition/Interlock view"

You have the option of switching between the transition and interlock views in the display of your LAD/FBD user program. The transition/interlock view is available when the detail view is displayed.

To switch between the transition/interlock views in Runtime, use the "Transition/Interlock" button. When the button is enabled the interlock network is displayed, when disabled the transition network.

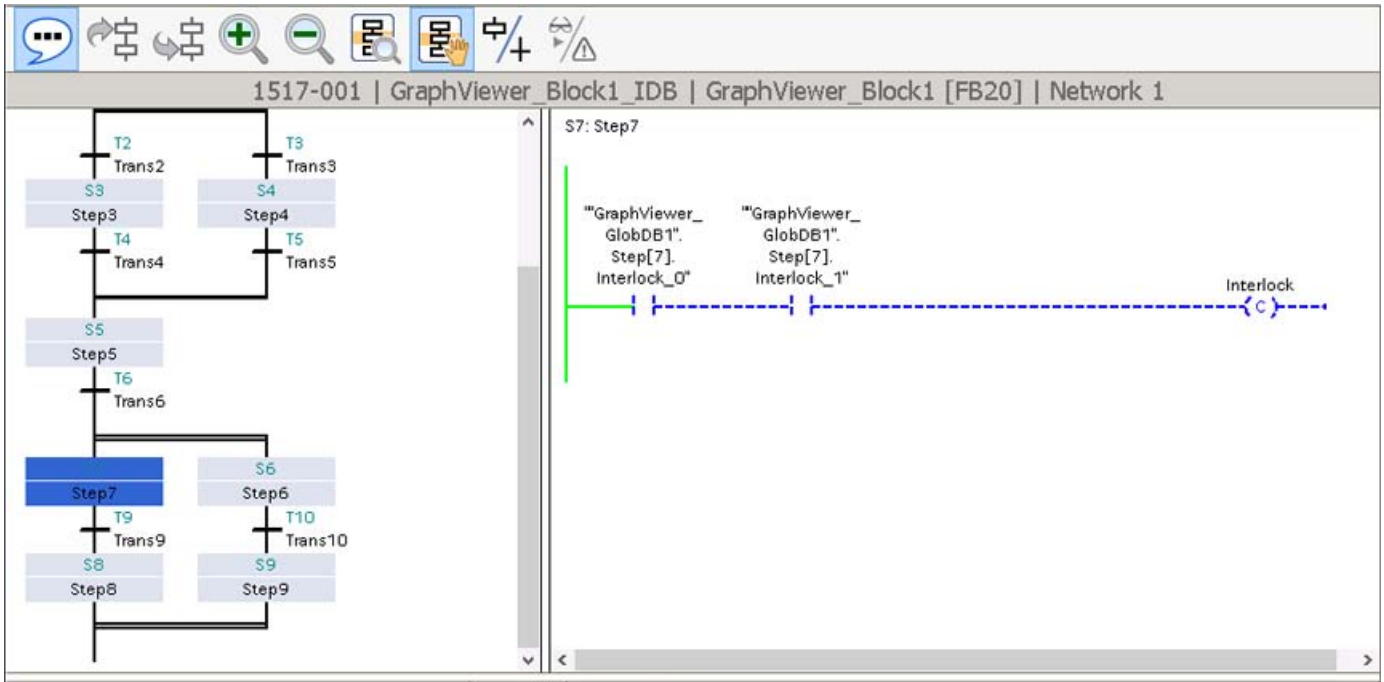

Name	Address	Comment
"GraphViewer_GlobDB1".Step[7].Interlock_0		Interlock_0
"GraphViewer_GlobDB1".Step[7].Interlock_1		Interlock 1

Initial value/actual value view

If you have activated actual value acquisition for the corresponding function block in the user program, you can switch between the initial value view and actual value view in the PLC code PLC code view. The actual value view uses the current values from the PLC and displays the current program status. The initial value view uses the values that were recorded at the time of the error and displays the operands with error and criteria. The operands with error are visually highlighted in the initial value view.

To switch between the actual values and initial values, use the "Initial values or actual values" button. When initial value acquisition is activated, the initial value view is displayed by default in the PLC code view after the jump.

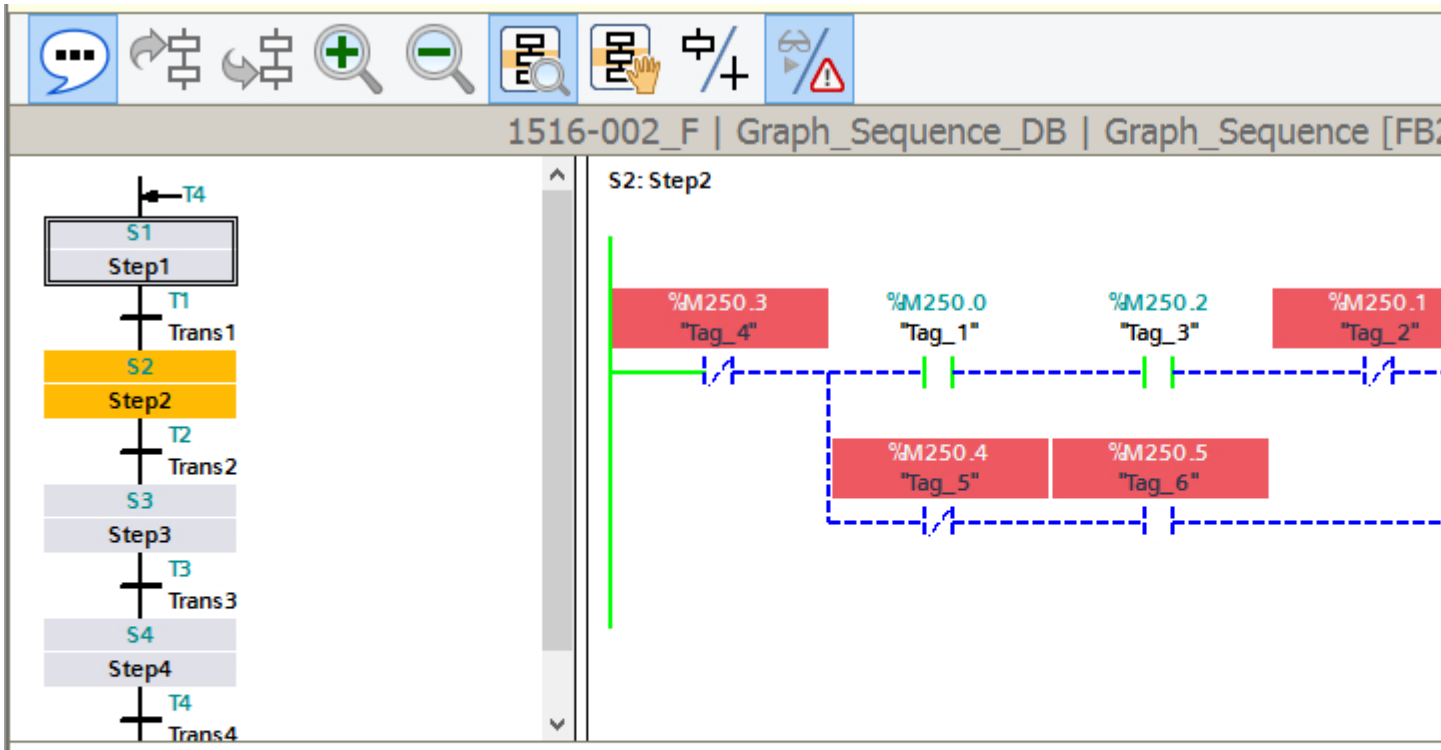

Name	Address	Comment
Tag_4	%M250.3	Interlock_2
Tag_2	%M250.1	Interlock_1
Tag_5	%M250.4	Hand mode
Tag_6	%M250.5	Hand switch

Buttons on the toolbar

The table below shows the buttons on the toolbar and their meaning.

Operator control	Designation	Function
	"Symbol area"	Shows the symbol area.
	"Previous network"	Navigates to the previous network
	"Next network"	Navigates to the next network
	"Zoom in"	Enlarges the information area.

Operator control	Designation	Function
	"Zoom out"	Reduces the information area.
	"Detail"	Shows the detail view.
	"Step mode"	Switches between manual and automatic step selection for the active step.
	"Transition or Interlock"	Switches between transition view and interlock view.
	"Actual values or initial values"	Switches between actual value view and initial value view.

Supported instructions (RT Professional)

Introduction

The "PLC code view" object supports instructions in the FBD and LAD programming languages.

Instructions

The following instructions are supported in the PLC code view:

- Bit logic operations
- Comparator operations
- Timers
- Counters

Bit logic operations

Instruction	LAD	FBD
Normally closed contact		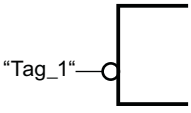
Normally open contact		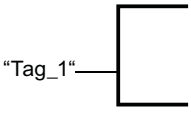

Instruction	LAD	FBD
Set output	“Tag_1“ (S)	“Tag_1“ S
Reset output	“Tag_1“ (R)	“Tag_1“ R
Assignment	“Tag_1“ ( )	“Tag_1“ =
NOT: Invert RLO	NOT	“Tag_1“ -
Negated contact	“Tag_1“ (/)	“Tag_1“ /=
SR: Set/reset flip-flop	“Tag_1“ SR S Q “Tag_2“ R1	“Tag_1“ SR S “Tag_2“ R1 Q
RS: Reset/set flip-flop	“Tag_1“ RS R Q “Tag_2“ S1	“Tag_1“ RS R “Tag_2“ S1 Q
AND	Is implemented by the corresponding interconnection of normally closed contacts or normally open contacts.	“Tag_1“ “Tag_2“ &
OR	Is implemented by the corresponding interconnection of normally closed contacts or normally open contacts.	“Tag_1“ “Tag_2“ >=1
EXCLUSIVE OR	Is implemented by the corresponding interconnection of normally closed contacts or normally open contacts.	“Tag_1“ “Tag_2“ x

You can find additional information on this in the section "Programming PLC > Instructions".

Comparator operations

Instruction	LAD	FBD
CMP ==: Equal	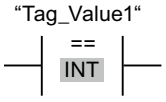	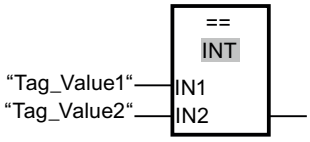
CMP <>: Not equal	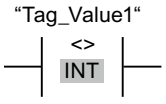	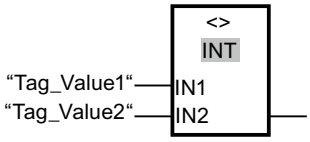
CMP >=: Greater or equal	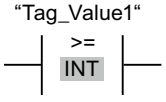	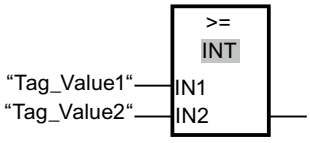
CMP <=: Less or equal	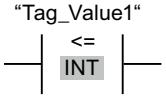	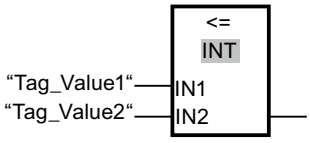
CMP >: Greater than	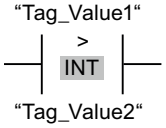	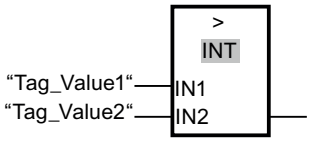
CMP <: Less than	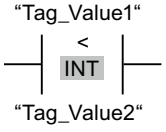	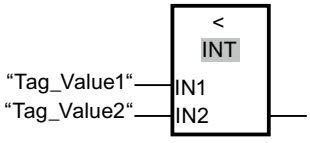

Note

PLC code view supports the following data types for displaying the comparator: Binary numbers, integers, floating-point numbers and timers with exception of S5TIME.

You can find additional information on this in the section "Programming PLC > Instructions".

Timers

Instruction	LAD	FBD
TP: Generate pulse	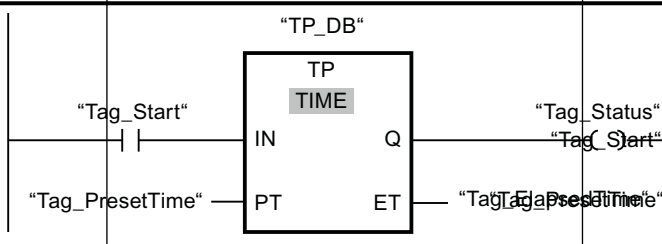	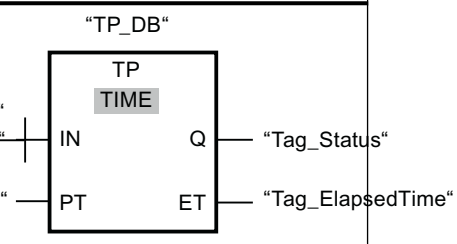
TON: Generate on-delay	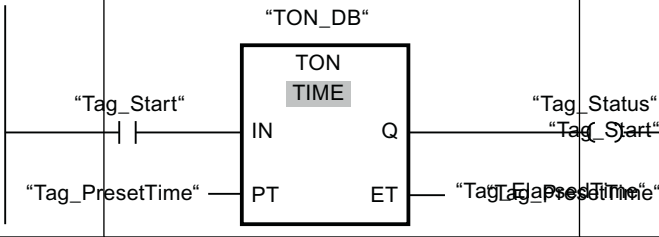	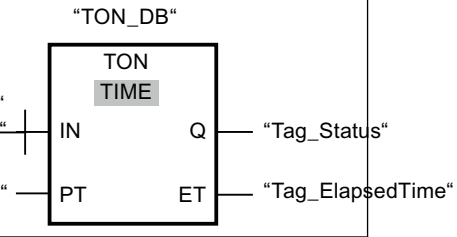
TOF: Generate off-delay	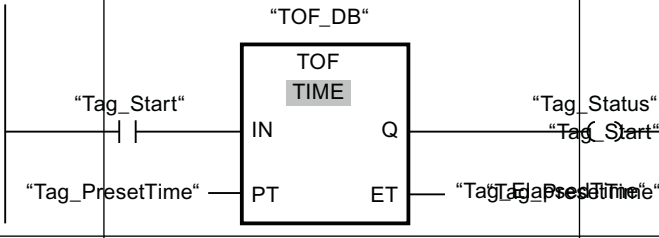	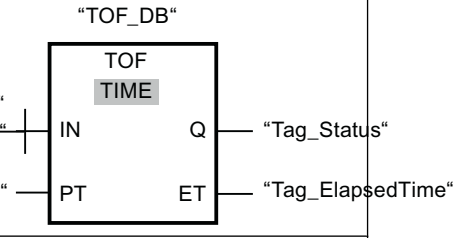
TONR: Time accumulator	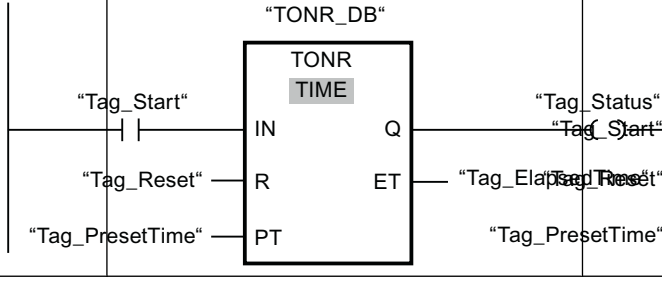	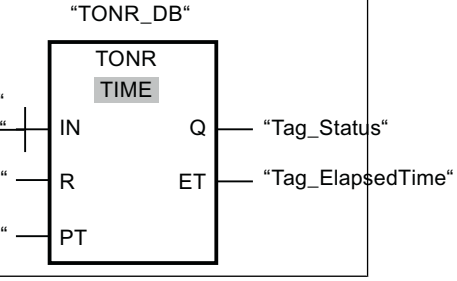

You can find additional information on this in the section "Programming PLC > Instructions".

Counters*

* Counters support the integer data types

Instruction	LAD	FBD
CTU: Count up	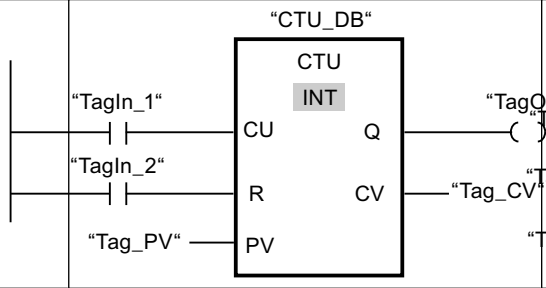	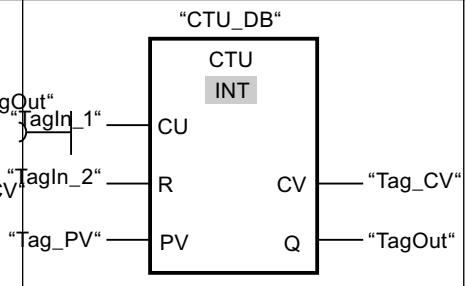
CTD: Count down	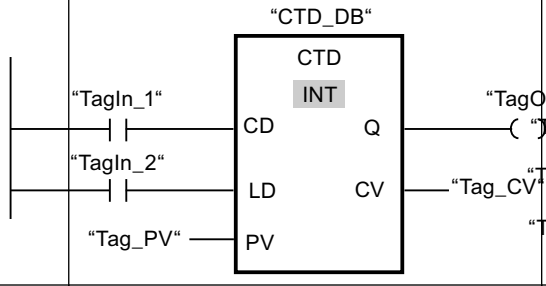	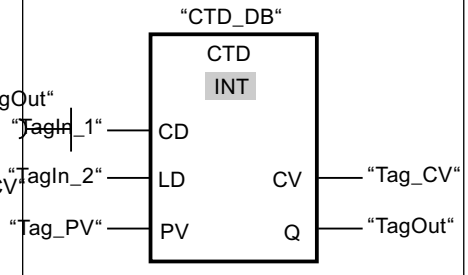
CTUD: Count up and down	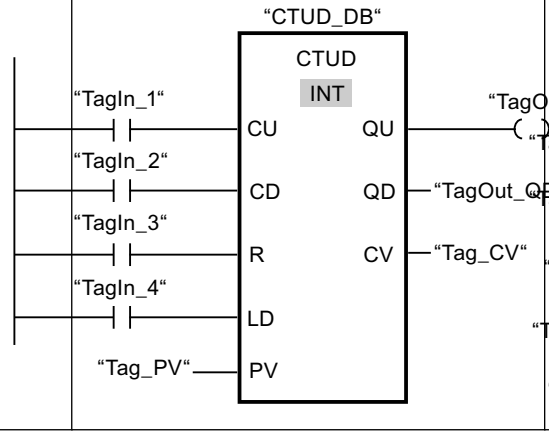	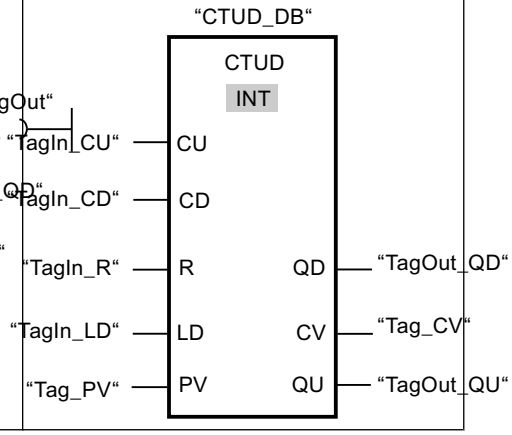

You can find additional information on this in the section "Programming PLC > Instructions".

Supported data types (RT Professional)

Supported data types

The following table shows data types supported in the "PLC Code View" object and their display formats:

Data type	Length	Format
Binary numbers		
BOOL*	1 bit	Bool
BYTE	8-bit	Hex

Data type	Length	Format
WORD	16-bit	Hex
DWORD	32-bit	Hex
LWORD	64-bit	Hex
Integers		
SINT	8-bit	Dec
USINT	8-bit	Dec
INT	16-bit	Dec
UINT	16-bit	Dec
DINT	32-bit	Dec
UDINT	32-bit	Dec
LINT	64-bit	Dec
ULINT	64-bit	Dec
Floating-point numbers		
REAL	32-bit	Floating point number
LREAL	64-bit	Floating point number
Timers		
S5TIME	16-bit	SIMATIC_Timer
TIME	32-bit	Time
LTIME	64-bit	LTime
Date and time		
DTL**	12 bytes	Date and Time
* Only with Boolean operands ** Complete DTL structures are not supported. Only single elements of DTL structures are supported.		

Note

You can find more information on the data types under "PLC Programming > Data Types".

Note

Operands and UDTs

Operands that are declared in the "#Temp" or "#InOut" area are generally not supported by the PLC code view. This applies both to elementary data types and to data types that are contained in UDTs. Data types of a UDT can be declared in the "#In" and "#Out" area and displayed in the PLC code view. The same limitations as for elementary data types apply to the data types of the UDT.

Note

The upstream network must not contain any tags from the Temp or InOut section of an FB.

The following data types may not be used for tags: STRING, WSTRING, CHAR, WCHAR, S5TIME

Note

The 64-bit PLC data types LINT, ULINT and LWORD are mapped to the HMI data type LREAL in the HMI channel. A loss of accuracy can be expected when the value exceeds the size 2^50.

See also

Data types for SIMATIC S7-1500 (Page 609)

Restrictions for the PLC code view (RT Professional)

Operands and UDTs

Operands that are declared in the "#Temp" or "#InOut" area are generally not supported by the PLC code view. This applies both to elementary data types and to data types that are contained in UDTs. Data types of a UDT can be declared in the "#In" and "#Out" area and displayed in the PLC code view. The same limitations as for elementary data types apply to the data types of the UDT.

Restrictions to data types

The use of the data types STRING, WSTRING, CHAR, WCHAR, S5TIME results in the network not being able to be displayed in the PLC code view.

Use of 64-bit data types

The use of 64-bit data types may result in a slight loss of accuracy as these data types are mapped in the HMI channel to the data type Double. It is therefore possible that integer data types are displayed with decimal places.

Addressing of tags

Please note the following restrictions when addressing tags:

- The "PLC code view" object only supports symbolic addressing of tags. If the operand is not addressed symbolically, the network with the operand cannot be displayed and an error message is generated.
- The use of array elements with a tag as index is supported, e.g. #myArray[MyTag]. Please use numerical values when addressing array elements, for example #myArray[6].
- Slice access enables you to address specific areas within declared variables. With Slice access, PLC code view only supports the access width "Bit" to Boolean tags.

Jump to the PLC code view

For the jump from a supervision alarm to the PLC code view, the instance name must conform to the following naming convention when using supported local operands in a function block: <FB-Name>_DB.

The jump to a function or an organization block is only possible if only global operands are used.

Configuring a jump from the alarm (RT Professional)

Introduction

You have the option of configuring a jump from the alarm view to the PLC code view or to the project in TIA Portal from runtime. Two functions are available to you:

- Jump to the PLC code view using the system function "ShowPLCCodeViewFromAlarm (Page 1701)".
- Jump from runtime to the configuration in TIA Portal using the system function "ShowBlockInTIAPortalFromAlarm (Page 1699)" .

In this example you configure the jump to the PLC code view from the corresponding alarm in the alarm view.

Requirement

- The "Alarm" screen is opened.
- The alarm view "ProDiag_Alarm" has been created.
- The PLC code view "PLC code view" is created in the "Program code" screen.

Configuring the button for the jump

1. Drag-and-drop a button from the "Tools" task card into the open "Alarm" screen.
2. Assign a name, for example, "Jump to PLC Code" under "Miscellaneous > Object > Name".
3. Switch to "Events > Click".
4. Click "Add function".

5. Select the function "ShowPLCCodeViewFromAlarm" from the selection list.
6. Specify the name of the screen in which the alarm view is configured under "Alarm screen name", "Alarm".
7. Specify the name of the alarm view under "Alarm view", "ProDiag_Alarm".
8. Enter "Program code" under "Screen name of the PLC code view".
9. Enter "PLC code view" under "Name of the PLC code view".

Checking the jump option

1. Select the alarm view "ProDiag Alarm".
2. Switch to the "Events" tab.
3. Click on the "Selection changed" event.
4. Create a user-defined C script at this event:

```
#include "GlobalDefinitions.h"
void OnSelectedIdChanged(char* screenName, char* objectName, long
id) {
#pragma code("KOPAPI.dll")
#include "kopapi.h"
#pragma code()
CMN_ERROR errorStruct;
BOOL bResult;
bResult = IsJumpableProDiagAlarm (screenName, objectName, id, &err
orStruct);
SetPropBOOL (screenName, "Jump to PLC Code", "Enabled", bResult);
```

Result

You have configured a button for jumping to the PLC code view. The button which enables the jump is only displayed if a ProDiag- relevant alarm appears in the alarm view.

See also

Jumping to the configuration (Page 2328)

12.3.4.8 Correcting display problems in the PLC code view (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

If an error occurs in visualizing the program code, an error message with the associated error number is output in the PLC code view. To correct the display error, first work through the recommended first measures.

If the error persists, contact Customer Support.

First measures for error correction

Observe the following general information about troubleshooting:

1. Make sure that the current configuration was loaded onto the HMI device.
2. If you display program code in the PLC code view using a Runtime API function, check the notation and the sequence of the parameters of the call.
3. Restart Runtime on your HMI device.

Notes on error messages

The table below shows the error messages and the notes for correcting some specific errors in displaying the program code (if any):

Error number	Error message	Instructions
3101, 3107, 3109, 3113, 3402, 3501, 3502, 3601, 3602, 4000 - 4107, 4110 - 4113, 4115 - 4121 4200 - 4203, 4300 - 4604	A general error occurred in determining the display data.	
3102 4114, 4122, 4123	The parameters of the call are not correct.	Check whether all parameters of the call were entered in the specified order. Make sure that the information is complete and correct.
3103	The PLC was not found or the name of the PLC is incorrect.	Check the specified PLC name.
3104	The address of the code block could not be determined.	
3105	The address of the instance data block could not be determined.	Make sure that the name of the instance data black and not the name of the function block was specified in the parameter.
3106	The block is know-how protected.	
3108	The version of the Engineering System is not compatible.	Make sure that the user program and the visualization were configured with the corresponding version of the TIA Portal. The PLC code view with WinCC V14 only supports the configuration with the Engineering System of the TIA Portal V14.
3112	The network contains an operand that is not supported.	
3117	The network language is not supported.	
3201	The operand was not found.	
3301	The called block was not found.	
3302	No block was found in which this UDT is used.	
3303	No matching entries were found, or the entries contain data types that are not supported.	

Error number	Error message	Instructions
3401	An error occurred in determining the data of a GRAPH block.	Make sure that the specified block is a GRAPH block.
3114, 3110, 3111 4103	The network contains an element that is not supported.	
4106	An error occurred during the calculation of the online status.	Disconnect the HMI device by unplugging the network cable. Wait a moment and then re-connect the network cable.
4108	The PLC name was not specified.	
4109	The DB name was not specified.	
4202	The ProDiag UDT was not specified.	

12.3.5 Example: Visualization of the ProDiag functionality (RT Professional, RT Advanced, Panels, Comfort Panels)

12.3.5.1 Procedures overview (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

ProDiag enables you to supervise faults in your plant and display information on the supervisions on your HMI device.

The following example shows you how to use objects for the supervision and diagnostics of plants in WinCC.

Example

In this example you configure the visualization for supervision of a conveyor belt operation.

You track the movement of the production units on the conveyor belt with a ProDiag supervision that you created beforehand.

You create your user program in STEP 7 and set up ProDiag supervisions for the operands. In WinCC you then configure the screens and objects required to display the program, the current program status and the output of error messages.

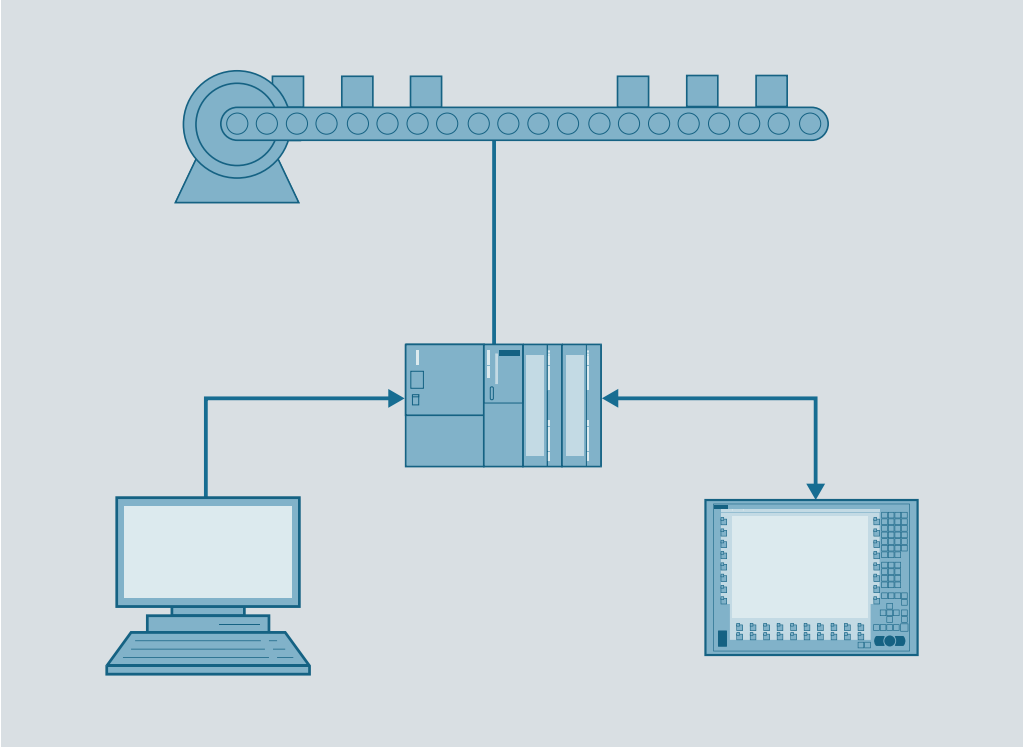

Configuration steps

The example is divided into several steps:

- Creating screens
  - Overview screen with the ProDiag screen objects
  - Screen for the alarm view
  - Screen for the PLC code view
- Inserting objects in the screens
  - ProDiag overview for display of the ProDiag supervisions
  - GRAPH overview for display of the GRAPH sequencer
  - Alarm view for display of the ProDiag alarms
  - PLC code view for display of the program code in the languages LAD, FBD and GRAPH
- Configuring objects
  Interconnecting the objects to enable targeted navigation to the cause of the error in runtime.

12.3.5.2 Requirements (RT Professional, RT Advanced, Panels, Comfort Panels)

Requirements

- An S7-1500 controller with firmware version 2.0 or higher has been created.
- A user program for controlling the conveyor belt has been created.
- The user program contains a GRAPH function block with the motion sequence of the conveyor belt.
- The user program contains a ProDiag function block with the configured operand monitoring.
  You can find more information about creating ProDiag supervisions in the section "Programming PLC > Supervising machinery and plants with ProDiag".
- The HMI device TP1200 Comfort has been created.
- A connection has been established between the controller and your HMI device.

Control program

A program for the sequence of the conveyor belt and a ProDiag function block with operand monitoring was previously created in STEP 7. The following table shows the blocks and tags which are used in this example of visualization with WinCC:

Name	Designation/type	Explanation
Conv_Seq [FB]	GRAPH function block	Contains the program sequence for moving the conveyor belt.
Conv_Seq_DB	GRAPH data block	Associated GRAPH instance data block.
OFF_SQ	Tag of the type Bool from the GRAPH data block	PLC tag which is used as the process tag for displaying the status of the GRAPH sequencer.
Conveyor [FB]	ProDiag function block	ProDiag function block contains the configured ProDiag monitoring for the operands which control the operation of the conveyor.
Conveyor_DB	ProDiag data block	Associated ProDiag instance data block.
State	Tag of the type SV_FB_State from the ProDiag data block	The status tag of the "SV_FB_State" PLC data type contains the group error bits of all the supervisions that are assigned to the ProDiag function data block.

12.3.5.3 Creating screens (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You will create three screens for displaying various objects to visualize the ProDiag supervisions in your plant.

Procedure

1. Open the project tree.
2. Click "Add new screen" in the Screens folder.
3. Open the shortcut menu of the screen and assign the name "Overview".
4. Create two additional screens named "Alarm" and "Program code".

Result

You have created an overview screen and two additional screens for visualization of the supervisions in Runtime.

12.3.5.4 Inserting diagnostic objects (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

To visualize the program code and the program status on your HMI device in runtime, insert the diagnostic objects into the created screens.

In the GRAPH overview, you can see the currently executed steps of the GRAPH sequencer that you have configured in Runtime.

ProDiag overview shows you whether a fault has occurred and a ProDiag supervision was triggered.

Requirements

- The screens "Program code", "Overview" and "Alarm" have been created.
- The "Tools" task card is open.

Procedure

1. Open the screen "Program code".
2. Drag-and-drop a "PLC code view" from the "Tools" task card into the screen.
3. Enter the object name "PLC code view" under "Properties > Properties > Miscellaneous > Name" in the Inspector window.
4. Open the screen "Overview".
5. Drag-and-drop the "ProDiag overview" object from the "Tools" task card into the screen. If you want to supervise multiple ProDiag blocks from your program, create a ProDiag overview for each block.
6. Enter a meaningful object name, such as "Overview_Conveyor", under "Properties > Properties > Miscellaneous > Name" in the Inspector window.
7. Drag-and-drop a "GRAPH overview" from the "Tools" task card into the screen.
8. Enter the name "GRAPH_Sequence" under "Properties > Properties > Miscellaneous > Name" in the Inspector window.

9. Open the screen "Alarm".
10. Drag-and-drop an alarm view from the "Tools" task card into the screen.
11. Enter the name "ProDiag_Alarm" under "Properties > Properties > Miscellaneous > Name" in the Inspector window.

Result

You have created objects for supervision and diagnostics in the respective screens.

- Objects for display of the current program status in the screen "Overview".
- PLC code view for display of program code in the languages LAD, FBD and GRAPH in the screen "Program code".
- Alarm view for display of ProDiag alarms in the screen "Alarm".

12.3.5.5 Configuring diagnostic objects (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In this example, you connect the objects for visualization of program code and supervisions to your control program. You have previously created a program in STEP 7 for transportation of the production units on the conveyor belt. You have also created a ProDiag function block in which you set up the supervisions for the operands.

Requirements

- The "Overview" screen has been created.
- A GRAPH overview and a ProDiag overview have been created in the screen.
- A ProDiag function block was created in the control program.
- A GRAPH instance data block "Conv_Seq_DB" was created in the control program.
- GRAPH instance data block "Conv_Seq_DB" contains the "OFF_SQ" tag which is visible in HMI and accessible from HMI.

Procedure

1. Select the GRAPH overview.
2. Click on the selection button under "Properties> General> Process > Tag" in the Inspector window.
   A dialog opens.
3. Click on the GRAPH instance data block "Conv_Seq_DB" in the "Program blocks" folder.

4. Select the PLC tag "OFF_SQ" of the GRAPH instance data block and confirm your selection.

Note

The process tag you are using for the GRAPH overview must be visible in HMI and accessible from HMI.

To identify the tags of the GRAPH data block as visible and accessible for HMI, open the GRAPH function block, select the block in the work area, and select "Edit > Internal parameters visible/accessible from HMI" in the menu bar. Then compile the program blocks.

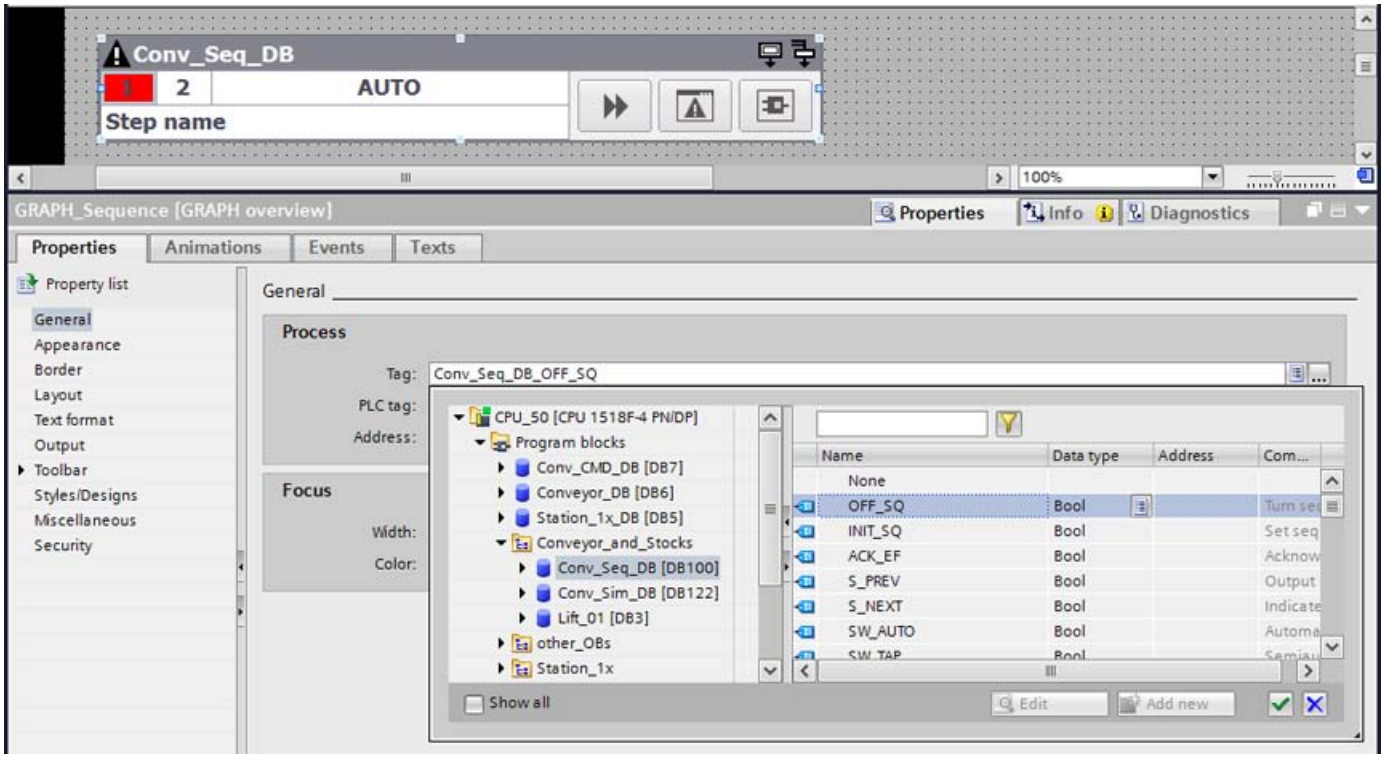

The GRAPH instance data block and the "GRAPH overview" object are connected.

5. Select the ProDiag overview "Overview_Conveyor".
6. Click on the selection button under "Properties> General> Process > Tag" in the Inspector window.
   A dialog opens.

7. Click on the data block "Coveyor_DB" in the "Program blocks" folder.
8. Select the status tag "State" of the "Conveyor_DB" data block.

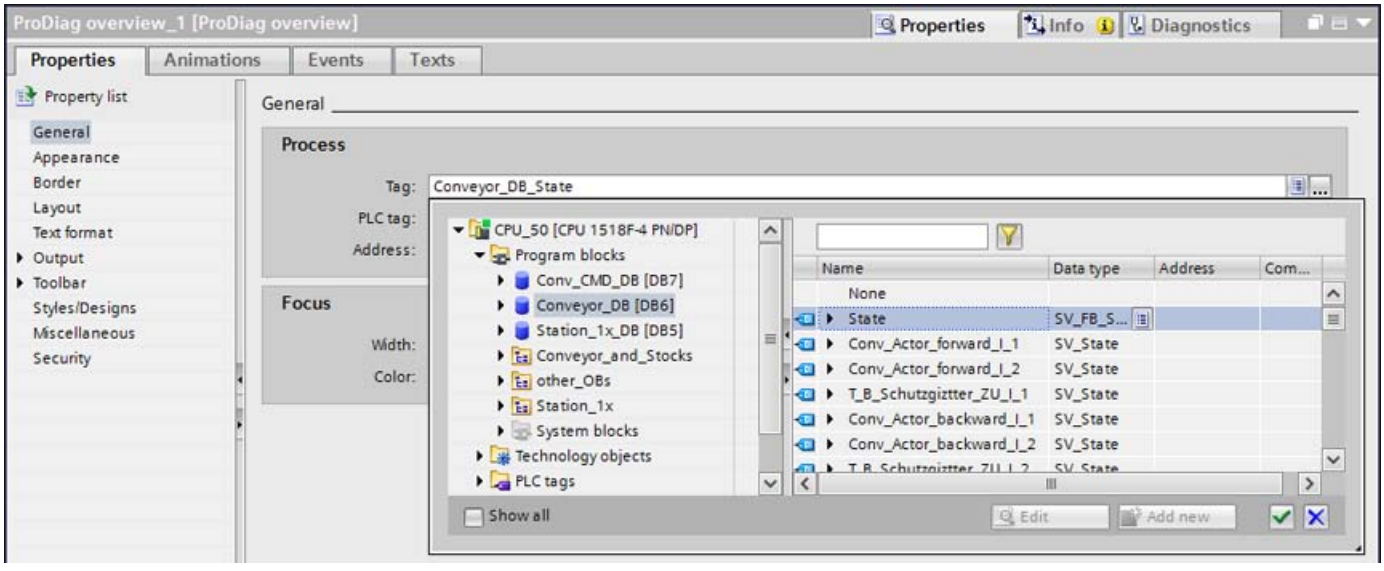

The ProDiag data block and the "ProDiag overview" object are connected.

Result

GRAPH overview and ProDiag overview are connected to the data blocks of the control program. In Runtime, the objects show the incoming monitoring errors and the current status of the GRAPH sequencer.

12.3.5.6 Configuring the alarm view for ProDiag (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In this example you configure the alarm view for display of the monitoring alarms and jump to the PLC code view with the program code.

Requirements

- The "Alarm" screen has been created.
- The alarm view "ProDiag_Alarm" has been created.
- The PLC code view "PLC code view" is created in the "Program code" screen.

Procedure

1. Open the "Alarm" screen and select the alarm view.
2. In the Inspector window, select "Properties > General".

3. Select the alarm classes of alarms that are displayed in the alarm view, for example, the system-internal alarm classes "Acknowledgement" and "No Acknowledgement".
   You have the option to also display alarms of alarm classes that you have defined previously in STEP 7.

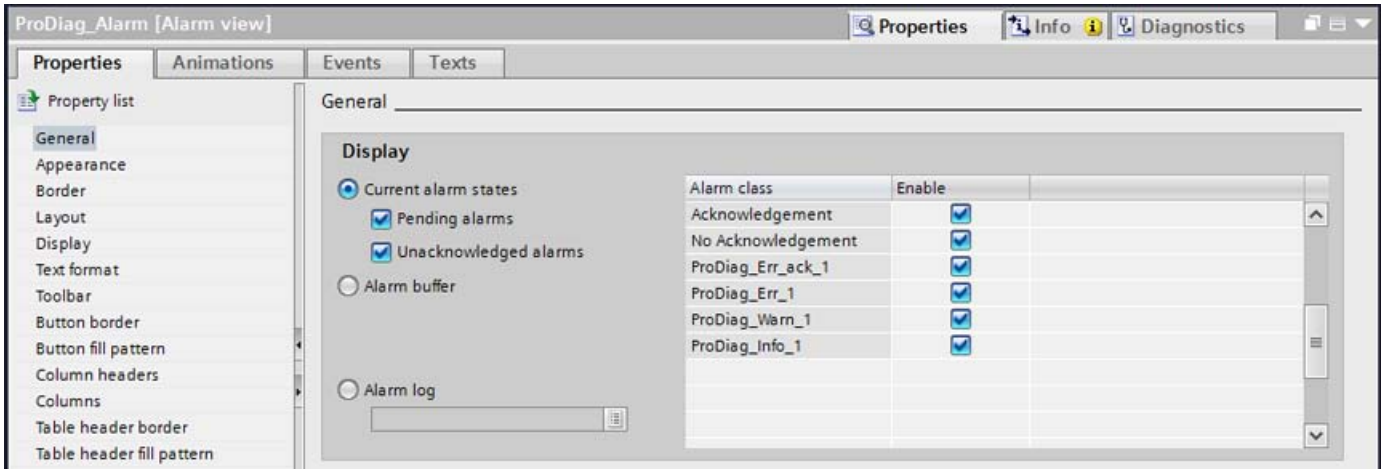

4. Specify a Boolean tag under "Properties > Display > Control tag for PLC code view" that will control the jump from the last active ProDiag alarm to the PLC code view.
   This tag is used to evaluate whether the jump from the selected alarm to the PLC code view is possible.

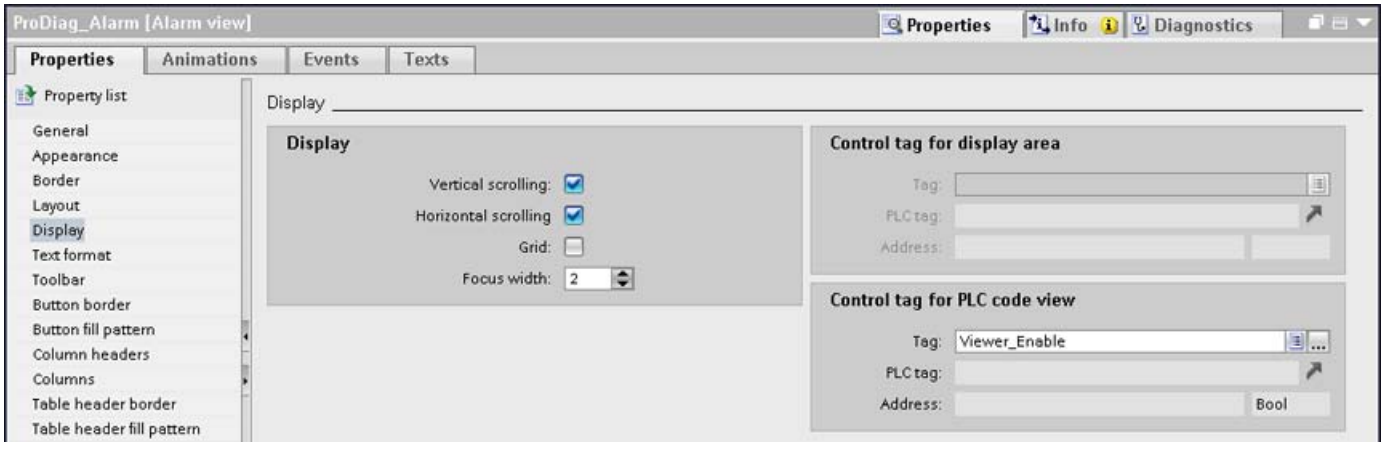

5. Drag-and-drop a button from the "Tools" task card into the "Alarm" screen.
   Enter the name "Program code" under "Properties > Properties > Miscellaneous > Name" in the Inspector window.
6. Select "Events > Click" in the Inspector window.
7. Select the "ActivatePLCCodeView" system function under "Add function".

8. Enter the screen name "Program code" and the object name "PLC code view" as parameters.
9. Under "Properties > Animations > Visibility > Tag", select the previously specified Boolean control tag for PLC code view.

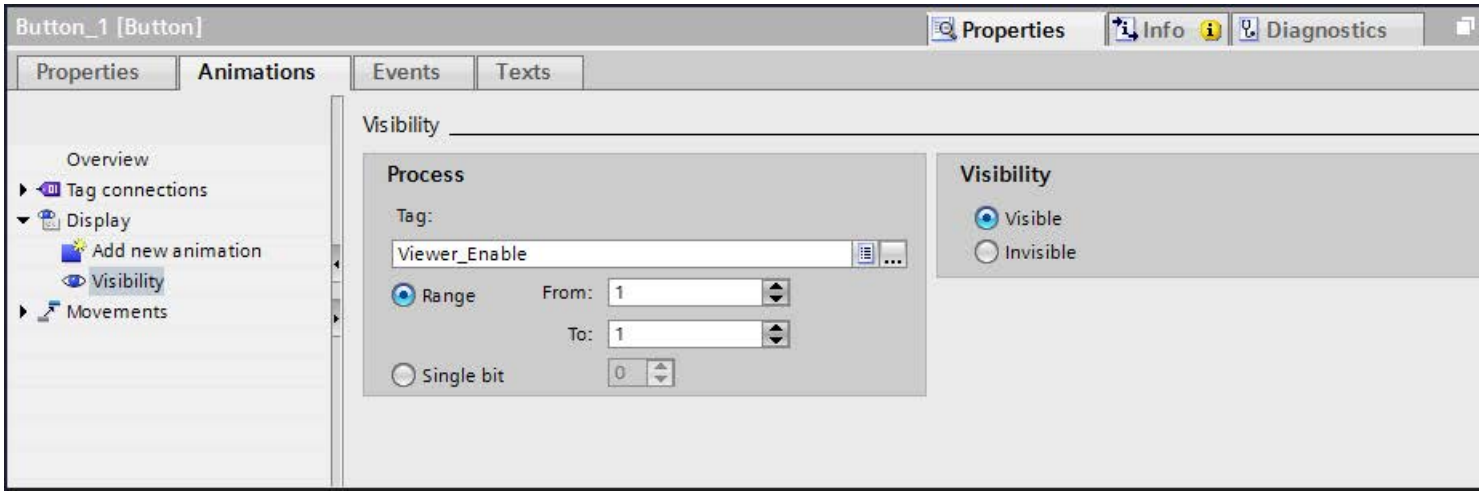

If the jump from the incoming ProDiag alarm to the PLC code view is possible, the "Program code" button is visible.

Result

You have activated the display of alarms of the defined alarm classes in the alarm view. In Runtime, the alarms of the corresponding alarm classes are displayed in the alarm view.

Using the configured button, you jump to Runtime in the PLC code view with the program instance that has triggered the ProDiag alarm.

12.3.5.7 Configuring the jump from the GRAPH overview (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In this step you configure the basic navigation between the GRAPH overview, the alarm view and the PLC code view.

Configuring the jump from the GRAPH overview

1. Select the GRAPH overview "GRAPH_Sequence".
2. Select "Events > Alarm view button click" in the Inspector window.
3. Select the "ActivateScreen" system function under "Add function".
4. Specify the name of the "Alarm" screen that contains the alarm view.
5. Select "Events > PLC code view button click" in the Inspector window.

6. Select the "ActivatePLCCodeView" system function under "Add function".
7. Enter the name of the "Program code" screen and of the PLC code view "PLC code view".

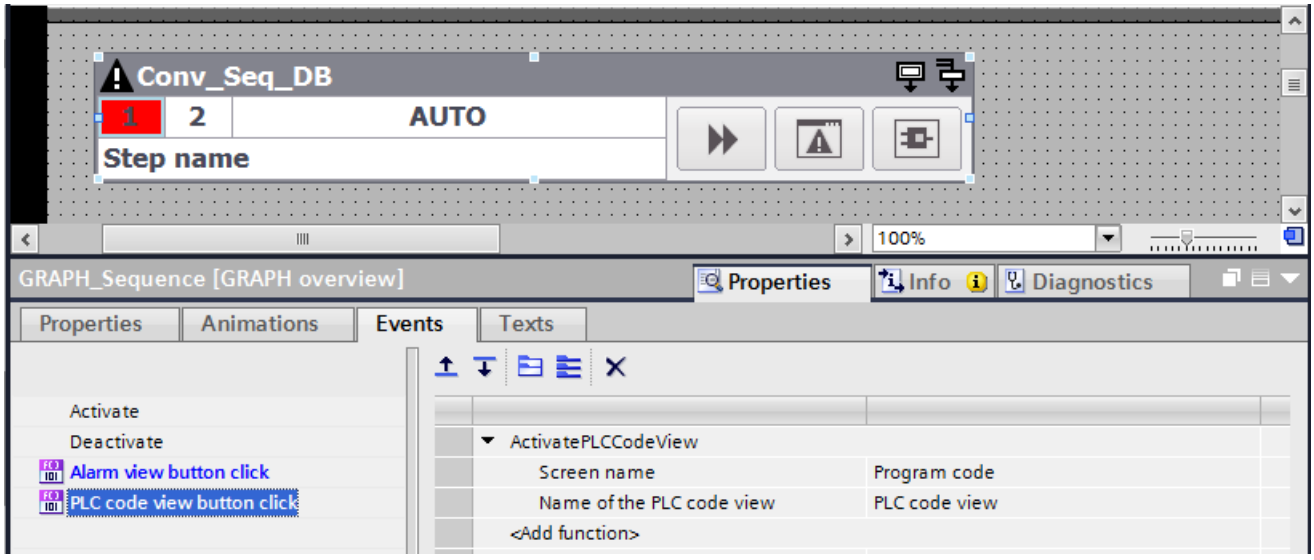

You have connected the GRAPH overview to the PLC code view. By clicking the "PLC code view" button, the screen with the display of the GRAPH sequencer is shown in Runtime.

Result

You monitor the current status of the sequencer using the GRAPH overview in Runtime. You have the option of viewing the GRAPH sequencer at any time in the PLC code view.

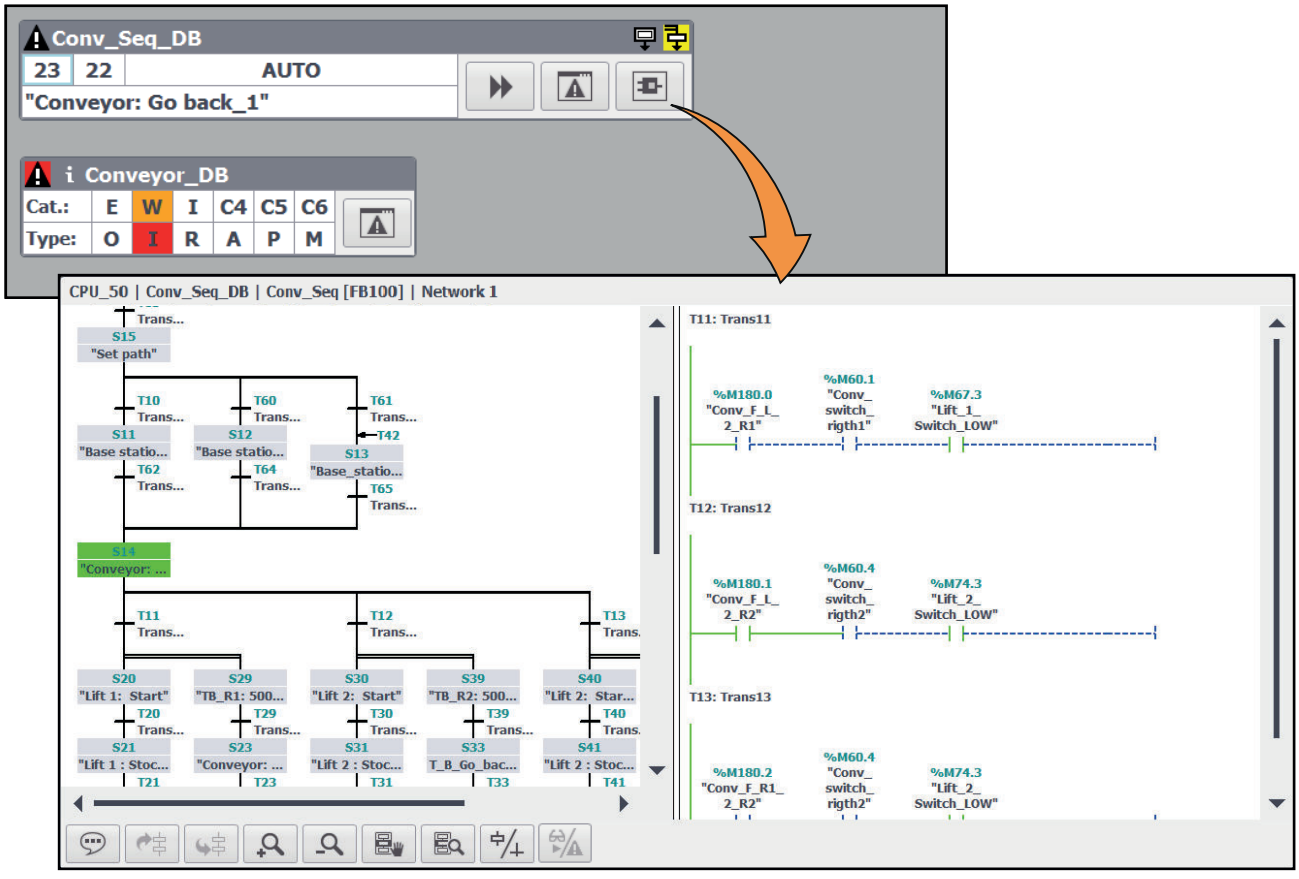

In case of a fault, you use the configured buttons to jump to the corresponding alarm in the alarm view or directly to the PLC code view with display of the step sequencer.

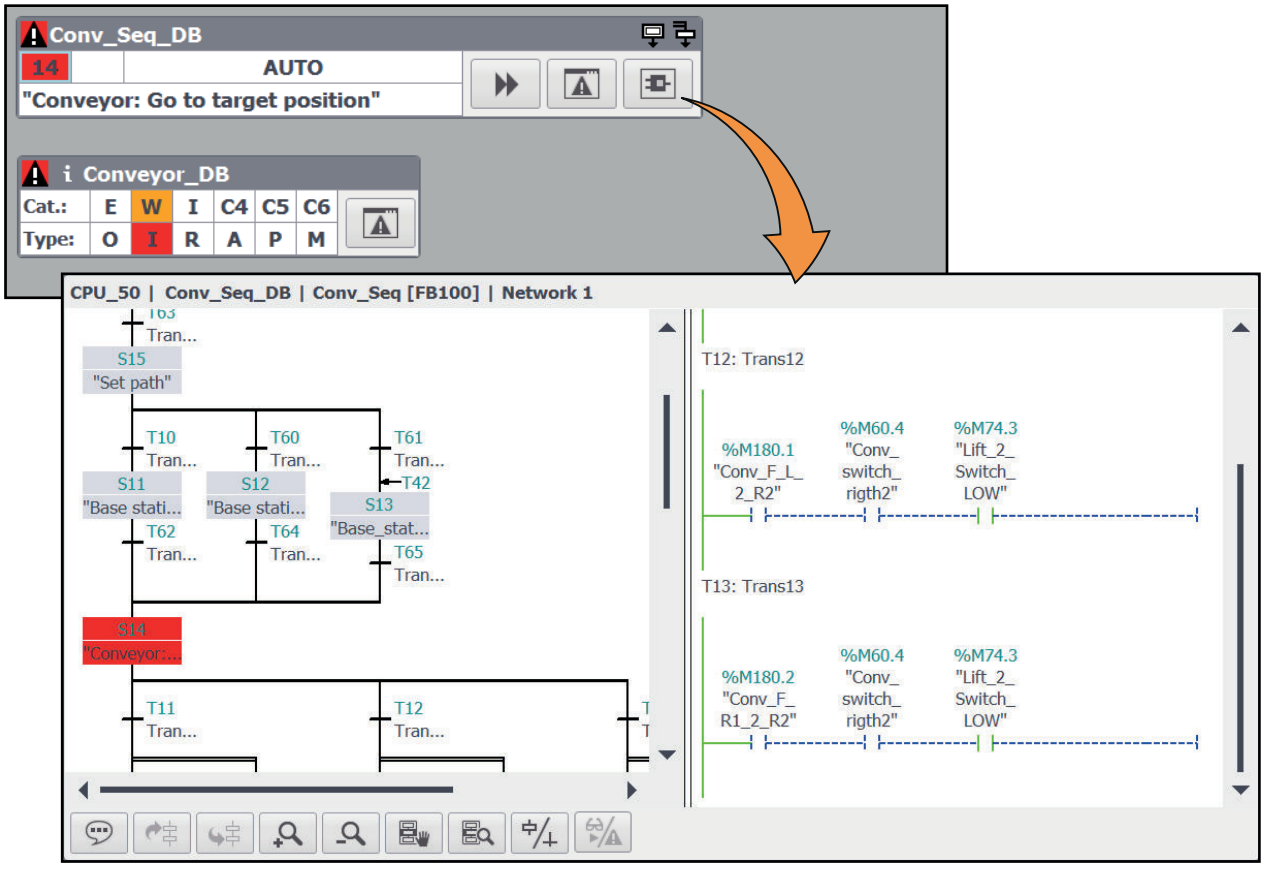

12.3.5.8 Configuring the jump from the ProDiag overview (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In this step you configure the basic navigation between the ProDiag overview, the alarm view and the PLC code view.

Configuring the jump from the ProDiag overview

1. Select the ProDiag overview "Overview_Conveyor".
2. Select "Events > Alarm view button click" in the Inspector window.
3. Select the "ActivateScreen" system function under "Add function".
4. Specify the name of the "Alarm" screen that contains the alarm view.

Result

You have connected the diagnostic objects to each other.

You monitor the status of the ProDiag supervisions using the ProDiag overview in Runtime. In case of an error, the corresponding supervision icon is visually highlighted.

By clicking the "Alarm view" button, the screen with the alarm view and the currently pending alarms is displayed. Using the configured buttons, you jump, for example, from a ProDiag alarm to the display of the network in the PLC code view.

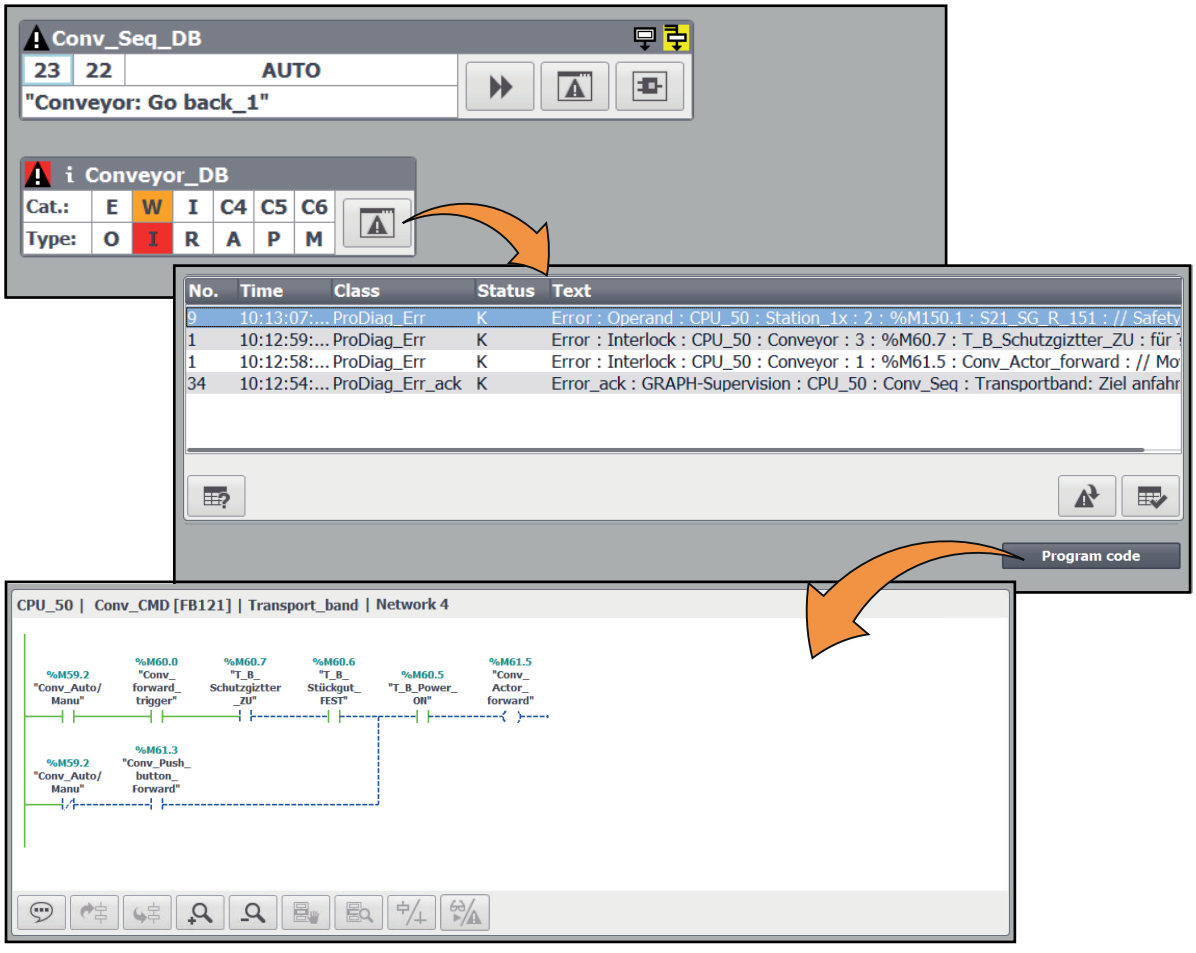

12.3.6 Initial value acquisition and criteria analysis (RT Professional, RT Advanced, Panels, Comfort Panels)

12.3.6.1 Overview of initial value acquisition and criteria analysis (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

In the TIA Portal you have the option of testing the execution of your user program on the HMI device. The data and values on the HMI device are continuously synchronized with the PLC and updated. You therefore see the current program status with the actual values of the signal states on the HMI device.

If an error occurs in your plant, you have the option of jumping to the program code from the corresponding error message and displaying the error location in the network in the “PLC code view". In the "Criteria analysis view" object, you see the operands with error for a selected GRAPH or ProDiag alarm at a glance.

The initial value acquisition and criteria analysis functions enable you to record the values at the time of the error and to quickly identify the operands with error in the program.

The actual value acquisition and criteria analysis functions are available for GRAPH function blocks, ProDiag function blocks and safety programs (F-blocks).

Requirement

- Initial value acquisition is available in TIA Portal for the following blocks:
  - Version V14 SP1 or higher for GRAPH function blocks Version 4.0 or higher.
  - Version V15 or higher for ProDiag function blocks Version 2.0 or higher.
- Maximum of 32 statuses can be recorded. The initial values for a network that contains more than 32 elements are not recorded.

Initial value acquisition

With the help of initial value acquisition you can acquire the values at the time of the error in the PLC, display them in the PLC code view and compare them with the actual values. With initial value acquisition you continuously record the signal states of Boolean operands and results of comparators in transitions and interlocks.

The signal states are recorded in a defined order from top left to bottom right:

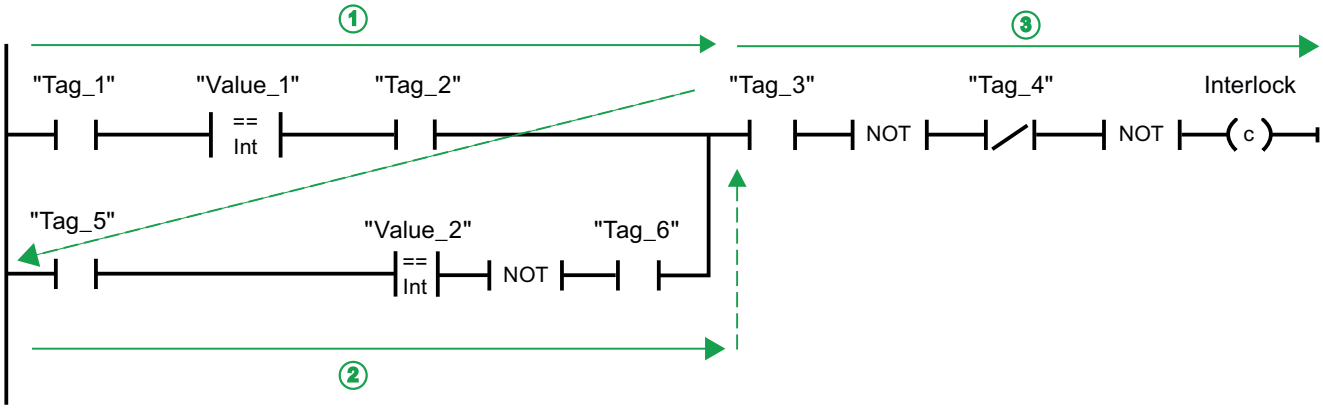

You activate initial value acquisition individually for each GRAPH block in the user program. A maximum of 32 signal states of Boolean operands can be recorded per interlock or per transition of a GRAPH step. Each individual signal state occupies one bit. The values are saved in a DWORD.

In the following example you can see the principle and order in which the initial values are recorded in the interlock:

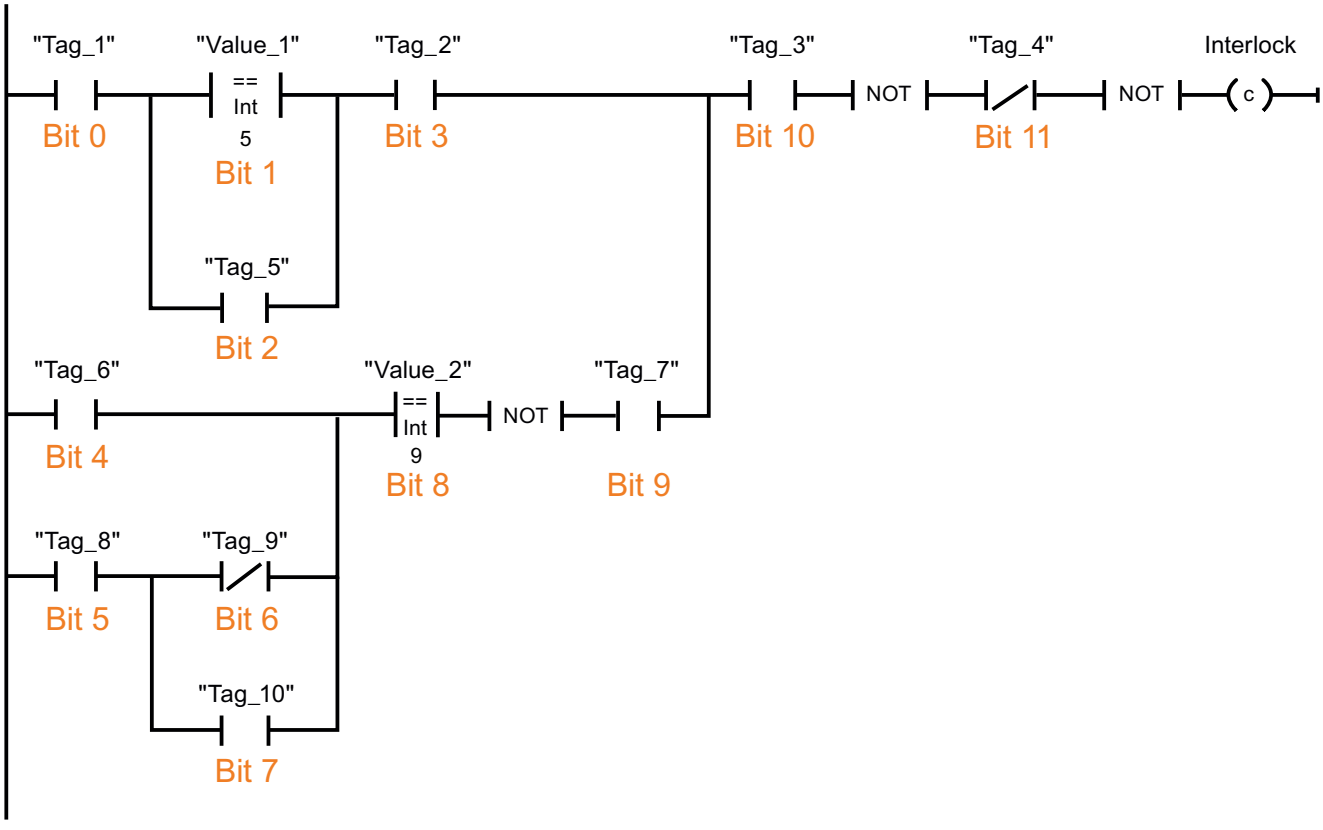

Criteria analysis

Initial value acquisition in the PLC enables the analysis of criteria and operands with error in the program. You see the evaluation of the criteria analysis on your HMI in the PLC code view. In addition, you can use the criteria analysis to have the operands with error displayed for a selected GRAPH or ProDiag alarm.

For the blocks for which you have activated initial value acquisition, after the jump the initial value view is displayed by default in the PLC code view. In addition, the operands with error and criteria are highlighted visually in the initial value view.

The names, absolute address and comments of the operands with error can be seen in the symbol table. You can switch between the view of the actual values and initial values.

In the event of an error in a comparator, both operands are marked as having errors. Only the recorded values are shown in the initial value view. To see the actual values of the tags, change to the actual value view.

Note

Observe the evaluation order during programming

As the operands are listed in the symbol table in accordance with the rule of initial value acquisition, in the event of an error the top left operand is output first in the symbol table in the criteria analysis. Observe this evaluation order when programming the network in order to see the most important operands first in the PLC code view.

Further information on the order of the listing can be found in the Example of order of the criteria (Page 1919).

See also

Example of order of the criteria analysis (Page 1919)

12.3.6.2 Supported instructions (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You see the initial values and the results of the criteria analysis in the "PLC code view" object.

For global supervisions, the initial values of all Boolean operands in the network are recorded. If the network contains multiple individual power rails that are not connected to each other, only the initial values of the respective power rail are recorded.

For local supervisions, only the initial values that are specified as conditions for the supervised parameter for the block call are recorded.

Supported instructions

The following instructions are supported in LAD and FBD for initial value acquisition:

Instructions	Display on the HMI device (HMI)
Bit logic operations	
Normally open contact	Initial values and criteria analysis
Normally closed contact	
Invert RLO	The instruction is supported but it is not relevant for initial values or the criteria analysis.
Assignment	
Negate assignment	
Reset output	Initial values
Set output	
Set/reset flip-flop	
Reset/set flip-flop	
Comparator operations	
Equal	Initial values and criteria analysis
Not equal	
Greater or equal	
Less or equal	
Greater than	
Less than	
Timers	
TP	Initial values
TON	Initial values and criteria analysis
TOF	Initial values
TONR	
Counters	
CTU	Initial values
CTD	
CTUD	

For bit logic operations, the status of the operand is recorded. For comparators, the result of the comparison is recorded.

For flip-flops, both inputs (R und S) are recorded if they are interconnected.

For timers and counters, the status of the operand at the output, and the inputs if they are interconnected, are recorded. (For example, for CTUD: CU, CD, R, LD)

Note

The FBD instruction EXCLUSIVE OR is not supported by the initial value acquisition and criteria analysis.

See also

Supported data types (Page 1896)

Supported data types (Page 1884)

12.3.6.3 Example of order of the criteria analysis (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

The evaluation of the criteria analysis in the PLC code view is performed in a defined order. In this example you see how initial value acquisition and the criteria analysis work on an HMI device.

Initial value acquisition

You record the signal state of an interlock in automatic and manual mode in the user program and have configured the following operands:

Name of symbol	Comment	Meaning
Tag_1	Auto mode	Automatic mode
Tag_2	Interlock_1	First interlock
Tag_3	Motor On	Switch on the motor
Tag_4	Interlock_2	Second interlock
Tag_5	Hand mode	Manual mode
Tag_6	Hand switch	Manual button

The signal states of the operands are recorded in the following order and written to the DWORD:

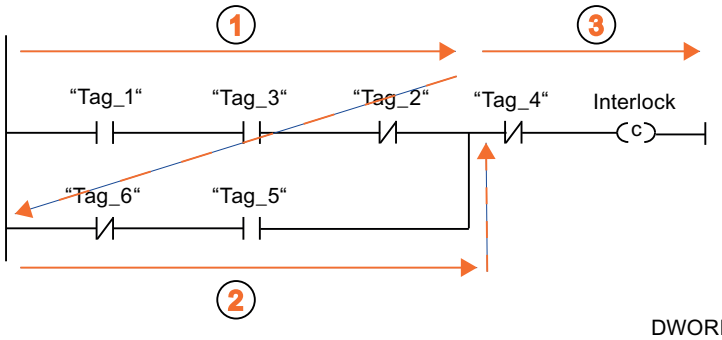

DWORD

Bit 31	Bit ...	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	Tag_4	Tag_5	Tag_6	Tag_2	Tag_3	Tag_1

Criteria analysis in the event of an error

If an error occurs in the interlock, use the configured button to jump to the PLC code view from the error message and see the error location in the program. As you have already activated initial value acquisition, the initial value view is automatically shown with the criteria analysis in the PLC code view.

The operands triggering the error are highlighted in color. You only see the names, absolute addresses and comments of the operands with error in the symbol table of the PLC code view in accordance with the order of the initial value acquisition.

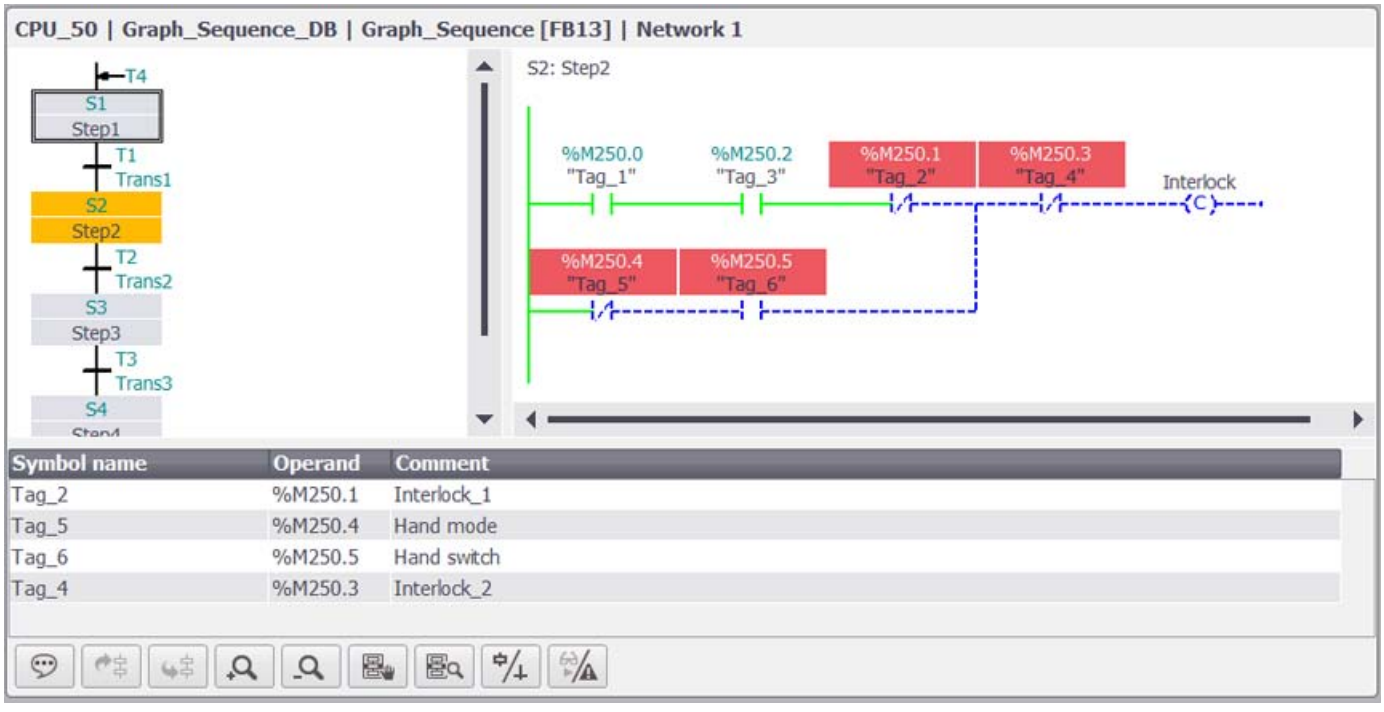

You have the option of switching to manual mode and enabling criteria analysis. This allows you to see the error-triggering network in the detail view even if the error situation has been corrected in the meantime.

Optimized programming of the interlock

So that you quickly see at a glance that the light barrier is responsible for the error, in this example it is beneficial to first program the operand Tag_4. The information about the open light barrier is then shown in the first line of the symbol table.

The following example shows the programming for the optimum view in the "PLC code view" object:

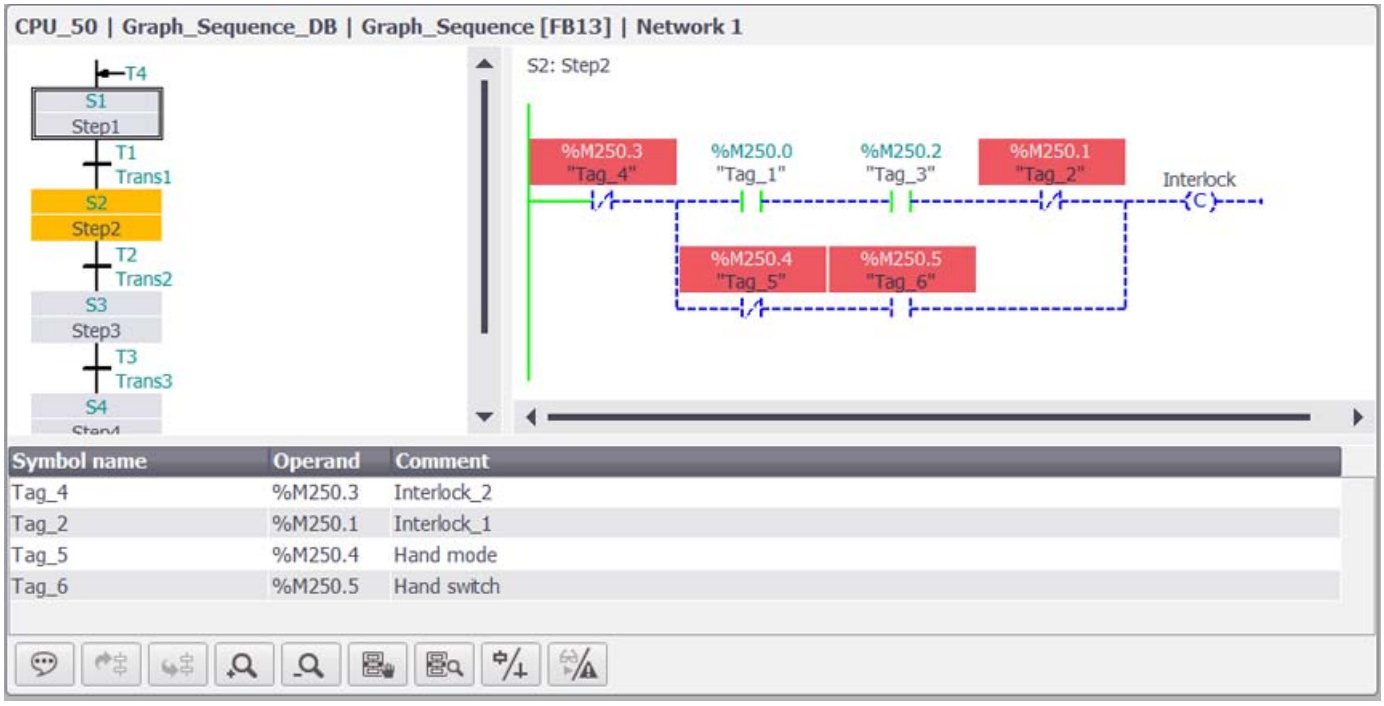

12.3.6.4 Configuring the criteria analysis view (RT Advanced, Panels, Comfort Panels)

Use

The "Criteria analysis view" object shows you the operands with error in the user program that have triggered a selected ProDiag or GRAPH alarm. To establish the connection to the corresponding alarms, you configure control tags at both objects.

If you select the incoming ProDiag or GRAPH alarm in the alarm view in Runtime, you see the operands that were determined in the criteria analysis in the criteria analysis view. You see the alarms and operands at a glance in Runtime if you configure the criteria analysis view and its linked alarm view in the same screen.

You have the option of connecting multiple alarm views to a criteria analysis view, and vice versa. You can configure the criteria analysis view in a pop-up screen or in a slide-in screen on KTP Mobile Panels, Comfort Panels and RT Advanced. As a result, you can connect multiple alarm views configured in various screens to one criteria analysis view, which you display as needed in a slide-in screen.

Note

The "Criteria analysis view" object is not available in the global screen.

Requirement

- The HMI device is connected to the controller
- A ProDiag program Version 2.0 or higher or a GRAPH program Version 4.0 or higher is installed on the controller
- Initial value acquisition is activated for the function blocks
- An alarm view is configured

Procedure

1. Move the criteria analysis view from the toolbox window using drag-and-drop.
2. Select "Properties > Properties > General" in the Inspector window.
3. Open the selection button under the "Tag" property.
4. Select the status tag of the corresponding alarm view under "Process > Tag".
5. Click on the alarm view.
6. Select "Properties > Properties > View" in the Inspector window.
7. Select the status tag of the criteria analysis view under "Control tag for criteria analysis".

Note

Data type and length of the status tags

The status tag must be of the WString data type and have a length of 50.

Result

The criteria analysis view is configured in the screen and connected to the alarm view. For a selected alarm you can see detailed information in Runtime about that operands that triggered this alarm.

12.3.6.5 Configuring the criteria analysis view (RT Professional)

Use

The "Criteria analysis view" object shows you the operands with error in the user program that have triggered a selected ProDiag or GRAPH alarm. To establish the connection to the corresponding alarms, you configure control tags at both objects.

If you select the incoming ProDiag or GRAPH alarm in the alarm view in Runtime, you see the operands that were determined by the criteria analysis in the criteria analysis view.

You configure the criteria analysis view and its linked alarm view in the same screen.

Requirement

- The HMI device is connected to the controller
- A ProDiag program Version 2.0 or higher or a GRAPH program Version 4.0 or higher is installed on the controller
- Initial value acquisition is activated for the function blocks
- An alarm view is configured

Procedure

1. Move the criteria analysis view from the toolbox window using drag-and-drop.
2. Select "Properties > Properties > General" in the Inspector window.
3. Open the selection button under the "Data source" property.
4. Select the configured alarm view.

Result

The criteria analysis view is configured in the screen and connected to the alarm view. For a selected alarm you can see detailed information in Runtime about that operands that triggered this alarm.

12.3.6.6 Outputting alarms with criteria (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

When initial value acquisition is activated, the values of the operand with error are recorded at the time of the error and missing criteria are analyzed and determined.

You have the option of adding additional information about the operands with error to the GRAPH and ProDiag alarms and outputting them on your HMI device. If an error occurs in the program flow in runtime, the error message also indicates the first operand with error in the faulty network. You see detailed information on all operands with error of the error message in the "Criteria analysis view" object.

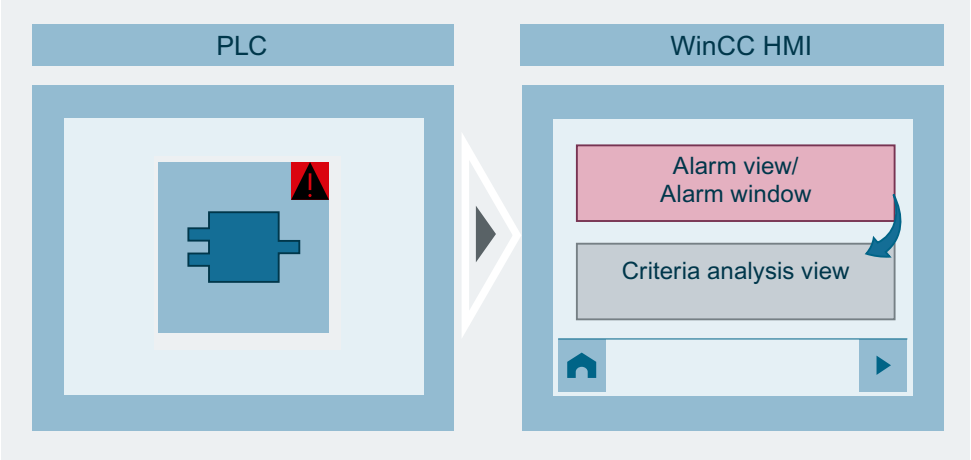

The following information can be added for an alarm:

- Symbol name: The symbolic name of the first operand that triggered the selected error message
- Absolute address: The address of the first operand with error
- Value of the operand: The value of the first operand with error at the time of the error
- Comment: Multilingual comments that were configured in the user program

The additional information is displayed in the message after the alarm text, separated by a semicolon and a space in each case.

Alternatively, you can specify that the additional information is displayed in the Info text.

In WinCC Runtime Professional you also have the option of expanding the additional alarm texts of the supervision alarms. For this, select the appropriate additional alarm text under "Runtime settings > Alarms > Criteria analysis > Expand text".

Note

The order of the additional information that is added to the alarm is predefined and cannot be changed.

Note

To completely display the alarms from the controller on the HMI device, the "Automatic update" option must be selected under "Runtime Settings > Alarms > Controller alarms" for the relevant connection. You can find additional information on complete alarms under "Sending a complete alarm from the controller to the HMI device (Page 948)".

Criteria analysis in the alarm system

You visualize the alarms for the criteria analysis in the following steps:

- You activate initial value acquisition in the properties of the ProDiag or GRAPH function block of the user program
- You activate the options for expanding the alarm texts in the runtime settings of the HMI device
- You configure an alarm view and a criteria analysis view and interconnect the two objects

Expanding alarms

1. Open the "Runtime settings" editor of the HMI device.
2. Select "Runtime settings > Alarms > Criteria analysis".
3. Select which alarms you want to expand under "Criteria analysis > Expand text".
4. Select the additional information to be added to the alarm text in the alarm, e.g. symbol name and address of the first operand with error.

Result

If an error occurs, you see not only the alarm text in the alarm view but also the operands that triggered the error message. You select the relevant alarm and see the operands with error determined by the criteria analysis in the criteria analysis view.

Communicating with controllers 13

13.1 WinCC Engineering V15.1 - Communication

Communicating with controllers

You can find a detailed description of communicating with controllers in the "WinCC Engineering V15.1 – Communication" manual.

Using global functions 14

14.1 Working with the HMI Device Wizard

14.1.1 HMI device wizard basics (Basic Panels, Panels, Comfort Panels)

Introduction

The HMI device wizard will automatically start when you create a new HMI device in your project.

HMI device wizard

The HMI device wizard will guide you through each dialog step by step and help you set up a device. You use the HMI device wizard to specify the basic settings for your HMI device, such as screen layout and the connection to your PLC.

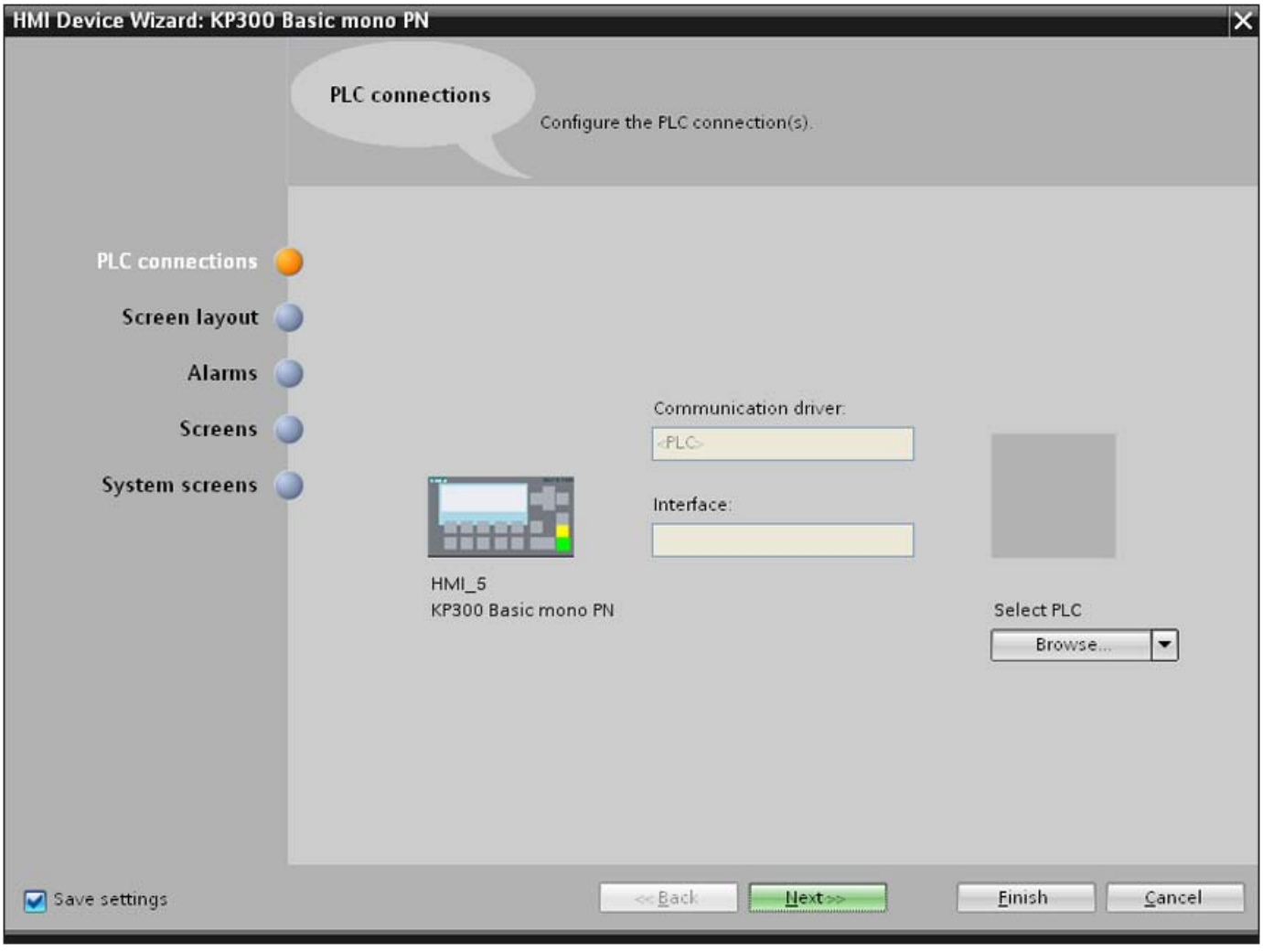

Note

When you create a device with a color display using the HMI device wizard, the graphics of the navigation buttons may be displayed in black and white. This error only occurs, however, if the new device is created with the same name as a device with a monochrome display which has been deleted in the meantime.

You can avoid this error by always deleting the associated graphics in the Graphics collection whenever you delete a device from the project.

Note

Multiple languages in the HMI device wizard

If you create multilingual screen navigation with the HMI device wizard, WinCC uses the user interface language for user-defined labels in all project languages.

If the character set of the user interface language is not supported in a project language, these names will not appear in this project language, for example, with a Chinese interface language and English project language.

Note

Interface type of the connection

If possible, the HMI device wizard makes a connection between the HMI device and controller with the default interface type of the HMI device.

After configuration with the HMI device wizard, check the connection in the "Devices & Networks" editor.

See also

HMI device wizard basics (Page 1932)

General information about cross references (Page 2003)

Languages in WinCC (Page 2012)

14.1.2 HMI device wizard basics (RT Professional, RT Advanced)

Introduction

The HMI device wizard will guide you through each dialog step by step and help you set up a device. You use the HMI device wizard to specify the basic settings for your HMI device, such as screen layout and the connection to your PLC.

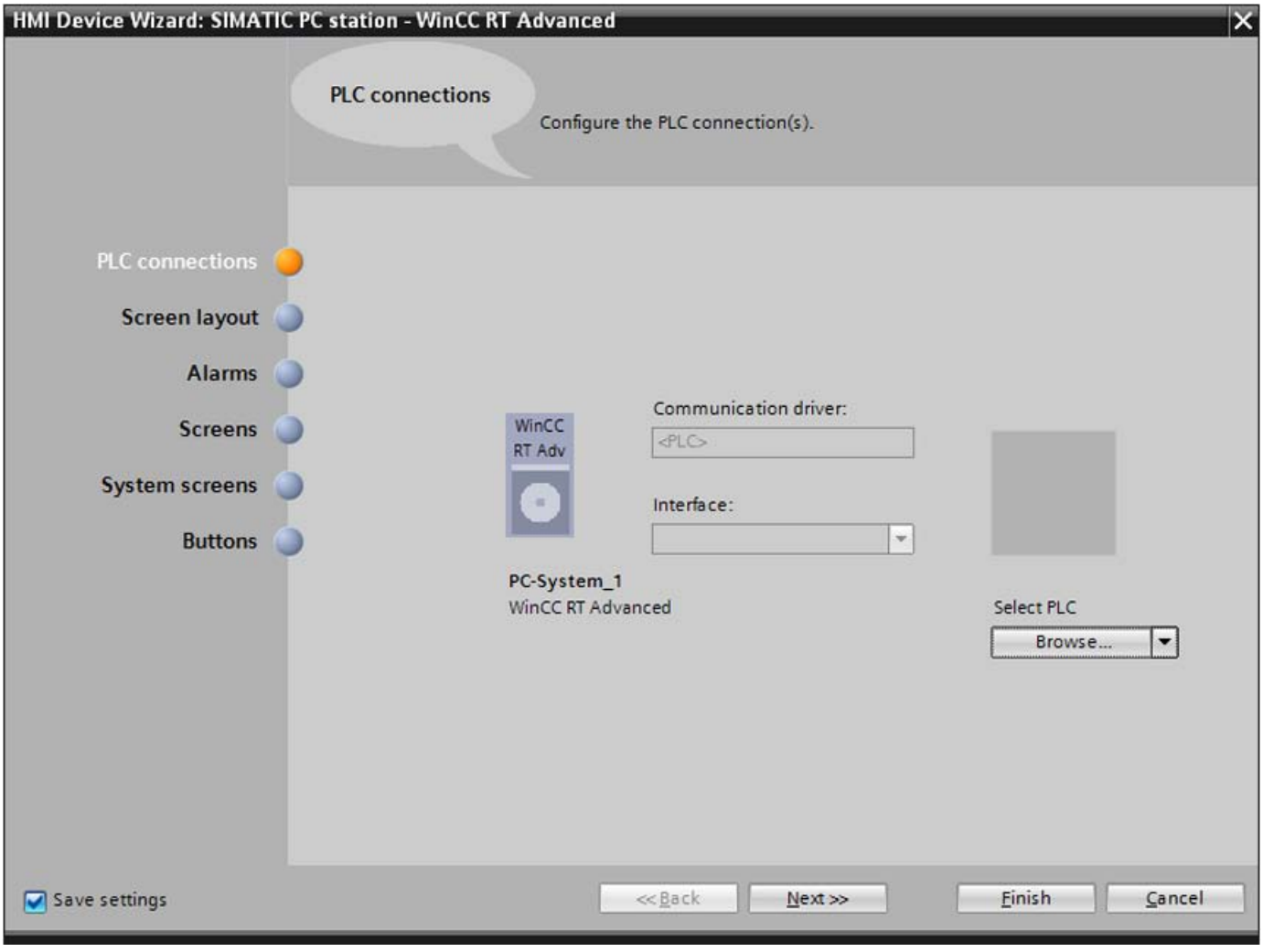

Opening the HMI device wizard

1. Add a new device to your project, for example, RT Advanced.
2. Click on the device in the project tree.
3. Select "Start the HMI device wizard" in the shortcut menu. The "HMI device wizard" opens.

Note

If you make changes after adding the device, such as adding a new screen, the HMI device wizard will not open again.

Note

When you create a device with a color display using the HMI device wizard, the graphics of the navigation buttons may be displayed in black and white. This error only occurs, however, if the new device is created with the same name as a device with a monochrome display which has been deleted in the meantime.

You can avoid this error by always deleting the associated graphics in the Graphics collection whenever you delete a device from the project.

Note

Device dependency - quotation marks in project names

WinCC Professional projects whose names are in quotation marks have not been compiled.

When you create an HMI device for WinCC RT Professional, do not use quotation marks within the project name.

Note

Multiple languages in the HMI device wizard

If you create multilingual screen navigation with the HMI device wizard, WinCC uses the user interface language for user-defined labels in all project languages.

If the character set of the user interface language is not supported in a project language, these names will not appear in this project language, for example, with a Chinese interface language and English project language.

See also

HMI device wizard basics (Page 1929)

Load files using the "HMI files" editor (Page 2107)

14.2 Working with libraries

14.2.1 Basics on libraries

Introduction

Store all objects you need frequently in the libraries. An object that is stored in the library only has to be configured once. It can then be used repeatedly as often as required. Library objects extend the number of available screen objects and increase the effectiveness during configuration through the multiple usage of ready-to-use objects.

Your WinCC software package is supplied with comprehensive libraries that contain, for example, "Motor" or "Valve" objects. You may, however, define your own library objects.

You manage libraries in the "Libraries" task card or library view. The following libraries are available:

- Project library
- Global libraries

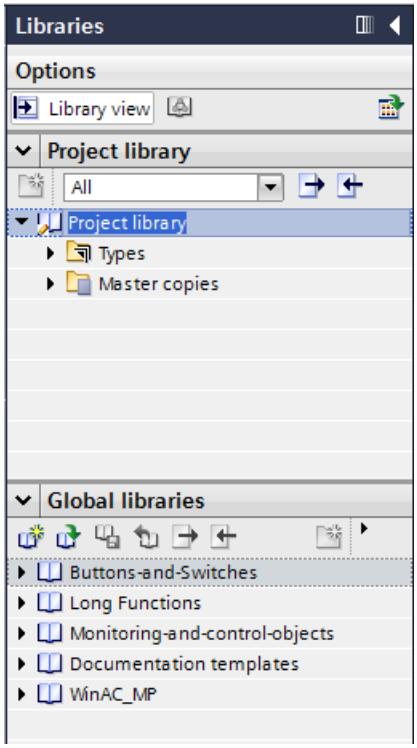

Note

There is a symbol library in the "Tools" task card in the "Graphics" palette.

Project library

There is one library for each project. Objects of the project library are stored alongside with the project data and are available only for the project in which the library was created. If the project is moved to another PC, any project library created in it is also moved.

To use the library object of the project library in other objects, move or copy the object into a global library.

Global libraries

A global library is saved independently of the project data in its own file with the extension ".aIxx", whereby "xx" stands for the current WinCC version number.

A project can access several global libraries. A global library may be used concurrently in several projects.

When a library object is changed by a project, this library will be changed in all projects in which these libraries are open.

Library objects

A library can contain all WinCC objects. Examples:

- Complete HMI device
- Screens
- Display and control objects including tags and functions
- Graphics
- Tags
- Alarms
- Text and graphics lists
- Faceplates
- User data types
- Styles and style sheets

See also

Screen basics (Page 39)

Displaying library objects (Page 1952)

Storing an object in a library (Page 1956)

Inserting a library object (Page 1957)

Creating a global library (Page 1944)

Saving a global library (Page 1945)

Opening a global library (Page 1946)

Master copies and types (Page 1938)

14.2.2 Overview of the library view

Function of the library view

The library view combines the functionality of the "Libraries" task card and the overview window. In the library view, the elements of a library are displayed in various views. In the details view, for example, you see additional properties of the individual elements. You can also edit and version the types in the library view.

Layout of the library view

The following figure shows the components of the library view:

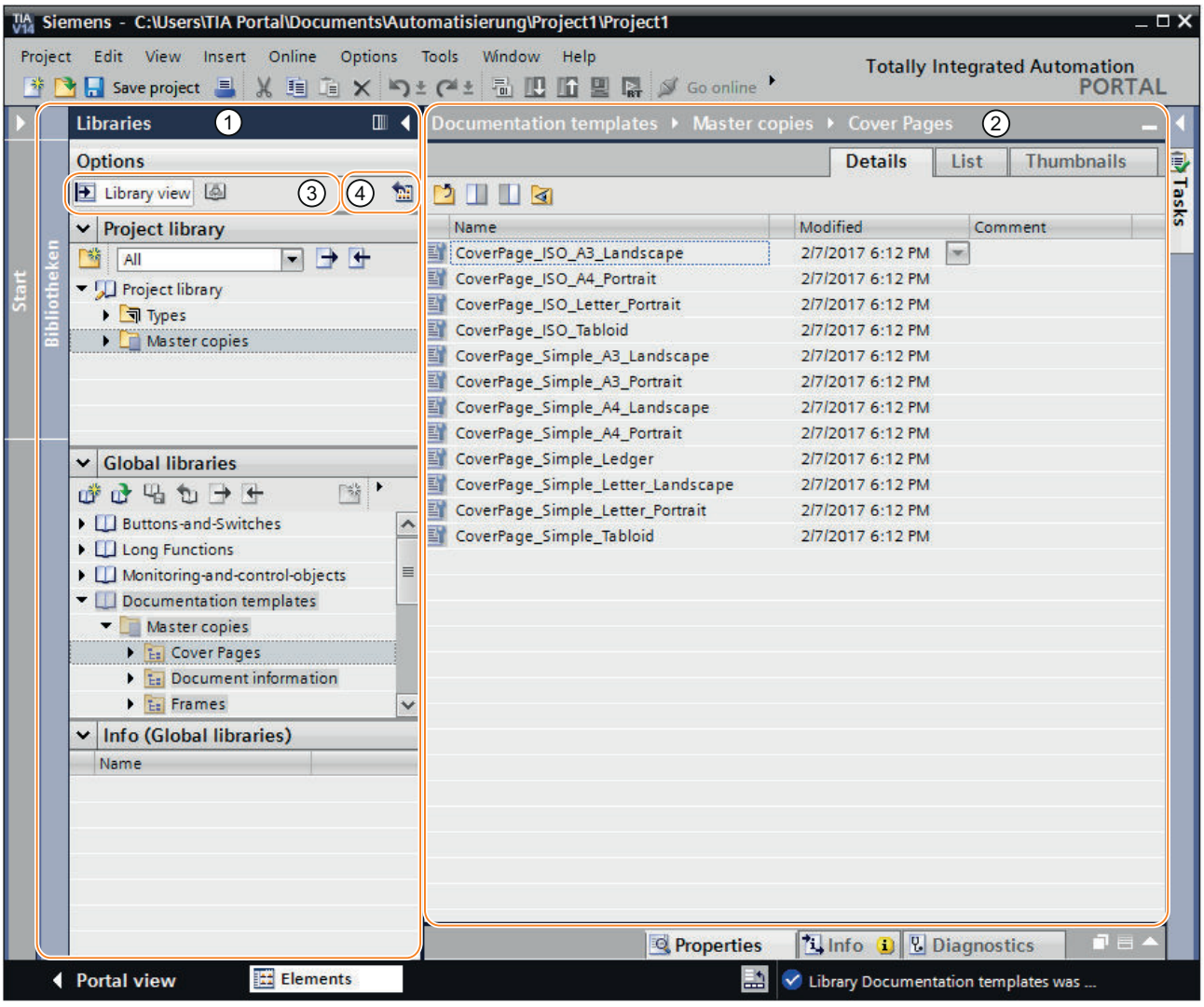

1 Library tree

2 Library overview

3 "Library view" button

4 "Open or close library overview" button

Library tree

The library tree is similar to the "Libraries" task card, apart from a few minor differences. In contrast to the task card, there is no "Elements" palette, because the elements are displayed in the library overview. In addition, you can close the library view in the library tree, or open and close the library overview.

Library overview

The library overview corresponds to the overview window and displays the elements of the currently selected object in the library tree. You can display the elements in three different views. You can also perform the following actions in the library overview, for example:

- Copying elements
- Moving elements
- Versioning types
- Editing faceplates and HMI user data types
- Editing type instances

14.2.3 Master copies and types

Introduction

Both the "Project library" and the "Global library" contain the two folders "Master copies" and "Types". You can create or use the library objects either as a master copy or a type.

Master copies

Use master copies to create independent copies of the library object.

Types

Create instances of objects of the "Types" folder and use these in your project. The instances are bound to their respective type.

Administration of the library objects

You can copy and move library objects to another library. You can only copy master copies to the "Master copies" folder or any sub-folder of "Master copies". You can also only insert types in the "Types" folder or any sub-folder of "Types".

See also

Basics on libraries (Page 1934)

State of type versions (Page 1958)

14.2.4 Libraries in WinCC

Introduction

The WinCC software package includes extensive libraries. Sorted by topic into folders, they contain preassembled graphic objects which you can use in screens for operation and monitoring of your plant.

Global library "Buttons and Switches"

The libraries "Buttons and Switches" offer a wide selection of buttons and switches.

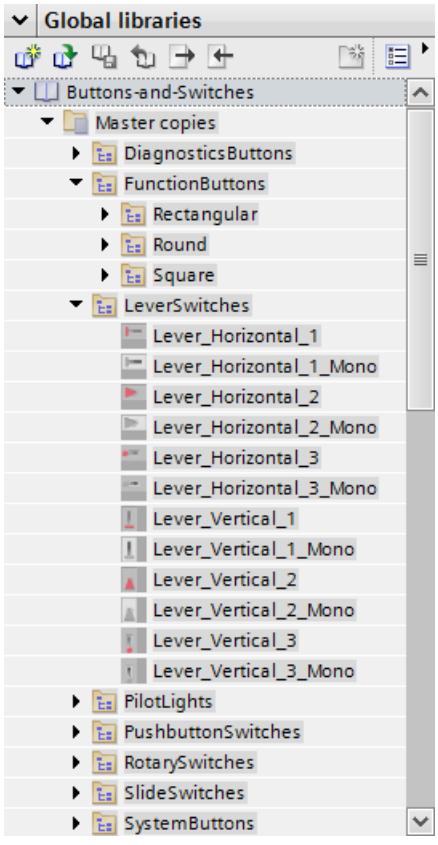

The folders divide switches and buttons into categories. The "DiagnosticsButtons" folder contains the object "System diagnostics indicator", for example. You use the "System diagnostics indicator" object for system diagnostics in your plant.

Note

You can only use the objects in the "DiagnosticsButtons" folder on Comfort Panels.

You cannot use objects which have "Switch" in the object name or in the associated folder name in Runtime Professional.

Global library "Monitoring and Control objects"

The "Monitoring and Control objects" library offers more complex display and operating objects in several designs and corresponding control lights, buttons and switches.

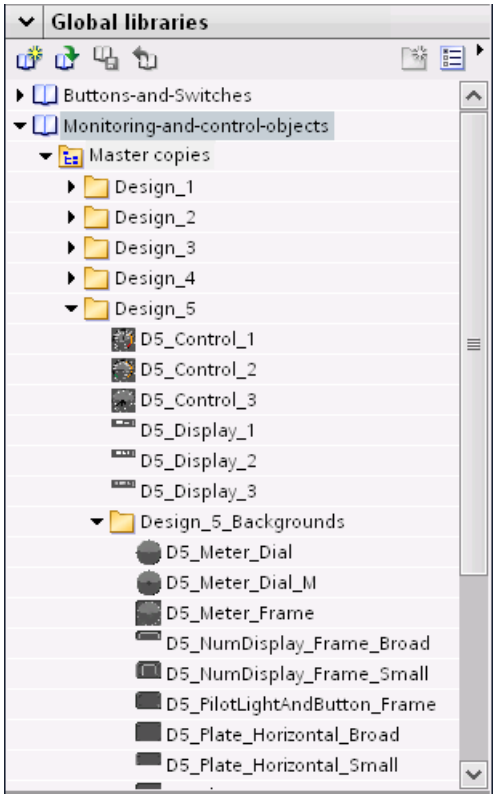

In addition, graphics views for the designs are stored in the "Design_Backgrounds" folder; they can be used as object backgrounds for the custom extension of the scope of the library.

Note

You cannot use objects which have "Switch" in the object name in Runtime Professional. The same applies for the object "D5_Display_3" with the date/time field it contains.

14.2.5 Managing libraries

14.2.5.1 Overview of the library management

Function of the library management

Master copies and types with dependencies to other library elements are subject to some functional restrictions. They can, for example, not be deleted as long as dependencies still exist. This prevents other library elements from becoming useless. The library management is used to identify the dependencies and to create an overview of the work progress.

The library management offers the following functions:

- Display of the correlations of types and master copies
  If a type is referenced in other types or master copies, the correlations are displayed in the library management. You will also be able to see which library elements reference a type or a master copy.
- Display of points of use of types in the project
- Display all types that include a version with the "In test" or "In progress" status

Layout of the library management

The figure shows the library management, consisting of the following components:

- Toolbar of the library management
- "Types" area
- "Use" area

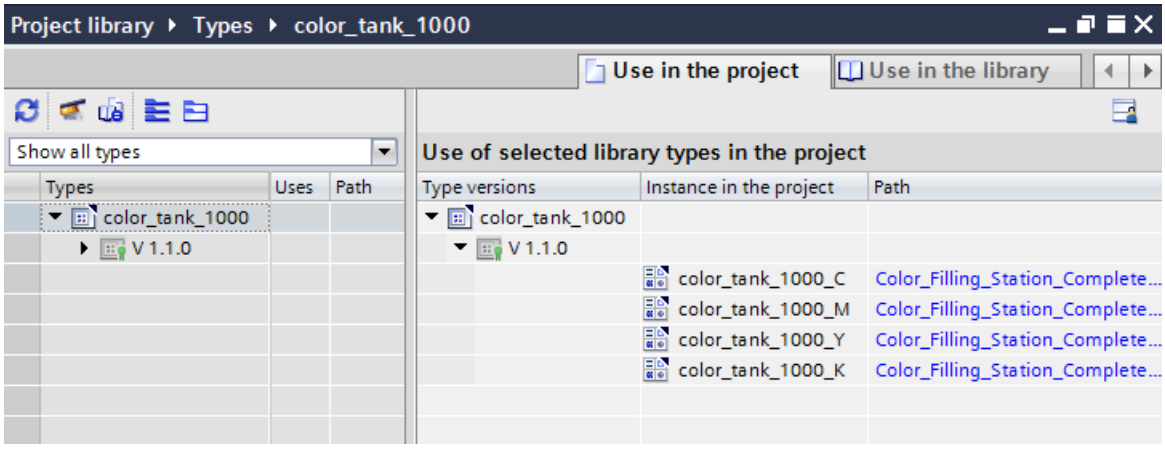

Toolbar of the library management

You can perform the following tasks in the toolbar of the library management:

- Update view
  If the project was changed, you can update the view of the library management.
- Clean up library
  You can clean up the project library and global libraries. By cleaning up a library, you delete all types and type versions that are not linked to an instance in the project.
- Harmonize project
  By harmonizing a project, you adapt the names and the path structures of type uses in the project to the corresponding names and path structures of the types within a library.

"Types" area

The "Types" area displays the contents of the folder that you have selected in the library view. You can open or close all types by using the buttons in the toolbar of the "Types" area. You can also use the "Filter" drop-down list to filter the view and display all types with the "In test" or "In progress" status only. For each type, the types that it references are displayed.

"Uses" area

The "Uses" area gives you an overview of the points of use of the selected types and master copies. The "Uses" area is divided into two tabs:

- "Uses in the project" tab
  The "Uses in the project" tab is used to show the instances of type versions and their respective point of use in the project. When you select an instance, you can show the cross references of the instance in the project in the Inspector window.
- "Uses in the library" tab
  The "Uses in the library" tab is used to show all points within the library at which a type or a master copy is used.

14.2.5.2 Opening library management

Procedure

To open the library management, follow these steps:

1. Open the library view.
2. Select a type or any folder that contains types.
3. Select the "Library management" command from the shortcut menu.

Result

The library management opens and the types are displayed with their versions.

14.2.5.3 Filtering types in the library management

Introduction

A filter function in the library management enables you to limit the displayed types. The following filters are available:

- Display all types that include a version with "in progress" status
- Display all types that have no instances in the project
- Display all types that have more than one version
- Display all released types

Requirement

At least one type has been created.

Filtering according to all types with pending changes

1. Select the "Types" folder in the project library.
2. Select the "Library management" command in the shortcut menu of the "Types" folder.
   The library management opens.
3. Select "Types with pending changes" from the "Filter" drop-down list.
   The "Types" area only displays types that have the "in progress" status.

Filtering for all types that have no instances in the project

1. Select the "Types" folder in the project library.
2. Select the "Library management" command in the shortcut menu of the "Types" folder.
   The library management opens.
3. Select "Types not used in the project" from the "Filter" drop-down list.
   The "Types" area only displays types that have no instances in the project.

Filtering for types with multiple versions

1. Select the "Types" folder in the project library.
2. Select the "Library management" command in the shortcut menu of the "Types" folder.
   The library management opens.
3. Select "Types with multiple versions" from the "Filter" drop-down list.
   The "Types" area only displays types that have more than one version.

Filtering for all released types

1. Select the "Types" folder in the project library.
2. Select the "Library management" command in the shortcut menu of the "Types" folder.
   The library management opens.
3. Select "Released types" from the "Filter" drop-down list.
   The "Types" area only displays types that have released versions.

Filtering according by types with new versions for upgrading other types

1. Select the "Types" folder in the project library.
2. Select the "Library management" command in the shortcut menu of the "Types" folder.
   The library management opens.
3. Select "Types with new versions for upgrading other types" from the "Filter" drop-down list.
   The "Types" area only displays types that can be potentially used as a source for upgrading other types.

See also

Opening library management (Page 1942)

Creating a faceplate type (Page 270)

14.2.5.4 Creating a global library

Introduction

In the libraries you store the configured objects that you want to use several times in your configuration. To use objects in several projects, create a global library.

Requirement

- A project is open.
- The "Libraries" task card is displayed or the "Library view" is open.

Procedure

1. Click the icon under "Global library".
   The "Create new global library" dialog opens.

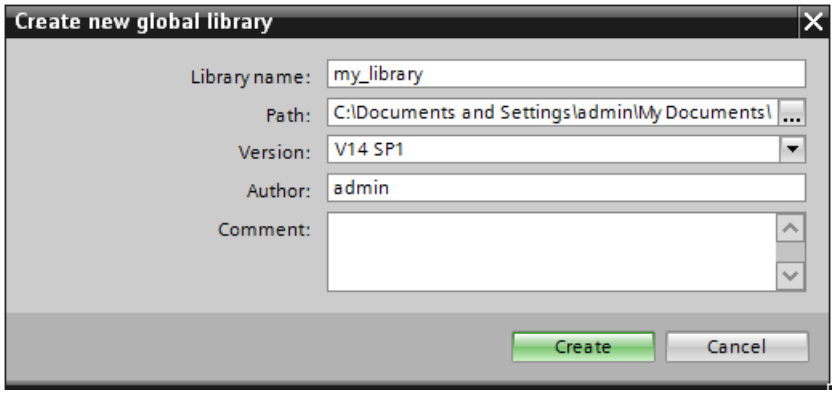

2. Enter a name.
3. Select the path where the new library is to be stored.
4. Click "Create".

Result

The new library is shown in the "Global libraries" palette. The global library contains the "Types", "Master copies" and "Common data" folders. Under the "Common data" you can find reports for the global library.

A folder with the name of the global library is created in the file system at the storage location of the global library. This actual library file is given the file name extension ".alxx", whereby "xx" stands for the current WinCC version number.

See also

Basics on libraries (Page 1934)

14.2.5.5 Saving a global library

Introduction

A global library is stored in a separate file on your hard disk drive. The file contains the objects of the global library including the referenced objects. E.g. the reference of a tag which was configured on an I/O field is also saved in the library.

WinCC prompts you to save the global libraries when you close WinCC or your project without saving. You also can store the global library during configuration, without storing the entire project.

Requirement

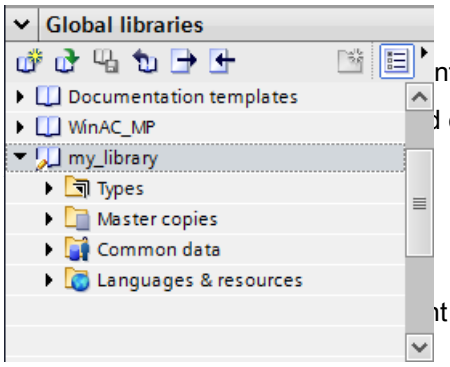

- ntains at least one library.
- d or the library view is open.
-

Procedure

1. nt to save.
2. Click the icon in the "Global library" palette.

You can alternatively select the "Save Library" command in the shortcut menu.

If you want to store the global library in a different folder, select "Save as" in the shortcut menu. Select the path in which you want to store the new library and enter a file name.

Result

The global libraries are saved under their current file name or a file name you have specified.

See also

Basics on libraries (Page 1934)

14.2.5.6 Opening a global library

Introduction

In WinCC, the global libraries are stored in separate files. You can use a global library in every project.

Requirement

- You have saved a global library.
- A project is open.
- The "Libraries" task card is displayed or the library view is open.

Procedure

1. Click the icon in the "Global library" palette.
   The "Open global library" dialog box is displayed.
2. Select the path in which the library is stored.
3. Click "Open".

Note

To have the access to a global library from multiple projects, open the global library read-only. As soon as a global library is not opened read-only, access from other projects is blocked.

Result

WinCC displays the opened global library in the "Global library" palette.

See also

Basics on libraries (Page 1934)

14.2.5.7 Displaying logs of global libraries

Logs listing all changes made to the global library are created when global libraries are updated. The logs are stored together with the global library and are always available once you have opened the global library.

Procedure

To open the logs of a global library, follow these steps:

1. Open the global library in the "Libraries" task card or in the library view.
2. Open "Common data > Logs" in the lower-level folder.
3. Double-click the required log.
   The log opens in the work area.

14.2.5.8 Updating a project with the contents of a library

Introduction

After you have edited several types in the project library, update all instances in the project to the most recent version of the types from the project library.

Each of the following elements can be selected as source for the update:

- Individual folders within a library
- Individual types

Requirement

The "Libraries" task card or the library view is open.

Procedure

1. Select a folder within the project library or individual types.
2. Select "Update > Project..." from the shortcut menu.
   A dialog opens.
3. Select either the entire project or individual devices for the update.
4. Select the "Delete all unused versions of affected types" check box to delete all older versions of the updated types from the project library.
5. Click "OK" to confirm.

Result

All instances of the types are updated in the project to the most recent version of the selected types in the project library.

You can find a log of the update process in the project tree under "Common data".

14.2.5.9 Updating a library with the contents of another library

The following options are available for updating libraries:

- Updating a global library with types from another global library
- Updating the project library with types from a global library

Each of the following elements can be selected as source for the update:

- Individual folders within a library
- Individual types

Requirement

To update a global library, open the library with write rights.

Requirement

The "Libraries" task card or the library view is open.

Procedure

To update a library with the contents of a different library, follow these steps:

1. Select a folder within the library or individual types.
2. Right-click on the source and select "Update > Library..." from the shortcut menu.
   The "Update library" dialog box opens.

3. Select the type of library you want to update:
   - Select "Update the project library" to update the project library with types from a global library.
   - Select "Update a global library" if you want to update a global library.
4. Optional: Select the global library you want to update from the drop-down list.
5. Select the "Delete all unused versions of affected types" check box to delete all older versions of the updated types from the project library.
6. Click "OK" to confirm.

Result

- Types not yet available in the target library are supplemented there with all their versions. More recent versions are added to the types that already exist in the target library.
  If a more recent version of a type already exists in the target library, the latest version is nevertheless copied and automatically assigned a newer version number.
- A log listing all performed changes to the target library is created for the update process. If you have updated the project library, you can find the log in the project tree under "Common data > Logs".
  If you have updated a global library, you can find the log in the "Common data > Logs" folder in the level below the global library.

14.2.5.10 Upgrading types of a library

Description

Your project contains objects which use or reference other objects, e.g. user data types. You have created a project library and stored the library types used in the project library. You have continued to edit the types outside the project. Reintegrate the new versions of the edited types into the project library.

With the help of the "Upgrading types" function, you can recognize all the referenced types which are dependent on a changed type and automatically upgrade them in the entire project. The upgraded types are set to the status "in progress" or "in test" in the project library. You can thus see at a glance which types were changed in order to test and release them.

Requirement

- The project library has been updated with the types of the global library.
- The new type versions have been integrated into the existing project library.

Procedure

1. Open the library management.
2. In the "Types" area, select "Types with new versions for upgrading other types" from the drop-down list.
   All types for which a new type version is available are displayed in the project library.

3. Click the "Types" folder and select "Show upgradable types" from the shortcut menu.
   All referenced types are displayed in the "Upgrading types" tab.
4. Select one of the options under "Upgrading types":
   – "Only in a test environment" in order to display all types that reference an instance in the test environment.
   – "Compatible with selected test environment" in order to display all types that can reference an instance in the test environment. The instance was not yet created.
5. Select a test environment.
   All of the types that must be upgraded are displayed.

   Note

   If one of the types to be upgraded has the "Error" status, the types cannot be upgraded. First correct the error and restart the procedure.

6. Click "Start".
   The "Upgrading types" dialog opens after successful upgrade.
7. Check the upgraded types and then release them.

Result

All types in your project were upgraded and reference the current type versions.

See also

Updating a project with the contents of a library (Page 1947)

Updating a library with the contents of another library (Page 1948)

14.2.5.11 Exporting and importing library texts

Introduction

You can export the texts of the library objects to an .xlsx file to edit them in MS Excel, for example, or export them for compilation.

You export and import texts of the following objects in the library:

- Individual library types and master copies
- Multiple library types and master copies
- All library objects of the project library or a global library

After editing or external compilation, you import the texts into the TIA Portal.

When the texts are imported, all texts from the import file are imported for the entire library, even if you have only selected one library object. The target languages of the import file must be activated in the project.

During import to a master copy, the texts of the template are overwritten in the library with the new texts from the import file. During import of the texts to a library type, the latest version is

overwritten in the library with the new texts from the import file. If a version of a type has not been released yet in a project library, no texts can be imported for the entire project library.

Note

Restrictions for text import

The texts which belong to the following library objects cannot be imported:

- Type instances which are contained in a master copy
- Library types whose versions have not yet been released and which have the status "in progress" or "in test"
- Write-protected global library

Defining the source language and target language

You define the source language and target language for export of the texts in the Export dialog. The selection of available languages depends on the project languages defined.

With the global library, the selection of available source and target language depends on the languages defined by the creator of the library. To see the available languages of the global library, double-click the entry "Library languages" in the project folder "Language and Resources" of the library in question.

Exporting texts

To export the texts of a single or several library objects, follow these steps:

1. Open the library.
2. Select the desired library object in the library.
3. Select the command "Export library texts" in the shortcut menu of the object.
   Alternatively, click the "Export library texts" button in the toolbar.
   The "Export" dialog box opens.
4. Select the source language and the target language for export in the dialog.

5. Enter the name and the path for the export file.

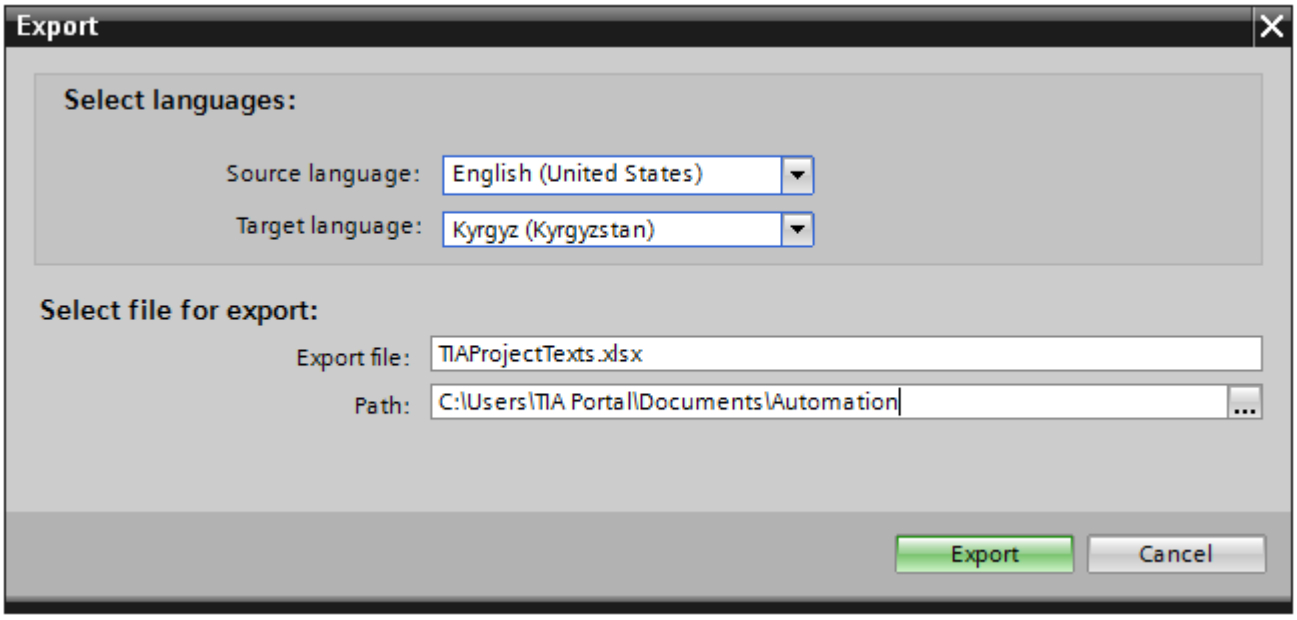

6. Click "Export".
   After successful export, the export file is stored under the specified path.

Importing texts

To import the texts after editing or compilation into the TIA Portal again, follow these steps:

1. Open the project library or the global library.
2. Select the command "Import library texts" in the shortcut menu of the object.
   Alternatively, click the "Import library texts" button in the toolbar.
   The "Import" dialog box opens.
3. Select the path and the name of the import file from the "Select file for import" field.
   Activate the "Import source language" check box if you have made changes to the source language in the export file and would like to overwrite the entries in the project with the changes.
4. Click on "Import".

14.2.6 Managing objects in a library

14.2.6.1 Displaying library objects

Introduction

The libraries are displayed as file folders in the corresponding palette. The elements contained in the library are displayed in the file folder and in the "Elements" palette.

Requirement

- At least one library object has been created in a library.
- The "Libraries" task card is opened.

Procedure

1. Select the library in the corresponding palette whose library objects you want to display.

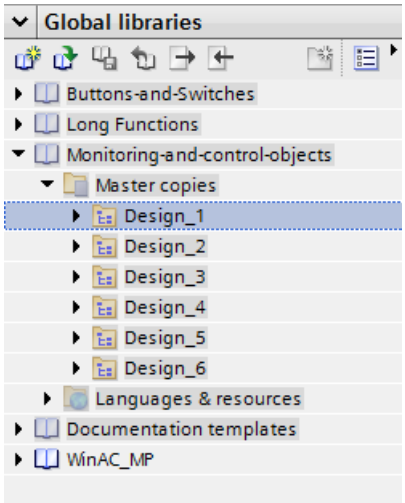

2. Click .
   The contained library objects are displayed in the "Elements" palette.

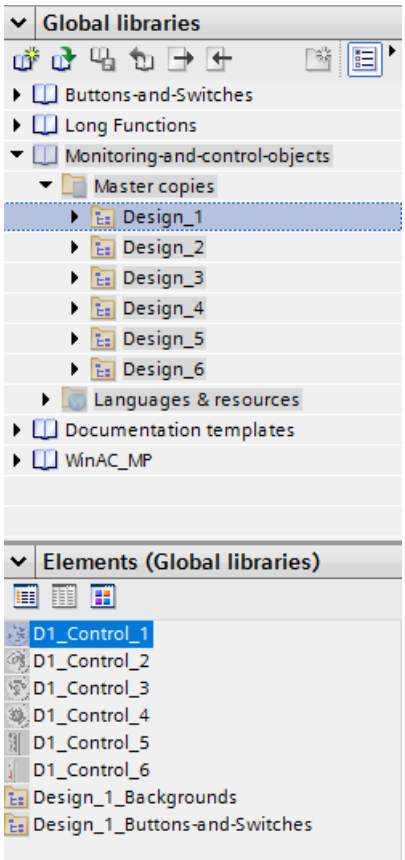

3. Click one of the following icons:

Icon	Description
	Element view in detailed mode
	Element view in list mode
	Element view in overview mode with icons

When several objects are assigned to the library with a multiple selection, only one of the objects is shown in the "Elements" palette. The individual components of this element are displayed in the "Parts" palette.

Show parts of the library objects

1. Select the library in the corresponding palette from which you want to view the components of an element.
2. Click .
3. The contained library objects are displayed in the "Elements" palette.
4. Select the element.
   The "Parts" palette shows the objects of which the element consists.

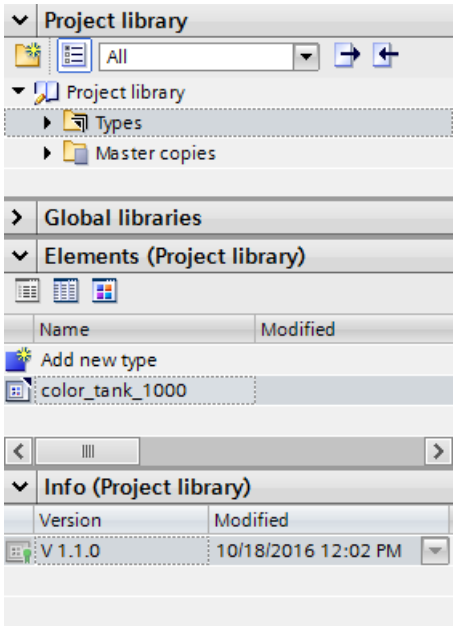

Result

The library objects are displayed in accordance with the configuration. The components of the faceplates are displayed.

See also

Basics on libraries (Page 1934)

14.2.6.2 Storing an object in a library

Introduction

You can store all of WinCC objects, such as screens, tags, graphic objects or alarms in your libraries. You can use drag-and-drop to move the corresponding object from the work area, project window or detail view to the library. In a library you have divided into categories, you can directly add objects to a specific category.

Requirement

- The "Screens" editor is open.
- A screen object has been created in the work area of the screen.
- The created libraries are displayed.

Procedure

1. Select the object in the work area of the "Screens" editor.
2. Drag-and-drop the object from the work area to the desired library.
   The mouse pointer is transformed into a crosshair with an appended object icon.

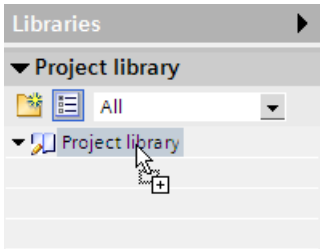

Result

The object is saved to the library for further use in multiple instances of your configuration.

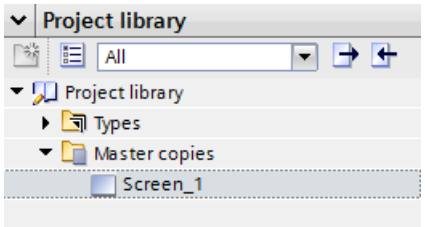

See also

Basics on libraries (Page 1934)

14.2.6.3 Inserting a library object

Introduction

The system always assigns the inserted library object a name which consists of the name of the object type and of a consecutive number.

If the inserted object already exists, you have the option of replacing the object or of saving it under a new name.

You cannot insert library objects that are not supported by the HMI device.

Note

If you insert a screen with interconnected template from the library, the template will also be inserted. Any existing matching template is not used.

Requirement

- The "Libraries" task card is opened.
- The editor in which you want to insert the library object is open.

Procedure

1. Select a library object from the library.
2. Drag-and-drop the library object to the position in the work area where you want to insert the object.
   The library object is inserted.

Result

If the object was contained in the "Copy templates" folder, you have inserted an independent copy of the library object in the editor.

If the object was contained in the "Types" folder, you have inserted an instance of the library object in the editor.

See also

Basics on libraries (Page 1934)

14.2.7 Using types and their versions

14.2.7.1 State of type versions

Introduction

Depending on the point of use, the version of a type has different statuses.

Released version

The "released version" status is available for all types, regardless of the point of use.

If you want to edit a released version, you must first create a new test version or an "in progress" version.

Released type versions of scripts and screens can be opened and viewed at their instance.

"In progress" version

The "in progress" status is possible for the following types:

- Faceplates
- User data types (HMI UDT)
- Styles
- Style sheets

When you create a new type or a new version of a released type, the type is assigned the status "in progress".

Types with the "in progress" status can be edited in the library view without the need for a reference to an instance in the project. Upon release, the compatibility of the type is tested by a consistency check.

"In test" version

Only versions of scripts and HMI screens have the "In test" status.

When you create a new version of a type, the type is assigned the status "In test".

A version with "In test" status is linked to an instance in the project. You can set only one version to "In test" for each type at a given time.

A version in testing may only be linked to a single instance in the project. Therefore, it is not possible to copy an instance to the clipboard, to duplicate it or to create an additional type from the instance as long as it has "In test" status.

See also

Generating a script as a type (Page 1959)

Generating a screen as a type (Page 1959)

Create a new version of a type (Page 1961)

Master copies and types (Page 1938)

Creating a faceplate type (Page 270)

14.2.7.2 Generating a script as a type

Requirement

- A project is open.
- An HMI device has been created and opened.
- The project tree is open.
- The "Libraries" task card is opened.

Procedure

1. Open the "Scripts" editor in the project tree.
2. Create a new script.
3. Select the script in the project tree.
4. Drag-and-drop the script into a library in the "Libraries" task card. A dialog opens.
5. Enter a name.
6. Enter a comment.

Result

You have created a type version of a script in the library. The created type is stored as a released version in the library. An instance of the type is used in the project.

To change the script create a new version of the script.

See also

State of type versions (Page 1958)

14.2.7.3 Generating a screen as a type

Requirement

A project is open.

An HMI device has been created and opened.

The project tree is open.

The "Libraries" task card is opened.

Procedure

1. Open the "Screens" editor in the project tree.
2. Create a new screen.
3. Select the screen in the project tree.
4. Drag-and-drop the screen into a library in the "Libraries" task card. A dialog opens.
5. Enter a name.
6. Enter a comment.

Result

You have created a type of a screen in the library.

The created type is stored as a released version in the library. An instance of the type is used in the project.

To change the screen create a new version of the screen.

See also

State of type versions (Page 1958)

14.2.7.4 Generating a style as a type

Introduction

To define a new style, add a new type in the project library.

Adding a new style

1. Open the "Libraries" task card.
2. Select the "Add new type" command under "Types" in the shortcut menu of the project library. A dialog opens.
3. Select "HMI style".

Result

The new style is created and displayed under its selected name in the project library. The HMI style type is assigned the status "in progress" and the version 0.0.1.

14.2.7.5 Creating a style sheet as a type

Introduction

To define a new style sheet, add a new type in the project library.

Adding a new style sheet

1. Open the "Libraries" task card.
2. Select the "Add new type" command under "Types" in the shortcut menu of the project library. A dialog opens.
3. Select "HMI style sheet".
4. Assign a meaningful name to the style sheet.
5. Select the style sheet category from the list.

Result

The new style sheet is created and displayed under its selected name in the project library.

The HMI style sheet type is assigned the status "in progress" and the version 0.0.1.

14.2.7.6 Create a new version of a type

Principle

If you create a new version of a type, the point of use of the type determines the status of the newly created version.

Requirement

The "Libraries" task card is opened.

A type has been created and released.

Procedure

1. Select the released type.
2. Select "Edit type" in the shortcut menu.

Result for types of faceplates, user data, and styles

A new version of the type is created.

The version has the status "in progress". The library view opens.

Result for types of scripts and screens

A dialog opens.

After you have selected the settings in the dialog, the version is set to the status "In test". The instance used in the project is set to the status "In test". The library view opens.

See also

State of type versions (Page 1958)

14.3 Importing and exporting project data

14.3.1 Data exchange between projects and external applications

Introduction

WinCC gives you the option of exchanging project data between the projects or copying them to external applications.

Exporting and importing between projects

You can export the following project data from a project and import them into another project.

- Recipe data records
- Alarms
- Tags
- Text lists
- Project texts

Exporting and importing reduces the workload. Instead of creating new data records, you use data already created in previous projects.

Editing the export file

The following file formats are available for export and import depending on the editor:

- *.xlsx for alarms, tags, project texts and text lists
- *.csv for recipe data records

You can edit the import file in Excel, for example.

XLSX file format

XLSX format is a file format for Excel tables based on the Open XML format. XLSX files are optimized for Microsoft Excel 2007.

You can sort the columns as required in the XLSX file.

CSV file format

CSV stands for Comma Separated Value. In this format, the columns of the table that contain the names and the value of the entry are separated by semicolons. Each table row terminates with a line break. You can also open the CSV file for editing in Excel.

Importing project data

When project data is imported, the objects in the project are created.

The syntax of the import file is checked during import. The accuracy of the values imported and dependencies between the imported values are not checked.

Any errors found in the imported data are reported when the project is compiled.

Copying in Excel format

In all spreadsheet editors you can copy the content in Excel format into the buffer of your PC. Then you insert the project data in Excel format directly into any application outside the TIA Portal. To this purpose you use the corresponding command in the shortcut menu of the work area:

- If you select the command in the shortcut menu of the line header, the complete line is copied into the buffer.
- If you select the command in the shortcut menu of a cell, only the cell content is copied into the buffer.
- If you select several lines and select the command, all marked data are always copied into the buffer.

This data exchange is only possible as an export.

See also

HMI device wizard basics (Page 1929)

14.3.2 Importing and exporting recipe data records

14.3.2.1 Exporting recipe data records (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

WinCC features an export function for exporting data records from recipes.

Requirements

- The WinCC project for the export is open.
- Recipes have been created in a project.
- The "Recipes" editor is open.

Exporting recipes

1. In the "Recipes" editor, select the recipe with the data records you want to export.
2. Click .
   The "Export" dialog box opens.

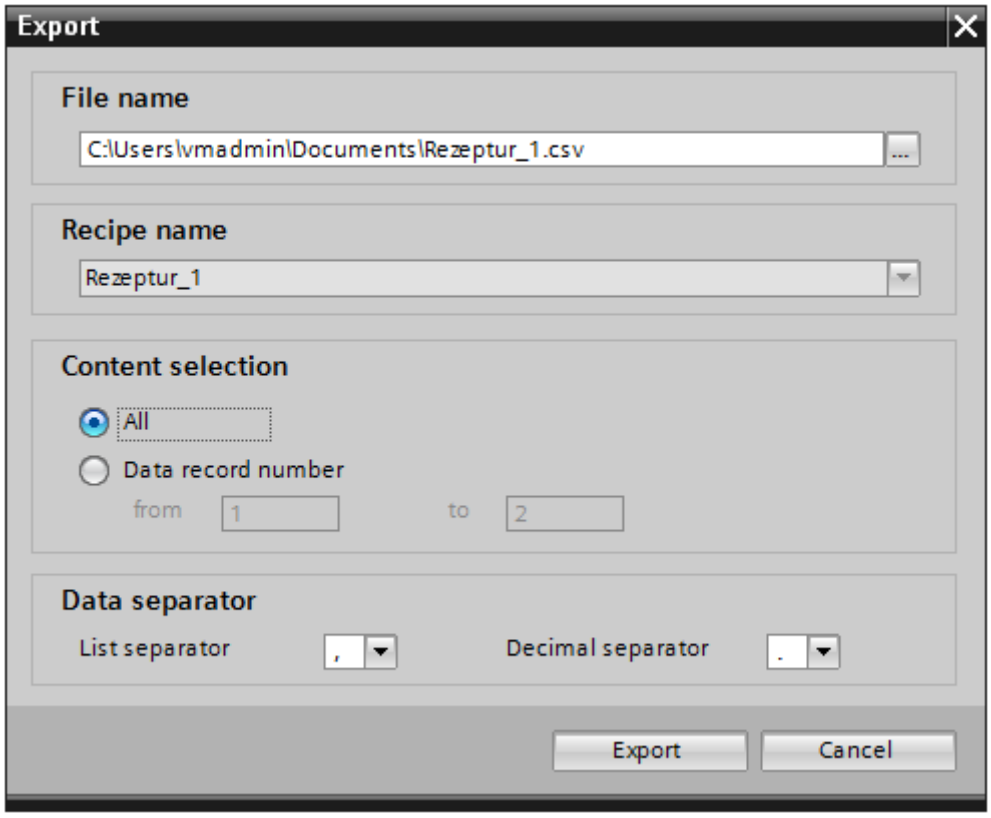

   The selected recipe is shown under "Recipe selection".
3. Under "Content selection", specify if all or only selected data records are to be exported.
4. Under "File selection", specify the file in which the recipe data is to be stored.
5. Specify the list separator and decimal separator under "Data separation".
6. Click "Export."
   The export will start.

Result

The exported data has been written to a CSV file. The CSV file will be stored in the specified directory.

See also

Importing recipe data records (Page 1966)

14.3.2.2 Importing recipe data records (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Recipes are identified by their name. The recipe name must therefore be unique. Open the import file in a simple text editor to check that it has the correct data structure.

Specify whether or not existing data records should be overwritten by records with the same name during the import.

Requirements

- A CSV file containing at least one recipe has been created.
- The WinCC project for the import is open.
- The "Recipes" editor is open with at least one recipe.

Importing a recipe

1. In the "Recipes" editor, select the recipe with the data records you want to import.
2. Click .
   The "Import" dialog box opens.

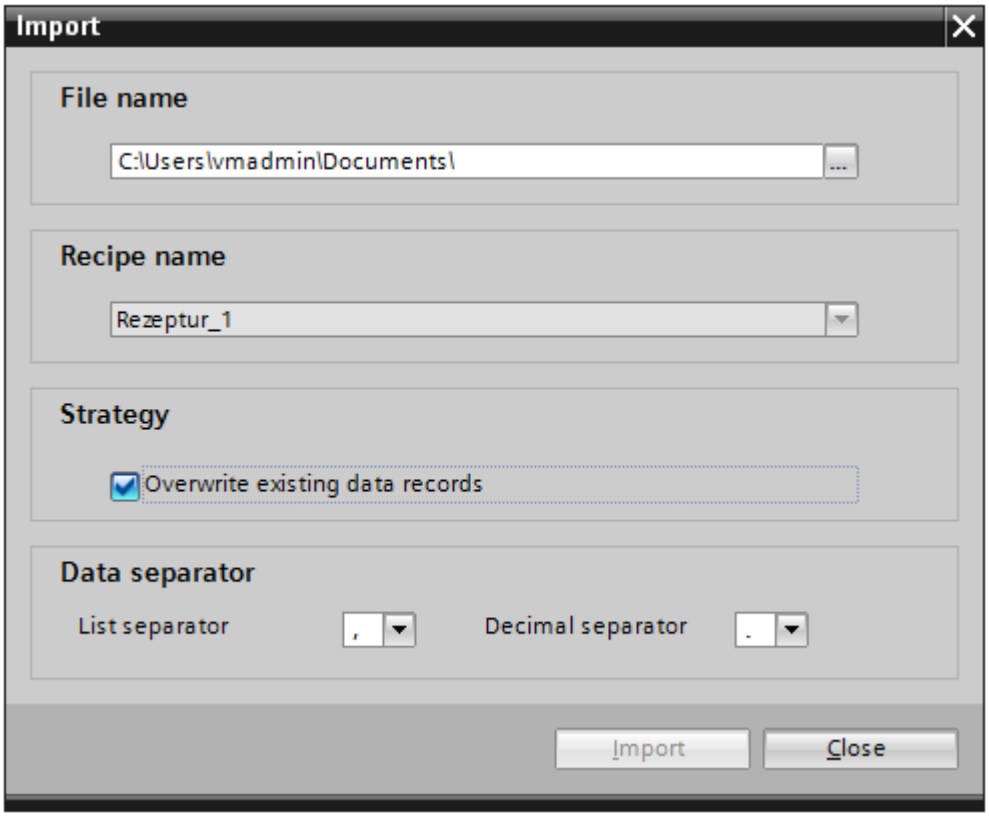

The selected recipe is shown under "Recipe selection".

3. Select the file you want to import under "File selection".
4. Under "Strategy", specify if existing data records should be overwritten by records of the same name.
5. Under "Data separation", select the list separator and the decimal separator to use in the CSV file.
6. Click "Import".
   The import will start.

Result

The data records are created in the selected recipe. Depending on the setting for "Strategy", existing data records are overwritten by records with the same name from the CSV file.

Existing data records with the same name will also be imported from the CSV file if you deactivate the "Overwrite existing data records" option.

See also

Exporting recipe data records (Page 1964)

14.3.2.3 Exporting recipe data records (RT Professional)

Introduction

WinCC features an export function for exporting data records from recipes.

Requirements

- The WinCC project for export is open.
- Recipes have been created in a project.
- The "Recipes" editor is open.

Exporting recipes

1. In the "Recipes" editor, select the recipe with the data records you want to export.
2. Click .
   The "Export" dialog box opens.

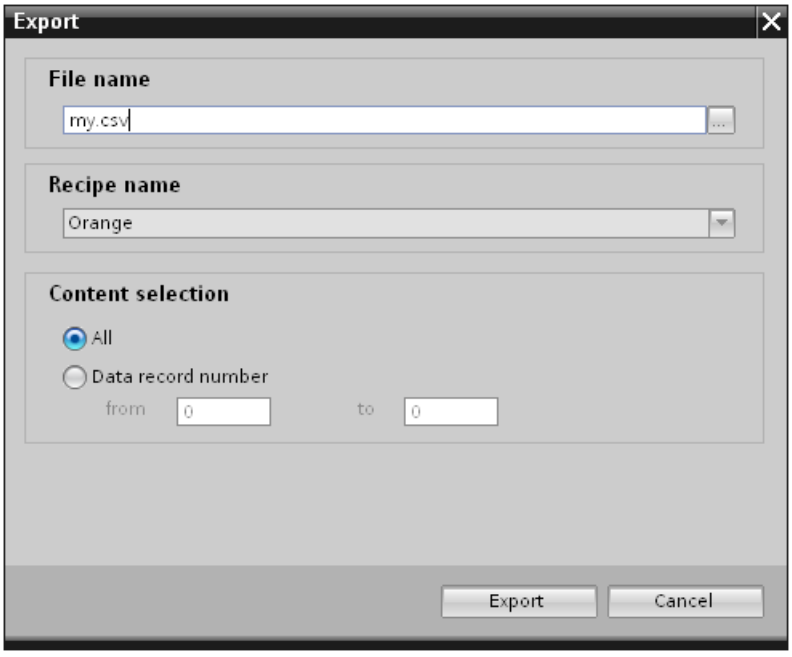

   The selected recipe is displayed under "Recipe name".
3. Under "File name", specify the file in which the recipe data is to be stored.
4. Under "Content selection", specify if all or only selected data records are to be exported.
5. Click "Export."
   The export will start.

Result

The exported data has been written to a CSV file. The CSV file will be stored in the specified directory.

14.3.2.4 Importing recipe data records (RT Professional)

Introduction

Recipes are identified by their name. The recipe name must therefore be unique. Open the import file in a simple text editor to check that it has the correct data structure.

Specify whether or not existing data records should be overwritten by records with the same name during the import.

Requirements

- A CSV file containing at least one recipe has been created.
- The WinCC project for import is open.
- The "Recipes" editor is open with at least one recipe.

Importing a recipe

1. In the "Recipes" editor, select the recipe with the data records you want to import.
2. Click .
   The "Import" dialog box opens.

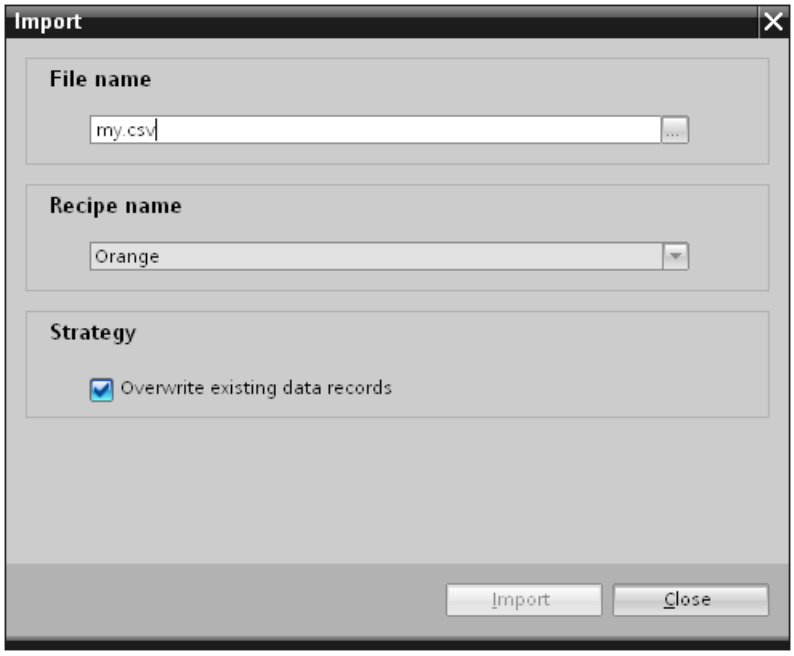

   The selected recipe is displayed under "Recipe name".
3. Select the file you want to import under "File name".
4. Under "Strategy", specify if existing data records should be overwritten by records of the same name.
5. Click "Import".
   The import will start.

Result

The data records are created in the selected recipe. Existing data records may be overwritten by records with the same name from the CSV file depending on the setting for "Strategy".

14.3.2.5 Editing exported recipe data records

Introduction

You can edit the exported recipe data records and import them again. You can, for example, add new data records to the file or modify existing records. You edit the exported recipe data records in CSV files in Microsoft Excel.

Procedure

1. Start Microsoft Excel.
2. In the "File" menu, click "Open".
   The "Open File" dialog box opens.
3. Select the path in which the exported data record file is stored.
   In the "Open" dialog, select "All Files (*. *)" option in the drop-down list so that the exported file is detected.
4. Click "Open".
   The "Text Conversion Wizard" dialog opens.
5. Click the "Next" button in the next step.
6. In the second step, select the semicolon as a delimiter.
   Click "Next".
7. Close the wizard with the "Finish" button.
8. Make the desired changes to the file.
9. Save the changed file in CSV format (comma delimited).
10. Import the changed file.
    If you have saved the file under a different name, adapt the file name in the "Import" dialog before importing.

Result

The exported recipe data records were edited and saved in a recipe.

See also

Importing recipe data records (Page 1968)

14.3.2.6 Format of data from recipe data records (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

This section describes the required format of the file for the import of recipes. The file containing the data of the recipes must be available in "*.csv" format. :

Structure of recipe data

The structure of the import file is fixed. The following example shows the structure of a recipe containing two recipe elements, each with two data records:

```
"ID";<Name, recipe element 1>;< Name, recipe
element 2>;"LastUser";"LastAccess[UTC]"<Line break>
<Recipe data record number 1>;<Recipe data record 1 value 1>;<Recipe
data record 1 value 2>;"<Last user 1>";<Date and time 1><Line break>
<Recipe data record number 2>;<Recipe data record 2 value 1>;<Recipe
data record 2 value 2>;"<Last user 2>";<Date and time 2><Line break>
```

ID of the language

The recipe always has the first line, which is the header and describes the column headings. Only the names of the recipe elements are added dynamically.

Each additional line represents one recipe data record with its recipe element values and change information.

14.3.3 Importing and exporting alarms

14.3.3.1 Exporting alarms (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

WinCC has an export function for alarms.

Requirements

- The WinCC project for the export is open.
- Alarms have been created in the project.
- The "HMI alarms" editor is open.

Exporting alarms

1. Click the button in "Discrete alarms" or "Analog alarms".
   The "Export" dialog box opens.

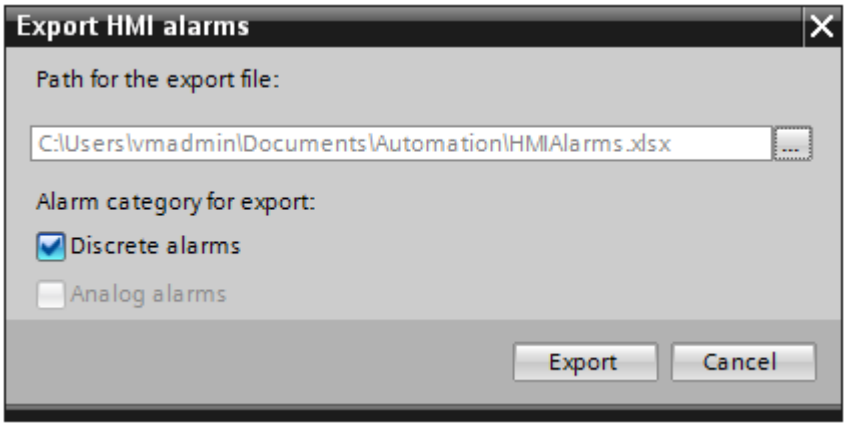

2. Click the "..." button and specify in which file the data are saved.
3. Specify whether you want to export "Disrete alarms" or "Analog alarms".
4. Click "Export". The export will start.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

See also

Format of the analog alarm data (Page 1980)

Format of the discrete alarm data (Page 1983)

14.3.3.2 Importing alarms (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Alarms are identified by their alarm number. The alarm numbers must be unique in the analog and discrete alarm types. Alarms with redundant alarm numbers will be overwritten. An alarm without an existing alarm number is created.

Any empty list entries for existing alarms contained in an xlsx file will be ignored for the purposes of the import. The entries of the existing alarms remain active and will not be replaced by empty ones.

Requirements

- An xlsx file with alarms has been created.
- The structure of the xlsx file meets the requirements.

- The WinCC project for the import is open.
- The "HMI alarms" editor is open.

Importing alarms

1. Click the button in "Discrete alarms" or "Analog alarms". The "Import" dialog box opens.

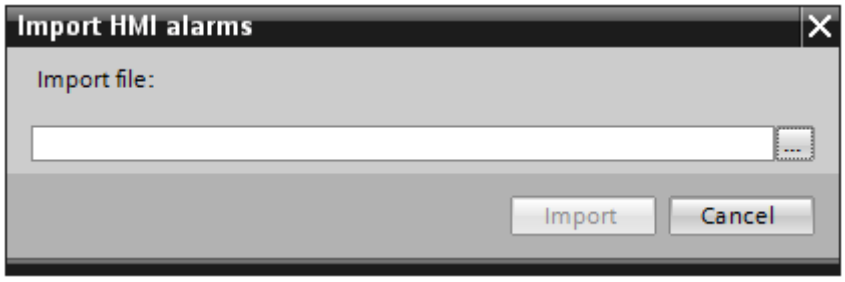

2. Click the "..." button and select the file that you want to import.
3. Click on the "Import" button. The import will start. A progress bar indicates the progress of the import operation.

Result

The corresponding alarms including alarm texts are created in WinCC on the basis of the import data. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example dynamic parameters such as tags. These parameters must be available in all languages to be imported.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign an alarm a trigger tag of the incorrect type, such as String. An error will be reported during compilation.

See also

Format of the analog alarm data (Page 1980)

Format of the discrete alarm data (Page 1983)

14.3.3.3 Importing alarms (RT Professional)

Introduction

Alarms are identified by their alarm name. An existing alarm will be overwritten with the data from the xlsx file if the alarm name already exists in the project. A new alarm is created if the alarm does not yet exist.

The alarm numbers must be unique in the project. If an alarm number is assigned more than once by import, the alarm is imported and the alarm number marked as invalid.

Requirements

- An xlsx file with alarms has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.
- The "HMI alarms" editor is open.

Importing alarms

1. Click the button in "Discrete alarms" or "Analog alarms". The "Import" dialog box opens.

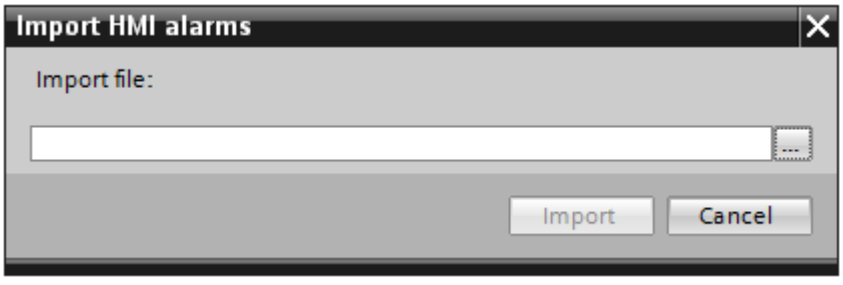

2. Click the "..." button and select the file that you want to import.
3. Click on the "Import" button. The import will start. A progress bar indicates the progress of the import operation.

Note

There are no option settings for alarm import.

Result

The corresponding alarms including alarm texts are created in WinCC on the basis of the import data. Messages relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign a trigger tag of an incorrect type, such as string, to an alarm. An error will be reported during compilation.

14.3.3.4 Format of the analog alarm data (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

This chapter describes the required format of the file for the import of analog alarms. The file containing the analog alarm data must be in "*.xlsx" format.

Structure of the alarm data

The import file in Microsoft Excel consists of a number of worksheets:

- Analog alarms(Analog alarms)
- Limits (Limits)

Each alarm is assigned a separate row in the import file. The import file with the analog alarms must be formatted as follows:

Example of the worksheet "Analog alarms"

	A	B	C	D	E
1	ID	Name	Event text [en-US], Alarm text	FieldInfo [Alarm text]	Class
2	1	Analog_alarm_1	AA1 Error-AC with maximum text length: abcdefghijklmnopqrstuvwxyz		Errors
3	2	Analog_alarm_2	AA2 Warning-AC this text should be bold		Warnings
4	3	Analog_alarm_3	AA3 SDm-AC this text should be italic		SDm
5	4	Analog_alarm_4	AA4 SDo-AC <u>this text should be underlined</u>		SDo
6	5	Analog_alarm_5	AA5 SystemAcknowledgement-AC <blink>this text should be flashing<		System_Acknowle
7	25	Analog_alarm_25	Internal AA23 switchDT: Deadband mode in case of violation - value HL		AADT-Internal
8	26	Analog_alarm_26	Internal AA23 switchDT: Deadband mode in case of violation - percent		AADT-Internal
9	31	Analog_alarm_31	Internal AA23 switchDT: Low limit violation static		AADT-Internal
10	32	Analog_alarm_32	Internal AA23 switchDT: High limit violation static		AADT-Internal
11	33	Analog_alarm_33	Internal AA23 switchDT: Low limit violation dynamic		AADT-Internal
12	34	Analog_alarm_34	Internal AA23 switchDT: High limit violation dynamic		AADT-Internal
13	35	Analog_alarm_35	Internal AA23 switchDT: delay 3 seconds High limit violation		AADT-Internal
14	42	Analog_alarm_40	Internal AA23 switchDT: delay 3 seconds Low limit violation LLV		AADT-Internal
15	23	Analog_alarm_41	AA23 DT in event text: <field ref="0" /> Bool, <field ref="1" /> Byte, <fi	<ref id = 0; type = AlarmTag; T	ACAT
16	24	Analog_alarm_42	AA24 DT in event text: <field ref="0" /> Timer, <field ref="1" /> Counte	<ref id = 0; type = AlarmTag; T	ACAT

Table 14-1 Meaning of the entries

List entry	Meaning
ID	The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Name	Name of the analog alarm
Event text [de-DE], Alarm text	Displays the alarm text. The field designation contains a language ID. Alarm texts must be assigned a language ID for import. An expression with a reference ID will be added to the alarm text if the text has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign dynamic parameters to alarm texts.
FeldInfo	Specifies whether the alarm text contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = AlarmTag; Tag = Tag1; DisplayType = Decimal; Length = 5;> Text list: <ref id = 1; type = CommonTextList; TextList = Textlist1; Tag = tag 2; Length = 5;>
Class	The class of an alarm determines whether or not the alarm must be acknowledged. It can also be used to determine how the alarm appears when it is displayed on the HMI device. The alarm class also determines whether and where the corresponding alarm will be logged.
Group	Indicates the allocation to an alarm group. If an alarm belongs to a group with other alarms, it can be acknowledged together with these alarms of the same group in a single operation.
Trigger tag	Specifies the tag monitored for limit value violation.
Delay time value	Specifies the delay time. The alarm is not triggered until the duration of the limit value violation equals the specified delay time.

List entry	Meaning
Delay time unit	Specifies the time unit for the delay.
Report	Enables reporting of the specific alarm on a printer. True or "1" = Reporting enabled. False or "0" = Reporting disabled. Reporting must also be globally enabled in the project.
Info text [de-DE], Info text	The tooltip is an optional property of an alarm. Tooltips can contain additional information about the alarm. A tooltip will be displayed in a separate window on the HMI device when the operator presses the <HELP> key. The field designation contains a language ID.

Example of the worksheet "Limits"

	A	B	C	D	E	F	G
1	Alarm ID	Limit type	Limit value	Limit mode	Deadband mo	Deadband valu	Deadband in percent
2	1	Constant	0	Upper limit	Off	0	False
3	2	Constant	1	Upper limit	Off	0	False
4	3	Constant	2	Upper limit	Off	0	False
5	4	Constant	3	Upper limit	Off	0	False
6	5	Constant	4	Upper limit	Off	0	False
7	25	Constant	50	Upper limit	On both	5	False
8	26	Constant	50	Upper limit	On both	10	True
9	31	Constant	50	Lower limit	Off	0	False
10	32	Constant	50	Upper limit	Off	0	False
11	33	Tag	AASDTdyn	Lower limit	Off	0	False
12	34	Tag	AASDT1dyn	Upper limit	Off	0	False
13	35	Constant	50	Upper limit	Off	0	False
14	36	Constant	50	Lower limit	On both	5	False

Table 14-2 Meaning of the entries

List entry	Meaning
Alarm ID	Alarm number The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Limit mode	Trigger mode Indicates the method used for monitoring the limit value.
Limit type	Specifies the limit that will be monitored. Both a tag and a constant can be used as limit value.
Limit value	Limit value Indicates the tag or constant monitored for limit violation.

List entry	Meaning
Deadband mode	Hysteresis mode Specifies whether and in which cases hysteresis will be used. For "Outgoing" For "Incoming" For "Incoming" and "Outgoing"
Deadband in percent	0 = The value specified for "Hysteresis" is considered to be absolute. 1 = The value specified for "Hysteresis" is referred to as a percentage of the limit value.
Deadband mode	Hysteresis Specifies a constant as a value of the hysteresis.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing alarm of the same name.

See also

Exporting alarms (Page 1971)

Importing alarms (Page 1972)

Format of the discrete alarm data (Page 1978)

14.3.3.5 Format of the discrete alarm data (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

This chapter describes the required format of the file for the import of discrete alarms. The file containing the discrete alarm data must be in "*.xlsx" format.

Structure of the alarm data

The import file in Microsoft Excel consists of the worksheets "Discrete alarms" (discrete alarms). Each alarm is assigned a separate row in the import file. Structure of the import file containing the discrete alarms:

Example of the worksheet "Discrete alarms"

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ID	Name	Event text [en-US], Alarm text	FieldInfo [Alarm text]	Class	Trigger tag	Trigge	Acknowle	Ackno	PLC ackno	PLC ackn	Group	Report	Info text [en-US], Info text
2	1	Discrete_alarm_1	DA1 Error-AC with maximum text length: abcdefghijklmnopqrstuvwxyz 1234/na		Errors	HMI_AC-Test	0	<No value	0	<No value	0	<No value	False	Infotext DA1 Error
3	2	Discrete_alarm_2	DA2 Warning-AC this text should be bold		Warnings	HMI_AC-Test	1	<No value	0	<No value	0	<No value	False	Infotext DA2 Warning
4	3	Discrete_alarm_3	DA3 SDm-AC this text should be italic		SDm	HMI_AC-Test	2	<No value	0	<No value	0	<No value	False	Infotext DA3 SDm
5	4	Discrete_alarm_4	DA4 SDo-AC <u>this text should be underlined</u>		SDo	HMI_AC-Test	3	<No value	0	<No value	0	<No value	False	Infotext DA4 SDo
6	5	Discrete_alarm_5	DA5 SystemAcknowledgement-AC <blink>this text should be flashing</blink		System_Ackn	HMI_AC-Test	4	<No value	0	<No value	0	<No value	False	Infotext DA5 SystemAcknowledgement
7	6	Discrete_alarm_6	DA6 SystemNoAcknowledgement-AC mixed text: Bold, Italic, <i		System_No_A	HMI_AC-Test	5	<No value	0	<No value	0	<No value	False	Infotext DA6 SystemNoAcknowledgement
8	7	Discrete_alarm_7	DA7 DT in event text: <field ref="0" /> Integer, <field ref="1" /> Real, <field re	<ref id = 0; type = AlarmTag; Tag = PI	ACAT	HMI_TriggerAT	0	<No value	0	<No value	0	<No value	False	
9	8	Discrete_alarm_8	DA8 DT in event text: <field ref="0" /> S5Time, <field ref="1" /> Timer, <field r	<ref id = 0; type = AlarmTag; Tag = PI	ACAT	HMI_TriggerAT	1	<No value	0	<No value	0	<No value	False	
10	11	Discrete_alarm_9	DA11 DT in event text: <field ref="0" /> Int, <field ref="1" /> Real, <field ref="2	<ref id = 0; type = AlarmTag; Tag = In	ACAT	HMI_TriggerInte	0	<No value	0	<No value	0	<No value	False	
11	12	Discrete_alarm_10	DA12 DT in event text: <field ref="0" /> UDint, <field ref="1" /> UInt;	<ref id = 0; type = AlarmTag; Tag = U	ACAT	HMI_TriggerInte	1	<No value	0	<No value	0	<No value	False	
12	13	Discrete_alarm_11	DA13 Textformat: Integer: <field ref="0" /> decimal, <field ref="1" /> binary, <	<ref id = 0; type = AlarmTag; Tag = H	ACAT	HMI_TFAuD	0	<No value	0	<No value	0	<No value	False	

Table 14-3 Meaning of the entries

List entry	Meaning
ID	The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Name	Name of the analog alarm
Event text [de-DE], Alarm text	Displays the alarm text. The field designation contains a language ID. For import, a language ID must be assigned to alarm text. An expression with a reference ID will be added to the alarm text if the text has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign dynamic parameters to alarm texts.
FeldInfo	Specifies whether the alarm text contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = AlarmTag; Tag = Tag1; DisplayType = Decimal; Length = 5;> Text list: <ref id = 1; type = CommonTextList; TextList = Textlist1; Tag = tag 2; Length = 5;>
Class	The class of an alarm determines whether or not the alarm must be acknowledged. It can also be used to determine how the alarm appears when it is displayed on the HMI device. The alarm class also determines whether and where the corresponding alarm will be logged.
Group	Indicates the allocation to an alarm group. If an alarm belongs to a group with other alarms, it can be acknowledged together with these alarms of the same group in a single operation.
Trigger tag	Specifies the tag containing the bit that triggers the alarm.
Trigger bit	Specifies the number of the bit that triggers the alarm.
Acknowledge tag	Specifies the tag containing the bit that is set by the operator upon acknowledgment. Only available if the selected alarm class requires alarm acknowledgment.
Acknowledgment bit	Specifies the number of the bit that is set when the operator acknowledges the alarm.
PLC acknowledgement tag	Specifies the tag containing the bit that acknowledges the alarm of the control program. Only available if the selected alarm class requires alarm acknowledgment.
PLC acknowledgment bit	Specifies the number of the bit that acknowledges the alarm of the control program.
Delay time value	Specifies the delay time. The alarm is not triggered until the duration of the limit value violation equals the specified delay time.
Delay time unit	Specifies the time unit for the delay.

List entry	Meaning
Report	Enables reporting of the specific alarm on a printer. True or "1" = Reporting enabled. False or "0" = Reporting disabled. Reporting must also be globally enabled in the project.
Info text [de-DE], Info text	The tooltip is an optional property of an alarm. Tooltips can contain additional information about the alarm. A tooltip will be displayed in a separate window on the HMI device when the operator presses the <HELP> key. The field designation contains a language ID.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing alarm of the same name.

See also

Exporting alarms (Page 1971)

Importing alarms (Page 1972)

Format of the analog alarm data (Page 1975)

14.3.3.6 Format of the analog alarm data (RT Professional)

Introduction

This chapter describes the required format of the file for the import of analog alarms. The file containing the analog alarm data must be in "*.xlsx" format.

Structure of the alarm data

The import file in Microsoft Excel consists of two worksheets:

- Analog alarms(Analog alarms)
- Limits (Limits)

Each alarm is assigned a separate line in the import file. The import file with the analog alarms must be formatted as follows:

Example of the worksheet "Analog alarms"

ID	Name	Class	Group	Priority	CPU number	PLC number	Trigger tag	Delay value	Delay time unit
1	AnalogAlarm_1	Errors	AlarmGroup_1	1	1	1	Tag_1	5	Second
2	AnalogAlarm_2	Errors	AlarmGroup_2	1	1	1	Tag_2	5	Second
3	AnalogAlarm_3	Warnings		1	1	1	Tag_3	0	
4	AnalogAlarm_4	Warnings		1	1	1	Tag_4	0	

Acknowledgement tag	Acknowledgement bit	Single acknowledgement	Status tag	Status bit	Alarm annunciation	Display suppression mask
Tag_2	0	true	Tag_3	0	false	0x00000001
Tag_2	1	true	Tag_3	1	false	0x00000001
			<No Value>		false	
			<No Value>		false	

Event text [de-DE], Alarm text 1	Additional text 1 [de-DE], Alarm text 2	Additional text 3 [de-DE], Alarm text 4	Info text [de-DE], Info text
Anwendertext 1	Zusatztext 1		Infotext 1
Anwendertext 2		Zusatztext 2	Infotext 2
Anwendertext 3			Infotext 3
Anwendertext 4 <ref domaintype="AnalogAlarmParameterFieldInfo"> <link reference="1" valuetype="ID"/> <format displaytype="" length="5" alignment="Right" precision="0" zeropadding="False"/> </ref>			Infotext 4

Table 14-4 Meaning of the entries

List entry	Meaning
ID	Shows the number of the analog alarm. The alarm number must be unique in the project.
Name	Shows the name of the analog alarm. The name is used to reference an alarm. Alarms with identical names are overwritten during import. An alarm without an existing name is created.
Event text [de-DE], Alarm text	Displays the alarm text. The field designation contains a language ID. Alarm texts must be assigned a language ID for import. An expression with a reference ID will be added to the alarm text if the text has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign dynamic parameters to alarm texts.
FeldInfo	Specifies whether the alarm text contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Parameter: <ref id = 0; type = AlarmAnalogParameter; Parameter = Limit; DisplayType = Decimal; Length = 1; Precision = 0; Alignment = Right; ZeroPadding = False;> System parameters: <ref id = 0; type = CommonSystemParameter; SystemParameter = UserName;>
Class	The alarm class of an alarm determines whether or not the alarm has to be acknowledged. It can also be used to determine how the alarm appears when it is displayed on the HMI device. The alarm class also determines whether and where the corresponding alarm will be logged.

List entry	Meaning
Group	Indicates the allocation to an alarm group. If an alarm belongs to a group with other alarms, it can be acknowledged along with these alarms of the same group in a single operation.
Priority	Indicates alarm priority. You can sort the alarms in the alarm view by priority. Range: 0 -16
CPU number	Specifies the CPU number. This option is only activated if you have enabled the "CPU/PLC number" alarm text block in the "Alarm blocks > System blocks" tab in the "HMI alarms" editor.
PLC number	Shows the PLC number. This option is only activated if you have enabled the "CPU/PLC number" alarm text block in the "Alarm blocks > System blocks" tab in the "HMI alarms" editor.
Trigger tag	Specifies the tag monitored for limit value violation.
Verzögerung	Specifies the delay time. The alarm is not triggered until the duration of the limit value violation equals the specified delay time.
Delay time unit	Specifies the time unit for the delay.
Acknowledge tag	Specifies the tag containing the bit that is set by the operator upon acknowledgment. Only available if the selected alarm class requires alarm acknowledgment.
Acknowledgment bit	Specifies the number of the bit that is set when the operator acknowledges the alarm.
Single acknowledgment	Specifies whether each individual alarm must be acknowledged. True = Alarms must be acknowledged individually.
Status tag	Shows the tag which saves the Incoming and Outgoing alarm statuses and the Acknowledgement request saved.
Status bit	Specifies the number of the bit that is set and acknowledges the alarm.
Alarm annonciation	Indicates whether an alarm is being suppressed. True = suppressed alarm.
Display suppresion mask	Indicates the display suppression method.
Additonal Text	Shows the additional alarm text.
Info text [de-DE], Info text	The tooltip is an optional property of an alarm. Tooltips can contain additional information about the alarm. A tooltip will be displayed in a separate window on the HMI device when the operator presses the <HELP> key. The field designation contains a language ID.

Example of the worksheet "Limits"

Alarm ID	Name	Limit mode	Limit type	Limit value	Suppression	Deadband mode	Deadband in percent	Deadband value
1	AnalogAlarmLimit_1	Upper limit	Constant	100	false	On above thresold	false	105
1	AnalogAlarmLimit_2	Upper limit	Constant	110	false	On above thresold	false	115
2	AnalogAlarmLimit_1	Upper limit	Constant	100	false	On above thresold	false	105
2	AnalogAlarmLimit_2	Upper limit	Constant	110	false	On above thresold	false	115
3	AnalogAlarmLimit_1	Lower limit	Tag	LimitTag_1	false	Off		
4	AnalogAlarmLimit_1	Lower limit	Tag	LimitTag_2	false	Off		

Table 14-5 Meaning of the entries

List entry	Meaning
Alarm ID	The alarm number is used to reference an alarm. The alarm number is unique. The alarm number allows alarms to be assigned limits.
Name	Shows the name of the analog alarm.
Limit mode	Indicates the method used for monitoring the limit value.
Grenzen Limit type	Specifies the limit that will be monitored. Both a tag and a constant can be used as limit value.
Limit value	Indicates the tag or constant monitored for limit violation.
Suppression	Indicates whether an alarm is being suppressed. True = suppressed alarm.
Deadband mode	Specifies whether and in which cases hysteresis will be used. For "Outgoing" For "Incoming" For "Incoming" and "Outgoing"
Deadband in percent	0 = The value specified for "Hysteresis" is interpreted as absolute. 1 = The value specified for "Hysteresis" is a percentage value of the limit value.
Deadband mode	Specifies a constant as value of the hysteresis.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing alarm of the same name.

See also

Exporting alarms (Page 1971)

Importing alarms (Page 1972)

14.3.3.7 Format of the discrete alarm data (RT Professional)

Introduction

This chapter describes the required format of the file for the import of discrete alarms. The file containing the discrete alarm data must be in "*.xlsx" format.

Structure of the alarm data

The import file in Microsoft Excel consists of the worksheet "Discrete alarms" (discrete alarms). Each alarm is assigned a separate line in the import file. Structure of the import file containing the discrete alarms:

Example of the worksheet "Discrete alarms"

ID	Name	Class	Group	Priority	CPU number	PLC number	Trigger tag	Trigger bit	Trigger mode	Alarm parameter 1	Alarm par
1	DiscreteAlarm_1	Errors	AlarmGroup_1	1			Tag_1	0	On rising edge	Tag_1	<No Value
2	DiscreteAlarm_2	Errors	AlarmGroup_2	1			Tag_1	0	On rising edge	Tag_1	<No Value
3	DiscreteAlarm_3	Warning		1			Tag_1	0	On falling edge	<No Value>	<No Value
4	DiscreteAlarm_4	Warning		1			Tag_1	0	On falling edge	<No Value>	<No Value

Alarm parameter 4	Alarm parameter 5	Alarm parameter 6	Alarm parameter 7	Alarm parameter 8	Alarm parameter 9	Alarm parameter 10
<No Value>	<No Value>	<No Value>	<No Value>	<No Value>	<No Value>	<No Value>
<No Value>	<No Value>	<No Value>	<No Value>	<No Value>	<No Value>	<No Value>
<No Value>	<No Value>	<No Value>	<No Value>	<No Value>	<No Value>	<No Value>
<No Value>	<No Value>	<No Value>	<No Value>	<No Value>	<No Value>	<No Value>

Acknowledgement tag	Acknowledgement bit	Single acknowledgement	Status tag	Status bit	Alarm annunciation	Display suppression mask
Tag_2	0	true	Tag_3	0	false	0x00000000
Tag_2	1	true	Tag_3	1	false	0x00000000
			<No Value>		false	
			<No Value>		false	

Event text [de-DE], Alarm text 1	Additional text 1 [de-DE], Alarm text 2	Additional text 3 [de-DE], Alarm text 4	Info text [d
Anwendertext 1	Zusatztext 1		Infotext 1
Anwendertext 2		Zusatztext 2	Infotext 2
Anwendertext 3			Infotext 3
Anwendertext 4 <ref domaintype="ParameterFieldInfo"><link reference="MyGroup.TagTable1.Tag_1" valuetype="ID"/> <format displaytype="Decimal" length="5" alignment="Right" precision="2" zeropadding="0"/> </ref>			Infotext 4

Table 14-6 Meaning of the entries

List entry	Meaning
ID	Shows the number of the discrete alarm. The alarm number is unique.
Name	Shows the name of the discrete alarm. The name is used to reference an alarm. Alarms with identical names are overwritten during import. An alarm without an existing name is created.
Event text [de-DE], Alarm text	Displays the alarm text. The field designation contains a language ID. Alarm texts must be assigned a language ID for import. An expression with a reference ID will be added to the alarm text if the text has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign dynamic parameters to alarm texts.

List entry	Meaning
FeldInfo	Specifies whether the alarm text contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Parameter: <ref id = 0; type = AlarmDiscreteParameter; Parameter = Parameter 1; Tag = HMI_Variable_1; DisplayType = Decimal; Length = 5; Precision = 0; Alignment = Right; ZeroPadding = False;> System parameters: <ref id = 0; type = CommonSystemParameter; System Parameter = UserName;>
Class	The alarm class of an alarm determines whether or not the alarm has to be acknowledged. It can also be used to determine how the alarm appears when it is displayed on the HMI device. The alarm class also determines whether and where the corresponding alarm will be logged.
Group	Indicates the allocation to an alarm group. If an alarm belongs to a group with other alarms, it can be acknowledged along with these alarms of the same group in a single operation.
Priority	Indicates alarm priority. You can sort the alarms in the alarm view by priority. Range: 0 -16
CPU number	Specifies the CPU number. This option is only activated if you have enabled the "CPU/PLC number" alarm text block in the "Alarm blocks > System blocks" tab in the "HMI alarms" editor.
PLC number	Shows the PLC number. This option is only activated if you have enabled the "CPU/PLC number" alarm text block in the "Alarm blocks > System blocks" tab in the "HMI alarms" editor.
Trigger tag	Specifies the tag containing the bit that triggers the alarm.
Trigger bit	Specifies the number of the bit that triggers the alarm.
Trigger mode	Indicates the method used for monitoring the limit value.
Alarm parameter	Indicates tag assigned to a parameter block to issue process values in alarm texts.
Acknowledge tag	Specifies the tag containing the bit that is set by the operator upon acknowledgment. Only available if the selected alarm class requires alarm acknowledgment.
Acknowledgment bit	Specifies the number of the bit that is set when the operator acknowledges the alarm.
Single acknowledgment	Specifies whether each individual alarm must be acknowledged. True = Alarms must be acknowledged individually.
Status tag	Shows the tag which saves the Incoming and Outgoing alarm statuses and the Acknowledgement request saved.
Status bit	Specifies the number of the bit that is set and acknowledges the alarm.
Alarm annonciation	Indicates whether an alarm is being suppressed. True = suppressed alarm.
Display suppresion mask	Indicates the display suppression method.
Additonal Text	Shows the additional alarm text.
Info text [de-DE], Info text	The tooltip is an optional property of an alarm. Tooltips can contain additional information about the alarm. A tooltip will be displayed in a separate window on the HMI device when the operator presses the <HELP> key. The field designation contains a language ID.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing alarm of the same name.

See also

Exporting alarms (Page 1971)

Importing alarms (Page 1972)

14.3.4 Importing and exporting tags

14.3.4.1 Exporting tags (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

WinCC has an export function for tags.

Requirements

- The WinCC project for the export is open.
- Tags have been created in the project.
- The "HMI tags" editor is open.

Exporting tags

1. Click on the button in the "HMI Tags" tab.
   The "Export" dialog box opens.

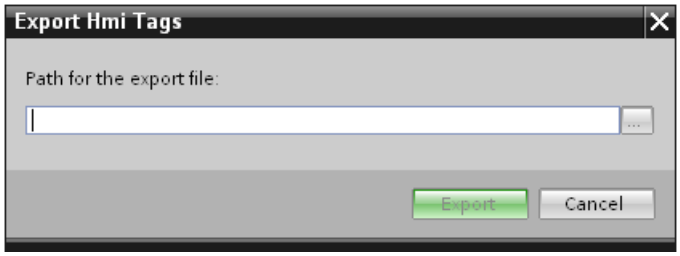

2. Click the "..." button and specify in which file the data are saved.
3. Click "Export". The export will start.

Note

The version number of the xlsx file with the exported tags depends on the TIA Portal version. If the xlsx file was exported from a project in WinCC V13 SP1, it has the version number 1.2. If the xlsx file was exported from a project prior to WinCC V13 SP1, it has the version number 1.1.

Note

It is not possible to export HMI tags of the data type "UDT" which contain structured elements via Excel for subsequent editing.

After export, only the higher-level HMI tag appears in Excel. Its lower-level elements cannot be edited.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

14.3.4.2 Importing tags (WinCC V13 SP1 or higher) (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Tags are identified by the tag name. An existing tag is overwritten with the data from the xlsx file if the tag name already exists in the project. A new tag is created if the tag does not yet exist.

Requirements

- An xlsx file with tags has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.

Importing tags

1. Click "HMI tags" in the project navigation.
2. Double-click "Show all tags". The "HMI tags" editor opens.
3. Click the button. The "Import" dialog box opens.

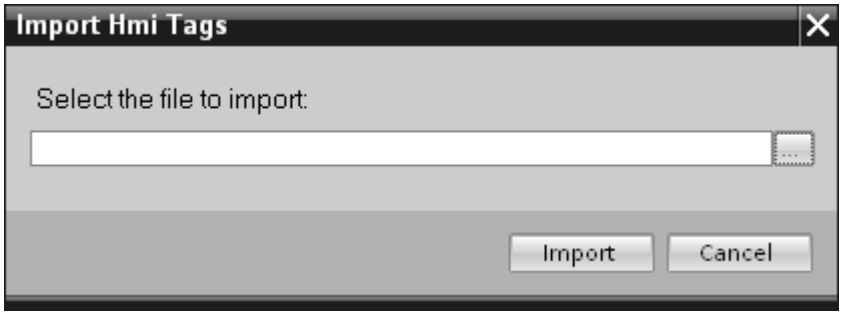

4. Click the "..." button and select the file that you want to import.
5. Click on the "Import" button. The import will start.

Result

The relevant tags have been created in WinCC. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign a tag a trigger tag of the wrong type, for example, string. An error will be reported during compilation.

14.3.4.3 Importing tags (prior to WinCC V13 SP1)

Introduction

Tags are identified by the tag name. An existing tag is overwritten with the data from the xlsx file if the tag name already exists in the project. A new tag is created if the tag does not yet exist.

The version number of the xlsx file depends on the TIA Portal version. If the xlsx file was exported from a project in WinCC V13 SP1, it has the version number 1.2. If the xlsx file was exported from a project prior to WinCC V13 SP1, it has the version number 1.1.

To import files with the version number 1.2 into a project that was created prior to WinCC V13 SP1, you must prepare the files.

Requirements

- An xlsx file with tags has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.

Preparing files for the import

1. Open the corresponding file in Microsoft Excel.
2. Select "File > Properties > Advanced properties".
   The "Properties" dialog is opened.
3. Switch to the "Fit to size" task card.
4. Select "TIA_Version" in the "Properties" area.
5. Change the value in the column from "1.2" to "1.1".
6. Click the "Change" button.
7. Click "OK".
8. Delete the right column "Synchronization" in the file.
9. Save the file.

Importing tags

1. Click "HMI tags" in the project navigation.
2. Double-click "Show all tags". The "HMI tags" editor opens.
3. Click the button. The "Import" dialog box opens.
4. Click the "..." button and select the file that you want to import.
5. Click on the "Import" button. The import will start.

Result

The relevant tags have been created in WinCC. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign a tag a trigger tag of the wrong type, for example, string. An error will be reported during compilation.

14.3.4.4 Format of the tag data (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

This section describes the format required for the file with tag data used for imports. The tag data file must be in "*.xlsx" format.

Tag data structure

The import file in Microsoft Excel consists of a number of worksheets:

- HMI Tags (HMI tags)
- Multiplexing (multiplex tags)

Each tag is on a separate line in the import file. The import file with the tag data must have the following format:

Example of the worksheet "HMI Tags"

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Name	Path	Connection	PLC tag	DataType	Length	Address	Access Meth	Indirect a	Index tag	Start value	ID tag	Coding	Display na	Comment	Acquisitio	Acqui	High High
2	HMI_Int	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
3	Mux_Tag_1	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	True	Mux_Tag_	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
4	Mux_Tag_2	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	True	Mux_Tag_	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
5	Mux_11	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
6	Mux_21	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
7	Mux_13	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
8	Mux_12	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
9	Mux_23	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
10	Mux_22	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
11	Mux_Tag_1_Index	Internal Tags	<No Value>	<No Value>	UInt	2	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
12	Mux_Tag_12	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value>	True	Mux_Tag_	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
13	Mux_Tag_11	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value>	True	Mux_Tag_	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
14	Mux_Tag_13	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value>	True	Mux_Tag_	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
15	HMI_UDInt	Internal Tags	<No Value>	<No Value>	UDInt	4	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
16	Gauge_Process	Default tag table	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
17	Button_Tag_4	Default tag table	<No Value>	<No Value>	Int	2	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
18	HMI_USInt	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value>	False	<No Value	<No Value>	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit
19	Data_block_2_PLC_DateTime_2	Default tag table	HMI_connection_1	Data_block_2.PLC_I	Date_And_Time	8	%DB28.DBX598.(	<absolute ac	False	<No Value	01.01.1999 12:00	0	Binary	<No Value	<No Value	Cyclic in o	1 s	NoLimit

Table 14-7 Meaning of the entries

List entry	Meaning
Name	Indicates the configured name of an HMI tag.
Path	Specifies which folders in the project tree contain the tag. The folder structure is represented by "\" : "FolderName1\FolderName2\TagName".
PLC Tag	Indicates whether the tag is linked to a PLC tag.
Connection	Indicates the name of the connection to the PLC.
Data type	Specifies the data type of a tag. The data types allowed depend on the communication driver being used. See the "Communication" section of the documentation for additional information on the data types permitted for the various communication drivers.
Length	Specifies the length of the tag. This entry is only useful for data types with a dynamic length such as strings; it is left empty for all other data types.
Address	Specifies the tag address in the PLC. The tag address must exactly match the one used in WinCC, for example, "%DB1.DBW0". The tag address is empty for internal tags.
Multiplexing	Specifies whether multiplexing is used.
Index tag	Shows the name of the index tag for multiplexing. In Runtime, the system first reads the value of the index tag. It then accesses the tag in the corresponding place in the tag list.
StartValue	Specifies the start value of a tag.
ID tag	The update ID updates the value of a tag with the aid of a function or a PLC job. The update ID must be unique within an HMI device.
Coding	Shows the coding method.
DiplayName [de_DE]	Shows the display name of an HMI tag. The field designation contains a language ID. The field designation contains a language ID. Display names must be assigned a language ID for import. Texts are imported to the corresponding project language.
Acquisition mode	Specifies the tag acquisition mode.

List entry	Meaning
Acquisition cycle	Specifies the tag acquisition cycle. The acquisition cycle must correspond exactly to the one used in WinCC. The value is not language-dependent and should therefore be the same in every language. The default value is "1 s". The acquisition cycle is undefined if the tag acquisition mode is "on demand". User-defined acquisition cycles must be created beforehand as the file will otherwise not be imported.
High High Limit type	Indicates whether the limit value "High high" is monitored by a constant, a tag or not at all.
High High Limit	Displays the limit value "High High".
High Limit type	Indicates whether the limit value "High" is monitored by a constant, a tag or not at all.
High Limit	Displays the limit value "High".
Low Limit type	Indicates whether the limit value "Low" is monitored by a constant, a tag or not at all.
Low Limit	Displays the limit value "Low".
Low Low Limit type	Indicates whether the limit value "Low Low" is monitored by a constant, a tag or not at all.
Low Low Limit	Displays the limit value "Low Low".
Linear scaling	Indicates whether linear scaling is enabled. This entry can only be used for external tags.
End value PLC	Specifies the end value of the PLC tag.
Start value PLC	Specifies the start value of the PLC tag.
End value HMI	Specifies the end value of the HMI tag.
Start value HMI	Specifies the start value of the HMI tag.

Example of the worksheet "Multiplexing"

	A	B	C
1	HMI Tag name	Multiplex Tag	Index
2	Mux_Tag_1	Mux_11	0
3	Mux_Tag_1	Mux_12	1
4	Mux_Tag_1	Mux_13	2
5	Mux_Tag_2	Mux_21	0
6	Mux_Tag_2	Mux_22	1
7	Mux_Tag_2	Mux_23	2
8	Mux_Tag_12	HMI_Array_Mux2	-1
9	Mux_Tag_11	HMI_Array_Mux1	-1
10	Mux_Tag_13	HMI_Array_Mux3	-1

Table 14-8 Meaning of the entries

List entry	Meaning
Name	Indicates the configured name of an HMI tag which uses indirect addressing. The HMI tag must be available in the "HMI Tags" worksheet.
Index	Shows the value which governs which tag is selected.
Multiplex Tag	Displays the tag from the tag list corresponding to the index value.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing tag of the same name.

14.3.4.5 Format of the tag data (RT Professional)

Introduction

This section describes the format required for the file with tag data used for imports. The tag data file must be in "*.xlsx" format.

Tag data structure

The import file in Microsoft Excel consists of a number of worksheets:

- HMI Tags (tags)
- SubstituteValueUsage (substitute value)

Each tag is on a separate line in the import file. The import file with the tag data must have the following format:

Example of the worksheet "HMI Tags"

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Name	Path	Connection	PLC tag	DataType	HMI DataTyp	Length	Address	Access Me	Start valu	Quality Co	Persisten	Substitute	Tag value	Update M	Coding	Comment	High High
2	Tag_1	Default table	HMI_Connection_1	Data_block_1.Static_1	Real	Float	4	%DB1.DBD	<No Value	<No Value	True	False	25	<No Value	Client/Ser	Binary	<No Value	NoLimit
3	Tag_2	Default table	<No Value>	<No Value>	Int	Int	2	<No Value>	<No Value	<No Value	False	False	<No Value	<No Value	Client/Ser	Binary	<No Value	NoLimit

Table 14-9 Meaning of the entries

List entry	Meaning
Name	Indicates the configured name of an HMI tag.
Path	Specifies which folders in the project tree contain the tag. The folder structure is represented by ",": "FolderName1,FolderName2,TagName".
PLC Tag	Specifies which PLC tag is linked to the HMI tag.
Connection	Indicates the name of the connection.
Date type	Specifies the data type of the PLC tag. The data types allowed depend on the communication driver being used. See the "Communication" section of the documentation for additional information on the data types permitted for the various communication drivers.

List entry	Meaning
HMI Data type	Specifies the data type of the HMI tag. The data types allowed depend on the communication driver being used. See the "Communication" section of the documentation for additional information on the data types permitted for the various communication drivers.
Length	Specifies the length of the tag in bytes. This entry is only useful for data types with a dynamic length such as strings; it is left empty for all other data types.
Address	Specifies the tag address in the PLC. The tag address must exactly match the one used in WinCC, for example, "%DB1.DBW0". The tag address is empty for internal tags.
Start Value	Specifies the start value of a tag.
Substitute Value	Indicates the substitute value. The substitute value is used when a process value with errors is read.
ID tag	The update ID updates the value of a tag with the aid of a function or a PLC job within a device. The update ID must be unique.
High High Limit type	Indicates whether the limit value "High high" is monitored by a constant or not at all.
High High Limit	Displays the limit value "High High".
High Limit type	Indicates whether the limit value "High" is monitored by a constant or not at all.
High Limit	Displays the limit value "High".
Low Limit type	Indicates whether the limit value "Low" is monitored by a constant or not at all.
Low Limit	Displays the limit value "Low".
Low Low Limit type	Indicates whether the limit value "Low Low" is monitored by a constant or not at all.
Low Low Limit	Displays the limit value "Low Low".
Linear scaling	Indicates whether linear scaling is enabled. This entry can only be used for external tags.
End value PLC	Specifies the end value of the PLC tag.
Start value PLC	Specifies the start value of the PLC tag.
End value HMI	Specifies the end value of the HMI tag.
Start value HMI	Specifies the start value of the HMI tag.

Example of the worksheet "SubstituteValueUsage"

	A	B
1	HMI Tag name	Substitute Value Usage
2	Tag_1	StartValue
3	Tag_1	ConnectionError

Table 14-10 Meaning of the entries

List entry	Meaning
Parent	Specifies the tag for which a substitute value has been defined. The tag must be available in the "HMI-Tags" worksheet.
Substitute Value Usage	Indicates the substitute value. The substitute value can be used in the following situations: • As start value • After communication error • High High limit value • Low Low limit value

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing tag of the same name.

14.3.5 Importing and exporting text lists

14.3.5.1 Exporting text lists (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

WinCC has an export function for text lists.

Requirements

- The WinCC project for the export is open.
- Text lists have been created in the project.
- The "Text & graphics lists" editor is open.

Exporting text lists

1. Click on the button in the "TextLists" tab.
   The "Export" dialog box opens.

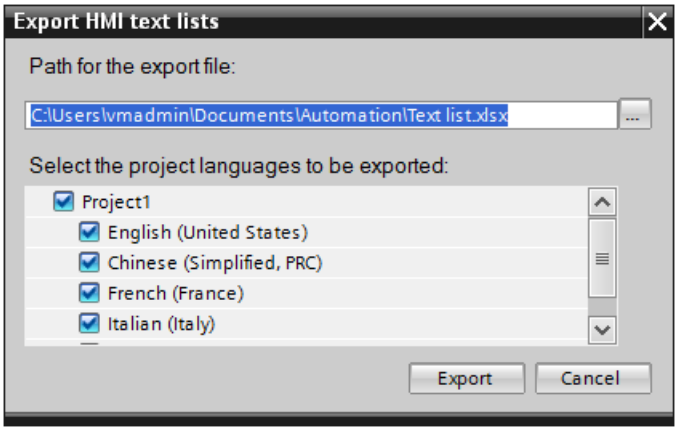

2. Click the "..." button and specify in which file the data are saved.
3. By default, texts are exported in all defined project languages.
   If you do not want to export specific languages, disable the unnecessary languages in the dialog.
4. Click "Export". The export will start.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

See also

Importing text lists (Page 1996)

Format of text list data (Page 1997)

14.3.5.2 Importing text lists (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You then import text lists from an xlsx file to WinCC.

Requirements

- An xlsx file with text lists has been created.
- The structure of the xlsx file meets the requirements.

- The WinCC project for the import is open.
- The "Text & graphics lists" editor is open.

Importing text lists

1. Click on the button in the "Text lists" tab.
   The "Import" dialog box opens.

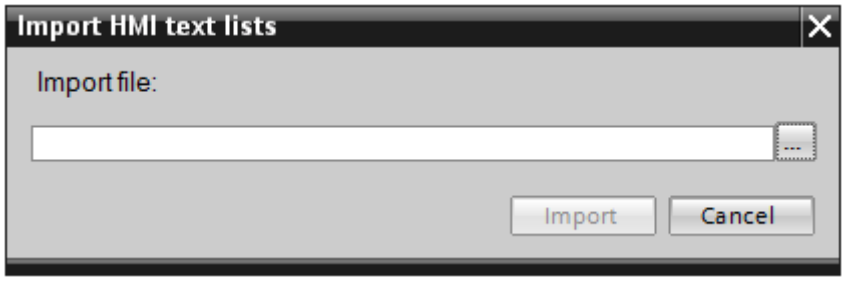

2. Select the file you want to import under "File selection".
3. Click on the "Import" button. The import will start.

Result

You have now imported the text lists. The relevant text lists have been created in WinCC. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

See also

Exporting text lists (Page 1995)

14.3.5.3 Format of text list data (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

This section describes the format required for the file with the text lists used for imports. The text list data file must be in "*.xlsx" format.

Tag data structure

The import file in Microsoft Excel consists of two worksheets:

- TextList (Text lists)
- TextListEntry (Text list entry)

Each text list is assigned a separate line in the import file. The import file containing the data must be structured as follows:

Example of the worksheet "TextList"

	A	B	C
1	Name	ListRange	Comment
2	TLValue/Range	Decimal	
3	TLBit	Bit	
4	TLBitnumber	Binary	
5	TLall_1	Decimal	
6	TLall_2	Decimal	

Table 14-11 Meaning of the entries

List entry	Meaning
Name	Shows the name of the text list.
ListRange	Shows the text list range: Number = Bit number (0-31) Range = value/range (...-...) Bit = Bit (0;1)
Comment	Any comments on the text list. You can use up to 500 characters

Example of the worksheet "TextListEntry"

	A	B	C	D	E	F
1	Name	Parent	DefaultEntry	Value	Text	FieldInfos
2	Text_list_entry_1	TLValue/Range	TRUE	0 - 1	Default entry TLValue/Range ->	
3	Text_list_entry_2	TLValue/Range		2 - 3	TLValue/Range = 2-3 ->	
4	Text_list_entry_3	TLValue/Range		1	TLValue Range - single value = 1 ->	
5	Text_list_entry_1	TLBit		0	TLBit = 0 ->	
6	Text_list_entry_2	TLBit		1	TLBit = 1 ->	
7	Text_list_entry_1	TLBitnumber	TRUE	0	Default entry TLBitnumber; ->	
8	Text_list_entry_2	TLBitnumber		0	TLBitnumber - Bitnumber 0 is set ->	
9	Text_list_entry_3	TLBitnumber		1	TLBitnumber - Bitnumber 1 is set ->	
10	Text_list_entry_4	TLBitnumber		2	TLBitnumber - Bitnumber 2 is set ->	
11	Text_list_entry_5	TLBitnumber		3	TLBitnumber - Bitnumber 3 is set ->	
12	Text_list_entry_1	TL1	TRUE	0 - 1	Default entry TL1	
13	Text_list_entry_2	TL1		1 - 3	TL1 Value between 1 - 3 ->	
14	Text_list_entry_3	TL1		4 - 6	TL1 Value between 4 - 6 ->	
15	Text_list_entry_4	TL1		7	TL1 Single value = 7 ->	
16	Text_list_entry_1	TL2	TRUE	0 - 1	<field ref="0" /> ->	<ref id = 0; type = CommonTextList; TextList = TL1; Tag = HMI_TL1control; Length = 100;>
17	Text_list_entry_2	TL2		1	TL2 Single value = 1 ->	
18	Text_list_entry_3	TL2		2 - 3	TL2 Range between 2 - 3 ->	
19	Text_list_entry_1	TLMultilined	TRUE	0 - 1	Default entry TLMultilined; last row	
20	Text_list_entry_2	TLMultilined		0 - 3	TLMultilined Value between 0-3\nwith test of"\n"	
21	Text_list_entry_1	TLall_1	TRUE	0 - 1	Default entry TLall_1	
22	Text_list_entry_1	TLall_2	TRUE	0 - 1	Default entry TLall_2	

Table 14-12 Meaning of the entries

List entry	Meaning
Name	Shows the name of the text list entry.
Parent	Specifies the name of the corresponding text list.
DefaultEntry	Indicates whether the text list entry is a default entry. The default entry is always displayed when the tag has an undefined value.
Value	Specifies the tag integer values or value ranges which are assigned to the text entries in the text list.
Text	Shows the text list entry. The field designation contains a language ID. Text list entries must be assigned a language ID for import. An expression with a reference ID will be added to the text if the text list entry has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign the dynamic parameter to a text list entry.
FeldInfo	Specifies whether the text list contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = CommonTagDisplayFormat; Tag = tag 1; DisplayType = Decimal; DisplayFormat = 9;> Text list: <ref id = 1; type = CommonTextList; TextList = Textliste_1; Tag = tag 2; Length = 5;> PLC tag: <ref id = 0; type = CommonControlTagDisplayFormat; DisplayType = Decimal; DisplayFormat = 9;>

See also

Exporting text lists (Page 1995)

14.3.6 Importing and exporting project texts

14.3.6.1 Importing project texts

Edit the xlsx file or send it to a translator. Import the texts once they have been translated. The foreign languages will be imported to the relevant object in the project.

Note

In WinCC, you only import the previously exported project texts into the same project. Importing into a different project is not supported.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Importing project texts

To import a project text file, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The lower-level elements will be displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Click on the button. The "Import" dialog opens.
4. Select the path and file name of the import file from the "Import file" field.
5. Activate the "Import source language" check box if you have made changes to the source language in the export file and would like to overwrite the entries in the project with the changes.
6. Click on "Import".

Result

You have imported the project texts.

14.3.6.2 Exporting project texts

Project texts are exported for translation. Texts are exported to Office Open XML files ending in ".xlsx". These files can be edited in Microsoft Excel, for example.

You can exchange the file with the translators and import it back to the project as soon as it has been translated.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Exporting project texts

To export individual project texts, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree. The child elements are displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Select the texts you want to export.
4. Click on the button. The "Export" dialog opens.

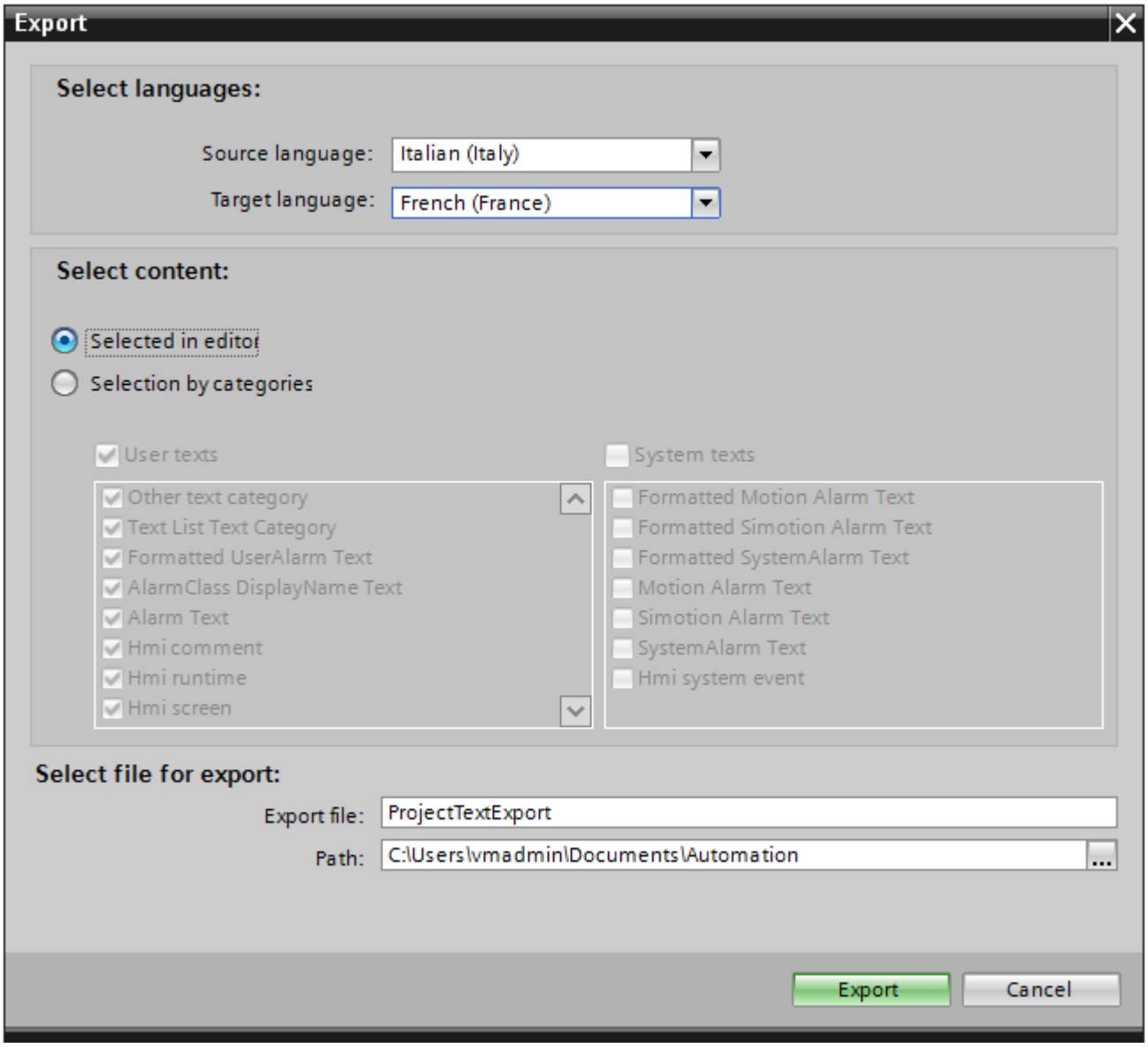

5. From the "Source language" drop-down list, select the language from which you wish to translate, for example Italian.
6. From the "Target language" drop-down list, select the language into which the texts are to be translated, for example, French.
7. Enter a file path and a file name for the export file in the "Export file" input field.
8. Click "Export".

Result

The texts selected in the "Project texts" editor are written to an xlsx file. The xlsx file will be stored in the specified folder.

You can alternatively select and export all project texts from categories. Select "User texts" or "System texts" in the "Export" dialog in line with the type of texts you wish to export. In this case, export can additionally be limited by categories.

Note

Project texts in library objects cannot be exported.

14.4 Using cross-references

14.4.1 General information about cross references (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The cross-reference list provides an overview of the use of objects and devices within the project and project library. The cross-reference list can be used to show the relationships and the dependencies of objects among themselves.

Uses of cross-reference list

The cross-reference list offers you the following advantages:

- When creating and changing a program, you retain an overview of the devices, objects, tags, alarms, scripts, etc. that you have used.
- From the cross-references, you can jump directly to the object location of use.
- You can see whether the object uses other objects or is used by itself.
- You can learn the following when debugging:
  - Which objects are used in which screen and on which devices.
  - The alarms and recipes shown in a specific display.
  - The tags used in a specific alarm or object.
- As part of the project documentation, the cross-references provide a comprehensive overview of all object, alarms, recipes, tags and screens.

For additional information on working with cross-reference lists, please refer to "Displaying cross-reference lists".

See also

HMI device wizard basics (Page 1929)

14.4.2 Special features when working with cross-reference lists in WinCC (RT Professional, RT Advanced, Panels, Comfort Panels)

Working with cross-references in WinCC

In WinCC, there are a number of special features that you must take into account when working with cross-references.

For additional information on working with cross-reference lists, please refer to "Displaying cross-reference lists".

"Change cross-reference" dialog

You can change the references of tags in the properties of screen objects using the "Change object references" dialog.

Using textual cross-references when working with VB scripts

When working with VB scripts, you can reference tags directly with the help of textual cross-references without having to access the SmartTags list.

Adding a new object to the cross-reference list

The auto-complete function only offers HMI tags for selection. Array elements of an HMI tag are not displayed in auto-complete. You can only add array elements by entering the name of an array element.

Displaying cross-references in the Inspector window

If you select an object in the screen and an HMI tag is being used as the process tag at the object, the object and the linked HMI tag are displayed in the cross-references. All locations of use of the object and the HMI tags are also listed.

If the HMI tag is interconnected with a PLC tag or a DB tag, the locations of use of the interconnected PLC tag or DB tag are displayed.

Shortcut menu in the cross-reference list

The shortcut menu is object-oriented. The shortcut menu is available for each row when the "Object" column actually contains an object.

14.4.3 Textual cross-references (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

Textual cross-references are cross-references that are stored as pure text. You cannot navigate to a different object with the help of textual cross-references. Such cross-references are displayed in red in the cross-reference list.

Textual cross-references are created as follows:

- You create a textual cross-reference for a non-existent object manually by entering the object name under Property.
- The object used has already been deleted, and the connection to it is shown as textual cross-reference.

Basics

When references and subordinate objects of a source object are deleted, they are converted into textual cross-references. A textual cross-reference is not stored when the source object itself is deleted.

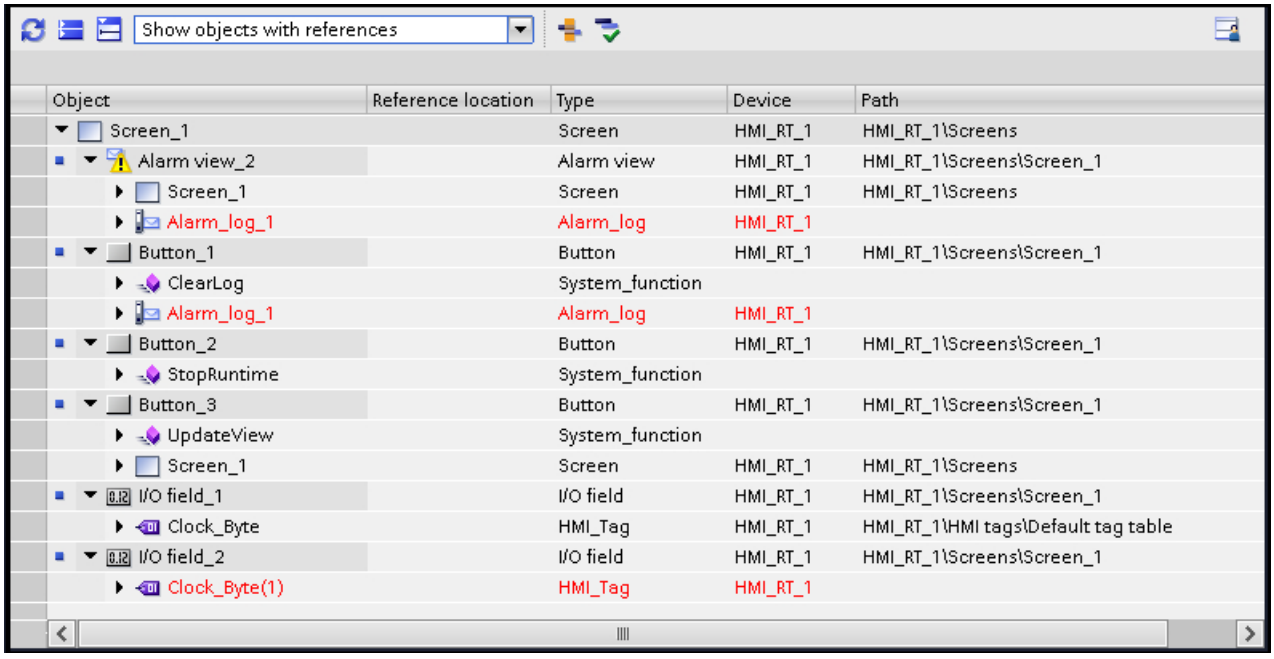

Object	Reference location	Type	Device	Path
Screen_1		Screen	HMI_RT_1	HMI_RT_1\Screens
Alarm view_2		Alarm view	HMI_RT_1	HMI_RT_1\Screens\Screen_1
Screen_1		Screen	HMI_RT_1	HMI_RT_1\Screens
Alarm_log_1		Alarm_log	HMI_RT_1	
Button_1		Button	HMI_RT_1	HMI_RT_1\Screens\Screen_1
ClearLog		System_function		
Alarm_log_1		Alarm_log	HMI_RT_1	
Button_2		Button	HMI_RT_1	HMI_RT_1\Screens\Screen_1
StopRuntime		System_function		
Button_3		Button	HMI_RT_1	HMI_RT_1\Screens\Screen_1
UpdateView		System_function		
Screen_1		Screen	HMI_RT_1	HMI_RT_1\Screens
I/O field_1		I/O field	HMI_RT_1	HMI_RT_1\Screens\Screen_1
Clock_Byte		HMI_Tag	HMI_RT_1	HMI_RT_1\HMI tags\Default tag table
I/O field_2		I/O field	HMI_RT_1	HMI_RT_1\Screens\Screen_1
Clock_Byte(1)		HMI_Tag	HMI_RT_1	

Note

Almost all objects, with the exception of data logs, support textual cross-references.

Note

Textual cross-references cannot be deleted.

You can also create a textual cross-reference manually to establish a connection to a non-existent object. This procedure can be helpful if you want to access tags directly in VB scripts without SmartTags.

Note

Textual cross-references for tags in RT Professional

If no user data type (UDT) is assigned to a screen or the tags are referenced as user data type element with @NOP, @NOTP, @NOSP, textual cross-references appear for the tags.

Using textual cross-references when working with VB scripts

When working with VB scripts, you can reference tags directly with the help of textual cross-references without having to access the SmartTags list.

When you enter an object name, the script parser looks for the referenced object.

For example: `DIM x`

```
x = MyObject
```

A reference is made to any VB script or tag that already exists under this name. If a VB script and a tag exist under this name, the script parser establishes the reference to the VB script.

If neither a tag nor a VB script exists under this name, the system creates two textual cross-references: one for the tag and one for the VB script.

The first object that is created under this name (a tag or a VB script) is assigned one of the two textual cross-references. The textual cross-reference turns into an actual cross-reference. The second textual cross-reference is deleted.

Note

In RT Professional, only one textual cross-reference is created for the VB script, because direct addressing of tags is not possible in this context.

See also

Invalid cross-references (Page 2006)

14.4.4 Invalid cross-references (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

A cross-reference can become invalid as a result of certain changes to the configuration. Invalid cross-references are shown grayed out in the cross-reference list and in the Inspector window. A question mark is displayed beside the cross-reference text. Clicking on the question mark opens the online help.

Invalid cross-references do not have any negative implications for your project.

Basics

A cross-reference can become invalid for the following reasons:

- A change of device type
- Copying a cross-reference from another HMI device to an HMI device that does not support the object in question or object property.
- A change to an object property or the assignment of invalid values

The following objects can be invalid in the cross-reference list:

- Source and target object
- Subordinate objects and their higher-level objects
- Location of use

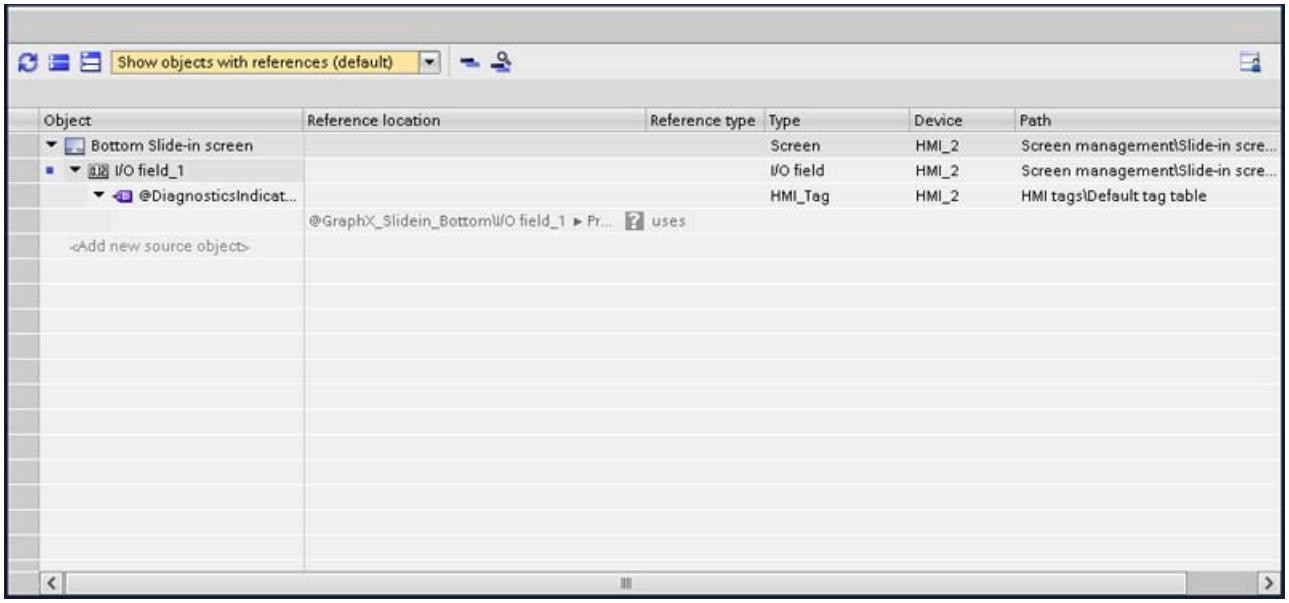

If an object or location of use is invalid, you cannot navigate to that object or location of use. You cannot change the properties of an invalid location of use. For invalid objects, you can access the commands "Delete" and "Properties" over the shortcut menu. No other shortcut menu commands are available. No shortcut menu is available for invalid locations of use.

Note

Textual cross-reference invalid

An object can be an invalid object and a textual cross-reference in the cross-reference list at the same time. In such a case, the invalid cross-reference is shown in red like a textual cross-reference.

Examples

Source object invalid

You have created a screen for a Comfort Panel and added a VB scripts reference to the function list. You change the device type from "SIMATIC Comfort Panel" to "SIMATIC Basic Panel". As VB scripts are not available for Basic Panels, the cross-reference for the VB script is invalid.

Target object invalid

For a Comfort Panel, you have created a screen that has a cross-reference to the audit trail log with the system function "ClearLog". You copy this screen to Runtime Professional. As the audit trail log is not available for Runtime Professional, the cross-reference is invalid.

Property that links the source object to the target object is invalid

In Runtime Professional, you have created an entry in the graphics list that has a cross-reference to the "Flashing" property. You copy this graphics list to a Comfort Panel. As this property is not available on Control Panels, the cross-reference from the graphics list to the "Flashing" propery is invalid.

Property invalid if source object invalid

For a Comfort Panel, you have saved a slide-in screen that has a cross-reference to a tag over an I/O field. You change the device type from "SIMATIC Comfort Panel" to "SIMATIC Basic Panel". As the slide-in screen is not available on Basic Panels, the cross-reference is invalid.

Textual cross-reference invalid

You have created a tag for a Comfort Panel and linked the VB function "SetTag" to that tag. You have also created a screen that is connected to the "SetTag" VB function over the "Established" event. You delete the VB function. A textual cross-reference is created for the deleted VB function. You change the device type from "SIMATIC Comfort Panel" to "SIMATIC Basic Panel". As VB scripts are not available for Basic Panels, the textual cross-reference for the VB script is invalid. The invalid cross-reference is shown in red.

See also

Textual cross-references (Page 2004)

14.4.5 Rewiring tags in the screens (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can change the references of tags in the properties of screen objects using the "Change object references" dialog.

This function lets you to replace a large number of tags in various screen objects. Replacing references is useful, for example, if you have created a screen by copying and pasting and need to adapt its references.

You search for and replace individual or multiple characters in the tag names.

You can open the "Change object references" dialog with the command in the shortcut menu. You can open the dialog for one or more screen objects or a screen.

The "Change object references" dialog is available for the following screens.

- Overview screen (permanent area)
- Templates
- Pop-up screens
- Slide-in screens
- Global screen

How to replace a tag

1. Select an object in the screen that contains a tag that you want to change.
2. Select the "Change object references" command in the shortcut menu or in the "Edit" menu. The "Change object references" dialog opens. You can see the selected screen objects and the tags used in the "Object" column.

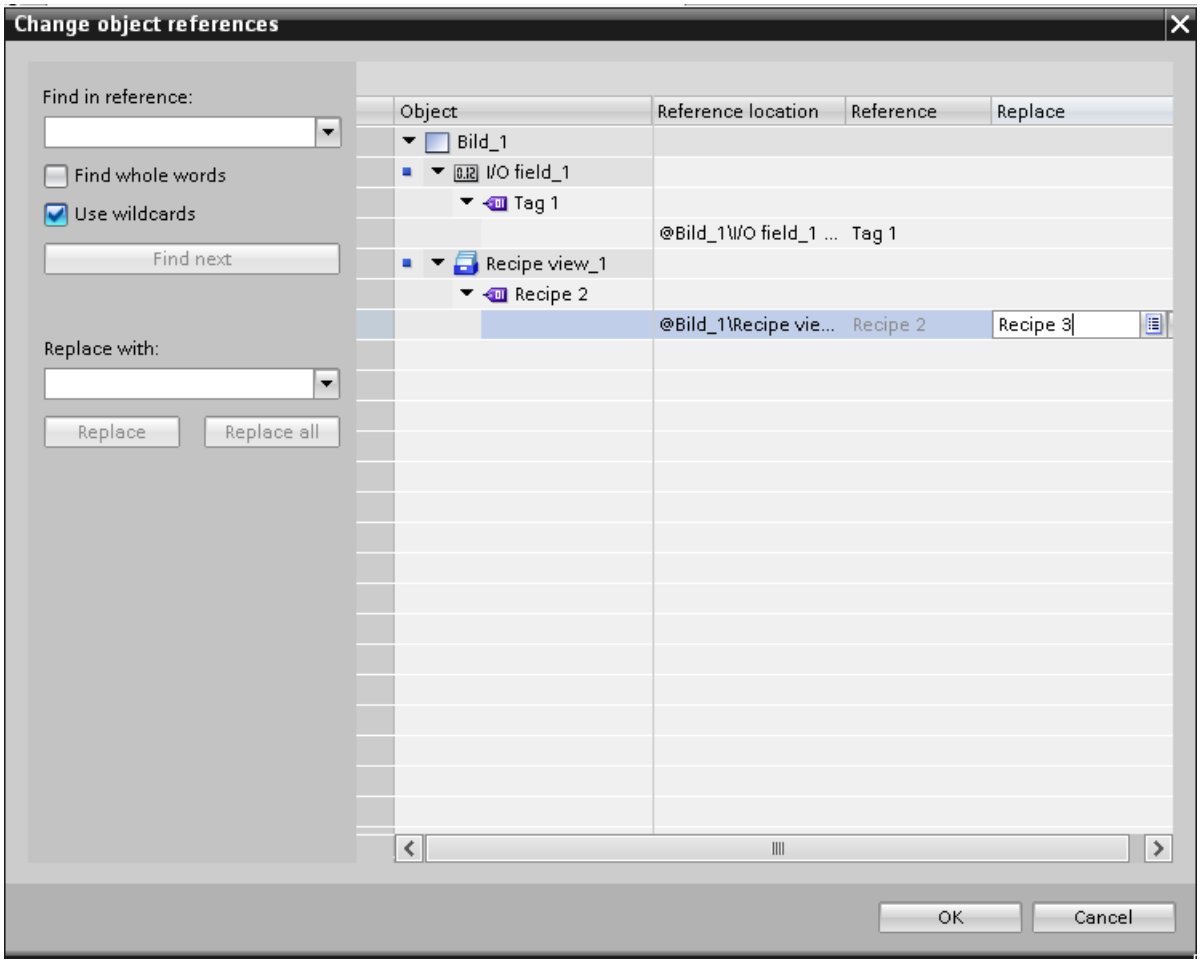

3. Enter the name or part of the name you want to find in the "Find in reference" input box.
4. Start the search with the "Find next" button. If a reference is found, it is highlighted in the "Reference" column.
   Another click on the "Find next" button continues the search process until the list has been completely searched.
   After this, the search process starts again from the top of the list.
5. Enter the name or part of the name you want to replace for the found name in the "Replace with" input box.
6. Click the "Replace" button to replace the selected tag.
   All references found are replaced by clicking the "Replace all" button. In this way, you can replace a large number of references with another
   reference.

7. To select a tag from the list of all tags created in the project, click on the button in the "Replace" column.
8. To select a tag from the object list or to create a new tag, click on the button in the "Replace" column.
9. Confirm your entry with "OK".

The rewiring of the tags is applied in the project.

Alternatively, you can enter a tag to replace the existing tag directly in the "Replace" column. You are supported by the auto-complete function in this process.

Note

When you search for a tag in the "Replace" column, you are supported by the auto-completion. After entering the first letter of the tag name in the "Replace", the auto-completion list contains the tags whose names contain the letters. Depending on the context, the auto-completion list only contains the tags that are permitted for the intended purpose.

Note

The tags shown in the column are displayed as read-only if the reference object is read-only, for example, a tag in a screen that has been stored in the library.

Note

It is possible to manually create a tag that does not yet exist. You create this tag at a later time.

Additional options for searching

You can narrow down your search by enabling the following options:

- Use wildcards
  Enter * for any number of characters. Example: You want to search for all tags starting with "HMI". Enter "HMI*" in the search key box.
  Enter ? for an individual character.
- Find whole words
  To search only for full names, enable the "Find whole words" option.

Replacing arrays and user-defined tags

Tags with names containing the special characters . [ ] are shown in quotation marks.

Array type tags and tags of user-defined data types (UDT) contain special characters to separate tags from their elements. However, they are not shown in quotation marks. To replace an array or a UDT with an HMI tag with special characters, enter the name of the HMI tag in quotation marks in the "Replace" input box. If you type the name without quotation marks, a reference to a non-existent array or UDT is created instead of a reference to the desired HMI tag with special characters.

Correcting entries

If an incorrect entry is made in the "Replace with" input box, the existing references can be replaced by references to non-existent tags.

If you detect an incorrect entry before confirming with "OK", you can correct the error. To do this, select the "Replace" column and press the "Remove" button. Any incorrect entries in the "Reference" column are deleted.

If you confirm the dialog with "OK", the affected properties of the screen objects are marked in red. Non-existent tags are highlighted in red in the "Change cross references" dialog.

If the new tag does not yet exist, you must subsequently create the tag.

14.5 Managing languages

14.5.1 Languages in WinCC

User interface language and project languages

A distinction is drawn between two different language levels in WinCC:

- User interface language
  During configuration, the text in the WinCC menus and dialogs is displayed in the user interface language. The user interface language also affects the labeling of operating elements, the parameters of the system functions, the online help, etc.
- Project languages
  Project languages are all languages in which a project will later be used. Project languages are used to create a project in multiple languages.

The two language levels are completely independent of one another. For example, you can create English projects at any time using a German user interface and vice versa.

Project languages

The following languages are differentiated within the project languages:

- Reference language
  The reference language is the language that you use to configure the project initially. During configuration, you select one of the project languages as the reference language. You use the reference language as a template for translations. All of the texts for the project are first created in the reference language and then translated. While you are translating the texts, you can have them displayed simultaneously in the reference language.
- Editing language
  You produce translations of the texts in the editing language.
  Once you have created your project in the reference language, you can translate the texts into the remaining project languages. Select a project language respectively as an edit language and edit the texts for the appropriate language variant. You can change the editing language at any time.

  Note

  When switching the project languages, the assignment to the keys on the keyboard also changes. For some languages (for example, Spanish), the operating system does not allow you to switch to the corresponding keyboard assignment. In this case, the keyboard assignment is switched to English.

- Runtime languages
  Runtime languages are those project languages that are transferred to the HMI device. You decide which project languages to transfer to the HMI device depending on your project requirements.
  You must provide appropriate controls so that the operator can switch between languages in runtime.

See also

Language settings in the operating system (Page 2013)

Operating system settings for Asian languages (Page 2014)

Selecting the user interface language (Page 2015)

"Project graphics" editor (Page 2025)

Languages and fonts in Runtime (Page 2029)

Example: Configuring a button for language switching (Page 2037)

HMI device wizard basics (Page 1929)

14.5.2 Language settings in the operating system

Introduction

The configuration PC operating system settings influence WinCC language management in the following areas:

- Selection of project languages
- Regional format of dates, times, currency, and numbers
- Displaying ASCII characters

Project language selection

A language is not available as a project language unless it is installed in the operating system.

Regional format of dates, times, currency, and numbers

WinCC specifies a fixed date and time format in the Date - Time field for the selected project language and runtime language.

In order for dates, times, and numbers to be presented correctly in the selected editing language, this language must be set in the Regional Options in the Control Panel.

Displaying ASCII characters

With text output fields, the display of ASCII characters as of 128 depends on the set language and the operating system being used.

If the same special characters are to be displayed on different PCs, the PCs must use the same operating system and regional settings.

See also

Languages in WinCC (Page 2012)

Operating system settings for Asian languages (Page 2014)

14.5.3 Operating system settings for Asian languages

Settings on Western operating systems

If you want to enter Asian characters, you must activate the support for this language in the operating system.

The Input Method Editor (IME) is available in Windows for configuring Asian texts. Without this editor, you can display Asian text but not edit it. For more information on the Input Method Editor, refer to the documentation for Windows. To enter Asian characters when configuring, switch to the Asian entry method in the "Input Method Editor".

Switch the operating system to the appropriate language to have language-specific project texts, such as alarm texts, displayed in the simulator in Asian characters.

Settings on Asian operating systems

If you are configuring on an Asian operating system, you must switch to the English default input language to enter ASCII characters, for example, for object names. As the English default input language is included in the basic installation of the operating system, you do not need to install an additional input locale.

Enable language support

1. Open the system controller.
2. Select "Regional and Language Options".
3. On the "Languages" tab, activate the check box "Install files for East Asian languages".
4. Then click on "Details" under "Text Services and Input Languages". The dialog "Text Services and Input Languages" is opened.
5. On the "Settings" tab add the required default input language under the "Installed Services".
6. Select the language of the operating system in the "Language for non-Unicode programs" area in the "Advanced" tab.

See also

Languages in WinCC (Page 2012)

Language settings in the operating system (Page 2013)

14.5.4 Setting project languages

14.5.4.1 Selecting the user interface language

Introduction

The user interface language is used for displaying menu entries, title bars, infotexts, dialog texts and other designations in the WinCC user interface.

You can switch between the installed user interface languages during configuration. The labeling of the operating elements remains in the language you set when you added the object even if you change the user interface language.

Procedure

1. Select "Options > Settings" in the menu.
   The "Settings" dialog box is opened.
2. Select the desired user interface language under "General > General settings".

Result

WinCC will use the selected language as user interface language.

See also

Languages in WinCC (Page 2012)

Enable project languages (Page 2015)

Selecting the reference language and editing language (Page 2016)

14.5.4.2 Enable project languages

Introduction

The project languages are set in the "Project languages" editor. You define which project language is to be the reference language and which the editing language.

Enable project languages

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The lower-level elements will be displayed.
2. Double-click on "Project languages".
   The possible project languages will be displayed in the working area.
3. Enable the relevant project languages.

Note

Copying multilingual objects

The copies of multilingual objects to a different project only include text objects in the project languages which are activated in the target project. Activate all project languages in the target project to include the corresponding text objects when transferring the copy.

Disabling project languages

1. Disable the languages which are not relevant for the project.

NOTICE
If you disable a project language, all text and graphic objects you have already created in this language will be deleted from the current project.

See also

Selecting the user interface language (Page 2015)

Selecting the reference language and editing language (Page 2016)

14.5.4.3 Selecting the reference language and editing language

Introduction

The project languages are set in the "Project languages" editor. You define which project language is to be the reference language and which the editing language. You can change the editing language at any time.

Requirements

The "Project languages" editor is open.

Several project languages have been activated.

Selecting the reference language and editing language

1. Click the arrow in the drop-down list in the "General > Editing language" section.
2. Click on the required language in the drop-down list, for example, German.

3. Click on the arrow in the drop-down list in the "General > Reference language" section.
4. Click on the required language in the drop-down list, for example, English.

The language selection is displayed in the list box.

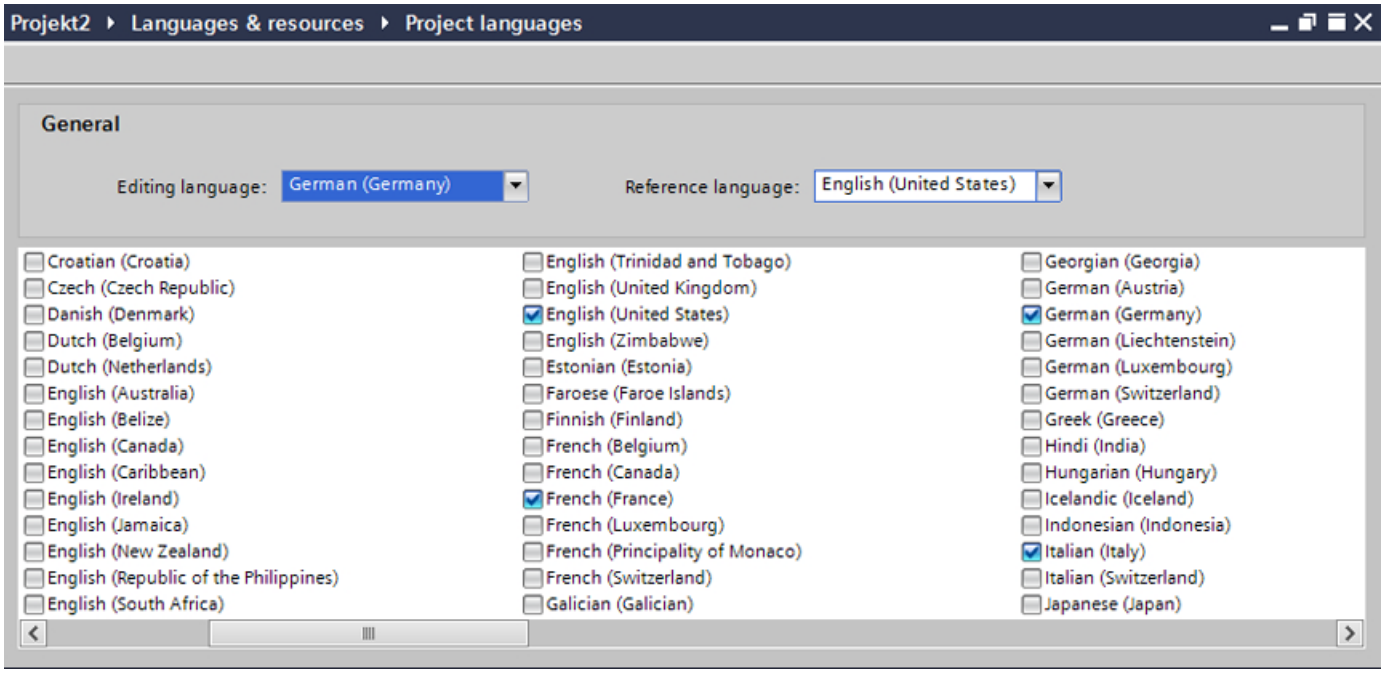

Result

You have now selected the editing and reference languages.

If you change the editing language, all future text input will be stored in the new editing language.

See also

Selecting the user interface language (Page 2015)

Enable project languages (Page 2015)

14.5.5 Creating one project in multiple languages

14.5.5.1 Working with multiple languages

Multilingual configuration in WinCC

You can configure your projects in multiple languages using WinCC. There are various reasons for creating a project in multiple languages:

- You would like to use a project in more than one country.
  You create the project in multiple languages but when the HMI device is commissioned, only the language spoken by the operators at the respective site will be transferred to the HMI device.
- The operators of a system speak a range of different languages.
  Example: An HMI device is used in China, but the service personnel understand only English.

Translating project texts

With WinCC, you can enter project texts directly in several languages in various different editors, for example, in the "Project texts" editor. WinCC also allows you to export and import your configuration for translation purposes. This is particularly advantageous if you configure projects containing a large amount of text and want to have it translated.

Language management and translation in WinCC

The following editors are used to manage languages and translate texts in WinCC:

Editor	Short description
Project languages	Selection of project languages, editing language and reference language.
Languages and fonts	Management of runtime languages and fonts used on the HMI device.
Project texts	Central management of configured texts in all project languages.
Graphics	Graphic library for managing graphics and their language-specific versions.

See also

Project text basics (Page 2019)

Translating texts directly (Page 2020)

Translating texts using reference texts (Page 2021)

Exporting project texts (Page 2022)

Importing project texts (Page 2024)

14.5.5.2 Project text basics

Texts in different languages in the project

Texts that are output on display devices during processing are typically entered in the language in which the automation solution is programmed. Comments and the names of objects are also entered in this language.

If operators do not understand this language, they require a translation of all operator-relevant texts into a language they understand. You can therefore translate all the texts into any language. In this way, you can ensure that anyone who is subsequently confronted with the texts in the project sees the texts in his/her language of choice.

User texts and system texts

In the interests of clarity, a distinction is drawn between user texts and system texts:

- User texts are texts created by the user.
- System texts are texts created automatically and which are a product of configuration in the project.

The project texts are managed in the project text editor. This can be found in the project tree under "Languages & Resources > Project texts".

Examples of multilingual project texts

You can, for example, manage the following types of text in more than one language:

- Display texts
- Alarm texts
- Comments in tables
- Labels of screen objects
- Text lists

Translating texts

There are two ways of translating texts.

- Translating texts directly
  You can enter the translations for the individual project languages directly in the "Project texts" editor.
- Translating texts using reference texts
  You can change the editing language for shorter texts. You can enter the new texts in the editing language while the texts of the reference language are displayed.

See also

Working with multiple languages (Page 2018)

Translating texts directly (Page 2020)

Translating texts using reference texts (Page 2021)

Exporting project texts (Page 2022)

Importing project texts (Page 2024)

14.5.5.3 Translating texts directly

Translating texts

If you use several languages in your project, you can translate individual texts directly. As soon as you change the language of the software user interface, the translated texts are available in the selected language.

Requirements

- You are in the project view.
- A project is open.
- You have selected at least two further project languages.

Procedure

Proceed as follows to translate individual texts:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The elements below this are displayed.
2. Double-click on "Project texts".
   A list with the texts in the project is displayed in the work area. There is a separate column for each project language.

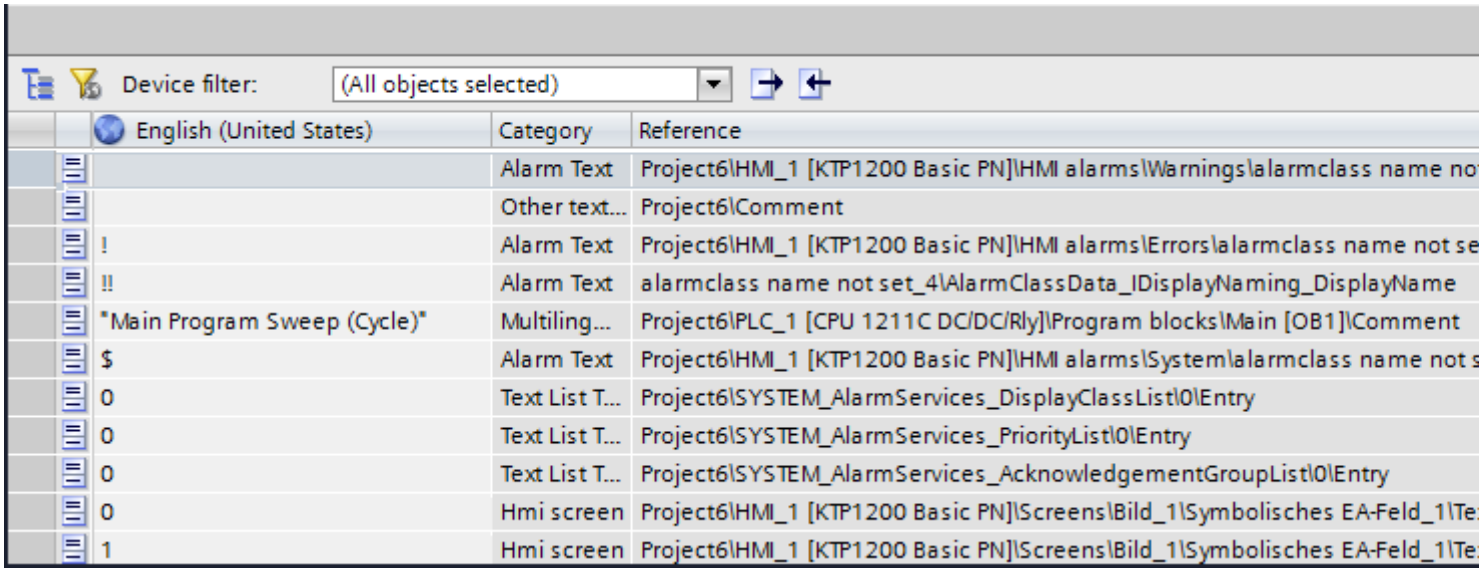

3. To group identical texts and translate them simultaneously, click on the "" button in the toolbar.

4. To hide texts that do not have a translation, click on the button in the toolbar.
5. Click on an empty column and enter the translation.

Result

You have translated individual texts in the "Project texts" editor. The texts will then be displayed in the runtime language.

See also

Working with multiple languages (Page 2018)

Project text basics (Page 2019)

Exporting project texts (Page 2022)

Importing project texts (Page 2024)

14.5.5.4 Translating texts using reference texts

Introduction

After changing the editing language, all texts are shown in input boxes in the new editing language. If there is not yet a translation available for this language, the input boxes are empty or filled with default values.

If you enter text again in an input field, this is saved in the current editing language. Following this, the texts exist in two project languages for this input field, in the previous editing language and in the current editing language. This makes it possible to create texts in several project languages.

You can display existing translations for an input box in other project languages. These serve as a comparison for text input in the current editing language and they are known as the reference language.

Requirement

There is at least one translation into a different project language for an input field.

Procedure

To display the translation of an input cell in a reference language, follow these steps:

1. Select "Tasks > Languages & resources" in the task card.
2. Select a reference language from the "Reference language" drop-down list.

Result

The reference language is preset. If you click in a text block, translations that already exist in other project languages are shown in the "Tasks > Reference text" task card.

See also

Working with multiple languages (Page 2018)

Project text basics (Page 2019)

Exporting project texts (Page 2022)

Importing project texts (Page 2024)

14.5.5.5 Exporting project texts

Project texts are exported for translation. Texts are exported to Office Open XML files ending in ".xlsx". These files can be edited in Microsoft Excel, for example.

You can exchange the file with the translators and import it back to the project as soon as it has been translated.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, German and English.

Exporting project texts

To export individual project texts, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The child elements are displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Select the texts you want to export.

4. Click . The "Export" dialog opens.

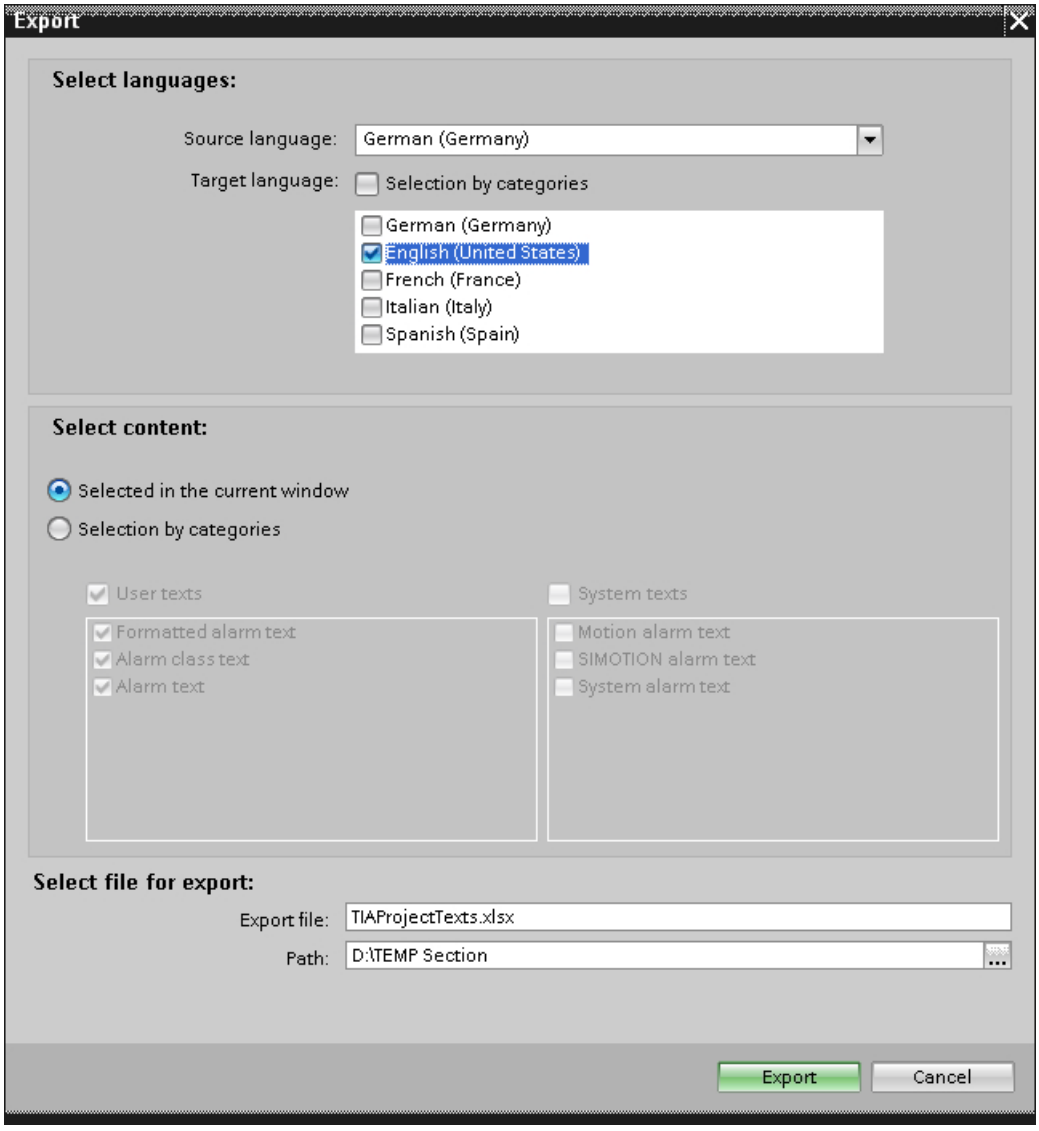

5. Select the required source language.
6. Select the required target languages.
   Select the "Select all" check box if you want to export all target languages.
7. Give the export file a name.
8. Set the path to the storage location of the export file.
9. Click "Export".

Result

The texts selected in the "Project texts" editor are written to an xlsx file. The xlsx file will be stored in the specified folder.

You can alternatively select and export all project texts from categories. Select "User texts" or "System texts" in the "Export" dialog in accordance with the type of texts you wish to export. In this case, export can additionally be limited by categories.

Note

Project texts in library objects cannot be exported.

See also

Working with multiple languages (Page 2018)

Project text basics (Page 2019)

Translating texts directly (Page 2020)

Translating texts using reference texts (Page 2021)

Importing project texts (Page 2024)

14.5.5.6 Importing project texts

Edit the xlsx file or send it to a translator. Import the texts once they have been translated. The foreign languages will be imported to the relevant object in the project.

Note

In WinCC, you only import the previously exported project texts into the same project. Importing into a different project is not supported.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Importing project texts

To import a project text file, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The lower-level elements will be displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Click on the button. The "Import" dialog opens.
4. Select the path and file name of the import file from the "Import file" field.
5. Activate the "Import source language" check box if you have made changes to the source language in the export file and would like to overwrite the entries in the project with the changes.
6. Click on "Import".

Result

You have imported the project texts.

See also

Working with multiple languages (Page 2018)
Project text basics (Page 2019)
Translating texts directly (Page 2020)
Translating texts using reference texts (Page 2021)
Exporting project texts (Page 2022)

14.5.6 Using language-specific graphics

14.5.6.1 "Project graphics" editor

Introduction

You use the "Project Graphics" editor to manage the configured graphic objects in different language versions. Multilingual projects sometimes also require language-specific versions of the graphics, for example, if

- the graphics contain text;
- cultural aspects play a role in the graphics.

Opening the "Project graphics" editor

Double-click in the project tree on "Languages and resources > Project graphics".

Work area

The work area displays all configured graphic objects in a table. There is a separate column in the table for each project language. Each column in the table contains the versions of the graphics for one particular language.

In addition, you can specify a default graphic for each graphic to be displayed whenever a language-specific graphic for a project language does not exist.

Preview

The preview shows you how the graphics will look on various devices.

See also

Languages in WinCC (Page 2012)

Storing an image in the graphics library (Page 2026)

Storing an external image in the graphics library (Page 2027)

14.5.6.2 Storing an image in the graphics library

Introduction

You use the "Graphics" editor to import graphics you want to use in screens in the "Screens" editor. It also allows you to manage language-specific versions of graphics. A preview shows the graphic displays on various HMI devices.

Note

File names for language-dependent graphics

Case is relevant in the file names of language-dependent graphics. Make sure that the format is consistent for all languages.

Requirement

- The language-dependent versions of a graphic are available.
- Multiple languages have been enabled in the "Project languages" editor.
- The "Graphics" editor is open.

Inserting graphics

1. Click "Add" in the "Graphics library" table. A dialog opens.
2. Select the required graphic file.
3. Click "Open" in the dialog box.
   The graphic will be imported to the project and displayed in all cells in this row in the "Graphics" editor.
4. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
5. Select "Add graphic" from the shortcut menu. A dialog box opens.
6. Select the desired graphic file and click "Open."
   The language-dependent version is inserted in the table in place of the reference language graphic.
7. Then, in the "Default graphic" column, import a graphic to be displayed in runtime for those languages for which there is no language-specific graphic.

You can also drag&drop a graphic from Windows Explorer to the relevant position in the "Graphics library" table.

Displaying graphics in the HMI device preview

1. Click on a graphic in the table.
2. Select the required HMI device under "Properties > Graphics settings > Device preview" in the Inspector window.
   The graphic will then be displayed as it will appear in runtime on the selected HMI device.

Result

The graphics added are available in the "Graphics" editor. The graphic assigned to the respective editing language will be displayed during editing. The default screen will be displayed in all editing languages for which no screen has been imported.

The screens assigned to the respective runtime language are displayed during runtime. The default screen is displayed in all runtime languages for which a screen has not been imported.

Note

If you disable a project language, all of the graphic objects you have already created in this language will be deleted from the current project.

See also

"Project graphics" editor (Page 2025)

14.5.6.3 Storing an external image in the graphics library

Introduction

To display graphics that have been created in an external graphics program in your screens, you will first have to store these graphics in the graphics browser of the WinCC project.

Requirement

- Multiple languages have been enabled in the "Project languages" editor.
- The "Graphics" editor is open.
- There is a graphic in the "Graphics" editor.

Creating and adding a new graphic as an OLE object

1. Click "Add" in the "Graphics library" table. A dialog box opens.
2. Navigate to the folder in which the graphic is stored.
3. Click "Open" in the dialog box.
   The graphic will be imported to the project and displayed in all cells in this row in the "Graphics" editor.

4. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
5. Select "Insert object" from the shortcut menu. The "Insert object" dialog box opens.

Note

In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

6. Select "Insert object > Create new" and an object type in the dialog.
7. Click "OK." The associated graphic program is opened.
8. Close the graphics program once you have created the graphic.
   The graphic will be stored in the graphic programming software standard format and added to the graphic browser.

Inserting created graphics in WinCC

1. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
2. Select "Insert object" from the shortcut menu. The "Insert object" dialog box opens.

Note

In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

3. From the "Insert object" dialog box, select "Create from file."
4. Click the "Browse" button.
5. Navigate to the created graphic and select it.

Note

To import graphics files, note the following size restrictions:

*.bmp, *.tif, *.emf, *.wmf ≤4 MB

*.jpg, *.jpeg, *.ico, *.gif "*≤1 MB

Result

The OLE objects added are available in the "Graphics" editor.

Versions of the graphics for the current editing language are displayed in the "Screens" editor. The default graphic is displayed in all editing languages for which no screen has been imported.

The graphic is displayed in runtime in the set runtime language. The default graphic is displayed in all runtime languages for which no graphic has been imported.

You can double-click OLE objects in your library to open them for editing in the corresponding graphic editor.

See also

"Project graphics" editor (Page 2025)

14.5.7 Languages in Runtime

14.5.7.1 Languages and fonts in Runtime

Using multiple Runtime languages

You can decide which project languages are to be used in Runtime on a particular HMI device. The number of Runtime languages that are available at one time on the HMI device depends on the device. To enable the operator to switch between languages in Runtime, you need to configure a corresponding operator control.

When Runtime starts, the project is displayed according to the most recent language setting. When Runtime starts the first time, the language with the lowest number in the "Order for language setting" is displayed.

"Language & font" Runtime setting

Configure the following under "Language & font":

- The project languages available as Runtime languages for the respective HMI device.
- The order in which the languages are switched.
- Default font for a Runtime language

The license status of the fonts is also shown in the "Download font" area. If a specific license is required for a protected font, this is indicated in the "License status" column.

Note

Post installing fonts

When a configured font on the configuration computer is missing, the project is not compiled. Re-installation is required.

After installing a font, restart the TIA Portal and completely re-compile your project. Only then can the HMI device be compiled without errors.

Note

Runtime language in controls (RT Professional)

If you have configured a control in a specific Runtime language that is not a user-interface language, it can happen that some texts, for example, system texts in the status bar, appear in the English language.

Note

Specify Runtime languages as input languages

To enter and edit data in Runtime, configure the specified Runtime languages also as input languages for the keyboard of your PC.

See also

Languages in WinCC (Page 2012)

Methods for language switching (Page 2030)

Enabling the runtime language (Page 2031)

Setting the runtime language order for language switching (Page 2032)

Setting the default font for a runtime language (Page 2034)

Selecting the log language (Page 2035)

14.5.7.2 Methods for language switching

Introduction

You need to configure language switching if you want to have multiple runtime languages available on the HMI device. This is necessary to enable the operator to switch between the various Runtime languages.

Methods for language switching

You can configure the following methods for language switching:

- Direct language selection
  Each language is set by means of a separate button. In this case, you create a button for each Runtime language.
- Language switching
  The operator switches the languages using a button.

Regardless of the method used, the button names must be translated into each of the languages used. You can also configure an output field that displays the current language setting.

See also

Languages and fonts in Runtime (Page 2029)

Enabling the runtime language (Page 2031)

Setting the runtime language order for language switching (Page 2032)

Setting the default font for a runtime language (Page 2034)

Selecting the log language (Page 2035)

14.5.7.3 Enabling the runtime language

Introduction

The "Language & Font" editor shows all project languages available in the project. Here you select which project languages are to be available as runtime languages on the HMI device.

Requirements

Multiple languages have been selected in the "Project languages" editor.

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font".
3. Select the following languages:
   - English
   - French
   - Italian

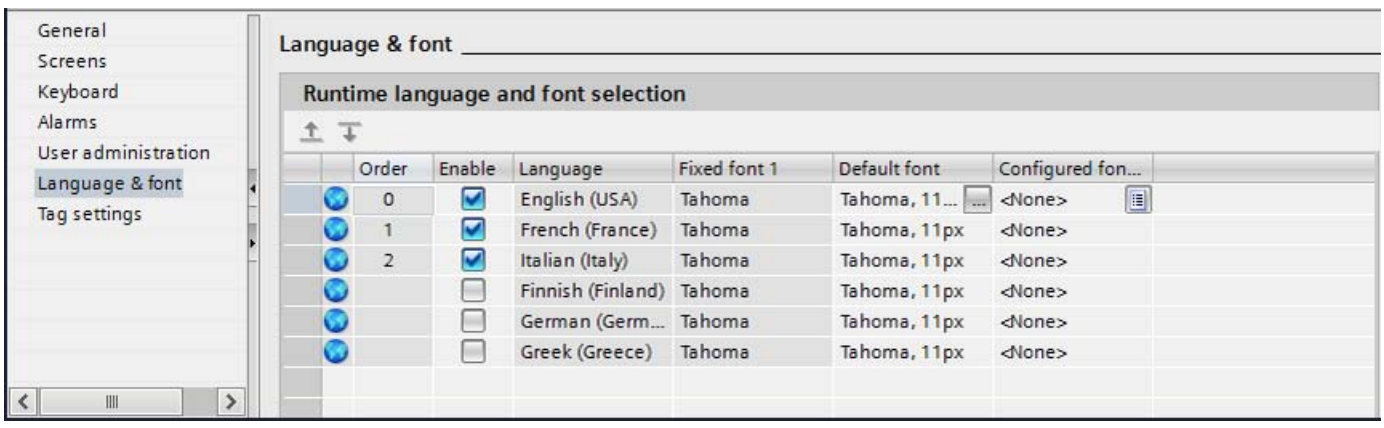

Result

You have now set three runtime languages. A number is automatically assigned to each language in the "Order" column. The enabled runtime languages are transferred with the compiled project to the HMI device.

If the number of languages selected exceeds the number that can be transferred to the HMI device, the table background changes color.

See also

Languages and fonts in Runtime (Page 2029)

Methods for language switching (Page 2030)

Setting the runtime language order for language switching (Page 2032)

Selecting the log language (Page 2035)

14.5.7.4 Setting the runtime language order for language switching

Introduction

You specify the language order for Runtime language switching. The first time Runtime starts, the project is displayed in the language with the lowest number in the "Order" column.

Requirements

- Multiple languages have been enabled in the "Project languages" editor.
- The "Language & Font" editor is open and three Runtime languages have been set in the following order:
  1. English
  2. Italian
  3. French

Procedure

1. Select the runtime language "English".
2. Click the button. The runtime language "English" is moved down a place. The number will automatically be changed to "1" in the "Order" column.

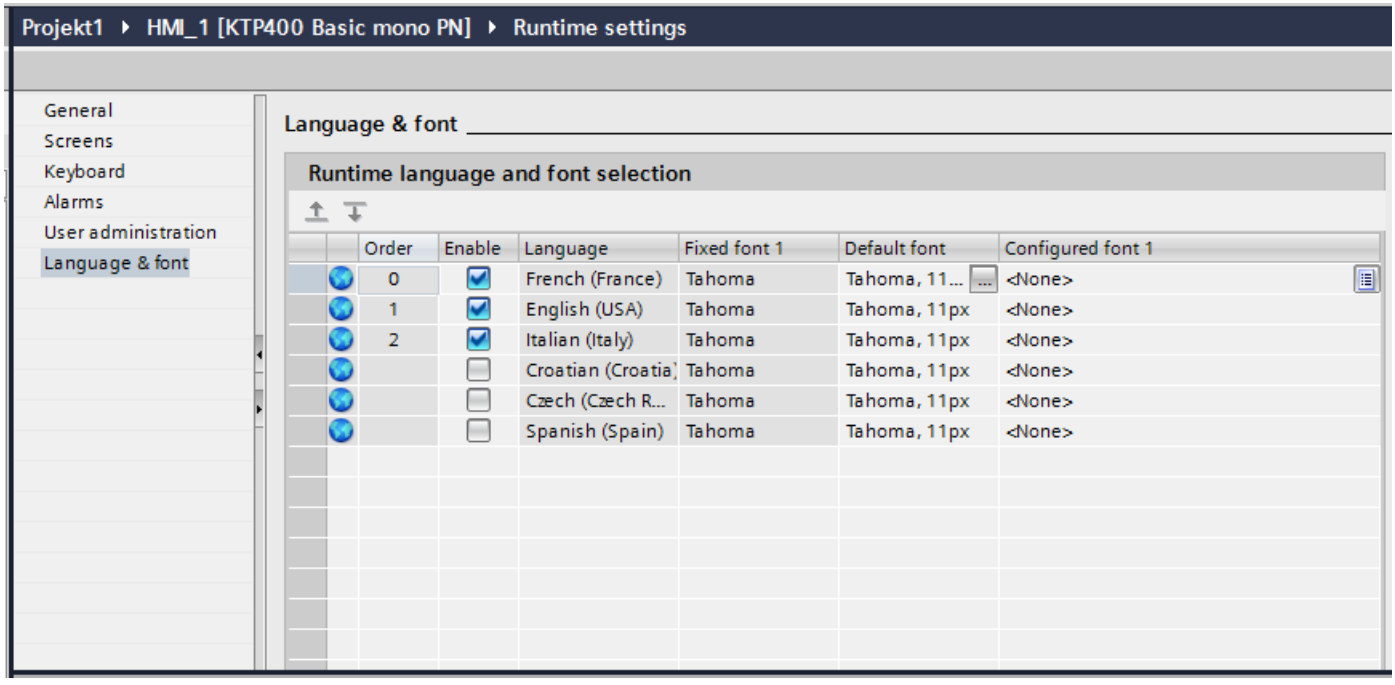

Result

You have changed the order of Runtime languages. The first time runtime starts, the project will be displayed in the language with the lowest number. If the language is switched, this will happen in numerical order.

See also

Languages and fonts in Runtime (Page 2029)

Methods for language switching (Page 2030)

Enabling the runtime language (Page 2031)

Setting the default font for a runtime language (Page 2034)

Selecting the log language (Page 2035)

14.5.7.5 Setting the default font for a runtime language

Introduction

You can specify the font used to display the texts for each runtime language on the HMI device in the "Language & Font" editor. The default font is used in all texts, such as dialog texts, for which you cannot define a specific font.

WinCC offers only fonts supported by the HMI device.

Note

"Bold" and "Italic" fonts

"Bold" and "italic" fonts are not supported on all HMI devices in Runtime.

Requirements

- Multiple languages have been enabled in the "Project languages" editor.
- Three runtime languages have been enabled in the "Language & Font" editor.
  1. Chinese
  2. German
  3. French

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font". The table shows the runtime languages and fonts set.
3. Click in the "French" row in the "Default font" column.
4. Select the font to be used by default if a font cannot be selected for a given text.

Result

The project texts for the runtime language "French" are displayed on the HMI device in the selected font.

These fonts are transferred to the HMI device during a transfer operation.

The default font is also used for the representation of dialogs in the operating system of the HMI device. Select a smaller font as default if the full length of the dialog texts or headers is not displayed.

See also

Languages and fonts in Runtime (Page 2029)

Methods for language switching (Page 2030)

Setting the runtime language order for language switching (Page 2032)

Selecting the log language (Page 2035)

14.5.7.6 Standardizing font for all languages

Introduction

You can standardize the font for all project languages during configuration with the "Use same font for all languages" option.

Requirement

- Multiple languages have been selected in the "Project languages" editor.
- Multiple languages have been selected in the "Language & font" editor.
- The same font is defined for the selected runtime languages under "Configured font".

Procedure

1. In the "Options > Settings > Visualization > General" menu, select the "Use same font for all languages" option.

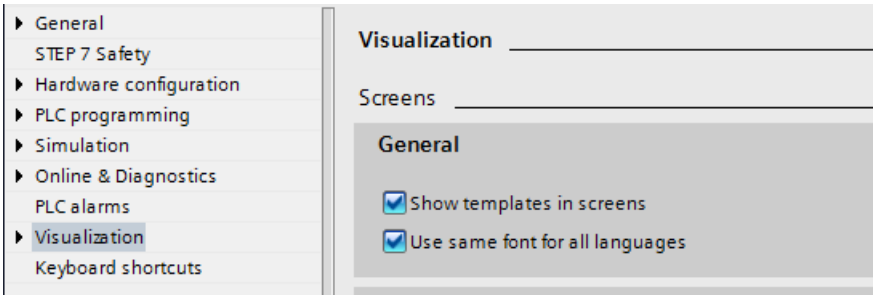

Result

You have enabled the option "Use same font for all languages". If you change the font of an object in one language during configuration, this font will be applied to all active languages.

14.5.7.7 Selecting the log language

Introduction

In the "Runtime settings > General" editor, select the language to be used for writing to logs in runtime.

Requirements

- The languages used in your project are activated in the "Project languages" editor, for example "German" and "English" .

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font".
3. Activate the runtime languages, for example, "German" and "English".
4. Specify the "order":
   – 1 German
   – 2 English
5. Click on "Runtime settings > General".
6. Select "German" for "Logs > Log language".

Result

After loading, the project will start in the runtime language "German". The logs are now written in German. During runtime, the operator switches the runtime language to English. The logs will still to be written in German.

See also

Languages and fonts in Runtime (Page 2029)

Methods for language switching (Page 2030)

Enabling the runtime language (Page 2031)

Setting the runtime language order for language switching (Page 2032)

Setting the default font for a runtime language (Page 2034)

14.5.7.8 Specific features of Asian and Eastern languages in Runtime (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)

Introduction

Note the following special considerations for the operation in Runtime of projects for Asian languages.

Note

During configuration, only use the Asian fonts that your configuration computer supports.

Memory requirement for Asian character sets

The memory requirement is greater when using Asian languages. Therefore look out for corresponding error messages when compiling the project.

Font size for Asian character sets

Use at least a font size of 10 points to display the text of projects created for Asian languages in Runtime. Asian characters will become illegible if smaller font sizes are used. This also applies to the default font in the Runtime settings under "Language & font".

Text field length for Asian languages

Make allowances for an appropriate length of the text fields when working on multilingual projects with Asian languages. Field contents may be partially hidden, depending on the font and the font size.

1. Open the "Properties > Appearance" text box in the Inspector window.
2. Under "Fit to size", disable the "Auto-size" option.
3. Verify the proper display in Runtime.

14.5.8 Example of multilingual configuration

14.5.8.1 Example: Configuring a button for language switching

Introduction

In this example, you configure a button that can be used to toggle between multiple runtime languages during runtime.

Requirements

- You have completed the "Configuring a button in multiple languages" example.
- The "Screen_1" screen is open.
- The button on the screen has been selected.

Procedure

1. In the Inspector window, select "Properties > Events > Press".
2. Click on "Add function" in the table.
3. Select the "SetLanguage" system function and the "Switch" setting.

Result

You have assigned the button the function "SetLanguage". Pressing the button during runtime will switch the runtime language. The runtime languages are switched in the order specified by the number sequence in the "Languages and fonts" editor.

See also

Example: Configuring a button in multiple languages (Page 2038)

Example: Configuring a button for language switching for each runtime language (Page 2039)

Languages in WinCC (Page 2012)

14.5.8.2 Example: Configuring a button in multiple languages

Introduction

In this example, you configure a "Sprache umschalten" button in German and "Switch language" button in English.

Requirements

- The languages "German" and "English" have been enabled in the "Project languages" editor.
- German has been set as editing and reference language.
- You have created and opened the "Screen_1" screen.
- The Inspector window is open.

Procedure

1. Drag-and-drop a button from the "Tools" task card into the screen.
   The button will be added to the screen.
2. In the Inspector window, open "Properties > Properties > General".
3. Enter the text ""Sprache umschalten" under "Text > off".
4. Press <Enter> to confirm. The button is named.
5. Open the "Tasks" task card.
6. Select "English" under "Languages & Resources > Editing language".
7. Enter the label "Switch Language" under "Properties > Properties > General > Text > Off" in the Inspector window.

Result

The button name is configured in German and English language. The button name corresponding with the current Runtime language is shown in Runtime.

See also

Example: Configuring a button for language switching (Page 2037)

Example: Configuring a button for language switching for each runtime language (Page 2039)

14.5.8.3 Example: Configuring a button for language switching for each runtime language

Introduction

In this example, you configure a "Sprache umschalten" button in German and "Switch language" button in English.

Requirements

- The following languages have been enabled in the "Project languages" editor
  - German
  - English
  - Italian
- All languages have been set as runtime languages in the "Runtime settings > Language & Font" editor.
- You have created and opened the "Screen_1" screen.
- Three buttons have been created on the screen:
  - Button_1 labelled "Deutsch"
  - Button_2 labelled "English"
  - Button_3 labelled "Italiano"
- The Inspector window is open.

Procedure

1. Select "Button_1".
2. In the Inspector window, select "Properties > Events > Press".
3. Click on <Add function> in the table.
4. Select the "SetLanguage" system function.
5. Click on the "Switch" field.
6. Click on the button.
7. Select "Runtime language". The field will be highlighted in red.
8. Select "German" from the drop-down list.
9. Repeat steps 1 - 8 for the other two buttons and select the corresponding runtime language.

Result

You have configured three buttons for switching language in runtime. Each button will switch to a different runtime language. For example, clicking on the "English" button during runtime will switch the runtime language to English.

See also

Example: Configuring a button for language switching (Page 2037)
Example: Configuring a button in multiple languages (Page 2038)

14.6 Replacing devices

14.6.1 Basics (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You can use existing configurations for new devices and optimize these configurations with little manual effort.

All data configured by you is retained in the configuration data. This means you do not need to copy individual objects of one device and paste them to another.

Principle

The following applies when you replace devices:

- Only functions supported by the new device are available. Only configuration data supported by the new device are displayed.
  This affects
  - recipes,
  - objects available on the screens,
  - available system functions,
  - available communication logs, etc.
- The number of supported objects, such as screens or tags, may be limited on the new device. If the existing objects exceed the limitations on the new device, the objects are displayed in full. The objects are, however, highlighted in color in the individual editors. An error is generated when the project data is compiled.
  Manual post processing is required when switching to a device with fewer features.
  Example: Limited number of connections
  All connections will be highlighted in color as invalid if fewer connections are supported on the new device than have been configured. Delete any excess connections.
- After an HMI device has been replaced, you should check the appearance of the configured screens. Changing the size of the display may result in changes to the position and appearance of screen objects, e.g. the recipe view and alarm view.

Note

If you replace a Panel and select a PC Station as your new device, for example, WinCC Runtime Advanced will automatically be moved below the PC Station in the project tree.

Note

Controller alarms after a Basic Panel device replacement

For the controller alarms (PLC monitoring alarms, program alarms or system alarms) to be displayed after a device replacement ("Device view > Change device") from a Basic Panel to a Comfort Panel, Mobile Panel or WinCC RT Advanced, you need to activate the "Display classes" setting for the respective connections under "Runtime settings > Alarms > Controller alarms".

See also

Device-specific functions (Page 2043)

Example: Replacing devices (Page 2052)

HMI device wizard basics (Page 1929)

Basic Panel (Page 2361)

Mobile Panel (Page 2368)

Comfort Panel (Page 2373)

WinCC Runtime Advanced (Page 2379)

Customizing the configuration of connections (Page 2042)

14.6.2 Customizing the configuration of connections

Introduction

If an HMI device is replaced, error messages of the type "... are not supported in the new configuration and will therefore be removed" may be generated. These alarms refer to configured connections of the device and are triggered, for example, if the HMI devices have a different number of interfaces. These connections are marked red after a device replacement. If you would like to continue to use these connections, you have to change the configuration of the connection.

Procedure

1. Open the "Devices and Networks" editor.
2. Click "Network" in the toolbar of the network view.
3. Network the interface of the HMI device with the interface of the CPU.
4. Click in the table area of the network view on the "Connections" table.
5. Select the connection marked red.
6. Enter the new interface under "Properties > General > Interface" in the Inspector window.

See also

Basics (Page 2041)

14.6.3 Device-specific functions

14.6.3.1 Device-specific functions (RT Advanced, Basic Panels, Panels, Comfort Panels)

Functions dependent on the device

Functions dependent on the device are implemented as follows:

- Colors
  The color is changed automatically when you switch from a device with full color display to one with a smaller color range.
  If you change the color manually and then change back again to a device with a larger range of colors, the reduced range of colors will be retained.
- Fonts
  Any configured font not available on a device will be replaced by a similar one or by the configured default font. The default font depends on the device selected.
- Font size
  Use small Windows fonts to display the text on devices. If you use large Windows fonts, then, depending on the size of the display, the text will not be displayed in full.
  The character scope is much greater for Asian languages. The use of different font sizes therefore has serious implications on the memory requirements of all devices.
  Use the same font type for all large characters throughout the project to ensure effective and efficient configuration.
- Screens and screen objects
  If the new device supports a different resolution than the previous device when you replace a device, there are several ways to adjust the screens.
  Adjust the size of the screens to the new device in the menu under "Options > Settings > Visualization > Fit to size screen".

See also

Basics (Page 2041)

Screen adjustment options (Page 2046)

14.6.3.2 Key assignment when replacing devices (Basic Panels)

Introduction

The devices available each have different function keys. The functions configured for these keys will be mapped to the available function keys of the new device if the device is replaced.

Function key mapping

The function keys below the display are mapped from left to right to the new device. If the new device has fewer keys, the keys it does not have are not mapped.

Note

HMI devices without LED keys

If you replace a device with configured LED keys with a device without LED keys in the Engineering System, runtime will not start after the project data have been transferred to the device.

Delete the LED key configuration before you replace the device.

Example: Replacing a KTP1000 Basic with a KTP600 Basic

You have configured a function for F2 in KTP1000 Basic. This function is triggered by F2 following replacement with a KTP600 Basic.

If you have used F7 in a KTP1000 Basic, this function will no longer be available if the panel is replaced with a KTP600 Basic.

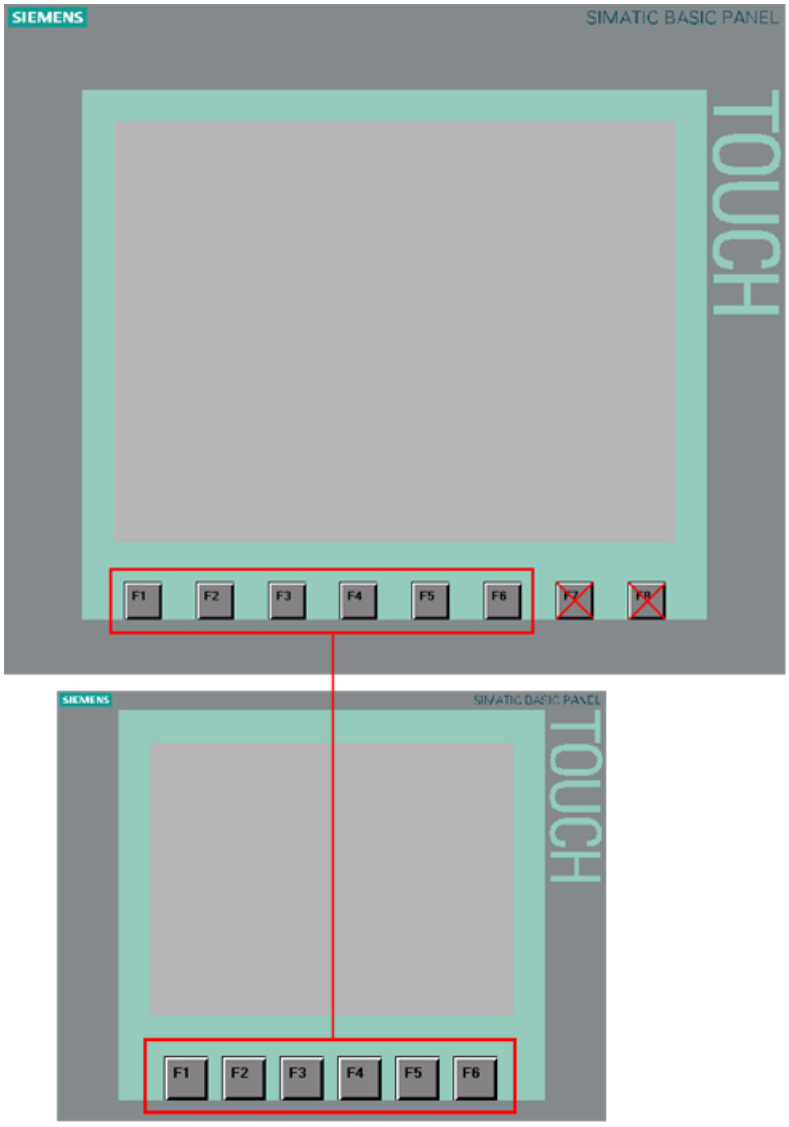

Mapping of control keys and cursor keys

The following keys are mapped only to the same keys of the new device:

- HELP
- ESC
- ACK
- ENTER
- PAGE UP
- PAGE DOWN
- CURSOR UP
- CURSOR DOWN

See also

Device-specific functions (Page 2043)

14.6.4 Adjusting screens to the new device

14.6.4.1 Screen adjustment options (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Select fit to size for screens before you replace a device. Fit to size is particularly important when switching devices with different display resolutions.

Object adjustment to content can be prevented for objects such as graphic views or text fields.

Note

The objects are distorted if you replace a device with a landscape format display with a device with a portrait format display. The difference in display format can, for example, result in object labels being cut off and content not being fitted to the object. You must therefore adjust the screens to the new device once you have replaced devices.

Screen adjustment when replacing devices

Adjust the size of the screens to the new device in the menu under "Tools > Settings > Visualization > Resize screen".

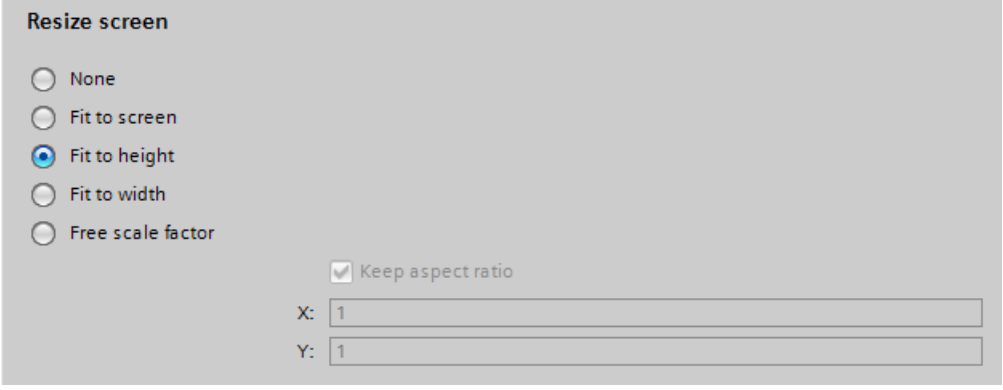

Select one of the following settings.

None (default setting)

The screens are not scaled. The objects in the screen retain their position and size. Use this setting as first test for checking of a possible exchange result because there are no rounding losses during forth and back exchange.

This option may result in objects being outside the configurable area if the display of the new device is smaller than the old one.

Fit to screen

The position and object size are adjusted to the new display size. Resizing takes place along the x-axis and the y-axis. Graphics and font size are adjusted accordingly.

Fit to height

The aspect ratio is maintained and the screens are adjusted to the height of the new device.

Use this option when you are replacing a device with display format 4:3, for example, with a device with widescreen.

Fit to width

The aspect ratio is maintained and the screens are adjusted to the width of the new device.

Use this option when you are replacing a device with widescreen, for example, with a device with display format 4:3.

Free scale factor

You select a free scale factor for screen adjustment. You can specify a factor for the x-axis and the y-axis.

Using a free scale factor of < 1 may distort the objects. Object labels may, for example, be cut off and the content may not be fitted to the object.

You must therefore adjust the screens to the new device once you have replaced devices.

Note

The aspect ratio is not adjusted for objects with a fixed aspect ratio, for example, gauge, circle. The objects are displayed on the new device with the same aspect ratio as prior to the replacement of the device.

See also

Specifying the position of screen objects (Page 2050)

14.6.4.2 Fit objects to contents (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

For some objects, you can specify fit to respective content in the Inspector window, for example:

- Text field: fit to text content
- I/O field: fit to text content
- Symbolic I/O field: fit to text content or to text list
- Graphic view: fit to included graphic

Fit to size for text and graphic objects

Disable automatic fit to size of the individual objects in the menu under "Options > Settings > Visualization > Resize screen and screen objects > Fit to content". This results in scaling of the objects as specified under "Options > Settings > Visualization > Resize screen and screen objects".

Select the objects which are not automatically fitted to size.

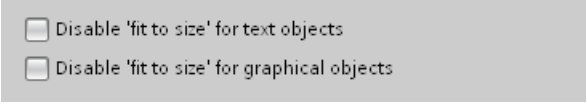

- If "Disable 'fit to size' for text objects" is activated, automatic fit to size is ignored in the text object properties.
  If you have activated "Fit screen to window height", the text field along with the other objects is scaled in accordance with the height of the new device.
- If "Disable 'fit to size' for graphic objects" is activated, automatic fit to size is ignored in the graphic object properties.
  If you have activated "Fit screen to window width", the graphics view along with the other objects is scaled in accordance with the width of the new device.

Note

The settings have no effect on screen objects whose size cannot be changed, such as alarm indicators or screen objects with a fixed aspect ratio.

"Disable 'fit to size' for text objects" and "Disable 'fit to size' for graphic objects" have no effect if:

- You have activated "Resize screen and screen objects > None".

- You have activated "Fit screen to window width and height" and the new device has the same resolution as the current device.

- You have activated "Fit screen to window height" and the new device has the same resolution as the current device.

- You have activated "Fit screen to window width" and the new device has the same resolution as the current device.

- You have selected "Resize screen and screen objects > None" and position "Top left" which means an adaptation is not necessary.

Example

The figure below shows the effects of automatic sizing using a graphic object with two buttons aligned as an example:

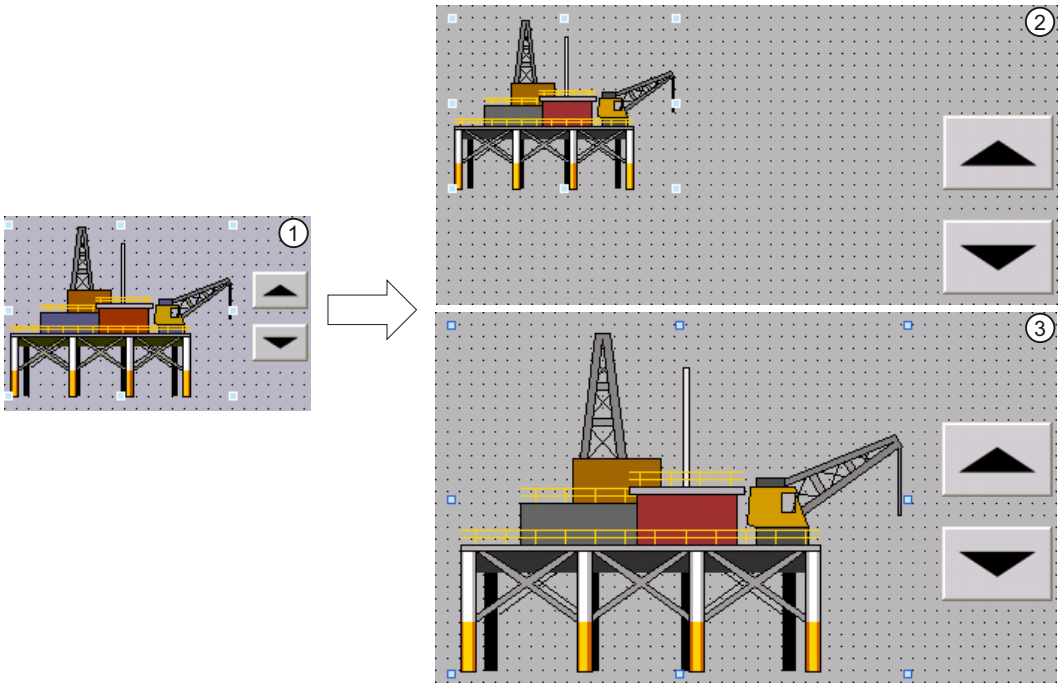

① Initial situation:

- Two buttons are aligned on a graphic object.
- The option "Fit object size to graphic" or "Adjust object size to graphic" is activated in the object properties of the graphic object under "Display > Sizing".

② Option 1: The original properties of the graphic object are to be maintained after switching the HMI device.

- Deactivate the option "Disable 'fit to size' for graphical objects" in the settings under "Size adaptation of objects".

Effect: The graphic object retains its original size after switching the HMI device. The alignment to the buttons is lost.

③ Option 2: The graphic object is to be placed relative to the new screen resolution after switching the HMI device.

- Activate the option "Disable 'fit to size' for graphical objects" in the settings under "Size adaptation of objects".

The option "Fit graphic to object size" is activated automatically in the object properties of the graphic object. The two buttons are properly aligned on the graphic object even after switching the HMI device.

See also

Specifying the position of screen objects (Page 2050)

14.6.4.3 Specifying the position of screen objects (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

There are various ways to adjust the position of screen objects to the new device.

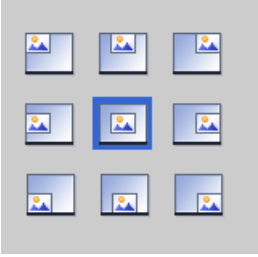

Select position

Adjust the position of the screen objects to the new device in the menu under "Options > Settings > Visualization > Fit to size screen > Select position".

Example

The following option aligns the objects with the top left edge.

The following object centers the objects in the middle of the screen.

See also

Screen adjustment options (Page 2046)

Fit objects to contents (Page 2048)

14.6.5 Example: Replacing devices

14.6.5.1 Example: Procedures overview (RT Advanced, Panels, Comfort Panels)

Introduction

The following example shows you how to replace a KTP700 Basic 7" PN with a TP1500 Comfort PRO.

Procedures overview

The procedure when replacing a device is as follows:

1. Adjust screens to the new device
2. Replacing devices

See also

Example: Adjusting screens (Page 2051)

Example: Replacing devices (Page 2052)

14.6.5.2 Example: Adjusting screens (RT Advanced, Panels, Comfort Panels)

Introduction

Adapt the screen before you replace a device. Screen adaptation is required as the display formats are different.

The KTP700 Basic format is 800 x 480 pixels while the format of the TP1500 Comfort PRO is 1366 x 768 pixels.

Requirements

- A project is open.
- The KTP700 Basic 7" PN device is used in the project.

Adjusting screens

1. Open the "Options > Settings" menu.
2. Click on "Visualization > Resize screen".

3. Activate "Fit to height".

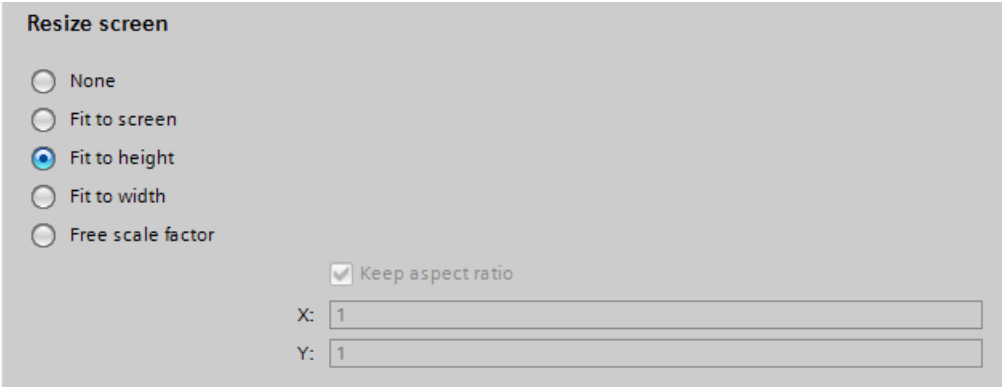

4. Select "Disable fit to size for text objects".
5. Select "Disable fit to size for graphical objects".

Result

You have carried out screen adjustment in preparation for replacing devices.

See also

Example: Procedures overview (Page 2051)
Example: Replacing devices (Page 2052)

14.6.5.3 Example: Replacing devices (RT Advanced, Panels, Comfort Panels)

Introduction

The following example shows you how to replace a device.

Requirements

- A project has been created and opened.
- The KTP700 Basic 7" PN device is used in the project.

Procedure

1. Double-click on "Devices & Networks" in the project navigation. The editor opens.
2. Click on the "KTP700 Basic PN" device.
3. Select "Change Device/Version" in the device shortcut menu. A dialog opens.
4. Select the TP1500 Comfort PRO device.

5. Select the matching image for the HMI device version that suits your configuration under "Version".
   When you change the image, the new image is automatically transferred to the target system with the next download. All runtime files are deleted during this step.
   Details of hardware differences can be found in the "Compatibility information".

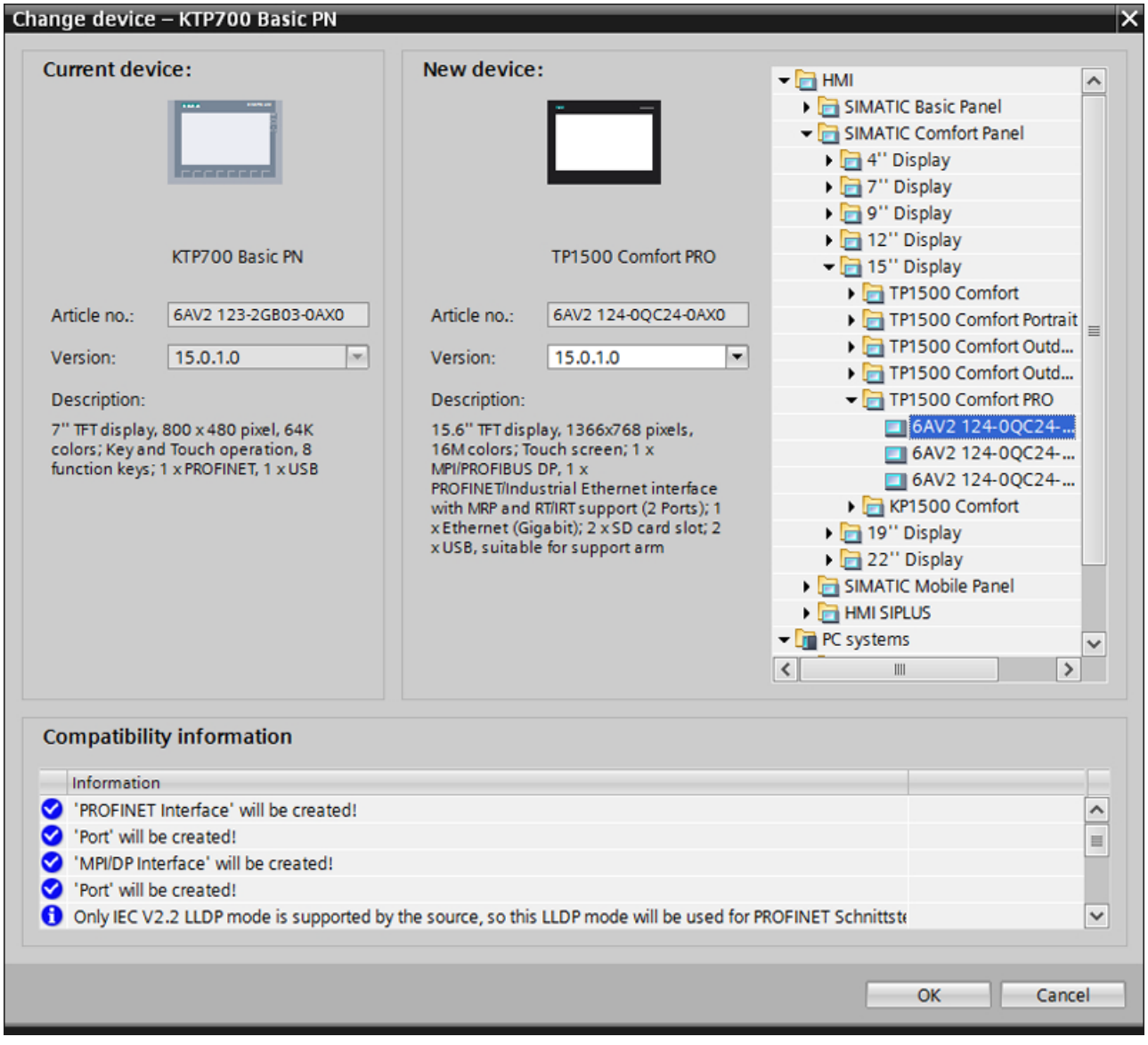

6. Click "OK". Device replacement is started.

Result

You have replaced the KTP700 BASIC device used in the project. You are now using the TP1500 Comfort PRO device.

See also

Example: Procedures overview (Page 2051)
Updating the operating system (Page 2340)
Changing the device version (Page 2102)
Changing between device versions (Page 2101)
Basics (Page 2041)
Example: Adjusting screens (Page 2051)

14.7 Copying between devices and editors

14.7.1 Basics

14.7.1.1 Basics (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Copying and pasting within an HMI device

You can copy and paste objects, such as display objects, within an HMI device. If the object is already created in the editor, when the object name is inserted a number is automatically attached, in accordance with the following principle:

- "<Object_name>_1" is renamed to "<Object_name>_2".
- "<Object_name>_2" is renamed to "<Object_name>_3".

Copying and pasting between HMI devices

You can also copy and paste between HMI devices. If an object with the same name already exists, you have the following options:

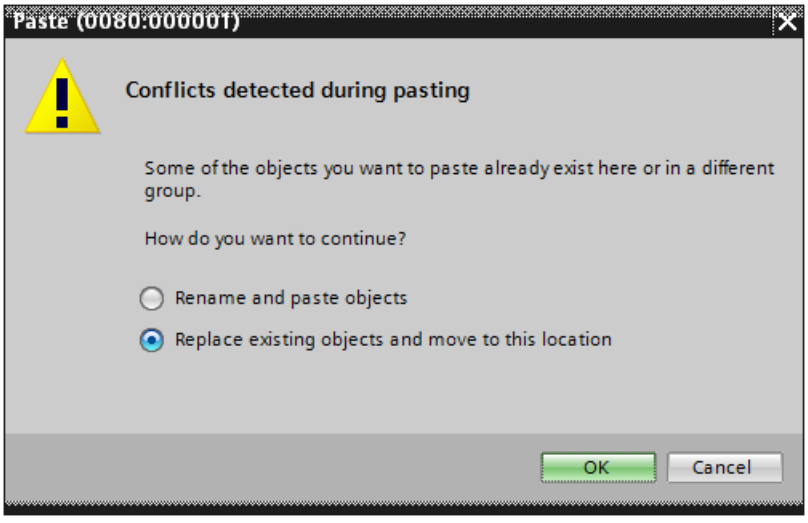

Note

Exception to this basic rule

Copying and pasting of an alarm class that has been generated from project-wide alarm class is handled differently than with this basic rule. When the copied alarm class already exists in the target HMI device within the same project, the "Paste" command is not performed.

Copying user-defined folders

You can create user-defined folders for editors, for example, for HMI tags, screens, etc.

You can copy user-defined folders and paste them into another HMI device. The objects contained in a user-defined folder may exceed the limitations applying to the other HMI device, such as the number of supported screens. After they have been pasted, all the objects are displayed. An error is displayed when the project data is compiled.

System folders cannot be copied.

See also

Unsupported objects and functionalities (Page 2057)

Copying objects with linked objects (Page 2059)

Linked objects copied automatically (Page 2059)

Copying screens (Page 2062)

Basics (Page 2064)

14.7.1.2 Copy and paste options (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Copy and paste options

Copy and pasting individual objects simplifies configuration.

WinCC offers a number of options for copying and pasting objects.

Shortcut menu

To copy and paste objects using the shortcut menu, proceed as follows:

1. Select an object, for example a button.
2. Select "Copy" in the shortcut menu.
3. Move the mouse cursor to the place on the screen where you want to paste the button.
4. Select "Paste" in the shortcut menu.
   The button will be pasted together with all properties already defined.

Drag&drop

To drag-and-drop objects, proceed as follows:

1. Click on "Screens > Start" in the project tree of a device.
2. Drag-and-drop the "Start" screen into the "Screens" folder of another device.
3. A dialog will appear if the second device already contains a screen with the same name.
4. Choose whether to replace or rename the existing screen.

14.7.1.3 Unsupported objects and functionalities (Professional) (RT Professional)

Introduction

When an object is copied, all its properties and settings are transferred to the target HMI device.

Unsupported objects

Objects that are not supported in the target HMI device cannot be pasted.

Note

When you copy a screen containing objects which are not supported by the destination HMI device, the objects remain in the background. When you copy the screen again and the new device supports the objects, they are displayed again.

Invalid objects

The following objects become invalid once they have been pasted into the target HMI device:

- Referenced objects that do not exist in the target HMI device.
- Objects with settings that are not supported in the target HMI device.
- System functions that were configured for objects and that are not supported in the target HMI device.

Invalid objects are highlighted by a color coding. Select a supported object or create a new one. If you retain an invalid object, an error will be displayed when the project data is compiled.

Colors and fonts

Colors and fonts are supported to varying degrees by HMI devices. When unsupported colors and fonts are pasted, they are replaced by supported colors and fonts. When you paste the same object back into the source HMI device, the original settings become active again.

14.7.1.4 Unsupported objects and functionalities (RT Advanced, Panels, Comfort Panels)

Introduction

When an object is copied, all its properties and settings are transferred to the target HMI device.

Unsupported objects

Objects that are not supported in the target HMI device cannot be pasted.

Note

When you copy a screen containing objects which are not supported by the destination HMI device, the objects remain in the background. When you copy the screen again and the new device supports the objects, they are displayed again.

Invalid objects

The following objects become invalid once they have been pasted into the target HMI device:

- Referenced objects that do not exist in the target HMI device.
- Objects with settings that are not supported in the target HMI device.
- System functions that were configured for objects and that are not supported in the target HMI device.

Invalid objects are highlighted by a color coding. Select a supported object or create a new one. If you retain an invalid object, an error will be displayed when the project data is compiled.

Colors and fonts

Colors and fonts are supported to varying degrees by HMI devices. When unsupported colors and fonts are pasted, they are replaced by supported colors and fonts. When you paste the same object back into the source HMI device, the original settings become active again.

Alarm classes

Only some of the enabled alarm classes are enabled.

When you configure an alarm window, for example, in the global screen in Project_1, you copy the alarm window and paste it to the global screen in Project_2.

Only some of the enabled alarm classes are enabled after pasting.

This behavior applies to the following display objects:

- Alarm window
- Alarm indicator
- Alarm view

See also

Basics (Page 2055)

14.7.2 Copying and pasting

14.7.2.1 Copying objects with linked objects (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

An object is linked to another object in the following situations, for example:

- You specify a tag for an alarm as a trigger tag.
  The alarm is the object. The tag is the linked object.
- You specify a connection for an external tag.
  The tag is the object. The connection is the linked object.

The object is always fully inserted during copying and pasting. Whether or not the linked object is pasted depends on the command used to insert it.

Simple pasting

The linked object is not copied. The linked object is transferred and handled as follows in the target HMI device:

- If an object with the same name exists, the existing object with its settings is used.
- If no object with the same name exists, the name of the object will be displayed. The object becomes invalid.

For some objects, linked objects are pasted automatically during simple pasting.

See also

Linked objects copied automatically (Page 2059)

Basics (Page 2055)

14.7.2.2 Linked objects copied automatically (Basic Panels)

Copying linked objects

The following table shows the objects for which linked objects are pasted automatically in simple pasting.

Object	Linked object
Screen	Template
Symbolic I/O field	Text list
Graphic I/O field	Graphics list
Graphic view	Graphic
Tag	Alarm
	Cycle

Object	Linked object
Recipe element	Text list
Scheduler	Triggers

See also

Copying objects with linked objects (Page 2059)

Basics (Page 2055)

Basics (Page 2064)

14.7.2.3 Linked objects copied automatically (RT Advanced, Panels, Comfort Panels)

Copying linked objects

The following table shows the objects for which linked objects are pasted automatically in simple pasting.

Object	Linked object
Screen	Cycle
	Template
Symbolic I/O field	Text list
Graphic I/O field	Graphics list
Graphic view	Graphic
Tag	Alarm
	Logging tag
	Cycle
Log	Logging tag
Logging tag	Log type
Recipe element	Text list
Scheduler	Triggers

14.7.2.4 Copying HMI devices with HMI connections

If you copy an HMI device with HMI connections to a PLC, the HMI connection in the new HMI device is not automatically connected to an existing PLC with the same name. This applies to copying within a project as well as copying across projects.

Completing the configuration of the HMI connection

To also access the PLC tag via an HMI tag in the new HMI device, follow these steps:

1. Open the "Devices & Networks" editor.
2. Connect the new HMI device to the required network.
3. Open the connection table.

4. Select the HMI connection of the new HMI device.
5. Select the required PLC under "Partner".

Note

If you compile the new HMI device or connect additional PLC tags in between copying the HMI device and completing the connection, there may be some instances in which an additional HMI connection to the same PLC is created. This is especially true if you connect HMI tags with DB array elements.

14.7.2.5 Copying linked screen objects with user data types

Introduction

If you copy a screen object from a screen that uses a user data type to a screen that does not use a user data type, the tag references in the target screen are shown with the prefix "@NOP::".

Interconnection of the user data type

If you want the tag references to function in Runtime, interconnect a user data type to the target screen. The interconnection of the target screen to the user data type can then be deleted afterwards.

14.7.2.6 Drag & drop from the details view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can improve configuration efficiency with just a few simple measures. Below are a few examples of efficient configuration.

Pasting objects to a screen from the details view

You can drag objects in the details view from various different editors to other editors.

Pasting a symbolic I/O field

1. Open a screen.
2. Click on the "Text and graphics lists" editor in the project tree. All existing text and graphics lists will be shown in the details view.
3. Click on a text list, for example, "Textlist1" in the Details view.
4. Drag-and-drop a text list from the Details view to a screen. A symbolic I/O field has been created and connected to the text list "Textlist1".

Pasting a graphic I/O field

1. Open a screen.
2. Click on the "Text and graphics lists" editor in the project tree. All existing text and graphics lists will be shown in the details view.
3. Click on a graphics list in the Details view, for example "Graficlist1".
4. Drag-and-drop a graphics list from the Details view to a screen. A graphic I/O field has been created and connected to the graphics list "Graficlist1".

Pasting an I/O field

1. Open a screen.
2. Click on the "HMI tags" editor in the project tree. All existing HMI tags will be shown in the Details view.
3. Click on an HMI tag in the Details view, for example "Tag1".
4. Drag-and-drop the HMI tag from the Details view to a screen. An I/O field has been created and connected to the HMI tag "Tag1".

14.7.2.7 Copying screens (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You copy one or more screens from the "Screens" folder and paste them into the "Screens" folder of another device.

Type and size of the displays

In the case of HMI devices with keys, the available keys are displayed automatically in the screen. When a screen is copied between HMI devices, the keys are either displayed or hidden; functions configured for function keys are not transferred.

If there is less space for the screen in the target HMI device than in the source HMI device, you can adjust the size of and the spacing between existing objects.

Automatic fit to size for objects

1. Select "Options > Settings > Visualization > Resize screen and screen objects" in the menu.
2. Activate, for example, "Fit to height".

See also

Basics (Page 2055)

14.7.2.8 Copying recipes within an HMI device (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

"Recipes" Editor

You can copy recipes, recipe elements and recipe data records within each table. You copy a recipe element to another recipe.

Only WinCC Runtime Professional: You can copy a recipe view element to another recipe view. If a recipe view element of the same name already exists, a conflict dialog is displayed. You can select whether to replace or rename the recipe element. You can copy recipe elements to the first empty row of the "Recipe views" editor, "Elements" tab.

You can copy a recipe data record to another recipe, if the other recipe contains the same number of recipe elements. If the data types differ, the value will be copied to the target data record but it is assigned an error flag.

"Tags" editor

You can drag-and-drop a tag to a recipe element in the "Tag" column. The tag is linked to the recipe element. If a tag is already linked, an error message will be generated.

"Screens" editor

If you drag-and-drop a recipe to a screen, a new recipe display will be created and linked to the recipe.

14.7.2.9 Differences between Runtime Advanced and Panels and Runtime Professional (RT Professional, RT Advanced, Panels, Comfort Panels)

Overview

The following table shows the availability of the various functions in Runtime Advanced and Panels and Runtime Professional.

Function in WinCC	Runtime Advanced and Panels	Runtime Professional
Logs	Yes Data log Alarm log	Yes Data log Compressed log Alarm log
Scheduler	Yes	Yes
User administration	Yes	Yes
Screens	Yes	Yes
Settings	Yes	Yes
Global elements	Yes	No

Function in WinCC	Runtime Advanced and Panels	Runtime Professional
Alarms	Yes Alarm indicator Alarm window Alarm view	Yes Alarm view with more functions than in Runtime Advanced and Panels
Menus and toolbars	No	Yes
Reports	Yes Printout using the "PrintReport" system function	Yes Printout using print job and scheduler
Recipes	Yes	User archives from WinCC V6.2
Scripts	Yes VBS	Yes VBS C
Structures	Yes	Yes
Text and graphics lists	Yes	Yes
Tags	Yes	Yes
Connections	Yes Different communication drivers	Yes Different communication drivers
Templates	Yes	No
Cycles	Yes	Yes

14.7.3 Copying and pasting between Runtime Advanced and Panels and Runtime Professional

14.7.3.1 Basics

Introduction

You can copy and paste project data such as screens, objects and tags between devices in different runtimes.

Properties and settings

Objects that are supported in both Runtimes can have different properties and settings. During copying and pasting these differences are implemented as follows:

- Properties not supported by the target HMI device are not displayed.
- Default settings are specified for properties that are only supported by the target HMI device.
- Settings which are not supported by the target HMI device are replaced by default settings.

If you copy an object back to the source HMI device without manually changing the settings in the target HMI device, all properties and settings originally set in the source HMI device will be reactivated.

Unsupported events and animations

When you paste an object into a different Runtime, the following cases are possible:

- The target HMI device does not support events or animations for the object copied.
  Events or animations configured for the copied object are not displayed in the target HMI device.
- Events or animations for the object copied are only supported by the target HMI device.
- Certain events and animations are not supported by the target HMI device for the object copied.
  Events or animations that were configured for the copied object and are not supported by the target HMI device are not displayed on the target HMI device.
- Certain events and animations for the object copied are only supported by the target HMI device.

All events and animations originally set will be reactivated if you copy the same object back to the source HMI device.

See also

Linked objects copied automatically (Page 2059)

Limitations during copying (Page 2066)

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2063)

Structure elements (Page 2072)

Objects (Page 2086)

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2088)

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2089)

Copying tags (Page 2070)

Basics (Page 2055)

14.7.3.2 System functions

Supported system functions

The following system functions are supported in Runtime Advanced and Panels as well as Runtime Professional:

- ActivateScreen
- IncreaseTag
- InvertBit
- InvertBitInTag
- InverseLinearScaling
- LinearScaling
- ResetBit

- ResetBitInTag
- SetBit
- SetBitInTag
- SetTag
- StopRuntime
- DecreaseTag
- ShowLogonDialog

Additional system functions are supported in Runtime Advanced and Panels.

See also

Screen objects (Page 2067)

14.7.3.3 Limitations during copying

Differences between the Runtimes

Some functions are supported exclusively in one Runtime. Some functionalities are supported in both Runtimes, but have different concepts. You cannot copy and paste These functions between the Runtimes.

Functionalities only supported by Runtime Advanced and Panels

- Templates
- Global elements

You cannot copy these functions to Runtime Professional.

Functionalities only supported in Runtime Professional

- Menus and toolbars
- C scripts

You cannot copy these functions to Runtime Advanced and Panels.

Functions with different concepts

- Configuration
- Recipes

These functions cannot be copied between the Runtimes.

See also

Basics (Page 2064)

14.7.3.4 Objects in screens

Introduction

When you copy an object from a screen and paste it into another screen, the object is inserted at the same position in the screen.

Text and graphics lists

When you copy a symbolic I/O field with a text list or a graphic I/O field with a graphics list and paste it into another runtime, the text list or graphics list will be copied to the target HMI device.

Copying objects from reports

You can copy objects that are supported in the screen from a report into a screen. Unsupported settings are replaced by supported settings. When you copy a copied object back into a report, the changed settings remain active.

Recipe view and alarm view

The "Recipe view" and "Alarm view" objects have such different concepts that you cannot copy and paste these objects between the two Runtimes.

See also

Screen objects (Page 2067)

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2069)

Objects (Page 2086)

14.7.3.5 Screen objects

Supported objects

The following table shows which objects in screens are supported in which Runtime:

Table 14-13 Basic objects

Object	Runtime Advanced and Panels	Runtime Professional
Line	Yes	Yes
Polyline	Yes	Yes
Polygon	Yes	Yes
Ellipse	Yes	Yes
Ellipse segment	No	Yes
Circle segment	No	Yes
Ellipse arc	No	Yes

Object	Runtime Advanced and Panels	Runtime Professional
Circular arc	No	Yes
Circle	Yes	Yes
Rectangle	Yes	Yes
Connector	No	Yes
Text field	Yes	Yes
Graphic view	Yes	Yes
Pipe	No	Yes
Double T-piece	No	Yes
T piece	No	Yes
Pipe bends	No	Yes

Table 14-14 Elements

Object	Runtime Advanced and Panels	Runtime Professional
I/O field	Yes	Yes
Editable text field	No	Yes
List box	No	Yes
Combo box	No	Yes
Button	Yes	Yes
Date/time field	Yes	No
Round button	No	Yes
Symbolic I/O field	Yes	Yes
Graphic I/O field	Yes	Yes
Bar	Yes	Yes
Switch	Yes	No
Symbol library	Yes	Yes
Slider	Yes	Yes
Scroll bar	No	Yes
Check box	No	Yes
Option buttons	No	Yes
Gauge	Yes	Yes
Clock	Yes	Yes
Disk space view	No	Yes
Charge condition	Yes	No
Effective range name	Yes	No
Effective range name (RFID)	Yes	No
Effective range signal	Yes	No
Zone name	Yes	No
Zone signal	Yes	No

Table 14-15 Controls

Object	Runtime Advanced and Panels	Runtime Professional
Screen window	No	Yes
Trend view	Yes	No
User view	Yes	Yes
HTML browser	Yes	Yes
Print job/Script diagnostics	No	Yes
Sm@rtClient view	Yes	No
Watch table	Yes	No
Recipe view	Yes	Yes
Alarm view	Yes	Yes
f(x) trend view	Yes	Yes
f(t) trend view	No	Yes
Table view	No	Yes
Value table	No	Yes
System diagnostics view	Yes	Yes
Media Player	No	Yes
Channel diagnostics view	No	Yes
PLC code view	Yes	Yes
GRAPH overview	Yes	Yes
ProDiag overview	Yes	Yes
Criteria analysis view	Yes	Yes
Camera view	Yes	No
PDF view	Yes	No

See also

Objects in reports (Page 2087)

Events for scheduled tasks (Page 2090)

Structure data types and properties (Page 2074)

Tag data types and properties (Page 2071)

System functions (Page 2065)

14.7.3.6 Connections

Differences between Runtime Advanced and Panels and Runtime Professional

Communication drivers

Please refer to the communications manual for information about the communication drivers possible in the Runtimes. Runtime Advanced and Panels and Runtime Professional support different communication drivers. There is no common communication driver for both runtimes.

Area pointer

The area pointers are not supported in Runtime Professional.

See also

Objects in screens (Page 2067)
Basics (Page 2064)
Copying connections (Page 2070)

Copying connections

Introduction

You can copy a connection in the "Connections" editor and paste it in the "Connections" editor of another Runtime.

Integrated connections

Integrated connections are created under "Devices & networks" in the network view. An integrated connection is displayed in the "Connections" editor. However, you cannot edit the integrated connection in the "Connections" editor. Any integrated connection you copy and paste from the "Connections" editor to another Runtime will become invalid in the target HMI device as it lacks a communication partner. You can only edit the copied integrated connection in the network view of the target device.

See also

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2069)

14.7.3.7 Copying tags

Introduction

You can copy a tag in the "Tags" editor and paste it in the "Tags" editor of another Runtime.

Unsupported data types

If you copy a tag with a data type that is not supported in the other Runtime, then the data type will become invalid.

Logging tags

If you have linked the tag to a logging tag, this logging tag will be copied to the target HMI device together with the tag. A logging tag has to be assigned to a data log. A new log will be created if there is no corresponding data log in the target HMI device.

Alarms

Any analog or discrete alarms you have configured for a tag will be copied to the target HMI device together with the tag.

System tags

System tags are supported in Runtime Professional only.

See also

Basics (Page 2064)
Tag data types and properties (Page 2071)
Discrete alarms (Page 2076)
Analog alarms (Page 2077)
Archive tags (Page 2081)

14.7.3.8 Tag data types and properties

Data types

The following table shows which data types are supported in which Runtime.

Data type	Runtime Advanced and Panels	Runtime Professional
Boolean	Yes	Yes
Byte (Signed Byte)	Yes	Yes
UByte (Unsigned Byte)	Yes	Yes
Short (Signed Int)	Yes	Yes
UShort (Unsigned Int)	Yes	Yes
Long (Signed Long)	Yes	Yes
ULong (Unsigned Long)	Yes	Yes
Float	Yes	Yes
Double	Yes	Yes
String	Yes	Yes
UString (Unicode String)	No	Yes
Raw (Raw Data Type)	No	Yes
Ref (Text Reference)	No	Yes
Date Time	Yes	Yes

Properties

The following properties are supported in Runtime Advanced and Panels as well as Runtime Professional:

- Address
- Content (comment)
- Connection
- Control variable
- Data type
- Length
- Limits:
  - Upper error
  - Generate analog alarm for upper error
  - Upper warning
  - Generate analog alarm for upper warning
  - Low
  - Generate analog alarm for lower warning
  - Lower error
  - Generate analog alarm for lower error
- Linear scaling
- Name
- HMI device end value
- HMI device start value
- PLC end value
- PLC start value
- Start value

See also

Screen objects (Page 2067)

14.7.3.9 Structures

Structure elements

Introduction

You can copy a structural element in the "Structures" editor and paste it in the "Structures" editor of another Runtime.

Unsupported data types

If you copy a structural element with a data type that is not supported in the other Runtime, the data type becomes invalid in the "Structure" editor.

See also

Structure types (Page 2073)

Structure data types and properties (Page 2074)

Basics (Page 2064)

Structure types

Introduction

You can copy a structure type from the "Structures" folder and paste it into the "Structures" folder of another Runtime.

Unsupported controller types

Since no controller type is supported in both Runtimes, the controller type will be invalid in the target HMI device.

Device family

The device family is adapted to the target HMI device when copied. If you copy a structure type from Runtime Advanced and Panels to Runtime Professional, for example, the "SmartCE" entry automatically becomes "SCADA."

Internal and external data types

In Runtime Advanced und Panels, a structure type can contain either exclusively internal data types or exclusively external ones. In Runtime Professional, both internal and external data types can be contained in one structure type.

If you copy a structure type that contains internal and external data types from Runtime Professional to Runtime Advanced und Panels, all internal data types become invalid. Since no communication peer is supported in both Runtimes, all external data types become invalid as well.

See also

Structure elements (Page 2072)

Structure data types and properties

Data types

The following table shows which data types are supported in which Runtime.

Data type	Runtime Advanced and Panels	Runtime Professional
Boolean	Yes	Yes
Byte (Signed Byte)	Yes	Yes
UByte (Unsigned Byte)	Yes	Yes
Short (Signed Int)	Yes	Yes
UShort (Unsigned Int)	Yes	Yes
Long (Signed Long)	Yes	Yes
ULong (Unsigned Long)	Yes	Yes
Float	Yes	Yes
Double	Yes	Yes
String	Yes	Yes
UString (Unicode String)	No	Yes
Raw (Raw Data Type)	No	Yes
Ref (Text Reference)	No	Yes
Date Time	Yes	No

Properties

The following properties are supported in Runtime Advanced and Panels as well as Runtime Professional:

Structure types:

- Comment: Contents
- Name
- Type of controller
- Device family

Structural elements:

- Comment: Contents
- Data type
- Length
- Linear scaling
- Name
- HMI device end value
- HMI device start value
- PLC end value
- PLC start value

- Error upper limit
- Error lower limit
- Start value
- Bit offset
- Offset

See also

Screen objects (Page 2067)

14.7.3.10 Alarms

Differences between Runtime Advanced and Panels and Runtime Professional

Supported objects in the screen

You can configure the following objects for screen alarm display in Runtime Advanced and Panels:

- Alarm view
- Alarm window
- Alarm indicator

In Runtime Professional, you can only configure an alarm view.

The concepts of the alarm view in the Runtimes are so different that it is not possible to copy an alarm view between the Runtimes.

Alarm text blocks

Alarm blocks are supported in Runtime Professional only.

See also

Discrete alarms (Page 2076)

Analog alarms (Page 2077)

Controller alarms and system alarms (Page 2078)

Alarm classes (Page 2079)

Alarm groups (Page 2079)

Basics (Page 2064)

Basics (Page 2055)

Discrete alarms

Introduction

You can copy a discrete alarm in the "Alarms" editor in the "Discrete alarms" tab and paste it to another Runtime. Pasting a tag with configured discrete alarms to the "Tags" editor will automatically also paste the corresponding discrete alarms to the "Alarms" editor.

Supported properties

The following properties are supported in Runtime Advanced and Panels as well as Runtime Professional:

- ID
- Alarm class
- Alarm group
- Acknowledgment: Tag
- Acknowledgment: Bit
- Tooltip
- Trigger: Tag
- Trigger: Bit
- Event text

Alarm number

Alarm numbers which do not yet exist in the target HMI device are transferred unchanged when copied. If the alarm number already exists, the next free number is assigned automatically as the alarm number for the discrete alarm.

Tooltip

Copying a tooltip with more than 255 characters to Runtime Professional will make the entire tooltip invalid.

Event text

The event text is displayed in the following dialog boxes of the Inspector window:

- Runtime Panels and Advanced:
  "Properties > Properties > General > Settings > Event text"
- Runtime Professional:
  "Properties > Properties > Texts > Settings > Event text"

The event texts are copied into the corresponding dialog boxes with all the format data.

Event texts will become invalid:

- if you copy an event text with more than 254 characters to Runtime Advanced and Panels.
- if you copy an event text with a dynamic parameter to Runtime Panels and Advanced.
- if you copy an event text with a text list from Runtime Panels and Advanced to Runtime Professional.

See also

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2075)

Analog alarms

Introduction

You can copy an analog alarm in the "Alarms" editor in the "Analog alarms" tab and paste it to another Runtime. Pasting a tag with configured analog alarms to the "Tags" editor will automatically also paste the corresponding analog alarms to the "Alarms" editor.

Supported properties

The following properties are supported in Runtime Advanced and Panels as well as Runtime Professional:

- Alarm number "ID"
- Event text
- Alarm class
- Alarm group
- Tooltip
- Trigger: Tag
- Delay
- Limit value
- Trigger mode
- Synchronize limit value
- Hysteresis mode
- Hysteresis
- Hysteresis as a percentage

Alarm number "ID"

Alarm numbers which do not yet exist in the target HMI device are transferred unchanged when copied. If the alarm number already exists, the next free number is assigned automatically as the alarm number for the analog alarm.

Event text

The event text is displayed in the following dialog boxes of the Inspector window:

- Runtime Advanced and Panels:
  "Properties > Properties > General > Settings > Event text"
- Runtime Professional:
  "Properties > Properties > Texts > Settings > Event text"

The event texts are copied into the corresponding dialog boxes with all the format data.

Event texts will become invalid:

- if you copy an event text with more than 254 characters to Runtime Advanced and Panels.
- if you copy an event text with a dynamic parameter to Runtime Panels and Advanced.
- if you copy an event text with a text list from Runtime Panels and Advanced to Runtime Professional.

Tooltip

Copying a tooltip with more than 255 characters to Runtime Professional will make the entire tooltip invalid.

Delay

If you copy a delay with the "Minute" or "Hour" unit from Runtime Professional to Runtime Advanced and Panels, the setting becomes invalid.

See also

Copying tags (Page 2070)

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2075)

Controller alarms and system alarms

Controller alarms

Since controller alarms are configured in the controller, it is not possible to copy controller alarms.

System alarms

Because system alarms are dependent on the HMI device, it is not possible to copy system alarms.

See also

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2075)

Alarm classes

Introduction

You can copy an alarm class in the "Alarms" editor in the "Alarm classes" tab and paste it to another Runtime.

Predefined alarm classes

You cannot copy and paste predefined alarm classes.

Log

In Runtime Advanced and Panels, you configure an archive in which the alarm is stored. In Runtime Professional, you specify whether the alarm is logged for each alarm of an alarm class.

During the copying process the logging specifications are implemented as follows:

- Copying from Runtime Advanced and Panels to Runtime Professional:
  If a log has been created in Runtime Advanced and Panels, "Log" is activated in Runtime Professional.
  If no log has been created in Runtime Advanced and Panels, "Log" is deactivated in Runtime Professional.
- Copying from Runtime Professional to Runtime Advanced and Panels:
  If "Log" is activated in Runtime Professional, you need to specify a log in Runtime Advanced and Panels.
  If "Log" is deactivated in Runtime Professional, the entry <No log> is specified in Runtime Advanced and Panels.

Flashing

The settings for the flash behavior are not copied.

See also

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2075)

Alarm groups

Copying and pasting

You can copy an alarm group in the "Alarms" editor in the "Alarm groups" tab and paste it to another Runtime.

Alarm groups from alarm classes

Alarm groups from alarm classes are only supported in Runtime Professional and cannot be copied and pasted to Runtime Advanced and Panels.

Hierarchy

Hierarchies of alarm groups are only supported in Runtime Professional. When copied between the Runtimes the hierarchy order is implemented as follows:

- Copying from Runtime Advanced and Panels to Runtime Professional
  If you copy an alarm group into the "Alarm groups" tab of the "Alarms" editor, the alarm group is inserted regardless of any existing alarm groups.
  If you copy an alarm group into a line containing an existing alarm group, the alarm group is inserted as an alarm group that is subordinate in the hierarchy.
- Copying from Runtime Professional to Runtime Advanced and Panels
  Hierarchies cannot be shown in Runtime Advanced and Panels. Alarm groups from all hierarchy levels are inserted independently of each other.

ID of the alarm group

Alarm numbers which do not yet exist in the target HMI device are transferred unchanged when copied.

See also

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2075)

14.7.3.11 Logs

Differences between Runtime Advanced and Panels and Runtime Professional

Logs

Data logs are supported in both Runtimes. You can copy and paste alarm logs and compressed logs between the runtimes.

Properties

The following properties are supported in Runtime Advanced and Panels as well as Runtime Professional:

Data logs

- Name
- Comment
- Storage location

Logging tags

- Name
- Process variable
- Triggers
- Logging cycle
- High limit
- Low limit
- Comment

Events

You cannot copy configured events between Runtime Advanced and Panels and Runtime Professional.

See also

Data logs (Page 2081)

Archive tags (Page 2081)

Basics (Page 2064)

Data logs

Copying and pasting

You can copy a data log in the "Historical Data" editor in the "Data log" tab and paste it to another Runtime. All logging tags contained in the data log are included automatically.

See also

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2080)

Archive tags (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can copy a logging tag in the "Historical Data" editor in the "Data log" tab and paste it in another Runtime.

In addition, you can copy a logging tag in the "Tags" editor of Runtime Professional and paste it into the "Historical Data" editor in Runtime Advanced and Panels. To copy a logging tag in Runtime Professional, open the "Tags" editor and select the "Alarms & Logs" tab in the upper work area. Copy the desired logging tag in the lower work area of the "Data logs" tab.

Tags used in the logging tags

Any logging tag you have linked to a tag will be automatically also copied if the tag is copied to the target HMI device. A new log will be created if there is no corresponding data log in the target HMI device. There may be performance problems if too much of this additional data is copied.

Multiple logging of tags

When copying from Runtime Professional to Runtime Advanced and Panels, the process tags of a logging tag become invalid in the following situations:

- The tag has been logged more than once in Runtime Professional.
- Several logging tags are linked to the same process tag.

Triggers

If you paste a logging tag with a cycle, the cycle will be included in the target HMI device.

In Runtime Professional, configure the log cycle of logging tags in the form "factor x cycle", for example, twice an hour. The "Factor" property is not supported in Runtime Advanced and Panels. The Factor property is therefore set to the default of "1" when copying from Runtime Advanced and Panels to Runtime Professional. When copying from Runtime Professional to Runtime Advanced and Panels, only the cycle can be seen in Runtime Advanced and Panels; the factor is not taken into consideration. If you copy a cycle of "3 x 1h" from Runtime Professional to Runtime Advanced and Panels, for example, this is set in Runtime Advanced and Panels as "1h".

Copying logging tags from the "Tags" editor

When you copy a logging tag from the "Tags" editor in Runtime Professional into the "Historical Data" editor in Runtime Advanced and Panels, the "Process tag" property is linked to an existing tag with the same name. If there is no tag with the same name, the "Process tag" property becomes invalid. The "Data log" property is linked to the selected data log.

If you copy a logging tag from the "Tags" editor in Runtime Professional to the "Tags" editor of another HMI device, the "Data log" property will not be transferred.

Copying logging tags from the "Historical Data" editor

When you copy a logging tag from the "Historical Data" editor in Runtime Advanced and Panels into the "Tags" editor in Runtime Professional, the "Data log" property will not be transferred.

When you copy a logging tag from the "Historical Data" editor into the "Logs" editor of another HMI device, the "Process tag" property is linked to an existing tag with the same name. If there is no tag with the same name, the "Process tag" property becomes invalid. The "Data log" property is linked to the selected data log.

Logging tags used in compressed logging tags

When you copy a logging tag used in a compressed logging tag in Runtime Professional to Runtime Advanced and Panels, the settings for the compressed logging tag are also copied in the background. There may be performance problems if too many of these unnecessary settings are copied to Runtime Advanced and Panels.

See also

Copying tags (Page 2070)

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2080)

14.7.3.12 Copying VB scripts

Object types, properties and methods in VB script

The objects, object types, properties and methods supported in Runtime Advanced and Panels and Runtime Professional are listed in the VBS reference.

Pasting

You can copy VB scripts

- using the shortcut menu or by drag & drop from the project tree
- from the code of an open VB script to another open VB script.

Objects called in VB script

You copy a VB script. Objects and their properties which do not exist in or are not supported by the target HMI device are called in the VB script. The syntax check reports an error. Correct the invalid object references.

Variables called up in a VB script

The syntax check will report errors if tags are called in the copied VB script that do not exist in the target HMI device or only exist with a different data type. Replace the invalid tags with tags that exist in the target HMI device.

If you are planning to copy scripts into different Runtimes, use the same variables in the Runtimes.

14.7.3.13 Copying recipes between Runtime Advanced and Panels and Runtime Professional

Recipe and recipe element

The recipe elements and recipe data records will also be copied if a recipe is copied.

Note

Inconsistent data

Properties not supported by the HMI device, for example data type, will be formally copied but will appear with an error flag.

Check the properties and reassign those flagged.

You can drag-and-drop a recipe element into the first empty line of the "Recipes Element" editor of another HMI device. The recipe element is inserted into the other recipe. If the recipe does not have the "Tags" communication type, the associated tag will be copied but it is assigned an error flag.

Points to note when copying from WinCC Runtime Advanced and Panels to WinCC Runtime Professional

The properties below are preset as follows and may subsequently be edited:

- "Tags" communication type
- "Limited" size type
- Number of data records

The storage location of the recipe is changed to "Database".

With a recipe element, the link will be copied but not the linked tag. You need to copy the tag separately or create a new tag with the same name. The same applies to the text list.

Note

Inconsistent data

Array tags are not supported in WinCC Runtime Professional. After copying the recipe, create as many recipe elements and tags as there are array elements in the linked array tag. Link the new recipe elements to the tags. Copy the values of the source recipe data records to the target recipe data records.

Points to note when copying from WinCC Runtime Professional to WinCC Runtime Advanced and Panels

The following properties will be changed but cannot be edited:

- "Tags" communication type
- "Limited" size type

- Number of data records per HMI device
- "File" storage location with default path

Recipe data record and recipe query

You can copy a recipe data record to another recipe, if the other recipe contains the same number of recipe elements. If the data types differ, the value will be copied to the target data record but it is assigned an error flag.

Features specific to WinCC Runtime Professional

You can drag-and-drop a recipe query into the first empty row of the "Recipe queries" editor of another HMI device. The recipe query is copied but not the linked recipe elements and their recipes.

14.7.3.14 Copying cycles

Introduction

You can copy cycle in the "Cycles" editor and paste it in the "Cycles" editor of another Runtime. Referenced cycles are also copied whenever tags, logging tags or tasks are copied. The cycles are then displayed in the "Cycles" editor.

Supported properties

The date and time of the last change are updated during pasting.

Properties

The "Starting point" property is only supported in Runtime Professional.

Cycle units

Runtime Professional and Runtime Advanced and Panels support different cycle units.

See also

Basics (Page 2064)

14.7.3.15 Reports

Objects

Introduction

When you copy an object from a report and paste it into another report, the object is inserted at the same position in the report.

Supported properties

The concepts of the "Recipe display" and "Alarm view" objects differ so greatly that they cannot be copied and pasted between the various Runtimes.

Text and graphics lists

Text and graphics lists are automatically copied along with symbolic and graphic I/O fields and are then displayed in the "Text and graphics lists" editor.

Copying objects from screens

You can copy objects supported in the report from a screen to a report. Unsupported settings are replaced by supported settings. When you copy a copied object back into a screen, the changed settings remain active.

See also

Objects in screens (Page 2067)

Copying a report (Page 2086)

Objects in reports (Page 2087)

Basics (Page 2064)

Copying a report

Introduction

You can copy a report from the "Reports" folder and paste it into another "Reports" folder.

Watermarks

Watermarks are only supported in Runtime Professional. When you copy a report from Runtime Professional to Runtime Advanced and Panels, the areas for the watermarks will no longer be shown.

Print jobs

The print jobs in the "Reports" folder are only supported in Runtime Professional and cannot be copied and pasted between Runtimes.

See also

Objects (Page 2086)

Objects in reports

Supported objects

The following table shows which objects are supported in which Runtime:

Basic object	Runtime Advanced and Panels	Runtime Professional
Line	Yes	Yes
Polyline	Yes	Yes
Ellipse	Yes	Yes
Circle	Yes	Yes
Ellipse segment	No	Yes
Circle segment	No	Yes
Elliptical arc	No	Yes
Circular arc	No	Yes
Rectangle	Yes	Yes
Polygon	Yes	Yes
Text field	Yes	Yes
I/O field	Yes	Yes
Symbolic I/O field	Yes	No
Date/time field	Yes	No
Graphic I/O field	Yes	No
Alarm view	No	Yes
Recipe view	No	Yes
Table view	No	Yes
f(t) trend view	No	Yes
f(x) trend view	No	Yes
Page number	Yes	Yes
Report name	No	Yes
Project name	No	Yes
Hardcopy	No	Yes
ODBC database field	No	Yes
ODBC database table	No	Yes
CSV provider tables	No	Yes
CSV provider trends	No	Yes
Alarm report	Yes	No

Basic object	Runtime Advanced and Panels	Runtime Professional
Recipe report	Yes	No
Control data	No	Yes
Control hard copy	No	Yes
Control Information	No	Yes

See also

Screen objects (Page 2067)

14.7.3.16 Text and graphics lists

Differences between Runtime Advanced and Panels and Runtime Professional

Ranges in text lists

When specifying a range the following settings are supported in both Runtimes:

- Individual value
- Range

The following settings are supported in Runtime Professional:

- To (<=)
- From (>=)

See also

Copying text lists and graphic lists (Page 2088)

Basics (Page 2064)

Copying text lists and graphic lists

Introduction

You can copy a text or graphics list in the "Text and graphics lists" editor and paste it to the same editor in another runtime. Text and graphics lists are also automatically copied along with symbolic and graphic I/O fields and displayed in the "Text and graphics lists" editor.

List entries

When a text or graphics list is copied, the corresponding list entries are included in copying. You can also copy and paste the list entries individually.

See also

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2088)

14.7.3.17 User management

Differences between Runtime Advanced and Panels and Runtime Professional

Users

Properties for dynamic logon and web client administration are only supported in Runtime Professional.

User groups

The "DisplayName" and "Number" properties are only supported in Runtime Advanced and Panels.

Properties for web client administration are only supported in Runtime Professional.

See also

Copying users, user groups and authorizations (Page 2089)

Basics (Page 2055)

Copying users, user groups and authorizations

Copying and pasting

You can copy users, user groups and authorizations in the "User administration" editor.

When you copy a user, the corresponding user group is not copied.

When you copy a user group, the corresponding authorizations are not copied.

Default entries and system authorizations

The following users, user groups and authorizations are predefined. They cannot be deleted and may only be edited to a limited extent:

- predefined user "Administrator"
- predefined groups "Administrator" and "Operators"
- predefined authorizations, such as "Operate" and "Enable remote control"

You can copy the predefined entries and paste them into another Runtime. The entries copied are pasted as user-defined entries and can then be edited.

See also

Differences between Runtime Advanced and Panels and Runtime Professional (Page 2089)

14.7.3.18 Scheduled tasks

Copying tasks

Introduction

You can copy a scheduled task in the "Scheduler" editor and paste it to the "Scheduler" editor of another device.

Task type

The "Print job" task type is supported in Runtime Professional only. Types that are not supported in both Runtimes become invalid after they have been pasted.

See also

Events for scheduled tasks (Page 2090)

Basics (Page 2064)

System functions (Page 2065)

Events for scheduled tasks

Events

The following table shows which events are supported in which Runtime:

Event	Runtime Advanced and Panels	Runtime Professional
Screen change	Yes	No
User change	Yes	No
Alarm buffer overflow	Yes	No
Runtime stop	Yes	Yes
When a dialog is opened	Yes	No
When a dialog is closed	Yes	No
Disabled	Yes	Yes
Once	Yes	Yes
250 milliseconds	No	Yes
500 milliseconds	No	Yes
1 second	No	Yes
2 seconds	No	Yes
5 seconds	No	Yes

Event	Runtime Advanced and Panels	Runtime Professional
10 seconds	No	Yes
1 minute	Yes	Yes
5 minutes	No	Yes
10 minutes	No	Yes
1 hour	Yes	Yes
Once daily	Yes	Yes
Once weekly	Yes	Yes
Once monthly	Yes	Yes
Once yearly	Yes	Yes
Tag triggers	No	Yes

See also

Screen objects (Page 2067)

14.7.4 Copying between different RT and ES versions (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can copy and paste project data such as screens, objects or tags between projects with different WinCC versions.

All configurations that are supported in the target version are retained when you copy data between projects of different WinCC versions. Configurations that are not supported by the target version are marked as invalid with a color code.

Copy of control between HMI devices from different device versions

WinCC supports all configurations of a previous WinCC version.

The following applies when copying within different ES versions:

- All the configurations that are also supported in the respective RT version are retained.
- Default settings are defined for configurations that are only supported in the WinCC version of the target project.

- Configurations that are not supported by the respective RT version are marked by color as invalid or are not displayed.
  All properties and settings originally defined in the source HMI device are reactivated when you copy an object back to the source HMI device unchanged.

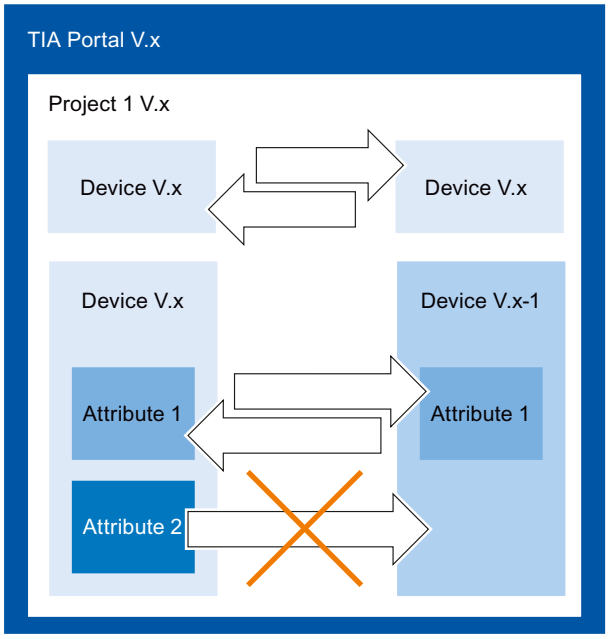

- The HMI device must be valid for the current Runtime version.

Copying between different ES versions

To copy between two TIA projects, open a second instance of the TIA Portal. You can only copy between projects that have the same ES version. The ES version of a project is indicated by the file extension *.ap<version_number>.

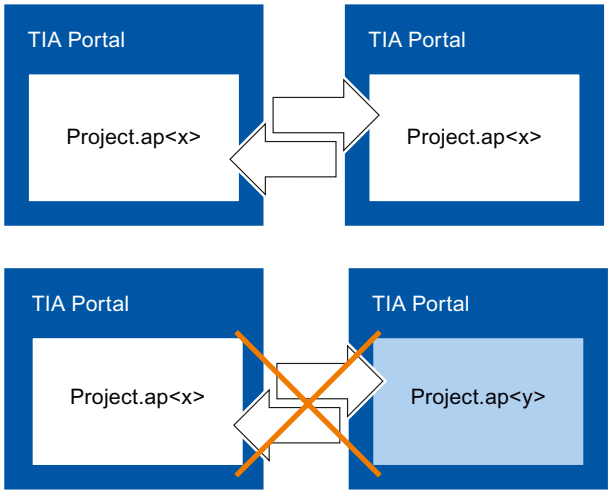

14.8 Using WinCC version compatibility

14.8.1 Basics on version compatibility (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Edit existing projects as follows with WinCC:

- You edit, compile and download existing projects with the range of functions of the previous version of WinCC. You can continue to edit these projects afterwards with the previous version of WinCC.
- You upgrade existing projects and use the functions of the current WinCC version.

Note

WinCC functions

While editing a project of a previous WinCC version, you can only access the functions and HMI devices of this previous WinCC version.

Versions in WinCC

In WinCC, you work with different version types:

- WinCC version
  The WinCC version installed on the configuration PC, for example, WinCC V12.
- Project version
  Projects are created using the WinCC version that is installed on the configuration PC. While you are editing a WinCC project from a previous version in the current WinCC version, the version ID is displayed behind the project name in the project tree.
- Runtime version
  You can configure HMI devices with different runtime versions in WinCC. You specify the runtime version once for an HMI device. The device version must match the Runtime version.
- Device versions
  Depending on the used HMI device, the image is a combination of the operating system and / or runtime software. For each HMI device, WinCC provides various images which can be loaded onto the HMI device, if necessary, according to the configuration. The device version corresponds to a specific image. The device version must match the configuration.

Compatibility of WinCC versions, Runtime versions and device versions

The figure below shows the interaction of the versions in the TIA Portal:

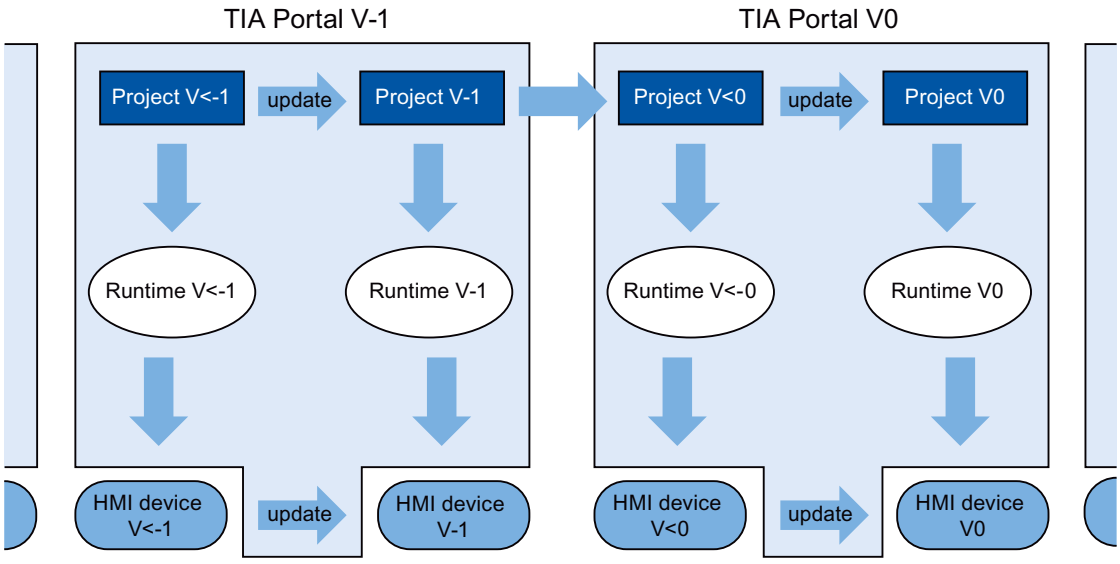

Creating projects

After you create a new project in WinCC, open and edit it in the WinCC version in which you created it.

Saving

To save a project of a predecessor WinCC version in this version again, save it in the usual way. If you manually upgrade the project to your WinCC version, please note that you will no longer be able to open the project in the previous WinCC version.

To save a project of the previous WinCC version in the current version, upgrade the project to your WinCC version. You can then no longer edit the project with a previous version of WinCC.

Compiling, simulating and loading

If you are using a project of a previous WinCC version, you can use your current version to generate Runtime data for this previous version. This also allows you to load HMI devices that are no longer compatible with your WinCC version.

Copying within projects with different WinCC versions

If objects and configurations are also available in the target version, copy these as required via the clipboard or using drag-and-drop.

Opening, editing and saving projects of a previous WinCC version

You can open and edit projects of previous WinCC versions as required. In doing so, you only utilize the functions of the previous WinCC version. Once you have completed editing, you can once again save and edit the project in the previous WinCC version.

Compiling, downloading and simulating projects of a previous WinCC version

You can compile, download and simulate projects of previous WinCC versions as required. Your current WinCC version will provide the Runtimes and device versions for the corresponding WinCC version.

14.8.2 Basic information on version compatibility (RT Professional)

Introduction

In the current version of WinCC, you can open and upgrade projects from the previous WinCC version.

Versions in WinCC

In WinCC, you work with different version types:

- WinCC version
  The WinCC version installed on the configuration PC, for example, WinCC V14.
- Project version
  Projects are created using the WinCC version that is installed on the configuration PC. While you open a WinCC project from a previous version in the current WinCC version, the version ID is displayed after the project name in the project tree.
- Runtime version
  The Runtime project is generated for a specific Runtime version, depending on the version of the project. The device version must match the Runtime version.

Compatibility of WinCC versions

You can only open and upgrade projects from a previous WinCC version in the current WinCC version. To configure the project of a WinCC predecessor version for the corresponding Runtime version, use the appropriate WinCC version on your configuration PC.

Compiling, simulating and loading

You can only compile, simulate and load a project of a WinCC predecessor version in the current WinCC version if you upgrade the project to your WinCC version.

To compile, simulate and load the project of a WinCC predecessor version for the corresponding Runtime version, use the appropriate WinCC version on your configuration PC.

See also

Upgrading projects (Page 2097)
Changing the device version (Page 2102)
Changing between device versions (Page 2101)
Copying between different RT and ES versions (Page 2091)

14.8.3 Upgrading projects (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

If the project version is older than the WinCC version, the version ID is displayed in the project navigation. Your WinCC version also contains the previous version that you can use to edit projects as required. In order to use the functions and options of your WinCC version in a project, upgrade the project to your WinCC version. Then change your device or Runtime version(s) to match the new project version.

Note

WinCC version compatibility

If you upgrade a project to your WinCC version, please note that you will no longer be able to edit the project with the previous version.

Requirements

- The project version is a predecessor of your WinCC version.
- You have write access to your project drive.
- The project drive provides sufficient storage capacity for another project of this size.

Procedure

Proceed as follows to upgrade a project to your WinCC version:

1. Select the project in the project navigation.
2. Select the "Upgrade project" command from the shortcut menu of the project.
   A dialog opens.

3. Click "Confirm".
   The project closes and the progress bar is displayed.

   Note

   If devices that are not supported in the project are configured, the project is not upgraded. An error message will inform you of this.

   To nevertheless upgrade a project, change the device version of the affected device or the device itself with the menu command "Change device/version".

   An alarm is output when the project has been upgraded.

4. Then compile each device in the project with the command "Edit > Compile".

   Note
   Sequence of compilation

   Compile the controllers first, and then the HMI devices in your project. In this way, you ensure that the controller data required to compile the HMI devices is available.

Result

- The project has been saved on the project drive in the previous WinCC version and with the corresponding file extension.
- The project is displayed on the project drive in the current WinCC version and with the corresponding file extension.
- The project is displayed without a WinCC version ID in the project navigation.

To make use of the functions of the new WinCC version, change your device or Runtime version(s) in accordance with the new project version in the next step.

See also

Basic information on version compatibility (Page 2096)

14.8.4 Upgrading a global library (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

In order to process objects of a global library in a project, the global library must be available in the product version of the project. You can upgrade each global library from an older product version to the current product version. You are prompted accordingly when you open the global library.

Requirements

- The version of the global library is a predecessor of your WinCC version.
- You have write access to your project drive.
- All types in the library have been released.

Note

Upgrading a user library

If you want to use a user library from an earlier version of WinCC, you must upgrade it. Make sure that all types of the library have been released. The library being upgraded cannot contain a type with the "in progress" status.

Procedure

Proceed as follows to upgrade a global library from TIA Portal V12.x or lower:

1. Open the global library.
   The "Upgrade global library" dialog box opens.
2. Click "OK".

A copy of the global library is created and upgraded. The copy of the global library receives the name extension "_V13". The global library opens.

Result

The global library is stored with the appropriate file extension.

See also

Basic information on version compatibility (Page 2096)

Managing libraries (Page 1941)

14.8.5 Editing projects of a previous WinCC version (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

WinCC provides the option of editing projects of a previous WinCC version. While editing a project of a previous WinCC version, you can only access the functions of this version. In order to use the functions of your current WinCC version for this project, upgrade the project to your WinCC version.

Note

When you upgrade a project, the content is saved in a new project with the current project version. The original project is not overwritten and can still be edited with a compatible older version of the TIA Portal.

Requirement

- A project of a previous WinCC version has been created.
- The current WinCC version is installed on the configuration PC.

Procedure

Proceed as follows to edit a project of a previous WinCC version:

1. Open the project.
2. Edit the project using the functions of the previous WinCC version.
3. Save the project.
4. Compile the project.
5. Download and simulate the project.
6. You can open the project in the previous WinCC version for further editing, if required.

Result

The modified project data can be edited further on a different configuration PC with the previous WinCC version for further processing. The Runtime project was generated and downloaded in the corresponding Runtime version.

14.8.6 Changing between device versions (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Selection of the device version

When you configure a new HMI device, WinCC automatically selects the latest version of the device.

If you want to use a device version other than the one set in WinCC, transfer an image to the HMI device. WinCC provides the images required for the supported HMI devices.

Information on the device versions used in WinCC is available in the FAQs on the Internet, entry ID 21742389.

NOTICE
Changing the device version deletes all data on the HMI device. Data is deleted on the target system if you change the device version. For this reason, you should save existing Runtime data before changing the device version.

HMI device configuration

The following figure shows the software components of an HMI device:

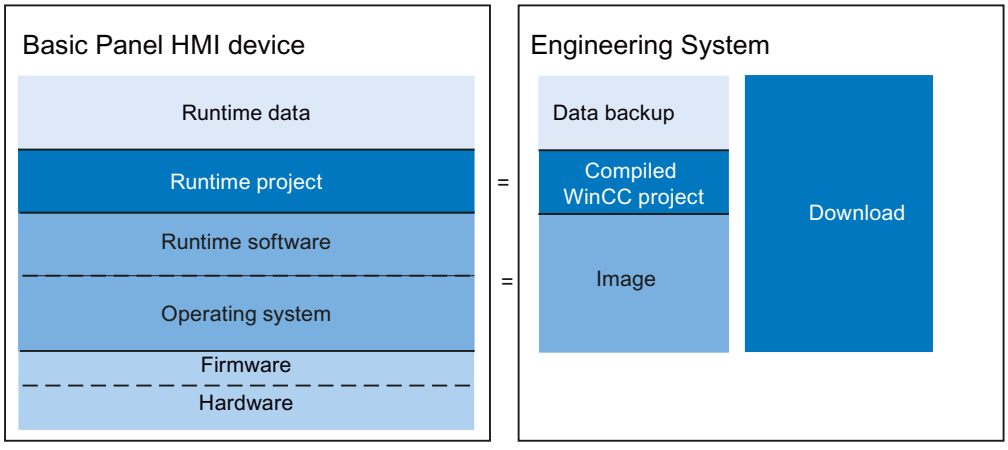

See also

Basic information on version compatibility (Page 2096)

14.8.7 Changing the device version (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

Depending on the required Runtime version, select the suitable device version for your configuration.

Note

Selection of device versions

The selection of available devices versions depends on the version of the project.

Requirements

- A project has been created and opened.
- The project contains an HMI device.

Procedure

To change the device version, follow these steps:

1. Double-click on "Devices & Networks" in the project tree.
   The editor opens.
2. Select the required HMI device from the device view.
3. Select "Change Device/Version" in the device shortcut menu of the HMI device.
   A dialog opens.
4. Select the required HMI device.
5. Select the required device version under "Version".
6. Confirm your selection with "OK".

Result

You have changed the device version in the WinCC project.

NOTICE
Changing the device version deletes all data on the HMI device. All data is deleted from the HMI device when you change the device version and compile/download the project. You should therefore backup your Runtime data prior to the download.

See also

Basic information on version compatibility (Page 2096)

14.9 Managing colors centrally

14.9.1 Basic principles for color management

Introduction

WinCC provides the option of changing the colors used in a project centrally. The "Change color reference" dialog box contains a hierarchical overview of all color-relevant properties contained in the selected object. In the display, you can navigate within the display and operating objects that are shown and thus get an overview of all colors used. Using search and filter you can specify a color selection and replace it with other colors, if necessary.

Supported objects

You have access to all colors used and configured in the project with the "Change color reference" dialog. Excluded from this are colors that are used as follows:

- In types and instances from a library
- In faceplates
- In scripts
- In designs
- In screens with write protection

14.9.2 Finding and replacing colors

Introduction

The scope of the objects displayed in the "Change color references" dialog depends on the location at which the dialog is called.

- If you select an HMI device and call the dialog, all color references used on the HMI device are displayed.
- If you select a display object within a screen and call the dialog, only those color references that are included in the display object are displayed.

Requirement

- You have created a project.
- Screens have been created.

Procedure

1. Select the object that contains the required color references.
2. Select "Change object color" in the shortcut menu.
   A dialog opens.
3. Select the color that you want to change.
   - Click the color field in the search box.
     The project color selection opens.
   - To select a standard color or a user-defined color, click "More colors".
   - To use a color directly from the selected object, drag a color field from the overview table to the search box.
   - To select similar colors as well, set a tolerance.

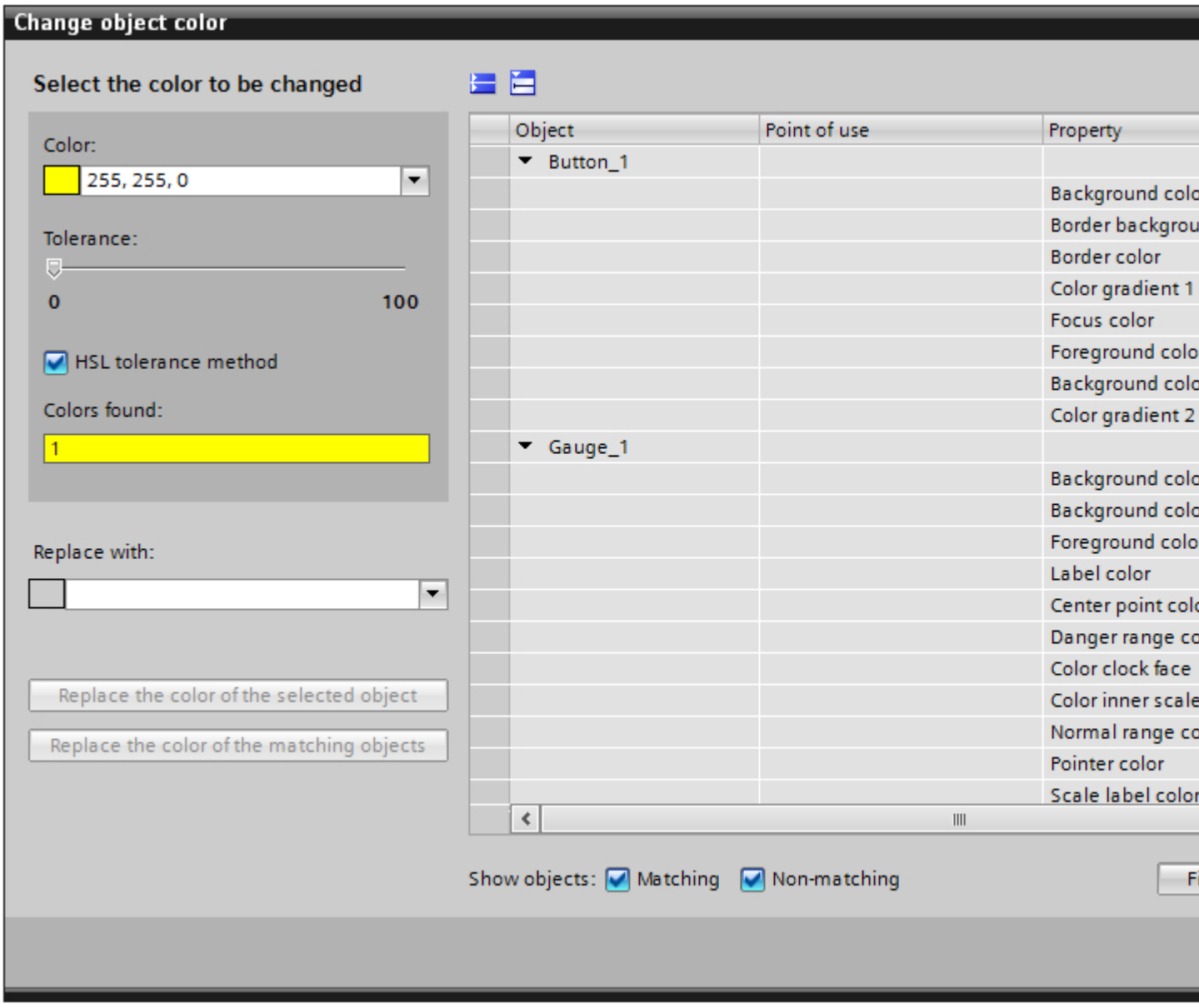

4. Filter the displayed table.
5. In the table column "Replace with" select the new color for the individual properties.
6. Click "OK".

Result

The new color references are configured in the selected object.

14.10 Load files using the "HMI files" editor

Introduction

You upload files of any data format from your computer to your project using the "HMI files" editor. These files are transferred to the PC during "Download to device". This way you link useful additional information such as system manuals, video tutorials, etc. with the respective project or distribute them to different devices. If required, you overwrite the downloaded files at any time with current files or delete them from the project.

Create folders to design a clear work area in the "HMI files" editor. Then you save the files in the folders.

If you modify these files or folders on your computer, the changes also appear in the "HMI files" editor.

Note

The "HMI files" editor is only visible in the project tree when a device is created with Runtime Professional in the project.

Requirements

The files that you want to download to the project are located on your computer.

Downloading files

1. Double-click "Common data" in the project tree.
2. Double-click the "HMI files" editor.
   The editor opens.

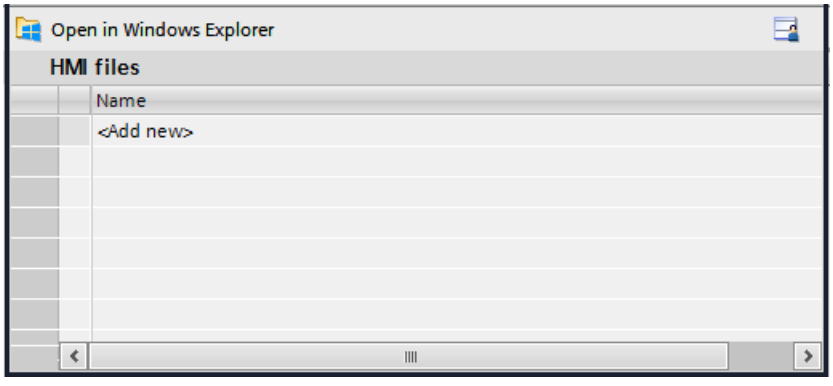

3. Click "Add new" in the work area.
   The Windows Explorer opens.
4. Navigate to the directory in which the files are stored that you want to download to the project.

5. Select one or more files.
6. Click "Open".
   The files appear in the work area numbered in alphabetical order and are saved under "<TIAProjectname>\UserFiles\HMI_globalFiles".

You can also open the Windows Explorer using the "Open in Windows Explorer" button and move the required files from the Windows Explorer into the work area with drag&drop.

Note

To go directly to the storage folder "<TIAProjectname>\UserFiles\HMI_globalFiles", click on the button "Open in Windows Explorer".

Overwriting files

You overwrite files by downloading files with the same name into the project folder.

1. You download files as described under "Downloading files".
2. Confirm the prompt with "Yes" for the files to be overwritten.

Deleting files

1. Select the line of the file to be deleted in the work area.
2. Select "Delete" from the shortcut menu.

Note

You delete multiple files by selecting several lines and executing the "Delete" command from the shortcut menu.

Creating folders

You must first create folders and then fill them with files.

1. Select an empty line in the work area.
2. Select "Create new folder" from the shortcut menu.
   The "Create new folder" dialog opens.
3. Enter a name for the folder.
4. Click "OK" to confirm your input.
   The folder appears in the work area.

Note

You create subfolders by selecting the line with the respective folder and executing the command "Create new folder" from the shortcut menu.

Saving files in folders

1. In the work area, select a folder that you want to fill with files.
2. Select "Insert object" from the shortcut menu.
   The Windows Explorer opens.
3. You download files as described under "Downloading files".
   The files are stored in the folder.

Result

The files and folders as well as their changes are transferred to all devices in the project after the following actions:

- Compilation of the project
- Downloading the project
- Starting runtime
- Starting the simulation

The contents of the "<TIAProjectname>\UserFiles\HMI_globalFiles" folder is mirrored under "<RTProject>\AdditionalFiles".

Note

When a file is being used in runtime, this file is not overwritten in the "<RTProject>\AdditionalFiles" folder. A warning informs you about it.

See also

Overview of compiling and loading projects (Page 2272)

14.11 Exchanging controller data from other projects

Inter-project engineering

You can use the IPE (Inter Project Engineering) function to read in controller data from other TIA Portal projects and use it for configuring.

IPE enables you to transfer and merge the HMI configuration and PLC programming communication into various TIA Portal projects.

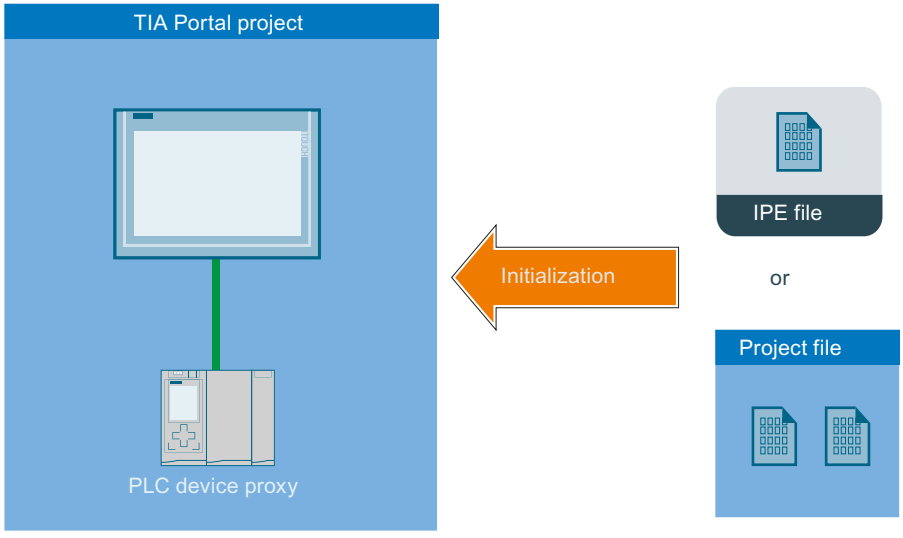

You can find more information and notes on configuration under: Basics of Inter Project Engineering (IPE)

14.12 Recognizing the project status by the SIMATIC WinCC RT symbol

Introduction

WinCC displays the "SIMATIC WinCC RT" symbol in the notification area of the taskbar. This symbol provides information about the Runtime status.

Project status

The following table shows which project status goes with which symbol:

Symbol	Status
	WinCC runtime is started.
	WinCC Runtime is not active. No Runtime project is open.
	WinCC Runtime is active.
	WinCC Runtime is active and the server has the "Fault" status.
	WinCC changes the status.
	WinCC opens a Runtime project.
	WinCC activates a Runtime project.
	WinCC deactivates a Runtime project.
	WinCC closes a Runtime project.

Control Options via the Pop-up Menu

You can operate an open Runtime project depending on its status. You have the following operating options from the shortcut menu of the "SIMATIC WinCC RT" symbol:

- Enable Runtime / Disable Runtime
- Exit Graphics Runtime / Start Graphics Runtime
- Open diagnostics window
- Runtime start options
- WinCC license analysis

When you click on the "SIMATIC WinCC RT" symbol, you can see the current project status and, in the case of a redundant project, the status of the partner server or client.

Compiling and loading 15

15.1 Establishing a connection to the HMI device

Introduction

To download a WinCC project to an HMI device, a properly configured connection must be set up between the configuration PC and HMI device. The connection cannot be set up, the download is cancelled.

Setting up a connection between the configuration PC and HMI device

1. Check the cable connection between the HMI device and configuration PC.
2. Open the "Devices & Networks" editor in WinCC and start the network view.
3. Select the subnet in the network view and check the settings for the subnet.
4. Select the interface of the HMI device in the network view or device view and check the connection parameters in the Inspector window.
5. Switch on the HMI device and press the "Control Panel" button in the loader.
   The Control Panel opens.
6. Press "Transfer" twice in the Control Panel.
   The "Transfer Settings" dialog box opens.
7. Check the settings and then press "Advanced".
   The [Protocol*] Settings" dialog opens.
   *: The title of the dialog depends on the protocol used, for example, "PROFIBUS Settings".
8. Check the advanced settings and close the dialog with "OK".

Important settings

Check the connection settings and in particular the following parameters:

- Network and station addresses
- Selected transmission rate
- Master on the bus; as a general rule, only one master is permitted.

Note

Connections over Ethernet

For Ethernet-based communication, check the required network settings. You can find additional information under "Readme > WinCC > Security information".

If using a configurable adapter for the connection, check the adapter settings, for example, transmission rate, master on the bus.

See also

Loading a project (Page 2184)

15.2 Basic Panels

15.2.1 Runtime settings

15.2.1.1 Runtime settings

Introduction

To edit the runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree.

Note

Device dependency

The options available for selection in the runtime settings depend on the HMI device.

Overview

You configure options from different areas of your project in the runtime settings:

- Display in runtime
  - Start screen
  - Default template
  - Default style of the project and style of the HMI device
  - Screen resolution
  - Full-screen mode
  - Project graphics
  - Bit selection for color and flashing
  - Bit selection for text and graphic lists
- Project ID for compatibility check with the controller
- Logging language
- Remote control via Sm@rt server in the "Services" area
  You can find additional information on this in the WinCC online help under "Options > Sm@rt options"

- Operator control in runtime
  - Task switching
  - Load additional information for debugging scripts: Object names and comments in scripts
  - Show limits as a tooltip
  - Screen keyboard
  - Disable function keys
  - Release buttons when cursor leaves the button
  - Keyboard layout and shortcuts
- GMP-compliant configuration
  You can find additional information on this in the WinCC online help under "Options > WinCC Audit".
- Alarms
  - Buffer overflow
  - Acknowledgment group text
  - Log
  - Colors of the alarm classes
  - Configuration of the system events
  - Persistent alarm buffer
- User administration
  - Configuration of the password and logoff time
  - Limit for logon attempts
  - Hierarchy levels
  - Configuration of SIMATIC Logon
- Language and font
  - Available runtime languages
  - Order of language switching
  - Default font
  - Download font with valid license
- Configuration of the OPC Unified Architecture Server
- Settings for synchronization of PLC and HMI tags

See also

Configuring display in Runtime (Page 2117)

Configuring operation in Runtime (Page 2118)

Compatibility check with the controller (Page 2120)

Editing system events (Page 802)

Creating alarm classes (Page 785)

Configuring alarm buffer overflow (Page 891)

Settings for synchronization (Page 544)

Settings for the user administration (Page 1316)

Managing languages (Page 2012)

Working with Logs (Page 1027)

15.2.1.2 Configuring display in Runtime (Comfort Panels)

Introduction

In the Runtime settings you specify how screens, alarms and language are displayed.

To edit the Runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree. The options available for selection depend on the HMI device.

Display on the target system

In WinCC, configure the visual representation of the generated project in Runtime:

- Select the full-screen or window mode for the project. The window has a smaller size than the screen. In full-screen mode, the project is zoomed to the full screen. There will be no window and operator controls for this view.
  Click "General" in the "Settings" editor to activate the full-screen display at startup. Activate the "Full-screen mode" check box in the "Screen" field.

  Note

  Truncated view

  If the HMI screen does not match the configured size (in pixels), the project only appears on part of the screen in full-screen mode.

- Under "General > Screen", you specify whether the HMI device uses the default style of the project or another pre-defined style.

Screen settings

In the Runtime settings, specify the start screen and default template. You can find additional information in the online help for WinCC under "Working with screens".

Language settings

- Logging language
  Under "General > Logs", you specify the language in which you want to write the logs. The selected logging language always remains the same, even if the user switches the language in Runtime.
  If you select "Startup language", the logging language remains the same until Runtime is stopped, even if the user has switched the language. However, the next time Runtime is started, the last configured Runtime language is used as the logging language.
- Runtime language
  Under "Language & font", you specify all Runtime languages and their display. Text entries may be truncated if you selected an unsuitable font size or style. This setting can, for example, affect the following texts:
  - Tooltips
  - Long alarm text
  - Text in dialogs

Setting for specific screen objects

- Specify bit selection
  Under "Screens > Bit selection", you specify whether bit selection will be used on this HMI device for the following:
  - Text and graphics (text lists and graphic lists)
  - Color and flashing (appearance analysis)
    When you enable an option, the selection that is configured in the set bit with the least significance is displayed:
    When you disable an option and several bits are set, the selection that has been configured only for the set bit is displayed.
- Tooltips for I/O fields
  Under "Screens > Display", you specify whether configured limit values are displayed as tooltips in Runtime when values are entered in I/O fields.
- Colored objects
  You specify the color depth of your HMI device under "General > Screens > Color depth". As needed, you assign screen objects a colored appearance. The range of colors is determined by the color depth supported on your selected HMI device.

15.2.1.3 Configuring operation in Runtime

Introduction

In the Runtime settings you specify how screens are operated, and how you use keys for operation in Runtime.

To edit the Runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree. The options available for selection depend on the HMI device.

Locking task switching

Depending on the HMI device, you can lock task switching on the HMI device. This will prevent the operator from opening other applications in Runtime.

In the Runtime settings of the HMI device, enable the "Lock task switching" option under "General > Screen".

Note

Stop Runtime

If you lock program switching, always configure in your project the system function "StopRuntime" for an object, e.g. a softkey or button.

Settings for data transfer to the HMI device

- Transferring names of the screen objects in scripts
  If you enable the "Load names" option under "General > Screen", the object names of the screen objects are transferred instead of coded addressing information. You need the object names, if you want to address the screen objects in a script via the object names. When you test a script in the debugger, the code becomes more comprehensible by displaying the object names.
- Displaying comments in scripts
  Under "Screens > Display", you specify whether the comments in scripts are also transferred to the HMI device.
  If you do not transfer the comments, you save storage space on your HMI device.
  If you transfer the comments, the script is more easily understood during testing in the debugger.

Operation using function keys

To symbolize the function of a function key for keyboard devices, you configure a graphic next to the function key in the display. Under "Screens > Function keys > User-defined pictogram size", you specify whether you want to use a graphic size that deviates from the standard size.

Operation with keyboard

Under "Keyboard > General > Use screen keyboard", you specify whether the screen keyboard is available on your HMI device.

Under "Keyboard > General > Release button on exit", you specify whether the "Release" event is triggered when the user exits a button without releasing it.

Under "Keyboard > Disable function keys in dialogs", you specify for keyboard HMI devices whether the function keys are disabled for the duration of the displayed dialog. Large dialogs can hide explanatory areas or graphics at function keys or that border function keys. Functions keys can be unintentionally pressed due to this.

15.2.1.4 Compatibility check with the controller (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)

Introduction

The "Project ID" area pointer is used to configure the project ID for checking the consistency of the project with the PLC. The area pointer reads the project version from the PLC. You can find additional information in the online help for WinCC under "Communicating with controllers".

Compatibility check with the controller

You select the project ID in the Runtime settings of the HMI device under "General > Identification > Project ID".

At startup, Runtime checks whether the project ID of the project matches the PLC project version. Runtime will only start if the two values match.

An area pointer "Project ID" is created for connections whose address you specify in the PLC. The value that you store at this address will become the project ID.

The project ID is not available on all HMI devices.

15.2.1.5 Password protection in Runtime

Password protection in Runtime: Basics (Basic Panels)

Overview

Connection establishment in the TIA Portal can be password-protected. The configured passwords are saved in encrypted form. When transferred to the target system, the passwords are also protected from unauthorized access and cannot be read. The passwords are securely decrypted on the HMI devices in runtime.

Password protection can be set up for the following connections:

- SIMATIC S7-1200
- SIMATIC S7-1500
- SIMATIC S7-1500 software controller (WinAC)
- PC Station 2.0 (station manager)

Note

The passwords become invalid if you switch to an older version.

Downloading certificates

The certificate for decrypting the passwords is saved in the certificate store of the HMI device. A system alarm is output if the certificate is not available.

You can restore the certificate by downloading the configuration again.

See also

Configuring a password for transport (Page 2121)

Resetting the password for transport (Page 2122)

Configuring a password for transport (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

To protect the private password key during transport, you can configure a password for transport under "Runtime settings".

Creating a password

1. Open the "Runtime settings > General" editor in the Project window.
2. Enter the password in the "Enter password" box in the "Transport password" area.
3. Confirm the password in the "Confirm password" box.

Note

If there was no previous password, the "Old password" box is disabled.

Result

You have now configured the password for transport. If a password for transport has been configured and you transfer the project to the HMI device, a dialog with a password prompt will appear once in runtime.

If the password entered is correct, the certificate is saved in the certificate store of the HMI device.

When you change the password, load the project with the changed password into the HMI device and start Runtime. No password prompt appears.

See also

Password protection in Runtime: Basics (Page 2120)

Resetting the password for transport (Page 2122)

Resetting the password for transport (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

If you have moved the transport password, for example, you can create a new security certificate. To do so, reset the transport password.

Resetting the password

1. Click the "Reset password" button.
   The password is reset. A new certificate without a password is generated.

   Note

   Other passwords, for example the password for the PLC, will become invalid. The next time you compile, an error message will appear telling you that these passwords are no longer valid.

2. Enter the new password and confirm it in the "Confirm password" field.

   Note

   The certificate is not password-protected until you enter a new password for transport.

   After download of the project to the HMI device, you will be prompted for the password.

See also

Password protection in Runtime: Basics (Page 2120)

Configuring a password for transport (Page 2121)

15.2.2 Overview of compiling and loading projects (RT Advanced, Panels, Comfort Panels)

Overview

The project is compiled in the background even as you are configuring it in WinCC. This reduces the time for final compilation. When you start compilation, you create a file that can be run on the corresponding HMI device.

If an error occurs during compilation, WinCC provides support in locating and correcting it.

Once you have corrected any problems, you download the compiled project to the HMI devices on which the project is to run. If the configuration PC is not connected to the HMI device, save the compiled project on a data medium of your choice. The compiled project is then transferred from a PC connected to the HMI device to the HMI device.

If you are using HMI tags in your project that are connected to PLC tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Project

The term "project" has two different meanings in the contexts of compilation and loading. "Project" is the WinCC project on the configuration PC. "Project" is also the Runtime project you create by compiling the configuration data of an HMI device and download to the HMI device.

- WinCC project: contains the configuration data of one or more HMI devices
- Runtime project: contains the compiled configuration data of an HMI device

The figure below illustrates the link between WinCC projects and Runtime projects using the example of the "Compile and load" process:

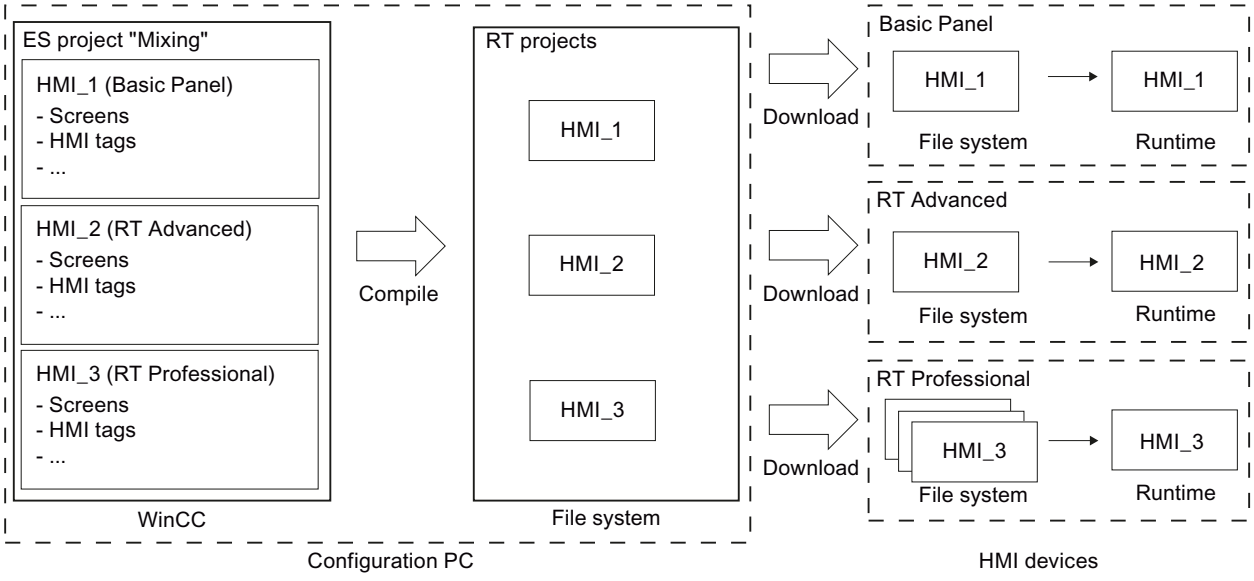

Runtime

Runtime is the software for process visualization. In Runtime, you execute the project in process mode.

A distinction is made between two types of Runtime:

1. Runtime on a panel
   Before running a Runtime project on a panel, you have to transfer the Runtime project to the panel before startup.
2. Runtime on a PC
   You can execute the Runtime project directly on the configuration PC if Runtime has been installed on the configuration PC.
   If you want to execute the Runtime project on a different PC, you have to transfer the Runtime project to the PC before startup.

Runtime version

The Runtime version depends on the image of the configured HMI device. The Runtime version of the compiled project is displayed under "Info" in the Inspector window.

Simulation

You test your configuration with a simulation. You can start a simulation without a link to the active process.

In a simulation, you test configured tags or screen changes, for example. During the simulation, the configured tags can be manipulated, activated and deactivated with the help of the tag simulator.

There are two types of simulation:

1. Simulating a panel
   If you created a panel in your project, the panel is displayed in the simulation. With the help of this type of simulation, you can test your configuration on the HMI device without transferring the project to the panel.
2. Simulating Runtime
   Simulating Runtime allows you to test the project directly on the configuration PC.

15.2.3 Compiling a project (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The changes made to the project are compiled in the background even as you are configuring a project in WinCC. Projects are compiled automatically when you load them. This ensures that the latest version of the project is loaded at all times.

WinCC checks consistency of the project during compilation. The error locations in the project are listed in the Inspector window. You can jump directly to the source of the error from the entry in the Inspector window. Check and correct errors found.

Scope of the compilation

Configuration data is compiled in the background as soon as you start configuring an HMI device. If you compile a project manually, only the changes in the configuration made since the last compilation process are compiled in the background.

You can start complete project compilation manually at any time; this may, for example, be done to test the consistency of the configured data.

Requirement

- A project is open.

Procedure

Proceed as follows to compile a project:

1. If you want to compile several HMI devices at the same time, select all the relevant HMI devices with multiple selection in the project tree.
2. Compile the project:
   - To only compile changes in the project, select the "Compile > Software (only changes)" command from the shortcut menu of the HMI device.
   - To compile all project data, select the "Compile > Software (compile all)" command from the shortcut menu.

Result

The configuration data of all selected HMI devices is compiled. Any errors that occur during compilation are shown in the Inspector window.

15.2.4 Simulating projects

15.2.4.1 Simulation basics

Introduction

You can use the simulator to test the performance of your configuration on the configuration PC. This allows you to quickly locate any logical configuration errors before productive operation.

You can start the simulator as follows:

- In the shortcut menu of the HMI device, or in a screen: "Start simulation"
- Menu command "Online > Simulation > [Start|With tag simulator|With script debugger]"
- Under "Visualization > Simulate device" in the portal view.

Requirements

The simulation/runtime component is installed on the configuration PC.

Field of application

Use the simulator to test various functions of the operator control and monitoring system, such as:

- Checking limit levels and alarm outputs
- Consistency of interrupts
- Configured interrupt simulation

- Configured warnings
- Configured error messages
- Check of status displays
- Interconnection and screen layout

See also

Simulating a project (Page 2126)

15.2.4.2 Simulating a project (RT Advanced, Panels, Comfort Panels)

Introduction

You simulate your project with one of the following two methods:

- Without a connected PLC
  You change the value of area pointers and tags in a tag simulator that is read for the simulation of WinCC Runtime.
- With a connected PLC without a running process
  You simulate your project by running it directly in Runtime. The tags and area pointers become active. This allows you to create an authentic simulation of your configured HMI device in Runtime.

Note

Simulation restrictions

You cannot simulate the following system functions:

- CalibrateTouchScreen

You cannot simulate the Media Player. A static screen appears in the simulation window instead of the Media Player.

File access via scripts is not possible for HMI devices with Windows CE.

Requirement

- Simulation without a connected PLC: Tags have been created
- Simulation with a connected PLC but no active process: A project with tags and area pointers has been created

Procedure

To simulate a project using the tag simulator, follow these steps:

1. Open the project on the configuration PC.
2. Select the "Online > Simulation > With tag simulator" menu command.
   For initial project simulation, the simulator is started with a new, empty table. The project is opened simultaneously in Runtime.
   Toggle between the tag simulator and Runtime using the <Alt +Tab> key combination.
3. To simulate a process value, select the corresponding "tag" from the tag simulator.
   The table lists all configured tags. You can simulate up to 300 tags simultaneously.
4. Select the simulation mode in the "Simulation" column.
5. Change the value of tags and area pointers in the respective columns.
6. Activate the "Start" check box to start the simulation for this tag.
7. To save the simulation, select the menu command "File > Save" and enter a descriptive name, for example, "Mixing".
   The file name is assigned the extension "*.cors".

Result

The process values are simulated in Runtime. The tag values are created at random, or incremented, depending on the simulation mode.

To specify tag values, change the simulation mode to "<Display>" and enter a value at "Set value".

The following figure shows a tag simulator with four tags whose values can be determined at random in a range of values from 10 to 1000:

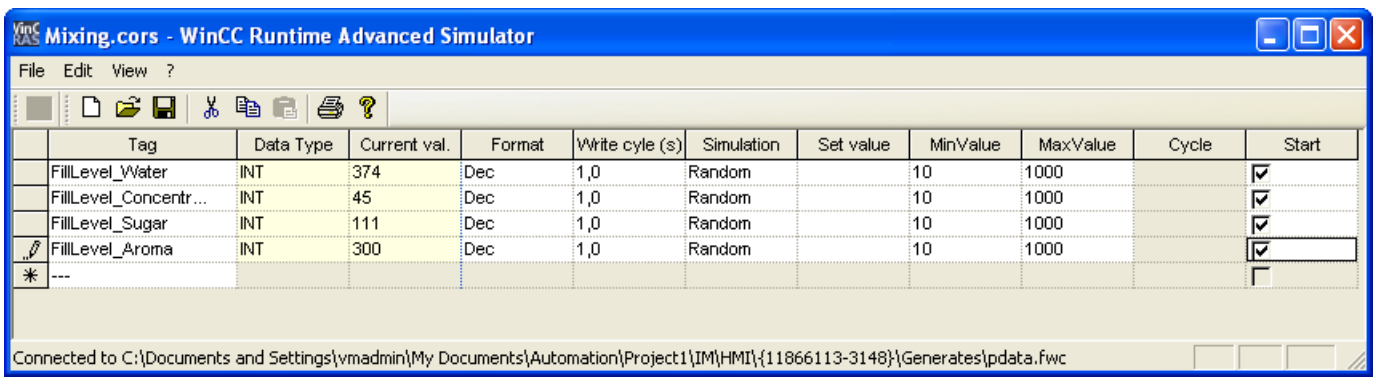

Tag	Data Type	Current val.	Format	Write cyle (s)	Simulation	Set value	MinValue	MaxValue	Cycle	Start
FillLevel_Water	INT	374	Dec	1,0	Random		10	1000		☑
FillLevel_Concentr...	INT	45	Dec	1,0	Random		10	1000		☑
FillLevel_Sugar	INT	111	Dec	1,0	Random		10	1000		☑
FillLevel_Aroma	INT	300	Dec	1,0	Random		10	1000		☑
---										☐

Managing simulation data

If you have saved data from a previous simulation, you can open the file at a later point in time and simulate your project again. The tags and area pointers listed in the tag simulator must still be available in the project.

Proceed as follows to open a simulation file:

1. Select the menu command "Online > Simulate Runtime > With tag simulator".
2. Select the menu command "File > Open" in the tag simulator.
3. Select the corresponding simulation file and click "Open".
   The simulator loads the stored data.

Enabling and disabling tags

Start and stop the simulation for each tag separately in order to facilitate the transition from offline to online engineering. Activate "Start" in the corresponding row.

If a tag is activated, the simulation values are calculated and transferred to the WinCC simulator.

Deleting a tag

To delete a tag from the tag simulator, follow these steps:

1. Select the cell that contains the tag name.
2. Select the "Edit > Cut" menu command.
   The tag is removed from the table.

See also

Simulation basics (Page 2125)

Working with the tag simulator (Page 2128)

15.2.4.3 Working with the tag simulator (RT Advanced, Basic Panels, Panels, Comfort Panels)

About the tag simulator

The tag simulator has the following columns:

Column	Description
Tag	Specifies the tags for the simulation.
Data type	Shows the data type of the selected tag.
Current value	Shows the simulated value of the defined tags.
Format	Specifies the selected format in which the tag values are simulated: • Decimal (1, 2, 3, 4, ...) • Hexadecimal (03CE, 01F3, ...) • Binary (0 and 1)
Write cycle	Specifies the selected time interval at which the current tag values are simulated. If you enter "2", for example, the current value of the tag will be shown every 2 seconds.
Simulation	Shows the method by which the tag values are processed during simulation.

Column	Description
Set value	Sets the selected value for the respective tag. The simulation start with the specified value.
minValue maxValue	Specifies the value range of the tag. You set a minimum and maximum value for this range. The default values are -32768 for the minimum and 32767 for the maximum.
Period	Contains the period during which the value of the tag is repeated for the "Increment" and "Decrement" simulation modes.
Start	Starts simulation of the tag based on the previously entered information.

Simulation modes

The simulator has six different simulation modes. The configured tags are supplied with nearly realistic values during the simulation.

Simulation mode	Description
Sinusoidal	Changes the tag value to form a sinusoidal curve. The value is visualized as a periodic, non-linear function.
Random	Provides randomly generated values. The tag value is changed by means of a random function.
Increment	Increases the value of the tag continuously up to a specified maximum value. Begins again at the minimum after the maximum has been reached. The value trend corresponds to a positive saw-tooth curve.
Decrement	Reduces the value of the tag continuously down to a specified minimum value. Begins again at the maximum after the minimum has been reached. The value curve corresponds to a negative saw-tooth curve.
Shift bit	Shifts a set bit continuously by one position. The previous position is always reset. This lets you test the alarms of an HMI device, for example.
<Display>	The current tag value is displayed statically.

Example: Simulate tags with the "Shift bit" simulation mode

Proceed as follows to simulate tags with the "Shift bit" simulation mode:

1. Open the project you want to simulate.
2. Select the menu command "Online > Simulate Runtime > With tag simulator".
   The tag simulator opens.
3. In the "Tag" column, select a tag from your project.
4. Select "Bin" in the "Format" column.
5. Enter the value "1" in the "Write cycle" column.
6. Select the "Shift bit" simulation mode in the "Simulation" column.
7. Enter the value "1" in the "Set value" column.
8. Enable the tag with the "Start" check box.

Result

The simulator tests the selected tag bit-by-bit as follows:

Simulation values	Byte for alarms
Set start value	00000001
1. Simulation value	00000010
2. Simulation value	00000100
3. Simulation value	00001000
....	...

In Runtime you see if the desired alarm is output at a given value.

15.2.4.4 Simulation restrictions

Alarms with dynamic parameters

If you use tags or text lists as external tags for alarms, the dynamic parameters of the alarms are not displayed.

Only internal tags are enabled for the simulation of alarms in the tag simulator .

Use PLCSim to simulate dynamic parameters.

Comfort Panels

If you want to use PLCSim with version V14 or higher for simulation with a Comfort Panel, the device version of the Comfort Panel must be 14.0.0.0 or higher.

15.2.5 Loading projects

15.2.5.1 Overview for loading of projects

Overview

Delta data of the project is automatically compiled before you download it to one or several HMI devices. This always ensures that the latest version of the project is transferred.

Loading a project to an HMI device

The following steps are completed prior to downloading:

1. The download settings are verified. The "Extended loading" dialog box is opened automatically during the initial download of a project to an HMI device. You use this dialog to define the protocol and interface or destination path for the project in accordance with the HMI device Runtime used.
   You can open the "Extended download" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens.
2. The project is compiled. Warnings and errors during compilation are displayed in the Inspector window and in the "Load preview" dialog.
3. The "Load preview" dialog shows you the following information for each HMI device:
   – The individual steps for loading
   – If the image of the target HMI device does not match the image from the configuration, a prompt is displayed asking whether you want to now change the image.

NOTICE
Changing the image deletes all data from the HMI device. Data is deleted on the target system if you change the image. For this reason, first back up the following data, if necessary: • User administration • Recipes

   – Presettings that take effect at loading. You can change the default settings for this download process, if necessary.
   – Occurring warnings (optional).
     You can download a project while ignoring the "warnings". The functionality may be restricted in runtime.
   – Occurring errors (optional).
     You cannot load the project.
     WinCC will open the invalid configuration in the corresponding editor if you double-click the error message in the Inspector window. Correct the errors and reload the project.

If you are using HMI tags in your project that are connected to PLC tags, you should also compile the user program before you compile the HMI device.

Loading with S7 routing

Configure the S7 routing settings in the "Devices & Networks" editor in the relevant controller. The settings depend on the device configured.

S7 routing supports the following protocols:

- MPI/PROFIBUS
- Ethernet

Loading a project without a connected HMI device

If you cannot establish a direct connection from the configuration PC to the HMI device, copy the compiled project to an external storage medium, such as a USB stick, and then load it onto the HMI device.

This function is available in connection with a HMI device image that is compatible with TIA Portal version V14 or higher.

Generate the required project data in WinCC by configuring the HMI device and then dragging the HMI device folder (such as "HMI_1 [<*Device type*>]") to an external storage medium under "Card Reader/USB memory".

Transferring Runtime add-ons

Projects can contain Runtime add-ons in the form of controls or CSP (Communication Support Packages). These Runtime add-ons are automatically transferred with the project.

See also

Updating the operating system on the HMI device (Page 2342)

15.2.5.2 Loading a project

Introduction

Before a project can run on an HMI device, you must first load it to the HMI device. During loading, you must most importantly specify whether existing data on the HMI device such as "user administration" and "recipe data" is to be overwritten.

If the HMI device supports PROFINET, the name of the HMI device entered in the project tree is used as the device name for the PROFINET communication. The name is written to the HMI device during download. If a device name for the PROFINET communication has already been entered in the HMI device, it will be overwritten.

As a general rule, only one project can be active in Runtime on an HMI device. An HMI device is generally configured to exit Runtime automatically when loading is started. If this is not the case, you will have to exit Runtime manually on the HMI device.

If the image of the target HMI device does not match the image from the configuration, a prompt is displayed asking whether you want to now change the image.

NOTICE
Changing the image deletes all data from the HMI device. Data is deleted on the target system if you change the image. For this reason, first back up the following data, if necessary: • User administration • Recipes

Controlling the transfer behavior on the HMI device

As a general rule, only one project can be active in Runtime on an HMI device. An HMI device is generally configured to exit Runtime automatically when loading is started. If this is not the case, you will have to exit Runtime manually on the HMI device.

You define how the HMI device reacts when the project is loaded in the "Start Center" under "Settings" on the HMI device:

Transfer mode	Effect
Off	The project cannot be loaded to the HMI device.
Manually	The project can only be loaded to the HMI device if the following requirements are met: • Runtime is not running • The HMI device is in "Transfer" mode.
Automatic	The project can always be loaded to the HMI device. If a transfer is started on the configuration PC and a project is in runtime on the HMI device, the running project is automatically closed. For Mobile Panels, this transfer mode is disabled for security reasons.

Note

Closing Runtime automatically

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the "Start Center".

Please refer to the documentation for the HMI device used for more detailed information on transfer settings.

Requirements

- You have created an HMI device in the project.
- The HMI device is connected to the configuration PC.
- The "Start Center" has been started on the HMI device.
- The protocol by which the project is loaded is set on the HMI device in the "Start Center" under "Settings".
- Transfer mode is set as "Automatically" or "Manually" in the HMI device.

Procedure

Proceed as follows to load a project:

1. To download a project simultaneously to several HMI devices, select the HMI devices by means of multiple selection in the project tree.
2. Select the "Download to device > Software" command from the shortcut menu of an HMI device.
3. If the "Extended loading" dialog is open, configure the "Settings for loading". Make sure that the "Settings for loading" correspond to the "Transfer settings in the HMI device".
   - Select the protocol used, for example, Ethernet or HTTP.
   - Configure the relevant interface parameters on the configuration PC.
   - Make any interface-specific or protocol-specific settings required in the HMI device.
   - Click "Download".

   You can open the "Extended download" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens. The project is compiled at the same time. The result is displayed in the "Load preview" dialog.
4. Check the displayed presettings and change them as necessary.
5. Click "Download".

Result

The project is loaded to all selected HMI devices. Any existing project is replaced. The data for user administration and / or recipes is replaced in accordance with the settings in the "Load preview" dialog.

During the download, you can keep track of the files that are transferred.

If errors or warnings occur during the download, corresponding alarms are displayed under "Info > Load" in the Inspector window.

On completion of the successful download of the project, you can execute it on the HMI device.

Note

If the transfer is interrupted, WinCC automatically ensures that no data is lost and that existing data is deleted on the HMI device only after complete transmission.

See also

Backing up and restoring data of the HMI device (Page 2339)

Updating the operating system on the HMI device (Page 2342)

Error messages during loading of projects (Page 2201)

Adapting the project for another HMI device (Page 2142)

Establishing a connection to the HMI device (Page 2113)

15.2.5.3 Loading project from external storage medium (Basic Panels)

Introduction

If you cannot establish a direct connection from the configuration PC to the HMI device, copy the compiled project to an external storage medium, such as a USB stick, and load it from the USB stick onto the HMI device.

This function is available in connection with a HMI device image that is compatible with TIA Portal version V14 or higher.

This function is available for Basic Panels 2nd Generation.

NOTICE
Data loss If you load a project with the option "Upgrade or downgrade Firmware", the operating system of the HMI device is updated. Existing data on the HMI device including the HMI device password is deleted. Settings in the Start Center are retained, license keys are stored on the external storage medium before the operating system is updated. If required, back up the data before loading.

You create the required project data in WinCC by configuring the HMI device and then dragging and dropping or copying and pasting the folder of the HMI device (e.g. "HMI_1 [<device type>]") to an external storage medium under "Card Reader/USB memory".

Requirements

- You have started the Start Center on the HMI device.
- The storage medium with the saved project is inserted in the HMI device.

Procedure

1. Click "Settings".
2. Open the "Service & Commissioning" dialog.
3. Confirm with the "Project Download" button.
   With "Project Download", you have the option of downloading a single project from an external storage medium to the HMI device.
4. Select the storage medium on which the project data is saved.
5. Select the project file.

   Note

   If no storage medium is inserted in the HMI device, the "Projects" list is empty.

6. Select the options for loading the project:
   - "User administration:"
     Overwrite user administration on the HMI device with the user administration of the project
   - "Recipe data:"
     Overwrite HMI device recipes with project recipes
   - "Upgrade or downgrade Firmware:"
     Update firmware on the HMI device if the firmware version of the HMI device is not incompatible with the firmware version of the project

   Note

   You can only change the settings for overwriting the user management or recipes if they were not previously interlocked in the configuration.

7. Confirm with the "Accept" button.

Result

After the loading process, the new project is started on the HMI device.

15.2.6 Compiling and loading with Multiuser Engineering

15.2.6.1 Compiling and loading with multiuser engineering (overview) (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

When using multiuser engineering for your projects, you should take into account the response when compiling the Runtime projects and the response when downloading them to HMI devices.

You can compile and download to an HMI device in both the server project view and in the local session.

You can find more information about the topic of "multiuser engineering" under "Using Multiuser Engineering".

Basics

For HMI devices and RT Advanced, the following scenarios are possible in multiuser engineering:

- Compiling in the server project view
- Compiling in the local session

- Loading from the server project view
- Loading from the local session

Note

Complete download from the server project view or local session is no different to complete download in a single-user project. With a complete download, the current Runtime project is loaded from the currently active view to an HMI device.

Note

Compiling and downloading in a local session is no different from compiling and downloading in a single-user project.

In principle, you can execute all commands for compiling and loading in multiuser engineering projects:

- "Software (rebuild all)"
- "Compile > Software (only changes)"
- Software (all)

The term "project"

The term "project" has two different meanings in the contexts of compilation and loading. "Project" is the WinCC project on the configuration PC. "Project" is also the Runtime project you create by compiling the configuration data of an HMI device and download to the HMI device.

- WinCC project: contains the configuration data of one or more HMI devices
- Runtime project: contains the compiled configuration data of one HMI device

Rules

The following basic rules apply to compiling and downloading in multiuser engineering:

- The Runtime project that has been compiled in a local session always remains local and is not uploaded to the multiuser server. It cannot be saved in the multiuser server project.
- Only Runtime projects compiled in the server project view can be saved in the multiuser server project.

You can find additional information on Multiuser Engineering on the Siemens YouTube channel: Multiuser Engineering - one team working simultaneously on a project (https://www.youtube.com/watch?v=n4oTZ2Gzg6U).

See also

Compiling in the server project view (Page 2138)

Compiling in the local session (Page 2139)

15.2.6.2 Compiling in the server project view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Basics

Compiling and downloading in the server project view is no different from compiling and downloading in a single-user project.

During the compiling of a project in the server project view, the multiuser server project is blocked. Other users cannot make changes to this server project during this time. The Runtime project compiled in the server project view is stored along with the engineering project in the central multiuser server. Blocking the multiuser server project ensures that the configuration data and the Runtime project remain in sync.

Note

When you compile and save in the server project view, other users obtain the Runtime project you have updated along with the engineering project when they "refresh" their local session. Other users do not have to recompile the changes you have made after an update.

Example: Compiling during check-in

You make changes to a tag in a local session. All prior changes have been compiled in the associated server project.

If there are no compilation errors, both projects - the modified engineering project (with the modified tags) and the compiled Runtime project - are saved in the central multiuser server project with the "Save changes" command.

If you skip compiling during the check-in, the project contains the changes that have been saved on the server.

The next user who creates a local session from the server project or updates an existing local session must compile your two changes in addition to his or her own changes.

Note

Working on a shared project through multiple local sessions increases the probability of error. It is therefore recommended to compile the project at check-in and eliminate any errors that are reported during compiling. In this way, you provide the next user with a project free of errors.

See also

Compiling and loading with multiuser engineering (overview) (Page 2136)

Compiling in the local session (Page 2139)

15.2.6.3 Compiling in the local session (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Basics

Compiling and downloading projects in the local session is no different from compiling and downloading in a single-user project.

Since the local session is a copy of the server project, the first compilation status of the local session is identical to that of the server project. If the server project contains contents that are not compiled or error messages occurred during the compiling, they are transferred to the local session.

Note

It is recommended to compile the project at check-in and eliminate any errors that are reported during compiling. In this way, you provide the next user with a project free of errors and avoid spreading errors.

Updating in the local session

If you update a project in the local session, the local session - including the compilation status - is completely replaced by the content of the server project. Only the changes marked for check-in are retained in the updated local session and generate additional compiling steps in the local session.

Example: Updating the local session

You make changes to a tag in a local session. All prior changes have been compiled in the associated server project.

You update the content of the local session by clicking the "Update" button. After the update, the local session obtains the compilation status of the server project. There are also compiling tasks for the acquisition of the modified tags.

See also

Compiling and loading with multiuser engineering (overview) (Page 2136)

Compiling in the server project view (Page 2138)

15.2.7 Starting Runtime

15.2.7.1 Starting Runtime on the HMI device (Basic Panels)

Introduction

On completion of the project download to the HMI device, you can start the project in Runtime. The project is saved in the HMI device to a file with the following extension:

- Basic Panels: "*.srt"

The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime.

The programs that you can use to start projects on the HMI device are available in the Runtime installation folder.

Note

Closing Runtime automatically

If automatic transfer is activated on the HMI device and a transfer is started on the configuration PC, the running project is automatically terminated.

The HMI device then automatically switches to "Transfer" operating mode.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the "Start Center".

Requirements

- WinCC Runtime is installed on the HMI device.
- The project was downloaded to the HMI device.
- The "Start Center" has been started.

Procedure

The project is stored on a panel in a folder you specify in the HMI device transfer settings. The "Start Center" application is started on a panel. The project loaded is started automatically after expiration of the configured delay.

If the project does not start automatically:

1. Click on "Start" in the "Start Center" to start the loaded project.

Refer to the documentation for the HMI device for additional information on startup of projects.

15.2.8 Error messages during loading of projects (Basic)

Possible problems during the download

When a project is being downloaded to the HMI device, status messages regarding the download progress are displayed in the output window.

Usually, problems arising during the download of the project to the HMI device are caused by one of the following errors:

- Incorrect download settings on the HMI device
- Incorrect HMI device type in the project
- The HMI device is not connected to the configuration PC.

The most common download failures and possible causes and remedies are listed below.

The serial download is cancelled

Possible remedy: Select a lower baud rate.

The download is cancelled due to a compatibility conflict

Possible cause	Remedy
The configuration PC is connected to the wrong device, e.g. a controller.	Check the cabling. Correct the communication parameters.

Project download fails

Possible cause	Remedy
Connection to the HMI device cannot be established (alarm in the output window)	Check the physical connection between the configuration PC and the HMI device. Check whether the HMI device is in transfer mode. Exception: Remote control
The default communication driver is not listed in the Windows Device Manager.	Check the device status of the COM connection in the properties window of the Device Manager.

Download over MPI/DP interface fails

Possible cause	Remedy
"Configured mode" is set on the CP, for example, if you are using the SIMATIC NET CD.	Set the CP to "PG mode" using the "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the CP. Set the CP back to "configured mode".
On the programming device/PC panel, the "S7ONLINE" access point is not set to a hardware device such as CP5611 (MPI). The cause may be the installation of "SIMATIC NET CD 7/2001".	Set the access point "S7ONLINE" on the selected device using the "PG/PC Panel" or "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the HMI device. Restore the "S7ONLINE" access point to the original device.

The configuration is too complex

Possible cause	Remedy
The configuration contains too many different objects or options for the HMI device selected.	Remove all objects of a type, e.g. all graphic views.

15.2.9 Adapting the project for another HMI device

Introduction

When you download a WinCC project to an HMI device, WinCC checks whether this is compatible with the HMI device type used in the project. If the types of HMI device do not match, you will see a message before the download starts.

The download is aborted.

Adapting the project for the HMI device

You need to adapt the project accordingly to be able to download the project to the connected HMI device.

- Add a new HMI device in the project tree. Select the correct type of HMI device from the HMI device selection.
- Copy the configured components from the previous to the new HMI device.
  Copy large amounts of components directly in the project navigation and details view.
  For example, copy the "Screens" folder to the screens folder of the new HMI device with the help of the shortcut menu.

- Use the detail view to copy entries in the project tree for which the "Copy" command is not available in the shortcut menu.
- Select the "Recipes" entry in the project tree, for example. The recipes are displayed in the detail view.
- Select the recipes in the detail view and drag them to the "Recipes" entry of the new HMI device. The recipes are copied. You can also select multiple objects in the detail view.
- Configure the components that cannot be copied, e.g. connections, area pointers, and alarms.
- Save the project at various points in time.
- Compile the full project.
- When the compilation is successfully completed, download the project to the HMI device.

Linking references

References to linked objects are included in the copying. The references are linked again once the linked objects are copied.

Example:

You copy a screen in which objects are linked to tags. The tag names are entered at the individual objects after the screen is added to the new HMI device. The tag names are marked in red because the references are open. When you then copy the tags and insert them into the new HMI device, the open references are closed. The red marking for the tag names disappears.

For complete references to connected objects in the PLC, you first need to configure a connection to the PLC.

Using the information area

When you compile the project for the HMI device, errors and warnings are displayed in the "Info" tab of the Inspector window. You can use the shortcut menu command "Go to" to go directly to the location where the error or warning can be corrected.

Work through the list of errors and warnings from top to bottom.

When the compilation is successfully completed, download the project to the HMI device.

15.2.10 Reducing the project size (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

When loading a large-scale project to an HMI device, the memory of the HMI device may be insufficient for the project. There are several ways to reduce the size of your project.

Options for reducing the project size

There are several ways to reduce the size of the project and save space:

- Reduce the number of available Runtime languages
  Check whether all selected Runtime languages are actually needed. You can disable the languages that you do not need under "Runtime settings > Language & Font > Runtime language and font selection".
- Do not use help texts for S7 diagnostic alarms
  To reduce the size of the project, you can disable the download of help texts for the S7 diagnostic alarms. In order to avoid downloading the help texts to the HMI device, disable the option "Download S7 diagnostics help texts" under "Runtime settings > Alarms > General".
- Rebuild all software
  In order to optimize the project data and to clean up obsolete changes, compile the entire project using the "Compile > Software (rebuild all)" command from the shortcut menu of the HMI device.
- Harmonize the presentation using styles (as of WinCC V13)
  It is recommended to harmonize screen objects using styles. Standardize the appearance of screen objects in a project to optimize project data. Use the specified preset or customized style for the configuration of the screen objects throughout the project.
- Enable automatic update of PLC alarms (for S7-1500 controllers and V14 HMI devices)
  To save space, you can specify that the PLC alarm texts are only to be loaded in Runtime when needed. To do this, enable the "Automatic update" option under "Runtime settings > Alarms > Controller alarms". Make sure that automatic update of alarms is also enabled in the corresponding controller.
  The "Automatic update" option is not available on Basic Panels.
  The amount of space that can be saved depends on the number of PLC alarms and the number of Runtime languages.
- Reduce the number of fonts loaded
  Check whether the number of downloaded user-defined fonts can be reduced. If necessary, configure only the standard fonts for the required Runtime languages under "Runtime settings > Language & font > Runtime language and font selection".
  To save space, use fewer font groups for the configuration.
- Reduce the size of the graphics
  Check the size of the graphics that you use in the project. If necessary, reduce the size of the graphics by reducing the resolution or choose a higher compression format without noticeable loss of quality for the project graphics.
  To keep the project size small for Basic HMI devices, harmonize the sizes of graphics used in the project.
  Select appropriate graphic formats for your screens: For example, use PNG images for drawings that are not vector graphics and JPEG images for photos.

15.2.11 Basics of operating in Runtime

15.2.11.1 Overview (Basic Panels)

Overview of operator control of a project

Depending on the HMI device, the following options are available for the input with Basic HMI devices.

- Touch screen
- Function keys
- Mouse and keyboard

Use the touch screen and function keys or mouse and keyboard to operate the Control Panel / Start Center or the project running on your HMI device.

Note

Incorrect operation

A project can contain certain operations that require in-depth knowledge about the specific plant on the part of the operator.
Make sure that your plant is only operated by qualified personnel.

Operating options for an HMI device

Depending on your HMI device, operate your plant as follows:

- Touch screen operation
  The display of the device is touch-sensitive. You operate the operating elements in the display with your finger or a stylus.
- Operation with control keys and function keys
  Control keys and function keys are integrated into the housing of the device.
  - Control keys have a specified function, for example, navigation or the acknowledgment of alarms.
  - Function keys are freely user-assignable and their function is therefore project-specific.
- Mouse and keyboard operation
  Mouse and keyboard are connected via an integrated USB port. You operate the operating objects with mouse and keyboard.

Individually configured operation

The configuration engineer has various options available for setting up operation.

Examples of actions whose execution is always determined on a project-specific basis:

- Screen change
- Reporting
- Changing Runtime language

There are no specific operating elements to execute certain functions. The configuration engineer specifies the project-specific execution. A screen change can be triggered with a button or a function key, for example.

Information on project-specific operations can be found in the system documentation.

15.2.11.2 Operation with the touch screen

Overview of operation with the touch screen

Use the touch screen to operate the HMI device of the project that is running on your HMI device.

Operating the touch screen

NOTICE
Damage to the touch screen Do not touch the touch screen with sharp or pointed objects. Avoid tapping the touch screen with hard objects, and avoid constantly using motion control. Both can significantly reduce the useful life or even cause the failure of the touch screen. **Triggering unintended actions** You can trigger unintended actions if you touch several operating elements at the same time. Always touch only one operating element on the screen. Operating elements are touch-sensitive symbols on the screen of the HMI device.

Special features of operation using the touch screen

Operation with the touch screen is characterized by the following special features:

- Enable
  To enable an operator control, touch the touch screen with your fingers or with a stylus. To generate a double-click, touch the operator control twice in rapid succession.
- Value input
  You enter numbers and letters on the touch screen with a screen keyboard.
- Careful operation
  If you touch multiple operator controls at the same time, you may trigger unintentional actions.

Value input using the screen keyboard

The screen keyboard is displayed as soon as you operate a screen object that requires an input. Depending on the HMI device and the configured operating element, the system displays different screen keyboards for entering numerical or alphanumerical values. The screen keyboard is hidden again when input is complete.

File browser (Windows 8 or higher)

You can only operate the file browser dialog with a mouse, keyboard or on-screen keyboard (without using the touch function) on a Windows 8 or higher PC with touch screen. Use the file browser dialog of the Windows operating system with the help of a script on a touch screen PC with Windows 8 or higher.

Screen keyboard

Layout

The figure below shows the basic layout of a screen keyboard on a TP1500 Basic.

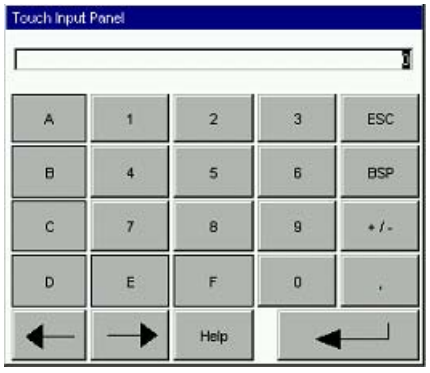

Numbers

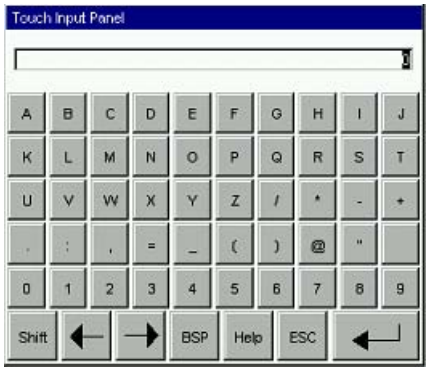

Letters

Operator controls

The following keys are available on the screen keyboard of all HMI devices:

Button	Name	Function
←	Cursor left	Navigates to the left
→	Cursor right	Navigates to the right
BSP	Backspace	Deletes a character
ESC	Escape	Cancels the input

Button	Name	Function
	Enter	Confirms the input
Help	Help	Displays the infotext This key is only displayed when an infotext has been configured for the operator control.

Screen keyboard for Basic Panels 2nd Generation

Layout

The figure below shows the basic layout of a screen keyboard on a Basic Panel 2nd Generation.

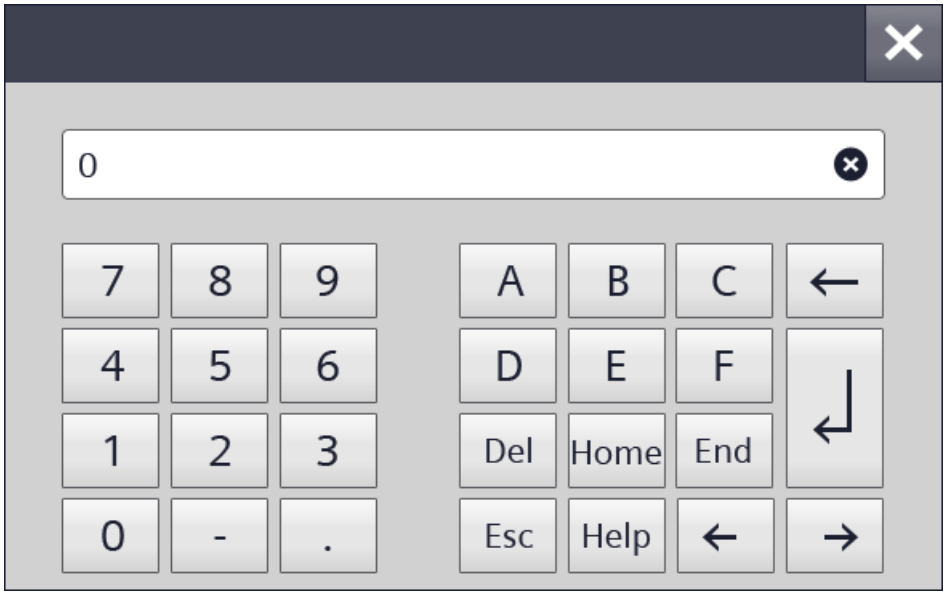

Numbers

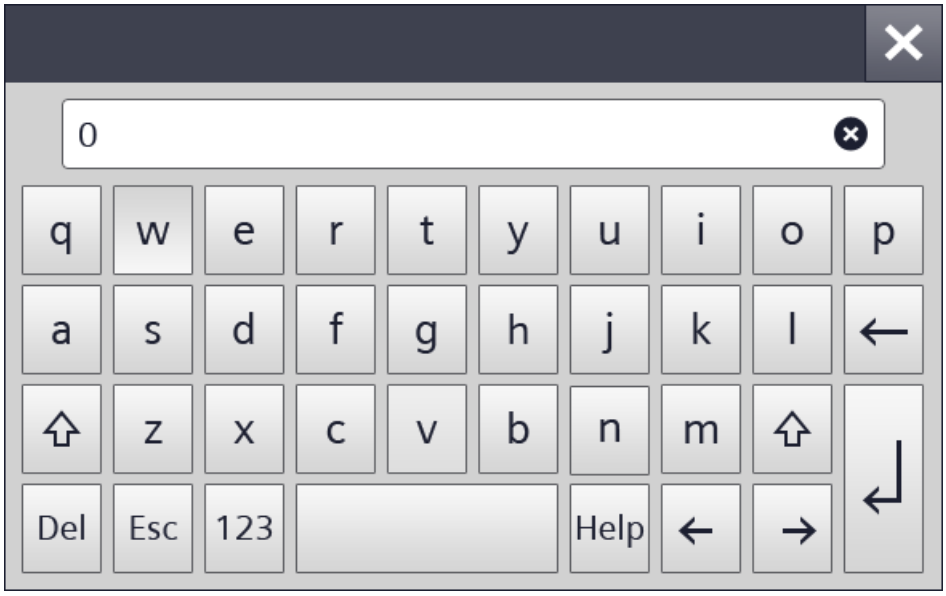

Letters

Operator controls

The following keys are available on the screen keyboard of all HMI devices:

Button	Name	Function
←	Cursor left	Navigates to the left
→	Cursor right	Navigates to the right
←	Backspace	Deletes a character
Esc	Escape	Cancels the input
↵	Enter	Confirms the input
Help	Help	Displays the infotext This key is only displayed when an infotext has been configured for the operator control.

15.2.11.3 Operation with keys

Overview of operation with keys

Introduction

Use the keys of the HMI device to operate the Control Panel / Start Center of your HMI device or the project that is running on your device. Depending on the device, control keys and function keys are available.

For more detailed information, refer to the operating instructions for your HMI device.

Control keys and shortcuts

Introduction

The following tables list the control keys available to operate the project.

Note

The availability of control keys is determined by the HMI device used.

You trigger functions on keyboard HMI devices either with a key or a shortcut. With shortcuts, you keep the first key pressed. Then you press the second key.

Navigating in the display

Key or shortcut		Function	Description
TAB		Tabulator	Selects the next operating element in the tab sequence
SHIFT	TAB	Tabulator	Selects the previous operating element in the tab sequence
◀ ▶ ▼ ▲		Cursor keys	Selects the next operating element to the left, to the right, above or below the current screen object Navigating in the operating element

Operation of operating elements

Key or shortcut		Function	Description
ENTER		ENTER key	• Controls buttons. • Applies and ends an entry • Opens a selection list • Toggles between character mode and standard mode within an input field A single character is selected in character mode. In this mode, you can scroll within the character set using the cursor keys.
SHIFT	▶ ◀ ▲ ▼	Positioning cursor	Positioning the cursor within an operating element, for example, in an I/O field
←		Delete characters	Deletes the character to the left of the current cursor position
INS DEL		Delete characters	Deletes the next character to the right of the current cursor position
ESC		Cancel	• Deletes the input characters of a value and restores the original value • Closes the active dialog.
FN	HOME	Scroll to start	Scrolls to the start page of a list
HOME		Scrolling back	Scrolls the list back by one page
FN	END	Scroll to end	Scrolls to the end of a list.
END		Scrolling up	Scrolls the list up by one page
ALT	▼	Open selection list	Opens a selection list
CTRL	ENTER	Accept value	Accepts the value selected in the selection list without closing the list

Enter shortcut

Key	Function	Purpose
A-Z	Toggle (numbers/letters)	Toggles the assignment from numbers to letters • No LED is lit: The number assignment is enabled. Pressing the key once switches to letter assignment. • The right or left LED is lit: The left or right letter assignment is enabled. Each time the key is pressed, the system toggles between the left letter assignment, the right letter assignment and the number assignment.
SHIFT	Shift (upper/lower case)	Used in shortcuts, for example, for switching to uppercase letters
FN	Toggle to additional keyboard layout	Some of the keys contain a blue special character in their bottom left corner, for example, the percent sign "%". To input these characters, press the relevant key in combination with the special character key shown on the left.
CTRL	General control function	Used in shortcuts, for example, for navigating trend views
ALT	General control function	Used in shortcuts, for example for the "Watch table" screen object

Acknowledge alarms

Key	Function	Purpose
ACK	Acknowledge	Acknowledges the currently displayed alarm or all alarms of an alarm group The LED is lit as long as an unacknowledged alarm is active.

Displaying infotext

Key	Function	Description
HELP	Displaying infotext	For the selected object, opens a window with the configured infotext, for example, for an alarm or an I/O field The LED is lit if an infotext is available for the selected object.

Key or shortcut

Function Keys

The assignment of the function keys (F1, F2, F3, etc.) is defined during configuration.

Function keys with global function assignment

A globally assigned function key always triggers the same action on the HMI device or in the PLC regardless of the screen displayed. The activation of a screen or the closing an alarm window, for example, is such an action.

Function keys with local function assignment

A function key with local function assignment is screen-specific and is therefore only effective within the active screen.

The function of a locally assigned function key can vary from screen to screen.

Within a screen, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Operating function keys

Note

Operating function key after screen change

If you press a function key after a screen change, the associated function may be triggered in the new screen before the new screen is fully displayed.

15.2.11.4 Navigating in the display (BS)

Introduction

You navigate on the display of your HMI device as follows:

- between configured screen objects
- within screen objects
  When you select a complex screen object, the cursor focus switches to the screen object and follows the tab sequence there.
- in tables of screen objects

Procedure

- To navigate in the specified tab sequence, press the <TAB> key.
- To navigate freely between the operator controls, press the cursor keys.

Depending on the configuration of your HMI device, you can also use function keys or shortcuts for navigation.

When you operate your HMI device with the touch screen or with the mouse, you implicitly navigate by triggering a desired action. For this purpose, touch or click the operator control.

Result

The operator controls receive the cursor focus according to the selected sequence. You can trigger an action on the selected operator control.

For more detailed information, refer to the operating instructions for your HMI device.

15.2.11.5 Triggering an action

Introduction

Triggering an action at an operator control can mean the following:

- A command is executed.
  Example: Click a button to trigger a script or to execute a predefined function.
- An object is enabled.
  Example: To enter a value, select a table cell with the <Enter> key.

Requirement

- You have navigated to the operator control on which you want to trigger the action.
- The operator control has the cursor focus.

Procedure

- Press <Enter>.
  Or
- Touch the operator control on the touch screen once or twice in rapid succession.
  Or
- Click or double-click the operator control with the mouse.

Result

The following results are possible:

- The requested command is executed.
- The screen keyboard is opened and/or the cursor blinks in the input area of the operator control.
- The element is selected and can be moved.

For more detailed information, refer to the operating instructions for your HMI device.

15.2.11.6 Entering a value

Introduction

Depending on the input format, you enter numerical or alphanumerical values in an input field.

You enter these values depending on the existing hardware using the screen keyboard, the control keys of the HMI device or an external keyboard.

Requirement

- The object is an input field or table field.
- The operator control is enabled.

Entering a value

1. Enter the desired value.
2. To confirm the value and exit the field, press the <Enter> key.
3. To discard the value and exit the field, press the <Esc> key.

Result

A value is entered or discarded. You navigate as needed to the next operator control.

For more detailed information, refer to the operating instructions for your HMI device.

15.2.11.7 Moving operator controls

Introduction

In Runtime, you can move the movable operator controls of a screen object with the mouse or using the touch screen, for example, a slider or a scroll bar. Operation with the keyboard is described below.

Requirement

- A movable operator control is selected.

Procedure

- To move the operator control, proceed as follows depending on the operating element:
  - Standard for touch screen: Press the cursor keys.
  - Standard for keyboard devices: Press <SHIFT> and the cursor keys.
  - Switches: Press <ENTER>
  - Slider: Press <PgUp> or <PgDn>

1. To finish the movement, navigate to another screen object or operator control.

Slider procedure

1. To move the operator control, press the cursor keys.
2. To finish the movement, navigate to another screen object or operator control.

Result

The position of the movable operator control and the display in the screen object have changed.

For more detailed information, refer to the operating instructions for your HMI device.

15.2.11.8 Displaying infotext

Introduction

Depending on the configuration, additional information and operating instructions are available as infotext. The infotext is assigned to an operating element, an alarm or to the open screen. The infotext of an I/O field may contain, for example, information about the value to be entered.

As an alternative to the <Help> key of the HMI device, use the <Help> key of the screen keyboard for input objects.

Requirement

- An infotext is configured for the operating element, the screen or an alarm.

Calling the infotext

1. Enable the desired operating element.
2. Press the <Help> key of the HMI device.
   The infotext for the operating element is displayed.

If you are operating your input object with the touch screen, the screen keyboard opens. If the <Help> key appears, an infotext is configured for the operating element or the current screen.

If there is no infotext for the selected screen object, the infotext for the current screen is displayed, if it has been configured.

Use the scroll bar for long infotexts.

Depending on your configuration, infotext can also be retrieved by means of a configured operating element.

Switching between infotexts

- To switch between the infotexts of the operating elements and the screen, enable the infotext window.

Hiding infotext

- To hide the infotext, press the <Esc> key or press the <Help> key again.

15.2.11.9 Changing Runtime language

Introduction

The HMI device supports multilingual projects. A corresponding operating element which lets you change the language setting on the HMI device in Runtime has been configured.

The project always starts with the language set in the previous session.

Requirement

- The required language for the project is available on the HMI device.
- The language switching function is linked to an operating element, for example, to a button.

Selecting a language

You can change project languages at any time. Language-specific objects are immediately displayed on the screen in the new language when you switch languages.

You can switch the language in Runtime in one of the following ways:

- Use a configured operating element to switch from one language to the next in a list.
- Use a configured operating element to directly set the required language.

15.2.12 Entering bar codes with optical handheld readers

Introduction

With the optical handheld readers you can identify components, machines and other objects optically and enter them directly into multiple operating elements on your HMI device.

Optical handheld reading devices read two dimensional data matrix codes, one dimensional barcodes and postal bar codes.

The following optical handheld readers are supported:

- SIMATIC MV320
- SIMATIC MV340

You can find templates for the settings and instructions on configuration in the manual for your optical handheld reader.

Note

The optical handheld reader is connected to the USB port of the HMI device. Only one device at a time can use the USB port. This is why it is not possible to use a USB keyboard and an optical handheld reader or two optical readers at the same time.

Note

When you connect the optical reader to a configuration PC, ensure that the USB port provides sufficient power for the reader.

Procedure

You read the marking of an object using the connected optical handheld reader. The value is entered in the object on the HMI device which has the focus during runtime.

After it has been read in, confirm the value with the Enter key or with the "Suffix - Enter" that you have previously configured in the settings of your optical handheld reader.

Basic Panels, Comfort Panels, Mobile Panels, RT Advanced

The following objects support input with optical handheld readers:

Object	Preconditions for input
I/O field Date/time field	The corresponding data type is selected. The object and the tag length are configured accordingly. The operator element has the cursor focus.
Recipe view	The recipe data record has the cursor focus.
User view	The logon dialog is open. The user name or password has the cursor focus.
HTML browser	The operator element has the cursor focus.
Runtime dialogs which support keyboard entry	The dialog is open and the corresponding input field has the cursor focus.

Comfort Panels, Mobile Panels, RT Advanced

The following objects support input with optical handheld readers:

Object	Preconditions for input
Sm@rtClient view	Write access is configured for Sm@tClient. The operator element has the cursor focus.
File browser	The field "File path" has the cursor focus.
Watch table	The input field "Control value" has the cursor focus.

Object	Preconditions for input
PDF view	The search field has the cursor focus.
SINUMERIK NC operator display	The operator element has the cursor focus.

Result

The code is read and entered into the corresponding input field.

15.2.13 Security on the HMI device

15.2.13.1 Signature basics (RT Advanced, Basic Panels, Panels, Comfort Panels)

Overview

The HMI device software and the runtime and device options are protected from manipulation by signatures. The device options available over ProSave are also signed.

HMI device signature

The validity of the HMI device signature is checked when the HMI device software is loaded. If the HMI device signature is invalid, loading of the HMI device software is canceled and an error message is output.

Signature for runtime and device options

If the signature check finds that the signature is invalid, an error message to this effect is output. An error message indicating an invalid signature is also output in ProSave.

Note

To allow loading of the HMI device software or the options without a signature (for versions prior to V14), you will need to switch off the signature check.

For details of how to do this, see Switching off the signature check (Page 2160).

See also

Switching off the signature check (Page 2160)

15.2.13.2 Switching off the signature check (RT Advanced, Basic Panels, Panels, Comfort Panels)

Procedure

1. Press "Transfer" twice in the control panel of the HMI device.
   The "Transfer Settings" dialog box opens.
   The option "Validate signatures" is enabled by default.

   Note

   If you wish to load an older WinCC project to an HMI device, you will need to check the settings for the signature check.

2. To allow the HMI device software to be loaded without a signature or to allow the download of a trustworthy option from a third-party provider, disable the "Validate signatures" option.

See also

Signature basics (Page 2159)

15.3 Runtime Advanced and Panels

15.3.1 Runtime settings

15.3.1.1 Runtime settings

Introduction

To edit the runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree.

Note

Device dependency

The options available for selection in the runtime settings depend on the HMI device.

Overview

You configure options from different areas of your project in the runtime settings:

- Display in runtime
  - Start screen
  - Default template
  - Default style of the project and style of the HMI device
  - Screen resolution
  - Full-screen mode
  - Project graphics
  - Bit selection for color and flashing
  - Bit selection for text and graphic lists
- Project ID for compatibility check with the controller
- Logging language
- Remote control via Sm@rt server in the "Services" area
  You can find additional information on this in the WinCC online help under "Options > Sm@rt options"

- Operator control in runtime
  - Task switching
  - Load additional information for debugging scripts: Object names and comments in scripts
  - Show limits as a tooltip
  - Screen keyboard
  - Disable function keys
  - Release buttons when cursor leaves the button
  - Keyboard layout and shortcuts
- GMP-compliant configuration
  You can find additional information on this in the WinCC online help under "Options > WinCC Audit".
- Alarms
  - Buffer overflow
  - Acknowledgment group text
  - Log
  - Colors of the alarm classes
  - Configuration of the system events
  - Configuration of the controller alarms
  - Persistent alarm buffer
- User administration
  - Configuration of the password and logoff time
  - Limit for logon attempts
  - Hierarchy levels
  - Configuration of SIMATIC Logon
- Language and font
  - Available runtime languages
  - Order of language switching
  - Default font
  - License status of the fonts
- Configuration of the OPC Unified Architecture Server
- Settings for synchronization of PLC and HMI tags

See also

Settings for synchronization (Page 544)

Configuring display in Runtime (Page 2163)

Configuring operation in Runtime (Page 2165)

Configuring alarm buffer overflow (Page 891)
Creating alarm classes (Page 785)
Editing system events (Page 802)
Editing controller alarms (Page 803)
Configuring reporting of alarms (Page 873)
Settings for the user administration (Page 1319)
Compatibility check with the controller (Page 2167)
Managing languages (Page 2012)
Working with Logs (Page 1027)

15.3.1.2 Configuring display in Runtime (Comfort Panels)

Introduction

In the Runtime settings, you specify how screens, alarms and language are displayed.

To edit the Runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree. The options available for selection depend on the HMI device.

Display on the target system

In WinCC, configure the visual representation of the generated project in Runtime:

- Select the full-screen or window mode for the project. The window has a smaller size than the screen. In full-screen mode, the project is zoomed to the full screen. There will be no window and operator controls for this view.
  Click "General" in the "Settings" editor to activate the full-screen display at startup. Activate the "Full-screen mode" check box in the "Screen" field.

  Note
  Truncated view

  If the HMI screen does not match the configured size (in pixels), the project only appears on part of the screen in full-screen mode.

- To hide the taskbar, disable the taskbar under Windows.
- Under "General > Screen", you specify whether the HMI device uses the default style of the project or another pre-defined style.

Language settings

- Logging language
  Under "General > Logs", you specify the language in which you want to write the logs. The selected logging language always remains the same, even if the user switches the language in Runtime.
  If you select "Startup language", the logging language remains the same until Runtime is stopped, even if the user has switched the language. However, the next time Runtime is started, the last configured Runtime language is used as the logging language.
- Runtime language
  Under "Language & font", you specify all Runtime languages and their display.

Note

"Bold" and "Italic" fonts

"Bold" and "italic" fonts are not supported on all HMI devices in Runtime.

Note

Basic Panels

Text entries may be truncated if you select an unfavorable font size or style. This setting may have an effect on the following text entries:

- Tooltips
- Long alarm text
- Text in the dialogs

Setting for specific screen objects

- Specify bit selection
  Under "Screens > Bit selection", you specify whether bit selection will be used on this HMI device for the following:
  - Text and graphics (text lists and graphic lists)
  - Color and flashing (appearance analysis)
    When you enable an option, the selection that is configured in the set bit with the least significance is displayed:
    When you disable an option and several bits are set, the selection that has been configured only for the set bit is displayed.
- Tooltips for I/O fields
  Under "Screens > Display", you specify whether configured limit values are displayed as tooltips in Runtime when values are entered in I/O fields.
- Colored objects
  You specify the color depth of your HMI device under "General > Screens > Color depth". As needed, you assign screen objects a colored appearance. The range of colors is determined by the color depth supported on your selected HMI device.

Design settings in the Windows Control Panel

Select the "Windows 7" design for the configuration PC and the Runtime PC in the system settings of Windows. Use this setting to ensure correct representation of the dialogs and the on-screen keyboard in Runtime.

15.3.1.3 Configuring operation in Runtime

Introduction

In the Runtime settings you specify how screens are operated, and how you use keys for operation in Runtime.

To edit the Runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree. The options available for selection depend on the HMI device.

Locking task switching

Depending on the HMI device, you can lock task switching on the HMI device. This will prevent the operator from opening other applications in Runtime.

In the Runtime settings of the HMI device, enable the "Lock task switching" option under "General > Screen".

To use the "Lock task switching" option on a Runtime PC, disable the Aero theme in Windows 7 and Windows 8.x. To disable the Aero theme, right-click on the desktop and select "Personalize". In the "Personalization" menu, select the designs "Windows Basic" or "Windows - Classic".

Note

Stop Runtime

If you lock program switching, always configure in your project the system function "StopRuntime" for an object, e.g. a softkey or button.

Note

The option "Lock task switching" affects the Runtime Advanced on-screen keyboard, but not the Windows on-screen keyboard.

Deactivating the Windows screen keyboard

To lock task switching for the Windows on-screen keyboard, you have to completely deactivate the Windows on-screen keyboard.

Proceed as follows to deactivate the Windows on-screen keyboard:

1. Open the on-screen keyboard.
2. Open the “Options” dialog via “Tools > Options".
3. In the "Initialization" tab, deactivate the options "Touch input, show the icon next to the textbox", "Show the icon on the taskbar", "Use the input panel" tab.

The Runtime Advanced on-screen keyboard is not affected by these changes and can continue to be opened and operated.

Note

To activate the Windows on-screen keyboard again, open Start > Programs > Accessories > Ease of Access > On-screen keyboard”.

Settings for data transfer to the HMI device

- Transferring names of the screen objects in scripts
  If you enable the "Load names" option under "General > Screen", the object names of the screen objects are transferred instead of coded addressing information. You need the object names if you want to address the screen objects in a script using the object names. When you test a script in the debugger, the code becomes more comprehensible through the display of object names.
- Displaying comments in scripts
  Under "Screens > Display", you specify whether the comments in scripts are also transferred to the HMI device.
  If you do not transfer the comments, you save storage space on your HMI device.
  If you transfer the comments, the script is more easily understood during testing in the debugger.

Operation using function keys

To symbolize the function of a function key for keyboard devices, you configure a graphic next to the function key in the display. Under "Screens > Function keys > User-defined pictogram size", you specify whether you want to use a graphic size that deviates from the standard size.

Operation with keyboard

Under "Keyboard > General > Use screen keyboard", you specify whether the screen keyboard is available on your HMI device.

Under "Keyboard > General > Release button on exit", you specify whether the "Release" event is triggered when the user exits a button without releasing it.

Under "Keyboard > Disable function keys in dialogs", you specify for key devices whether the function keys are disabled for the duration of the displayed dialog. Large dialogs can hide explanatory areas or graphics at function keys or bordering function keys. Functions keys can be unintentionally pressed due to this.

15.3.1.4 Compatibility check with the controller (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)

Introduction

The "Project ID" area pointer is used to configure the project ID for checking the consistency of the project with the PLC. The area pointer reads the project version from the PLC. You can find additional information in the online help for WinCC under "Communicating with controllers".

Compatibility check with the controller

You select the project ID in the Runtime settings of the HMI device under "General > Identification > Project ID".

At startup, Runtime checks whether the project ID of the project matches the PLC project version. Runtime will only start if the two values match.

An area pointer "Project ID" is created for connections whose address you specify in the PLC. The value that you store at this address will become the project ID.

The project ID is not available on all HMI devices.

15.3.1.5 Setting time base (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)

Setting the time base for the time of day

You set the time in the Control Panel of your HMI device. For more detailed information, refer to the operating instructions for your HMI device.

Synchronize date and time with the PLC

You can find additional information on this subject in the online help for WinCC under "Communicating with the PLC".

15.3.1.6 Printing in Runtime (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)

Print functions

Print functions available in online mode:

- Hardcopy
  You print out the currently displayed screen by means of an operator control that triggers the "PrintScreen" system function.
- Printing alarms
  Every alarm that has occurred and its state changes are reported along on a printer.
- Printing reports
  Reports are output in graphic mode. The use of a serial printer is not recommended because of the accumulated data volume.
  For proper output, the printer must support the paper format and page layout of the report.

Note

The value of a tag in the report is read and output at the moment of printing. A substantial time may elapse between printing out the first and the last page of a report consisting of several pages. This may lead to the same tag on the last page being output with a different value from that on the first page.

15.3.1.7 Password protection in Runtime

Password protection in Runtime (Advanced) (RT Advanced, Panels, Comfort Panels)

Overview

The connection establishment can be protected with passwords in the TIA Portal. The configured passwords are saved encrypted. The passwords are also protected against unauthorized access during transfer to the target system and cannot be read. These passwords are decrypted securely on the HMI devices in Runtime.

Password protection can be created for the following connections:

- SIMATIC S7-1200
- SIMATIC S7-1500
- SIMATIC S7-1500 Software Controller (WinAC)
- PC Station 2.0 (station manager)
- SIMATIC HMI HTTP

Password protection can also be created for the following system function:

- "SendEMail" system function

Note

The passwords are deleted when you change to an older version.

Download certificate

The certificate for decryption of the passwords is saved in the certificate store of the HMI device. You will be informed by a system message if the certificate is not available.

You restore the certificate by downloading the configuration once again.

Configuring a password for transport (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

To protect the private password key during transport, you can configure a password for transport under "Runtime settings".

Creating a password

1. Open the "Runtime settings > General" editor in the Project window.
2. Enter the password in the "Enter password" box in the "Transport password" area.
3. Confirm the password in the "Confirm password" box.

Note

If there was no previous password, the "Old password" box is disabled.

Result

You have now configured the password for transport. If a password for transport has been configured and you transfer the project to the HMI device, a dialog with a password prompt will appear once in runtime.

If the password entered is correct, the certificate is saved in the certificate store of the HMI device.

When you change the password, load the project with the changed password into the HMI device and start Runtime. No password prompt appears.

Resetting the password for transport (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

If you have moved the transport password, for example, you can create a new security certificate. To do so, reset the transport password.

Resetting the password

1. Click the "Reset password" button.
   The password is reset. A new certificate without a password is generated.

   Note

   Other passwords, for example the password for the PLC, will become invalid. The next time you compile, an error message will appear telling you that these passwords are no longer valid.

2. Enter the new password and confirm it in the "Confirm password" field.

   Note

   The certificate is not password-protected until you enter a new password for transport.

   After download of the project to the HMI device, you will be prompted for the password.

15.3.2 Overview of compiling and loading projects (RT Advanced, Panels, Comfort Panels)

Overview

The project is compiled in the background even as you are configuring it in WinCC. This reduces the time for final compilation. When you start compilation, you create a file that can be run on the corresponding HMI device.

If an error occurs during compilation, WinCC provides support in locating and correcting it.

Once you have corrected any problems, you download the compiled project to the HMI devices on which the project is to run. If the configuration PC is not connected to the HMI device, save the compiled project on a data medium of your choice. The compiled project is then transferred from a PC connected to the HMI device to the HMI device.

If you are using HMI tags in your project that are connected to PLC tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Project

The term "project" has two different meanings in the contexts of compilation and loading. "Project" is the WinCC project on the configuration PC. "Project" is also the Runtime project you create by compiling the configuration data of an HMI device and download to the HMI device.

- WinCC project: contains the configuration data of one or more HMI devices
- Runtime project: contains the compiled configuration data of an HMI device

The figure below illustrates the link between WinCC projects and Runtime projects using the example of the "Compile and load" process:

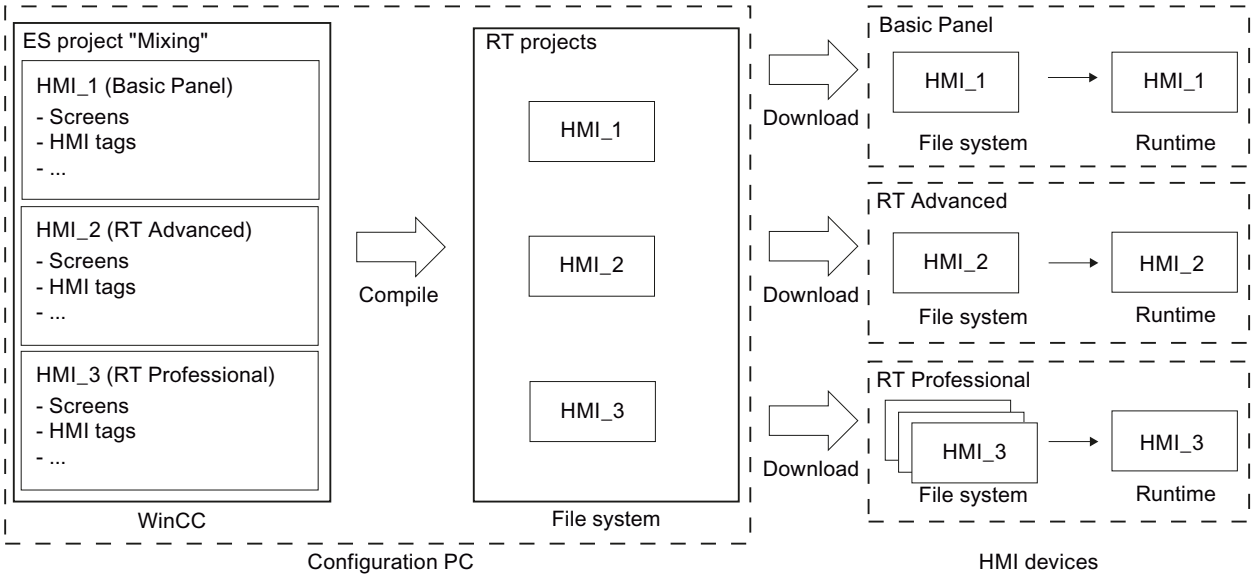

Runtime

Runtime is the software for process visualization. In Runtime, you execute the project in process mode.

A distinction is made between two types of Runtime:

1. Runtime on a panel
   Before running a Runtime project on a panel, you have to transfer the Runtime project to the panel before startup.
2. Runtime on a PC
   You can execute the Runtime project directly on the configuration PC if Runtime has been installed on the configuration PC.
   If you want to execute the Runtime project on a different PC, you have to transfer the Runtime project to the PC before startup.

Runtime version

The Runtime version depends on the image of the configured HMI device. The Runtime version of the compiled project is displayed under "Info" in the Inspector window.

Simulation

You test your configuration with a simulation. You can start a simulation without a link to the active process.

In a simulation, you test configured tags or screen changes, for example. During the simulation, the configured tags can be manipulated, activated and deactivated with the help of the tag simulator.

There are two types of simulation:

1. Simulating a panel
   If you created a panel in your project, the panel is displayed in the simulation. With the help of this type of simulation, you can test your configuration on the HMI device without transferring the project to the panel.
2. Simulating Runtime
   Simulating Runtime allows you to test the project directly on the configuration PC.

15.3.3 Compiling a project (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The changes made to the project are compiled in the background even as you are configuring a project in WinCC. Projects are compiled automatically when you load them. This ensures that the latest version of the project is loaded at all times.

WinCC checks consistency of the project during compilation. The error locations in the project are listed in the Inspector window. You can jump directly to the source of the error from the entry in the Inspector window. Check and correct errors found.

Scope of the compilation

Configuration data is compiled in the background as soon as you start configuring an HMI device. If you compile a project manually, only the changes in the configuration made since the last compilation process are compiled in the background.

You can start complete project compilation manually at any time; this may, for example, be done to test the consistency of the configured data.

Requirement

- A project is open.

Procedure

Proceed as follows to compile a project:

1. If you want to compile several HMI devices at the same time, select all the relevant HMI devices with multiple selection in the project tree.
2. Compile the project:
   - To only compile changes in the project, select the "Compile > Software (only changes)" command from the shortcut menu of the HMI device.
   - To compile all project data, select the "Compile > Software (compile all)" command from the shortcut menu.

Result

The configuration data of all selected HMI devices is compiled. Any errors that occur during compilation are shown in the Inspector window.

15.3.4 Simulating projects

15.3.4.1 Simulation basics (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can use the simulator to test the performance of your configuration on the configuration PC. This allows you to quickly locate any logical configuration errors before productive operation.

You can start the simulator as follows:

- In the shortcut menu of the HMI device or in a screen: "Start simulation"
- Menu command "Online > Simulation > [Start|With tag simulator|With script debugger]"
- Under "Visualization > Simulate device" in the portal view.

Requirement

The simulation/runtime component is installed on the configuration PC.

Field of application

You can use the simulator to test the following functions of the HMI system, for example:

- Checking limit levels and alarm outputs
- Consistency of interrupts
- Configured interrupt simulation
- Configured warnings

- Configured error messages
- Check of status displays

15.3.4.2 WinCC Runtime Advanced simulation (RT Advanced, Panels, Comfort Panels)

Introduction

There are two different simulation modes for Runtime Advanced simulation:

- Device simulation
- Tag simulation

Both types of simulation simulate the project without a direct process link to the configuration PC. Data such as logs or recipes generated during simulation are not deleted. This data is saved on the configuration PC in the paths configured in the project.

Device simulation

Use device simulation to simulate operator control of the HMI device. Device simulation is, for example, used to test screen switching.

Tag simulation

Use tag simulation to simulate the configured process tags. You can either have tag values generated automatically by a simulation table or define tag values yourself.

15.3.4.3 Simulating a project (RT Advanced, Panels, Comfort Panels)

Introduction

You simulate your project with one of the following two methods:

- Without a connected PLC
  You change the value of area pointers and tags in a tag simulator that is read for the simulation of WinCC Runtime.
- With a connected PLC without a running process
  You simulate your project by running it directly in Runtime. The tags and area pointers become active. This allows you to create an authentic simulation of your configured HMI device in Runtime.

Note

Simulation restrictions

You cannot simulate the following system functions:

- CalibrateTouchScreen

You cannot simulate the Media Player. A static screen appears in the simulation window instead of the Media Player.

File access via scripts is not possible for HMI devices with Windows CE.

Requirement

- Simulation without a connected PLC: Tags have been created
- Simulation with a connected PLC but no active process: A project with tags and area pointers has been created

Procedure

To simulate a project using the tag simulator, follow these steps:

1. Open the project on the configuration PC.
2. Select the "Online > Simulation > With tag simulator" menu command.
   For initial project simulation, the simulator is started with a new, empty table. The project is opened simultaneously in Runtime.
   Toggle between the tag simulator and Runtime using the <Alt +Tab> key combination.
3. To simulate a process value, select the corresponding "tag" from the tag simulator.
   The table lists all configured tags. You can simulate up to 300 tags simultaneously.
4. Select the simulation mode in the "Simulation" column.
5. Change the value of tags and area pointers in the respective columns.
6. Activate the "Start" check box to start the simulation for this tag.
7. To save the simulation, select the menu command "File > Save" and enter a descriptive name, for example, "Mixing".
   The file name is assigned the extension "*.cors".

Result

The process values are simulated in Runtime. The tag values are created at random, or incremented, depending on the simulation mode.

To specify tag values, change the simulation mode to "<Display>" and enter a value at "Set value".

The following figure shows a tag simulator with four tags whose values can be determined at random in a range of values from 10 to 1000:

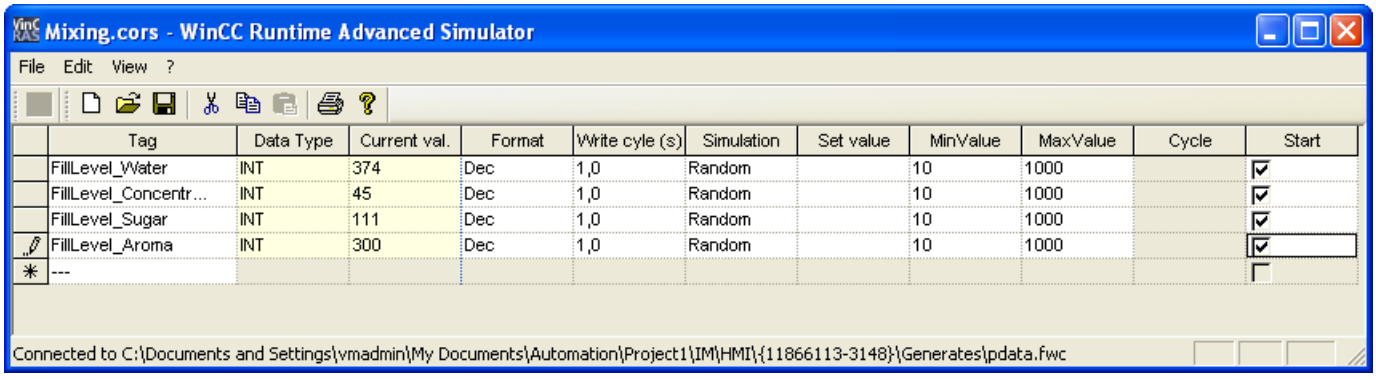

Tag	Data Type	Current val.	Format	Write cyle (s)	Simulation	Set value	MinValue	MaxValue	Cycle	Start
FillLevel_Water	INT	374	Dec	1,0	Random		10	1000		☑
FillLevel_Concentr...	INT	45	Dec	1,0	Random		10	1000		☑
FillLevel_Sugar	INT	111	Dec	1,0	Random		10	1000		☑
FillLevel_Aroma	INT	300	Dec	1,0	Random		10	1000		☑
---										☐

Managing simulation data

If you have saved data from a previous simulation, you can open the file at a later point in time and simulate your project again. The tags and area pointers listed in the tag simulator must still be available in the project.

Proceed as follows to open a simulation file:

1. Select the menu command "Online > Simulate Runtime > With tag simulator".
2. Select the menu command "File > Open" in the tag simulator.
3. Select the corresponding simulation file and click "Open".
   The simulator loads the stored data.

Enabling and disabling tags

Start and stop the simulation for each tag separately in order to facilitate the transition from offline to online engineering. Activate "Start" in the corresponding row.

If a tag is activated, the simulation values are calculated and transferred to the WinCC simulator.

Deleting a tag

To delete a tag from the tag simulator, follow these steps:

1. Select the cell that contains the tag name.
2. Select the "Edit > Cut" menu command.
   The tag is removed from the table.

15.3.4.4 Simulating a screen (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

If you have only made changes to one screen, you can temporarily specify this screen as the start screen for simulation. In this way, you can debug changes without having to modify the start screen, or opening the screen on the HMI device.

Requirements

You created a project that contains at least one screen.

Procedure

To define a screen as temporary start screen for simulation, follow these steps:

1. In the project navigation, select the image to display as the start screen in the simulation.
2. Select the "Start simulation" command from the shortcut menu of the screen.

Result

Project simulation is started. Instead of the configured start screen, the simulation window the screen you selected in the project navigation.

15.3.4.5 Working with the tag simulator (RT Advanced, Basic Panels, Panels, Comfort Panels)

About the tag simulator

The tag simulator has the following columns:

Column	Description
Tag	Specifies the tags for the simulation.
Data type	Shows the data type of the selected tag.
Current value	Shows the simulated value of the defined tags.
Format	Specifies the selected format in which the tag values are simulated: • Decimal (1, 2, 3, 4, ...) • Hexadecimal (03CE, 01F3, ...) • Binary (0 and 1)
Write cycle	Specifies the selected time interval at which the current tag values are simulated. If you enter "2", for example, the current value of the tag will be shown every 2 seconds.
Simulation	Shows the method by which the tag values are processed during simulation.
Set value	Sets the selected value for the respective tag. The simulation start with the specified value.
minValue maxValue	Specifies the value range of the tag. You set a minimum and maximum value for this range. The default values are -32768 for the minimum and 32767 for the maximum.

Column	Description
Period	Contains the period during which the value of the tag is repeated for the "Increment" and "Decrement" simulation modes.
Start	Starts simulation of the tag based on the previously entered information.

Simulation modes

The simulator has six different simulation modes. The configured tags are supplied with nearly realistic values during the simulation.

Simulation mode	Description
Sinusoidal	Changes the tag value to form a sinusoidal curve. The value is visualized as a periodic, non-linear function.
Random	Provides randomly generated values. The tag value is changed by means of a random function.
Increment	Increases the value of the tag continuously up to a specified maximum value. Begins again at the minimum after the maximum has been reached. The value trend corresponds to a positive saw-tooth curve.
Decrement	Reduces the value of the tag continuously down to a specified minimum value. Begins again at the maximum after the minimum has been reached. The value curve corresponds to a negative saw-tooth curve.
Shift bit	Shifts a set bit continuously by one position. The previous position is always reset. This lets you test the alarms of an HMI device, for example.
<Display>	The current tag value is displayed statically.

Example: Simulate tags with the "Shift bit" simulation mode

Proceed as follows to simulate tags with the "Shift bit" simulation mode:

1. Open the project you want to simulate.
2. Select the menu command "Online > Simulate Runtime > With tag simulator".
   The tag simulator opens.
3. In the "Tag" column, select a tag from your project.
4. Select "Bin" in the "Format" column.
5. Enter the value "1" in the "Write cycle" column.
6. Select the "Shift bit" simulation mode in the "Simulation" column.
7. Enter the value "1" in the "Set value" column.
8. Enable the tag with the "Start" check box.

Result

The simulator tests the selected tag bit-by-bit as follows:

Simulation values	Byte for alarms
Set start value	00000001
1. Simulation value	00000010

Simulation values	Byte for alarms
2. Simulation value	00000100
3. Simulation value	00001000
....	...

In Runtime you see if the desired alarm is output at a given value.

15.3.4.6 Simulation restrictions

Alarms with dynamic parameters

If you use tags or text lists as external tags for alarms, the dynamic parameters of the alarms are not displayed.

Only internal tags are enabled for the simulation of alarms in the tag simulator .

Use PLCSim to simulate dynamic parameters.

Comfort Panels

If you want to use PLCSim with version V14 or higher for simulation with a Comfort Panel, the device version of the Comfort Panel must be 14.0.0.0 or higher.

15.3.5 Loading projects

15.3.5.1 Overview for loading of projects (Panels, Comfort Panels)

Overview

Delta data of the project is automatically compiled before you download it to one or several HMI devices. This always ensures that the latest version of the project is transferred.

Loading a project to an HMI device

The following steps are completed prior to downloading:

1. The download settings are verified. The "Extended download to device" dialog box opens automatically during the initial download of a project to an HMI device. You use this dialog to define the protocol and interface or destination path for the project in accordance with the HMI device Runtime used.
   If the HMI device is part of a subnet, for example, you also select the subnet and the first gateway.
   You can open the "Extended download" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens.
2. The project is compiled. Warnings and errors during compilation are displayed in the Inspector window and in the "Load preview" dialog.
3. The "Load preview" dialog shows you the following information for each HMI device:
   – The individual steps for loading
   – If the device version of the target HMI device does not match the configured device version, you are queried if you want to change the device version at this time.
     As long as the project with the selected device version of the target HMI device is not executable, you cannot start the download.

NOTICE
Changing the device version deletes all data on the HMI device. Data is deleted on the target system if you change the device version. For this reason, you should first back up the following data: • User administration • Recipes • License keys With Comfort devices, the licenses can only be deleted when resetting to factory settings. Back up the license keys before you restore the factory settings.

   – Presettings that take effect at loading. You can change the default settings for this download process, if necessary.
   – Warning events (optional). You can download a project while ignoring the "warnings". The functionality may be restricted in runtime.
   – Error events (optional). You cannot load the project. Eliminate the errors and then reload the project.
     WinCC will open the invalid configuration in the corresponding editor if you double-click the error message in the Inspector window. Correct the errors and reload the project.

If you are using HMI tags in your project that are connected to PLC tags, you should also compile the user program before you compile the HMI device.

Note

Interruption of a download

If the download is interrupted, WinCC automatically ensures that no data is lost and that existing data is deleted on the HMI device only after complete transmission.

Loading a project without a connected HMI device

If you cannot establish a direct connection from the configuration PC to the HMI device, copy the compiled project to an external storage medium, e.g. a USB stick, and load it from the USB flash drive to the HMI device.

This function is available in connection with an HMI device image which is compatible with the TIA Portal version V14 or higher.

You create the required project data in WinCC by configuring the HMI device and then dragging and dropping the folder of the HMI device (e.g. "HMI_1 [<*device type*>]") per to an external storage medium under "Card Reader/USB memory".

Loading of projects without recipe data records

You are using recipes in a project. You transfer the project to a Basic Panel without recipe data records.

You may encounter inconsistencies if you have altered the structure of the recipe in the engineering system and the device already held recipe data records.

Check the consistency of the data records in this case. The device will not issue a note for all structural changes.

Pack&Go

If you cannot establish a direct connection from the configuration PC to the HMI device, copy the compiled project to a PC that is connected to the HMI device, for example via a network. Download the project from this PC to the HMI device.

Loading with S7 routing

Configure the S7 routing settings in the "Devices & Networks" editor in the relevant PLC. The settings depend on the device configured.

S7 routing supports the following protocols:

- MPI/PROFIBUS
- PN/IE

Transferring Runtime add-ons in WinCC

Projects can contain Runtime add-ons in the form of controls or CSP (Communication Support Packages). These Runtime add-ons are automatically transferred with the project.

See also

Load project from external storage medium (Page 2191)

Downloading projects from a "Pack&Go" file to the HMI device (Page 2189)

Downloading the project to an HMI device (Page 2287)

Generating a "Pack&Go" file (Page 2188)

Starting Runtime Advanced and Panel Runtime (Page 2200)

15.3.5.2 Overview for loading of projects (RT Advanced)

Overview

Delta data of the project is automatically compiled before you download it to one or several HMI devices. This always ensures that the latest version of the project is transferred.

Connection establishment

Ensure that the device version configured in the engineering system matches the installed Runtime version, otherwise, communication will not be established or will be delayed when the project is loaded.

If the device version does not match the installed Runtime version, you have the following options:

1. Change the device version configured in the engineering system.
2. Install the version of WinCC Runtime Advanced corresponding to the configured device version.

Loading a project to an HMI device

The following steps are completed prior to downloading:

1. The download settings are verified. The "Extended download to device" dialog box opens automatically during the initial download of a project to an HMI device. You use this dialog to define the protocol and interface or destination path for the project in accordance with the HMI device Runtime used.
   If the HMI device is part of a subnet, for example, you also select the subnet and the first gateway.
   You can open the "Extended download" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens.
2. The project is compiled. Warnings and errors during compilation are displayed in the Inspector window and in the "Load preview" dialog.
3. The "Load preview" dialog shows you the following information for each HMI device:
   - The individual steps for loading
   - Check the runtime version of the target HMI device
     If the version of WinCC Runtime Advanced installed on the target device does not match the configured device version, you cannot load the project.
   - Default settings that take effect when loading
     You can change the default settings for this download process, if necessary.
   - Occurring warnings (optional).
     You can download a project while ignoring the "warnings". The functionality may be restricted in runtime.
   - Occurring errors (optional).
     You cannot load the project. WinCC will open the invalid configuration in the corresponding editor if you double-click the error message in the Inspector window. Correct the errors and reload the project.

If you are using HMI tags in your project that are connected to PLC tags, you should also compile the user program before you compile the HMI device.

Note

Interruption of a download

If the download is interrupted, WinCC automatically ensures that no data is lost and that existing data is deleted on the HMI device only after complete transmission.

Loading a project without a connected HMI device

If you cannot establish a direct connection from the configuration PC to the HMI device, copy the compiled project to the HMI device.

Loading with S7 routing

Configure the S7 routing settings in the "Devices & Networks" editor in the relevant PLC. The settings depend on the device configured.

S7 routing supports the following protocols:

- MPI/PROFIBUS
- PN/IE

Transferring Runtime add-ons in WinCC

Projects can contain Runtime add-ons in the form of controls or CSP (Communication Support Packages). These Runtime add-ons are automatically transferred with the project.

See also

Loading a project (Page 2184)

15.3.5.3 Loading a project (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

Before a project can run on an HMI device, you must first load it to the HMI device. During loading, you must most importantly specify whether existing data on the HMI device such as "user administration" and "recipe data" is to be overwritten.

If the HMI device supports PROFINET, the name registered in the project tree is used for the PROFINET communication. The use of the name corresponds to the default settings of the PROFINET interface of the HMI device. For devices with more than one PROFINET interface, the name of the IE CP is automatically added to the device name with a separating period. The name is written to the HMI device during download. If a device name for the PROFINET communication has already been entered in the HMI device, it will be overwritten.

You can find additional information about these settings in the information system in the "Assigning a device name and IP address" section.

If the device version of the target HMI device does not match the configured device version, you are asked whether you wish to change the device version at this time.

NOTICE
Changing the device version deletes all data on the HMI device. Data is deleted on the target system if you change the device version. For this reason, you should first back up the following data: • User administration • Recipes • Licenses Resetting to factory settings also deletes the license keys. Back up the license keys before you reset the system to factory settings. With Comfort devices, the licenses can only be deleted when resetting to factory settings. Back up the licenses and license keys before you reset the system to factory settings.

Note

Transfer to an HMI device with PC station V1.0 or PC station V2.0

The version of the PC station set in the device settings of the HMI device must match the version installed on the HMI device.

Additional information on configuration settings for "PC Station" on the HMI device is available in the "SIMATIC NET" documentation.

Note

Overwriting existing data during loading

During loading, you must specify whether existing data on the HMI device is to be overwritten. By selecting the check boxes in the "Load preview" dialog you can always overwrite the following data during loading:

- Existing user administration data on the device
- Existing recipe data on the device

Establishing a connection to WinCC Runtime Advanced

Ensure that the device version configured in the engineering system matches the installed Runtime version; otherwise, communication when the project is loaded will not be established or will be delayed.

If the device version does not match the installed Runtime version, you have the following options:

1. Change the device version configured in the engineering system.
2. Install the version of WinCC Runtime Advanced corresponding to the configured device version.

Note

If your HMI device is a PC, the device version of the target system is not automatically updated during the download.

Verify that the configured device version corresponds to the one of the target HMI device before you compile and download your project. If necessary, install the matching Runtime version on your HMI device or change the device version manually using the properties of the HMI device.

Controlling the transfer behavior on the HMI device

As a general rule, only one project can be active in Runtime on an HMI device. An HMI device is generally configured to exit Runtime automatically when loading is started. If this is not the case, you will have to exit Runtime manually on the HMI device.

You define how the HMI device reacts when the project is loaded in the "Start Center" under "Settings" on the HMI device:

Transfer mode	Effect
Off	The project cannot be loaded to the HMI device.
Manually	The project can only be loaded to the HMI device if the following requirements are met: • Runtime is not running • The HMI device is in "Transfer" mode.
Automatic	The project can always be loaded to the HMI device. If a transfer is started on the configuration PC and a project is in runtime on the HMI device, the running project is automatically closed. For Mobile Panels, this transfer mode is disabled for security reasons.

Note

Closing Runtime automatically

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the "Start Center".

Please refer to the documentation for the HMI device used for more detailed information on transfer settings.

Requirement

- You have created an HMI device in the project.
- The HMI device is connected to the configuration PC.
- The "Start Center" has been started on the HMI device.
- The protocol by which the project is loaded is set on the HMI device in the "Start Center" under "Settings".
- Transfer mode is set as "Automatically" or "Manually" in the HMI device.

Procedure

Proceed as follows to load a project:

1. To download a project simultaneously to several HMI devices, select the HMI devices by means of multiple selection in the project tree.
2. Select the "Download to device > Software" command from the shortcut menu of an HMI device.

3. If the "Extended loading" dialog is open, configure the "Settings for loading". Make sure that the "Settings for loading" correspond to the "Transfer settings in the HMI device".
   - Select the protocol used, for example, Ethernet or HTTP.
   - Configure the relevant interface parameters on the configuration PC.
   - Make any interface-specific or protocol-specific settings required in the HMI device.
   - Click "Download".

   You can open the "Extended download" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens. The project is compiled at the same time. The result is displayed in the "Load preview" dialog.
4. Check the displayed presettings and change them as necessary.
5. Click "Download".

Result

The project is loaded to all selected HMI devices. In WinCC V13, the project with the Runtime add-ons is loaded to the selected HMI devices.

During the download, you can keep track of the files that are transferred.

If errors or warnings occur during the download, corresponding alarms are displayed under "Info > Load" in the Inspector window.

On completion of the successful download of the project, you can execute it on the HMI device.

Note

If the transfer is interrupted, WinCC V13 automatically ensures that no data is lost and that existing data is deleted on the HMI device only after complete transmission.

See also

Starting Runtime on the configuration PC (Page 2199)

Starting Runtime Advanced and Panel Runtime (Page 2200)

Starting Runtime Professional (Page 2304)

Updating the operating system on the HMI device (Page 2342)

Overview for loading of projects (Page 2182)

15.3.5.4 Generating a "Pack&Go" file (Basic Panels, Panels, Comfort Panels)

Introduction

Create a "Pack&Go" file if you cannot connect the HMI device to the configuration PC. The "Pack&Go" file is a ZIP file containing the following data:

- The compiled project
- A program for transferring the project to the HMI device
- Image for the configured HMI device

Typical example of an application: An engineering company creates a project variant for a new HMI device. The project engineer of the engineering company does not have direct access to the plant. The project engineer therefore e-mails the "Pack&Go" file to his local contact. The contact unpacks the "Pack&Go" file on a PC connected to the HMI device via a network. He then transfers the project from the PC to the HMI device.

Requirement

- You have created an HMI device in the project.

Procedure

To create a "Pack&Go" file, follow these steps:

1. Select the HMI device in the project tree.
2. Select the "Pack&Go" command from the menu under "Online > HMI Device maintenance".
3. The "Create Pack&Go file" opens.
4. Verify that the displayed device version corresponds with that of the target HMI device.
5. Select a protocol as transfer mode.
6. Select the storage location under "Pack&Go file" and enter the file name.
7. If necessary, specify if the "Pack&Go" file is split into several files.
8. Click "Create".

Note

To unzip a "Pack&Go" file which is distributed over several files, you need a compression program. The compression program integrated into the operating system is not adequate to unzip distributed files.

Result

The "Pack&Go" file is created and saved to the specified folder in the file system. Now copy the "Pack&Go" file to the PC connected to the HMI device.

15.3.5.5 Downloading projects from a "Pack&Go" file to the HMI device (Basic Panels, Panels, Comfort Panels)

Introduction

Pack&Go is a 64-bit application that can only run on a 64-bit operating system.

If the TIA Portal is not installed on your PC, install Microsoft Visual C++ 2015 Redisitributable Package (x64) and .Net Framework 4.6.1 to run the Pack&Go application.

- You can find the Microsoft Visual C++ 2015 Redistributable Package (x64) in the Download Center of Microsoft at the following URL:
  http://www.microsoft.com/en-us/download/details.aspx?id=48145
- You can find .Net Framework 4.6.1 at the following URL:
  https://www.microsoft.com/en-us/download/details.aspx?id=49982

Requirement

- .NET Framework with version 4.6.1
- ProSave is installed on the PC
- The "Pack&Go" file has been stored in the file system of a PC.
- The PC is connected to the HMI device.
- The device version of the HMI device is consistent with the device version in the project

Note

To unzip a "Pack&Go" file which is distributed over several files, you need a compression program. The compression program integrated into the operating system is not adequate to unzip distributed files.

Procedure

To download the project from a "Pack&Go" file to an HMI device, follow these steps:

1. Unpack the "Pack&Go" file to a folder of your choice in the file system of the PC.
2. From the "PackNGo" subfolder of this directory, run "Siemens.Simatic.Hmi.PackNgo.exe". The "Pack'n Go" dialog opens. The settings for loading are set by default. Exception: You must select the target device if you are using an S7 USB connection.
3. Adjust the settings for loading appropriately if they differ from the interfaces or protocols available on the PC.
4. Specify whether or not to overwrite the user management and recipe data already stored on the HMI device.
5. Click "Transfer".

Result

The connection to the HMI device is set up via the selected path. An alarm is output if the device version used in the project differs from that of HMI device.

If this is the case, update the operating system on the HMI device. To update the operating system on the HMI device, you need ProSave. A prompt for operating system update will appear automatically if ProSave has been installed and the selected connection supports the update of the operating system.

See also

Overview for loading of projects (Page 2179)

15.3.5.6 Loading via USB interface (RT Advanced, Panels, Comfort Panels)

Introduction

When you load a project via a USB port, connect the configuration PC and the HMI device with a USB cable.

Requirements for transfer via USB

The following conditions must be met to ensure successful data transfer using a USB port:

- Use a USB host-to-host cable, USB 2.0 standard.
- You have installed the provided USB driver.
  You can find the driver on the WinCC Product DVD under "Support\DeviceDriver\USB".
- The HMI device being used is based on Windows CE and has a USB port.

You can find more information on the cables used and the manufacturers/suppliers in the Internet at:

- http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/19142034)

Note

Driver installation

To avoid problems when transferring projects, only use the USB driver provided on the WinCC Product DVD.

Note

Comfort HMI devices

Use the mini USB port to load projects via USB with Comfort HMI devices. This driver is installed automatically.

Restrictions

A project can always be transferred to an HMI device. Simultaneous transfer to several HMI devices is not possible.

15.3.5.7 Load project from external storage medium (RT Advanced, Panels, Comfort Panels)

Introduction

If you cannot establish a direct connection from the configuration PC to the HMI device, copy the compiled project to an external storage medium, such as a USB stick, a memory card or network folder, and then load it onto your HMI device.

This function is available in connection with an HMI device image which is compatible with the TIA Portal version V14 or higher.

NOTICE
Loss of data When you activate the "Firmware upgrade" or "Firmware downgrade" options, then the operating system of the HMI device is updated. Existing data on the HMI device including the HMI device password is deleted. Settings in the Start Center are retained, license keys are saved to the external storage medium before update of the operating system. If required, save this data before loading.

You create the required project data in WinCC by configuring the HMI device and then dragging and dropping or copying and pasting the folder of the HMI device (e.g. "HMI_1 [<*device type*>]") to an external storage medium under "Card Reader/USB memory".

Requirements

- You have started the Start Center on the HMI device.
- The storage medium with the backed up project is inserted in the HMI device.
- For Runtime Advanced you have manually copied the files to a directory and adjusted the configuration path under "Start Center > Settings".

Procedure

1. Click "Settings".
2. Open the "Service & Commissioning" dialog.
3. Select the "Load Project" tab.
4. Press "Next".
   The "Load from external memory device" dialog opens.
5. In the "Accessible devices" group, select the storage medium to which the project data has been backed up.

6. Press "Next".
   An overview of all projects which are compatible with the HMI device and are located on the storage medium is displayed.
7. Select the project that you want to load into the HMI device.

   Note

   Use "Details" to receive additional information about the selected project.

8. Press "Next".
   The HMI device checks whether the project data can be loaded. The result of the check is displayed in the "Load Preview" dialog.
   The project can be loaded to the HMI device if no alarms of the type "Error" are issued.

   Note

   If a downgrade or upgrade is necessary, the "Update OS Image" dialog informs you about the possible loss of data and gives you further instructions.

9. Select "Load", to transfer the project data to the HMI device with the project data.

After the loading process the new project is started on the HMI device.

Alarms in the "Load Preview dialog"

The following messages can be displayed in the "Load Preview" dialog:

- Alarms of the type "Information":
- Alarms of the type "Warning", with options
- Alarms of the type "Error", with options

The following table shows alarms of the type "Information":

Icon	Status	Alarm	Meaning
	Info	Firmware version ... Runtime version ...	Firmware and Runtime version on the HMI device
	Info	Ready For Loading	Project data is suitable for the HMI device

The following table shows alarms of the type "Warning", with options:

Icon	Status	Alarm	Meaning
	Over-write	Select project data	The following lines contain options for overwriting data on the HMI device.
		Recipes	Overwrite recipes of the HMI device with the recipes of the project (optional).
		User administration data	Overwrite the user administration on the HMI device with the user administration of the project (optional).
	Upgrade	Runtime upgrade	Runtime version on the HMI device is older than the Runtime version of the project, versions are compatible, upgrade of Runtime version on the HMI device is optional.

Icon	Status	Alarm	Meaning
	Upgrade	Firmware upgrade	Firmware version on the HMI device is older than the firmware version of the project, versions are compatible, upgrade of firmware on the HMI device is optional.
	Down-grade	Runtime downgrade	Runtime version on the HMI device is newer than the Runtime version of the project, versions are compatible, downgrade of Runtime version on the HMI device is optional.
	Down-grade	Firmware downgrade	Firmware version on the HMI device is newer than the firmware version of the project, versions are compatible, downgrade of firmware on the HMI device is optional.

The following table shows alarms of the type "Error", with options:

Icon	Status	Alarm	Meaning
	Upgrade	Runtime upgrade	Runtime version on the HMI device is older than the Runtime version of the project, versions are incompatible, upgrade of Runtime version on the HMI device is required.
	Upgrade	Firmware upgrade	Firmware version on the HMI device is older than the firmware version of the project, versions are incompatible, upgrade of firmware on the HMI device is required.
	Down-grade	Runtime downgrade	Runtime version on the HMI device is newer than the Runtime version of the project, versions are incompatible, downgrade of Runtime version on the HMI device is required.
	Down-grade	Firmware downgrade	Firmware version on the HMI device is newer than the firmware version of the project, versions are incompatible, downgrade of firmware on the HMI device is optional.
	Down-load	Runtime download	There is no Runtime software on the HMI device, e.g. after update of the operating system. Runtime software must be downloaded.

Note

When loading projects to your HMI device, please note that there are compatible and incompatible firmware and Runtime versions.

If you are loading a compatible firmware and Runtime version of the project to your HMI device, an upgrade or downgrade is optional. You can download the project to the HMI device without an upgrade or downgrade. In this case, you can ignore the alarm of the type "Warning".

If you are loading an incompatible firmware and Runtime version of the project to your HMI device, an upgrade or downgrade is mandatory. Otherwise, you cannot download the project to the HMI device. An alarm of the type "Error" is displayed:

Note

Alarms of the type "Warning" also appear in the event of a potential overwriting of user data and recipes if this setting was not locked in the configuration.

See also

Overview for loading of projects (Page 2182)

Overview for loading of projects (Page 2179)

15.3.5.8 Installing a USB driver in Windows 7 (RT Advanced, Panels, Comfort Panels)

Introduction

To load a project via the USB port, install the USB drivers provided on the WinCC product DVD.

Note

Driver installation

To avoid problems when transferring projects, only use the USB driver provided on the WinCC product DVD.

Note

Comfort HMI devices

Use the mini USB port to load projects via USB with Comfort HMI devices. This driver is installed automatically.

Requirement

- The WinCC product DVD is available.
- You are using a USB host-to-host cable, USB 2.0 standard.
- The HMI device has a USB port.

Procedure

To install the USB driver under Windows 7, follow these steps:

1. Connect the USB host-to-host cable to the PC's USB port.
   The USB cable is detected, but the wizard for the driver installation does not find the driver.

   Note

   Connect the HMI device only when the driver has been completely installed on the PC.

2. Open the Device Manager in the Control Panel.
3. Select "Other devices > USB host-to-host cable".
4. Select "USB host-to-host cable" and then select "Update driver software..." in the shortcut menu.
5. When the "How do you want to search for driver software?" prompt appears, select "Search for driver software on the computer".

6. Click "Update driver software > Browse...".
7. Insert the WinCC product DVD as the source for driver installation.
8. Select "Include subfolders" and then click "Next".
   The system will search the WinCC product DVD for the corresponding driver for the USB host-to-host cable.
   The drivers available on the WinCC product DVD are displayed. The wizard highlights the appropriate driver for Windows 7.
9. Check the selection and click "Next".
10. To continue with the installation, click "Hardware Installation > Continue installation".
    The driver is installed.
11. Click "Finish".
    The installation is complete.

Result

The driver required for the USB host-to-host cable has been installed. To load the project, now connect the HMI device to the USB cable.

15.3.6 Compiling and loading with Multiuser Engineering

15.3.6.1 Compiling and loading with multiuser engineering (overview) (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

When using multiuser engineering for your projects, you should take into account the response when compiling the Runtime projects and the response when downloading them to HMI devices.

You can compile and download to an HMI device in both the server project view and in the local session.

You can find more information about the topic of "multiuser engineering" under "Using Multiuser Engineering".

Basics

For HMI devices and RT Advanced, the following scenarios are possible in multiuser engineering:

- Compiling in the server project view
- Compiling in the local session
- Loading from the server project view
- Loading from the local session

Note

Complete download from the server project view or local session is no different to complete download in a single-user project. With a complete download, the current Runtime project is loaded from the currently active view to an HMI device.

Note

Compiling and downloading in a local session is no different from compiling and downloading in a single-user project.

In principle, you can execute all commands for compiling and loading in multiuser engineering projects:

- "Software (rebuild all)"
- "Compile > Software (only changes)"
- Software (all)

The term "project"

The term "project" has two different meanings in the contexts of compilation and loading. "Project" is the WinCC project on the configuration PC. "Project" is also the Runtime project you create by compiling the configuration data of an HMI device and download to the HMI device.

- WinCC project: contains the configuration data of one or more HMI devices
- Runtime project: contains the compiled configuration data of one HMI device

Rules

The following basic rules apply to compiling and downloading in multiuser engineering:

- The Runtime project that has been compiled in a local session always remains local and is not uploaded to the multiuser server. It cannot be saved in the multiuser server project.
- Only Runtime projects compiled in the server project view can be saved in the multiuser server project.

You can find additional information on Multiuser Engineering on the Siemens YouTube channel: Multiuser Engineering - one team working simultaneously on a project (https://www.youtube.com/watch?v=n4oTZ2Gzg6U).

See also

Compiling in the server project view (Page 2197)

Compiling in the local session (Page 2198)

15.3.6.2 Compiling in the server project view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Basics

Compiling and downloading in the server project view is no different from compiling and downloading in a single-user project.

During the compiling of a project in the server project view, the multiuser server project is blocked. Other users cannot make changes to this server project during this time. The Runtime project compiled in the server project view is stored along with the engineering project in the central multiuser server. Blocking the multiuser server project ensures that the configuration data and the Runtime project remain in sync.

Note

When you compile and save in the server project view, other users obtain the Runtime project you have updated along with the engineering project when they "refresh" their local session. Other users do not have to recompile the changes you have made after an update.

Example: Compiling during check-in

You make changes to a tag in a local session. All prior changes have been compiled in the associated server project.

If there are no compilation errors, both projects - the modified engineering project (with the modified tags) and the compiled Runtime project - are saved in the central multiuser server project with the "Save changes" command.

If you skip compiling during the check-in, the project contains the changes that have been saved on the server.

The next user who creates a local session from the server project or updates an existing local session must compile your two changes in addition to his or her own changes.

Note

Working on a shared project through multiple local sessions increases the probability of error. It is therefore recommended to compile the project at check-in and eliminate any errors that are reported during compiling. In this way, you provide the next user with a project free of errors.

See also

Compiling and loading with multiuser engineering (overview) (Page 2195)

Compiling in the local session (Page 2198)

15.3.6.3 Compiling in the local session (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Basics

Compiling and downloading projects in the local session is no different from compiling and downloading in a single-user project.

Since the local session is a copy of the server project, the first compilation status of the local session is identical to that of the server project. If the server project contains contents that are not compiled or error messages occurred during the compiling, they are transferred to the local session.

Note

It is recommended to compile the project at check-in and eliminate any errors that are reported during compiling. In this way, you provide the next user with a project free of errors and avoid spreading errors.

Updating in the local session

If you update a project in the local session, the local session - including the compilation status - is completely replaced by the content of the server project. Only the changes marked for check-in are retained in the updated local session and generate additional compiling steps in the local session.

Example: Updating the local session

You make changes to a tag in a local session. All prior changes have been compiled in the associated server project.

You update the content of the local session by clicking the "Update" button. After the update, the local session obtains the compilation status of the server project. There are also compiling tasks for the acquisition of the modified tags.

See also

Compiling and loading with multiuser engineering (overview) (Page 2195)

Compiling in the server project view (Page 2197)

15.3.7 Starting Runtime

15.3.7.1 Starting Runtime on the configuration PC (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

You can start a project in Runtime on the configuration PC if Runtime has been installed. The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime.

Note

You can only simulate HMI devices with Runtime Panels on the configuration PC. Select the "Online > Start simulation" menu command.

Note

Exiting Runtime automatically

If automatic transfer is activated on the HMI device and if a transfer is started on the configuration PC, the running project is automatically ended.

The HMI device then switches autonomously to the "Transfer" operating mode.

Deactivate automatic transfer after the commissioning phase so that the HMI device does not inadvertently go into transfer mode.

Transfer mode can cause undesired reactions in the system.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the Start Center.

Requirement

- A project is open on the configuration PC.
- Runtime is installed on the configuration PC.
- There is no project in Runtime on the configuration PC.

Procedure

To start runtime on the configuration PC, follow these steps:

1. Select the desired HMI device in the project navigation.
2. Select the "Online > Start Runtime" menu command.

Result

Runtime is started and the project is displayed on the configuration PC.

15.3.7.2 Starting Runtime Advanced and Panel Runtime (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

You can start the project in Runtime as soon as you have downloaded the project to the HMI device. The project is generally started automatically on the HMI device.

There may be several projects in the file system of the HMI device for HMI devices with Runtime Advanced. You can start one of these projects in Runtime.

The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime. Make sure when defining the Runtime settings "Lock task switching" and "Full screen" that you will be able to stop Runtime again. You can, for example, configure a button with the system function "StopRuntime".

Note

Closing Runtime automatically

If automatic transfer is activated on the HMI device and a transfer is started on the configuration PC, the running project is automatically terminated.

The HMI device then automatically switches to "Transfer" operating mode.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

In order to restrict access to the transfer settings and thus avoid unauthorized changes, enter a password in the "Start Center".

Note

Using encrypted communication connections in Runtime

A Runtime project with encrypted communication connections to S7 PLCs must be loaded for each Windows user who wants to use these connections in Runtime.

Requirements

- WinCC Runtime Advanced or Panel Runtime is installed on the HMI device.
- The project was downloaded to the HMI device.
- The "Start Center" has been started.

Starting Runtime on a PC

The compiled project is saved in the PC file system with the extension "*.fwc" and is freely accessible.

1. Enter the project file with complete path information in the "Start Center" under "Settings > Configuration path".
2. To start the project automatically after booting the HMI device:
   - Select the required delay time in seconds under "Wait until Autostart".
   - Add the "Start Center" application to the "Autostart" group in the Windows Start menu.
3. Click on "Start" in the "Start Center" to manually start the project in the HMI device.

Starting Runtime on a panel

The project is stored on a panel in a folder you specify in the HMI device transfer settings. The "Start Center" application is started on a panel. The project loaded is started automatically after expiration of the configured delay.

If the project does not start automatically:

1. Click on "Start" in the "Start Center" to start the loaded project.

Refer to the documentation for the HMI device for additional information on startup of projects.

Result

Runtime is started on the HMI device.

15.3.8 Error messages during loading of projects

Possible problems during the download

When a project is being downloaded to the HMI device, status messages regarding the download progress are displayed in the output window.

Usually, problems arising during the download of the project to the HMI device are caused by one of the following errors:

- Wrong operating system version on the HMI device
- Incorrect download settings on the HMI device
- Incorrect HMI device type in the project
- The HMI device is not connected to the configuration PC.

The most common download failures and possible causes and remedies are listed below.

The serial download is cancelled

Possible remedy: Select a lower baud rate.

The download is cancelled due to a compatibility conflict

Possible cause	Remedy
Conflict between versions of the configuration software and the operating system of the HMI device	Synchronize the operating system of the HMI device with the version of the configuration software. To update the operating system on the HMI device, select the "Update operating system" command from the "Online > HMI device maintenance" menu in WinCC. You can also use ProSave. For additional information, refer to the operating instructions for the HMI device.
The configuration PC is connected to the wrong device, e.g. a controller.	Check the cabling. Correct the communication parameters.

Project download fails

Possible cause	Remedy
Connection to the HMI device cannot be established (alarm in the output window)	Check the physical connection between the configuration PC and the HMI device. Check whether the HMI device is in transfer mode. Exception: Remote control
The default communication driver is not listed in the Windows Device Manager.	Check the device status of the COM connection in the properties window of the Device Manager.

Download over MPI/DP interface fails

Possible cause	Remedy
"Configured mode" is set on the CP, for example, if you are using the SIMATIC NET CD.	Set the CP to "PG mode" using the "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the CP. Set the CP back to "configured mode".
On the programming device/PC panel, the "S7ONLINE" access point is not set to a hardware device such as CP5611 (MPI). This may be due to the installation of "SIMATIC NET CD".	Set the access point "S7ONLINE" on the selected device using the "PG/PC Panel" or "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the HMI device. Restore the "S7ONLINE" access point to the original device.

The configuration is too complex

Possible cause	Remedy
The configuration contains too many different objects or options for the HMI device selected.	Remove all objects of a specific type, for example all HTML browsers. Alternatively, remove options such as Sm@rtServer or OPC server.

15.3.9 Adapting the project for another HMI device

Introduction

When you download a WinCC project to an HMI device, WinCC checks whether this is compatible with the HMI device type used in the project. If the types of HMI device do not match, you will see a message before the download starts.

The download is aborted.

Adapting the project for the HMI device

You need to adapt the project accordingly to be able to download the project to the connected HMI device.

- Add a new HMI device in the project tree. Select the correct type of HMI device from the HMI device selection.
- Copy the configured components from the previous to the new HMI device.
  Copy large amounts of components directly in the project navigation and details view.
  For example, copy the "Screens" folder to the screens folder of the new HMI device with the help of the shortcut menu.
- Use the detail view to copy entries in the project tree for which the "Copy" command is not available in the shortcut menu.
- Select the "Recipes" entry in the project tree, for example. The recipes are displayed in the detail view.
- Select the recipes in the detail view and drag them to the "Recipes" entry of the new HMI device. The recipes are copied. You can also select multiple objects in the detail view.
- Configure the components that cannot be copied, e.g. connections, area pointers, and alarms.
- Save the project at various points in time.
- Compile the full project.
- When the compilation is successfully completed, download the project to the HMI device.

Linking references

References to linked objects are included in the copying. The references are linked again once the linked objects are copied.

Example:

You copy a screen in which objects are linked to tags. The tag names are entered at the individual objects after the screen is added to the new HMI device. The tag names are marked in red because the references are open. When you then copy the tags and insert them into the new HMI device, the open references are closed. The red marking for the tag names disappears.

For complete references to connected objects in the PLC, you first need to configure a connection to the PLC.

Using the information area

When you compile the project for the HMI device, errors and warnings are displayed in the "Info" tab of the Inspector window. You can use the shortcut menu command "Go to" to go directly to the location where the error or warning can be corrected.

Work through the list of errors and warnings from top to bottom.

When the compilation is successfully completed, download the project to the HMI device.

See also

Loading a project (Page 2184)

15.3.10 Reducing the project size (RT Advanced, Basic Panels, Panels, Comfort Panels)

Introduction

When loading a large-scale project to an HMI device, the memory of the HMI device may be insufficient for the project. There are several ways to reduce the size of your project.

Options for reducing the project size

There are several ways to reduce the size of the project and save space:

- Reduce the number of available Runtime languages
  Check whether all selected Runtime languages are actually needed. You can disable the languages that you do not need under "Runtime settings > Language & Font > Runtime language and font selection".
- Do not use help texts for S7 diagnostic alarms
  To reduce the size of the project, you can disable the download of help texts for the S7 diagnostic alarms. In order to avoid downloading the help texts to the HMI device, disable the option "Download S7 diagnostics help texts" under "Runtime settings > Alarms > General".
- Rebuild all software
  In order to optimize the project data and to clean up obsolete changes, compile the entire project using the "Compile > Software (rebuild all)" command from the shortcut menu of the HMI device.

- Harmonize the presentation using styles (as of WinCC V13)
  It is recommended to harmonize screen objects using styles. Standardize the appearance of screen objects in a project to optimize project data. Use the specified preset or customized style for the configuration of the screen objects throughout the project.
- Enable automatic update of PLC alarms (for S7-1500 controllers and V14 HMI devices)
  To save space, you can specify that the PLC alarm texts are only to be loaded in Runtime when needed. To do this, enable the "Automatic update" option under "Runtime settings > Alarms > Controller alarms". Make sure that automatic update of alarms is also enabled in the corresponding controller.
  The "Automatic update" option is not available on Basic Panels.
  The amount of space that can be saved depends on the number of PLC alarms and the number of Runtime languages.
- Reduce the number of fonts loaded
  Check whether the number of downloaded user-defined fonts can be reduced. If necessary, configure only the standard fonts for the required Runtime languages under "Runtime settings > Language & font > Runtime language and font selection".
  To save space, use fewer font groups for the configuration.
- Reduce the size of the graphics
  Check the size of the graphics that you use in the project. If necessary, reduce the size of the graphics by reducing the resolution or choose a higher compression format without noticeable loss of quality for the project graphics.
  To keep the project size small for Basic HMI devices, harmonize the sizes of graphics used in the project.
  Select appropriate graphic formats for your screens: For example, use PNG images for drawings that are not vector graphics and JPEG images for photos.

15.3.11 Supported network cameras and settings (RT Professional, RT Advanced, Panels, Comfort Panels)

Recommended cameras and settings

In the "Camera view" object you display screens of a connected network camera on your HMI device. Use the "Camera view" object with the network cameras which support the streaming protocol RTP/RTPS and codecs MJPEG, H.264 and MPEG4.

Depending on the camera and HMI device, the settings which provide the optimum image quality vary.

Note

Device dependency of the "Camera view" object

The "Camera view" object is available for Comfort Panels, KTP Mobile Panels (as of device version V13) and RT Advanced.

The following tables show you the settings for the tested cameras and the resulting recommendations for optimum display in the "Camera view" object:

Siemens CCMS2025

	Comfort Panels 4"	Mobile F Panels 4"	Comfort Panels 7" - 12"	Comfort Panels 15" - 22"	KTP Mobile Panels 7" - 9"	RT Advanced
Codec	MJPEG	MJPEG	MPEG-4	H.264	H.264	MJPEG
Resolution of camera view	320x240	320x240	640x480	640x480	320x240	800x600
Camera resolution	320x240	320x240	640x480	640x480	320x240	800x600 SVGA
Frame rate (fps)	10	10	10	10	10	Not relevant
Screen delay (seconds)	<1	<1	<1	<1	<1	<1
Quality	Standard	Standard	Standard	Standard	Standard	Standard
Recommendation with panel	Suitable	Suitable	Suitable	Suitable	Suitable	Suitable

AXIS M1011

	Comfort Panels 4"	Mobile F Panels 4"	Comfort Panels 7" - 12"	Comfort Panels 15" - 22"	KTP Mobile Panels 7" - 9"	RT Advanced
Codec	MJPEG	H.264/MJPEG	H.264	MJPEG	MJPEG	MJPEG
Resolution of camera view	320x240	320x240	640x480	640x480	480x360	640x480
Camera resolution	320x240	320x240	640x480	640x480	480x360	640x480
Frame rate (fps)	10	10	10	10	10	Not relevant
Screen delay (seconds)	>1	<1	<1	<1	<1	<1
Quality	Standard	Standard	Standard	Standard	Standard	Standard
Recommendation with panel	Suitable with restrictions	Suitable	Suitable	Suitable	Suitable	Suitable

AXIS M1013

	Comfort Panels 4"	Mobile F Panels 4"	Comfort Panels 7" - 12"	Comfort Panels 15" - 22"	KTP Mobile Panels 7" - 9"	RT Advanced
Codec	MJPEG	MJPEG	MJPEG	MJPEG	MJPEG	MJPEG
Resolution of camera view	320x240	320x240	800x600	800x600	480x360	800x600
Camera resolution	320x240	320x240	800x600	800x600	480x360	800x600
Frame rate (fps)	10	10	10	10	10	Not relevant
Screen delay (seconds)	<1	<1	>1	<1	<1	<1
Quality	Standard	Standard	Standard	Standard	Standard	Standard
Recommendation with panel	Suitable	Suitable	Suitable with restrictions	Suitable	Suitable	Suitable

D-Link DCS-942L

	Comfort Panels 4"	Mobile F Panels 4"	Comfort Panels 7" - 12"	Comfort Panels 15" - 22"	KTP Mobile Panels 7" - 9"	RT Advanced
Codec	MJPEG	MJPEG	MJPEG	MJPEG	MPEG4	MPEG4
Resolution of camera view	320x240	320x240	320x240	640x480	320x240	640x480
Camera resolution	320x240	320x240	320x240	640x480	320x240	640x480
Frame rate (fps)	10	10	10	10	10	Not relevant
Screen delay (seconds)	<1	<1	<1	<1	<1	<1
Quality	Standard	Standard	Standard	Standard	Standard	Standard
Recommendation with panel	Suitable	Suitable	Not suitable	Suitable	Suitable	Suitable

D-Link 4701E

	Comfort Panels 4"	Mobile F Panels 4"	Comfort Panels 7" - 12"		Comfort Panels 15" - 22"	KTP Mobile Panels 7" - 9"	RT Advanced
Codec	H.264	MJPEG	H.264/MJPEG	H.264	MJPEG	MJPEG	MJPEG
Resolution of camera view	320x240	320x240	320x240	640x480	640x480	320x240	960x720
Camera resolution	320x240	320x240	320x240	640x480	640x480	320x240	960x720
Frame rate (fps)	25	25	25	7	25	25	Not relevant
Screen delay (seconds)	<1	<1	<1	<2	<1	<1	<1
Quality	Standard	Standard	Standard	Standard	Standard	Standard	Standard
Recommendation with panel	Suitable	Suitable	Suitable	Suitable	Suitable	Suitable	Suitable

D-Link DCS-4201

	Comfort Panels 15" - 22"	RT Advanced
Codec	MJPEG	MJPEG
Resolution of camera view	640x480	960x720
Camera resolution	640x480	960x720
Screen delay (seconds)	<1	<1
Quality	Standard	Standard
Recommendation with panel	Suitable	Suitable

TL-SC2020

	RT Advanced
Codec	MJPEG
Resolution of camera view	640x480
Camera resolution	640x480
Screen delay (seconds)	<1
Quality	Standard
Recommendation with panel	Suitable

VANDERBILT CCMW3025

	Comfort Panels 4"	Mobile F Panels 4"	Comfort Panels 7" - 12"	Comfort Panels 15" - 22"	KTP Mobile Panels 7" - 9"	RT Advanced
Codec	H.264	MJPEG	MJPEG	MJPEG	MJPEG	MJPEG
Resolution of camera view	320x240 (QVGA)	320x240 (QVGA)	640x480	1280x720	640x480 (VGA)	1280x720
Camera resolution	320x240 (QVGA)	320x240 (QVGA)	320x240	1280x720	640x480 (VGA)	1280x720
Frame rate (fps)	5	5	5	5	5	5
Screen delay (seconds)	<1	<1	<1	<1	<1	<1
Quality	Standard	Standard	Standard	Standard	Standard	Standard
Recommendation with panel	Suitable	Suitable	Suitable	Suitable	Suitable	Suitable

Note

Test results

The optimum settings for RT Advanced were determined in tests on an IPC427D with the Windows 7 Ultimate operating system and cameras with default settings.

The test results are configuration-specific and depend on the hardware, operating system and installed codecs.

You may be able to obtain the missing codecs from the camera manufacturer, for example.

15.3.12 Viewing memory card data

15.3.12.1 Basics (Basic Panels, Panels, Comfort Panels)

Introduction

WinCC provides you with the possibility of viewing data stored on your memory card. The function supports the use of memory cards of the HMI device and of the CPU.

You have the following options:

Viewing a backup (Page 2209)

Renaming and deleting backups (Page 2211)

Viewing HMI device images (Page 2212)

Deleting HMI device images (Page 2213)

Creating HMI device images on memory card (Page 2214)

See also

Viewing a backup (Page 2209)

Renaming and deleting backups (Page 2211)

Viewing HMI device images (Page 2212)

Deleting HMI device images (Page 2213)

Creating HMI device images on memory card (Page 2214)

15.3.12.2 Working with backups

Viewing a backup (Basic Panels)

Introduction

The backup of a Basic Panel that is stored on a memory card can also be viewed in the TIA Portal.

Requirements

- WinCC is installed.
- A memory card with a backup is available.
- The card reader is connected to the configuration PC.
- The project view is open.

Backup on the memory card in the card reader

1. Insert the memory card into the card reader.
2. Open "SIMATIC Card Reader" in the project navigation.
3. Select the card reader drive.
   The "Online Card Data" folder is displayed.
4. Open the "Online Card Data" folder
5. Click the backup to open the shortcut menu.
6. Select "Properties".

Backup on the memory card of the PLC

Proceed as follows if the backup is stored on the memory card of the PLC:

1. Connect the PLC with the configuration PC.
2. Click on the PLC in the project navigation.
3. Select "Connect online" from the shortcut menu.
   A connection to the PLC is established.
   Once the PLC is connected, the "Online Card Data" folder is displayed.
4. Open the "Online Card Data" folder.

   Note

   Accessing a password-protected PLC

   When you attempt to access a PLC that is protected by a password, you will be prompted to enter the password.

   You need at least read access rights in order to view the data that is stored on the memory card.

5. Click the backup to open the shortcut menu.
6. Select "Properties".

Result

The backup properties are displayed in a separate dialog.

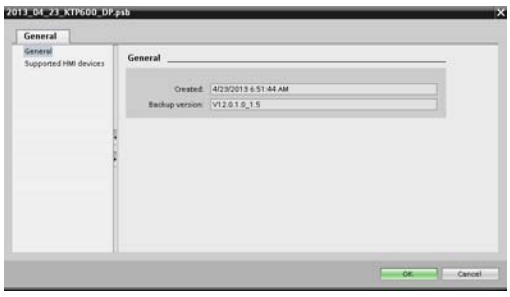

Renaming and deleting backups (Basic Panels)

Introduction

You can rename and delete backups from a memory card in the project navigation of the TIA Portal.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
  The PLC is connected online with the configuration PC.
- A memory card with a backup is available.
- The project view is open.
- The backup is displayed in the project navigation.

Note

Accessing a password-protected PLC

When you attempt to access a PLC that is protected by a password, you will be prompted to enter the password.

You need write access rights to rename or delete memory card data.

Procedure

1. Click on the backup in the project navigation.
2. Open the shortcut menu.
3. Select "Rename" to rename the file.
4. Enter a new name.
5. Select "Delete" to delete the file.

Result

The backup file is now renamed or deleted.

15.3.13 Working with HMI device images

15.3.13.1 Viewing HMI device images

Introduction

The HMI device image of a Comfort Panel that is stored on a memory card can be viewed in the TIA Portal.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
- A memory card with the HMI device image is available.
- The project view is open.

Procedure

1. Insert the memory card into the card reader.
2. Open "SIMATIC Card Reader" in the project navigation.
3. Select the card reader drive.
   The "Online Card Data" dialog is displayed.
4. Open the "Online Card Data" folder.
   The available images of the HMI device are displayed in additional folders.
5. Click the required HMI device image.
6. Select "Properties" in the shortcut menu.

Result

The properties of the HMI device image are displayed in a separate dialog.

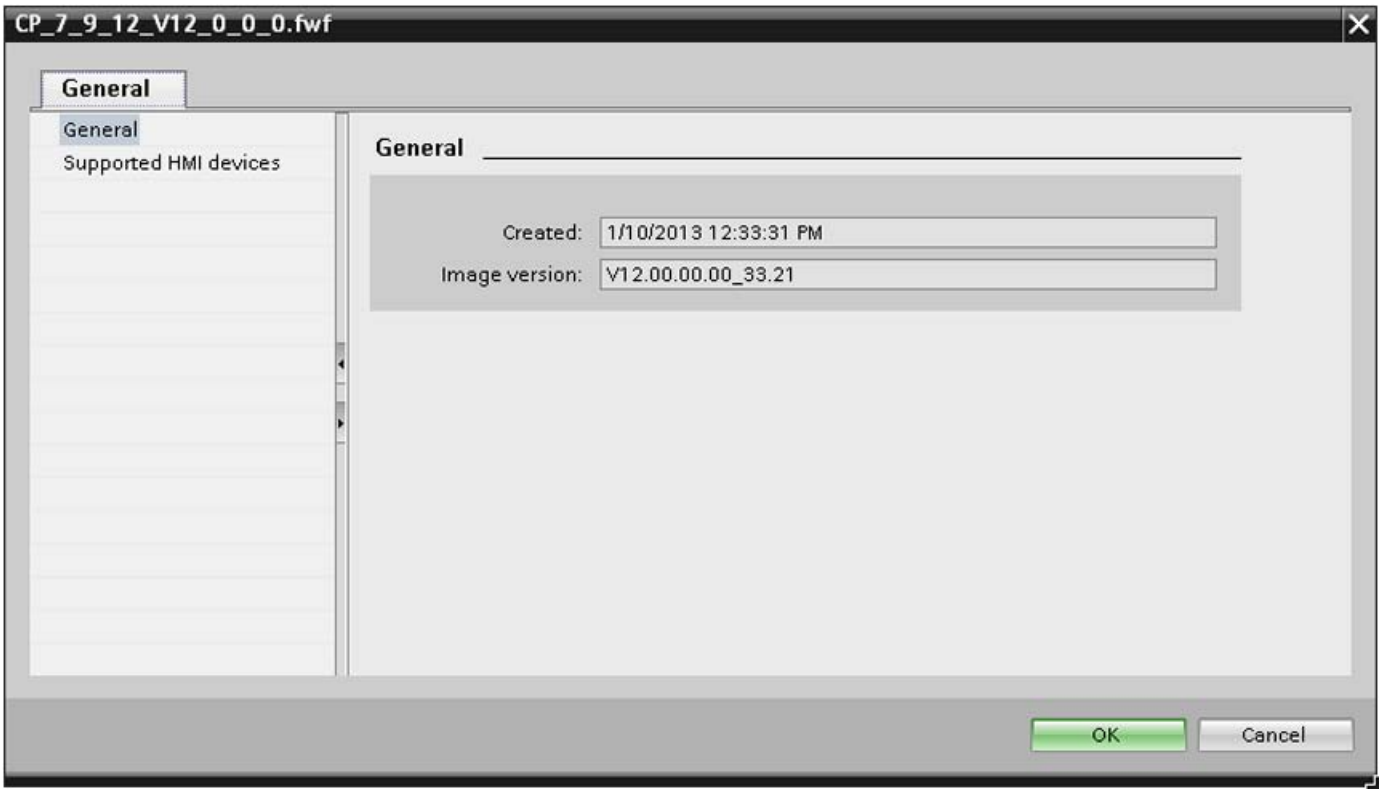

See also

HMI device images V15 and higher (Page 2216)

15.3.13.2 Deleting HMI device images

Introduction

You can delete the HMI device image of a Comfort Panel from a memory card in the project navigation of the TIA Portal.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
- A memory card with an HMI device image is available.
- The project view is open.
- The HMI device image is displayed in the project navigation.

Procedure

1. Click the HMI device image in the project navigation.
2. Open the shortcut menu.
3. Select "Delete" to delete the file.

Result

The HMI device image is deleted.

See also

HMI device images V15 and higher (Page 2216)

15.3.13.3 Creating HMI device images on memory card

Introduction

You may edit the HMI device image of a Comfort Panel without connecting the Comfort Panel to the configuration PC.

Simply create the HMI device image on an external memory card or USB stick and then transfer it from there to the Comfort Panel.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
- The project view is open.

Procedure

1. Insert the memory card into the card reader.
2. Click on the memory card in the project navigation.
3. Open the shortcut menu.

4. Select "Create HMI OS image on memory card".
   A dialog opens.

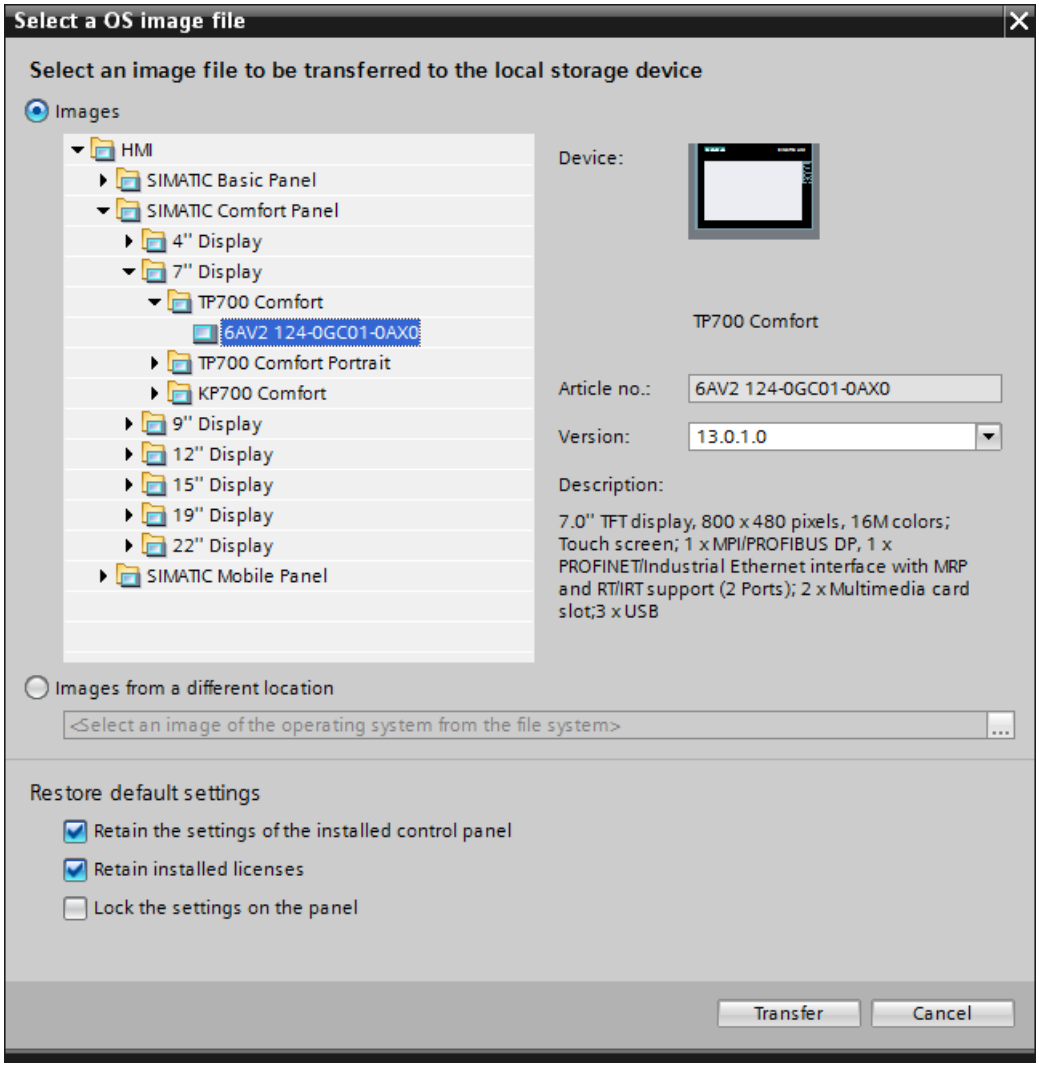

5. Select an HMI device image.
6. Select the settings for the restoration.
   - Activate "Retain installed settings of the Control Panel"
     The settings you have made on the Comfort Panel are retained.
   - Activate "Retain installed licenses":
     The licenses on the Panel are retained.
   - Activate "Lock settings on the Panel"
     You can no longer change the selected settings for "Retain installed settings of the Control Panel" and "Retain installed licenses" on the Comfort Panel.

Result

You have created an HMI device images on memory card. You may use a USB stick instead of a memory card.

See also

HMI device images V15 and higher (Page 2216)

15.3.13.4 HMI device images V15 and higher

Introduction

As TIA Portal V15 and higher not all HMI device images of the previous versions are included in the standard WinCC installation. This helps us reduce the size of the TIA Portal installation package and the subsequent updates compared to the previous versions. The HMI device images that are no longer included in the WinCC installation are supplied on two additional DVDs, DVD 1 and DVD 3, and can be installed at any time.

You also have the option of downloading the respective HMI device image for installation from the SIOS.

Availability of the HMI device images

The following table shows the availability of the HMI device images in combination with the device version:

HMI devices	V12.0	V13.0	V13.0.1	V13.0.2	V14.0	V14.0.1	V15.0
Basic Panels	DVD 1	-	-	-	-	-	-
Mobile Panels	DVD 1	-	-	-	-	-	-
Basic Panels 2nd Generation	-	DVD 3	DVD 3	DVD 3	DVD 3	DVD 1	DVD 1
2nd Genera-tion Mobile Panels	-	-	DVD 3	DVD 3	DVD 3	DVD 1	DVD 1
Comfort Panels	DVD 1	DVD 3	DVD 3	DVD 3	DVD 3	DVD 1	DVD 1

Installation options

Note the following options available for selection when installing TIA Portal:

- "SIMATIC WinCC Panel Images (current)": Can be selected during installation of DVD 1
- "SIMATIC WinCC Legacy Panel Images for V15.0": Can be selected during installation of DVD 3

The following table shows the availability of the device versions in combination with the two options that can be selected during installation:

DVD	Installation option	Available device version
DVD 1	"SIMATIC WinCC Panel Images (current)" activated	V12.0, V14.01., V15.0
DVD 1	"SIMATIC WinCC Panel Images (current)" deactivated	V12.0

DVD	Installation option	Available device version
DVD 1 and DVD 3	"SIMATIC WinCC Panel Images (current)" deactivated "SIMATIC WinCC Legacy Panel Images for V15.0" activated	V12.0, V13.0, V13.0.1, V14.0, V14.0.1, V15
DVD 1 and DVD 3	"SIMATIC WinCC Panel Images (current)" activated "SIMATIC WinCC Legacy Panel Images for V15.0" deactivated	V12.0, V13.0, V13.0.1, V14.0

If the respective HMI device image was not installed during installation of WinCC, the following actions can only be executed after the missing HMI device image has been installed:

- Create a "Pack&Go" file for the respective HMI device.
- Trigger an OS update for the respective HMI device.
- Download a project offline from a USB data storage medium to the respective HMI device.
- Download a project to the respective HMI device.
- Create an HMI device image on the memory card for the respective HMI device.

An alarm informs you that the missing HMI device image must be installed first.

See also

Viewing HMI device images (Page 2212)

Deleting HMI device images (Page 2213)

Creating HMI device images on memory card (Page 2214)

15.3.14 Basics of operating in Runtime

15.3.14.1 Overview (RT Advanced, Panels, Comfort Panels, Multipanels, Mobile Panels)

Operating options for an HMI device

Depending on your HMI device, operate your plant as follows:

- Operation with the touch screen
  The display of the device is touch-sensitive. You operate the operating elements in the display with your finger or a stylus.
- Operation with control keys and function keys
  Control keys and function keys are integrated into the housing of the device.
  - Control keys have a specified function, for example, navigation or the acknowledgment of alarms.
  - Function keys are freely user-assignable and their function is therefore project-specific.
- Mouse and keyboard operation
  You connect the mouse and keyboard to the HMI device and operate the operating elements in the display as with a PC.

Individually configured operation

The configuration engineer has various options available for setting up operation.

Examples of actions whose execution is always determined on a project-specific basis:

- Screen change
- Reporting
- Changing Runtime language

There are no specific operating elements to execute certain functions. The configuration engineer specifies the project-specific execution. A screen change can be triggered with a button or a function key, for example.

Information on project-specific operations can be found in the system documentation.

15.3.14.2 Operation with the touch screen

Overview of operation with the touch screen

Use the touch screen to operate the HMI device of the project that is running on your HMI device.

Operating the touch screen

NOTICE
Damage to the touch screen Do not touch the touch screen with sharp or pointed objects. Avoid tapping the touch screen with hard objects, and avoid constantly using motion control. Both can significantly reduce the useful life or even cause the failure of the touch screen. **Triggering unintended actions** You can trigger unintended actions if you touch several operating elements at the same time. Always touch only one operating element on the screen. Operating elements are touch-sensitive symbols on the screen of the HMI device.

Special features of operation using the touch screen

Operation with the touch screen is characterized by the following special features:

- Enable
  To enable an operator control, touch the touch screen with your fingers or with a stylus. To generate a double-click, touch the operator control twice in rapid succession.
- Value input
  You enter numbers and letters on the touch screen with a screen keyboard.
- Careful operation
  If you touch multiple operator controls at the same time, you may trigger unintentional actions.

Value input using the screen keyboard

The screen keyboard is displayed as soon as you operate a screen object that requires an input. Depending on the HMI device and the configured operating element, the system displays different screen keyboards for entering numerical or alphanumerical values. The screen keyboard is hidden again when input is complete.

File browser (Windows 8 or higher)

You can only operate the file browser dialog with a mouse, keyboard or on-screen keyboard (without using the touch function) on a Windows 8 or higher PC with touch screen. Use the file browser dialog of the Windows operating system with the help of a script on a touch screen PC with Windows 8 or higher.

Screen keyboard (RT Advanced)

Layout

The following figure shows the principal structure of a screen keyboard on a Runtime Advanced HMI device.

Alphanumeric screen keyboard

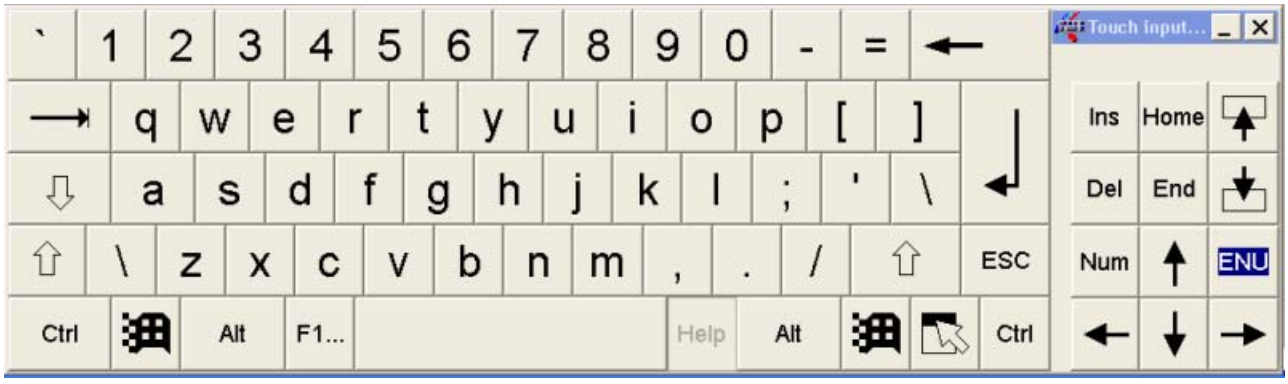

Note

Language switching

Language switching in the project has no influence on the alphanumerical screen keyboard. The entry of Cyrillic or Asian characters is therefore not possible.

Numerical screen keyboard

Operator controls

The following keys are available on the screen keyboard of all HMI devices:

Button	Name	Function
	Cursor left	Navigates to the left
	Cursor right	Navigates to the right
BSP	Backspace	Deletes a character
ESC	Escape	Cancels the input
	Enter	Confirms the input
Help	Help	Displays the infotext This key is only displayed when an infotext has been configured for the operator control.

Using multi-touch functions

Multi-touch operation

Introduction

RT Advanced supports multi-touch operation under Windows 7, Windows 8 and Windows 10. You operate objects on the multi-touch screens by touching them with one or two fingers. Multi-touch operation is supported by TIA Portal as of V13. Versions V12 or older only support single-touch operation.

Requirements

The used monitor must support multi-touch functions.

Scrolling in lists and controls

You can scroll through lists and controls by dragging the screen margin.

You use horizontal dragging to move the content of the screen to the left or right. You use vertical dragging to scroll up or down in the view. An indicator appears during scrolling to indicate your position. When you drag a list diagonally on the screen, the content is moved horizontally and vertically at the same time.

Horizontal scrolling is not supported in the "Trend view" object.

You scroll five times faster up or down on a page when you use two fingers to drag up or down.

Note

Scrolling in Windows 8.1

On the multitouch HMI devices with Windows 8.1, you can scroll continuously by double-tapping on the display and then dragging.

Note

Current view is not persistent

The position in the trend window changed by scrolling is not saved.
In case of a screen change, the trend view is reset to the default setting.

Zoom in and out

You enlarge or reduce the view in "Trend view" and "f(x) trend view" objects by pinch-to-zoom with two fingers.

Double tap to switch from the magnified trend view back to the normal view.

The zooming function is limited to the time axis in the "Trend view" object.

If you have enabled the option "Range > Auto-size" during configuration of the value axes in f(x) trend view, the axes are constantly calculated during zooming.

Note

Current view is not persistent

The changes of the zoom factor are not saved.

In case of a screen change, the trend view is reset to the default setting.

Supported gestures

Supported gestures

The following gestures are supported.

Gesture		Function
	Tap	You can select a button, for example, by tapping.
	Drag	Use one finger to scroll horizontally and vertically, for example, in lists. You scroll horizontally and vertically at the same time with a vertical drag.
		Use two fingers to scroll up or down a page quickly.
	Scale	Pinch-to-zoom to enlarge or reduce the display of the control.
	Double tap	Double tap the display to reset the view to the default zoom factor 100%.

Note

Operating with three of more fingers

Do not use three or more fingers to operate the device. This can lead to incorrect operations.

Two-hand operation of operator controls

Two-hand operation of operator controls

Introduction

WinCC supports two-hand operation of operator controls with RT Advanced. It ensures safe operation of operator controls which are used to change critical system settings, for example, control tags with machine limits.

Locked and unlocked operator controls

You define specific operator controls as "locked operator controls" for two-hand operation of operator controls. Locked operator controls usually cannot be operated in runtime. The operator can only operate the locked operator controls if he presses a release button at the same time.

In runtime, locked operator controls can only be accessed with the tab sequence if a release button is pressed at the same time.

Locked operator controls and release buttons

You can configure the following operator controls as locked operator controls:

- I/O field as input field or input/output field
- Button (visible)
- Button (invisible)
- Slider
- Date/time field

You must configure at least one button in the screen as release button. This can be any unlocked button.

Note

You cannot configure operator controls or release buttons in the permanent area.

Templates with locked operator controls

You can also use locked operator controls in screen templates. To lock or unlock operator controls in templates, proceed as you would with operator controls in screens.

You can also implement a release button in the template. Use the same approach as for release buttons in screens.

In a screen created from the template, the release buttons from the template and any release buttons of the screen are active.

Display in runtime

The locked operator controls are displayed as dimmed in runtime. The release buttons are optically highlighted as soon as the operator presses a locked operator control. The locked operator controls are completely visible when they are unlocked by means of the release buttons.

This means you can configure the release buttons as invisible operator controls.

Simulation of projects with multi-touch functions

WinCC supports the simulation of configured multi-touch functions. Requirement is that your monitor supports multi-touch operation.

Locking and unlocking operator controls

You can lock and unlock operator controls in projects for multi-touch devices. Locked operator controls can only be operated in runtime if the operator presses a release button at the same time.

You can lock and unlock individual operator controls or several operator controls simultaneously.

Procedure

1. Configure operator controls of the type I/O field, button or slider.
2. Select the required operator control(s).

3. To lock the operator controls, enable the "Activate two-hand operation" option under "Properties > Properties > Security".

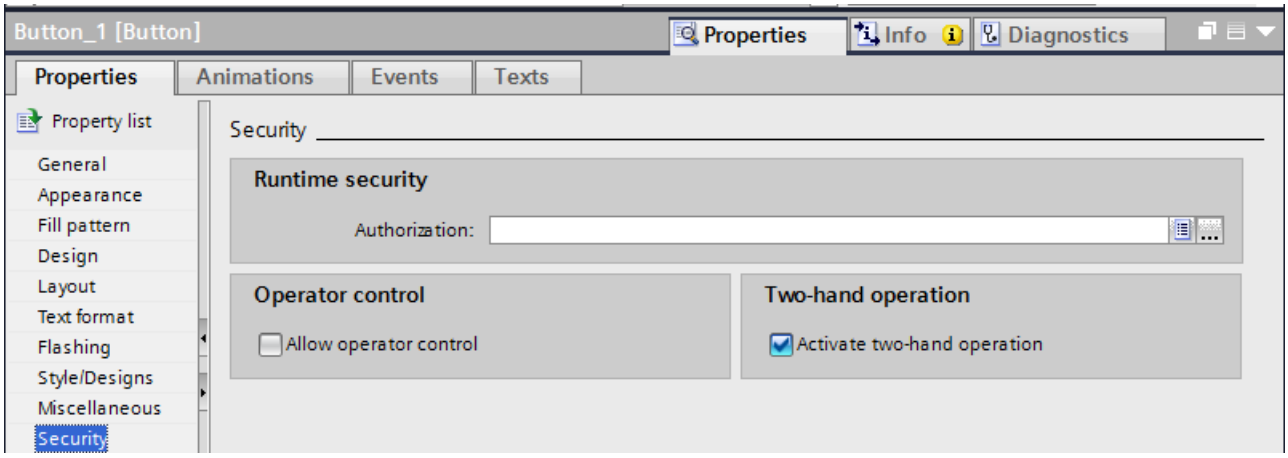

4. To unlock the operator controls, disable the "Activate two-hand operation" option under "Properties > Properties > Security".

In runtime, locked operator controls can only be operated if a release button is pressed at the same time.

Defining release button

To use the locked operator controls, you must configure at least one release button in the same screen.

If no release button is configured, WinCC makes you aware of this fact during compilation of the project.

Configuring release button in screen or template

For the operator to operate locked operator controls on multi-touch devices, at least one release button must be configured in the same screen.

If the screen is based on a template and a release button is already defined in the template, the release button from the template is automatically in effect as release button of the screen. You can then configure an additional release button in the actual screen.

Requirement

At least one unlocked button is configured in the screen or in the template.

Procedure

1. Select the required button of the screen or template under "Release button" under "Properties > Properties > General".

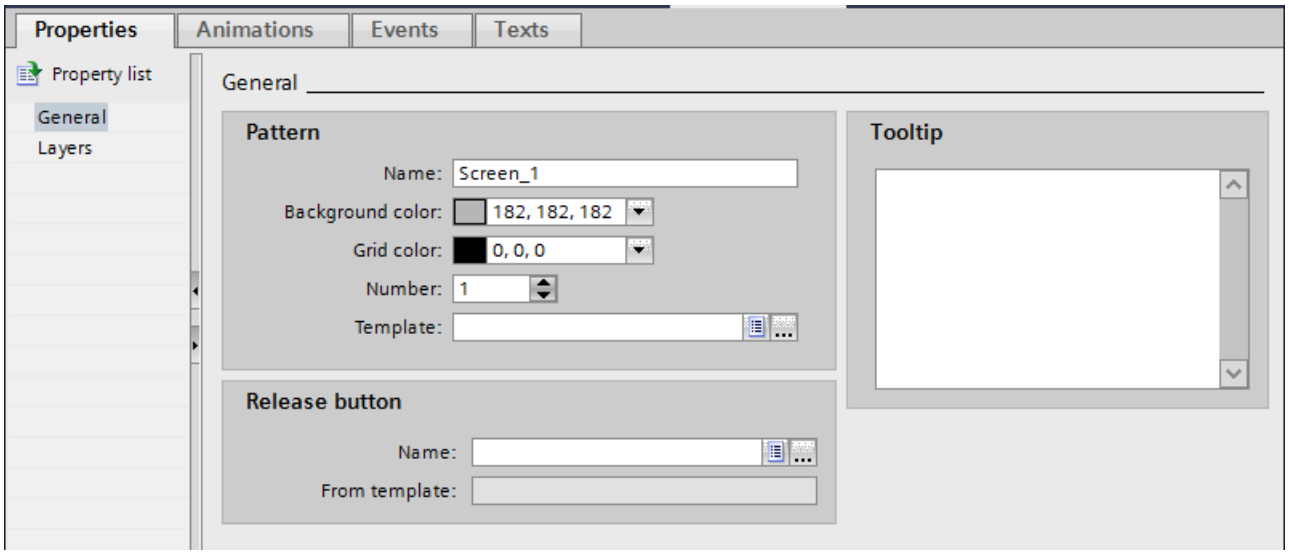

Note

You can also select a button from a group for this purpose. Buttons from faceplates, however, are not available.

The selected button can also be displayed as invisible. It is optically highlighted in runtime when you press a locked operator control.

If the screen is based on a template and a release button is already defined in this template, this button is automatically in effect as release button of the screen.

Locked operator controls can only be operated in runtime if the operator presses one of the configured release buttons at the same time.

Restrictions in Sm@rtServer/Client mode

Multi-touch server and single-touch client

The single-touch client device does not support operation by touch with two or more fingers as well as multi-touch functions. Single-touch client cannot be operated with gestures of the multi-touch Sm@rtServer.

Single-touch gestures work the same way as in WinCC V12.

Single-touch server and multi-touch client

The single-touch server device does not support operation by touch with two or more fingers as well as multi-touch functions. Multi-touch client cannot be operated with gestures of the single-touch Sm@rtServer.

Single-touch gestures work the same way as in WinCC V12.

15.3.14.3 Operation with keys

Overview of operation with keys

Introduction

Use the keys of the HMI device to operate the Control Panel / Start Center of your HMI device or the project that is running on your device. Depending on the device, control keys and function keys are available.

For more detailed information, refer to the operating instructions for your HMI device.

Control keys and shortcuts

Introduction

The following tables list the control keys available to operate the project.

Note

The availability of control keys is determined by the HMI device used.

You trigger functions on keyboard HMI devices either with a key or a shortcut. With shortcuts, you keep the first key pressed. Then you press the second key.

Navigating in the display

Key or shortcut		Function	Description
TAB		Tabulator	Selects the next operating element in the tab sequence
SHIFT	TAB	Tabulator	Selects the previous operating element in the tab sequence
◀ ▶ ▼ ▲		Cursor keys	Selects the next operating element to the left, to the right, above or below the current screen object Navigating in the operating element

Operation of operating elements

Key or shortcut		Function	Description
ENTER		ENTER key	• Controls buttons. • Applies and ends an entry • Opens a selection list • Toggles between character mode and standard mode within an input field A single character is selected in character mode. In this mode, you can scroll within the character set using the cursor keys.
SHIFT	▶ ◀ ▲ ▼	Positioning cursor	Positioning the cursor within an operating element, for example, in an I/O field
←		Delete characters	Deletes the character to the left of the current cursor position
INS DEL		Delete characters	Deletes the next character to the right of the current cursor position
ESC		Cancel	• Deletes the input characters of a value and restores the original value • Closes the active dialog.
FN	HOME	Scroll to start	Scrolls to the start page of a list
HOME		Scrolling back	Scrolls the list back by one page
FN	END	Scroll to end	Scrolls to the end of a list.
END		Scrolling up	Scrolls the list up by one page
ALT	▼	Open selection list	Opens a selection list
CTRL	ENTER	Accept value	Accepts the value selected in the selection list without closing the list

Enter shortcut

Key	Function	Purpose
A-Z	Toggle (numbers/letters)	Toggles the assignment from numbers to letters • No LED is lit: The number assignment is enabled. Pressing the key once switches to letter assignment. • The right or left LED is lit: The left or right letter assignment is enabled. Each time the key is pressed, the system toggles between the left letter assignment, the right letter assignment and the number assignment.
SHIFT	Shift (upper/lower case)	Used in shortcuts, for example, for switching to uppercase letters
FN	Toggle to additional keyboard layout	Some of the keys contain a blue special character in their bottom left corner, for example, the percent sign "%". To input these characters, press the relevant key in combination with the special character key shown on the left.
CTRL	General control function	Used in shortcuts, for example, for navigating trend views
ALT	General control function	Used in shortcuts, for example for the "Watch table" screen object

Acknowledge alarms

Key	Function	Purpose
ACK	Acknowledge	Acknowledges the currently displayed alarm or all alarms of an alarm group The LED is lit as long as an unacknowledged alarm is active.

Displaying infotext

Key	Function	Description
HELP	Displaying infotext	For the selected object, opens a window with the configured infotext, for example, for an alarm or an I/O field The LED is lit if an infotext is available for the selected object.

Key or shortcut

Function Keys

The assignment of the function keys (F1, F2, F3, etc.) is defined during configuration.

Function keys with global function assignment

A globally assigned function key always triggers the same action on the HMI device or in the PLC regardless of the screen displayed. The activation of a screen or the closing an alarm window, for example, is such an action.

Function keys with local function assignment

A function key with local function assignment is screen-specific and is therefore only effective within the active screen.

The function of a locally assigned function key can vary from screen to screen.

Within a screen, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Operating function keys

Note

Operating function key after screen change

If you press a function key after a screen change, the associated function may be triggered in the new screen before the new screen is fully displayed.

Direct keys (Advanced)

Introduction

Direct keys on the HMI device are used to set bits in the I/O area of a SIMATIC S7.

Direct keys enable operations with short reaction times that are, for example, a jog mode requirement.

Note

Direct keys are still active when the HMI device is in "offline" mode.

Note

If you operate a function key with direct key functionality in a running project, the direct key function is always executed, independent of the current screen contents.

Note

You can only use direct keys when there is a connection via PROFIBUS DP or PROFINET IO.

Direct keys result in additional basic load on the HMI device.

Direct keys

The following objects can be configured as a direct key:

- Buttons
- Function keys

You can also define image numbers in the case of HMI devices with touch operation. In this way, the project engineer can configure the direct keys on an image-specific basis.

For additional information on configuring direct keys, refer to "WinCC communication".

Operator controls on the Mobile Panel

Operator controls on the Mobile Panel

The following operator controls are available to you depending on your type of mobile panel:

- Function keys
  The number and assignment of the function keys depend on the selected device type. The respective function is project-specifically assigned.
- Handwheel (optional)
  The handwheel can be turned without an end stop, and has no zero position. In a running project you enter incremental values using the handwheel.
- Key switch (optional)
  The key switch permits locking functions, which you can trigger using the HMI device.
- Illuminated pushbutton
  The function of the illuminated pushbutton is specified by the running project.

For more detailed information, refer to the operating instructions for your HMI device, and to the plant documentation.

15.3.14.4 Navigating in the display (BS)

Introduction

You navigate on the display of your HMI device as follows:

- between configured screen objects
- within screen objects
  When you select a complex screen object, the cursor focus switches to the screen object and follows the tab sequence there.
- in tables of screen objects

Procedure

- To navigate in the specified tab sequence, press the <TAB> key.
- To navigate freely between the operator controls, press the cursor keys.

Depending on the configuration of your HMI device, you can also use function keys or shortcuts for navigation.

When you operate your HMI device with the touch screen or with the mouse, you implicitly navigate by triggering a desired action. For this purpose, touch or click the operator control.

Result

The operator controls receive the cursor focus according to the selected sequence. You can trigger an action on the selected operator control.

For more detailed information, refer to the operating instructions for your HMI device.

15.3.14.5 Triggering an action

Introduction

Triggering an action at an operator control can mean the following:

- A command is executed.
  Example: Click a button to trigger a script or to execute a predefined function.
- An object is enabled.
  Example: To enter a value, select a table cell with the <Enter> key.

Requirement

- You have navigated to the operator control on which you want to trigger the action.
- The operator control has the cursor focus.

Procedure

- Press <Enter>.
  Or
- Touch the operator control on the touch screen once or twice in rapid succession.
  Or
- Click or double-click the operator control with the mouse.

Result

The following results are possible:

- The requested command is executed.
- The screen keyboard is opened and/or the cursor blinks in the input area of the operator control.
- The element is selected and can be moved.

For more detailed information, refer to the operating instructions for your HMI device.

15.3.14.6 Entering a value

Introduction

Depending on the input format, you enter numerical or alphanumerical values in an input field.

You enter these values depending on the existing hardware using the screen keyboard, the control keys of the HMI device or an external keyboard.

Requirement

- The object is an input field or table field.
- The operator control is enabled.

Entering a value

1. Enter the desired value.
2. To confirm the value and exit the field, press the <Enter> key.
3. To discard the value and exit the field, press the <Esc> key.

Result

A value is entered or discarded. You navigate as needed to the next operator control.

For more detailed information, refer to the operating instructions for your HMI device.

15.3.14.7 Moving operator controls

Introduction

In Runtime, you can move the movable operator controls of a screen object with the mouse or using the touch screen, for example, a slider or a scroll bar. Operation with the keyboard is described below.

Requirement

- A movable operator control is selected.

Procedure

- To move the operator control, proceed as follows depending on the operating element:
  - Standard for touch screen: Press the cursor keys.
  - Standard for keyboard devices: Press <SHIFT> and the cursor keys.
  - Switches: Press <ENTER>
  - Slider: Press <PgUp> or <PgDn>

1. To finish the movement, navigate to another screen object or operator control.

Slider procedure

1. To move the operator control, press the cursor keys.
2. To finish the movement, navigate to another screen object or operator control.

Result

The position of the movable operator control and the display in the screen object have changed.

For more detailed information, refer to the operating instructions for your HMI device.

15.3.14.8 Displaying infotext

Introduction

Depending on the configuration, additional information and operating instructions are available as infotext. The infotext is assigned to an operating element, an alarm or to the open screen. The infotext of an I/O field may contain, for example, information about the value to be entered.

As an alternative to the <Help> key of the HMI device, use the <Help> key of the screen keyboard for input objects.

Requirement

- An infotext is configured for the operating element, the screen or an alarm.

Calling the infotext

1. Enable the desired operating element.
2. Press the <Help> key of the HMI device.
   The infotext for the operating element is displayed.

If you are operating your input object with the touch screen, the screen keyboard opens. If the <Help> key appears, an infotext is configured for the operating element or the current screen.

If there is no infotext for the selected screen object, the infotext for the current screen is displayed, if it has been configured.

Use the scroll bar for long infotexts.

Depending on your configuration, infotext can also be retrieved by means of a configured operating element.

Switching between infotexts

- To switch between the infotexts of the operating elements and the screen, enable the infotext window.

Hiding infotext

- To hide the infotext, press the <Esc> key or press the <Help> key again.

15.3.14.9 Changing Runtime language

Introduction

The HMI device supports multilingual projects. A corresponding operating element which lets you change the language setting on the HMI device in Runtime has been configured.

The project always starts with the language set in the previous session.

Requirement

- The required language for the project is available on the HMI device.
- The language switching function is linked to an operating element, for example, to a button.

Selecting a language

You can change project languages at any time. Language-specific objects are immediately displayed on the screen in the new language when you switch languages.

You can switch the language in Runtime in one of the following ways:

- Use a configured operating element to switch from one language to the next in a list.
- Use a configured operating element to directly set the required language.

15.3.14.10 Web browser of WebKit engine: Overview (Panels, Comfort Panels)

Introduction

In the "HTML browser" object, you have the option in RT Advanced to select between two Web browser types. Specify the type of the Web browser under "Properties > Properties > General > HTML browser type" in the Inspector window.

Only the Web browser of the WebKit engine is available on Panels.

The Web browser, which is based on a WebKit engine, offers no Active X support.

HTML5 functions

The following HTML5 standard functions are fully or partly supported by the Web browser of the WebKit engine:

- Parsing rules
- Elements
- Forms and fields
- Output
- Communication
- User interactions
- Performance
- Security
- History and Navigation
- 2D graphics
- Memory
- Animations
- Web applications
- Memory
- Files and file systems

Note

Microdata, input, peer to peer, position and orientation, video, audio, responsive images, 3D graphics, streams and web components are not supported in the Web browser of the WebKit engine.

The table below shows the availability of the HTML5 functions in the Web browser of the WebKit engine in detail:

Parsing rules	Available
<!DOCTYPE html> triggers the standard mode	Yes
HTML5 tokenizer	Yes
HTML 5 tree building	Yes
Parsing Inline SVG	Yes
Parsing Inline MathML	Yes

Elements	Available
Embedded invisible data	Yes
New or modified elements	
Section elements	Yes
Grouping content elements that belong together	Partly

Elements	Available
Semantic elements of the text level	Partly
Interactive elements	Partly
Global attributes and methods	
Hidden attributes	Yes
Inserting dynamic markups	Yes

Forms and fields	Available
Field types	
type = text	Yes
type = search	Yes
type = tel	Yes
type = URL	Yes
type = email	Yes
type = date	No
type = month	No
type = week	No
type = time	No
type = datetime	No
type = datetime-local	No
type = number	Yes
type = range	Yes
type = color	Yes
type = checkbox	Yes
type = image	Yes
type = file	Yes
textarea	Yes
select	Yes
fieldset	Yes
datalist	Yes
keygen	Yes
output	Yes
progress	Yes
meter	Yes
Fields	
Field validation	Yes
Assignment of forms and controls	Yes
Other attributes	Yes
CSS sectors	Yes
Events	Yes
Forms	
Form validation	Yes

Output	Available
Full-screen support	No
Web notifications	Yes

Communication	Available
Server-sent events	Yes
Web beacons	No
XML HttpRequest Level 2	
File upload	Yes
Response type	Yes
WebSocket	
Basic Socket Communication	Yes
ArrayBuffer and Blob	Yes

User interactions	Available
Drag-and-drop	
Attributes	Yes
Events	Yes
Editing HTML	
Editing elements	Yes
Editing documents	Yes
CSS sectors	No
APIs	Yes
Clipboard	
Clipboard for API and events	No
Spell check	
Spelling attributes	Yes

Performance	Available
Native binary data	Yes
Workers	
Web workers	Yes
Shared workers	Yes

Security	Available
Web Cryptography API	No
Content Security Policy 1.0	Yes
Content Security Policy 1.1	No

Security	Available
Cross-Origin Resource Sharing	Yes
Cross-Document Messaging	Yes
Iframes	
Sandboxes Iframe	Yes
Seamless Iframe	Yes
Iframe with Inline contents	Yes

History and Navigation	Available
Session history	Yes

2D graphics	Available
Canvas 2D graphics	Yes
2D primitives	
Text input in graphics	Yes
Path input in graphics	No
Drawing an ellipse in graphics	No
Drawing a dashed line in graphics	Yes
System focus ring	No
Functions	
Hit testing	No
Aperture mode	No
Formats for image export	
PNG	Yes
JPEG	Yes
JPEG-XR	No
WebP	No

Animation	Available
window.requestAnimationFrame	Yes

Web applications	Available
Offline resources	
Application cache	Yes
Service workers	No
Content and scheme handlers	No

Memory	Available
Key value storage	
Session memory	Yes
Local storage	No
Database storage	
IndexedDB	No
Blob object store	No
ArrayBuffer object store	No
Web SQL database	Yes

Files and file systems	Available
Reading files	
Basic support for reading files	Yes
Creating a blob from a file	Yes
Creating a data URL from a blob	Yes
Creating an ArrayBuffer from a blob	Yes
Creating a blob URL from a blob	Yes
Accessing a file system	
API file system	No
File API: Folders and system	No

Additional functions	Available
Styles	
Style items	No
Scripts	
Asynchronous script execution	Yes
Signaling script errors in Runtime	Yes
Events for script execution	No
Base 64 encoding and decoding	Yes
JSON coding and decoding	Yes
URL API	Yes
MutationObserver	"Yes" (pre-selected)
Promises	No
Page visibility	"Yes" (pre-selected)
Text selection	Yes
Scrolling (Scroll into view)	Yes

See also

HTML browser (Page 388)

15.3.15 Entering bar codes with optical handheld readers

Introduction

With the optical handheld readers you can identify components, machines and other objects optically and enter them directly into multiple operating elements on your HMI device.

Optical handheld reading devices read two dimensional data matrix codes, one dimensional barcodes and postal bar codes.

The following optical handheld readers are supported:

- SIMATIC MV320
- SIMATIC MV340

You can find templates for the settings and instructions on configuration in the manual for your optical handheld reader.

Note

The optical handheld reader is connected to the USB port of the HMI device. Only one device at a time can use the USB port. This is why it is not possible to use a USB keyboard and an optical handheld reader or two optical readers at the same time.

Note

When you connect the optical reader to a configuration PC, ensure that the USB port provides sufficient power for the reader.

Procedure

You read the marking of an object using the connected optical handheld reader. The value is entered in the object on the HMI device which has the focus during runtime.

After it has been read in, confirm the value with the Enter key or with the "Suffix - Enter" that you have previously configured in the settings of your optical handheld reader.

Basic Panels, Comfort Panels, Mobile Panels, RT Advanced

The following objects support input with optical handheld readers:

Object	Preconditions for input
I/O field Date/time field	The corresponding data type is selected. The object and the tag length are configured accordingly. The operator element has the cursor focus.
Recipe view	The recipe data record has the cursor focus.

Object	Preconditions for input
User view	The logon dialog is open. The user name or password has the cursor focus.
HTML browser	The operator element has the cursor focus.
Runtime dialogs which support key-board entry	The dialog is open and the corresponding input field has the cursor focus.

Comfort Panels, Mobile Panels, RT Advanced

The following objects support input with optical handheld readers:

Object	Preconditions for input
Sm@rtClient view	Write access is configured for Sm@tClient. The operator element has the cursor focus.
File browser	The field "File path" has the cursor focus.
Watch table	The input field "Control value" has the cursor focus.
PDF view	The search field has the cursor focus.
SINUMERIK NC operator display	The operator element has the cursor focus.

Result

The code is read and entered into the corresponding input field.

15.3.16 Security on the HMI device

15.3.16.1 Signature basics (RT Advanced, Basic Panels, Panels, Comfort Panels)

Overview

The HMI device software and the runtime and device options are protected from manipulation by signatures. The device options available over ProSave are also signed.

HMI device signature

The validity of the HMI device signature is checked when the HMI device software is loaded. If the HMI device signature is invalid, loading of the HMI device software is canceled and an error message is output.

Signature for runtime and device options

If the signature check finds that the signature is invalid, an error message to this effect is output. An error message indicating an invalid signature is also output in ProSave.

Note

To allow loading of the HMI device software or the options without a signature (for versions prior to V14), you will need to switch off the signature check.

For details of how to do this, see Switching off the signature check (Page 2244).

See also

Switching off the signature check (Page 2244)

15.3.16.2 Download of add-ons from third parties (RT Advanced, Panels, Comfort Panels)

Basics

By default, download of add-ons from third parties is not permitted due to a missing signature. To enable the download of add-ons from third parties anyway, you must disable the signature check for the time of the download.

For details of how to do this, see Switching off the signature check (Page 2244).

Note

We recommend that you reactivate the signature check once you have downloaded the add-ons.

See also

Switching off the signature check (Page 2244)

15.3.16.3 Switching off the signature check (RT Advanced, Basic Panels, Panels, Comfort Panels)

Procedure

1. Press "Transfer" twice in the control panel of the HMI device.
   The "Transfer Settings" dialog box opens.
   The option "Validate signatures" is enabled by default.

   Note

   If you wish to load an older WinCC project to an HMI device, you will need to check the settings for the signature check.

2. To allow the HMI device software to be loaded without a signature or to allow the download of a trustworthy option from a third-party provider, disable the "Validate signatures" option.

See also

Download of add-ons from third parties (Page 2244)
Signature basics (Page 2243)

15.4 Runtime Professional

15.4.1 Runtime settings

15.4.1.1 Settings in the Runtime software

Introduction

To edit the Runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree.

Note

Device dependency

The options available for selection in the Runtime settings depend on the HMI device.

Overview

You configure options from different areas of your project in the Runtime settings:

- Display in Runtime
  - Start screen
  - Menus and toolbars
  - Independent screen windows
  - Designs
  - Properties of windows
  - Bit selection for color and flashing
  - Screen buffer
  - Project graphics
- Scripts in Runtime
  - Start even with corrupt script
  - Value source for SmartTags
  - Language of the C scripts with "Use device setting"
- Options for compiling projects
  - Do not check deleted HMI tags in screens: If disabled, a warning is output during compiling when HMI tags are deleted in screens.
  - Hide WebUX warnings

- Start sequence
  Use the start sequence to specify the applications that will be started on activation of the project.
- Operating in Runtime
  - Window controller
  - Availability of online help in Runtime
  - On-screen keyboard
  - Keyboard layout and keyboard shortcuts
  - Cursor control
  - Zoom functions
- Alarms
  - Duration of display suppression
  - Alarm list entries after power failure
  - OPC alarms and events
  - Controller alarms
- Logging
  - Update cycle
  - Properties of log segments
- Configuration of the WinCC OPC servers
- User administration
  - Dynamic logon
  - SIMATIC Logon
- Language and font
  - Available Runtime languages
  - Runtime language for single-language objects
  - Sequence of language switching
  - Default font
  - Character set for menus
- Web access
  - Appearance
  - Server utilization
  - Local user groups
  - Alarms in the event view
  - Licenses

  For more information on the Runtime settings, refer to the online help for the options "WebNavigator", "WebUX" and "DataMonitoring".

- Settings for synchronization of PLC and HMI tags
- Redundancy
  - Redundant servers and partners
  - Client switchover
  - Down time for synchronization
  - Synchronization of logs and alarms
  - Recipes

  You can find additional information on the configuration of redundant servers in the online help for the option "Redundancy".
- Process Historian
  - Server selection
  - Buffer path

  You can find additional information on the Runtime settings in the online help for the option "Process Historian and Information Server".

See also

Configuring display in Runtime (Professional) (Page 2248)

Setting up the start sequence (Page 2258)

Setting the Display Suppression Timeout (Page 863)

Settings for the user administration (Page 1322)

Settings for synchronization (Page 544)

Configuring operation in Runtime (Page 2250)

Working with Logs (Page 1033)

Managing languages (Page 2012)

15.4.1.2 Configuring display in Runtime (Professional)

Introduction

In the Runtime settings you specify how the start screen, independent screen windows, menus and toolbars, etc. will be displayed.

To edit the Runtime settings for your HMI device, select "Runtime settings" under your HMI device in the project tree. The options available for selection depend on the HMI device.

Start screen settings

The start screen is the initial screen that is displayed after Runtime starts. The screen resolution is automatically adjusted to suit the HMI device when the screen is selected. You specify the start screen in the Runtime settings of the HMI device under "General > Settings".

You can find additional information in the online help for WinCC under "Defining the project's start screen".

Design settings in the Runtime settings

Under "General > Settings", you specify whether the HMI device uses the default design of the project or another predefined design. You can find more detailed information in "Working with designs".

Design settings in the Windows Control Panel

Select the "Windows 7" design for the configuration PC and the Runtime PC in the system settings of Windows. Use this setting to ensure correct representation of the dialogs and the on-screen keyboard in Runtime.

Menu and toolbar settings

Under "Screen management", you configure user defined menus and toolbars in the "Menus and Toolbars" editor.

For menus and toolbars, you make the following settings:

- Text
- Graphic
- Data: These data are passed as user data to the function.

For toolbars, the following setting options are also available to you:

- Button type: You specify whether the button consists of a graphic only, text only, or both.
- Screen size
- Transparent colors: You specify the color value that will be displayed as transparent in the selected graphic.

You can find additional information on this subject in the online help for WinCC under "Overview of working with menus and toolbars".

Settings for independent screen windows

If you enable the "Hide main window" option under "General > Independent screen windows", you specify that Runtime runs in the background. The Runtime window is hidden and is also not visible in the taskbar.

Settings for screens

You change the settings of the Runtime window under "Screens" and specify how additional functions are used.

You also make general settings. The general settings include:

- Use background effects in Runtime
- Use shadows for screen objects in Runtime

- Use WinCC Classic Design in Runtime
- Use bit selection for the design

Under "Screen buffer", you also specify how many screens are temporarily stored. The screens saved in the screen buffer are displayed using the screen navigation keys.

Graphic settings

Under "Graphic settings > Loaded project graphics", you specify which graphics from the project graphics are downloaded to the HMI device.

15.4.1.3 Configuring operation in Runtime

Configuring operation in Runtime (Professional)

Introduction

Numerous functions are available to you for operation in Runtime. For some of these functions you can define specific shortcut keys.

Window control keys

These shortcut key changes are activated after system restart.

Changing windows

This shortcut key is used to navigate between multiple screen windows, which are configured in a main screen. Whenever the shortcut key is actuated, the next screen window will be activated for an operation.

Note

The <DEL> and <HOME> keys cannot be used as shortcut keys.

Making help available in Runtime

Under Runtime settings > General, you specify that online help is available in Runtime. For this purpose you enable the "Make help available in Runtime" option.

Some objects in Runtime have an online help system. You can centrally disable calls to online help. You thereby prevent the online help on the hard disk from being accessed in Runtime.

Preventing access to Windows in Runtime

To prevent the user from accessing the Windows user interface with familiar shortcuts, such as <Ctrl>+<Alt>+<Del>, activate the function "Disable keyboard commands" under "Runtime settings > Keyboard > Keyboard commands of the operating system". In addition, deactivate the "Keep the taskbar on top of other windows" setting in Windows.

Using the screen keyboard

If you enable the "Use screen keyboard" option under General, the screen keyboard will be displayed in Runtime.

To enter data in Runtime, a screen keyboard will be displayed on the screen. The operator presses the individual keys with the mouse, or directly via the touch screen.

Note

The screen keyboard is not available with all HMI devices.

Disabling keyboard commands

The "Disable keyboard commands" function is not supported by the activatable Windows on-screen keyboard. Instead, use the on-screen keyboard provided by WinCC Runtime Professional.

Start even with corrupt scripts

Using the "Start even with corrupt scripts" option, you specify that Runtime Professional will be started even when a function in the project is corrupt.

Load names

Under General, you use the "Load names" option to specify that the objects' name information will be loaded along with them.

If the check box is activated, the objects' name information will be transferred to the HMI device.

If the check box is deactivated, then instead of the objects' name information only encoded addressing information will be transferred to the HMI device.

You need the objects' name information, if you want to address the objects in a script by the object name. When you test a script in the debugger, the code becomes more comprehensible by displaying the object names.

Hide main window

The "Screen window" object is used to display other screens from the project in the current screen. You can find the "Hide main window" function under "General > Independent screen windows". With this option, you specify that Runtime runs in the background. The Runtime window is hidden and is also not visible in the taskbar.

Select keyboard layout (Professional)

Introduction

You can define various shortcuts to operate the project. The shortcuts can be used as an alternative to the mouse.

- Keyboard commands of the operating system
  In the "Keyboard commands of the operating system", specify whether the operating system can be accessed with keyboard shortcuts.
- Action keys
  - "Logon" opens a window for user logon in Runtime.
  - "Logoff" opens a window for user logoff in Runtime.
  - "Hardcopy" opens a dialog for printing the screen in Runtime.
- Windows control keys
  In the "Windows control keys" area, determine whether the next screen window is activated using a specified keyboard shortcut in Runtime.
- Screen navigation keys
  - "Forward" moves to the next screen in the configured screen sequence.
  - "Back" moves to the previous screen in the configured screen sequence.
  - "Store screen" saves the currently displayed screen.
  - "Recall screen" opens to the previously saved screen.
  - "Start screen" opens the start screen.
- Keys for tab sequences
  - "Tab sequence mode" changes the cursor mode.
  - "Focus display of active object" activates the runtime cursor.
- Cursor control keys
  - "Enable hotkeys in Runtime" activates the shortcut keys for cursor control in Runtime.
  - "Up" makes a cursor movement up.
  - "Down" makes a cursor movement down.
  - "Left" makes a cursor movement left.
  - "Right" makes a cursor movement right.

Requirement

The "Runtime settings" editor is open.

Procedure

Set up the shortcuts as follows:

1. Click on "Runtime settings > Keyboard".
2. Select the desired action in the "Action keys" area, for example.

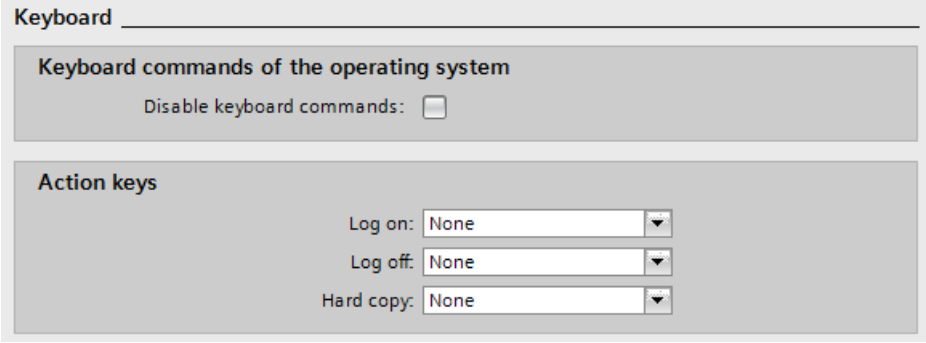

3. Open the drop-down list.
   The dialog for defining the shortcut is opened.
4. Define the required shortcuts.

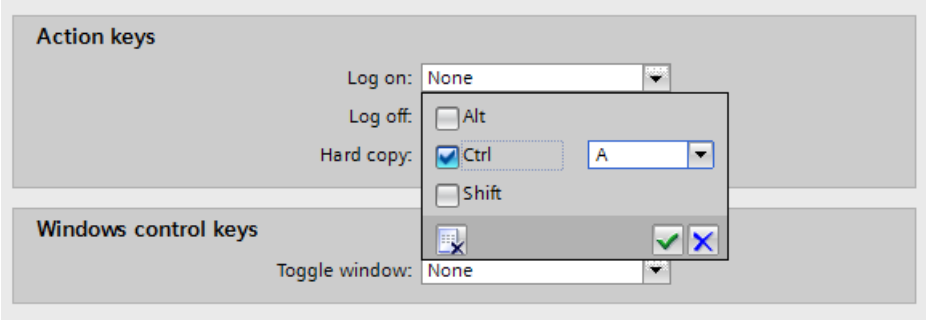

5. Accept the settings.
   The dialog for specifying the shortcut closes itself.

Configuring shortcut for operating and screen navigation

Introduction

Screens can also be operated if mouse control is not available. Define corresponding shortcuts for the required operating functions. The operator calls the specific function by operating the shortcut.

The most important shortcuts for operation without mouse in Runtime are defined in the "Runtime settings > Keyboard" editor.

Procedure

Set up the operator shortcuts as follows:

1. Click in the input field of the selected function.
2. Click the button in the selection list. The dialog for defining shortcuts opens.
3. Define the required shortcuts.
4. Click the button to close the dialog.

You can also define one of the function keys <F1> to <F12> instead of a shortcut.

Note

The function key F12 may not be configured as a system wide hotkey.

Window control keys

These shortcut changes are activated after system restart.

Changing windows

This hotkey is used to navigate between multiple screen windows, which are configured in a main picture. Whenever the hotkey is pressed, the next screen window is activated for operation.

Note

The <DEL> and <HOME> keys cannot be used as hotkeys.

Screen navigation keys

The "Forward" and "Back" buttons of most Internet browsers allow the navigation between Internet pages most recently called. A similar technique can be implemented in WinCC to navigate between individual process screens.

Configure corresponding shortcuts in the "Screen selection" dialog to facilitate screen navigation in Runtime. Screens are logged to a temporary list after they are called. You can page through this list using the "Forward" and "Back" functions, for example, to recall the last five screens. The valid maximum number of entries in this list is set in the field "Screen selection buffer". If this number is exceeded, the oldest entry is replaced after a new screen is called.

Note

The list logs up to 30 screen calls if the default is set for "Screen selection buffer". As a rule, it does not usually make sense to substantially increase this value, since the switchover can only be done in single steps. You can achieve more efficient screen navigation using faceplates.

Configuring the cursor control (Professional)

Introduction

You can configure object navigation without mouse using defined cursor control functions in screens with tabular layout of the objects.

Define the cursor control shortcuts in the "Runtime settings > Keyboard" editor. You specify the response of the cursor in tables under "Screens > Cursor response in tables".

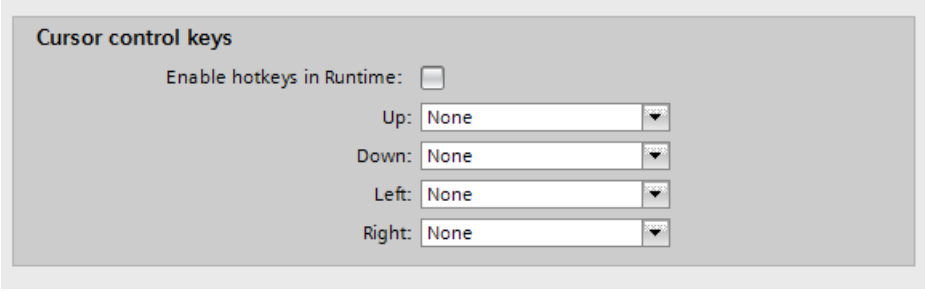

Procedure

Set up the operator shortcuts as follows:

1. Click in the input field of the selected function.
2. Open the drop-down list.
   The dialog for defining the shortcut is opened.
3. Define the required shortcuts.
4. Accept the settings.
   The dialog for specifying the shortcut closes itself.

You can also define one of the function keys <F1> to <F12> instead of a shortcut.

Activating the zoom functions (Professional)

Introduction

Zooming in Runtime is supported by three techniques:

- Declutter
  The layers and their embedded objects can be shown and hidden.
  For additional information, refer to "Dynamic visualization of an object (Page 219)".
- Extended Zooming
  A screen's view in Runtime can be enlarged or reduced using the scroll wheel of the mouse. For this purpose, hold down <CTRL> and move the mouse scroll wheel in the desired direction.
- Panning
  If a screen has a zoom factor that causes the screen to be displayed with scroll bars, then you can move a screen detail within the document. A navigation crosshair appears upon clicking on the mouse wheel. Moving the mouse pointer scrolls in the desired direction. The distance between mouse pointer and navigation crosshair determines the scrolling speed. A second click cancels panning.

Requirement

- A driver for a Logitech mouse or Microsoft Intellimouse
- The mouse wheel must be set to "Autoscroll" mode.
- The "Runtime settings" editor is open.

Enabling the zoom function

1. Click on "Screens".
2. Deactivate "Declutter" and "Extended zoom".
   The functions are disabled if you activate "Declutter" and "Extended zoom".

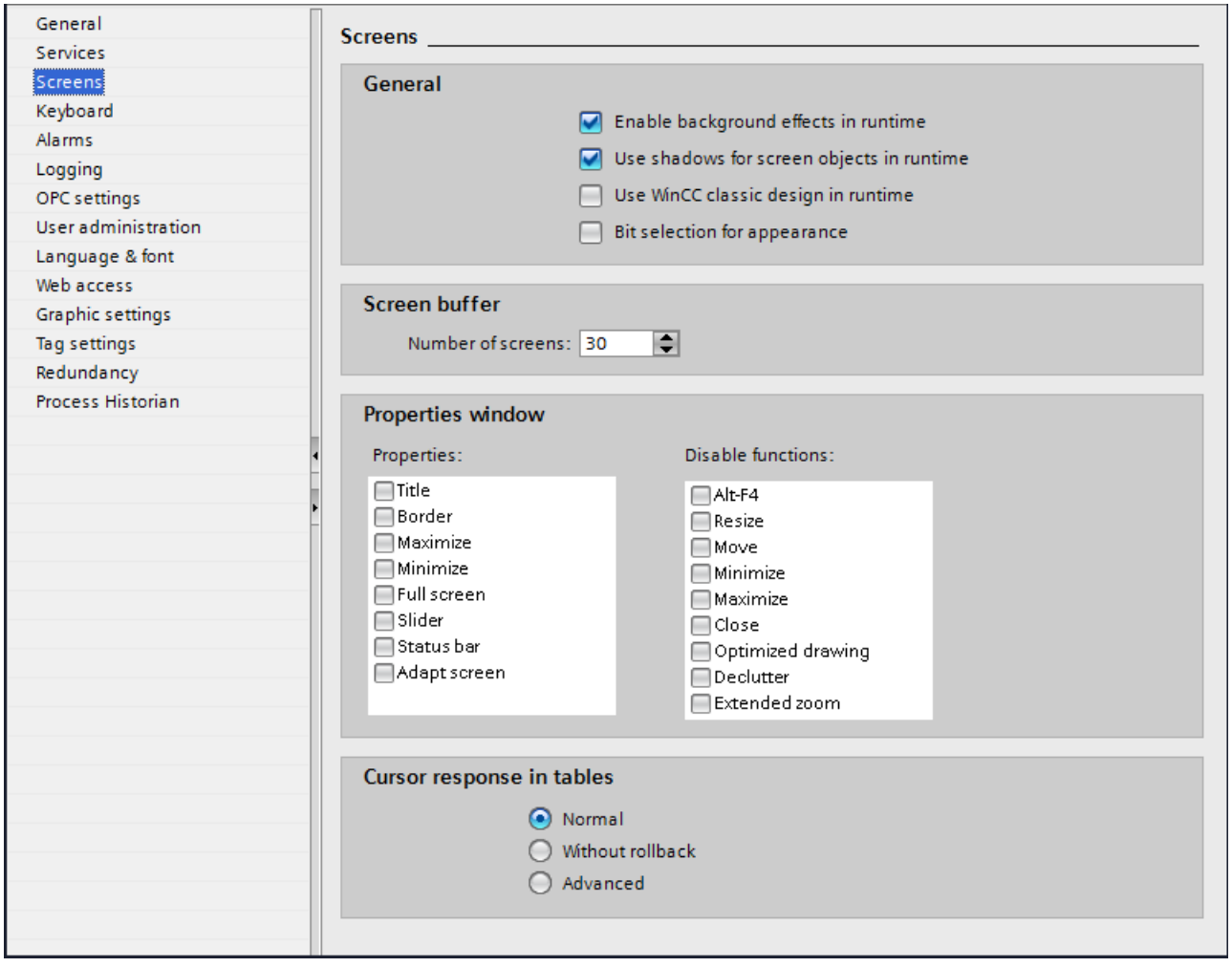

Configuring Extended Zooming Screen-Specific

The "Extended Zoom" function can be enabled/disabled for specific screens. Make the corresponding setting in the inspector window of a screen under "Properties > Properties > Miscellaneous".

See also

Dynamizing the visibility of an object (Page 219)

15.4.1.4 Setting up the start sequence

Introduction

When you activate a project, additional program modules will be loaded to execute Runtime.

Select "Start sequence of WinCC Runtime" to specify the applications that will be started on activation of a project. "Graphics in Runtime" is always started and is enabled by default.

Start only the applications required in Runtime to obtain a high performance level.

Procedure

Set up the startup list as follows:

1. Open the "Runtime settings > Services" editor.

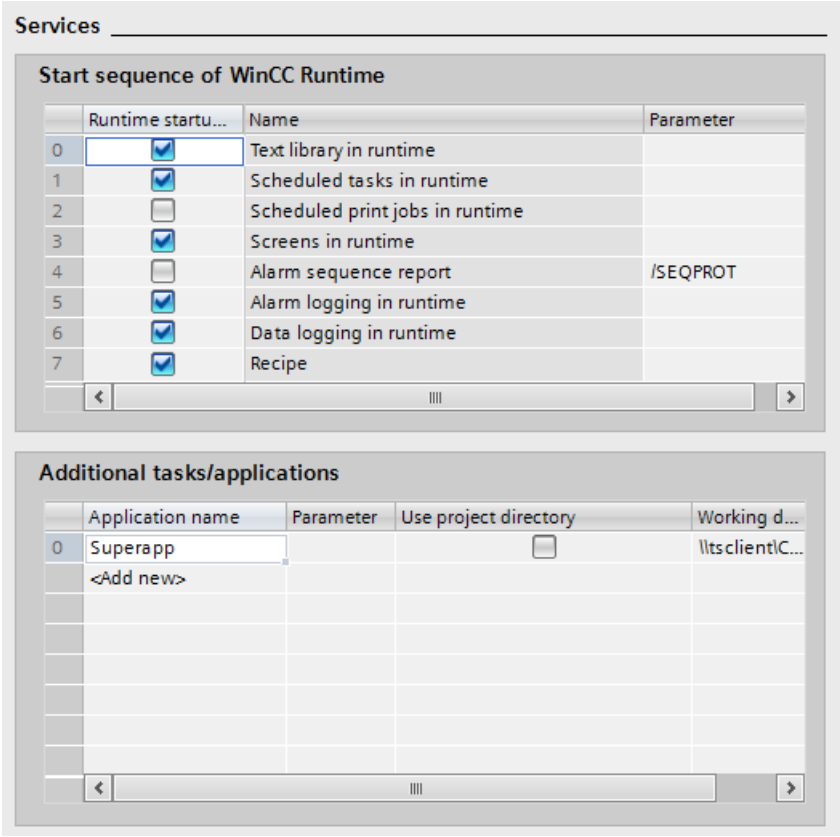

2. Enable the display of the online help in Runtime by activating "Enable help in Runtime".
3. Select the applications to load at the start of Runtime in the "Start sequence of WinCC Runtime" area.
4. You can load additional tasks or applications at the start of Runtime by clicking in the table row at "Additional tasks/application".
5. Enter a name for the additional task or application. Specify the working directory.

6. The project folder is used as working directory if you select the check box in the "Project directory" column.
7. If you clear the check box in the "Project directory" column, use the "Find" button to specify the folder in the "Working directory" column.
8. Specify any necessary parameters.
9. Specify the startup size of the task or application window in the "Window style" column, e.g. minimized, maximized or standard size.

15.4.1.5 Activating time synchronization (RT Professional)

Introduction

To have the same time throughout the plant, you can synchronize the time on various plant components using time synchronization. WinCC time synchronization is operated as a master-slave system.

One system component must be a clock for all components of a plant to work with identical time. The component functioning as the clock is referred to as the time master. The components that receive the time are time slaves.

The time master can be a computer or an external clock, such as a SICLOCK. This master transmits the time signal to the other components. Only one time master can be the active clock in the network.

To prevent the failure of time synchronization, two components can be configured as the time master. The redundant time master is a time slave as long as the active time master is transmitting the time frames.

The time slaves receive the time signal transmitted by the time master via the bus and use it to synchronize their own time.

Time synchronization in runtime

The time synchronization of the computer is entered in the start sequence of the computer for the following events:

- After configuration of the time synchronization

Note

Time synchronization takes effect a few minutes after activation in runtime. If you require precise time synchronization during startup, synchronize the local PC clocks before activating runtime. This is the case, for example, if you wish to receive process control messages during startup in chronological order.

Activating time synchronization

Proceed as follows to activate time synchronization:

1. Select the runtime "Properties" command from the shortcut menu in the project navigation and then select "Time synchronization > General".
2. Under "Synchronization", activate "Activate time synchronization".

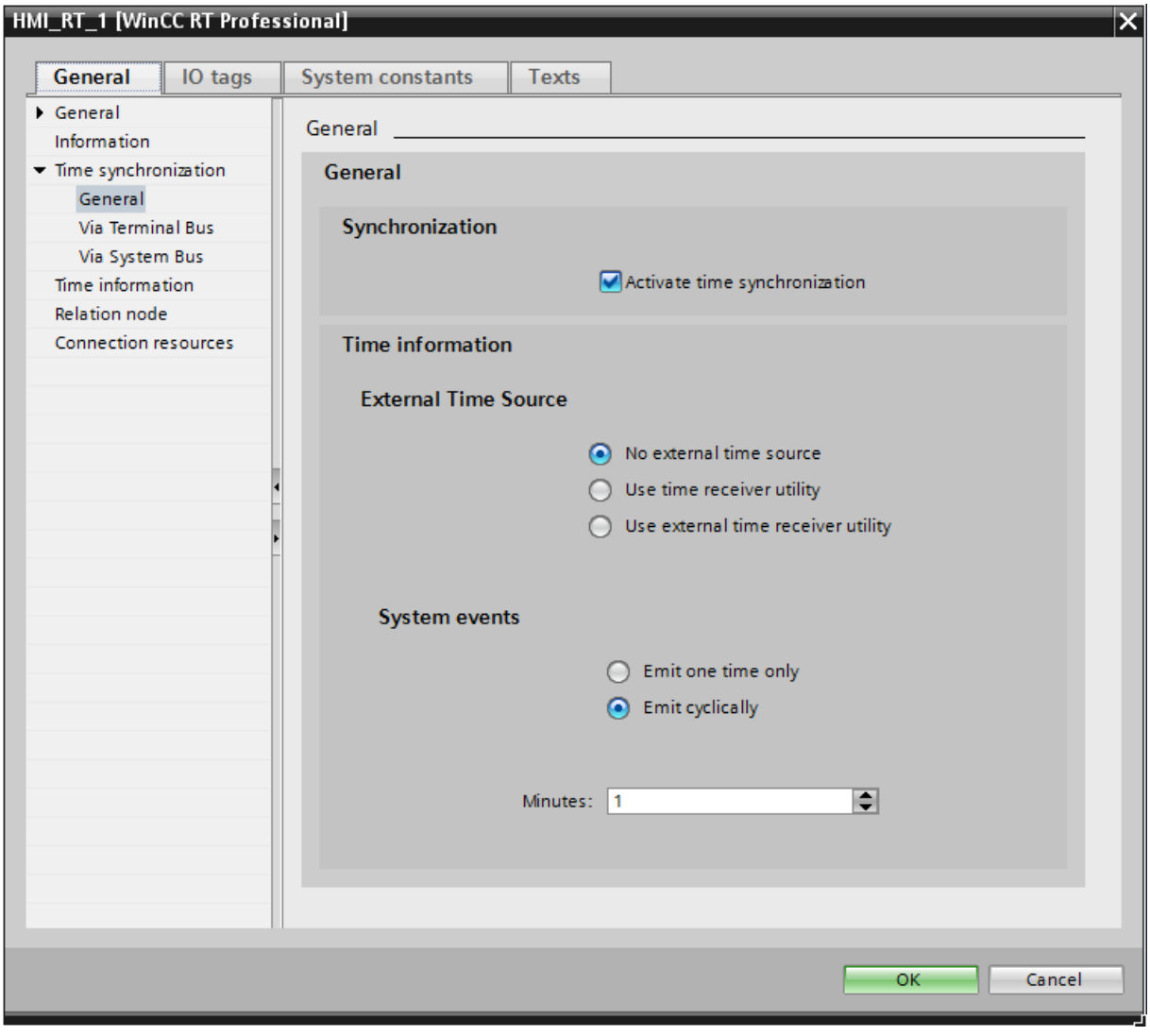

3. Under "Time information", select a time source, for example, "No external sources".
4. Specify how often the time message frame should be transmitted by the time master.

Synchronization via the terminal bus

Proceed as follows to synchronize the time via the terminal bus:

1. Select the runtime "Properties" command from the shortcut menu in the project navigation and then select "Time synchronization > Via terminal bus".

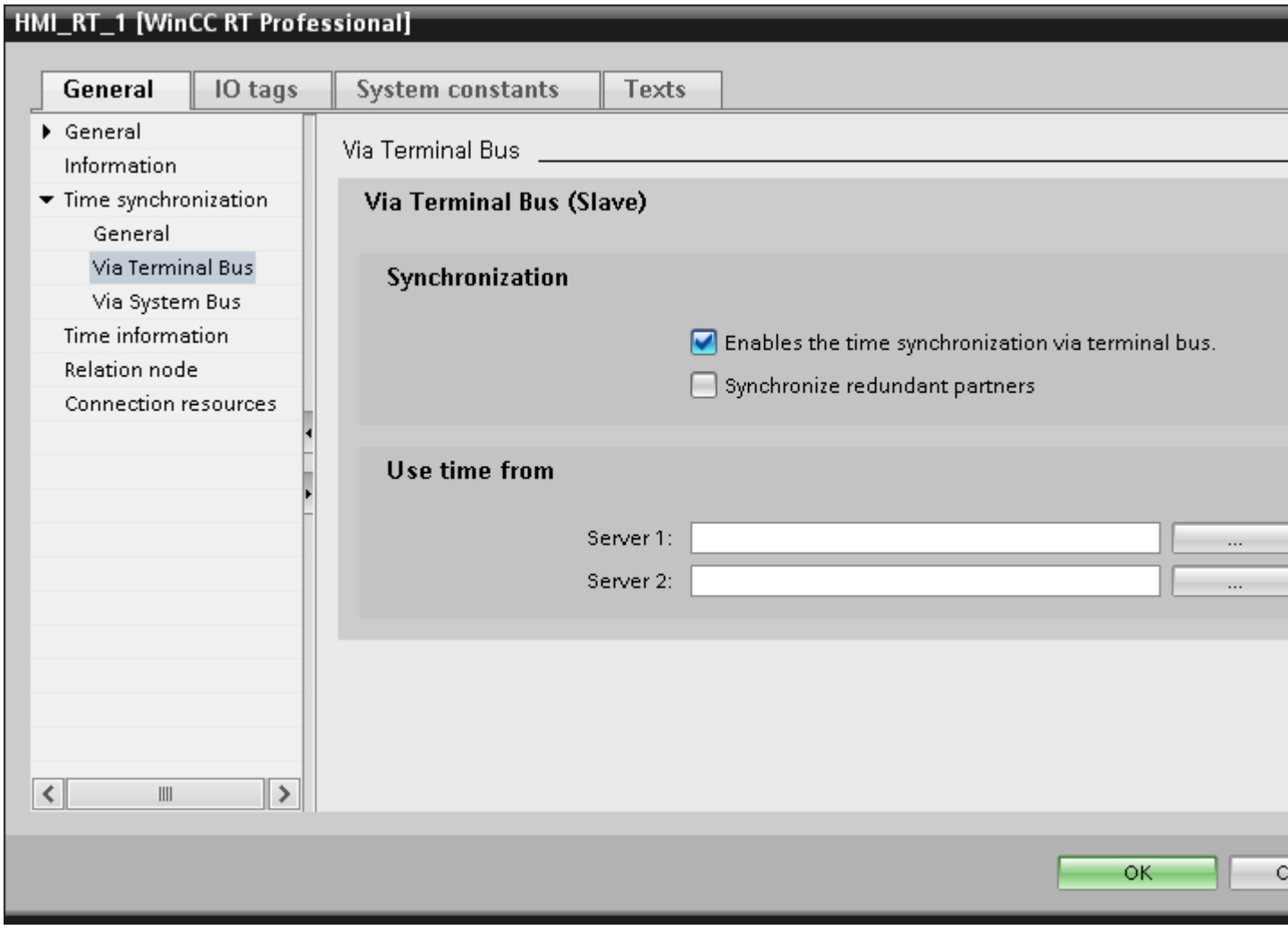

2. Under "Synchronization", activate the "Activate" check box.
3. If you also want to synchronize redundant components, activate the "Synchronize redundant partners" check box.
4. Under "Server 1", select the server from which the HMI device should receive the time message frame as the time slave.
5. Click "OK" or configure the synchronization via the system bus.

Synchronization via the system bus

Proceed as follows to synchronize the time via the system bus:

1. Select the runtime "Properties" command from the shortcut menu in the project navigation and then select "Time synchronization > Via system bus".

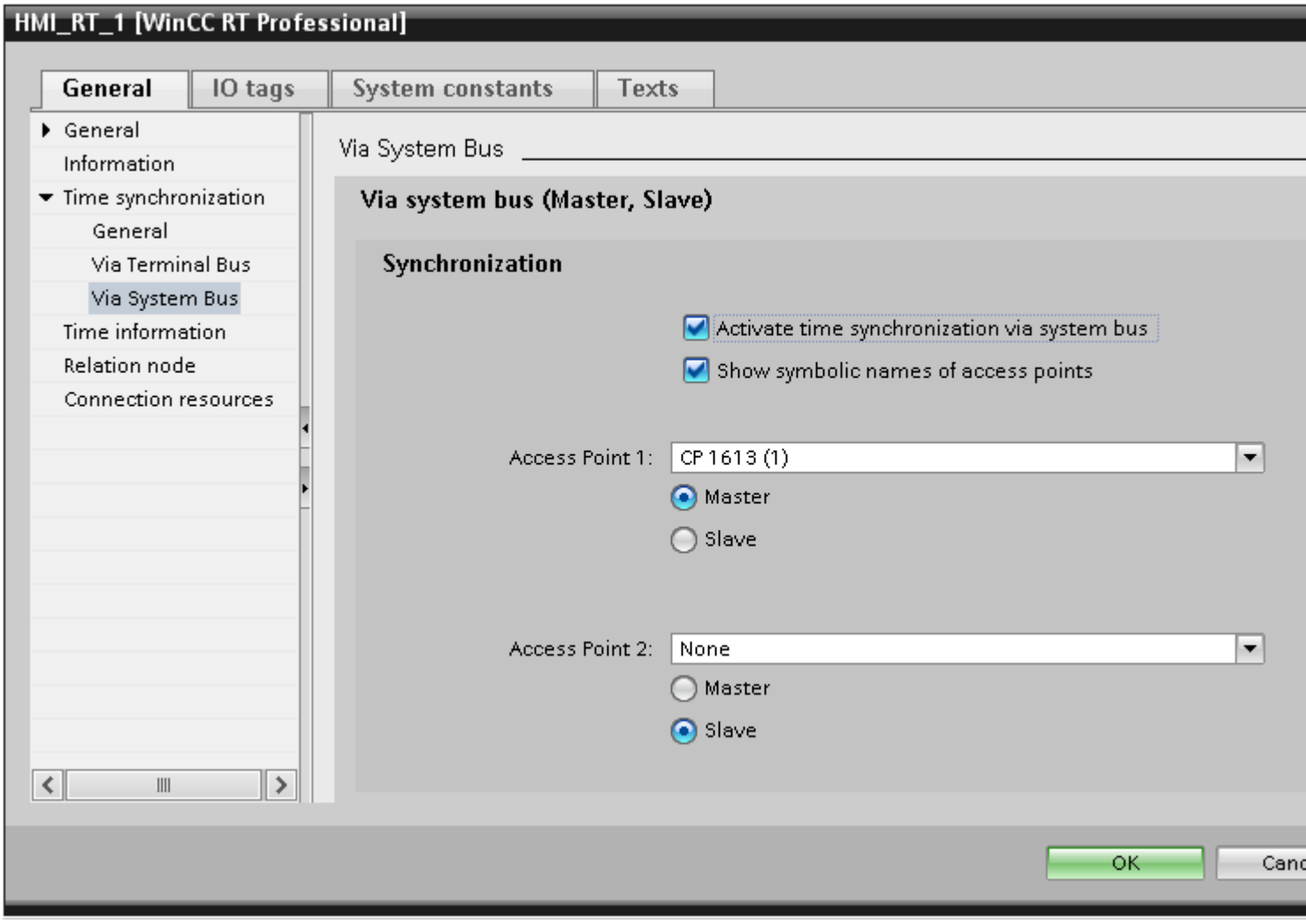

2. Under "Synchronization", activate the "Activate" check box.
3. Activate the "Show symbolic names of access points" check box.

Note

If you configure time synchronization from another computer, the installed access points are unknown. The symbolic names are shown, however, and assigned to the physical names of the access points when runtime starts.

4. Under "Access point 1", select an access point and assign it a role as master or slave.

5. You also select a second access point under the following conditions:
   - If you want to use a redundant device.
   - If you want to synchronize two system buses via a "bridge".
   - If you want to synchronize two system buses as the master.
6. Also assign this access point a role as master or slave. The combination of these roles defines the response as shown in the table below.

Access point 1	Access point 2	Behavior
Slave	Slave	The device always functions as a time slave.
Master	Master	The device functions as a cooperative master at both access points.
Master	Slave	The devices functions at access point 1 as the master and transmits time message frames; at access point 2 the device synchronizes with the time signals received there.
Slave	Master	The device functions at access point 2 as the master and transmits time message frames; at access point 1, the device synchronizes with the time signals received there.

15.4.1.6 Setting the time base

Setting the time base for the time of day

To set the time display on the HMI device, proceed as follows:

1. Select "Properties > Time information" from the Runtime shortcut menu in the project navigation.

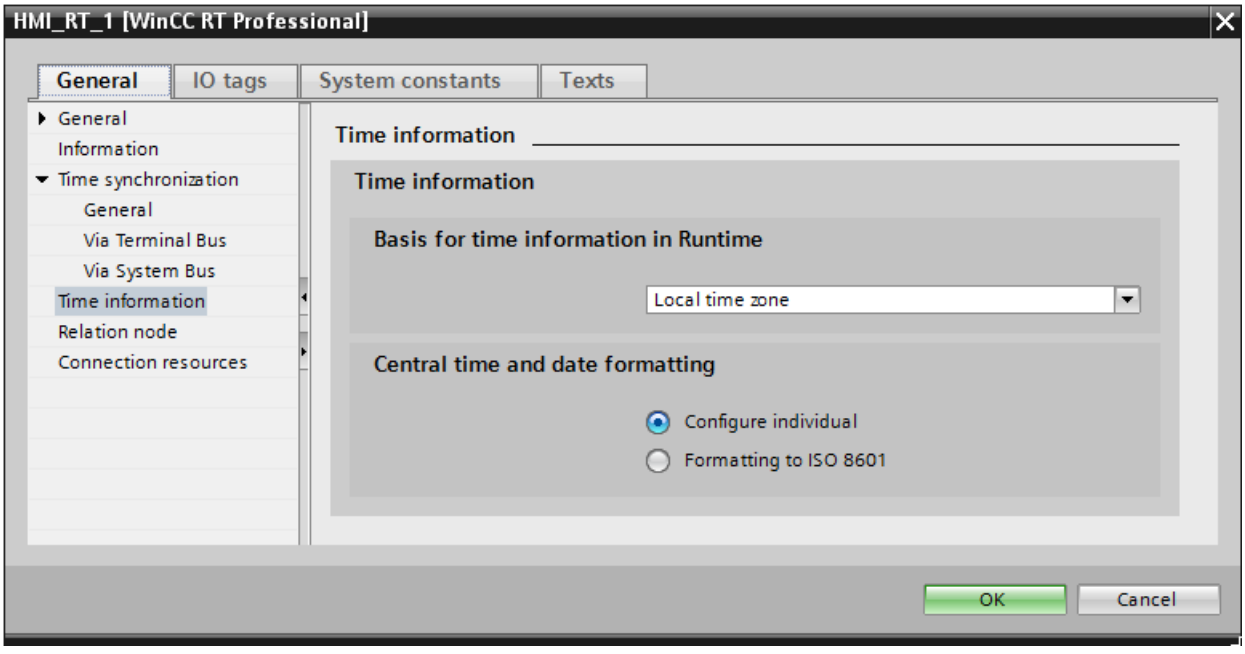

2. Select one of the following options under "Basis for time information in Runtime":
   - Time zone of server (migrated projects)
   - Local time zone
   - Coordinated Universal Time
3. Select one of the following options for the display format of the date and time:
   - "Configure individual": If you want to display the date in the configured format.
   - "Formatting to ISO 8610": If you want to display the date in ISO 8610 format on all components.

15.4.1.7 Configuring persistence

Introduction

You use the persistence to determine how long configuration changes that were configured directly on the HMI device in Runtime are saved.

The changes configured in Runtime are saved separately from the screen in the engineering system. The screen is thus retained in its original configuration.

Persistence defaults

Persistence can be configured in the engineering system, and depending on the configuration, in Runtime. There are three defaults for persistence:

- "No persistence": The online configurations are lost the next time the screen changes, and when the configuration is disabled/enabled.
- "Persistence": In Runtime, this default allows the user the options of "discard", "retain" or "reset". With the "retain" option the online configurations are retained the next time the screen changes, and when the project is disabled/enabled.
- "Persistence in Runtime": In Runtime, this default allows the user the options of "discard", "retain" or "reset". With the "retain" option the online configurations are retained the next time the screen changes, but are lost when the project is disabled/enabled.

Note

Changing a control

If you change a configured control and load it onto the HMI device, the changes are overwritten in Runtime regardless of the settings.

Online delta loading

In WinCC you can edit a project on the configuration PC while the same project is active in Runtime. If you change the project on the configuration PC, for most configuration changes you can load the changed components onto the HMI device by means of "Online delta loading". Online delta loading requires less time compared to the loading the entire project.

During online delta loading, The "Load preview" dialog is opened. In this dialog you specify whether user administration or logs, which have been changed on the HMI device in the meantime, are overwritten or retained during online delta loading.

Note

Changes on the HMI device

Interim changes in user administration or in the logs are lost during online delta loading.

15.4.1.8 Working with service mode

Basic information on service mode and autostart

Overview

Service mode provides you with the option of running WinCC Runtime Professional as service. As service, WinCC Runtime Professional can also be active when no interactive user is logged onto the computer.

In conjunction with the "Autostart" option, the service mode can be helpful in a power failure, for example, because the system is automatically powered up and the server is again available for the clients.

WinCC Runtime Professional can also be run on the computer if no interactive user is logged onto the computer.

In service mode, WinCC Runtime Professional can also be operated with logged on user, in which case interactive user inputs are possible.

Note

WinCC cannot run during interventions in the system

Changes to the processes and services of WinCC are not permitted in the Control Panel or in the Windows task manager. The following changes are affected:

- Changes to properties
- Manual interventions:
  - Starting
  - Close
  - Stop
  - Continue
  - Restart
- Change of priority

Dependencies exist between the individual processes and services.

Do not execute any changes.

See also

Configuring service mode (Page 2268)

Use of a server project and restrictions (Page 2267)

How a service project works (Page 2268)

Configurations for a service project

Overview

WinCC Runtime Professional can be run as service project in service mode in the following configurations on the server:

- WinCC Server with Windows Server 2008 R2 SP1 (64-bit)
  - Clients with Windows 7 SP1 (32-bit, 64-bit)
- WinCC Server with Windows Server 2012 R2 (64-bit)
  - Clients with Windows 8.1 (32-bit, 64-bit)

- WinCC WebNavigator server
  - WinCC Web clients
- DataMonitor server
  - DataMonitor clients

Use of a server project and restrictions

Use of a service project

The project is operated as service project in service mode on the server. WinCC Runtime Professional starts as service. A service project is started automatically or manually.

A service project is available for the following operations:

- Operation without logged-on user
  A service project can be run without an interactive logged onto the computer. If no interactive user is logged on, no interactive operation is possible.
- Operation with logged-on user
  In service projects, interactive operation is generally not desired. An interactive user can, for example, log on for service purposes. In this case, the user can enable the interactive operation of the service project.
- Automatic start
  With automatic start, WinCC Runtime Professional is started automatically and the set project enabled when the server is switched on. Automatic start is performed without an interactive user logged on.
- Manual start
  With manual start, the user has to log on to the server and then enable the project. When the user logs off again from the server, WinCC Runtime Professional continues to be active.
- Logging users on and off
  Interactive users can log on and log off at any time while the service project is enabled.

Restrictions

A service project is subject to the following restrictions:

- VB scripts and C scripts lead to problems when they require interactions, such as inputs, or display massage boxes.
  In service mode, there is also no common data area for C scripting. This means, for example, that no global C tags can be exchanged between user-defined C functions in the scheduled tasks and user-defined functions in the "Screens" editor.
- Additional tasks or applications cannot be loaded during starting of Runtime.
- OPC access is not enabled.
- Diagnostic information of a service project cannot be displayed on the server.

Note

Editing or migrating service projects

To edit or migrate a service project, you have to manage the service mode user appropriately on the computer. When the service mode user is not present, the logged on Window user must be managed accordingly so that the project can be edited or migrated.

See also

Basic information on service mode and autostart (Page 2265)

How a service project works

Basics

With a service project, WinCC Runtime Professional is started as service. Depending on the setting, these services are started at the following times:

- Automatically when the operating system starts.
- After a user has logged on and WinCC Runtime Professional has started.

WinCC Runtime Professional is still active even when the user logs off.

The Runtime data is still accessible.

The logged-on user can enable Runtime operation when required.

See also

Basic information on service mode and autostart (Page 2265)

Configuring service mode

Introduction

In the "Load preview" dialog, you specify whether the project will be operated as service project and whether autostart will be set up for the project.

Configuring projects as service projects

1. Activate the "Enable service mode" option in the "Load preview" dialog.
2. Enter your user name under "User".
3. Enter your password under "Password".

To be able to log on, you must be created as user in the SIMATIC HMI group on the HMI device.

Note

In the redundant system, you must log on in Runtime with the same user data.

Note

If the project has been loaded and service mode enabled, the "Enable service mode" option is already activated.

The "User" and "Password" fields are not filled in.

Setting up autostart

1. Activate the "Enable Autostart" option in the "Load preview" dialog.
   WinCC activates the required project when the computer is powered up.

Note

If the project has been loaded and autostart enabled, the "Enable autostart" option is already activated.

Special features when loading offline

If the configuration PC has no connection to the HMI device, you can load the project offline to the HMI device from a data storage medium. To do this, first copy the Runtime project to your HMI device and start the project in Runtime Professional with "WinCC RT Start". The Runtime project is recognized by WinCC.

Then you can configure autostart in combination with "WinCC RT Start" for operation. This configuration only works if the Runtime project has been started on the HMI device after it has been loaded offline.

See also

Basic information on service mode and autostart (Page 2265)

15.4.1.9 Password protection in Runtime

Password protection in Runtime: Basics (RT Professional)

Overview

The connection establishment can be protected with passwords in the TIA Portal. The configured passwords are saved in encrypted form. During transfer to the target system, the passwords are also protected against unauthorized access and cannot be read. These passwords are decrypted securely on the HMI devices in Runtime.

Password protection can be created for the following connections:

- SIMATIC S7-1200
- SIMATIC S7-1500
- SIMATIC S7-1500 Software Controller (WinAC)
- PC Station 2.0 (Station Manager)
- OPC UA connection (V15.0 and higher)

Note

The passwords become invalid at a change to an older version.

Note

If the user name on a HMI device is changed locally in V14 during configuration, the password becomes invalid and has to be entered again.

See also

Configuring a password for transport (Page 2270)

Resetting the password for transport (Page 2271)

Configuring a password for transport (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

To protect the private password key during transport, you can configure a password for transport under "Runtime settings".

Creating a password

1. Open the "Runtime settings > General" editor in the Project window.
2. Enter the password in the "Enter password" box in the "Transport password" area.
3. Confirm the password in the "Confirm password" box.

Note

If there was no previous password, the "Old password" box is disabled.

Result

You have now configured the password for transport. If a password for transport has been configured and you transfer the project to the HMI device, a dialog with a password prompt will appear once in runtime.

If the password entered is correct, the certificate is saved in the certificate store of the HMI device.

When you change the password, load the project with the changed password into the HMI device and start Runtime. No password prompt appears.

See also

Password protection in Runtime: Basics (Page 2269)

Resetting the password for transport (Page 2271)

Resetting the password for transport (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

If you have moved the transport password, for example, you can create a new security certificate. To do so, reset the transport password.

Resetting the password

1. Click the "Reset password" button.
   The password is reset. A new certificate without a password is generated.

   Note

   Other passwords, for example the password for the PLC, will become invalid. The next time you compile, an error message will appear telling you that these passwords are no longer valid.

2. Enter the new password and confirm it in the "Confirm password" field.

   Note

   The certificate is not password-protected until you enter a new password for transport.

   After download of the project to the HMI device, you will be prompted for the password.

See also

Password protection in Runtime: Basics (Page 2269)

Configuring a password for transport (Page 2270)

15.4.2 Overview of compiling and loading projects (RT Professional)

Overview

The project is compiled in the background even as you are configuring it in WinCC. This reduces the time for final compilation. When you start compilation, you create a file that can be run on the corresponding HMI device.

If an error occurs during compilation, WinCC provides support in locating and correcting it.

Once you have corrected any problems, you download the compiled project to the HMI devices on which the project is to run. If the configuration PC is not connected to the HMI device, save the compiled project on a data medium of your choice. The compiled project is then transferred from a PC connected to the HMI device to the HMI device.

If you are using HMI tags in your project that are connected to PLC tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Note

WinCC version compatibility

To compile a project of a predecessor WinCC version, upgrade the project to your WinCC version. You can then no longer open the project in the predecessor WinCC version.

Project

The term "project" has two different meanings in the contexts of compilation and loading. "Project" is the WinCC project on the configuration PC. "Project" is also the Runtime project you create by compiling the configuration data of an HMI device and download to the HMI device.

- WinCC project: contains the configuration data of one or more HMI devices
- Runtime project: contains the compiled configuration data of an HMI device

The figure below illustrates the link between WinCC projects and Runtime projects using the example of the "Compile and load" process:

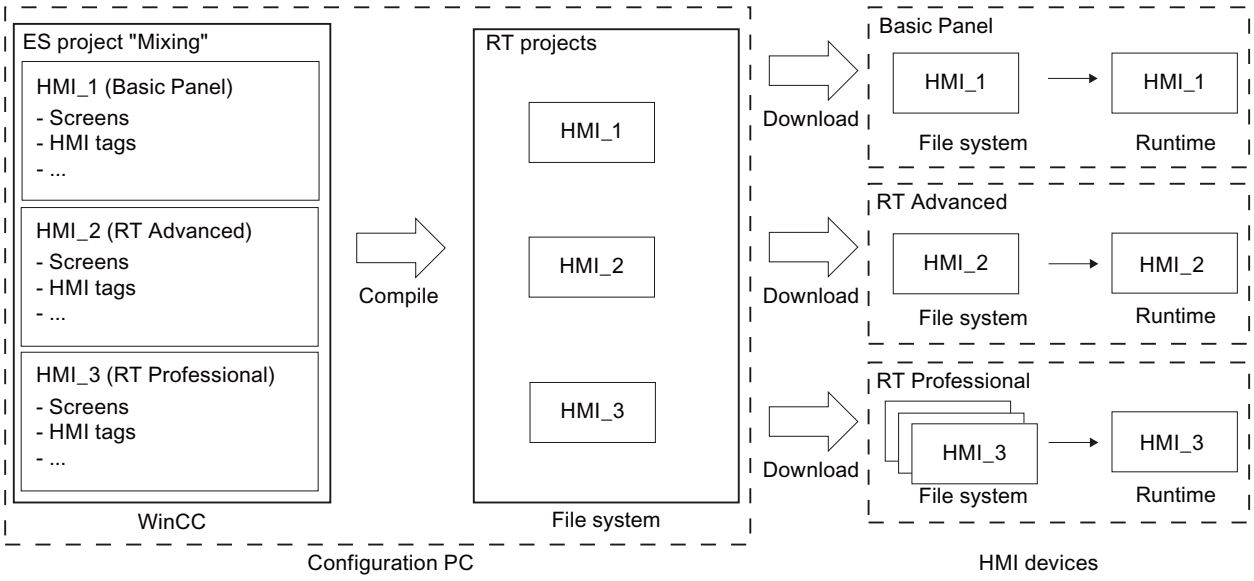

Runtime

Runtime is the software for process visualization. In Runtime, you execute the project in process mode.

A distinction is made between two types of Runtime:

1. Runtime on a panel
   Before running a Runtime project on a panel, you have to transfer the Runtime project to the panel before startup.
2. Runtime on a PC
   You can execute the Runtime project directly on the configuration PC if Runtime has been installed on the configuration PC.
   If you want to execute the Runtime project on a different PC, you have to transfer the Runtime project to the PC before startup.

Simulation

You test your configuration with a simulation. You can start a simulation without a link to the active process.

In a simulation, you test configured tags or screen changes, for example. During the simulation, the configured tags can be manipulated, activated and deactivated with the help of the tag simulator.

There are two types of simulation:

1. Simulating a panel
   If you created a panel in your project, the panel is displayed in the simulation. With the help of this type of simulation, you can test your configuration on the HMI device without transferring the project to the panel.
2. Simulating Runtime
   Simulating Runtime allows you to test the project directly on the configuration PC.

See also

Generating a "Pack&Go" file (Page 2188)

Starting Runtime Advanced and Panel Runtime (Page 2200)

Load files using the "HMI files" editor (Page 2107)

15.4.3 Compiling a project (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Introduction

The changes made to the project are compiled in the background even as you are configuring a project in WinCC. Projects are compiled automatically when you load them. This ensures that the latest version of the project is loaded at all times.

WinCC checks consistency of the project during compilation. The error locations in the project are listed in the Inspector window. You can jump directly to the source of the error from the entry in the Inspector window. Check and correct errors found.

Scope of the compilation

Configuration data is compiled in the background as soon as you start configuring an HMI device. If you compile a project manually, only the changes in the configuration made since the last compilation process are compiled in the background.

You can start complete project compilation manually at any time; this may, for example, be done to test the consistency of the configured data.

Requirement

- A project is open.

Procedure

Proceed as follows to compile a project:

1. If you want to compile several HMI devices at the same time, select all the relevant HMI devices with multiple selection in the project tree.
2. Compile the project:
   - To only compile changes in the project, select the "Compile > Software (only changes)" command from the shortcut menu of the HMI device.
   - To compile all project data, select the "Compile > Software (compile all)" command from the shortcut menu.

Result

The configuration data of all selected HMI devices is compiled. Any errors that occur during compilation are shown in the Inspector window.

15.4.4 Simulating projects

15.4.4.1 Simulation basics (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

You can use the simulator to test the performance of your configuration on the configuration PC. This allows you to quickly locate any logical configuration errors before productive operation.

You can start the simulator as follows:

- In the shortcut menu of the HMI device or in a screen: "Start simulation"
- Menu command "Online > Simulation > [Start|With tag simulator|With script debugger]"
- Under "Visualization > Simulate device" in the portal view.

Requirement

The simulation/runtime component is installed on the configuration PC.

Field of application

You can use the simulator to test the following functions of the HMI system, for example:

- Checking limit levels and alarm outputs
- Consistency of interrupts
- Configured interrupt simulation
- Configured warnings
- Configured error messages
- Check of status displays

15.4.4.2 Runtime Professional simulation (RT Professional)

Introduction

Simulating Runtime Professional allows you to test the project with process link directly on the configuration PC. You can also test the tags using the tag simulator.

Function

Both process values and alarms are saved to a simulation log. The data generated is deleted from this log when you exit simulation mode.

The Runtime database is restored to the original state it had prior to simulation. Any changes to project data made during Runtime will be lost. You use the tag simulator to simulate the assignment of various values to the tags in your project.

Project logging and backup operations are continued during simulation.

WARNING

Possible damage to the plant

If you have changed tag values during the simulation with a PLC connection, unforeseen plant states can result in irreparable damage to the plant. Therefore, limit manual value manipulation to a minimum and select the test values carefully.

Limitations

Client-server operation is not possible during simulation.

15.4.4.3 Simulating a project (RT Professional)

Requirement

The project is open in the configuration PC.

Note

The tag values of faulty connections to the process cannot be simulated.

Procedure

Proceed as follows to simulate a project:

1. Disconnect the connection to the controller or switch the controller off if you do not want to test the function of the entire process.
2. Select the menu command "Online > Simulate Runtime > With tag simulator".
   Runtime starts and the simulator opens with all the configured tags.
3. Select the "New tag" command in the "Edit" menu to add a tag to the tag simulator.
4. Select the tag from the WinCC tag management.
5. In the tag simulator, select the required tag on the "Tags" tab to supply a tag with simulated values.

6. Configure the simulation mode on the "Properties" tab:
   - Select the simulation mode.
   - Enter values for the simulation.
   - Click "Active" to include the tag in the simulation.

   WARNING

   Make sure once again that the configuration computer cannot pass any changed tags to the PLC in the active process. If you have not switched the PLC off, the simulated tag values will be passed to the PLC and the runtime process.

7. Click the "Start simulation" button on the "Properties" tab.
8. Observe the effect the simulation has on your project in Runtime.
9. Click on the "Stop simulation" button to stop the simulation.
10. To save the settings for the simulation, select the menu command "File > Save" and enter a corresponding name.
    The file is automatically given the *.sim extension.

Managing simulation data

If you have saved data from a previous simulation, you can open the file at a later point in time and simulate your project again. The tags and area pointers to be simulated must remain unchanged in the project, otherwise the saved simulation data will no longer match your project.

Proceed as follows to open a simulation file:

1. Select the menu command "Online > Simulate Runtime > With tag simulator".
2. Select the menu command "File > Open" in the tag simulator.
3. Select the corresponding simulation file and click "Open".
   The simulator loads the stored data.

Enabling and disabling tags

You can start and stop the simulation for each individual tag to enable a bumpless transition from the offline configuration to online configuration. Enable the "Active" check box in the "Properties" tab for this purpose.

When a tag is enabled, the simulation values are calculated and transferred to the tag simulator.

Deleting a tag

To delete a tag from the tag simulator, follow these steps:

1. Select the tag name.
2. Select the menu command "Edit > Delete".

The tag is removed from the table.

15.4.4.4 Simulating a screen (RT Professional, RT Advanced, Panels, Comfort Panels)

Introduction

If you have only made changes to one screen, you can temporarily specify this screen as the start screen for simulation. In this way, you can debug changes without having to modify the start screen, or opening the screen on the HMI device.

Requirements

You created a project that contains at least one screen.

Procedure

To define a screen as temporary start screen for simulation, follow these steps:

1. In the project navigation, select the image to display as the start screen in the simulation.
2. Select the "Start simulation" command from the shortcut menu of the screen.

Result

Project simulation is started. Instead of the configured start screen, the simulation window the screen you selected in the project navigation.

15.4.4.5 Working with the tag simulator (RT Professional)

Simulation modes

The simulator has six different simulation modes. The configured tags are supplied with nearly realistic values during the simulation depending on the simulation mode.

Simulation mode	Description
Sinusoidal	Changes the value of the tag to form a sinusoidal curve. The value is visualized as a periodic, non-linear function. Settings: • Amplitude • Offset (zero point for value range) • Period of oscillation
Oscillation	For simulating jumps in a setpoint. Settings: • Overshoot • Setpoint • Period of oscillation • Damping

Simulation mode	Description
Random	Provides randomly generated values. The tag value is changed by means of a random function. Settings: • Minimum value • Maximum value
Increment	Increases the value of the tag continuously up to a specified maximum value. Begins again at the minimum after the maximum has been reached. The value trend corresponds to a positive saw-tooth curve. Settings: • Start value • End value
Decrement	Decreases the value of the tag continuously down to a specified minimum value. Begins again at the maximum after the minimum has been reached. The value curve corresponds to a negative saw-tooth curve. Settings: • Start value • End value
Slider	Sets a fixed value. Settings: • Minimum • Maximum

15.4.4.6 Simulation restrictions

Alarms with dynamic parameters

If you use tags or text lists as external tags for alarms, the dynamic parameters of the alarms are not displayed.

Only internal tags are enabled for the simulation of alarms in the tag simulator .

Use PLCSim to simulate dynamic parameters.

Comfort Panels

If you want to use PLCSim with version V14 or higher for simulation with a Comfort Panel, the device version of the Comfort Panel must be 14.0.0.0 or higher.

15.4.5 Loading projects

15.4.5.1 Overview for loading of projects (Panels, Comfort Panels)

Overview

Delta data of the project is automatically compiled before you download it to one or several HMI devices. This always ensures that the latest version of the project is transferred.

Loading a project to an HMI device

The following steps are completed prior to downloading:

1. The download settings are verified. The "Extended download to device" dialog box opens automatically during the initial download of a project to an HMI device. You use this dialog to define the protocol and interface or destination path for the project in accordance with the HMI device Runtime used.
   If the HMI device is part of a subnet, for example, you also select the subnet and the first gateway.
   You can open the "Extended download" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens.
2. The project is compiled. Warnings and errors during compilation are displayed in the Inspector window and in the "Load preview" dialog.
3. The "Load preview" dialog shows you the following information for each HMI device:
   – The individual steps for loading
   – If the device version of the target HMI device does not match the configured device version, you are queried if you want to change the device version at this time.
     As long as the project with the selected device version of the target HMI device is not executable, you cannot start the download.

NOTICE
Changing the device version deletes all data on the HMI device. Data is deleted on the target system if you change the device version. For this reason, you should first back up the following data: • User administration • Recipes • License keys With Comfort devices, the licenses can only be deleted when resetting to factory settings. Back up the license keys before you restore the factory settings.

   – Presettings that take effect at loading. You can change the default settings for this download process, if necessary.
   – Warning events (optional). You can download a project while ignoring the "warnings". The functionality may be restricted in runtime.
   – Error events (optional). You cannot load the project. Eliminate the errors and then reload the project.
     WinCC will open the invalid configuration in the corresponding editor if you double-click the error message in the Inspector window. Correct the errors and reload the project.

If you are using HMI tags in your project that are connected to PLC tags, you should also compile the user program before you compile the HMI device.

Note

Interruption of a download

If the download is interrupted, WinCC automatically ensures that no data is lost and that existing data is deleted on the HMI device only after complete transmission.

Loading a project without a connected HMI device

If you cannot establish a direct connection from the configuration PC to the HMI device, copy the compiled project to an external storage medium, e.g. a USB stick, and load it from the USB flash drive to the HMI device.

This function is available in connection with an HMI device image which is compatible with the TIA Portal version V14 or higher.

You create the required project data in WinCC by configuring the HMI device and then dragging and dropping the folder of the HMI device (e.g. "HMI_1 [<*device type*>]") per to an external storage medium under "Card Reader/USB memory".

Loading of projects without recipe data records

You are using recipes in a project. You transfer the project to a Basic Panel without recipe data records.

You may encounter inconsistencies if you have altered the structure of the recipe in the engineering system and the device already held recipe data records.

Check the consistency of the data records in this case. The device will not issue a note for all structural changes.

Pack&Go

If you cannot establish a direct connection from the configuration PC to the HMI device, copy the compiled project to a PC that is connected to the HMI device, for example via a network. Download the project from this PC to the HMI device.

Loading with S7 routing

Configure the S7 routing settings in the "Devices & Networks" editor in the relevant PLC. The settings depend on the device configured.

S7 routing supports the following protocols:

- MPI/PROFIBUS
- PN/IE

Transferring Runtime add-ons in WinCC

Projects can contain Runtime add-ons in the form of controls or CSP (Communication Support Packages). These Runtime add-ons are automatically transferred with the project.

15.4.5.2 Overview for loading of projects (RT Advanced)

Overview

Delta data of the project is automatically compiled before you download it to one or several HMI devices. This always ensures that the latest version of the project is transferred.

Connection establishment

Ensure that the device version configured in the engineering system matches the installed Runtime version, otherwise, communication will not be established or will be delayed when the project is loaded.

If the device version does not match the installed Runtime version, you have the following options:

1. Change the device version configured in the engineering system.
2. Install the version of WinCC Runtime Advanced corresponding to the configured device version.

Loading a project to an HMI device

The following steps are completed prior to downloading:

1. The download settings are verified. The "Extended download to device" dialog box opens automatically during the initial download of a project to an HMI device. You use this dialog to define the protocol and interface or destination path for the project in accordance with the HMI device Runtime used.
   If the HMI device is part of a subnet, for example, you also select the subnet and the first gateway.
   You can open the "Extended download" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens.
2. The project is compiled. Warnings and errors during compilation are displayed in the Inspector window and in the "Load preview" dialog.
3. The "Load preview" dialog shows you the following information for each HMI device:
   - The individual steps for loading
   - Check the runtime version of the target HMI device
     If the version of WinCC Runtime Advanced installed on the target device does not match the configured device version, you cannot load the project.
   - Default settings that take effect when loading
     You can change the default settings for this download process, if necessary.
   - Occurring warnings (optional).
     You can download a project while ignoring the "warnings". The functionality may be restricted in runtime.
   - Occurring errors (optional).
     You cannot load the project. WinCC will open the invalid configuration in the corresponding editor if you double-click the error message in the Inspector window. Correct the errors and reload the project.

If you are using HMI tags in your project that are connected to PLC tags, you should also compile the user program before you compile the HMI device.

Note

Interruption of a download

If the download is interrupted, WinCC automatically ensures that no data is lost and that existing data is deleted on the HMI device only after complete transmission.

Loading a project without a connected HMI device

If you cannot establish a direct connection from the configuration PC to the HMI device, copy the compiled project to the HMI device.

Loading with S7 routing

Configure the S7 routing settings in the "Devices & Networks" editor in the relevant PLC. The settings depend on the device configured.

S7 routing supports the following protocols:

- MPI/PROFIBUS
- PN/IE

Transferring Runtime add-ons in WinCC

Projects can contain Runtime add-ons in the form of controls or CSP (Communication Support Packages). These Runtime add-ons are automatically transferred with the project.

15.4.5.3 Overview for loading of projects (RT Professional)

Overview

Delta data of the project is automatically compiled before you download it to one or several HMI devices. This always ensures that the latest version of the project is transferred.

When you execute the "Load (complete)" command, you need to confirm overwriting of the project.

Loading a project to an HMI device

The following steps are completed prior to downloading:

1. The download settings are verified. The "Extended download to device" dialog box opens automatically during the initial download of a project to an HMI device. You use this dialog to define the protocol and interface or destination path for the project in accordance with the HMI device Runtime used.
   If the HMI device is part of a subnet, for example, you also select the subnet and the first gateway.
   You can open the "Extended download to device" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens.
2. The project is compiled. Warnings and errors during compilation are displayed in the Inspector window and in the "Load preview" dialog.
3. The "Load preview" dialog shows you the following information for each HMI device:
   - The individual steps for loading
   - Check the runtime version of the target HMI device
     If the version of WinCC Runtime Professional installed on the target device does not match the configured device version, you cannot start the loading.
   - Presettings that take effect at loading. You can change the default settings for this download process, if necessary.
   - Warning events (optional). You can download a project while ignoring the "warnings". The functionality may be restricted in runtime.
   - Error events (optional). You can download a project even if it has errors. In the "Load preview" dialog under "Actions", select the check box "The project has not been compiled without errors. Do you want to continue loading?" and click "Load".
     WinCC will open the invalid configuration in the corresponding editor if you double-click the error message in the Inspector window. Correct the errors and reload the project.

If you are using HMI tags in your project that are connected to PLC tags, you should also compile the user program before you compile the HMI device.

Loading a project without a connected HMI device

If you cannot establish a direct connection from the configuration PC to the HMI device, load the compiled project on a USB stick and then copy it to the appropriate directory on the HMI device using the Windows Explorer.

Loading with S7 routing

Configure the S7 routing settings in the "Devices & Networks" editor in the relevant PLC. The settings depend on the device configured.

S7 routing supports the following protocols:

- MPI/PROFIBUS
- PN/IE

Transferring Runtime add-ons in WinCC

Projects can contain Runtime add-ons in the form of controls or CSP (Communication Support Packages). These Runtime add-ons are automatically transferred with the project.

Loading inconsistent projects

Loading inconsistent projects to a device can result in the loss of the online delta load capability of the device. Therefore, the system prevents the loading of inconsistent projects in Runtime Professional. You can only simulate a Runtime project or load it onto a device if no errors were output during compiling. The options "Complete load", "Online delta loading" and the loading via data carrier are not available with inconsistent projects. If the system determines inconsistencies, an error message in the "Load preview" dialog informs you that the project cannot be loaded on the device.

Inconsistencies in VB scripting and C scripting are an exception. You can load or simulate a project with errors in VB scripting and C scripting on a device if you have selected the check box "Start even with corrupt scripts" under "Runtime settings > General > Script options".

15.4.5.4 Downloading the project to an HMI device (RT Professional)

Introduction

Before a project can run on an HMI device, you must first load it to the HMI device. During loading, you must most importantly specify whether existing data on the HMI device such as "user administration" and "recipe data" is to be overwritten. As a general rule, only one project can be active in Runtime on an HMI device.

You can use a project from engineering system to an HMI device using the command "Download (complete)" or with the command "Delta load".

If the HMI device supports PROFINET, the name of the HMI device entered in the project tree is used as the device name for the PROFINET communication. The name is written to the HMI device during download. If a device name for the PROFINET communication has already been entered in the HMI device, it will be overwritten.

Please note the following when downloading a project to an HMI device:

- You can download a project from the configuration PC to the HMI device if no project is open or in Runtime on the HMI device.
- If the configuration PC also serves as HMI device, download the project to your project directory in the file system.
- If a project is open on the HMI device, you can only completely reload this particular project from the configuration PC to the HMI device. Close the project open on the HMI device to download a different project to the HMI device.
  Example: The project "Mixing" is open on the HMI device but not in Runtime. If you change the project, download the changes to the HMI device with "Software (all)". Close the project "Mixing" on the HMI device to download the project "Bottling" to the HMI device, for example.

- You can load certain changes to the configuration data without stopping Runtime (delta loading) if a project is in Runtime on the HMI device. You will be notified if a change means delta loading is no longer possible. If delta loading is no longer possible, you will only be able to download the project to the HMI device using "Download (complete)". Runtime is thereby disabled and started again, which may result in plant downtime.
  You will have to exit Runtime and close the project on the HMI device to download another project to the HMI device.
- To load a project of a predecessor WinCC version, first upgrade the project to your WinCC version.

NOTICE
Device version If the device version of the target HMI device does not correspond to that of the configured device version, Runtime cannot be started after loading. Verify that the device version of the target HMI device conforms to your configuration before you compile and download your project. If necessary, change the device version manually via the properties of the HMI device.

- If the selected destination directory on the HMI device is empty, The "Download to device (complete)" command is executed automatically once the "Extended download to device" dialog is filled out.
- If the selected directory contains a runtime project of the same name and you execute the "Download (complete)" command, you need to confirm overwriting of the project.
- If the selected directory contains a runtime project of the same name and Runtime is activated, you can alternatively execute the "Delta load" command.
- If the name of the runtime project does not match the project name on the engineering project, the project is not loaded.
- Loading inconsistent projects is prevented by the system in Runtime Professional. You can only simulate a Runtime project or load it onto a device if no errors were output during compiling. If the system determines inconsistencies, an error message in the "Load preview" dialog informs you that the project cannot be loaded on the device. Inconsistencies in VB scripting and C scripting are an exception. You can load or simulate a project with errors in VB scripting and C scripting on a device if you have selected the check box "Start even with corrupt scripts" under "Runtime settings > General > Options for scripts".

Requirements

- You have created an HMI device in the project.
- The HMI device is connected to the configuration PC or the configuration PC serves as HMI device.
- The same type of communication is set, for example, encrypted communication, on the configuration PC and the HMI device. Alternatively, use the migration mode on one of the devices (see AUTOHOTSPOT).
- The same password is assigned at both ends with encrypted communication.

Procedure

Proceed as follows to load a project:

1. To download a project simultaneously to several HMI devices, select the HMI devices by means of multiple selection in the project tree.
2. Select the "Download to device > Software" command from the shortcut menu of an HMI device.
3. If the "Extended download to device" dialog opens:
   - Select the "destination path" for the compiled project on the HMI device.
     The Windows user group "SIMATIC HMI" must have full access to this destination path. Full access is usually set during installation of WinCC Runtime for the "WinCC Projects" file. The location of this folder in the file system depends on the operating system of the HMI device.

     Note

     With multiple runtime projects, create a separate directory for each project. The directory must be empty when you load a project to an HMI device for the first time.

   - Click "Load".

   You can open the "Extended download to device" dialog at any time with the menu command "Online > Extended download to device...".
   The "Load preview" dialog opens. The project is compiled at the same time. The result is displayed in the "Load preview" dialog.
4. Check the displayed defaults and change them as necessary:
   - Select the data to be transferred.
   - Select whether Runtime is to be started on the target system after loading.
5. Click "Load".

Result

The project is loaded to all selected HMI devices. If errors or warnings occur during the download, corresponding alarms are displayed under "Info > Load" in the Inspector window.

On completion of the successful download of the project, you can execute it on the HMI device. If starting of Runtime on the target system was activated in the "Load preview" dialog, the project is started in Runtime after loading.

See also

Overview of online delta transfers (Page 2291)

15.4.5.5 Downloading the project to a file system

Introduction

If the configuration PC has no connection to the HMI device, download the project to the HMI device offline via a USB data storage medium or a CD. To do this, first copy the runtime project to your HMI device and start the Runtime Professional project with "WinCC RT Start".

The PC on which Runtime is activated is referred to as the HMI device.

The "Load preview" dialog opens. In this dialog, you can specify whether the user administration or logs, which have been changed on the HMI device in the meantime, are to be overwritten or retained.

Note

Existing data in the user management or in the logs are overwritten when you start the downloaded runtime project.

To retain changes of the user administration, export the user administration and import the data again after starting the copied runtime project. You make the settings for overwriting in the "Overwrite" area of the "Load preview" dialog.

Requirement

A project is open.

Procedure

To download a project to a file system, follow these steps:

1. Select the project in the project tree.
2. Select the "Online > Download to file system" menu command.
   A dialog opens.
3. Select the storage path and click "OK".
   The project is compiled.
4. When the "Load preview" dialog opens, select the data to be transferred.
5. Select one of the following options under "Options for data logging":
   - "Retain": If the project has already been transferred from the Engineering System to the PC, the data on the PC is not overwritten by the data of the Engineering System.
   - "Reset": The data on the PC is overwritten by the data of the Engineering System.
   - "Transfer": The data is transferred from the Engineering System to the PC.
6. Click "Load".
   The Runtime project is saved on the drive.
   If necessary, transfer the runtime project to a USB data storage medium or burn the project on a CD.

7. Transfer the runtime project to your HMI device.
8. Start Runtime and open the downloaded project.

15.4.5.6 Online delta transfer for Runtime Professional

Overview of online delta transfers

Online delta loading

WinCC lets you edit a project on the configuration PC while the same project is active in Runtime. You can in most cases load the configuration changes to the HMI devices using "Online delta loading" if you change the project on the configuration PC. Online delta loading requires less time compared to the transfer of the entire project. The "Load preview" dialog opens for online delta loading. In this dialog you specify whether user administration or logs, which have been changed on the HMI device in the meantime, are overwritten or retained during online delta loading.

Note

Changes to the project made in the meantime on the HMI device, for example in the user administration or in the logs, are lost during the online delta loading.

To retain changes to the user administration, export the user administration and import again following the online delta loading.

Note

Device version

If the device version of the target HMI device does not correspond to that of the configured device version, delta transfer is not possible.

Before the delta transfer of your project, check whether the device version of the target HMI device corresponds to your configuration and, if necessary, update the version of your HMI device.

Note the following for online delta loading:

- If online delta loading is initiated without Runtime being active, the complete project is loaded.
- Online delta loading is not possible during a simulation.
  To terminate the simulation automatically before online delta loading and to resume after loading, select the appropriate option in the "Load preview" dialog.

Note

Loading inconsistent projects to a device can result in the loss of the online delta load capability of the device. Therefore, the system prevents the loading of inconsistent projects in Runtime Professional.

Complete transfer

Fundamental changes to configuration data may prevent the online delta transfer. In this case, a message informs you of the failure of the online delta loading after you initiated the transfer. You then have to reload the entire project to the HMI device. You must explicitly confirm this Runtime restart on the HMI device in the "Load" dialog. You can cancel loading at this point.

The following will happen if you start complete loading while the project on the configuration PC is in Runtime on the HMI device:

- Runtime is stopped
- The entire project is downloaded to the HMI device
- Runtime is restarted

Note that starting and stopping of Runtime can result in plant downtime.

Application scenarios

You will have to adapt your configuration in various different phases of your project, for example, commissioning, operation and maintenance. You can then transfer these changes to the HMI device online without having to exit Runtime.

Cases in which online delta loading is used include:

- Continuous automation tasks:
  All changes should be made online from a central configuration PC. This saves you having to make changes to the configuration locally. You can also add, change and delete Runtime objects such as tags, alarms and logs without having to exit Runtime.
- Testing changes in a protected environment:
  You can perform the planned changes offline on the configuration PC before these are downloaded to the HMI device. Configuration engineers can simulate the changes in a protected environment before activating the delta data in the process. This allows you to identify and clear configuration errors before they can cause any problems in the process or lead to a plant shutdown.

See also

Starting Runtime on the configuration PC (Page 2199)

Exporting the user administration (Page 1352)

Importing the user administration (Page 1353)

Use and limitations of online delta loading

Introduction

The online delta loading capability is lost in some configurations.

Files and elements

The table below shows for which actions the online delta transfer capability is retained and for which actions it is not retained:

Element	Create	Delete	Edit	Remarks
Project name/properties of the project	-	-	No	-
HMI device	Yes	No	No	-
Properties of the HMI device	-	-	No	Edit: Changes are only activated in Runtime by deactivating the project and restarting Runtime.
User cycle (project properties)	No	No	No	-
Connections ("HMI Tags" editor)	No	No	No	-
Parameter of a connection ("HMI Tags" editor)	No	No	No	-
Alarms	-	-	-	See the "Online configuration for alarms" table below
Menus and toolbars	No	No	No	-

Online configuration for alarms

Element	Attribute	Create	Delete	Edit	Remarks
-	• Alarm number • Alarm group	-	-	No	-
-	• Alarm class	Yes	No	Yes	Inactive alarms are updated immediately if you change the alarm class without changing mandatory acknowledgment. If the mandatory acknowledgment of a currently active alarm and of an unacknowledged alarm is changed, the system acknowledges the active alarm. An active alarm that is changed to an alarm without "deactivated" state will be deactivated.
User-defined alarm group	-	Yes	No	Yes	-
Limit values for analog alarms	-	No	No	No	-
Alarm classes	-	Yes	No	Yes	Edit: Valid for most attributes. Exceptions are listed individually:

Note

Displaying PLC alarms

If you change the configuration of the option "Automatic update" after compiling and loading under "Runtime settings > Alarms > PLC alarms", the online delta load capability is lost.

See also

Compiling and loading with Multiuser Engineering (overview) (Page 2295)

Rules on online delta loading (Page 2298)

15.4.5.7 Updating a runtime project on an HMI device to V13 SP1 (RT Professional)

Introduction

If you have updated a TIA Portal V13 project to V13 SP1 on your engineering system, you can also update your runtime project on the HMI device from V13 to V13 SP1. The project is completely transferred from engineering system to the destination directory of your HMI device. The existing runtime data can be transferred as well.

To transfer an updated V13 SP1 project from engineering system to an HMI device, note the following:

- The corresponding old version of the runtime project must be available in the selected destination directory. In this case, you need to confirm the overwriting of the runtime project in the "Load preview" dialog.

Procedure

To transfer to an updated V13 SP1 project from engineering system to an HMI device and update an existing runtime project on the HMI device to version 13 SP1, follow these steps:

1. Select the project in the project tree on the engineering system.
2. Select the "Online > Download to device" menu command.
   Alternatively, you can select the command "Download to device> Software (all)" from the shortcut menu of the HMI device.
   The "Extended download to device" dialog opens.
3. Select the destination directory of the old runtime project under "Path of the target folder".
4. Click "OK".
   The project is compiled.
5. When the "Load preview" dialog opens, select the data to be transferred.
6. Confirm the overwriting of the target project.
7. Click "Load".
   The updated runtime project is transferred to the HMI device.

15.4.6 Compiling and loading with Multiuser Engineering

15.4.6.1 Compiling and loading with Multiuser Engineering (overview) (RT Professional)

Introduction

When using Multiuser Engineering for your projects, you should take into account the response when compiling the Runtime projects and the response when downloading them to HMI devices.

Compilation and loading to an HMI device is possible both in the server project view and the local session.

You can find more information on Multiuser Engineering in "Using Multiuser Engineering".

Basics

For HMI devices and RT Advanced, the following scenarios are possible in Multiuser Engineering:

- Compiling in the server project view
- Compiling in the local session
- Loading from the server project view
- Loading from the local session

Note

Complete loading from the server project view or from the local session does not differ from complete loading in a single-user project. With complete loading, the current Runtime project is loaded from the currently active view to an HMI device.

The following additional scenarios are possible for RT Professional:

- Online delta loading from the server project view
- Online delta loading from the local session

Note

Compiling and loading projects in a local session does not differ from compiling and loading in a single-user project, with the exception of online delta loading.

All rules, e.g. actions which revoke and restrictions for online delta loading capability also apply to online delta loading in Multiuser Engineering.

You can find additional information in Use and limitations of online delta loading (Page 2292).

In principle, you can execute all commands for compiling and loading in Multiuser Engineering projects:

- "Software (rebuild all)"
- "Compile > Software (only changes)"
- "Software (all)"
- "Online delta loading" (RT Professional only)

The term "project"

The term "project" has two different meanings in the context of compilation and loading. "Project" is the WinCC project on the configuration PC. "Project" is also the Runtime project that you create by compiling the configuration data of an HMI device and download to the HMI device.

- WinCC project: contains the configuration data of one or more HMI devices
- Runtime project: contains the compiled configuration data of an HMI device

Rules

The following basic rules apply to compiling and downloading in Multiuser Engineering:

- The Runtime project which was compiled in a local session always remains local and is not uploaded to the multiuser server. It cannot be saved in the multiuser server project.
- Only the Runtime projects which were compiled in the server project view are saved in the multiuser server project.

Note

Observe the rules on online delta loading.

You can find additional information in Online delta transfer for Runtime Professional (Page 2291).

You can find additional information on Multiuser Engineering on the Siemens YouTube channel: Multiuser Engineering - one team working simultaneously on a project (https://www.youtube.com/watch?v=n4oTZ2Gzg6U).

See also

Compiling in the server project view (Page 2297)

Compiling in the local session (Page 2298)

Rules on online delta loading (Page 2298)

Online delta transfer for Runtime Professional (Page 2291)

Online delta loading from the server project view (Page 2300)

Online delta loading from the local session (Page 2301)

Use and limitations of online delta loading (Page 2292)

15.4.6.2 Compiling in the server project view (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Basics

Compiling and downloading in the server project view is no different from compiling and downloading in a single-user project.

During the compiling of a project in the server project view, the multiuser server project is blocked. Other users cannot make changes to this server project during this time. The Runtime project compiled in the server project view is stored along with the engineering project in the central multiuser server. Blocking the multiuser server project ensures that the configuration data and the Runtime project remain in sync.

Note

When you compile and save in the server project view, other users obtain the Runtime project you have updated along with the engineering project when they "refresh" their local session. Other users do not have to recompile the changes you have made after an update.

Example: Compiling during check-in

You make changes to a tag in a local session. All prior changes have been compiled in the associated server project.

If there are no compilation errors, both projects - the modified engineering project (with the modified tags) and the compiled Runtime project - are saved in the central multiuser server project with the "Save changes" command.

If you skip compiling during the check-in, the project contains the changes that have been saved on the server.

The next user who creates a local session from the server project or updates an existing local session must compile your two changes in addition to his or her own changes.

Note

Working on a shared project through multiple local sessions increases the probability of error. It is therefore recommended to compile the project at check-in and eliminate any errors that are reported during compiling. In this way, you provide the next user with a project free of errors.

See also

Compiling and loading with Multiuser Engineering (overview) (Page 2295)

Compiling in the local session (Page 2298)

15.4.6.3 Compiling in the local session (RT Professional, RT Advanced, Basic Panels, Panels, Comfort Panels, RT Uni)

Basics

Compiling and downloading projects in the local session is no different from compiling and downloading in a single-user project.

Since the local session is a copy of the server project, the first compilation status of the local session is identical to that of the server project. If the server project contains contents that are not compiled or error messages occurred during the compiling, they are transferred to the local session.

Note

It is recommended to compile the project at check-in and eliminate any errors that are reported during compiling. In this way, you provide the next user with a project free of errors and avoid spreading errors.

Updating in the local session

If you update a project in the local session, the local session - including the compilation status - is completely replaced by the content of the server project. Only the changes marked for check-in are retained in the updated local session and generate additional compiling steps in the local session.

Example: Updating the local session

You make changes to a tag in a local session. All prior changes have been compiled in the associated server project.

You update the content of the local session by clicking the "Update" button. After the update, the local session obtains the compilation status of the server project. There are also compiling tasks for the acquisition of the modified tags.

See also

Compiling and loading with Multiuser Engineering (overview) (Page 2295)

Compiling in the server project view (Page 2297)

15.4.6.4 Rules on online delta loading

Overview

Online delta loading is only supported by WinCC RT Professional. All rules and restrictions for online delta loading also apply to online delta loading in Multiuser Engineering.

Observing these rules helps you to identify any potential errors in good time and prevent them. You can find additional information in Use and limitations of online delta loading (Page 2292).

The following figure shows an overview of online delta loading with Multiuser-Engineering:

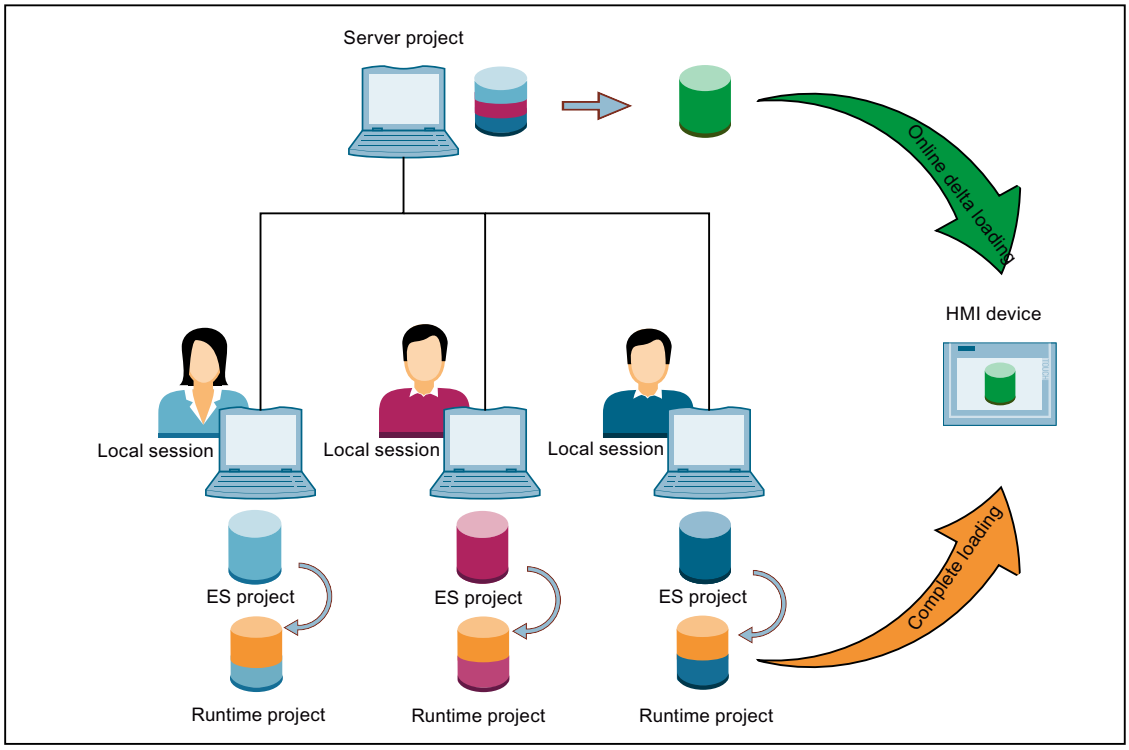

Rules for online delta loading from the server project view

- All rules and restrictions for online delta loading of the single-user projects also apply to online delta loading in Multiuser Engineering.
- Continuous online delta loading is only possible via the server project view without loading from local sessions.
- If you make a change in the server project view which revokes the online delta loading capability, the online delta loading capability in the multiuser server project is lost.

Rules for online delta loading from the local session

- All rules and restrictions for online delta loading of the single-user projects also apply to online delta loading in Multiuser Engineering.
- The local session must be created from a server project which features online delta loading capability.
- A local session itself has online delta loading capability.
- If you make a change in the local session which revokes the online delta loading capability, the local session loses the online delta loading capability.
- The online delta loading capability in the local session is lost during update.

See also

Compiling and loading with Multiuser Engineering (overview) (Page 2295)

Online delta transfer for Runtime Professional (Page 2291)

Use and limitations of online delta loading (Page 2292)

15.4.6.5 Online delta loading from the server project view (RT Professional)

Basics

To maintain online delta loading capability of the server project, all users may only make changes to the configuration which do not impair the online delta loading capability.

As long as all the users in the engineering project only make changes which do not revoke online delta loading capability, you can load the changed sections of configurations in WinCC Runtime Professional from different local sessions to the same HMI device via the server project view.

Note

With complete loading, the online delta loading capability of the HM device is lost.

Several users can make configuration changes in parallel in different local sessions. The steps "Compile > Software (only changes)" or "Online delta loading" must, however, be executed in succession in the server project view, as exclusive access rights are required for this.

You can find more information on Multiuser Engineering in "Using Multiuser Engineering".

Note

Continuous online delta loading from different local sessions is only supported via the server project view.

When you load to an HMI device from the server project view and close the server project view with "Discard changes", the online delta loading capability is lost. During the next load process to this device, the command "Load all" is executed.

See also

Compiling and loading with Multiuser Engineering (overview) (Page 2295)

Loading projects (Page 2280)

Online delta transfer for Runtime Professional (Page 2291)

15.4.6.6 Online delta loading from the local session (RT Professional)

Basics

Online delta loading to an HMI device is not only possible from the server project view, but also from the local session.

Online delta loading from a local session is subject to the following restrictions:

- After online delta loading from a local session, online delta loading can only be executed for this HMI device from this local session in the future.
- After this, only complete loading of the HMI device is possible in the server project view.
- Each update of the local session makes complete loading of the HMI device necessary.
- The Runtime project of the local session is never transferred to the multiuser server project and cannot be stored there.

Note

Online delta loading to an HMI device from the local session is not suitable for all configuration scenarios. An exact inspection of the configuration scenario beforehand is therefore recommended.

Note

Continuous online delta loading is only supported via loading from the server project view.

During the first load process after the creation or the update of the local session, the warning "Projects do not match" is displayed in the "Load preview" dialog. Carry out complete loading.

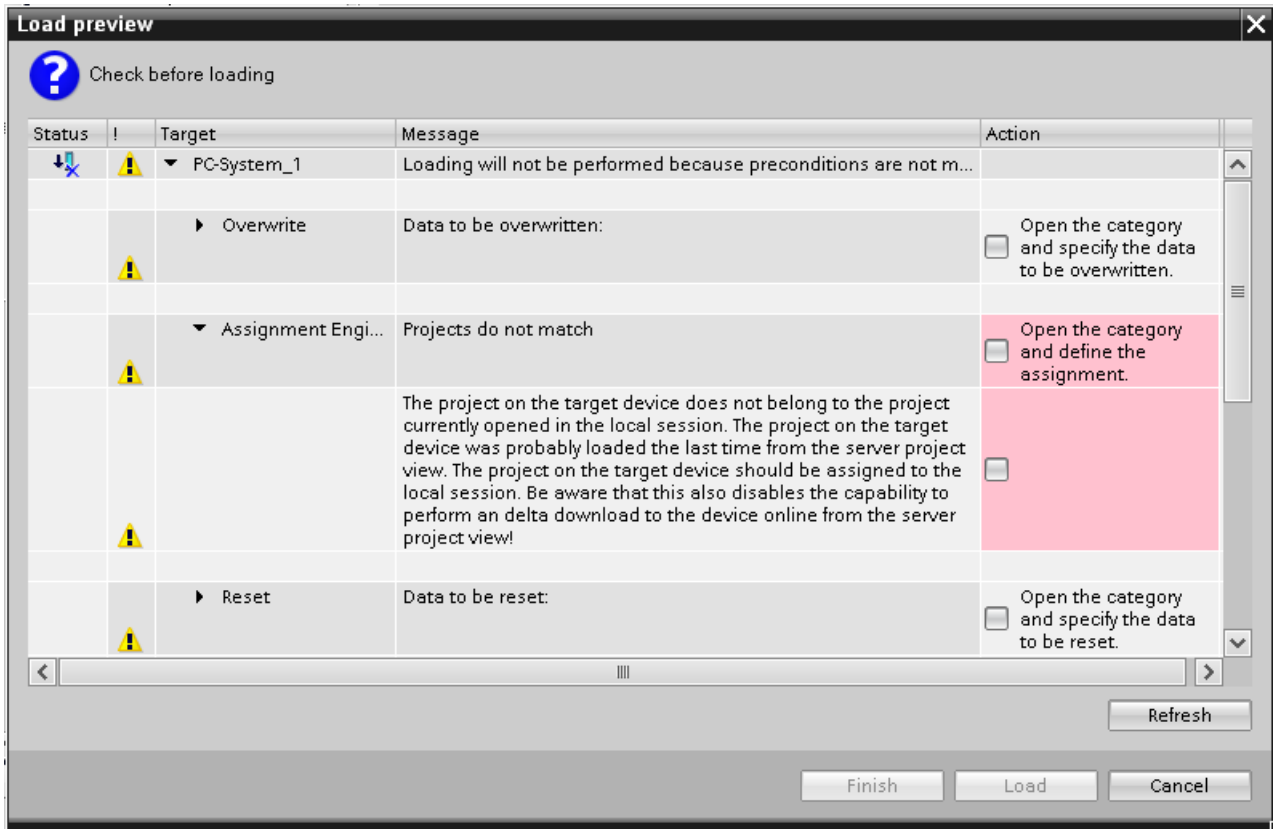

Preventing the loss of online delta loading capability in the server project

To maintain online delta loading capability of an HMI device in the server project, all users may only make changes to the configuration which do not impair the online delta loading capability.

If the online delta loading capability of an HMI device in the local session is lost due to a change, the online delta loading capability of this HMI device in the server project is only lost when you check in the change in question. Not all the errors reported in the local session lead to the loss of online delta loading capability in the server project on check-in of this change. If the check-in would result in the loss of online delta loading capability in the server project, automatic check-in is stopped. An error message will inform you of this. You can then decide whether you wish to continue or cancel check-in.

If you cancel check-in using "Discard changes", the online delta loading capability of the HMI device is retained. Your configuration changes are not applied in the server project.

You can find more information on the topic of "checking in objects in the server project" under "Working in the local session".

See also

Compiling and loading with Multiuser Engineering (overview) (Page 2295)

Online delta transfer for Runtime Professional (Page 2291)

15.4.7 Starting Runtime

15.4.7.1 Starting Runtime on the configuration PC (RT Professional)

Introduction

Once you have loaded your project locally you can start your project in Runtime on the configuration PC while you are performing configuration in WinCC. The following restrictions apply to online configuration:

- The project cannot be compiled in the background while Runtime is running on the configuration PC.
- The delta data of the project is compiled automatically when you load the project to an HMI device after having closed Runtime. You can also start compilation manually.
- The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime.
- If Runtime is running on the configuration PC, the changes are applied in the simulation after compilation. After loading, all changes are applied to the Runtime project.
  - Certain changes require a restart of Runtime for their activation.
  - Changes to the screen that is currently open become visible after a screen change.
  - With some changes, the option to compile configuration changes to the active Runtime is lost. If your changes lead to the loss of the delta load capability, you are notified accordingly during configuration.

Note

Closing Runtime automatically

If automatic transfer is activated on the HMI device and a transfer is started on the configuration PC, the running project is automatically terminated.

The HMI device then automatically switches to "Transfer" operating mode.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

To block access to the transfer settings and thus avoid unauthorized changes, assign a password in the Control Panel.

Requirements

- A project is open on the configuration PC.
- Runtime is installed on the configuration PC.

Procedure

To start Runtime on the configuration PC and to apply changes to the configuration to the active Runtime, follow these steps:

1. Activate the HMI device in the project tree.
2. Select "Online > Download to device".
3. When the "Load preview" dialog opens, select to start Runtime after loading.

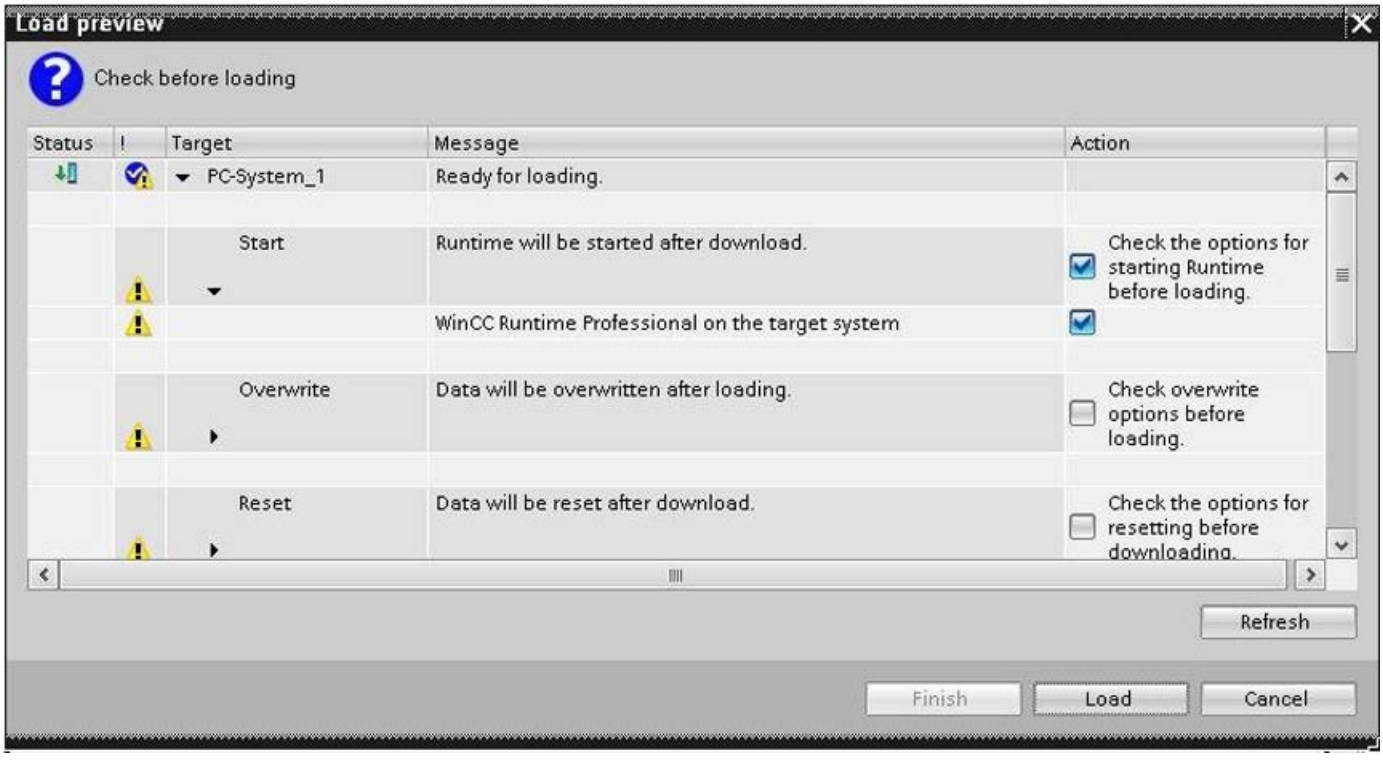

4. Click "Download".

Alternatively, use the "WinCC StartTool" to start the project.

Result

Runtime is started and the updated project displayed on the configuration PC.

See also

Use and limitations of online delta loading (Page 2292)

15.4.7.2 Starting Runtime Professional (RT Professional)

Introduction

If WinCC Runtime is also installed on a configuration PC, you start Runtime during loading. To do this, activate the corresponding option in the "Load preview" dialog.

If only WinCC Runtime is installed on an HMI device, load the project to the HMI device. To start WinCC Runtime on the HMI device use the "WinCC StartTool" or activate the subsequent start of WinCC Runtime during loading.

Requirements

"WinCC Runtime" is installed on the HMI device.

Note

A least 100 MB free memory must be available on the storage medium before the Runtime of a project can be started.

Note

Closing Runtime automatically

If automatic transfer is activated on the HMI device and a transfer is started on the configuration PC, the running project is automatically terminated.

The HMI device then automatically switches to "Transfer" operating mode.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

To block access to the transfer settings and thus avoid unauthorized changes, assign a password in the Control Panel.

Starting Runtime at an Engineering Station

1. Select the required project in the WinCC project tree.
2. Select "Online > Download to device".

3. Activate the start of WinCC Runtime after loading.

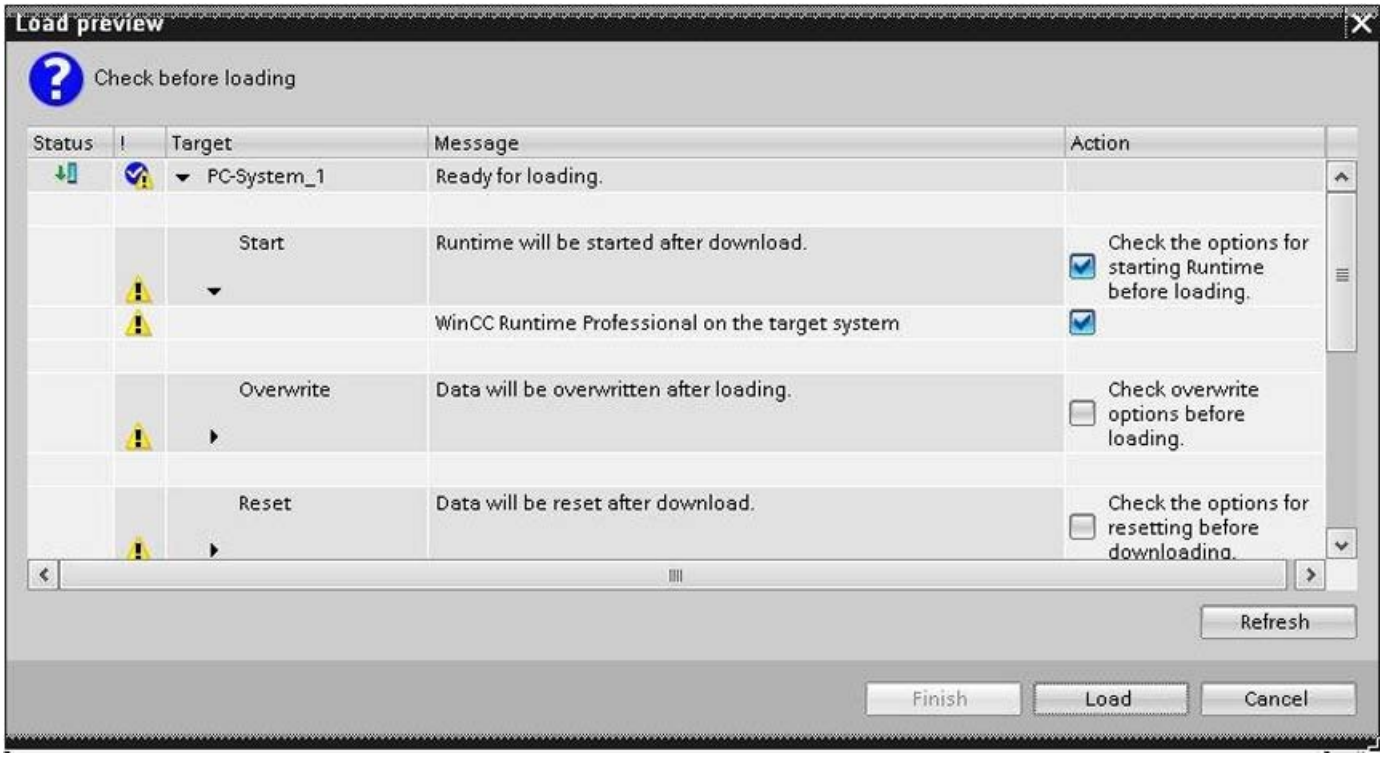

4. Click "Download".

Result

The project is compiled and loaded. During loading, all Runtime services that you selected in the Runtime settings of the HMI devices are started.

15.4.8 Basics of operating in Runtime

15.4.8.1 Overview of operating in Runtime (Professional)

Operation options

The hardware configuration determines how you operate the HMI device:

- Touch screen
  The operating elements shown in the dialogs are touch-sensitive. Touch objects are basically operated in the same way as mechanical keys. You activate operating elements by touching them with your finger. To double-click them, touch an operating element twice in succession.
- HMI device keyboard
  The operating elements shown in the screens are selected and operated using the keys of the HMI device.
- External keyboard
- External mouse

Operation feedback from operating elements

The HMI device provides optical feedback as soon as it detects that an operating element has been selected. The operating element receives the focus and is selected. This selection is independent of any communication with the PLC. Therefore this selection does not indicate whether the relevant action is actually executed or not.

The configuration engineer can also configure the selection of an operating element so that it deviates from the standard.

Optical feedback from operating elements

The type of optical feedback depends on the operating element:

- Buttons
  The HMI device outputs different views of the "Pressed" and "Unpressed" states, provided the configuration engineer has configured a 3D effect:
  - "Pressed" state:

  - "Unpressed" state:

  The configuration engineer determines the appearance of a selected field, for example, line width and color for the focus.
- Invisible buttons
  By default, invisible buttons are not displayed as pressed when they are touched. No optical operation feedback is provided in this case.
  The configuration engineer may, however, configure invisible buttons so that their outline appears as lines when touched. This outline remains visible until you select another operating element.
- I/O fields
  When you select an I/O field, the content of the I/O field is displayed against a colored background. With touch operation, a screen keyboard is displayed for the entering of values.

15.4.8.2 Operation with the touch screen

Overview of operation with the touch screen

Use the touch screen to operate the HMI device of the project that is running on your HMI device.

Operating the touch screen

NOTICE
Damage to the touch screen Do not touch the touch screen with sharp or pointed objects. Avoid tapping the touch screen with hard objects, and avoid constantly using motion control. Both can significantly reduce the useful life or even cause the failure of the touch screen. **Triggering unintended actions** You can trigger unintended actions if you touch several operating elements at the same time. Always touch only one operating element on the screen. Operating elements are touch-sensitive symbols on the screen of the HMI device.

Special features of operation using the touch screen

Operation with the touch screen is characterized by the following special features:

- Enable
  To enable an operator control, touch the touch screen with your fingers or with a stylus. To generate a double-click, touch the operator control twice in rapid succession.
- Value input
  You enter numbers and letters on the touch screen with a screen keyboard.
- Careful operation
  If you touch multiple operator controls at the same time, you may trigger unintentional actions.

Value input using the screen keyboard

The screen keyboard is displayed as soon as you operate a screen object that requires an input. Depending on the HMI device and the configured operating element, the system displays different screen keyboards for entering numerical or alphanumerical values. The screen keyboard is hidden again when input is complete.

File browser (Windows 8 or higher)

You can only operate the file browser dialog with a mouse, keyboard or on-screen keyboard (without using the touch function) on a Windows 8 or higher PC with touch screen. Use the file browser dialog of the Windows operating system with the help of a script on a touch screen PC with Windows 8 or higher.

Screen keyboard (Basic Panels)

Layout

The following figure shows the principal structure of a screen keyboard on a Runtime Professional HMI device.

Numbers

Letters

Using multi-touch functions

Multi-touch operation

Introduction

RT Professional supports multi-touch operation under Windows 7, Windows 8 and Windows 10. You operate objects on the multi-touch screens by touching them with one or two fingers. Multi-touch operation is supported by TIA Portal as of V13. Versions V12 or older only support single-touch operation.

Requirement

The used monitor must support multi-touch functions.

Basics of multi-touch operation

The multi-touch capable monitor can be operated by tapping, dragging, opening and pinching and swiping and detects the type of contact exactly. Tapping is briefly touching the screen with one finger and activating a button in this way, for example. Dragging is pressing a finger on the screen and moving it in a constant direction, similar to the drag-and-drop with the mouse. Opening and pinching involves touching the screen with two fingers and widening or narrowing the fingers. Swiping is briefly touching the screen with one finger and moving your finger in one direction.

You can use multi-touch in screens to scroll, enlarge or reduce screen sections or switch between screens in a screen stack. You use horizontal dragging to move the content of the screen to the left or right. You use vertical dragging to scroll up or down in the view. While you scale, an indicator is displayed to show the scale factor.

Supported gestures

Supported gestures

The following gestures are supported.

Gesture		Function
	Tap	You can select a button, for example, by typing.
	Drag	Use one finger to scroll horizontally and vertically, for example, in screens. You scroll horizontally and vertically at the same time by dragging vertically in screens.

Gesture		Function
	Scale	Pinch-to-zoom to enlarge or reduce the display of the screen. Objects can be displayed or hidden when scaling based on the configured zoom factor. This means, for example, that an object is only shown if a sufficient zoom factor allows the required object to also be displayed.
	Swipe	By swiping from the center to the right or to the left, you can navigate between the screens within a screen stack.

Note

Operating with three of more fingers

Do not use three or more fingers to operate the device. This can lead to incorrect operations.

Note

Settings for dragging and scaling

To unlock the "Drag" and "Scale" functions, select the "Slider" check box and clear the "Extended zoom" check box under "Runtime Settings > Screens > Properties Window". Also select the "Extended zoom" check box under "Properties > Properties > Miscellaneous" in the Inspector window of each screen.

See also

Hiding layers and objects in Runtime (Page 262)

Two-hand operation of operator controls

Two-hand operation of operator controls

Introduction

WinCC supports two-hand operation of operator controls with RT Professional. It ensures safe operation of operator controls which are used to change critical system settings, for example, control tags with machine limits.

Locked and unlocked operator controls

You define specific operator controls as "locked operator controls" for two-hand operation of operator controls. Locked operator controls usually cannot be operated in runtime. The operator can only operate the locked operator controls if he presses a release button at the same time.

In runtime, locked operator controls can only be accessed with the tab sequence if a release button is pressed at the same time.

Locked operator controls and release buttons

You can configure all operator controls as locked.

You must configure at least one button in the screen as release button. This can be any unlocked button.

Display in runtime

The locked operator controls are displayed as dimmed in runtime. The locked operator controls are completely visible when they are unlocked by means of the release buttons.

Simulation of projects with multi-touch functions

WinCC supports the simulation of configured multi-touch functions. Requirement is that your monitor supports multi-touch operation.

Locking and unlocking operator controls

You can lock and unlock operator controls in projects for multi-touch devices. Locked operator controls can only be operated in runtime if the operator presses a release button at the same time.

You can lock and unlock individual operator controls or several operator controls simultaneously.

Procedure

1. Configure operator controls of the type I/O field,button or slider.
2. Select the required operator control(s).

3. To lock the operator controls, disable the "Allow operator control" option under "Properties > Properties > Security".

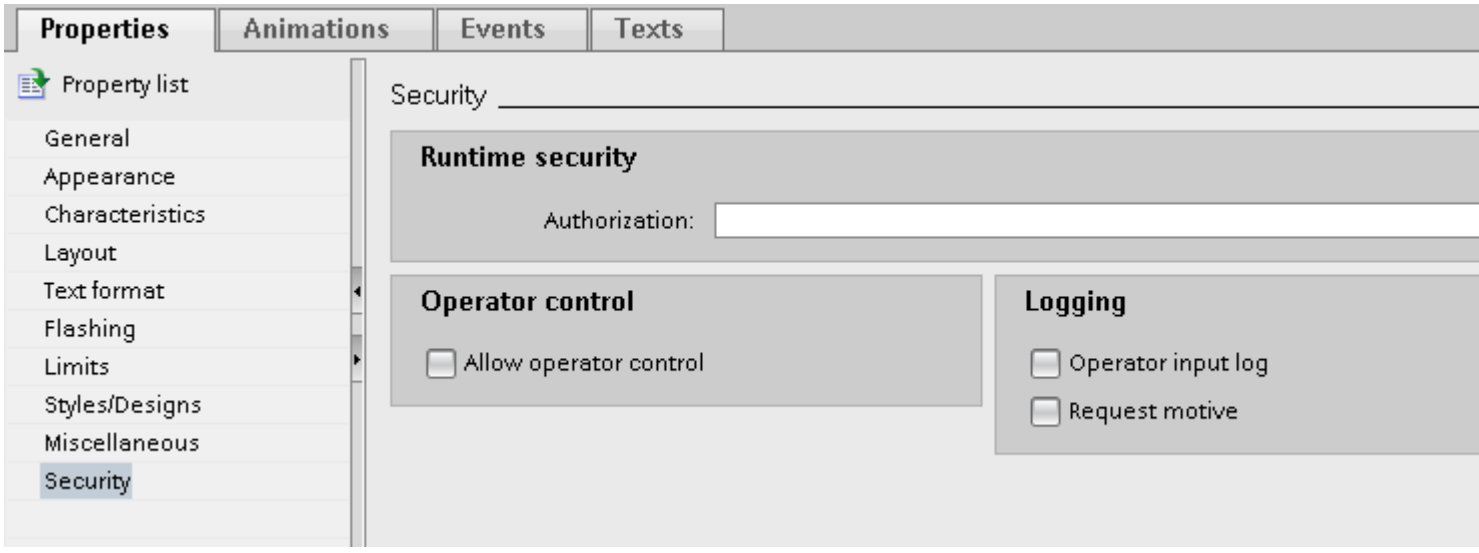

In Runtime, locked operator controls can only be operated by simultaneously pressing a release button that has been made dynamic with a corresponding system function.

4. To unlock the operator controls, enable the "Allow operator control" option under "Properties > Properties > Security".

Configuring release button in screen or template

For the operator to operate locked operator controls on multi-touch devices, at least one release button must be configured in the same screen.

Configure a release button so that when you press the button it releases the operator controls and locks them when it is no longer pressed.

If several objects have to be unlocked in a screen with a release button, expand the function list of the release button with an additional entry for each object.

Requirement

- One or more buttons are created in a screen or a template.

Procedure

1. Select the button that you want to configure as the release button.
2. Click "Properties > Events" in the Inspector window.
3. Select the "Press left mouse button" event.
4. Click on "Add function" in the table.
5. Select the "SetPropertyByConstant" system function.
6. Select the respective screen from the "Screen name" list.

7. Select the respective button from the "Screen object" list.
8. Select the "Allow operator control" property from the "Property name" list.
9. Enter "1" for the value.

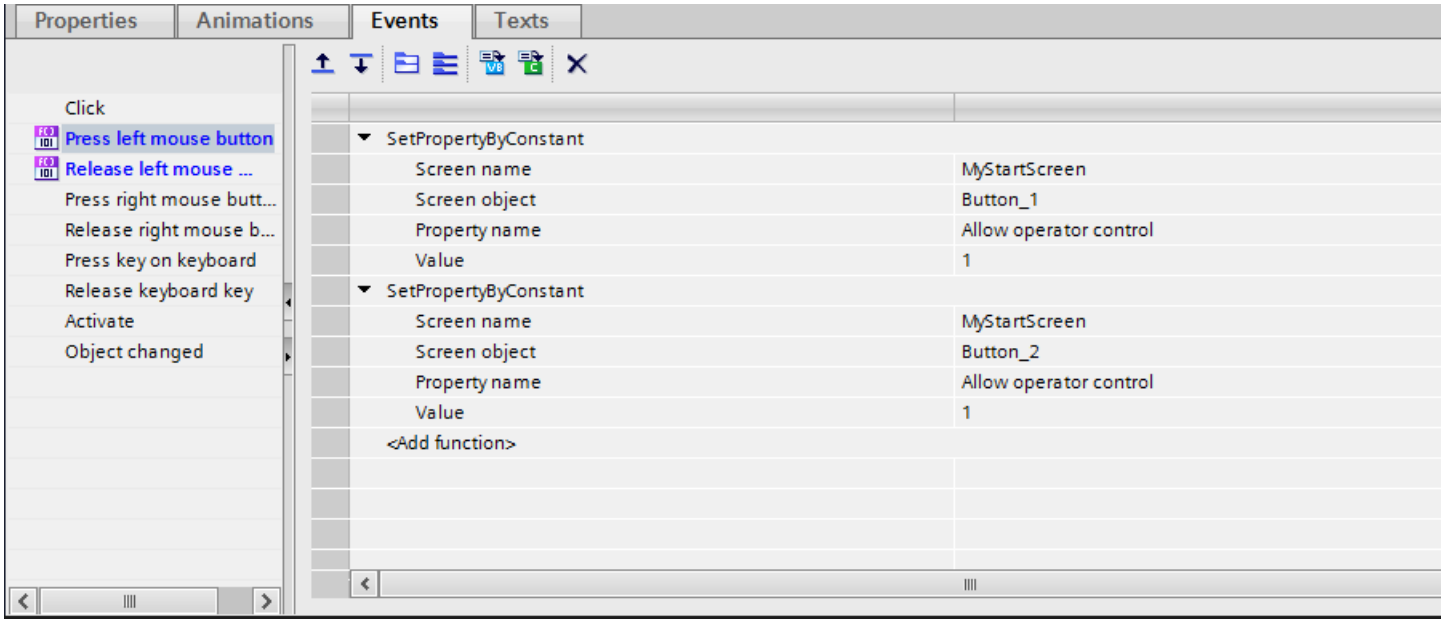

10. To have the control enable revoked when you let go of the release button, select the "Release left mouse button" event and enter "0" for the value.

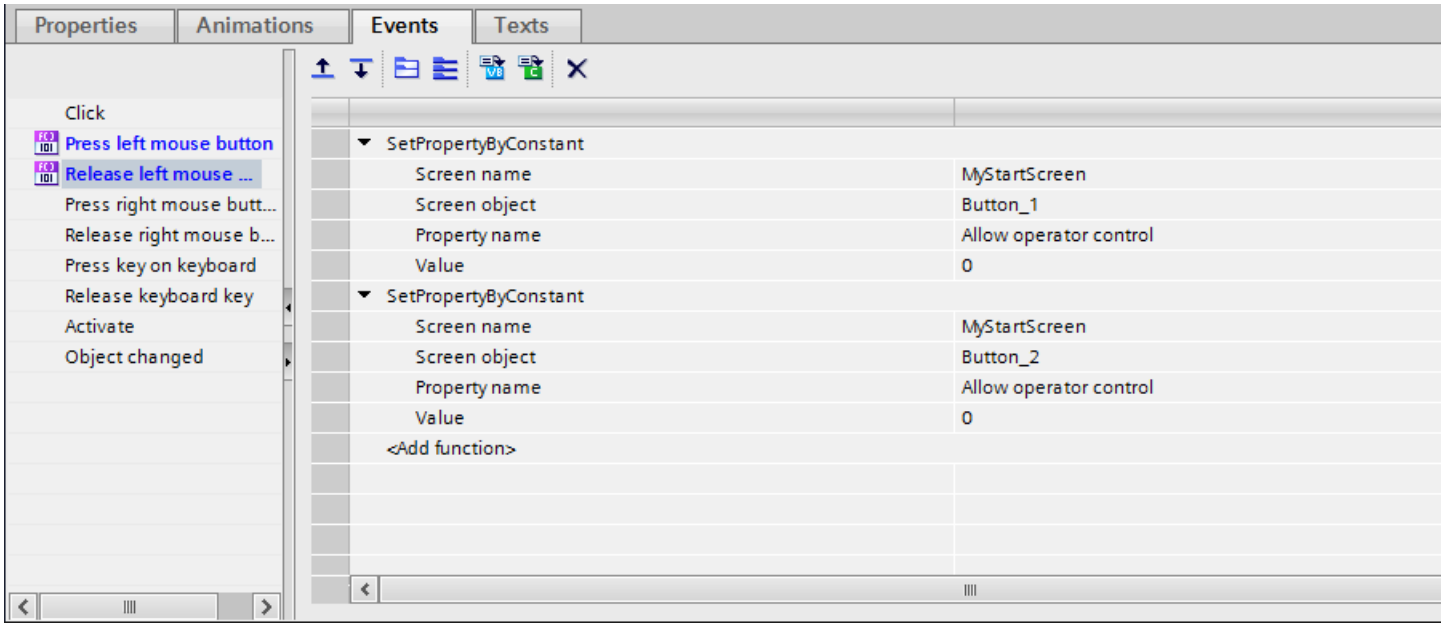

Locked operator controls can only be operated in runtime if the operator presses one of the configured release buttons at the same time.

15.4.8.3 Operation with keys

Overview of operation with keys

Introduction

Use the keys of the HMI device to operate the Control Panel / Start Center of your HMI device or the project that is running on your device. Depending on the device, control keys and function keys are available.

For more detailed information, refer to the operating instructions for your HMI device.

Control keys and shortcuts

Introduction

The following tables list the control keys available to operate the project.

Note

The availability of control keys is determined by the HMI device used.

You trigger functions on keyboard HMI devices either with a key or a shortcut. With shortcuts, you keep the first key pressed. Then you press the second key.

Navigating in the display

Key or shortcut		Function	Description
TAB		Tabulator	Selects the next operating element in the tab sequence
SHIFT	TAB	Tabulator	Selects the previous operating element in the tab sequence
◀ ▶ ▼ ▲		Cursor keys	Selects the next operating element to the left, to the right, above or below the current screen object Navigating in the operating element

Operation of operating elements

Key or shortcut		Function	Description
ENTER		ENTER key	• Controls buttons. • Applies and ends an entry • Opens a selection list • Toggles between character mode and standard mode within an input field A single character is selected in character mode. In this mode, you can scroll within the character set using the cursor keys.
SHIFT	▶ ◀ ▲ ▼	Positioning cursor	Positioning the cursor within an operating element, for example, in an I/O field
←		Delete characters	Deletes the character to the left of the current cursor position
INS DEL		Delete characters	Deletes the next character to the right of the current cursor position
ESC		Cancel	• Deletes the input characters of a value and restores the original value • Closes the active dialog.
FN	HOME	Scroll to start	Scrolls to the start page of a list
HOME		Scrolling back	Scrolls the list back by one page
FN	END	Scroll to end	Scrolls to the end of a list.
END		Scrolling up	Scrolls the list up by one page
ALT	▼	Open selection list	Opens a selection list
CTRL	ENTER	Accept value	Accepts the value selected in the selection list without closing the list

Enter shortcut

Key	Function	Purpose
A-Z	Toggle (numbers/letters)	Toggles the assignment from numbers to letters • No LED is lit: The number assignment is enabled. Pressing the key once switches to letter assignment. • The right or left LED is lit: The left or right letter assignment is enabled. Each time the key is pressed, the system toggles between the left letter assignment, the right letter assignment and the number assignment.
SHIFT	Shift (upper/lower case)	Used in shortcuts, for example, for switching to uppercase letters
FN	Toggle to additional keyboard layout	Some of the keys contain a blue special character in their bottom left corner, for example, the percent sign "%". To input these characters, press the relevant key in combination with the special character key shown on the left.
CTRL	General control function	Used in shortcuts, for example, for navigating trend views
ALT	General control function	Used in shortcuts, for example for the "Watch table" screen object

Acknowledge alarms

Key	Function	Purpose
ACK	Acknowledge	Acknowledges the currently displayed alarm or all alarms of an alarm group The LED is lit as long as an unacknowledged alarm is active.

Displaying infotext

Key	Function	Description
HELP	Displaying infotext	For the selected object, opens a window with the configured infotext, for example, for an alarm or an I/O field The LED is lit if an infotext is available for the selected object.

Key or shortcut

Function Keys

The assignment of the function keys (F1, F2, F3, etc.) is defined during configuration.

Function keys with global function assignment

A globally assigned function key always triggers the same action on the HMI device or in the PLC regardless of the screen displayed. The activation of a screen or the closing an alarm window, for example, is such an action.

Function keys with local function assignment

A function key with local function assignment is screen-specific and is therefore only effective within the active screen.

The function of a locally assigned function key can vary from screen to screen.

Within a screen, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Operating function keys

Note

Operating function key after screen change

If you press a function key after a screen change, the associated function may be triggered in the new screen before the new screen is fully displayed.

Direct keys

Introduction

Direct keys on the HMI device are used to set bits in the I/O area of a SIMATIC S7.

Direct keys enable operations with short reaction times that are, for example, a jog mode requirement.

Note

Direct keys are still active when the HMI device is in "offline" mode.

Note

If you operate a function key with direct key functionality in a running project, the direct key function is always executed, independent of the current screen contents.

Note

You only use direct keys when there is a connection via PROFIBUS DP or PROFINET IO.

Direct keys result in additional basic load on the HMI device.

Direct keys

The following objects can be configured as a direct key:

- Buttons
- Function keys

Depending on the HMI device and the configuration, direct keys are screen-specifically configured.

Additional information on direct keys can be found in your plant documentation.

15.4.8.4 Navigating in the display (BS)

Introduction

You navigate on the display of your HMI device as follows:

- between configured screen objects
- within screen objects
  When you select a complex screen object, the cursor focus switches to the screen object and follows the tab sequence there.
- in tables of screen objects

Procedure

- To navigate in the specified tab sequence, press the <TAB> key.
- To navigate freely between the operator controls, press the cursor keys.

Depending on the configuration of your HMI device, you can also use function keys or shortcuts for navigation.

When you operate your HMI device with the touch screen or with the mouse, you implicitly navigate by triggering a desired action. For this purpose, touch or click the operator control.

Result

The operator controls receive the cursor focus according to the selected sequence. You can trigger an action on the selected operator control.

For more detailed information, refer to the operating instructions for your HMI device.

15.4.8.5 Triggering an action

Introduction

Triggering an action at an operator control can mean the following:

- A command is executed.
  Example: Click a button to trigger a script or to execute a predefined function.
- An object is enabled.
  Example: To enter a value, select a table cell with the <Enter> key.

Requirement

- You have navigated to the operator control on which you want to trigger the action.
- The operator control has the cursor focus.

Procedure

- Press <Enter>.
  Or
- Touch the operator control on the touch screen once or twice in rapid succession.
  Or
- Click or double-click the operator control with the mouse.

Result

The following results are possible:

- The requested command is executed.
- The screen keyboard is opened and/or the cursor blinks in the input area of the operator control.
- The element is selected and can be moved.

For more detailed information, refer to the operating instructions for your HMI device.

15.4.8.6 Entering a value

Introduction

Depending on the input format, you enter numerical or alphanumerical values in an input field.

You enter these values depending on the existing hardware using the screen keyboard, the control keys of the HMI device or an external keyboard.

Requirement

- The object is an input field or table field.
- The operator control is enabled.

Entering a value

1. Enter the desired value.
2. To confirm the value and exit the field, press the <Enter> key.
3. To discard the value and exit the field, press the <Esc> key.

Result

A value is entered or discarded. You navigate as needed to the next operator control.

For more detailed information, refer to the operating instructions for your HMI device.

15.4.8.7 Moving operator controls

Introduction

In Runtime, you can move the movable operator controls of a screen object with the mouse or using the touch screen, for example, a slider or a scroll bar. Operation with the keyboard is described below.

Requirement

- A movable operator control is selected.

Procedure

- To move the operator control, proceed as follows depending on the operating element:
  - Standard for touch screen: Press the cursor keys.
  - Standard for keyboard devices: Press <SHIFT> and the cursor keys.
  - Switches: Press <ENTER>
  - Slider: Press <PgUp> or <PgDn>

1. To finish the movement, navigate to another screen object or operator control.

Slider procedure

1. To move the operator control, press the cursor keys.
2. To finish the movement, navigate to another screen object or operator control.

Result

The position of the movable operator control and the display in the screen object have changed.

For more detailed information, refer to the operating instructions for your HMI device.

15.4.8.8 Displaying infotext (Professional)

Application

The configuration engineer uses tooltips to provide additional information and operating instructions. The configuration engineer can configure tooltips for screens and operating elements.

The tooltip of an I/O field may, for example, contain information about the value to be entered.

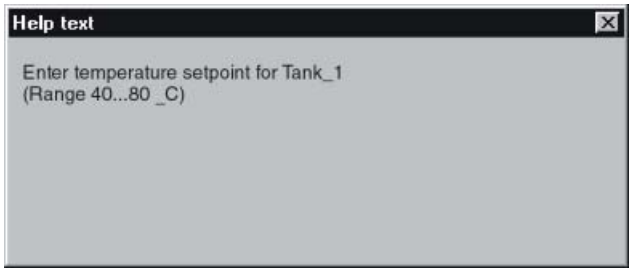

Procedure

Depending on your configuration, tooltips can be called via a configured operating element.

Additional information on this topic may be available in your plant documentation.

15.4.8.9 Changing Runtime language

Introduction

The HMI device supports multilingual projects. A corresponding operating element which lets you change the language setting on the HMI device in Runtime has been configured.

The project always starts with the language set in the previous session.

Requirement

- The required language for the project is available on the HMI device.
- The language switching function is linked to an operating element, for example, to a button.

Selecting a language

You can change project languages at any time. Language-specific objects are immediately displayed on the screen in the new language when you switch languages.

You can switch the language in Runtime in one of the following ways:

- Use a configured operating element to switch from one language to the next in a list.
- Use a configured operating element to directly set the required language.

15.4.8.10 Runtime data

Export data

To export data to a CSV file in Runtime, follow these steps:

1. Click "Export data" button in the toolbar of the display and operating object.
   A dialog opens, depending on the configuration.
2. If you have the appropriate permissions, select the file and the directory for the export.

   Note
   File name

   Even if you specify a file name extension other than "*.csv" for the file name, the runtime data are exported to a CSV file.

3. Start the export.

Note
Export dialog

If no dialog is displayed, the data export to the predefined file begins automatically.

15.5 Jumping to the configuration

15.5.1 Jumping to the configuration

Introduction

With WinCC Runtime Advanced, you have the option of defining a context-related switch from HMI to the configuration in the TIA Portal Engineering System. An operator can use this to jump to a specific project or a block in the engineering system from runtime or to check or modify the data. In addition, the operator can open or edit other objects of the opened project in the corresponding editors of the TIA Portal. Changing to the configuration enable fast and simple diagnostics of faults.

Depending on requirements, you can configure read/write access or read-only access in read mode.

Configuring jump to the TIA Portal

You configure the system function ShowBlockinTIAPortal for the jump from an alarm view or a GRAPH overview.

You switch to the project in the TIA Portal using the system function which you configure either at a button for the alarm view or at a GRAPH overview.

System function

You configure a change to an event in your program that triggered a hardware interrupt using the system function ShowBlockInTIAPortalFromAlarm. This provides you with the possibility to respond appropriately to this event and check the program for faults, such as a missing contact in the interlock.

You configure this system function at a screen object, for example, a button or a button of the toolbar in an alarm view. When you click the button, the necessary context is stored by the system. The corresponding position is automatically displayed in the program code after the switch to the engineering system.

For a description of the system function, refer to "System functions for Runtime Advanced > ShowBlockInTIAPortal"

Jumping to a protected project

You enable project protection for your entire project and configure roles and users for access. When a user switches from runtime to a protected project, access is granted on the basis of function rights that you have defined for the user in question. For example, the function right "Open the project in read-only" set in project protection disables the read and write access that you have configured in WinCC.

Enter the user name and the password in a logon dialog when you open a protected project.

15.5.2 Read mode in the TIA Portal

Introduction

With WinCC Runtime Advanced, you configure a context-related switch from HMI to the TIA Portal. You have the option here of configuring the jump with two different access types

- Read and write access
  After the jump to the TIA Portal, the operator can edit all project data in the engineering system and has read and write access to the hardware.
- Read-only access
  After the jump to the TIA Portal, the operator has only restricted access to the project data. The operator may not change the data in the engineering system and has only online read access to the hardware.
  Read-only access is implemented using read mode. It is not possible to change the project data in read mode.

Read mode

In read mode, the operator can only perform those configuration and diagnostic tasks that require no change to the project and online data. Online read access is also possible in read mode for the additional criteria analysis in the case of process error.

In read mode, the operator can use the navigation functions of the TIA Portal for the diagnostics of faults and analysis of the program.

- Project tree in the open project.
  The operator can access all available objects in the project through the project tree and can open these as read-only in the editors that support read mode. Editors that do not support read mode are not opened.
- Functions "Go to", read/write access, definition and device.
  The go-to functions help the operator to analyze the process states and data.
- Editors "Cross references", "Cross-reference information" and "Program information".
  The cross-reference functions provide the operator with an overview of the use of objects within the project.

Actions in read mode

In read mode, the operator can perform the following actions in the engineering system:

- Switch between the portal view and the project view.
- Open another project in TIA Portal using main menu command or using toolbar button.
- Access any connected device.
- Exit TIA Portal engineering system using main menu command.

Editors in read mode

The following TIA Portal editors can be opened in read mode:

- Program editor for displaying
  - Program blocks in the FBD, LAD, GRAPH, STL, SCL programming languages
  - Data blocks and UDTs
  - ProDiag function blocks
- Tag table
- Program information
- Cross-reference list
- Cross-reference information
- Devices & networks
- HMI editors
- Safety Administration Editor
- Search in project

Restrictions in read mode

In read mode, the operator cannot perform the following actions in the TIA Portal:

- Upload project, download project, compile project
- Start simulation
- Create new objects
- Edit project
- Modify tags in the program status
- Display watch table and force table
- Open "Online & diagnostics" editor
- Open editors of technology objects
- Open editors of local modules

15.6 Jumping to the configuration (RT Professional)

15.6.1 Jumping to the configuration (RT Professional)

Introduction

With WinCC Professional, you have the option of defining a context-related switch from HMI to the configuration in the TIA Portal Engineering System. An operator can use this to jump to a specific project or a block in the engineering system from Runtime or to check or modify the data. In addition, the operator can open or edit other objects of the opened project in the corresponding editors of the TIA Portal. Changing to the configuration enable fast and simple diagnostics of faults.

Depending on requirements, you can configure read/write access or read-only access in read mode.

Configuring jump to the TIA Portal

Depending on the application, two options are available for the change to the project in the TIA Portal:

- Change using the four Runtime API functions
  You configure a screen object that executes one of four available API functions in a C script.
- Change using the system function
  You configure a system function which enables the operator to switch to the program position that triggered the hardware interrupt after this interrupt has occurred.

Runtime API functions

You can use the API function to switch from a screen in WinCC Runtime directly to the program code. A check is first made at the jump to determine whether the TIA Portal is already open. If this is not the case, the TIA Portal is started automatically. The corresponding editor opens in the TIA Portal and a search is performed for the point of use, assignment, call or step.

Use one of the four API functions to open a project in the TIA Portal:

- OpenTIAPortalProject opens the project in the TIA Portal.
- OpenTIAPortalIECPLByCall is used for the LAD and FBD languages and shows the preceding logic of a network input of a standard block.
- OpenTIAPortalIECPLByAssignment is used for the LAD and FBD languages and shows the assignment to an operand and its preceding logic.
- OpenTIAPortalS7GraphByBlock is used for the GRAPH language and shows a step in a GRAPH sequencer.

For a description of the API functions, refer to "Interfaces > Runtime API > Functions for displaying the PLC code"

System function

You configure a change to an event in your program that triggered a hardware interrupt using the system function ShowBlockInTiaPortalFromAlarm. This provides you with the possibility to respond appropriately to this event and check the program for faults, such as a missing contact in the interlock.

You configure this system function at a screen object, for example, a button or a button of the toolbar in an alarm view. When you click the button, the necessary context is stored by the system. The corresponding position is automatically displayed in the program code after the switch to the engineering system.

For a description of the system function, refer to "System functions for Runtime Professional > ShowBlockInTiaPortalFromAlarm" (Page 1699)

Accessing a protected project

You enable project protection for your entire project and configure roles and users for access. When a user switches from Runtime to a protected project, access is granted on the basis of function rights that you have defined for the user in question. For example, the function right "Open the project in read-only" set in project protection disables the read and write access that you have configured in WinCC.

Enter the user name and the password in a logon dialog when you open a protected project.

15.6.2 Read mode in the TIA Portal (RT Professional)

Introduction

With WinCC Professional, you configure a context-related switch from HMI to the TIA Portal. You have the option here of configuring the jump with two different access types

- Read and write access
  After the jump to the TIA Portal, the operator can edit all project data in the engineering system and has read and write access to the hardware.
- Read-only access
  After the jump to the TIA Portal, the operator has only restricted access to the project data. The operator may not change the data in the engineering system and has only online read access to the hardware.
  Read-only access is implemented using read mode. It is not possible to change the project data in read mode.

Read mode

In read mode, the operator can only perform those configuration and diagnostic tasks that require no change to the project and online data. Online read access is also possible in read mode for the additional criteria analysis in the case of process error.

In read mode, the operator can use the navigation functions of the TIA Portal for the diagnostics of faults and analysis of the program.

- Project tree in the open project.
  The operator can access all available objects in the project through the project tree and can open these as read-only in the editors that support read mode. Editors that do not support read mode are not opened.
- Functions "Go to", read/write access, definition and device.
  The go-to functions help the operator to analyze the process states and data.
- Editors "Cross references", "Cross-reference information" and "Program information".
  The cross-reference functions provide the operator with an overview of the use of objects within the project.

Actions in read mode

In read mode, the operator can perform the following actions in the engineering system:

- Switch between the portal view and the project view.
- Open another project in TIA Portal using main menu command or using toolbar button.
- Access any connected device.
- Exit TIA Portal engineering system using main menu command.

Editors in read mode

The following TIA Portal editors can be opened in read mode:

- Program editor for displaying
  - Program blocks in the FBD, LAD, GRAPH, STL, SCL programming languages
  - Data blocks and UDTs
  - ProDiag function blocks
- Tag table
- Program information
- Cross-reference list
- Cross-reference information
- Devices & networks
- HMI editors
- Safety Administration Editor
- Search in project

Restrictions in read mode

In read mode, the operator cannot perform the following actions in the TIA Portal:

- Upload project, download project, compile project
- Start simulation
- Create new objects
- Edit project
- Modify tags in the program status
- Display watch table and force table
- Open "Online & diagnostics" editor
- Open editors of technology objects
- Open editors of local modules

15.6.3 Configuring read mode (RT Professional)

Introduction

In WinCC Professional, you have the option of configuring a context-sensitive switch to the TIA Portal in read mode. Read mode does not permit the operator to make any changes to project data.

Configuring read mode via a Runtime API function

If you implement the change to the TIA Portal using the API functions in C scripts, configure the use of read mode as a parameter of the API function. Set the read-only access or the read/write protection for the operators using a bit in the bit field dwFlags. Depending on the bit value, the TIA Portal is opened in read mode (TRUE) or not in read mode (FALSE).

For a description of the API functions refer to "Interfaces > Runtime API > Functions for displaying the PLC code".

Configuring read mode via the system function

If you configure the jump to the TIA Portal using the system function ShowBlockInTiaPortalFromAlarm, set the parameter Read mode:

- TRUE (default): TIA Portal is opened in read mode, the user cannot make any changes to project data.
- FALSE: TIA Portal is opened with read and right authorization.

For a description of the system function refer to "System functions for Runtime Professional > ShowBlockInTiaPortalFromAlarm". (Page 1699)

See also

ShowBlockInTiaPortalFromAlarm (Page 1699)

15.6.4 Example: Configuring read mode using the API function (RT Professional)

Introduction

The example below shows the program code for the call of the change to the TIA Portal. Activation of the read mode depends on the access authorization of the operator who logged on.

Requirements

- The "Supervisors" group is created in the user administration
- The "Supervisors" group is assigned the authorization "Go to TIA Portal" with the number 4

Example

```
#pragma code("kopapi.dll")
#include "kopapi.h"
#pragma code()
#pragma code("UseAdmin.dll")
#include "pwrt_api.h"
#pragma code()

BOOL result;
CMN_ERROR error;

const DWORD permLevelTiaP = 4;

DWORD dwFlags = KOPAPI_FLAG_TIAPORTAL_OPEN_BRINGINFRONT |

KOPAPI_FLAG_TIAPORTAL_OPEN_READONLY;
char* plc = "PLC_1";
char* contBlock = "OPC1_DB";
char* operand = "#s1";

if(PWRTCheckPermission(permLevelTiaP, 1))
{

dwFlags &= ~KOPAPI_FLAG_TIAPORTAL_OPEN_READONLY;
}

result = OpenTIAPortalIECPLByAssignment(dwFlags,
 GetTagChar("TiaPortalProjectPath"), plc, contBlock, operand, 444
"TIAPErrorTag", &error);
```

Result

Checks whether the logged on user has the operator authorization.

An operator who is assigned to the "Supervisors" group has the authorization to change to the TIA Portal. A user who is not assigned to this group cannot change to the TIA Portal.

15.7 Servicing the HMI device

15.7.1 Overview of HMI device maintenance (Basic Panels)

Structure

The following figure shows the software components of an HMI device and their relation to the engineering system.

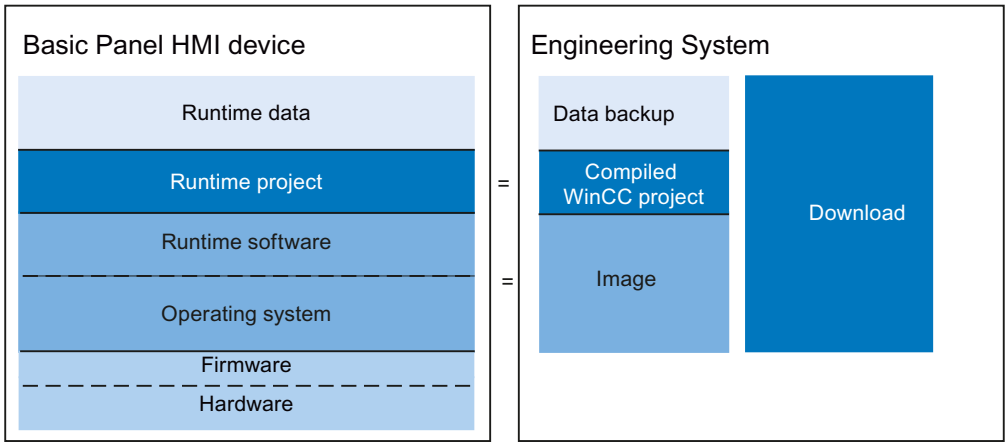

Runtime data

The Runtime data is generated during operation of the plant and stored on the HMI device. This data includes, for example, recipes and data for the user administration. This data is overwritten during loading. If required, save this data before loading a Runtime project.

Runtime project

The Runtime project contains the compiled configuration data for an HMI device. You download the Runtime project from WinCC to the HMI device.

Runtime software and operating system

Together, the Runtime software and operating system of an HMI device form the image. Various images are available for the HMI device. All images of an HMI device are available in WinCC. Depending on the configuration, download the appropriate image along with the Runtime project to the HMI device as required.

Firmware and hardware

The HMI device is delivered with preconfigured firmware and hardware.

15.7.2 Overview of HMI device maintenance tasks (Basic Panels)

Structure

The following figure shows the software components of an HMI device and their relation to the engineering system.

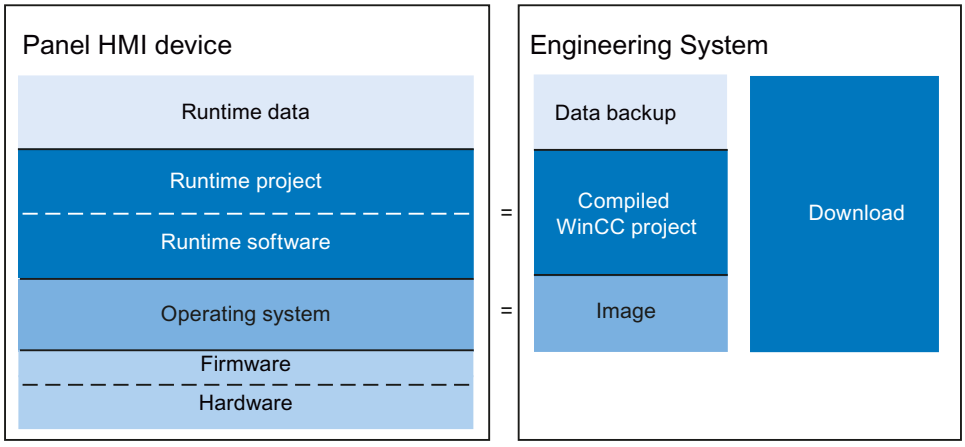

Runtime data

The Runtime data is generated during operation of the plant and stored on the HMI device. This data includes, for example, recipes and data for the user administration. This data is overwritten during loading. If required, save this data before loading a Runtime project.

Runtime project and Runtime software

The Runtime project contains the compiled configuration data for an HMI device. Download the Runtime project along with the Runtime software from WinCC to the HMI device.

Operating system

The HMI operating system is referred to as image. Various images are available for the HMI device. All images of an HMI device are available in WinCC. Depending on the configuration, download the appropriate image along with the Runtime project to the HMI device as required.

Firmware and hardware

The HMI device is delivered with preconfigured firmware and hardware.

15.7.3 Maintenance overview of the HMI device (RT Adv, RT Prof)

Structure

The following figure shows the software components of an HMI device and their relation to the engineering system.

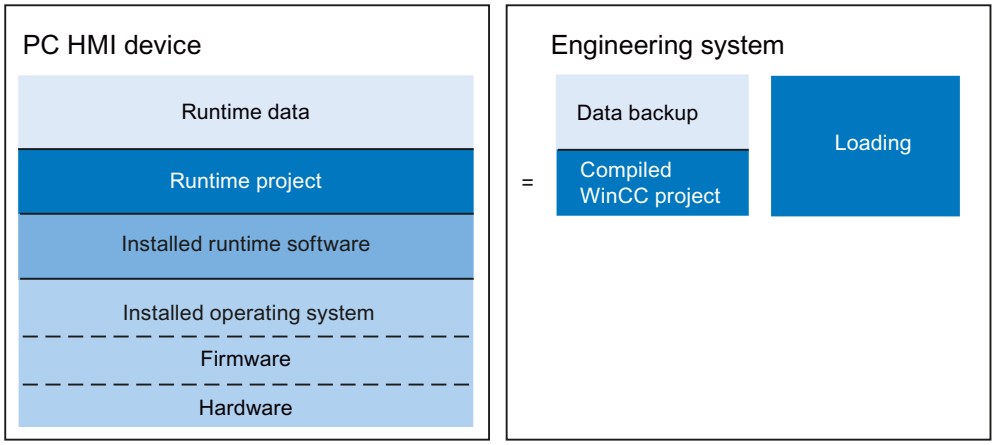

Runtime data

The Runtime data is generated during operation of the plant and stored on the HMI device. This data includes, for example, recipes and data for the user administration. This data is overwritten during loading. If required, save this data before loading a Runtime project.

Runtime project

The Runtime project contains the compiled configuration data for an HMI device. You download the Runtime project from WinCC to the HMI device.

Operating system and runtime software

The HMI device operating system and runtime software are installed on the HMI device independently of WinCC. This software is maintained in the same way as with every other PC.

Firmware and hardware

The HMI device is delivered with preconfigured firmware and hardware.

15.7.4 ProSave

Introduction

The "ProSave" service tool is installed by default when WinCC is installed. The ProSave functions are accessed in WinCC with the menu "Online > HMI Device maintenance".

Functional scope

ProSave provides all of the functions you need to manage data on the HMI device:

- Data backup and restoration of backed-up data
- Operating system update for HMI devices with Windows CE or below
- Transferring License Keys
- Installing and uninstalling drivers and options as well as information on installed options and options that can be installed on an HMI device
- Communication settings (transferred from WinCC)

See also

Overview of HMI device maintenance (Basic Panels) (Page 2334)

15.7.5 Backup of HMI data

Introduction

Regular backups of HMI device data keep downtimes to a minimum, for example, when you replace a device. You simply transfer the backup data to the HMI device, restoring the original status.

Data backup with WinCC

If an HMI device is connected to the configuration PC, you can back up and restore HMI device data from the configuration PC using WinCC.

Scope of data backup

Which data is backed up and restored depends on the type of HMI device:

- Complete backup
  Depending on the HMI device: Runtime, firmware, operating system, configuration, recipes, user administration, settings

  Note

  License Keys can only be stored only in the following HMI devices:
  - Mobile Panel Wireless
  - Comfort Panel

  The License Keys are not saved for the following HMI devices:
  - all other Windows CE HMI devices

- Firmware/configuration
- Recipes only
- Only recipes in CSV format
- User administration only

A backup file with the extension *.psb is generated when you backup the data of an HMI device.

As a general rule, you can backup the data to any storage medium. If the HMI device is networked, you can also backup the data to a server.

Note

Scope of data backup for Windows CE HMI devices

Data backup secures the contents of the flash memory. Alarm logs and process value logs are generally saved on the external storage medium. Alarm logs and process value logs are therefore not backed up. If necessary, back up the contents of the memory card separately.

Please note the following when performing a complete data file backup and restore operation for Windows CE devices:

- A full backup includes all options installed. As a rule, the backup includes all options data that is still available after "POWER OFF".
- All data on the device, including License Keys and the operating system, are permanently deleted when you perform complete data restoration.
- If the data restoration was interrupted, execute the command "Reset to factory settings". Restart data restoration.

Note

Use interfaces with high bandwidths, e.g. USB or Ethernet to backup and restore data.

Note

For Windows CE devices, you can also backup the data directly to a CF or PC card, independent of ProSave and WinCC. For additional information, refer to the relevant operating instructions.

See also

Backing up and restoring data of the HMI device (Page 2339)

Overview of HMI device maintenance (Basic Panels) (Page 2334)

15.7.6 Backing up and restoring data of the HMI device

Note

Use the restore function for project data only on operating devices which were configured using the same configuration software.

Requirement

- The HMI device is connected to the configuration PC
- The HMI device is selected in the project tree.
- If a server is used for data backup: The configuration PC has access to the server

Backup of the data of the HMI device

Proceed as follows to backup the data of the HMI device:

1. Select the "Backup" command from the "Online > HMI Device maintenance" menu.
   The "Create backup" dialog box opens.
2. Select the type of the PG/PC interface and the target device, and click "Create".
   The "SIMATIC ProSave" dialog box opens.
3. Select the data to backup for the HMI device under "Data type".
4. Enter the name of the backup file under "Save as".
5. Click "Start Backup".

This starts the data backup. The backup operation takes some time, depending on the connection selected.

Restoring the data of the HMI device

Proceed as follows to restore the data of the HMI device:

1. Select the "Restore" command from the "Online > HMI Device maintenance" menu.
   The "Restore backup" dialog box opens.
2. Select the type of the PG/PC interface and the target device, and click "Load".
   The "SIMATIC ProSave" dialog box opens.

3. Enter the name of the backup file under "Save as".
   Information about the selected backup file is displayed under "File information".
4. Click "Start Restore".

This starts the restoration. This operation takes some time, depending on the connection selected.

Backup/Restore from the "Backup/Restore" dialog in the Start Center of the HMI device

The "Backup/Restore" function is enabled for MMC, SD memory cards and USB mass storage devices. The storage media supported depend on the HMI device.

Note

Connections between Basic Panels and controllers

If you use the "Backup/Restore" function, a maximum of two connections from Basic Panels to the following controllers are possible at any given time:

- SIMATIC S7-1200
- SIMATIC S7-1500

For more information on this topic, refer to the device manual of the employed HMI device.

See also

Backup of HMI data (Page 2337)

Overview of HMI device maintenance (Basic Panels) (Page 2334)

15.7.7 Updating the operating system

Introduction

Update the HMI device image if the version is incompatible with the configuration. The image version matches the device version.

Update the operating system and Runtime software of the HMI device with the help of the device version. While loading the project, you may be prompted to run an automatic update of the device version, depending on the protocol used.

Loading will then continue. Loading of the project is otherwise aborted. In this case, perform the update of the device version manually.

Note

A device version can only be updated on HMI devices that are not PC-based.

Updating the device version

Connect the HMI device to the configuration PC to update the device version. If possible, use the interface providing the highest bandwidth for this connection, e.g. Ethernet. An update of the device version via serial connection may take up to an hour.

"Reset to factory settings"

If the operating system on the HMI device is no longer operational, update the operating system and reset the HMI device to the factory settings.

Note

Resetting to factory settings via Ethernet for Basic Panels

You require the following to reset the HMI device to the factory settings via Ethernet:

- the MAC address of the HMI device
- the available IP address
- the programming device/PC interface of the configuration PC set to Ethernet TCP/IP

The programming device/PC interface is configured using the control panel of the configuration PC. Select" "S7ONLINE (STEP7) -> TCP/IP" in the "Access point of the application" field.

Note

Reset to factory settings for Comfort Panels

To reset the Comfort Panels to factory settings, please use ProSave.

With the exception of Comfort Panels, you can reset all HMI devices to factory settings with the menu item "Online > HMI Device maintenance > Reset factory settings".

See also

Updating the operating system on the HMI device (Page 2342)

Overview of HMI device maintenance (Basic Panels) (Page 2334)

15.7.8 Updating the operating system on the HMI device

If possible, you should use the interface with the highest bandwidth for this connection, such as Ethernet. Updating the operating system via a serial connection can take up to an hour. When you update the operating system, the Runtime software on the HMI device is also updated and the device version is changed.

NOTICE
Updating the operating system deletes all data on the HMI device When you update the operating system you delete data on the target system. For this reason, you should back up the following data beforehand: • User administration • Recipes Resetting to factory settings also deletes the License Keys. Back up the License Keys before you reset the system to factory settings.

Requirement

- The HMI device is connected to the configuration PC.
- The HMI device is selected in the project tree.

Updating the operating system

Proceed as follows to update the operating system:

1. Select the "Update operating system" command from the "Online > HMI Device maintenance" menu.
   The "Update operating system" dialog box opens.
2. Select the type of the PG/PC interface and the target device, and click "Update".
   The "SIMATIC ProSave [OS-Update]" dialog opens. The path to the image is preset.
3. If required, you can select a different path for the image that you want to transfer to the HMI device.
4. Click "Update OS".

This starts the update. The update operation can take time, depending on the connection selected.

Resetting the HMI device to factory settings

To reset the HMI device to factory settings, proceed as follows:

1. Switch off power to the HMI device.
2. Connect the HMI device to the Engineering Station.
3. Select the "Update operating system" command from the menu under "Online > HMI Device maintenance" on the configuration PC in WinCC.
   The "Update operating system" dialog box opens.

4. Select the type of the PG/PC interface and the target device, and click "Update".
   The "SIMATIC ProSave [OS-Update]" dialog opens. The path to the image is preset.
5. If required, you can select a different path for the image that you want to transfer to the HMI device.
6. Activate "Reset to factory settings".
7. Click "Update OS".
8. To reset to factory settings, switch on the power to the HMI device again.
   This operation can take time.

Result

The operating system of the HMI device is operational and up-to-date.

See also

Updating the operating system (Page 2340)

Transferring license keys (Page 2345)

Managing licenses (Page 2345)

Overview for loading of projects (Page 2179)

Backing up and restoring data of the HMI device (Page 2339)

Overview of HMI device maintenance (Basic Panels) (Page 2334)

15.7.9 Updating the operating system of the HMI device from a data carrier

Introduction

You can update the operating system using a data carrier for Comfort Panels and Mobile Panels.

You can find the HMI image files, for example, in the installation directory of WinCC under: "\Siemens\Automation\Portal V1x\Data\Hmi\Transfer\<HMI device image version>\Images".

NOTICE
Data loss All data on the HMI device, including the project and HMI device password, is deleted during a restore operation. License keys are only deleted after a security prompt. Back up your data before the restore operation, if necessary.

Requirements

- The HMI device image file is located in the "SIMATIC.HMI\Firmware\" directory on your data carrier, e.g. a SIMATIC HMI Memory card or an industry-grade USB stick.
- The data carrier with the relevant HMI device image file including operating system is inserted in the HMI device.

Procedure

1. Open the "Control Panel" on your HMI device.
2. Open the "Service & Commissioning" dialog using the "Service & Commissioning" icon.
3. Change to the "OS Update" tab.
4. Press "Next".
   The "Update OS image from external memory" dialog opens.

   Note

   If there is no storage medium or a defective storage medium in the HMI device, the "0 devices found" message is displayed. Insert the storage medium or replace the storage medium.

5. Press the "Refresh" button, if necessary.
   The "Accessible devices:" group is updated. The HMI device checks the storage medium. You can find information on this storage medium in the "status information" field.
6. Select the storage medium with the required HMI device image in the "Accesible devices:" group.
7. Press "Next".
   The "Update OS image from:\Storage Card USB" dialog opens.
8. Select the required HMI device image file in the "Firmware files on" group.
9. If you need information about the selected file, press the "Details" button.
   The "Properties of image file" dialog containing the following information is displayed:
   - "Supported": HMI devices that are compatible with the HMI device image
   - "Image version": Version of the HMI device image
   - "Image size": Size of the image file
   - "Creation": Date the image file was created
10. To delete the selected file, press the "Delete" button.
    The "Delete confirmation" dialog is displayed. The file will be deleted when you press "OK".
11. To restore the data of the selected file, press the "Update" button.
    The "Update settings" dialog informs you that the settings in the Control Panel will be kept and offers you the option of keeping or deleting license keys present on the HMI device.
12. Press "Update".
    The "Update OS Image" dialog opens.
13. To start restoring the operating system, press "Yes".
    The "Transfer" dialog is displayed. A progress bar shows the course of the restore. The HMI device then restarts.

Note

After the restore, it may be necessary to recalibrate the touch screen.

15.7.10 Transferring license keys

Introduction

You need a license for certain WinCC Runtime options you may want to install on an HMI device. Usually, the necessary licenses supplied as "License Keys" on a data medium, e.g. USB stick. The "License Keys" can also be made available on a license server.

Use "Automation License Manager" to transfer the "License Keys" to or from an HMI device. The "Automation License Manager" is included automatically when you install WinCC.

NOTICE
Backing up License Keys In the following cases you have to backup the "License Keys" in order to prevent deletion of the "License Keys": • Prior to the update of the operating system of a Windows CE HMI device • Prior to restoring the data from a full backup "License Keys" on an HMI device are backed up depending on the HMI device configuration. For more information, refer to the operating instructions of the corresponding HMI device.

See also

Managing licenses (Page 2345)

15.7.11 Managing licenses

Requirement

- The HMI device is connected to the configuration PC or the PC running the "Automation License Manager".
- If you are using the configuration PC: The HMI device is selected in the project tree.

Procedure

To transfer license keys, follow these steps:

1. Open the "Automation License Manager". Go to the Windows Start menu and start "Automation License Manager" on a PC on which WinCC is not installed.
   The "Automation License Manager" starts.
2. Select the "Connect HMI device" command from the "Edit > Connect target system" menu.
   The "Connect target system" dialog opens.
3. Select the HMI device type in the "Device type" area.
4. Select the "Connection".
5. Configure the "connection parameters" associated with the selected connection.

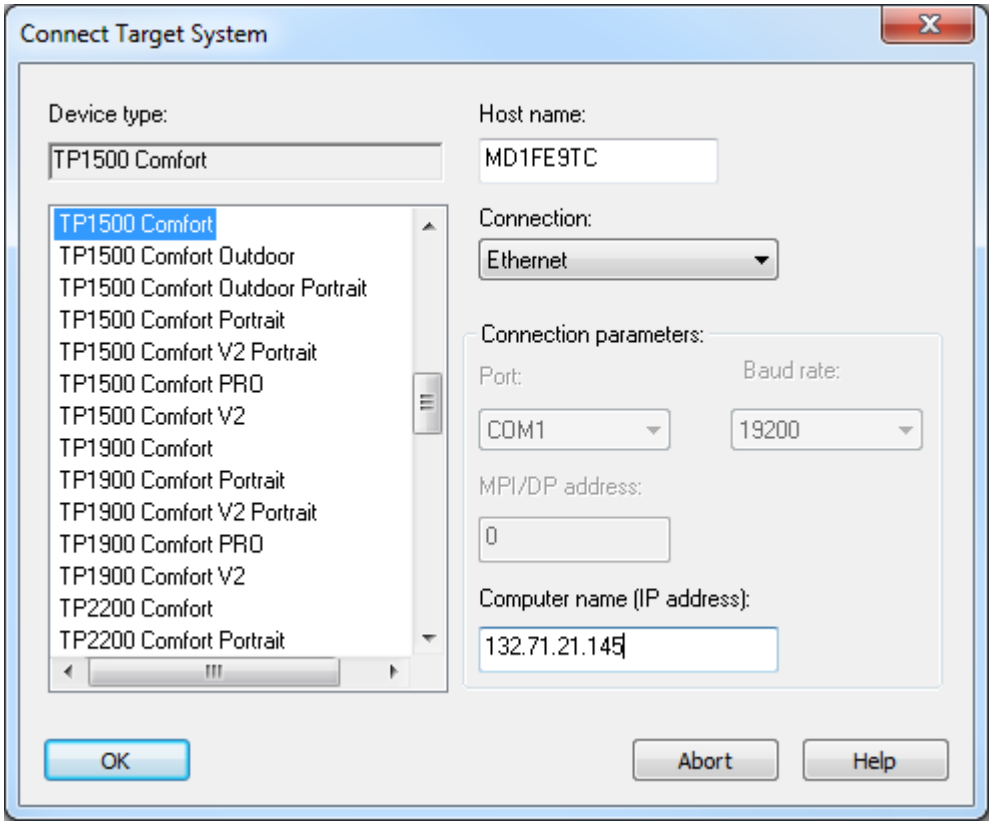

6. Click "OK".
   The connection to the HMI device is now set up. The connected HMI device is displayed in the left pane of "Automation License Manager".
7. Transfer the "License Keys" to the HMI device:
   - In the left pane, select the drive on which the "License Keys" are located.
     The "License Keys" are displayed on the right pane.
   - Select the "License Keys"
   - Drag-and-drop the "License Keys" to the HMI device.

You can also remove License Keys from the HMI device using drag-and-drop.

Alternative procedure

You can also start the "Automation License Manager" from WinCC on a PC with a WinCC installation: Select the "Authorize/License" command in the "Online > HMI Device maintenance" menu.

Result

The "License Keys" are transferred to the HMI device.

To backup the "License Keys" from the HMI device, drag-and-drop the "License Keys" from the HMI device to an available drive.

See also

Transferring license keys (Page 2345)

15.7.12 Installing and uninstalling an option

Introduction

You can install the following options on an HMI device:

- Additional options supplied with WinCC
- Options purchased in addition to WinCC

Which options you can install depends on the HMI device type.

Requirement

- The HMI device is connected to the Engineering Station, or to the PC with ProSave.
- The HMI device is selected in the project tree.

Procedure

To install an option on the HMI device, proceed as follows:

1. Select the "Options" command from the "Online > HMI Device maintenance" menu.
   The "Load options" dialog opens.
2. Select the type of the PG/PC interface and the target device, and click "Load".
   The "SIMATIC ProSave" dialog box opens.
   All available options and the previously installed options are displayed.
3. To display the installed options on the HMI device, click "Device status".
4. To install an option on the HMI device, select the option under "Available options" and click ">>".
5. To uninstall an option from the HMI device, select the option under "Available options" and click "<<".

Result

The selected options are installed on or uninstalled from the HMI device.

15.7.13 Loading Asian fonts to the HMI device (RT Advanced, Panels, Comfort Panels)

Introduction

When using Asian languages, only the Asian characters that were used in configuring the HMI device are loaded to the HMI device during the transfer. If the texts are sent directly from the controller to the HMI device as with controller alarms or with ProDiag, you may encounter display problems. You can load the complete Asian font to the HMI device in these cases.

Procedure

1. Select the HMI device in the project tree.
2. Select the "Options" command from the "Online > HMI device maintenance" menu.
   The "Load options" dialog opens.
3. Select the type of the PG/PC interface and the target device, and click "Load".
   The "SIMATIC ProSave" dialog box opens.
4. Select the respective font under "Available options" and click ">>".
   The font is loaded to the HMI device.

See also

Sending a complete alarm from the controller to the HMI device and automatic update (Page 950)

Sending a complete alarm from the controller to the HMI device and automatic update (Page 948)

Making characters available for HMI tags of the type WString/WChar (Page 554)

15.8 Printer driver (Comfort Panels)

15.8.1 Validity (Comfort Panels)

Validity

This document includes a description of the enhancements made possible with optional package printer driver.

Install the optional package to implement the extended functionality. The installation provides new printer drivers for several HMI devices.

If necessary you can install individual printer drivers with the tool SIMATIC ProSave on your HMI device. For more details on which printer drivers are available for which HMI devices please refer to the section "Supported HMI devices".

Installation

Please follow the instructions in the "Installation" section to install the "Optional package printer driver V1.4".

Application example

You can find an application example on the subject "Printing with HMI Panels" on the internet at

"Printing with SIMATIC Comfort Panels" application example (https://support.industry.siemens.com/cs/ww/en/view/58205602)

See also

"Printing with SIMATIC Comfort Panels" application example (https://support.industry.siemens.com/cs/ww/en/view/58205602)

15.8.2 Supported HMI devices (Comfort Panels)

Contents of the add-on package

The "Add-on package printer driver V1.4" contains the following printer drivers for the specified HMI devices.

The printer driver supports all printer options of the HMI device.

HMI device	Brother P-Touch QL 650 TD [1]	Brightek WH-AB, WH-C1, WH-E19	Postscript
KP400 Comfort KTP400 Comfort	USB	USB	Yes
KP700 Comfort TP700 Comfort	USB	USB	Yes
KP900 Comfort TP900 Comfort	USB	USB	Yes
KP1200 Comfort TP1200 Comfort	USB	USB	Yes
KP1500 Comfort TP1500 Comfort	USB	USB	Yes
TP1900 Comfort	USB	USB	Yes
TP2200 Comfort	USB	USB	Yes
USB: Connect printer directly to HMI device via USB port [1] Report printout only.			

Note

Please note that Postscript is a well-known and open communication protocol and theoretically, for this reason, statements could be changed by third parties. It is recommended that the printer be connected by USB in plants where security is especially critical.

15.8.3 Setting up the printer driver (Comfort Panels)

15.8.3.1 Postscript (Comfort Panels)

Define PostScript V1.4 as printer on the HMI device

1. Open the Control Panel.
2. Start the "Printer" application.
3. Under "Printer Language" in the "Printer Properties" dialog, select: the entry "PostScript V1.4".

Note

Not all printers support PostScript. Make sure that your printer supports this printer language.

Select the "Draft mode" option to increase the print speed.

15.8.3.2 HTML (Comfort Panels)

Introduction

The application "HTMLSettings.exe" is available on the desktop of the HMI device after installation of the "HMTL V1.3" printer driver.

Selecting the storage location

1. Open the application "HTMLSettings.exe" on the desktop of your HMI device.
2. Open "Storage Location" in the tab.
3. Under "Primary Location", select "USB-Stick", for example, for the storage location.
4. Select the directory in which the file is to be saved after printing.

Selecting the file name

1. Open the "HTMLSettings.exe" application
2. Open the "HTML File Names" tab
3. Select the settings for the names of the HTML file for the print-out.
4. Click on "Apply" to confirm your settings.
5. Select "Default" to retain the default settings for file name assignment.

Selecting columns for printing reports

1. Open the "HTMLSettings.exe" application
2. Open the "Columns Attributes" tab.
3. Select an attribute in the "Available Columns" category:
4. Add this attribute to your selection with the ">>" button.
5. If required, define whether the selected attribute is to be displayed as an image or as a hyperlink in the HTML file.
6. To deselect attributes from the "Selected Columns" category, select an attribute and click the "<<" button.

Selecting style sheets

1. Open the "HTMLSettings.exe" application
2. Open the "Advanced Options" tab.
3. Under "Enter Style Sheet File Path", select the storage location of the style sheets you wish to use.
4. Select the file to be used.

5. Select the directory for links.
6. Select the directory for images.

Defining HTML V1.3 as printer on the HMI device

1. Open the Control Panel.
2. Start the "Printer" application.
3. Under "Printer Language" in the "Printer Properties" dialog, select: the entry "HTML V1.3".
4. Select "Draft Mode" for rapid storage of the HTML file.

15.8.3.3 PDF (Comfort Panels)

Introduction

The application "PDFSettings.exe" is available on the desktop of the HMI device after installation of the "PDF V1.3" printer driver.

Selecting the storage location

1. Start the application "PDFSettings.exe" on the desktop of your HMI device.
2. Open "Storage Location" in the tab.
3. Under "Primary Location", select "USB-Stick", for example, for the storage location.
4. Select the directory in which the file is to be saved after printing.

Selecting the file name

1. Open the "PDFSettings.exe" application
2. Open the "PDF File Names" tab
3. Select the settings for the names of the PDF file for the print-out.
4. Click on "Apply" to confirm your settings.
5. Select "Default" to retain the default settings for file name assignment.

Define PDF V1.3 as printer on the HMI device

1. Open the Control Panel.
2. Start the "Printer" application.
3. In the "Printer Properties" dialog, select the entry "PDF V1.3" under "Printer Language:".
4. Select "Draft Mode" for rapid storage of the PDF file.

Rule

The table below shows the device-specific differences for printing PDFs:

Mode	KP700 Comfort TP700 Comfort KP900 Comfort TP900 Comfort KP1200 Comfort TP1200 Comfort	KP1500 Comfort TP1500 Comfort TP1900 Comfort TP2200 Comfort
Normal Mode	10 seconds per page	2 seconds per page
Draft Mode	2-3 seconds per page	1 second per page

15.8.4 Installation (Comfort Panels)

Requirements

ProSave is installed on the PC.

Procedure

1. Double click the "PROSAVE-OPT.exe" file.
2. Select "<Setup>" in the "WinZip Self-Extractor" dialog.
3. Select the language for the installation menu.
4. Select "<Continue>" in the "Optional package printer driver setup" dialog.
5. Confirm the next box with "<Install>".
6. Confirm the "Optional package printer driver setup" message with "<Finish>".

Note

To transfer the printer driver to a HMI device, please follow the instructions in the chapter "Transfer of options".

15.8.5 Transferring the Options (Comfort Panels)

Requirements

The HMI device is connected to a PC on which ProSave is installed.

Procedure

1. Restart ProSave on your PC.
2. Select the corresponding HMI device type in the "General" tab.
3. Select the type of connection between the HMI device and the PC.
4. Set the connection parameters.
5. Select the "Options" tab.
6. Select the desired printer driver under "Available options".
7. Set "Transfer" mode on the HMI device.
   If automatic transfer mode is enabled on the HMI device, the device automatically sets "Transfer" mode when a transfer is initiated.
8. Select ">>" on the PC to start the transfer.
9. Follow the ProSave instructions and the instructions which appear on the HMI device.
   During the installation of the printer driver, a status display appears indicating the progress of the operation.

Result

When the installation is complete, the new printer driver is displayed in ProSave in the "Installed options" area. The new printer driver can now be used on the HMI device.

Performance features 16

16.1 General technical specifications

16.1.1 Permitted characters

Introduction

The following table shows the restrictions that must be observed when allocating names and passwords.

Permitted characters

Names / passwords	Restriction
Computer names	The name must start with a letter.
Names of objects	Do not use the characters ? " / \ * < > %
Names and passwords in user view	Do not use the characters ? " / \ § & $ %
Recipe data	Do not use the characters , ;
Names of alarm logs	In the storage location file - RDB, file - CSV (ASCII) and file -TXT (Unicode) not the characters \ / * ? : " < > |
	For the storage location database, use only the characters a-z A-Z 0-9 _ @ # $ The characters _ @ # $ may not be used as the first character of a name
Name of HMI tags	HMI tag names may not start with the @ character
Names of screens	Do not use the special characters ? " / \ * < > Avoid the character . in screens for Runtime Professional:
Tag names in faceplates	Do not use the characters . @
Property names of the faceplates	Do not use any UNICODE characters (for example, Chinese characters) in the names. The name must start with a letter. The name may not contain more than 255 characters.
Names of AuditTrails	Do not use any special characters
Names of customized user data types as library objects (RT Professional)	Do not use the special characters : ? " \\ \ * $ % " ' The name may not contain more than 128 characters.
Names of C scripts and VB scripts as library objects	The name must start with an ASCII letter, followed by an alphanumeric ASCII character or underscore. The name may not contain more than 128 characters.
Names of connections	Do not use spaces or special characters
Name of VB functions	Do not use spaces, special characters or VB keywords

Names / passwords	Restriction
Name of Windows user groups and Windows users when using SIMATIC Logon	Do not use the special characters / \
Communication / OPC: names used	Do not use either spaces or the characters . : ? " ' \ * %

Note

Device dependency - quotes in project name

WinCC Professional projects whose names are in quotes have not been compiled.

When you create an HMI device for WinCC RT Professional, do not use quotation marks within the project name.

16.1.2 Recommended printers

Recommended printers

The current list of printers recommended for use with the HMI devices is available on the Internet at:

Link to the current printer list (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&caller=view&extranet=standard&viewreg=WW&nodeid0=10805558&objaction=csopen)

Note

All HMI devices except for a PC and Panel PC support only one printer at their USB port, even if several ports are available.

See also

Printer list (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&caller=view&extranet=standard&viewreg=WW&nodeid0=10805558&objaction=csopen)

16.1.3 Printing via print server

Introduction

Print servers enable access to network printers. Print jobs are routed to a corresponding printer via the print server.

Note

Printing takes place via the print server for the following HMI devices:

- Mobile Panel 177
- Mobile Panel 277, Mobile Panel 277 IWLAN, Mobile Panel 277F IWLAN
- Mobile Panel 277 IWLAN V2, Mobile Panel 277F IWLAN V2, Mobile Panel 277F IWLAN (RFID Tag)
- KTP Mobile Panels (as of device version V13)
- Comfort Panels
- PC with WinCC Runtime Advanced

Requirements

- The print server deploys the "RAW" mode.

  Note

  For additional information on settings, refer to the relevant print server documentation.

- The device is interconnected via Ethernet with the print server or with a printer featuring an integrated print server.
- External print servers are interconnected with a printer via USB port.
- The IP address and port used are identical in the "Printer Properties" of the HMI device and on the print server.

Procedure

1. Open the "Control Panel" on your HMI device.
2. Select "Printer". The "Printer Properties" dialog box opens.

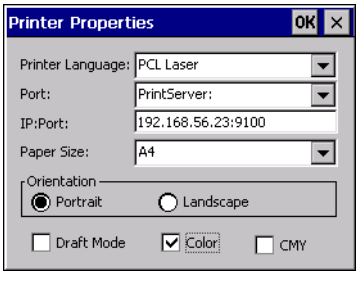

3. Select a printer from the "Printer Language" selection list.

4. Select the "PrintServer" entry from the "Port" selection list.
5. Enter the IP address and the port used for communication with the printer in the "IP:Port" input field.

   Note

   Use ":" as delimiter between the IP address and the port number. For example: "192.168.56.23:**9100**".

6. Select any additional printer settings.

Note

You can also print over the network.

Select the "Port" entry from the "Network:" selection list for this.

In the "Network:" selection box, select the network path, consisting of the computer name and the printer name, for example "\\server12\\printer01".

16.1.4 Memory requirement of recipes

Introduction

The following calculation of memory requirements of recipes is only valid for Windows CE devices.

Calculation of memory requirements

The memory space required by each recipe (in KB) is derived from the sum of D1 + D2 + D3.

Valid is:

- D1 = (number of entries x 5 + M + 8):1024
  Applies to M:
  M = Accumulated length of all tag names = sum of characters in all tag names (UTF8 coded, max. 255 bytes per tag name) used in the entries.
- D2 = [(number of data records x 12) + 4]:1024
- D3 = [number of data records x (data record length + N) + 4]:1024
  Applies to N:

The total length of the name of the corresponding data record in all languages (max. 255 bytes per language) + overhead per data record (1 byte + number of languages * 3 bytes).

D1, D2 and D3 are rounded to the next higher number.

Memory requirements for using arrays

The memory required by each recipe (in KB) is derived from the sum of D1 + D2 + D3.

Valid is:

- D1 = (number of entries x 5 + M + 8):1024
  Each element of the tag array used counts as a single entry.
  Applies to M:
  M = (length of the array tag name + K) x number of array elements
  Applies to K:
  K = 3: 2 to 9 elements in the array
  K = 4: 10 to 99 elements in the array
  K = 5: 100 to 999 elements in the array
  K = 6: 1000 to 9999 elements in the array
  K = 7: 10000 to 12000 elements in the array
- D2 = [(number of data records x 12) + 4] : 1024
- D3 = [number of data records x (data record length + N) + 4] : 1024
  Applies to N:

The total length of the name of the corresponding data record in all languages (max. 255 bytes per language) + overhead per data record (1 byte + number of languages * 3 bytes).

D1, D2 and D3 are rounded to the next higher number.

Note

If you use both tags and arrays in a recipe, you have to add the results of both formulas to calculate the total memory required.

16.1.5 Memory requirements of recipes for Basic Panels

Introduction

The following calculation of the memory requirements for recipes applies to Basic Panels.

Restrictions

The HMI device provides 39 KB of memory space for recipes. This memory space may not be exceeded. The total memory space for recipes is calculated as follows: Total of all recipes + recipe with highest memory requirement.

Each recipe may not exceed a maximum memory space of 19 KB.

Calculation of memory requirements

The memory space requirement of each recipe (in KB) is calculated based on the three addends D1 + D2 + D3.

Valid is::

- D1 = number of data records x M
  Rule for M (size of a data record):
  M = 1 x number of elements of a byte + 2 x number of elements of 2 bytes + 4 x number of elements of 4 bytes + 8 x number of elements of 8 bytes + K
  Rule for K (size of the string elements):
  K = number of string elements x (string length + 1) x 2

- D2 - data record size
  D2 = 4 + number of languages x 8 + number of languages x (4 + 4 x number of data records + (length of the data record name + 1) x 2 x number of data records) + 8 + 8 x number of data records
  Or rewritten:
  D2 = 12 + 8 x number of data records + number of languages x (12 + number of data records x (4 + (length of the data record name +1) x 2))

- D3 - shared memory
  D3 = 14 + number of elements

Note

Arrays and single elements can be calculated as described above.

16.2 Performance features of the devices

16.2.1 Basic Panel (Basic Panels)

Basic Panel

The following table helps you assess whether your project meets the performance features of the HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of tags in the project	250	500	250 (mono) 500 (color)	500	500	500
Number of PowerTags	--	--	--	--	--	--
Number of elements per array	100	100	100	100	100	100
Number of local tags	--	--	--	--	--	--
Number of structures	--	--	--	--	--	--
Number of structure elements	--	--	--	--	--	--

Alarms

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of alarm classes	32	32	32	32	32	32
Number of discrete alarms	200	200	200	200	200	200
Number of analog alarms	15	15	15	15	15	15
Length of an alarm in characters	80	80	80	80	80	80
Number of process values per alarm	8	8	8	8	8	8
Size of the alarm buffer	256	256	256	256	256	256
Number of queued alarm events	64	64	64	64	64	64

Screens

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of screens	50	50	50	50	50	50
Number of fields per screen	30	30	30	30	30	30
Number of tags per screen	30	30	30	30	30	30
Number of complex objects per screen [1)]	5	5	5	5	5	5
Number of array elements per screen [2)]	100	100	100	100	100	100

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

2) Array elements contained in recipes are included in the count.

Recipes

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of recipes	5	5	5	5	5	5
Number of elements per recipe[1)]	20	20	20	20	20	20
User data length in bytes per data record	--	--	--	--	--	--
Number of data records per recipe	20	20	20	20	20	20
Reserved memory for data records in the internal Flash	40 KB	40 KB	40 KB	40 KB	40 KB	40 KB

1) If arrays are used, each array element counts as one recipe element

Logs

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of logs	--	--	--	--	--	--
Number of entries per log (including all log segments) [1)]	--	--	--	--	--	--
Number of log segments	--	--	--	--	--	--
Cyclic trigger for tag logging	--	--	--	--	--	--
Number of tags that can be logged per log	--	--	--	--	--	--

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of trends	25	25	25	25	25	25

Text lists and graphics lists

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of graphics lists	100	100	100	100	100	100
Number of text lists	150	150	150	150	150	150
Number of entries per text or graphics list	30	30	30	30	30	30
Number of graphic objects	500	500	500	500	500	500
Number of text elements	500	500	500	500	500	500

Scripts

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of scripts	--	--	--	--	--	--

Communication

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of connections	4	4	4	4	4	4
Number of connections based on "SIMATIC HMI HTTP"	--	--	--	--	--	--

Help system

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of characters in a help text	500	500	500	500	500	500

Languages

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of runtime languages	5	5	5	5	5	5

Scheduler

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Time-triggered tasks [1]	--	--	--	--	--	--

1) Event-triggered tasks are not relevant for the system limits

User administration

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of user groups	50	50	50	50	50	50
Number of authorizations	32	32	32	32	32	32
Number of users	50	50	50	50	50	50

Project

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Size of the project file "*.srt"	1024 KB	1024 KB	1024 KB	1024 KB	1024 KB	1024 KB

See also

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

16.2.2 Basic Panel 2nd Generation (Basic Panels)

Basic Panel 2nd Generation

The following table helps you assess whether your project meets the performance features of the HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of tags in the project	800	800	800	800
Number of PowerTags	--	--	--	--

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of elements per array	100	100	100	100
Number of local tags	--	--	--	--
Number of structures	--	--	--	--
Number of structure elements	--	--	--	--

Alarms

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of alarm classes	32	32	32	32
Number of discrete alarms	1000	1000	1000	1000
Number of analog alarms	25	25	25	25
Length of an alarm in characters	80	80	80	80
Number of process values per alarm	8	8	8	8
Size of the alarm buffer	256	256	256	256
Number of queued alarm events	64	64	64	64

Screens

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of screens	250	250	250	250
Number of fields per screen	100	100	100	100
Number of tags per screen	100	100	100	100
Number of complex objects per screen [1)]	This information is not relevant for Basic Panels 2nd Generation.			
Number of recipe displays per screen	10	10	10	10
Number of trend displays per screen	8	8	8	8
Number of alarm displays per screen	20	20	20	20
Number of user displays per screen	1	1	1	1
Number of system diagnostic displays per screen	5	5	5	5
Number of system functions per screen	150	150	150	150
Number of multiplex tags per screen	100	100	100	100

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of recipes	50	50	50	50
Number of elements per recipe[1)]	100	100	100	100
User data length in KB per data record	32	32	32	32

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of data records per recipe	100	100	100	100
Reserved memory for data records in the internal Flash	256 KB	256 KB	256 KB	256 KB

1) If arrays are used, each array element counts as one recipe element

Logs

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of data logs	1	1	1	1
Number of alarm logs	1	1	1	1
Number of entries per log (including all log segments) [1]	10000	10000	10000	10000
Number of log segments	400	400	400	400
Number of tags that can be logged per log	10	10	10	10
Cyclic trigger for tag logging	1 s	1 s	1 s	1 s

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of trends	25	25	25	25

Text lists and graphics lists

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of graphics lists	100	100	100	100
Number of text lists	300	300	300	300
Number of entries per text or graphics list	100	100	100	100
Number of graphic objects	1000	1000	1000	1000
Number of text elements	2500	2500	2500	2500

Scripts

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of scripts	--	--	--	--

Communication

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of connections	4	4	4	4
Number of connections based on "SIMATIC HMI HTTP"	--	--	--	--

Help system

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of characters in a help text	500	500	500	500

Languages

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of runtime languages	10	10	10	10

Scheduler

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Time-triggered tasks [1)]	--	--	--	--

1) Event-triggered tasks are not relevant for the system limits

User administration

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of user groups	50	50	50	50
Number of authorizations	32	32	32	32
Number of users	50	50	50	50

Project

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Size of the project file "*.srt"	10 MB	10 MB	10 MB	10 MB

16.2.3 Mobile Panel (Panels, Comfort Panels)

Mobile Panel

The following tables of system limits help you assess whether your project conforms to the system limits of a given HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile TP1000F Mobile
Number of tags in the project	1000	2048	2048	2048	2048	2048	2048	2048
Number of PowerTags	--	--	--	--	--	--	--	--
Number of elements per array	1000	1000	1000	1000	1000	1000	1000	1000
Number of local tags	500	1000	1000	1000	1000	1000	1000	1000
Number of structures	--	999	999	999	999	999	999	999
Number of structure elements	--	400	400	400	400	400	400	400

Alarms

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F MobileKTP700 F Mobile
Number of alarm classes	32	32	32	32	32	32	32	32
Number of discrete alarms	2000	4000	4000	4000	4000	4000	4000	4000
Number of analog alarms	50	200	200	200	200	200	200	200
Length of an alarm in characters	80	80	80	80	80	80	80	80
Number of process values per alarm	8	8	8	8	8	8	8	8

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F MobileKTP700 F Mobile
Size of the alarm buffer	256	512	512	512	512	1024	1024	1024
Number of queued alarm events	64	250	250	250	250	500	500	500

Screens

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F MobileKTP700 F Mobile
Number of screens	500	500	500	500	500	500	500	500
Number of fields per screen	50	200	200	200	200	50	400	400
Number of tags per screen	50	200	200	200	200	50	400	400
Number of complex objects per screen [1)]	5	10	10	10	10	5	20	20

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F MobileKTP700 F Mobile
Number of recipes	100	300	300	300	300	300	300	300
Number of elements per recipe[1)]	200	1000	1000	1000	1000	1000	1000	1000
User data length in bytes per data record	800	4000	4000	4000	4000	262144	262144	262144
Number of data records per recipe	200	500	500	500	500	500	500	500
Reserved memory for data records in the internal Flash	32 KB	64 KB	64 KB	64 KB	64 KB	2 MB	2 MB	2 MB

1) If arrays are used, each array element counts as one recipe element

Logs

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F MobileKTP700F MobileKTP700F Mobile
Number of logs	--	20	20	20	20	50	50	50
Number of entries per log (including all log segments) [1]	--	10000	10000	10000	10000	20000	20000	20000
Number of log segments	--	400	400	400	400	400	400	400
Cyclic trigger for tag logging	--	1 s	1 s	1 s	1 s	1 s	1 s	1 s
Number of tags that can be logged per log	--	2048	2048	2048	2048	2048	2048	2048

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of trends	50	300	300	300	300	300	300	300

Text lists and graphics lists

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of graphics lists	100	400	400	400	400	500	500	500
Number of text lists	300	500	500	500	500	500	500	500

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of entries per text or graphics list	30	256	256	256	256	500	500	500
Number of graphic objects	1000	1000	1000	1000	1000	4000	4000	4000
Number of text elements	2500	10000	10000	10000	10000	40000	40000	40000

Scripts

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of scripts	--	50	50	50	50	100	100	100

Communication

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of connections	4	6	6	6	6	8	8	8
Number of connections based on "SIMATIC HMI HTTP"	4	8	8	8	8	8	8	8
Maximum number of connected Sm@rtClients (including a service client)	2	2	8": 3 10": 2	8": 3 10": 2	8": 3 10": 2	3	3	3

Mobile Wireless

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of zones	--	--	254	254	--	254	254	254
Number of effective ranges	--	--	--	127	--	--	--	--

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of assigned transponders - zones	--	--	255	255	--	--	--	--
Number of assigned transponders - effective ranges	--	--	--	127	--	--	--	--
Number of effective ranges (RFID)	--	--	--	--	127	--	--	--
Number of RFID tags that can be assigned to effective ranges (RFID) in a project	--	--	--	--	127	--	--	--

Help system

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of characters in a help text	320	320	320	320	320	500	500	500

Languages

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of runtime languages	5	16	16	16	16	32	32	32

Scheduler

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Time-triggered tasks [1)]	10	48	48	48	48	48	48	48

1) Event-triggered tasks are not relevant for the system limits

User administration

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Number of user groups	50	50	50	50	50	50	50	50
Number of authorizations	32	32	32	32	32	32	32	32
Number of users	50	50	50	50	50	50	50	50

Project

	Mobile Panel 177	Mobile Panel 277	Mobile Panel 277 IWLAN V2	Mobile Panel 277F IWLAN V2	Mobile Panel 277F IWLAN (RFID Tag)	KTP400F Mobile	KTP700 Mobile / KTP700F Mobile	KTP900 Mobile / KTP900F Mobile
Size of the project file "*.fwc"	2 MB	6 MB	6 MB	6 MB	6 MB	12 MB	12 MB	12 MB

See also

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

16.2.4 Comfort Panel (Panels, Comfort Panels)

Comfort Panel

The following tables of system limits help you assess whether your project conforms to the system limits of a given HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Note

Graphic objects on Comfort Panels

The use of 32-bit graphic objects increases memory requirements for the project file on Comfort Panels.
The use of jpeg graphic objects reduces memory requirements for the project file on Comfort Panels.

Tags

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of tags in the project	1024	2048	4096	4096	4096
Number of PowerTags	--	--	--	--	--
Number of elements per array	1000	1000	1000	1000	1000
Number of local tags	500	1000	2000	2000	2000
Number of structures	999	999	999	999	999
Number of structure elements	400	400	400	400	400

Alarms

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of alarm classes	32	32	32	32	32
Number of discrete alarms	2000	4000	6000	6000	6000
Number of analog alarms	50	200	200	200	200
Length of an alarm in characters	80	80	80	80	80
Number of process values per alarm	8	8	8	8	8
Size of the alarm buffer	256	1024	1024	1024	1024
Number of queued alarm events	64	500	500	500	500

Screens

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of screens	500	500	750	750	750
Number of fields per screen	50	400	600	600	600
Number of tags per screen	50	400	400	400	400
Number of complex objects per screen [1)]	5	20	40	40	40

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of recipes	100	300	500	500	500
Number of elements per recipe[1)]	200	1000	2000	2000	2000
User data length in KB per data record	32	256	512	512	512
Number of data records per recipe	200	500	1000	1000	1000
Reserved memory for data records in the internal Flash	512 KB	2 MB	4 MB	4 MB	4 MB

1) If arrays are used, each array element counts as one recipe element

Logs

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of logs	10	50	50	50	50
Number of entries per log (including all log segments) [1]	10000	20000	50000	50000	50000
Number of log segments	400	400	400	400	400
Cyclic trigger for tag logging	1 s	1 s	1 s	1 s	1 s
Number of tags that can be logged per log	100	2048	2048	2048	2048

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of trends	50	300	400	400	400

Text lists and graphics lists

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of graphics lists	100	500	500	500	500
Number of text lists	300	500	500	500	500
Number of entries per text or graphics list	30	500	500	500	500

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of graphic objects	1000	4000	4000	4000	4000
Number of text elements	2500	40000	40000	40000	40000

Scripts

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of scripts	50	100	200	200	200

Communication

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of connections	4	8	8	8	8
Number of OPC connections including OPC UA	4	8	8	8	8
Number of connections based on "SIMATIC HMI HTTP"	4	8	8	8	8
Maximum number of connected Sm@rtClients (including a service client)	2	3	3	2	1

Help system

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of characters in a help text	500	500	500	500	500

Languages

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of runtime languages	32	32	32	32	32

Scheduler

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Time-triggered tasks [1)]	10	48	48	48	48

1) Event-triggered tasks are not relevant for the system limits

User administration

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Number of user groups	50	50	50	50	50
Number of authorizations	32	32	32	32	32
Number of users	50	50	50	50	50

Project

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200 TP700 Comfort Outdoor TP1500 Comfort Outdoor	TP1500 KP1500	TP1900	TP2200
Size of the project file "*.fwc"	4 MB	12 MB	24 MB	24 MB	24 MB

See also

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

16.2.5 WinCC Runtime Advanced (RT Advanced)

WinCC Runtime Advanced

The following tables of system limits help you assess whether your project conforms to the system limits of a given HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of tags in the project	6144	12288	24576
Number of PowerTags	128 - 4096	128 – 8192	128-16384
Number of elements per array	1600	1600	1600
Number of internal tags	2048	4096	8192
Number of structures	999	999	999
Number of structure elements	400	400	400

Alarms

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of alarm classes	32	32	32
Number of discrete alarms	4000	6000	6000
Number of analog alarms	500	500	500
Length of an alarm in characters	80	80	80
Number of process values per alarm	8	8	8
Size of the alarm buffer	1024	2048	2048
Number of queued alarm events	500	500	500

Screens

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of screens	500	750	750
Number of fields per screen	400	600	600
Number of tags per screen	400	400	400
Number of complex objects per screen [1)]	40	40	40

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of recipes	999	1000	1000
Number of elements per recipe [1]	2000	2000	2000
User data length in KB per data record	256	512	512
Number of data records per recipe	5000	5000	5000
Reserved memory for data records in the internal Flash	--	--	--

1) If arrays are used, each array element counts as one recipe element

Logs

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of logs	100	100	100
Number of entries per log (including all log segments) [1]	500000	500000	500000
Number of log segments	400	400	400
Cyclic trigger for tag logging	1 s	1 s	1 s
Number of tags that can be logged per log	6144	12288	24576

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of trends	800	800	800

Text lists and graphics lists

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of graphics lists	500	500	500
Number of text lists	500	500	500
Number of entries per text or graphics list	3500	3500	3500
Number of graphic objects	2000	4000	4000
Number of text elements	30000	40000	40000

Scripts

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of scripts	200	200	200

Communication

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of connections	8	8	8
Number of OPC connections including OPC UA	8	8	8
Number of connections based on "SIMATIC HMI HTTP"	16	16	16
Maximum number of connected Sm@rtClients (including a service client)	4 [1]	4 [1]	4 [1]

Help system

	KTP400 KP400	TP700, KP700 TP900, KP900 TP1200, KP1200	TP1500 KP1500	TP1900	TP2200
Number of characters in a help text	320 [1] 500 [2]	320 [1] 500 [2]	320 [1] 500 [2]	320 [1] 500 [2]	320 [1] 500 [2]

1) For devices with HMI device version 12 or lower
2) For devices with HMI device version higher than 12

Languages

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of runtime languages	32	32	32

Scheduler

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Time-triggered tasks [1)]	48	48	48

1) Event-triggered tasks are not relevant for the system limits

User administration

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Number of user groups	50	50	50
Number of authorizations	32	32	32
Number of users	100	100	100

Project

	WinCC Runtime Advanced Device version prior to 13	WinCC Runtime Advanced Device version as of 13	WinCC Runtime Advanced device version greater than or equal to 14
Size of the project file "*.fwc"	No limiting	No limiting	No limiting

See also

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

16.2.6 WinCC Runtime Professional (RT Professional)

WinCC Runtime Professional

The following tables of system limits help you assess whether your project conforms to the system limits of a given HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	WinCC Runtime Professional
Number of tags in the project	500,000
Number of PowerTags	Depends on the license
Number of elements per array	2,000

Screens

It is possible that image files larger than 100 MB are not displayed in WinCC Runtime.

	WinCC Runtime Professional
Objects per screen [1)]	No limit [2)]
Layers per screen	32
Screens per project	No limit [2)]
Screen size in pixels	10,000 x 10,000
Screen object nesting	20
Number of colors	24-bit color depth, 8-bit alpha channel for transparency

1) The number and complexity of objects affects performance
2) Limited by system resources.

Alarms

	WinCC Runtime Professional
Configurable alarms per server/single-user station	150,000
Process tags per alarm line	10
User text blocks per alarm line	10

	WinCC Runtime Professional
Alarm classes	256
Alarm priorities	17 (0...16)

Alarms in Runtime

	Maximum
Alarms per alarm log	No limit [1]
Alarms per historical alarm list (short-term)	1,000
Alarms per historical alarm list (long-term)	1,000 [2]
Alarms per alarm view	5,000 [3]

1) Limited by system resources.
2) On single-user station, or server or client per server if "LongTimeArchiveConsistency" has been set to "no". On single-user station, server, or client if "LongTimeArchiveConsistency" has been set to "yes".
3) On single-user station or server, or on client per server.

Note

Message bursts and continuous alarm load may be generated simultaneously on a single-user station or server.

Logs

	WinCC Runtime Professional
Trend view per screen	25
Trends per trend view	80
Values per trend view	A maximum of 134,217,728 bytes can be transferred from the Tag Logging Server to the Trend views. If a process value contains 20 bytes, approximately 6.7 million values can be displayed or exported to a *.csv file.
Table view per screen	25
Columns per table view	12
Values per table view	30,000
Logs per single-user station/server	100
Logging tags per single-user station/server [1]	80,000

1) Depends on the Logging PowerPack used for logging tags. The basic version contains 500 logging tags.

Note

Screen selection times may be extremely long in the event of multiple maximum values.

Recipes

	WinCC Runtime Professional
Number of recipes	No limit
Number of recipe elements [1]	500
Number of recipe data records	10,000
Number of views	No limit

1) Maximum 1,000,000 recipe elements in total

Example of a permissible configuration limit in Runtime

You reach the maximum number of 1,000,000 recipe elements, for example, with the following configurations:

- 10,000 recipe data records each with 100 recipe elements
- 2,000 recipe data records each with 500 recipe elements

Assumption: Each recipe element is connected to a tag.

Keep in mind that such configuration limits are also influenced by the hardware used and by other configurations.

Reports

	WinCC Runtime Professional
Configurable reports	No limit [1]
Report lines per detail view	66
Tags per report [2]	300

1) Limited by system resources.
2) The number of tags per report depends on the performance of process communication.

Reports in runtime

	Maximum
Alarm sequence reports running simultaneously per server/client	1
Alarm log reports running simultaneously	3

User management

	Maximum
Number of users	200

Communication

	S7-1200	S7-1511	S7-1513	S7-1516	S7-1518
Maximum number of WinCC system	2	20	30	41	62
ISO-on-TCP connections	-	64, 4 of which are reserved for ES	96, 4 of which are reserved for ES	128, 4 of which are reserved for ES	192, 4 of which are reserved for ES
Maximum number of connected PLCs per Runtime	64				
Maximum number of connected PLCs with Softnet via a network adapter	64				

- Note that a maximum of 200 tags per controller should be configured for communication with S7-1200 PLCs.
- If several HMI devices access a PLC, this limit applies for all HMI devices.
- You can learn how many HMI devices can simultaneously access a SIMATIC S7-1200 in the manual for the PLC.
  Entry ID: 36932465 (https://support.industry.siemens.com/cs/document/36932465/simatic-s7-s7-1200-automatisierungssystem?dti=0&lc=de-WW)

Relation between HMI connection and connection resource

An HMI connection to WinCC RT Professional requires up to three connection resources.

Note

Different limits for connections over PROFIBUS and PROFINET

Note that connections over PROFIBUS have other limits regarding relations between the HMI connections and the connection resources than PROFINET.

For communication over PROFIBUS and SIMATIC NET you must also take into account the limits of the SIMATIC NET components (CPs).

You can find additional information about SIMATIC NET components in the respective SIMATIC NET manual: "PC software SIMATIC NET PC".

Configurations - configuration limits in the multi-station system

	WinCC Runtime Professional
WinCC clients in a system	32 [1) 2)]
Web clients in a system	150 [2) 3)]

1) The number of clients is reduced to four if the server also used as operator station.
2) Mixed configuration: 32 clients + 3 Web clients
3) Mixed configuration: 50 Web clients + 1 WinCC client

16.2.7 Orientation help for performance in Runtime (WinCC Runtime Professional)

WinCC Runtime Professional

The following tables help you to assess operational performance of your project in Runtime.

Note

The values specified greatly depend on the hardware used.

Screens in Runtime

Screen change from an empty screen to...	Time in seconds
Screen with standard objects (100 objects)	1
Screen with 2480 I/O fields (8 internal tags)	2
Screen with 1000 I/O fields (1000 internal tags)	1
10 MB screen (bitmap)	1
Alarm view	2
Table view with 4 columns, each with 120 values [1)]	1

1) The values specified are valid for data from "Tag Logging Fast".

Alarms in Runtime

	Maximum
Continuous alarm load without losses (single-user station/server)	10/sec
Message burst (single-user station/server)	2000/10 seconds at intervals of 5 minutes [1)]

1) Alarms are possibly lost if the interval before the next message burst is less than five minutes.

Note

Message bursts and continuous alarm load may be generated simultaneously on a single-user station or server.

Logs in runtime

	Maximum
Logging in database for server/single-user station (Tag Logging Fast)	5000 values/second [1]
Logging in database for server/single-user station (Tag Logging Slow)	1000 values/second [1] [2]
Number of values printed in a report	The number of values printed depends on the number of values displayed in the trend view.

1) The values specified are valid for logging data without signature.
2) With Tag Logging Slow, expect longer screen selection times for a given amount of data than for Tag Logging Fast.

Recipes in Runtime - boundary conditions

Hardware used	Software used	Configuration
Intel® Core™ i3-6100U 2.30 GHz 4 GB RAM No hardware connection	Windows 7 WinCC V14.0 SP1	One WinCC tag per field 300,000 entries each: - 10 fields with 30,000 data records - 500 fields with 600 data records

Recipes in Runtime - measured values

	10 fields	500 fields
Screen change to a screen with integrated recipe view. Result of measurement depends on the fill level of the control: It can take up to 15 seconds for complete display upon initial loading or if considerable configuration changes are made in the user log.	1 second	5 seconds
Read data record: Read the value to the corresponding tag after the button in the recipe view was clicked.	1 - 2 seconds [1]	n seconds [2]
Write data record: Write the value to the corresponding tag and display of the tag in the I/O field after the button in the recipe view was clicked.	1 - 3 seconds [1]	n seconds [2]
Focus change from the first to the last data record.	1 - 2 seconds	1 - 2 seconds

1) 10 fields with a total of 10 tags.
2) 500 fields with a total of 500 tags.

Reports in runtime

	Maximum
Alarm sequence reports running simultaneously per server/client	1
Alarm log reports running simultaneously	3

16.2.8 Orientation help for communication in Runtime (WinCC Runtime Professional)

WinCC Runtime Professional

Install the SIMATIC NET software from the supplied DVD. To use WinCC Runtime Professional with a S7-1200 / S7-1500 controller, you must not exceed the specific system limitations for the corresponding S7 controller. The following explains the system limits for tags, whose values are reported by an S7-1200/S7-1500. The "WinCC Channel Diagnosis" tool is used to make the determination.

Note

Refer to the parameter tables (https://support.industry.siemens.com/cs/ww/de/view/98699910/en) for the specific system limits of your S7-1200 / S7-1500 controller .

Attributes

Attribute	Description
Plc Attributes (max)	This value refers to the maximum number of tags that can be read cyclically. By default, the communication load of an S7 controller is not configured under 30% (default value: 50%).
Minimum Plc Attributes (free)	The "Minimum Plc Attributes (free)" value is based on empirical values. The value applies to the steady state with no screen change. The minimum number „Minimum Plc Attributes (free)" "may be temporarily lower. If the number approaches 0, cyclic reading with low cycle times begins, briefly slowing down communication with the S7 controller.
Number WinCC Runtime Professional	The "Number WinCC Runtime Professional" indicates the maximum number of WinCC Runtime Professional systems which can be connected to a S7 controller. The number includes a WinCC ES that is connected to the S7 controller.

Note

"Plc Attributes" can be used by one or more WinCC Runtime systems. The number of concurrent tags of all connected WinCC Runtime systems for each S7 controller therefore needs to be added up. You can learn how to determine the number of concurrent tags in the section "Critical factors of concurrent tags".

Example

Note

To ensure stable communication with the S7 controller, you should not exceed the "Number of concurrent tags per S7".

You can read the values for the "Plc Attributes (max)", "Minimum Plc Attributes (free)" columns of the parameter table from the WinCC Channel Diagnosis Tool under "CCS7PLUSCHANNEL" and the corresponding connection. The number of concurrent tags per S7 controller can be determined from the "Plc Tag Subscriptions" counter: The number of registered tags per cycle time is displayed. In the following example, the number of registered tags is 2003 (500 ms: 103 tags, 1 s: 400 tags, 2 s: 500 tags, 5 s: 1000 tags).

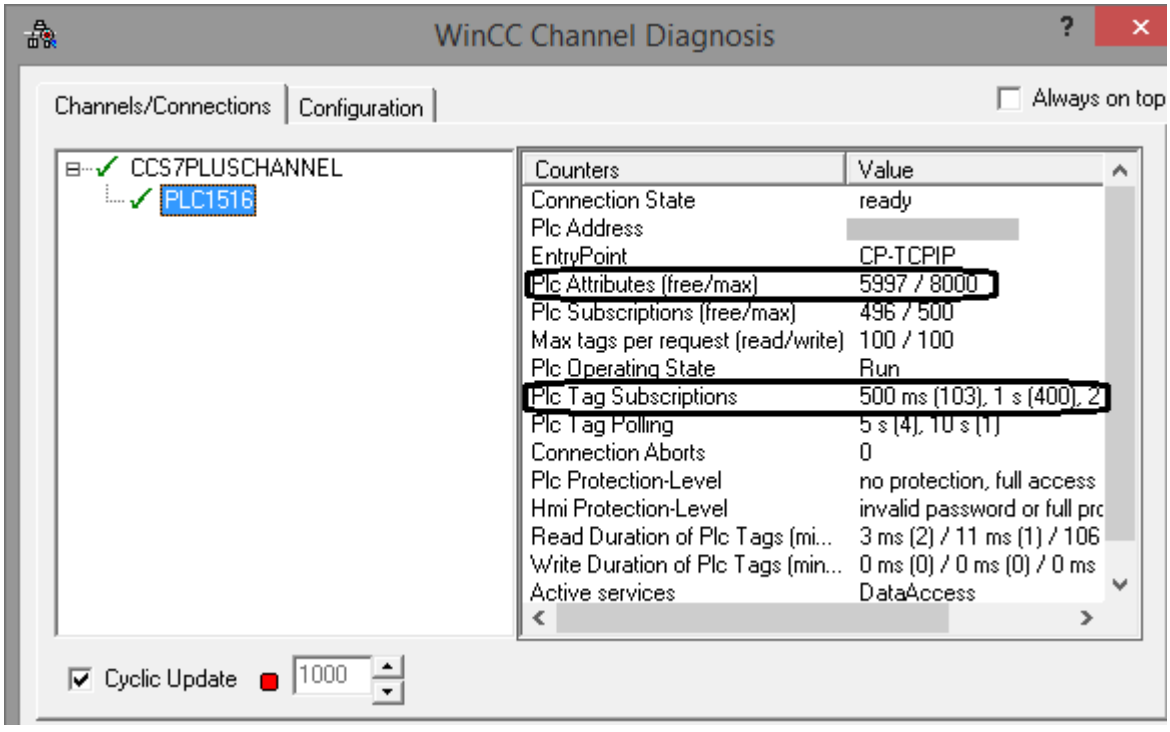

Critical factors of concurrent tags

The number of concurrent tags can be obtained from the following table. Cycle times less than 5 seconds are relevant here. Generally, only the tags which are detected by an S7 controller are relevant. In the example above, these are the tags of the "PLC1516" connection.

Functionality	Concurrent tags
Archiving	Sum of all tags per acquisition cycle.
HMI alarms	Number tags for discrete alarms + number tags for analog alarms + number associated values from tags per alarm.
Scheduler	Sum of all unused tags per cyclic event. With SmartTags, only the Smart Tags with cache usage are relevant. Number of trigger tags used.
Scripts	This affects cyclic scripts, specifically the functions within that use a cache. Information about using the cache is provided in the documentation. Examples: • HmiRuntime.Tags().Read 0. • SmartTags when using the cache. All affected tags are included in the count.
Screen	Number of tags directly related to object properties. Number tags used directly in controls with cycles (for example, gauge control, trend views, table views). Number of tags used in cyclic animations, C scripts, VB scripts. Number of trigger tags used.
OPC server (DA / UA)	Sum of all cyclically reported items.
Control Development	Sum of all cyclically read tags.

See also

S7-1200 / S7-1500 parameter tables (https://support.industry.siemens.com/cs/ww/de/view/98699910/en)

Options

17

17.1 WinCC Engineering V15.1 - Options

WinCC options

You can find a detailed description of WinCC options in the "WinCC Engineering V15.1 – Options" manual.

Migrating to WinCC in the TIA Portal 18

18.1 Overview

Overview of the section "Migration to WinCC in the TIA Portal"

SIMATIC WinCC offers a number of functional changes in the TIA Portal. Some functions differ from the functions that you know from familiar environments such as WinCC V7 or WinCC flexible.

This document provides an overview of the special functions and procedures in SIMATIC WinCC in the TIA Portal.

These functions and procedures are fundamentally different from the WinCC V7 and WinCC flexible version, or have a different name.

18.2 WinCC V7 (RT Professional)

18.2.1 Actions (RT Professional)

Actions in WinCC V7

WinCC supports the use of actions for dynamic response of the processes in your WinCC project. These actions are written in ANSI-C code. Actions are started by a trigger, i.e. a triggering event.

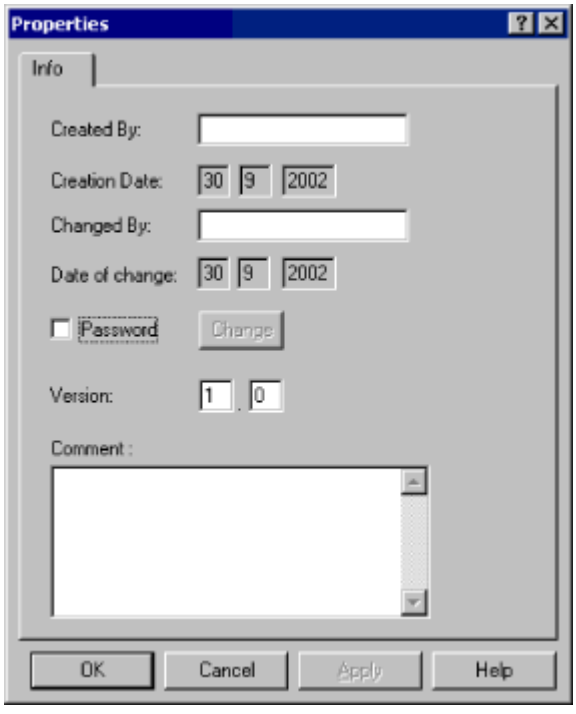

Actions can be used to run the following background tasks independent of the screen:

- Daily printout of a report
- Tag monitoring
- Calculations

How do I configure tasks with WinCC in the TIA Portal?

To configure tasks in the TIA Portal, the WinCC Scheduler is used.

You can use the Scheduler to configure tasks to run independent of the screen in the background. Actions are mapped to tasks of the function list type.

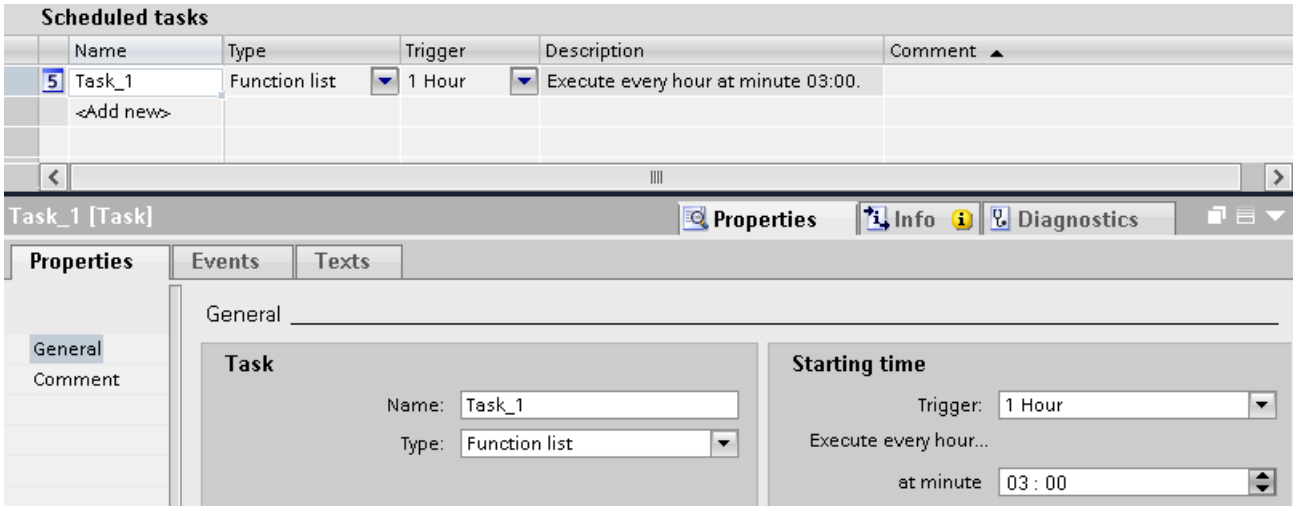

At each task, you configure exactly one function list which is processed line by line. Alternatively, you configure a C or VB script. To start the task, you define a trigger.

For more detailed information, see:

Work area of the "Scheduler" editor (Page 1794)

Example: Reading the current user name (Page 1812)

See also

Work area of the "Scheduler" editor (Page 1794)

Example: Reading the current user name (Page 1812)

18.2.2 User objects (RT Professional)

User objects in WinCC V7

Objects of the object pallet can be combined to form a group or user object based on a multiple selection. You can also insert a group or combined object into the project library.

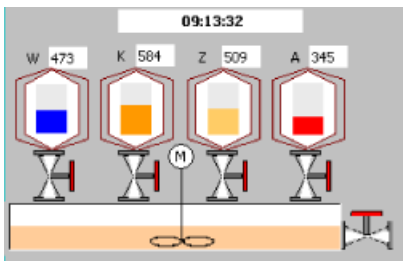

In contrast to a group, a user object enables the individual customization of properties and events that can be displayed and edited in the "Object properties" window. A user object can be edited in Graphics Designer, similar to an individual object of the object pallet.

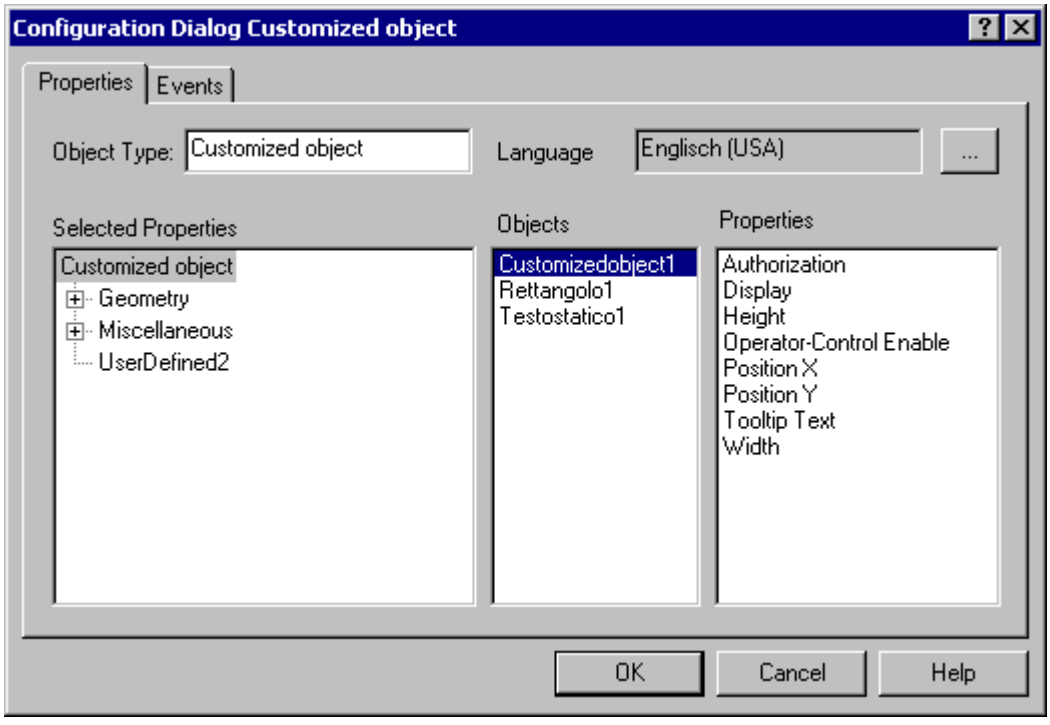

How do I configure faceplates with WinCC in the TIA Portal?

Faceplates are used to create individually configured display and operating objects in WinCC. In contrast to user objects, faceplates can be modified centrally.

WinCC provides the "Faceplates" Editor within the project library for creating faceplates and editing their properties.

Faceplates are based on a type-instance model to support the centralized editing. You create central properties for a faceplate in the types. The instances represent local places of use for the types.

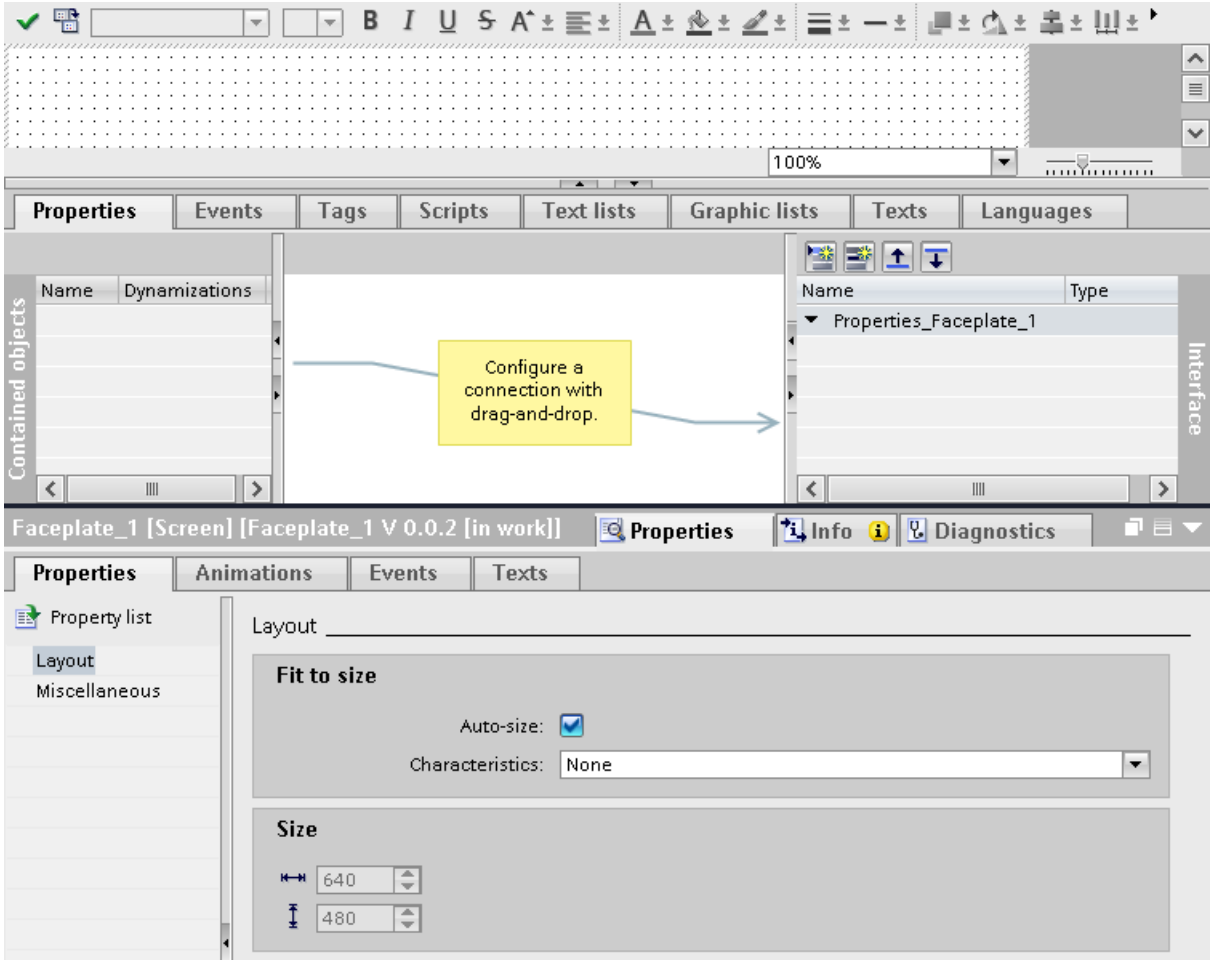

For more detailed information, see:

Basics on faceplates (Page 265)

Example: Configuring a faceplate (Page 294)

See also

Basics on faceplates (Page 265)

Example: Configuring a faceplate (Page 294)

18.2.3 User administration (RT Professional)

User administration in WinCC V7

The "User Administrator" editor is used to set up and manage user administration.

The user administrator enables the setup and management of access rights for WinCC functions and WinCC users.

In the user administration, you can assign functions such as logon timeout, or log on using a key switch / smart card to the users.

The "User Administrator" editor is used for engineering and in runtime.

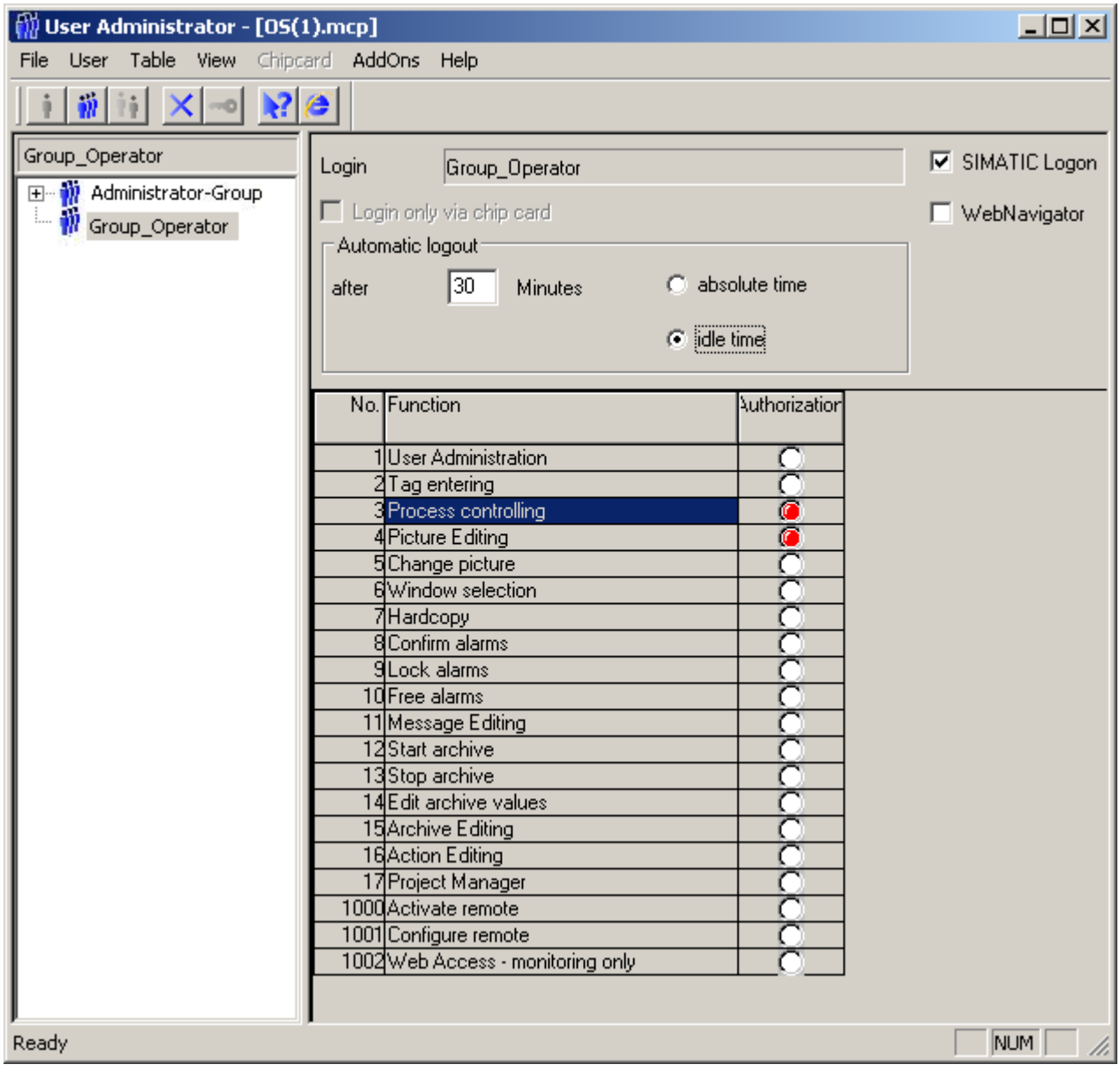

How do I configure user administration with WinCC in the TIA Portal?

- User administration in the engineering system
  You manage users, user groups and authorizations centrally in the WinCC user administration. You transfer users and user groups along with the project to the HMI device. Users are always assigned to a group and therefore inherit the group authorizations.

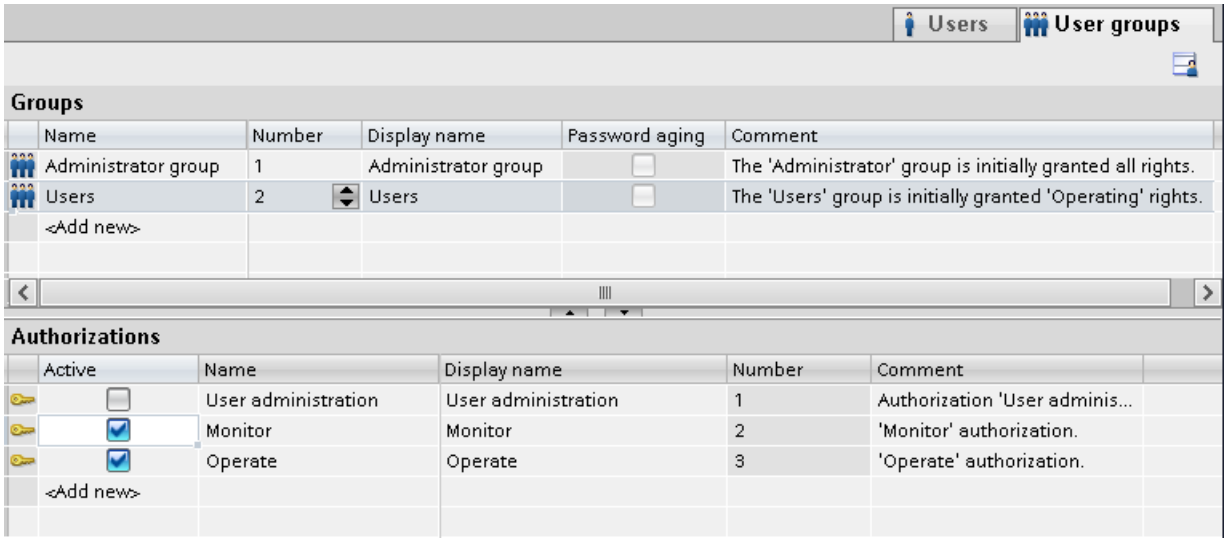

- User administration in runtime
  You can employ a user view in a screen to manage users in runtime as well. You manage users and passwords directly in the user view on the HMI device.

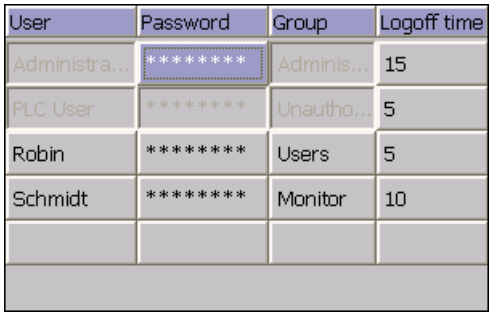

For more detailed information, see:

Form of the user administration (Page 1310)

User view (Page 1342)

See also

Form of the user administration (Page 1310)

User view (Page 1342)

18.2.4 Direct connection (RT Professional)

Direct connections in WinCC V7

A direct connection is used to respond to events. Whenever this event occurs in runtime, the value of a source element is applied to a target element.

Example of the "Mouse click" event:

- The fill pattern of the "Rectangle1" object is synchronized with the fill pattern of the "Circle1" object.
- A screen change is executed.

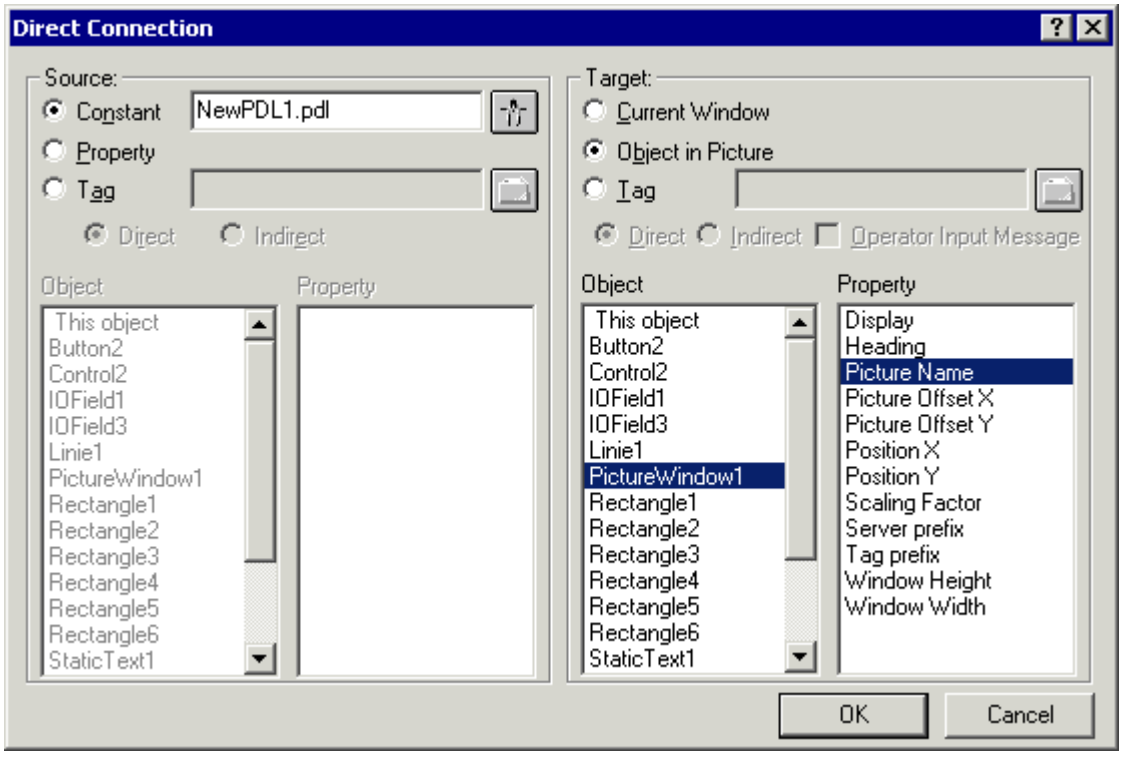

How do I configure system functions with WinCC in the TIA Portal?

In WinCC, direct connections are mapped to system functions.

You use system functions in a function list or in a user-defined function. When the event occurs, execute the configured system functions using the function list.

When configuring a function list, select the system functions from a selection list that is sorted by categories.

For the implementation of the direct connection, specific functions are integrated, such as "SetPropertyByProperty".

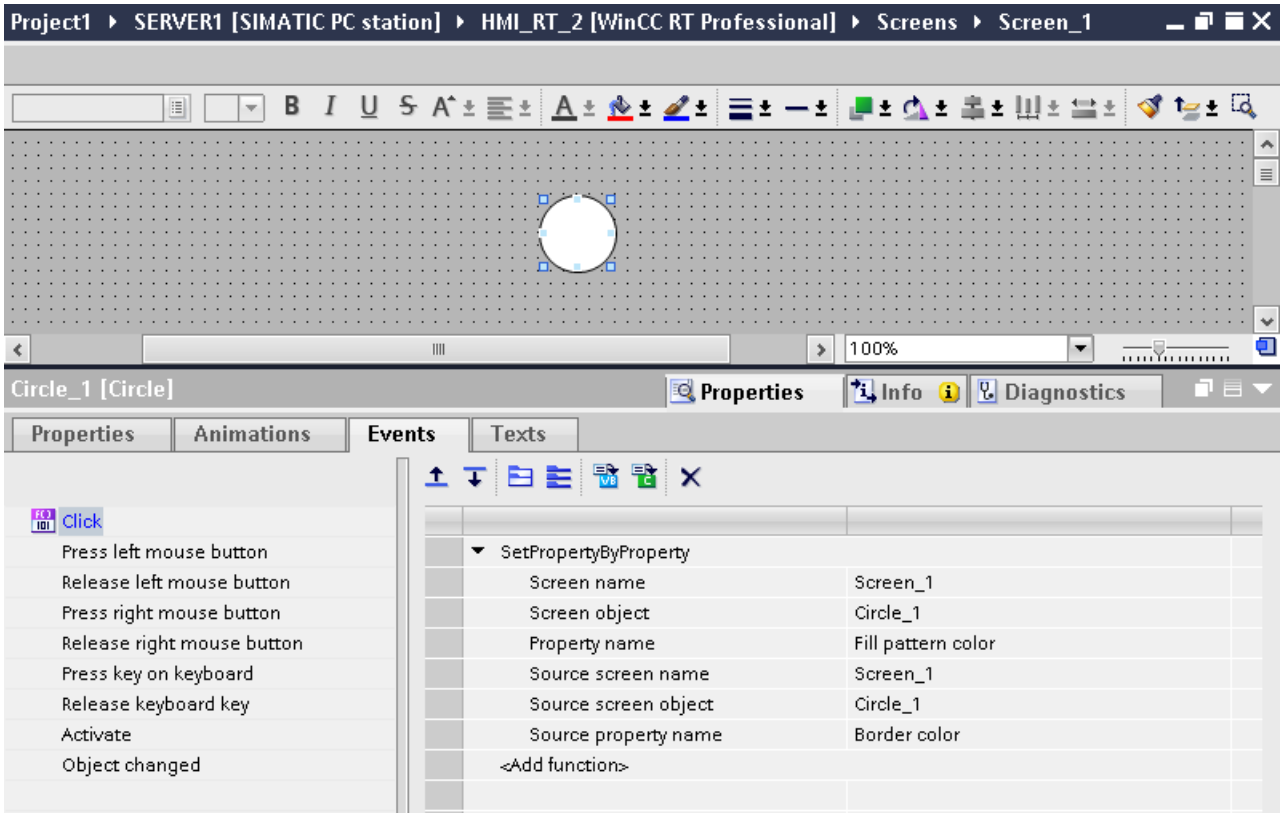

For more detailed information, see:

System functions (Page 1373)

Configuring a function list (Page 1380)

See also

System functions (Page 1373)

Configuring a function list (Page 1380)

18.2.5 Dynamic Wizard (RT Professional)

Dynamic Wizard in WinCC V7

The Dynamic Wizard can be used to set up the dynamic response of an object by means of C-script actions. When a wizard runs, prepared C-script actions such as set bit/reset bit and trigger events are defined and saved in the object properties.

You can always edit the C-script actions in the Events tab of the object properties.

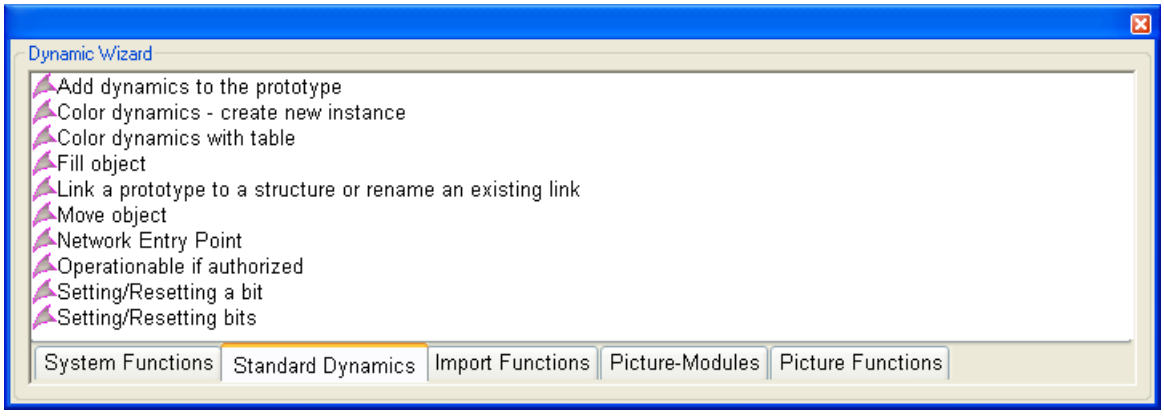

How do I configure system functions with WinCC in the TIA Portal?

In WinCC, objects are provided with dynamic properties, for example, using system functions.

You use system functions in a function list or in a user-defined function. When the event occurs, execute the configured system functions using the function list.

When configuring a function list, select the system functions from a selection list that is sorted by categories.

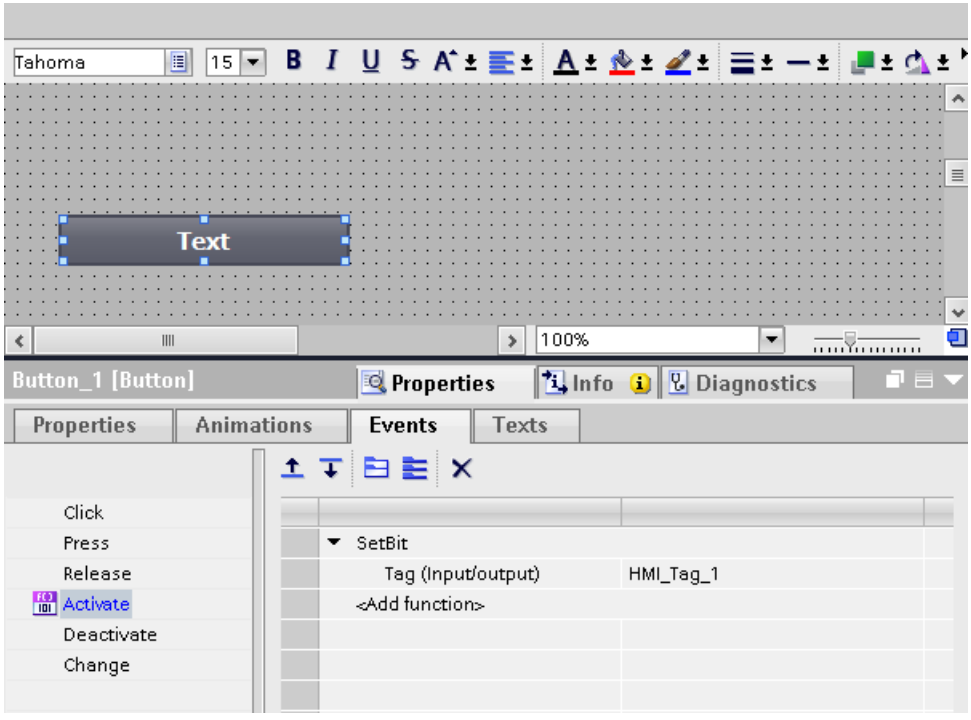

For more detailed information, see:

System functions (Page 1373)

Configuring a function list (Page 1380)

See also

System functions (Page 1373)

Configuring a function list (Page 1380)

18.2.6 Dynamics dialog (RT Professional)

Dynamic dialog in WinCC V7

The Dynamic dialog is used to configure dynamic object properties. In the Dynamic dialog, you formulate an expression by using tags, functions and arithmetic operators.

The value of the expression, the status, as well as the quality code of the tags used in the expression are used to generate the value of the object property in runtime.

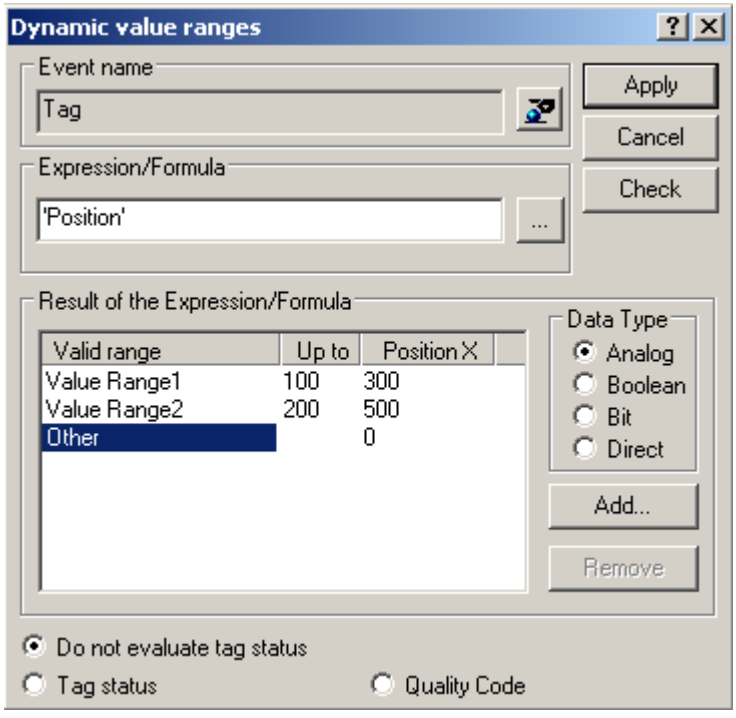

The Dynamic dialog is used, for example, for the following purposes:

- Mapping the range of values of a tag to colors.
- Monitoring of individual tag bits and mapping the bit value to colors or texts.
- Monitoring of a Boolean tag and mapping the tag value to colors or texts.
- Monitoring the status of a tag.
- Monitoring the quality code of a tag.

How do I configure animations with WinCC in the TIA Portal?

Dynamic object properties can be realized, for example, using animations that you configure in the object property list, "Properties > Animations".

These animations dynamically change the background color of an object, for example.

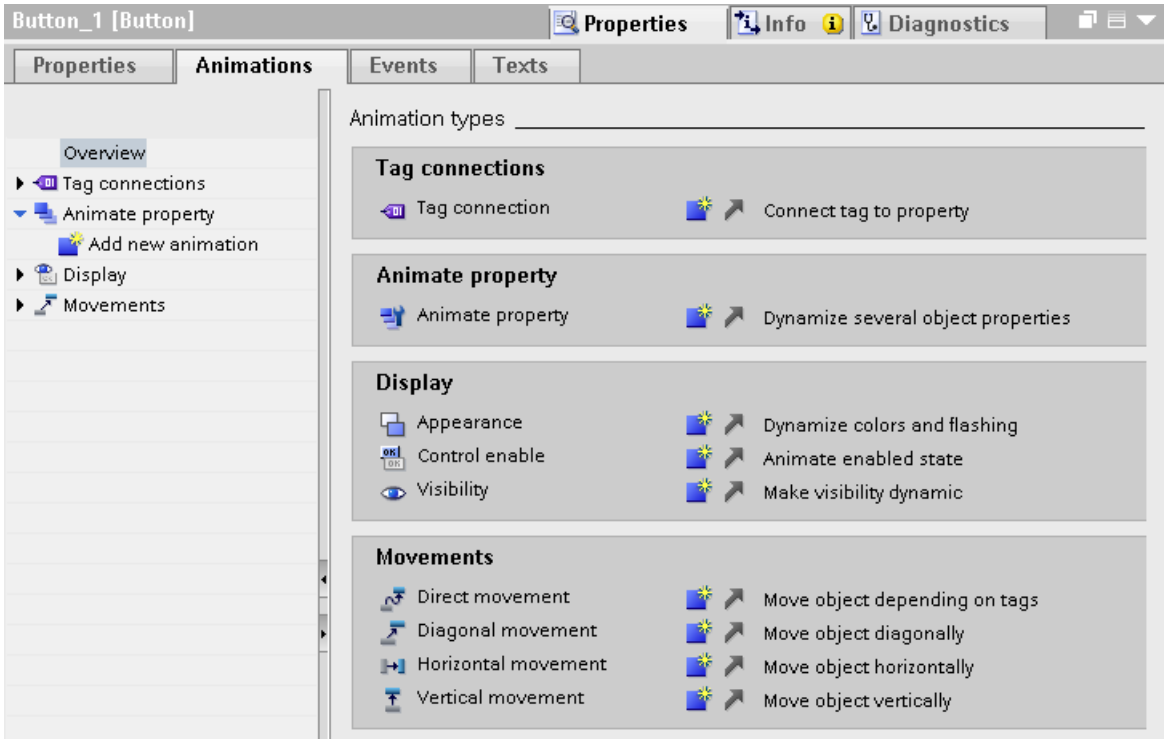

The function "Animate property" is of particular help to you when configuring dynamic properties for the dynamic dialog.

Simple animations are also possible via a tag connection at display and operating objects.

For more detailed information, see:

Basics on dynamization in the property list (Page 232)

Configuring a new animation (Page 212)

See also

Basics on dynamization in the property list (Page 232)

Configuring a new animation (Page 212)

18.2.7 Working efficiently (RT Professional)

Mass data processing in WinCC V7

The WinCC Configuration Tool is an Excel add-in. The tool enables the simple configuration of mass data, such as tags.

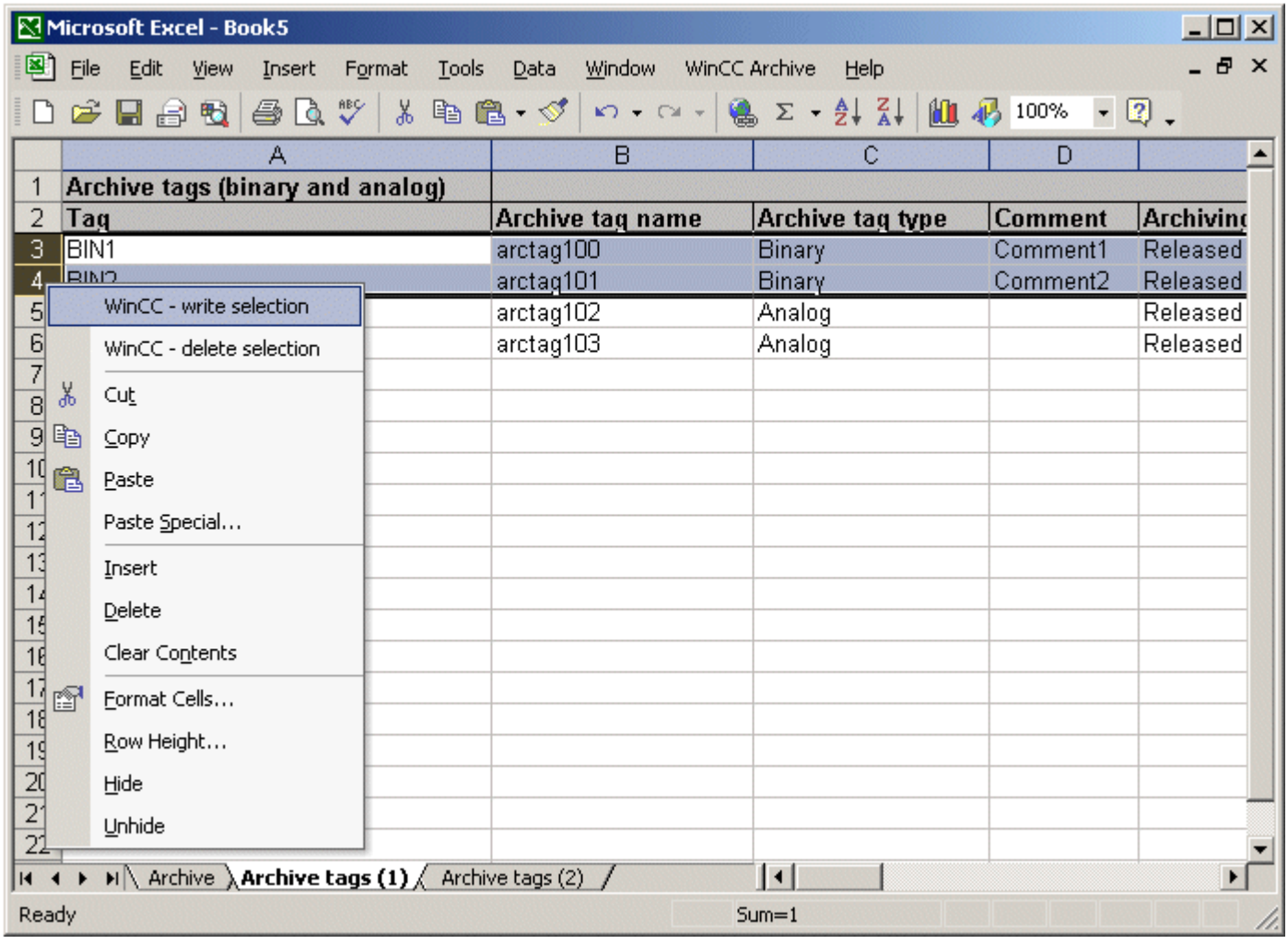

How do I work efficiently with WinCC in the TIA Portal?

WinCC enables efficient processing of process data in table editors. WinCC provides the following corresponding functions:

- Importing and exporting project data

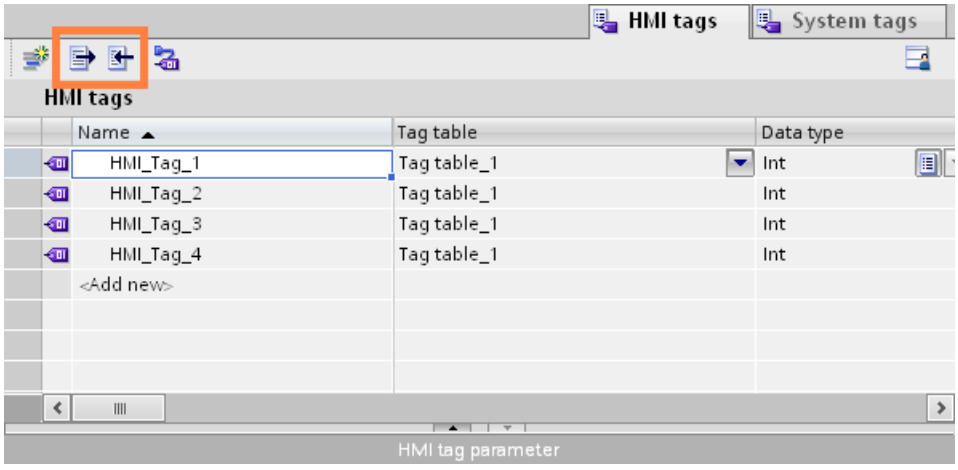

- Multiple selection
- Global changes to a property

You export or import the following project data:

- Recipe data records
- Alarms
- Tags
- Text lists
- Project texts for translations

Exporting and importing reduces the workload.

Instead of creating new data records, use the data you have already created in previous projects.

For more detailed information, see:

Data exchange between projects and external applications (Page 1963)

See also

Data exchange between projects and external applications (Page 1963)

18.2.8 Structures and structure tags (RT Professional)

Structure tags in WinCC V7

Structures are used to enable grouping of a larger number of tags and tag types that form a logical unit.

In WinCC V7, you also create structures which you can address by their names at the tag.

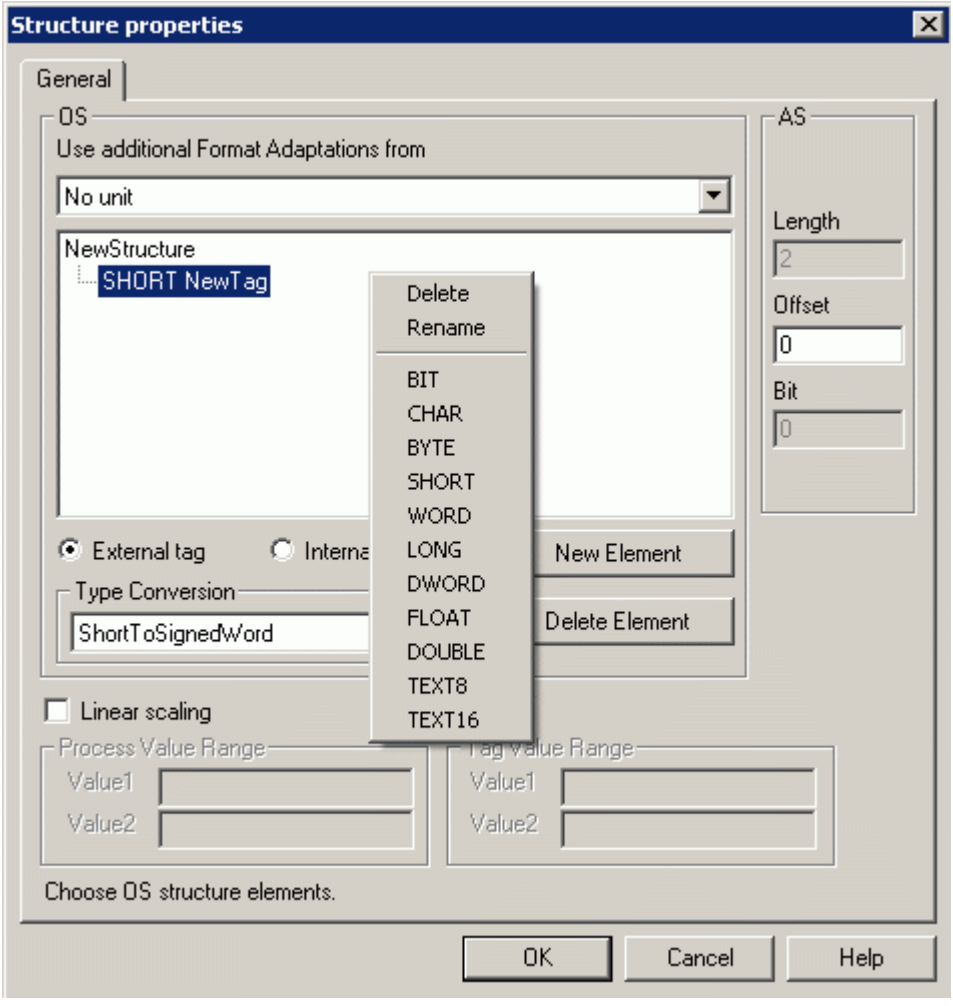

How do I configure user data types with WinCC in the TIA Portal?

In WinCC, user data types enable you to group different tags that form a logical unit.

You create user data types as a type and use instances of this type in the project.

User data types are project-specific data and are available for all HMI devices of the project.

You create user data types and user data type elements in the project library. User data types must be enabled before they can be used in the project.

The configured properties of a user data type are used in the instances of the user data type. If needed, you can change individual properties directly at the point of use, for example, at a tag. Changes to a property at the tag do not affect the user data type created.

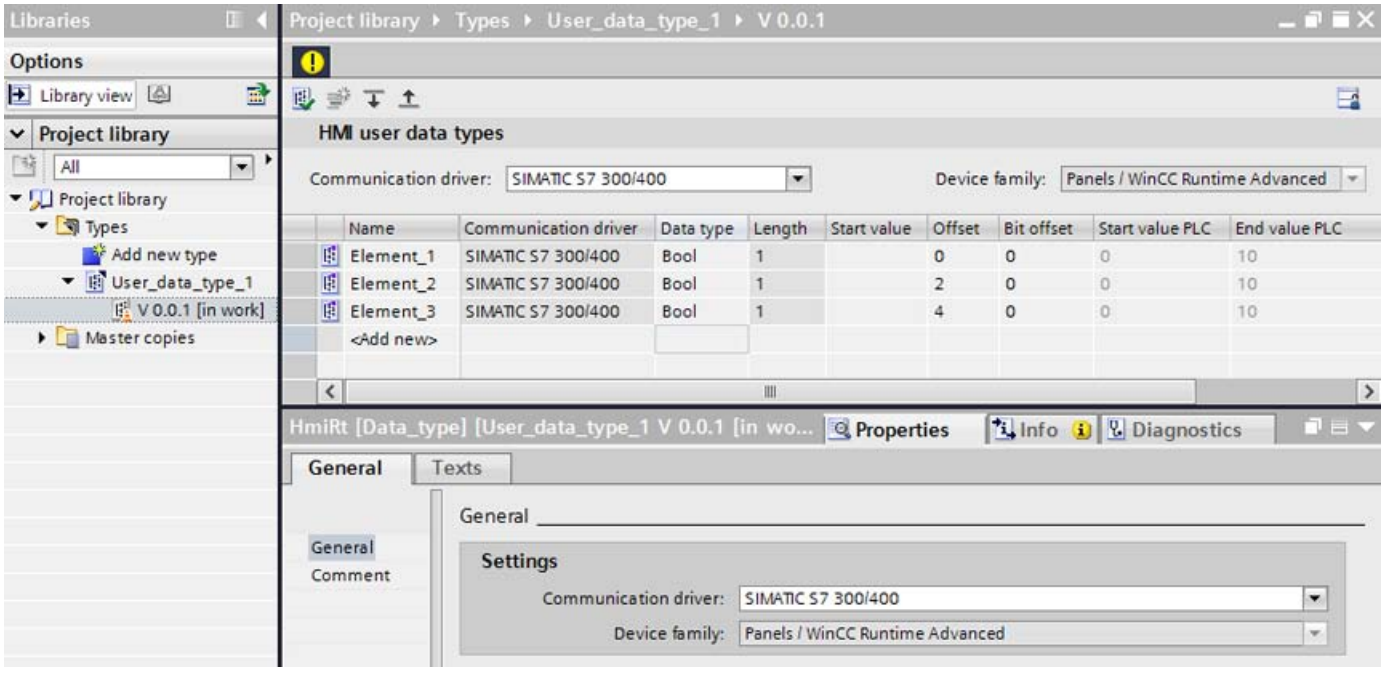

Checking tags

In order to determine whether a tag with the specified name exists, a check takes place at the point of use.

If the tag does not exist, the point of use is marked. The compiler does not output a warning.

If the tag name at the point of use begins with ".", no check takes place in the engineering system.

In this case, it is assumed that the name is formed for Runtime Professional from a tag prefix and the string at the point of use and therefore does not exist in the engineering system.

For user data types, the name of the instance of the user data type is typically used for the tag prefix. You use the name of the relevant user data type elements preceded by a period at the points of use in the screen.

For more detailed information, see:

Basics on user data types (Page 585)

See also

Basics on user data types (Page 585)

18.3 WinCC flexible (RT Advanced, Basic Panels, Panels, Comfort Panels)

18.3.1 Libraries (RT Advanced, Basic Panels, Panels, Comfort Panels)

Libraries in WinCC flexible

Libraries are a collection of pre-configured screen objects. They expand the number of available screen objects and increase engineering efficiency, because library objects are always available for reuse; there is no need to reconfigure them.

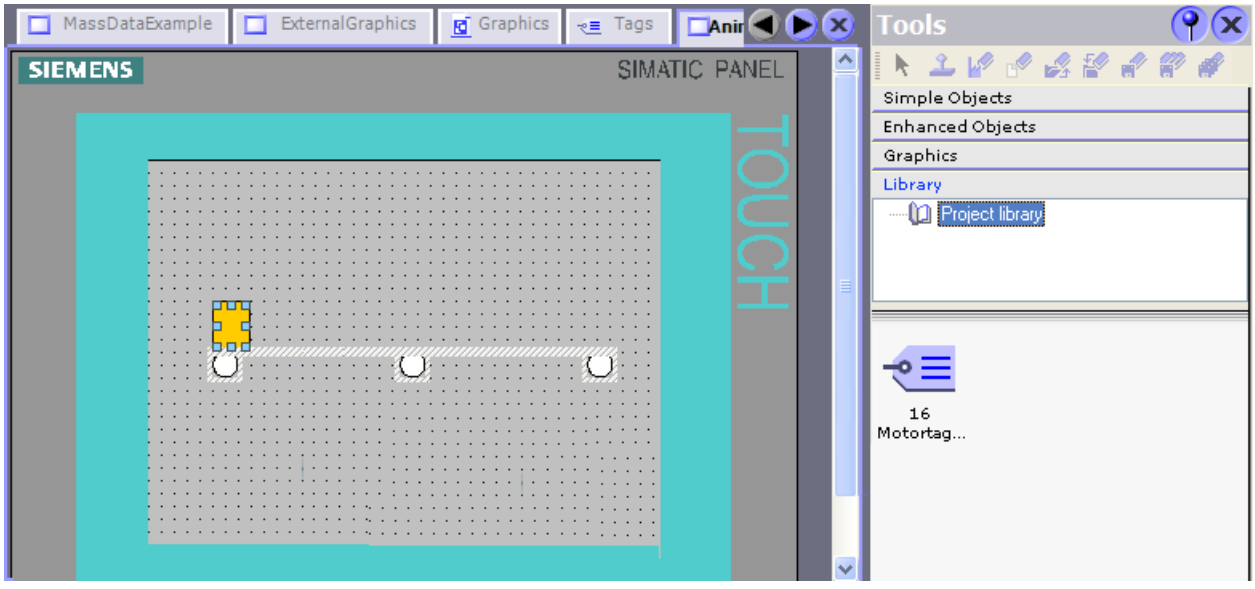

WinCC flexible enables you to create two library types:

- Project library
- Global library

A library can contain all the WinCC flexible objects, such as screens, tags, graphic objects, or alarms.

How do I configure libraries with WinCC in the TIA Portal?

In WinCC, you also configure the "Project library" and the "Global library".

You can no longer store any system functions in libraries, as was the case in WinCC flexible.

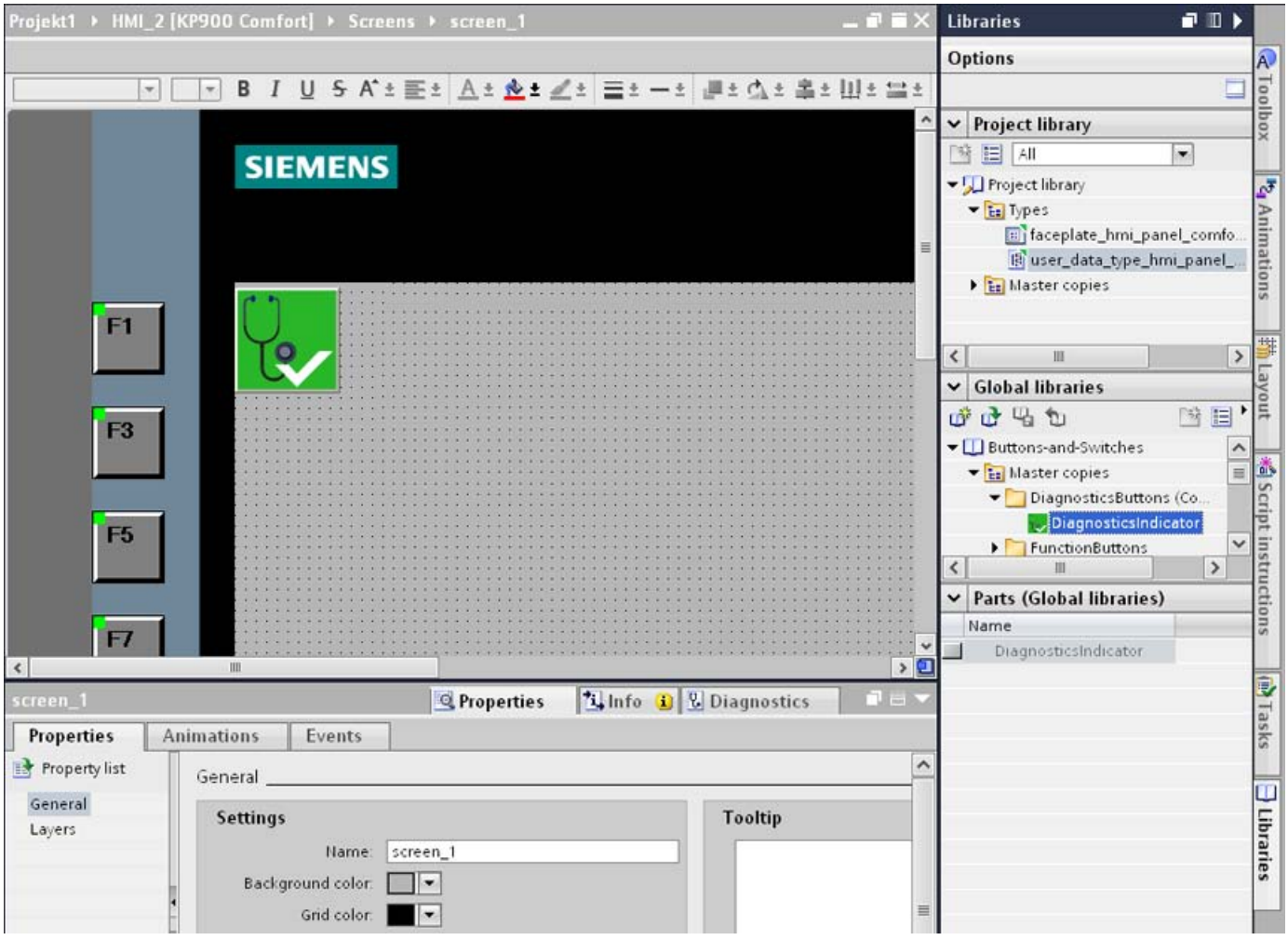

Both the "Project library" and the "Global library" contain the two folders, "Copy templates" and "Types". You can create or use the library objects as a copy template, or as a type.

- Copy templates
  Use copy templates to create independent copies of the library object.
- Types
  Create instances of objects of the "Types" folder and use the instances in your project. The instances are bound to their respective type. Changes to an instance also change all other instances. Types are marked by a green triangle in the "Libraries" task card.
- Managing the library objects
  You can only copy and move library objects within the same library.

For more detailed information, see:

Libraries in WinCC (Page 1939)

See also

Libraries in WinCC (Page 1939)

18.3.2 Screens and templates (RT Advanced, Basic Panels, Panels, Comfort Panels)

Screens and templates in WinCC flexible

In WinCC flexible, you create screens that an operator can use to control and monitor machines and plants. When you create your screens, the object templates provided support you in visualizing your plant, displaying processes and defining process values.

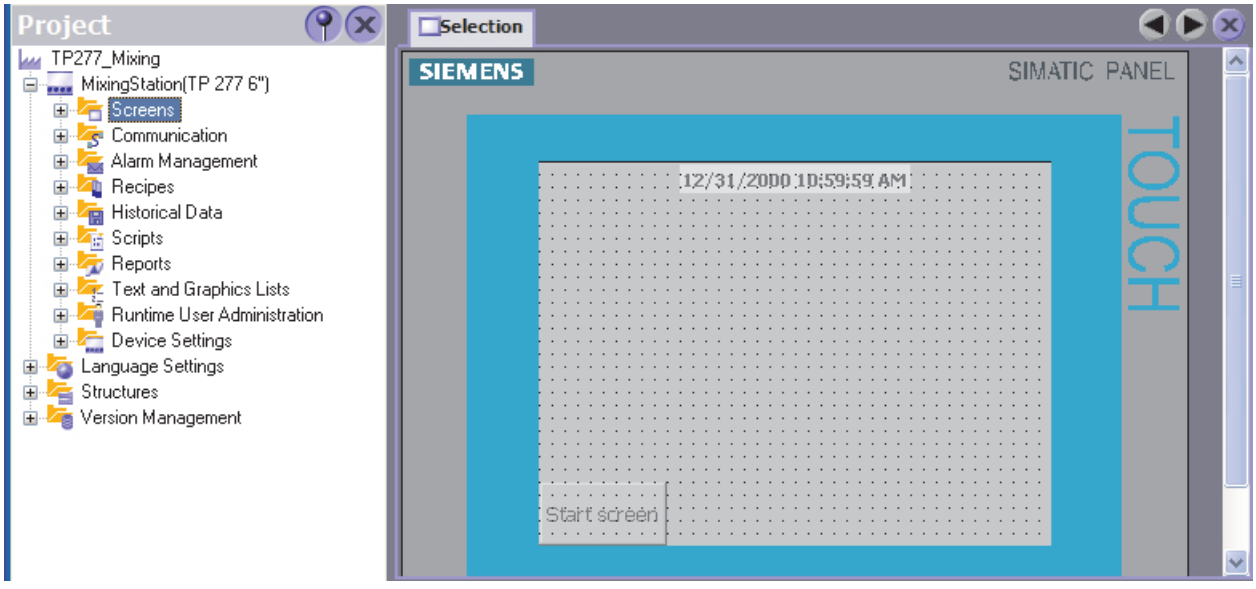

The project has a template for every HMI device. You can centrally configure the function keys and objects for your project in these templates.

Every screen based on this template will contain the function keys and objects that you configured in the template. Changes to an object or of a function key assignment in the template are applied to the object in all the screens, which are based on this template.

How do I configure screens and templates with WinCC in the TIA Portal?

In WinCC, you also configure "Templates" and a "Global screen" along with the "Screens".

You determine functions and objects in the template which then apply to all screens based on this template. You can create multiple templates in WinCC.

In the "Global screen", define the elements which are independent of the template used for all screens of an HMI device. The "Alarm window" and "Alarm indicator" objects are available for use as global objects. For HMI devices with function keys, configure the function keys in the "Global Screen" editor.

You can also configure a "System Diagnostics Window" in the global screen of Comfort Panels.

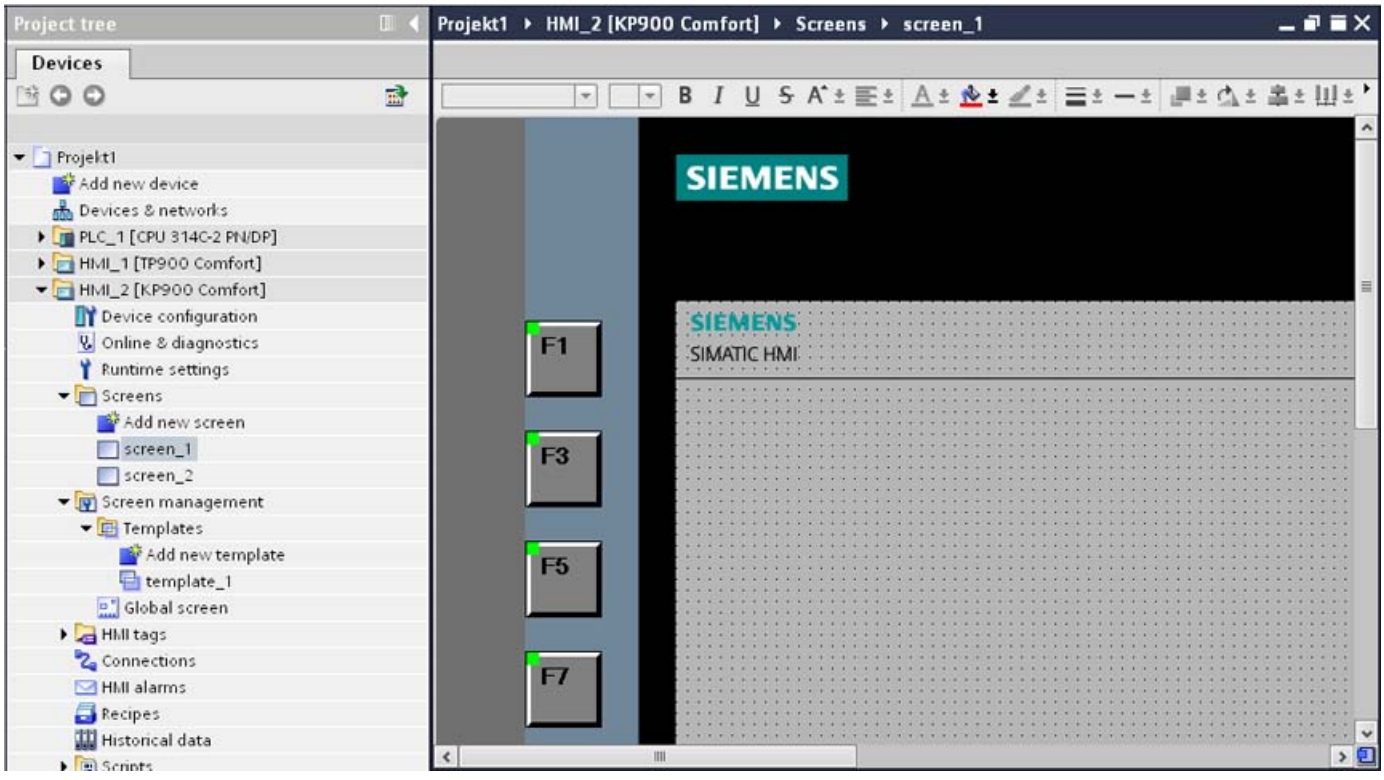

Excluding the controls, the screens are displayed in runtime in the following order:

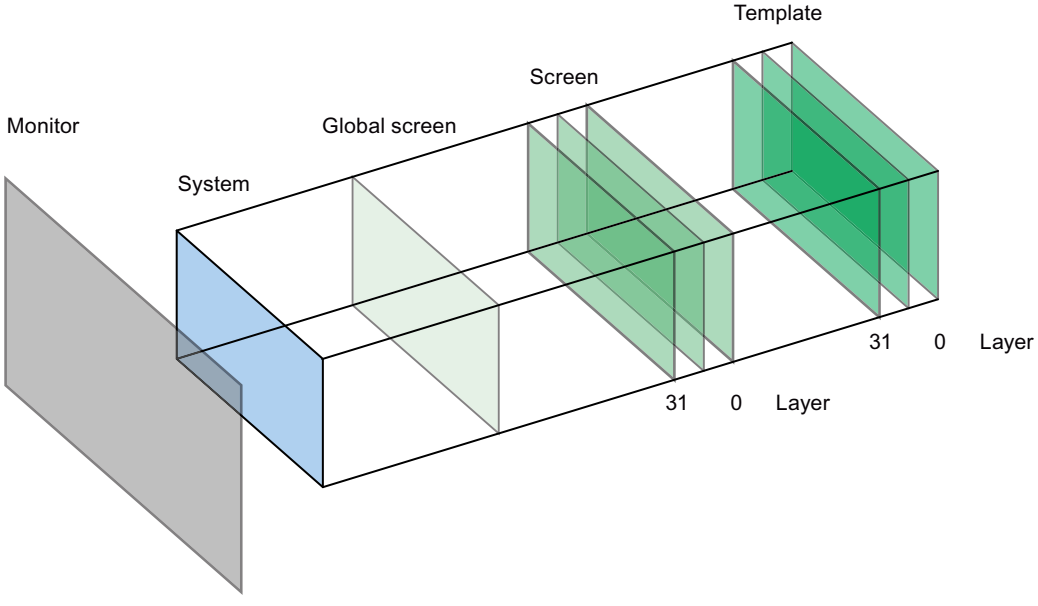

For more detailed information, see:

Screen basics (Page 39)

See also

Screen basics (Page 39)

18.3.3 Scripts in faceplates (RT Advanced, Basic Panels, Panels, Comfort Panels)

Scripts in faceplates in WinCC flexible

You configure a script in the "Scripts" tab of the "Faceplate configuration" dialog. This script is only available within the faceplate.

Interconnect the script directly to the events of the objects contained in the faceplate, for example, with the "Click" event of a button. If you use the faceplate in a screen, a faceplate instance is generated.

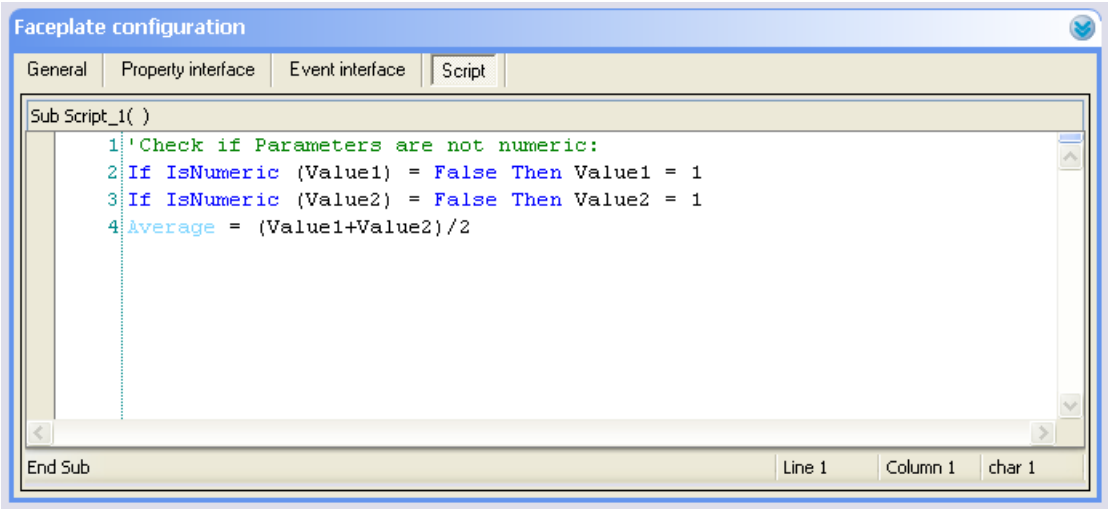

How do I configure scripts in faceplates with WinCC in the TIA Portal?

In the configuration area of the "Faceplates" editor, you create scripts that you only use within a faceplate type.

In contrast to WinCC flexible, WinCC allows you to configure several scripts for a faceplate.

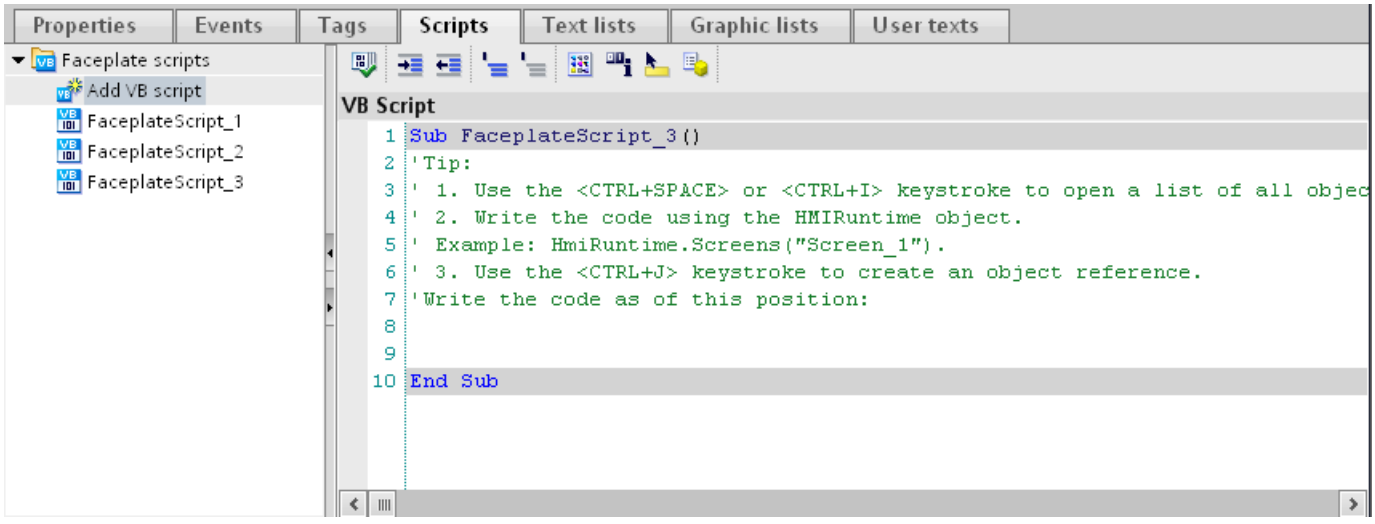

18.3.4 Synchronization of recipes (RT Advanced, Basic Panels, Panels, Comfort Panels)

How do I configure a synchronization of recipes with WinCC in the TIA Portal?

You configure the synchronization of recipes in the "Recipes" editor in WinCC.

A few terms have changed in WinCC as compared to WinCC flexible.

Synchronization for Panels and RT Advanced

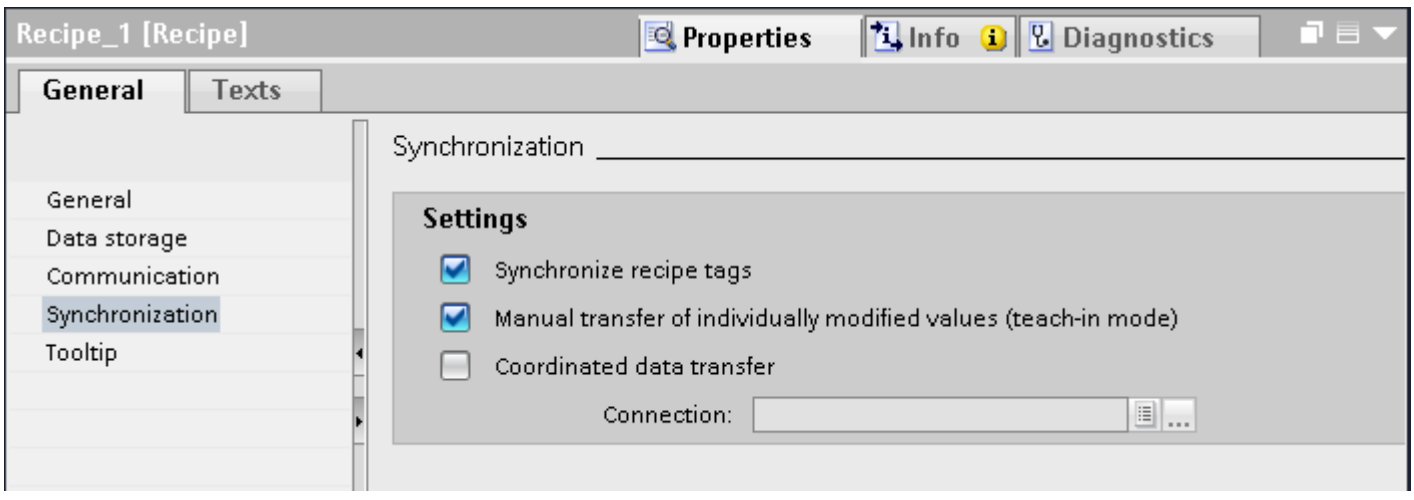

- To synchronize recipe tags that are configured in I/O fields with the recipe view in Runtime, activate "Synchronize recipe view and recipe tags".
- Deactivate "Direct transfer of individually modified values" to specify that the recipe tags are automatically transferred to the PLC when you edit the I/O fields (teach-in mode).
- Activate "Coordinated data transfer" to monitor the transfer of recipe data in Runtime using area pointers.

For more detailed information, see:

AUTOHOTSPOT

18.3.5 Special considerations for converting (RT Advanced, Basic Panels, Panels, Comfort Panels)

Project languages in WinCC

WinCC does not support all project languages that were available in WinCC flexible, such as Arabic. If you receive an empty project as the result of your migration, you may want to check the set editing language. Do not set the project languages that are not supported as editing language in the source project. Proceed as follows:

1. Open the project with WinCC flexible.
2. Change the editing language to English, for example.
3. Save the project.
4. Restart the migration.

Objects with object references in the project library

Two copying methods can be used in WinCC flexible.

- With "simple copy", a WinCC flexible screen including an IO field is copied, for example. Only the object name of a tag configured on the IO field is copied, as this is a reference.
- With "copy", a screen, an IO field contained there and a tag configured on the IO field together with its properties are copied.

These two methods can also be used for storing an object in a library. Project libraries and the objects contained there are migrated during migration and can be used in WinCC.

In WinCC, however, only one copying method is available. With regard to tags, it functions like "simple copy" in WinCC flexible. With regard to graphics, graphics lists and text lists, it functions like "copy" in WinCC flexible.

If you stored objects with references to tags in a library in WinCC flexible, you must reconfigure the referenced objects when using them in WinCC.

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Errors	Errors
System	System
Warnings	Warnings

The display names of the alarm classes can be changed as necessary after migration.

Objects with object references in the project library

Two copying methods can be used in WinCC flexible.

- With "simple copy", a WinCC flexible screen including an IO field is copied, for example. Only the object name of a tag configured on the IO field is copied, as this is a reference.
- With "copy", a screen, an IO field contained there and a tag configured on the IO field together with its properties are copied.

These two methods can also be used for storing an object in a library. Project libraries and the objects contained there are migrated during migration and can be used in WinCC.

In WinCC, however, only one copying method is available. It functions like "simple copy" in WinCC flexible.

If you have stored objects with references to other objects in a library in WinCC flexible, you must reconfigure the referenced objects when using them in WinCC.

Index

"

"Change color reference" dialog, 2105
"Local scripts" editor
    Availability, 1421
    Properties, 1420
"Menus and Toolbars" editor, 308
"Transition/Interlock view"
    PLC code view, 1876

*

*.bmp, 148
*.emf, 148
*.gif, 148
*.ico, 148
*.jpeg, 148
*.jpg, 148
*.svg, 148
*.tif, 148
*.wmf, 148

.

.Net control
    Add, 145
    Remove, 146

A

Absolute Addressing
    of a tag, 526
Access
    Tag, 1390
Access Point, 1757, 1792
Access protection, 1360
    Alarm view, 427
    Configuring, 1365
    User administration, 1356, 1357
Access protection in Runtime
    Alarm view, 423
Access Via the WinCC OLE DB, 1061
ACK
    Key, 2152, 2230, 2318
Acknowledge, 766, 1744
    Key, 2152, 2230, 2318
Acknowledge alarm
    Alarm group, 773
AcknowledgeAlarm, 1620
Acknowledgment, 771
    Configuring, 876, 877, 879
Acknowledgment bit, 1022
    Resetting, 1022
Acknowledgment concept
    Alarm with simple acknowledgment, 773
    Alarm without acknowledgment, 773
Acknowledgment group
    Text, 2116, 2162
Acknowledgment model, 773
    Alarm with dual-mode acknowledgment, 773
    Alarm with simple acknowledgment, 773
    Alarm without "outgoing" status with acknowledgment, 773
    Alarm without "outgoing" status without acknowledgment, 774
    Alarm without acknowledgment, 773
    First value alarm with flashing and single acknowledgment, 774
    New value alarm with flashing and dual-mode acknowledgment, 774
    New value alarm with flashing and single acknowledgment, 774
Acknowledgment request, 778
Acknowledgment tag, 1022
Acquisition cycle, 634, 640
    Defining, 570
    Tag, 567, 568, 593, 596
Acquisition mode
    Tag, 567
Action
    Runtime project to V13 SP1, 2294
activate, 1726
    Autoscroll,
    Project language,
ActivateCleanScreen, 1647
ActivatePLCCodeView, 1603
ActivatePreviousScreen, 1507
ActivateScreen, 1505, 1673
ActivateScreenByNumber, 1506
ActivateScreenInCurrentScreenWindow, 1674
ActivateScreenInScreenWindow, 1674
ActivateSystemDiagnosticsView, 1612
Activating
    Alarms blocks for the alarm view, 854
    Font size adjustment, 84

ActiveX control
Add, 145
Remove, 146
Acyclic triggers, 1803, 1811
Adapting a project
For a different HMI device, 2142, 2203
Add
.Net control, 145
ActiveX control, 145
Graphic to graphics library, 2026
Objects to the group, 161
Adding
User text block for alarm, 824
Additional text for an alarm, 824
Add-ons of third parties, 2244
Address multiplexing
with absolute addresses, 555
with symbolic addresses, 556
Addressing
Addressing tags indirectly, 564, 565, 572
Multiplexing, 564
AdjustContrast, 1581
Advanced recipe view, 1109, 1119
Aggregate functions for analysis, 1066
Alarm
acknowledge, 919, 922
Adding user text blocks, 824
Calling the infotext, 922
Components, 781
Configuring, 794, 809, 828, 831
Display in Runtime, 935
Display suppression mask, 860
Displaying, 429, 431
Editing, 919
Evaluation in Runtime, 871
Event, 954, 955
Exporting, 1971
Filtering, 932
Hiding, 883
Importing, 1973, 1974
In Runtime, 912, 914, 926
Loop-in alarm, 930
Output, 421
Parameter blocks, 806
Priority, 957
Sequence is not adhered to, 1440
Suppress display, 860
System blocks, 806
System function, 953
User text blocks, 806
Alarm annunciator, 850
Configuring, 869
Alarm buffer, 891, 2116, 2162
In Runtime, 913, 915, 927
Memory size, 891
Alarm buffer overflow, 891, 1742
Alarm class, 781
Diagnostics, 842
Identifying, 434
In Runtime, 912, 914, 926
Layout, 917, 920
Alarm classes, 767, 769
Colors, 428, 434, 2116, 2162
Common, 769
Custom, 768, 769
Predefined, 768, 769
Use, 768, 769
Alarm Control
Emergency acknowledgement of alarms, 943
Alarm event
Acknowledge, 766
Incoming, 766
Outgoing, 766
Alarm group, 775, 782
Acknowledge alarm, 773
Configuring, 791
Creating, 791, 792
Status tag, 850
Tags, 778
Alarm Groups from Alarm Classes, 776
Alarm indicator, 438, 836, 923
Alarm classes, 438
Application, 923
Configuring, 845
Events, 438
In Runtime, 913, 916, 928
Layout, 923
Operation, 923
Operation using the mouse, 923
Alarm line, 425, 431
Alarm list, 932, 935
Alarm list entries after power failure, 2247
Alarm log, 885, 1027, 1033
Backup, 904
Configuring, 888, 900, 903
Configuring an alarm view, 849
Connecting, 909
Creating, 885, 888
Disconnecting from project, 909
Displaying contents, 849
In Runtime, 915, 927
Naming conventions, 1031
Signing off of backup, 904
Swapping, 904

Alarm logging, 755, 757
Memory medium, 901
Alarm number, 779, 780, 781
Alarm Priority, 957
Alarm report
Layout in reports, 1295
Use in reports, 1295
Alarm sequence report, 875, 1243
Use in reports, 1294
Alarm statistics, 850, 856
Configuring, 858
Alarm statistics column property changed, 1733
Alarm status, 782
Acknowledged, 766
Incoming, 766
Outgoing, 766
Alarm system, 753
Alarm text, 782
Formatting, 799
Removing format settings, 799
Alarm text block, 809
Configuring, 807
Alarm types, 758, 759, 761
Alarm view, 420, 429, 431, 435, 835, 850, 916, 921, 929, 935
~ configuring for logged alarms, 844, 890
Access protection, 427
Activating alarm text blocks for ~, 854
Alarm line, 431
Alarm text window, 921
Application, 916
Column, 424, 431, 433
Column headers, 429
Column sequence, 427
Configuring, 838, 852
Configuring a filter, 855
Configuring data export, 871
Configuring the display of S7 diagnostic alarms, 842
Configuring the export of Runtime data, 707, 720, 728
Configuring the layout, 841
Configuring the toolbar and status bar, 854
Define columns, 427
Define display area, 428
Enable sorting, 434
Filter alarms, 427
Filter display, 847
Filtering alarms, 932
Identify alarm classes, 427
Identifying an alarm class, 434
Layout, 430, 917, 921
Online configuration, 708, 721, 730
Operation, 917, 921
Operation using the keyboard, 919, 930
Operation using the mouse, 918, 929
Operator control, 421, 430, 433, 918
Operator controls, 426
Select alarm classes, 426
Sorting, 427
Status bar, 706, 719, 725
Toolbar, 706, 719, 725
Use in reports, 1293
Alarm window, 835, 916, 935, 936
Application, 916
Configuring, 844, 936
In Runtime, 913, 916, 928
Operation, 917
Operation using the keyboard, 919, 930
Operation using the mouse, 918
Operator control, 918
Output of log data, 936
Alarm with simple acknowledgment, 773
Alarm without acknowledgment, 773
AlarmControl
Online configuration, 1187
Runtime data, 1186
AlarmHitView, 1066, 1069
Alarms
continuous reporting, 1243
Creating comments, 787
in reports, 1242
Output of a tag value, 800
Output of texts from a text list, 801
Parameter output in discrete alarms, 817
Parameter output in the analog alarm, 813
Parameter output in user alarms, 820
Reloading, 910
Alarms in the event view
Web access, 2247
ALARMVIEW, 1055
AlarmViewAcknowledgeAlarm, 1619
AlarmViewEditAlarm, 1618
AlarmViewShowOperatorNotes, 1619
AlarmViewUpdate, 1618
Align
Object flush, 113, 1273
Allen Bradley Ethernet IP
Data types, 614
Alphanumeric key assignment, 2152, 2230, 2318
ALT
Key, 2152, 2230, 2318

Analog alarm, 759, 760, 761
Configuring, 794, 809, 833
Parameter output, 813
Analog alarm limits
Configuring, 816
Analog alarm types, 759, 760, 761
Analog alarms
Configuring, 783, 784
Analysis functions, 1066
Alarm logs, 1066
Alarm Message Archives, 1069
Process value archives, 1066
Recordset with the analysis of message archives, 1069
Animation, 209
Configuring, 212, 1103
Diagonal movement, 216
Direct movement, 217
Green arrow in Overview, 212
Horizontal movement, 215
Multiple selection, 224
Object group, 222, 223
Overview, 212
Tag binding, 221
Vertical movement, 216
Appearance
Dynamization of an object, 213
Object, 119
Appearance analysis
Bit selection, 2246
Application, 1045, 1046, 1047, 1048
Access to local WinCC user archives, 1047
Alarm indicator, 923
Alarm view, 916
Alarm window, 916
Control Data, 1280
Control Hardcopy, 1281
Control Information, 1282
Date/time field, 1285
f(x) trend view, 1290
Recipe view, 1108, 1119
Remote access to WinCC archive databases, 1046
Remote Access to WinCC RT Databases, 1045
Remote Access to WinCC User Archives, 1048
Simple alarm view, 920
simple alarm window, 920
Simple user view, 341
Application example
Entering recipe data offline, 1155
Recipe with manual production sequence, 1156
Archive Connector, 1073
Archive data, 1040, 1055, 1059, 1064
Access via WinCC OLE DB Provider, 1040
Query, 1055, 1059, 1064
Archive database, 1041
Attach, 1041
ArchiveLogFile, 1538
Area pointer
Data record, 1090
Arrange
Object in the screen, 104
Array, 575, 578, 581, 583
Creating, 578, 583
Indirect addressing, 564
Array tag, 575, 581
Asian characters
Input on the HMI device, 2036
Interpretation, 2036
Memory requirements, 2036
Asian fonts
loading to the HMI device, 2348
Asian languages
Configuration, 2037
Font size, 2037
Text field length, 2037
Asian operating system, 2014
Assign
a graphic to a function key, 250
Object of a layer, 257
Assigning
a function to a function key, 246
Function key, 241, 243, 245
Audit Trail
Checksum, 629, 895, 1032
Authorization
Assignment, 1326, 1366
Changing the name, 1331
Configuring, 1354
Creating, 1324, 1364
Deleting, 1331
Managing, 1329
Autocompletion, 1384
Automatic
Reporting, 796
Automatic display suppression, 860
Automatic hide, 1023
Automatic update of the controller alarms
Configuring, 952
Autoscroll
Activate, 423
Autostart, 2266, 2269
Availability
"Local scripts" editor, 1421

Object for Basic Panels, 326
Object for Comfort Panel, 327
Object for Mobile Panels, 329
Object for WinCC Runtime Advanced, 331
Availability for specific devices
Screen, 41
Availability in Runtime
Online help, 2247
Available system functions
Basic Panels, 1474, 1479
Comfort Panels, 1484
Mobile Panels, 1491
WinCC Runtime, 1498
WinCC Runtime, 1503
Axis
common, 697
Multiple, 696

B

Back page
in reports, 1237
Report, 1228, 1230
Back up RAM file system, 1522
Background color
Dynamization, 213
BACKSPACE key, 2151, 2229, 2317
Backup, 904, 2191, 2344
Deleting, 2211
of alarm log, configuring, 904
Rename, 2211
BackupRAMFileSystem, 1522
Bar, 335, 337
Color transition, 336, 338
Define scale gradation, 339
Display limit lines, 336, 338
Bar segment
Defining, 336, 338
Barcode, 2158, 2242
Barcode scanner, 2157, 2242
Basic Panel
Display and operating object, 326
Runtime start, 2140
Basic Panels
Available system functions, 1474, 1479
Behavior, 2255
Simple recipe view, 1099
Bit (0, 1)
Graphics list, 198
Text list, 183
Bit number (0 - 31)
Graphics list,
Text list,
Bit selection
Appearance analysis, 2246
Design analysis, 2115, 2161
Text and graphic lists, 2115, 2161
Block property changed, 1731
Boundary reached, 1744
Box ID, 1765
Buffer overflow, 2116, 2162
Buffer path
Process Historian, 2248
Button, 472
Adding a system diagnostics indicator, 1832
as release button, 2227
Configured as release button, 2314
Configuring, 1365
Configuring access protection, 1360
Creating, 313
Define hotkey, 473
for release of operator control, 2224, 2313
Graphic, 472
HTML browser, 391
Lock/unlock, 2225, 2313
Mode, 472
PDF view, 443
Recipe view, 1205
Text, 472
Buttons and Switches
Library, 1939

C

C
Examples in WinCC, 1468
C function
Example, 1470, 1471, 1472, 1473
C text lists
Configuration steps, 203
Create entry, 204
Use, 203
CalibrateTouchScreen, 1582
Camera URL, 395
Camera view
Configuring, 395
Cancel
Connection to the faceplate type, 291
Certificate check, 2159, 2243
Change, 1727
Change layout, 1740
Change property time axis, 1735

Change ruler, 1740
Change slider, 1746
Change video format, 1752
ChangeConnection, 1578
ChangeConnectionEIP, 1579
Changing
    Colors, 2105
    Displayed name of user group, 1330
    Logoff time in runtime, 1347
    Name of the user, 1346
    Object property, 1442
    Object size, 110, 1269
    Password, 1329
    Recipe data record in runtime, 1116, 1128
    User group in runtime, 1347
Changing and displaying operating mode (example), 1447
Changing the name
    Authorization, 1331
    User group, 1330
    Users, 1328
Changing windows, 2250
Channel diagnostics display, 399
Character set
    Menus, 2247
Charging condition, 410
    Operation, 411
Checking
    Device version, 2102
Checksum, 896
    Audit Trail, 629, 895
    Log, 629, 895, 1032
    Updating WinCC, 629, 895
Circle, 402
    Design, 119
    Fill level, 119
    Radius, 402
Circular log, 886
    Select size, 622, 899
ClearAlarmBuffer, 1616
ClearAlarmBufferProTool, 1617
ClearAlarmBufferProtoolLegacy, 1617
ClearDataRecord, 1635
ClearDataRecordMemory, 1636
cleared, 1725
ClearLog, 1541
Click, 1738
Click the Alarm view button, 1746
Click the PLC code view button, 1746
Click toolbar button, 1746
Click when flashing, 1738
Client switchover
    Redundancy, 2248
Clock, 503
    Display, 503
    Display dial, 503
    Length of the pointer, 503
    Width of hands, 503
CloseAllLogs, 1542
Code page, 1408
    Runtime settings, 1409
Code templates
    inserting in user-defined functions, 1385
Color of individual ranges
    Gauge, 516
Color transition
    Bar, 336, 338
Colors
    Alarm classes, 428, 434, 2116, 2162
    Changing, 2105
    Find and replace, 2104
Column
    Alarm view, 424, 431, 433
Column headers
    Alarm view, 429
    Recipe view, 464
Column property changed, 1732
Column sequence
    Alarm view, 427
Column width
    User view, 340
Comfort Panel
    Deleting HMI device images, 2213
    Display and operating object, 327
Comfort Panels
    Available system functions, 1484
CommandText, 1064
Comment, 957
Comments
    For alarms, 787
Comments in runtime, 2116, 2162
Common alarm classes, 769
Communication
    via raw data tags, 1193
Compatibility
    WinCC version, 2095, 2096
Compatibility check, 2115, 2161
    Project ID, 2115, 2161
Compatibility mode, 545
Compiling
    Project, 2124, 2172, 2274
Complex user view
    Configuring, 342

Components
Alarm, 781
Compressed log, 645, 653, 1033
Configuring, 653
How it works, 645
Compressed tag, 635
Compression, 646
Computer properties, 1427
Runtime Tab, 1427
Configuration
Analog logging tag, 657
Binary logging tag, 655
Compressed log, 653
Compressed tag, 655
Data log, 651
Large data log, 661
Large log, 661
Log, 651
Logging tag, 655
Process-controlled tag, 655
Tag, 540
Configuration file
Menu, 316
Toolbar, 316
Configuration PC
Starting Runtime, 2199, 2304
Configuring, 798, 816, 819, 834, 1150
Access protection, 1365
Alarm, 809
Alarm acknowledgment, 876, 877, 879
Alarm annunciator, 869
Alarm group, 791
Alarm log, 888, 900, 903
Alarm statistics, 858
Alarm text block, 807
Alarm view, 838, 852
Alarm view filters, 855
Alarm view for logged alarms, 844, 890
Analog alarm, 794, 809, 833
Analog alarm limits, 816
Analog alarms, 783, 784
Animation, 1103
Authorization, 1354
Camera view, 395
Controller alarms, 783, 784
Customized C function, 1409
Data export from the alarm view, 871
Data source of the table view, 704, 717
Data source of the trend view, 704, 717
Discrete alarm, 797, 831
Discrete alarms, 783, 784
Display of S7 diagnostic alarms, 842
Evaluating alarms in Runtime, 871
Event, 1102, 1113, 1124
Event-driven tasks, 796, 812
Function list, 1380
GRAPH overview, 1870
Handle for slide-in screen, 66
Historical alarm list (long-term), 870
Historical alarm list (short-term), 870
Layout of the alarm view, 841
Linking a log backup, 908
Local script at event, 1422
Logging tag, 632, 671
Multiple tags, 541
PLC code view, 448, 1873, 1886
ProDiag overview, 1865
Project-wide alarm class, 788
Properties for slide-in screen, 66
Recipe screen, 1150
Recipe view, 1146, 1148, 1184
Rectangle, 173
Selection in table display, 715
Slide-in screen, 66
Slide-in screen for devices without multi-touch function, 68
Sorting a table column, 716, 728
System events, 783, 784
Table elements, 714, 727
Table view, 711
Time axis, 701
Toolbar and status bar in the alarm view, 854
Trend, 704
Trend view, 699
Trend view for logging, 683
Trend view for values from the PLC, 676, 681
Trigger user alarm, 821
User alarm parameter, 822
User-defined VB function, 1398
Value axis, 702
Value column, 713
Value table, 725
X axis, 702
Configuring a filter
for fixed character string, 847
Historical alarm list (long-term), 870
Historical alarm list (short-term), 870
Configuring HMI array tags, 579, 584
Configuring the sorting
Historical alarm list (long-term), 870
Historical alarm list (short-term), 870
Connected, 1752

Connecting, 669
Alarm log, 909
Log backup, 669
Connection
integrated connection, 526
ConnectionString, 1041
Connectivity Pack, 1073
Archive Connector, 1073
MS OLE DB Provider, 1073
OPC, 1073
WinCC OLE DB Provider, 1073
Connector, 504
Control Data
Application, 1280
Layout, 1280
Control element
Simple alarm view, 921
Control Hardcopy
Application, 1281
Layout, 1281
Control Information
Application, 1282
Layout, 1282
Control tag
Communication between the recipe and the PLC, 1165
Control tags
Example, 1221
Controller alarm, 760, 761, 763
System-defined, 765
Controller alarms, 2162, 2247
Configuring, 783, 784
System-defined, 760, 761
Transferring to the HMI device, 952
Controller data
Initializing, 2110
Controls
Camera view, 395
PDF view, 440
ControlSmartServer, 1527
ControlWebServer, 1528
Coordinated transfer, 1090
With PLC, 1090
Copy
Recipe data record in runtime, 1115, 1128
Screen, 48
Tag, 539
Template, 58
Copying
Adjusting screen size, 2062
Alarm class, 2079
Alarm group, 2079
Analog alarm, 2077
Authorization, 2089
Color, 2057, 2058
Connection, 2070
Cycle, 2085
Data log, 2081
Discrete alarm, 2076
Excel format, 1964
Font, 2057, 2058
Function key, 2062
Graphics list, 2088
HMI devices with HMI connections, 2060
Invalid object, 2057, 2058
Linked objects, 2059, 2060
Logging tag, 2081
Pop-up screen, 63
Principle, 2055, 2056, 2064
Report, 2086
Report object, 2086
Restrictions, 2066
Scheduled task, 2090
Screen, 2062
Screen object, 2067
Screen objects with user data types, 2061
Structural element, 2072
Structure type, 2073
Style sheet, 91
Tag, 2070
Text list, 2088
User group, 2089
User-defined folders, 2056
Users, 2089
VB script, 2083
Corner points, 455
Corners, 453
cp_TagStatistic, 1066
Create
Faceplate type, 270
Global library, 1944
Group, 158
Report, 1238
Report (overview), 1232
Screen, 47
Template, 56
Users, 1326
Users in runtime, 1344
Create entry
C text lists, 204
Create script
Faceplate type, 296
Create text
C text lists, 204

Creating, 1137
Alarm group, 791, 792
Alarm log, 885, 888
Array, 578, 583
Button, 313
Configuration file, 316
Customized C function, 1409
Cycle, 594, 598
Data log, 620
External tag, 531
Folder in HMI files editor, 2107
Hierarchical alarm groups, 793
Info text, 812, 819
Infotext, 796, 798, 816
Internal tag, 533
Main menu command, 310
Menu item, 311
Pop-up screen, 62
Project-wide alarm class, 788
Recipe, 1137, 1172
Recipe data record on the HMI device, 1105, 1106, 1115, 1127
Recipe query, 1179
Report (overview), 1232
Screen, 46
Style sheet, 88
Submenu command, 311
Toolbar, 312
User group, 1324, 1325, 1366
User-defined VB function, 1398
Users, 1368
Creating a new type
Style, 1960
Style sheet, 1961
Creating a type
Screen, 1959
Script, 1959
Creating an analog logging tag, 657
Creating binary logging tag, 655
Criteria analysis view, 1856
Cross-references
Introduction, 2003
Linking, 2008
Replace, 2008
Uses, 2003
csv file,
Example, 684, 956
Layout, 684, 956
Structure for tabular output,
Structure for trend output,
CSV provider table
CSV file structure, 1247
Use in report, 1284
CSV provider trends
CSV file structure, 1244
Layout in reports, 1283
Use in reports, 1283
CTRL
Key, 2152, 2230, 2318
Cursor Control, 2255
Cursor key, 2150, 2228, 2316
Cursor mode
Defining, 167
Custom .Net control
Add, 146
Remove, 146
Custom alarm classes, 768, 769
Custom alarm groups, 776
customized
Menu, 305
Toolbar, 305
Customized C function, 1371
Rename, 1412
Scope of validity, 1406
Customized VB function, 1371
Rename, 1400
Cycle, 640
Creating, 594, 598
Define starting point, 598
Cyclic
Continuous, 567
In operation, 567
Cyclic triggers, 1796, 1804, 1809, 1813, 1816, 1818

D

Data
External data in reports, 1242
Data backup
HMI device, 2337
Data buffer, 646
Data Flow, 1083
Data format differences
between SIMATIC and Microsoft, 1192
Data log, 651, 1027, 1033
Acquisition cycle, 617
Configuring, 651
Configuring the size, 661
Creating, 620
Fast, 667
Logging cycle, 617
Name, 653, 654

Naming conventions, 1031
Output of the tag value, 680
Processing method, 640
Slow, 667
Tags, 618, 623
Term definitions, 633
Tolerance band, 623
Data logging, 616, 634
Application, 616, 634
Basics, 634
Configuration, 634
Cyclic, 642
Cyclic-selective, 643
How it works, 634
On demand, 644
On value change, 645
Data mailbox
For recipes, 1090
Data record
Exporting, 1131
Importing, 1131
read, 1117, 1130
Reading, 1106
Recipe, 1162
Transfer, 1107, 1118, 1131
Data type, 524, 536
Format adjustment, 536
Internal tag, 528, 529
Data types
PLC code view, 1884, 1896
DataConnector, 1073
Date/time field, 350
Application, 1285
Display system time, 351
Format, 351
Layout in reports, 1285, 1286
Use in reports, 1286
Using tags, 351
Deactivate, 1730
Deadband
Setting, 795, 812
Debugger, 1425, 1427
Activating in WinCC, 1427
Opening after error message, 1427
Opening automatically, 1427
Starting for Global Script, 1427
Starting for Graphics Runtime, 1427
Debugger:Basic Principles, 1425
Debugger:Change Picture During Debug, 1425
DecreaseFocusedValue, 1531
DecreaseTag, 1556, 1671
Default font, 2034, 2116, 2162, 2247
Default style, 2115, 2161
Defining, 83
Default template, 2115, 2161
Define
Bar segment, 338
Cursor mode, 167
Reference object, 112, 1271
Scale gradation on the bar, 339
Define columns
Alarm view, 427
Define display area
Alarm view, 428
Define hotkey, 473
Defining, 630, 897
Text format, 423
Defining alarm parameters for a user alarm
Example, 822
DEL key, 2151, 2229, 2317
Delay time
Setting, 795, 811
Delete
Template, 58
Deleted tags
Checking, 540, 2246
Deleting
Authorization, 1331
Local script at event, 1422
Local script at object property, 238, 1423
Object, 107, 1264
Pop-up screen, 63
Recipe data record in runtime, 1116, 1128
Report page, 1250
Screen, 48
Style sheet, 90
Tag, 539
User group, 1330
Users, 1329
Deleting a category
Faceplate type, 276
Deleting a property
Faceplate type, 276
Delta transfer, 2291
Dependency on HMI device
Logging, 1029
Design
Applying design templates to controls, 76
Applying layout specifications to basic elements and elements, 75
Applying screen-specific layout templates, 75
Applying the layout specifications of a design, 74
Changing, 73

Creating, 72
Delete, 73
Design analysis
Bit selection, 2115, 2161
Designing a background, 119
Designing a border, 119
Designs, 2246
Default design for screens, 77
Detail page
Adding to report, 1250
remove from report, 1250
Report, 1228, 1230
Sorting, 1250
Detail view
PLC code view, 1875
Device version, 2132, 2342
Checking, 2102
Switching, 2101
Device wizard, 1929, 1932
Device-dependent
Font size, 85
Diagnostic information
Service project, 2267
Diagnostics, 1424, 1435, 1436
Alarm class, 842
Diagnostics:Debugger, 1424
Diagnostics:GSC Diagnostics, 1436
Diagnostics:Inserting GSC diagnostics window in a screen, 1436
Diagonal movement
Animation, 216
Differences
Runtimes, 2063
Direct key, 2232, 2320
DirectKey, 1510
Configuring, 252
DirectKeyScreenNumber, 1512
Disabling
Project language, 2016
Disconnecting
Alarm log, from project, 909
Log backup, 670
PROFIsafe, 1776
Discrete alarm, 759, 760, 761, 762
Configuring, 797, 831
Parameter output, 817
Structure for the import, 1984
Discrete alarm types, 759, 760, 761
Discrete alarms
Configuring, 783, 784
display,
HMI backup,
Infotext, 2152, 2156, 2230, 2235, 2318, 2323
Limit lines on the bar,
Display classes, 765
Display dial, 503
Display format
I/O field, 358, 360
Display mode
Ruler window, 724
Statistics area window, 724
Statistics window, 724
Display number, 468
Recipe view, 464
Display object
Availability, 333
Availability for Basic Panels, 326
Availability for Comfort Panels, 327
Availability for Mobile Panels, 329
Availability for WinCC Runtime Advanced, 331
Display peak value
Gauge, 515
Display problems
Asian characters, 2348
Controller alarms, 2348
Display program status
GRAPH overview, 385
Display suppression
Define duration, 864
Duration, 2247
Display suppression mask, 860, 1023
Display suppression tag, 860, 1023
of a customized group alarm, 1023
Display suppression timeout
Defining, 864
Display system time, 351
Displaying
Limit lines on the bar, 336
Log contents, 886
Displaying values
As a trend, 676
Distribute
Objects evenly, 114, 1273
Double T-piece, 351
double-click, 1731
Down time
Redundancy, 2248
Download,
"Pack&Go" file to HMI device, 2189
Error messages, 2141, 2201
Project, 2134, 2186
via USB, 2190
Dual-mode acknowledgment, 773

Duration
Display suppression, 2247
DVD 1, 2216
DVD 3, 2216
Dynamic logon, 1322, 1332
Dynamic time range, 695
Dynamization
Color of an object, 213
Control enable state of an object, 218
Flashing, 213
Object, 205
Object property, 205
Object property with a local script, 238, 1423
Dynamize
Appearance of an object, 213
Direct movement, 217
Faceplate, 292, 293
Faceplate type, 292
Green arrow in Overview, 212
Movement of an object, 215
Object appearance, 1832
Property animation, 225
Property list, 208
Quality Code, 226, 233
Tag binding, 221
Dynamizing value column
Table view, 500

E

Eastern characters
Input on the HMI device, 2036
Edit
Object within a group, 163
Recipe, 1970
Editable text field, 363
EditAlarm, 1616
Editing, (Tags)
Folder link, 149
Function list, 1382
Style sheet, 90
System event, 802, 825
Editing language, 2012
Selecting, 2016
Editor
Designs, 70, 71
Graphics, 2025
HMI files, 2107
Effective range, 1757, 1766, 1768, 1771
Calculating signal quality, 512
Configure effective range object, 1787
Configuring, 1785
Configuring logoff, 1786
Configuring logon, 1786
Logoff, 509
Logon, 508
Override function, 512
Principle of Operation, 1758
Runtime, 1768
Work area, 1767
Effective range (RFID), 1769
Configuring, 1789
Configuring logoff, 1790
Configuring logon, 1790
Overview, 1788
Runtime, 1771
Work area, 1769
Effective range name, 507
Effective range name (RFID), 509
Effective range signal, 511
Effective range (RFID)
Logoff, 511
Logon, 510
Effective ranges editor, 1766, 1768
Elements
in reports, 1240
Elements of a user data type
Linking with objects, 143
Ellipse, 363
Horizontal radius, 364
Vertical radius, 364
Ellipse segment, 366
Elliptical arc, 364
E-mail notification, 836
Emergency acknowledgement, 943
Enable sorting
Alarm view, 434
Enabling
Reporting, 874, 875
S7 diagnostic alarm, 805
Encode, 1508
EncodeEx, 1509
End Angle
Circle segment, 405
Circular arc,
Ellipse segment,
Elliptical arc,
END key, 2151, 2229, 2317
Energy-saving mode, 1791
Engineering station
Starting Runtime, 2199, 2304
ENTER key, 2151, 2229, 2317
Entering
Parameters in user-defined functions, 1384

Error, 1736
Error messages
Download, 2141, 2201
Error types, 1425
Error types:Logical error, 1425
Error types:Runtime Error, 1425
Error types:Syntax Error, 1425
ESC key, 2151, 2229, 2317
Event
Acknowledge, 1744
activate, 1726
Alarm buffer overflow, 1742
Alarm statistics column property changed, 1733
Block property changed, 1731
Boundary reached, 1744
Change, 1727
Change layout, 1740
Change property time column, 1735
Change ruler, 1740
Change slider, 1746
Change video format, 1752
cleared, 1725
Click, 1738
Click the Alarm view button, 1746
Click the PLC code view button, 1746
Click toolbar button, 1746
Click when flashing, 1738
Column property changed, 1732
Configuring, 1102, 1113, 1124
Connected, 1752
Deactivate, 1730
double-click, 1731
Dynamize, 210
Error, 1736
Execute, 1727
Exit full screen, 1753
For function lists, 566
Free space critically low, 1754
Full screen, 1753
In the alarm range, 1737
In the tolerance range, 1737
In the warning range, 1738
Incoming, 1737
Input finished, 1735
Loaded, 1727
Lock changed, 1747
Loop-In-Alarm, 1741
Low free storage space, 1753
Medium closed, 1743
Medium opened, 1742
Menu, 314
Move mouse, 1742
Object changed, 1743
On exceeding, 1727
On falling below, 1728
Operator input alarm property changed, 1731
Outgoing, 1737
Overflow, 625, 1750
Pause, 1743
Play ended, 1726
Press, 1731
Press ESC twice, 1736
Press key, 1749
Press key on keyboard, 1749
Press left mouse button, 1740
Press right mouse button, 1745
Property changed, 1732
Property diagram changed, 1732
release, 1741
Release key, 1749
Release key on keyboard, 1750
Release left mouse button, 1741
Release right mouse button, 1745
Replay, 1755
Runtime Stop, 1745
Screen change, 1729
Selection changed, 1727
Shortcut menu, 1739
Statistics area column name property changed, 1733
Status bar element property changed, 1733
Status changed, 1748
StepBack, 1747
StepForward, 1747
Stop, 1748
Switch OFF, 1751
Switch ON, 1752
Tags, 566
Time column property changed, 1735
Time expired, 1755
Toggle, 1751
Toggle toolbar, 1748
Toolbar, 314
Toolbar button property changed, 1733
Trend property changed, 1732
User change, 1729
Value change, 1754
Value column property changed, 1734
Volume, 1739
When dialog is closed, 1729
When dialog is opened, 1728
X-axis property changed, 1734
Y-axis property changed, 1734
Event trigger, 1797, 1806, 1815

Event-driven output
Report, 1252
Event-driven tasks
Configuring, 812
Configuring:event-driven tasks, 798, 816, 819
Events, 640
Start, 640
Stop, 640
Example
Accessing objects (C), 1469
Accessing objects (VBS), 1451
Application for alarm classes, 767, 769
Changing and displaying operating mode, 1447
Configuring a language switch (VBS), 1470
Defining alarm parameters for a user alarm, 822
Determining color of objects (C), 1469
Start script on server (VBS), 1466
System alarm, 764
Triggering user alarm, 821
User-defined function for temperature conversion, 1443
Example C function in WinCC
Configuring a screen change via a property, 1471
Configuring diagnostic output via a trace, 1471
Deactivating Runtime, 1470
Reading tag values, 1472
Writing object properties, 1473
Writing tag values, 1472
Example:
Discrete alarm, 762
System event, 764
Examples
C in WinCC, 1468
Configuring a language switch (VBS), 1452
Configuring a recipe view (VBS), 1462
Configuring a screen change via a property (VBS), 1453
Configuring an alarm view (VBS), 1463
Configuring diagnostic output via a trace (VBS), 1453
Deactivating Runtime (VBS), 1453
Determining color of objects (VBS), 1452
Evaluating tag values (VBS), 1465
Reading tag values (VBS), 1457
VBS in WinCC, 1451
Writing object properties (VBS), 1459
Writing tag values (VBS), 1455
Execute, 1727
Executing
User-defined functions in Runtime, 1438
Exit full screen, 1753
Export
Alarm, 1971
Excel format, 1964
Project texts, 2001, 2022
Recipe, 1624, 1964, 1967
Runtime data, 1186
Tag, 1986
Text list, 1995
User administration, 1352
Export of Runtime data
Configuring, 707, 720, 728
ExportDataRecords, 1622
ExportDataRecordsWithChecksum, 1624
ExportImportUserAdministration, 1547, 1657
Exporting
Recipe, 1131
Recipe data record, 1085, 1131
Expression
Dynamize, 226
Property animation, 226
External graphic
Edit folder, 148
Link folder, 148
Removing the folder link, 148
Rename folder, 148
External image file
add to graphics library, 2027
Managing, 104
Storing in the image browser, 149

F

f(t) trend view, 367, (See trend view)
Online configuration, 708, 721, 730
Operation in Runtime, 733
Toolbar, 369
Use in reports, 1289
f(x) trend view, 371, 374, (See trend view)
Application, 1290
Online configuration, 708, 721, 730
Rules for the data source, 693
Toolbar, 372, 376
Faceplate
Cancel connection to faceplate type, 291
Controlling properties dynamically, 293
Create, 294
Dynamically controlling included objects, 300
Dynamize, 292
Example, 294
Faceplate type, 265
Instance of the faceplate, 265
Reset properties, 265

Resetting the properties to values of faceplate type, 265
Response to sizing, 288, 289
Faceplate type, 265
Cancel connection, 291
Create, 270
Creating a tag, 296
Defining properties, 295
Deleting a category, 276
Deleting a connection, 274
Deleting a property, 276
Deleting a property connection, 274
Dynamize, 292
Edit, 288
Enable, 288
Event, 269
Faceplate, 290
Graphics list, 270
Properties, 269
Response to resizing, 288, 289
Script, 269
Tag, 269
Text list, 269
Updating, 287
Use, 290
User text, 270
factory settings
Resetting to, 2341
Feedback
Optical, 2308
FIFO buffer, 646
Basics, 646
Fill level, 119
Fill style, 484
Filter alarms
Alarm view, 427
Filtering
Alarm list:filter, 932
the alarm view, 847
Filtering alarms, 932
Find and replace
Colors, 2104
First value alarm, 774
Fixed aspect ratio, 484
Flashing, 137, 138, 213, 809
Alarm text block, 809
Flip, 484
Object, 104
Folder link
Editing, 149
Removing, 149
Renaming, 149
Folders
Creating HMI files in the editor, 2107
Font size
Asian languages, 2037
Device-dependent, 85
Font size adjustment
Activating, 84
Footer
Report, 1228, 1230
Foreground color
Dynamization, 213
Format, 351
Format adjustment
Data type, 536
Format DLL, 636
Formatting
Alarm text, 799
Formatting codes
I/O field, 358, 360
Free space critically low, 1754
Full screen, 1753
Full-screen mode, 2115, 2161
Function, 1371
Assigning to a function key, 246
Function list, 1385
Function key, 241
Assigning a function, 246
Assigning a graphic, 250
Global assignment, 243, 2153, 2231, 2319
Global screen, 55
Local assignment, 55, 245, 2153, 2231, 2319
protect with password, 248
Use global assignment, 55
used for screen navigation, 254
Function keys, 2116, 2162
Function list, 566, 1379, 1799
Asynchronous execution, 1439
Configuring, 1380
Converting to local script, 1422
Deleting local script, 1422
Editing, 1382
Execution in Runtime, 1439
Synchronous execution, 1439
Functional scope
ProSave, 2337
Functions
Updating tag value, 523
FunctionTrendControl
Online configuration, 1187
Runtime data, 1186

G

Gauge, 514
  Color of individual ranges, 516
  Display peak value, 515
  GMP-relevant tag, 514
  Maximum value, 515
  Minimum value, 515
  Normal range visible, 515
GetBrightness, 1585
GetDataRecordFromPLC, 1631
GetDataRecordName, 1632, 1633
GetDataRecordTagsFromPLC, 1634
GetGroupNumber, 1549
GetPassword, 1549
GetPLCMode, 1571, 1697
GetUserName, 1548
Global assignment
  Of a function key, 241, 243
Global C tags, 1406
  Application, 1406
  Definition, 1406
Global library, 1934
  Create, 1944
  Displaying logs, 1947
  Open, 1946
  Save, 1945
  Upgrade, 2099
Global screen, 54
  Function key, 55
  Template, 56
Global softkey, 243
GMP, 2116, 2162
GMP-relevant tag
  Gauge, 514
GoTo object, 1837
GoToEnd, 1655
GoToHome, 1654
GRAPH overview, 382, 1856
  Compatibility mode, 384
  Configuring, 1870
  Display program status, 385
  Operating mode, 383
  Single-line mode, 384
Graphic
  Adapting, 378
  Add to graphics library, 2026
  Adjust, 380
  Alignment, 128
  Assigning to a function key, 250
  Button, 472
  Distance, 128
  Graphic view, 378, 1291
  Inserting, 105, 1263
  managing, 148
  Size adjustment, 128
  Stretching, 1291
  Using from the graphic browser, 147
  With transparent background, 147
Graphic browser, 147, 2027
Graphic I/O field, 379, 381
  Output graphics list, 202
  Scroll bar type, 382
  Use in reports, 1292
Graphic lists
  Bit selection, 2115, 2161
Graphic view, 377, 1291
  transparent color, 379
Graphics, 149
  Editor, 2025
Graphics list
  Application, 189
  Bit (0, 1), 198
  Bit number (0 - 31),
  Create, 192
  Creating, 190
  Graphic I/O field, 202
  Outputting configuration data, 201
  Range (0 - 31),
  Range (... - ...), 192, 195, 196
Group
  Adding objects, 161
  Cancel, 160
  Creating, 158
  Edit,
  Editing, 158
  Removing an object, 162
  Ungroup, 160
Group alarm
  Display suppression tag, 1023

H

Handle for slide-in screen
  Configuring, 66
Handwheel, 386
Hardcopy
  Layout in reports, 1292
Harmonizing
  Object size, 110, 1269
Header
  Report, 1228, 1230
Help indicator, 387

Hidden input, 356
Hiding
Report sections, 1251
Hierarchical alarm groups, 776
Creating, 793
Hierarchy levels
User rights, 2116, 2162
Historical alarm list (long-term)
Configuring, 870
Configuring a filter, 870
Configuring the sorting, 870
Historical alarm list (short-term)
Configuring, 870
Configuring a filter, 870
Configuring the sorting, 870
Hit list, 935
HMI backup
Deleting, 2211
Display, 2209
Rename, 2211
HMI device
Changing the device type, 2142, 2203
Configuration, 2101
Data backup, 2334, 2335, 2336, 2337, 2339
Image, 2334, 2335, 2336
Load, 2188, 2334, 2335, 2336
Performance features, 2361, 2364, 2368, 2374, 2379, 2384
Reloading the operating system, 2341
Replacing, 2052
Reset to factory settings, 2342
Restoring data, 2337, 2339
Software, 2334, 2335, 2336
Style, 2115, 2161
System limits, 2361, 2364, 2368, 2374
Transferring license key, 2346
Updating the operating system (Windows CE), 2342
WinCC Runtime Professional performance, 2388
HMI device configuration, 2101
HMI device dependency
in system function, 1388
HMI device image, 2216
Deleting, 2213
HMI device signature, 2159, 2243
HMI device type
Changing, 2142, 2203
HMI device version, 2052, 2101, 2102, (Device version)
HMI device wizard, 1929, 1932
HMI devices with HMI connections
Copying, 2060
HMI files, 2107
HMI style, 1960
HMI style sheet, 1961
HMI tags
Connect PLC array tags, 578, 583
HMI user data types
Interconnecting with faceplates, 274
HOME key, 2151, 2229, 2317
Horizontal radius, 364
Ellipse segment, 366
Elliptical arc, 365
Hotkey, 2252
Defining, 2252
Hardcopy, 2252
Logging off, 2252
Logon, 2252
HTML Browser, 393
Button,
Connection to the FTP server, 393
File explorer, 393
HTMLBrowerZoomIn, 1587
HTMLBrowserBack, 1591
HTMLBrowserForward, 1590
HTMLBrowserHome, 1590
HTMLBrowserPageDown, 1589
HTMLBrowserPageUp, 1589
HTMLBrowserRefresh, 1586
HTMLBrowserScrollDown, 1586
HTMLBrowserScrollLeft, 1588
HTMLBrowserScrollRight, 1588
HTMLBrowserScrollUp, 1587
HTMLBrowserStop, 1585
HTMLBrowserZoomOut, 1588

I

I/O field, 355
application in reports, 1288
Data format, 355, 358, 360
Display format, 358, 360
Display format in reports, 1288
Formatting codes, 358, 360
Hidden input, 356
Lock/unlock, 2225, 2313
Mode, 355
Identify alarm classes
Alarm view, 427
Identifying
Alarm class, 434
Illuminated pushbutton, 411
Tag, 412

Image, 2101, 2131, 2132
HMI device, 2334, 2335, 2336
Image file
Storing in the image browser, 149, 2027
Import
Alarm, 1973, 1974
Analog alarm structure, 1975, 1980
Project texts, 2000, 2024
Recipe, 1966, 1968
Structure of discrete alarms, 1978, 1984
Structure of recipe data, 1970
Tag, 1988, 1989
Tag data structure, 1993
Text list, 1997
User administration, 1353
ImportDataRecords, 1627
ImportDataRecordsWithChecksum, 1628
Importing
Recipe, 1131
Recipe data record, 1085, 1131
Importing a recipe
Structure for the import, 1970
Importing alarms
Structure for the import, 1984
Importing analog alarms
Structure for the import, 1975, 1980
Importing discrete alarms
Structure for the import, 1978, 1984
In the alarm range, 1737
in the screen
Arrange object, 115, 1268
In the tolerance range, 1737
In the warning range, 1738
Incoming, 766, 1737
IncreaseFocusedValue, 1514
IncreaseTag, 1551, 1659
independent screen windows, 2246
Index tag, 564, 565
Indirect addressing, 564
Tags, 572
Info text
Creating, 812, 819
Information area
PLC code view, 1874
Infotext, 782
Creating, 796, 798, 816
display, 2152, 2156, 2230, 2235, 2318, 2323
Key, 2152, 2230, 2318
Initializing
Controller data, 2110
Input field
Character mode, 2151, 2229, 2317
Standard mode, 2151, 2229, 2317
Input finished, 1735
Input on the HMI device
By means of function key, 2153, 2231, 2319
Inserting, 1436
Code template in user-defined functions, 1385
Graphic, 105, 1263
Library object, 1957
Object, 104, 105, 1263
Rectangle, 173
Inserting in a screen, 1436
Inserting:GSC diagnostics, 1436
Installing
Option, 2347
Instructions
Function list, 1385
PLC code view, 1880, 1892
Integrated connection, 526
Intel format, 1192
Inter-Project_Engineering
IPE, 2110
Invalid cross-references, 2006
InverseLinearScaling, 1552, 1660
InvertBit, 1556, 1687
InvertBitInTag, 1558, 1689
InvertLinearScaling, 1552, 1660
IPE, 2110

J

Job types and error codes
Raw data communication for recipes, 1194

K

Key
Backspace, 2151, 2229, 2317
Cursor, 2150, 2228, 2316
Delete, 2151, 2229, 2317
END, 2151, 2229, 2317
ENTER, 2151, 2229, 2317
ESC, 2151, 2229, 2317
HOME, 2151, 2229, 2317
Infotext, 2152, 2230, 2318
Scrolling back, 2151, 2229, 2317
Scrolling up, 2151, 2229, 2317
TAB, 2150, 2228, 2316
Key operation, 2150, 2228, 2316
Key pair for password protection, 2120, 2168

Key switch
Tag, 476
Use, 476
Keyboard commands of the operating system
Activating, 2252
Keyboard layout, 2116, 2162, 2247
Keyboard shortcut
Move object, 114
Keyboard shortcuts, 2116, 2162, 2247
Defining, 2253
Keys, 2255
Know-how protection
Changing a password, 1417
opening a user-defined function, 1416
remove, 1417
Setting up, 1416

L

Label
Recipe view, 464
Language
Activate project language, 2016
Asian languages, 2014
Asian operating system, 2014
Disabling the project language, 2016
Editing language, 2016
Language support, 2014
Language-dependent format, 2013
Language-specific graphic, 2025
Log, 630, 896, 2035
multilingual project, 2018
Reference language, 2016
Regional format of the date, time, currency, and numbers, 2013
Language and font, 2116, 2162, 2247
Language switching, 2030
In Runtime, 1442
Log, 630, 896, 2035
Order, 2116, 2162
Runtime language, 2032
Sequence, 2247
Layer
Assigning objects to a layer, 257
Layout
Alarm indicator, 923
Alarm line, 431
Alarm report, 1295
Alarm view, 420, 429, 430, 432, 436, 917, 921
Bar, 336, 338
Button, 472
Camera view, 395
Charging condition, 410
Circle, 402
Clock, 503
Connector, 504
Control Data, 1280
Control Hardcopy, 1281
Control Information, 1282
CSV provider trends, 1283
Date/time field, 350
Date/time field in reports, 1285, 1286
Editable text field, 363
Effective range name, 507
Effective range name (RFID), 510
Effective range signal, 511
Ellipse, 364
Ellipse segment, 366
Elliptical arc, 365
f(t) trend view, 367
f(x) trend view, 371, 374
Gauge, 514
GRAPH overview, 383
Graphic I/O field, 380, 381
Graphic I/O field in reports, 1292
Graphic view, 378, 1291
Handwheel, 386
Hardcopy in reports, 1292
HTML Browser, 393
I/O field, 355
I/O field in reports, 1288
Illuminated pushbutton, 412
Line, 415
List box, 416
Media Player, 417
ODBC database field, 1297
ODBC database table, 1298
Page number in reports, 1304, 1305
Polygon, 453
Polyline, 454
Print job/Script diagnostics, 352
Project name in reports, 1300
Recipe report, 1303
Recipe view, 460, 463, 467
Rectangle, 459
Report name in reports, 1301
Simple recipe view, 1097
Simple user view, 341
Slider, 474
Sm@rtClient view, 479
Switch, 471
Symbol library, 483
Symbolic I/O field, 485, 487
Symbolic I/O field in reports, 1306

Table view, 498
Text field, 501, 1307
Trend view, 408
User view, 339, 343
Value table, 505
Watch table, 344
WLAN reception, 512
Zone name, 516
Zone signal, 517
Library, 1934
Buttons and Switches, 1939
Copying a library object, 1938
Master copy, 1938
Open, 1946
Save, 1945
Storing an object, 1956
Style, 1960
Style sheet, 1961
System diagnostics indicator, 1829, 1847
Type, 1938
Library management
Filtering, 1943
Opening, 1942
Overview, 1941
Library object, 1934, 1935
Inserting, 1957
Library view
Overview, 1936
License
Managing, 2346
License Keys, 2345
Transfer to an HMI device, 2346
Light emitting diode
Acknowledge, 2152, 2230, 2318
Toggle, 2152, 2230, 2318
Limit, 782
Login attempts, 2116, 2162
Limit value
Tag, 548
Limits
Display, 2116
Show in tooltip, 2162
Tag, 547
Line, 415
Design, 119
Line end, 119, 415
Line start, 415
Line end
Line, 415
Polyline, 454
Line start
Line, 415
Polyline, 454
LinearScaling, 1553, 1662
Link folder
External graphic, 148
Linked objects
Copying, 2059, 2060
Linked Server, 1073
Linking, 2008
List box, 416
List of Hidden Alarms, 883
load
Files in runtime, 2107
Files into the project, 2107
HMI device,
Project,
Recipe,
Recipe data record in runtime, 1116, 1128
Load names, 2116, 2162
LoadDataRecord, 1630
Loaded, 1727
Loading a project
from external storage medium, 2132
with connected HMI device, 2131
Without connected HMI device, 2132
Loading inconsistent projects, 2287
Loading projects
from the external storage medium, 2181, 2282
with connected HMI device, 2180, 2183, 2281, 2284, 2286
Without connected HMI device, 2181, 2183, 2282, 2284, 2286
Loading with S7 routing, 2131
Local assignment
Of a function key, 241, 245
Local script, 1371
Configuring at event, 1422
Deleting at event, 1422
Deleting at object property, 238, 1423
Dynamic control of object properties, 238, 1423
Use, 1419
Using, 1419
Local tag, 1392
Local user groups
Web access, 2247
Lock
Operator control, 2224, 2225, 2313
Lock bit, 1023
Lock changed, 1747
Lock list, 935

Lock tag, 1023
Alarm group, 1023
Log, 2116, 2162
Alarm log, 1027, 1033
Automatic entries, 1029
Characters which may be used in tag names, 653
Checksum, 629, 895, 1032
Compressed log, 1033
Configuration, 651
Configuring the size, 661
Connecting, 669
Data log, 1027, 1033
Disconnecting, 670
Language switching, 630, 896, 2035
Level-dependent management, 622, 898
Runtime language, 630, 896, 2035
Log backup
Connecting, 669, 908
Disconnecting, 670
Log behavior
Control with system function, 625
Level-dependent management, 622, 898
Managing upon system start, 621, 897
Log contents
Displaying, 617, 886
Log data, 936
as reports in the network, 1062
output in alarm window, 936
Log database, 646
Direct access with ODBC, 685, 891
Log entries, 1029
Log language, 630, 896, 2035
Log segments, 2247
Configuring, 661
Log type, 886
Log types, 667
Log with level-dependent execution of system functions, 886
Log with level-dependent system alarm, 886
Logging
Circular log, 617
Cyclic, 642
Cyclic-selective, 643
Log type, 617, 886
On demand, 644
On value change, 645
Segmented circular log, 617
Start, 634
Tag value, 616
Tags, 618, 623
Tolerance band, 623
Within/outside the limit values, 623
Logging cycle, 623, 634, 640
Tag, 593, 596
Logging language, 2115, 2161
Logging method, 634
Circular log, 617
Level-dependent, 617
Segmented circular log, 617
Logging methods, 640
Compression, 640
Cyclic, 640
Cyclic-selective, 640
On change, 640
On demand, 640
Logging tag, 635, 655
analog, 635
binary, 635
Configuration, 655
Configuring, 632, 671
Creating, 655
Logical errors, 1425
Login attempts
Limit, 2116, 2162
Logoff, 1546
Configuring on the effective range, 1786
Configuring to effective range (RFID), 1790
Effective range, 509
Effective range (RFID), 511
Logoff time, 2116, 2162
Changing, 1329
Changing in runtime, 1347
Logon, 1546
Configuring on the effective range, 1786
Configuring to effective range (RFID), 1790
Dynamic, 1322, 2247
Effective range, 508
Effective range (RFID), 510
Logon failed, 1350, 1351
Reporting, 1361
User, 1348
Users, 1350
Logon dialog
Configuring access protection, 1360
LogTag, 1539
Long-term archive list, 935
Long-term archive server, 648
Installation (overview), 648
LookupText, 1529, 1656
Loop-in alarm
trigger, 922
Loop-in-alarm,
Configured, 829
Setting, 424

Low free storage space, 1753

M

Making characters available for
  WString/WChar, 554
Managing
  Graphic,
  License, 2346
  User group, 1329
  Users, 1328
  Users in runtime, 1346
Manual display suppression, 860
Manual hiding, 883
Master copy
  Library, 1938
Media Player, 417
  Number of replays, 418
Medium closed, 1743
Medium opened, 1742
Memory requirements
  Recipe, 2358, 2359
Memory size
  Alarm buffer, 891
Menu
  Configuring an event, 314
  Creating main menu commands, 310
  Creating menu commands, 311
  Creating submenu commands, 311
  customized, 305, 318
Menu command
  Simple recipe view, 1099
Menus
  Character set, 2247
Menus and toolbars, 2246
Message archive, 1055, 1064
  Displaying data, 1057
  Querying archive data, 1055, 1064
  Recordset, 1057
Microsoft Script Debugger, 1424
Mitsubishi MC
  Data types, 615
Mobile Panel
  Display and operating object, 329
Mobile Panels
  Available system functions, 1491
Mobile Wireless, 1767, 1768
  Work area, 1767
  Zone ID / connection point ID, 1792
Mobile Wireless (RFID)
  Work area, 1769
Mobile Wireless V2, 1769
Mode, 472
  Graphic I/O field, 380, 382
  I/O field, 355
  Symbolic I/O field, 487
Modicon Modbus TCP/IP
  Data types, 614
Modifying recipe structures, 1133, 1177
Move
  Screen, 48
Move mouse, 1742
Move object
  Keyboard shortcut, 114
Moving columns
  User view, 340
Moving in a group
  Pop-up screen, 63
MS OLE DB, 1071
Multiple selection, 111, 1270
Multiplexing, 564
  Address multiplexing, 555
  with absolute addresses, 555
  with symbolic addresses, 556
Multi-touch devices
  Gestures, 2223, 2311
  Scrolling, 2221
  Two-hand operation, 2224, 2313
  Zoom in and out, 2222
  Zooming, 2222
Multi-touch operation, 2311
Multiuser engineering
  Compiling and loading, 2136, 2195
  Compiling in the local session, 2139, 2198, 2298
  Compiling in the server project view, 2138, 2197, 2297
  Rules for compiling and loading, 2137, 2196

N

Naming conventions
  Alarm log, 1031
  Data log, 1031
New key pair
  Generation, 2122, 2170, 2271
New value alarm, 774
Normal range visible, 515
NotifyUserAction, 1513
Number of lines
  User view, 340
Number of replays
  Media Player, 418
Numerical key assignment, 2152, 2230, 2318

NOT DEFINED

add external graphic, 2027

O

Object
Arrange, 104, 115, 1268
Assigning to a layer, 257
Availability, 333
Availability for Basic Panels, 326
Availability for Comfort Panels, 327
Availability for Mobile Panels, 329
Availability for WinCC Runtime Advanced, 331
Creating a new OLE object, 104
Creating an OLE object from a file, 104
Deleting, 107, 1264
Deleting local script, 238, 1423
Design, 119
Designing the background color, 121
Designing the fill pattern, 122
Dynamization, 205
Dynamization of the appearance, 213
Dynamize direct movement, 217
Dynamizing a property with a local script, 238, 1423
editing within a group, 163
Evenly distributing, 114, 1273
Flashing, 137, 138
Flip, 104
Flush alignment, 113, 1273
Group,
Grouping, 158
Inserting, 104, 105, 1263
Inserting of the same type, 104
Inserting several times, 138, 1265
Multiplying, 138, 1265
outside the area, 116
Paste,
Referencing, 1395
Repositioning and resizing multiple objects, 104, 140, 1272
Repositioning objects, 108, 1266
Resize, 110, 1269
Rotate, 104, 117, 129, 1274
Select multiple objects, 111, 1270
Selecting multiple objects, 104
Storing in a library, 1956
Tab sequence, 104
Transparency, 122
Object changed, 1743
Object group
Animation, 222, 223
Editing an object within a group, 163
Removing an object, 162
Object list
Tag, 532
Object property
Dynamization, 205
Object size
Harmonizing, 110, 1269
Objects
Alarm report, 1295
Alarm sequence report, 1294
Alarm view, 1293
CSV provider table, 1284
CSV provider trends, 1283
Date/time field, 1286
f(t) trend view, 1289
ODBC database field, 1297
ODBC database table, 1298
Page number, 1304, 1305
Pop-up screen, 61
Position in reports, 1236
Project name, 1300
Recipe report, 1303
Recipe view, 1302
Report name, 1301
Size in reports, 1236
Slide-in screen, 65
Table view, 1306
with dynamic length in reports, 1237
ODBC database field
Layout in reports, 1297
Use in reports, 1297
ODBC database table
Layout in reports, 1298
Use in reports, 1298
OLE object
create from a file, 104
Recreate, 104
Save in Graphics, 149
Storing in the image browser, 2027
On demand, 567
On exceeding, 1727
On falling below, 1728
Online
Recipe tag, 1087
Online configuration, 708, 721, 730, 1187
OnlineTrendControl, 737

Table view, 744, 747
Trend view, 747
Online delta loading, 2291
Configuring, 2292
Restrictions, 2292
WinCC Web Navigator support, 2292
Online help
Availability in Runtime, 2247
OnlineTableControl
Online configuration, 1187
Runtime data, 1186
OnlineTrendControl
Online configuration, 737, 1187
Runtime data, 1186
On-screen keyboard, 2247
OP 177B mono
Symbol library, 484
OPC, 1073
Alarms and events, 2247
OPC access
Service project, 2267
OPC DA
Data types, 612
OPC Historical Access, 2247
OPC Unified Architecture Server, 2116, 2162
OPC XML DA
Data types, 613
Open
Global library, 1946
Selection list, 2151, 2229, 2317
OpenAllLogs, 1542
OpenCommandPrompt, 1516
OpenControlPanel, 1518
OpenControlPanelDialog, 1515
OpenFileBrowser, 1648
opening
Menus and Toolbars, 308
OpenInternetExplorer, 1517
OpenScreenKeyboard, 1649
OpenTaskManager, 1649
Operability
Dynamization, 218
Operating element
Availability, 333
Availability for Basic Panels, 326
Availability for Comfort Panels, 327
Availability for Mobile Panels, 329
Availability for WinCC Runtime Advanced, 331
Operating mode
GRAPH overview, 383
Operating object
operation, 2151, 2229, 2317
Recipe view, 1109
Operating principle
Service project, 2268
Operating recipes
Copying a recipe data record, 1115, 1128
Creating recipe data records, 1105, 1106, 1115, 1127
Deleting a recipe data record, 1116, 1128
Load recipe data record, 1116, 1128
Modifying a recipe data record, 1116, 1128
Modifying the recipe structure, 1133, 1177
Read recipe data record, 1106
Reading data records, 1117, 1130
Transfer data record, 1107
Transferring a data record, 1118, 1131
Operating system
Asian language setting, 2014
Setting to Western, 2014
operation, (Two-hand)
Alarm indicator,
Alarm view,
Alarm window,
Feedback,
Key, 2150, 2228, 2316
Operating object, 2151, 2229, 2317
Recipe view,
Simple alarm view,
Simple recipe view,
Operation feedback, 2307
Operation using the keyboard
Alarm view, 919, 930
Alarm window, 919, 930
Simple alarm view, 922
Operation using the mouse
Alarm indicator, 923
Alarm view, 918, 929
Alarm window, 918, 929
Simple alarm view, 922
Operation with the keyboard
PLC code view, 448
Recipe view, 1113, 1124
Operation with the mouse
Recipe view, 1113, 1124
Operator control
Alarm view, 421, 430, 433, 437, 918
Alarm window, 918
f(t) trend view, 369
f(x) trend view, 372, 376
HTML browser, 391
Lock/unlock, 2225, 2313

Locked, 2224, 2313
Media Player, 418
PDF view, 443
Recipe view, 460, 467, 1120
Table view, 499
Trend view, 409
Value table, 506
Watch table, 345
Operator controls
Alarm view, 426
Recipe view, 463
Operator input alarm property changed, 1731
Optical feedback, 2308
Optical handheld reader, 2157, 2242
Option
Installing, 2347
Uninstalling, 2347
Order
Language switching, 2116, 2162
Outgoing, 766, 1737
Output alarm, 421
Overflow, 1750
Override function, 512

P

Pack&Go file
Create, 2188
Download to HMI device, 2189
Page
Adding to report, 1250
Deleting report page, 1250
Maximum number of configurable in report, 1233
Sorting report page, 1250
Page number
Layout in reports, 1304, 1305
Use in reports, 1304, 1305
PageDown, 1653
PageUp, 1654
Parameter block
Configuring, 807
Parameter blocks, 806
Parameter transfer
User-defined function, 1389
Parameters
Output in analog alarm, 813
output in discrete alarms, 817
output in user alarms, 820
Password, 2116, 2162
Changing, 1329
for function key, 248
Hierarchy level, 1317, 1320
Password aging, 1318
Password complexity, 1318, 1321
Password for transport
Creating, 2121, 2169, 2270
Reset, 2122, 2170, 2271
Password protection in Runtime, 2168, 2269
Pasting
Adjusting screen size, 2062
Color, 2057, 2058
Font, 2057, 2058
Function key, 2062
Invalid object, 2057, 2058
Principle, 2055, 2064
Pause, 1743
PDF view
Button, 443
Configuring, 440
Multi-touch operation, 443
Toolbar, 441
PDFFitToHeight, 1597
PDFFitToWidth, 1597
PDFGotoFirstPage, 1598
PDFGotoLastPage, 1598
PDFGotoNextPage, 1599
PDFGotoPage, 1599
PDFGotoPreviousPage, 1600
PDFScrollDown, 1596
PDFScrollLeft, 1601
PDFScrollRight, 1601
PDFScrollUp, 1596
PDFZoomIn, 1600
PDFZoomOriginal, 1602
PDFZoomOut, 1600
Performance features
HMI device, 2361, 2364, 2368, 2374, 2379, 2384
WinCC Runtime, 2379
WinCC Runtime Professional, 2384
WinCC Runtime Professional performance, 2388
Permanent area, 58
Picture cycle, 596
Pipe, 469
Pipe bends, 470
Angle, 470
Radius, 470
Play ended, 1726
PLC
Enabling system diagnostics, 1828, 1840
System diagnostics view, 1828, 1840
Tag, 551
PLC code view, 446, 450, 1856
"Transition/Interlock view", 1876
Configuring, 448, 1873, 1886

Data types, 1884, 1896
Detail view, 1875
Information area, 1874
Instructions, 1880, 1892
Operation with the keyboard, 448
Symbol table, 1875
Toolbar, 447, 451, 1878
PLC data type, 609, 612, 613, 614, 615
PLC data type in WinCC, 524
PLC data types, 608, 611
PLC prefix, 545
PLC tag
Reconnecting, 546
PLC UDTs
WinCC, 611
PLC user data types
Interconnecting with faceplates, 274
PLCCodeViewActualOrTriggerValues, 1607
PLCCodeViewDetails, 1605
PLCCodeViewMoveScrollBar
PLCCodeViewMoveScrollbar, 1604
PLCCodeViewNextNetwork, 1605
PLCCodeViewPreviousNetwork, 1609
PLCCodeViewStepMode, 1606
PLCCodeViewSymbols, 1606
PLCCodeViewTransitionInterlock, 1607
PLCCodeViewZoomIn, 1608
PLCCodeViewZoomOut, 1608
Polygon, 453
Configuring corners, 453
Layout, 453
Radii, 453
Polyline, 454
Configuring corners, 455
Line end, 454
Line start, 454
Radii, 455
Pop-up screen, 60
Copying, 63
Creating, 62
Deleting, 63
Moving in a group, 63
Objects, 61
Position
Editing multiple objects, 140, 1272
Of objects, 108, 1266
of objects in the report, 1233
Power failure
Alarm list entries, 2247
Power Management, 1791
Predefined alarm classes, 768, 769, 770
Predefined styles, 124
Press, 1731
Press ESC twice, 1736
Press key, 1749
Press key on keyboard, 1749
Press left mouse button, 1740
Press right mouse button, 1745
PressButton, 1521
Print
Network, 2358
Print job
Creating a task for automatic start, 1258
creating for the report, 1255
Print job/Script diagnostics, 353
Print job list, 1252
Print job/Script diagnostics, 352
Function, 353
Print job, 353
Print server, 2357
Printing
Print server, 2357
Report, 1251, 1277
Printing of alarms
Configuring:printing of alarms, 834
PrintReport, 1570
PrintScreen, 1569
Priority, 957
Setting, 812
Process diagnostics overview, 1856
Process Historian
Buffer path, 2248
Server selection, 2248
Process value
display, 690
Saving to database, 646
Swapping out, 648
Process value archive, 1049, 1064
Displaying data, 1049
Query, 1049
Querying archive data, 1064
Recordset, 1049
Process value logging, 616
Process-controlled tag, 636
Processing method, 641
Data log, 640
ProDiag overview, 456
Configuring, 1865
Symbols, 1863
Use, 456
PROFIBUS DP
DirectKey, 1513
PROFIsafe
Disconnecting, 1776

PROFIsafe address, 1791
Project, 2276
compile (overview), 2122, 2170, 2272
Compiling, 2124, 2172, 2274
Download, 2134, 2186
download (overview), 2122, 2170, 2272
Enable, 2304
Load, 2289
multilingual, 2018
Simulation with a tag simulator, 2127, 2175
Upgrade, 2097
WinCC status, 2111
Project data
Export alarm, 1971
Export text list, 1995
Exporting a recipe, 1964, 1967
Exporting tag, 1986
Importing a tag, 1987, 1989
Importing alarms, 1972, 1974
Project graphics, 2115, 2161, 2246
Project ID, 2115, 2161
Project language, 2012
Activate, 2016
Disabling, 2016
Project languages
System texts, 2019
User texts, 2019
Project library, 1934, 2417
Project name
Layout in reports, 1300
Use in reports, 1300
Project size
Reducing, 2144, 2204
Project text
Exporting, 2001, 2022
Project texts
Displaying reference text, 2021
Export, 286
Importing, 2000, 2024
Translating individual texts, 284, 2020
Translation into project languages, 2019
Project version
Upgrade, 2097
Project-wide alarm class
Create, 788
Use, 789
Properties
"Local scripts" editor, 1420
Window, 2246
Properties for slide-in screen
Configuring, 66
Property animation
Expression, 226
Quality Code, 226, 233
Type, 225
Type binary, 225
Type bit, 226
Type Bool, 226
Type direct, 226
Type range floating point, 225
Type range integer, 225
Property changed, 1732
Property diagram changed, 1732
Property list
Dynamize, 208
ProSave, 2337
Asian fonts, 554
Protect
Function key with password, 248

Q

Quality Code
Dynamize, 226, 233
Query, 1049, 1055, 1059
Message archive, 1055
Process value archive, 1049
User archive, 1059

R

Radius, 402, 453, 455
Circle segment, 404
Circular arc, 403
Range (... - ...)
Graphics list, 195, 196
Text list, 182
Raw data tag, 636, 1198
Acknowledgement Data, 1198
Acknowledgement header structure, 1197
Acknowledgment message frame, 1197
Communication between the recipe and the PLC, 1165
for BSEND/BRCV functions,
Structure of job header, 1196
Structure of the message frame header, 1195
Raw data tags, 1193
Raw data type, 636
Read continuously
Tag, 567

Recipe, 1077, 1079, 1080, 1088, 1160, 1161, 1970
Application example: Machine parameter assignment, 1079
Application example:Batch production, 1079
Basics, 1077, 1079, 1161
Communication by means of control tags, 1165
Configuration option, 1086
Creating new, 1137, 1172
Data Flow, 1083
Data record, 1080, 1162
Display in I/O fields, 1182
Display in Runtime, 1164
Edit, 1970
Element, 1162
Entry, 1080
Example, 1162
Export format, 1624
Exporting, 1131, 1964, 1967
Importing, 1131, 1966, 1968
Load, 1171
Memory requirements, 2358, 2359
Principles, 1160
Raw data communication, 1165
Raw data tag communication type, 1189
Synchronizing recipe tag, 1116, 1129
Table view in Runtime, 1181
Tags communication type, 1189
Use, 1079, 1160
Use of text lists, 1092
Recipe data
in reports, 1241
Recipe data record
Changing, 1116, 1128
Copy, 1115, 1128
Creating new, 1137, 1172
Creating on the HMI device, 1105, 1106, 1115, 1127
Deleting, 1116, 1128
Export format, 1624
Exporting, 1131
Importing, 1131
Importing and exporting, 1085
load, 1116, 1128
Synchronizing, 1116, 1129
Transfer option, 1083
Use of text lists, 1092
Recipe data record name
Writing to a tag, 1101, 1112, 1123
Recipe data record number
Writing to a tag, 1101, 1112, 1123
Recipe element
Creating new, 1137, 1172
Recipe name
Writing to a tag, 1101, 1112, 1123
Recipe number
Writing to a tag, 1101, 1112, 1123
Recipe query
Creating new, 1179
Restrictions in Runtime, 1178
Recipe query editor, 1178
Recipe report
application in reports, 1303
Layout in reports, 1303
Recipe screen, 1094
Automatic transfer, 1094
Configuring, 1150
Recipe tag, 1082
Synchronizing recipe value, 1094
Visual machine simulation, 1094
Recipe tag
Synchronizing, 1086, 1116, 1129
Teach-in mode, 1087
Recipe value, 1094
Automatic transfer in the recipe screen, 1094
Synchronizing in the recipe screen, 1094
Recipe view, 460, 466, 1108, 1109, 1119, 1164, 1181
Advanced, 1109, 1119
Application, 1108, 1119
Behavior with screen change, 1114, 1127, 1134
Buttons, 1205
Column headers, 464
Configurable events, 1102, 1113, 1124
Configuring, 1146, 1148, 1184
Configuring the export of Runtime data, 707, 720, 728
Display number, 464, 468
Displaying one recipe only, 1111, 1122
Displaying values only, 1101
Dynamic properties, 1103
Edit data, 1207, 1214
Label, 464
Layout, 460, 463, 465
Online configuration, 708, 721, 730
Operating object, 1109
Operation, 1109, 1119
operation using function keys, 1104, 1114, 1125
Operation using the function key, 1134
Operation with the keyboard, 1113, 1124
Operation with the mouse, 1113, 1124
Operator control, 460, 467, 1120
Operator controls, 463
Recipe data record, 1081, 1082, 1135
Simple recipe view, 466

Status bar, 706, 719, 725, 1205
Toolbar, 465, 706, 719, 725, 1205
Updating, 1110, 1121
Use, 465
Use in reports, 1302
UserArchiveControl, 1207
Using as a drop-down list, 1112, 1123
Recipes
Acknowledgement data for the raw data tag, 1198
Acknowledgment message frame structure for raw data tag, 1197
Communication via raw data tags, 1195, 1197
Export CSV file from ES, 1145, 1177
Import CSV file into ES, 1176
Import CSV file to ES, 1143
Job types and error codes for raw data communication, 1194
Message frame structure for raw data tags, 1195
Redundancy, 2248
Structure of acknowledgement header for raw data tag, 1197
Structure of job header for raw data tag, 1196
Structure of message frame header for raw data tag, 1195
RecipeViewBack, 1642
RecipeViewClearDataRecord, 1638
RecipeViewGetDataRecordFromPLC, 1637
RecipeViewMenu, 1638
RecipeViewNewDataRecord, 1637
RecipeViewOpen, 1639
RecipeViewRenameDataRecord, 1641
RecipeViewSaveAsDataRecord, 1640
RecipeViewSaveDataRecord, 1640
RecipeViewSetDataRecordToPLC, 1639
RecipeViewShowOperatorNotes, 1642
RecipeViewSynchronizeDataRecordWithTags, 1641
Reconnecting
Tag, 546
Reconnecting a PLC tag, 546
Recordset, 1049, 1057
Rectangle
Configuring, 173
Inserting, 173
Inserting and configuring, 172
Radius corners X, 459
Radius corners Y, 459
Reducing
Project size, 2144, 2204
Redundancy
Client switchover, 2248
Down time, 2248
Recipes, 2248
Synchronization, 2248
Redundant servers and partners, 2248
Reference language, 2012
Selecting, 2016
Reference object
Defining, 112, 1271
Referencing
Object, 1395
Release,
User data type, 589
Release button, 2224, 2313
Configuring, 2227, 2314
Release key, 1749
Release key on keyboard, 1750
Release left mouse button, 1741
Release right mouse button, 1745
ReleaseButton, 1521
Reloading
Alarms, 910
Remove
Formatting in the alarm text, 799
Object from the group, 162
Removing
Folder link, 149
Rename
Customized VB function, 1400
Screen, 48
Tag, 539
Template, 58
user-defined C function, 1412
Renaming recipe data records, 1128
Replace
Cross-references, 2008
HMI device, 2052
Replace delimiter, 545
Replacing devices
Limitations on connections, 2041
Principle, 2041
Replay, 1755
Report
Adding a detail page, 1250
Back page, 1228, 1230
Create a print job, 1255
Creating, 1238
creating (overview), 1232, 1235
Creating a task for the automatic start of a print job, 1258
Deleting page, 1250
Detail page, 1228, 1230
Event-driven output, 1252
Footer, 1228, 1230

Header, 1228, 1230
Maximum configurable number of pages, 1233, 1238
Objects with dynamic length, 1237
Position of objects, 1236
print, 1251
printing, 1277
Showing and hiding sections, 1251
Size and position of objects, 1233
Size of objects, 1236
Sorting pages, 1250
Steps for configuring report output in Runtime (Advanced), 1251
Steps for configuring report output in Runtime Professional, 1253
Time-driven output, 1252
Title page, 1228, 1230
Title page and back page, 1237
Watermarks, 1230
Report name
Layout in reports, 1301
Use in reports, 1301
Report objects
in reports, 1242
Reporting
Activating, 875
enabling, 874
Reporting Services
Requirements, 1062
Reports
Alarm report, 1295
Alarm sequence report, 1243
CSV provider trends, 1283
Data output in tabular format, 1241
Data output in trends, 1241
Date/time field, 1285, 1286
Elements, 1240
Graphic I/O field, 1292
Hardcopy, 1292
I/O field, 1288
ODBC database field, 1297
ODBC database table, 1298
Output of a Runtime snapshot, 1242
Output of alarms, 1242
Output of external data via ODBC interface, 1242
Output of recipe data, 1241
Page number, 1304, 1305
Project name, 1300
Recipe report, 1303
Report name, 1301
Report objects, 1242
Symbolic I/O field, 1306
Reports with log data
available in the network, 1062
Reset
Properties of a faceplate, 265
to factory settings, 2341
ResetBit, 1560, 1690
ResetBitInTag, 1561, 1692
ResetTagToHandWheel, 1530
Resetting
Acknowledgment bit, 1022
HMI device to factory settings, 2342
to factory settings, 2341
Resizing
Faceplate, 288, 289
Response to resizing
Faceplate type, 288, 289
Restore
Data of the HMI device, 2337, 2339
Restoring
From external storage device, 2191
from external storage medium, 2344
Restoring data
HMI device, 2337, 2339
RFID tag, 1760
Rotate
Object, 104, 129
Rotation, 129
T piece, 502
RulerControl, 505
Online configuration, 1187
Runtime data, 1186
Runtime, 1768, 1771, 2255, 2290, 2291
Buttons, 1205
Changing object property, 1442
configuring in Runtime, (See Online Configuration)
Displaying logged alarms, 936
Effective range, 1768
Effective range (RFID), 1771
Emergency acknowledgement of alarms, 943
Executing user-defined functions, 1438
Execution of the function list, 1439
Filtering alarms, 932
Language switching, 1442
Multiple-station system, 2304
Redundant system, 2304
Requirements, 2304
Simulating, 2125
Sorting alarms, 933
Start screen, 50, 51
Starting, 2304
starting on a panel, 2140, 2201
starting on PC, 2201

Runtime language, 2012, 2029
Font, 2034
Log, 630, 896, 2035
Order for language switching, 2032
Selecting, 2031
single-language objects, 2247
Runtime languages, 2116, 2162, 2247
Runtime Professional
Defining shortcuts, 2252
Setting up the cursor control, 2250
Setting up the startup list, 2258
Setting up the zoom functions, 2256
Runtime project
Action, 2294
Runtime screen
in reports, 1242
Runtime scripting, 1371
Functions, 1371
Local script, 1371
System functions, 1371
User-defined functions, 1371
Runtime settings, 2306
Code page, 1409
Dynamic logon, 1322
User administration, 50, 52, 1316, 1319, 1322, 1331, 1332, 1337, 1338, 1347, 1348
Runtime start
Basic Panel, 2140
Runtime Stop, 1745
Runtimes
Differences, 2063

S

S7 1500
Data types, 609
S7 300/400
Data types, 608
S7 diagnostic alarm
enabling, 805
S7 diagnostic alarms, 765
S7 Protocol Suite
Raw data tag, 1198
Raw data tag for BSEND/BRCV functions, 1200
S7 routing
Loading, 2131
S7-1200 V2
Data types, 609, 611
SafelyRemoveHardware, 1515
Safety instruction
Direct key, 2231, 2319
Save
Data of the HMI device, 2337, 2339
Global library, 1945
SaveDataRecord, 1646
Scaling
Applying linear scaling to a tag, 551, 553
Scheduler, 1793
Acyclic triggers, 1796, 1803
Administer task, 1808
Cyclic triggers, 1796, 1813, 1816, 1818
Deactivated task, 1796
Event trigger, 1796, 1797, 1806, 1815
Function list, 1799
Print report, 1816
Tag triggers, 1798
Timers for cyclic and acyclic triggers, 1797
Triggers, 1795, 1796
Work area, 1794
Screen
Availability for specific devices, 41
Copy, 48
Copying, 2062
Create, 47
Creating, 46
Creating a type, 1959
Define customized menu, 318
Define customized toolbars, 318
Deleting, 48
Font, 41
Inserting, 48
Library, 1959
Move, 48
Rename, 48
Use template, 60
Working steps in creating, 45, 46
Zooming, 43
Screen buffer, 2246
Screen change, 1729
Displaying the screen on entering a zone, 1776, 1778, 1782
Screen designs, 70
Screen keyboard, 2116, 2162
Basic Panels, 2147
Basic Panels 2nd Generation, 2148
RT Advanced, 2220
Screen object
Dynamic movement, 215
Dynamization of the control enable state, 218
Screen resolution, 2115, 2161
Screen selection
Setting up, 2250
Screen template, 243

Screen window
Configuring a menu, 318
Configuring a toolbar, 318
Screen windows
independent, 2246
ScreenObjectCursorDown, 1650
ScreenObjectCursorLeft, 1653
ScreenObjectCursorRight, 1651
ScreenObjectCursorUp, 1651
ScreenObjectPageDown, 1652
ScreenObjectPageUp, 1652
Screens
Applying a design, 74
Designs, 70
Script
Creating a type, 1959
Library, 1959
Print job/Script diagnostics, 353
Service project, 2267
Updating tag value, 523
Scripts in Runtime, 2246
Scrolling
Multi-touch devices, 2221
Scrolling back key, 2151, 2229, 2317
Scrolling up key, 2151, 2229, 2317
Segment, 646
Segmented circular log, 886
Segments
Define for log, 661
Select
Multiple objects, 111, 1270
Select alarm classes
Alarm view, 426
Selecting
Multiple objects, 104
Selection
Selection dialog box, 1212
Selection changed, 1727
Selection list
Open, 2151, 2229, 2317
SendEMail, 1519
Sequence
Language switching, 2247
Sequence of columns, 345
Server selection
Process Historian, 2248
Server utilization, 2247
Service mode, 2265
Service project
Configuration, 2266
Configuring, 2268
Diagnostic information, 2267
OPC access, 2267
Restriction, 2267
Script, 2267
Task, 2267
Use, 2267
Set up
Sort, 424
SetAccessModeViaWeb, 1577
SetAcousticSignal, 1583
SetAlarmReportingMode, 1570
SetAlarmReportMode, 1570
SetBacklightColor, 1584
SetBit, 1564, 1694
SetBitInTag, 1566, 1695
SetBitWhileKeyPressed, 1568
SetBrightness, 1583
SetConnectionMode, 1576
SetDataRecordTagsToPLC, 1644
SetDataRecordToPLC, 1643
SetDaylightSavingTime, 1520
SetDeviceMode, 1572
SetDisplayMode, 1572
SetLanguage, 1575, 1698
SetPLCMode, 1574, 1575
SetPropertyByConstant, 1679
SetPropertyByProperty, 1678
SetPropertyByTag, 1681
SetPropertyByTagIndirect, 1683
SetPropertyCurrentWindow, 1686
SetPropertyCurrentWindowByProperty, 1685
SetPropertyCurrentWindowByTagIndirect, 1686
SetRecipeTags, 1644
SetScreenKeyboardMode, 1573, 1699
SetTag, 1554, 1663
SetTagByProperty, 1665
SetTagIndirect, 1666
SetTagIndirectByProperty, 1667, 1669
SetTagIndirectByTagIndirect, 1668
SetTagIndirectWithOperatorInputAlarm, 1670
SetTagToHandWheel, 1531
Setting
Deadband, 795, 812
Delay time, 795, 811
Language, 2157, 2236, 2323
Languages in the operating system, 2013
Priority, 812
Style, 125
Time base, 423
Setting language, 2157, 2236, 2323
Setting up, 2269
Settings for tags, 544
SetWebAccess, 1577

SHIFT
Key, 2152, 2230, 2318
Shortcut menu, 1739
Short-term archive list, 935
ShowAlarmWindow, 1615, 1620
ShowBlockInTiaPortalFromAlarm
ShowBlockInTIAPortalFromAlarm, 1700
Showing
Report sections, 1251
ShowLogonDialog, 1550, 1658
ShowOperatorNotes, 1532
ShowPLCCodeViewFromAlarm
ShowPLCCodeViewFromAlarm, 1701
ShowPopupScreen, 1533, 1534
ShowSlideInScreen, 1537
ShowSoftwareVersion, 1533
ShowSystemAlarm, 1621
ShowSystemDiagnosticsWindow, 1615
Signal quality
Calculating the effective range, 512
signature check
Switching off, 2160, 2244
Signing the log backup file, 665, 904
SIMATIC
Differences in data format from Microsoft, 1192
SIMATIC Logon, 1333
SIMATIC Logon, 1322, 2116, 2162, 2247
Certificate, 1335
encrypted communication, 1335
Simple alarm view, 920
Application, 920
Control element, 921
Operation, 921
Operation using the keyboard, 922
Operation using the mouse, 922
simple alarm window
Application, 920
Simple recipe view, 466
Behavior, 1099
Configuring, 466
Layout, 1097
Menu command, 1099
Operation, 1098
Restrictions, 1103
Simple user view
Application, 341
Configuring, 343, 432
Display in Runtime, 342
Layout, 341
Number of lines, 341
SimulateSystemKey, 1522
SimulateTag, 1523
Simulating, 2276
Define temporary start screen, 2177, 2278
Project with tag simulator, 2127, 2175
Simulation
Project:simulate, 2276
Runtime, 2125
Simulation data, 2277
Single alarm
Filtering, 932
Hiding alarms, 883
single-language objects
Runtime language, 2247
Single-line mode
GRAPH overview, 384
Size
Editing multiple objects, 140, 1272
of objects in the report, 1233
Slide-in screen
Configuring, 66
Configuring for devices without multi-touch function, 68
Objects, 65
Use, 69
Slider, 474
Display bar, 475
Display current value, 475
Lock/unlock, 2225, 2313
Maximum value, 474
Minimum value, 474
Sm@rtClient view, 479
Shared use, 480
View only, 480
SmartClientViewConnect, 1611
SmartClientViewDisconnect, 1611
SmartClientViewLeave, 1612
SmartClientViewReadOnlyOff, 1610
SmartClientViewReadOnlyOn, 1610
SmartClientViewRefresh, 1609
SmartServer, 1527
Softkey, 241, 245
global, 243
Software
HMI device, 2334, 2335, 2336
Sort
Set up, 424
Sort order
Sorting dialog, 1214
Sorting
Alarm view, 427
Report page, 1250
Sorting alarms, 933
SQL, 1168

SQL queries
to WinCC databases, 1061
SQL Server Import/Export-Wizard, 1061
SQL statements for filtering alarms, 960
Stamping, 138, 1265
Standard SQL queries
to WinCC databases, 1061
Start
Logging, 634
Start Angle
Circle segment, 405
Circular arc,
Ellipse segment,
Elliptical arc,
Start screen, 50, 51, 302, 2115, 2161, 2246
Start sequence, 2247
Start value
Tag, 549, 550
Start value of danger range
Gauge, 515
Start value of warning range
Gauge, 515
Starting
Runtime at Engineering Station, 2199, 2304
Runtime on a panel, 2140, 2201
Runtime on PC, 2201
Runtime on the configuration PC, 2199, 2304
Starting point
Defining, 598
Starting point of logging, 640
StartLogging, 1543
StartNextLog, 1544
StartProgram, 1524
StartSequenceLog, 1544
Startup language
determine:Startup language, 630, 897
Startup list, 2247, 2258
Setting up, 2258
Static time range, 695
Statistics area column name property changed, 1733
Status bar, 423, 706, 719, 725
Recipe view, 1205
Table view, 751
Trend view, 751
Status bar element property changed, 1733
Status changed, 1748
Status display
Channel diagnostics display, 400
Status tag, 778
Alarm group, 850
StatusForceGetValues, 1526
StatusForceSetValues, 1527
StepBack, 1747
StepForward, 1747
Stop, 1748
StopLogging, 1545
StopRuntime, 1528, 1655
Storage location
Database, 1032
File - CSV (ASCII), 1031
File - RDB, 1032
File - TXT (Unicode), 1031
Logs with checksum, 1032
Storage medium
Restoring from external, 2191, 2344
Storing
External graphic, 149
Style
Changing, 83
Creating a new type, 1960
Define, 81
Deleting, 82
Duplicate, 82
Edit, 83
HMI device, 2115, 2161
Setting, 125
Style sheet
Copying, 91
Creating, 88
Creating a new type, 1961
Deleting, 90
Editing, 90
Use, 92
Subdivision
Trend view, 702
Substitute value, 569
Defining, 569
Swapping, 904
Alarm log, 904
Signing off alarm log, 904
Swapping out, 648
of process values, 648
Swapping out of process values, 648
Switch, 471
Type, 471
Switch OFF, 1751
Switch ON, 1752
Symbol
Alarm classes, 836
Symbol library, 483
Fill style, 484
Fixed aspect ratio, 484
Flip, 484

OP 177B mono, 484
Rotate, 484
TP 177B mono, 484
Symbol table
PLC code view, 1875
Symbolic addressing
of a tag, 527
Symbolic I/O field, 485, 487
application in reports, 1306
Mode, 485, 487
Text list, 186, 188, 486, 487
Symbols
ProDiag overview, 1863
Synchronization
Redundancy, 2248
Synchronization of tags, 2116, 2162
Synchronize, 523, 525, 544
Synchronizing
Object in user-defined functions, 1386
Recipe data record, 1116, 1129
Recipe tag, 1086
Recipe view and recipe screen, 1082
Tag in user-defined functions, 1386
Synchronizing a PLC tag, 546
Syntax highlighting
User-defined function, 1386
System alarm, 764
Meaning, 965, 974, 982
System block, 957
Configuring, 807
System blocks, 806
System diagnostic alarm, 1829, 1840
System diagnostic view
Detail view, 489
Device view, 489
Diagnostic buffer view, 489
System diagnostics, 1821, 1824, 1837
Button, 1832
Detail view, 1821, 1825, 1838
Device view, 1821, 1825, 1838
Diagnostic buffer view, 1822, 1825, 1838
Settings in the PLC, 1828, 1840
System diagnostics view, 1821, 1824, 1837
System diagnostics window, 1824, 1831, 1837
System diagnostics indicator, 1837
Button as system diagnostics indicator, 1832
Inserting, 1829, 1847
System diagnostics window, 1829, 1847
System diagnostics view, 1824
Configuring, 1833, 1834, 1835, 1841, 1842
Icon, 491, 494
Layout, 490
Settings in the PLC, 1828, 1840
Show split view, 491
Single display,
System diagnostics indicator,
System diagnostics window, 1824, 1831
Configuring, 1831
Layout, 496
System event, 759, 760, 761, 764
Editing, 802, 825
Meaning, 962, 963, 964, 966, 969, 972, 973, 974, 975, 978, 979, 980, 981, 982, 983, 984, 985, 987, 988, 991, 992, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1007, 1008, 1009, 1010, 1011, 1012, 1015, 1017, 1018
Parameters, 962
System events, 2116, 2162
Configuring, 783, 784
System function, 1371
Application, 1373, 1374
DirectKeyScreenNumber, 1512
HMI device dependency, 1388
HTMLBrowserScrollLeft, 1588
In function list, 1373
in user-defined functions, 1373, 1438
Language dependency in function lists, 1374
SafelyRemoveHardware, 1515
Sequence is not adhered to, 1440
System functions
AcknowledgeAlarm, 1620
ActivateCleanScreen, 1647
ActivatePLCCodeView, 1603
ActivatePreviousScreen, 1507
ActivateScreen, 1505, 1673
ActivateScreenByNumber, 1506
ActivateScreenInCurrentScreenWindow, 1674
ActivateScreenInScreenWindow, 1674
ActivateSystemDiagnosticsView, 1612
AdjustContrast, 1581
AlarmViewAcknowledgeAlarm, 1619
AlarmViewEditAlarm, 1618
AlarmViewShowOperatorNotes, 1619
AlarmViewUpdate, 1618
ArchiveLogFile, 1538
Available for WinCC Runtime, 1498, 1503
Available on Comfort Panels, 1484
Available on mobile panels, 1491
BackupRAMFileSystem, 1522
CalibrateTouchScreen, 1582
ChangeConnection, 1578
ChangeConnectionEIP, 1579
ClearAlarmBuffer, 1616
ClearAlarmBufferProTool, 1617

ClearDataRecord, 1635
ClearDataRecordMemory, 1636
ClearLog, 1541
CloseAllLogs, 1542
ControlSmartServer, 1527
ControlWebServer, 1528
DecreaseFocusedValue, 1531
DecreaseTag, 1556, 1671
DirectKey, 1510
EditAlarm, 1616
Encode, 1508
EncodeEx, 1509
ExportDataRecords, 1622
ExportDataRecordsWithChecksum, 1624
ExportImportUserAdministration, 1547, 1657
GetBrightness, 1585
GetDataRecordFromPLC, 1631
GetDataRecordName, 1632
GetDataRecordTagsFromPLC, 1634
GetGroupNumber, 1549
GetPassword, 1549
GetUserName, 1548
GoToEnd, 1655
GoToHome, 1654
HTMLBrowerZoomIn, 1587
HTMLBrowserBack, 1591
HTMLBrowserForward, 1590
HTMLBrowserHome, 1590
HTMLBrowserPageDown, 1589
HTMLBrowserPageUp, 1589
HTMLBrowserRefresh, 1586
HTMLBrowserScrollDown, 1586
HTMLBrowserScrollRight, 1588
HTMLBrowserScrollUp, 1587
HTMLBrowserStop, 1585
HTMLBrowserZoomOut, 1588
ImportDataRecords, 1627
ImportDataRecordsWithChecksum, 1628
IncreaseFocusedValue, 1514
IncreaseTag, 1551, 1659
InvertBit, 1556, 1687
InvertBitInTag, 1558, 1689
InvertLinearScaling, 1552, 1660
LinearScaling, 1553, 1662
LoadDataRecord, 1630
Logoff, 1546
Logon, 1546
LogTag, 1539
LookupText, 1529, 1656
NotifyUserAction, 1513
OpenAllLogs, 1542
OpenCommandPrompt, 1516
OpenControlPanel, 1518
OpenControlPanelDialog, 1515
OpenInternetExplorer, 1517
OpenScreenKeyboard, 1649
OpenTaskManager, 1649
PageDown, 1653
PageUp, 1654
PDFFitToHeight, 1597
PDFFitToWidth, 1597
PDFGotoFirstPage, 1598
PDFGotoLastPage, 1598
PDFGotoNextPage, 1599
PDFGotoPage, 1599
PDFGotoPreviousPage, 1600
PDFScrollDown, 1596
PDFScrollLeft, 1601
PDFScrollRight, 1601
PDFScrollUp, 1596
PDFZoomIn, 1600
PDFZoomOriginal, 1602
PDFZoomOut, 1600
PLCCodeViewActualOrTriggerValues, 1607
PLCCodeViewDetails, 1605
PLCCodeViewMoveScrollBar, 1604
PLCCodeViewNextNetwork, 1605
PLCCodeViewPreviousNetwork, 1609
PLCCodeViewStepMode, 1606
PLCCodeViewSymbols, 1606
PLCCodeViewTransitionInterlock, 1607
PLCCodeViewZoomIn, 1608
PLCCodeViewZoomOut, 1608
PressButton, 1521
PrintReport, 1570
PrintScreen, 1569
RecipeViewBack, 1642
RecipeViewClearDataRecord, 1638
RecipeViewGetDataRecordFromPLC, 1637
RecipeViewMenu, 1638
RecipeViewNewDataRecord, 1637
RecipeViewOpen, 1639
RecipeViewRenameDataRecord, 1641
RecipeViewSaveAsDataRecord, 1640
RecipeViewSaveDataRecord, 1640
RecipeViewSetDataRecordToPLC, 1639
RecipeViewShowOperatorNotes, 1642
RecipeViewSynchronizeDataRecordWithTags, 1641
ReleaseButton, 1521
ResetBit, 1560, 1690
ResetBitInTag, 1561, 1692
ResetTagToHandWheel, 1530
SaveDataRecord, 1646

ScreenObjectCursorDown, 1650
ScreenObjectCursorLeft, 1653
ScreenObjectCursorRight, 1651
ScreenObjectCursorUp, 1651
ScreenObjectPageDown, 1652
ScreenObjectPageUp, 1652
SendEMail, 1519
SetAcousticSignal, 1583
SetAlarmReportMode, 1570
SetBit, 1564, 1694
SetBitInTag, 1566, 1695
SetBitWhileKeyPressed, 1568
SetBrightness, 1583
SetConnectionMode, 1576
SetDataRecordTagsToPLC, 1644
SetDataRecordToPLC, 1643
SetDaylightSavingTime, 1520
SetDeviceMode, 1572
SetDisplayMode, 1572
SetLanguage, 1575, 1698
SetPropertyByConstant, 1679
SetPropertyByProperty, 1678
SetPropertyByTag, 1681
SetPropertyByTagIndirect, 1683
SetPropertyCurrentWindow, 1686
SetPropertyCurrentWindowByProperty, 1685
SetPropertyCurrentWindowByTagIndirect, 1686
SetRecipeTags, 1644
SetScreenKeyboardMode, 1573, 1699
SetTag, 1554, 1663
SetTagByProperty, 1665
SetTagIndirect, 1666
SetTagIndirectByProperty, 1667, 1669
SetTagIndirectByTagIndirect, 1668
SetTagToHandWheel, 1531
SetTagWithOperatorInputAlarm, 1670
SetWebAccess, 1577
ShowAlarmWindow, 1620
ShowBlockInTiaPortalFromAlarm, 1700
ShowLogonDialog, 1550, 1658
ShowOperatorNotes, 1532
ShowPLCCodeViewFromAlarm, 1701
ShowPopupScreen, 1533, 1534
ShowSlideInScreen, 1537
ShowSoftwareVersion, 1533
ShowSystemAlarm, 1621
ShowSystemDiagnosticsWindow, 1615
SimulateSystemKey, 1522
SimulateTag, 1523
SmartClientViewConnect, 1611
SmartClientViewDisconnect, 1611
SmartClientViewLeave, 1612
SmartClientViewReadOnlyOff, 1610
SmartClientViewReadOnlyOn, 1610
SmartClientViewRefresh, 1609
StartLogging, 1543
StartNextLog, 1544
StartProgram, 1524
StatusForceGetValues, 1526
StatusForceSetValues, 1527
StopLogging, 1545
StopRuntime, 1528, 1655
SystemDiagnosticViewBack, 1615
SystemDiagnosticViewDetailView, 1613
SystemDiagnosticViewDeviceView, 1614
SystemDiagnosticViewDiagnosticBuffer, 1614
SystemDiagnosticViewUpdatePlcBuffer, 1613
TerminatePROFIsafe, 1518
TraceUserChange, 1550
TrendViewBackToBeginning, 1595
TrendViewCompress, 1592
TrendViewExtend, 1592
TrendViewRulerLeft, 1594
TrendViewRulerRight, 1593
TrendViewScrollBack, 1592
TrendViewScrollForward, 1591
TrendViewSetRulerMode, 1594
TrendViewStartStop, 1595
UpdateTag, 1508
System limits
HMI device, 2361, 2364, 2368, 2374
System print job
Changing, 1260
Deleting, 1261
System report
Changing, 1260
Deleting, 1261
System status, 860, 1023
System-defined controller alarm, 765
System-defined controller alarms, 760, 761
SystemDiagnosticViewBack, 1615
SystemDiagnosticViewDetailView, 1613
SystemDiagnosticViewDeviceView, 1614
SystemDiagnosticViewDiagnosticBuffer, 1614
SystemDiagnosticViewUpdatePlcBuffer, 1613

T

T piece, 502
Rotation, 502
TAB key, 2150, 2228, 2316
Table
Configuring table elements, 714, 727
Table view, (See table view)

Table column
Configuring the sorting, 716, 728
Table view, 497, 690
Buttons, 731
Changing the data connection in Runtime, 744
Changing the time range in Runtime, 745
Configuring, 711
Configuring a data source, 498
Configuring selection, 715
Configuring the data source, 704, 717
Configuring the export of Runtime data, 707, 720, 728
Configuring the value column, 713
Dynamizing value column, 500
Editing the table field in Runtime, 746
Elements of the status bar, 751
Hide column in Runtime, 745
Hide columns in Runtime, 747
Move columns in Runtime, 747
Online configuration, 708, 721, 730, 744, 747
Opening the configuration dialog in Runtime, 744
Operation in Runtime, 731, 735
Scrolling through archived data, 750
Show column in Runtime, 745
Show columns in Runtime, 747
Status bar, 706, 719, 725
Toolbar, 499, 698, 706, 719, 725, 731
Use in reports, 1306
Table:Designing a border, 131
Tables
in reports, 1241
Tag, 635
absolute addressing, 526
Access in user-defined function, 1390
Access the local script, 1390
Acquisition cycle, 567, 568, 593
Acquisition mode, 567
Addressing, 531
Changing the PLC, 551
Coding, 602
Comment, 533
Configuration, 540
Configuring several tags, 541
Connection to PLC, 531
Copy, 539
Creating external tags, 531
Creating internal tags, 533
Creating the user data type, 586
Data log, 618, 623
Data type, 533, 536
Defining a substitute value, 569
Defining acquisition cycle, 570
Deleting, 539
Displaying values, 676
Event, 566
Exporting, 1986
external, 635
External tag, 522, 524
Faceplate type, 296
Illuminated pushbutton, 412
Importing, 1988, 1989
in Runtime, 567
Index tag, 564, 565
Indirect addressing, 564, 565, 572
internal, 635
Internal tag, 528, 529
Key switch, 476
Length, 533
Limit value, 548
Limits, 547
Linear scaling, 551, 553
Local, 1392
Logging, 618, 623
Logging cycle, 593, 596
Multiplexing, 564
Name, 533
Object list, 532
Output in alarm, 800
R, 1049
Raw data tag, 636
Read continuously, 567
Reconnecting, 546
Rename, 539
RFID tag, 1758
Start value, 549, 550
symbolic addressing, 527
Tag type, 536
Tolerance band, 618, 623
Transponder, 1758
Update, 567, 568
User data type, 585
User data type element, 585
Value range, 602
Tag array, 575, 581
Tag binding
Animation, 221
Tag count with arrays, 576
Tag counting with arrays, 582
Tag data
Structure for the import, 1990, 1993, 1998
Tag import
Tag data structure, 1990, 1998
Tag list
Indirect addressing, 564, 565

Tag name
Valid names in logs, 653
Tag simulator, 2128, 2177
Tag synchronization, 2248
Tag table
Default, 520
for HMI devices, 520
user-defined, 520
Tag triggers, 1798, 1812
Tag type, 536
Tag value
Output, 676, 680
Updating, 525
Tags
Alarm group, 778
Basics, 519
Editing, 540
Task, 1794
Changing during Runtime, 1816
Changing the name, 1807
Creating for the automatic start of a print job, 1258
Deactivate, 1796, 1797
Deleting, 1808
Service project, 2267
Task card, 42
Toolbox,
Tools,
Task switching, 2116, 2162
Teach-in mode, 1087
Recipe tag, 1087
Telegram tag, 636
Template, 241
Copy, 58
Creating, 56
Deleting, 58
Global screen, 56
Inserting, 58
Move, 58
Rename, 58
Template,
Use in screen, 60
TerminatePROFIsafe, 1518
Testing, 1424
User-defined function, 1393
Testing:with the Debugger, 1424
Text
Acknowledgment group, 2116, 2162
Button, 472
Define text format, 423
Editable text field, 363
Text field, 501, 1307
Text field, 501, 1307
Size, 501, 1308
Text field length
Asian languages, 2037
Text list
Application, 176
Bit (0, 1), 183
Bit number (0 - 31),
Creating, 178
Exporting, 1995
Importing, 1997
Output in alarm, 801
Range (... - ...), 182
Symbolic I/O field, 186, 188, 486, 487
Value/Range, 180
Text lists
Bit selection, 2115, 2161
Use in recipe data records, 1092
Textual cross-references
show, 2004
VB scripts, 2005
Time axis
Configuring, 701
Time base
Setting, 423
Time column property changed, 1735
Time expired, 1755
Time range, 695
Dynamic, 695
static, 695
Time stamp, 781
Time-driven output
Report, 1252
Timer, 1797
Acyclic triggers, 1797
Cyclic triggers, 1797, 1809
TIMESTEP, 1049
Title page
in reports, 1237
Report, 1228, 1230
Toggle, 1751
Between Runtime languages, 2030
Key, 2152, 2230, 2318
Toggle toolbar, 1748
Tolerance band
Tags, 618, 623
Toolbar, 369, 372, 376, 499, 706, 719, 725
Button with screen change:, 322
Configuring a system function, 322
Configuring an event, 314
Creating, 312
customized, 305, 318

Order, 104, 115, 1268
PDF view, 441
PLC code view, 447, 451, 1878
Recipe view, 1205
Table view, 698
Toolbar button property changed, 1733
Toolbox, 93, 97
Tools, 102
for the design of reports, 1240
Tooltip
Show limits, 2116, 2162
TP 177B mono
Symbol library, 484
TraceUserChange, 1550
Transfer
License key to HMI device, 2346
Transferring recipe values
between HMI device and PLC, 1188
Translate
Editor, 2018
Transmission protocol, 395
Transparency
In graphic, 147
transparent color, 379
Display on panels, 379
Transponder, 1758
Trend, 678
Common axes, 697
Configuring, 704
Multiple axes, 696
Representation formats, 694
Text orientation, 695
Trend curve, 695
Trend display
Resolution, 691
Trend property changed, 1732
Trend view, 408, 678, 692
Button, 409
Buttons, 731
Changing the order of trends, 741
Configure trend, 704
Configuring, 699
Configuring for logging, 683
Configuring for values from the PLC, 676, 681
Configuring the data source, 704, 717
Configuring the export of Runtime data, 707, 720, 728
Configuring the value axis, 702
Configuring the X axis, 702
Configuring the Y axis, 702
Configuring time axis, 701
Elements of the status bar, 751
Moving a trend area in Runtime, 741
Number of trends, 691
Online configuration, 747
Operation in Runtime, 731, 735
Range proportion, 703
Representation formats, 694
Resolution of Trend Display, 691
Scrolling through archived data, 750
Status bar, 706, 719, 725
Subdivision, 702
Text orientation, 695
Toolbar, 706, 719, 725, 731
Trend curve, 695
Trend in the foreground, 741
Zooming in on ~ area in Runtime,
Zooming in on a trends in Runtime, 740
Zooming in on the time axis in Runtime, 740
Zooming in on the value axis in runtime, 740
Zooming out on a trend in Runtime, 740
Trends
in reports, 1241
TrendViewBackToBeginning, 1595
TrendViewCompress, 1592
TrendViewExtend, 1592
TrendViewRulerLeft, 1594
TrendViewRulerRight, 1593
TrendViewScrollBack, 1592
TrendViewScrollForward, 1591
TrendViewSetRulerMode, 1594
TrendViewStartStop, 1595
Trigger mode, 640
Trigger tag, 782
Trigger user alarm
Configuring, 821
Triggering user alarm
Example, 821
Triggers, 1795
Acyclic, 1796, 1803, 1811
Changing, 1808
Cyclic, 1796, 1804, 1813, 1816, 1818
Event trigger, 1796, 1797, 1806, 1815
Standard cycle, 1796
Tag triggers, 1798, 1812
Two-hand operation
Multi-touch devices, 2224, 2313
Type
Library, 1938

U

Uninstalling
Option, 2347

Unlock
Operator control, 2225, 2313
User, 1348
Update
Operating system of the HMI device (Windows CE), 2342
Tag, 567, 568
Update cycle, 593, 2247
UpdateTag, 1508
Updating
Faceplate type, 287
Tag, 525
Updating a library, 1948
Updating the device version, 2341
Updating the firmware, 2341
Updating the operating system, 2341
Updating types to the latest version, 1948
Upgrade
Global library, 2099
Project, 2097
Project version, 2097
URL, 394
USB
Download, 2190
Loading, 2194
USB driver, 2194
Installation under Windows 7, 2194
USB port, 2194
Use, 1079
Faceplate type, 290
Key switch, 476
Local script, 1419
Of recipes, 1079
ProDiag overview, 456
Project-wide alarm class, 789
Recipe view, 465
Screen navigation function keys, 254
Slide-in screen, 69
Style sheet, 92
User view, 339
Use global assignment
Function key, 55
Use WebNavigator license for WebUX, 2247
User
Changing, 1346
Deleting in runtime, 1347
Dynamic logon, 1332
Logon, 1348
Unlock, 1348
User administration, 1309, 2116, 2162, 2247
Central user administration, 1333
exporting, 1352
importing, 1353
Object with access protection, 1356, 1357
Runtime settings, 50, 52, 1316, 1319, 1322, 1331, 1332, 1337, 1338, 1347, 1348
Setting up, 1362
SIMATIC Logon, 1333
User alarm, 763
Parameter output, 820
User alarm parameter
Configuring, 822
User archive, 1059, 1065
Displaying data, 1065
Querying archive data, 1059
User change, 1729, 1814
User data
Backup, 1311
Restoring, 1311
User data type, 585
Assigning to a screen, 141
Create, 586
Creating, 585
Creating a user data type element, 586
Deleting, 590
Editing, 589
Release, 589
Tags, 585
User data type element
Create, 586
User data types
Interconnecting with faceplates, 274
User group
Administer authorizations, 1329
Assigning, 1368
Assigning users, 1328
Change displayed name, 1330
Changing in runtime, 1347
Changing the name, 1330
Creating, 1325, 1366
Deleting, 1330
Managing, 1329
Unauthorized, 1350
User interface language, 2012
Selecting, 2015
User log
Recipe view, 1214
UserArchiveControl, 1212
User rights
Hierarchy levels, 2116, 2162
User text block
Configuring, 807
User text blocks, 806

User view, 1342
Column width, 340
Columns moveable, 343
Complex user view, 342, 1342
Configuring, 1343
Layout, 339
Moving columns, 340
Number of lines, 340, 343
Simple user view, 343, 432
Use, 339
UserArchiveControl, 1202
Configuring, 1209
Edit data, 1212
Online configuration, 1187
Runtime data, 1186
User-defined function, 1371, 1375
Adapting display properties, 1386
Autocompletion, 1384
Changing object property, 1442
Code templates, 1385
configuring in C, 1409
configuring with VBS, 1398
Entering parameters, 1384
Executing in Runtime, 1438
Opening know-how protected functions, 1416
Parameter transfer, 1389
Return value in C, 1414
Return value in VBS, 1402
Sequence is not adhered to, 1440
Synchronizing object, 1386
Synchronizing tag, 1386
Syntax highlighting, 1386
Temperature conversion, 1443
Testing, 1393
Using a system function, 1438
Users
Assigning a user group, 1328
Changing the name, 1328
Create, 1326
Creating, 1368
Creating in runtime, 1344
Deleting, 1329
Logging logons, 1361
Logon, 1350
Managing, 1328, 1346
Updating after change of user, 1814
Using
Local script, 1419
Using tags, 351
Using the recipe in Runtime
Access protection for control tags, 1190
Using user data types (UDTs)
in faceplates, 273

V

Validity range, 1406
Value axis
Configuring, 702
Value change, 1754
Value column
Configuring, 713
Value column property changed, 1734
Value display
Configuring the export of Runtime data, 707, 720, 728
Status bar, 706, 719, 725
Toolbar, 706, 719, 725
Value range
Tag coding, 602
Value table, 723
Configuring, 725
Display modes, 724
Displaying data in the statistics area window, 748
Displaying the coordinates of a point, 742
Online configuration, 708, 721, 730
Operation in Runtime, 736
Operator controls, 506
Value/Range
Text list, 180
VBS
Examples in WinCC, 1451
VBScript:debugging, 1424
Versions in WinCC, 2094, 2096
Vertical movement
Animation, 216
Vertical radius, 364
Ellipse segment, 366
Elliptical arc, 365
View
Purpose, 1160
Visible column, 345
Volume, 1739

W

Watch table, 344
Operator control, 345
Sequence of columns, 345
StatusForceGetValues, 345
StatusForceSetValues, 345
Visible column, 345

Watermarks
Report, 1230
WChar, 554
Web access
Alarms in the event view, 2247
Appearance, 2247
Local user groups, 2247
Server utilization, 2247
Web address, 389, 394
Web browser, 2237
WebClient
Setting start screen and language for users, 1340
Setting the start screen and language for user groups, 1339
WebKit engine, 2237
WebNavigator
Setting start screen and language for users, 1340
Setting the start screen and language for user groups, 1339
WebNavigator license, 2247
WebNavigator licenses
for WebUX, 1341
WebUX
Configure number of reserved licenses for users, 1340
Setting start screen and language for users, 1340
Setting the start screen and language for user groups, 1339
WebUX license, 2247
When dialog is closed, 1729
When dialog is opened, 1728
WinCC
Starting Runtime, 2304
WinCC Archive Connector, 1073
WinCC DataConnector, 1073
WinCC OLE DB, 1071
Basics, 1071
Microsoft, 1071
WinCC, 1071
WinCC OLE DB Provider, 1040, 1073
Access to archive data, 1040
Linked Server, 1073
SQL Server, 1073
WinCC Runtime
Available system functions, 1498
WinCC Runtime Advanced
Display and operating object, 331
WinCC system events, 1000
WinCC UserArchiveControl, (See UserArchiveControl)
WinCC version
Compatibility, 2095, 2096
WinCC Runtime
Available system functions, 1503
Window, 1436
Properties, 2246
Window controller, 2247
Wizard
Device wizard, 1929, 1932
WLAN area, 1757
WLAN reception, 512
Layout, 512
Work area
Effective range, 1767
Effective range (RFID), 1769
Mobile Wireless, 1767
Mobile Wireless (RFID), 1769
Zone, 1764, 1765
Working step
to create screens, 45, 46
WString, 554

X

X axis
Configuring, 702
X-axis property changed, 1734
xlsx file, 1972, 1974, 1988, 1989, 1995, 1996
Importing tags, 1987, 1989
Preparing for the import, 1987, 1989

Y

Y axis
Configuring, 702
Y-axis property changed, 1734

Z

Zone, 1758, 1763
Calculating quality, 518
Configuring a screen change, 1778, 1782
Displaying an object in relation to the zone, 1779
ID Zone, 1779
On entry, 517
Work area, 1764, 1765
Zone ID, 517
Zone ID / connection point ID, 1792
Zone name, 516
Zone signal, 517
Zones editor, 1763
Zoom functions, 2247
Setting up, 2256

Zooming
Multi-touch devices, 2222
Screen, 43

SIEMENS

Using technology functions

Programming and Operating Manual

Counting, measurement and position input 1

Cam control (S7-300, S7-400) 2

PID control 3

Time-based IO (S7-1500) 4

Pulse output (S7-1500) 5

Motion Control 6

SIMATIC Ident 7

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Counting, measurement and position input....11

1.1 Counting, measurement and position input (S7-1500) (S7-1500)....11
1.1.1 The basics of counting, measurement and position input (S7-1500)....11
1.1.1.1 Overview of modules and properties (S7-1500)....11
1.1.1.2 Basics of counting, measuring and position input (TM Count, TM PosInput, Compact CPU) (S7-1500)....15
1.1.1.3 Basics of counting (TM Timer DIDQ) (S7-1500)....68
1.1.1.4 Basics of counting (digital modules) (S7-1500)....72
1.1.2 Use technology object High_Speed_Counter (S7-1500)....80
1.1.2.1 Convention (S7-1500)....80
1.1.2.2 High_Speed_Counter technology object (S7-1500)....80
1.1.2.3 Overview of the configuration steps (S7-1500)....80
1.1.2.4 Add technology object (S7-1500)....81
1.1.2.5 Configuring the High_Speed_Counter (S7-1500)....82
1.1.2.6 Programming the High_Speed_Counter (S7-1500)....101
1.1.2.7 Commissioning the High_Speed_Counter (S7-1500)....103
1.1.2.8 High_Speed_Counter diagnostics (S7-1500)....104
1.1.3 Using technology object SSI_Absolute_Encoder (S7-1500)....105
1.1.3.1 Technology object SSI_Absolute_Encoder (S7-1500)....105
1.1.3.2 Overview of the configuration steps (S7-1500)....106
1.1.3.3 Add technology object (S7-1500)....106
1.1.3.4 Configuring SSI_Absolute_Encoder (S7-1500)....108
1.1.3.5 SSI_Absolute_Encoder programming (S7-1500)....122
1.1.3.6 Commissioning SSI_Absolute_Encoder (S7-1500)....123
1.1.3.7 SSI_Absolute_Encoder diagnostics (S7-1500)....124
1.1.4 Using the module (S7-1500)....126
1.1.4.1 Using the technology module (S7-1500)....126
1.1.4.2 Using the digital module (S7-1500)....173

1.2 Counting and measurement (S7-300, S7-400)....179
1.2.1 Counting and measuring (S7-300, S7-400)....179
1.2.1.1 Counting and measuring basics (S7-300, S7-400)....179
1.2.1.2 Using FM 350-1 (S7-300, S7-400)....208
1.2.1.3 Using FM 350-2 (S7-300, S7-400)....260
1.2.1.4 Using FM 450-1 (S7-300, S7-400)....287
1.2.1.5 Using ET 200S 1Count24V (S7-300, S7-400)....316
1.2.1.6 Using ET 200S 1Count5V (S7-300, S7-400)....376

2 Cam control (S7-300, S7-400)....439

2.1 Cam control (S7-300, S7-400)....439
2.1.1 Cam control basics (S7-300, S7-400)....439
2.1.1.1 What is cam control? (S7-300, S7-400)....439
2.1.1.2 Properties of the cam types (S7-300, S7-400)....439
2.1.1.3 Normal tracks and track results (S7-300, S7-400)....441
2.1.1.4 Special tracks (S7-300, S7-400)....442
2.1.1.5 Hysteresis (S7-300, S7-400)....444

2.1.1.6 Dynamic adjustment (S7-300, S7-400)....445
2.1.1.7 Interfaces of the cam controller (S7-300, S7-400)....446
2.1.1.8 Encoder signals (S7-300, S7-400)....447
2.1.2 Functions of the FM x52 (S7-300, S7-400)....453
2.1.2.1 Writing and enabling machine data (S7-300, S7-400)....453
2.1.2.2 Reading machine data (S7-300, S7-400)....455
2.1.2.3 Writing cam data (S7-300, S7-400)....455
2.1.2.4 Reading cam data (S7-300, S7-400)....456
2.1.3 Using FM 352 (S7-300, S7-400)....457
2.1.3.1 Terminal assignment of FM 352 front connector (S7-300, S7-400)....457
2.1.3.2 Cam control functions (S7-300, S7-400)....460
2.1.3.3 Configuring FM 352 (S7-300, S7-400)....483
2.1.3.4 Configuring FM 352 (S7-300, S7-400)....483
2.1.3.5 Commissioning the FM 352 (S7-300, S7-400)....517
2.1.3.6 Using FM 352 from the user program (S7-300, S7-400)....517
2.1.3.7 FM 352: Online & Diagnostics (S7-300, S7-400)....518
2.1.4 Using FM 452 (S7-300, S7-400)....533
2.1.4.1 Terminal assignment of the front connector of FM 452 (S7-300, S7-400)....533
2.1.4.2 Cam control functions (S7-300, S7-400)....535
2.1.4.3 Configuring FM 452 (S7-300, S7-400)....559
2.1.4.4 Configuring FM 452 (S7-300, S7-400)....559
2.1.4.5 Commissioning the FM 452 (S7-300, S7-400)....594
2.1.4.6 Using FM 452 from the user program (S7-300, S7-400)....594
2.1.4.7 FM 452: Online & Diagnostics (S7-300, S7-400)....595

3 PID control....611

3.1 Principles for control....611
3.1.1 Controlled system and actuators....611
3.1.2 Controlled systems....612
3.1.3 Characteristic values of the control section....614
3.1.4 Pulse controller....616
3.1.5 Response to setpoint changes and disturbances....620
3.1.6 Control Response at Different Feedback Structures....621
3.1.7 Selection of the controller structure for specified controlled systems....629
3.1.8 PID parameter settings....630

3.2 Configuring a software controller....631
3.2.1 Overview of software controller....631
3.2.2 Steps for the configuration of a software controller....632
3.2.3 Add technology objects....633
3.2.4 Configure technology objects....634
3.2.5 Call instruction in the user program....635
3.2.6 Downloading technology objects to device....636
3.2.7 Commissioning software controller....637
3.2.8 Save optimized PID parameter in the project....637
3.2.9 Comparing values....638
3.2.9.1 Comparison display and boundary conditions....638
3.2.9.2 Comparing values....639
3.2.10 Parameter view....641
3.2.10.1 Introduction to the parameter view....641
3.2.10.2 Structure of the parameter view....643
3.2.10.3 Opening the parameter view....645
3.2.10.4 Default setting of the parameter view....646

3.2.10.5 Working with the parameter view....648
3.2.11 Display instance DB of a technology object....659

3.3 PID control (S7-1200, S7-1500) (S7-1200, S7-1500)....660
3.3.1 Using PID_Compact (S7-1200, S7-1500)....660
3.3.1.1 Technology object PID_Compact (S7-1200, S7-1500)....660
3.3.1.2 PID_Compact V2 (S7-1200, S7-1500)....661
3.3.1.3 PID_Compact V1 (S7-1200, S7-1500)....680
3.3.2 Using PID_3Step (S7-1200, S7-1500)....697
3.3.2.1 Technology object PID_3Step (S7-1200, S7-1500)....697
3.3.2.2 PID_3Step V2 (S7-1200, S7-1500)....698
3.3.2.3 PID_3Step V1 (S7-1200, S7-1500)....717
3.3.3 Using PID_Temp (S7-1200, S7-1500)....734
3.3.3.1 Technology object PID_Temp (S7-1200, S7-1500)....734
3.3.3.2 Configuring PID_Temp (S7-1200, S7-1500)....735
3.3.3.3 Commissioning PID_Temp (S7-1200, S7-1500)....755
3.3.3.4 Cascade control with PID_Temp (S7-1200, S7-1500)....763
3.3.3.5 Multi-zone controlling with PID_Temp (S7-1200, S7-1500)....769
3.3.3.6 Override control with PID_Temp (S7-1200, S7-1500)....772
3.3.3.7 Simulating PID_Temp with PLCSIM (S7-1200, S7-1500)....776

3.4 PID control (S7-300, S7-400, S7-1500) (S7-300, S7-400, S7-1500)....777
3.4.1 Using PID basic functions (S7-300, S7-400, S7-1500)....777
3.4.1.1 CONT_C (S7-300, S7-400, S7-1500)....777
3.4.1.2 CONT_S (S7-300, S7-400, S7-1500)....780
3.4.1.3 TCONT_CP (S7-300, S7-400, S7-1500)....783
3.4.1.4 TCONT_S (S7-300, S7-400, S7-1500)....803

3.5 PID control (S7-300, S7-400) (S7-300, S7-400)....806
3.5.1 Using standard PID control (S7-300, S7-400)....806
3.5.1.1 Using PID_CP (S7-300, S7-400)....806
3.5.1.2 Using PID_ES (S7-300, S7-400)....853
3.5.2 Using modular PID control (S7-300, S7-400)....899
3.5.2.1 Scope of Modular PID Control functions (S7-300, S7-400)....899
3.5.2.2 Basic steps for setting up a controller FB (S7-300, S7-400)....900
3.5.2.3 Programming the controller FB (S7-300, S7-400)....901
3.5.2.4 Complex controller structures (S7-300, S7-400)....917
3.5.2.5 Commissioning the controller FB (S7-300, S7-400)....929
3.5.3 Using PID Self Tuner (S7-300, S7-400)....930
3.5.3.1 Tuning continuous controllers and pulse controllers (S7-300, S7-400)....930
3.5.3.2 Tuning step controllers (S7-300, S7-400)....938
3.5.4 Basics on controller modules (S7-300, S7-400)....945
3.5.4.1 Inputs (S7-300, S7-400)....945
3.5.4.2 Error signal (S7-300, S7-400)....950
3.5.4.3 Control algorithm (S7-300, S7-400)....955
3.5.4.4 Controller output (S7-300, S7-400)....970
3.5.4.5 Shared Device support (S7-300, S7-400)....982
3.5.5 Using FM 355 (S7-300, S7-400)....983
3.5.5.1 Configuring FM 355 (S7-300, S7-400)....983
3.5.5.2 Parameter assignment FM 355 (S7-300, S7-400)....983
3.5.5.3 Programming FM 355 (S7-300, S7-400)....998
3.5.5.4 Commissioning the FM 355 (S7-300, S7-400)....998
3.5.5.5 FM 355: Diagnostics with LED display (S7-300, S7-400)....1008
3.5.5.6 Example projects for FM 355 (S7-300, S7-400)....1015

3.5.6 Using FM 355-2 (S7-300, S7-400)....1015
3.5.6.1 Configuring FM 355-2 (S7-300, S7-400)....1015
3.5.6.2 Parameter assignment FM 355-2 (S7-300, S7-400)....1015
3.5.6.3 Comissioning FM 355-2 (S7-300, S7-400)....1031
3.5.6.4 FM 355-2: Diagnostics with LED display (S7-300, S7-400)....1056
3.5.7 Using FM 455 (S7-300, S7-400)....1063
3.5.7.1 Configuring FM 455 (S7-300, S7-400)....1063
3.5.7.2 Parameter assignment FM 455 (S7-300, S7-400)....1063
3.5.7.3 Programming FM 455 (S7-300, S7-400)....1079
3.5.7.4 Commissioning FM 455 (S7-300, S7-400)....1079
3.5.7.5 FM 455: Diagnostics with LED display (S7-300, S7-400)....1088
3.5.7.6 Example projects for FM 455 (S7-300, S7-400)....1098
3.5.8 Using ET 200S 2PULSE (S7-300, S7-400)....1098
3.5.8.1 2PULSE terminal assignment (S7-300, S7-400)....1098
3.5.8.2 Operating modes and functions (S7-300, S7-400)....1099
3.5.8.3 Configuring and assigning parameters to 2PULSE (S7-300, S7-400)....1142
3.5.8.4 2PULSE diagnostics (S7-300, S7-400)....1151

3.6 Auxiliary functions (S7-1200, S7-1500)....1153
3.6.1 Polyline (S7-1200, S7-1500)....1153
3.6.2 SplitRange (S7-1200, S7-1500)....1153
3.6.3 RampFunction (S7-1200, S7-1500)....1154

4 Time-based IO (S7-1500)....1155

4.1 High-precision input/output with time-based IO (S7-1500)....1155
4.1.1 Time-based IO basics (S7-1500)....1155
4.1.1.1 Using Time-based IO (S7-1500)....1155
4.1.1.2 Time behavior of Time-based IO (S7-1500)....1156
4.1.1.3 System environment (S7-1500)....1158
4.1.1.4 Hardware requirements (S7-1500)....1158
4.1.1.5 Configuration software requirements (S7-1500)....1159
4.1.1.6 Technical implementation (S7-1500)....1160
4.1.2 Configuring and parameter assignment (S7-1500)....1163
4.1.2.1 Requirements (S7-1500)....1163
4.1.2.2 Settings for Time-based IO (S7-1500)....1164
4.1.2.3 Configure IO-Link and activate Time-based IO (S7-1500)....1165
4.1.3 Programming (S7-1500)....1166
4.1.3.1 Overview of instructions (S7-1500)....1166
4.1.3.2 Programming of Time-based IO (S7-1500)....1166
4.1.4 Using the technology module (S7-1500)....1171
4.1.4.1 Configuration and parameter assignment of TIO-Modul (S7-1500)....1171
4.1.4.2 TIO-Modul Online & Diagnostics (S7-1500)....1184

5 Pulse output (S7-1500)....1185

5.1 Basic principles of pulse output (S7-1500)....1185
5.1.1 Overview of modules and properties (S7-1500)....1185
5.1.2 Basic principles of pulse output (TM Pulse 2x24V) (S7-1500)....1186
5.1.2.1 Overview (S7-1500)....1186
5.1.2.2 Pulse output (single pulse) mode (S7-1500)....1189
5.1.2.3 Pulse width modulation (PWM) mode (S7-1500)....1196
5.1.2.4 Pulse train mode (S7-1500)....1207
5.1.2.5 On/Off-delay mode (S7-1500)....1216
5.1.2.6 Frequency output mode (S7-1500)....1226

5.1.2.7 DC motor mode (S7-1500)....1233
5.1.2.8 Function: High-speed output (S7-1500)....1241
5.1.2.9 Function: Sequence counter (S7-1500)....1242
5.1.2.10 Function: Current measurement (S7-1500)....1243
5.1.2.11 Function: Current control (S7-1500)....1245
5.1.2.12 Function: Dither PWM output (S7-1500)....1248
5.1.2.13 Function: Isochronous mode (S7-1500)....1251
5.1.2.14 Function: Direct control of DQ digital outputs (S7-1500)....1252
5.1.3 Basic principles of pulse output (Compact CPU) (S7-1500)....1256
5.1.3.1 Operating modes (S7-1500)....1256
5.1.3.2 Functions (S7-1500)....1266
5.1.4 Basic principles of pulse output (TM PTO 4) (S7-1500)....1268
5.1.4.1 Overview (S7-1500)....1268
5.1.4.2 Pulse Train Output (PTO) (S7-1500)....1268

5.2 Using the module (S7-1500)....1271
5.2.1 Using TM Pulse 2x24V (S7-1500)....1271
5.2.1.1 Configuration and parameter assignment of technology module (S7-1500)....1271
5.2.1.2 Online & Diagnostics module (S7-1500)....1281
5.2.2 Using TM PTO 4 (S7-1500)....1282
5.2.2.1 Configuration and parameter assignment of technology module (S7-1500)....1282
5.2.2.2 Online & Diagnostics module (S7-1500)....1288
5.2.3 Using Compact CPU (S7-1500)....1288
5.2.3.1 Configuring and assigning parameters for Compact CPU (S7-1500)....1288
5.2.3.2 Online & Diagnostics module (S7-1500)....1295

6 Motion Control....1297

6.1 Motion Control (S7-1200, S7-1500) (S7-1200, S7-1500)....1297
6.1.1 Using S7-1200 Motion Control (S7-1200)....1297
6.1.1.1 Introduction (S7-1200)....1297
6.1.1.2 Basics for working with S7-1200 Motion Control (S7-1200)....1300
6.1.1.3 Guidelines on use of motion control (S7-1200)....1326
6.1.1.4 Using versions (S7-1200)....1326
6.1.1.5 Positioning axis technology object (S7-1200)....1336
6.1.1.6 Technology object command table (S7-1200)....1401
6.1.1.7 Download to CPU (S7-1200)....1420
6.1.1.8 Commissioning (S7-1200)....1421
6.1.1.9 Programming (S7-1200)....1427
6.1.1.10 Axis - Diagnostics (S7-1200)....1448
6.1.1.11 Appendix (S7-1200)....1453
6.1.2 Using S7-1500 Motion Control (S7-1500)....1622
6.1.2.1 Introduction (S7-1500)....1622
6.1.2.2 Technology objects (S7-1500)....1631
6.1.2.3 Using versions (S7-1500)....1766
6.1.2.4 Configuring (S7-1500)....1775
6.1.2.5 Programming (S7-1500)....1920
6.1.2.6 Downloading to CPU (S7-1500)....1947
6.1.2.7 Commissioning (S7-1500)....1947
6.1.2.8 Diagnostics (S7-1500)....1962
6.1.2.9 Technology Version V3.0 (S7-1500)....1986
6.1.2.10 Appendix (S7-1500)....2012
6.1.3 Using S7-1500T Kinematics Functions....2148
6.1.3.1 Introduction....2148

6.1.3.2 Basics....2151
6.1.3.3 Version overview....2262
6.1.3.4 Configuring....2263
6.1.3.5 Programming....2282
6.1.3.6 Commissioning....2290
6.1.3.7 Diagnostics....2303
6.1.3.8 Appendix....2308

6.2 Motion Control (S7-300, S7-400) (S7-300, S7-400)....2351
6.2.1 Position feedback basics (S7-300, S7-400)....2351
6.2.1.1 What is position feedback? (S7-300, S7-400)....2351
6.2.1.2 Encoder value acquisition (S7-300, S7-400)....2351
6.2.1.3 Gray/dual converter (S7-300, S7-400)....2352
6.2.1.4 Encoder value and scaling (S7-300, S7-400)....2352
6.2.2 Using SM 338 (S7-300, S7-400)....2354
6.2.2.1 Terminal assignment of the front connector of SM 338 (S7-300, S7-400)....2354
6.2.2.2 Configuring SM 338 (S7-300, S7-400)....2356
6.2.2.3 Assigning parameters for SM 338 (S7-300, S7-400)....2356
6.2.2.4 SM 338: Online & Diagnostics (S7-300, S7-400)....2362
6.2.3 Using ET 200S 1SSI (S7-300, S7-400)....2364
6.2.3.1 Terminal Assignment of the 1SSI (S7-300, S7-400)....2364
6.2.3.2 Configuring and assigning parameters to 1SSI (S7-300, S7-400)....2366
6.2.3.3 Diagnostics of the 1SSI (S7-300, S7-400)....2386
6.2.4 Basics of positioning with function modules (S7-300, S7-400)....2387
6.2.4.1 What is positioning? (S7-300, S7-400)....2387
6.2.4.2 Controlled positioning with rapid traverse/creep speed (S7-300, S7-400)....2388
6.2.4.3 Ranges and switching points of the positioning module (S7-300, S7-400)....2388
6.2.4.4 Encoder signals (S7-300, S7-400)....2390
6.2.5 Using FM 351 (S7-300, S7-400)....2394
6.2.5.1 Terminal assignment of FM 351 front connector (S7-300, S7-400)....2394
6.2.5.2 Functions of the FM 351 (S7-300, S7-400)....2397
6.2.5.3 Configuring FM 351 (S7-300, S7-400)....2434
6.2.5.4 Parameter assignment FM 351 (S7-300, S7-400)....2434
6.2.5.5 Using FM 351 from the user program (S7-300, S7-400)....2467
6.2.5.6 Commissioning the FM 351 (S7-300, S7-400)....2468
6.2.5.7 FM 351: Online & Diagnostics (S7-300, S7-400)....2468
6.2.6 Using FM 451 (S7-300, S7-400)....2485
6.2.6.1 Terminal assignment of FM 451 front connector (S7-300, S7-400)....2485
6.2.6.2 Functions of the FM 451 (S7-300, S7-400)....2489
6.2.6.3 Configuring FM 451 (S7-300, S7-400)....2532
6.2.6.4 Parameter assignment FM 451 (S7-300, S7-400)....2532
6.2.6.5 Using FM 451 from the user program (S7-300, S7-400)....2566
6.2.6.6 Commissioning the FM 451 (S7-300, S7-400)....2567
6.2.6.7 FM 451: Online & Diagnostics (S7-300, S7-400)....2567
6.2.7 Using ET 200S 1PosU (S7-300, S7-400)....2584
6.2.7.1 Terminal Assignment of the 1PosU (S7-300, S7-400)....2584
6.2.7.2 Basics of controlled positioning by means of rapid/creep feed (S7-300, S7-400)....2585
6.2.7.3 Functions of 1PosU (S7-300, S7-400)....2587
6.2.7.4 Configuring and assigning parameters to 1PosU (S7-300, S7-400)....2628
6.2.7.5 Diagnostics of 1PosU (S7-300, S7-400)....2649
6.2.8 Using ET 200S 1STEP 5V (S7-300, S7-400)....2650
6.2.8.1 Terminal assignment of the 1STEP 5V (S7-300, S7-400)....2650
6.2.8.2 Fundamentals of positioning with stepper motor (S7-300, S7-400)....2652

6.2.8.3 Functions of 1STEP 5V (S7-300, S7-400)....2658
6.2.8.4 1STEP 5V configuration and parameter assignment (S7-300, S7-400)....2679
6.2.8.5 1STEP 5V diagnostics (S7-300, S7-400)....2690
6.2.9 Using the IM 174 (S7-300, S7-400)....2691
6.2.9.1 Overview (S7-300, S7-400)....2691
6.2.9.2 Configuring and programming IM174 (S7-300, S7-400)....2692
6.2.10 Using Easy Motion Control (S7-300, S7-400)....2707
6.2.10.1 Positioning with Easy Motion Control (S7-300, S7-400)....2707
6.2.10.2 Functions with Easy Motion Control (S7-300, S7-400)....2715
6.2.10.3 Configuring Easy Motion Control (S7-300, S7-400)....2718
6.2.10.4 Download to CPU (S7-300, S7-400)....2741
6.2.10.5 Programming Easy Motion Control (S7-300, S7-400)....2743
6.2.10.6 Commissioning Easy Motion Control (S7-300, S7-400)....2757
6.2.10.7 Diagnostics (S7-300, S7-400)....2764

7 SIMATIC Ident....2775

7.1 Configuring Ident systems....2775
7.1.1 Technology object "SIMATIC Ident"....2775
7.1.2 Adding a technology object....2775
7.1.3 Configuring devices with the aid of the technology object....2777
7.1.3.1 Working with the configuration dialog....2777
7.1.3.2 Configuring basic parameters....2778
7.1.3.3 Configuring Ident device parameters....2778
7.1.3.4 Configuring reader parameters....2780
7.1.4 Calling the Ident instruction in the user program....2783
7.1.5 Loading the configuration on the CPU....2784

7.2 Diagnosing Ident systems....2785
7.2.1 Block diagnostics....2785
7.2.2 Hardware diagnostics....2786

Index....2789

Counting, measurement and position input 1

1.1 Counting, measurement and position input (S7-1500) (S7-1500)

1.1.1 The basics of counting, measurement and position input (S7-1500)

1.1.1.1 Overview of modules and properties (S7-1500)

Modules for the S7-1500 and ET 200MP systems

The table below summarizes the performance features of the modules for counting, measuring and position input for the S7-1500 and ET 200MP systems.

Property	S7-1500 / ET 200MP				
	Technology module			Digital input module	Compact CPU
	TM Count 2x24V	TM PosInput 2	TM Timer DIDQ 16x24V	DI 32x24VDC HF DI 16x24VDC HF	CPU 1511C-1 PN CPU 1512C-1 PN
Number of counters	2	2	4[1]	2	6
Use of counters can be activated/deactivated	—	—	X	X	X
Maximum signal frequency	200 kHz	1 MHz	50 kHz	1 kHz	100 kHz
Maximum count frequency for incremental encoders with quadruple evaluation (Page 64)	800 kHz	4 MHz	200 kHz	—	400 kHz
Maximum counting range	32 bits (Page 20)	32 bits (Page 20)	32 bits (Page 69)	32 bits (Page 73)	32 bits (Page 20)
Maximum position value range (Page 22)	—	32 bits	—	—	—
24 V incremental encoder connection	X (Page 60)	—	X (Page 70)	—	X (Page 60)
24 V pulse encoder connection	X (Page 60)	—	X (Page 70)	X (Page 79)	X (Page 60)
RS422/TTL incremental and pulse encoder (Page 62) connection	—	X	—	—	—
SSI absolute encoder (Page 64) connection	—	X	—	—	—

Property	S7-1500 / ET 200MP				
	Technology module			Digital input module	Compact CPU
	TM Count 2x24V	TM PosInput 2	TM Timer DIDQ 16x24V	DI 32x24VDC HF DI 16x24VDC HF	CPU 1511C-1 PN CPU 1512C-1 PN
Position input for Motion Control (Page 60)	X	X	—	—	X
5 V encoder supply	—	X	—	—	—
24 V encoder supply	X	X	X	—	X
Number of additional digital inputs per counter	3	2	—	—	2
Number of physical digital outputs per counter	2	2	—	—	1
Number of logical digital outputs per counter	2	2	—	1	2
Software gate	X (Page 25)	X (Page 25)	—	X (Page 75)	X (Page 25)
Hardware gate	X (Page 25)	X (Page 25)	—	—	X (Page 25)
Capture function (Latch) (Page 28)	X	X	—	—	X
Synchronization (Page 32)	X	X	—	—	X
Comparison functions	X (Page 39)	X (Page 39)	—	X (Page 76)	X (Page 39)
Hysteresis (Page 56)	X	X	—	—	X
Frequency, velocity and period duration measurement (Page 50)	X	X	—	—	X
Isochronous mode support	X (Page 67)	X (Page 67)	X (Page 71)	X (Page 79)	—
Support for diagnostic interrupts for sensor signals	X (Page 59)	X (Page 59)	—	—	X (Page 59)
Hardware interrupt support	X (Page 59)	X (Page 59)	—	X (Page 78)	X (Page 59)
Configurable filter for count signals and digital inputs	X	X	—	X	X

[1] The number of available counters is dependent on the channel configuration. In order to use four counters, you must choose the use of eight inputs in the channel configuration. If you choose the use of three inputs, you can use one counter. Other channel configurations do not allow any counter use.

Modules for the ET 200SP system

The following table provides an overview of the performance features of the modules for counting, measuring and position input for the ET 200SP system.

Property	ET 200SP			
	Technology module			Digital input module
	TM Count 1x24V	TM PosInput 1	TM Timer DIDQ 10x24V	DI 8x24VDC HS
Number of counters	1	1	3[1]	4
Use of counters can be activated/deactivated	—	—	X	X
Maximum signal frequency	200 kHz	1 MHz	50 kHz	10 kHz
Maximum count frequency for incremental encoders with quadruple evaluation (Page 64)	800 kHz	4 MHz	200 kHz	—
Maximum counting range	32 bits (Page 20)	32 bits (Page 20)	32 bits (Page 69)	32 bits (Page 73)
Maximum position value range (Page 22)	—	32 bits	—	—
24 V incremental encoder connection	X (Page 60)	—	X (Page 70)	—
24 V pulse encoder connection	X (Page 60)	—	X (Page 70)	X (Page 79)
RS422/TTL incremental and pulse encoder (Page 62) connection	—	X	—	—
SSI absolute encoder (Page 64) connection	—	X	—	—
Position input for Motion Control (Page 60)	X	X	—	—
24 V encoder supply	X	X	X	X
Number of additional digital inputs per counter	3	2	—	1
Number of physical digital outputs per counter	2	2	—	—
Number of logical digital outputs per counter	2	2	—	1
Software gate	X (Page 25)	X (Page 25)	—	X (Page 75)
Hardware gate	X (Page 25)	X (Page 25)	—	X (Page 75)
Capture function (Latch) (Page 28)	X	X	—	—
Synchronization (Page 32)	X	X	—	—
Comparison functions	X (Page 39)	X (Page 39)	—	X (Page 76)
Hysteresis (Page 56)	X	X	—	—
Frequency, velocity and period duration measurement (Page 50)	X	X	—	—
Isochronous mode support	X (Page 67)	X (Page 67)	X (Page 71)	X (Page 79)
Support for diagnostic interrupts for sensor signals	X (Page 59)	X (Page 59)	—	—

Property	ET 200SP			
	Technology module			Digital input module
	TM Count 1x24V	TM PosInput 1	TM Timer DIDQ 10x24V	DI 8x24VDC HS
Hardware interrupt support	X (Page 59)	X (Page 59)	—	—
Configurable filter for count signals and digital inputs	X	X	—	X

[1] One counter for incremental encoder (A, B phase-shifted) and two counters for pulse encoder

Modules for the ET 200AL system

The following table provides an overview of the performance features of the modules for counting, measuring and position input for the ET 200AL system.

Property	ET 200AL
	Digital input/digital output module
	DIQ 16x24VDC/0.5A 8xM12
Number of counters	4
Use of counters can be activated/deactivated	X
Maximum signal frequency	2 kHz
Maximum counting range	32 bits
24 V incremental encoder connection	—
24 V pulse encoder connection	X
24 V encoder supply	X
Number of additional digital inputs per counter	2
Number of physical digital outputs per counter	1
Number of logical digital outputs per counter	1
Software gate	X
Hardware gate	X
Capture function (Latch)	—
Synchronization	—
Comparison functions	X
Frequency, velocity and period duration measurement	—
Isochronous mode support	—
Support for diagnostic interrupts for sensor signals	X
Hardware interrupt support	X
Configurable filter digital inputs	X

1.1.1.2 Basics of counting, measuring and position input (TM Count, TM PosInput, Compact CPU) (S7-1500)

Convention (S7-1500)

Technology module: We use the term "technology module" in this documentation both for the technology modules TM Count and TM PosInput and the technology component of the compact CPUs.

Overview of applications (S7-1500)

Introduction

The technology module is configured and assigned parameters using the configuration software.

The operation and control of the technology module functions is effected either via the technology object or through the application program via the control and feedback interface.

System environment for TM Count and TM PosInput

The technology modules can be used in the following system environments:

Application scenarios	Components required	Configuration software	In the user program
Central operation with an S7-1500 CPU or 151xSP CPU	• S7-1500 automation system or ET 200SP CPU • Technology module	STEP 7 (TIA Portal): Operating with "Counting and measurement" technology object • Device configuration with hardware configuration (Page 127) • Parameter setting with High_Speed_Counter or SSI_Absolute_Encoder technology object	For incremental / pulse encoders: High_Speed_Counter instruction for the technology object For SSI absolute encoder: SSI_Absolute_Encoder instruction for the technology object
		STEP 7 (TIA Portal): Position input for "Motion Control" technology object • Device configuration with hardware configuration (Page 127) • Parameter setting with axis and measuring input technology object (Page 133)	Instructions for "Motion Control" technology object
		STEP 7 (TIA Portal): Manual operation (without technology object) • Device configuration and parameter setting (Page 127) with hardware configuration	Direct access to control and feedback interface of the technology module in the I/O data

Application scenarios	Components required	Configuration software	In the user program
Distributed operation with a S7-1500 CPU	• S7-1500 automation system • ET 200 distributed I/O system • Technology module	STEP 7 (TIA Portal): Operating with "Counting and measurement" technology object • Device configuration with hardware configuration (Page 127) • Parameter setting with High_Speed_Counter or SSI_Absolute_Encoder technology object	For incremental / pulse encoders: High_Speed_Counter instruction for the technology object For SSI absolute encoder: SSI_Absolute_Encoder instruction for the technology object
		STEP 7 (TIA Portal): Position input for "Motion Control" technology object • Device configuration with hardware configuration (Page 127) • Parameter setting with axis and measuring input technology object (Page 133)	Instructions for "Motion Control" technology object
		STEP 7 (TIA Portal): Manual operation (without technology object) • Device configuration and parameter setting (Page 127) with hardware configuration	Direct access to control and feedback interface (Page 166) of the technology module in the IO data
Distributed operation with an S7-1200 CPU	• S7-1200 automation system • ET 200 distributed I/O system • Technology module	STEP 7 (TIA Portal): Device configuration and parameter setting (Page 127) with hardware configuration	Direct access to control and feedback interface (Page 166) of the technology module in the IO data

Application scenarios	Components required	Configuration software	In the user program
Distributed operation with a S7-300/400 CPU	• S7-300/400 automation system • ET 200 distributed I/O system • Technology module	STEP 7 (TIA Portal): Device configuration and parameter setting (Page 127) with hardware configuration STEP 7: Device configuration and parameter setting with HSP (up to firmware version V1.2) or GSD file	Direct access to control and feedback interface (Page 166) of the technology module in the IO data
Distributed operation in a third-party system	• Third-party automation system • ET 200 distributed I/O system • Technology module	Third-party configuration software: Device configuration and parameter settings with GSD file	Direct access to control and feedback interface (Page 166) of the technology module in the IO data

System environment for a Compact CPU

The Compact CPUs can be used in the following system environments:

Application scenarios	Components required	Configuration software	In the user program
Central operation with a S7-1500 Compact CPU	• S7-1500 automation system • Compact CPU	STEP 7 (TIA Portal): Operating with "Counting and measurement" technology object • Device configuration with hardware configuration (Page 127) • Parameter setting with High_Speed_Counter technology object	High_Speed_Counter instruction for the technology object
		STEP 7 (TIA Portal): Position input for "Motion Control" technology object • Device configuration with hardware configuration (Page 127) • Parameter setting with axis and measuring input technology object (Page 133)	Instructions for "Motion Control" technology object
		STEP 7 (TIA Portal): Manual operation (without technology object) • Device configuration and parameter setting (Page 127) with hardware configuration	Direct access to control and feedback interface of the technology module in the I/O data

Parameter assignment options

In an S7-1500 system, you have two options for parameter assignment and control of technology module functions:

- Configuration using the technology object and control using the associated instruction
  Access to the control and feedback interface of the technology module takes place through the technology object.
- Parameter setting via hardware configuration
  The control and feedback interface of the technology module is accessed using direct access to the I/O data.

Configuration via technology object

For central and distributed use, we recommend the convenient, graphics-assisted configuration using a technology object. A detailed description of this configuration can be found in section High_Speed_Counter technology object or SSI_Absolute_Encodertechnology object.

You specify the "Operation with technology object "Counting and measuring"" in the device configuration of the technology module: see section Operating mode (Page 132).

You can assign the technology module and counting channel in the basic parameters of the technology object: see section Basic parameters (Page 84).

Parameter setting via hardware configuration

You specify the "Manual operation (without technology object)" in the device configuration of the technology module: see section Operating mode (Page 132).

Additional support for parameter setting via hardware configuration is available in the context-sensitive help for the parameters in STEP 7 (TIA Portal). A description of the control and feedback interface is available in the following sections:

- Assignment of the control interface (Page 166)
- Assignment of the feedback interface (Page 168)

Recording of count signals (S7-1500)

Counting with incremental or pulse encoder (S7-1500)

Counting refers to the recording and adding up of events. The counters of the technology modules capture and evaluate pulse and incremental signals. The count direction can be specified using encoder or pulse signals or through the user program.

You can control counting processes using the digital inputs of the technology module. You can switch the digital outputs exactly at defined counter values, regardless of the user program.

You can configure the response of the counters using the functionalities described below.

Counter limits

The counter limits define the counter value range used. The counter limits are configurable and can be modified during runtime using the user program.

The highest counter limit that can be set is 2147483647 (2^{31}–1). The lowest counter limit that can be set is –2147483648 (–2^{31}).

You can configure the response of the counter at the counter limits:

- Continue or stop counting upon violation of a counter limit (automatic gate stop)
- Set counter value to start value or to opposite counter limit upon violation of a counter limit

Start value

You can configure a start value within the counter limits. The start value can be modified during runtime by the user program.

The technology module can, as configured, set the current counter value to the start value upon synchronization, upon Capture function activation, upon violation of a counter limit or when the gate is opened.

Gate control

Opening and closing the hardware gate and software gate defines the period of time during which the counting signals are captured.

The hardware gate is controlled externally via a digital input of the technology module. The software gate is controlled via the user program. The hardware gate can be enabled through parameter assignment. The software gate (bit in the control interface of the cyclic I/O data) cannot be disabled.

Capture (Latch)

You can configure an external reference signal edge that triggers the saving of the current counter value or position value as a Capture value. The following external signals can trigger the Capture function:

- Rising or falling edge of a digital input
- Both edges of a digital input
- Rising edge of the N signal at the encoder input

When using a digital input, you can specify whether counting is to continue from the current counter value or from the start value after the Capture function. When using the rising edge of the N signal at the encoder input, counting is to continue from the current counter value after the Capture function. The use of a digital input and the use of the N signal are not mutually exclusive for the Capture function.

The parameter "Frequency of the Capture function" determines if the function is executed for each configured edge or only once after each enable.

Measuring input

If you use the position input for Motion Control (Page 60), you can use the "Measuring input" technology object to execute a measuring input function with a hardware digital input.

Synchronization

You can configure an external reference signal edge to load the counter with the specified start value. The following external signals can load the counter with the start value:

- Rising or falling edge of a digital input
- Rising edge of signal N at the encoder input
- Rising edge of signal N at the encoder input depending on the level of the assigned digital input

The parameter "Frequency of synchronization" determines whether the function is executed for each configured edge or only once after each enable.

Hysteresis

You can specify hysteresis for the comparison values, within which a digital output is prevented from switching again. An encoder can come to a standstill at a specific position, and slight movements may make the counter value fluctuate around this position. If a comparison value or a counting limit lies within this fluctuation range, the corresponding digital output will be switched on and off with corresponding frequency if hysteresis is not used. The hysteresis prevents these unwanted switching operations.

Position input with SSI absolute encoder (S7-1500)

Description

You can use the TM PosInput technology modules with an SSI absolute encoder for position detection. The technology module reads the position value via a synchronous serial interface from the SSI absolute encoder and makes it available to the controller.

You can switch the digital outputs of the technology module exactly at defined position values, regardless of the user program. Position input with an SSI absolute encoder does not involve gate control. Due to system constraints, synchronization is not possible with an SSI absolute encoder.

Gray- and dual-code

Gray-code and dual-code SSI absolute encoders are supported.

Capture (Latch)

You can configure one or both edges of a digital input that triggers a saving of the current position value as Capture value.

The parameter "Frequency of the Capture function" determines if the function is executed for each configured edge or only once after each enable.

Measuring input

If you use the position input for Motion Control (Page 60), you can use the "Measuring input" technology object to execute a measuring input function with a hardware digital input.

Hysteresis

You can specify hysteresis for the comparison values, within which a digital output is prevented from switching again. An encoder can come to a standstill at a specific position, and slight movements may make the position value fluctuate around this position. If a comparison value or a limit lies within this fluctuation range, the corresponding digital output is switched on and off with corresponding frequency if hysteresis is not used. The hysteresis prevents these unwanted switching operations.

Range for position value

You can specify a frame length of 10 bits to 40 bits for the SSI absolute encoder. The configurable bit numbers of the LSB and the MSB of the position value in the frame define the value range. The technology module can read a position value with a maximum length of 32 bits and communicate it to the controller.

If you use an encoder with a position value length of up to 31 bits, the position value is handled unsigned as a positive value and can assume values between 0 and $2^{(MSB-LSB+1)}-1$. If you use an encoder with a position value length of 32 bits, the MSB of the position value corresponds to the sign and the position value can assume values between –2147483648 and 2147483647. If you use a 32 bit position value for the comparison function, the position value is interpreted as DINT.

Complete SSI frame

Instead of having a measured variable returned, you can choose to have the least significant 32 bits of the current unprocessed SSI frame returned. This provides you with encoder-specific additional bits, such as error bits, in addition to the position value. If the SSI frame is shorter than 32 bits, the complete SSI frame is returned right-aligned and the top unused bits are returned with "0" in the feedback interface.

Behavior at the counting limits (S7-1500)

Violation of a counting limit

The high counting limit is violated when the current counter value is equal to the high counting limit and another upward count pulse is received. The counter low limit is violated when the current counter value is equal to the counter low limit and another downward count pulse is received.

The appropriate status bit is set in the feedback interface in the event of limit violation:

Counting limit violated	Status bit
High counting limit	EVENT_OFLW is set
Low counting limit	EVENT_UFLW is set

You can reset the status bits with RES_EVENT .

You can configure whether or not and from which counter value counting is to continue following counting limit violation.

Note

The high counting limit and the start value define the value range of the counter:

Value range of the counter = (high limit – start value) + 1

Examples

The figure below shows an example for terminating the counting process (automatic gate stop) after an overflow and setting the counter to the start value:

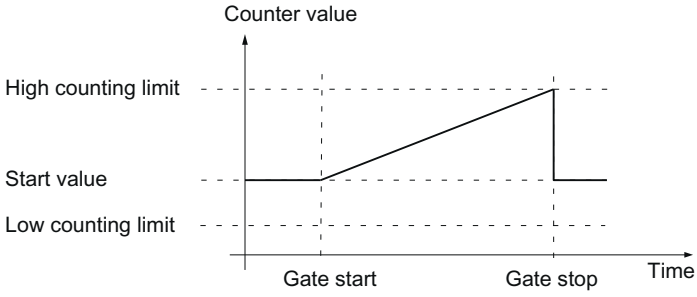

The figure below shows an example for continuing the counting process after an overflow and setting the counter to the start value:

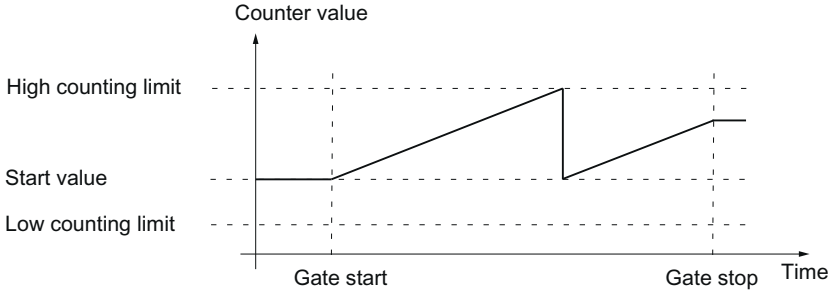

The figure below shows an example for terminating counting after an overflow and setting the counter to the opposite counting limit:

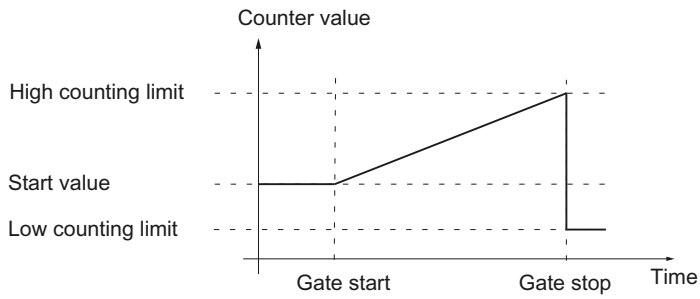

The figure below shows an example for continuing the counting process after an overflow and setting the counter to the opposite counting limit:

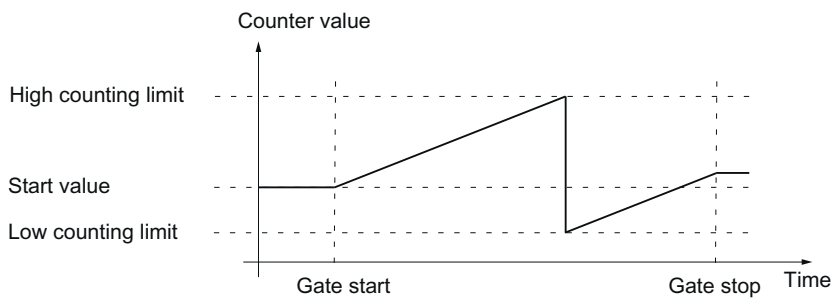

Gate control with incremental or pulse encoder (S7-1500)

Many applications require counting processes to be started or stopped in accordance with other events. In such cases, counting is started and stopped using the gate function.

The technology modules have two gates for each channel. These define the resulting internal gate:

- Software gate
- Hardware gate

Software gate (S7-1500)

The software gate of the channel is opened and closed with the control bit (Page 166) SW_GATE. The status of the software gate is indicated by the feedback bit (Page 168) STS_SW_GATE .

Hardware gate (S7-1500)

The hardware gate is optional. You open and close the hardware gate by means of signals at the configured digital inputs of the channel.

Note

The configured input filters delay the control signal of the digital input.

The status of a DIm digital input is indicated by the respective feedback bit (Page 168) STS_DIm .

Level-triggered opening and closing of the hardware gate with a digital input

The figure below shows an example of level-triggered opening and closing with a digital input. The digital input is configured to be active with high level:

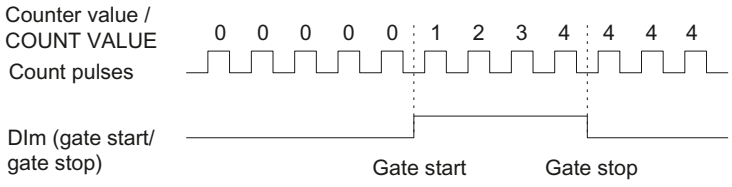

As long as the digital input is active, the hardware gate is open and the count pulses are counted. The hardware gate is closed when the digital input becomes inactive. The counter value stays constant and ignores any further count pulses.

Edge-triggered opening and closing of the hardware gate with two digital inputs

The figure below shows an example of opening and closing with two digital inputs. The two digital inputs are configured so that the rising edge is evaluated:

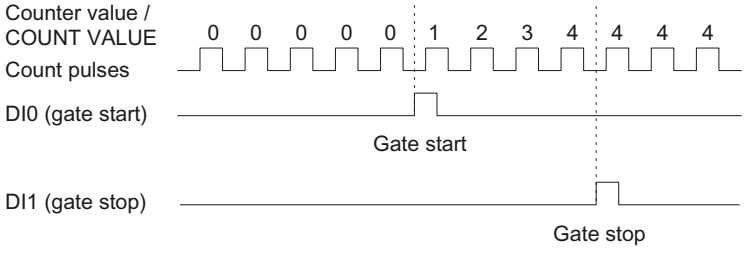

The hardware gate is opened with the configured edge at the digital input that is configured for opening. The hardware gate is closed with the configured edge at the digital input that is configured for closing.

Internal gate (S7-1500)

Internal gate

The internal gate is open if the software gate is open and the hardware gate is open or has not been configured. The status of the internal gate is indicated by the feedback bit (Page 168) STS_GATE.

If the internal gate is open, counting is started. If the internal gate is closed, all other count pulses are ignored and counting is stopped.

If you want to control a counting process with the hardware gate only, the software gate must be open. If you do not configure a hardware gate, the hardware gate is considered to be always open. In this case, you open and close the internal gate with the software gate only.

Hardware gate	Software gate	Internal gate
Open/not configured	open	open
Open/not configured	closed	closed
closed	open	closed
closed	closed	closed

When you configure the counter behavior, you can specify whether the counting process is to start from the start value or from the current counter value when the internal gate is opened.

The internal gate can also be automatically closed upon violation of a counting limit. The software or hardware gate must then be closed and reopened to continue counting.

Counter behavior at gate start (S7-1500)

You have the following configuration options for counter behavior upon gate start:

- Setting counter to start value
- Continuing with the current counter value

Setting counter to start value

Counter behavior is as follows for this configuration:

Each counting process starts with the start value when the internal gate is opened.

The figure below shows an example for continuing the counting process after counter is set to the start value:

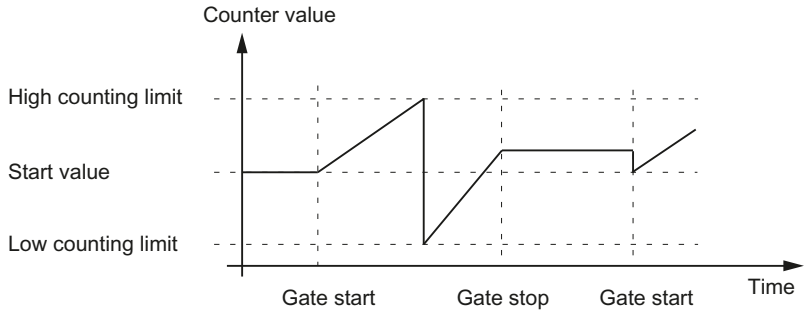

Continuing with the current counter value

Counter behavior is as follows for this configuration:

Each counting process starts from the current counter value after the internal gate is reopened.

The figure below shows an example for continuing the counting process with the current counter value:

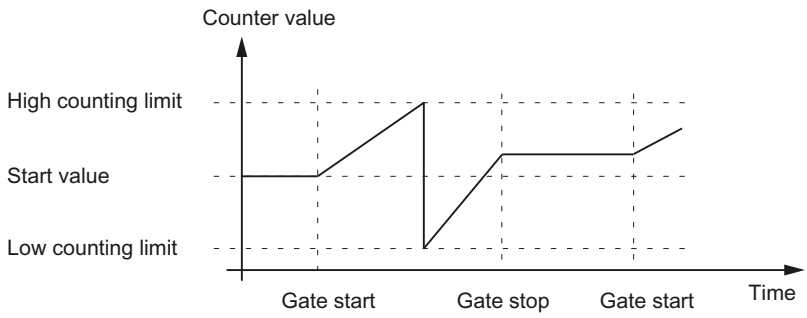

Capture (Latch) (S7-1500)

Capture with incremental or pulse encoder (S7-1500)

Description

The "Capture" function is used to save the current counter value with an external reference signal. You can configure the Capture function for the following reference signals:

- Rising or falling edge at a digital input
- Rising and falling edge at a digital input
- Rising edge of signal N at the encoder input

Function principle

The Capture value is always the exact counter value at the time of the respective edge (delayed by the configured input filter time). The Capture function is effective regardless of the status of the internal gate. The unchanged counter value is saved when the gate is closed.

The figure below shows an example of the Capture function with the following configuration:

- Start value = 0
- Capture event upon rising edge at configured digital input
- Set counter to start value at gate start
- Continue counting after Capture event

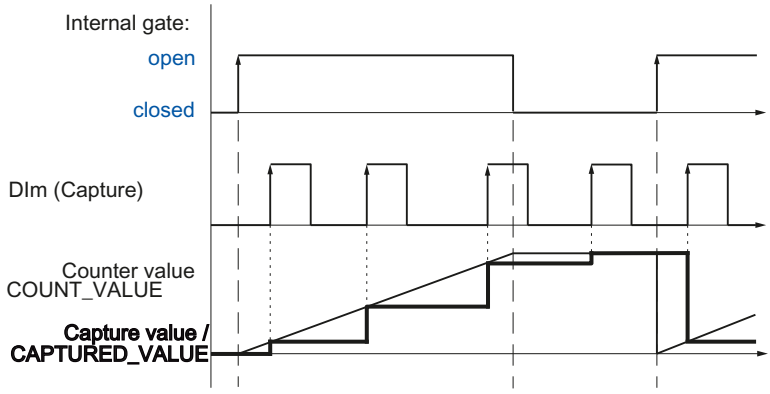

The figure below shows another example of the Capture function with the following configuration:

- Start value = 0
- Capture event upon rising edge at configured digital input
- Set counter to start value at gate start
- Reset counter value to start value after Capture event and continue counting.

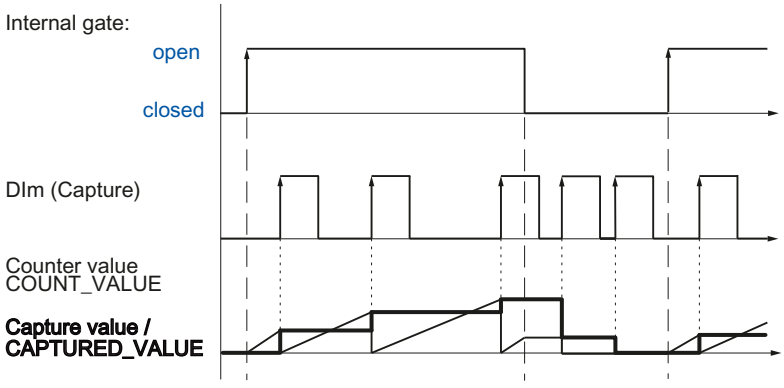

The control bit (Page 166) EN_CAPTURE is used to enable the Capture function. The feedback bit (Page 168) EVENT_CAP indicates that a counter value has been saved as a Capture in the feedback interface. If you reset EN_CAPTURE, EVENT_CAP is also reset. The status of a digital input is indicated by the respective feedback bit (Page 168) STS_DIm .

The figure below shows an example of the EN_CAPTURE and EVENT_CAP bits with use of the one-time Capture function by the rising edge at a digital input:

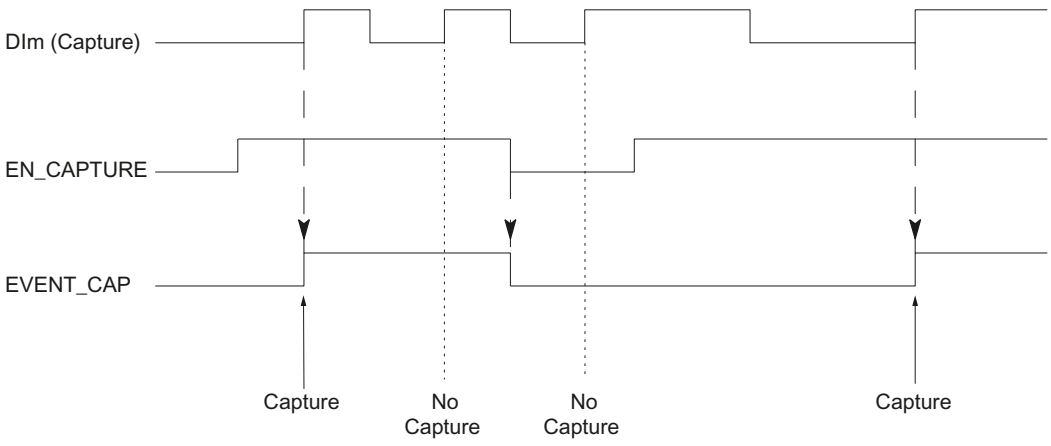

The figure below shows an example of the EN_CAPTURE and EVENT_CAP bits with use of the periodic Capture function by the rising edge at a digital input:

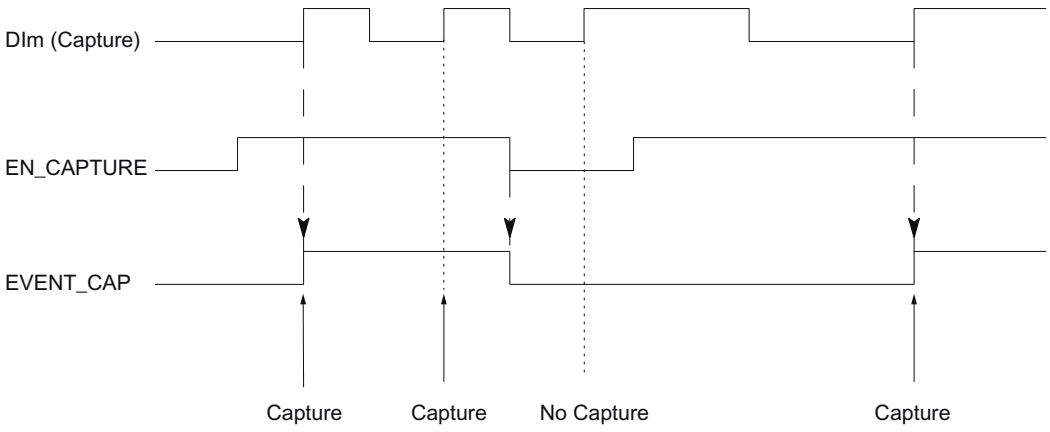

Note

The configured input filters delay the control signal of the corresponding digital input.

The Capture function has no effect on the feedback bit STS_CNT and the LEDs UP and DN.

Hardware interrupt

You can configure a hardware interrupt for the Capture function. If the hardware interrupts are triggered more quickly by the system than they can be acknowledged, hardware interrupts are lost and the "Hardware interrupt" diagnostics interrupt is signaled.

Capture with SSI absolute encoder (S7-1500)

Description

The "Capture" function is used to save the current position value using an external reference signal. You can configure the Capture function for the following reference signals:

- Rising **or** falling edge at a digital input
- Rising **and** falling edge at a digital input

Function principle

At the time of each edge, the position value of the last valid SSI frame is stored in the Capture value.

The figure below shows an example of the Capture event by a rising edge at the configured digital input:

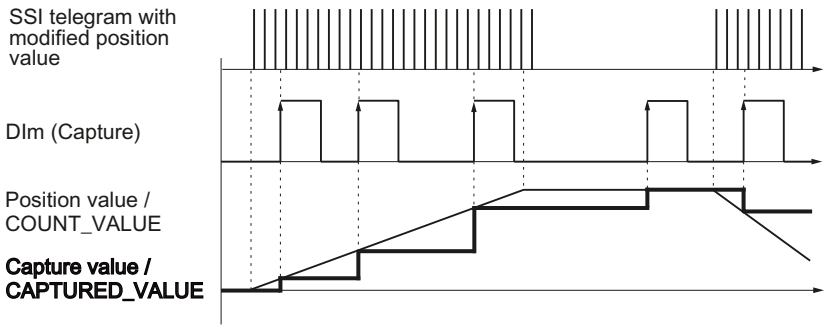

The control bit (Page 166) EN_CAPTURE is used to enable the Capture function. The feedback bit (Page 168) EVENT_CAP indicates that a position value has been saved as Capture value in the feedback interface. If you reset EN_CAPTURE, EVENT_CAP is also reset. The status of a digital input is indicated by the respective feedback bit (Page 168) STS_DIm .

The figure below shows an example of the EN_CAPTURE and EVENT_CAP bits with use of the one-time Capture function by the rising edge at a digital input:

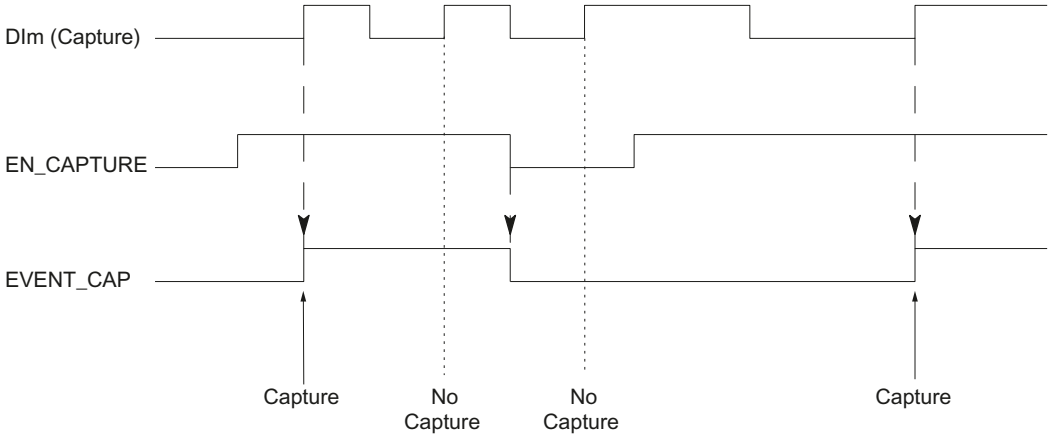

The figure below shows an example of the EN_CAPTURE and EVENT_CAP bits with use of the periodic Capture function by the rising edge at a digital input:

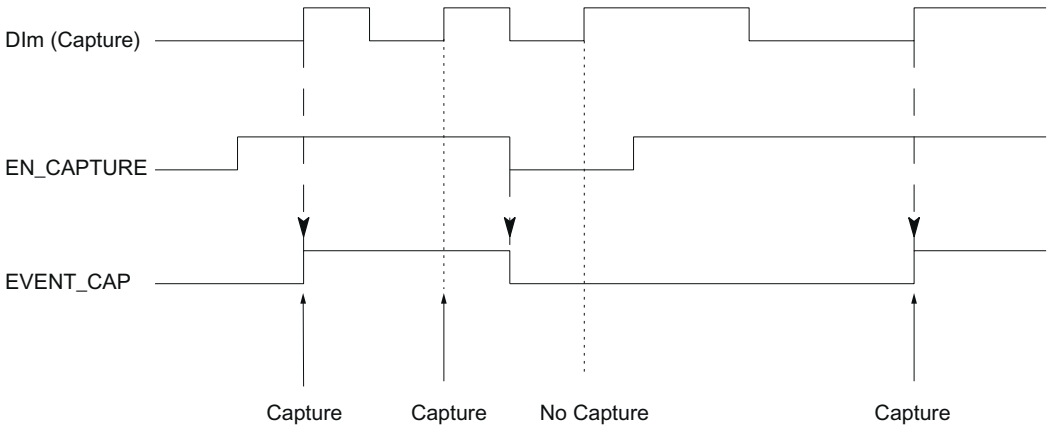

Note

The configured input filters delay the control signal of the corresponding digital input.

Hardware interrupt

You can configure a hardware interrupt for the Capture function. If the hardware interrupts are triggered more quickly by the system than they can be acknowledged, hardware interrupts are lost and the "Hardware interrupt" diagnostics interrupt is signaled.

Synchronization (S7-1500)

Description

You use the "Synchronization" function to set the counter to the pre-defined start value with an external reference signal. You can configure synchronization for the following reference signals:

- Rising or falling edges at a digital input
- Rising edge of signal N at the encoder input
- Rising edge of signal N at the encoder input defined by the level of a digital input

Function principle

Synchronization always takes place exactly at the time of the reference signal. Synchronization is effective regardless of the status of the internal gate.

You use the control bit (Page 166) EN_SYNC_UP to enable synchronization for counting in an upwards direction. Use the control bit (Page 166) EN_SYNC_DN to enable synchronization

for counting down. The feedback bit (Page 168) EVENT_SYNC indicates that synchronization has been performed. Resetting EN_SYNC_UP or EN_SYNC_DN also resets EVENT_SYNC.

Note

The configured input filters delay the control signal of the corresponding digital input.

Synchronization has no effect on the feedback bit (Page 168) STS_CNT and the LEDs UP and DN.

Single synchronization

The figure below shows an example of the EN_SYNC_UP, EN_SYNC_DNand EVENT_SYNC bits with single synchronization by an edge at a digital input for count pulses in an upwards direction:

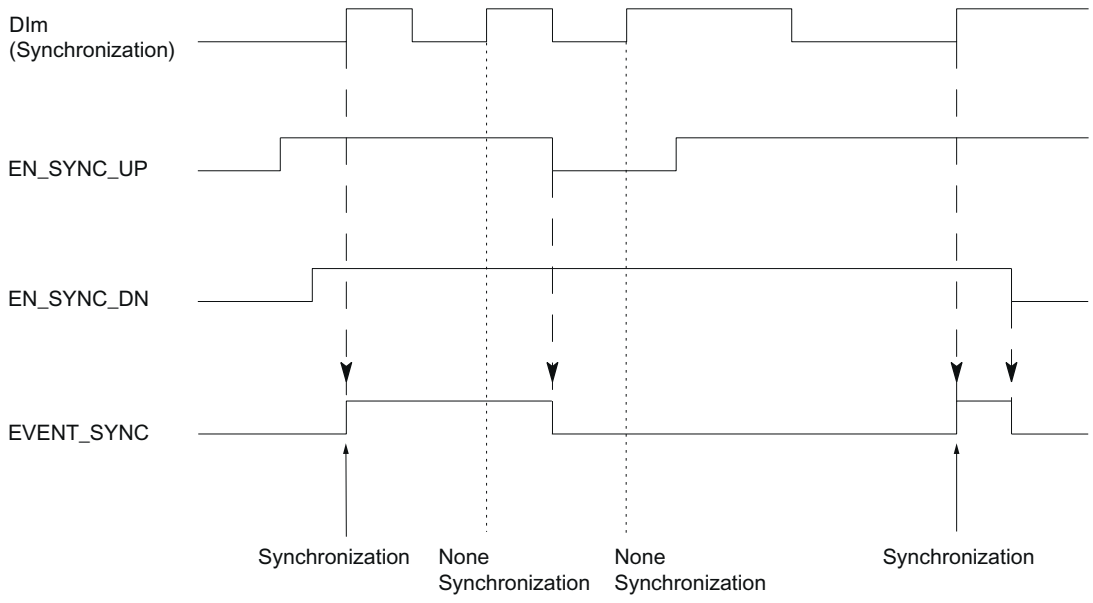

After synchronization is enabled for counting in an upwards direction, the counter is synchronized at the first rising edge at the configured digital input. The counter can only be synchronized again once the control bit (Page 166) EN_SYNC_UP has been reset and set again.

Periodic synchronization

The figure below shows an example of the EN_SYNC_UP, EN_SYNC_DN and EVENT_SYNC bits with periodic synchronization by an edge at a digital input for count pulses in an upwards direction:

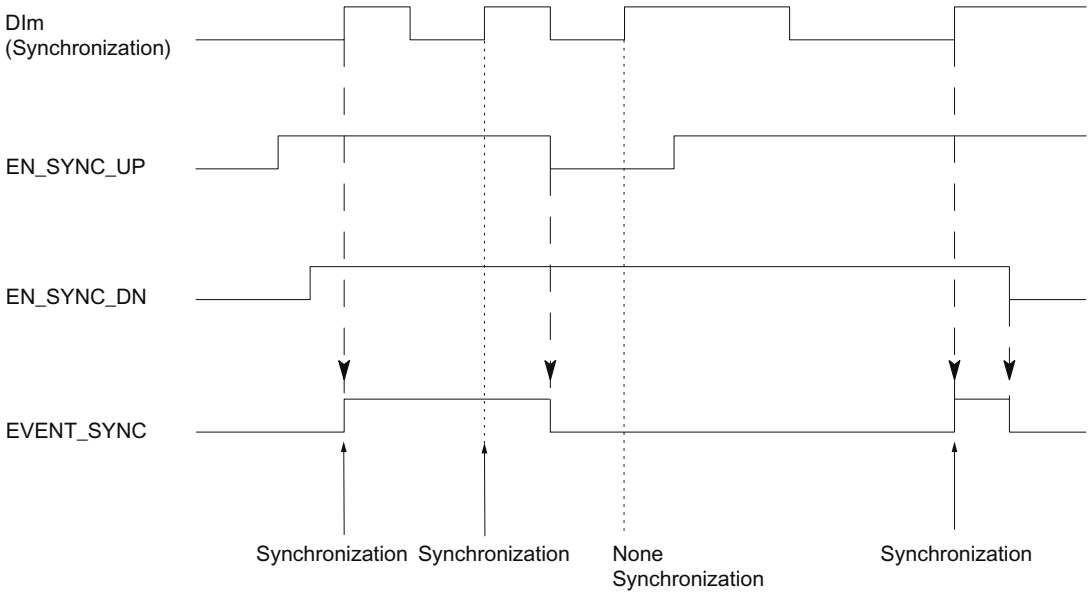

As long as synchronization for counting in an upwards direction is enabled, the counter is synchronized at each rising edge at the configured digital input.

Hardware interrupt

You can configure a hardware interrupt for the synchronization. If the hardware interrupts are triggered more quickly by the system than they can be acknowledged, hardware interrupts are lost and the "Hardware interrupt" diagnostics interrupt is signaled.

Synchronization by digital input (S7-1500)

You can trigger synchronization by edges at a digital input.

Single synchronization

The figure below shows an example for single synchronization by an edge at a digital input:

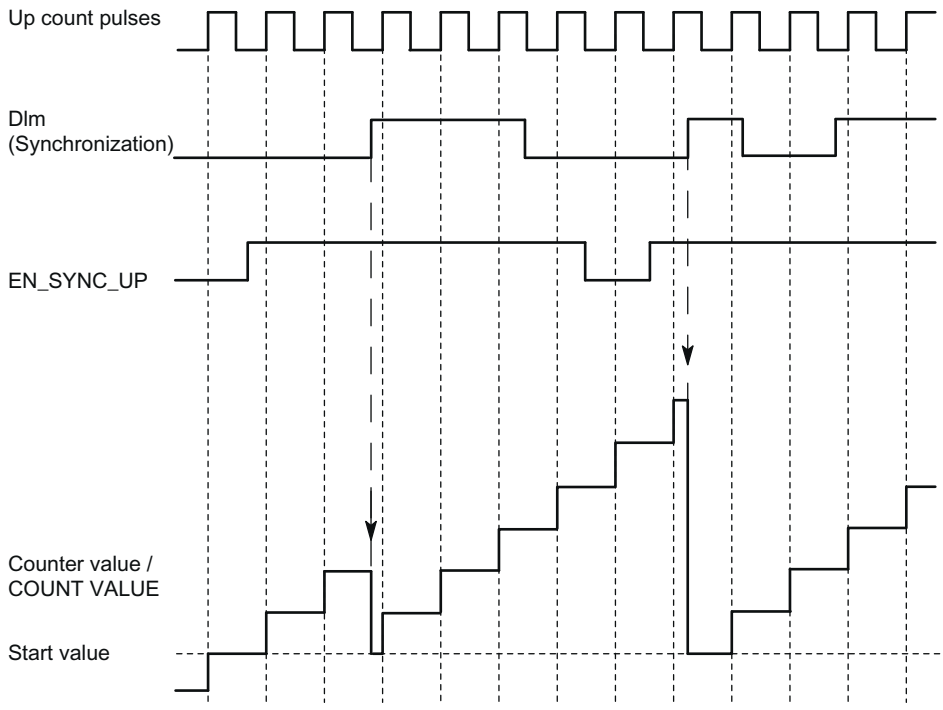

After synchronization is enabled for counting in an upwards direction, the counter is synchronized at the first rising edge at the configured digital input. Until the control bit (Page 166) EN_SYNC_UP has been reset and set again, any additional rising edge at the digital output is ignored. The counter can then be synchronized again.

Periodic synchronization

The figure below shows an example for periodic synchronization by an edge at a digital input:

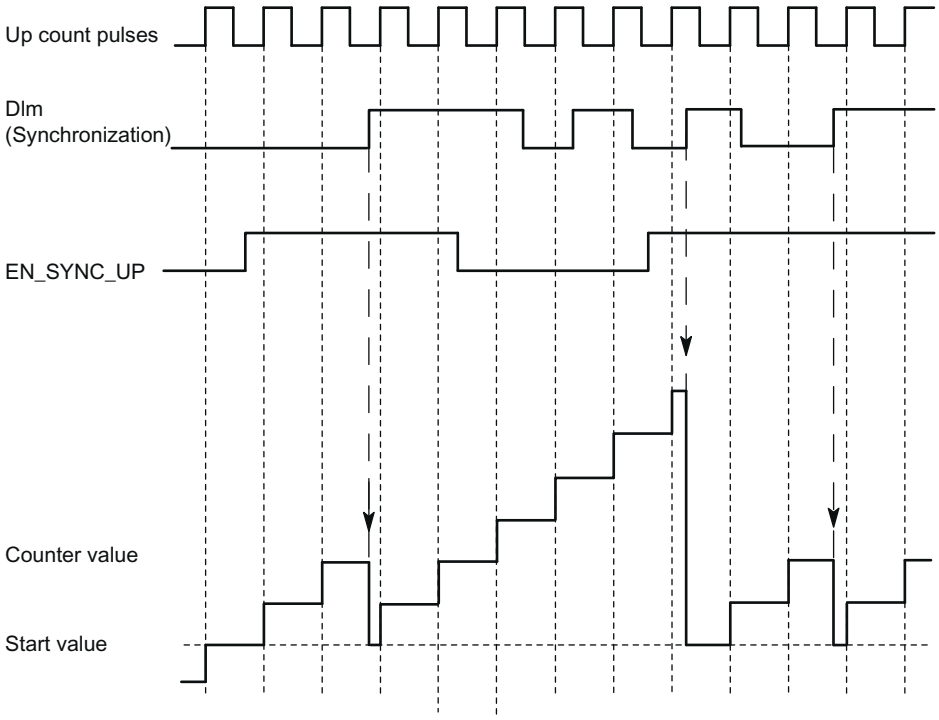

As long as synchronization for counting in an upwards direction is enabled, the counter is synchronized at each rising edge at the configured digital input.

Synchronization at signal N (S7-1500)

You can trigger synchronization at signal N at the encoder input either directly or depending on the status of a digital input.

Single synchronization

The figure below shows an example of single synchronization at signal N (not dependent on a digital input):

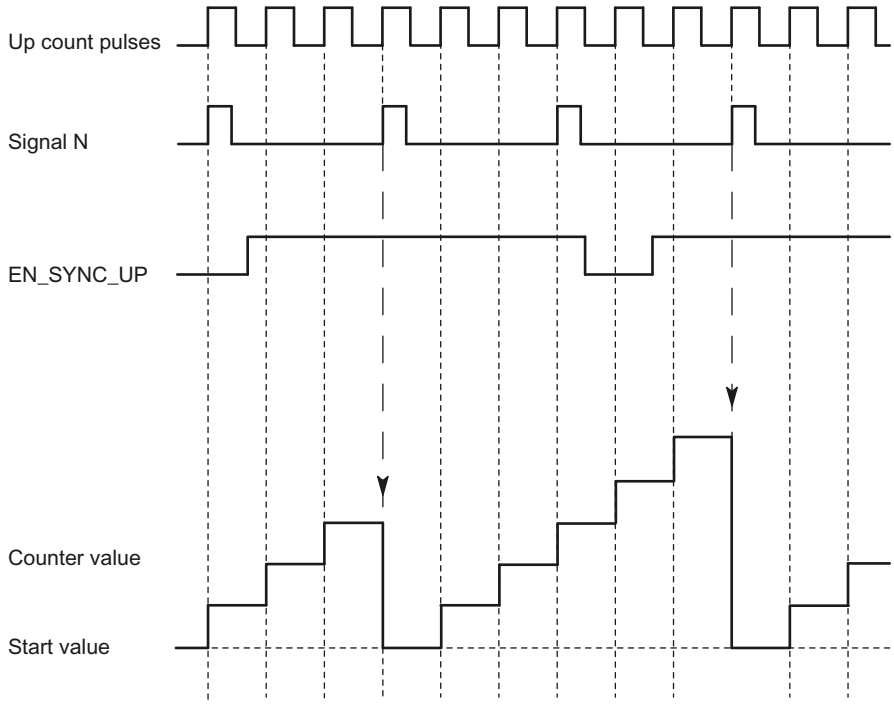

After synchronization is enabled for counting in an upwards direction, the counter is synchronized at the first signal N. After resetting and setting the control bit (Page 166) EN_SYNC_UP once again, the counter can be synchronized again.

Periodic synchronization

The figure below shows an example for periodic synchronization at signal N:

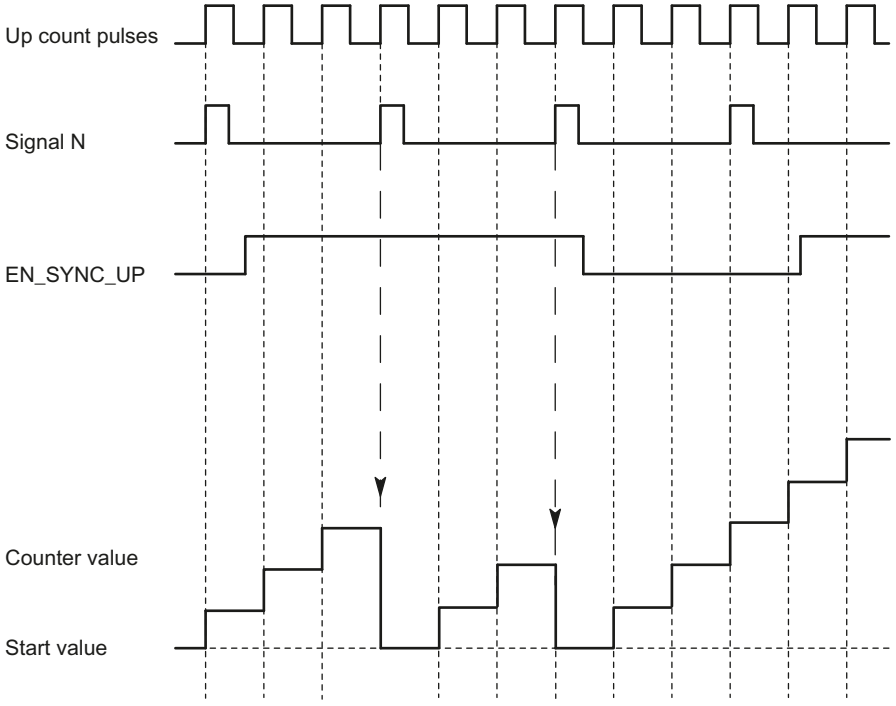

As long as synchronization for counting in an upwards direction is enabled, the counter is synchronized at each signal N.

Enable by a digital input

The figure below shows an example for periodic synchronization at signal N depending on the status of a digital input:

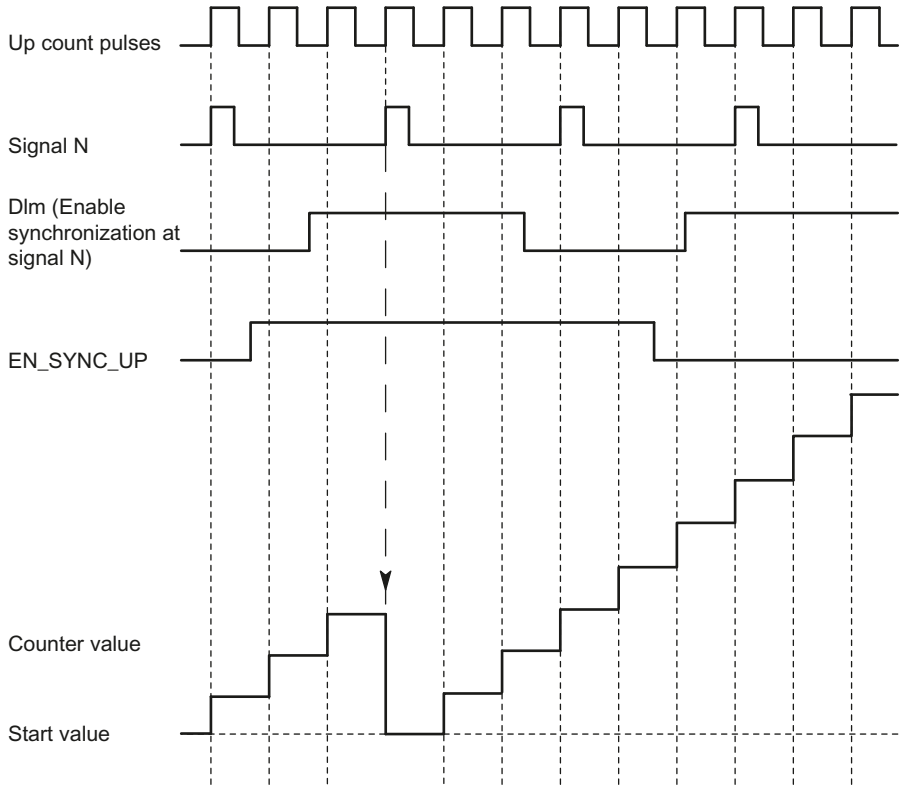

As long as synchronization for counting up is enabled and the corresponding digital input is active, the counter is synchronized at each signal N. If one of the conditions is not met, the counter is not synchronized at the signal N.

Comparison values (S7-1500)

Comparison values and outputs (S7-1500)

Description

You can specify two comparison values to control both digital outputs of the channel independently of the user program:

- Comparison value 0 for digital output DQ0
- Comparison value 1 for digital output DQ1

Depending on the operating mode and the encoder used, define two position, counter or measured values as comparison value. The comparison values are configurable and can be modified during runtime using the user program.

Note

DQ0 of a counter of a Compact CPU

With a Compact CPU, the respective digital output DQ0 is available via the feedback interface, but not as a physical output.

Switching digital outputs from the user program

The control bits (Page 166) TM_CTRL_DQ0 and TM_CTRL_DQ1 are used to control use of the digital outputs.

If TM_CTRL_DQm is set to 0, you can control the relevant digital output from the user program with the control bit SET_DQm regardless of the configured technological function. If TM_CTRL_DQm is set to 1, the technological function of the controller of the respective digital output is enabled.

The status of a digital output is indicated by the respective STS_DQm feedback bit.

Switching at comparison values with counter value as reference (S7-1500)

The comparison values are compared with the current counter value. If the counter value meets the assigned comparison condition and the technological function of the associated digital output is enabled, the digital output is set. If you assign "Between comparison value 0 and 1" for digital output DQ1, both comparison values affect DQ1.

You can make switching for a digital output dependent on one of the following comparison events:

Setting between comparison value and high counting limit

The digital output is set to 1 if:

Comparison value <= counter value <= high counting limit

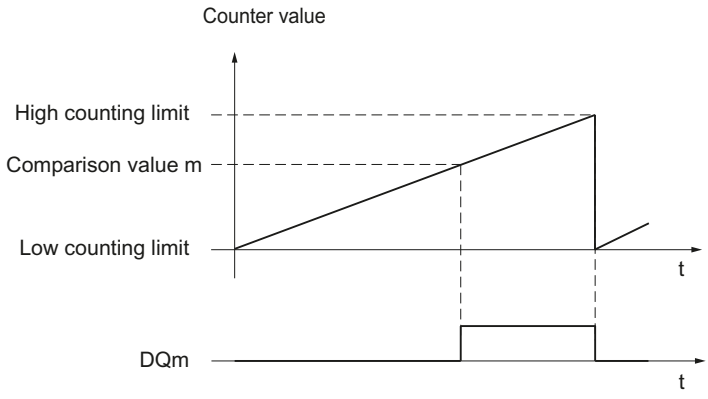

The comparison event is independent of the count direction.

Setting between comparison value and low limit

The digital output is set to 1 if:

Low counting limit <= counter value <= comparison value

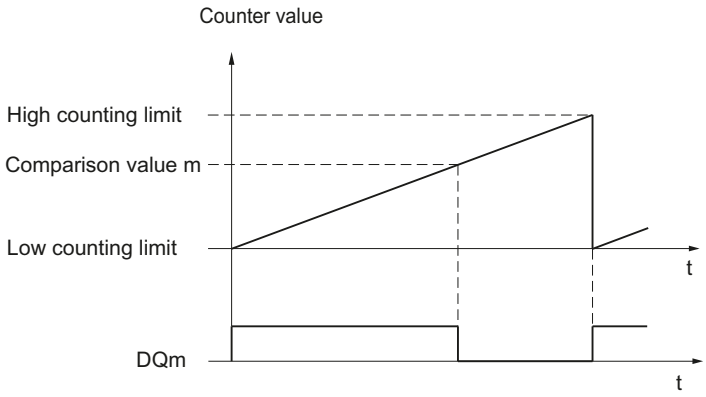

The comparison event is independent of the count direction.

Setting between comparison value 0 and comparison value 1

The comparison event can be configured for the digital output DQ1 if "Use by user program" has been configured for the digital output DQ0. Comparison value 1 must be greater than Comparison value 0.

DQ1 is set to 1 if:

Comparison value 0 <= counter value <= comparison value 1

The figure below shows an example of the comparison event when counting in an upwards direction:

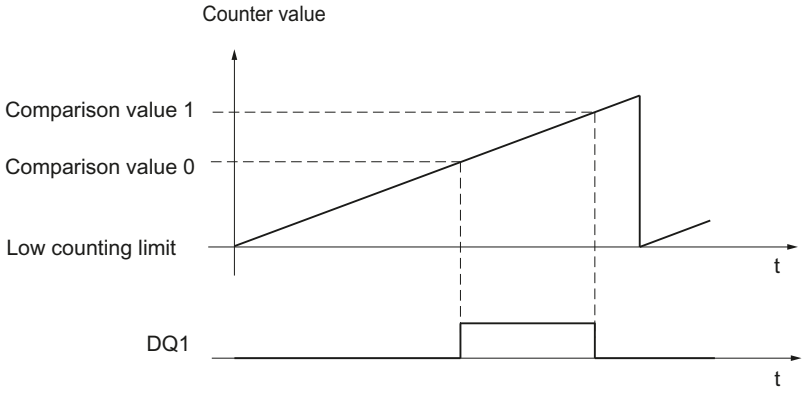

Setting at comparison value for one pulse duration

The respective digital output is set to 1 for a specified period of time when the following conditions are fulfilled:

- Counter value = comparison value
- Current count direction = configured count direction for the comparison event

The figure below shows an example of the comparison event when counting in an upwards direction:

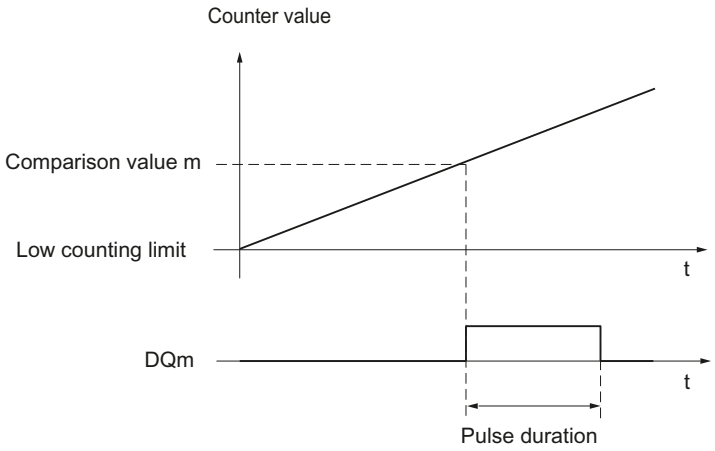

The figure below shows an example of the comparison event when counting in a downward direction:

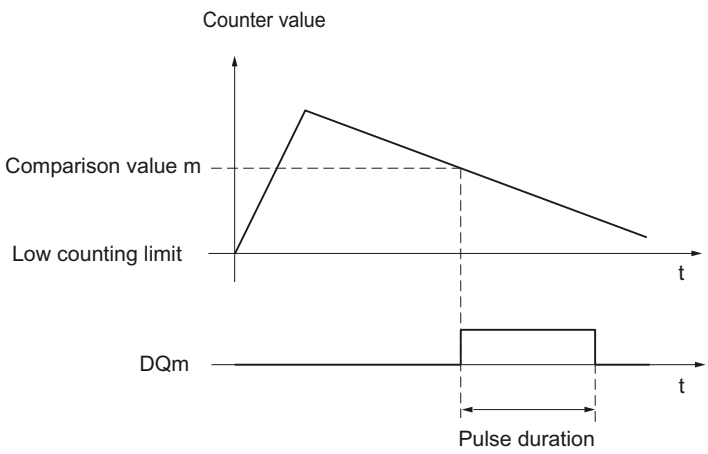

To repeat the comparison event, the counter value must change and then correspond to the respective comparison value again.

If the pulse duration has been defined as "0" and the counter value is equal to the comparison value, the digital output is set to 1 until the next count pulse:

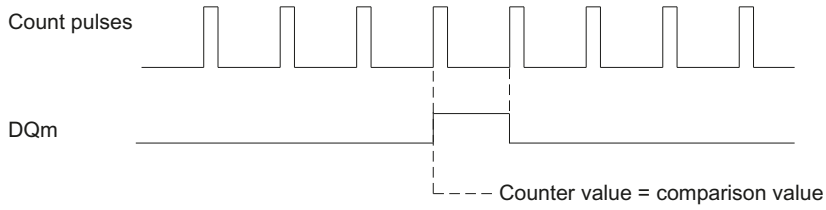

Note

This comparison event switches the relevant digital output if a count pulse reaches the comparison value. The digital output does not switch when the counter value is set, by synchronization for example.

Setting by the user program up to comparison value

You can set the respective digital output to 1 (edge) by setting the control bit (Page 166) SET_DQm. The respective digital output is set to 0 by any of the following events:

- Match of the counter value and the comparison value in the configured direction of the comparison event
- Reset of the corresponding SET_DQm control bit.

The figure below shows an example of the comparison event when counting in an upwards direction:

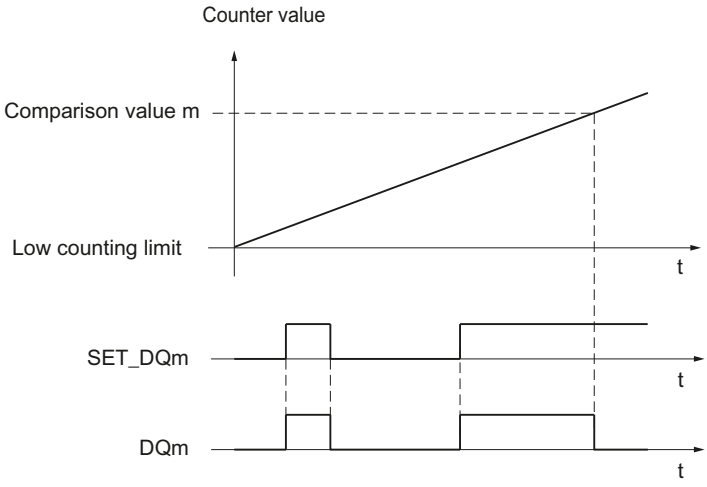

You can disable the digital output before the counter value reaches the comparison value by setting the control bit SET_DQm to 0.

Note

If the comparison value is reached in the configured count direction, the feedback bit EVENT_CMPm is set independently of the state of the control bit SET_DQm.

The comparison event switches a digital output when a count pulse reaches the respective comparison value. The digital output does not switch when the counter value is set, by synchronization for example.

Switching at comparison values with position value (SSI absolute value) as reference (S7-1500)

The comparison values are compared with the current position value. If the position value meets the assigned comparison condition and the technological function of the associated digital output is enabled, the digital output is set. If you assign "Between comparison value 0 and 1" for digital output DQ1, both comparison values affect DQ1.

If you use a 32 bit position value for the comparison function, the position value is interpreted as DINT.

You can make switching for a digital output dependent on one of the following comparison events:

Setting between comparison value and high limit

The high limit corresponds to the maximum position value.

The digital output is set to 1 if:

Comparison value <= position value <= maximum position value

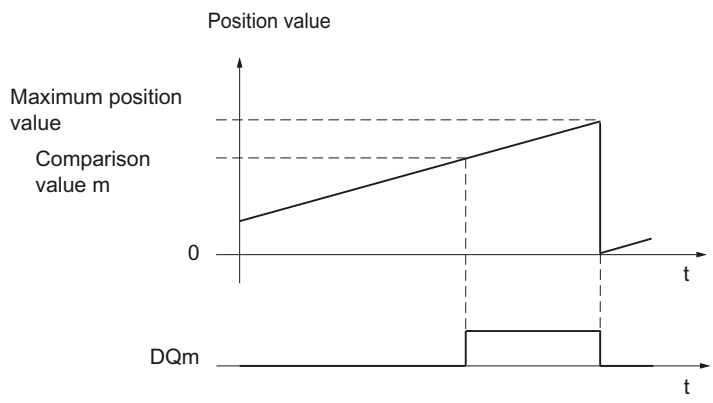

The comparison event is independent of the direction of the position value change. The maximum position value depends on the resolution of the SSI absolute encoder.

Setting between comparison value and low limit

The low limit corresponds to the position value "0".

The digital output is set to 1 if:

0 <= position value <= comparison value

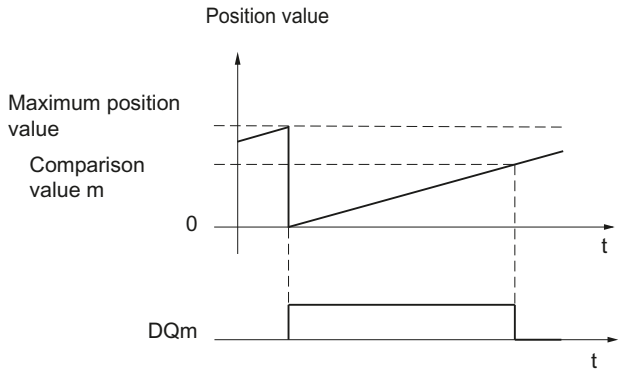

The comparison event is independent of the direction of the position value change.

Setting between comparison value 0 and comparison value 1

The comparison event can be configured for the digital output DQ1 if "Use by user program" has been configured for the digital output DQ0. Comparison value 1 must be greater than Comparison value 0.

DQ1 is set to 1 if:

Comparison value 0 <= position value <= comparison value 1

The figure below shows an example of the comparison event when counting up:

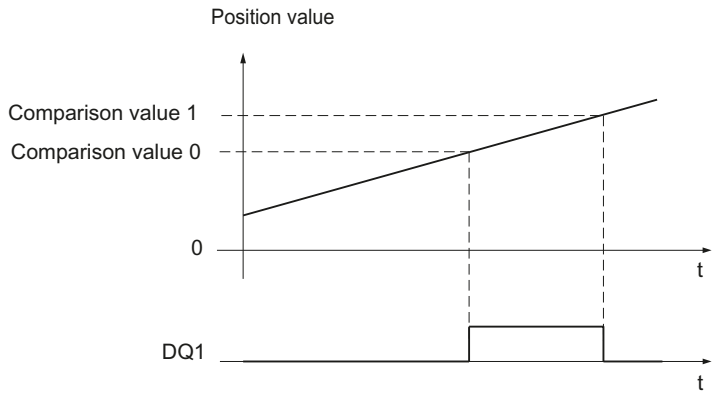

Setting at comparison value for one pulse duration

The respective digital output is set to 1 for a specified period of time when the following conditions are fulfilled:

- Matching of the position value and comparison value or crossing of the comparison value
- Current direction of the position value change = assigned direction for the comparison event

The figure below shows an example of the comparison event when counting up:

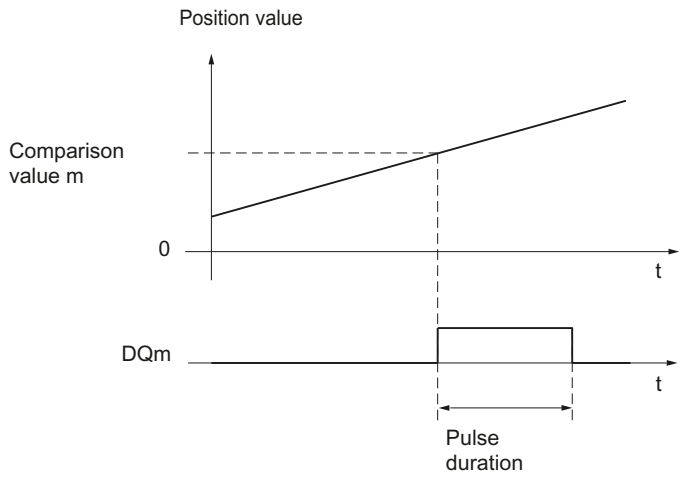

The figure below shows an example of the comparison event when counting down:

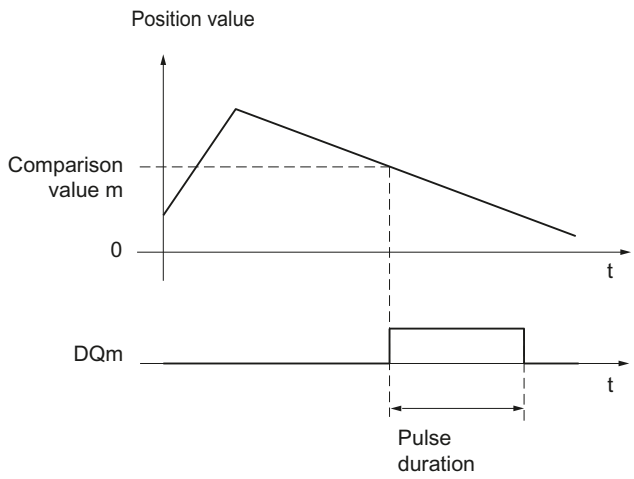

To repeat the comparison event, the position value must change and then correspond to or cross the respective comparison value again.

Setting by the user program up to comparison value

You can set each digital output to 1 (edge) by setting the control bit (Page 166) SET_DQm. The respective digital output is set to 0 by any of the following events:

- Matching of the position value and the comparison value or crossing of the comparison value in the configured direction of the comparison event
- Resetting of the SET_DQm control bit.

The figure below shows an example of the comparison event when counting up:

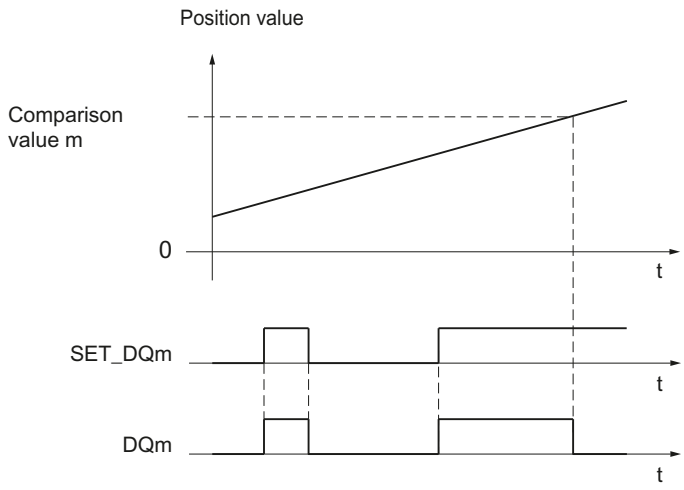

You can disable the digital output before the position value corresponds to or exceeds the comparison value by setting the control bit SET_DQm to 0.

Note

If the comparison value is reached or exceeded in the assigned direction, feedback bit EVENT_CMPm is set independently of the status of control bit SET_DQm.

Switching at comparison values with measured value as reference (S7-1500)

The comparison values are compared with the current measured value. If the measured value meets the configured comparison condition and the technological function of the corresponding digital output is enabled, the digital output is set. If you configure "Between comparison value 0 and 1" or "Not between comparison value 0 and 1" for digital output DQ1, both comparison values affect DQ1.

You can make switching for a digital output dependent on one of the following comparison events:

Setting above the comparison value

The digital output is set to 1 if:

Measured value >= comparison value

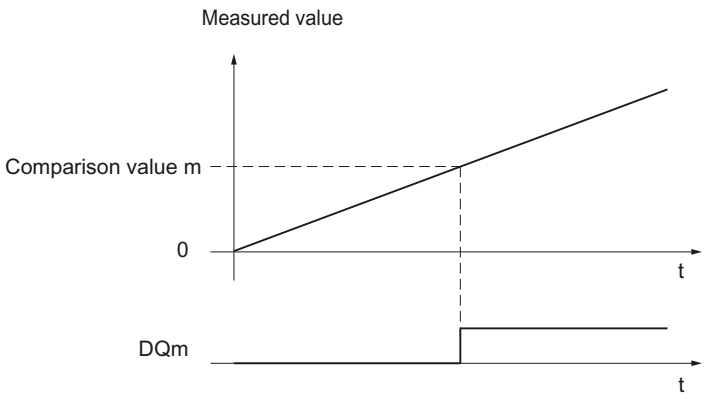

Setting below the comparison value

The digital output is set to 1 if:

Measured value <= comparison value

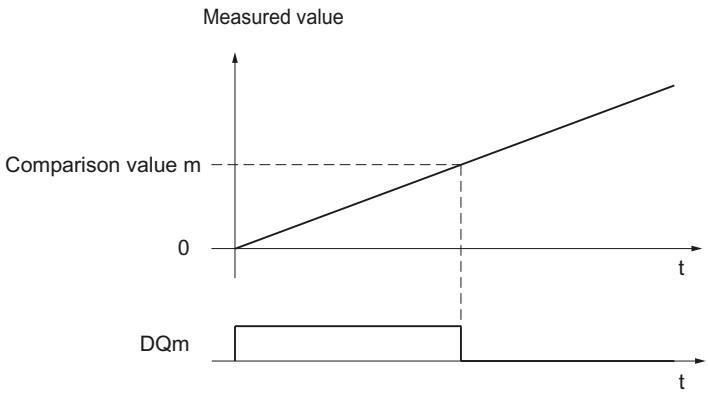

Setting between comparison value 0 and comparison value 1

The comparison event can be configured for the digital output DQ1 if "Use by user program" has been configured for the digital output DQ0. Comparison value 1 must be greater than Comparison value 0.

DQ1 is set to 1 if:

Comparison value 0 <= measured value <= comparison value 1

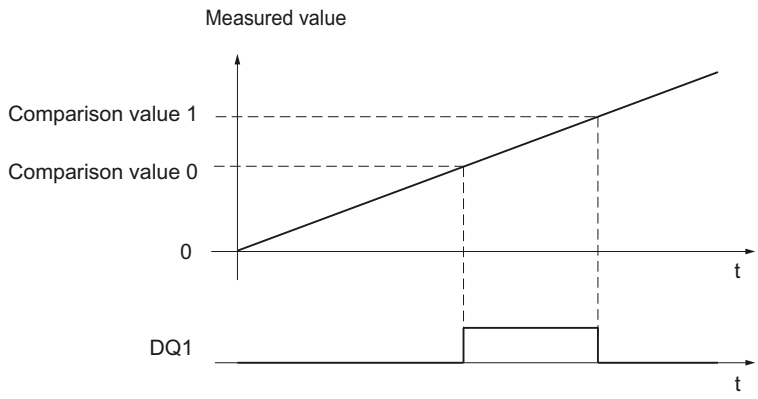

Not setting between comparison value 0 and comparison value 1

The comparison event can be configured for the digital output DQ1 if "Use by user program" has been configured for the digital output DQ0. Comparison value 1 must be greater than Comparison value 0.

DQ1 is set to 1 if:

Comparison value 1 <= measured value <= comparison value 0

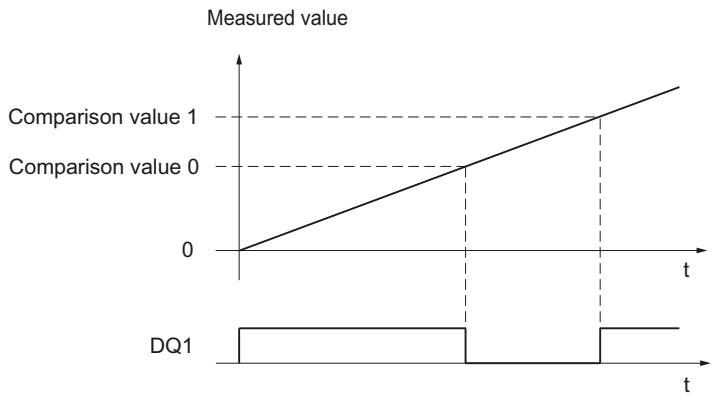

Measured value determination (S7-1500)

Overview of measuring functions (S7-1500)

The following highly accurate measuring functions are available (accuracy up to 100 ppm):

Measurement type (Page 50)	Description
Frequency measurement	The mean frequency is calculated at set measuring intervals on the basis of the time profile of the count pulses or position value changes and returned in Hertz as floating point number.
Period measurement	The mean period duration is calculated at set measuring intervals on the basis of the time profile of the count pulses or position value changes and returned in seconds as floating point number.
Velocity measurement	The mean velocity is calculated at set measuring intervals on the basis of the time profile of the count pulses or position value changes and other parameters, and returned in the configured unit of measurement.

Measured values and counter values are available concurrently in the feedback interface.

Update time

You can configure the interval at which the technology module updates the measured values cyclically as the update time. Setting longer update time intervals allows uneven measured variables to be smoothed and increases measuring accuracy.

Gate control for incremental and pulse encoders

Opening and closing the internal gate defines the period of time during which the count pulses are captured. The update time is asynchronous to the opening of the gate, which means that the update time is not started when the gate is opened. After the internal gate is closed, the last measured value captured is still returned.

Measured value determination with incremental or pulse encoder (S7-1500)

Measuring range (TM Count and TM PosInput)

The measuring functions have the following measuring limits:

Measurement type	Low measuring range limit	High measuring range limit
Frequency measurement	0.04 Hz	800 kHz* / 4 MHz**
Period measurement	1.25 µs* / 0.25 µs**	25 s
Velocity measurement	Depending on the configured number of "increments per unit" and the "time base for velocity measurement"	

* Applies to 24 V incremental encoders and "quadruple" signal evaluation.

** Applies to RS422 incremental encoders and "quadruple" signal evaluation.

All measured values are returned as signed values. The sign indicates whether the counter value increased or decreased during the relevant time period.

Measuring range (Compact CPU)

The measuring functions have the following measuring range limits:

Measurement type	Low measuring range limit	High measuring range limit
Frequency measurement	0.04 Hz	400 kHz*
Period measurement	2.5 µs*	25 s
Velocity measurement	Depends on the configured number of "Increments per unit" and the "Time base for velocity measurement"	

* Applies to 24 V incremental encoders and "quadruple“ signal evaluation.

All measured values are returned as signed values. The sign indicates whether the counter value increased or decreased during the relevant time period.

Measuring principle

The technology module assigns a time value to each count pulse. The measuring interval is defined as the time between each last count pulse before and during the previous update time. The measuring interval and the number of pulses in the measuring interval are evaluated to calculate measured variables.

If there is no count pulse within an update time, the measuring interval is dynamically adjusted. In this case, a pulse is assumed at the end of the update time and the measuring interval is calculated as the time between that point and the last pulse which occurred. The number of pulses is then 1.

The feedback bit STS_M_INTERVAL indicates whether a count pulse occurred in the previous measuring interval. This allows for a differentiation between an assumed and an actual count pulse.

The following figures show the principle of measurement and the dynamic adjustment of the measuring interval:

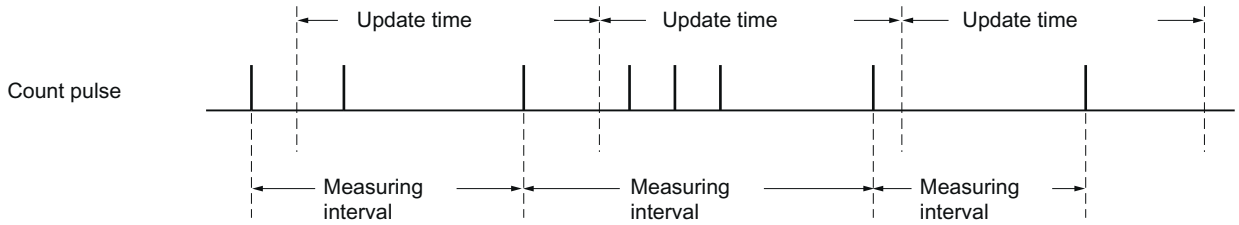

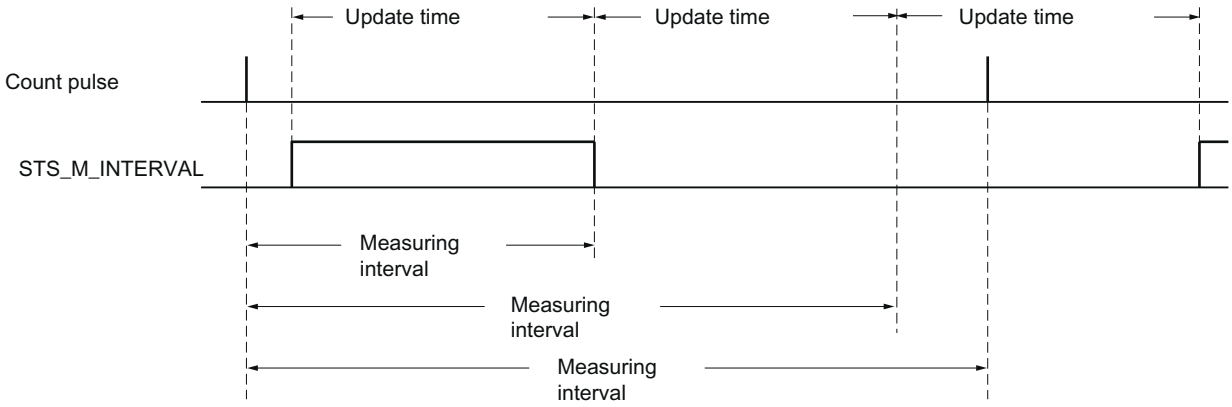

Frequency measurement

A value "0" is returned until the first measured value is available.

The measurement process begins with the first pulse detected once the internal gate has been opened. The first measured value can be calculated after the second pulse at the earliest.

The measured value is updated in the feedback interface (Page 168) upon completion of each update time. If the internal gate is closed, measuring stops and the measured value is no longer updated.

The figures below shows an example of frequency measurement with an update time of 1 s:

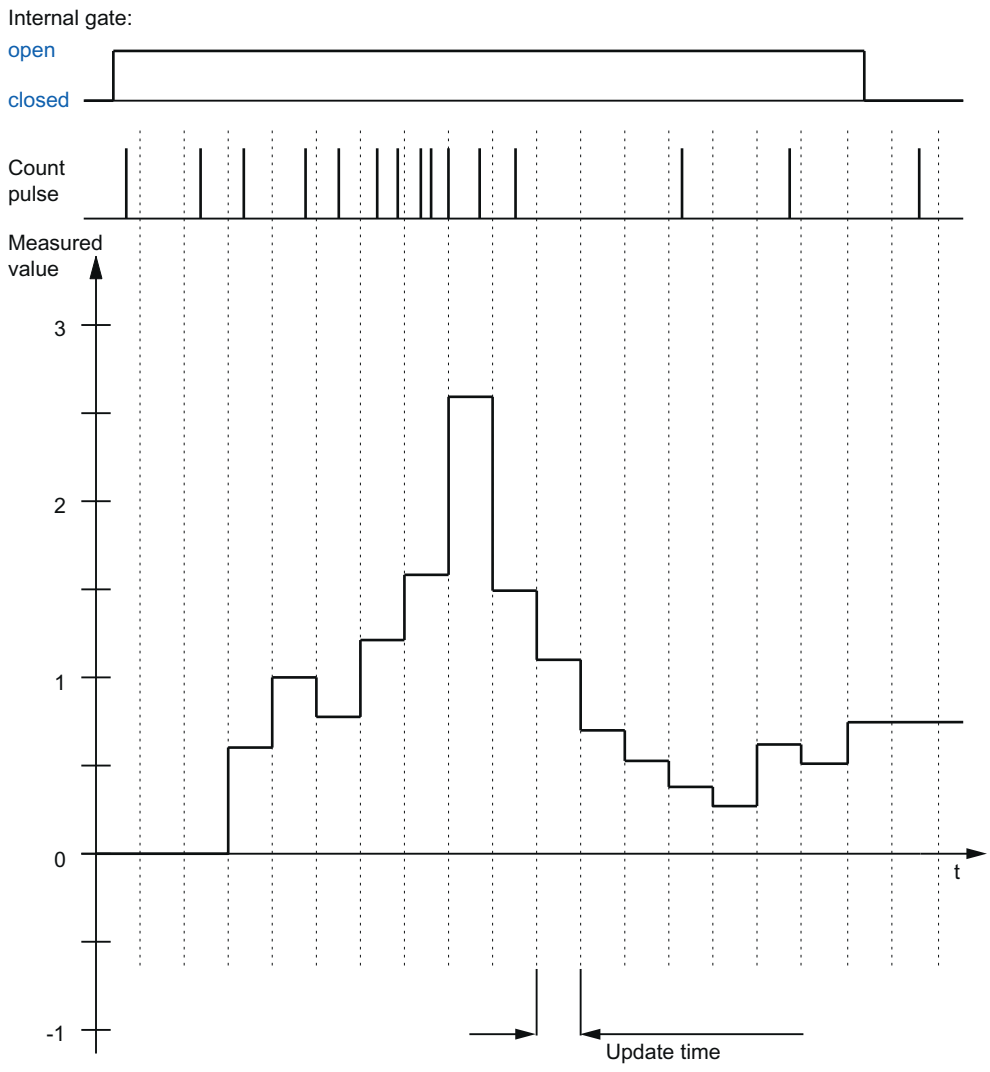

Period measurement

The reciprocal of the frequency is output as the measured value for period measurement.

A value "25 s" is returned until the first measured value is available.

Velocity measurement

The normalized frequency is output as the measured value in velocity measurement. You can configure the scaling using the time basis and the number of increments that your encoder delivers per unit.

Example:

Your encoder delivers 4000 increments per meter. The velocity is to be measured in meters per minute.

In this case, you need to configure 4000 Increments per unit and a time basis of one minute.

Measured value determination with SSI absolute encoder (S7-1500)

Measuring range SSI absolute encoder

The measuring functions have the following measuring limits:

Measurement type	Low measuring range limit	High measuring range limit
Frequency measurement	0,04 Hz	4 MHz
Period measurement	0,25 µs	25 s
Velocity measurement	Depending on the configured number of "increments per unit" and the "time base for velocity measurement"	

All measured values are returned as signed values. The sign indicates whether the position value increased or decreased during the relevant time period.

Measuring principle

The technology module assigns a time value to each SSI frame. The measuring interval is defined as the time between the last SSI frame with a change of position value before and during the previous update time. The measuring interval and the total change in position value in the measuring interval are evaluated to calculate a measured variable. The total change in position value in a measuring interval corresponds to the number of encoder increments in the same measuring interval.

If there is no change in position value within an update time, the measuring interval is dynamically adjusted. In this case, a change in position value is assumed at the end of the update time and the measuring interval is calculated as the time between that point and the last SSI frame with a change in position value. The change in position value is then 1.

The feedback bit STS_M_INTERVAL indicates whether a change in position value occurred in the previous measuring interval. This allows for a differentiation between an assumed and an actual change in position value. If the technology module cannot calculate measured values because the measuring range limit has been violated, the feedback bit STS_M_INTERVAL is not set.

Frequency measurement

The value "0.0" is reported in the time up to the first available measured value.

The measuring process begins with the first detected change in position value. The first measured value can be calculated after the second detected change in position value at the earliest.

The measured value is updated in the feedback interface (Page 168) upon completion of each update time.

The figures below shows an example of frequency measurement with an update time of 1 s:

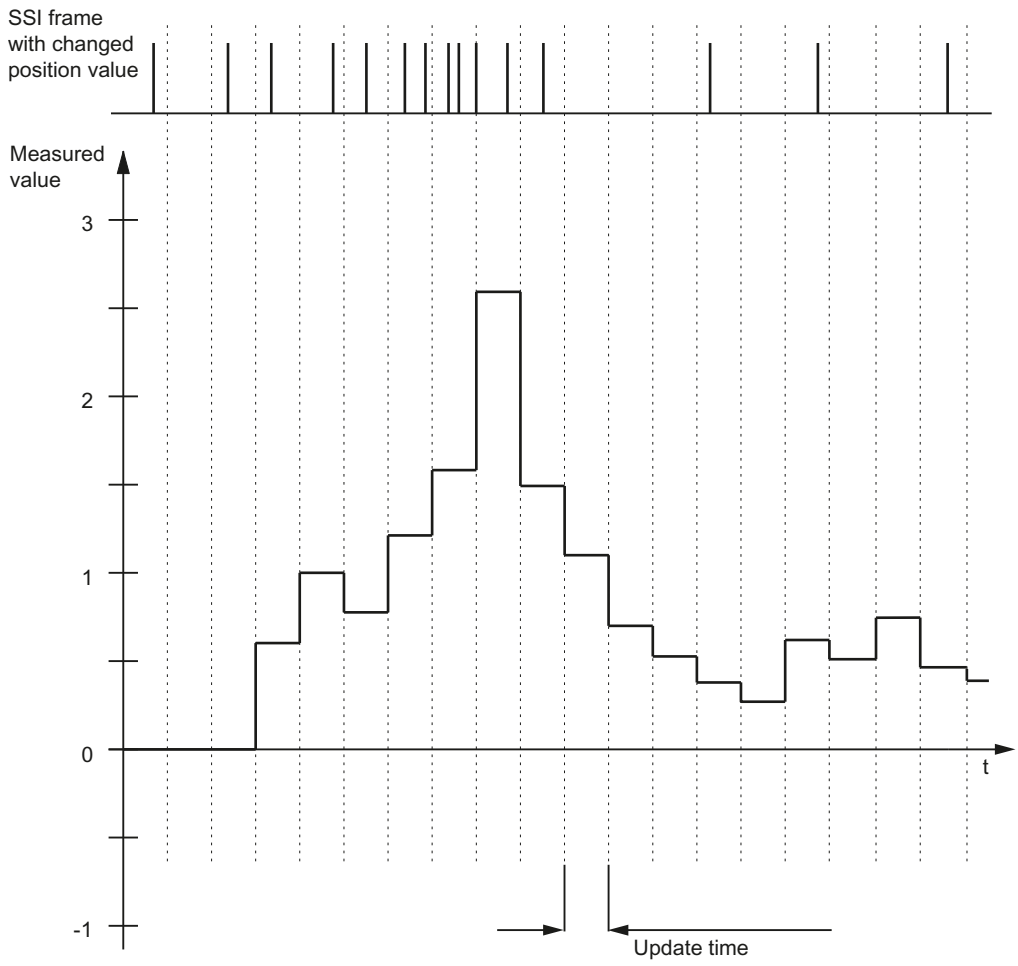

Period measurement

The reciprocal of the frequency is output as the measured value for period measurement.

A value "25 s" is returned until the first measured value is available.

Velocity measurement

The normalized frequency is output as the measured value in velocity measurement. You can configure the scaling using the time basis and the number of increments that your encoder delivers per unit.

Example:

Your SSI absolute encoder operates with a resolution of 12 bits per revolution and performs 4096 increments per revolution. The velocity should be measured in revolutions per minute.

In this case, you need to configure 4096 Increments per unit and a time basis of one minute.

Note

Excessive encoder speed can provide wrong direction of rotation.

If an SSI absolute encoder rotates so quickly that more than half of the value range is covered within one module cycle[1], the velocity and direction of rotation can not be determined correctly anymore. This can result in an incorrect function at:

- DQ functions
- Feedback bits EVENT_OFLW, EVENT_UFLW, EVENT_ZERO, EVENT_CMP0, EVENT_CMP1 and STS_DIR

[1] Non isochronous mode: 500 µs; isochronous mode: PROFINET cycle time

Note

If you are using an SSI absolute encoder whose value range does not correspond to the power of 2, the calculated speed measured value may be incorrect at the moment of the overflow.

Hysteresis (S7-1500)

Hysteresis with incremental or pulse encoder (S7-1500)

Description

Hysteresis allows you to specify a range around the comparison values within which the digital outputs are not to be switched again until the counter value has gone outside this range.

Slight movements by the encoder can result in the counter value fluctuating around a certain value. If a comparison value or a counting limit lies within this fluctuation range, the corresponding digital output is switched on and off with corresponding frequency if hysteresis is not used. Hysteresis prevents this unwanted switching, and configured hardware interrupts when a comparison event occurs.

The hysteresis becomes active when the respective comparison value is reached by a count pulse. If the counter value is set to the start value during an active hysteresis, the hysteresis becomes inactive.

Regardless of the hysteresis value, the hysteresis range ends at the low/high counting limits.

Function principle

The figure below shows an example for the hysteresis with the following configuration:

- Setting of a digital output between comparison value and high counting limit
- Comparison value = 5
- Hysteresis = 0 or 2 (gray background)

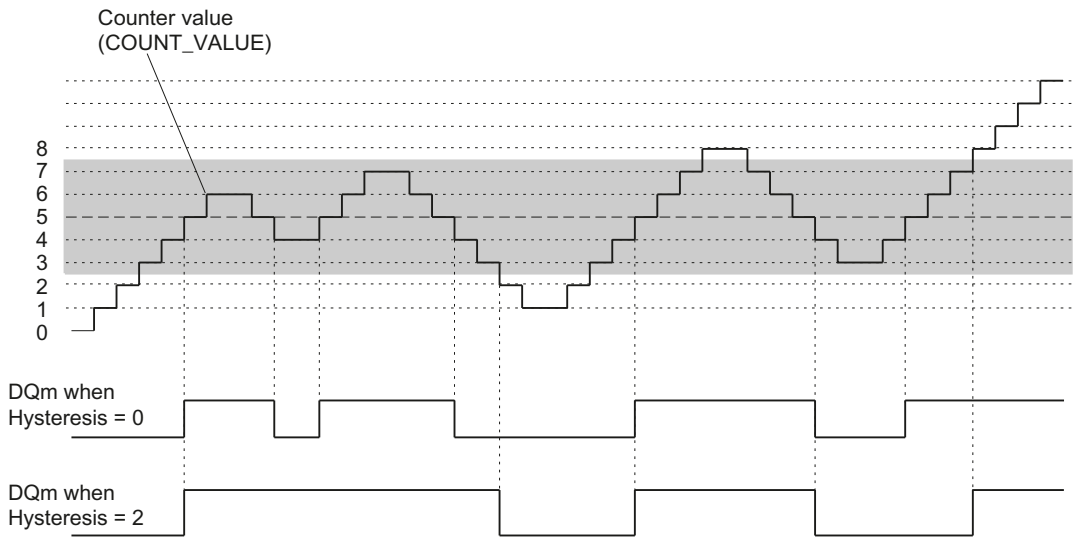

Hysteresis is enabled when the counter value 5 is reached. When the hysteresis is active, the comparison result remains unchanged. Hysteresis is disabled when the counter value 2 or 8 is reached.

The figure below shows an example for the hysteresis with the following configuration:

- Setting at comparison value for one pulse duration
- Comparison value = 5
- Comparison in both count directions
- Hysteresis = 0 or 2 (gray background)

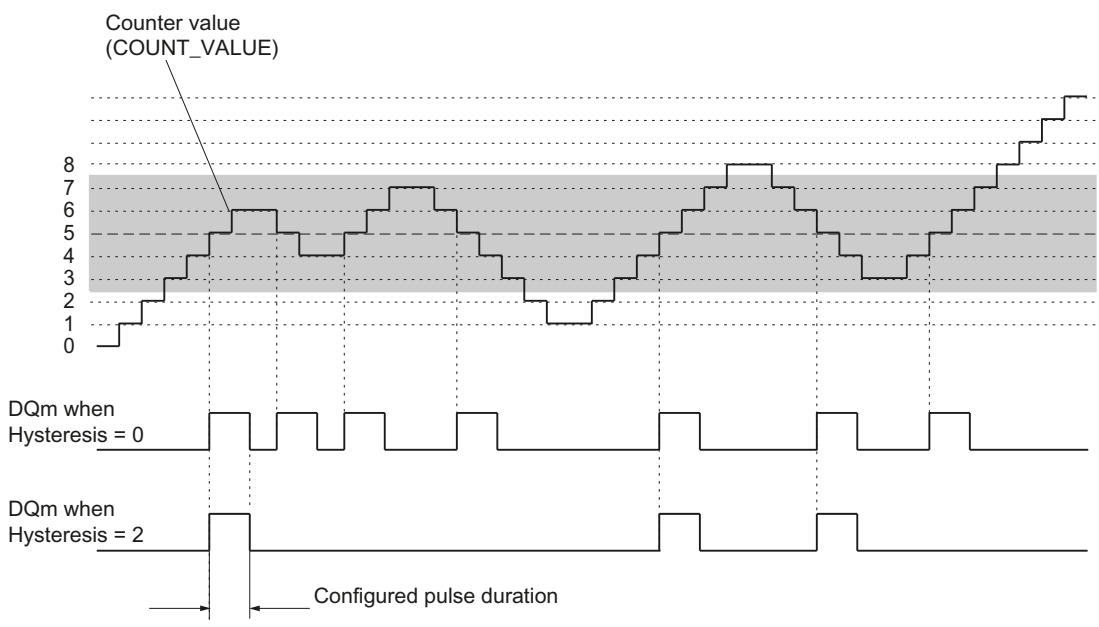

Hysteresis with SSI absolute encoder (S7-1500)

Description

Hysteresis allows you to specify a range around the comparison values within which the digital outputs are not to be switched again until the position value has gone outside this range.

Slight movements by the encoder can result in the position value fluctuating around a certain value. If a comparison value, the minimum or maximum position value, lies within the fluctuation range, the associated digital output is switched on and off if a hysteresis is not used. Hysteresis prevents this unwanted switching, and configured hardware interrupts when a compare event occurs.

Regardless of the hysteresis value, the hysteresis range ends at the respective minimum or maximum position value.

Function principle

The figure below shows an example for the hysteresis with the following parameter assignment:

- Setting of a digital output between comparison value and high limit
- Comparison value = 10
- Hysteresis = 0 or 2 (gray background)

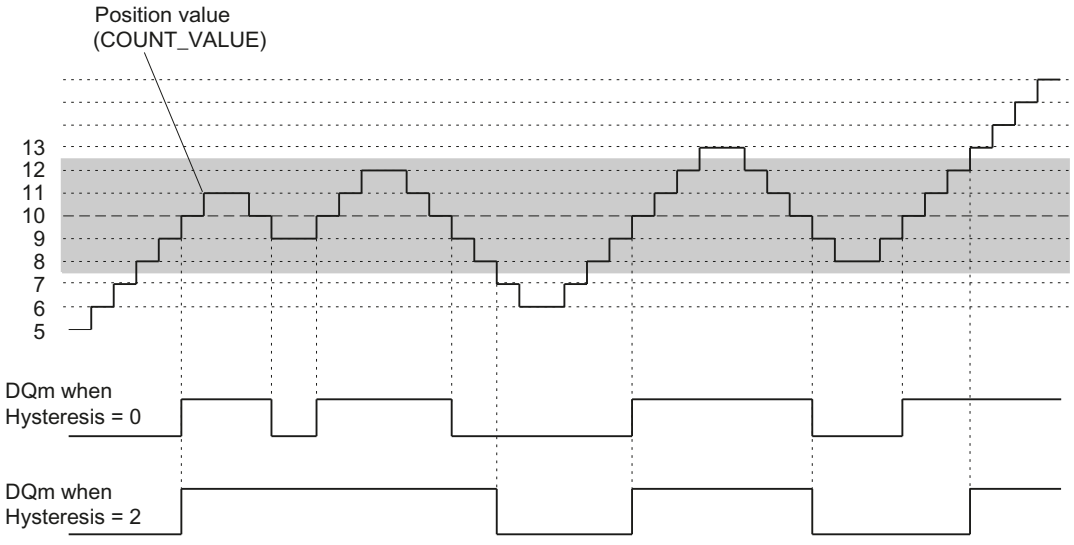

Hysteresis is enabled when the position value 10 is reached. When the hysteresis is active, the comparison result remains unchanged. Hysteresis is disabled when the position values 7 or 13 are reached.

The figure below shows an example for the hysteresis with the following parameter assignment:

- Setting at comparison value for one pulse duration
- Comparison value = 10

- Comparison in both directions of position value changes
- Hysteresis = 0 or 2 (gray background)

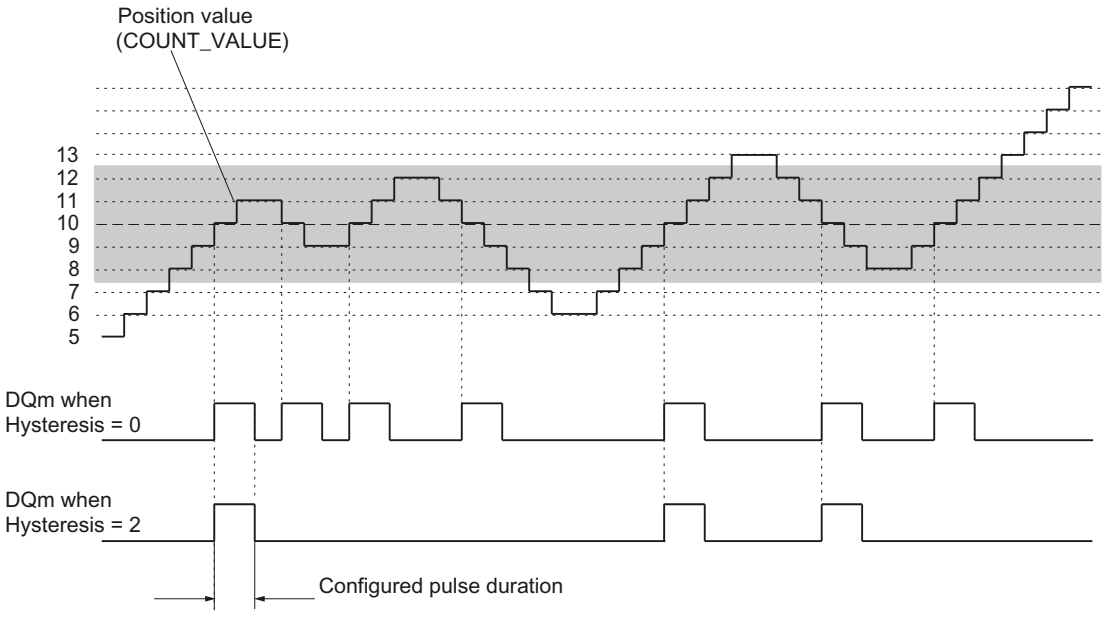

Interrupts (S7-1500)

Hardware interrupt

The technology module can trigger a hardware interrupt in the CPU if, for example, a comparison event occurs; in the event of overflow or underflow; in the event of a zero crossing of the counter and/or of a change in count direction (direction reversal). You can specify which events are to trigger a hardware interrupt during operation.

Diagnostic interrupt

The technology module can trigger diagnostic interrupts in the event of errors. You enable the diagnostic interrupts for certain errors in the device configuration. Refer to the device manual for the technology module to learn about the events that can trigger a diagnostic interrupt during operation.

Position detection for Motion Control (S7-1500)

Description

You can use the technology module for example with an incremental encoder for the following axis technology objects of S7-1500 Motion Control for position detection :

- TO_PositioningAxis
- TO_SynchronousAxis
- TO_ExternalEncoder

When using an incremental or pulse encoder, the position detection is based on the counting function of the technology module. With an SSI absolute encoder, the absolute value is read in via a synchronous, serial interface and prepared according to the parameter assignment and made available for S7-1500 Motion Control.

The range of functions of the technology module has the following limitations in this case:

- Counter behavior not configurable
- No functions for digital inputs available apart from measuring input function
- No comparison functions for digital outputs available
- No hardware interrupts available

In the device configuration of the technology module in STEP 7 (TIA Portal), select the "Position input for technology object "Motion Control"" operating mode and use the corresponding technology object in the program. This reduces the configuration options to the parameters that are essential. For TM Count or TM PosInput, the mode automatically applies to all channels of the technology module. For a Compact CPU, the mode automatically applies to the respective channel.

In this operating mode you can use the (TO_MeasuringInput) measuring input technology object to execute a measuring input function with a hardware digital input. To do this select the measuring input type "Measuring via PROFIdrive telegram" in the measuring input technology object and the value "1" as the number of the measuring input.

You can find information about further configuring in the help for the axis technology objects and the measuring input technology object of S7-1500 Motion Control.

Encoder signals (S7-1500)

24 V and TTL count signals (S7-1500)

24 V and TTL incremental encoder count signals

The 24 V incremental encoder returns the 24 V signals A, B, and N to the technology module. The A and B signals are phase-shifted by 90°. You can also connect incremental encoders without an N signal.

A 24 V incremental encoder uses the A and B signals for counting. If configured accordingly, the N signal is used for setting the counter to the start value or for saving the current counter value to the Capture value.

The figure below shows an example of the time profile of the signals of an 24 V incremental encoder:

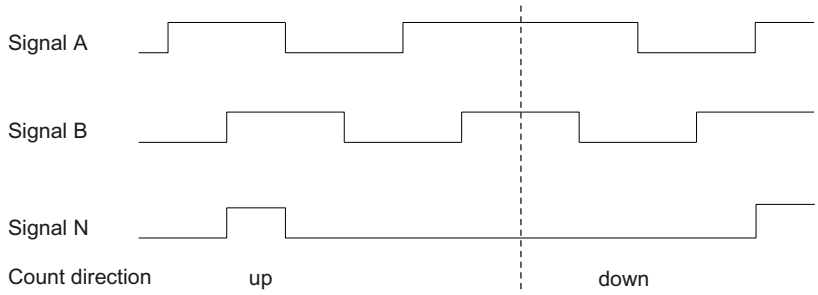

The technology module detects the count direction by evaluating the sequence of edges of the A and B signals. You can specify an inversion of the count direction.

24 V and TTL pulse encoder count signals without/with direction signal

The encoder, for example an initiator (BERO) or a light barrier, returns only a count signal that is connected to terminal A of the counter.

In addition, you can connect a signal for direction detection to terminal B of the counter. In case of a high level the direction signal is counted backwards. If your encoder does not return a corresponding signal, you can specify the count direction with the user program using the control interface.

The figure below shows an example of the time profile of the signals of a 24 V pulse encoder with direction signal and the resulting count pulses:

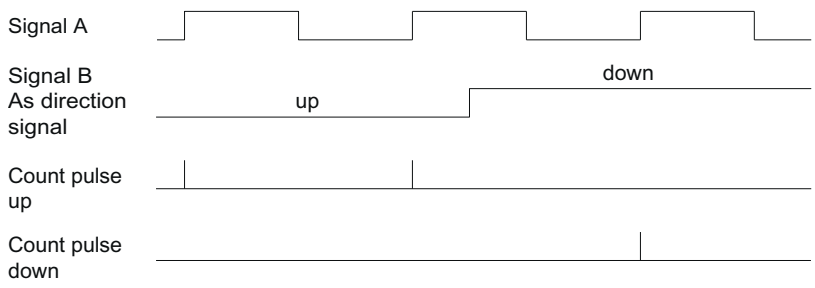

24 V and TTL pulse encoder count signals with Up/Down count signal

The Up count signal is connected to terminal A. The Down count signal is connected to terminal B.

The figure below shows an example of the time profile of the signals of a pulse encoder with Up/Down count signal and the resulting count pulses:

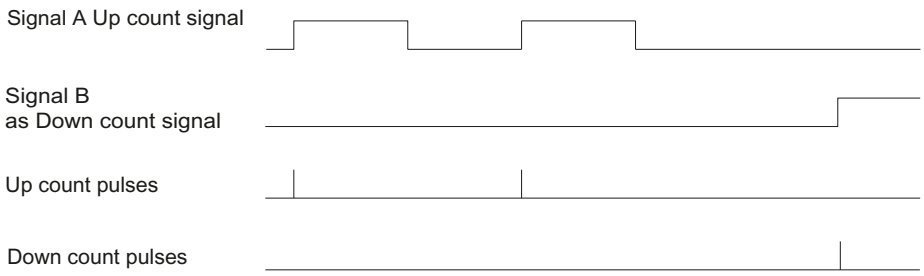

Sourcing output/sinking output for 24 V counter signals (TM Count)

You can connect the following encoders/sensors to the counter inputs:

- Sourcing output:
  The A, B, and N inputs are wired to 24VDC .
- Sinking output:
  The A, B, and N inputs are wired to ground M .
- Push-pull (sourcing and sinking output):
  The A, B, and N inputs are wired alternately to 24VDC and ground M .

Sourcing output for 24 V counter signals (Compact CPU)

You can connect the sourcing output and push-pull encoders or sensors to the counter inputs.

Monitoring of the encoder signals (TM Count and TM PosInput)

The signals of push-pull 24 V encoders are monitored for wire breaks by the technology module. TTL signals are monitored by the technology module for faulty supply voltages.

If you enable the diagnostic interrupt in the device configuration, the technology module triggers a diagnostic interrupt in the event of encoder signal errors.

RS422 count signals (S7-1500)

RS422 incremental encoder count signals

The RS422 incremental encoder sends the following differential signals to the technology module:

- +A and -A
- +B and -B
- +N and -N

The signal information for RS422 signals is encoded in the differential voltage between A and -A, B and -B, and +N and -N. The A and B signals are phase-shifted by 90°. You can also connect incremental encoders without an N signal.

RS422 incremental encoders use the A and B signals for counting. If configured accordingly, the N signal is used for setting the counter to the start value or for saving the current counter value as the Capture value.

The figure below shows an example of the time profile of the signals of an RS422 incremental encoder:

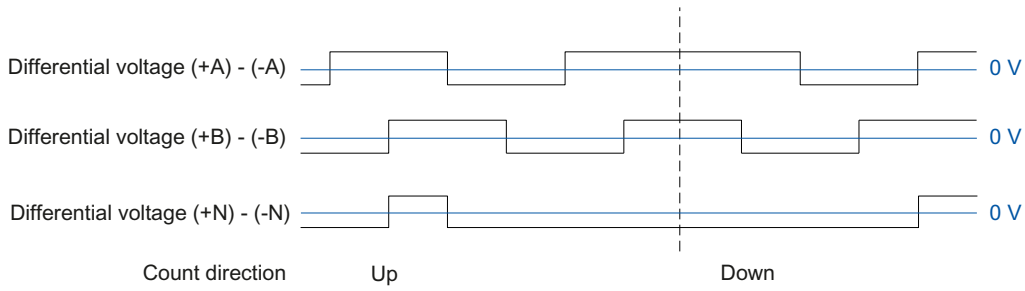

The technology module detects the count direction by evaluating the sequence of edges of the A and B signals. You can specify an inversion of the count direction.

Count signals of RS422 pulse encoders without/with direction signal

The encoder, for example a light barrier, only returns a count signal that is connected to terminal A.

You can also connect a signal for direction detection to terminal B. In case of a high level the direction signal is counted backwards. If your encoder does not return a corresponding signal, you can specify the count direction with the user program using the control interface.

The figure below shows an example of the time profile of the signals of a RS422 pulse encoder with direction signal and the resulting count pulses:

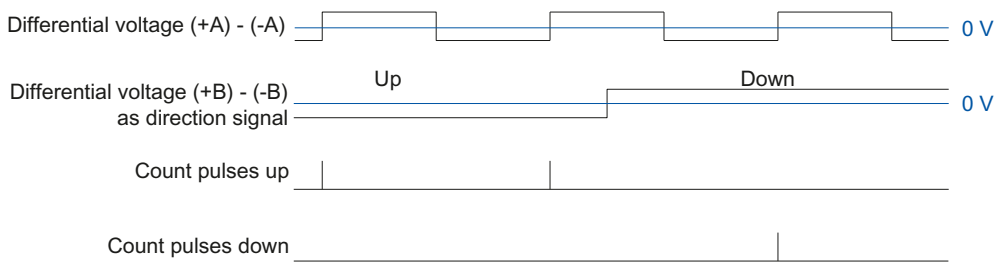

Count signals of RS422 pulse encoders with Up/Down count signal

The Up count signal is connected to the A terminals. The Down count signal is connected to the B terminals.

The figure below shows an example of the time profile of the signals of an RS422 pulse encoder with Up/Down count signal and the resulting count pulses:

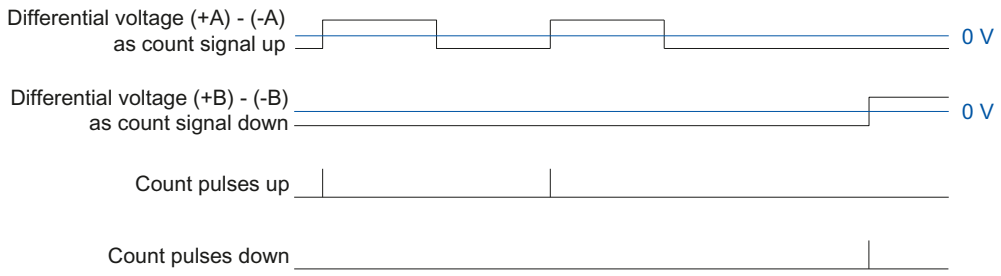

Monitoring of encoder signals

RS422 signals are monitored by the technology module for wire breaks, short-circuits, and incorrect supply voltage.

If you enable the diagnostic interrupt in the device configuration, the technology module triggers a diagnostic interrupt in the event of encoder signal errors.

SSI signals (S7-1500)

Signals from SSI absolute encoders

The SSI absolute encoder and the technology module communicate by means of the SSI data signals +D and -D and the SSI clock signals +C and -C. SSI uses the RS422 signal standard. The signal information is coded in the respective differential voltage between +C and -C as well as +D and -D.

Monitoring of the encoder signals and the SSI frames

The signals of an SSI absolute encoder are monitored for wire breaks, short-circuits and incorrect supply voltage. The technology module also monitors the SSI frames for errors.

If you enable the diagnostic interrupts in the device configuration, the technology module triggers a diagnostic interrupt in the event of encoder signal or SSI frame errors.

Signal evaluation of incremental signals (S7-1500)

Overview (S7-1500)

The technology module counter counts the edges of encoder signals A and B. For incremental encoders with phase-shifted signals A and B, you can select either single or multiple evaluation to improve the resolution.

You can configure the following signal evaluations:

- Single evaluation (Page 65)
- Double evaluation (Page 65)
- Quadruple evaluation (Page 66)

Note

The phase offset between the edges of signals A and B is evaluated. If no phase shift is identifiable, an encoder error (invalid transition of A/B signals) is reported via the ENC_ERROR feedback bit.

Single evaluation (S7-1500)

Single evaluation evaluates the rising and falling edge at signal A when signal B has a low level.

Count pulses in an upwards direction are generated with a rising edge at signal A during a low level at signal B. Count pulses in a downwards direction are generated with a falling edge at signal A during a low level of signal B.

The following figure shows an example for single evaluation of 24 V and TTL count signals:

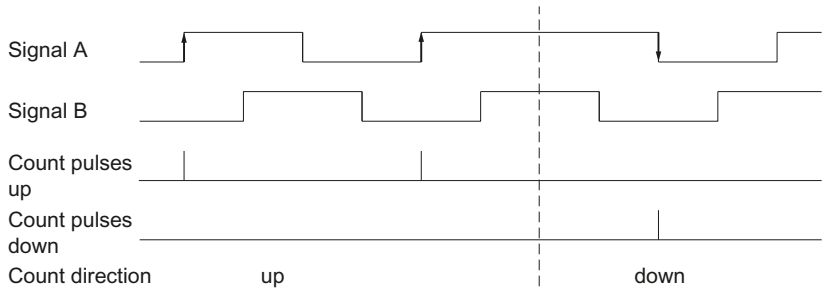

The following figure shows an example for single evaluation of RS422 count signals:

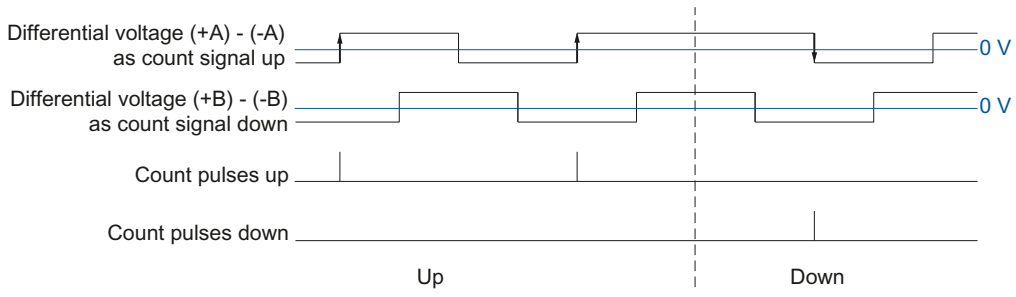

Double evaluation (S7-1500)

With double evaluation, the rising and falling edges of signal A are evaluated.

The edge direction of signal A and the level at signal B determines whether count pulses are generated in an upward or downward direction.

The following figure shows an example for double evaluation of 24 V and TTL count signals:

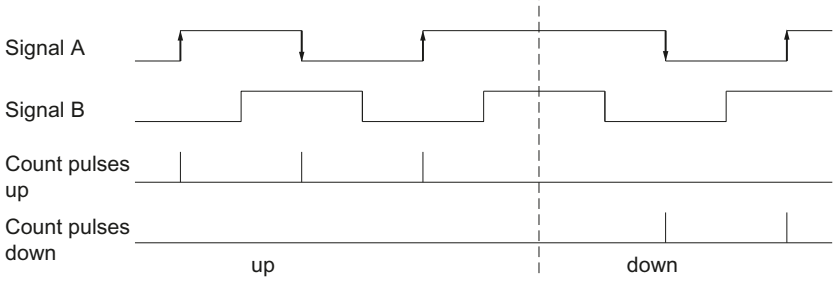

The following figure shows an example for double evaluation of RS422 count signals:

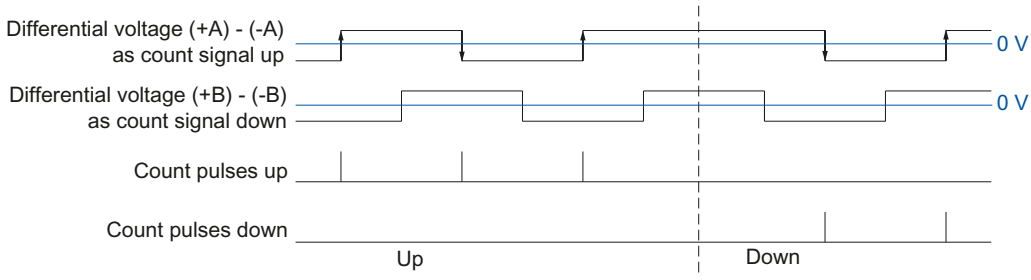

Quadruple evaluation (S7-1500)

With quadruple evaluation, the rising and falling edges of signals A and B are evaluated.

The edge direction of one signal and the level of the other signal determines whether count pulses are generated in an upward or downward direction.

The figure below shows an example for quadruple evaluation of 24 V and TTL count signals:

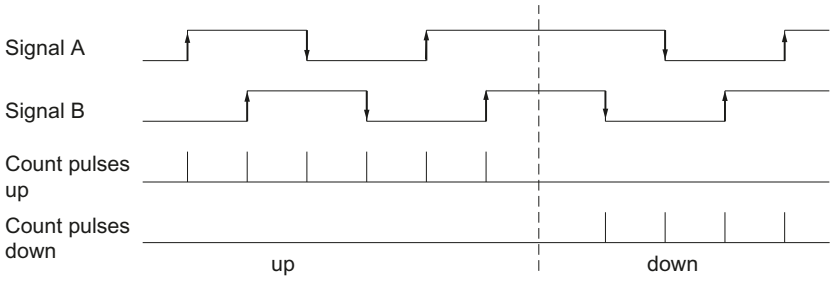

The figure below shows an example for quadruple evaluation of RS422 count signals:

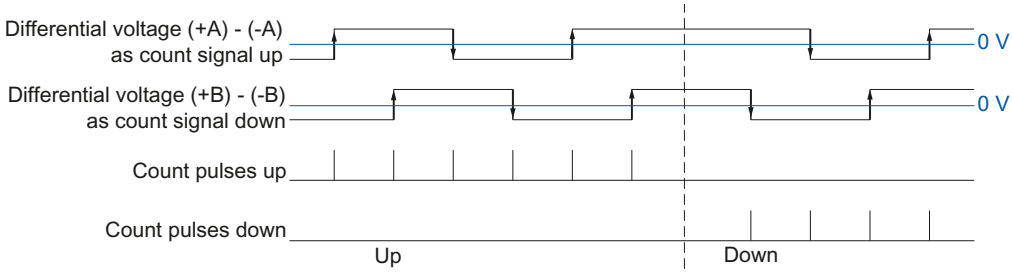

Clock synchronization (TM Count and TM PosInput) (S7-1500)

The technology module supports the system function "Isochronous mode". This system function enables position, count and measured values to be recorded in a defined system cycle.

In isochronous mode, the cycle of the user program, the transmission of the input and output data and the processing in the module are synchronized with each other. The output signals switch immediately if the relevant comparison condition is met. A change in the state of a digital input immediately affects the planned reaction of the technology module and changes the status bit of the digital input in the feedback interface.

In the case of operation with a "Counting and measurement" technology object use an OB of the type "Synchronous Cycle" (for example OB61). In the assigned OB the instruction High_Speed_Counter or SSI_Absolute_Encoder is called.

For position detection of a "Motion Control" technology object use the OB of the type "MC-Servo". When using the technology objects cam and cam track, isochronous mode is required. When using the measuring input technology object in connection with the hardware digital input DI1, no isochronous mode is required.

In case of manual operation use an OB of the "Synchronous Cycle" type (for example OB61). In the assigned OB the input and output data is processed.

Data processing

The data that was transmitted to the technology module in the current bus cycle via the control interface takes effect when it is processed in the internal technology module cycle. At the moment when the input data (T_i) are read, the position and counter value and, if appropriate, the measured value as well as status bits are detected and made available in the feedback interface for retrieval in the current bus cycle.

The update time for the measured value is synchronized in a suitable relationship to the system cycle and, if required, adapted in length. If you configure "0", the measured value is updated once per system cycle.

Isochronous mode parameter

In isochronous mode, the following parameters can have an effect on the isochronous parameters of the Sync domain.

- Filter frequency
- Frame length[1]
- Transmission rate[1]
- Monoflop time[1]
- Parity[1]

[1] Only when using an SSI absolute encoder

Because the isochronous parameters are not checked in RUN, overflows can occur when you change one or more of the named parameters in RUN: You prevent overflows by already selecting the option with the greatest time demand during the offline parameter assignment.

Additional information

For a detailed description of the isochronous mode refer to:

- The isochronous mode function manual available as a download from the Internet (https://support.industry.siemens.com/cs/ww/en/view/109755401).
- The PROFINET with STEP 7 function manual as a download from the Internet (https://support.industry.siemens.com/cs/ww/en/view/49948856).

1.1.1.3 Basics of counting (TM Timer DIDQ) (S7-1500)

Overview of applications (S7-1500)

Introduction

You configure the TM Timer DIDQ and assign its parameters with the configuration software.

The module's functions are controlled and monitored via the user program with the control and feedback interface.

System environment

The respective module can be used in the following system environments with its counter functions:

Applications	Components required	Configuration software	In the user program
Central operation with an S7-1500 CPU or 151xSP CPU	• S7-1500 automation system or ET 200SP CPU • TM Timer DIDQ	STEP 7 (TIA Portal): Device configuration and parameter setting with hardware configuration	Direct access to feedback interface of the technology module in the I/O data
Distributed operation with a S7-1500 CPU	• S7-1500 Automation System • ET 200 Distributed I/O System • TM Timer DIDQ		
Distributed operation with a S7-300/400 CPU	• S7-300/400 automation system • ET 200 distributed I/O system • TM Timer DIDQ	STEP 7 (TIA Portal): Device configuration and parameter setting with hardware configuration STEP 7: Device configuration and parameter setting with hardware configuration (only ET 200SP)	

Note

A description of the control and feedback interface is available in the device manual for the TM Timer DIDQ.

Counting with incremental encoder (S7-1500)

You can use a few channels of a TM Timer DIDQ for simple counting tasks with an incremental encoder. Counting refers to the recording and adding up of events. The channels configured as counters each acquire the two incremental signals and evaluate these accordingly.

Count direction

The technology module can count up and down with an incremental encoder. You can invert the counting direction to adapt it to the process.

Counting limits

The counting limits define the counter value range used.

The minimum counter value is -2147483648 (-2^{31}). The maximum counter value is 2147483647 (2^{31}-1). The respective counter counts continuously. At an overflow, the counter jumps to the other counting limit in each case and continues counting.

The counter value cannot be influenced by the user program.

Parameter assignment

For use of a counter for an incremental encoder, two digital inputs each of a channel group are combined. For this purpose, you choose the configuration "Incremental encoder (A, B phase-shifted)" in the channel parameters for the respective group.

Note

Counters of the TM Timer DIDQ 16x24V

The number of available counters of the TM Timer DIDQ 16x24V is dependent on the channel configuration. In order to use 4 counters, you must choose the use of 8 inputs in the channel configuration. If you choose the use of 3 inputs, you can use 1 counter. Other channel configurations do not allow any counter use.

Counter value feedback

The current counter value is indicated in the feedback interface in the TEC_IN value (DIm). DIm corresponds to the first of the two grouped digital inputs in each case. For the second digital input, "0" is returned in the value TEC_IN (DIm+1).

Counting with pulse encoder (S7-1500)

You can use a few channels of a TM Timer DIDQ for simple counting tasks with a pulse encoder. Counting refers to the recording and adding up of events. The channels configured as counters each acquire one pulse signal and evaluate this accordingly.

Count direction

The technology module can count up and down with a pulse encoder.

Counting limits

The counting limits define the counter value range used.

The minimum counter value is -2147483648 (-2^{31}). The maximum counter value is 2147483647 ($2^{31}-1$). The respective counter counts continuously. At an overflow, the counter jumps to the other counting limit in each case and continues counting.

The counter value cannot be influenced by the user program.

Parameter assignment

For use of a counter for a pulse encoder, you choose the configuration "Use inputs individually" or "Use input/output individually" in the channel parameters for the respective group. You can configure the first digital input of a group as a counter.

Note

Counters of the TM Timer DIDQ 16x24V

The number of available counters of the TM Timer DIDQ 16x24V is dependent on the channel configuration. In order to use 4 counters, you must choose the use of 8 inputs in the channel configuration. If you choose the use of 3 inputs, you can use 1 counter. Other channel configurations do not allow any counter use.

Counter value feedback

The current counter value is indicated in the feedback interface in the TEC_IN value (DIm). DIm corresponds to the respective digital input.

24 V count signals (S7-1500)

Count signals of 24 V incremental encoders

The 24 V incremental encoder returns the 24 V signals A and B to the technology module. The A and B signals are phase-shifted by 90°.

The figure below shows an example of the time profile of the signals of an 24 V incremental encoder:

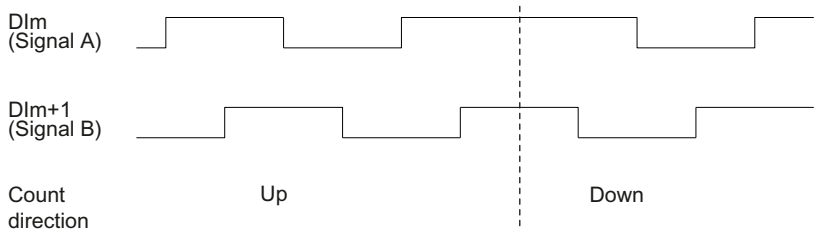

The technology module detects the count direction by evaluating the sequence of edges of the A and B signals. You can specify an inversion of the count direction.

Signal evaluation

The two phase-shifted signals of an incremental encoder are evaluated four times. With quadruple evaluation, the positive and negative edges of signal A and signal B are evaluated.

Whether count pulses are generated in an upward or downward direction depends on the edge direction of the one signal and the level of the other signal in the meantime.

The figure below shows an example of the quadruple evaluation of 24 V count signals:

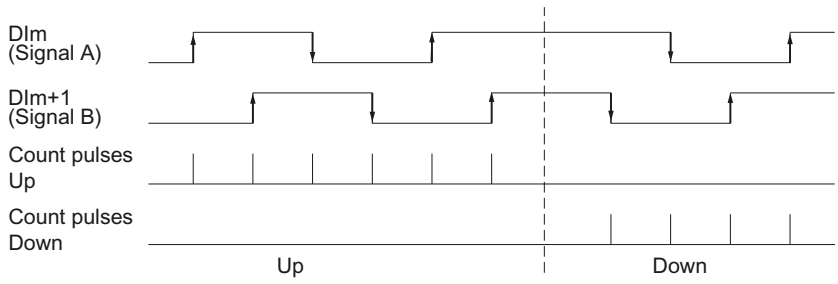

Count signals of 24 V pulse encoders

An encoder, for example, a proximity switch (BERO) or a light barrier, returns only one count signal that is connected to the digital input of a counter.

You can count the positive or the negative edges of the signal.

Isochronous mode (S7-1500)

The TM Timer DIDQ supports the system function "Isochronous mode". This system function enables counter values to be acquired in a defined system cycle.

In isochronous mode, the cycle of the user program, the transmission of the input and output data and the processing in the module are synchronized with each other.

Data processing

The data that was transmitted to the module in the current bus cycle via the control interface takes effect when it is processed in the module's internal cycle. The counter value and status bits are detected at time T_i and made available in the feedback interface for retrieval in the current bus cycle.

Additional information

You will find detailed description of the isochronous mode:

- In the Isochronous mode function manual as a download from the Internet (https://support.industry.siemens.com/cs/ww/en/view/109755401) .
- In the PROFINET with STEP 7 function manual as a download from the Internet (https://support.industry.siemens.com/cs/ww/en/view/49948856).

1.1.1.4 Basics of counting (digital modules) (S7-1500)

Overview of applications (S7-1500)

Introduction

You configure the digital module and assign its parameters using the configuration software.

The module's functions are controlled and monitored via the user program with the control and feedback interface.

System environment

The respective module can be used in the following system environments:

Applications	Components required	Configuration software	In the user program
Central operation with a S7-1500 CPU or ET 200SP CPU	• S7-1500 automation system or ET 200SP CPU • Digital module	STEP 7 (TIA Portal): Device configuration and parameter setting with hardware configuration	Direct access to the control and feedback interface of the digital module in the I/O data
Distributed operation with a S7-1500 CPU	• S7-1500 Automation System • ET 200 Distributed I/O System • Digital module		
Distributed operation with a S7-300/400 CPU	• S7-300/400 automation system • ET 200 distributed I/O system • Digital module	STEP 7 (TIA Portal): Device configuration and parameter setting with hardware configuration STEP 7: Device configuration and parameter setting with hardware configuration (ET 200SP) or GSD file (ET 200MP)	
Distributed operation in a third-party system	• Third-party automation system • ET 200 distributed I/O system • Digital module	Third-party configuration software: Device configuration and parameter settings with GSD file	

Note

A description of the control and feedback interface is available in the device manual for the digital module.

Counting with pulse encoders (S7-1500)

Counting refers to the detection and summation of events. The modules' counters record and evaluate pulse signals. The counting direction can be specified using encoder or pulse signals or through the configuration.

You can use feedback bits to switch the digital outputs of digital output modules at defined counter values.

You can configure the characteristics of the counters using the functionalities described below.

Counter limits

The counter limits define the counter value range used. The counter limits are configurable and can be modified during runtime using the user program. See the module's device manual for the maximum and minimum configurable counter limits.

You can configure whether the counting processes are terminated or continue when a counter limit is violated (automatic gate stop).

Start value

You can configure a start value within the counter limits. The start value can be modified during runtime with the user program.

Gate control

Opening and closing the hardware gate and software gate defines the period of time during which the counting signals are recordd.

The hardware gate is controlled externally via a digital input of the digital module. The hardware gate can be enabled through parameter assignment. The software gate is controlled via the user program. A description of the control and feedback interface is available in the device manual for the digital module.

Behavior at the counting limits (S7-1500)

Violation of a counting limit

The high counting limit is violated when the current counter value is equal to the high counting limit and another upward count pulse is received. The counter low limit is violated when the current counter value is equal to the counter low limit and another downward count pulse is received.

For digital modules of ET 200SP and ET 200AL, the corresponding event bit is set in the feedback interface when the limit is exceeded. You can reset an event bit with the respective control bit:

Counting limit violated	Event bit	Reset bit
High counting limit	EVENT_OFLW	RES_EVENT_OFLW
Low counting limit	EVENT_UFLW	RES_EVENT_UFLW

Note

A description of the control and feedback interface is available in the device manual for the digital module.

You can configure whether or not you want to continue counting to another counter limit after a counting limit violation.

Note

The high counting limit and the start value define the value range of the counter:

Value range of the counter = (high limit – start value) + 1

Examples

The figure below shows an example for terminating counting after overflow and setting the counter to the opposite counting limit:

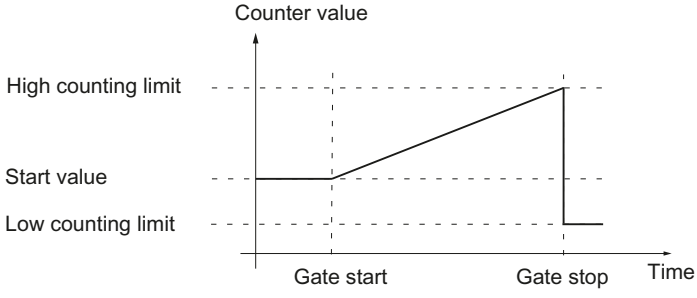

The figure below shows an example for continuing the counting process after an overflow and setting the counter to the opposite counting limit:

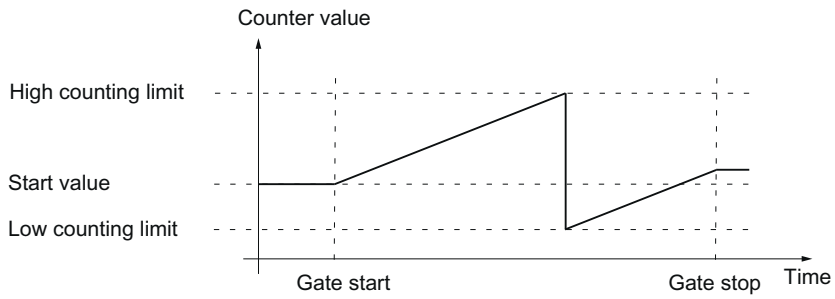

Gate control (S7-1500)

Many applications require counting processes to be started or stopped in accordance with other events. In such cases, counting is started and stopped using the gate function.

The digital modules have two gates for each counting channel. These define the resulting internal gate:

- Software gate
- Hardware gate

Note

The hardware gate is not available for all digital modules.

Software gate (S7-1500)

The software gate of the channel is opened and closed with the SW_GATEcontrol bit.

Note

A description of the control and feedback interface is available in the device manual for the digital module.

Hardware gate (S7-1500)

The hardware gate is optional. You open and close the hardware gate using signals at the corresponding digital input.

Note

A configurable input delay delays the control signal of the digital input.

The status of a DIm digital input is indicated by the respective STS_DIm feedback bit. A description of the control and feedback interface is available in the device manual for the digital module.

Opening and closing the hardware gate

The figure below shows an example of opening and closing with a digital input:

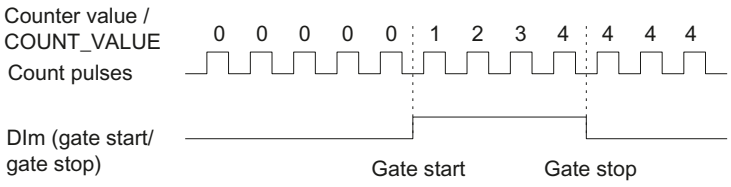

As long as the digital input is set, the hardware gate is open and the count pulses are counted. The hardware gate is closed when the digital input is reset. The counter value stays constant and ignores any further count pulses.

Internal gate (S7-1500)

Internal gate

The internal gate is open if the software gate is open and the hardware gate is open or has not been configured. The status of the internal gate is indicated by the STS_GATE feedback bit. A description of the control and feedback interface is available in the device manual for the digital module.

If the internal gate is open, counting is started. If the internal gate is closed, all other count pulses are ignored and counting is stopped.

If you want to control a counting process with the hardware gate only, the software gate must be open. If you do not configure a hardware gate, the hardware gate is considered to be always open. In this case, you open and close the internal gate with the software gate only.

Hardware gate	Software gate	Internal gate
Open/not configured	open	open
Open/not configured	closed	closed
closed	open	closed
closed	closed	closed

The internal gate can also be automatically closed upon violation of a counting limit. The software or hardware gate must then be closed and reopened to continue counting.

Comparison values (S7-1500)

Depending on the module, you can define up to two comparison values that control a feedback bit for the channel, independent of the user program.

When there are two comparison values, comparison value 1 must be greater than comparison value 0. The comparison values are configurable and can be modified during runtime using the user program.

The comparison values are compared with the current counter value. If the counter value meets the configured comparison condition, the respective STS_DQ feedback bit is set.

You can use the respective feedback bit in order to switch a digital output module's digital output. You can make setting the respective STS_DQ feedback bit dependent on one of the following comparison events. See the device manual for the digital module to find out which comparison events can be configured.

Setting between comparison value and high counter limit

The respective STS_DQ feedback bit is set to 1 when:

Comparison value <= counter value <= high counter limit

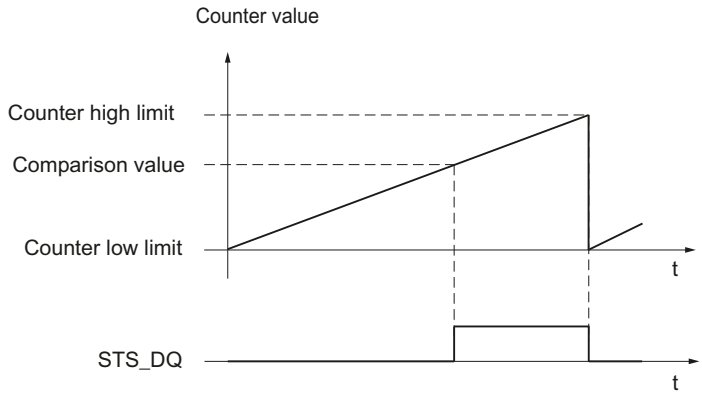

Setting between comparison value and low counter limit

The respective STS_DQ feedback bit is set to 1 when:

Low counter limit <= counter value <= comparison value

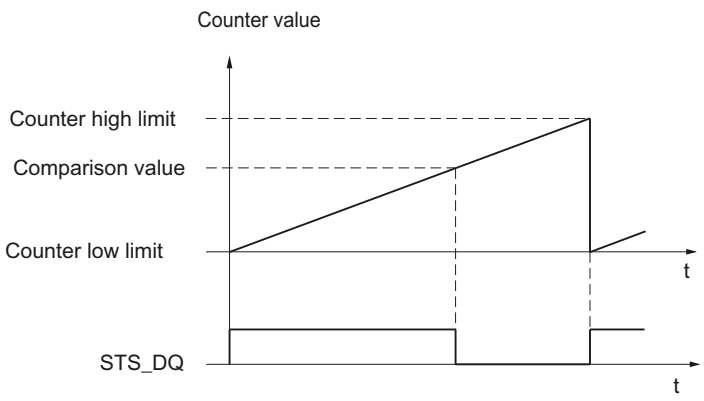

Setting between comparison value 0 and comparison value 1

The respective STS_DQ feedback bit is set to 1 when:

Comparison value 0 <= counter value <= comparison value 1

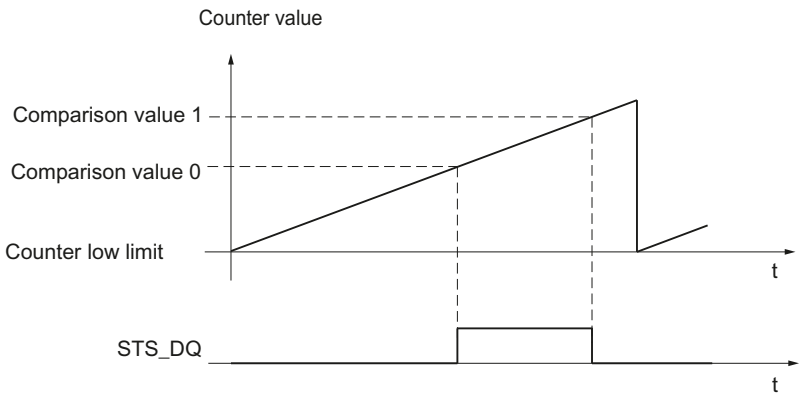

Not setting between comparison value 0 and comparison value 1

The respective STS_DQ feedback bit is set to 1 when:

Comparison value 0 <= counter value <= comparison value 1

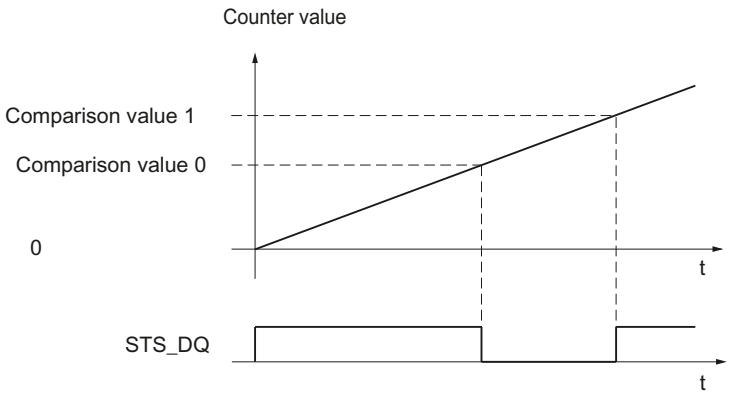

Interrupts (S7-1500)

Hardware interrupt

The modules can trigger a hardware interrupt in the CPU for certain events during operation. Process interrupts are enabled via the parameter assignment. Refer to the device manual for the module for information about the events that can trigger a hardware interrupt during operation.

Note

Hardware interrupts for counting are not available at all the modules.

24 V count signals (S7-1500)

24 V pulse encoder count signals

An encoder, for example an initiator (BERO) or a light barrier, returns a count signal that is connected to the terminal of a counter. For some modules you change the count direction via the parameter assignment of the DI function.

The figure below shows an example of the time profile of the signals of a 24 V pulse encoder with direction signal and the resulting count pulses:

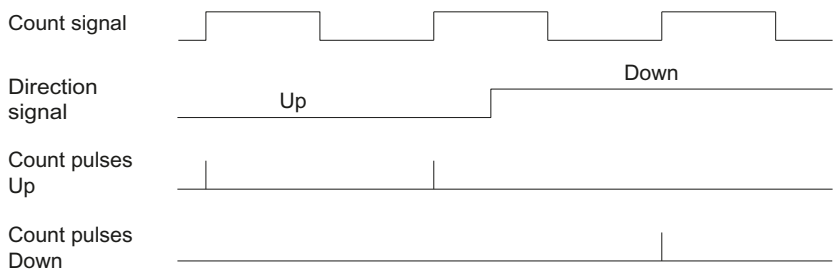

Note

A signal for direction detection is not connectable for all digital modules.

Isochronous mode (S7-1500)

The digital module supports the system function "Isochronous mode". This system function enables counter values to be acquired in a defined system cycle.

In isochronous mode, the cycle of the user program, the transmission of the input and output data and the processing in the module are synchronized with each other.

Data processing

The data that was transmitted to the module in the current bus cycle via the control interface takes effect when it is processed in the module's internal cycle. The counter value and status bits are detected at time T_i and made available in the feedback interface for retrieval in the current bus cycle.

Additional information

You will find detailed description of the isochronous mode:

- In the Isochronous mode function manual as a download from the Internet (https://support.industry.siemens.com/cs/ww/en/view/109755401) .
- In the PROFINET with STEP 7 function manual as a download from the Internet (https://support.industry.siemens.com/cs/ww/en/view/49948856).

1.1.2 Use technology object High_Speed_Counter (S7-1500)

1.1.2.1 Convention (S7-1500)

Technology module: We use the term "technology module" in this documentation both for the technology modules TM Count and TM PosInput and the technology component of the compact CPUs.

1.1.2.2 High_Speed_Counter technology object (S7-1500)

STEP 7 (TIA Portal) supports you in the configuration, commissioning and diagnostics of counting and measuring functions for the following technology modules with the "Technology objects" function:

- You configure the High_Speed_Counter technology object in STEP 7 (TIA Portal) with the settings for the counting and measuring functions.
- The corresponding High_Speed_Counter instruction is programmed in the user program. This instruction supplies the control and feedback interface of the technology module.

The High_Speed_Counter technology object corresponds to the instance DB of the High_Speed_Counter instruction. The configuration of the counting and measuring functions is saved in the technology object. The technology object is located in the folder "PLC > Technology objects".

The High_Speed_Counter technology object can be used for technology modules of both the S7-1500 and ET 200SP systems.

Operating mode

In order to assign the technology module parameters using the technology object, you specify the operating mode (Page 132) "Operating with "Counting and measurement" technology object" in the hardware configuration of the technology module. This selection is already preset.

1.1.2.3 Overview of the configuration steps (S7-1500)

Introduction

The overview below shows the basic procedure for configuring the counting and measuring functions of the technology module with the High_Speed_Counter technology object.

Requirement (TM Count and TM PosInput)

Before you can use the High_Speed_Counter technology object, a project with an S7-1500 CPU or an ET 200SP CPU must be created in STEP 7 (TIA Portal) .

Requirement (Compact CPU)

Before you can use the High_Speed_Counter technology object, a project with a Compact CPU S7-1500 must be created in STEP 7 (TIA Portal).

Procedure

Proceed in the recommended sequence outlined below:

Step	Description
1	Configure a technology module (Page 127)
2	Add technology object (Page 81)
3	Configure a technology object according to your application (Page 82)
4	Call instruction in the user program (Page 102)
5	Load to CPU
6	Commissioning the technology object (Page 103)
7	Diagnostics of the technology object (Page 104)

1.1.2.4 Add technology object (S7-1500)

Adding a technology object in the project navigation

When a technology object is added, an instance DB is created for the instruction of this technology object. The configuration of the technology object is stored in this instance DB.

Requirement (TM Count and TM PosInput)

A project with a CPU S7-1500 has been created.

Requirement (Compact CPU)

A project with a Compact CPU S7-1500 has been created.

Procedure

To add a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click on "Add new object".
   The "Add new object" dialog opens.
4. Select the "Counting and measurement" technology.
5. Select the "High_Speed_Counter" object.
6. Enter an individual name for the technology object in the "Name" text box.

7. Click "Additional information" if you want to add your own information to the technology object.
8. Confirm with "OK".

Result

The new technology object has now been created and stored in the project tree in the "Technology objects" folder.

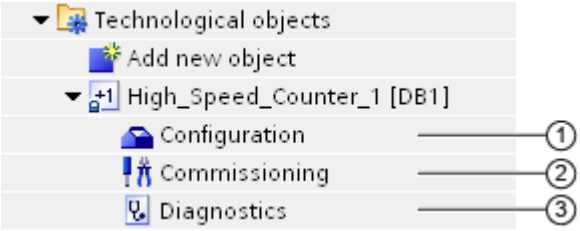

	Object	Description
①	Configuration (Page 82)	In the configuration dialog: • Assignment of technology module and channel • Technology object parameter settings for counting and measurement functions When you change the configuration of the technology object, you must download the technology object **and** the hardware configuration to the CPU.
②	Commissioning (Page 103)	Commissioning and function test of the technology object: Simulating parameters of the High_Speed_Counter instruction and monitoring the effects
③	Diagnostics (Page 104)	Monitoring the counting and measuring functions

1.1.2.5 Configuring the High_Speed_Counter (S7-1500)

Working with the configuration dialog (S7-1500)

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click on the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters include the selection of the technology module and the number of the counting channel for which the technology object is configured.
- **Extended parameters**
  The extended parameters include the parameters for adapting the counting and measuring functions and for setting the characteristics of the digital inputs and digital outputs.

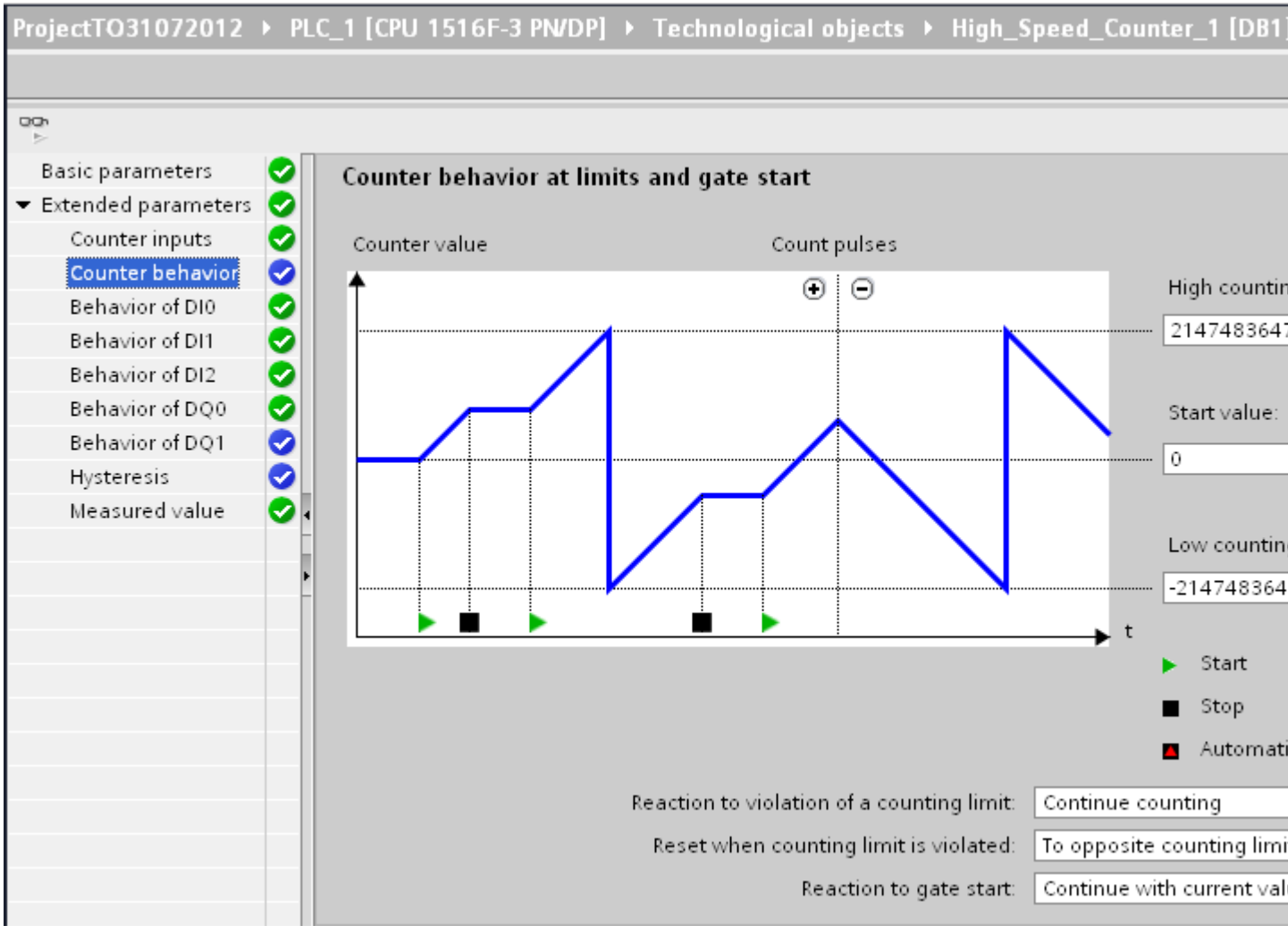

Configuration window icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values, you can use the technology object without additional changes.
	The configuration contains values set by the user or automatically adapted values and is complete All text boxes of the configuration contain valid values and at least one default value was changed.
	The configuration is incomplete or incorrect At least one text box or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to indicate the cause of error.

Basic parameters (S7-1500)

You can establish the connection between the High_Speed_Counter technology object and the technology module under "Basic parameters".

Module (TM Count and TM PosInput)

You select the technology module in a subsequent dialog box. All technology modules (central or distributed) that are configured for use with a "Counting and measurement" technology object under the S7-1500 CPU or ET 200SP CPU are available for selection.

After selecting the technology module, you can open the device configuration associated with the technology module by clicking the"Device configuration" button.

The technology module parameter settings required for the use of the technology object are made in the "Extended parameters" of the technology object.

Module (Compact CPU)

You can select a high-speed counter for the Compact CPU in a subsequent dialog. You can choose any of the high-speed counters which are enabled and configured for use with a technology object from "Counting and measuring".

After selecting the high-speed counter, you can open the device configuration associated with the Compact CPU by clicking the"Device configuration" button.

The parameter settings of the high-speed counter required for the use of the technology object are made in the "Extended parameters" of the technology object.

Channel

For a technology module with several counting channels, you can also select the number of the counting channel for which the High_Speed_Counter technology object is to be valid.

Note

A channel can be assigned to only *one* technology object. A channel that is already assigned to a technology object can no longer be selected.

Synchronization of the parameter values

If, after assignment of the channel to the technology object, there is an inconsistency between the parameter values in the property dialog and in the technology object, a button with a corresponding inquiry appears. When you click the button, the parameter values in the property dialog are overwritten by the parameter values in the property dialog within STEP 7 (TIA Portal). The current parameter values of the technology object are displayed in the property dialog.

Counter inputs (High_Speed_Counter) (S7-1500)

Signal type

You can choose from the following signal types (Page 60):

Symbol	Signal type	Meaning	Additional option-specific parameters
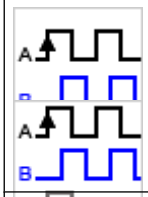	Incremental encoder (A, B phase-shifted)	An incremental encoder with phase-shifted A and B signals is connected.	• Signal evaluation • Invert direction • Filter frequency • Sensor type or Interface standard
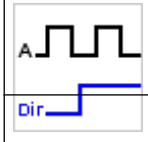	Incremental encoder (A, B, N)	An incremental encoder with phase-shifted signals A and B and a zero signal N is connected.	• Signal evaluation • Invert direction • Filter frequency • Sensor type or Interface standard • Reaction to signal N • Frequency of synchronization • Frequency of the Capture function
	Pulse (A) and direction (B)	A pulse encoder (signal A) with direction signal (signal B) is connected.	• Filter frequency • Sensor type or Interface standard

	Signal type	Meaning	Additional option-specific parameters
	Pulse (A)	A pulse encoder (signal A) without direction signal is connected. You can specify the count direction by means of the control interface (Page 166).	• Filter frequency • Sensor type or Interface standard
	Count up (A), count down (B)	Signals for counting up (signal A) and down (signal B) are connected.	• Filter frequency • Sensor type or Interface standard

Invert direction

You can invert the counting direction to adapt it to the process.

The inverting of the direction is configurable and active for the following signal types:

- Incremental encoder (A, B phase-shifted)
- Incremental encoder (A, B, N)

Signal evaluation

By configuring signal evaluation (Page 64), you can specify which edges of the signals are counted.

You can select from the following options:

Symbol	Signal evaluation	Meaning
	Single (Page 65) (default)	The edges of signal A are evaluated during a low level of signal B.
	Double (Page 65)	Each edge of signal A is evaluated.
	Quadruple (Page 66)	Each edge of signals A and B is evaluated.

The parameter can be assigned with the following signal types:

- Incremental encoder (A, B phase-shifted)
- Incremental encoder (A, B, N)

Filter frequency

By configuring the filter frequency, you suppress interferences at the counting inputs A, B and N.

The selected filter frequency is based on a pulse/break ratio of between around 40:60 and around 60:40. This results in a specific minimum pulse/break time. Signal changes with a duration shorter than the minimum pulse/break time are suppressed.

You can select from the following filter frequencies:

Filter frequency	Minimum pulse/break time
100 Hz	4.0ms
200 Hz	2.0 ms
500 Hz	800 µs
1 kHz	400 µs
2 kHz	200 µs
5 kHz	80 µs
10 kHz	40 µs
20 kHz	20 µs
50 kHz	8.0 µs
100 kHz (preset for Compact CPU)	4.0 µs
200 kHz** (preset for TM Count)	2.0 µs
500 kHz*	0.8 µs
1 MHz* (default at TM PosInput)	0.4 µs

* Only available with TM PosInput

** * Only available with TM Count and TM PosInput

Sensor type (TM Count)

By configuring the sensor type, you specify how the counter inputs are switched for the TM Count.

You can select from the following options:

Sensor type	Meaning
Sourcing output (default)	The encoder/sensor switches the inputs A, B and N to 24VDC.
Sinking output	The encoder/sensor switches the inputs A, B and N to M.
Push-pull (sinking and sourcing output)	The encoder/sensor alternately switches the inputs A, B and N to M and 24VDC.

"Push-pull" is usually selected when incremental encoders are used. If using 2-wire sensors, such as light barriers or proximity switches, you need to select the corresponding wiring "sourcing output" or "sinking output".

The data sheet of the encoder includes information on whether your incremental encoder is a push-pull encoder.

Note

If you use a push-pull encoder and the sensor type "Push-pull (sinking and sourcing output)" is configured, you can monitor the encoder signals for wire break.

Sensor type (Compact CPU)

The "Sourcing output" sensor type is set for the Compact CPU and cannot be changed. The encoder or sensor switches the inputs A, B and N to 24V DC.

You can operate both sourcing output encoders as well as push-pull encoders on the Compact CPU. You can find additional information on the sensor type in the data sheet for the encoder.

Interface Standard (TM PosInput)

You use this parameter to specify whether the encoder outputs symmetric (RS422) or asymmetric (TTL) signals for the TM PosInput.

You can select from the following options:

Interface standard	Meaning
RS422, symmetrical (default)	The encoder outputs symmetric signals according to the RS422 standard (Page 62).
TTL (5 V), asymmetrical	The encoder outputs asymmetric 5 V signals according to the TTL standard (Page 60).

Note

The RS422 standard provides greater interference immunity than the TTL standard. If your incremental or pulse encoder supports the RS422 standard **and** the TTL standard, we recommend using the RS422 standard.

Reaction to signal N

You use this parameter to specify which reaction is triggered to signal N.

You can select from the following options:

Option	Meaning
No reaction to signal N (default)	The counter is not affected by signal N.
Synchronization at signal N (Page 36)	The counter is set to the start value at signal N. If you select the function "Enable synchronization at signal N" for a digital input, the synchronization depends on the level at the digital input.
Capture at signal N (Page 28)	The counter value is stored in the Capture value at signal N. The use of a digital input and the use of the N signal are not mutually exclusive for the Capture function. The technology object displays the Capture value at the output parameter CapturedValue.

Note

You can only select the reaction to signal N if you have selected the "Incremental encoder (A, B, N)" signal type.

Note

If "Synchronization at signal N" is selected, you can select the function "Enable synchronization at signal N" for a digital input (Page 91).

Note

For High_Speed_Counter as of V3.0, the following applies:

You can choose "Capture at signal N" only in operating mode "Use count value as reference":

Frequency of synchronization

This parameter is used to define the frequency of the following events:

- Synchronization at signal N
- Synchronization as function of a digital input

You can select from the following options:

Option	Meaning
Once (default)	The counter is only set at the first signal N or the first configured edge of the digital input.
Periodic	The counter is set at each signal N or each configured edge of the digital input.

Frequency of the Capture function

This parameter is used to define the frequency of Capture events for the following functions:

- Capture at Signal N
- Capture as function of a digital input

You can select from the following options:

Option	Meaning
Once	The first configured edge at the respective digital input or first rising edge of the N signal saves the current counter value as a Capture value.
Periodic (default)	Each configured edge at the respective digital input or each rising edge of the N signal saves the current counter value as a Capture value.

Note

This parameter is available for High_Speed_Counter as of V3.2.

Counter behavior (S7-1500)

Counting limits and start value (S7-1500)

High counting limit

You limit the counting range by setting the high counting limit. You can enter a value up to 2147483647 (2^{31}-1). You must enter a value that lies above the low counting limit.

The default setting is "2147483647".

Low counting limit

You limit the counting range by setting the low counting limit. You can enter a value up to -2147483648 (-2^{31}). You must enter a value below the high counting limit.

The default setting is "-2147483648".

Start value

By configuring the start value, you specify the value at which counting is to start and where it is to continue in the case of defined events. You must enter a value between the counting limits or equal to the counting limits.

The default setting is "0".

Additional information

For more information, see Behavior at the counting limits (Page 23) and Counter behavior at gate start (Page 27).

Counter behavior at limits and gate start (S7-1500)

Reaction to violation of a counting limit

You can configure the following characteristics for violation of a counting limit (Page 23):

Reaction	Meaning
Stop counting	If a counting limit is violated, counting is stopped and the internal gate is closed. To restart counting, you must also close and reopen the software/hardware gate.
Continue counting (default)	Counting continues either with the start value or at the opposite counting limit, depending on the additional parameter assignment.

Reset when counting limit is violated

You can reset the counter to the following values when a counting limit is violated:

Reset the value	Meaning
To start value	The counter value is set to the start value.
To opposite counting limit (default)	The counter value is set to the opposite counting limit.

Reaction to gate start

You can set the following Reaction to gate start (Page 27):

Reaction	Meaning
Set to start value	When the gate is opened, the counter value is set to the start value.
Continue with current value (default)	When the gate is opened, counting is continued with the last counter value.

Behavior of a DI (High_Speed_Counter) (S7-1500)

Setting function of the DI

By configuring a digital input, you specify which functions the digital input triggers at switching.

You can select from the following options:

Function of a digital input	Meaning	Additional option-specific parameters
Gate start/stop (level-triggered)	The level at the respective digital input opens and closes the hardware gate (Page 25).	• Input delay • Select level
Gate start (edge-triggered)	The configured edge at the respective digital input opens the hardware gate (Page 25).	• Input delay • Edge selection
Gate stop (edge-triggered)	The configured edge at the respective digital input closes the hardware gate (Page 25).	• Input delay • Edge selection
Synchronization (Page 32)	The configured edge at the respective digital input sets the counter to the start value. The technology object indicates whether a synchronization has occurred at the output parameter SyncStatus.	• Input delay • Edge selection • Frequency of synchronization
Enable synchronization at signal N	The active level at the respective digital input enables synchronization of the counter at signal N (Page 36).	• Input delay • Select level

Function of a digital input	Meaning	Additional option-specific parameters
Capture	The configured edge at the respective digital input saves the current counter value (Page 28) as a Capture value. The use of a digital input and the use of the N signal are not mutually exclusive for the Capture function. The technology object displays the Capture value at the output parameter CapturedValue.	• Input delay • Edge selection • Frequency of the Capture function • Behavior of counter value after Capture
Digital input without function	No technological function is assigned to the respective digital input. You can read the signal state of the digital input via the respective static variable of the technology object: • UserStatusFlags.StatusDI0 • UserStatusFlags.StatusDI1 • UserStatusFlags.StatusDI2	• Input delay

Note

With the exception of "Digital input without function", each function can only be used once for each counter, and if used for one digital input is not available to the others.

Note

For High_Speed_Counter as of V3.0, the following applies:

You can choose the "Capture" function only in operating mode "Use count value as reference".

Input delay (TM Count and TM PosInput)

You use this parameter to suppress signal interference at the digital inputs. Changes to the signal are only detected if they remain stable for longer than the configured input delay time.

You can select from the following input delays:

- None
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms
- 1.6 ms
- 3.2 ms
- 12.8 ms
- 20 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs.

Note

You configure the input delay under "Behavior of DI0" for all digital inputs together. The input delay is also displayed under "Behavior of DI1" and for TM Count also under "Behavior of DI2".

Input delay (Compact CPU)

You use this parameter to suppress interference at the digital inputs of the DIn signals. Changes to the signal are only detected if they remain stable for longer than the configured input delay time.

You can configure the input delay for a digital input of a Compact CPU in the Inspector window of the device configuration under "Properties > DI 16/DQ 16 > Inputs > Channel n".

You can select from the following input delays:

- None
- 0.05 ms
- 0.1 ms
- 0.4 ms
- 1.6 ms
- 3.2 ms (default)
- 12.8 ms
- 20 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs.

Select level

You use this parameter to specify the level at which the digital input is active.

You can select from the following options:

Level	Meaning
Active with high level (default)	The respective digital input is active when it is set.
Active with low level	The respective digital input is active when it is not set.

The parameter can be set for the following functions of a digital input:

- Gate start/stop (level-triggered)
- Enable synchronization at signal N

Edge selection

You can use this parameter to specify the edge of the digital input at which the configured function is triggered.

The following options may be available depending on the function selected:

- At rising edge (default)
- At falling edge
- At rising and falling edge

The parameter can be set for the following functions of a digital input:

- Gate start (edge-triggered)
- Gate stop (edge-triggered)
- Synchronization
- Capture

Note

At rising and falling edge" can only be configured for the "Capture" function.

Frequency of synchronization

This parameter is used to define the frequency of the following events:

- Synchronization at signal N
- Synchronization as a function of a digital input

You can select from the following options:

Option	Meaning
Once (default)	The counter is only set at the first signal N or the first configured edge of the digital input.
Periodic	The counter is set at each signal N or each configured edge of the digital input.

Frequency of the Capture function

This parameter is used to define the frequency of Capture events for the following functions:

- Capture at Signal N
- Capture as function of a digital input

You can select from the following options:

Option	Meaning
Once	The first configured edge at the respective digital input or first rising edge of the N signal saves the current counter value as a Capture value.
Periodic (default)	Each configured edge at the respective digital input or each rising edge of the N signal saves the current counter value as a Capture value.

Note

This parameter is available for High_Speed_Counter as of V3.2.

Behavior of counter value after Capture

You can configure the following characteristics for the counter after a capture event (Page 28):

Reaction	Meaning
Continue counting (default)	After saving the current counter value as Capture value, counting is continued unchanged.
Set to start value and continue counting	After saving the current counter value as Capture value, counting is continued with the start value.

Behavior of a DQ (High_Speed_Counter) (S7-1500)

Operating mode (High_Speed_Counter V3.0 or higher)

The operating mode determines which value comparison functions work.

Operating mode	Meaning
Use count value as reference (default)	The comparison functions and hardware interrupts for compare events work with the counter value. This functionality corresponds to the functionality of the High_Speed_Counter in versions before V3.0.
Use measured value as reference	The comparison functions and hardware interrupts for compare events work with the measured value.

Note

You configure the operating mode under "Behavior of DQ0" for both digital outputs together. The operating mode is also displayed under "Behavior of DQ1".

Set output

With the parameter assignment of a digital output, you specify the condition upon which the digital output switches.

You can select from the following options:

Function of a digital output (Page 40) in operating mode "Use count value as reference"	Meaning	Additional option-specific parameters
Between comparison value and high limit (default)	The respective digital output is active if: Comparison value <= counter value <= high counting limit	• Comparison value 0 • Comparison value 1 • Hysteresis (in increments)
Between comparison value and low limit	The respective digital output is active if: Low counting limit <= counter value <= comparison value	• Comparison value 0 • Comparison value 1 • Hysteresis (in increments)
Between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 0 <= counter value <= comparison value 1	• Comparison value 0 • Comparison value 1 • Hysteresis (in increments)
At comparison value for a pulse duration	The respective digital output is active once for the configured time and count direction when the counter value reaches the comparison value.	• Comparison value 0 • Comparison value 1 • Count direction • Pulse duration • Hysteresis (in increments)
After set command from CPU until comparison value	When a set command is sent from the CPU, the respective digital output is active until the counter value is equal to the comparison value.	• Comparison value 0 • Comparison value 1 • Count direction • Hysteresis (in increments)
Use by user program	The respective digital output can be switched by the CPU via the control interface (Page 39).	—

Note

DQ0 of a counter of a Compact CPU

With a Compact CPU, the respective digital output DQ0 is available via the feedback interface, but not as a physical output.

Note

You can only select the "Between comparison value 0 and 1" function for digital output DQ1 and only if you have selected the "Use by user program" function for digital output DQ0.

Note

The "At comparison value for a pulse duration" and "After set command from CPU until comparison value" functions only switch the digital output in question if a count pulse reaches the comparison value. When the counter value is set, e.g. by synchronization, the digital output does not switch.

Function of a digital output (Page 47) in operating mode "Use measured value as reference"	Meaning	Additional option-specific parameters
Measured value >= comparison value (default)	The respective digital output is active if the measured value is greater than or equal to the comparison value.	• Comparison value 0 • Comparison value 1
Measured value <= comparison value	The respective digital output is active if the measured value is less than or equal to the comparison value.	• Comparison value 0 • Comparison value 1
Between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 0 <= measured value <= comparison value 1	• Comparison value 0 • Comparison value 1
Not between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 1 <= measured value <= comparison value 0	• Comparison value 0 • Comparison value 1
Use by user program	The respective digital output can be switched by the CPU via the control interface (Page 39).	—

Note

You can select the "Between comparison value 0 and 1" and "Not between comparison value 0 and 1" functions only for digital output DQ1 and only if you have selected the "Use by user program" function for digital output DQ0.

Comparison value 0(TM Count and TM PosInput)

Operating mode "Use count value as reference"

With the parameter assignment of the comparison value (Page 40), you specify the counter value at which the digital output DQ0 switches as a result of the selected comparison event.

You must enter an integer (DINT) that is greater than or equal to the low counting limit. If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The default setting is "0".

Operating mode "Use measured value as reference"

With the parameter assignment of the comparison value (Page 47), you specify the measured value at which the digital output DQ0 switches as a result of the selected comparison event.

You must enter a floating point number (REAL). If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The minimum value is -7.922816×10^{28}. The default setting is "0.0". The unit of the comparison value depends on the measured variable.

Comparison value 0(Compact CPU)

Operating mode "Use count value as reference"

With the parameter assignment of the Comparison value (Page 40), you specify the counter value at which the STS_DQ0 bit is set in the feedback interface of the selected comparison event. The digital output DQ0 is not available as a physical output in a Compact CPU.

You must enter an integer (DINT) that is greater than or equal to the low counting limit. If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The default setting is "0".

Operating mode "Use measured value as reference"

With the parameter assignment of the Comparison value (Page 47), you specify the measured value at which the STS_DQ0 bit is set in the feedback interface of the selected comparison event. The digital output DQ0 is not available as a physical output in a Compact CPU.

You must enter a floating point number (REAL). If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The minimum value is -7.922816×10^{28}. The default setting is "0.0". The unit of the comparison value depends on the measured variable.

Comparison value 1

Operating mode "Use count value as reference"

With the parameter assignment of the comparison value (Page 40), you specify the counter value at which the digital output DQ1 switches as a result of the selected comparison event.

You must enter an integer (DINT) that is smaller than or equal to the high counting limit. If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The default setting is "10".

Operating mode "Use measured value as reference"

With the parameter assignment of the comparison value (Page 47), you specify the measured value at which the digital output DQ1 switches as a result of the selected comparison event.

You must enter a floating point number (REAL). If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The maximum value is 7.922816×10^{28}. The default setting is "10.0". The unit of the comparison value depends on the measured variable.

Count direction

You use this parameter to specify the count direction for which the selected function is valid.

You can select from the following options:

Count direction	Meaning
In both directions (default)	The comparison and switching of the respective digital output take place regardless of the count direction.
Up	The comparison and switching of the respective digital output only takes place when the counter counts up.
Down	The comparison and switching of the respective digital output only takes place when the counter counts down.

The parameter can be configured for the following functions:

- At comparison value for a pulse duration
- After set command from CPU until comparison value

Pulse duration

By configuring the pulse duration for the function "At comparison value for a pulse duration", you specify the number of milliseconds for which the respective digital output is active.

If you enter "0" and the counter value corresponds to the comparison value, the digital output remains active until the next count pulse.

You can enter a value between 0.0 and 6553.5.

The default setting is "500.0", which is equivalent to a pulse duration of 0.5 s.

Hysteresis (in increments)

By configuring the hysteresis (Page 56), you can define a range around the comparison values. For the functions "Between comparison value and upper counter limit" and "Between comparison value and lower counter limit" the hysteresis also applies at the counter limits. Within the hysteresis range, the digital outputs cannot switch again until the counter value is outside the range.

Choose a small enough hysteresis. When the hysteresis range, starting from the configured comparison value, spans the entire counter value range, proper functioning of the comparison values cannot be guaranteed.

If a comparison value lies so close to the counter limit that the hysteresis range would extend beyond this counter limit, the hysteresis range ends at it.

If you enter "0", the hysteresis is turned off. You can enter a value between 0and 255. The default setting is "0".

Note

The following applies for High_Speed_Counter as of V3.0:

You configure the hysteresis under "Behavior of DQ0" for both digital outputs together. The hysteresis is also displayed under "Behavior of DQ1".

The hysteresis is only available in operating mode "Use count value as reference".

Specify measured value (High_Speed_Counter) (S7-1500)

Measured variable

This parameter is used to specify the measured variable (Page 50) to be provided by the technology module. The technology object displays the measured value at the output parameter MeasuredValue.

You can select from the following options:

Measured variable	Meaning	Additional option-specific parameters
Frequency (default)	The measured variable shows the number of increments per second. The value is a floating point number (REAL). The unit is Hz. The technology object displays the measured value at the output parameter MeasuredValue.	• Update time
Period	The measured variable is the average period between two increments. The value is an integer (DINT). The unit is s. The technology object displays the measured value at the output parameter MeasuredValue.	• Update time
Velocity	The measured variable is a velocity. Examples of a velocity measurement can be found in the explanation of the "Increments per unit" parameter. The technology object displays the measured value at the output parameter MeasuredValue.	• Update time • Time base for velocity measurement • Increments per unit

Update time

By configuring the update time (Page 50) in milliseconds, you can specify the time interval between two measured value updates.

The update time and the signal type effect the accuracy of the measurement. In the case of update times of at least 100 ms, the effect of the signal type is negligible.

In the case of update times of less than 100 ms, you achieve maximum measurement accuracy using the following signal types:

- Incremental encoder (A, B phase-shifted) with Signal evaluation "Single"
- Incremental encoder (A, B, N) with Signal evaluation "Single"
- Pulse (A) and direction (B)
- Pulse (A)

In the case of other signal types, measurement accuracy depends on the encoder and cable used.

If you enter "0", the measured value is updated once per module-internal cycle. Up to three decimal places can be entered. A value from 0.0 to 25000.0 is permissible. The default setting is "10.0".

Time base for velocity measurement

This parameter defines the time base on which the velocity is to be returned.

You can select from the following options:

- 1 ms
- 10 ms
- 100 ms

- 1 s
- 60 s

The default setting is "60 s".

Increments per unit

You can use this parameter to define the number of count pulses per relevant unit that the incremental or absolute encoder supplies for the velocity measurement.

The number of count pulses depends on the configured signal evaluation. You can enter a value between 1 and 65535.

Example 1:

You encoder delivers 4000 count pulses for travel over one meter. The velocity is to be measured in meters per second. "Double" is configured as signal evaluation.

In this case, you need to assign the following parameters:

- Increments per unit: 8000
- Time base for velocity measurement: 1 s

Example 2:

Your encoder delivers 4096 count pulses per revolution. The velocity is to be measured in revolutions per minute. "Single" is configured as the signal evaluation.

In this case, you need to assign the following parameters:

- Increments per unit: 4096
- Time base for velocity measurement: 60 s

1.1.2.6 Programming the High_Speed_Counter (S7-1500)

High_Speed_Counter instruction (S7-1500)

High_Speed_Counter

The High_Speed_Counter instruction is part of the High_Speed_Counter technology object. It supplies the control and feedback interface of the technology module.

The High_Speed_Counter instruction thereby forms the software interface between the user program and the technology module. It must be called cyclically from the user program in order to synchronize the input and output data.

The High_Speed_Counter instruction can be used for the technology modules of both S7-1500 and ET 200SP. The modules can be used centrally and decentrally. The instruction applies in each case to the channel of the technology module that was assigned to the associated technology object.

Additional information

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

Call instruction in the user program (S7-1500)

The High_Speed_Counter instruction can be called once for each counter in the cycle or, alternatively, in a time-controlled program. The call is not permitted in an event-controlled interrupt program.

Procedure

Proceed as follows to call the instruction in the user program:

1. Open the CPU folder in the project tree.
2. Open the "Program blocks" folder.
3. Double-click the OB for cyclic program execution.
   The block is opened in the work area.
4. In the "Instructions" window, open the "Technology" group and the "Counting and measurement" folder.
   The folder contains the instruction.
5. Select the instruction and drag it to your OB.
   The "Call options" dialog opens.
6. Select a technology object from the "Name" list or enter the name for a new technology object.
7. Confirm with "OK".

Result

If the technology object does not exist yet, it is added. The instruction is added in the OB. The technology object is assigned to this call of the instruction.

Note

If you click one of the buttons "Configuration", "Commissioning" or "Diagnostics" in the user interface of the instruction, the respective editor opens.

1.1.2.7 Commissioning the High_Speed_Counter (S7-1500)

Commissioning the technology object (S7-1500)

A graphic display of the block in the commissioning editor helps you with commissioning and the function test for the technology object. You can change specific parameters of the High_Speed_Counter instruction in CPU/IM online mode and monitor their effects.

Requirements

- There is an online connection between STEP 7 (TIA Portal) and the CPU.
- The CPU is in RUN.
- The corresponding High_Speed_Counter instruction is called cyclically from the user program.
- The parameters of the technology object are not overwritten by the user program.

Procedure

To open the commissioning editor of a technology object and to simulate a parameter value change, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the High_Speed_Counter technology object in the project tree.
3. Double-click on the "Commissioning" object.
   The functions for commissioning the High_Speed_Counter technology object are displayed.
4. In the commissioning dialog, click on the "Monitor all" button.
   The parameters (online values) of the High_Speed_Counter technology object are loaded and displayed.
5. If the parameter you want to change has a text box, enter the new value there.
6. Select the check box of the parameter.
   The new parameter value becomes effective and the effects of the change are simulated.

Online mode

In online mode, you can modify the following parameters to test the technology object function:

- New counter value (NewCountValue)
- New high counting limit (NewUpperLimit)
- New low counting limit (NewLowerLimit)
- New comparison value 0 (NewReferenceValue0 or NewReferenceValue0_M)
- New comparison value 1 (NewReferenceValue1 or NewReferenceValue1_M)
- New start value (NewStartValue)
- Start and stop counter (SwGate)
- Enable Capture (CaptureEnable)

- Enable synchronization (SyncEnable)
- Acknowledgment of signaled error states (ErrorAck)
- Resetting the status flag (EventAck)

1.1.2.8 High_Speed_Counter diagnostics (S7-1500)

Monitoring counter values, measured values, DIs and DQs (S7-1500)

You use the diagnostic functions to monitor the counting and measuring functions.

Requirements

- There is an online connection between STEP 7 (TIA Portal) and the CPU.
- The CPU is in RUN.

Procedure

To open the display editor for the diagnostic functions, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the High_Speed_Counter technology object in the project tree.
3. Double-click on the "Diagnostics" object.
4. Click the "Monitor all" button.

Display

The following values are read by the technology object from the feedback interface and displayed:

- Event display/diagnostics information
- Signal states of the digital inputs and digital outputs
- Counter value
- Capture value
- Measured value

Additional information on status displays is available in the context-sensitive help for each event in STEP 7 (TIA Portal). When the CPU is in STOP, the status display is not updated.

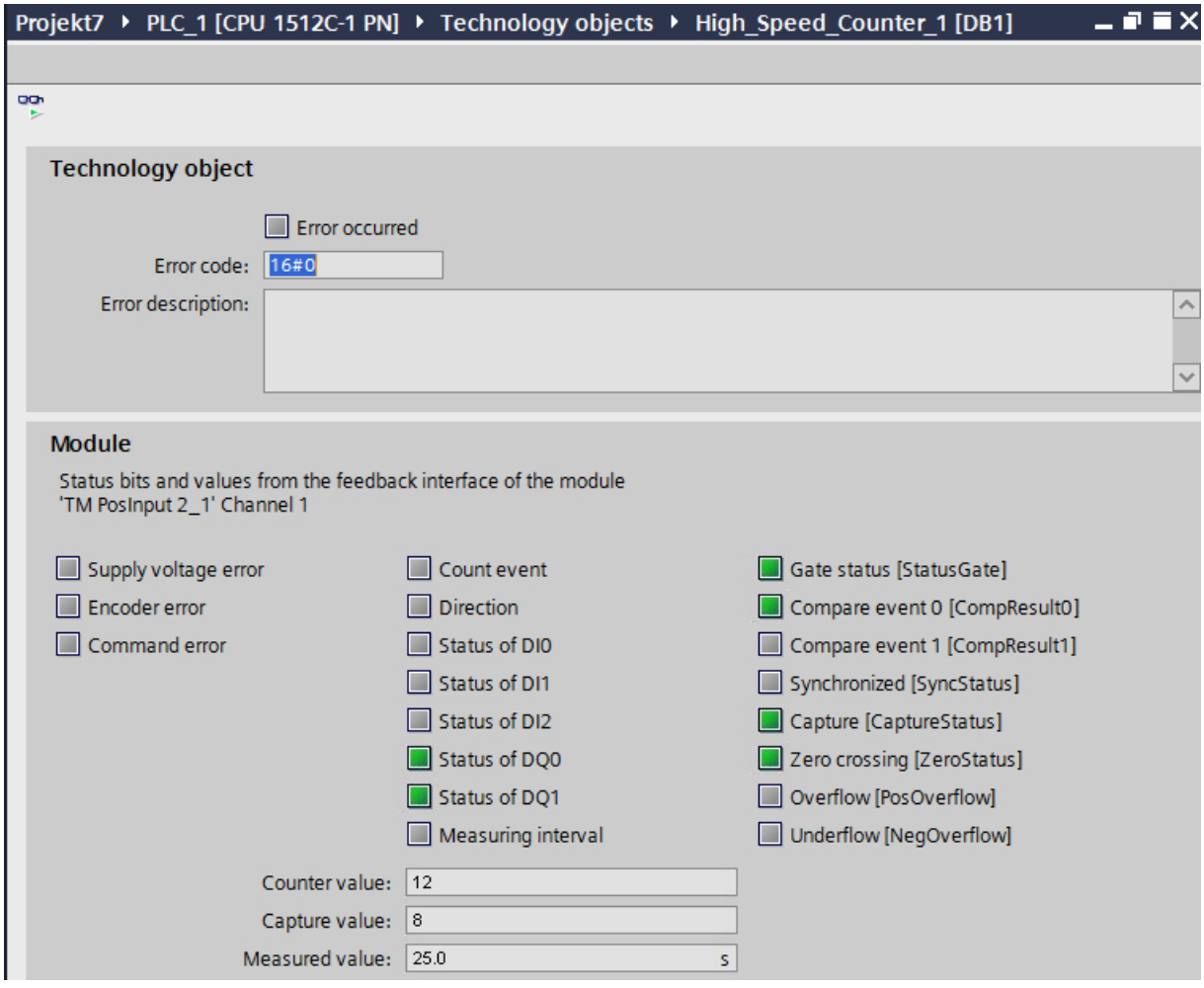

1.1.3 Using technology object SSI_Absolute_Encoder (S7-1500)

1.1.3.1 Technology object SSI_Absolute_Encoder (S7-1500)

STEP 7 (TIA Portal) supports you in the configuration, commissioning and diagnostics of counting and measuring functions for the TM PosInput technology module with the "Technology objects" function in combination with the SSI absolute encoders:

- In STEP 7 (TIA Portal) you configure the SSI_Absolute_Encoder technology object by entering the encoder parameters.
- The corresponding SSI_Absolute_Encoder instruction is programmed in the user program. This instruction supplies the control and feedback interface of the technology module.

The SSI_Absolute_Encoder technology object corresponds to the instance DB of the SSI_Absolute_Encoder instruction. The configuration of the position input and measuring functions is saved in the technology object. The technology object is located in the folder "PLC > Technology objects".

The SSI_Absolute_Encoder technology object can be used for the TM PosInput of both the S7-1500 and ET 200SP systems alike.

Operating mode

In order to assign the technology module parameters using a TM PosInput, you specify the operating mode (Page 132) "Operation with technology objects for "counting and measurement"" in the hardware configuration of the TM PosInput. This selection is already preset.

1.1.3.2 Overview of the configuration steps (S7-1500)

Introduction

The overview below shows the basic procedure for configuring the position input and measuring functions of the technology module with the SSI_Absolute_Encoder technology object.

Requirement

Before you can use the technology object, a project with an S7-1500 CPU or an ET 200SP CPU must be created in STEP 7 (TIA Portal).

Procedure

Proceed in the recommended sequence outlined below:

Step	Description
1	Configure a technology module (Page 127)
2	Add technology object (Page 106)
3	Configure a technology object according to your application (Page 108)
4	Call instruction in the user program (Page 123)
5	Load to CPU
6	Commissioning the technology object (Page 123)
7	Diagnostics of the technology object (Page 124)

1.1.3.3 Add technology object (S7-1500)

Adding a technology object in the project navigation

When a technology object is added, an instance DB is created for the instruction of this technology object. The configuration of the technology object is stored in this instance DB.

Requirement

A project with a CPU S7-1500 has been created.

Procedure

To add a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click on "Add new object".
   The "Add new object" dialog opens.
4. Select the "Counting and measurement" technology.
5. Select the "SSI_Absolute_Encoder" object.
6. Enter an individual name for the technology object in the "Name" text box.
7. Click "Additional information" if you want to add your own information to the technology object.
8. Confirm with "OK".

Result

The new technology object has now been created and stored in the project tree in the "Technology objects" folder.

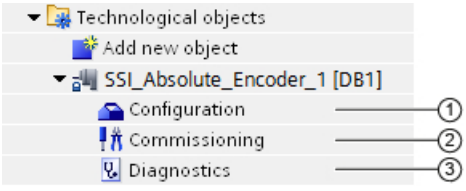

	Object	Description
①	Configuration (Page 108)	In the configuration dialog: • Assignment of technology module and channel • Technology object parameter settings for position input and measurement functions When you change the configuration of the technology object, you must download the technology object **and** the hardware configuration to the CPU.
②	Commissioning (Page 123)	Commissioning and function test of the technology object: Simulating parameters of the SSI_Absolute_Encoder instruction and monitoring the effects
③	Diagnostics (Page 124)	Monitoring of the position input and measurement functions

1.1.3.4 Configuring SSI_Absolute_Encoder (S7-1500)

Working with the configuration dialog (S7-1500)

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click on the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters include the selection of the technology module and the number of the channel for which the technology object is configured.
- **Extended parameters**
  The extended parameters include the parameters for adapting the position input and measuring functions and for setting the characteristics of the digital inputs and digital outputs.

Configuration window icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values, you can use the technology object without additional changes.
	The configuration contains values set by the user or automatically adapted values and is complete All text boxes of the configuration contain valid values and at least one default value was changed.
	The configuration is incomplete or incorrect At least one text box or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to indicate the cause of error.

Basic parameters (S7-1500)

Under "Basic parameters", you can establish the connection between the technology object and the TM PosInput technology module.

Module

You select the technology module in a subsequent dialog box. All TM PosInput technology modules (central or distributed) that are configured for use with a "Counting and measurement" technology object under the S7-1500 CPU or ET 200SP CPU are available for selection.

After selecting the technology module, you can open the device configuration associated with the technology module by clicking the"Device configuration" button.

The technology module parameter settings required for the use of the technology object are made in the "Extended parameters" of the technology object.

Channel

For a technology module with several channels, you select the number of the channel for which the technology object is valid.

Note

A channel can be assigned to only *one* technology object. A channel that is already assigned to a technology object can no longer be selected.

Synchronization of the parameter values

If, after assignment of the channel to the technology object, there is an inconsistency between the parameter values in the property dialog of the TM PosInput module and in the technology object, a button with a corresponding inquiry appears. When you click the button, the parameter values in the property dialog of the assigned module are overwritten by the parameter values of the technology object within STEP 7 (TIA Portal). The current parameter values of the technology object are displayed in the property dialog of the module (read-only).

SSI absolute encoder (S7-1500)

Frame length

With the parameter assignment of the frame length, you specify the number of bits of an SSI frame of the SSI absolute encoder (Page 22) used. You can find the frame length of your SSI absolute encoder in the data sheet of the encoder. Special bits are also included in the frame length. A parity bit does not count in the frame length.

A frame length of between 10 bits and 40 bits is permitted. The default setting is "13 Bit".

You can find two examples of the SSI frame format at Examples of the frame format (Page 120).

Code type

You use the parameter assignment of the code type to specify whether the encoder supplies Dual code or Gray code.

You can select from the following options:

Code type	Meaning
Gray (default)	The gray-coded position value returned by the SSI absolute encoder is converted to dual code.
Dual	The value returned by the SSI absolute encoder is not converted.

Transmission rate

With the parameter assignment of the transmission rate, you specify the data transmission rate between the technology module and the SSI absolute encoder.

You can select from the following options:

- 125 kHz (preset)
- 250 kHz
- 500 kHz
- 1 MHz
- 1.5 MHz
- 2 MHz

The maximum transmission rate depends on the cable length and the technical specifications of the SSI absolute encoder. For additional information, refer to the product manual of the TM PosInput and the encoder description.

Monoflop time

With the parameter assignment of the monoflop time, you specify the idle time between two SSI frames.

The configured monoflop time must be at least equal to the monoflop time of the SSI absolute encoder used. You can find this value in the technical specifications of the SSI absolute encoder.

You can select from the following options:

- Automatically (default)
- 16 µs
- 32 µs
- 48 µs
- 64 µs

Note

If you select the "Automatic" option, the monoflop time automatically adapts to the encoder used.

In isochronous mode, the "Automatic" option corresponds to a monoflop time of 64 µs. If the monoflop time of the employed SSI absolute value encoder is less than 64 µs, you can select the value of the encoder to achieve faster isochronous times.

Parity

With the parameter assignment of the parity, you specify whether the SSI absolute encoder transfers a parity bit.

If, for example, a 25-bit encoder with parity is assigned, the technology module reads 26 bits. A parity error is indicated by the technology object at the ErrorID output parameter with the value 80A2.

Bit number LSB of the position value

This parameter is used to specify the bit number of the LSB (least significant bit) of the position value in the frame of the SSI absolute encoder. In this way you limit the range in the frame that supplies the position value.

The value must be less than the bit number of the MSB of the position value. The difference between the bit numbers of the MSB and the LSB of the position value must be less than 32.

The default setting is "0".

Note

If you have selected the Code type "Gray", only the range from the LSB to the MSB of the position value is converted to binary code.

Bit number MSB of the position value

This parameter is used to specify the bit number of the MSB (most significant bit) of the position value in the frame of the SSI absolute encoder. In this way you limit the range in the frame that supplies the position value.

The value must be less than the frame length and higher than the bit number of the LSB of the position value. The difference between the bit numbers of the MSB and the LSB of the position value must be less than 32.

The default setting is "12".

Note

If you have selected the Code type "Gray", only the range from the LSB to the MSB of the position value is converted to binary code.

Invert direction

You can use this parameter to invert the values supplied by the SSIabsolute encoder. This allows you to adapt the detected direction of the encoder to the motor's direction of rotation.

Note

This parameter acts only on the range from the LSB to the MSB of the position value in the frame.

SSI frame

You can also set the following parameters in the graphic with drag-and-drop:

- Frame length
- Bit number LSB of the position value
- Bit number MSB of the position value

Complete SSI frame

If "Complete SSI frame" has been selected as the measured variable, the module returns the least significant 32 bits of the unprocessed current SSI frame as the measured value. The graphic shows the corresponding meaning of a supplied bit. The following abbreviations are used:

V	Value: Position value as gray or dual code
S	Special: Special bit
P	Parity: Parity bit When you have configured a parity bit, the module returns the least significant 31 bits of the SSI frame *and* the parity bit.

Behavior of a DI (SSI_Absolute_Encoder) (S7-1500)

Setting function of the DI

By configuring a digital input, you specify which functions the digital input triggers at switching.

You can select from the following options:

Function of a digital input	Meaning	Additional option-specific parameters
Capture (Page 31)	The configured edge at the respective digital input saves the current position value as a Capture value. The technology object displays the Capture value at the output parameter CapturedValue. The function can only be used for one of the two digital inputs.	• Input delay • Edge selection • Frequency of the Capture function
Digital input without function	No technological function is assigned to the respective digital input. You can read the signal state of the digital input via the respective static variable of the technology object: • UserStatusFlags.StatusDI0 • UserStatusFlags.StatusDI1	• Input delay

Note

You can choose the "Capture" function only in operating mode "Use position value (SSI absolute value) as reference".

Input delay

By configuring the input delay, you suppress interferences at the digital inputs. Signals with a pulse duration below the configured input delay are suppressed.

You can select from the following input delays:

- None
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms
- 1.6 ms
- 3.2 ms
- 12.8 ms
- 20 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs.

Note

You configure the input delay under "Behavior of DI0" for all digital inputs together. The input delay is also displayed under "Behavior of DI1".

Edge selection

You can use this parameter to specify the edge of the digital input at which the configured function is triggered for the "Capture" function.

You can select from the following options:

- At rising edge (default)
- At falling edge
- At rising and falling edge

Frequency of the Capture function

This parameter is used to define the frequency of Capture events (Page 31):

You can select from the following options:

Option	Meaning
Once	The first configured edge at the respective digital input saves the current counter value as Capture value.
Periodic (default)	Each configured edge at the respective digital input saves the current counter value as a Capture value.

Note

This parameter is available for SSI_Absolute_Encoder as of V3.0.

Behavior of a DQ (SSI_Absolute_Encoder) (S7-1500)

Operating mode

With the operating mode, you specify the reference value with which the comparison functions work.

Operating mode	Meaning
Use position value (SSI absolute value) as reference (default)	The comparison functions and hardware interrupts for compare events work with the position value.
Use measured value as reference	The comparison functions and hardware interrupts for compare events work with the measured value.

Note

You configure the operating mode under "Behavior of DQ0" for both digital outputs together. The operating mode is also displayed under "Behavior of DQ1".

Set output

With the parameter assignment of a digital output, you specify the condition upon which the digital output switches.

You can select from the following options depending on the operating mode:

Function of a digital output (Page 44) in operating mode "Use position value (SSI absolute value) as reference"	Meaning	Additional option-specific parameters
Between comparison value and high limit (default)	The respective digital output is active if: Comparison value <= position value <= maximum position value	• Comparison value 0 • Comparison value 1 • Hysteresis (in increments)
Between comparison value and low limit	The respective digital output is active if: Minimum position value <= Position value <= Comparison value	• Comparison value 0 • Comparison value 1 • Hysteresis (in increments)
Between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 0 <= position value <= comparison value 1	• Comparison value 0 • Comparison value 1 • Hysteresis (in increments)
At comparison value for a pulse duration	The respective digital output is active once for the assigned time and direction of the position value change when the position value is equal to the comparison value or has fallen below or exceeded it.	• Comparison value 0 • Comparison value 1 • Count direction • Pulse duration • Hysteresis (in increments)
After set command from CPU until comparison value	When a set command is sent from the CPU, the respective digital output is active for the assigned direction of the position value change until the position value is equal to the comparison value or has fallen below or exceeded it.	• Comparison value 0 • Comparison value 1 • Count direction • Hysteresis (in increments)
Use by user program	The respective digital output can be switched by the CPU via the control interface (Page 39).	—

Note

You can only select the "Between comparison value 0 and 1" function for digital output DQ1 and only if you have selected the "Use by user program" function for digital output DQ0.

Function of a digital output (Page 47) in operating mode "Use measured value as reference"	Meaning	Additional option-specific parameters
Measured value >= comparison value (default)	The respective digital output is active if the measured value is greater than or equal to the comparison value.	• Comparison value 0 • Comparison value 1
Measured value <= comparison value	The respective digital output is active if the measured value is less than or equal to the comparison value.	• Comparison value 0 • Comparison value 1
Between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 0 <= measured value <= comparison value 1	• Comparison value 0 • Comparison value 1

Function of a digital output (Page 47) in operating mode "Use measured value as reference"	Meaning	Additional option-specific parameters
Not between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 1 <= measured value <= comparison value 0	• Comparison value 0 • Comparison value 1
Use by user program	The respective digital output can be switched by the CPU via the control interface (Page 39).	—

Note

You can select the "Between comparison value 0 and 1" and "Not between comparison value 0 and 1" functions only for digital output DQ1 and only if you have selected the "Use by user program" function for digital output DQ0.

Comparison value 0

Operating mode "Use position value (SSI absolute value) as reference"

With the parameter assignment of the comparison value (Page 44), you specify the position value at which the digital output DQ0 switches as a result of the selected comparison event.

If you use an SSI absolute encoder with a position value length of up to 31 bits, you must enter a positive integer (DINT) with a value between 0 and $2^{(MSB-LSB+1)}$-1. If you use an SSI absolute encoder with a position value length of 32 bits, you must enter a signed integer (DINT) with a value between –2147483648 and 2147483647.

If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The default setting is "0".

Operating mode "Use measured value as reference"

With the parameter assignment of the comparison value (Page 47), you specify the measured value at which the digital output DQ0 switches as a result of the selected comparison event.

You must enter a floating point number (REAL). If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The minimum value is -7.922816 x 10^{28}. The default setting is "0.0". The unit of the comparison value depends on the measured variable.

Comparison value 1

Operating mode "Use position value (SSI absolute value) as reference"

With the parameter assignment of the comparison value (Page 44), you specify the position value at which the digital output DQ1 switches as a result of the selected comparison event.

If you use an SSI absolute encoder with a position value length of up to 31 bits, you must enter a positive integer (DINT) with a value between 0 and $2^{(MSB-LSB+1)}$-1. If you use an SSI absolute encoder with a position value length of 32 bits, you must enter a signed integer (DINT) with a value between –2147483648 and 2147483647.

If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The default setting is "10".

Operating mode "Use measured value as reference"

With the parameter assignment of the comparison value (Page 47), you specify the measured value at which the digital output DQ1 switches as a result of the selected comparison event.

You must enter a floating point number (REAL). If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The maximum value is 7.922816 x 10^{28}. The default setting is "10.0". The unit of the comparison value depends on the measured variable.

Count direction

You use this parameter to specify the direction of position value change for which the selected function is valid.

You can select from the following options:

Direction of position value change	Meaning
In both directions (default)	The comparison and switching of the respective digital output is carried out regardless of whether the position value increases of decreases.
Up	The comparison and switching of the respective digital output only takes place when the position value increases.
Down	The comparison and switching of the respective digital output only takes place when the position value decreases.

The parameter can be configured for the following functions:

- At comparison value for a pulse duration
- After set command from CPU until comparison value

Pulse duration

By configuring the pulse duration for the function "At comparison value for a pulse duration", you specify the number of milliseconds for which the respective digital output is active.

A value from 0.1 to 6553.5 ms is permissible.

The default setting is "500.0", which is equivalent to a pulse duration of 0.5 s.

Hysteresis (in increments)

By configuring the hysteresis (Page 58), you can define a range around the comparison values. For the functions "Between comparison value and upper counter limit" and "Between comparison value and lower counter limit" the hysteresis also applies at the counter limits. Within the hysteresis range, the digital outputs cannot switch again until the position value has left this range once.

Choose a small enough hysteresis. When the hysteresis range, starting from the configured comparison value, spans the entire position value range, proper functioning of the comparison values cannot be guaranteed.

If a comparison value lies so close to the counter limit that the hysteresis range would extend beyond this counter limit, the hysteresis range ends there.

If you enter "0", the hysteresis is turned off. You can enter a value between 0and 255. The default setting is "0".

Note

You configure the hysteresis under "Behavior of DQ0" for both digital outputs together. The hysteresis is also displayed under "Behavior of DQ1".

Note

The hysteresis is only available in operating mode "Use position value (SSI absolute value) as reference".

Specify measured value (SSI_Absolute_Encoder) (S7-1500)

Measured variable

With this parameter you specify whether the technology module is to provide a certain measured variable (Page 54) or the complete SSI frame.

You can select from the following options:

Option	Meaning	Additional option-specific parameters
Frequency (default)	The measured variable shows the number of increments per second, where each increment corresponds to one position value change. The value is a floating point number (REAL). The unit is Hz. The technology object displays the measured value at the output parameter MeasuredValue.	• Update time
Period	The measured variable is the average period between two increments of the position value. The value is an integer (DINT). The unit is s. The technology object displays the measured value at the output parameter MeasuredValue.	• Update time
Velocity	The measured variable is a velocity. Examples of a velocity measurement can be found in the explanation of the "Increments per unit" parameter. The technology object displays the measured value at the output parameter MeasuredValue.	• Update time • Time base for velocity measurement • Increments per unit
Complete SSI frame	Instead of a measured variable, the first 32 bits of the SSI frame are returned (bit 0 to bit 31). In so doing, special bits that do not belong to the position information are also supplied. A configured inversion of the direction is ignored. The technology object displays the value at the output parameter CompleteSSIFrame. You can find examples under Examples of the frame format (Page 120). This option is only available in operating mode "Use position value (SSI absolute value) as reference".	—

Note

If the increment per revolution is required for calculation of the measured values, it is automatically calculated from the parameterized telegram length as the power of 2, e.g. 8192 increments per revolution with a telegram length of 13 bit. If you are using an SSI absolute encoder whose increment per revolution does not correspond to the power of 2, the calculated measured value may be incorrect for a short time.

Update time

By configuring the update time (Page 54) in milliseconds, you can specify the time interval between two measured value updates. Unsteady measured variables can be smoothed through longer update times

If you enter "0", the measured value is updated once per module-internal cycle. Up to three decimal places can be entered. A value from 0.0 to 25000.0 is permissible. The default setting is "10.0".

Time base for velocity measurement

This parameter defines the time base on which the velocity is to be returned.

You can select from the following options:

- 1 ms
- 10 ms
- 100 ms
- 1 s
- 60 s

The default setting is "60 s".

Increments per unit

With this parameter you define the number of increments per relevant unit that the SSI absolute encoder supplies for the velocity measurement.

You can enter a value between 1 and 65535.

Example 1:

Your absolute encoder operates with a resolution of 12 bits per revolution and performs 4096 increments per revolution. The velocity should be measured in revolutions per minute.

In this case, you need to assign the following parameters:

- Increments per unit: 4096
- Time base for velocity measurement: 60 s

Example 2:

You encoder delivers 10000 increments for travel over one meter. The velocity is to be measured in meters per second.

In this case, you need to assign the following parameters:

- Increments per unit: 10000
- Time base for velocity measurement: 1 s

Examples of the frame format (S7-1500)

Example 1

In this example the SSI absolute encoder has the following specification:

- The encoder has a resolution of 13 bits per revolution and a value range of 12 bits of revolutions. The SSI frame has a length of 25 bits.
- The MSB of the position value is bit 24.
- The LSB of the position value is bit 0.
- The position value is Gray coded.
- A parity bit is not available.

The frame has the following format:

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Bit:	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG
	MSB																								LSB

Position value

MG Multiturn bit as Gray code

SG Singleturn bit as Gray code

Complete SSI frame

If you configure "Complete SSI frame", the technology module returns the unprocessed SSI frame in the return interface right-justified:

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG

MG Multiturn bit as Gray code

SG Singleturn bit as Gray code

Position value feedback value

The position value supplied in Gray code is converted into binary code by the technology module and returned right-justified in the feedback interface:

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	MD	MD	MD	MD	MD	MD	MD	MD	MD	MD	MD	MD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD

MD Multiturn bit as binary code

SD Singleturn bit as binary code

Example 2

In this example the SSI absolute encoder has the following specification:

- The encoder has a resolution of 17 bits per revolution and a value range of 11 bits of revolutions. The SSI frame has a length of34 bits.
- The MSB of the position value is bit 33.
- The LSB of the position value is bit 6.
- The position value is Gray coded.
- The SSI frame has six special bits.
- A parity bit is available. A parity bit does not count in the frame length.

The frame has the following format:

Clock:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
Bit:	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	*	3	2	1	0
	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	S1	S2	P	S3	S4	S5	S6
	MSB																											LSB							

Position value

MG Multiturn bit as Gray code

SG Singleturn bit as Gray code

Sn Special bit n

P Parity bit

Complete SSI frame

If you configure "Complete SSI frame", the technology module returns the least significant 32 bits of the SSI frame as an unprocessed bit string. The technology module returns the bit following the LSB as parity bit. In this example, the technology module therefore returns only the least significant 31 bits of the SSI frame. With the complete SSI frame, you can evaluate the additional special bits in your application.

The returned bit string is structured as follows:

Clock:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MG	MG	MG	MG	MG	MG	MG	MG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	P	S1	S2	S3	S4	S5	S6

MG Multiturn bit as Gray code

SG Singleturn bit as Gray code

Sn Special bit n

P Parity bit

Position value

The position value supplied in Gray code is converted into binary code by the technology module and returned right-justified in the feedback interface: The special bits are ignored in this case. The parity bit is evaluated but is not returned with the position value:

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	MD	MD	MD	MD	MD	MD	MD	MD	MD	MD	MD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD

MD Multiturn bit as binary code

SD Singleturn bit as binary code

1.1.3.5 SSI_Absolute_Encoder programming (S7-1500)

Instruction SSI_Absolute_Encoder (S7-1500)

SSI_Absolute_Encoder

The SSI_Absolute_Encoder instruction is part of the SSI_Absolute_Encoder technology object. It supplies the control and feedback interface of the TM PosInput technology module.

The SSI_Absolute_Encoder instruction thereby forms the software interface between the user program and the technology module. It must be called cyclically from the user program in order to synchronize the input and output data.

The SSI_Absolute_Encoder instruction can be used for the TM PosInput of both the S7-1500 and ET 200SP systems alike. The TM PosInput modules can be used centrally and decentrally. The instruction applies in each case to the channel of the technology module that was assigned to the associated technology object.

Additional information

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

Call instruction in the user program (S7-1500)

The SSI_Absolute_Encoder instruction can be called once per channel, either cyclically or in a time-controlled program. The call is not permitted in an event-controlled interrupt program.

Procedure

Proceed as follows to call the instruction in the user program:

1. Open the CPU folder in the project tree.
2. Open the "Program blocks" folder.
3. Double-click the OB for cyclic program execution.
   The block is opened in the work area.
4. In the "Instructions" window, open the "Technology" group and the "Counting and measurement" folder.
   The folder contains the instruction.
5. Select the instruction and drag it to your OB.
   The "Call options" dialog opens.
6. Select a technology object from the "Name" list or enter the name for a new technology object.
7. Confirm with "OK".

Result

If the technology object does not exist yet, it is added. The instruction is added in the OB. The technology object is assigned to this call of the instruction.

Note

If you click one of the buttons "Configuration", "Commissioning" or "Diagnostics" in the user interface of the instruction, the respective editor opens.

1.1.3.6 Commissioning SSI_Absolute_Encoder (S7-1500)

Commissioning the technology object (S7-1500)

A graphic display of the block in the commissioning editor helps you with commissioning and the function test for the technology object. You can change specific parameters of the SSI_Absolute_Encoder instruction in CPU/IM online mode and monitor their effects.

Requirements

- There is an online connection between STEP 7 (TIA Portal) and the CPU.
- The CPU is in RUN.

- The corresponding SSI_Absolute_Encoder instruction is called cyclically from the user program.
- The parameters of the technology object are not overwritten by the user program.

Procedure

To open the commissioning editor of a technology object and to simulate a parameter value change, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the SSI_Absolute_Encoder technology object in the project tree.
3. Double-click on the "Commissioning" object.
   The functions for commissioning the SSI_Absolute_Encoder technology object are displayed.
4. In the commissioning dialog, click on the "Monitor all" button.
   The parameters (online values) of the SSI_Absolute_Encoder technology object are loaded and displayed.
5. If the parameter you want to change has a text box, enter the new value there.
6. Select the check box of the parameter.
   The new parameter value becomes effective and the effects of the change are simulated.

Online mode

In online mode, you can modify the following parameters to test the technology object function:

- New comparison value 0 (NewReferenceValue0 or NewReferenceValue0_M)
- New comparison value 1 (NewReferenceValue1 or NewReferenceValue1_M)
- Enable Capture (CaptureEnable)
- Acknowledgment of signaled error states (ErrorAck)
- Resetting the status flag (EventAck)

1.1.3.7 SSI_Absolute_Encoder diagnostics (S7-1500)

Monitoring counter values, measured values, DIs and DQs (S7-1500)

The diagnostic functions are used to monitor the position input and measuring functions.

Requirements

- There is an online connection between STEP 7 (TIA Portal) and the CPU.
- The CPU is in RUN.

Procedure

To open the display editor for the diagnostic functions, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the SSI_Absolute_Encoder technology object in the project tree.
3. Double-click on the "Diagnostics" object.
4. Click the "Monitor all" button.

Display

The following values are read by the technology object from the feedback interface and displayed:

- Event display/diagnostics information
- Signal states of the digital inputs and digital outputs
- Position value
- Capture value
- Measured value

Additional information on status displays is available in the context-sensitive help for each event in STEP 7 (TIA Portal). When the CPU is in STOP, the status display is not updated.

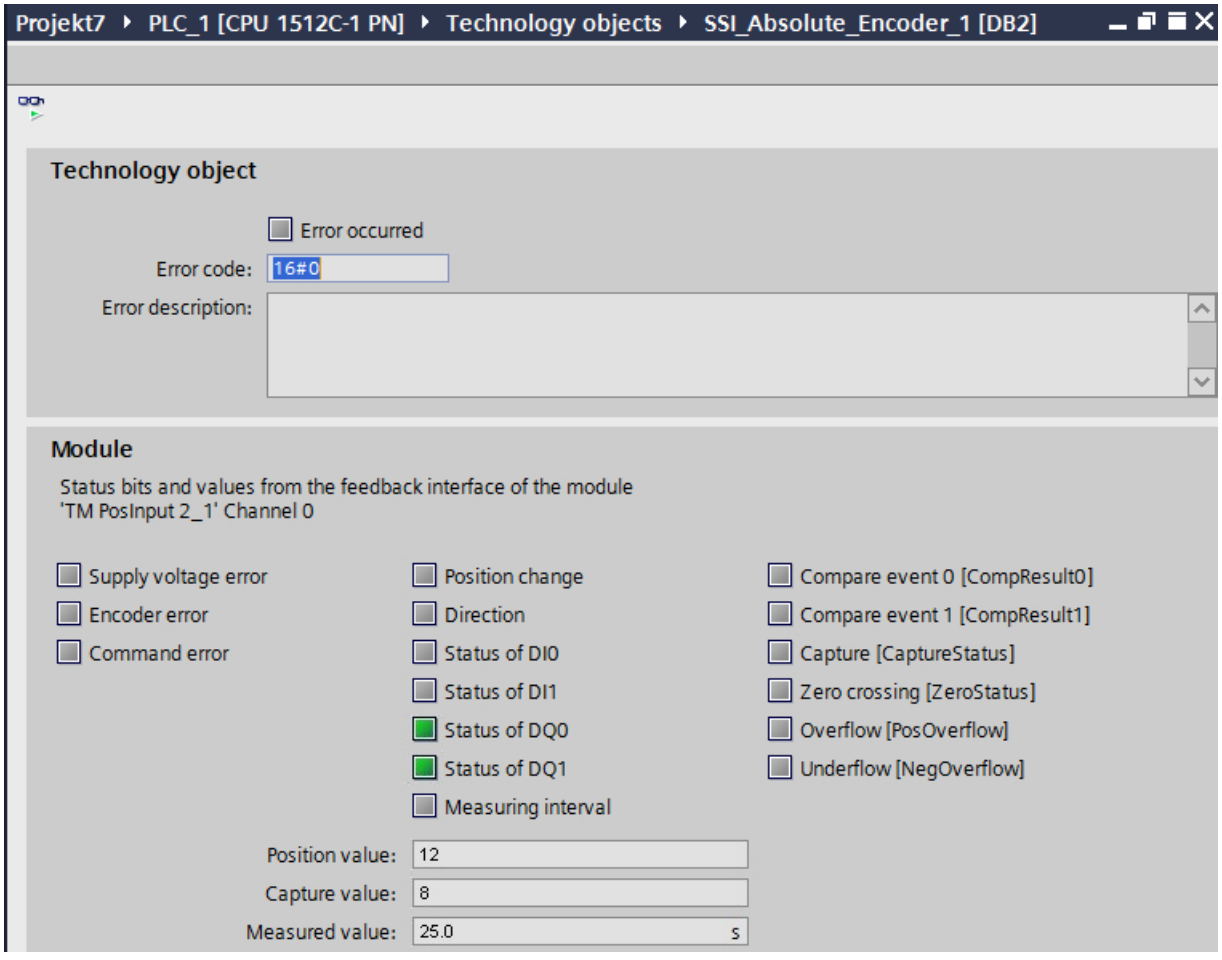

1.1.4 Using the module (S7-1500)

1.1.4.1 Using the technology module (S7-1500)

Convention (S7-1500)

Technology module: We use the term "technology module" in this documentation both for the technology modules TM Count and TM PosInput and the technology component of the compact CPUs.

Configuring a module (S7-1500)

Adding a technology module for hardware configuration (TM Count and TM PosInput) (S7-1500)

Requirement

- The project has been created.
- The CPU S7-1500 has been created.
- At decentralized operation a distributed I/O ET 200 is created.

Procedure

To add a technology module to the hardware configuration, proceed as follows:

1. Open the device configuration of the CPU or IM.
2. Select a module rack.
3. Select the technology module from the module catalog:
   "Technology module > Counting or Position input > Technology module > Article number"
4. Drag the technology module to the required slot in the module rack.

Adding a technology module to hardware configuration (Compact CPU) (S7-1500)

Requirement

The project has been created.

Procedure

To add a Compact CPU to the project tree, follow these steps:

1. Double-click "Add new device".
   The "Add new object" dialog opens.
2. Select Controller".
3. Select the Compact CPU:
   "SIMATIC S7-1500 > CPU > Compact CPU > Article number"
4. Confirm with "OK".

Result

The new Compact CPU is displayed with the following objects in the project tree. Double-click to open the required editor.

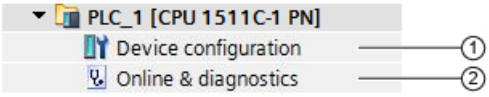

	Object	Description
①	Device configuration (Page 129)	In the Inspector window (per channel): • Activation of the counter (Page 139) • Assign signals to inputs and outputs (Page 139) • Setting the reaction to CPU STOP (Page 129) • Enable diagnostic interrupts (Page 131) • Setting the operating mode (Page 132) • Enable hardware interrupts (Page 136) • Setting the module addresses
②	Online & Diagnostics (Page 165)	• Hardware diagnostics • Obtain information about the Compact CPU • Run firmware update

Open hardware configuration (S7-1500)

Opening

To open the hardware configuration, follow these steps:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Click on the module.

Parameter assignment options (S7-1500)

Counting, measuring and position input with SSI absolute encoder

For the counting and measuring function as well as the position input with an SSI absolute encoder, you have the following alternatives for parameter assignment and control of the technology module:

- Configuration of a technology object and control using the related instruction:
  When using an incremental or pulse encoder, we recommend the simple configuration with graphic support using thee High_Speed_Counter technology object. A detailed description of this configuration can be found in section Technology Object High_Speed_Counter.
  When using an SSI absolute value encoder, we recommend the simple configuration with graphic support using the SSI_Absolute_Encoder technology object. A detailed description of this configuration can be found in section Technology Object High_Speed_Counter.
  For configuration of a technology object, select the operating mode (Page 132) "Operating with "Counting and measurement" technology object".
- Parameter setting via hardware configuration and control via the control and feedback interface of the technology module:
  Select the operating mode (Page 132) "Manual operation (without technology object)" for this.
  A description of the control and feedback interface for TM Count and TM PosInput is available in the following sections:
  Assignment of the control interface (Page 166)
  Assignment of the feedback interface (Page 168)
- Parameter settings via GSD file and control via the control and feedback interface of the technology module.
  A description of the control and feedback interface for TM Count and TM PosInput is available in the following sections:
  Assignment of the control interface (Page 166)
  Assignment of the feedback interface (Page 168)

Position input for Motion Control

Alternatively, you have the option of using the technology module for position input for Motion Control.

Select the operating mode (Page 132) "Position input for "Motion Control" technology object" and assign the parameters of the encoder using the module parameters (Page 133) in the device configuration of the technology module. You perform the rest of the configuration of this application using an axis technology object of S7-1500 Motion Control .

Basic parameters (S7-1500)

Reaction to CPU STOP (S7-1500)

Reaction to CPU STOP

You set the response of the technology module for each channel to CPU STOP in the basic parameters of the device configuration.

Option	Meaning
Continue operation	The technology module remains fully functional. Incoming count pulses are processed or the actual position is read. The digital outputs continue to switch according to the parameter assignment.
Output substitute value	The technology module outputs the configured substitute values at the digital outputs until the next CPU STOP-RUN transition. The technology module is returned to its startup state after a STOP-RUN transition: The counter value is set to the Start value (with incremental encoders or pulse encoders) and the digital outputs switch according to the parameter assignment.
Keep last value	The technology module outputs the values at the digital outputs that were valid when the transition to STOP took place until the next CPU STOP-RUN transition. If a digital output with the function "In case of comparison value for a pulse duration" is set at CPU stop, the digital output is reset after expiry of pulse duration. The technology module is returned to its startup state after a STOP-RUN transition: The counter value is set to the Start value (with incremental encoders or pulse encoders) and the digital outputs switch according to the parameter assignment.

Substitute value for DQ0 (TM Count und TM PosInput)

This parameter lets you specify which value the technology module is to output to the digital output DQ0 in the event of a CPU STOP under "Output substitute value".

Note

In operating mode "Operating with "Counting and measurement" technology object", you assign this parameter using the technology object.

Substitute value for DQ0 (Compact CPU)

You can use this parameter to specify the value to be output for a STOP of the Compact CPU in the feedback interface for DQ0 in the context of the "Substitute value for DQ0" behavior.

Note

In operating mode "Operating with "Counting and measurement" technology object", you assign this parameter using the technology object.

Substitute value for DQ1

This parameter lets you specify which value the technology module is to output to the digital output DQ1 in the event of a CPU STOP under "Output substitute value".

Note

In operating mode "Operating with "Counting and measurement" technology object", you assign this parameter using the technology object.

Diagnostic interrupts (TM Count and TM PosInput) (S7-1500)

The technology module can trigger additional diagnostic interrupts when you enable the diagnostic interrupts in the basic parameters. These diagnostic interrupts are processed in an interrupt OB.

Detailed information on the error event is available in the error organization block with the instruction "RALRM" (read alarm supplementary information) and in the Diagnostics function manual (https://support.industry.siemens.com/cs/ww/en/view/59192926), section "System diagnostics by means of the user program".

Enable diagnostic interrupt on wire break

You use this parameter to specify whether a diagnostic interrupt is to be triggered in the event of wire breaks of the utilized signals for the following encoders:

- Push-pull 24 V encoder (Page 60)
- RS422 encoder (Page 62) (additionally monitored for short-circuits and incorrect supply voltage)
- SSI absolute encoder (Page 64) (additionally monitored for short-circuits and incorrect supply voltage)

Note

If you use an encoder with a different sensor type or interface standard, wire break cannot be detected.

Enable additional diagnostic interrupts

You use this parameter to specify whether diagnostic interrupts are to be triggered for additional errors.

See the device manual for the technology module to find out which errors during operation can trigger a diagnostic interrupt.

Diagnostics interrupts (Compact CPU) (S7-1500)

Enable diagnostic interrupts

A Compact CPU can trigger diagnostic interrupts for certain faults if you have enabled diagnostic interrupts. These diagnostic interrupts are processed in an interrupt OB.

You use this parameter to determine if the Compact CPU should trigger diagnostic interrupts when a given error occurs.

See the device manual for the Compact CPU to find out which errors can trigger a diagnostic interrupt during operation. The diagnostic interrupts are not enabled in the default setting.

Operating mode (S7-1500)

Selection of the operating mode for the channel

This setting defines how the channel counting and measuring functions are to be configured and controlled.

Operating mode	Description
Operating with "Counting and measurement" technology object	A technology object is used for the parameter assignment of the channel. The related instruction for the technology object in the user program provides the access to the control and feedback interfaces of the technology module. You specify the assignment between the technology module/channel and technology object in the basic parameters of the technology object.
Position input for "Motion Control" technology object	The technology module is used for position input for a higher-level Motion Control controller. This operating mode affects all channels of the technology module for TM Count and TM PosInput. Setting the operating mode only affects the given channel of the Compact CPU. Parameter assignment is implemented via the Device configuration of the technology module. Parameter assignment of the encoder signals is implemented by means of the Module parameters (Page 133).
Manual operation (without technology object)	Parameter assignment of the channel is effected via the hardware configuration of the technology module: • Manual operation (incremental or pulse encoder) (Page 140) • Manual operation (SSI absolute encoder) (Page 155) You have direct access to the control and feedback interface of the channel through the user program.
Fast Mode	The technology module is used for very fast detection of the count or position value. No control interface is available. The parameter setting (hardware configuration) of the module is used for the parameter assignment of the channel. You have direct access to the feedback interface of the channel through the user program. This operating mode is not supported by every technology module.

Note

GSD file

When using a GSD file you determine the operating mode via the selection of the module name in the hardware catalog. You can select manual operation or Fast Mode in the process.

Selection of the operating mode for the channel

You use this parameter in "Manual operation" to specify the main task for which the channel of the technology module is used. This defines the setting options under "Parameters" (hardware configuration).

Operating mode	Description
Counting (Page 20) / Position input (Page 22)	The main function of the channel is counting or position input. The comparison functions and hardware interrupts work with the counter value or position value. The measured values are available concurrently.
Measuring (Page 50)	The main function of the channel is counting. The comparison functions (Page 47) and hardware interrupts for compare events work with the measured value. The counter value is available concurrently.

Note

GSD file

When using a GSD file you determine the operating type via the selection of the module name in the hardware catalog.

Module parameters (position input for Motion Control) (S7-1500)

In operating mode "Position input for "Motion Control" technology object", the parameters for the encoder signals of the channel are assigned under "Module parameters". The parameters depend on the encoder used.

Note

You cannot use this operating mode with a GSD file.

Module parameters for incremental encoders and pulse encoders

If you use an incremental or pulse encoder, you need to configure the following parameters for the encoder signals of the channel.

- Signal type
- Invert direction
- Signal evaluation
- Filter frequency
- Sensor type (for TM Count)
- Interface standard (for TM PosInput)
- Signal selection for reference mark 0
- Measuring input
- Increments per revolution
- Steps per revolution
- Reference speed

You can find a description of the first six parameters in section Counter inputs (Page 140).

Module parameters for SSI absolute encoders

If you use an TM PosInput with an SSI absolute encoder, you need to configure the following parameters for the encoder signals of the channel.

- Signal type
- Invert direction
- Frame length
- Code type
- Transmission rate
- Monoflop time
- Parity
- Bit number LSB of the position value
- Bit number MSB of the position value
- Measuring input
- Steps per revolution
- Number of revolutions
- Reference speed

You can find additional information on the first new parameters in section Counter inputs (Page 155).

Signal selection for reference mark 0

You can use this parameter to specify the external reference signal upon which a new reference mark is saved for the encoder position.

You can select from the following options:

Option	Meaning
None[1]	No external reference signal is used.
DI0 (preset for TM Count and TM PosInput)	The current counter value is saved as the new encoder position reference mark upon a positive edge at digital input DI0.
Signal N of incremental encoder (default for Compact CPU[2])	The current counter value is saved as the new encoder position reference mark upon a positive edge of the signal N of incremental encoder.

[1] Only available with Compact CPU

[2] Exceptions: "DI0" is the default for HSC 3 and HSC 6 on the 1511C as well as on the 1512C in compatibility mode

Measuring input (TM Count and TM PosInput)

The parameter specifies that the hardware digital input DI1 serves as a measuring input. For rising, falling or both edges of the DI1, the current counter value or position value is saved as the current encoder position. The value of the parameter cannot be changed.

Measuring input (compact CPU)

This parameter is used to define the hardware input which is used as the external measuring input for saving the encoder position.

You can select from the following options:

Option	Meaning
None (default)	No external measuring input is used.
DI1	The hardware digital input DI1 serves as the measuring input. For rising, falling or both edges of the DI1, the current counter value is saved as current encoder position.

Increments per revolution

With this parameter you specify the line count of the incremental and pulse encoder. The line count can be found in the data sheet for the encoder.

Steps per revolution (incremental and pulse encoder)

The parameter specifies the number of count pulses per encode rotation.

The number of count pulses depends on the increment per rotation and the parameterized signal evaluation.

Example:

Your incremental or pulse encoder delivers 2048 increments per revolution. Depending on the signal evaluation the following value is displayed:

Signal evaluation	Steps per revolution
Single	2048
Double	4096
Quadruple	8192

Steps per revolution (SSI absolute encoder)

With this parameter you define the number of increments the SSI absolute encoder supplies per revolution.

Number of revolutions

The parameter specifies how many rotations the value range of the SSI absolute encoder encompasses.

The value is calculated from the parameterized values for the bit numbers LSB and MSB of the position value as well as the steps per revolution.

Reference speed

The encoder transmits the process value of the speed as a percentage of the reference speed. This parameter defines the speed in rpm which is to be correspond to the value 100 % . The reference speed must be identical to the setting in the controller.

You can enter a value between 6.00 and 210000.00. The default setting is "3000.00".

Further configuring

You can perform the further configuration by using an axis or measuring input technology object of S7-1500 Motion Control . See the S7-1500 Motion Control (https://support.industry.siemens.com/cs/ww/en/view/59381279) and S7-1500T Motion Control (https://support.industry.siemens.com/cs/ww/en/view/109481326) function manuals for further details on configuring and commissioning position input and the measuring input technology object.

See also

Position detection for Motion Control (Page 60)

Hardware interrupts (S7-1500)

You can set for each channel the events which are to trigger a hardware interrupt during operation when assigning the basic parameters of the technology module.

In an S7-1500 system, you enter a matching event name for each enabled hardware interrupt and assign a corresponding hardware interrupt OB to each hardware interrupt. If a hardware interrupt is triggered, the corresponding OB is started to evaluate the hardware interrupt data.

Note

In the following cases you can only release only *one* hardware interrupt per module.

- Distributed operation with an S7-300/400 CPU
- Use of a GSD file

A hardware interrupt is triggered if the condition for changing the respective status bit or event bit in the feedback interface is fulfilled.

Lost hardware interrupt

If the hardware interrupts are triggered more quickly by the system than they can be acknowledged, hardware interrupts are lost and the "Hardware interrupt" diagnostics interrupt is signaled.

Hardware interrupts that can be activated

Hardware interrupt	Available in counting mode using		Available in measuring mode using		Available in operating mode Position input for "Motion Control" technology object	Description	Event Type number
	Incremental or pulse encoder	SSI absolute encoder	Incremental or pulse encoder	SSI absolute encoder			
New Capture value available	Yes	Yes	No	No	No	Hardware interrupt when current counter value or position value is saved as Capture value	1000_B
Synchronization of the counter by an external signal	Yes	No	Yes	No	No	Hardware interrupt upon synchronization of the counter by signal N or DI edge	1001_B
Gate start	Yes	No	Yes	No	No	Hardware interrupt when internal gate opens	0001_B
Gate stop	Yes	No	Yes	No	No	Hardware interrupt when internal gate closes	0010_B
Overflow (high counting limit violated)	Based on counter value	No	Based on counter value	No	No	Hardware interrupt when counter value exceeds high counting limit	0011_B
Underflow (low counting limit violated)	Based on counter value	No	Based on counter value	No	No	Hardware interrupt when counter value falls below low counting limit	0100_B
Change of direction[1]	Based on counter value	Based on position value	Based on counter value	Based on position value	No	Hardware interrupt when counter value or position value changes direction	1010_B
Zero crossing	Based on counter value	Based on position value	Based on counter value	Based on position value	No	Hardware interrupt with zero crossing of counter value or position value	0111_B

Hardware interrupt	Available in counting mode using		Available in measuring mode using		Available in operating mode Position input for "Motion Control" technology object	Description	Event Type number
	Incremental or pulse encoder	SSI absolute encoder	Incremental or pulse encoder	SSI absolute encoder			
Comparison event for DQ0 occurred	Based on counter value	Based on position value	Based on measured value	Based on measured value	No	Hardware interrupt when a comparison event for DQ0 occurs as a result of the selected comparison condition. No hardware interrupt when the change of the counter value for an incremental or pulse encoder was not caused by a count pulse	0101_B
Comparison event for DQ1 occurred	Based on counter value	Based on position value	Based on measured value	Based on measured value	No	Hardware interrupt when a comparison event for DQ1 occurs as a result of the selected comparison condition. No hardware interrupt when the change of the counter value for an incremental or pulse encoder was not caused by a count pulse	0110_B

[1] The STS_DIR feedback bit has a default of "0". A hardware interrupt is not triggered when the first change to the counter value or position value immediately after the technology module is switched on is in the down direction.

Default setting

No hardware interrupts are enabled in the default setting.

Additional parameters for Compact CPU (S7-1500)

Introduction (S7-1500)

When using a Compact CPU, the following parameters are additionally available for the signals of the high-speed counter.

Compatibility 1511C (high-speed counters of Compact CPU 1512C-1 PN) (S7-1500)

Front connector assignment same as CPU 1511C

You can use this parameter to specify if the pin assignment for the front connector of the CPU 1511C-1 PN is to be used for the high-speed counter of the CPU 1512C-1 PN:

Option	Meaning
Disabled (default)	CPU 1512C-1 PN uses the pin assignment of the onboard front connector. 1512C-1 PN supports the use of the connections of both front connectors of the digital onboard I/Os for the high-speed counter. The assignment of hardware inputs and outputs for the HSC channels is described in the CPU 1512C-1 PN manual.
Enabled	CPU 1512C-1 PN uses the pin assignment of the front connector of the CPU 1511C-1 PN. 1511C-1 PN supports the use of the connections of the first front connector of the digital onboard I/Os for the high-speed counter. The assignment of hardware inputs and outputs for the HSC channels is described in the CPU 1511C-1 PN manual.

General (S7-1500)

Activate this high-speed counter

You can use this parameter to specify whether the respective high speed counter is to be used:

Option	Meaning
Disabled (default)	The high-speed counter is not used. The counter uses no connections of the onboard front connector and can not trigger interrupts. Writing to its control interface is ignored and its feedback interface returns zero values.
Enabled	The high-speed counter is used. The assignment of HSC addresses to the connections of the onboard front connector is described in the device manual of the Compact CPU.

Hardware inputs/outputs (S7-1500)

Clock generator input (A) / Pulse input (A) / Clock generator forward (A)

This parameter specifies which input is used for the encoder signal A for the respective counter. The value cannot be changed.

Clock generator input (B) / Pulse input (B) / Clock generator forward (B)

If you use an encoder with multiple signals for the respective counter, this parameter specifies which input is used for the encoder signal B. The value cannot be changed.

Reset input (N)

If you use an incremental encoder for the respective counter, this parameter specifies which input is used for the reset input (encoder signal N). The value cannot be changed.

HSC DI0 / HSC DI1

This parameter determines which digital input of the Compact CPU is to be used as the DIm of the counter.

Note

You can configure the input delay for a digital input in the Inspector window of the device configuration under "Properties > DI 16/DQ 16 > Inputs > Channel n".

HSC DQ0

You can read the status of the DQ0 via the feedback interface. You cannot assign DQ0 to a physical digital output of the Compact CPU.

HSC DQ1

This parameter determines which digital output of the Compact CPU is to be used as DQ1. You can select an output with an output delay of 5 µs or 500 µs.

You can find an overview of the output delay for all digital outputs in the device manual of the Compact CPU.

Manual operation (incremental or pulse encoder) (S7-1500)

Counter inputs (S7-1500)

Signal type

You can choose from the following signal types (Page 60):

Signal type	Meaning	Additional option-specific parameters
Incremental encoder (A, B phase-shifted)	An incremental encoder with phase-shifted A and B signals is connected.	• Invert direction • Signal evaluation • Filter frequency • Sensor type or Interface standard
Incremental encoder (A, B, N)	An incremental encoder with phase-shifted signals A and B and a zero signal N is connected.	• Invert direction • Signal evaluation • Filter frequency • Sensor type or Interface standard • Reaction to signal N • Frequency of synchronization • Frequency of the Capture function
Pulse (A) and direction (B)	A pulse encoder (signal A) with direction signal (signal B) is connected.	• Filter frequency • Sensor type or Interface standard

Signal type	Meaning	Additional option-specific parameters
Pulse (A)	A pulse encoder (signal A) without direction signal is connected. You can specify the count direction by means of the control interface (Page 166).	• Filter frequency • Sensor type or Interface standard
Count up (A), count down (B)	Signals for counting up (signal A) and down (signal B) are connected.	• Filter frequency • Sensor type or Interface standard

Invert direction

You can invert the counting direction to adapt it to the process.

The inverting of the direction is configurable and active for the following signal types:

- Incremental encoder (A, B phase-shifted)
- Incremental encoder (A, B, N)

Signal evaluation

By configuring signal evaluation (Page 64), you can specify which edges of the signals are counted.

You can select from the following options:

Signal evaluation	Meaning
Single (default)	The edges of signal A are evaluated during a low level of signal B.
Double	Each edge of signal A is evaluated.
Quadruple	Each edge of signals A and B is evaluated.

The parameter can be assigned with the following signal types:

- Incremental encoder (A, B phase-shifted)
- Incremental encoder (A, B, N)

Filter frequency

By configuring the filter frequency, you suppress interferences at the counting inputs A, B and N.

The selected filter frequency is based on a pulse/break ratio of between around 40:60 and around 60:40. This results in a specific minimum pulse/break time. Signal changes with a duration shorter than the minimum pulse/break time are suppressed.

You can select from the following filter frequencies:

Filter frequency	Minimum pulse/break time
100 Hz	4.0ms
200 Hz	2.0 ms

Filter frequency	Minimum pulse/break time
500 Hz	800 µs
1 kHz	400 µs
2 kHz	200 µs
5 kHz	80 µs
10 kHz	40 µs
20 kHz	20 µs
50 kHz	8.0 µs
100 kHz (preset for Compact CPU)	4.0 µs
200 kHz** (preset for TM Count)	2.0 µs
500 kHz*	0.8 µs
1 MHz* (default at TM PosInput)	0.4 µs

* Only available with TM PosInput

** * Only available with TM Count and TM PosInput

Sensor type (TM Count)

By configuring the sensor type, you specify how the counter inputs are switched for the TM Count.

You can select from the following options:

Sensor type	Meaning
Sourcing output (default)	The encoder/sensor switches the inputs A, B and N to 24VDC.
Sinking output	The encoder/sensor switches the inputs A, B and N to M.
Push-pull (sinking and sourcing output)	The encoder/sensor alternately switches the inputs A, B and N to M and 24VDC.

"Push-pull" is usually selected when incremental encoders are used. If using 2-wire sensors, such as light barriers or proximity switches, you need to select the corresponding wiring "sourcing output" or "sinking output".

The data sheet of the encoder includes information on whether your incremental encoder is a push-pull encoder.

Note

If you use a push-pull encoder and the sensor type "Push-pull (sinking and sourcing output)" is configured, you can monitor the encoder signals for wire break.

Sensor type (compact CPU)

The "Sourcing output" sensor type is set for the Compact CPU and cannot be changed. The encoder or sensor switches the inputs A, B and N to 24V DC.

You can operate both sourcing output encoders as well as push-pull encoders on the Compact CPU. You can find additional information on the sensor type in the data sheet for the encoder.

Interface standard (TM PosInput)

You use this parameter to specify whether the encoder outputs symmetric (RS422) or asymmetric (TTL) signals for the TM PosInput.

You can select from the following options:

Interface standard	Meaning
RS422, symmetrical (default)	The encoder outputs symmetric signals according to the RS422 standard (Page 62).
TTL (5 V), asymmetrical	The encoder outputs asymmetric 5 V signals according to the TTL standard (Page 60).

Note

The RS422 standard provides greater interference immunity than the TTL standard. If your incremental or pulse encoder supports the RS422 standard **and** the TTL standard, we recommend using the RS422 standard.

Reaction to signal N

You use this parameter to specify which reaction is triggered to signal N.

You can select from the following options:

Option	Meaning
No reaction to signal N (default)	The counter is not affected by signal N.
Synchronization at signal N (Page 36)	The counter is set to the start value at signal N. If you select the function "Enable synchronization at signal N" for a digital input, the synchronization depends on the level at the digital input.
Capture at signal N (Page 28)	The counter value is stored in the Capture value at signal N. The use of a digital input and the use of the N signal are not mutually exclusive for the Capture function. The technology object displays the Capture value at the output parameter CapturedValue.

Note

You can only select the reaction to signal N if you have selected the "Incremental encoder (A, B, N)" signal type.

Note

If "Synchronization at signal N" is selected, you can select the function "Enable synchronization at signal N" for a digital input (Page 146).

Note

For High_Speed_Counter as of V3.0, the following applies:

You can choose "Capture at signal N" only in operating mode "Use count value as reference":

Frequency of synchronization

This parameter is used to define the frequency of the following events:

- Synchronization at signal N
- Synchronization as function of a digital input

You can select from the following options:

Option	Meaning
Once (default)	The counter is only set at the first signal N or the first configured edge of the digital input.
Periodic	The counter is set at each signal N or each configured edge of the digital input.

Frequency of the Capture function

This parameter is used to define the frequency of Capture events for the following functions:

- Capture at Signal N
- Capture as function of a digital input

You can select from the following options:

Option	Meaning
Once	The first configured edge at the respective digital input or first rising edge of the N signal saves the current counter value as a Capture value.
Periodic (default)	Each configured edge at the respective digital input or each rising edge of the N signal saves the current counter value as a Capture value.

Counting limits and start value (S7-1500)

High counting limit

You limit the counting range by setting the high counting limit. You can enter a value up to 2147483647 (2^{31}-1). You must enter a value that lies above the low counting limit.

The default setting is "2147483647".

Low counting limit

You limit the counting range by setting the low counting limit. You can enter a value up to -2147483648 (-2^{31}). You must enter a value below the high counting limit.

The default setting is "-2147483648".

Start value

By configuring the start value, you specify the value at which counting is to start and where it is to continue in the case of defined events. You must enter a value between the counting limits or equal to the counting limits.

The default setting is "0".

Additional information

For more information, see Behavior at the counting limits (Page 23) and Counter behavior at gate start (Page 27).

Counter behavior at limits and gate start (S7-1500)

Reaction to violation of a counting limit

You can configure the following characteristics for violation of a counting limit (Page 23):

Reaction	Meaning
Stop counting	If a counting limit is violated, counting is stopped and the internal gate is closed. To restart counting, you must also close and reopen the software/hardware gate.
Continue counting (default)	Counting continues either with the start value or at the opposite counting limit, depending on the additional parameter assignment.

Reset when counting limit is violated

You can reset the counter to the following values when a counting limit is violated:

Reset the value	Meaning
To start value	The counter value is set to the start value.
To opposite counting limit (default)	The counter value is set to the opposite counting limit.

Reaction to gate start

You can set the following Reaction to gate start (Page 27):

Reaction	Meaning
Set to start value	When the gate is opened, the counter value is set to the start value.
Continue with current value (default)	When the gate is opened, counting is continued with the last counter value.

Behavior of a DI (S7-1500)

Setting function of the DI

By configuring a digital input, you specify which functions the digital input triggers at switching.

You can select from the following options:

Function of a digital input	Meaning	Additional option-specific parameters
Gate start/stop (level-triggered)	The level at the respective digital input opens and closes the hardware gate (Page 25).	• Input delay • Select level
Gate start (edge-triggered)	The configured edge at the respective digital input opens the hardware gate (Page 25).	• Input delay • Edge selection
Gate stop (edge-triggered)	The configured edge at the respective digital input closes the hardware gate (Page 25).	• Input delay • Edge selection
Synchronization (Page 32)	The configured edge at the respective digital input sets the counter to the start value. The EVENT_SYNC feedback bit indicates whether a synchronization has occurred.	• Input delay • Edge selection • Frequency of synchronization
Enable synchronization at signal N	The active level at the respective digital input enables synchronization of the counter at signal N (Page 36).	• Input delay • Select level
Capture	The configured edge at the respective digital input saves the current counter value (Page 28) as a Capture value. The use of a digital input and the use of the N signal are not mutually exclusive for the Capture function. The CAPTURED_VALUE value in the feedback interface indicates the Capture value.	• Input delay • Edge selection • Frequency of the Capture function • Behavior of counter value after Capture
Digital input without function	No technological function is assigned to the respective digital input. You can read the signal state of the digital input via the respective feedback bit: • STS_DI0 • STS_DI1 • STS_DI2	• Input delay

Note

With the exception of "Digital input without function", each function can only be used once for each counter, and if used for one digital input is not available to the others.

Input delay (TM Count and TM PosInput)

You use this parameter to suppress signal interference at the digital inputs. Changes to the signal are only detected if they remain stable for longer than the configured input delay time.

You can select from the following input delays:

- None
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms
- 1.6 ms
- 3.2 ms
- 12.8 ms
- 20 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs.

Note

You configure the input delay under "Behavior of DI0" for all digital inputs together. The input delay is also displayed under "Behavior of DI1" and for TM Count also under "Behavior of DI2".

Input delay (compact CPU)

You use this parameter to suppress interference at the digital inputs of the DIn signals. Changes to the signal are only detected if they remain stable for longer than the configured input delay time.

You can configure the input delay for a digital input of a Compact CPU in the Inspector window of the device configuration under "Properties > DI 16/DQ 16 > Inputs > Channel n".

You can select from the following input delays:

- None
- 0.05 ms
- 0.1 ms
- 0.4 ms
- 1.6 ms
- 3.2 ms (default)
- 12.8 ms
- 20 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs.

Select level

You use this parameter to specify the level at which the digital input is active.

You can select from the following options:

Level	Meaning
Active with high level (default)	The respective digital input is active when it is set.
Active with low level	The respective digital input is active when it is not set.

The parameter can be set for the following functions of a digital input:

- Gate start/stop (level-triggered)
- Enable synchronization at signal N

Edge selection

You can use this parameter to specify the edge of the digital input at which the configured function is triggered.

The following options may be available depending on the function selected:

- At rising edge (default)
- At falling edge
- At rising and falling edge

The parameter can be set for the following functions of a digital input:

- Gate start (edge-triggered)
- Gate stop (edge-triggered)
- Synchronization
- Capture

Note

At rising and falling edge" can only be configured for the "Capture" function.

Frequency of synchronization

This parameter is used to define the frequency of the following events:

- Synchronization at signal N
- Synchronization as a function of a digital input

You can select from the following options:

Option	Meaning
Once (default)	The counter is only set at the first signal N or the first configured edge of the digital input.
Periodic	The counter is set at each signal N or each configured edge of the digital input.

Frequency of the Capture function

This parameter is used to define the frequency of Capture events for the following functions:

- Capture at Signal N
- Capture as function of a digital input

You can select from the following options:

Option	Meaning
Once	The first configured edge at the respective digital input or first rising edge of the N signal saves the current counter value as a Capture value.
Periodic (default)	Each configured edge at the respective digital input or each rising edge of the N signal saves the current counter value as a Capture value.

Behavior of counter value after Capture

You can configure the following characteristics for the counter after a capture event (Page 28):

Reaction	Meaning
Continue counting (default)	After saving the current counter value as Capture value, counting is continued unchanged.
Set to start value and continue counting	After saving the current counter value as Capture value, counting is continued with the start value.

Behavior of a DQ (S7-1500)

Set output

With the parameter assignment of a digital output, you specify the condition upon which the digital output switches.

You can select from the following options:

Function of a digital output (Page 40) in the operating mode "Counting"	Meaning	Additional option-specific parameters
Between comparison value and high limit (default)	The respective digital output is active if: Comparison value <= counter value <= high counting limit	• Comparison value 0 • Comparison value 1 • Hysteresis (in increments)
Between comparison value and low limit	The respective digital output is active if: Low counting limit <= counter value <= comparison value	• Comparison value 0 • Comparison value 1 • Hysteresis (in increments)
Between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 0 <= counter value <= comparison value 1	• Comparison value 0 • Comparison value 1 • Count direction • Hysteresis (in increments)
At comparison value for a pulse duration	The respective digital output is active once for the configured time and count direction when the counter value reaches the comparison value.	• Comparison value 0 • Comparison value 1 • Count direction • Pulse duration • Hysteresis (in increments)
After set command from CPU until comparison value	When a set command is sent from the CPU, the respective digital output is active until the counter value is equal to the comparison value.	• Comparison value 0 • Comparison value 1 • Count direction • Hysteresis (in increments)
Use by user program	The respective digital output can be switched by the CPU via the control interface (Page 39).	—

Note

DQ0 of a counter of a Compact CPU

With a Compact CPU, the respective digital output DQ0 is available via the feedback interface, but not as a physical output.

Note

You can only select the "Between comparison value 0 and 1" function for digital output DQ1 and only if you have selected the "Use by user program" function for digital output DQ0.

Note

The "At comparison value for a pulse duration" and "After set command from CPU until comparison value" functions only switch the digital output in question if a count pulse reaches the comparison value. When the counter value is set, e.g. by synchronization, the digital output does not switch.

Function of a digital output (Page 47) in the operating mode "Measuring"	Meaning	Additional option-specific parameters
Measured value >= comparison value (default)	The respective digital output is active if the measured value is greater than or equal to the comparison value.	• Comparison value 0 • Comparison value 1
Measured value <= comparison value	The respective digital output is active if the measured value is less than or equal to the comparison value.	• Comparison value 0 • Comparison value 1
Between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 0 <= measured value <= comparison value 1	• Comparison value 0 • Comparison value 1
Not between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 1 <= measured value <= comparison value 0	• Comparison value 0 • Comparison value 1
Use by user program	The respective digital output can be switched by the CPU via the control interface (Page 39).	—

Note

You can select the "Between comparison value 0 and 1" and "Not between comparison value 0 and 1" functions only for digital output DQ1 and only if you have selected the "Use by user program" function for digital output DQ0.

Comparison value 0 (TM Count and TM PosInput)

Operating mode "Counting":

With the parameter assignment of the comparison value (Page 40), you specify the counter value at which the digital output DQ0 switches as a result of the selected comparison event.

You must enter an integer (DINT) that is greater than or equal to the low counting limit. If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The default setting is "0".

Operating mode "Measurement":

With the parameter assignment of the comparison value (Page 47), you specify the measured value at which the digital output DQ0 switches as a result of the selected comparison event.

You must enter a floating point number (REAL). If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The minimum value is -7.922816×10^{28}. The default setting is "0.0". The unit of the comparison value depends on the measured variable.

Comparison value 0 (compact CPU)

Operating mode "Counting":

With the parameter assignment of the Comparison value (Page 40), you specify the counter value at which the STS_DQ0 bit is set in the feedback interface of the selected comparison event. The digital output DQ0 is not available as a physical output in a Compact CPU.

You must enter an integer (DINT) that is greater than or equal to the low counting limit. If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The default setting is "0".

Operating mode "Measurement":

With the parameter assignment of the Comparison value (Page 47), you specify the measured value at which the STS_DQ0 bit is set in the feedback interface of the selected comparison event. The digital output DQ0 is not available as a physical output in a Compact CPU.

You must enter a floating point number (REAL). If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The minimum value is -7.922816 x 10^{28}. The default setting is "0.0". The unit of the comparison value depends on the measured variable.

Comparison value 1

Operating mode "Counting":

With the parameter assignment of the comparison value (Page 40), you specify the counter value at which the digital output DQ1 switches as a result of the selected comparison event.

You must enter an integer (DINT) that is smaller than or equal to the high counting limit. If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The default setting is "10".

Operating mode "Measurement":

With the parameter assignment of the comparison value (Page 47), you specify the measured value at which the digital output DQ1 switches as a result of the selected comparison event.

You must enter a floating point number (REAL). If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The maximum value is 7.922816 x 10^{28}. The default setting is "10.0". The unit of the comparison value depends on the measured variable.

Count direction

You use this parameter to specify the count direction for which the selected function is valid.

You can select from the following options:

Count direction	Meaning
In both directions (default)	The comparison and switching of the respective digital output take place regardless of the count direction.
Up	The comparison and switching of the respective digital output only takes place when the counter counts up.
Down	The comparison and switching of the respective digital output only takes place when the counter counts down.

The parameter can be configured for the following functions:

- Between Comparison value 0 and 1 (Operating mode "Counting")
- At comparison value for a pulse duration
- After set command from CPU until comparison value

Pulse duration

By configuring the pulse duration for the function "At comparison value for a pulse duration", you specify the number of milliseconds for which the respective digital output is active.

If you enter "0" and the counter value corresponds to the comparison value, the digital output remains active until the next count pulse.

You can enter a value between 0.0 and 6553.5.

The default setting is "500.0", which is equivalent to a pulse duration of 0.5 s.

Hysteresis (in increments)

By configuring the hysteresis (Page 56), you can define a range around the comparison values. For the functions "Between comparison value and upper counter limit" and "Between comparison value and lower counter limit" the hysteresis also applies at the counter limits. Within the hysteresis range, the digital outputs cannot switch again until the counter value is outside the range.

Choose a small enough hysteresis. When the hysteresis range, starting from the configured comparison value, spans the entire counter value range, proper functioning of the comparison values cannot be guaranteed.

If a comparison value lies so close to the counter limit that the hysteresis range would extend beyond this counter limit, the hysteresis range ends there.

If you enter "0", the hysteresis is turned off. You can enter a value between 0and 255. The default setting is "0".

Note

The hysteresis is only available in operating mode "Counting".

Specifying the measured value (S7-1500)

Measured variable

This parameter is used to specify the measured variable (Page 50) to be provided by the technology module. The MEASURED_VALUE value in the feedback interface indicates the measured value.

You can select from the following options:

Measured variable	Meaning	Additional option-specific parameters
Frequency (default)	The measured variable shows the number of increments per second. The value is a floating point number (REAL). The unit is Hz.	• Update time
Period	The measured variable is the average period between two increments. The value is an integer (DINT). The unit is s.	• Update time
Velocity	The measured variable is a velocity. Examples of a velocity measurement can be found in the explanation of the "Increments per unit" parameter.	• Update time • Time base for velocity measurement • Increments per unit

Update time

By configuring the update time (Page 50) in milliseconds, you can specify the time interval between two measured value updates.

The update time and the signal type (Page 140) effect the accuracy of the measurement. In the case of update times of at least 100 ms, the effect of the signal type is negligible.

In the case of update times of less than 100 ms, you achieve maximum measurement accuracy using the following signal types:

- Incremental encoder (A, B phase-shifted) with Signal evaluation "Single"
- Incremental encoder (A, B, N) with Signal evaluation "Single"
- Pulse (A) and direction (B)
- Pulse (A)

In the case of other signal types, measurement accuracy depends on the encoder and cable used.

If you enter "0", the measured value is updated once per module-internal cycle. Up to three decimal places can be entered. A value from 0.0 to 25000.0 is permissible. The default setting is "10.0".

Time base for velocity measurement

This parameter defines the time base on which the velocity is to be returned.

You can select from the following options:

- 1 ms
- 10 ms
- 100 ms
- 1 s
- 60 s

The default setting is "60 s".

Increments per unit

You can use this parameter to define the number of count pulses per relevant unit that the incremental or absolute encoder supplies for the velocity measurement.

The number of count pulses depends on the configured signal evaluation. You can enter a value between 1 and 65535.

Example 1:

You encoder delivers 4000 count pulses for travel over one meter. The velocity is to be measured in meters per second. "Double" is configured as signal evaluation.

In this case, you need to assign the following parameters:

- Increments per unit: 8000
- Time base for velocity measurement: 1 s

Example 2:

Your encoder delivers 4096 count pulses per revolution. The velocity is to be measured in revolutions per minute. "Single" is configured as the signal evaluation.

In this case, you need to assign the following parameters:

- Increments per unit: 4096
- Time base for velocity measurement: 60 s

Manual operation (SSI absolute encoder) (S7-1500)

Counter inputs (S7-1500)

Signal type

If an SSI absolute encoder with the data signal (Signal ID) and cycle signal (Signal C) is connected, select the signal type (Page 64) "Absolute encoder (SSI)".

Invert direction

You can use this parameter to invert the values supplied by the SSI absolute encoder. This allows you to adapt the detected direction of the encoder to the motor's direction of rotation.

Note

This parameter acts only on the range from the LSB to the MSB of the position value in the frame.

Frame length

With the parameter assignment of the frame length, you specify the number of bits of an SSI frame of the SSI absolute encoder (Page 22) used. You can find the frame length of your SSI absolute encoder in the data sheet of the encoder. Special bits are also included in the frame length. A parity bit does not count in the frame length.

A frame length of between 10 bits and 40 bits is permitted. The default setting is "13 Bit".

You can find two examples of the SSI frame format at Examples of the frame format (Page 120).

Code type

You use the parameter assignment of the code type to specify whether the encoder supplies Dual code or Gray code.

You can select from the following options:

Code type	Meaning
Gray (default)	The gray-coded position value returned by the SSI absolute encoder is converted to dual code.
Dual	The value returned by the SSI absolute encoder is not converted.

Transmission rate

With the parameter assignment of the transmission rate, you specify the data transmission rate between the technology module and the SSI absolute encoder.

You can select from the following options:

- 125 kHz (preset)
- 250 kHz
- 500 kHz
- 1 MHz
- 1.5 MHz
- 2 MHz

The maximum transmission rate depends on the cable length and the technical specifications of the SSI absolute encoder. For additional information, refer to the product manual of the TM PosInput and the encoder description.

Monoflop time

With the parameter assignment of the monoflop time, you specify the idle time between two SSI frames.

The configured monoflop time must be at least equal to the monoflop time of the SSI absolute encoder used. You can find this value in the technical specifications of the SSI absolute encoder.

You can select from the following options:

- Automatically (default)
- 16 µs
- 32 µs

- 48 µs
- 64 µs

Note

If you select the "Automatic" option, the monoflop time automatically adapts to the encoder used.

In isochronous mode, the "Automatic" option corresponds to a monoflop time of 64 µs. If the monoflop time of the employed SSI absolute value encoder is less than 64 µs, you can select the value of the encoder to achieve faster isochronous times.

Parity

With the parameter assignment of the parity, you specify whether the SSI absolute encoder transfers a parity bit.

If, for example, a 25-bit encoder with parity is assigned, the technology module reads 26 bits. A parity error is reported in the feedback interface by means of the ENC_ERROR bit.

Bit number LSB of the position value

This parameter is used to specify the bit number of the LSB (least significant bit) of the position value in the frame of the SSI absolute encoder. In this way you limit the range in the frame that supplies the position value.

The value must be less than the bit number of the MSB of the position value. The difference between the bit numbers of the MSB and the LSB of the position value must be less than 32.

The default setting is "0".

Note

If you have selected the Code type "Gray", only the range from the LSB to the MSB of the position value is converted to binary code.

Bit number MSB of the position value

This parameter is used to specify the bit number of the MSB (most significant bit) of the position value in the frame of the SSI absolute encoder. In this way you limit the range in the frame that supplies the position value.

The value must be less than the frame length and higher than the bit number of the LSB of the position value. The difference between the bit numbers of the MSB and the LSB of the position value must be less than 32.

The default setting is "12".

Note

If you have selected the Code type "Gray", only the range from the LSB to the MSB of the position value is converted to binary code.

Behavior of a DI (S7-1500)

Setting function of the DI

By configuring a digital input, you specify which functions the digital input triggers at switching.

You can select from the following options:

Function of a digital input	Meaning	Additional option-specific parameters
Capture	The configured edge at the respective digital input saves the current position value as a Capture value. The CAPTURED_VALUE feedback bit indicates the capture value. The function can only be used for one of the two digital inputs.	• Input delay • Edge selection • Frequency of the Capture function
Digital input without function	No technological function is assigned to the respective digital input. You can read the signal state of the digital input via the respective feedback bit: • STS_DI0 • STS_DI1	• Input delay

Note

You can choose the "Capture" function only in operating mode "Use position value (SSI absolute value) as reference".

Input delay

By configuring the input delay, you suppress interferences at the digital inputs. Signals with a pulse duration below the configured input delay are suppressed.

You can select from the following input delays:

- None
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms

- 1.6 ms
- 3.2 ms
- 12.8 ms
- 20 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs.

Note

You configure the input delay under "Behavior of DI0" for all digital inputs together. The input delay is also displayed under "Behavior of DI1".

Edge selection

You can use this parameter to specify the edge of the digital input at which the configured function is triggered for the "Capture" function.

You can select from the following options:

- At rising edge (default)
- At falling edge
- At rising and falling edge

Frequency of the Capture function

This parameter is used to define the frequency of Capture events:

You can select from the following options:

Option	Meaning
Once	The first configured edge at the respective digital input saves the current counter value as Capture value.
Periodic (default)	Each configured edge at the respective digital input saves the current counter value as a Capture value.

Behavior of a DQ (S7-1500)

Set output

With the parameter assignment of a digital output, you specify the condition upon which the digital output switches.

You can select from the following options depending on the operating mode:

Function of a digital output (Page 44) in the operating mode "Position input"	Meaning	Additional option-specific parameters
Between comparison value and high limit (default)	The respective digital output is active if: Comparison value <= position value <= maximum position value	• Comparison value 0 • Comparison value 1 • Hysteresis (in increments)
Between comparison value and low limit	The respective digital output is active if: Minimum position value <= Position value <= Comparison value	• Comparison value 0 • Comparison value 1 • Hysteresis (in increments)
Between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 0 <= position value <= comparison value 1	• Comparison value 0 • Comparison value 1 • Count direction • Hysteresis (in increments)
At comparison value for a pulse duration	The respective digital output is active once for the assigned time and direction of the position value change when the position value is equal to the comparison value or has fallen below or exceeded it.	• Comparison value 0 • Comparison value 1 • Count direction • Pulse duration • Hysteresis (in increments)
After set command from CPU until comparison value	When a set command is sent from the CPU, the respective digital output is active for the assigned direction of the position value change until the position value is equal to the comparison value or has fallen below or exceeded it.	• Comparison value 0 • Comparison value 1 • Count direction • Hysteresis (in increments)
Use by user program	The respective digital output can be switched by the CPU via the control interface (Page 39).	—

Note

You can only select the "Between comparison value 0 and 1" function for digital output DQ1 and only if you have selected the "Use by user program" function for digital output DQ0.

Function of a digital output (Page 47) in the operating mode "Measuring"	Meaning	Additional option-specific parameters
Measured value >= comparison value (default)	The respective digital output is active if the measured value is greater than or equal to the comparison value.	• Comparison value 0 • Comparison value 1
Measured value <= comparison value	The respective digital output is active if the measured value is less than or equal to the comparison value.	• Comparison value 0 • Comparison value 1
Between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 0 <= measured value <= comparison value 1	• Comparison value 0 • Comparison value 1

Function of a digital output (Page 47) in the operating mode "Measuring"	Meaning	Additional option-specific parameters
Not between comparison value 0 and 1	The digital output DQ1 is active if: Comparison value 1 <= measured value <= comparison value 0	• Comparison value 0 • Comparison value 1
Use by user program	The respective digital output can be switched by the CPU via the control interface (Page 39).	—

Note

You can select the "Between comparison value 0 and 1" and "Not between comparison value 0 and 1" functions only for digital output DQ1 and only if you have selected the "Use by user program" function for digital output DQ0.

Comparison value 0

Operating mode "Position input":

With the parameter assignment of the comparison value (Page 44), you specify the position value at which the digital output DQ0 switches as a result of the selected comparison event.

If you use an SSI absolute encoder with a position value length of up to 31 bits, you must enter a positive integer (DINT) with a value between 0 and $2^{(MSB-LSB+1)}$-1. If you use an SSI absolute encoder with a position value length of 32 bits, you must enter a signed integer (DINT) with a value between –2147483648 and 2147483647.

If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The default setting is "0".

Operating mode "Measurement":

With the parameter assignment of the comparison value (Page 47), you specify the measured value at which the digital output DQ0 switches as a result of the selected comparison event.

You must enter a floating point number (REAL). If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The minimum value is -7.922816 x 10^{28}. The default setting is "0.0". The unit of the comparison value depends on the measured variable.

Comparison value 1

Operating mode "Position input"

With the parameter assignment of the comparison value (Page 44), you specify the position value at which the digital output DQ1 switches as a result of the selected comparison event.

If you use an SSI absolute encoder with a position value length of up to 31 bits, you must enter a positive integer (DINT) with a value between 0 and $2^{(MSB-LSB+1)}$-1. If you use an SSI absolute encoder with a position value length of 32 bits, you must enter a signed integer (DINT) with a value between –2147483648 and 2147483647.

If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The default setting is "10".

Operating mode "Measurement"

With the parameter assignment of the comparison value (Page 47), you specify the measured value at which the digital output DQ1 switches as a result of the selected comparison event.

You must enter a floating point number (REAL). If you use the DQ function "Between comparison value 0 and 1", the comparison value 0 has to be smaller than comparison value 1. The maximum value is 7.922816 x 10^{28}. The default setting is "10.0". The unit of the comparison value depends on the measured variable.

Count direction

You use this parameter to specify the direction of position value change for which the selected function is valid.

You can select from the following options:

Direction of position value change	Meaning
In both directions (default)	The comparison and switching of the respective digital output is carried out regardless of whether the position value increases of decreases.
Up	The comparison and switching of the respective digital output only takes place when the position value increases.
Down	The comparison and switching of the respective digital output only takes place when the position value decreases.

The parameter can be configured for the following functions:

- Between Comparison value 0 and 1 (operating mode "Position input")
- At comparison value for a pulse duration
- After set command from CPU until comparison value

Pulse duration

By configuring the pulse duration for the function "At comparison value for a pulse duration", you specify the number of milliseconds for which the respective digital output is active.

A value from 0.1 to 6553.5 ms is permissible.

The default setting is "500.0", which is equivalent to a pulse duration of 0.5 s.

Hysteresis (in increments)

By configuring the hysteresis (Page 58), you can define a range around the comparison values. For the functions "Between comparison value and upper counter limit" and "Between comparison value and lower counter limit" the hysteresis also applies at the counter limits. Within the hysteresis range, the digital outputs cannot switch again until the position value has left this range once.

Choose a small enough hysteresis. When the hysteresis range, starting from the configured comparison value, spans the entire position value range, proper functioning of the comparison values cannot be guaranteed.

If a comparison value lies so close to the counter limit that the hysteresis range would extend beyond this counter limit, the hysteresis range ends there.

If you enter "0", the hysteresis is turned off. You can enter a value between 0and 255. The default setting is "0".

Note

The hysteresis is only available in "Position input" operating mode.

Specifying the measured value (S7-1500)

Measured variable

With this parameter you specify whether the technology module is to provide a certain measured variable (Page 50) or the complete SSI frame.

You can select from the following options:

Option	Meaning	Additional option-specific parameters
Frequency (default)	The measured variable shows the number of increments per second, where each increment corresponds to one position value change. The value is a floating point number (REAL). The unit is Hz. The MEASURED_VALUE value in the feedback interface indicates the measured value.	• Update time
Period	The measured variable is the average period between two increments of the position value. The value is an integer (DINT). The unit is s. The MEASURED_VALUE value in the feedback interface indicates the measured value.	• Update time
Velocity	The measured variable is a velocity. Examples of a velocity measurement can be found in the explanation of the "Increments per unit" parameter. The MEASURED_VALUE value in the feedback interface indicates the measured value.	• Update time • Time base for velocity measurement • Increments per unit
Complete SSI frame	Instead of a measured variable, the first 32 bits of the SSI frame are returned (bit 0 to bit 31). In so doing, special bits that do not belong to the position information are also supplied. A configured inversion of the direction is ignored. The MEASURED_VALUE value in the feedback interface indicates the 32 bits. You can find examples under Examples of the frame format (Page 120). This option is only available in operating mode "Use position value (SSI absolute value) as reference".	—

Note

If the increment per revolution is required for calculation of the measured values, it is automatically calculated from the parameterized telegram length as the power of 2, e.g. 8192 increments per revolution with a telegram length of 13 bit. If you are using an SSI absolute encoder whose increment per revolution does not correspond to the power of 2, the calculated measured value may be incorrect for a short time.

Update time

By configuring the update time (Page 50) in milliseconds, you can specify the time interval between two measured value updates. Unsteady measured variables can be smoothed through longer update times

If you enter "0", the measured value is updated once per module-internal cycle. Up to three decimal places can be entered. A value from 0.0 to 25000.0 is permissible. The default setting is "10.0".

Time base for velocity measurement

This parameter defines the time base on which the velocity is to be returned.

You can select from the following options:

- 1 ms
- 10 ms
- 100 ms
- 1 s
- 60 s

The default setting is "60 s".

Increments per unit

With this parameter you define the number of increments per relevant unit that the SSI absolute encoder supplies for the velocity measurement.

You can enter a value between 1 and 65535.

Example 1:

Your absolute encoder operates with a resolution of 12 bits per revolution and performs 4096 increments per revolution. The velocity should be measured in revolutions per minute.

In this case, you need to assign the following parameters:

- Increments per unit: 4096
- Time base for velocity measurement: 60 s

Example 2:

You encoder delivers 10000 increments for travel over one meter. The velocity is to be measured in meters per second.

In this case, you need to assign the following parameters:

- Increments per unit: 10000
- Time base for velocity measurement: 1 s

Online & diagnostics module (S7-1500)

Displaying and evaluating diagnostics (S7-1500)

The online and diagnostics view enables hardware diagnostics. You can also

- Obtain information on the technology module (e.g., Firmware version and serial number)
- Execute a firmware update if required

Procedure (TM Count and TM PosInput)

To open the display editor for the diagnostic functions, follow these steps:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Right-click the module and select "Online & Diagnostics".
1. Select the required display in the diagnostics navigation.

Procedure (Compact CPU)

To open the display editor for the diagnostic functions, follow these steps:

1. Open the Compact CPU folder in the project tree.
2. Double-click on the "Online & diagnostics" object.
3. Select the required display in the diagnostics navigation.

Additional information

Additional information on the diagnostic alarms and possible remedies can be found in the technology module device manual.

Note

Position input for Motion Control

In operating mode "Position input for "Motion Control" technology object", channel diagnostics is not available for the technology module.

Control and feedback interface (TM Count, TM PosInput) (S7-1500)

Information on using the control and feedback interface can be found under Overview of application options (Page 15).

Note

The following description does not apply for the operating modes "Fast Mode" and "Position input for technology object "Motion Control"". You can find a description of the feedback interface in the respective device manual for those technology modules that support the "Fast Mode" operating mode.

Assignment of the control interface (S7-1500)

The user program uses the control interface to influence the behavior of the technology module.

Control interface per channel

The following table shows control interface assignment:

Byte offset to start address Channel 0/1 ↓	↓	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0 … 3	12 … 15	SLOT_0: DINT or REAL: Load value (significance of the value is specified in LD_SLOT_0) Value range: –2147483648 to 2147483647_D or 80000000 to $7FFFFFFF_H$							
4 … 7	16 … 19	SLOT_1: DINT or REAL: Load value (significance of the value is specified in LD_SLOT_1) Value range: –2147483648 to 2147483647_D or 80000000 to $7FFFFFFF_H$							
8	20	LD_SLOT_1				LD_SLOT_0			
9	21	EN_ CAPTURE	EN_ SYNC_DN	EN_ SYNC_UP	SET_DQ1	SET_DQ0	TM_ CTRL_DQ1	TM_ CTRL_DQ0	SW_GATE
10	22	SET_DIR	Reserved					RES_ EVENT	RES_ ERROR
11	23	Reserved							

Notes

Control bit/value	Notes
EN_CAPTURE	Use this bit to enable the Capture function. Resetting this bit resets a set EVENT_CAP in the feedback interface.
EN_SYNC_DN	Use this bit to enable the synchronization of the counter when counting in downward direction with an incremental encoder or pulse encoder. Resetting this bit resets a set EVENT_SYNC in the feedback interface.
EN_SYNC_UP	Use this bit to enable the synchronization of the counter when counting in upward direction with an incremental encoder or pulse encoder. Resetting this bit resets a set EVENT_SYNC in the feedback interface.
LD_SLOT_m	With this load request you specify the meaning of the value in SLOT_m: • 0000 means: No action, idle • 0001 means: Load count value (with incremental or pulse encoder) • 0010 not permitted • 0011 means: Load start value (with incremental or pulse encoder) • 0100 means: Load Comparison value 0 • 0101 means: Load comparison value 1 • 0110 means: Load low counting limit (with incremental or pulse encoder) • 0111 means: Load high counting limit (with incremental or pulse encoder) • 1000 to 1111 not permitted The technology module executes the respective action as soon as LD_SLOT_m changes. If values are loaded simultaneously via LD_SLOT_0 and LD_SLOT_1, the first value is applied internally from SLOT_0 and then the value from SLOT_1 is applied. This may lead to unexpected intermediate states.
RES_EVENT	Use this bit to trigger the reset of the saved events in the EVENT_ZERO, EVENT_OFLW, EVENT_UFLW, EVENT_CMP0, EVENT_CMP1 feedback bits.
Reserved	Reserved bits must be set to 0.
RES_ERROR	Use this bit to trigger the reset of the saved error states LD_ERROR and ENC_ERROR .
SET_DIR	Use this bit to specify the count direction for signal type "Pulse (A)". • 0 means: Up • 1 means: Down
SET_DQ0	Use this bit to set digital output DQ0 when TM_CTRL_DQ0 is set to 0. In the case of the function "After set command from CPU until comparison value",SET_DQ0 is effective regardless of TM_CTRL_DQ0 as long as the counter value is not equal to the comparison value.
SET_DQ1	Use this bit to set digital output DQ1 when TM_CTRL_DQ1 is set to 0. In the case of the function "After set command from CPU until comparison value",SET_DQ1 is effective regardless of TM_CTRL_DQ1 as long as the counter value is not equal to the comparison value.

Control bit/value	Notes
SW_GATE	Use this bit to open and close the software gate when using an incremental encoder or pulse encoder. Together, the software gate and the hardware gate form the internal gate. The technology module only counts when the internal gate is open. • 0 means: Software gate closed • 1 means: Software gate open The control of the hardware gate takes place externally via the digital inputs of the technology module. The hardware gate can be activated through parameter assignment. The software gate cannot be deactivated.
TM_CTRL_DQ0	Use this bit to enable the technological function of digital output DQ0. • 0 means: SET_DQ0 defines the state of DQ0 • 1 means: assigned function defines the state of DQ0
TM_CTRL_DQ1	Use this bit to enable the technological function of digital output DQ1. • 0 means: SET_DQ1 defines the state of DQ1 • 1 means: assigned function defines the state of DQ1

Assignment of the feedback interface (S7-1500)

The user program receives current values and status information from the technology module via the feedback interface.

Feedback interface per channel

The following table shows the assignment of the feedback interface:

Byte offset to start address Channel 0/1 ↓	↓	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0 … 3	16 … 19	COUNT_VALUE: DINT: Current counter value or position value							
4 … 7	20 … 23	CAPTURED_VALUE: DINT: The last acquired Capture value							
8 ... 11	24 ... 27	MEASURED_VALUE: REAL: Current measured value or DWORD: Full SSI frame							
12	28	Reserved					LD_ERROR	ENC_ ERROR	POWER_ ERROR
13	29	Reserved		STS_SW_ GATE	STS_ READY	LD_STS_ SLOT_1	LD_STS_ SLOT_0	RES_EVEN T_ACK	Reserved

Byte offset to start address Channel 0/1 ↓	↓	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
14	30	STS_DI2[1]	STS_DI1	STS_DI0	STS_DQ1	STS_DQ0	STS_GATE	STS_CNT	STS_DIR
15	31	STS_M_ INTERVAL	EVENT_ CAP	EVENT_ SYNC	EVENT_ CMP1	EVENT_ CMP0	EVENT_ OFLW	EVENT_ UFLW	EVENT_ ZERO

[1] For TM PosInput applies: Reserved

Note

Validity of the position value

The position value of an SSI absolute encoder is valid if STS_READY is set to 1 and ENC_ERROR is set to 0. In the module startup STS_READY is set to 0.

Notes

Feedback bit/value	Notes
CAPTURED_VALUE	This DINT value indicates the last detected Capture value. The following external signals can trigger the Capture function: • Rising or falling edge of a digital input • Both edges of a digital input The parameter "Frequency of the Capture function" determines if the function is executed for each configured edge or only once after each enable.
COUNT_VALUE	This DINT value shows the current counter value or position value. If you use an SSI absolute encoder with a position value length of up to 31 bits, the position value is treated unsigned as a positive value and can assume values between 0 and $2^{(MSB-LSB+1)}$-1. If you use an SSI absolute encoder with a position value length of 32 bits, the MSB of the position value corresponds to the sign and the position value can assume values between –2147483648 and 2147483647. If you use a 32 bit position value for the comparison function, the position value is interpreted as DINT.

Feedback bit/value	Notes
ENC_ERROR	This bit indicates that one of the following errors has occurred at the encoder signals (retentive) for the respective technology module: TM Count: • Wire break of digital input A, B, or N (with push-pull encoder) • Invalid transition of A/B signals (with incremental encoder) TM PosInput: • Invalid transition of A/B signals (with incremental encoder) • RS422/TTL error • SSI encoder error or SSI frame error (with SSI absolute encoder) If you have enabled the diagnostic interrupts, the respective diagnostic interrupt is triggered in the event of encoder signal errors. For information on the meaning of the diagnostic interrupts, refer to the manual for the respective technology module. The bit is reset once you have acknowledged the error with RES_ERROR .
EVENT_CAP	This bit indicates that a Capture event has occurred and a counter value has been saved in CAPTURED_VALUE . You reset the status by resetting EN_CAPTURE .
EVENT_CMP0	This bit indicates the saved status that a comparison event (status change) has occurred for the digital output DQ0 based on the selected comparison condition. You reset the status by acknowledgment with RES_EVENT. If the counter value is set to the start value in counting mode, EVENT_CMP0 is not set.
EVENT_CMP1	This bit indicates the saved status that a comparison event (status change) has occurred for the digital output DQ1 based on the selected comparison condition. You reset the status by acknowledgment with RES_EVENT. If the counter value is set to the start value in counting mode, EVENT_CMP1 is not set.
EVENT_OFLW	This bit indicates the saved state which shows that there was a counter value overflow. You reset the status by acknowledgment with RES_EVENT.
EVENT_SYNC	This bit indicates the saved status for which the counter was loaded with the start value by an external reference signal (synchronization) when using an incremental or pulse encoder. You reset the state by resetting EN_SYNC_UP or EN_SYNC_DN .
EVENT_UFLW	This bit indicates the saved state which shows that there was a counter value underflow. You reset the status by acknowledgment with RES_EVENT.
EVENT_ZERO	This bit indicates the saved status with which the counter value or position value experienced a zero crossing. You reset the status by acknowledgment with RES_EVENT.
LD_ERROR	This bit indicates that an error occurred (latching) during loading via the control interface. The loaded values were not applied. When using an incremental or pulse encoder, one of the following conditions is not fulfilled: • Low counting limit <= counter value <= high counting limit • Low counting limit <= start value <= high counting limit • Low counting limit <= comparison value 0/1 <= high counting limit When using an SSI absolute encoder, one of the following conditions is not fulfilled: • 0 <= position value <= maximum position value • 0 <= comparison value 0/1 <= maximum position value The bit is reset once you have acknowledged the error with RES_ERROR .
LD_STS_SLOT_0	This bit indicates by a status change (toggling) that the load request for Slot 0 (LD_SLOT_0) has been detected and executed.
LD_STS_SLOT_1	This bit indicates by a status change (toggling) that the load request for Slot 1 (LD_SLOT_1) has been detected and executed.

Feedback bit/value	Notes
MEASURED_VALUE	This value shows the current measurement value with the REAL data type or the complete SSI frame with the DWORD data type: • Frequency: The mean frequency is calculated at set measuring intervals on the basis of the time profile of the count pulses or position value changes and returned in Hertz as floating point number. • Period duration: The mean period duration is calculated at set measuring intervals on the basis of the time profile of the count pulses or position value changes and returned in seconds as floating point number. • Velocity: The mean velocity is calculated at set measuring intervals on the basis of the time profile of the count pulses or position value changes and other parameters, and returned in the configured unit of measurement. • Complete SSI frame: Instead of a measured variable, the least significant 32 bits of the unprocessed current SSI frame are returned. This provides you with encoder-specific additional bits, such as error bits, in addition to the position value. If the SSI frame is shorter than 32 bits, the complete SSI frame is returned right-aligned and the top unused bits are returned with "0" in the feedback interface. The measured values are returned as signed values. The sign indicates whether the counter value or position value increased or decreased during the relevant time period. The update time is asynchronous to the opening of the internal gate, which means that the update time is not started when the gate is opened. After the internal gate is closed, the last measured value captured is still returned.
POWER_ERROR	For an S7-1500 technology module, this bit indicates that the supply voltage L+ is not available or too low or that the front plug is not plugged in. For an ET 200SP technology module, this bit indicates that the supply voltage L+ is too low. If you have enabled the diagnostic interrupts, the respective diagnostic interrupt is triggered in the event of errors in the supply voltage. For details on the diagnostic interrupts and the corrective measures they require, refer to the device manual for the respective technology module. If the supply voltage L+ is available at a sufficient level once again, POWER_ERROR is automatically set to 0.
RES_EVENT_ACK	This bit indicates that the reset of event bit EVENT_SYNC, EVENT_CMP0, EVENT_CMP1, EVENT_OFLW, EVENT_UFLW, EVENT_ZERO is active.
Reserved	Reserve bits are set to 0.
STS_CNT	This bit indicates that at least one count pulse or a position value change has occurred in the last ca. 0.5 s.
STS_DI0	This bit indicates the status of digital input DI0.
STS_DI1	This bit indicates the status of digital input DI1.
STS_DI2	This bit indicates the status of digital input DI2 of the TM Count .
STS_DIR	This bit indicates the count direction of the last count pulse or the direction of the last position value change. 0 means: Down 1 means: Up
STS_DQ0	This bit indicates the status of digital output DQ0.
STS_DQ1	This bit indicates the status of digital output DQ1.

Feedback bit/value	Notes
STS_GATE	This bit indicates the status of the internal gate when using an incremental or pulse encoder. 0 means: Gate closed 1 means: Gate open Information for TM PosInput: In order for the counting logic including the gate control to function correctly, the startup of the technology module must complete correctly at least once with a connected incremental or pulse encoder (STS_READY at 1). If a connected encoder is not ready yet during startup, the function of the STS_GATE feedback bit is delayed until the encoder for the technology module is available. If the technology module starts up without a connected encoder, the startup does not complete correctly and STS_READY as well as STS_GATE remain at 0. As soon as an encoder is then connected, the startup completes and the STS_GATE functions correctly. An encoder error after a completed startup has no influence on STS_GATE.
STS_M_INTERVAL	This bit indicates that at least one count pulse or a position value change was detected in the previous measurement interval.
STS_READY	This bit indicates that the technology module supplies valid user data. The technology module has been started up and configured.
STS_SW_GATE	This bit indicates the status of the software gate. 0 means: Gate closed 1 means: Gate open

Complete acknowledgment principle

Saving bits are acknowledged according to the complete acknowledgment principle.

The figure below shows an example of the sequence of the complete acknowledgment principle in the event of an overflow:

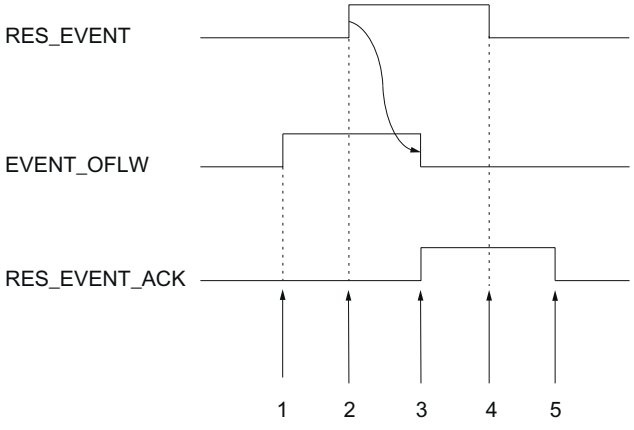

① The EVENT_OFLW feedback bit is set as a saving event upon overflow.

② You set the RES_EVENT control bit to trigger EVENT_OFLW reset.

③ The RES_EVENT_ACK feedback bit is set when reset of EVENT_OFLW is detected.

④ You then rest the control bit RES_EVENT .

⑤ The RES_EVENT_ACK feedback bit is reset.

1.1.4.2 Using the digital module (S7-1500)

Configuring and assigning parameters to the module (S7-1500)

Adding a module to the hardware configuration (S7-1500)

Requirements

- The project has been created.
- The CPU has been created.
- The ET 200 distributed I/O has been created.

Procedure

1. Open the device configuration of the CPU or IM.
2. Select a module rack.
3. Select the digital input module from the module catalog:
   "DI > Digital input module > Article number" or
   "DIQ > Digital input module /Digital output module > Article number"
4. Drag the module to the required slot in the module rack.

Open hardware configuration (S7-1500)

Opening

To open the hardware configuration, follow these steps:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Click on the module.

Counting operating mode (S7-1500)

In counting operating mode or in the counter configuration you can set the following parameters for each channel.

Note

Some of the parameters and options are not available for all digital modules. See the module's device manual for the associated parameters and options.

Channel enabled

You use this parameter to specify whether the respective channel is enabled or disabled.

Each channel is enabled by default.

Input delay

By configuring the input delay, you suppress signal errors at the digital inputs. Changes to the signal are only detected if they are constantly pending longer than the set input delay time.

Isochronous configuration is only possible if there is an input delay of 0.05 ms configured for at least one channel. In isochronous mode, the feedback interface is updated at the time T_i (time for reading the input data).

You can select from the following options:

- 0.05 ms
- 0.1 ms
- 0.4 ms
- 0.8 ms
- 1.6 ms
- 3.2 ms (default)
- 12.8 ms
- 20 ms

Note

If you select the "0.05 ms" option for the input delay, you have to use shielded cables for connection of the digital inputs.

Reaction to violation of a counting limit

The following behavior can be configured for Violation of the counter high limit in the upward direction or the counter low limit in the downward direction (Page 73):

Reaction	Meaning
Stop counting (default at ET200SP and ET200AL)	After a counting limit is violated, the internal gate is closed (automatic gate stop). As a result, the counting process is stopped and the module ignores any further counting signals. The counter value is set to the opposite counting limit. To restart counting, you must close and reopen the software/hardware gate.
Continue counting (default at S7-1500)	After a counting limit is violated, the counter value is set to the opposite counting limit and counting continues.

Edge selection

This parameter is used to specify which edge the respective counter counts:

Edge selection	Meaning
At rising edge (default)	The respective counter counts all rising edges at the digital input.
At falling edge	The respective counter counts all falling edges at the digital input.
At rising and falling edge	The respective counter counts all edges at the digital input.

Count direction

Use this parameter to specify the counting direction of the respective counter.

You can select from the following options:

- Up
- Down

Set output

Use this parameter to specify the Function (Page 76) that controls the STS_DQ feedback bit. You can use the STS_DQ reset bit in order to control a digital output module's digital output.

You can select from the following options:

Option	Meaning
Off (DQ = 0)	Regardless of the counter value, STS_DQ is permanently not set.
Off (DQ = 1)	Regardless of the counter value, STS_DQ is permanently set.
Between comparison value 0 and 1	STS_DQ is set if the counter value between comparison values 0 and 1.
Not between comparison value 0 and 1	STS_DQ is set if the counter value is outside the range between comparison values 0 and 1.
Between comparison value and counter high limit	STS_DQ is set if the counter value is between the comparison value and the counter high limit.
Between comparison value and counter low limit	STS_DQ is set if the counter value is between the comparison value and the counter low limit.

Setting function of the DI

Use this parameter to specify which function the respective digital input DI_{n+4} triggers.

You can select from the following options:

Option	Meaning
Digital input without function	No function is assigned to the respective digital input DI_{n+4}. The signal status of DI_{n+4} can be read by the CPU using the feedback interface.
Gate start/stop	Setting the respective digital input DI_{n+4} opens the hardware gate (Page 75) for DI_n. Resetting the respective digital input DI_{n+4} closes the hardware gate for DI_n.
Count direction	The respective DI_{n+4} digital input determines the counting direction at DI_n, in order to adjust it to the process. If DI_{n+4} is not set, DI_n counts up. If DI_{n+4} is set, DI_n counts down.

Note

If "Count direction" is selected and the counting direction in the process changes, the counting edge is automatically adjusted (opposite edges).

High counting limit

You limit the counting range by setting the counter high limit. The maximum value for the counter high limit depends on the module:

High counting limit	DI 8x24VDC HS, DIQ 16x24VDC/0.5A 8xM12	DI 32x24VDC HF, DI 16x24VDC HF
Maximum value	2147483647 (2^{31}–1)	4294967295 (2^{32}–1)
Default	2147483647	4294967295

You must enter a value that lies above the counter low limit.

Counter low limit

You limit the counting range by setting the counter low limit. The minimum value for the counter low limit depends on the module:

Counter low limit	DI 8x24VDC HS, DIQ 16x24VDC/0.5A 8xM12	DI 32x24VDC HF, DI 16x24VDC HF
Minimum value	–2147483648 (–2^{31})	0 (not configurable)
Default	0	0

You must enter a value below the counter high limit.

Start value

By configuring the start value, you specify the value at which counting is to start. You must enter a value between the counting limits or equal to the counting limits.

The default setting is "0".

Comparison value

By configuring a comparison value (Page 76), you specify the counter value that controls the STS_DQ reset bit based on the comparison function that was selected under "Set output".

You have to enter a value that is greater than or equal to the low counting limit as well as smaller than or equal to the high counting limit.

The default setting depends on the module:

Comparison value	DI 8x24VDC HS, DIQ 16x24VDC/0.5A 8xM12	DI 32x24VDC HF, DI 16x24VDC HF
Default	10	1

Comparison value 0

By configuring a comparison value (Page 76), you specify the counter value that controls the STS_DQ reset bit based on the comparison function that was selected under "Set output".

You must enter a value which is greater than or equal to the counter low limit and less than comparison value 1.

The default setting is "0".

Comparison value 1

By configuring the second comparison value (Page 76), you specify the additional counter value that controls the STS_DQ reset bit based on the comparison function that was selected under "Set output".

You must enter a value which is greater than comparison value 0 and less than or equal to the counter high limit.

The default setting is "10".

Hardware interrupt: Comparison event for DQ occurred

With this parameter, you specify for S7-1500 and ET200AL whether a process alarm is given off when a comparison event occurs based on the comparison function that was selected under "Set output".

The hardware interrupt is not enabled in the default setting.

Online & diagnostics module (S7-1500)

Displaying and evaluating diagnostics (S7-1500)

The online and diagnostics view enables hardware diagnostics. You can also

- Obtain information on the module (e.g., Firmware version and serial number)
- Execute a firmware update if required

Procedure

To open the display editor for the diagnostic functions, follow these steps:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Right-click on the module and select "Online & Diagnostics".
4. Select the required display in the diagnostics navigation.

Additional information

Additional information on the diagnostic alarms and possible remedies can be found in the module's device manual.

1.2 Counting and measurement (S7-300, S7-400)

1.2.1 Counting and measuring (S7-300, S7-400)

1.2.1.1 Counting and measuring basics (S7-300, S7-400)

What is counting and measuring? (S7-300, S7-400)

Counting and measuring

Counting is the capturing and totalizing of events. The counter modules detect and analyze encoder signals (pulses).

The pulses are detected in real time in measuring modes and used to calculate the relevant measured values (e.g. frequency).

Counting basics (S7-300, S7-400)

Counting ranges and count limits

The module counts within various counting limits in the two counting ranges "0 to +32 bits" and "-31 to +31 bits". An high limit violated or an low limit violated is detected at the respective count limits.

In the "-31 to +31 bits" counting range, the counter value is represented in 2's complement.

Counting range		High limit violated	Low limit violated
0 to +32 bits [1]	0 to 4,294,967,295 0 to FFFF FFFFH	When the counter value changes from 4 294 967 295 to 0	When the counter value changes from 0 to 4 294 967 295
-31 to +31 bits	-2,147,483,648 to 2,147,483,647 8000 0000H to 7FFF FFFFH	When the counter value changes from +2 147 483 647 to –2 147 483 648	When the counter value changes from -2,147,483,648 to 2,147,483,647
[1] This counting range is only available with FM x50-1. You can only specify and evaluate hexadecimal values.			

Load value

The load value is the counter value from which the module starts the counting process.

You can assign a load value LOAD_VAL to the module during operation. This will overwrite the initial value.

You can assign this load value **directly**. It will then be accepted directly by the module as a new counter value and loaded in preparation.

You can only load the load value **in preparation**. A load value loaded in preparation is accepted by the module as the new counter value in response to the following events:

- In all Count modes
  - The counting process is started by the interrupting software or hardware gate (the counter continues counting from the current counter value when the software or hardware gate is started if the interrupting gate function is activated; the load value is not applied).
  - Synchronization (via corresponding DI)
  - Latch/Retrigger (via corresponding DI)
- In the "Count once" and " Count periodically" modes also:
  - When the set high count limit is reached for main counting direction up.
  - Reaching 0 when counting down.

Gate control

You can use the hardware gate and software gate to control i.e. start and stop the module's counting processes.

Main counting direction

When you set a main counting direction (up or down), you can limit the maximum count range to a smaller range by setting the high count limit. The set count range is then between 0 and the set high count limit. This can be used to create incrementing or decrementing counting applications, for example. The set main counting direction has no effect on the direction evaluation when the count pulses are detected.

Starting Counts According to Parameterization

Value	Main counting direction	Start value
Load value	None	0
	Up	0
	Down	Parameterized high count limit
Count value	None	0
	Up	0
	Down	Parameterized high count limit
Comparison value 1 and 2	None	0
	Up	0
	Down	Parameterized high count limit
Latch value	None	0
	Up	0
	Down	Parameterized high count limit

Commands for the Counting Modes

You can apply commands to the module counting process during operation:

Command	Description
Open and close gate	The counting process starts when a gate opens and ends when it closes.
Set counter [1])	The counter will be set to the load value via various signals (DI, zero mark).
Latch/Retrigger [2])	Saves the counter level and loads the counter with the load value in response to a positive edge at DI Start.
Latching [2])	Saves the counter level in response to a positive edge at DI.Start.
Measuring times between two edges[3])	Measures the times between two immediately successive edges at the DI Start digital input.
[1]) only with FM 350-1 and FM 450-1 [2]) only with FM 350-1, ET 200S 1Count24V and ET 200S 1Count5V [3]) only with FM 350-1	

Isochronous mode

In isochrone mode, the module accepts control bits and control values from the control interface in each PROFIBUS / PROFINET cycle and returns its response to them within the same cycle.

In each PROFIBUS / PROFINET cycle, the module transfers the counter and latch value that were valid at time T_i and the status bits valid at time T_i.

A counter value influenced by hardware input signals will only be transferred in the same cycle if the input signal occurs before time T_i.

Count modes (S7-300, S7-400)

Overview of the count modes (S7-300, S7-400)

You select module functionality by setting a default operating mode. The table below gives an overview of the counting modes.

Operating mode	Description
Count continuously (Page 182) (with or without software or hardware gate)	The module counts continuously from the current counter value.
Count once (Page 183) (with software gate or hardware gate)	When the gate opens, the module counts from the load value to the count limit.
Count periodically (Page 184) (with software gate or hardware gate)	When the gate opens, the module counts between the load value and the count limit.

Count continuously (S7-300, S7-400)

Description

In this operating mode, the module counts continuously from the counter value:

- If the counter reaches the high limit when counting up and a further count pulse is received, it jumps to the low count limit and continues to count from there without any pulse losses.
- If the counter reaches the low limit when counting down and a further count pulse is received, it jumps to the high limit and continues to count from there without any pulse losses.

The following applies to the ±31 bit counting range:

- The high count limit is set to +2,147,483,647 (2^{31} - 1).
- The low count limit is set to -2,147,483,648 (-2^{31}).

The following applies to the 32 bit counting range:

- The high count limit is set at +4,294,967,295 (2^{32} - 1).
- The low count limit is set to 0 (zero).

Behavior at the Count Limits

If the counter reaches the high or low count limit and a further count pulse is received, then the counter is set to the other count limit. An appropriate status bit is set in the feedback interface and if applicable in the DB:

Count limit reached	Status bit
High count limit	STS_OFLW is set
Low count limit	STS_UFLW is set

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Without gate (default)
- Software gate (Page 200)
- Hardware gate (Page 201), level-controlled or edge-controlled

The figure below shows an example of continuous counting with gate control.

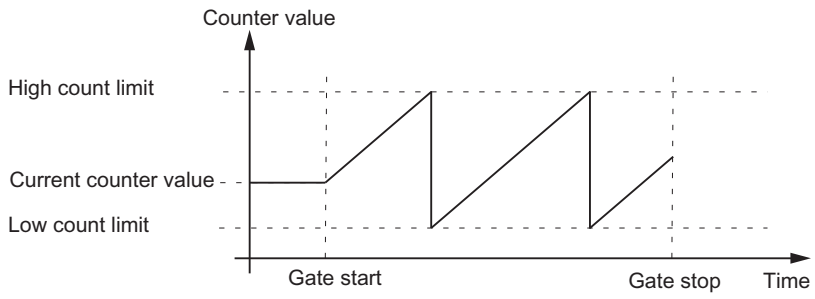

Count once (S7-300, S7-400)

Description

In this operating mode, the module counts once in the parameterized main counting direction and then stops the counting process automatically.

Behavior at the Count Limits

If the counter has reached the high or low count limit and a further count pulse is received, the counter is set

- to the other count limit when counting without a main count direction (Page 185)
- to the load value when counting with a main count direction (Page 185).

The gate is then closed and the counting process terminated, even if the parameter / the control bitSW_GATE is still set or the hardware gate is still open. The appropriate status bit is set in the feedback interface / in the DB:

Count limit reached	Status bit
High count limit	STS_OFLW is set
Low count limit	STS_UFLW is set

If you want to restart the counter, you must reset the parameter / the control bit SW_GATE and/or reopen the hardware gate. The counting process is then continued from the load value.

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 200)
- Hardware gate (Page 201), level-controlled or edge-controlled

The figure below shows an example of continuous counting with load value and gate control.

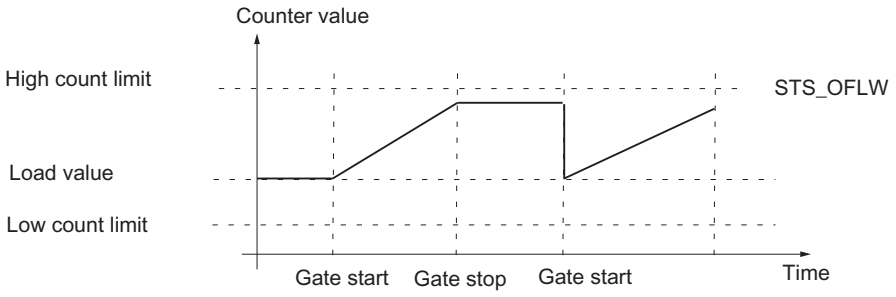

Count periodically (S7-300, S7-400)

Description

In this operating mode, the module counts periodically when the gate is open and does not stop counting at the count limits.

Behavior at the Count Limits

If the counter

- has reached one of the count limits when counting without a main count direction (Page 185)
- has reached the high count limit when counting with the main count direction up (Page 185)
- has reached the low count limit when counting with main count direction down (Page 185)

The gate is then closed and the count is terminated even if the SW_GATE parameter is still set or the hardware gate is still open. The appropriate status bit is set in the feedback interface / in the DB.

Count limit reached	Status bit
High count limit	STS_OFLW is set
Low count limit	STS_UFLW is set

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 200)
- Hardware gate (Page 201), level-controlled or edge-controlled

The figure below shows an example of count periodically with load value and gate control.

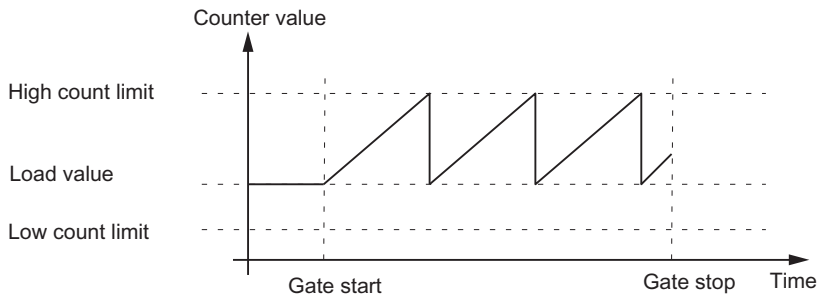

Main count direction (S7-300, S7-400)

A main count direction can only be selected in the "Count once", " Count periodically" and "Dosing" modes.

You can parameterize the following behavior:

- None (no main count direction)
- Up
- Down

Count once without main count direction

When the gate is opened in "Count once" mode without a main count direction, the module will count up or down from the load value until one of the count limits is exceeded.

If one of the count limits is exceeded,

- the gate is closed,
- the STS_OFLW or STS_UFLW bit is set in the feedback interface,
- the counter is set to the other count limit.

The count limits are fixed at the maximum count range.

The STS_ZERO and/or STS_ND bit is set if the counter value is zero (not in FM 350-2).

You must open the gate again to restart the count.

Count Periodically without Main Counting Direction

When the gate is opened in " Count periodically" mode without a main count direction, the module counts up or down from the load value until one of the count limits is exceeded.

If one of the count limits is exceeded

- the STS_OFLW or STS_UFLW bit is set in the feedback interface,
- the counter is set to the load value from which it resumes counting.

The count limits are fixed at the maximum count range.

The STS_ZERO and/or STS_ND bit is set if the counter value is zero (not in FM 350-2).

The counting is resumed when the gate is closed.

Count once with main count direction up

When the gate is opened in "Count once" or "Dosing" mode with the main count direction up, the module counts up or down from the load value until the high count limit is exceeded.

If the high count limit is exceeded,

- the gate is closed,
- the STS_OFLW bit in the feedback interface is set,
- the counter is set to the load value.

The high count limit can be set. The load value has a starting count and can be changed.

You must open the gate again to restart the count.

Count Periodically with Main Counting Direction Up

When the gate is opened in " Count periodically" mode with main count direction up, the module counts up or down from the load value until the high count limit is exceeded.

If the high count limit is exceeded,

- the STS_OFLW bit in the feedback interface is set,
- the counter is set to the load value from which it resumes counting.

The high count limit can be set. The load value has a starting count and can be changed.

The counting is resumed when the gate is closed.

Count once with main count direction down

When the gate is opened in "Count once" or "Dosing" mode with main count direction down, the module counts up or down from the load value until the low count limit is violated.

If the low count limit is under-run,

- the gate is closed,
- the STS_UFLW bit in the feedback interface is set,
- the counter is set to the load value.

The low count limit is fixed at 0; the load value has an initial value and can be changed.

You must open the gate again to restart the count.

Count Periodically with Main Counting Direction Down

When the gate is opened in " Count periodically" mode with main count direction down, the module counts up or down from the load value until the low count limit is violated.

If the low count limit is under-run,

- the STS_UFLW bit in the feedback interface is set,
- the counter is set to the load value from which it resumes counting.

The low count limit is fixed at 0; the load value has an initial value and can be changed.

The counting is resumed when the gate is closed.

Measuring basics (S7-300, S7-400)

Measuring ranges

The modules have the following measuring ranges in measuring modes:

Operating mode	Measuring range
Frequency measurement	0.1 Hz [1] to 650 000 Hz (depending on the module)
Rotational speed measurement	Up to 25,000 /min
Period measurement	Up to 120 s

[1] The low limit of 0.1 Hz is defined by the maximum update time of 10 s. The resolution of the value is 0.001 Hz.

Update time / Integration time

The module updates the measured values cyclically. Set the "Measuring time" parameter as factor n to calculate the update time / integration time.

Calculation of the update time / integration time:

Boundary conditions		Update time / Integration time	Value range of n	
			n_{min}	n_{max}
Non-isochronous mode	Any T_{DP}	n × 10 ms	1	1000 [3]
Isochronous mode [2]	T_{DP} < 10 ms	n × T_{DP}	(10 ms/T_{DP} [ms]) + 1 [1]	1000 [3]
	T_{DP} ≥ 10 ms	n x T_{DP}	1	10000 [3] ms/T_{DP} [ms] [1]

[1] Any decimal places that result from dividing by T_{DP} are omitted.

It is prohibited to exceed these limits. If these limits are violated, the module will generate a parameter assignment error and will not go into isochrone mode.

[2] not with FM 350-2

[3] for period measurement: 120000

The measured value is updated in the feedback interface and the STS_CMP1 or STS_COMP1 bit is set at the end of update time / integration time.

Measuring principle

The module counts each positive edge of a pulse and allocates it a time value in µs.

The dynamic measuring time is defined as the difference between two time values.

At a pulse sequence with one or several pulses per update interval:

Dynamic measuring time = Time value of last pulse in the current update time interval
minus
time value of the last pulse in the previous update interval

If no pulses are received within the next update intervals after the dynamic measuring time is calculated, the measuring time is extended by these update intervals. If the value "1 pulse per

dynamic measuring time" is less than the last measured value, it will be output as the new value. The figure below shows the measuring principle.

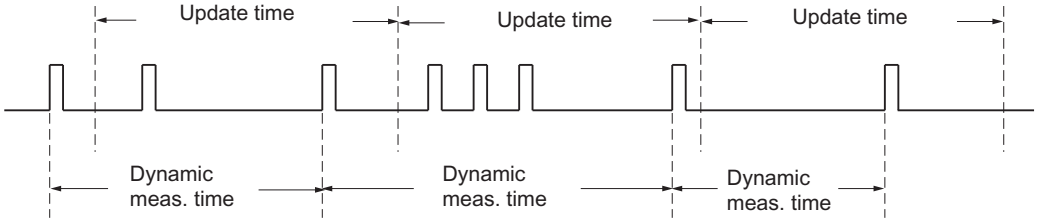

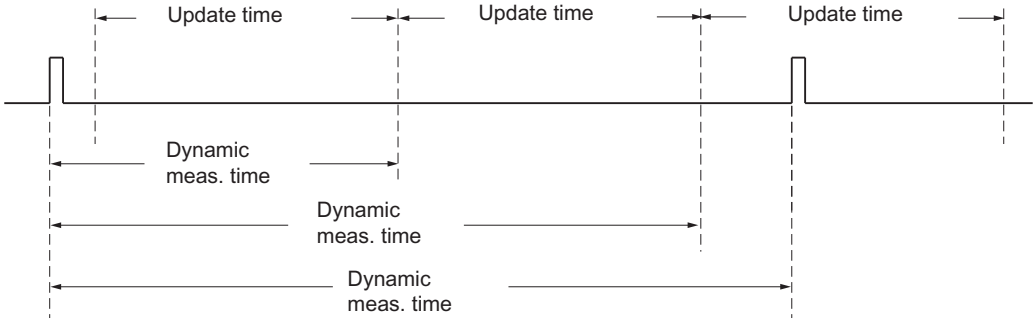

Measurement sequence

The module measures continuously. You define a specific update time in the parameter settings.

The module returns the value "-1" until the first update time has expired. The first update time starts when the gate has opened.

When the gate has opened, continuous measurement starts at the first pulse of the pulse sequence to measure. The first measured value can be calculated after the second pulse at the earliest.

On each expiration of the update time, a measured value is output at the checkback interface (frequency, period or rpm.) The end of measurement is reported at the STS_CMP1 or STS_COMP1 (for FM 350-2) status bits. This bit is reset by the RES_ZERO and STS_RES_ZERO or RES_STS and RES_STS_A bits by applying the complete acknowledgement principle.

If the direction of rotation is reversed during an update time, the measured value for this measurement period will be undefined. You can respond to any irregularities in the process by evaluating the feedback bits for direction.

The following figure illustrates the principle of continuous measurement taking frequency measurement as an example.

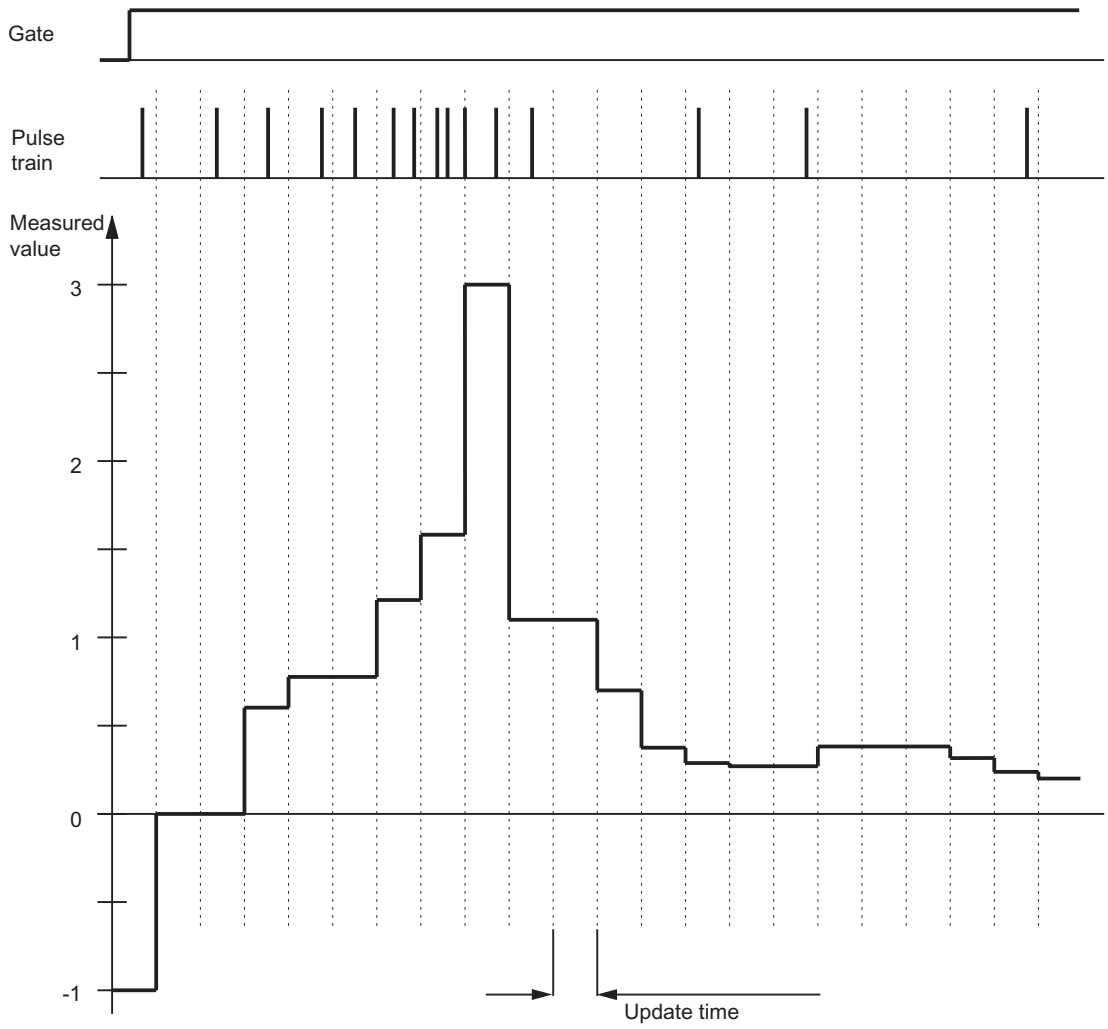

Limit Monitoring

The limit values must meet each of the following conditions:

- Low limit value ≥ low measuring range limit
- High limit value ≥ high measuring range limit
- High limit value ≥ (low limit value + 1)

Upon expiry of the update time, the module compares the measured value (frequency, speed or period) with the low and high limit values parameterized. The result of the comparison is as follows.

If the measured value...	...the following bit will be set in the status:
is greater than the high limit value	STS_OFLW
is smaller than the low limit value	STS_UFLW

A hardware interrupt may also be generated.

The image below shows the value of the status bits for limit monitoring in measuring modes.

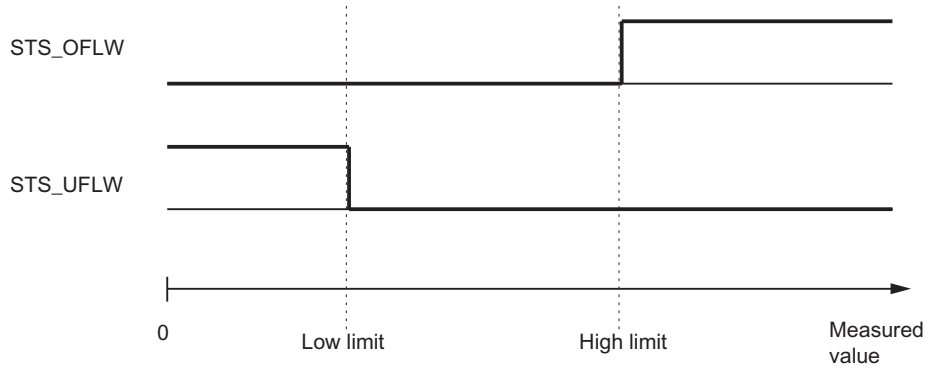

Reset the STS_OFLW and STS_UFLW bits by setting the RES_ZERO and STS_RES_ZERO or RES_STS bits by applying the complete acknowledgement principle. The module sets the status bits again if it detects high limit violated of the measured value after the acknowledgement bits were set.

If parameters are assigned accordingly, you also can use limit monitoring to set output DOn.

Gate control

You can use the hardware gate and software gate to control i.e. start and stop the module's measuring processes.

Commands in the measuring modes

You can apply commands to the module measuring process during operation:

Designation	Description
Open and close gate	You start the measurement by opening a gate, and stop it by closing this gate.

Isochronous mode

When operating in isochrone mode, the module accepts the control signals output by the control interface at time T_o in each PROFIBUS / PROFINET cycle. As a result, all control operations are executed in isochronous mode and take effect at the time T_o. The response to the control signals is reported within the same PROFIBUS / PROFINET cycle.

The module returns a measured value and the status bits at time T_i in each PROFIBUS / PROFINET cycle.

Each measurement starts and ends at the time T_i.

Note

As you define the update time as an integer multiple of 10 ms for operation in non-isochrone mode and in integer multiples of the bus cycle time for operation in isochrone mode, you should also always adapt the update time parameter when you toggle these modes if you wish to retain the actual update time.

Measuring modes (S7-300, S7-400)

Overview of measuring modes (S7-300, S7-400)

You select module functionality by setting a default operating mode. The table below gives an overview of measuring modes.

Operating mode	Description
Frequency measurement (Page 191)	The module counts the pulses it receives within a dynamic measuring time [1] and uses them to calculate the frequency.
Rotational speed measurement (Page 193)	The module counts pulses received from a tachometer generator within a dynamic measuring time [1] and calculates the velocity based on this value and the number of pulses per encoder revolution.
Period measurement (Page 194)	The module displays the dynamic measuring time [1] as a period. If the period is less than the update time, a mean value is calculated for the period.

[1] Dynamic measuring time means: if no 2 edges are detected by the measuring signal before the end of the parameterized measuring time, an estimated measured value will be calculated with the dynamic measuring time. It is assumed that a pulse was sent at the end of the update time. This gives you a constant measured value.

Note

FM 450-1 does not support any measuring modes.

Frequency measurement (S7-300, S7-400)

Description

In this operating mode, the module counts the pulses it receives within a dynamic measuring time or a specified integration time. The frequency is then calculated with the edge time stamps.

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 200)
- Hardware gate (Page 201), level-controlled or edge-controlled

The figure below shows an example of frequency measurement with gate control.

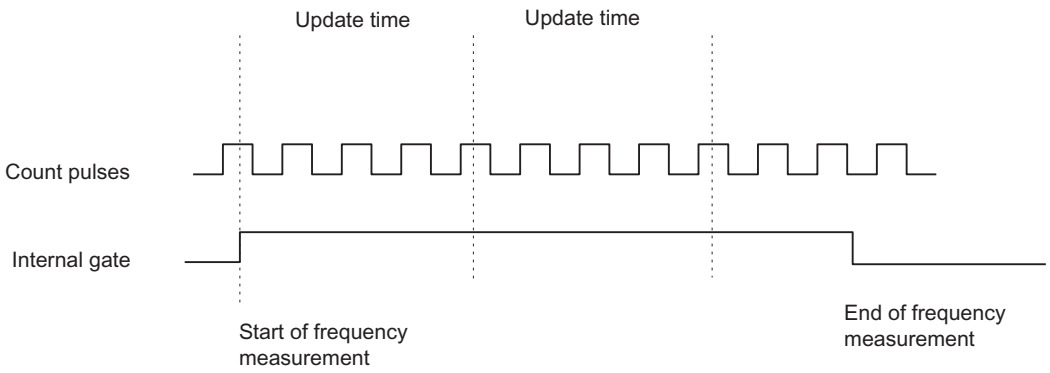

Possible Measuring Ranges with Error Indication

The table below sets out the measuring ranges and absolute errors for continuous frequency measurements.

Frequency f	Absolute error	
	FM 350-1 ET 200S 1Count5V	ET 200S 1Count24V
0.1 Hz	± 0.001 Hz	
1 Hz	± 0.001 Hz	
10 Hz	± 0.003 Hz	
100 Hz	± 0.02 Hz	
1,000 Hz	± 0.18 Hz	
10,000 Hz	± 1.8 Hz	
100,000 Hz	± 18 Hz	
500,000 Hz	± 90 Hz	–

The table below sets out the measuring ranges and absolute errors for continuous frequency measurements with integration time.

Integration time	f_{min} ± absolute error	f_{max} ± absolute error	
		ET 200S 1Count24V	ET 200S 1Count5V
10 s	0.1 Hz ± 0.001 Hz	100,000 Hz ± 18 Hz	500,000 Hz ± 90 Hz
1 s	1 Hz ± 0.001 Hz	100,000 Hz ± 11 Hz	500,000 Hz ± 55 Hz
0.1 s	10 Hz ± 0.002 Hz	100,000 Hz ± 10 Hz	500,000 Hz ± 52 Hz
0.01 s	100 Hz ± 0.013 Hz	100,000 Hz ± 13 Hz	500,000 Hz ± 63 Hz

Values that can be changed during operation

The following values can be changed during operation:

- Low limit
- High limit
- Update time (measuring time) [1)]
- Function of the digital output [1)]

[1] not with FM 350-2

Result

The end of a frequency measurement is reported at the STS_CMP1 or STS_COMP1 (for FM 350-2) status bit. The value of the calculated frequency is displayed in units of 10^{-3} Hz. You can read the frequency value measured from the feedback interface.

Rotational speed measurement (S7-300, S7-400)

Description

In this operating mode, the module counts the pulses it receives from a tachometer generator within a dynamic measuring time or a specified integration time and measures the time interval between the edges. On this basis it then calculates the velocity with the pulses per encoder revolution.

For "Rotational speed measurement" mode, you must also assign the pulses per encoder revolution.

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 200)
- Hardware gate (Page 201), level-controlled or edge-controlled

The figure below shows an example of rotational speed measurement with gate control.

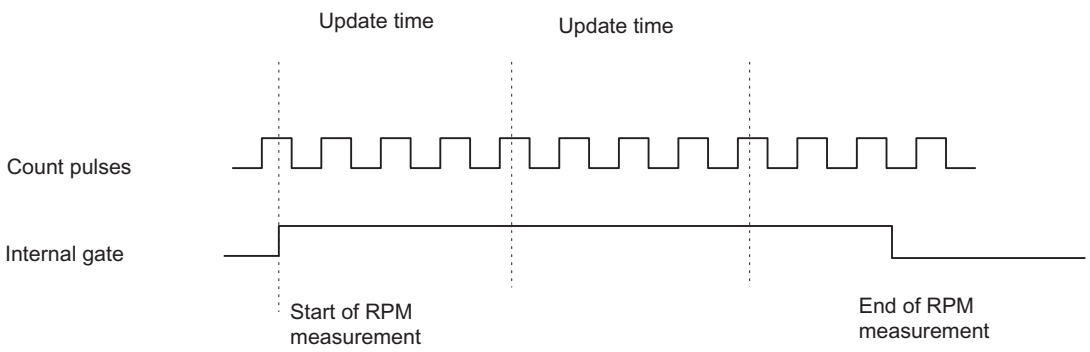

Possible measuring ranges with error indication

The table below sets out the measuring ranges and absolute errors for continuous rotational speed measurement when the number of pulses per encoder rotation = 60.

Speed n	Absolute error
1 /min	± 0.04 /min
10 /min	± 0.04 /min

Speed n	Absolute error
100 /min	± 0.05 /min
1,000 /min	± 0.21 /min
10,000 /min	± 1.82 /min
25,000 /min	± 4.50 /min

The table below sets out the measuring ranges and absolute errors for rotational speed measurement with integration time when the number of pulses per encoder rotation = 60.

Integration time	n_{min} ± absolute error	n_{max} ± absolute error
10 s	1 /min ± 0.03 /min	25000 /min ± 4.5 /min
1 s	1 /min ± 0.03 /min	25000 /min ± 2.8 /min
0.1 s	10 /min ± 0.03 /min	25000 /min ± 2.6 /min
0.01 s	100 /min ± 0.04 /min	25000 /min ± 3.2 /min

Values that can be changed during operation

The following values can be changed during operation:

- Low limit
- High limit
- Update time (measuring time) [1)]
- Function of the digital output [1)]

[1)] not with FM 350-2

Result

The end of a rotational speed measurement is reported at the STS_CMP1 or STS_COMP1 (for FM 350-2) status bit. The value of the velocity calculated is displayed in 10^{-3} /min. You can read the velocity measured from the feedback interface.

Period measurement (S7-300, S7-400)

Description

In this operating mode, the module indicates the dynamic measuring time as a period. If the period is less than the update time, a mean value is calculated for the period.

In "Period measurement with integration time" mode, 1Count24V measures the time between two positive edges of the count signal by counting the pulses of an internal quartz-accurate reference frequency (16 MHz) within a specified integration time.

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 200)
- Hardware gate (Page 201), level-controlled or edge-controlled

The figure below shows an example of period measurement with gate control.

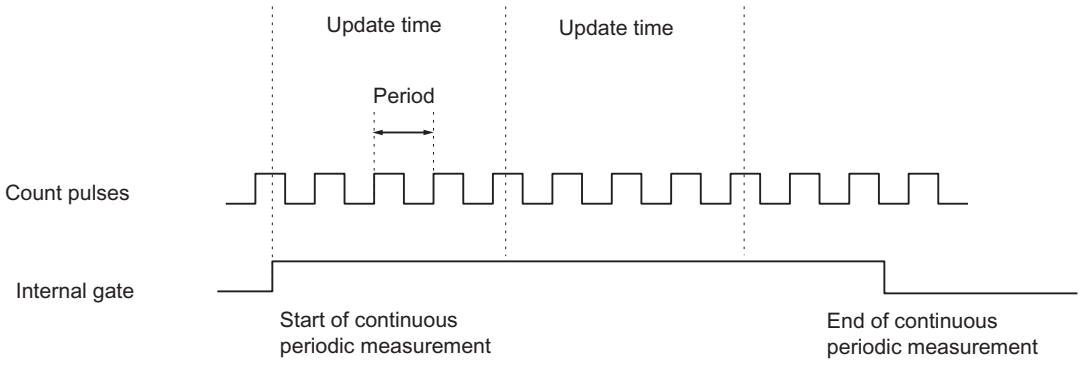

Possible measuring ranges with error indication

Continuous period measurement:

- The absolute error for continuous period measurement with a **resolution of 1 µs** is equal to 0.1‰ of the measured value.
- The table below sets out the measuring ranges and absolute error for continuous period measurements with a **resolution of 1/16 µs**.

Period T_{min} ± Absolute error
1/16 µs x (160 ± 1)
1/16 µs x (1,600 ± 1)
1/16 µs x (16,000 ± 3)
1/16 µs x (160,000 ± 20)
1/16 µs × (1,600,000 ± 160)
1/16 µs × (16,000,000 ± 1,600)
1/16 µs × (160,000,000 ± 16,000)
1/16 µs × (1,600,000,000 ± 160,000)

Period measurements with integration time:

- The table below sets out the measuring ranges and absolute error for period measurements with integration time with a **resolution of 1 µs**.

Integration time	T_{min} ± absolute error	T ± absolute error
100 s	1 µs × (10 ± 0)	1 µs × (100,000,000 ± 10,000)
10 s	1 µs × (10 ± 0)	1 µs × (10,000,000 ± 1,000)
1 s	1 µs × (10 ± 0)	1 µs × (1,000,000 ± 100)
0.1 s	1 µs × (10 ± 0)	1 µs × (100,000 ± 10)
0.01 s	1 µs × (10 ± 0)	1 µs × (10,000 ± 1)

- The table below sets out the measuring ranges and absolute error for period measurements with integration time with a **resolution of 1/16 µs**.

Integration time	T_{min} ± absolute error	T ± absolute error
100 s	1/16 µs × (160 ± 0)	1/16 µs × (1,600,000,000 ± 160,000)
10 s	1/16 µs × (160 ± 0)	1/16 µs × (160,000,000 ± 16,000)
1 s	1/16 µs × (160 ± 0)	1/16 µs × (16,000,000 ± 1,600)
0.1 s	1/16 µs × (160 ± 0)	1/16 µs × (1,600,000 ± 160)
0.01 s	1/16 µs × (160 ± 0)	1/16 µs × (160,000 ± 16)

Values that can be changed during operation

The following values can be changed during operation:

- Low limit
- High limit
- Update time (measuring time) [1)]
- Function of the digital output [1)]

[1)] not with FM 350-2

Result

The end of a period measurement is reported at the STS_CMP1 or STS_COMP1 (for FM 350-2) status bit. The value of the calculated period is displayed in 1 µs or 1/16 µs. You can read the period measured from the feedback interface.

Dosing (only for FM 350-2) (S7-300, S7-400)

Description

In this mode, four counter channels are grouped together on the FM 350-2 to form one dosing channel. When the gate is open, FM 350-2 counts once in the main count direction:

- In the main count direction "up" between 0 and the parameterized high count limit.
  If you set the main count direction as "up," the start value is 0 and you specify the end value.
- In the main count direction "down" between the parameterized high count limit and 0.
  If you set the main count direction as "down," you specify the start value and the end value is 0.

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 200)
- Software gate (Page 200) and HW gate (Page 201)
  You can add a hardware gate to the software gate. The two gates act together like a logical AND operation, meaning the FM 350-2 only counts when both gates are open.

Behavior at the Count Limits, Software Gate

Main count direction up: If the counter receives a further count pulse after reaching the "High count limit - 1", the module sets the counter to 0, closes the internal gate and terminates the count even if the SW_GATE0, 4 bit is still set. The relevant status bit STS_OFLW0, 4 is set. The high count limit itself is therefore never reached.

Main count direction down: If the counter receives a further count pulse after reaching the value "1", the module sets the counter to the high count limit, closes the internal gate and terminates the count even if the SW_GATE0, 4 bit is still set. The relevant status bit STS_UFLW0, 4 is set. The value "0" is therefore never reached.

If you want to start the counter again, you must reset the SW_GATE0, 4 bit and then set it again.

Behavior at the Count Limits, Hardware Gate

Main count direction up: If the counter receives a further count pulse after reaching the "High count limit - 1", the module sets the counter to 0, closes the internal gate and terminates the count even if the SW_GATE0, 4 bit and the I0, I4 input are still set. The high count limit itself is therefore never reached. The relevant status bit STS_OFLW0, 4 is set.

Main count direction down: If the counter receives a further count pulse after reaching the value "1", the module sets the counter to the high count limit, closes the internal gate and terminates the count even if the SW_GATE0, 4 bit and the I0, I4 input are still set. The value "0" is therefore never reached. The relevant status bit STS_UFLW0, 4 is set.

If you want to start the counter again, you must reset the I0, I4 input and then set it again. You can only start a new count with the hardware gate.

Comparison values

You can configure four comparison values within the parameterized counting range for each dosing channel of the FM 350-2. You can also configure whether and under what conditions a digital output should be set and/or a hardware interrupt triggered in conjunction with a comparison value. You can set the following conditions for this:

- A hardware interrupt is triggered if the current counter value matches a comparison value.
- A digital output is set if the current counter value is greater than or equal to a comparison value.
- A digital output is set if the current counter value is less than or equal to a comparison value.

A digital output is only set if you have enabled the corresponding output with the CTRL_DQ0…7 bit.

Note

Outputs can only be set while the CPU is in RUN mode if the gate is open.

The figure below shows an example of dosing in the main count direction "down" with comparison values.

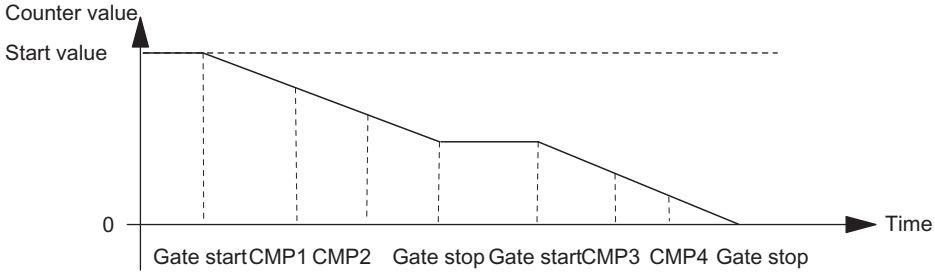

Load Value in Preparation

You can define a load value within the parameterized counting range while the CPU is in RUN mode. This value is used by the counter as the new start value every time after the following events:

- When the high count limit is reached for main count direction up.
- Reaching 0 when counting down
- Canceling of the count process by a software gate or a hardware gate (when the count process is interrupted the load value is not used)

The load value is then the new initial value from which the next and all subsequent dosing processes begin. The set output and interrupt behavior remains the same.

Load Value Directly

You can change the current counter value while the CPU is in RUN mode. The new count value is used by the counter directly as the dosing value.

Value Range for Load Values

The value range for load values depends on the set main count direction. The range is:

- 0 to high count limit - 2 in main count direction "up"
- High count limit to 2 in main count direction down

Gate control (S7-300, S7-400)

Gate functions (S7-300, S7-400)

Counting with Gate Functions

Many applications require that the count be started or stopped at a defined time depending on other events. The counting process in counter modules is started and stopped like this using a gate function. When the gate opens, the counter receives count pulses and the count starts. When the gate is closed, the counter can no longer receive count pulses and the count stops.

Software Gate and Hardware Gate

The counter modules have two gates:

- A software gate (Page 200) which is controlled by the input parameter / control bit SW_GATE.
- A hardware gate (Page 201) which is controlled by signals at the module's digital inputs.

Please also see the information on gate function in isochrone mode (Page 202).

Internal Gate

The internal gate is the logic AND operation combining a hardware gate and a software gate. The status at the internal gate is indicated by the STS_GATE bit.

If no hardware gate was assigned, only the setting of the software gate is relevant. The count process is activated, interrupted, resumed, and canceled via the internal gate. The internal gate can also be closed by events dependent on the counter value in the operating modes "Count once" and "Dosing".

Hardware gate	Software gate	Internal gate	Count process
open	open	open	active
open	closed	closed	inactive
closed	open	closed	inactive
closed	closed	closed	inactive

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- When cancelled (Page 202), the count restarts at its initial value after gate stop and gate start.
- When interrupted (Page 202), the count is resumed from the last current count value following gate stop and gate start.

Example

The gate is opened and the count pulses are counted by setting the gate signal. If the gate signal is taken away, the gate will be closed and the count pulses no longer detected by the counter. The count value remains constant.

The diagram below shows the opening and closing of a gate and pulse counting.

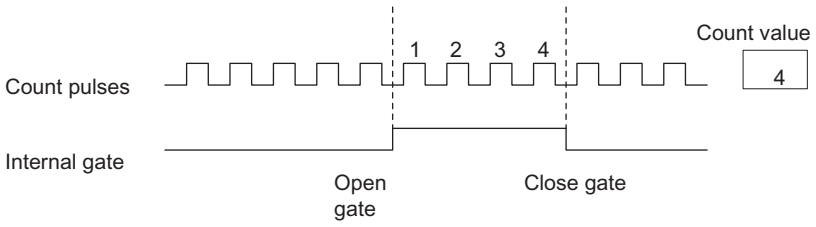

Stopping the count using the gate stop function

You can also stop the count in the FM x50-1 modules by setting the gate stop function, regardless of the signals set at or status of the software gate. You do this by setting the GATE_STP input parameter.

When you reset this parameter, you can only reopen the gate by setting a positive edge at the digital input "DI start" (hardware gate) or by setting the input parameter SW_GATE (software gate) again.

Software gate (S7-300, S7-400)

You open and close the software gate with the input parameter / control bit SW_GATE.

Opening and closing the software gate

The image below shows how the software gate is opened and closed. No hardware gates should be configured.

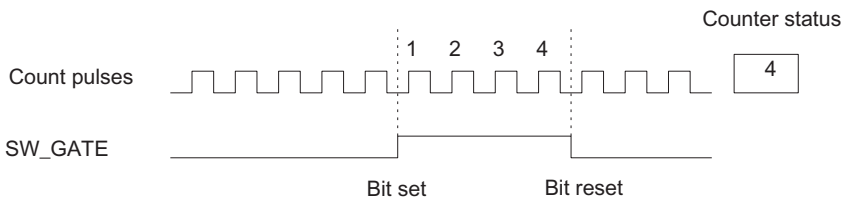

The software gate is:

- opened by setting the input parameter / control bit SW_GATE.
- closed by resetting the input parameter / control bit SW_GATE.

You can re-open the closed gate by resetting input parameter SW_GATE.

Hardware gate (S7-300, S7-400)

You open and close the hardware gate by applying the relevant signals to or removing the signals from the digital inputs.

The software gate must be open.

Level-triggered opening and closing of the HW gate

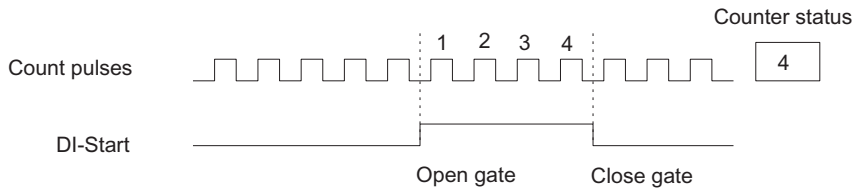

Count pulses will be received and counted by the counter as long as the digital input is set. Resetting the digital input closes the gate. The counter stops and ignores any further count pulses.

If the gate is closed as a result of high limit violated or low limit violated, you open it again by a reset > set cycle at the digital input.

The level-controlled hardware gate is activated by the first positive edge at the digital input after you assign parameters.

ET 200S 1Count24Vand1Count5V:

1Count24V / 1Count5V has a hardware gate, which is controlled by the digital input on the module. You configure the hardware gate as a function of the digital input (Function DI "Hardware Gate"). The gate is opened in response to a positive edge on the digital input and closed in response to a negative edge.

Edge-triggered opening and closing of the hardware gate

Edge-controlled opening and closing of the hardware gate is only possible for FM x50-1.

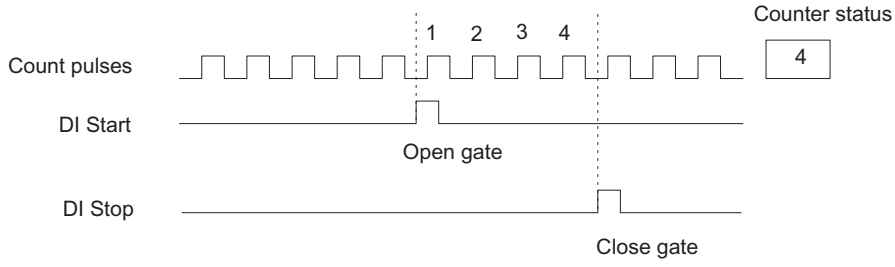

An edge-controlled hardware gate is opened by a positive edge at digital input "DI start". The gate is closed by a positive edge at digital input "DI stop".

If a positive edges is set in parallel at both inputs, an open gate will be closed, whereas a closed gate remains closed. A positive edge at digital input "DI start" cannot open the gate if digital input "DI stop" is set.

Cancel and interrupt function of the gate (S7-300, S7-400)

In your gate function parameters, you can define whether the gate should cancel or interrupt the count.

Cancel gate function

When the "cancel" gate function is active, the count stops when the gate closes and is restarted at the load value when the gate is restarted (time ① in the figure below).

The figure below shows an example of continuous counting down with the "cancel" gate function.

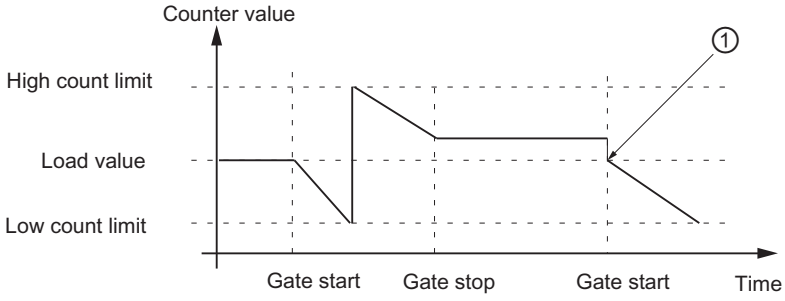

"Interrupt" gate function

When the "interrupt" gate function is active, the count stops when the gate closes, and resumes at the last current counter value when the gate is restarted (time ① in the figure below).

The figure below shows an example of continuous counting up with the "interrupt" gate function.

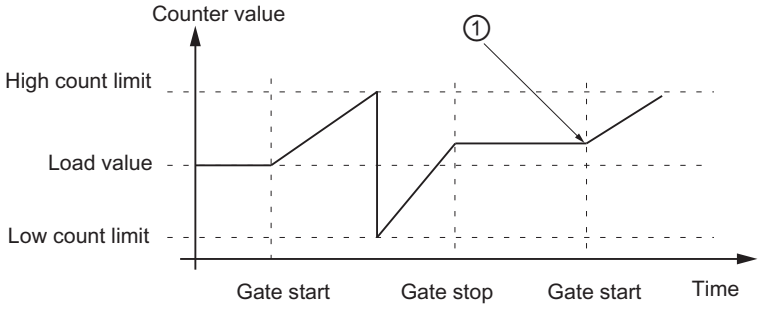

Gate control in isochronous mode (S7-300, S7-400)

Gate control operates as follows in isochrone mode:

Software gate control

The software gate is controlled by setting and resetting the SW_GATE control bit. After the control bit has changed, the count starts and ends at time T_o in the next PROFIBUS DP cycle:

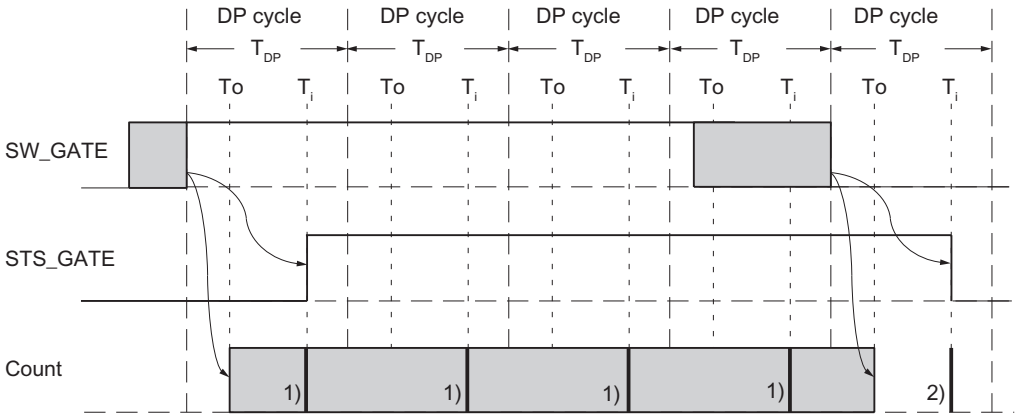

1) Provision of actual count value
2) Provision of the count value that was valid at the end of the count

Hardware gate control

In hardware gate control mode, the count starts or stops instantaneously when the hardware gate opens or closes:

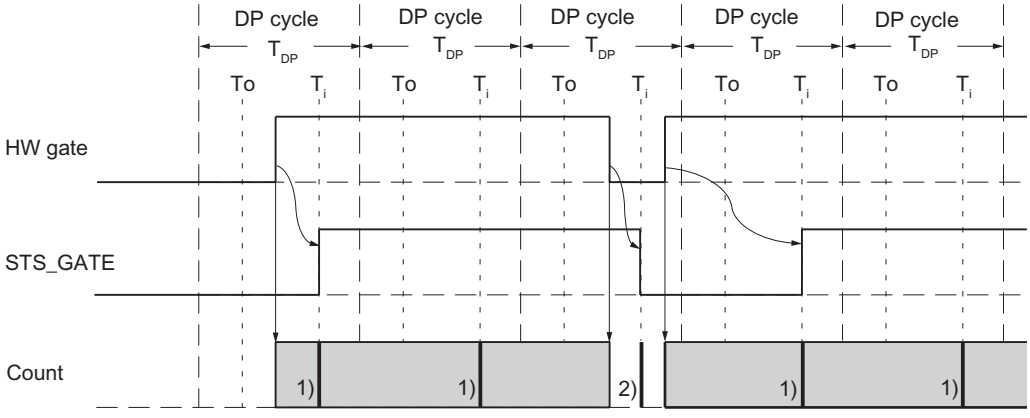

1) Provision of actual count value
2) Provision of the count value that was valid at the end of the count

Encoder signals (S7-300, S7-400)

Encoders which can be connected (S7-300, S7-400)

The counter module can process square wave signals generated by incremental encoders or pulse generators.

- **Incremental encoders** scan a grating to generate square wave electrical pulses. They differ in terms of pulse amplitude and number of signals. Incremental encoders output two tracks out of phase by 90°.
- **Pulse generators** such as light barriers and initiators (BEROs) only supply a square wave signal with a specific voltage level.

5 V differential signal (S7-300, S7-400)

5 V incremental encoder count signals

The 5 V incremental encoder sends the following RS 422 differential signal to the module:

- A and /A
- B and /B
- N and /N

The signals /A, /B and /N are the inverted signals of A, B and N. Signals A and B are phase-shifted by 90°.

5 V incremental encoders use tracks A and B for counting. Track N is used to initialize the counter with the load value, if programmed accordingly.

Encoders which return inverted signals for each signal sent are symmetrical encoders.

The diagram below shows the time profile of the encoder signals of a 5 V incremental encoder.

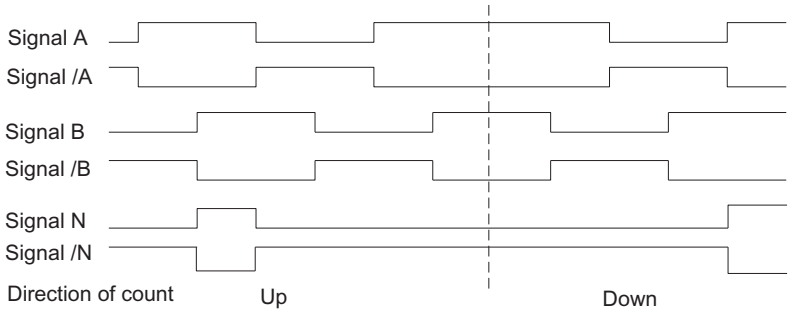

The module detects the count direction by evaluating the ratio of signals A and B.

Changing the count direction

You can change the count direction using the "Count direction normal" and "Count direction inverted" parameters without having to modify the wiring.

Monitoring encoder signals

The module monitors the cable connection, and detects wire-break or short-circuit.

You can define which of the three signal pairs to include in monitoring in your program. There is no need to wire any unused signal pairs, if you have disabled the corresponding diagnostics functions in the program (monitoring.)

An error at all three signals indicates a defective encoder, a short circuit in the encoder supply or that no encoder is connected.

When programming is completed, and the module detects an error, the error information will be written to the diagnostics data records DS0 and DS1. This situation my lead to a diagnostics interrupt if programmed accordingly.

24 V signals (S7-300, S7-400)

24 V incremental encoder count signals

The 24 V incremental encoder returns the 24 V signals A*, B* and N* to the module. The A* and B* signals are phase-shifted by 90°.

24-V signals are marked with an asterisk "*" character.

24 V incremental encoders use tracks A* and B* for counting. Track N* is used to initialize the counter with the load value, if programmed accordingly.

Encoders which do not return inverted signals asymmetrical encoders.

The diagram below shows the time profile of the encoder signals of a 24 V incremental encoder.

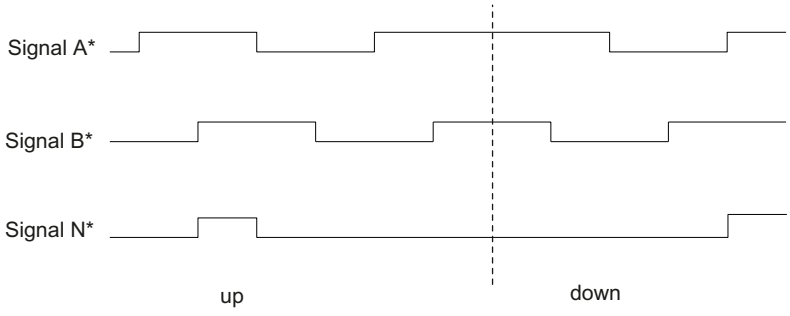

The module detects the count direction by evaluating the ratio of signals A* and B*.

You can configure the inputs of 24 V encoder signals for the connection of Sourcing output/ push pull or Sinking output to the counter. For further information, refer to the encoder manual.

Changing the count direction

You can change the count direction using the "Count direction normal" and "Count direction inverted" parameters without having to modify the wiring.

Count signals from 24 V pulse generators without/with direction level

Encoders such as proximity switches (BERO) or light barriers return only a count signal which you wire to terminal A* of the front connector.

in additional, you can wire a signal for direction detection to terminal B* of the relevant counter. If your encoder does not return a corresponding signal, you can wire a corresponding ID signal you generate within the user program, or use a corresponding process signal.

The diagram below shows the time profile of the encoder signals of a 24 V pulse generator with direction level and the resultant count pulses.

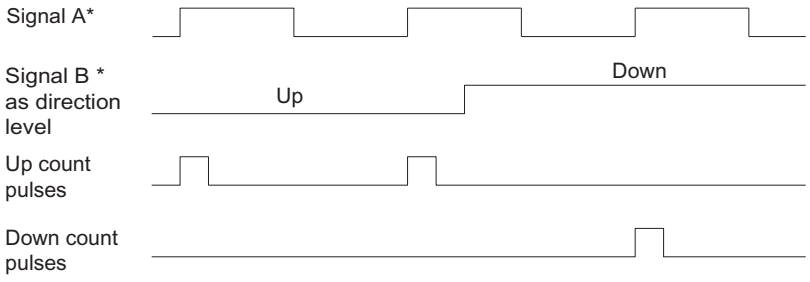

Monitoring encoder signals

The 24 V count signals are not monitored to detect wire breaks or short circuits.

Signal evaluation (S7-300, S7-400)

Overview (S7-300, S7-400)

The counter module counter counts the signal edges. It usually evaluates the edge at A (A*) (single evaluation). Options in the program of increasing the resolution:

- Single evaluation (Page 206)
- Double evaluation (Page 207)
- Quadruple evaluation (Page 207)

Multiple evaluation is supported only:

- for incremental 5 V encoders with signals A and B out of phase by 90°.
- for incremental 24 V encoders with signals A* and B* out of phase by 90°.

Default setting

The default setting is "Single evaluation".

Single evaluation (S7-300, S7-400)

In this mode, the module evaluates only one edge of signal A.

- "Up" count pulses are recorded at the negative edge at A and a low level at B.
- Down count pulses are recorded if the edge at A falls or the level at B is low.

The diagram below illustrates the single evaluation of the signals.

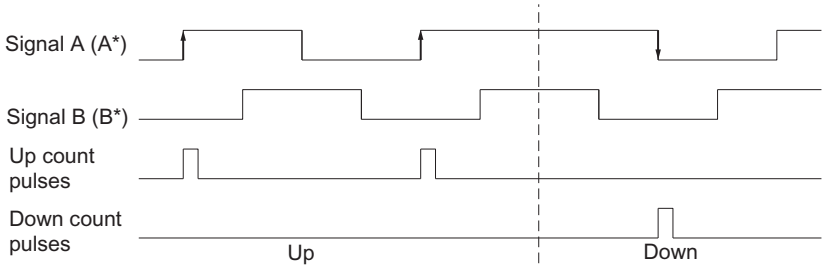

Double evaluation (S7-300, S7-400)

Double evaluation refers to the evaluation of the positive and negative edges of signal A. The level of signal B determines whether up or down count pulses are generated.

The diagram below shows double evaluation of signals.

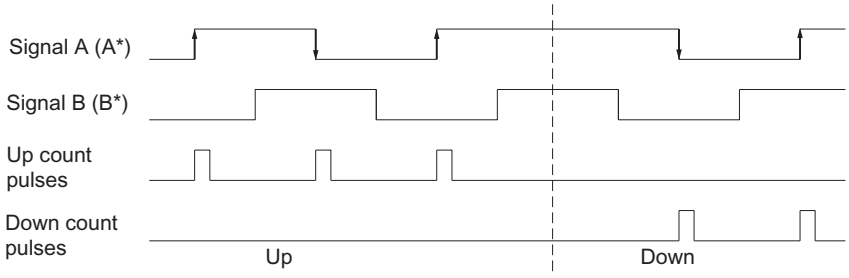

Quadruple evaluation (S7-300, S7-400)

Quadruple evaluation refers to the evaluation of the positive and negative edges of signals A and B. The level of signals A and B determines whether up or down count pulses are generated.

The diagram below illustrates the quadruple evaluation of signals.

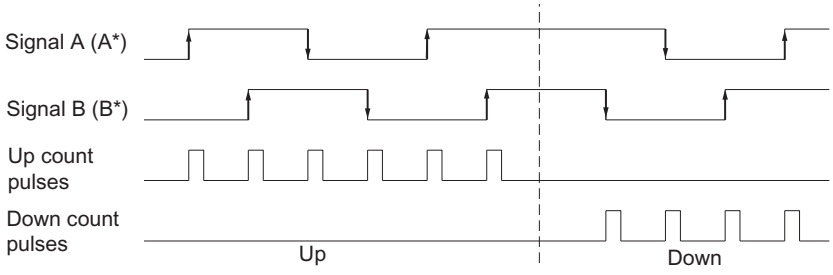

1.2.1.2 Using FM 350-1 (S7-300, S7-400)

FM 350-1 front connector terminal assignment (S7-300, S7-400)

Front connector

You connect the following to the 20-pin front connector:

- Count signals
- Digital inputs
- Digital outputs
- Encoder power supply
- Auxiliary voltage and load voltage

The figure below shows the front of the module, the front connector and the inside of the front panel with terminal assignment.

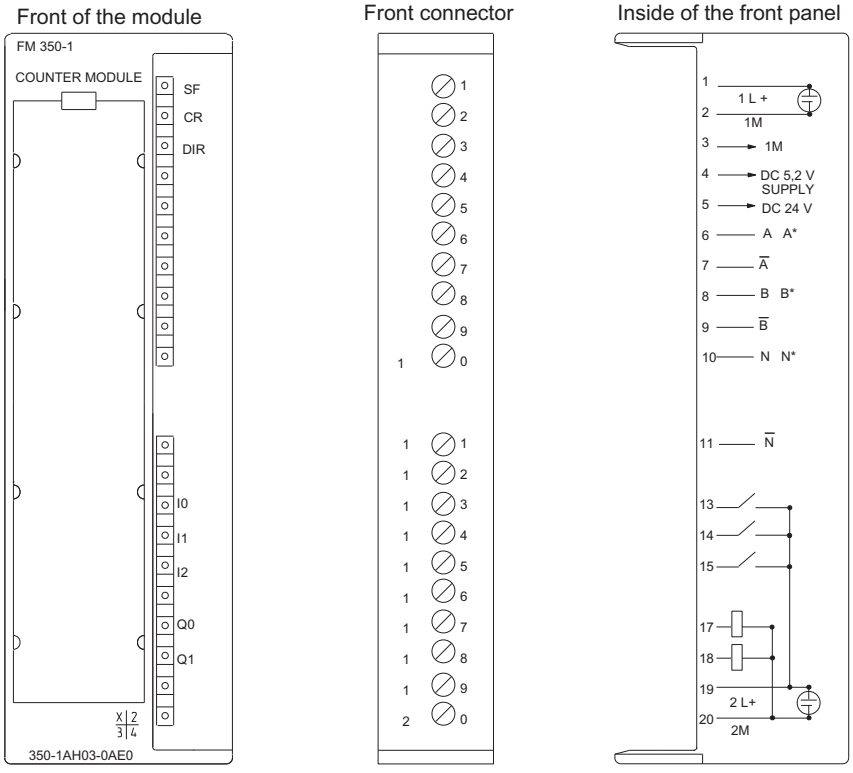

Pin assignments of the front connector

The following table shows the terminal assignment for the front connector of the FM 350-1.

Connection	Name	Input/output	Function			
Auxiliary voltage						
1	1L+	Input	24 V auxiliary voltage			
2	1M	Input	Auxiliary voltage ground			
Encoder supply and count signals			**5 V encoder RS 422, symmetrical**	**24 V encoder, asymmetrical**	**24 V pulse generator with direction level**	**24 V initiator**
3	1M	Output	Encoder supply ground			
4	5.2 V DC	Output	Encoder power supply 5.2 V			
5	24 V DC	Output	Encoder power supply 24 V			
6	A A*	Input	Encoder signal A	Encoder signal A *		
7	/A	Input	Encoder signal /A	—		
8	B B*	Input	Encoder signal B	Encoder signal B *	Directional signal	—
9	/B	Input	Encoder signal /B	—		
10	N N*	Input	Encoder signal N	Encoder signal N *	—	
11	/N	Input	Encoder signal /N	—		
12	—	—	—			
Digital Inputs and Digital Outputs						
13	I0	Input	Digital input DI Start			
14	I1	Input	Digital input DI Stop			
15	I2	Input	Digital input DI Set			
16	—	—	—			
17	Q0	Output	Digital output DO0			
18	Q1	Output	Digital output DO1			
Load voltage						
19	2L+	Input	24 V load voltage			
20	2M	Input	Load voltage ground for the digital inputs and outputs			

Note

The circuits for the counter inputs (encoder supply, encoder signals) are non-isolated from the ground of the CPU. You should therefore connect terminal 2 (1M) to the CPU ground using a low-resistance connection. The lack of such a connection can result in malfunction or a defect in the FM 350-1.

If you supply the encoders externally, you must also connect the ground of this external voltage with the ground of the CPU.

You will find more information on construction and wiring in the manual (http://support.automation.siemens.com/WW/view/en/1086726) for the FM 350-1 counter module.

Auxiliary voltage 1L+/1M

Connect a direct voltage of 24 V to the 1L+ and 1M terminals for the 5 V and 24 V encoder voltage supply.

An integrated diode protects the module from a reversal in the polarity of the auxiliary voltage.

The module monitors the connection of the auxiliary voltage.

5.2V DC encoder power supply

The module generates a voltage of 5.2 V from the auxiliary voltage 1L+/1M at a maximum current of 300 mA, supplied at the "DC5.2V" connection for the short circuit proof supply of a 5 V encoder. The encoder supply is monitored for short-circuit.

24 V DC encoder supply

1L+/1M is provided at output "DC24V" for 24V short-circuit proof encoder voltage supply. The encoder supply is monitored for short-circuit.

5 V encoder signals A and /A, B and /B, N and /N

You can connect incremental encoders with 5 V differential signal to the front connector in accordance with RS 422, i.e. incremental encoders with the differential signal A and /A, B and / B, N and /N.

The signals A and /A, B and /B, N and /N are connected via the correspondingly labeled terminals.

The signals N and /N are only connected if you want to set the counter to the zero mark of the encoder.

The inputs are not isolated from the S7-300 bus.

24 V encoder signals A*, B* and N*

24 V signals are identified with the letters A*, B* and N*.

You can connect three different encoder types to each counter:

- Incremental encoder with 24 V signals:
  The signals A*, B* and N* are connected via the correspondingly labeled terminals.
- Pulse encoders without direction level:
  The signal is connected to terminal A*.
- Pulse encoders with direction level:
  The count signal is connected to terminal A*. The direction level is connected to terminal B*.

The inputs are not isolated from the S7-300 bus.

Input filter for 24 V encoder signals

To suppress faults, you can configure input filters (RC elements) with a uniform filtering time for the 24 V inputs A*, B* and N*. The following two input filters are available.

Characteristics	Input Filter 1 (default)	Input Filter 2
Typical input delay	1 µs	15 µs
Maximum count frequency	200 kHz	20 kHz
Minimum pulse width of the count signals	2.5 µs	25 µs

Digital Inputs DI Start, DI Stop and DI Set

You can use digital inputs DI Start and DI Stop for the gate control of the counter. The gate control can be either level-controlled or edge-controlled.

Digital input DI Set is used to set the counter to the load value.

The digital inputs are operated with a nominal voltage of 24 V.

The digital inputs are isolated from the bus of the S7-300 and the counter inputs.

Input Filters for Digital Inputs

For the purpose of suppressing interference, you can parameterize input filters DI-Start, DI-Stop and DI-Set (RC elements) with a uniform filter time for digital inputs. The following two input filters are available:

Characteristics	Input Filter 1 (default)	Input Filter 2
Typical input delay	1 µs	15 µs
Minimum pulse width of the input signals	2.5 µs	25 µs

Digital Outputs DO0 and DO1

The FM 350-1 has two digital outputs, DO0 and DO1, for directly triggering control processes.

The digital outputs are supplied via the load voltage 2L+.

The digital outputs are isolated from the bus of the S7-300 and the counter inputs.

The digital outputs are Sourcing outputs and can carry load current of 0.5 A. They are protected from overload and short-circuit.

Note

Relays and contactors can be connected directly without external circuitry.

The time response of the digital outputs depends on parameter assignment.

2L+/2M load voltage

For supplying digital outputs DO0 and DO1, a load voltage of 24 V must be supplied to the module via terminals 2L+ and 2M.

An integrated diode protects the module from polarity reversal of the load voltage.

The load voltage 2L+/2M is not monitored by the FM 350-1.

Configuring FM 350-1 (S7-300, S7-400)

Configuring FM 350-1 (S7-300, S7-400)

Operating modes (S7-300, S7-400)

Selecting an operating mode (S7-300, S7-400)

Description

You define counter module functionality by setting an operating mode.

Operating mode	Description
Count continuously (Page 182) (with or without gate)	The FM 350-1 counts continuously from the current counter value.
Count once (Page 183) (with hardware or software gate)	When the gate opens, the FM 350-1 counts from the load value to the count limit.
Count periodically (Page 184) (with hardware or software gate)	When the gate opens, the FM 350-1 counts from the load value between the count limits.
Frequency measurement (Page 191) (with hardware or software gate)	The FM 350-1 counts the pulses that occur during a dynamic measuring time.
Rotational speed measurement (Page 193) (with hardware or software gate)	The FM 350-1 counts pulses received from a tachometer generator within a dynamic measuring time and calculates the velocity based on this value and the number of pulses per encoder revolution.
Period measurement (Page 194) (with hardware or software gate)	The FM 350-1 indicates the dynamic measuring time as a period. The mean value of the period is calculated if its duration is less than the update time.

The default setting is the "Count continuously" operating mode.

Specifying the high count limit (S7-300, S7-400)

Description

By configuring an high count limit together with a main count direction (Page 218) (up or down), you can reduce the maximum counting range. The parameterized counting range is then between 0 and the high count limit parameterized.

The high count limit can be any value between > 1 and the high count range limit.

Specifying measuring time (S7-300, S7-400)

Description

If you have selected a measuring mode, the module will update the measured values on a cyclic basis. The update time is the product of the measuring time x 10ms. You set the measuring time as the factor n.

Detailed descriptions of the value ranges for the measuring time can be found under "Measuring basics (Page 187)".

Selecting gate control (S7-300, S7-400)

Description

You define the count process in the selected operating mode by setting a gate control.

- without gate
  All gates are open.
- Software gate (Page 200)
  The software gate is opened and closed by the input parameter / control bit SW_GATE.
- Hardware gate (Page 201), level-controlled or edge-controlled
  The hardware gate is opened and closed by the corresponding signals at the digital inputs.
- Latching (Page 213)
  You can save (latch) the counter values by generating edges at the digital input "DI start".
- Latch/Retrigger (Page 215)
  You can save (latch) the counter values by generating edges at the digital input "DI start". The counter is set to the load value each time the values are saved.

The default setting is "Without gate".

Latching (S7-300, S7-400)

Description

If you select "Latch" for the gate control, you can save (latch) the counter values with edges at the "DI start" digital input. This does not change the counter value.

Requirements

You must use the software gate (Page 200) and select the edge for latching (Page 218).

The minimum interval between latch edges is 1 ms. Values can be lost if the time interval between edges is less than this value.

Function principle

The counter and latch values are assigned an initial value. The initial value is 0 for operating modes without a main count direction and for main count direction up. For main count direction down, the initial value is the parameterized high count limit.

The counting function is started when the software gate is opened. The counter starts at the load value.

The latch value is always exactly equivalent to the counter value at the time the edge trigger is generated.

The "DI start" value is indicated in the DB at status bit STS_STA.

The latch value is indicated by LATCH_LOAD in the DB.

The diagram below shows an example of "Latching" when load value = 0 and there is a positive edge at "DI start".

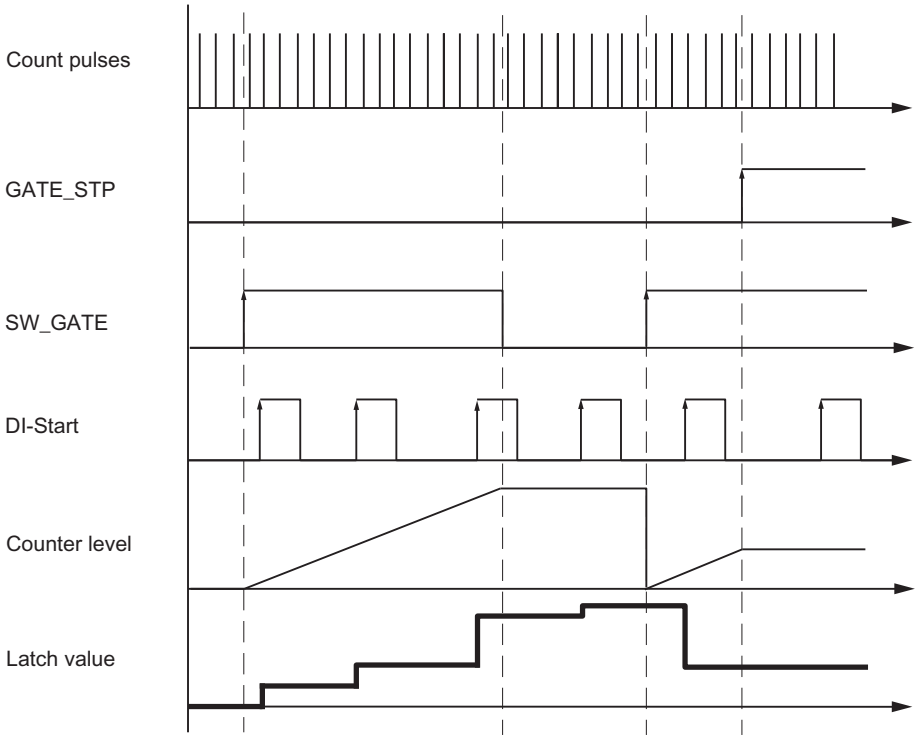

Interrupting and exiting

Closing the software gate by resetting SW_GATE will cancel counting.
The unchanged counter value can be resaved with edges at "DI start" even if the software gate is closed.

However, the count will be cancelled if you close the software gate by executing GATE_STP of instruction CNT_CTL1. "DI start" can in this case no longer be used to save any counter values.

Hardware interrupt upon "Latching"

A hardware interrupt can be triggered each time a counter value is saved by "Latching". As a consequence, you may need to increase the interval between the edges. The hardware interrupts will be lost if the interrupt rate is higher than the system acknowledgement rate. This situation is reported by a diagnostic interrupt.

Latch/Retrigger (S7-300, S7-400)

Description

If you select "Latch/Retrigger" gate control, you can save (latch) the counter values with edges at the "DI start" digital input. After each latching operation, the counter will be initialized with the load value, and resumes the count starting at the load value (retrigger.)

Requirements

You must use the software gate (Page 200) and select the edge for latching (Page 218) for "Latch/Retrigger".

The minimum interval between latch edges is 1 ms. Values can be lost if the time interval between edges is less than this value.

Function principle

The counter function is enabled by opening the SW gate.

The counter and latch values are assigned an initial value. These values do not change by opening the SW gate. The initial value is 0 for operating modes without a main count direction and for main count direction up. For main count direction down, the initial value is the parameterized high count limit.

The first edge at input "DI start" starts the count with the load value.

The load value is loaded again at each further edge at the input "DI start".

The latch value is always exactly equivalent to the counter value at the time the edge trigger is generated.

The "DI start" value is indicated in the DB at status bit STS_STA.

The latch value is indicated by LATCH_LOAD in the DB.

The diagram below shows an example of "Latch/Retrigger" when load value = 0 and there is a positive edge at "DI start".

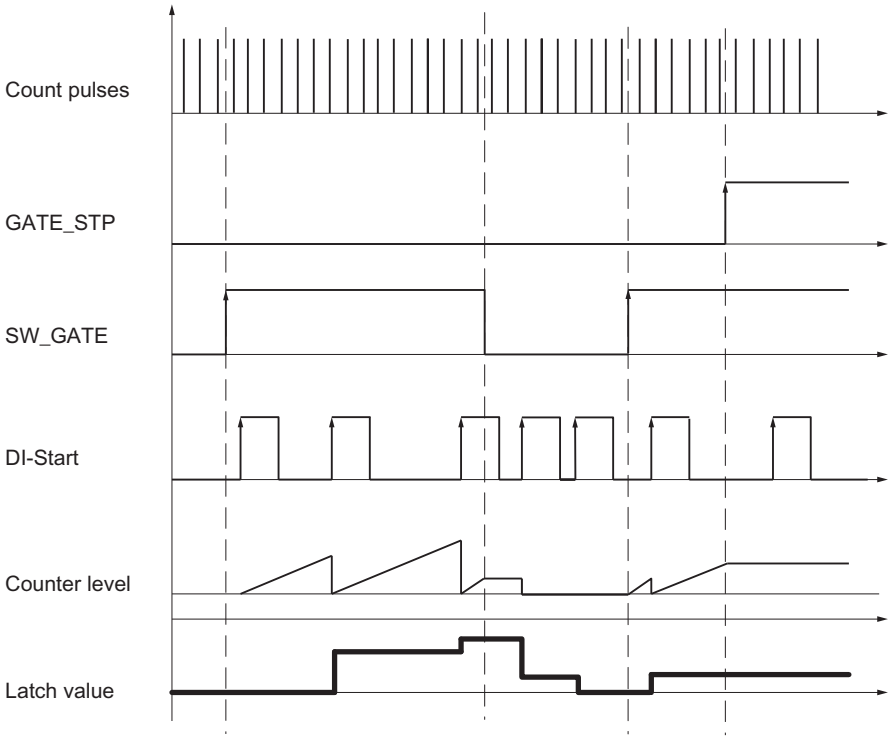

Measuring times between two edges (Page 227) using the internal time base 1 MHz (Page 221) is a special application for the "Latch/Retrigger" gate control.

Interrupting and exiting

Closing the software gate by resetting SW_GATE will interrupt counting, i.e. the count will continue when the software gate is reopened.
The unchanged counter value can be resaved with edges at "DI start" even if the software gate is closed.

However, the count will be cancelled if you close the software gate by executing GATE_STP of instruction CNT_CTL1. "DI start" can in this case no longer be used to save any counter values.

Hardware interrupt upon "Latch/Retrigger"

A hardware interrupt may be triggered each time a counter value is saved by executing "Latch/Retrigger". As a consequence, you may need to increase the interval between the edges. The hardware interrupts will be lost if the interrupt rate is higher than the system acknowledgement rate. This situation is reported by a diagnostic interrupt.

Selecting the counting range (S7-300, S7-400)

Description

There is a 32-bit wide count register on the module. With the Count range, you select whether the module is to count only in the positive range or whether the 32nd bit is interpreted as a sign bit thus allowing negative numbers to be represented. You can only select a count range if no main counting direction is set.

Select the desired counting range.

For more information, please see "Counting ranges and count limits (Page 179)".

Violation of the high limit, low limit and zero pass

A bit is set in the DB of the CNT_CTL1 instruction for both count limits when the high or low limit is violated.

A bit is also set in the DB in the "-31 to +31-bits" counting range in the event of a zero pass.

Depending on the count direction, violation of the high or low limit is also indicated in the "0 to +32 bits" counting range upon zero pass.

Event	Status bit in DB
Violation of the high limit	STS_OFLW is set.
Violation of the low limit	STS_UFLW is set.
Zero pass	STS_ZERO is set.

Initiating Hardware Interrupts

You can also signal the violation of the high limit, low limit and the zero pass events using hardware interrupts.

Selecting the gate function (S7-300, S7-400)

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 202): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 202): The counting process starts from the last current counter value after gate stop and restart.

Selecting main count direction (S7-300, S7-400)

Description

By configuring a main count direction (Page 185) (up or down), you can reduce the maximum counting range by setting an high count limit (Page 213). The parameterized counting range is then between 0 and the high count limit parameterized.

A main count direction can only be selected in the "Count once" and " Count periodically" modes.

You can select the following behavior:

- None (no main count direction)
- Up
- Down

Selecting an edge for latching (S7-300, S7-400)

Selecting edges

You can select the edge at the digital input "DI start" at which the counter values are to be saved (latched). The following behavior can be configured:

- Latching at a positive edge at "DI-Start"
- Latching at a negative edge at "DI-Start"
- Latching at both edges at "DI-Start"

Selecting period resolution (S7-300, S7-400)

Description

The value of the calculated period is displayed in 1 µs or 1/16 µs.

Select the required unit of resolution.

Possible Measuring Ranges with Error Indication

- The absolute error for a **resolution of 1 µs** is equal to 0.1‰ of the measured value.
- The table below sets out the measuring ranges and errors for a**resolution of 1/16 µs**:

Period T_{min} ± Absolute error
1/16 µs x (160 ± 1)
1/16 µs x (1,600 ± 1)
1/16 µs x (16,000 ± 3)
1/16 µs x (160,000 ± 20)
The absolute error for a period of 100ms or more is equal to 0.1‰ of the measured value.

Encoder (S7-300, S7-400)

Selecting the encoder signal type (S7-300, S7-400)

Selecting the signal type

1. Select the type of signal for the encoder used.
   In line with your selection and the operating mode, any option not possible for this specific encoder will be disabled. A disabled option is identified by grayed font in the corresponding selection window.
2. Select the options to be used with the selected encoder.

You can select the signal type from one of the following encoders:

- Encoder: 5 V incremental (Page 219) (5 V differential signal)
- Encoder: 24 V differential signal (Page 220) (24 V track A+B phase shifted)
- Encoder: 24 V pulse and direction (Page 220)
- Encoder: 24 V initiator (Page 220)
- Internal time base 1 MHz (Page 221)

Encoder: 5 V incremental (S7-300, S7-400)

5 V incremental encoder

Select the signal type "5V differential signal" to use a 5 V incremental encoder with differential signal A, /A, B, /B and N, /N.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation (Page 206)	Single	Only one edge of signal A is evaluated.
	Double	Both edges of signal A are evaluated.
	Quadruple	Both edges of signals A and B are evaluated.
Count direction (Page 204)	Normal	To adapt the count direction to the process
	Inverted	To adapt the count direction to the process
Max. count frequency	–	The maximum count frequency for this encoder is 500 kHz.
Encoder inputs	–	Not relevant
Monitoring (Page 204)	A+B+N	All signal pairs are monitored for short-circuit and wire break.
	A+B	Signal pairs A and B are monitored.
	A	Only signal pair A is monitored.
	None	No signal pair is monitored.

Encoder: 24 V incremental (S7-300, S7-400)

24 V incremental encoder

Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation (Page 206)	Single	Only one edge of signal A is evaluated.
	Double	Both edges of signal A are evaluated.
	Quadruple	Both edges of signals A and B are evaluated.
Count direction (Page 205)	Normal	To adapt the count direction to the process
	Inverted	To adapt the count direction to the process
Max. count frequency (Page 222)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 221)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Encoder: 24 V pulse and direction (S7-300, S7-400)

24 V pulse generator

Select signal type "24 V pulse and direction" to use a 24 V pulse generator with count signal A* and direction signal B*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation	–	Not relevant
Count direction	–	Not relevant
Max. count frequency (Page 222)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 221)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Encoder: 24 V initiator (S7-300, S7-400)

24 V initiator

Select the signal type "24 V initiator" to use a 24 V pulse generator with count signal A*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation	–	Not relevant
Count direction	–	Not relevant
Max. count frequency (Page 222)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 221)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Internal timebase 1 MHz (S7-300, S7-400)

Internal time base 1 MHz

Select the signal type "Internal time base 1 MHz " to use the internal 1 MHz time base instead of an external encoder.

This option allows you to measure the intervals between two successive edges at the DI start digital input, for example.

See also

Measuring times between two edges (Page 227)

Encoder input parameter assignment (S7-300, S7-400)

You have selected the "24 V pulse and direction" for the signal type (Page 219) in parameter assignment.

Encoder input parameter assignment

The count direction is defined by configuring the encoder inputs as follows. The table shows a change in count direction based on parameter assignment.

Parameter assignment	Terminal B*	Count direction
Sinking output	not wired	Down
	Short-circuited to ground	Up
Sourcing output/push pull	not wired	Up
	24 V connected	Down

You cannot reverse the direction of these count signals by inverting the B* signal.

Note

As all signals are added, this type of evaluation may cause the count value to "drift off" at the edge if the count signal oscillates.

Input filters for 24 V counter inputs (S7-300, S7-400)

Input filters for 24 V counter inputs

To suppress faults, you can configure input filters with a uniform filtering time for the 24 V counter inputs A*, B* and N*. Input filters available:

Features	Input filter 1 (default setting)	Input filter 2
Typical input delay	1 µs	15 µs
Maximum count frequency	200 kHz	20 kHz
Minimum pulse width of the count signals	2.5 µs	25 µs

Entering pulses per encoder revolution (S7-300, S7-400)

Pulses per encoder revolution

For "Rotational speed measurement (Page 193)" mode, you must also configure the pulses per encoder revolution.

The value for the number of pulses per encoder revolution can be found on the rating plate, or in the technical data of your encoder.

Inputs (S7-300, S7-400)

Selecting gate signals (S7-300, S7-400)

Controlling the hardware gate

The following options are available for hardware gate control:

- Level-controlled hardware gate (with level at "DI start")
- Edge-controlled hardware gate (with edges at "DI start", "DI stop")

For additional information on this topic, please refer to "Hardware gate (Page 201)".

Selecting minimum pulse width (S7-300, S7-400)

Minimum pulse width

To suppress faults, you can configure input filters (RC elements) with a uniform filtering time for the digital inputs.

Two input filters are available with the following data:

Features	Input filter 1 (default setting)	Input filter 2
Typical input delay	1 µs	15 µs
Minimum pulse width of input signals	2.5 µs	25 µs

Setting the counter (DI set) (S7-300, S7-400)

Setting the counter (S7-300, S7-400)

Overview

If you want to start the count from a specific value, the so-called load value, you must define the signal that is to be used to set the counter to the load value. You can set the counter:

- With the L_DIRECT or L_PREPAR input parameter of the CNT_CTL1 instruction
- With an external signal, using either the "DI set" input or "DI set" in combination with the zero mark of the encoder

Load value

Any number within the counting range can be specified as the load value.

The load value is interpreted according to the selected counting range. If, for example, you specify FFFF FFFF H as the load value, this will be interpreted as 4 294 967 295 in the "0 to +32 bits" counting range and as -1 in the count range "-31 to +31 bits".

Enter the load value in the DB of the CNT_CTL1 instruction and transfer it to the module with CNT_CTL1.

The following ranges of values apply for the load value:

Value range for load values	Main count direction		
	None	Up	Down
Low limit	Maximum low count limit	$-2^{31} + 1$	2
High limit	Maximum high count limit	Parameterized high count limit - 2	$2^{31} - 1$

Setting the counter with an external signal

You can select two different external signals with which to set the counter to the load value:

- "DI set" (Page 224) only
- "DI set" and zero mark (Page 226) of the encoder

You use the encoder zero mark if you want to synchronize the counter to a specific counter value at a specific point in your process. This improves the precision of the counting process.

The counter is set independently of the counting mode.

When the counter has been set with an external signal, bit STS_SYNC will be set in the DB.

Note

Synchronizing a counter with the zero mark only makes sense if the gate is open.

If you have only enabled one count direction when setting a counter with an external signal, you must note that the current count direction will be saved (frozen) when the gate is closed. This allows the counter to be synchronized against the enabled count direction.

Setting the counter with "DI set" (S7-300, S7-400)

Setting the counter with "DI set"

The counter can be loaded with the load value by means of a rising edge at "DI set".

You can define the response of the FM 350-1 to a rising edge at DI set with variables ENSET_UP and ENSET_DN in the DB of the CNT_CTL1 instruction and by assigning parameters.

Parameter	Behavior of the FM 350-1
ENSET_UP set	The counter is only set if counting up.
ENSET_DN set	The counter is only set if counting down.
ENSET_UP and ENSET_DN set	The counter is set for counting up and counting down.
"Single setting of counter" parameter assignment	The counter is set only in response to the first rising edge at "DI set". If the counter is to be set again, you must first set ENSET_UP or ENSET_DN again. The counter will then be set again at the next rising edge at "DI set".
"Multiple setting of counter" parameter assignment	The counter is set in response to each rising edge at "DI set" as long as ENSET_UP and/or ENSET_DN are set.

Note

It is imperative that you set one of the two variables ENSET_UP and ENSET_DN so the counter can be set by the digital input "DI set".

Single setting with "DI set"

The figure below shows single setting of the counter with digital input "DI Set". In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

The counter is set in response to the first rising edge at digital input "DI set" provided that ENSET_UP is also set. If you want to set the counter again, you must first reset ENSET_UP and then set it again. The next rising edge at digital input "DI set" will then result in the counter being set.

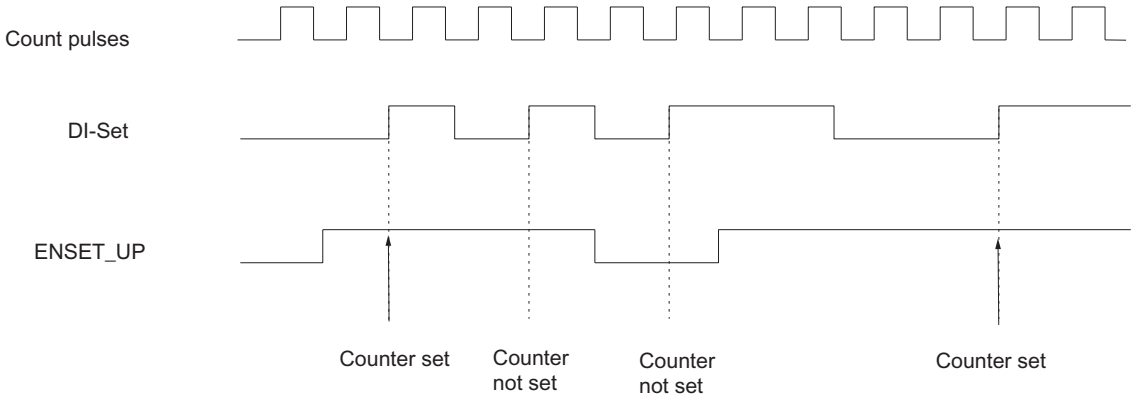

Multiple setting with "DI Set"

The figure below shows multiple setting of the counter with "DI set". In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

The counter is set at every rising edge at "DI set" provided that ENSET_UP is also set. If you reset ENSET_UP, the counter will not be set by "DI set". Only once you have set ENSET_UP again will the next positive edge at "DI set" set the counter.

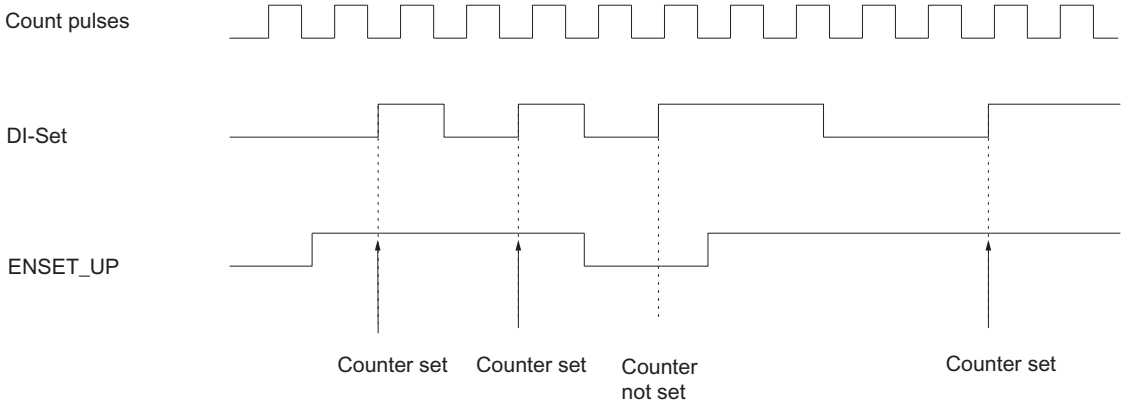

Setting the counter with "DI set" and zero mark (S7-300, S7-400)

Setting the counter with "DI set" and zero mark

If you assign parameters so that the counter is set by the encoder zero mark ("Evaluate zero mark for setting"), the counter will be set to the load value at the rising edge of the zero mark.

Setting is performed only if the "DI set" input is also set at the time of the rising edge of the zero mark.

You can define the response of the FM 350-1 in the event of a zero mark rising edge with the variables ENSET_UP and ENSET_DN in the DB of the CNT_CTL1 instruction and by assigning parameters.

Input parameters	Behavior of the FM 350-1
ENSET_UP set	The counter is only set if counting up.
ENSET_DN set	The counter is only set if counting down.
ENSET_UP and ENSET_DN set	The counter is set for counting up and counting down.
"Single setting of counter" parameter assignment	The counter is only set in response to the first rising edge of the zero mark. If the counter is to be set again, you must first set ENSET_UP or ENSET_DN again (edge evaluation). The counter is then set again in response to the next rising edge of the zero mark.
"Multiple setting of counter" parameter assignment	The counter is set at every rising edge of the zero mark as long as ENSET_UP and/or ENSET_DN are set.

Note

It is imperative that you set one of the two variables ENSET_UP and ENSET_DN so that the counter can be set with the zero mark.

Single setting with "DI set" and zero mark

The figure below shows single setting of the counter with the zero mark. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

The counter is set upon the first rising edge of the zero mark, provided that ENSET_UP and "DI set" are also set.

If you want to set the counter again, you must first reset ENSET_UP and then set it again. If "DI set" is not set, setting is performed upon the first zero mark after "DI set" is set. If "DI set" is set, setting is performed with the next zero mark.

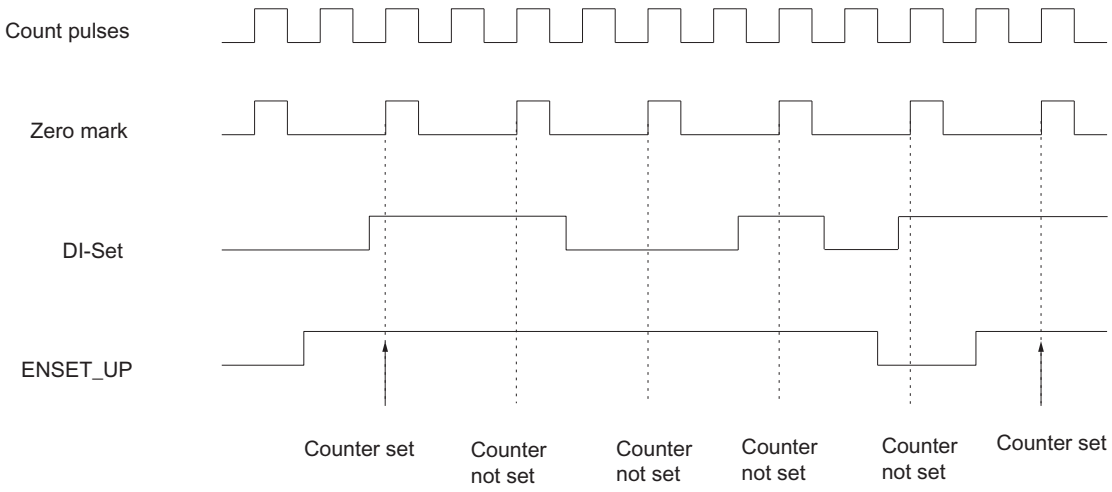

Multiple setting with "DI set" and zero mark

The figure below shows multiple setting of the counter with the zero mark. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

The counter is set with every rising edge of the zero mark provided that ENSET_UP and "DI set" are set.

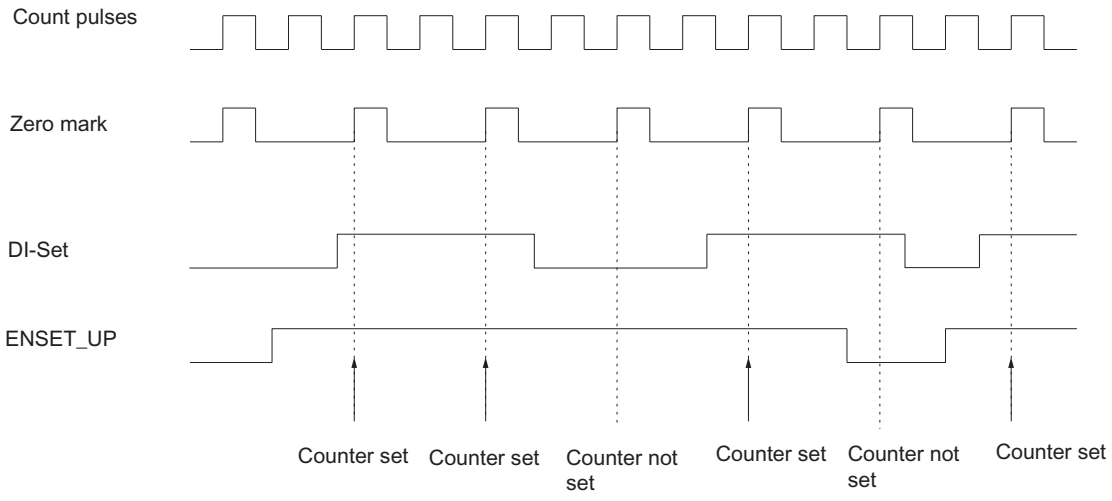

Measuring times between two edges (S7-300, S7-400)

Introduction

You can measure the time intervals between two successive edges at the "DI start" digital input.

Requirements

Ensure the following conditions are met before measurement:

- Do not connect any encoders to the FM 350-1.
- Configure a counting mode (Page 212) under operating modes.
- Configure the "Latch/Retrigger" (Page 213) gate control under operating modes.
- Configure the signal type "Internal time base 1 MHz" (Page 219) under encoder.

Selecting edges

To measure the time between two immediately successive ...	... Parameterize
Rising edges at "DI start"	Latch with positive edge
Falling edges at "DI start"	Latch with negative edge
Any edges at "DI start"	Latch with both edges

See also: Selecting an edge for latching (Page 218)

Function principle

The FM 350-1 uses an internal time base 1 MHz to measure times. The time measurement starts upon the first edge at "DI start". At every subsequent edge at "DI start", the time in µs that has elapsed since the last edge was received is saved as the latch value LATCH_LOAD in the feedback interface.

Outputs (S7-300, S7-400)

Defining behavior of the outputs in counting modes (S7-300, S7-400)

Comparison values and outputs (S7-300, S7-400)

Description

You can store two comparison values (Comparison value 1 and 2) in the FM 350-1. These comparison values are assigned to the two digital outputs (Comparison value 1: DO0, Comparison value 2: DO1). The comparison values are compared with the current counter value. When the counter value reaches a comparison value, the relevant output can be set.

Both comparison values are assigned to DO1 in "Switch at comparison values (Page 233)".

Comparison values 1 and 2

You enter the two comparison values in the DB of the CNT_CTL1 instruction (CMP_V1, CMP_V2) and transfer them to the FM 350-1 by setting the bits T_CMP_V1 or T_CMP_V2. This does not affect counting.

The comparison values must be within the limits of the selected counting range. The comparison value is interpreted according to the selected counting range. If you specify FFFF FFFF H as the comparison value, for example, this value will be interpreted as 4,294,967,295 in the "0 to +32 bits" counting range and as -1 in the"-31 to +31 bits" counting range.

The following ranges of values are permitted for the comparison values:

Value ranges for comparison values	Main count direction		
	None	Up	Down
Low limit	Maximum low count limit	-2^{31}	1
High limit	Maximum high count limit	Parameterized high count limit -1	$2^{31} - 1$

Enabling the outputs

Before the outputs can be set, you must first enable them by setting the relevant bits in the DB. If you reset one of these bits, the associated output will be switched off immediately even if you have configured a pulse duration for it.

The following bits must be set to enable the outputs:

Output	... is enabled by enable bit
DO0	CTRL_DO0
DO1	CTRL_DO1

Setting and resetting the outputs

If you have configured "No comparison" for output behavior, you can set and reset enabled outputs using the corresponding bits in the DB.

Outputs are set and reset with the following bits:

Output	... is set by	... is reset by
DO0	SET_DO0 = 1	SET_DO0 = 0
DO1	SET_DO1 = 1	SET_DO1 = 0

Behavior of the outputs in counting modes (S7-300, S7-400)

Specifying the behavior of the outputs (S7-300, S7-400)

Behavior of the DO0 and DO1 outputs

Select the required behavior for the output in question: Activate the corresponding option button:

- Substitute value "1" at CPU STOP (Page 230)
- No comparison (Page 231)
- On when counter value >= comparison value (Page 231)

- On when counter value <= comparison value (Page 231)
- Active on reaching the comparison value for pulse duration up (Page 232)
- Active on reaching the comparison value for pulse duration down (Page 233)
- Active on reaching the comparison value for pulse duration up/down (Page 232)
- Switch at comparison values (Page 233) (only possible for output DO1)

Note

To activate the settings, enter values for comparison values 1 and 2 in the DB with the counter data and transfer them to the module with the CNT_CTL1 instruction; you must also set the CTRL_DO0 and CTRL_DO1 bits in the DB in order to enable the outputs. See also "Comparison values and outputs (Page 228)".

Output default setting

The default setting for the outputs is deactivated.

Behavior of the digital outputs in isochrone mode

In isochrone mode, the outputs DO0 and DO1 switch as soon as the comparison conditions are fulfilled, and are thus independent of the PROFIBUS DP cycle.

Exception:

If you have configured "No comparison" for the behavior of the digital outputs and, after enabling with CTRL_DO0 or CTRL_DO1, you activate the outputs with the control signals SET_DO0 or SET_DO1, the outputs will be set and reset at the time T_o.

Substitute value "1" at CPU STOP (S7-300, S7-400)

Substitute value "1" at CPU STOP

If you selected the "Substitute value" option under "Reaction to CPU STOP" in the module's basic parameters and the CPU goes into STOP mode, the module will output the substitute values you configured in this dialog at the digital outputs. Check the checkbox for DO0, for example, and output DO0 will be set at CPU STOP.

The substitute values are retained at the digital outputs after the CPU transition from STOP to RUN until the next value is input.

The outputs are reset if you change the "Reaction to CPU STOP" basic parameter.

⚠ CAUTION
Danger of damage to equipment. Parameterized substitute values are always output when the CPU goes into STOP mode even if the gate is closed and even at outputs that are not enabled. Only configure substitute values which will not lead to dangerous states in the system if output.

No comparison (S7-300, S7-400)

No comparison

The output remains deactivated and is not affected by comparison value, zero pass, high limit violated or low limit violated events. The output DOx may be used as a purely digital output. When enabled, it can be set and reset with the SET_DOx bit.

On when counter value >= comparison value (S7-300, S7-400)

On when counter value >= comparison value

The output is set to 1 if the counter is in the range between the comparison value and high limit violated.

Setting the counter to a value between the comparison value and high limit violated sets the output to 1.

The hatched area in the graphic below means that the output is set in this range.

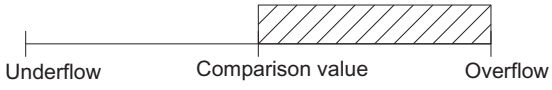

Please note the basic conditions for the behavior of the outputs (Page 235).

See also

Effect of the hysteresis (Page 235)

On when counter value <= comparison value (S7-300, S7-400)

On when counter value <= comparison value

The output is set to 1 if the counter is in the range between the comparison value and low limit violated.

Setting the counter to a value between the comparison value and low limit violated sets the ouput to 1.

The hatched area in the graphic below means that the output is set in this range.

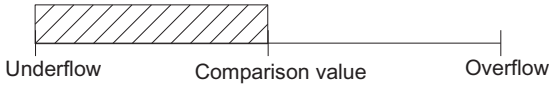

Please note the basic conditions for the behavior of the outputs (Page 235).

See also

Effect of the hysteresis (Page 235)

Active on reaching the comparison value for pulse duration up/down (S7-300, S7-400)

Active on reaching the comparison value for pulse duration up/down

The output is set to 1 for the duration of the pulse when the counter has reached the comparison value independently of the count direction.

Requirements:

- Main count direction: "None"

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).

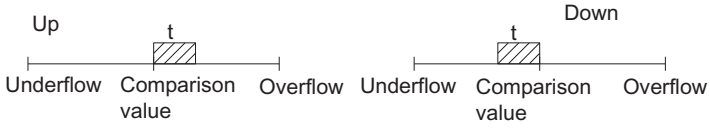

Active on reaching the comparison value for pulse duration up (S7-300, S7-400)

Active on reaching the comparison value for pulse duration up

The output is set to 1 for the pulse duration when the counter reaches the comparison value when counting up.

This requires one of the following:

- Main count direction: "None"
- Main count direction: "Up"

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).

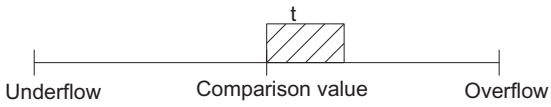

Please note the basic conditions for the behavior of the outputs (Page 235).

See also

Effect of the hysteresis (Page 235)

Active on reaching the comparison value for pulse duration down (S7-300, S7-400)

Active on reaching the comparison value for pulse duration down

The output is set to 1 for the pulse duration when the counter reaches the comparison value when counting down.

This requires one of the following:

- Main count direction: "None"
- Main count direction: "Down"

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).

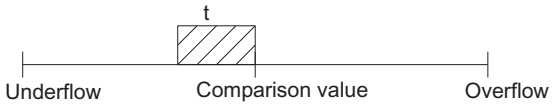

Please note the basic conditions for the behavior of the outputs (Page 235).

See also

Effect of the hysteresis (Page 235)

Switch at comparison values (S7-300, S7-400)

Switch at comparison values

Output DO1 switches at two comparison values if the following conditions are met:

- You have configured "No comparison" for DO0.
- You have configured "Switching at comparison values" for DO1.
- You have loaded the two comparison values CMP_V1 and CMP_V2 .
- You have enabled output DO1 with CRTL_DO1.

The following table shows when DO1 is switched on or off:

Comparison values CMP_V1 and CMP_V2	DO1 is switched on when	DO1 is switched off when
CMP_V1 < CMP_V2	CMP_V1 ≤ Counter value ≤ CMP_V2	Counter value < CMP_V1 or counter value > CMP_V2
CMP_V1 = CMP_V2	CMP_V1 = Counter value = CMP_V2	CMP_V1 ≠ Counter value ≠ CMP_V2
CMP_V1 > CMP_V2	Counter value < CMP_V2 or counter value > CMP_V1	CMP_V2 ≤ Counter value ≤ CMP_V1

The result of the comparison is indicated by the STS_COMP2 status bit.

You cannot acknowledge and reset the STS_COMP2 status bit until the comparison condition is no longer met.

The status at output DO1 is indicated by the STS_CMP2 status bit.

There is no hysteresis in the case of this output behavior.

It is not possible to control the DO1 output with the SET_DO1 control bit in the case of this output behavior.

The figures below show the status of output DO1 as determined by the counter value and comparison values:

- CMP_V2 > CMP_V1 at the start of counting

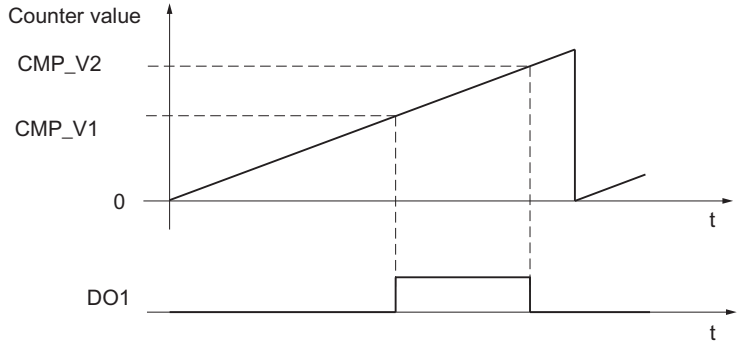

- CMP_V2 < CMP_V1 at the start of counting

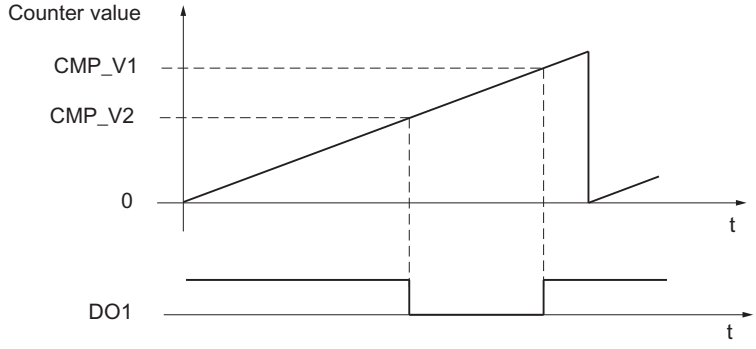

Basic conditions for the behavior of the digital outputs (S7-300, S7-400)

Basic conditions for the behavior of the digital outputs

You must observe the following basic conditions when configuring the behavior of the digital outputs:

If...	then...
... ...you want to configure an output for "On when counter value >= comparison value or <= comparison value",	... ...you must ensure that the time between these events is greater than the minimum switching time of the outputs (switching time: 300 µs); the control pulses at the outputs will otherwise be lost. If the counter value reaches the relevant comparison value again while the output is still active, no new pulse will be initiated. A further pulse can only be initiated when the output is no longer active.
... you want to configure an output "On when counter value >= comparison value",	... you must not enable a hardware interrupt on "Reaching associated comparison value up or down".
... you want to configure an output "On when counter value <= comparison value",	... you must not enable a hardware interrupt on "Reaching associated comparison value up or down".
... you want to configure an output that is "Active on reaching the comparison value for pulse duration up",	... you must not enable a hardware interrupt on "Associated comparison value down is reached".
... ...you want to configure an output "Active on reaching comparison value for pulse duration down",	... ...you must not enable a hardware interrupt on "Reaching the relevant comparison value up".

Effect of the hysteresis (S7-300, S7-400)

Function of hysteresis with parameter assignment "On when counter value >= comparison value or <= comparison value"

The example in the figure below demonstrates the effect of hysteresis. The figure shows the differences in the behavior of an output when hysteresis of 0 (= switched off) is parameterized as opposed to hysteresis of 3. In the example, the comparison value = 5

Parameter assignment in this example is as follows:

- Main count direction: "Up"
- Output response: "On when counter value >= comparison value"

The hysteresis is set when the comparison condition (counter value = 5) is fulfilled. When the hysteresis is active, the comparison result remains unchanged.

If the count value overshoots/undershoots the range of the hysteresis (at counter value 2 or 8 in the example) the hysteresis ceases to be active. The comparator switches again according to its comparison conditions, i.e., in the example at counter value 5.

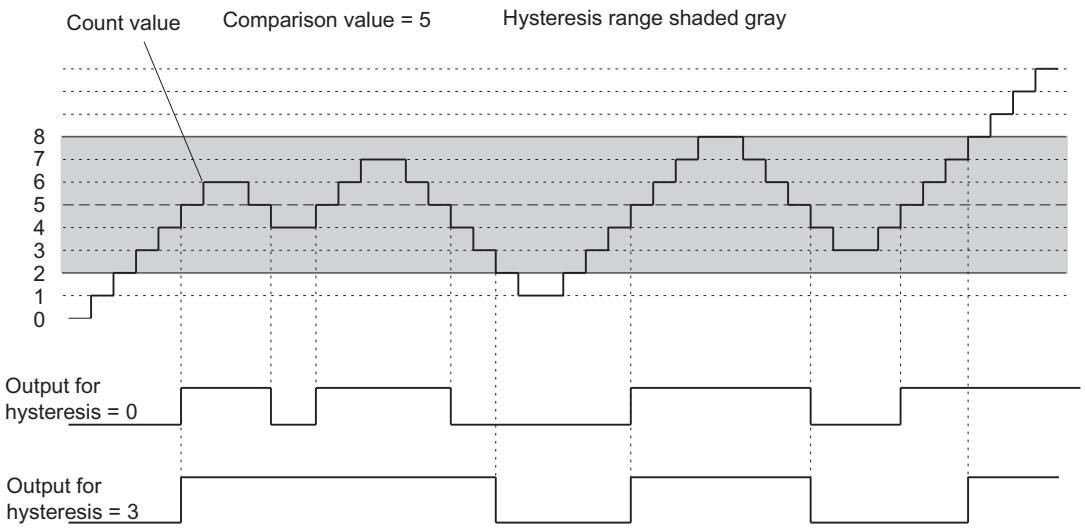

Note

If the counter value is equal to the comparison value, and the hysteresis is active, FM 350-1 resets the output when the count direction changes at the comparison value (see the diagram below).

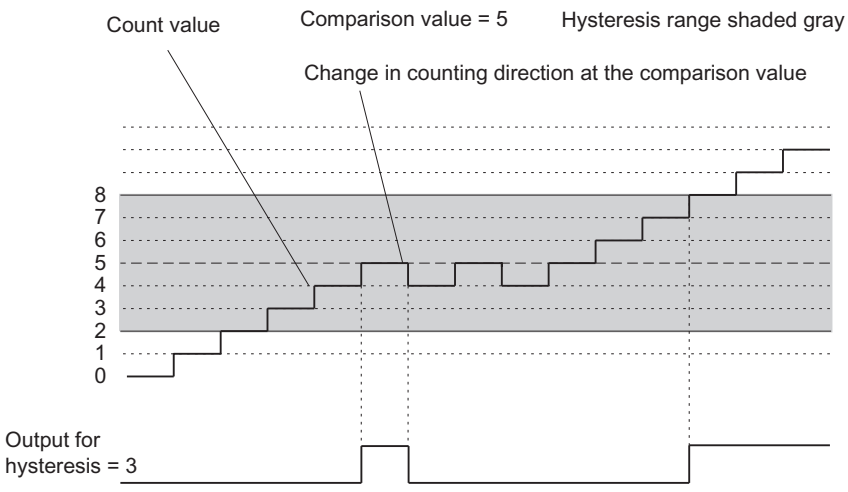

Effect of hysteresis with parameter assignment "Active on reaching the comparison value for pulse duration up or down"

The example in the figure below demonstrates the effect of hysteresis. The figure shows the differences in the behavior of an output when hysteresis of 0 (= switched off) is parameterized as opposed to hysteresis of 3. In the example, the comparison value = 5

Parameter assignment in this example is as follows:

- Main count direction: "None"
- Output response: "Active on reaching the comparison value for pulse duration up"
- Pulse duration > 0

The hysteresis is set when the comparison condition (counter value = 5) is fulfilled, and a pulse of the programmed duration is output.

If the counter value leaves the hysteresis range, the hysteresis ceases to be active.

When hysteresis is enabled, the FM 350-1 stores the count direction. A pulse is output if the signal overshoots hysteresis range in opposite direction of the direction saved previously.

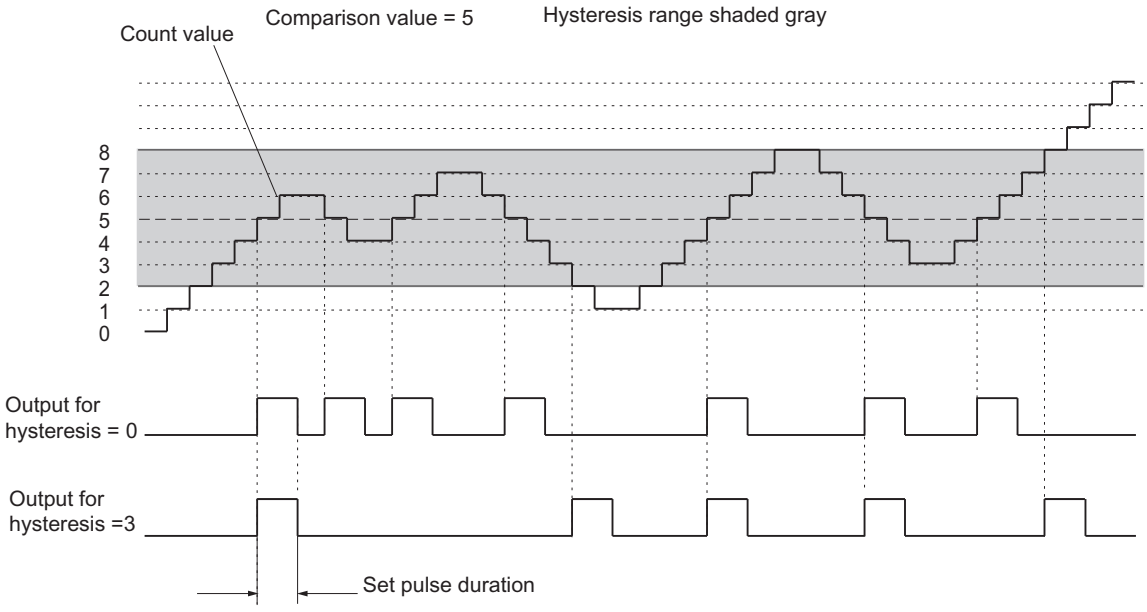

Defining behavior of the outputs in measuring modes (S7-300, S7-400)

Limits and outputs (S7-300, S7-400)

Low and high limit

You can save an high and a low limit for frequency, speed and period measurements and activate digital output DO0 if one of these limits is exceeded. You can configure these limits and change them via the control interface.

The following value ranges are possible for the low and high limits:

Measuring mode	Low limit (LL)	High limit
Frequency measurement (5 V encoders)	0 to 499,999,999 × 10^{-3} Hz	LL+1 to 500,000,000 × 10^{-3} Hz
Frequency measurement (24 V encoders)	0 to 199,999,999 × 10^{-3} Hz	LL+1 to 200,000,000 × 10^{-3} Hz
Rotational speed	0 to 24,999,999 × 10^{-3} /min	LL+1 to 25,000,000 × 10^{-3} /min
Period measurement (resolution = 1 µs)	0 to 119,999,999 µs	LL+1 to 120,000,000 µs
Period measurement (resolution = 1/16 µs)	0 to 1,919,999,999 µs	LL+1 to 1,920,000,000 µs

Enabling the outputs

Before the outputs can be set, you must first enable them by setting the relevant bits in the DB. If you reset one of these bits, the associated output will be switched off immediately.

Output	... is enabled by enable bit
DO0	CTRL_DO0
DO1	CTRL_DO1

Behavior of the outputs in measuring modes (S7-300, S7-400)

Behavior of the digital outputs

Digital output DO0

You can select one of four possible responses to reaching limits for digital output DO0. The various options are shown in the table below.

Digital output DO0 parameter assignment	Behavior of digital output DO0	Switching time	
		Isochrone mode	Non-isochrone mode
No comparison	Not affected by limit monitoring. If output DO0 is already set, it can be reset by changing the parameter to "No comparison". You can use output DO0 freely as a digital output and set and reset it with the control signal SET_DO0 if you have enabled it with the control signal CTRL_DO0.	At time T_o.	Immediately after setting or resetting the output
Outside limits	DO0 is set in each of the following cases: • measured value < low limit • measured value > high limit	At the end of the update time at time T_i	At the end of the update time
Violating low limit	DO0 is set if • measured value < low limit		
Exceeding high limit	DO0 is set if • measured value > high limit		

Select how you want output DO0 to respond. Activate the corresponding option button.

Digital output DO1

You can use output DO1 freely as a digital output and set and reset it with the control signal SET_DO1 provided it has been enabled.

DO1 is not affected by limit monitoring.

In isochrone mode, DO1 switches at the end of the update time at time T_o.

Specifying pulse duration (S7-300, S7-400)

Pulse duration

You can set a pulse duration in order to adapt to the actuators (contactors, actuators, etc) used in your process. The pulse duration indicates how long outputs DO0 and DO1 are active on reaching a comparison value.

The pulse duration only affects the main count direction when "Main count direction up" or "Main count direction down" has been assigned.

The pulse duration can affect both main count directions if a main count direction has not been parameterized.

The pulse duration starts when the output is set. The pulse duration inaccuracy is less than 1 ms.

A value between 0 and 500 ms may be set for the pulse duration. This value applies to both outputs.

If the pulse duration = 0 ms, the output is set when the comparison value is reached and is reset at the next count pulse.

The value 0 is set by default for the pulse duration.

Note

If you specify pulse duration = 0 ms, the output will remain active as long as the counter value equals the comparison value (see top half of figure below).

Control pulses may be lost at the outputs if the time interval between the count pulses is less than the switching times of the digital outputs (up to 300 µs) (see bottom half of figure below).

You should therefore make sure that the interval between the count pulses is greater than the switching times of the digital outputs.

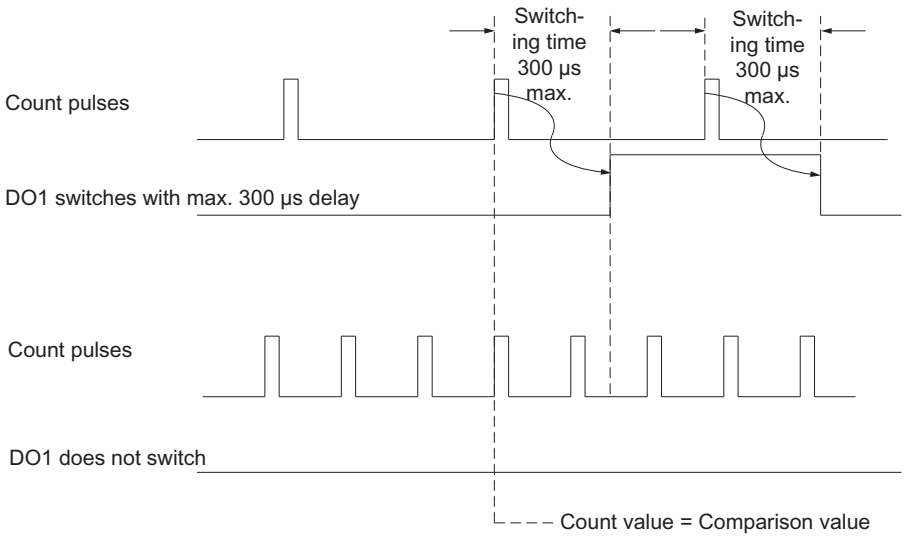

Setting hysteresis (S7-300, S7-400)

Function principle of the hysteresis

An encoder can stop at a certain position, and then "oscillate" around this position. This state causes the counter level to fluctuate around a certain value. If there is a comparison value within this fluctuation range, for example, the associated output would be switched on and off in time with these fluctuations. The FM 350-1 has configurable hysteresis to prevent this switching in response to small fluctuations.

A value between 0 and 255 can be set for hysteresis.

The effect of hysteresis is as follows in accordance with the value parameterized:

Hysteresis	Effect
Hysteresis value n = 0, 1	The hysteresis has no effect (switched off) The output reacts to the slightest change in the counter value.
2 ≤ Hysteresis value n ≤ 255	The hysteresis takes effect. The output does not react until the counter value is offset by n units to the comparison value.

The hysteresis applies to both high limit violated and low limit violated.

Enable hardware interrupts (S7-300, S7-400)

Enabling hardware interrupt trigger (S7-300, S7-400)

Introduction

The FM 350-1 allows you to set which events are to trigger a hardware interrupt during operation.

What is a Hardware Interrupt?

If you want to set a response to a specific event independently of the CPU cycle, the FM 350-1 can initiate a hardware interrupt. The CPU interrupts the cyclic program on receiving the interrupts and executes the hardware interrupt OB.

Enabling the Hardware Interrupt

Hardware interrupts are enabled in two steps:

1. When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.
2. You then select the relevant events under "Enable hardware interrupts":
   - Events to trigger a hardware interrupt in counting modes (Page 241)
   - Events to trigger a hardware interrupt in measuring modes (Page 242)

Hardware Interrupt OB, OB 40

If a hardware interrupt occurs, the user program is interrupted, the data is transferred from the module to the start information of OB40 (Page 242) and OB40 is called. The hardware interrupt is acknowledged by exiting OB 40.

If no OB 40 is programmed, the CPU goes into STOP mode. If you then switch back to RUN, the hardware interrupt requirements are deleted.

Lost hardware interrupt

If an event occurs that is to initiate a hardware interrupt and the same previous event has not yet been acknowledged, no further hardware interrupt is initiated; the hardware interrupt is lost. This may lead to the "Hardware interrupt lost" diagnostic interrupt, depending on the parameterization.

Default setting

No hardware interrupts are enabled in the default setting.

Events to trigger a hardware interrupt in counting modes (S7-300, S7-400)

Which events can trigger a hardware interrupt in counting modes?

The following events can trigger a hardware interrupt in counting modes during operation of the FM 350-1:

- Opening of the gate (in the operating modes with hardware or software gate)
- Closing of the gate (in the operating modes with hardware or software gate)
- Violation of the high limit
- Violation of the low limit

- Zero pass
- Reaching comparison value 1 or 2 in the up direction
- Reaching Comparison value 1 or 2 in the backward direction
- Setting the counter (DI set)
- Latching

Select the check box for the relevant events.

You can select any number of events for triggering a hardware interrupt. Observe the basic conditions for output behavior (Page 235) applicable to hardware interrupts on reaching a comparison value.

Events to trigger a hardware interrupt in measuring modes (S7-300, S7-400)

Which events can trigger a hardware interrupt in measuring modes?

The following events can trigger a hardware interrupt in measuring modes during operation of the FM 350-1:

- Opening of the gate (in modes with hardware or software gate)
- Closing of the gate (in operating modes with hardware or software gate)
- Violating low limit
- Exceeding high limit
- End of measurement

Select the check box for the relevant events.

You can select any number of events for triggering a hardware interrupt.

OB 40 start information (S7-300, S7-400)

OB 40 start information

The temporary variable OB40_POINT_ADDR is described in the start information of OB 40.

The OB40_POINT_ADDR variable (bytes 8 to 11) comprises 4 bytes. The information about the event which has triggered the hardware interrupt is entered in bytes 8 and 9.

The table shows which bits are set for which interrupt. Any bits not listed are irrelevant and are zero.

Byte	Bit	Meaning: Interrupt in the case of...
8	0	Opening the gate
	1	Closing the gate
	2	Violation of the high limit (for count mode)
		Measured value outside limits (in measuring mode)
	3	Violation of the low limit (for count mode)
		End of measurement (at Measure mode)
	4	Reaching comparison value 1 in the up direction
	5	Reaching comparison value 1 in the down direction
	6	Reaching comparison value 2 in the up direction
	7	Reaching comparison value 2 in the down direction
9	0	Zero pass
	5	Setting the counter (DI set) (synchronization)
	7	Latching

Control and feedback interface of the FM 350-1 (S7-300, S7-400)

Overview (S7-300, S7-400)

If you want to operate the FM 350-1 without instructions, you can control and monitor the FM 350-1 directly via the control and feedback interface (user data interface).

The user data interface begins with the start address of the module and is 16 bytes long.

You can read the feedback interface with the load commands.

You can write to the control interface with the transfer commands.

You cannot mix load/transfer commands and the programming with instructions.

Control interface for the count modes (S7-300, S7-400)

Control interface for the count modes

Parameter LOAD_VAL (bytes 0 to 3) has two meanings:

- If you set the L_DIRECT or L_PREPAR bit, LOAD_VAL is interpreted as a load value.
- If you set the C_DOPARA bit, you can define the behavior of outputs DO0 and DO1 in byte 0; bytes 1 and 2 are interpreted as hysteresis and pulse duration.

The table below shows the assignment of the FM 350-1 control interface (outputs) for the count modes.

Offset to the start address	Parameters	Meaning				
Bytes 0 to 3	LOAD_VAL	Load value; direct and preparatory loading with bit L_DIRECT Load value; preparatory loading with bit L_PREPAR				
Byte 0		Characteristics of the DO0 and DO1 outputs, hysteresis and the pulse duration; defined with the C_DOPARA bit				
		Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	**Behavior of output DO0**
		x	0	0	0	No comparison
		x	0	0	1	On when counter value >= comparison value
		x	0	1	0	On when counter value <= comparison value
		x	0	1	1	Active on reaching the comparison value for pulse duration up/down
		x	1	0	0	Active on reaching the comparison value for pulse duration up
		x	1	0	1	Active on reaching the comparison value for pulse duration down
		Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Behavior of output DO1**
		x	0	0	0	No comparison
		x	0	0	1	On when counter value >= comparison value
		x	0	1	0	On when counter value <= comparison value
		x	0	1	1	Active on reaching the comparison value for pulse duration up/down
		x	1	0	0	Active on reaching the comparison value for pulse duration up
		x	1	0	1	Active on reaching the comparison value for pulse duration down
		x	1	1	0	Switch to Comparison value
		x = irrelevant				
Byte 1		Hysteresis (value range 0 to 255)				
Byte 2		Pulse duration (value range 0 to 250)				
Byte 3		Reserve = 0				
Bytes 4 to 7	CMP_V1	Comparison value 1; load with the T_CMP_V1 bit				
Bytes 8 to 11	CMP_V2	Comparison value 2; load with the T_CMP_V2 bit				
Byte 12	– NEUSTQ – – OT_ERR_A – – –	Bit 7: Reserve = 0 Bit 6: Restart acknowledgement Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: Operator error acknowledgement Bit 2: Reserve = 0 Bit 1: Reserve = 0 Bit 0: Reserve = 0				

Offset to the start address	Parameters	Meaning
Byte 13	–	Bit 7: Reserve = 0
	–	Bit 6: Reserve = 0
	–	Bit 5: Reserve = 0
	–	Bit 4: Reserve = 0
	SW_GATE	Bit 3: SW gate control bit
	GATE_STP	Bit 2: General gate stop
	ENSET_DN	Bit 1: Enable synchronization down
	ENSET_UP	Bit 0: Enable synchronization up
Byte 14	–	Bit 7: Reserve = 0
	–	Bit 6: Reserve = 0
	–	Bit 5: Reserve = 0
	–	Bit 4: Reserve = 0
	SET_DO1	Bit 3: Control bit DO1
	SET_DO0	Bit 2: Control bit DO0
	CTRL_DO1	Bit 1: Enable DO1
	CTRL_DO0	Bit 0: Enable DO0
Byte 15	–	Bit 7: Reserve = 0
	C_DOPARA[1)]	Bit 6: Change function of DO0/DO1, hysteresis or pulse duration
	RES_ZERO	Bit 5: Reset status bits for zero pass, violation of high and low limit, and comparator
		Bit 4: Reset synchronization status bit
	RES_SYNC	Bit 3: Load comparison value 2
	T_CMP_V2[2)]	Bit 2: Load comparison value 1
	T_CMP_V1[2)]	Bit 1: Load counter in preparation
	L_PREPAR[2)]	Bit 0: Direct and preparatory loading of counter
	L_DIRECT[2)]	
[1)] Do not set this bit at the same time as bits 0, 1, 2 or 3 of byte 15. [2)] Do not set this bit at the same time as bit 6 of byte 15.		

Explanation of the control bits for the count modes

Control bit	Explanation
C_DOPARA	Set this bit to change the function and behavior of DO0 and DO1, and the hysteresis and pulse duration. The values of bytes 0 to 2 are accepted as a new function, hysteresis and pulse duration at DO0 and DO1. Transfer the old values if you want to discard all changes.
CTRL_DO0	Enable DO0 Set this bit to enable output DO0.
CTRL_DO1	Enable DO1 Set this bit to enable output DO1.
ENSET_DN	Set this bit to enable loading of the counter for up counts.
ENSET_UP	Set this bit to enable loading of the counter for down counts.
GATE_STP	Set this bit to close the internal gate.

Control bit	Explanation
L_DIRECT	Set this bit to enable direct and preparatory loading of the counter.
L_PREPAR	Set this bit to enable preparatory loading of the counter.
NEUSTQ	Set this bit to acknowledge a start of the FM 350-1. After its restart, FM 350-1 does not recognize any control or data input unless this bit is set. The NEUSTQ bit is set by the CNT_CNTL1 instruction as soon as the feedback signal FM_NEUST is set and the feedback signal is FM_NEUSTQ = 0. It is reset by the CNT_CNTL1 instruction when the FM_NEUST bit is reset by the FM 350-1 and the FM_NEUSTQ bit has been set by the FM 350-1. If you do not use the CNT_CNTL1 instruction, you need to coordinate the restart via the user program.
OT_ERR_A	Set this bit to acknowledge an operator error. For detailed information on operator errors, read the feedback interface before you acknowledge the error. The error message is no longer valid after its acknowledgement.
RES_SYNC	Use this bit to reset and acknowledge the feedback bit STS_SYNC and thereby enable loading of the counter by synchronization input DI-Set.
RES_ZERO	Use this bit to reset the feedback bits STS_ZERO, STS_OFLW, STS_UFLW, STS_COMP1 and STS_COMP2.
SET_DO0	Provided you have set the "No comparison" output behavior and the CRTL_DO0 enable bit is set, you can use this bit to toggle the digital output DO0 on and off.
SET_DO1	Provided you have set the "No comparison" output behavior and the CRTL_DO1 enable bit is set, you can use this bit to toggle the digital output DO1 on and off.
SW_GATE	Set/reset this bit to open/close the SW gate.
T_CMP_V1	Set this bit to load the value of bytes 4 to 7 to comparison value 1.
T_CMP_V2	Set this bit to load the value of bytes 8 to 11 to comparison value 2.

Checkback interface for count modes (S7-300, S7-400)

Feedback interface for count modes

The table below shows the assignment of the FM 350-1 feedback interface (inputs) for the count modes.

Offset to the start address	Parameters	Meaning
Bytes 0 to 3	LATCH_LOAD	Load value which can be returned, or stored counter value for the latch function at the digital input
Bytes 4 to 7	ACT_CNTV	Counter value
Bytes 8 to 9	DA_ERR_W	Data error
Byte 10	OT_ERR_B	Operator error

Offset to the start address	Parameters	Meaning
Byte 11	PARA	Bit 7: Parameter assignment done
	FM_NEUST	Bit 6: Restart request
	FM_NEUSTQ	Bit 5: Restart acknowledgement done
	DATA_ERR	Bit 4: Data error
	OT_ERR	Bit 3: Operator error
	DIAG	Bit 2: Diagnostics event
	–	Bit 1: Reserve = 0
	–	Bit 0: Reserve = 0
Byte 12	–	Reserve = 0
Byte 13	STS_SW_GATE	Bit 7: Software gate status
	STS_GATE	Bit 6: Gate status
	STS_SYNC	Bit 5: Synchronization
	STS_UFLW	Bit 4: Violation of the low limit
	STS_OFLW	Bit 3: Violation of the high limit
	STS_ZERO	Bit 2: Zero pass
	STS_DIR	Bit 1: Direction bit
	STS_RUN	Bit 0: Counter active
Byte 14	STS_COMP2	Bit 7: Latched state of comparator 2
	STS_COMP1	Bit 6: Latched state of comparator 1
	STS_CMP2	Bit 5: Status at output DO1
	STS_CMP1	Bit 4: Status at output DO0
	STS_STP	Bit 3: Status at digital input DI-Stop
	STS_STA	Bit 2: Status at digital input DI-Start
	STS_LATCH	Bit 1: New latch value for isochronous mode
	STS_SET	Bit 0: Status at digital input DI-Set
Byte 15	–	Bit 7: Reserve = 0
	STS_C_DOPARA	Bit 6: Change function of DO0/DO1, hysteresis or pulse duration
	STS_RES_ZERO	Bit 5: Reset status bit for zero pass, violation of high and low limit, or comparator
		Bit 4: Reset synchronization status bit
	STS_RES_SYNC	Bit 3: Load comparison value 2
	STS_T_CMP_V2	Bit 2: Load comparison value 1
	STS_T_CMP_V1	Bit 1: Load counter in preparation
	STS_L_PREPAR	Bit 0: Direct and preparatory loading of counter
	STS_L_DIRECT	

Description of the feedback bits for the count modes

Feedback bit	Explanation
DATA_ERR	This bit indicates a faulty data (parameter assignment error) entry in the feedback interface.
DIAG	The bit will be set if diagnostics data record DS1 was updated to signal a diagnostics event. The bit will be reset when data record DS1 has been read. If no diagnostics interrupt is enabled, this bit may be used as init bit for the DIAG_INF instruction embedded in OB1.

Feedback bit	Explanation
FM_NEUST	FM 350-1 sets this bit each time it performs a restart or detects a system startup, regardless of whether the system starts up automatically or manually. The FM_NEUST bit is reset at the next positive edge at the NEUSTQ bit. FM 350-1 then accepts control commands and allows the input and output of values.
FM_NEUSTQ	FM 350-1 clears this bit each time it performs a restart or detects a system startup, regardless of whether the system starts up automatically or manually. It is set after you reset the FM_NEUST bit.
OT_ERR	This bit will be set after an operator error was logged at the feedback interface. It is reset when the OT_ERR_A bit is set. As long as OT_ERR bit is set, the function does not report any additional operator errors.
PARA	This bit will be set when the module parameters are free of errors. The parameter data record on the module does not contain any errors. This bit is only set when the FM_NEUSTQ bit is set. From this moment on, the values in the feedback interface are valid and up to date.
STS_C_DOPARA	Bit used to acknowledge simultaneous changes of the behavior of DO0 and DO1, and of the hysteresis and pulse duration. Transfer the old values if you want to discard all changes.
STS_CMP1	Status at output DO0
STS_CMP2	Status at output DO1
STS_T_CMP_V1	Bit used to acknowledge loading of comparison value 1
STS_T_CMP_V2	Bit used to acknowledge loading of comparison value 2
STS_COMP1	This bit indicates the stored status that output DO0 was set. That also applies if output DO0 was not enabled with CTRL_DO0. The saved status is reset by acknowledgment with RES_ZERO.
STS_COMP2	This bit indicates the stored status that output DO1 was set. That also applies if output DO1 was not enabled with CTRL_DO1. The saved status is reset by acknowledgment with RES_ZERO.
STS_DIR	This bit indicates the count direction of the counter: 0 = Up (LED DIR is off) 1 = Down (LED DIR is lit)
STS_GATE	This bit indicates the gate status. 0 = Gate closed 1 = Gate open
STS_LATCH	In isochronous mode, this bit indicates whether at least one new latch value was saved between the one to last Ti and the last Ti. If the bit is set, the last latch value is in LATCH_LOAD. The bit will not be set if no new latch value was saved. The bit is not set in non-isochronous mode.
STS_L_DIRECT	Acknowledgement bit for direct and preparatory loading of the counter and load value.
STS_L_PREPAR	Acknowledgement bit for preparatory loading of the load value.
STS_OFLW	This bit indicates a high limit violation. The saved status is reset by acknowledgment with RES_ZERO.
STS_RES_SYNC	Resets the STS_SYNC feedback bit.
STS_RES_ZERO	Acknowledgement bit for resetting saved states in the feedback bits STS_ZERO, STS_OFLW, STS_UFLW, STS_COMP1 and STS_COMP2
STS_RUN	This bit corresponds to counter bit 2^0. 0 = LED CR is off 1 = LED CR is lit
STS_SET	Status at digital input DI-Set
STS_STA	Status at digital input DI-Start
STS_STP	Status at digital input DI-Stop
STS_UFLW	This bit indicates a low limit violation. The saved status is reset by acknowledgment with RES_ZERO.

Feedback bit	Explanation
STS_SYNC	This bit indicates the saved state, which shows that the counter was loaded by an event at DI-Set (synchronization). The saved status is reset by acknowledgment with RES_SYNC.
STS_ZERO	This bit indicates the saved state, which shows that the counter value has passed a zero pass. The saved status is reset by acknowledgment with RES_ZERO.

Control interface for measuring modes (S7-300, S7-400)

Control interface for measuring modes

Parameter LOAD_VAL (bytes 0 to 3) has two meanings:

- If you set the L_PREPAR bit, LOAD_VAL is interpreted as a low limit.
- You set the C_DOPARA bit in byte 0 to define the behavior at output DO0.

The table below shows the assignment of the FM 350-1 control interface (outputs) for measuring modes.

Offset to the start address	Parameter	Assignment			
Bytes 0 to 3	LOAD_VAL	Load low limit with the L_PREPAR bit			
Byte 0		Define behavior of output DO0 with the C_DOPARA bit			
		Bits 2 to 7	Bit 1	Bit 0	Behavior of output DO0
		Irrelevant	0	0	No comparison
		Irrelevant	0	1	Outside limits
		Irrelevant	1	0	Violating low limit
		Irrelevant	1	1	Exceeding high limit
Byte 1		Reserve = 0			
Byte 2		Reserve = 0			
Byte 3		Reserve = 0			
Bytes 4 to 7	CMP_V1	High limit; load with the T_CMP_V1 bit			
Bytes 8 to 9	CMP_V2	Update time; load with the T_CMP_V2 bit			
Bytes 10 to 11	–	–			
Byte 12	–	Bit 7: Reserve = 0			
	NEUSTQ	Bit 6: Restart acknowledgement			
	–	Bit 5: Reserve = 0			
	–	Bit 4: Reserve = 0			
	OT_ERR_A	Bit 3: Operator error acknowledgement			
	–	Bit 2: Reserve = 0			
	–	Bit 1: Reserve = 0			
	–	Bit 0: Reserve = 0			

Offset to the start address	Parameter	Assignment
Byte 13	– – – – SW_GATE GATE_STP – –	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: SW gate control bit Bit 2: General gate stop Bit 1: Reserve = 0 Bit 0: Reserve = 0
Byte 14	– – – – SET_DO1 SET_DO0 CTRL_DO1 CTRL_DO0	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: Control bit DO1 Bit 2: Control bit DO0 Bit 1: Enable DO1 Bit 0: Enable DO0
Byte 15	– C_DOPARA[1)] RES_ZERO – T_CMP_V2[2)] T_CMP_V1[2)] L_PREPAR[2)] –	Bit 7: Reserve = 0 Bit 6: Change function DO0 Bit 5: Reset status bits for violation of high limit, low limit and end of measurement Bit 4: Reserve = 0 Bit 3: Change refresh time Bit 2: Load high limit Bit 1: Load low limit Bit 0: –
[1)] Do not set this bit at the same time as bit 1, 2 or 3 of byte 15. [2)] Do not set this bit at the same time as bit 6 of byte 15.		

Description of the control bits for measuring modes

Control bit	Explanation
C_DOPARA	Set this bit to change the behavior and function of DO0. The values at byte 0 are accepted as new function of DO0. Transfer the old values if you want to discard all changes.
CTRL_DO0	Enable DO0 Set this bit to enable output DO0.
CTRL_DO1	Enable DO1 Set this bit to enable output DO1.
GATE_STP	Set this bit to close the internal gate.
L_PREPAR	Set this bit to load the low limit.

Control bit	Explanation
NEUSTQ	Set this bit to acknowledge a start of the FM 350-1. After its restart, FM 350-1 does not recognize any control or data input unless this bit is set. The NEUSTQ bit is set by the CNT_CNTL1 instruction as soon as the feedback signal FM_NEUST is set and the feedback signal is FM_NEUSTQ = 0. It is reset by the CNT_CNTL1 instruction when the FM_NEUST bit is reset by the FM 350-1 and the FM_NEUSTQ bit has been set by the FM 350-1. If you do not use the CNT_CNTL1 instruction, you need to coordinate the restart via the user program.
OT_ERR_A	Set this bit to acknowledge an operator error. For detailed information on operator errors, read the feedback interface before you acknowledge the error. The error message is no longer valid after its acknowledgement.
RES_ZERO	Use this bit to reset the feedback bits STS_OFLW, STS_UFLW and STS_COMP1.
SET_DO0	Provided you have set the "No comparison" output behavior and the CRTL_DO0 enable bit is set, you can use this bit to toggle the digital output DO0 on and off.
SET_DO1	Provided you have set the "No comparison" output behavior and the CRTL_DO1 enable bit is set, you can use this bit to toggle the digital output DO1 on and off.
SW_GATE	Set/reset this bit to open/close the SW gate.
T_CMP_V1	Set this bit to load the high limit.
T_CMP_V2	Set this bit to load the refresh interval time.

Checkback interface for the measuring modes (S7-300, S7-400)

Feedback interface for the measuring modes

The table below shows the assignment of the FM 350-1 feedback interface (inputs) for measuring modes.

Offset to the start address	Parameter	Assignment
Bytes 0 to 3	LATCH_LOAD	Measured value
Bytes 4 to 7	ACT_CNTV	Counter value
Bytes 8 to 9	DA_ERR_W	Data error
Byte 10	OT_ERR_B	Operator error
Byte 11	PARA FM_NEUST FM_NEUSTQ DATA_ERR OT_ERR DIAG – –	Bit 7: Parameter assignment done Bit 6: Restart request Bit 5: Restart acknowledgement done Bit 4: Data error Bit 3: Operator error Bit 2: Diagnostics event Bit 1: Reserve = 0 Bit 0: Reserve = 0
Byte 12		Reserve = 0

Offset to the start address	Parameter	Assignment
Byte 13	– STS_GATE – STS_UFLW STS_OFLW STS_COMP1 STS_DIR STS_RUN	Bit 7: Reserve = 0 Bit 6: Gate status Bit 5: Reserve = 0 Bit 4: Low limit violated Bit 3: High limit violated Bit 2: End of measurement Bit 1: Direction bit Bit 0: Counter active
Byte 14	– – STS_CMP2 STS_CMP1 STS_STP STS_STA – STS_SET	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Status at output DO1 Bit 4: Status at output DO0 Bit 3: Status at digital input DI-Stop Bit 2: Status at digital input DI-Start Bit 1: Reserve = 0 Bit 0: Status at digital input DI-Set
Byte 15	– STS_C_DOPARA STS_RES_ZERO – STS_T_CMP_V2 STS_T_CMP_V1 STS_L_PREPAR –	Bit 7: Reserve = 0 Bit 6: Change function DO0 Bit 5: Reset end of measurement status bit Bit 4: Reserve = 0 Bit 3: Change refresh time Bit 2: Load high limit Bit 1: Load low limit Bit 0: Reserve = 0

Description of the acknowledgement bits for the measuring modes

Feedback bit	Explanation
DATA_ERR	This bit indicates that a data error was entered in the feedback interface.
DIAG	The bit will be set if diagnostics data record DS1 was updated to signal a diagnostics event. The bit will be reset when data record DS1 has been read. If no diagnostics interrupt is enabled, this bit may be used as init bit for the DIAG_INF instruction embedded in OB1.
FM_NEUST	FM 350-1 sets this bit each time it performs a restart or detects a system startup, regardless of whether the system starts up automatically or manually. The FM_NEUST bit is reset at the next positive edge at the NEUSTQ bit. FM 350-1 then accepts control commands and allows the input and output of values.
FM_NEUSTQ	FM 350-1 clears this bit each time it performs a restart or detects a system startup, regardless of whether the system starts up automatically or manually. It is set after you reset the FM_NEUST bit.
OT_ERR	This bit will be set after an operator error was logged at the feedback interface. It is reset when the OT_ERR_A bit is set. As long as OT_ERR bit is set, the function does not report any additional operating errors.
PARA	This bit will be set when the module parameters are free of errors. The parameter data record on the module does not contain any errors. This bit is only set when the FM_NEUSTQ bit is set. From this moment on, the values in the feedback interface are valid and up to date.

Feedback bit	Explanation
STS_C_DOPARA	Bit used to acknowledge simultaneous changes of the behavior of DO0 and DO1, and of the hysteresis and pulse duration. Transfer the old values if you want to discard all changes.
STS_CMP1	Status at output DO0
STS_CMP2	Status at output DO1
STS_CMP_T_VAL1	Acknowledgment bit for loading the high value
STS_CMP_T_VAL2	Acknowledgement bit for loading of the refresh time
STS_DIR	This bit indicates the count direction of the counter: 0 = Up (LED DIR is off) 1 = Down (LED DIR is lit)
STS_GATE	This bit indicates the gate status. 0 = Gate closed 1 = Gate open
STS_L_PREPAR	Acknowledgment bit for loading the low value
STS_OFLW	This bit indicates the saved status, which shows that a measured value is greater than the high limit. The saved status is reset by acknowledgment with RES_ZERO.
STS_RES_ZERO	Acknowledgement bit for resetting saved states in the feedback bits STS_OFLW, STS_UFLW and STS_COMP1
STS_RUN	This bit corresponds to counter bit 2^0. 0 = LED CR is off 1 = LED CR is lit
STS_SET	Status at digital input DI-Set
STS_STA	Status at digital input DI-Start
STS_STP	Status at digital input DI-Stop
STS_UFLW	This bit indicates the saved status, which shows that a measured value is less than the low limit. The saved status is reset by acknowledgment with RES_ZERO.

Using FM 350-1 from the user program (S7-300, S7-400)

Integrating the FM 350-1 into a user program

Instructions are available for integrating the FM 350-1 into a user program. These enable you to handle the desired functions with ease.

The instructions CNT_CTRL, CNT_CTL1 and CNT_CTL2 can be called in the CPU user program. They ensure communication between the CPU and the FM 350-1.

All data associated with a channel of the module are in the DB of the instruction CNT_CTRL, CNT_CTL1 or CNT_CTL2 (counter DB). Use the command "Program blocks > Add new block" to add a CNT_CHANTYPE1 type data block to your project.

The DIAG_INF instruction is also available for FM 350-1. This operation allows you to transfer the diagnostic information to the counter DB.

The instruction CNT_CTL2 is only used in isochrone mode.

FM 350-1: Online & Diagnostics (S7-300, S7-400)

Current module information (S7-300, S7-400)

Default diagnostics (S7-300, S7-400)

FM 350-1 group error LED fault display (S7-300, S7-400)

Where is the Fault Indicated?

If the red group error LED lights up, a fault has occurred either on the module (internal fault) or at the cable connections (external fault).

Which Errors Are Displayed?

The table below shows the error types displayed by the group error LED.

Type of error	Cause of the error	Correction
Internal faults	Error in EPROM test	Module replacement
	RAM test error	Module replacement
	Watchdog tripped	Module replacement
	Lost hardware interrupt	Increase time between the interrupt causes
	Module parameterization missing	Assign parameters and transfer
External errors	Coding plug incorrectly connected	Correct the position of the coding plug
	Auxiliary voltage 1L+/1M not connected or DC24V encoder supply short-circuited	Correct connection
	DC5.2V encoder supply short-circuited or overloaded	Correct connection
	Fault in 5V encoder signals (wire break, short circuit, cable missing)	Correct connection
	Module parameterization does not match the position of the coding plug	Correct parameterization and transfer, or reconnect the coding plug

Initiating a Diagnostics Interrupt

All errors except an EPROM test error can trigger a diagnostic interrupt provided you have enabled the diagnostic interrupt in the basic parameters during parameter assignment. The diagnostic data records DS0 and DS1 (Page 256) will show you which fault has caused the LED to light up.

Channel-specific diagnostics (S7-300, S7-400)

Triggering diagnostics interrupts (S7-300, S7-400)

Definition of a diagnostics interrupt

You can determine reactions to internal or external errors in the user program by configuring a diagnostic interrupt which interrupts cyclic program execution on the CPU and calls diagnostic interrupt OB.

Events which can trigger a diagnostics interrupt

The following events can trigger a diagnostic interrupt:

- Faults in the external auxiliary 1L+/1M voltage.
- 5.2 V DC encoder supply short circuit or overload
- No module configuration
- Faulty module parameters
- Time monitoring (watchdog) triggered
- RAM defective
- Hardware interrupt lost
- Signal A error (wire break, short-circuit, cable missing)
- Signal B error (wire break, short-circuit, cable missing)
- Signal N error (wire break, short-circuit, cable missing)
- Coding plug incorrectly connected

Enabling diagnostics interrupts

When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.

Reactions to a diagnostics interrupt

When an event occurs that could trigger a diagnostic interrupt:

- The diagnostics information is stored in the diagnostic data records DS0 and DS1 (Page 256) in the module.
- The group error LED is lit.
  The group error LED does dark when the error is eliminated.
- The diagnostic interrupt OB (e.g. OB 82) is called.

- The diagnostic data record DS0 is entered in the start information of the diagnostic interrupt OB.
- The count continues unchanged if a diagnostic interrupt OB has been programmed. If no diagnostic interrupt OB has been programmed, the CPU goes to STOP.

Default setting

No diagnostic interrupts are enabled in the default setting.

Diagnostics data record DS0 and DS1 (S7-300, S7-400)

Diagnostic data record DS0 and DS1

The information showing the event which has triggered a diagnostics interrupt is written to the diagnostics data records DS0 and DS1. The diagnostic data record DS0 comprises four bytes. DS1 comprises 16 bytes; the first four bytes are identical to DS0.

Reading data records from the module

Diagnostic data record DS0 (Page 257) is automatically transferred to the start information when the diagnostic OB is called. These four bytes are written to the local date (bytes 8 to 11) of the diagnostic interrupt OB.

You can read out the diagnostic data record DS1 (Page 258) (and thus also the content of the DS0) from the module with the instruction DIAG_INF. However, this is only useful if DS0 reports a channel error.

Assignments of diagnostics data record DS0 (S7-300, S7-400)

Assignments of diagnostics data record DS0 in the start information

The following table shows the assignment of the diagnostic data record DS0 in the start information. Any bits not listed are insignificant and are zero.

Byte	Bit	Meaning	Comments	Event no.
0	0	Module in error state	Set upon all diagnostics events	8:x:00
	1	Internal error	Set at all internal error events: • RAM test error • Time monitoring (watchdog) triggered • Lost hardware interrupt	8:x:01
	2	External error	Set at all external error events: • Auxiliary voltage 1L+/1M not connected or short-circuit in the 5.2 V DC encoder supply. • 5.2 V DC encoder supply short circuit or overload • 5 V signal faults • Parameter error	8:x:02
	3	Channel error	See DS1, byte 4 (Page 258) for a more detailed breakdown	8:x:03
	4	Failure of the external auxiliary voltage	Check the voltage	8:x.04
	6	No configuration	Assign parameters	8:x:06
	7	Parameter error	See Data errors (Page 258) for a more detailed breakdown	8:x:07
1	0...3	Type class	Always assigned the value 8	
	4	Channel information	Always assigned the value 1	
2	3	Time monitoring (watchdog) triggered	Module defective, or heavy interference	8:x:33
3	3	RAM defective	Module defective, or heavy interference	8:x:43
	6	Hardware interrupt lost	Check the configuration. A hardware interrupt was detected and can not be reported, because the same event is not yet acknowledged by the CPU	8:x:46

Assignment of diagnostic data record DS1 (S7-300, S7-400)

Assignment of diagnostic data record DS1

Diagnostic data record DS1 consists of 16 bytes. The first 4 bytes are identical to diagnostic data record DS0. The following table shows the assignment of the remaining bytes. Any bits not listed are insignificant and are zero. The instruction DIAG_INF writes this data record to the DB of the instruction CNT_CTL1, starting at DW54.

Byte	Bit	Meaning	Comments	Event no.
4	0...6	Channel type	Always assigned the value 76H	
	7	Further channel types	Always assigned zero value	
5	0...7	Diagnostics data length	Always assigned the value 10H	
6	0...7	Number of channels	Always assigned the value 1	
7	0	Channel error vector	Assigned 1 on channel error	
8	0	Signal A error		8:x:B0
	1	Signal B error		8:x:B1
	2	Signal N error		8:x:B2
	4	Error at the 5.2-V encoder supply		8:x:B4
	5...7	Reserved		
9 ... 15		Reserved		

Data error (S7-300, S7-400)

Data error events

FM 350-1 checks all new parameters transferred to the module. The module reports any errors returned in this check.

Where data errors are indicated

Instruction CNT_CTL1 enters the data errors and the error numbers in the CNT_CTL1 DB. You can access this data word in the user program using the DA_ERR_W variable.

What data errors are possible?

The table below lists the data errors and their error numbers and meaning.

No.	Meaning
0	No error
200	Coding plug in wrong position or missing
201	The position of the coding plug does not match the programmed encoder
202	Invalid diagnostics value of the signal pair
203	Incorrect value for signal evaluation
204	Incorrect 24 V count signal input filter value

No.	Meaning
205	Invalid value at digital input filters
206	Reversal of direction not allowed
207	Incorrect configuration of the reaction of DO0
208	Incorrect configuration of the reaction of DO1
209	Pulse duration out of limits
211	Wrong operating mode selected
212	No gate or both gates defined
213	Faulty parameters of main count direction
214	Count high limit violated
215	A different count direction was selected for configuring output behavior "Active on reaching the comparison value for pulse duration up or down" in parameter assignment for the hardware interrupt "Reaching the comparison value up or down". The directions set must match.
216	Gate control interrupts are only possible in modes with gate control.
217	Interrupt on "Reaching comparison values" is not valid for output behavior "On when counter value >= comparison value" or "On when counter value <= comparison value".
218	Interrupt triggering at the zero transition is not allowed
219	Incorrect coding of the "Latch" setting
220	Faulty gate control parameters
221	Low limit or load value violated
222	High limit or comparison value 1 exceeded
223	Update time or comparison value 2 exceeded
224	Pulses per encoder revolution out of limits

How to acknowledge data errors

Correct the parameter values according to specifications. Transfer the corrected parameter set to the FM 350-1 again. It will recheck the parameters and delete the data error in the DB.

Operator error (S7-300, S7-400)

Operator errors events

Operator errors develop as a result of improper operation of the module caused by the incorrect input of control signals.

Objects which indicate operator errors

The instruction CNT_CTL1 enters the operator error numbers in the DB. Instruction CNT_CTL1 sets output parameter OT_ERR to indicate an operator error has occurred. You can access this data byte in your program using the OT_ERR_B variable.

What operator errors are possible?

The table below lists the operator errors and their error numbers and meaning.

No.	Meaning
0	No error
1	Operating mode can not be started using the SW gate
2	Operating mode can not be canceled
4	Only allowed when the CPU is in STOP
5	Only the parameter assignment control bit may be set
6	Illegal job
10	Low limit or load value violated
11	High limit or comparison value 1 exceeded
12	Update time or comparison value 2 exceeded
20	Incorrect configuration of the reaction of DO0
21	Incorrect configuration of the reaction of DO1
22	Pulse duration out of limits
90	See CNT_CTL2
91	See CNT_CTL2

How to acknowledge operator errors

You acknowledge the error with the input parameter OT_ERR_A of instruction CNT_CTL1 or CNT_CTL2.

1.2.1.3 Using FM 350-2 (S7-300, S7-400)

FM 350-2 front connector terminal assignment (S7-300, S7-400)

Front connector

You connect the following to the 40-pin front connector:

- Count signals
- Digital inputs
- Digital outputs
- Encoder supply and module power supply

The figure below shows the front of the module, the front connector and the inside of the front panel with terminal assignment.

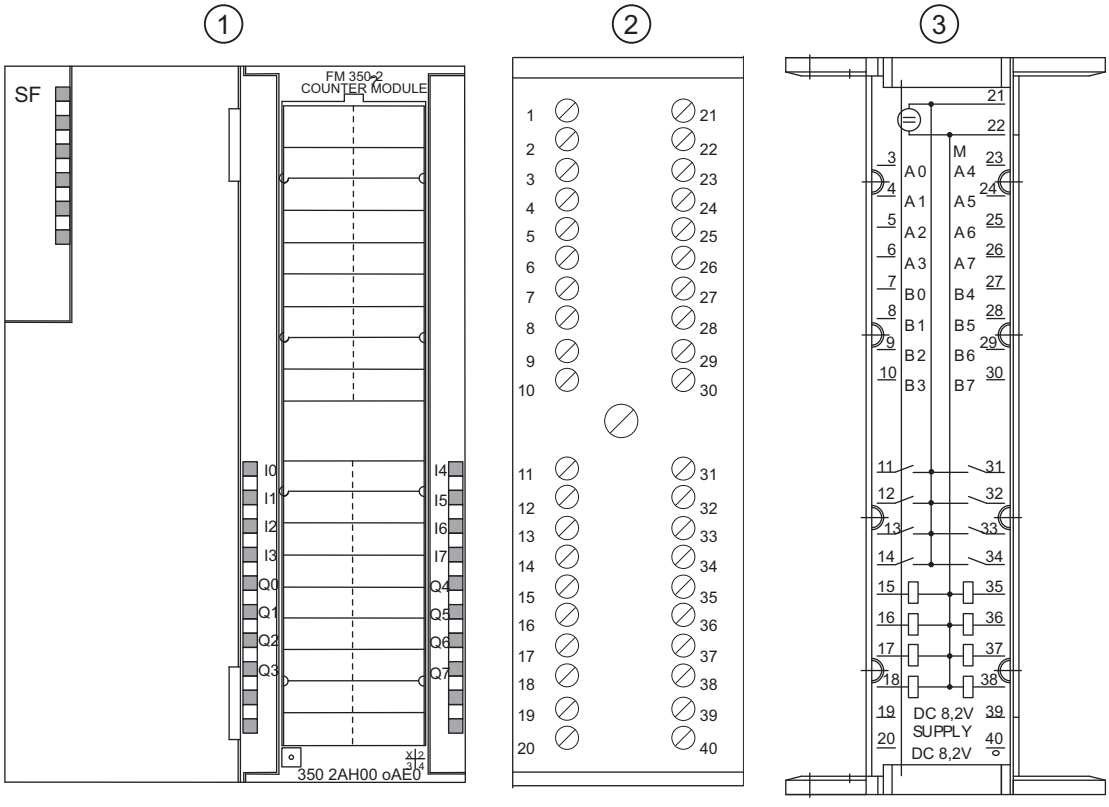

① Front of the module
② Front connectors
③ Inside of the front panel cover

Front Connector Assignments

The following table shows the terminal assignment for the front connector of the FM 350-2.

Terminal	Name	Input/output	Function
1	-	-	Not connected
2	-	-	Not connected
3	A0	Input	Channel 0 count input NAMUR / BERO
4	A1	Input	Channel 1 count input NAMUR / BERO
5	A2	Input	Channel 2 count input NAMUR / BERO
6	A3	Input	Channel 3 count input NAMUR / BERO
7	B0	Input	Channel 0 direction input BERO
8	B1	Input	Channel 1 direction input BERO
9	B2	Input	Channel 2 direction input BERO
10	B3	Input	Channel 3 direction input BERO
11	I0	Input	Channel 0 hardware gate input BERO
12	I1	Input	Channel 1 hardware gate input BERO
13	I2	Input	Channel 2 hardware gate input BERO
14	I3	Input	Channel 3 hardware gate input BERO

Terminal	Name	Input/output	Function
15	Q0	Output	Channel 0 digital output 0.5 A
16	Q1	Output	Channel 1 digital output 0.5 A
17	Q2	Output	Channel 2 digital output 0.5 A
18	Q3	Output	Channel 3 digital output 0.5 A
19	P8V2	Output	8.2 V NAMUR encoder supply
20	P8V2	Output	8.2 V NAMUR encoder supply
21	L+	Input	24 V module supply
22	M	Input	Ground module supply
23	A4	Input	Channel 4 count input NAMUR / BERO
24	A5	Input	Channel 5 count input NAMUR / BERO
25	A6	Input	Channel 6 count input NAMUR / BERO
26	A7	Input	Channel 7 count input NAMUR / BERO
27	B4	Input	Channel 4 direction input BERO
28	B5	Input	Channel 5 direction input BERO
29	B6	Input	Channel 6 direction input BERO
30	B7	Input	Channel 7 direction input BERO
31	I4	Input	Channel 4 hardware gate input BERO
32	I5	Input	Channel 5 hardware gate input BERO
33	I6	Input	Channel 6 hardware gate input BERO
34	I7	Input	Channel 7 hardware gate input BERO
35	Q4	Output	Channel 4 digital output 0.5 A
36	Q5	Output	Channel 5 digital output 0.5 A
37	Q6	Output	Channel 6 digital output 0.5 A
38	Q7	Output	Channel 7 digital output 0.5 A
39	P8V2	Output	8.2 V NAMUR encoder supply
40	P8V2	Output	8.2 V NAMUR encoder supply

Note

The circuits for the counter inputs (encoder supply, encoder signals) are isolated from the ground of the CPU.

All inputs are not isolated from each other but are isolated from the S7 300 bus.

You will find more information on structure and wiring in the Commissioning and Hardware Installation Manual (http://support.automation.siemens.com/WW/view/en/1105178) for the FM 350-2 counter module.

24 V voltage supply

Connect a direct voltage of 24 V to the L+ and M terminals for the FM 350-2 voltage supply.

8.2 V DC encoder power supply

The module generates a voltage of 8.2 V (max. 200 mA) from the 24 V supply voltage. This voltage is available at the terminals P8V2 (pins 19, 20, 39, and 40) for the voltage supply to the NAMUR encoders and is resistant to short circuits.

The encoder supply is monitored for 8.2 V.

Encoder Signals A0 to A7, B0 to B7

You can connect four different types of encoder:

- NAMUR encoders pursuant to DIN 19234 (with diagnostics):
  The signals are wired to terminals A0 to A7.
- 24 V incremental encoder:
  The signals A0/B0 to A7/B7 are connected via the terminals so labeled.
- 24 V pulse encoder with direction level:
  The count signals are wired to terminals A0 to A7. The directional signals are wired to terminals B0 to B7.
- 24 V initiators:
  The signals are connected to terminals A0 to A7.

Note

An external 24 V DC voltage supply line must be used for the 24 V encoder supply.

Digital Inputs I0 to I7 (Hardware Gates)

You can use the digital inputs I0 to I7 for the gate control of the counter.

One digital input is available for each count channel with which you can start and stop the corresponding counter.

The digital inputs are operated at a rated voltage of 24 V.

Digital Outputs Q0 to Q7

The FM 350-2 can trigger control processes directly through digital outputs Q0 to Q7.

One digital output is available per counter.

The digital outputs are powered by the FM 350-2 24 V voltage supply.

The digital outputs are current-sourcing switches and support a load current of 0.5 A. These outputs are protected against overload and short circuit-proof.

Note

Relays and contactors can be connected directly without external circuitry.

Configuring FM 350-2 (S7-300, S7-400)

Configuring FM 350-2 (S7-300, S7-400)

Specifying channels (S7-300, S7-400)

Selecting channel settings (S7-300, S7-400)

Channel setting

This is the field in which you set the function of the module channels:

- Single counter: Each channel in the selected group operates as a separate counter.
- Dosing counter: 4 counter channels are combined to form one dosing channel. This channel can control up to 4 dosing units.

You can make the following selections:

- Channels 0 to 7 as single counters
- Channel 0 as dosingcounter, channels 4..7 as single counters
- Channels 0..3 as single counters, channel 4 as dosingcounter
- Channel 0 as dosing counter and channel 4 as dosing counter

Enabling counter input for NAMUR encoders (S7-300, S7-400)

NAMUR encoder

This is the field in which you define the counter inputs to which a NAMUR encoder is to be connected.

You can make the following selections:

- Counter input channel 0 to 3 NAMUR
  and/or
- Counter input channel 4 to 7 NAMUR

See also: "NAMUR signals (Page 264)" and "Enabling hardware monitoring for NAMUR encoders (Page 271)"

NAMUR Signals (S7-300, S7-400)

NAMUR encoder

The encoder supplies one count signal in accordance with DIN 19234 and this must be connected to terminal A0 to A7 of the front connector.

You can wire a directional signal (24 V) to terminal B0 to B7 of the relevant counter. If your encoder does not return a corresponding signal, you can also generate and interconnect a relevant identifier within the S7 system, or use a suitable process signal.

The diagram below shows the time profile of a NAMUR encoder with directional level.

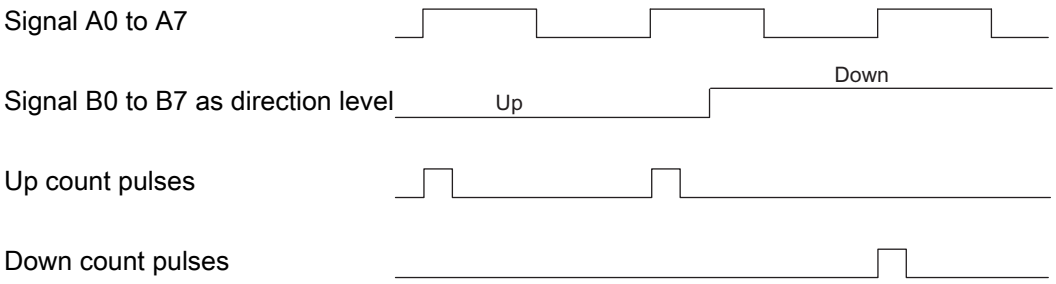

You can enable inputs A0 to A3, A4 to 7, or all inputs for NAMUR encoder connections.

How are the Signals Monitored?

You can assign parameters for the FM 350-2 so the module monitors A0 to A7 for wire break and short circuit. This also indirectly results in the 8.2 V encoder supply being monitored. This is only possible with NAMUR encoders.

If a wire break or short circuit is detected, the group error LED lights up, you can also assign a diagnostic interrupt to be triggered.

CAUTION

Danger of property damage.

Using another encoder on a FM 350-2 channel which was parameterized for the connection of a NAMUR encoder can damage the module.

You should only connect NAMUR encoders to FM 350-2 channels parameterized for the connection of a NAMUR encoder.

Selecting feedback interface assignment (S7-300, S7-400)

Feedback interface assignment, bytes 8 to 15

This is the field in which you select how bytes 8 to 15 of the feedback interface are assigned.

The following options are available for variables USER_STAT_WORD_0 to USER_STAT_WORD_3:

- Data type
  - WORD
  - DWORD
- Channel
  - Channels 0 to 7
- Value
  - Counter value
  - Measured value
- Word
  - Either the low word or the high word of a value or both can be stored. You have direct access to these values.

Mapping the count and measurement values of all channels in the expanded user data (S7-300, S7-400)

Mapping of all count and measurement values in the user data image

Under certain conditions, all count and measurement values can be mapped in the user data image (use of expanded user data).

The following basic conditions apply to use of expanded user data:

- Configure the module "6ES7350-2AH01-0AE0 E" from the module catalog.
- The check box "Use extended user data interface" must be selected.
- The extended user data can only be used on a distributed basis behind an IM 153-2 V5.0 (article number 6ES7153-2BA01-0XB0 V5.0) or higher.
- 40 bytes of input data are now available.
- The update cycle of the extended user data differs from the update cycle of the first 16 bytes of input data. The update cycle of the extended user data is approximately 100 to 200 ms depending on the PROFIBUS DP cycle time.

The structure of the first 16 bytes in the user data is the same as before in each operating mode. The additional 24 bytes are defined by the operating mode.

Depending on the operating mode, the values (count or measurement value) of channels 2 to 7 are stored in the expanded user data. These values are preassigned to "0". The input data are occupied with 0 as long as valid values have not been detected.

Structure of the extended user data as defined by the operating mode:

Byte	Counting modes	Measuring modes	Dosing mode (two dosing counters)
0..15	Same as previous – structure can be influenced via parameter assignment	Same as previous – structure can be influenced via parameter assignment	Same as previous – structure can be influenced via parameter assignment
16..19	Count value channel 2	Measurement value channel 2	0
20..23	Count value channel 3	Measurement value channel 3	0
24..27	Count value channel 4	Measurement value channel 4	Dosing counter channel 4
28..31	Count value channel 5	Measurement value channel 5	0
32..35	Count value channel 6	Measurement value channel 6	0
36..39	Count value channel 7	Measurement value channel 7	0

Copying channels (S7-300, S7-400)

Copying channels

You can copy the parameters of one channel to one or more other channels.

1. Select the channel whose parameters are to be copied (source channel).
2. Select the channels to which the parameters are to be copied (target channel).
3. Press the "Copy" button.

Channel n operating modes (S7-300, S7-400)

Selecting an operating mode (S7-300, S7-400)

Description

You define counter module functionality by specifying an operating mode.

Mode	Description
Count continuously (Page 182) (with or without gate)	The counter module counts from the current counter reading continuously.
Count once (Page 183) (with hardware or software gate)	When the gate opens, the counter module counts from zero up to the parameterized high count limit or from the parameterized high count limit to zero.

Mode	Description
Count periodically (Page 184) (with hardware or software gate)	The counter module counts between zero and the parameterized high count value when the gate is opened.
Frequency measurement (Page 191) (with hardware or software gate)	The counter module detects the incoming pulses in a parameterized time window of n x 10 ms and uses this data to determine the current frequency.
Rotational speed measurement (Page 193) (with hardware or software gate)	The counter module detects the incoming pulses in a parameterized time window of n x 10 ms and determines the speed on the basis of the parameterized number of pulses per encoder revolution.
Period measurement (Page 194) (with hardware or software gate)	The counter module detects the incoming pulses in a parameterized time window of n x 10 ms and uses this data to determine the period.
Dosing (Page 197) (with hardware or software gate)	In this mode, four count channels are grouped together to form a dosingchannel. This mode functions in exactly the same way as the "Count once" mode.

The default setting is the "Count continuously" operating mode.

Specifying the high count limit (S7-300, S7-400)

Description

By configuring an high count limit together with a main count direction (up or down), you can reduce the maximum counting range. The parameterized counting range is then between 0 and the high count limit parameterized.

The high count limit can be any value between > 1 and the high count range limit.

Specifying measuring time (S7-300, S7-400)

Description

If you have selected a measuring mode, the module will update the measured values on a cyclic basis. The integration time is the product of the measuring time x 10ms. You set the measuring time as the factor n.

Detailed descriptions of the value ranges for the measuring time can be found under "Measuring basics (Page 187)".

Using a hardware gate (S7-300, S7-400)

Using a hardware gate

You can decide when or not the channel is to count with a hardware gate.

If counting is to be done with a hardware gate, check the check box "Use hardware gate".

Selecting gate function (S7-300, S7-400)

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 202): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 202): The counting process starts from the last counter value after gate stop and restart.

Selecting main count direction (S7-300, S7-400)

Description

By configuring a main count direction (Page 185) (up or down), you can limit the maximum counting range by setting an high count limit. The parameterized counting range is then between 0 and the high count limit parameterized.

A main count direction can only be selected in the "Count once", " Count periodically" and "Dosing" modes.

You can select the following behavior:

- Up
- Down

Channel n encoder (S7-300, S7-400)

Selecting the encoder signal type (S7-300, S7-400)

Selecting the signal type

1. Select the type of signal for the encoder used.
   In line with your selection and the operating mode, any option not possible for this specific encoder will be disabled. A disabled option is identified by grayed font in the corresponding selection window.
2. Select the options to be used with the selected encoder.

You can select the signal type for one of the following encoders:

- Encoder: 24 V differential signal (Page 270) (24 V track A+B phase shifted)
- Encoder: 24 V pulse and direction (Page 270)

Encoder: 24 V incremental (S7-300, S7-400)

24 V incremental encoder

Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A* and B*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation (Page 206)	Single	Only one edge of signal A is evaluated.
	Double	Both edges of signal A are evaluated.
	Quadruple	Both edges of signal A, and B are evaluated.
Count direction (Page 270)	Normal	To adapt the count direction to the process
	Inverted	To adapt the count direction to the process

Encoder: 24 V pulse and direction (S7-300, S7-400)

24 V pulse generator

Select signal type "24 V pulse and direction" to use a 24 V pulse generator with count signal A* and direction signal B*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation	–	Not relevant
Count direction (Page 270)	Inverted	To adapt the count direction to the process
	Normal	To adapt the count direction to the process

Selecting the count direction (S7-300, S7-400)

Changing the count direction

You can change the count direction using the "Count direction normal" and "Count direction inverted" parameters without having to modify the wiring.

Assigning the parameter "Main count direction down (Page 185)" does **not** automatically reverse the count direction.

Entering pulses per encoder revolution (S7-300, S7-400)

Pulses per encoder revolution

For "Rotational speed measurement (Page 193)" mode, you must also configure the pulses per encoder revolution.

The value for the number of pulses per encoder revolution can be found on the rating plate, or in the technical data of your encoder.

Setting hysteresis (S7-300, S7-400)

Function principle of the hysteresis

An encoder can stop at a certain position and then "oscillate" around this position. This state causes the counter level to fluctuate around a specific value. If there is a comparison value within this fluctuation range, for example, the associated output is switched on and off in accordance with the rhythm of the fluctuations. The FM 350-2 has configurable hysteresis to prevent this switching in response to small fluctuations.

Values between 0 and 255 can be set for hysteresis.

The effect of hysteresis is as follows in accordance with the value parameterized:

Hysteresis	Effect
Hysteresis value n = 0, 1	The hysteresis has no effect (switched off) The output reacts to the slightest change in the counter value.
2 ≤ Hysteresis value n ≤ 255	The hysteresis takes effect. The output does not react until the counter value is offset by n units to the comparison value.

The hysteresis applies to both high limit violated and low limit violated.

Enabling hardware monitoring for NAMUR encoders (S7-300, S7-400)

How are the signals from NAMUR encoders monitored?

FM 350-2 can monitor A0 to A7 for wire break and short circuit. This also indirectly results in the 8.2 V encoder supply being monitored.

Select the check box "Hardware monitoring for NAMUR encoders" if FM 350-2 is to monitor the encoder signals.

Channel n outputs (S7-300, S7-400)

Comparison values and outputs (S7-300, S7-400)

Description

You can store eight comparison values on the FM 350-2, each for one of the eight digital outputs (comparison value 0: Q0, comparison value 1: Q1, etc.). The relevant output can be set depending on the counter level and the comparison value.

You can assign four comparison values for each dosing channel of the FM 350-2 in "Dosing (Page 197)" mode.

Comparison values

The comparison values are entered during parameter assignment.

When the CPU is in RUN mode you can enter the comparison values in the data block of the CNT2_WR / CNT2WRPN instruction (CMP_VAL0…7) and transfer them to the FM 350-2 using jobs 30 to 37 or 60 to 62. The count is not affected by this.

The comparison values must lie within the count range of the respective operating mode; the count range limits cannot be used.

If you want to set the comparison value to the initial or final value in an operating mode with a main count direction, the digital outputs will behave as follows:

Parameterized behavior of digital outputs	Response of assigned digital outputs
On when counter value >= comparison value	The output is always switched on irrespective of the counter value
On when counter value <= comparison value	The output is always switched off irrespective of the counter value

Enabling the Outputs

Before the outputs can be set, you must first enable them by setting the relevant bits in the counter DB. If you reset one of these bits, the associated output will be switched off immediately. The bits are transferred between the data block and the module by the CNT2_CTR instruction.

Output	...is enabled by
Q0…7	CTRL_DQ0…7

Output switch-off

The outputs are switched off by the following events regardless of the parameter assignment:

- Module watchdog triggered (internal error)
- Removal of the enable bits (CTRL_DQ0...7 for Q0...7 in the DB)

Controlling the outputs

Irrespective of output switching via the comparators, you can set and reset the outputs when they are enabled by CTRL_DQ0…7 using the SET_DQ0…7 control signals.

The following applies to the relationship between controlling and switching: Controlling has priority over switching using the comparators. This means:

- The output can be used as a digital output if switching using the comparator is not set for the output.
- You can continue to control the outputs using SET_DQ0…7 if you have selected a compare function for the outputs. You can use this setup in the user program to simulate the effect of the compare functions:
  - The output is set with the positive edge of SET_DQ0…7.
  - A SET_DQ0…7 negative edge resets the output.

  Note that the comparators are still active and can set or reset the output if the comparison result changes.

Note

An output set with SET_DQ0…7 cannot be reset with the comparator.

Behavior of the outputs in counting modes (S7-300, S7-400)

Specifying the behavior of the outputs (S7-300, S7-400)

Behavior of the outputs DQ0 to DQ7

Select the required behavior for the output in question. Activate the corresponding option button:

- No comparison (Page 274)
- On when counter value >= comparison value (Page 274)
- On when counter value <= comparison value (Page 274)

Note

To activate the settings, enter values for comparison values in the DB with the counter data and transfer them to the module with the instruction CNT2_WR / CNT2WRPN; you must also set the CTRL_DQ0…7 bits in the DB in order to enable the outputs. See also "Comparison values and outputs (Page 271)".

Note

You cannot configure the outputs in "Frequency Measurement", "Rotational Speed Measurement" or "Period Duration Measurement" mode.

Output default setting

The default setting for the outputs is deactivated.

No comparison (S7-300, S7-400)

No comparison

The output remains deactivated and is not influenced by the events comparison value, zero pass, jump from final value to initial value (high limit violated) or jump from initial value to final value (low limit violated).

On when counter value >= comparison value (S7-300, S7-400)

On when counter value >= comparison value

The output is set to 1 if the counter is in the range between the comparison value and the final value.

Setting the counter to a value between the comparison value and final value sets the output to 1.

The hatched area in the graphic below means that the output is set in this range.

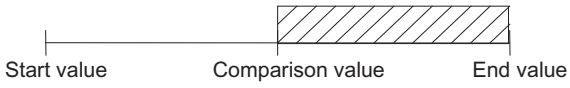

Constraint:

The following constraint applies if you want to configure an output "On when counter value >= comparison value": The time between reaching the comparison value and the initial or final value must be longer than the outputs' minimum switching time (switching time: 300 µs). The control pulses at the outputs will otherwise be lost.

See also

Effect of the hysteresis (Page 275)

On when counter value <= comparison value (S7-300, S7-400)

On when counter value <= comparison value

The output is set to 1 if the counter is in the range between the comparison value and the initial value.

Setting the counter to a value between the comparison value and initial value sets the output to 1.

The hatched area in the graphic below means that the output is set in this range.

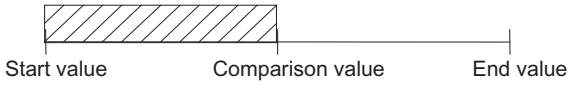

Constraint:

The following constraint applies if you want to configure an output "On when counter value <= comparison value": The time between reaching the comparison value and the initial or final value must be longer than the outputs' minimum switching time (switching time: 300 µs). The control pulses at the outputs will otherwise be lost.

See also

Effect of the hysteresis (Page 275)

Effect of the hysteresis (S7-300, S7-400)

Function of hysteresis with parameter assignment "On when counter value >= comparison value or <= comparison value"

The example in the figure below demonstrates the effect of hysteresis. The diagram shows the differences between the behavior of an output at a set hysteresis of 0 (= switched off) and a hysteresis of 3. In the example, the comparison value = 5

Parameter assignment in this example is as follows:

- Main count direction: "Up"
- Output response: "On when counter value >= comparison value"

The hysteresis is set when the comparison condition (counter value = 5) is fulfilled. When the hysteresis is active, the comparison result remains unchanged.

If the count value overshoots/undershoots the range of the hysteresis (at counter value 2 or 8 in the example) the hysteresis ceases to be active. The comparator switches again according to its comparison conditions, i.e., in the example at counter value 5.

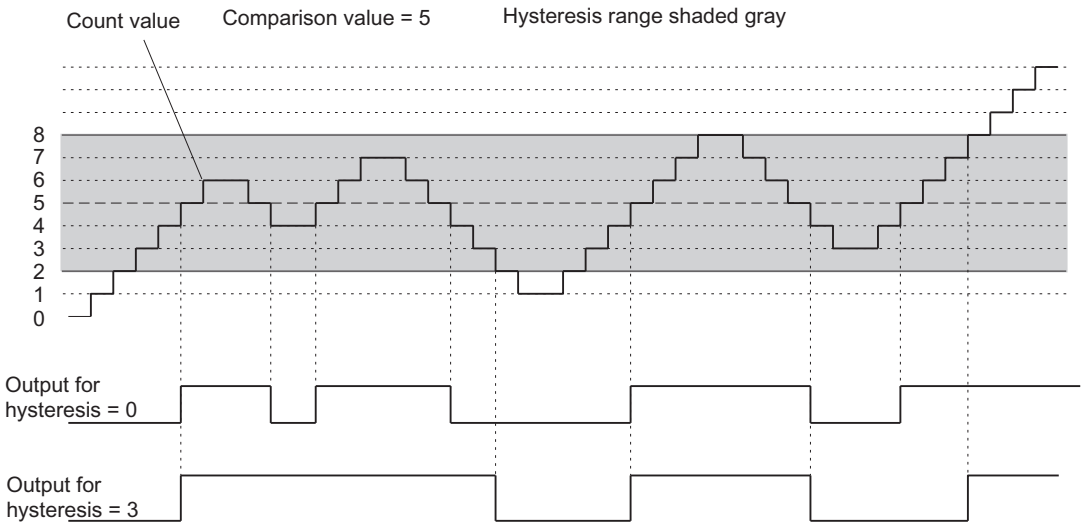

Note

If the counter value is equal to the comparison value, and hysteresis is active, FM 350-2 resets the output when the count direction changes at the comparison value (see the diagram below).

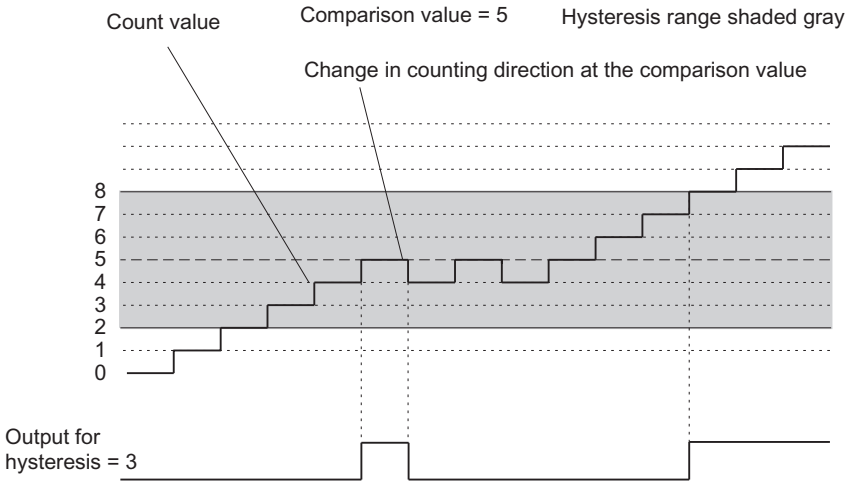

Enabling substitute value '1' at CPU STOP (S7-300, S7-400)

Substitute value "1" at CPU STOP

If you selected the "Substitute value" option under "Reaction to CPU STOP" in the module's basic parameters and the CPU goes into STOP, the module will return the parameterized substitute values at the digital outputs. Check the checkbox for DQ0, for example, and the output Q0 will be set at CPU STOP.

You can set up to four substitute values in "Dosing" mode and one in each of the other counting modes.

After the CPU transition from STOP to RUN, the substitute values are retained at the digital outputs until the next value is input.

The outputs are reset if you change the "Reaction to CPU STOP" basic parameter.

CAUTION

Danger of damage to equipment.

Parameterized substitute values are always output when the CPU goes into STOP mode even if the gate is closed and even at outputs that are not enabled.

Only configure substitute values which will not lead to a dangerous system status if output.

Channel n enable hardware interrupts (S7-300, S7-400)

Enabling hardware interrupt trigger (S7-300, S7-400)

Introduction

The FM 350-2 allows you to set which events are to trigger a hardware interrupt during operation.

What is a hardware interrupt?

If you want to set a response to a specific event independently of the CPU cycle, the FM 350-2 can initiate a hardware interrupt. The CPU then interrupts the cyclical program and processes the hardware interrupt OB.

Enabling the hardware interrupt

Hardware interrupts are enabled in two steps:

1. When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.
2. You then select the relevant events under "Enable hardware interrupts":
   - Events to trigger a hardware interrupt in counting modes (Page 278)
   - Events to trigger a hardware interrupt in measuring modes (Page 278)

Hardware Interrupt OB, e.g. OB40

If a hardware interrupt occurs, the user program is interrupted, the data is transferred from the module to the start information of OB40 (Page 279) and OB40 is called. The hardware interrupt is acknowledged by exiting OB 40.

If no hardware interrupt has been programmed, the CPU goes into STOP mode. If you then switch back to RUN, the hardware interrupt requirements will be deleted.

Lost hardware interrupt

If an event occurs which is supposed to trigger a hardware interrupt but there is an identical, previous event which has not yet been acknowledged, no further hardware interrupt will be triggered; the hardware interrupt will be lost. This may lead to the "Hardware interrupt lost" diagnostic interrupt depending on parameter assignment.

If there is less that 2 ms between two events which should trigger a hardware interrupt, the second hardware interrupt will be lost without a diagnostic interrupt being triggered.

Default setting

No hardware interrupts are enabled in the default setting.

Events to trigger a hardware interrupt in counting modes (S7-300, S7-400)

Which events can trigger a hardware interrupt in counting modes?

The following events can trigger a hardware interrupt in counting modes during operation of the FM 350-2:

- The hardware gate opening
- The hardware gate closing
- High and low limit violation
- Reaching a comparison value
  Enter the comparison value in the text box. The comparison values must lie within the count range of the respective operating mode; the count range limits cannot be used.

Select the check box for the relevant events.

You can select a number of different events to trigger a hardware interrupt depending on the operating mode set. You must observe the relevant basic conditions for hardware interrupts on reaching a comparison value (for "On when counter value >= comparison value (Page 274)" and "On when counter value <= comparison value (Page 274)").

Events to trigger a hardware interrupt in measuring modes (S7-300, S7-400)

Which events can trigger a hardware interrupt in measuring modes?

The following events can trigger a hardware interrupt in measuring modes during operation of the FM 350-2:

- The hardware gate opening
- The hardware gate closing
- End of measurement
- Violating low limit
  Enter the low limit in the text box. Information on the value ranges for the low limits can be found under "Measuring basics (Page 187)".
- Exceeding high limit
  Enter the high limit in the text box. Information on the value ranges for the high limits can be found under "Measuring basics (Page 187)".

Select the check box for the relevant events.

You can select a number of different events to trigger a hardware interrupt depending on the operating mode set.

OB 40 start information (S7-300, S7-400)

OB 40 start information

The temporary variable OB40_POINT_ADDR is described in the start information of OB 40.

The OB40_POINT_ADDR variable (bytes 8 to 11) comprises 4 bytes. The information on the event that triggered the hardware interrupt is entered in these bytes.

The table below shows which bits are set for which interrupt. Any bits not listed are irrelevant and are zero.

Operating mode: Count continuously, count once, count periodically

Channel	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	8					Comparison value reached	High and low limit violation	Hardware gate closed	Hardware gate opened
1		Comparison value reached	High and low limit violation	Hardware gate closed	Hardware gate opened				
2 to 7	9 to 11	See byte 8							

Operating mode: Frequency measurement, rotational speed measurement, period measurement

Channel	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	8					High limit violated / low limit violated	End of measurement	Hardware gate closed	Hardware gate opened
1		High limit violated / low limit violated	End of measurement	Hardware gate closed	Hardware gate opened				
2 to 7	9 to 11	See byte 8							

Operating mode: Dosing

Channel	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	8	–	High and low limit violation	Comparison value 4 reached	Comparison value 3 reached	Comparison value 2 reached	Comparison value 1 reached	Hardware gate closed	Hardware gate opened
4	10	See byte 8							

Commissioning the FM 350-2 (S7-300, S7-400)

Using FM 350-2 from the user program (S7-300, S7-400)

Integrating the FM 350-2 into a user program

Instructions are available for integrating the FM 350-2 into a user program. These enable you to handle the desired functions with ease.

The instructions CNT2WRPN, CNT2RDPN and CNT2_CTR can be called in the CPU user program. They ensure communication between the CPU and the FM 350-2.

All data associated with a channel of the module are in the DB of the instruction CNT2_CTR, CNT2WRPN or CNT2RDPN (counter DB). Use the command "Program blocks > Add new block" to add a CNT2_CHANTYPE type data block to your project.

The DIAG_RD instruction is also available for FM 350-2. This operation allows you to transfer the diagnostic information to the counter DB.

FM 350-2: Online & Diagnostics (S7-300, S7-400)

Current module information (S7-300, S7-400)

Default diagnostics (S7-300, S7-400)

FM 350-2 group error LED fault display (S7-300, S7-400)

Where the error is indicated

If the red group error LED lights up, either a fault has occurred in the module (internal fault) or cable connections (external fault), or there is a parameter assignment error.

Which Errors are Indicated?

The table below shows the error types displayed by the group error LED.

Type of Error	Cause of error	Remedy
Internal errors	Watchdog tripped	Module replacement
	Lost hardware interrupt	acknowledged by hardware interrupt processing
External errors	No module parameters	Assign parameters and transfer
	Parameter assignment error	
External channel errors	Encoder supply short-circuited or overloaded	Correct connection
	Fault in NAMUR encoder signals (wire break, short circuit, cable missing)	Correct connection

Channel-specific diagnostics (S7-300, S7-400)

Triggering diagnostics interrupts (S7-300, S7-400)

What is a diagnostic interrupt?

You can determine reactions to internal or external errors in the user program by configuring a diagnostic interrupt which interrupts the CPU cyclic program and calls the diagnostic interrupt OB 82.

What events can trigger a diagnostic interrupt?

The following events can trigger a diagnostic interrupt:

- Channel error in parameterized channel
- No module parameter assignment
- Error in module parameter assignment
- Watchdog reacts
- Hardware interrupt lost
- Signal line monitoring NAMUR encoder reports error
- Fault in 8.2 V NAMUR encoder supply

Enabling a diagnostic interrupt

When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.

Responses to a diagnostic interrupt

When an event occurs that could trigger a diagnostic interrupt:

- The diagnostics information is stored in the diagnostic data records DS0 and DS1 (Page 282) in the module.
- The group error LED lights up.
  The group error LED goes out when the error is eliminated.
- The diagnostic interrupt OB (OB 82) is called.
- The diagnostic data record DS0 is entered in the start information of the diagnostic interrupt OB.
- The count continues unchanged if an OB 82 has been programmed. If no OB 82 has been programmed, the CPU will go into STOP mode.

Default setting

No diagnostic interrupts are enabled in the default setting.

Diagnostic data record DS0 and DS1 (S7-300, S7-400)

Diagnostic data record DS0 and DS1

The information about which event triggered a diagnostic interrupt is stored in the diagnostic data records DS0 and DS1. The diagnostic data record DS0 comprises four bytes. DS1 comprises 16 bytes; the first four bytes are identical to DS0.

Reading a data record from the module

Diagnostic data record DS0 (Page 282) is automatically transferred to the start information when the diagnostic OB is called. These four bytes are written to the local date (bytes 8 to 11) of OB 82.

You can read out the diagnostic data record DS1 (Page 283) (and thus also the content of the DS0) from the module with the instruction DIAG_RD. This only makes sense if an error is reported in a channel in the DS0.

Assignment of diagnostic data record DS0 (S7-300, S7-400)

Assignment of the diagnostic data record DS0 and the start information

The following table shows the assignment of the diagnostic data record DS0 in the start information. Any bits not listed are irrelevant and are zero.

Byte	Bit	Meaning	Note	Event no.
0	0	Module fault	Is set for each diagnostic event	8:x:00
	1	Internal error	Is set for every internal error	8:x:01
	2	External error	Set for all external errors	8:x:02
	3	Error in one channel	See DS1, byte 4 (Page 283) for further details	8:x:03
	6	No parameter assignment	Assign parameters	8:x:06
	7	Parameter assignment faults	See Data errors (Page 283) for a more detailed breakdown	8:x:07
1	0 ... 3	Type class	Is always assigned the value 8	
	4	Channel information	Is always assigned the value 1	
2	3	Watchdog reacts	Module defective, or heavy interference	8:x:33
3	6	Hardware interrupt lost	Check the configuration. A hardware interrupt has been detected and cannot be reported because an identical event has not yet been acknowledged by the CPU.	8:x:46

Assignment of diagnostic data record DS1 (S7-300, S7-400)

Assignment of diagnostic data record DS1

Diagnostic data record DS1 consists of 16 bytes. The first 4 bytes are identical to diagnostic data record DS0. The following table shows the assignment of the remaining bytes. Any bits not listed are irrelevant and are zero. DIAG_RD writes this data record to the DB of the instruction CNT2_CTR, starting at DW212.

Byte	Bit	Meaning	Note	Event no.
4	0 ... 6	Channel type	Always assigned the value 76H	
	7	Other channel types	Is always assigned the value 0	
5	0 ... 7	Diagnostic information length	Is always assigned the value 8	
6	0 ... 7	Number of channels	Is always assigned the value 8	
7	0	Channel fault vector	Channel bit	
8	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 0)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
9	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 1)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
10	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 2)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
11	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 3)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
12	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 4)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
13	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 5)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
14	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 6)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
15	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 7)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96

Data error (S7-300, S7-400)

Where do data errors occur?

Checks are performed when jobs are given to the module by the programming device or by means of the CNT2WRPN instruction. The module reports any errors returned in this check.

Incorrect jobs are not accepted by the module.

Where are data errors displayed?

Data errors are displayed in the parameter assignment interface.

The CHECKBACK_SIGNALS, DATA_ERR = 1 bit is set in the counter DB if an error is located when jobs are checked.

How are data errors acknowledged?

Correct the jobs in accordance with the specifications. Transfer the corrected jobs to the FM 350-2 again.

FM 350-2 diagnostics buffer

Entries in the FM 350-2 diagnostics buffer are displayed in the parameter assignment interface.

Error codes in the diagnostics buffer (S7-300, S7-400)

Error codes

The error numbers and corresponding error descriptions are listed below. An error with an error ID is entered in the diagnostics buffer in the following form:

16#Fx83 yyzz

Where:

- yy = ID for the error class, for example 04 for a channel parameter assignment error.
- zz = Error number, for example 18 for the error "invalid main count direction"

Channel parameter assignment error	Error number	Description
04	01	Job not allowed, because the channel is being used for proportioning
04	18	Illegal main count direction
04	19	Bad count range
04	1A	Comparison value is not in the achievable count range
04	1B	Illegal output response
04	1C	Hardware interrupt enable not possible
04	1D	Invalid parameter assignment for hardware monitoring
04	1E	Invalid parameter assignment for hardware gate
04	1F	Signal selection not possible
04	20	Bad signal selection
04	21	Value for hysteresis too high
04	22	Substitute value illegal
04	23	Reaction on closing gate illegal

Channel parameter assignment error	Error number	Description
04	24	Reserved parameter set to value other than 0
04	25	Reserved parameter set to value other than 0
04	26	Reserved parameter set to value other than 0
04	27	Bad encoder selection
04	28	Counter input diagnostics only possible with NAMUR encoder
04	29	Value to load not in permitted count range
04	2B	The counting range cannot be extended
04	2C	Comparison value 1 is not in the count range
04	2D	Comparison value 2 is not in the count range
04	2E	Comparison value 3 is not in the count range
04	2F	Comparison value 4 is not in the count range
04	30	Invalid low limit for frequency or rotational speed
04	31	Invalid high limit for frequency or rotational speed
04	32	Low limit >= high limit
04	33	Illegal integration time
04	34	Pulses per encoder rotation illegal
04	35	Invalid low limit for period
04	36	Invalid high limit for period

Module parameters assignment error	Error number	Description
05	0A	Illegal interrupt generation
05	0B	Illegal interrupt selection
05	0C	Illegal reaction with active output disable
05	0D	Reserved parameter set to value other than 0
05	18	Time stamp illegal
05	19	Illegal encoder signal selection
05	1A	Illegal channel mode
05	1B	Invalid distribution of CHECKBACK_SIGNALS.USER_STAT in the data block

Data error	Error number	Description
06	18	Illegal main count direction
06	19	Bad count range
06	1A	Comparison value is not in the achievable count range
06	1B	Illegal output response
06	1C	Hardware interrupts cannot be enabled
06	1D	Invalid parameter assignment for hardware monitoring
06	1E	Invalid parameter assignment for hardware gate
06	1F	Signal selection not possible

Data error	Error number	Description
06	20	Bad signal selection
06	21	Value for hysteresis too high
06	22	Substitute value illegal
06	23	Reaction on closing gate illegal
06	24	Reserved parameter set to value other than 0
06	25	Reserved parameter set to value other than 0
06	26	Reserved parameter set to value other than 0
06	27	Bad encoder selection
06	28	Counter input diagnostics only possible with NAMUR encoder
06	29	Value to load not in permitted count range
06	2B	The counting range cannot be extended
06	2C	Comparison value 1 is not in the count range
06	2D	Comparison value 2 is not in the count range
06	2E	Comparison value 3 is not in the count range
06	2F	Comparison value 4 is not in the count range
06	30	Invalid low limit for frequency or rotational speed
06	31	Invalid high limit for frequency or rotational speed
06	32	Low limit >= high limit
06	33	Illegal integration time
06	34	Pulses per encoder rotation illegal
06	35	Invalid low limit for period
06	36	Invalid high limit for period

Message	Error number	Description
0F	01	End of module start-up
0F	02	No parameters on the module
0F	03	Module has the default parameter settings
0F	04	Module parameters set
0F	05	PG mode is set
0F	06	PG mode is deselected
0F	07	PG mode has failed
0F	08	PG mode is set, no job from the CPU permitted
0F	09	Bad job sequence

Diagnostic error	Error number	Description
80	04	Missing external auxiliary supply
80	05	Front connector missing
80	06	Module parameters not set
80	07	Bad parameters on module

Diagnostic error	Error number	Description
80	30	Wrong/missing user module
80	31	Communication error
80	32	STOP mode
80	33	Time monitoring responded
80	34	Internal module power supply failed
80	35	Battery 1 flat
80	36	Total backup failed
80	41	Processor failure
80	42	EPROM fault
80	43	RAM error
80	44	ADC/DAC fault
80	45	Fuse fault
80	46	Hardware interrupt was lost
80	90	Bad encoder signal A on channel
80	91	Bad encoder signal B on channel
80	92	Bad encoder signal N on channel
80	93	Bad value between channels
80	94	Fault in 5.2 V or NAMUR encoder supply in channel
80	95	Fault in the 24 V encoder supply
80	96	NAMUR encoder short circuit/wire break at channel

1.2.1.4 Using FM 450-1 (S7-300, S7-400)

FM 450-1 front connector terminal assignment (S7-300, S7-400)

Front connector

You connect the following to the 48-pin front connector:

- Count signals
- Digital inputs
- Digital outputs
- Encoder power supply
- Auxiliary voltage and load voltage

The following illustration shows the front of the front connector, the strip printed with terminal assignment and the labeling strips.

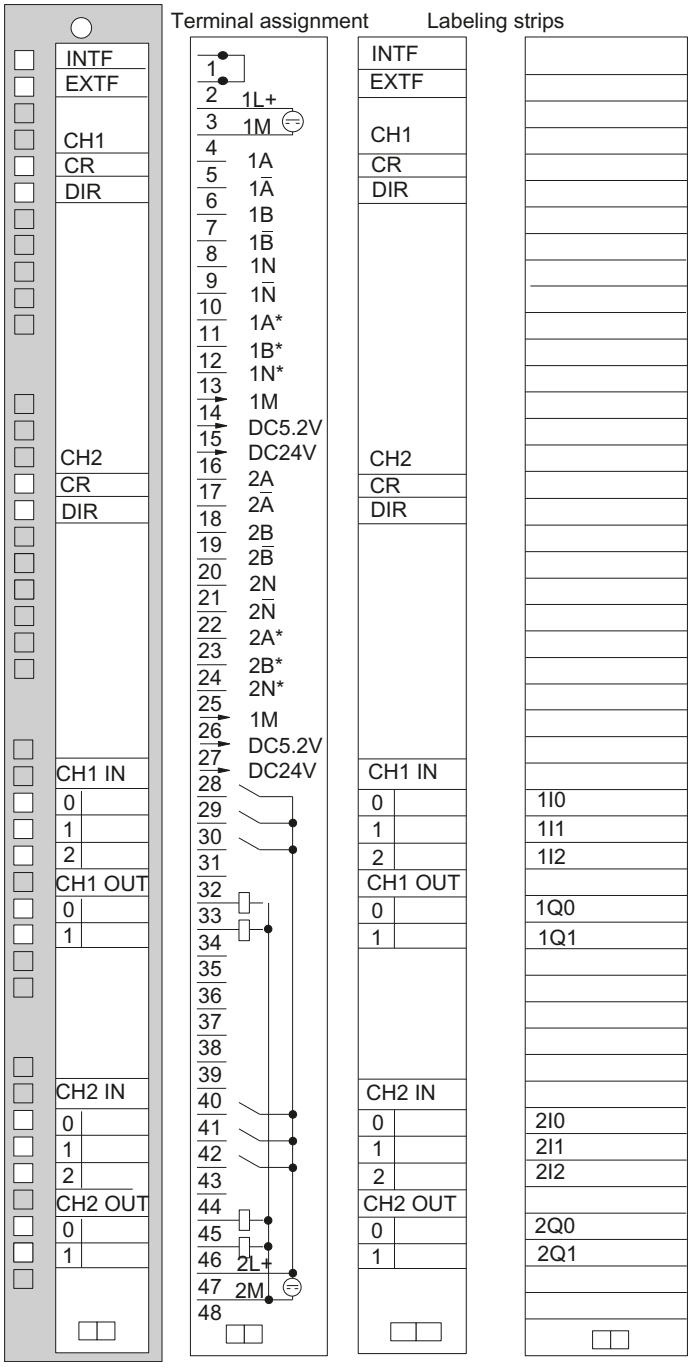
Terminal assignment
Labeling strips
INTF
EXTF
CH1
CR
DIR
CH2
CR
DIR
CH1 IN
0
1
2
CH1 OUT
0
1
CH2 IN
0
1
2
CH2 OUT
0
1
1
2 1L+
3 1M
4
5 1A
6 1Ā
7 1B
8 1B̄
9 1N
10 1N̄
11 1A*
12 1B*
13 1N*
14 1M
15 DC5.2V
16 DC24V
17 2A
18 2Ā
19 2B
20 2B̄
21 2N
22 2N̄
23 2A*
24 2B*
25 2N*
26 1M
27 DC5.2V
28 DC24V
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46 2L+
47 2M
48
INTF
EXTF
CH1
CR
DIR
CH2
CR
DIR
CH1 IN
0
1
2
CH1 OUT
0
1
CH2 IN
0
1
2
CH2 OUT
0
1
1I0
1I1
1I2
1Q0
1Q1
2I0
2I1
2I2
2Q0
2Q1

Terminal assignment of the front connector

The following table shows the terminal assignment for the front connector of the FM 450-1.

Terminal	Name	Input/ output	Function			
			5 V encoder RS 422, symmetrical	24 V encoder, asymmetrical	24 V pulse generator with direction level	24 V initiator
1			-			
2			-			
3	1L+	Input	24 V auxiliary voltage supply for encoders			
4	1M	Input	Auxiliary voltage ground to supply encoders			
Counter 1						
5	1 A	Input	Encoder signal A	-		
6	1 /A	Input	Encoder signal /A	-		
7	1 B	Input	Encoder signal B	-		
8	1 /B	Input	Encoder signal /B	-		
9	1 N	Input	Encoder signal N	-		
10	1 /N	Input	Encoder signal /N	-		
11	1 A*	Input	-	Encoder signal A*		
12	1 B*	Input	-	Encoder signal B*	Directional signal	-
13	1 N*	Input	-	Encoder signal N*	-	
14	1M	Output	Ground for encoder power supply			
15	5.2 V DC	Output	Encoder power supply 5.2 V	-		
16	24 V DC	Output	-	Encoder power supply 24 V		
Counter 2						
17	2 A	Input	Encoder signal A	-		
18	2 /A	Input	Encoder signal /A	-		
19	2 B	Input	Encoder signal B	-		
20	2 /B	Input	Encoder signal /B	-		
21	2 N	Input	Encoder signal N	-		
22	2 /N	Input	Encoder signal /N	-		
23	2 A*	Input	-	Encoder signal A*		
24	2 B*	Input	-	Encoder signal B*, directional signal		
25	2 N*	Input	-	Encoder signal N*		
26	1M	Output	Ground for encoder power supply			
27	5.2 V DC	Output	Encoder power supply 5.2 V	-		
28	24 V DC	Output	-	Encoder power supply 24 V		
Counter 1						
29	1I0	Input	Digital input 1I0			
30	1I1	Input	Digital input 1I1			
31	1I2	Input	Digital input 1I2 (set counter)			
32			-			
33	1Q0	Output	Digital output 1Q0			

Terminal	Name	Input/ output	Function			
			5 V encoder RS 422, symmetrical	24 V encoder, asymmetrical	24 V pulse generator with direction level	24 V initiator
34	1Q1	Output	Digital output 1Q1			
35			-			
36			-			
37			-			
38			-			
39			-			
40			-			
Counter 2						
41	2I0	Input	Digital input 2I0			
42	2I1	Input	Digital input 2I1			
43	2I2	Input	Digital input 2I2 (set counter)			
44			-			
45	2Q0	Output	Digital output 2Q0			
46	2Q1	Output	Digital output 2Q1			
47	2L+	Input	24 V load voltage for the digital inputs and outputs			
48	2M	Input	Load voltage ground for digital inputs and outputs			

Note

The circuits for the counter inputs (encoder supply, encoder signals) are non-isolated from the ground of the CPU. You should therefore connect terminal 4 (1M) to the CPU ground using a low-resistance connection.

If you supply the encoders with external voltage, you must also connect the mass of this external voltage supply to the ground of the CPU.

You will find more information on construction and wiring in the manual (http://support.automation.siemens.com/WW/view/en/1118412) for the FM 450-1 counter module.

Auxiliary voltage 1L+, 1M

Connect a direct voltage of 24 V to the 1L+ and 1M terminals for the 5 V and 24 V encoder voltage supply.

An integrated diode protects the module from a reversal in the polarity of the auxiliary voltage.

The module monitors whether the auxiliary voltage is connected.

5.2 V DC encoder power supply

The module generates a voltage of 5.2 V from the auxiliary voltage 1L+/1M at a maximum current of 300 mA per count channel; this is provided at the corresponding "DC5.2V" connection for the short-circuit proof supply of a 5 V encoder.

24 V DC encoder power supply

1L+/1M is provided at output "DC24V" for 24V short-circuit proof encoder voltage supply. The encoder power supply is short-circuit checked.

5 V encoder signals A and /A, B and /B, N and /N

You can connect incremental encoders with 5 V differential signal to each counter in accordance with RS 422, i.e. incremental encoders with the differential signal A and /A, B and / B, N and /N.

The A and /A, B and /B, N and /N signals are connected via the correspondingly labeled terminals

The signals N and /M are only connected if you wish to set the counter to the zero mark of the encoder.

The inputs are not isolated from the S7-400 bus.

24 V encoder signals A*, B* and N*

24 V signals are identified with the letters A*, B* and N*.

You can connect three different types of encoders to each counter:

- Incremental encoder with 24 V signals:
  The signals A*, B* and N* are connected via the correspondingly labeled pins.
- Pulse encoders without directional level:
  The signal is connected to terminal A*
- Pulse encoders with directional level:
  The count signal is connected to the terminal A*. The directional level is connected to terminal B*.

The inputs are not isolated from the S7-400 bus.

Input filter for 24 V encoder signals

To suppress faults, you can configure input filters (RC elements) with a uniform filtering time for the 24 V inputs A*, B* and N*. The following two input filters are available for each counter:

Characteristics	Input filter 1 (default setting)	Input filter 2
Typical input delay	1 µs	15 µs
Maximum count frequency	200 kHz	20 kHz
Minimum pulse width of the count signals	2.5 µs	25 µs

Digital inputs

You can use digital inputs 1I0 and 1I1 to control the gates of counter 1.
Digital inputs 2I0 and 2I1 are used to control the gates of counter 2.
The gates can be both level-controlled and edge-controlled.

The 1I2 digital input is for setting counter 1 to the load value.
Digital input 2I2 is for setting counter 2 to the load value.

The digital inputs are operated with a 24 V nominal voltage.

The digital inputs are isolated from the bus of the S7-400 and the counter inputs.

Input filter for digital inputs

To suppress faults you can assign parameters to the input filters (RC elements) with a uniform filtering time for the digital inputs 1I0, 1I1 and 1I2 or 2I0, 2I1 and 2I2. The following two input filters are available:

Characteristics	Input filter 1 (default setting)	Input filter 2
Typical input delay	1 µs	15 µs
Minimum pulse width of input signals	2.5 µs	25 µs

Digital outputs

FM 450-1 has the digital outputs 1Q0 and 1Q1 (for counter 1) and 2Q0 and 2Q1 (for counter 2) for directly triggering control processes.

The digital outputs are supplied by load voltage 2L+.

The digital outputs are isolated from the bus of the S7-400 and the counter inputs.

The digital outputs are Sourcing outputs and can carry load current of 0.5 A. They are protected against overload and short circuit.

Note

The direct connection of relays and cutouts can be executed without external wiring

The time response of the digital outputs depends on parameter assignment.

2L+/2M load voltage

To ensure the power supply for digital outputs 1Q0 and 1Q1 or 2Q0 and 2Q1, you must connect a load voltage of 24 V to the module terminals 2L+ and 2M.

An integrated diode protects the module from reversing the polarity of the load voltage.

The load voltage 2L+/2M is not monitored by the FM 450-1.

Configuring FM 450-1 (S7-300, S7-400)

Configuring FM 450-1 (S7-300, S7-400)

Specify channel (S7-300, S7-400)

Copy channel

You can copy the parameters of one channel to another.

1. Select the channel whose parameters are to be copied (source channel).
2. Select the channel to which the parameters are to be copied (target channel).
3. Press the "Copy" button.

Channel n operating modes (S7-300, S7-400)

Selecting an operating mode (S7-300, S7-400)

Description

You define counter module functionality by specifying an operating mode.

Operating mode	Description
Count continuously (Page 182) (with or without gate)	The FM 450-1 counts continuously from the current counter value.
Count once (Page 183) (with hardware or software gate)	When the gate opens, the FM 450-1 counts from the load value to the count limit.
Count periodically (Page 184) (with hardware or software gate)	When the gate opens, the FM 450-1 counts from the load value between the count limits.

The default setting is the "Count continuously" operating mode.

Selecting the Gate Function (S7-300, S7-400)

Description

You define the count process in the selected operating mode by specifying a gate control.

- without gate
  All gates are open.
- Software gate (Page 200)
  The software gate is opened and closed by the input parameter / control bit SW_GATE.
- Hardware gate (Page 201), level-controlled or edge-controlled
  The hardware gate is opened and closed by the corresponding signals at the digital inputs.

The default setting is "Without gate".

Selecting the counting range (S7-300, S7-400)

Description

There is a 32-bit wide count register on the module. With the Count range, you select whether the module is to count only in the positive range or whether the 32nd bit is interpreted as a sign bit thus allowing negative numbers to be represented.

Select the desired counting range.

For more information, please see "Counting ranges and count limits (Page 179)".

Violation of the high limit, low limit and zero pass

A bit is set in the DB of the CNT_CTRL instruction for both count limits when the high or low limit is violated.

A bit is also set in the DB in the "-31 to +31-bits" counting range in the event of a zero pass.

Depending on the count direction, violation of the high or low limit is also indicated in the "0 to +32 bits" counting range upon zero pass.

Event	Status bit in DB
Violation of the high limit	STS_OFLW is set.
Violation of the low limit	STS_UFLW is set.
Zero pass	STS_ZERO is set.

Initiating Hardware Interrupts

You can also signal the violation of the high limit and low limit and the zero pass events using hardware interrupts.

Channel n encoder (S7-300, S7-400)

Selecting the encoder signal type (S7-300, S7-400)

Selecting the signal type

1. Select the type of signal for the encoder used.
   In line with your selection and the operating mode, any option not possible for this specific encoder will be disabled. A disabled option is identified by grayed font in the corresponding selection window.
2. Select the options to be used with the selected encoder.

You can select the signal type for one of the following encoders:

- Encoder: 5 V incremental (Page 295) (5 V differential signal)
- Encoder: 24 V differential signal (Page 295) (24 V track A+B phase shifted)
- Encoder: 24 V pulse and direction (Page 296)
- Encoder: 24 V initiator (Page 296)

Encoder: 5 V differential signal (S7-300, S7-400)

5 V incremental encoder

Select the signal type "5V differential signal" to use a 5 V differential encoder with differential signal A, /A, B, /B and N, /N.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation (Page 206)	Single	Only one edge of signal A is evaluated.
	Double	Both edges of signal A are evaluated.
	Quadruple	Both edges of signals A and B are evaluated.
Count direction (Page 204)	Normal	To adapt the count direction to the process
	Inverted	To adapt the count direction to the process
Max. count frequency	–	The maximum count frequency for this encoder is 500 kHz.
Encoder inputs	–	Not relevant
Monitoring (Page 204)	A+B+N	All signal pairs are monitored for short-circuit and wire break.
	A+B	Signal pairs A and B are monitored.
	A	Only signal pair A is monitored.
	None	No signal pair is monitored.

Encoder: 24 V incremental (S7-300, S7-400)

24 V incremental encoder

Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation (Page 206)	Single	Only one edge of signal A is evaluated.
	Double	Both edges of signal A are evaluated.
	Quadruple	Both edges of signals A and B are evaluated.
Count direction (Page 205)	Normal	To adapt the count direction to the process
	Inverted	To adapt the count direction to the process

Parameter	Option	Meaning
Max. count frequency (Page 297)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 297)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Encoder: 24 V pulse and direction (S7-300, S7-400)

24 V pulse generator

Select signal type "24 V pulse and direction" to use a 24 V pulse generator with count signal A* and direction signal B*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation	–	Not relevant
Count direction	–	Not relevant
Max. count frequency (Page 297)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 297)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Encoder: 24 V initiator (S7-300, S7-400)

24 V initiator

Select the signal type "24 V initiator" to use a 24 V pulse generator with count signal A*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation	–	Not relevant
Count direction	–	Not relevant
Max. count frequency (Page 297)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 297)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Encoder input parameter assignment (S7-300, S7-400)

Encoder input parameter assignment

The count direction is specified by means of the encoder output parameter assignment. The table shows a change in count direction based on parameter assignment.

Parameter assignment	Terminal B*	Count direction
Sinking output	not wired	Down
	Short-circuited to ground	Up
Sourcing output/push pull	not wired	Up
	24 V connected	Down

Set the "24 V pulse and direction" parameter when selecting the encoder.

You cannot reverse the direction of these count signals by inverting the B* signal.

Note

As all signals are added, this type of evaluation may cause the count value to "drift off" at the edge if the count signal oscillates.

Input filters for 24 V counter inputs (S7-300, S7-400)

Input filters for 24 V counter inputs

For the purpose of suppressing interference, you can configure input filters with a uniform filter time for the 24 V counter inputs A*, B* and N* and for the digital inputs. Input filters available:

Characteristics	Input filter 1 (default setting)	Input filter 2
Typical input delay	1 µs	15 µs
Maximum count frequency	200 kHz	20 kHz
Minimum pulse width of the count signals	2.5 µs	25 µs

Channel n inputs (S7-300, S7-400)

Selecting gate signals (S7-300, S7-400)

Controlling the hardware gate

The following options are available for hardware gate control:

- Level-controlled hardware gate (with level at input 1I0 or 2I0)
- Level-controlled hardware gate (for opening with edge at input 1I0 or 2I0 and closing with edge at input 1I1 or 2I1)

For additional information on this topic, please refer to "Hardware gate (Page 201)".

Selecting minimum pulse width (S7-300, S7-400)

Minimum pulse width

To suppress faults, you can configure input filters (RC elements) with a uniform filtering time for the 24 V inputs A*, B* and N* and for the digital inputs.

Two input filters are available with the following data:

Characteristics	Input Filter 1 (default setting)	Input Filter 2
Typical input delay	1 µs	15 µs
Maximum count frequency	200 kHz	20 kHz
Minimum pulse width of the count signals	2.5 µs	25 µs

Setting the counter (S7-300, S7-400)

Setting the counter (S7-300, S7-400)

Overview

If you want to start the count from a specific value (the load value), you must parameterize the signal that is to be used to set this counter to the load value. You can set the counter as follows:

- With the L_DIRECT input parameter of the CNT_CTRL instruction
- With an external signal, using either the 1I2 (2I2) input or the input in combination with the encoder zero mark

Load value

Any number within the limits of the count range can be set for the count range.

The load value is interpreted according to the selected counting range. If, for example, you specify FFFF FFFF H as the load value, this will be interpreted as 4 294 967 295 in the "0 to +32 bits" counting range and as -1 in the counting range "-31 to +31 bits".

Enter the load value in the DB of instruction CNT_CTRL.

The following ranges of values are permitted for the load value:

Range of values for load values	
Low limit	Maximum low count limit
High limit	Maximum high count limit

Setting the Counter with an External Signal

You can select two different external signals with which you can set a counter to the load value:

- Input 1I2 (2I2) (Page 299) only
- Input 1I2 (2I2) and the zero mark (Page 301) of the encoder

You use the zero mark of the encoder if you want to synchronize the counter to a specific counter status at a specific point in your process. This improves the precision of the counting process.

The counter is set independently of the Count mode.

When the counter has been set with an external signal, bit STS_SYNC will be set in the DB.

Note

Synchronization of a counter with zero mark only makes sense if the gate is open.

If you have only enabled one count direction when setting a counter with an external signal, you must note that when the gate is closed the current count direction is saved (frozen). This enables synchronization of the counter against the enabled count

Setting the counter with input 1I2 (2I2) (S7-300, S7-400)

Setting the counter with input 1I2 (2I2)

The counter can be loaded with the load value by means of a rising edge at input 1I2 (2I2) (DI set).

You can define the behavior of the FM 450-1 with a rising edge at input 1I2 (2I2) with the variables ENSET_UP and ENSET_DN in the DB of the CNT_CTRL instruction and by assigning parameters.

Parameter	Behavior of the FM 450-1
ENSET_UP set	The counter is only set if counting up.
ENSET_DN set	The counter is only set if counting down.
ENSET_UP and ENSET_DN set	The counter is set when counting up and counting down.
Parameterization "Single setting of counter"	The counter is set only on the first rising edge at input 1I2 (2I2). If the counter is to be set again, you must first set ENSET_UP or ENSET_DN again. The counter is then set again on the next rising edge at input 1I2 (2I2).
Parameterization "Multiple setting of counter"	The counter is set at every rising edge at input 1I2 (2I2) as long as ENSET_UP and/or ENSET_DN is set.

Note

It is imperative that you set one of the two variables ENSET_UP and/or ENSET_DN so that the relevant counter can be set via the input 1I2 (2I2).

Set once with input 1I2 (2I2)

The figure below shows setting counter 1 once with input 1I2. The same procedure applies for setting counter 2 with input 2I2. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

Counter 1 is set with the first rising edge at input 1I2 as long as ENSET_UP is also set. If you want to set the counter again, you must first reset ENSET_UP and then set it again. The next rising edge at input 1I2 will then result in the counter being set.

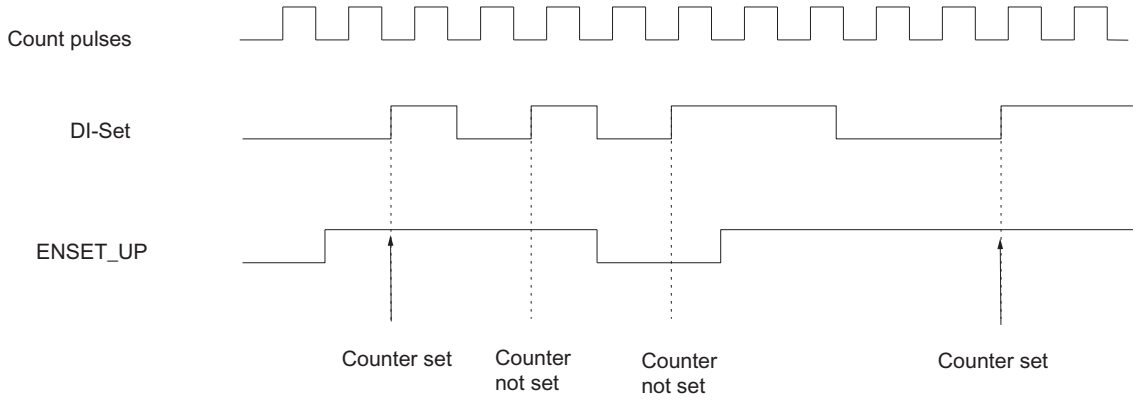

Setting repeatedly with input 1I2 (2I2)

The figure below shows the repeated setting of counter 1 with input 1I2. The same procedure applies for setting counter 2 with input 2I2. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

Counter 1 is set with every rising edge at input 1I2 as long as ENSET_UP is also set. If you reset ENSET_UP, the counter will not be set with a rising edge at input 1I2. Only once you have set ENSET_UP again will the next rising edge at input 1I2 set the counter.

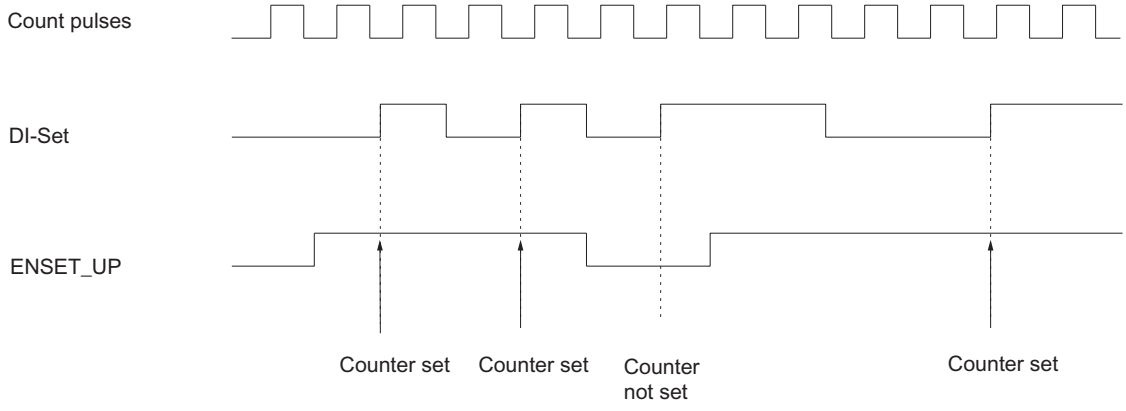

Setting the counter with input 1I2 (2I2) and the zero mark (S7-300, S7-400)

Setting the counter with input 1I2 (2I2) and the zero mark

If you assign parameters so that the counter is set by the encoder zero mark ("Evaluate zero mark for setting"), the counter will be set to the load value at the rising edge of the zero mark.

Setting is performed only if the 1I2 (2I2) input is also set at the time of the rising edge of the zero mark.

You can define the behavior of the FM 450-1 in the event of a zero mark rising edge with the variables ENSET_UP and ENSET_DN in the DB of the CNT_CTRL instruction and by assigning parameters.

Input parameters	Behavior of the FM 450-1
ENSET_UP set	The counter is only set if counting up.
ENSET_DN set	The counter is only set if counting down.
ENSET_UP and ENSET_DN set	The counter is set when counting up and counting down.
Parameterization "Single setting of counter"	The counter is set only at the first rising of the zero mark. If the counter is to be set again, you must first set ENSET_UP or ENSET_DN again (edge evaluation). The counter is then set again with the next rising edge of the zero mark.
Parameterization "Multiple setting of counter"	The counter is set at every rising edge of the zero mark as long as ENSET_UP and/or ENSET_DN are set.

Note

It is imperative that you set one of the two variables ENSET_UP and/or ENSET_DN and input 1I2 (2I2) so that the counter can be set with the zero mark.

Set once with input 1I2 (2I2) and the zero mark

The figure below shows setting the counter 1 once with the zero mark. The same procedure applies for setting counter 2 with input 2I2. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

Counter 1 is set on the first rising edge of the zero mark provided that ENSET_UP and input 1I2 are also set.

If you want to set counter 1 again, you must first reset ENSET_UP and then set it again. If input 1I2 is not set, the counter will be set upon the first zero mark after it is. If input 1I2 is set, setting will be carried out with the next zero mark.

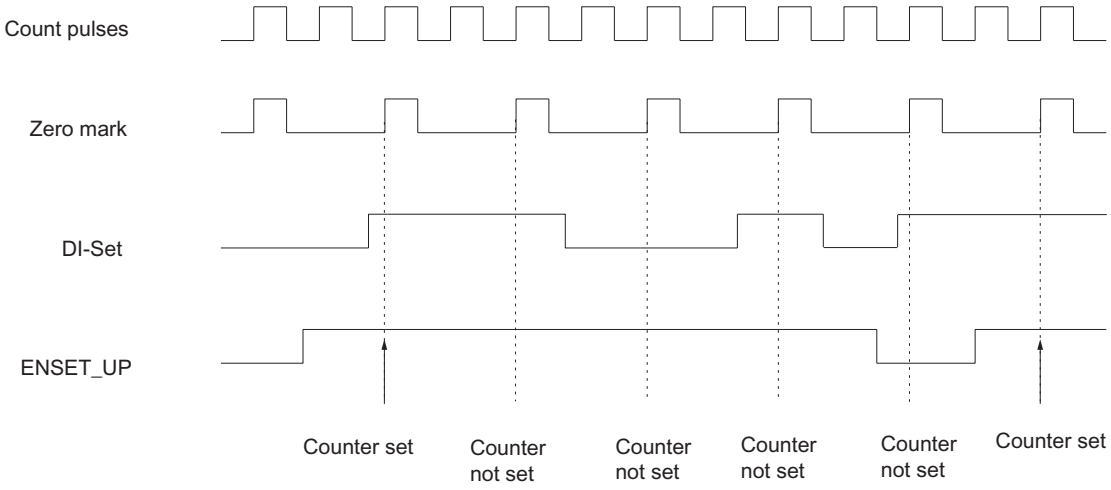

Repeated setting with input 1I2 (2I2) and the zero mark

The figure below shows the repeated setting of counter 1 with the zero mark. The same procedure applies for setting counter 2 with input 2I2. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

The counter is set with every rising edge of the zero mark provided that ENSET_UP and input 1I2 are set.

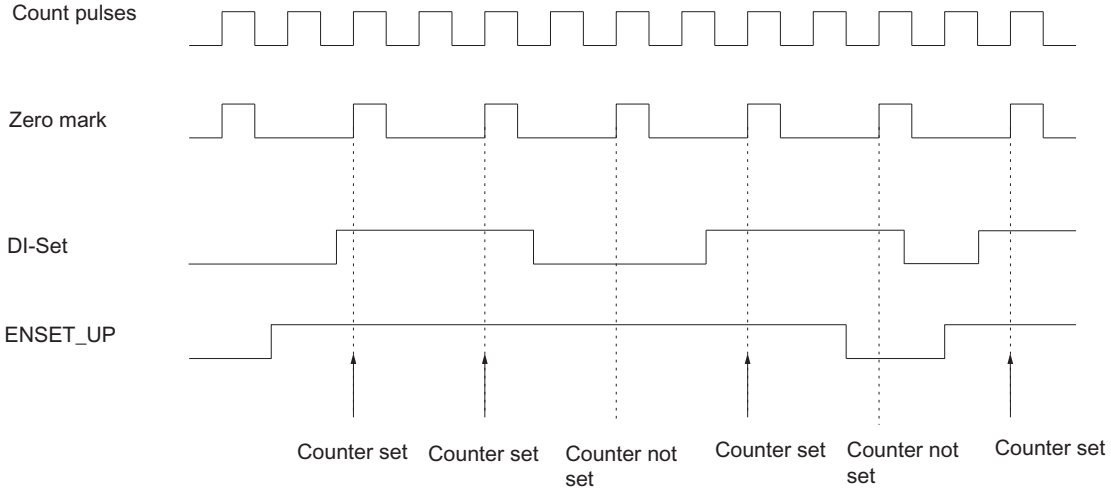

Channel n outputs (S7-300, S7-400)

Specifying pulse duration (S7-300, S7-400)

Pulse duration

You can set a pulse duration in order to adapt to the actuators (contactors, actuators, etc) used in your process. The pulse duration indicates how long outputs Q0 and Q1 are active when a comparison value is reached.

The pulse duration specifies the minimum period for which the output is to be set. This setting is only effective if you preselect the output behavior accordingly. The pulse duration setting is not effective if the output is to be activated at a value between the comparison value and high limit violated or low limit violated.

The pulse duration starts when the output is set. The pulse duration inaccuracy is less than 1 ms.

A value between 0 and 500 ms may be set for the pulse duration. This value applies to both outputs together.

If the pulse duration = 0 ms, the output is set when the comparison value is reached and is reset at the next count pulse.

The value 0 is set by default for the pulse duration.

Note

If you specify a pulse duration of 0 ms, the output will be activated for as long as the counter value is equal to the comparison value (see top of figure below).

Control pulses may be lost at the outputs if the time interval between the count pulses is less than the digital output switching times (up to 300 µs) (see bottom half of figure below).

You should therefore make sure that the interval between the count pulses is greater than the switching times of the digital outputs.

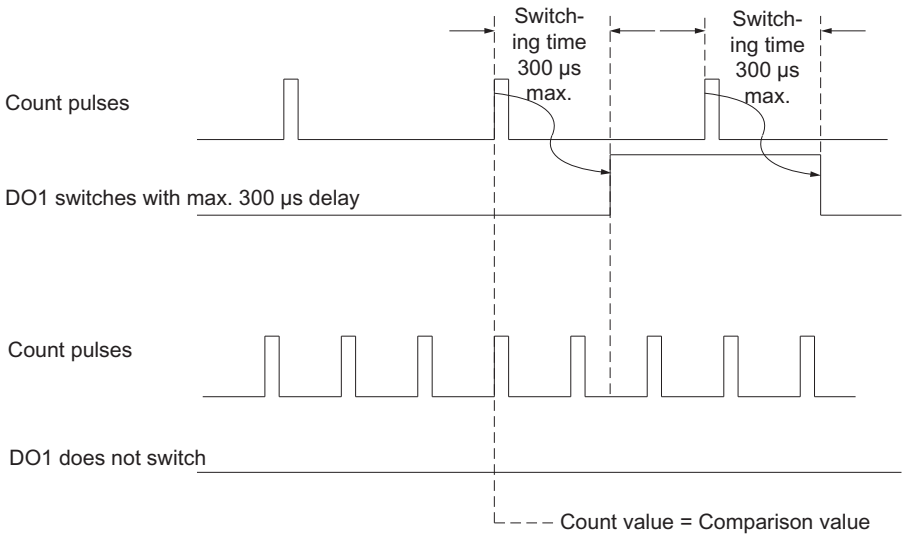

Specifying the behavior of the outputs (S7-300, S7-400)

Comparison values and outputs (S7-300, S7-400)

Description

You can store two comparison values (comparison value 1 and 2) for the counters in the FM 450-1; these are assigned to the two relevant digital outputs (comparison value°1: 1Q0 or 2Q0, comparison value 2: 1Q1 or 2Q1). The comparison values are compared with the current counter value. When the counter value reaches a comparison value, the relevant output can be set.

Comparison Values 1 and 2

You enter the two comparison values in the DB of the CNT_CTRL instruction (CMP_V1, CMP_V2) and transfer them to the FM 450-1 by setting the bits T_CMP_V1 or T_CMP_V2. This does not affect the count.

The comparison values must be within the limits of the selected count range. The comparison value is interpreted according to the selected count range. If you specify FFFF FFFF H as the comparison value, for example, this value will be interpreted as 4,294,967,295 in the "0 to +32 bits" counting range and as -1 in the"-31 to +31 bits" counting range.

The valid value ranges for the comparison values are as follows:

Range of values for comparison values	
Low limit	Maximum low count limit
High limit	Maximum high count limit

Enabling the Outputs

Before the outputs can be controlled, you must first enable them by setting the relevant bits in the DB. If you reset one of these bits the associated output is switched off immediately even if you have parameterized a pulse duration for them.

The following bits must be set before you can enable the outputs:

Output	... Is enabled by enable bit
Q0	CTRL_DQ0
Q1	CTRL_DQ1

Behavior of the outputs in counting modes (S7-300, S7-400)

Specifying the behavior of the outputs (S7-300, S7-400)

Behavior of outputs Q0 and Q1

Select the required behavior for the output in question. Activate the corresponding option button:

- No comparison (Page 305)
- On when counter value >= comparison value (Page 306)
- On when counter value <= comparison value (Page 306)
- Active on reaching the comparison value for pulse duration up (Page 307)
- Active on reaching the comparison value for pulse duration down (Page 307)
- Active on reaching the comparison value for pulse duration up/down (Page 306)

Note

To activate the settings, enter values for comparison values 1 and 2 in the DB with the counter data and transfer them to the module with the CNT_CTRL instruction; you must also set the CTRL_DQ0 and CTRL_DQ1 bits in the DB in order to enable the outputs. See also "Comparison values and outputs (Page 304)".

No comparison (S7-300, S7-400)

No comparison

The output remains deactivated and is not affected by comparison value, zero pass, high limit violated or low limit violated. The output Qx may be used as a purely digital output.

On when counter value >= comparison value (S7-300, S7-400)

On when counter value >= comparison value

The output is set to 1 if the counter is in the range between the comparison value and high limit violated.

Setting the counter to a value between the comparison value and high limit violated sets the output to 1.

The hatched area in the graphic below means that the output is set in this range.

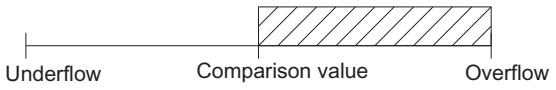

Please note the basic conditions for the behavior of the outputs (Page 308).

On when counter value <= comparison value (S7-300, S7-400)

On when counter value <= comparison value

The output is set to 1 if the counter is in the range between the comparison value and low limit violated.

Setting the counter to a value between the comparison value and low limit violated activates the output at 1.

The hatched area in the graphic below means that the output is set in this range.

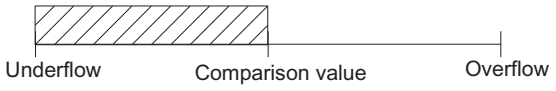

Please note the basic conditions for the behavior of the outputs (Page 308).

Active on reaching the comparison value for pulse duration up/down (S7-300, S7-400)

Active on reaching the comparison value for pulse duration up/down

The output is set to 1 for the duration of the pulse when the counter has reached the comparison value irrespective of the count direction.

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).

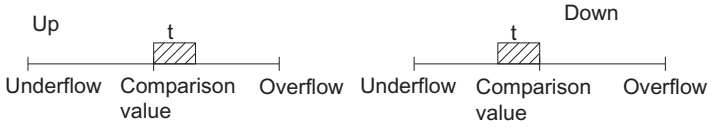

Active on reaching the comparison value for pulse duration up (S7-300, S7-400)

Active on reaching the comparison value for pulse duration up

The output is set to 1 for the pulse duration when the counter reaches the comparison value when counting up.

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).

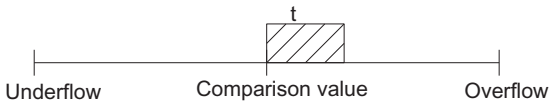

Please note the basic conditions for the behavior of the outputs (Page 308).

Active on reaching the comparison value for pulse duration down (S7-300, S7-400)

Active on reaching the comparison value for pulse duration down

The output is set to 1 for the pulse duration when the counter reaches the comparison value when counting down.

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).

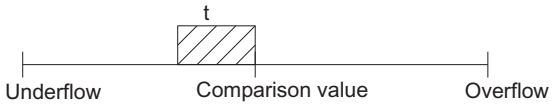

Please note the basic conditions for the behavior of the outputs (Page 308).

Basic conditions for the behavior of the digital outputs (S7-300, S7-400)

Marginal Conditions for the Behavior of the Digital Outputs

You must observe the following basic conditions when configuring the behavior of the digital outputs:

If...	Then...
... ...you want to configure an output "On when counter value >= comparison value or <= comparison value",	... ...You must ensure that the time between these events is greater than the minimum switching time of the outputs (switching time: 300 µs); the control pulses at the outputs will otherwise be lost. No new pulse will be triggered if the counter value reaches the relevant comparison value again while the output is still active. A further pulse can only be triggered when the output is no longer active.
... you want to configure an output that is "Active on reaching the comparison value for pulse duration up",	... ...You must not enable a hardware interrupt on `Reaching the relevant comparison value down'.
... you want to configure an output that is "Active on reaching the comparison value for pulse duration down",	... ...You must not enable a hardware interrupt on `Reaching the (relevant) comparison value up'.

Channel n enable hardware interrupts (S7-300, S7-400)

Enabling hardware interrupt trigger (S7-300, S7-400)

Introduction

You can set which events are to trigger a hardware interrupt for the FM 450-1 counters.

What is a Hardware Interrupt?

If you want there to be a response to a specific event irrespective of the CPU cycle, each FM 450-1 counter can trigger a hardware interrupt. The CPU interrupts the cyclic program on receiving the interrupts and executes the hardware interrupt OB.

Enabling the Hardware Interrupt

Hardware interrupts are enabled in two steps:

1. When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.
2. You then select the relevant events (Page 309) under "Enable hardware interrupts".

Process interrupt OB, OB 4x

If a hardware interrupt occurs, the user program is interrupted, the data is transferred from the module to the start information of OB 4x (Page 309) and OB 4x is called. The process interrupt is acknowledged by exiting the OB 4x.

If no OB 4x is programmed, the CPU goes into STOP mode. If you then switch back to RUN, the hardware interrupt requirements are deleted.

Lost hardware interrupt

If an event occurs that is to initiate a hardware interrupt and the same previous event has not yet been acknowledged, no further hardware interrupt is initiated; the hardware interrupt is lost. This may lead to the "Hardware interrupt lost" diagnostic interrupt, depending on the parameterization.

Default setting

No hardware interrupts are enabled in the default setting.

Events which can trigger a hardware interrupt (S7-300, S7-400)

Which events can trigger a hardware interrupt?

The following events can trigger a hardware interrupt (independently of each other for each of the two counters) during operation of the FM 450-1:

- Opening of the gate (in the operating modes with hardware or software gate)
- Closing of the gate (in the operating modes with hardware or software gate)
- Violation of the high limit
- Violation of the low limit
- Zero pass
- Reaching comparison value 1 or 2 in the up direction
- Reaching comparison value 1 or 2 in the down direction
- Setting the counter (DI set)

Select the check box for the relevant events.

You can select any number of events for triggering a hardware interrupt. Observe the basic conditions for output behavior (Page 308) applicable to hardware interrupts on reaching a comparison value.

OB 4x start information (S7-300, S7-400)

OB 4x start information

The temporary variable OB4x_POINT_ADDR is described in the OB 4x start information.

The OB4x_POINT_ADDR variable (bytes 8 to 11) comprises 4 bytes. The information on the event that triggered the hardware interrupt is entered in these 4 bytes.

The table shows which bits are set for which interrupt. Any bits not listed are irrelevant and are zero.

Byte	Bit	Meaning: Interrupt in the case of...
Counter 1		
8	0	Opening the gate
	1	Closing the gate
	2	Violation of the high limit
	3	Violation of the low limit
	4	Reaching comparison value 1 in the up direction
	5	Reaching comparison value 1 in the down direction
	6	Reaching comparison value 2 in the up direction
	7	Reaching comparison value 2 in the down direction
9	0	Zero pass
	5	Setting the counter (DI set) (synchronization)
Counter 2		
10	0	Opening the gate
	1	Closing the gate
	2	Violation of the high limit
	3	Violation of the low limit
	4	Reaching comparison value 1 in the forward direction
	5	Reaching comparison value 1 in the backward direction
	6	Reaching comparison value 2 in the forward direction
	7	Reaching comparison value 2 in the backward direction
11	0	Zero pass
	5	Setting the counter (DI set) (synchronization)

Using FM 450-1 from the user program (S7-300, S7-400)

Integrating the FM 450-1 into a user program

Instructions are available for integrating the FM 450-1 into a user program. These enable you to handle the desired functions with ease.

The instruction CNT_CTRL can be called in the CPU user program. It ensures communication between the CPU and the FM 450-1.

All data associated with a channel of the module are in the DB of the instruction CNT_CTRL (counter DB). Use the command "Program blocks > Add new block" to add a CNT_CHANTYPE1 type data block to your project.

The DIAG_INF instruction is also available for FM 450-1. This operation allows you to transfer the diagnostic information to the counter DB.

FM 450-1: Online & Diagnostics (S7-300, S7-400)

Current module information (S7-300, S7-400)

Default diagnostics (S7-300, S7-400)

FM 450-1 diagnostic LED fault display (S7-300, S7-400)

Where are the faults displayed?

Faults are indicated by the two red diagnostics LEDs:

- The INTF diagnostics LED displays internal faults in the module.
- The EXTF diagnostics LED displays external faults in the cable connections.

Which fauts are displayed?

The table below shows the error types displayed by the diagnostic LEDs.

Fault type/LED	Cause of fault	Correction
Internal fault Diagnostics LEDINTF	Error in EPROM test	Module replacement
	RAM test error	Module replacement
	Watchdog tripped	Module replacement
	Lost hardware interrupt	Is acknowledged by processing the process interrupt.
External fault Diagnostics LEDEXTF	Auxiliary voltage 1L+/1M not connected or DC24V encoder supply short-circuited	Correct connection
	DC5.2V encoder supply short-circuited or overloaded	Correct connection
	Fault in 5V encoder signals (wire break, short circuit, cable missing)	Correct connection

Initiation of a diagnostics interrupt

All errors except an EPROM test error can trigger a diagnostic interrupt provided you have enabled the diagnostic interrupt in the basic parameters during parameter assignment. The diagnostic data records DS0 and DS1 (Page 313) will show you which error has caused the LED to light up.

Channel-specific diagnostics (S7-300, S7-400)

Triggering diagnostics interrupts (S7-300, S7-400)

Definition of a diagnostics interrupt

You can determine reactions to internal or external errors in the user program, by programming a diagnostics interrupt which interrupts cyclic program execution on the CPU, and triggers a call of diagnostics interrupt OB 82.

Events which can trigger a diagnostics interrupt

The following events can trigger a diagnostic interrupt:

- Faults in the external auxiliary 1L+/1M voltage.
- Short-circuit or overload at the 5.2 V DC encoder supply
- No module configuration
- Faulty module parameters
- Time monitoring (watchdog) triggered
- RAM defective
- Hardware interrupt lost
- Signal A error (wire break, short-circuit, cable missing)
- Signal B error (wire break, short-circuit, cable missing)
- Signal N error (wire break, short-circuit, cable missing)

Enabling diagnostics interrupts

When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.

Reactions to a diagnostics interrupt

When an event occurs that could trigger a diagnostic interrupt:

- The diagnostics information is stored in the diagnostic data records DS0 and DS1 (Page 313) in the module.
- The group error LED is lit.
  The group error LED does dark when the error is eliminated.
- Call of the diagnostics interrupt OB 82.
- The diagnostic data record DS0 is entered in the start information of the diagnostic interrupt OB.
- The count continues unchanged if an OB 82 has been programmed. If no OB 82 has been programmed, the CPU will go into STOP mode.

Default setting

No diagnostic interrupts are enabled in the default setting.

Diagnostics data record DS0 and DS1 (S7-300, S7-400)

Diagnostics data record DS0 and DS1

The information showing the event which has triggered a diagnostics interrupt is written to the diagnostics data records DS0 and DS1. The diagnostic data record DS0 comprises four bytes. DS1 comprises 16 bytes; the first four bytes are identical to DS0.

Reading data records from the module

Diagnostic data record DS0 (Page 313) is automatically transferred to the start information when the diagnostic OB is called. These four bytes are written to the local date (bytes 8 to 11) of OB 82.

You can read the diagnostic data record DS1 (Page 314) (and thus also the content of the DS0) from the module with the DIAG_INF instruction. However, this is only useful if DS0 reports a channel error.

Assignments of diagnostics data record DS0 (S7-300, S7-400)

Assignments of diagnostics data record DS0 in the start information

The following table shows the assignment of the diagnostic data record DS0 in the start information. Any bits not listed are insignificant, and zero.

Byte	Bit	Meaning	Remarks	Event no.
0	0	Module in error state	Set upon all diagnostics events	8:x:00
	1	Internal error	Set at all internal error events: • RAM test error • Time monitoring (watchdog) triggered • Lost hardware interrupt	8:x:01
	2	External error	Set at all external error events: • Auxiliary voltage 1L+/1M not connected, or short-circuit at the 5.2 V DC encoder supply. • Short-circuit or overload at the 5.2 V DC encoder supply • 5 V signal faults • Parameter error	8:x:02
	3	Channel error	Is always assigned 1 for internal and external faults.	8:x:03
	4	Failure of the external auxiliary voltage	Check the voltage	8:x.04
	7	Parameter error	See Data errors (Page 315) for a more detailed breakdown	8:x:07

Byte	Bit	Meaning	Remarks	Event no.
1	0...3	Type class	Always assigned the value 8	
	4	Channel information	Always assigned the value 1	
2	3	Time monitoring (watchdog) triggered	Module defective, or heavy interference	8:x:33
3	3	RAM defective	Module defective, or heavy interference	8:x:43
	6	Hardware interrupt lost	Check the configuration. A hardware interrupt was detected and can not be reported, because the same event is not yet acknowledged by the CPU	8:x:46

Assignment of diagnostic data record DS1 (S7-300, S7-400)

Assignment of diagnostic data record DS1

Diagnostic data record DS1 consists of 12 bytes. The first 4 bytes are identical to diagnostic data record DS0. The following table shows the assignment of the remaining bytes. Any bits not listed are irrelevant and are zero. The DIAG_INF instruction writes this data record to the DB of the CNT_CTRL instruction starting at DW54.

Byte	Bit	Meaning	Remarks	Event no.
4	0...6	Channel type	Always assigned the value 76H	
	7	Further channel types	Always assigned zero value	
5	0...7	Diagnostics data length	Always assigned CH	
6	0...7	Number of channels	Is always assigned 2	
7	0…1	Channel error vector	Bit 0= 1: Fault on channel 1 Bit 1= 1: Fault on channel 2	
8	0	Channel 1 signal A defective		8:x:B0
	1	Channel 1 signal B defective		8:x:B1
	2	Channel 1 signal N defective		8:x:B2
	4	Channel 1 5.2 V encoder supply defective		8:x:B4
	5...7	Reserved		
9		Reserved		
10	0	Channel 2 signal A defective		8:x:B0
	1	Channel 2 signal B defective		8:x:B1
	2	Channel 2 signal N defective		8:x:B2
	4	Channel 2 5.2 V encoder supply defective		8:x:B4
	5...7	Reserved		
11		Reserved		

Data error (S7-300, S7-400)

Data error events

FM 450-1 checks all new parameters transferred to the module. The module reports any errors returned in this check.

Where are data errors displayed?

The CNT_CTRL instruction enters the data errors with the error numbers in the CNT_CTRL DB. You can access this data word in the user program using the DA_ERR_W variable.

Possible data errors

The table below lists the data errors and their error numbers and meaning.

No.	Meaning
0	No error
201	Incorrect receiver selection value
202	Invalid diagnostics value of the signal pair
203	Incorrect value for signal evaluation
204	Incorrect 24 V count signal input filter value
205	Invalid value at digital input filters
206	Reversal of direction not allowed
207	Incorrect 1Q0 (2Q0) behavior configuration
208	Incorrect 1Q1 (2Q1) behavior configuration
209	Pulse duration out of limits
211	Wrong operating mode selected
212	No gate or both gates defined
215	A different count direction was selected for configuring output behavior "Active on reaching the comparison value for pulse duration up or down" in parameter assignment for the hardware interrupt "Reaching the comparison value up or down". The directions set must match.
216	Gate control interrupts are only possible in modes with gate control.
217	Interrupt on "Reaching comparison values" is not valid for output behavior "On when counter value >= comparison value" or "On when counter value <= comparison value".

How are data errors acknowledged?

Correct the parameter values according to specifications. Transfer the corrected parameter set to the FM 450-1 again. It will recheck the parameters and delete the data error in the DB.

Operator error (S7-300, S7-400)

Operator errors events

Operator errors occur as a result of improper operation of the module caused by the incorrect input of control signals.

Objects which indicate operator errors

The CNT_CTRL instruction enters the operator error numbers in the DB. Instruction CNT_CTRL indicates an operator error has occurred when output parameter OT_ERR is set. You can access this data word in your program using the OT_ERR_B variable.

Possible operator errors

The table below lists the operator errors and their error numbers and meaning.

No.	Meaning
0	No error
1	Operating mode can not be started using the SW gate
2	Operating mode can not be canceled
4	Only allowed when the CPU is in STOP

How to acknowledge operator errors

You acknowledge the error with the OT_ERR_A input parameter of the CNT_CTRL instruction.

1.2.1.5 Using ET 200S 1Count24V (S7-300, S7-400)

1Count24V terminal assignment (S7-300, S7-400)

Wiring Rules

The cables (terminals 1 and 5 and terminals 2 and 8) must be shielded. The shield must be supported at both ends. To do this, use the shield support (see the appendix to the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal assignment of the 1Count24V

The following table shows the terminal assignment for 1Count24V.

View	Terminal Assignment	Remarks
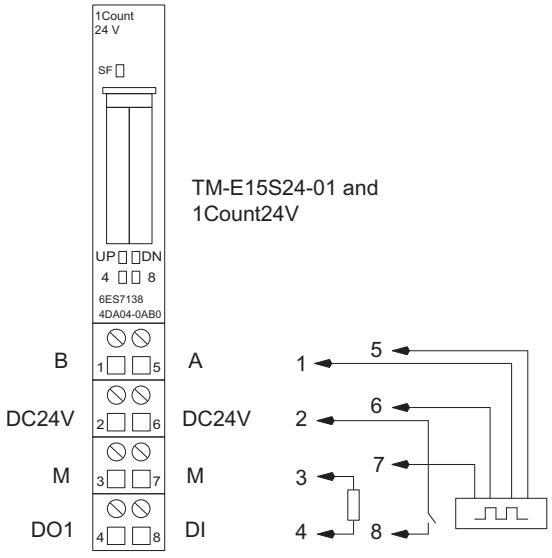		B: Direction input or track B A: Pulse input or track A 24V DC: Sensor supply M: Chassis ground DI: Digital input DO1: Digital output

Pulse generator connection

The following table shows the pulse generator connections to the terminal module and the possible count directions.

Encoder Type	Connection	Count Direction
Pulse generator without direction indicator	24 V count pulses at terminal 5 (A)	Up
Pulse generator with direction indicator	24 V count pulses at terminal 5 (A) and 24 V direction at terminal 1 (B)	Up, down
Pulse generator with 2 tracks that are 90° out of phase	Track A terminal 5 (A) and track B terminal 1 (B)	Up, down

Configuring and assigning parameters to 1Count24V (S7-300, S7-400)

Operation (S7-300, S7-400)

Selecting an operating mode (S7-300, S7-400)

Operating mode

You define 1Count24V functionality by setting an operating mode.

Operating mode	Description
Counting (Page 318)	Preselection for counting modes
Measuring (Page 340)	Preselection for measuring modes
Position feedback (Page 357)	In this mode, 1Count24V counts continuously from the load value: • If another count pulse is received after 1Count24V reaches the high count limit when counting up, it will jump to the low count limit and continue counting from there without losing a pulse. • If another count pulse is received after 1Count24V reaches the low count limit when counting down, it will jump to the high count limit and continue counting from there without losing a pulse.
Fast mode (Page 369) (supported as of firmware version V2.0)	In this mode, 1Count24V counts continuously from the initial value: • If another count pulse is received after 1Count24V reaches the maximum value that can be represented by 25 bits when counting up (all counter bits are set), the count value will jump to "0" and resume counting from there without losing a pulse. • If 1Count24V reaches the value "0" when counting down and another count pulse is then received, the count value will jump to the maximum value that can be represented by 25 bits (all counter bits set) and resume counting from there without losing a pulse. This operating mode can be used for position feedback in particularly short isochrone cycles.

Counting parameters (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP (S7-300, S7-400)

Setting response to CPU/Master STOP

You can configure the response of the 1Count24V to failure of the higher-level controller.

Parameter	Status of the 1Count24V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO1	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode will continue working. The gate and digital output status will not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.
DO 1 substitute a value	The current mode is cancelled. The gate is closed and the substitute value set for the digital output is activated. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters. If "Pulse when comparison value is reached" has been set for the output, the only available pulse duration substitute value will be 1.	The changed parameters are accepted and become effective.
DO 1 keep last value	The current mode is cancelled. The gate is closed and the status of the digital output is maintained. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working upon transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count24V.

Leaving the Assigned State

Under what conditions does the 1Count24V leave the assigned state?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 336).

Automatic New Parameter Assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon power on of the CPU/DP master
- Upon IM 151-1 / IM 151-1 FO power on

- After failure of the DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count24V is plugged
- Upon power on or inserting of the appropriate power module

Signal evaluation (S7-300, S7-400)

Selecting count and direction evaluation (S7-300, S7-400)

Signal evaluation A*, B*

Signal evaluation by means of A*, B* allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Pulse and Direction (Page 321)
  In the case of 24 V pulse generators with a direction level, there must be an interval of at least 5 µs/50 µs (depending on the input filter set) between the direction signal (B*) and the count signal (A*).
  The following figure shows the interval between direction signal and count signal.

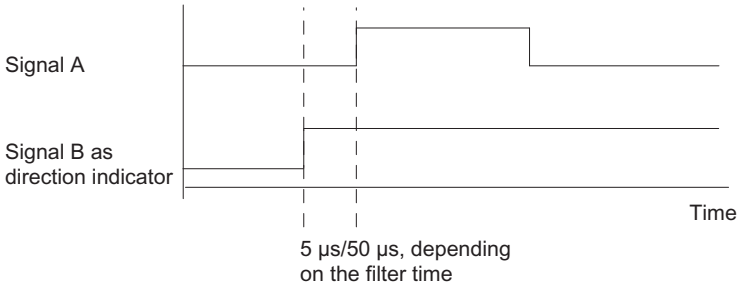

- Rotary Encoder (Page 206)
  If you connect a 24 V rotary encoder with two tracks that are 90 degrees out of phase at the count and direction inputs, you can set single evaluation (Page 206) in all counting modes. You can alternatively set dual (Page 207) or quadruple evaluation (Page 207). You can invert the count direction (Page 322) at direction input B* in all evaluation modes. A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with 1Count24V for the "Encoder inputs" parameter, you must use the M-switching sensors.

Note

The count frequency of 100 kHz is the maximum frequency of the A* and B* signals. The maximum frequency for count pulses is therefore 200 kHz in double evaluation and 400 kHz in quadruple evaluation.

"Pulse and direction" evaluation (S7-300, S7-400)

Pulse and direction

The level at direction input B* is used as the direction setting in this type of evaluation.

An unwired input corresponds to the "Up" count direction if you have selected "Pulse and direction" for the "Signal evaluation" parameter.

The diagram below shows the signals of a 24 V pulse generator with direction level.

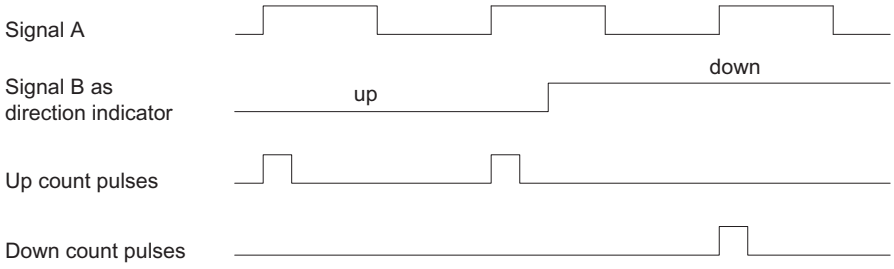

Selecting filters for input signals (S7-300, S7-400)

Input filter

You can activate a filter to filter signals at count input A*, direction input B* and digital input DI in line with the minimum pulse duration or the maximum signal frequency.

You can select a minimum pulse width of 2.5 µs or 25 µs for each input.

Selecting encoder inputs (S7-300, S7-400)

Encoder inputs

A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with for the "Encoder inputs" parameter, you must use the M-switching sensors.

Selecting the count direction (S7-300, S7-400)

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Behavior of the outputs in counting modes (S7-300, S7-400)

Comparison values and outputs (S7-300, S7-400)

Introduction

The 1Count24V allows you to store two comparison values, which are assigned to the digital outputs. The outputs can be activated, depending on the counter status and the comparison values.

Digital outputs

The 1Count24V has a real digital output and a virtual digital output that exists only as a status bit in the feedback interface.

Parameters can be assigned for both outputs ("Function DO1" and "Function DO2" parameters).

You can change the function and the behavior of the digital outputs during operation. The new function takes effect immediately.

Controlling the outputs and comparators simultaneously

You can continue to control the outputs using SET_DO1 or SET_DO2 if you have selected a compare function for the outputs. This allows you to simulate the effect of the compare functions in your control program:

- The output is set with the positive edge of SET_DO1 or SET_DO2.
  However, if the "Pulse on reaching the comparison value" function is selected, only one pulse with the specified duration will be output. SET_DO1 and SET_DO2 have no effect when pulse duration = 0.
  The SET_DO1 control bit cannot be used with output behavior "Switch at comparison values".
- A SET_DO1 or SET_DO2 negative edge resets the output.

Please note that the comparators are still active and can set or reset the output if the comparison result changes.

Note

An output set by SET_DO1 or SET_DO2 cannot be reset by the comparator.

Loading comparison values

You transfer the comparison values to the 1Count24V. This does not affect the count.

Valid range for the two comparison values

Main count direction: None	Main count direction: Up	Main count direction: Down
Low count limit to high count limit	–2147483648 to high count limit -1	1 to 2147483647

Changing the function and behavior of digital outputs

You can change the functions and behavior of the outputs during operation using the control interface (Page 336). The 1Count24V clears the outputs and accepts the values as follows:

- **Function of digital outputs DO1 and DO2:** If you change this function so that the comparison condition is satisfied, the output will not be set until after the next count pulse. However, 1Count24V will not make any changes at the output if hysteresis (Page 327) is enabled.
- **Hysteresis:** Active hysteresis (Page 327) remains active after a change. The new hysteresis range will be applied the next time the comparison value is reached.
- **Pulse duration:** The new pulse duration (Page 330) will take effect with the next pulse.

Selecting output behavior (S7-300, S7-400)

Behavior of outputs DO1 and DO2

Select the behavior of the output in question in the drop down list:

- Output (Page 324)
- On when counter value ≥ comparison value (Page 324)
- On when counter value ≤ comparison value (Page 324)
- Pulse on reaching the comparison value (Page 325)
- Switch at comparison values (Page 325) (DO1 only)

Setting or Modifying the Function and Behavior of the Digital Output DO1

When setting or modifying the behavior of DO1, you must take all assignable interdependencies into account. Failure to do so will generate a parameter assignment error or a loading error.

Boundary conditions:

If you assign "Switch at comparison values" for DO1, you must:

- Set hysteresis = 0, and
- also configure "DO2 function" as "Output".

Output (S7-300, S7-400)

Output

You can switch the outputs on and off with the control bits (Page 336)SET_DO1 and SET_DO2.

The control bits (Page 336)CTRL_DO1 or CTRL_DO2 must be set.

You can query the status of the outputs with the status bits STS_DO1 and STS_DO2 in the feedback interface (Page 338).

The status bits (Page 338)STS_CMP1 and STS_CMP2 indicate that the relevant output is or was switched on. These status bits retain their status until they are acknowledged. If the output is still switched, the corresponding bit is set again immediately. These status bits are also set when the control bit SET_DO1 or SET_DO2 is operated without DO1 or DO2 being enabled.

Isochrone mode

Output DO1 is switched at time T_o in isochrone mode. The status of the virtual output DO2 is signaled at time T_i.

On when counter value ≤ comparison value or when counter value ≥ comparison value (S7-300, S7-400)

Counter status ≤ Comparison Value and Counter Status ≥ Comparison Value

If the comparison conditions are fulfilled, the respective comparator switches on the output. The status of the output (Page 338) is indicated by STS_DO1 and STS_DO2.

The control bits (Page 336)CTRL_DO1 or CTRL_DO2 must be set.

The result of the comparison is indicated by the status bits (Page 338)STS_CMP1 or STS_CMP2. You cannot acknowledge and thus reset these bits until the comparison conditions are no longer fulfilled.

Isochronous mode

In isochrone mode, too, output DO1 is switched as soon as the comparison condition is fulfilled and is therefore independent of the bus cycle. The status of the virtual output DO2 is reported at time T_i.

Pulse on reaching the comparison value (S7-300, S7-400)

Pulse on reaching the comparison value

If the counter status reaches the comparison value, the comparator switches on the respective digital output for the assigned pulse duration.

The control bit (Page 336)CTRL_DO1 or CTRL_DO2 must be set.

The status of the status bits (Page 338)STS_DO1 and STS_DO2 is always the same as that of the corresponding digital output.

The comparison result is indicated by the status bit (Page 338)STS_CMP1 or STS_CMP2 and cannot be reset by acknowledgment until the pulse duration has elapsed.

If a main count direction is assigned, the comparator switches only when the comparison value in the main count direction is reached.

If a main count direction is not assigned, the comparator switches when the comparison value is reached from either direction.

If the digital output is set by control bit (Page 336)SET_DO1 or SET_DO2, it will be reset when the pulse duration has elapsed.

Isochronous mode

In isochrone mode, too, output DO1 is switched as soon as the comparison condition is fulfilled and is therefore independent of the bus cycle. The status of virtual output DO2 is reported at time T_i.

Switch at comparison values (S7-300, S7-400)

Switch at comparison values

The comparator switches the output when the following conditions are met:

- The two comparison values must be loaded with the load functions (Page 336)CMP_VAL1 and CMP_VAL2 and
- The output DO1 (Page 336) must be enabled with CRTL_DO1 after the comparison values are loaded.

The following table shows you when the DO1 is switched on or off:

	DO1 is switched on when	DO1 Is switched off when
V2 < V1 (see Figure below)	V2 ≤ counter status ≤ V1	V2 > counter status or counter status > V1
V2 = V1	V2 = counter status = V1	V2 ≠ counter status ≠ V1
V2 > V1 (see Figure below)	V1 > counter status or counter status > V2	V1 ≤ counter status ≤ V2

The figures below show the status of output DO1 as determined by the counter value and comparison values:

- V2 < V1 at the start of counting

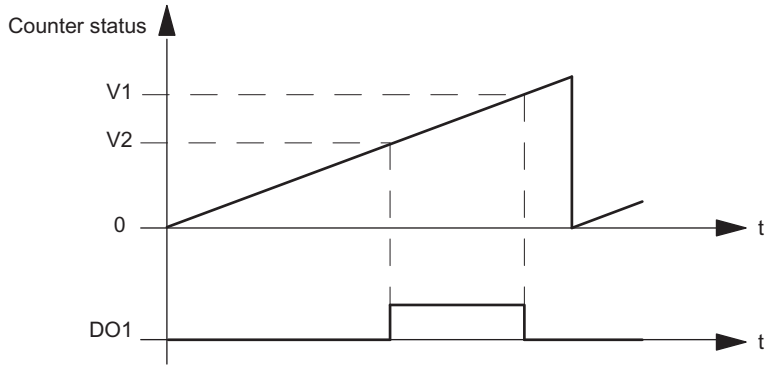

- V2 > V1 at the start of counting

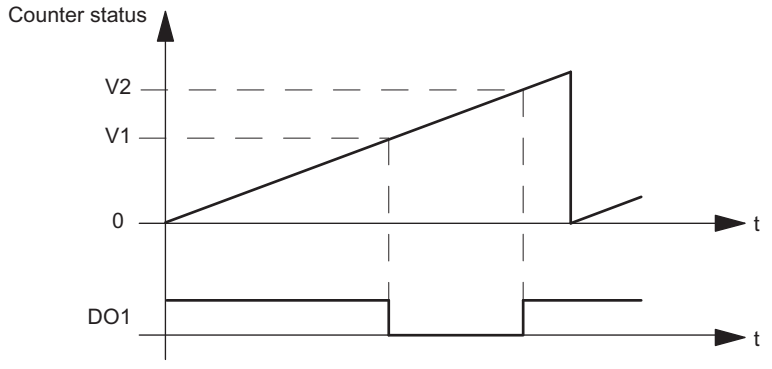

The result of the comparison is shown by the status bit (Page 338)STS_CMP1. You can only acknowledge and thus reset this bit when the comparison condition is no longer fulfilled.

There is no hysteresis in the case of this output behavior.

It is not possible to control the DO1 output with control bit (Page 336)SET_DO1 if this is the output behavior.

Isochronous mode

In isochrone mode, too, output DO1 is switched as soon as the comparison condition is fulfilled and is therefore independent of the bus cycle.

Selecting substitute value DO1 (S7-300, S7-400)

Substitute value DO1

This is where you select the substitute value which is to be output at digital output DO1 if you have selected the parameter "DO1substitute a value" under "Reaction to CPU/Master STOP (Page 319)".

Activating DO1 diagnostics (S7-300, S7-400)

DO1 diagnostics

"DO1 diagnostics" (wire break, short circuit) are possible only with pulse lengths of > 90 ms at digital output DO1.

Select the check box for diagnostics at output DO1.

Setting hysteresis (S7-300, S7-400)

Hysteresis

An encoder can remain at a particular position and then fluctuate around this position. This state causes the counter status to fluctuate around a particular value. If there is a comparison value in this fluctuation range, for example, the associated output is switched on and off in accordance with the rhythm of the fluctuations. The 1Count24V is equipped with programmable hysteresis to prevent this switching in response to small fluctuations. You can assign a range between 0 and 255 (0 means: hysteresis switched off).

Hysteresis also works with high limit violated and low limit violated.

Method of Operation with Counter Status ≤ Comparison Value and Counter Status ≥ Comparison Value

The example in the figure below demonstrates the effect of hysteresis. The figure shows the differences in the behavior of an output when hysteresis of 0 (= switched off) is parameterized as opposed to hysteresis of 3. In the example, the comparison value is 5.

The following settings are assigned for the counter: "Main count direction" = "Up" and "Turn on at counter status≥ comparison value".

When the comparison condition is met, hysteresis becomes active. While the hysteresis is active, the comparison result remains unchanged.

Hysteresis will no longer be active if the counter value is outside the hysteresis range. The comparator switches again according to its comparison conditions.

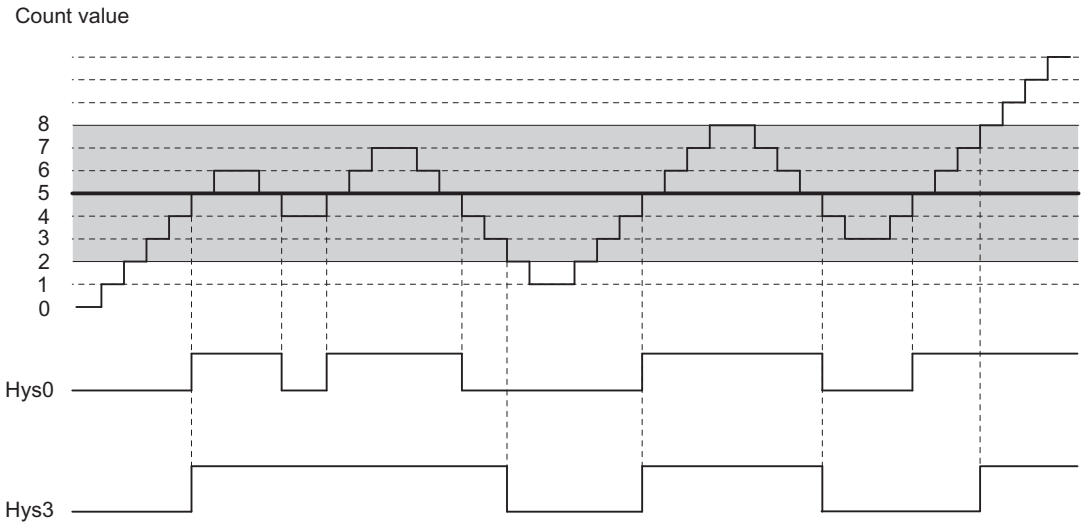

Note

If the count direction changes on the comparison value when hysteresis is active, the output is reset.

Method of Operation when the Comparison Value Is Reached and the Pulse Duration = 0

The example in the figure below demonstrates the effect of hysteresis. The figure shows the various responses of an output with a hysteresis of 0 (= switched off) as opposed to hysteresis of 3. In the example, the comparison value is 5.

The following settings are assigned for the counter: "Pulse on reaching the comparison value", "No main count direction" and "Pulse duration = 0".

When the comparison conditions are met, hysteresis becomes active. While the hysteresis is active, the comparison result remains unchanged.

If the counter value goes outside the hysteresis range, hysteresis is no longer active. The comparator deletes the result of the comparison.

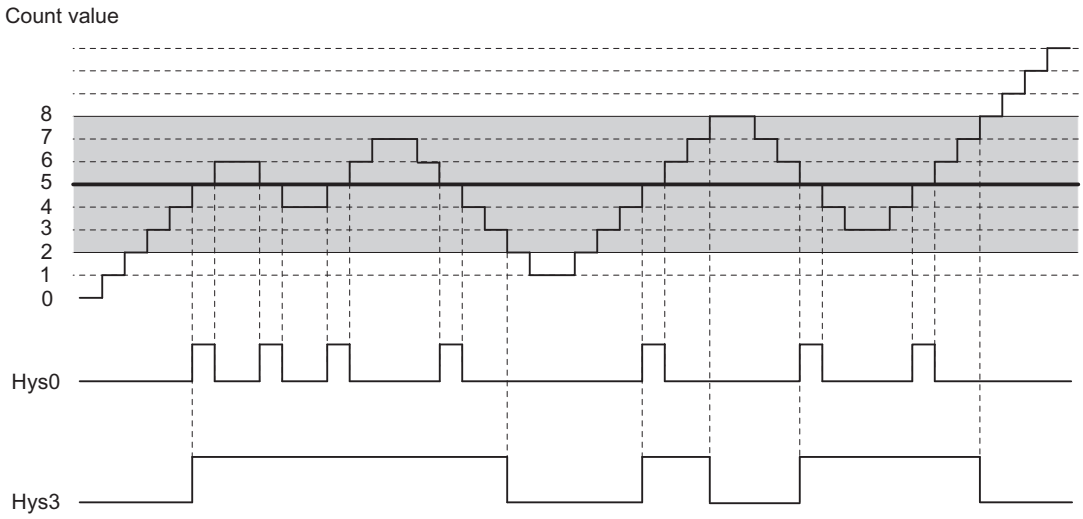

Method of Operation when the Comparison Value Is Reached, Output Pulse Duration

The example in the figure below demonstrates the effect of hysteresis. The figure shows the various responses of an output with a hysteresis of 0 (= switched off) as opposed to hysteresis of 3. In the example, the comparison value is 5.

The following settings are assigned for the counter: "Pulse on reaching the comparison value", "No main count direction" and "pulse duration > 0".

When the comparison conditions have been met, hysteresis becomes active and a pulse of the assigned duration is output.

If the counter value goes outside the hysteresis range, hysteresis is no longer active.

When hysteresis is enabled, the 1Count24V stores the count direction.

If the hysteresis range is exited in a different direction to the one stored, a pulse is output.

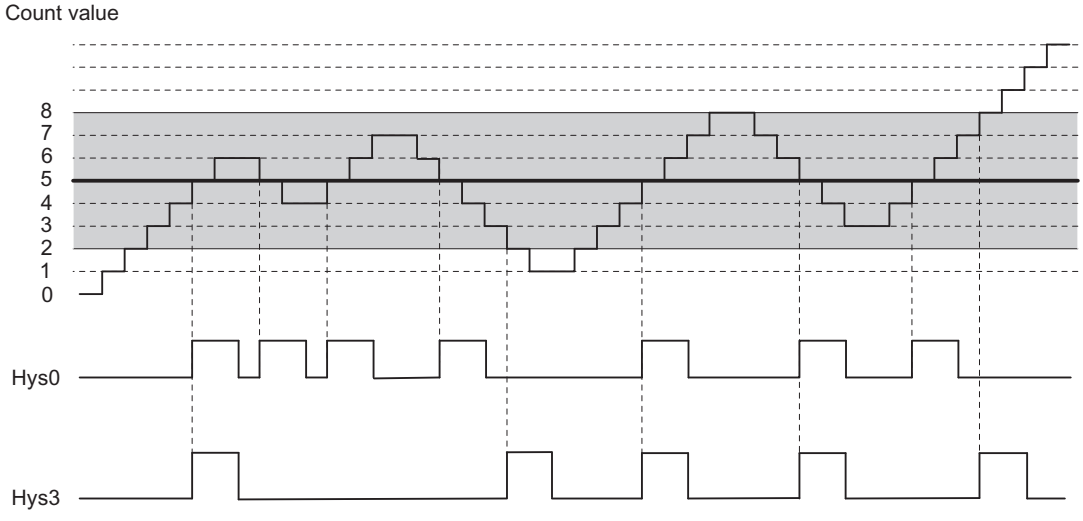

Specifying pulse duration (S7-300, S7-400)

Pulse duration

The pulse duration begins when the respective digital output is set. The pulse duration is accurate to at least 2 ms.

The pulse duration can be set to suit the actuators used. The pulse duration specifies how long the output is to be set for. The pulse duration can be preselected between 0 ms and 510 ms in increments of 2 ms.

If the pulse duration = 0, the output is set until the comparison condition is no longer fulfilled. Note that the count pulse times must be greater than the minimum switching times of the digital output.

Isochronous mode

In isochrone mode, too, output DO1 is switched as soon as the comparison condition is fulfilled and is therefore independent of the bus cycle. The status of virtual output DO2 is reported at time T_i.

Selecting counting mode (S7-300, S7-400)

Description

You define 1Count24V functionality by setting a counting mode.

Operating mode	Description
Count continuously (Page 182) (with or without gate)	1Count24V counts continuously from the current counter value.
Count once (Page 183) (with hardware or software gate)	When the gate opens, 1Count24V counts from the load value to the count limit.
Count periodically (Page 184) (with hardware or software gate)	When the gate opens, 1Count24V counts from the load value between the count limits.

The default setting is the "Count continuously" operating mode.

Selecting main count direction (S7-300, S7-400)

Description

By configuring a main count direction (Page 185) (up or down), you can reduce the maximum counting range by setting an high count limit (Page 336). The parameterized counting range is then between 0 and the high count limit parameterized.

A main count direction can only be selected in the "Count once" and " Count periodically" modes.

You can select the following behavior:

- None (no main count direction)
- Up
- Down

Selecting the gate function (S7-300, S7-400)

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 202): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 202): The counting process starts from the last current counter value after gate stop and restart.

DI function (S7-300, S7-400)

Selecting DI function (S7-300, S7-400)

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 201), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Latch/Retrigger with positive edge (Page 332)
  Counter values are saved (latched) with positive edges at the digital input. The counter is set to the load value each time the values are saved.
- Synchronization with positive edge (Page 333)
  The counter is set to the load value with positive edges at the digital input.
- Latching with positive edge (Page 335)
  Counter values are saved (latched) with positive edges at the digital input.

The default setting is "Input".

Latch/Retrigger with positive edge (S7-300, S7-400)

Latch/Retrigger

This function stores the current internal counter status of the 1Count24V and retriggers counting when there is a positive edge on the digital input. This means that the current internal counter status at the time of the positive edge is stored (latch value), and the 1Count24V is then loaded again with the load value, from which counting resumes.

The following figure shows latch/retrigger when load value = 0.

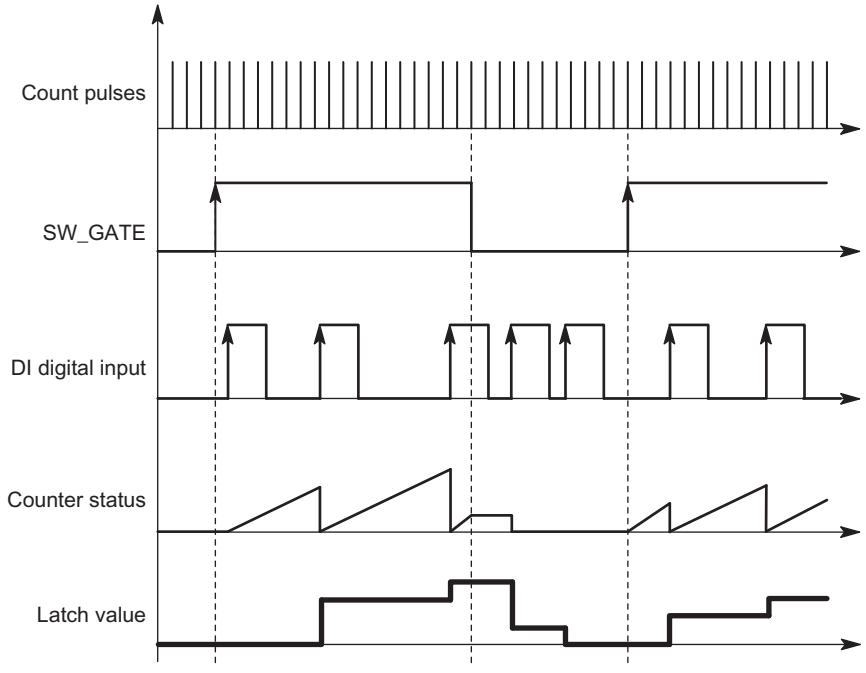

The counting mode must be enabled with the SW gate before the function can be executed. It is started with the first positive edge on the digital input.

The saved rather than the current counter value is displayed in the feedback interface (Page 338). The STS_DI bit indicates the status of the latch and retrigger signal.

The latch value is pre-assigned its RESET status:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

It is not changed when the SW gate is opened.

Direct loading of the counter does not cause the indicated stored counter status to be changed.

If you close the SW gate, counting is only interrupted; this means that when you open the SW gate again, counting is continued. The digital input DI remains active even when the SW gate is closed.

Counting is also latched and triggered in isochronous mode with each edge on the digital input. The counter status that was valid at the time of the last edge before T_i is displayed in the feedback interface.

Extended feedback interface

If the 1Count24V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current counter value can be read from bytes 8 to 11 of the feedback interface (Page 338).

Synchronization with positive edge (S7-300, S7-400)

Synchronization

If you have configured "Synchronization with positive edge", the rising edge of a reference signal at the input will set 1Count24V to the load value.

You can select between once-only and periodic synchronization ("Synchronization" parameter).

The figure below shows the counter values in once-only and periodic synchronization.

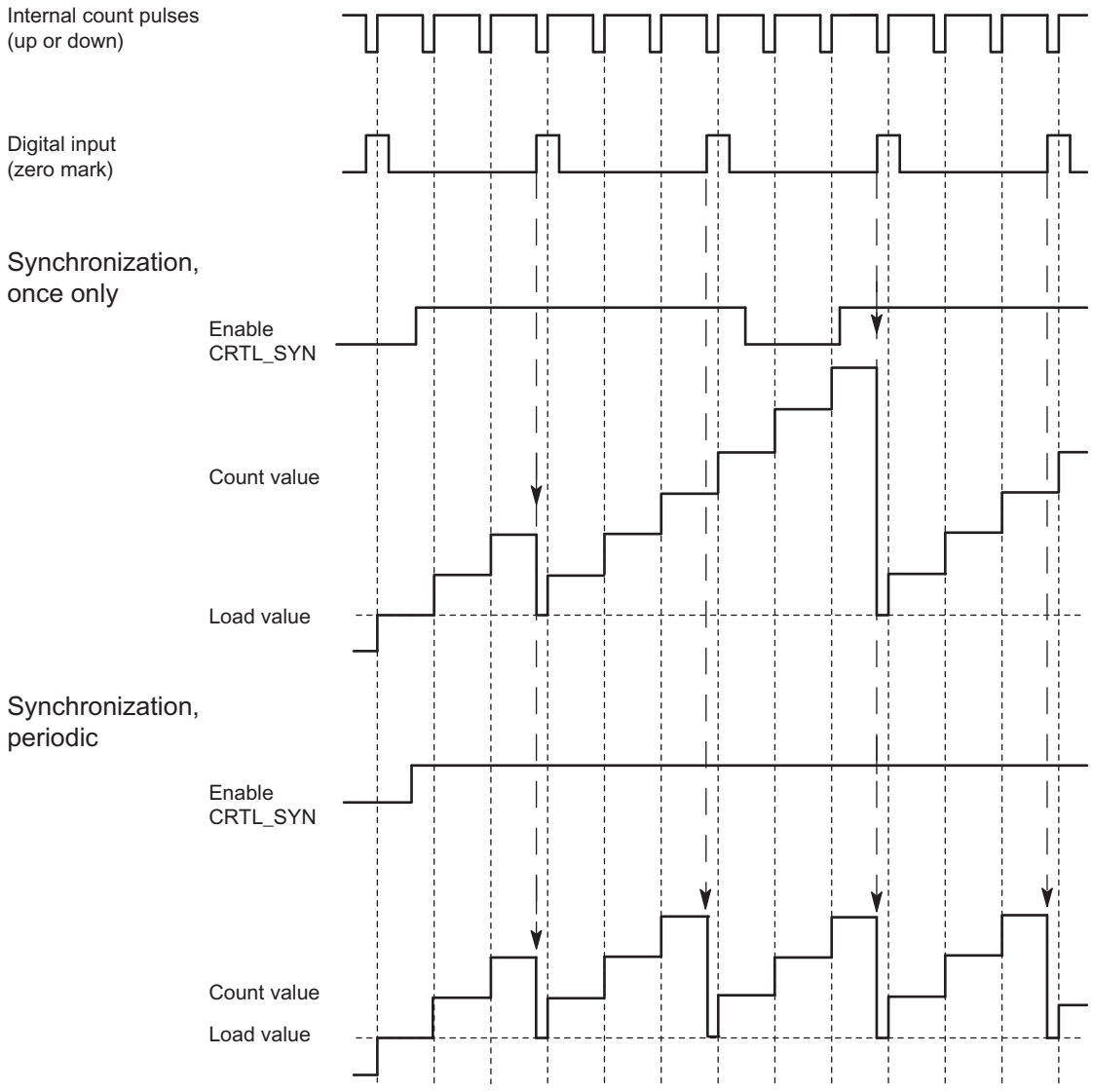

The following conditions apply:

- The counting mode must have been started with the SW gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 336) must be set.
- Synchronize once only: If the enable bit is set, the first edge loads the 1Count24V with the load value.
- Synchronization periodically: If the enable bit is set, the first and each subsequent edge loads the 1Count24V with the load value.
- The feedback bit (Page 338)STS_SYN is set after successful synchronization. It must be reset by the control bit (Page 336)RES_STS.
- The signal of a bounce-free switch or the zero mark of a rotary encoder can serve as the reference signal.
- The feedback bit (Page 338)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 338)STS_SYN indicates that the rising edge at the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Latching with positive edge (S7-300, S7-400)

Latching

Counter value and latch value are pre-assigned their RESET statuses:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

The counting function is started when the SW gate is opened. 1Count24V begins at the load value:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

The following figure shows latching when load value = 0.

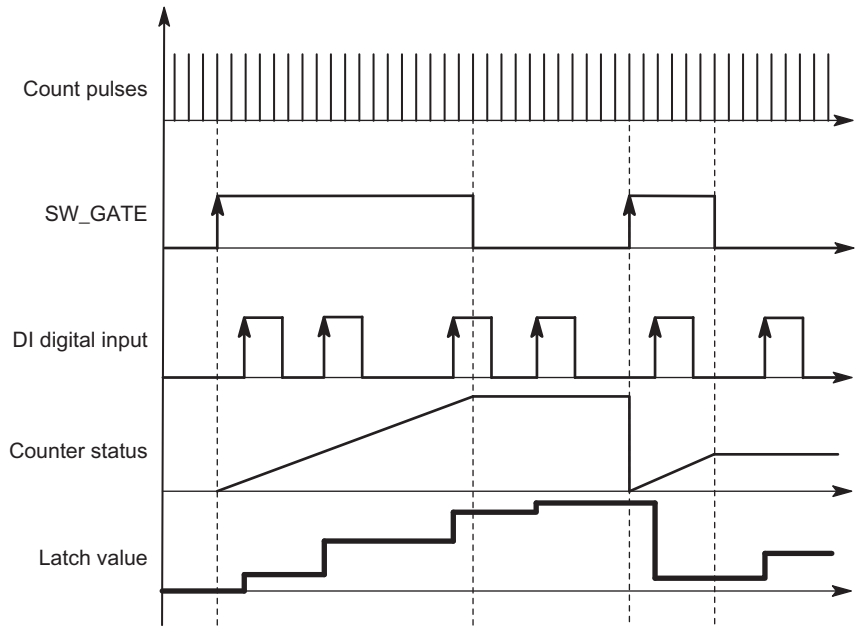

The latch value is always the exact counter status at the time of the positive edge on the digital input DI.

The saved rather than the current counter value is displayed in the feedback interface (Page 338). The STS_DI bit indicates the level of the latch signal.

Direct loading of the counter does not cause the indicated stored counter status to be changed.

In isochronous mode, the counter status that was latched at the time of the last positive edge before T_i is displayed in the feedback interface.

When you close the SW gate, the effect is either canceling or interrupting, depending on the parameter assignment. The digital input DI remains active even when the software gate is closed.

Extended feedback interface

If the 1Count24V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current counter value can be read from bytes 8 to 11 of the feedback interface (Page 338).

Selecting hardware gate input signal (S7-300, S7-400)

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 338)STS_DI indicates the level of the digital input.

Selecting synchronization (S7-300, S7-400)

Synchronization

If you have selected "Synchronization with positive edge" at the "DI function (Page 331)" parameter, this is where you can select either once-only or periodic synchronization:

- Once-only: If the enable bit is set, the first edge loads the 1Count24V with the load value.
- Periodic: If the enable bit is set, the first and each subsequent edge loads 1Count24V with the load value.

Specifying the high count limit (S7-300, S7-400)

Description

By configuring an high count limit together with a main count direction (Page 185) (up or down), you can reduce the maximum counting range. The parameterized counting range is then between 0 and the high count limit parameterized.

The high count limit can be any value between > 1 and the high count range limit.

Control and feedback interface for counting modes (S7-300, S7-400)

Control interface for counting modes (S7-300, S7-400)

Note

The following control interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count24V control interface (outputs) for counting modes.

Address		Assignment			
Bytes 0 to 3		**Load value direct, preparatory, comparison value 1 or 2**			
	Byte 0	**Behavior of DO1, DO2 of 1Count24V**			
		Bit 2	Bit 1	Bit 0	Function DO1
		0	0	0	Output
		0	0	1	Turn on at counter status ≥ comparison value
		0	1	0	Turn on at counter status ≤ comparison value
		0	1	1	Pulse on reaching the comparison value
		1	0	0	Switch at comparison values
		1	0	1	blocked
		1	1	0	blocked
		1	1	1	blocked
			Bit 5	Bit 4	Function DO2
			0	0	Output
			0	1	Turn on when counter status ≥ comparison value
			1	0	Turn on when counter status ≤ comparison value
			1	1	Pulse on reaching the comparison value
		Bits 3, 6, and 7: Reserved = 0			
	Bytes 1 to 3	Byte 1	Hysteresis DO1, DO2 (value range 0 to 255)		
		Byte 2	Pulse duration DO1, DO2 (value range 0 to 255)		
		Byte 3	Reserved = 0		
					Name
Byte 4		Bit 7	Error acknowledgment		EXTF_ACK
		Bit 6	Enable DO2		CTRL_DO2
		Bit 5	Control bit DO2		SET_DO2
		Bit 4	Enable DO1		CTRL_DO1
		Bit 3	Control bit DO1		SET_DO1
		Bit 2	Trigger status bits reset		RES_STS
		Bit 1	Enable synchronization		CTRL_SYN
		Bit 0	Software gate control bit		SW_GATE
Byte 5		Bit 7	Reserved = 0		
		Bit 6	Reserved = 0		
		Bit 5	Reserved = 0		
		Bit 4	Change function and behavior of DO1, DO2		C_DOPARAM
		Bit 3	Load comparison value 2		CMP_VAL2
		Bit 2	Load comparison value 1		CMP_VAL1
		Bit 1	Load counter in preparation		LOAD_PREPARE
		Bit 0	Load counter direct		LOAD_VAL
Bytes 6 to 7		Reserved = 0 [1]			

[1] Not used for extended user interface

Notes on the Control Bits

Bit	Explanation
C_DOPARAM	Change function and behavior of DO1, DO2 The values from bytes 0 to 2 are applied as new function, hysteresis, and pulse duration of DO1, DO2. This may result in the following error: The conditions for the "Switch at comparison values" behavior are not fulfilled.
CMP_VAL1	Load comparison value 1 The value from bytes 0 to 3 is transferred to comparison value 1 with the control bit "Load comparison value 1" (CMP_VAL1).
CMP_VAL2	Load comparison value 2 The value from bytes 0 to 3 is transferred to comparison value 2 with the control bit "Load comparison value 2" (CMP_VAL2).
CTRL_DO1	Enable DO1 You use this bit to enable the DO1 output.
CTRL_DO2	Enable DO2 You use this bit to enable the DO2 output.
CTRL_SYN	You use this bit to enable synchronization.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load counter in preparation The value from bytes 0 to 3 is applied as the load value.
LOAD_VAL	The value from bytes 0 to 3 is loaded directly as the new counter value.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SET_DO1	Control bit DO1 Switches the DO1 digital output on and off when CRTL_DO1 is set.
SET_DO2	Control bit DO2 Switches the DO2 digital output on and off when CRTL_DO2 is set.
SW_GATE	Software gate control bit The software gate is opened/closed via the control interface with the SW_GATE bit.

Feedback interface for counting modes (S7-300, S7-400)

Note

The following feedback interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count24V feedback interface (inputs) for counting modes.

Address	Assignment		Name
Bytes 0 to 3	Count value or stored counter value in the case of the latch function on the digital input		
Byte 4	Bit 7	Encoder supply short circuit	ERR_24V
	Bit 6	Short circuit / wire break / overtemperature	ERR_DO1
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Status down	STS_C_DN
	Bit 6	Status up	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	DO2 status	STS_DO2
	Bit 3	DO1 status	STS_DO1
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 7	Zero-crossing in the count range when counting without a main count direction	STS_ND
	Bit 6	Low count limit	STS_UFLW
	Bit 5	High count limit	STS_OFLW
	Bit 4	Comparator 2 status	STS_CMP2
	Bit 3	Comparator 1 status	STS_CMP1
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Synchronization status	STS_SYN
Byte 7	Reserved = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the Feedback Bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged with the control bit (Page 336)EXTF_ACK Diagnostic alarm if parameterized.
ERR_DO1	Short circuit/wire break/overtemperature due to overload at output DO1 The error bit must be acknowledged with the control bit (Page 336)EXTF_ACK Diagnostic alarm if parameterized.

Bit	Explanation
ERR_LOAD	Load function error The LOAD_VAL, LOAD_PREPARE, CMP_VAL1, CMP_VAL2 and C_DOPARAMcontrol bits (Page 336) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_CMP1	Comparator 1 status The STS_CMP1 status bit indicates that the output is or was switched on. It must be acknowledged with the control bit (Page 336)RES_STS. If the status bit is acknowledged when the output is still switched on, the bit is set again immediately. This bit is also set if the SET_DO1 control bit is used when DO1 is not enabled.
STS_CMP2	Comparator 2 status The STS_CMP2 status bit indicates that the output is or was switched on. It must be acknowledged with the control bit (Page 336)RES_STS. If the status bit is acknowledged when the output is still switched on, the bit is set again immediately. This bit is also set if the SET_DO2 control bit is operated when DO2 is not enabled.
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_DO1	DO1 status The STS_DO1 status bit indicates the status of the DO1 digital output.
STS_DO2	DO2 status The STS_DO2 status bit indicates the status of the virtual DO2 digital output.
STS_GATE	Internal gate status: Counting.
STS_LOAD	Load function is running
STS_ND	Zero-crossing in the count range when counting without a main count direction. The bit must be reset by the control bit (Page 336)RES_STS.
STS_OFLW STS_UFLW	High count limit violated Low count limit violated Both bits must be reset.
STS_SYN	Synchronization status The STS_SYN bit is set after successful synchronization. It must be reset by the control bit (Page 336)RES_STS.

Measurement parameters (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP (S7-300, S7-400)

Setting response to CPU/Master STOP

You can configure the response of the 1Count24V to failure of the higher-level controller.

Parameter	Status of the 1Count24V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO1	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode will continue working. The gate and digital output status will not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.
DO 1 substitute a value	The current mode is cancelled. The gate is closed and the substitute value set for the digital output is activated. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters. If "Pulse when comparison value is reached" has been set for the output, the only available pulse duration substitute value will be 1.	The changed parameters are accepted and become effective.
DO 1 keep last value	The current mode is cancelled. The gate is closed and the status of the digital output is maintained. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working upon transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count24V.

Leaving the Assigned State

Under what conditions does the 1Count24V leave the assigned state?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 354).

Automatic New Parameter Assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon power on of the CPU/DP master
- Upon IM 151-1 / IM 151-1 FO power on

- After failure of the DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count24V is plugged
- Upon power on or inserting of the appropriate power module

Signal evaluation (S7-300, S7-400)

Selecting count and direction evaluation (S7-300, S7-400)

Signal evaluation A*, B*

Signal evaluation by means of A*, B* allows direction-specific counting. Different evaluation modes are possible depending on the parameter assignment:

- Pulse and Direction (Page 343)
  In the case of 24 V pulse generators with a direction level, there must be an interval of at least 5 µs/50 µs (depending on the input filter set) between the direction signal (B*) and the count signal (A*).
  The following figure shows the interval between direction signal and count signal.

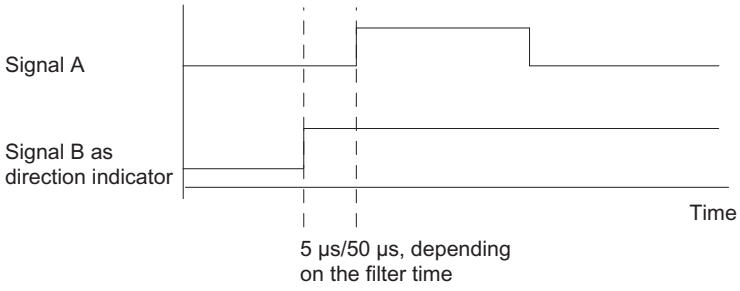

- Rotary Encoder
  If you connect a 24 V rotary encoder with two tracks that are 90 degrees out of phase at the count and direction inputs, you can configure single evaluation (Page 206) in all measuring modes.
  You can invert the count direction (Page 348) at direction input B*.
  A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with 1Count24V for the "Encoder inputs" parameter, you must use the M-switching sensors.

Note

The count frequency of 100 kHz is the maximum frequency of the A* and B* signals.

"Pulse and direction" evaluation (S7-300, S7-400)

Pulse and direction

The level at direction input B* is used as the direction setting in this type of evaluation.

An unwired input corresponds to the "Up" count direction if you have selected "Pulse and direction" for the "Signal evaluation" parameter.

The diagram below shows the signals of a 24 V pulse generator with direction level.

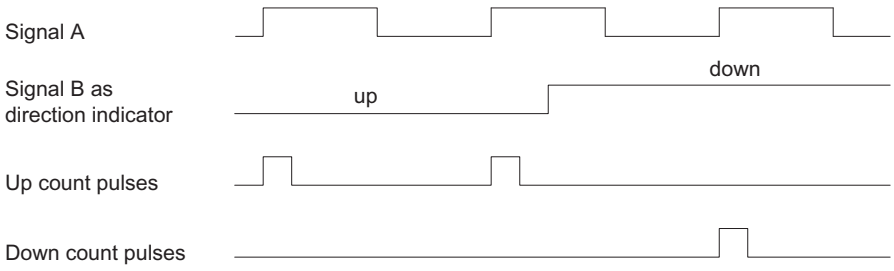

Measuring methods (S7-300, S7-400)

Overview (S7-300, S7-400)

Measuring methods

You can choose from the following measuring methods for the "Measuring procedure" parameter:

- With integration time (Page 343)
- Continuous (Page 344)

Measurement with integration time (S7-300, S7-400)

Measurement with integration time

The measurement is carried out during the assigned integration time. When the integration time elapses, the measured value is updated.

The end of a measurement is indicated by the STS_CMP1status bit (Page 356). This bit is reset by the control bit (Page 354)RES_STS in the control interface.

If there were not at least two rising edges in the assigned integration time, 0 is returned as the measured value.

A value of -1 is returned up until the end of the first integration time.

You can change the integration time for the next measurement during operation.

Direction Reversal

If the direction of rotation is reversed during integration time, the measured value for this measurement period will be undefined. Evaluating the STS_C_UP and STS_C_DNfeedback bits (Page 356) (direction evaluation) allows you to respond to any process irregularities.

See also

Integration Time and Update Time in Isochronous Mode (Page 347)

Continuous measurement (S7-300, S7-400)

Measuring Principle

The 1Count24V counts each positive edge of a pulse and assigns it a time value in µs.

The update time indicates the time interval at which the measured value is updated by the module in the feedback interface.

Rule for pulse sequences with one or several pulses per update time:

Dynamic measuring time =Time of last pulse in the current update time interval
minus
Time of last pulse in the previous update time interval

When the update time has elapsed, a new measured value is calculated and output with the dynamic measuring time.

The following dynamic measuring time will result if the current update time does not contain a pulse:

Dynamic measuring time =Time of current, elapsed update time
minus
Time of last pulse

When the update time has elapsed, an estimated measured value is calculated with the dynamic measuring time under the assumption that a pulse occurred at the end of the update time.

If the "1 Pulse per dynamic measuring time" estimated measured value is less than the last measured value during the frequency and speed measurement, this estimated measured value is output as the new measured value. With the period measurement, the dynamic measuring time is output as the estimated period if the dynamic measuring time is greater than the last measured period.

The figure below shows the measuring principle.

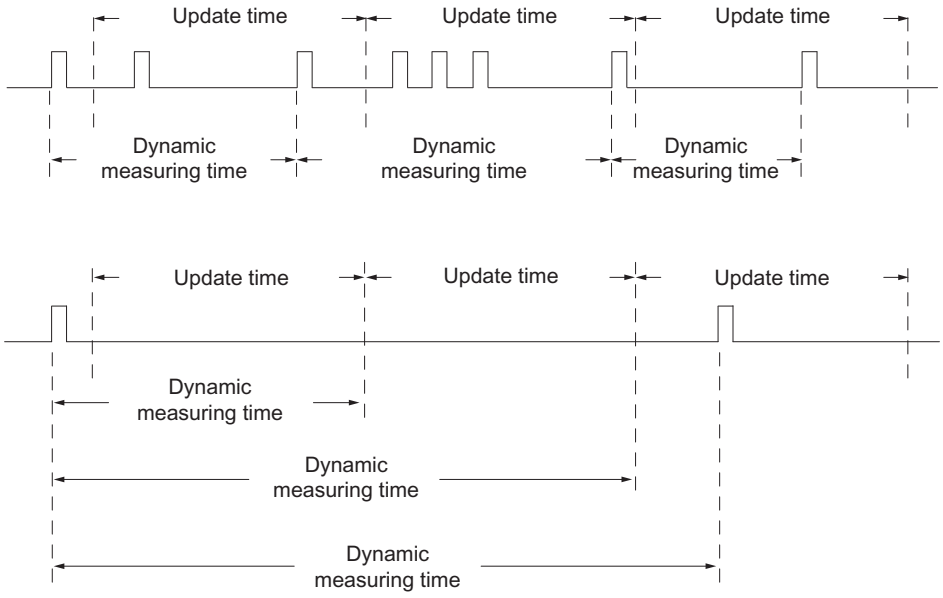

The 1Count24V measures continuously. When assigning parameters, you specify an update time.

During the time until the end of the first elapsed update time, a value of "-1" is returned.

The continuous measurement begins after the gate is opened with the first pulse of the pulse train to be measured. The first measured value can be calculated after the second pulse, at the earliest.

A measured value (frequency, period, or speed) is output in the feedback interface each time the update time elapses. The end of a measurement is indicated by the STS_CMP1status bit (Page 356). This bit is reset with the RES_STS and RES_STS_A bits by applying the complete acknowledgement principle.

If the direction of rotation is reversed during an update time, the measured value for this measurement period will be undefined. Evaluating the STS_C_DN and STS_C_UPfeedback bits (Page 356) (direction evaluation) enables you to respond to any process irregularities.

The following figure illustrates the principle of continuous measurement taking frequency measurement as an example.

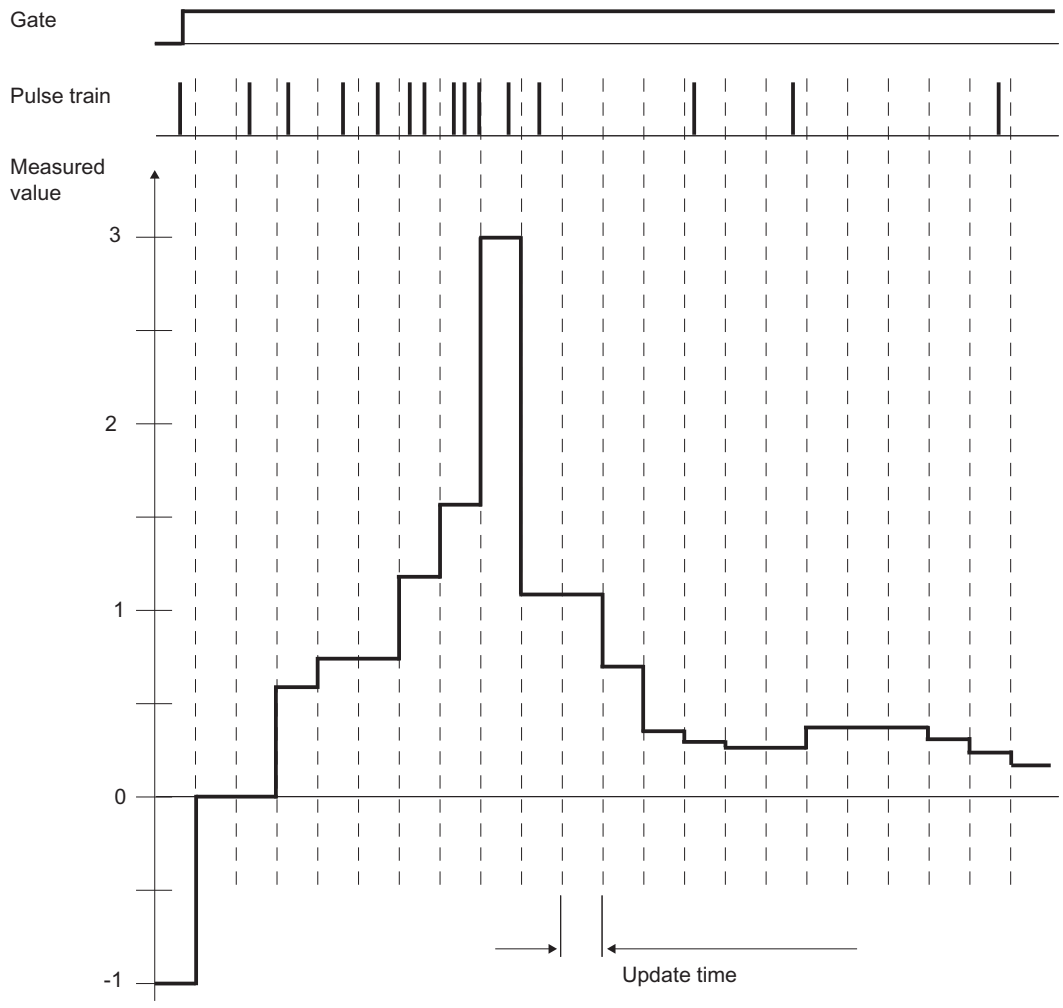

Gate Control

To control the 1Count24V, you have to use the gate functions.

Isochronous mode

In isochronous mode, the 1Count24V accepts control bits and control values from the control interface in each bus cycle and reports back the response in the same cycle.

In each cycle, the 1Count24V transfers a measured value and the status bits that were valid at time T_i.

The measurement starts and ends at time T_i.

See also

Integration Time and Update Time in Isochronous Mode (Page 347)

Integration Time and Update Time in Isochronous Mode (S7-300, S7-400)

Integration time and update time in isochrone mode

If the integration time/update time lasts several T_{DP} cycles, the new measured value will be indicated in the user program by the STS_CMP1status bit (Page 356) (measurement completed) of the feedback interface. This enables monitoring of the measuring operation or a synchronization with the measuring operation. It takes 4 T_{DP} cycles, however, for this alarm to be acknowledged. The minimum integration time/update time in this case is (4 x T_{DP}).

You do not need to continually evaluate status bit STS_CMP1 if the application can tolerate a jitter in the integration time/update time of T_{DP} and a measured value that remains constant for several cycles. Integration times/update times of (1 x T_{DP}) to (3 x T_{DP}) are then possible.

A loss of isochrone operation in the last T_{DP} cycle of the integration time will increase the integration time/update time by one T_{DP} cycle. This does not corrupt the measured value.

Note

The value range limits for the integration time/update time may not be exceeded (see tables for the individual measuring modes).

Violation of the value range limits will result in a parameter assignment error and 1Count24V will not go into isochrone mode.

Note

You must always adjust the integration time/update time parameter if you want to retain the integration time/update time length when you change the configuration from non-isochrone to isochrone mode and vice versa.

Selecting filters for input signals (S7-300, S7-400)

Input filter

You can activate a filter to filter signals at count input A*, direction input B* and digital input DI in line with the minimum pulse duration or the maximum signal frequency.

You can select a minimum pulse width of 2.5 µs or 25 µs for each input.

Selecting encoder inputs (S7-300, S7-400)

Encoder inputs

A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with for the "Encoder inputs" parameter, you must use the M-switching sensors.

Selecting the count direction (S7-300, S7-400)

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Behavior of the output in measuring modes (S7-300, S7-400)

Selecting behavior of output DO1 (S7-300, S7-400)

Behavior of the Output in Measuring Modes

You can assign parameters for the digital output of the 1Count24V.

You can store a high and a low limit for frequency measurement, rotational speed measurement or period measurement. If the limits are violated, digital output DO1 is activated. These limit values can be assigned and changed with the load function.

You can change the function and the behavior of the digital output during operation. The new function takes effect immediately. In isochronous mode it always takes effect at time T_i.

You can choose between the following functions in the drop-down list box.

- Output (Page 349)
- Outside limits (Page 349) (limit monitoring)
- Violating low limit (Page 349) (limit monitoring)
- Exceeding high limit (Page 349) (limit monitoring)

Output (S7-300, S7-400)

Output

If you want to switch the output on or off, you must enable it with the CTRL_DO1control bit (Page 354).

You can switch the output on and off with the SET_DO1control bit (Page 354).

You can query the status of the output with the STS_DO1status bit (Page 356) in the feedback interface.

In isochronous mode, the output is switched at time T_o.

Limit-Value Monitoring (S7-300, S7-400)

Limit monitoring

The figure below illustrates limit monitoring.

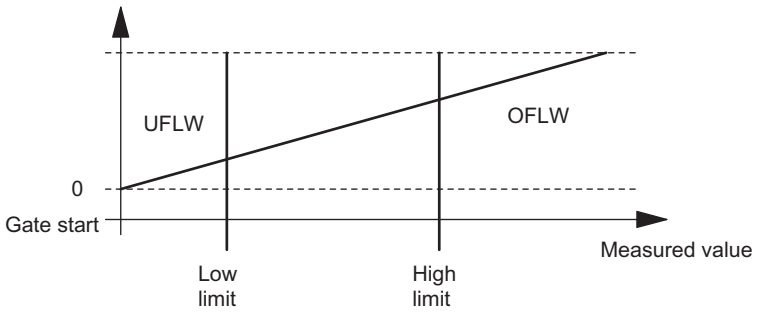

When the integration time has elapsed, the measured value obtained (frequency, rotational speed or period) is compared with the assigned limits.

The STS_UFLW = 1 bit will be set in the feedback (Page 356) interface if the current measured value is below the parameterized low limit (measured value < low limit).

The STS_OFLW = 1 bit will be set in the feedback (Page 356) interface if the current measured value is above the parameterized high limit (measured value > high limit).

You must acknowledge these bits with the control bit (Page 354)RES_STS.

The corresponding status bit will be set again if the measured value still or once again violates the limits after acknowledgment.

Setting the low limit to 0 switches off dynamic monitoring of "Violating low limit".

The table below shows how the enabled digital output DO1 can be set by limit monitoring in accordance with parameter assignment:

"Function DO1" parameter	DO1 is Set ...
Outside limits	Measured value < low limit OR measured value > high limit
Violating low limit	Measured value < low limit
Exceeding high limit	Measured value > high limit

In isochronous mode the output is switched at the end of measurement at time T_i.

Selecting substitute value DO1 (S7-300, S7-400)

Substitute value DO1

This is where you select the substitute value which is to be output at digital output DO1 if you have selected the parameter "DO1substitute a value" under "Reaction to CPU/Master STOP (Page 341)".

Activating DO1 diagnostics (S7-300, S7-400)

DO1 diagnostics

"DO1 diagnostics" (wire break, short circuit) are possible only with pulse lengths of > 90 ms at digital output DO1.

Select the check box for diagnostics at output DO1.

Selecting a measuring mode (S7-300, S7-400)

Description

You define 1Count24V functionality by specifying a measuring mode.

	Description	
Operating mode	Measurement with integration time	Continuous measurement
Frequency measurement (Page 191) (with hardware or software gate)	1Count24V counts the pulses that occur during a specified integration time.	1Count24V counts the pulses that occur during a dynamic measurement time.
Rotational speed measurement (Page 193) (with hardware or software gate)	1Count24V counts the pulses received from a tachometer generator within a specified integration time and calculates the speed on the basis of this value and the number of pulses per encoder revolution.	1Count24V counts the pulses received from a tachometer generator within a dynamic measuring time and calculates the speed on the basis of this value and the number of pulses per encoder revolution.
Period measurement (Page 194) (with hardware or software gate)	1Count24V measures the time between two positive edges of the count signal by counting the pulses of an internal quartz accurate reference frequency (16 MHz) over a specified integration time.	1Count24V indicates the dynamic measuring time as a period. The mean value of the period is calculated if its duration is less than the update time.

The default setting is "Frequency measurement" mode.

Selecting period resolution (S7-300, S7-400)

Description

The value of the calculated period is displayed in 1 µs or 1/16 µs.

Select the required unit of resolution.

Possible Measuring Ranges with Error Indication

- Period measurement with integration time
  - Measuring ranges and errors at a resolution of 1 µs:

Integration time	T_{min} ± absolute error	T_{min} ± absolute error
100 s	1 µs* (10 ± 0)	1 µs* (100,000,000 ± 10,000)
10 s	1 µs* (10 ± 0)	1 µs* (10,000,000 ± 1,000)
1 s	1 µs* (10 ± 0)	1 µs* (1,000,000 ± 100)
0.1 s	1 µs* (10 ± 0)	1 µs* (100,000 ± 10)
0.01 s	1 µs* (10 ± 0)	1 µs* (10,000 ± 1)

  - Measuring ranges and errors at a resolution of 1/16 µs:

Integration time	T_{min} ± absolute error	T_{min} ± absolute error
100 s	1/16 µs* (160 ± 0)	1/16 µs* (1,600,000,000 ± 160,000)
10 s	1/16 µs* (160 ± 0)	1/16 µs* (160,000,000 ± 16,000)
1 s	1/16 µs* (160 ± 0)	1/16 µs* (16,000,000 ± 1,600)
0.1 s	1/16 µs* (160 ± 0)	1/16 µs* (1,600,000 ± 160)
0.01 s	1/16 µs* (160 ± 0)	1/16 µs* (160,000 ± 16)

- Continuous period measurement:
  - Measuring ranges and errors at a resolution of 1 µs:

Period T_{min} ± Absolute error	Period T_{min} ± Absolute error
1 µs* (10 ± 0)	1 µs* (100,000 ± 10)
1 µs* (100 ± 0)	1 µs* (1 000 000 ± 100)
1 µs* (1 000 ± 0)	1 µs* (10 000 000 ± 1 002)
1 µs* (10 000 ± 1)	1 µs* (100,000,000 ± 10,020)

  - Measuring ranges and errors at a resolution of 1/16 µs:

Period T_{min} ± Absolute error	Period T_{min} ± Absolute error
1/16 µs* (160 ± 1)	1/16 µs* (1,600,000 ± 160)
1/16 µs* (1,600 ± 1)	1/16 µs* (16,000,000 ± 1,600)
1/16 µs* (16 000 ± 3)	1/16 µs* (160 000 000 ± 16 000)
1/16 µs* (160 000 ± 20)	1/16 µs* (1,600,000,000 ± 160,000)

Selecting DI function (S7-300, S7-400)

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 201), level-controlled
  The hardware gate is opened and closed by signals at the digital input.

The default setting is "Input".

Selecting hardware gate input signal (S7-300, S7-400)

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 356)STS_DI indicates the level of the digital input.

Setting low and high limit (S7-300, S7-400)

Low and high limit

You can save a high and low limit for frequency measurement, speed measurement and period duration measurement. If these limits are exceeded, digital output DO1 (Page 348) will be activated. You can configure these limits and change them using the load function.

The following value ranges are possible for the low and high limits:

Measuring mode	Low limit (LL)	High limit
Frequency measurement	0 to 99,999,999 × 10^{-3} Hz	LL+1 to 100,000,000 × 10^{-3} Hz
Rotational speed measurement	0 to 24,999,999 × 10^{-3} /min	LL+1 to 25,000,000 × 10^{-3} /min
Period duration measurement (resolution = 1 µs)	0 to 119,999,999 µs	LL+1 to 120,000,000 µs
Period duration measurement (resolution = 1/16 µs)	0 to 1 919 999 999 × 1/16 µs	LL+1 to 1 920 000 000 × 1/16 µs

Specifying measuring time (S7-300, S7-400)

Description

If you have selected a measuring mode, the module will update the measured values on a cyclic basis. The update time/integration time is the product of measuring time x 10ms. You set the measuring time as the factor n.

Detailed descriptions of the value ranges for the measuring time can be found under "Measuring basics (Page 187)".

Entering pulses per encoder revolution (S7-300, S7-400)

Pulses per encoder revolution

For "Rotational speed measurement (Page 193)" mode, you must also configure the pulses per encoder revolution.

The value for the number of pulses per encoder revolution can be found on the rating plate, or in the technical data of your encoder.

Control and feedback interface for measuring modes (S7-300, S7-400)

Control interface for measuring modes (S7-300, S7-400)

Note

The following control interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count24V control interface (outputs) for measuring modes.

Address	Assignment			
Bytes 0 to 3	**Low or high limit**			
	Function of DO1			
	Byte 0	Bit 1	Bit 0	Function DO1
		0	0	Output
		0	1	Outside limits
		1	0	Violating low limit
		1	1	Exceeding high limit
	Bytes 1 to 3	Reserved = 0		
	Integration time			
	Bytes 0, 1	Integration time [n × 10 ms] (Value range 1 to 1000/12000)		
	Bytes 2, 3	Reserved = 0		
				Name

Address	Assignment		
Byte 4	Bit 7	Error acknowledgment	EXTF_ACK
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Enable DO1	CTRL_DO1
	Bit 3	Control bit DO1	SET_DO1
	Bit 2	Trigger status bits reset	RES_STS
	Bit 1	Reserved = 0	
	Bit 0	Software gate control bit	SW_GATE
Byte 5	Bit 7	Reserved = 0	
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Change function of DO1	C_DOPARAM
	Bit 3	Reserved = 0	
	Bit 2	Change integration time	C_INTTIME
	Bit 1	Load high limit	LOAD_PREPARE
	Bit 0	Load low limit	LOAD_VAL
Bytes 6 to 7	Reserved = 0 [1]		

[1] Not used for extended user interface

Notes on the control bits

Bit	Explanation
C_DOPARAM	Change function of DO1 The value from byte 0 is adopted as the new function of DO1.
C_INTTIME	Change integration time The value from bytes 0 and 1 is adopted as the new integration time for the next measurement.
CTRL_DO1	Enable DO1 You use this bit to enable the DO1 output.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load high limit The value from bytes 0 to 3 is applied as the new high limit.
LOAD_VAL	Load low limit The value from bytes 0 to 3 is applied as the new low limit.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SET_DO1	Control bit DO1 Switches the DO1 digital output on and off when CRTL_DO1 is set.
SW_GATE	Software gate control bit The software gate is opened/closed via the control interface with the SW_GATE bit.

Feedback interface for measuring modes (S7-300, S7-400)

Note

The following feedback interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count24V feedback interface (inputs) for measuring modes.

Address	Assignment		Name
Bytes 0 to 3	Measured value		
Byte 4	Bit 7	Encoder supply short circuit	ERR_24V
	Bit 6	Short circuit / wire break / overtemperature	ERR_DO1
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Status down	STS_C_DN
	Bit 6	Status up	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	DO1 status	STS_DO1
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 7	Reserved = 0	
	Bit 6	Low limit, measuring range	STS_UFLW
	Bit 5	High limit, measuring range	STS_OFLW
	Bit 4	Reserved = 0	
	Bit 3	Measurement completed	STS_CMP1
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Reserved = 0	
Byte 7	Reserve = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the feedback bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged with the control bit (Page 354)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_DO1	Short circuit/wire break/overtemperature at output DO1 The error bit must be acknowledged with the control bit (Page 354)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_LOAD	Load function error The LOAD_VAL, LOAD_PREPARE, C_DOPARAM and C_INTTIMEcontrol bits (Page 354) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_CMP1	Measurement completed The measured value is updated each time a time interval (update time/integration time) elapses. **Measurement with integration time:** The end of a measurement (after the interval has elapsed) is indicated by the STS_CMP1 status bit. **Continuous measurement:** The end of the measurement is signaled by the status bit STS_CMP1 at the end of the update time if a measured value is output. The bit remains at "0" if an estimated measured value is output. This bit is reset by the control bit (Page 354)RES_STS in the control interface.
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_DO1	DO1 status
STS_GATE	Internal gate status: Measuring.
STS_LOAD	Load function is running
STS_OFLW STS_UFLW	High limit violated Low limit violated Both bits must be reset.

Position feedback parameters (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP (S7-300, S7-400)

Setting response to CPU/Master STOP

You can configure the response of the 1Count24V to failure of the higher-level controller.

Parameter	Status of the 1Count24V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO1	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count24V.

Exiting parameterized status

Under what conditions does the 1Count24V exit the parameterized status?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 365).

Automatic new parameter assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon CPU/DP master power on
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count24V is inserted
- Upon power on or insertion of the corresponding power module

Signal evaluation (S7-300, S7-400)

Selecting count and direction evaluation (S7-300, S7-400)

Signal evaluation A*, B*

Signal evaluation by means of A*, B* allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Pulse and Direction (Page 359)
  In the case of 24 V pulse generators with a direction level, there must be an interval of at least 5 µs/50 µs (depending on the input filter set) between the direction signal (B*) and the count signal (A*).
  The following figure shows the interval between direction signal and count signal.

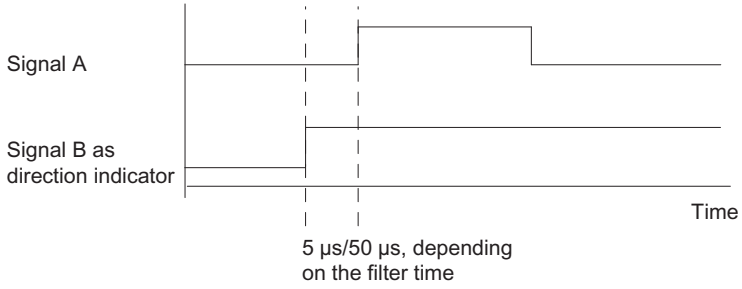

- Rotary Encoder (Page 206)
  If you connect a 24 V rotary encoder with two tracks that are 90 degrees out of phase at the count and direction inputs, you can set single evaluation (Page 206) in all counting modes. You can alternatively set dual (Page 207) or quadruple evaluation (Page 207). You can invert the count direction (Page 360) at direction input B* in all evaluation modes. A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with 1Count24V for the "Encoder inputs" parameter, you must use the M-switching sensors.

Note

The count frequency of 100 kHz is the maximum frequency of the A* and B* signals. The maximum frequency for count pulses is therefore 200 kHz in double evaluation and 400 kHz in quadruple evaluation.

"Pulse and direction" evaluation (S7-300, S7-400)

Pulse and direction

The level at direction input B* is used as the direction setting in this type of evaluation.

An unwired input corresponds to the "Up" count direction if you have selected "Pulse and direction" for the "Signal evaluation" parameter.

The diagram below shows the signals of a 24 V pulse generator with direction level.

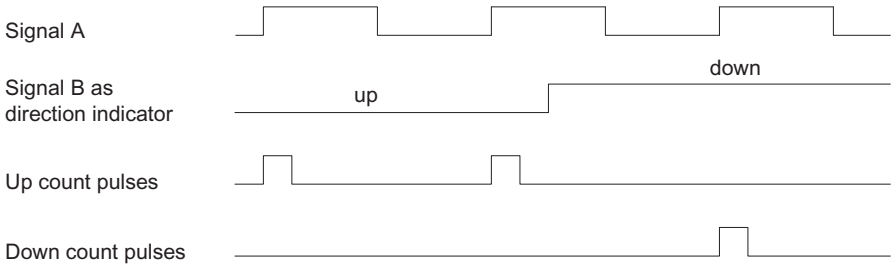

Selecting filters for input signals (S7-300, S7-400)

Input filter

You can activate a filter to filter signals at count input A*, direction input B* and digital input DI in line with the minimum pulse duration or the maximum signal frequency.

You can select a minimum pulse width of 2.5 µs or 25 µs for each input.

Selecting encoder inputs (S7-300, S7-400)

Encoder inputs

A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with for the "Encoder inputs" parameter, you must use the M-switching sensors.

Selecting the count direction (S7-300, S7-400)

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Selecting the gate function (S7-300, S7-400)

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 202): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 202): The counting process starts from the last current counter value after gate stop and restart.

DI function (S7-300, S7-400)

Selecting DI function (S7-300, S7-400)

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 201), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Latch/Retrigger with positive edge (Page 361)
  Counter values are saved (latched) with positive edges at the digital input. The counter is set to the load value each time the values are saved.
- Synchronization with positive edge (Page 362)
  The counter is set to the load value with positive edges at the digital input.
- Latching with positive edge (Page 364)
  Counter values are saved (latched) with positive edges at the digital input.

The default setting is "Input".

Latch/Retrigger with positive edge (S7-300, S7-400)

Latch/Retrigger

This function saves the current internal counter value of the 1Count24V and retriggers counting when there is a positive edge at the digital input. This means that the current internal counter value at the time of the positive edge is stored (latch value), and the 1Count24V is then loaded again with the load value, from which counting resumes.

The following figure shows latch/retrigger when load value = 0.

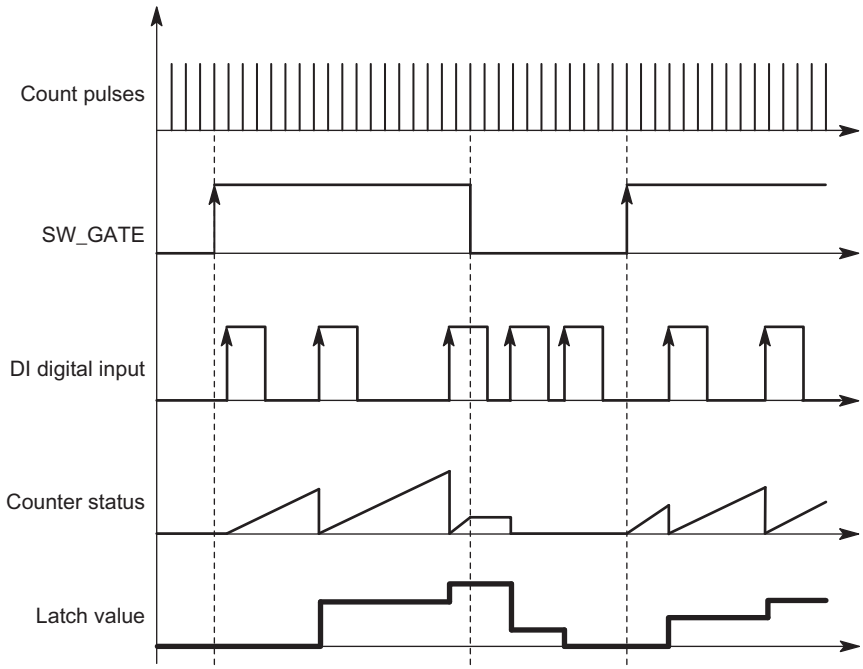

The counting mode must be enabled with the software gate before the function can be executed. It is started with the first positive edge at the digital input.

The saved rather than the current counter value is displayed in the feedback interface (Page 367). The STS_DI bit indicates the status of the latch and retrigger signal.

The latch value is preassigned its RESET status ("0"). It is not changed when the software gate is opened.

Directly loading the counter does not alter the saved counter value displayed.

Closing the software gate only interrupts counting; this means that counting resumes when you open the software gate again. The digital input DI remains active even when the software gate is closed.

Counting is also latched and triggered in isochrone mode with each edge at the digital input. The counter status that was valid at the time of the last edge before T_i is displayed in the feedback interface.

Extended feedback interface

If the 1Count24V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 367).

Synchronization with positive edge (S7-300, S7-400)

Synchronization

If you have configured "Synchronization with positive edge", the rising edge of a reference signal at the input will set 1Count24V to the load value.

You can select between once-only and periodic synchronization ("Synchronization" parameter).

The figure below shows the counter values in once-only and periodic synchronization.

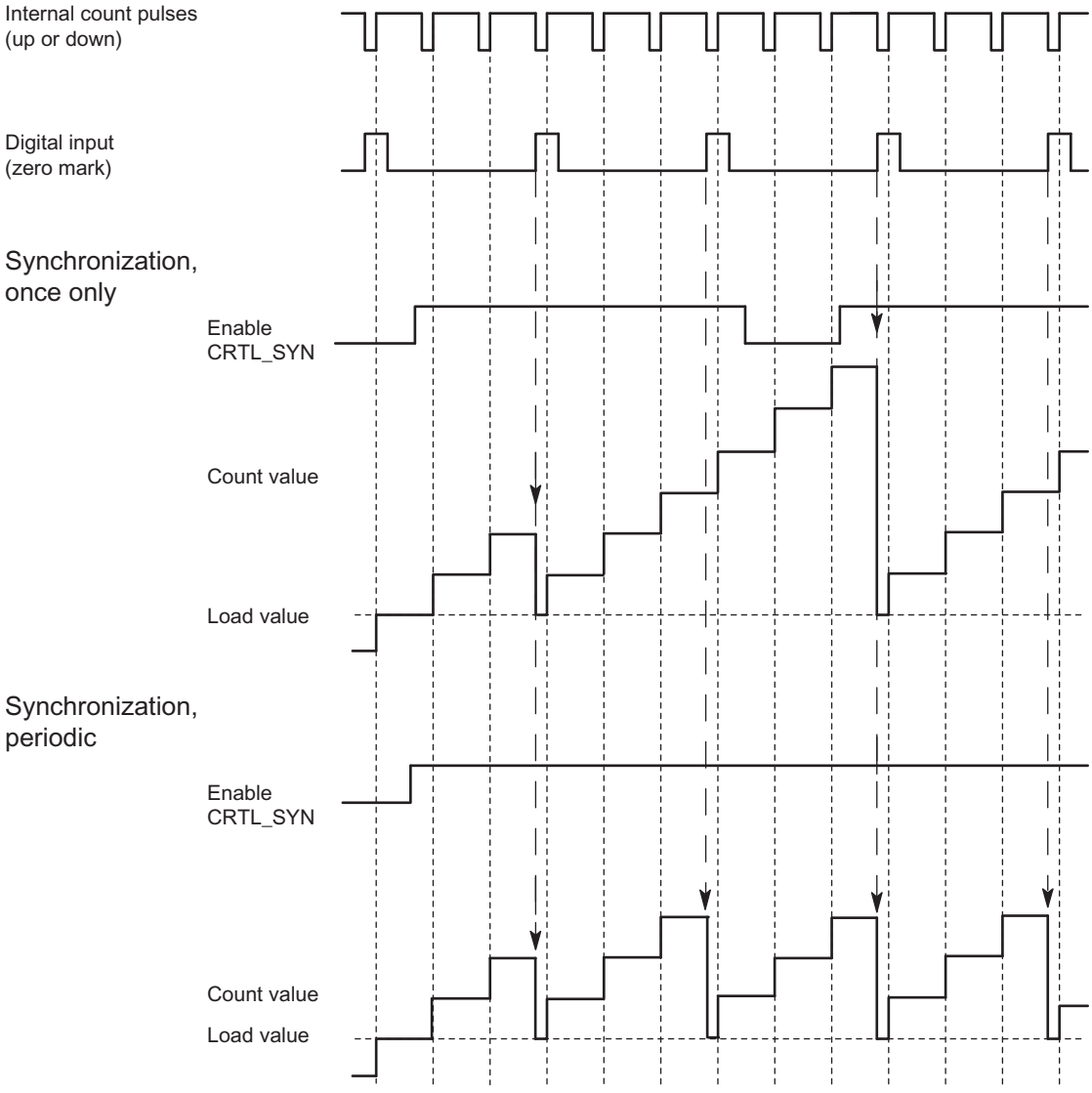

The following conditions apply:

- The counting mode must have been started with the SW gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 365) must be set.
- Synchronize once only: If the enable bit is set, the first edge loads the 1Count24V with the load value.
- Synchronization periodically: If the enable bit is set, the first and each subsequent edge loads the 1Count24V with the load value.
- The feedback bit (Page 367)STS_SYN is set after successful synchronization. It must be reset by the control bit (Page 365)RES_STS.

- The signal of a bounce-free switch or the zero mark of a rotary encoder can serve as the reference signal.
- The feedback bit (Page 367)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 367)STS_SYN indicates that the rising edge at the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Latching with positive edge (S7-300, S7-400)

Latching

Counter value and latch value are pre-assigned their RESET statuses ("0"):

The counting function is started when the software gate is opened. 1Count24V begins at the load value ("0").

The following figure shows latching when load value = 0.

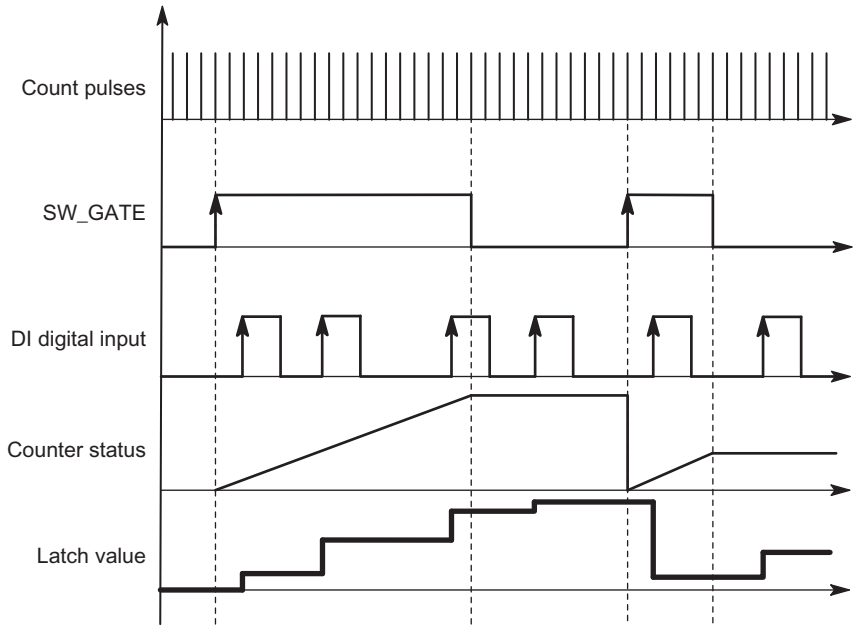

The latch value is always the exact counter value at the time of the positive edge at the digital input DI.

The saved rather than the current counter value is displayed in the feedback interface (Page 367). The STS_DI bit indicates the level of the latch signal.

Directly loading the counter does not alter the saved counter value displayed.

In isochrone mode, the counter status that was latched at the time of the last positive edge before T_i is displayed in the feedback interface.

Closing the software gate will either cancel or interrupt depending on the parameter assignment. The digital input DI remains active even when the software gate is closed.

Extended feedback interface

If the 1Count24V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 367).

Selecting hardware gate input signal (S7-300, S7-400)

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 367)STS_DI indicates the level of the digital input.

Selecting synchronization (S7-300, S7-400)

Synchronization

If you have selected "Synchronization with positive edge" at the "DI function (Page 361)" parameter, this is where you can select either once-only or periodic synchronization:

- Once-only: If the enable bit is set, the first edge loads the 1Count24V with the load value.
- Periodic: If the enable bit is set, the first and each subsequent edge loads 1Count24V with the load value.

Control and feedback interface for position feedback (S7-300, S7-400)

Position feedback control interface (S7-300, S7-400)

Note

The following control interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count24V control interface (outputs) for position feedback.

Address	Assignment		Name
Bytes 0 to 3	Load value directly, in preparation		
Byte 4	Bit 7	Error acknowledgment	EXTF_ACK
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Trigger status bits reset	RES_STS
	Bit 1	Enable synchronization	CTRL_SYN
	Bit 0	Software gate control bit	SW_GATE
Byte 5	Bit 7	Reserved = 0	
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	Load counter in preparation	LOAD_PREPARE
	Bit 0	Load counter directly	LOAD_VAL
Bytes 6 to 7	Reserved = 0 [1]		

[1] Not used for extended user interface

Notes on the control bits

Bit	Explanation
CTRL_SYN	You use this to enable synchronization.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load counter in preparation The value from bytes 0 to 3 is applied as the load value.
LOAD_VAL	The value from bytes 0 to 3 is loaded directly as the new counter value.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SW_GATE	The software gate is opened/closed via the control interface with the SW_GATE bit.

Position feedback feedback interface (S7-300, S7-400)

Note

The following feedback interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count24V feedback interface (inputs) for position feedback.

Address	Assignment		Name
Bytes 0 to 3	Counter value or stored counter value in the case of the latch function at the digital input		
Byte 4	Bit 7	Encoder supply short circuit	ERR_24V
	Bit 6	Reserved = 0	
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Status down	STS_C_DN
	Bit 6	Status up	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 7	Zero pass	STS_ND
	Bit 6	Low count limit	STS_UFLW
	Bit 5	High count limit	STS_OFLW
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Synchronization status	STS_SYN
Byte 7	Reserved = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the feedback bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged with the control bit (Page 365)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_LOAD	Load function error TheLOAD_VAL and LOAD_PREPAREcontrol bits (Page 365) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_GATE	Internal gate status: Counting.
STS_LOAD	Load function is running
STS_ND	Zero pass in the counting range during counting without a main count direction. The bit must be reset by the control bit (Page 365)RES_STS.
STS_OFLW STS_UFLW	High count limit exceeded Low count limit violated Both bits must be reset.
STS_SYN	Synchronization status: The STS_SYN bit is set after successful synchronization. It must be reset by the control bit (Page 365)RES_STS.

Fast mode parameters (S7-300, S7-400)

Selecting reaction to CPU/Master STOP (S7-300, S7-400)

Setting response to CPU/Master STOP

You can configure the response of the 1Count24V to failure of the higher-level controller.

Parameter	Status of the 1Count24V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO1	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count24V.

Automatic new parameter assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon CPU/DP master power on
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count24V is inserted
- Upon power on or insertion of the corresponding power module

Signal evaluation (S7-300, S7-400)

Selecting count and direction evaluation (S7-300, S7-400)

Signal evaluation A*, B*

Signal evaluation by means of A*, B* allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Pulse and Direction (Page 370)
  In the case of 24 V pulse generators with a direction level, there must be an interval of at least 5 µs/50 µs (depending on the input filter set) between the direction signal (B*) and the count signal (A*).
  The following figure shows the interval between direction signal and count signal.

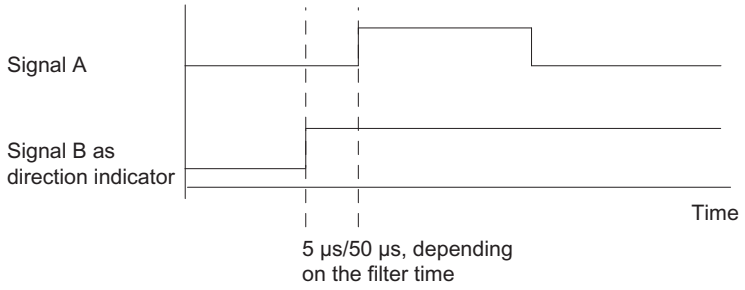

- Rotary Encoder (Page 206)
  If you connect a 24 V rotary encoder with two tracks that are 90 degrees out of phase at the count and direction inputs, you can set single evaluation (Page 206) in all counting modes. You can alternatively set dual (Page 207) or quadruple evaluation (Page 207). You can invert the count direction (Page 371) at direction input B* in all evaluation modes. A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with 1Count24V for the "Encoder inputs" parameter, you must use the M-switching sensors.

Note

The count frequency of 100 kHz is the maximum frequency of the A* and B* signals. The maximum frequency for count pulses is therefore 200 kHz in double evaluation and 400 kHz in quadruple evaluation.

"Pulse and direction" evaluation (S7-300, S7-400)

Pulse and direction

The level at direction input B* is used as the direction setting in this type of evaluation.

An unwired input corresponds to the "Up" count direction if you have selected "Pulse and direction" for the "Signal evaluation" parameter.

The diagram below shows the signals of a 24 V pulse generator with direction level.

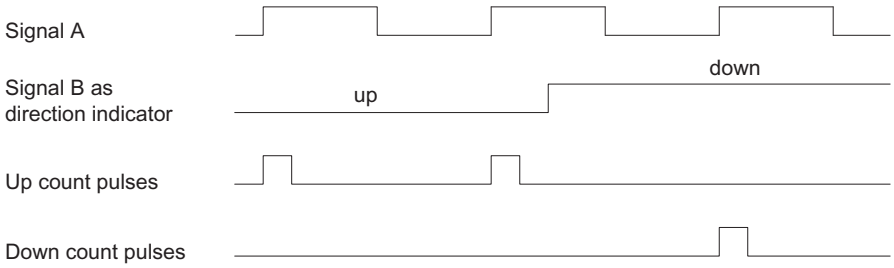

Selecting filters for input signals (S7-300, S7-400)

Input filter

You can activate a filter to filter signals at count input A*, direction input B* and digital input DI in line with the minimum pulse duration or the maximum signal frequency.

You can select a minimum pulse width of 2.5 µs or 25 µs for each input.

Selecting encoder inputs (S7-300, S7-400)

Encoder inputs

A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with for the "Encoder inputs" parameter, you must use the M-switching sensors.

Selecting the count direction (S7-300, S7-400)

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Selecting the gate function (S7-300, S7-400)

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 202): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 202): The counting process starts from the last current counter value after gate stop and restart.

DI function (S7-300, S7-400)

Selecting DI function (S7-300, S7-400)

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 201), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Synchronization with positive edge (Page 372)
  The counter is set to the load value with positive edges at the digital input.

The default setting is "Input".

Synchronization with positive edge (S7-300, S7-400)

Synchronization

If you have configured "Synchronization with positive edge", the positive edge of a reference signal at the input will set 1Count24V to the initial value (= load value).

The figure below shows the counter values for synchronization with a positive edge.

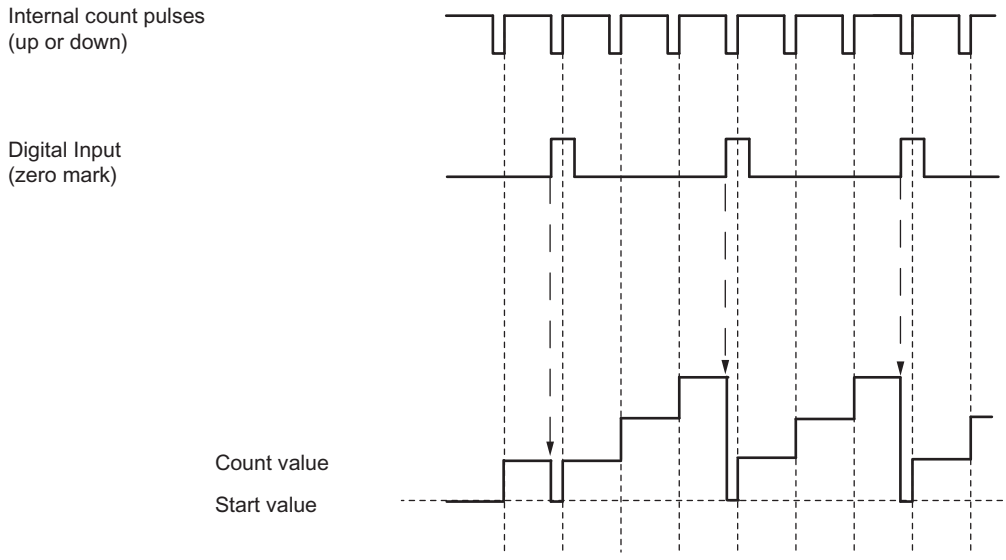

The following conditions apply:

- "Fast mode" must be active (hardware gate).
  - When synchronization is activated, the first edge and each additional edge loads 1Count24V with the initial value (=load value).
- The signal of a bounce-free switch or the zero mark of a rotary encoder can serve as the reference signal.
- The STS_DI feedback bit indicates the level of the reference signal.

Selecting hardware gate input signal (S7-300, S7-400)

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 374)STS_DI indicates the level of the digital input.

Specifying load value (S7-300, S7-400)

Load value

You can specify a load value for 1Count24V.

This load value is applied directly as the initial value.

Valid value range: -16777216 to +16777215

Fast mode feedback interface (S7-300, S7-400)

Note

The following feedback interface data is consistent for 1Count24V:

- Bytes 0 ... 3

Feedback interface assignment

The table below shows the assignment of the 1Count24V feedback interface (inputs) for fast mode.

Address	Assignment		Name
Bytes 0 to 3	Bit 31	Lifebeat	LZ
	Bit 30	Isochronous mode applied	STS_TIC
	Bit 29	Parameter assignment error	ERR_PARA
	Bit 28	Group error	EXTF
	Bit 27	DI status	STS_DI
	Bit 26	Status of direction up/down	STS_DIR
	Bit 25	Internal gate status	STS_GATE
	Bits 0 to 24	Count value	

Notes on the Feedback Bits

Bit	Explanation
ERR_PARA	Errors in assigned module parameters.
EXTF	Group error Possible cause: • Encoder supply short circuit EXTF is reset when the causes of the errors have been eliminated.
LZ	The lifebeat is toggled each time the feedback interface is updated, i.e. the last value sent is inverted.
STS_DI	The bit displays the status of the digital input DI.
STS_DIR	Direction status: • for encoder value change from larger to smaller encoder positions (including zero pass) → "1 " • for encoder value change from smaller to larger encoder positions (including zero pass) → "0 "
STS_GATE	Internal gate status: Counting.
STS_TIC	Isochronous mode (if assigned) was applied.

1Count24V diagnostics (S7-300, S7-400)

Diagnostics using the LED display (S7-300, S7-400)

LED display on the 1Count24V

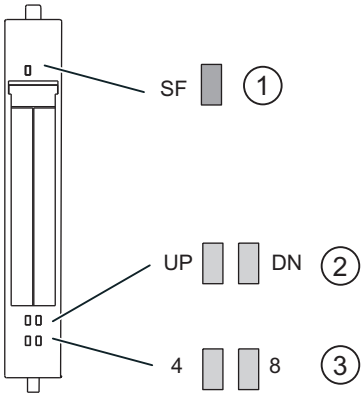

① Group error (red)
② Status display for counting direction (green)
③ Status display for digital input/digital output (green)

Status and error displays by means of LEDs on the 1Count24V

The table below shows the status and fault displays on the 1Count24V.

Event (LED)					Cause	Remedy
SF	UP	DN	4	8		
On					No configuration or incorrect module plugged in. There is a diagnostic message.	Check the parameter assignment. Evaluate the diagnostics data.
	On				Status of the low-order bits of the counter, if the counter counts up.	
		On			Status of the inverse low-order bits of the counter, if the counter counts down.	
			On		DO (direct control, comparator output) activated.	
				On	DI (HW gate, synchronization, latch) activated.	

Error types (S7-300, S7-400)

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

1Count24V error types

The following table shows the 1Count24V error types.

Error type		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the sensor supply or the actuator.	Check the wiring to the sensor. Correct the process wiring.
5_D	00101: Overtemperature	The digital output is overloaded.	Correct the process wiring.
6_D	00110: Open circuit	Line to the actuator is interrupted.	Correct the process wiring.
9_D	01001: Error	Internal module error occurred.	Replace the module.
		Load voltage from the power module is too low.	Correct the process wiring. Check the load voltage.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.

1.2.1.6 Using ET 200S 1Count5V (S7-300, S7-400)

1Count5V terminal assignment (S7-300, S7-400)

Wiring Requirements

The cables (terminals 1 and 8 and terminals 15 and 16) must be shielded. The shield must be supported at both ends. To do this, use the shield support (see the appendix to the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal assignment of the 1Count5V

The following table shows the terminal assignment for 1Count5V.

View	Terminal Assignment	Remarks
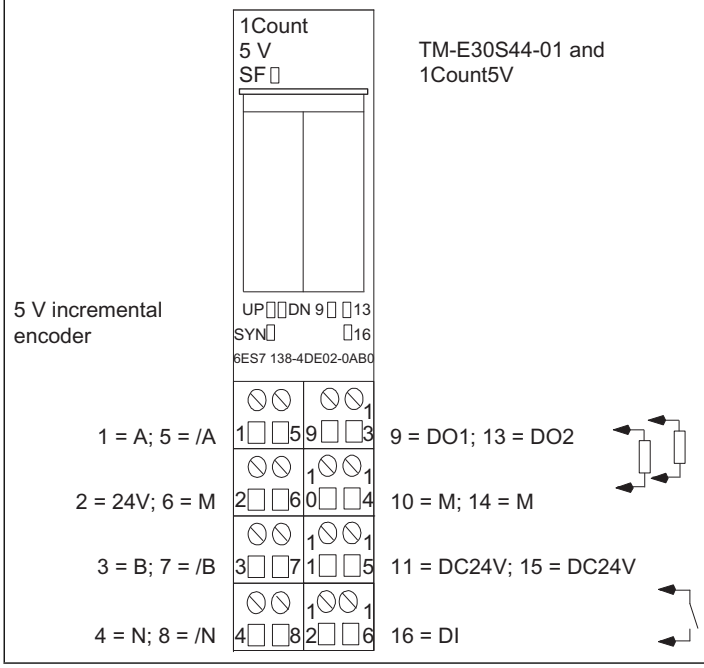		A, /A: Track A B, /B: Track B N, /N: Track N 24V: Load voltage 24V DC: Sensor supply M: Chassis ground DI: Digital input DO1: Digital output DO2: Digital output

Configuring and assigning parameters to 1Count5V (S7-300, S7-400)

Operation (S7-300, S7-400)

Selecting an operating mode (S7-300, S7-400)

Operating mode

You define 1Count5V functionality by specifying an operating mode.

Operating mode	Description
Counting (Page 378)	Preselection for counting modes
Measuring (Page 402)	Preselection for measuring modes

Operating mode	Description
Position feedback (Page 417)	In this mode, 1Count5V counts continuously from the load value: • If another count pulse is received after 1Count5V reaches the high count limit when counting up, it will jump to the low count limit and continue counting from there without losing a pulse. • If another count pulse is received after 1Count5V reaches the low count limit when counting down, it will jump to the high count limit and continue counting from there without losing a pulse.
Fast mode (Page 430) (supported as of firmware version V2.0)	In this mode, 1Count5V counts continuously from the initial value: • If another count pulse is received after 1Count5V reaches the maximum value that can be represented by 25 bits when counting up (all counter bits are set), the count value will jump to "0" and resume counting from there without losing a pulse. • If 1Count5V reaches the value "0" when counting down and another count pulse is then received, the count value will jump to the maximum value that can be represented by 25 bits (all counter bits set) and resume counting from there without losing a pulse. This operating mode can be used for position feedback in particularly short isochrone cycles.

Counting parameters (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP (S7-300, S7-400)

Setting response to CPU/Master STOP

You can configure the response of the 1Count5V to failure of the higher-level controller.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1)]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
DO substitute a value	The current mode is cancelled. The gate is closed and the substitute value set for the digital output is activated. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters. If "Pulse when comparison value is reached" has been set for the output, the only available pulse duration substitute value will be 1.	The changed parameters are accepted and become effective.
DO keep last value	The current mode is cancelled. The gate is closed and the status of the digital output is maintained. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.

1) The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count5V.

Exiting parameterized status

Under what conditions does the 1Count5V exit the parameterized status?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 397).

Automatic new parameter assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon CPU/DP master power on
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count5V is inserted
- Upon power on or insertion of the corresponding power module

Selecting signal evaluation (S7-300, S7-400)

Signal Evaluation A, B

Signal evaluation by means of A, B allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Rotary Encoder (Page 206)
  The 1Count5V can count the edges of the signals. Normally, only the edges at A are evaluated (single evaluation). To obtain a higher resolution, you can select whether the signals are to undergo single (Page 206), double (Page 207) or quadruple (Page 207) evaluation when assigning parameters ("Signal evaluation" parameter).
  Multiple evaluation is only possible with asymmetric incremental encoders with A and B signals that are 90 degrees out of phase.

Note

500 kHz for count frequency is the maximum frequency of the A and B signals. This means a maximum frequency of 1 MHz for the count pulses in double evaluation and a maximum of 2 MHz in quadruple evaluation.

Activating signal pair diagnostics (S7-300, S7-400)

Signal pair monitoring

Signal pairs A and /A, B and /B, and N and /N can be monitored.

Activate the corresponding check box.

Note

You cannot activate "N, /N diagnostics" until you have activated "A, /A and B, /B diagnostics".

Selecting the count direction (S7-300, S7-400)

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Behavior of the outputs in counting modes (S7-300, S7-400)

Comparison values and outputs (S7-300, S7-400)

Introduction

The 1Count5V allows you to store two comparison values, which are assigned to the digital outputs. The outputs can be activated in line with counter value and comparison values.

Digital outputs

1Count5V has two digital outputs.

Parameters can be assigned for both outputs ("Function DO1" and "Function DO2" parameters).

You can change the function and the behavior of the digital outputs during operation.

Controlling the outputs and comparators simultaneously

You can continue to control the outputs using SET_DO1 or SET_DO2 if you have selected a compare function for the outputs. This allows you to simulate the effect of the compare functions in your control program:

- The output is set with the positive edge of SET_DO1 or SET_DO2.
  However, if the "Pulse on Reaching the Comparison Value" function is selected, only one pulse with the specified duration is output. When pulse duration = 0, the output can be set with SET_DO1 or SET_DO2 as long as the counter value is at the comparison value or hysteresis is active.
  The SET_DO1 control bit cannot be used with output behavior "Switch at comparison values".
- A SET_DO1 or SET_DO2 negative edge resets the output.

Note that the comparators are still active and can set or reset the output if the comparison result changes.

Note

An output set by SET_DO1 or SET_DO2 cannot be reset (by the comparator) at the comparison value.

Loading Comparison Values

You transfer the comparison values to the 1Count5V. The counting is not affected by this.

Valid Range for the Two Comparison Values

Main count direction: None	Main count direction: Up	Main count direction: Down
Low count limit to high count limit	-2147483648 to high count limit -1	1 to 2147483647

Modifying the Function and Behavior of Digital Outputs

You can change the functions and behavior of the outputs during operation using the control interface (Page 397). The 1Count5V deletes the outputs and accepts the values as follows:

- **Function of digital outputs DO1 and DO2:** If you change this function so that the comparison condition is satisfied, the output will not be set until after the next count pulse. However, 1Count5V will not make any changes at the output if hysteresis (Page 386) is enabled.
- **Hysteresis:** Active hysteresis (Page 386) remains active after a change. The new hysteresis range will be applied the next time the comparison value is reached.
- **Pulse duration:** The new pulse duration (Page 389) will take effect with the next pulse.

Selecting output behavior (S7-300, S7-400)

Behavior of outputs DO1 and DO2

Select the behavior of the output in question in the drop down list:

- Output (Page 383)
- On when counter value ≥ comparison value (Page 383)
- On when counter value ≤ comparison value (Page 383)
- Pulse on reaching the comparison value (Page 383)
- Switch at comparison values (Page 384) (DO1 only)

Setting or Modifying the Function and Behavior of the Digital Output DO1

When setting or modifying the behavior of DO1, you must take all assignable interdependencies into account. Failure to do so will generate a parameter assignment error or a loading error.

Boundary conditions:

If you assign "Switch at comparison values" for DO1, you must:

- Set hysteresis = 0, and
- also configure "DO2 function" as "Output".

Output (S7-300, S7-400)

Output

You can switch the outputs on and off with the control bits (Page 397)SET_DO1 and SET_DO2.

The control bits (Page 397)CTRL_DO1 or CTRL_DO2 must be set.

You can query the status of the outputs with the status bits STS_DO1 and STS_DO2 in the feedback interface (Page 399).

The status bits (Page 399)STS_CMP1 and STS_CMP2 indicate that the relevant output is or was switched on. These status bits retain their status until they are acknowledged. If the output is still activated, the corresponding bit will be set again immediately. These status bits are also set when the control bit SET_DO1 or SET_DO2 is operated without DO1 or DO2 being enabled.

Isochrone mode

Outputs DO1 and DO2 are switched at time T_o in isochrone mode.

On when counter value ≤ comparison value or when counter value ≥ comparison value (S7-300, S7-400)

Counter value ≤ comparison value and Counter value ≥ comparison value

If the comparison conditions are fulfilled, the respective comparator will switch on the output. The status of the output (Page 399) is indicated by STS_DO1 and STS_DO2.

The control bits (Page 397)CTRL_DO1 or CTRL_DO2 must be set.

The result of the comparison is indicated by the status bits (Page 399)STS_CMP1 or STS_CMP2. You cannot acknowledge and thus reset these bits until the comparison conditions are no longer fulfilled.

Isochrone mode

In isochrone mode, too, the DO1 and DO2 outputs are switched as soon as the comparison condition is fulfilled and are therefore independent of the bus cycle.

Pulse on reaching the comparison value (S7-300, S7-400)

Pulse on reaching the comparison value

If the counter status reaches the comparison value, the comparator switches on the respective digital output for the assigned pulse duration.

The control bit (Page 397)CTRL_DO1 or CTRL_DO2 must be set.

The status of the status bits (Page 399)STS_DO1 and STS_DO2 is always the same as that of the corresponding digital output.

The comparison result is indicated by the status bit (Page 399)STS_CMP1 or STS_CMP2 and cannot be reset by acknowledgment until the pulse duration has elapsed.

If a main count direction is assigned, the comparator will only switch when the comparison value in the main count direction is reached.

If a main count direction is not assigned, the comparator switches when the comparison value is reached from either direction.

If the digital output is set by control bit (Page 397)SET_DO1 or SET_DO2, it will be reset when the pulse duration has elapsed.

Isochrone mode

In isochrone mode, too, the DO1 and DO2 outputs are switched as soon as the comparison condition is fulfilled and are therefore independent of the bus cycle.

Switch at comparison values (S7-300, S7-400)

Switch at comparison values

The comparator switches the output when the following conditions are met:

- The two comparison values must be loaded with the load functions (Page 397)CMP_VAL1 and CMP_VAL2 and
- The output DO1 (Page 397) must be enabled with CRTL_DO1 after the comparison values are loaded.

The following table shows you when the DO1 is switched on or off:

	DO1 is switched on when	DO1 Is switched off when
V2 < V1 (see Figure below)	V2 ≤ counter status ≤ V1	V2 > counter status or counter status > V1
V2 = V1	V2 = counter status = V1	V2 ≠ counter status ≠ V1
V2 > V1 (see Figure below)	V1 > counter status or counter status > V2	V1 ≤ counter status ≤ V2

The figures below show the status of output DO1 as determined by the counter value and comparison values:

- V2 < V1 at the start of counting

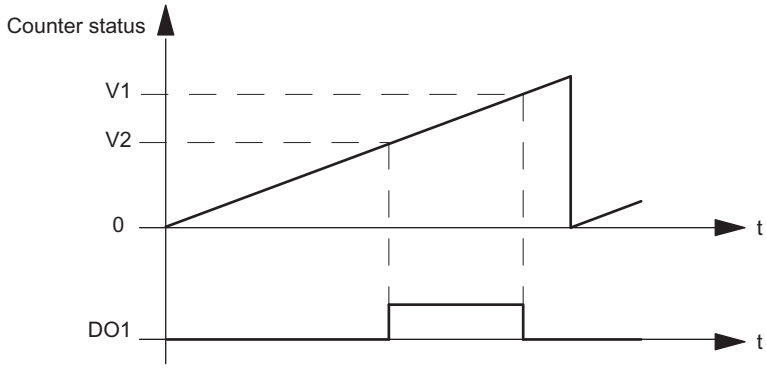

- V2 > V1 at the start of counting

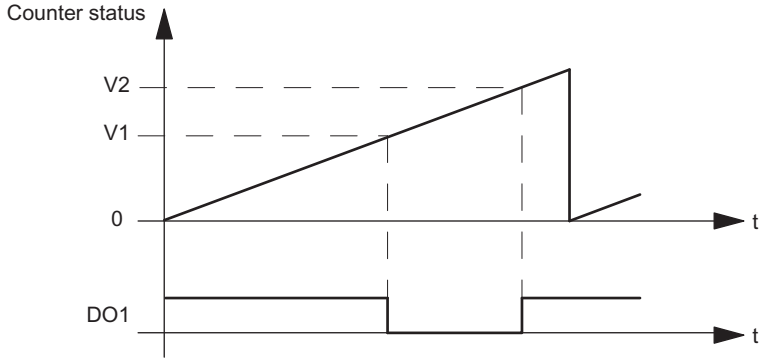

The result of the comparison is shown by the status bit (Page 399)STS_CMP1. You can only acknowledge and thus reset this bit when the comparison condition is no longer fulfilled.

There is no hysteresis in the case of this output behavior.

It is not possible to control the DO1 output with control bit (Page 397)SET_DO1 if this is the output behavior.

Isochronous mode

In isochrone mode, too, output DO1 is switched as soon as the comparison condition is fulfilled and is therefore independent of the bus cycle.

Selecting substitute value DOx (S7-300, S7-400)

Substitute values DO1 and DO2

This is where you select the substitute value which is to be output at digital output DO1 or DO2 if you have selected the parameter "DO substitute a value" under "Reaction to CPU/Master STOP (Page 378)".

Activating DOx diagnostics (S7-300, S7-400)

DO1 and DO2 diagnostics

"DO1 diagnostics" or "DO2 diagnostics" (wire break, short circuit) are possible only with pulse lengths of > 90 ms at digital output DO1 or DO2.

Select the check box for diagnostics at output DO1 or DO2.

Setting hysteresis (S7-300, S7-400)

Hysteresis

An encoder can remain at a particular position and then fluctuate around this position. This state causes the counter status to fluctuate around a particular value. If there is a comparison value in this fluctuation range, for example, the associated output is switched on and off in accordance with the rhythm of the fluctuations. The 1Count5V is equipped with programmable hysteresis to prevent this switching in response to small fluctuations. You can assign a range between 0 and 255 (0 means: hysteresis switched off).

Hysteresis also works with high limit violated and low limit violated.

Effect when counter value ≤ comparison value and counter value ≥ comparison value

The example in the figure below demonstrates the effect of hysteresis. The figure shows the various responses of an output with a hysteresis of 0 (= switched off) as opposed to a hysteresis of 3. In the example, the comparison value is 5.

The following settings are assigned for the counter: "Main count direction" = "Up" and "Turn on at counter status≥ comparison value".

When the comparison condition is met, hysteresis becomes active. While the hysteresis is active, the comparison result remains unchanged.

Hysteresis will no longer be active if the counter value is outside the hysteresis range. The comparator switches again in line with its comparison conditions.

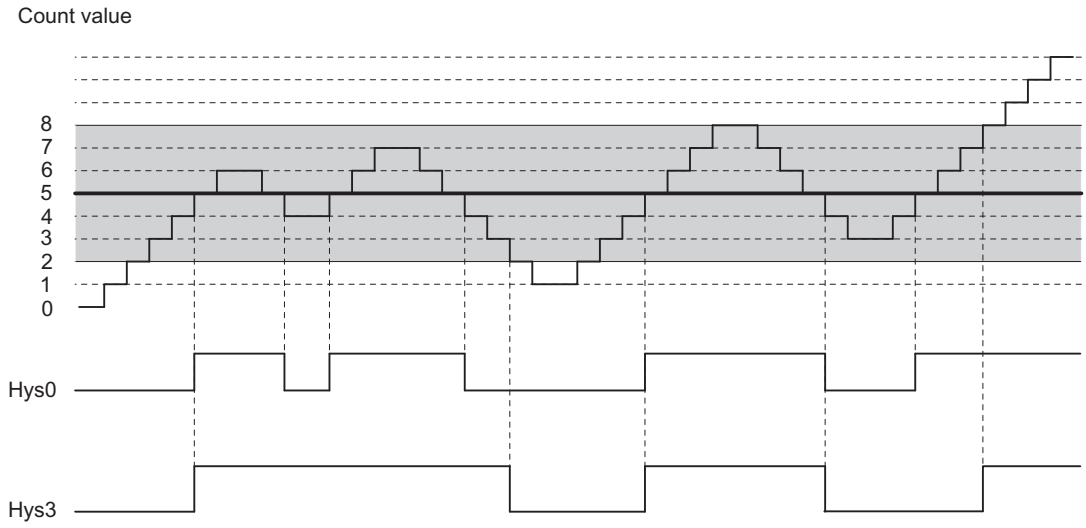

Note

If the count direction changes at the comparison value when hysteresis is active, the output will be reset.

Effect when comparison value is reached and pulse duration = 0

The example in the figure below demonstrates the effect of hysteresis. The figure shows the various responses of an output with a hysteresis of 0 (= switched off) as opposed to hysteresis of 3. In the example, the comparison value is 5.

The following settings are assigned for the counter: "Pulse on reaching the comparison value", "No main count direction" and "Pulse duration = 0".

When the comparison conditions are met, hysteresis becomes active. While the hysteresis is active, the comparison result remains unchanged.

Hysteresis will no longer be active if the counter value is outside the hysteresis range. The comparator deletes the result of the comparison.

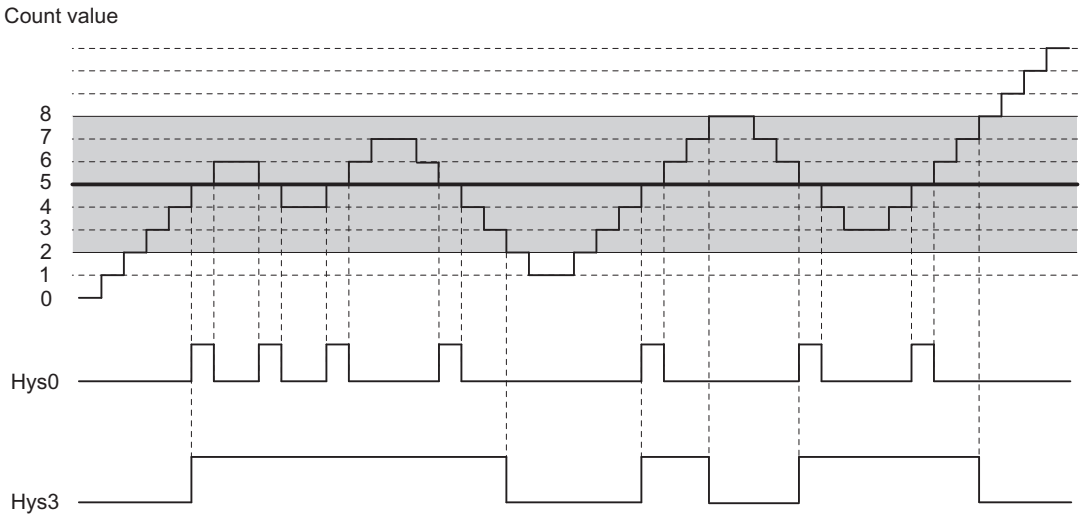

Effect when comparison value is reached, output pulse duration

The example in the figure below demonstrates the effect of hysteresis. The figure shows the various responses of an output with a hysteresis of 0 (= switched off) as opposed to hysteresis of 3. In the example, the comparison value is 5.

The following settings are assigned for the counter: "Pulse on reaching the comparison value", "No main count direction" and "Pulse duration > 0".

When the comparison conditions have been met, hysteresis becomes active and a pulse of the assigned duration is output.

Hysteresis will no longer be active if the counter value is outside the hysteresis range.

When hysteresis is enabled, 1Count5V stores the count direction.

A pulse is output if the hysteresis range is exited in a different direction to the one stored.

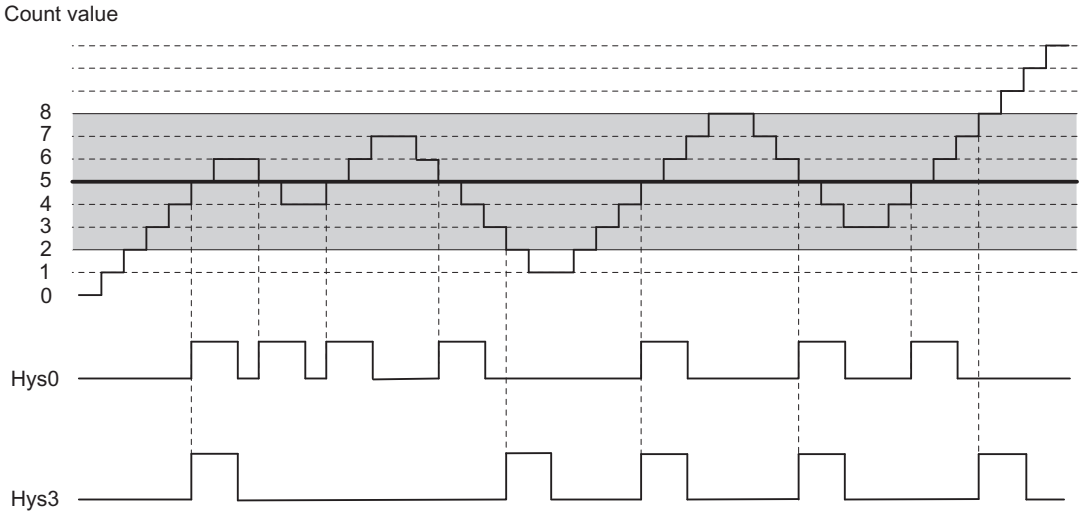

Specifying pulse duration (S7-300, S7-400)

Pulse duration

The pulse duration begins when the respective digital output is set. The pulse duration is accurate to at least 2 ms.

The pulse duration can be set to suit the actuators used. The pulse duration specifies how long the output is to be set for. The pulse duration can be preselected in increments of 2 ms between 0 ms and 510 ms.

If the pulse duration = 0, the output will be set until the comparison condition is no longer fulfilled. Note that the count pulse times must be greater than the minimum switching times of the digital output.

Isochrone mode

In isochrone mode, too, the DO1 and DO2 outputs are switched as soon as the comparison condition is fulfilled and are therefore independent of the bus cycle.

Selecting counting mode (S7-300, S7-400)

Description

You define 1Count5V functionality by setting a counting mode.

Operating mode	Description
Count continuously (Page 182) (with or without gate)	1Count5V counts continuously from the current counter value.
Count once (Page 183) (with hardware or software gate)	When the gate opens, 1Count5V counts from the load value to the count limit.
Count periodically (Page 184) (with hardware or software gate)	When the gate opens, 1Count5V counts from the load value between the count limits.

The default setting is the "Count continuously" operating mode.

Selecting main count direction (S7-300, S7-400)

Description

By configuring a main count direction (Page 185) (up or down), you can reduce the maximum counting range by setting an high count limit (Page 397). The parameterized counting range is then between 0 and the high count limit parameterized.

A main count direction can only be selected in the "Count once" and " Count periodically" modes.

You can select the following behavior:

- None (no main count direction)
- Up
- Down

Selecting the gate function (S7-300, S7-400)

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 202): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 202): The counting process starts from the last current counter value after gate stop and restart.

DI function (S7-300, S7-400)

Selecting DI function (S7-300, S7-400)

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 201), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Latch/Retrigger with positive edge (Page 391)
  Counter values are saved (latched) with positive edges at the digital input. The counter is set to the load value each time the values are saved.
- Synchronization with positive edge (Page 392)
  The counter is set to the load value with positive edges at the digital input.
- Latching with positive edge (Page 394)
  Counter values are saved (latched) with positive edges at the digital input.
- Hardware enable for synchronization (Page 395)
  The digital input acts as a hardware enable. The encoder zero mark sets the counter to the load value when the hardware enable is active.

The default setting is "Input".

Latch/Retrigger with positive edge (S7-300, S7-400)

Latch/Retrigger

This function saves the current internal counter value of the 1Count5V and retriggers counting when there is a positive edge at the digital input. This means that the current internal counter value at the time of the positive edge is stored (latch value), and the 1Count5V is then loaded again with the load value, from which counting resumes.

The following figure shows latch/retrigger when load value = 0.

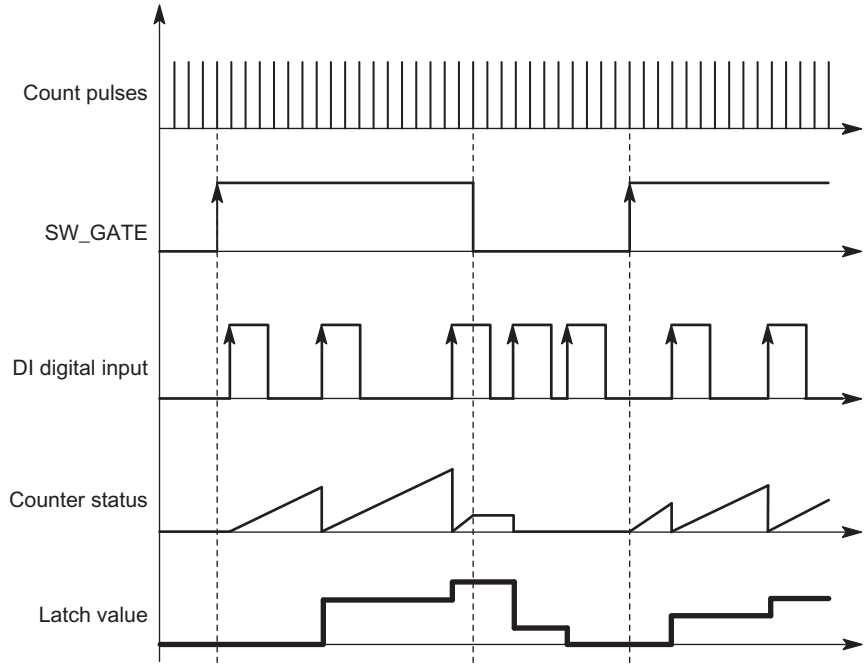

The counting mode must be enabled with the software gate before the function can be executed. It is started with the first positive edge at the digital input.

The saved rather than the current counter value is displayed in the feedback interface (Page 399). The STS_DI bit indicates the status of the latch and retrigger signal.

The latch value is pre-assigned its RESET status:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

It is not changed when the software gate is opened.

Directly loading the counter does not alter the saved counter value displayed.

Closing the software gate only interrupts counting; this means that counting resumes when you open the software gate again. The digital input DI remains active even when the software gate is closed.

Counting is also latched and triggered in isochrone mode with each edge at the digital input. The counter status that was valid at the time of the last edge before T_i is displayed in the feedback interface.

Extended feedback interface

If the 1Count5V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 399).

Synchronization with positive edge (S7-300, S7-400)

Synchronization

If you have configured "Synchronization with positive edge", the rising edge of a reference signal at the input will set 1Count5V to the load value.

You can choose between once only and periodic synchronization ("Synchronization" parameter).

The figure below shows the counter values in once only and periodic synchronization.

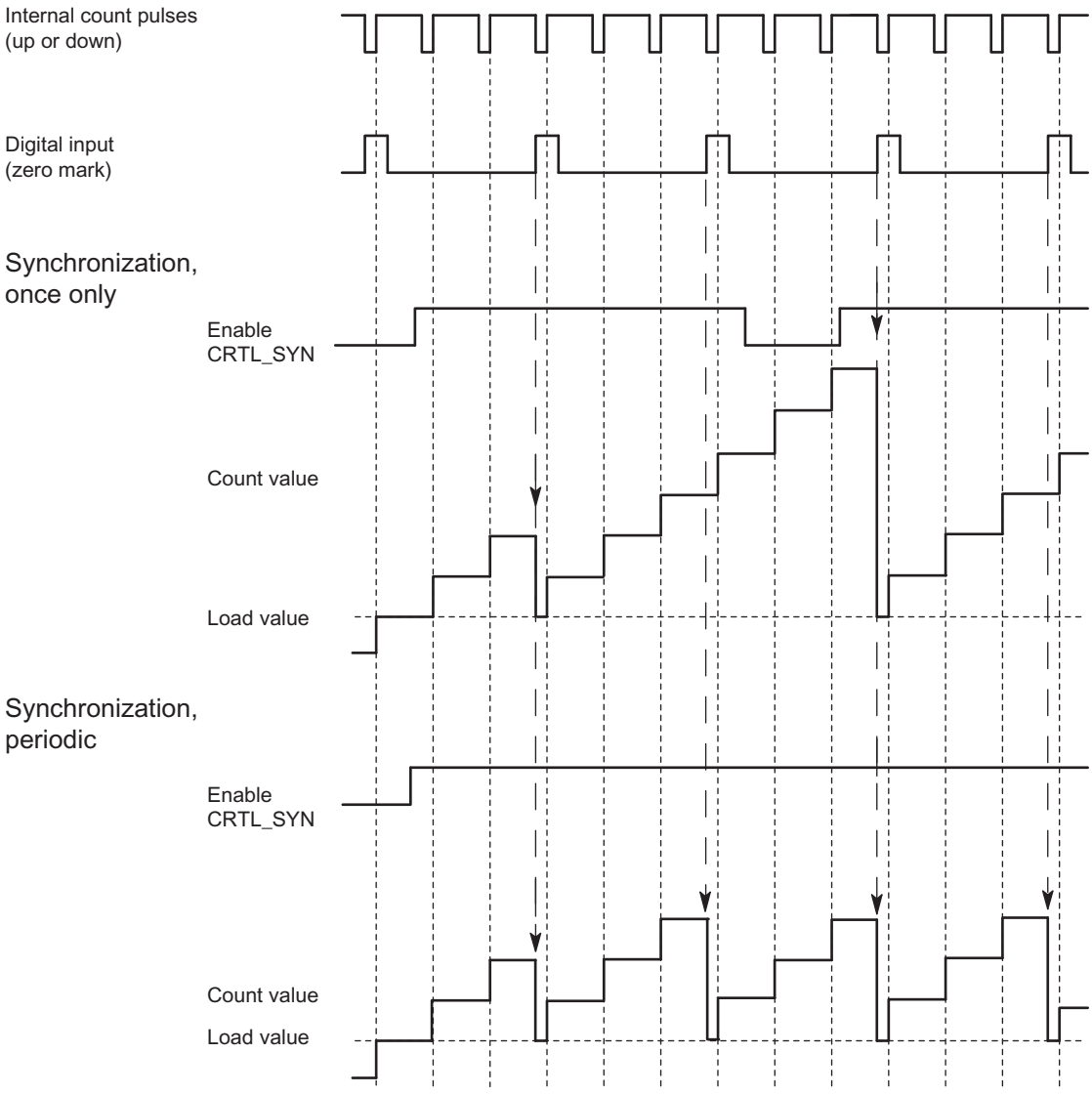

The following conditions apply:

- The counting mode must have been started with the software gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 397) must be set.
- Synchronize once only: If the enable bit is set, the first edge loads 1Count5V with the load value.
- Synchronization periodically: If the enable bit is set, the first and each subsequent edge loads the 1Count5V with the load value.
- After successful synchronization, the STS_SYNfeedback bit (Page 399) is set and the SYN LED comes on. The RES_STScontrol bit (Page 397) resets the feedback bit and switches off the LED.
- The signal of a bounce-free switch can serve as the reference signal.
- The feedback bit (Page 399)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 399)STS_SYN indicates that the rising edge at the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Latching with positive edge (S7-300, S7-400)

Latching

Counter value and latch value are pre-assigned their RESET statuses:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

The counting function is started when the software gate is opened. 1Count5V begins at the load value:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

The following figure shows latching when load value = 0.

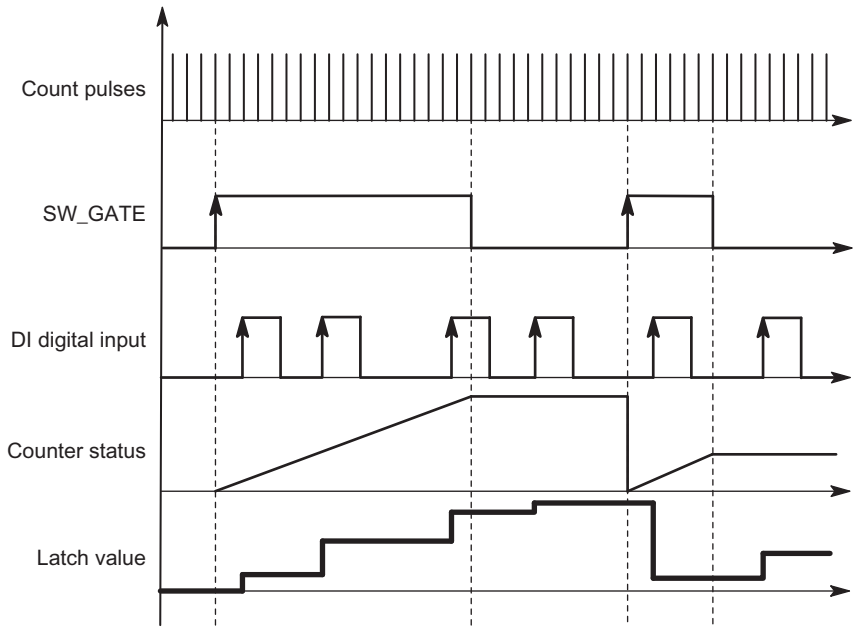

The latch value is always the exact counter value at the time of the positive edge at the digital input DI.

The saved rather than the current counter value is displayed in the feedback interface (Page 399). The STS_DI bit indicates the level of the latch signal.

Directly loading the counter does not alter the saved counter value displayed.

In isochrone mode, the counter status that was latched at the time of the last positive edge before T_i is displayed in the feedback interface.

Closing the software gate will either cancel or interrupt depending on the parameter assignment. The digital input DI remains active even when the software gate is closed.

Extended feedback interface

If the 1Count5V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 399).

Hardware enable for synchronization (S7-300, S7-400)

Synchronization with DI and Zero Mark

The DI will act as the hardware enable if you have configured "Hardware enable for synchronization". When the HW enable is active, the 1Count5V is loaded with the load value by the zero mark of the encoder.

You can choose between once only and periodic synchronization ("Synchronization (Page 397)" parameter).

The figure below shows the counter values in once only and periodic synchronization.

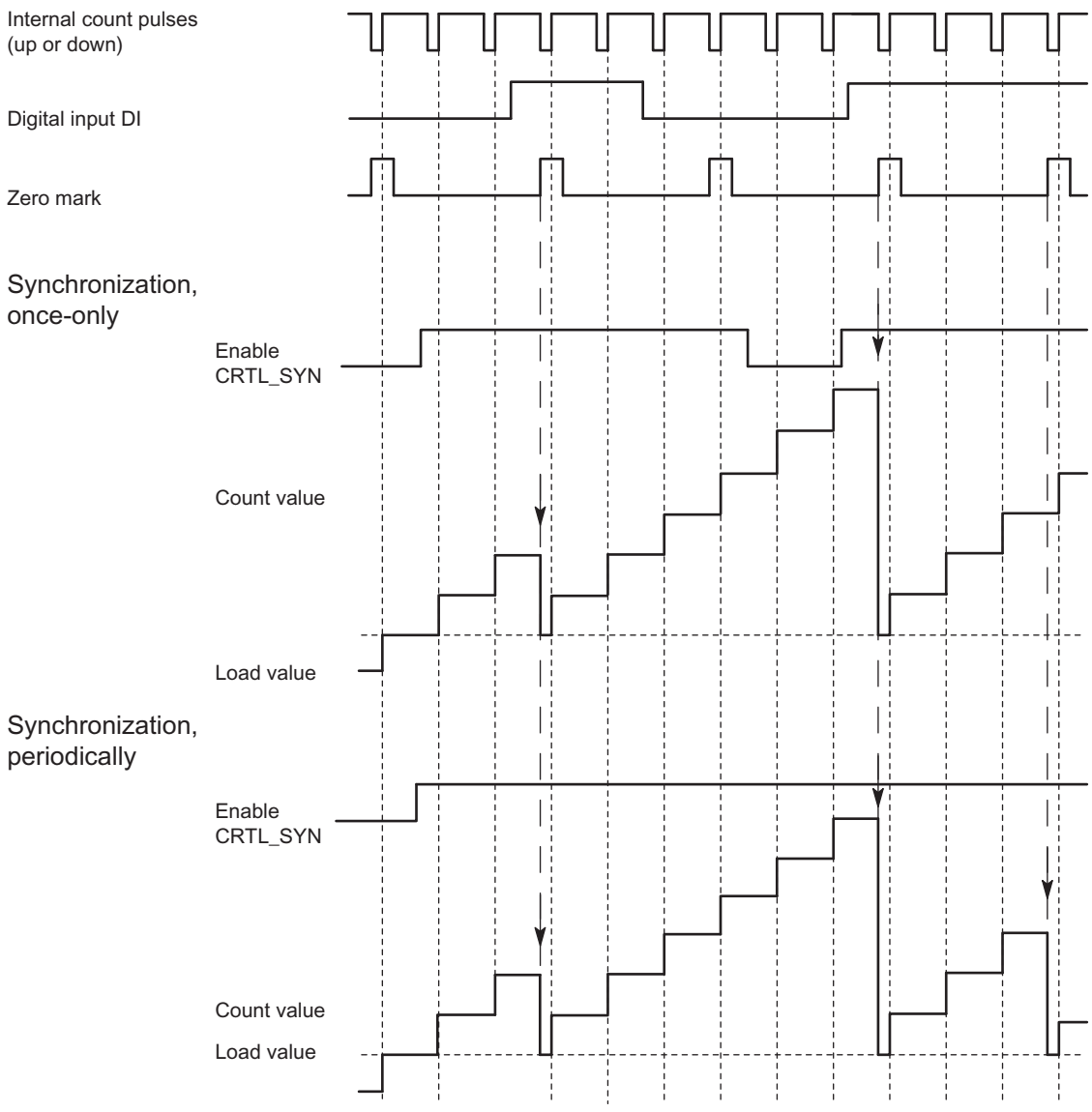

The following conditions apply:

- The counting mode must have been started with the software gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 397) must be set.
- Synchronize once only: If the enable bit is set and the HW enable is present, the first zero mark loads the 1Count5V with the load value.
- Synchronization periodically: If the enable bit is set and the HW enable is present, the first and each subsequent zero mark will load 1Count5V with the load value.
- After successful synchronization, the STS_SYNfeedback bit (Page 399) is set and the SYN LED comes on. The RES_STScontrol bit (Page 397) resets the feedback bit and switches off the LED.
- The signal of a bounce-free switch can serve as the reference signal.
- The feedback bit (Page 399)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 399)STS_SYN indicates that the positive edge on the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Selecting hardware gate input signal (S7-300, S7-400)

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 399)STS_DI indicates the level of the digital input.

Selecting synchronization (S7-300, S7-400)

Synchronization

This is where you can select either once only or periodic synchronization if you have selected "Synchronization with positive edge" at the "DI function (Page 390)" parameter:

- Once: If the enable bit is set, the first edge loads the 1Count5V with the load value.
- Periodic: If the enable bit is set, the first and each subsequent edge loads 1Count5V with the load value.

Specifying the high count limit (S7-300, S7-400)

Description

By configuring an high count limit together with a main count direction (Page 185) (up or down), you can reduce the maximum counting range. The parameterized counting range is then between 0 and the high count limit parameterized.

The high count limit can be any value between > 1 and the high count range limit.

Control and feedback interface for counting modes (S7-300, S7-400)

Control interface for counting modes (S7-300, S7-400)

Note

The following control interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count5V control interface (outputs) for counting modes.

Address		Assignment			
Bytes 0 to 3		**Load value direct, in preparation, comparison value 1 or 2**			
	Byte 0	**Behavior of DO1, DO2 of 1Count5V**			
		Bit 2	Bit 1	Bit 0	Function DO1
		0	0	0	Output
		0	0	1	On when counter value ≥ comparison value
		0	1	0	On when counter value ≤ comparison value
		0	1	1	Pulse on reaching the comparison value
		1	0	0	Switch at comparison values
		1	0	1	disabled
		1	1	0	disabled
		1	1	1	disabled
			Bit 5	Bit 4	Function DO2
			0	0	Output
			0	1	On when counter value ≥ comparison value
			1	0	On when counter value ≤ comparison value
			1	1	Pulse on reaching the comparison value
		Bits 3, 6, and 7: Reserve = 0			
	Bytes 1 to 3	Byte 1	Hysteresis DO1, DO2 (value range 0 to 255)		
		Byte 2	Pulse duration DO1, DO2 (value range 0 to 255)		
		Byte 3	Reserved = 0		
					Name
Byte 4		Bit 7	Error acknowledgment		EXTF_ACK
		Bit 6	Enable DO2		CTRL_DO2
		Bit 5	Control bit DO2		SET_DO2
		Bit 4	Enable DO1		CTRL_DO1
		Bit 3	Control bit DO1		SET_DO1
		Bit 2	Trigger status bits reset		RES_STS
		Bit 1	Enable synchronization		CTRL_SYN
		Bit 0	Software gate control bit		SW_GATE
Byte 5		Bit 7	Reserved = 0		
		Bit 6	Reserved = 0		
		Bit 5	Reserved = 0		
		Bit 4	Change function and behavior of DO1, DO2		C_DOPARAM
		Bit 3	Load comparison value 2		CMP_VAL2
		Bit 2	Load comparison value 1		CMP_VAL1
		Bit 1	Load counter in preparation		LOAD_PREPARE
		Bit 0	Load counter directly		LOAD_VAL
Bytes 6 to 7		Reserved = 0 [1]			

[1] Not used for extended user interface

Notes on the control bits

Bit	Explanation
C_DOPARAM	Change function and behavior of DO1, DO2 The values from bytes 0 to 2 are applied as new function, hysteresis, and pulse duration of DO1, DO2. This may result in the following error: The conditions for the "Switch at comparison values" behavior are not fulfilled.
CMP_VAL1	Load comparison value 1 The value from bytes 0 to 3 is transferred to comparison value 1 with the control bit "Load comparison value 1" (CMP_VAL1).
CMP_VAL2	Load comparison value 2 The value from bytes 0 to 3 is transferred to comparison value 2 with the control bit "Load comparison value 2" (CMP_VAL2).
CTRL_DO1	Enable DO1 You use this bit to enable the DO1 output.
CTRL_DO2	Enable DO2 You use this bit to enable the DO2 output.
CTRL_SYN	You use this to enable synchronization.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load counter in preparation The value from bytes 0 to 3 is applied as the load value.
LOAD_VAL	The value from bytes 0 to 3 is loaded directly as the new counter value.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SET_DO1	Control bit DO1 Switches the DO1 digital output on and off when CRTL_DO1 is set.
SET_DO2	Control bit DO2 Switches the DO2 digital output on and off when CRTL_DO2 is set.
SW_GATE	Software gate control bit The software gate is opened/closed via the control interface with the SW_GATE bit.

Feedback interface for counting modes (S7-300, S7-400)

Note

The following feedback interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count5V feedback interface (inputs) for counting modes.

Address	Assignment		Name
Bytes 0 to 3	Counter value or stored counter value in the case of the latch function on the digital input		
Byte 4	Bit 7	Short circuit of the encoder supply	ERR_24V
	Bit 6	Short circuit / wire break / overtemperature	ERR_DO1
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Short circuit / wire break / overtemperature	ERR_DO2
	Bit 3	Short circuit / wire break / encoder signal	ERR_ENCODER
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Down direction status	STS_C_DN
	Bit 6	Up direction status	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	DO2 status	STS_DO2
	Bit 3	DO1 status	STS_DO1
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 7	Zero-crossing in the count range when counting without a main count direction	STS_ND
	Bit 6	Low count limit	STS_UFLW
	Bit 5	High count limit	STS_OFLW
	Bit 4	Comparator 2 status	STS_CMP2
	Bit 3	Comparator 1 status	STS_CMP1
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Synchronization status	STS_SYN
Byte 7	Reserved = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the Feedback Bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged by the EXTF_ACKcontrol bit (Page 397). Diagnostic alarm, if parameterized.
ERR_DO1	Short circuit/wire break/overtemperature due to overload at output DO1 The error bit must be acknowledged by the EXTF_ACKcontrol bit (Page 397). Diagnostic alarm, if parameterized.

Bit	Explanation
ERR_DO2	Short circuit/wire break/overtemperature at output DO2 The error bit must be acknowledged by the EXTF_ACKcontrol bit (Page 397). Diagnostic alarm, if parameterized.
ERR_ENCODER	Short circuit / wire break in 5 V encoder signal The error bit must be acknowledged by the EXTF_ACKcontrol bit (Page 397). Diagnostic alarm, if parameterized.
ERR_LOAD	Load function error The LOAD_VAL, LOAD_PREPARE, CMP_VAL1, CMP_VAL2 and C_DOPARAMcontrol bits (Page 397) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_CMP1	Comparator 1 status The STS_CMP1 status bit indicates that the output is or was switched on. It must be acknowledged with the control bit (Page 397)RES_STS. If the status bit is acknowledged when the output is still switched on, the bit is set again immediately. This bit is also set if the SET_DO1 control bit is used when DO1 is not enabled.
STS_CMP2	Comparator 2 status The STS_CMP2 status bit indicates that the output is or was switched on. It must be acknowledged with the control bit (Page 397)RES_STS. If the status bit is acknowledged when the output is still switched on, the bit is set again immediately. This bit is also set if the SET_DO2 control bit is operated when DO2 is not enabled.
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_DO1	DO1 status The STS_DO1 status bit indicates the status of the DO1 digital output.
STS_DO2	DO2 status The STS_DO2 status bit indicates the status of the DO2 digital output.
STS_GATE	Internal gate status: Counting
STS_LOAD	Load function is running
STS_ND	Zero-crossing in the count range when counting without a main count direction. The bit must be reset by the control bit (Page 397)RES_STS.
STS_OFLW STS_UFLW	High count limit violated Low count limit violated Both bits must be reset.
STS_SYN	Synchronization status The STS_SYN bit is set after successful synchronization. It must be reset by the control bit (Page 397)RES_STS.

Measurement parameters (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP (S7-300, S7-400)

Setting response to CPU/Master STOP

You can configure the response of the 1Count5V to failure of the higher-level controller.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.
DO substitute a value	The current mode is cancelled. The gate is closed and the substitute value set for the digital output is activated. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters. If "Pulse when comparison value is reached" has been set for the output, the only available pulse duration substitute value will be 1.	The changed parameters are accepted and become effective.
DO keep last value	The current mode is cancelled. The gate is closed and the status of the digital output is maintained. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count5V.

Exiting parameterized status

Under what conditions does the 1Count5V exit the parameterized status?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 413).

Automatic new parameter assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon CPU/DP master power on
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count5V is inserted
- Upon power on or insertion of the corresponding power module

Measuring methods (S7-300, S7-400)

Overview (S7-300, S7-400)

Measuring methods

You can choose from the following measuring methods for the "Measuring procedure" parameter:

- With integration time (Page 403)
- Continuous (Page 404)

Measurement with integration time (S7-300, S7-400)

Measurement with integration time

The measurement is carried out during the assigned integration time. When the integration time elapses, the measured value is updated.

The end of a measurement is indicated by the STS_CMP1status bit (Page 415). This bit is reset by the control bit (Page 413)RES_STS in the control interface.

If there were not at least two rising edges in the assigned integration time, 0 is returned as the measured value.

A value of -1 is returned up until the end of the first integration time.

You can change the integration time for the next measurement during operation.

Direction Reversal

If the direction of rotation is reversed during integration time, the measured value for this measurement period will be undefined. Evaluating the STS_C_UP and STS_C_DNfeedback bits (Page 415) (direction evaluation) allows you to respond to any process irregularities.

See also

Integration Time and Update Time in Isochronous Mode (Page 407)

Continuous measurement (S7-300, S7-400)

Measuring Principle

The 1Count5V counts each positive edge of a pulse and assigns it a time value in µs.

The update time indicates the time interval at which the measured value is updated by the module in the feedback interface.

Rule for pulse sequences with one or several pulses per update time:

Dynamic measuring time =Time of last pulse in the current update time interval
minus
Time of last pulse in the previous update time interval

When the update time has elapsed, a new measured value is calculated and output with the dynamic measuring time.

The following dynamic measuring time will result if the current update time does not contain a pulse:

Dynamic measuring time =Time of current, elapsed update time
minus
Time of last pulse

When the update time has elapsed, an estimated measured value is calculated with the dynamic measuring time under the assumption that a pulse occurred at the end of the update time.

If the "1 Pulse per dynamic measuring time" estimated measured value is less than the last measured value during the frequency and speed measurement, this estimated measured value is output as the new measured value. With the period measurement, the dynamic measuring time is output as the estimated period if the dynamic measuring time is greater than the last measured period.

The figure below shows the measuring principle.

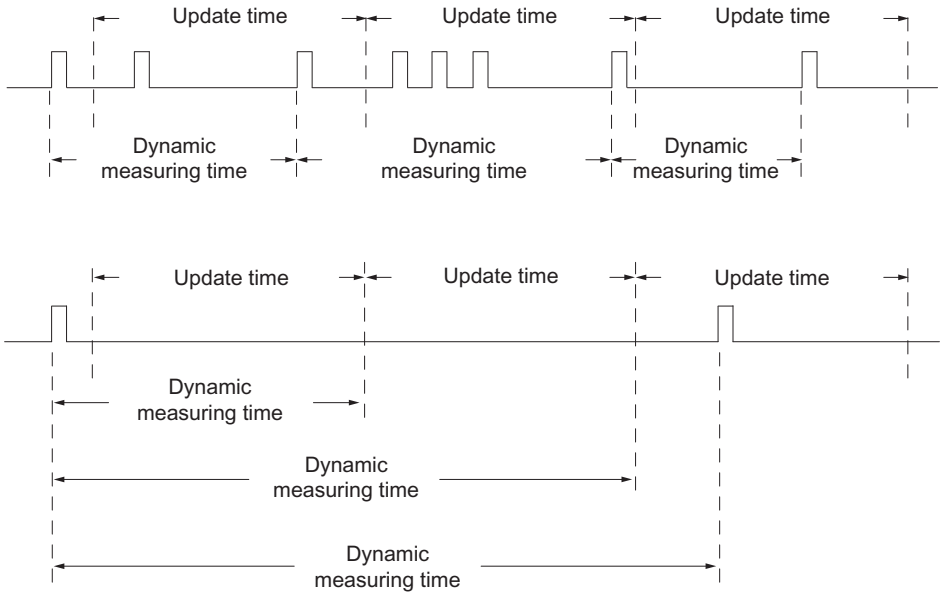

1Count5V measures continuously. When assigning parameters, you specify an update time.

During the time until the end of the first elapsed update time, a value of "-1" is returned.

The continuous measurement begins after the gate is opened with the first pulse of the pulse train to be measured. The first measured value can be calculated after the second pulse, at the earliest.

A measured value (frequency, period, or speed) is output in the feedback interface each time the update time elapses. The end of a measurement is indicated by the STS_CMP1status bit (Page 415). This bit is reset with the RES_STS and RES_STS_A bits by applying the complete acknowledgement principle.

If the direction of rotation is reversed during an update time, the measured value for this measurement period will be undefined. Evaluating the STS_C_DN and STS_C_UPfeedback bits (Page 415) (direction evaluation) enables you to respond to any process irregularities.

The following figure illustrates the principle of continuous measurement taking frequency measurement as an example.

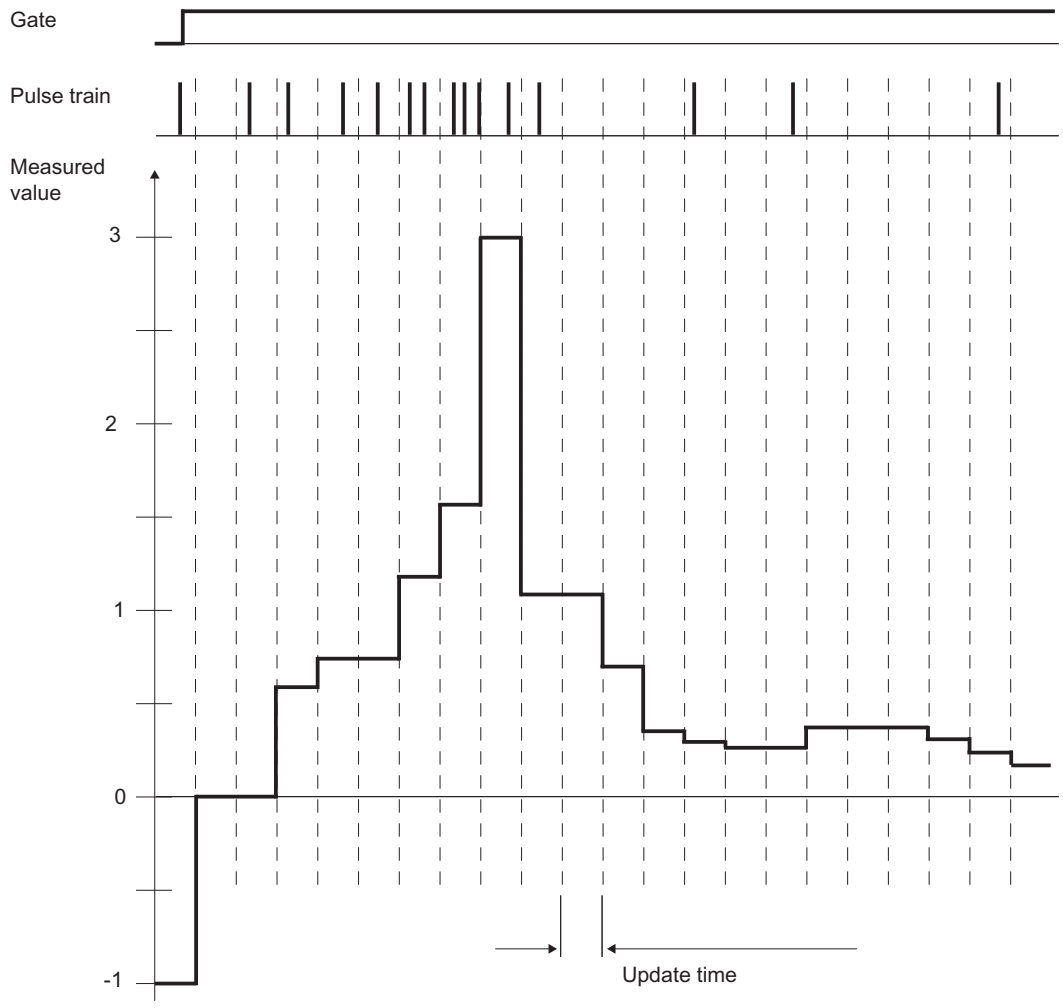

Gate Control

To control the 1Count5V, you have to use the gate functions.

Isochronous Mode

In isochrone mode, the 1Count5V accepts control bits and control values from the control interface in each bus cycle and reports back the response in the same cycle.

The 1Count5V transfers a measured value and the status bits that were valid at time T_i in each cycle.

The measurement starts and ends at time T_i.

See also

Integration Time and Update Time in Isochronous Mode (Page 407)

Integration Time and Update Time in Isochronous Mode (S7-300, S7-400)

Integration time and update time in isochrone mode

If the integration time/update time lasts several T_{DP} cycles, the new measured value will be indicated in the user program by the STS_CMP1status bit (Page 415) (measurement completed) of the feedback interface. This enables monitoring of the measuring operation or a synchronization with the measuring operation. It takes 4 T_{DP} cycles, however, for this alarm to be acknowledged. The minimum integration time/update time in this case is (4 x T_{DP}).

You do not need to continually evaluate status bit STS_CMP1 if the application can tolerate a jitter in the integration time/update time of T_{DP} and a measured value that remains constant for several cycles. Integration times/update times of (1 x T_{DP}) to (3 x T_{DP}) are then possible.

A loss of isochrone operation in the last T_{DP} cycle of the integration time will increase the integration time/update time by one T_{DP} cycle. This does not corrupt the measured value.

Note

The value range limits for the integration time/update time may not be exceeded (see tables for the individual measuring modes).

Violation of the value range limits will result in a parameter assignment error and 1Count5V will not go into isochrone mode.

Note

You must always adjust the integration time/update time parameter if you want to retain the integration time/update time length when you change the configuration from non-isochrone to isochrone mode and vice versa.

Activating signal pair diagnostics (S7-300, S7-400)

Signal pair monitoring

Signal pairs A and /A, and B and /B can be monitored.

Activate the check box.

Selecting the count direction (S7-300, S7-400)

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Behavior of the outputs in measuring modes (S7-300, S7-400)

Selecting behavior of output DO1 (S7-300, S7-400)

Behavior of the Output in Measuring Modes

You can assign parameters for the digital output of the 1Count5V.

You can store a high and a low limit for frequency measurement, rotational speed measurement or period measurement. If the limits are violated, digital output DO1 is activated. These limit values can be assigned and changed with the load function.

You can change the function and the behavior of the digital output during operation. The new function takes effect immediately. In isochronous mode it always takes effect at time T_i.

You can choose between the following functions in the drop-down list box.

- Output (Page 408)
- Outside limits (Page 409) (limit monitoring)
- Violating low limit (Page 409) (limit monitoring)
- Exceeding high limit (Page 409) (limit monitoring)

Output (S7-300, S7-400)

Output

If you want to switch the output on or off, you must enable it with the CTRL_DO1 or CTRL_DO2control bit (Page 413).

You can switch the outputs on and off with the control bits (Page 413)SET_DO1 and SET_DO2.

You can query the status of the output with the STS_DO1 or STS_DO2status bit (Page 415) in the feedback interface.

The outputs are switched at time T_o in isochronous mode.

Limit-Value Monitoring (S7-300, S7-400)

Limit monitoring

The figure below illustrates limit monitoring.

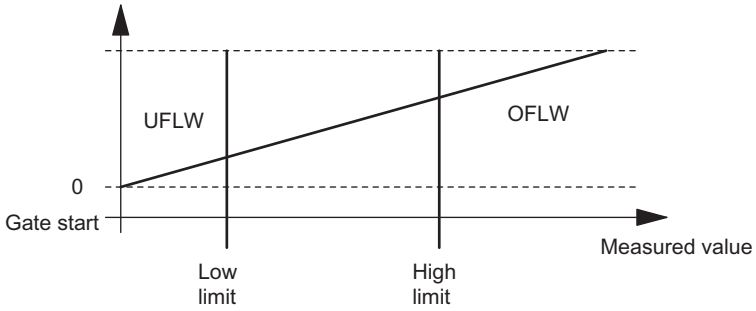

When the integration time has elapsed, the measured value obtained (frequency, rotational speed or period) is compared with the assigned limits.

The STS_UFLW = 1 bit will be set in the feedback (Page 415) interface if the current measured value is below the parameterized low limit (measured value < low limit).

The STS_OFLW = 1 bit will be set in the feedback (Page 415) interface if the current measured value is above the parameterized high limit (measured value > high limit).

You must acknowledge these bits with the control bit (Page 413)RES_STS.

The corresponding status bit will be set again if the measured value still or once again violates the limits after acknowledgment.

Setting the low limit to 0 switches off dynamic monitoring of "Violating low limit".

The table below shows how the enabled digital output DO1 can be set by limit monitoring in accordance with parameter assignment:

"Function DO1" parameter	DO1 is Set ...
Outside limits	Measured value < low limit OR measured value > high limit
Violating low limit	Measured value < low limit
Exceeding high limit	Measured value > high limit

In isochronous mode the output is switched at the end of measurement at time T_i.

Selecting substitute value DOx (S7-300, S7-400)

Substitute values DO1 and DO2

This is where you select the substitute value which is to be output at digital output DO1 or DO2 if you have selected the parameter "DO substitute a value" under "Reaction to CPU/Master STOP (Page 402)".

Activating DOx diagnostics (S7-300, S7-400)

DO1 and DO2 diagnostics

"DO1 diagnostics" or "DO2 diagnostics" (wire break, short circuit) are possible only with pulse lengths of > 90 ms at digital output DO1 or DO2.

Select the check box for diagnostics at output DO1 or DO2.

Selecting a measuring mode (S7-300, S7-400)

Description

You define 1Count5V functionality by specifying a measuring mode.

	Description	
Operating mode	Measurement with integration time	Continuous measurement
Frequency measurement (Page 191) (with hardware or software gate)	1Count5V counts the pulses that occur during a specified integration time.	1Count5V counts the pulses that occur during a dynamic measurement time.
Rotational speed measurement (Page 193) (with hardware or software gate)	1Count5V counts the pulses received from a tachometer generator within a specified integration time and calculates the speed on the basis of this value and the number of pulses per encoder revolution.	1Count5V counts the pulses received from a tachometer generator within a dynamic measuring time and calculates the speed on the basis of this value and the number of pulses per encoder revolution.
Period measurement (Page 194) (with hardware or software gate)	1Count5V measures the time between two positive edges of the count signal by counting the pulses of an internal quartz accurate reference frequency (16 MHz) over a specified integration time.	1Count5V indicates the dynamic measuring time as a period. The mean value of the period is calculated if its duration is less than the update time.

The default setting is "Frequency measurement" mode.

Selecting period resolution (S7-300, S7-400)

Description

The value of the calculated period is displayed in 1 µs or 1/16 µs.

Select the required unit of resolution.

Possible Measuring Ranges with Error Indication

- Period measurement with integration time
  - Measuring ranges and errors at a resolution of 1 µs:

Integration time	T_{min} ± absolute error	T_{min} ± absolute error
100 s	1 µs* (10 ± 0)	1 µs* (100,000,000 ± 10,000)
10 s	1 µs* (10 ± 0)	1 µs* (10,000,000 ± 1,000)
1 s	1 µs* (10 ± 0)	1 µs* (1,000,000 ± 100)
0.1 s	1 µs* (10 ± 0)	1 µs* (100,000 ± 10)
0.01 s	1 µs* (10 ± 0)	1 µs* (10,000 ± 1)

  - Measuring ranges and errors at a resolution of 1/16 µs:

Integration time	T_{min} ± absolute error	T_{min} ± absolute error
100 s	1/16 µs* (160 ± 0)	1/16 µs* (1,600,000,000 ± 160,000)
10 s	1/16 µs* (160 ± 0)	1/16 µs* (160,000,000 ± 16,000)
1 s	1/16 µs* (160 ± 0)	1/16 µs* (16,000,000 ± 1,600)
0.1 s	1/16 µs* (160 ± 0)	1/16 µs* (1,600,000 ± 160)
0.01 s	1/16 µs* (160 ± 0)	1/16 µs* (160,000 ± 16)

- Continuous period measurement:
  - Measuring ranges and errors at a resolution of 1 µs:

Period T_{min} ± Absolute error	Period T_{min} ± Absolute error
1 µs* (10 ± 0)	1 µs* (100,000 ± 10)
1 µs* (100 ± 0)	1 µs* (1 000 000 ± 100)
1 µs* (1 000 ± 0)	1 µs* (10 000 000 ± 1 002)
1 µs* (10 000 ± 1)	1 µs* (100,000,000 ± 10,020)

  - Measuring ranges and errors at a resolution of 1/16 µs:

Period T_{min} ± Absolute error	Period T_{min} ± Absolute error
1/16 µs* (160 ± 1)	1/16 µs* (1,600,000 ± 160)
1/16 µs* (1,600 ± 1)	1/16 µs* (16,000,000 ± 1,600)
1/16 µs* (16 000 ± 3)	1/16 µs* (160 000 000 ± 16 000)
1/16 µs* (160 000 ± 20)	1/16 µs* (1,600,000,000 ± 160,000)

Selecting DI function (S7-300, S7-400)

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 201), level-controlled
  The hardware gate is opened and closed by signals at the digital input.

The default setting is "Input".

Selecting hardware gate input signal (S7-300, S7-400)

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 415)STS_DI indicates the level of the digital input.

Setting low and high limit (S7-300, S7-400)

Low and high limit

You can save a high and low limit for frequency measurement, speed measurement and period duration measurement. If these limits are exceeded, digital output DO1 (Page 408) will be activated. You can configure these limits and change them using the load function.

The following value ranges are possible for the low and high limits:

Measuring mode	Low limit (LL)	High limit
Frequency measurement	0 to 99,999,999 × 10^{-3} Hz	LL+1 to 100,000,000 × 10^{-3} Hz
Rotational speed measurement	0 to 24,999,999 × 10^{-3} /min	LL+1 to 25,000,000 × 10^{-3} /min
Period duration measurement (resolution = 1 µs)	0 to 119,999,999 µs	LL+1 to 120,000,000 µs
Period duration measurement (resolution = 1/16 µs)	0 to 1 919 999 999 × 1/16 µs	LL+1 to 1 920 000 000 × 1/16 µs

Specifying measuring time (S7-300, S7-400)

Description

If you have selected a measuring mode, the module will update the measured values on a cyclic basis. The update time/integration time is the product of measuring time x 10ms. You set the measuring time as the factor n.

Detailed descriptions of the value ranges for the measuring time can be found under "Measuring basics (Page 187)".

Entering pulses per encoder revolution (S7-300, S7-400)

Pulses per encoder revolution

For "Rotational speed measurement (Page 193)" mode, you must also configure the pulses per encoder revolution.

The value for the number of pulses per encoder revolution can be found on the rating plate, or in the technical data of your encoder.

Control and feedback interface for measuring modes (S7-300, S7-400)

Control interface for measuring modes (S7-300, S7-400)

Note

The following control interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count5V control interface (outputs) for measuring modes.

Address	Assignment			
Bytes 0 to 3	**Low or high limit**			
	Function of DO1			
	Byte 0	Bit 1	Bit 0	Function DO1
		0	0	Output
		0	1	Outside limits
		1	0	Violating low limit
		1	1	Exceeding high limit
	Bytes 1 to 3	Reserved = 0		
	Integration time/update time			
	Bytes 0, 1	Integration time [n × 10 ms] (Value range 1 to 1000/12000)		
	Bytes 2, 3	Reserved = 0		
				Name

Address	Assignment		
Byte 4	Bit 7	Error acknowledgment	EXTF_ACK
	Bit 6	Enable DO2	CTRL_DO2
	Bit 5	Control bit DO2	SET_DO2
	Bit 4	Enable DO1	CTRL_DO1
	Bit 3	Control bit DO1	SET_DO1
	Bit 2	Start resetting of status bit	RES_STS
	Bit 1	Reserved = 0	
	Bit 0	SW gate control bit	SW_GATE
Byte 5	Bit 7	Reserved = 0	
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Change function of DO1	C_DOPARAM
	Bit 3	Reserved = 0	
	Bit 2	Change integration time	C_INTTIME
	Bit 1	Load high limit	LOAD_PREPARE
	Bit 0	Load low limit	LOAD_VAL
Bytes 6 to 7	Reserved = 0 [1]		

[1] Not used for extended user interface

Notes on the Control Bits

Bit	Explanation
C_DOPARAM	Change function of DO1 The value from byte 0 is adopted as the new function of DO1.
C_INTTIME	Change integration time The value from bytes 0 and 1 is adopted as the new integration time for the next measurement.
CTRL_DO1	Enable DO1 You use this bit to enable the DO1 output.
CTRL_DO2	Enable DO2 You use this bit to enable the DO2 output.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load high limit The value from bytes 0 to 3 is applied as the new high limit.
LOAD_VAL	Load low limit The value from bytes 0 to 3 is applied as the new low limit.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SET_DO1	Control bit DO1 Switches the DO1 digital output on and off when CRTL_DO1 is set.

Bit	Explanation
SET_DO2	Control bit DO2 Switches the DO2 digital output on and off when CRTL_DO2 is set.
SW_GATE	Software gate control bit The software gate is opened/closed via the control interface with the SW_GATE bit.

Feedback interface for measuring modes (S7-300, S7-400)

Note

The following feedback interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count5V feedback interface (inputs) for measuring modes.

Address	Assignment		Name
Bytes 0 to 3	Measured value		
Byte 4	Bit 7	Encoder supply short circuit	ERR_24V
	Bit 6	Short circuit / wire break / overtemperature	ERR_DO1
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Short circuit / wire break / overtemperature	ERR_DO2
	Bit 3	Short circuit / wire break / encoder signal	ERR_ENCODER
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Status down	STS_C_DN
	Bit 6	Status up	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	DO2 status	STS_DO2
	Bit 3	DO1 status	STS_DO1
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE

Address	Assignment		Name
Byte 6	Bit 70	Reserved = 0	
	Bit 6	Low limit, measuring range	STS_UFLW
	Bit 5	High limit, measuring range	STS_OFLW
	Bit 4	Reserved = 0	
	Bit 3	Measurement completed	STS_CMP1
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Reserved = 0	
Byte 7	Reserved = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the feedback bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged with the control bit (Page 413)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_DO1	Short circuit/wire break/overtemperature at output DO1 The error bit must be acknowledged with the control bit (Page 413)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_DO2	Short circuit/wire break/overtemperature at output DO2 The error bit must be acknowledged with the control bit (Page 413)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_ENCODER	Short circuit / wire break / encoder signal The error bit must be acknowledged with the control bit (Page 413)EXTF_ACK Diagnostic alarm if parameterized.
ERR_LOAD	Load function error The LOAD_VAL, LOAD_PREPARE, C_DOPARAM and C_INTTIMEcontrol bits (Page 413) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_CMP1	Measurement completed The measured value is updated each time a time interval (update time/integration time) elapses. **Measurement with integration time:** The end of a measurement (after the interval has elapsed) is indicated by the STS_CMP1 status bit. **Continuous measurement:** The end of the measurement is signaled by the status bit STS_CMP1 at the end of the update time if a measured value is output. The bit remains at "0" if an estimated measured value is output. This bit is reset by the control bit (Page 413)RES_STS in the control interface.

Bit	Explanation
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_DO1	DO1 status
STS_DO2	DO2 status
STS_GATE	Internal gate status: Measuring.
STS_LOAD	Load function is running
STS_OFLW STS_UFLW	High limit violated Low limit violated Both bits must be reset.

Position feedback parameters (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP (S7-300, S7-400)

Setting response to CPU/Master STOP

You can configure the response of the 1Count5V to failure of the higher-level controller.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count5V.

Leaving the Assigned State

Under what conditions does the 1Count5V exit the parameterized status?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 426).

Automatic New Parameter Assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon power on of the CPU/DP master
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count5V is inserted
- Upon power on or inserting of the appropriate power module

Selecting signal evaluation (S7-300, S7-400)

Signal Evaluation A, B

Signal evaluation by means of A, B allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Rotary Encoder (Page 206)
  The 1Count5V can count the edges of the signals. Normally, only the edges at A are evaluated (single evaluation). To obtain a higher resolution, you can select whether the signals are to undergo single (Page 206), double (Page 207) or quadruple (Page 207) evaluation when assigning parameters ("Signal evaluation" parameter).
  Multiple evaluation is only possible with asymmetric incremental encoders with A and B signals that are 90 degrees out of phase.

  Note

  500 kHz for count frequency is the maximum frequency of the A and B signals. This means a maximum frequency of 1 MHz for the count pulses in double evaluation and a maximum of 2 MHz in quadruple evaluation.

Activating signal pair diagnostics (S7-300, S7-400)

Signal pair monitoring

Signal pairs A and /A, B and /B, and N and /N can be monitored.

Activate the corresponding check box.

Note

You cannot activate "N, /N diagnostics" until you have activated "A, /A and B, /B diagnostics".

Selecting the count direction (S7-300, S7-400)

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Selecting the gate function (S7-300, S7-400)

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 202): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 202): The counting process starts from the last current counter value after gate stop and restart.

DI function (S7-300, S7-400)

Selecting DI function (S7-300, S7-400)

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 201), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Latch/Retrigger with positive edge (Page 420)
  Counter values are saved (latched) with positive edges at the digital input. The counter is set to the load value each time the values are saved.
- Synchronization with positive edge (Page 421)
  The counter is set to the load value with positive edges at the digital input.
- Latching with positive edge (Page 423)
  Counter values are saved (latched) with positive edges at the digital input.
- Hardware enable for synchronization (Page 424)
  The digital input acts as a hardware enable. The encoder zero mark sets the counter to the load value when the hardware enable is active.

The default setting is "Input".

Latch/Retrigger with positive edge (S7-300, S7-400)

Latch/Retrigger

This function saves the current internal counter value of the 1Count5V and retriggers counting when there is a positive edge at the digital input. This means that the current internal counter value at the time of the positive edge is stored (latch value), and the 1Count5V is then loaded again with the load value, from which counting resumes.

The following figure shows latch/retrigger when load value = 0.

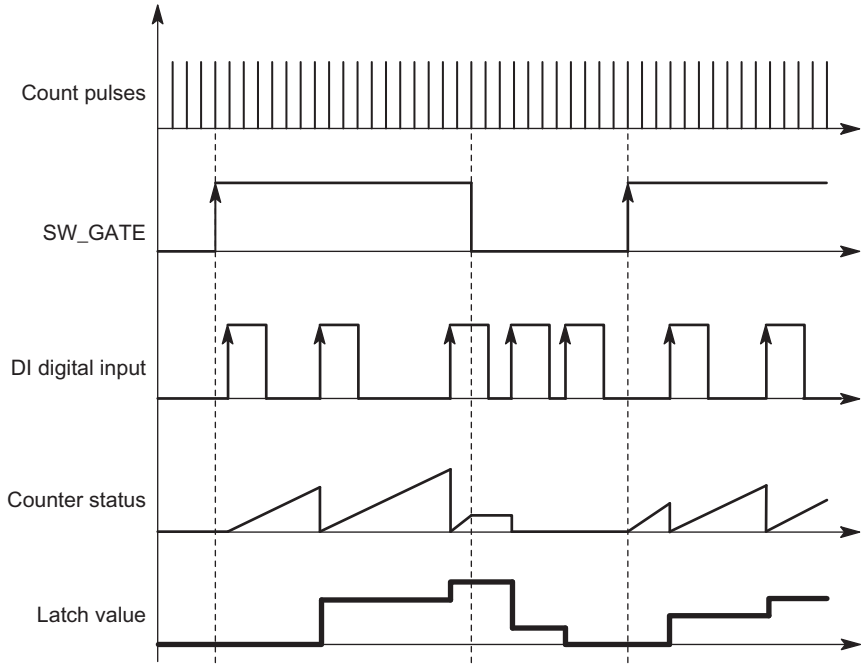

The counting mode must be enabled with the SW gate before the function can be executed. It is started with the first positive edge on the digital input.

The saved rather than the current counter value is displayed in the feedback interface (Page 428). The STS_DI bit indicates the status of the latch and retrigger signal.

The latch value is preassigned its RESET status ("0"). It is not changed when the SW gate is opened.

Direct loading of the counter does not cause the indicated stored counter status to be changed.

If you close the SW gate, counting is only interrupted; this means that when you open the SW gate again, counting is continued. The digital input DI remains active even when the SW gate is closed.

Counting is also latched and triggered in isochronous mode with each edge on the digital input. The counter status that was valid at the time of the last edge before T_i is displayed in the feedback interface.

Extended feedback interface

If the 1Count5V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 428).

Synchronization with positive edge (S7-300, S7-400)

Synchronization

If you have configured "Synchronization with positive edge", the rising edge of a reference signal at the input will set 1Count5V to the load value.

You can choose between once only and periodic synchronization ("Synchronization" parameter).

The figure below shows the counter values in once only and periodic synchronization.

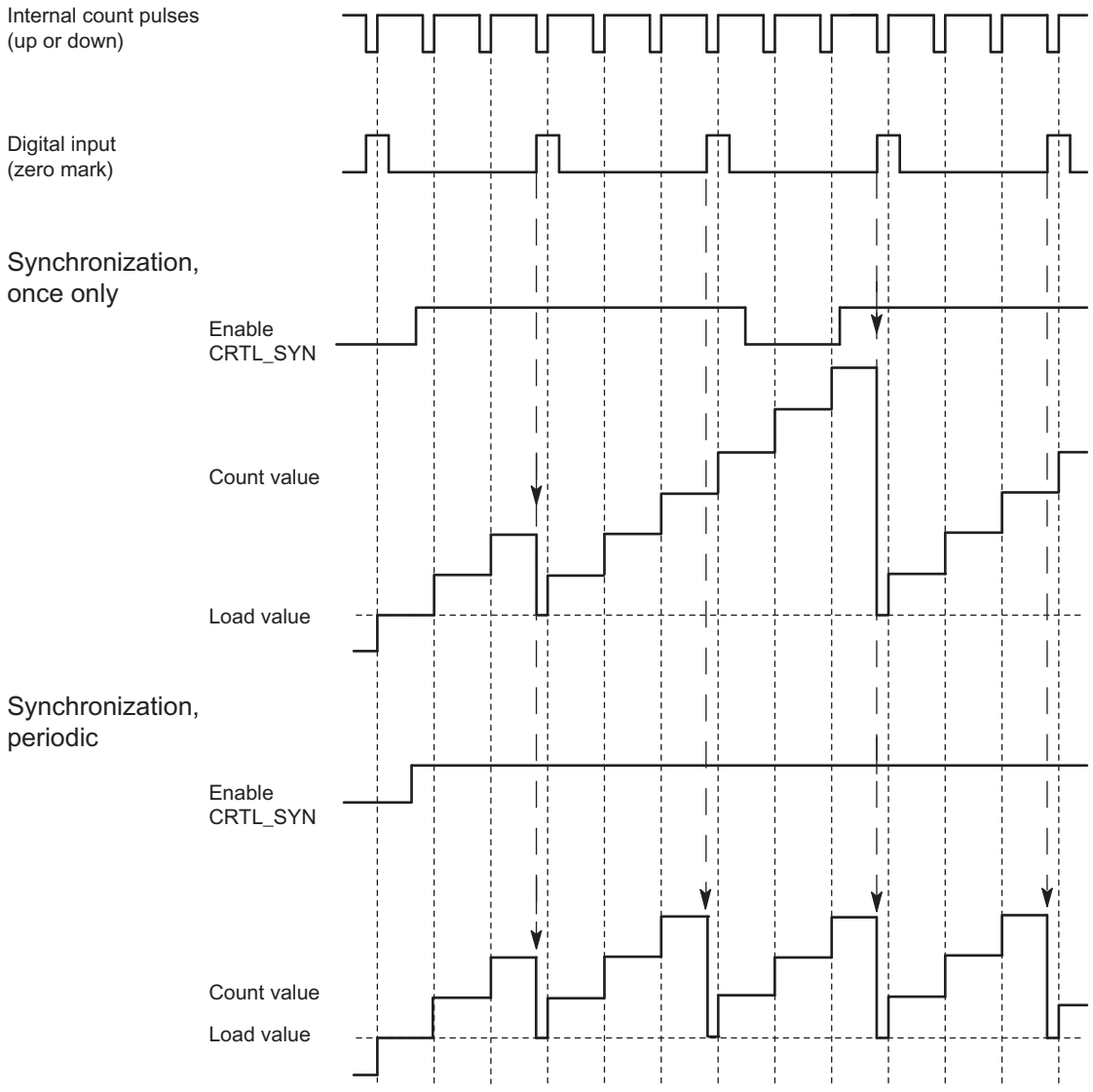

The following conditions apply:

- The counting mode must have been started with the software gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 426) must be set.
- Synchronize once only: If the enable bit is set, the first edge loads 1Count5V with the load value.
- Synchronization periodically: If the enable bit is set, the first and each subsequent edge loads the 1Count5V with the load value.
- After successful synchronization, the STS_SYNfeedback bit (Page 428) is set and the SYN LED comes on. The RES_STScontrol bit (Page 426) resets the feedback bit and switches off the LED.
- The signal of a bounce-free switch can serve as the reference signal.
- The feedback bit (Page 428)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 428)STS_SYN indicates that the rising edge at the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Latching with positive edge (S7-300, S7-400)

Latching

Counter value and latch value are pre-assigned their RESET statuses ("0").

The counting function is started when the SW gate is opened. 1Count5V begins at the load value ("0").

The following figure shows latching when load value = 0.

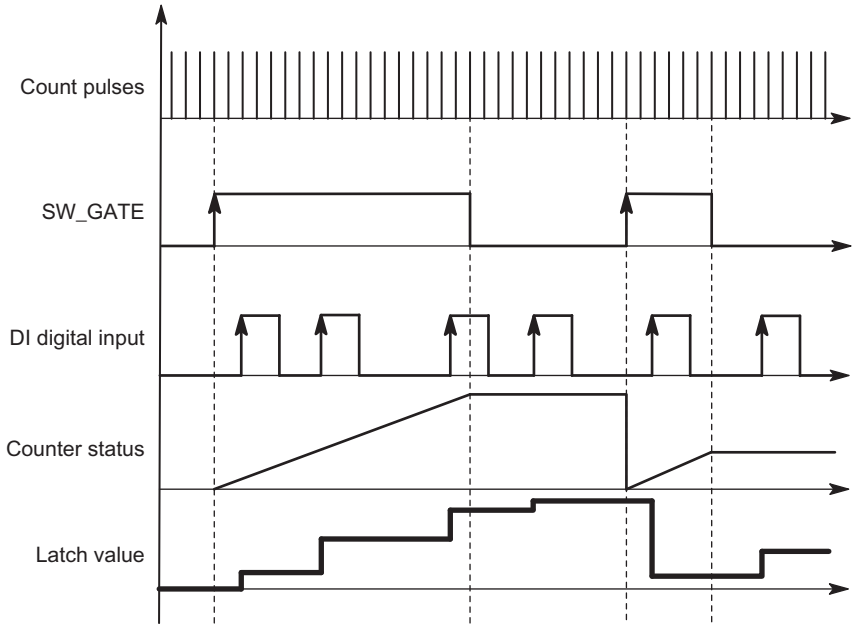

The latch value is always the exact counter status at the time of the positive edge on the digital input DI.

The saved rather than the current counter value is displayed in the feedback interface (Page 428). The STS_DI bit indicates the level of the latch signal.

Directly loading the counter does not alter the saved counter value displayed.

In isochronous mode, the counter status that was latched at the time of the last positive edge before T_i is displayed in the feedback interface.

When you close the SW gate, the effect is either canceling or interrupting, depending on the parameter assignment. The digital input DI remains active even when the software gate is closed.

Extended feedback interface

If the 1Count5V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 428).

Hardware enable for synchronization (S7-300, S7-400)

Synchronization with DI and Zero Mark

The DI will act as the hardware enable if you have configured "Hardware enable for synchronization". When the HW enable is active, the 1Count5V is loaded with the load value by the zero mark of the encoder.

You can choose between once only and periodic synchronization ("Synchronization (Page 426)" parameter).

The figure below shows the counter values in once only and periodic synchronization.

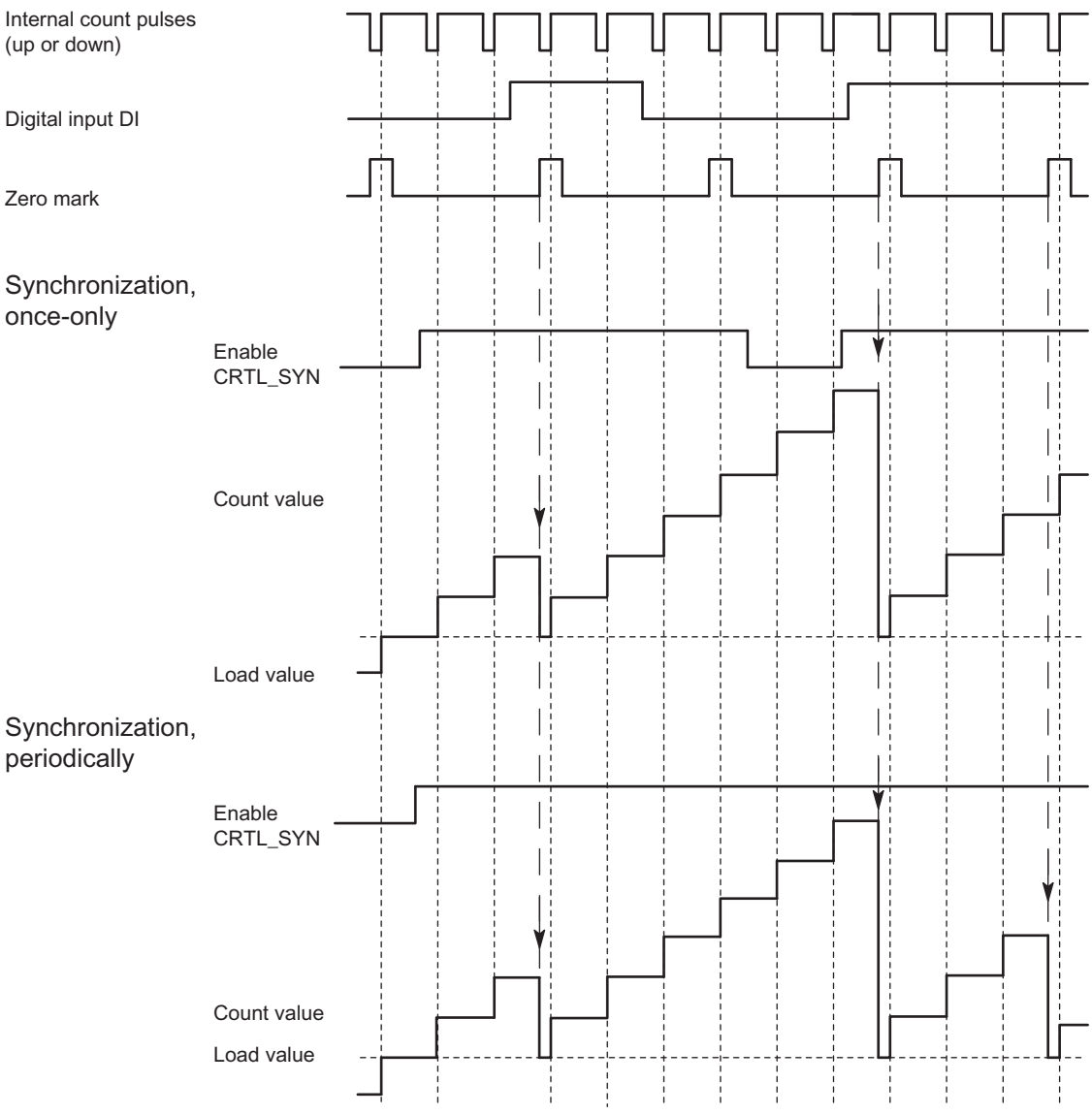

The following conditions apply:

- The counting mode must have been started with the software gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 426) must be set.
- Synchronize once only: If the enable bit is set and the HW enable is present, the first zero mark loads the 1Count5V with the load value.
- Synchronization periodically: If the enable bit is set and the HW enable is present, the first and each subsequent zero mark will load 1Count5V with the load value.
- After successful synchronization, the STS_SYNfeedback bit (Page 428) is set and the SYN LED comes on. The RES_STScontrol bit (Page 426) resets the feedback bit and switches off the LED.
- The signal of a bounce-free switch can serve as the reference signal.
- The feedback bit (Page 428)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 428)STS_SYN indicates that the positive edge on the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Selecting hardware gate input signal (S7-300, S7-400)

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 428)STS_DI indicates the level of the digital input.

Selecting synchronization (S7-300, S7-400)

Synchronization

This is where you can select either once only or periodic synchronization if you have selected "Synchronization with positive edge" at the "DI function (Page 419)" parameter:

- Once: If the enable bit is set, the first edge loads the 1Count5V with the load value.
- Periodic: If the enable bit is set, the first and each subsequent edge loads 1Count5V with the load value.

Control and feedback interface for position feedback (S7-300, S7-400)

Position feedback control interface (S7-300, S7-400)

Note

The following control interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count5V control interface (outputs) for position feedback.

Address	Assignment		Name
Bytes 0 to 3	Load value directly, in preparation		
Byte 4	Bit 7	Error acknowledgment	EXTF_ACK
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Trigger status bits reset	RES_STS
	Bit 1	Enable synchronization	CTRL_SYN
	Bit 0	Software gate control bit	SW_GATE
Byte 5	Bit 7	Reserved = 0	
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	Load counter in preparation	LOAD_PREPARE
	Bit 0	Load counter direct	LOAD_VAL
Bytes 6 to 7	Reserved = 0 [1]		

[1] Not used for extended user interface

Notes on the control bits

Bit	Explanation
CTRL_SYN	You use this bit to enable synchronization.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load counter in preparation The value from bytes 0 to 3 is applied as the load value.
LOAD_VAL	The value from bytes 0 to 3 is loaded directly as the new count value.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SW_GATE	Software gate control bit The software gate is opened/closed via the control interface with the SW_GATE bit.

Position feedback feedback interface (S7-300, S7-400)

Note

The following feedback interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count5V feedback interface (inputs) for position feedback.

Address	Assignment		Name
Bytes 0 to 3	Counter value or stored counter value in the case of the latch function at the digital input		
Byte 4	Bit 7	Encoder supply short circuit	ERR_24V
	Bit 6	Reserved = 0	
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Status down	STS_C_DN
	Bit 6	Status up	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 7	Zero crossing	STS_ND
	Bit 6	Low count limit	STS_UFLW
	Bit 5	High count limit	STS_OFLW
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Synchronization status	STS_SYN
Byte 7	Reserved = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the feedback bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged with the control bit (Page 426)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_LOAD	Load function error The LOAD_VAL and LOAD_PREPAREcontrol bits (Page 426) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_GATE	Internal gate status: Counting.
STS_LOAD	Load function is running
STS_ND	Zero crossing in the count range during counting without a main count direction. The bit must be reset by the control bit (Page 426)RES_STS.
STS_OFLW STS_UFLW	High count limit violated Low count limit violated Both bits must be reset.
STS_SYN	Synchronization status The STS_SYN bit is set after successful synchronization. It must be reset by the control bit (Page 426)RES_STS.

Fast mode parameters (S7-300, S7-400)

Selecting reaction to CPU/Master STOP (S7-300, S7-400)

Setting response to CPU/Master STOP

You can configure the response of the 1Count5V to failure of the higher-level controller.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count5V.

Automatic new parameter assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon power on of the CPU/DP master
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count5V is inserted
- Upon power on or inserting of the appropriate power module

Selecting signal evaluation (S7-300, S7-400)

Signal Evaluation A, B

Signal evaluation by means of A, B allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Rotary Encoder (Page 206)
  The 1Count5V can count the edges of the signals. Normally, only the edges at A are evaluated (single evaluation). To obtain a higher resolution, you can select whether the signals are to undergo single (Page 206), double (Page 207) or quadruple (Page 207) evaluation when assigning parameters ("Signal evaluation" parameter).
  Multiple evaluation is only possible with asymmetric incremental encoders with A and B signals that are 90 degrees out of phase.

Note

500 kHz for count frequency is the maximum frequency of the A and B signals. This means a maximum frequency of 1 MHz for the count pulses in double evaluation and a maximum of 2 MHz in quadruple evaluation.

Activating signal pair diagnostics (S7-300, S7-400)

Signal pair monitoring

Signal pairs A and /A, B and /B, and N and /N can be monitored.

Activate the corresponding check box.

Note

You cannot activate "N, /N diagnostics" until you have activated "A, /A and B, /B diagnostics".

Selecting the count direction (S7-300, S7-400)

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Selecting the gate function (S7-300, S7-400)

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 202): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 202): The counting process starts from the last current counter value after gate stop and restart.

DI function (S7-300, S7-400)

Selecting DI function (S7-300, S7-400)

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 201), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Synchronization with positive edge (Page 432)
  The counter is set to the load value with positive edges at the digital input.
- Hardware enable for synchronization (Page 433)
  The digital input acts as a hardware enable. The encoder zero mark sets the counter to the load value when the hardware enable is active.

The default setting is "Input".

Synchronization with positive edge (S7-300, S7-400)

Synchronization

If you have configured "Synchronization with positive edge", the positive edge of a reference signal at the input will set 1Count5V to the initial value (= load value).

The figure below shows the counter values for synchronization with a positive edge.

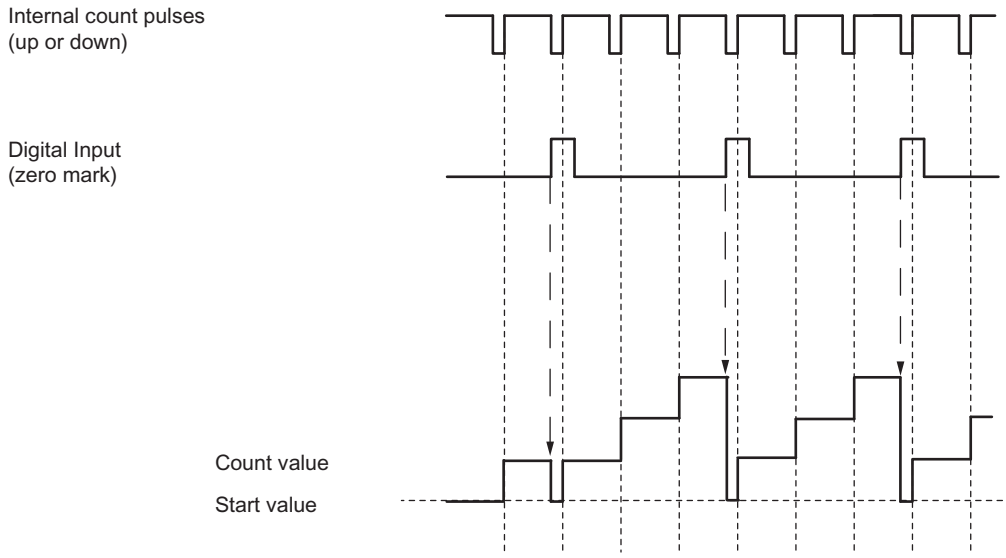

The following conditions apply:

- "Fast mode" must be active (hardware gate).
  - When synchronization is activated, the first edge and each additional edge loads 1Count5V with the initial value (=load value).
- The signal of a bounce-free switch or the zero mark of a rotary encoder can serve as the reference signal.
- The STS_DI feedback bit indicates the level of the reference signal.

Hardware enable for synchronization (S7-300, S7-400)

Synchronization with DI and Zero Mark

The DI will act as the hardware enable if you have configured "Hardware enable for synchronization". When the HW enable is active, 1Count5V is loaded with the initial value (load value) by the zero mark of the encoder.

The figure below shows the counter status following hardware enable at the DI and synchronization with a positive edge.

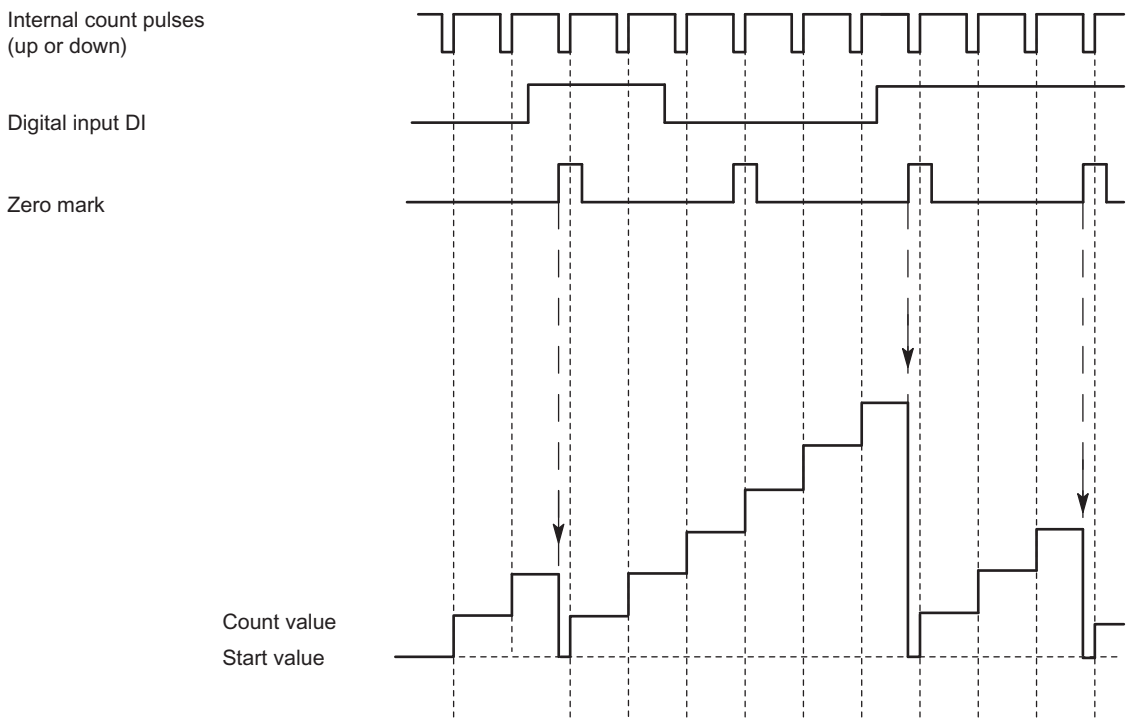

The following conditions apply:

- The counting mode must have been started with the SW gate.
- The "Enable synchronization" (CTRL_SYN) control bit must be set.
- After successful synchronization, the STS_SYNfeedback bit (Page 435) is set and the SYN LED comes on. The RES_STS control bit resets the feedback bit and switches the LED off.
- The signal of a bounce-free switch can serve as the reference signal.
- The feedback bit (Page 435)STS_DI indicates the level of the reference signal.

In isochronous mode, the set STS_SYN feedback bit indicates that the positive edge on the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Selecting hardware gate input signal (S7-300, S7-400)

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 435)STS_DI indicates the level of the digital input.

Specifying load value (S7-300, S7-400)

Load value

You can specify a load value for 1Count5V.

This load value is applied directly as the initial value.

Valid value range: -16777216 to +16777215

Fast mode feedback interface (S7-300, S7-400)

Note

The following feedback interface data is consistent for 1Count5V:

- Bytes 0 ... 3

Feedback interface assignment

The table below shows the assignment of the 1Count5V feedback interface (inputs) for fast mode.

Address	Assignment		Name
Bytes 0 to 3	Bit 31	Lifebeat	LZ
	Bit 30	Isochronous mode applied	STS_TIC
	Bit 29	Parameter Assignment Error	ERR_PARA
	Bit 28	Group error • Encoder supply short circuit • Short circuit / wire break / encoder signal	EXTF
	Bit 27	DI status	STS_DI
	Bit 26	Status of direction up/down	STS_DIR
	Bit 25	Internal gate status	STS_GATE
	Bits 0 to 24	Count value	

Notes on the Feedback Bits

Bit	Explanation
ERR_PARA	Errors in assigned module parameters.
EXTF	Group error Possible causes: • Encoder supply short circuit • Short circuit or encoder signal wire break EXTF is reset when the causes of the errors have been eliminated.
LZ	The lifebeat is toggled each time the feedback interface is updated, i.e. the last value sent is inverted.
STS_DI	The bit displays the status of the digital input DI.
STS_DIR	Direction status: • for encoder value change from larger to smaller encoder positions (including zero pass) → "1 " • for encoder value change from smaller to larger encoder positions (including zero pass) → "0 "
STS_GATE	Internal gate status: Counting.
STS_TIC	Isochronous mode (if assigned) was applied.

1Count5V diagnostics (S7-300, S7-400)

Diagnostics using the LED display (S7-300, S7-400)

LED display on the 1Count5V

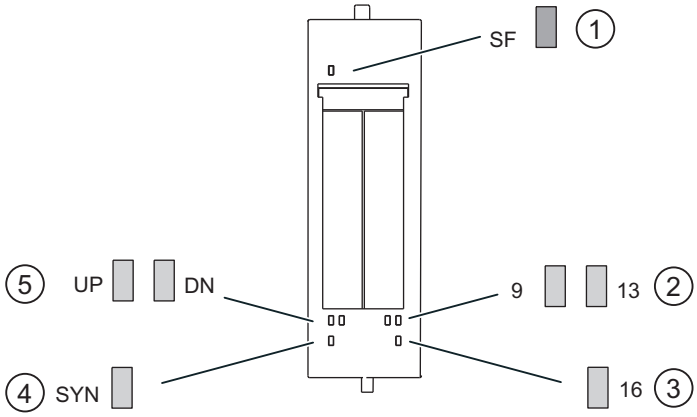

① Group error (red)
② Status display for the digital output (green)
③ Status display for digital input (green)
④ Status display for synchronization (green)
⑤ Status display for counting direction (green)

Status and error displays by means of LEDs on the 1Count5V

The table below shows the status and fault displays on the 1Count5V.

Event (LED)							Cause	Remedy
SF	UP	DN	SYN	9	13	16		
On							No parameter assignment. There is a diagnostic message.	Check the parameter assignment. Evaluate the diagnostics data.
	On						Status of the low-order bits of the counter, if the counter counts up.	
		On					Status of the inverse low-order bits of the counter, if the counter counts down.	
			On				Synchronization is performed (only in counting modes; image of STS_SYN acknowledgement bit).	
				On			DO 1 is activated.	
					On		DO 2 is activated.	
						On	DI is activated.	

Error types (S7-300, S7-400)

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

1Count5V error types

The following table shows the 1Count5V error types.

Error class		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the sensor supply or the actuator.	Check the wiring to the sensor. Correct the process wiring.
5_D	00101: Overtemperature	The digital output is overloaded.	Correct the process wiring.
6_D	00110: Open circuit	Line to the actuator is interrupted.	Correct the process wiring.
9_D	01001: Error	Internal module error occurred.	Replace the module.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.
26_D	11010: External error	Wire break/short circuit of the 5 V sensor signals: A, /A, B, /B, N, /N,	Correct the parameter assignment.

Cam control (S7-300, S7-400) 2

2.1 Cam control (S7-300, S7-400)

2.1.1 Cam control basics (S7-300, S7-400)

2.1.1.1 What is cam control? (S7-300, S7-400)

Cam control

Cam controllers are implemented to activate position-dependent or time-dependent functions. They are far superior to mechanical cam controllers, in particular due to their high flexibility; changes can for example be implemented during operation using software.

A cam generates position-dependent switching signals depending on the position values of the axis or an external encoder. You distinguish between position-based and time-based cams.

A cam is defined by the following:

- Initial position and final position (position-based cams (Page 439))
- Initial position and ON period (time-based cams (Page 439))

2.1.1.2 Properties of the cam types (S7-300, S7-400)

Cam types

You can assign each cam for operation as a position-based cam or time-based cam.

The following table sets outs the properties of the two cam types:

	Position-based cams	Time-based cams
Representation	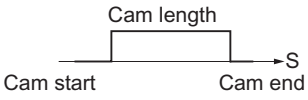	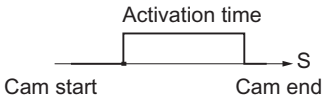
Parameter assignment	The following parameters are assigned: • Cam start • Cam end • Effective direction • Lead time	The following parameters are assigned: • Cam start • Activation time • Effective direction • Lead time

	Position-based cams	Time-based cams
Effective direction	Two effective directions are supported: • positive: The cam is activated at the cam start, if the axis is moving in direction of increasing actual values. • negative: The cam is activated at the cam end, if the axis is moving in direction of decreasing actual values. You can set both effective directions in parallel.	Two effective directions are supported: • positive: The cam is activated at the cam start, if the axis is moving in direction of increasing actual values. • negative: The cam is activated at the cam start if the axis is moving in direction of decreasing actual values. You can set both effective directions in parallel.
Enabling	The cam is activated: • at the cam start, if the axis is moving in positive direction, and a positive effective direction is set. • at the cam end if the axis is moving in a negative direction and a negative effective direction is set. • when the actual value lies within the cam range.	The cam is activated: • at the cam start if the axis is moving in positive direction and a positive effective direction is set or the axis is moving in a negative direction and a negative effective direction is set. After it has been activated, the full cam activation time elapses. even if the direction of motion of the axis changes after the cam is activated. The cam is **not** retriggered if its start position is passed once again within the cam activation time.
Deactivating	The cam is deactivated if: • the assigned distance has been traveled • its effective direction is opposite to the direction of the axis motion, and you have not assigned a hysteresis • the actual value is out of the cam range.	The cam is deactivated on expiration of the assigned time.
Path length	The path length of the cam is defined by the cam start and cam end. The cam start and cam end belong to the active section of the cam.	The path length of the cam is determined by the axis velocity within the cam activation time.
ON period	The on period of the cam is determined by the speed at which the axis travels across the path length of the cam.	The on period of the cam is assigned with the activation time.

Direction recognition

The direction of the axis motion is determined as follows:

- At each pulse of the incremental encoder.
- With each error-free frame of an SSI encoder.

Inverse cam

An inverse cam can briefly be generated with "Change cam edge" if the new cam start is greater than the old cam end. The table below shows the effect of an inverse cam on a linear and on a rotary axis.

Inverse cam with a linear axis	Inverse cam with a rotary axis
	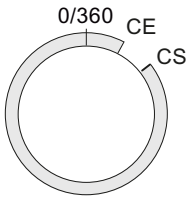
The cam start is greater than the cam end	The cam start is more positive than the cam end
With both types of axis there must always be a minimum clearance of 4 pulses between the cam start and the cam end.	

2.1.1.3 Normal tracks and track results (S7-300, S7-400)

Cam tracks

The 32 tracks can be used to control up to 32 different switching actions. You can evaluate the tracks based on the checkback signals.

The first 13 tracks (track 0 to 12) are each assigned a digital output (Q0 to Q12) of FM 352. This output can be used for the direct control of a connected contactor, for example.
The first 16 tracks (track 0 to 15) in the FM 452 are each assigned a digital output (Q0 to Q15).

Track result

The system provides up to 128 cams which can be assigned to user-specific tracks.

Each track can be assigned several cams. The track result is a logic OR operation derived from all cam values of this track.

Example of a track result

During parameter assignment, you define the following cams for track 3:

Cam	Cam start	Cam end
1	101 µm	106 µm
2	100 µm	104 µm

The figure below shows the track result.

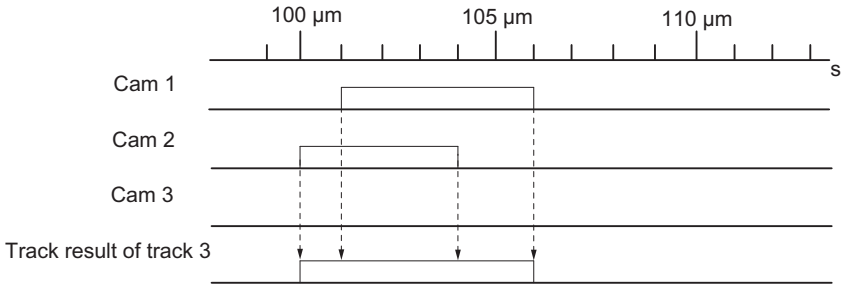

Track enable

You must activate the tracks used in order to transfer the track results of tracks 0 to 12 as track signals to the digital outputs Q0 to Q12 of FM 352. The FM 452 has tracks 0 to 15.

External enable of a track

You can assign an external enable signal for track 3 in the machine data. Track signal 3 is then logically linked to digital input I3 by an AND operation before it can switch digital output Q3 of the FM 352.

Digital output Q3 is thus only set if the following conditions are satisfied:

- The corresponding track is enabled.
- At least one cam on this track is active (track result = 1).
- The corresponding digital input I3 was set by an external event.

Tracks 3 to 10 are available for external enabling in the FM 452.

Setting the track signals

The track signals 0 to 12 (according to digital outputs Q0 to Q12) can be set by the cam control system, or by the CPU. Track signals 0 to 15 can be set in the FM 452.

2.1.1.4 Special tracks (S7-300, S7-400)

Definition

You can program tracks 0 to 2 for operation as special tracks:

- Track 0 or 1: Counter cam track
- Track 2: Brake cam track
  Input I0 must be evaluated in order to enable the track.

Requirements

Requirements of working with special tracks:

- You programmed the cams for the track
- Cam processing is enabled
- The corresponding track is enabled
- The track is programmed as special track

Counter cam track

A counter cam track counts the status transitions of the track results on this track.

Define a counter high counter value for the counter cam track and then start the counter function.

The counter value of the relevant track decrements by the count of 1 at each positive edge of the track result signal.

The track flag bit = 0 as long as the value of the counter cam track is not equal to zero.

The controller sets the track flag bit and, if assigned parameters accordingly, the track signal (Page 446) when the counter value = 0.

It resets the track flag bit and sets the default value of the counter at the next negative edge of the track result signal (all cams on this track are disabled).

The figure below shows how a counter cam track is switched.

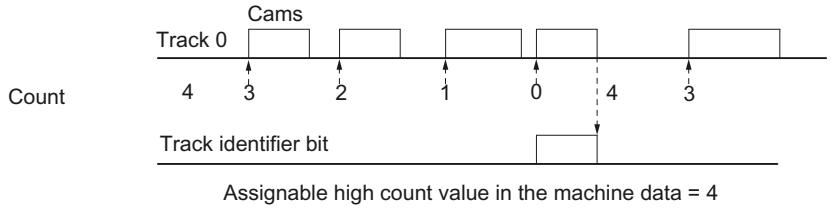

Brake cam track

In order to use track 2 as a brake cam track, interconnect digital input I0.

A positive edge of the I0 signal sets the track flag bit.

The track flag bit is reset again when:

- there is no longer a "1" signal at I0 **and** afterwards
- the controller has detected a negative edge at the track 2 result signal.

The figure below shows how a brake cam track is switched.

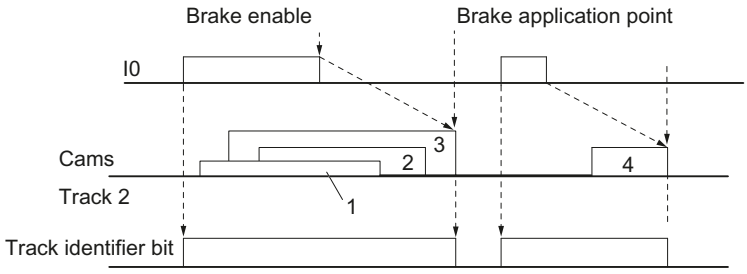

1 ...4 denote 4 output cams that act on the brake cam track

In the example, the track flag bit is reset by a negative edge at cam 3 or 4.

2.1.1.5 Hysteresis (S7-300, S7-400)

Definition

Mechanical imbalance at the axis may cause fluctuation of the actual position value. If the actual position value is offset by one edge of a cam, or within an active cam with only one effective direction, this cam's activation would be cycled on and off continuously. A hysteresis prevents this flutter.

A hysteresis setting is dependent on the actual value, and applies globally to all cams. It is enabled when a direction reversal is detected. A hysteresis will always take effect, regardless of whether or not a cam is set at the current axis position.

Rules for the hysteresis range

Rules applicable to the hysteresis range:

- The hysteresis will always be set when a directional reversal is detected.
- The indication of the actual value remains constant within the hysteresis.
- The direction is not redefined within the hysteresis.
- A position-based cam is neither activated nor deactivated within the hysteresis.
- A time-based cam is not activated within the hysteresis. An active time-based cam is deactivated on expiration of the assigned activation (not only on reaching the hysteresis limit).
- When the value is out of the hysteresis range, the FM x52 sets:
  - the actual position value
  - the current direction of motion of the axis
  - the current states of all cams
- The hysteresis range applies to all cams.

Direction reversal of a cam with hysteresis

The table below gives an example of the response to cam direction reversal. A distinction must be made between the reaction of position- and time-based cams. The effective direction of the cam is **positive**.

Position-based cam	Time-based cam
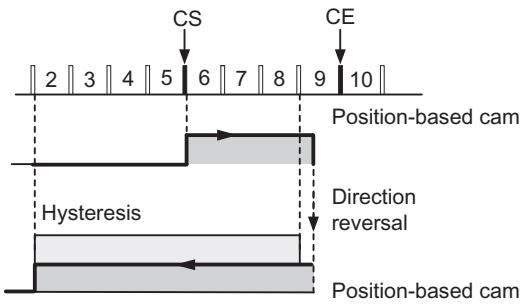	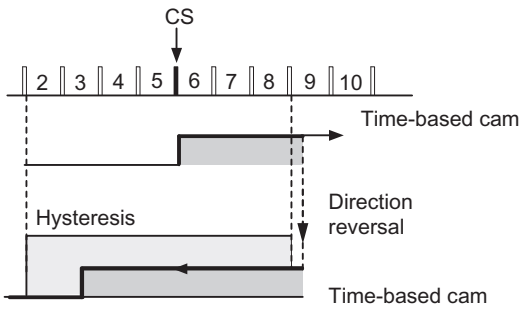
The hysteresis will be set when a directional reversal is detected. The cam is deactivated immediately when the the hysteresis range is violated.	The cam always remains active for the duration of the assigned activation time.
Cam Hysteresis	

2.1.1.6 Dynamic adjustment (S7-300, S7-400)

Introduction

The dynamic adjustment is used to compensate delay times of the connected control elements.

Lead time

You can program a delay time and assign it as derivative-action time to each cam. You can assign one derivative-action time to each cam. The derivative-action time applies to the cam start and end position.

Correction distance

The actuation distance of a cam is calculated continuously based on the current velocity and derivative-action time. The entire cam is shifted in direction of the actual value by this value. The programmed range is the "static range," and the range calculated based on the derivative-action time represents the "dynamic range."

Actuation distance = lead time x actual velocity of the axis

FM x52 calculates the actuation distance of all cams within 1/4 of the longest parameterized lead time.

An extremely high derivative-action time of a cam reduces the dynamic performance of cam processing.

2.1.1.7 Interfaces of the cam controller (S7-300, S7-400)

Overview

Taking FM 352 as an example, the schematic diagram below sets out the major interfaces of the cam controller to illustrate the connection between data, inputs and outputs. Differences to the FM 452 are set out in the table below.

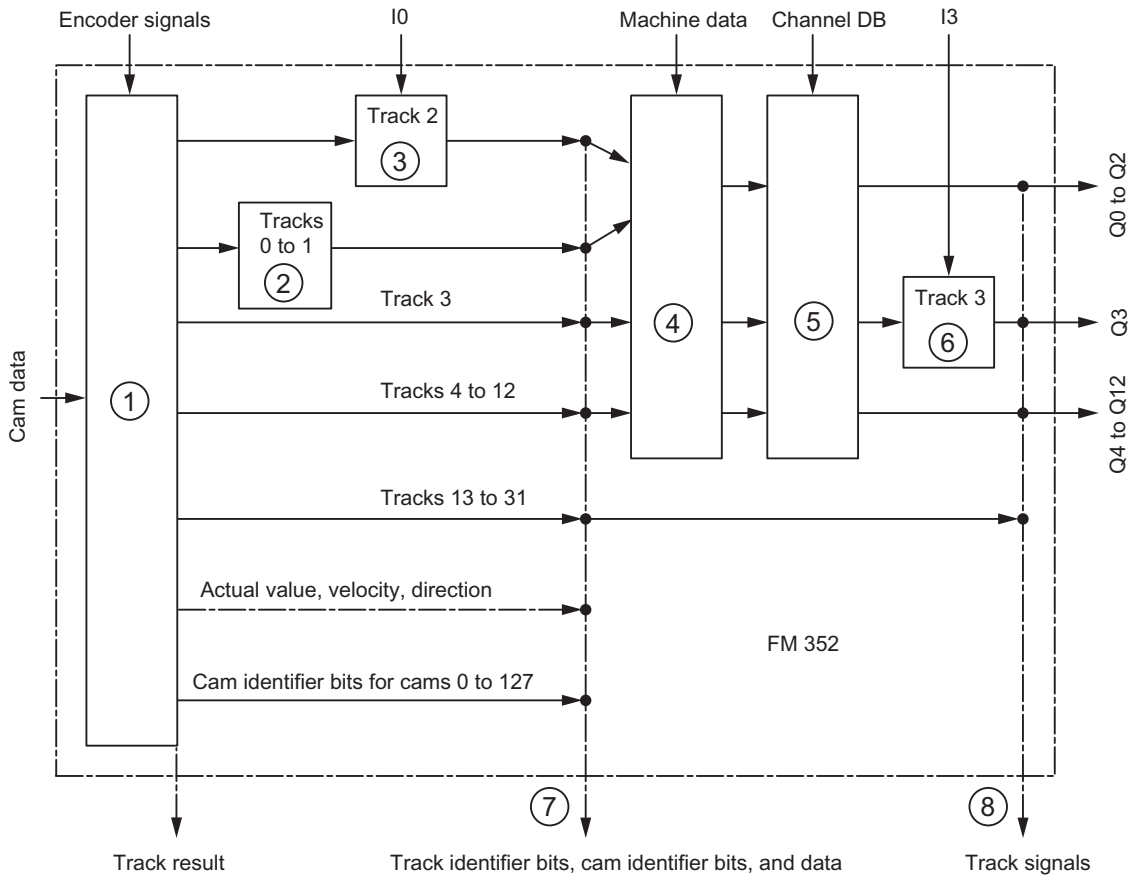

No.	Description
①	The cam processing functions of FM x52 calculate the cam identifier bits on the basis of the switching conditions and actual value. The module also determines the track results according to the assignment of the cams to the tracks.
②	If you assigned track 0 or 1 as a counter cam track, the track result of the cam controller (point 1) is logically linked to the counter result in order to produce the track identifier bit. The track identifier bit is otherwise equivalent to the track result.
③	If you assigned track 2 as a brake cam track, the track result of the cam controller (point 1) is logically linked to input I0 to produce the track identifier bit. The track identifier bit is otherwise equivalent to the track result.
④	Using machine data, you can control whether the previously determined track identifier bits of tracks 0 to 12 (in FM 352) or 0 to 15 (in FM 452) of the cam controller are transferred to the channel DB, or whether they are set directly by the track enable signal (TRACK_EN).

No.	Description
⑤	You enable the track signals of tracks 0 to 12 (or 0 to 15) by setting TRACK_EN and the count function by setting CNTC0_EN / CNTC1_EN.
⑥	FM 352: The track signal of track 3 and digital input I3 can be logically linked by an AND operation provided you have set this option in the machine data (EN_IN_I3).
	FM 452: The track signals of tracks 3 to 10 can be logically linked to digital inputs I3 to I10 with an AND operation if you have set this option in the machine data (EN_IN_I3 to EN_IN_I10).
⑦	All track and cam identifier bits can be read at this location (i.e. before being logically linked with machine and channel data) using the ACTPOS_EN or CAMOUT_EN job. At tracks 3 to 31, the track identifier bit is equivalent to the track result (point 1).
⑧	FM 352: The track signals of tracks 0 to 12 are available in the checkback signals after having been logically linked with the machine data (e.g. axis type, software limit switch, distance per encoder revolution...) and channel data. The track signals of tracks 13 to 31 and the track identifier bits of point 7 are identical. The track signals of tracks 0 to 12 are also available at the digital outputs Q0 to Q12.
	FM 452: The track signals of tracks 0 to 15 are available in the checkback signals after having been logically linked with the machine and channel data. The track signals of tracks 16 to 31 and the track identifier bits of point 7 are identical. The track signals of tracks 0 to 15 are also available at the digital outputs Q0 to Q15.

2.1.1.8 Encoder signals (S7-300, S7-400)

Supported encoders (S7-300, S7-400)

Connectable incremental encoders

Incremental encoders (Page 449) with two pulses 90° out of phase with or without zero marks are supported:

- Encoders with asymmetrical 24 V output signals
  - Limit frequency = 50 kHz
  - Cable up to 100 m long, shielded
- Encoders with symmetrical output signals with 5 V differential interface in accordance with RS 422
  - Limit frequency = 1 MHz
  - With 5 V voltage supply: Cable up to 32 m long, shielded
  - With 24 V voltage supply: Cable up to 100 m long, shielded

Note

If the 5 V encoder does not output a zero mark signal and wire-break monitoring is enabled, you must interconnect the zero mark signal inputs N and /N externally so that the inputs have different signal levels (for example, N to 5 V, /N to ground).

Supported initiators

The FM x52 supports the following initiators (Page 451):

- Initiators with 24 V signal level (proximity switches)
- Limit frequency = 50 kHz
- Cable up to 100 m long

Absolute encoders supported

Absolute encoders (Page 451) with serial interface are supported. Position data are transferred synchronously using the SSI protocol (**S**ynchronous**S**erial**I**nterface). The FM x52 only supports the GRAY code. Due to the arrangement of the data bits in the transferred frames, the data formats "fir tree", "half fir tree" and "right-justified" are used.

Absolute encoders are divided into the categories:

- Single-turn encoder
  The total range of single-turn encoders is scaled to one revolution.
- Multiturn encoder
  The total range of multiturn encoders is scaled to several revolutions.

Encoder type	Frame length / type
Single-turn encoder	13-bit half fir tree
Single-turn encoder	13-bit right-justified
Single-turn encoder	25-bit right-justified
Multiturn encoder	25-bit fir tree
Multiturn encoder	25-bit right-justified
Listen in	Fir tree
Listen in	Right-justified
Special setting: Multiturn encoder in single-turn mode	 25-bit half fir tree

The transmission rate for data transmission depends on the cable length:

- Max. 125 kHz for 320 m of cable, shielded
- Max. 250 kHz for 160 m of cable, shielded
- Max. 500 kHz for 60 m of cable, shielded
- Max. 1 MHz for 20 m of cable, shielded

Incremental encoder (S7-300, S7-400)

Signal shapes

The following figure illustrates the signal forms of encoders with asymmetrical and symmetrical output signals.

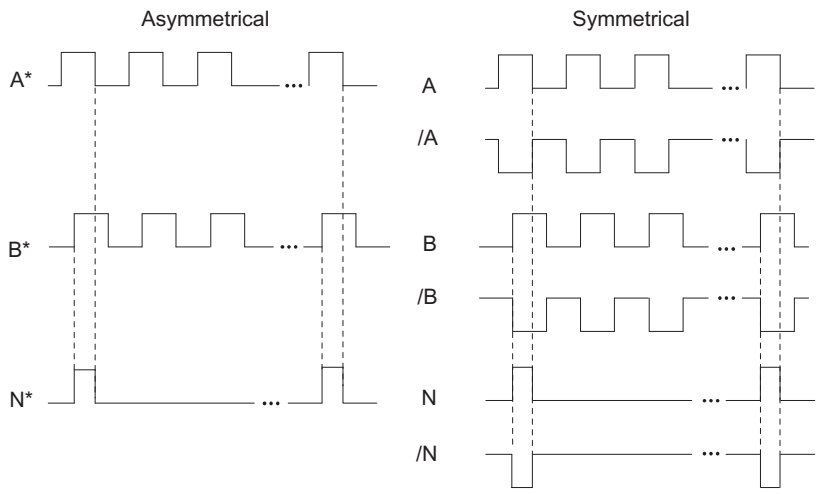

Signal evaluation

Increments

An increment identifies a signal period of the encoders signals A and B. This value is specified in the technical data and/or on the rating plate of the encoder.

The figure below shows the ratio of increments to pulses.

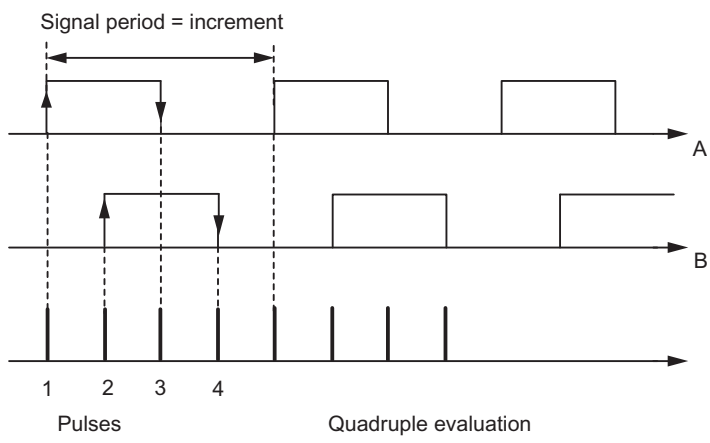

Pulses

FM x52 evaluates all 4 edges of signals A and B in each increment (quadruple evaluation).

1 increment (encoder default) = 4 pulses (FM evaluation)

Reaction times

FM x52 has the following response times for connected incremental encoders:

- Minimum reaction time = cam cycle time + switching time of the connected switching elements
- Maximum reaction time = 2 x cam cycle time + switching time of the connected switching elements

Example of reaction times

Example of the min./max. reaction time with 16 cams:

- Cam cycle time: approx. 20 µs
- Switching time of the hardware: approx. 150 µs

Minimum reaction time = 20 µs + 150 µs = 170 µs

Maximum reaction time = 2 x 20 µs + 150 µs = 190 µs

Note

You can compensate the reaction time by assigning the cam parameters accordingly or using dynamic adjustment.

Flat gain

The flat gain is equivalent to the difference between the min./max. reaction time. For incremental encoders this is:

Flat gain = cam cycle time

Note

If the switching time of the FM x52 hardware and of the connected switching elements can be disregarded, reliable cam activation is always ensured if the cam is longer than the distance covered within the cam cycle time.

Proximity switches (S7-300, S7-400)

Definition

Initiators are simple switches which output pulse-shaped signals, and do not return a directional signal. You define the direction with the machine data "Initiator direction".

CAUTION

Risk of material damage!

Incorrect direction settings can cause serious errors in the system (for example, faulty control of a unit of equipment).

Check the direction settings in the commissioning phase, and whenever you replace an initiator.

Signal evaluation

With an initiator, the rising edge of signal A* is counted.

Absolute encoder (SSI) (S7-300, S7-400)

Absolute encoder pulse evaluation

1 increment (encoder default) = 1 pulse (FM evaluation)

Listen in

"Listen in" means: An absolute encoder is operated in parallel in two modules (for example FM x51 and FM x52). The FM x51 positioning module is the master and clocks the absolute encoder; the FM x52 electronic cam controller acts as the slave listening in to the signals of the SSI frame.

Set "Increments/Encoder revolution" and "Number of revolutions" to the master setting. The "Transmission rate" is not relevant. Depending on the encoder type select "Listen in" or "Listen in Right-Justified" for "Frame Length".

Reaction times

The FM x52 has the following reaction times for absolute encoders:

- Minimum reaction time = frame run time + cam cycle time + switching time of the connected switching elements
- Maximum reaction time = 2 x frame runtime + monoflop time + 2 x cam cycle time + switching time of the connected switching elements
- For programmable absolute encoders:
  Maximum reaction time = frame runtime + monoflop time + 2 x cam cycle time + switching time of the connected switching elements +1/max. step sequence rate

Monoflop time

The following limit values apply to the monoflop time:

- Minimum monoflop time: > 15 µs
- Maximum monoflop time: < 64 µs

Encoders with values outside the limits shown here are not permitted.

Frame run time

The frame run times depend on the transmission rate:

Transmission rate	Frame run time for 13 bits	Frame run time for 25 bits
0.125 MHz	112 µs	208 µs
0.250 MHz	56 µs	104 µs
0.500 MHz	28 µs	52 µs
1000 MHz	14 µs	26 µs

Example of Reaction Times

The following example shows how to calculate the minimum and maximum reaction times. In the example a programmable encoder is not used.

- Cam cycle time: approx. 20 µs for max. 16 cams
- Switching time of the hardware: approx. 150 µs
- Frame run time: 26 µs at a transmission rate of 1 MHz (25-bit frame)
- Monoflop time: 20 µs (depends on the encoder: typical 20 µs to 40 µs

Maximum reaction time = 26 µs + 20 µs + 150 µs = 196 µs

Maximum reaction time = 2 x 26 µs + 20 µs + 2 x 20 µs + 150 µs = 262 µs

Note

You can compensate the reaction time by assigning the cam parameters accordingly or by using dynamic adjustment.

Flat gain

The flat gain is equivalent to the difference between the minimum and maximum reaction times.

For an absolute encoder, it is as follows:

- Flat gain = cam cycle time + frame run time + monoflop time

With programmable absolute encoders, it is as follows:

- Flat gain = cam cycle time + frame run time + monoflop time
  + 1/max. step sequence frequency

Note

If the switching time of the FM x52 hardware and of the connected switching elements can be disregarded, reliable cam activation is always ensured if the cam is longer than the distance covered within the cam cycle time.

2.1.2 Functions of the FM x52 (S7-300, S7-400)

2.1.2.1 Writing and enabling machine data (S7-300, S7-400)

Writing and activating machine data

You can use the machine data to adapt the FM x52 to the axis and the encoder.

You can change parameters during operation using the user program.

All parameters are stored in the parameter DB:

- Machine data are stored in the parameter DB at addresses 3.1 to 104.0.

Enter the parameter DB number in the corresponding channel DB.

You can enter the parameters with the DB Editor, or indeed simply and easily enter them in the "System of units", "Axis", "Encoder" and "Tracks" dialogs and write them to the parameter DB with the "Export" function.

You can use the "Import" function to import parameters from an existing parameter DB to the dialogs.

Initial parameter assignment

If the module does not yet contain any machine data (feedback signal PARA = 0), proceed as follows for initial parameter assignment without the parameter assignment dialogs:

1. Enter the new values in the parameter DB.
2. Download the parameter DB to the CPU.
3. Set the following trigger bit in the channel DB:
   – Write machine data (MDWR_EN = 1)
4. Call the CAM_CTRL or CAM_CTRL_452 instruction in the cyclic user program.

Modifying machine data

To modify existing machine data (feedback signal PARA = 1) using the user program, proceed as follows:

1. Enter the new values in the parameter DB.
2. Set the trigger bits at the channel DB:
   – Write machine data (MDWR_EN = 1)
   – Activate machine data (MD_EN = 1)
3. Call the CAM_CTRL or CAM_CTRL_452 instruction in the cyclic user program.
4. Check whether the previous cam data and the modified machine data are compatible.
5. Always rewrite the cam data of the parameterized cams, irrespective of whether or not they have been changed (CAM1WR_EN … CAM8WR_EN).

If you set the trigger bits simultaneously, the CAM_CTRL or CAM_CTRL_452 instruction ensures the jobs are processed in the correct order.

Otherwise, you must always change the machine data in the following order:

1. Write machine data
2. Activate machine data

Data used in the channel DB

Address	Name	Type	Start value	Comment
35.0	MDWR_EN	BOOL	FALSE	1 = write machine data
35.1	MD_EN	BOOL	FALSE	1 = enable machine data

Note

If any parameters relevant for synchronization were modified, the synchronization will be canceled when the machine data are activated. The functions will also be reset and all machine and cam data deleted from the module.

Parameters relevant to synchronization:

- Axis type
- End of rotary axis
- Signal type
- Distance per encoder revolution
- Increments per encoder revolution
- Number of revolutions
- Reference point coordinate
- Absolute encoder adjustment
- Type of reference point retriggering
- Direction adaptation
- Number of cams
- Software limit switch start and end

2.1.2.2 Reading machine data (S7-300, S7-400)

Reading machine data

To read actual machine data from the module:

1. Set the following trigger bit in the channel DB:
   – Reading machine data (MDRD_EN = 1)
2. Call the CAM_CTRL or CAM_CTRL_452 instruction in the cyclic user program.

This saves the current machine data to the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.1	MDRD_EN	BOOL	FALSE	1 = read machine data

2.1.2.3 Writing cam data (S7-300, S7-400)

Writing cam data

Cam data define the type and function principle of the cams and their assignment to the tracks.

Cam data are stored in the parameter DB, starting at address 108.0. These data are grouped in packets, each consisting of 16 cams.

Cam data are active immediately after having been written.

To write cam data without the parameter assignment dialogs, proceed as follows:

1. Enter the new values in the parameter DB.
2. Download the parameter DB to the CPU.
3. Set the trigger bits in the channel DB (CAM1WR_EN … CAM8WR_EN)
4. Call the CAM_CTRL or CAM_CTRL_452 instruction in the cyclic user program.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.3	CAM1WR_EN	BOOL	FALSE	1 = write cam data 1 (cams 0 to 15)
35.4	CAM2WR_EN	BOOL	FALSE	1 = write cam data 2 (cams 16 to 31)
35.5	CAM3WR_EN	BOOL	FALSE	1 = write cam data 3 (cams 32 to 47)
35.6	CAM4WR_EN	BOOL	FALSE	1 = write cam data 4 (cams 48 to 63)
35.7	CAM5WR_EN	BOOL	FALSE	1 = write cam data 5 (cams 64 to 79)
36.0	CAM6WR_EN	BOOL	FALSE	1 = write cam data 6 (cams 80 to 95)
36.1	CAM7WR_EN	BOOL	FALSE	1 = write cam data 7 (cams 96 to 111)
36.2	CAM8WR_EN	BOOL	FALSE	1 = write cam data 8 (cams 112 to 127)

2.1.2.4 Reading cam data (S7-300, S7-400)

Reading cam data

To read actual cam data from the module:

1. Set the following trigger bit in the channel DB:
   – Read cam data (CAM1RD_EN … CAM8RD_EN)
2. Call the CAM_CTRL or CAM_CTRL_452 instruction in the cyclic user program.

This saves the actual cam data to the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.2	CAM1RD_EN	BOOL	FALSE	1 = read cam data 1 (cams 0 to 15)
37.3	CAM2RD_EN	BOOL	FALSE	1 = read cam data 2 (cams 16 to 31)
37.4	CAM3RD_EN	BOOL	FALSE	1 = read cam data 3 (cams 32 to 47)
37.5	CAM4RD_EN	BOOL	FALSE	1 = read cam data 4 (cams 48 to 63)
37.6	CAM5RD_EN	BOOL	FALSE	1 = read cam data 5 (cams 64 to 79)
37.7	CAM6RD_EN	BOOL	FALSE	1 = read cam data 6 (cams 80 to 95)

Address	Name	Type	Initial value	Comment
38.0	CAM7RD_EN	BOOL	FALSE	1 = read cam data 7 (cams 96 to 111)
38.1	CAM8RD_EN	BOOL	FALSE	1 = read cam data 8 (cams 112 to 127)

2.1.3 Using FM 352 (S7-300, S7-400)

2.1.3.1 Terminal assignment of FM 352 front connector (S7-300, S7-400)

Description of the encoder interface (S7-300, S7-400)

Location of the sub D socket

The figure below shows the position and designation of the connector on the module. You can connect an initiator, incremental encoder or absolute encoder (SSI) to the sub D socket.

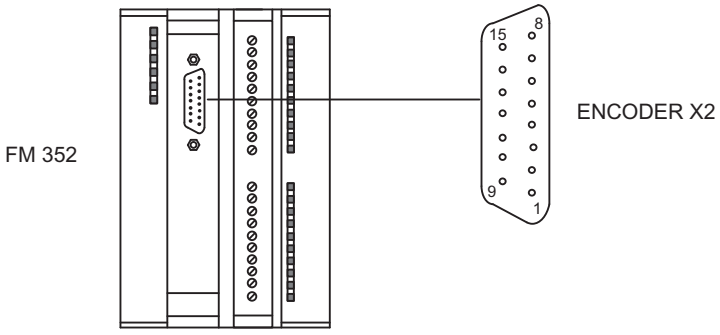

Pin assignment of the encoder interface (X2)

Pin	Name	Proximity switch	Incremental encoder	Absolute encoder
1	A*	Encoder signal A (24 V)		---
2	CLS	---	---	SSI shift clock
3	/CLS	---	---	SSI shift clock inverse
4	B*	---	Encoder signal B (24 V)	---
5	24 VDC	Encoder supply 24 V		
6	5.2 VDC	---	Encoder supply 5.2 V	
7	M	Ground		
8	N*	---	Zero mark signal (24 V)	---
9	RE	---	Current sourcing/current sinking[2)]	---
10	N	---	Zero mark signal (5 V)	---
11	/N	---	Zero mark signal inverse (5 V)	---

Pin	Name	Proximity switch	Incremental encoder	Absolute encoder
12	/B / /CLI [1]	---	Encoder signal B inverse (5 V)	SSI shift clock inverse
13	B / CLI [1]	---	Encoder signal B (5 V)	SSI shift clock
14	/A / /DAT	---	Encoder signal A inverse (5 V)	SSI data inverse
15	A / DAT	---	Encoder signal A (5 V)	SSI data

[1] In listen-in mode

[2]**Note:** If you are connecting an incremental encoder from a third party in series-mode switching (current sourcing/current sinking), the following must be observed:

- Current sourcing: Connect RE (9) with ground (7).
- Current sinking: Connect RE (9) with +24 V (5).

Terminal assignment of the front connector (S7-300, S7-400)

Front connector

Connect the following to the 20-pin front connector:

- the digital outputs and switching elements (inputs)
- the supply voltage of the encoders and digital outputs

Terminal assignment of front connector (X1)

The table below shows the terminal assignment of the front connector of FM 352.

Wiring	Name	Function
1	L+	24 V DC encoder supply, and digital outputs
2	M	Ground of the encoder supply and digital outputs
3	I 0	Brake enable
4	I 1	Length measurement/ edge detection/ set actual value on-the-fly
5	I 2	Home position switch
6	I 3	Enable track signal 3
7	Q 0	Digital output 0
8	Q 1	Digital output 1
9	Q 2	Digital output 2
10	Q 3	Digital output 3
11	Q 4	Digital output 4
12	Q 5	Digital output 5
13	Q 6	Digital output 6
14	Q 7	Digital output 7
15	Q 8	Digital output 8
16	Q 9	Digital output 9
17	Q 10	Digital output 10
18	Q 11	Digital output 11

Wiring	Name	Function
19	Q 12	Digital output 12
20	–	–

Additional information about design and wiring can be found in the Operating Instructions (http://support.automation.siemens.com/WW/view/en/2103044) of the electronic cam controller FM 352.

Auxiliary voltage for encoders and digital outputs (L+, M)

The 24 VDC auxiliary voltage of the encoders and digital outputs is monitored:

- for wire-break of the 24 V feed line
- for power failure

This supplies 24 V DC and 5.2 V DC at the encoder interface (sub D socket X2) to the various types of encoders.

Load current supplies

The DC load power supply must meet the following requirements:

Only a safe, isolated extra-low voltage of ≤ 60 V DC may be used as the load current supply. Safe isolation can be implemented, for example, by adhering to the specifications

- VDE 0100 Part 410 / HD 384-4-41 / IEC 364-4-41
  (as protective extra low voltage) and/or
- VDE 0805 / EN 60950 / IEC 950
  (as safety low voltage SELV) or
- VDE 0106 Part 101.

4 digital inputs (I0 to I3)

You can connect bounce-free switches (24 V current sourcing) or non-contact sensors (2- or 3-wire proximity switches) to the 4 digital inputs.

The digital inputs are not monitored for short circuits or wire break and have a non-isolated connection to the module chassis.

13 digital outputs (Q0 to Q12)

The state (on/off) of tracks 0 to 12 is output at 13 digital outputs. The digital outputs have a non-isolated connection to the module chassis.

Loads supported:

- Operating voltage 24 V
- Current load 0.5 A / short-circuit proof

The state of each output can be read off from the respective LED.

2.1.3.2 Cam control functions (S7-300, S7-400)

Influence of functions on the switching characteristics of time-based cams (S7-300, S7-400)

Actual value changes

A time-based cam can be skipped by the following functions that change the actual value:

- Set actual value
- Set actual value on-the-fly
- Zero offset
- Retrigger reference point

Activating a time-based cam

If you skip the start of a time-based cam with one of the above functions, this cam will be activated provided the actual direction of the axis motion is the same as the effective direction assigned for the cam. The programmed cam activation time starts.

Note

If the axis is at a standstill, the direction of movement is influenced by fluctuations of the actual value.

Set a hysteresis higher than the fluctuation of the actual value signal in order to suppress flutter when the axis is at a standstill.

This retains the last determined direction of movement while the axis is at a standstill.

WARNING

Risk of injury and material damage.

Functions which modify the actual value at rotary axes may cause the unwanted activation of time-based cams.

You should always set the "invalid" option for the time-based cams of a rotary axis if you want to influence the actual value using the functions mentioned above.

Set Actual Value/Set Actual Value on-the-fly/Cancel Set Actual Value (S7-300, S7-400)

Definition

Use the "Set process value" and "Set process value on-the-fly" functions to assign a new coordinate to the current encoder value. This shifts the coordinate system by the value: ACT_{new} - $ACT_{current}$

Whereby:

- ACT_{new} is the default value
- $ACT_{current}$ is the process value at the time of execution

Determining new coordinates

Calculate all your default positions in the shifted coordinate system based on the following formula:

$$Coordinate_{new} = Coordinate_{old} + (ACT_{new} - ACT_{current})$$

Requirements

- The axis must be synchronized.
- With "set process value on-the-fly": Digital input I1 must be interconnected.

Function sequence

1. Enter the coordinate for the process value, or for the process value to set on-the-fly in the channel DB.
   - Linear axis:
     You must select the specified process value so that the software limit switches are still within the valid traversing range after the function is called.
     The absolute value of the offset derived from $(ACT_{new} - ACT_{current})$ must be less than or equal to the absolute value of the permitted traversing range (maximum 100 m or 1000 m).
   - Rotary axis:
     Rule for the defined process value:
     0 ≤ process value < end of rotary axis
2. Set the appropriate trigger bits at the channel DB.
3. Call the CAM_CTRL instruction.
   "Set process values" is executed immediately.
   "Set process value on-the-fly" is executed at the next positive edge at digital input I1. The FVAL_DONE bit is set.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.4	AVAL_EN	BOOL	FALSE	1 = set process value
36.5	FVAL_EN	BOOL	FALSE	1 = set process value on-the-fly
90.0	AVAL	DINT	L#0	Process value coordinate
94.0	FVAL	DINT	L#0	Coordinate for on-the-fly process value
25.5	FVAL_DONE	BOOL	FALSE	1 = set process value on-the-fly executed

Effects of the function

The example "set process value" to 400 mm (at position 200 mm) demonstrates how this function shifts the coordinate system. Resultant effects:

- The position of the working range is **not** physically shifted.
- The individual points (such as the software limit switches) are assigned new coordinate values.
- The cams retain their coordinate values, and are therefore located at a different physical position.
- If the axis is synchronized and cam processing is enabled, the positional process value may skip cam edges or entire cams as a result of this function.
- Status changes of the cam which would normally trigger an interrupt may be lost.

Note

Make allowances for the switching characteristics of time-based cams (Page 460).

The table below shows how "Set process value" / "Set process value on-the-fly" shifts the coordinate system.

Set process value	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
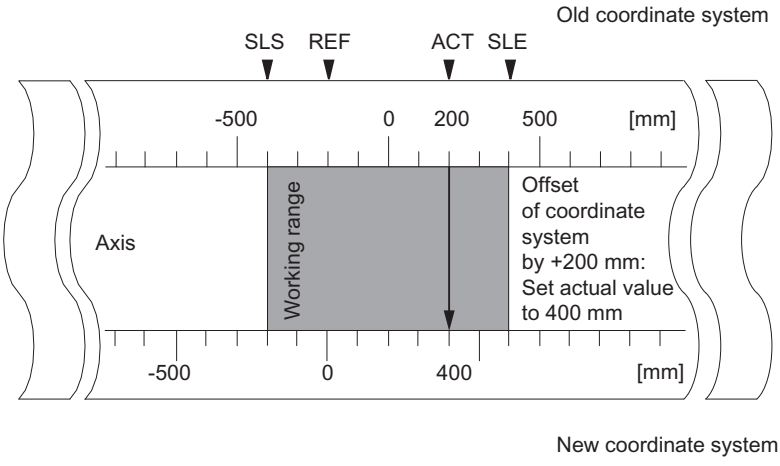	-400	-200	200	400
	-200	0	400	600

Cancelling the function

The "Undo set process value" function can be used to reset the coordinate shift caused by "Set process value" or "Set process value on-the-fly".

Once "Set process value on-the-fly" has been triggered, it can no longer be deleted before execution by a positive edge at input I1. However, it can be overwritten by a new "Set process value on-the-fly".

These functions are reset when a module is started up.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.2	AVALREM_EN	BOOL	FALSE	1: Cancel process value setting

Possible causes of error

"Set process value on-the-fly" and "Retrigger reference point" may not be executed in parallel.

The "Set actual value on-the-fly" function can signal an operator error if the function would cause a software limit switch to exceed the valid traversing range with a positive edge at I1. This operating error is signaled by a diagnostic interrupt, and written to the diagnostic buffer.

Set zero offset (S7-300, S7-400)

Definition

The "Zero offset" function lets you shift the zero point in the coordinate system by a defined value. The sign determines the offset direction.

Determining a new coordinate

All values of the shifted coordinate system can be calculated according to the following formula:

$\text{Coordinate}_{new} = \text{Coordinate}_{old} - (\text{ZPO}_{new} - \text{ZPO}_{old})$

ZPO_{old} identifies any existing zero offset. If no zero offset was active prior to the call, set a 0 value at ZPO_{old}.

This equation allows you to calculate the coordinates for the software limit switches, for example.

Function sequence

1. Write the zero offset value to the channel DB.
   - Linear axis:
     The zero offset must be selected so that the software limit switches remain within the permissible numerical range after the function is called.
   - Rotary axis:
     Rule for the zero offset:
     value of zero offset ≤ end of the rotary axis.
2. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.6	ZOFF_EN	BOOL	FALSE	1 = set zero offset
86.0	ZOFF	DINT	L#0	Zero offset

Effects on a linear axis

The example of a zero offset of -200 mm demonstrates that this function shifts the coordinate system in positive direction. Resultant effects:

- The working range is **not** physically shifted.
- The individual points (such as the software limit switches) are assigned new coordinate values.
- The cams retain their coordinate values, and are therefore located at a different physical position.
- If the axis is synchronized and cam processing is enabled, the positional process value may skip cam edges or entire cams as a result of this function.
- Status changes of the cam which would normally trigger an interrupt may be lost.

Note

Make allowances for the switching characteristics of time-based cams (Page 460).

The following table shows how the coordinate system is offset by "Zero offset".

Zero offset	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
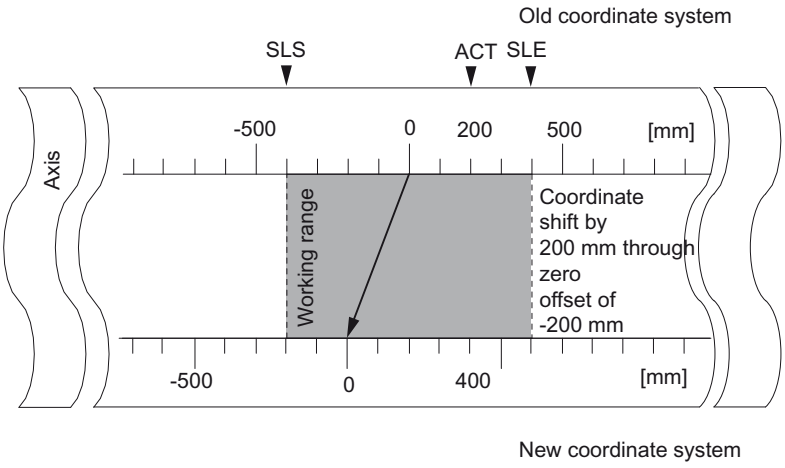	-400	-200	200	400
	-200	0	400	600

Effects on a rotary axis

The example of a -45° zero offset demonstrates how this function rotates the coordinate system.

The following table shows how the coordinate system is rotated by "Zero offset".

Tool when ACT = 340°	Tool when ACT = 25°
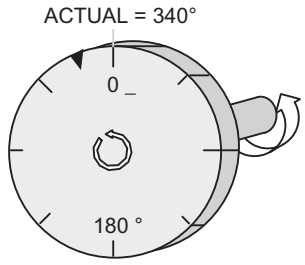	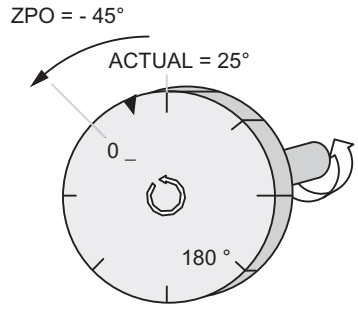
The zero mark rotates by -45 degrees. All axis points are assigned new coordinate values.	

Including a **ZPO_{old} = 0**, the result is a new value of 385°

As process value resumes at 0 at the end of the rotary axis operating in positive directional rotation, the calculation returns an process value of 25°:

$Coordinate_{new}$ = $Coordinate_{old}$ - (ZPO_{new} - ZPO_{old}) - end of rotary axis

The **end of rotary axis** value only needs to be subtracted when

$Coordinate_{old}$ - (**ZPO_{new} - NPV_{old}**) is greater than the end of rotary axis.

Loss of synchronization

If synchronization is lost due to an error, or reset due to "retrigger reference point", a zero offset is **retained**.

Cancelling function

A zero offset of 0 resets any existing zero offset.

Set reference point (S7-300, S7-400)

Definition

The "set reference point" function is used to synchronize the axis. The function shifts the working area. All shifts generated by a zero offset or "set actual value" are retained.

The function projects the operating range on the axis. Different value settings will thus move the working range to different positions within the physical range of the axis.

Requirements

Cam processing must be disabled.

Function sequence

1. Write the value for the reference point coordinate to the channel DB.
   - Linear axis:
     The reference point coordinates must not lie outside the software limit switches. This also applies to the reference point coordinate in a shifted coordinate system.
   - Rotary axis:
     Rule for the reference point coordinate:
     0 ≤ reference point coordinate < end of rotary axis
2. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.3	REFPT_EN	BOOL	FALSE	1 = set reference point coordinate
98.0	REFPT	DINT	L#0	Reference point coordinate
25.0	SYNC	BOOL	FALSE	1 = axis synchronized

Effects of the function

The example "Set reference point" to 300 mm demonstrates how this function shifts the axis operating range.

This has the following effects:

- The actual position is set to the value of the reference point coordinate.
- The working range is physically shifted on the axis.
- The various points retain their original coordinates, but are now at new physical positions.
- The SYNC bit is set in the checkback signals.

The following table shows how "Set reference point" shifts the operating range on the axis.

Set reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
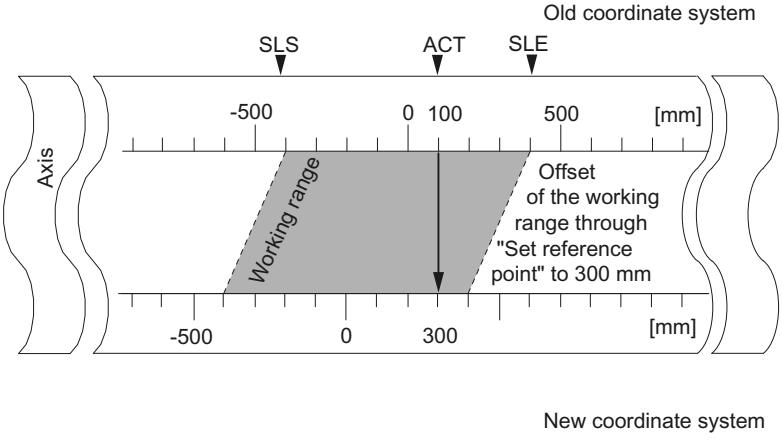	-400	-200	100	400
	-400	-200	300	400

Special features of the absolute encoder

This function is necessary for absolute encoder adjustment (Page 488).

Change cam edges (S7-300, S7-400)

Definition

The "change cam edges" function can be used to change the cam start and, for position-based cams, the end of a specific position-based cam in runtime.

Requirements

The cam you want to change must be valid.

Function sequence

1. Enter the cam number in the channel DB.
2. At a position-based cam:
   Enter the cam start and cam end in the channel DB.
   For a time-based cam:
   Enter the cam start value in the channel DB.
3. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.7	CH01CAM_EN	BOOL	FALSE	1 = write function for cam edges (1 cam)
102.0	CAM_NO	INT	0	Cam number
104.0	CAM_START	DINT	L#0	Cam start
108.0	CAM_END	DINT	L#0	Cam end

Effects of the function

The FM x52 shifts first the on edge and then the off edge of the cam. This sequence does not depend on the direction in which the cam is shifted.

Exception:

The sequence described above may briefly generate an inverse cam (Page 439) if the new cam start is greater than the old cam end.

The figure below shows the each stage in cam edge change.

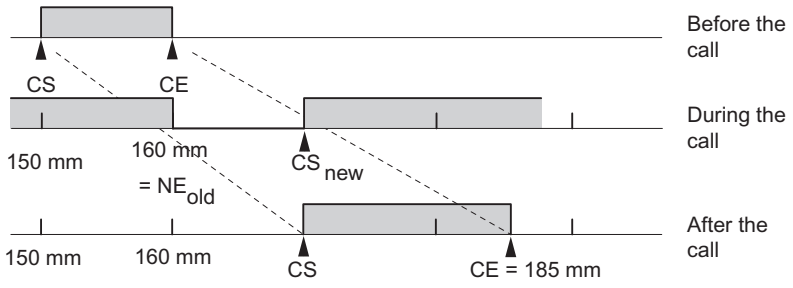

Note

If a hardware interrupt has been enabled for this cam, FM x52 can trigger one or two hardware interrupts when it detects the inverse cam depending on parameter assignment.

Changing the on and/or off edge can result in a cam edge or the entire cam being skipped.

Make allowances for the switching characteristics of time-based cams (Page 460).

Cam status changes that would normally trigger a hardware interrupt can be lost.

Reading modified values

You can read out the modified values by calling one of the jobs CAM1RD_EN to CAM8RD_EN.

Cancelling the function

The modified values are lost when you restart the module.

Perform "Fast Cam Parameter Change" (S7-300, S7-400)

Definition

The "Fast cam parameter change" function can be used to modify a group of up to 16 cams in runtime.

Requirements

The cams you want to modify must be valid.

Function sequence

1. Enter the number of cams to be modified in the channel DB.
2. Enter the number of the first cam to be modified in the channel DB.
3. Set the trigger bits for the required modifications.
4. Declare the new values at the channel DB.

5. Repeat steps 2 to 4 for each cam to be modified.
6. Set the relevant trigger bit in the channel DB.

Data used in the channel DB

Address Absolute	Name	Type	Initial value	Comment
37.0	CH16CAM_EN	BOOL	FALSE	1 = write fast cam parameter change function (16 cams)
176.0	C_QTY	BYTE	B#16#0	Number of cams to modify
177.0	DIS_CHECK	BOOL	FALSE	1 = disable data check

Address relative	Name	Type	Initial value	Comment
+0.0	CAM_NO	BYTE	B#16#0	Number of the cam to modify
+1.0	C_EFFDIR	BOOL	FALSE	1 = change the effective direction of the cam
+1.1	C_CBEGIN	BOOL	FALSE	1 = change the cam start to the value CBEGIN
+1.2	C_CEND	BOOL	FALSE	1 = change the cam end / activation time to the value CEND
+1.3	C_LTIME	BOOL	FALSE	1 = change the lead time to the value LTIME
+1.4	CAM_OFF	BOOL	FALSE	1 = deactivate the cam during the cam modification
+1.5	EFFDIR_P	BOOL	FALSE	1 = positive effective direction (plus)
+1.6	EFFDIR_M	BOOL	FALSE	1 = negative effective direction (minus)
+2.0	CBEGIN	DINT	L#0	New cam start
+6.0	CEND	DINT	L#0	New cam end / new activation time
+10.0	LTIME	INT	L#0	New lead time

Deactivating a cam when making modifications

To maintain consistency, always deactivate the cam (CAM_OFF) when modifying its start and end settings.

Data check by the module

Use the DIS_CHECK (channel DB) parameter to specify whether or not to disable the validation of transferred data by FM x52. If you disable the data check, you must take steps to verify that only permitted values are transferred. If you transfer illegal values without a check, the module can behave unexpectedly.

- FALSE: The module checks all data to be transferred
- TRUE: The check of cam parameter data is disabled. This allows for faster activation of the data to be changed on FM x52.
  Regardless of this function, the module always checks whether
  - the axis is assigned
  - the number of cams to be changed (C_QTY) is valid
  - the cam (cam number) to change is valid.

The data are only enabled on the module after having been checked and found faultless.

Any faulty data are rejected.

Effects of the function

Note

Make allowances for the switching characteristics of time-based cams (Page 460).

Reading modified values

You can read the modified values by calling one of the jobs CAM1RD_EN to CAM8RD_EN.

Cancelling the function

The modified values are lost when you restart the module.

Executing "Length measurement" and "Edge detection" (S7-300, S7-400)

Definition

You can establish the length of a workpiece using the "Length measurement" and "Edge detection" functions.

Length measurement and edge detection are active and remain active until you disable these functions or select a different measuring method. The CAM_CTRL instruction enables length measurement if you select both measuring methods in parallel.

Requirement

A **bounce-free** switch must be connected to input I1.

Function sequence

The FM 352 updates the data on the module at different times depending on the type of measurement. FM 352 reports each update at a parameter on the feedback interface.

Length measurement

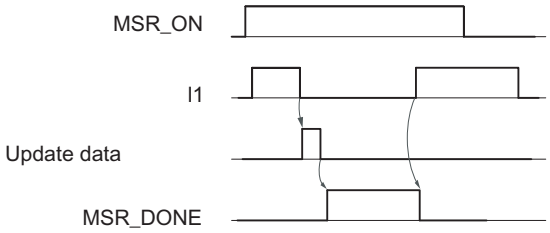

1. Set the function switch for "length measurement".
2. A positive edge at input I1 starts the length measurement.
3. The negative edge at input I1 stops the current measurement. The FM 352 updates the initial value, final value and length data.
4. When parameter MSR_DONE is set, FM 352 reports the data update. The parameter indicates completion of the measurement. The results of the measurement can be read out.
5. The start of the next measurement with a rising edge at I1 resets the MSR_DONE parameter.

The FM 352 does not update the data if the function is disabled during length measurement. The MSR_DONE parameter remains reset.

Edge detection

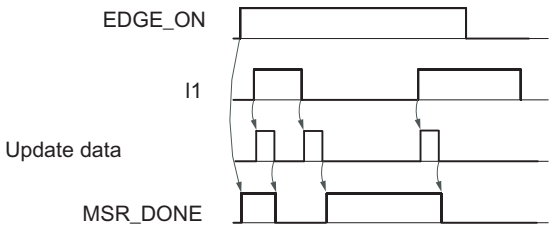

1. Enter a value for the minimum edge distance in the parameter DB. Write and enable machine data.
2. Set the "edge detection" function switch. This sets the MSR_DONE parameter.
3. The positive edge at input I1 starts edge detection. The results of the measurement are updated and can be read out, and the start value of the measurement is entered; the end value and length assume the value -1.
4. After the update, FM 352 reports the change by resetting the MSR_DONE parameter.
5. The negative edge at input I1 stops the current measurement. The FM 352 updates the data for the end measurement and length values.

6. After the update, FM 352 reports the change by setting the MSR_DONE parameter. The results of the measurement can be read out.
7. The start of the next measurement with a rising edge at I1 resets the MSR_DONE parameter.

The FM 352 does not update the data if the function is disabled during edge detection. The MSR_DONE parameter remains reset.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
25.1	MSR_DONE	BOOL	FALSE	1 = length measurement completed
34.0	EDGE_ON	BOOL	FALSE	1 = edge detection on
34.2	MSR_ON	BOOL	FALSE	1 = length measurement on
38.2	MSRRD_EN	BOOL	FALSE	1 = read measured values
112.0	BEG_VAL	DINT	L#0	Initial value
116.0	END_VAL	DINT	L#0	End value
120.0	LEN_VAL	DINT	L#0	Length

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	**Minimum edge distance for edge detection** Valid value range: • 0 ... 1 000 000 000 µm

The minimum edge distance is used to define a range after the start of a measurement is detected with an edge detection. The measurement will be discarded if the measuring operation ends within this range.

The start of the measurement is not signaled unless the minimum edge distance has been traveled.

Conditions of length measurement

- The CPU program requires an interval of sufficient length between the off and on edges at input I1 in order to be able to evaluate the result of the measurement before a new measurement is triggered.
- The minimum interval between the positive and negative edges at input I1 and between the negative edge and the next positive edge at input I1 must be greater than 2 ms.

Faulty measurement

The FM 352 returns the length value -1 in the event of errors in length measurement / edge detection.

A "length measurement" or "edge detection" may only perform up to 126 zero passes in one direction. The zero pass represents the rotary axis transition from the end of rotary axis value

to 0 and vice versa. The FM 352 reports incorrect "length measurement" or "edge detection" if it detects more than 126 zero passes in one direction, regardless of whether or not it then detects any zero passes in the opposite direction.

A length measurement is also considered faulty if:

- the length measured at a rotary axis is greater than 2^{31}.
- the on and off edges are detected simultaneously by the FM 352 (for example due to switch bounce).

This event can be signaled as a hardware interrupt.

Shift of the coordinate system during a length measurement

Conditions under which a shift of the coordinate system will influence the measured length:

- You are using an incremental encoder or an initiator or are operating the FM 352 in simulation mode.
- You are executing a "set reference point" or "retrigger reference point" function while a length measurement is active.

Example

To utilize the above mentioned influences on the measured length:

Your system always develops slip when you perform a length measurement.

The "Retrigger reference point" function can be used to correct this slip to allow correct length measurement values to be output.

Retrigger reference point (S7-300, S7-400)

Definition

The "Retrigger reference point" function can be used to synchronize the axis as a reaction to a recurring external event.

The function remains active until you deactivate it.

Requirements

- You are using an incremental encoder or an initiator.
- The external event may represent the zero mark signal of an incremental encoder or reference point switch at input I2.

Function sequence

1. Set the value for the reference point coordinate at the parameter DB.
2. Set the type of "retrigger reference point" at the parameter DB.
   Options:
   – Evaluation of the zero mark of the encoder only (RETR_TYPE = 7).
   – Evaluation of the reference point switch only (RETR_TYPE = 6).
   – Evaluation of the zero mark signal
     in positive direction: Evaluation of the first rising edge of the zero mark after passing the reference point switch in positive direction (RETR_TYPE = 0).
     in negative direction: Evaluation of the first falling edge of the zero mark after passing the reference point switch in negative direction (RETR_TYPE = 1).
3. Write and enable machine data.
4. Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.3	REFTR_ON	BOOL	FALSE	1 = retrigger reference point
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	Reference point coordinate
52.0	RETR_TYPE	DINT	L#0	Reference point retrigger mode

Effects of the function

- FM x52 evaluates the zero mark and reference point switch in line with the axis direction of motion.
  – It evaluates the positive edges when the axis moves in positive direction.
  – It evaluates the negative edges when the axis moves in negative direction.
- It sets the actual position to the value of the reference point coordinate.
- The working range is physically shifted on the axis.
- The various points retain their original values, but are now at new physical positions.
- Cam status changes that would normally trigger a hardware interrupt can be lost.
- The SYNC bit in the checkback signals is set.

Note

Make allowances for the switching characteristics of time-based cams (Page 460).

Example

Rules for the example:

- The module evaluates the positive edges of the reference point switch and zero mark signals (axis moving in positive direction).
- Value of the reference point coordinate = 300 mm.
- No zero offset is active at the time of execution.

The following table shows how "Retrigger reference point" shifts the operating range on the axis.

Retrigger reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
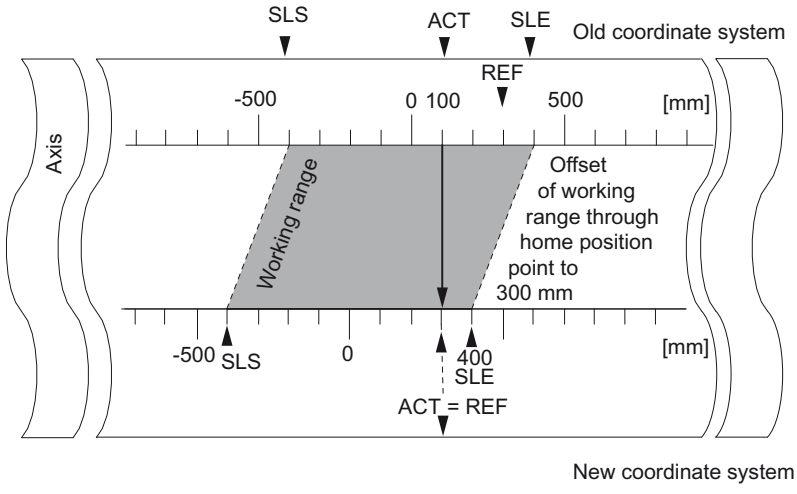	-400	300	100	400
	-400	300	300	400

Inclusion of a zero offset

Any active zero offset is included in the "Retrigger reference point" function. The reference point coordinate setting is thus calculated according to the formula:

Ref = Ref_{MD} - Zero offset

Ref_{MD} is the value of the reference point coordinate store in machine data.

The following table shows how "Retrigger reference point" with zero offset shifts the operating range on the axis.

Retrigger reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
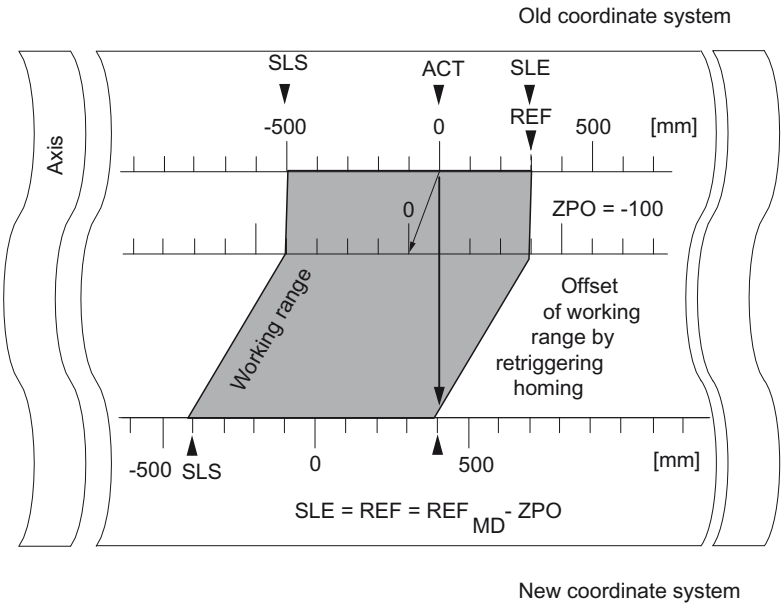	-500	300	0	300
	-400	400	100	400
	-400	400	400	400

Deactivate software limit switch (S7-300, S7-400)

Definition

Use the "Deactivate software limit switch" function to disable monitoring of the software limit switches at a linear axis.

The function remains active until you deactivate it. This re-activates the software limit switches originally parameterized.

Function sequence

Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.4	SSW_OFF	BOOL	FALSE	1 = Software limit switches deactivated

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L#-1000000000	Software limit switch start
68.0	SSW_END	DINT	L#1000000000	Software limit switch end

Effects of the function

- simulation
  - Simulation mode stops when the axis overruns a software limit switch.
  - You can resume simulation mode by disabling software limit switch monitoring. The axis moves in the defined direction.
- Zero offset when monitoring is disabled
  If a zero offset is set in which software limit switches are located within traversing range limits, the actual value may still be outside the valid number range.
- Cams located outside the parameterized software limit switches can be switched.

CAUTION

Risk of material damage!

If you reduce the traversing range by setting software limit switches to protect your system, you risk damage to your equipment when these switches are disabled.

In the planning and engineering phases for your plant, you should be certain that the drive can travel in the entire physical traversing range.

Executing "Simulation" (S7-300, S7-400)

Definition

The "Simulation" function allows you to activate the cam controller without connected encoders.

Function sequence

1. Set the simulation velocity at the parameter DB.
2. Write and enable machine data.
3. Set either a positive or negative simulation direction at the channel DB.
4. Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.2	DIR_M	BOOL	FALSE	1 = simulation in negative direction
15.3	DIR_P	BOOL	FALSE	1 = simulation in positive direction
34.1	SIM_ON	BOOL	FALSE	1 = simulation on

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	SIM_SPD	DINT	L#0	Simulation speed

Effects when simulation mode is activated

- Encoder signals will be ignored.
- The encoder input monitoring functions are disabled.
- All encoder errors reported will be reset.
- FM x52 simulates axis motion at a constant simulation velocity.
- Cam processing is disabled while simulation mode is active. However, you can then enable these operations again. Synchronism will be retained in this case.
- Starting at the current actual value, the actual position value changes dynamically based on the simulation velocity and direction.

Effects when simulation mode is deactivated

- Cam processing will be disabled.
- The synchronization of an incremental encoder or initiator will be cleared.. The actual value will be reset to the value of the reference point coordinate.
- The module reports the actual position value which corresponds with the absolute encoder value. The module then evaluates the encoder signals as defined at the machine parameters.

Limit values

The minimum and maximum simulation velocity depend on the resolution (Page 493).

Velocity

There may be differences between the set velocity and the actual velocity (Page 493) at which the module operates.

Read "count values of counter cam tracks" (S7-300, S7-400)

Definition

The "Counter cam track counter values" function is used to read out the current counter values.

Function sequence

1. Specify the counter cam tracks and the high count values in the machine data.
2. Write and enable machine data.
3. Enable the count function.
4. The module sets the count value to its high limit.
5. The count value decrements by the count of 1 at each positive edge of the track result signal.
6. Set the trigger bit at the channel DB to read the count values.
7. The module writes both count values to the channel DB. The module outputs a 0 value for tracks not assigned as a counter cam track.
8. The module sets track identifier bit = 1 at the counter cam track when the count value = 0.
9. It resets the track identifier bit = 0 at the next negative edge of the track result signal, and resets the counter to its high limit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.5	CNTC0_EN	BOOL	FALSE	1 = enable count function of counter cam track 0
15.6	CNTC1_EN	BOOL	FALSE	1 = enable count function of counter cam track 1
38.3	CNTTRC_EN	BOOL	FALSE	1 = read count values of counter cam tracks
124.0	CNT_TRC0	INT	0	Current count value of counter cam track 0
126.0	CNT_TRC1	INT	0	Current count value of counter cam track 1

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
99.0	SPEC_TRC0	BOOL	FALSE	1 = track 0 is counter cam track
99.1	SPEC_TRC1	BOOL	FALSE	1 = track 1 is counter cam track
100.0	CNT_LIM0	DINT	L#2	High count value of counter cam track 0
104.0	CNT_LIM1	DINT	L#2	High count value of counter cam track 1

Read out "Position and track data" (S7-300, S7-400)

Definition

The "Position and track data" function can be used to read the current position, the velocity and the track identifier bits. The track identifier bits are recorded before being logically linked to machine and channel data.

The algorithm implemented in the FM x52 calculates velocity changes greater than 1 pulse/ 4 ms. The indicated velocity includes this inaccuracy, and is thus unsuitable in particular for closed-loop control. The internal velocity value used for dynamic cam control offers a higher precision.

Function sequence

1. Set the trigger bit in the channel DB.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
38.4	ACTPOS_EN	BOOL	FALSE	1 = read position and track data
128.0	ACTPOS	DINT	L#0	Current position
132.0	ACTSPD	DINT	L#0	Current velocity
136.0	TRACK_ID	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31

Read out "Encoder data" (S7-300, S7-400)

Definition

The "Encoder data" function can be used to read out current encoder data and the value for absolute encoder adjustment.

Requirements

You can read out the value for absolute encoder adjustment (Page 488) after you have implemented the "Set reference point" (Page 465) function.

Function sequence

1. Set the trigger bit in the channel DB.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
38.5	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
140.0	ENCVAL	DINT	L#0	Encoder value / counter value (internal representation)
144.0	ZEROVAL	DINT	L#0	Counter value at the last zero mark (internal representation)
148.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment

Read out "Cam and track data" (S7-300, S7-400)

Definition

The "Cam and track data" setting can be used to read the current cam and track identifier bits and the position. The track identifier bits are recorded before being logically linked to machine and channel data.

Function sequence

1. Enter type ID = 1 at the FM_TYPE parameter of the channel DB. This allows you to read 24 bytes of cam and track data.
   If you enter type ID = 0, only the cam identifier bits (16 bytes) will be read.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
12.0	FM_TYPE	BOOL	FALSE	0 = FM 352 to V4.0 1 = FM 352 V5.0 or higher and FM 452
38.6	CAMOUT_EN	BOOL	FALSE	1 = read cam and track data
152.0	CAM_00_31	DWORD	DW#16#0	Cam identifier bits for cams 0 to 31
156.0	CAM_32_63	DWORD	DW#16#0	Cam identifier bits for cams 32 to 63
160.0	CAM_64_95	DWORD	DW#16#0	Cam identifier bits for cams 64 to 95
164.0	CAM_96_127	DWORD	DW#16#0	Cam identifier bits for cams 96 to 127
168.0	TRACK_ID1	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31
172.0	ACTPOS1	DINT	L#0	Current position

Setting "Control signals for the cam controller" (S7-300, S7-400)

Definition

The "Control signals for the cam controller" function can be used to enable cam processing and the tracks.

Function sequence

1. Set the required bits at the channel DB.
2. The data are transferred to the module each time the CAM_CTRL instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.4	CAM_EN	BOOL	FALSE	1 = enable cam processing
16.0	TRACK_EN	WORD	W#16#0	Enable cam tracks 0 to 12 Bit 0 = track 0

Effects

Cam processing will be started or stopped based on the enable status.

The track identifier bits of enabled tracks are transferred to the track signals and digital outputs.

Querying "Checkback signals for the cam controller" (S7-300, S7-400)

Definition

The "Checkback signals for the cam controller" function provides information about the current status of the cam controller and track signals. Consistency between the reported position and track signals is not guaranteed.

Function sequence

The data is stored in the channel DB each time the CAM_CTRL instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
23.4	CAM_ACT	BOOL	FALSE	1 = cam processing busy
26.0	ACT_POS	DINT	L#0	Current axis position
30.0	TRACK_OUT	DWORD	DW#16#0	Current track signals of tracks 0 to 31 Bit 0 = track 0

Querying the "Checkback signals for diagnostics" (S7-300, S7-400)

Definition

The "Checkback signals for diagnostics" function is used to report diagnostics events.

Function sequence

1. If the module enters a new event in the diagnostic buffer, it sets the DIAG bit in the feedback interface. Error (Page 524) events are logged in the diagnostics buffer.
2. The module sets the DATA_ERR bit in the feedback interface when it detects incorrect data in a write job. The cause of the error is logged to the diagnostic buffer.
3. The feedback interface stores the CAM_CTRL instruction in the channel DB.
4. The module resets the DIAG bit to 0 after the diagnostic buffer has been read by the CAM_DIAG instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
22.2	DIAG	BOOL	FALSE	1 = diagnostic buffer modified
22.4	DATA_ERR	BOOL	FALSE	1 = data error

2.1.3.3 Configuring FM 352 (S7-300, S7-400)

2.1.3.4 Configuring FM 352 (S7-300, S7-400)

Measuring system (S7-300, S7-400)

Selecting the system of units (S7-300, S7-400)

Selecting the system of units

For the input and output of the data, you can select from the following systems of units:

- Millimeter (default setting)
- Inches
- Degrees
- Pulses

Note

If you change the system of units in the "Measuring system" dialog, the values will be converted to the new system. This may lead to rounding errors.

If you change the system of units via the machine date, the values are not automatically converted.

Any change of the system of units from or to "pulses" disables cam processing and prevents axis synchronization.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
8.0	UNITS	DINT	L#1	**Measuring system** Valid values: • 1 = 10^{-3} mm • 2 = 10^{-4} inch • 3 = 10^{-4} degrees • 4 = 10^{-2} degrees • 5 = pulses • 6 = 10^{-3} degrees

Standard system of units in the online help

To specify the limits, we use the mm system of units. To define the limits in other systems of units, convert the values as shown below:

To convert....		you calculate...
mm → inch		Limit value (inches) = limit value (mm) × 0.1 [1)]
mm → degrees	10^{-4} (4 decimal places)	Limit value (degrees) = limit value (mm) x 0.1
	10^{-3} (3 decimal places)	Limit value (degrees) = limit value (mm) x 1
	10^{-2} (2 decimal places)	Limit value(degrees) = limit value (mm) x 10
mm → pulse		Limit value (pulses) = limit value (mm) x 1000

1) When set to the maximum value, the number of decimal places affects the number of digits before the decimal point. In the inch measurement system, 4 decimal places are used; you can therefore enter a maximum of 100,000.0000 inches. The millimeter measurement system uses 3 decimal places; a maximum of 1,000,000.000 mm can be entered.

Axis (S7-300, S7-400)

Specifying axis type (S7-300, S7-400)

Selecting axis type (S7-300, S7-400)

Axis types

The cam controller supports the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.

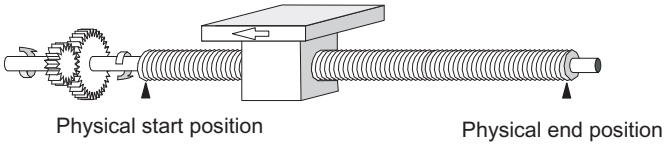

- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.

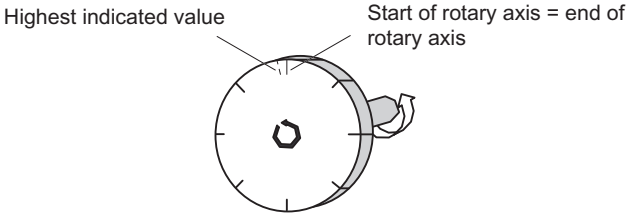

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
12.0	AXIS_TYPE	DINT	L#0	**Axis type** Valid values: • 0 = linear axis • 1 = Rotary axis

Specifying software limit switches (S7-300, S7-400)

Software limit switch

Software limit switches only exist for the linear axis.

The operating range is the range limited by the software limit switches. The FM x52 can monitor the operating range limits.

The software limit switch start (SLS) must always be smaller than the software limit switch end (SLE).

The software limit switches are active when FM x52 is synchronized.

- When using an incremental encoder or initiator:
  The axis is initially not synchronized after each FM x52 startup. The parameterized software limit switches are not monitored until the axis is synchronized.
- When using an absolute encoder (SSI):
  The FM x52 is synchronized once it has received a complete frame without errors. The software limit switches are monitored from this time on. The absolute encoder you use must cover at least the operating range including the software limit switches.

Relationship between the operating range, encoder range and traversing range

- The operating range is the range which you specify for your job using the software limit switches.
- The encoder range represents the range explicitly covered by the encoder. For a linear axis, the module assigns the encoder range symmetrically across across the operating range. In other words, the module shifts the encoder range in order to equalize the distances between the software limit switches and the ends of the encoder range (see figure below).
- This traversing range is the value range the FM x52 can process. The traversing range is dependent on the resolution.

The following applies: Traversing range ≥ encoder range ≥ operating range

The following figure shows the relationship between software limit switches, the operating range, the encoder range and the traversing range.

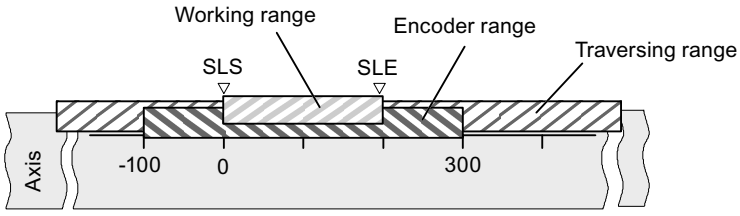

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L# -100 000 000	**Software limit switch start**
68.0	SSW_END	DINT	L# 100 000 000	**Software limit switch end** Valid value range: • -1,000,000,000 µm to 1,000,000,000 µm

Specifying end of rotary axis (S7-300, S7-400)

End of rotary axis

The "End of rotary axis" value is the theoretical maximum actual value of the axis. However, the theoretical maximum value is never indicated, because it also represents the physical start position of the rotary axis (= zero).

The highest rotary axis value displayed is:

End of rotary axis [µm] - resolution [µm / pulse] x 1 [pulse]

Example: End of rotary axis = 1000 mm

The displayed value jumps:

- from 999 mm to 0 mm when the rotational direction is positive.
- from 0 mm to 999 mm when rotational direction is negative.

Rotary axis with absolute encoders

The rotary range (0 to end of rotary axis) of a rotary axis with an absolute encoder must correspond exactly to the total number of steps of the absolute encoder:

$$\text{Rotary axis end[µm]} = \text{Number of revolutions(encoder)} * \frac{\text{steps(encoder)[pulse]}}{\text{Revolution}} * \text{RES}[\frac{\text{µm}}{\text{Pulse}}]$$

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
16.0	ENDROTAX	DINT	L#100000	**End of rotary axis** Valid value range: • 1 µm to +1,000,000,000 µm

Specifying synchronization (S7-300, S7-400)

Selecting "Retrigger reference point" type (S7-300, S7-400)

Retrigger reference point

By setting the "Retrigger reference point (Page 473)" type, you define the conditions for axis synchronization in operation with an incremental encoder or an initiator.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
52.0	RETR_TYPE	DINT	L#0	**Type of reference point retrigger** Valid values: • 0 = reference point switch and zero mark direction + • 1 = reference point switch and zero mark direction - • 6 = reference point switch only • 7 = zero mark only

Specifying reference point coordinate (S7-300, S7-400)

Reference point coordinate

- **Incremental encoder and initiator:**
  Using the "Retrigger reference point" function switch and a synchronization event defined by the "Retrigger reference point" type (Page 487), the reference point coordinate is assigned to this event.
- **Absolute encoder (SSI)**
  A parameterized axis with absolute encoder is always synchronized after transmission of the first error-free SSI frame.
  For more information, refer to the description of absolute encoder adjustment (Page 488) which explains the interaction of the absolute encoder adjustment with other data.

The value of the reference point coordinate must lie within the operating range:

- **Linear axis**
  Including the software limit switches.
- **Rotary axis**
  Greater than or equal to 0 and less than the "end of rotary axis" value (0 ≤ reference point coordinate < end of rotary axis).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1,000,000,000 µm to +1,000,000,000 µm

Specifying absolute encoder adjustment (S7-300, S7-400)

Determining the absolute encoder adjustment (S7-300, S7-400)

Definition

Absolute encoder adjustment is only necessary for an absolute encoder.

The absolute encoder adjustment and reference point coordinate map the encoder's range of values to the axis coordinate system. The encoder value corresponding to the reference point coordinate on the axis is determined. The value must be less than the total number of steps of the absolute encoder.

Additional information can be found under "Absolute encoder adjustment procedure (Page 489)" and in an example (Page 490).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
48.0	ENC_ADJ	DINT	L#0	**Absolute encoder adjustment** Valid value range: • 0 to (2^{25} - 1)

Data used in the channel DB

Address	Name	Type	Initial value	Comment
98.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1,000,000,000 µm to +1,000,000,000 µm

Absolute encoder adjustment procedure (S7-300, S7-400)

Determining the correct absolute encoder adjustment

After initial parameter assignment, further steps must be taken to ensure that a correct relationship is created between the encoder and coordinate system. The procedure shown in the next section is based on the use of the parameter assignment dialogs.

1. Move the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch end".
2. Call the "Set reference point (Page 465)" setting using the coordinate defined in point 1. FM x52 now determines an encoder value for the reference point coordinate entered in the channel DB (REFPT in the channel DB), namely the absolute encoder adjustment. You can read this value with the "Encoder data (Page 480)" setting.
3. Enter the value read in the "absolute encoder adjustment" field.
4. Save your parameter assignment to the corresponding parameter DB using the export function.
5. Download the data in the HWCN to the CPU.
6. Restart the CPU to activate the data.

Note

You perform this adjustment once during commissioning. Following parameter assignment, the FM x52 is synchronized during start-up as soon as a complete, faultless encoder frame is received after start-up.

Otherwise: Mechanical adjustment of the encoder

A correct relationship between the coordinate system and the encoder can be achieved as follows:

1. Traverse the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch start".
2. Enter this coordinate value in the machine data as "reference point coordinate".
3. Read the encoder value displayed at this position with the "Encoder data (Page 480)" setting.
4. Enter this value in the machine data as the "absolute encoder adjustment".

After the parameter assignment a correct actual value will always be shown.

Instead of steps 3 and 4, you can also set the encoder to zero using "Reset", if available. Then enter the value "0" in the machine data as the absolute encoder adjustment.

Example of absolute encoder adjustment (S7-300, S7-400)

Example of an absolute encoder adjustment

The following assumptions apply to the example:

- Reference point coordinate = -125 mm
- Operating range from SSW_STRT = -1000 mm to SSW_END = 1000 mm
- Absolute encoder adjustment = 0
- Encoder range = 2048 increments with a resolution of 1 mm/pulse
- The absolute encoder used cannot be adjusted mechanically with precision and it also is not possible to set the encoder value specifically.

The following figure shows an example of absolute encoder adjustment.

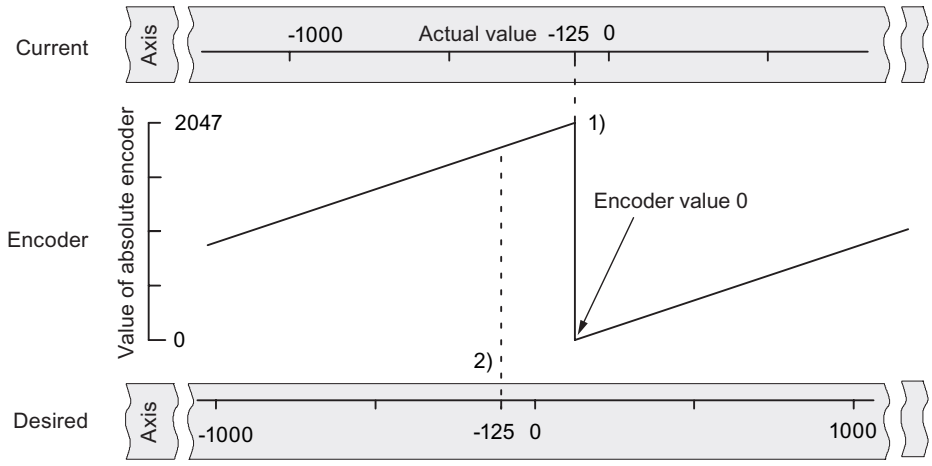

1) Assignment of the coordinate system to encoder values, based on the absolute encoder adjustment. The encoder value 0 corresponds to the actual value -125.

2) Required assignment of the coordinate system to the encoder. At this position the coordinate should be -125.

Result in accordance with "Set reference point"

After "Set reference point" the relationship looks as follows:

The reference point coordinate on the axis (-125) is assigned to the encoder value (1798) determined from absolute encoder adjustment. This assignment is set out in the figure below.

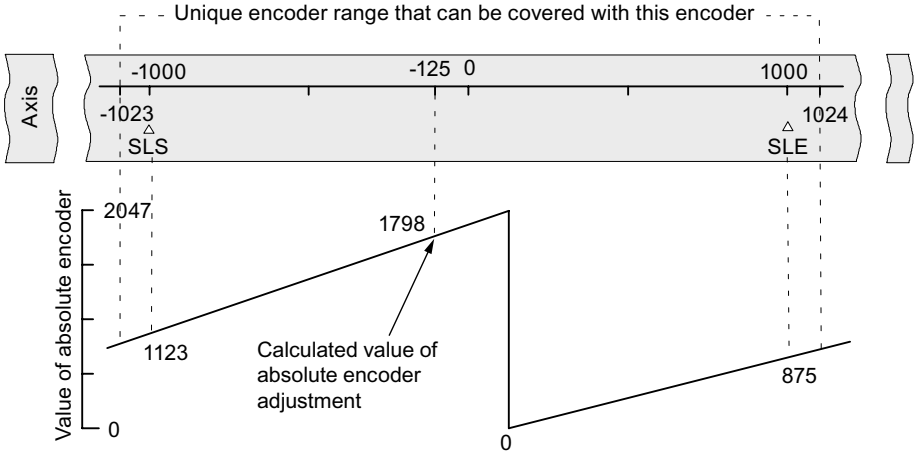

The encoder supplies 2048 unique values. The operating range is defined by the software limit switches. The selected resolution of 1 mm per pulse means the encoder can, however, cover a greater operating range than the one set by the software limit switches.

With the set resolution, the operating range is already covered with 2001 values. Therefore, in the example, 47 pulses "remain" and these are symmetrically positioned around the operating range.

Setting hysteresis (S7-300, S7-400)

Hysteresis

The hysteresis (Page 444) value range depends on the resolution (Page 497).

Maximum input value:

- For linear axes: max. input value < ¼ of the operating range
- For rotary axes: max. input value < ¼ of the rotary axis range

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
80.0	HYS	DINT	L#0	**Hysteresis** Valid value range: • 0 to 65 535 [Imp] x resolution [$^{\mu m}/_{Imp}$]

Position-based cams and hysteresis

A position-based cam is activated when the following conditions are met:

- The detected actual value lies within the position-based cam.
- Hysteresis is not active.

Switching points may vary depending on the position at which the change in direction takes place.

The figure below shows the switching characteristics of a position-based cam with hysteresis.

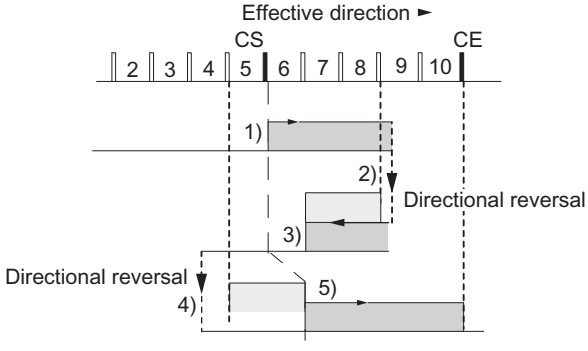

	Cam
	Hysteresis
1)	Cam is activated, as the cam start is crossed in the positive effective direction.
2)	Cam remains active due to hysteresis, irrespective of changes of direction.
3)	Cam is deactivated when hysteresis ends.
4)	Cam remains deactivated after a direction change.
5)	Cam is activated again at the end of hysteresis.

Note

When the direction is changed, hysteresis cannot reduce a cam actuation time shorter than the hysteresis.

Time-based cams and hysteresis

A time-based cam is activated when the following conditions are met:

- The cam start is crossed in the effective direction.
- Hysteresis is not active.

Note

The hysteresis will hide a time-based cam if its range between the reversal point and the cam start is less than the hysteresis.

The figure below shows the switching characteristics of a time-based cam with hysteresis. The figure shows a time-based cam that is **not** activated again.

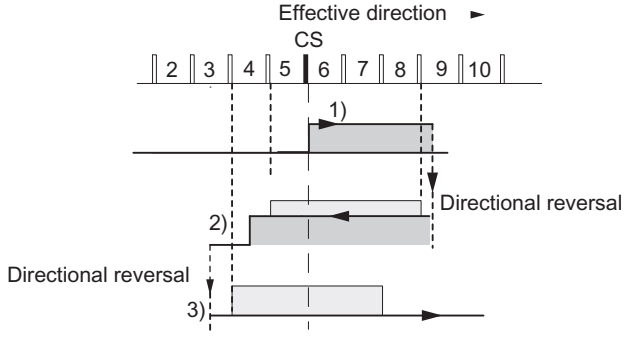

	Cam
	Hysteresis
1)	Cam is activated, as the cam start is crossed in the positive effective direction.
2)	Cam remains active after the change of direction until the activation time has expired, regardless of hysteresis.
3)	Cam start is hidden by the hysteresis; the cam is not activated.

Specifying simulation speed (S7-300, S7-400)

Simulation speed

This machine data determines the simulation speed. Please also see the information under "Influence of settings on the switching characteristics of time-based cams (Page 460)".

The actual simulation speed V_{sim} may deviate from the setting at $V_{sim, V}$, and is calculated using the following formula:

$$V_{Sim} = \frac{6*10exp7*RES}{\text{Integer}\left(\frac{6*10exp7*RES}{V_{Sim, V}}\right)}$$

In this equation:

- V_{sim}: Simulation speed set by FM 352. Unit: $^{\mu m}/_{min}$
- $V_{sim, V}$: simulation speed set in the machine parameters. Unit: $^{\mu m}/_{min}$
- RES: Resolution derived from encoder data. Unit: $^{\mu m}/_{pulse}$
- Integer ( ): Of this expression, only the decimal integer is used in subsequent calculation. This expression must lie within the range of 2 to 65536 in all calculations.

The actual simulation speed changes abruptly as a result of correlations (see above formula).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	SIM_SPD	DINT	L#0	**Simulation speed** Valid value range: • 0 = standstill • 5×10^8 = highest setting supported by the module Within this range, the simulation speed depends on the resolution (Page 497): 1000 x resolution ≤ simulation velocity ≤ 3 x 10^7 x resolution

Specifying minimum edge-to-edge distance (S7-300, S7-400)

Minimum edge-to-edge distance

This machine data is used to define a range after detection of the start of a measurement for edge detection. The measurement is discarded if the end of the measurement is within this range.

The start of measurement is not reported unless the "minimum edge-to-edge distance" has been traveled.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	**Minimum edge-to-edge distance** Valid value range: • 0 ... 1,000,000,000 µm

Encoder (S7-300, S7-400)

Selecting the signal type (S7-300, S7-400)

Signal type

The encoder returns position data to the module for evaluation and conversion to an process value based on the resolution.

You can select the signal type from one of the following encoders (Page 447):

- 5 V differential signal (Page 449)
  Select the signal type "5 V differential signal" to use a 5 V incremental encoder with differential signal A, /A, B, /B and N, /N.
- 24 V track A+B phase shifted (Page 449)
  Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.
- SSI absolute (Page 451)
  Select the "SSI absolute" signal type if you use an absolute encoder with a serial interface.
- 24 V initiator (Page 451)
  Select the signal type "24 V initiator" to use a 24 V initiator with count signal A*.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
20.0	ENC_TYPE	DINT	L#1	**Signal type** Valid values: • 1 = 5 V, incremental • 2 = 24 V, incremental • 3 = SSI 13-bit frame length • 4 = SSI 25-bit frame length • 5 = listen in • 6 = 24 V initiator up (positive) • 7 = 24 V initiator down (negative) • 8 = SSI 13-bit (right-justified) • 9 = SSI 25-bit (right-justified) • 10 = listen in (right-justified)

Specifying resolution (S7-300, S7-400)

Specify distance per encoder revolution (S7-300, S7-400)

Distance per encoder revolution

You inform the FM x52 about the distance covered by the drive system per encoder revolution using the "Distance per encoder revolution" machine data.

The "Distance per encoder revolution" value depends on the structure of the axis and the way in which the encoder is mounted. You must take all transmission elements, such as couplings and gears, into account. The following figure shows an example of two different encoder positions.

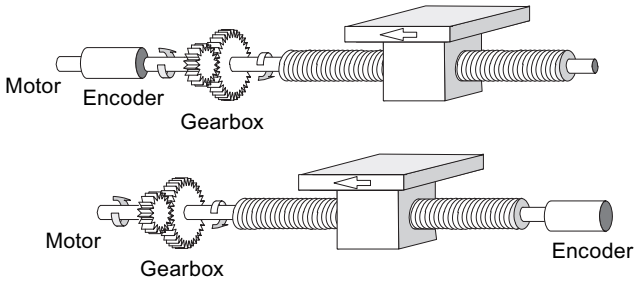

The relationship between the "Distance per encoder revolution" and "Increments per encoder revolution" machine data is described under resolution (Page 497).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
24.0	DISP_REV	DINT	L#80000	**Distance per encoder revolution** Valid value range: • 1 µm to 1,000,000,000 µm

Specifying increments per encoder revolution (S7-300, S7-400)

Increments per encoder revolution

The "Increments per encoder revolution" machine data defines the number of increments an encoder returns per revolution. The FM x52 calculates the resolution based on this value and the "Distance per encoder revolution" machine data.

Incremental encoder

Any value within the range of values can be entered (Page 449). The module evaluates the increments four times.

Initiator

Any value within the range of values is allowed.

Absolute encoder

Different encoder types have different limits:

Encoder type	Frame length / type	Value range	Can be used as linear axis
Single-turn encoder	13-bit half fir tree	64 ... 8192 in powers of 2	
Single-turn encoder	13-bit right-justified	64 ... 8192 all values	X
Single-turn encoder	25-bit right-justified	64 ... 2^{25} all values	X
Multiturn encoder	25-bit fir tree	64 ... 8192 in powers of 2	
Multiturn encoder	25-bit right-justified	64 ... 2^{24} all values	
Listen-in	Fir tree	64 ... 8192 in powers of 2	
Listen-in	Right justified	64 ... 2^{25} all values	X
Special setting: Multiturn encoder in single-turn mode	25-bit half fir tree	64 ... 8192 in powers of 2	

Note

The number of encoder pulses is the product of the "Increments per encoder revolution" and the "Number of revolutions" (see "Resolution (Page 497)").

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
32.0	INC_REV	DINT	L#500	**Increments per encoder revolution** Valid value range: • 1 to 2^{25} Note: This entry is irrelevant when you use "pulse" units.

Resolution (S7-300, S7-400)

Resolution

The resolution reflects the accuracy of cam processing. It also determines the maximum possible traversing range.

The resolution (RES) is calculated as shown in the table below:

	Incremental encoder	Absolute encoder / initiator
Input values	• Distance per encoder revolution • Increments per encoder revolution: – Pulse evaluation: quadruple – 1 increment = 4 pulses	• Distance per encoder revolution • Increments per encoder revolution. – 1 increment = 1 pulse
Calculation	$RES = \frac{\frac{Distance}{Encoder\ revolution}}{\frac{Pulses}{Encoder\ revolution}}$	

Note

Resolution in the "pulse" system of units is always 1.

All position values are rounded to the integer multiple of the resolution. This allows you to distinguish between the set and used values.

Range of values of the resolution

Convert the range of values for the resolution according to the defined physical units. The resolution must be kept within this range when setting the "distance per encoder revolution" and "increments per encoder revolution" values.

The chosen system of units determines the resolution value range:

Measuring system	Representation in...	Range of values of the resolution
mm	10^{-3} mm	0,1 × 10^{-3} mm to 1000 x 10^{-3} mm/pulse
inch	10^{-4} inch	0,1 × 10^{-4} inches to 1000 x 10^{-4} inches/pulse
degrees	10^{-4} degrees 10^{-3} degrees 10^{-2} degrees	0,1 × 10^{-4} degrees to 1000 x 10^{-4} degrees/pulse 0,1 × 10^{-3} degrees to 1000 x 10^{-3} degrees/pulse 0,1 × 10^{-2} degrees to 1000 x 10^{-2} degrees/pulse
Pulses	1 pulse	1

Example

- An incremental encoder has the following data:
  - Increments per encoder revolution: 5000
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 4 pulses

  Resultant resolution (quadruple evaluation):

$$\text{Resolution} = \frac{1000\ \text{mm}}{5000\ \text{increments}} = 0.2000\ \frac{\text{mm}}{\text{Increment}} = 0.2000\ \frac{\text{mm}}{4\ \text{pulses}}$$

$$= 0.0500\ \frac{\text{mm}}{\text{Pulse}}$$

- An SSI encoder has the following data:
  - Increments per encoder revolution: 4096
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 1 pulse

  Resultant resolution:

$$\text{Resolution} = \frac{1000\ \text{mm}}{4096\ \text{increments}} = 0.2441\ \frac{\text{mm}}{\text{Increment}} = 0.2441\ \frac{\text{mm}}{\text{Pulse}}$$

Traversing range dependency on resolution

The traversing range is restricted by the numerical representation in FM x52. This numerical representation varies depending on the resolution. You should thus make sure that your specifications are always within the permitted limits.

The table below sets out the maximum traversing range:

Resolution (RES) is in the range	Maximum traversing range
0.1 µm/pulse ≦ RES < 1 µm/pulse	-10^8 µm to 10^8 µm (-100 m to +100 m)
1 µm/pulse ≦ RES ≦ 1000 µm/pulse	-10^9 µm to 10^9 µm (-1000 m to +1000 m)

Dependency between the velocity and resolution

The velocity indicated may vary within the following limits, depending on the resolution (data relate to mm units):

- from 1 µm/min to 90 m/min, at a resolution < 1 µm/pulse
- from 1 µm/min to 900 m/min, at a resolution ≥ 1 µm/pulse

The velocity value is calculated and filtered by the module at intervals of 4 ms.

Its minimum inaccuracy of one pulse/4 ms rules it out for closed-loop control.

Selecting the count direction (S7-300, S7-400)

Count direction

You can adapt the position feedback detection to the direction of axis motion using the "Count direction" machine data. You must also take into account the rotational direction of all transmission elements (for example, couplings and gears).

- Normal = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent increasing positional actual values
- Inverted = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent decreasing positional actual values

Note:
You cannot implement a rate time in combination with an absolute encoder (SSI) and inverted parameterized count direction.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
59.0	CNT_DIR	BOOL	FALSE	Count direction Valid values: • 0 = normal • 1 = inverted

Selecting monitoring (S7-300, S7-400)

Monitoring

Wire break

When its monitoring function is enabled, the FM x52 monitors the A, /A, B, /B, N, and /N signals of an incremental encoder. The monitoring function detects:

- Wire break
- Short circuits in individual cables
  For incremental encoders without a zero mark, you must either:
  - Disable the wire-break monitoring function
  - Interconnect the N and /N signals externally (Page 447)
- Edge-to-edge distance of the count pulses
- Encoder supply failure

Frame error

You cannot switch off the monitoring of frame errors with absolute encoders (SSI). It monitors:

- the frame for start and stop bit errors
- the monoflop time of the connected encoder

Missing pulses (incremental encoder)

An incremental encoder must always return the same number of increments between two successive zero marks.

FM x52 checks whether the zero mark of an incremental encoder coincides with the correct encoder value.

Disable missing pulse monitoring for encoders without zero marks.

Also disable wire-break monitoring, or interconnect the zero mark inputs N and /N externally.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
				Monitoring functions
63.0	MON_WIRE	BOOL	TRUE	1 = wire break
63.1	MON_FRAME	BOOL	TRUE	1 = frame error (must always be 1)
63.2	MON_PULSE	BOOL	TRUE	1 = missing pulses

Selecting data for absolute encoder (SSI) (S7-300, S7-400)

Number of encoder revolutions (S7-300, S7-400)

Number of encoder revolutions

The machine data "Number of encoder revolutions" only applies to absolute encoders. You use it to define the maximum possible number of revolutions for this encoder. For additional information on this topic, please refer to "Absolute encoders (SSI) (Page 451)".

- Single-turn encoder
  Only the value 1 is possible.
- Multiturn encoder
  - Multiturn encoder / listen-in (fir tree): 2 ... 4096 in powers of 2
  - Multiturn encoder / listen-in (right-justified): 1 ... 4096 all values
- Linear scale
  You can also connect a linear scale. To do so, enter the value "1".

The total number of steps of the encoder is not machine data. It is calculated as follows:

Overall number of steps = increments per encoder revolution x number of revolutions

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
36.0	NO_REV	DINT	L#1024	**Number of encoder revolutions** Valid value range: • 1 (single-turn encoder) • 1 to 4096 (multiturn encoder)

Transmission rate (S7-300, S7-400)

Transmission rate

You define the data transmission rate between the SSI encoder and the FM x52 with the "transmission rate" machine data.

The maximum transmission rate depends on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 63 m → 500 kHz
- 20 m → 1,000 kHz

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
40.0	BAUDRATE	DINT	L#0	**Transmission rate** Valid values: • 0 = 125 kHz • 1 = 250 kHz • 2 = 500 kHz • 3 = 1000 kHz

Frame length (S7-300, S7-400)

Frame length

You use "Frame length" to define the clock frame output by the FM x52 for absolute encoders.

Selecting "listen-in" will disable the FM x52 cycle. The FM x52 can then listen in on other SSI frames with a 13 or 25-bit frame. The transmission rate is determined by the cycle clock rate of the master module.

Data used in the parameter DB

The frame length selected here is stored in the parameter DB under the ENC_TYPE (Page 495) date in the parameter DB.

Initiator direction (S7-300, S7-400)

Initiator direction

You define the direction of a connected initiator with "Initiator direction":

- Positive: = up (incremental count pulses)
- Negative: = down (downwards count pulses)

Tracks (S7-300, S7-400)

Selecting track output control mode (S7-300, S7-400)

Track output control

The "Track output control" machine data defines the control mode for track outputs 0 to 12. Tracks can be activated by:

- The cam controller: Track signals are activated and deactivated by FM 352 cam processing.
- CPU: The track signals represent the corresponding values of the track enables in the channel DB.

This allows the explicit activation of track outputs in the user program.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
90.0	TRACK_OUT	WORD	W#16#0	**Activation of track outputs** Valid values: • 0 = cam controller • 1 = CPU Bit number = track number Bits 13 to 15 must be 0.

Selecting special tracks (S7-300, S7-400)

Special tracks

You can parameterize tracks 0, 1 and 2 for use as special tracks.

- Track 0 and track 1 as counter cam track (Page 442)
- Track 2 as brake cam track (Page 442)

Data used in the parameter DB

Address	Name	Type	Start value	Comment
				Special tracks
99.0	SPEC_TRC0	BOOL	FALSE	1 = track 0 is counter cam track
99.1	SPEC_TRC1	BOOL	FALSE	1 = track 1 is counter cam track
99.2	SPEC_TRC2	BOOL	FALSE	1 = track 2 is brake cam track

Specifying high count value for counter cam track (S7-300, S7-400)

High count value for counter cam track

With this machine data, you define the high count value of the assigned counter cam track.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CNT_LIM0	DINT	L#2	**High count value of counter cam track** (track 0)
104.0	CNT_LIM1	DINT	L#2	**High count value of counter cam track** (track 1) Valid value range: • 2 … 65535

Selecting enable input (S7-300, S7-400)

Enable input

The track signal Q3 is activated when all the following conditions have been met:

- The track is enabled.
- The relevant external enable input I3 is set.
- The track result is 1.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
95.0	EN_IN_I3	BOOL	FALSE	Enable input 1 = track signal track 3 AND enable input I3 Bits 95.1 to 95.7 must be 0.

Cams (S7-300, S7-400)

Number of cams (S7-300, S7-400)

Number of cams

The number of cams determines the maximum number of programmable cams and therefore the cam cycle time.

Number of cams	Cam cycle time
16 cams	20.48 µs
32 cams	40.96 µs
64 cams	81.92 µs
128 cams	163.84 µs

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	C_QTY	DINT	UDT3: L#0 UDT4: L#1 UDT5: L#2 UDT6: L#3	Number of cams 0 = max. 16 cams 1 = max. 32 cams 2 = max. 64 cams 3 = max. 128 cams

Cam data (S7-300, S7-400)

Cam data (S7-300, S7-400)

Cam data

Cam data describe the properties of a cam, the assignment of each cam to a track, and the switching characteristic of the cam.

The cam data below is set separately for each cam:

- Validity (Page 506)
- Track number (Page 506)

- Cam type (Page 507)
- Cam start and cam end of position-based cams (Page 507)
- Cam start and activation time of time-based cams (Page 508)
- Lead time (Page 509)
- Effective direction (Page 511)
- Hardware interrupt (Page 513)

The following applies to cam processing in general:

- The module interprets and processes only the cams with "valid" settings.
- Cams 0 to 7 support hardware interrupts.
- The number of assignable cams depends on the number of cams available.

Cam validity (S7-300, S7-400)

Cam "valid"

All cams to be processed by the cam controller must be set as "valid".

Note

Unused cams should always be set to "invalid" CAMVALID = FALSE).

Data used in the parameter DB

Address relative	Name	Type	Initial value	Comment
+0.0	CAMVALID	BOOL	FALSE	1 = cam "valid"

Specifying a cam track number (S7-300, S7-400)

Track number

You define the track controlled by the cam by setting the track number.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+1.0	TRACK_NO	BYTE	B#16#0	**Track number** Valid value range: • 0 to 31

Selecting cam type (S7-300, S7-400)

Cam type

Select the required cam type.

- Position-based cam
- Time-based cam

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.3	CAM_TYPE	BOOL	FALSE	**Cam type** Valid values: • 0 = position-based cam • 1 = time-based cam

Defining cam start and cam end for position-based cams (S7-300, S7-400)

Cam start (CS) / Cam end (CE) of position-based cams

The cam start and cam end of a position-based cam define the cam length.

Minimum length of a position-based cam

The figure below shows the shortest cam with axis motion in positive direction

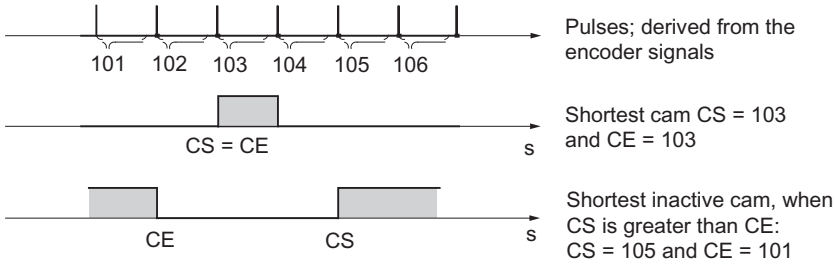

The following general rules apply:

- The inactive part of a cam must always have a minimum interval of 4 pulses between the cam end (CE) and cam start (CS).
- If CE = CS, the cam is activated for the duration of one pulse.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment Of position-based cams
+2.0 +6.0	CBEGIN CEND	DINT DINT	L#-100000000 L#100000000	**Cam start (CS)** **Cam end (CE)** Valid value range: • -1,000,000,000 µm to 1,000,000,000 µm

Specifying cam start and activation time for time-based cams (S7-300, S7-400)

Cam start (CS) / Cam end (CE) ≙ activation time of time-based cams

With a time-based cam, you must specify a cam start and an activation time in place of the cam end. Define the activation time with a resolution of 100 µs. The time runs starting with the activation of the cam.

Conditions of setting default times:

- 0 µs: A cam with an activation time of 0 µs will never be activated
- 0 µs < t ≤ 400 µs: The FM x52 sets a minimum cam activation time of approx. 330 µs.
- t > 400 µs: FM x52 calculates the actual activation time t_{act} based on the default activation time t_{def} using this formula:

$$t_{act} = \text{integer}\left(\frac{t_{def}}{\text{Cam cycle time}}\right) * \text{Cam cycle time}$$

The maximum error is always less than the cam cycle time.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment of time cams
+2.0 +6.0	CBEGIN CEND	DINT DINT	L#-100000000 L#100000000	**Cam start (CS)** **Cam end (CE) ≙ activation time** Valid cam start value range: • -1,000,000,000 µm to 1,000,000,000 µm Valid cam activation time value range: • (0 to 13421) x 100 µs with up to 16 cams • (0 to 26843) x 100 µs with up to 32 cams • (0 to 53686) x 100 µs with up to 64 cams • (0 to 65535) x 100 µs with up to 128 cams

Specifying lead time (S7-300, S7-400)

Lead time

You can compensate any delays caused by the connected switchgear by setting a lead time. Define a lead time with a resolution of 100 µs. You can assign one lead time to each cam. The lead time applies to the cam start and cam end.

Actuation distance

The actuation distance of a cam is calculated continuously based on the current velocity and lead time. The entire cam is shifted in direction of the actual value by this value. The assigned range is the "static range", and the range calculated based on the lead time represents the "dynamic range".

Actuation distance = lead time x current velocity

FM x52 calculates the actuation distance of all cams within ¼ of the longest parameterized lead time. A very high lead time setting for a cam reduces the frequency of the dynamic adjustment calculation.

Actual lead time

To calculate the actual lead time:

1. Determine the cam cycle time: This is the time in which the FM x52 fully processes all cams. It depends on the number of cams parameterized.
2. Calculate the actual lead time based on the following formula:

$$\text{Lead time}_{act} = \text{integer}\left(\frac{\text{Lead time}_{d}}{\text{Cam cycle time} * 4}\right) * \text{Cam cycle time} * 4$$

   The identifiers have the following meanings:

   – Lead time $_{act}$ is the lead time set by the FM x52
   – Lead time $_{d}$ is your default setting.
   – Integer ( ) means that the values after the decimal point are ignored in the calculation of the parenthesis.

The maximum error of the lead time$_{act}$ is always less than [cam cycle time] x 4.

Example:

The following values are defined:

- Number of cams: Maximum 32 cams
- Cam cycle time: 40.96 µs
- Lead time$_{d}$ = 1000 µs

Result: An actual lead time of 983 µs.

Note

The actual is always less than the assigned lead time. It may be 0 even though the parameterized lead time ≥100 µs.

The actuation distance of a rotary axis must be less than the rotary axis range and the inactive part of the cam. This must be guaranteed for all velocities.

Note

You may not implement a lead time in combination with an absolute encoder (SSI) and inverted parameterized count direction.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+ 10.0	LTIME	INT	0	**Lead time** Valid value range: • (0 to 53686) x 100 µs with max. 16 cams • (0 to 65535) × 100 µs with max. 32, 64 or 128 cams

Dynamic cam adjustment

There are two distinct situations relating to the cam range:

1. The static and dynamic range of the cam overlap.
2. The static and dynamic range of the cam do not overlap.

The table below shows the calculation of dynamic adjustment in each of the two cases.

Dynamic adjustment	Description
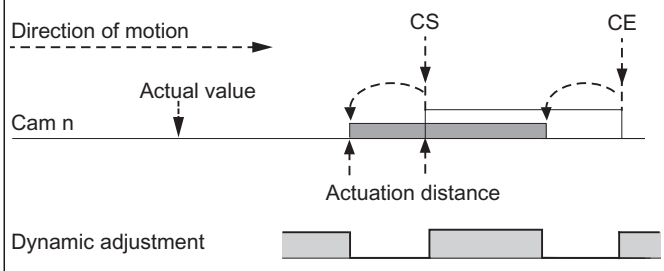	If the dynamic range of the cam overlaps its static range: • The cam is activated when its dynamic range is reached. At the same time, calculation of a new dynamic adjustment is disabled. • When the actual value reaches the static range of the cam, the calculation of a new dynamic adjustment is enabled again. A change in velocity affects the cam end. • When the cam is deactivated at the end of the dynamic range, dynamic adjustment will be disabled again until the end of the static range of the cam.
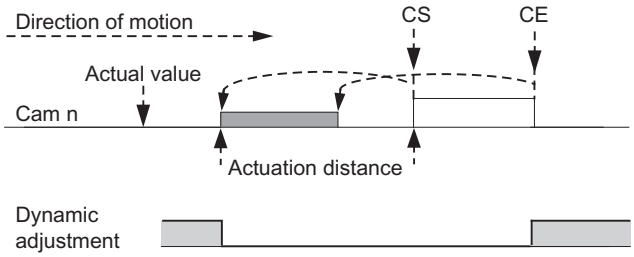	The following applies if the cam's dynamic and static ranges do not overlap: • The cam is activated when its dynamic range is reached. At the same time the calculation of a new dynamic adjustment is disabled. • At the end of the static range of the cam, dynamic adjustment is enabled again.
Dynamic range Static range A **new** dynamic adjustment is possible	

Note

When the direction of rotation changes, calculation of the dynamic adjustment is enabled again.

Selecting effective direction (S7-300, S7-400)

Effective direction

Two effective directions are supported:

- Positive: The cam is activated at the cam start if the axis is moving in the direction of increasing process values.
- Negative: The cam is activated at the cam end if the axis is moving in the direction of decreasing process values.

You may set both effective directions in parallel.

Selecting "None" disables the cam.

Switching characteristics of cams based on their effective direction

The table below shows the switching characteristics of position-based and time-based cams as determined by the effective direction. A positive effective direction is always assumed, with the exception of example 5.

No.	Description	Position-based cams	Time-based cams
1	A cam is overrun in the effective direction	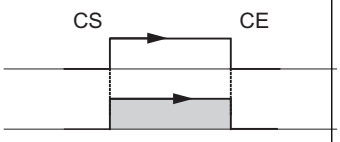	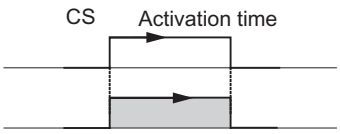
2	A cam is traversed in reversed effective direction	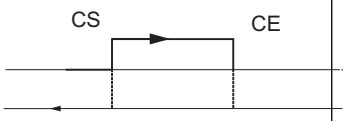	
3	A cam is approached in the effective direction; the direction of motion of the axis is reversed while the cam is active	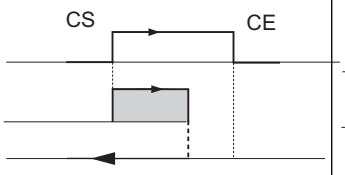	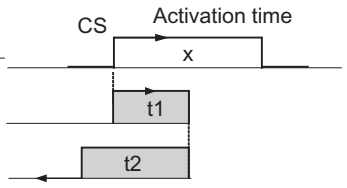 Activation time x = t1 + t2
4	A cam is approached in reversed effective direction; the axis reverses its direction of movement on the cam in effective direction	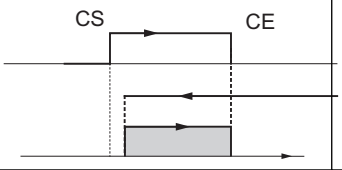	Cam not switched
5	A cam is approached from any direction, and exited into any direction; **both directions** are set as effective direction	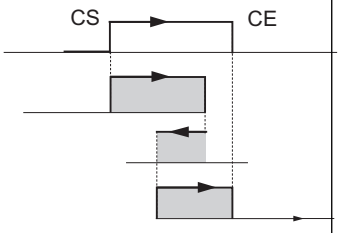	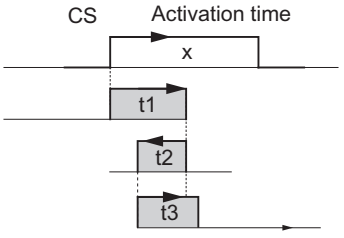 Activation time x = t1 + t2 + t3

parameterized cam
switched cam

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.1	EFFDIR_P	BOOL	TRUE	1 = positive effective direction (plus)
+0.2	EFFDIR_M	BOOL	TRUE	1 = negative effective direction (minus)

Selecting hardware interrupt for cam (S7-300, S7-400)

Hardware interrupt when cam switches on / off

You can specify whether hardware interrupts are to be generated when cams 0 to 7 are enabled, disabled or enabled and disabled.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.4	PI_SW_ON	BOOL	FALSE	1 = hardware interrupt on activation
+0.5	PI_SW_OFF	BOOL	FALSE	1 = hardware interrupt on deactivation

Enable hardware interrupts (S7-300, S7-400)

Enabling hardware interrupts (S7-300, S7-400)

Enabling hardware interrupts

This machine data is used to enable hardware interrupts.

You can set when hardware interrupts are to be generated:

- Cam on/off
  In the cam data (Page 513), you can specify whether hardware interrupts are to be generated when cams 0 to 7 are activated and/or deactivated.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
3.2	PI_CAM	BOOL	FALSE	1 = enable hardware interrupt: Cam on/off

Evaluation of a hardware interrupt (S7-300, S7-400)

If FM 352 triggers a hardware interrupt, interrupt information is available in the OB4x_POINT_ADDR tags of the hardware interrupt OB.

The following table shows the content of the double word OB4x_POINT_ADDR.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	Cam	0	0
2	Cam 7 on	Cam 7 off	Cam 6 on	Cam 6 off	Cam 5 on	Cam 5 off	Cam 4 on	Cam 4 off
3	Cam 3 on	Cam 3 off	Cam 2 on	Cam 2 off	Cam 1 on	Cam 1 off	Cam 0 on	Cam 0 off

You can see the cause of the interrupt in byte 1.

Cam: Evaluate bytes 2 and 3 according to the table.

Lost hardware interrupts

If the hardware interrupt OB is still busy processing a hardware interrupt, the module registers all subsequent hardware interrupt events. If an event occurs again before the hardware interrupt could be triggered, the module triggers a "hardware interrupt lost" diagnostic interrupt.

Control and feedback interface of the FM 352 (S7-300, S7-400)

Application

Very fast access to checkback and control signals is required in special applications or in an alarm level. You can obtain those data directly by reading the IO areas of the module.

To coordinate startup following each module startup, for example, after plugging a module or after a STOP-RUN transition of the CPU, you must call the CAM_CTRL instruction until the end of the startup is indicated by RETVAL = 0.

Note

For direct access to FM 352 data, always use the non-internal data and method described in this section. Otherwise, your user program will encounter difficulties accessing the module.

Reading checkback signals via direct access

The byte addresses are defined with an offset to the output address of the module. The names of the parameters correspond to the names in the channel DB.

In STL, you access the data using the PIB (read 1 byte) and PID (read 4 bytes) commands.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	PARA	Internal	Internal	DATA_ERR	Internal	DIAG	Internal	Internal
Byte 1	0	0	0	CAM_ACT	0	0	0	0
Byte 2	Internal							
Byte 3	0	0	FVAL_DONE	HYS	GO_P	GO_M	MSR_DONE	SYNC

Address	Bit number
Byte 4	ACT_POS
Byte 5	
Byte 6	
Byte 7	
Byte 8	TRACK_OUT
Byte 9	
Byte 10	
Byte 11	

Direct write access to control signals

The byte addresses are defined with an offset to the input address of the module. The names of the parameters correspond to the names in the channel DB.

In STL, you access the data using the PQB (write 1 byte) and PQW (write 2 bytes) commands.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	Internal							
Byte 1	0	CNTC1_EN	CNTC0_EN	CAM_EN	DIR_P	DIR_M	0	0
Byte 2	TRACK_EN							
Byte 3								

Example: Position process value ACT_POS

The start address of the module is 512.

```
STL
L PID 516      //Read the position process value (ACT_POS) with direct access:
               //Start address of the module + 4
```

Parameter transmission paths (S7-300, S7-400)

The term parameter refers to the following machine and cam data.

The following figure shows the parameter transmission paths of the FM 352.

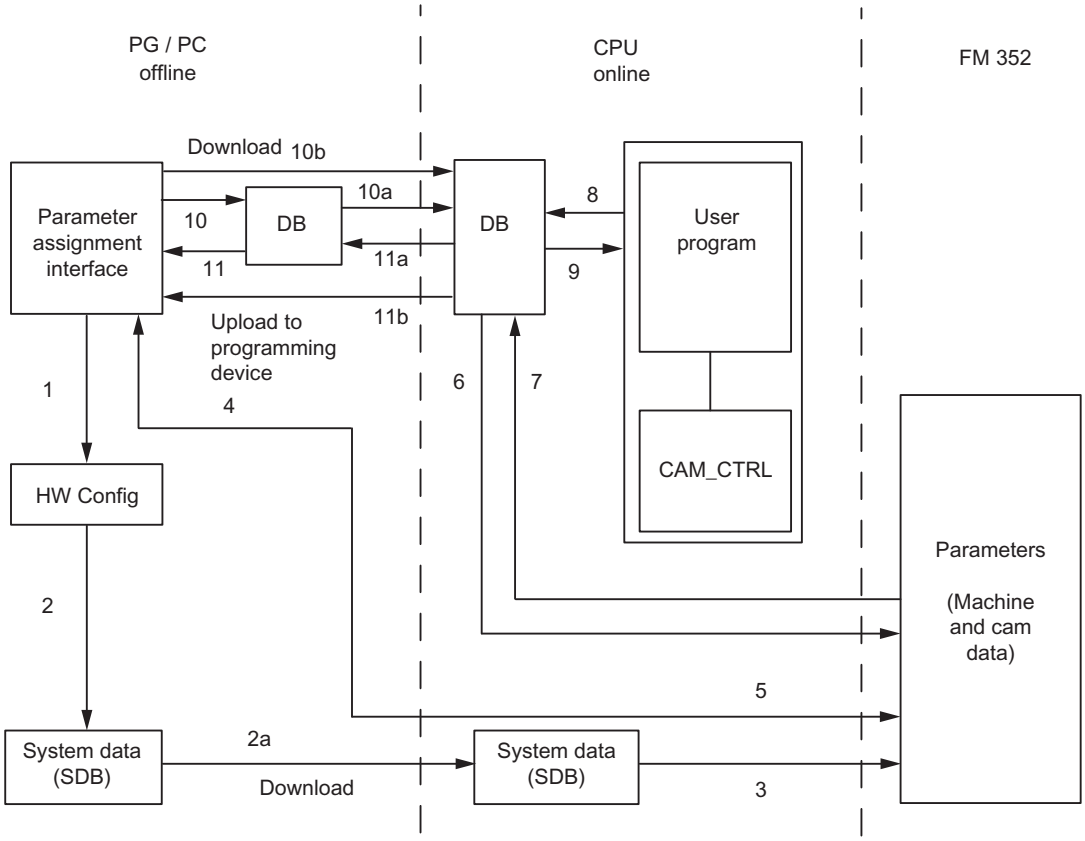

1 Save the parameters in the parameter assignment interface.

2 Saving and compiling the HW configuration, and download to the CPU.

3 The CPU writes the parameters to the module during system parameter assignment.

4 Upload the module parameters to the PG by selecting the "PLC > Upload to PG" command.

5 Download the parameters from the parameter assignment interface to the module with the "PLC > Download" command.

6 Write parameters to the module using jobs in the user program.

7 Read parameters from the module using jobs in the user program.

8 Save parameters from the user program to the online DB.

9 Read in parameters from the online DB to the user program.

10 Export parameters from the parameter assignment interface to the DB (offline or online DB); an offline DB must then be downloaded to the CPU.

11 Import parameters from an online or offline DB into the parameter assignment interface.

Some use cases for the transfer of parameters

Use case	Steps
You edit the parameters on the parameter assignment interface. The parameter should then be assigned automatically to the module during startup.	Perform steps 1, 2, and 3.
You edit parameters in the parameter configuration interface during commissioning in test mode.	Perform steps 4 and 5

Use case	Steps
The parameters edited during commissioning should be downloaded automatically during startup.	Perform steps 1, 2, and 3.
Create the parameters using the parameter assignment interface. When it starts up, the module should be assigned parameters only by the user program using data blocks.	Perform steps 10 and 6.
You need a comfortable means of creating data records for recipes.	Perform step 10.
Create the parameters using the parameter assignment interface. These should be available to the user program for temporary modifications.	Perform steps 1, 2, and 3 for automatic parameter assignment.
	Perform steps 10, 7 to enable access by the user program.
You modify existing parameters (exclusively) using the user program.	Perform steps 7, 9, 8, and 6.
You want to view the data modified by the user program on the parameter assignment interface.	Perform step 11.
Parameters modified by the user program should also be loaded automatically during startup.	Perform steps 6, 11, 1, 2, and 3.

2.1.3.5 Commissioning the FM 352 (S7-300, S7-400)

2.1.3.6 Using FM 352 from the user program (S7-300, S7-400)

Integrating the FM 352 into a user program

Instructions are available for integrating the FM 352 into a user program. These enable you to handle the desired functions with ease.

The instructions CAM_INIT and CAM_CTRL can be called in the CPU user program. They ensure communication between the CPU and the FM 352. The channel DB is created automatically as the instance DB for the CAM_CTRL instruction when you insert the CAM_CTRL instruction into the project.

All the parameter data associated with the channel of the module (machine data, cam data) is located in the parameter DB for the module. Use the command "Program blocks > Add new block" to add a CAM_PxxxTYPE type data block to your project. Depending on the quantity structure, you need the following DB types:

- Up to 16 cams: CAM_P016TYPE
- Up to 32 cams: CAM_P032TYPE
- Up to 64 cams: CAM_P064TYPE
- Up to 128 cams: CAM_P128TYPE

The instruction CAM_DIAG is also available for FM 352. This instruction allows you to transfer the diagnostic information to the diagnostic DB. The diagnostic DB is created automatically as the instance DB for the CAM_DIAG instruction when you insert the CAM_DIAG instruction into the project.

Commissioning the module

The module can be initially tested using the "Commissioning" dialog, even when no user program is available. Note the following warning in particular in this case.

WARNING

Injury and material damage could occur.

To prevent personal injury and material damage, observe the following points:

- Install an emergency stop switch in the vicinity of the computer.
  This is the only way you can ensure that the plant can be safely shut down in the event of computer or software failure.
- Install hardware limit switches, which act directly on the power units of all drives.
- Make sure that no one has access to the area of the plant in which moving parts are present.
- When the FM 352 is controlled and monitored simultaneously by your program and the "Commissioning" dialog, conflicts can occur, the effects of which are not clear. Therefore, always set the CPU to STOP mode or disable your program when you work with the "Commissioning" dialog.

2.1.3.7 FM 352: Online & Diagnostics (S7-300, S7-400)

Current module information (S7-300, S7-400)

Default diagnostics (S7-300, S7-400)

Error display through group error LED of FM 352 (S7-300, S7-400)

Where is the error displayed?

If the red group error LED is lit, an error occurred either at the module (internal error) or at the cable ports (external error). The LED is lit, even with errors that occur briefly, for at least 3 seconds.

Which errors are displayed?

The table below shows the error types indicated by the group error LED.

LEDs	Meaning	Notes
SF	Group error for internal and external errors	This LED indicates the following error states of FM 352: • Hardware interrupt lost • Watchdog expired • FM 352 not configured • FM 352 incorrectly configured • No external 24 V auxiliary supply • Front connector missing • Encoder wire break • Process error • Absolute encoder frame error • Incremental encoder pulse missing or zero mark signal missing To remove the error, see Error classes (Page 524).

Channel-specific diagnostics (S7-300, S7-400)

Triggering diagnostic interrupts (S7-300, S7-400)

Interrupt processing

The FM 352 can trigger hardware and diagnostic interrupts. You process these interrupts in an interrupt OB. If an interrupt is triggered without the corresponding OB being loaded, the CPU switches to STOP.

You enable the processing of diagnostic interrupts as follows:

1. Select the module in HWCN
2. Select "Edit" > Object Properties > Basic Parameters" to enable the diagnostic interrupt.
3. Save and compile the configuration.
4. Download the configuration to the CPU.

Overview of diagnostic interrupts

Events and errors which trigger a diagnostic interrupt:

- Operating error
- Machine data error
- Incorrect cam data
- Diagnostic errors

These errors are explained in detail in "Error classes (Page 524)".

Module response to an error with diagnostic interrupt

- Cam processing will be disabled.
- The synchronization will be cleared by the following diagnostic interrupts:
  - Front connector missing, no external auxiliary voltage
  - Zero mark error detected, cable fault (5 V encoder signals)
  - Traversing range exceeded (indicated by an operating error)
  - Set process value cannot be executed (indicated by an operating error).
- With one exception, control signals are no longer processed.
  Exception:
  When the software limit switches are overrun, a change of direction is still possible in simulation mode.
- Function switch and job processing continues.

Module detects an error ("incoming")

A diagnostic interrupt is an "incoming" event if at least one error is pending. If any errors are pending clearance, the queued errors are reported again as "incoming" events.

Sequence:

1. The FM 352 detects one or more errors and triggers a diagnostic interrupt. The error LED lights up. The error event is logged to the diagnostic buffer.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. You can now evaluate the start information of OB82.
4. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
5. For additional information, call the CAM_DIAG instruction.

The module detects the transition to an error-free state ("outgoing")

A diagnostic interrupt is only registered "outgoing" if all errors on the module are cleared.

Sequence:

1. The FM 352 detects that all errors have been eliminated and triggers a diagnostic interrupt. The error LED goes out. The diagnostic buffer remains unchanged.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
4. Evaluate the OB82_MDL_DEFECT bit.
   When this bit is "0", no errors are present on the module. You can close the evaluation session at this point.

Diagnostic interrupt control by CPU states

- Diagnostic interrupts are blocked by FM 352 when the CPU is in STOP mode.
- If not all of the queued errors are eliminated while the CPU is in STOP mode, the FM 352 will report the remaining errors as "incoming" events again at the next CPU transition to RUN.
- If all queued errors are eliminated when the CPU is in STOP mode, the error-free state of the FM 352 will **not** be reported by a diagnostic interrupt after the transition to RUN mode.

Error evaluation in the user program (S7-300, S7-400)

Fault reaction in the user program

The user program allows a targeted response to errors. The following options are available:

- Return values (RETVAL) of the integrated instructions as a group display for errors that occurred during execution of the instructions.
- Error bits _ERR of the jobs as a group display for errors that occurred while a job was being processed.
- The DATA_ERR error bit as a group display for an error that the FM 352 has detected during a write job.
- The error identifier in JOB_ERR for the cause of an error in communication between instruction and FM 352.
- The CAM_DIAG instruction for reading the diagnostic buffer of the FM 352. This is where you find out the causes of errors in jobs and asynchronous events (operator errors, diagnostic errors).
- Diagnostic interrupts for fast reaction to events.

Diagnostics buffer of the module (S7-300, S7-400)

Diagnostic events

The module´s diagnostic buffer contains a maximum of 4 diagnostic events and is organized as a ring buffer.

A diagnostic event is written in the buffer if an "incoming" (error) message is detected. This can be an alarm, a synchronous error (data error) or even an asynchronous error (operational errors and diagnostics errors). From a cause of error several entries can occur as follow-up errors. Outgoing messages do not create any entries in the diagnostic buffer.

For each diagnostic event the following is given:

- Status (always incoming)
- Internal error
- External error
- Error class

- Error number
- Channel number
- Cam number (for cam data errors)

When a diagnostic event is written in the diagnostic buffer, the checkback signal DIAG = 1 is set.

Using the CAM_DIAG instruction, the diagnostic buffer can be transferred as a whole to a data block (diagnostic DB) or displayed by the error evaluation dialog of the configuration software. If the diagnostic buffer is read, the module sets the checkback signal DIAG = 0.

Note

If the diagnostic buffer is read simultaneously from the CAM_DIAG instruction and the error evaluation dialog, it is possible that a new diagnostic event not be detected by the program.

Data and Structure of the Diagnostic DB (S7-300, S7-400)

Data and Structure of the Diagnostic DB

Note

Do not modify any data not listed in this table.

The following table shows the structure of the diagnostic DB:

Address	Name	Type	Initial value	Comment
0.0	MOD_ADDR (Enter!)	INT	0	Module address
256.0	JOB_ERR	INT	0	Communication errors
258.0	JOBBUSY	BOOL	FALSE	1 = job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = read diagnostic buffer unconditional
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostic data latest entry
272.0	DIAG[2]	STRUCT		Diagnostic data second entry
282.0	DIAG[3]	STRUCT		Diagnostic data third entry
292.0	DIAG[4]	STRUCT		Diagnostic data oldest entry

The diagram below shows the structure of a DIAG[n] diagnostic entry.

Address	Name	Type	Initial value	Comment
+0.0	STATE	BOOL	FALSE	0 = outgoing event 1 = incoming event
+0.1	INTF	BOOL	FALSE	1 = Internal error
+0.2	EXTF	BOOL	FALSE	1 = external error

Address	Name	Type	Initial value	Comment
+2.0	FCL	INT	0	Error class: 1: Operating error 4: Data error 5: Machine data error 7: Cam data error 15: Messages 128: Diagnostic errors
+4.0	FNO	INT	0	Error number 0 to 255
+6.0	CH_NO	INT	0	Channel number (always 1)
+8.0	CAMNO	INT	0	Cam numbers 0 to 127 with error class = cam data error

List of JOB_ERR messages (S7-300, S7-400)

JOB_ERR alarms

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgment when reading from module. Module removed during read operation or module defective.
80A1	32929	-32607	Negative acknowledgment when writing to module. Module removed during write operation or module defective.
80A2 [1)]	32930	-32606	Protocol error on Layer 2 (data transfer interrupted within PROFINET / PROFIBUS DP, for example, due to wire break, missing connector, parameter assignment error, etc.)
80A3 [1)]	32931	-32605	Protocol error at user interface/user (data transfer interrupted within PROFINET / PROFIBUS DP, for example, due to wire break, missing connector, parameter assignment error, etc.)
80A4 [1)]	32932	-32604	Communication problem on K bus
80B1	32945	-32591	Specified length wrong. FM_TYPE parameter in the channel DB is not correctly set for the module being used.
80B2	32946	-32590	The configured slot is empty.
80B3	32947	-32589	Actual module type does not match configured module type.
80C0 [1)]	32960	-32576	The module has not yet prepared the data to be read.
80C1 [1)]	32961	-32575	The data of a write job of the same type have not yet been processed on the module.
80C2 [1)]	32962	-32574	The module is currently processing the maximum number of jobs.
80C3 [1)]	32963	-32573	Required resources (memory etc.) currently in use.
80C4 [1)]	32964	-32572	Communication errors
80C5 [1)]	32965	-32571	Distributed I/Os not available.
80C6 [1)]	32966	-32570	Priority class abort (restart or background)
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read from the DB. (write job)
8532	34098	-31438	DB number of the parameter DB too high (write job)
853A	34106	-31430	Parameter DB does not exist. (write job)
8544	34116	-31420	Error in nth (n > 1) read access to a DB after error occurred. (write job)

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written to the DB. (read job)
8730	34608	-30928	Parameter DB on the CPU write-protected. The data cannot be written to the DB (read job).
8732	34610	-30926	DB number of the parameter DB too high (read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error in nth (n > 1) write access to a DB after error occurred. (read job)

[1] Errors 80A2 to 80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time. Messages in the form 7xxx indicate temporary states in communication.

Error classes (S7-300, S7-400)

Operating error (S7-300, S7-400)

Error class 1: Operating error

Operating errors are detected asynchronous to operator input/commands.

The following table shows the possible operating errors and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Software limit switch start overrun**		Yes
2	**Software limit switch end overrun**		Yes
3	**Traversing range start passed**		Yes
4	**Traversing range end passed**		Yes
13	**Set actual value on-the-fly cannot be executed**		Yes
	Cause	The software limit switches are outside the traversing range after the actual value on the fly is set (-100 m to +100 m or -1000 m to +1000 m). The shift resulting from set actual value / set actual value on-the-fly is more than ±100 m or ± 1000 m.	
	Effect	Axis not synchronized.	

Data error (S7-300, S7-400)

Error class 4: Data error

Data errors are detected synchronous to operator input/commands.

The following table shows the possible data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
10	**Zero offset error**		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. The software limit switches are outside the traversing range after zero offset (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	**Faulty actual value setting**		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
12	**Reference point error**		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
20	**Enable machine data not permitted**		No
	Cause	There are no new (error-free) machine data on the module.	
21	**Set actual value on-the-fly not permitted**		No
	Cause	An attempt was made to execute "set actual value on-the-fly" while "retrigger reference point" was active.	
27	**Illegal bit-coded setting**		No
	Cause	Unused and, in this case, unwritten bits are not 0. An attempt was made to select "length measurement" and "edge detection" at the same time.	
28	**Retrigger reference point is not permitted**		No
	Cause	An attempt was made to execute "retrigger reference point" while "set actual value on-the-fly" was active. An attempt was made to execute "retrigger reference point" with an SSI encoder.	
29	**Illegal bit-coded command**		No
	Cause	Unused and, in this case, unwritten bits are not 0.	
30	**Incorrect lead time**		No
31	**Incorrect cam number**		No
	Cause	Cam is invalid. The cam number is not in the range 0 to 127.	
32	**Incorrect cam start**		No
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam start is < 0 or greater than the end of the rotary axis.	

No.	Meaning		Diagnostic interrupt
33	**Incorrect cam end / incorrect activation time**		No
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. The cam is not activated at least for the duration of one pulse. With an inverse cam, there are not at least 4 pulses between the cam start and cam end.	
34	**Cancel set actual value not possible**		No
	Cause	The actual position value would be outside the working range with an SSI encoder and linear axis after making the setting.	
35	**Incorrect actual value specified by "set actual value" / "set actual value on-the-fly"**		No
	Cause	The specified actual value is outside the permitted numeric range of ±100 m or ±1000 m. The software limit switches would be outside the traversing range (-100m to +100 m or -1000 m to +1000 m) were this setting to be applied. The offset derived from set actual value / set actual value on-the-fly would be more than ±100 m or ±1000 m.	
107	**Parameters not set for the axis**		No
	Cause	There are no machine data on the axis. There are no machine data activated on the axis.	
108	**Axis not synchronized**		No
	Cause	One of the settings "set actual value" or "set actual value on-the-fly" was initiated although the axis is not synchronized.	
109	**Cam processing running.**		No
110	**Incorrect number of cams to be modified**		No

Machine data error (S7-300, S7-400)

Error class 5: Machine data error

The diagnostic interrupt is only triggered if an error is detected in the system data block (SDB).

The following table shows the possible machine data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
5	**Error in hardware interrupt setting**		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	**Incorrect minimum edge distance**		Yes
	Cause	You have entered a value < 0 or > 10^9 µm as the minimum edge-to-edge distance.	
8	**Incorrect axis type**		Yes
	Cause	You have specified neither 0 nor 1 as the axis type	

No.	Meaning		Diagnostic interrupt
9	**Incorrect end of rotary axis**		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or 1 to 10^8 µm (depending on the resolution).	
10	**Incorrect encoder type**		Yes
	Cause	The value for the encoder type is outside the valid range of 1 to 10	
11	**Incorrect distance per encoder revolution**		Yes
	Cause	The value for distance/encoder revolution is outside the permitted range of 1 to 10^9 µm (regardless of the resolution).	
13	**Incorrect number of increments per encoder revolution** (see "Increments per encoder revolution (Page 496)")		Yes
14	**Incorrect number of revolutions** (see "Number of encoder revolutions (Page 501)")		Yes
15	**Incorrect transmission rate**		Yes
	Cause	You have specified a transmission rate outside the permitted range of 0 to 3.	
16	**Incorrect reference point coordinate**		Yes
	Cause	The coordinate is outside the range of -100 m to +100 m or -1000 m to +1000 m, depending on the resolution. Linear axis: The coordinate is outside the working range. Rotary axis: The coordinate is greater than the end of the rotary axis or < 0.	
17	**Incorrect absolute value adjustment**		Yes
	Cause	SSI encoder: The value of the absolute encoder adjustment is not in the encoder range (increments per encoder revolution x number of revolutions - 1).	
18	**Incorrect "Retrigger reference point" type**		Yes
	Cause	You have specified a value other than 0, 1, 6 and 7.	
19	**Incorrect direction adaptation**		Yes
	Cause	You have specified a value other than 0 and 1.	
20	**Hardware monitoring not possible**		Yes
	Cause	You have set frame error monitoring to "FALSE" in the parameter DB.	
21	**Incorrect software limit switch start**		Yes
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any existing zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	**Incorrect software limit switch end**		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution) or is less than the software limit switch start. The software limit switch end (including any existing zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
144	**Incorrect number of cams**		Yes
	Cause	You have specified a value other than 0 to 3 for the number of cams.	

No.	Meaning		Diagnostic interrupt
145	**Incorrect hysteresis**		Yes
	Cause	The hysteresis is outside the range 0 to 65535 × resolution. Hysteresis is greater than ¼ × operating range or ¼ × rotary axis range.	
146	**Incorrect simulation velocity**		Yes
	Cause	The simulation speed is outside the range 1000 × RES to 3 × 10^7 × RES, or is greater than 5 × 10^8 µm/min. The simulation velocity cannot be set internally.	
147	**Incorrect track**		Yes
	Cause	Control of a track outside 0 to 15 (bits 0 to 15) was selected.	
148	**Incorrect selection of enable inputs**		Yes
	Cause	You wanted to activate a track outside 3 to 10 (bits 0 to 7) using an external signal.	
149	**Incorrect selection of special tracks**		Yes
	Cause	You wanted to define a track outside 0, 1 and 2 (bits 0, 1 and 2) as a special track.	
150	**Incorrect high count value track 0**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
151	**Incorrect high count value track 1**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
200	**Incorrect resolution**		Yes
	Cause	You have specified a resolution < 0.1 µm/pulse or >1000 µm/pulse. You have specified a distance per encoder revolution and a number of pulses per encoder revolution that results in a resolution of < 0.1 or > 1000.	
201	**Position encoder does not match the working range/rotary axis range**		Yes
	Cause	SSI position encoder and rotary axis: The encoder does not exactly cover the rotary axis range. Linear axis: The position encoder does not cover at least the operating range (including software limit switches).	

Cam data error (S7-300, S7-400)

Error class 7: Cam data error

The diagnostic interrupt is only triggered if an error is detected in the system data block (SDB).

The following table shows the possible cam data errors and their respective causes.

No.	Meaning		Diagnostic interrupt
1	**Illegal hardware interrupt**		Yes
	Cause	You attempted to define a process interrupt for a cam with a cam number > 7.	
2	**Incorrect track number**		Yes
	Cause	The track number is outside the range from 0 to 31.	

No.	Meaning		Diagnostic interrupt
3	**Incorrect cam start**		Yes
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam start is < 0 or greater than the end of the rotary axis.	
4	**Incorrect cam end**		Yes
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). The cam is at least one pulse long. Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. There are not at least 4 pulses between the cam start and cam end for an inverse cam.	
5	**Incorrect activation time**		Yes
	Cause	The activation time is < 0 µs. The maximum value (Page 508) depends on the number of cams.	
6	**Incorrect lead time**[1]		Yes
	Cause	The lead time is < 0 µs. The maximum value (Page 509) depends on the number of cams.	
50	**Too many cam sets**		Yes
	Cause	You wanted to enter more cam records than is possible with this number of cams.	
51	**Axis in operation**		Yes
	Cause	You wanted to enter cam records while the cam controller is active.	
52	**Parameters not set for the axis**		Yes
	Cause	You want to enter cam data although not machine data are active yet.	

[1] The error message can also be issued if you have assigned the parameter "inverted" as the count direction with an absolute encoder (SSI).

Alarms (S7-300, S7-400)

Error class 15: Alarms

The following table shows the possible alarms and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Start of parameter assignment**		No
	Cause	The module has detected parameter assignment via a system data block.	
2	**End of parameter assignment**		No
	Cause	The module has processed the parameter assignment via a system data block error-free.	

Diagnostic errors (S7-300, S7-400)

Error class 128: Diagnostic errors

The following table shows the possible diagnostic errors and their respective causes, effects and remedies.

No.	Meaning		Diagnostic interrupt
4	**External auxiliary voltage missing**		Yes
	Cause	External auxiliary 24 V voltage is not connected or has failed, front connector missing	
	Effect	See "Triggering diagnostic interrupts (Page 519)": • Cam processing is disabled • Track outputs are deactivated • With incremental encoders, synchronization is canceled • FM 352 has not been configured (checkback signal PARA = 0).	
	Correction	Make sure that the 24 V connection is correct (if the 24 V connection is correct, then the module is defective)	
51	**Watchdog expired**		Yes
	Cause	• Strong interference on FM 352 • Error in the FM 352	
	Effect	• Module is reset • If, after resetting the module, no module fault is detected, the module is ready for operation again • The module reports the expired watchdog with "incoming" and "outgoing"	
	Correction	• Eliminate the interference • Contact the relevant sales department which will require details of the exact circumstances leading to the error • Replace the FM 352	
52	**Internal module power supply failed**		Yes
	Cause	Error in the FM 352	
	Effect	• Module is reset • If, after resetting the module, no module fault is detected, the module is ready for operation again	
	Correction	Replace the FM 352	

No.	Meaning		Diagnostic interrupt
70	**Hardware interrupt lost**		Yes
	Cause	A hardware interrupt event detected by FM 352 cannot be reported because an identical event has not yet been processed by the user program/ CPU.	
	Effect	• Cam processing is disabled • Track outputs are deactivated • Synchronization is cancelled in incremental encoders	
	Correction	• Integrate OB 40 into the user program • Check bus connection of the module • Deactivate hardware interrupt • Adapt your hardware and software to suit your process requirements (for example, faster CPU, optimize user program)	
144	**Encoder wire break**		Yes
	Cause	• Encoder cable cut or not plugged in • Encoder has no transverse signals • Incorrect pin assignment • Cable length too long • Encoder signals short-circuited	
	Effect	• Cam processing is disabled • Track outputs are deactivated • Synchronization is cancelled in incremental encoders	
	Correction	• Check the encoder cable • Keep within encoder specifications • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog • Keep to the module's technical data	

No.	Meaning		Diagnostic interrupt
145	**Absolute encoder frame error**		Yes
	Cause	The frame traffic between FM 352 and the absolute encoder (SSI) is incorrect or has been interrupted: • Encoder cable cut or not plugged in • Incorrect encoder type • Encoder incorrectly set (programmable encoders) • Frame length incorrectly specified • Encoder returns incorrect values (encoder defective) • Interference on measuring system cable • Transmission rate selected too high	
	Effect	• Cam processing is disabled • Track outputs are deactivated • The last correct actual value remains unchanged till the end of the next correct SSI transfer	
	Correction	• Check the encoder cable • Check the encoder • Check the frame traffic between the encoder and FM 352	
146	**Missing pulses, incremental encoder**		Yes
	Cause	• Encoder monitoring has detected missing pulses • Number of increments per encoder revolution is incorrectly entered • Encoder defective: does not return the defined number of pulses • Bad or missing zero marker • Interference on the encoder cable	
	Effect	• Cam processing is disabled • Track outputs are deactivated • Synchronization is canceled	
	Correction	• Enter the correct number of increments/encoder revolution • Check the encoder and its cable • Comply with shielding and grounding regulations • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog	

2.1.4 Using FM 452 (S7-300, S7-400)

2.1.4.1 Terminal assignment of the front connector of FM 452 (S7-300, S7-400)

Front connector

Connect the following to the 48-pin front connector:

- the encoder
- the digital outputs and inputs
- the supply voltage of the encoders and digital outputs

Terminal assignment of front connector (X1)

The table below shows the terminal assignment of the front connector of FM 452.

Wiring	Name	Proximity switch	Incremental encoder	Absolute encoder
1		–		
2		–		
3	1L+	Auxiliary supply 24 VDC		
4	A / DAT	–	Encoder signal A (5 V)	SSI data
5	/A / /DAT	–	Encoder signal A inverse (5 V)	SSI data inverse
6	B / CLI [1)]	–	Encoder signal B (5 V)	SSI shift clock input [1)]
7	/B / /CLI [1)]	–	Encoder signal B inverse (5 V)	SSI shift clock input inverse [1)]
8	N	–	Zero mark signal (5 V)	–
9	/N	–	Zero mark signal inverse (5 V)	–
10	CLS [2)]	–	–	SSI shift clock output
11	/CLS [2)]	–	–	SSI shift clock output inverse
12	A*	Encoder signal A (24 V)		–
13	B*	–	Encoder signal B (24 V)	–
14	N*	–	Zero mark signal (24 V)	–
15	Q0	Digital output 0		
16	Q1	Digital output 1		
17	Q2	Digital output 2		
18	Q3	Digital output 3		
19	Q4	Digital output 4		
20	Q5	Digital output 5		
21	Q6	Digital output 6		
22	Q7	Digital output 7		
23	5.2 VDC	–	Encoder supply (5.2 V)	
24	24 VDC	Encoder supply (24 V)		

Wiring	Name	Proximity switch	Incremental encoder	Absolute encoder
25	M [3]	Encoder ground		
26	2L+	Auxiliary supply 24 VDC		
27	RE	–	Current sourcing/current sinking[4]	–
28	Q8	Digital output 8		
29	Q9	Digital output 9		
30	Q10	Digital output 10		
31	Q11	Digital output 11		
32	Q12	Digital output 12		
33	Q13	Digital output 13		
34	Q14	Digital output 14		
35	Q15	Digital output 15		
36	3L+	Auxiliary supply 24 VDC		
37	I0	Brake cam track enable		
38	I1	Length measurement/ edge detection/ set actual value on-the-fly		
39	I2	Reference point switch		
40	I3	Enable track signal 3		
41	I4	Enable track signal 4		
42	I5	Enable track signal 5		
43	I6	Enable track signal 6		
44	I7	Enable track signal 7		
45	I8	Enable track signal 8		
46	I9	Enable track signal 9		
47	I10	Enable track signal 10		
48	M [3]	Ground for auxiliary voltages		

[1] In listen-in mode

[2] In master mode

[3] Ground connections wired on module

[4] **Note:** If you are connecting an incremental encoder from a third party in series-mode switching (current sourcing/current sinking), the following must be observed:

- Sourcing output: Connect RE (27) with ground (25).
- Sinking output: Connect RE (27) with +24 V (24).

Additional information about design and wiring can be found in the Operating Instructions (http://support.automation.siemens.com/WW/view/en/1123332) of the electronic cam controller FM 452.

Auxiliary voltage for sensors and digital outputs (1L+, 2L+, 3L+)

The 24 VDC auxiliary voltage of the encoders and digital outputs is monitored:

- for wire-break of the 24 V feed line
- for power failure

This means that 24 V DC (terminal 24) and 5.2 V DC (terminal 23) are available at the front connector (X1) for the different types of encoders.

Load current supplies

The DC load current supply must meet the following requirements:

Only a safe, isolated extra-low voltage of ≤ 60 V DC may be used as the load current supply. Safe isolation can be implemented, for example, by adhering to the specifications

- VDE 0100 Part 410 / HD 384-4-41 / IEC 364-4-41
  (as protective extra low voltage) and/or
- VDE 0805 / EN 60950 / IEC 950
  (as safety low voltage SELV) or
- VDE 0106 Part 101.

11 digital inputs (I0 to I10)

You can connect bounce-free switches (24 V current sourcing) or non-contact sensors (2 or 3-wire proximity switches) via 11 digital inputs.

The digital inputs are not monitored for short circuits or wire break and have a non-isolated connection to the module chassis.

The state of each input can be read off from the respective LED.

16 digital outputs (Q0 to Q15)

The state (on/off) of tracks 0 to 15 is output at 16 digital outputs. The digital outputs have a non-isolated connection to the module chassis.

Loads supported:

- Operating voltage 24 V
- Current load 0.5 A / short-circuit proof

The associated LED indicates the state of each output.

2.1.4.2 Cam control functions (S7-300, S7-400)

Influence of functions on the switching characteristics of time-based cams (S7-300, S7-400)

Actual value changes

A time-based cam can be skipped by the following functions that change the actual value:

- Set actual value
- Set actual value on-the-fly

- Zero offset
- Retrigger reference point

Activating a time-based cam

If you skip the start of a time-based cam with one of the above functions, this cam will be activated provided the actual direction of the axis motion is the same as the effective direction assigned for the cam. The programmed cam activation time starts.

Note

If the axis is at a standstill, the direction of movement is influenced by fluctuations of the actual value.

Set a hysteresis higher than the fluctuation of the actual value signal in order to suppress flutter when the axis is at a standstill.

This retains the last determined direction of movement while the axis is at a standstill.

WARNING

Risk of injury and material damage.

Functions which modify the actual value at rotary axes may cause the unwanted activation of time-based cams.

You should always set the "invalid" option for the time-based cams of a rotary axis if you want to influence the actual value using the functions mentioned above.

Set Actual Value/Set Actual Value on-the-fly/Cancel Set Actual Value (S7-300, S7-400)

Definition

Use the "Set actual value" and "Set actual value on-the-fly" functions to assign a new coordinate to the current encoder value. This shifts the coordinate system by the value: ACT_{new} - $ACT_{current}$

Whereby:

- ACT_{new} is the default value
- $ACT_{current}$ is the process value at the time of execution

Determining new coordinates

Calculate all your default positions in the shifted coordinate system based on the following formula:

$$\text{Coordinate}_{new} = \text{Coordinate}_{old} + (ACT_{new} - ACT_{current})$$

Requirements

- The axis must be synchronized.
- With "Set process value on-the-fly": Digital input I1 must be interconnected.

Function sequence

1. Enter the coordinate for the process value, or for the process value to set on-the-fly in the channel DB.
   - Linear axis:
     You must select the specified process value so that the software limit switches are still within the valid traversing range after the function is called.
     The absolute value of the offset derived from **(ACT_{new} - $ACT_{current}$)** must be less than or equal to the absolute value of the permitted traversing range (maximum 100 m or 1000 m).
   - Rotary axis:
     Rule for the defined process value:
     0 ≤ process value < end of rotary axis
2. Set the appropriate trigger bits at the channel DB.
3. Call the CAM_CTRL or CAM_CTRL_452 instruction.
   "Set process values" is executed immediately.
   "Set process value on-the-fly" is executed at the next positive edge at digital input I1. The FVAL_DONE bit is set.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.4	AVAL_EN	BOOL	FALSE	1 = set process value
36.5	FVAL_EN	BOOL	FALSE	1 = set process value on-the-fly
90.0	AVAL	DINT	L#0	Process value coordinate
94.0	FVAL	DINT	L#0	Coordinate for on-the-fly process value
25.5	FVAL_DONE	BOOL	FALSE	1 = set process value on-the-fly executed

Effects of the function

The example "set process value" to 400 mm (at position 200 mm) demonstrates how this function shifts the coordinate system. Resultant effects:

- The position of the working range is **not** physically shifted.
- The individual points (such as the software limit switches) are assigned new coordinate values.
- The cams retain their coordinate values, and are therefore located at a different physical position.

- If the axis is synchronized and cam processing is enabled, the position process value may skip cam edges or entire cams as a result of this function.
- Status changes of the cam which would normally trigger an interrupt may be lost.

Note

Make allowances for the switching characteristics of time-based cams (Page 535).

The table below shows how "Set process value" / "Set process value on-the-fly" shifts the coordinate system.

Set process value	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
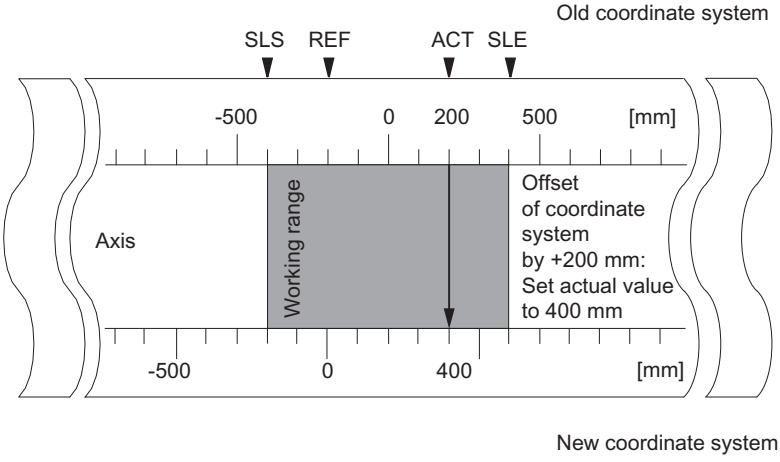	-400	-200	200	400
	-200	0	400	600

Cancelling the function

The "Undo set process value" function can be used to reset the coordinate shift caused by "Set process value" or "Set process value on-the-fly".

Once "Set process value on-the-fly" has been triggered, it can no longer be deleted before execution by a positive edge at input I1. However, it can be overwritten by a new "Set process value on-the-fly".

These functions are reset when a module is started up.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.2	AVALREM_EN	BOOL	FALSE	1: Undo process value setting

Possible causes of error

"Set process value on-the-fly" and "Retrigger reference point" may not be executed in parallel.

The "Set process value on-the-fly" function can signal an operator error if the function would cause a software limit switch to exceed the valid traversing range with a positive edge at I1. This operating error is signaled by a diagnostic interrupt, and written to the diagnostic buffer.

Set zero offset (S7-300, S7-400)

Definition

The "Zero offset" function lets you shift the zero point in the coordinate system by a defined value. The sign determines the offset direction.

Determining a new coordinate

All values of the shifted coordinate system can be calculated according to the following formula:

$\text{Coordinate}_{new} = \text{Coordinate}_{old} - (\text{ZPO}_{new} - \text{ZPO}_{old})$

ZPO_{old} identifies any existing zero offset. If no zero offset was active prior to the call, set a 0 value at ZPO_{old}.

This equation allows you to calculate the coordinates for the software limit switches, for example.

Function sequence

1. Write the zero offset value to the channel DB.
   - Linear axis:
     The zero offset must be selected so that the software limit switches remain within the permissible numerical range after the function is called.
   - Rotary axis:
     Rule for the zero offset:
     value of zero offset ≤ end of the rotary axis.
2. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.6	ZOFF_EN	BOOL	FALSE	1 = set zero offset
86.0	ZOFF	DINT	L#0	Zero offset

Effects on a linear axis

The example of a zero offset of -200 mm demonstrates that this function shifts the coordinate system in positive direction. Resultant effects:

- The working range is **not** physically shifted.
- The individual points (such as the software limit switches) are assigned new coordinate values.

- The cams retain their coordinate values, and are therefore located at a different physical position.
- If the axis is synchronized and cam processing is enabled, the positional process value may skip cam edges or entire cams as a result of this function.
- Status changes of the cam which would normally trigger an interrupt may be lost.

Note

Make allowances for the switching characteristics of time-based cams (Page 535).

The following table shows how the coordinate system is offset by "Zero offset".

Zero offset	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
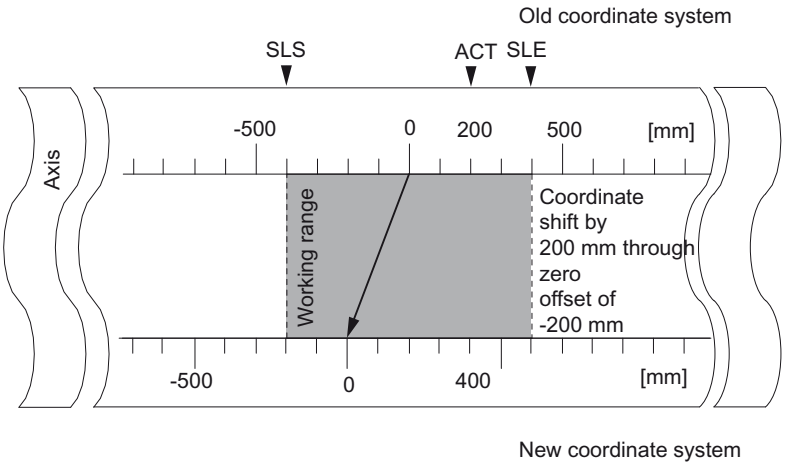	-400	-200	200	400
	-200	0	400	600

Effects on a rotary axis

The example of a -45° zero offset demonstrates how this function rotates the coordinate system.

The following table shows how the coordinate system is rotated by "Zero offset".

Tool when ACT = 340°	Tool when ACT = 25°
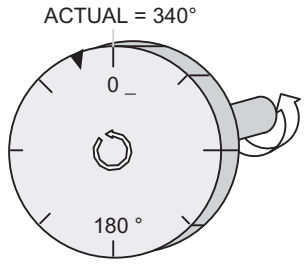	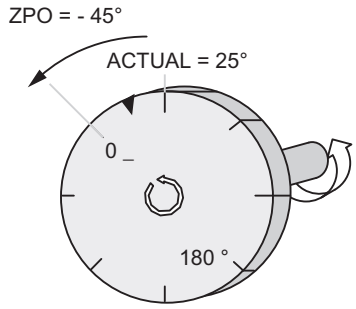
The zero mark rotates by -45 degrees. All axis points are assigned new coordinate values.	

Including a **ZPO_{old} = 0**, the result is a new value of 385°

As process value resumes at 0 at the end of the rotary axis operating in positive directional rotation, the calculation returns an process value of 25°:

$Coordinate_{new}$ = $Coordinate_{old}$ - (ZPO_{new} - ZPO_{old}) - end of rotary axis

The **end of rotary axis** value only needs to be subtracted when

$Coordinate_{old}$ - (**ZPO_{new} - NPV_{old}**) is greater than the end of rotary axis.

Loss of synchronization

If synchronization is lost due to an error, or reset due to "retrigger reference point", a zero offset is **retained**.

Cancelling function

A zero offset of 0 resets any existing zero offset.

Set reference point (S7-300, S7-400)

Definition

The "set reference point" function is used to synchronize the axis. The function shifts the working area. All shifts generated by a zero offset or "set actual value" are retained.

The function projects the operating range on the axis. Different value settings will thus move the working range to different positions within the physical range of the axis.

Requirements

Cam processing must be disabled.

Function sequence

1. Write the value for the reference point coordinate to the channel DB.
   - Linear axis:
     The reference point coordinates must not lie outside the software limit switches. This also applies to the reference point coordinate in a shifted coordinate system.
   - Rotary axis:
     Rule for the reference point coordinate:
     0 ≤ reference point coordinate < end of rotary axis
2. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.3	REFPT_EN	BOOL	FALSE	1 = set reference point coordinate
98.0	REFPT	DINT	L#0	Reference point coordinate
25.0	SYNC	BOOL	FALSE	1 = axis synchronized

Effects of the function

The example "Set reference point" to 300 mm demonstrates how this function shifts the axis operating range.

This has the following effects:

- The actual position is set to the value of the reference point coordinate.
- The working range is physically shifted on the axis.
- The various points retain their original coordinates, but are now at new physical positions.
- The SYNC bit is set in the checkback signals.

The following table shows how "Set reference point" shifts the operating range on the axis.

Set reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
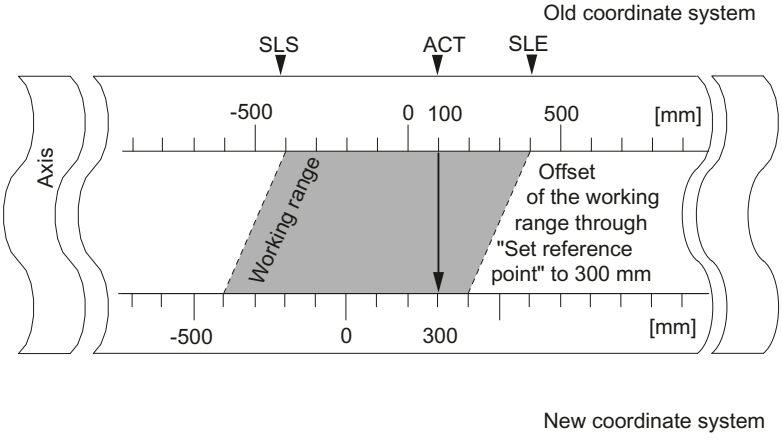	-400	-200	100	400
	-400	-200	300	400

Special features of the absolute encoder

This function is necessary for absolute encoder adjustment (Page 564).

Change cam edges (S7-300, S7-400)

Definition

The "change cam edges" function can be used to change the cam start and, for position-based cams, the end of a specific position-based cam in runtime.

Requirements

The cam you want to change must be valid.

Function sequence

1. Enter the cam number in the channel DB.
2. At a position-based cam:
   Enter the cam start and cam end in the channel DB.
   For a time-based cam:
   Enter the cam start value in the channel DB.
3. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.7	CH01CAM_EN	BOOL	FALSE	1 = write function for cam edges (1 cam)
102.0	CAM_NO	INT	0	Cam number
104.0	CAM_START	DINT	L#0	Cam start
108.0	CAM_END	DINT	L#0	Cam end

Effects of the function

The FM x52 shifts first the on edge and then the off edge of the cam. This sequence does not depend on the direction in which the cam is shifted.

Exception:

The sequence described above may briefly generate an inverse cam (Page 439) if the new cam start is greater than the old cam end.

The figure below shows the each stage in cam edge change.

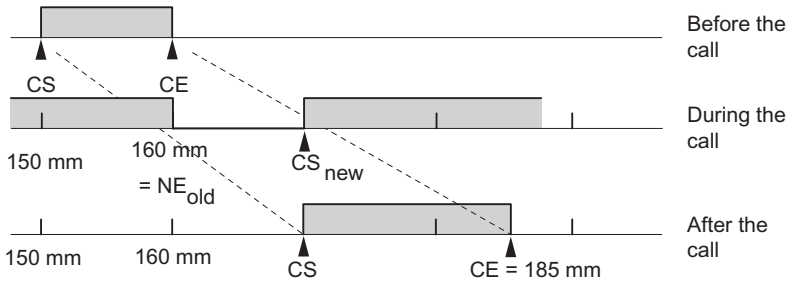

Note

If a hardware interrupt has been enabled for this cam, FM x52 can trigger one or two hardware interrupts when it detects the inverse cam depending on parameter assignment.

Changing the on and/or off edge can result in a cam edge or the entire cam being skipped.

Make allowances for the switching characteristics of time-based cams (Page 535).

Cam status changes that would normally trigger a hardware interrupt can be lost.

Reading modified values

You can read out the modified values by calling one of the jobs CAM1RD_EN to CAM8RD_EN.

Cancelling the function

The modified values are lost when you restart the module.

Perform "Fast Cam Parameter Change" (S7-300, S7-400)

Definition

The "Fast cam parameter change" function can be used to modify a group of up to 16 cams in runtime.

Requirements

The cams you want to modify must be valid.

Function sequence

1. Enter the number of cams to be modified in the channel DB.
2. Enter the number of the first cam to be modified in the channel DB.
3. Set the trigger bits for the required modifications.
4. Declare the new values at the channel DB.

5. Repeat steps 2 to 4 for each cam to be modified.
6. Set the relevant trigger bit in the channel DB.

Data used in the channel DB

Address Absolute	Name	Type	Initial value	Comment
37.0	CH16CAM_EN	BOOL	FALSE	1 = write fast cam parameter change function (16 cams)
176.0	C_QTY	BYTE	B#16#0	Number of cams to modify
177.0	DIS_CHECK	BOOL	FALSE	1 = disable data check

Address relative	Name	Type	Initial value	Comment
+0.0	CAM_NO	BYTE	B#16#0	Number of the cam to modify
+1.0	C_EFFDIR	BOOL	FALSE	1 = change the effective direction of the cam
+1.1	C_CBEGIN	BOOL	FALSE	1 = change the cam start to the value CBEGIN
+1.2	C_CEND	BOOL	FALSE	1 = change the cam end / activation time to the value CEND
+1.3	C_LTIME	BOOL	FALSE	1 = change the lead time to the value LTIME
+1.4	CAM_OFF	BOOL	FALSE	1 = deactivate the cam during the cam modification
+1.5	EFFDIR_P	BOOL	FALSE	1 = positive effective direction (plus)
+1.6	EFFDIR_M	BOOL	FALSE	1 = negative effective direction (minus)
+2.0	CBEGIN	DINT	L#0	New cam start
+6.0	CEND	DINT	L#0	New cam end / new activation time
+10.0	LTIME	INT	L#0	New lead time

Deactivating a cam when making modifications

To maintain consistency, always deactivate the cam (CAM_OFF) when modifying its start and end settings.

Data check by the module

Use the DIS_CHECK (channel DB) parameter to specify whether or not to disable the validation of transferred data by FM x52. If you disable the data check, you must take steps to verify that only permitted values are transferred. If you transfer illegal values without a check, the module can behave unexpectedly.

- FALSE: The module checks all data to be transferred
- TRUE: The check of cam parameter data is disabled. This allows for faster activation of the data to be changed on FM x52.
  Regardless of this function, the module always checks whether
  - the axis is assigned
  - the number of cams to be changed (C_QTY) is valid
  - the cam (cam number) to change is valid.

The data are only enabled on the module after having been checked and found faultless.

Any faulty data are rejected.

Effects of the function

Note

Make allowances for the switching characteristics of time-based cams (Page 535).

Reading modified values

You can read the modified values by calling one of the jobs CAM1RD_EN to CAM8RD_EN.

Cancelling the function

The modified values are lost when you restart the module.

Executing "Length measurement" and "Edge detection" (S7-300, S7-400)

Definition

You can establish the length of a workpiece using the "Length measurement" and "Edge detection" functions.

Length measurement and edge detection are active and remain active until you disable these functions or select a different measuring method. The CAM_CTRL or CAM_CTRL_452 instruction enables length measurement if you select both measuring methods in parallel.

Requirement

A **bounce-free** switch must be connected to input I1.

Function sequence

The FM 452 updates the data on the module at different times Depending on the type of measurement. FM 452 reports each update at a parameter on the feedback interface.

Length measurement

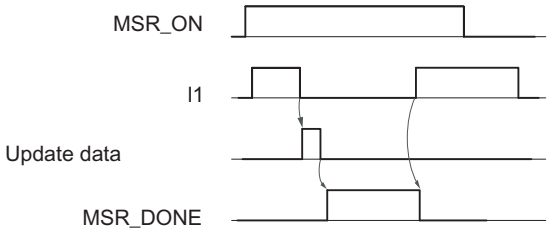

1. Set the function switch for "length measurement".
2. A positive edge at input I1 starts the length measurement.
3. The negative edge at input I1 stops the current measurement. The FM 452 updates the initial value, final value and length data.
4. When parameter MSR_DONE is set, FM 452 reports the data update. The parameter indicates completion of the measurement. The results of the measurement can be read out.
5. The start of the next measurement with a rising edge at I1 resets the MSR_DONE parameter.

The FM 452 does not update the data if the function is disabled during length measurement. The MSR_DONE parameter remains reset.

Edge detection

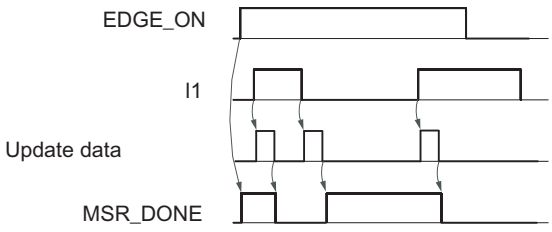

1. Enter a value for the minimum edge distance in the parameter DB. Write and enable machine data.
2. Set the "edge detection" function switch. This sets the MSR_DONE parameter.
3. The positive edge at input I1 starts edge detection. The results of the measurement are updated and can be read out, and the start value of the measurement is entered; the end value and length assume the value -1.
4. After the update, FM 452 reports the change by resetting the MSR_DONE parameter.
5. The negative edge at input I1 stops the current measurement. The FM 452 updates the data for the end measurement and length values.

6. After the update, FM 452 reports the change by setting the MSR_DONE parameter. The results of the measurement can be read out.
7. The start of the next measurement with a rising edge at I1 resets the MSR_DONE parameter.

The FM 452 does not update the data if the function is disabled during edge detection. The MSR_DONE parameter remains reset.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
25.1	MSR_DONE	BOOL	FALSE	1 = length measurement completed
34.0	EDGE_ON	BOOL	FALSE	1 = edge detection on
34.2	MSR_ON	BOOL	FALSE	1 = length measurement on
38.2	MSRRD_EN	BOOL	FALSE	1 = read measured values
112.0	BEG_VAL	DINT	L#0	Initial value
116.0	END_VAL	DINT	L#0	End value
120.0	LEN_VAL	DINT	L#0	Length

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	**Minimum edge distance for edge detection** Valid value range: • 0 ... 1,000,000,000 µm

The minimum edge distance is used to define a range after the start of a measurement is detected with an edge detection. The measurement will be discarded if the measuring operation ends within this range.

The start of the measurement is not signaled unless the minimum edge distance has been traveled.

Conditions of length measurement

- The CPU program requires an interval of sufficient length between the off and on edges at input I1 in order to be able to evaluate the result of the measurement before a new measurement is triggered.
- The minimum interval between the positive and negative edges at input I1 and between the negative edge and the next positive edge at input I1 must be greater than 2 ms.

hardware interrupts

A hardware interrupt can report the start and end of a measurement in the FM 452.

Faulty measurement

The FM 452 returns the length value -1 in the event of errors in length measurement / edge detection.

A "length measurement" or "edge detection" may only perform up to 126 zero passes in one direction. The zero pass represents the rotary axis transition from the end of rotary axis value to 0 and vice versa. The FM 452 reports incorrect "length measurement" or "edge detection" if it detects more than 126 zero passes in one direction, regardless of whether or not it then detects any zero passes in the opposite direction.

A length measurement is also considered faulty if:

- the length measured at a rotary axis is greater than 2^{31}.
- the on and off edges are detected simultaneously by the FM 452 (for example due to switch bounce).

This event can be signaled as a hardware interrupt.

Shift of the coordinate system during a length measurement

Conditions under which a shift of the coordinate system will influence the measured length:

- You are using an incremental encoder or an initiator or are operating the FM 452 in simulation mode.
- You are executing a "set reference point" or "retrigger reference point" function while a length measurement is active.

Example

To utilize the above mentioned influences on the measured length:

Your system always develops slip when you perform a length measurement.

The "Retrigger reference point" function can be used to correct this slip to allow correct length measurement values to be output.

Retrigger reference point (S7-300, S7-400)

Definition

The "Retrigger reference point" function can be used to synchronize the axis as a reaction to a recurring external event.

The function remains active until you deactivate it.

Requirements

- You are using an incremental encoder or an initiator.
- The external event may represent the zero mark signal of an incremental encoder or reference point switch at input I2.

Function sequence

1. Set the value for the reference point coordinate at the parameter DB.
2. Set the type of "retrigger reference point" at the parameter DB.
   Options:
   – Evaluation of the zero mark of the encoder only (RETR_TYPE = 7).
   – Evaluation of the reference point switch only (RETR_TYPE = 6).
   – Evaluation of the zero mark signal
     in positive direction: Evaluation of the first rising edge of the zero mark after passing the reference point switch in positive direction (RETR_TYPE = 0).
     in negative direction: Evaluation of the first falling edge of the zero mark after passing the reference point switch in negative direction (RETR_TYPE = 1).
3. Write and enable machine data.
4. Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.3	REFTR_ON	BOOL	FALSE	1 = retrigger reference point
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	Reference point coordinate
52.0	RETR_TYPE	DINT	L#0	Reference point retrigger mode

Effects of the function

- FM x52 evaluates the zero mark and reference point switch in line with the axis direction of motion.
  – It evaluates the positive edges when the axis moves in positive direction.
  – It evaluates the negative edges when the axis moves in negative direction.
- It sets the actual position to the value of the reference point coordinate.
- The working range is physically shifted on the axis.
- The various points retain their original values, but are now at new physical positions.
- Cam status changes that would normally trigger a hardware interrupt can be lost.
- The SYNC bit in the checkback signals is set.

Note

Make allowances for the switching characteristics of time-based cams (Page 535).

Example

Rules for the example:

- The module evaluates the positive edges of the reference point switch and zero mark signals (axis moving in positive direction).
- Value of the reference point coordinate = 300 mm.
- No zero offset is active at the time of execution.

The following table shows how "Retrigger reference point" shifts the operating range on the axis.

Retrigger reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
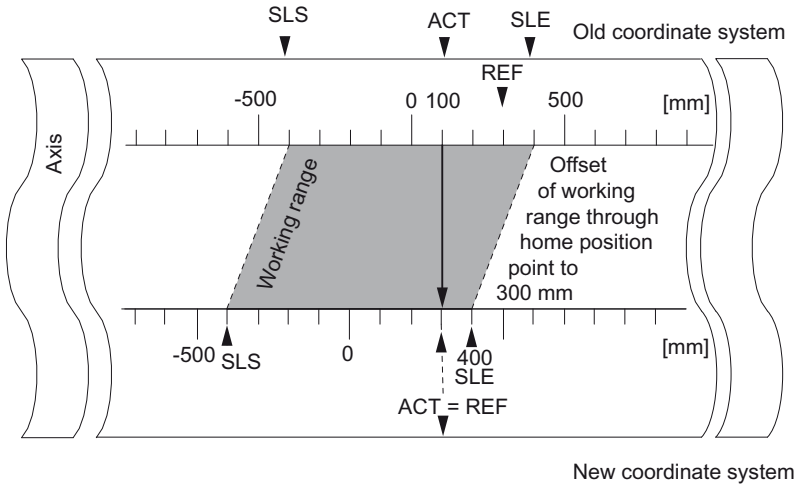	-400	300	100	400
	-400	300	300	400

Inclusion of a zero offset

Any active zero offset is included in the "Retrigger reference point" function. The reference point coordinate setting is thus calculated according to the formula:

Ref = Ref_{MD} - Zero offset

Ref_{MD} is the value of the reference point coordinate store in machine data.

The following table shows how "Retrigger reference point" with zero offset shifts the operating range on the axis.

Retrigger reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
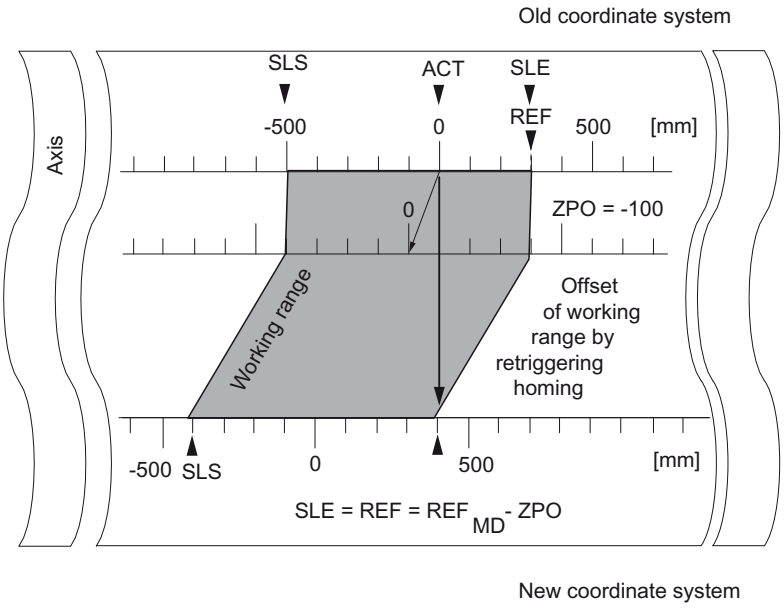	-500	300	0	300
	-400	400	100	400
	-400	400	400	400

Deactivate software limit switch (S7-300, S7-400)

Definition

Use the "Deactivate software limit switch" function to disable monitoring of the software limit switches at a linear axis.

The function remains active until you deactivate it. This re-activates the software limit switches originally parameterized.

Function sequence

Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.4	SSW_OFF	BOOL	FALSE	1 = Software limit switches deactivated

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L#-1000000000	Software limit switch start
68.0	SSW_END	DINT	L#1000000000	Software limit switch end

Effects of the function

- simulation
  - Simulation mode stops when the axis overruns a software limit switch.
  - You can resume simulation mode by disabling software limit switch monitoring. The axis moves in the defined direction.
- Zero offset when monitoring is disabled
  If a zero offset is set in which software limit switches are located within traversing range limits, the actual value may still be outside the valid number range.
- Cams located outside the parameterized software limit switches can be switched.

CAUTION

Risk of material damage!

If you reduce the traversing range by setting software limit switches to protect your system, you risk damage to your equipment when these switches are disabled.

In the planning and engineering phases for your plant, you should be certain that the drive can travel in the entire physical traversing range.

Executing "Simulation" (S7-300, S7-400)

Definition

The "Simulation" function allows you to activate the cam controller without connected encoders.

Function sequence

1. Set the simulation velocity at the parameter DB.
2. Write and enable machine data.
3. Set either a positive or negative simulation direction at the channel DB.
4. Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.2	DIR_M	BOOL	FALSE	1 = simulation in negative direction
15.3	DIR_P	BOOL	FALSE	1 = simulation in positive direction
34.1	SIM_ON	BOOL	FALSE	1 = simulation on

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	SIM_SPD	DINT	L#0	Simulation speed

Effects when simulation mode is activated

- Encoder signals will be ignored.
- The encoder input monitoring functions are disabled.
- All encoder errors reported will be reset.
- FM x52 simulates axis motion at a constant simulation velocity.
- Cam processing is disabled while simulation mode is active. However, you can then enable these operations again. Synchronism will be retained in this case.
- Starting at the current actual value, the actual position value changes dynamically based on the simulation velocity and direction.

Effects when simulation mode is deactivated

- Cam processing will be disabled.
- The synchronization of an incremental encoder or initiator will be cleared.. The actual value will be reset to the value of the reference point coordinate.
- The module reports the actual position value which corresponds with the absolute encoder value. The module then evaluates the encoder signals as defined at the machine parameters.

Limit values

The minimum and maximum simulation velocity depend on the resolution (Page 569).

Velocity

There may be differences between the set velocity and the actual velocity (Page 569) at which the module operates.

Read "count values of counter cam tracks" (S7-300, S7-400)

Definition

The "Counter cam track counter values" function is used to read out the current counter values.

Function sequence

1. Specify the counter cam tracks and the high count values in the machine data.
2. Write and enable machine data.
3. Enable the count function.
4. The module sets the count value to its high limit.
5. The count value decrements by the count of 1 at each positive edge of the track result signal.
6. Set the trigger bit at the channel DB to read the count values.
7. The module writes both count values to the channel DB. The module outputs a 0 value for tracks not assigned as a counter cam track.
8. The module sets track identifier bit = 1 at the counter cam track when the count value = 0.
9. It resets the track identifier bit = 0 at the next negative edge of the track result signal, and resets the counter to its high limit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.5	CNTC0_EN	BOOL	FALSE	1 = enable count function of counter cam track 0
15.6	CNTC1_EN	BOOL	FALSE	1 = enable count function of counter cam track 1
38.3	CNTTRC_EN	BOOL	FALSE	1 = read count values of counter cam tracks
124.0	CNT_TRC0	INT	0	Current count value of counter cam track 0
126.0	CNT_TRC1	INT	0	Current count value of counter cam track 1

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
99.0	SPEC_TRC0	BOOL	FALSE	1 = track 0 is counter cam track
99.1	SPEC_TRC1	BOOL	FALSE	1 = track 1 is counter cam track
100.0	CNT_LIM0	DINT	L#2	High count value of counter cam track 0
104.0	CNT_LIM1	DINT	L#2	High count value of counter cam track 1

Read out "Position and track data" (S7-300, S7-400)

Definition

The "Position and track data" function can be used to read the current position, the velocity and the track identifier bits. The track identifier bits are recorded before being logically linked to machine and channel data.

The algorithm implemented in the FM x52 calculates velocity changes greater than 1 pulse/ 4 ms. The indicated velocity includes this inaccuracy, and is thus unsuitable in particular for closed-loop control. The internal velocity value used for dynamic cam control offers a higher precision.

Function sequence

1. Set the trigger bit in the channel DB.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
38.4	ACTPOS_EN	BOOL	FALSE	1 = read position and track data
128.0	ACTPOS	DINT	L#0	Current position
132.0	ACTSPD	DINT	L#0	Current velocity
136.0	TRACK_ID	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31

Read out "Encoder data" (S7-300, S7-400)

Definition

The "Encoder data" function can be used to read out current encoder data and the value for absolute encoder adjustment.

Requirements

You can read out the value for absolute encoder adjustment (Page 564) after you have implemented the "Set reference point" (Page 541) function.

Function sequence

1. Set the trigger bit in the channel DB.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
38.5	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
140.0	ENCVAL	DINT	L#0	Encoder value / counter value (internal representation)
144.0	ZEROVAL	DINT	L#0	Counter value at the last zero mark (internal representation)
148.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment

Read out "Cam and track data" (S7-300, S7-400)

Definition

The "Cam and track data" setting can be used to read the current cam and track identifier bits and the position. The track identifier bits are recorded before being logically linked to machine and channel data.

Function sequence

1. Enter type ID = 1 at the FM_TYPE parameter of the channel DB. This allows you to read 24 bytes of cam and track data.
   If you enter type ID = 0, only the cam identifier bits (16 bytes) will be read.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
12.0	FM_TYPE	BOOL	FALSE	0 = FM 352 to V4.0 1 = FM 352 V5.0 or higher and FM 452
38.6	CAMOUT_EN	BOOL	FALSE	1 = read cam and track data
152.0	CAM_00_31	DWORD	DW#16#0	Cam identifier bits for cams 0 to 31
156.0	CAM_32_63	DWORD	DW#16#0	Cam identifier bits for cams 32 to 63
160.0	CAM_64_95	DWORD	DW#16#0	Cam identifier bits for cams 64 to 95
164.0	CAM_96_127	DWORD	DW#16#0	Cam identifier bits for cams 96 to 127
168.0	TRACK_ID1	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31
172.0	ACTPOS1	DINT	L#0	Current position

Setting "Control signals for the cam controller" (S7-300, S7-400)

Definition

The "Control signals for the cam controller" function can be used to enable cam processing and the tracks.

Function sequence

1. Set the required bits at the channel DB.
2. The data are transferred to the module each time the CAM_CTRL or CAM_CTRL_452 instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.4	CAM_EN	BOOL	FALSE	1 = enable cam processing
16.0	TRACK_EN	WORD	W#16#0	Enable cam tracks 0 to 12 Bit 0 = track 0

Effects

Cam processing will be started or stopped based on the enable status.

The track identifier bits of enabled tracks are transferred to the track signals and digital outputs.

Querying "Checkback signals for the cam controller" (S7-300, S7-400)

Definition

The "Checkback signals for the cam controller" function provides information about the current status of the cam controller and track signals. Consistency between the reported position and track signals is not guaranteed.

Function sequence

The data is stored in the channel DB each time the CAM_CTRL or CAM_CTRL_452 instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
23.4	CAM_ACT	BOOL	FALSE	1 = cam processing busy
26.0	ACT_POS	DINT	L#0	Current axis position
30.0	TRACK_OUT	DWORD	DW#16#0	Current track signals of tracks 0 to 31 Bit 0 = track 0

Querying the "Checkback signals for diagnostics" (S7-300, S7-400)

Definition

The "checkback signals for diagnostics" setting is used to report diagnostics events.

Programming steps

1. If the module enters a new event in the diagnostic buffer, it sets the DIAG bit in the feedback interface. Error (Page 601) events are logged in the diagnostics buffer.
2. The module sets the DATA_ERR bit in the feedback interface when it detects incorrect data in a write job. The cause of the error is logged to the diagnostic buffer.
3. The feedback interface stores the CAM_CTRL or CAM_CTRL_452 instruction in the channel DB.
4. The module resets the DIAG bit to 0 after the diagnostic buffer has been read by the CAM_DIAG or CAM_DIAG_452 instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
22.2	DIAG	BOOL	FALSE	1 = diagnostic buffer modified
22.4	DATA_ERR	BOOL	FALSE	1 = data error

2.1.4.3 Configuring FM 452 (S7-300, S7-400)

2.1.4.4 Configuring FM 452 (S7-300, S7-400)

Measuring system (S7-300, S7-400)

Selecting the system of units (S7-300, S7-400)

Selecting the system of units

For the input and output of the data, you can select from the following systems of units:

- Millimeter (default setting)
- Inches
- Degrees
- Pulses

Note

If you change the system of units in the "Measuring system" dialog, the values will be converted to the new system. This may lead to rounding errors.

If you change the system of units via the machine date, the values are not automatically converted.

Any change of the system of units from or to "pulses" disables cam processing and prevents axis synchronization.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
8.0	UNITS	DINT	L#1	**Measuring system** Valid values: • 1 = 10^{-3} mm • 2 = 10^{-4} inch • 3 = 10^{-4} degrees • 4 = 10^{-2} degrees • 5 = pulses • 6 = 10^{-3} degrees

Standard system of units in the online help

To specify the limits, we use the mm system of units. To define the limits in other systems of units, convert the values as shown below:

To convert....		you calculate...
mm → inch		Limit value (inches) = limit value (mm) × 0.1 [1]
mm → degrees	10^{-4} (4 decimal places)	Limit value (degrees) = limit value (mm) x 0.1
	10^{-3} (3 decimal places)	Limit value (degrees) = limit value (mm) x 1
	10^{-2} (2 decimal places)	Limit value(degrees) = limit value (mm) x 10
mm → pulse		Limit value (pulses) = limit value (mm) x 1000

[1] When set to the maximum value, the number of decimal places affects the number of digits before the decimal point. In the inch measurement system, 4 decimal places are used; you can therefore enter a maximum of 100,000.0000 inches. The millimeter measurement system uses 3 decimal places; a maximum of 1,000,000.000 mm can be entered.

Axis (S7-300, S7-400)

Specifying axis type (S7-300, S7-400)

Selecting axis type (S7-300, S7-400)

Axis types

The cam controller supports the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.

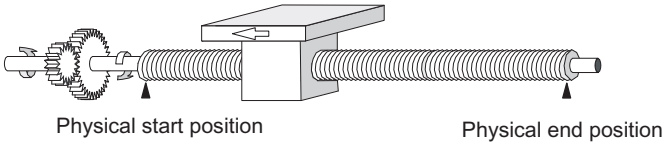

- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.

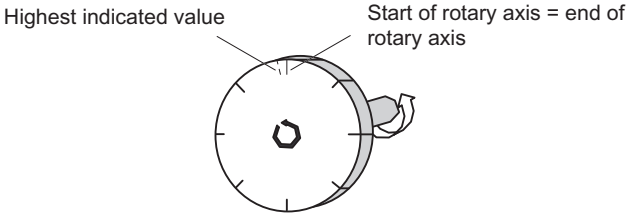

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
12.0	AXIS_TYPE	DINT	L#0	**Axis type** Valid values: • 0 = linear axis • 1 = Rotary axis

Specifying software limit switches (S7-300, S7-400)

Software limit switch

Software limit switches only exist for the linear axis.

The operating range is the range limited by the software limit switches. The FM x52 can monitor the operating range limits.

The software limit switch start (SLS) must always be smaller than the software limit switch end (SLE).

The software limit switches are active when FM x52 is synchronized.

- When using an incremental encoder or initiator:
  The axis is initially not synchronized after each FM x52 startup. The parameterized software limit switches are not monitored until the axis is synchronized.
- When using an absolute encoder (SSI):
  The FM x52 is synchronized once it has received a complete frame without errors. The software limit switches are monitored from this time on. The absolute encoder you use must cover at least the operating range including the software limit switches.

Relationship between the operating range, encoder range and traversing range

- The operating range is the range which you specify for your job using the software limit switches.
- The encoder range represents the range explicitly covered by the encoder. For a linear axis, the module assigns the encoder range symmetrically across across the operating range. In other words, the module shifts the encoder range in order to equalize the distances between the software limit switches and the ends of the encoder range (see figure below).
- This traversing range is the value range the FM x52 can process. The traversing range is dependent on the resolution.

The following applies: Traversing range ≥ encoder range ≥ operating range

The following figure shows the relationship between software limit switches, the operating range, the encoder range and the traversing range.

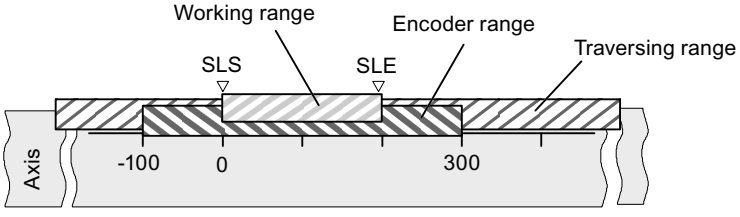

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L# -100 000 000	**Software limit switch start**
68.0	SSW_END	DINT	L# 100 000 000	**Software limit switch end** Valid value range: • -1,000,000,000 µm to 1,000,000,000 µm

Specifying end of rotary axis (S7-300, S7-400)

End of rotary axis

The "End of rotary axis" value is the theoretical maximum actual value of the axis. However, the theoretical maximum value is never indicated, because it also represents the physical start position of the rotary axis (= zero).

The highest rotary axis value displayed is:

End of rotary axis [µm] - resolution [µm / pulse] x 1 [pulse]

Example: End of rotary axis = 1000 mm

The displayed value jumps:

- from 999 mm to 0 mm when the rotational direction is positive.
- from 0 mm to 999 mm when rotational direction is negative.

Rotary axis with absolute encoders

The rotary range (0 to end of rotary axis) of a rotary axis with an absolute encoder must correspond exactly to the total number of steps of the absolute encoder:

$$\text{Rotary axis end[µm]} = \text{Number of revolutions(encoder)} * \frac{\text{steps(encoder)[pulse]}}{\text{Revolution}} * \text{RES}[\frac{\text{µm}}{\text{Pulse}}]$$

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
16.0	ENDROTAX	DINT	L#100000	**End of rotary axis** Valid value range: • 1 µm to +1,000,000,000 µm

Specifying synchronization (S7-300, S7-400)

Selecting "Retrigger reference point" type (S7-300, S7-400)

Retrigger reference point

By setting the "Retrigger reference point (Page 549)" type, you define the conditions for axis synchronization in operation with an incremental encoder or an initiator.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
52.0	RETR_TYPE	DINT	L#0	**Type of reference point retrigger** Valid values: • 0 = reference point switch and zero mark direction + • 1 = reference point switch and zero mark direction - • 6 = reference point switch only • 7 = zero mark only

Specifying reference point coordinate (S7-300, S7-400)

Reference point coordinate

- **Incremental encoder and initiator:**
  Using the "Retrigger reference point" function switch and a synchronization event defined by the "Retrigger reference point" type (Page 563), the reference point coordinate is assigned to this event.
- **Absolute encoder (SSI)**
  A parameterized axis with absolute encoder is always synchronized after transmission of the first error-free SSI frame.
  For more information, refer to the description of absolute encoder adjustment (Page 564) which explains the interaction of the absolute encoder adjustment with other data.

The value of the reference point coordinate must lie within the operating range:

- **Linear axis**
  Including the software limit switches.
- **Rotary axis**
  Greater than or equal to 0 and less than the "end of rotary axis" value (0 ≤ reference point coordinate < end of rotary axis).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1,000,000,000 µm to +1,000,000,000 µm

Specifying absolute encoder adjustment (S7-300, S7-400)

Determining the absolute encoder adjustment (S7-300, S7-400)

Definition

Absolute encoder adjustment is only necessary for an absolute encoder.

The absolute encoder adjustment and reference point coordinate map the encoder's range of values to the axis coordinate system. The encoder value corresponding to the reference point coordinate on the axis is determined. The value must be less than the total number of steps of the absolute encoder.

Additional information can be found under "Absolute encoder adjustment procedure (Page 565)" and in an example (Page 566).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
48.0	ENC_ADJ	DINT	L#0	**Absolute encoder adjustment** Valid value range: • 0 to (2^{25} - 1)

Data used in the channel DB

Address	Name	Type	Initial value	Comment
98.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1,000,000,000 µm to +1,000,000,000 µm

Absolute encoder adjustment procedure (S7-300, S7-400)

Determining the correct absolute encoder adjustment

After initial parameter assignment, further steps must be taken to ensure that a correct relationship is created between the encoder and coordinate system. The procedure shown in the next section is based on the use of the parameter assignment dialogs.

1. Move the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch end".
2. Call the "Set reference point (Page 541)" setting using the coordinate defined in point 1. FM x52 now determines an encoder value for the reference point coordinate entered in the channel DB (REFPT in the channel DB), namely the absolute encoder adjustment. You can read this value with the "Encoder data (Page 556)" setting.
3. Enter the value read in the "absolute encoder adjustment" field.
4. Save your parameter assignment to the corresponding parameter DB using the export function.
5. Download the data in the HWCN to the CPU.
6. Restart the CPU to activate the data.

Note

You perform this adjustment once during commissioning. Following parameter assignment, the FM x52 is synchronized during start-up as soon as a complete, faultless encoder frame is received after start-up.

Otherwise: Mechanical adjustment of the encoder

A correct relationship between the coordinate system and the encoder can be achieved as follows:

1. Traverse the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch start".
2. Enter this coordinate value in the machine data as "reference point coordinate".
3. Read the encoder value displayed at this position with the "Encoder data (Page 556)" setting.
4. Enter this value in the machine data as the "absolute encoder adjustment".

After the parameter assignment a correct actual value will always be shown.

Instead of steps 3 and 4, you can also set the encoder to zero using "Reset", if available. Then enter the value "0" in the machine data as the absolute encoder adjustment.

Example of absolute encoder adjustment (S7-300, S7-400)

Example of an absolute encoder adjustment

The following assumptions apply to the example:

- Reference point coordinate = -125 mm
- Operating range from SSW_STRT = -1000 mm to SSW_END = 1000 mm
- Absolute encoder adjustment = 0
- Encoder range = 2048 increments with a resolution of 1 mm/pulse
- The absolute encoder used cannot be adjusted mechanically with precision and it also is not possible to set the encoder value specifically.

The following figure shows an example of absolute encoder adjustment.

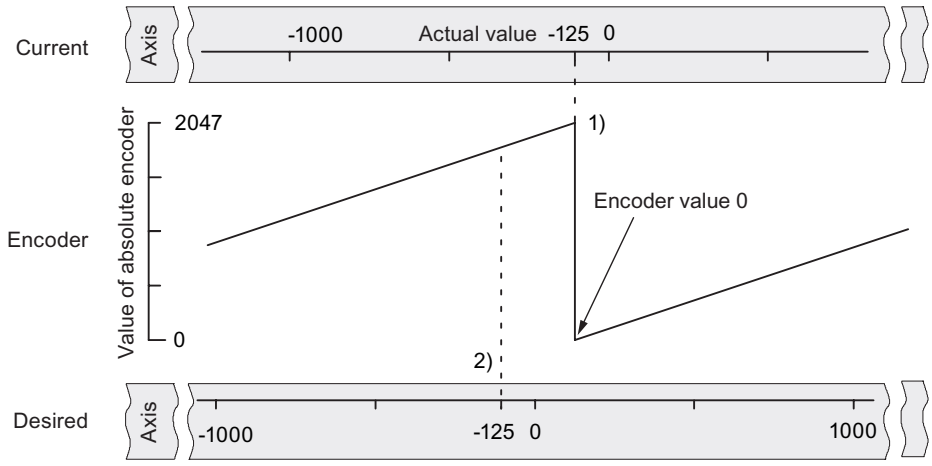

1) Assignment of the coordinate system to encoder values, based on the absolute encoder adjustment. The encoder value 0 corresponds to the actual value -125.

2) Required assignment of the coordinate system to the encoder. At this position the coordinate should be -125.

Result in accordance with "Set reference point"

After "Set reference point" the relationship looks as follows:

The reference point coordinate on the axis (-125) is assigned to the encoder value (1798) determined from absolute encoder adjustment. This assignment is set out in the figure below.

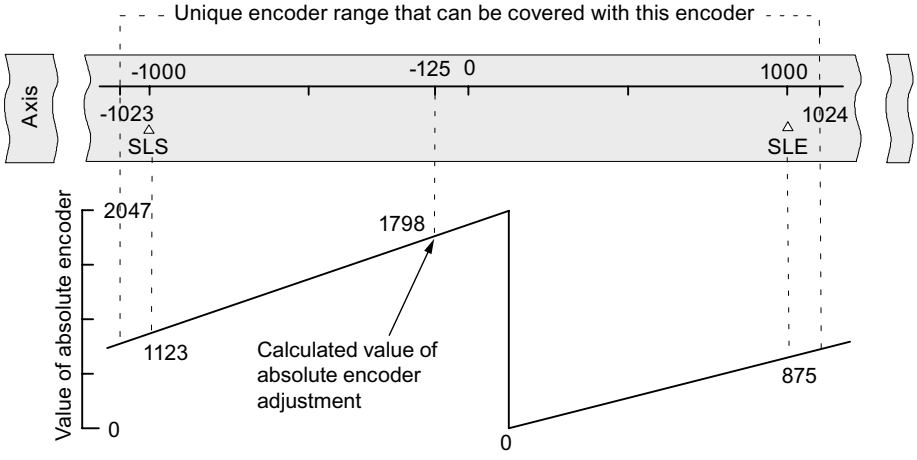

The encoder supplies 2048 unique values. The operating range is defined by the software limit switches. The selected resolution of 1 mm per pulse means the encoder can, however, cover a greater operating range than the one set by the software limit switches.

With the set resolution, the operating range is already covered with 2001 values. Therefore, in the example, 47 pulses "remain" and these are symmetrically positioned around the operating range.

Setting hysteresis (S7-300, S7-400)

Hysteresis

The hysteresis (Page 444) value range depends on the resolution (Page 573).

Maximum input value:

- For linear axes: max. input value < ¼ of the operating range
- For rotary axes: max. input value < ¼ of the rotary axis range

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
80.0	HYS	DINT	L#0	**Hysteresis** Valid value range: • 0 to 65 535 [Imp] x resolution [$^{\mu m}/_{Imp}$]

Position-based cams and hysteresis

A position-based cam is activated when the following conditions are met:

- The detected actual value lies within the position-based cam.
- Hysteresis is not active.

Switching points may vary depending on the position at which the change in direction takes place.

The figure below shows the switching characteristics of a position-based cam with hysteresis.

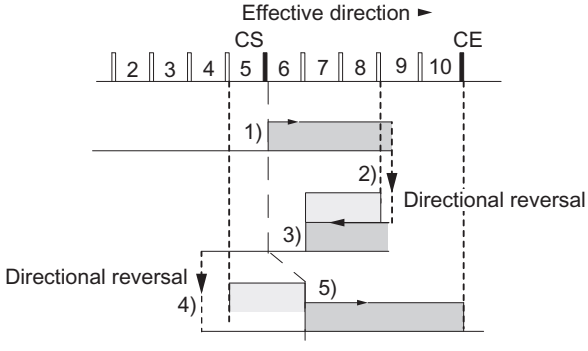

	Cam
	Hysteresis
1)	Cam is activated, as the cam start is crossed in the positive effective direction.
2)	Cam remains active due to hysteresis, irrespective of changes of direction.
3)	Cam is deactivated when hysteresis ends.
4)	Cam remains deactivated after a direction change.
5)	Cam is activated again at the end of hysteresis.

Note

When the direction is changed, hysteresis cannot reduce a cam actuation time shorter than the hysteresis.

Time-based cams and hysteresis

A time-based cam is activated when the following conditions are met:

- The cam start is crossed in the effective direction.
- Hysteresis is not active.

Note

The hysteresis will hide a time-based cam if its range between the reversal point and the cam start is less than the hysteresis.

The figure below shows the switching characteristics of a time-based cam with hysteresis. The figure shows a time-based cam that is **not** activated again.

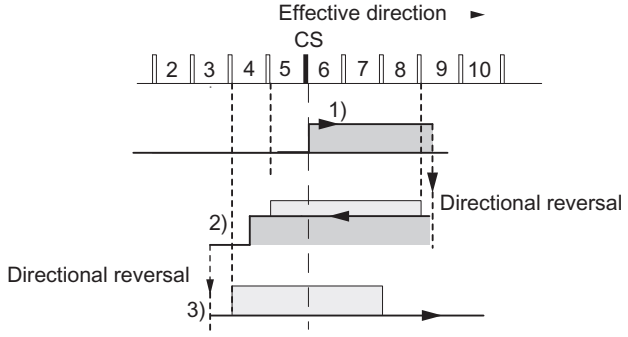

	Cam
	Hysteresis
1)	Cam is activated, as the cam start is crossed in the positive effective direction.
2)	Cam remains active after the change of direction until the activation time has expired, regardless of hysteresis.
3)	Cam start is hidden by the hysteresis; the cam is not activated.

Specifying simulation speed (S7-300, S7-400)

Simulation speed

This machine data determines the simulation speed. Please also see the information under "Influence of settings on the switching characteristics of time-based cams (Page 535)".

The actual simulation speed V_{sim} may deviate from the setting at $V_{sim, V}$, and is calculated using the following formula:

$$V_{Sim} = \frac{6*10exp7*RES}{\text{Integer}\left(\frac{6*10exp7*RES}{V_{Sim, V}}\right)}$$

In this equation:

- V_{sim}: Simulation speed set by FM 352. Unit: $^{\mu m}/_{min}$
- $V_{sim, V}$: simulation speed set in the machine parameters. Unit: $^{\mu m}/_{min}$
- RES: Resolution derived from encoder data. Unit: $^{\mu m}/_{pulse}$
- Integer ( ): Of this expression, only the decimal integer is used in subsequent calculation. This expression must lie within the range of 2 to 65536 in all calculations.

The actual simulation speed changes abruptly as a result of correlations (see above formula).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	SIM_SPD	DINT	L#0	**Simulation speed** Valid value range: • 0 = standstill • 5×10^8 = highest setting supported by the module Within this range, the simulation speed depends on the resolution (Page 573): 1000 x resolution ≤ simulation velocity ≤ 3 x 10^7 x resolution

Specifying minimum edge-to-edge distance (S7-300, S7-400)

Minimum edge-to-edge distance

This machine data is used to define a range after detection of the start of a measurement for edge detection. The measurement is discarded if the end of the measurement is within this range.

The start of measurement is not reported unless the "minimum edge-to-edge distance" has been traveled.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	**Minimum edge-to-edge distance** Valid value range: • 0 ... 1,000,000,000 µm

Encoder (S7-300, S7-400)

Selecting the signal type (S7-300, S7-400)

Signal type

The encoder returns position data to the module for evaluation and conversion to an process value based on the resolution.

You can select the signal type from one of the following encoders (Page 447):

- 5 V differential signal (Page 449)
  Select the signal type "5 V differential signal" to use a 5 V incremental encoder with differential signal A, /A, B, /B and N, /N.
- 24 V track A+B phase shifted (Page 449)
  Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.
- SSI absolute (Page 451)
  Select the "SSI absolute" signal type if you use an absolute encoder with a serial interface.
- 24 V initiator (Page 451)
  Select the signal type "24 V initiator" to use a 24 V initiator with count signal A*.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
20.0	ENC_TYPE	DINT	L#1	**Signal type** Valid values: • 1 = 5 V, incremental • 2 = 24 V, incremental • 3 = SSI 13-bit frame length • 4 = SSI 25-bit frame length • 5 = listen in • 6 = 24 V initiator up (positive) • 7 = 24 V initiator down (negative) • 8 = SSI 13-bit (right-justified) • 9 = SSI 25-bit (right-justified) • 10 = listen in (right-justified)

Specifying resolution (S7-300, S7-400)

Specify distance per encoder revolution (S7-300, S7-400)

Distance per encoder revolution

You inform the FM x52 about the distance covered by the drive system per encoder revolution using the "Distance per encoder revolution" machine data.

The "Distance per encoder revolution" value depends on the structure of the axis and the way in which the encoder is mounted. You must take all transmission elements, such as couplings and gears, into account. The following figure shows an example of two different encoder positions.

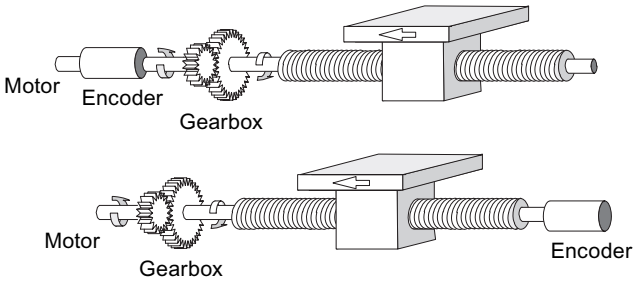

The relationship between the "Distance per encoder revolution" and "Increments per encoder revolution" machine data is described under resolution (Page 573).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
24.0	DISP_REV	DINT	L#80000	**Distance per encoder revolution** Valid value range: • 1 µm to 1,000,000,000 µm

Specifying increments per encoder revolution (S7-300, S7-400)

Increments per encoder revolution

The "Increments per encoder revolution" machine data defines the number of increments an encoder returns per revolution. The FM x52 calculates the resolution based on this value and the "Distance per encoder revolution" machine data.

Incremental encoder

Any value within the range of values can be entered (Page 449). The module evaluates the increments four times.

Initiator

Any value within the range of values is allowed.

Absolute encoder

Different encoder types have different limits:

Encoder type	Frame length / type	Value range	Can be used as linear axis
Single-turn encoder	13-bit half fir tree	64 ... 8192 in powers of 2	
Single-turn encoder	13-bit right-justified	64 ... 8192 all values	X
Single-turn encoder	25-bit right-justified	64 ... 2^{25} all values	X
Multiturn encoder	25-bit fir tree	64 ... 8192 in powers of 2	
Multiturn encoder	25-bit right-justified	64 ... 2^{24} all values	
Listen-in	Fir tree	64 ... 8192 in powers of 2	
Listen-in	Right justified	64 ... 2^{25} all values	X
Special setting: Multiturn encoder in single-turn mode	25-bit half fir tree	64 ... 8192 in powers of 2	

Note

The number of encoder pulses is the product of the "Increments per encoder revolution" and the "Number of revolutions" (see "Resolution (Page 573)").

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
32.0	INC_REV	DINT	L#500	**Increments per encoder revolution** Valid value range: • 1 to 2^{25} Note: This entry is irrelevant when you use "pulse" units.

Resolution (S7-300, S7-400)

Resolution

The resolution reflects the accuracy of cam processing. It also determines the maximum possible traversing range.

The resolution (RES) is calculated as shown in the table below:

	Incremental encoder	Absolute encoder / initiator
Input values	• Distance per encoder revolution • Increments per encoder revolution: – Pulse evaluation: quadruple – 1 increment = 4 pulses	• Distance per encoder revolution • Increments per encoder revolution. – 1 increment = 1 pulse
Calculation	$RES = \frac{\frac{Distance}{Encoder\ revolution}}{\frac{Pulses}{Encoder\ revolution}}$	

Note

Resolution in the "pulse" system of units is always 1.

All position values are rounded to the integer multiple of the resolution. This allows you to distinguish between the set and used values.

Range of values of the resolution

Convert the range of values for the resolution according to the defined physical units. The resolution must be kept within this range when setting the "distance per encoder revolution" and "increments per encoder revolution" values.

The chosen system of units determines the resolution value range:

Measuring system	Representation in...	Range of values of the resolution
mm	10^{-3} mm	0,1 × 10^{-3} mm to 1000 x 10^{-3} mm/pulse
inch	10^{-4} inch	0,1 × 10^{-4} inches to 1000 x 10^{-4} inches/pulse
degrees	10^{-4} degrees 10^{-3} degrees 10^{-2} degrees	0,1 × 10^{-4} degrees to 1000 x 10^{-4} degrees/pulse 0,1 × 10^{-3} degrees to 1000 x 10^{-3} degrees/pulse 0,1 × 10^{-2} degrees to 1000 x 10^{-2} degrees/pulse
Pulses	1 pulse	1

Example

- An incremental encoder has the following data:
  - Increments per encoder revolution: 5000
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 4 pulses

  Resultant resolution (quadruple evaluation):

$$\text{Resolution} = \frac{1000\ \text{mm}}{5000\ \text{increments}} = 0.2000\ \frac{\text{mm}}{\text{Increment}} = 0.2000\ \frac{\text{mm}}{4\ \text{pulses}}$$

$$= 0.0500\ \frac{\text{mm}}{\text{Pulse}}$$

- An SSI encoder has the following data:
  - Increments per encoder revolution: 4096
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 1 pulse

  Resultant resolution:

$$\text{Resolution} = \frac{1000\ \text{mm}}{4096\ \text{increments}} = 0.2441\ \frac{\text{mm}}{\text{Increment}} = 0.2441\ \frac{\text{mm}}{\text{Pulse}}$$

Traversing range dependency on resolution

The traversing range is restricted by the numerical representation in FM x52. This numerical representation varies depending on the resolution. You should thus make sure that your specifications are always within the permitted limits.

The table below sets out the maximum traversing range:

Resolution (RES) is in the range	Maximum traversing range
0.1 µm/pulse ≦ RES < 1 µm/pulse	-10^8 µm to 10^8 µm (-100 m to +100 m)
1 µm/pulse ≦ RES ≦ 1000 µm/pulse	-10^9 µm to 10^9 µm (-1000 m to +1000 m)

Dependency between the velocity and resolution

The velocity indicated may vary within the following limits, depending on the resolution (data relate to mm units):

- from 1 µm/min to 90 m/min, at a resolution < 1 µm/pulse
- from 1 µm/min to 900 m/min, at a resolution ≥ 1 µm/pulse

The velocity value is calculated and filtered by the module at intervals of 4 ms.

Its minimum inaccuracy of one pulse/4 ms rules it out for closed-loop control.

Selecting the count direction (S7-300, S7-400)

Count direction

You can adapt the position feedback detection to the direction of axis motion using the "Count direction" machine data. You must also take into account the rotational direction of all transmission elements (for example, couplings and gears).

- Normal = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent increasing positional actual values
- Inverted = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent decreasing positional actual values

Note:
You cannot implement a rate time in combination with an absolute encoder (SSI) and inverted parameterized count direction.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
59.0	CNT_DIR	BOOL	FALSE	**Count direction** Valid values: • 0 = normal • 1 = inverted

Selecting monitoring (S7-300, S7-400)

Monitoring

Wire break

When its monitoring function is enabled, the FM x52 monitors the A, /A, B, /B, N, and /N signals of an incremental encoder. The monitoring function detects:

- Wire break
- Short circuits in individual cables
  For incremental encoders without a zero mark, you must either:
  - Disable the wire-break monitoring function
  - Interconnect the N and /N signals externally (Page 447)
- Edge-to-edge distance of the count pulses
- Encoder supply failure

Frame error

You cannot switch off the monitoring of frame errors with absolute encoders (SSI). It monitors:

- the frame for start and stop bit errors
- the monoflop time of the connected encoder

Missing pulses (incremental encoder)

An incremental encoder must always return the same number of increments between two successive zero marks.

FM x52 checks whether the zero mark of an incremental encoder coincides with the correct encoder value.

Disable missing pulse monitoring for encoders without zero marks.

Also disable wire-break monitoring, or interconnect the zero mark inputs N and /N externally.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
				Monitoring functions
63.0	MON_WIRE	BOOL	TRUE	1 = wire break
63.1	MON_FRAME	BOOL	TRUE	1 = frame error (must always be 1)
63.2	MON_PULSE	BOOL	TRUE	1 = missing pulses

Selecting data for absolute encoder (SSI) (S7-300, S7-400)

Number of encoder revolutions (S7-300, S7-400)

Number of encoder revolutions

The machine data "Number of encoder revolutions" only applies to absolute encoders. You use it to define the maximum possible number of revolutions for this encoder. For additional information on this topic, please refer to "Absolute encoders (SSI) (Page 451)".

- Single-turn encoder
  Only the value 1 is possible.
- Multiturn encoder
  - Multiturn encoder / listen-in (fir tree): 2 ... 4096 in powers of 2
  - Multiturn encoder / listen-in (right-justified): 1 ... 4096 all values
- Linear scale
  You can also connect a linear scale. To do so, enter the value "1".

The total number of steps of the encoder is not machine data. It is calculated as follows:

Overall number of steps = increments per encoder revolution x number of revolutions

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
36.0	NO_REV	DINT	L#1024	**Number of encoder revolutions** Valid value range: • 1 (single-turn encoder) • 1 to 4096 (multiturn encoder)

Transmission rate (S7-300, S7-400)

Transmission rate

You define the data transmission rate between the SSI encoder and the FM x52 with the "transmission rate" machine data.

The maximum transmission rate depends on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 63 m → 500 kHz
- 20 m → 1,000 kHz

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
40.0	BAUDRATE	DINT	L#0	**Transmission rate** Valid values: • 0 = 125 kHz • 1 = 250 kHz • 2 = 500 kHz • 3 = 1000 kHz

Frame length (S7-300, S7-400)

Frame length

You use "Frame length" to define the clock frame output by the FM x52 for absolute encoders.

Selecting "listen-in" will disable the FM x52 cycle. The FM x52 can then listen in on other SSI frames with a 13 or 25-bit frame. The transmission rate is determined by the cycle clock rate of the master module.

Data used in the parameter DB

The frame length selected here is stored in the parameter DB under the ENC_TYPE (Page 571) date in the parameter DB.

Initiator direction (S7-300, S7-400)

Initiator direction

You define the direction of a connected initiator with "Initiator direction":

- Positive: = up (incremental count pulses)
- Negative: = down (downwards count pulses)

Tracks (S7-300, S7-400)

Selecting track output control mode (S7-300, S7-400)

Track output control

The "Track output control" machine data defines the control mode for track outputs 0 to 15. Tracks can be activated by:

- The cam controller: Track signals are activated and deactivated by FM 452 cam processing.
- CPU: The track signals represent the corresponding values of the track enables in the channel DB.

This allows the explicit activation of track outputs in the user program.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
90.0	TRACK_OUT	WORD	W#16#0	**Track output control** Valid values: • 0 = cam controller • 1 = CPU Bit number = track number

Selecting special tracks (S7-300, S7-400)

Special tracks

You can parameterize tracks 0, 1 and 2 for use as special tracks.

- Track 0 and track 1 as counter cam track (Page 442)
- Track 2 as brake cam track (Page 442)

Data used in the parameter DB

Address	Name	Type	Start value	Comment
				Special tracks
99.0	SPEC_TRC0	BOOL	FALSE	1 = track 0 is counter cam track
99.1	SPEC_TRC1	BOOL	FALSE	1 = track 1 is counter cam track
99.2	SPEC_TRC2	BOOL	FALSE	1 = track 2 is brake cam track

Specifying high count value for counter cam track (S7-300, S7-400)

High count value for counter cam track

With this machine data, you define the high count value of the assigned counter cam track.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CNT_LIM0	DINT	L#2	**High count value of counter cam track** (track 0)
104.0	CNT_LIM1	DINT	L#2	**High count value of counter cam track** (track 1) Valid value range: • 2 … 65535

Selecting enable input (S7-300, S7-400)

Enable input

Track signal Q3 to Q10 is activated if the following conditions are met:

- The track is enabled.
- The relevant external enable input I3 to I10 has been set.
- The track result is 1.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
				Enable input
95.0	EN_IN_I3	BOOL	FALSE	1 = track signal track 3 AND enable input I3
…	…	…	…	…
95.7	EN_IN_I10	BOOL	FALSE	1 = track signal track 10 AND enable input I10

Cams (S7-300, S7-400)

Number of cams (S7-300, S7-400)

Number of cams

The number of cams determines the maximum number of programmable cams and therefore the cam cycle time.

Number of cams	Cam cycle time
16 cams	20.48 µs
32 cams	40.96 µs
64 cams	81.92 µs
128 cams	163.84 µs

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	C_QTY	DINT	UDT3: L#0 UDT4: L#1 UDT5: L#2 UDT6: L#3	Number of cams 0 = max. 16 cams 1 = max. 32 cams 2 = max. 64 cams 3 = max. 128 cams

Cam data (S7-300, S7-400)

Cam data (S7-300, S7-400)

Cam data

Cam data describe the properties of a cam, the assignment of each cam to a track, and the switching characteristic of the cam.

The cam data below is set separately for each cam:

- Validity (Page 582)
- Track number (Page 582)
- Cam type (Page 583)
- Cam start and cam end of position-based cams (Page 583)
- Cam start and activation time of time-based cams (Page 584)
- Lead time (Page 585)
- Effective direction (Page 587)
- Hardware interrupt (Page 589)

The following applies to cam processing in general:

- The module interprets and processes only the cams with "valid" settings.
- Cams 0 to 7 support hardware interrupts.
- The number of assignable cams depends on the number of cams available.

Cam validity (S7-300, S7-400)

Cam "valid"

All cams to be processed by the cam controller must be set as "valid".

Note

Unused cams should always be set to "invalid" CAMVALID = FALSE).

Data used in the parameter DB

Address relative	Name	Type	Initial value	Comment
+0.0	CAMVALID	BOOL	FALSE	1 = cam "valid"

Specifying a cam track number (S7-300, S7-400)

Track number

You define the track controlled by the cam by setting the track number.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+1.0	TRACK_NO	BYTE	B#16#0	**Track number** Valid value range: • 0 to 31

Selecting cam type (S7-300, S7-400)

Cam type

Select the required cam type.

- Position-based cam
- Time-based cam

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.3	CAM_TYPE	BOOL	FALSE	**Cam type** Valid values: • 0 = position-based cam • 1 = time-based cam

Defining cam start and cam end for position-based cams (S7-300, S7-400)

Cam start (CS) / Cam end (CE) of position-based cams

The cam start and cam end of a position-based cam define the cam length.

Minimum length of a position-based cam

The figure below shows the shortest cam with axis motion in positive direction

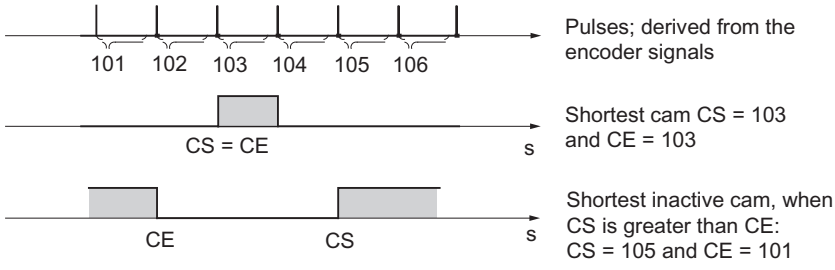

The following general rules apply:

- The inactive part of a cam must always have a minimum interval of 4 pulses between the cam end (CE) and cam start (CS).
- If CE = CS, the cam is activated for the duration of one pulse.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment Of position-based cams
+2.0 +6.0	CBEGIN CEND	DINT DINT	L#-100000000 L#100000000	**Cam start (CS)** **Cam end (CE)** Valid value range: • -1,000,000,000 µm to 1,000,000,000 µm

Specifying cam start and activation time for time-based cams (S7-300, S7-400)

Cam start (CS) / Cam end (CE) ≙ activation time of time-based cams

With a time-based cam, you must specify a cam start and an activation time in place of the cam end. Define the activation time with a resolution of 100 µs. The time runs starting with the activation of the cam.

Conditions of setting default times:

- 0 µs: A cam with an activation time of 0 µs will never be activated
- 0 µs < t ≤ 400 µs: The FM x52 sets a minimum cam activation time of approx. 330 µs.
- t > 400 µs: FM x52 calculates the actual activation time t_{act} based on the default activation time t_{def} using this formula:

$$t_{act} = \text{integer}\left(\frac{t_{def}}{\text{Cam cycle time}}\right) * \text{Cam cycle time}$$

The maximum error is always less than the cam cycle time.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment of time cams
+2.0 +6.0	CBEGIN CEND	DINT DINT	L#-100000000 L#100000000	**Cam start (CS)** **Cam end (CE) ≙ activation time** Valid cam start value range: • -1,000,000,000 µm to 1,000,000,000 µm Valid cam activation time value range: • (0 to 13421) x 100 µs with up to 16 cams • (0 to 26843) x 100 µs with up to 32 cams • (0 to 53686) x 100 µs with up to 64 cams • (0 to 65535) x 100 µs with up to 128 cams

Specifying lead time (S7-300, S7-400)

Lead time

You can compensate any delays caused by the connected switchgear by setting a lead time. Define a lead time with a resolution of 100 µs. You can assign one lead time to each cam. The lead time applies to the cam start and cam end.

Actuation distance

The actuation distance of a cam is calculated continuously based on the current velocity and lead time. The entire cam is shifted in direction of the actual value by this value. The assigned range is the "static range", and the range calculated based on the lead time represents the "dynamic range".

Actuation distance = lead time x current velocity

FM x52 calculates the actuation distance of all cams within ¼ of the longest parameterized lead time. A very high lead time setting for a cam reduces the frequency of the dynamic adjustment calculation.

Actual lead time

To calculate the actual lead time:

1. Determine the cam cycle time: This is the time in which the FM x52 fully processes all cams. It depends on the number of cams parameterized.
2. Calculate the actual lead time based on the following formula:

$$\text{Lead time}_{act} = \text{integer}\left(\frac{\text{Lead time}_{d}}{\text{Cam cycle time} * 4}\right) * \text{Cam cycle time} * 4$$

   The identifiers have the following meanings:

   – Lead time $_{act}$ is the lead time set by the FM x52
   – Lead time $_{d}$ is your default setting.
   – Integer ( ) means that the values after the decimal point are ignored in the calculation of the parenthesis.

The maximum error of the lead time$_{act}$ is always less than [cam cycle time] x 4.

Example:

The following values are defined:

- Number of cams: Maximum 32 cams
- Cam cycle time: 40.96 µs
- Lead time$_{d}$ = 1000 µs

Result: An actual lead time of 983 µs.

Note

The actual is always less than the assigned lead time. It may be 0 even though the parameterized lead time ≥100 µs.

The actuation distance of a rotary axis must be less than the rotary axis range and the inactive part of the cam. This must be guaranteed for all velocities.

Note

You may not implement a lead time in combination with an absolute encoder (SSI) and inverted parameterized count direction.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+ 10.0	LTIME	INT	0	**Lead time** Valid value range: • (0 to 53686) x 100 µs with max. 16 cams • (0 to 65535) × 100 µs with max. 32, 64 or 128 cams

Dynamic cam adjustment

There are two distinct situations relating to the cam range:

1. The static and dynamic range of the cam overlap.
2. The static and dynamic range of the cam do not overlap.

The table below shows the calculation of dynamic adjustment in each of the two cases.

Dynamic adjustment	Description
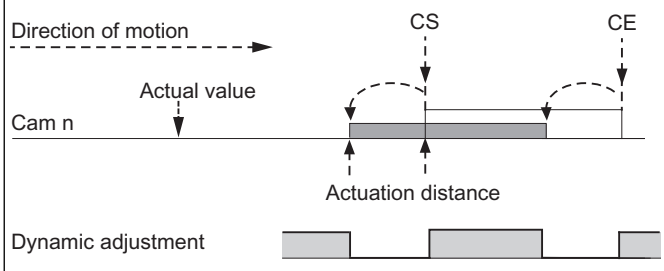	If the dynamic range of the cam overlaps its static range: • The cam is activated when its dynamic range is reached. At the same time, calculation of a new dynamic adjustment is disabled. • When the actual value reaches the static range of the cam, the calculation of a new dynamic adjustment is enabled again. A change in velocity affects the cam end. • When the cam is deactivated at the end of the dynamic range, dynamic adjustment will be disabled again until the end of the static range of the cam.
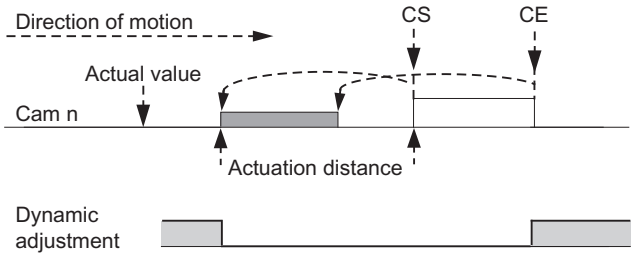	The following applies if the cam's dynamic and static ranges do not overlap: • The cam is activated when its dynamic range is reached. At the same time the calculation of a new dynamic adjustment is disabled. • At the end of the static range of the cam, dynamic adjustment is enabled again.
Dynamic range Static range A **new** dynamic adjustment is possible	

Note

When the direction of rotation changes, calculation of the dynamic adjustment is enabled again.

Selecting effective direction (S7-300, S7-400)

Effective direction

Two effective directions are supported:

- Positive: The cam is activated at the cam start if the axis is moving in the direction of increasing process values.
- Negative: The cam is activated at the cam end if the axis is moving in the direction of decreasing process values.

You may set both effective directions in parallel.

Selecting "None" disables the cam.

Switching characteristics of cams based on their effective direction

The table below shows the switching characteristics of position-based and time-based cams as determined by the effective direction. A positive effective direction is always assumed, with the exception of example 5.

No.	Description	Position-based cams	Time-based cams
1	A cam is overrun in the effective direction	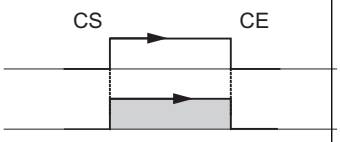	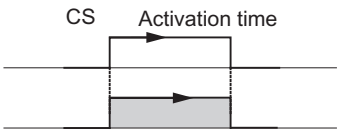
2	A cam is traversed in reversed effective direction	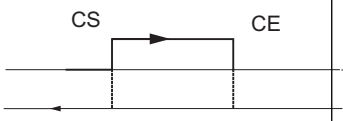	
3	A cam is approached in the effective direction; the direction of motion of the axis is reversed while the cam is active	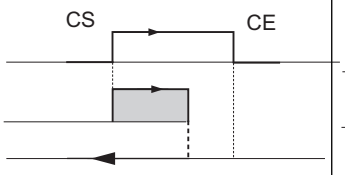	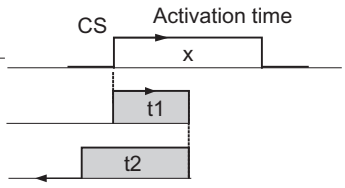 Activation time x = t1 + t2
4	A cam is approached in reversed effective direction; the axis reverses its direction of movement on the cam in effective direction	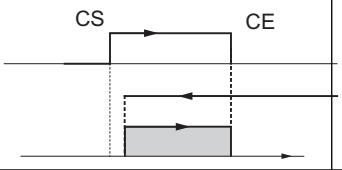	Cam not switched
5	A cam is approached from any direction, and exited into any direction; **both directions** are set as effective direction	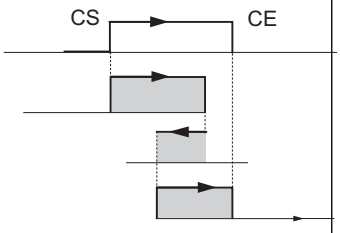	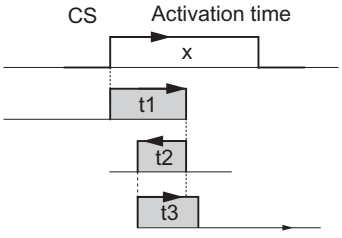 Activation time x = t1 + t2 + t3

parameterized cam
switched cam

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.1	EFFDIR_P	BOOL	TRUE	1 = positive effective direction (plus)
+0.2	EFFDIR_M	BOOL	TRUE	1 = negative effective direction (minus)

Selecting hardware interrupt for cam (S7-300, S7-400)

Hardware interrupt when cam switches on / off

You can specify whether hardware interrupts are to be generated when cams 0 to 7 are enabled, disabled or enabled and disabled.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.4	PI_SW_ON	BOOL	FALSE	1 = hardware interrupt on activation
+0.5	PI_SW_OFF	BOOL	FALSE	1 = hardware interrupt on deactivation

Enable hardware interrupts (S7-300, S7-400)

Enabling hardware interrupts (S7-300, S7-400)

Enabling hardware interrupts

Hardware interrupts are enabled in the machine data.

You can set when hardware interrupts are to be generated:

- Start of measurement
  A rising edge at digital input I1 can trigger a hardware interrupt in the "Edge detection (Page 546)" setting.
- End of measurement
  With both the "Edge detection (Page 546)" and "Length measurement (Page 546)" setting, a falling edge at digital input I1 can trigger a hardware interrupt.
- Cam on/off
  In the cam data (Page 589), you can specify whether hardware interrupts are to be generated when cams 0 to 7 are activated and/or deactivated.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
3.1	PI_MEND	BOOL	FALSE	1 = enable process interrupt: End of measurement
3.2	PI_CAM	BOOL	FALSE	1 = enable process interrupt: Cam on/off
3.5	PI_MSTRT	BOOL	FALSE	1 = enable process interrupt: Start of measurement

Evaluation of a hardware interrupt (S7-300, S7-400)

If FM 452 triggers a hardware interrupt, interrupt information is available in the OB4x_POINT_ADDR tags of the hardware interrupt OB.

The following table shows the content of the double word OB4x_POINT_ADDR.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	0	0
1	0	0	Start of measure-ment	0	0	Cams	End of measure-ment	0
2	Cam 7 on	Cam 7 off	Cam 6 on	Cam 6 off	Cam 5 on	Cam 5 off	Cam 4 on	Cam 4 off
3	Cam 3 on	Cam 3 off	Cam 2 on	Cam 2 off	Cam 1 on	Cam 1 off	Cam 0 on	Cam 0 off

Byte 1 identifies the cause of the interrupt:

- Cam:
  Evaluate bytes 2 and 3 according to the table.
- Start / end of measurement:
  Use the CAM_MSRM_452 instruction to read the current measured values from the module.

Lost hardware interrupts

If the hardware interrupt OB is still busy processing a hardware interrupt, the module registers all subsequent hardware interrupt events. If an event occurs again before the hardware interrupt could be triggered, the module triggers a "hardware interrupt lost" diagnostic interrupt.

Control and feedback interface of the FM 452 (S7-300, S7-400)

Application

Very fast access to checkback and control signals is required in special applications or in an alarm level. You can obtain those data directly by reading the IO areas of the module.

To coordinate startup following each module startup, for example, after plugging a module or after a STOP-RUN transition of the CPU, you must call the CAM_CTRL instruction until the end of the startup is indicated by RETVAL = 0.

Note

For direct access to FM 452 data, always use the non-internal data and method described in this section. Otherwise, your user program will encounter difficulties accessing the module.

Reading checkback signals via direct access

The byte addresses are defined with an offset to the output address of the module. The names of the parameters correspond to the names in the channel DB.

In STL, you access the data using the PIB (read 1 byte) and PID (read 4 bytes) commands.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	PARA	Internal	Internal	DATA_ERR	Internal	DIAG	Internal	Internal
Byte 1	0	0	0	CAM_ACT	0	0	0	0
Byte 2	Internal							
Byte 3	0	0	FVAL_DONE	HYS	GO_P	GO_M	MSR_DONE	SYNC
Byte 4	ACT_POS							
Byte 5								
Byte 6								
Byte 7								
Byte 8	TRACK_OUT							
Byte 9								
Byte 10								
Byte 11								

Direct write access to control signals

The byte addresses are defined with an offset to the input address of the module. The names of the parameters correspond to the names in the channel DB.

In STL, you access the data using the PQB (write 1 byte) and PQW (write 2 bytes) commands.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	Internal							
Byte 1	0	CNTC1_EN	CNTC0_EN	CAM_EN	DIR_P	DIR_M	0	0
Byte 2	TRACK_EN							
Byte 3								

Example: Position process value ACT_POS

The start address of the module is 512.

```
STL
L PID 516      //Read the position process value (ACT_POS) with direct access:
               //Module start address + 4
```

Parameter transmission paths (S7-300, S7-400)

The term parameter refers to the following machine and cam data.

The following figure shows the parameter transmission paths of the FM 452.

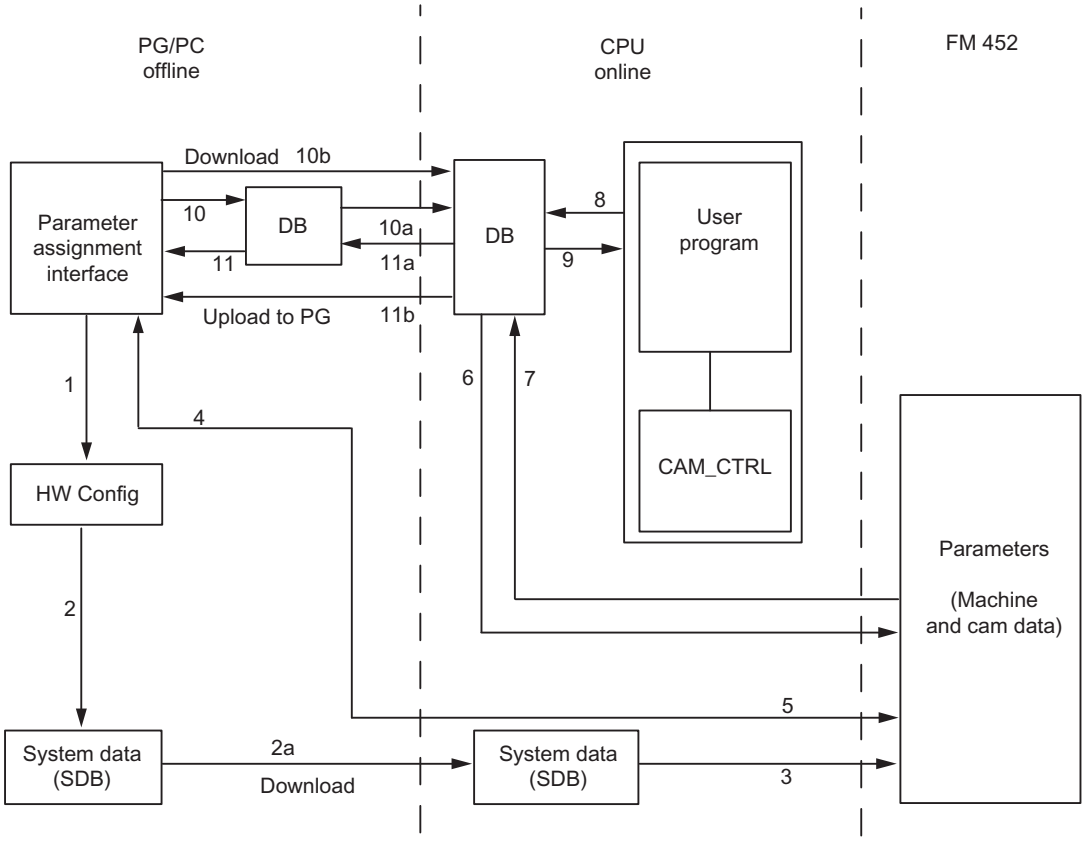

1 Save the parameters in the parameter assignment interface.

2 Saving and compiling the HW configuration, and download to the CPU.

3 The CPU writes the parameters to the module during system parameter assignment.

4 Upload the module parameters to the PG by selecting the "PLC > Upload to PG" command.

5 Download the parameters from the parameter assignment interface to the module with the "PLC > Download" command.

6 Write parameters to the module using jobs in the user program.

7 Read parameters from the module using jobs in the user program.

8 Save parameters from the user program to the online DB.

9 Read in parameters from the online DB to the user program.

10 Export parameters from the parameter assignment interface to the DB (offline or online DB); an offline DB must then be downloaded to the CPU.

11 Import parameters from an online or offline DB into the parameter assignment interface.

Some use cases for the transfer of parameters

Use case	Steps
You edit the parameters on the parameter assignment interface. The parameter should then be assigned automatically to the module during startup.	Perform steps 1, 2, and 3.
You edit parameters in the parameter configuration interface during commissioning in test mode.	Perform steps 4 and 5

Use case	Steps
The parameters edited during commissioning should be downloaded automatically during startup.	Perform steps 1, 2, and 3.
Create the parameters using the parameter assignment interface. When it starts up, the module should be assigned parameters only by the user program using data blocks.	Perform steps 10 and 6.
You need a comfortable means of creating data records for recipes.	Perform step 10.
Create the parameters using the parameter assignment interface. These should be available to the user program for temporary modifications.	Perform steps 1, 2, and 3 for automatic parameter assignment.
	Perform steps 10, 7 to enable access by the user program.
You modify existing parameters (exclusively) using the user program.	Perform steps 7, 9, 8, and 6.
You want to view the data modified by the user program on the parameter assignment interface.	Perform step 11.
Parameters modified by the user program should also be loaded automatically during startup.	Perform steps 6, 11, 1, 2, and 3.

2.1.4.5 Commissioning the FM 452 (S7-300, S7-400)

2.1.4.6 Using FM 452 from the user program (S7-300, S7-400)

Integrating the FM 452 into a user program

Instructions are available for integrating the FM 452 into a user program. These enable you to handle the desired functions with ease.

You call the CAM_INIT and CAM_CTRL or CAM_CTRL_452 instructions in the CPU user program. They ensure communication between the CPU and the FM 452.

The channel DB is created automatically as the instance DB for the CAM_CTRL instruction when you insert the CAM_CTRL instruction into the project.
When you insert the CAM_CTRL_452 instruction into the project, use the command "Program blocks > Add new block" to add a CAM_CHANTYPE type data block to your project as the channel DB.

All the parameter data associated with the channel of the module (machine data, cam data) is located in the parameter DB for the module. Use the command "Program blocks > Add new block" to add a CAM_PxxxTYPE type data block to your project. Depending on the quantity structure, you need the following DB types:

- Up to 16 cams: CAM_P016TYPE
- Up to 32 cams: CAM_P032TYPE
- Up to 64 cams: CAM_P064TYPE
- Up to 128 cams: CAM_P128TYPE

The instruction CAM_MSRM_452 allows you to evaluate data from FM 452 length measurement or edge detection immediately in the hardware interrupt OB.

The CAM_DIAG or CAM_DIAG_452 instruction is also available for FM 452. This instruction lets you transfer the diagnostic information to the diagnostic DB.

The diagnostic DB is created automatically as the instance DB for the CAM_DIAG instruction when you insert the CAM_DIAG instruction into the project.
When you insert the CAM_DIAG_452 instruction into the project, use the command "Program blocks > Add new block" to add a CAM_DIAGTYPE type data block to your project as the diagnostic DB.

Commissioning the module

The module can be initially tested using the "Commissioning" dialog, even when no user program is available. Note the following warning in particular in this case.

WARNING

Injury and material damage could occur.

To prevent personal injury and material damage, observe the following points:

- Install an emergency stop switch in the vicinity of the computer.
  This is the only way you can ensure that the plant can be safely shut down in the event of computer or software failure.
- Install hardware limit switches, which act directly on the power units of all drives.
- Make sure that no one has access to the area of the plant in which moving parts are present.
- When the FM 452 is controlled and monitored simultaneously by your program and the "Commissioning" dialog, conflicts can occur, the effects of which are not clear. Therefore, always set the CPU to STOP mode or disable your program when you work with the "Commissioning" dialog.

2.1.4.7 FM 452: Online & Diagnostics (S7-300, S7-400)

Current module information (S7-300, S7-400)

Default diagnostics (S7-300, S7-400)

Error display through error LEDs of FM 452 (S7-300, S7-400)

Where is the error displayed?

The red error LEDs on the module display different error states. The LEDs are lit, even with errors that occur briefly, for at least 3 seconds.

The following figure shows the location of the error LEDs on FM 452.

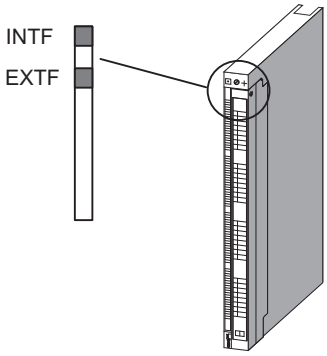

Which errors are displayed?

The table below shows the error types indicated by the error LEDs.

LED	Meaning	Explanations
INTF	Group error for internal error	This LED indicates the following error states of FM 452: • Hardware interrupt lost • Watchdog expired • FM 452 not configured • FM 452 incorrectly configured To remove the error, see Error classes (Page 601).
EXTF	Group error for external error	This LED indicates the following error states: • No external 24 V auxiliary supply • Front connector missing • Encoder wire break • Operating error • Absolute encoder frame error • Incremental encoder pulse missing or zero mark signal missing
INTF + EXTF	Module defective	All outputs are disabled. The module must be replaced.

Channel-specific diagnostics (S7-300, S7-400)

Triggering diagnostic interrupts (S7-300, S7-400)

Interrupt processing

The FM 452 can trigger hardware and diagnostic interrupts. You process these interrupts in an interrupt OB. If an interrupt is triggered without the corresponding OB being loaded, the CPU switches to STOP.

You enable the processing of diagnostic interrupts as follows:

1. Select the module in HWCN
2. Select "Edit" > Object Properties > Basic Parameters" to enable the diagnostic interrupt.
3. Save and compile the configuration.
4. Download the configuration to the CPU.

Overview of diagnostic interrupts

Events and errors which trigger a diagnostic interrupt:

- Operating error
- Machine data error
- Incorrect cam data
- Diagnostic errors

These errors are explained in detail in "Error classes (Page 601)".

Module response to an error with diagnostic interrupt

- Cam processing will be disabled.
- The synchronization will be cleared by the following diagnostic interrupts:
  - Front connector missing, no external auxiliary voltage
  - Zero mark error detected, cable fault (5 V encoder signals)
  - Traversing range exceeded (indicated by an operating error)
  - Set process value cannot be executed (indicated by an operating error).
- With one exception, control signals are no longer processed.
  Exception:
  When the software limit switches are overrun, a change of direction is still possible in simulation mode.
- Function switch and job processing continues.

Module detects an error ("incoming")

A diagnostic interrupt is an "incoming" event if at least one error is pending. If any errors are pending clearance, the queued errors are reported again as "incoming" events.

Sequence:

1. The FM 452 detects one or more errors and triggers a diagnostic interrupt. The error LED lights up. The error event is logged to the diagnostic buffer.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. You can now evaluate the start information of OB82.
4. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
5. For additional information, call the CAM_DIAG or CAM_DIAG_452 instruction.

The module detects the transition to an error-free state ("outgoing")

A diagnostic interrupt is only registered "outgoing" if all errors on the module are cleared.

Sequence:

1. The FM 452 detects that all errors have been eliminated and triggers a diagnostic interrupt. The error LED goes out. The diagnostic buffer remains unchanged.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
4. Evaluate the OB82_MDL_DEFECT bit.
   When this bit is "0", no errors are present on the module. You can close the evaluation session at this point.

Diagnostic interrupt control by CPU states

- Diagnostic interrupts are blocked by FM 452 when the CPU is in STOP mode.
- If not all of the queued errors are eliminated while the CPU is in STOP mode, the FM 452 will report the remaining errors as "incoming" events again at the next CPU transition to RUN.
- If all queued errors are eliminated when the CPU is in STOP mode, the error-free state of the FM 452 will **not** be reported by a diagnostic interrupt after the transition to RUN mode.

Error evaluation in the user program (S7-300, S7-400)

Fault reaction in the user program

The user program allows a targeted response to errors. The following options are available:

- Return values (RET_VAL) of the integrated instructions as a group display for errors that occurred during execution of the instructions.
- Error bits _ERR of the jobs as a group display for errors that occurred while a job was being processed.
- The DATA_ERR error bit as a group display for an error that the FM 452 has detected during a write job.
- The error identifier in JOB_ERR for the cause of an error in communication between instruction and FM 452.
- The CAM_DIAG or CAM_DIAG_452 instruction for reading the diagnostic buffer of the FM 452. This is where you find out the causes of errors in jobs and asynchronous events (operator errors, diagnostic errors).
- Diagnostic interrupts for fast reaction to events.

Diagnostics buffer of the module (S7-300, S7-400)

Diagnostic events

The module´s diagnostic buffer contains a maximum of 4 diagnostic events and is organized as a ring buffer.

A diagnostic event is written in the buffer if an "incoming" (error) message is detected. This can be an alarm, a synchronous error (data error) or even an asynchronous error (operational errors and diagnostics errors). From a cause of error several entries can occur as follow-up errors. Outgoing messages do not create any entries in the diagnostic buffer.

For each diagnostic event the following is given:

- Status (always incoming)
- Internal error
- External error
- Error class
- Error number
- Channel number
- Cam number (for cam data errors)

When a diagnostic event is written in the diagnostic buffer, the checkback signal DIAG = 1 is set.

With the CAM_DIAG or CAM_DIAG_452 instruction, the diagnostic buffer can be transferred as a whole into a data block (diagnostic DB) or displayed via the error evaluation dialog of the configuration software. If the diagnostic buffer is read, the module sets the checkback signal DIAG = 0.

Note

If the diagnostic buffer is read simultaneously from the CAM_DIAG or CAM_DIAG_452 instruction and the error evaluation dialog, it is possible that a new diagnostic event is not detected by the program.

Data and Structure of the Diagnostic DB (S7-300, S7-400)

Data and Structure of the Diagnostic DB

Note

Do not modify any data not listed in this table.

The following table shows the structure of the diagnostic DB:

Address	Name	Type	Initial value	Comment
0.0	MOD_ADDR (Enter!)	INT	0	Module address
256.0	JOB_ERR	INT	0	Communication errors
258.0	JOBBUSY	BOOL	FALSE	1 = job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = read diagnostic buffer unconditional
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostic data latest entry
272.0	DIAG[2]	STRUCT		Diagnostic data second entry
282.0	DIAG[3]	STRUCT		Diagnostic data third entry
292.0	DIAG[4]	STRUCT		Diagnostic data oldest entry

The diagram below shows the structure of a DIAG[n] diagnostic entry.

Address	Name	Type	Initial value	Comment
+0.0	STATE	BOOL	FALSE	0 = outgoing event 1 = incoming event
+0.1	INTF	BOOL	FALSE	1 = Internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 4: Data error 5: Machine data error 7: Cam data error 15: Messages 128: Diagnostic errors
+4.0	FNO	INT	0	Error number 0 to 255
+6.0	CH_NO	INT	0	Channel number (always 1)
+8.0	CAMNO	INT	0	Cam numbers 0 to 127 with error class = cam data error

List of JOB_ERR messages (S7-300, S7-400)

JOB_ERR alarms

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgment when reading from module. Module removed during read operation or module defective.
80A1	32929	-32607	Negative acknowledgment when writing to module. Module removed during write operation or module defective.
80A2 [1)]	32930	-32606	Protocol error on Layer 2 (data transfer interrupted within PROFINET / PROFIBUS DP, for example, due to wire break, missing connector, parameter assignment error, etc.)
80A3 [1)]	32931	-32605	Protocol error at user interface/user (data transfer interrupted within PROFINET / PROFIBUS DP, for example, due to wire break, missing connector, parameter assignment error, etc.)

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A4 [1]	32932	-32604	Communication problem on K bus
80B1	32945	-32591	Specified length wrong. FM_TYPE parameter in the channel DB is not correctly set for the module being used.
80B2	32946	-32590	The configured slot is empty.
80B3	32947	-32589	Actual module type does not match configured module type.
80C0 [1]	32960	-32576	The module has not yet prepared the data to be read.
80C1 [1]	32961	-32575	The data of a write job of the same type have not yet been processed on the module.
80C2 [1]	32962	-32574	The module is currently processing the maximum number of jobs.
80C3 [1]	32963	-32573	Required resources (memory etc.) currently in use.
80C4 [1]	32964	-32572	Communication errors
80C5 [1]	32965	-32571	Distributed I/Os not available.
80C6 [1]	32966	-32570	Priority class abort (restart or background)
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read from the DB. (write job)
8532	34098	-31438	DB number of the parameter DB too high (write job)
853A	34106	-31430	Parameter DB does not exist. (write job)
8544	34116	-31420	Error in nth (n > 1) read access to a DB after error occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written to the DB. (read job)
8730	34608	-30928	Parameter DB on the CPU write-protected. The data cannot be written to the DB (read job).
8732	34610	-30926	DB number of the parameter DB too high (read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error in nth (n > 1) write access to a DB after error occurred. (read job)

[1] Errors 80A2 to 80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time. Messages in the form 7xxx indicate temporary states in communication.

Error classes (S7-300, S7-400)

Operating error (S7-300, S7-400)

Error class 1: Operating error

Operating errors are detected asynchronous to operator input/commands.

The following table shows the possible operating errors and their respective causes:

No.	Meaning	Diagnostic interrupt
1	Software limit switch start overrun	Yes
2	Software limit switch end overrun	Yes
3	Traversing range start passed	Yes

No.	Meaning		Diagnostic interrupt
4	**Traversing range end passed**		Yes
13	**Set actual value on-the-fly cannot be executed**		Yes
	Cause	The software limit switches are outside the traversing range after the actual value on the fly is set (-100 m to +100 m or -1000 m to +1000 m). The shift resulting from set actual value / set actual value on-the-fly is more than ±100 m or ± 1000 m.	
	Effect	Axis not synchronized.	

Data error (S7-300, S7-400)

Error class 4: Data error

Data errors are detected synchronous to operator input/commands.

The following table shows the possible data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
10	**Zero offset error**		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. The software limit switches are outside the traversing range after zero offset (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	**Faulty actual value setting**		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
12	**Reference point error**		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
20	**Enable machine data not permitted**		No
	Cause	There are no new (error-free) machine data on the module.	
21	**Set actual value on-the-fly not permitted**		No
	Cause	An attempt was made to execute "set actual value on-the-fly" while "retrigger reference point" was active.	
27	**Illegal bit-coded setting**		No
	Cause	Unused and, in this case, unwritten bits are not 0. An attempt was made to select "length measurement" and "edge detection" at the same time.	

No.	Meaning		Diagnostic interrupt
28	**Retrigger reference point is not permitted**		No
	Cause	An attempt was made to execute "retrigger reference point" while "set actual value on-the-fly" was active. An attempt was made to execute "retrigger reference point" with an SSI encoder.	
29	**Illegal bit-coded command**		No
	Cause	Unused and, in this case, unwritten bits are not 0.	
30	**Incorrect lead time**		No
31	**Incorrect cam number**		No
	Cause	Cam is invalid. The cam number is not in the range 0 to 127.	
32	**Incorrect cam start**		No
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam start is < 0 or greater than the end of the rotary axis.	
33	**Incorrect cam end / incorrect activation time**		No
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. The cam is not activated at least for the duration of one pulse. With an inverse cam, there are not at least 4 pulses between the cam start and cam end.	
34	**Cancel set actual value not possible**		No
	Cause	The actual position value would be outside the working range with an SSI encoder and linear axis after making the setting.	
35	**Incorrect actual value specified by "set actual value" / "set actual value on-the-fly"**		No
	Cause	The specified actual value is outside the permitted numeric range of ±100 m or ±1000 m. The software limit switches would be outside the traversing range (-100m to +100 m or -1000 m to +1000 m) were this setting to be applied. The offset derived from set actual value / set actual value on-the-fly would be more than ±100 m or ±1000 m.	
107	**Parameters not set for the axis**		No
	Cause	There are no machine data on the axis. There are no machine data activated on the axis.	
108	**Axis not synchronized**		No
	Cause	One of the settings "set actual value" or "set actual value on-the-fly" was initiated although the axis is not synchronized.	
109	**Cam processing running.**		No
110	**Incorrect number of cams to be modified**		No

Machine data error (S7-300, S7-400)

Error class 5: Machine data error

The diagnostic interrupt is only triggered if an error is detected in the system data block (SDB).

The following table shows the possible machine data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
5	**Error in hardware interrupt setting**		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	**Incorrect minimum edge distance**		Yes
	Cause	You have entered a value < 0 or > 10^9 µm as the minimum edge-to-edge distance.	
8	**Incorrect axis type**		Yes
	Cause	You have specified neither 0 nor 1 as the axis type	
9	**Incorrect end of rotary axis**		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or 1 to 10^8 µm (depending on the resolution).	
10	**Incorrect encoder type**		Yes
	Cause	The value for the encoder type is outside the valid range of 1 to 10	
11	**Incorrect distance per encoder revolution**		Yes
	Cause	The value for distance/encoder revolution is outside the permitted range of 1 to 10^9 µm (regardless of the resolution).	
13	**Incorrect number of increments per encoder revolution** (see "Increments per encoder revolution (Page 572)")		Yes
14	**Incorrect number of revolutions** (see "Number of encoder revolutions (Page 577)")		Yes
15	**Incorrect transmission rate**		Yes
	Cause	You have specified a transmission rate outside the permitted range of 0 to 3.	
16	**Incorrect reference point coordinate**		Yes
	Cause	The coordinate is outside the range of -100 m to +100 m or -1000 m to +1000 m, depending on the resolution. Linear axis: The coordinate is outside the working range. Rotary axis: The coordinate is greater than the end of the rotary axis or < 0.	
17	**Incorrect absolute value adjustment**		Yes
	Cause	SSI encoder: The value of the absolute encoder adjustment is not in the encoder range (increments per encoder revolution x number of revolutions - 1).	
18	**Incorrect "Retrigger reference point" type**		Yes
	Cause	You have specified a value other than 0, 1, 6 and 7.	
19	**Incorrect direction adaptation**		Yes
	Cause	You have specified a value other than 0 and 1.	
20	**Hardware monitoring not possible**		Yes
	Cause	You have set frame error monitoring to "FALSE" in the parameter DB.	

No.	Meaning		Diagnostic interrupt
21	**Incorrect software limit switch start**		Yes
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any existing zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	**Incorrect software limit switch end**		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution) or is less than the software limit switch start. The software limit switch end (including any existing zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
144	**Incorrect number of cams**		Yes
	Cause	You have specified a value other than 0 to 3 for the number of cams.	
145	**Incorrect hysteresis**		Yes
	Cause	The hysteresis is outside the range 0 to 65535 × resolution. Hysteresis is greater than ¼ × operating range or ¼ × rotary axis range.	
146	**Incorrect simulation velocity**		Yes
	Cause	The simulation speed is outside the range 1000 × RES to 3 × 10^7 × RES, or is greater than 5 × 10^8 µm/min. The simulation velocity cannot be set internally.	
147	**Incorrect track**		Yes
	Cause	Control of a track outside 0 to 15 (bits 0 to 15) was selected.	
148	**Incorrect selection of enable inputs**		Yes
	Cause	You wanted to activate a track outside 3 to 10 (bits 0 to 7) using an external signal.	
149	**Incorrect selection of special tracks**		Yes
	Cause	You wanted to define a track outside 0, 1 and 2 (bits 0, 1 and 2) as a special track.	
150	**Incorrect high count value track 0**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
151	**Incorrect high count value track 1**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
200	**Incorrect resolution**		Yes
	Cause	You have specified a resolution < 0.1 µm/pulse or >1000 µm/pulse. You have specified a distance per encoder revolution and a number of pulses per encoder revolution that results in a resolution of < 0.1 or > 1000.	
201	**Position encoder does not match the working range/rotary axis range**		Yes
	Cause	SSI position encoder and rotary axis: The encoder does not exactly cover the rotary axis range. Linear axis: The position encoder does not cover at least the operating range (including software limit switches).	

Cam data error (S7-300, S7-400)

Error class 7: Cam data error

The diagnostic interrupt is only triggered if an error is detected in the system data block (SDB).

The following table shows the possible cam data errors and their respective causes.

No.	Meaning		Diagnostic interrupt
1	**Illegal hardware interrupt**		Yes
	Cause	You attempted to define a process interrupt for a cam with a cam number > 7.	
2	**Incorrect track number**		Yes
	Cause	The track number is outside the range from 0 to 31.	
3	**Incorrect cam start**		Yes
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam start is < 0 or greater than the end of the rotary axis.	
4	**Incorrect cam end**		Yes
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). The cam is at least one pulse long. Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. There are not at least 4 pulses between the cam start and cam end for an inverse cam.	
5	**Incorrect activation time**		Yes
	Cause	The activation time is < 0 µs. The maximum value (Page 584) depends on the number of cams.	
6	**Incorrect lead time**[1]		Yes
	Cause	The lead time is < 0 µs. The maximum value (Page 585) depends on the number of cams.	
50	**Too many cam sets**		Yes
	Cause	You wanted to enter more cam records than is possible with this number of cams.	
51	**Axis in operation**		Yes
	Cause	You wanted to enter cam records while the cam controller is active.	
52	**Parameters not set for the axis**		Yes
	Cause	You want to enter cam data although not machine data are active yet.	

[1] The error message can also be issued if you have assigned the parameter "inverted" as the count direction with an absolute encoder (SSI).

Alarms (S7-300, S7-400)

Error class 15: Alarms

The following table shows the possible alarms and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Start of parameter assignment**		No
	Cause	The module has detected parameter assignment via a system data block.	
2	**End of parameter assignment**		No
	Cause	The module has processed the parameter assignment via a system data block error-free.	

Diagnostic errors (S7-300, S7-400)

Error class 128: Diagnostic errors

The following table shows the possible diagnostic errors and their respective causes, effects and remedies.

No.	Meaning		Diagnostic interrupt
4	**External auxiliary voltage missing**		Yes
	Cause	External auxiliary 24 V voltage is not connected or has failed, front connector missing	
	Effect	See "Triggering diagnostic interrupts (Page 596)": • Cam processing will be disabled • Track outputs are deactivated • With incremental encoders, synchronization is canceled • FM 452 has not been configured (checkback signal PARA = 0).	
	Correction	Make sure that the 24 V connection is correct (if the 24 V connection is correct, then the module is defective)	
5	**No front connector**		Yes
	Cause	Front connector not inserted	
	Effect	• No external 24 V auxiliary voltage • Module not ready for operation	
	Correction	Insert the front connector	

No.	Meaning		Diagnostic interrupt
51	**Watchdog expired**		Yes
	Cause	• Strong interference on FM 452 • Error in the FM 452	
	Effect	• Module is reset • If, after resetting the module, no module fault is detected, the module is ready for operation again • The module reports the expired watchdog with "incoming" and "outgoing"	
	Correction	• Eliminate the interference • Contact the relevant sales department which will require details of the exact circumstances leading to the error • Replace the FM 452	
52	**Internal module power supply failed**		Yes
	Cause	Error in the FM 452	
	Effect	• Module is reset • If, after resetting the module, no module fault is detected, the module is ready for operation again	
	Correction	Replace the FM 452	
70	**Hardware interrupt lost**		Yes
	Cause	A hardware interrupt event detected by FM 452 cannot be reported because an identical event has not yet been processed by the user program/ CPU.	
	Effect	• Cam processing will be disabled • Track outputs are deactivated • With incremental encoders, synchronization is canceled	
	Correction	• Integrate OB 40 into the user program • Check bus connection of the module • Deactivate hardware interrupt • Adapt your hardware and software to suit your process requirements (for example, faster CPU, optimize user program)	

No.	Meaning		Diagnostic interrupt
144	**Encoder wire break**		Yes
	Cause	• Encoder cable cut or not plugged in • Encoder has no transverse signals • Incorrect pin assignment • Cable length too long • Encoder signals short-circuited	
	Effect	• Cam processing will be disabled • Track outputs are deactivated • With incremental encoders, synchronization is canceled	
	Correction	• Check the encoder cable • Keep within encoder specifications • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog • Keep to the module's technical data	
145	**Absolute encoder frame error**		Yes
	Cause	The frame traffic between FM 452 and the absolute encoder (SSI) is incorrect or has been interrupted: • Encoder cable cut or not plugged in • Incorrect encoder type • Encoder incorrectly set (programmable encoders) • Frame length incorrectly specified • Encoder returns incorrect values (encoder defective) • Interference on measuring system cable • Transmission rate selected too high	
	Effect	• Cam processing will be disabled • Track outputs are deactivated • The last correct actual value remains unchanged till the end of the next correct SSI transfer	
	Correction	• Check the encoder cable • Check the encoder • Check the frame traffic between the encoder and FM 452	

No.	Meaning		Diagnostic interrupt
146	**Missing pulses, incremental encoder**		Yes
	Cause	• Encoder monitoring has detected missing pulses • Number of increments per encoder revolution is incorrectly entered • Encoder defective: does not return the defined number of pulses • Bad or missing zero marker • Interference on the encoder cable	
	Effect	• Cam processing will be disabled • Track outputs are deactivated • Synchronization is canceled	
	Correction	• Enter the correct number of increments/encoder revolution • Check the encoder and its cable • Comply with shielding and grounding regulations • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog	

PID control

3

3.1 Principles for control

3.1.1 Controlled system and actuators

Controlled system

Room temperature control by means of a heating system is a simple example of a controlled system. A sensor measures the room temperature and transfers the value to a controller. The controller compares the current room temperature with a setpoint and calculates an output value (manipulated variable) for heating control.

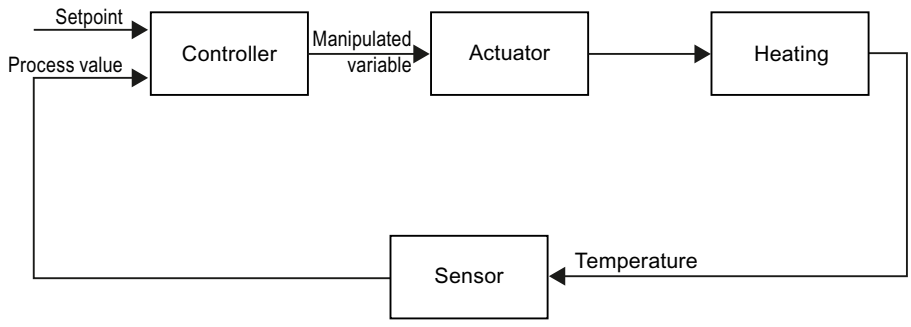

A properly set PID controller reaches this setpoint as quickly as possible and then holds it a constant value. After a change in the output value, the process value often changes only with a time delay. The controller has to compensate for this response.

Actuators

The actuator is an element of the controlled system and is influenced by the controller. Its function modifies mass and energy flows.

The table below provides an overview of actuator applications.

Application	Actuator
Liquid and gaseous mass flow	Valve, shutter, gate valve
Solid mass flow, e.g., bulk material	Articulated baffle, conveyor, vibrator channel
Flow of electrical power	Switching contact, contactor, relay, thyristor
	Variable resistor, variable transformer, transistor

Actuators are distinguished as follows:

- Proportional actuators with constant actuating signal
  These elements set degrees of opening, angular positions or positions in proportion to the output value. The output value has an analog effect on the process within the control range. Actuators in this group include spring-loaded pneumatic drives, as well as motorized drives with position feedback for which a position control system is formed.
  An continuous controller, such as PID_Compact, generates the output value.
- Proportional actuators with pulse-width modulated signal
  These actuators are used to generate the output of pulses with a length proportional to the output value within the sampling time intervals. The actuator - e.g. a heating resistor or cooling apparatus - is switched on in isochronous mode for durations that differ depending on the output value.
  The actuating signal can assume unipolar "On" or "Off" states, or represent bipolar states such as "open/close", "forward/backward", "accelerate/brake".
  The output value is generated by a two-step controller such as PID_Compact with pulse-width modulation.
- Actuators with integral action and three-step actuating signal
  Actuators are frequently operated by motors with an on period that is proportional to the actuator travel of the choke element. This includes elements such as valves, shutters, and gate valves. In spite of their different design, all of these actuators follow the effect of an integral action at the input of the controlled system.
  A step controller, such as PID_3Step. generates the output value.

3.1.2 Controlled systems

The properties of a controlled system can hardly be influenced as these are determined by the technical requirements of the process and machinery. Acceptable control results can only be achieved by selecting a suitable controller type for the specific controlled system and adapting the controller to the time response of the controlled system. Therefore, it is is indispensable for the configuration of the proportional, integral and derivative actions of the controller to have precise knowledge of the type and parameters of the controlled system.

Controlled system types

Controlled systems are classified based on their time response to step changes of the output value.

We distinguish between the following controlled systems:

- Self-regulating controlled systems
  - Proportional-action controlled systems
  - PT1 controlled systems
  - PT2 controlled systems
- Non-self-regulating controlled systems
- Controlled systems with and without dead time

Self-regulating controlled systems

Proportional-action controlled systems

In proportional-action controlled systems, the process value follows the output value almost immediately. The ratio between the process value and output value is defined by the proportional Gain of the controlled system.

Examples:

- Gate valve in a piping system
- Voltage dividers
- Step-down function in hydraulic systems

PT1 controlled systems

In a PT1 controlled system, the process value initially changes in proportion to the change of the output value. The rate of change of the process value is reduced as a function of the time until the end value is reached, i.e., it is delayed.

Examples:

- Spring damping system
- Charge of RC elements
- Water container that is heated with steam.

The time constants are often identical for heating and cooling processes, or for charging and discharge characteristics. With different time constants, controlling is clearly more complex.

PT2 controlled systems

In a PT2 controlled system, the process value does not immediately follow a step change of the output value, i.e., it increases in proportion to the positive rate of rise and then approaches the setpoint at a decreasing rate of rise. The controlled system shows a proportional response characteristic with second order delay element.

Examples:

- Pressure control
- Flow rate control
- Temperature control

Non-self-regulating controlled systems

Non-self-regulating controlled systems have an integral response. The process value approaches an infinite maximum value.

Example:

- Liquid flow into a container

Controlled systems with dead time

A dead time always represents the runtime or transport time that has to expire before a change to the system input can be measured at the system output.

In controlled systems with dead time, the process value change is delayed by the amount of the dead time.

Example:

Conveyor

3.1.3 Characteristic values of the control section

Determining the time response from the step response

Time response of the controlled system can be determined based on the time characteristic of process value x following a step change of output value y. Most controlled systems are self-regulating controlled systems.

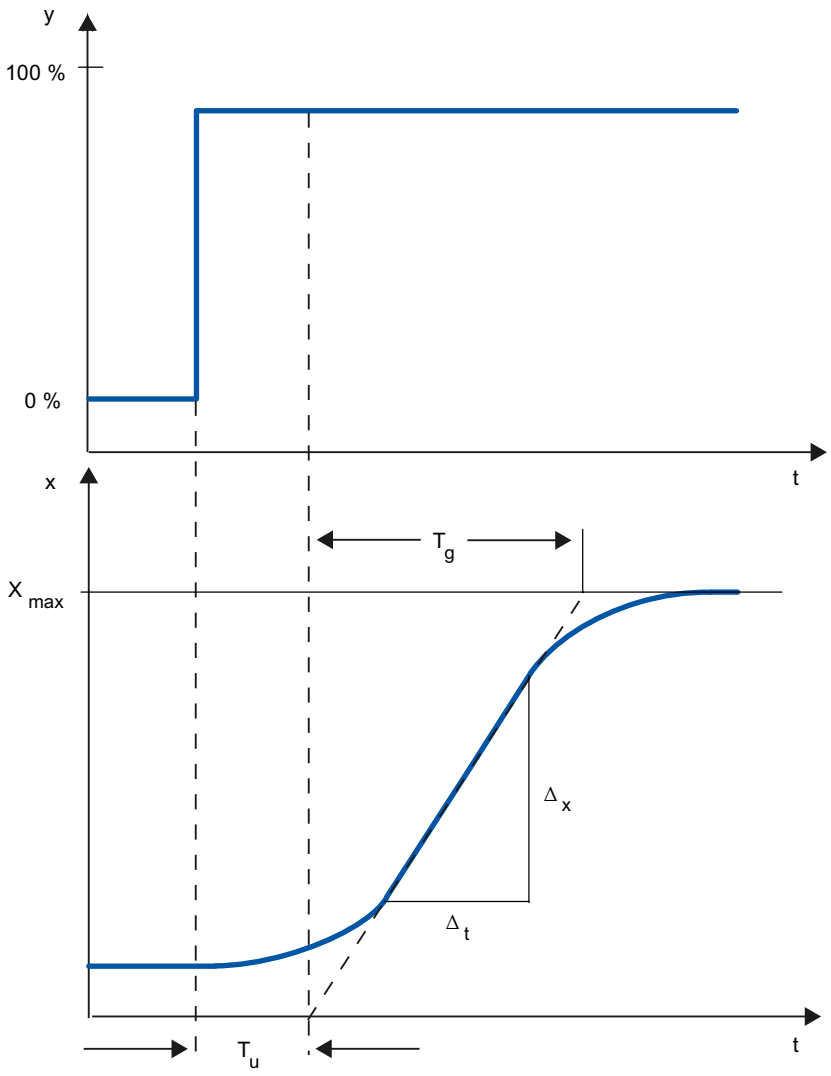

The time response can be determined by approximation using the variables Delay time T_u, Recovery time T_g and Maximum value X_{max}. The variables are determined by applying tangents to the maximum value and the inflection point of the step response. In many situations, it is not possible to record the response characteristic up to the maximum value because the process value cannot exceed specific values. In this case, the rate of rise v_{max} is used to identify the controlled system ($v_{max} = \Delta_x/\Delta_t$).

The controllability of the controlled system can be estimated based on the ratio T_u/T_g, or $T_u \times v_{max}/X_{max}$. Rule:

Process type	T_u / T_g	Suitability of the controlled system for controlling
I	< 0.1	can be controlled well
II	0.1 to 0.3	can still be controlled
III	> 0.3	difficult to control

Influence of the dead time on the controllability of a controlled system

A controlled system with dead time and recovery reacts as follows to a jump of the output value.

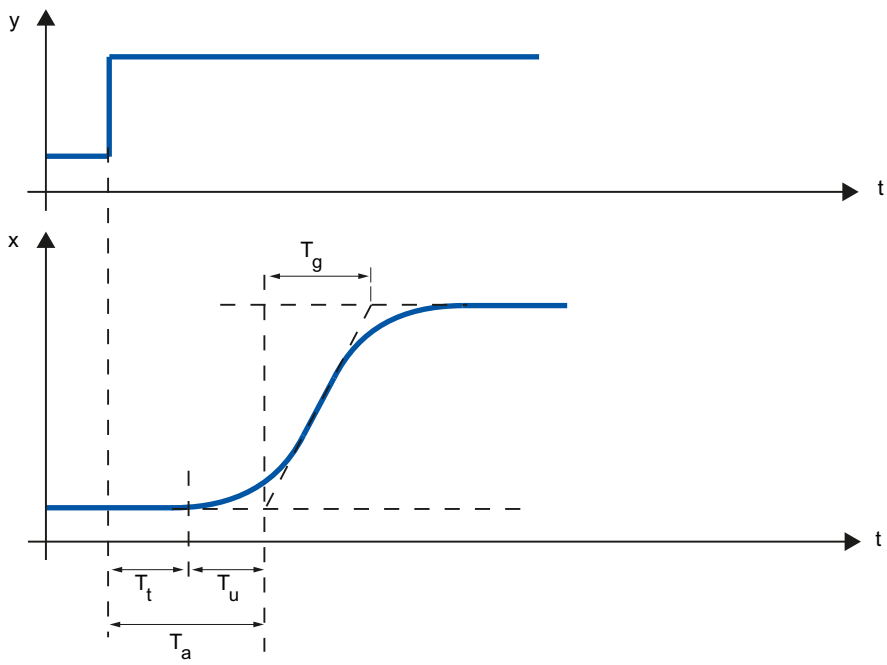

T_t	Dead time
T_u	Delay time
T_g	Recovery time
y	Output value
x	Process value

The controllability of a self-regulating controlled system with dead time is determined by the ratio of T_t to T_g. T_t must be small compared to T_g. Rule:

$T_t/T_g \leq 1$

Response rate of controlled systems

Controlled systems can be judged on the basis of the following values:

T_u < 0.5 min, T_g < 5 min = fast controlled system

T_u > 0.5 min, T_g > 5 min = slow controlled system

Parameters of certain controlled systems

Physical quantity	Controlled system	Delay time T_u	Recovery time T_g	Rate of rise v_{max}
Temperature	Small electrically heated furnace	0.5 to 1 min	5 to 15 min	Up to 60 K/min.
	Large electrically heated annealing furnace	1 to 5 min	10 to 20 min	Up to 20 K/min.
	Large gas-heated annealing furnace	0.2 to 5 min	3 to 60 min	1 to 30 K/min
	Distillation tower	1 to 7 min	40 to 60 min	0.1 to 0.5° C/s
	Autoclaves (2.5 m^3)	0.5 to 0.7 min	10 to 20 min	Not specified
	High-pressure autoclaves	12 to 15 min	200 to 300 min	Not specified
	Steam superheater	30 s to 2.5 min	1 to 4 min	2°C/s
	Injection molding machines	0.5 to 3 min	3 to 30 min	5 to 20 K/min
	Extruders	1 to 6 min	5 to 60 min	
	Packaging machines	0.5 to 4 min	3 to 40 min	2 to 35 K/min
	Room heating	1 to 5 min	10 to 60 min	1° C/min
Flow rate	Pipeline with gas	0 to 5 s	0.2 to 10 s	Not relevant
	Pipeline with liquid	None	None	
Pressure	Gas pipeline	None	0.1 s	Not relevant
	Drum boiler with gas or oil firing	None	150 s	Not relevant
	Drum boiler with impact grinding mills	1 to 2 min	2 to 5 min	Not relevant
Vessel level	Drum boiler	0.6 to 1 min	Not specified	0.1 to 0.3 cm/s
Speed	Small electric drive	None	0.2 to 10 s	Not relevant
	Large electric drive	None	5 to 40 s	Not relevant
	Steam turbine	None	Not specified	50 min^{-1}
Voltage	Small generators	None	1 to 5 s	Not relevant
	Large generators	None	5 to 10 s	Not relevant

3.1.4 Pulse controller

Two-step controllers without feedback

Two-step controllers have the state "ON" and "OFF" as the switching function. This corresponds to 100% or 0% output. This behavior generates a sustained oscillation of process value x around setpoint w.

The amplitude and duration of the oscillation increase in proportion to the ratio between the delay time T_u and recovery time T_g of the controlled system. These controllers are used mainly

for simple temperature control systems (such as electrically directly heated furnaces) or as limit-value signaling units.

The following diagram shows the characteristic of a two-step controller

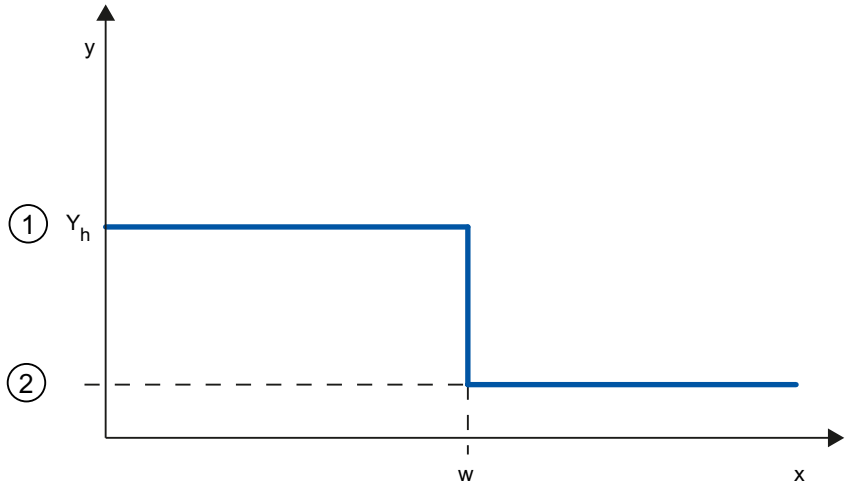

①	ON
②	OFF
Y_h	Control range
w	Setpoint

The following diagram shows the control function of a two-step controller

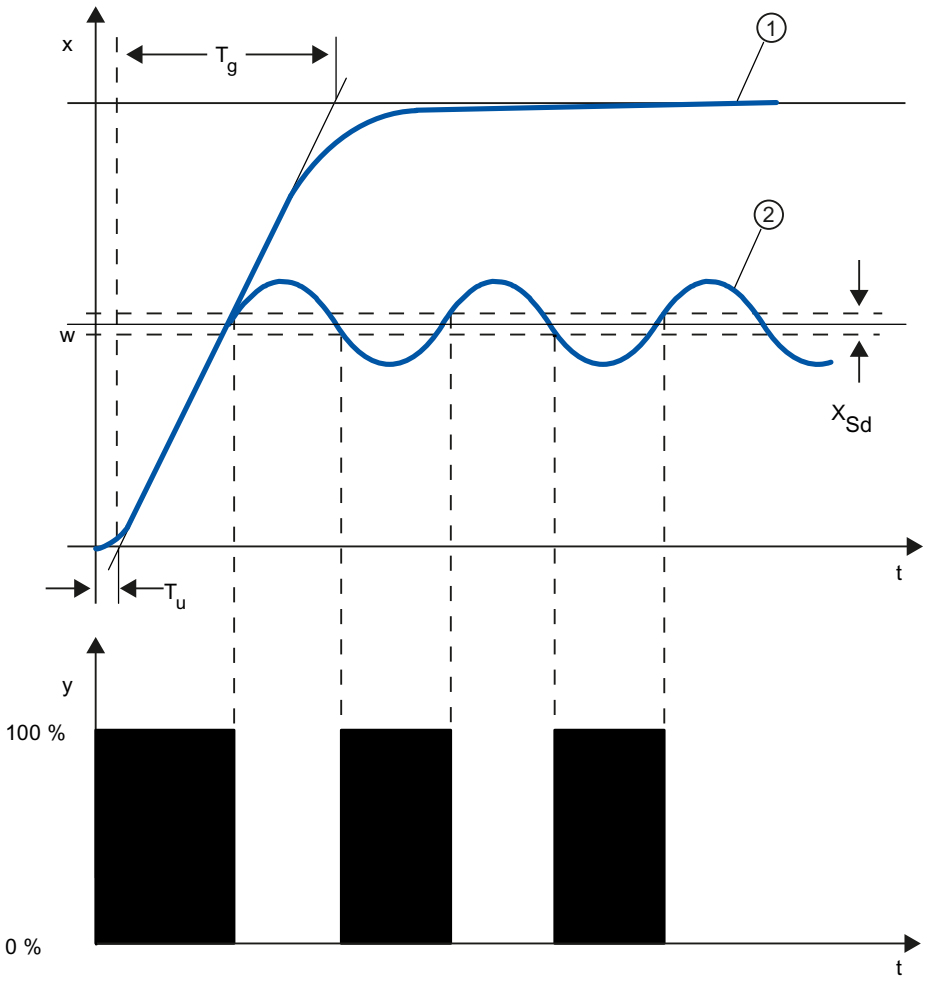

①	Response characteristic without controller
②	Response characteristic with two-step controller
T_u	Delay time
T_g	Recovery time
X_{Sd}	Switching difference

Two-step controllers with feedback

The behavior of two-step controllers in the case of controlled systems with larger delay times, such as furnaces where the functional space is separated from the heating, can be improved by the use of electronic feedback.

The feedback is used to increase the switching frequency of the controller, which reduces the amplitude of the process value. In addition, the control-action results can be improved substantially in dynamic operation. The limit for the switching frequency is set by the output level. It should not exceed 1 to 5 switches per minute at mechanical actuators, such as relays and contactors. In the case of voltage and current outputs with downstream thyristor or Triac controllers high switching frequencies can be selected that exceed the limit frequency of the controlled system by far.

Since the switching pulses can no longer be determined at the output of the controlled system, results comparable with those of continuous controllers are obtained.

The output value is generated by pulse-width modulation of the output value of a continuous controller.

Two-step controllers with feedback are used for temperature control in furnaces, at processing machines in the plastics, textile, paper, rubber and foodstuff industries as well as for heating and cooling devices.

Three-step controllers

Three-step controllers are used for heating / cooling. These controllers have two switching points as their output. The control-action results are optimized through electronic feedback structures. Fields of applications for such controllers are heating, low-temperature, climatic chambers and tool heating units for plastic-processing machines.

The following diagram shows the characteristic of a three-step controller

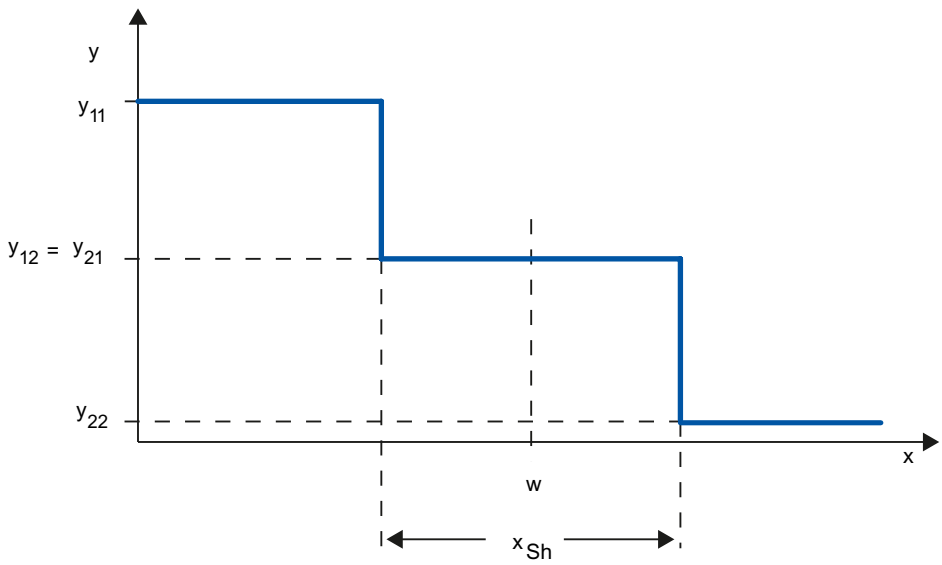

y	Output value, e.g. y11 = 100% heating y12 = 0% heating y21 = 0% cooling y22 = 100% cooling
x	Physical quantity of the process value, e.g., temperature in° C
w	Setpoint
x_{Sh}	Distance between Switching Point 1 and Switching Point 2

3.1.5 Response to setpoint changes and disturbances

Response to setpoint changes

The process value should follow a setpoint change as quickly as possible. The response to setpoint changes is improved by minimizing fluctuation of the process value and the time required to reach the new setpoint.

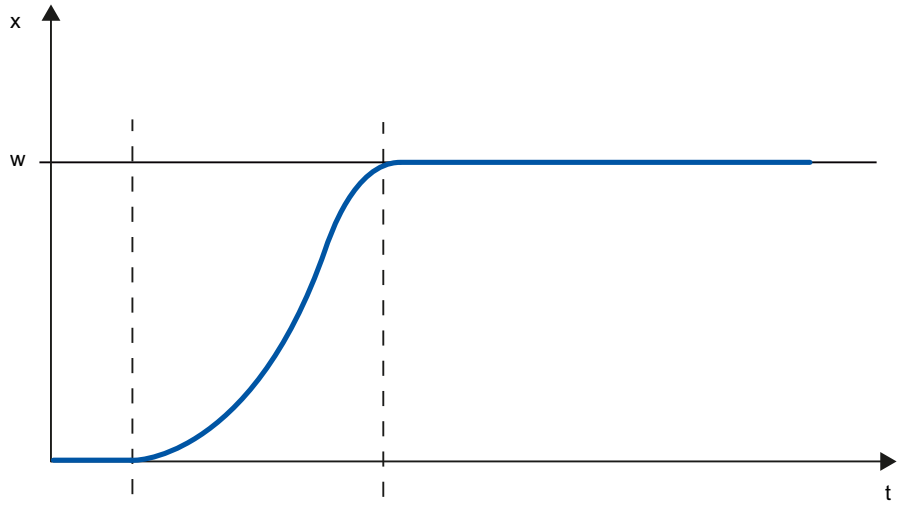

x Process value

w Setpoint

Response to disturbances

The setpoint is influenced by disturbance variables. The controller has to eliminate the resulting control deviations in the shortest time possible. The response to disturbances is improved by minimizing fluctuation of the process value and the time required to reach the new setpoint.

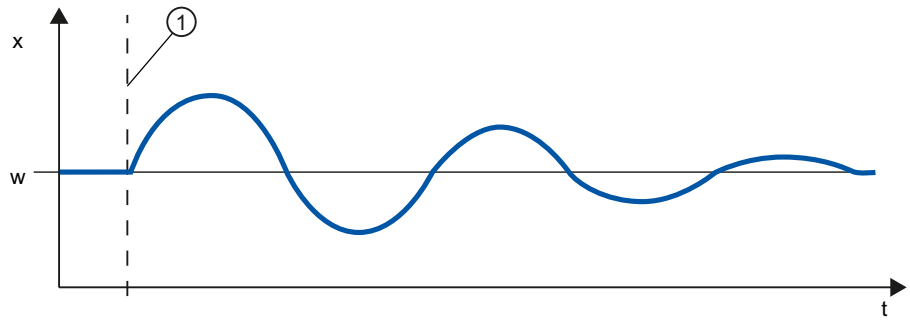

x Process value

w Setpoint

① Influencing a disturbance variable

Disturbance variables are corrected by a controller with integral action. A persistent disturbance variable does not reduce control quality because the control deviation is relatively constant. Dynamic disturbance variables have a more significant impact on control quality

because of control deviation fluctuation. The control deviation is eliminated again only by means of the slow acting integral action.

A measurable disturbance variable can be included in the controlled system. This inclusion would significantly accelerated the response of the controller.

3.1.6 Control Response at Different Feedback Structures

Control behavior of controllers

A precise adaptation of the controller to the time response of the controlled system is decisive for the controller's precise settling to the setpoint and optimum response to disturbance variables.

The feedback circuit can have a proportional action (P), proportional-derivative action (PD), proportional-integral action (PI), or proportional-integral-derivative action (PID).

If step functions are to be triggered by control deviations, the step responses of the controllers differ depending on their type.

Step response of a proportional action controller

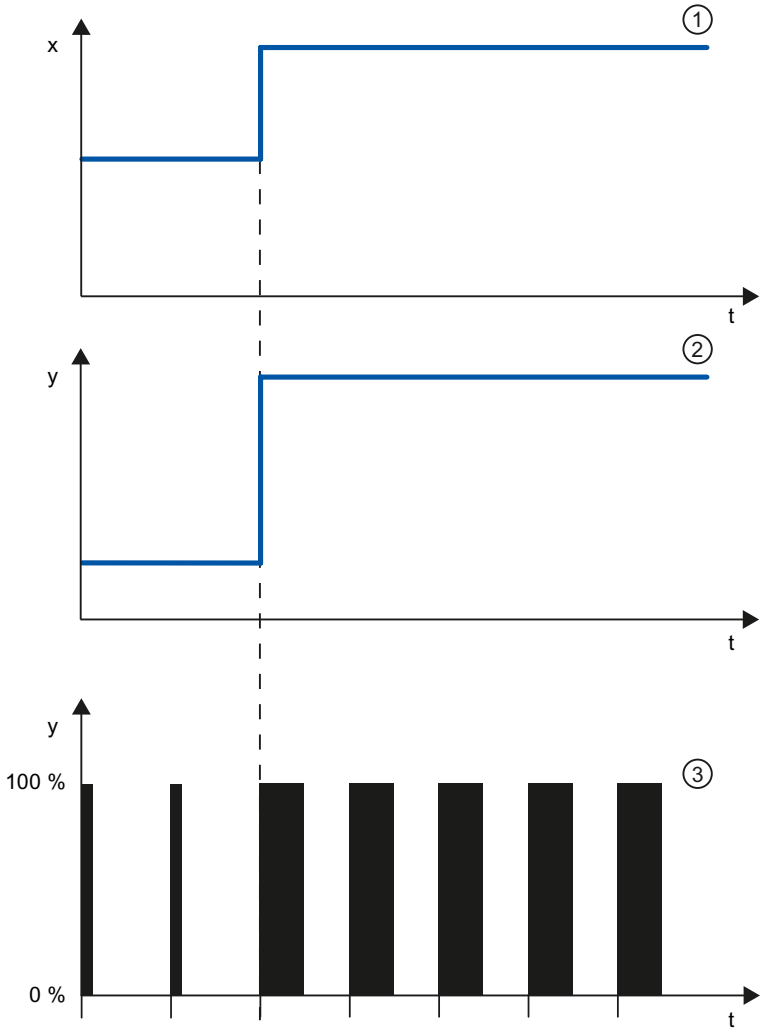

① Control deviation
② Output value of a continuous controller
③ Output value of a pulse controller

Equation for proportional action controller

Output value and control deviation are directly proportional, meaning:

Output value = proportional gain × control deviation

y = GAIN × x

Step response of a PD-action controller

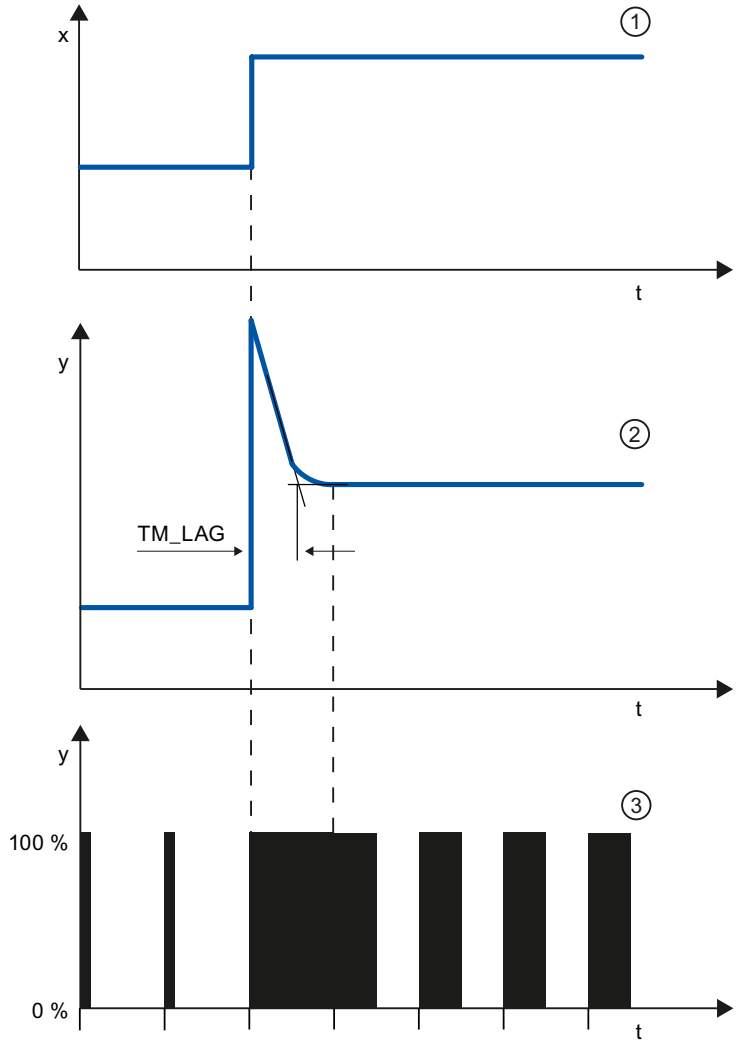

①	Control deviation
②	Output value of a continuous controller
③	Output value of a pulse controller
TM_LAG	Delay of the Derivative action

Equation for PD-action controller

The following applies for the step response of the PD-action controller in the time range:

$$y = GAIN \cdot X_W \cdot \left(1 + \frac{TD}{TM_LAG} \cdot e^{\frac{-t}{TM_LAG}} \right)$$

t = time interval since the step of the control deviation

The derivative action generates a output value as a function of the rate of change of the process value. A derivative action by itself is not suitable for controlling because the output value only

follows a step of the process value. As long as the process value remains constant, the output value will no longer change.

The response to disturbances of the derivative action is improved in combination with a proportional action. Disturbances are not corrected completely. The good dynamic response is advantageous. A well attenuated, non-oscillating response is achieved during approach and setpoint change.

A controller with derivative action is not appropriate if a controlled system has pulsing measured quantities, for example, in the case of pressure or flow control systems.

Step response of a PI-action controller

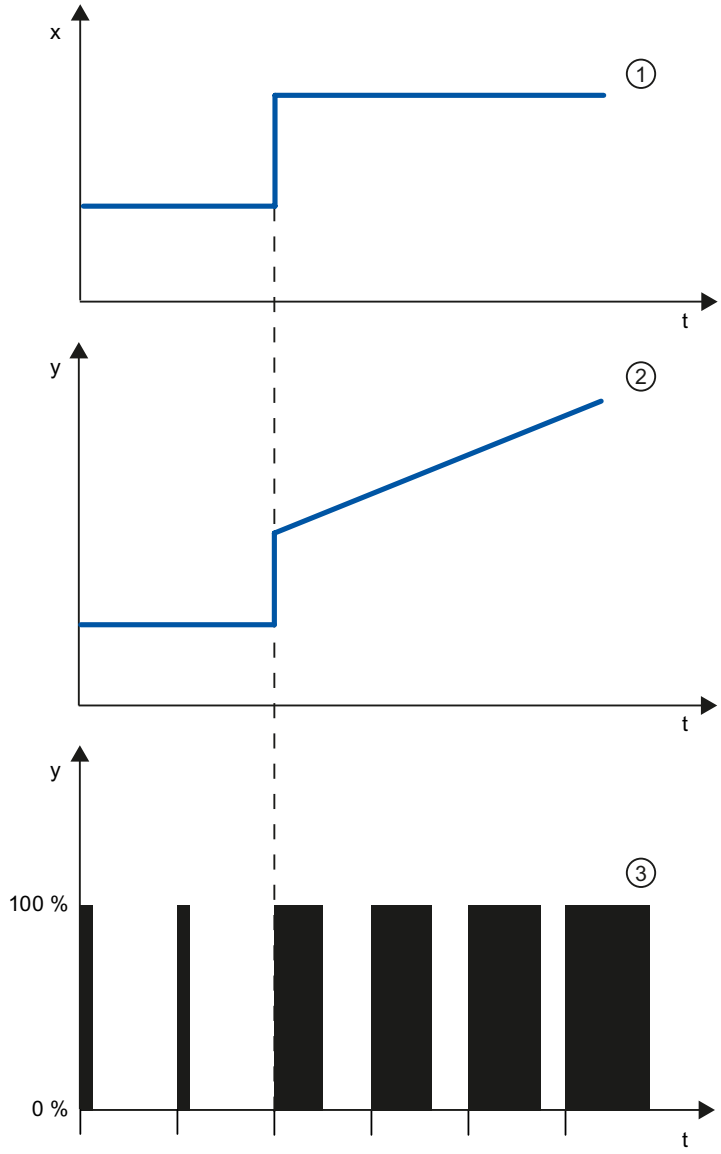

① Control deviation
② Output value of a continuous controller
③ Output value of a pulse controller

An integral action in the controller adds the control deviation as a function of the time. This means that the controller corrects the system until the control deviation is eliminated. A sustained control deviation is generated at controllers with proportional action only. This effect can be eliminated by means of an integral action in the controller.

In practical experience, a combination of the proportional, integral and derivative actions is ideal, depending on the requirements placed on the control response. The time response of the individual components can be described by the controller parameters proportional gain GAIN, integral action time TI (integral action), and derivative action time TD (derivative action).

Equation for PI-action controller

The following applies for the step response of the PI-action controller in the time range:

$$y = GAIN \cdot X_W \cdot \left(1 + \frac{1}{TI \cdot t}\right)$$

t = time interval since the step of the control deviation

Step response of a PID controller

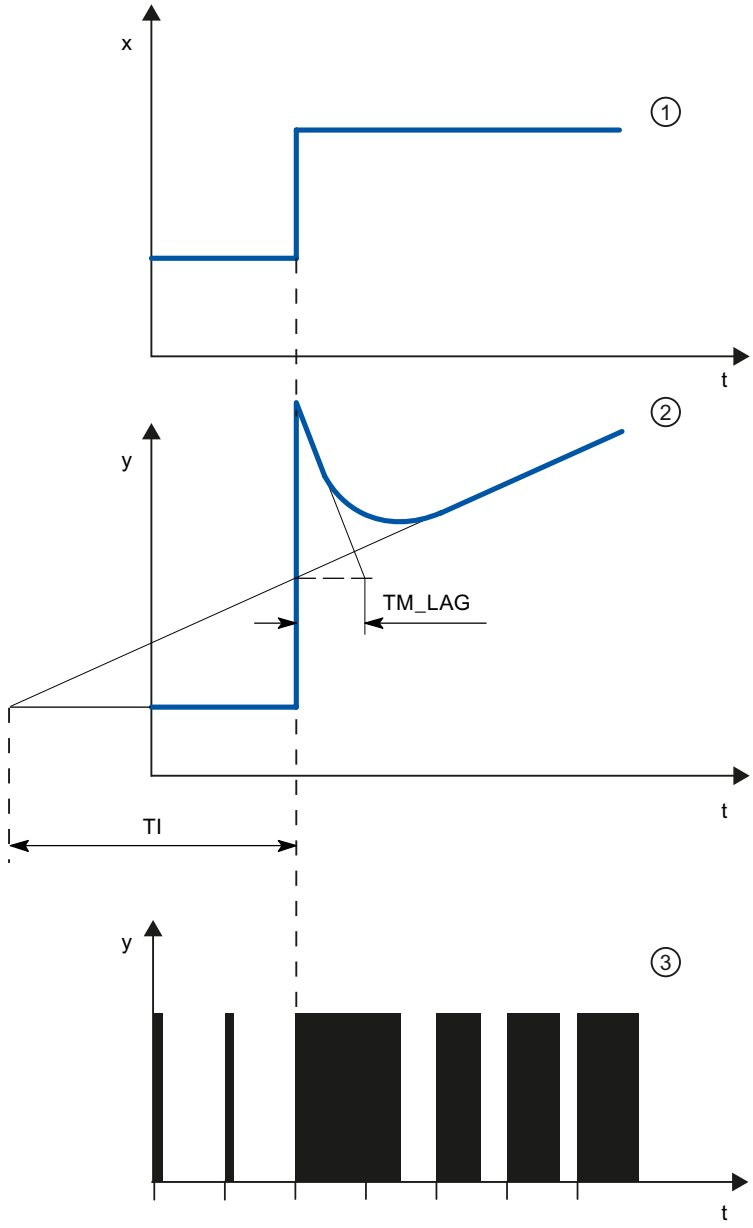

①	Control deviation
②	Output value of a continuous controller
③	Output value of a pulse controller
TM_LAG	Delay of the Derivative action
T_i	Integral action time

Equation for PID controller

The following applies for the step response of the PID controller in the time range:

$$y = GAIN \cdot X_w \cdot \left(1 + \frac{1}{TI \cdot t} + \frac{TD}{TM_LAG} \cdot e^{\frac{-t}{TM_LAG}} \right)$$

t = time interval since the step of the control deviation

Response of a controlled system with different controller structures

Most of the controller systems occurring in process engineering can be controlled by means of a controller with PI-action response. In the case of slow controlled system with a large dead time, for example temperature control systems, the control result can be improved by means of a controller with PID action.

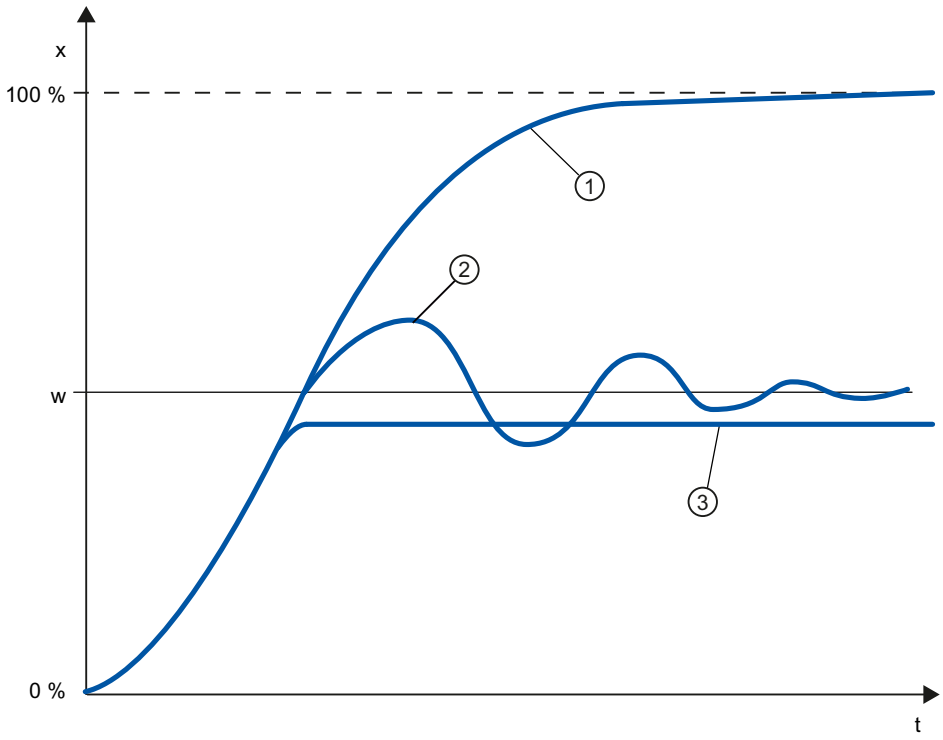

①	No controller
②	PID controller
③	PD-action controller
w	Setpoint
x	Process value

Controllers with PI and PID action have the advantage that the process value does not have any deviation from the setpoint value after settling. The process value oscillates over the setpoint during approach.

3.1.7 Selection of the controller structure for specified controlled systems

Selection of the Suitable Controller Structures

To achieve optimum control results, select a controller structure that is suitable for the controlled system and that you can adapt to the controlled system within specific limits.

The table below provides an overview of suitable combinations of a controller structure and controlled system.

Controlled system		Controller structure			
		P	PD	PI	PID
	With dead time only	Unsuitable	Unsuitable	Suitable	Unsuitable
	PT1 with dead time	Unsuitable	Unsuitable	Well suited	Well suited
	PT2 with dead time	Unsuitable	Suited conditionally	Well suited	Well suited
	Higher order	Unsuitable	Unsuitable	Suited conditionally	Well suited
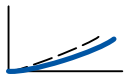	Not self-regulating	Well suited	Well suited	Well suited	Well suited

The table below provides an overview of suitable combinations of a controller structure and physical quantity.

Physical quantity	Controller structure			
	P	PD	PI	PID
	Sustained control deviation		No sustained control deviation	
Temperature	For low performance requirements and proportional action controlled systems with $T_u/Tg < 0,1$	Well suited	The most suitable controller structures for high performance requirements (except for specially adapted special controllers)	
Pressure	Suitable, if the delay time is inconsiderable	Unsuitable	The most suitable controller structures for high performance requirements (except for specially adapted special controllers)	
Flow rate	Unsuitable, because required GAIN range is usually too large	Unsuitable	Suitable, but integral action controller alone often better	Hardly required

3.1.8 PID parameter settings

Rule of Thumb for the Parameter Setting

Controller structure	Setting
P	GAIN ≈ v_{max} × T_u [° C]
PI	GAIN ≈ 1.2 × v_{max} × T_u [° C] TI ≈ 4 × T_u [min]
PD	GAIN ≈ 0.83 × v_{max} × T_u [° C] TD ≈ 0.25 × v_{max} × T_u [min] TM_LAG ≈ 0.5 × TD[min]
PID	GAIN ≈ 0.83 × v_{max} × T_u [° C] TI ≈ 2 × T_u [min] TD ≈ 0.4 × T_u [min] TM_LAG ≈ 0.5 × TD[min]
PD/PID	GAIN ≈ 0.4 × v_{max} × T_u [° C] TI ≈ 2 × T_u [min] TD ≈ 0.4 × T_u [min] TM_LAG ≈ 0.5 × TD[min]

Instead of $v_{max} = \Delta_x / \Delta_t$, you can use X_{max} / T_g.

In the case of controllers with PID structure the setting of the integral action time and differential-action time is usually coupled with each other.

The ratio TI / TD lies between 4 and 5 and is optimal for most controlled systems.

Non-observance of the differential-action time TD is uncritical at PD controllers.

In the case of PI and PID controllers, control oscillations occur if the integral action time TI has been select by more than half too small.

An integral action time that is too large slows down the settling times of disturbances. One cannot expect that the control loops operate "optimally" after the first parameter settings. Experience shows that adjusting is always necessary, when a system exists that is "difficult to control" with $T_u / T_g > 0.3$.

3.2 Configuring a software controller

3.2.1 Overview of software controller

For the configuration of a software controller, you need an instruction with the control algorithm and a technology object. The technology object for a software controller corresponds with the instance DB of the instruction. The configuration of the controller is saved in the technology object. In contrast to the instance DBs of other instructions, technology objects are not stored for the program resources, but rather under CPU > Technology objects.

Technology objects and instructions

CPU	Library	Instruction	Technology object	Description
S7-1200	Compact PID	PID_Compact V1.x	PID_Compact V1.x	Universal PID controller with integrated tuning
S7-1200		PID_3Step V1.x	PID_3Step V1.x	PID controller with integrated tuning for valves
S7-1500 S7-1200 V4.x		PID_Compact V2.x	PID_Compact V2.x	Universal PID controller with integrated tuning
S7-1500 S7-1200 V4.x		PID_3Step V2.x	PID_3Step V2.x	PID controller with integrated tuning for valves
S7-1500 ≥ V1.7 S7-1200 ≥ V4.1		PID_Temp V1.x	PID_Temp V1.x	Universal PID temperature controller with integrated tuning
S7-1500/300/400	PID basic functions	CONT_C	CONT_C	Continuous controller
S7-1500/300/400		CONT_S	CONT_S	Step controller for actuators with integrating behavior
S7-1500/300/400		PULSEGEN	-	Pulse generator for actuators with proportional behavior
S7-1500/300/400		TCONT_CP	TCONT_CP	Continuous temperature controller with pulse generator
S7-1500/300/400		TCONT_S	TCONT_S	Temperature controller for actuators with integrating behavior
S7-300/400	PID Self Tuner	TUN_EC	TUN_EC	Optimization of a continuous controller
S7-300/400		TUN_ES	TUN_ES	Optimization of a step controller
S7-300/400	Standard PID Control (PID Professional optional package)	PID_CP	PID_CP	Continuous controller with pulse generator
S7-300/400		PID_ES	PID_ES	Step controller for actuators with integrating behavior
S7-300/400		LP_SCHED	-	Distribute controller calls

CPU	Library	Instruction	Technology object	Description
S7-300/400	Modular PID Control (PID Professional optional package)	A_DEAD_B	-	Filter interfering signal from control deviation
S7-300/400		CRP_IN	-	Scale analog input signal
S7-300/400		CRP_OUT	-	Scale analog output signal
S7-300/400		DEAD_T	-	Delay output of input signal
S7-300/400		DEADBAND	-	Suppress small fluctuations to the process value
S7-300/400		DIF	-	Differentiate input signals over time
S7-300/400		ERR_MON		Monitor control deviation
S7-300/400		INTEG	-	Integrate input signals over time
S7-300/400		LAG1ST	-	First-order delay element
S7-300/400		LAG2ND	-	Second-order delay element
S7-300/400		LIMALARM	-	Report limit values
S7-300/400		LIMITER	-	Limiting the manipulated variable
S7-300/400		LMNGEN_C	-	Determine manipulated variable for continuous controller
S7-300/400		LMNGEN_S	-	Determine manipulated variable for step controller
S7-300/400		NONLIN	-	Linearize encoder signal
S7-300/400		NORM	-	Scale process value physically
S7-300/400		OVERRIDE	-	Switch manipulated variable from 2 PID controllers to 1 actuator
S7-300/400		PARA_CTL	-	Switch parameter sets
S7-300/400		PID	-	PID algorithm
S7-300/400		PUSLEGEN_M	-	Generate pulse for proportional actuators
S7-300/400		RMP_SOAK	-	Specify setpoint according to ramp / soak
S7-300/400		ROC_LIM	-	Limit rate of change
S7-300/400		SCALE_M	-	Scale process value
S7-300/400		SP_GEN	-	Specify setpoint manually
S7-300/400		SPLT_RAN	-	Split manipulated variable range
S7-300/400		SWITCH	-	Switch analog values
S7-300/400		LP_SCHED_M	-	Distribute controller calls

3.2.2 Steps for the configuration of a software controller

All SW-controllers are configured according to the same scheme:

Step	Description
1	Add technology object (Page 633)
2	Configure technology object (Page 634)
3	Call instruction in the user program (Page 635)
4	Download technology object to device (Page 636)
5	Commission software controller (Page 637)

Step	Description
6	Save optimized PID parameters in the project (Page 637)
7	Comparing values (Page 639)
8	Display instances of a technology object (Page 659)

3.2.3 Add technology objects

Add technology object in the project navigator

When a technology object is added, an instance DB is created for the instruction of this technology object. The configuration of the technology object is stored in this instance DB.

Requirement

A project with a CPU has been created.

Procedure

To add a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click "Add new object".
   The "Add new object" dialog box opens.
4. Click on the "PID" button.
   All available PID-controllers for this CPU are displayed.
5. Select the instruction for the technology object, for example, PID_Compact.
6. Enter an individual name for the technology object in the "Name" input field.
7. Select the "Manual" option if you want to change the suggested data block number of the instance DB.
8. Click "Further information" if you want to add own information to the technology object.
9. Confirm with "OK".

Result

The new technology object has been created and stored in the project tree in the "Technology objects" folder. The technology object is used if the instruction for this technology object is called in a cyclic interrupt OB.

Note

You can select the "Add new and open" check box at the bottom of the dialog box. This opens the configuration of the technology object after adding has been completed.

3.2.4 Configure technology objects

The properties of a technology object on a S7-1200 CPU can be configured in two ways.

- In the Inspector window of the programming editor
- In the configuration editor

The properties of a technology object on a S7-300/400 CPU can only be configured in the configuration editor.

Inspector window of the programming editor

In the Inspector window of the programming editor you can only configure the parameters required for operation.

The offline values of the parameters are also shown in online mode. You can only change the online values in the commissioning window.

To open the Inspector window of the technology object, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Double click the block (cyclic interrupt OB) in which you open the instruction of the SW-controller.
   The block is opened in the work area.
3. Click on the instruction of the SW-controller.
4. In the Inspector window, select the "Properties" and "Configuration" tabs consecutively.

Configuration window

For each technology object, there is a specific configuration window in which you can configure all properties.

To open the configuration window of the technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.

Symbols

Icons in the area navigation of the configuration and in the Inspector window show additional details about the completeness of the configuration:

	The configuration contains default values and is complete. The configuration exclusively contains default values. With these default values the use of the technology object is possible without further changes.
	The configuration contains user-defined or automatically adjusted values and is complete All input fields of the configuration contain valid values and at least one default setting was changed.
	The configuration is incomplete or faulty At least one input field or a collapsible list contains no or one invalid value. The corresponding field or the drop-down list box has a red background. When clicked, the roll-out error message indicates the cause of the error.

The properties of a technology object are described in detail in the section for the technology object.

3.2.5 Call instruction in the user program

The instruction of the software controller must be called in a cyclic interrupt OB. The sampling time of the software controller is determined by the interval between the calls in the cyclic interrupt OB.

Requirement

The cyclic interrupt OB is created and the cycle time of the cyclic interrupt OB is correctly configured.

Procedure

Proceed as follows to call the instruction in the user program:

1. Open the CPU folder in the project tree.
2. Open the "Program blocks" folder.
3. Double-click the cyclic interrupt OB.
   The block is opened in the work area.
4. Open the "Technology" group in the "Instructions" window and the "PID Control" folder.
   The folder contains all instructions for software controllers that can be configured on the CPU.
5. Select the instruction and drag it to your cyclic interrupt OB.
   The "Call options" dialog box opens.
6. Select a technology object or type the name for a new technology object from the "Name" list.

Result

If the technology object does not exist yet, it is added. The instruction is added in the cyclic interrupt OB. The technology object is assigned to this call of the instruction.

3.2.6 Downloading technology objects to device

A new or modified configuration of the technology object must be downloaded to the CPU for the online mode. The following characteristics apply when downloading retentive data:

- Software (changes only)
  - S7-1200, S7-1500:
    Retentive data is retained.
  - S7-300/400:
    Retentive data is updated immediately. CPU does not change to Stop.
- Download PLC program to device and reset
  - S7-1200, S7-1500:
    Retentive data is updated at the next change from Stop to RUN. The PLC program can only be downloaded completely.
  - S7-300/400:
    Retentive data is updated at the next change from Stop to RUN.

Downloading retentive data to an S7-1200 or S7-1500 CPU

Note

The download and reset of the PLC program during ongoing system operation can result in serious damages or injuries in the case of malfunctions or program errors.

Make sure that dangerous states cannot occur before you download and reset the PLC program.

Proceed as follows to download the retentive data:

1. Select the entry of the CPU in the project tree.
2. Select the command "Download and reset PLC program" from the "Online" menu.
   - If you have not established an online connection yet, the "Extended download" dialog opens. In this case, set all required parameters for the connection and click "Download".
   - If the online connection has been defined, the project data is compiled, if necessary, and the dialog "Load preview" opens. This dialog displays messages and recommends actions necessary for download.
3. Check the messages.
   As soon as download is possible, the "Download" button becomes active.
4. Click on "Download".
   The complete PLC program is downloaded and the "Load results" dialog opens. This dialog displays the status and the actions after the download.

5. If the modules are to restart immediately after the download, select the check box "Start all".
6. Close the dialog "Download results" with "Finish".

Result

The complete PLC program is downloaded to the device. Blocks that only exist online in the device are deleted. By downloading all affected blocks and by deleting any blocks in the device that are not required, you avoid inconsistencies between the blocks in the user program.

The messages under "Info > General" in the Inspector window indicate whether the download was successful.

3.2.7 Commissioning software controller

Procedure

To open the "Commissioning" work area of the technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Commissioning" object.

The commissioning functions are specific for each controller and are described there.

3.2.8 Save optimized PID parameter in the project

The software controller is optimized in the CPU. Through this, the values in the instance-DB on the CPU no longer agree with those in the project.

To update the PID parameter in the project with the optimized PID parameters, proceed as follows:

Requirement

- An online connection to the CPU is established and the CPU is in "RUN" mode.
- The functions of the commissioning window have been enabled by means of the "Start" button.

Procedure

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Open a technology object.
4. Double click on "Commissioning".

5. Click on the icon "Upload PID parameters".
6. Save the project.

Result

The currently active PID parameters are stored in the project data. When reloading the project data in the CPU, the optimized parameters are used.

3.2.9 Comparing values

3.2.9.1 Comparison display and boundary conditions

The "Compare values" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of actual values in the project
- Transfer of start values of the project to the CPU as actual values

Icons and operator controls

The following icons and operator controls are available:

Icon	Function
	Start value PLC matches the configured Start value project
	Start value PLC does not match the configured Start value project
	The comparison of the Start value PLC with the configured Start value project cannot be performed
	At least one of the two comparison values has a process-related or syntax error.
	Transfers actual values to the offline project
	Transfers updated start values in the project to the CPU (initialize setting values)
	Opens the "Compare values" dialog

Boundary conditions

The "Compare values" function is available for S7-1200 and S7-1500 without limitations.

The following limitation applies to S7-300 and S7-400:

In monitoring mode, an S7-300/S7-400 cannot transfer the start values to the CPU. These values cannot be displayed online with "Compare values".

The actual values of the technology object are displayed and can be changed directly.

3.2.9.2 Comparing values

The procedure is shown in the following using "PID Parameters" as an example.

Requirements

- A project with a software controller is configured.
- The project is downloaded to the CPU.
- The configuration dialog is open in the project navigator.

Procedure

1. Open the desired software controller in the project navigation.
2. Double-click the "Configuration" object.
3. Navigate within the configuration window to the "PID Parameters" dialog.
4. Click the icon to activate monitoring mode.
   The icons and operator controls (Page 638) of the "Compare values" function are shown behind the parameters.
5. Click the desired parameter in the input box and change the parameter values manually by entering them directly.
   – If the background of the input box is gray, this value is a read-only value and cannot be changed.
   – To change the values in the "PID Parameters" dialog, enable manual entry by selecting the "Enable manual entry" check box beforehand.
6. Click the icon to open the dialog for the start values.
   This dialog indicates two values of the parameter:
   – Start value in CPU: The start value in the CPU is shown in the top part.
   – Start value in the project: The configured start value in the project is shown in the bottom part.
7. Enter the desired value in the input box for the project.

Error detection

The input of incorrect values is detected. Corrections are suggested in this case.

If you enter a value with incorrect syntax, a rollout containing the corresponding error message opens below the parameter. The incorrect value is not applied.

If you enter a value that is incorrect for the process, a dialog opens containing the error message and a suggested correction:

- Click "No" to accept this suggested correction and correct your input.
- Click "OK" to apply the incorrect value.

NOTICE
Malfunctions of the controller Values incorrect for the process can result in controller malfunctions.

Backing up actual values

Click the icon to transfer the actual controller values to the start values of your configured project.

Transferring project values to the CPU

Click the icon to transfer the configured values of your project to the CPU.

CAUTION
Prevent personal injury and property damage! Downloading and resetting of the user program while the plant is operating may result in significant property damage and severe personal injuries in the event of malfunctions or program errors. Make sure that dangerous states cannot occur before you download and reset the user program.

3.2.10 Parameter view

3.2.10.1 Introduction to the parameter view

The Parameter view provides you with a general overview of all relevant parameters of a technology object. You obtain an overview of the parameter settings and can easily change them in offline and online mode.

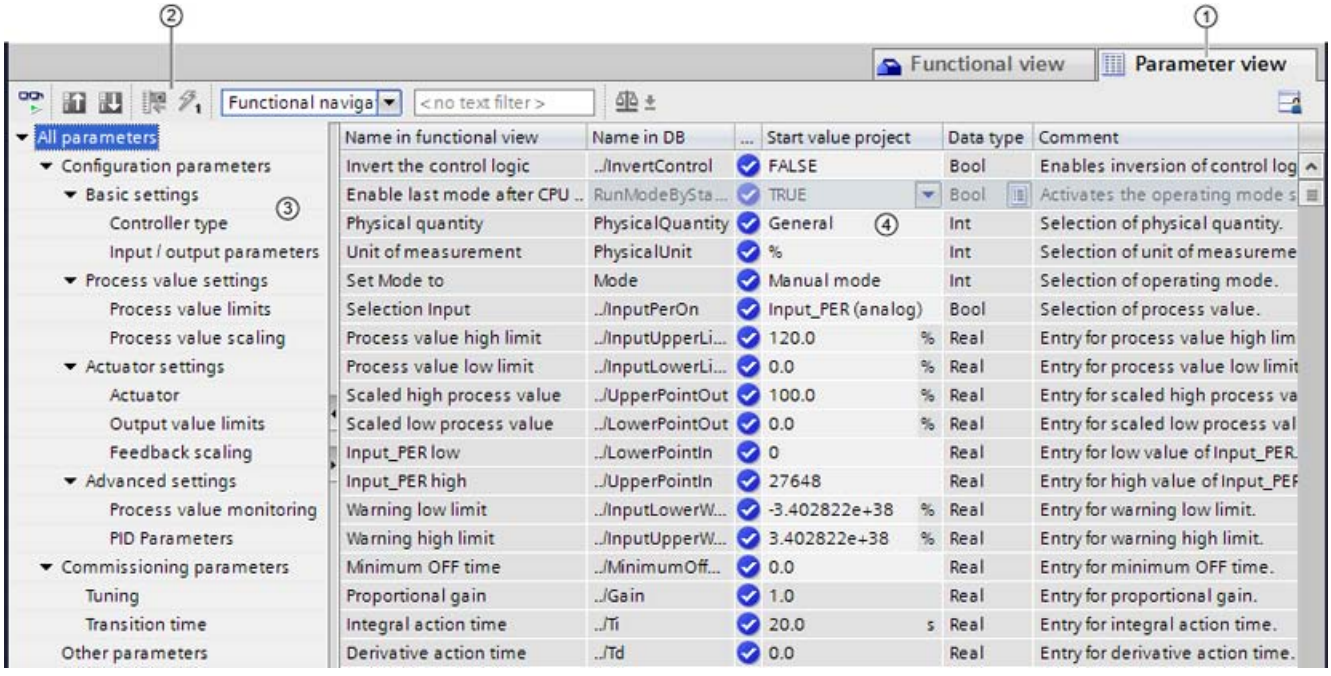

① "Parameter view" tab

② Toolbar (Page 643)

③ Navigation (Page 643)

④ Parameter table (Page 644)

Function scope

The following functions are available for analyzing the parameters of the technology objects and for enabling targeted monitoring and modification.

Display functions:

- Display of parameter values in offline and online mode
- Display of status information of the parameters
- Display of value deviations and option for direct correction
- Display of configuration errors
- Display of value changes as a result of parameter dependencies

- Display of all memory values of a parameter: Start value PLC, Start value project, Monitor value
- Display of the parameter comparison of the memory values of a parameter

Operator control functions:

- Navigation for quickly changing between the parameters and parameter structures.
- Text filter for faster searches for particular parameters.
- Sorting function for customizing the order of parameters and parameter groups to requirements.
- Memory function for backing up structural settings of the Parameter view.
- Monitoring and modifying of parameter values online.
- Change display format of value.
- Function for saving a snapshot of parameter values of the CPU in order to capture momentary situations and to respond to them.
- Function for applying a snapshot of parameter values as start values.
- Download of modified start values to the CPU.
- Comparison functions for comparing parameter values with one another.

Validity

The Parameter view described here is available for the following technology objects:

- PID_Compact
- PID_3Step
- PID_Temp
- CONT_C (S7-1500 only)
- CONT_S (S7-1500 only)
- TCONT_CP (S7-1500 only)
- TCONT_S (S7-1500 only)
- TO_Axis_PTO (S7-1200 Motion Control)
- TO_Positioning_Axis (S7-1200 Motion Control)
- TO_CommandTable_PTO (S7-1200 Motion Control)
- TO_CommandTable (S7-1200 Motion Control)

3.2.10.2 Structure of the parameter view

Toolbar

The following functions can be selected in the toolbar of the parameter view.

Icon	Function	Explanation
	Monitor all	Starts the monitoring of visible parameters in the active Parameter view (online mode).
	Create snapshot of monitor values and accept setpoints of this snapshot as start values	Applies the current monitor values to the “Snapshot” column and updates the start values in the project. Only in online mode for PID_Compact, PID_3Step and PID_Temp.
	Initialize setpoints	Transfers the start values updated in the project to the CPU. Only in online mode for PID_Compact, PID_3Step and PID_Temp.
	Create snapshot of monitor values	Applies the current monitor values to the “Snapshot” column. Only in online mode.
	Modify all selected parameters immediately and once	This command is executed once and as quickly as possible without reference to any particular point in the user program. Only in online mode.
Functional naviga	Select navigation structure	Toggles between functional navigation and data navigation.
< no text filter >	Text filter...	After entry of a character string: Display of all parameters containing the specified string in one of the currently visible columns.
	Selection of compare values	Selection of parameter values that are to be compared with one another in online mode (Start value project, Start value PLC, Snapshot) Only in online mode.
	Save window settings	Saves your display settings for the Parameter view (e.g., selected navigation structure, activated table columns, etc.)

Navigation

Within the "Parameter view" tab, the following alternative navigation structures can be selected.

Navigation		Explanation
Functional navigation	All parameters Configuration parameters Commissioning parameters Other parameters	In the functional navigation, the structure of the parameters is based on the structure in the configuration dialog ("Functional view" tab), commissioning dialog, and diagnostics dialog. The last group "Other parameters" contains all other parameters of the technology object.
Data navigation	All parameters Input Output InOut Static Other parameters	In the data navigation, the structure of the parameters is based on the structure in the instance DB / technology DB. The last group "Other parameters" contains the parameters that are not contained in the instance DB / technology DB.

You can use the "Select navigation structure" drop-down list to toggle the navigation structure.

Parameter table

The table below shows the meaning of the individual columns of the parameter table. You can show or hide the columns as required.

- Column "Offline" = X: Column is visible in offline mode.
- Column "Online" = X: Column is visible in online mode (online connection to the CPU).

Column	Explanation	Offline	Online
Name in functional view	Name of the parameter in the functional view. The display field is empty for parameters that are not configured via the technology object.	X	X
Full name in DB	Complete path of the parameter in the instance DB / technology DB. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Name in DB	Name of the parameter in the instance DB / technology DB. If the parameter is part of a structure or UDT, the prefix ". ./" is added. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Status of configuration	Display of the completeness of the configuration using status symbols. see Status of configuration (offline) (Page 652)	X	
Compare result	Result of the "Compare values" function. This column is shown if there is an online connection and the "Monitor all" button is selected.		X
Start value project	Configured start value in the project. Error indication if entered values have a syntax or process-related error.	X	X
Default value	Value that is pre-assigned to the parameter. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Snapshot	Snapshot of the current values in the CPU (monitor values). Error indication if values have a process-related error.	X	X
Start value PLC	Start value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Monitor value	Current value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Modify value	Value that is to be used to change the monitor valuet. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if entered values have a syntax or process-related error.		X
Selection for transmission	Selection of the Modify values that are to be transmitted using the "Modify all selected parameters immediately and once" button. This column is displayed together with the "Modify value" column.		X

Column	Explanation	Offline	Online
Minimum value	Minimum process-related value of the parameter. If the minimum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Maximum value	Maximum process-related value of the parameter. If the maximum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Setpoint	Designates the parameter as a setpoint. These parameters can be initialized online.	X	X
Data type	Data type of the parameter. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Retain	Designates the value as a retentive value. The values of retentive parameters are retained even after the voltage supply is switched off.	X	X
Accessible from HMI	Indicates whether the HMI can access this parameter during runtime.	X	X
Visible in HMI	Indicates whether the parameter is visible in the selection list of the HMI by default.	X	X
Comment	Brief description of the parameter.	X	X

See also

Comparing values (Page 638)

3.2.10.3 Opening the parameter view

Requirement

The technology object has been added in the project tree, i.e., the associated instance DB / technology DB of the instruction has been created.

Procedure

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.
4. Select the "Parameter view" tab in the top right corner.

Result

The Parameter view opens. Each displayed parameter is represented by one row in the parameter table.

The displayable parameter properties (table columns) vary depending on whether you are working with the Parameter view in offline or online mode.

In addition, you can selectively display and hide individual table columns.

See also

Default setting of the parameter view (Page 646)

3.2.10.4 Default setting of the parameter view

Default settings

To enable you to work efficiently with the Parameter view, you can customize the parameter display and save your settings.

The following customizations are possible and can be saved:

- Show and hide columns
- Change column width
- Change order of the columns
- Toggle navigation
- Select parameter group in the navigation
- Selection of compare values

Show and hide columns

To show or hide columns in the parameter table, follow these steps:

1. Position the cursor in the header of the parameter table.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the check box for the column.
4. To hide a column, clear the check box for the column.

or

1. Position the cursor in the header of the parameter table.
2. Select the "Show all columns" command in the shortcut menu if all columns of the offline or online mode are to be displayed.

Some columns can only be displayed in online mode: see Parameter table (Page 644).

Change column width

To customize the width of a column so that all texts in the rows can be read, follow these steps:

1. Position the cursor in the header of the parameter table to the right of the column to be customized until the shape of the cursor changes to a cross.
2. Then double-click this location.

or

1. Open the shortcut menu on the header of the parameter table.
2. Click
   - "Optimize column width" or
   - "Optimize width of all columns".

If the column width setting is too narrow, the complete content of individual fields are shown if you hover the cursor briefly over the relevant field.

Change order of the columns

The columns of the parameter table can be arranged in any way.

To change the order of the columns, follow these steps:

1. Click on the column header and use a drag-and-drop operation to move it to the desired location.
   When you release the mouse button, the column is anchored to the new position.

Toggle navigation

To toggle the display form of the parameters, follow these steps:

1. Select the desired navigation in the “Select navigation structure” drop-down list.
   - Data navigation
   - Functional navigation

See also Navigation (Page 643).

Select parameter group in the navigation

Within the selected navigation, you choose between the “All parameters” display or the display of a subordinate parameter group of your choice.

1. Click the desired parameter group in the navigation.
   The parameter table only displays the parameters of the parameter group.

Selection of compare values (online)

To set the compare values for the “Compare values” function, follow these steps:

1. Select the desired compare values in the “Selection of compare values” drop-down list.
   – Start value project / Start value PLC
   – Start value project / Snapshot
   – Start value PLC / Snapshot

The “Start value project / Start value PLC” option is set by default.

Saving the default setting of the Parameter view

To save the above customizations of the Parameter view, follow these steps:

1. Customize the Parameter view according to your requirements.
2. Click the “Save window settings” button at the top right of the Parameter view.

3.2.10.5 Working with the parameter view

Overview

The following table provides an overview of the functions of the Parameter view in online and offline mode described in the following.

- Column "Offline" = X: This function is possible in offline mode.
- Column "Online" = X: This function is possible in online mode.

Function/action	Offline	Online
Filtering the parameter table (Page 649)	X	X
Sorting the parameter table (Page 649)	X	X
Transferring parameter data to other editors (Page 650)	X	X
Indicating errors (Page 650)	X	X
Editing start values in the project (Page 651)	X	X
Status of configuration (offline) (Page 652)	X	
Monitoring values online in the parameter view (Page 652)		X
Create snapshot of monitor values (Page 654)		X
Modifying values (Page 655)		X
Comparing values (Page 656)		X
Applying values from the online program as start values (Page 657)		X
Initializing setpoints in the online program (Page 658)		X

Filtering the parameter table

You can filter the parameters in the parameter table in the following ways:

- With the text filter
- With the subgroups of the navigation

Both filter methods can be used simultaneously.

With the text filter

Texts that are visible in the parameter table can be filtered. This means only texts in displayed parameter rows and columns can be filtered.

1. Enter the desired character string for filtering in the “Text filter...” input box.
   The parameter table displays only the parameters containing the character string.

The text filtering is reset.

- When another parameter group is selected in the navigation.
- When navigation is changed from data navigation to functional navigation, or vice versa.

With the subgroups of the navigation

1. Click the desired parameter group in the navigation, e.g., "Static".
   The parameter table only shows the static parameters. You can select further subgroups for some groups of the navigation.
2. Click “All parameters” in the navigation if all parameters are to be shown again.

Sorting the parameter table

The values of the parameters are arranged in rows. The parameter table can be sorted by any displayed column.

- In columns containing numerical values, sorting is based on the magnitude of the numerical value.
- In text columns, sorting is alphabetical.

Sorting by column

1. Position the cursor in the header cell of the desired column.
   The background of this cell turns blue.
2. Click the column header.

Result

The entire parameter table is sorted by the selected column. A triangle with tip facing up appears in the column header.

Clicking the column header again changes the sorting as follows:

- Symbol "▲": Parameter table is sorted in ascending order.
- Symbol "▼": Parameter table is sorted in descending order.
- No symbol: The sorting is removed again. The parameter table assumes the default display.

The "../" prefix in the "Name in DB" column is ignored when sorting.

Transferring parameter data to other editors

After selecting an entire parameter row of the parameter table, you can use the following:

- Drag-and-drop
- <Ctrl+C>/<Ctrl+V>
- Copy/Paste via shortcut menu

Transfer parameters to the following editors of the TIA Portal:

- Program editor
- Watch table
- Signal table for trace function

The parameter is inserted with its full name: See information in "Full name in DB" column.

Indicating errors

Error indication

Parameter assignment errors that result in compilation errors (e.g. limit violation) are indicated in the Parameter view.

Every time a value is input in the Parameter view, a check is made for process-related and syntax errors and the result is indicated.

Bad values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Table field with red background
  If you click the bad field, a roll-out error message appears with information of the permissible value range or the required syntax (format)

Compilation error

From the error message of the compiler, you can directly open the Parameter view (functional navigation) containing the parameter causing the error in situations where the parameter is not displayed in the configuration dialog.

Editing start values in the project

With the Parameter view, you can edit the start values in the project in offline mode and online mode.

- You make value changes in the “Start value project” column of the parameter table.
- In the “Status of configuration” column of the parameter table, the progress of the configuration is indicated by the familiar status symbols from the configuration dialog of the technology object.

Boundary conditions

- If other parameters depend on the parameter whose start value was changed, the start value of the dependent parameters are also adapted.
- If a parameter of a technology object is not editable, it is also not editable in the parameter view. The ability to edit a parameter can also depend on the values of other parameters.

Defining new start values

To define start values for parameters in the Parameter view, follow these steps:

1. Open the Parameter view of the technology object.
2. Enter the desired start values in the "Start value project" column. The value must match the data type of the parameter and must not exceed the value range of the parameter. The limits of the value range can be seen in the “Maximum value” and “Minimum value” columns.

The "Status of configuration" column indicates the progress of the configuration with colored symbols.

See also Status of configuration (offline) (Page 652)

Following adaptation of the start values and downloading of the technology object to the CPU, the parameters take the defined value at startup if they are not declared as retentive (“Retain” column).

Error indication

When a start value is input, a check is made for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Red background in the “Start value project” field
  If you click on the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Correcting bad start values

1. Correct bad start values using information from the roll-out error message.
   Red error symbol, red field background, and roll-out error message are no longer displayed.

The project cannot be successfully compiled unless the start values are error-free.

Status of configuration (offline)

The status of the configuration is indicated by icons:

- In the “Status of configuration” column in the parameter table
- In the navigation structure of the functional navigation and data navigation

Symbol in “Status of configuration” column

Symbol	Meaning
	The start value of the parameter corresponds to the default value and is valid. A start value has not yet been defined by the user.
	The start value of the parameter contains a value defined by the user or an automatically adjusted value. The start value is different than the default value. The start value is error-free and valid.
	The start value of the parameter is invalid (syntax or process-related error). The input box has a red background. When clicked, the roll-out error message indicates the cause of the error.
	Only for S7-1200 Motion Control: The start value of the parameter is valid but contains warnings. The input box has a yellow background.
	The parameter is not relevant in the current configuration.

Symbol in the navigation

The symbols in the navigation indicate the progress of the configuration in the same way as in the configuration dialog of the technology object.

See also

Configure technology objects (Page 634)

Monitoring values online in the parameter view

You can monitor the values currently taken by the parameters of the technology object in the CPU (monitor values) directly in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.

- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

Procedure

1. Start the monitoring by clicking .
   As soon as the Parameter view is online, the following columns are additionally displayed:
   – Compare result
   – Start value PLC
   – Monitor value
   – Modify value
   – Selection for transmission

   The "Monitor value" column shows the current parameter values on the CPU.
   Meaning of the additional columns: see Parameter table (Page 644)
2. Stop the monitoring by clicking again.

Display

All columns that are only available online have an orange background:

- Values in light-orange cells can be changed.
- Values in cells with a dark orange background cannot be changed.

Change display format of value

The display format of the value can be selected via the shortcut menu of a table row in the Parameter view of the technology object.

The display format of the following values can be changed both in online mode and in offline mode:

- Start value project
- Start value PLC
- Maximum value
- Minimum value
- Snapshot
- Monitor value
- Default value
- Modify value

The set display format applies to all values of the table row.

The following display formats of the value can be changed:

- Default
- Hex
- Octal
- Bin
- Dec (+/-)
- DEC

Depending on the parameter selected in the parameter view, only the supported display formats can be selected.

Requirements

- The Parameter view of the technology object is open.

Procedure

To change the display format of the value, proceed as follows:

1. Select one or more table rows in which you want to change the display format.
2. Select the "Display format" command in the shortcut menu.
3. Select the desired display format.

Note

To change the display format of a certain data type in multiple table rows, sort the Parameter view by this data type. Then select the first and last table row with this data type while keeping the <Shift> key pressed and change the display format for the selected table rows.

Create snapshot of monitor values

You can back up the current values of the technology object on the CPU (monitor values) and display them in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To show the current parameter values, follow these steps:

1. In the Parameter view, click the “Create snapshot of monitor values" icon .

Result

The current monitor values are transferred once to the "Snapshot" column of the parameter table.

You can analyze the values "frozen" in this way while the monitor values continue to be updated in the "Monitor values" column.

Modifying values

With the Parameter view, you can modify values of the technology object in the CPU.

You can assign values to the parameter once (Modify value) and modify them immediately. The modify request is executed as quickly as possible without reference to any particular point in the user program.

DANGER

Danger when modifying:

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you use the "Modify" function.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.
- The parameter can be modified (associated field in the "Modify value" column has a light-orange background).

Procedure

To modify parameters immediately, follow these steps:

1. Enter the desired modify values in the “Modify values” column of the parameter table.
2. Check whether the check box for modifying is selected in the "Select for transmission" column.
   The modify values and associated check boxes of dependent parameters are automatically adapted at the same time.
3. Click the “Modify all selected parameters immediately and once” icon .

The selected parameters are modified once and immediately with the specified values and can be monitored in the "Modify values" column. The check boxes for modifying in the "Selection for transmission" column are automatically cleared after the modify request is complete.

Error indication

When a start value is input, a check is made immediately for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red background in the “Modify value” field

and

- If you click the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Bad modify values

- Modify values with process-related errors can be transmitted.
- Modify values with syntax errors **cannot** be transmitted.

Comparing values

You can use comparison functions to compare the following memory values of a parameter:

- Start value project
- Start value PLC
- Snapshot

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To compare the start values on the various target systems, follow these steps:

1. Click the "Selection of compare values" icon .
   A selection list containing the comparison options opens:
   - Start value project - Start value PLC (default setting)
   - Start value project - Snapshot
   - Start value PLC - Snapshot
2. Select the desired comparison option.
   The selected comparison option is executed as follows:
   - A scales symbol appears in the header cells of the two columns selected for comparison.
   - Symbols are used in the "Compare result" column to indicate the result of the comparison of the selected columns.

Symbol in "Compare result" column

Symbol	Meaning
	The compare values are equal and error-free.
	The compare values are not equal and error-free.
	At least one of the two compare values has a process-related or syntax error.
	The comparison cannot be performed. At least one of the two comparison values is not available (e.g. snapshot).
	Comparison of the value is inappropriate since it is not relevant in one of the configurations.

Symbol in the navigation

The symbols are shown in the same way in the navigation if the comparison result applies to at least one of the parameters below the displayed navigation structure.

Applying values from the online program as start values

In order to apply optimized values from the CPU to the project as start values, you create a snapshot of the monitor values. Values of the snapshot marked as a "Setpoint" are then applied to the project as start values.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").

- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To apply optimized values from the CPU, follow these steps:

1. Click the "Create snapshot of monitor values and accept setpoints of this snapshot as start values" icon .

Result

The current monitor values are applied to the "Snapshot" column and their setpoints are copied to the "Start value project" column as new start values.

Note

Applying values of individual parameters

You can also apply the values of individual parameters that are not marked as a setpoint from the "Snapshot" column to the "Start values project" column. To do so, copy the values and insert them into the "Start value project" column using the "Copy" and "Paste" commands in the shortcut menu.

Initializing setpoints in the online program

You can initialize all parameters that are marked as a "Setpoint" in the Parameter view with new values in the CPU in one step. In so doing, the start values are downloaded from the project to the CPU. The CPU remains in "RUN" mode.

To avoid data loss on the CPU during a cold restart or warm restart, you must also download the technology object to the CPU.

DANGER

Danger when changing parameter values

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you reinitialize the setpoints.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").

- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.
- The parameters marked as a "Setpoint" have a "Start value project" that is free of process-related and syntax errors

Procedure

To initialize all setpoints, follow these steps:

1. Enter the desired values in the "Start value project" column.
   Ensure that the start values are free of process-related and syntax errors.
2. Click the "Initialize setpoints" icon .

Result

The setpoints in the CPU are initialized with the start values from the project.

3.2.11 Display instance DB of a technology object.

An instance DB, in which the parameter and static variables are saved, is created for each technology object.

Procedure

To display the instance DB of a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Highlight a technology object.
4. Select the command "Open DB editor" in the shortcut menu.

3.3 PID control (S7-1200, S7-1500) (S7-1200, S7-1500)

3.3.1 Using PID_Compact (S7-1200, S7-1500)

3.3.1.1 Technology object PID_Compact (S7-1200, S7-1500)

The technology object PID_Compact provides a continuous PID controller with integrated optimization. You can alternatively configure a pulse controller. Both manual and automatic mode are possible.

PID-Compact continuously acquires the measured process value within a control loop and compares it with the required setpoint. From the resulting control deviation, the instruction PID_Compact calculates an output value by which the process value is adapted as quickly and stable as possible to the setpoint. The output value for the PID controller consists of three actions:

- **Proportional** action
  The proportional action of the output value increases in proportion to the control deviation.
- **I** action
  The integral action of the output value increases until the control deviation has been balanced.
- **D** action
  The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID_Compact calculates the proportional, integral and derivative parameters for your controlled system during pretuning. Fine tuning can be used to tune the parameters further. You do not need to manually determine the parameters.

Additional information

- Overview of software controller (Page 631)
- Add technology objects (Page 633)
- Configure technology objects (Page 634)
- Configuring PID_Compact V2 (Page 661)
- Configuring PID_Compact V1 (Page 680)

FAQ

For more information, see the following FAQs in the Siemens Industry Online Support:

- Entry ID 79047707 (https://support.industry.siemens.com/cs/ww/en/view/79047707)

3.3.1.2 PID_Compact V2 (S7-1200, S7-1500)

Configuring PID_Compact V2 (S7-1200, S7-1500)

Basic settings V2 (S7-1200, S7-1500)

Introduction V2 (S7-1200, S7-1500)

Configure the following properties of the "PID_Compact" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)

Setpoint, process value and output value

You can only configure the setpoint, process value and output value in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Control mode V2 (S7-1200, S7-1500)

Physical quantity

Select the physical quantity and unit of measurement for setpoint, process value, and disturbance variable in the "Controller type" group. Setpoint, process value, and disturbance variable is displayed in this unit of measurement.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_Compact does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Startup characteristics

1. To switch to "Inactive" mode after CPU restart, clear the "Activate Mode after CPU restart" check box.
   To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.
2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.
   After a complete download to the device, PID_Compact starts in the selected operating mode. With each additional restart, PID_Compact starts in the mode that was last saved in Mode.

Example

You have selected the "Activate Mode after CPU restart" check box and the entry "Pretuning" in the "Set Mode to" list. After a complete download to the device, PID_Compact starts in the "Pretuning" mode. If pretuning is still active, PID_Compact starts in "Pretuning" mode again after restart of the CPU. If pretuning was successfully completed and automatic mode is active, PID_Compact starts in "Automatic mode" after restart of the CPU.

Setpoint V2 (S7-1200, S7-1500)

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value V2 (S7-1200, S7-1500)

PID_Compact will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Output value V2 (S7-1200, S7-1500)

PID_Compact offers three output values. Your actuator will determine which output value you use.

- Output_PER
  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output
  The output value needs to be processed by the user program, for example because of nonlinear actuator response.
- Output_PWM
  The actuator is controlled via a digital output. Pulse width modulation creates minimum ON and minimum OFF times.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output_PER (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to process the output value using the user program:

1. Select the entry "Output" in the drop-down list "Output".
2. Select "Instance DB".
   The calculated output value is saved in the instance data block.
3. For the preparation of the output value, use the output parameter Output.
4. Transfer the processed output value to the actuator via a digital or analog CPU output.

Proceed as follows to use the digital output value:

1. Select the entry "Output_PWM" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the digital output.

Process value settings V2 (S7-1200, S7-1500)

Process value scaling V2 (S7-1200, S7-1500)

If you have configured the use of Input_PER in the basic setting, you must convert the value of the analog input to the physical quantity of the process value. The current configuration is displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are stored in the hardware configuration. To use the value pairs from the hardware configuration, follow these steps:

1. Select the PID_Compact instruction in the programming editor.
2. Interconnect Input_PER with an analog input in the basic settings.
3. Click the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

Process value limits V2 (S7-1200, S7-1500)

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can configure how PID_Compact reacts to an error in automatic mode in the output value settings.

Advanced settings V2 (S7-1200, S7-1500)

Process value monitoring V2 (S7-1200, S7-1500)

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_Compact instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98 °C; warning high limit = 90 °C

Warning low limit = 10 °C; process value low limit = 0 °C

PID_Compact will respond as follows:

Process value	InputWarning_H	InputWarning_L	ErrorBits	Operating mode
> 98 °C	TRUE	FALSE	0001h	Inactive or Substitute output value with error monitoring
≤ 98 °C and > 90 °C	TRUE	FALSE	0000h	Automatic mode
≤ 90 °C and ≥ 10 °C	FALSE	FALSE	0000h	Automatic mode
< 10 °C and ≥ 0 °C	FALSE	TRUE	0000h	Automatic mode
< 0 °C	FALSE	TRUE	0001h	Inactive or Substitute output value with error monitoring

In the output value settings, you can specify the reaction of PID_Compact when the process value high limit or low limit is violated.

PWM limits V2 (S7-1200, S7-1500)

The value at the output parameter Output is transformed into a pulse sequence that is output at output parameter Output_PWM by means of a pulse width modulation. Output is calculated in the PID algorithm sampling time, Output_PWM is output in the PID_Compact sampling time.

The PID algorithm sampling time is determined during pretuning or fine tuning. If manually setting the PID parameters, you will also need to configure the PID algorithm sampling time. The PID_Compact sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the value at Output and is always an integer multiple of the PID_Compact sampling time.

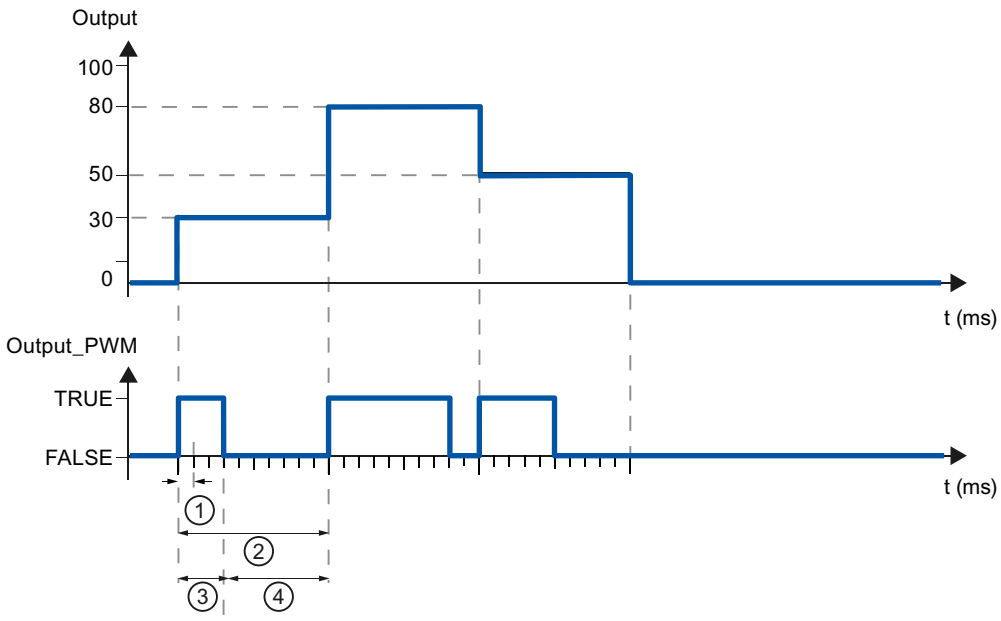

① PID_Compact sampling time
② PID algorithm sampling time
③ Pulse duration
④ Break time

The "Minimum ON time" and the "Minimum OFF time" are rounded to an integer multiple of the PID_Compact sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

Example

PID_Compact sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

Output is a constant 15%. The smallest pulse that PID_Compact can output is 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.

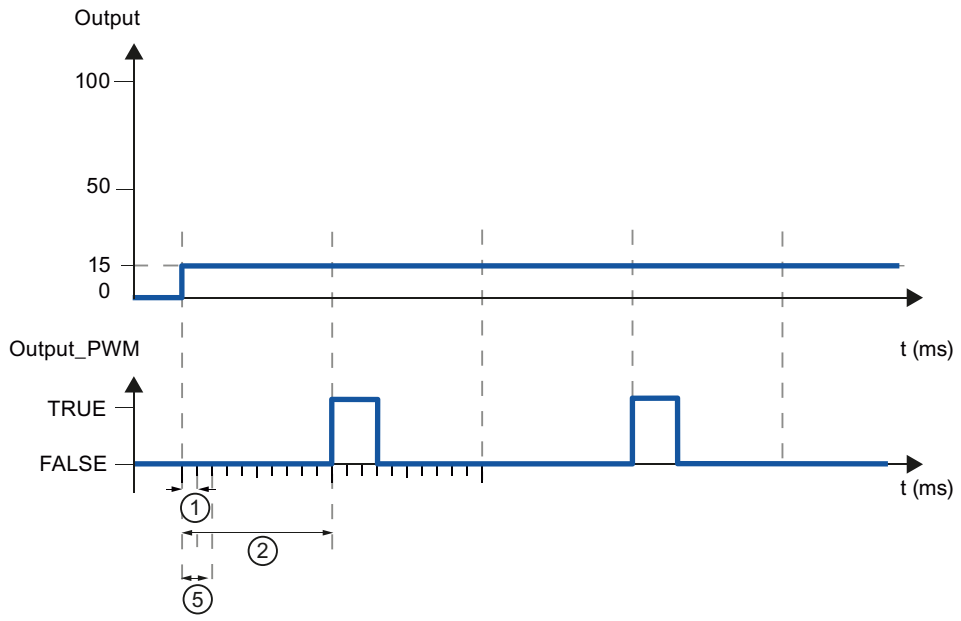

① PID_Compact sampling time
② PID algorithm sampling time
⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you are using "Output" or "Output_PER", you must configure the value 0.0 for the minimum ON and OFF times.

Note

The minimum ON and OFF times only affect the output parameter Output_PWM and are not used for any pulse generators integrated in the CPU.

Output value V2 (S7-1200, S7-1500)

Output value limits

In the "Output value limits" configuration window, configure the absolute limits of your output value in percent. Absolute output value limits are not violated in neither manual mode nor automatic mode. If an output value outside the limits is specified in manual mode, the effective value is limited in the CPU to the configured limits.

The output value limits must match the control logic.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0%
Output_PER	-100.0 to 100.0%
Output_PWM	0.0 to 100.0%

Reaction to error

NOTICE
Your system may be damaged. If you output "Current value while error pending " or "Substitute output value while error pending" in the event of an error, PID_Compact remains in automatic mode. This may cause a violation of the process value limits and damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID_Compact is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

PID_Compact generates a programmable output value in response to an error:

- Zero (inactive)
  PID_Compact outputs 0.0 as output value for all errors and switches to "Inactive" mode. The controller is only reactivated by a falling edge at Reset or a rising edge at ModeActivate.
- Current value while error is pending
  If the following errors occur in **automatic mode**, PID_Compact returns to automatic mode as soon as the errors are no longer pending.
  If one or more of the following errors occur, PID_Compact stays in automatic mode:
  - 0001h: The "Input" parameter is outside the process value limits.
  - 0800h: Sampling time error
  - 40000h: Invalid value at Disturbance parameter.

  If one or more of the following errors occur in **automatic mode**, PID_Compact switches to "Substitute output value with error monitoring" mode and outputs the last valid output value:
  - 0002h: Invalid value at Input_PER parameter.
  - 0200h: Invalid value at Input parameter.
  - 0400h: Calculation of output value failed.
  - 1000h: Invalid value at Setpoint parameter.

  If an error occurs in **manual mode**, PID_Compact continues using the manual value as the output value. If the manual value is invalid, the substitute output value is used. If the manual value and substitute output value are invalid, the output value low limit is used.
  If the following error occurs during a **pretuning or fine tuning**, PID_Compact remains in active mode:
  - 0020h: Pretuning is not permitted during fine tuning.

  When any other error occurs, PID_Compact cancels the tuning and switches to the mode from which tuning was started.
  As soon as no errors are pending, PID_Compact returns to automatic mode.
- Substitute output value while error is pending
  PID_Compact outputs the substitute output value.
  If the following error occurs, PID_Compact stays in "Substitute output value with error monitoring" mode and outputs the output value low limit:
  - 20000h: Invalid value at SubstituteOutput tag.

  For all other errors, PID_Compact reacts as described for "Current value while error is pending".

PID parameters V2 (S7-1200, S7-1500)

The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

Note

The currently active PID parameters are located for PID_Compact V1 in the sRet structure and for PID_Compact V2 in the Retain.CtrlParams structure.

Change the currently active PID parameters only in "Inactive" mode online to prevent malfunction of the PID controller.

If you want to change the PID parameters in "Automatic mode" or "Manual mode" online, change the PID parameters as follows:

- PID_Compact V1: Change the PID parameters in the sBackUp structure and apply these changes with sPid_Cmpt.b_LoadBackUp = TRUE to the sRet structure.
- PID_Compact V2: Change the PID parameters in the CtrlParamsBackUp structure and apply these changes with LoadBackUp = TRUE to the Retain.CtrlParams structure.

Online changes to the PID parameters in "Automatic mode" can result in jumps at the output value.

The PID algorithm operates according to the following equation:

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

y	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:

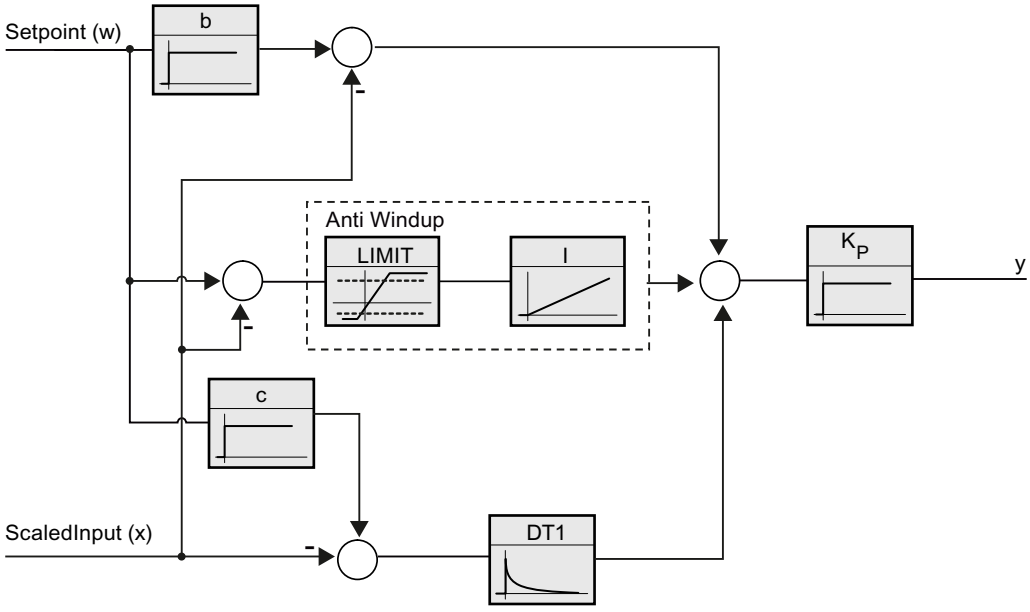

All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_Compact.

Downloading technology objects to device (Page 636)

Proportional gain

The value specifies the proportional gain of the controller. PID_Compact does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0. When the integral action time is changed from a different value to 0.0 online in "Automatic mode", the previous integral action is deleted and the output value jumps.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

If you use Output_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The PID algorithm sampling time corresponds to the time period of the pulse width modulation. The cycle time should be at least 10 times the PID algorithm sampling time.

Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list.

- **PID**
  Calculates PID parameters during pretuning and fine tuning.
- **PI**
  Calculates PI parameters during pretuning and fine tuning.
- **User-defined**
  The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program.

See also

Selection of the controller structure for specified controlled systems (Page 629)

Commissioning PID_Compact V2 (S7-1200, S7-1500)

Pretuning V2 (S7-1200, S7-1500)

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The PID parameters are backed up before being recalculated.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- PID_Compact is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is > 50% of the setpoint.

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_Compact > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list.
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it can be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_Compact responds with the configured reaction to errors.

Fine tuning V2 (S7-1200, S7-1500)

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

Requirement

- The PID_Compact instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The setpoint and the process value lie within the configured limits.
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.

- No disturbances are expected.
- PID_Compact is in one of the following operating modes: Inactive, automatic mode, or manual mode.

Process depends on initial situation

Fine tuning can be started from the following operating modes: "Inactive", "automatic mode", or "manual mode". Fine tuning proceeds as follows when started from:

- Automatic mode
  Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.
  PID_Compact controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact responds with the configured reaction to errors.
  An attempt is made to reach the setpoint with the minimum or maximum output value if the process value for pretuning is already too near the setpoint. This can produce increased overshoot.

Procedure

To perform fine tuning, follow these steps:

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed that tuning is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If no errors occurred during fine tuning, the PID parameters have been tuned. PID_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during "fine tuning", PID_Compact responds with the configured response to errors.

"Manual" mode V1 (S7-1200, S7-1500)

The following section describes how you can use the "manual mode" operating mode in the commissioning window of the "PID_Compact" technology object. Manual mode is also possible when an error is pending.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established and the CPU is in the "RUN" mode.

Procedure

Use "Manual mode" in the commissioning window if you want to test the controlled system by specifying a manual value. To define a manual value, follow these steps:

1. Click the "Start" icon.
2. Select the "Manual mode" check box in the "Online status of controller" area.
   PID_Compact operates in manual mode. The most recent current output value remains in effect.
3. Enter the manual value in the "Output" field as a % value.
4. Click the icon.

Result

The manual value is written to the CPU and immediately goes into effect.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller. The switchover to automatic mode is bumpless.

Override control with PID_Compact V2 (S7-1200, S7-1500)

Override control

In case of override control, two or more controllers share one actuator. Only one controller has access to the actuator at any time and influences the process.

A logic operation decides which controller has access to the actuator. This decision is often made based on a comparison of the output values of all controllers, for example, in case of a maximum selection, the controller with the largest output value gets access to the actuator.

The selection based on the output value requires that all controllers operate in automatic mode. The controllers that do not have an effect on the actuator are updated. This is necessary to prevent windup effects and their negative impacts on the control response and the switchover between the controllers.

PID_Compact supports override controls as of version 2.3 by offering a simple process for updating the controllers that are not active:

- By using the OverwriteInitialOutputValue and PIDCtrl.PIDInit tags, you can preassign the integral action of the controller in automatic mode as though the PID algorithm had calculated Output = OverwriteInititalOutputValue for the output value in the last cycle.
- To do this, OverwriteInitialOutputValue is interconnected with the output value of the controller that currently has access to the actuator.
- By setting the bit PIDCtrl.PIDInit, you trigger the pre-assignment of the integral action as well as the restart of the controller cycle and the PWM period.
- The subsequent calculation of the output value in the current cycle takes place based on the pre-assigned (and synchronized for all controllers) integral action as well as the proportional action and integral action from the current control deviation.
- The derivative action is not active during the call with PIDCtrl.PIDInit = TRUE and therefore does not contribute to the output value.

This procedure ensures that the calculation of the current output value and thus the decision on which controller is to have access to the actuator is only based on the current process state and the PI parameters. Windup effects for controllers that are not active and thus incorrect decisions of the switchover logic are prevented.

Requirements

- PIDCtrl.PIDInit is only effective if the integral action is activated (Retain.CtrlParams.Ti tag > 0.0).
- You must assign PIDCtrl.PIDInit and OverwriteInitialOutputValue in your user program yourself (see example below). PID_Compact does not automatically change these tags.
- PIDCtrl.PIDInit is only effective when PID_Compact is in automatic mode (parameter State = 3)
- If possible, select the sampling time of the PID algorithm (Retain.CtrlParams.Cycle tag) in such a way that it is identical for all controllers, and call all controllers in the same cyclic interrupt OB. In this way, you ensure that the switchover does not take place within a controller cycle or a PWM period.

Note

Constant adaptation of the output value limits

Instead of the active updating of the controllers without access to the actuator described here, this is implemented alternatively by constant adaptation of the output value limits in other controller systems.

This is not possible with PID_Compact, because a change of the output value limits is not supported in automatic mode.

Example: Control of a gas pipeline

PID_Compact is used for control of a gas pipeline.

The main goal is to control the flow rate Input1. The controller PID_Compact_1 is used for this purpose. In addition, the pressure Input2 (measured in flow direction in front of the valve) is to be kept below the high limit with the limiting controller PID_Compact_2.

Flow rate and pressure are controlled by a single solenoid valve. The output value of the controller corresponds to the valve opening: The valve is opened when the output value increases. This means the flow rate increases (normal control logic) while the pressure drops (inverted control logic).

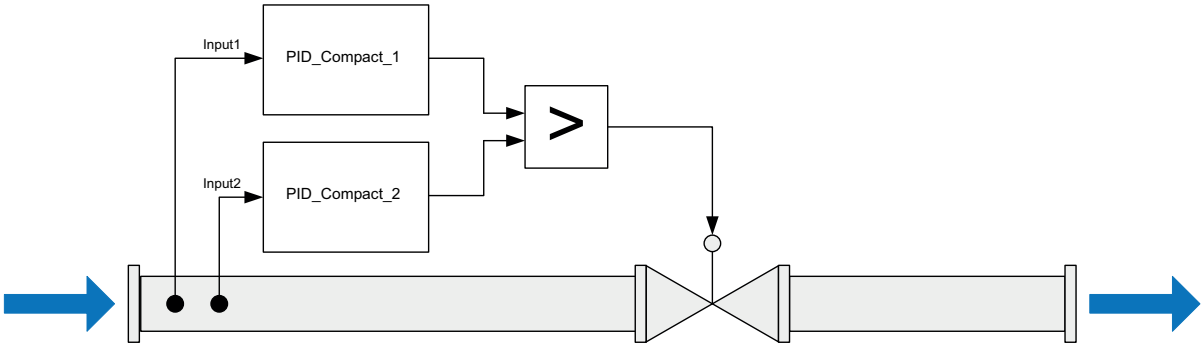

The valve is controlled with the output value of PID_Compact in I/O format (parameter Output_PER) by writing the program tag ActuatorInput.

The setpoint for the flow rate is specified at the parameter PID_Compact_1.Setpoint.

The pressure high limit is specified as setpoint at the parameter PID_Compact_2.Setpoint.

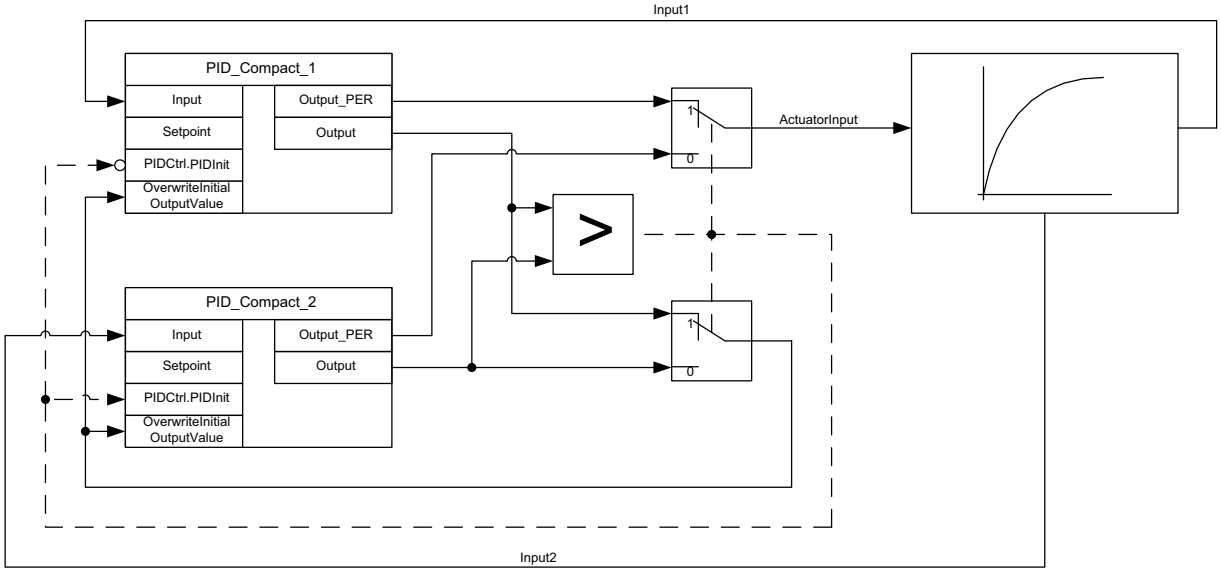

Both controllers must share one valve as shared actuator. The logic that decides which controller gets access to the actuator is implemented by a maximum selection of the output

value (in Real format, parameter Output) in this case. Because the output value corresponds to the opening of the valve, the controller that requires the larger valve opening gets the control.

Note

Activate inversion of the control logic

Because a decrease of the actual value (pressure) is to be achieved with the pressure regulator PID_Compact_2 when the output value increases (valve opening), the inversion of the control logic must be activated: PID_Compact_2.Config.InvertControl = TRUE.

In normal operation of the plant, the actual value of the flow rate corresponds to the setpoint. The flow controller PID_Compact_1 has settled on a stationary output value PID_Compact_1.Output. The actual value of the pressure in normal operation is significantly below the high limit that is specified as setpoint for PID_Compact_2. The pressure regulator therefore wants to close the valve even further to increase the pressure, which means it will calculate an output value PID_Compact_2.Output that is smaller than the output value of the flow controller PID_Compact_1.Output. The maximum selection of the switchover logic therefore gives the flow controller PID_Compact_1 continued access to the actuator. In addition, it is ensured that PID_Compact_2 is updated by means of the assignments PID_Compact_2.OverwriteInitialOutputValue = PID_Compact_1.Output and PID_Compact_2.PIDCtrl.PIDInit = TRUE.

If the pressure now approaches the high limit or exceeds it, for example due to a fault, the pressure regulator PID_Compact_2 calculates a higher output value to open the valve even further and thus reduce the pressure. If PID_Compact_2.Output is greater than PID_Compact_1.Output, the pressure regulator PID_Compact_2 receives access to the actuator through the maximum selection and opens it. It is ensured that PID_Compact_1 is updated by means of the assignments PID_Compact_1.OverwriteInitialOutputValue = PID_Compact_2.Output and PID_Compact_1.PIDCtrl.PIDInit = TRUE.

The pressure is reduced while the flow rate increases and can no longer be kept at the setpoint.

Once the fault has been remedied, the pressure will continue to drop and the opening of the valve is reduced by the pressure regulator. If the flow controller calculates a larger opening as output value, the plant returns to normal operation so that the flow controller PID_Compact_1 once again has access to the actuator.

This example can be implemented with the following SCL program code:

```
"PID_Compact_1"(Input := "Input1");
"PID_Compact_2"(Input := "Input2");
IF "PID_Compact_1".Output >= "PID_Compact_2".Output THEN
  "ActuatorInput" := "PID_Compact_1".Output_PER;
  "PID_Compact_1".PIDCtrl.PIDInit := FALSE;
  "PID_Compact_2".PIDCtrl.PIDInit := TRUE;
  "PID_Compact_2".OverwriteInitialOutputValue :=
  "PID_Compact_1".Output;
ELSE
  "ActuatorInput" := "PID_Compact_2".Output_PER;
  "PID_Compact_1".PIDCtrl.PIDInit := TRUE;
  "PID_Compact_2".PIDCtrl.PIDInit := FALSE;
```

```
  "PID_Compact_1".OverwriteInitialOutputValue :=
  "PID_Compact_2".Output;
END_IF;
```

Simulating PID_Compact V2 with PLCSIM (S7-1200, S7-1500)

Note

Simulation with PLCSIM

The simulation of PID_Compact V2.x with PLCSIM for CPU S7-1200 is not supported.

PID_Compact V2.x can only be simulated for CPU S7-1500 with PLCSIM.

For the simulation with PLCSIM, the time behavior of the simulated PLC is not exactly identical to that of a "real" PLC. The actual cycle clock of a cyclic interrupt OB can have larger fluctuations with a simulated PLC than with "real" PLCs.

In the standard configuration, PID_Compact determines the time between calls automatically and monitors them for fluctuations.

For the simulation of PID_Compact with PLCSIM, for example, a sampling time error (ErrorBits = DW#16#00000800) can therefore be detected.

This results in ongoing tuning being aborted.

The response in automatic mode depends on the value of the ActivateRecoverMode tag.

To prevent this from happening, you should configure PID_Compact for simulation with PLCSIM as follows:

- CycleTime.EnEstimation = FALSE
- CycleTime.EnMonitoring = FALSE
- CycleTime.Value: Assign the cycle clock of the calling cyclic interrupt OB in seconds to this tag.

3.3.1.3 PID_Compact V1 (S7-1200, S7-1500)

Configuring PID_Compact V1 (S7-1200, S7-1500)

Basic settings V1 (S7-1200, S7-1500)

Introduction V1 (S7-1200, S7-1500)

Configure the following properties of the "PID_Compact" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset

- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)

Setpoint, process value and output value

You can only configure the setpoint, process value and output value in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Control mode V1 (S7-1200, S7-1500)

Physical quantity

Select the unit of measurement and physical quantity for the setpoint and process value in the "Controller type" group. The setpoint and process value will be displayed in this unit.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_Compact does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Start-up behavior after reset

To change straight to the last active mode after restarting the CPU, select the "Enable last mode after CPU restart" check box.

PID_Compact will remain in "Inactive" mode if the check box is cleared.

Setpoint V1 (S7-1200, S7-1500)

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value V1 (S7-1200, S7-1500)

PID_Compact will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Output value V1 (S7-1200, S7-1500)

PID_Compact offers three output values. Your actuator will determine which output value you use.

- Output_PER
  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output
  The output value needs to be processed by the user program, for example because of nonlinear actuator response.
- Output_PWM
  The actuator is controlled via a digital output. Pulse width modulation creates minimum ON and minimum OFF times.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output_PER (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to process the output value using the user program:

1. Select the entry "Output" in the drop-down list "Output".
2. Select "Instance DB".
   The calculated output value is saved in the instance data block.
3. For the preparation of the output value, use the output parameter Output.
4. Transfer the processed output value to the actuator via a digital or analog CPU output.

Proceed as follows to use the digital output value:

1. Select the entry "Output_PWM" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the digital output.

Process value settings V1 (S7-1200, S7-1500)

Configure the scaling of your process value and specify the process value absolute limits In the "Process value settings" configuration window.

Scaling the process value

If you have configured the use of Input_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration will be displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID_Compact in the programming editor.
2. Connect Input_PER with an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

Monitoring process value

Specify the absolute high and low limit of the process value. As soon as these limits are violated during operation, the controller switches off and the output value is set to 0%. You must enter reasonable limits for your controlled system. Reasonable limits are important during optimization to obtain optimal PID parameters.

The default for the "High limit process value" is 120 %. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "High limit process value". Only a wire-break and a short-circuit are recognized and the PID_Compact switches to "Inactive" mode.

WARNING
If you set very high process value limits (for example $-3.4*10^{38}$...$+3.4*10^{38}$), process value monitoring will be disabled. Your system may then be damaged if an error occurs.

See also

Process value monitoring V1 (Page 685)

PWM limits V1 (Page 685)

Output value limits V1 (Page 687)

PID parameters V1 (Page 688)

Advanced settings V1 (S7-1200, S7-1500)

Process value monitoring V1 (S7-1200, S7-1500)

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_Compact instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_Compact will respond as follows:

Process value	InputWarning_H	InputWarning_L	Operating mode
> 98° C	TRUE	FALSE	Inactive
≤ 98° C and > 90° C	TRUE	FALSE	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	Automatic mode
< 0° C	FALSE	TRUE	Inactive

See also

Process value settings V1 (Page 683)

PWM limits V1 (Page 685)

Output value limits V1 (Page 687)

PID parameters V1 (Page 688)

PWM limits V1 (S7-1200, S7-1500)

The value at the output parameter Output is transformed into a pulse sequence that is output at output parameter Output_PWM by means of a pulse width modulation. Output is calculated in the PID algorithm sampling time, Output_PWM is output in the PID_Compact sampling time.

The PID algorithm sampling time is determined during pretuning or fine tuning. If manually setting the PID parameters, you will also need to configure the PID algorithm sampling time. The PID_Compact sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the value at Output and is always an integer multiple of the PID_Compact sampling time.

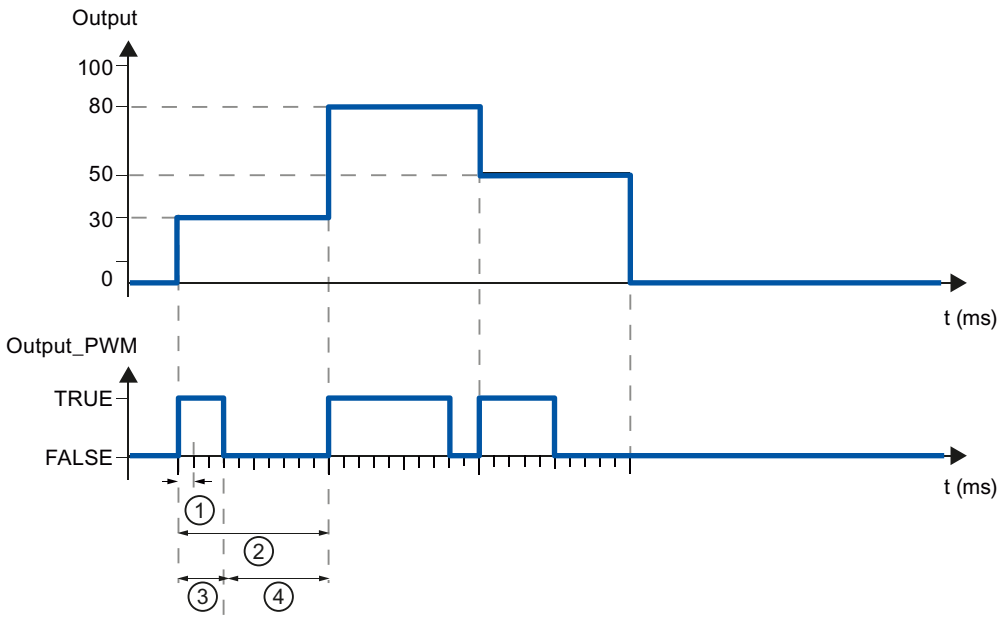

① PID_Compact sampling time
② PID algorithm sampling time
③ Pulse duration
④ Break time

The "Minimum ON time" and the "Minimum OFF time" are rounded to an integer multiple of the PID_Compact sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

Example

PID_Compact sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

Output is a constant 15%. The smallest pulse that PID_Compact can output is 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.

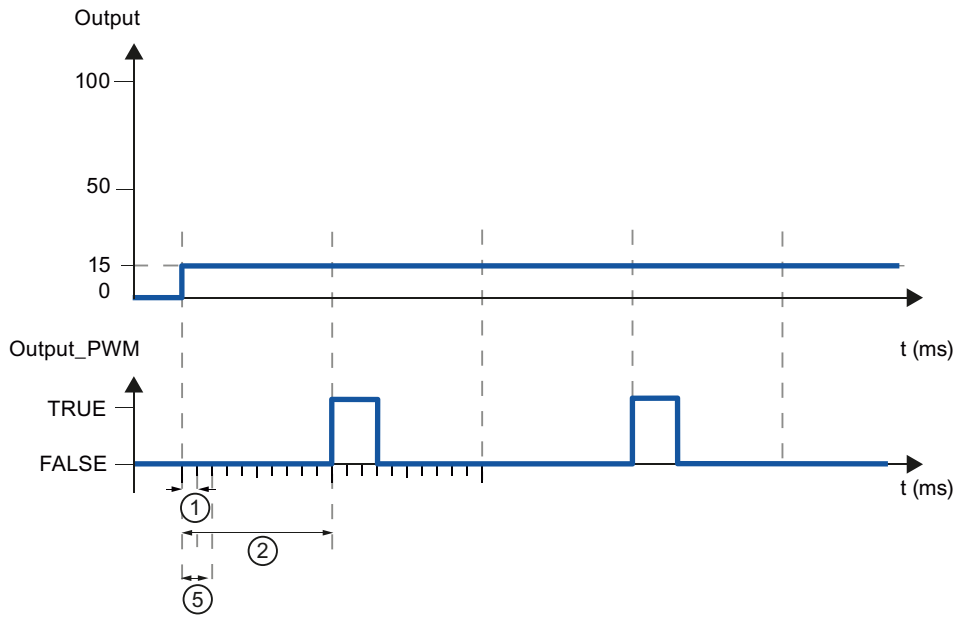

① PID_Compact sampling time
② PID algorithm sampling time
⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you are using "Output" or "Output_PER", you must configure the value 0.0 for the minimum ON and OFF times.

Note

The minimum ON and OFF times only affect the output parameter Output_PWM and are not used for any pulse generators integrated in the CPU.

See also

Process value settings V1 (Page 683)

Process value monitoring V1 (Page 685)

Output value limits V1 (Page 687)

PID parameters V1 (Page 688)

Output value limits V1 (S7-1200, S7-1500)

In the "Output value limits" configuration window, configure the absolute limits of your output value in percent. Absolute output value limits are not violated in neither manual mode nor in automatic mode. If a output value outside the limits is specified in manual mode, the effective value is limited in the CPU to the configured limits.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0
Output_PER	-100.0 to 100.0
Output_PWM	0.0 to 100.0

PID_Compact sets the output value to 0.0 if an error occurs. 0.0 must therefore always be within the output value limits. You will need to add an offset to Output and Output_PER in the user program if you want an output value low limit of greater than 0.0.

See also

Process value settings V1 (Page 683)

Process value monitoring V1 (Page 685)

PWM limits V1 (Page 685)

PID parameters V1 (Page 688)

PID parameters V1 (S7-1200, S7-1500)

The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

Note

The currently active PID parameters are located for PID_Compact V1 in the sRet structure and for PID_Compact V2 in the Retain.CtrlParams structure.

Change the currently active PID parameters only in "Inactive" mode online to prevent malfunction of the PID controller.

If you want to change the PID parameters in "Automatic mode" or "Manual mode" online, change the PID parameters as follows:

- PID_Compact V1: Change the PID parameters in the sBackUp structure and apply these changes with sPid_Cmpt.b_LoadBackUp = TRUE to the sRet structure.
- PID_Compact V2: Change the PID parameters in the CtrlParamsBackUp structure and apply these changes with LoadBackUp = TRUE to the Retain.CtrlParams structure.

Online changes to the PID parameters in "Automatic mode" can result in jumps at the output value.

The PID algorithm operates according to the following equation:

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

y	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:

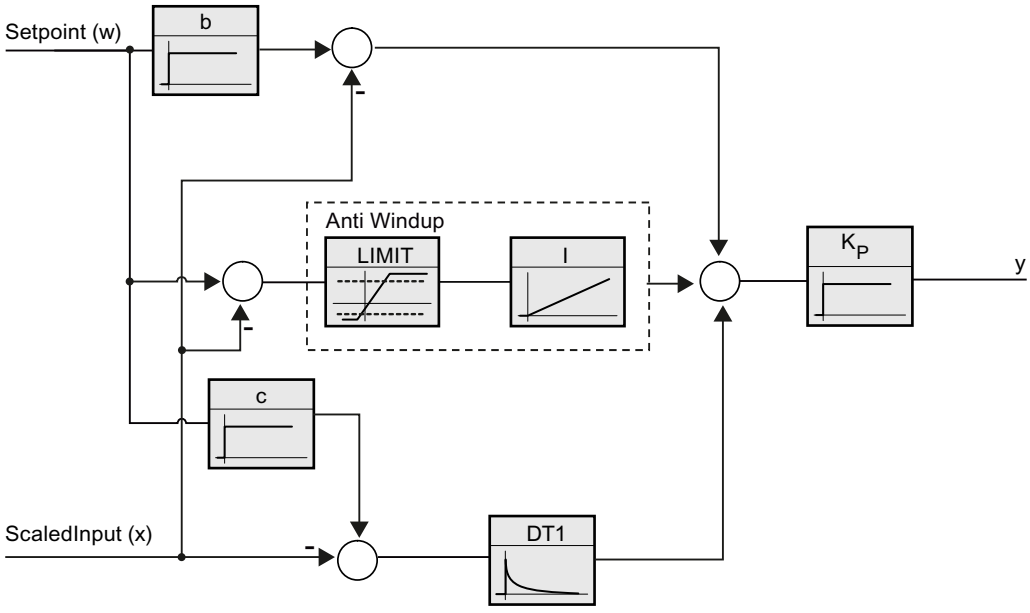

All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_Compact.

Downloading technology objects to device (Page 636)

Proportional gain

The value specifies the proportional gain of the controller. PID_Compact does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0. When the integral action time is changed from a different value to 0.0 online in "Automatic mode", the previous integral action is deleted and the output value jumps.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

If you use Output_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The PID algorithm sampling time corresponds to the time period of the pulse width modulation. The cycle time should be at least 10 times the PID algorithm sampling time.

Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list.

- **PID**
  Calculates PID parameters during pretuning and fine tuning.
- **PI**
  Calculates PI parameters during pretuning and fine tuning.
- **User-defined**
  The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program.

See also

Selection of the controller structure for specified controlled systems (Page 629)

Commissioning PID_Compact V1 (S7-1200, S7-1500)

Commissioning V1 (S7-1200, S7-1500)

The commissioning window helps you commission the PID controller. You can monitor the values for the setpoint, process value and output value along the time axis in the trend view. The following functions are supported in the commissioning window:

- Controller pretuning
- Controller fine tuning
  Use fine tuning for fine adjustments to the PID parameters.
- Monitoring the current closed-loop control in the trend view
- Testing the controlled system by specifying a manual output value

All functions require an online connection to the CPU to have been established.

Basic handling

- Select the desired sampling time in the "Sampling time" drop-down list.
  All values in the commissioning window are updated in the selected update time.
- Click the "Start" icon in the measuring group if you want to use the commissioning functions.
  Value recording is started. The current values for the setpoint, process value and output value are entered in the trend view. Operation of the commissioning window is enabled.
- Click the "Stop" icon if you want to end the commissioning functions.
  The values recorded in the trend view can continue to be analyzed.

Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

See also

Pretuning V1 (Page 692)

Fine tuning V1 (Page 693)

"Manual" mode V1 (Page 695)

Pretuning V1 (S7-1200, S7-1500)

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The PID parameters are backed up before being recalculated.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- PID_Compact is in "inactive" or "manual" mode.
- The setpoint may not be changed during controller tuning. PID_Compact will otherwise be deactivated.
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is > 50% of the setpoint.

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_Compact > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list.
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_Compact will change to "Inactive" mode.

See also

Commissioning V1 (Page 691)

Fine tuning V1 (Page 693)

"Manual" mode V1 (Page 695)

Fine tuning V1 (S7-1200, S7-1500)

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.

PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

Requirement

- The PID_Compact instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- The setpoint may not be changed during controller tuning.
- PID_Compact is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning can be started in "inactive", "automatic" or "manual" mode. Fine tuning proceeds as follows when started in:

- Automatic mode
  Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.
  PID_Comact will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  If the requirements for pretuning are met, pretuning is started. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode.
  An attempt is made to reach the setpoint with a minimum or maximum output value if the process value for pretuning is already too near the setpoint. This can produce increased overshoot.

Procedure

Proceed as follows to carry out "fine tuning":

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

The PID parameters will have been optimized if fine tuning has been executed without errors. PID_Compact changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during "fine tuning", PID_Compact will change to "inactive" mode.

See also

Commissioning V1 (Page 691)

Pretuning V1 (Page 692)

"Manual" mode V1 (Page 695)

"Manual" mode V1 (S7-1200, S7-1500)

The following section describes how you can use the "Manual" operating mode in the commissioning window of the "PID Compact" technology object.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established and the CPU is in the "RUN" mode.
- The functions of the commissioning window have been enabled via the "Start" icon.

Procedure

Use "Manual mode" in the commissioning window if you want to test the process by specifying a manual value. To define a manual value, proceed as follows:

1. Select the check box "Manual mode" in the "Online status of the controller" area. PID_Compact operates in manual mode. The most recent current output value remains in effect.
2. Enter the manual value in the "Output" field as a % value.
3. Click the control icon .

Result

The manual value is written to the CPU and immediately goes into effect.

Note

PID_Compact continues to monitor the process value. If the process value limits are exceeded, PID_Compact is deactivated.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller. The change to automatic mode is bumpless.

See also

Commissioning V1 (Page 691)

Pretuning V1 (Page 692)

Fine tuning V1 (Page 693)

Simulating PID_Compact V1 with PLCSIM (S7-1200, S7-1500)

Note

Simulation with PLCSIM

For the simulation with PLCSIM, the time behavior of the simulated PLC is not exactly identical to that of a "real" PLC. The actual cycle clock of a cyclic interrupt OB can have larger fluctuations with a simulated PLC than with "real" PLCs.

In the standard configuration, PID_Compact determines the time between calls automatically and monitors them for fluctuations.

For a simulation of PID_Compact with PLCSIM, for example, a sampling time error (ErrorBits = DW#16#00000800) can therefore be detected.

PID_Compact switches to "Inactive" mode (State = 0) in this case.

To prevent this from happening, you should configure PID_Compact for simulation with PLCSIM as follows:

- sb_EnCyclEstimation = FALSE
- sb_EnCyclMonitoring = FALSE
- sPid_Calc.r_Cycle: Assign the cycle clock of the calling cyclic interrupt OB in seconds to this tag.

3.3.2 Using PID_3Step (S7-1200, S7-1500)

3.3.2.1 Technology object PID_3Step (S7-1200, S7-1500)

The technology object PID_3Step provides a PID controller with tuning for valves or actuators with integral response.

You can configure the following controllers:

- Three-point step controller with position feedback
- Three-point step controller without position feedback
- Valve controller with analog output value

PID_3Step continuously acquires the measured process value within a control loop and compares it with the setpoint. From the resulting control deviation, PID_3Step calculates an

output value through which the process value reaches the setpoint as quickly and steadily as possible. The output value for the PID controller consists of three actions:

- **Proportional** action
  The proportional action of the output value increases in proportion to the control deviation.
- **I** action
  The integral action of the output value increases until the control deviation has been balanced.
- **D** action
  The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID_3Step calculates the proportional, integral and derivative parameters for your controlled system during pretuning. Fine tuning can be used to tune the parameters further. You do not need to manually determine the parameters.

Additional information

- Overview of software controller (Page 631)
- Add technology objects (Page 633)
- Configure technology objects (Page 634)
- Configuring PID_3Step V2 (Page 698)
- Configuring PID_3Step V1 (Page 717)

Principle

For more information, see the following FAQs in the Siemens Industry Online Support:

- Entry ID 68011827 (https://support.industry.siemens.com/cs/ww/en/view/68011827)

3.3.2.2 PID_3Step V2 (S7-1200, S7-1500)

Configuring PID_3Step V2 (S7-1200, S7-1500)

Basic settings V2 (S7-1200, S7-1500)

Introduction V2 (S7-1200, S7-1500)

Configure the following properties of the "PID_3Step" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset

- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)
- Position feedback (only in the Inspector window)

Setpoint, process value, output value and position feedback

You can only configure the setpoint, process value, output value and position feedback in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Control mode V2 (S7-1200, S7-1500)

Physical quantity

Select the physical quantity and unit of measurement for setpoint, process value, and disturbance variable in the "Controller type" group. Setpoint, process value, and disturbance variable is displayed in this unit of measurement.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_3Step does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Startup characteristics

1. To switch to "Inactive" mode after CPU restart, clear the "Activate Mode after CPU restart" check box.
   To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.
2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.
   After a complete download to the device, PID_3Step starts in the selected operating mode. With each additional restart, PID_3Step starts in the mode that was last saved in Mode.

Example

You have selected the "Activate Mode after CPU restart" check box and the entry "Pretuning" in the "Set Mode to" list. After a complete download to the device, PID_3Step starts in the "Pretuning" mode. If pretuning is still active, PID_3Step starts in "Pretuning" mode again after restart of the CPU. If pretuning was successfully completed and automatic mode is active, PID_3Step starts in "Automatic mode" after restart of the CPU.

Setpoint V2 (S7-1200, S7-1500)

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value V2 (S7-1200, S7-1500)

PID_3Step will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Position feedback V2 (S7-1200, S7-1500)

Position feedback configuration depends upon the actuator used.

- Actuator without position feedback
- Actuator with digital endstop signals
- Actuator with analog position feedback
- Actuator with analog position feedback and endstop signals

Actuator without position feedback

Proceed as follows to configure PID_3Step for an actuator without position feedback:

1. Select the entry "No Feedback" in the drop-down list "Feedback".

Actuator with digital endstop signals

Proceed as follows to configure PID_3Step for an actuator with endstop signals:

1. Select the entry "No Feedback" in the drop-down list "Feedback".
2. Activate the "Actuator endstop signals" check box.
3. Select "Instruction" as source for Actuator_H and Actuator_L.
4. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Actuator with analog position feedback

Proceed as follows to configure PID_3Step for an actuator with analog position feedback:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
   - Use the analog input value for Feedback_PER. Configure Feedback_PER scaling in the actuator settings.
   - Process the analog input value for Feedback using your user program.
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.

Actuator with analog position feedback and endstop signals

Proceed as follows to configure PID_3Step for an actuator with analog position feedback and endstop signals:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.
4. Activate the "Actuator endstop signals" check box.
5. Select "Instruction" as source for Actuator_H and Actuator_L.
6. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Output value V2 (S7-1200, S7-1500)

PID_3Step offers an analog output value (Output_PER) and digital output values (Output_UP, Output_DN). Your actuator will determine which output value you use.

- Output_PER
  The actuator has a relevant motor transition time and is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V or 4...20 mA. The value at Output_PER corresponds to the target position of the valve, e.g. Output_PER = 13824, when the valve is to be opened by 50%.
  For auto-tuning and anti windup behavior, for example, PID_3Step takes into consideration that the analog output value has a delayed effect on the process due to the motor transition time. If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID_Compact instead.
- Output_UP, Output_DN
  The actuator has a relevant motor transition time and is controlled by two digital outputs.
  Output_UP moves the valve in the open state direction.
  Output_DN moves the valve in the closed state direction.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior. You should therefore configure the motor transition time under "Actuator settings" with the value that the motor requires to move the actuator from the closed to the opened state.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the digital output value:

1. Select the entry "Output (digital)" in the drop-down list "Output".
2. Select "Instruction" for Output_UP and Output_DN.
3. Enter the addresses of the digital outputs.

Proceed as follows to process the output value using the user program:

1. Select the entry corresponding to the actuator in the drop-down list "Output".
2. Select "Instruction".
3. Enter the name of the tag you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.

Process value settings V2 (S7-1200, S7-1500)

Process value scaling V2 (S7-1200, S7-1500)

If you have configured the use of Input_PER in the basic setting, you must convert the value of the analog input to the physical quantity of the process value. The current configuration is displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" text boxs.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are stored in the hardware configuration. To use the value pairs from the hardware configuration, follow these steps:

1. Select the PID_3Step instruction in the programming editor.
2. Interconnect Input_PER with an analog input in the basic settings.
3. Click the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

Process value limits V2 (S7-1200, S7-1500)

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can specify how PID_3Step responds to errors in automatic mode in the actuator settings.

Final controlling element settings V2 (S7-1200, S7-1500)

Final controlling element V2 (S7-1200, S7-1500)

Actuator-specific times

Configure the motor transition time and the minimum ON and OFF times to prevent damage to the actuator. You can find the specifications in the actuator data sheet.

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. You can measure the motor transition time during commissioning.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior.

If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID_Compact instead.

The motor transition time is retentive. If you enter the motor transition time manually, you must completely download PID_3Step.

Downloading technology objects to device (Page 636)

If you are using "Output_UP" or "Output_DN", you can reduce the switching frequency with the minimum on and minimum OFF time.

The on or off times calculated are totaled in automatic mode and only become effective when the sum is greater than or equal to the minimum on or OFF time.

Manual_UP = TRUE or Manual_DN = TRUE in manual mode operates the actuator for at least the minimum ON or OFF time.

If you have selected the analog output value Output_PER, the minimum ON time and the minimum OFF time are not evaluated and cannot be changed.

Reaction to error

PID_3Step is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

NOTICE
Your system may be damaged. If you output "Current value while error pending" or "Substitute output value while error pending" in the event of an error, PID_3Step remains in automatic mode even if the process value limits are violated. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID_3Step generates a programmable output value in the case of an error:

- Current value
  PID_3Step is switched off and no longer modifies the actuator position.
- Current value for error while error is pending
  The controller functions of PID_3Step are switched off and the position of the actuator is no longer changed.
  If the following errors occur in automatic mode, PID_3Step returns to automatic mode as soon as the errors are no longer pending.
  - 0002h: Invalid value at Input_PER parameter.
  - 0200h: Invalid value at Input parameter.
  - 0400h: Calculation of output value failed.
  - 1000h: Invalid value at Setpoint parameter.
  - 2000h: Invalid value at Feedback_PER parameter.
  - 4000h: Invalid value at Feedback parameter.
  - 8000h: Error during digital position feedback.
  - 20000h: Invalid value at SavePosition tag.

  If one or more of the following errors occur, PID_3Step stays in automatic mode:
  - 0001h: The Input parameter is outside the process value limits.
  - 0800h: Sampling time error
  - 40000h: Invalid value at Disturbance parameter.

  PID_3Step remains in manual mode if an error occurs in manual mode.
  If an error occurs during tuning or transition time measurement, PID_3Step switches to the mode in which tuning or transition time measurement was started. Only in the event of the following error is tuning not aborted:
  - 0020h: Pretuning is not permitted during fine tuning.

- Substitute output value
  PID_3Step moves the actuator to the substitute output value and then switches off.
- Substitute output value while error is pending
  PID_3Step moves the actuator to the substitute output value. When the substitute output value is reached, PID_3Step reacts as it does with "Current value for while error is pending".

Enter the substitute output value in "%".

Only substitute output values 0% and 100% can be approached precisely in the case of actuators without analog position feedback. A substitute output value not equal to 0% or 100% is approached via an internally simulated position feedback. This procedure does not, however, allow the exact approach of substitute output value.

All substitute output values can be approached precisely with actuators with analog position feedback.

Scaling V2 position feedback (S7-1200, S7-1500)

Scaling position feedback

If you have configured the use of Feedback_PER in the basic settings, you will need to convert the value of the analog input into %. The current configuration will be displayed in the "Feedback" display.

Feedback_PER is scaled using a low and high value pair.

1. Enter the low pair of values in the "Low endstop" and "Low" input boxes.
2. Enter the high pair of values in the "High endstop" and "High" input boxes.

"Low endstop" must be less than "High endstop"; "Low" must be less than "High".

The valid values for "High endstop" and "Low endstop" depend upon:

- No Feedback, Feedback, Feedback_PER
- Output (analog), Output (digital)

Output	Feedback	Low endstop	High endstop
Output (digital)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (digital)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (digital)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (analog)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%

Output value limits V2 (S7-1200, S7-1500)

Limiting the output value

You can exceed or undershoot the output value limits during the transition time measurement and with mode = 10. The output value is limited to these values in all other modes.

Enter the absolute output value limits in the "Output value high limit" and "Output value low limit" input boxes. The output value limits must be within "Low endstop" and "High endstop".

If no Feedback is available and Output (digital) is set, you cannot limit the output value. Output_UP and Output_DN are then reset upon Actuator_H = TRUE or Actuator_L = TRUE. If no endstop signals are available, Output_UP and Output_DN are reset after a travel time of 150% of the motor actuating time.

The default value of 150% can be adjusted using the tagConfig.VirtualActuatorLimit. As of PID_3Step Version 2.3 the monitoring and limiting of the travel time can be deactivated with Config.VirtualActuatorLimit = 0.0.

Advanced settings V2 (S7-1200, S7-1500)

Actual value monitoring V2 (S7-1200, S7-1500)

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_3Step instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_3Step will respond as follows:

Process value	InputWarning_H	InputWarning_L	ErrorBits	Operating mode
> 98° C	TRUE	FALSE	0001h	As configured
≤ 98° C and > 90° C	TRUE	FALSE	0000h	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	0000h	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	0000h	Automatic mode
< 0° C	FALSE	TRUE	0001h	As configured

In the actuator settings, you can configure the response of PID_3Step when the process value high limit or low limit is violated.

PID parameters V2 (S7-1200, S7-1500)

The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

Note

The currently active PID parameters are located in the Retain.CtrlParams structure.

Change the currently active PID parameters only in "Inactive" mode online to prevent malfunction of the PID controller.

If you want to change the PID parameters in "Automatic mode" or "Manual mode" online, change the PID parameters in the CtrlParamsBackUp structure and apply these changes to the Retain.CtrlParams structure as follows:

- PID_3Step V1: Apply the changes with Config.LoadBackUp = TRUE
- PID_3Step V2: Apply the changes with LoadBackUp = TRUE

Online changes to the PID parameters in "Automatic mode" can result in jumps at the output value.

The PID algorithm operates according to the following equation:

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right]$$

Δy	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integration time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:

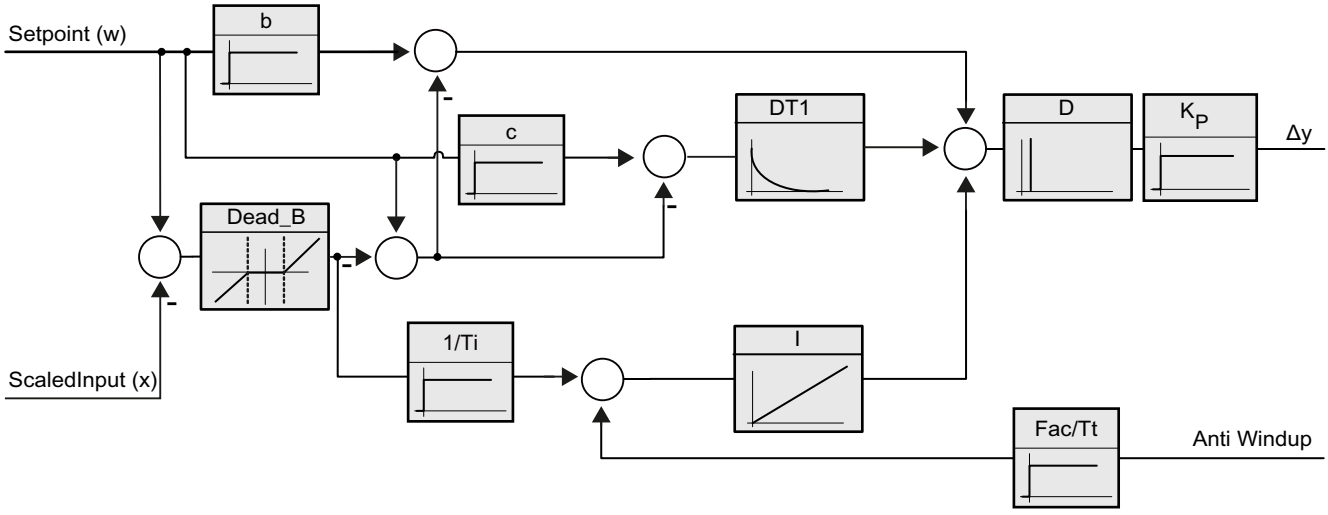

All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_3Step.

Downloading technology objects to device (Page 636)

Proportional gain

The value specifies the proportional gain of the controller. PID_3Step does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integration time

The integration time determines the time behavior of the integral action. The integral action is deactivated with integration time = 0.0. When the integral action time is changed from a different value to 0.0 online in "Automatic mode", the previous integral action is deleted and the output value jumps.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the PID_3Step sampling time. All other functions of PID_3Step are executed at every call.

Dead band width

The deadband suppresses the noise component in the steady controller state. The dead band width specifies the size of the dead band. The dead band is off if the dead band width is 0.0.

If values not equal to 1.0 are configured for the proportional action weighting or the derivative action weighting, setpoint changes even within the dead zone affect the output value. Process value changes within the dead zone do not affect the output value, regardless of the weighting.

See also

Selection of the controller structure for specified controlled systems (Page 629)

Commissioning PID_3Step V2 (S7-1200, S7-1500)

Pretuning V2 (S7-1200, S7-1500)

The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The PID parameters are backed up before being recalculated.

The setpoint is frozen during pretuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.
- PID_3Step is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_3Step > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list in the working area "Tuning".
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_3Step responds with the configured reaction to errors.

Fine tuning V2 (S7-1200, S7-1500)

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

The setpoint is frozen during fine tuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID_3Step is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning proceeds as follows when started from:

- Automatic mode
  Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.
  PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  Pretuning is always started first. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

Procedure

To perform fine tuning, follow these steps:

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If no errors occurred during fine tuning, the PID parameters have been tuned. PID_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID_3Step responds with the configured response to errors.

Commissioning with manual PID parameters V2 (S7-1200, S7-1500)

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.
- PID_3Step is in "inactive" mode.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

Procedure

Proceed as follows to commission PID_3Step with manual PID parameters:

1. Double-click on "PID_3Step > Configuration" in the project tree.
2. Click on "Advanced settings > PID Parameters" in the configuration window.
3. Select the check box "Enable direct input".
4. Enter the PID parameters.
5. Double-click the "PID_3Step > Commissioning" entry in the project tree.
6. Establish an online connection to the CPU.
7. Load the PID parameters to the CPU.
8. Click the "Start PID_3Step" icon.

Result

PID_3Step changes to automatic mode and controls using the current PID parameters.

See also

PID parameters V2 (Page 708)

Measuring the motor transition time V2 (S7-1200, S7-1500)

Introduction

PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

Requirement

- Feedback or Feedback_PER has been selected in the basic settings and the signal has been connected.
- An online connection to the CPU has been established.

1. Select the "Use position feedback" check box.
2. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. The difference between "Target position" and "Position feedback" must be at least 50% of the valid output value range.
3. Click the "Start" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. When the transition time measurement is ended and ActivateRecoverMode = TRUE, PID_3Step switches to the operating mode from which the transition time measurement was started. If the transition time measurement is ended and ActivateRecoverMode = FALSE, PID_3Step changes to "Inactive" mode.

Note

Click on the icon "Upload measured transition time" to load the motor transition time measured to the project.

Actuators with endstop signals

Proceed as follows to measure the transition time of actuators with endstop signals:

Requirement

- The "Endstop signals" check box in the basic settings has been selected and Actuator_H and Actuator_L are connected.
- An online connection to the CPU has been established.

Proceed as follows to measure motor transition time with endstop signals:

1. Select the "Use actuator endstop signals" check box.
2. Select the direction in which the actuator is to be moved.
   - Open - Close - Open
     The actuator is moved first to the high endstop, then to the low endstop and then back to the high endstop.
   - Close - Open - Close
     The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop.
3. Click the "Start" icon.

Result

The actuator is moved in the selected direction. Time measurement will start once the actuator has reached the first endstop and will end when the actuator reaches this endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. When the transition time measurement is ended and ActivateRecoverMode = TRUE, PID_3Step switches to the operating mode from which the transition time measurement was started. If the transition time measurement is ended and ActivateRecoverMode = FALSE, PID_3Step changes to "Inactive" mode.

Cancelling transition time measurement

PID_3Step switches to "Inactive" mode if you cancel transition time measurement by pressing the Stop button.

Simulating PID_3Step V2 with PLCSIM (S7-1200, S7-1500)

Note

Simulation with PLCSIM

The simulation of PID_3Step V2.x with PLCSIM for CPU S7-1200 is not supported.

PID_3Step V2.x can only be simulated for CPU S7-1500 with PLCSIM.

For the simulation with PLCSIM, the time behavior of the simulated PLC is not exactly identical to that of a "real" PLC. The actual cycle clock of a cyclic interrupt OB can have larger fluctuations with a simulated PLC than with "real" PLCs.

In the standard configuration, PID_3Step determines the time between calls automatically and monitors them for fluctuations.

For a simulation of PID_3Step with PLCSIM, for example, a sampling time error ((ErrorBits = DW#16#00000800) can therefore be detected.

This results in ongoing tuning being aborted.

The response in automatic mode depends on the value of the ActivateRecoverMode tag.

To prevent this from happening, you should configure PID_3Step for simulation with PLCSIM as follows:

- CycleTime.EnEstimation = FALSE
- CycleTime.EnMonitoring = FALSE
- CycleTime.Value: Assign the cycle clock of the calling cyclic interrupt OB in seconds to this tag.

3.3.2.3 PID_3Step V1 (S7-1200, S7-1500)

Configuring PID_3Step V1 (S7-1200, S7-1500)

Basic settings V1 (S7-1200, S7-1500)

Introduction V1 (S7-1200, S7-1500)

Configure the following properties of the "PID_3Step" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)
- Position feedback (only in the Inspector window)

Setpoint, process value, output value and position feedback

You can only configure the setpoint, process value, output value and position feedback in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Control mode V1 (S7-1200, S7-1500)

Physical quantity

Select the unit of measurement and physical quantity for the setpoint and process value in the "Controller type" group. The setpoint and process value will be displayed in this unit.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_3Step does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Start-up behavior after reset

To change straight to the last active mode after restarting the CPU, select the "Enable last mode after CPU restart" check box.

PID_3Step will remain in "Inactive" mode if the check box is cleared.

Setpoint V1 (S7-1200, S7-1500)

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value V1 (S7-1200, S7-1500)

PID_3Step will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Position feedback V1 (S7-1200, S7-1500)

Position feedback configuration depends upon the actuator used.

- Actuator without position feedback
- Actuator with digital endstop signals
- Actuator with analog position feedback
- Actuator with analog position feedback and endstop signals

Actuator without position feedback

Proceed as follows to configure PID_3Step for an actuator without position feedback:

1. Select the entry "No Feedback" in the drop-down list "Feedback".

Actuator with digital endstop signals

Proceed as follows to configure PID_3Step for an actuator with endstop signals:

1. Select the entry "No Feedback" in the drop-down list "Feedback".
2. Activate the "Actuator endstop signals" check box.
3. Select "Instruction" as source for Actuator_H and Actuator_L.
4. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Actuator with analog position feedback

Proceed as follows to configure PID_3Step for an actuator with analog position feedback:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
   - Use the analog input value for Feedback_PER. Configure Feedback_PER scaling in the actuator settings.
   - Process the analog input value for Feedback using your user program.
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.

Actuator with analog position feedback and endstop signals

Proceed as follows to configure PID_3Step for an actuator with analog position feedback and endstop signals:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.
4. Activate the "Actuator endstop signals" check box.
5. Select "Instruction" as source for Actuator_H and Actuator_L.
6. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Output value V1 (S7-1200, S7-1500)

PID_3Step offers an analog output value (Output_PER) and digital output values (Output_UP, Output_DN). Your actuator will determine which output value you use.

- Output_PER
  The actuator has a relevant motor transition time and is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V or 4...20 mA. The value at Output_PER corresponds to the target position of the valve, e.g. Output_PER = 13824, when the valve is to be opened by 50%.
  For auto-tuning and anti windup behavior, for example, PID_3Step takes into consideration that the analog output value has a delayed effect on the process due to the motor transition time. If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID_Compact instead.
- Output_UP, Output_DN
  The actuator has a relevant motor transition time and is controlled by two digital outputs.
  Output_UP moves the valve in the open state direction.
  Output_DN moves the valve in the closed state direction.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior. You should therefore configure the motor transition time under "Actuator settings" with the value that the motor requires to move the actuator from the closed to the opened state.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the digital output value:

1. Select the entry "Output (digital)" in the drop-down list "Output".
2. Select "Instruction" for Output_UP and Output_DN.
3. Enter the addresses of the digital outputs.

Proceed as follows to process the output value using the user program:

1. Select the entry corresponding to the actuator in the drop-down list "Output".
2. Select "Instruction".
3. Enter the name of the tag you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.

Process value settings V1 (S7-1200, S7-1500)

Configure the scaling of your process value and specify the process value absolute limits In the "Process value settings" configuration window.

Scaling the process value

If you have configured the use of Input_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration will be displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID_3Step in the programming editor.
2. Connect Input_PER to an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.
   The existing values will be overwritten with the values from the hardware configuration.

Monitoring process value

Specify the absolute high and low limit of the process value. You must enter reasonable limits for your controlled system. Reasonable limits are important during optimization to obtain optimal PID parameters. The default for the "High limit process value" is 120 %. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). This setting ensures that an error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error.

NOTICE
Your system may be damaged. If you set very high process value limits (for example $-3.4*10^{38}$...$+3.4*10^{38}$), process value monitoring will be disabled. Your system may then be damaged if an error occurs. You need to configure useful process value limits for your controlled system.

V1 final controlling element setting (S7-1200, S7-1500)

Actuator-specific times

Configure the motor transition time and the minimum ON and OFF times to prevent damage to the actuator. You can find the specifications in the actuator data sheet.

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The maximum time that the actuator is moved in one direction is 110% of the motor transition time. You can measure the motor transition time during commissioning.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior.

If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID_Compact instead.

If you are using "Output_UP" or "Output_DN", you can reduce the switching frequency with the minimum on and minimum OFF time.

The on or off times calculated are totaled in automatic mode and only become effective when the sum is greater than or equal to the minimum on or OFF time.

A rising edge at Manual_UP or Manual_DN in manual mode will operate the actuator for at least the minimum on or OFF time.

If you have selected the analog output value Output_PER, the minimum ON time and the minimum OFF time are not evaluated and cannot be changed.

Reaction to error

PID_3Step is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

PID_3Step generates a programmable output value in response to an error:

- Current value
  PID_3Step is switched off and no longer modifies the actuator position.
- Current value for error while error is pending
  The controller functions of PID_3Step are switched off and the position of the actuator is no longer changed.
  If the following errors occur in automatic mode, PID_3Step returns to automatic mode as soon as the errors are no longer pending.
  - 0002h: Invalid value at Input_PER parameter.
  - 0200h: Invalid value at Input parameter.
  - 0800h: Sampling time error
  - 1000h: Invalid value at Setpoint parameter.
  - 2000h: Invalid value at Feedback_PER parameter.
  - 4000h: Invalid value at Feedback parameter.
  - 8000h: Error during digital position feedback.

  If one of these error occurs in manual mode, PID_3Step remains in manual mode.
  If an error occurs during the tuning or transition time measurement, PID_3Step is switched off.
- Substitute output value
  PID_3Step moves the actuator to the substitute output value and then switches off.
- Substitute output value while error is pending
  PID_3Step moves the actuator to the substitute output value. When the substitute output value is reached, PID_3Step reacts as it does with "Current value for while error is pending".

Enter the substitute output value in "%".

Only substitute output values 0% and 100% can be approached precisely in the case of actuators without analog position feedback. The actuator is moved in one direction at 110% of the motor transition time to ensure the high or low endstop is reached. There endstop signals take priority. A substitute output value not equal to 0% or 100% is approached via an internally simulated position feedback. This procedure does not, however, allow the exact approach of substitute output value.

All substitute output values can be approached precisely with actuators with analog position feedback.

Scaling position feedback

If you have configured the use of Feedback_PER in the basic settings, you will need to convert the value of the analog input into %. The current configuration will be displayed in the "Feedback" display.

Feedback_PER is scaled using a low and high value pair.

1. Enter the low pair of values in the "Low endstop" and "Low" input boxes.
2. Enter the high pair of values in the "High endstop" and "High" input boxes.

"Low endstop" must be less than "High endstop"; "Low" must be less than "High".

The valid values for "High endstop" and "Low endstop" depend upon:

- No Feedback, Feedback, Feedback_PER
- Output (analog), Output (digital)

Output	Feedback	Low endstop	High endstop
Output (digital)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (digital)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (digital)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (analog)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%

Limiting the output value

You can only exceed or undershoot the output value limits during the transition time measurement. The output value is limited to these values in all other modes.

Enter the absolute output value limits in the "Output value high limit" and "Output value low limit" input boxes. The output value limits must be within "Low endstop" and "High endstop".

If no Feedback is available and Output (digital) is set, you cannot limit the output value. The digital outputs are reset with Actuator_H = TRUE or Actuator_L = TRUE, or after a travel time amounting to 110% of the motor transition time.

Advanced settings V1 (S7-1200, S7-1500)

Actual value monitoring V1 (S7-1200, S7-1500)

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_3Step instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_3Step will respond as follows:

Process value	InputWarning_H	InputWarning_L	Operating mode
> 98° C	TRUE	FALSE	Inactive
≤ 98° C and > 90° C	TRUE	FALSE	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	Automatic mode
< 0° C	FALSE	TRUE	Inactive

PID parameters V1 (S7-1200, S7-1500)

The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

Note

The currently active PID parameters are located in the Retain.CtrlParams structure.

Change the currently active PID parameters only in "Inactive" mode online to prevent malfunction of the PID controller.

If you want to change the PID parameters in "Automatic mode" or "Manual mode" online, change the PID parameters in the CtrlParamsBackUp structure and apply these changes to the Retain.CtrlParams structure as follows:

- PID_3Step V1: Apply the changes with Config.LoadBackUp = TRUE
- PID_3Step V2: Apply the changes with LoadBackUp = TRUE

Online changes to the PID parameters in "Automatic mode" can result in jumps at the output value.

The PID algorithm operates according to the following equation:

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Δy	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integration time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:

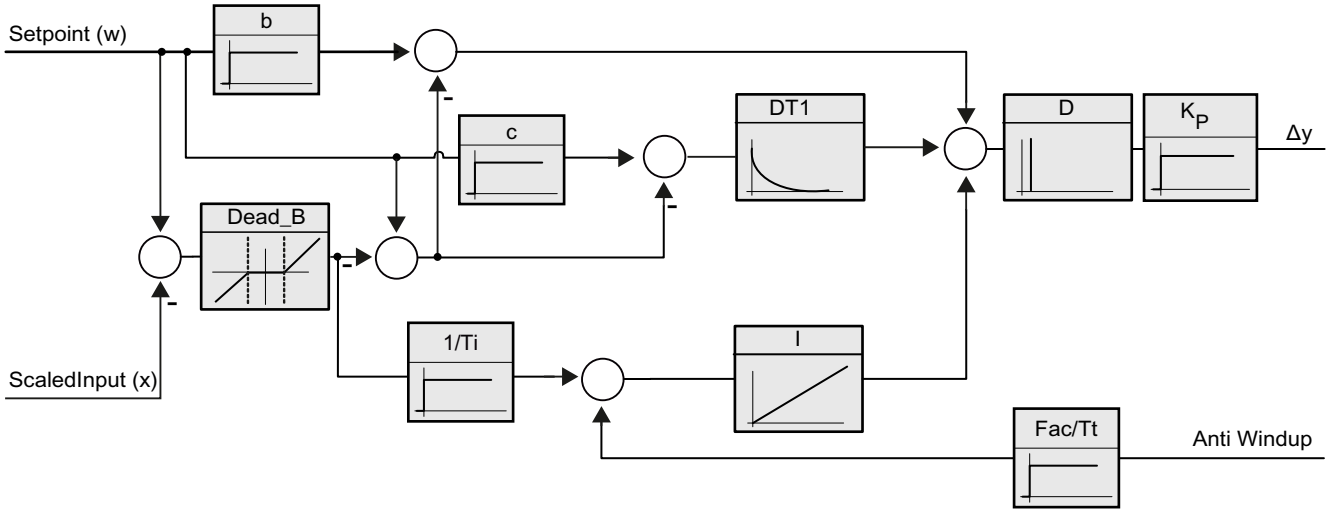

All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_3Step.

Downloading technology objects to device (Page 636)

Proportional gain

The value specifies the proportional gain of the controller. PID_3Step does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integration time

The integration time determines the time behavior of the integral action. The integral action is deactivated with integration time = 0.0. When the integral action time is changed from a different value to 0.0 online in "Automatic mode", the previous integral action is deleted and the output value jumps.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the PID_3Step sampling time. All other functions of PID_3Step are executed at every call.

Dead band width

The deadband suppresses the noise component in the steady controller state. The dead band width specifies the size of the dead band. The dead band is off if the dead band width is 0.0.

If values not equal to 1.0 are configured for the proportional action weighting or the derivative action weighting, setpoint changes even within the dead zone affect the output value. Process value changes within the dead zone do not affect the output value, regardless of the weighting.

See also

Selection of the controller structure for specified controlled systems (Page 629)

Commissioning PID_3Step V1 (S7-1200, S7-1500)

Commissioning V1 (S7-1200, S7-1500)

You can monitor the setpoint, process value and output value over time in the "Tuning" working area. The following commissioning functions are supported in the curve plotter:

- Controller pretuning
- Controller fine tuning
- Monitoring the current closed-loop control in the trend view

All functions require an online connection to the CPU to have been established.

Basic handling

- Select the desired sampling time in the "Sampling time" drop-down list.
  All values in the tuning working area are updated in the selected update time.
- Click the "Start" icon in the measuring group if you want to use the commissioning functions. Value recording is started. The current values for the setpoint, process value and output value are entered in the trend view. Operation of the commissioning window is enabled.
- Click the "Stop" icon if you want to end the commissioning functions.
  The values recorded in the trend view can continue to be analyzed.
- Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

Pretuning V1 (S7-1200, S7-1500)

The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The PID parameters are backed up before being recalculated.

The setpoint is frozen during pretuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- PID_3Step is in "inactive" or "manual" mode.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_3Step > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list in the working area "Tuning".
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_3Step changes to "Inactive" mode.

Fine tuning V1 (S7-1200, S7-1500)

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.

PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

The setpoint is frozen during fine tuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- The motor transition time has been configured or measured.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID_3Step is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning proceeds as follows when started in:

- Automatic mode
  Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.
  PID_3Step will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  Pretuning is always started first. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

Procedure

Proceed as follows to carry out "fine tuning":

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

The PID parameters will have been optimized if fine tuning has been executed without errors. PID_3Step changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID_3Step will change to "inactive" mode.

Commissioning with manual PID parameters V1 (S7-1200, S7-1500)

Procedure

Proceed as follows to commission PID_3Step with manual PID parameters:

1. Double-click on "PID_3Step > Configuration" in the project tree.
2. Click on "Advanced settings > PID Parameters" in the configuration window.
3. Select the check box "Enable direct input".
4. Enter the PID parameters.
5. Double-click on "PID_3Step > Commissioning" in the project tree.
6. Establish an online connection to the CPU.
7. Load the PID parameters to the CPU.
8. Click on the "Activate controller" icon.

Result

PID_3Step changes to automatic mode and controls using the current PID parameters.

Measuring the motor transition time V1 (S7-1200, S7-1500)

Introduction

PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

Requirement

- Feedback or Feedback_PER has been selected in the basic settings and the signal has been connected.
- An online connection to the CPU has been established.

1. Select the "Use position feedback" check box.
2. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. The difference between "Target position" and "Position feedback" must be at least 50% of the valid output value range.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. PID_3Step will change to "Inactive" mode once transition time measurement is complete.

Note

Click on the icon "Upload measured transition time" to load the motor transition time measured to the project.

Actuators with endstop signals

Proceed as follows to measure the transition time of actuators with endstop signals:

Requirement

- The "Endstop signals" check box in the basic settings has been selected and Actuator_H and Actuator_L are connected.
- An online connection to the CPU has been established.

Proceed as follows to measure motor transition time with endstop signals:

1. Select the "Use actuator endstop signals" check box.
2. Select the direction in which the actuator is to be moved.
   - Open - Close - Open
     The actuator is moved first to the high endstop, then to the low endstop and then back to the high endstop.
   - Close - Open - Close
     The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved in the selected direction. Time measurement will start once the actuator has reached the first endstop and will end when the actuator reaches this endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. PID_3Step will change to "Inactive" mode once transition time measurement is complete.

Cancelling transition time measurement

PID_3Step will change to "Inactive" mode immediately if you cancel transition time measurement. The actuator will stop being moved. You can reactive PID-3Step in the curve plotter.

Simulating PID_3Step V1 with PLCSIM (S7-1200, S7-1500)

Note

Simulation with PLCSIM

For the simulation with PLCSIM, the time behavior of the simulated PLC is not exactly identical to that of a "real" PLC. The actual cycle clock of a cyclic interrupt OB can have larger fluctuations with a simulated PLC than with "real" PLCs.

In the standard configuration, PID_3Step determines the time between calls automatically and monitors them for fluctuations.

For a simulation of PID_3Step with PLCSIM, for example, a sampling time error (ErrorBits = DW#16#00000800) can therefore be detected.

This results in ongoing tuning being aborted.

The response in automatic mode depends on the value of the ActivateRecoverMode tag.

To prevent this from happening, you should configure PID_3Step for simulation with PLCSIM as follows:

- CycleTime.EnEstimation = FALSE
- CycleTime.EnMonitoring = FALSE
- CycleTime.Value: Assign the cycle clock of the calling cyclic interrupt OB in seconds to this tag.

3.3.3 Using PID_Temp (S7-1200, S7-1500)

3.3.3.1 Technology object PID_Temp (S7-1200, S7-1500)

The PID_Temp technology object provides a continuous PID controller with integrated tuning. PID_Temp is especially designed for temperature control and is suited for heating or heating/cooling applications. Two outputs are available for this purpose, one each for heating and cooling. PID_Temp can also be used for other control tasks. PID_Temp is cascadable and can be used in manual or automatic mode.

PID_Temp continuously acquires the measured process value within a control loop and compares it with the set setpoint. From the resulting control deviations, the instruction PID_Temp calculates the output value for heating and/or cooling which is used to adjust the process value to the setpoint. The output values for the PID controller consist of three actions:

- Proportional action
  The proportional action of the output value increases in proportion to the control deviation.
- Integral action
  The integral action of the output value increases until the control deviation has been balanced.
- Derivative action
  The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID_Temp calculates the proportional, integral and derivative parameters for your controlled system during "pretuning". "Fine tuning" can be used to tune the parameters further. You do not need to manually determine the parameters.

Either a fixed cooling factor or two PID parameter sets can be used for heating-and-cooling applications.

Additional information

- Overview of software controller (Page 631)
- Add technology objects (Page 633)
- Configure technology objects (Page 634)
- Configuring PID_Temp (Page 735)

3.3.3.2 Configuring PID_Temp (S7-1200, S7-1500)

Basic settings (S7-1200, S7-1500)

Introduction (S7-1200, S7-1500)

Configure the following properties of the "PID_Temp" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Start-up behavior after reset
- Source and input of the setpoint (only in the Inspector window)
- Selection of the process value
- Source and input of the process value (only in the Inspector window)
- Selection of the heating output value
- Source and input of the heating output value (only in the Inspector window)
- Activation and selection of the cooling output value
- Source and input of the cooling output value (only in the Inspector window)
- Activation of PID_Temp as master or slave of a cascade
- Number of slaves
- Selection of the master (only in the Inspector window)

Setpoint, process value, heating output value and cooling output value

You can select the source and enter values or tags for the setpoint, process value, heating output value and cooling output value in the Inspector window of the programming editor.

Select the source for each value:

- Instance DB:
  The value saved in the instance DB is used. The value must be updated by the user program in the instance DB. There should be no value at the instruction. Can be changed using HMI.
- Instruction:
  The value connected to the instruction is used. The value is written to the instance DB each time the instruction is called. Cannot be changed using HMI.

Controller type (S7-1200, S7-1500)

Physical quantity

Select the unit of measurement and physical quantity for the setpoint and the process value in the "Controller type" group. The setpoint and the process value are displayed in this unit.

Startup characteristics

1. To switch to "Inactive"mode after CPU restart, clear the "Activate Mode after CPU restart"check box.
   To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.
2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.
   After a complete "Download to device", PID_Temp starts in the selected operating mode. With each additional restart, PID_Temp starts in the mode that was last saved in Mode. When selecting pretuning or fine tuning, you also have to set or reset the Heat.EnableTuning and Cool.EnableTuning tags in order to choose between tuning for heating and tuning for cooling.

Example:

You have selected the "Activate Mode after CPU restart" check box and the "Pretuning" entry in the "Set Mode to" list. After a complete "Download to device", PID_Temp starts in the "Pretuning" mode. If pretuning is still active, PID_Temp starts in "Pretuning" mode again after restart of the CPU (heating/cooling depends on the tags Heat.EnableTuning and Cool.EnableCooling). If pretuning was successfully completed and automatic mode is active, PID_Temp starts in "Automatic mode" after restart of the CPU.

Setpoint (S7-1200, S7-1500)

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL tag in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL tag is possible, for example for the time-controlled change of the setpoint.

Process value (S7-1200, S7-1500)

PID_Temp will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Heating and cooling output value (S7-1200, S7-1500)

The PID_Temp instruction provides a PID controller with integrated tuning for temperature processes. PID_Temp is suitable for heating or heating-and-cooling applications.

PID_Temp provides the following output values. Your actuator will determine which output value you use.

- OutputHeat
  Heating output value (floating-point format): The output value for heating needs to be processed by the user program, for example, because of non-linear actuator response.
- OutputHeat_PER
  Analog heating output value: The actuator for heating is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V, 4...20 mA.
- OutputHeat_PWM
  Pulse-width modulated heating output value: The actuator for heating is controlled via a digital output. Pulse width modulation creates variable ON and OFF times.
- OutputCool
  Cooling output value (floating-point format): The output value for cooling needs to be processed by the user program, for example because of non-linear actuator response.
- OutputCool_PER
  Analog cooling output value: The actuator for cooling is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V, 4...20 mA.
- OutputCool_PWM
  Pulse-width modulated cooling output value: The actuator for cooling is controlled via a digital output. Pulse width modulation creates variable ON and OFF times.

The cooling output is only available if it was activated via the "Activate cooling" check box.

- If the check box is cleared, the output value of the PID algorithm (PidOutputSum) is scaled and output at the outputs for heating.
- If the check box is selected, positive output values of the PID algorithm (PidOutputSum) are scaled and output at the outputs for heating. Negative output values of the PID algorithm are scaled and output at the outputs for cooling. You can choose between two methods for output value calculation at the output settings.

Note

Note:

- The OutputHeat_PWM, OutputHeat_PER, OutputCool_PWM, OutputCool_PER outputs are only calculated if you select these correspondingly from the drop-down list.
- The OutputHeat output is always calculated.
- The OutputCool output is calculated if the check box for cooling is selected.
- The "Activate cooling" check box is only available if the controller is not configured as a master in a cascade.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "OutputHeat_PER" or "OutputCool_PER" in the drop-down list "OutputHeat" or "OutputCool".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the pulse-width modulated output value:

1. Select the entry "OutputHeat_PWM" or "OutputCool_PWM" in the drop-down list "OutputHeat" or "OutputCool".
2. Select "Instruction".
3. Enter the address of the digital output.

Proceed as follows to process the output value using the user program:

1. Select the entry "OutputHeat" or "OutputCool" in the drop-down list "OutputHeat" or "OutpuCool".
2. Select "Instruction".
3. Enter the name of the variable you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.

Cascade (S7-1200, S7-1500)

If a PID_Temp instance receives its setpoint from a higher-level master controller and outputs its output value in turn to a subordinate slave controller, this PID_Temp instance is both a master controller and a slave controller simultaneously. Both configurations listed below then have to be carried out for such a PID_Temp instance. This is the case, for example, for the middle PID_Temp instance in a cascade control system with three concatenated measured variables and three PID_Temp instances.

Configuring a controller as master in a cascade

A master controller defines the setpoint of a slave controller with its output.

In order to use PID_Temp as master in a cascade, you have to deactivate the cooling in the basic settings. In order to configure this PID_Temp instance as a master controller in a cascade, activate the "Controller is master" check box. The selection of the output value for heating is set automatically to OutputHeat.

OutputHeat_PWM and OutputHeat_PER cannot be used at a master in a cascade.

Subsequently specify the number of directly subordinate slave controllers that receive their setpoint from this master controller.

If no own scaling function is used when assigning the OutputHeat parameter of the master to the Setpoint parameter of the slave, it may be necessary to adapt the output value limits and the output scaling of the master to the setpoint/process value range of the slave. This can be done in the output settings of the master in the "OutputHeat / OutputCool" section.

Configuring a controller as a slave in a cascade

A slave controller receives its setpoint (Setpoint parameter) from the output of its master controller (OutputHeat parameter).

In order to configure this PID_Temp instance as a slave controller in a cascade, activate the "Controller is slave" check box in the basic settings.

Subsequently select the PID_Temp instance that is to be used as the master controller for this slave controller in the Inspector window of the programming editor. The Master and Setpoint parameters of the slave controller are interconnected with the selected master controller through this selection (the existing interconnections at these parameters are overwritten). This interconnection allows the exchange of information and the setpoint specification between master and slave. If required, the interconnection can be changed subsequently at the Setpoint parameter of the slave controller in order, for example, to insert an additional filter. The interconnection at the parameter Master may not be changed subsequently.

The "Controller is master" check box has to be selected and the number of slaves has to be configured correctly at the selected master controller. The master controller has to be called before the slave controller in the same cyclic interrupt OB.

Additional information

Additional information about program creation, configuration and commissioning when PID_Temp is used in cascade control systems is available under Cascade control with PID_Temp (Page 763).

Process value settings (S7-1200, S7-1500)

Process value limits (S7-1200, S7-1500)

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can specify how PID_Temp responds to errors in automatic mode in the output settings.

Process value scaling (S7-1200, S7-1500)

If you have configured the use of Input_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration is displayed in the Input_PER display.

Input_PER is scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input fields.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID_Temp in the programming editor.
2. Interconnect Input_PER with an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.

The existing values are overwritten with the values from the hardware configuration.

Output settings (S7-1200, S7-1500)

Basic settings of output (S7-1200, S7-1500)

Method for heating and cooling

If cooling is activated in the basic settings, two methods are available for calculating the PID output value:

- PID parameter switching (Config.AdvancedCooling = TRUE):
  The output value calculation for cooling takes place by means of a separate PID parameter set. Based on the calculated output value and the control deviation, the PID algorithm decides whether the PID parameter for heating or cooling is used. This method is suitable if the heating and cooling actuators have different time responses and different gains. Pretuning and fine tuning for cooling are only available if this method is selected.
- Cooling factor (Config.AdvancedCooling = FALSE):
  Output value calculation for cooling is effected with the PID parameters for heating under consideration of the configurable cooling factor Config.CoolFactor. This method is suitable if the heating and cooling actuators have a similar time response but different gains. If this method is selected, pretuning and fine tuning for cooling as well as the PID parameter set for cooling are not available. You can only execute the tuning for heating.

Cooling factor

If the cooling factor is selected as the method for heating/cooling, this factor is used in the calculation of the output value for cooling. This allows different gains of heating and cooling actuators to be taken into account.

The cooling factor is not set automatically or adjusted during tuning. You have to configure the correct cooling factor manually by using the ratio "Heating actuator gain/Cooling actuator gain".

Example: Cooling factor = 2.0 means that the heating actuator gain is twice as high as the cooling actuator gain.

The cooling factor is only effective and can only be changed if "Cooling factor" is selected as the method for heating/cooling.

Reaction to error

NOTICE
Your system may be damaged. If you output "Current value while error is pending " or "Substitute output value while error is pending" in the event of an error, PID_Temp remains in automatic mode or in manual mode. This may cause a violation of the process value limits and damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID_Temp is preset so that the controller stays active in most cases in the event of an error.

If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error.

PID_Temp generates a programmable output value in response to an error:

- Zero (inactive)
  At all errors, PID_Temp switches to the "Inactive" operating mode and outputs the following:
  - 0.0 as PID output value (PidOutputSum)
  - 0.0 as output value for heating (OutputHeat) and output value for cooling (OutputCool)
  - 0 as analog output value for heating (OutputHeat_PER) and analog output value for cooling (OutputCool_PER)
  - FALSE as PWM output value for heating (OutputHeat_PWM) and PWM output value for cooling (OutputCool_PWM)

  This is independent of the configured output value limits and the scaling. The controller is only reactivated by a falling edge at Reset or a rising edge at ModeActivate.
- Current value while error is pending
  The error response depends on the error occurring and the operating mode.
  If one or more of the following errors occur in automatic mode, PID_Temp stays in automatic mode:
  - 0000001h: The Input parameter is outside the process value limits.
  - 0000800h: Sampling time error
  - 0040000h: Invalid value at Disturbance parameter.
  - 8000000h: Error during the calculation of the PID parameters.

  If one or more of the following errors occur in automatic mode, PID_Temp switches to "Substitute output value with error monitoring" mode and outputs the last valid PID output value (PidOutputSum):
  - 0000002h: Invalid value at Input_PER parameter.
  - 0000200h: Invalid value at Input parameter.
  - 0000400h: Calculation of output value failed.
  - 0001000h: Invalid value at Setpoint or SubstituteSetpoint parameter.

  The values at the outputs for heating and cooling resulting from the PID output value are produced by the configured output scaling.
  As soon as the errors are no longer pending, PID_Temp switches back to automatic mode.
  If an error occurs during manual mode, PID_Temp remains in manual mode and continues to use the manual value as the PID output value.
  If the manual value is invalid, the configured substitute output value is used.
  If the manual value and substitute output value are invalid, the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit) is used.
  If the following error occurs during pretuning or fine tuning, PID_Temp remains in active mode:
  - 0000020h: Pretuning is not permitted during fine tuning.

  When any other error occurs, PID_Temp cancels the tuning and switches to the mode from which tuning was started.
- Substitute output value while error is pending

PID_Temp behaves as described at "Current value while error is pending", but outputs the configured substitute output value (SubstituteOutput) as a PID output value (PidOutputSum) in "Substitute output value with error monitoring" operating mode.
The values at the outputs for heating and cooling resulting from the PID output value are produced by the configured output scaling.
In the case of controllers with activated cooling output (Config.ActivateCooling = TRUE), enter:

– A positive substitute output value to output the value at the outputs for heating.

– A negative substitute output value to output the value at the outputs for cooling.

If the following error occurs, PID_Temp stays in "Substitute output value with error monitoring" mode and outputs the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit):

– 0020000h: Invalid value at SubstituteOutput tag.

Output value limits and scaling (S7-1200, S7-1500)

Depending on the operating mode, the PID output value (PidOutputSum) is calculated automatically by the PID algorithm or by the manual value (ManualValue) or the configured substitute output value (SubstituteOutput).

The PID output value is limited depending on the configuration:

- If the cooling is deactivated in the basic settings (Config.ActivateCooling = FALSE), the value is limited to the high limit of the PID output value (heating) (Config.Output.Heat.PidUpperLimit) and the low limit of the PID output value (heating) (Config.Output.Heat.PidLowerLimit).
  You can configure both limits at the horizontal axis of the scaling characteristic line in the "OutputHeat / OutputCool" section. These are displayed in the "OutputHeat_PWM / OutputCool_PWM" and "OutputHeat_PER / OutputCool_PER" sections, but cannot be changed.
- If the cooling is activated in the basic settings (Config.ActivateCooling = TRUE), the value is limited to the high limit of the PID output value (Config.Output.Heat.PidUpperLimit) and the low limit of the PID output value (cooling) (Config.Output.Cool.PidLowerLimit).
  You can configure both limits at the horizontal axis of the scaling characteristic line in the "OutputHeat / OutputCool" section. These are displayed in the "OutputHeat_PWM / OutputCool_PWM" and "OutputHeat_PER / OutputCool_PER" sections, but cannot be changed.
  The low limit of the PID output value (heating) (Config.Output.Heat.PidLowerLimit) and the high limit of the PID output value (cooling) (Config.Output.Cool.PidUpperLimit) cannot be changed and have to be assigned the value 0.0.

The PID output value is scaled and output at the outputs for heating and cooling. Scaling can be specified separately for each output and is specified across 2 value pairs each, consisting of a limit value of the PID output value and a scaling value:

Output	Value pair	Parameter
OutputHeat	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high output value (heating) Config.Output.Heat.UpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low output value (heating) Config.Output.Heat.LowerScaling
OutputHeat_PWM	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high PWM output value (heating) Config.Output.Heat.PwmUpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low PWM output value (heating) Config.Output.Heat.PwmLowerScaling
OutputHeat_PER	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high analog output value (heating) Config.Output.Heat.PerUpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low analog output value (heating) Config.Output.Heat.PerLowerScaling
OutputCool	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high output value (cooling) Config.Output.Cool.UpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low output value (cooling) Config.Output.Cool.LowerScaling
OutputCool_PWM	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high PWM output value (cooling) Config.Output.Cool.PwmUpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low PWM output value (cooling) Config.Output.Cool.PwmLowerScaling

Output	Value pair	Parameter
OutputCool_PER	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high analog output value (cooling) Config.Output.Cool.PerUpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low analog output value (cooling) Config.Output.Cool.PerLowerScaling

The low limit of PID output value (heating) (Config.Output.Heat.PidLowerLimit) has to have the value 0.0, if the cooling is activated (Config.ActivateCooling = TRUE).

The high limit of PID output value (cooling) (Config.Output.Cool.PidUpperLimit) must always have the value 0.0.

Example:

Output scaling when the OutputHeat output is used (cooling deactivated. The low limit of PID output value (heating) (Config.Output.Heat.PidLowerLimit) may be unequal to 0.0):

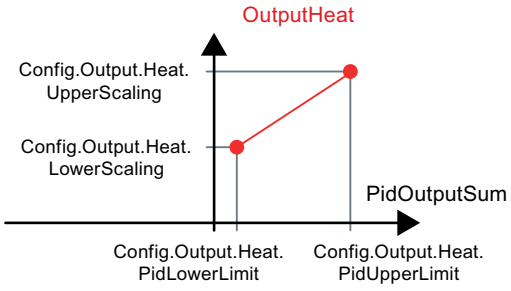

Example:

Output scaling when the OutputHeat_PWM and OutputCool_PER outputs are used (cooling activated. The low limit of PID output value (heating) (Config.Output.Heat.PidLowerLimit) must be 0.0):

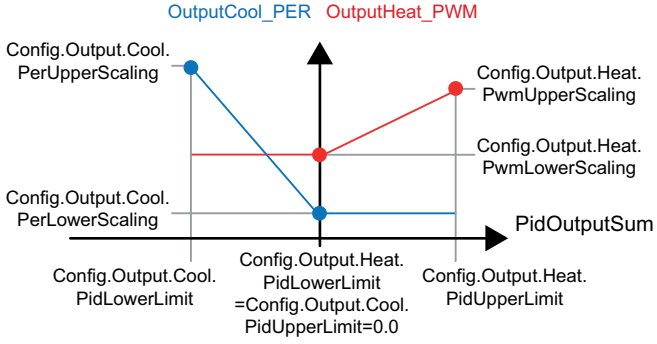

With the exception of the "Inactive" operating mode, the value at an output always lies between its scaled high output value and the scaled low output value, for example for OutputHeat always

between the scaled high output value (heating) (Config.Output.Heat.UpperScaling) and the scaled low output value (heating) (Config.Output.Heat.LowerScaling).

If you want to limit the value at the associated output, you therefore have to adapt these scaling values as well.

You can configure the scaling values of an output at the vertical axes of the scaling characteristic line. Each output has two separate scaling values. These can only be changed for OutputHeat_PWM, OutputCool_PWM, OutputHeat_PER and OutputCool_PER if the corresponding output is selected in the basic settings. The cooling has to be activated additionally in the basic settings at all the outputs for cooling.

The trend view in the commissioning dialog box only records the values of OutputHeat and OutputCool, irrespective of the selected output in the basic settings. Therefore, if necessary, adapt the scaling values for OutputHeat or OutputCool if you use OutputHeat_PWM or OutputHeat_PER or OutputCool_PWM or OutputCool_PER and want to use the trend view in the commissioning dialog.

Advanced settings (S7-1200, S7-1500)

Process value monitoring (S7-1200, S7-1500)

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning is displayed at the PID_Temp instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits are used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_Temp will respond as follows:

Process value	InputWarning_H	InputWarning_L	ErrorBits
> 98 °C	TRUE	FALSE	0001h
≤ 98° C and > 90° C	TRUE	FALSE	0000h
≤ 90° C and ≥ 10° C	FALSE	FALSE	0000h
< 10° C and ≥ 0° C	FALSE	TRUE	0000h
< 0° C	FALSE	TRUE	0001h

You can configure the response of PID_Temp when the process value high limit or low limit is violated in the output settings.

PWM limits (S7-1200, S7-1500)

The PID output value PidOutputSum is scaled and transformed via a pulse width modulation into a pulse train that is output at the output parameter OutputHeat_PWM or OutputCool_PWM. The "Sampling time of PID algorithm" represents the time between two calculations of the PID output value. The sampling time is used as time period of the pulse width modulation.

During heating, the PID output value is always calculated in the "Sampling time of PID algorithm for heating".

Calculation of the PID output value during cooling depends on the type of cooling selected in "Basic settings Output":

- If the cooling factor is used, the "Sampling time of PID algorithm for heating" applies.
- If the PID parameter switching is used, the "Sampling time of PID algorithm for cooling" applies.

OutputHeat_PWM and OutputCool_PWM are output in the sampling time PID_Temp (corresponds to the cycle time of the calling OB).

The PID algorithm sampling time for heating or cooling is determined during pretuning or fine tuning. If you set the PID parameters manually, you will also need to configure the PID algorithm sampling time for heating or cooling. The PID_Temp sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the PID output value and is always an integer multiple of the PID_Temp sampling time.

Example for OutputHeat_PWM

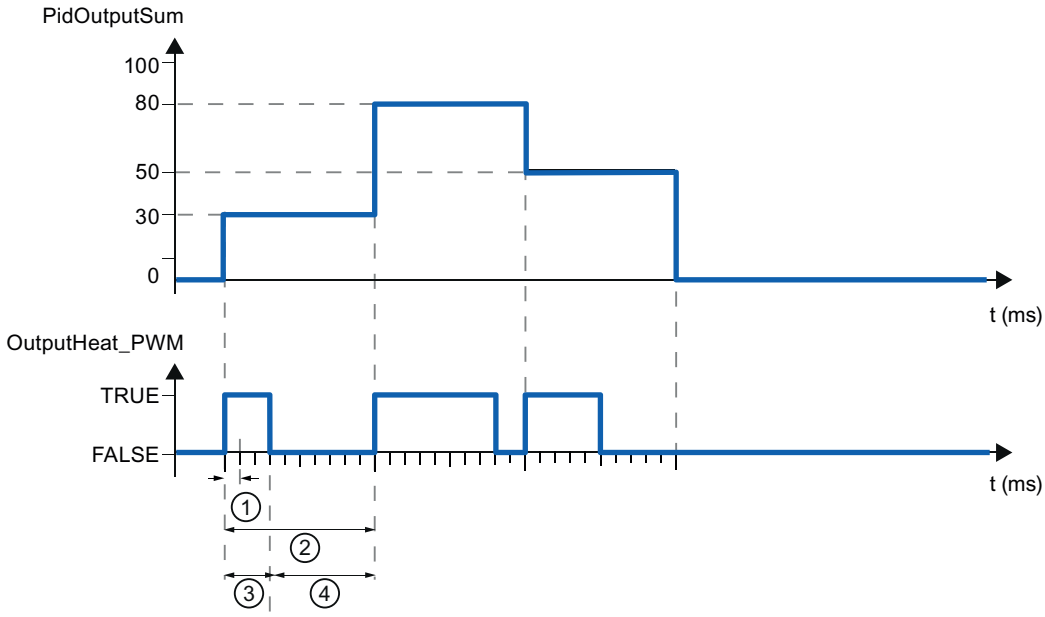

① PID_Temp sampling time
② PID algorithm sampling time for heating
③ Pulse duration
④ Break time

The "Minimum ON time" and the "Minimum OFF time" can be set separately for heating and cooling, rounded to an integer multiple of the PID_Temp sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

Example for OutputHeat_PWM

PID_Temp sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

The PID output value PidOutputSum amounts to 15% constantly. The smallest pulse that PID_Temp can output corresponds to 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.

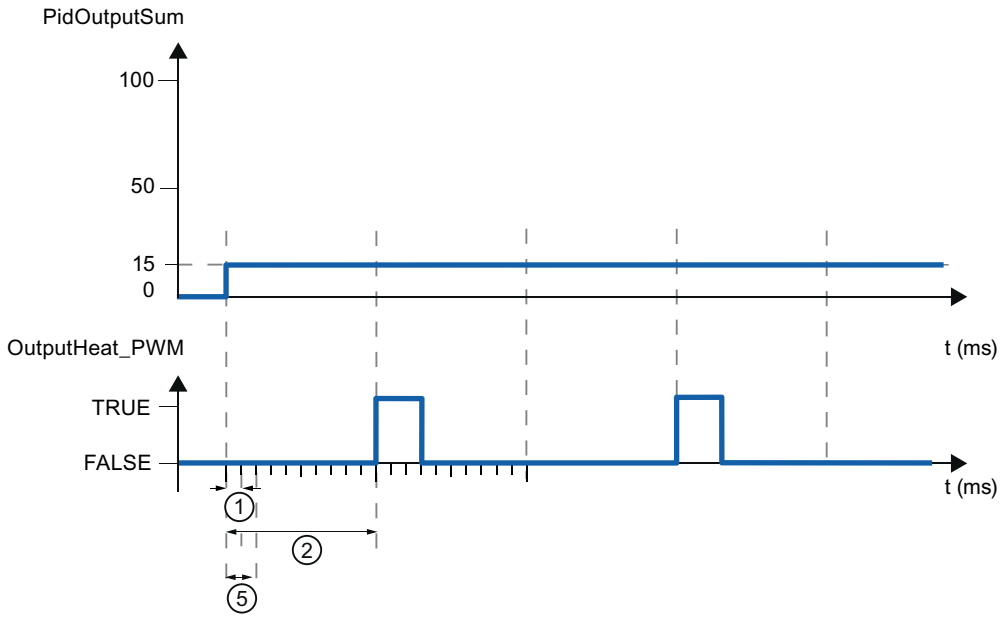

① PID_Temp sampling time

② PID algorithm sampling time for heating

⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you have selected OutputHeat/OutputCool or OutputHeat_PER/OutputCool_PER as the output in the basic settings, the minimum ON time and the minimum OFF time are not evaluated and cannot be changed.

If the "Sampling time of PID algorithm" (Retain.CtrlParams.Heat.Cycle or Retain.CtrlParams.Cool.Cycle) and thus the period duration of the pulse width modulation is very high when OutputHeat_PWM or OutputCool_PWM is used, you can specify a deviating shorter period duration at the parameters Config.Output.Heat.PwmPeriode or

Config.Output.Cool.PwmPeriode in order to improve smoothness of the process value (see also AUTOHOTSPOT).

Note

The minimum ON and OFF times only affect the output parameters OutputHeat_PWM or OutputCool_PWM and are not used for any pulse generators integrated in the CPU.

PID parameters (S7-1200, S7-1500)

The PID parameters are displayed in the "PID Parameters" configuration window.

If cooling is activated in the basic settings and PID parameter switching is selected as the method for heating/cooling in the output settings, two parameter sets are available: One for heating and one for cooling.

In this case, the PID algorithm decides on the basis of the calculated output value and the control deviation whether the PID parameters for heating or cooling are used.

If cooling is deactivated or the cooling factor is selected as the method for heating/cooling, the parameter set for heating is always used.

During tuning, the PID parameters are adapted to the controlled system with the exception of the dead zone width that has to be configured manually.

Note

The currently active PID parameters are located in the Retain.CtrlParams structure.

Change the currently active PID parameters only in "Inactive" mode online to prevent malfunction of the PID controller.

If you want to change the PID parameters in "Automatic mode" or "Manual mode" online, change the PID parameters in the CtrlParamsBackUp structure and apply these changes with LoadBackUp = TRUE to the Retain.CtrlParams structure.

Online changes to the PID parameters in "Automatic mode" can result in jumps at the output value.

PID_Temp is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions.

The PID algorithm operates according to the following equation (control zone and dead zone deactivated):

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description	Associated parameters of the PID_Temp instruction
y	Output value of the PID algorithm	-
K_p	Proportional gain	Retain.CtrlParams.Heat.Gain Retain.CtrlParams.Cool.Gain CoolFactor
s	Laplace operator	-
b	Proportional action weighting	Retain.CtrlParams.Heat.PWeighting Retain.CtrlParams.Cool.PWeighting
w	Setpoint	CurrentSetpoint
x	Process value	ScaledInput
T_I	Integral action time	Retain.CtrlParams.Heat.Ti Retain.CtrlParams.Cool.Ti
T_D	Derivative action time	Retain.CtrlParams.Heat.Td Retain.CtrlParams.Cool.Td
a	Coefficient for derivative-action delay (Derivative delay T1 = a × T_D)	Retain.CtrlParams.Heat.TdFiltRatio Retain.CtrlParams.Cool.TdFiltRatio
c	Derivative action weighting	Retain.CtrlParams.Heat.DWeighting Retain.CtrlParams.Cool.DWeighting
DeadZone	Dead zone width	Retain.CtrlParams.Heat.DeadZone Retain.CtrlParams.Cool.DeadZone
ControlZone	Control zone width	Retain.CtrlParams.Heat.ControlZone Retain.CtrlParams.Cool.ControlZone

The diagram below illustrates the integration of the parameters into the PID algorithm:

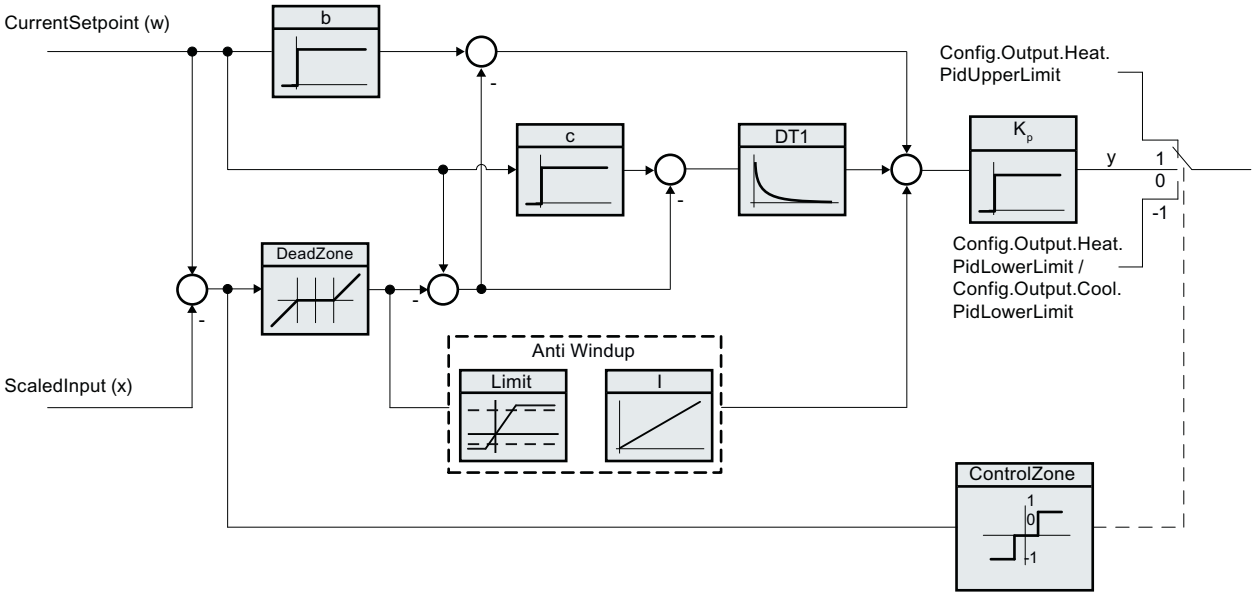

All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_Temp (Downloading technology objects to device (Page 636)).

PID_Temp block diagram

The following block diagram shows how the PID algorithm is integrated in the PID_Temp.

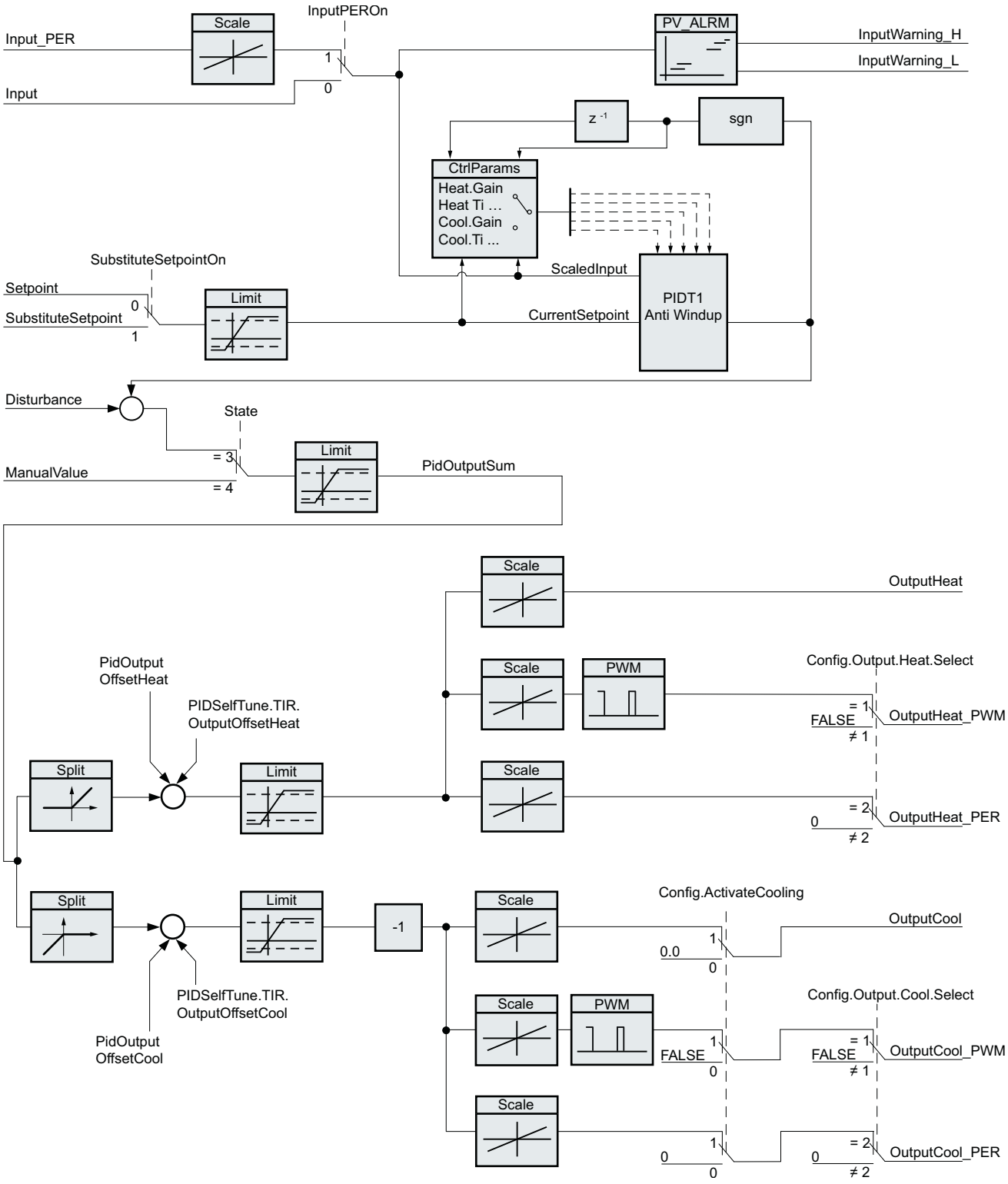

Proportional gain

The value specifies the proportional gain of the controller. PID_Temp does not operate with a negative proportional gain and only supports the normal control direction, meaning that an increase in the process value is achieved by an increase in the PID output value (PidOutputSum).

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0. When the integral action time is changed from a different value to 0.0 online in "Automatic mode", the previous integral action is deleted and the output value jumps.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with one dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of "PID algorithm" represents the time between two calculations of the PID output value. It is calculated during tuning and rounded to a multiple of the PID_Temp sampling time (cycle time of the cyclic interrupt OB). All other functions of PID_Temp are executed at every call.

If you use OutputHeat_PWM or OutputCool_PWM, the sampling time of the PID algorithm is used as the period duration of the pulse width modulation. The accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The cycle time should be no more than a tenth of the PID algorithm sampling time.

The sampling time of the PID algorithm that is used as the period duration of the pulse width modulation at OutputCool_PWM depends on the method for heating/cooling selected in "Basic settings Output":

- If the cooling factor is used, the "sampling time of the PID algorithm for heating" also applies to OutputCool_PWM.
- If PID parameter switching is used, the "sampling time PID algorithm for cooling" applies as the period duration for OutputCool_PWM.

If the sampling time of the PID algorithm and thus the period duration of the pulse width modulation is very high when OutputHeat_PWM or OutputCool_PWM is used, you can specify a deviating shorter period duration at the parameters Config.Output.Heat.PwmPeriode or Config.Output.Cool.PwmPeriode in order to improve smoothness of the process value.

Dead zone width

If the process value is affected by noise, the noise can also have an effect on the output value. The output value may fluctuate considerably when controller gain is high and the derivative action is activated. If the process value lies within the dead zone around the setpoint, the control deviation is suppressed so that the PID algorithm does not react and unnecessary fluctuations of the output value are reduced.

The dead zone width for heating or cooling is not set automatically during tuning. You have to correctly configure the dead zone width manually. The dead zone is deactivated by setting the dead zone width = 0.0.

When the dead zone is switched on, the result can be a permanent control deviation (deviation between setpoint and process value). This can have a negative effect on fine tuning.

If cooling is activated in the basic settings and PID parameter switching is selected as the method for heating/cooling in the output settings, the dead zone lies between "Setpoint - dead zone width (heating)" and "Setpoint + dead zone width (cooling)".

If cooling is deactivated in the basic settings or the cooling factor is used, the dead zone lies symmetrically between "Setpoint - dead zone width (heating)" and "Setpoint + dead zone width (heating)".

If values not equal to 1.0 are configured for the proportional action weighting or the derivative action weighting, setpoint changes even within the dead zone affect the output value.

Process value changes within the dead zone do not affect the output value, regardless of the weighting.

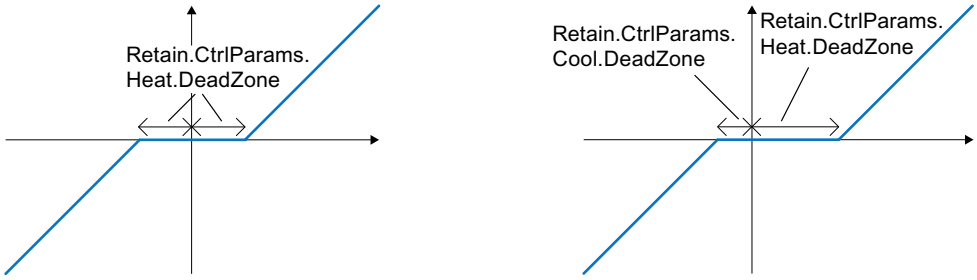

Dead zone with deactivated cooling or cooling factor (left) or activated cooling and PID parameter switching (right). The x / horizontal axis displays the control deviation = setpoint - process value. The y / vertical axis shows the output signal of the dead zone that is passed to the PID algorithm.

Control zone width

If the process value exits the control zone around the setpoint, the minimum or maximum output value is output. This means that the process value reaches the setpoint faster.

If the process value lies within the control zone around the setpoint, the output value is calculated by the PID algorithm.

The control zone width for heating or cooling is only set automatically during the pretuning, if "PID (temperature)" is selected as the controller structure for cooling or heating.

The control zone is deactivated by setting the control zone width = 3.402822e+38.

If cooling is deactivated in the basic settings or the cooling factor is used, the control zone lies symmetrically between "Setpoint - control zone width (heating)" and "Setpoint + control zone width (heating)".

If cooling is activated in the basic settings and PID parameter switching is selected as the method for heating/cooling in the output settings, the control zone lies between "Setpoint - control zone width (heating)" and "Setpoint + control zone width (cooling)".

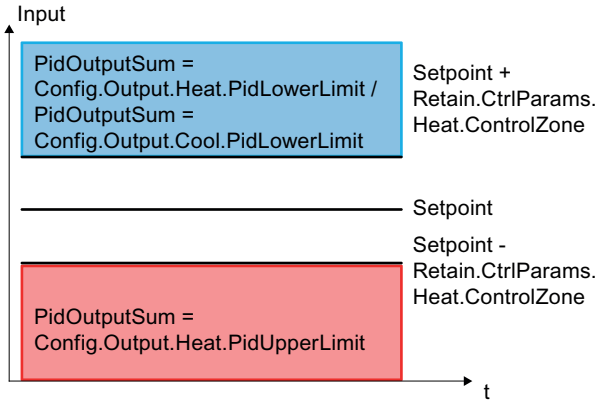

Control zone with deactivated cooling or cooling factor.

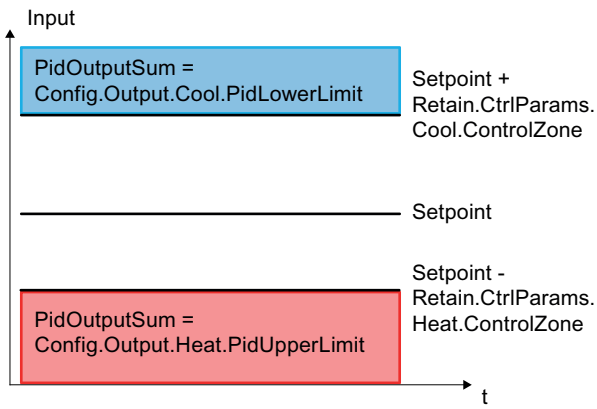

Control zone with activated cooling and PID parameter switching.

Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list. You can specify the rules for tuning for heating and for tuning for cooling separately.

- PID (temperature)
  Calculates PID parameters during pretuning and fine tuning.
  Pretuning is designed for temperature processes and results in a slower and rather asymptotic control response with lower overshoot than with the "PID" option. Fine tuning is identical to the "PID" option.
  The control zone width is determined automatically during pretuning only if this option is selected.
- PID
  Calculates PID parameters during pretuning and fine tuning.
- PI
  Calculates PI parameters during pretuning and fine tuning.
- User-defined
  The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program or the parameter view.

3.3.3.3 Commissioning PID_Temp (S7-1200, S7-1500)

Commissioning (S7-1200, S7-1500)

The commissioning window helps you commission the PID controller. You can monitor the values for the setpoint, process value and the output values for heating and cooling along the time axis in the trend view. The following functions are supported in the commissioning window:

- Controller pretuning
- Controller fine tuning
  Use fine tuning for fine adjustments to the PID parameters.
- Monitoring the current closed-loop control in the trend view

- Testing the controlled system by specifying a manual PID output value and a substitute setpoint
- Saving the actual values of the PID parameters to an offline project.

All functions require an online connection to the CPU.

The online connection to the CPU is established, if it does not exist already, and operation of the commissioning window is enabled by means of the "Monitor all" or "Start" buttons of the trend view.

Operation of the trend view

- Select the desired sampling time in the "Sampling time" drop-down list.
  All the values of the trend view are updated in the selected sampling time.
- Click the "Start" icon in the Measurement group if you want to use the trend view.
  Value recording is started. The current values for the setpoint, process value and output values for heating and cooling are entered in the trend view.
- Click the "Stop" icon if you want to end the trend view.
  The values recorded in the trend view can continue to be analyzed.

Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

Pretuning (S7-1200, S7-1500)

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" or "Manual mode". The PID parameters are backed up before being recalculated.

PID_Temp offers different pretuning types depending on the configuration:

- Pretuning heating
  A jump is output at the output value heating, the PID parameters for heating are calculated and then the setpoint is used as the control variable in automatic mode.
- Pretuning heating and cooling
  A jump is output at the output value heating.
  As soon as the process value is close to the setpoint, a jump change is output at the output value cooling.
  The PID parameters for heating (Retain.CtrlParams.Heat structure) and cooling (Retain.CtrlParams.Cool structure) are calculated and then the setpoint is used as the control variable in automatic mode.
- Pretuning cooling
  A jump is output at the output value cooling.
  The PID parameters for cooling are calculated and then the setpoint is used as the control variable in automatic mode.

If you want to tune the PID parameters for heating and cooling, you can expect a better control response with "Pretuning heating" followed by "Pretuning cooling" rather than with "Pretuning heating and cooling". However, carrying out pretuning in two steps takes more time.

General requirements

- The PID_Temp instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- PID_Temp is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".
- The setpoint and the process value lie within the configured limits (see Process value monitoring (Page 746) configuration).

Requirements for pretuning heating

- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is greater than 50% of the setpoint.
- The setpoint is greater than the process value.

Requirements for pretuning heating and cooling

- The cooling output in the "Basic settings" is activated (Config.ActivateCooling = TRUE).
- The PID parameter switching in the "Basic settings of output value" is activated (Config.AdvancedCooling = TRUE).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is greater than 50% of the setpoint.
- The setpoint is greater than the process value.

Requirements for pretuning cooling

- The cooling output in the "Basic settings" is activated (Config.ActivateCooling = TRUE).
- The PID parameter switching in the "Basic settings of output value" is activated (Config.AdvancedCooling = TRUE).
- "Pretuning heating" or "Pretuning heating and cooling" has been carried out successfully (PIDSelfTune.SUT.ProcParHeatOk = TRUE). The same setpoint should be used for all tunings.
- The difference between setpoint and process value is smaller than 5% of the difference between process value high limit and process value low limit.

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all" button or start the trend view.
   An online connection will be established.
3. Select the desired pretuning entry from the "Tuning mode" drop-down list.
4. Click the "Start" icon.
   – Pretuning is started.
   – The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar ("Progress" tag) has not changed for a long period and it is to be assumed that the tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_Temp switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_Temp responds with the configured reaction to errors.

Fine tuning (S7-1200, S7-1500)

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. The PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID_Temp automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

PID_Temp offers different fine tuning types depending on the configuration:

- Fine tuning heating:
  PID_Temp generates an oscillation of the process value with periodic changes at the output value heating and calculates the PID parameters for heating.
- Fine tuning cooling:
  PID_Temp generates an oscillation of the process value with periodic changes at the output value cooling and calculates the PID parameters for cooling.

Temporary tuning offset for heating/cooling controllers

If PID_Temp is used as a heating/cooling controller (Config.ActivateCooling = TRUE), the PID output value (PidOutputSum) at the setpoint has to fulfill the following requirements so that process value oscillation can be generated and fine tuning can be carried out successfully:

- Positive PID output value for fine tuning heating
- Negative PID output value for fine tuning cooling

If this condition is not fulfilled, you can specify a temporary offset for fine tuning that is output at the opposing output.

- Offset for cooling output (PIDSelfTune.TIR.OutputOffsetCool) with fine tuning heating. Before starting tuning, enter a negative tuning offset cooling that is smaller than the PID output value (PidOutputSum) at the setpoint in the stationary state.
- Offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) at fine tuning cooling Before starting tuning, enter a positive tuning offset heating that is greater than the PID output value (PidOutputSum) at the setpoint in the stationary state.

The defined offset is balanced by the PID algorithm so that the process value remains at the setpoint. The height of the offset allows the PID output value to be adapted correspondingly so that it fulfills the requirement mentioned above.

To avoid larger overshoots of the process value when defining the offset, it can also be increased in several steps.

If PID_Temp exits the fine tuning mode, the tuning offset is reset.

Example: Specification of an offset for fine tuning cooling

- Without offset
  - Setpoint = Process value (ScaledInput) = 80 °C
  - PID output value (PidOutputSum) = 30.0
  - Output value heating (OutputHeat) = 30.0
  - Output value cooling (OutputCool) = 0.0
    Oscillation of the process value around the setpoint cannot be generated with the cooling output alone. Fine tuning would fail here.
- With offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) = 80.0
  - Setpoint = Process value (ScaledInput) = 80 °C
  - PID output value (PidOutputSum) = -50.0
  - Output value heating (OutputHeat) = 80.0
  - Output value cooling (OutputCool) = -50.0
    Thanks to the specification of an offset for the heating output, the cooling output can now generate oscillation of the process value around the setpoint. Fine tuning can now be carried out successfully.

General requirements

- The PID_Temp instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
  When the dead zone is switched on, the result can be a permanent control deviation (deviation between setpoint and actual value). This can have a negative effect on fine tuning.
- No disturbances are expected.
- PID_Temp is in inactive mode, automatic mode or manual mode.

Requirements for fine tuning heating

- Heat.EnableTuning = TRUE
- Cool.EnableTuning = FALSE
- If PID_Temp is configured as a heating-and-cooling controller (Config.ActivateCooling = TRUE), the heating output has to be active at the operating point where tuning is to be carried out.
  PidOutputSum > 0.0 (see tuning offset)

Requirements for fine tuning cooling

- Heat.EnableTuning = FALSE
- Cool.EnableTuning = TRUE
- The cooling output is activated (Config.ActivateCooling = TRUE).
- The PID parameter switching is activated (Config.AdvancedCooling = TRUE).
- The cooling output has to be active at the operating point where tuning is to be carried out.
  PidOutputSum < 0.0 (see tuning offset)

Process depends on initial situation

Fine tuning can be started from the following operating modes: "Inactive", "automatic mode", or "manual mode".

Fine tuning proceeds as follows when started from:

- Automatic mode with PIDSelfTune.TIR.RunIn = FALSE (default)
  Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.
  PID_Temp controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive, manual mode or automatic mode with PIDSelfTune.TIR.RunIn = TRUE
  An attempt is made to reach the setpoint with the minimum or maximum output value (two-point control):
  - With minimum or maximum output value heating at fine tuning heating.
  - With minimum or maximum output value cooling for fine tuning cooling.

  This can produce increased overshoot. Fine tuning starts when the setpoint is reached.
  If the setpoint cannot be reached, PID_Temp does not automatically abort tuning.

Procedure

To perform fine tuning, follow these steps:

1. Double-click the "PID_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all" button or start the trend view.
   An online connection will be established.
3. Select the desired fine tuning entry from the "Tuning mode" drop-down list.
4. If required (see tuning offset), specify a tuning offset and wait until the stationary state is reached again.
5. Click the "Start" icon.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group if the progress bar ("Progress" tag) has not changed for a long period and it is to be assumed that the tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

In the following phases in particular, tuning is not aborted automatically if the setpoint cannot be reached.

- "Attempting to reach setpoint for heating with two-point control."
- "Attempting to reach setpoint for cooling with two-point control."

Result

If fine tuning was performed without errors, the PID parameters have been tuned. PID_Temp switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID_Temp responds with the configured response to errors.

"Manual" mode (S7-1200, S7-1500)

The following section describes how you can use "Manual mode" in the commissioning window of the "PID_Temp" technology object.

Manual mode is also possible when an error is pending.

Requirement

- The "PID_Temp" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established.
- The CPU is in "RUN" mode.

Procedure

If you want to test the controlled system by specifying a manual value, use "Manual mode" in the commissioning window.

To define a manual value, follow these steps:

1. Double-click the "PID_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all" button or start the trend view.
   An online connection will be established.
3. Select the "Manual mode" check box in the "Online status of controller" area.
   PID_Temp operates in manual mode. The most recent current output value remains in effect.
4. Enter the manual value in the editable field as a % value.
   If cooling is activated in the basic settings, enter the manual value as follows:
   – Enter a positive manual value to output the value at the outputs for heating.
   – Enter a negative manual value to output the value at the outputs for cooling.
5. Click the icon.

Result

The manual value is written to the CPU and immediately goes into effect.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller.

The switchover to automatic mode is bumpless.

Substitute setpoint (S7-1200, S7-1500)

The following section describes how you can use the substitute setpoint in the commissioning window of the "PID_Temp" technology object.

Requirement

- The "PID_Temp" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established.
- The CPU is in "RUN" mode.

Procedure

If you want to use a different value as the setpoint than that specified at the "Setpoint" parameter (for example to tune a slave in a cascade), use the substitute setpoint in the commissioning window.

Proceed as follows to specify a substitute setpoint:

1. Double-click the "PID_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all" button or start the trend view.
   An online connection will be established.
3. Select the "Subst.Setpoint" check box in the "Online status of controller" section.
   The substitute setpoint (SubstituteSetpoint tag) is initialized with the most recently updated setpoint and now used.
4. Enter the substitute setpoint in the editable field.
5. Click the icon.

Result

The substitute setpoint is written to the CPU and immediately goes into effect.

Clear the "Subst.Setpoint" check box if the value at the "Setpoint" parameter is to be used again as setpoint.

The switchover is not bumpless.

Cascade commissioning (S7-1200, S7-1500)

Information about cascade commissioning with PID_Temp is available under Commissioning (Page 767).

3.3.3.4 Cascade control with PID_Temp (S7-1200, S7-1500)

Introduction (S7-1200, S7-1500)

In cascade control, several control loops are nested within each other. In the process, slaves receive their setpoint (Setpoint) from the output value (OutputHeat) of the respective higher-level master.

A prerequisite for establishing a cascade control system is that the controlled system can be divided into subsystems, each with its own measured variable.

Setpoint specification for the controlled variable is carried out at the outmost master.

The output value of the innermost slave is applied to the actuator and thus acts on the controlled system.

The following major advantages result from the use of a cascade control system in comparison with a single-loop control system:

- Thanks to the additional subordinate control loops, disturbances which occur there are corrected quickly. Their influence on the controlled variable is reduced considerably. The disturbance behavior is thus improved.
- The subordinate control loops act in linearizing form. The negative effects of such non-linearities on the controlled variable are thus moderated.

PID_Temp offers the following functionality especially for use in cascade control systems:

- Specification of a substitute setpoint
- Exchange of status information between master and slave (for example, current operating mode)
- Different Anti-Wind-Up modes (response of the master to limitation of its slave)

Example

The following block diagram shows a cascade control system with PID_Temp using the simplified example of a chocolate melting unit:

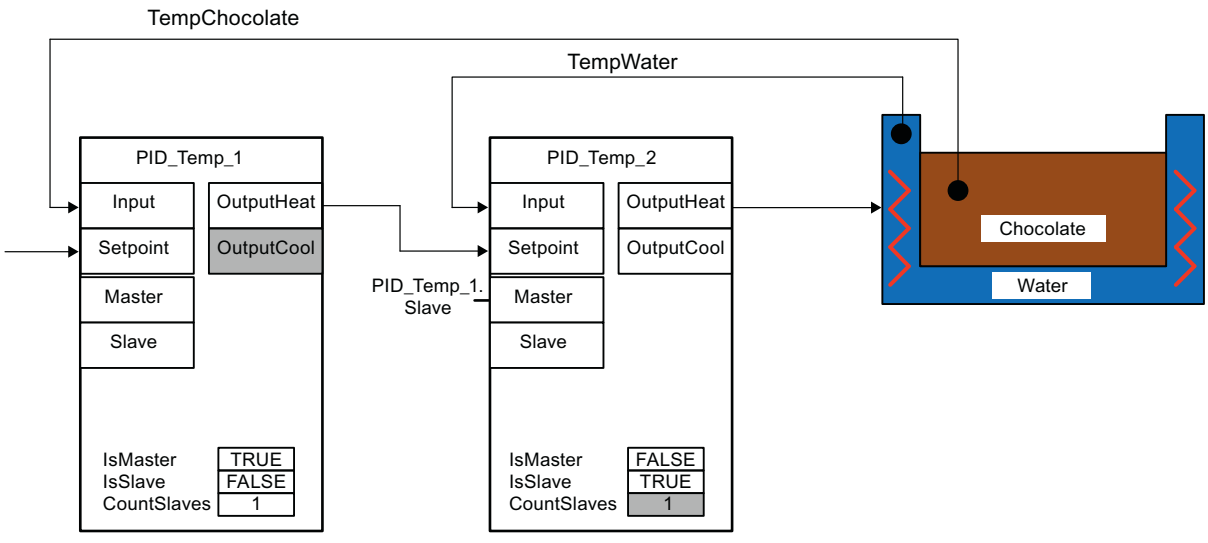

The PID_Temp_1 master compares the process value of the chocolate temperature (TempChocolate) with the setpoint specification by the user at the Setpoint parameter. Its output value OutputHeat forms the setpoint of the slave PID_Temp_2.

PID_Temp_2 attempts to regulate the process value of the water-bath temperature (TempWater) to this setpoint. The output value of PID_Temp_2 acts directly on the actuator of the controlled system (heating of the water bath) and thus influences the water-bath temperature. The water-bath temperature in turn has an effect on the chocolate temperature.

FAQ

For more information, see the following FAQs in the Siemens Industry Online Support:

- Entry ID 103526819 (https://support.industry.siemens.com/cs/ww/en/view/103526819)

See also

Program creation (Page 765)

Program creation (S7-1200, S7-1500)

Observe the following points during program creation:

- Number of PID_Temp instances
  The number of different PID_Temp instances called up in a cyclic interrupt OB has to agree with the number of concatenated measured variables in the process.
  There are two concatenated measured variables in the example: TempChocolate and TempWater. Therefore two PID_Temp instances are required.
- Call sequence
  A master has to be called before its slaves in the same cyclic interrupt OB.
  The outermost master at which the user setpoint is specified is called first.
  The slave whose setpoint is specified by the outermost master is called next, etc.
  The innermost slave that acts on the actuator of the process with its output value is called last.
  In the example, PID_Temp_1 is called before PID_Temp_2.
- Interconnection of the measured variables
  The outermost master is interconnected with the outermost measured variable that is to be regulated to the user setpoint.
  The innermost slave is interconnected with the innermost measured variable that is influenced directly by the actuator.
  Interconnection of the measured variables with PID_Temp is carried out with the parameters Input or Input_PER.
  In the example, the outermost measured variable TempChocolate is interconnected with PID_Temp_1 and the innermost measured variable TempWater with PID_Temp_2.

- Interconnection of the output value of the master to the setpoint of the slave
  The output value (OutputHeat) of a master has to be assigned to the setpoint (Setpoint) of its slave.
  This interconnection can be carried out in the programming editor or automatically in the Inspector window of the slave in the basic settings via the selection of the master.
  If required, you can insert your own filter or scaling functions, for example in order to adapt the output value range of the master to the setpoint/process value range of the slave.
  In the example, OutputHeat of PID_Temp_1 is assigned to Setpoint of PID_Temp_2.
- Interconnection of the interface for information exchange between master and slave
  The "Slave" parameter of a master has to be assigned to the "Master" parameter of all its directly subordinate slaves (which receive their setpoint from this master). The assignment should be carried out via the interface of the slave in order to allow the interconnection of a master with multiple slaves and the display of the interconnection in the Inspector window of the slave in the basic settings.
  This interconnection can be carried out in the programming editor or automatically in the Inspector window of the slave in the basic settings via the selection of the master.
  The Anti-Wind-Up functionality and the evaluation of the slave operating modes at the master can only function correctly if this interconnection is carried out.
  In the example, the "Slave" parameter of PID_Temp_1 is assigned to the "Master" parameter of PID_Temp_2.

Program code of the example using SCL (without assignment of the output value of the slave to the actuator):

```
"PID_Temp_1"(Input:="TempChocolate");
```

```
"PID_Temp_2"(Input:="TempWater", Master := "PID_Temp_1".Slave,
Setpoint := "PID_Temp_1".OutputHeat);
```

Configuration (S7-1200, S7-1500)

You can carry out the configuration via your user program, the configuration editor or the Inspector window of the PID_Temp call.

When using PID_Temp in a cascade control system, ensure the correct configuration of the settings specified below.

If a PID_Temp instance receives its setpoint from a superior master controller and outputs its output value in turn to a subordinate slave controller, this PID_Temp instance is both a master controller and a slave controller simultaneously. Both configurations listed below have to be carried out for such a PID_Temp instance. This is the case, for example, for the middle PID_Temp instance in a cascade control system with three concatenated measured variables and three PID_Temp instances.

Configuration of a master

Setting in the configuration editor or Inspector window	DB parameter	Explanation
Basic settings → Cascade: Activate "Controller is master" check box	Config.Cascade.IsMaster = TRUE	Activates this controller as a master in a cascade
Basic settings → Cascade: Number of slaves	Config.Cascade.CountSlaves	Number of directly subordinate slaves that receive their setpoint directly from this master

Setting in the configuration editor or Inspector window	DB parameter	Explanation
Basic settings → Input/output parameters: Selection of the output value (heating) = OutputHeat	Config.Output.Heat.Select = 0	The master only uses the output parameter OutputHeat. OutputHeat_PWM and OutputHeat_PER are deactivated.
Basic settings → Input/output parameters: Clear "Activate cooling" check box	Config.ActivateCooling = FALSE	The cooling has to be deactivated at a master.
Output settings → Output limits and scaling → OutputHeat / OutputCool: PID output value low limit (heating), PID output value high limit (heating), Scaled low output value (heating), Scaled high output value (heating)	Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PidUpperLimit, Config.Output.Heat.LowerScaling, Config.Output.Heat.UpperScaling	If no own scaling function is used when assigning OutputHeat of the master to Setpoint of the slave, it may be necessary to adapt the output value limits and the output scaling of the master to the setpoint/process value range of the slave.
This tag is not available in the Inspector window or in the functional view of the configuration editor. You can change it via the parameter view of the configuration editor.	Config.Cascade.AntiWindUpMode	The Anti-Wind-Up mode determines how the integral action of this master is treated if directly subordinate slaves reach their output value limits. Options are: • AntiWindUpMode = 0: The AntiWindUp functionality is deactivated. The master does not react to the limitation of its slaves. • AntiWindUpMode = 1 (default): The integral action of the master is reduced in the relationship "Slaves in limitation/ Number of slaves". This reduces the effects of the limitation on the control behavior. • AntiWindUpMode = 2: The integral action of the master is held as soon as a slave is in limitation.

Configuration of a slave

Setting in the configuration editor or Inspector window	DB parameter	Explanation
Basic settings → Cascade: Select the "Controller is slave" check box	Config.Cascade.IsSlave = TRUE	Activates this controller as a slave in a cascade

Commissioning (S7-1200, S7-1500)

After compiling and loading of the program, you can start commissioning of the cascade control system.

Begin with the innermost slave at commissioning (implementation of tuning or change to automatic mode with existing PID parameters) and continue outwards until the outermost master has been reached.

In the above example, commissioning starts with PID_Temp_2 and is continued with PID_Temp_1.

Tuning the slave

Tuning of PID_Temp requires a constant setpoint. Therefore, activate the substitute setpoint of a slave (SubstituteSetpoint and SubstituteSetpointOn tags) to tune the slave or set the associated master to manual mode with a corresponding manual value. This ensures that the setpoint of the slave remains constant during tuning.

Tuning the master

In order for a master to influence the process or to carry out tuning, all the downstream slaves have to be in automatic mode and their substitute setpoint has to be deactivated. A master evaluates these conditions through the interface for information exchange between master and slave (Master parameter and Slave parameter) and displays the current state at the AllSlaveAutomaticState and NoSlaveSubstituteSetpoint tags. Corresponding status messages are output in the commissioning editor.

Status message in the commissioning editor of the master	DB parameter of the master	Correction
One or more slaves are not in automatic mode.	AllSlaveAutomaticState = FALSE, NoSlaveSubstituteSetpoint = TRUE	First, carry out commissioning of all downstream slaves. Ensure that the following conditions are fulfilled before carrying out tuning or activating manual mode or automatic mode of the master: • All downstream slaves are in automatic mode (state = 3). • All downstream slaves have deactivated the substitute setpoint (SubstituteSetpointOn = FALSE).
One or more slaves have activated the substitute setpoint.	AllSlaveAutomaticState = TRUE, NoSlaveSubstituteSetpoint = FALSE	
One or more slaves are not in automatic mode and have activated the substitute setpoint.	AllSlaveAutomaticState = FALSE, NoSlaveSubstituteSetpoint = FALSE	

If pretuning or fine tuning is started for a master, PID_Temp aborts tuning in the following cases and displays an error with ErrorBits = DW#16#0200000:

- One or more slaves are not in automatic mode (AllSlaveAutomaticState = FALSE)
- One or more slaves have activated the substitute setpoint (NoSlaveSubstituteSetpoint = FALSE).

The subsequent operating mode changeover depends on ActivateRecoverMode.

Substitute setpoint (S7-1200, S7-1500)

In order to specify a setpoint, PID_Temp offers a substitute setpoint at the SubstituteSetpoint tag in addition to the Setpoint parameter. This can be activated by setting SubstituteSetpointOn = TRUE or by selecting the corresponding check box in the commissioning editor.

The substitute setpoint allows you to specify the setpoint temporarily directly at the slave, for example during commissioning or tuning.

In this case, the interconnection of the output value of the master with the setpoint of the slave that is required for normal operation of the cascade control system does not have to be changed in the program

In order for a master to influence the process or to carry out tuning, the substitute setpoint has to be deactivated at all downstream slaves.

You can monitor the currently effective setpoint as it is used by the PID algorithm for calculation at the CurrentSetpoint tag.

Operating modes and fault response (S7-1200, S7-1500)

The master or slave of a PID_Temp instance does not change the operating mode of this PID_Temp instance.

If a fault occurs at one of its slaves, the master remains in its current operating mode.

If a fault occurs at its master, the slave remains in its current operating mode. However, further operation of the slave then depends on the fault and the configured fault response of the master since the output value of the master is used as the setpoint of the slave:

- If ActivateRecoverMode = TRUE is configured at the master. and the fault does not prevent the calculation of OutputHeat, the fault does not have any effect on the slave.
- If ActivateRecoverMode = TRUE is configured at the master and the fault prevents the calculation of OutputHeat, the master outputs the last output value or the configured substitute output value SubstituteOutput, depending on SetSubstituteOutput. This is then used by the slave as the setpoint.
  PID_Temp is preconfigured so that the substitute output value 0.0 is output in this case (ActivateRecoverMode = TRUE, SetSubstituteOutput = TRUE, SubstituteOutput = 0.0). Configure a suitable substitute output value for your application or activate the use of the last valid PID output value (SetSubstituteOutput = FALSE).
- If ActivateRecoverMode = FALSE is configured at the master, the master changes to the "Inactive" mode when a fault occurs and outputs OutputHeat = 0.0. The slave then uses 0.0 as the setpoint.

The fault response is located in the output settings in the configuration editor.

3.3.3.5 Multi-zone controlling with PID_Temp (S7-1200, S7-1500)

Introduction

In a multi-zone control system, several sections, so-called zones, of a plant are controlled simultaneously to different temperatures. A multi-zone control system is characterized by the mutual influence of the temperature zones through thermal coupling, i.e. the process value of one zone can influence the process value of a different zone through thermal coupling. The strength that this influence has depends on the structure of the plant and the selected operating points of the zones.

Example: Extrusion plant as it is used, for example, in plastics processing.

The substance mixture that passes through the extruder has to be controlled to different temperatures for optimal processing. For example, different temperatures can be required at the filling point of the extruder than at the outlet nozzle. The individual temperature zones mutually influence each other through thermal coupling.

When PID_Temp is used in multi-zone control systems, each temperature zone is controlled by a separate PID_Temp instance.

Observe the following explanations if you want to use the PID_Temp in a multi-zone control system.

Separate pretuning for heating and cooling

Initial commissioning of a plant as a rule begins with the carrying out of pretuning in order to carry out initial setting of the PID parameters and control to the operating point. The pretuning for multi-zone control systems is often carried out simultaneously for all zones.

PID_Temp offers the possibility of carrying out pretuning for heating and cooling in one step (Mode = 1, Heat.EnableTuning = TRUE, Cool.EnableTuning = TRUE) for controllers with activated cooling and PID parameter switching as the method for heating/cooling (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE).

However, it is advisable not to use this tuning for simultaneous pretuning of several PID_Temp instances in a multi-zone control system. Instead, first carry out the pretuning for heating (Mode = 1, Heat.EnableTuning = TRUE, Cool.EnableTuning = FALSE) and the pretuning for cooling (Mode = 1, Heat.EnableTuning = FALSE, Cool.EnableTuning = TRUE) separately.

Pretuning for cooling should not be started until all zones have completed pretuning for heating and have reached their operating points.

This reduces mutual influencing through thermal coupling between the zones during tuning.

Adapting the delay time

If PID_Temp is used in a multi-zone control system with strong thermal couplings between the zones, you should ensure that the adaption of the delay time is deactivated for pretuning with PIDSelfTune.SUT.AdaptDelayTime = 0. Otherwise, the determination of the delay time can be incorrect if the cooling of a zone is prevented by the thermal influence of other zones during the adapting of the delay time (heating is deactivated in this phase).

Temporary deactivation of cooling

PID_Temp offers the possibility of deactivating cooling temporarily in automatic mode for controllers with active cooling (Config.ActivateCooling = TRUE) by setting DisableCooling = TRUE.

This ensures that this controller does not cool in automatic mode during commissioning while the controllers of other zones have not yet completed tuning of heating. The tuning could otherwise be influenced negatively by the thermal coupling between the zones.

Procedure

You can proceed as follows during the commissioning of multi-zone control systems with relevant thermal couplings:

1. Set DisableCooling = TRUE for all controllers with activated cooling.
2. Set PIDSelfTune.SUT.AdaptDelayTime = 0 for all controllers.
3. Specify the desired setpoints (Setpoint parameter) and start pretuning for heating (Mode = 1, Heat.EnableTuning = TRUE, Cool.EnableTuning = FALSE) simultaneously for all controllers.
4. Wait until all the controllers have completed pretuning for heating.
5. Set DisableCooling = FALSE for all controllers with activated cooling.
6. Wait until the process values of all the zones are steady and close to the respective setpoint. If the setpoint cannot be reached permanently for a zone, the heating or cooling actuator is too weak.
7. Start pretuning for cooling (Mode = 1, Heat.EnableTuning = FALSE, Cool.EnableTuning = TRUE) for all controllers with activated cooling.

Note

Limit violation of the process value

If the cooling is deactivated in automatic mode with DisableCooling = TRUE, this can cause the process value to exceed the setpoint and the process value limits while DisableCooling = TRUE. Observe the process values and intervene, if appropriate, if you use DisableCooling.

Note

Multi-zone control systems

For multi-zone control systems, the thermal couplings between the zones can result in increased overshoots, permanent or temporary violation of limits and permanent or temporary control deviations during commissioning or operation. Observe the process values and be ready to intervene. Depending on the system, it can be necessary to deviate from the procedure described above.

Synchronization of several fine tuning processes

If fine tuning is started from automatic mode with PIDSelfTune.TIR.RunIn = FALSE, PID_Temp tries to reach the setpoint with PID controlling and the current PID parameters. The actual tuning does not start until the setpoint is reached. The time required to reach the setpoint can be different for the individual zones of a multi-zone control system.

If you want to carry out fine tuning for several zones simultaneously, PID_Temp offers the possibility to synchronize these by waiting with the further tuning steps after the setpoint has been reached.

Procedure

This ensures that all the controllers have reached their setpoint when the actual tuning steps start. This reduces mutual influencing through thermal coupling between the zones during tuning.

Proceed as follows for controllers for whose zones you want to carry out fine tuning simultaneously:

1. Set PIDSelfTune.TIR.WaitForControlIn = TRUE for all controllers.
   These controllers have to be in automatic mode with PIDSelfTune.TIR.RunIn = FALSE.
2. Specify the desired setpoints (Setpoint parameters) and start fine tuning for all controllers.
3. Wait until PIDSelfTune.TIR.ControlInReady = TRUE at all controllers.
4. Set PIDSelfTune.TIR.FinishControlIn = TRUE for all controllers.

All controllers then start the actual tuning simultaneously.

3.3.3.6 Override control with PID_Temp (S7-1200, S7-1500)

Override control

In case of override control, two or more controllers share one actuator. Only one controller has access to the actuator at any time and influences the process.

A logic operation decides which controller has access to the actuator. This decision is often made based on a comparison of the output values of all controllers, for example, in case of a maximum selection, the controller with the largest output value gets access to the actuator.

The selection based on the output value requires that all controllers operate in automatic mode. The controllers that do not have an effect on the actuator are updated. This is necessary to prevent windup effects and their negative impacts on the control response and the switchover between the controllers.

PID_Temp supports override controls as of version 1.1 by offering a simple process for updating the controllers that are not active: By using the tags OverwriteInitialOutputValue and PIDCtrl.PIDInit, you can pre-assign the integral action of the controller in automatic mode as though the PID algorithm had calculated PidOutputSum = OverwriteInititalOutputValue for the PID output value in the last cycle. To do this, OverwriteInitialOutputValue is interconnected with the PID output value of the controller that currently has access to the actuator. By setting the bit PIDCtrl.PIDInit, you trigger the pre-assignment of the integral action as well as the restart of the controller cycle and the PWM period. The subsequent calculation of the PID output value in the current cycle takes place based on the pre-assigned (and synchronized for all controllers) integral action as well as the proportional action and integral action from the current control deviation. The derivative action is not active during the call with PIDCtrl.PIDInit = TRUE and therefore does not contribute to the output value.

This procedure ensures that the calculation of the current PID output value and thus the decision on which controller is to have access to the actuator is only based on the current process state and the PI parameters. Windup effects for controllers that are not active and thus incorrect decisions of the switchover logic are prevented.

Requirement

- PIDCtrl.PIDInit is only effective if the integral action is activated (tags Retain.CtrlParams.Heat.Ti and Retain.CtrlParams.Cool.Ti > 0.0).
- You must assign PIDCtrl.PIDInit and OverwriteInitialOutputValue in your user program yourself (see example below). PID_Temp does not automatically change these tags.
- PIDCtrl.PIDInit is only effective when PID_Temp is in automatic mode (parameter State = 3).
- If possible, select the sampling time of the PID algorithm (Retain.CtrlParams.Heat.Cycle and Retain.CtrlParams.Cool.Cycle tags) so that it is identical for all controllers, and call all controllers in the same cyclic interrupt OB. In this way, you ensure that the switchover does not take place within a controller cycle or a PWM period.

Note

Constant adaptation of the output value limits

Instead of the active updating of the controllers without access to the actuator described here, this is implemented alternatively by constant adaptation of the output value limits in other controller systems.

This is not possible with PID_Temp, because a change of the output value limits is not supported in automatic mode.

Example: Control of a large boiler

PID_Temp is used for control of a large boiler.

The main goal is to control the temperature Input1. The controller PID_Temp_1 is used for this purpose. In addition, the temperature Input2 is to be kept below a high limit at an additional measuring point with the limiting controller PID_Temp_2.

Both temperatures are influenced by only one heater. The output value of the controller corresponds to the heating power.

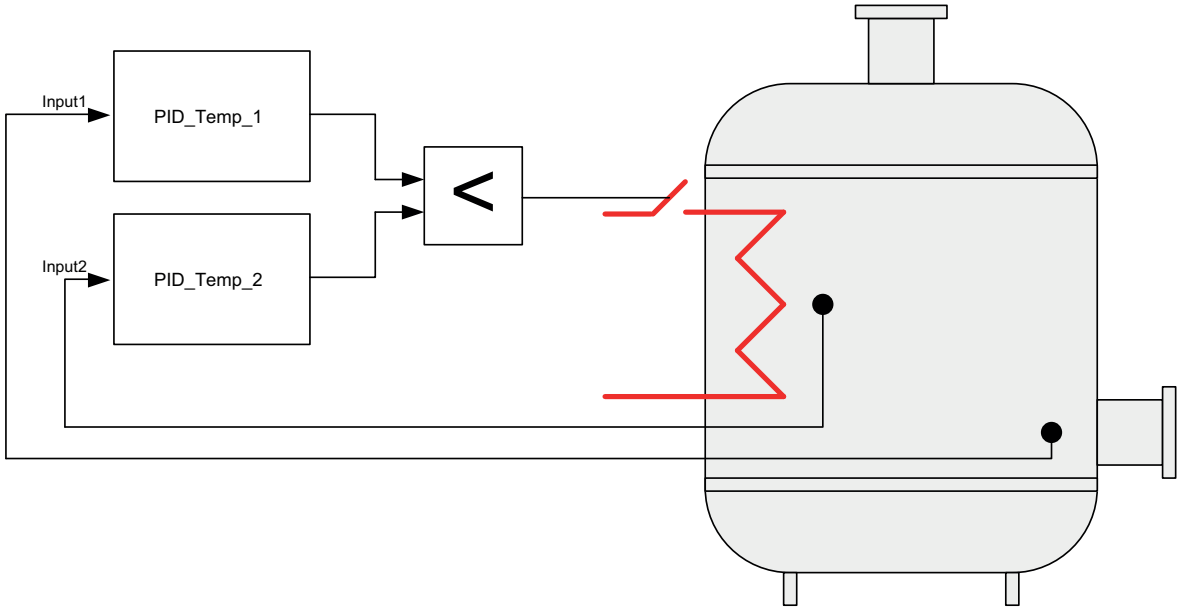

The heater is controlled with the pulse-width modulated output value of PID_Temp (parameter OutputHeat_PWM) by writing the program tag ActuatorInput. The setpoint for the temperature Input1 is specified at the parameter PID_Temp_1.Setpoint. The temperature high limit for the additional measuring point is specified as setpoint at the parameter PID_Temp_2.Setpoint.

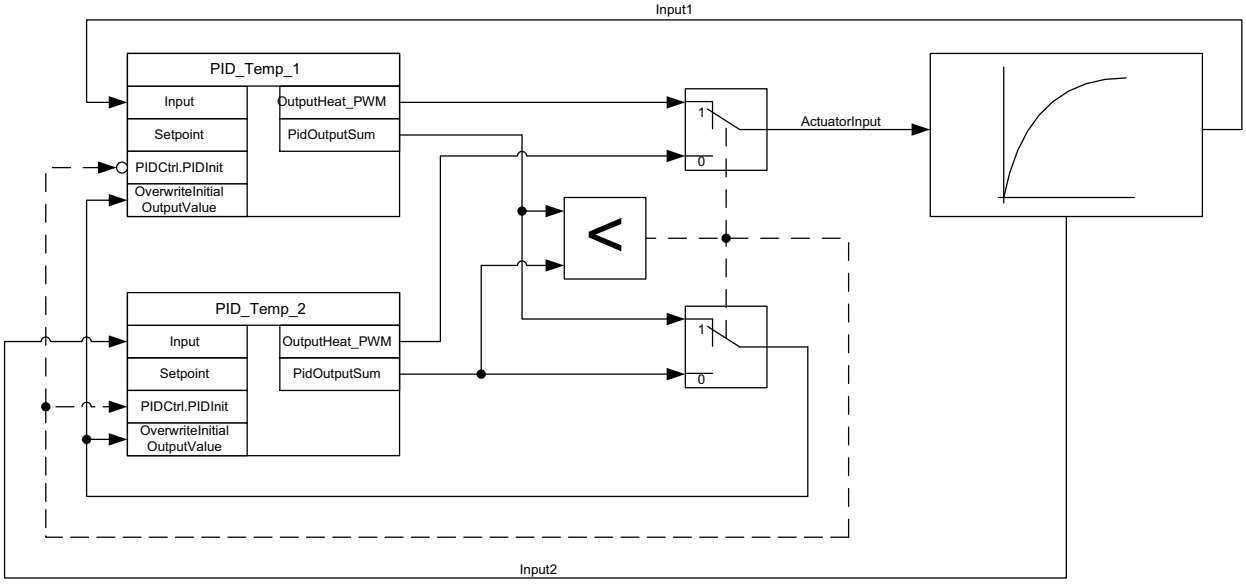

Both controllers must share one heater as shared actuator. The logic that decides which controller gets access to the actuator is implemented by a minimum selection of the PID output value (in Real format, parameter PidOutputSum) in this case. Because the PID output value corresponds to the heating power, the controller that requires lower heating power gets the control.

In normal operation of the plant, the process value of the main controlled variable corresponds to the setpoint. The main controller PID_Temp_1 has settled on a stationary PID output value

PID_Temp_1.PidOutputSum. The process value of the limiting controller Input2 in normal operation is significantly below the high limit that is specified as setpoint for für PID_Temp_2. The limiting controller therefore wants to increase the heating power to increase its process value, which means it will calculate a PID output value PID_Temp_2.PidOutputSum that is greater than the main controller PID_Temp_1.PidOutputSum. The minimum selection of the switchover logic therefore gives the main controller PID_Temp_1 continued access to the actuator. In addition, it is ensured that PID_Temp_2 is updated by means of the assignments PID_Temp_2.OverwriteInitialOutputValue = PID_Temp_1.PidOutputSum and PID_Temp_2.PIDCtrl.PIDInit = TRUE.

If Input2 now approaches the high limit or exceeds it, for example due to a fault, the limiting controller PID_Temp_2 calculates a smaller PID output value to restrict the heating power and thus reduce Input2. If PID_Temp_2.PidOutputSum is smaller than PID_Temp_1.PidOutputSum, the limiting controller PID_Temp_2 receives access to the actuator through the minimum selection and reduces the heating power. It is ensured that PID_Temp_1 is updated by means of the assignments PID_Temp_1.OverwriteInitialOutputValue = PID_Temp_2.PidOutputSum and PID_Temp_1.PIDCtrl.PIDInit = TRUE.

The temperature at the additional measuring point Input2 drops. The temperature of the main controlled variable Input1 drops as well and cannot be held at the setpoint any longer.

Once the fault has been remedied, the Input2 will continue to drop and the heating power is further increased by the limiting controller. As soon as the main controller has calculated a lower heating power as output value, the plant returns to normal operation so that the main controller PID_Temp_1 once again has access to the actuator. This example can be implemented with the following SCL program code:

```
"PID_Temp_1"(Input := "Input1");
"PID_Temp_2"(Input := "Input2");
IF "PID_Temp_1".PidOutputSum <= "PID_Temp_2".PidOutputSum THEN
  "ActuatorInput" := "PID_Temp_1".OutputHeat_PWM;
  "PID_Temp_1".PIDCtrl.PIDInit := FALSE;
  "PID_Temp_2".PIDCtrl.PIDInit := TRUE;
  "PID_Temp_2".OverwriteInitialOutputValue := "PID_Temp_1".PidOutputSum;
ELSE
  "ActuatorInput" := "PID_Temp_2".OutputHeat_PWM;
  "PID_Temp_1".PIDCtrl.PIDInit := TRUE;
  "PID_Temp_2".PIDCtrl.PIDInit := FALSE;
  "PID_Temp_1".OverwriteInitialOutputValue := "PID_Temp_2".PidOutputSum;
END_IF;
```

3.3.3.7 Simulating PID_Temp with PLCSIM (S7-1200, S7-1500)

Note

Simulation with PLCSIM

The simulation of PID_Temp with PLCSIM for CPU S7-1200 is not supported.

PID_TEMP can be simulated only for CPU S7-1500 with PLCSIM.

For the simulation with PLCSIM, the time behavior of the simulated PLC is not exactly identical to that of a "real" PLC. The actual cycle clock of a cyclic interrupt OB can have larger fluctuations with a simulated PLC than with "real" PLCs.

In the standard configuration, PID_Temp determines the time between calls automatically and monitors them for fluctuations.

For the simulation of PID_Temp with PLCSIM, for example, a sampling time error (ErrorBits = DW#16#00000800) can therefore be detected.

This results in ongoing tuning being aborted.

The response in automatic mode depends on the value of the ActivateRecoverMode tag.

To prevent this from happening, you should configure PID_Temp for simulation with PLCSIM as follows:

- CycleTime.EnEstimation = FALSE
- CycleTime.EnMonitoring = FALSE
- CycleTime.Value: Assign the cycle clock of the calling cyclic interrupt OB in seconds to this tag.

3.4 PID control (S7-300, S7-400, S7-1500) (S7-300, S7-400, S7-1500)

3.4.1 Using PID basic functions (S7-300, S7-400, S7-1500)

3.4.1.1 CONT_C (S7-300, S7-400, S7-1500)

Technology object CONT_C (S7-300, S7-400, S7-1500)

The technology object CONT_C provides a continual PID-controller for automatic and manual mode. It corresponds to the instance data block of the instruction CONT_C. You can configure a pulse controller using the PULSEGEN instruction.

The proportional, integral (INT) and differential components (DIF) are switched parallel to each other and can be turned on and off individually. With this, P-, I, PI-, PD- and PID-controller can be set.

S7-1500
All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download CONT_C.

See also

Overview of software controller (Page 631)

Add technology objects (Page 633)

Configure technology objects (Page 634)

Downloading technology objects to device (Page 636)

Configure controller difference CONT_C (S7-300, S7-400, S7-1500)

Use process value periphery

To use the process value in the periphery format at the PV_PER input parameter, follow these steps:

1. Select the "Enable I/O" check box.
2. If the process value is available as a physical size, enter the factor and offset for the scaling in percent.
   The process value is then determined according to the following formula:
   PV = PV_PER × PV_FAC + PV_OFF

Use internal process values

To use the process value in the floating-point format at the PV_IN input parameter, follow these steps:

1. Clear the "Enable I/O" check box.

Control deviation

Set a dead zone range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the output value in this case. The dead zone suppresses the noise component in the steady controller state. The dead zone range specifies the size of the dead zone. With a dead zone range of 0.0, the dead zone is turned off.

Configure the controller algorithm CONT_C (S7-300, S7-400, S7-1500)

General

To determine which components of the control algorithm are activated, proceed as follows:

1. Select an entry from the "Controller structure" list.
   You can only specify required parameters for the selected controller structure.

Proportional action

1. If the controller structure contains a proportional action, enter the "proportional gain".

Integral action

1. If the controller structure contains an integral action, enter the integral action time.
2. To give the integral action an initialization value, select the check box "Initialize integral action" and enter the initialization value.
3. In order to permanently set the integral action to this initialization value, select the "Integral action hold" check box.

Derivative action

1. If the controller structure contains a derivative action, enter the derivative action time, the derivative action weighting and the delay time.

Configure the output value CONT_C (S7-300, S7-400, S7-1500)

General

You can set CONT_C in the manual or automatic mode.

1. To set a manual manipulated value, activate the option "Activate manual mode" option check box.
   You can specify a manual manipulated value on the input parameter MAN.

Manipulated value limits

The manipulated value is limited at the top and bottom so that it can only accept valid values. You cannot turn off the limitation. Exceeding the limits is displayed through the output parameters QLMN_HLM and QLMN_LLM.

1. Enter a value for the high and low manipulated value limits.
   If the manipulated value is a physical size, the units for the high and low manipulated value limits must match.

Scaling

The manipulated value can be scaled for output as a floating point and periphery value through a factor and an offset according to the following formula.

Scaled manipulated value = manipulated value x factor + offset

Default is a factor of 1.0 and an offset of 0.0.

1. Enter a value for the factor and offset.

Programming a pulse controller (S7-300, S7-400, S7-1500)

With the continuous controller CONT_C and the pulse shaper PULSEGEN, you can implement a fixed setpoint controller with a switching output for proportional actuators. The following figure shows the signal flow of the control loop.

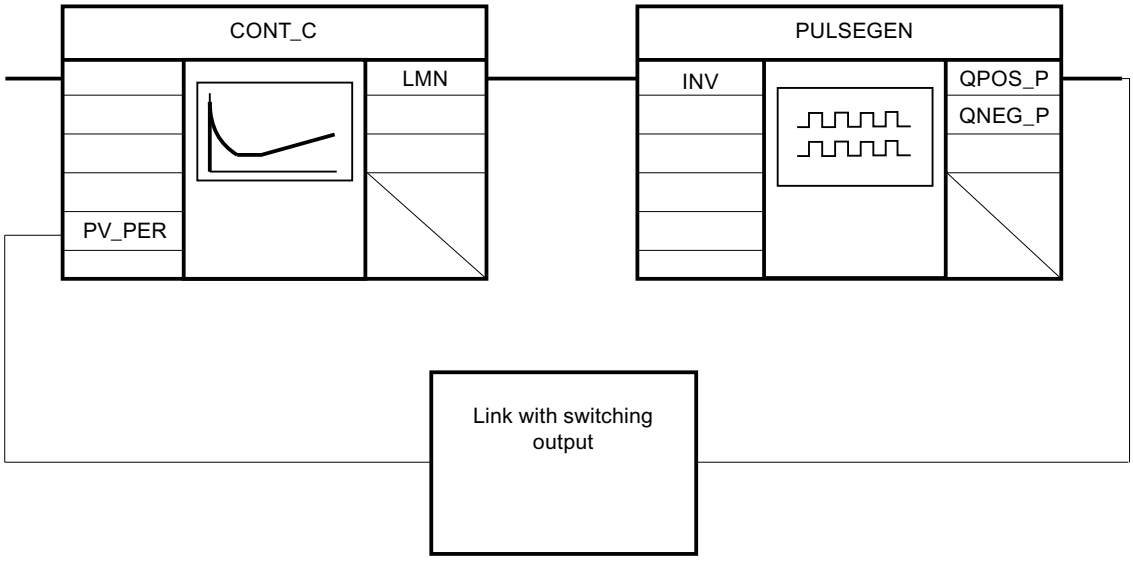

The continuous controller CONT_C forms the output value LMN that is converted by the pulse shaper PULSEGEN into pulse/break signals QPOS_P or QNEG_P.

Commissioning CONT_C (S7-300, S7-400, S7-1500)

Requirements

- The instruction and the technology object are loaded on the CPU.

Procedure

In order to manually determine the optimal PID parameter, proceed as follows:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Enter new PID parameters in the "P", "I", "D" and "Delay time" fields.
3. Click on the icon "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
6. Clear the "Manual mode" check box.
   The controller works with the new PID parameters and controls the new setpoint.
7. Check the quality of the PID parameter to check the curve points.
8. Repeat steps 2 to 6 until you are satisfied with the controller results.

3.4.1.2 CONT_S (S7-300, S7-400, S7-1500)

Technology object CONT_S (S7-300, S7-400, S7-1500)

The technology object CONT_S provides a step controller for actuators with integrating behavior and is used to control technical temperature processes with binary output value output signals. The technology object corresponds to the instance data block of the CONT_S instruction. The operating principle is based on the PI control algorithm of the sampling controller. The step controller operates without a position feedback signal. Both manual and automatic mode are possible.

S7-1500
All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download CONT_S.

See also

Overview of software controller (Page 631)

Add technology objects (Page 633)

Configure technology objects (Page 634)

Downloading technology objects to device (Page 636)

Configure controller difference CONT_S (S7-300, S7-400, S7-1500)

Use process value periphery

To use the process value in the periphery format at the PV_PER input parameter, follow these steps:

1. Select the "Enable I/O" check box.
2. If the process value is available as a physical quantity, enter the factor and offset for the scaling in percent.
   The process value is then determined according to the following formula:
   PV = PV_PER × PV_FAC + PV_OFF

Use internal process values

To use the process value in the floating-point format at the PV_IN input parameter, follow these steps:

1. Clear the "Enable I/O" check box.

Control deviation

Set a deadband range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the manipulated variable in this case. The deadband suppresses the noise component in the steady controller state. The deadband range specifies the size of the deadband. With a deadband range of 0.0, the deadband is turned off.

Configuring control algorithm CONT_S (S7-300, S7-400, S7-1500)

PID algorithm

1. Enter the "proportional amplification" for the P-component.
2. Enter the integration time for the time behavior of the I-component.
   With an integration time of 0.0, the I-component is switched off.

Configure manipulated value CONT_S (S7-300, S7-400, S7-1500)

General

You can set CONT_S in the manual or automatic mode.

1. To set a manual manipulated value, activate the "Activate manual mode" option check box.
   Enter a manual manipulated value for the input parameters LMNUP and LMNDN.

Pulse generator

1. Enter the minimum impulse duration and minimum pause duration.
   The values must be greater than or equal to the cycle time for the input parameter CYCLE.
   The frequency of operation is reduced through this.
2. Enter the motor setting time.
   The value must be greater than or equal to the cycle time of the input parameter CYCLE.

Commissioning CONT_S (S7-300, S7-400, S7-1500)

Requirements

- The instruction and the technology object have been loaded to the CPU.

Procedure

To manually determine the optimal PID parameters, proceed as follows:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. In the fields "P" and "I", enter a new proportional value and a new integration time.
3. Click on the icon "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
6. Clear the "Manual mode" check box.
   The controller works with the new parameters and controls the new setpoint.
7. Check the quality of the PID parameter to check the curve points.
8. Repeat steps 2 to 6 until you are satisfied with the controller results.

3.4.1.3 TCONT_CP (S7-300, S7-400, S7-1500)

Technology object TCONT_CP (S7-300, S7-400, S7-1500)

The technology object TCONT_CP provides a continual temperature controller with pulse generator. It corresponds to the instance data block of the instruction TCONT_CP. The operation is based on the PID control algorithm of the sampling controller. Both manual and automatic mode are possible.

The instruction TCONT_CP calculates the proportional, integral and derivative parameters for your controlled system during pretuning. "Fine tuning" can be used to tune the parameters further. You can also enter the PID parameters manually.

S7-1500
All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download TCONT_CP.

See also

Overview of software controller (Page 631)

Add technology objects (Page 633)

Configure technology objects (Page 634)

Downloading technology objects to device (Page 636)

Configure TCONT_CP (S7-300, S7-400, S7-1500)

Controller difference (S7-300, S7-400, S7-1500)

Use process value periphery

To use the input parameter PV_PER, proceed as follows:

1. Select the entry "Periphery" from the "Source" list.
2. Select the "sensor type".
   Depending on the sensor type, the process value is scaled according to different formulas.
   - Standard
     Thermoelements; PT100/NI100
     PV = 0.1 × PV_PER × PV_FAC + PV_OFFS
   - Cooling;
     PT100/NI100
     PV = 0.01 × PV_PER × PV_FAC + PV_OFFS
   - Current/voltage
     PV = 100/27648 × PV_PER × PV_FAC + PV_OFFS
3. Enter the factor and offset for the scaling of the process value periphery.

Use internal process values

To use the input parameter PV_IN, proceed as follows:

1. Select the entry "Internal" from the "Source" list.

Control deviation

Set a deadband range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the manipulated variable in this case. The deadband suppresses the noise component in the steady controller state. The deadband range specifies the size of the deadband. With a deadband range of 0.0, the deadband is turned off.

Controlling algorithm (S7-300, S7-400, S7-1500)

General

1. Enter the "Sampling time PID algorithm".
   A controller sampling time should not exceed 10 % of the determined integratl action time of the controller (TI).
2. If the controller structure contains a proportional action, enter the "proportional gain".
   A negative proportional gain inverts the rule meaning.

Proportional action

For changes of the setpoint, it may lead to overshooting of the proportional action. Through the weighting of the proportional action, you can select how strongly the proportional action should react when setpoint changes are made. The weakening of the proportional action is reached through a compensation of the integral action.

1. To weaken the proportional action for setpoint changes, enter a "Proportional action weighting".
   - 1.0: Proportional action for setpoint change is fully effective
   - 0.0: Proportional action for setpoint change is not effective

Integral action

With a limitation of the manipulated value, the integral action is stopped. With a control deviation that moves the integral action in the direction of an internal setting range, the integral action is released again.

1. If the controller structure contains an integral action, enter the "integral action time".
   With an integral action time of 0.0, the integral action is switched off.
2. To give the integral action an initialization value, select the "Initialize integral action" check box and enter the "Initialization value".
   Upon restart or COM_RST = TRUE, the integral action is set to this value.

Derivative action

1. If the controller structure contains a derivative action, enter the derivative action time (TD) and the coefficients DT1 (D_F).
   With switched derivative action, the following equation should be maintained:
   TD = 0.5 × CYCLE× D_F.
   The delay time is calculated from this according to the formula:
   delay time = TD/D_F

Set PD-controller with operating point

1. Enter the integral action time 0.0.
2. Activate the "Initialize integral action" check box.
3. Enter the operating point as the initialization value.

Set P-controller with operating point

1. Set a PD-controller with an operating point.
2. Enter the derivative action time 0.0.
   The derivative action is disabled.

Control zone

The control zone limits the value range of the control deviation. If the control deviation is outside of this value range, the manipulated value limits are used.

With an occurrence in the control zone, the derivative action leads to a very quick reduction of the manipulated variable. Thus, the control zone only makes sense for switched on derivative actions. Without control zone, only the reducing proportional action would reduce the manipulated value. The control zone leads to a quick oscillation without over/under shooting if the emitted minimum or maximum manipulated values are removed from the manipulated value required for the new operating point.

1. Activate the "Activate" check box in the "control zone" group.
2. Enter a setpoint value in the "Width" input field from which the process value may deviate above or below.

Manipulated value continual controller (S7-300, S7-400, S7-1500)

Manipulated value limits

The manipulated value is limited at the top and bottom so that it can only accept valid values. You cannot turn off the limitation. Exceeding the limits is displayed through the output parameters QLMN_HLM and QLMN_LLM.

1. Enter a value for the high and low manipulated value limits.

Scaling

The manipulated value can be scaled for output as a floating point and periphery value through a factor and an offset according to the following formula.

Scaled manipulated value = manipulated value x factor + offset

Default is a factor of 1.0 and an offset of 0.0.

1. Enter a value for the factor and offset.

Pulse generator

The pulse generator must be turned on for a continual controller.

1. Disable the "Activate" option check box in the "Pulse generator" group.

Manipulated value pulse controller (S7-300, S7-400, S7-1500)

Pulse generator

The analog manipulated value (LmnN) can be emitted through pulse-duration modulation on the output parameter QPULSE as an impulse sequence.

To use the pulse generator, proceed as follows:

1. Activate the "Activate" option check box in the "pulse generator" group.
2. Enter the "sampling time pulse generator", the "minimum impulse/break duration" and the "period duration".

The following graphics clarify the connection between the "sampling pulse generator" (CYCLE_P), the "minimum impulse/break duration" (P_B_TM) and the "period duration" (PER_TM):

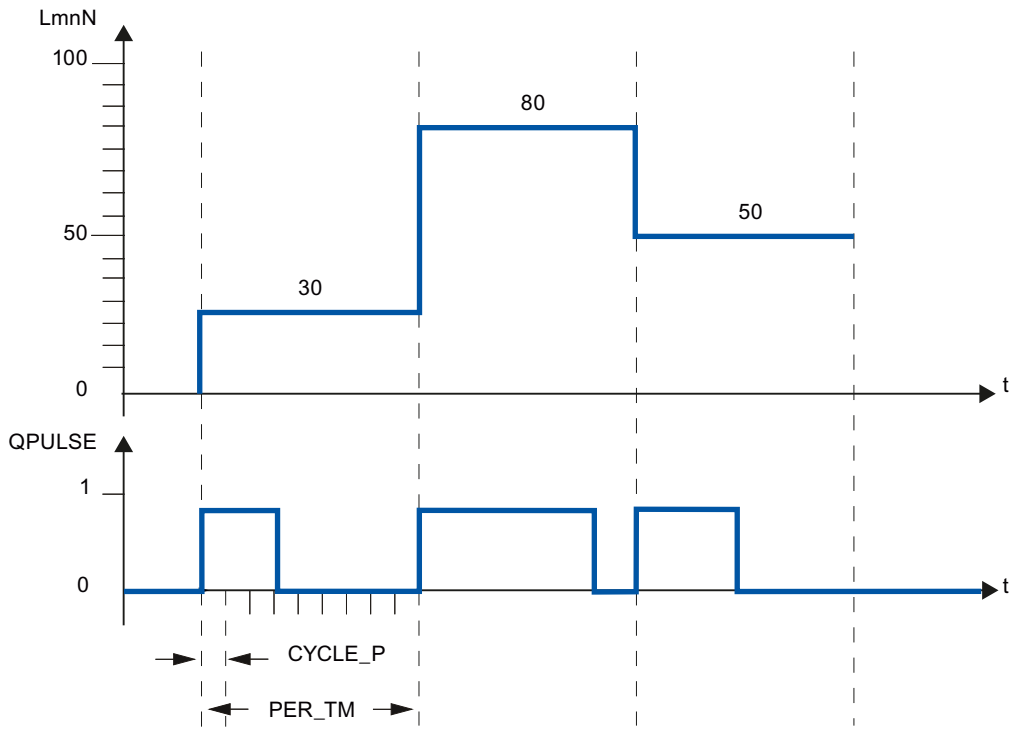

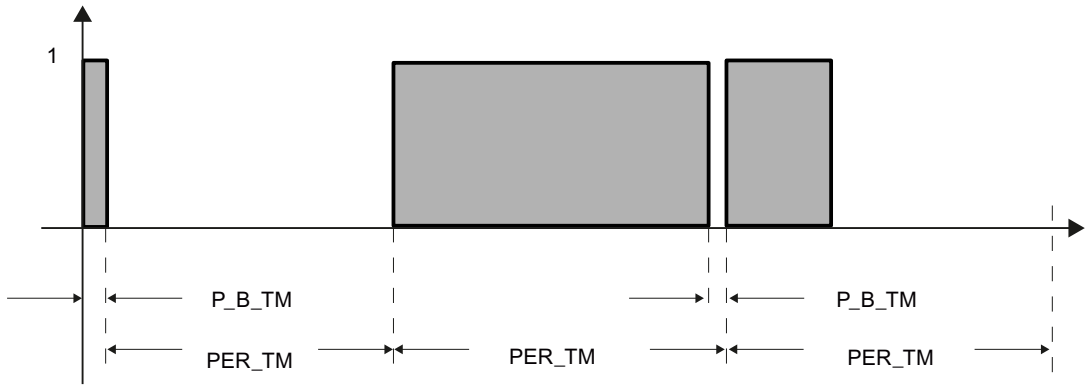

Sampling time pulse generator

The sampling time pulse generator must agree with the time tact of the cyclic interrupt OB being called. The duration of the created impulse is always a whole number factor of this value. For an adequately precise manipulated value resolution, the following relationship should apply:
CYCLE_P ≤ PER_TM/50

Minimum impulse/break duration

Through the minimum impulse/break duration, short on or off times on the actuator are avoided. An impulse smaller than P_B_TM is suppressed.

Recommended are values P_B_TM ≤ 0.1 × PER_TM.

Period duration

The period duration should not exceed 20% of the determined integration time of the controller (TI):
PER_TM ≤ TI/5

Example for the effect of the parameter CYCLE_P, CYCLE and PER_TM:

Period duration PER_TM = 10 s

Sampling time PID-algorithm CYCLE = 1 s

Sampling time pulse generator CYCLE_P = 100 ms.

Every second, a new manipulated value, every 100 ms the comparison of the manipulated value occurs with the previously emitted impulse length and break length.

- If an impulse is emitted, there are 2 possibilities:
  - The calculated manipulated value is larger than the previous impulse length/PER_TM. Then the impulse is extended.
  - The calculated manipulated value is less than or equal to the previous impulse length/ PER_TM. Then no impulse signal will be emitted.
- If no impulse is emitted, there are also 2 possibilities:
  - The value (100 % - calculated manipulated value) is greater than the previous break length / PER_TM. Then the break is extended.
  - The value (100 % - calculated manipulated value) is less than or equal to the previous break length / PER_TM. Then an impulse signal will be emitted.

Commissioning TCONT_CP (S7-300, S7-400, S7-1500)

Optimization of TCONT_CP (S7-300, S7-400, S7-1500)

Application possibilities

The controller optimization for heating or cooling processes from process type I is applicable. But you can use the block for processes with higher levels like process type II or III.

The PI/PID parameters are automatically determined and set. The controller draft is designed for an optimal disruption behavior The "precise" parameters resulting from this lead to overshooting of 10% to 40% of the jump height for setpoint jump heights.

Phases of controller optimization

For the controller optimization, individual phases are run through, which you can read on the parameter PHASE .

PHASE = 0

No tuning is running. TCONT_CP works in automatic or manual mode.

During PHASE = 0, you can make sure that the controlled system fulfills the requirements for an optimization.

At the end of the optimization, TCONT_CP changes back into PHASE = 0.

PHASE = 1

TCONT_CP is prepared for optimization. PHASE = 1 may only be started if the requirements for an optimization are fulfilled.

During PHASE = 1, the following values are determined:

- Process value noise NOISE_PV
- Initial slope PVDT0
- Average of the manipulated variable
- Sampling time PID algorithm CYCLE
- Sampling time pulse generator CYCLE_P

PHASE = 2

In phase 2, the process value attempts to detect the point of inflection with a constant manipulated variable. This method prevents the point of inflection from being found too early as a result of process variable noise.

With the pulse controller, the process variable is averaged over N pulse cycles and then made available to the controller stage. There is a further averaging of the process variable in the controller stage: Initially, this averaging is inactive; in other words, averaging always takes place over 1 cycle. As long as the noise exceeds a certain level, the number of cycles is doubled.

The period and amplitude of the noise are calculated. The search for the point of inflection is canceled and phase 2 is exited only when the gradient is always smaller than the maximum rise during the estimated period. TU and T_P_INF are, however, calculated at the actual point of inflection.

Tuning, however, is only ended when the following two conditions are met:

1. The process value is more than 2*NOISE_PV away from the point of inflection.
2. The process value has exceeded the point of inflection by 20%.

Note

When exciting the process using a setpoint step change, tuning is ended at the latest when the process value exceeds 75% of the setpoint step change (SP_INT-PV0) (see below).

PHASE = 3, 4, 5

The phases 3, 4 and 5 last 1 cycle each.

In Phase 3, the valid PI/PID parameters are saved before the optimization and the process parameter is calculated.

In Phase 4, the new PI/PID parameters are calculated.

In Phase 5, the new manipulated variable is calculated and the controlled system is given.

PHASE = 7

The process type is inspected in Phase 7, because TCONT_CP always changes to automatic mode after optimization. The automatic mode starts with LMN = LMN0 + 0.75*TUN_DLMN as a manipulated variable. The testing of the process type occurs **in the automatic mode** with the recently recalculated controller parameters and ends at the latest 0.35*TA (equilibrium time) after the point of inflection. If the process order deviates strongly from the estimated value, the controller parameters are newly calculated and STATUS_D is counted up by 1, otherwise, the controller parameters remain unchanged.

Then the optimization mode is complete and TCONT_CP is back in PHASE = 0. At the STATUS_H parameter, you can identify whether the tuning was successfully completed.

Premature cancellation of the optimization

In Phase 1, 2 or 3, you can cancel the optimization by resetting TUN_ON = FALSE without calculating new parameters. The controller starts in the automatic mode with LMN = LMN0 + TUN_DLMN. If the controller was in manual mode before the tuning, the old manual manipulated variable is output.

If the tuning is canceled in Phase 4, 5 or 7 with TUN_ON = FALSE, the determined controlled parameters are contained until then.

Requirements for an optimization (S7-300, S7-400, S7-1500)

Transient response

The process must have a stable, asymptotic transient response with time lag.

The process value must settle to steady state after a step change of the manipulated variable. This therefore excludes processes that already show an oscillating response without control, as well as processes with no recovery (integrator in the control system).

WARNING

This may result in death, severe injury or considerable property damage.

During an tuning, the parameter MAN_ON is ineffective. Through this, the output value or process value may take on undesired - even extreme - values.

The output value is defined through the tuning. To cancel the tuning, you first have to set TUN_ON = FALSE. This makes MAN_ON effective again.

Guaranteeing a stationary initial state (phase 0)

With lower-frequency oscillations of the process value, for example, due to incorrect controller parameters, the controller must be put in manual mode before the tuning is started and wait for the oscillation to stop. Alternatively, you could switch to a "soft" set PI controller (small loop gain large integration time).

Now you have to wait until the stationary state is reached, this means, until the process value and output value have a steady state. It is also permissible to have an asymptotic transient oscillation or slow drifting of the process value (stationary state, see the following image). The output value must be constant or fluctuate by a constant average.

Note

Avoid changing the manipulated variable shortly before starting the tuning. A change of the manipulated variable can occur in an unintended manner through the establishment of the test conditions (for example, closing an oven door)! If this does happen, you have to at least wait until the process value has an asymptotic transient oscillation in a stationary state again. Better controller parameters can be reached if you wait until the transient effect has completely subsided.

In the following image, the transient oscillation is illustrated in the stationary state:

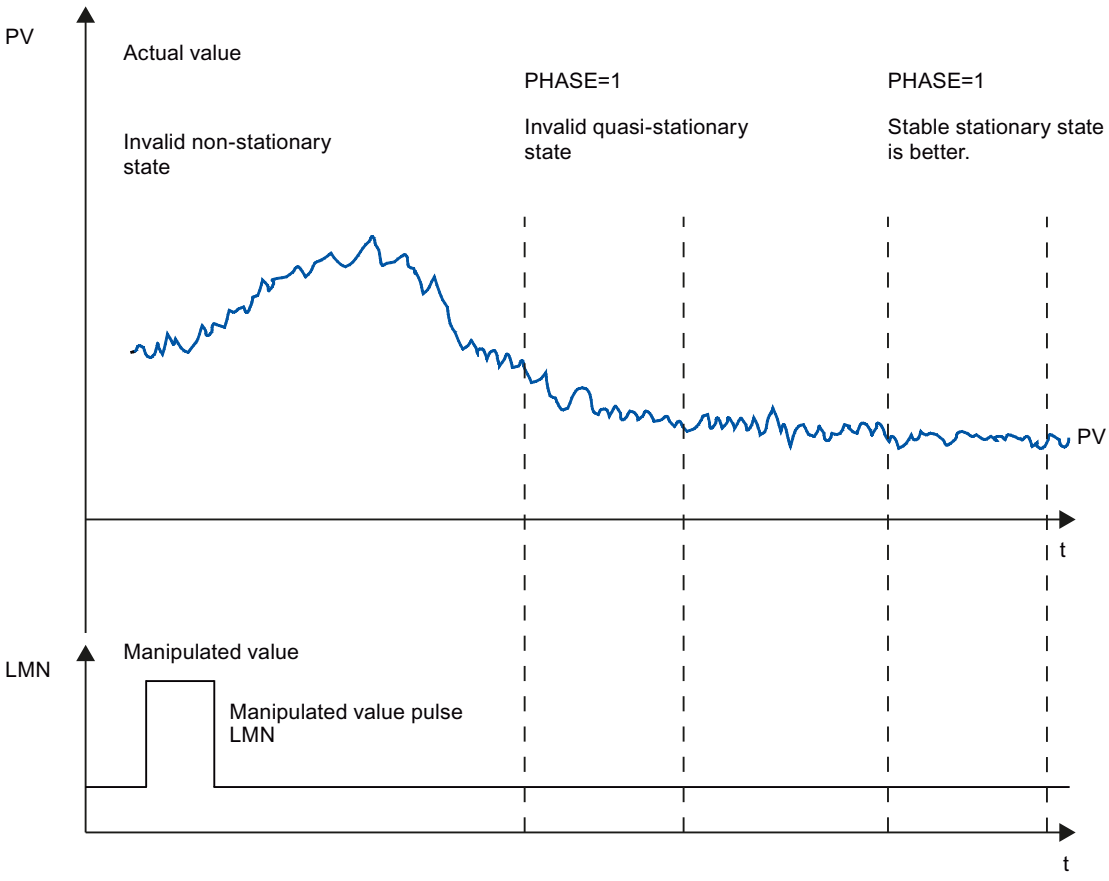

Linearity and operating range

The process response must be linear across the operating range. Non-linear response occurs, for example, when an aggregation state changes. Tuning must take place in a linear part of the operating range.

This means, during tuning and normal control operation non-linear effects within the operating range must be insignificant. It is, however possible to retune the process when the operating point changes, providing tuning is repeated in the close vicinity of the new operating point and non-linearity does not occur during tuning.

If a specific static non-linearity (e.g., valve characteristics) is known, it is always advisable to compensate this with a polyline to linearize the process response.

Disturbance in temperature processes

Disturbances such as the transfer of heat to neighboring zones must not affect the overall temperature process too much. For example, when optimizing the zones of an extruder, all zones must be heated simultaneously.

Possibilities for optimization (S7-300, S7-400, S7-1500)

The following possibilities for tuning exist:

- Pretuning
- Fine tuning
- Manual fine-tuning in control mode

Pretuning

During this tuning, the working point is approached from the cold state through a setpoint jump.

With TUN_ON = TRUE, you can establish the tuning readiness. The controller switches from PHASE = 0 to PHASE = 1.

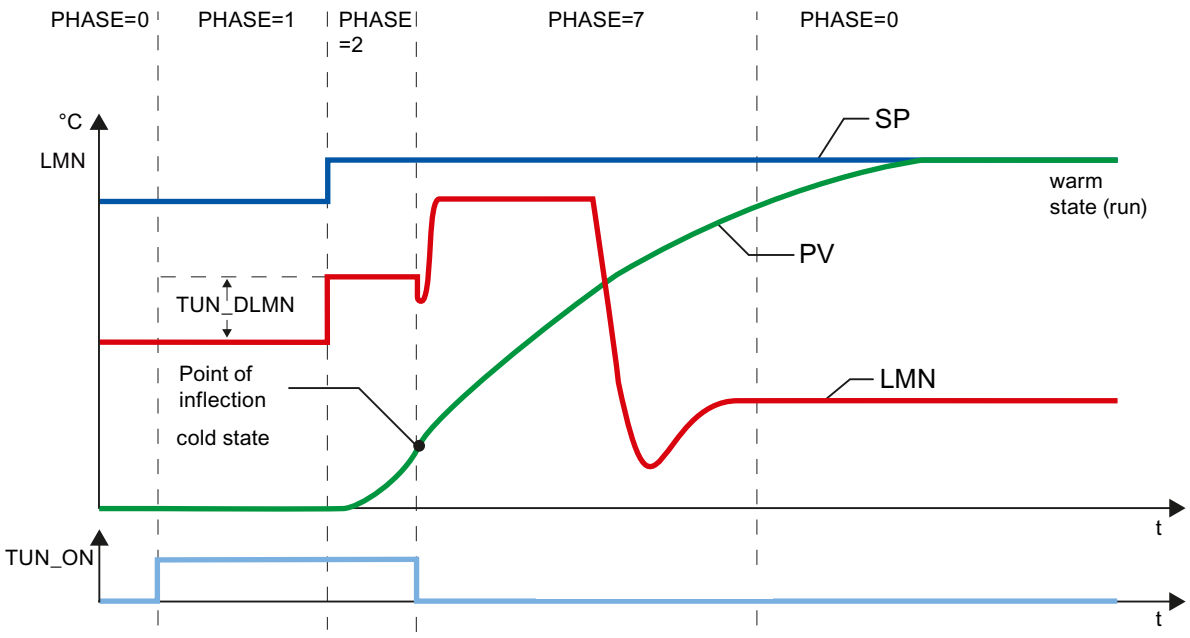

The tuning manipulated variable (LMN0 + TUN_DLMN) is activated by a setpoint change (transition phase 1 -> 2). The setpoint is not effective until the inflection point has been reached (automatic mode is not enabled until this point is reached).

The user is responsible for defining the output excitation delta (TUN_DLMN) according to the permitted process value change. The sign of TUN_DLMN must be set depending on the intended process value change (take into account the direction in which the control is operating).

The setpoint step change and TUN_DLMN must be suitably matched. If the value of TUN_DLMN is too high, there is a risk that the point of inflection will not be found before 75% of the setpoint step change is reached.

TUN_DLMN must nonetheless be high enough to ensure that the process value reaches at least 22 % of the setpoint step change. Otherwise, the process will remain in tuning mode (phase 2).

Remedy: Reduce the setpoint value during the inflection point search.

Note

If processes are extremely sluggish, it is advisable during tuning to specify a target setpoint that is somewhat lower than the desired operating point and to monitor the status bits and PV closely (risk of overshooting).

Tuning only in the linear range:

The signals of certain processes (e.g., zinc or magnesium smelters) will pass a non-linear area at the approach of the operating range (change in the state of aggregation).

By selecting a suitable setpoint step change, tuning can be limited to the linear range. When the process value has passed 75% of the setpoint step change (SP_INT-PV0), tuning is ended.

At the same time, TUN_DLMN should be reduced to the extent that the point of inflection is guaranteed to be found before 75% of the setpoint step change is reached.

Fine tuning

During this tuning, the process with a constant setpoint is activated through a output value jump.

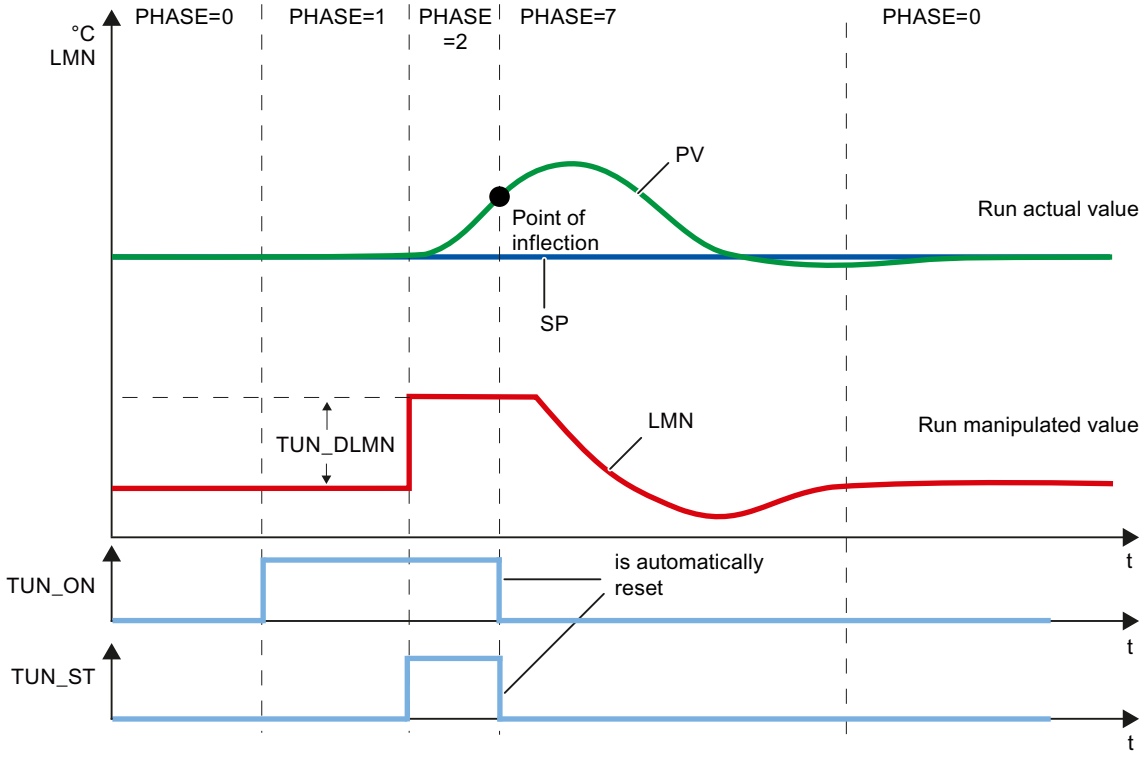

The tuning manipulated variable (LMN0 + TUN_DLMN) is activated by setting the start bit TUN_ST (transition from phase 1 -> 2). When you modify the setpoint value, the new value will not take effect until the point of inflection has been reached (automatic mode will not be enabled until this point has been reached).

The user is responsible for defining the output excitation delta (TUN_DLMN) according to the permitted process value change. The sign of TUN_DLMN must be set depending on the intended process value change (take into account the direction in which the control is operating).

NOTICE
Safety off at 75% is not available when you excite the process via TUN_ST. Tuning is ended when the point of inflection is reached. However, in noisy processes the point of inflection may be significantly exceeded.

Manual fine-tuning in control mode

The following measures can be employed to achieve an overshoot-free setpoint response:

- Adapting the control zone
- Optimize command action
- Attenuation of control parameters
- Modifying control parameters

Tuning result (S7-300, S7-400, S7-1500)

The left cipher of STATUS_H displays the tuning status

STATUS_H	Result
0	Default, i.e., new controller parameters have not (yet) been found.
10000	Suitable control parameters found.
2xxxx	Control parameters have been found via estimated values; check the control response or check the STATUS_H diagnostic message and repeat controller tuning.
3xxxx	An operator error has occurred; check the STATUS_H diagnostic message and repeat controller tuning.

The CYCLE and CYCLE_P sampling times were already checked in phase 1.

The following controller parameters are updated on TCONT_CP:

- P (proportional GAIN)
- I (integration time TI)
- D (derivative time TD)
- Weighting of the proportional action PFAC_SP
- Coefficient DT1 (D_F)
- Control zone on/off CONZ_ON
- Control zone width CON_ZONE

The control zone is only activated if the process type is suitable (process type I and II) and a PID controller is used (CONZ_ON = TRUE).

Depending on PID_ON, control is implemented either with a PI or a PID controller. The old controller parameters are saved and can be retrieved with UNDO_PAR. A PI parameter record and a PID parameter record are saved additionally in the PI_CON and PID_CON structures. Using LOAD_PID and making a suitable setting for PID_ON, it is also possible to switch later between the tuned PI or PID parameters.

Parallel tuning of controller channels (S7-300, S7-400, S7-1500)

Adjacent zones (strong heat coupling)

If two or more controllers are controlling the temperature, on a plate, for example (in other words, there are two heaters and two measured process values with strong heat coupling), proceed as follows:

1. OR the two outputs QTUN_RUN.
2. Interconnect each TUN_KEEP input with the output of the OR element.
3. Start both controllers by specifying a setpoint step change at the same time or by setting TUN_ST at the same time.

The following schematic illustrates the parallel tuning of controller channels.

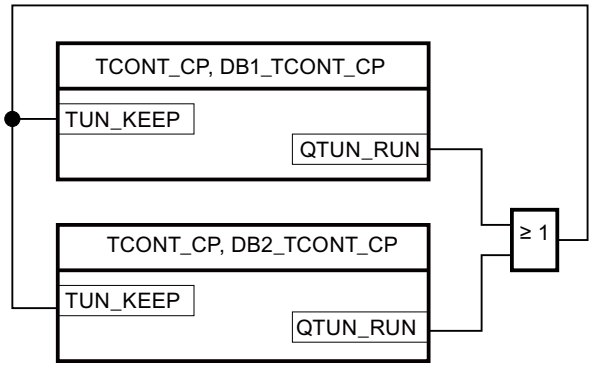

Advantage:

Both controllers output LMN0 + TUN_DLMN until both controllers have left phase 2. This prevents the controller that completes tuning first from falsifying the tuning result of the other controller due to the change in its manipulated variable.

NOTICE
Reaching 75% of the setpoint step change causes an exiting of phase 2 and resetting of output QTUN_RUN. However, automatic mode does not start until TUN_KEEP is also 0.

Adjacent zones (weak heat coupling)

In general terms, tuning should be carried out to reflect the way in which the controller will operate subsequently. If zones are operated together during production such that the temperature differences between the zones remain the same, the temperature of the adjacent zones ought to be increased accordingly during tuning.

Differences in temperature at the beginning of the tuning are irrelevant since they will be compensated by the initial heating (-> initial rise = 0).

Fault descriptions and corrective measures (S7-300, S7-400, S7-1500)

Compensating operator errors

Operator error	STATUS and action	Comment
TUN_ON and setpoint step change or TUN_ST are set simultaneously	Transition to phase 1; however, tuning is not started. • SP_INT = SP_{old} or • TUN_ST = FALSE	The setpoint change is canceled. This prevents the controller from settling to the new setpoint value and from leaving the stationary operating point unnecessarily.
Effective TUN_DLMN < 5% (end of phase 1)	STATUS_H = 30002 • Transition to phase 0 • TUN_ON = FALSE • SP = SP_{old}	Tuning is canceled. The setpoint change is canceled. This prevents the controller from settling to the new setpoint value and from leaving the stationary operating point unnecessarily.

Point of inflection not reached (only if excited by setpoint step change)

At the latest, tuning is ended when the process value has passed 75% of the setpoint step change (SP_INT-PV0). This is signaled by "inflection point not reached" in STATUS_H (2xx2x).

The currently valid setpoint always applies. By reducing the setpoint, it is possible to achieve an earlier end of the tuning function.

In typical temperature processes, cancelation of tuning at 75% of the setpoint step change is normally adequate to prevent overshoot. However, **caution** is advised, particularly in processes with a greater delay (TU/TA > 0.1, process type III). If manipulated variable excitation is too strong compared to the setpoint step change, the process value can overshoot heavily (up to a factor of 3).

In higher-order processes, if the point of inflection is still a long way off after reaching 75% of the setpoint step change, there will be significant overshoot. In addition, the controller parameters are too stringent. In this case, you should reduce the controller parameters or repeat the attempt.

The following schematic illustrates the overshoot of the process variable when the excitation is too strong (process type III):

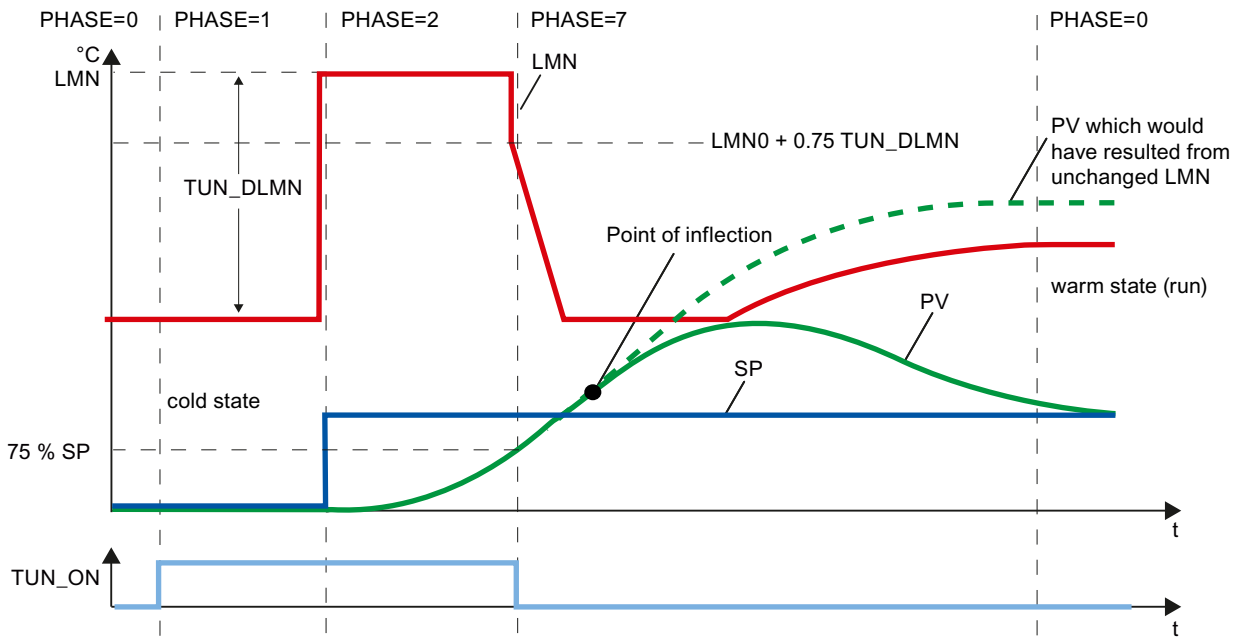

In typical temperature processes, cancelation shortly before reaching the point of inflection is not critical in terms of the controller parameters.

If you repeat the attempt, reduce TUN_DLMN or increase the setpoint step change.

Principle: The value of the manipulated variable used for tuning must be suitable for the setpoint step change.

Error estimating the delay time or order

The delay time (STATUS_H = 2x1xx or 2x3xx) or order (STATUS_H = 21xxx or 22xxx) were not acquired correctly. Operation continues with an estimate that can lead to non-optimum controller parameters.

Repeat the tuning procedure and ensure that disturbances do not occur at the process value.

Note

The special case of a PT1-only process is also indicated by STATUS_H = 2x1xx (TU <= 3*CYCLE). In this case, it is not necessary to repeat the attempt. Reduce the controller parameters if the control oscillates.

Quality of measuring signals (measurement noise, low-frequency interference)

The results of tuning can be distorted by measurement noise or by low-frequency interference. Note the following:

- If you encounter measurement noise, set the sampling frequency higher rather than lower. During one noise period, the process value should be sampled at least twice. In pulse mode, integrated mean value filtering can be helpful. This assumes, however, that the process variable PV is transferred to the instruction in the fast pulse cycle. The degree of noise should not exceed 5% of the useful signal change.
- High-frequency interference cannot be filtered out by TCONT_CP. This should be filtered earlier in the measuring sensor to prevent the aliasing effect.
  The following schematic illustrates the aliasing effect when the sampling time is too long:

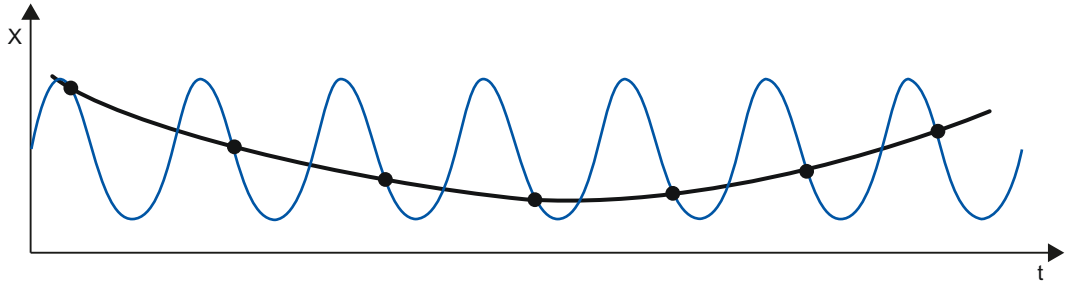

- With low-frequency interference, it is relatively easy to ensure an adequately high sampling rate. However, the TCONT_CP must then generate a uniform measuring signal by having a large interval in the mean value filtering. Mean value filtering must extend over at least two noise periods. Internally in the block, this soon results in higher sampling times such that the accuracy of the tuning is adversely affected. Adequate accuracy is guaranteed with at least 40 noise periods up to the point of inflection.
  Possible remedy when repeating the attempt:
  Increase TUN_DLMN.

Overshoot

Overshoot can occur in the following situations:

Situation	Cause	Remedy
End of tuning	• Excitation by a too high manipulated value change compared with the setpoint step change (see above). • PI controller activated by PID_ON = FALSE.	• Increase the setpoint step change or reduce the manipulated value step change. • If the process permits a PID controller, start tuning with PID_ON = TRUE.
Tuning in phase 7	Initially, less aggressive controller parameters were determined (process type III); these can lead to an overshoot in phase 7.	-
Control mode	PI controller with PFAC_SP = 1.0 for process type I.	If the process permits a PID controller, start tuning with PID_ON = TRUE.

Performing pretuning (S7-300, S7-400, S7-1500)

Requirements

- The instruction and the technology object are loaded on the CPU.

Procedure

To manually determine the optimum PID parameters for initial commissioning, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Select "Pretuning" from the "Mode" drop-down list.
   TCONT_CP is ready for tuning.
3. In the "Output value jump" field, specify how much the output value should be increased.
4. Enter a setpoint in the "Setpoint" field. The output value jump only takes effect when another setpoint is entered.
5. Click the ▶ "Start tuning" icon.
   The pretuning starts. The status of the tuning is displayed.

Performing fine tuning (S7-300, S7-400, S7-1500)

Requirements

- The instruction and the technology object are loaded on the CPU.

Procedure

To determine the optimal PID parameters at the operating point, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Select "Fine tuning" from the "Mode" drop-down list.
   TCONT_CP is ready for tuning.
3. In the "Output value jump" field, specify how much the output value should be increased.
4. Click the ▶ "Start tuning" icon.
   Fine tuning starts. The status of the tuning is displayed.

Cancelling pretuning or fine tuning (S7-300, S7-400, S7-1500)

To cancel pretuning or fine tuning, click on the ■ icon, "Stop tuning".

If the PID parameters have not yet been calculated and stored, TCONT_CP starts in automatic mode LMN = LMN0 + TUN_DLMN. If the controller was in manual mode before the tuning, the old manual manipulated variable is output.

If the calculated PID parameters have already been saved, TCONT_CP starts in automatic mode and works with the previously determined PID parameters.

Manual fine-tuning in control mode (S7-300, S7-400, S7-1500)

The following measures can be employed to achieve an overshoot-free setpoint response:

Adapting the control zone

During tuning, "TCONT_CP" determines a control zone CON_ZONE and activated if the process type is suitable (process type I and II) and a PID controller is used (CONZ_ON = TRUE). In control mode, you can modify the control zone or switch it off completely (with CONZ_ON = FALSE).

Note

Activating the control zone with higher-order processes (process type III) does not normally provide any benefit since the control zone is then larger than the control range that can be achieved with a 100% manipulated variable. There is also no advantage in activating the control zone for PI controllers.

Before you switch on the control zone manually, make sure that the control zone is not too narrow. If the control zone is set too narrow, oscillations occur in the manipulated variable and the process value.

Continuous attenuation of the control response with PFAC_SP

The control response can be attenuated with the PFAC_SP parameter. This parameter specifies the percentage of proportional component that is effective for setpoint step changes.

Regardless of the process type, PFAC_SP is set to a default value of 0.8 by the tuning function; you can later modify this value if required. To limit overshoot during setpoint step changes (with otherwise correct controller parameters) to approximately 2%, the following values are adequate for PFAC_SP:

	Process type I	Process type II	Process type III
	Typical temperature process	Intermediate range	Higher-order temperature process
PI	0.8	0.82	0.8
PID	0.6	0.75	0.96

Adjust the default factor (0.8) in the following situations, in particular:

- Process type I with PID (0.8 →0.6): Setpoint step changes within the control zone still lead to approximately 18% overshoot with PFAC_SP = 0.8.
- Process type III with PID (0.8 →0.96): Setpoint step changes with PFAC_SP = 0.8 are attenuated too strongly. This leads to a significantly slower response time.

Attenuation of control parameters

When a closed-loop control circuit oscillates or if overshoot occurs after setpoint step changes, you can reduce the controller GAIN (e.g., to 80% of the original value) and increase integral time TI (e.g., to 150% of the original value). If the analog output value of the continuous controller is converted to binary actuating signals by a pulse shaper, quantization noise may cause minor permanent oscillation. You can eliminate this by increasing the controller deadband DEADB_W.

Modifying control parameters

Proceed as follows to modify control parameters:

1. Save the current parameters with SAVE_PAR.
2. Modify the parameters.
3. Test the control response.

If the new parameter settings are worse than the old ones, retrieve the old parameters with UNDO_PAR.

Performing fine tuning manually (S7-300, S7-400, S7-1500)

Requirements

- The instruction and the technology object have been loaded to the CPU.

Procedure

To manually determine the optimal PID parameters, proceed as follows:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Select "Manual" from the "Mode" drop-down list.
3. Enter the new PID parameters.
4. Click on the icon "Send parameter to CPU" in the "Tuning" group.
5. Select the "Change setpoint" check box in the "Current values" group.
6. Enter a new setpoint and click in the "Current Values" group on the icon .
7. Clear the "Manual mode" check box.
   The controller works with the new PID parameters and controls the new setpoint.
8. Check the quality of the PID parameter to check the curve points.
9. Repeat steps 3 to 8 until you are satisfied with the controller results.

3.4.1.4 TCONT_S (S7-300, S7-400, S7-1500)

Technology object TCONT_S (S7-300, S7-400, S7-1500)

The technology object TCONT_S provides a step controller for actuators with integrating behavior and is used to control technical temperature processes with binary output value output signals. The technology object corresponds to the instance data block of the TCONT_S instruction. The operating principle is based on the PI control algorithm of the sampling controller. The step controller operates without a position feedback signal. Both manual and automatic mode are possible.

S7-1500
All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download TCONT_S.

See also

Overview of software controller (Page 631)

Add technology objects (Page 633)

Configure technology objects (Page 634)

Downloading technology objects to device (Page 636)

Configure controller difference TCONT_S (S7-300, S7-400, S7-1500)

Use process value periphery

To use the input parameter PV_PER, proceed as follows:

1. Select the entry "Periphery" from the "Source" list.
2. Select the "sensor type".
   Depending on the sensor type, the process value is scaled according to different formulas.
   - Standard
     Thermoelements; PT100/NI100
     PV = 0.1 × PV_PER × PV_FAC + PV_OFFS
   - Cooling;
     PT100/NI100
     PV = 0.01 × PV_PER × PV_FAC + PV_OFFS
   - Current/voltage
     PV = 100/27648 × PV_PER × PV_FAC + PV_OFFS
3. Enter the factor and offset for the scaling of the process value periphery.

Use internal process values

To use the input parameter PV_IN, proceed as follows:

1. Select the entry "Internal" from the "Source" list.

Control deviation

Set a dead zone range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the output value in this case. The dead zone suppresses the noise component in the steady controller state. The dead zone range specifies the size of the dead zone. With a dead zone range of 0.0, the dead zone is turned off.

Configure controller algorithm TCONT_S (S7-300, S7-400, S7-1500)

General

1. Enter the "Sampling time PID algorithm".
   A controller sampling time should not exceed 10 % of the determined integral action time of the controller (TI).
2. If the controller structure contains a proportional action, enter the "proportional gain".
   A negative proportional gain inverts the rule meaning.

Proportional action

For changes of the setpoint, it may lead to overshooting of the proportional action. Through the weighting of the proportional action, you can select how strongly the proportional action should react when setpoint changes are made. The weakening of the proportional action is reached through a compensation of the integral action.

1. To weaken the proportional action for setpoint changes, enter a "Proportional action weighting".
   - 1.0: Proportional action for setpoint change is fully effective
   - 0.0: Proportional action for setpoint change is not effective

Integral action

1. If the controller structure contains an integral action, enter the "integral action time".
   With an integral action time of 0.0, the integral action is switched off.

Configure manipulated value TCONT_S (S7-300, S7-400, S7-1500)

Pulse generator

1. Enter the minimum impulse duration and minimum pause duration.
   The values must be greater than or equal to the cycle time for the input parameter CYCLE.
   The frequency of operation is reduced through this.
2. Enter the motor setting time.
   The value must be greater than or equal to the cycle time of the input parameter CYCLE.

Commissioning TCONT_S (S7-300, S7-400, S7-1500)

Requirements

- The instruction and the technology object have been loaded to the CPU.

Procedure

To manually determine the optimal PID parameters, proceed as follows:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Enter new PID parameters in the "P", "I" and weighting proportional action fields.
3. Click on the icon "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
6. Clear the "Manual mode" check box.
   The controller works with the new parameters and controls the new setpoint.
7. Check the quality of the PID parameter to check the curve points.
8. Repeat steps 2 to 6 until you are satisfied with the controller results.

3.5 PID control (S7-300, S7-400) (S7-300, S7-400)

3.5.1 Using standard PID control (S7-300, S7-400)

3.5.1.1 Using PID_CP (S7-300, S7-400)

Technology object PID_CP (S7-300, S7-400)

The technology object PID_CP provides a continuous PID controller for automatic and manual mode. It corresponds to the instance data block of the instruction PID_CP. You can configure the following controllers:

- Fixed setpoint control with continuous P, PI, PD, PID controllers
- Fixed setpoint control with disturbance variable selection
- Cascade control
- Two-loop ratio control
- Blending control

You configure the following functions of the technology object PID_CP:

- Scaling the process value signal
- Suppression of high-frequency vibrations in the process value branch by smoothing (time delay) of the process value signal.
- Linearization of the quadratic functions of the process value (flow-rate control with differential pressure encoders)
- Calling of "own functions" in the process value branch
- Monitoring the process value for violation of alarm and warning limits
- Monitoring the rate of change of the process value
- Adjustment of the setpoint by ramp/soak
- Scaling an external setpoint signal with ratio controllers
- Calling of "own functions" in the setpoint branch
- Limiting the rate of change of the setpoint
- Limiting the absolute values of the setpoint
- Filtering the control deviation signal with a deadband
- Monitoring the control deviation for violation of alarm and warning limits
- The proportional, integral, and derivative actions can be activated separately.
- Option of proportional and derivative effect in the controller feedback path.
- Selecting a disturbance variable
- Control of the manipulated variable in manual mode from a PG or HMI)

- Calling "own functions" in the manipulated value branch
- Limiting the rate of change of the manipulated variable
- Limiting the absolute values of the manipulated variable
- Scaling the manipulated variable
- Pulse width modulation for pulse generator

Control deviation generation (S7-300, S7-400)

Process value (S7-300, S7-400)

Scaling (PV_NORM) (S7-300, S7-400)

The "I/O process value ON" check box is used to specify which analog input variables are scaled to the physical unit of the process value.

- Activated: Input variable is the I/O process value (PV_PER)
- Disabled: Input variable is the internal process value (PV_IN)

In order to uniquely specify the scaling levels, you define the following four parameters:

- High input (NM_PIHR)
- Low input (NM_PILR)
- High output (NM_PVHR)
- Low output (NM_PVLR)

The parameters are preset depending on the sensor type.

The process value is calculated from the respective input value (PV_PER or PV_IN) according to the following formula:

MP4 = (PV_PER - NM_PILR) × (NM_PVHR - NM_PVLR) / (NM_PIHR - NM_PILR) + NM_PVLR

or

MP4 = (PV_IN - NM_PILR) × (NM_PVHR - NM_PVLR) / (NM_PIHR - NM_PILR) + NM_PVLR

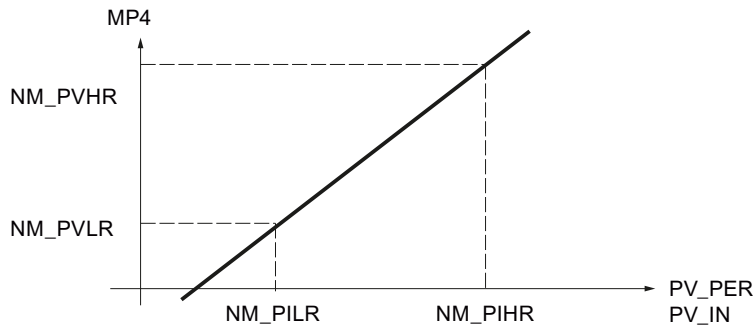

The process value is output in the static variable MP4.

No values are limited and the parameters are not checked. If you enter the same value for input high and input low, a division by zero may occur based on the formulas above. The instruction will not detect this.

Filter functions (S7-300, S7-400)

Smooth controlled variable (LAG1ST) (S7-300, S7-400)

In order to smooth process signals with superimposed rapid variations (noise), a first-order delay element is applied in the process value branch. This function stabilizes the analog controlled variable to a greater or lesser degree, depending on the time constant (PV_TMLAG). In this way, disturbances are effectively suppressed. As a result of this, however, the time constant of the overall control loop also increases, i.e., the closed-loop control becomes more sluggish.

The stabilizing effect is achieved by a first-order delay algorithm.

The step response within the time range is:

$outv(t) = MP4(0)\ (1 - e^{-t/PV_TMLAG})$

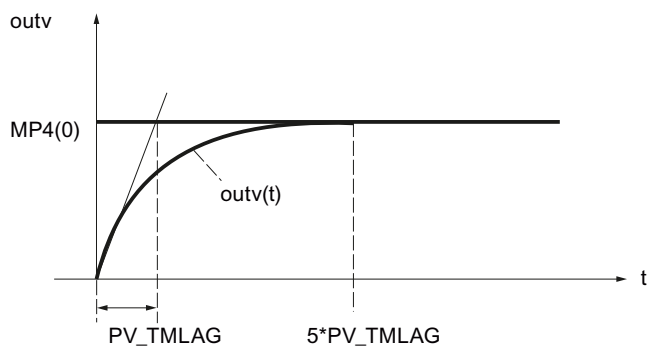

MP4(0)	The height of the process value step at the input
outv(t)	The output variable
PV_TMLAG	The time lag constant
t	Time

Smoothing of the process value

If smoothing of the process value (LAG1STON = FALSE) is not activated, then the I/O input (PV_PER) or the internal input (PV_IN) is switched through to the process value branch without a delay.

Time lag

The PV_TMLAG input parameter specifies the time lag for the process value. Any value within the value range is permitted.

If PV_TMLAG ≤ 0.5 * CYCLE, there is no longer any lag in effect.

A sampling time (CYCLE) of less than one-fifth of the time lag is required in order to achieve a lag response approaching that of the analog response.

Square root (SQRT) (S7-300, S7-400)

The "Square root" function (SQRT) is used to linearize the controlled variable. If the process value supplied by an encoder is a physical quantity and there is a quadratic relation between this quantity and the measured process variable, the characteristic of the controlled variable must be linearized prior to any further processing in the controller.

The measured signal must always be linearized by the square-root function if flow measurements are made using metering orifices, venturi tubes, or the like. The measured pressure differential is then proportional to the square of the flow.

The SQRT_ON = TRUE input signal is used to activate the square root function in the process value branch. The square root function algorithm has the following form:

outv = SQRT (MP5) × (SQRT_HR – SQRT_LR) / 10.0 + SQRT_LR

Here, outv is the scaled output value of the square root function (physical flow).

This formula stipulates that the input value of the square root is scaled to the number range 0 to 100. The parameters for the "Measuring range high" and "Measuring range low" must be set to 100.0 and 0.0 in the scaling of the process value. The square root of this value yields a measuring range from 0 to 10. This range of numbers will be scaled using the high limit (SQRT_HR) and the low limit of the physical measurement range (SQRT_LR).

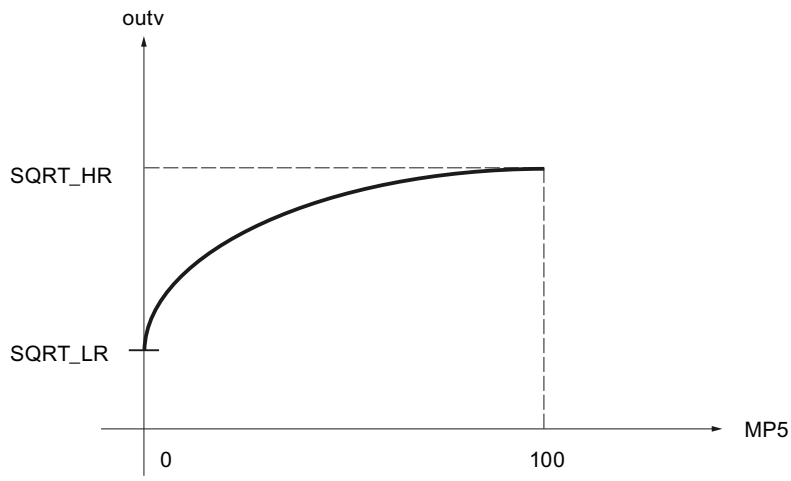

Example of scaling

The input value PV_IN of the controller is the differential pressure in mbar:

Start of measuring range NM_PILR	End of measuring range NM_PIHR	Sample value for PV_IN
20.0 mbar	200.0 mbar	150.0 mbar

The scaled differential pressure is calculated with the help of the scaling function PV_NORM, whereby NM_PVHR = 100.0 and NM_PVLR = 0.0 are selected:

MP4 = (PV_IN - NM_PILR) * (NM_PVHR - NM_PVLR) /
(NM_PIHR - NM_PILR) + NM_PVLR
= (PV_IN - 20.0 mbar) * (100.0 - 0.0) /
(200.0 mbar - 20.0 mbar) + 0.0
= (PV_IN - 20.0 mbar) * 100 / 180.0 mbar

Initial value of MP4	End value of MP4	Sample value for MP4
0.0	100.0	72.222

Since no smoothing is used in this example, the following applies: MP5 = MP4.

For the square root from the scaled differential pressure MP5, the result is:

Initial value according to the square root	End value according to the square root	Sample value for SQRT(MP5)
0.0	10.0	8.498

For the scaled output value outv of the square root function (physical flow-rate) with SQRT_HR = 20000.0 m^3/h und SQRT_LR = 0.0 m3/h, there follows:

outv = SQRT(MP5) * (SQRT_HR - SQRT_LR) / 10.0 + SQRT_LR
= SQRT(MP5) * (20000.0 m^3/h - 0.0 m3/h) / 10.0 + 0.0 m3/h
= 2000.0 m^3/h * SQRT(MP5)

Start of measuring range outv	End of measuring range outv	Sample value for outv
0.0 m^3/h	20000.0 m^3/h	16996.732 m^3/h

FC call in the process value branch (PVFC) (S7-300, S7-400)

Insertion of a user-specific function in the process value branch allows the controlled variable to undergo signal processing, e.g., signal delay or linearization, prior to any further processing in the controller.

Calling a function

If you want to insert a user-specific function into the process value branch, enable "Enable FC call" (PVFC_ON). The number of the utilized function is entered in the "FC number" input field (PVFC_NBR). The controller calls the function. In so doing, existing input/output parameters of the user function are not supplied. You have to program the data transfer in your function yourself.

You can monitor the output value of the function at the static variable MP6.

Note

No check is made to determine whether a function exists. If the function does not exist, the CPU switches to STOP with an internal system error.

Alarm (S7-300, S7-400)

Monitor process value for limit values (PV_ALARM) (S7-300, S7-400)

If the values of the process variables (e.g., speed, pressure, filling level, temperature, etc.) exceed or fall below certain values in closed-loop controls, an illegal process status or plant status can occur. In such cases, monitoring of the process value for limit values (PV_ALARM) is used in order to monitor the process value for violation of the permissible operating range. Limit violations are detected and signaled so that an appropriate reaction can be initiated.

Note

The monitoring of the process value for limit values cannot be deactivated. The violation of the configured limits is always signaled.

Monitoring of the process value for limit values (PV_ALARM) monitors the controlled variable PV(t) for four assignable limits in two tolerance bands. If the limits are reached or exceeded, the function signals a “warning” at the first limit and an “alarm” at the second limit.

Limit values

The numeric values of the limits are set in the input parameters for “warning” and “alarm”. If the process value (PV) exceeds or falls below these limits, the associated output message bits are set for the high alarm (QPVH_ALM), the high warning (QPVH_WRN), the low warning (QPVL_WRN), and the low alarm (QPVL_ALM).

During a restart, the message bits are reset.

The message bits are set as follows:

QPVH_ALM	QPVH_WRN	QPVL_WRN	QPVL_ALM	If:	And:
TRUE	TRUE	FALSE	FALSE	PV ↗ PV ↘	PV ≥ PVH_ALM PV ≥ PVH_ALM - PV_HYS
FALSE	TRUE	FALSE	FALSE	PV ↗ PV ↘	PV ≥ PVH_WRN PV ≥ PVH_WRN - PV_HYS
FALSE	FALSE	TRUE	FALSE	PV ↘ PV ↗	PV ≤ PVL_WRN PV ≤ PVL_WRN + PV_HYS
FALSE	FALSE	TRUE	TRUE	PV ↘ PV ↗	PV ≤ PVL_ALM PV ≤ PVL_ALM + PV_HYS

The following has to apply so that the block operates properly:

High alarm < High warning < Low warning < Low alarm

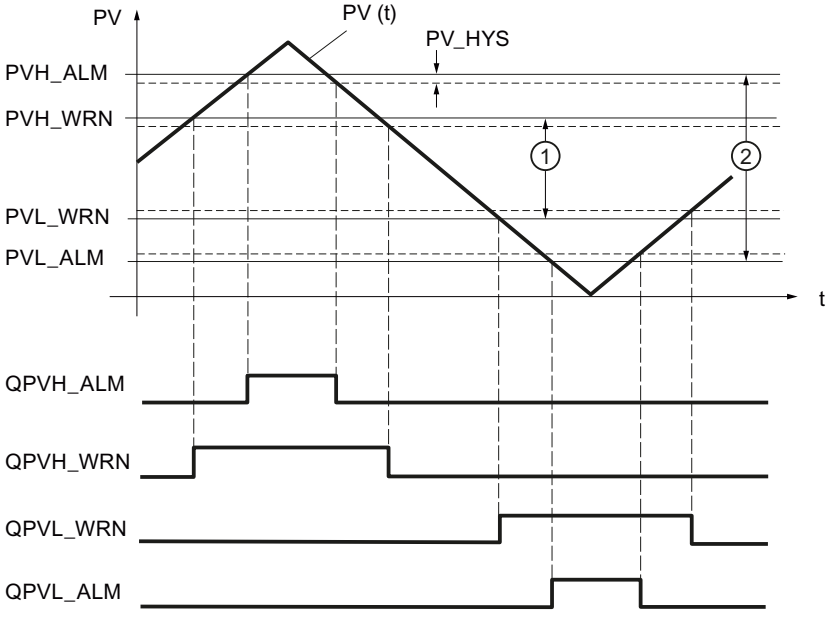

①	1. Tolerance band
②	2. Tolerance band

Hysteresis

In order to prevent "flickering" of the message bits when the input variable undergoes slight changes or in the event of rounding errors, a hysteresis PV_HYS is used. The controlled variable must violate the hysteresis before the message bits are reset.

Monitor rate of change of the process value (ROCALARM) (S7-300, S7-400)

If the change in the process variable (e.g., speed, pressure, filling level, temperature, etc.) is too great in closed-loop controls, an illegal process status or plant status can occur. In such cases, monitoring the rate of change of the process value (ROCALARM) is used to monitor the process value for violations of a permissible rate of change. Limit violations are detected and signaled so that an appropriate reaction can be initiated.

Note

Monitoring of the rate of change of the process value (ROCALARM) cannot be deactivated. The violation of the configured limits is always signaled.

The ROCALARM function monitors the controlled variable PV(t) for permissible rate-of-change limits that are assignable according to sign.

The numeric values of the limit slopes are set at the input parameters for "Maximum increase" and "Maximum decrease" in the positive and negative range of the controlled variable. The slopes refer to a rising slope or falling slope in percent per second. If the rate of change of the controlled variable violates these limits, the associated output message bit QPVURLMP ... QPVDRLMN is set.

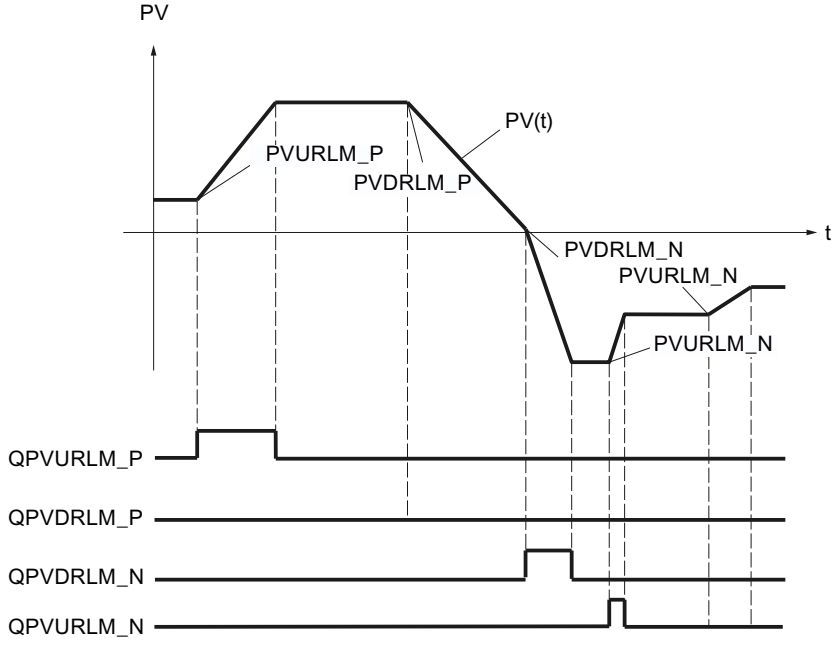

The ramp parameters are identified in accordance with the following scheme:

Parameter	PV change
Maximum increase in the positive range (PVURLM_P)	PV > 0 and |PV| rising
Maximum decrease in the positive range (PVDRLM_P)	PV > 0 and |PV| falling
Maximum increase in the negative range (PVURLM_N)	PV > 0 and |PV| rising
Maximum decrease in the negative range (PVDRLM_N)	PV < 0 and |PV| falling

Setpoint (S7-300, S7-400)

Specifying a setpoint (S7-300, S7-400)

Select a source for the setpoint. The following options are available:

- Internal setpoint
  The input parameter SP_INT is used as setpoint
- Setpoint generator
  The input parameter SP_INT is used as setpoint and can be incremented or decremented using the static variables SPUP and SPDN.
- Ramp/soak
  The input parameter SP_INT is used as setpoint and can be changed according to a specified ramp/soak.
- External setpoint
  The input parameter SP_EXT is used as setpoint for blending, ratio and cascade controllers. The external setpoint can be scaled.

The effective setpoint is shown at the output parameter SP.

Setpoint generator (SP_GEN) (S7-300, S7-400)

The internal setpoint SP_INT can be incremented or decremented in steps using the SPUP and SPDN switches. The modified value can be monitored at MP1. The configured setpoint limits are taken into account.

Effect of the switches SPUP and SPDN

For changes involving small steps, the controller should have a maximum sampling time of 100 ms.

The rate of change of the output variable depends on how long the SPUP or SPDN switch is actuated and on the selected limits. The following relationship applies:

- During the first 3 s after setting SPUP or SPDN:
  doutv/dt = (SP_HLM - SP_LLM) / 100 s
- Afterwards:
  doutv/dt = (SP_HLM - SP_LLM) / 10 s

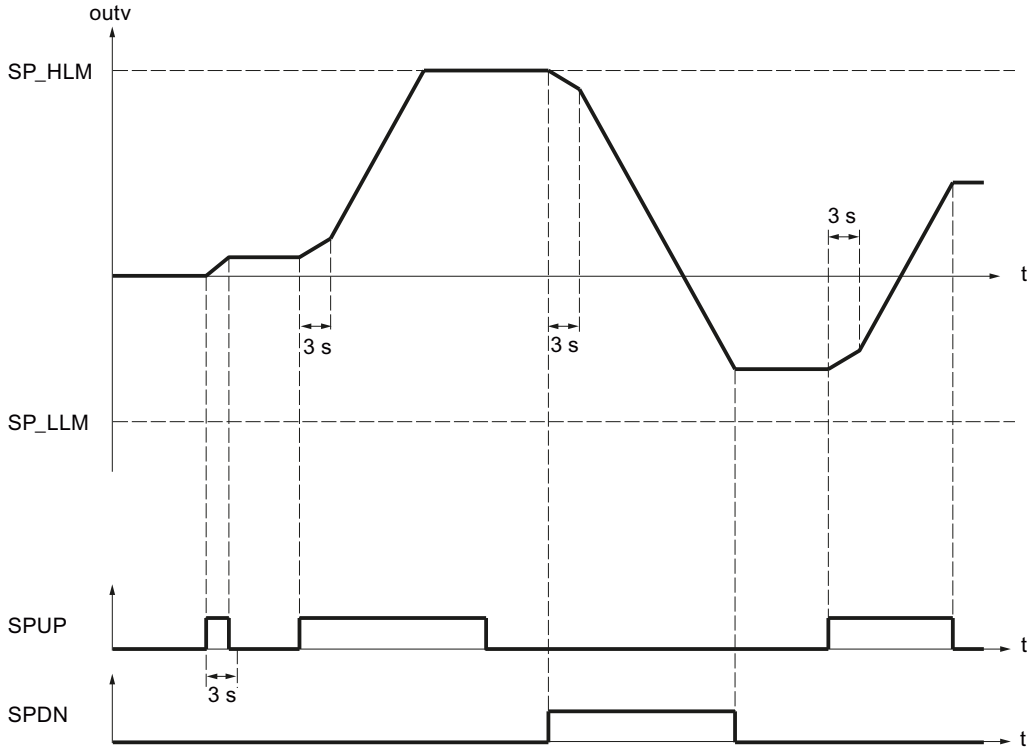

For a sampling time of 100 ms and a setpoint range of -100.0 to 100.0, the setpoint, for example, changes by 0.2 per cycle during the first 3 seconds. If SPUP is TRUE for more than 3 seconds, the rate of changes increases to ten times the value, in this case therefore to 2 per cycle.

Startup and operating principles of the setpoint generator

- During a complete restart, output outv is reset to 0.0.
- If you activate the setpoint generator (SPGEN_ON = TRUE), the signal value SPFC_IN is output initially at output outv. Therefore, the switchover to the setpoint generator from another operating mode is always smooth. Provided the SPUP and SPDN (up/down keys) are not activated, the signal value SPFC_IN is retained at the output.

Parameters of the SP_GEN function

Output parameter outv is an implicit parameter. It can be monitored with the configuration tool via measuring point MP1.

Parameter	Meaning	Permitted value range
SPFC_IN	FC input setpoint	Engineering value range
SP_INT	Internal setpoint	Engineering value range

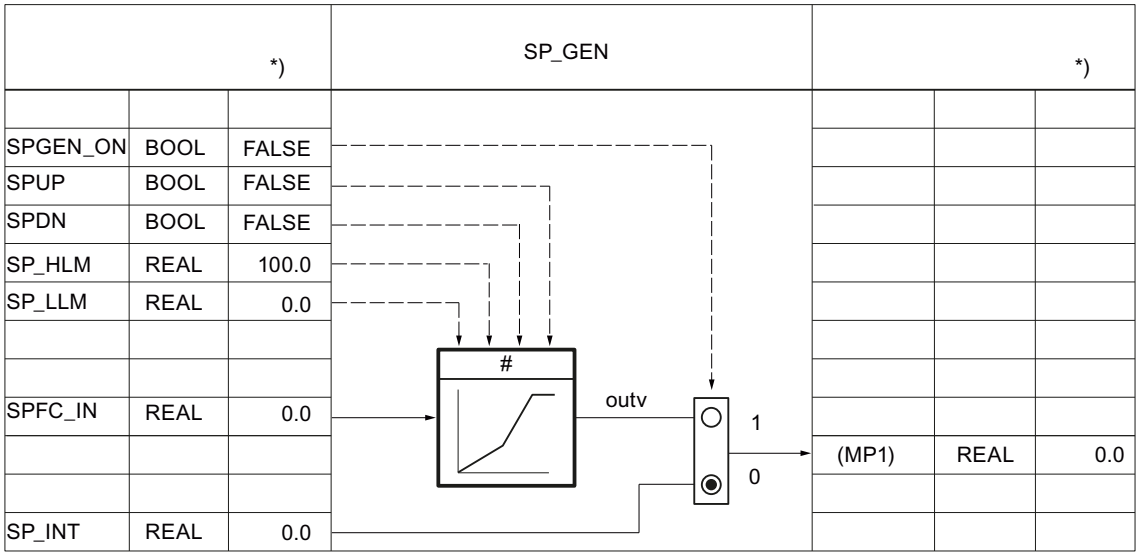

Figure 3-1 Function scheme and parameters of the setpoint generator

Ramp/soak (RMP_SOAK) (S7-300, S7-400)

If you want the setpoint SP_INT to change autonomously depending on the time, for example, during the control of processes after a time-controlled temperature program, this is possible by configuring a ramp/soak and activating the ramp/soak function. The setpoint change is formed from the level sections with a maximum of 256 time slices.

Note

The block function does not check whether a global DB with the number 'DB_NBR' exists or whether the parameter 'Number of time slices' NBR_PTS corresponds to the data block length. If the parameter assignment is incorrect, the CPU goes to **STOP** and issues an internal system error.

How the ramp/soak works

The ramp/soak supplies the output variable outv according to a function that runs to a defined ramp/soak. The ramp/soak process is defined by specifying a series of time slices in a global data block DB_RMPSK with time values PI[i].TMV and the associated output values PI[i].OUTV.

- PID-CP
  AUTOHOTSPOT is not included in the library. You must create this global data block yourself.
- PID-ES
  AUTOHOTSPOT is not included in the library. You must create this global data block yourself.

The following figure shows a ramp/soak with 6 time slices.

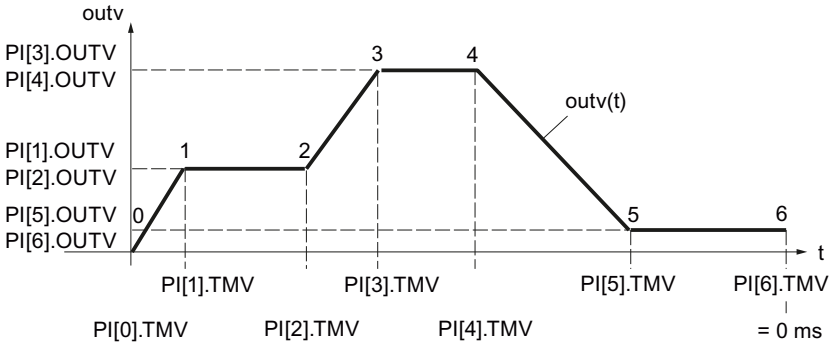

The time value for the last time slice n is PI[n].TMV = 0 ms. The processing time of a ramp/soak is counted down from the initial value to zero.

PI[i].TMV specifies the intervals between time slices. Between the time slices, interpolation is according to the following function:

$$\text{outv}(t) = \text{PI}[n + 1].\ \text{OUTV} - \frac{\text{RS_TM}}{\text{PI}[n].\text{TMV}}\ (\text{PI}[n + 1].\text{OUTV} - \text{PI}[n].\text{OUTV})$$

$0 \leq n \leq (\text{NBR_PTS} - 1)$

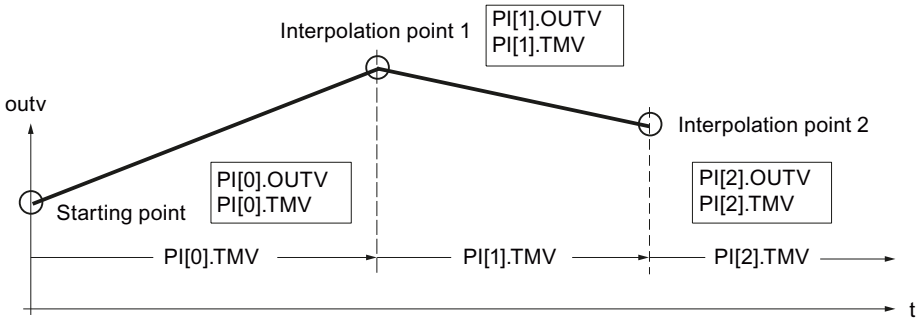

Note

Note

During the interpolation of the ramp/soak between the time slices, pauses can sometimes occur at the initial value if the sampling time CYCLE is very small compared to the time between the time slices PI[n].TMV. Due to the computational accuracy of the CPU, the ramp/soak function cannot linearize all flat ramp/soak profiles. If the ramp/soak profile is too flat, the initial value pauses for a time at the respective time slice and continues, after a certain time, to integrate with the minimum rate of rise to the next time slice.

To correct or avoid error: Shorten the time between the time slices by inserting additional time slices. The output ramp/soak profile thus approximates the desired flat ramp/soak profile.

Ramp/soak function ON

The ramp/soak function is activated by changing RMPSK_ON from FALSE to TRUE. The ramp/soak is terminated after the last time slice has been reached. When the function is started again by the operator, RMPSK_ON first has to be set to 'FALSE' and then back to 'TRUE' .

During a **restart**, the output outv is reset to 0.0 and the total time or total remaining time is determined. During the transition to normal operation, the ramp/soak profile is processed immediately from the starting point according to the set operating mode. If this is not desired, the parameter RMPSK_ON must be set to FALSE in the restart OB.

Ramp/soak functions

The instruction has the following functions:

- Activate ramp/soak for single execution
- Pre-assign fixed value to the output
- Cyclic repetition on
- Hold ramp/soak
- Continue ramp/soak
- Update the total processing time and remaining total time

The following table applies for the significance of the control inputs when setting a desired operating mode:

Operating mode	RMPSK_ ON	DFOUT_ ON	RMP_ HOLD	CONT_ ON	CYC_ ON	TUPDT_ ON	Output signal OUTV
Ramp/soak on	TRUE	FALSE	FALSE	**)	FALSE	**)	OUTV(t) End value is held after the end of processing.
Set output	TRUE	TRUE	**)	**)	**)	**)	DF_OUTV
Cyclic repetition on	TRUE	FALSE	FALSE	**)	TRUE	**)	OUTV(t) After end: Automatic start
Hold ramp/soak	TRUE	FALSE	TRUE	FALSE	**)	**)	Current value of OUTV(t) is held *)

Operating mode	RMPSK_ ON	DFOUT_ ON	RMP_ HOLD	CONT_ ON	CYC_ ON	TUPDT_ ON	Output signal OUTV
Continue ramp/soak	TRUE	FALSE	TRUE	TRUE	**)	**)	OUTV (old) *)
			FALSE				Operation is continued with newly defined values.
Updating the total time and remaining total time	**)	**)	**)	**)	**)	FALSE	Does not influence OUTV
						TRUE	Does not influence OUTV

*) The ramp/soak profile does not have the form configured by the user until the next time slice.

**) The respective operating mode set is executed irrespective of the significance of the control signals in the fields marked with **)

Set output of ramp/soak

If you enable this function (DFRMP_ON = TRUE), the output value of the ramp/soak is set to signal value SP_INT or output value SP_GEN.

Note

Preassignment with the internal setpoint has an effect only if the ramp/soak function is enabled (RMPSK_ON = TRUE).

After the switchover to DFRMP_ON = FALSE, outv moves linearly from the specified setpoint (e.g., SP_INT) to the output value of the current time slice number PI[NBR_ATMS].OUTV.

The internal time processing continues running even when a fixed setpoint (RMPSK_ON = TRUE and DFRMP_ON = TRUE) is switched through.

On the start of the ramp/soak profile (RMPSK_ON = TRUE), the fixed setpoint SP_INT continues to be output until DFRMP_ON changes from TRUE to FALSE after the duration of T*.

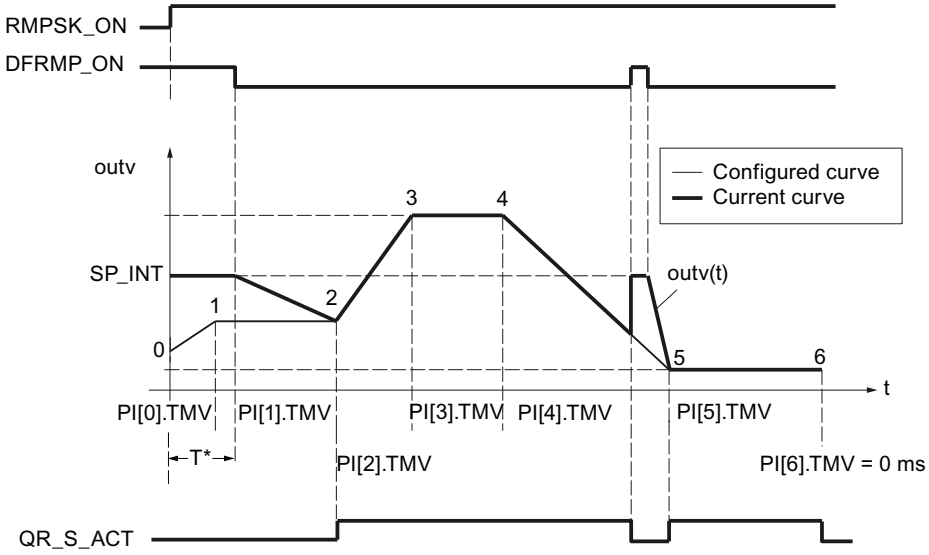

At this moment, the time PI[0].TMV and a part of the time PI[1].TMV has expired. OUTV is moved from SP_INT to PI[2].OUTV, which means to the setpoint at time slice 2.

The configured ramp/soak profile is output only starting from time slice 2. The agreement of the configured and current ramp/soak profile is displayed by setting QR_S_ACT = TRUE. On a rising edge at DFRMP_ON during processing of the ramp/soak profile, the output value OUTV jumps to SP_INT without a delay.

Cyclic repetition on

If "Cyclic repetition" operating mode (CYC_ON = TRUE) is activated, the ramp/soak function automatically returns to the starting point after the last time slice value has been output and begins a new run.

There is no interpolation between the last time slice and the starting point. For a smooth transition, the following must be true: PI[NBR_PTS].OUTV = PI[0].OUTV.

Hold ramp/soak

When RMP_HOLD = TRUE, the value of the output variable (including the time processing) is frozen. When RMP_HOLD = FALSE is reset, the function resumes at the interruption point PI[x].TMV.

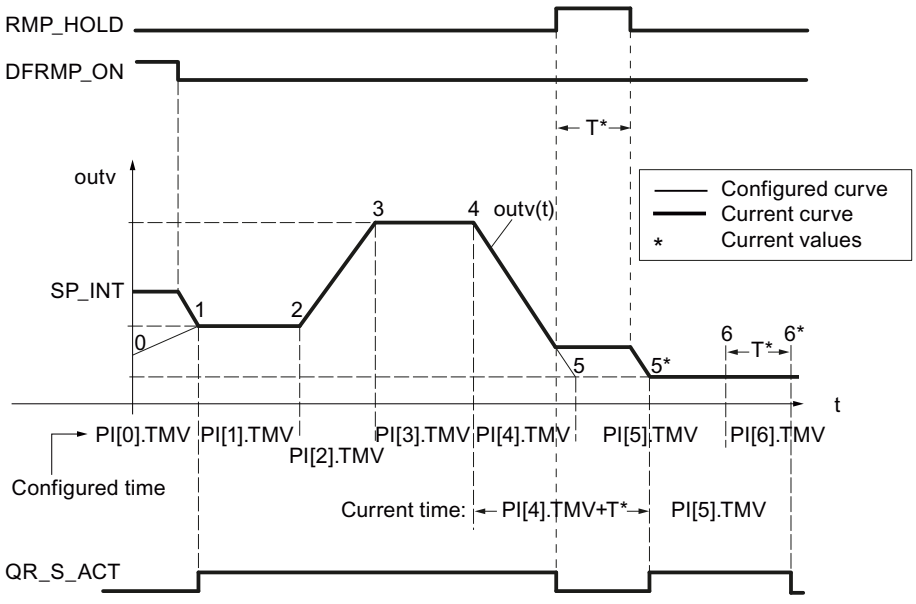

The processing time of the ramp/soak profile is extended by the amount of the time T*. The ramp/soak profile has the configured course from the starting point until the rising edge at RMP_HOLD and from time slice 5* to time slice 6*, i.e., output signal QR_S_ACT has the value TRUE.

If switch CONT_ON is set, the held ramp/soak function continues processing the ramp/soak profile at preassigned point TM_CONT.

Continue ramp/soak

With CONT_ON, a held ramp/soak profile is restarted at a defined point. If the ramp/soak function is not held, CONT_ON has no effect.

If the control input for continuing (CONT_ON = TRUE) is set, the held ramp/soak function is continued at time slice TM_SNBR. TM_CONT determines the remaining time that the ramp/soak function requires to reach time slice TM_SNBR.

The following figure shows the effect of RMP_HOLD and CONT_ON when the configuration is as follows:

Continue at time slice 5 (TM_SNBR = 5) and continue in T* s: TM_CONT = T*

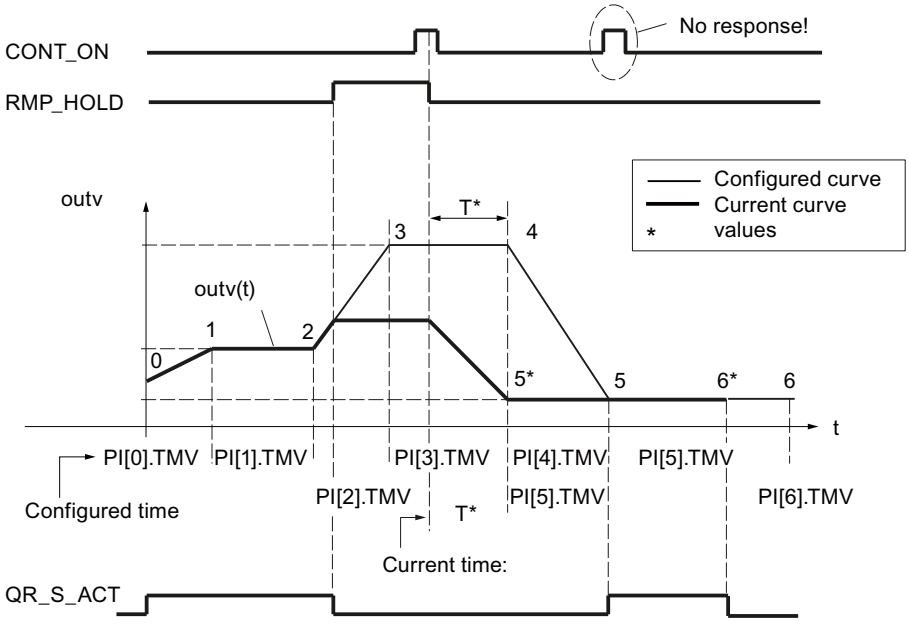

Time slices 3 and 4 are omitted in this processing cycle of the ramp/soak function. On a signal change of RMP_HOLD from TRUE to FALSE, the configured curve progression is only reached again starting from time slice 5.

Output QR_S_ACT is only set when the ramp/soak function executes the ramp/soak profile assigned by the user.

Updating the total time and remaining total time

The current time slice number NBR_ATMS, the current remaining time until the time slice is reached (RS_TM), the total time T_TM, and the total remaining time until the end of the ramp/soak profile (RT_TM) are updated in every cycle.

In the case of online changes of PI[n].TMV, the total time and the total remaining time of the ramp/soak profile change. Since the calculation of T_TM and RT_TM with many time slices significantly increases the processing time of the function block, the calculation is only carried after a restart or when TUPDT_ON = TRUE. The time intervals PI[0 ... NBR_PTS].TMV between the individual time slices are added up and indicated at the total time T_TM and total remaining time RT_TM outputs.

Please note that determining the total times requires significant CPU runtime.

Procedure

To configure the ramp/soak function, proceed as follows:

1. Select "Ramp/soak" from the "Setpoint source" drop-down list.
2. Create a AUTOHOTSPOT (PID-CP) or AUTOHOTSPOT (PID-ES).
3. Define the required parameters for the ramp/soak.
4. Enter the number of the data block DB_RMPSK in the "DB number" input field.
5. Enable or disable the additional functions of the ramp/soak.
6. If you have enabled the Continue ramp/soak function, enter at which time slice and when the ramp/sock is to be continued.

Scaling (S7-300, S7-400)

Scaling of an external setpoint

If the external setpoint is not expressed in the physical unit of the process value (e.g., expressed as a percent in the case of cascade control), then this value and its setting range must be scaled to the physical unit of the process value. The "Scaling in setpoint branch" function is used for this.

The SP_NORM function scales an analog input variable. The analog external setpoint is transformed to output variable outv using the scaling level. Output value OUTV is accessible in the configuration tool via measuring point MP2.

The output value of the function has an effect when control input SPEXT_ON = TRUE. In order to uniquely specify the scaling levels, you define the following four parameters:

- High input: SP_EXT: NM_SPEHR
- Low input: SP_EXT: NM_SPELR
- High output: NM_PVHR (You specify this value in the scaling function of the process value.)
- Low output: NM_PVLR (You specify this value in the scaling function of the process value.)

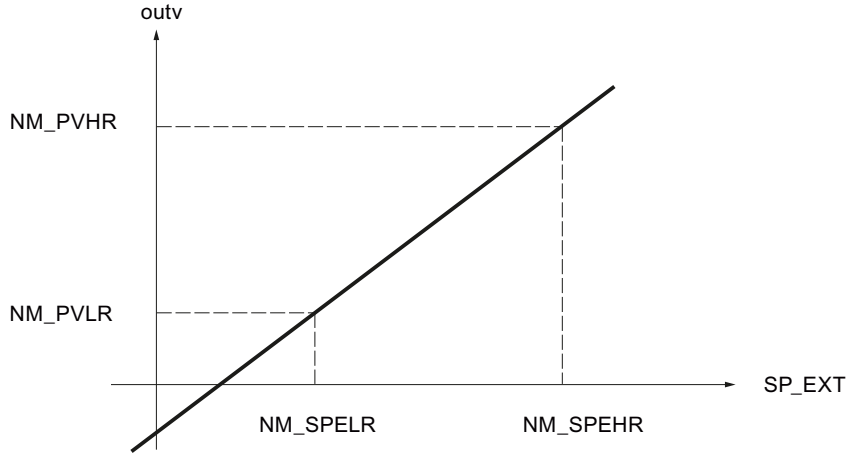

Output value outv is calculated from the respective input value SP_EXT according to the following formula:

outv = (SP_EXT – NM_SPELR) x (NM_PVHR – NM_PVLR) / (NM_SPEHR – NM_SPELR) + NM_PVLR

For the special case in which a square root function is activated in the process value branch, the scaling values of the square root function (SQRT_HR and SQRT_LR) apply as the high and low limits of output value outv.

In this case, output value outv of the scaling function is calculated from the respective input value SP_EXT according to the following formula:

outv = (SP_EXT – NM_SPELR) x (SQRT_HR – SQRT_LR) / (NM_SPEHR – NM_SPELR) + SQRT_LR

There is no internal value limiting in the function. The parameters are not checked. If you enter the same value for NM_SPEHR and NM_SPELR, a division by zero is possible based on the formulas above. The instruction will not detect this.

Combination/ratio factor

With ratio control, the setpoints of the control loop are coordinated with one another in such a way that the set ratio matches the actual ratio of the process values.

With blending control, the desired quantities of the individual components are set. The total of the blending factors FAC must be 1.

Procedure

To configure the scaling, proceed as follows:

1. Enter the values for the scaling in the input fields "High input" and "Low input".
2. Enter a blending/ratio factor (FAC).

Limiting functions (S7-300, S7-400)

FC call in the setpoint branch

Insertion of a user-specific function in the setpoint branch allows an externally specified reference variable to undergo signal processing (e.g., signal delay or linearization) prior to application in the controller.

If you activate the FC call in the setpoint branch, a user-specific function is called. The controller calls the FC. In so doing, existing input/output parameters of the user FC are not initialized. Therefore, you must use S7-STL to program the data transfer in the user FC. You can monitor the prepared setpoint at MP3.

The value of SPFC_ON then determines whether a user-programmed function in the form of a standard FC (e.g., a characteristic curve) will be inserted at this point in the setpoint channel or whether the setpoint will continue to be processed without any influence of this type.

Note

The statement does not check to determine if a function exists. If the function does not exist, the CPU switches to STOP with an internal system error.

Procedure

To call a function in the setpoint branch, proceed as follows:

1. Select the "Enable FC call" check box.
2. In the "FC" input field, enter the number of the function to be called.

Limiting of the rate of change of the setpoint

Ramp functions are used in the setpoint branch in situations where the process does not tolerate abrupt changes in the actuating signal, since an abrupt setpoint change generally also causes an abrupt change in the manipulated variable of the controller. These abrupt manipulated variable changes must be avoided, for example, when a gearbox is connected in between an adjustable-speed motor and the load to be driven and an excessively rapid increase of the motor speed would result in overloading of the gearbox.

The SP_ROC function limits the rate of change of the setpoints processed in the controller. This is carried out separately for the rising slope and falling slope and separately for the positive and negative range of the reference variable.

The limiting for each slope of the ramp function in the positive range and in the negative range of the reference variable is entered in the four inputs SPURLM_P, SPDRLM_P, SPURLM_N, and SPDRLM_N. The slopes refer to a rising or falling slope per second. Faster setpoint changes are ramped down to these maximum rates.

For example, if SPURLM_P is assigned the value 10.0 [engineering value range/s], the following values are added to the "old value" of outv in each sampling cycle for as long as inv > outv:

Sampling time	1 s	$outv_{old}$ + 10
	100 ms	$outv_{old}$ + 1
	10 ms	$outv_{old}$ + 0.1

The following figure shows an example of how the signal processing functions. The step functions at input inv(t) become ramp functions at output outv(t).

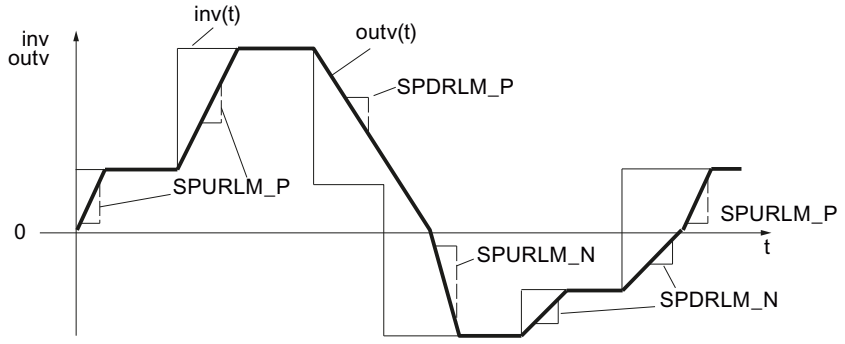

Figure 3-2 Limiting of the rate of change of the reference variable SP(t)

When slope limits are reached, this is not signaled.

Procedure

To limit the rate of change, proceed as follows:

1. Activate the "Limit rate of change" check box.
2. Specify the following values for the positive range as well as for the negative range:
   - Rising slope for setpoint value
   - Falling slope for setpoint value

Limiting of the absolute value of the reference variable

The setting range of the reference variable determines the range in which the controlled variable can fluctuate, i.e., the range in which the process varies within the scope of permissible status values.

To avoid a critical or impermissible process status, the setting range of the reference variable is subject to high and low limits (SP_LIMIT) in the setpoint branch of standard PID control.

The SP_LIMIT function limits the setpoint SP at the assignable low and high values (SP_LLM and SP_HLM) as long as input variable INV is outside these limits. Because the function cannot be deactivated, the specification of a low limit and high limit must be taken into consideration in the configuration.

The numeric values of the limits are set in the input parameters for the low and high limits. In the event of violations by input variable inv(t), the associated displays are output via the message outputs.

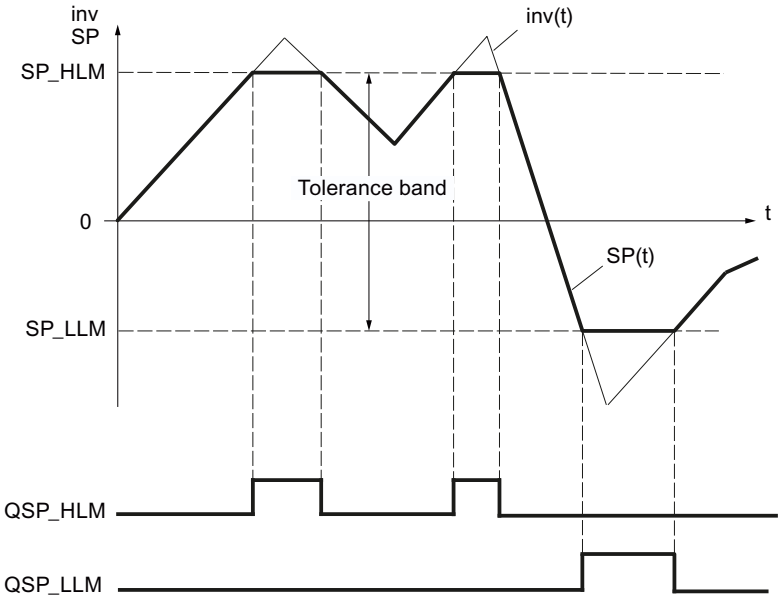

Figure 3-3 Absolute value limiting of reference variable SP(t)

During a restart, the message bits are reset.

The message bits are set as follows:

SP =PVH_ALM	QSP_HLM	QSP_LLM	If:
SP_HLM	TRUE	FALSE	inv ≥ SP_HLM
SP_LLM	FALSE	TRUE	inv ≤ SP_LLM
inv	FALSE	FALSE	SP_HLM < inv < SP_LLM

Procedure

To limit the absolute value of the reference variable, proceed as follows:

1. Enter the high limit for the absolute setpoint (SP_HLM) in the "High limit" input field.
2. Enter the low limit for the absolute setpoint (SP_LLM) in the "Low limit" input field.

Control deviation (S7-300, S7-400)

Deadband (S7-300, S7-400)

In an optimally tuned controller, when a high-frequency interfering signal is superimposed on the control variable or reference variable, the noise portion will also have an effect on the controller output. This can result in strong fluctuations of the manipulated variable, for example, if the controller gain is high and derivative action is enabled. Due to the increased switching frequency (step controller), this causes increased wear of the actuator.

The function reduces the noise portion in the control deviation signal while the controller is in steady-state and thus reduces undesired oscillating of the controller output.

The deadband can be disabled.

Signal filtering through deadband function

The DEADBAND function suppresses small fluctuations of the input variable around a specified zero point within an adjustable range. Outside this fluctuation range, the control deviation ER rises or falls proportional to the input variable. The width of the deadband can be specified with the DEADB_W parameter. Only positive values are permitted for the deadband width.

If the input variable is within the deadband, the value 0 (control deviation = 0) is output at the output. Only when the input variable exits the deadband will the output rise or fall by the same values as input variable inv. This results in corruption of the transmitted signal outside the deadband, too. However, this is accepted in order to avoid step changes at the deadband limits (see figure below). The corruption corresponds to the value DEADB_W and can therefore be controlled easily.

The DEADBAND function operates according to the following functions:

(ER) = inv + DEADB_W for inv < - DEADB_W

(ER) = 0 for -DEADB_W ≤ inv ≤ + DEADB_W

(ER) = inv - DEADB_W for inv > + DEADB_W

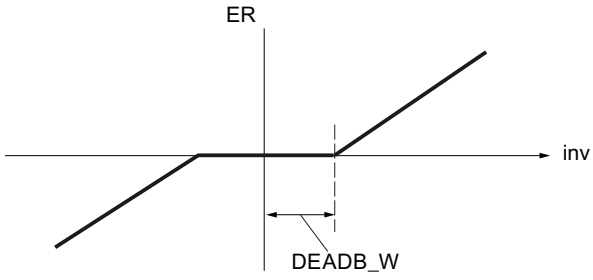

Effects of the signal filtering can be monitored at the output parameter ER.

Alarm (S7-300, S7-400)

Excessive deviations of the controlled variable from the specified setpoint can cause an undesired process status. In such cases, the ER_ALARM function is used to monitor the control deviation for violation of the permissible operating range. ER_ALARM detects and signals any limit violations that occur so that an appropriate reaction can be initiated.

Note

The monitoring of the control deviation for limit values cannot be disabled. The violation of the configured limits is always signaled.

The ER_ALARM function monitors the size of control deviation ER(t) for four assignable limits in two tolerance bands. If the limits are reached or exceeded, the function signals a “warning” at the first limit and an “alarm” at the second limit.

Limit values

The numeric values of the limits are set in the input parameters for “warning” and “alarm”. If the control deviation (ER) violates these limits, the associated output message bits QERN_ALM ... QERP_ALM are set.

During a restart, the message bits are reset.

The message bits are set as follows:

QERP_ALM	QERP_WRN	QERN_WRN	QERN_ALM	If:	And:
TRUE	TRUE	FALSE	FALSE	ER ↗ ER ↘	ER ≥ ERP_ALM ER ≥ ERP_ALM - ER_HYS
FALSE	TRUE	FALSE	FALSE	ER ↗ ER ↘	ER ≥ ERP_WRN ER ≥ ERP_WRN - ER_HYS
FALSE	FALSE	TRUE	FALSE	ER ↘ ER ↗	ER ≤ ERP_WRN ER ≤ ERP_WRN + ER_HYS
FALSE	FALSE	TRUE	TRUE	ER ↘ ER ↗	ER ≤ ERP_ALM ER ≤ ERP_ALM + ER_HYS

The following has to apply so that the block operates properly:

High alarm < High warning < Low warning < Low alarm

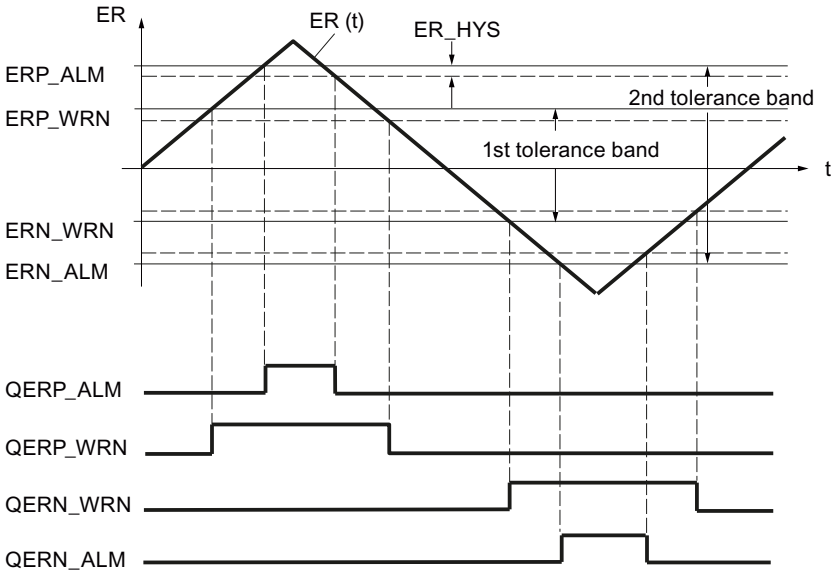

Hysteresis

In order to avoid "flickering" of the message bits when the input variable undergoes slight changes or in the event of rounding errors, a hysteresis ER_HYS is used. The control deviation must violate the hysteresis before the message bits are reset.

Controller (S7-300, S7-400)

Control algorithm (S7-300, S7-400)

The manipulated variable of the continuous controller is calculated within the configured sampling time cycle on the basis of the control deviation in the PID position algorithm. The controller is implemented in a purely parallel structure. The proportional, integral, and derivative actions can each be disabled individually.

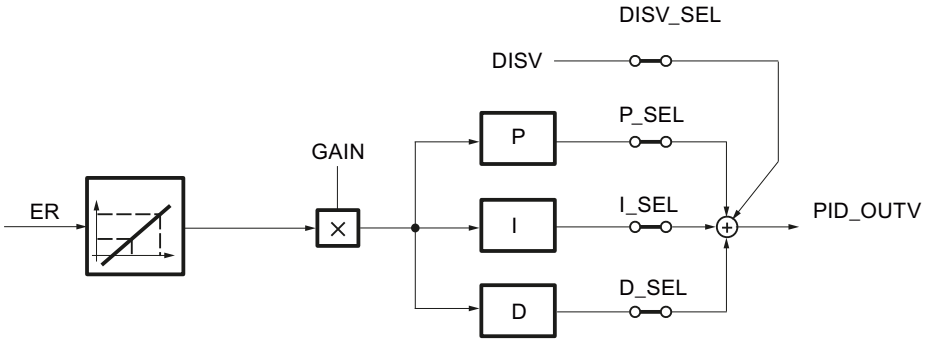

Feedforward control

A disturbance variable DISV can be added in the output signal of the controller (PID_OUTV). The variable is switched-in or switched-off in the PID window of the configuration tool using the DISV_SEL structure selector or through "Disturbance variable on".

Proportional and derivative action in the feedback path

In the parallel structure, each action in the control algorithm receives the control deviation as input signal. Via the proportional action and derivative action, the manipulated variable is influenced directly by setpoint steps. However, a different controller structure in which the formation of the proportional and the derivative actions is moved to the feedback path ensures a smooth response of the manipulated variable to setpoint step changes. In this structure, the integral action processes the control deviation as the input signal. Only the negative controlled variable (factor = - -1) is applied to the proportional action and the derivative action.

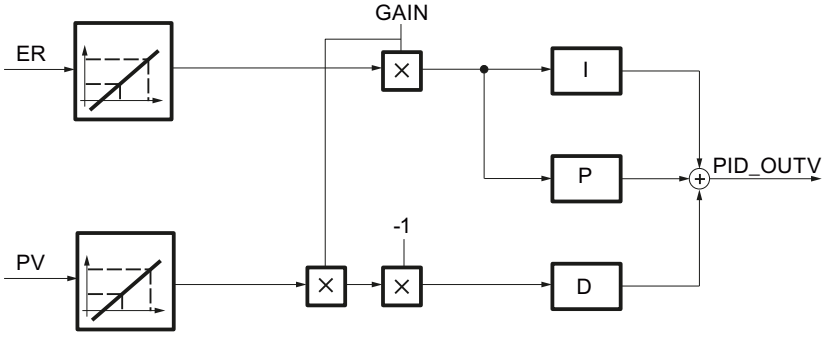

Scaling the input variable ER and PV

The input variables ER and PV of the PID controller are scaled to the range 0 to 100 prior to the controller processing according to the following formulas:

- with square-root function disabled (SQRT_ON = FALSE):
  - ER_{scaled} = ER * 100.0 / (NM_PVHR – NM_PVLR)
  - PV_{scaled} = (PV - NM_PVLR) * 100.0 / (NM_PVHR – NM_PVLR)
- with square-root function enabled (SQRT_ON = TRUE):
  - ER_{scaled} = ER * 100.0 / (SQRT_HR – SQRT_LR)
  - PV_{scaled} = (PV - SQRT_LR) * 100.0 / (SQRT_HR – SQRT_LR)

This scaling is performed so that the proportional gain GAIN of the PID controller can be entered without dimensions. If the high limit and low limit of the physical measuring range change (for example, from bar to mbar), the gain factor does not then have to be changed.

The scaled input variables ER_{scaled} and PV_{scaled} cannot be monitored.

See also

Controller structures (Page 830)

Proportional component (Page 834)

Derivative component (Page 837)

Integral action (Page 834)

Controller structures (S7-300, S7-400)

Use and parameter assignment of the PID controller

The majority of the controlled systems occurring in the process industry can be controlled with the PI controller function/PID controller function of standard PID control. Only in special cases are additional methods and measures needed for a particular closed-loop control.

However, a major problem in the field continues to be how to assign the PI controller/PID controller parameters, i.e., how to determine the "correct" settings for the controller parameters. The quality of this parameter assignment is critical in order for the PID control to meet the requirements of the task at hand. It requires either a lot of practical experience, special expertise, or a great amount of time.

P control

The integral and derivative actions are disabled in a proportional-action controller (P-controller). (I_SEL and D_SEL = FALSE). This means that when control deviation ER = 0, output signal OUTV = 0. If an operating point ≠ 0 is intended, i.e., a numeric value will be set for the output signal when the control deviation is zero, the integral branch must be activated (see the following figure).

In the integral action, an operating point ≠ 0 can be specified for the P-controller via a corresponding setting of initialization value I_ITLVAL. Set switches 'I_ITL_ON' and 'I_SEL' = TRUE for this purpose.

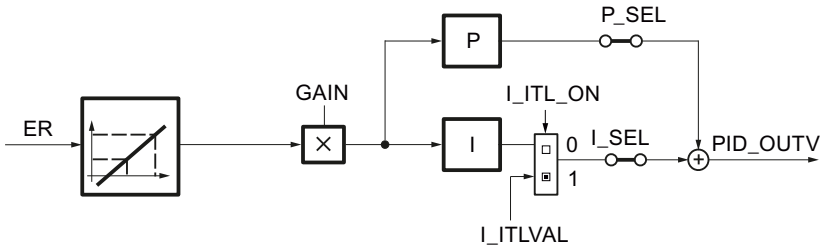

The step response of the P-controller within the time range is:
PID_OUTV (t) = I_ITLVAL + GAIN * ER $_{scaled}$ (t)

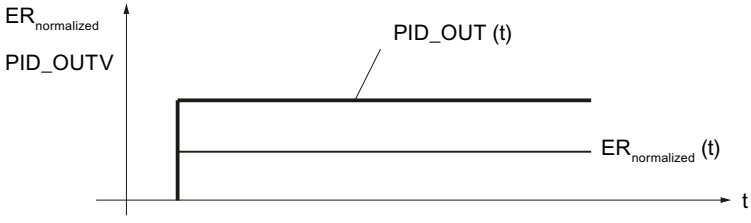

PID_OUTV(t)	Manipulated variable when controller is in automatic mode
I_ITLVAL	Operating point of the P-controller
GAIN	Controller gain
ER $_{scaled}$ (t)	Scaled control deviation

PI control

The derivative action is disabled in the proportional-integral controller (PI controller). (D_SEL = FALSE). A PI controller adjusts output variable PID_OUTV using the integral action until control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the control range.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{1}{TI} * t\right)$$

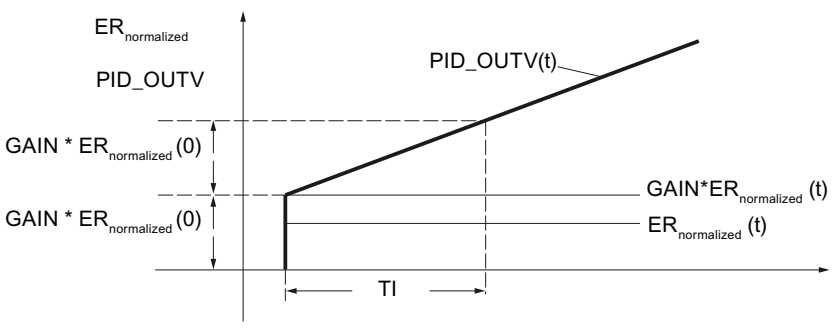

PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TI	Integral-action time constant

PD control

The integral action is disabled in the proportional-derivative controller (PD-controller) (I–SEL = FALSE). This means that when control deviation ER = 0, output signal OUTV = 0. If an operating point ≠ 0 is intended, i.e., a numeric value will be set for the output signal when the control deviation is zero, the integral action branch must be activated (see also the block diagram of the P control).

In the integral action, an operating point ≠ 0 can be specified for the PD-controller via a corresponding setting of initialization value I_ITLVAL. The switches 'I_ITL_ON' and 'I_SEL' must be set to TRUE for this purpose.

The PD-controller maps input variable ER(t) proportionally to the output signal and adds to this the derivative action generated by differentiation of ER(t); the derivative action is calculated with double precision according to the trapezoid rule (Padé approximation). The time response is determined by the derivative time constant TD.

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time: TM_LAG) is integrated into the algorithm for generating the derivative action. A small value for TM_LAG is usually sufficient to achieve the desired result. If TM_LAG ≤ CYCLE/2 is configured, the time delay is deactivated.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{TD}{TM_LAG} * e^{-\frac{t}{TM_LAG}}\right)$$

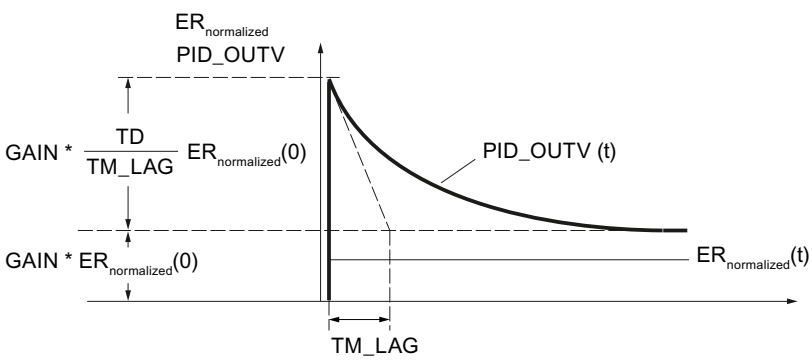

PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TD	Derivative time
TM_LAG	Time delay

PID control

In the proportional-integral-derivative controller (PID-controller), the proportional action, integral action, and derivative action are enabled (P–SEL = TRUE, I–SEL = TRUE, D_SEL = TRUE). A PID controller adjusts output variable PID_OUTV using the integral action until control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the control range. If manipulated variable limits are exceeded, the integral action maintains the value that was reached at the limit (anti-reset windup).

The PID-controller maps scaled input variable $ER_{scaled}(t)$ proportionally to the output signal and adds to this the components generated by differentiation and integration of $ER_{scaled}(t)$; these components are calculated with double precision according to the trapezoid rule (Padé approximation). The time response is determined by the derivative-action time TD and integral-action time TI.

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time constant: TM_LAG) is integrated into the algorithm for generating the derivative action. A small value for TM_LAG is usually sufficient to achieve the desired result. If TM_LAG ≤ CYCLE/2 is configured, the time delay is deactivated.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0) \left(1 + \frac{1}{TI} * t + \frac{TD}{TM_LAG} * e^{-\frac{t}{TM_LAG}} \right)$$

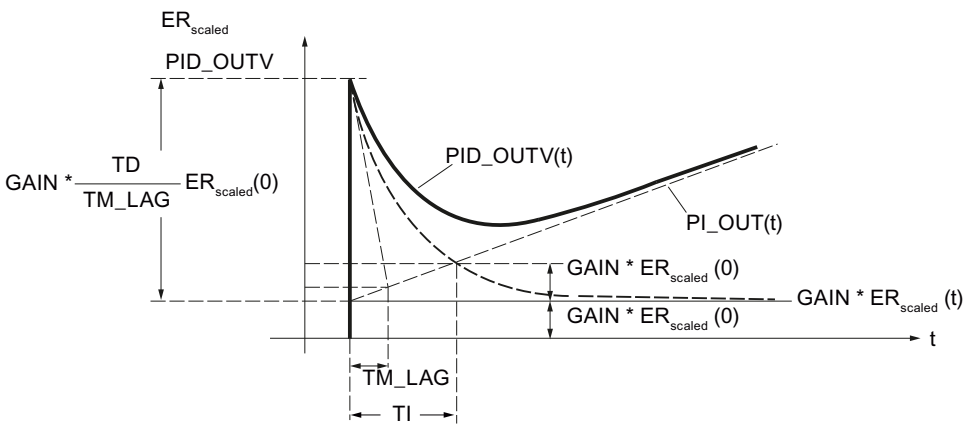

PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TI	Integral-action time constant
TD	Derivative time
TM_LAG	Time delay

Note

If TD is changed, you should also adjust TM_LAG accordingly.

Recommendation: 5 ≤ (TM / TM_LAG) ≤ 10

See also

Control algorithm (Page 829)

Proportional component (Page 834)

Derivative component (Page 837)

Integral action (Page 834)

Proportional component (S7-300, S7-400)

Proportional gain

The sign of this proportional gain determines the effective direction of the continuous controller.

- Positive proportional gain
  increasing control deviation → rising manipulated variable
- Negative proportional gain
  increasing control deviation → falling manipulated variable

Applying proportional action in feedback path

If the proportional action is applied to the feedback path, only the process value has an effect on the proportional action. Therefore, the manipulated variable does not undergo a step change due to the proportional action during a changeover from manual to automatic (also in the case of a setpoint step-change); the changeover is bumpless.

See also

Control algorithm (Page 829)

Integral action (S7-300, S7-400)

Function of the integral action

The integral action ensures that the control deviation can go to 0 for any manipulated variables by resetting the operating point.

The integral action generates an output signal whose rate of change is proportional to the change in the absolute value of the input variable. The time response is determined by the integral action time TI.

The step response to the input step inv_0 is:

$$OUTV(t) = \frac{1}{TI} \int inv(t)\, dt$$

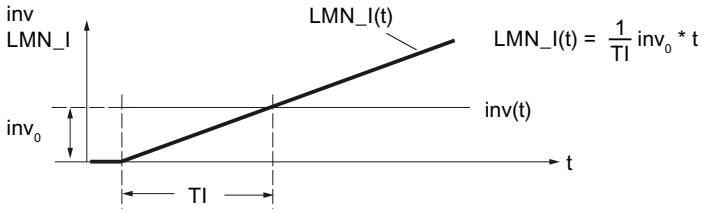

LMN_I(t)	The output variable of the integral action
inv_0	The step height at the input of the integral action
TI	Integral action time

The integral action can be monitored at the static tag LMN_I.

In the deactivated state (I_SEL = FALSE), the integral action, i.e., the internal memory and the static tag LMN_I, is set to zero.

The output and the memory of the integral action are limited by the absolute limits of the manipulated variables LMN_HLM and LMN_LLM (Anti Reset Wind–up).

If the actuating signal is influenced in manual mode, i.e., if MAN_ON, LMNOP_ON, or CAS_ON = TRUE, the internal memory value of the integral action is corrected to the value LMNFC_IN - LMN_P - DISV.

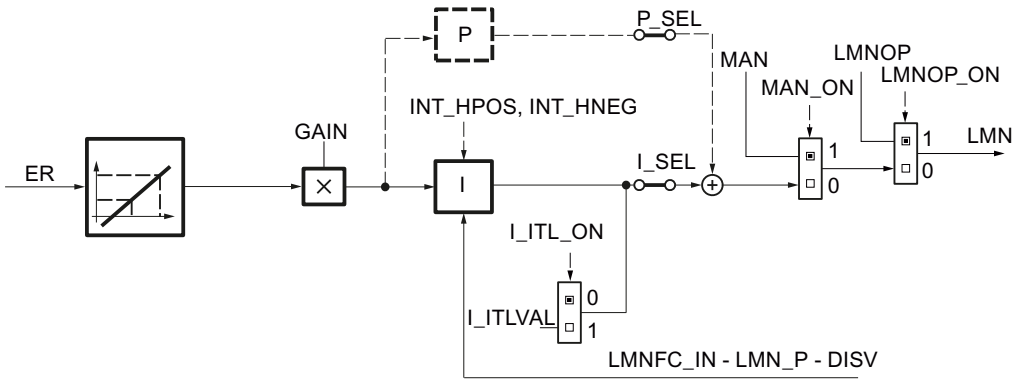

Smooth changeover from manual to automatic

In manual mode, the integral action of the controller is corrected in such a manner that the controller starts with an appropriate manipulated variable upon switchover to automatic mode.

- Smooth changeover enabled (SMOO_CHG = TRUE)
  The integral action is set in manual mode in such a way that the manipulated variable initially remains unchanged on the switchover from manual to automatic.
  In manual mode, the derivative action is set to zero. The switchover to automatic is performed without manipulated variable step.
  A pending system deviation will be slowly corrected.
- Smooth changeover disabled (SMOO_CHG = FALSE)
  The manipulated variable makes a step from the actual position during the switchover from manual to automatic. The height of this step is determined by the proportional action. The step height of the proportional action corresponds to the manipulated variable change on a setpoint step change from the current process value to the current setpoint.
  In manual mode, the derivative action is set to a value that corresponds to the pending control deviation. The switchover to automatic mode is performed with a manipulated variable step; the control deviation is corrected faster.
  However, if the proportional action is applied to the feedback path, only the process value has an effect on the proportional action. Therefore, the manipulated variable does not undergo a step change due to the proportional action during a switchover from manual to automatic; the changeover is smooth. The same applies to the derivative action, if this was moved to the feedback path (DFDB_SEL = TRUE).
  The pending control deviation is corrected faster. This is desirable, for example, for temperature control processes.

Permissible ranges for the integral action time TI and CYCLE

The limited accuracy of the REAL numbers calculated in the CPU means that the following can occur during integration: If the selected sampling time CYCLE of the control is too short in comparison with the integral action time TI and the input value inv of the integral action is small compared to its output value OUTV, the integral action does not respond and remains at its current output value.

This effect can be avoided if the following sizing rule is observed during configuration:

$CYCLE > 10^{-4} \times TI$

As a result, the integral action still responds to changes in the input values in the range of millionths of one part per thousand of the current output variable:

$inv > 10^{-10} \times OUTV$

In order for the response characteristic of the integrator algorithm to conform to the analog response, the sampling time should be less than 20% of the specified integral action time or TI should be at least five times the value of the specified sampling time:

$CYCLE < 0.2 \times TI$

The algorithm permits sampling time values up to $CYCLE \leq 0.5 * TI$.

Initializing the integral action

If initialization of the integral action is enabled, the initialization value is switched through to the static tag LMN_I at a restart of the CPU or COM_RST = TRUE. On the transition to normal operation, the integral action starts the integration of its input variable from the initialization value due to the resetting of the initialization.

Hold integral action

The integral action can be blocked in positive or negative direction by the static tags INT_HPOS and INT_HNEG. This can be useful for controller cascades. If, for example, the manipulated variable of the follow-up controller approaches the high limit, a further increase in the manipulated variable of the master controller can be prevented by its integral action. You program this response using the following instructions, for example, in STL:

```
A    "Follow-up controller".QLMN_HLM
=    "Master controller".INT_HPOS
A    "Follow-up controller".QLMN_ LLM
=    "Master controller".INT_HNEG
```

See also

Control algorithm (Page 829)

Controller structures (Page 830)

Derivative component (S7-300, S7-400)

Function of the derivative action

The derivative action generates an output signal whose size changes proportionally to the rate of change of the input variable. The time response is determined by the derivative time TD and the time delay TM_LAG.

The step response to the input step inv_0 is:

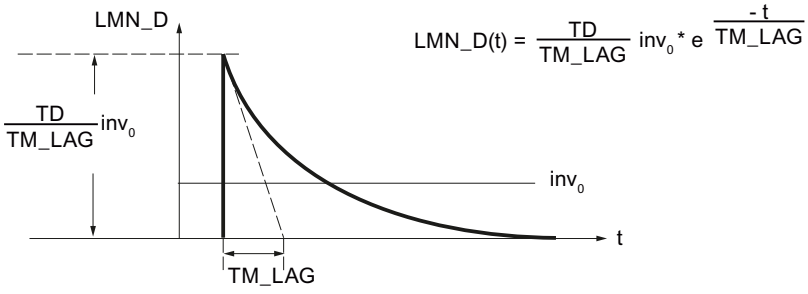

LMN_D(t)	The output variable of the derivative action
inv_0	The step height at the input of the derivative action
TD	Derivative-action time
TM_LAG	Time delay

The derivative action can be monitored at the static tag LMN_D.

In the disabled state (D_SEL = FALSE), the derivative action, i.e., the internal memory and the static tag LMN_D, is set to zero.

Apply derivative action in feedback path

If the derivative action is applied to the feedback path, only the process value has an effect on the derivative action. Consequently – as in a setpoint jump – the manipulated variable does not undergo a step change due to the derivative action during a changeover from manual to automatic; the switchover is smooth.

Permitted values for derivative time and time delay.

To calculate the derivative action correctly, the following conditions must be satisfied for the derivative time and time delay:

- TD ≥ CYCLE and
- TM_LAG ≥ 0.5 × CYCLE

If TD < CYCLE, the derivative action operates as though the TD has the value CYCLE.

If TM_LAG < 0.5 × CYCLE, the derivative action operates without delay. An input step is then multiplied with the factor TD/CYCLE and this value is applied as "needle pulse" at the output. In other words, LMN_D is reset to zero in the subsequent processing cycle.

Coefficient DT1

The coefficient DT1 is calculated as follows:

Coefficient DT1 = TD/TM_LAG

Time delay

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time constant: TM_LAG) is integrated into the algorithm for generating the derivative action.

A small value for the time delay is usually sufficient to achieve the desired result. If the assigned time delay is less than half the controller sampling time, the delay is deactivated.

Procedure

To configure the derivative action, proceed as follows:

1. To apply the derivative action to the feedback path, select the corresponding check box.
2. Enter the derivative time in seconds.
3. Enter the time delay in seconds.

See also

Control algorithm (Page 829)

Manipulated variable (S7-300, S7-400)

Manual mode (S7-300, S7-400)

In addition to "automatic" mode, in which the output of the PID algorithm (PID_OUTV) is switched through to the output, there are two other operating modes in which the actuating signal can be influenced manually: "Manual mode without switch operation" and "Manual mode using more/less switch" (MAN_GEN).

The MAN parameter can be used to switch through an external manipulated variable or to specify it from the user program. Input value MAN is limited by the manipulated variable limits (LMN_HLM (high) and LMN_LLM (low)).

The structure of the manual value generation and manual value application emerges from the following figure. When MAN_GEN is activated from another mode, the current manipulated variable in effect is applied at the output of MAN_GEN. The changeover to the manual value generator is thus always bumpless.

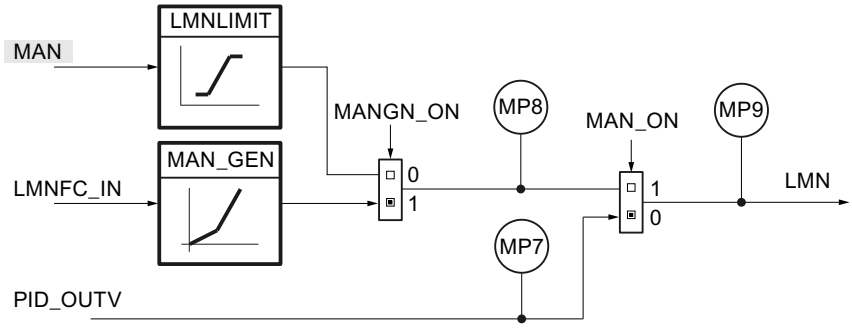

Activating automatic mode

If manual mode is disabled (MAN_ON = FALSE), the manipulated variable of the PID algorithm is switched through to the output. In manual mode (MAN_ON = TRUE), the integral action of the controller is corrected such that the controller begins with an appropriate manipulated variable on changeover to automatic mode. The output of the PID algorithm is stored in measuring point MP7.

In automatic mode, the manual value MAN of the manipulated variable (minus the derivative action) is corrected. On changeover to manual mode, the manipulated variable is kept at the last calculated value. It can be changed via operator input.

1. Clear the "Activate manual mode" check box.

Activate manual mode with manual value

In this mode (MANGN_ON = FALSE and MAN_ON = TRUE), the manual output value is entered as an absolute value in the MAN input. The manual output value is displayed at measuring point MP8.

1. Select the "Activate manual mode" check box.
2. Choose the "Manual value" option.

Activate the manual value generator

The manual value MAN can be incremented or decremented in steps using the MANUP and MANDN switches. The modified value can be monitored at MP8. The configured manipulated variable limits are taken into account.

1. Select the "Activate manual mode" check box.
2. Select the "Manual value generator" option.

Effect of the switches MANUP and MANDN

For changes involving small steps, the controller should have a maximum sampling time of 100 ms.

The rate of change of the manipulated variable depends on how long the MANUP or MANDN switch is actuated and on the selected limits. The following relationship applies:

- During the first 3 s after setting MANUP or MANDN:
  doutv/dt = (LMN_HLM - LMN_LLM) / 100 s
- Afterwards:
  doutv/dt = (LMN_HLM - LMN_LLM) / 10 s

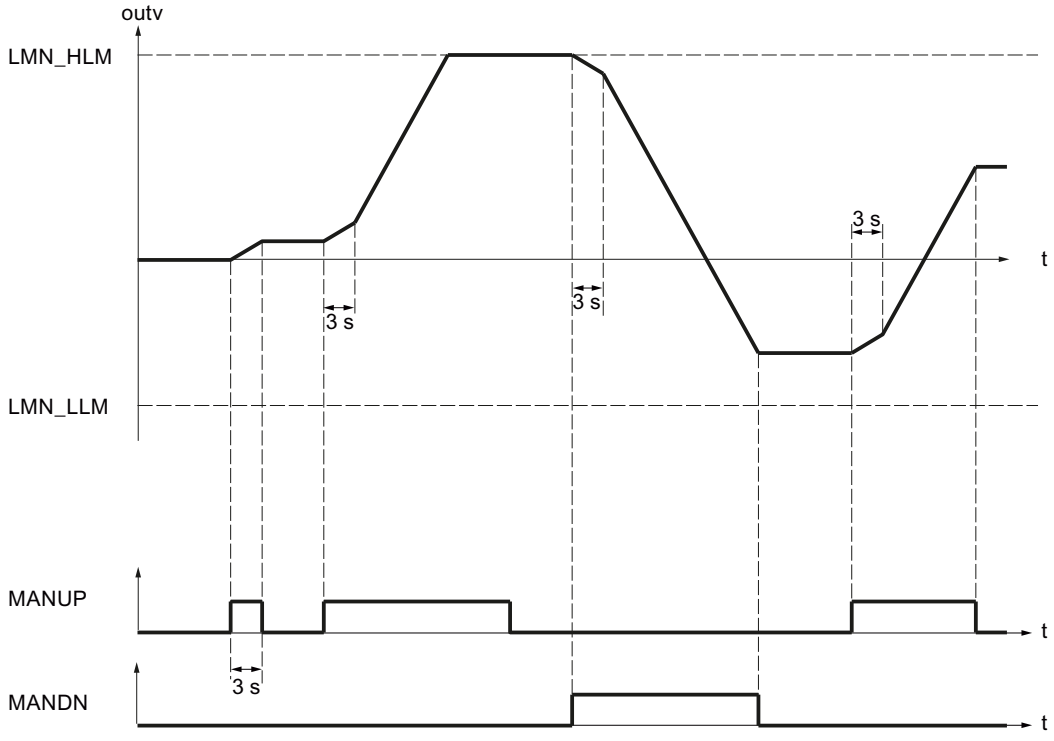

For a sampling time of 100 ms and a manipulated variable range of -100.0 to 100.0, the manipulated variable, for example, changes by 0.2 per cycle during the first 3 seconds. If MANUP is TRUE for more than 3 seconds, the rate of changes increases to ten times the value, in this case therefore to 2 per cycle.

The manual value generator is activated bumpless.

Operating modes

The following table shows the possible operating modes of the continuous controller with the required values of the switches.

Operating mode	MAN_ON	MANGN_ON
Automatic mode	FALSE	Not relevant
Manual mode with manual value	TRUE	FALSE
Manual mode with manual value generator	TRUE	TRUE

Limiting functions (S7-300, S7-400)

Calling a function

If you insert a user-specific function into the manipulated variable branch, the manipulated variable PID_OUTV generated in the controller can undergo signal processing (e.g., a signal delay) prior to application to the output of the controller.

The controller calls the function. In so doing, existing input/output parameters of the user function are not supplied. You have to program the data transfer in your function yourself.

Note

A check is not performed to determine whether a function exists. If the function does not exist, the CPU switches to STOP with an internal system error.

1. Select the "Enable FC call" check box.
2. Enter the number of the function in the "FC" input field.

Limiting the rate of change

Ramp functions will be used in the manipulated variable branch if the process does not tolerate abrupt changes in the controlled system input signal. These abrupt manipulated variable changes must be avoided, for example, when a gearbox is connected in between an adjustable-speed motor and the load to be driven and an excessively rapid increase of the motor speed would result in overloading of the gearbox.

The rate of change of the manipulated variable at the output of the controller is limited separately for the rising and falling slope. Two ramps with rising or falling values can be assigned for the entire value range, when seen from the zero point. The function is enabled if LMNRC_ON = TRUE.

The limiting for each slope of the ramp function in the positive range and in the negative range of the manipulated variable is entered in the two static variables LMN_URLM and LMN_DRLM. The slopes refer to a rising slope or falling slope in percent per second. Faster manipulated variable changes are ramped down to these maximum rates.

For example, if 'LMN_URLM' is assigned the value 10.0 [%/s], the following values are added to the "old value" of outv in each sampling cycle for as long as |inv| > |outv|:

Sampling time 1 s → $outv_{old}$ + 10%
100 ms → $outv_{old}$ + 1%
10 ms → $outv_{old}$ + 0.1 %

The figure below shows how the signal processing works: The step functions at input inv (t) become ramp functions at output outv (t).

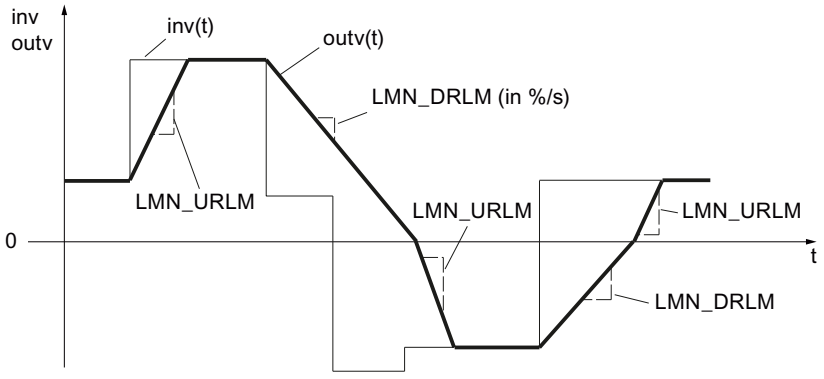

When slope limits are reached, this is not signaled.

The ramp parameters are identified in accordance with the following scheme:

Parameter	Ramp
LMN_URLM	|outv| rising
LMN_DRLM	|outv| falling

1. Activate the "Limit rate of change" check box.
2. Enter the maximum increase of the manipulated variable in the "Maximum increase" field. Faster manipulated variable changes will be ramped down to this value.
3. Enter the maximum decrease of the manipulated variable in the "Maximum decrease" field. Faster manipulated variable changes will be ramped down to this value.

Limiting the absolute value of the manipulated variable

The operating range, i.e., the range in which the actuator can vary within the scope of permissible manipulated variables, is determined by the setting range of the manipulated variable. Since the limits for allowed manipulated variables usually do not coincide with the 0% or 100% limit of the correcting range, depending on the type of actuator, you can configure the high and low limit of the manipulated variable.

To avoid an impermissible status in the particular process, high and low limits for the setting range of the manipulated variable will be set in the manipulated variable branch of the closed-loop control.

The manipulated variable LMN(t) is limited to the assignable high limit (LMN_HLM) and low limit (LMN_LLM). However, the input variable inv must be outside these limits. Because the

function cannot be deactivated, the specification of a low limit and high limit must be taken into consideration in the configuration.

In the event of violations by input variable inv(t), the associated displays are output via the message outputs.

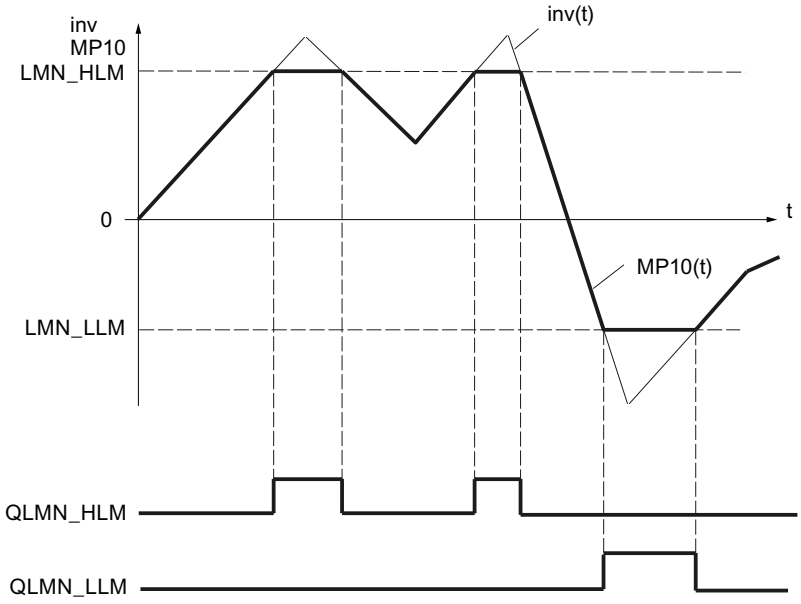

Startup and operating principles

- During a restart, all message outputs are set to zero.
- The limiting operates according to the following relationships:

LMN =	QLMN_HLM	QLMN_LLM	If:
LMN_HLM	TRUE	FALSE	inv ≥ LMN_HLM
LMN_LLM	FALSE	TRUE	inv ≤ LMN_LLM
inv	FALSE	FALSE	LMN_HLM < inv < LMN_LLM

The effective manipulated value of the controller is displayed at the output, i.e. at the parameter LMN or at the measuring point MP10.

Procedure

To limit the absolute value of the manipulated variable, proceed as follows:

1. In the "High limit" input field, enter the value for the maximum permissible value.
2. In the "Low limit" input field, enter the value for the minimum permissible value.

Scaling (S7-300, S7-400)

If a physical quantity is required for the manipulated variable at the controlled process input, the floating-point values in the range from 0 to 100% must be unscaled to the physical range (e.g., 150 to 3000 rpm) of the manipulated variable.

Two parameters are defined for the unique specification of the scaling levels:

- The factor for the slope (LMN_FAC)
- The offset of the scaling level at the zero point (LMN_OFF)

Internal percentage values (in REAL format) ⇒ external physical values

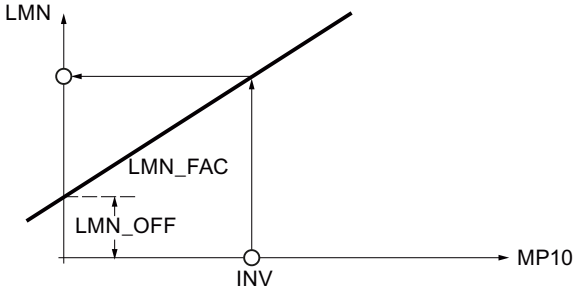

The scaling value is calculated from the respective input value MP10 according to the following function:

LMN = MP10 * LMN_FAC + LMN_OFF

Example

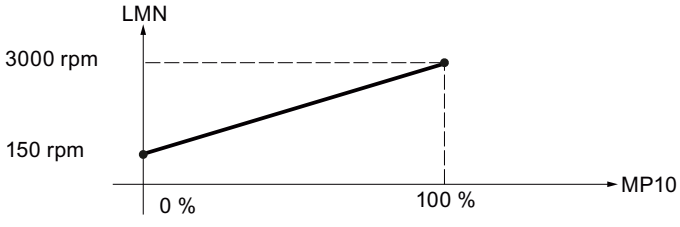

$$\text{LMNFAC} = \frac{(3000 - 150)\ \text{rpm}}{(100 - 0)\%} = 28.5\ \frac{\text{Rpm}}{\%}$$

LMNOFF = 150 rpm

Values are not limited internally in the function and the parameters are not checked.

Performing scaling

1. Enter the slope of the scaling level in the "Factor" input field.
2. Enter the offset of the scaling level with respect to the zero point in the "Offset" input field.

Pulse generator (S7-300, S7-400)

Application

The pulse generator transforms the input variable “Manipulated variable of PID controller at measuring point MP10" by modulating the pulse width into a pulse train with a constant period. The duration of a pulse per period is proportional to the input variable.

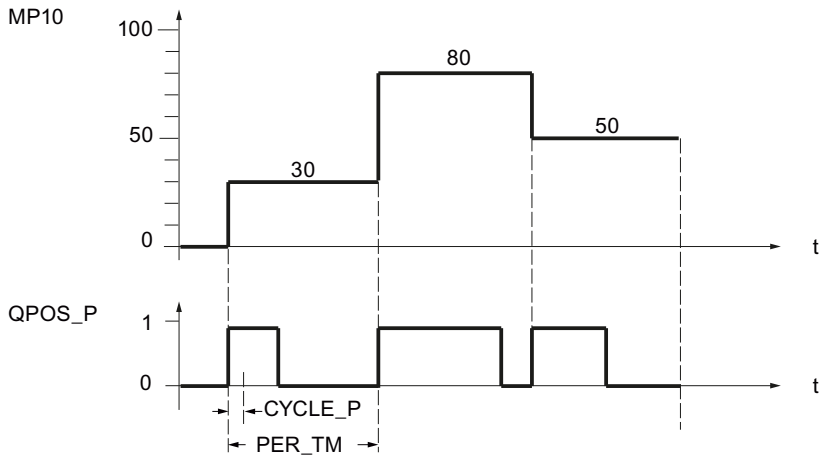

An input variable of 30% and 10 calls of the pulse generator per period therefore means:

- "One" at the QPOS output for the first three calls (30% of 10 calls).
- "Zero" at the QPOS output for seven further calls (70% of 10 calls).

Operating modes of controller with pulse output

Depending on the parameter assignment of the pulse generator, PID controllers with a three-step response or with a bipolar or unipolar two-step output can be configured. The following table illustrates the setting of the switch combinations for the possible operating modes.

Operating mode	Switch		
	MAN_ON	STEP3_ON	ST2BI_ON
Three-step control	FALSE	TRUE	Any
Two-step control with bipolar Correcting range (-100% to 100%)	FALSE	FALSE	TRUE
Two-step control with unipolar Correcting range (0% to 100%)	FALSE	FALSE	FALSE
Manual mode	TRUE	Any	Any

Three-step control

In "Three-step control" mode, three control signal states can be generated, e.g., depending on the actuator and process: More – OFF – Less, Forward – Stop – Backward, Heat – OFF – Cool, etc. The status values of the binary output signals QPOS_P and QNEG_P are assigned to the respective operating states of the actuator, based on the requirements of the process being controlled. The table shows two examples.

	Heat Forward	Off Stop	Cool Backward
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

Selecting sizes for the minimum pulse time or minimum break time P_B_TM_P can prevent very short activation or deactivation times, which reduce the operating life of switch elements and actuating devices considerably. In so doing, a response value for the pulse output is superimposed on the proportional output characteristic curve used to calculate the pulse duration.

Note

Small absolute values of the "Manipulated variable of the PID controller at measuring point MP10" input variable that would generate a pulse duration less than P_B_TM_P are suppressed. For large input values that would generate a pulse duration longer than PER_TM_P – P_B_TM_P, the pulse duration is set to 100% or -100%.

Set values P_B_TM ≤ 0.1 * PER_TM are recommended.

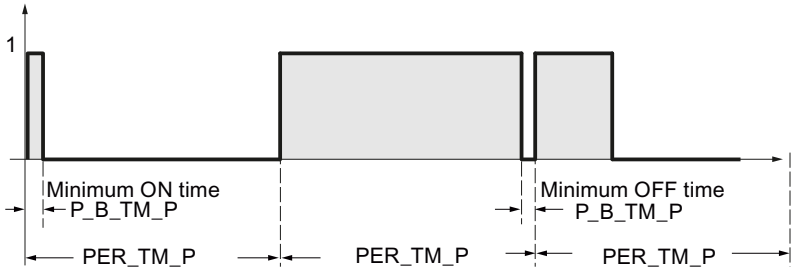

The duration of the positive or negative pulses are calculated by multiplying the "Manipulated variable of PID controller at measuring point MP10" input variable (in %) by the period:

$$\text{Pulse duration} = \frac{\text{MP10}}{100} * \text{PER_TM_P[s]}$$

The suppression of the minimum pulse time or minimum break time causes bend points near the start and end of the conversion characteristic curve.

The above statements also apply to P_B_TM_N and PER_TM_N.

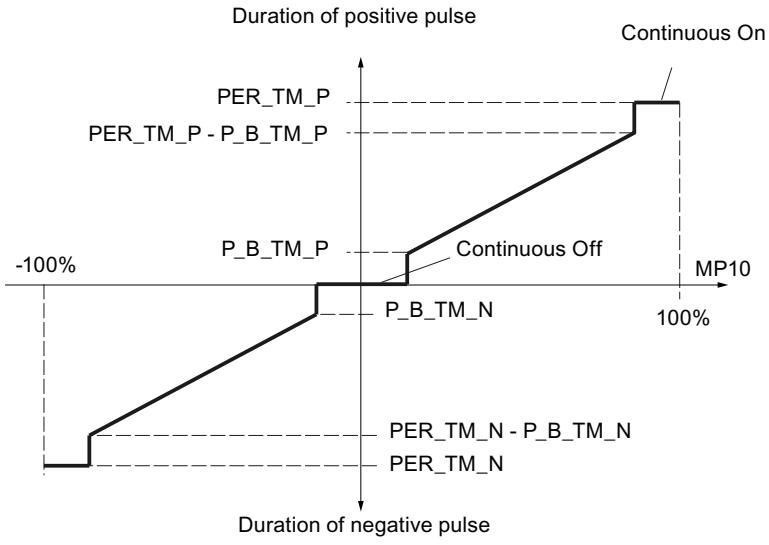

Three-step control asymmetrical

Using the ratio factor RATIOFAC, the ratio of the duration of positive to negative pulses can be changed. In a thermal process, for example, this would allow different process time constants for heating and cooling to be included.

Assuming the absolute value of the "Manipulated variable of the PID controller at measuring point MP10" is the same, a shorter pulse duration at the negative pulse output than for the positive pulse requires a ratio factor setting less than 1:

Pos. pulse > Neg. pulse: RATIOFAC < 1

Pulse duration negative: $\frac{MP10}{100} * PER_TM_N * RATIOFAC$

Pulse duration positive: $\frac{MP10}{100} * PER_TM_N$

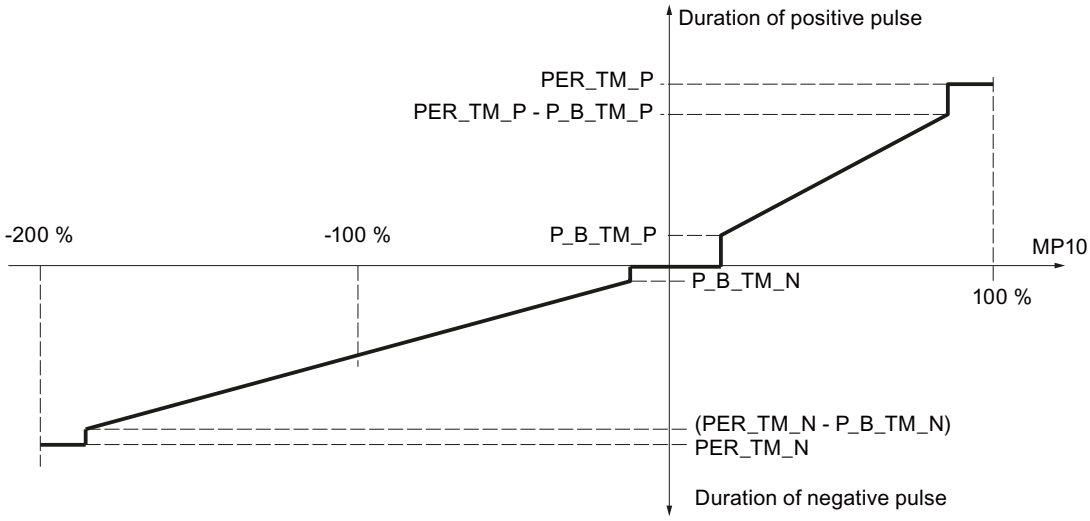

Figure 3-4 Asymmetrical characteristic curve of the three-step controller (ratio factor = 0.5)

On the other hand, for the same absolute value |MP10|, a shorter pulse duration at the positive pulse output compared to the negative pulse requires a ratio factor setting greater than 1:

Pos. pulse < Neg. pulse: RATIOFAC > 1

Pulse duration negative: $\frac{MP10}{100} * PER_TM_N$

Pulse duration positive: $\frac{MP10 * PER_TM_P}{100 * RATIOFAC}$

Mathematically, this means that when RATIOFAC < 1, the response value for negative pulses is multiplied by the ratio factor and when RATIOFAC > 1, the response value for positive pulses is divided by the ratio factor.

Note

For asymmetrical three-step control RATIOFAC ≠ 1, you must adapt the manipulated variable limits according to the following formulas:

RATIOFAC < 1:

LMN_HLM = 100

LMN_LLM = –100 * (1 / RATIOFAC)

RATIOFAC > 1:

LMN_HLM = 100 * RATIOFAC

LMN_LLM = –100

Examples:

RATIOFAC	=	1	RATIOFAC	=	0.5	RATIOFAC	=	2.0
LMN_HLM	=	100	LMN_HLM	=	100	LMN_HLM	=	200
LMN_LLM	=	–100	LMN_LLM	=	–200	LMN_LLM	=	–100

Two-step control

In two-step control, only the positive pulse output QPOS_P is connected to the affected on/off actuator. Depending on the correcting range used, MP10 = -100.0% to +100.0% or MP10 = 0.0% to 100.0%, the two-step controller has a bipolar or unipolar correcting range.

In unipolar operating mode, MP10 may assume values between 0.0% and 100.0% only.

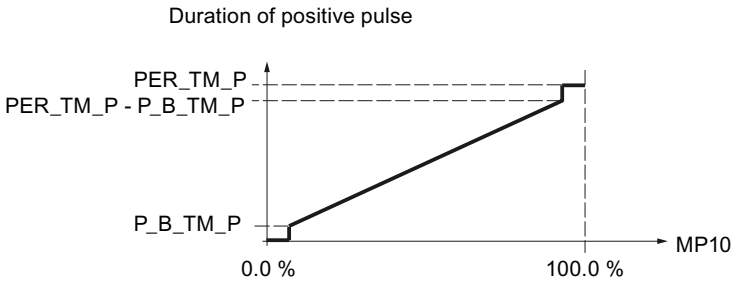

In bipolar operating mode, MP10 may assume values between -100.0% and 100.0%.

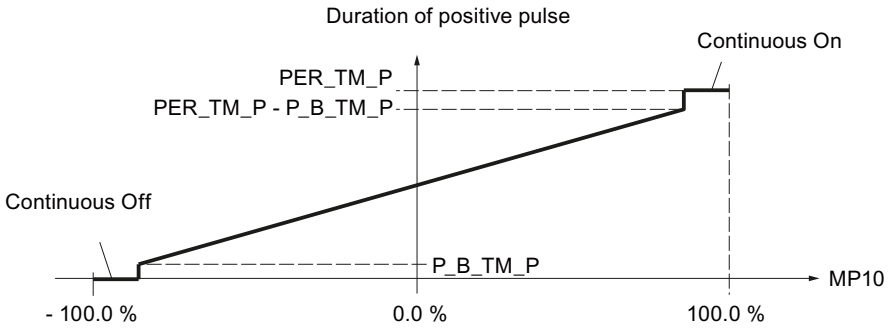

The negated output signal is available at QNEG_P in case the interconnection of the two-step controller in the control loop requires a logically inverted binary signal for the actuating pulses.

	On	Off
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

Controller sampling time CYCLE and pulse pattern width CYCLE_P

If you have activated the pulse generator stage (PULSE_ON = TRUE), you must specify the cycle clock of the calling cyclic interrupt OB in the CYCLE_P input. The duration of the generated pulse is always an integer multiple of this value.

At the CYCLE input, you specify the sampling time for the remaining control functions of the PID_CP. PID_CP determines the time scaling and processes the control functions with the sampling time CYCLE.

CYCLE must be an integer multiple of CYCLE_P. If you do not adhere to this condition, PID_CP rounds up the sampling time for the control functions to an integer multiple of CYCLE_P. The time-dependent functions (e.g., smoothing, integration, differentiation) will then not work correctly to some extent.

The value selected for CYCLE can be less than the period PER_TM_P or PER_TM_N. This is useful if the longest possible period is desired to protect the actuators, but at the same time very short sampling times are required due to a fast process.

The same guideline applies here to the selection of the sampling time CYCLE as for continuous controllers without a pulse generator stage: the value of CYCLE should not be less than approximately 10% of the dominating process time constants of the controlled system.

Example of the effect of the CYCLE_P, CYCLE and PER_TM_P or PER_TM_N parameters:

- PER_TM_P = 10 s, CYCLE = 1 s, CYCLE_P = 100 ms.

A new manipulated variable is calculated every second, and the manipulated variable is compared with the previously output pulse length or break length every 100 ms.

If a pulse is output and calculate manipulated variable is greater than the previous pulse length/ PER_TM_P, the pulse is extended. Otherwise, no more pulse signals are output. If no pulse is output and (100% - calculated manipulated variable) is greater than the previous break length/PER_TM_P, the break is extended. Otherwise, a pulse signal is output.

As a result of a special pulse generation procedure, an increase or decrease in the manipulated variable during the period causes the output pulse to be extended or shortened. In this case (CYCLE < PER_TM_P), if the configured period is so long that it would cause the process value to fluctuate, the effective period of PID_CP is automatically reduced to an appropriate value.

Accuracy of pulse generation

The period PER_TM is not identical with the processing cycle of the pulse generator. A period is comprised of several processing cycles of the pulse generator. The number of calls of the pulse generator per period is a measure for the accuracy of the pulse width.

The smaller the sampling time CYCLE_P is compared to the period duration PER_TM_P (or PER_TM_N), the more precise the pulse width modulation is. To achieve sufficiently accurate control, the following condition should be met:

$$\text{CYCLE_P} \leq \frac{\text{PER_TM}}{20 \ldots 50}$$

Realization of very short sampling times

A fast process requires very small sampling times (for example, 10 ms). Due to the program runtime in this case, it is not appropriate to process the control functions in the same cyclic interrupt OB where the pulse output is calculated. You then transfer the processing of the control functions either to the OB 1 or to a slower cyclic interrupt OB. The processing of the control functions in OB 1 is only recommended if the cycle time of OB 1 is significantly less than the sampling time CYCLE of the controller.

Use the SELECT parameter to specify which program section is to be processed. The following table provides a quick overview of the parameter assignment of the SELECT input parameter:

SELECT	Block functionality used	Underlying method
0	Control functions and pulse output	Control functions and pulse output in one and the same block
1	Call in OB 1 (control functions)	Control functions in OB 1, pulse output in fast cyclic interrupt OB
2	Call in cyclic interrupt OB (pulse output)	
3	Call in slow cyclic interrupt OB (control functions)	Control functions in slow cyclic interrupt OB, pulse output in fast cyclic interrupt OB
2	Call in fast cyclic interrupt OB (pulse output)	

The methods for achieving very short pulse pattern widths indicated in the table above are described in more detail below.

- Control functions in OB 1, pulse output in cyclic interrupt OB
  When called with SELECT = 2, the pulse output is calculated and a test is performed to determine whether the sampling time configured in CYCLE has already elapsed since the last time the control functions were processed. If the sampling time has elapsed, PID_CP writes the value TRUE in the QC_ACT variable in the instance DB.
  When called with SELECT = 1, the QC_ACT variable is evaluated in the instance DB: If the value of QC_ACT is FALSE, the block is stopped immediately; thus, only a very brief execution time is required. If the value of QC_ACT is TRUE, the control functions are executed once and the QC_ACT is then reset.
  As a result of this procedure, the exact sampling time for the control functions cannot be adhered to. It fluctuates by the amount of the execution time of OB 1 (including all interruptions). This process is therefore only suitable when the execution time of OB 1 is less than the sampling time CYCLE.
- Control functions in slow cyclic interrupt OB, pulse output in fast cyclic interrupt OB
  When called with SELECT = 2, the pulse output is always calculated.
  When called with SELECT = 3, the control functions are always processed.

Note

To prevent having to program the PID_CP instruction call twice, along with its many formal operands, it is advisable to program this call once in an FC that also has a SELECT parameter as an input parameter. Interconnect this input parameter to PID_CP.SELECT. You then call only this FC in OB 1 or in the cyclic interrupt OB.

This procedure is easy to modify and saves program memory.

Commissioning PID_CP (S7-300, S7-400)

Performing tuning manually (S7-300, S7-400)

Requirements

- The instruction and the technology object have been loaded to the CPU.
- MAN_ON = FALSE

Procedure

In order to manually determine the optimal PID parameters, proceed as follows:

1. Click the "Measurement on" icon.
   An online connection will be established, if one is not already present. The current values for the setpoint, process value and manipulated variable are recorded.
2. Enter new PID parameters in the "P", "I", "D" and "Time delay" fields.
3. Click on the icon "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
   If the setpoint's rate of change is limited, the changeover to the new setpoint will be bumpless
6. Clear the "Manual mode" check box, if necessary.
   The controller works with the new PID parameters and controls the new setpoint.
7. Check the quality of the PID parameter based on the curve points.
8. Repeat steps 3 to 8 until you are satisfied with the controller results.

Commissioning with the PID Self Tuner (S7-300, S7-400)

You can commission the PID_CP instruction and determine the optimum PID parameters with the PID Self Tuner.

Procedure

To commission a continuous controller, proceed as follows:

1. Call the instruction PID_CP in a cyclic interrupt OB and interconnect the input and output parameters.
2. Call the instruction TUN_EC in the same cyclic interrupt OB.

3. Interconnect the instruction TUN_EC as follows with PID_CP:
   - TUN_EC.PV with PID_CP.PV
   - TUN_EC.LMN with PID_CP.LMN
   - TUN_EC.SP_OUT with PID_CP.SP_IN or PID_CP.SP_EXT
   - TUN_EC.GAIN with PID_CP.GAIN
   - TUN_EC.TI with PID_CP.TI
   - TUN_EC.TD with PID_CP.TD
   - TUN_EC.TM_LAG with PID_CP.TM_LAG
   - If you are using a three-step controller, TUN_EC.RATIOFAC with PID_CP.RATIOFAC
   - TUN_EC.QMAN_ON with PID_CP.MAN_ON
4. Click the "" icon on the instruction TUN_EC.
   The commissioning editor for TUN_EC opens. The operation of the commissioning editor is described with the PID Self Tuner.

3.5.1.2 Using PID_ES (S7-300, S7-400)

Technology object PID_ES (S7-300, S7-400)

The PID_ES technology object provides a step controller for automatic and manual mode. It corresponds to the instance data block of the instruction PID_ES. You can configure the following controllers:

- Fixed setpoint control with P, PI, PD, PID step controllers
- Step controllers with and without position feedback
- Fixed setpoint control with disturbance variable selection
- Cascade control with step controller in the follow-up circuit
- Two-loop ratio control
- Blending control

You configure the following functions of the technology object PID_ES:

- Scaling the process value signal
- Suppression of high-frequency vibrations in the process value branch by smoothing (time delay) of the process value signal.
- Linearization of the quadratic functions of the process value (flow-rate control with differential pressure encoders)
- Calling of "own functions" in the process value branch
- Monitoring the process value for violation of alarm and warning limits
- Monitoring the rate of change of the process value
- Adjustment of the setpoint by ramp/soak
- Scaling an external setpoint signal with ratio controllers

- Calling of "own functions" in the setpoint branch
- Limiting the rate of change of the setpoint
- Limiting the absolute values of the setpoint
- Filtering the control deviation signal with a deadband
- Monitoring the control deviation for violation of alarm and warning limits
- The proportional, integral, and derivative actions can be activated separately.
- Option of proportional and derivative effect in the controller feedback path.
- Selecting a disturbance variable
- Enabling position feedback
- Control of the manipulated variable in manual mode from a PG or HMI)
- Limiting the absolute values of the manipulated variable
- Taking into account actuator-specific times when generating the pulse

Control deviation generation (S7-300, S7-400)

Process value (S7-300, S7-400)

Scaling (PV_NORM) (S7-300, S7-400)

The "I/O process value ON" check box is used to specify which analog input variables are scaled to the physical unit of the process value.

- Activated: Input variable is the I/O process value (PV_PER)
- Disabled: Input variable is the internal process value (PV_IN)

In order to uniquely specify the scaling levels, you define the following four parameters:

- High input (NM_PIHR)
- Low input (NM_PILR)
- High output (NM_PVHR)
- Low output (NM_PVLR)

The parameters are preset depending on the sensor type.

The process value is calculated from the respective input value (PV_PER or PV_IN) according to the following formula:

MP4 = (PV_PER - NM_PILR) × (NM_PVHR - NM_PVLR) / (NM_PIHR - NM_PILR) + NM_PVLR

or

MP4 = (PV_IN - NM_PILR) × (NM_PVHR - NM_PVLR) / (NM_PIHR - NM_PILR) + NM_PVLR

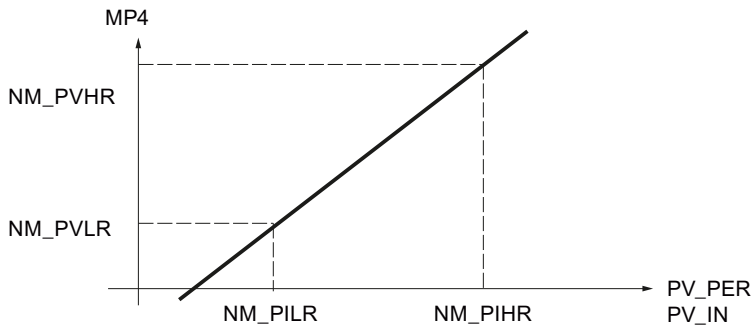

The process value is output in the static variable MP4.

No values are limited and the parameters are not checked. If you enter the same value for input high and input low, a division by zero may occur based on the formulas above. The instruction will not detect this.

Filter functions (S7-300, S7-400)

Smooth controlled variable (LAG1ST) (S7-300, S7-400)

In order to smooth process signals with superimposed rapid variations (noise), a first-order delay element is applied in the process value branch. This function stabilizes the analog controlled variable to a greater or lesser degree, depending on the time constant (PV_TMLAG). In this way, disturbances are effectively suppressed. As a result of this, however, the time constant of the overall control loop also increases, i.e., the closed-loop control becomes more sluggish.

The stabilizing effect is achieved by a first-order delay algorithm.

The step response within the time range is:

$outv(t) = MP4(0)\ (1 - e^{-t/PV_TMLAG})$

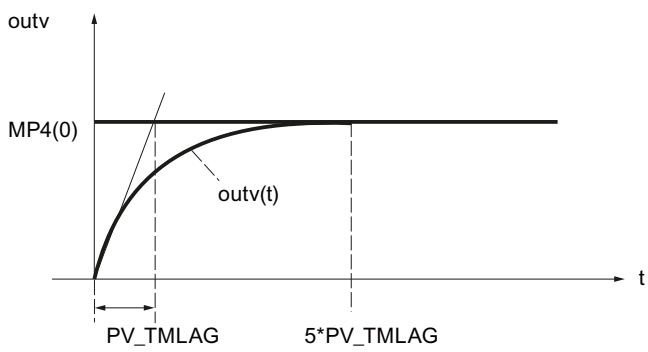

MP4(0)	The height of the process value step at the input
outv(t)	The output variable
PV_TMLAG	The time lag constant
t	Time

Smoothing of the process value

If smoothing of the process value (LAG1STON = FALSE) is not activated, then the I/O input (PV_PER) or the internal input (PV_IN) is switched through to the process value branch without a delay.

Time lag

The PV_TMLAG input parameter specifies the time lag for the process value. Any value within the value range is permitted.

If PV_TMLAG ≤ 0.5 * CYCLE, there is no longer any lag in effect.

A sampling time (CYCLE) of less than one-fifth of the time lag is required in order to achieve a lag response approaching that of the analog response.

Square root (SQRT) (S7-300, S7-400)

The "Square root" function (SQRT) is used to linearize the controlled variable. If the process value supplied by an encoder is a physical quantity and there is a quadratic relation between this quantity and the measured process variable, the characteristic of the controlled variable must be linearized prior to any further processing in the controller.

The measured signal must always be linearized by the square-root function if flow measurements are made using metering orifices, venturi tubes, or the like. The measured pressure differential is then proportional to the square of the flow.

The SQRT_ON = TRUE input signal is used to activate the square root function in the process value branch. The square root function algorithm has the following form:

outv = SQRT (MP5) × (SQRT_HR – SQRT_LR) / 10.0 + SQRT_LR

Here, outv is the scaled output value of the square root function (physical flow).

This formula stipulates that the input value of the square root is scaled to the number range 0 to 100. The parameters for the "Measuring range high" and "Measuring range low" must be set to 100.0 and 0.0 in the scaling of the process value. The square root of this value yields a measuring range from 0 to 10. This range of numbers will be scaled using the high limit (SQRT_HR) and the low limit of the physical measurement range (SQRT_LR).

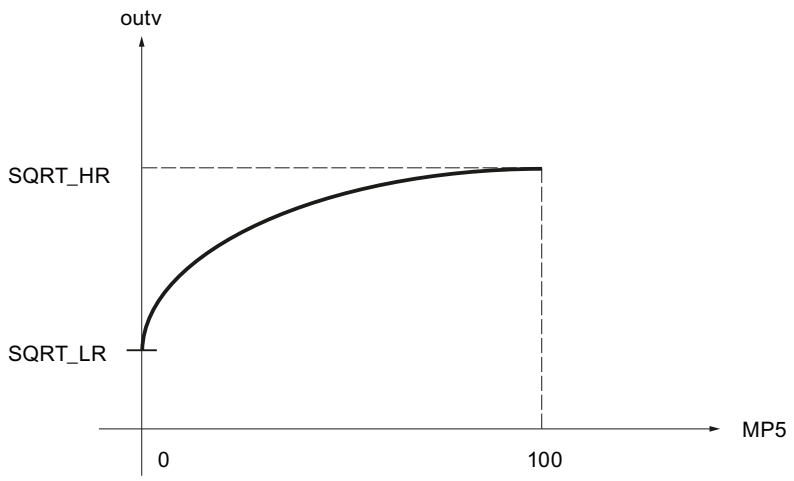

Example of scaling

The input value PV_IN of the controller is the differential pressure in mbar:

Start of measuring range NM_PILR	End of measuring range NM_PIHR	Sample value for PV_IN
20.0 mbar	200.0 mbar	150.0 mbar

The scaled differential pressure is calculated with the help of the scaling function PV_NORM, whereby NM_PVHR = 100.0 and NM_PVLR = 0.0 are selected:

MP4 = (PV_IN - NM_PILR) * (NM_PVHR - NM_PVLR) /
(NM_PIHR - NM_PILR) + NM_PVLR
= (PV_IN - 20.0 mbar) * (100.0 - 0.0) /
(200.0 mbar - 20.0 mbar) + 0.0
= (PV_IN - 20.0 mbar) * 100 / 180.0 mbar

Initial value of MP4	End value of MP4	Sample value for MP4
0.0	100.0	72.222

Since no smoothing is used in this example, the following applies: MP5 = MP4.

For the square root from the scaled differential pressure MP5, the result is:

Initial value according to the square root	End value according to the square root	Sample value for SQRT(MP5)
0.0	10.0	8.498

For the scaled output value outv of the square root function (physical flow-rate) with SQRT_HR = 20000.0 m^3/h und SQRT_LR = 0.0 m3/h, there follows:

outv = SQRT(MP5) * (SQRT_HR - SQRT_LR) / 10.0 + SQRT_LR
= SQRT(MP5) * (20000.0 m^3/h - 0.0 m3/h) / 10.0 + 0.0 m3/h
= 2000.0 m^3/h * SQRT(MP5)

Start of measuring range outv	End of measuring range outv	Sample value for outv
0.0 m^3/h	20000.0 m^3/h	16996.732 m^3/h

FC call in the process value branch (PVFC) (S7-300, S7-400)

Insertion of a user-specific function in the process value branch allows the controlled variable to undergo signal processing, e.g., signal delay or linearization, prior to any further processing in the controller.

Calling a function

If you want to insert a user-specific function into the process value branch, enable "Enable FC call" (PVFC_ON). The number of the utilized function is entered in the "FC number" input field (PVFC_NBR). The controller calls the function. In so doing, existing input/output parameters of the user function are not supplied. You have to program the data transfer in your function yourself.

You can monitor the output value of the function at the static variable MP6.

Note

No check is made to determine whether a function exists. If the function does not exist, the CPU switches to STOP with an internal system error.

Alarm (S7-300, S7-400)

Monitor process value for limit values (PV_ALARM) (S7-300, S7-400)

If the values of the process variables (e.g., speed, pressure, filling level, temperature, etc.) exceed or fall below certain values in closed-loop controls, an illegal process status or plant status can occur. In such cases, monitoring of the process value for limit values (PV_ALARM) is used in order to monitor the process value for violation of the permissible operating range. Limit violations are detected and signaled so that an appropriate reaction can be initiated.

Note

The monitoring of the process value for limit values cannot be deactivated. The violation of the configured limits is always signaled.

Monitoring of the process value for limit values (PV_ALARM) monitors the controlled variable PV(t) for four assignable limits in two tolerance bands. If the limits are reached or exceeded, the function signals a “warning” at the first limit and an “alarm” at the second limit.

Limit values

The numeric values of the limits are set in the input parameters for “warning” and “alarm”. If the process value (PV) exceeds or falls below these limits, the associated output message bits are set for the high alarm (QPVH_ALM), the high warning (QPVH_WRN), the low warning (QPVL_WRN), and the low alarm (QPVL_ALM).

During a restart, the message bits are reset.

The message bits are set as follows:

QPVH_ALM	QPVH_WRN	QPVL_WRN	QPVL_ALM	If:	And:
TRUE	TRUE	FALSE	FALSE	PV ↗ PV ↘	PV ≥ PVH_ALM PV ≥ PVH_ALM - PV_HYS
FALSE	TRUE	FALSE	FALSE	PV ↗ PV ↘	PV ≥ PVH_WRN PV ≥ PVH_WRN - PV_HYS
FALSE	FALSE	TRUE	FALSE	PV ↘ PV ↗	PV ≤ PVL_WRN PV ≤ PVL_WRN + PV_HYS
FALSE	FALSE	TRUE	TRUE	PV ↘ PV ↗	PV ≤ PVL_ALM PV ≤ PVL_ALM + PV_HYS

The following has to apply so that the block operates properly:

High alarm < High warning < Low warning < Low alarm

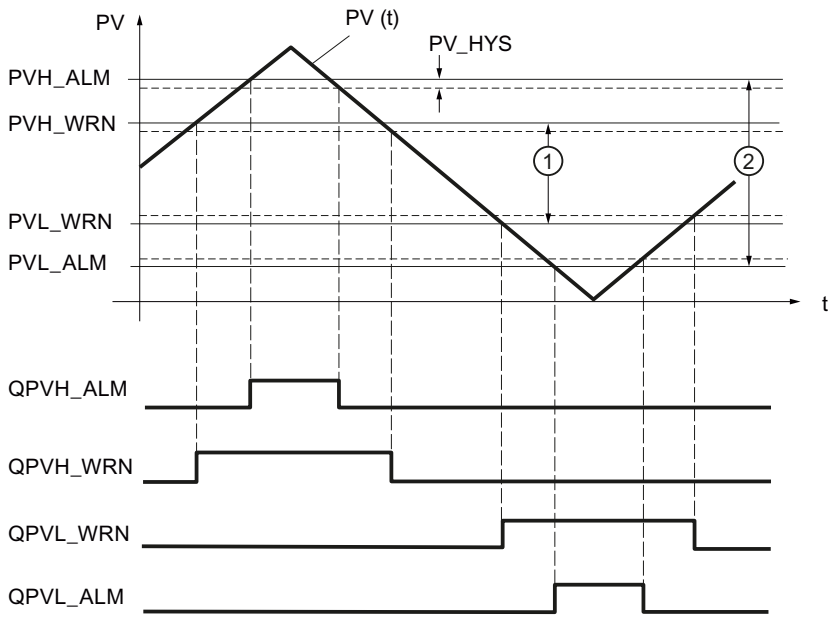

①	1. Tolerance band
②	2. Tolerance band

Hysteresis

In order to prevent "flickering" of the message bits when the input variable undergoes slight changes or in the event of rounding errors, a hysteresis PV_HYS is used. The controlled variable must violate the hysteresis before the message bits are reset.

Monitor rate of change of the process value (ROCALARM) (S7-300, S7-400)

If the change in the process variable (e.g., speed, pressure, filling level, temperature, etc.) is too great in closed-loop controls, an illegal process status or plant status can occur. In such cases, monitoring the rate of change of the process value (ROCALARM) is used to monitor the process value for violations of a permissible rate of change. Limit violations are detected and signaled so that an appropriate reaction can be initiated.

Note

Monitoring of the rate of change of the process value (ROCALARM) cannot be deactivated. The violation of the configured limits is always signaled.

The ROCALARM function monitors the controlled variable PV(t) for permissible rate-of-change limits that are assignable according to sign.

The numeric values of the limit slopes are set at the input parameters for "Maximum increase" and "Maximum decrease" in the positive and negative range of the controlled variable. The slopes refer to a rising slope or falling slope in percent per second. If the rate of change of the controlled variable violates these limits, the associated output message bit QPVURLMP ... QPVDRLMN is set.

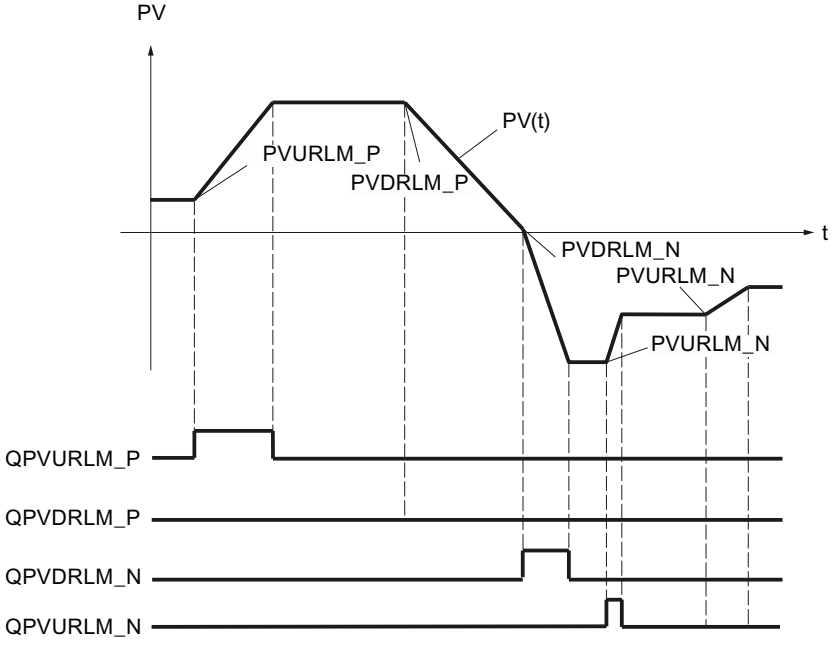

The ramp parameters are identified in accordance with the following scheme:

Parameter	PV change
Maximum increase in the positive range (PVURLM_P)	PV > 0 and |PV| rising
Maximum decrease in the positive range (PVDRLM_P)	PV > 0 and |PV| falling
Maximum increase in the negative range (PVURLM_N)	PV > 0 and |PV| rising
Maximum decrease in the negative range (PVDRLM_N)	PV < 0 and |PV| falling

Setpoint (S7-300, S7-400)

Specifying a setpoint (S7-300, S7-400)

Select a source for the setpoint. The following options are available:

- Internal setpoint
  The input parameter SP_INT is used as setpoint
- Setpoint generator
  The input parameter SP_INT is used as setpoint and can be incremented or decremented using the static variables SPUP and SPDN.
- Ramp/soak
  The input parameter SP_INT is used as setpoint and can be changed according to a specified ramp/soak.
- External setpoint
  The input parameter SP_EXT is used as setpoint for blending, ratio and cascade controllers. The external setpoint can be scaled.

The effective setpoint is shown at the output parameter SP.

See also

Ramp/soak (RMP_SOAK) (Page 863)

Setpoint generator (SP_GEN) (S7-300, S7-400)

The internal setpoint SP_INT can be incremented or decremented in steps using the SPUP and SPDN switches. The modified value can be monitored at MP1. The configured setpoint limits are taken into account.

Effect of the switches SPUP and SPDN

For changes involving small steps, the controller should have a maximum sampling time of 100 ms.

The rate of change of the output variable depends on how long the SPUP or SPDN switch is actuated and on the selected limits. The following relationship applies:

- During the first 3 s after setting SPUP or SPDN:
  doutv/dt = (SP_HLM - SP_LLM) / 100 s
- Afterwards:
  doutv/dt = (SP_HLM - SP_LLM) / 10 s

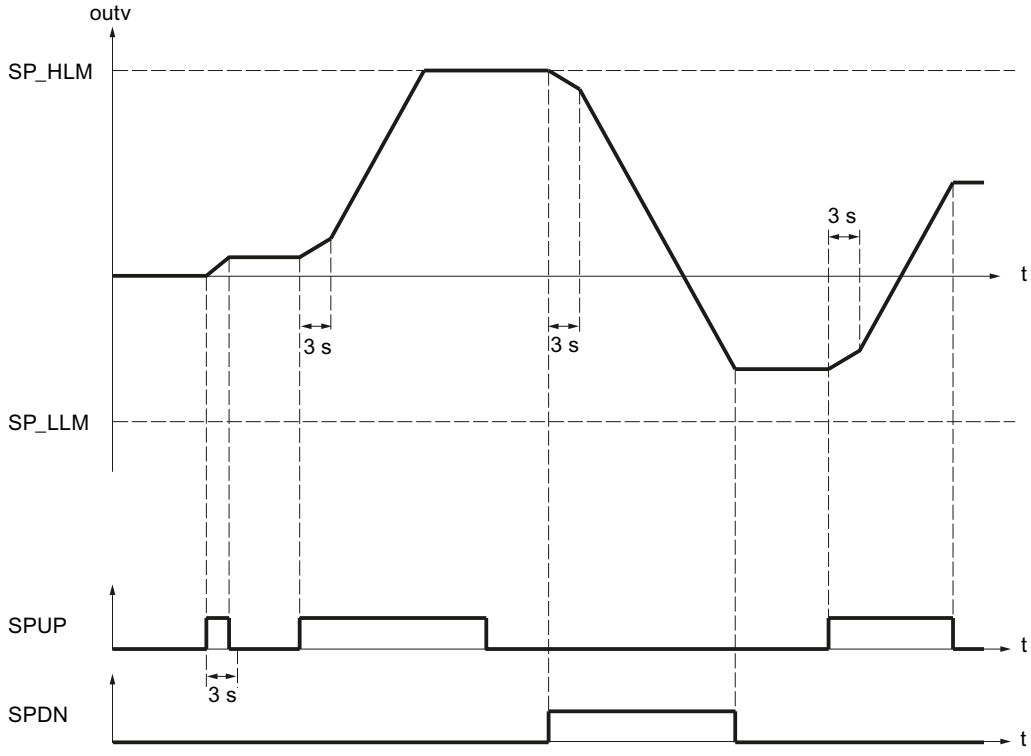

For a sampling time of 100 ms and a setpoint range of -100.0 to 100.0, the setpoint, for example, changes by 0.2 per cycle during the first 3 seconds. If SPUP is TRUE for more than 3 seconds, the rate of changes increases to ten times the value, in this case therefore to 2 per cycle.

Startup and operating principles of the setpoint generator

- During a complete restart, output outv is reset to 0.0.
- If you activate the setpoint generator (SPGEN_ON = TRUE), the signal value SPFC_IN is output initially at output outv. Therefore, the switchover to the setpoint generator from another operating mode is always smooth. Provided the SPUP and SPDN (up/down keys) are not activated, the signal value SPFC_IN is retained at the output.

Parameters of the SP_GEN function

Output parameter outv is an implicit parameter. It can be monitored with the configuration tool via measuring point MP1.

Parameter	Meaning	Permitted value range
SPFC_IN	FC input setpoint	Engineering value range
SP_INT	Internal setpoint	Engineering value range

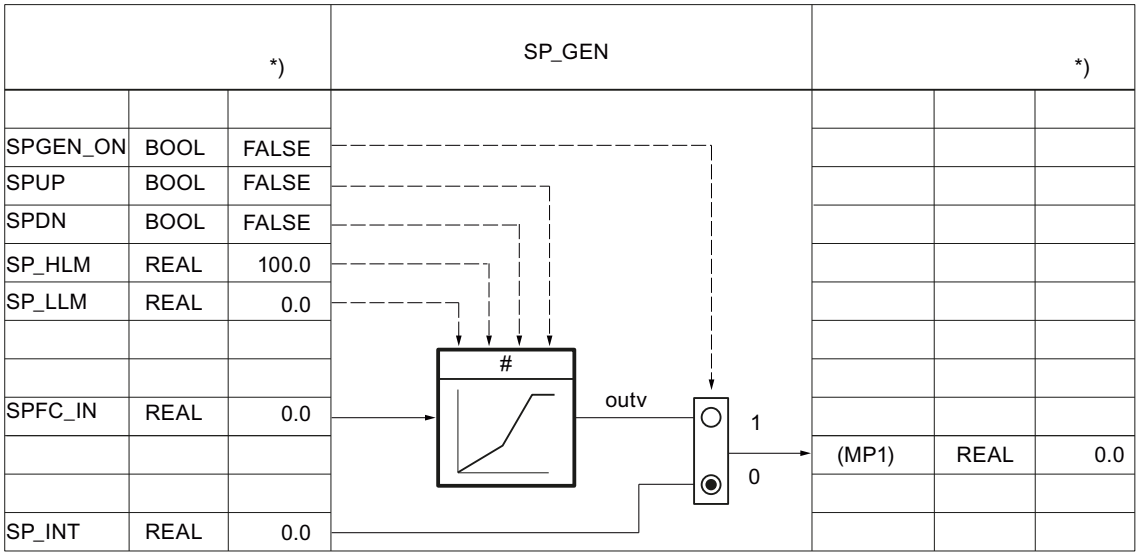

Figure 3-5 Function scheme and parameters of the setpoint generator

Ramp/soak (RMP_SOAK) (S7-300, S7-400)

If you want the setpoint SP_INT to change autonomously depending on the time, for example, during the control of processes after a time-controlled temperature program, this is possible by configuring a ramp/soak and activating the ramp/soak function. The setpoint change is formed from the level sections with a maximum of 256 time slices.

Note

The block function does not check whether a global DB with the number 'DB_NBR' exists or whether the parameter 'Number of time slices' NBR_PTS corresponds to the data block length. If the parameter assignment is incorrect, the CPU goes to **STOP** and issues an internal system error.

How the ramp/soak works

The ramp/soak supplies the output variable outv according to a function that runs to a defined ramp/soak. The ramp/soak process is defined by specifying a series of time slices in a global data block DB_RMPSK with time values PI[i].TMV and the associated output values PI[i].OUTV.

- PID-CP
  AUTOHOTSPOT is not included in the library. You must create this global data block yourself.
- PID-ES
  AUTOHOTSPOT is not included in the library. You must create this global data block yourself.

The following figure shows a ramp/soak with 6 time slices.

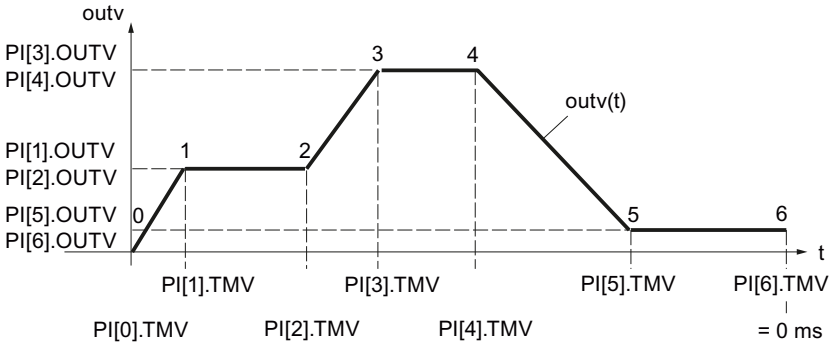

The time value for the last time slice n is PI[n].TMV = 0 ms. The processing time of a ramp/soak is counted down from the initial value to zero.

PI[i].TMV specifies the intervals between time slices. Between the time slices, interpolation is according to the following function:

$$\text{outv}(t) = \text{PI}[n + 1].\ \text{OUTV} - \frac{\text{RS_TM}}{\text{PI}[n].\text{TMV}}\ (\text{PI}[n + 1].\text{OUTV} - \text{PI}[n].\text{OUTV})$$

$0 \leq n \leq (\text{NBR_PTS} - 1)$

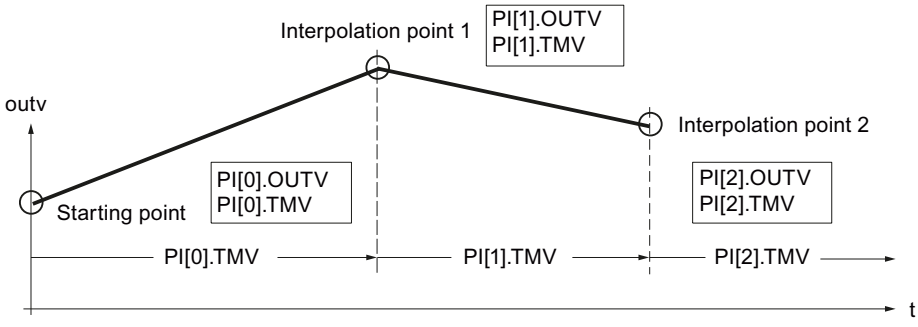

Note

Note

During the interpolation of the ramp/soak between the time slices, pauses can sometimes occur at the initial value if the sampling time CYCLE is very small compared to the time between the time slices PI[n].TMV. Due to the computational accuracy of the CPU, the ramp/soak function cannot linearize all flat ramp/soak profiles. If the ramp/soak profile is too flat, the initial value pauses for a time at the respective time slice and continues, after a certain time, to integrate with the minimum rate of rise to the next time slice.

To correct or avoid error: Shorten the time between the time slices by inserting additional time slices. The output ramp/soak profile thus approximates the desired flat ramp/soak profile.

Ramp/soak function ON

The ramp/soak function is activated by changing RMPSK_ON from FALSE to TRUE. The ramp/soak is terminated after the last time slice has been reached. When the function is started again by the operator, RMPSK_ON first has to be set to 'FALSE' and then back to 'TRUE' .

During a **restart**, the output outv is reset to 0.0 and the total time or total remaining time is determined. During the transition to normal operation, the ramp/soak profile is processed immediately from the starting point according to the set operating mode. If this is not desired, the parameter RMPSK_ON must be set to FALSE in the restart OB.

Ramp/soak functions

The instruction has the following functions:

- Activate ramp/soak for single execution
- Pre-assign fixed value to the output
- Cyclic repetition on
- Hold ramp/soak
- Continue ramp/soak
- Update the total processing time and remaining total time

The following table applies for the significance of the control inputs when setting a desired operating mode:

Operating mode	RMPSK_ ON	DFOUT_ ON	RMP_ HOLD	CONT_ ON	CYC_ ON	TUPDT_ ON	Output signal OUTV
Ramp/soak on	TRUE	FALSE	FALSE	**)	FALSE	**)	OUTV(t) End value is held after the end of processing.
Set output	TRUE	TRUE	**)	**)	**)	**)	DF_OUTV
Cyclic repetition on	TRUE	FALSE	FALSE	**)	TRUE	**)	OUTV(t) After end: Automatic start
Hold ramp/soak	TRUE	FALSE	TRUE	FALSE	**)	**)	Current value of OUTV(t) is held *)

Operating mode	RMPSK_ ON	DFOUT_ ON	RMP_ HOLD	CONT_ ON	CYC_ ON	TUPDT_ ON	Output signal OUTV
Continue ramp/soak	TRUE	FALSE	TRUE	TRUE	**)	**)	OUTV (old) *)
			FALSE				Operation is continued with newly defined values.
Updating the total time and remaining total time	**)	**)	**)	**)	**)	FALSE	Does not influence OUTV
						TRUE	Does not influence OUTV
*) The ramp/soak profile does not have the form configured by the user until the next time slice. **) The respective operating mode set is executed irrespective of the significance of the control signals in the fields marked with **)							

Set output of ramp/soak

If you enable this function (DFRMP_ON = TRUE), the output value of the ramp/soak is set to signal value SP_INT or output value SP_GEN.

Note

Preassignment with the internal setpoint has an effect only if the ramp/soak function is enabled (RMPSK_ON = TRUE).

After the switchover to DFRMP_ON = FALSE, outv moves linearly from the specified setpoint (e.g., SP_INT) to the output value of the current time slice number PI[NBR_ATMS].OUTV.

The internal time processing continues running even when a fixed setpoint (RMPSK_ON = TRUE and DFRMP_ON = TRUE) is switched through.

On the start of the ramp/soak profile (RMPSK_ON = TRUE), the fixed setpoint SP_INT continues to be output until DFRMP_ON changes from TRUE to FALSE after the duration of T*.

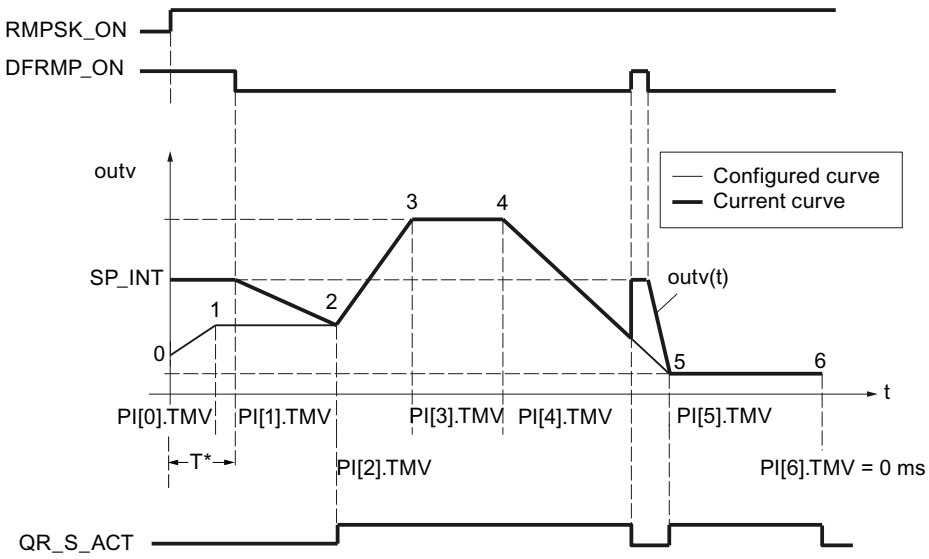

At this moment, the time PI[0].TMV and a part of the time PI[1].TMV has expired. OUTV is moved from SP_INT to PI[2].OUTV, which means to the setpoint at time slice 2.

The configured ramp/soak profile is output only starting from time slice 2. The agreement of the configured and current ramp/soak profile is displayed by setting QR_S_ACT = TRUE. On a rising edge at DFRMP_ON during processing of the ramp/soak profile, the output value OUTV jumps to SP_INT without a delay.

Cyclic repetition on

If "Cyclic repetition" operating mode (CYC_ON = TRUE) is activated, the ramp/soak function automatically returns to the starting point after the last time slice value has been output and begins a new run.

There is no interpolation between the last time slice and the starting point. For a smooth transition, the following must be true: PI[NBR_PTS].OUTV = PI[0].OUTV.

Hold ramp/soak

When RMP_HOLD = TRUE, the value of the output variable (including the time processing) is frozen. When RMP_HOLD = FALSE is reset, the function resumes at the interruption point PI[x].TMV.

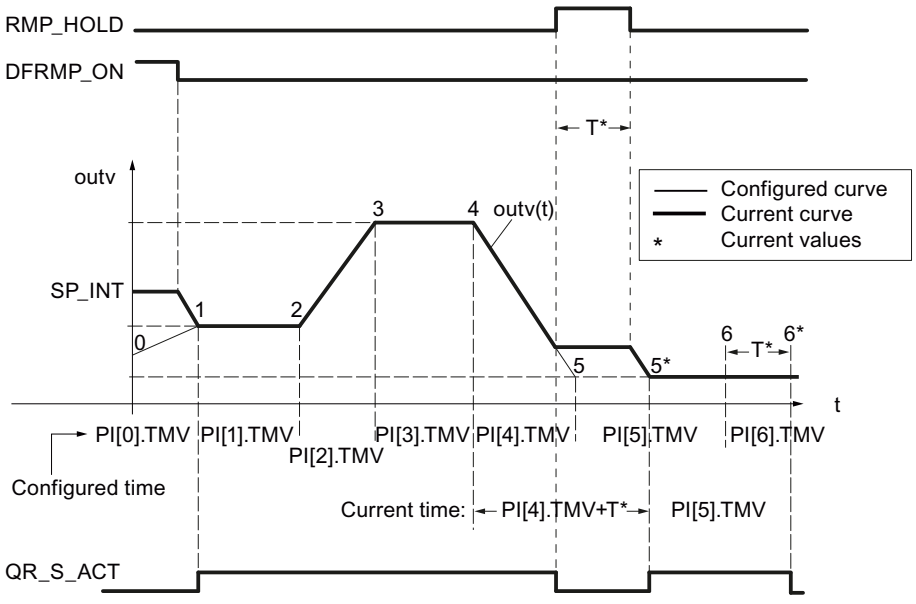

The processing time of the ramp/soak profile is extended by the amount of the time T*. The ramp/soak profile has the configured course from the starting point until the rising edge at RMP_HOLD and from time slice 5* to time slice 6*, i.e., output signal QR_S_ACT has the value TRUE.

If switch CONT_ON is set, the held ramp/soak function continues processing the ramp/soak profile at preassigned point TM_CONT.

Continue ramp/soak

With CONT_ON, a held ramp/soak profile is restarted at a defined point. If the ramp/soak function is not held, CONT_ON has no effect.

If the control input for continuing (CONT_ON = TRUE) is set, the held ramp/soak function is continued at time slice TM_SNBR. TM_CONT determines the remaining time that the ramp/soak function requires to reach time slice TM_SNBR.

The following figure shows the effect of RMP_HOLD and CONT_ON when the configuration is as follows:

Continue at time slice 5 (TM_SNBR = 5) and continue in T* s: TM_CONT = T*

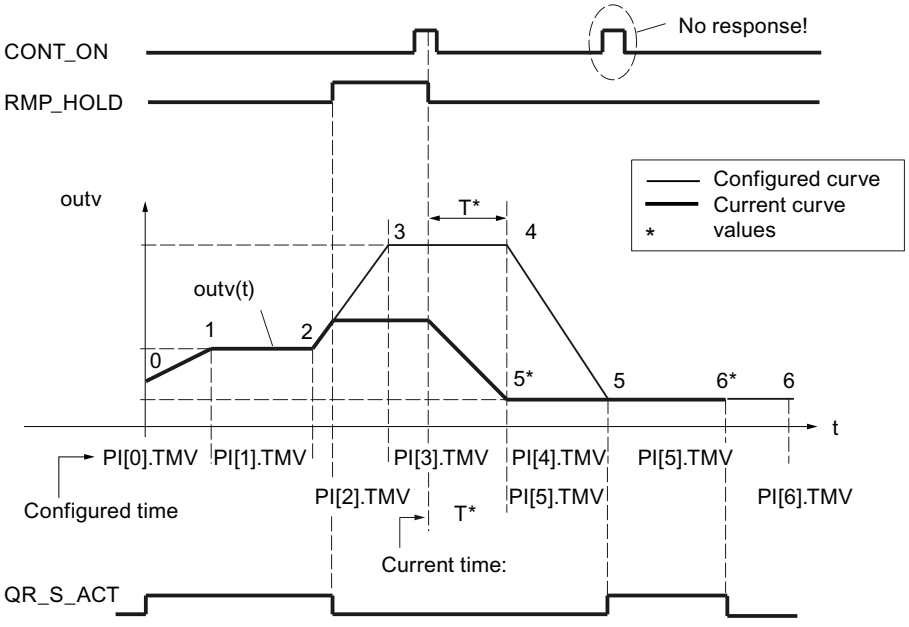

Time slices 3 and 4 are omitted in this processing cycle of the ramp/soak function. On a signal change of RMP_HOLD from TRUE to FALSE, the configured curve progression is only reached again starting from time slice 5.

Output QR_S_ACT is only set when the ramp/soak function executes the ramp/soak profile assigned by the user.

Updating the total time and remaining total time

The current time slice number NBR_ATMS, the current remaining time until the time slice is reached (RS_TM), the total time T_TM, and the total remaining time until the end of the ramp/soak profile (RT_TM) are updated in every cycle.

In the case of online changes of PI[n].TMV, the total time and the total remaining time of the ramp/soak profile change. Since the calculation of T_TM and RT_TM with many time slices significantly increases the processing time of the function block, the calculation is only carried after a restart or when TUPDT_ON = TRUE. The time intervals PI[0 ... NBR_PTS].TMV between the individual time slices are added up and indicated at the total time T_TM and total remaining time RT_TM outputs.

Please note that determining the total times requires significant CPU runtime.

Procedure

To configure the ramp/soak function, proceed as follows:

1. Select "Ramp/soak" from the "Setpoint source" drop-down list.
2. Create a AUTOHOTSPOT (PID-CP) or AUTOHOTSPOT (PID-ES).
3. Define the required parameters for the ramp/soak.
4. Enter the number of the data block DB_RMPSK in the "DB number" input field.
5. Enable or disable the additional functions of the ramp/soak.
6. If you have enabled the Continue ramp/soak function, enter at which time slice and when the ramp/sock is to be continued.

Scaling (S7-300, S7-400)

Scaling of an external setpoint

If the external setpoint is not expressed in the physical unit of the process value (e.g., expressed as a percent in the case of cascade control), then this value and its setting range must be scaled to the physical unit of the process value. The "Scaling in setpoint branch" function is used for this.

The SP_NORM function scales an analog input variable. The analog external setpoint is transformed to output variable outv using the scaling level. Output value OUTV is accessible in the configuration tool via measuring point MP2.

The output value of the function has an effect when control input SPEXT_ON = TRUE. In order to uniquely specify the scaling levels, you define the following four parameters:

- High input: SP_EXT: NM_SPEHR
- Low input: SP_EXT: NM_SPELR
- High output: NM_PVHR (You specify this value in the scaling function of the process value.)
- Low output: NM_PVLR (You specify this value in the scaling function of the process value.)

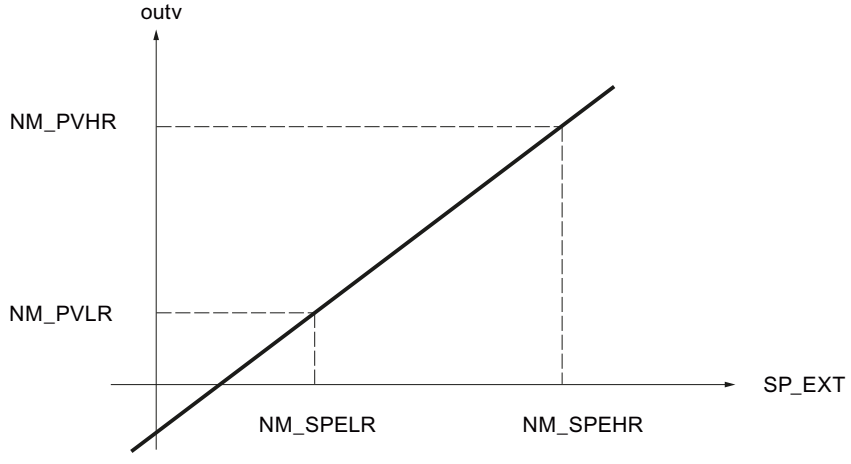

Output value outv is calculated from the respective input value SP_EXT according to the following formula:

outv = (SP_EXT – NM_SPELR) x (NM_PVHR – NM_PVLR) / (NM_SPEHR – NM_SPELR) + NM_PVLR

For the special case in which a square root function is activated in the process value branch, the scaling values of the square root function (SQRT_HR and SQRT_LR) apply as the high and low limits of output value outv.

In this case, output value outv of the scaling function is calculated from the respective input value SP_EXT according to the following formula:

outv = (SP_EXT – NM_SPELR) x (SQRT_HR – SQRT_LR) / (NM_SPEHR – NM_SPELR) + SQRT_LR

There is no internal value limiting in the function. The parameters are not checked. If you enter the same value for NM_SPEHR and NM_SPELR, a division by zero is possible based on the formulas above. The instruction will not detect this.

Combination/ratio factor

With ratio control, the setpoints of the control loop are coordinated with one another in such a way that the set ratio matches the actual ratio of the process values.

With blending control, the desired quantities of the individual components are set. The total of the blending factors FAC must be 1.

Procedure

To configure the scaling, proceed as follows:

1. Enter the values for the scaling in the input fields "High input" and "Low input".
2. Enter a blending/ratio factor (FAC).

Limiting functions (S7-300, S7-400)

FC call in the setpoint branch

Insertion of a user-specific function in the setpoint branch allows an externally specified reference variable to undergo signal processing (e.g., signal delay or linearization) prior to application in the controller.

If you activate the FC call in the setpoint branch, a user-specific function is called. The controller calls the FC. In so doing, existing input/output parameters of the user FC are not initialized. Therefore, you must use S7-STL to program the data transfer in the user FC. You can monitor the prepared setpoint at MP3.

The value of SPFC_ON then determines whether a user-programmed function in the form of a standard FC (e.g., a characteristic curve) will be inserted at this point in the setpoint channel or whether the setpoint will continue to be processed without any influence of this type.

Note

The statement does not check to determine if a function exists. If the function does not exist, the CPU switches to STOP with an internal system error.

Procedure

To call a function in the setpoint branch, proceed as follows:

1. Select the "Enable FC call" check box.
2. In the "FC" input field, enter the number of the function to be called.

Limiting of the rate of change of the setpoint

Ramp functions are used in the setpoint branch in situations where the process does not tolerate abrupt changes in the actuating signal, since an abrupt setpoint change generally also causes an abrupt change in the manipulated variable of the controller. These abrupt manipulated variable changes must be avoided, for example, when a gearbox is connected in between an adjustable-speed motor and the load to be driven and an excessively rapid increase of the motor speed would result in overloading of the gearbox.

The SP_ROC function limits the rate of change of the setpoints processed in the controller. This is carried out separately for the rising slope and falling slope and separately for the positive and negative range of the reference variable.

The limiting for each slope of the ramp function in the positive range and in the negative range of the reference variable is entered in the four inputs SPURLM_P, SPDRLM_P, SPURLM_N, and SPDRLM_N. The slopes refer to a rising or falling slope per second. Faster setpoint changes are ramped down to these maximum rates.

For example, if SPURLM_P is assigned the value 10.0 [engineering value range/s], the following values are added to the "old value" of outv in each sampling cycle for as long as inv > outv:

Sampling time	1 s	$outv_{old}$ + 10
	100 ms	$outv_{old}$ + 1
	10 ms	$outv_{old}$ + 0.1

The following figure shows an example of how the signal processing functions. The step functions at input inv(t) become ramp functions at output outv(t).

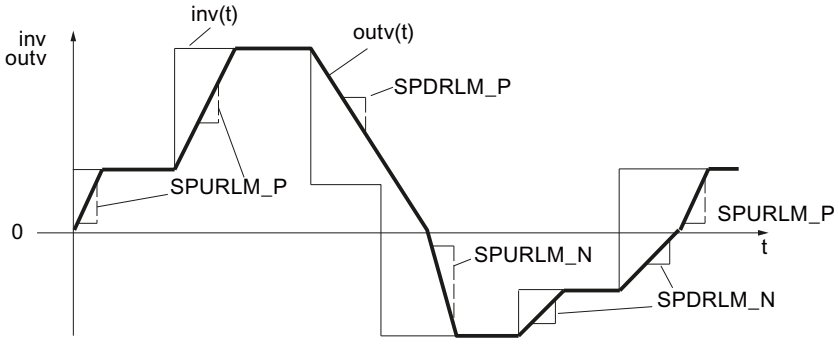

Figure 3-6 Limiting of the rate of change of the reference variable SP(t)

When slope limits are reached, this is not signaled.

Procedure

To limit the rate of change, proceed as follows:

1. Activate the "Limit rate of change" check box.
2. Specify the following values for the positive range as well as for the negative range:
   - Rising slope for setpoint value
   - Falling slope for setpoint value

Limiting of the absolute value of the reference variable

The setting range of the reference variable determines the range in which the controlled variable can fluctuate, i.e., the range in which the process varies within the scope of permissible status values.

To avoid a critical or impermissible process status, the setting range of the reference variable is subject to high and low limits (SP_LIMIT) in the setpoint branch of standard PID control.

The SP_LIMIT function limits the setpoint SP at the assignable low and high values (SP_LLM and SP_HLM) as long as input variable INV is outside these limits. Because the function cannot be deactivated, the specification of a low limit and high limit must be taken into consideration in the configuration.

The numeric values of the limits are set in the input parameters for the low and high limits. In the event of violations by input variable inv(t), the associated displays are output via the message outputs.

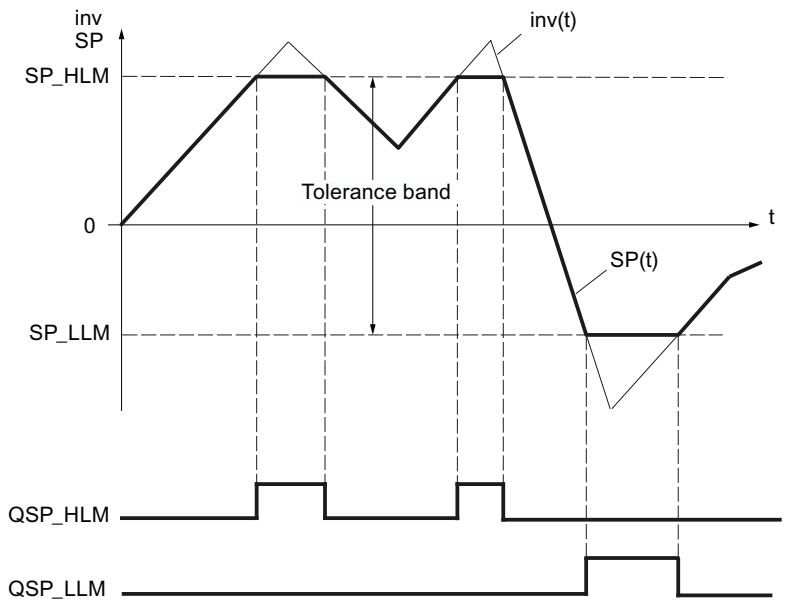

Figure 3-7 Absolute value limiting of reference variable SP(t)

During a restart, the message bits are reset.

The message bits are set as follows:

SP =PVH_ALM	QSP_HLM	QSP_LLM	If:
SP_HLM	TRUE	FALSE	inv ≥ SP_HLM
SP_LLM	FALSE	TRUE	inv ≤ SP_LLM
inv	FALSE	FALSE	SP_HLM < inv < SP_LLM

Procedure

To limit the absolute value of the reference variable, proceed as follows:

1. Enter the high limit for the absolute setpoint (SP_HLM) in the "High limit" input field.
2. Enter the low limit for the absolute setpoint (SP_LLM) in the "Low limit" input field.

Control deviation (S7-300, S7-400)

Deadband (S7-300, S7-400)

In an optimally tuned controller, when a high-frequency interfering signal is superimposed on the control variable or reference variable, the noise portion will also have an effect on the controller output. This can result in strong fluctuations of the manipulated variable, for example, if the controller gain is high and derivative action is enabled. Due to the increased switching frequency (step controller), this causes increased wear of the actuator.

The function reduces the noise portion in the control deviation signal while the controller is in steady-state and thus reduces undesired oscillating of the controller output.

The deadband can be disabled.

Signal filtering through deadband function

The DEADBAND function suppresses small fluctuations of the input variable around a specified zero point within an adjustable range. Outside this fluctuation range, the control deviation ER rises or falls proportional to the input variable. The width of the deadband can be specified with the DEADB_W parameter. Only positive values are permitted for the deadband width.

If the input variable is within the deadband, the value 0 (control deviation = 0) is output at the output. Only when the input variable exits the deadband will the output rise or fall by the same values as input variable inv. This results in corruption of the transmitted signal outside the deadband, too. However, this is accepted in order to avoid step changes at the deadband limits (see figure below). The corruption corresponds to the value DEADB_W and can therefore be controlled easily.

The DEADBAND function operates according to the following functions:

(ER) = inv + DEADB_W for inv < - DEADB_W

(ER) = 0 for -DEADB_W ≤ inv ≤ + DEADB_W

(ER) = inv - DEADB_W for inv > + DEADB_W

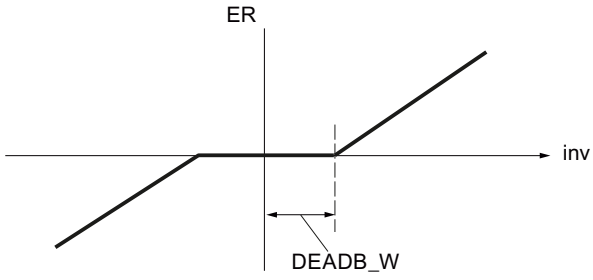

Effects of the signal filtering can be monitored at the output parameter ER.

Alarm (S7-300, S7-400)

Excessive deviations of the controlled variable from the specified setpoint can cause an undesired process status. In such cases, the ER_ALARM function is used to monitor the control deviation for violation of the permissible operating range. ER_ALARM detects and signals any limit violations that occur so that an appropriate reaction can be initiated.

Note

The monitoring of the control deviation for limit values cannot be disabled. The violation of the configured limits is always signaled.

The ER_ALARM function monitors the size of control deviation ER(t) for four assignable limits in two tolerance bands. If the limits are reached or exceeded, the function signals a “warning” at the first limit and an “alarm” at the second limit.

Limit values

The numeric values of the limits are set in the input parameters for “warning” and “alarm”. If the control deviation (ER) violates these limits, the associated output message bits QERN_ALM ... QERP_ALM are set.

During a restart, the message bits are reset.

The message bits are set as follows:

QERP_ALM	QERP_WRN	QERN_WRN	QERN_ALM	If:	And:
TRUE	TRUE	FALSE	FALSE	ER ↗ ER ↘	ER ≥ ERP_ALM ER ≥ ERP_ALM - ER_HYS
FALSE	TRUE	FALSE	FALSE	ER ↗ ER ↘	ER ≥ ERP_WRN ER ≥ ERP_WRN - ER_HYS
FALSE	FALSE	TRUE	FALSE	ER ↘ ER ↗	ER ≤ ERP_WRN ER ≤ ERP_WRN + ER_HYS
FALSE	FALSE	TRUE	TRUE	ER ↘ ER ↗	ER ≤ ERP_ALM ER ≤ ERP_ALM + ER_HYS

The following has to apply so that the block operates properly:

High alarm < High warning < Low warning < Low alarm

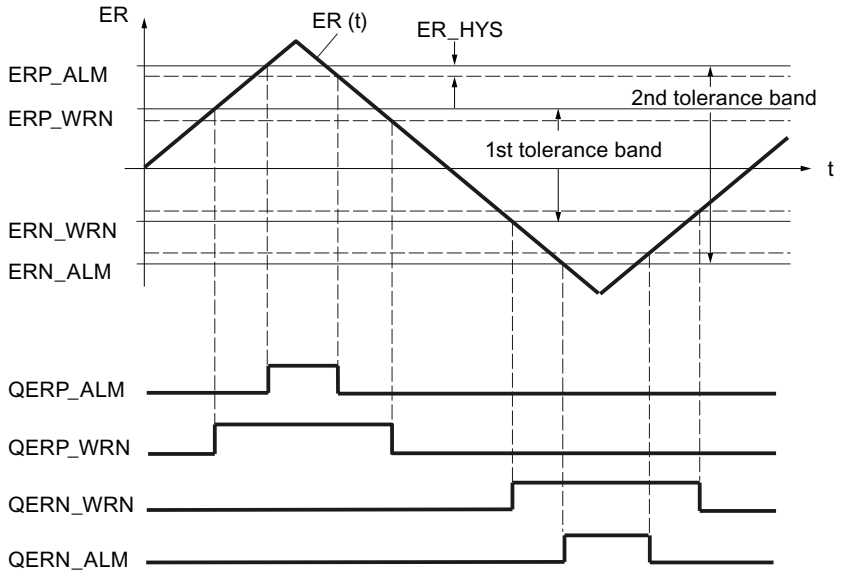

Hysteresis

In order to avoid "flickering" of the message bits when the input variable undergoes slight changes or in the event of rounding errors, a hysteresis ER_HYS is used. The control deviation must violate the hysteresis before the message bits are reset.

Controller (S7-300, S7-400)

Control algorithm (S7-300, S7-400)

The manipulated variable of the continuous controller is calculated within the configured sampling time cycle on the basis of the control deviation in the PID position algorithm. The controller is implemented in a purely parallel structure. The proportional, integral, and derivative actions can each be disabled individually.

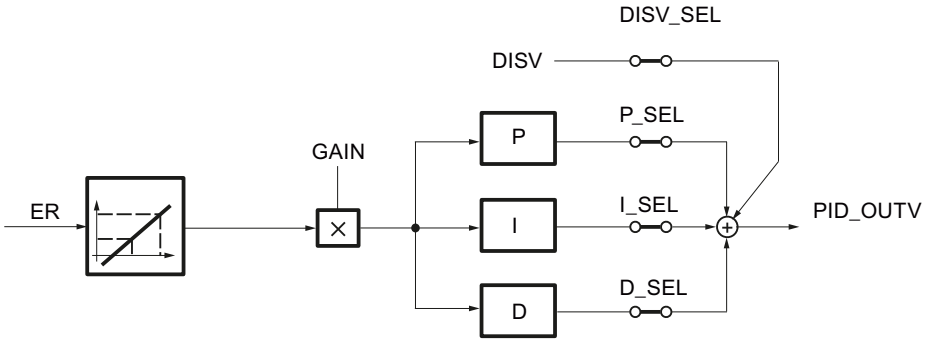

Feedforward control

A disturbance variable DISV can be added in the output signal of the controller (PID_OUTV). The variable is switched-in or switched-off in the PID window of the configuration tool using the DISV_SEL structure selector or through "Disturbance variable on".

Proportional and derivative action in the feedback path

In the parallel structure, each action in the control algorithm receives the control deviation as input signal. Via the proportional action and derivative action, the manipulated variable is influenced directly by setpoint steps. However, a different controller structure in which the formation of the proportional and the derivative actions is moved to the feedback path ensures a smooth response of the manipulated variable to setpoint step changes. In this structure, the integral action processes the control deviation as the input signal. Only the negative controlled variable (factor = - -1) is applied to the proportional action and the derivative action.

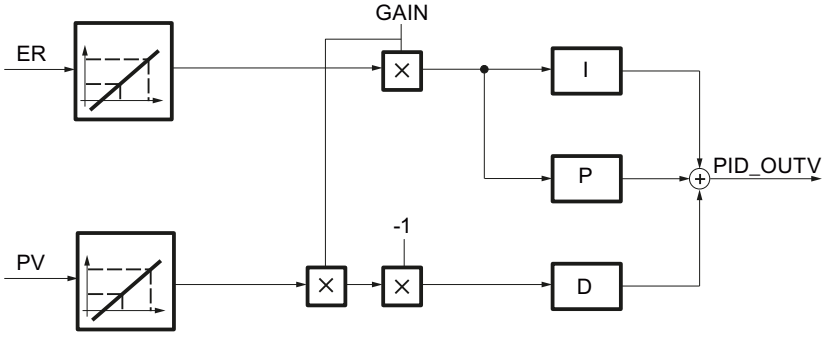

Scaling the input variable ER and PV

The input variables ER and PV of the PID controller are scaled to the range 0 to 100 prior to the controller processing according to the following formulas:

- with square-root function disabled (SQRT_ON = FALSE):
  - ER_{scaled} = ER * 100.0 / (NM_PVHR – NM_PVLR)
  - PV_{scaled} = (PV - NM_PVLR) * 100.0 / (NM_PVHR – NM_PVLR)
- with square-root function enabled (SQRT_ON = TRUE):
  - ER_{scaled} = ER * 100.0 / (SQRT_HR – SQRT_LR)
  - PV_{scaled} = (PV - SQRT_LR) * 100.0 / (SQRT_HR – SQRT_LR)

This scaling is performed so that the proportional gain GAIN of the PID controller can be entered without dimensions. If the high limit and low limit of the physical measuring range change (for example, from bar to mbar), the gain factor does not then have to be changed.

The scaled input variables ER_{scaled} and PV_{scaled} cannot be monitored.

See also

Controller structures (Page 877)

Proportional component (Page 881)

Derivative component (Page 884)

Integral action (Page 881)

Controller structures (S7-300, S7-400)

Use and parameter assignment of the PID controller

The majority of the controlled systems occurring in the process industry can be controlled with the PI controller function/PID controller function of standard PID control. Only in special cases are additional methods and measures needed for a particular closed-loop control.

However, a major problem in the field continues to be how to assign the PI controller/PID controller parameters, i.e., how to determine the "correct" settings for the controller parameters. The quality of this parameter assignment is critical in order for the PID control to meet the requirements of the task at hand. It requires either a lot of practical experience, special expertise, or a great amount of time.

P control

The integral and derivative actions are disabled in a proportional-action controller (P-controller). (I_SEL and D_SEL = FALSE). This means that when control deviation ER = 0, output signal OUTV = 0. If an operating point ≠ 0 is intended, i.e., a numeric value will be set for the output signal when the control deviation is zero, the integral branch must be activated (see the following figure).

In the integral action, an operating point ≠ 0 can be specified for the P-controller via a corresponding setting of initialization value I_ITLVAL. Set switches 'I_ITL_ON' and 'I_SEL' = TRUE for this purpose.

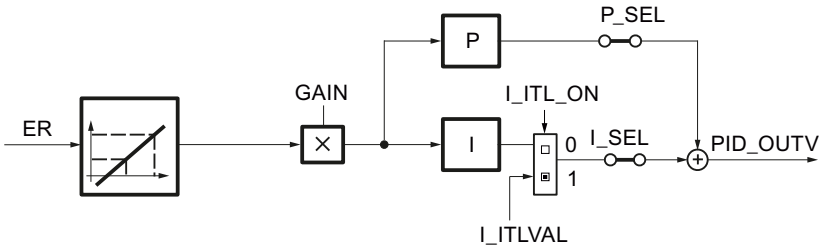

The step response of the P-controller within the time range is:
PID_OUTV (t) = I_ITLVAL + GAIN * ER $_{scaled}$ (t)

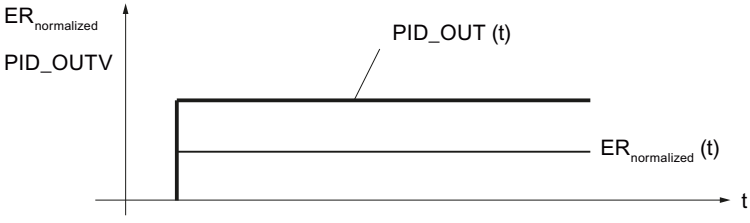

PID_OUTV(t)	Manipulated variable when controller is in automatic mode
I_ITLVAL	Operating point of the P-controller
GAIN	Controller gain
ER $_{scaled}$ (t)	Scaled control deviation

PI control

The derivative action is disabled in the proportional-integral controller (PI controller). (D_SEL = FALSE). A PI controller adjusts output variable PID_OUTV using the integral action until control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the control range.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{1}{TI} * t\right)$$

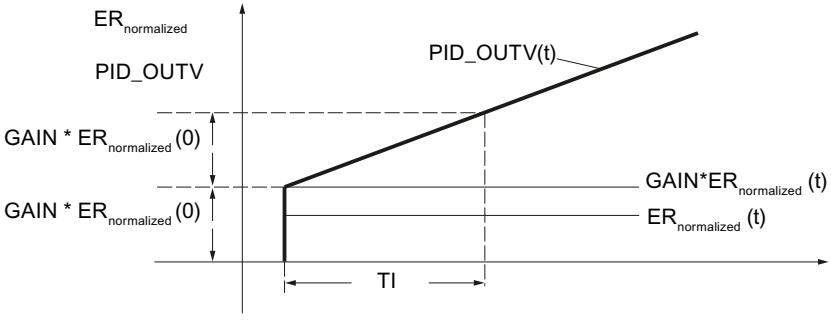

PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TI	Integral-action time constant

PD control

The integral action is disabled in the proportional-derivative controller (PD-controller) (I–SEL = FALSE). This means that when control deviation ER = 0, output signal OUTV = 0. If an operating point ≠ 0 is intended, i.e., a numeric value will be set for the output signal when the control deviation is zero, the integral action branch must be activated (see also the block diagram of the P control).

In the integral action, an operating point ≠ 0 can be specified for the PD-controller via a corresponding setting of initialization value I_ITLVAL. The switches 'I_ITL_ON' and 'I_SEL' must be set to TRUE for this purpose.

The PD-controller maps input variable ER(t) proportionally to the output signal and adds to this the derivative action generated by differentiation of ER(t); the derivative action is calculated with double precision according to the trapezoid rule (Padé approximation). The time response is determined by the derivative time constant TD.

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time: TM_LAG) is integrated into the algorithm for generating the derivative action. A small value for TM_LAG is usually sufficient to achieve the desired result. If TM_LAG ≤ CYCLE/2 is configured, the time delay is deactivated.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{TD}{TM_LAG} * e^{-\frac{t}{TM_LAG}}\right)$$

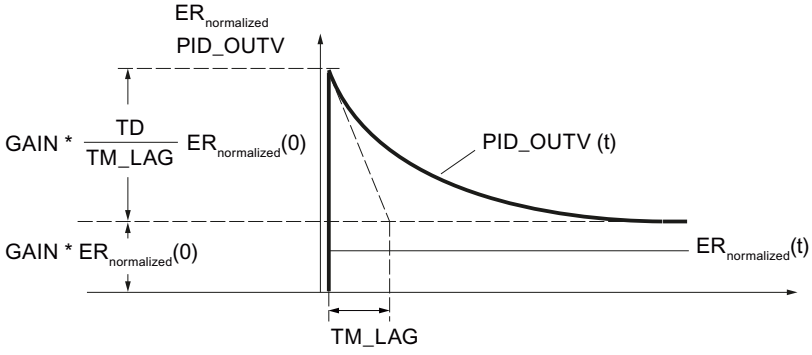

PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TD	Derivative time
TM_LAG	Time delay

PID control

In the proportional-integral-derivative controller (PID-controller), the proportional action, integral action, and derivative action are enabled (P–SEL = TRUE, I–SEL = TRUE, D_SEL = TRUE). A PID controller adjusts output variable PID_OUTV using the integral action until control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the control range. If manipulated variable limits are exceeded, the integral action maintains the value that was reached at the limit (anti-reset windup).

The PID-controller maps scaled input variable $ER_{scaled}(t)$ proportionally to the output signal and adds to this the components generated by differentiation and integration of $ER_{scaled}(t)$; these components are calculated with double precision according to the trapezoid rule (Padé approximation). The time response is determined by the derivative-action time TD and integral-action time TI.

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time constant: TM_LAG) is integrated into the algorithm for generating the derivative action. A small value for TM_LAG is usually sufficient to achieve the desired result. If TM_LAG ≤ CYCLE/2 is configured, the time delay is deactivated.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0) \left(1 + \frac{1}{TI} * t + \frac{TD}{TM_LAG} * e^{-\frac{t}{TM_LAG}} \right)$$

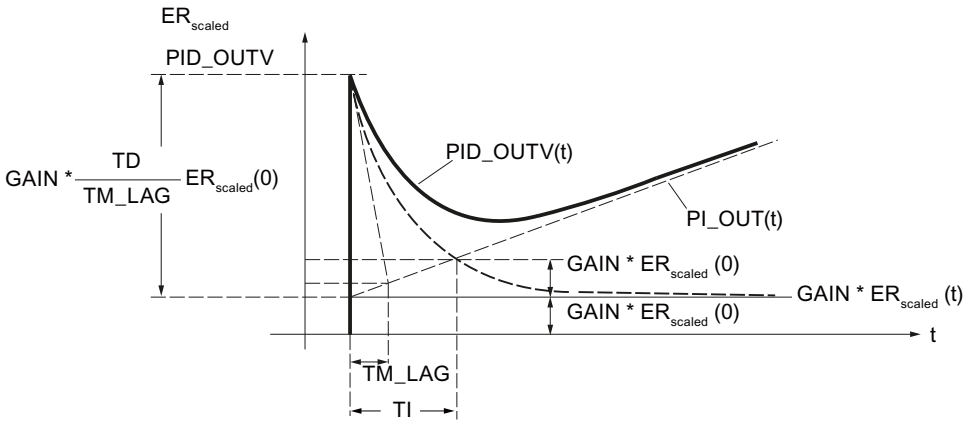

PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TI	Integral-action time constant
TD	Derivative time
TM_LAG	Time delay

Note

If TD is changed, you should also adjust TM_LAG accordingly.

Recommendation: 5 ≤ (TM / TM_LAG) ≤ 10

See also

Control algorithm (Page 876)

Proportional component (Page 881)

Derivative component (Page 884)

Proportional component (S7-300, S7-400)

Proportional gain

The sign of this proportional gain determines the effective direction of the continuous controller.

- Positive proportional gain
  increasing control deviation → rising manipulated variable
- Negative proportional gain
  increasing control deviation → falling manipulated variable

Applying proportional action in feedback path

If the proportional action is applied to the feedback path, only the process value has an effect on the proportional action. Therefore, the manipulated variable does not undergo a step change due to the proportional action during a changeover from manual to automatic (also in the case of a setpoint step-change); the changeover is bumpless.

See also

Control algorithm (Page 876)

Integral action (S7-300, S7-400)

Function of the integral action

The integral action ensures that the control deviation can go to 0 for any manipulated variables by resetting the operating point.

The integral action generates an output signal whose rate of change is proportional to the change in the absolute value of the input variable. The time response is determined by the integral action time TI.

The response characteristic in the time range is:

$$OUTV(t) = \frac{1}{TI} \int inv(t)\, dt$$

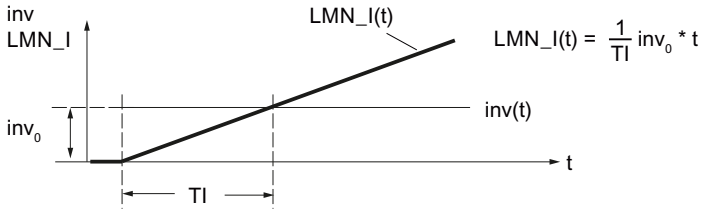

LMN_I(t)	The output variable of the integral action
inv_0	The step height at the input of the integral action
TI	Integral action time

The integral action can be monitored at the static tag LMN_I.

Parameter	Meaning	Permitted value range
TI	Integral action time	≥ 5 * CYCLE
I_ITLVAL	Initialization value for I action	-100.0 to +100.0 [%]

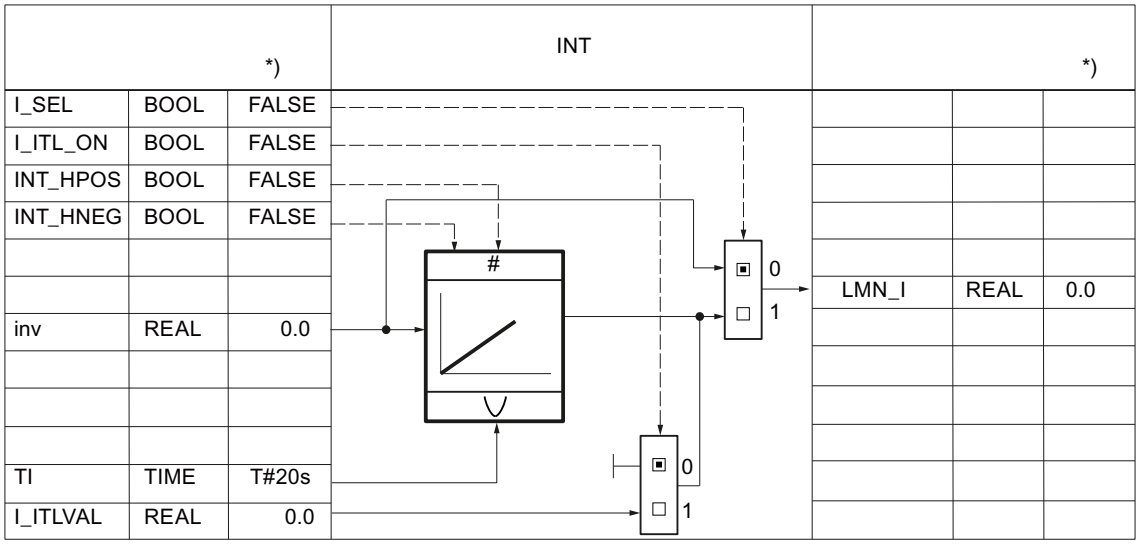

*) Default for creation of new technology object

In the deactivated state (I_SEL = FALSE), the integral action, i.e., the internal memory and the static tag LMN_I, is set to zero.

The output and the memory of the integral action are limited by the absolute limits of the manipulated variables LMN_HLM and LMN_LLM (Anti Reset Wind–up).

If the actuating signal is influenced in manual mode, i.e., if MAN_ON, LMNOP_ON or CAS_ON = TRUE, the internal memory value of the integral action is corrected to the value LMN.

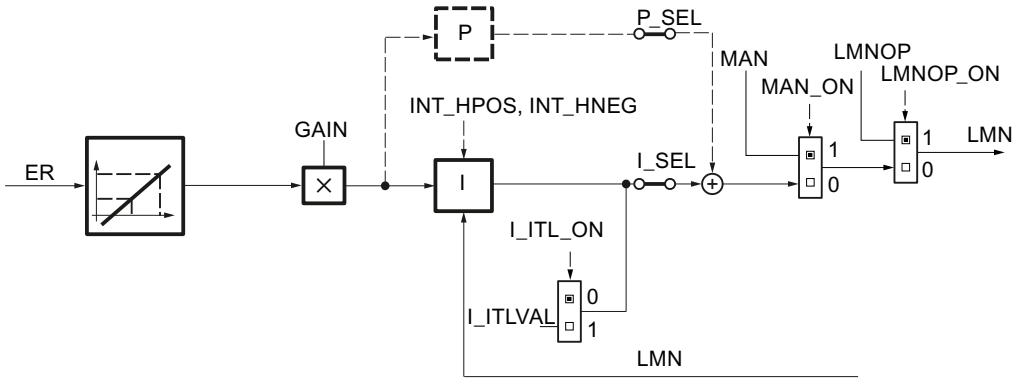

Changeover from manual to automatic mode

The integral action is set in manual mode in such a manner that the manual-to-automatic transition causes the manipulated variable to move from the current position. The height of the step change is determined by the proportional action. The step height of the proportional action corresponds to the manipulated variable change on a setpoint step change from the current process value to the current setpoint.

However, if the proportional action is applied to the feedback path, only the process value has an effect on the proportional action. Therefore, the manipulated variable does not undergo a step change due to the proportional action during a switchover from manual to automatic; the changeover is smooth.

Permissible ranges for the integral action time TI and CYCLE

The limited accuracy of the REAL numbers calculated in the CPU means that the following can occur during integration: If the selected sampling time CYCLE of the control is too short in comparison with the integral action time TI and the input value inv of the integral action is small compared to its output value OUTV, the integral action does not respond and remains at its current output value.

This effect can be avoided if the following sizing rule is observed during configuration:

$CYCLE > 10^{-4} \times TI$

As a result, the integral action still responds to changes in the input values in the range of millionths of one part per thousand of the current output variable:

$inv > 10^{-10} \times OUTV$

In order for the response characteristic of the integrator algorithm to conform to the analog response, the sampling time should be less than 20% of the specified integral action time or TI should be at least five times the value of the specified sampling time:

$CYCLE < 0.2 \times TI$

The algorithm permits sampling time values up to $CYCLE \leq 0.5 * TI$.

Initializing the integral action

If initialization of the integral action is enabled, the initialization value is switched through to the static tag LMN_I at a restart of the CPU or COM_RST = TRUE. On the transition to normal operation, the integral action starts the integration of its input variable from the initialization value due to the resetting of the initialization.

Hold integral action

The integral action can be blocked in positive or negative direction by the static tags INT_HPOS and INT_HNEG. This can be useful for controller cascades. If, for example, the manipulated variable of the follow-up controller approaches the high limit, a further increase in the manipulated variable of the master controller can be prevented by its integral action. You program this response using the following instructions, for example, in STL:

```
A    "Follow-up controller".QLMN_HLM
=    "Master controller".INT_HPOS
A    "Follow-up controller".QLMN_ LLM
=    "Master controller".INT_HNEG
```

See also

Control algorithm (Page 876)

Derivative component (S7-300, S7-400)

Function of the derivative action

The derivative action generates an output signal whose size changes proportionally to the rate of change of the input variable. The time response is determined by the derivative time TD and the time delay TM_LAG.

The step response to the input step inv_0 is:

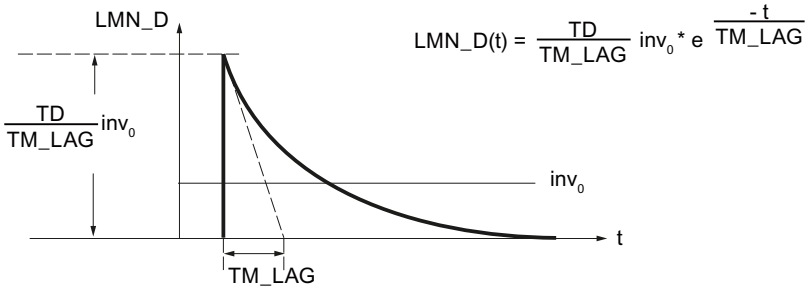

LMN_D(t)	The output variable of the derivative action
inv_0	The step height at the input of the derivative action
TD	Derivative-action time
TM_LAG	Time delay

The derivative action can be monitored at the static tag LMN_D.

In the disabled state (D_SEL = FALSE), the derivative action, i.e., the internal memory and the static tag LMN_D, is set to zero.

Apply derivative action in feedback path

If the derivative action is applied to the feedback path, only the process value has an effect on the derivative action. Consequently – as in a setpoint jump – the manipulated variable does not undergo a step change due to the derivative action during a changeover from manual to automatic; the switchover is smooth.

Permitted values for derivative time and time delay.

To calculate the derivative action correctly, the following conditions must be satisfied for the derivative time and time delay:

- TD ≥ CYCLE and
- TM_LAG ≥ 0.5 × CYCLE

If TD < CYCLE, the derivative action operates as though the TD has the value CYCLE.

If TM_LAG < 0.5 × CYCLE, the derivative action operates without delay. An input step is then multiplied with the factor TD/CYCLE and this value is applied as "needle pulse" at the output. In other words, LMN_D is reset to zero in the subsequent processing cycle.

Coefficient DT1

The coefficient DT1 is calculated as follows:

Coefficient DT1 = TD/TM_LAG

Time delay

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time constant: TM_LAG) is integrated into the algorithm for generating the derivative action.

A small value for the time delay is usually sufficient to achieve the desired result. If the assigned time delay is less than half the controller sampling time, the delay is deactivated.

Procedure

To configure the derivative action, proceed as follows:

1. To apply the derivative action to the feedback path, select the corresponding check box.
2. Enter the derivative time in seconds.
3. Enter the time delay in seconds.

See also

Control algorithm (Page 876)

Manipulated variable (S7-300, S7-400)

Step controller with position feedback (S7-300, S7-400)

Function chart of the step controller with position feedback in the control loop

The functional principles of the step controller with position feedback are based on the PID control algorithm and are supplemented by the function elements for generating the binary output signal from the analog actuating signal of the controller.

Depending on the sign, the three-step element hereby converts deviations between manipulated variable and position feedback into positive or negative pulses of the output signal in order, for example, to apply these to a motor-operated valve drive. Basically, this amounts to a cascade control with a lower-level position control loop.

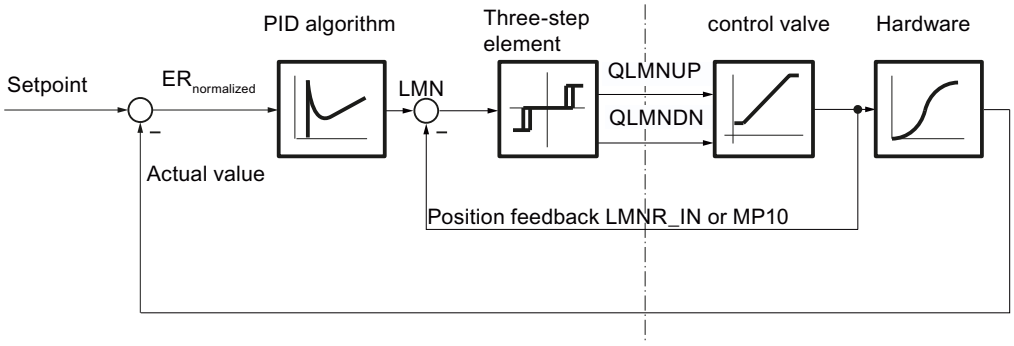

Selecting position feedback

For applying the position feedback to the comparator in the manipulated variable branch of the step controller, inputs with appropriate signal processing are available, depending on the format of the value to be processed. LMNR_PER can be used to apply signals in the SIMATIC I/O format, and LMNR_IN can be used to apply signals in floating-point format.

The associated internal value in the percent range is stored at measuring point MP10.

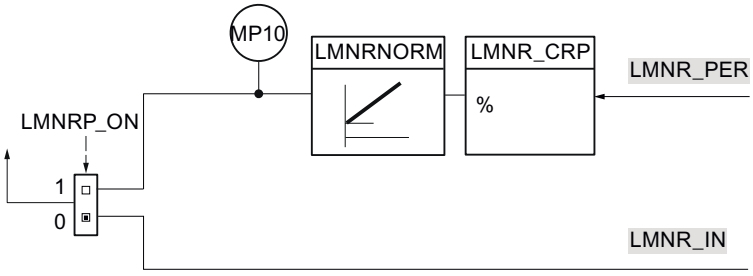

If the position feedback is provided by an analog input module, the numerical value of the position feedback at LMNR_PER must be converted to a floating-point value scaled to a percentage.

In this process, no check is carried out for positive/negative overflow, reaching of the overrange/underrange, or wire break.

The following table provides an overview of ranges and numerical values before and after processing.

I/O value LMNR_PER	Output value as a %
32767	118.515
27648	100.000
1	0.003617
0	0.000
- 1	- 0.003617
- 27648	- 100.000
- 32768	- 118.519

Procedure

Note

It is not permissible to enable or disable position feedback in online mode. In other words, LMNR_ON may not be changed online.

To activate the position feedback, proceed as follows:

1. Select the "Enable" check box in the "Select position feedback" area.
2. Choose the "Position feedback signal IO" option in order to apply signals in the SIMATIC I/O format.
   Choose the "Position feedback" option in order to apply signals in floating-point format.

Scaling position feedback

If the position feedback is available as a physical variable (e.g. 240 to 800 mm or in the angle from 0° to 60°), the position feedback in floating-point format (%) must be scaled to the physical variable required for the further processing of the required internal floating-point value.

The following parameters must be defined for the unique specification of the scaling levels:

- The factor (for the slope): **LMNR_FAC**
- The offset of the scaling level at the zero point: LMNR_OFF

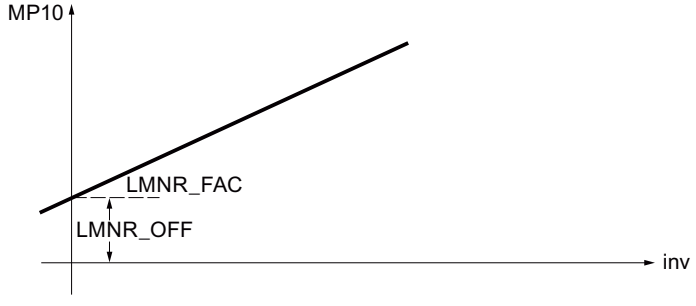

The scaled position feedback is calculated according to the following formula from inv (LMNR_PER) and output at MP10:

MP10 = inv * LMNR_FAC + LMNR_OFF

The scaling is effective if the I/O position feedback is enabled (LMNRP_ON = TRUE). No values are limited internally in the function. There is no parameter check.

Procedure

To scale the signal of the position feedback, proceed as follows:

1. Enter the slope of the scaling level in the "Factor" input field.
2. Enter the offset of the scaling level with respect to the zero point in the "Offset" input field.

Step controller without position feedback (S7-300, S7-400)

Structure and function of the step controller

The step controller without position feedback consists of two parts: The PD controller part, which operates with continuous signals, and a second part in which the binary actuating signals are generated from the difference between the PD action and the feedback.

The integral action of the step controller without position feedback is calculated in the feedback path. The feedback signal compared with the controller output of the PD controller is generated in this process by integrating the following two signal components:

- Signal component for simulated position feedback
  ± 100/MTR_TM
- Signal component for the integral action
  ER_{scaled}*GAIN/TI

The feedback signal is thus the difference between simulated position feedback and integral action.

Depending on the sign, the three-step element converts deviations between manipulated variable and feedback signal into positive or negative pulses of the output in order, for example, to apply these to a motor-operated valve drive.

In steady state, the output of the integral action and the PD action go to zero. Because the input of the three-step element also goes to zero, the binary actuating signals QLMNUP and QLMNDN remain at FALSE.

Because the integration take place in the feedback path, the integral action is not calculated in the PID algorithm. However, the integral action in the PID algorithm can be set to a fixed value with I_ITL_ON and I_ITLVAL.

With step controller without position feedback, the freezing of the integral action with INT_HPOS and INT_HNEG is not supported.

In this case, there is no manual mode option using the MAN parameter because of the lack of information regarding the actuator position.

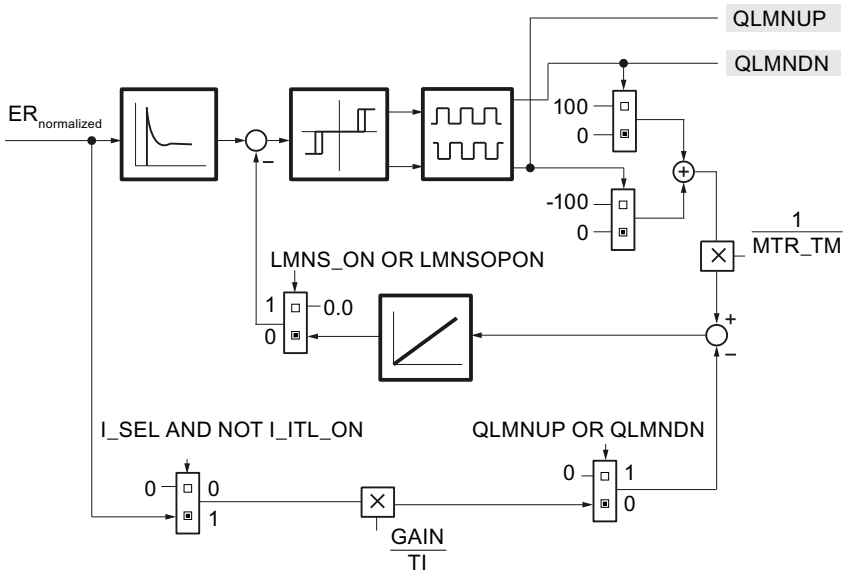

Addressing the endstop signals of the actuator (LMNR_HS/LMNR_LS) causes the manipulated variable LMNUP or LMNDN to be blocked.

If the actuator returns no endstop signals, the input parameters LMNR_HS and LMNR_LS = FALSE must be set.

Note

If end stop signals are not available, the controller cannot detect whether a mechanical end stop has been reached. For example, it is possible that the controller will output signals for opening the valve even though the valve is already at its upper end stop.

Manual mode

The step controller without position feedback has no manual mode via the parameter MAN or the manual output variable generator MAN_GEN because there is no information on the position of the actuator. Manual mode is only possibly by direct switching of the actuating signal (LMNS_ON = TRUE).

Simulation of position feedback

If a measurable value for the position feedback is not available, it can also be simulated (LMNRS_ON = TRUE).

The position feedback is simulated by an integrator with motor transition time MTR_TM as the integration time constant. When LMNRS_ON = FALSE, the start value of the LMNRSVAL parameter is output at the output of the LMNR_ SIM integrator. On switchover to LMNRS_ON = TRUE, the simulation starts up with this start value.

If LMNR_HS = TRUE, the integration is held up, and if LMNR_LS = TRUE, it is held down. The simulated position feedback is not adapted to the end positions.

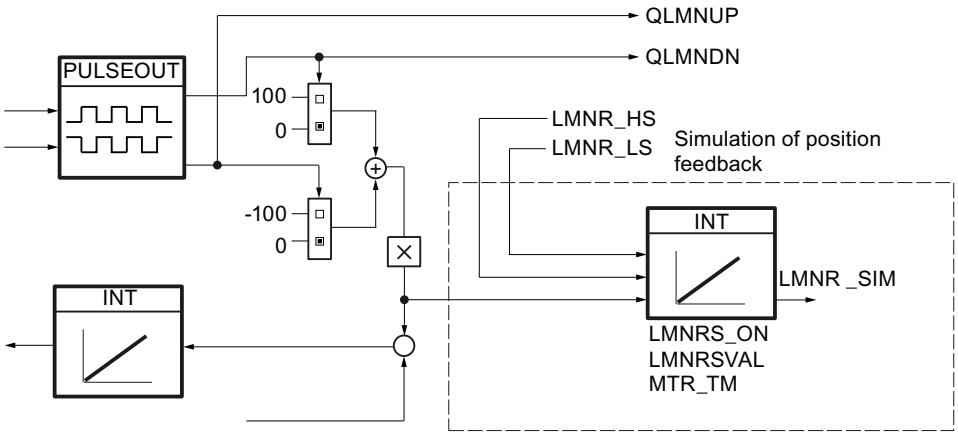

Note

The position feedback is merely simulated. It does not have to match the actual position of the actuator. If a real position feedback is available, it should always be used.

Manual mode (S7-300, S7-400)

Manual mode

The structure of the manual value generation and manual value selection is shown in the next figure.

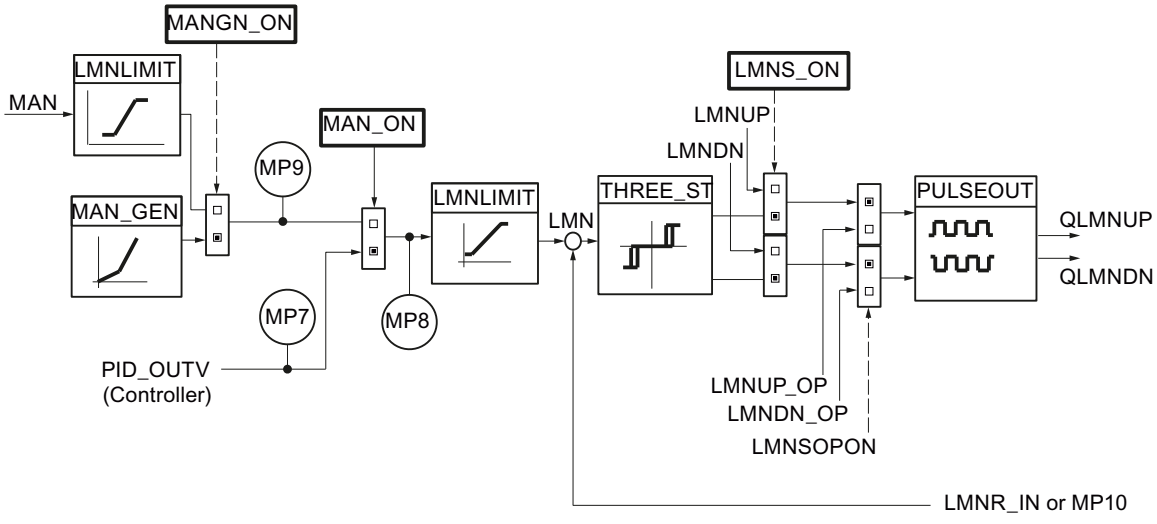

With step controllers, there are three operating modes in which the manipulated variable can be influenced:

- Manual value via the input parameter MAN
  The MAN parameter (-100.0% to 100.0%) allows you to activate influencing by an external manipulated variable in the form of an absolute value or to specify it in the user program. This is possible only when position feedback is enabled.
- Manual mode via more/less switch of the manual value generator (MANUP, MANDN).
  If MAN_GEN is activated from another operating mode, the manipulated variable MP9 is applied. The changeover to the manual value generator is thus always bumpless. The manual output variable can be increased or decreased within the LMN_HLM and LMN_LLM limits.
  This is possible only when position feedback is enabled.
- Manual mode by means of direct connection of the actuating signals
  Manual mode by direct connection of the actuating signals LMNUP and LMNDN always has priority.
  When LMNS_ON = FALSE is reset, the switch to any other activated operating mode will be "bumpless".

Automatic mode

If manual mode is disabled (MAN_ON = FALSE), the manipulated variable of the PID algorithm is connected to the three-step element. The changeover from manual mode to automatic mode causes a step change in the manipulated variable LMN. However, this is not a disadvantage because the integrating actuator means that the process is controlled according to a ramp function. The output of the PID algorithm is stored in measuring point MP7.

Manual value follow-up in automatic mode

If the I/O position feedback is enabled (LMNRP_ON = TRUE), the input parameter MAN of the position of the actuator (MP10) is corrected.

If position feedback is enabled (LMNRP_ON = FALSE), the input parameter MAN of the position of the actuator (LMNR_IN) is corrected.

Therefore, on switchover to manual mode, the manipulated variable remains at the value that corresponds to the actuator position. It can now be changed by operator input.

Activate manual mode with manual value

In this mode (MANGN_ON = FALSE and MAN_ON = TRUE), the manual output value is entered as an absolute value in the MAN input. The manual output value is displayed at measuring point MP8.

1. Select the "Activate manual mode" check box.
2. Choose the "Manual value" option.

Activate the manual value generator

The manual value MAN can be incremented or decremented in steps using the MANUP and MANDN switches. The modified value can be monitored at MP8. The configured manipulated variable limits are taken into account.

1. Select the "Activate manual mode" check box.
2. Select the "Manual value generator" option.

Effect of the switches MANUP and MANDN

For changes involving small steps, the controller should have a maximum sampling time of 100 ms.

The rate of change of the manipulated variable depends on how long the MANUP or MANDN switch is actuated and on the selected limits. The following relationship applies:

- During the first 3 s after setting MANUP or MANDN:
  doutv/dt = (LMN_HLM - LMN_LLM) / 100 s
- Afterwards:
  doutv/dt = (LMN_HLM - LMN_LLM) / 10 s

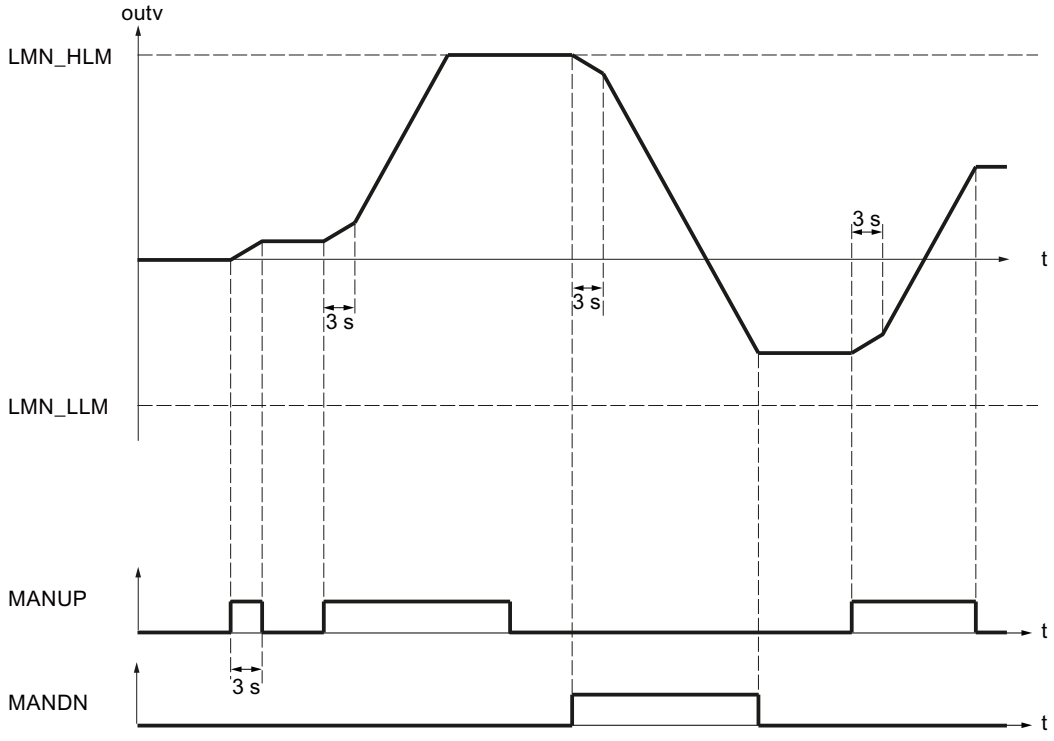

For a sampling time of 100 ms and a manipulated variable range of -100.0 to 100.0, the manipulated variable, for example, changes by 0.2 per cycle during the first 3 seconds. If MANUP is TRUE for more than 3 seconds, the rate of changes increases to ten times the value, in this case therefore to 2 per cycle.

The manual value generator is activated bumpless.

Switching the operating modes

The following table shows the possible operating modes of the step controller with the required values of the structure-defining switches.

Table 3-1 Operating modes of step controller

	MAN_ON	MNGN_ON	LMNS_ON
Automatic mode	FALSE	Any	FALSE
Manual mode without switch operation	TRUE	FALSE	FALSE
Manual mode with manual value generator	TRUE	TRUE	FALSE
Manual mode using actuating signals	Any	Any	TRUE

Limiting functions (S7-300, S7-400)

With step controllers with position feedback, you can limit the manipulated variable.

Limiting the absolute value of the manipulated variable

The operating range, i.e., the range in which the actuator can vary within the scope of permissible manipulated variables, is determined by the setting range of the manipulated variable. Since the limits for allowed manipulated variables usually do not coincide with the 0% or 100% limit of the correcting range, depending on the type of actuator, you can configure the high and low limit of the manipulated variable.

To avoid an impermissible status in the particular process, high and low limits for the setting range of the manipulated variable will be set in the manipulated variable branch of the closed-loop control.

The LMNLIMIT function limits manipulated variable LMN(t) to the low and high values LMN_HLM and LMN_LLM. These values can be specified. However, input variable inv must be outside these limits. Because the function cannot be deactivated, the specification of a low limit and high limit must always be taken into consideration in the configuration.

The numeric values of the limits (in percent) are set in the input parameters for the low and high limits. In the event of violations by input variable inv(t), the associated displays are output via the message outputs.

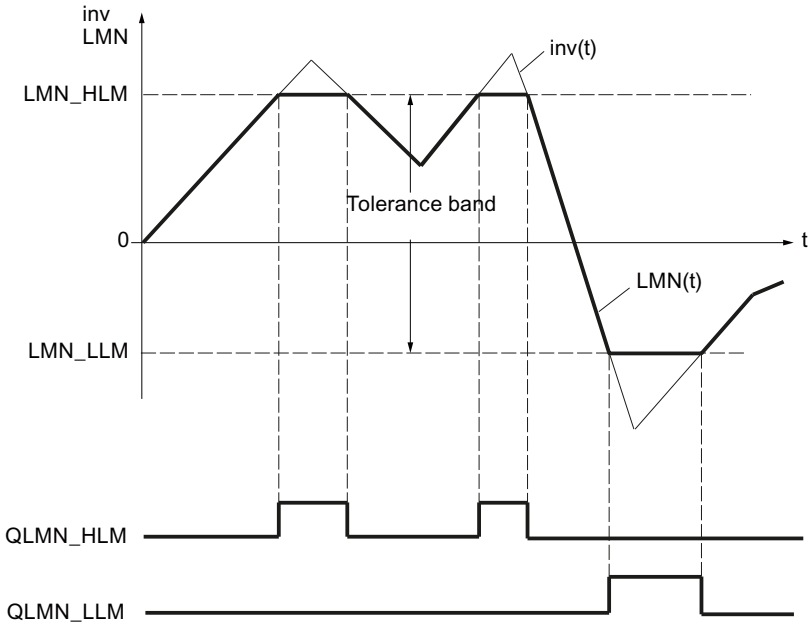

Startup and operating principles

- During a restart, all message outputs are set to zero.
- The limiting operates according to the following relationships:

LMN =	QLMN_HLM	QLMN_LLM	If:
LMN_HLM	TRUE	FALSE	inv ≥ LMN_HLM
LMN_LLM	FALSE	TRUE	inv ≤ LMN_LLM
inv	FALSE	FALSE	LMN_HLM < inv < LMN_LLM

The effective manipulated value of the controller is displayed at the output, i.e. at the parameter LMN or at the measuring point MP10.

Procedure

To limit the absolute value of the manipulated variable, proceed as follows:

1. In the "High limit" input field, enter the value for the maximum permissible value.
2. In the "Low limit" input field, enter the value for the minimum permissible value.

Pulse generator (S7-300, S7-400)

The pulse generator ensures that a minimum on and off time is adhered to in order to protect the actuators. Addressing the endstop signals of the actuator (LMNR_HS/LMNR_LS) causes the manipulated variable LMNUP or LMNDN to be blocked. This applies to automatic and manual mode.

If both signal switches are set for manipulated variable signal input (LMNUP = LMNDN = TRUE and LMNUP_OP = LMNDN_OP = TRUE), outputs QLMNUP and QLMNDN always return FALSE.

It is not possible to change directly from "Manipulated variable signal up" (QLMNUP = TRUE, QLMNDN = FALSE) to "Manipulated variable signal down" (QLMNUP = FALSE, QLMNDN = TRUE). The pulse generator inserts a cycle with QLMNUP = QLMNDN = FALSE.

Input signal INV with step controllers with position feedback

The difference between manipulated variable LMN and position feedback LMNR is switched-through to the three-step element with hysteresis THREE_ST.

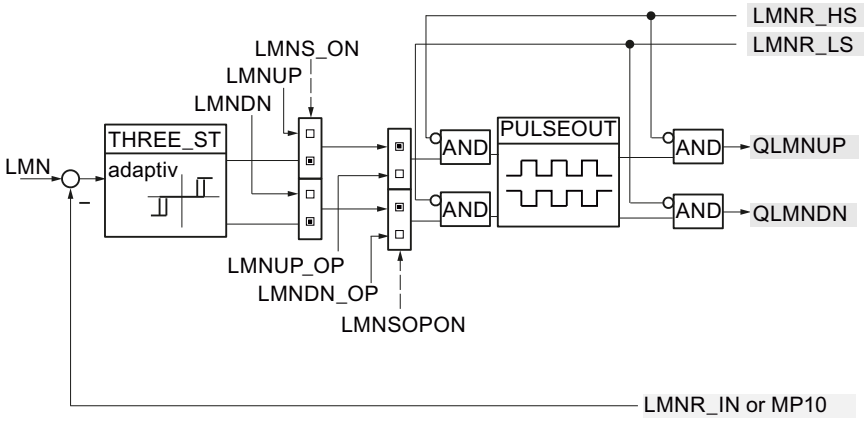

Input signal INV with step controllers without position feedback

The difference between the PD action of the controller and the feedback value (MP11) is switched to the three-step element with hysteresis THREE_ST.

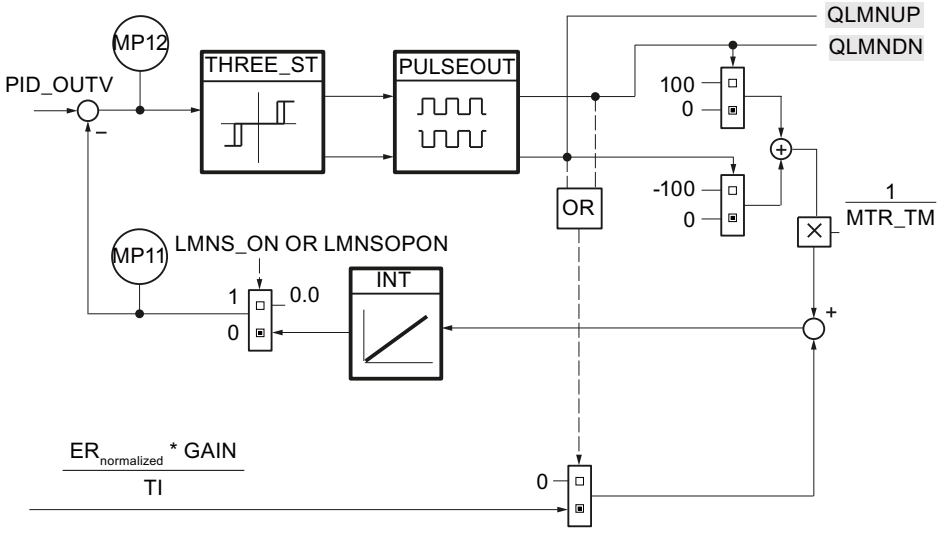

Influence of motor transition time on the three-step element

The three-step element generates two binary signals which are TRUE or FALSE depending on the value and sign of the differential value at the input.

The three-step element THREE_ST responds to the input signal INV according to the following relationships and hereby adopts in each case one of the three possible combinations of its output signals UP/DOWN:

UP	DOWN	Input combination
TRUE	FALSE	|INV| ≥ ThrOn or INV > ThrOff and UP_{old} = TRUE
FALSE	TRUE	|INV| ≤ -ThrOn or INV < -ThrOff and $DOWN_{old}$ = TRUE
FALSE	FALSE	|INV| ≤ ThrOff

ThrOn = Switch-on threshold, ThrOff = Switch-off threshold

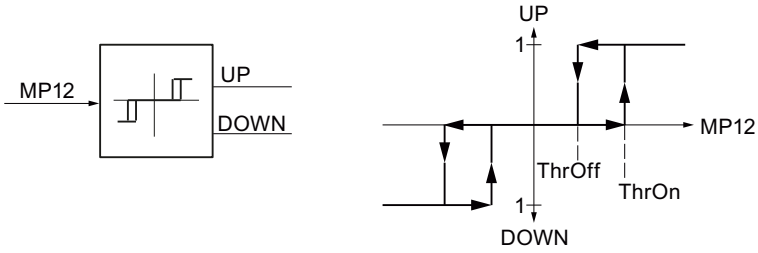

The switch-off threshold ThrOff must be greater than the change in the position feedback or the position change after one pulse duration. This value is dependent on the transition time of the motor MTR_TM and is calculated as follows:

$$\text{ThrOff} = 0.5 * \frac{110}{\text{MTR_TM}} * \text{CYCLE}$$

MTR_TM must be an integer multiple of CYCLE.

Note

An excessively long motor transition time setting (10% over the actual transition time) will cause the manipulated variable signals QLMNUP and QLMNDN to constantly switch on and off.

Adaptation of the switch-on threshold ThrOn

To reduce the switching frequency when larger control deviations are corrected, the switch-on threshold ThrOn is adapted automatically during operation, while ThrOff remains constant. ThrOn is limited to:

$$\text{Min ThrOn} = \frac{100}{\text{MTR_TM}} * \text{MAX (PULSE_TM, CYCLE)}$$

$$\text{Max ThrOn} = 10$$

The adaptation of the switch-on threshold is disabled in pure P, D, and PD controllers. For example:

ThrOn = Min ThrOn

Minimum ON time and minimum OFF time.

The pulse generator ensures that a minimum on and off time is adhered to when the output pulses are switched on and off.

Parameters for a minimum ON time PULSE_TM and a minimum OFF time BREAK_TM can therefore be assigned to protect the actuator. The duration of the output pulse QLMNUP or QLMNDN is always greater than the minimum ON time and the break between two pulses always greater than the minimum OFF time. Using the UP signal as an example, the following figure shows how this works.

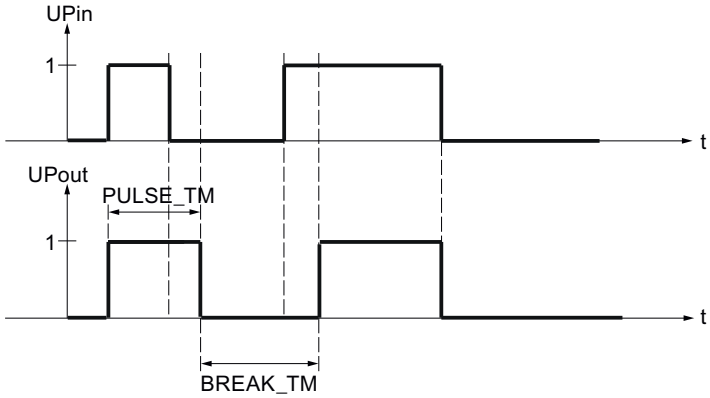

Procedure

To configure the pulse generator, proceed as follows:

1. Enter the transition time of the motor in the "Motor transition time" input field.
2. Enter the "Minimum ON time".
3. Enter the "Minimum OFF time".

Commissioning PID_ES (S7-300, S7-400)

Performing tuning manually (S7-300, S7-400)

Requirements

- The instruction and the technology object have been loaded to the CPU.
- MAN_ON = FALSE

Procedure

In order to manually determine the optimal PID parameters, proceed as follows:

1. Click the "Measurement on" icon.
   An online connection will be established, if one is not already present. The current values for the setpoint, process value and manipulated variable are recorded.
2. Enter new PID parameters in the "P", "I", "D" and "Time delay" fields.
3. Click on the icon "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
   If the setpoint's rate of change is limited, the changeover to the new setpoint will be bumpless
6. Clear the "Manual mode" check box, if necessary.
   The controller works with the new PID parameters and controls the new setpoint.
7. Check the quality of the PID parameter based on the curve points.
8. Repeat steps 3 to 8 until you are satisfied with the controller results.

Commissioning with the PID Self Tuner (S7-300, S7-400)

You can commission the PID_ES instruction and determine the optimum PID parameters with the PID Self Tuner.

Procedure

To commission a step controller, proceed as follows:

1. Call the instruction PID_ES in a cyclic interrupt OB and interconnect the input and output parameters.
2. Call the instruction TUN_ES in the same cyclic interrupt OB.

3. Interconnect the instruction TUN_ES as follows with PID_ES:
   - TUN_ES.PV with PID_ES.PV
   - If you are using a position feedback, TUN_ES.LMNR with PID_ES.LMNR_IN
   - TUN_ES.C_LMNUP and TUN_ES.C_LMNDN with PID_ES.QLMN_UP and PID_ES.QLMNDN
   - TUN_ES.LMNR_HS with PID_ES.LMNR_HS
   - TUN_ES.LMNR_ON with PID_ES.LMNR_ON
   - TUN_ES.LMNS_ON with PID_ES.LMNS_ON
   - TUN_ES.SP_OUT with PID_ES.SP_IN or PID_ES.SP_EXT
   - TUN_ES.GAIN with PID_ES.GAIN
   - TUN_ES.TI with PID_ES.TI
   - TUN_ES.TD with PID_ES.TD
   - TUN_ES.TM_LAG with PID_ES.TM_LAG
   - TUN_ES.MTR_TM with PID_ES.MTR_TM
   - If you are using a deadband, TUN_ES.DEADB_W with PID_ES.DEADB_W
   - TUN_ES.QLMNUP and TUN_ES.QLMNDN with PID_ES.LMNUP and PID_ES.LMNDN
4. Click the "" icon on the instruction TUN_ES.
   The commissioning editor for TUN_ES opens. The operation of the commissioning editor is described with the PID Self Tuner.

3.5.2 Using modular PID control (S7-300, S7-400)

3.5.2.1 Scope of Modular PID Control functions (S7-300, S7-400)

In many control tasks, the classic PID controller which controls the processes is not the only key element. Complex signal processing is also required.

Modular PID Control includes instructions with functions for processing the process value and setpoint and calculating and tuning the manipulated variable.

The following controllers are possible:

- Continual controller
- Step controller for actuators with integrating behavior
- Pulse controller

You can implement controllers for different applications by connecting individual instructions.

- Fixed value controller
- Cascade controller
- Ratio controller
- Mixing controller

- Split range controller
- Override controller
- Controller with disturbance variable processing
- Multi-variable controller

Both slow (temperature, filling levels etc.) and fast controlled systems (flow, pressure, etc.) can be controlled.

You can call multiple control loops equidistantly but with different sampling intervals using the instruction LP_SCHED_M. You alter the sampling time of the control loop in accordance with the slowness of the controlled system.

This ensures the even distribution of the CPU load.

Use the PID Self-Tuner to commission and fine-tune the PID parameters.

3.5.2.2 Basic steps for setting up a controller FB (S7-300, S7-400)

Basic procedure for programming a controller

You program a controller as the function block (controller FB), for example with STL. Your controller FB must have at least the following parameters:

- Input parameter COM_RST
- Input parameter CYCLE
- Input parameter for setpoint
- Input parameter for process value
- Output parameter for the manipulated variable
- In/out parameter for the manual output value (continual controller)

Depending on controller requirements, you can also set the following parameters

- Input parameter for position feedback (step controller)
- Input parameter for disturbance variable
- Output parameter for the processed process value
- Output parameter for the processed setpoint
- Output parameter for limit violation

The details of how to set the inputs and outputs depend on your specific situation, for example on whether the process value is in I/O or floating-point format, on the type of position feedback, etc.

Call the Modular PID Control instructions in the controller FB and connect the input and output parameters. Call the controller FB you have created, for example in OB 100 or in a cyclic interrupt OB.

The following steps are needed to program a controller with Modular PID Control.

Step	Description
1.	Processing setpoints (Page 901)
2.	Processing the process value (Page 903)
3.	Generating and processing the control deviation (Page 904)
4.	Applying the disturbance variable (Page 906) (optional)
5.	Calculating the manipulated variable • Creating a continual controller (Page 908) • Creating a pulse controller (Page 910) • Creating a step controller (Page 913)
6.	Processing position feedback (Page 914)

See also

Processing setpoints (Page 901)
Processing the process value (Page 903)
Generating and processing the control deviation (Page 904)
Applying the disturbance variable (Page 906)
Creating a continual controller (Page 908)
Creating a pulse controller (Page 910)
Creating a step controller (Page 913)
Processing position feedback (Page 914)

3.5.2.3 Programming the controller FB (S7-300, S7-400)

Processing setpoints (S7-300, S7-400)

Instructions for processing the setpoint

You can use the following instructions:

- SP_GEN: Enter setpoint manually
- ROC_LIM: Limit rate of change
- LAG1ST: Smooth setpoint
- NONLIN: Linearize setpoint
- LIMITER: Limit setpoint
- RMP_SOAK: Specify setpoint according to ramp/soak
- SCALE_M: Scale setpoint
- NORM: Normalize setpoint

Example of setpoint processing

It should be possible to increase or decrease a specified setpoint manually. The rate of change is limited to prevent jumps in the setpoint.

Proceed as follows to process the setpoint as described in the example:

1. At the controller FB, in this case PIDCTR, declare an input parameter SP_IN of the data type "Real".
2. Connect SP_IN to a variable containing the required setpoint.
3. Call the instruction SP_GEN as a multi-instance in the controller FB and connect SP_IN to SP_GEN.DF_OUTV.
4. Assign parameters for the high and low setpoint limit in the SP_GEN_Instance structure of the multi-instance.
5. Call the instruction ROC_LIM as a multi-instance in the controller FB and connect SP_GEN.OUTV to ROC_LIM.INV via a temporary variable.
6. Assign parameters for the maximum rates of change in the ROC_LIM_Instance structure of the multi-instance.
7. To generate the control deviation, subtract the processed process value from ROC_LIM.OUTV or connect ROC_LIM.OUTV to ERR_MON.SP.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR.

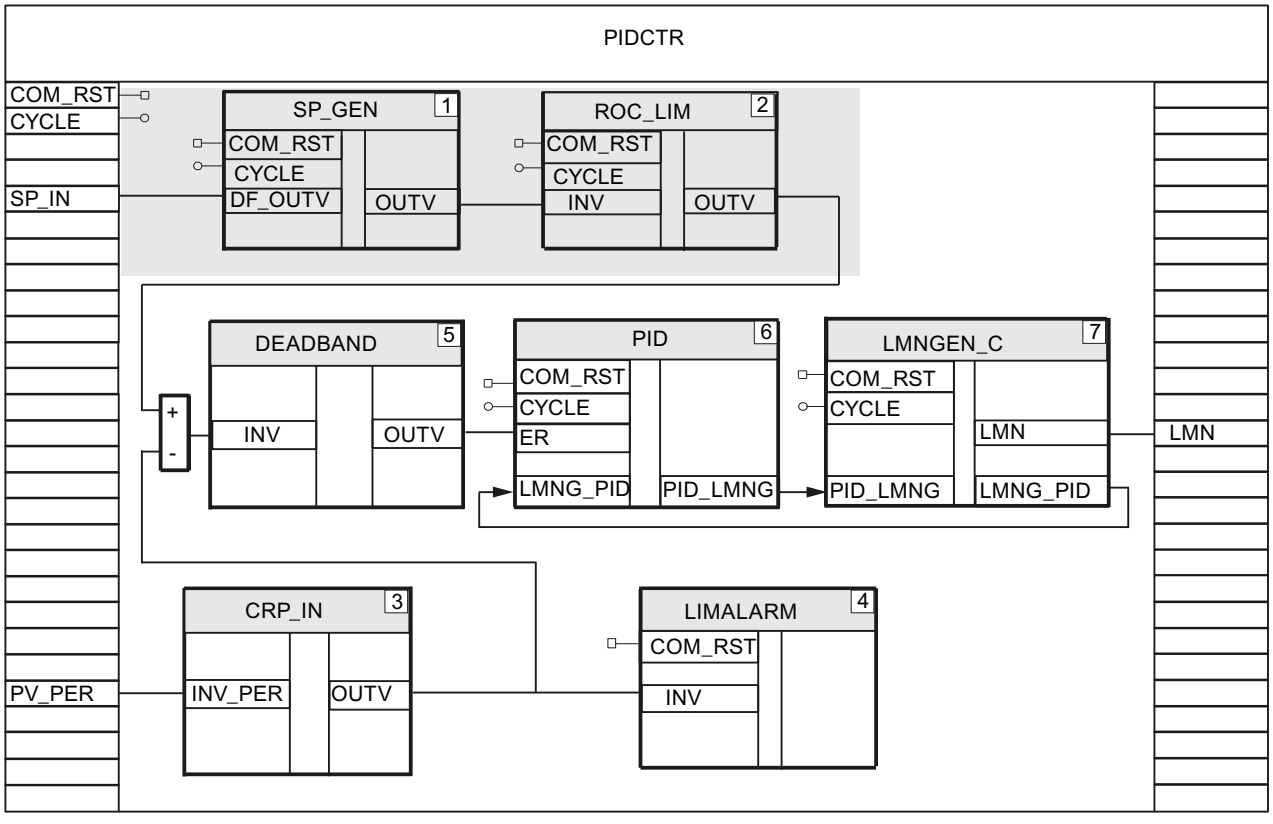

See also

Processing the process value (Page 903)

Processing the process value (S7-300, S7-400)

Instructions for processing the process value

You can use the following instructions:

- CRP_IN: Scale I/O value
- LAG1ST: Smooth process value
- LIMITER: Limit process value
- LIMALARM: Monitor process value
- NONLIN: Linearize process value
- SCALE_M: Scale process value
- NORM: Normalize process value

Example of process value processing

The process value is in I/O format and is to be monitored to ensure compliance with the process value limits.

Proceed as follows to process the process value as described in the example:

1. At the controller FB, in this case PIDCTR, declare an input parameter PV_PER of the data type "Word".
2. Connect PV_PER to the I/O input at which the process value is measured.
3. Call the instruction CRP_IN as a multi-instance in the controller FB, for example with the name CRP_IN_Instance_PV.
4. Connect PV_PER to CRP_IN.INV_PER.
5. Assign parameters for factor and offset for scaling in the CRP_IN_Instance_PV structure of the multi-instance.
6. Call the instruction LIMALARM as a multi-instance in the controller FB and connect CRP_IN.OUTV to LIMALARM.INV via a temporary variable.
7. Assign parameters for the process value limits in the LIMALARM_Instance structure of the multi-instance.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR.

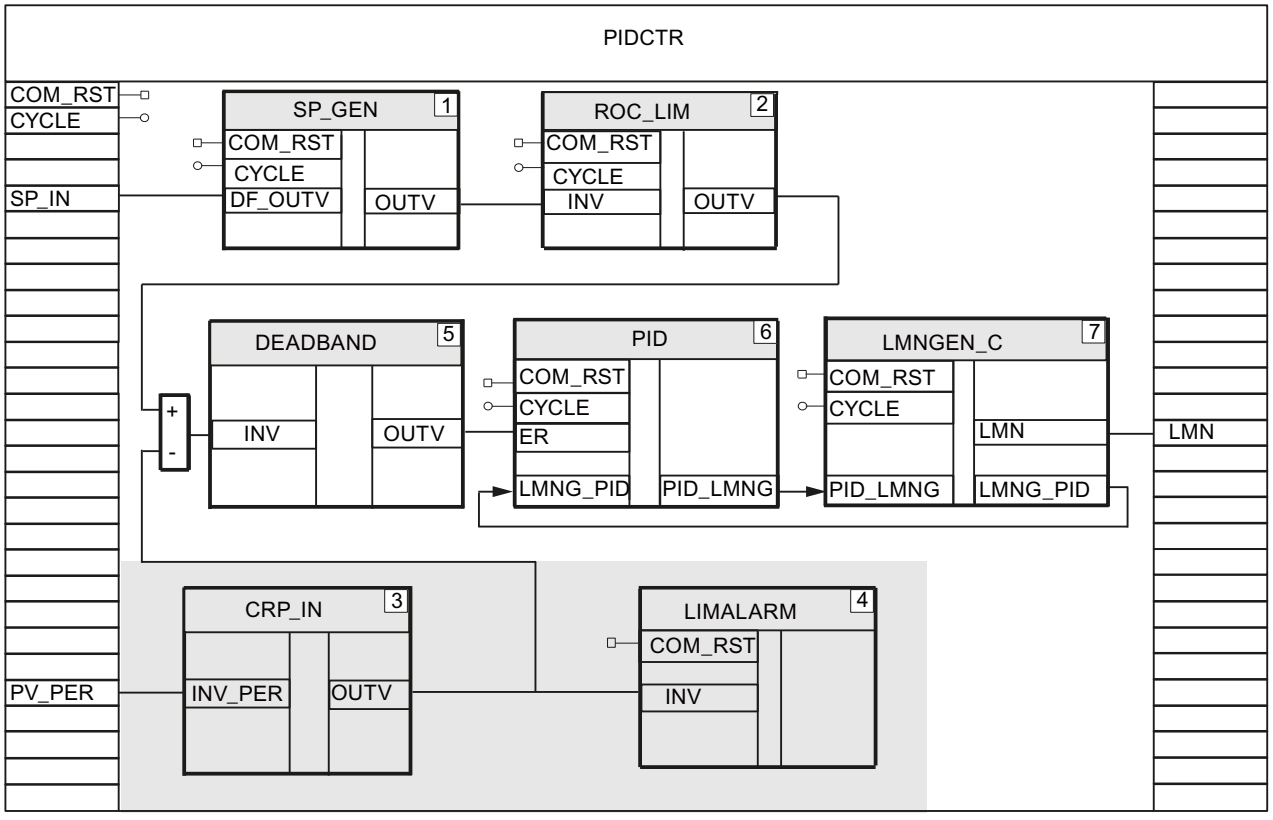

See also

Processing setpoints (Page 901)

Generating and processing the control deviation (Page 904)

Generating and processing the control deviation (S7-300, S7-400)

Establishing and limiting the control deviation

There are two ways to establish the control deviation. Either program the subtraction of the process value from the setpoint, or use the instruction ERR_MON. The instruction ERR_MON already includes control deviation monitoring.

Instructions for processing the control deviation

You can use the following instructions:

- A_DEAD_B: Route control deviation through adaptive deadband
- DEADBAND: Route control deviation through deadband

Example of subtraction

The control deviation is produced by subtraction and routed through a deadband.

Proceed as follows to determine the control deviation as described in the example:

1. Call the instruction DEADBAND as multi-instance and connect DEADBAND.INV to the difference between setpoint and process value, e.g. ROC_LIM.OUTV - CRP_IN.OUTV.
2. Assign parameters for the deadband width and a deadband displacement in the DEADBAND_Instance structure of the multi-instance.
   The processed control deviation is output at DEADBAND.OUTV.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR.

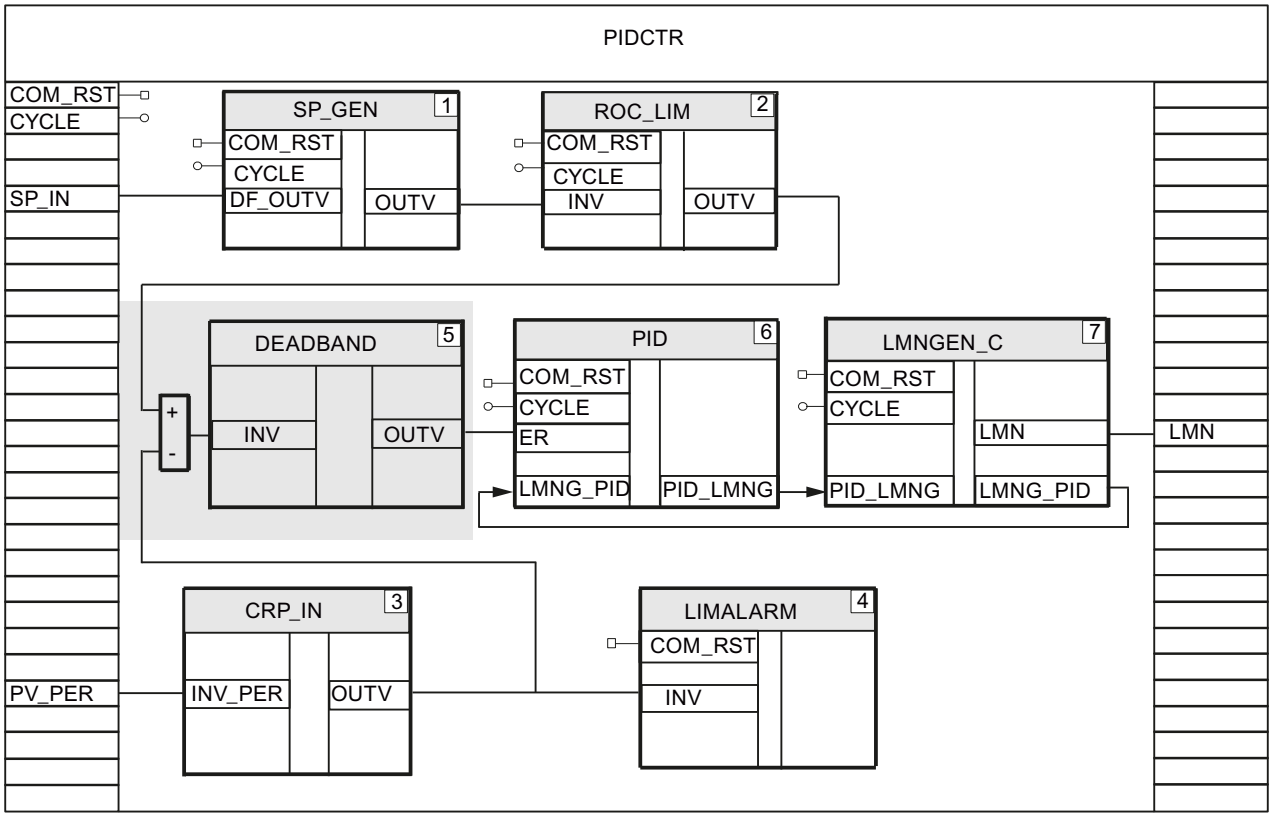

Example of the use of ERR_MON

The control deviation is calculated and monitored with the instruction ERR_MON.

Proceed as follows to determine the control deviation as described in the example:

1. Call the ERR_MON instruction as a multi-instance.
2. Using temporary variables, connect the processed setpoint to ERR_MON.SP, e.g. ROC_LIM.OUTV, and the processed process value to ERR_MON.PV, e.g. CRP_IN.OUTV. The control deviation is output at ERR_MON.ER.
3. Assign parameters for control deviation monitoring in the ERR_MON_Instance structure of the multi-instance.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR_ER.

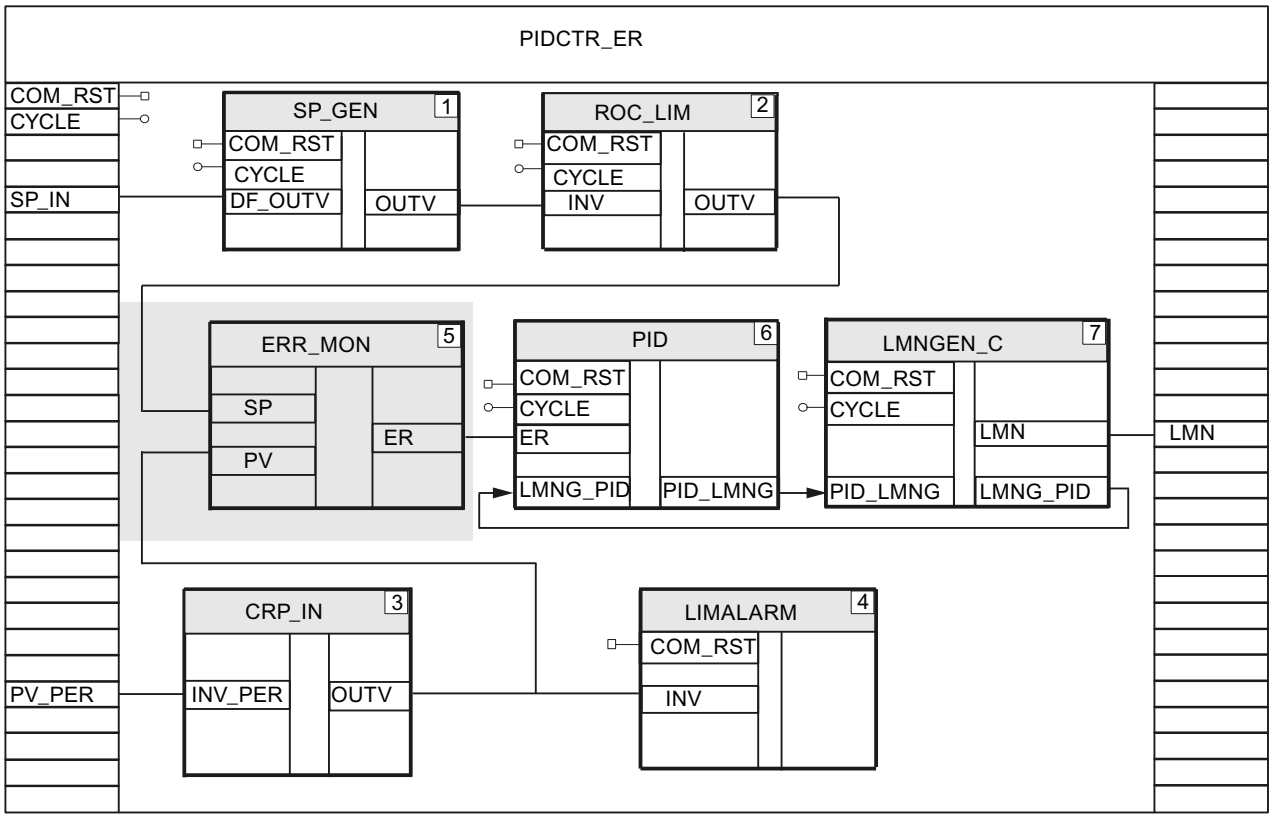

See also

Processing the process value (Page 903)

Applying the disturbance variable (Page 906)

Applying the disturbance variable (S7-300, S7-400)

Instructions for processing the disturbance variable

You can use the following instructions:

- CRP_IN: Scale I/O value
- DIF: Respond to fast changes
- LAG1ST: Filter disturbances
- NON_LIN: Linearize disturbance variable
- SCALE_M: Scale disturbance variable
- NORM: Normalize disturbance variable

Example of disturbance variable processing

The disturbance variable is in I/O format. The signal is to be smoothed and linearized with a characteristic curve.

Proceed as follows to process the disturbance variable as described in the example:

1. At the controller FB, declare an input parameter DISV_PER of the data type "Word".
2. Connect DISV_PER to the I/O input at which the disturbance variable is measured.
3. Call the instruction CRP_IN as a multi-instance in the controller FB, for example with the name CRP_IN_Instance_DISV.
4. Connect DISV_PER to CRP_IN.INV_PER.
5. Assign parameters for factor and offset for scaling in the CRP_IN_Instance_DISV structure of the multi-instance.
6. Call the instruction LAG1ST as a multi-instance in the controller FB and connect CRP_IN.OUTV to LAG1ST.INV via a temporary tag.
7. Assign parameters for the time delay in the LAG1ST_Instance structure of the multi-instance.
8. Call the instruction NONLIN as a multi-instance in the controller FB and connect LAG1ST.OUTV to NONLIN.INV via a temporary tag.
9. Create a AUTOHOTSPOT in which you save the time slices of the characteristic curve.
10. Assign parameters for the number of the global data block with the characteristic curve in the NONLIN_Instance structure of the multi-instance.
11. The processed disturbance variable is output at NONLIN.OUTV.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB CTR_C_FF.

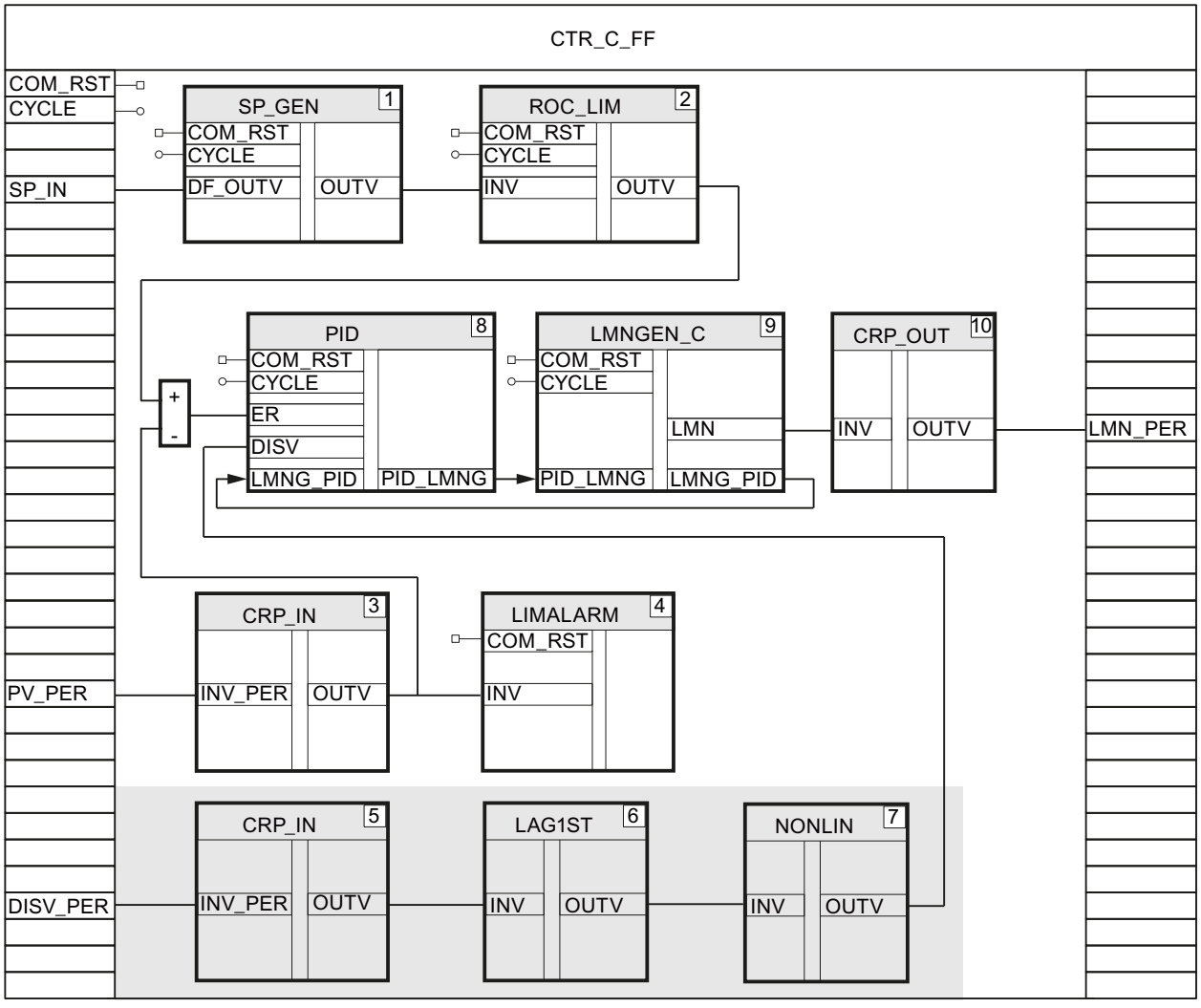

See also

Generating and processing the control deviation (Page 904)

Creating a continual controller (Page 908)

Creating a pulse controller (Page 910)

Creating a step controller (Page 913)

Creating a continual controller (S7-300, S7-400)

You will need the instructions PID, LMNGEN_C and CRP_OUT for a continual controller with an analog manipulated variable.

Procedure

Follow these steps to program a continual controller with an analog manipulated variable:

1. At the controller FB, declare an in/out parameter MAN of the data type "Real".
   You will require MAN to commission the controller with the PID Self-Tuner.
2. Call the instructions PID, LMNGEN_C and CRP_OUT as multi-instances in the controller FB.
3. At the controller FB, declare an in/out parameter MAN of the data type "Real" and connect MAN to LMNGEN_C.MAN.
4. Assign the parameters for the response of the control algorithm, e.g. whether the derivative action is used, in the PID_Instance structure of the controller DB.
5. Assign parameters for the manipulated value and rate of change limits, for example, in the LMNGEN_C_Instance structure of the multi-instance.
6. Assign parameters for the factor and offset for converting the floating-point value to I/O format in the CRP_OUT_Instance structure of the multi-instance.
7. Connect PID.ER to the control deviation via a temporary variable, e.g. ROC_LIM.OUTV - CRP_IN.OUTV or ERR_MON.ER.
8. If you are connecting a disturbance variable, connect PID.DISV to the processed signal of the disturbance variable, e.g. NONLIN.OUTV, using a temporary variable.
9. Connect the output parameter PID.PID_LMNG to the input parameter LMNGEN_C.PID_LMNG and the output parameter LMNGEN_C.LMNG_PID to the input parameter PID.LMNG_PID with the instruction BLKMOV.
10. Connect LMNGEN_C.LMN to CRP_OUT.INV using a temporary variable.
11. At the controller FB, declare an output parameter LMN_PER of the data type "Word" and connect CRP_OUT.OUTV to LMN_PER.
12. Connect LMN_PER to the analog output of the CPU which controls the actuator.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB CTR_C_FF .

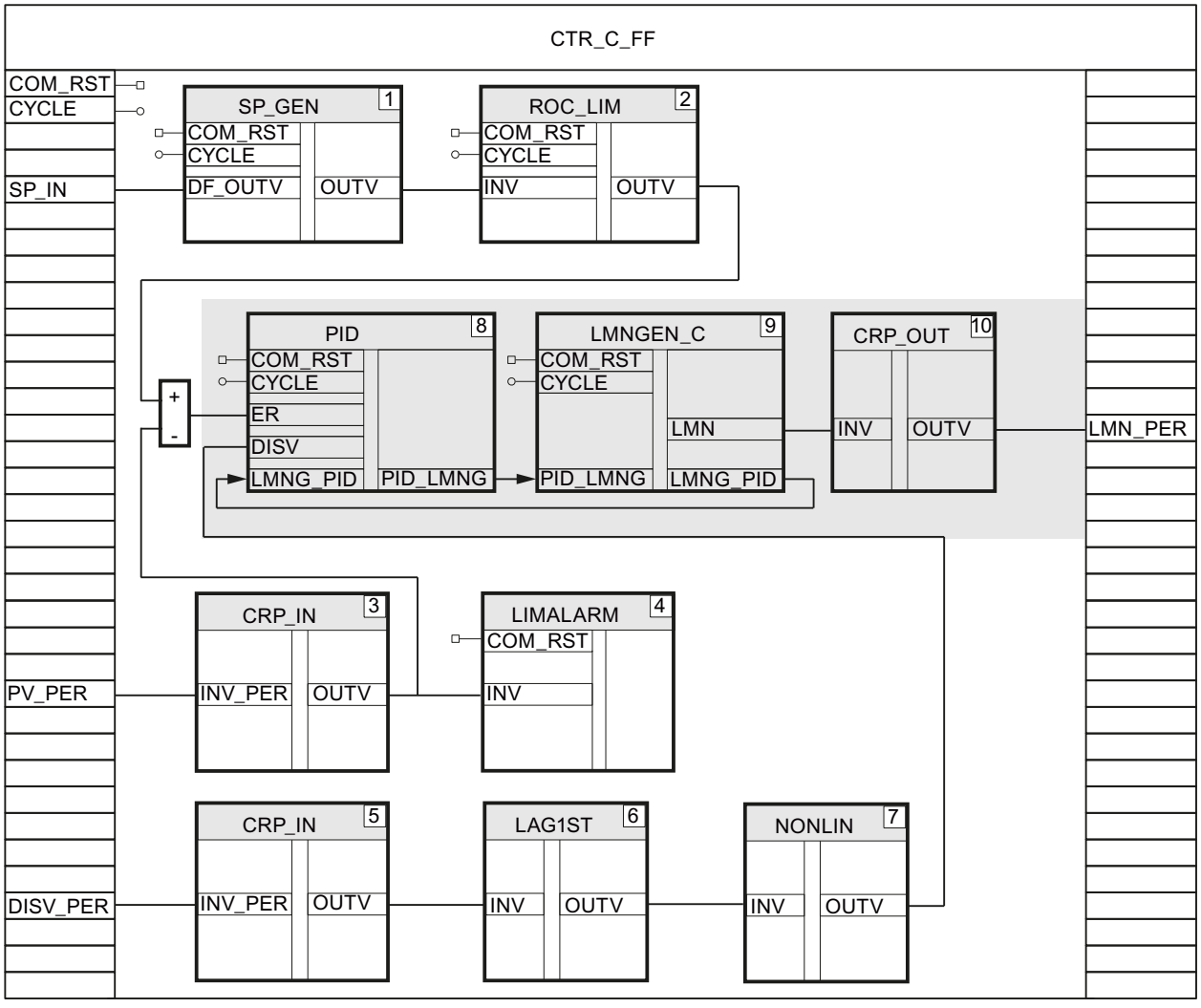

See also

Applying the disturbance variable (Page 906)

Creating a pulse controller (S7-300, S7-400)

For a pulse controller, you generate a continuous manipulated variable in floating-point format with the instructions PID and LMNGEN_C. On the basis of this continuous manipulated variable, you then generate a pulse train with pulse width modulation using the instruction PULSEGEN_M.

Generating a continuous manipulated variable in floating-point format

1. At the controller FB, in this case PIDCTR, declare an output parameter LMN of the data type "Real".
2. Call the instructions PID and LMNGEN_C as multi-instances in the controller FB.

3. Connect and assign parameters for PID and LMNGEN_C as described for the continual controller.
4. Connect LMNGEN_C.LMN to the output parameter LMN of the controller FB.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR.

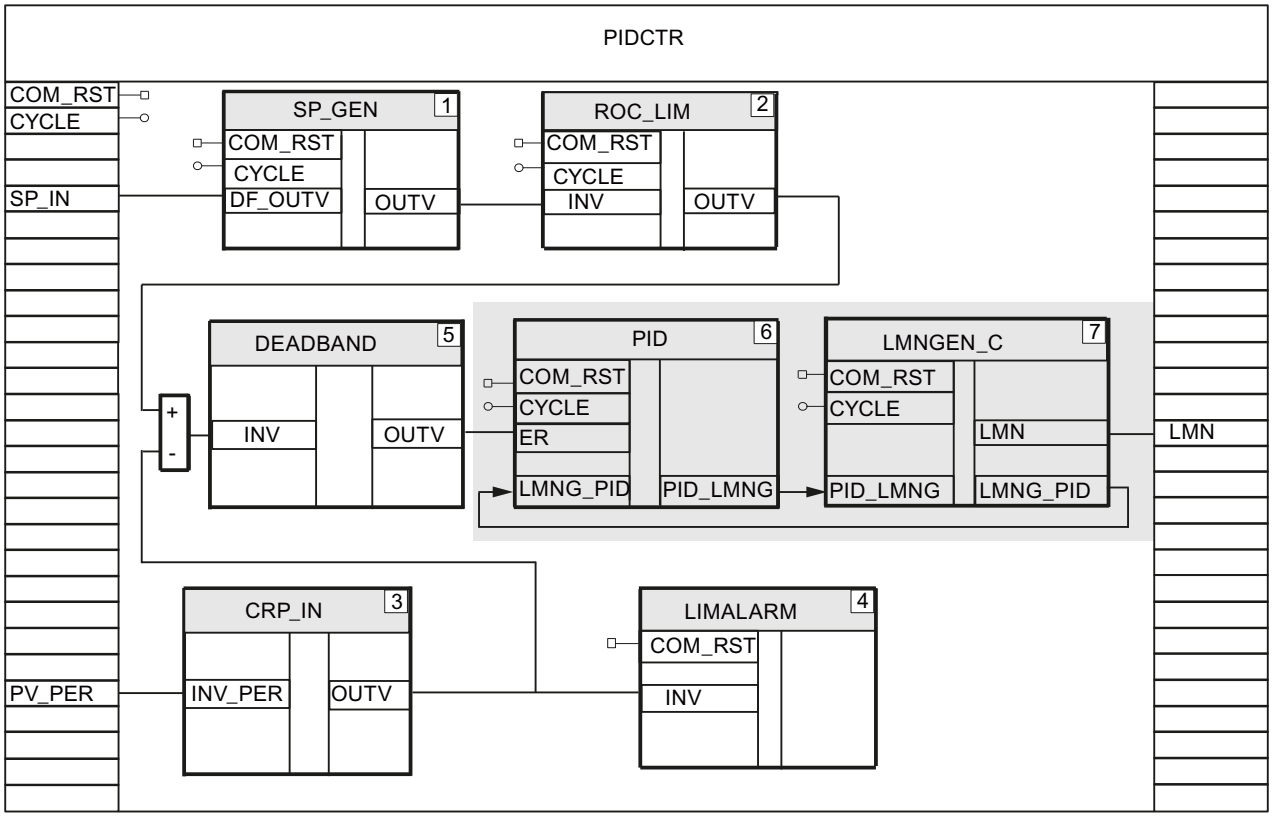

Generating a pulse train

Follow these steps to generate a pulse train from the continuous manipulated variable:

1. Create a function, e.g. FC50, with the input parameters COM_RST and CYCLE.
2. Create a AUTOHOTSPOT in which to assign parameters for the cycle times for calling the controller FB and the pulse generator PULSEGEN_M.
   The ratio of cycle time of the controller FB call to PULSEGEN_M should be ≥ 20.
3. Call the LP_SCHED_M instruction in the function.
4. Connect LP_SCHED_M.DB_NBR to the global data block .
5. Call your controller FB, e.g. PIDCTR, if DB_LOOP.LOOP_DAT[1].ENABLE = TRUE.
6. Connect DB_LOOP.LOOP_DAT[1].COM_RST and DB_LOOP.LOOP_DAT[1].CYCLE to input parameters COM_RST and CYCLE of the controller FB.
7. Set DB_LOOP.LOOP_DAT[1].ENABLE = FALSE.
8. Call the instruction PULSEGEN_M if DB_LOOP.LOOP_DAT[2].ENABLE = TRUE.

9. Connect DB_LOOP.LOOP_DAT[2].COM_RST and DB_LOOP.LOOP_DAT[2].CYCLE to PULSEGEN_M.COM_RST and PULSEGEN_M.CYCLE.
10. Connect PULSEGEN_M.INV to the output parameter LMN of your controller FB.
11. Interconnect PULSEGEN_M.PER_TM with the input parameter CYCLE of your controller FB.
12. Set DB_LOOP.LOOP_DAT[2].ENABLE = FALSE.
13. Connect PULSEGEN_M.QPOS_P and, if necessary, PULSEGEN_M.QNEG_P to the digital outputs of the CPU which control the actuator.
14. Call the function in OB 100 and in the cyclic interrupt OB.

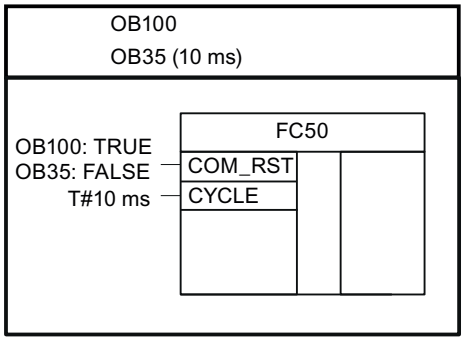

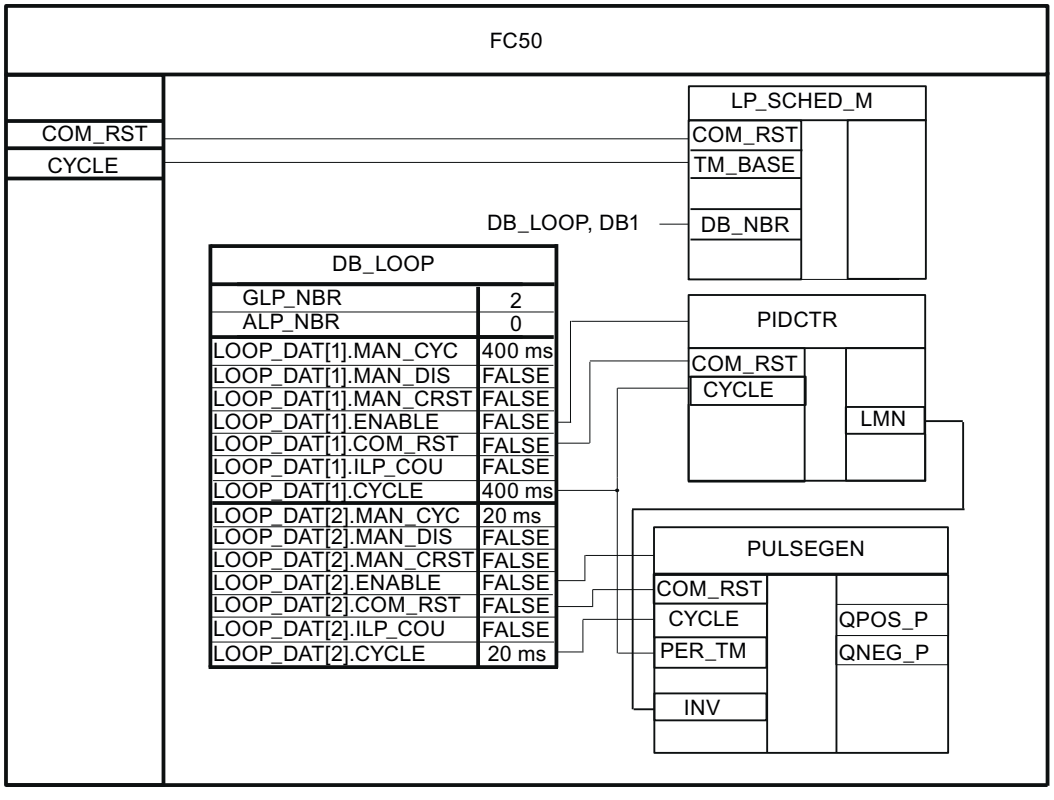

See also

Applying the disturbance variable (Page 906)

Creating a step controller (S7-300, S7-400)

You will require the instructions PID and LMNGEN_S for a step controller.

Procedure

Proceed as follows to program a step controller:

1. Call the instructions PID and LMNGEN_S as multi-instances in the controller FB.
2. Assign the parameters for the response of the control algorithm, e.g. whether the derivative action is used, in the PID_Instance structure of the multi-instance.
3. Assign parameters in the LMNGEN_S_Instance structure of the multi-instance, for example specify that no position feedback is to be used, to LMNR_ON = FALSE.
4. Connect PID.ER to the control deviation via a temporary variable, e.g. DEADBAND.OUTV or ERR_MON.ER.
5. If you are connecting a disturbance variable, connect PID.DISV to the processed signal of the disturbance variable, e.g. CRP_IN.OUTV, using a temporary variable.
6. Connect the output parameter PID.PID_LMNG to the input parameter LMNGEN_S.PID_LMNG and the output parameter LMNGEN_S.LMNG_PID to the input parameter PID.LMNG_PID with the instruction BLKMOV.
7. At the controller FB, declare the output parameters QLMNUP and QLMNDN of the data type "Bool".
8. Connect LNMGEN_S.QLMNUP and LMNGEN_S. QLMNDN to these output parameters.
9. Connect the output parameters of the controller FB to the digital outputs of the CPU which control the actuator.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR_S.

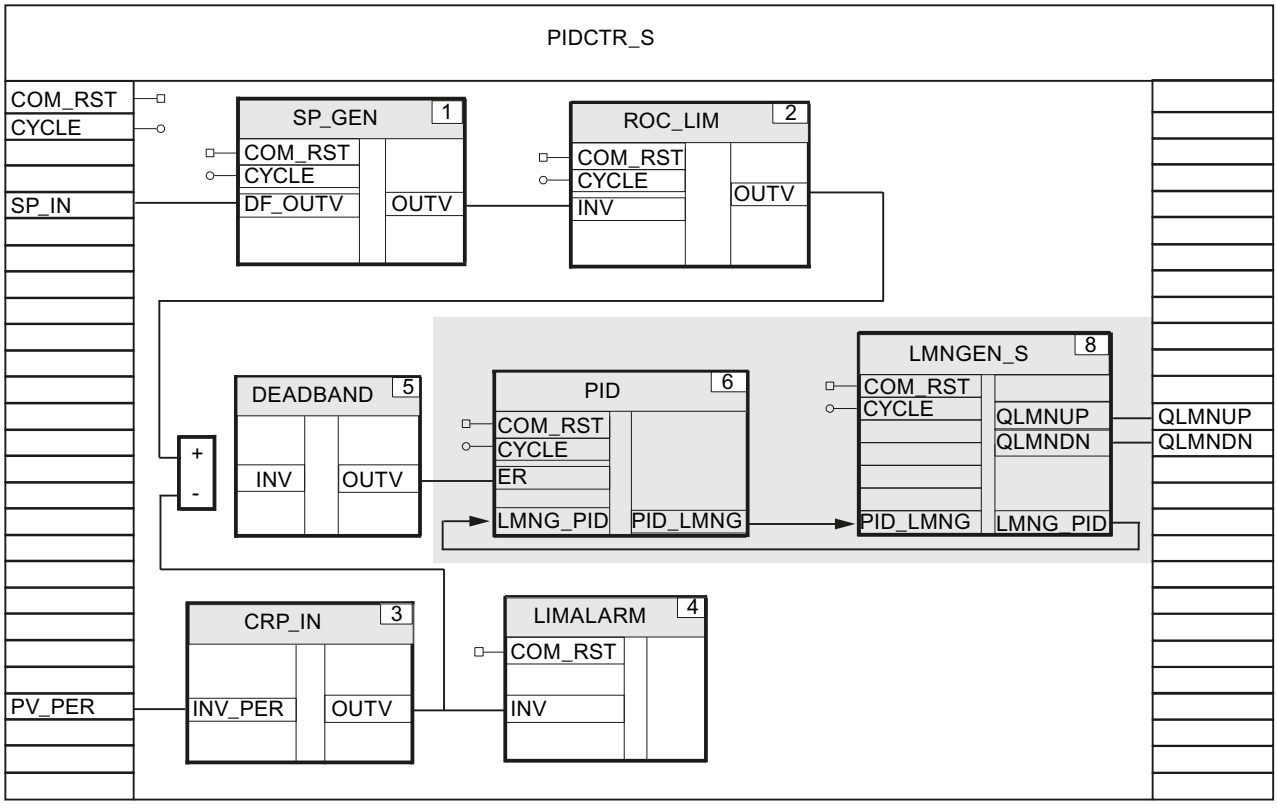

See also

Applying the disturbance variable (Page 906)

Processing position feedback (Page 914)

Processing position feedback (S7-300, S7-400)

Position feedback configuration depends upon the actuator used.

- Actuator with analog position feedback
- Actuator with digital endstop signals

Requirement

You have already programmed a step controller.

Creating a step controller with analog position feedback

The following instructions are useful for processing analog position feedback.

- CRP_IN: Scale I/O value
- NONLIN: Linearize position feedback

- SCALE_M: Scale position feedback
- NORM: Normalize position feedback

Example of analog position feedback

An actuator provides position feedback in I/O format; this should to be factored into the manipulated variable calculation.

1. At the controller FB, declare an input parameter LMNR_PER of the data type "Word".
2. Connect LMNR_PER to the I/O input.
3. Call the instruction CRP_IN as a multi-instance in the controller FB, for example with the name CRP_IN_Instance_LMNR.
4. Connect LMNR_PER to CRP_IN.INV_PER.
5. Assign parameters for factor and offset for scaling in the CRP_IN_Instance_LMNR structure of the multi-instance.
6. Connect CRP_IN.OUTV to LMNGEN_S.LMNR_IN using a temporary variable.
7. Activate position feedback with LMNR_ON = TRUE in the LMNGEN_S_Instance structure of the multi-instance.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR_SAF.

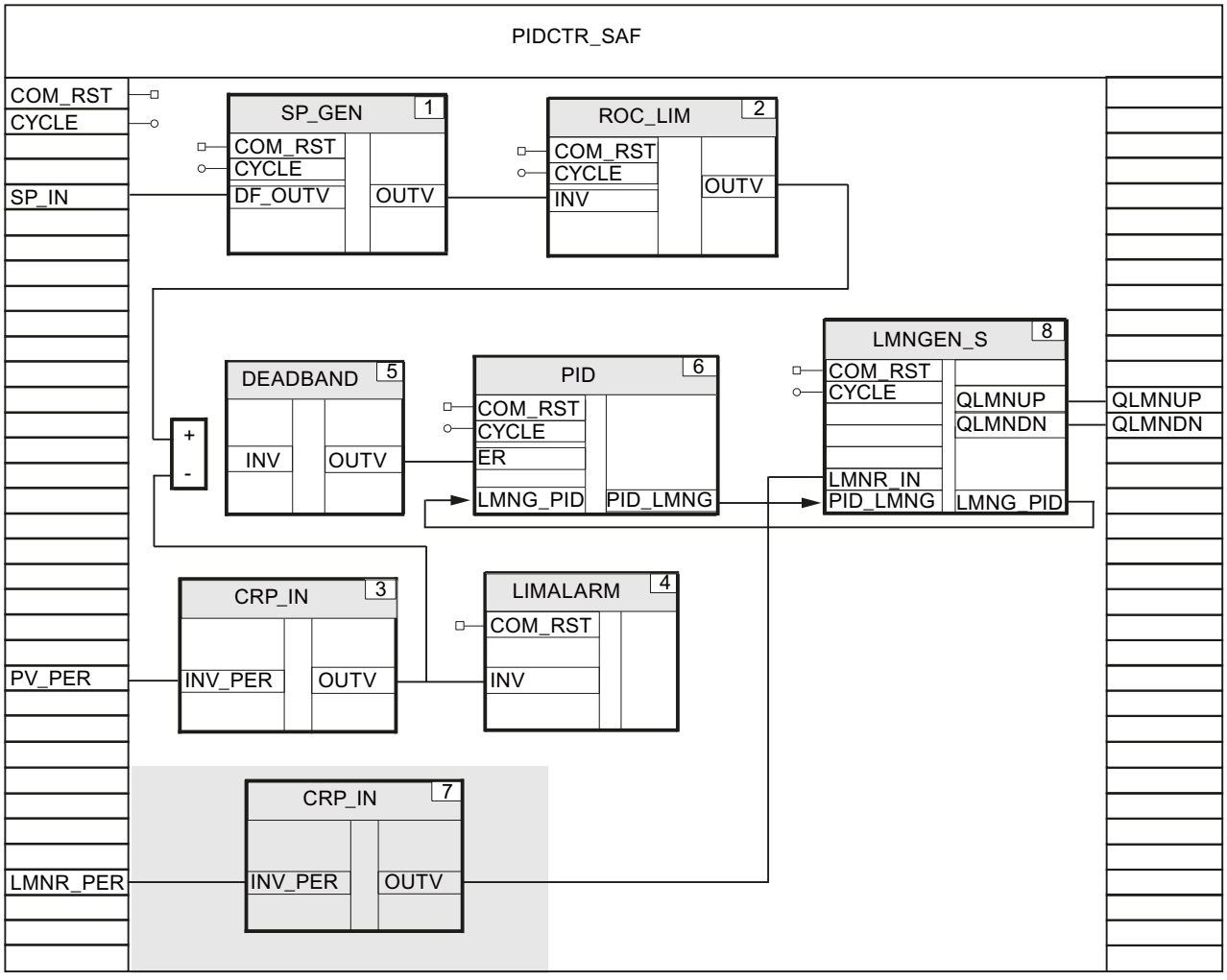

Creating a step controller with digital position feedback

1. At the controller FB, declare the input parameters LMNR_HS and LMNR_LS of the data type "Bool".
2. Connect the input parameters LMNR_HS and LMNR_LS to LMNGEN_S.LMNR_HS and LMNGEN_S.LMNR_LS.
3. Activate position feedback with LMNR_ON = TRUE in the LMNGEN_S_Instance structure.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR_SDF.

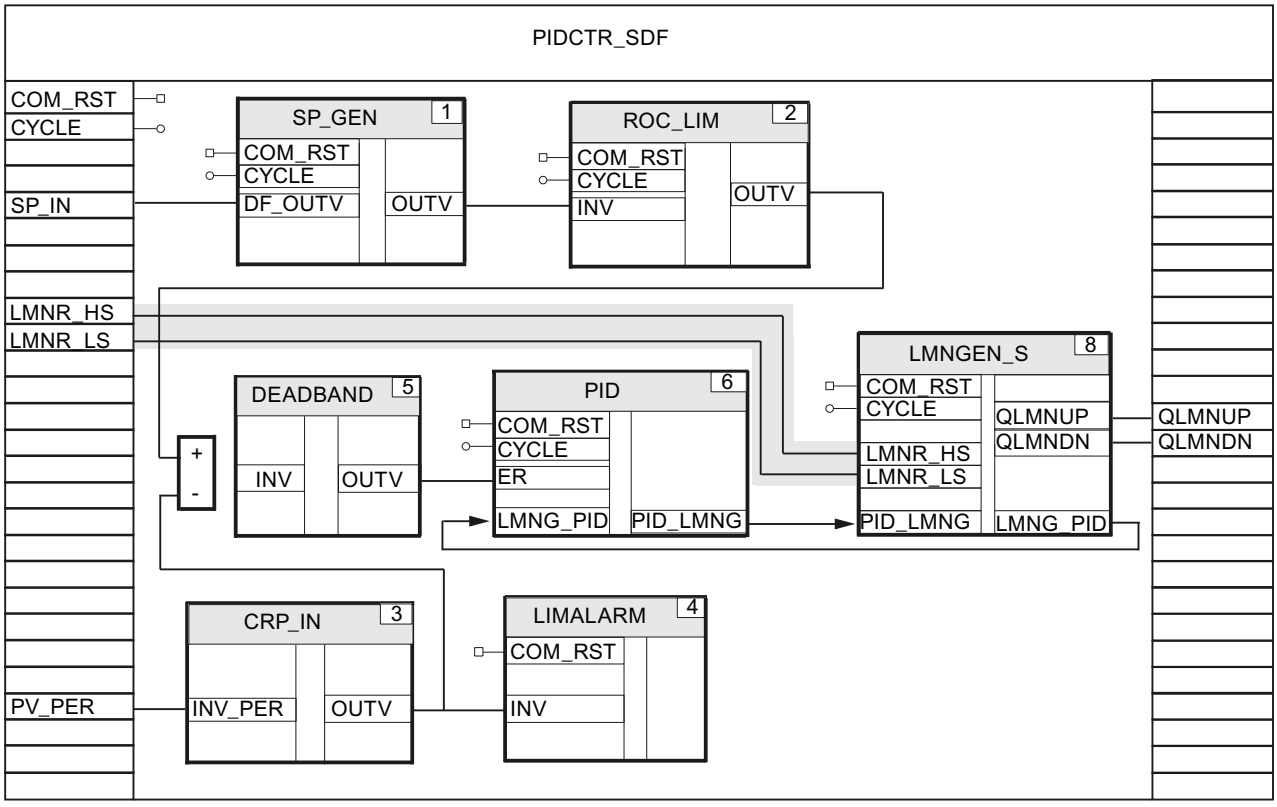

See also

Creating a step controller (Page 913)

3.5.2.4 Complex controller structures (S7-300, S7-400)

Single-loop ratio controller (S7-300, S7-400)

The diagram below shows a typical control loop of a single-loop ratio controller.

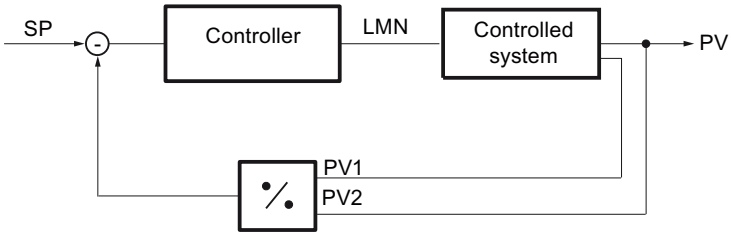

The ratio between two process values is adjusted to a defined setpoint with a PID algorithm.

Procedure

Proceed as follows to create a single-loop ratio controller:

1. Program process value processing as outlined below in your controller FB.
2. To avoid division by zero, you must limit the value range for the process value PV_PER2 to Values > Zero with the instruction LIMITER.

The diagram below shows a controller FB, in this case RATIOCTR, for a single-loop ratio controller with a continual manipulated variable.

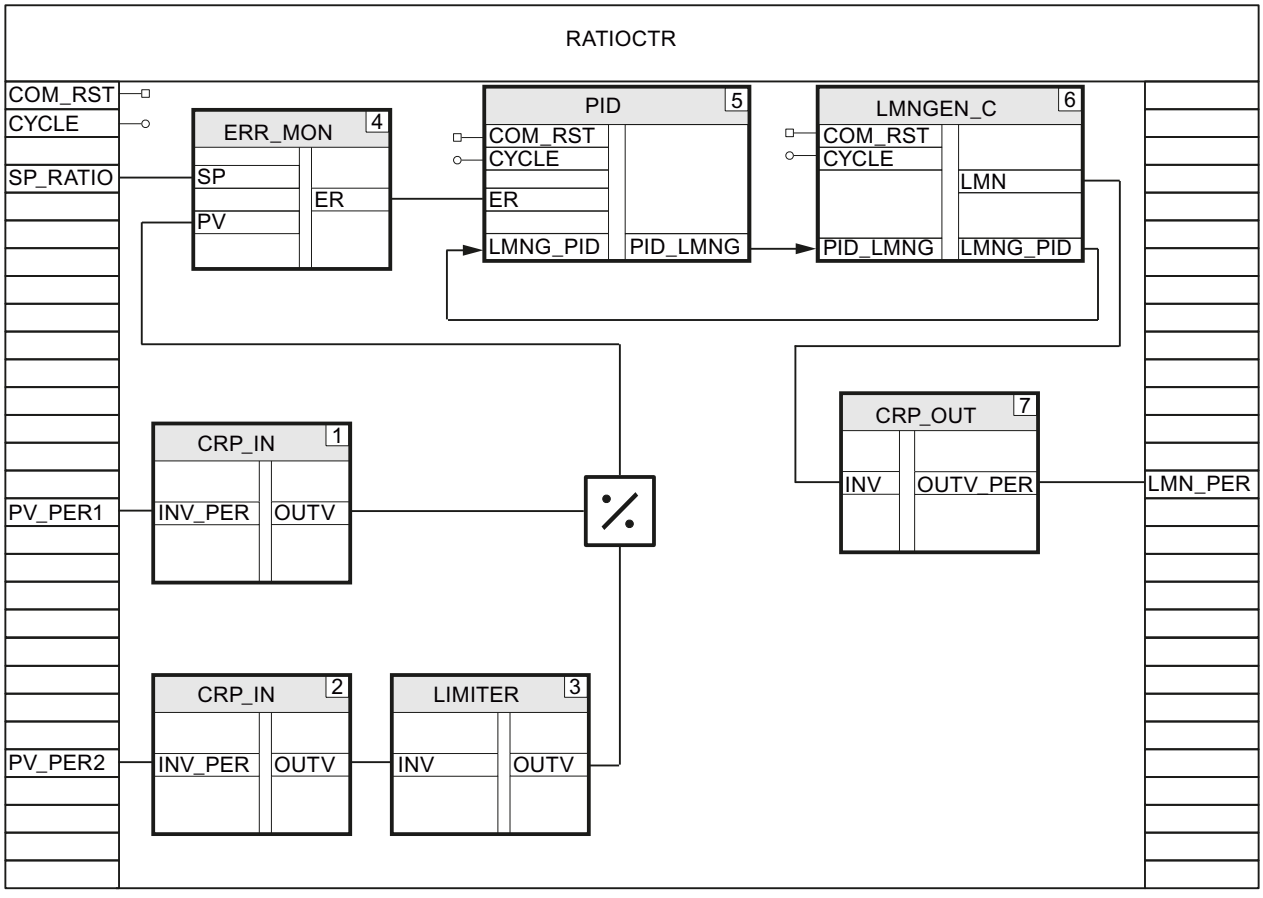

Multi-loop ratio controller (S7-300, S7-400)

The diagram below shows a typical control loop for a multi-loop ratio controller.

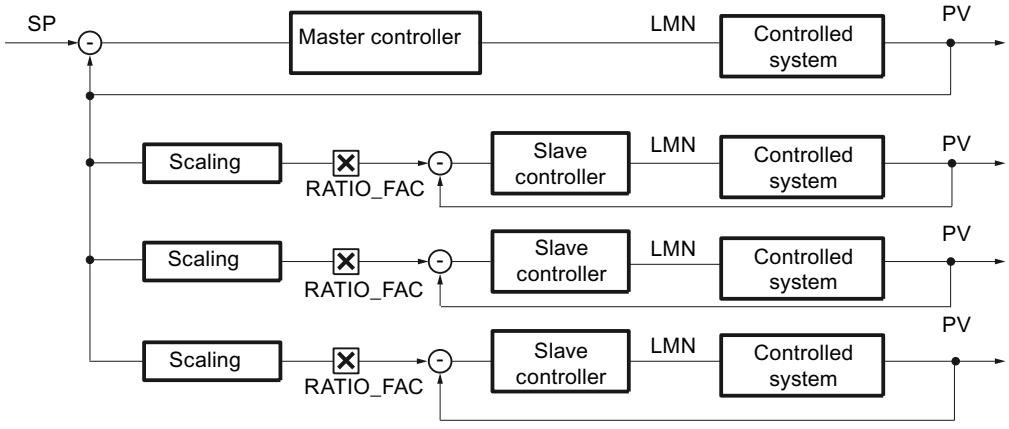

Programming a control loop

Follow these steps to program this example with Modular PID Control:

1. Program a controller FB for the master controller, e.g. PIDCTR_PV.
2. Program one or more controller FBs for the slave controllers, e.g. RB_CTR_S.
3. Program a function in which you can call and connect the four controllers.

Programming a master controller

Proceed as follows to program the master controller:

1. Program the master controller in line with your requirements as described in section Programming the controller FB (Page 901).
2. At the controller FB, in this case PIDCTR_PV, declare an output parameter PV of the data type "Real" at which you output the processed process value of the master controller.

The diagram below shows the master controller interconnection, using the controller FB PIDCTR_PV as an example.

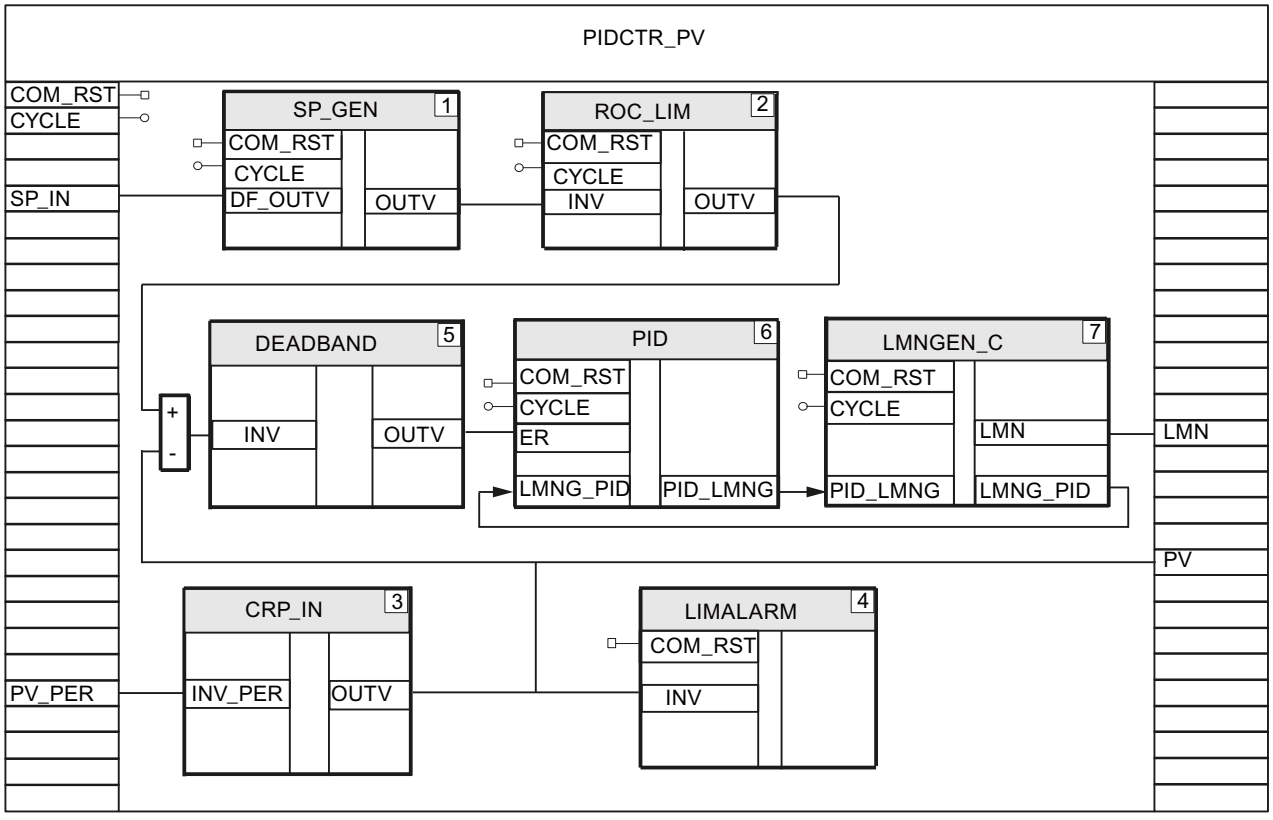

Programming slave controllers

Proceed as follows to program the slave controllers:

1. Program the slave controllers in line with your requirements as described in section Programming the controller FB (Page 901).
2. At the controller FB, in this case RB_CTR_S, declare an input parameter RATIO_FAC of the data type "Int". The processed setpoint, e.g. SCALE.OUTV, of the slave controller is multiplied by this value.

The diagram below shows the slave controller interconnection, using the controller FB RB_CTR_S as an example.

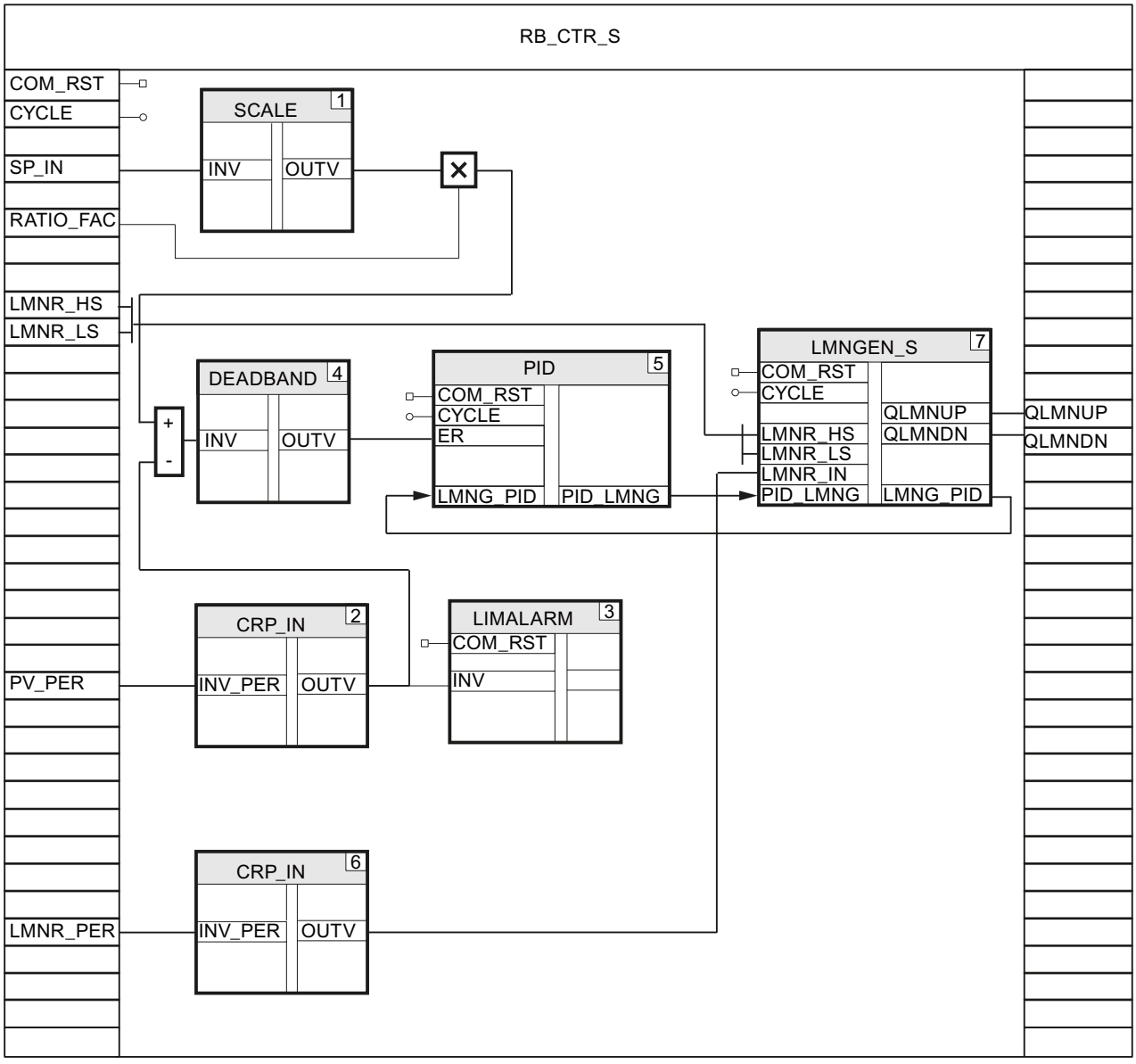

Calling and connecting controllers

Proceed as follows to call and connect master and slave controllers:

1. Create a function, e.g. FC50,, with the input parameters COM_RST and CYCLE.
2. Call the LP_SCHED_M instruction in the function.
3. Create a AUTOHOTSPOT in which to assign parameters for the number of controllers and the cycle times for calling the master controller and slave controller.
   The cycle times of the slave controllers must be shorter than or equal to the cycle time of the master controller.
4. Call the controller FB for the master controller, in this case PIDCTR_PV, if DB_LOOP.LOOP_DAT[1].ENABLE = TRUE.

5. Connect DB_LOOP.LOOP_DAT[1].COM_RST and DB_LOOP.LOOP_DAT[1].CYCLE to the input parameters COM_RST and CYCLE of the master controller.
6. Set DB_LOOP.LOOP_DAT[1].ENABLE = FALSE.
7. Call the controller FB for the first slave controller, in this case RB_CTR_S, if DB_LOOP.LOOP_DAT[2].ENABLE = TRUE.
8. Connect DB_LOOP.LOOP_DAT[2].COM_RST and DB_LOOP.LOOP_DAT[2].CYCLE to COM_RST and CYCLE of the slave controller.
9. Connect the output parameter PV of the master controller to the input parameter SP_IN of the slave controller.
10. Set DB_LOOP.LOOP_DAT[2].ENABLE = FALSE.
11. Repeat the last four steps for the two other slave controllers. Increase the array counter for each controller.
12. Connect the manipulated variables to the outputs of the CPU.
13. Call the function in OB 100 and in the cyclic interrupt OB.

The diagram below shows controller calls and interconnection, using the function FC50 as an example.

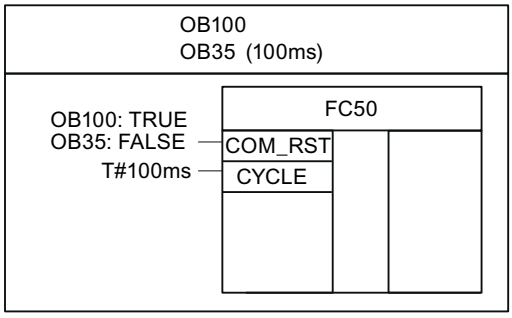

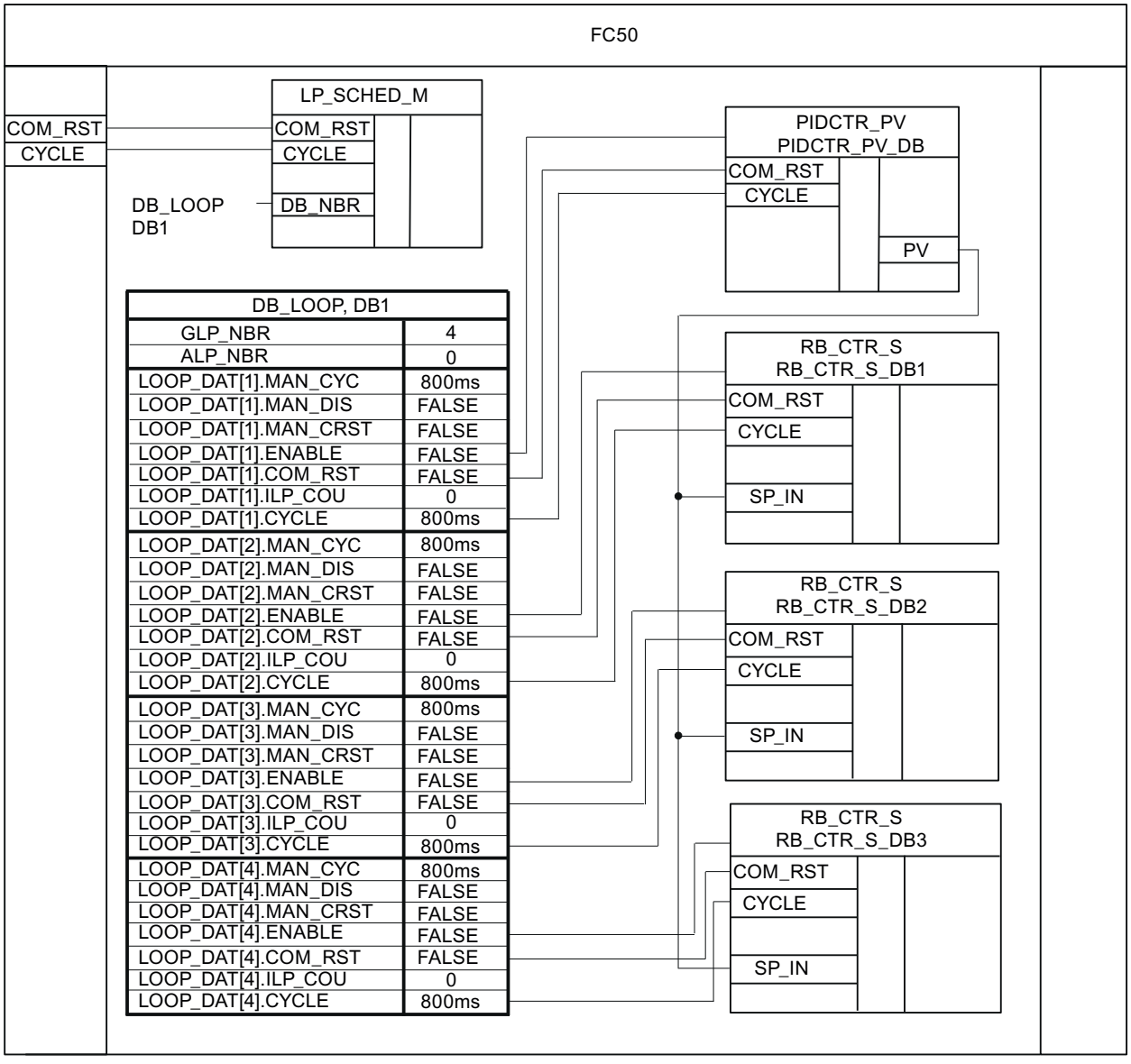

DB_LOOP, DB1	
GLP_NBR	4
ALP_NBR	0
LOOP_DAT[1].MAN_CYC	800ms
LOOP_DAT[1].MAN_DIS	FALSE
LOOP_DAT[1].MAN_CRST	FALSE
LOOP_DAT[1].ENABLE	FALSE
LOOP_DAT[1].COM_RST	FALSE
LOOP_DAT[1].ILP_COU	0
LOOP_DAT[1].CYCLE	800ms
LOOP_DAT[2].MAN_CYC	800ms
LOOP_DAT[2].MAN_DIS	FALSE
LOOP_DAT[2].MAN_CRST	FALSE
LOOP_DAT[2].ENABLE	FALSE
LOOP_DAT[2].COM_RST	FALSE
LOOP_DAT[2].ILP_COU	0
LOOP_DAT[2].CYCLE	800ms
LOOP_DAT[3].MAN_CYC	800ms
LOOP_DAT[3].MAN_DIS	FALSE
LOOP_DAT[3].MAN_CRST	FALSE
LOOP_DAT[3].ENABLE	FALSE
LOOP_DAT[3].COM_RST	FALSE
LOOP_DAT[3].ILP_COU	0
LOOP_DAT[3].CYCLE	800ms
LOOP_DAT[4].MAN_CYC	800ms
LOOP_DAT[4].MAN_DIS	FALSE
LOOP_DAT[4].MAN_CRST	FALSE
LOOP_DAT[4].ENABLE	FALSE
LOOP_DAT[4].COM_RST	FALSE
LOOP_DAT[4].ILP_COU	0
LOOP_DAT[4].CYCLE	800ms

Cascade controller (S7-300, S7-400)

The diagram below shows a typical control loop for a cascade controller.

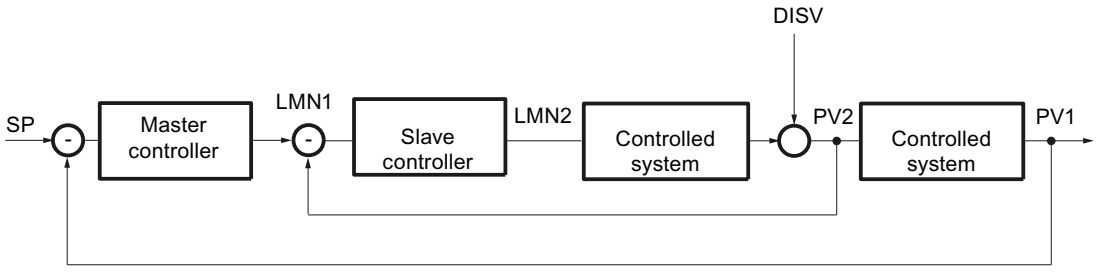

Programming a control loop

Follow these steps to program this example with Modular PID Control:

1. Program a controller FB for the master controller, e.g. PIDCTR.
2. Program a controller FB for the slave controller, e.g. PIDCTR_SAF.
3. Program a function in which you can call and connect the two controllers.
4. Call the LP_SCHED_M instruction in the function.
5. Create a AUTOHOTSPOT in which to assign parameters for the number of controllers and the cycle times for calling the master controller and slave controller.
   The cycle time of the slave controller must be less than or equal to the cycle time of the master controller.
6. Call the controller FB for the master controller if DB_LOOP.LOOP_DAT[1].ENABLE = TRUE.
7. Connect DB_LOOP.LOOP_DAT[1].COM_RST and DB_LOOP.LOOP_DAT[1].CYCLE to the input parameters COM_RST and CYCLE of the master controller.
8. Set DB_LOOP.LOOP_DAT[1].ENABLE = FALSE.
9. Call the controller FB for the slave controller if DB_LOOP.LOOP_DAT[2].ENABLE = TRUE.
10. Connect DB_LOOP.LOOP_DAT[2].COM_RST and DB_LOOP.LOOP_DAT[2].CYCLE to COM_RST and CYCLE of the slave controller.
11. Connect the output parameter LMN of the master controller to the input parameter SP_IN of the slave controller.
12. Set DB_LOOP.LOOP_DAT[2].ENABLE = FALSE.
13. Connect the manipulated variable of the slave controller to the analog output of the CPU which controls the actuator.
14. Call the function in OB 100 and in the cyclic interrupt OB.

The diagram below shows controller calls and interconnection, using the function FC50 as an example.

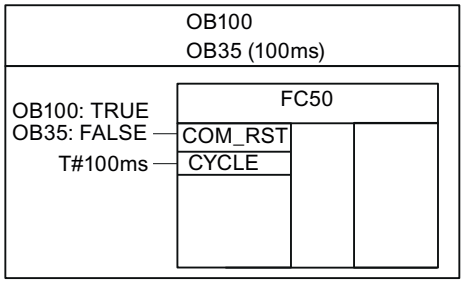

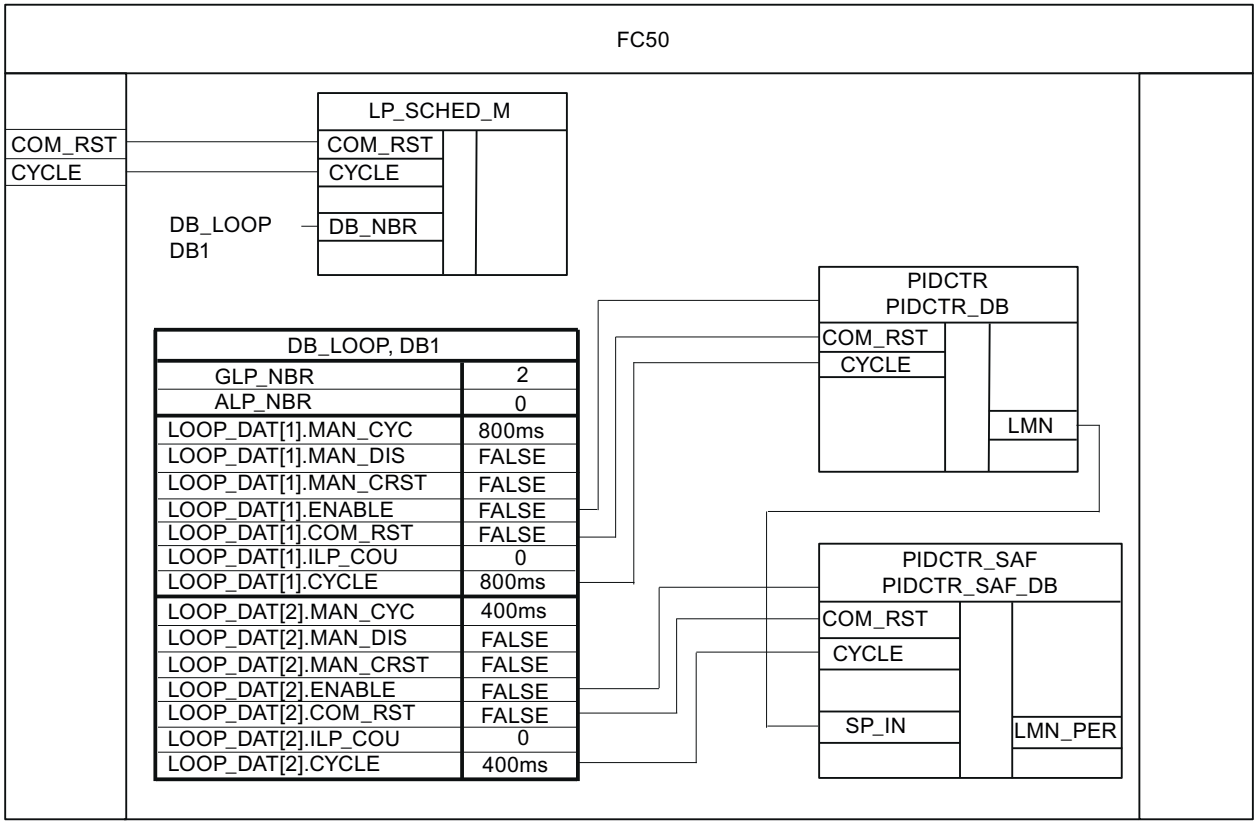

DB_LOOP, DB1	
GLP_NBR	2
ALP_NBR	0
LOOP_DAT[1].MAN_CYC	800ms
LOOP_DAT[1].MAN_DIS	FALSE
LOOP_DAT[1].MAN_CRST	FALSE
LOOP_DAT[1].ENABLE	FALSE
LOOP_DAT[1].COM_RST	FALSE
LOOP_DAT[1].ILP_COU	0
LOOP_DAT[1].CYCLE	800ms
LOOP_DAT[2].MAN_CYC	400ms
LOOP_DAT[2].MAN_DIS	FALSE
LOOP_DAT[2].MAN_CRST	FALSE
LOOP_DAT[2].ENABLE	FALSE
LOOP_DAT[2].COM_RST	FALSE
LOOP_DAT[2].ILP_COU	0
LOOP_DAT[2].CYCLE	400ms

Multi-variable controller (S7-300, S7-400)

The diagram below shows a typical control loop for a multi-variable controller. Two PID controllers with continuous output are switched to the process.

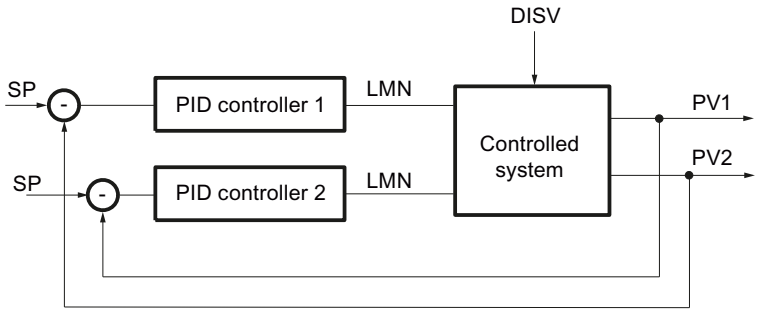

Programming a control loop

Follow these steps to program this example with Modular PID Control:

1. At the controller FB, in this case MUL_CTR, declare the following input parameters:
   - COM_RST of the data type "Bool"
   - CYCLE of the data type "Time"
   - SP_IN1 and SP_IN2of the data type "Real"
   - PV_PER1 and PV_PER2 of the data type "Word"
2. At the controller FB, declare the output parameters LMN_PER1 and LMN_PER2 of the data type "Word".
3. Program setpoint and process value processing and calculation of the manipulated variable for both PID controllers.
4. To calculate the control deviation of the first PID controller, subtract the processed process value PV_PER1 from the product of the setpoint SP_IN1 and the manipulated variable of the second PID controller.
5. To calculate the control deviation of the second PID controller, subtract the processed process value PV_PER2 from the product of the setpointSP_IN2 and the manipulated variable of the first PID controller.
6. Connect the manipulated variables LMN_PER1 and LMN_PER2 to two analog outputs which control the actuators.
7. Call the controller FB in OB 100 and in the cyclic interrupt OB.

The diagram below shows the call, using the controller FB MUL_CTR as an example.

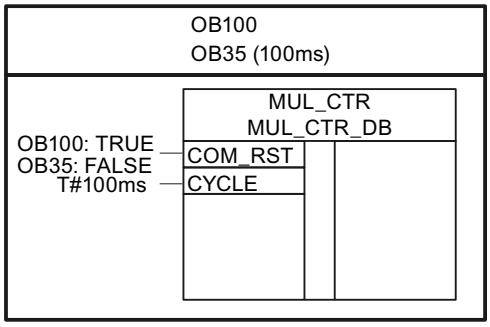

The diagram below shows the controller interconnection, using the controller FB MUL_CTR as an example.

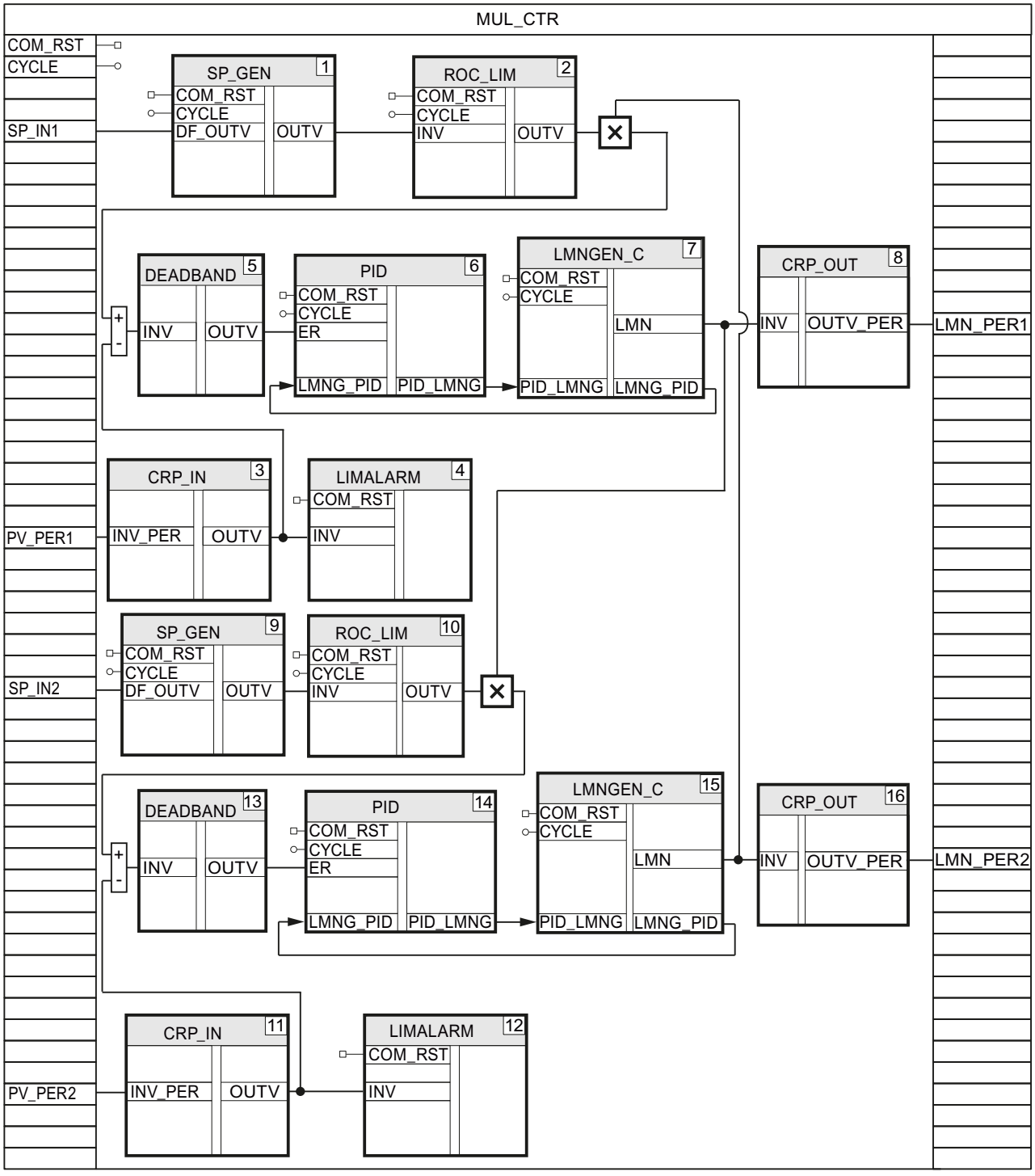
MUL_CTR
COM_RST
CYCLE
SP_IN1
SP_GEN 1
COM_RST
CYCLE
DF_OUTV
OUTV
ROC_LIM 2
COM_RST
CYCLE
INV
OUTV
DEADBAND 5
INV
OUTV
PID 6
COM_RST
CYCLE
ER
LMNG_PID
PID_LMNG
LMNGEN_C 7
COM_RST
CYCLE
LMN
PID_LMNG
LMNG_PID
CRP_OUT 8
INV
OUTV_PER
LMN_PER1
CRP_IN 3
INV_PER
OUTV
PV_PER1
LIMALARM 4
COM_RST
INV
SP_GEN 9
COM_RST
CYCLE
DF_OUTV
OUTV
SP_IN2
ROC_LIM 10
COM_RST
CYCLE
INV
OUTV
DEADBAND 13
INV
OUTV
PID 14
COM_RST
CYCLE
ER
LMNG_PID
PID_LMNG
LMNGEN_C 15
COM_RST
CYCLE
LMN
PID_LMNG
LMNG_PID
CRP_OUT 16
INV
OUTV_PER
LMN_PER2
CRP_IN 11
INV_PER
OUTV
PV_PER2
LIMALARM 12
COM_RST
INV

3.5.2.5 Commissioning the controller FB (S7-300, S7-400)

Commissioning the continual controller (S7-300, S7-400)

You can commission your controller FB with the PID Self-Tuner and determine the optimum PID parameters.

Procedure

Proceed as follows to commission a continual controller or a pulse controller:

1. Call the controller FB in a cyclic interrupt OB and connect the input and output parameters.
2. Call the instruction TUN_EC in the same cyclic interrupt OB.
3. Connect the instruction TUN_EC to your controller FB as follows:
   - TUN_EC.PV to the processed process value, e.g. the static variable CRP_IN_Instance_PV.OUTV or the output parameter PV
   - TUN_EC.LMN to the manipulated variable in real format, e.g. the static variable LMNGEN_C_Instance.LMN
   - TUN_EC.MAN to the in/out parameter for the manual value, e.g. MAN.
   - TUN_EC.SP_OUT to the input parameter for the setpoint, e.g. SP_IN
   - TUN_EC.GAIN to the proportional gain, e.g. PID_Instance.GAIN
   - TUN_EC.TI to the integral action time, e.g. PID_Instance.TI
   - TUN_EC.TD to the derivative action time, e.g. PID_Instance.TD
   - TUN_EC.TM_LAG to the time delay, e.g. PID_Instance.TM_LAG
4. Click the "" icon in the TUN_EC instruction.
   The commissioning editor for TUN_EC opens. The operation of the commissioning editor is explained under PID Self-Tuner.

Commissioning the step controller (S7-300, S7-400)

You can commission your controller FB with the PID Self-Tuner and determine the optimum PID parameters.

Procedure

Follow the steps below to commission a step controller:

1. Call the controller FB in a cyclic interrupt OB and connect the input and output parameters.
2. Call the instruction TUN_ES in the same cyclic interrupt OB.

3. Connect the instruction TUN_ES to your controller FB as follows:
   - TUN_ES.PV to the processed process value, e.g. the static variable CRP_IN_Instance _PV.OUTV or the output parameter PV
   - TUN_ES.LMNR_IN to the processed position feedback, e.g. LMNGEN_S_instance.LMNR_IN
   - TUN_ES.C_LMNUP and TUN_ES.C_LMNDN to the manipulated variable signals, e.g. the output parameters QLMN_UP and QLMNDN
   - TUN_ES.LMNR_HS to the high endstop signal, e.g. LMNGEN_S_Instance.LMNR_HS
   - TUN_ES.LMNR_ON to the position feedback switch, e.g. LMNGEN_S_Instance.LMNR_ON
   - TUN_ES.SP_OUT to the input parameter for the setpoint, e.g. SP_IN
   - TUN_ES.GAIN to the proportional gain, e.g. PID_Instance.GAIN
   - TUN_ES.TI to the integral action time, e.g. PID_Instance.TI
   - TUN_ES.TD to the derivative action time, e.g. PID_Instance.TD
   - TUN_ES.TM_LAG to the time delay, e.g. PID_Instance.TM_LAG
   - TUN_ES.MTR_TM to the motor transition time, e.g. LMNGEN_S_Instance.MTR_TM
   - If you are using a deadband, TUN_ES.DEADB_W to the deadband width, e.g. DEADBAND_Instance.OUTV
   - TUN_ES.QLMNUP and TUN_ES.QLMNDN to the manipulated variable signals LMNGEN_S_Instance.LMNUP and LMNGEN_S_Instance.LMNDN
4. Click the " " icon in the TUN_ES instruction.
   The commissioning editor for TUN_ES opens. The operation of the commissioning editor is explained under PID Self-Tuner.

3.5.3 Using PID Self Tuner (S7-300, S7-400)

3.5.3.1 Tuning continuous controllers and pulse controllers (S7-300, S7-400)

Interconnecting TUN_EC with controller (S7-300, S7-400)

The TUN_EC instruction is used to tune a continuous controller or a pulse controller. Depending on the controller, you interconnect TUN_EC with one of the following instructions:

Controller	Instruction
FM 355 C, FM 355 S pulse controller	PID_FM
FM 455 C, FM 455 S pulse controller	PID_FM
PID Basic Functions	CONT_C with or without PULSEGEN

Controller	Instruction
Standard PID Control (PID Professional optional package)	PID_CP
Modular PID Control (PID Professional optional package)	PID LMNGEN_C

Interconnect the input and output parameters of the TUN_EC instruction with the instruction of the controller as follows.

TUN_EC	PID_FM	CONT_C	PID_CP
PV	PV	PV	PV
LMN	LMN	LMN	LMN
NORM_FAC	--	--	(NM_PVHR-NM_PVLR)*0.01
WRITE_DIS	OR operation from LOAD_OP and LOAD_PAR	--	--
MAN_OUT	LMN_RE	MAN	MAN
SP_OUT	SP_RE	SP_INT	SP_INT
GAIN	GAIN	GAIN	GAIN
TI	TI	TI	TI
TD	TD	TD	TD
TM_LAG	TM_LAG	TM_LAG	TM_LAG
RATIOFAC	--	--	RATIOFAC
PHASE	--	--	PHASE
QMAN_ON	LMN_REON	MAN_ON	MAN_ON
QI_SEL	--	I_SEL	I_SEL
QD_SEL	--	D_SEL	D_SEL
QWRITE	LOAD_PAR and LOAD_OP	--	--

Tuning options (S7-300, S7-400)

Pretuning

The PID parameters are tuned during the heating process. During this tuning, process identification with subsequent controller design is carried out.

You can request tuning with ADAPT1ST.

When you set ADAPT1ST = TRUE, then ADAPT_ON = TRUE is set, which forces pretuning to be performed. For pretuning, you specify the manipulated variable jump at the LHLM_TUN parameter.

The ADAPT1ST parameter is only required if serious adaptation errors resulted in process parameters that make the automatically calculated manipulated variable jump unsuitable for a re-adaptation.

When ADAPT1ST = TRUE , the initialization routine of the TUN_EC instruction is run. The following pre-assignments are made:

- The system switches to PI controller. The parameters of PI_CON are transferred to the controller for this purpose. If the PI_CON.GAIN and PI_CON.TI parameters both have the value 0.0, they are pre-assigned with PI_CON.GAIN = 1.5 and PI_CON.TI = 3600.0 s.
- PROCESS.GAIN = 999.0
- RATIOFAC = 1.0
- STATUS_H = 0; STATUS_D = 0; STATUS_C = 0

Note

Unless explicitly required, do **not** call TUN_EC during a warm restart of the CPU in its initialization routine as is usual for other instructions. If you do, all process parameters will be lost.

Tuning is started with ADAPT1ST = TRUE and a subsequent setpoint jump ≥ MIN_STEP in the positive direction.

As a rule, the setpoint jump changes from the setpoint of the cold process (ambient temperature) to a point close to the operating point during pretuning. Any setpoint can be used as the initial state as long as it is ensured that the process is in a steady state. You are not permitted to specify any further setpoint jumps during tuning.

Pretuning trend characteristics

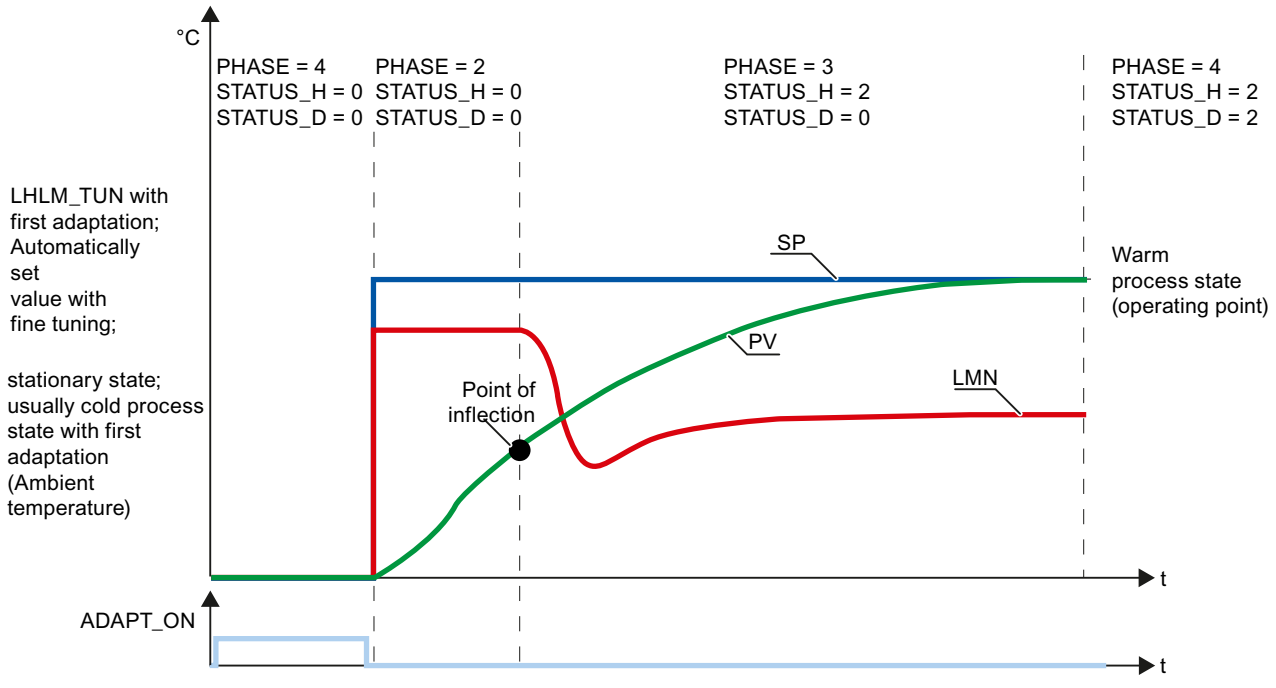

Fine tuning

Fine tuning is possible only after pretuning has been carried out correctly. With ADAPT_ON = TRUE, TUN_EC decides whether it will perform pretuning or fine tuning. If information about the process is already available, TUN_EC automatically calculates a manipulated variable jump and performs fine tuning. If information on the process is not available, pretuning is carried out.

Starting tuning

You can start tuning from either automatic mode or manual mode:

- **Starting tuning from automatic mode**
  In automatic mode, set ADAPT_ON or ADAPT1ST = TRUE and specify a setpoint jump ≥ MIN_STEP.
- **Starting tuning from manual mode**
  In manual mode, set ADAPT_ON or ADAPT1ST = TRUE and specify a setpoint jump ≥ MIN_STEP. You must then reset MAN_ON = FALSE in order for the setpoint jump to become effective and tuning to begin.

Cooling tuning

Cooling tuning is possible only after pretuning has been carried out correctly. It is started in PHASE 4. When COOLID_ON = TRUE is set, the controller is set to manual mode and LLLM_TUN is output: MAN_OUT = LLLM_TUN. TUN_EC switches to PHASE = 3, and COOLID_ON is reset again immediately. As soon as a point of inflection in the process value trend is found, TUN_EC switches back to closed-loop control (PHASE = 4). TUN_EC determines a ratio factor RATIOFAC, whose value is forwarded to the pulse generator of the controller.

Cooling tuning trend characteristics

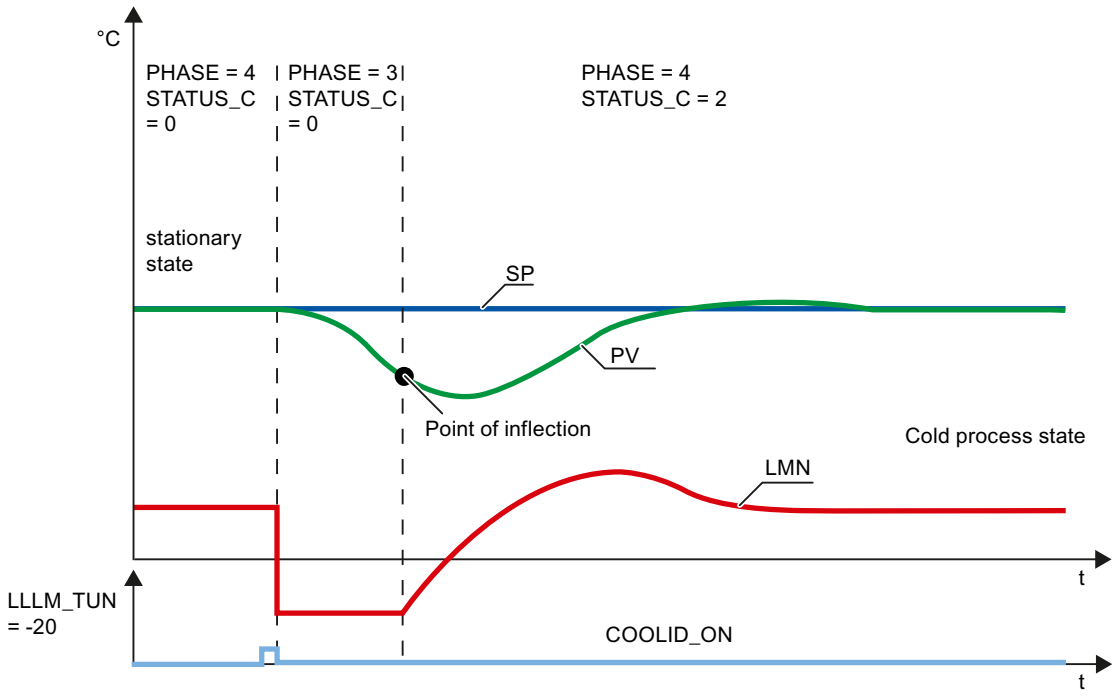

Note

- The COOLID_ON start bit can be set at the same time as ADAPT1ST or ADAPT_ON, or during tuning of the controller parameters. This ensures that cooling tuning can be started immediately after tuning of the controller parameters for heating has been completed.
- In the case of a plant with several thermally coupled temperature zones (e.g., plastics machines), cooling tuning should not be started until all (!) zones have finished pretuning; otherwise, the pretuning results are invalid.

Canceling tuning

To cancel the tuning before the setpoint jump, you must set ADAPT_ON = FALSE . To cancel tuning after the setpoint jump (PHASE 2 or 3) or cancel cooling tuning, you must switch to manual mode (MAN_ON = TRUE).

By setting ADAPT_ON, ADAPT1ST or COOLID_ON during active tuning (PHASE 2 or 3), the tuning is not canceled but is instead completed. Because ADAPT_ON, ADAPT1ST or COOLID_ON is not reset, tuning is restarted once the start conditions are met.

Tuning phases (S7-300, S7-400)

Individual phases are run through during controller tuning. You can read these phases at the PHASE parameter.

PHASE = 0

When an instance DB for TUN_EC is created, the PHASE parameter is preassigned with zero and MAN_ON with TRUE. For this reason, the system switches to manual mode (PHASE = 7) after the first cycle. You can start the pretuning directly from manual mode (ADAPT_ON = TRUE) or switch to automatic mode. When you switch to automatic mode MAN_ON = FALSE for the first time, the parameters of PI_CON are applied. The default values for GAIN and TI are: GAIN = 1.5, TI = 1h.

PHASE = 2

As soon as you specify a setpoint jump ≥ MIN_STEP in the positive direction toward the operating point of the warm process state, LHLM_TUN is assigned to MAN_OUT in the case of pretuning, while a value calculated by TUN_EC is assigned to MAN_OUT in the case of fine tuning. QMAN_ON = TRUE is also set. Both values are transferred to the PID controller. The PID controller is controlled with them in manual mode. MIN_STEP should be greater than 10% of the operating range of the setpoint and the process value.

PHASE = 3

When the point of inflection of the jump response is recognized (STATUS_H = 2) or when the process value has reached 70% of the setpoint jump height (STATUS_H = 3), a PI controller with conservative settings is designed. The controller gain GAIN is limited to the range from 0.2 to 20.0. As of this point, the controller immediately starts working as a PI controller and attempts to bring the controlled system to a steady-state condition. TUN_EC stores the newly determined PI controller parameters in GAIN, TI, and PI_CON. The previous values of the PI controller are saved in PI_CON_OLD. If it takes a long time to reach steady-state condition (creeping transient response during temperature processes), you can set STEADY = TRUE to initiate controller design with the current data when steady-state condition is almost reached. You can also set STEADY = TRUE to re-initiate the controller design with the current data at a later stage in PHASE = 3 or PHASE = 4. This will often provide some additional improvement to the controller design.

If an overshoot occurs or if no point of inflection is found, the manipulated variable jump size LHLM_TUN selected for pretuning may have been too large. This does not necessarily result in poor controller tuning. For the next pretuning, the value you select for LHLM_TUN should be approximately 20% smaller.

When TUN_EC has recognized steady-state condition or when time 30 × TI (TI: integral action time of the PI controller set in PHASE = 3) has elapsed since the setpoint jump, the controller design is improved and the system switches to PHASE = 4. TUN_ES stores the newly determined PI/PID controller parameters in GAIN, TI, TD, and PI_CON or PID_CON. The previous values of the PID controller are saved in PID_CON_OLD. If possible, a PID controller and a PI controller are designed. If PID_ON = TRUE, the PID controller is activated. Otherwise, the PI controller is activated. GAIN is limited in such a way that the loop gain of the open loop (product of GAIN and process gain) is less than 80.0.

PHASE = 4

In this phase, the controller controls using its tuned parameters. From this state, you can start cooling tuning, fine tuning, or pretuning.

You can switch bumplessly between the PI controller and PID controller at any time.

Performing pretuning (S7-300, S7-400)

Requirements

- The instruction and the technology object are loaded on the CPU.
- TUN_EC is interconnected with a continuous controller or a pulse controller.
- MAN_ON = FALSE
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To determine the optimal PID parameters during initial commissioning, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Pretuning" from the "Mode" drop-down list.
   TUN_EC is ready for tuning.
3. Choose a controller structure.
4. In the "Manipulated variable" field, enter a manipulated variable to be used until the transition to PHASE = 3.
5. Enter a setpoint in the "Setpoint" field. The setpoint jump must be ≥ MIN_STEP. The new manipulated variable only takes effect when another setpoint is entered.
6. Click the "Start tuning" icon.
   Setpoint and manipulated variable are transferred to the controller. The pretuning starts. The status of the tuning is displayed.

Performing fine tuning (S7-300, S7-400)

Requirements

- The instruction and the technology object are loaded on the CPU.
- TUN_EC is interconnected with a continuous controller or a pulse controller.
- MAN_ON = FALSE
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To determine the optimal PID parameters at the operating point, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Fine tuning" from the "Mode" list.
   TUN_EC is ready for tuning.
3. Choose a controller structure.
4. Enter a setpoint in the "Setpoint" field. The setpoint jump must be ≥ MIN_STEP.
5. Click the ▶ "Start tuning" icon.
   The new setpoint is sent to the controller. TUN-EC calculates a manipulated variable jump. Fine tuning starts. The status of the tuning is displayed.

Performing cold tuning (S7-300, S7-400)

During cooling tuning, the ratio factor is determined for closed-loop controls with two opposing actuators (e.g. heating/cooling).

Requirements

- Pretuning has been carried out correctly.
- The instruction and the technology object are loaded on the CPU.
- TUN_EC is interconnected with a continuous controller or a pulse controller.
- MAN_ON = FALSE
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To determine the ratio factor, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Cooling tuning" from the "Mode" drop-down list.
   TUN_EC is ready for tuning.
3. Choose a controller structure.
4. In the "Manipulated variable" field, enter a manipulated variable that will be used until the point of inflection is found.
5. Click the ▶ "Start tuning" icon.
   Cooling tuning starts. The status of the tuning is displayed.

Performing tuning manually (S7-300, S7-400)

You can manually change the PID parameters if oscillations occur or overshoots following setpoint jump changes are present in the closed control loop after pretuning or fine tuning. For example, you can reset the controller gain GAIN to 80% of the original value and the integral action time TI to 150% of the original setting. In a pulse controller, small continuous oscillations may occur, which you can eliminate by expanding the dead zone DEADB_W.

Requirements

- The instruction and the technology object are loaded on the CPU.

Procedure

To manually determine the optimal PID parameters, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Manual" from the "Mode" list.
3. Choose a controller structure.
4. Enter new PID parameters in the "P", "I" and "D" fields.
5. Click the "Send parameters to the controller" icon in the "Tuning" group.
6. Select the "Change setpoint" check box in the "Current values" group.
7. Enter a new setpoint and click the icon in the "Current values" group.
8. Clear the "Manual mode" check box, if necessary.
   The controller works with the new PID parameters and controls to the new setpoint.
9. Check the quality of the PID parameters based on the trend characteristics.
10. Repeat steps 3 to 9 until you are satisfied with the controller results.

3.5.3.2 Tuning step controllers (S7-300, S7-400)

Interconnecting TUN_ES with controller (S7-300, S7-400)

The TUN_ES instruction is used to tune a step controller. Depending on the controller, you interconnect TUN_ES with one of the following instructions:

Controller	Instruction
FM 355 S step controller	PID_FM
FM 455 S step controller	PID_FM
PID Basic Functions	CONT_S TCONT_S
Standard PID Control (optional package)	PID_ES
Modular PID Control (optional package)	PID LMNGEN_S

Interconnect the input and output parameters of the TUN_ES instruction with the instruction of the controller.

TUN_ES	PID_FM	CONT_S	TCONT_S	PID_ES
PV	PV	PV	PV	PV
LMNR	LMN_A	--	--	LMNR_IN
NORM_FAC	--	--	--	(NM_PVHR-NM_PVLR)*0.01
C_LMNUP	QLMNUP	QLMNUP	QLMNUP	QLMNUP
C_LMNDN	QLMNDN	QLMNDN	QLMNDN	QLMNDN
LMNR_HS	QLMNR_HS	LMNR_HS	LMNR_HS	LMNR_HS
LMNR_ON	QLMNR_ON	--	--	LMNR_ON
WRITE_DIS	OR operation from LOAD_OP and LOAD_PAR	--	--	--
MAN_OUT	LMN_RE	--	--	MAN
SP_OUT	SP_RE	SP_INT	SP_INT	SP_INT or SP_EXT
GAIN	GAIN	GAIN	GAIN	GAIN
TI	TI	TI	TI	TI
TD	TD	--	--	TD
TM_LAG	TM_LAG	--	--	TM_LAG
MTR_TM	MTR_TM	MTR_TM	MTR_TM	MTR_TM
DEADB_W	DEADB_W	DEADB_W	DEADB_W	DEADB_W
PHASE	--	--	--	PHASE
QMAN_ON	LMN_REON	--	--	MAN_ON
QLMNS_ON	LMNSOPON	LMNS_ON	LMNS_ON	LMNS_ON
QLMNUP	LMNUP_OP	LMNUP	LMNUP	LMNUP
QLMNDN	LMNDN_OP	LMNDN	LMNDN	LMNDN
QI_SEL	--	--	--	I_SEL
QD_SEL	--	--	--	D_SEL
QWRITE	LOAD_PAR and LOAD_OP	--	--	--

Tuning options (S7-300, S7-400)

Pretuning

The PID parameters are tuned during the heating process. During this tuning, process identification with subsequent controller design is carried out.

You can request pretuning with ADAPT1ST.

If you set ADAPT1ST = TRUE, then ADAPT_ON = TRUE and the valve are closed (PHASE = 1).

When ADAPT1ST = TRUE , the initialization routine of the TUN_ES instruction is run. The following pre-assignments are made:

- The system switches to PI controller. The parameters of PI_CON are transferred to the controller for this purpose. If the PI_CON.GAIN and PI_CON.TI parameters both have the value 0.0, they are pre-assigned with PI_CON.GAIN = 1.5 and PI_CON.TI = 3600.0 s.
- PROCESS.GAIN = 999.0
- STATUS_H = 0; STATUS_D = 0

Note

Unless explicitly required, do **not** call TUN_ES during a warm restart of the CPU in its initialization routine as is usual for other instructions. If you do, all process parameters will be lost.

After a setpoint jump ≥ MIN_STEP, the motor transition time is measured after which pretuning is carried out. For process excitation during pretuning, TUN_ES uses the LHLM_TUN parameter for the manipulated variable jump. In the case of step controllers with position feedback, the value LHLM_TUN is approached directly. In the case of step controllers without position feedback, the control valve is opened completely and then moved back toward the closed position until the fraction of the motor transition time specified by (100 - LHLM_TUN) has elapsed.

Pretuning with measurement of motor transition time for step controller without position feedback

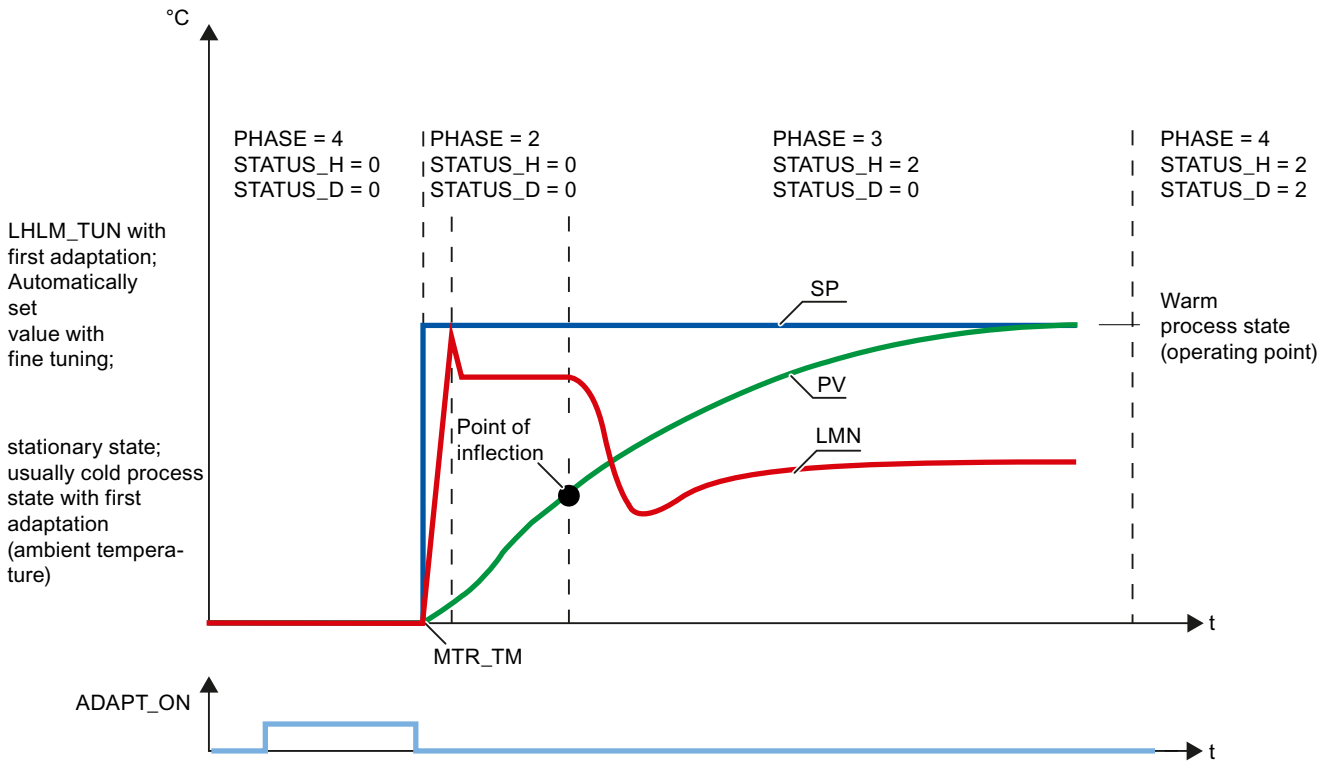

Fine tuning

You can use ADAPT_ON to start the tuning only for step controllers with position feedback. TUN_ES uses old process values to determine whether pretuning without measurement of the motor transition time or fine tuning will be performed. Tuning is started with ADAPT_ON = TRUE and a subsequent setpoint jump ≥ MIN_STEP in the positive direction. TUN_ES then immediately enters PHASE = 2 and performs a manipulated variable jump to LHLM_TUN in the case of pretuning or to an automatically calculated manipulated variable in the case of fine tuning.

As a rule, the setpoint jump changes from the setpoint of the cold process (ambient temperature) to a point close to the operating point during pretuning. Any setpoint can be used as the initial state as long as it is ensured that it is a steady state. You are not permitted to specify any further setpoint jumps during tuning.

Starting tuning

You can start tuning from either automatic mode or manual mode:

- **Starting tuning from automatic mode**
  In automatic mode, set ADAPT_ON or ADAPT1ST = TRUE and specify a setpoint jump ≥ MIN_STEP.
- **Starting tuning from manual mode**
  In manual mode, set ADAPT_ON or ADAPT1ST = TRUE and specify a setpoint jump ≥ MIN_STEP. You must then reset MAN_ON = FALSE in order for the setpoint jump to become effective and tuning to begin.

Canceling tuning

To cancel the tuning before the setpoint jump, you must set ADAPT_ON = FALSE . To cancel the tuning after the setpoint jump (PHASE 2 or 3), you must switch to manual mode (MAN_ON = TRUE or LMNS_ON = TRUE).

By setting ADAPT_ON or ADAPT1ST during active tuning (PHASE 2 or 3), the tuning is not canceled but is instead completed. Because ADAPT_ON or ADAPT1ST is not reset, tuning is restarted once the start conditions are met.

Tuning phases (S7-300, S7-400)

Individual phases are run through during controller tuning. You can read these phases at the PHASE parameter.

PHASE = 0

When an instance DB for TUN_ES is created, the PHASE parameter is preassigned with zero and LMNS_ON with TRUE. For this reason, the system switches to manual mode (PHASE = 7) after the first cycle. You can start the pretuning directly from manual mode (ADAPT_ON = TRUE) or switch to automatic mode. When you switch to automatic mode MAN_ON = FALSE for the first time, the parameters of PI_CON are applied. The default values for GAIN and TI are: GAIN = 1.5, TI = 1h.

PHASE = 1

After tuning has been started with ADAPT1ST = TRUE, TUN_ES closes the control valve (manipulated variable = 0 or LMNS_ON = LMNDN = TRUE) and waits for a positive setpoint jump ≥ MIN_STEP.

PHASE = 2

As soon as you specify a setpoint jump ≥ MIN_STEP in the positive direction toward the operating point of the warm process state, LHLM_TUN is assigned to MAN_OUT in the case of pretuning and a value calculated by TUN_ES is assigned to MAN_OUT in the case of fine tuning. MIN_STEP should be greater than 10% of the operating range of the setpoint and process value.

In the case of step control with position feedback (LMNR_ON=TRUE), the value TRUE is assigned to QMAN_ON. The controller controls the valve to the value of MAN_OUT, and TUN_ES calculates the motor transition time MTR_TM.

In the case of step control without position feedback, QLMNS_ON and QLMNUP are set to TRUE, and the valve is moved to the high endstop position. When the high endstop is reached (LMNR_HS =TRUE), TUN_ES calculates the motor transition time and forwards it to the controller. The valve is then moved toward the closed position until the value of MAN_OUT is reached (QLMNDN = TRUE).

PHASE = 3

When the point of inflection of the jump response is recognized (STATUS_H = 2) or when the process value has reached 70% of the setpoint jump height (STATUS_H = 3), a PI controller with conservative settings is designed. The controller gain GAIN is limited to the range from 0.2 to 20.0. As of this point, the controller immediately starts working as a PI controller and attempts to bring the controlled system to a steady-state condition. TUN_ES stores the newly determined PI controller parameters in GAIN, TI, and PI_CON. The previous values of the PI controller are saved in PI_CON_OLD. If it takes a long time to reach steady-state condition (creeping transient response during temperature processes), you can set STEADY = TRUE to initiate controller design with the current data when steady-state condition is almost reached. You can also set STEADY = TRUE to re-initiate the controller design with the current data at a later stage in PHASE = 3 or PHASE = 4. This will often provide some additional improvement to the controller design.

If an overshoot occurs or if no point of inflection is found, the manipulated variable jump size LHLM_TUN selected for pretuning may have been too large. This does not necessarily result in poor controller tuning. For the next pretuning, the value you select for LHLM_TUN should be approximately 20% smaller.

When TUN_ES has recognized steady-state condition or when time 30 × TI (TI: integral action time of the PI controller set in PHASE = 3) has elapsed since the setpoint jump, the controller design is improved and the system switches to PHASE = 4. TUN_ES stores the newly determined PI/PID controller parameters in GAIN, TI, TD, and PI_CON or PID_CON. The previous values of the PID controller are saved in PID_CON_OLD. If possible, a PID controller and a PI controller are designed. If PID_ON = TRUE, the PID controller is activated, otherwise the PI controller is activated. For complicated processes, TUN_ES always designs a PI controller, irrespective of PID_ON. GAIN is limited in such a way that the loop gain of the open loop (product of GAIN and process gain) is less than 40.0.

PHASE = 4

In this phase, the controller controls using its tuned parameters. From this state, you can start a fine tuning or a pretuning.

You can switch bumplessly between the PI controller and PID controller at any time.

Performing pretuning (S7-300, S7-400)

The controller parameters are pretuned during the heating process. During this tuning, process identification with subsequent controller design is carried out.

Requirements

- The instruction and the technology object are loaded on the CPU.
- TUN_ES is interconnected with a step controller.
- MAN_ON = FALSE
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To determine the optimal PID parameters during initial commissioning, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Pretuning" from the "Mode" drop-down list.
   TUN_ES is ready for tuning.
3. Choose a controller structure.
4. In the "Manipulated variable" field, enter a manipulated variable to be used until the transition to PHASE = 3.
5. Enter a new setpoint. The setpoint jump must be ≥ MIN_STEP.
6. Click the "Start tuning" icon.
   The pretuning starts. The status of the tuning is displayed.

Performing fine tuning (S7-300, S7-400)

For step controllers with position feedback, you can start fine tuning.
The controller can be fine tuned online during a positive setpoint jump when there is a change from one operating point to another or when there are changes in the process response.

Requirements

- The instruction and the technology object are loaded on the CPU.
- TUN_ES is interconnected with a step controller.

- MAN_ON = FALSE
- LMNR_ON = TRUE for the TUN_ES instruction and the controller.
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To determine the optimal PID parameters during initial commissioning, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Fine tuning" from the "Mode" list.
   TUN_ES is ready for tuning.
3. Choose a controller structure.
4. Enter a new setpoint. The setpoint jump must be ≥ MIN_STEP.
5. Click the "Start tuning" icon.
   Fine tuning starts. The manipulated variable is determined by TUN_ES. The status of the tuning is displayed.

Performing tuning manually (S7-300, S7-400)

You can manually change the PID parameters if oscillations occur or overshoots following setpoint jump changes are present in the closed control loop after pretuning or fine tuning. For example, you can reset the controller gain GAIN to 80% of the original value and the integral action time TI to 150% of the original setting.

Requirements

- The instruction and the technology object are loaded on the CPU.
- TUN_ES is interconnected with a step controller.
- MAN_ON = FALSE
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To manually determine the optimal PID parameters, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Manual" from the "Mode" list.
3. Choose a controller structure.

4. Enter new PID parameters in the "P", "I" and "D" fields.
5. Click the "Send parameters to the controller" icon in the "Tuning" group.
6. Select the "Change setpoint" check box in the "Current values" group.
7. Enter a new setpoint and click the icon in the "Current values" group.
8. Clear the "Manual mode" check box, if necessary.
   The controller works with the new PID parameters and controls to the new setpoint.
9. Check the quality of the PID parameters based on the trend characteristics.
10. Repeat steps 3 to 9 until you are satisfied with the controller results.

3.5.4 Basics on controller modules (S7-300, S7-400)

3.5.4.1 Inputs (S7-300, S7-400)

Introduction (S7-300, S7-400)

You can connect different types of sensors to the analog inputs. The sensor input signals are then processed to suit your requirements.

The digital inputs can be used to set different operating modes for the module.

The structure of analog and digital inputs is identical on continuous and step controllers.

Number of inputs

Input	FM 355	FM 355-2	FM 455
Analog inputs for analog value processing	4	4	16
Reference junction input for the compensation of thermocouples	1	1	1
Digital inputs	8	8	16

Analog inputs (S7-300, S7-400)

Function blocks of an analog input

The following figure shows how analog values are prepared:

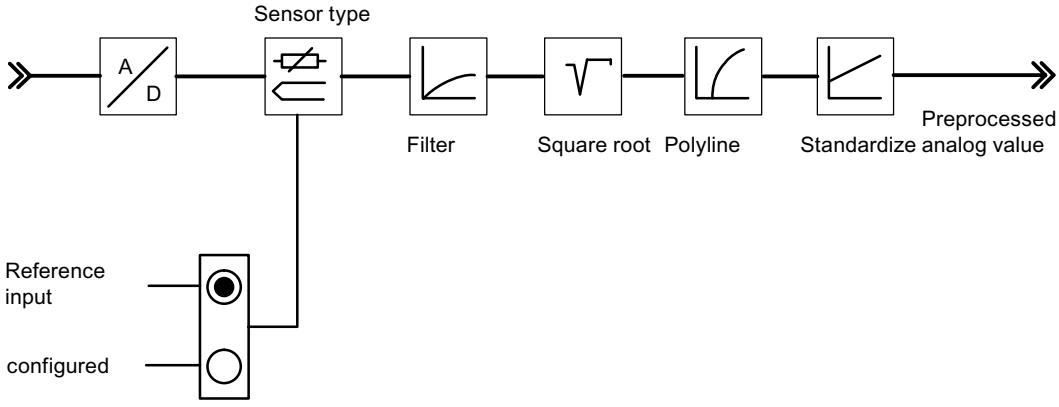

Adaptation to sensors

You can set up the analog input parameters to suit different sensors. The following settings are available:

- Analog input is not processed (e.g., unused input)
  With this setting, all PID parameters are set to zero, the controller channel is not processed, and a manipulated variable is not calculated.
- Current sensor, 0 mA to 20 mA
- Current sensor, 4 mA to 20 mA
- Voltage sensor, 0 V to 10 V
- Pt 100, –200 °C to +850 °C
- Pt 100, –200 °C to +556 ºC (double resolution)
- Pt 100, –200 °C to +130 ºC (quadruple resolution)
- Thermocouples type B, J, K, R, and S (analog input set to ±80 mV)
- Free thermocouple (analog input set to ±80 mV)

You assign the parameters for analog inputs in the "Analog input" screen form.

Toggling between Celsius / Fahrenheit

Temperatures can be measured in °C or °F units.

Adaptation to the line frequency

Input signal processing can be adapted to the line frequency in order to suppress interference during the measurement of analog signals. The following settings are available:

- 50 Hz mode
- 60 Hz mode

Reference junction

If you set up a thermocouple as sensor on an analog input, you can connect a Pt 100 to the reference junction input of the module in order to compensate for the reference junction temperature at the thermocouple elements. You can also configure a static reference junction temperature.

Alternatively, with the FM 355, you can assign parameters for an internal compensation for thermocouples J, K and E. An internal temperature sensor measures the reference junction temperature directly in the module.

When using the reference junction input, the sampling time of each controller extends by the conversion time for the reference junction input.

Analog value processing

The analog value processing offers various assignable options for preparing input signals. The following table offers an overview of these parameters and their assignable values.

Parameter	Assignable values	Note
Resolution	• 12 bits • 14 bits	Conversion time 20 ms (50 Hz) Conversion time $16^2/_3$ ms (60 Hz) Conversion time 100 ms
Filters	• ON / OFF • Time constant in s	Filter of the first order whose time response is defined by the time constant.
Square root	• ON / OFF	For the linearization of sensor signals, where the process value is available as physical quantity with quadratic relation with the measured process variable.
Polyline	• ON / OFF • 13 interpolation points can be selected in – mA with current input – mV with voltage input	To linearize encoder characteristics
Standardization	• High value • Low value	For conversion of the input signal to a different unit of measurement by means of linear interpolation between the start value (low) and end value (high)

Note

Standardization/polyline: The conversion of the unit mA or mV into a unit of measurement takes place either via the polyline or - if this is not enabled - via scaling. The polyline is used for linearization of a free thermocouple, or for any other linearization.

Copying parameters

Using the analog input of the FMs, you can transfer the parameters of one input to one or more other inputs. In doing so, all assigned parameters of the output channel are transferred to the destination channels.

Digital Inputs (S7-300, S7-400)

Operating modes

The digital inputs are used to change the operating mode of individual controller channels.

The direction of control action of the digital inputs can be configured. The following settings are available for the digital inputs:

- Active high
- Active low, or open circuit

You can select the following operating modes:

- Changeover to manipulated value setpoint by calling FB PID_FM
- Changeover to following mode (manipulated value setpoint via analog input)
- Changeover to safety manipulated value

For the step controller, you can also define the following signals via digital inputs:

- Feedback: Actuating device at high limit stop
- Feedback: Actuating device at low limit stop

Parallel tuning (S7-300, S7-400)

The "Parallel tuning" function is only available for FM 355-2.

Adjacent zones (heavy thermal coupling)

If using two or several temperature controls of a module to control the temperature, for example, of a plate (e.g. two heaters and two measured process values with heavy thermal coupling), you can tune these in parallel. Proceed as follows:

1. Define the channel group in zone A or zone B.
2. Start tuning in one of these channels. Tuning of the other channels will then be started automatically.

Note

However, for excitation by means of setpoint step, you have to ensure that the setpoint steps are set simultaneously. When using the wizard, you can specify the PID_ON and TUN_DLMN/ TUN_CLMN parameters only for one channel. Specify the parameters for the other channels by means of the corresponding instance DBs of FB FMT_PID before you start tuning.

Cancelling tuning

Tuning of all channels is cancelled if an error occurs in one of the channels (phase 0, or end of phase 1).

The user can reset tuning by setting TUN_ON=FALSE at any one of the participating channels.

When a channel goes into safety mode, tuning of all other participating channels is also disabled.

The wizard displays only the status information of the selected channel. For all other channels, the causes of error are found in the corresponding instance DBs.

Advantage

Each participating controller will output its tuning manipulated variable until all controllers have exited phase 2. This prevents the controller that completes tuning first from corrupting the tuning results of the other controllers due to the change of its manipulated variable.

NOTICE
When 75% of the setpoint step is reached at one of the participating channels, tuning is not terminated (risk of overshoot). Automatic mode is not started unless all participating controllers have completed tuning.

Adjacent zones (weak thermal coupling)

As a general rule, tuning should be carried out to reflect the way in which the control will later work. When zones are operated in parallel during production so that temperature differences between the zones remain the same, the temperature level of the adjacent zones should also be raised accordingly during tuning.

Temperature differences at the start of testing are irrelevant since they will be compensated for in the initial heating process (initial rise = 0).

3.5.4.2 Error signal (S7-300, S7-400)

Copying channels (S7-300, S7-400)

The FMs support the transfer of a controller channel to one or several other channels. All output channel settings are transferred to the target channels.

Controller types (S7-300, S7-400)

The error signal is derived from the same structure in all controller modules.

1. The signal for the actual value is conditioned to the effective actual value.
2. The setpoint signal is conditioned to the effective setpoint.
3. The effective setpoint is deducted from the effective actual value.
4. The error signal is transferred to the control algorithm.

You can select the signal sources for the setpoint and actual value and use the controller modules as:

- Fixed setpoint and cascade controllers
- Three-component controllers
- Ratio/blending controllers

Fixed setpoint and cascade controllers

The setpoint value of a master controller is set as the manipulated value of a slave controller in a cascade control.

Three-component controllers

Three-component controllers have three actual value inputs. For a blending control, the actual values A, B, and C are added to produce the total quantity PV.

- Blending controls
  A blending control is always slave controller. The associated master controller is a three-component controller.
  The manipulated variable of the master controller is interconnectd via the actual value D input. The blending factor is defined by the value at the setpoint input of the controller.

Ratio controllers

A ratio controller is always a slave controller. The associated master controller is a fixed-point controller.

The actual value of the master controller is selected as actual value D. The ratio factor is defined by the value at the setpoint input. If a controller output is activated as ratio factor, the setpoint value is converted from "0 .. 100%" to the value range "low threshold ... high threshold" using the high and low thresholds (normalized).

Blending controls

A blending control is always slave controller. The associated master controller is a three-component controller.

The manipulated variable of the master controller is activated as process value D. The blending factor is defined by the value at the setpoint input of the controller.

The manipulated variable of the total quantity controller is set to a range of values from 0% and 100%. The slave controller converts this variable to the range of actual value A at actual value input D. The range of actual value A is defined by the "high" and "low" normalization values of the selected analog input.

When the manipulated variable of a slave controller has reached the limiting threshold, or the setpoint gradient of a slave controller is limited by the ramp function in the setpoint value branch, the integral component of the master controller is blocked depending on the direction (anti-reset windup) until the cause of limiting has been eliminated in the slave controller.

Signal selection for setpoint value, actual value, D-action input and disturbance variable (S7-300, S7-400)

Signal selection

You can activate different signal sources for the setpoint, actual value, value at the D input (derivative action input), and disturbance variable at each controller channel. The following table provides an overview of the signal selection options.

Table 3-2 Signal selection for setpoints, actual values, D input, and disturbance variables.

Affected values	Signal sources available
Setpoint	• A value defined by the function block in the user program • The conditioned analog value of an analog input • The manipulated value (LMN, LMN_A or LMN_B) of another controller channel (when cascading controllers)
Actual values A, B, and C	• Zero • The conditioned analog value of an analog input (the actual values B and C can also be evaluated by factors)
Actual value D	• Zero • The conditioned analog value of an analog input • Setpoint of another controller channel
Value for D input (only relevant for PD or PID controllers)	• The error signal following the deadband of the internal controller channel • The conditioned analog value of an analog input • The negated effective actual value of the controller channel
Disturbance variable	• Zero • The conditioned analog value of an analog input

Preparation setpoint (S7-300, S7-400)

The conditioning of a setpoint to the effective process value can be influenced by the following functions:

Setpoint source

The setpoint can be defined by:

- an instruction
- an analog input
- the conditioned output value of another controller channel in the module
  This source for the output value is selected for a blending controller, or slave controller in a cascade control.
- the conditioned output value A of another controller channel in the module (FM 355-2 only)
- the conditioned output value B of another controller channel in the module (FM 355-2 only)

Safety setpoint

When the CPU fails or with a CPU stop, the module can set a safety setpoint.

Enter a safety setpoint and set "Setpoint = Safety setpoint" as behavior to CPU stop.

Reaction to CPU failure

The controller module can set the following setpoints if the CPU fails or with a CPU stop:

- Setpoint = last valid setpoint
  If the "from instruction" is set as the setpoint selection, the setpoint will remain constant at the last value set after CPU failure or CPU stop.
  If the setpoint is defined by the output value of an FM controller, or by an analog input, the setpoint will change in line with the activated value.
- Setpoint = safety setpoint
  If you entered a safety setpoint, the module corrects the setpoint to this value.

Reaction to module start (FM 355 and FM 455 only)

The controller module can correct the values to the following setpoints during the start of module:

- Setpoint = last valid setpoint
- Setpoint = safety setpoint

Ramp

You can limit the setpoint rate of change by selecting a ramp up time that covers the physical start to the end value.

Setpoint limits

If the setpoint is defined by the function block, or by a conditioned analog value of an analog input, the setpoint is limited to configurable high and low limits.

If the conditioned output value of another controller channel has been activated as setpoint for the ratio controller, this value is used as multiplication factor for process value D. The setpoint, given as a percentage at the input, can in this case be scaled using the high and low limits.

If the output value of another controller channel is used as setpoint for a fixed setpoint or cascade controller, this value can be scaled to a physical value with the help of the normalization constants of the process value channel selected.

Multiplication

Process value A is used as controlled variable, while process value D is used as ratio variable for the "ratio controller" type. The setpoint input acts as the ratio factor. It is conditioned to form an effective setpoint by multiplication with process value D and adding an adjustable offset. If process value D is disabled, only the offset is added to the setpoint.

Preparation actual value (S7-300, S7-400)

Actual value conditioning to form the effective actual value depends on the controller structure.

Effective actual value for fixed setpoint and cascade controllers

The effective actual value is equivalent to actual value A that can be derived from an analog input source. Actual values B, C, and D are ignored.

Effective actual value for three-component controllers

The effective actual value is calculated by adding the actual values A, B, and C plus an adjustable offset. The actual values B and C can also be evaluated by factors. Actual value D is ignored.

Effective actual value in ratio controllers

Select the following actual values for the ratio controller

- Actual value A: The conditioned signal of an analog input
- Actual value D: Actual value of the master controller.

The ratio factor is defined as setpoint.

If a controller output has been activated as ratio factor FAC, the setpoint is converted from "0 ... 100%" to the value range "low threshold ... high threshold" using the low and high thresholds (scaled).

Effective actual value in blending controllers

Select the following actual values for the blending controller

- Actual values A, B, and C The conditioned signals of the associated analog inputs.
- Actual value D: Actual value of the master controller.

The ratio factors of the individual controlled systems are defined as setpoint.

The manipulated variable of the total quantity controller is set in the range of values from "0% to 100%". The slave controller converts this variable at input D to the range of values between "low threshold ... high threshold" of actual value A (scaled).

The manipulated variable of the master controller is interconnected via "Actual value D" input of the slave controller. Blending factor FAC is defined by the value set at the controller's setpoint input.

Interrupt (S7-300, S7-400)

Limit monitoring

The controller module contains a limit monitoring function. This enables monitoring of the high and low limits

- of the control deviation
- of the effective process value

for high and low limits warnings and alarms. You can set up a hysteresis for these limits.

The following figure shows the hysteresis for warning and alarm limit:

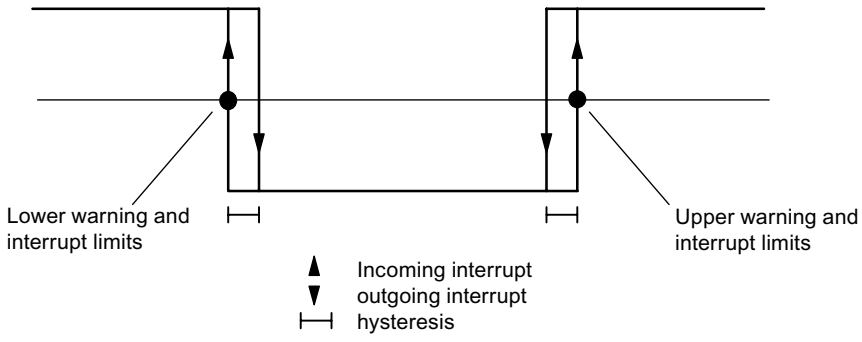

3.5.4.3 Control algorithm (S7-300, S7-400)

FM 355-2 control algorithm (S7-300, S7-400)

Components of the control algorithm

Continuous controllers (FM 355-2 C) and pulse controllers (FM 355-2 S) have the same control algorithm structure. The cooling and control zone buttons cannot be selected for the step controller (FM 355-2 S).

The following figure shows the block diagram of the control algorithm for continuous and pulse controllers.

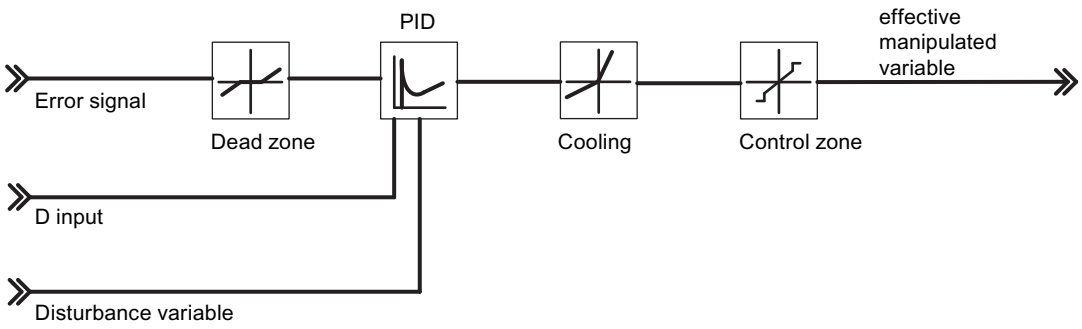

Control algorithm: PID in parallel structure

During the cycle of the configured sampling time, the controller's manipulated variable is calculated based on the control deviation of the PID position algorithm. The controller is implemented in a purely parallel structure. The proportional, integral and derivative actions can each be disabled separately. To disable the integral and derivative actions, set the respective parameter TI or TD to zero.

The following figure shows the control algorithm of the FM 355-2 (parallel structure):

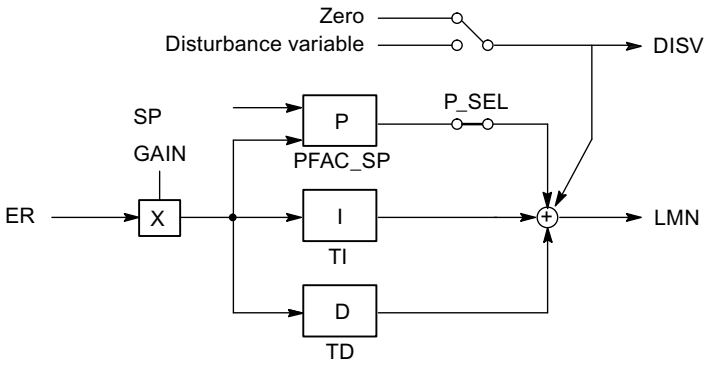

FM 355 and FM 455 control algorithm (S7-300, S7-400)

Components of the control algorithm

You can choose between the following operating modes of the control algorithm:

- Temperature controller (self-tuning fuzzy controller)
- PID controller

C and S controllers have the same control algorithm structure. The following figure shows the block diagram of the control algorithm:

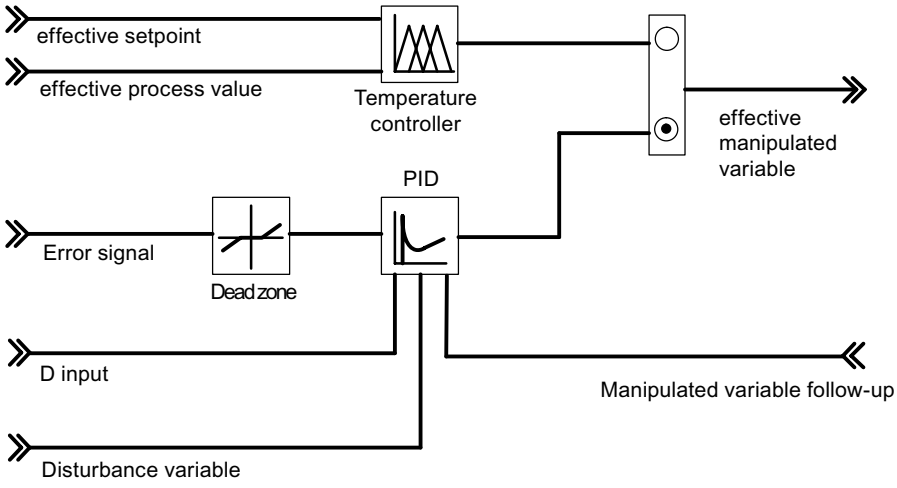

Control algorithm and controller structure

The manipulated variable of the continuous controller is calculated within the configured sampling time on the basis of the control deviation in the PID positioning algorithm. The controller is designed with a purely parallel structure (see figure below). The proportional, integral and derivative actions can each be disabled separately. To disable the integral and derivative actions, set the respective parameter TI or TD to zero.

The following figure shows the control algorithm of the FM 355 (parallel structure):

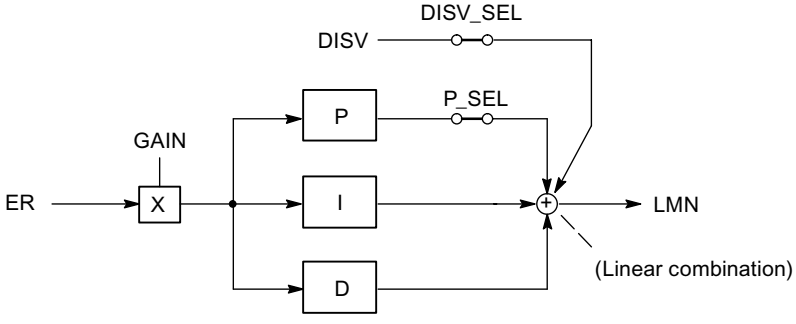

Selecting the operating mode (S7-300, S7-400)

Selecting the operating mode

You can choose between different operating modes if the "PID controller" entry has been set as controller type:

- P, for implementing a proportional controller
- PI, for implementing a PI controller
- PD, for implementing a PD controller
- PID, for implementing a PID controller
- I, for implementing an integrating controller
- ID, for implementing an ID controller
- D, for implementing a D controller

Bumpless change from manual to automatic mode

If you selected “Bumpless change from manual to automatic mode” (not available for step controllers), the integrator is tracked in manual mode so that the manipulated variable does not take a step through the proportional and derivative actions during the manual to automatic changeover. An error signal is corrected only slowly by means of the integral action. If bumpless manual to automatic changeover is not selected, the manipulated variable performs a step that corresponds with the current error signal during the change from manual to automatic mode. This function allows for the fast correction of active control error.

Note

A step controller is always subject to pulse action at the changeover from manual to automatic mode. The existing error signal and GAIN leads to a jump in the internal manipulated variable. The integral effect of the actuator, however, results in a ramp-shaped excitation of the process.

Disturbance variable compensation

An additional disturbance variable DISV can be added to the controller’s output signal. This is activated and deactivated in the “System deviation” window of the configuration tool using the switch “Signal selection, controller disturbance variable”.

Dead zone (S7-300, S7-400)

Purpose of the deadband

A deadband is interconnected upstream of the PID controller. In settled controller state, the deadband suppresses the noise component in the error signal that could be generated by superimposition of a disturbance signal of higher frequency on the control or reference variable. This function prevents unwanted oscillation of the controller output.

Deadband width

The deadband width can be configured. If the error signal is within the set dead band width, the value 0 (error signal = 0) is returned at the deadband output. The output value does not follow the input variable until the input variable moves outside the sensitivity range.

This results in corruption of the transferred signal even outside the deadband. However, this is accepted in order to avoid jumps at the limits of the deadband. The corruption corresponds to the value of the deadband width and can therefore be controlled easily.

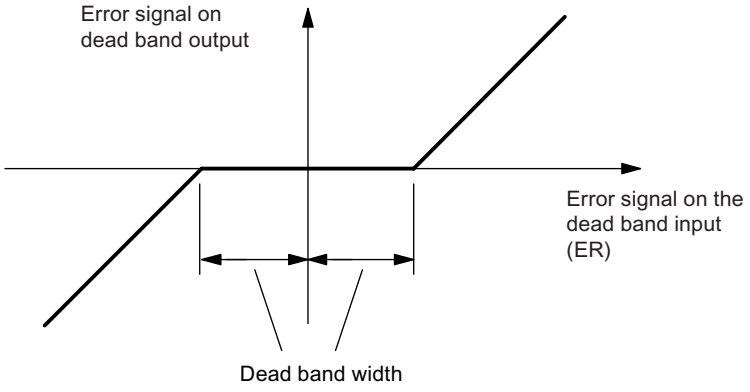

Figure 3-8 Deadband

Proportional control (S7-300, S7-400)

Proportional controller

The following figure shows the step response of a proportional controller:

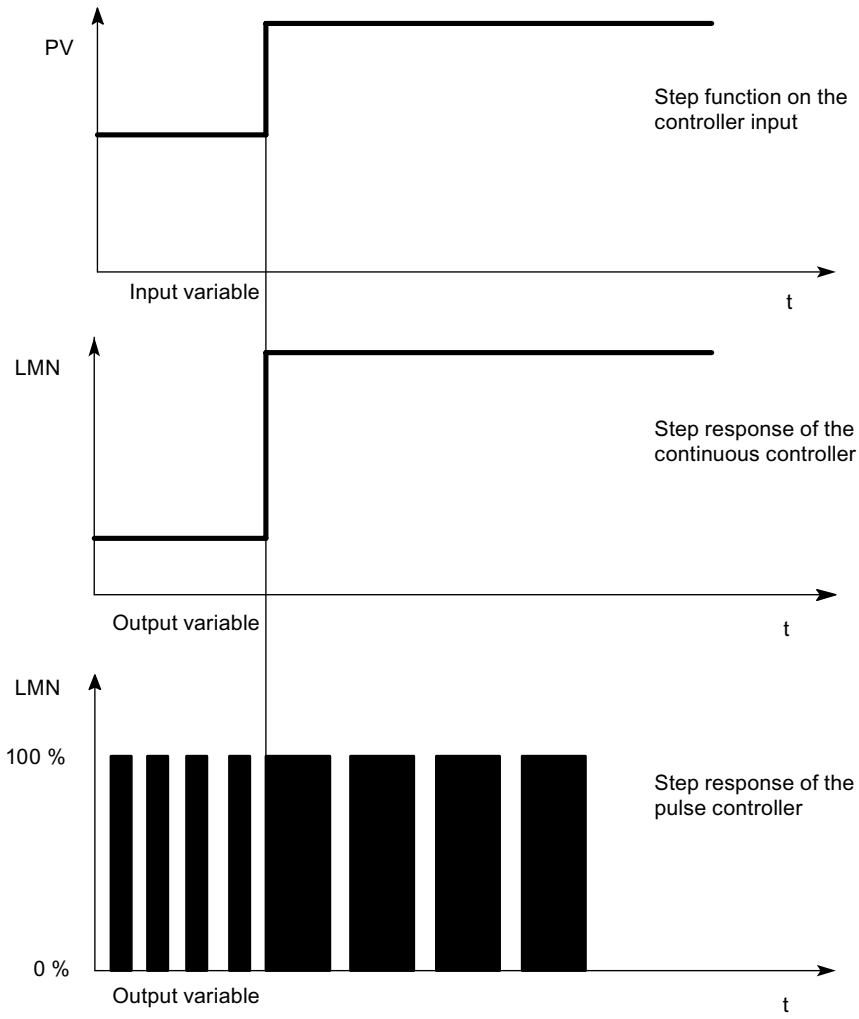

Formula for proportional controllers

The output and input variables are directly proportional, i.e.:

Change to the output variable = proportional coefficient x change to input variable

LMN = GAIN x ER

Proportional control

The integral and derivative actions are disabled in a proportional controller. This means that the manipulated variable is also "0" when control deviation ER = 0. If an operating point ≠0, in other words, a numeric value is to be set for the manipulated variable when control deviation = 0, the following is possible via the operating point:

- Automatic operating point: The controller automatically sets the operating point to the value of the current (manual) manipulated variable when you switch from manual to automatic mode.
- Operating point not automatic: You can assign the operating point parameters.
- Example: Operating point AP = 5% results in a manipulated variable of 5% when control deviation ER = 0.

The following figure shows a proportional controller with operating point setting by means of an integral element:

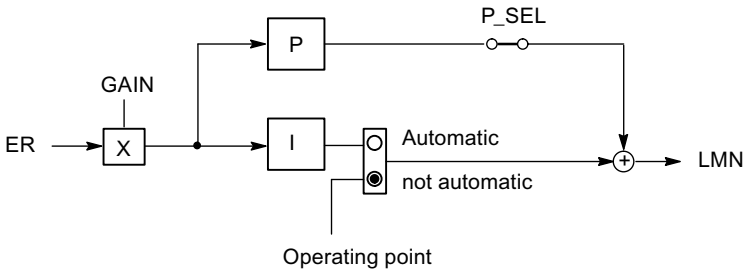

The following figure shows the step response of the proportional controller:

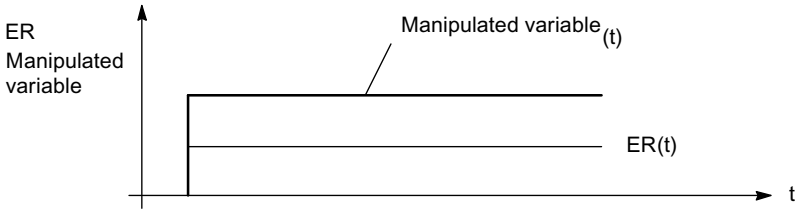

PI control (S7-300, S7-400)

PI controller

The following figure shows the step response of a PI controller:

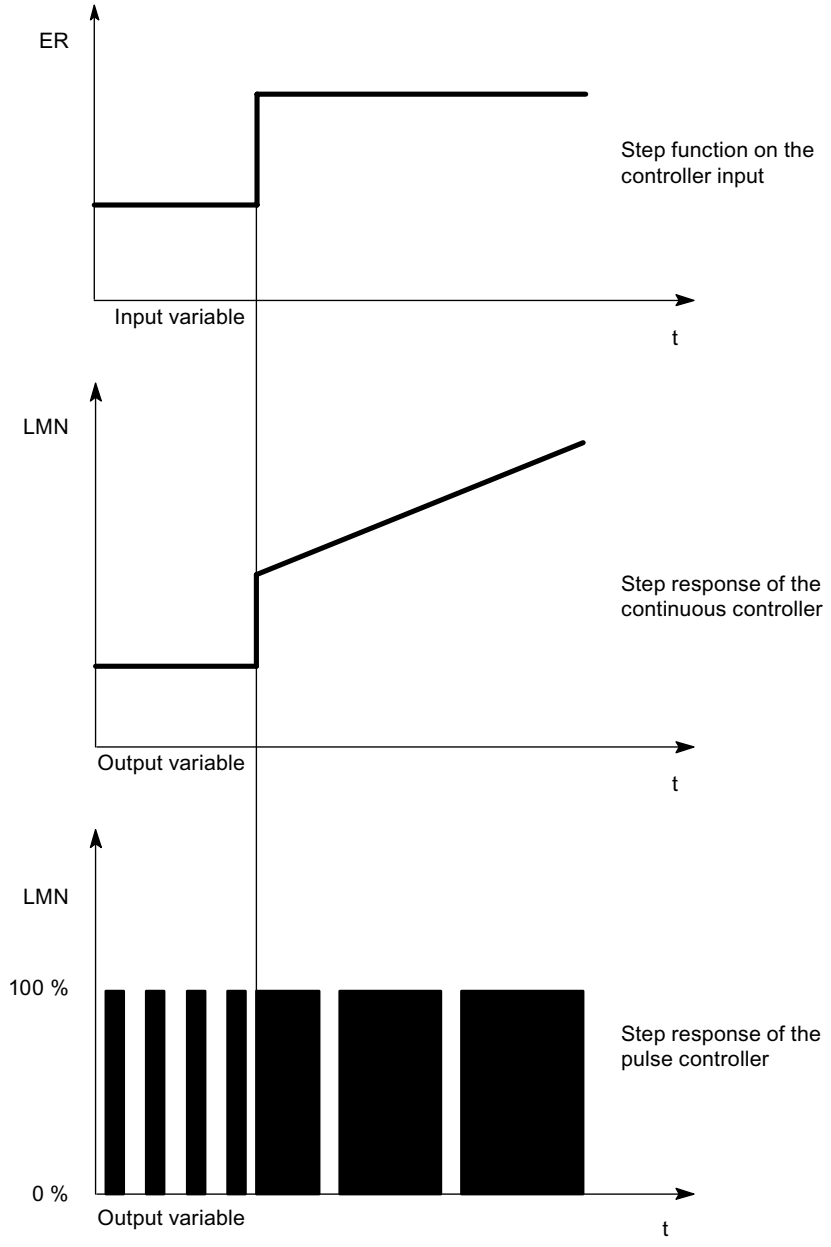

Integral control elements have the integral of the input variable as the output variable, i.e. the controller adds up the deviation from the setpoint value over time. This means that the controller continues correction until the deviation from the setpoint is eliminated. In practice, depending on the demands on the controlling, a combination of the different timing elements is ideal. The

timing of the individual elements can be described by the controller parameters Proportional band GAIN, Reset time TI (integral action) and Rate time TD (derivative action) .

Formula for PI controllers

The following applies for the step response of the PI controller in the time range:

$$LMN = GAIN \times ER \times \left(1 + \frac{t}{TI}\right)$$

t = time interval since the input variable step

PI control

The derivative action is disabled in the PI controller (TD=0.0). A PI controller corrects the output variable by means of integration until control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the output range. The integral action retains the value reached at the output value limit if these limits are exceeded (anti-reset windup).

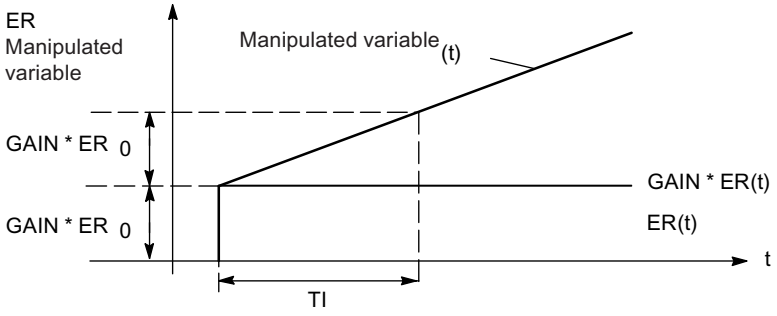

PD control (S7-300, S7-400)

PD controller

The following figure shows the step response of a PD controller:

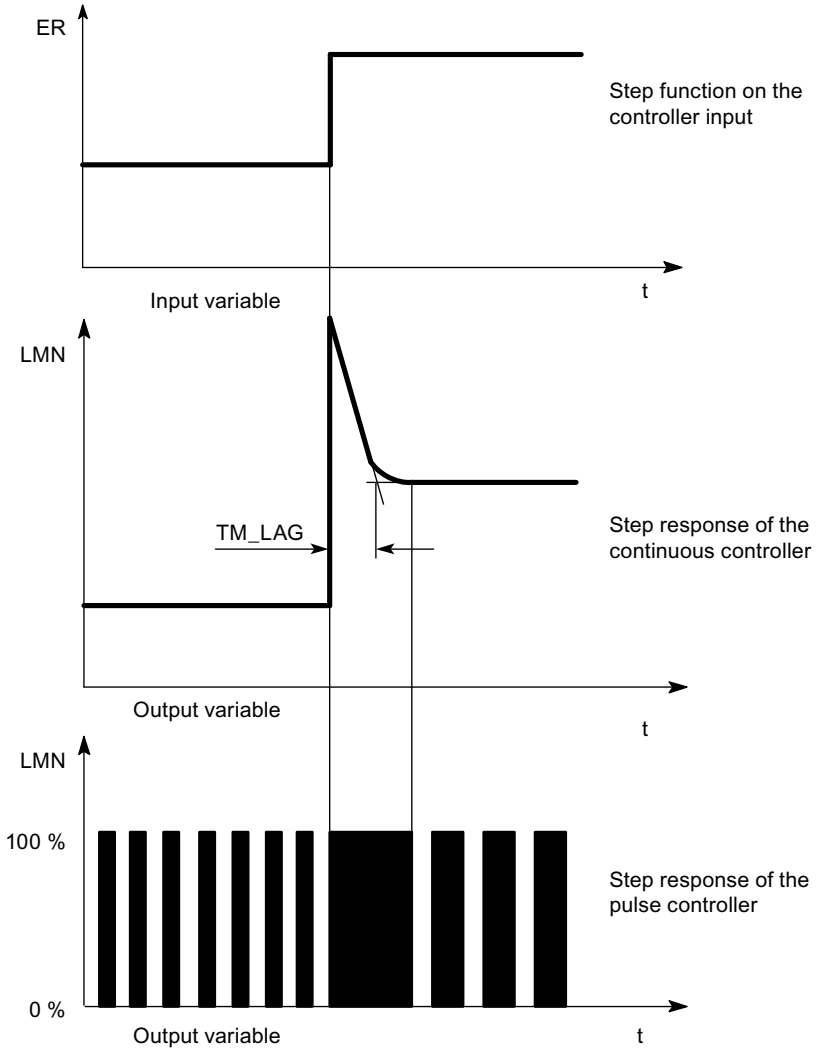

The derivative control elements are the only ones unsuitable for controlling, because they no longer output actuating commands when the input variable has settled to a static value.

In combination with proportional control elements, the derivative action is used to generate a control pulse based on the rate of change at the controlled variable. If a disturbance variable z is affecting the controlled system, the PD controller sets a different control deviation as a result of the changed degree of correction. Disturbance factors are not fully corrected. Of advantage is the good dynamic response. A well attenuated, non-oscillating transfer is achieved during startup and when changing the reference input variable. However, a controller with derivative action is not appropriate if a control system has oscillating measured variables, e.g. in pressure or flow control systems.

Formula for PD controllers

The following applies for the step response of the PD controller in the time range:

$$LMN = GAIN \times ER \times \left(1 + \frac{TD}{TM_LAG} \times e^{\frac{-t}{TM_LAG}}\right)$$

t = time interval since the input variable step

PD control

The integral action is disabled in the PD controller (TI=0.0). This means also that output signal = 0 when control deviation ER = 0. If an operating point ≠0, in other words, a numeric value is to be set for the manipulated variable when control deviation = 0, the following is possible via the operating point:

- Automatic operating point: The controller automatically sets the operating point to the value of the current (manual) manipulated variable when you switch from manual to automatic mode.
- Operating point not automatic: You can assign the operating point parameters.

The PD controller maps input variable ER(t) proportionally to the output signal and adds the derivative action formed by the differentiation of ER(t); the derivative action is calculated with double precision according to the trapezoid rule (Padé approximation). The time response (strength of the derivative action or actuating area) is determined by the derivative action time TD (rate time).

For signal smoothing and interference suppression, the derivative action is implemented with a time lag of the first order (adjustable time constant: TM_LAG). A small value for TM_LAG is usually sufficient to achieve the desired result.

The higher the derivative factor D_F,

- the smaller the effective time constant TD/D_F of the delay,
- the higher the maximum initial manipulated variable,
- and the better the control action;
- the higher the noise sensitivity.

D_F is limited to the value range 5.0 through 10.0.

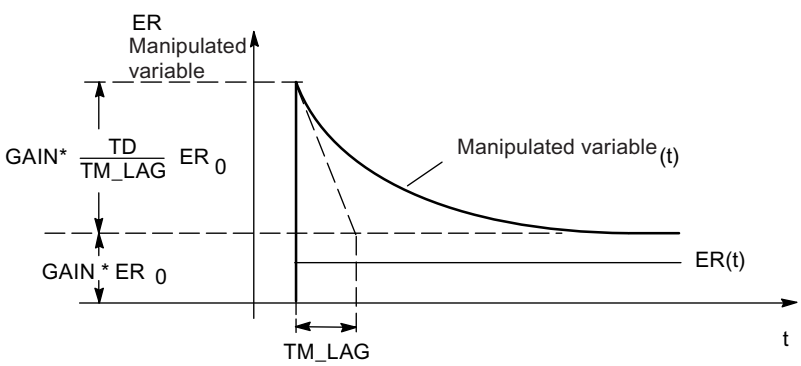

PID control (S7-300, S7-400)

PID controller

The following figure shows the step response of a PID controller:

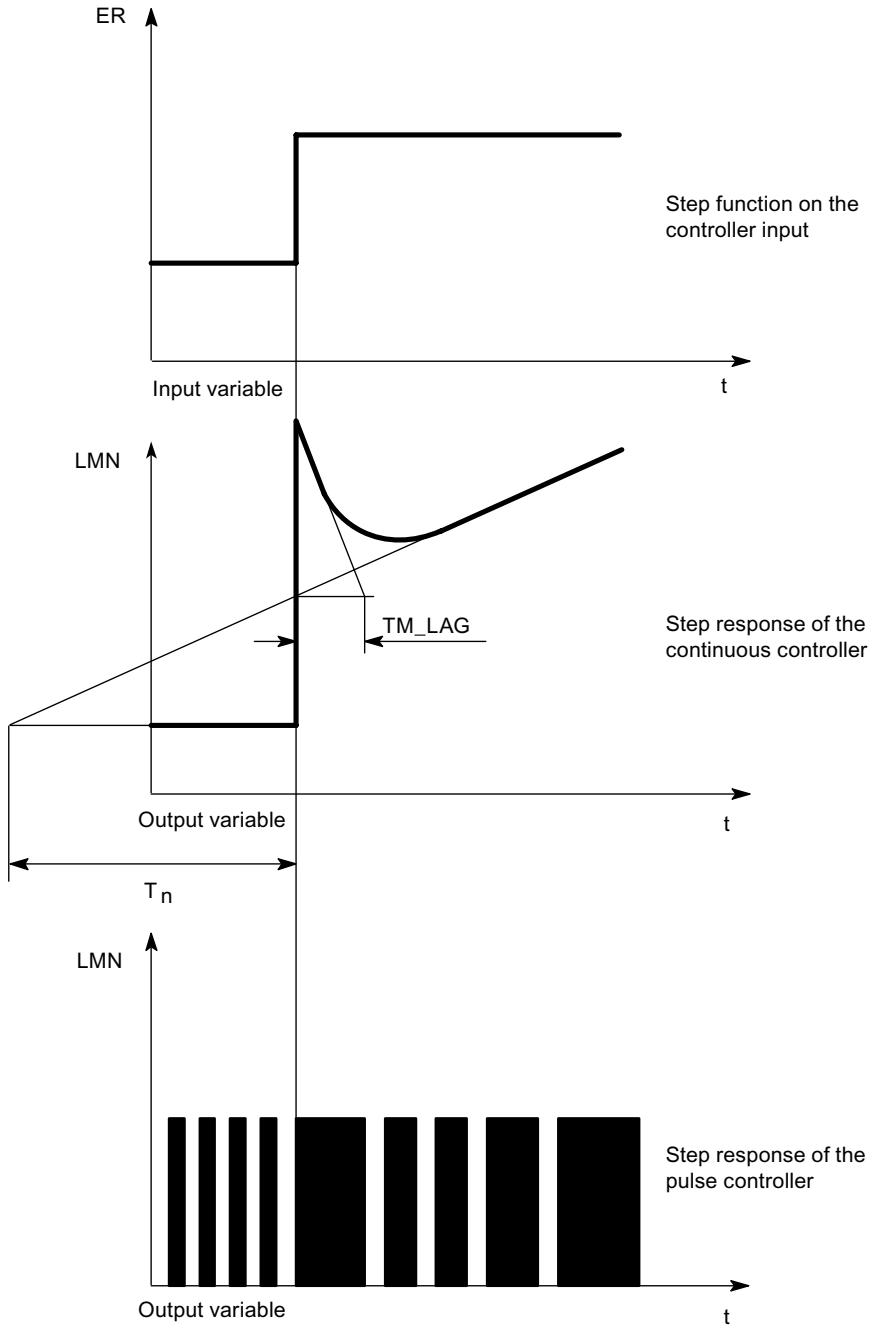

Most control systems in process engineering can be managed with a controller with PI component. In the case of slow control systems with long dead times, for example temperature control systems, the control result can be improved with a controller with PID action.

The following figure shows the step response for various proportional controllers:

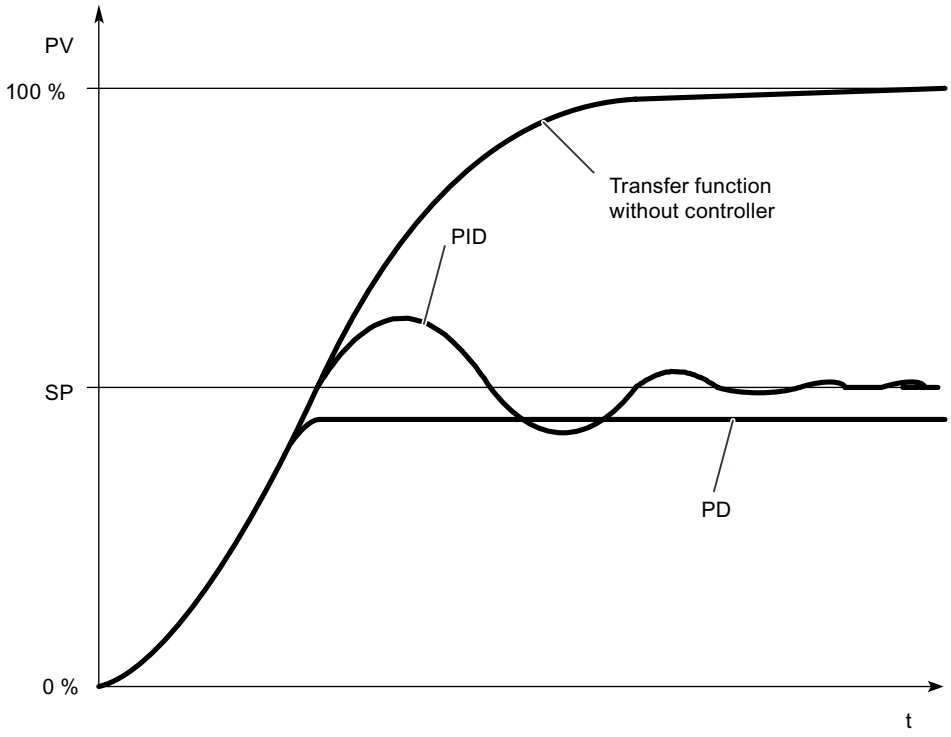

Controllers with PI and PID action have the advantage that the process variable does not deviate from the setpoint value after settling. The controlled variable oscillates above the setpoint value upon start-up.

Formula for PID controllers

The following applies for the step response of the PI controller in the time range:

$$LMN = GAIN \times ER \times \left(1 + \frac{t}{TI} + \frac{TD}{TM_LAG} \times e^{\frac{-t}{TM_LAG}}\right)$$

t = time interval since the input variable step

PID control

The P, I and D actions are switched on in PID controllers. A PID controller adjusts the output variable via the integral action until the control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the output range. The integrator maintains the value it has at the point where the limits of the output value are exceeded (Anti-Reset-Windup).

The PID controller proportionally maps the input variable ER(t) to the output signal and then adds the actions generated by the differentiation and integration of ER(t). The time action is determined by the derivative action time TD (rate time) and the integration time TI (reset time).

The derivative action is realized with a delay circuit of the first order for signal smoothing and interference suppression.

The higher the derivative factor D_F,

- the smaller the effective time constant TD/D_F of the delay,
- the higher the maximum initial manipulated variable,
- and the better the control action;
- the higher the noise sensitivity.

D_F is limited to the value range 5.0 through 10.0.

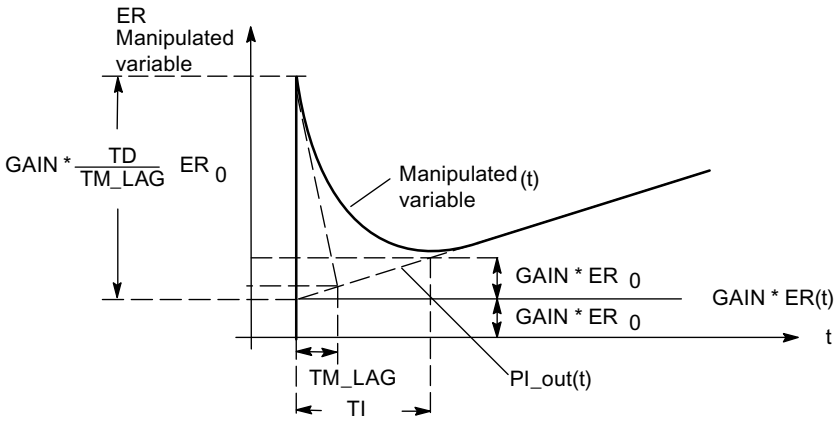

I control (S7-300, S7-400)

Integral controller

You can deactivate the proportional action in order to realize a pure integral-action control. This can also be done using the P_SEL parameter of the PID_FM function block.

Proportional action (S7-300, S7-400)

Attenuation of the proportional action in the event of setpoint changes

You can avoid process value overshoot or excessive manipulated variable amplitude with attenuation of the proportional action using the parameter “Proportional factor at setpoint change” (PFAC_SP). Using PFAC_SP, you can select continuously between 0.0 and 1.0 to decide the effect of the proportional action when the setpoint changes:

- PFAC_SP=1.0: Proportional action for setpoint change is fully effective
- PFAC_SP=0.0: Proportional action for setpoint change is not effective

There are two ways of limiting the rate of setpoint change in the FM355-2:

- Activate ramp (> 0.0s)
- Setpoint change factor < 1.0

Use only one of the two limits. If both limits are activated at the same time, a setpoint jump will produce a change in the manipulated variable in the opposite direction to the setpoint change (step response).

Specific features of the step controller

A value of PFAC_SP < 1.0 can reduce overshoot if the motor transition time MTR_TM is small compared to the recovery time TA and if the ratio TU/TA < 0.2. Should MTR_TM reach 20 % of TA, only a slight improvement can be achieved.

Inversion of the controller effect

You can enable controller inversion, that is, conversion from

- increasing control deviation = rising manipulated variable
  to
- rising control deviation = falling manipulated variable

by setting a negative proportional gain (GAIN). The sign of this parameter value determines the effective direction of the controller.

Derivative action (S7-300, S7-400)

Derivative action in feedback path

When the setpoint changes, you can avoid pulse-shaped peaks of the derivative action of the manipulated variable by moving the derivative action into the feedback path.

In this structure only the negative control variable (Factor = -1) is fed forward to the derivative action. Changeover for the derivative action is carried out in the feedback in the “Control

deviation” window via the “Derivative input controller” switch by selecting the negated effective process value as the input signal.

Note

If you move the derivative action to the feedback path, you should also reduce the value of PFAC_SP, otherwise you would increase overshoot of the process value.

The following figure shows the control algorithm with derivative action in the feedback branch:

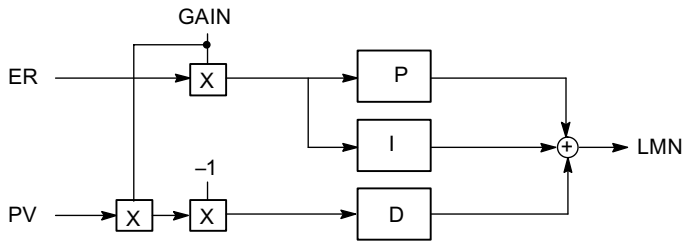

Proportional/Derivative action (S7-300, S7-400)

P/D action in the feedback

In the parallel structure, each action in the control algorithm receives the control deviation as input signal. In this structure, setpoint steps act directly on the controller. The manipulated variable is influenced directly by the proportional and derivative actions by setpoint steps.

However, a different controller structure, in which the formation of the proportional and derivative action is moved to the feedback, ensures a smooth response of the manipulated variable to setpoint step changes (see figure below).

In this structure, the integral action processes the control deviation as the input signal. Only the **negative** controlled variable (factor = - 1) is applied to the proportional and derivative actions. Changeover for the derivative action is carried out in the feedback in the “Control deviation” window via the “Derivative input controller” switch by selecting the negated effective process value as the input signal.

The following figure shows the control algorithm with proportional and derivative action in the feedback branch:

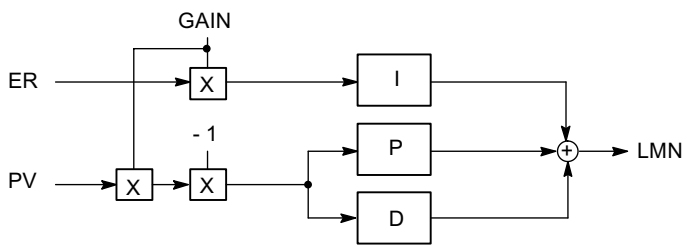

Fuzzy temperature controller module (S7-300, S7-400)

Temperature controller

The temperature controller is a self-regulating fuzzy controller, which, after identifying the control section, works using control parameters it has calculated itself.

The following temperature controller settings are possible:

- Cooling controller
- Heating controller
- Aggressivity
  The aggressivity parameter can be used to influence the speed of the transient response.

Possible aggressivity values:	
-1 ≤ Aggressivity < 0 :	Slower transient response than determined by identification
Aggressivity = 0	Transient response as determined via identification
0 < Aggressivity ≤ 1	Faster transient response than determined by identification

3.5.4.4 Controller output (S7-300, S7-400)

Differences between controllers (S7-300, S7-400)

The FM 355, FM 355-2 and FM 455 controllers are basically divided into two types: step controllers and continuous controllers. Step controllers are identified by an added S, such as FM 355 S, FM 355-2 S and FM 455 S, whereas continuous controllers are identified with a C, such as FM 355 C, FM 355-2 C and FM 455 C. The way the controller modules work do not differ, an FM 355 S and an FM 355 have identical algorithms. The differences lie in the controller output.

Step controller

The continuous step controller has digital outputs. Therefore, a control valve can only be addressed by the signals "open" or "closed". If you wish to control a control valve for 50% of the capacity, for example, this can only be realized by setting "open" and "closed" to an equal length.

You can find more detailed information on step controllers at the following locations:

- Digital controller outputs (Page 972)
- Step controller without position feedback (Page 973)
- Step controller with position feedback (Page 973)
- Pulse controller (Page 973)

Continual controller

The continuous step controller has analog outputs. A control valve can not only be "open" or "closed", it can also be opened by specific percentages. If you wish to control a control valve for 50% of the capacity, for example, the controller module outputs the value "50%" and the door opens halfway.

You can find more detailed information on continuous controllers at the following locations:

- Controller analog outputs (Page 971)
- Continuous controller (Page 972)

Controller analog outputs (S7-300, S7-400)

Analog outputs

You can assign parameters for the following functions for each analog output:

- Signal source
- Signal type

Selection of source for the signal at the analog outputs

Selection of the signal source allows you to define which signal value is to be output at the analog output in question.

The following signal sources can be assigned:

- The value zero
- The conditioned analog value of one of the analog inputs
- Manipulated value A of one of the controller channels
- Manipulated value B of one of the controller channels

The FM 355 and FM 355-2 each have four analog inputs and four controller channels.

The FM 455 has 16 analog inputs and 16 controller channels.

Signal Type at the analog outputs

You can determine the signal type for each analog output.

The following signal types can be assigned:

- Current output 0 ... 20 mA
- Current output 4 ... 20 mA
- Voltage output 0 ... 10 V
- Voltage output -10 ... 10 V

Digital controller outputs (S7-300, S7-400)

Digital controller outputs

The digital outputs of the controller are used to control integrating and non-integrating actuators.

You can assign the following functions to each digital output:

- Output type
- Signal selection
- Signal processing

Signal output type at the digital output

You can define the signal type for each digital output.

The following signal types are available:

- Step controller w/o position feedback input
- Step controller with position feedback input
- Pulse controller

Signal selection

A range of values can be assigned for signal selection depending on the signal type at the output:

- External manipulated variable (Page 974)
- Compensation (Page 974)
- Position feedback input (Page 975)
- Safety manipulated value (Page 976)

Signal processing

A range of values can be assigned for signal processing depending on the signal type at the output:

- Output signal limit (Page 977)
- Split range (Page 977)
- Pulse shaper (Page 980)

Continuous controller (S7-300, S7-400)

Signal type at the digital outputs

The following parameters can only be set for an analog module.

Available parameters

- Activateexternal manipulated variable (Page 974)
- Activatecompensation (Page 974)
- Setsafety manipulated value (Page 976)
- Limit manipulated value (Page 977)
- Enable two actuators (Page 977)
- Select output signal and signal type (Page 980)

Step controller without position feedback (S7-300, S7-400)

Signal type at the digital outputs

The following parameters can only be set for a digital module.

Available parameters

- Activateexternal manipulated variable (Page 974)
- Setsafety manipulated value (Page 976)
- Pulse shaper (Page 980)

Step controller with position feedback (S7-300, S7-400)

Signal type at the digital outputs

The following parameters can only be set for a digital module.

Available parameters

- Activateexternal manipulated variable (Page 974)
- Activatecompensation (Page 974)
- Setsafety manipulated value (Page 976)
- Limit manipulated value (Page 977)
- Setposition feedback (Page 975)
- Pulse shaper (Page 980)

Pulse controller (S7-300, S7-400)

Signal type at the digital outputs

The following parameters can only be set for a digital module.

Available parameters

- Activateexternal manipulated variable (Page 974)
- Activatecompensation (Page 974)
- Setsafety manipulated value (Page 976)
- Limit manipulated value (Page 977)
- Conditioning analog signal for conversion to a digital signal (Page 977)
- Converting analog signal to digital signal (Page 946)

Functions of the controller output (S7-300, S7-400)

External manipulated variable (S7-300, S7-400)

Overview

There are various interconnection possibilities for the manipulated value, the compensation input and the safety manipulated value at the controller output (manipulated value changeover).

A limit ensures that the manipulated value cannot assume invalid values for the process.

Switching external manipulated value

The changeover between external manipulated value and effective manipulated value from the controller can alternatively take place using

- A binary value from the function block
- A signal from a logical OR link of a digital value from a function block and a digital input.

Compensation (S7-300, S7-400)

Overview

Manipulated value compensation prevents a jump in the manipulated value during the changeover from manual to automatic mode.

The manipulated value remains unchanged during the changeover from manual to automatic mode. Manipulated value compensation is not active if a purely proportional controller with a fixed operating point has been implemented (”automatic” is not checked in the PID controller screen).

Compensation input

The following alternative settings are available:

- The value at the compensation input = zero
- The compensation input is the conditioned analog value of an analog input

Switching to compensation

You can also change from manipulated value to compensation input using

- A binary value from the function block
- A signal from a logical OR link of a digital value from a function block and a digital input.

Position feedback (S7-300, S7-400)

Overview

A position feedback input is only available for step controllers.

Position feedback input

The following alternative settings are available:

- The position feedback input value = zero
- The position feedback input is the conditioned analog value of an analog input

Controller output of the step controller

The following figure shows the principle of the S controller output (step controller operating mode with position feedback):

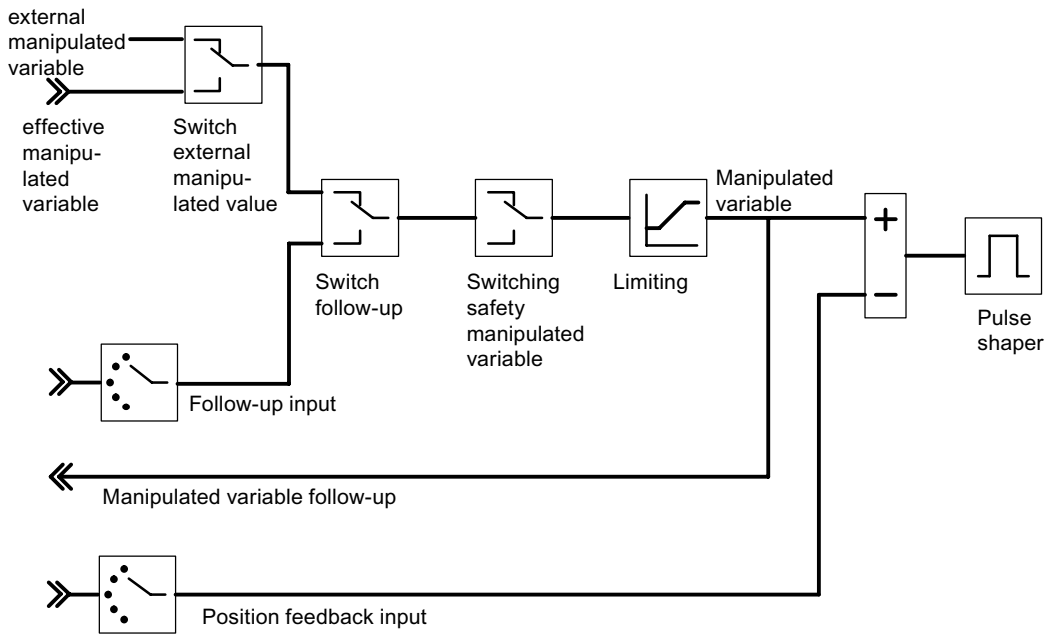

The following figure shows the principle of the S controller output (step controller operating mode without position feedback):

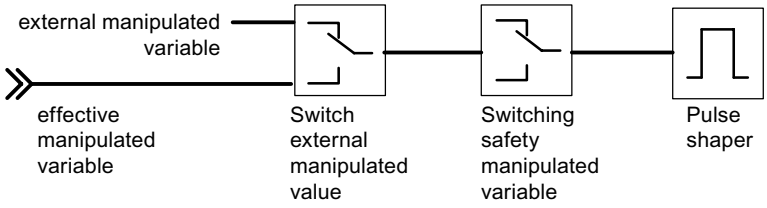

In a step controller without analog position feedback, the external output value and the safety output value act as follows:

If a value between 40.0% and 60.0% is specified, no binary output is set and the actuating device remains unchanged.

If a value > 60.0% is specified, "Actuating signal high" is output until the feedback "Actuating device at high limit" is triggered.

If a value < 40.0% is specified, "Actuating signal low" is output until the feedback "Actuating device at low limit" is triggered.

Safety manipulated value (S7-300, S7-400)

Overview

There are various interconnection possibilities for the manipulated value, the compensation input and the safety manipulated value at the controller output (manipulated value changeover).

Safety manipulated value switching

- Establishing the safety manipulated value
- Alternative reaction of the function module upon start-up:
  - The function module switches to closed-loop operation
  - The safety manipulated value is output as the manipulated value
- Changeover to the safety manipulated value is also possible with
  - A binary value from the function block
  - A signal from a logical OR link of a digital value from a function block and a digital input.

- Reaction to measuring transducer fault of actual value A:
  - The operating mode of the controller remains unchanged when "Closed-loop control operation" is set
  - It changes to the safety manipulated value when "manipulated value = safety manipulated value" is set
- Reaction to analog input measuring transducer fault:
  - The operating mode of the controller remains unchanged when "Closed-loop control operation" is set
  - It changes to the safety manipulated value when "manipulated value = safety manipulated value" is set

Manipulated value limitation (S7-300, S7-400)

Overview

A limit ensures that the manipulated value cannot assume invalid values for the process.

Manipulated value limit

- High and low limit (cannot be switched off)

Split range (S7-300, S7-400)

Overview

The split range function generates two differently standardized output signals, output value A and output value B, from the output value as input signal. This allows two valves, for example, to be controlled with one output value.

Formation of the split range output value

- In / out (C controllers only)
- Initial and final value - input signal
- Initial and final value of the output signal

Split range

The splitrange function allows two control valves to be controlled with one manipulated variable. The split range function generates the two output signals output value A and output value B from the output value LMN as input signal.

The following figure shows the effect of the parameters for the output of output value A:

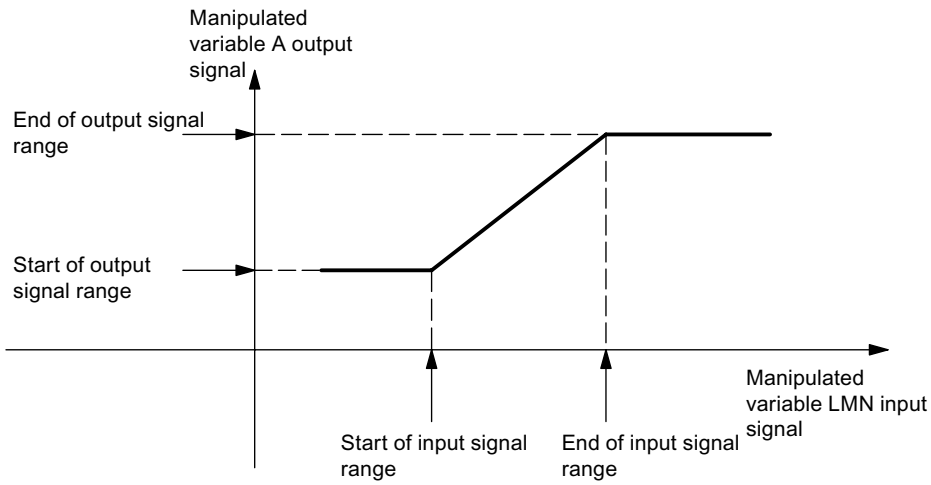

The following figure shows the effect of the parameters for the output of output value B:

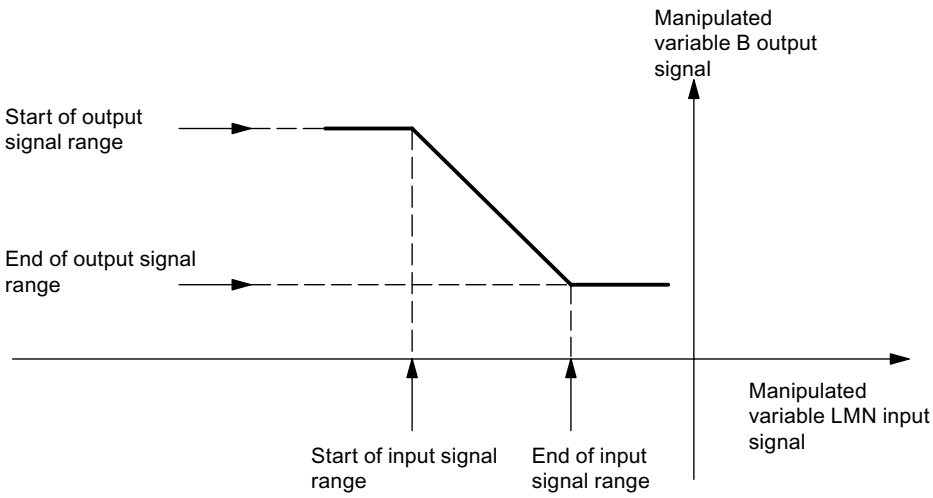

The start of the input signal area must be less than the end of the input signal area.

Split range / pulse shaper

The split range function is the preparation of the analog signal for conversion to a binary signal.

In the case of a **two-step controller** (e. g. a heating controller), only the output value A is of relevance. The conversion of the output value to the output value A is shown in the following figure. Conversion to a binary output signal is carried out so that the ratio of pulse length to period duration corresponds to the output value A at the assigned digital output.

For example, a output value A of 40% and a period of 60 seconds result in a pulse length of 24 seconds and a pause duration of 36 seconds.

The figure below shows the split range function on a two-step controller:

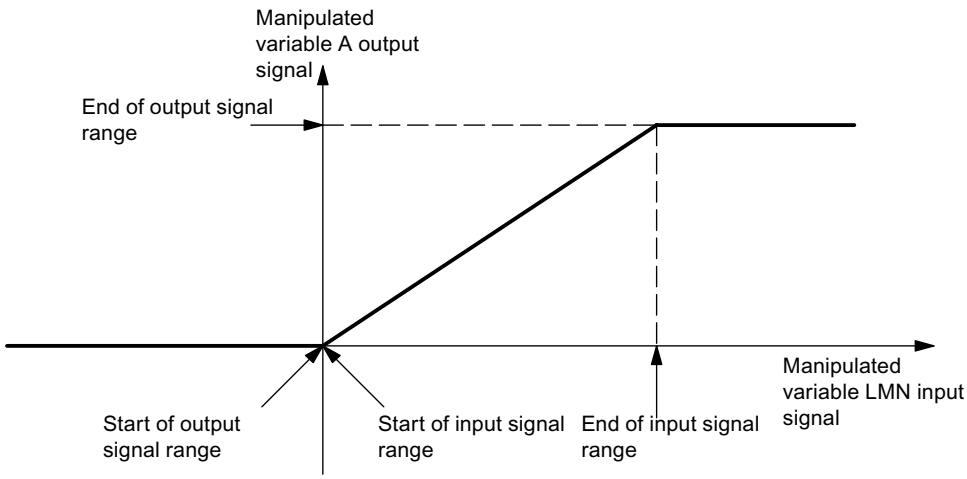

The statements above apply for the output value A in the case of a **three-step controller** (for example, a heating and cooling controller). The second signal for controlling the cooling is generated via output value B. The conversion of the output value to the output values A and B is shown in the figure below. The conversion to a binary output signal is carried out so that the ratio of pulse length to period duration corresponds to the output values A and B at the assigned digital outputs.

The figure below shows the split range function on a three-step controller:

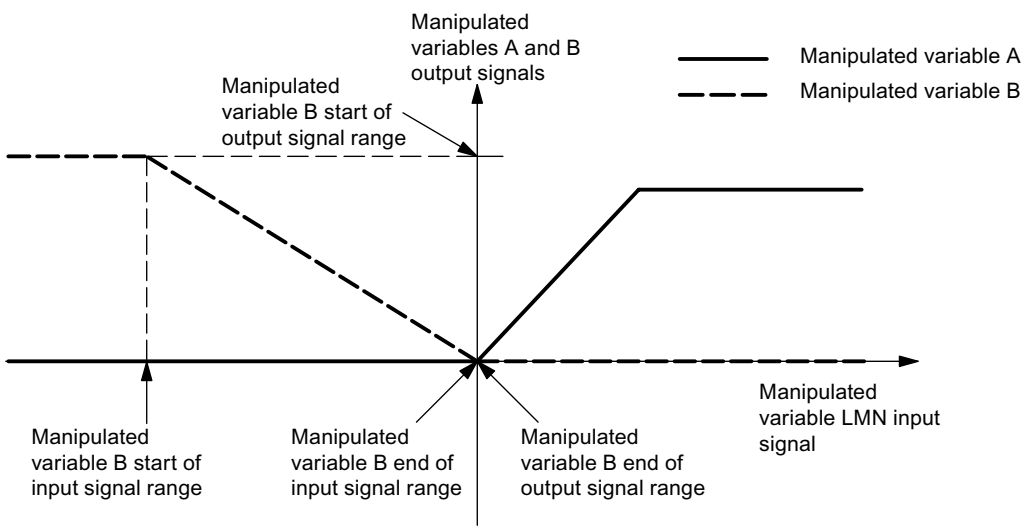

Pulse shaper of the step controller (S7-300, S7-400)

Overview

The pulse shaper is only available for step controllers (FM 355 S, FM 355-2 S and FM 455). The followings settings can be configured.

Pulse shaper (step controllers only)

- Motor control time
- Minimum duration of pulse
- Minimum pause time

Select output signal and signal type (S7-300, S7-400)

Analog output

You can select which signal is to be output for each channel at the analog output. This is usually the output value A of a controller. However, you can also select the output value B of a controller or indeed an analog input value. The latter can be used for the linearization of an analog value. This allows, for example, the signal supplied by a thermocouple to be linearized and converted to a range of 0 V to 10 V.

Controller output of the pulse controller

The following figure shows the controller output of the step controller (pulse control mode):

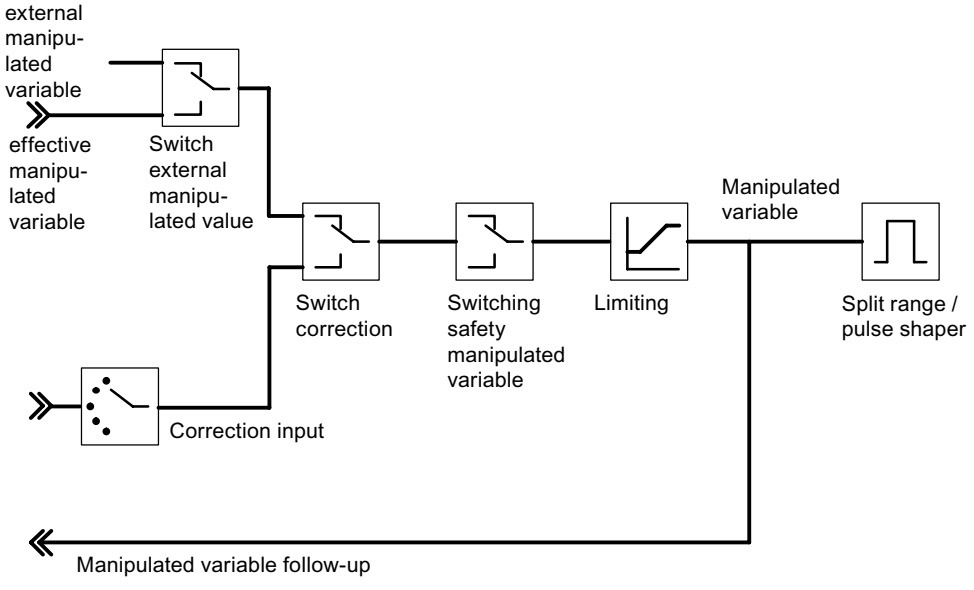

Controller output of the step controller

The following figure shows the controller output of the step controller (step control with position feedback mode):

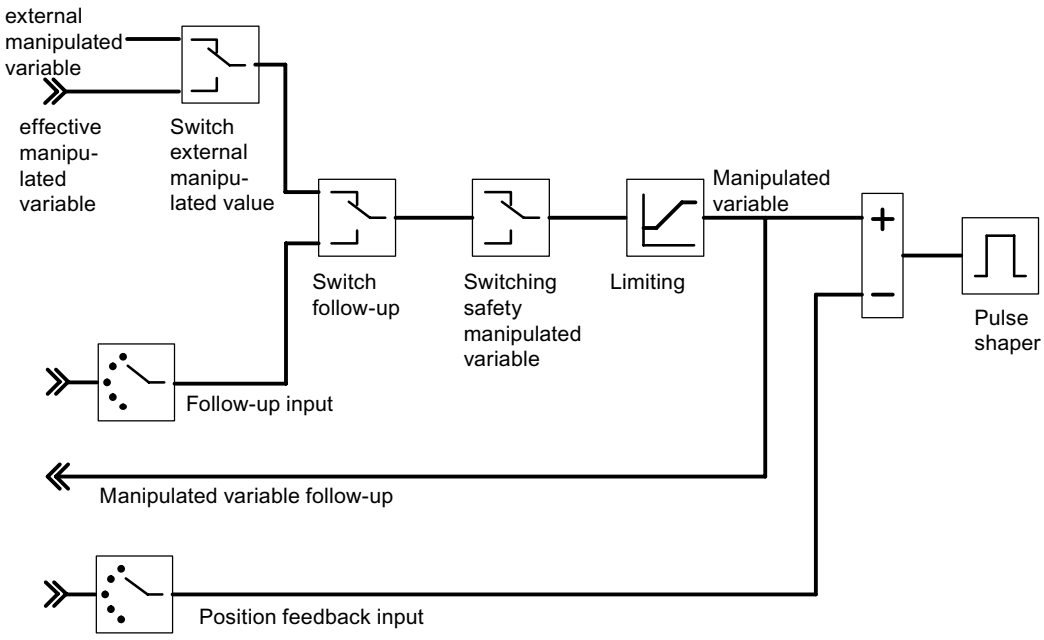

The following figure shows the controller output of the step controller (step control without position feedback mode):

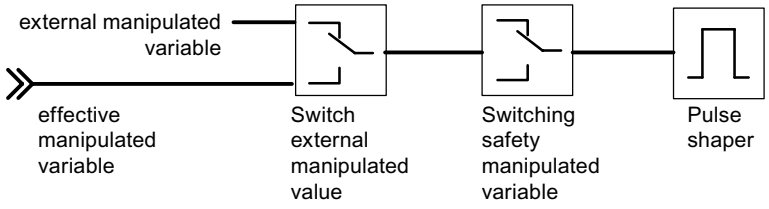

In a step controller without analog position feedback, the external output value and the safety output value act as follows:

If a value between 40.0% and 60.0% is specified, no binary output is set and the actuating device remains unchanged.

If a value > 60.0% is specified, "Actuating signal high" is output until the feedback "Actuating device at high limit" is triggered.

If a value < 40.0% is specified, "Actuating signal low" is output until the feedback "Actuating device at low limit" is triggered.

3.5.4.5 Shared Device support (S7-300, S7-400)

The controller modules FM 355 and FM 355-2 support the Shared Device functionality when using an ET 200M interface module.

General information on Shared Device is available at Configuring Shared Device.

Use of Shared Device with the controller modules

To use Shared Device with the controller modules, the following additional organization blocks must be included in your program:

- Insert/remove module interrupt OB (OB 83):
  For configuration as a Shared Device, the OB 83 must be integrated into the block list, because the transition from STOP to RUN will generate so-called "Return of submodule" interrupts. The alarms are intercepted by the OB 83, otherwise the CPU would go into STOP with the event "STOP caused by removing/inserting module".
- I/O access error OB (OB 122):
  As long as CPU A is in STOP and CPU B is in RUN, "I/O access errors, writing" will occur in CPU B if instructions from the "FM 355 PID Control" or "FM 355-2 Temp Control" libraries are called in the program or other write access operations to module addresses are carried out. OB 122 intercepts these errors; otherwise, CPU B would switch to STOP with the event "STOP caused by I/O access error".
- Priority class error OB (OB 85):
  OB 85 must only be included if the option "No OB85 call" was not configured for the setting "OB85 call if I/O access error occurs" in the CPU properties under "Cycle". For all other options, an OB 85 call is also requested for the I/O access errors that are described above for OB 122. If OB 85 does not exist in this case, the CPU goes to STOP with the event "STOP caused by program sequence error".

Both CPUs must be in "RUN" so that write access to the module addresses and the instructions from the "FM355 PID Control" and "FM355-2 Temp Control" libraries can be executed correctly.

As long as one CPU is in STOP and the other is in RUN, all controller modules of this interface module work in backup mode. This is independent of the "access" that took place in the "Shared Device" settings of the interface module. Only when all CPUs are in RUN do the controller modules work in normal operation.

3.5.5 Using FM 355 (S7-300, S7-400)

3.5.5.1 Configuring FM 355 (S7-300, S7-400)

Configuring the hardware (S7-300, S7-400)

Procedure

Configuration assumes that you have set up a project in which you can save the parameter assignment. The following briefly explains the most important steps.

1. Start the TIA portal and open the device configuration.
2. Select one module rack.
3. Select one controller module from your module catalog.
4. Move the controller module to the appropriate place on the module rack.
5. From the configuration table, note the input address of the module, for example 272.
   The value that you read is displayed in decimal format.

3.5.5.2 Parameter assignment FM 355 (S7-300, S7-400)

Module parameters (S7-300, S7-400)

Procedure

When assigning parameters you set the module parameters.

1. Double click in the device view under the "Parameter setting" module. Select the "Properties" tab in the inspector window.
   Result: The properties are displayed in the inspector window.
2. Set the basic settings of the module and the other parameters.
3. Set the further block settings in the project view.
4. Save the entered parameters with "Project > Save".
5. Translate the project with "Edit > Translate".
6. Transfer the parameter data with the CPU in STOP mode by selecting "Online > Load in device..."
   Result: The data is located in the CPU´s memory and will be directly transferred from there to the module.
7. Carry out a CPU start-up.

Analog input parameter (S7-300, S7-400)

Evaluation of the analog input (S7-300, S7-400)

Procedure

To select a type of sensor, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Analog Input > Input (No.) > Evaluation of Analog Input".
3. Set the following parameter:
   - Resolution of analog input
   - Analog input sensor type
     The "wire-break monitoring" options is open for different sensor types. Depending on the requirements, the option can be activated.
   - reference junction temperature
4. Save the entered parameters with "Project > Save".

Filter functions (S7-300, S7-400)

Procedure

To set a filter function, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Analog Input > Input (No.) > Filter functions".
3. Set the following parameter:
   - To filter according to a time constant, activate the "Signal filter" option.
     If the "Signal filter" option is selected, the filter time constant can be set.
   - Filter time constant
     Can only be set if the "Signal filter" option is activated.
   - To linearize the transmitter signal, activate the "Square root filter" option.
   - To linearize the transmitter signal characteristic, activate the "Polygon filter" option.
4. Save the entered parameters with "Project > Save".

Signal filter

Filter 1. order, whose time behavior is set through the time constants.

Square root filter

Serves for the linearization of the transmitter signals where the actual value exists as a physical size, which is in a quadratic connection with the measured process size.

Polygon filter

Serves for the linearization of transmitter characteristics.

The 13 optional supporting points can be selected in:

- mA for current input
- mV for voltage input

Scaling (S7-300, S7-400)

Procedure

To change the input signal into a physical unit, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Analog Input > Input (No.) > Scaling".
3. Set the following parameter:
   - High value
   - Low value
4. Save the entered parameters with "Project > Save".

Scaling

For the transfer of the input signal into a different physical unit through linear interpolation between the initial value (low value) and final value (high value).

Controller difference (S7-300, S7-400)

Realizing fixed value or cascade controller (S7-300, S7-400)

Fixed value or cascade controller

For the fixed value or cascade controller structures, the effective actual value is identical with the actual value A.

Select controller type

To realize a fixed value or cascade controller, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.)".
3. Select the option "Fixed value or cascade controller" in the "Controller type" area.
4. Save the entered parameters with "Project > Save".

Prepare setpoint

To prepare the setpoint, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. Set the corresponding parameters in the different areas.
   - Setpoint source
     Select the source of the setpoint.
   - Safety setpoint
     Set the safety setpoint here. Also set the reaction to a CPU-failure and the start-up behavior.
   - Setpoint ramp
     Set the start-up time.
   - Setpoint limit
     Select the high and low setpoint limits.
4. Save the entered parameters with "Project > Save".

Select signal source for actual value A

To select the signal source for the actual value A, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. Select the corresponding source in the "Manipulated value" area. The following selections are available for selection:
   - Zero
   - Prepared analog input
4. Save the entered parameters with "Project > Save".

Realizing three component controller (S7-300, S7-400)

Three component controller

For the three component controller structure, the effective actual value is formed by adding the three actual values A, B and C and through adding an adjustable offset. The actual values B and C can be additionally evaluated through factors.

Select controller type

Proceed as follows to implement a three-term controller:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.)".

3. Select the option "Three component controller" in the "Controller type" area.
4. Save the entered parameters with "Project > Save".

Prepare setpoint

To set how the setpoint should be prepared, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. In the "Setpoint source" area, select from which source the setpoint should be referenced.
4. Enter a value in the "safety setpoint" area that should serve as a safety value in the case of a CPU-error and during start-up.
5. Select the behavior of the controller module that should apply as "reaction to CPU error" and for "start-up behavior". The selection consists of:
   – Setpoint = last valid setpoint: upon the occurrence of the event, the setpoint from the last valid setpoint will be assigned. The last valid setpoint remains constant.
   – Setpoint = safety setpoint: upon the occurrence of the event, the setpoint will be assigned the safety setpoint. In this way, a secured emergency mode can be realized.
6. In the "Setpoint ramp" area, set the rise time that should be held with a change of the setpoint in order to reduce an overshooting.
7. In the "setpoint limit" area, set both an high and low limit for the setpoint.
8. Save the entered parameters with "Project > Save".

Select signal source for actual value A, B and C

To set the signal sources for the actual values A, B and C, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. In the "Manipulated value" area, select the respective signal source for each actual value. Here you can select if you want to use no signal ("zero") as a source or if you want to use prepared analog inputs.
4. Save the entered parameters with "Project > Save".

Enter factors for actual values B and C and offset

To set the signal sources for the actual values A, B and C, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. In the "Add" area, set the required weight factor for the actual values B and C.
4. Set the required "offset" that should be added to the actual values.
5. Save the entered parameters with "Project > Save".

Realizing ratio or mixture controller (S7-300, S7-400)

Select controller type

Proceed as follows to implement a ratio or blending controller:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.)".
3. Select the option "Ratio or Mixture controller" in the "Controller type" area.
4. Save the entered parameters with "Project > Save".

Prepare setpoint

To set how the setpoint should be prepared, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. In the "Setpoint source" area, select from which source the setpoint should be referenced.
4. Enter a value in the "safety setpoint" area that should serve as a safety value in the case of a CPU-error and during start-up.
5. Select the behavior of the controller module that should apply as "reaction to CPU error" and for "start-up behavior". The selection consists of:
   - Setpoint = last valid setpoint: upon the occurrence of the event, the setpoint from the last valid setpoint will be assigned. The last valid setpoint remains constant.
   - Setpoint = safety setpoint: upon the occurrence of the event, the setpoint will be assigned the safety setpoint. In this way, a secured emergency mode can be realized.
6. In the "Setpoint ramp" area, set the rise time that should be held with a change of the setpoint in order to reduce an overshooting.
7. In the "setpoint limit" area, set both an high and low limit for the setpoint.
8. Save the entered parameters with "Project > Save".

Select signal source for actual value A and D

To set the signal sources for the actual values A and D, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. In the "Manipulated value" area, select the respective signal source for each actual value. Here you can select if you want to use no signal ("zero") as a source or if you want to use prepared analog inputs, for the actual value D you can select the manipulated value from one of the controllers.
4. Save the entered parameters with "Project > Save".

Enter multiplier

To set the signal sources for the actual values A and D, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. Select the desired offset in the "Multiply" area.
4. Save the entered parameters with "Project > Save".

For ratio controllers, the actual value A is used as the controller size, the actual value D as the ratio size. The setpoint input serves as the ratio factor. Through multiplication with the actual value D and addition of an adjustable offset, an effective setpoint is prepared. If the actual value D is turned off, then only the offset will be added to the setpoint.

Setting alarm generation (S7-300, S7-400)

Procedure

Proceed as follows to set interrupt generation:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. Select the required options in the "alarm generation" area:
   - At "Limit monitor for", select for which signal the alarm should be set.
   - Under "Alarm high limit" you can set from which value the alarm should be activated upon reaching the high limit.
   - Under "Warning high limit" you can set from which value an alarm should be activated upon reaching the high limit.
   - Under "Warning low limit" you can set from which value an alarm should be activated upon reaching the low limit.
   - Under "Alarm low limit" you can set from which value the alarm should be activated upon reaching the low limit.
   - To avoid flickering on the monitoring displays, a hysteresis can be set under "Hysteresis".
4. Save the entered parameters with "Project > Save".

Controlling algorithm (S7-300, S7-400)

Realizing P-control (S7-300, S7-400)

Procedure

To realize a P-control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "PID-controller" in the "Type" area.
4. Select the option "P" in the "General" area under "Mode".
   All options that can be set for a P-control are activated, all other options are disabled.
5. Save the entered parameters with "Project > Save".

Realizing PI-control (S7-300, S7-400)

Procedure

To realize a PI-control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "PID-controller" in the "Type" area.
4. Select the option "PI" in the "General" area under "Mode".
   All options that can be set for a PI-control are activated, all other options are disabled.
5. Save the entered parameters with "Project > Save".

Realizing I-control (S7-300, S7-400)

Procedure

To realize a I-control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "PID-controller" in the "Type" area.
4. Select the option "I" in the "General" area under "Mode".
   All options that can be set for a I-control are activated, all other options are disabled.
5. Save the entered parameters with "Project > Save".

Realizing PID-control (S7-300, S7-400)

Procedure

To realize a PID-control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "PID-controller" in the "Type" area.
4. Select the option "PID" in the "General" area under "Mode".
   All options that can be set for a PID-control are activated, all other options are disabled.
5. Save the entered parameters with "Project > Save".

Realizing PD-control (S7-300, S7-400)

Procedure

To realize a PD-control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "PID-controller" in the "Type" area.
4. Select the option "PD" in the "General" area under "Mode".
   All options that can be set for a PD-control are activated, all other options are disabled.
5. Save the entered parameters with "Project > Save".

Realizing fuzzy temperature controller (S7-300, S7-400)

Procedure

To implement a fuzzy temperature control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "Fuzzy temperature controller module" in the "Type" area.
   All options that can be set for a fuzzy temperature controller module are activated, all other options are disabled.
4. Save the entered parameters with "Project > Save".

Suppressing the noise of the controller difference (S7-300, S7-400)

Procedure

To suppress the sound of the controller difference, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Set the "Dead zone width" setting in the "General" area.
4. Save the entered parameters with "Project > Save".

Turning on disturbance factor (S7-300, S7-400)

Procedure

To switch on the disturbance variable, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. In the "General" area, select the source for the disturbance variable from the "Disturbance variable source" list.
   Select the entry "Zero" in order not to switch on a disturbance variable.
4. Under "Disturbance variable factor", enter a value with which you want to weigh the disturbance variable.
   A value < 0 weakens the disturbance variable, a value > 0 strengthens the disturbance variable. With the value 1, the disturbance variable remains unchanged.
5. Save the entered parameters with "Project > Save".

Place P-component in feedback path (S7-300, S7-400)

Procedure

To realize a fixed value or cascade controller, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "In return" in the "P-component" area.
   If the option is activated, the P-component is placed in return.
4. Save the entered parameters with "Project > Save".

Weaken D-component for setpoint changes (S7-300, S7-400)

Select controller type

To realize a fixed value or cascade controller, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. In the "D-component" area, select the source for the D-component with the "D-component" option.
4. Set the factor for the D-component in the "Difference factor" area.
5. Save the entered parameters with "Project > Save".

Reverse controller effect (S7-300, S7-400)

Principle

Reversing the controller from the assignment "increasing controller difference = increasing manipulated variable" to "increasing controller difference = decreasing manipulated variable" can be done by setting a negative proportion value (GAIN).

Procedure

To reverse the controller effect, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Set a proportion value with the reverse prefix in the "General" area.
   If a positive proportion value is specified and the controller effect should be reversed, set a negative proportional value.
4. Save the entered parameters with "Project > Save".

Controller output (S7-300, S7-400)

Implementing a continuous controller (S7-300, S7-400)

A continuous controller can be created only with an analog module.

Procedure

1. Switching external manipulated value (Page 995)
2. Setting safety manipulated value (Page 996)
3. Switching tracing (Page 995)
4. Limiting manipulated value (Page 996)

5. Controlling two actuators - split range (Page 997)
6. Selecting output signal and signal type (Page 998)

Realizing a step controller without position feedback (S7-300, S7-400)

A step controller without position feedback can be realized only with a digital module.

Procedure

1. Select controller output step controller without position feedback
2. Switching external manipulated value (Page 995)
3. Setting safety manipulated value (Page 996)
4. Pulse shaper step controller (Page 997)

Realizing step-by-step controller with position feedback (S7-300, S7-400)

A step-by-step controller with position feedback can only be realized with a digital module.

Procedure

1. Select controller output step-by-step controller with position feedback
2. Switching external manipulated value (Page 995)
3. Switching tracing (Page 995)
4. Setting safety manipulated value (Page 996)
5. Limiting manipulated value (Page 996)
6. Setting position feedback (Page 995)
7. Pulse shaper step controller (Page 997)

Realizing pulse controller (S7-300, S7-400)

An pulse controller can only be realized with a digital module.

Procedure

1. Select controller output pulse controller
2. Switching external manipulated value (Page 995)
3. Switching tracing (Page 995)
4. Setting safety manipulated value (Page 996)
5. Limiting manipulated value (Page 996)
6. Prepare analog signal for conversion to a digital signal
7. Converting analog signal into a digital signal (Page 998)

Setting functions of the controller output (S7-300, S7-400)

Switching external manipulated value (S7-300, S7-400)

Procedure

To switch the external manipulated value, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller output".
3. In the "External Manipulated value" area, select the option through which the external manipulated value should be activated. Here you can choose from the following options:
   - Activation of function block.
   - Activation of a function block or from one of the digital inputs.
4. Save the entered parameters with "Project > Save".

Switching tracing (S7-300, S7-400)

Procedure

To realize a fixed value or cascade controller, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller output".
3. In the "Tracing" area, select the option through which the external manipulated value should be activated. Here you can choose from the following options:
   - Activation of function block.
   - Activation of a function block or from one of the digital inputs.
4. In the "tracing" area, select the source of the signal for the tracing. Here you can choose from the following options:
   - Zero, no source is used.
   - One of the analog inputs.
5. Save the entered parameters with "Project > Save".

Setting position feedback (S7-300, S7-400)

Procedure

To set the position feedback, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Digital output > Output (No.)".

3. In the "Position feedback input", select the "Source" option in order to select the signal for the position feedback.
   - Zero: no position feedback.
   - One of the prepared analog inputs.
4. Save the entered parameters with "Project > Save".

Setting safety manipulated value (S7-300, S7-400)

Procedure

To set the safety manipulated value, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller output".
3. In the "Safety manipulated value", set the value that should be used as the safety manipulated value.
4. Under "Activation", select the option through which the external manipulated value should be activated. Here you can choose from the following options:
   - Activation of function block.
   - Activation of a function block or from one of the digital inputs.
5. Under "Reaction upon startup of the module", select the option that should be executed upon starting up the module. Here you can choose from the following options:
   - Control mode: sets the module immediately into control mode upon startup.
   - Safety manipulated value: initially, the safety manipulated value is controlled.
6. Under "Reaction upon fault of sensor actual value A", select the option that should be executed if there is a fault of sensor actual value A.
7. Under "Reaction upon transducer disruption of an input", select the option that should be executed if there is a transducer disruption of an input.
8. Save the entered parameters with "Project > Save".

Limiting manipulated value (S7-300, S7-400)

Procedure

To limit the manipulated value, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller output".

3. In the "Output signal limit", set the following value:
   - High value: illustrates the high limit for the output signal.
   - Low value: illustrates the low limit for the output signal.
4. Save the entered parameters with "Project > Save".

Controlling two actuators - split range (S7-300, S7-400)

Procedure

To control two actuators, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller output".
3. In the "Split Range" area, select the "Activate" option to control two actuators.
4. Set the following value for the input and output signals from A and also for the input and output signals from B:
   - Low input signal.
   - High input signal.
   - Low output signal.
   - High output signal.
5. Save the entered parameters with "Project > Save".

Pulse shaper step controller (S7-300, S7-400)

Procedure

To set the pulse shaper, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Digital output > Output (No.)".
3. Set the following parameter in the "Pulse shaper" area:
   - Motor manipulating time.
   - Minimum impulse duration.
   - Minimum break duration.
4. Select the fitting option for the high and low value as to where the value should be taken over from:
   - From the function block.
   - From the function block or from one of the digital inputs.
5. Save the entered parameters with "Project > Save".

Converting analog signal into a digital signal (S7-300, S7-400)

Procedure

To convert an analog signal into a digital signal, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Digital output > Channel (No.)".
3. Set the following parameter in the "Pulse shaper" area:
   - Minimum impulse duration.
   - Minimum break duration.
4. Save the entered parameters with "Project > Save".

Selecting output signal and signal type (S7-300, S7-400)

Procedure

To select the output signal and the signal type, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Analog output > Output (No.)".
3. Under the "Source" option, select which signal should be used as the output signal. You can choose from the following:
   - Zero: no source for the output signal.
   - One of the analog inputs.
   - One of the manipulated values for controller A or B.
4. Under the "Signal type" option, select the type in which the output signal should be emitted.
5. Save the entered parameters with "Project > Save".

3.5.5.3 Programming FM 355 (S7-300, S7-400)

3.5.5.4 Commissioning the FM 355 (S7-300, S7-400)

Creating and supplying instance DBs (S7-300, S7-400)

Procedure

To meet the requirements for commissioning, follow these steps:

1. Create a user program that calls the PID_FM instruction cyclically in a cyclic interrupt OB.
2. Set READ_VAR = TRUE cyclically in the user program.

3. Create a single-instance DB of the PID_FM instruction for each controller channel.
4. Enter the module address for each instance DB in the PID_FM.MOD_ADDR parameter. Use the start address that you specified when you configured your hardware as the module address of the FM 355.
5. Enter the number of the controller channel (1 ... 4) in the PID_FM.CHANNEL parameter.
6. If you use the PID_PAR instruction, enter the number of the controller channel in the PID_PAR.CHANNEL parameter.
7. Check to ensure that the "Sensor type analog input" of this channel is selected.
   If the analog input is not processed, all PID parameters are set to zero and a manipulated variable is not calculated.

Note

Multi-instance DBs

If you use multi-instance DBs in your project, you must commission the FM via a watch table.

Commissioning the controller (S7-300, S7-400)

Introduction

You tune the PID parameters as follows:

- PID/PI controllers (step controller, continuous controller, pulse controller): Manually
- Fuzzy temperature controller: Automatically
- Step controller: Automatically with instruction TUN_ES
- Continuous controller: Automatically with instruction TUN_EC
- Pulse controller: Automatically with instruction TUN_EC

Procedure

Proceed as follows to tune a controller:

1. Put the CPU in RUN mode.
2. If you are using a step controller, measure the motor actuating time.
3. Tune the PID parameters with one of the options listed above.
4. Monitor the control loop with the trend view.

See also

Measuring the motor transition time (Page 1006)

Tuning PID parameters manually (S7-300, S7-400)

Requirement

You have selected the "PID controller" control algorithm during parameter assignment.

Procedure

To set a PID/PI controller manually, proceed as follows:

1. Select the instance DB of the channel you want to tune.
2. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
3. Enter values for the proportional action, integral action, derivative action and delay.
4. Click the icon "Send PID parameters to FM" in the "Tuning" group.
5. Select the "Change setpoint" check box in the "Current values" group.
6. Enter a new setpoint.
7. Clear the check box "Manual mode", if necessary, and click the icon in the "Current values" group.
   The controller works with the new parameters and controls the new setpoint.
8. Check the quality of the PID parameters based on the trend characteristics.
9. Repeat steps 3 to 8 until you are satisfied with the controller results.

Fuzzy system identification (S7-300, S7-400)

Requirements on process with fuzzy temperature controller (S7-300, S7-400)

Requirement

The process should fulfill the following requirements in order to achieve optimal control using the fuzzy temperature controller:

- At bath heatings the liquid to be heated has to be mixed thoroughly.
- In case of bath-in-bath control systems both liquids have to be mixed thoroughly. At the same time good heat transitions between all the heat-transferring media have to be ensured. In the case of materials with poor heat-transferring properties large transfer surfaces should ensure good heat transportation.
- In case of room temperature control systems thorough mixing (for example with fans) has to be ensured.
- The controlling system gain may not exceed the factor of 3.

- The delay time may not exceed 3% of the recovery time.
- The temperature to be controlled should change by a maximum of 1‰ of the specified maximum temperature at the maximum manipulated value output within the sampling time of the controller.

Classification of the Controlled Systems

A controlled system or a process to be controlled is characterized by parameters such as the heat output, the heating mass or the heating capacity of the medium to be heated. With regard to the fuzzy controller a difference is made between "critical" and "non-critical" temperature controlled systems as follows: The control system becomes increasingly critical:

- The greater the heat output,
- the greater the heating capacity of the heating,
- The lower the heating capacity of the medium to be heated,
- the greater the heat transition resistance,
- the smaller the heat transfer surface.

After a manipulated value step change has been applied to the controlled system, it reacts with a step response. The controlled system can also be classified on the basis of this step response: The control system becomes increasingly critical the greater the ratio t_u / t_g is and the greater the controlled system gain is. With t_u / t_g < 1/10 you have a non-critical control section.

See also

Characteristic values of the control section (Page 614)

Process of fuzzy system identification (S7-300, S7-400)

Process of parameter tuning with fuzzy temperature controller

Parameter optimization is based on a self-optimizing fuzzy controller.

The identification of the controlled system begins with a monitoring phase during which no heat output takes place. The duration of the monitoring phase is as follows:

- Monitoring phase continuous controller: Approx. 1 min
- Monitoring phase step controller: Approx. 1 min + 1/2 x actuating time of final control element

This time is used in order to determine temperature trends in the heating medium. Afterwards the maximum heating output of 100% is output. This is visible at the LMN output parameter in the instance DB of the PID_FM instruction.

The range of the first 4% of the temperature increase of the setpoint value range is used for the identification, whereby information about the process response is obtained at 1% and 4% respectively of the temperature increase on the basis of the time that has passed.

The identification is complete when the heating output becomes less than 100%. This behavior can be used, for example, to deactivate the identification via the user program. After the identification phase the controller continues to operate with the determined parameters.

If the setpoint value step change is not adjusted and if the heating output remains permanently at zero, the identification has been terminated unsuccessfully, meaning that the controller cannot control the connected controlled system.

Controller behavior with different control sections

Problems do not arise in case of an "uncritical" controlled system, neither during identification nor during controlling.

The identification of a controlled system that is "too critical" is cancelled. Controlling of an identified "critical" controlled system is carried out very "carefully" or slowly.

Performing fuzzy system identification (S7-300, S7-400)

Requirement

You have selected the "Fuzzy controller" control algorithm during parameter assignment.

You have to ensure as far as possible that the controlled system has settled (no heating-up or cooling-down process) or is changing slowly and monotonously before the identification is carried out. The criterion can be that the temperature change is to approach a straight during a period of one minute. At faster processes this requirement is particularly relevant.

Since the manipulated variable zero is output for approx. one minute by the controller at the beginning of identification, the temperature to be controlled has to lie near the ambient temperature.

Starting fuzzy system identification

Follow these steps to tune a fuzzy controller:

1. Select the instance DB of the channel you want to tune.
2. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
3. Enter a new setpoint (SP_OP) in the "Tuning" group.
   The new setpoint must meet the following conditions:
   – Setpoint step change > 5%
   – New setpoint > process value + setpoint high limit x 0.12
4. Click "Fuzzy process identification" in the "Tuning" group.

Result

The new setpoint (SP_OP), SP_OP_ON = TRUE and FUZID_ON = TRUE are sent to the FM. Starting the fuzzy system identification. The IDSTATUS parameter of the CH_DIAG instruction provides information on the identification state. You see in the status display field if a fuzzy system identification is activated or running.

Deactivating fuzzy system identification

As long as the bit FUZID_ON = TRUE, the next identification is started whenever the setpoint step change is sufficiently large.

Proceed as follows to turn off the tuning mode:

1. Enter FUZID_ON in a watch table.
2. Set FUZID_ON = FALSE.

Note

Deactivation of the tuning mode FUZID_ON = FALSE before identification has been completed does not stop the identification. An identification process that has been started continues to run under all circumstances – with the exception of a negative setpoint value step change.

Repeating Fuzzy system identification

A renewed identification is carried out with each additional setpoint step change ≥12% unless the identification has been deactivated again with FUZID_ON = FALSE.

It is also possible to restart the identification by reducing and then increasing the setpoint by 12> %.

Canceling fuzzy system identification

Identification can be canceled in the following cases:

- By the controller if a "critical" controlled system is identified. After canceling the controller is in the error state. This state is indicated by the fact that the manipulated variable is reset permanently by the controller. This is also not changed by deactivating the optimization. The error state is deleted by starting a new identification.
  Information about the state of the identification is available through the IDSTATUS parameter of the CH_DIAG FB. More information on this can be found at the end of this section under "Controller State Information".
- By the operator by generating a negative setpoint value step change, whereby the setpoint value must lie below that of Condition 2 (see above "How To Start identification").

Proceed as follows to cancel tuning:

1. Enter a new setpoint that is less than the current process value and meets the same conditions as for the start of fuzzy system identification.
2. Click ▶ "Fuzzy process identification" in the "Tuning" group.

Downloading tuned parameters in device (S7-300, S7-400)

Introduction

You can proceed as follows to prevent the parameters on the FM from being overwritten with the system data the next time you start up the CPU:

- You download the tuned PID parameters from the instance DB of the PID_FM to the FM after each startup of the FM with LOAD_PAR=TRUE.
- Download the PID parameters to the SDB.

Note

The PI, PID and SAVE parameter records cannot be stored in the SDB.

Procedure

Proceed as follows to download the parameters to the SDB:

1. Click on the icon "Transfer online data to an offline project".
   The PID parameters are saved in the parameter setting of the FM.
2. Stop CPU.
3. Click "Download to device".

Working with the trend view (S7-300, S7-400)

Monitoring

1. Double click in the device view under the "Commissioning" module.
2. Navigate in the project view to "Trend view".
3. Select the instance DB you wish to monitor. The channel to which the instance DB has been assigned will be displayed.
4. Specify the update cycle in which the trend view is to record.
5. Click on "Measurement on" to start measuring.
6. Click on "Measurement off" to stop measuring.

Procedure

To adjust the scale, proceed as follows:

1. Double click on "Commissioning" in the device view of the module.
2. Navigate to "Trend view" in the configuration view.

3. To change the time axis:
   - Click on the arrow-button next to the date to open a calendar. Select a date from the calendar. To lock the selection, click on the blue lock. This possibility exists for the beginning and end dates.
   - Click on the time scale. The mouse pointer will be illustrated as a left/right arrow. Press the mouse and drag it to the left or right.
     If the beginning or end time is locked, the time scaling will change.
     If neither the beginning nor the end time is locked, the specified period will change, the scaling remains the same.
4. To change the value axis:
   - Click on the input field for the value displayed above or below. Enter your desired value here. To lock the selection, click on the blue lock. This possibility exists for the high and low value.
   - Click on the value scale. The mouse pointer will be illustrated as an up/down arrow. Press the mouse and drag it up or down.
     If the high or low value is locked, the scaling of the value axis will change.
     If neither the high nor the low value is locked, the specified range will change, the scaling remains the same.

Changing parameters with the HMI (S7-300, S7-400)

Procedure

Proceed as follows to change the controller parameters of the instruction PID_FM with the HMI:

1. Write the parameter you want to change from the HMI to an auxiliary DB on the CPU (see ①).
2. Set PID_FM.COM_RST to TRUE.
   The instruction performs an initialization run. The parameters are synchronized between FM and CPU (see ②).
3. Transfer the parameters you want to changed from the auxiliary DB to the instance DB of the instruction PID_FM (see ③).
4. Set PID_FM.LOAD_PAR to TRUE.
   The changed parameters are transferred to the FM (see ④).

It is necessary to store the parameters in an auxiliary DB because PID_FM after the start of the CPU with COM_RST = TRUE reads those parameters from the module that the CPU has previously transferred from the system data to the FM.

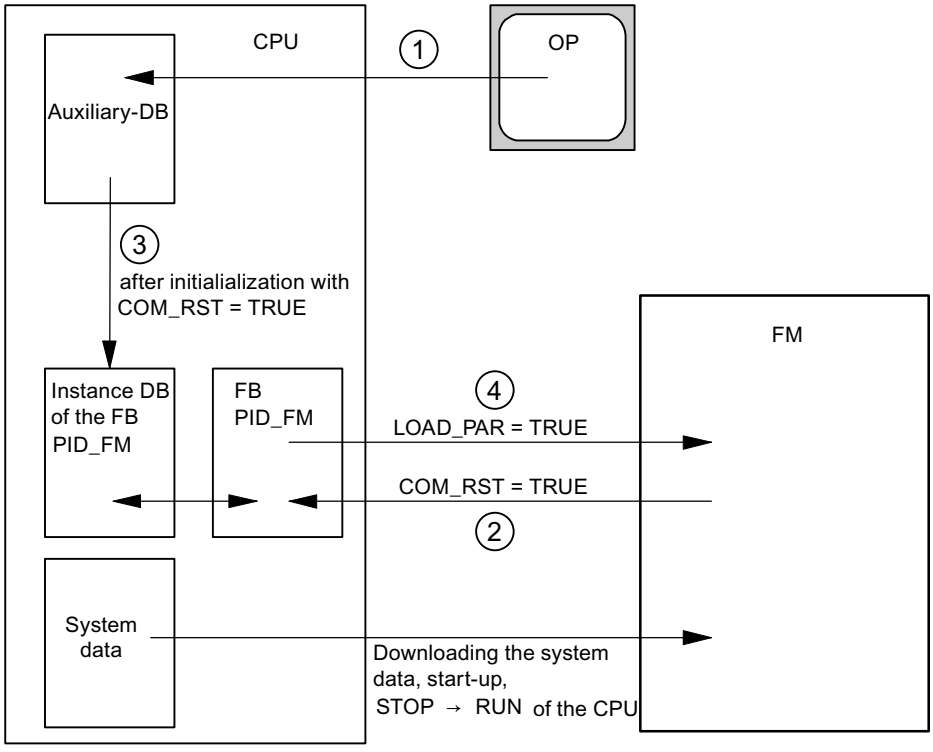

Measuring the motor transition time (S7-300, S7-400)

Introduction

The motor transition time has to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Requirement

- The configured motor transition time must be half the size of the real transition time. The actuator is moved for a maximum of twice the transition time in one direction.
- The position feedback must be interconnected for actuators with analog position feedback.
- The endstop signals must be interconnected for actuators with endstop signals.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use position feedback" check box.
3. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. Click the "Start transition time measurement" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (LMNSOPON = TRUE). The actuator will stop being moved.

Actuators with endstop signals

Proceed as follows to measure motor transition time with endstop signals:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use actuator endstop signals" check box.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop. Time measurement will start once the actuator has reached the lower endstop and will end when the actuator reaches the lower endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (LMNSOPON = TRUE). The actuator will stop being moved.

Loading measured transition time

1. Click "Transfer online data to an offline project" to save the measured motor transition time locally in the project.
   The measured transition time is saved in the parameter setting of the FM.
2. Click "Download to device" to activate all changes the next time you start the CPU.

Cancelling transition time measurement

When you cancel the transition time measurement, the manual mode of the actuator signals is switched on (LMNSOPON = TRUE) and the manual mode with manipulated value specification is switched off (LMNOP_ON = FALSE). The actuator will stop being moved.

3.5.5.5 FM 355: Diagnostics with LED display (S7-300, S7-400)

Error display from the group error LED (S7-300, S7-400)

Introduction

If the red group error LED lights up, there is either an error on the module (internal error) or in the line connections (external error).

If the yellow LED flashes, then the firmware has been deleted. This status can only occur in the case of faulty hardware or if the loading procedure of the firmware is aborted.

Which errors are displayed?

The following errors are displayed by the group error LED lighting up:

Type of error	Diagnostic message	Possible cause	Correction
Internal errors	Module defective	Hardware error	Replace the module
	Time watchdog tripped	Hardware error	Replace the module
	EEPROM content is invalid	Failure of the supply voltage when configuring	Reconfigure module
External errors	Incorrect parameters in module	Incorrect parameters have been transferred to the module	Reassign the module parameters
	Errors with the analog inputs or analog outputs	Analog input hardware error	Replace module
		Analog input wire break	Remedy wire break
		Analog input measuring range violation (underrange)	Check measuring signal
		Analog input measuring range violation (overrange)	Check the measuring signal
		Analog output wire break	Remedy wire break
		Analog output short circuit	Eliminate short circuit
	Missing external auxiliary supply	24 V supply missing	Restore 24 V supply

Diagnostic interrupt in the case of errors

All errors can trigger a diagnostic interrupt if you have enabled the diagnostic interrupt in the respective parameterization screen. From the diagnostic data records DS0 and DS1 you can see which errors have caused the LED to light up. The assignment of the diagnostic data records DS0 and DS1 is described in the next section.

Triggering diagnostic interrupts (S7-300, S7-400)

What is a diagnostic interrupt?

If the user program reacts to an internal or external error, you can configure a diagnostic interrupt that interrupts the cyclic program of the CPU and calls the diagnostic interrupt OB, OB 82.

Which events can trigger a diagnostic interrupt?

The list shows which events can trigger a diagnostic interrupt:

- The module has not been configured or is incorrectly configured
- Module defective
- Wire break with analog inputs (only 4 to 20 mA)
- High limit violated and low limit violated with analog inputs
- Load break and short circuit with analog outputs

Default setting

The diagnostic interrupt is blocked by default.

Enabling the diagnostic interrupt

In the "Basic parameters" screen you can block or enable the diagnostic interrupt for the module.

Responses to an interrupt-triggering event

The following happens when an event occurs that could trigger a diagnostic interrupt:

- The diagnostic information is stored in the diagnostic records DS0 and DS1 on the module.
- The group error LED lights up.
- The diagnostic interrupt OB (OB 82) is called.
- The diagnostic record DS0 is entered in the start information of the diagnostic interrupt OB.
- If there is no hardware fault, the module continues to control.

If no OB 82 is programmed, the CPU goes to STOP.

Diagnostic data record DS0 and DS1

The information as to which event has triggered a diagnostic interrupt is stored in the diagnostic data records DS0 and DS1. The diagnostic data record DS0 contains four bytes, the DS1 16 bytes, whereby the first four bytes are identical to those of the DS0.

Diagnostic data records DS0 and DS1 (S7-300, S7-400)

Read data record from the module

The diagnostic data record DS0 is automatically transferred to the start information when the diagnostic OB is called. There these four bytes are stored in the local data (byte 8 - 11) of the OB 82.

The diagnostic data record DS1 (and hence also the content of the DS0) can be read from the module using the "CH_DIAG" instruction. This only makes sense if an error is reported in a channel in the DS0.

The "CH_DIAG" instruction must be called in the same OB as the FB "PID_FM". This is achieved by the following measure: Set a bit while the OB 82 is being executed. Query this bit in the OB in which the FB "PID_FM" is called and then call the "CH_DIAG" instruction with a set bit.

How does the diagnosis text appear in the diagnostics buffer?

If you want to enter the diagnostics message in the diagnostics buffer, call the "WR_REC" instruction "Enter user-specific message in the diagnostics buffer" in the user program. The event number of the respective diagnostics message is specified in the input parameter EVENTN. The interrupt is entered in the diagnostics buffer defined as x = 1 for incoming and x = 0 for outgoing. The diagnostics buffer contains the relevant diagnostics text in the "Meaning" column as well as the time of the entry.

Assignments of the Diagnostics Record DS0 in the Start Information

The following table shows the assignments of the diagnostics record DS0 in the start information. All unlisted bits are not significant and are set to zero.

Table 3-3 Assignments of diagnostics record DS0

Byte	Bit	Meaning	Remark	Event No.
0	0	Module malfunction	Is set at every diagnostics event	8:x:00
	1	Internal fault	Is set at all internal faults: • Watchdog time-out • EEPROM contents invalid. Module starts up without controlling and waits for renewed parameter configuration by the CPU. • EPROM error • ADC/DAC fault • Analog input, hardware fault	8:x:01
	2	External error	Is set at all external errors: • Missing external auxiliary voltage • Parameter configuration faulty • Analog input, wire breakage (only range 4 to 20 mA) • Analog input, measuring range violation (underrange) • Analog input, measuring range violation (overrange) • Analog output, wire breakage • Analog output, short circuit	8:x:02
	3	Fault in a channel	See DS1, from byte 7, for further breakdown	8:x:03
	4	Missing external auxiliary voltage	24 V power supply of the FM 355 failed	8:x:04
	6	EEPROM contents invalid	Failure of the supply voltage during a writing process to the EEPROM. The module starts up with default parameters.	8:x:03
	7	Parameter configuration faulty	The module cannot use a parameter. Reason: Parameter unknown or impermissible combination of parameters. Refer to the menu **PLC > Parameter Assignment Error**	8:x:07
1	0 ... 3	Module class	Always has 8 assigned	–
	4	Channel-specific diagnostics	Is set if the module can supply additional channel information and if a channel error exists (refer to DS 1 Byte 7 to 12)	–
2	3	Watchdog time-out	Hardware fault	8:x:33
3	2	EPROM error	Module defective	8:x:42
	4	ADC/DAC fault	Module defective	8:x:44

Assignments of Diagnostics Record DS1

The diagnostics record DS1 consists of 16 bytes. The first 4 bytes are identical with diagnostics record DS0. The following table shows the assignment of the remaining bytes. All unlisted bits are not significant and are set to zero.

Table 3-4 Assignment of Bytes 4 to 12 of the diagnostics record DS1

Byte	Bit	Meaning	Remark		Event No.
4	0 ... 7	Channel type	Always has 75H assigned		–
5	0 ... 7	Length of the diagnostic information	Always has 8 assigned		–
6	0 ... 7	Number of channels	Always has 5 assigned (4 controllers + 1 reference channel)		–
7	0 ... 7	Channel error vector	One bit is assigned to each channel		–
8	0	Analog input, hardware fault	Channel-specific diagnostics channel 1		8:x:B0
	1	Unused			8:x:B1
	2	Analog input wire break (only area 4 to 20 mA)			8:x:B2
	3	Unused			8:x:B3
	4	Analog input below measuring range			8:x:B4
	5	Analog input, measuring range violation (overrange)			8:x:B5
	6	Analog output, wire breakage		Only for current output of continuous controller	8:x:B6
	7	Analog output, short circuit		Only with the current output of the C controller	8:x:B7
9	0 ... 7	See Byte 8	Channel-specific diagnostics channel 2		See above
10	0 ... 7	See Byte 8	Channel-specific diagnostics channel 3		See above
11	0 ... 7	See Byte 8	Channel-specific diagnostics channel 4		See above
12	0 ... 5	See Byte 8	Diagnostics for reference channel		See above

Additional information

For more information on the "RET_VALU" parameter, refer to List of RET_VALU messages (Page 1014) and the "Extended instructions" section of the help system.

Measuring transducer fault (S7-300, S7-400)

Faults at Measuring Transducers

The following measuring transducer faults can be recognized by the controller module:

- Measuring range violation (underrange)
- Measuring range violation (overrange)
- Wiring breakage (not at all measuring ranges)

If one of these faults occurs, the group error bit "External error" is set in the diagnostics record DS0 and the channel-specific error bits in the diagnostics record DS1 (refer to the tables in the previous section). When these faults disappear, the corresponding bits are reset.

The following table shows at which limits in the individual measuring ranges the error bits are set and reset:

Measuring range	Error bit measuring range violation (underrange) at ...	Error bit measuring range violation (overrange) at ...	Error bit wire breakage indication
	DS1: Byte 10 to 26, Bit 4	DS1: Byte 10 to 26, Bit 5	DS1: Byte 10 to 26, Bit 2
0 to 20 mA	< –3.5 mA	> 23.5 mA	–
4 to 20 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at < 3.8 mA	> 22.8 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at > 3.8 mA
0 V to 10 V	<–1.175 V	> 11.75 V	–
Pt 100 (–200 to 850°C) (–328 to 1562°F)	< 30.82 mV	> 650.46 mV	–
Pt 100 (–200 to 556°C) (–328 to 1032°F)	< 30.82 mV	> 499.06 mV	–
Pt 100 (–200 to 130°C) (–328 to 264°F)	< 30.82 mV	> 254.12 mV	–
Thermocouple Type B	< 0 mV	> 13.81 mV	–
Thermocouple Type J	< –8.1 mV	> 69.54 mV	–
Thermocouple Type K	< –6.45 mV	> 54.88 mV	–
Thermocouple Type R	< –0.23 mV	> 21.11 mV	–
Thermocouple Type S	< –0.24 mV	> 18.7 mV	–
Free thermocouple	< Low input value of the polyline	< High input value of the polyline	–

List of RET_VALU messages (S7-300, S7-400)

RET_VALU messages

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
7000	28672	-32624	First call with REQ=0: no data transmission active; BUSY has the value 0.
7001	28673	-32624	First call with REQ=1: data transmission initiated; BUSY has the value 1.
7002	28674	-32624	Interim call (REQ irrelevant). Data transfer already active; BUSY has the value 1.
8090	32912	-32624	Specified logical base address invalid: There is no assignment in the SDB1/SDB2x, or it is not a base address.
80A0	32928	-32608	Negative acknowledgment when reading from the module. Module was removed during the read operation or the module is defective.
80A1	32929	-32607	Negative acknowledgment when writing to the module. Module was removed during the write operation or the module is defective.
80A2	32930	-32606	Protocol error at layer 2
80A3	32931	-32605	Protocol error involving user interface/user
80A4	32932	-32604	Communication bus error
80B1	32945	-32591	Incorrect length specification. FM_TYPE parameter in channel DB not set correctly for the module in use.
80B2	32946	-32590	The configured slot is not being used.
80B3	32947	-32589	Actual module type is not match configured module type.
80C0	32960	-32576	Module data not ready for reading.
80C1	32961	-32575	Data of a write job of the same type have not yet been processed by the module.
80C2	32962	-32574	The module is currently processing the maximum possible number of jobs.
80C3	32963	-32573	Required resources (memory etc.) currently occupied.
80C4	32964	-32572	Communication error
80C5	32965	-32571	Distributed I/O not available.
80C6	32966	-32570	Priority class abort (restart or background).
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read off the DB. (Write job)
8532	34098	-31438	DB number of the parameter DB too high (Write job)
853A	34106	-31430	Parameter DB not present. (write job)
8544	34116	-31420	Error at n-th (n > 1) read access to a DB after an error has occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written to the DB. (Read job)
8730	34608	-30928	Parameter DB in the CPU is write protected. The data cannot be written to the DB (read job)
8732	34610	-30926	DB number of the parameter DB too high (Read job)

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error at n-th (n > 1) write access to a DB after an error has occurred. (Read job)
80ff	33023	-32513	Incorrect index specification with block FMT_PAR
Errors 80A2..80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time. Messages of the 7xxx form indicate temporary operating states of communication.			

3.5.5.6 Example projects for FM 355 (S7-300, S7-400)

3.5.6 Using FM 355-2 (S7-300, S7-400)

3.5.6.1 Configuring FM 355-2 (S7-300, S7-400)

Configuring the hardware (S7-300, S7-400)

Procedure

Configuration assumes that you have set up a project in which you can save the parameter assignment. The following briefly explains only the most important steps.

1. Start the TIA portal and open the device configuration.
2. Select one module rack.
3. Select one controller module from your module catalog.
4. Move the controller module to the appropriate place on the module rack.
5. From the configuration table, note the input address of the module, for example 272.
   The value that you read is displayed in decimal format.

3.5.6.2 Parameter assignment FM 355-2 (S7-300, S7-400)

Module parameters (S7-300, S7-400)

Procedure

When assigning parameters you set the module parameters.

1. Double-click on the "Parameter setting" in the Device view of the module. In the Inspector window, select the "Properties" tab.
   Result: The properties are displayed in the inspector window.
2. Configure the basic parameters and the remaining parameters of the module.

3. Configure the remaining device parameters in the Design view.
4. Save the parameters entered with "Project> Save".
5. Translate the project with "Edit > Translate".
6. Load the configuration data with the CPU in STOP mode by selecting "Online > Device load..."
   Result: The data stored in the memory of the CPU are then transferred directly to the module.
7. Carry out a CPU start-up.

Analog input parameter (S7-300, S7-400)

Evaluation of analog inputs (S7-300, S7-400)

Procedure

To select a sensor type, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Digital input > input [No]> Evaluation of analog inputs".
3. Enter the following parameters:
   – Sensor type analog input
     For different sensor types the option "open-circuit monitoring" will be enabled. The option may be activated depending on requirements.
   – Junction temperature
4. Save the parameters entered with "Project> Save".

Filter functions (S7-300, S7-400)

Procedure

To set the filter functions, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Analog input > input [No]> Filter functions".

3. Enter the following parameters:
   - To filter on a time constant you should enable the "signal filter" option.
     If the "signal filter" option is selected, the time constant filter can be set.
   - Filter time constant
     Can only be set if the "signal filter" option is enabled.
   - To linearize the encoder signal you should activate the "square root filter" option.
   - To linearize the encoder signal characteristics you should activate the "polygon filter" option.
4. Save the parameters entered with "Project> Save".

Signal filter

Filter 1. order, whose time behavior is set through the time constants.

Square root filter

Used to linearize encoder signals in which the actual value is present as a physical quantity, which is square in relation to the measured process variable.

Polygon filter

Used to linearize encoder characteristics.

The 13 selectable points are selectable in:

- mA at current input
- mV with voltage input

Scaling (S7-300, S7-400)

Procedure

To convert the input signal into a physical unit, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Analog input > input [No]> Scaling".
3. Enter the following parameters:
   - High value
   - Low value
4. Save the parameters entered with "Project> Save".

Scaling

To convert the input signal into another physical unit through linear interpolation between the initial value (low value) and the ending value (high value).

Deviation (S7-300, S7-400)

Implementing a constant or a cascade controller (S7-300, S7-400)

Constant or cascade controller

In the controller structure of constant or cascade controllers the effective current value is equal to the actual value A.

Selecting the controller type

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]".
3. In the "Encoder type" section select the "Constant or Cascade controller".
4. Save the parameters entered with "Project> Save".

Processing setpoints

To edit setpoints, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> Channel [No]> Deviation".
3. Set in the different areas the corresponding parameters:
   - Setpoint source
     Select the source of the setpoint.
   - Safety setpoint
     Set your safety setpoints here. Set also the response to a CPU fault and the startup behavior.
   - Setpoint ramp
     Set the start-up time.
   - Setpoint limit
     Select the high and low setpoint limits.
4. Save the parameters entered with "Project> Save".

Selecting a signal source for actual value A

To select the signal source for the actual value A, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> Channel [No]> Deviation".

3. In the "Control value" section select the corresponding source. The following choices are available:
   - Zero
   - Processed analog input
4. Save the parameters entered with "Project> Save".

Implementing three component controllers (S7-300, S7-400)

Three component controller

In the controller structure of a three component controller the effective actual value is obtained by summing the three actual values of A, B and C and then adding an adjustable offset. The actual values of B and C can also be evaluated via factors.

Selecting the controller type

Proceed as follows to implement a three-term controller:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]".
3. In the "Controller type" section select the "Constant or Cascade controller" option.
4. Save the parameters entered with "Project> Save".

Processing setpoints

To adjust the setpoint processing mode, proceed as follows:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> Channel [No]> Deviation".
3. In the "Setpoint source" section select the source from which the setpoint shall be obtained.
4. In the "Safety setpoint" section enter a value to be utilized upon CPU fault and start-up as a safety value.
5. Select the behavior of the controller module considering the values of the "Reaction to CPU errors" and the "Start-up behavior" options. You may select amongst the following options:
   - Setpoint = last valid setpoint: the setpoint will be assigned the last valid setpoint upon occurrence of the event. The last valid setpoint will remain unchanged.
   - Setpoint = safety setpoint: the setpoint will be assigned the safety setpoint upon the occurrence of the event. Thus, a secured emergency operation can be performed.
6. Set in the "setpoint ramp" the rise time which shall be met upon a setpoint change to reduce overshoot.
7. In the "Setpoint limits" section set both the high and a low limits for the setpoint.
8. Save the parameters entered with "Project > Save".

Selecting a signal source for actual values A, B. and C

To set the signal sources for the actual values of A, B and C, proceed as follows:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Control value" section select the signal source for each actual value. Here you can select whether you use no signal ("zero") as a source or one of the processed analog inputs.
4. Save the parameters entered with "Project > Save".

Registering the factors for actual values B and C and the offset value

Proceed as follows to set the signal sources for the actual values A, B and C:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the"Sum" section set the necessary weight factors for the actual values B and C.
4. Set the required "offset" value that should be added to the actual values.
5. Save the parameters entered with "Project > Save".

Implementing ratio or mixture controllers (S7-300, S7-400)

Selecting the controller type

Proceed as follows to implement a ratio or blending controller:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller> channel [No]".
3. In the "Encoder type" section select the "Ratio or Mixture controller".
4. Save the parameters entered with "Project > Save".

Processing setpoints

Proceed as follows to adjust the setpoint processing mode:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Setpoint source" section, select the source from which the setpoint is to be obtained.
4. In the "Safety setpoint" section, enter a value to be utilized upon CPU fault and start up as a safety value.

5. Select the behavior of the controller module for "Reaction to CPU errors" and "Start-up behavior". You may select from amongst the following options:
   - Setpoint = last valid setpoint: the setpoint will be assigned the last valid setpoint upon occurrence of the event. The last valid setpoint will remain unchanged.
   - Setpoint = safety setpoint: the setpoint will be assigned the safety setpoint upon the occurrence of the event. This enables secured emergency operation.
6. In the "setpoint ramp" section, set the rise time to be adhered to upon a setpoint change to reduce overshoot.
7. In the "Setpoint limits" section, set both an high and a low limit for the setpoint.
8. Save the parameters entered with "Project > Save".

Selecting the signal source for actual values A and D

To set the signal sources for the actual values of A and D, proceed as follows:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. Select the signal source for each actual value in the "Manipulated value" section. You can choose whether using a no signal ("zero") as a source or one the processed analog inputs. For the actual value for D, you may also select the controller control value.
4. Save the parameters entered with "Project > Save".

Registering Multipliers

Proceed as follows to set the signal sources for actual values A and D:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Multiply" section select the desired offset.
4. Save the parameters entered with "Project > Save".

For ratio controllers the actual value A is used as control variable while the actual value D stores the ratio magnitude. The setpoint input value is used as a ratio factor. Multiplying it by the actual value of D and adding the adjustable offset will yield the effective setpoint value. If actual value D is off only the offset should be added to the setpoint.

Setting the alarm generation (S7-300, S7-400)

Procedure

Proceed as follows to set interrupt generation:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".

3. In the "Alarm generation" section select the necessary options:
   - In the "Limit monitor for" section enter the signal for which the alarm should be set.
   - In "Alarm high limit", set the high limit beyond which the alarm should be triggered.
   - In "Warning high limit", set the value beyond which the "approaching high limit" warning should be issued.
   - In "Warning low limit", set the value below which the "approaching low limit" warning should be issued.
   - In "Alarm low limit", set the low limit below which the alarm should be triggered.
   - To avoid flickering of the monitor you may set the hysteresis parameter in "hysteresis".
4. Save the parameters entered with "Project > Save".

Control algorithms (S7-300, S7-400)

Implementing a P control (S7-300, S7-400)

Procedure

To implement a P control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "P" option.
   All configurable options for a P control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PI control (S7-300, S7-400)

Procedure

To implement a PI control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PI" option.
   All configurable options for a PI control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing an I control (S7-300, S7-400)

Procedure

To implement an I control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "I" option.
   All configurable options for an I control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PID control (S7-300, S7-400)

Procedure

To implement a PID control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PID" option.
   All configurable options for a PID control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PD control (S7-300, S7-400)

Procedure

To implement a PD control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PD" option.
   All configurable options for a PD control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a Fuzzy temperature controller (S7-300, S7-400)

Procedure

To implement a P control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "Fuzzy temperature controller module" option.
   All configurable options for a fuzzy temperature controller module will be activated and all other options disabled.
4. Save the parameters entered with "Project> Save".

Suppressing the deviation noise component (S7-300, S7-400)

Procedure

To suppress the noise component in the deviation, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "General" section, set the "Dead zone width" parameter.
4. Save the parameters entered with "Project> Save".

Switching disturbances (S7-300, S7-400)

Procedure

To switch a disturbance, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. From the "Disturbance source" list in the "General" section select the disturbance source.
   Select the entry "zero" to perform no disturbance switching.
4. Enter the "Disturbance factor" value you wish to use as a disturbance weight factor.
   A value <0 weakens the disturbance whereas a value> 0 increases the disturbance. A value of 1 keeps the disturbance unchanged.
5. Save the parameters entered with "Project> Save".

Setting the feedback P portion (S7-300, S7-400)

Procedure

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "P portion" section select the "In feedback" option.
   If the option is enabled, the P portion is set in feedback.
4. Save the parameters entered with "Project> Save".

Attenuating the D portion for setpoint changes (S7-300, S7-400)

Selecting the controller type

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "D portion" section enter the D portion source in the "D portion" option.
4. In the "Differentiation factor" section, set the D portion factor.
5. Save the parameters entered with "Project> Save".

Reversing the controller effect (S7-300, S7-400)

Principle

Controller reversal may be achieved by setting a negative proportional gain thus moving from "increasing deviation value = increasing correction value" to "increasing deviation value = decreasing correction value".

Procedure

To reverse the controller effect, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "General" section, set a proportional gain of opposite sign.
   As an example, if the current setting shows a positive proportional gain and you wish to reverse the controller effect, set a negative proportional gain.
4. Save the parameters entered with "Project> Save".

Controller output (S7-300, S7-400)

Implementing a continuous controller (S7-300, S7-400)

A continuous controller can be created only with an analog module.

Procedure

1. Changing an external control value (Page 1027)
2. Setting the safety control value (Page 1028)
3. Changing the tracking (Page 1027)
4. Setting control value limits (Page 1029)
5. Controlling two actuators - split range (Page 1029)
6. Selecting the output signal and the signal type (Page 1030)

Step controller without position feedback (S7-300, S7-400)

A step controller without position feedback can be implemented only with a digital module.

Procedure

1. Select controller output step controller without position feedback
2. Changing an external control value (Page 1027)
3. Setting the safety control value (Page 1028)
4. Using the pulse shaper step controller (Page 1029)

Implementing a step controller with position feedback (S7-300, S7-400)

A step controller with position feedback can be implemented only with a digital module.

Procedure

1. Select controller output step controller with position feedback
2. Changing an external control value (Page 1027)
3. Changing the tracking (Page 1027)
4. Setting the safety control value (Page 1028)
5. Setting control value limits (Page 1029)
6. Setting position feedback (Page 1028)
7. Using the pulse shaper step controller (Page 1029)

Implementing a pulse controller (S7-300, S7-400)

A pulse controller can be implemented only with a digital module.

Procedure

1. Select controller output pulse controller
2. Changing an external control value (Page 1027)
3. Changing the tracking (Page 1027)
4. Setting the safety control value (Page 1028)
5. Setting control value limits (Page 1029)
6. Prepare the analog-to-digital signal conversion
7. Converting analog into digital signal (Page 1030)

Parameterize the functions of the controller output (S7-300, S7-400)

Changing an external control value (S7-300, S7-400)

Procedure

For changing the external control variable, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "External control variable" select the option to activate the external control variable. You may choose from the following options:
   - Activation of the function module.
   - Activation of a function module or one of the digital inputs.
4. Save the parameters entered with "Project> Save".

Changing the tracking (S7-300, S7-400)

Procedure

To change the tracking proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Tracking" section select the option to activate the external control variable. You may choose from the following options:
   - Activation of the function module.
   - Activation of a function module or one of the digital inputs.

4. In the "Tracking" section select the source of the signal you wish to track. You may choose from the following options:
   - Zero, while no source is used.
   - One of the analog inputs.
5. Save the parameters entered with "Project> Save".

Setting position feedback (S7-300, S7-400)

Procedure

To set the position feedback, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to 'Digital Output> output [No.] ".
3. In the "Position feedback input" section select the "Source" option to choose the signal for position feedback.
   - Zero: no position feedback.
   - One of the processed analog inputs.
4. Save the parameters entered with "Project> Save".

Setting the safety control value (S7-300, S7-400)

Procedure

To set the safety control value, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "safety control value" section select the value that should be used as a safety control value.
4. Under "Activation" select the option to activate the external control variable. You may choose from the following options:
   - Activation of the function module.
   - Activation of a function module or one of the digital inputs.
5. Under "Action on module start" select the option which should be executed as the module starts. You may choose from the following options:
   - Control mode: sets the module immediately into control mode upon startup.
   - Safety manipulated value: initially, the safety manipulated value is controlled.
6. Under "Action on controller interference of the actual value A" select the option which should be executed as a controller interference of the actual value A occurs.

7. Under "Action on input controller interference" select the option which should be executed as an input controller interference occurs.
8. Save the parameters entered with "Project> Save".

Setting control value limits (S7-300, S7-400)

Procedure

To set control value limits you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Output signal limit" enter the following values:
   - High value: illustrates the high limit for the output signal.
   - Low value: illustrates the low limit for the output signal.
4. Save the parameters entered with "Project> Save".

Controlling two actuators - split range (S7-300, S7-400)

Procedure

To control two actuators you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Split range" section select the option "Enable" to effectively control two actuators.
4. Set the following values for both the input and output signals A and for the input and output signals B:
   - Low input signal.
   - High input signal.
   - Low output signal.
   - High output signal.
5. Save the parameters entered with "Project> Save".

Using the pulse shaper step controller (S7-300, S7-400)

Procedure

To configure the pulse shaper you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to 'Digital Output> output [No.] ".

3. In the "Pulse shaper" section, enter the following parameters:
   - Engine operating time.
   - Minimum pulse duration.
   - Minimum break duration.
4. Select the best acceptable option for the high and the low values:
   - From the function module.
   - From the function module or one of the digital inputs.
5. Save the parameters entered with "Project> Save".

Converting analog into digital signal (S7-300, S7-400)

Procedure

To convert an analog signal into a digital signal you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate to the "Digital output> channel [No.]" in the design view.
3. In the "Pulse shaper" area, enter the following parameters:
   - Minimum pulse duration.
   - Minimum break duration.
4. Save the parameters entered with "Project> Save".

Selecting the output signal and the signal type (S7-300, S7-400)

Procedure

To select the output signal and the signal type you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate to the "Digital output> channel [No]" in the design view.
3. Under the option "Source" select the signal you wish to use as output signal. You may choose among the following options:
   - Zero: no source for the output signal.
   - One of the analog inputs.
   - One of the control values for controller A or B.
4. Under the option "Signal type" select the type for the output signal you wish to use.
5. Save the parameters entered with "Project> Save".

3.5.6.3 Comissioning FM 355-2 (S7-300, S7-400)

Creating and supplying instance DB (S7-300, S7-400)

Creating and initializing an instance DB

1. Create a user program that calls the FMT_PID instruction cyclically.
2. Set READ_OUT = TRUE cyclically in the user program.
3. Create an instance DB of the FMT_PID instruction for each controller channel.
4. Enter the module address for each instance DB in the MOD_ADDR parameter.
   Use the start address that you specified when you configured your hardware as the module address of the FM 355-2.
5. Enter the channel number of the controller channel (0, 1, 2 or 3) in the FMT_PID.CHANNEL parameter.

1. If you use the FMT_PAR and FMT_TUN instructions, enter the channel number of the controller channel in the FMT_PAR.CHANNEL and FMT_TUN.CHANNEL parameters.
2. Check to ensure that the "Sensor type analog input" of this channel is selected.
   If the analog input is not processed, all PID parameters are set to zero and a manipulated variable is not calculated.

Note

Multi-instance DBs

If you use multi-instance DBs in your project, you must commission the FM via a watch table.

Commissioning the controller (S7-300, S7-400)

Introduction

You tune the PID parameters as follows:

- Pretuning
- Fine tuning heating
- Fine tuning cooling
- Manual fine tuning

Procedure

Proceed as follows to tune a controller:

1. Put the CPU in RUN mode.
2. If using a step controller, measure the motor actuating time.

3. Perform controller tuning.
4. Monitor the control loop with the trend view.

Options of controller tuning (S7-300, S7-400)

Pretuning

During this tuning, the working point is approached from the cold state through a setpoint jump.

With TUN_ON = TRUE, you can establish the tuning readiness. The controller switches from PHASE = 0 to PHASE = 1.

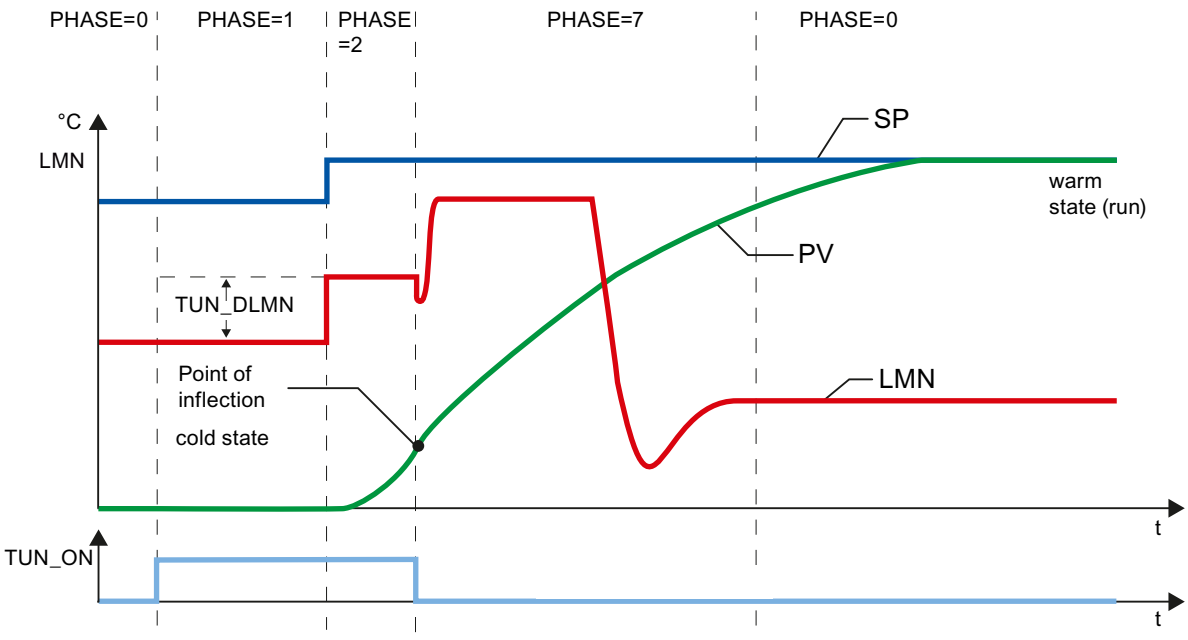

The tuning manipulated variable (LMN0 + TUN_DLMN) is activated by a setpoint change (transition PHASE 1 -> 2). The setpoint is not effective until the inflection point has been reached (automatic mode is not enabled until this point is reached).

The user is responsible for defining the output excitation delta (TUN_DLMN) according to the permitted process value change. The sign of TUN_DLMN must be set depending on the intended process value change (take into account the direction in which the control is operating).

The setpoint step change and TUN_DLMN must be suitably matched. If the value of TUN_DLMN is too high, there is a risk that the point of inflection will not be found before 75% of the setpoint step change is reached.

TUN_DLMN must still be high enough to ensure that the process value reaches at least 22% of the setpoint step change. Otherwise, the process will remain in tuning mode (phase 2).

Solution: Reduce the setpoint value during the inflection point search.

Note

If processes are extremely sluggish, it is advisable to specify a target setpoint that is somewhat lower than the desired operating point during tuning and to monitor the status bits and PV closely (risk of overshooting).

Tuning only in the linear range:

The signals of certain processes (e.g., zinc or magnesium smelters) will pass a non-linear area at the approach of the operating range (change in the state of aggregation).

By selecting a suitable setpoint step change, tuning can be limited to the linear range. When the process value has passed 75% of the setpoint step change (SP_INT-PV0), tuning is ended.

At the same time, TUN_DLMN should be reduced to the extent that the point of inflection is guaranteed to be found before 75% of the setpoint step change is reached.

Fine tuning heating

During this tuning, the process with a constant setpoint is activated through a output value jump.

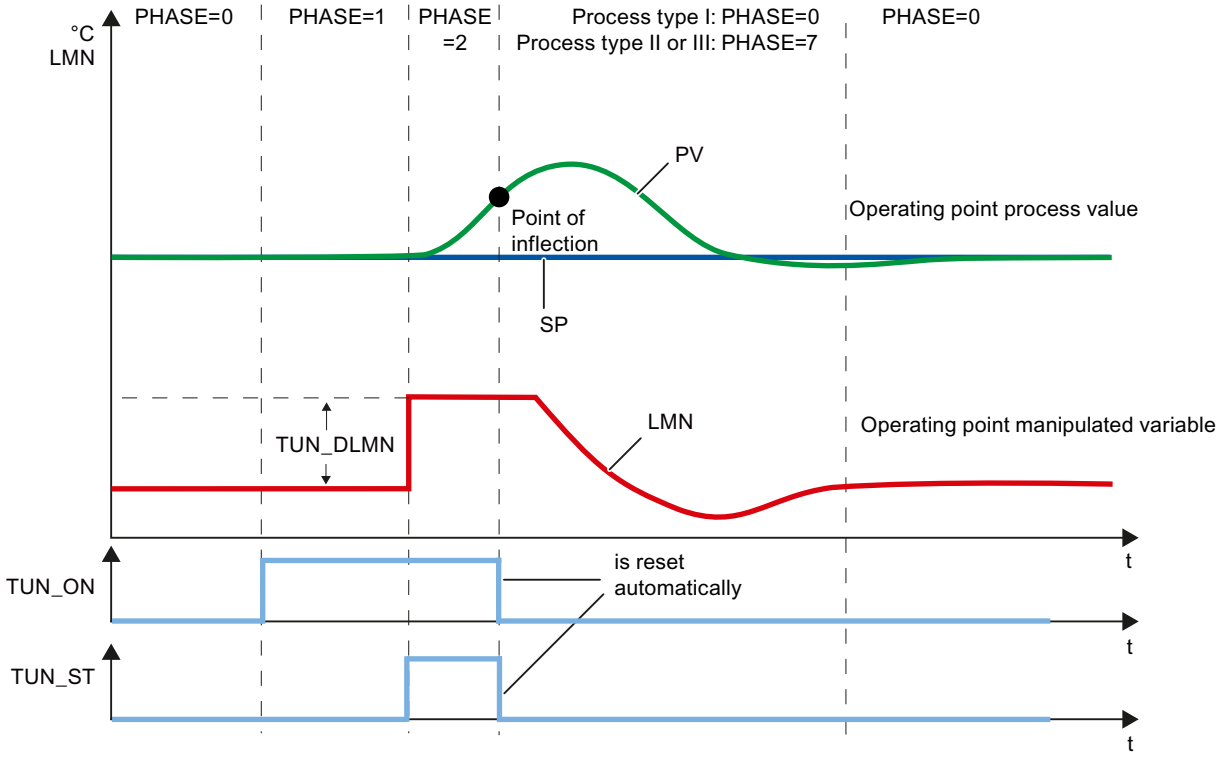

The tuning manipulated variable (LMN0 + TUN_DLMN) is activated by setting the start bit TUN_ST (transition from phase 1 -> 2). When you modify the setpoint value, the new value will not take effect until the point of inflection has been reached (automatic mode will not be enabled until this point has been reached).

The user is responsible for defining the output excitation delta (TUN_DLMN) according to the permitted process value change. The sign of TUN_DLMN must be set depending on the intended process value change (take into account the direction in which the control is operating).

NOTICE
Safety off at 75% is not available when you excite the process via TUN_ST. Tuning is ended when the point of inflection is reached. However, in noisy processes the point of inflection may be significantly exceeded.

Fine tuning cooling

During this tuning, the process with a constant setpoint is activated through a output value jump.

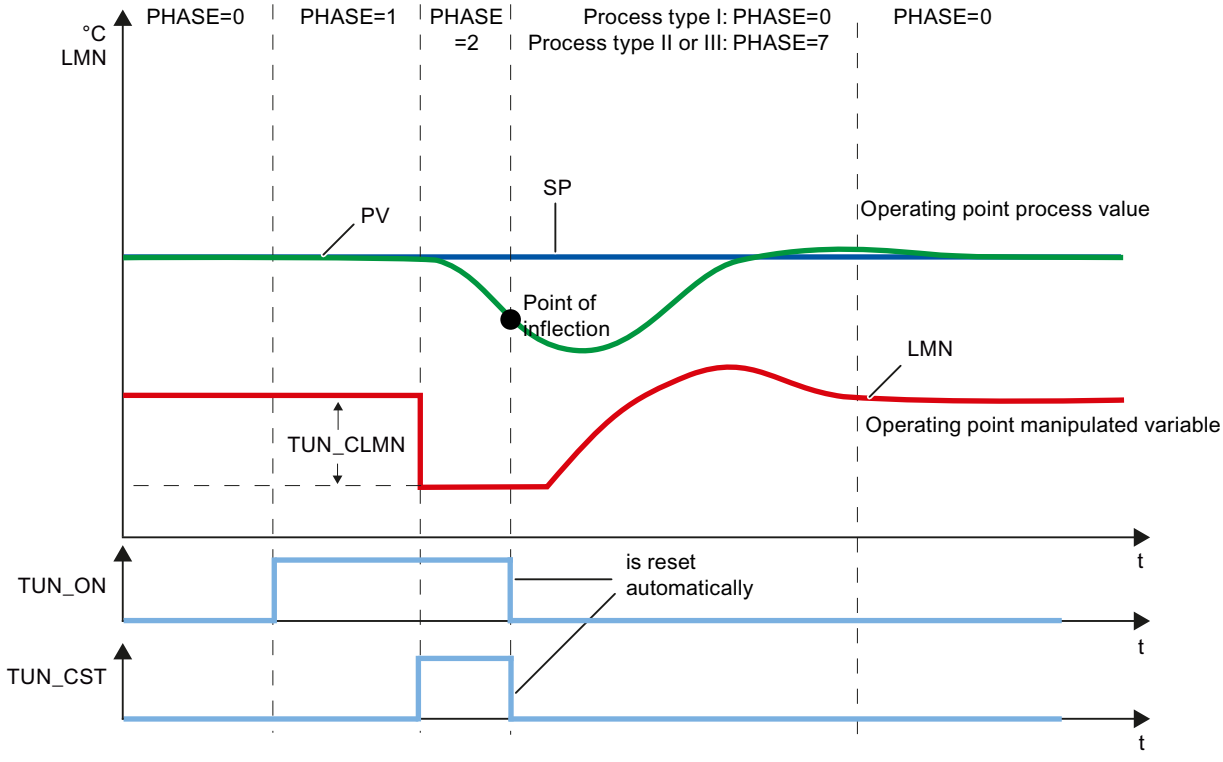

The tuning manipulated variable (LMN0 + TUN_CLMN) is activated by setting the start bit TUN_CST (transition from phase 1 -> 2). The previously determined manipulated value LMN0 is kept constant during this process and TUN_CLMN is activated directly in the split-range function so that only the cooling power is changed.TUN_CST is reset immediately.

When you modify the setpoint during phase 2, the new setpoint will not take effect until the point of inflection has been reached (automatic mode will not be enabled until this point has been reached).

The user is responsible for defining the output excitation delta (TUN_CLMN) according to the permitted process value change. The sign of TUN_CLMN must be set depending on the intended process value change (take into account the direction in which the control is operating). Note that a negative TUN_CLMN causes an increase in cooling power. But you can also use TUN_CLMN > 0.0 to activate a setpoint jump by reducing the cooling power.

NOTICE
Safety off at 75% is not available when you excite the process via TUN_CST. Tuning is ended when the point of inflection is reached. However, in noisy processes the point of inflection may be significantly exceeded.

Split-range function

During optimization, the gradient of the split-range function is added to the process. If you want to modify the gradient of the split-range function after optimization, you must adapt the controller's GAIN or RATIOFAC parameter accordingly.

Manual fine-tuning in control mode

The following measures can be employed to achieve an overshoot-free setpoint response:

- Adapt control zone
- Optimize command action
- Attenuation of control parameters
- Control parameter modification

Automatic tuning (S7-300, S7-400)

Process requirements (S7-300, S7-400)

Transient response

The process must have a stable, asymptotic transient response with time lag.

The actual value must settle to steady state after a jump of the manipulated variable. This therefore excludes processes that already show an oscillating response without control, as well as processes with no compensation (integrator in the control system).

Guaranteeing a stationary initial state (phase 0)

With lower-frequency oscillations of the actual value, for example, due to incorrect controller parameters, the controller must be put in manual mode before the optimization is started and wait for the oscillation to stop. You can also switch to a PI controller with small proportional gain and high integration time.

Now you have to wait until the stationary state is reached, this means, until the process value and output value have a steady state. It is also permissible to have an asymptotic transient oscillation or slow drifting of the process value (stationary state, see the following image). The output value must be constant or fluctuate by a constant average.

Note

Avoid changing the manipulated variable shortly before starting the tuning. A change of the manipulated variable can occur in an unintended manner through the establishment of the test conditions (for example, closing an oven door)! If this does happen, you have to at least wait until the process value has an asymptotic transient oscillation in a stationary state again. Better controller parameters can be reached if you wait until the transient effect has completely subsided.

In the following image, the transient oscillation is illustrated in the stationary state:

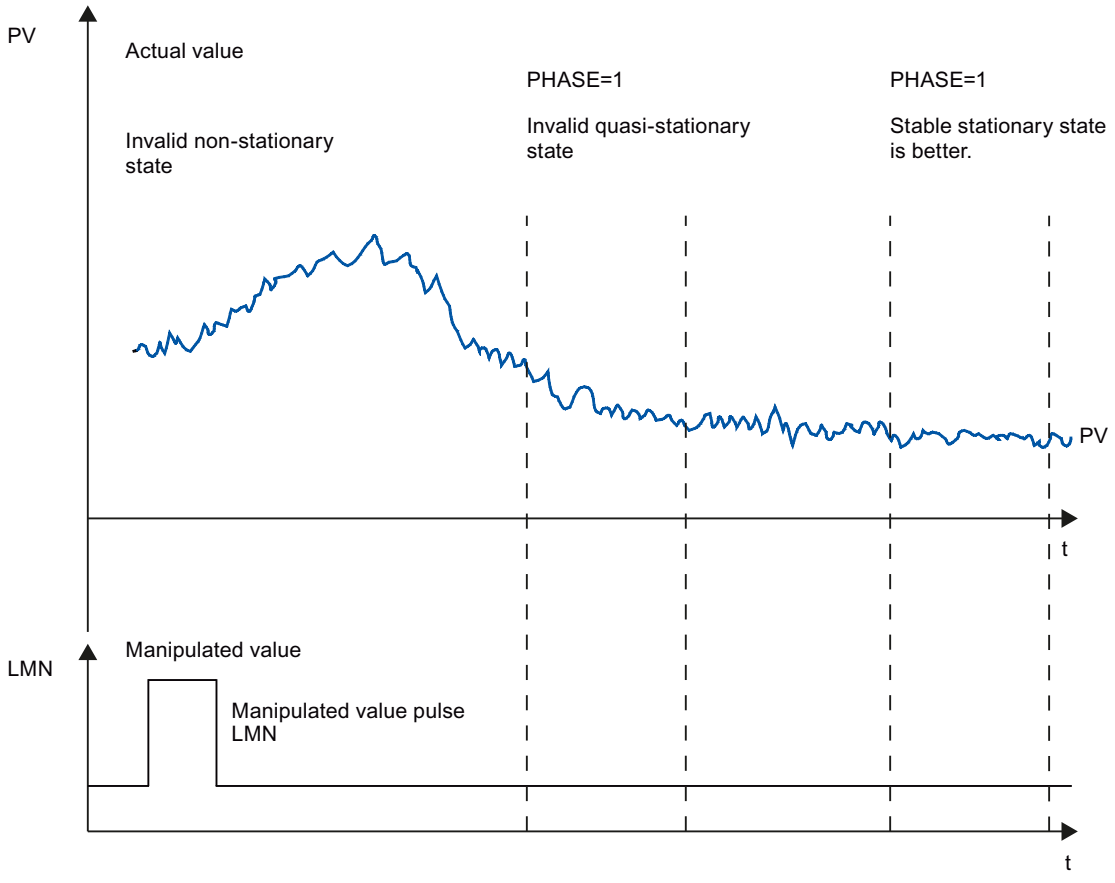

Linearity and operating range

The process response must be linear across the operating range. Non-linear response occurs, for example, when a state of aggregation changes. Optimization must take place in a linear part of the operating range.

This means, during optimization and normal control operation non-linear effects within the operating range must be insignificant. It is, however possible to retune the process when the operating point changes, providing optimization is repeated in the close vicinity of the new working point and non-linearity does not occur during the optimization.

If a specific static non-linearity (e.g. valve characteristics) is known, it is always advisable to compensate this with a polyline to linearize the process response.

Disturbance in temperature processes

Disturbances such as the transfer of heat to neighboring zones must not affect the overall temperature process too much. For example, when optimizing the zones of an extruder, all zones must be heated simultaneously.

Process of controller tuning (S7-300, S7-400)

PI/PID controller parameters

The FM 355-2 auto-tuning feature automatically sets the PI/PID controller parameters. You can tune the heating and cooling processes as well as split-range processes with two counteracting final controlling elements (e.g. the final controlling elements for heating and cooling processes).

Phases of controller tuning

For the controller tuning, individual phases are run through, which you can read on the parameter PHASE .

PHASE = 0

Tuning is not running. The FM 355-2 works in automatic or manual mode.

During PHASE = 0, you can make sure that the controlled system fulfills the requirements for tuning.

At the end of tuning, FM 355-2 switches back into PHASE = 0.

PHASE = 1

FM 355-2 is ready for tuning. PHASE = 1 may only be started if the requirements for tuning are fulfilled.

During PHASE = 1, the following values are determined:

- Process value noise NOISE_PV
- Initial slope PVDT0
- Average of the manipulated variable

PHASE = 2

In phase 2, the process value attempts to detect the point of inflection with a constant manipulated variable. This method prevents the point of inflection from being found too early as a result of process variable noise.

Initially, this averaging is inactive; in other words, averaging always takes place over one cycle. As long as the noise exceeds a certain level, the number of cycles is doubled.

The period and amplitude of the noise are calculated. The search for the point of inflection is canceled and phase 2 is exited only when the gradient is always smaller than the maximum rise during the estimated period. TU and T_P_INF are, however, calculated at the actual point of inflection.

Tuning, however, is only ended when the following two conditions are met:

1. The process value is more than 2*NOISE_PV away from the point of inflection.
2. The process value has exceeded the point of inflection by 20%.

Note

When exciting the process using a setpoint step change, tuning is ended at the latest when the process value exceeds 75% of the setpoint step change (SP_INT-PV0) (see below).

PHASE = 3, 4, 5, 6

The phases 3, 4, 5 and 6 last 1 cycle each.

In Phase 3, the valid PI/PID parameters are saved before the optimization and the process parameter is calculated.

In Phase 4, the new PI/PID parameters are calculated.

In phases 5 and 6, the new manipulated variable is calculated and the controlled system is given.

If a controlled system of type I was identified, tuning mode stops and tuning is once again in phase 0. The controller now starts with LMN = LMN0 + 0.75*TUN_DLMN always in automatic mode (even if you controlled in manual mode before you started tuning).

If a controlled system of type II or III was identified, tuning changes to phase 7.

PHASE = 7

There is a risk with typical temperature systems (process type I) that the point of inflection is detected too early as a result of noise. The lower point of inflection time T_P_INF may possibly result in the detection of a process type II or III.

There is a check in phase 7 to see if the process type is correct. This testing occurs in the automatic mode with the recently recalculated controller parameters and ends at the latest 6*TA (equilibrium time) after the point of inflection. If process type I is detected, the controller parameters are recalculated (STATUS_D = 122) otherwise they remain the same.

The checking of the process type is cancelled, if the process value during tuning at the operating point reaches the process value once again prior to tuning PV0.

If phase 7 is canceled with TUN_ON = FALSE, the already determined controller parameters are maintained!

Note

It is generally possible to start the heat tuning during cooling. The order check in phase 7 will not take place in this case.

The reverse case (starting cooler tuning during heating) is not critical, because a cooler tuning never runs through phase 7!

Premature cancellation of the optimization

In Phase 1, 2 or 3, you can cancel the optimization by resetting TUN_ON = FALSE without calculating new parameters. The controller starts in automatic mode with LMN = LMN0. If the controller was in manual mode before the tuning, the old manual manipulated variable is output.

If tuning is canceled in Phase 4, 5, 6 or 7 with TUN_ON = FALSE, the controller parameters determined until then are maintained.

Tuning result (S7-300, S7-400)

Optimization result

The left number of STATUS_H and STATUS_C displays the tuning state.

STATUS_H STATUS_C	Result
0	Default or new controller parameters have not yet been found.
10000	Suitable controller parameters were found
2xxxx	Control parameters have been found via estimated values; check the control response or check the STATUS_H/C diagnostic message and repeat controller optimization.
3xxxx	An operator input error has occurred; check the STATUS_H/C diagnostic message and repeat controller optimization.

After the recognition of the inflection point in phase 6, the following controller parameters are updated on the FM 355-2 and at the online instance DB of instruction FMT_PID:

- Factor for the attenuation of the proportional component PFAC_SP = 0.8
- Controller GAIN
- Integration time TI (limited to ≥0.5 s)
- Derivative time TD (limited to ≥1.0 s)
- Derivative factor D_F = 5.0
- Control zone on/off CONZ_ON=TRUE/FALSE
- Control zone width CON_ZONE=250/GAIN
- P_SEL = TRUE (even if it was previously a controller with integral action only)

If RATIOFAC<>0.0, then CON_ZONE is multiplied by the factor 1.5.

If a value TD<1.0 s is calculated in the optimization process, only a PI controller is determined and PID_ON as well as the PID parameters will be set to zero.

The control zone is only enabled for matching process types (process type I and II) and PID controllers (CONZ_ON = TRUE).

Depending on PID_ON, control is implemented either with a PI or a PID controller. The old controller parameters are saved and can be retrieved with UNDO_PAR. A PI and a PID parameter record is saved additionally. Using LOAD_PID and making a suitable setting for PID_ON, it is also possible to switch later between the tuned PI or PID parameters.

A previously split structure (derivative component in the feedback path) will be maintained.

Note

Verify correct operation of your controller parameters immediately after the controller optimization process is completed.

Split-range function

During optimization, the gradient of the split-range function is added to the process. If you want to modify the gradient of the split-range function after optimization, you must adapt the controller's GAIN parameter accordingly.

Fault descriptions and corrective measures (S7-300, S7-400)

Compensating operator errors

Tuning is canceled in case of all errors listed in this table.

Errors	STATUS and actions	Comment
TUN_ON and TUN_ST or TUN_CST are set simultaneously	STATUS_H/C = 30001 • Transition to phase 0 • TUN_ON = FALSE	Cannot occur with assistant, because TUN_ST and TUN_CST are set to FALSE in the first template.
Sign of setpoint change does not match process value change.	End tuning at the point of inflection Safety off at 75% ineffective	

Errors	STATUS and actions	Comment
|TUN_DLMN| or |Effective TUN_DLMN| < 5 % (end of phase 1) Special case STATUS_H=30004: Effective manipulated value difference limited by split–range limits and not by manipulated value limits	STATUS_H/C = 30002/30004 • Transition in phase 99 (Phase 0 in a parallel optimization process) • Output LMN = LMN0	Either you configured a value for TUN_DLMN that is too low, or the manipulated variable was near to a control limit prior to start of tuning. If STATUS_H=30004: Also take into account that, for example, with LMN_A < 5% a heating tuning with negative TUN_DLMN is not possible (reason: the cooling power must not be adjusted). In this case, you must prevent the controller from settling to the new setpoint and from unnecessary leaving the stationary operating point (not possible with parallel tuning process). You should now proceed as follows: • Withdrawing setpoint • TUN_ON=FALSE • Correcting TUN_DLMN • Restart tuning
Only for step controllers without analog position feedback: |Effective TUN_DLMN| < 5 % (start of phase 2)	STATUS_H = 30002 • Transition to phase 0 • TUN_ON = FALSE	Error occurs if |TUN_DLMN| >= 5 % but manipulated value was near to a control limit prior to start of tuning.
If TUN_CST: |TUN_CLMN| or |Effective TUN_CLMN| < 5 % (end of phase 1) Special case STATUS_H=30004: Effective manipulated value difference limited by split–range limits and not by manipulated value limits.	STATUS_C = 30002/30004 • Transition to phase 0 • TUN_ON = FALSE	Either you configured a value for TUN_CLMN that is too small or the manipulated variable was near to a control limit prior to start of tuning.
If TUN_CST: Special case of "|Effective TUN_CLMN| < 5 % (end of phase 1)": TUN_CLMN <= -5%, but LMN_LLM > -5%.	STATUS_C = 30003 • Transition to phase 0 • TUN_ON = FALSE	The low limit LMN_LLM is too high (e.g. 0%) and therefore prevents the output of cooling power.
TUN_CST but without previous heating tuning	STATUS_C = 30008 • Transition to phase 0 • TUN_ON = FALSE	
Safety mode	STATUS_C = 30009 • Transition to phase 0 • TUN_ON = FALSE	If FM 355-2 enters safety mode in PHASE > 2, it is displayed by STATUS_C=30009.

Point of inflection not reached (only if excited by setpoint step change)

At the latest, tuning is ended when the process value has passed 75% of the setpoint step change (SP_INT-PV0). This is signaled by "inflection point not reached" in STATUS_H/ STATUS_C (2xx2x).

The currently valid setpoint always applies. By reducing the setpoint, it is possible to achieve an earlier end of the tuning function.

In typical temperature processes, cancelation of tuning at 75% of the setpoint step change is normally adequate to prevent overshoot. However, **caution** is advised, particularly in processes with a greater delay (TU/TA > 0.1, process type III). If manipulated variable excitation is too strong compared to the setpoint step change, the process value can overshoot heavily (up to a factor of 3).

In higher-order processes, if the point of inflection is still a long way off after reaching 75% of the setpoint step change, there will be significant overshoot. In addition, the controller parameters are too stringent. In this case, you should reduce the controller parameters or repeat the attempt.

The following schematic illustrates the overshoot of the process variable when the excitation is too strong (process type III):

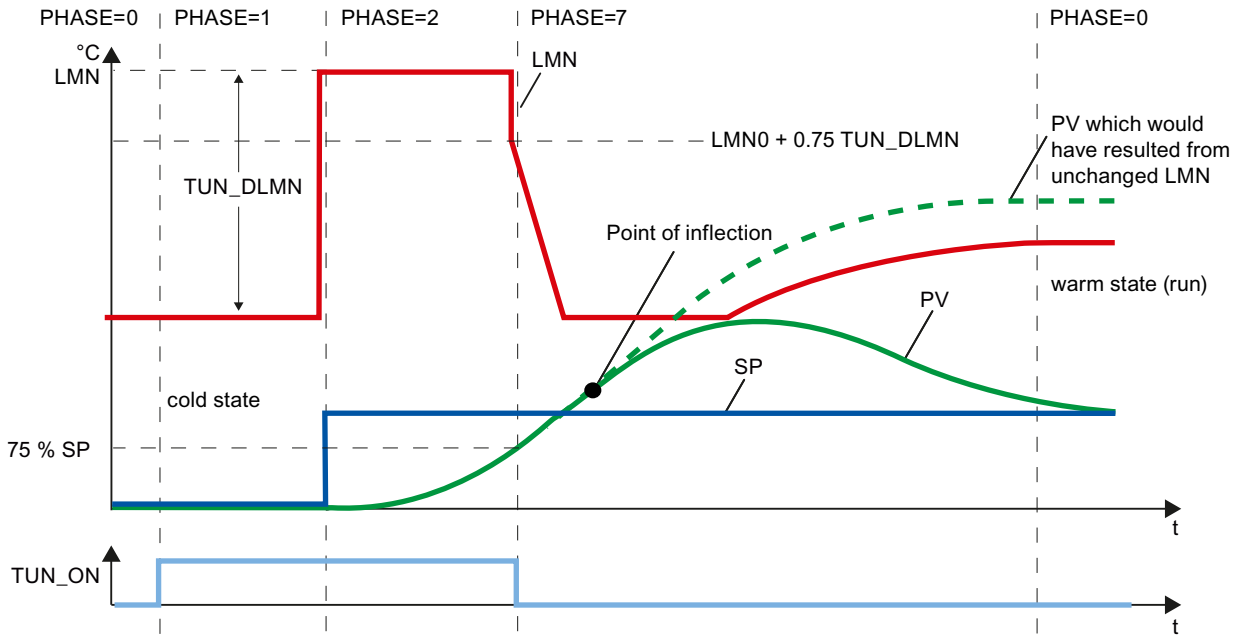

In typical temperature processes, cancelation shortly before reaching the point of inflection is not critical in terms of the controller parameters.

If you repeat the attempt, reduce TUN_DLMN or TUN_CLMN or increase the setpoint step change.

Principle: The value of the manipulated variable used for tuning must be suitable for the setpoint step change.

Error estimating the delay time or order

The delay time (STATUS_H/STATUS_C = 2x1xx, 2x2xx or 2x3xx) or order (STATUS_H/STATUS_C = 21xxx or 22xxx) were not acquired correctly. Operation continues with an estimate that can lead to non-optimum controller parameters.

Repeat the tuning procedure and ensure that disturbances do not occur at the process value.

Note

The special case of a PT1-only process is also indicated by STATUS_H/STATUS_C = 2x2xx (TU < CYCLE). In this case, it is not necessary to repeat the attempt. Reduce the controller parameters if the control oscillates.

Quality of measuring signals (measurement noise, low-frequency interference)

The results of tuning can be distorted by measurement noise or by low-frequency interference. Note the following:

- During one noise period, the process value should be sampled at least twice. The degree of noise should not exceed 5% of the useful signal change.
- High-frequency interference cannot be filtered out by a software block. This should be filtered earlier in the measuring sensor to prevent the aliasing effect.
  The following schematic illustrates the aliasing effect when the sampling time is too long:

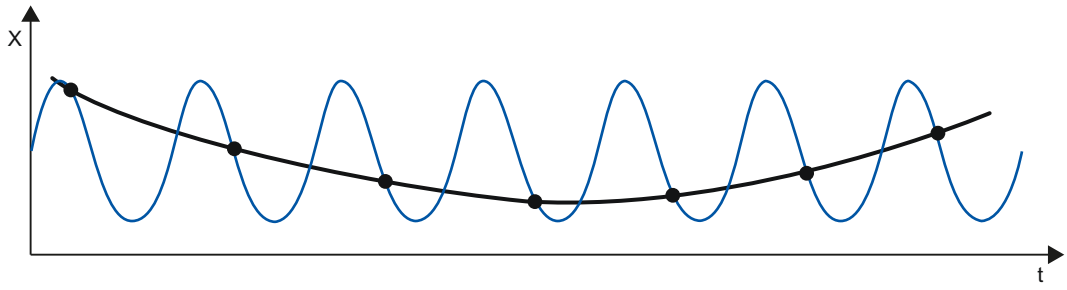

- In case of low-frequency interferences, we assume that the sampling time of the FM 355-2 is small enough. However, the FM 355-2 must then generate a uniform measuring signal by having a large interval in the mean value filtering. Mean value filtering must extend over at least two noise periods. Internally in the block, this soon results in higher sampling times such that the accuracy of the tuning is adversely affected. Adequate accuracy is guaranteed with at least 40 noise periods up to the point of inflection. Possible remedy when repeating the attempt: Increase TUN_DLMN/TUN_CLMN.

Overshoot

Overshoot can occur in the following situations:

Situation	Cause	Remedy
End of tuning	• Excitation by a too high manipulated value change compared with the setpoint step change (see above). • PI controller activated by PID_ON = FALSE.	• Increase the setpoint step change or reduce the manipulated value step change. • If the process permits a PID controller, start tuning with PID_ON = TRUE.
Tuning in phase 7	Initially, less aggressive controller parameters were determined (process type III); these can lead to an overshoot in phase 7.	-
Control mode	PI controller with PFAC_SP = 1.0 for process type I.	If the process permits a PID controller, start tuning with PID_ON = TRUE.

Tuning with step controller (S7-300, S7-400)

Introduction

The general information on controller optimization applies.

Peculiarities of step controllers

- Step controllers on the FM 355-2 operate without control zone.
- Phase 7 is not executed.
- No cooling optimization.

Note

The motor transition time is not determined by tuning and must be measured or set before you start tuning.

Controller design

The motor actuating time MTR_TM should be as small as possible compared to the inflection point time T_P_INF and the equivalent dead time TU.

Controllers with a softer action are automatically generated for longer motor actuating times.

The higher the process excitation TUN_DLMN, the higher is the influence of the motor actuating time on the controller design.

PI or PID parameters

PI parameters are attenuated by 25% compared to a controller designed for closed-loop controllers and pulse controllers. PID parameters are also determined (but without the 25% safety margin). The PID parameters should only be used if the motor actuating time is not too high compared to the process parameters and when the load on the final controlling element stays within the hysteresis due to the derivative component.

Step controller without position feedback

The output of Open/Close instructions is stopped right at the start of phase 1 (quasi manual mode).

The mean value of the manipulated variable in phase 1 (LMN0) will not be calculated.

At the start of phase 2 and during the time MTR_TM * TUN_DLMN / 100, an Open instruction (or with negative TUN_DLMN, a close instruction) is output.

When an end stop signal is triggered during the pulse action in phase 2, the effective TUN_DLMN is calculated as: 100 * time / MTR_TM.

When the value of the effective TUN_DLMN < 5%, the error message STATUS_H=30002 is output and optimization is canceled. This error can only occur if the final controlling element is unexpectedly close to a limit.

End or cancellation of optimization

The controller starts with LMN = LMN0 + TUN_DLMN.

Performing pretuning (S7-300, S7-400)

Requirement

- The requirements on the process are met.
- An instance DB has been created for the channel and is supplied.
- "Instruction" is configured as source for the setpoint.
- The FM 355-2 is not in safety mode (SAFE_ON = FALSE). An ongoing tuning is cancelled when safety mode is switched on.

Procedure

WARNING

This may result in death, severe injury or considerable property damage.

During an tuning, the parameter LMN_REON is ineffective. Even tracking circuits that are derived from alarm limits, for example, do not have an effect. Through this, the manipulated value or process value may take on undesired - even extreme - values.

The manipulated value is defined through the tuning. To cancel the tuning, you first have to set TUN_ON = FALSE. This will make LMN_REON effective once again.

To manually determine the optimum PID parameters for initial commissioning, follow these steps:

1. Select the instance DB of the channel you want to tune.
2. Click the "Measurement on" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
3. Select "Pretuning" from the "Mode" drop-down list.
   The FM 355-2 is ready for tuning (PHASE = 1).
4. Wait until the steady state has been reached.
5. In the "Output value jump" field, specify how much the output value should be increased.
6. Enter a setpoint in the "Setpoint" field. The output value jump only takes effect when another setpoint is entered.
7. Click the "Start tuning" icon.
   The pretuning starts. The status of the tuning is displayed.

   Note

   During phase 1 a tuning process can also be triggered by minor setpoint changes (e.g. measurement noise at an analog input). In this case, tuning is terminated very quickly (wrong controller parameters; risk of instability).

Performing fine tuning for heating (S7-300, S7-400)

Requirement

- The requirements on the process are met.
- An instance DB has been created for the channel and is supplied.
- "Instruction" is configured as source for the setpoint.
- The FM 355-2 is not in safety mode (SAFE_ON = FALSE). An ongoing tuning is cancelled when safety mode is switched on.

Procedure

To determine the optimal PID parameters at the operating point, follow these steps:

1. Select the instance DB of the channel you want to tune.
2. Click the "Measurement on" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
3. Select "Fine tuning heating" from the "Mode" drop-down list.
   The FM 355-2 is ready for tuning (PHASE = 1).
4. In the "Output value jump" field, specify how much the output value should be increased.
5. Click the "Start tuning" icon.
   Fine tuning starts. The status of the tuning is displayed.

Performing cooling fine tuning (S7-300, S7-400)

Introduction

For controllers operating with two counteracting final actuators (heating and cooling actuator), FM 355-2 determines the process gain ratio RATIOFAC (heating/cooling gain) after a manipulated variable jump using the cooling actuator. Even the control zone width CON_ZONE is recalculated. The other controller parameters remain the same.

Requirement

- "Instruction" is configured as source for the setpoint.
- The FM 355-2 is not in safety mode (SAFE_ON = FALSE). An ongoing tuning is cancelled when safety mode is switched on.
- A heating fine tuning of the channel has been successfully completed. Heating tuning has to be repeated after a voltage failure of the FM 355-2.
- In a plant with several thermally interconnected temperature zones (e.g. plastics machines) the heating fine tuning has been successfully completed for all zones and the steady state has been reached.
- Use manipulated variable A for heating and manipulated variable B for cooling.
- RATIOFAC is effective at LMN<0.0, which means the split-range function must be defined as follows: A for LMN>=0 .0 and B for LMN<0.0.
- To start cooling fine tuning during heating, the heating and cooling signals have to become effective in the process at the same time.

Procedure

To determine the optimal PID parameters at the operating point, follow these steps:

1. Select the instance DB of the channel you want to tune.
2. Click the "Measurement on" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
3. Select "Fine tuning cooling" from the "Mode" drop-down list.
   The FM 355-2 is ready for tuning (PHASE = 1).
4. In the "Manipulated variable jump" field, specify how much the manipulated variable should be increased or decreased.
5. Click the "Start tuning" icon.
   Fine tuning starts. The status of the tuning is displayed.

Result

As soon as a point of inflection is detected in the process value curve, the FM 355-2 switches back to control mode (PHASE 0).

Contrary to heat tuning, the old controller parameters are not saved (the old RATIOFAC was already backed up during heat tuning) and phase 4 will not run through.

STATUS_D remains the same and still refers to heat tuning. Even the PI and PID parameter sets of the last heat tuning will be maintained so that you can still load them after cooling tuning with LOAD_PID.

Parallel tuning of controller channels (S7-300, S7-400)

Neighboring zones with strong temperature coupling effect

When two or multiple controllers of a module are used to control the temperature, for example, on a plate (e.g. two heaters and two measured actual values with strong temperature coupling), you have the option of parallel controller tuning.

Procedure

Follow these steps to tune several channels at once:

1. Configure the channel group in Zone A or Zone B.
2. Start tuning in one of these channels. Tuning of the other channels will then be started automatically.

Note

The user must ensure that the setpoint jumps are set concurrently when the process is excited via setpoint jump. You can use the parameters PID_ON and TUN_DLMN/TUN_CLMN for one channel only. You must specify the parameters for the other channels in the corresponding instance DBs of the FMT_PID instruction before you start tuning.

Tuning is aborted

Tuning of all channels is aborted if an error occurs in one of the channels (Phase 0 or end of phase 1).

The user can reset tuning by setting TUN_ON=FALSE at any of the participating channels.

When a channel goes into safety mode, tuning of all other participating channels is also switched off.

Only the status information of the selected channel is displayed. For all other channels, the causes of error are found in the corresponding instance DBs.

Advantage

Each participating controller will output its tuning manipulated variable until all controllers have exited phase 2. This avoids that the controller that completes tuning first corrupts the tuning result of the other controller due to the change in its manipulated variable.

⚠ CAUTION
When 75% of the setpoint jump is reached at one of the participating channels, tuning is not terminated (risk of overshoot). Automatic mode is only started after all participating controllers have terminated tuning.

Neighboring zones with weak temperature coupling effect

As a general rule, tuning should be carried out to reflect the way in which the control will work later. When zones are operated in parallel during production so that the temperature differences between the zones remain the same, the temperature level of the neighboring zones ought to be increased accordingly during tuning.

Temperature differences at the beginning of a trial are irrelevant since they will be compensated in the initial heating process (→initial rise = 0).

Manual fine tuning in control mode (S7-300, S7-400)

Introduction

The following measures can be employed in order to achieve a non-overshooting response to setpoint changes:

Adapt control zone

During tuning, the FM 355-2 determines the control zone width CON_ZONE and, with an appropriate process type (process type I and II), a PID controller is activated: CONZ_ON = TRUE. During control mode, you can modify the control zone or switch it off completely (with CONZ_ON = FALSE).

No control zone for process type III, PI controller, step controller

Activating the control zone with higher order processes (process type III) does not normally bring any benefit since the control zone is then larger than the control range that can be achieved with a 100% manipulated variable. There is also no advantage in activating the control zone for PI controllers.

Note

Before you switch on the control zone manually, make sure that the control zone width is not too small. This means, the manipulated variable and actual value will oscillate if the control zone width is too small.

Weakening control response continuously with PFAC_SP

The control zone offers the best dynamic option of achieving a control response that is free of overshoot. Use it wherever possible, so for PID controllers for process types I and II.

For PI controllers, process type III or setpoint jumps within the control zone you can weaken the control response with parameter PFAC_SP. This parameter specifies the amount of P action that is affective for setpoint jumps.

Regardless of the process type, PFAC_SP is set to a default value of 0.8 by the tuning function; you can later modify this value if required. To limit overshoot during setpoint jumps (with otherwise correct controller parameters) to approximately 2%, the following values are adequate for PFAC_SP:

	Process type I	Process type II	Process type III
	Typical temperature process	Intermediate range	Higher order temperature process
PI	0.8	0.82	0.8
PID	0.6	0.75	0.96

Adapt the default factor (0.8) especially under the following situations:

- Process type I with PID (0.8 →0.6): With PFAC_SP = 0.8, setpoint jumps will still lead to approx. 18% overshoot.
- Process type III with PID (0.8 →0.96): With PFAC_SP = 0.8, the setpoint jumps is attenuated too heavily. This leads to a significant loss of tuning time.

Attenuation of control parameters

When a closed-loop control circuit oscillates or if overshoot occurs after setpoint jumps, you can reduce the controller's GAIN (e.g. to 80% of the original value) and increase the integration time TI (e.g. to 150% of the original value). If the analog manipulated variable of the closed-loop controller is converted to binary actuating signals by a pulse shaper, quantization noise may cause minor permanent oscillation. You can eliminate this effect by increasing the dead band width DEADB_W.

Tuning PID parameters manually (S7-300, S7-400)

Requirement

- The requirements on the process are met.
- An instance DB has been created for the channel and is supplied.
- "Instruction" is configured as source for the setpoint.

Procedure

To set a controller manually, proceed as follows:

1. Select the instance DB of the channel you want to tune.
2. Click the "Measurement on" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
3. Enter the respective values for the following parameters:
   - Proportional action
   - Proportional action weighting with setpoint jumps
   - Integral action
   - Derivative action
   - Coefficient DT1
   - Heating / cooling ratio
4. Click the icon "Send PID parameter to FM" in the "Tuning" group.
5. Select the "Change setpoint" check box in the "Current values" group.
6. Enter a new setpoint.
7. Deselect the check box "Manual mode", if necessary, and click the icon in the "Current Values" group.
   The controller works with the new parameters and controls the new setpoint.
8. Check the quality of the PID parameter to check the curve points.
9. Repeat steps 3 to 8 until you are satisfied with the controller results.

Note

You can use FMT_PID.OP.UNDO_PAR = TRUE to set the old controller parameters again.

Downloading tuned parameters in device (S7-300, S7-400)

Introduction

You can proceed as follows to prevent the parameters on the FM from being overwritten with the system data the next time you start up the CPU:

- You download the tuned PID parameters from the instance DB of the FMT_PID to the FM after each startup of the FM 355-2 with LOAD_PAR=TRUE.
- Download the PID parameters to the SDB.

Note

The PI, PID and SAVE parameter records cannot be stored in the SDB.

Procedure

Proceed as follows to download the parameters to the SDB:

1. Click on the icon "Transfer online data to an offline project".
   The PID parameters are saved in the parameter setting of the FM.
2. Stop CPU.
3. Click "Download to device".

Saving and retrieving controller parameters (S7-300, S7-400)

Back-up file records

The FM 355-2 features an effective parameter record system as well as a backed up parameter record system. Save and retrieval operations are performed by means of SAVE_PAR or UNDO_PAR in OP structure of the FMT_PID instruction.

For example, prior to a manual change you can save the current parameters via SAVE_PAR = TRUE. You can retrieve the last saved controller parameters via UNDO_PAR = TRUE and enable them for the controller. SAVE_PAR or UNDO_PAR is reset by FMT_PID at the end of this action.

If you want to know the values in the current parameter record or in the backed up parameter record, you can retrieve this information via FMT_PID.READ_PAR = TRUE or FMT_TUN.READ_OUT = TRUE:

Current parameters in the PAR structure of FMT_PID:	Parameters included in the backup of the FMT_TUN output parameters:
PFAC_SP	SAV_PFAC
GAIN	SAV_GAIN
TI	SAV_TI
TD	SAV_TD
D_F	SAV_D_F
CON_ZONE	SAV_CONZ

Current parameters in the PAR structure of FMT_PID:	Parameters included in the backup of the FMT_TUN output parameters:
RATIOFAC	SAV_RATI
CONZ_ON	SAV_CZON
P_SEL	SAV_PSEL

Note

At the end of controller tuning, the saved parameters are overwritten with parameter data that were valid prior to tuning (Exception: this intermediate saving operation is discarded in a cooling tuning process).

Change between PI and PID parameters

After tuning, the PI and PID parameters are saved on the FM355-2. You load these parameter records by setting FMT_PID.OP.LOAD_PID. If PID_ON in the structure PAR = TRUE, the PID parameter record is copied to the effective controller parameters; otherwise, the PI parameter record is copied. LOAD_PID is reset by FMT_PID at the end of this action. Of course, the value of PID_ON is retained, as it is not an operator control parameter.

If you want to know which values are stored in the PI or PID parameter record, you can view these data via FMT_TUN. Read out READ_OUT = TRUE:

	PI parameter record	PID parameter record
Controller GAIN	PI_GAIN	PID_GAIN
Integration time	PI_TI	PID_TI
Derivative time	0.0	PID_TD

Note

Please note the following:

- Controller parameters are only written back with UNDO_PAR or LOAD_PID if the controller gain (SAV_GAIN, PID_GAIN or PI_GAIN) is not equal to zero:
- This strategy takes into account the situation that no tuning has yet been made or that PID parameters are missing. For example, if PID_ON was TRUE and PID_GAIN = 0.0, PID_ON is set to FALSE and the PI parameters will be copied.
- D_F, PFAC_SP are set to default values by the tuning. The user can later modify them. LOAD_PID does not modify these parameters.
- With LOAD_PID, the control zone is always recalculated (CON_ZONE = 250/GAIN or 375/GAIN if RATIOFAC <> 0.0), irrespective of the status CONZ_ON = FALSE.

Pulse-free changeover with LOAD_PID and UNDO_PAR

The changeover is performed so that the sum of the proportional action and integral action stays equal. A changeover from PID to PI triggers a jump of the manipulated variable, as the derivative action is disabled. Initially, the changeover from PI to PID is pulse-free. After a few cycles, however, the derivative action will lead to a fast change.

Measuring motor actuating time (S7-300, S7-400)

Introduction

The motor transition time has to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Requirement

- The configured motor transition time must be half the size of the real transition time. The actuator is moved for a maximum of twice the transition time in one direction.
- The position feedback must be interconnected for actuators with analog position feedback.
- The endstop signals must be interconnected for actuators with endstop signals.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use position feedback" check box.
3. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. Click the ▶ "Start transition time measurement" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (OP.LMNS_ON = TRUE). The actuator will stop being moved.

Actuators with endstop signals

Proceed as follows to measure motor transition time with endstop signals:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use actuator endstop signals" check box.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop. Time measurement will start once the actuator has reached the lower endstop and will end when the actuator reaches the lower endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (OP.LMNS_ON = TRUE). The actuator will stop being moved.

Loading measured transition time

1. Click "Transfer online data to an offline project" to save the measured motor transition time locally in the project.
   The measured transition time is saved in the parameter setting of the FM.
2. Click "Download to device" to activate all changes the next time you start the CPU.

Cancelling transition time measurement

When you cancel the transition time measurement, the manual mode of the actuator signals is switched on (OP.LMNS_ON = TRUE) and the manual mode with manipulated value specification is switched off (OP.LMN_REON = FALSE). The actuator will stop being moved.

Working with the trend view (S7-300, S7-400)

Monitoring

1. Double click on "Commissioning" in the device view of the module.
2. Navigate to "Trend view" in the configuration view.
3. Select the instance DB you wish to monitor. The channel to which the instance DB has been assigned will be displayed.
4. Specify the update cycle in which the trend view is to record.
5. Click on "Measurement on" to start measuring.
6. Click on "Measurement off" to stop measuring.

Procedure

Proceed as follows to adjust the scale:

1. Double click on "Commissioning" in the device view of the module.
2. Navigate to "Trend view" in the configuration view.
3. To change the time axis:
   - Click on the arrow button next to the date to open a calendar. Select a date from the calendar. To lock the selection, click on the blue lock. This can be done for both start and end dates.
   - Click on the time scale. The cursor will appear as a left/right arrow. Hold down the cursor and drag it to the left or right.
     Time scaling will change if the start or end time is locked.
     If neither the start nor end time is locked, the specified period will change and the scaling remain the same.
4. To change the value axis:
   - Click on the input field for the high and low value displayed. Enter the desired value here. To lock the selection, click on the blue lock. This option is available for both the high and low value.
   - Click on the value scale. The cursor will appear as an up/down arrow. Hold down the cursor and drag it up or down.
     If the high or low value is locked, the scaling of the value axis will change.
     If neither the high nor the low value is locked, the specified range will change and the scaling remain the same.

3.5.6.4 FM 355-2: Diagnostics with LED display (S7-300, S7-400)

Error display from the group error LED (S7-300, S7-400)

Introduction

If the red group error LED lights up, there is either an error on the module (internal error) or in the line connections (external error).

If the yellow LED flashes, then the firmware has been deleted. This status can only occur in the case of faulty hardware or if the loading procedure of the firmware is aborted.

Which errors are displayed?

The following errors are displayed by the group error LED lighting up:

Type of error	Diagnostic message	Possible cause	Correction
Internal errors	Module defective	Hardware error	Replace the module
	Time watchdog tripped	Hardware error	Replace the module
	EEPROM content is invalid	Failure of the supply voltage when configuring	Reconfigure module
External errors	Incorrect parameters in module	Incorrect parameters have been transferred to the module	Reassign the module parameters
	Errors with the analog inputs or analog outputs	Analog input hardware error	Replace module
		Analog input wire break	Remedy wire break
		Analog input measuring range violation (underrange)	Check measuring signal
		Analog input measuring range violation (overrange)	Check the measuring signal
		Analog output wire break	Remedy wire break
		Analog output short circuit	Eliminate short circuit
	Missing external auxiliary supply	24 V supply missing	Restore 24 V supply

Diagnostic interrupt in the case of errors

All errors can trigger a diagnostic interrupt if you have enabled the diagnostic interrupt in the respective parameterization screen. From the diagnostic data records DS0 and DS1 you can see which errors have caused the LED to light up. The assignment of the diagnostic data records DS0 and DS1 is described in the next section.

Triggering diagnostic interrupts (S7-300, S7-400)

What is a diagnostic interrupt?

If the user program is to react to an internal fault or external error, you can configure a diagnostic interrupt. The cyclical CPU program is interrupted and the diagnostic interrupt OB, OB 82 is called.

Which events may trigger a diagnostic interrupt?

The list shows you which events can trigger a diagnostic interrupt:

- Module configuration missing or faulty
- Module defective
- Wire break when analog inputs are made (4 up to 20 mA only)
- High limit violated and low limit violated during analog inputs
- Load break or short-circuit during analog outputs

Enabling the diagnostic interrupt

You can enable or disable the diagnostic interrupt for the module in the basic parameter tab.

The default setting for the diagnostic interrupt is disabled.

Reactions to an event triggering an interrupt

If an event occurs that can trigger a diagnostic interrupt, the following occurs:

- The diagnostics information is written to the diagnostics data records DS0 and DS1 on the module.
- The group error LED lights up.
- Diagnostic interrupt OB is called (OB 82).
- Diagnostic data record DS0 is written to the start information of diagnostic interrupt OB.
- If there is no hardware defect, the module resumes control operation.

If no OB 82 is programmed, the CPU goes into STOP.

Diagnostics data record DS0 and DS1

The information about which event triggered a diagnostic interrupt is written to the diagnostic data records DS0 and DS1. The length of the diagnostic data record DS0 is four bytes; the record length of DS1 is 16 bytes of which the first 4 bytes are identical with DS0.

Diagnostic data records DS0 and DS1 (S7-300, S7-400)

Read data record from the module

The diagnostic data record DS0 is automatically transferred to the start information when the diagnostic OB is called. There these four bytes are stored in the local data (byte 8 - 11) of the OB 82.

The diagnostic data record DS1 (and hence also the content of the DS0) can be read from the module by means of FB 55 "FMT_DS1". This only makes sense if an error is reported in a channel in the DS0.

The FB 55 "FMT_DS1" must be called in the same OB as an FB FMT_PID that might be present (e.g. OB 35). You can do this as follows: While OB 82 is being processed, set the READ_DS1 bit. The FMT_DS1 in OB 35 then reads the diagnostic data record DS1.

How does the diagnosis text appear in the diagnostics buffer?

If you want to enter the diagnostics message in the diagnostics buffer, call the "WR_REC" instruction "Enter user-specific message in the diagnostics buffer" in the user program. The event number of the respective diagnosis message is specified on the input parameter EVENTN. The interrupt is entered in the diagnostics buffer defined as x = 1 for incoming and x = 0 for outgoing. In addition to the time of the entry, the appropriate diagnosis text is displayed in the "Meaning" column.

The assignment of the diagnostic data records DS0 and the Start information

The following table shows the assignment of the diagnostic data record DS0 in the Start information. Any bits not listed have no meaning and are zero.

Table 3-5 Assignment of diagnostic data record DS0

Byte	Bit	Meaning	Remark	Event no.
0	0	Module fault	Is set at each diagnostics event	8:x:00
	1	Internal error	Is set at every internal error: • Watchdog timeout • EPROM error • ADC/DAC error • Analog input hardware error	8:x:01
	2	External error	Is set at every external error: • Missing external auxiliary voltage • Faulty parameter assignment • Analog input wire break (only in range 4 to 20 mA) • Analog input below measuring range • Analog input measuring range exceeded • Analog output load break • Analog output short circuit	8:x:02
	3	Error in one channel	See DS1, from byte 7 for further breakdown	8:x:03
	4	Missing external auxiliary voltage	24 V supply for FM 355-2 has failed	8:x:04
	6	Unused	-	-
	7	Faulty parameter assignment	The module is unable to use a parameter. Reason: Parameters unknown or illegal combination of parameters. See menu PLC > Parameter assignment error display.	8:x:07
1	0 ... 3	Module class	8 is always assigned.	-
	4	Channel-specific diagnosis	If set, the module can provide additional channel information and a channel error exists (see DS1, byte 7 to 12)	-
2	3	Time watchdog responds	Hardware fault	8:x:33
3	2	EPROM error	Module defective	8:x:42
	4	ADC/DAC error	Module defective	8:x:44

Diagnostic data record DS1

The length of diagnostic data record DS1 is 16 bytes. The first 4 bytes are identical to diagnostic data record DS0. The table below shows the assignment of the remaining bytes. Any bits not listed have no meaning and are zero.

Diagnostic data record DS1 of the FM 355-2

Table 3-6 Assignments of bytes 4 to 12 of diagnostic data record DS1 of the FM 355-2

Byte	Bit	Meaning	Remark		Event no.
4	0 ... 7	Channel type	75H is always assigned.		-
5	0 ... 7	Length of the diagnostics information	8 is always assigned.		-
6	0 ... 7	Number of channels	5 always assigned (4 controllers + 1 reference channel)		-
7	0 ... 4	Channel error vector	One bit is assigned to each channel (0...3; 4 for reference channel).		-
8	0	Analog input hardware error	Channel-specific diagnostics channel 0		8:x:B0
	1	Unused			8:x:B1
	2	Analog input wire break (only in range 4 to 20 mA)			8:x:B2
	3	Unused			8:x:B3
	4	Analog input below measuring range			8:x:B4
	5	Analog input measuring range exceeded			8:x:B5
	6	Analog output wire breakage		Only with the current output of the C controller	8:x:B6
	7	Analog output short circuit		Only with the voltage output of the C controller	8:x:B7
9	0 ... 7	see Byte 8	Channel-specific diagnostics channel 1		See above
10	0 ... 7	see Byte 8	Channel-specific diagnostics channel 2		See above
11	0 ... 7	see Byte 8	Channel-specific diagnostics channel 3		See above
12	0 ... 5	see Byte 8	Diagnostics for reference channel		See above

Additional information

For more information on the "RET_VALU" parameter, refer to List of RET_VALU messages (Page 1062) and the "Extended instructions" section of the help system.

Measuring transducer fault (S7-300, S7-400)

Measuring transducer faults

The FM 355-2 can identify the following measuring transducer faults:

- Measuring range violation (underrange)
- Measuring range violation (overrange)
- Wire break (not for all measuring ranges)

If one of these faults occur, the group error bit "external error" is set in diagnostic data record DS0 and channel-specific error bits are set in diagnostic data record DS1 (see table above). When these faults disappear, the respective bits are reset.

The following table shows the specific measuring range limits at which the error bits are set or reset:

Measuring range	Error bit measuring range violation (underrange) at...	Error bit measuring range violation (overrange) at...	Error bit wire break display
	DS1: Byte 10 to 26, bit 4	DS1: Byte 10 to 26, bit 5	DS1: Byte 10 to 26, bit 2
0 to 20 mA	< - 3.5 mA	> 23.5 mA	-
4 to 20 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at > 3.8 mA	> 22.8 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at > 3.8 mA
0 to 10 V	<- 1.175 V	> 11.75 V	-
Pt 100 (-200 to 850 °C) (-328 to 1562 °F)	< 30.82 mV	> 650.46 mV	-
Pt 100 (-200 to 556 °C) (-328 to 1032 °F)	< 30.82 mV	> 499.06 mV	-
Pt 100 (-200 to 130 °C) (-328 to 264 °F)	< 30.82 mV	> 254.12 mV	-
Type B thermocouple	< 0 mV	> 13.81 mV	-
Type E thermocouple (-270.0 to 1,000.03 °C)	< -9.84 mV	> 76.36 mV	-
Type J thermocouple	< -8.1 mV	> 69.54 mV	-
Type K thermocouple	< -6.45 mV	> 54.88 mV	-
Type R thermocouple	< -0.23 mV	> 21.11 mV	-
Type S thermocouple	< -0.24 mV	> 18.7 mV	-
Free thermocouple	< low input value of the polyline	> high input value of the polyline	-

List of RET_VALU messages (S7-300, S7-400)

RET_VALU messages

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
7000	28672	-32624	First call with REQ=0: no data transmission active; BUSY has the value 0.
7001	28673	-32624	First call with REQ=1: data transmission initiated; BUSY has the value 1.
7002	28674	-32624	Interim call (REQ irrelevant). Data transmission already active; BUSY has the value 1.
8090	32912	-32624	Specified logical base address invalid: There is no assignment in the SDB1/SDB2x, or it is not a base address.
80A0	32928	-32608	Negative acknowledgment when reading from the module. Module was removed during the read operation or the module is defective.
80A1	32929	-32607	Negative acknowledgment when writing to the module. Module was removed during the write operation or the module is defective.
80A2	32930	-32606	DP protocol error at layer 2
80A3	32931	-32605	DP protocol error in user interface/user
80A4	32932	-32604	Communication bus error
80B1	32945	-32591	Incorrect length specification. FM_TYPE parameter in channel DB not set correctly for the module in use.
80B2	32946	-32590	The configured slot is not being used.
80B3	32947	-32589	Actual module type is not match configured module type.
80C0	32960	-32576	Module data not ready for reading.
80C1	32961	-32575	Data of a write job of the same type have not yet been processed by the module.
80C2	32962	-32574	The module is currently processing the maximum possible number of jobs.
80C3	32963	-32573	Required resources (memory etc.) currently occupied.
80C4	32964	-32572	Communication error
80C5	32965	-32571	Distributed I/O not available.
80C6	32966	-32570	Priority class abort (restart or background).
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read off the DB. (Write job)
8532	34098	-31438	DB number of the parameter DBs too high. (Write job)
853A	34106	-31430	Parameter DB not present. (Write job)
8544	34116	-31420	Error at n-th (n > 1) read access to a DB after an error has occurred. (Write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written to the DB. (Read job)
8730	34608	-30928	Parameter DB in the CPU is write protected. The data cannot be written to the DB (read job)
8732	34610	-30926	DB number of the parameter DBs too high (Read job)

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
873A	34618	-30918	Parameter DB not present. (Read job)
8745	34629	-30907	Error at n-th (n > 1) write access to a DB after an error has occurred. (Read job)
80ff	33023	-32513	Incorrect index specification with block FMT_PAR
Errors 80A2..80A4 and 80Cx are temporary, i.e. after a waiting period they can be eliminated without any action on your part. Messages of the 7xxx form indicate temporary operating states of communication.			

3.5.7 Using FM 455 (S7-300, S7-400)

3.5.7.1 Configuring FM 455 (S7-300, S7-400)

Configuring the hardware (S7-300, S7-400)

Procedure

Configuration assumes that you have set up a project in which you can save the parameter assignment. The following briefly explains the most important steps.

1. Start the TIA portal and open the device configuration.
2. Select one module rack.
3. Select one controller module from your module catalog.
4. Move the controller module to the appropriate place on the module rack.
5. From the configuration table, note the input address of the module, for example 512.
   The value that you read is displayed in decimal format.

3.5.7.2 Parameter assignment FM 455 (S7-300, S7-400)

Module parameters (S7-300, S7-400)

Procedure

When assigning parameters you set the module parameters.

1. Double-click on the "Parameter setting" in the device view of the module. In the Inspector window, select the "Properties" tab.
   Result: The properties are displayed in the inspector window.
2. Configure the basic parameters and the remaining parameters of the module.
3. Configure the remaining device parameters in the Design view.

4. Save the parameters entered with "Project> Save".
5. Translate the project with "Edit > Translate".
6. Load the configuration data with the CPU in STOP mode by selecting "Online > Device load..."
   Result: The data stored in the memory of the CPU are then transferred directly to the module.
7. Carry out a CPU start-up.

Analog input parameter (S7-300, S7-400)

Evaluation of analog inputs (S7-300, S7-400)

Procedure

To select a sensor type, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Digital input > input [No]> Evaluation of analog inputs".
3. Enter the following parameters:
   - Analog input resolution
   - Sensor type analog input
     For different sensor types the option "open-circuit monitoring" will be enabled. The option may be activated depending on requirements.
   - Junction temperature
4. Save the parameters entered with "Project> Save".

Filter functions (S7-300, S7-400)

Procedure

To set the filter functions, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Analog input > input [No]> Filter functions".

3. Enter the following parameters:
   - To filter on a time constant you should enable the "Signal filter" option.
     If the "signal filter" option is selected, the time constant filter can be set.
   - Filter time constant
     Can only be set if the "signal filter" option is enabled.
   - To linearize the encoder signal you should activate the "square root filter" option.
   - To linearize the encoder signal characteristics you should activate the "polygon filter" option.
4. Save the parameters entered with "Project> Save".

Signal filter

Filter 1. order, whose time behavior is set through the time constants.

Square root filter

Used to linearize encoder signals in which the actual value is present as a physical quantity, which is square in relation to the measured process variable.

Polygon filter

Used to linearize encoder characteristics.

The 13 selectable points are selectable in:

- mA at current input
- mV with voltage input

Scaling (S7-300, S7-400)

Procedure

To convert the input signal into a physical unit, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Analog input > input [No]> Scaling".
3. Enter the following parameters:
   - High value
   - Low value
4. Save the parameters entered with "Project> Save".

Scaling

To convert the input signal into another physical unit through linear interpolation between the initial value (low value) and the ending value (high value).

Deviation (S7-300, S7-400)

Implementing a constant or a cascade controller (S7-300, S7-400)

Constant or cascade controller

In the controller structure of constant or cascade controllers the effective current value is equal to the actual value A.

Selecting the controller type

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]".
3. In the "Encoder type" section select the "Constant or cascade controller".
4. Save the parameters entered with "Project> Save".

Processing setpoints

To edit setpoints, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> Channel [No]> Deviation".
3. Set in the different areas the corresponding parameters:
   - Setpoint source
     Select the source of the setpoint.
   - Safety setpoint
     Set your safety setpoints here. Set also the response to a CPU fault and the startup behavior.
   - Setpoint ramp
     Set the start-up time.
   - Setpoint limit
     Select the high and low setpoint limits.
4. Save the parameters entered with "Project> Save".

Selecting a signal source for actual value A

To select the signal source for the actual value A, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> Channel [No]> Deviation".

3. In the "Control value" section select the corresponding source. The following choices are available:
   - Zero
   - Processed analog input
4. Save the parameters entered with "Project> Save".

Implementing three component controllers (S7-300, S7-400)

Three-term controller

In the controller structure of a three-term controller, the effective actual value is obtained by totaling the three actual values A, B and C and then adding an adjustable offset. The actual values B and C can also be evaluated using factors.

Selecting the controller type

Proceed as follows to implement a three-term controller:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller> channel [No]".
3. In the "Controller type" section, select the "Three-term controller" option.
4. Save the parameters entered with "Project > Save".

Processing setpoints

Proceed as follows to adjust the setpoint processing mode:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Setpoint source" section, select the source from which the setpoint is to be obtained.
4. In the "Safety setpoint" section, enter a value to be utilized upon CPU fault and start up as a safety value.
5. Select the behavior of the controller module for "Reaction to CPU errors" and "Start-up behavior". You may select from amongst the following options:
   - Setpoint = last valid setpoint: the setpoint will be assigned the last valid setpoint upon occurrence of the event. The last valid setpoint will remain unchanged.
   - Setpoint = safety setpoint: the setpoint will be assigned the safety setpoint upon the occurrence of the event. This enables secured emergency operation.
6. In the "setpoint ramp" section, set the rise time to be adhered to upon a setpoint change to reduce overshoot.
7. In the "Setpoint limits" section, set both an high and a low limit for the setpoint.
8. Save the parameters entered with "Project > Save".

Selecting a signal source for actual values A, B and C

Proceed as follows to set the signal sources for the actual values A, B and C:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. Select the signal source for each actual value in the "Manipulated value" section. You can select whether you use no signal ("zero") as a source or one of the processed analog inputs.
4. Save the parameters entered with "Project > Save".

Determining factors for actual values B and C and the offset value

Proceed as follows to set the signal sources for the actual values A, B and C:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the"Total" section, set the necessary weighting factor for the actual values B and C.
4. Set the required "offset" value to be added to the actual values.
5. Save the parameters entered with "Project > Save".

Implementing ratio or mixture controllers (S7-300, S7-400)

Selecting the controller type

Proceed as follows to implement a ratio or blending controller:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller> channel [No]".
3. Select the "Ratio or blending controller" in the "Controller type" section.
4. Save the parameters entered with "Project > Save".

Processing setpoints

Proceed as follows to adjust the setpoint processing mode:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Setpoint source" section, select the source from which the setpoint is to be obtained.
4. In the "Safety setpoint" section, enter a value to be utilized upon CPU fault and start up as a safety value.

5. Select the behavior of the controller module for "Reaction to CPU errors" and "Start-up behavior". You may select from amongst the following options:
   - Setpoint = last valid setpoint: the setpoint will be assigned the last valid setpoint upon occurrence of the event. The last valid setpoint will remain unchanged.
   - Setpoint = safety setpoint: the setpoint will be assigned the safety setpoint upon the occurrence of the event. This enables secured emergency operation.
6. In the "setpoint ramp" section, set the rise time to be adhered to upon a setpoint change to reduce overshoot.
7. In the "Setpoint limits" section, set both an high and a low limit for the setpoint.
8. Save the parameters entered with "Project > Save".

Selecting the signal source for actual values A and D

Proceed as follows to set the signal sources for actual values A and D:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. Select the signal source for each actual value in the "Manipulated value" section. You can choose whether to use no signal ("zero") as a source or one of the processed analog inputs. For the actual value D, you may also select the manipulated value of one of the controllers.
4. Save the parameters entered with "Project > Save".

Determining multipliers

Proceed as follows to set the signal sources for actual values A and D:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. Select the desired offset in the "Multiply" section.
4. Save the parameters entered with "Project > Save".

For ratio controllers, the actual value A is used as control variable while the actual value D stores the ratio magnitude. The setpoint input value is used as a ratio factor. Multiplying it by the actual value of D and adding the adjustable offset will yield the effective setpoint value. If actual value D is off, only the offset will be added to the setpoint.

Setting the alarm generation (S7-300, S7-400)

Procedure

Proceed as follows to set interrupt generation:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".

3. Select the necessary options in the "Generate interrupt" section:
   - Enter the signal for which the alarm is to be set in the "Limit monitor for" section.
   - In "Alarm high limit", set the high limit beyond which the alarm is to be triggered.
   - In "Warning high limit", set the value beyond which the "approaching high limit" warning is to be issued.
   - In "Warning low limit", set the value below which the "approaching low limit" warning is to be issued.
   - In "Alarm low limit", set the low limit below which the alarm is to be triggered.
   - You can set hysteresis under "Hysteresis" to prevent the monitor flickering.
4. Save the parameters entered with "Project > Save".

Control algorithms (S7-300, S7-400)

Implementing a P control (S7-300, S7-400)

Procedure

To implement a P control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "P" option.
   All configurable options for a P control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PI control (S7-300, S7-400)

Procedure

To implement a PI control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PI" option.
   All configurable options for a PI control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing an I control (S7-300, S7-400)

Procedure

To implement an I control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "I" option.
   All configurable options for an I control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PID control (S7-300, S7-400)

Procedure

To implement a PID control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PID" option.
   All configurable options for a PID control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PD control (S7-300, S7-400)

Procedure

To implement a PD control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PD" option.
   All configurable options for a PD control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a Fuzzy temperature controller (S7-300, S7-400)

Procedure

To implement a fuzzy temperature control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "Fuzzy temperature controller module" option.
   All configurable options for a fuzzy temperature controller module will be activated and all other options disabled.
4. Save the parameters entered with "Project> Save".

Suppressing the deviation noise component (S7-300, S7-400)

Procedure

To suppress the noise component in the deviation, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "General" section, set the "Dead zone width" parameter.
4. Save the parameters entered with "Project> Save".

Switching disturbances (S7-300, S7-400)

Procedure

To switch a disturbance, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. From the "Disturbance source" list in the "General" section select the disturbance source.
   Select the entry "zero" to perform no disturbance switching.
4. Enter the "Disturbance factor" value you wish to use as a disturbance weight factor.
   A value <0 weakens the disturbance whereas a value> 0 increases the disturbance. A value of 1 keeps the disturbance unchanged.
5. Save the parameters entered with "Project> Save".

Setting the feedback P portion (S7-300, S7-400)

Procedure

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "P portion" section select the "In feedback" option.
   If the option is enabled, the P portion is set in feedback.
4. Save the parameters entered with "Project> Save".

Attenuating the D portion for setpoint changes (S7-300, S7-400)

Selecting the controller type

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "D portion" section enter the D portion source in the "D portion" option.
4. In the "Differentiation factor" section, set the D portion factor.
5. Save the parameters entered with "Project> Save".

Reversing the controller effect (S7-300, S7-400)

Principle

Controller reversal may be achieved by setting a negative proportional gain thus moving from "increasing deviation value = increasing correction value" to "increasing deviation value = decreasing correction value".

Procedure

To reverse the controller effect, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "General" section, set a proportional gain of opposite sign.
   As an example, if the current setting shows a positive proportional gain and you wish to reverse the controller effect, set a negative proportional gain.
4. Save the parameters entered with "Project> Save".

Controller output (S7-300, S7-400)

Implementing a continuous controller (S7-300, S7-400)

A continuous controller can be created only with an analog module.

Procedure

1. Changing an external control value (Page 1075)
2. Setting the safety control value (Page 1076)
3. Changing the tracking (Page 1075)
4. Setting control value limits (Page 1077)
5. Controlling two actuators - split range (Page 1077)
6. Selecting the output signal and the signal type (Page 1078)

Realizing a step controller without position feedback (S7-300, S7-400)

A step controller without position feedback can be realized only with a digital module.

Procedure

1. Select controller output step controller without position feedback
2. Changing an external control value (Page 1075)
3. Setting the safety control value (Page 1076)
4. Using the pulse shaper step controller (Page 1077)

Realizing a step controller with position feedback (S7-300, S7-400)

A step controller with position feedback can be realized only with a digital module.

Procedure

1. Select controller output step controller with position feedback
2. Changing an external control value (Page 1075)
3. Changing the tracking (Page 1075)
4. Setting the safety control value (Page 1076)
5. Setting control value limits (Page 1077)
6. Setting position feedback (Page 1076)
7. Using the pulse shaper step controller (Page 1077)

Realizinga pulse controller (S7-300, S7-400)

A pulse controller can be realized only with a digital module.

Procedure

1. Select controller output pulse controller
2. Changing an external control value (Page 1075)
3. Changing the tracking (Page 1075)
4. Setting the safety control value (Page 1076)
5. Setting control value limits (Page 1077)
6. Prepare the analog-to-digital signal conversion
7. Converting analog into digital signal (Page 1078)

Assigning parameters for the functions of the controller output (S7-300, S7-400)

Changing an external control value (S7-300, S7-400)

Procedure

For changing the external control variable, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "External control variable" select the option to activate the external control variable. You may choose from the following options:
   - Activation of the function module.
   - Activation of a function module or one of the digital inputs.
4. Save the parameters entered with "Project> Save".

Changing the tracking (S7-300, S7-400)

Procedure

To change the tracking proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Tracking" section select the option to activate the external control variable. You may choose from the following options:
   - Activation of the function module.
   - Activation of a function module or one of the digital inputs.

4. In the "Tracking" section select the source of the signal you wish to track. You may choose from the following options:
   - Zero, while no source is used.
   - One of the analog inputs.
5. Save the parameters entered with "Project> Save".

Setting position feedback (S7-300, S7-400)

Procedure

To set the position feedback, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to 'Digital Output> output [No.] ".
3. In the "Position feedback input" section select the "Source" option to choose the signal for position feedback.
   - Zero: no position feedback.
   - One of the processed analog inputs.
4. Save the parameters entered with "Project> Save".

Setting the safety control value (S7-300, S7-400)

Procedure

To set the safety control value, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "safety control value" section select the value that should be used as a safety control value.
4. Under "Activation" select the option to activate the external control variable. You may choose from the following options:
   - Activation of the function module.
   - Activation of a function module or one of the digital inputs.
5. Under "Action on module start" select the option which should be executed as the module starts. You may choose from the following options:
   - Control mode: sets the module immediately into control mode upon startup.
   - Safety manipulated value: initially, the safety manipulated value is controlled.
6. Under "Action on controller interference of the actual value A" select the option which should be executed as a controller interference of the actual value A occurs.

7. Under "Action on input controller interference" select the option which should be executed as an input controller interference occurs.
8. Save the parameters entered with "Project> Save".

Setting control value limits (S7-300, S7-400)

Procedure

To set control value limits you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Output signal limit" enter the following values:
   - High value: illustrates the high limit for the output signal.
   - Low value: illustrates the low limit for the output signal.
4. Save the parameters entered with "Project> Save".

Controlling two actuators - split range (S7-300, S7-400)

Procedure

To control two actuators you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Split range" section select the option "Enable" to effectively control two actuators.
4. Set the following values for both the input and output signals A and for the input and output signals B:
   - Low input signal.
   - High input signal.
   - Low output signal.
   - High output signal.
5. Save the parameters entered with "Project> Save".

Using the pulse shaper step controller (S7-300, S7-400)

Procedure

To configure the pulse shaper you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to 'Digital Output> output [No.] ".

3. In the "Pulse shaper" area, enter the following parameters:
   - Engine operating time.
   - Minimum pulse duration.
   - Minimum break duration.
4. Select the best acceptable option for the high and the low values:
   - From the function module.
   - From the function module or one of the digital inputs.
5. Save the parameters entered with "Project> Save".

Converting analog into digital signal (S7-300, S7-400)

Procedure

To convert an analog signal into a digital signal you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate to the "Digital output> channel [No.]" in the design view.
3. In the "Pulse shaper" area, enter the following parameters:
   - Minimum pulse duration.
   - Minimum break duration.
4. Save the parameters entered with "Project> Save".

Selecting the output signal and the signal type (S7-300, S7-400)

Procedure

To select the output signal and the signal type you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate to the "Digital output> channel [No]" in the design view.
3. Under the option "Source" select the signal you wish to use as output signal. You may choose among the following options:
   - Zero: no source for the output signal.
   - One of the analog inputs.
   - One of the control values for controller A or B.
4. Under the option "Signal type" select the type for the output signal you wish to use.
5. Save the parameters entered with "Project> Save".

3.5.7.3 Programming FM 455 (S7-300, S7-400)

3.5.7.4 Commissioning FM 455 (S7-300, S7-400)

Creating and supplying instance DBs (S7-300, S7-400)

Procedure

To meet the requirements for commissioning, follow these steps:

1. Create a user program that calls up the PID_FM_455 instruction cyclically in a cyclic interrupt OB.
2. Set READ_VAR = TRUE cyclically in the user program.
3. Create an individual instance DB of the PID_FM_455 instruction for each controller channel.
4. Enter the module address for each instance DB in the PID_FM_455.MOD_ADDR parameter.
   Use the start address that you specified when configuring your hardware as the module address of the FM 455.
5. Enter the number of the controller channel (1 to 16) in the PID_FM_455.CHANNEL parameter.
6. If you use the PID_PAR_455 instruction, enter the number of the controller channel in the PID_PAR_455.CHANNEL parameter.
7. Check to ensure that the "Sensor type analog input" of this channel is selected.
   If the analog input is not processed, all PID parameters are set to zero and a manipulated variable is not calculated.

Note

Multi-instance DBs

If you use multi-instance DBs in your project, you must commission the FM via a watch table.

Commissioning the controller (S7-300, S7-400)

Introduction

You tune the PID parameters as follows:

- PID/PI controllers (step controller, continuous controller, pulse controller): Manually
- Fuzzy temperature controller: Automatically
- Step controller: Automatically with instruction TUN_ES
- Continuous controller: Automatically with instruction TUN_EC
- Pulse controller: Automatically with instruction TUN_EC

Procedure

Proceed as follows to tune a controller:

1. Put the CPU in RUN mode.
2. If you are using a step controller, measure the motor actuating time.
3. Tune the PID parameters with one of the options listed above.
4. Monitor the control loop with the trend view.

See also

Measuring the motor transition time (Page 1087)

Tuning PID parameters manually (S7-300, S7-400)

Requirement

You have selected the "PID controller" control algorithm during parameter assignment.

Procedure

To set a PID/PI controller manually, proceed as follows:

1. Select the instance DB of the channel you want to tune.
2. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
3. Enter values for the proportional action, integral action, derivative action and delay.
4. Click the icon "Send PID parameters to FM" in the "Tuning" group.
5. Select the "Change setpoint" check box in the "Current values" group.
6. Enter a new setpoint.
7. Clear the check box "Manual mode", if necessary, and click the icon in the "Current values" group.
   The controller works with the new parameters and controls the new setpoint.
8. Check the quality of the PID parameters based on the trend characteristics.
9. Repeat steps 3 to 8 until you are satisfied with the controller results.

Fuzzy system identification (S7-300, S7-400)

Requirements on process with fuzzy temperature controller (S7-300, S7-400)

Requirement

The process should fulfill the following requirements in order to achieve optimal control using the fuzzy temperature controller:

- At bath heatings the liquid to be heated has to be mixed thoroughly.
- In case of bath-in-bath control systems both liquids have to be mixed thoroughly. At the same time good heat transitions between all the heat-transferring media have to be ensured. In the case of materials with poor heat-transferring properties large transfer surfaces should ensure good heat transportation.
- In case of room temperature control systems thorough mixing (for example with fans) has to be ensured.
- The controlling system gain may not exceed the factor of 3.
- The delay time may not exceed 3% of the recovery time.
- The temperature to be controlled should change by a maximum of 1‰ of the specified maximum temperature at the maximum manipulated value output within the sampling time of the controller.

Classification of the Controlled Systems

A controlled system or a process to be controlled is characterized by parameters such as the heat output, the heating mass or the heating capacity of the medium to be heated. With regard to the fuzzy controller a difference is made between "critical" and "non-critical" temperature controlled systems as follows: The control system becomes increasingly critical:

- The greater the heat output,
- the greater the heating capacity of the heating,
- The lower the heating capacity of the medium to be heated,
- the greater the heat transition resistance,
- the smaller the heat transfer surface.

After a manipulated value step change has been applied to the controlled system, it reacts with a step response. The controlled system can also be classified on the basis of this step response: The control system becomes increasingly critical the greater the ratio t_u / t_g is and the greater the controlled system gain is. With t_u / t_g < 1/10 you have a non-critical control section.

See also

Characteristic values of the control section (Page 614)

Process of fuzzy system identification (S7-300, S7-400)

Process of parameter tuning with fuzzy temperature controller

Parameter optimization is based on a self-optimizing fuzzy controller.

The identification of the controlled system begins with a monitoring phase during which no heat output takes place. The duration of the monitoring phase is as follows:

- Monitoring phase continuous controller: Approx. 1 min
- Monitoring phase step controller: Approx. 1 min + 1/2 x actuating time of final control element

This time is used in order to determine temperature trends in the heating medium. Afterwards the maximum heating output of 100% is output. This is visible at the LMN output parameter in the instance DB of the PID_FM instruction.

The range of the first 4% of the temperature increase of the setpoint value range is used for the identification, whereby information about the process response is obtained at 1% and 4% respectively of the temperature increase on the basis of the time that has passed.

The identification is complete when the heating output becomes less than 100%. This behavior can be used, for example, to deactivate the identification via the user program. After the identification phase the controller continues to operate with the determined parameters.

If the setpoint value step change is not adjusted and if the heating output remains permanently at zero, the identification has been terminated unsuccessfully, meaning that the controller cannot control the connected controlled system.

Controller behavior with different control sections

Problems do not arise in case of an "uncritical" controlled system, neither during identification nor during controlling.

The identification of a controlled system that is "too critical" is cancelled. Controlling of an identified "critical" controlled system is carried out very "carefully" or slowly.

Performing fuzzy system identification (S7-300, S7-400)

Requirement

You have selected the "Fuzzy controller" control algorithm during parameter assignment.

You have to ensure as far as possible that the controlled system has settled (no heating-up or cooling-down process) or is changing slowly and monotonously before the identification is carried out. The criterion can be that the temperature change is to approach a straight during a period of one minute. At faster processes this requirement is particularly relevant.

Since the manipulated variable zero is output for approx. one minute by the controller at the beginning of identification, the temperature to be controlled has to lie near the ambient temperature.

Starting fuzzy system identification

Follow these steps to tune a fuzzy controller:

1. Select the instance DB of the channel you want to tune.
2. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
3. Enter a new setpoint (SP_OP) in the "Tuning" group.
   The new setpoint must meet the following conditions:
   – Setpoint step change > 5%
   – New setpoint > process value + setpoint high limit x 0.12
4. Click "Fuzzy process identification" in the "Tuning" group.

Result

The new setpoint (SP_OP), SP_OP_ON = TRUE and FUZID_ON = TRUE are sent to the FM. Starting the fuzzy system identification. The IDSTATUS parameter of the CH_DIAG instruction provides information on the identification state. You see in the status display field if a fuzzy system identification is activated or running.

Deactivating fuzzy system identification

As long as the bit FUZID_ON = TRUE, the next identification is started whenever the setpoint step change is sufficiently large.

Proceed as follows to turn off the tuning mode:

1. Enter FUZID_ON in a watch table.
2. Set FUZID_ON = FALSE.

Note

Deactivation of the tuning mode FUZID_ON = FALSE before identification has been completed does not stop the identification. An identification process that has been started continues to run under all circumstances – with the exception of a negative setpoint value step change.

Repeating Fuzzy system identification

A renewed identification is carried out with each additional setpoint step change ≥12% unless the identification has been deactivated again with FUZID_ON = FALSE.

It is also possible to restart the identification by reducing and then increasing the setpoint by 12> %.

Canceling fuzzy system identification

Identification can be canceled in the following cases:

- By the controller if a "critical" controlled system is identified. After canceling the controller is in the error state. This state is indicated by the fact that the manipulated variable is reset permanently by the controller. This is also not changed by deactivating the optimization. The error state is deleted by starting a new identification.
  Information about the state of the identification is available through the IDSTATUS parameter of the CH_DIAG FB. More information on this can be found at the end of this section under "Controller State Information".
- By the operator by generating a negative setpoint value step change, whereby the setpoint value must lie below that of Condition 2 (see above "How To Start identification").

Proceed as follows to cancel tuning:

1. Enter a new setpoint that is less than the current process value and meets the same conditions as for the start of fuzzy system identification.
2. Click "Fuzzy process identification" in the "Tuning" group.

Downloading tuned parameters in device (S7-300, S7-400)

Introduction

You can proceed as follows to prevent the parameters on the FM from being overwritten with the system data the next time you start up the CPU:

- You download the tuned PID parameters from the instance DB of the PID_FM to the FM after each startup of the FM with LOAD_PAR=TRUE.
- Download the PID parameters to the SDB.

Note

The PI, PID and SAVE parameter records cannot be stored in the SDB.

Procedure

Proceed as follows to download the parameters to the SDB:

1. Click on the icon "Transfer online data to an offline project".
   The PID parameters are saved in the parameter setting of the FM.
2. Stop CPU.
3. Click "Download to device".

Working with the trend view (S7-300, S7-400)

Monitoring

1. Double click in the device view under the "Commissioning" module.
2. Navigate in the project view to "Trend view".
3. Select the instance DB you wish to monitor. The channel to which the instance DB has been assigned will be displayed.
4. Specify the update cycle in which the trend view is to record.
5. Click on "Measurement on" to start measuring.
6. Click on "Measurement off" to stop measuring.

Procedure

To adjust the scale, proceed as follows:

1. Double click on "Commissioning" in the device view of the module.
2. Navigate to "Trend view" in the configuration view.
3. To change the time axis:
   - Click on the arrow-button next to the date to open a calendar. Select a date from the calendar. To lock the selection, click on the blue lock. This possibility exists for the beginning and end dates.
   - Click on the time scale. The mouse pointer will be illustrated as a left/right arrow. Press the mouse and drag it to the left or right.
     If the beginning or end time is locked, the time scaling will change.
     If neither the beginning nor the end time is locked, the specified period will change, the scaling remains the same.
4. To change the value axis:
   - Click on the input field for the value displayed above or below. Enter your desired value here. To lock the selection, click on the blue lock. This possibility exists for the high and low value.
   - Click on the value scale. The mouse pointer will be illustrated as an up/down arrow. Press the mouse and drag it up or down.
     If the high or low value is locked, the scaling of the value axis will change.
     If neither the high nor the low value is locked, the specified range will change, the scaling remains the same.

Changing parameters with the HMI (S7-300, S7-400)

Procedure

Proceed as follows to change the controller parameters of the instruction PID_FM with the HMI:

1. Write the parameter you want to change from the HMI to an auxiliary DB on the CPU (see ①).
2. Set PID_FM.COM_RST to TRUE.
   The instruction performs an initialization run. The parameters are synchronized between FM and CPU (see ②).
3. Transfer the parameters you want to changed from the auxiliary DB to the instance DB of the instruction PID_FM (see ③).
4. Set PID_FM.LOAD_PAR to TRUE.
   The changed parameters are transferred to the FM (see ④).

It is necessary to store the parameters in an auxiliary DB because PID_FM after the start of the CPU with COM_RST = TRUE reads those parameters from the module that the CPU has previously transferred from the system data to the FM.

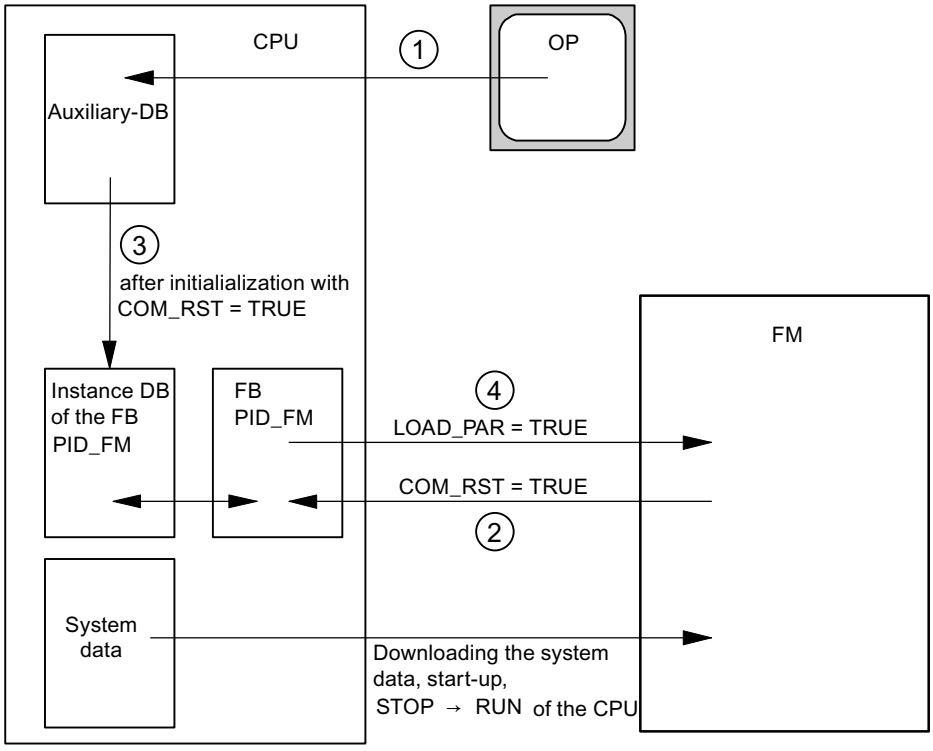

Measuring the motor transition time (S7-300, S7-400)

Introduction

The motor transition time has to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Requirement

- The configured motor transition time must be half the size of the real transition time. The actuator is moved for a maximum of twice the transition time in one direction.
- The position feedback must be interconnected for actuators with analog position feedback.
- The endstop signals must be interconnected for actuators with endstop signals.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use position feedback" check box.
3. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. Click the ▶ "Start transition time measurement" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (LMNSOPON = TRUE). The actuator will stop being moved.

Actuators with endstop signals

Proceed as follows to measure motor transition time with endstop signals:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use actuator endstop signals" check box.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop. Time measurement will start once the actuator has reached the lower endstop and will end when the actuator reaches the lower endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (LMNSOPON = TRUE). The actuator will stop being moved.

Loading measured transition time

1. Click "Transfer online data to an offline project" to save the measured motor transition time locally in the project.
   The measured transition time is saved in the parameter setting of the FM.
2. Click "Download to device" to activate all changes the next time you start the CPU.

Cancelling transition time measurement

When you cancel the transition time measurement, the manual mode of the actuator signals is switched on (LMNSOPON = TRUE) and the manual mode with manipulated value specification is switched off (LMNOP_ON = FALSE). The actuator will stop being moved.

3.5.7.5 FM 455: Diagnostics with LED display (S7-300, S7-400)

Error Display via the Error LEDs (S7-300, S7-400)

Where are the errors displayed?

The red LED INTF indicates an internal fault on the module.

The red LED EXTF indicates an external fault, for example fault on the connections.

If the yellow LED flashes, then the firmware has been deleted. This status can only occur in the case of faulty hardware or if the loading procedure of the firmware is aborted.

Which errors are displayed?

The following faults are indicated by the error LEDs lighting up:

Type of error	Diagnostic message	Possible cause	Correction
Internal errors	Module defective	Hardware error	Replace module
	Time monitoring (watchdog) tripped	Hardware error	Replace module
	EEPROM content is invalid	Failure of the supply voltage when configuring	Reconfigure module
External errors	Incorrect parameters in module	Incorrect parameters have been transferred to the module	Reconfigure module
	Errors with the analog inputs or analog outputs	Analog input hardware error	Replace module
		Analog input, wire breakage	Remedy wire break
		Analog input measuring range violation (underrange)	Check the measuring signal
		Analog input measuring range violation (overrange)	Check the measuring signal
		Analog output wire break	Remedy wire break
		Analog output short circuit	Eliminate the short-circuit
	Missing external auxiliary supply	Left-hand front connector not plugged 24 V supply missing	Plug in the left-hand front connector Restore 24 V supply

Diagnostic interrupt in the case of errors

All errors can trigger a diagnostic interrupt if you have enabled the diagnostic interrupt in the respective parameterization screen. From the diagnostic data records DS0 and DS1 you can see which errors have caused the LED to light up. The assignment of the diagnostic data records DS0 and DS1 is described in chapter "Triggering diagnostic interrupts (Page 1089)".

Triggering diagnostic interrupts (S7-300, S7-400)

What is a diagnostic interrupt?

If a user program is to respond to an internal or external fault, you can set a diagnostics interrupt parameter that interrupts the cyclic program of the CPU and calls the diagnostics interrupt OB, OB8 82.

Which events may trigger a diagnostic interrupt?

The list shows which events can trigger a diagnostic interrupt:

- The module has not been configured or is incorrectly configured
- Module defective
- Wire break in analog inputs (4 to 20 mA)

- High limit violated and low limit violated with analog inputs
- Load break and short circuit at analog outputs
- Wire break at thermocouples and Pt 100

Default setting

The diagnostic interrupt is blocked by default.

Enabling the diagnostic interrupt

You disable or enable the diagnostics interrupt for the module in the “Basic parameters” dialog box.

Responses to an interrupt-triggering event

The following happens when an event occurs that could trigger a diagnostic interrupt:

- The diagnostic information is stored in diagnostic data records DS0 and DS1.
- One or both error LEDs light up.
- The diagnostic interrupt OB (OB 82) is called.
- The diagnostic record DS0 is entered in the start information of the diagnostic interrupt OB.
- If there is no hardware fault, the module continues to control.

If no OB 82 is programmed, the CPU goes to STOP.

Diagnostic data record DS0 and DS1 (S7-300, S7-400)

Introduction

The information showing the event which has triggered a diagnostics interrupt is written to the diagnostics data records DS0 and DS1. The diagnostics data record DS0 comprises four bytes. DS1 comprises 27 bytes with the first four bytes being identical to DS0.

Read data record from the module

The diagnostic data record DS0 is automatically transferred to the start information when the diagnostic OB is called. There these four bytes are stored in the local data (byte 8 - 11) of the OB 82.

The diagnostic data record DS1 (and hence also the content of the DS0) can be read from the module using the "CH_DIAG_455" instruction. This only makes sense if an error is reported in a channel in the DS0.

How does the diagnostics text appear in the diagnostics buffer?

If you want to enter the diagnostics message in the diagnostics buffer, call the "WR_REC" instruction "Enter user-specific message in the diagnostics buffer" in the user program. The event number of the respective diagnostics report is specified on the input parameter EVENTN. The interrupt is entered in the diagnostics buffer with x = 1 as incoming and x = 0 as outgoing. Alongside the time of the entry, the diagnostics buffer also shows the respective diagnostics text that is specified in the "Meaning" column.

Assignment of the diagnostic data record DS0 in the start information

The following table shows the assignment of the diagnostic data record DS0 in the start information. Bits that are not listed are of no significance and are zero.

Table 3-7 Assignment of the diagnostic data record DS0

Byte	Bit	Meaning	Note	Event no.
0	0	BG fault	Is set for every diagnostic event	8:x:00
	1	Internal error	If set with all internal errors: • Time watchdog tripped • EEPROM content invalid; module continues without controlling and waits for a new configuration by the CPU • EPROM fault • ADC/DAC fault • Analog input hardware error	8:x:01
	2	External error	Is set with all external errors: • Missing external auxiliary supply • Faulty parameters • Analog input wire break (only area 4 to 20 mA) • Analog input measuring range violation (underrange) • Analog input measuring range violation (overrange) • Analog output wire break • Analog output short circuit	8:x:02
	3	Error in a channel	Refer to DS 1, as of byte 7 for a further breakdown	8:x:03
	4	Missing external auxiliary supply	24 V supply of the FM 455 failed	8:x:04
	6	EEPROM content is invalid	Missing supply voltage during a write operation in EEPROM. The module starts with default parameters.	8:x:03
	7	Faulty parameters	The module cannot make use of a parameter. Reason: Parameter unknown or invalid combination of parameters. See menu **Target system > Parameterization error display**	8:x:07
1	0 ... 3	Module class	Is always occupied with 8	–
	4	Channel-specific diagnosis	Is set when the module can provide additional channel information and a channel error exists (see DS 1 byte 7 to 12)	–

Byte	Bit	Meaning	Note	Event no.
2	0	User module incorrect / missing	The position (A, B, C, D) of the coding key does not agree with the parameter assignment in the FM 455	8:x:30
	3	Time monitoring (watchdog) tripped	Hardware error	8:x:33
3	2	EPROM fault	Module defective	8:x:42
	4	ADC/DAC fault	Module defective	8:x:44

Diagnostics Data Record DS 1 of the FM 455

The length of diagnostic data record DS1 is 27 bytes. The first 4 bytes are identical with the diagnostic data record DS0. The following table illustrates the assignment of the remaining bytes. Bits that are not listed are of no significance and are zero.

Table 3-8 Assignment of the Bytes 4 to 12 of the Diagnostics Data Record DS1 of the FM 455

Byte	Bit	Meaning	Note	Event no.
4	0 ... 7	Channel type	Is always occupied with 75H	–
5	0 ... 7	Length of diagnostics information	Is always occupied with 8	–
6	0 ... 7	Number of channels	17 always assigned (16 controllers + 1 reference channel)	–
7	0 … 7	Channel error vector	One bit each is assigned to channels 1 to 8	-
8	0 … 7	Channel error vector	One bit each is assigned to channels 9 to 16	-
9	0	Error on the reference channel		-

Byte	Bit	Meaning	Note		Event no.
10	0	Analog input hardware error	Channel-specific diagnostics channel 1	–	8:x:B0
	1	Parameterization error		The position (A, B, C, D) of the coding key does not agree with the parameter assignment in the FM 455	8:x:B1
	2	Analog input wire break (only area 4 to 20 mA)			8:x:B2
	3	Unused			8:x:B3
	4	Analog input measurement range low limit violated[1]			8:x:B4
	5	Analog input measurement range high limit violated[1]			8:x:B5
	6	Analog output wire break		Only for current output of continuous controller	8:x:B6
	7	Analog output short circuit		Only with the current output of the C controller	8:x:B7
11	0 ... 7	See Byte 10	Channel-specific diagnostics - channel 2		See above
12	0 ... 7	See Byte 10	Channel-specific diagnostics - channel 3		See above
13	0 ... 7	See Byte 10	Channel-specific diagnostics - channel 4		See above
14	0 ... 7	See Byte 10	Channel-specific diagnostics channel 5		See above
15	0 ... 7	See Byte 10	Channel-specific diagnostics channel 6		See above
16	0 ... 7	See Byte 10	Channel-specific diagnostics channel 7		See above
17	0 ... 7	See Byte 10	Channel-specific diagnostics channel 8		See above
18	0 ... 7	See Byte 10	Channel-specific diagnostics channel 9		See above
19	0 ... 7	See Byte 10	Channel-specific diagnostics channel 10		See above
20	0 ... 7	See Byte 10	Channel-specific diagnostics channel 11		See above
21	0 ... 7	See Byte 10	Channel-specific diagnostics channel 12		See above
22	0 ... 7	See Byte 10	Channel-specific diagnostics channel 13		See above
23	0 ... 7	See Byte 10	Channel-specific diagnostics channel 14		See above
24	0 ... 7	See Byte 10	Channel-specific diagnostics channel 15		See above
25	0 ... 7	See Byte 10	Channel-specific diagnostics channel 16		See above
26	0 ... 7	See Byte 10	Diagnosis for reference channel		See above
[1] refer to the chapter "Measuring transducer fault (Page 1095)"					

What you should particularly note

The FM 455 is supplied with voltage solely via the left front connector.

In the following cases, the CPU therefore recognizes the "Module removed / cannot respond":

- if the left front connector of the FM 455 is not connected
- if there is no 24 V supply voltage on the left front connector

Note

If, in the diagnostics buffer of the CPU, the entry "Module removed / cannot respond" is entered, check to see if the left front connector is connected and if the 24 V supply voltage of the FM 455 exists.

Additional information

For more information on the "RET_VALU" parameter, refer to List of RET_VALU messages (Page 1096) and the "Extended instructions" section of the help system.

Measuring transducer fault (S7-300, S7-400)

Measuring transducer fault

The following faults in measuring transducers can be recognized by the FM 455:

- Measuring range violation (underrange)
- Measuring range violation (overrange)
- Wire break (not with all measuring ranges)

Whenever one of these faults occurs, the group error ”External error” is set in the diagnostics data record DS0 and channel-specific error bits are set in the diagnostics data record DS1 (refer to the tables in the previous section). When these errors disappear, the respective bits are reset.

The following table shows at which limits in the individual measuring ranges the error bits are set or reset.

Measuring range	Error bit Low Measuring Range Limit violated at ...	Error bit Measuring Range high limit violated at ...	Error bit Wire Break Display
	DS1: Byte 10 to 26, bit 4	DS1: Byte 10 to 26, bit 5	DS1: Byte 10 to 26, bit 2
0 to 20 mA	< - 3.5 mA	> 23.5 mA	–
4 to 20 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at < 3.8 mA	> 22.8 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at < 3.8 mA
0 V to 10 V	<- 1.175 V	> 11.75 V	–
Pt 100 (-200 to 850 °C) (-328 to 1562 °F)	< 30.82 mV	> 650.46 mV	Yes, parameter can be set, Display: 650.46 mV

Measuring range	Error bit Low Measuring Range Limit violated at ...	Error bit Measuring Range high limit violated at ...	Error bit Wire Break Display
	DS1: Byte 10 to 26, bit 4	DS1: Byte 10 to 26, bit 5	DS1: Byte 10 to 26, bit 2
Pt 100 (-200 to 556 °C) (-328 to 1032 °F)	< 30.82 mV	> 499.06 mV	Yes, parameter can be set, Display: 499.06 mV
Pt 100 (-200 to 130 °C) (-328 to 264 °F)	< 30.82 mV	> 254.12 mV	Yes, parameter can be set, Display: 254.12 mV
Thermocouple element type B	< 0 mV	> 13.81 mV	Yes, parameter can be set, the last value will be stored.
Thermocouple element type J	< - 8.1 mV	> 69.54 mV	Yes, parameter can be set, the last value will be stored.
Thermocouple element type K	< - 6.45 mV	> 54.88 mV	Yes, parameter can be set, the last value will be stored.
Thermocouple element type R	< - 0.23 mV	> 21.11 mV	Yes, parameter can be set, the last value will be stored.
Thermocouple element type S	< - 0.24 mV	> 18.7 mV	Yes, parameter can be set, the last value will be stored.
Free thermocouple element	< low input value of the polyline	> high input value of the polyline	Yes, parameter can be set, the last value will be stored.

List of RET_VALU messages (S7-300, S7-400)

RET_VALU messages

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
7000	28672	-32624	First call with REQ=0: no data transmission active; BUSY has the value 0.
7001	28673	-32624	First call with REQ=1: data transmission initiated; BUSY has the value 1.
7002	28674	-32624	Interim call (REQ irrelevant). Data transfer already active; BUSY has the value 1.
8090	32912	-32624	Specified logical base address invalid: There is no assignment in the SDB1/SDB2x, or it is not a base address.
80A0	32928	-32608	Negative acknowledgment when reading from the module. Module was removed during the read operation or the module is defective.
80A1	32929	-32607	Negative acknowledgment when writing to the module. Module was removed during the write operation or the module is defective.
80A2	32930	-32606	Protocol error at layer 2
80A3	32931	-32605	Protocol error involving user interface/user
80A4	32932	-32604	Communication bus error

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80B1	32945	-32591	Incorrect length specification. FM_TYPE parameter in channel DB not set correctly for the module in use.
80B2	32946	-32590	The configured slot is not being used.
80B3	32947	-32589	Actual module type is not match configured module type.
80C0	32960	-32576	Module data not ready for reading.
80C1	32961	-32575	Data of a write job of the same type have not yet been processed by the module.
80C2	32962	-32574	The module is currently processing the maximum possible number of jobs.
80C3	32963	-32573	Required resources (memory etc.) currently occupied.
80C4	32964	-32572	Communication error
80C5	32965	-32571	Distributed I/O not available.
80C6	32966	-32570	Priority class abort (restart or background).
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read off the DB. (Write job)
8532	34098	-31438	DB number of the parameter DB too high (Write job)
853A	34106	-31430	Parameter DB not present. (write job)
8544	34116	-31420	Error at n-th (n > 1) read access to a DB after an error has occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written to the DB. (Read job)
8730	34608	-30928	Parameter DB in the CPU is write protected. The data cannot be written to the DB (read job)
8732	34610	-30926	DB number of the parameter DB too high (Read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error at n-th (n > 1) write access to a DB after an error has occurred. (Read job)
80ff	33023	-32513	Incorrect index specification with block FMT_PAR
Errors 80A2..80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time. Messages of the 7xxx form indicate temporary operating states of communication.			

3.5.7.6 Example projects for FM 455 (S7-300, S7-400)

3.5.8 Using ET 200S 2PULSE (S7-300, S7-400)

3.5.8.1 2PULSE terminal assignment (S7-300, S7-400)

Wiring Rules

The cables (terminals 1 and 2 and terminals 5 and 6) must be shielded. The shield must be supported at both ends. To do this, use the shield contact (see the appendix of the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal assignment of the 2PULSE

The following table shows the terminal assignment for 2PULSE.

View	Terminal Assignment	Meaning
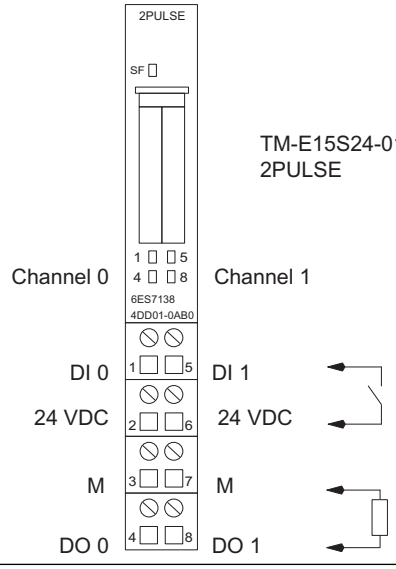		Channel 0: Terminal 1 to 4 Channel 1: Terminal 5 to 8 24V DC: Encoder supply M: Chassis ground DI: Input Signal DO: Output signal (max. 2 A per channel)

3.5.8.2 Operating modes and functions (S7-300, S7-400)

Overview (S7-300, S7-400)

Principle

The 2PULSE has two channels. You can select a separate mode for each channel. The operating mode is specified during the configuration. The mode that has been assigned parameters can then no longer be changed with your control program.

You can select various modes for each channel:

- Pulse output (Page 1100)
- Pulse width modulation (Page 1106)
- Pulse train (Page 1116)
- On/Off delay (Page 1121)
- Frequency output (Page 1129) (as of 6ES7138-4DD01-0AB0)

In addition to the set mode, the 2PULSE also has the following functions:

- Current measurement (Page 1135) in the "Pulse width modulation" and "Pulse train" modes (as of 6ES7138-4DD01-0AB0)
- Direct control of the DO digital output (Page 1136) by means of your control program; can be controlled separately for each channel.
- Error detection/diagnostics (Page 1137); the 2PULSE recognizes the errors for each channel separately.
- Parallel connection of both channels (Page 1139) to achieve a higher output current (as of 6ES7138-4DD01-0AB0).
- Reaction to CPU/Master STOP (Page 1141); the 2PULSE recognizes the CPU/Master STOP for both channels and responds in accordance with your parameter assignment.

The following figure shows the operating principle of the 2PULSE.

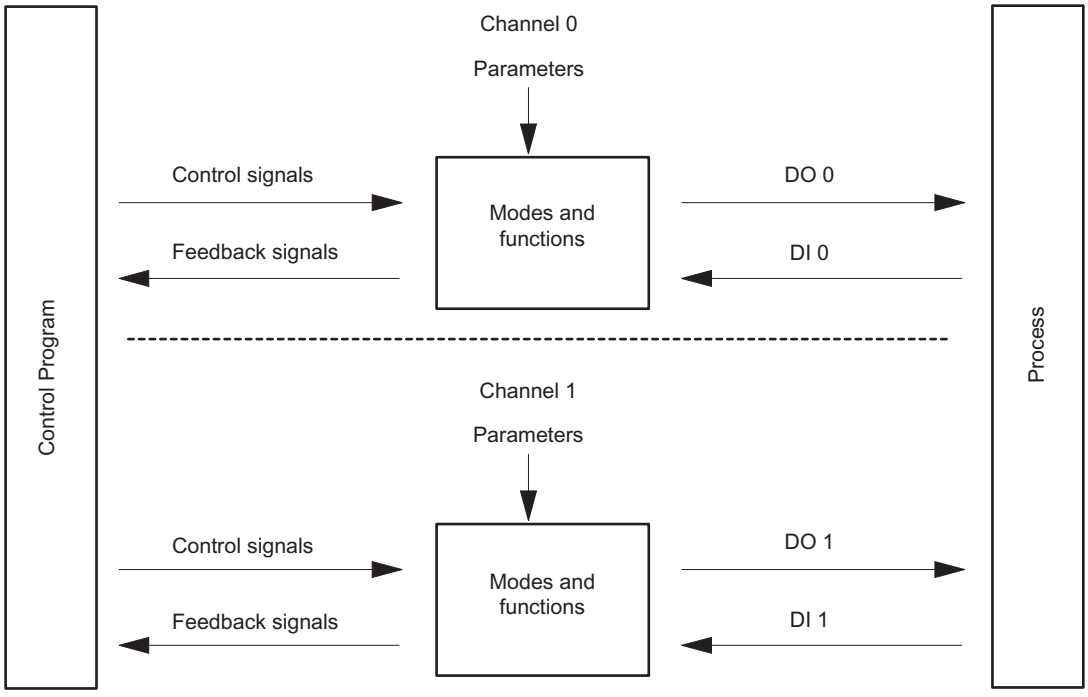

Interfaces to the Control Program and Process

To run the modes and execute functions, the 2PULSE has as an interface to the process of a digital input and a digital output for each channel (DI 0, DO 0 for Channel 0 and DI 1, DO1 for Channel 1).

You can modify and monitor the modes and functions with your control program using control signals (Page 1146) and feedback signals (Page 1149).

Parameters are assigned to the various modes.

You will find the following in the descriptions on operating modes and functions:

- The relevant parameters
- The control and feedback signals

The description of the modes and functions applies to both channels and the channels are therefore not referred to separately in the description.

"Pulse output" operating mode (S7-300, S7-400)

"Pulse output" operating mode (S7-300, S7-400)

Pulse output

For the pulse duration you set, the 2PULSE outputs a pulse at the DO digital output (output sequence) on expiration of the set on-delay (Page 1144).

The following figure shows the block diagram for the operating mode "Pulse output".

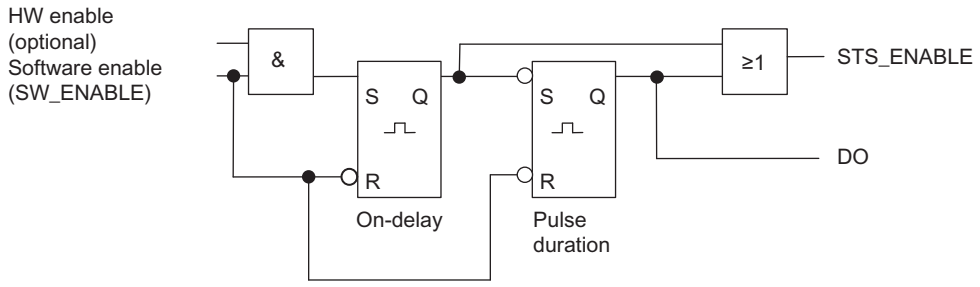

Starting the output sequence

You must always issue the enable for the output sequence by means of the software enable (Page 1146) (SW_ENABLE 0 → 1; MANUAL_DO = 0) in your control program.

The feedback bit (Page 1149)ACK_SW_ENABLE indicates the software enable pending at the 2PULSE.

You can also set the DI digital input of the 2PULSE as a hardware enable (HW enable) with the "DI function (Page 1144)" parameter.

If you want to work with the software enable and hardware enable at the same time, when the software enable has been issued, the output sequence starts at the first positive edge of the hardware enable. Further positive edges of the hardware enable during the current output sequence are ignored by the 2PULSE. When the software enable has been issued, a positive edge of the hardware enable is enough to start the next output sequence.

When the enable is issued (positive edge), the on-delay is started and the STS_ENABLE set. On expiration of the on-delay, the pulse is output with the set pulse duration. The output sequence (Page 1134) finishes with the end of the pulse; STS_ENABLE is cleared.

If you make a prohibited change to the pulse duration during runtime, the signal (Page 1149)ERR_PULS indicates a pulse output error. You will then have to restart the output sequence.

The next time the output sequence starts, the 2PULSE deletes the ERR_PULS feedback bit.

Pulse Diagram

The following figure shows an output sequence for the operating mode "Pulse output".

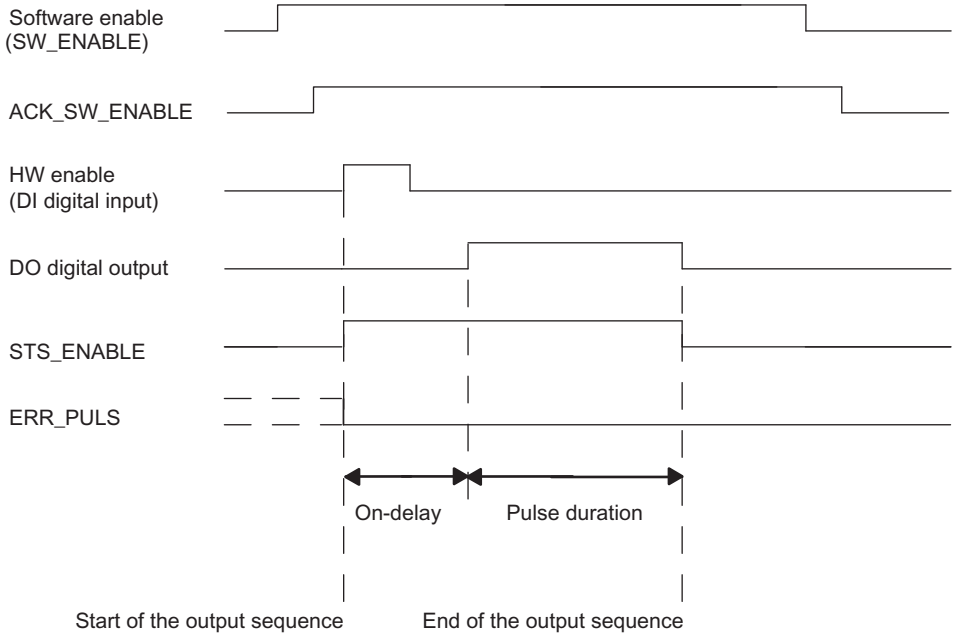

Canceling the output sequence

Deleting the software enable (SW_ENABLE = 0) during the on-delay or the pulse output cancels the output sequence, and STS_ENABLE and the DO digital output are canceled.

You will then have to restart the output sequence.

Truth table

The table below shows how the 2PULSE responds to certain signals.

Software enable SW_ENABLE	Hardware enable (digital input DI)	DO Digital Output	STS_ENABLE	Output Sequence
1	0 → 1	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Start
0 → 1	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	Any status	0	0	Terminate
1	Any status	Previous status remains		-
0 → 1	0	0	0	-

Setting Times Using a Time Base

By means of the time base (Page 1144) that can be assigned parameters, you can select the resolution and range of the pulse duration and the on-delay.

Time base = 0.1 ms:	You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
Time base = 1 ms:	You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Setting and Changing the Pulse Duration

Set the pulse duration (Page 1145) directly in your control program as a numerical value between 0 and 65535.

Pulse duration = Time basis x Specified numerical value

If you change the pulse duration when an output sequence is running, the time already output will be subtracted from the new pulse duration and the pulse will continue to be output.

Reducing the Pulse Duration

If you have reduced the pulse duration to a time that is shorter than the time already output, the output sequence is terminated, STS_ENABLE and the DO digital output are deleted, and the ERR_PULS status bit is set. At the next output sequence, the ERR_PULS status bit is cleared.

Set the on-delay and change when using the short control interface

You specify the on-delay (Page 1144) as a value between 0 and 65535 in the parameters.

Configured on-delay = Time base x specified numerical value

Using the factor for the on-delay, you can adjust the time that has been assigned parameters in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

On-delay = Factor x 0.1 x configured on-delay

If you change the on-delay factor during the output sequence, the new on-delay is activated at the next output sequence.

Isochronous mode (as of 6ES7138-4DD01-0AB0)

In the "Pulse output" operating mode, isochronous mode does not have any influence on the functionality.

Parameters of the "Pulse output" operating mode (S7-300, S7-400)

Parameters of the "Pulse output" operating mode

The table below shows the parameters of the operating mode "Pulse output".

Parameter	Meaning	Value range	Default
Operating mode (Page 1143)	Set the "Pulse output" operating mode.	• Pulse output • Pulse width modulation • Pulse train • On/Off-delay • Frequency output	Pulse width modulation
Time base (Page 1144)	Using the time base, select the resolution and range of the pulse duration and the on-delay.	• 0.1 ms • 1 ms	0.1 ms
Function DI (Page 1144)	You can use the DI digital input as an input or as a hardware enable.	• Input • HW enable	Input
On-delay (Page 1144) [1]	The time from the start of the output sequence to the output of the pulse. You can change the on-delay in your control program using the "On-delay factor".	0 to 65535	0
[1] Only if the short control interface (Page 1146) is used If the long control interface is used, enter this value directly in the control interface.			

Control and feedback signals of the "Pulse output" operating mode (S7-300, S7-400)

Control and feedback signals of the "Pulse output" operating mode

The table below shows the control and feedback signals of the operating mode "Pulse output".

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Control signals when the short control interface is used (8 bytes)				
Software enable SW_ENABLE	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 2: Bit 0	Byte 6: Bit 0
Pulse duration	The time that is set for the DO digital output on expiration of the on-delay.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the 2PULSE will not output a pulse.	Word 0	Word 4

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
On-delay factor	The on-delay that has been assigned parameters can be changed before the start of the output sequence: On-delay = Factor x 0.1 x configured on-delay	0 to 255 If the on-delay <0.2 ms or if factor = 0, the effective on-delay is = 0. If there is an on-delay > 65.535 s, the on-delay is limited to 65.535 s.	Byte 3	Byte 7
Control signals when the long control interface is used (12 bytes) (as of 6ES7138-4DD01-0AB0)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 4: Bit 0	Byte 10: Bit 0
Pulse duration	The time that is set for the DO digital output on expiration of the on-delay.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the 2PULSE will not output a pulse.	Word 0	Word 6
On-delay	The on-delay can be changed before the start of the output sequence.	0 to 65535 On-delay = Time basis x Specified numerical value	Word 2	Word 8
Feedback signals				
STS_ENABLE	Indicates an output sequence is running.	0 = Pulse output blocked 1 = Pulse output running	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1
STS_DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2
ACK_SW_ENABLE	Indicates the status of SW_ENABLE.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 0: Bit 3	Byte 4: Bit 3
ERR_PULS	Indicates a pulse output error.	0 = no pulse output error 1 = pulse output error	Byte 0: Bit 4	Byte 4: Bit 4
SEQ_CNT (as of 6ES7138-4DD01-0AB0)	Is incremented after completion of an output sequence.	Without HW enable: 0 … 1 With HW enable: 0 … 15	Byte 1: Bit 0 to 3	Byte 5: Bit 0 … 3
Reserved	–	0	Word 2	Word 6

Input and output signals of "Pulse output" operating mode (S7-300, S7-400)

Input and output signals of "Pulse output" operating mode

The table below shows the input and output signals of the operating mode "Pulse output".

Input and output signal	Meaning	Value range	Channel 0 terminal	Channel 1 terminal
Input signal				
HW enable	You can select the HW enable with the "Function DI" parameter. The signal of the digital input DI is then interpreted by the 2PULSE at the start of the output sequence.	0 = HW enable cleared 1 = HW enable issued 0 → 1 = Start of the output sequence; dependent on the software enable (SW_ENABLE)	1	5
Output signal				
Pulse at the DO digital output	A pulse is output at the DO digital output for the set pulse duration.	0 = no pulse 1 = pulse	4	8

"Pulse width modulation" operating mode (S7-300, S7-400)

"Pulse width modulation" operating mode (S7-300, S7-400)

Pulse width modulation

You specify an output value for the 2PULSE. The 2PULSE generates continuous pulses on this basis. The output value determines the pulse/interpulse ratio within a period (pulse-width modulation). The period can be adjusted.

The pulse train is output on expiration of the assigned on-delay (Page 1144) on the DO digital output of the 2PULSE (output sequence).

The following figure shows the block diagram for the operating mode "Pulse-width modulation".

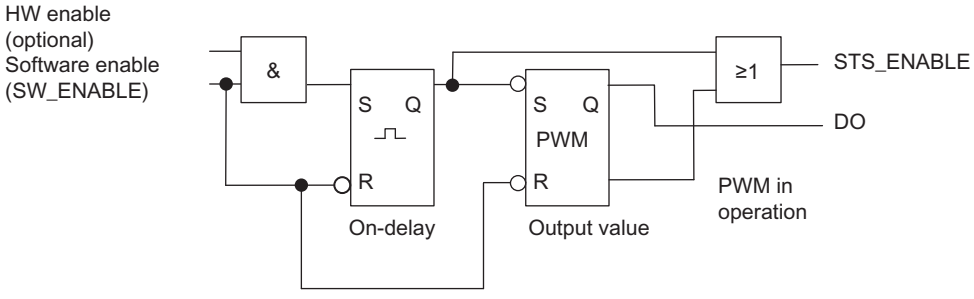

Starting the output sequence

You must always issue the enable for the output sequence by means of the software enable (Page 1146) (SW_ENABLE 0 → 1; MANUAL_DO = 0) in your control program.

The feedback bit (Page 1149)ACK_SW_ENABLE indicates the software enable pending at the 2PULSE.

You can also set the digital input DI of the 2PULSE as a HW enable with the "Function DI (Page 1144)" parameter. In isochronous mode (as of 6ES7138-4DD01-0AB0), configuration of a hardware enable has no effect.

If you want to work with the software enable and hardware enable at the same time, when the software enable has been issued, the output sequence starts with the first positive edge of the hardware enable. Further positive edges of the hardware enable during the current output sequence are ignored by the 2PULSE.

When the enable is issued (positive edge), the on-delay is started and the STS_ENABLE set. The pulse train is output on expiration of the on-delay. The output sequence runs continuously as long as SW_ENABLE is set.

Pulse Diagram

The following figure shows an output sequence for the operating mode "Pulse-width modulation".

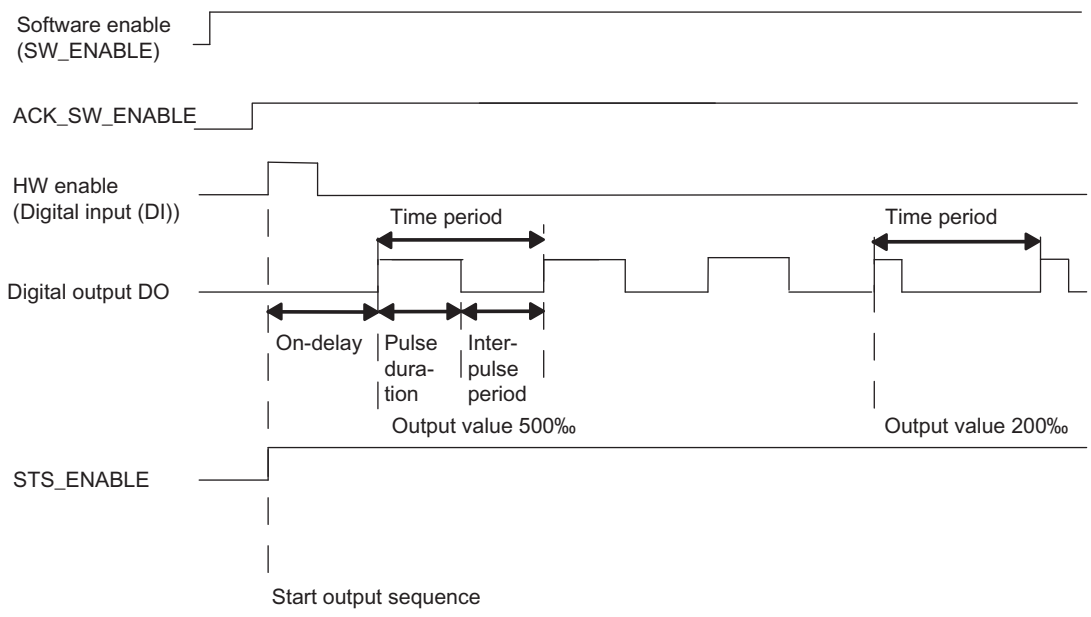

Canceling the output sequence

Deleting the software enable (SW_ENABLE = 0) during the on-delay or the pulse output cancels the output sequence, and STS_ENABLE and the DO digital output are canceled.

You will then have to restart the output sequence.

Truth table

The table below shows how the 2PULSE responds to certain signals.

Software enable SW_ENABLE	Hardware enable (digital input DI)	Digital output DO	STS_ENABLE	Output sequence
1	0 → 1	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0 → 1	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	Any status	0	0	Canceling
1	Any status	Previous status remains		-
0 → 1	0	0	0	-

Minimum Pulse Duration and Minimum Interpulse Period

The minimum pulse duration and minimum interpulse period are superimposed on the proportional output characteristic.

You assign the minimum pulse duration and minimum interpulse period using the "Minimum pulse duration (Page 1145)" parameter; they always have the same value.

A pulse duration calculated by the 2PULSE that is shorter than the minimum pulse duration is suppressed.

A pulse duration calculated by the 2PULSE that is longer than the period minus the minimum interpulse period is set at 1000‰.

The following figure shows the modulation of the pulse duration.

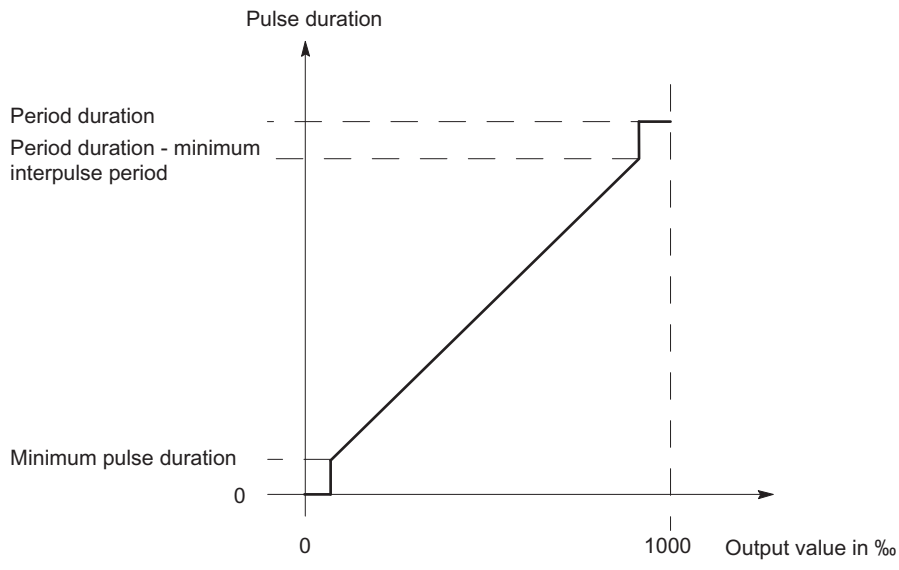

Setting Times Using a Time Base

You use the assignable time base (Page 1144) to select the resolution and range of the period, the minimum pulse duration and the on-delay.

Time base = 0.1 ms:	You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
Time base = 1 ms:	You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Setting and Changing the Output Value

You select the range of the output value with the "Output format (Page 1144)" parameter.

- Output format "Per mill": Value range between 0 and 1000
  The 2PULSE uses this specified output value to calculate the pulse duration:
  Pulse duration = (output value/1000) x time period.
- "S7 analog output" output format: Value range between 0 and 27648
  The 2PULSE uses this specified output value to calculate the pulse duration:
  Pulse duration = (output value/27648) x time period.

You set the output value directly using your control program. The new output value is applied at the next rising edge of the output.

Setting the time period

Depending on the configuration, the time period is specified as a parameter value or in the control interface. The table below shows when a specific value is used.

For isochronous mode (as of 6ES7138-4DD01-0AB0), you have to specify the time period when parameters are assigned. In contrast to non-isochronous mode, you cannot change the time period later during runtime.

	Configuration with ...	
	Short control interface	Long control interface
Isochronous mode	Time period = "Time period" parameter value The value from the control interface is ignored.	
Non-isochronous mode	Time period = "Time period" parameter value × "Time period factor" from control interface	Time period = "Time period" setpoint from control interface The "Time period" parameter value is ignored.

Time period at configuration with short control interface

You specify the time period (Page 1145) in the parameters as a numerical value depending on the time base:

- With time base 1 ms: from 1 to 65535
- With time base 0.1 ms: from 2 to 65535

Configured time period = Time base x specified numerical value

Using the factor for the period duration, you can adjust the time that has been assigned parameters in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

Time period = Factor x 0.1 x configured time period

The new setpoints for the time period are applied with the next rising edge of the output.

Note

If the time period changes in runtime, a brief minor increase in the time period can result (approx. 10 µs).

Note

A change in the time period using the control interface has no effect in isochronous mode (as of 6ES7138-4DD01-0AB0).

Setting the Minimum Pulse Duration and Minimum Interpulse Period

You specify the minimum pulse duration and the minimum interpulse period as a numerical value between 0 and 65535 using the "Minimum pulse duration (Page 1145)" parameter:

Configured minimum pulse duration/minimum pause between pulses = Time base x specified numerical value

Setting the On-Delay

You specify the on-delay (Page 1144) as a value between 0 and 65535 in the parameters.

Configured on-delay = Time base x specified numerical value

When the long control interface is used, the factor for the on-delay can be used to adjust the configured time in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

On-delay = Factor x 0.1 x configured on-delay

Current measurement (as of 6ES7138-4DD01-0AB0)

An important application for current measurement in the "Pulse width modulation" operating mode is the current regulation of a proportional valve. This allows you to compensate for thermal effects at the valve, for example.

Additional information is available under "Function: Current measurement (as of 6ES7138-4DD01-0AB0) (Page 1135)".

Isochronous mode in the operating mode "Pulse width modulation" (S7-300, S7-400)

Isochronous mode

Note

Isochronous mode is available with 2PULSE as of article number 6ES7138-4DD01-0AB0.

The following particular aspects apply to isochronous mode in the "Pulse width modulation" operating mode:

- The output always begins without an on-delay, a configured on-delay is ignored.
- The output always begins after the software enable has been issued, a configured hardware enable is ignored.
- A change of the time period during runtime is not possible. Corresponding specifications via the control interface are ignored.

In isochronous mode, the output sequence is synchronized with the moment T_o. The time period is coordinated to the DP cycle. It is possible that your defined time period cannot be implemented exactly.

The configured value of the time period is adjusted by the 2PULSE to the DP cycle based on a calculation algorithm. The calculation is performed to minimize the difference between the specified and calculated time period. In the most unfavorable case, the deviation amounts to half the DP cycle. The table below shows examples.

DP cycle time T_{DP} [1]	Specified time period $T_{Setpoint}$	Calculated time period T_{Actual} [1]
10 ms	5 ms	5 ms
10 ms	2 ms	2 ms
10 ms	6 ms	5 ms
10 ms	16 ms	20 ms

[1] The calculated time period and the DP cycle time always have an integer ratio.

The correlation between the and the DP cycle is represented graphically below. The pulse-pause ratio is shown in the examples with 50% each. For additional information, refer to the

Isochronous Mode (http://support.automation.siemens.com/WW/view/en/15218045) function manual.

- The time period is equal to the DP cycle

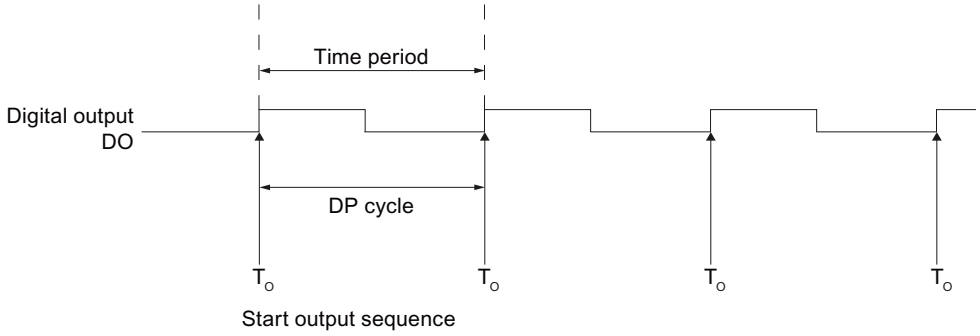

- The time period is less than the DP cycle

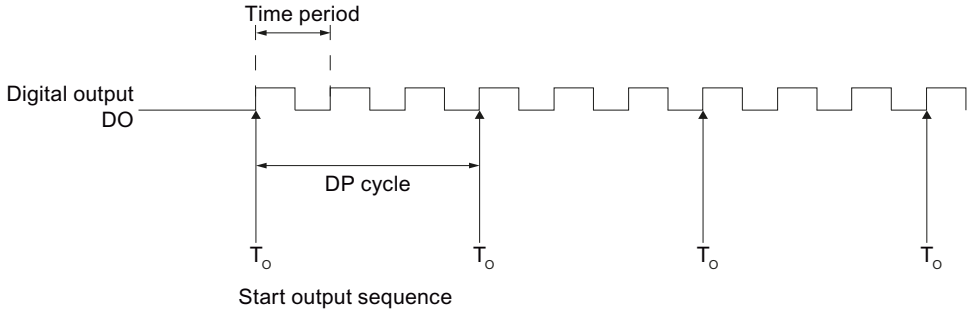

- The time period is greater than the DP cycle

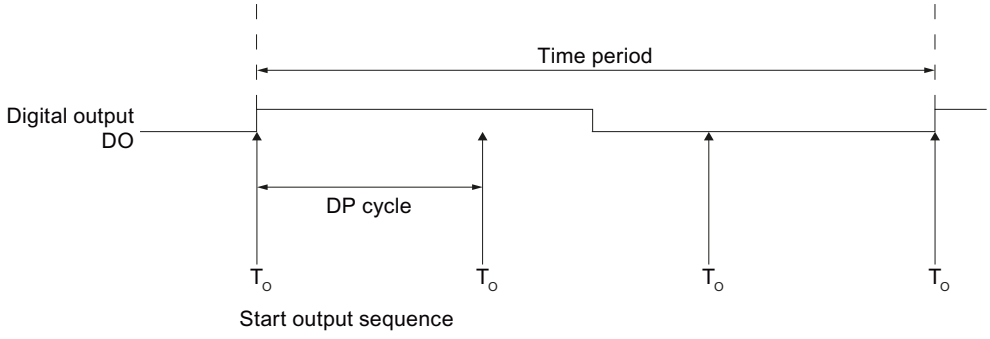

Parameters of the "Pulse width modulation" operating mode (S7-300, S7-400)

Parameters of the "Pulse width modulation" operating mode

The table below shows the parameters of the operating mode "Pulse width modulation".

Parameter	Meaning	Value Range	Default
Operating mode (Page 1143)	Set the "Pulse width modulation" operating mode.	• Pulse output • Pulse-width modulation • Pulse train • On/Off-delay • Frequency output	Pulse width modulation
Time base (Page 1144)	Use the time base to select the resolution and the value range of the time period, the minimum pulse duration, and the on-delay.	• 0.1 ms • 1 ms	0.1 ms
Output format (Page 1144)	Choose between the Per mil, and the S7 analog output formats, depending on which output value resolution you need.	• Per mill • S7 analog output	Per mill
Function DI (Page 1144)	You can use the digital input DI as an input or as a HW enable.	• Input • HW enable	Input
On-delay (Page 1144)	The time from the start of the output sequence to the output of the pulse train.	0 to 65535	0
Minimum pulse duration (Page 1145)	Minimum pulse duration and minimum interpulse period: Enter the response time of the actuator connected on your DO digital output.	0 to 65535	0
Time period (Page 1145) [1]	Setpoint of the time period When specifying the time period take into account the set minimum pulse duration and the response time of your actuator that is connected to the digital output DO. When the short control interface is used, you can use the "Time period factor" to change the time period in your control program in non-isochronous mode.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535	20000 → 2 s

[1] Read the information provided above on "Setting the time period (Page 1106)" and on "Isochronous mode (Page 1111)".

Control and feedback signals of the "Pulse width modulation" operating mode (S7-300, S7-400)

Control and feedback signals of the "Pulse width modulation" operating mode

The table below shows the control and feedback signals of the operating mode "Pulse width modulation".

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Control signals when the short control interface is used (8 bytes)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 2: Bit 0	Byte 6: Bit 0
Output value	The value that is output in pulse-width modulated format on the digital output DO.	Depending on the output format: • Per mill: 0 ... 1000 • S7 analog output: 0 to 27648 If you enter an output value > 1000 or 27648, the 2PULSE limits this to 1000 or 27648.	Word 0	Word 4
Period duration factor	You can change the assigned period: Time period = Factor x 0.1 x configured time period	Factor: 0 to 255 Period duration: 2 × Minimum pulse duration up to 65.635 s. If a period duration of < 2 x minimum pulse duration occurs or < 2 µs or if Factor = 0, the effective period = 2 x minimum pulse duration, but at least 200 µs. If there is a period > 65.535 s, it is limited to 65.535 s.	Byte 3	Byte 7
Control signals when the long control interface is used (12 bytes) (as of 6ES7138-4DD01-0AB0)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 4: Bit 0	Byte 10: Bit 0
Output value	The value that is output in pulse-width modulated format on the digital output DO.	Depending on the output format: • Per mill: 0 ... 1000 • S7 analog output: 0 to 27648 If you enter an output value > 1000 or 27648, the 2PULSE limits this to 1000 or 27648.	Word 0	Word 6
Time period [1)]	Setpoint of the time period in non-isochronous mode	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535	Word 2	Word 8

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
On-delay factor	The on-delay that has been assigned parameters can be changed before the start of the output sequence: On-delay = Factor x 0.1 x configured on-delay	0 to 255 If the on-delay <0.2 ms or if factor = 0, the effective on-delay is = 0. If there is an on-delay > 65.535 s, the on-delay is limited to 65.535 s.	Byte 5	Byte 11
Feedback signals				
STS_ENABLE	Indicates an output sequence is running.	0 = pulse output blocked 1 = pulse output running	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1
STS_DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2
ACK_SW_ENABLE	Indicates the status of SW_ENABLE.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 0: Bit 3	Byte 4: Bit 3
SEQ_CNT (as of 6ES7138-4DD01-0AB0)	No function in this operating mode.	–	Byte 1: Bit 0 to 3	Byte 5: Bit 0 … 3
Current measurement (as of 6ES7138-4DD01-0AB0)	Current at the DO as a SIMATIC S7 analog value.	0 to 27648 (overrange up to 32767)	Word 2	Word 6
[1] Read the information provided on "Setting the time period (Page 1106)".				

Input and output signals of the "Pulse width modulation" operating mode (S7-300, S7-400)

Input and output signals of the "Pulse width modulation" operating mode

The table below shows the input and output signals of the operating mode "Pulse width modulation".

Input and Output Signals	Meaning	Value range	Channel 0 terminal	Channel 1 terminal
Input signal				
HW enable	You can select the HW enable with the "Function DI" parameter. The signal of the digital input DI is then interpreted by the 2PULSE at the start of the output sequence.	0 = HW enable cleared 1 = HW enable issued 0 → 1 = Start of the output sequence; dependent on the software enable	1	5
Output signal				
Pulse train on the digital output DO	The pulse train is output on the digital output DO.	0 = no pulse 1 = pulse	4	8

"Pulse train" operating mode (S7-300, S7-400)

"Pulse train" operating mode (S7-300, S7-400)

Definition

The 2PULSE outputs the number of pulses you specified as a pulse train at the DO digital output on expiration of the set on-delay (Page 1144) (output sequence). The period duration and pulse duration of the pulses can be adjusted.

The following figure shows the block diagram for the operating mode "Pulse train".

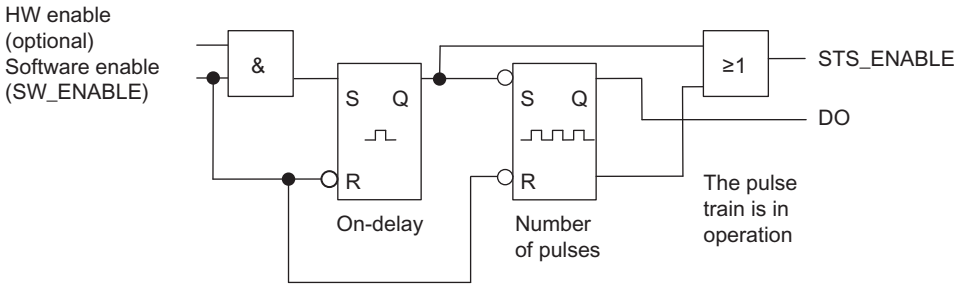

Starting the Output Sequence

You must always issue the enable for the output sequence by means of the software enable (Page 1146) (SW_ENABLE 0 → 1; MANUAL_DO = 0) in your control program. The feedback bit (Page 1149)ACK_SW_ENABLE indicates the software enable pending at the 2PULSE.

You can also set the digital input DI of the 2PULSE as a HW enable with the "Function DI (Page 1144)" parameter.

If you want to work with the software enable and hardware enable at the same time, when the software enable has been issued, the output sequence starts with the first positive edge of the hardware enable. Further positive edges of the hardware enable during the current output sequence are ignored by the 2PULSE. When the software enable has been issued, a positive edge of the hardware enable is enough to start the next output sequence.

When the enable is issued (positive edge), the on-delay is started and the STS_ENABLE set. On expiration of the on-delay, the pulse train is output with the set number of pulses. The output sequence (Page 1134) finishes as soon as the last pulse has been output; STS_ENABLE is cleared.

If you make a prohibited change to the number of pulses during runtime, the signal (Page 1149)ERR_PULS indicates a pulse output error.

At the next output sequence, the 2PULSE deletes the ERR_PULS feedback bit.

Pulse diagram

The following figure shows an output sequence for the operating mode "Pulse train".

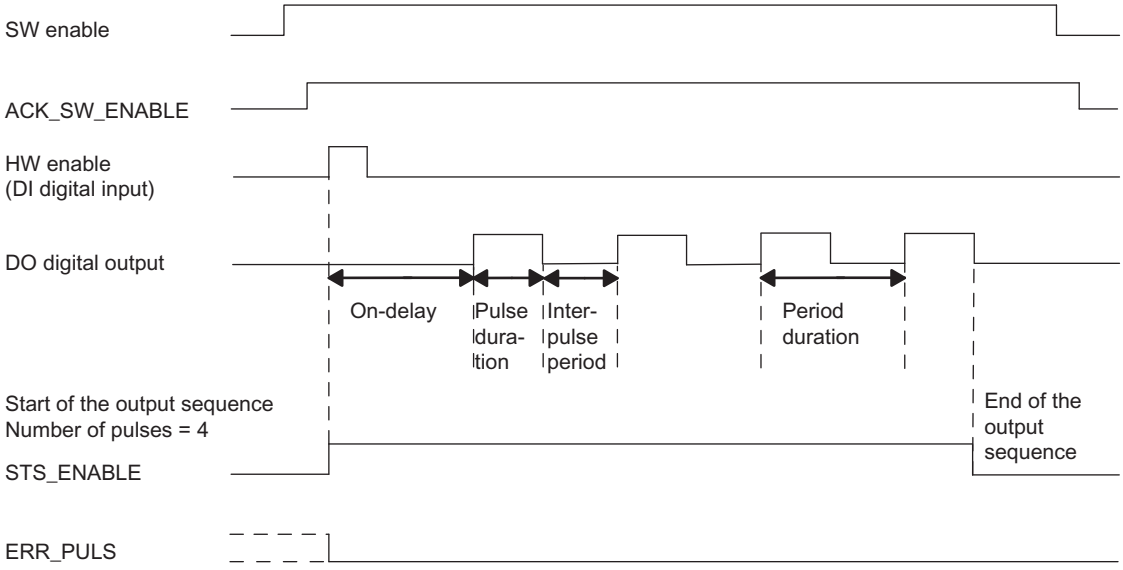

Canceling the output sequence

Deleting the software enable during the on-delay or the pulse train terminates the output sequence, and STS_ENABLE and the DO digital output are deleted.

You will then have to restart the output sequence.

Truth table

The table below shows how the 2PULSE responds to certain signals.

Software enable SW_ENABLE	Hardware enable (digital input DI)	Digital output DO	STS_ENABLE	Output sequence
1	0 → 1	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0 → 1	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	Any status	0	0	Canceling
1	Any status	Previous status remains		-
0 → 1	0	0	0	-

Setting Times Using a Time Base

By means of the time base (Page 1144) that can be assigned parameters, you can select the resolution and range of the period, pulse duration and on-delay.

Time base = 0.1 ms:	You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
Time base = 1 ms:	You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Setting and Changing the Number of Pulses

Set the number of pulses (Page 1146) directly as a numerical value between 0 and 65535 with your control program.

If you change the number of pulses on expiration of the on-delay, the new value takes effect immediately:

- If you have increased the number of pulses, the new, higher number of pulses is output.
- If you have reduced the number of pulses, and if the lower number of pulses has already been output, the output sequence is terminated, STS_ENABLE and the DO digital output are deleted, and ERR_PULS is set. At the next output sequence, ERR_PULS is cleared.

Set the time period and change when using the short control interface

Set the period duration (Page 1145) as a value between 2 and 65535 in the parameters:

Configured time period = Time base x specified numerical value

Using the factor for the period, you can adjust the assigned time in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

Time period = Factor x 0.1 x configured time period

If you change the factor during the output sequence, the new period duration will take effect at the start of the next output sequence.

Setting the Pulse Duration

Set the pulse duration (Page 1145) as a numerical value between 1 and 65535 with the "Pulse duration" parameter:

Configured pulse duration = Time base x Specified numerical value

When the long control interface is used, you can set the ratio of the pulse duration to the interpulse period:

Pulse-pause ratio = (Setpoint / 255) × time period

Setting the On-Delay

You specify the on-delay (Page 1144) as a value between 0 and 65535 in the parameters.

Configured on-delay = Time base x specified numerical value

Isochronous mode (as of 6ES7138-4DD01-0AB0)

In the "Pulse train" operating mode, isochronous mode does not have any influence on the functionality.

Current measurement (as of 6ES7138-4DD01-0AB0)

Additional information is available under "Function: Current measurement (as of 6ES7138-4DD01-0AB0) (Page 1135)".

Parameters of the "Pulse train" operating mode (S7-300, S7-400)

Parameters of the "Pulse train" operating mode

The table below shows the parameters of the operating mode "Pulse train".

Parameter	Meaning	Value range	Default
Operating mode (Page 1143)	Set the "Pulse train" operating mode.	• Pulse output • Pulse width modulation • Pulse train • On/Off-delay • Frequency output	Pulse width modulation
Time base (Page 1144)	Using the time base, select the resolution and range of the period duration, pulse duration, and on-delay.	• 0.1 ms • 1 ms	0.1 ms
Function DI (Page 1144)	You can use the digital input DI as an input or as a HW enable.	• Input • HW enable	Input
On-delay (Page 1144)	The time from the start of the output sequence to the output of the pulse train.	0 to 65535	0
Pulse duration (Page 1145) [1]	Pulse duration: Enter the response time of the actuator connected on your DO digital output.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the 2PULSE sets the pulse duration to 0.2 ms or 1 ms.	10000 → 1 s
Time period (Page 1145) [1]	The period duration should always be a multiple of the response time of the actuator connected to the DO digital output. Define the period duration according to the required repetition rate of the pulses. You can change the time period in your control program using the "Time period factor".	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535	20000 → 2 s

[1] Only if the short control interface (Page 1146) is used
If the long control interface is used, enter this value directly in the control interface.

Control and feedback signals of the "Pulse train" operating mode (S7-300, S7-400)

Control and feedback signals of the "Pulse train" operating mode

The table below shows the control and feedback signals of the operating mode "Pulse train".

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Control signals when the short control interface is used (8 bytes)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set 0 → 1 = Start of the output sequence; may be dependent on the HW enable	Byte 2: Bit 0	Byte 6: Bit 0
Number of pulses	Number of pulses that are output at the DO digital output on expiration of the on-delay.	0 to 65535 If the number of pulses is 0, the 2PULSE does not output any pulses. The output sequence is terminated with ERR_PULS = 1.	Word 0	Word 4
Period duration factor	The on-delay that can be assigned parameters can be changed before the start of the output sequence: Time period = Factor x 0.1 x configured time period	Factor: 0 to 255 Period duration: > Pulse duration up to 65.535 s If there is a period duration > 65.535 s, it is set to 65.535 s. If a period duration ≤ pulse duration, it is set to a pulse duration of + 0.2 ms (or pulse duration + 1 ms).	Byte 3	Byte 7
Control signals when the long control interface is used (12 bytes) (as of 6ES7138-4DD01-0AB0)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set 0 → 1 = Start of the output sequence; may be dependent on the HW enable	Byte 4: Bit 0	Byte 10: Bit 0
Number of pulses	Number of pulses that are output at the DO digital output on expiration of the on-delay.	0 to 65535 If the number of pulses is 0, the 2PULSE does not output any pulses. The output sequence is terminated with ERR_PULS = 1.	Word 0	Word 6
Period duration	Setpoint of the time period	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535	Word 2	Word 8
Pulse-pause ratio	You can set the ratio of the pulse duration to the interpulse period	0 to 255 E.g the value of 128 corresponds to a ratio of 50:50.	Byte 5	Byte 11
Feedback Signals				
STS_ENABLE	Indicates an output sequence is running.	0 = pulse output blocked 1 = pulse output running	Byte 0: Bit 0	Byte 4: Bit 0

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1
STS_DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2
ACK_SW_ENABLE	Indicates the status of SW_ENABLE.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 0: Bit 3	Byte 4: Bit 3
ERR_PULS	Indicates a pulse output error.	0 = no pulse output error 1 = pulse output error	Byte 0: Bit 4	Byte 4: Bit 4
SEQ_CNT (as of 6ES7138-4DD01-0AB0)	Is incremented after completion of an output sequence.	Without HW enable: 0 … 1 With HW enable: 0 … 15	Byte 1: Bit 0 … 3	Byte 5: Bit 0 … 3
Current measurement (as of 6ES7138-4DD01-0AB0)	Current at the DO as a SIMATIC S7 analog value.	0 … 27648 (overrange up to 32767)	Word 2	Word 6

Input and output signals of "Pulse train" operating mode (S7-300, S7-400)

Input and output signals of "Pulse train" operating mode

The table below shows the input and output signals of the operating mode "Pulse train".

Input and Output Signals	Meaning	Value range	Channel 0 terminal	Channel 1 terminal
Input signal				
HW enable	You can select the HW enable with the "Function DI" parameter. The signal of the digital input DI is then interpreted by the 2PULSE at the start.	0 = HW enable cleared 1 = HW enable issued 0 → 1 = Start of the output sequence; dependent on the software enable (SW_ENABLE)	1	5
Output Signal				
Pulse train at the DO digital output	The preset number of pulses is output at the DO digital output.	0 = no pulse 1 = pulse	4	8

"On/Off-delay" operating mode (S7-300, S7-400)

"On/Off-delay" operating mode (S7-300, S7-400)

Definition

The signal pending at the digital input DI is output with an on/off-delay at the digital output DO by the 2PULSE.

The following figure shows the block diagram for the operating mode "On/Off-delay".

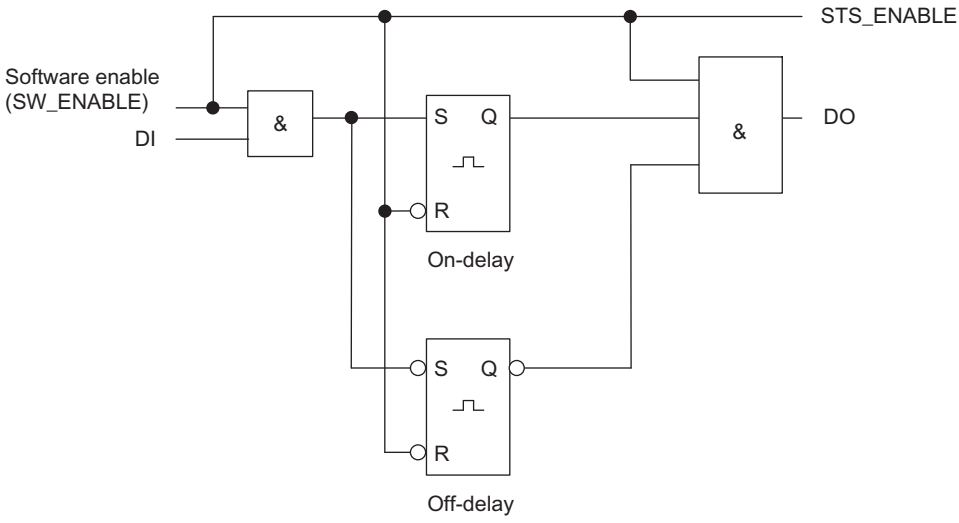

Output Sequence Enable

You must always issue the enable for the output sequence by means of the software enable (Page 1146) (SW_ENABLE 0 → 1; MANUAL_DO = 0) in your control program, STS_ENABLE is then set. The feedback bit (Page 1149)ACK_SW_ENABLE indicates the software enable pending at the 2PULSE.

The positive edge on the DI digital input (0 → 1) starts the on-delay, and the DO digital output (Page 1134) is set on expiration of the on-delay.

The negative edge on the DI digital input (1 → 0) starts the off-delay, and the DO digital output (Page 1134) is deleted on expiration of the off-delay.

If the 2PULSE recognizes a pulse duration or interpulse period that is too short, this is displayed by the pulse output error (Page 1149)ERR_PULS.

At the next edge at the DI digital input, the 2PULSE deletes the ERR_PULS feedback bit.

Pulse diagram

SW_ENABLE is set, while DI = 0:

The following figure shows an output sequence in the operating mode "On/Off-delay" if DI = 0 at the start.

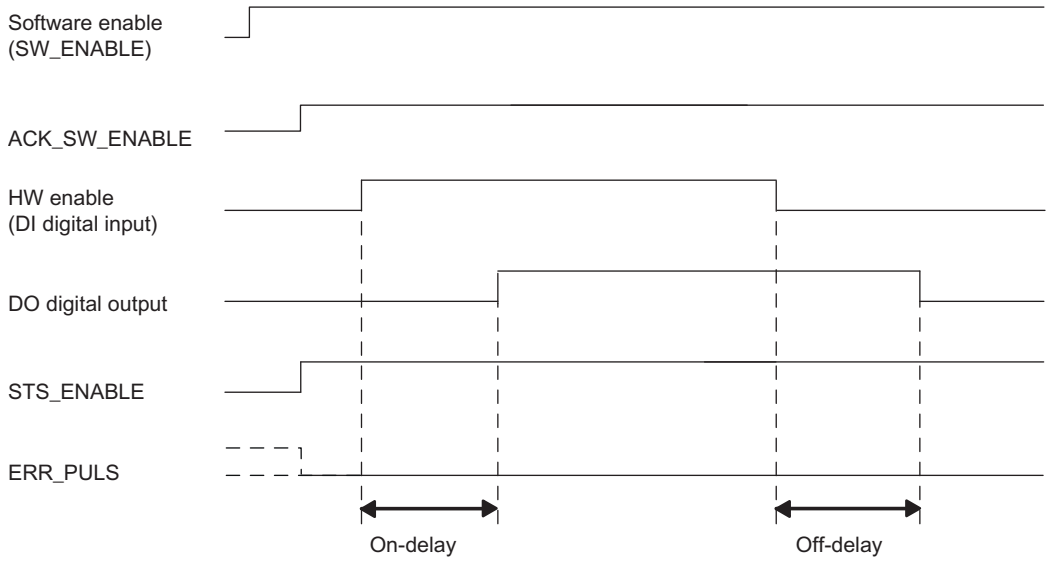

SW_ENABLE is set, while DI = 1:

If SW_ENABLE is set while DI = 1, the first edge at DI (falling edge) is ignored. The following figure shows the corresponding output sequence.

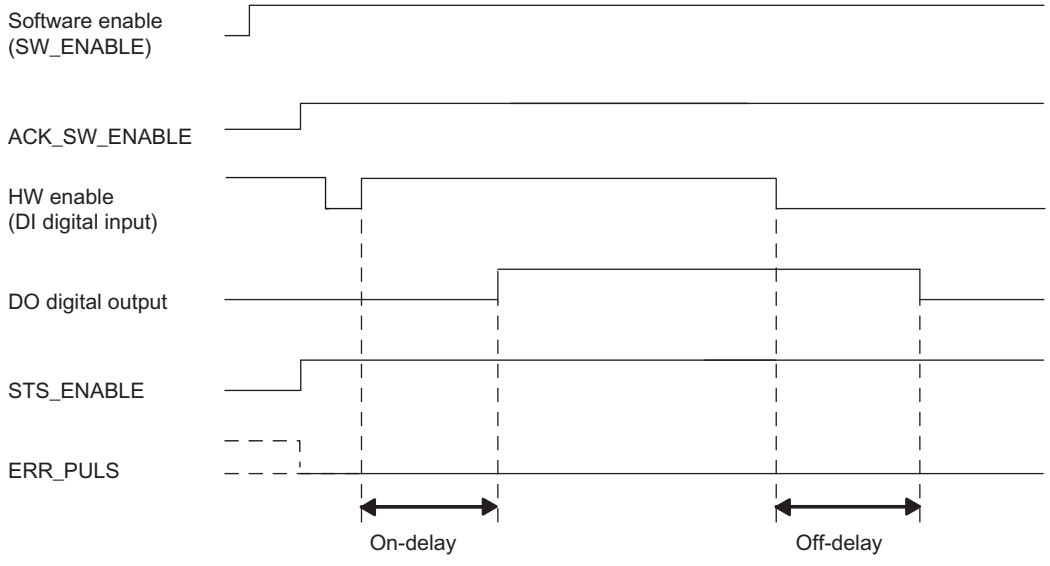

Canceling the output sequence

Clearing the software enable (SW_ENABLE = 0) during the output sequence causes the output sequence to be aborted, along with STS_ENABLE and the digital output.

Truth table

The table below shows how the 2PULSE responds to certain signals.

Software enable SW_ENABLE	Digital input DI	Digital output DO	STS_ENABLE	Output sequence
1	0 → 1	0, if on-delay > 0 1, if on-delay = 0	1	Starting
1	1 → 0	1, if off-delay > 0 0, if off-delay = 0	1	Starting
0	Any status	0	0	Canceling
1	Any status	Previous status remains	1	-
0 → 1	0	0	1	-

Minimum Pulse Duration/Minimum Interpulse Period of the Digital Output DO

The minimum pulse duration/minimum interpulse period of the digital output DO amounts to 100 µs.

Make sure you take this into consideration when you set the on/off-delay and the pulse duration/interpulse period of the digital input DI; otherwise, the response at the digital output DO is not defined.

The Pulse Duration of the Digital Input DI Is Too Short

The 2PULSE detects a pulse that is too short on the negative edge on the digital input DI if:

Pulse duration + Off-delay ≤ On-delay.

Response of the 2PULSE to a pulse duration that is too short:

- ERR_PULS is set
- The current on-delay is deleted.
- The Off-delay is not started.
- The signal level at the digital output DO remains at 0.

ERR_PULS is deleted at the next positive edge on the digital input DI.

The following figure shows the behavior if the pulse duration is too short.

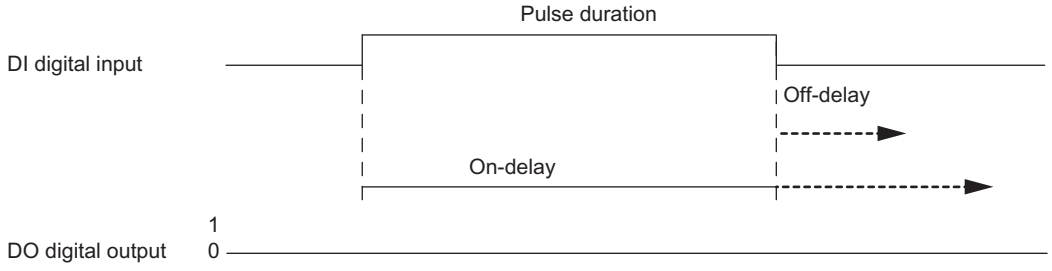

The Interpulse Period of the Digital Input DI Is Too Short

The 2PULSE detects an interpulse period that is too short on the positive edge on the digital input DI if:

Interpulse period + on-delay ≤ Off-delay.

Response of the 2PULSE to a interpulse period that is too short:

- ERR_PULS is set
- The current Off-delay is deleted.
- The on-delay is not started.
- The signal level at the digital output DO remains at 1.

ERR_PULS is deleted with the next negative edge on the digital input DI.

The following figure shows the behavior if the interpulse period is too short.

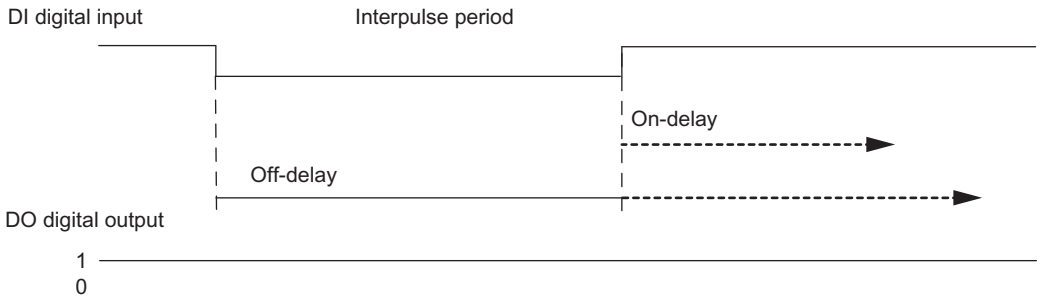

Retriggering the current on-delay

The 2PULSE starts a new on-delay on the positive edge on the digital input DI if:

On-delay > pulse duration + interpulse period

This deletes the current Off-delay.

The digital output DO is only set if signal level 1 is present on the digital input DI longer than the on-delay. This enables you to filter rapid pulse trains.

The following figure shows the retriggering of the current on-delay.

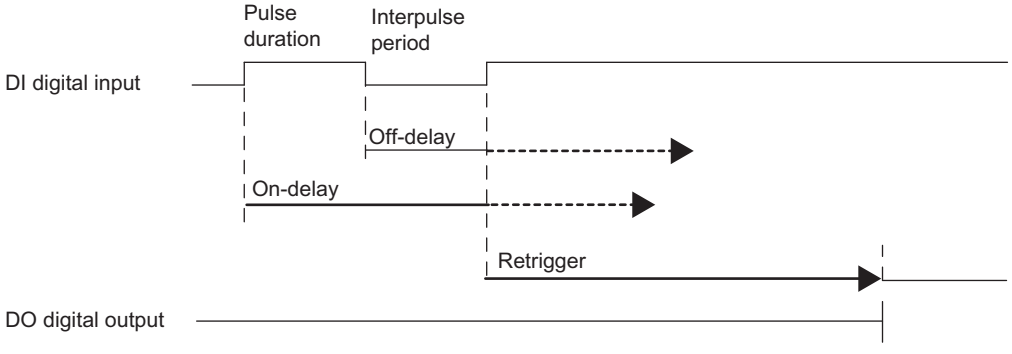

Retriggering the current Off-delay

The 2PULSE starts a new off-delay on the negative edge on the digital input DI if:

Off-delay > pulse duration + interpulse period.

This deletes the current on-delay.

The digital output DO is only deleted if signal level 0 is present on the digital input DI longer than the off-delay.

The following figure shows the retriggering of the current off-delay.

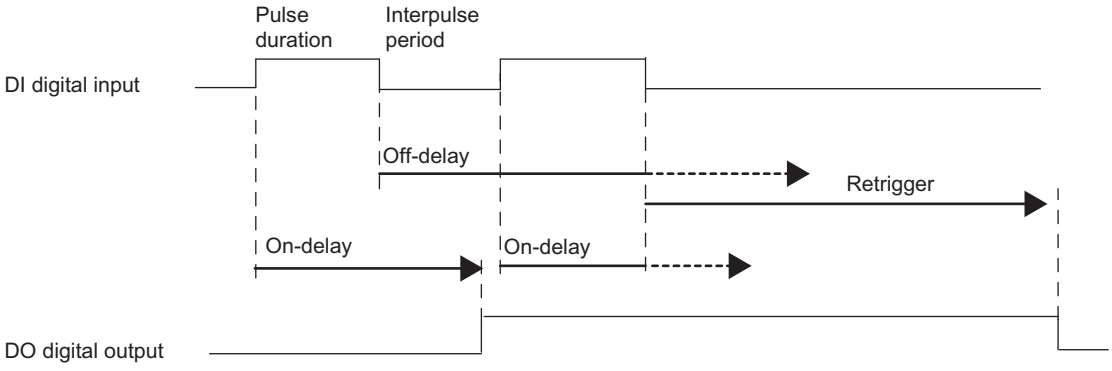

Setting Times Using a Time Base

Use the assigned time base (Page 1144) to select the resolution and the value range of the on-delay and the off-delay.

Time base = 0.1 ms:	You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
Time base = 1 ms:	You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Set the on-delay and change when using the short control interface

You specify the on-delay (Page 1144) as a value between 0 and 65535 in the parameters.

Configured on-delay = Time base x specified numerical value

Using the factor for the on-delay, you can adjust the assigned time in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

On-delay = Factor x 0.1 x configured on-delay

If you change the on-delay factor, the new on-delay is activated with the next positive edge on the digital input DI.

Setting and changing the off-delay

Set the off-delay directly as a numerical value between 0 and 65535 in your control program (Page 1146).

Off-delay = Time base x Specified numerical value

If you change the off-delay factor, the new Off-delay is activated with the next negative edge on the digital input DI.

Isochronous mode (as of 6ES7138-4DD01-0AB0)

In the "On/Off-delay" operating mode, isochronous mode does not have any influence on the functionality.

Parameters for the "On/Off-delay" operating mode (S7-300, S7-400)

Parameters for the "On/Off-delay" operating mode

The table below shows the parameters of the operating mode "On/Off-delay".

Parameter	Meaning	Value range	Default
Operating mode (Page 1143)	Set the "On/Off-delay" operating mode.	• Pulse output • Pulse width modulation • Pulse train • On/Off-delay • Frequency output	Pulse width modulation
Time base (Page 1144)	Use the time base to select the resolution and the value range of the on-delay and Off-delay.	• 0.1 ms • 1 ms	0.1 ms
On-delay (Page 1144) [1]	The time between a positive edge of digital input DI and its output on the digital output DO. You can change the on-delay with your control program with "On-delay factor".	0 to 65535	0
[1] Only if the short control interface (Page 1146) is used If the long control interface is used, enter this value directly in the control interface.			

Control and feedback signals of the "On/Off-delay" operating mode (S7-300, S7-400)

Control and feedback signals of the "On/Off-delay" operating mode

The table below shows the control and feedback signals of the operating mode "On/Off-delay".

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Control signals when the short control interface is used (8 bytes)				
Software enable (SW_ENABLE)	You must always issue the software enable in your control program. If you cancel the software enable, the current output sequence will be canceled.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 2: Bit 0	Byte 6: Bit 0

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Off-delay	The time between a negative edge of the digital input DI and its output on the digital output DO.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the off-delay will not function.	Word 0	Word 4
On-delay factor	You can change the assigned on-delay: On-delay = Factor x 0.1 x configured on-delay	Factor: 0 to 255 On-delay: 0.2 ms to 65.535 s If the on-delay is < 0.2 ms or if the factor = 0, the effective on-delay = 0. If the on-delay is > 65.535 s, the on-delay is limited to 65.535 s.	Byte 3	Byte 7
Control signals when the long control interface is used (12 bytes) (as of 6ES7138-4DD01-0AB0)				
Software enable (SW_ENABLE)	You must always issue the software enable in your control program. If you cancel the software enable, the current output sequence will be canceled.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 4: Bit 0	Byte 10: Bit 0
Off-delay	The time between a negative edge of the digital input DI and its output on the digital output DO.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the off-delay will not function.	Word 0	Word 6
On-delay	The time between a positive edge of the digital input DI and its output on the digital output DO.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the on-delay will not function.	Word 2	Word 8
Feedback Signals				
STS_ENABLE	Indicates the status of the software enable (SW_ENABLE).	0 = software enable blocked 1 = software enable issued	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1
STS_DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2
ACK_SW_ENABLE	Indicates the status of SW_ENABLE.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 0: Bit 3	Byte 4: Bit 3
ERR_PULS	Indicates a pulse output error if the pulse duration or interpulse period is too short.	0 = no pulse output error 1 = pulse output error	Byte 0: Bit 4	Byte 4: Bit 4

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
SEQ_CNT (as of 6ES7138-4DD01-0AB 0)	Is incremented at DO with each edge (positive and negative).	0 … 15	Byte 1: Bit 0 … 3	Byte 5: Bit 0 … 3
Reserved	–	0	Word 2	Word 6

Input and output signals of the "On/Off-delay" operating mode (S7-300, S7-400)

Input and output signals of the "On/Off-delay" operating mode

The table below shows the input and output signals of the operating mode "On/Off-delay".

Input and Output Signals	Meaning	Value range	Channel 0 terminal	Channel 1 terminal
Input signal				
Digital input DI	The signal of the digital input DI is output with an on/off-delay on digital output DO by the 2PULSE.	0 = no pulse 1 = pulse	1	5
Output Signal				
Pulse on the digital output DO	The signal of the digital input DI is output with an on/off-delay on digital output DO by the 2PULSE.	0 = no signal 1 = signal	4	8

"Frequency output" mode (as of 6ES7138-4DD01-0AB0) (S7-300, S7-400)

"Frequency output" operating mode (S7-300, S7-400)

Definition

A square wave signal with a specified frequency is output at the digital output of the 2PULSE.

The output sequence is started on expiration of the configured on-delay (Page 1144) on the DO digital output of the 2PULSE.

The following figure shows the block diagram for the operating mode "Frequency output".

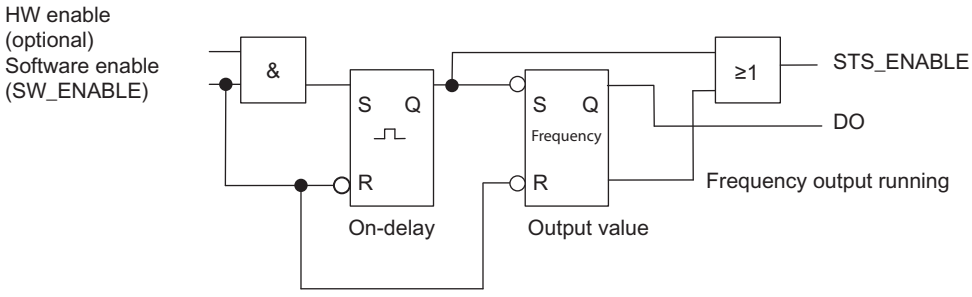

Starting the Output Sequence

You must always issue the enable for the output sequence by means of the software enable (Page 1146) (SW_ENABLE 0 → 1; MANUAL_DO = 0) in your control program.

The feedback bit (Page 1149)ACK_SW_ENABLE indicates the software enable pending at the 2PULSE.

You can also set the digital input DI of the 2PULSE as a HW enable with the "Function DI (Page 1144)" parameter.

If you want to work with the software enable and hardware enable at the same time, when the software enable has been issued, the output sequence starts with the first positive edge of the hardware enable. Further positive edges of the hardware enable during the current output sequence are ignored by the 2PULSE.

When the enable is issued (positive edge), the on-delay is started and the STS_ENABLE set. The frequency output is started on expiration of the on-delay time. The output sequence runs continuously as long as SW_ENABLE is set.

Pulse diagram

The following figure shows an output sequence for the operating mode "Frequency output".

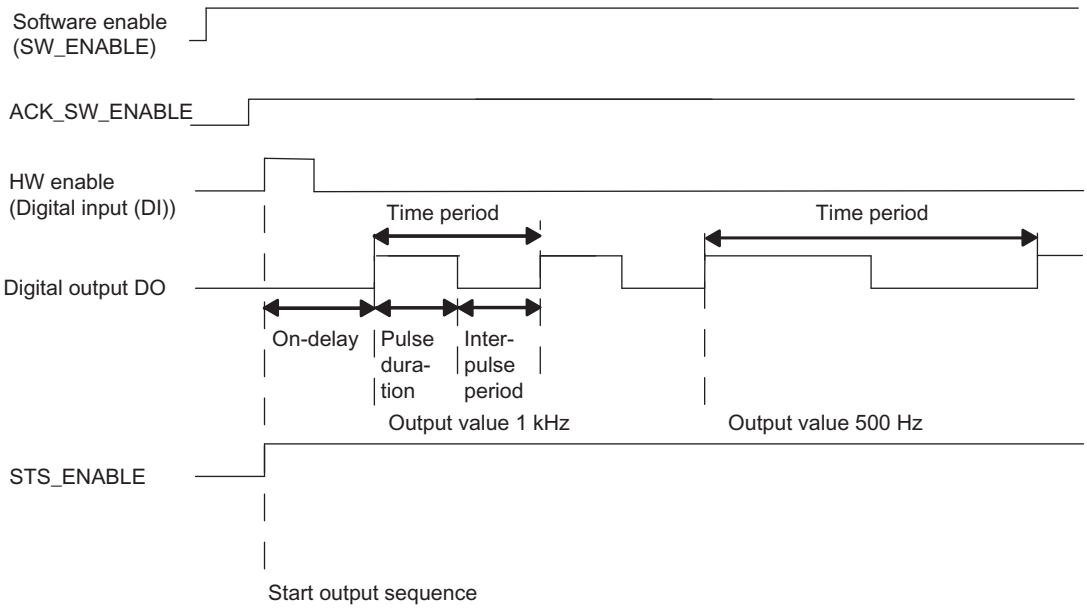

Canceling the output sequence

Deleting the software enable (SW_ENABLE = 0) during the on-delay or the frequency output cancels the output sequence, and STS_ENABLE and the DO digital output are canceled.

You will then have to restart the output sequence.

Truth Table

The table below shows how the 2PULSE responds to certain signals.

Software enable SW_ENABLE	Hardware enable (digital input DI)	Digital output DO	STS_ENABLE	Output sequence
1	0 → 1	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0 → 1	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	Any status	0	0	Canceling
1	Any status	Previous status remains		-
0 → 1	0	0	0	-

Setting Times Using a Time Base

Use the configured time base (Page 1144) to select the resolution and the value range of the on-delay.

Time base = 0.1 ms:	You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
Time base = 1 ms:	You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Setting and Changing the Output Value

You select the range of the output value with the "Output format (Page 1144)" parameter.

- If your output value is between 1 Hz and 5000 Hz, select the output format "1 Hz".
- If your output value is a SIMATIC S7 analog value (between 0 and 27648), select the "S7 analog output" output format. With "S7 analog output" you attain a higher accuracy at the setpoint specification.
  The 2PULSE calculates the frequency according to the following equation:
  Frequency = (Setpoint / 27648) × 5 kHz

You set the output value directly using your control program. The new frequency value is applied at the next rising edge of the output.

Output frequency accuracy

The configured output frequency is output with the following accuracy at the digital output DO:

- 0.01 Hz at output frequencies of 0.1 Hz to 800 Hz
- 0.1 Hz at output frequencies > 800 Hz to 2000 Hz
- 0.5 Hz at output frequencies > 2000 Hz to 5000 Hz

Set the on-delay when using the short control interface

You specify the on-delay (Page 1144) as a value between 0 and 65535 in the parameters.

Configured on-delay = Time base x specified numerical value

When the short control interface is used, the factor for the on-delay can be used to adjust the configured time in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

On-delay = Factor x 0.1 x configured on-delay

Isochronous mode

In the "Frequency output" operating mode, isochronous mode does not have any influence on the functionality.

Parameters of the "Frequency output" operating mode (S7-300, S7-400)

Parameters of the "Frequency output" operating mode

The table below shows the parameters of the operating mode "Frequency output".

Parameter	Meaning	Value range	Default
Operating mode (Page 1143)	Set the "Frequency output" operating mode.	• Pulse output • Pulse width modulation • Pulse train • On/Off-delay • Frequency output	Pulse width modulation
Time base (Page 1144)	Use the time base to select the resolution and the value range of the on-delay.	• 0.1 ms • 1 ms	0.1 ms
Output format (Page 1144)	Choose between 1 Hz and the S7 analog output format, depending on which output frequency resolution you need.	• 1 Hz • S7 analog output	1 Hz
Function DI (Page 1144)	You can use the digital input DI as an input or as a HW enable.	• Input • HW enable	Input
On-delay (Page 1144) [1]	The time from the start of the output sequence to the output of the pulse train. You can change the on-delay in your control program using the "On-delay factor".	0 to 65535	0

[1] Only if the short control interface (Page 1146) is used
If the long control interface is used, enter this value directly in the control interface.

Control and feedback signals of the "Frequency output" operating mode (S7-300, S7-400)

Control and feedback signals of the "Frequency output" operating mode

The table below shows the control and feedback signals of the operating mode "Frequency output".

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Control signals when the short control interface is used (8 bytes)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 2: Bit 0	Byte 6: Bit 0
Output value	The value that is output in frequency format on the digital output DO.	Depending on the output format: • 1 Hz: 0 ... 5000 • S7 analog output: 0 to 27648 If you enter an output value > 5000 or 27648, the 2PULSE limits this to 5000 or 27648.	Word 0	Word 4
On-delay factor	The on-delay that has been assigned parameters can be changed before the start of the output sequence: On-delay = Factor x 0.1 x configured on-delay	0 to 255 If the on-delay <0.2 ms or if factor = 0, the effective on-delay is = 0. If there is an on-delay > 65.535 s, the on-delay is limited to 65.535 s.	Byte 3	Byte 7
Control signals when the long control interface is used (12 bytes)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 4: Bit 0	Byte 10: Bit 0
Output value	The value that is output in frequency format on the digital output DO.	Depending on the output format: • 1 Hz: 0 ... 5000 • S7 analog output: 0 to 27648 If you enter an output value > 5000 or 27648, the 2PULSE limits this to 5000 or 27648.	Word 0	Word 6
On-delay	The on-delay can be changed before the start of the output sequence.	0 to 65535 On-delay = Time basis x Specified numerical value	Word 2	Word 8
Feedback Signals				
STS_ENABLE	Indicates an output sequence is running.	0 = pulse output blocked 1 = pulse output running	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
STS_DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2
ACK_SW_ENABLE	Indicates the status of SW_ENABLE.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 0: Bit 3	Byte 4: Bit 3
SEQ_CNT	No function in this operating mode.	–	Byte 1: Bit 0 … 3	Byte 5: Bit 0 … 3
Reserved	–	0	Word 2	Word 6

Input and output signals of the "Frequency output" operating mode (S7-300, S7-400)

Input and output signals of the "Frequency output" operating mode

The table below shows the input and output signals of the operating mode "Frequency output".

Input and Output Signals	Meaning	Value range	Channel 0 terminal	Channel 1 terminal
Input signal				
HW enable	You can select the HW enable with the "Function DI" parameter. The signal of the digital input DI is then interpreted by the 2PULSE at the start of the output sequence.	0 = HW enable cleared 1 = HW enable issued 0 → 1 = Start of the output sequence; dependent on the software enable	1	5
Output Signal				
Frequency on the digital output DO	The frequency is output on the digital output DO.	0 = no pulse 1 = pulse	4	8

Sequence counter (as of 6ES7138-4DD01-0AB0) (S7-300, S7-400)

The 2PULSE has a sequence counter for each channel with which the completed output sequences are counted.
You can monitor the completion of an output sequence with the sequence counter SEC_CNT in the feedback interface.

The counter has a width of 4 bits. After an high limit violated it jumps back to 0.

Function of the sequence counter

The sequence counter has the following function in the individual operating modes:

- Pulse output (Page 1116) and pulse train (Page 1100)
  - Without hardware enable the counter is set to 1 after completion of the output sequence.
  - With hardware enable, the counter is incremented after every completed output sequence.
- On/Off-delay (Page 1121)
  The counter is incremented at DO with each edge (positive and negative).
- Pulse width modulation and frequency output
  The counter does not have any function.

When the software enable (SW_ENABLE = 0) is created, the counter is reset to 0.

Application options

The sequence counter can be used in particular for:

- Detecting (counting) very short pulses
- Counting output sequences in connection with the hardware enable

Function: Current measurement (as of 6ES7138-4DD01-0AB0) (S7-300, S7-400)

The measurement of current can be set in a variety of applications for control and diagnostics purposes. You can read the measured value of the current from the feedback interface.

Principle of operation

A measurement of current only takes place during the pulse output in the "Pulse width modulation (Page 1106)" and "Pulse train (Page 1116)" operating modes. Without a pulse output (at STS_ENABLE = 0), 0 is supplied as the measured value. The measured value is valid at STS_ENABLE = 1.

In the "Pulse output", On/Off-delay" and "Frequency output" operating modes, 0 is always measured as the measured value of the current.

The returned measured value of the current is a mean value of measured values that is recorded during the duration of a period. At period durations < 5 ms, the interval between two individual measuring points amounts to a maximum of 40 µs. When longer periods are involved, 128 measuring points per period are always recorded. This results in correspondingly longer intervals between the measuring points.

In isochronous mode, a new measured value of the current is always made available in the feedback interface at the moment T_i. In non-isochronous mode, a new measured value of the current is made available about every 500 µs.

Note

In order for the current measurement to work correctly, no freewheeling diode may be applied to the load.

Measuring range and measured value

The measuring range is set fixed to a rated current of 2 A.

The measured value of the current is supplied as a SIMATIC S7 analog value in the feedback interface (Page 1149).

- A current of 2 A corresponds to the SIMATIC S7 analog value of 27648.
- With a parallel connection, a current of 4 A corresponds to the SIMATIC S7 analog value of 27648.

Measurements of current are possible up to a SIMATIC S7 analog value of 32767. This corresponds to a current of 2.4 A. Currents above 2 A may occur only briefly.

Accuracy

The accuracy of the measurement of current amounts to 3% of the measuring range.

Note

This accuracy during current measurement can only be achieved if the time period is not changed during the measurement.

Function: Direct control of the digital output DO (S7-300, S7-400)

Definition

You can directly control the digital output DO of the 2PULSE to test the actuator you have connected. To do this, you have to select the function from your control program (Page 1146) with the MANUAL_DO control bit set and SW_ENABLE control bit cleared. The function

- Supports you in commissioning a plant
- Allows you to utilize an unused channel of the 2PULSE as DO

After you have selected the function, the feedback bits (Page 1149)STS_ENABLE and ERR_PULS are cleared by the 2PULSE, and an active output sequence is aborted. The function takes precedence over the currently selected operating mode.

You specify the status of the digital output DO with the control bit (Page 1146)SET_DO.

When you clear the MANUAL_DO control bit, you deselect the "Direct control of the digital output DO" function. This deletes the digital output DO. You can then restart an output sequence with the rising edge of SW_ENABLE.

Control and Feedback Signals/Output Signal

Signals	Meaning	Value range	When the short control interface is used (8 bytes)		When the long control interface is used (12 bytes)	
			Channel 0 address	Channel 1 address	Channel 0 address	Channel 1 address
Control signals						
SW_ENABLE	To select the function, the control bit must be deleted.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 2: Bit 0	Byte 6: Bit 0	Byte 4: Bit 0	Byte 10: Bit 0
MANUAL_DO	You can select and deselect the function with the control bit.	0 = direct control of the DO not selected. 1 = direct control of the DO selected.	Byte 2: Bit 1	Byte 6: Bit 1	Byte 4: Bit 1	Byte 10: Bit 1
SET_DO	You use the control bit to set the status of the digital output DO.	0 = signal 0 on digital output DO 1 = signal 1 on digital output DO	Byte 2: Bit 2	Byte 6: Bit 2	Byte 4: Bit 2	Byte 10: Bit 2
Feedback signals						
STS_ENABLE	Deleted after the function has been selected.	0 = pulse output blocked 1 = pulse output running	Byte 0: Bit 0	Byte 4: Bit 0	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1	Byte 0: Bit 1	Byte 4: Bit 1
STS_ DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2	Byte 0: Bit 2	Byte 4: Bit 2
Output Signal						
Digital output DO	The status preset with the SET_DO control bit is output on the digital output DO.	0 = No signal 1 = Signal	4	8	4	8

Function: Error detection / diagnostics (S7-300, S7-400)

Parameter Assignment ErrorERR_PARA

If the 2PULSE cannot identify the parameters as its own, it generates a parameter assignment error (Page 1149). The two channels are then not assigned parameters.

The 2PULSE slot you configure must match the setup.

Pulse output error ERR_PULS

The 2PULSE detects a channel-specific pulse output error in the operating modes "Pulse output (Page 1100)", "On/off-delay (Page 1121)", and "Pulse train (Page 1116)".

You can find information about causes and responses in the respective operating mode descriptions and under "Feedback interface of 2PULSE (Page 1149)".

The pulse output error detected is displayed for the relevant channel with the feedback bit (Page 1149)ERR_PULS.

Short circuit of the encoder supply ERR_24V

The 2PULSE detects a short circuit in the encoder supply that it makes available at terminals 2 and 6.

The short circuit error detected is displayed for the two channels with the feedback bit (Page 1149)ERR_24V.

A drop in the 24 V supply voltage can also result in a short circuit message.

Short circuit digital output ERR_DO

The 2PULSE detects a short circuit on the digital output of the channel.

The detected short circuit is displayed for the relevant channel with the feedback bit (Page 1149)ERR_DO.

Note

If a short circuit occurs, the digital output is able to briefly supply a current substantially higher than the rated value.

Diagnostic Message

If a short circuit occurs in the encoder supply or in the digital output, the 2PULSE generates a diagnostic message for the connected CPU/master. You must enable the "Group diagnostics (Page 1142)" parameter in this case.

Note

Diagnostic messages provide you the information that an error has occurred in one or both channels of the 2PULSE. You can evaluate the diagnostics with STEP 7.

The diagnostic record DS1 only contains the information for Channel 0. However, a diagnostic interrupt is also triggered at errors in Channel 1. The feedback signals provide detailed information about the occurring errors from both channels. Evaluate the feedback interface (Page 1149) in new user programs (see below).

Parameters

Parameter	Meaning	Value Range	Default
Group diagnostics (Page 1142)	When group diagnostics is enabled, the 2PULSE generates a diagnostic message for the CPU/master.	Disable/enable	Disable
Diagnostics DO (Page 1143)	The 2PULSE detects a short circuit of the digital outputs when "Diagnostics DO" = On.	Off/on	On

Feedback Signals

Feedback Signals	Meaning	Value Range	Channel 0 address	Channel 1 address
ERR_PARA	Indicates a parameter assignment error.	0 = no parameter assignment error 1 = parameter assignment error	Byte 0: Bit 5	Byte 4: Bit 5
ERR_PULS	Indicates a pulse output error.	0 = no pulse output error 1 = pulse output error	Byte 0: Bit 4	Byte 4: Bit 4
ERR_24V	Indicates a short circuit of the encoder supply.	0 = no encoder supply short circuit 1 = encoder supply short circuit	Byte 0: Bit 7	Byte 4: Bit 7
ERR_DO	Indicates a short circuit of the digital output DO.	0 = no digital output short circuit 1 = digital output short circuit	Byte 0: Bit 6	Byte 4: Bit 6

Parallel connection of both channels (as of 6ES7138-4DD01-0AB0) (S7-300, S7-400)

Definition

In order to achieve higher output current, you can connect both channels of the 2PULSE in parallel. You then obtain a rated current of 4 A. This allows you to also operate actuators with a higher power requirement directly at the 2PULSE.

Parameter assignment and operation

You can enable parallel connection of the two channels during the configuration by selecting the check box "Parallel connection Channel 0 and 1 (Page 1142)".

Once enabled, the 2PULSE operates like a single-channel module. All the parameter assignments and operations via the control interface (Page 1146) have to be executed with Channel 0. Channel 1 cannot be configured and operated.

Feedback signals (Page 1149) are also only provided for Channel 0.

Measurement of current

During the measurement of current, the measured values from both channels are averaged once more. This results in a measuring range up to 4 A.

The measured value of the current is supplied as a SIMATIC S7 analog value in the feedback interface. A current of 4 A corresponds here to the SIMATIC S7 analog value of 27648.

The accuracy of the current measurement also amounts to 3% of the measuring range in case of parallel connection.

Pin assignment for the parallel connection

The following terminals have to be connected at the terminal module for the parallel connection of both channels of the 2PULSE:

- Terminal 3 and Terminal 7
- Terminal 4 and Terminal 8

You then connect the load (e.g. a actuator) directly to Terminals 3 and 4 or to Terminals 7 and 8. The following figure shows a connection layout.

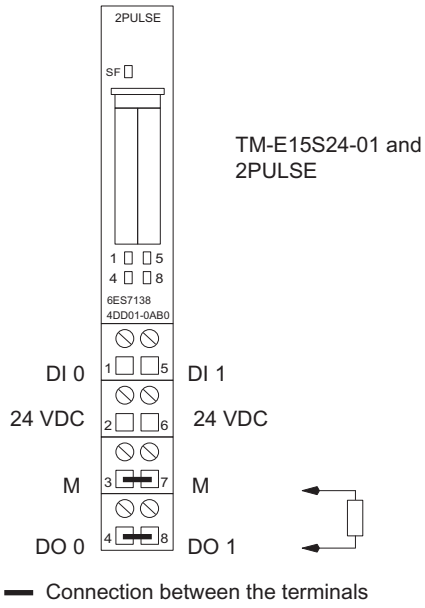

Reaction to CPU/Master STOP (S7-300, S7-400)

Definition

You can configure the reaction (Page 1142) of the 2PULSE to failure of the higher-level controller for the two channels together.

Reaction to CPU/Master STOP	Channel-specific response and status of the 2PULSE
Turn off DO	Delete the DO digital output Delete STS_ENABLE and Terminate the current output sequence
Continue working mode	The DO digital output remains unchanged STS_ENABLE remains unchanged and The current output sequence is continued
DO substitute a value	Output of the channel-specific, substitute value that has been assigned parameters of the DO digital output Delete STS_ENABLE and Terminate the current output sequence
DO keeps last value	The DO digital output remains unchanged Delete STS_ENABLE and Terminate the current output sequence

Startup

To start a new output sequence after CPU/Master STOP and a set ACK_SW_ENABLE, first delete SW_ENABLE. Keep deleting until ACK_SW_ENABLE is also deleted.

If the mode is to continue during a change from CPU/Master STOP to RUN (startup), the CPU/Master cannot clear the outputs. **Possible solution:** In the part of the user program that is executed during startup, set the "Software enable" (SW_ENABLE = 1) control bit.

Modified Parameter Assignment

The status assumed by the 2PULSE at CPU/Master STOP remains even in the case of parameter assignment or configuration of the ET 200S station. This occurs, for example, at POWER ON of the CPU/master or the IM 151 or at the resumption of DP transfer.

In "Continue working mode", however, and after loading changed parameters or configuration of the ET 200S station to the CPU/master, the 2PULSE terminates the process. As a result, the 2PULSE does the following:

- Deletes the DO digital output.
- Deletes STS_ENABLE and
- Terminates the current output sequence.

3.5.8.3 Configuring and assigning parameters to 2PULSE (S7-300, S7-400)

Parameters (S7-300, S7-400)

Selecting an operating mode (S7-300, S7-400)

Operating mode

Select the operating mode of the 2PULSE in accordance with your requirements:

- 2 Pulse: 8 bytes output data – compatible to predecessor module
  The control interface uses fewer addresses in the process image.
- 2 Pulse ext. control interface: 12 bytes output data – improved controlling possibilities
  You can enter the setpoints more conveniently.

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

Group diagnostics are enabled in the respective check box.

When group diagnostics is enabled, the 2PULSE will generate a diagnostic message for the CPU/master.

Selecting reaction to CPU/Master STOP (S7-300, S7-400)

Reaction to CPU/Master STOP

Select the desired response to CPU/Master STOP from the dropdown list:

- Turn off DO
- Continue working mode
- DO substitute a value
- DO keep last value

You will find further information on the response and mode of the 2PULSE under "Reaction to CPU/Master STOP (Page 1141)".

Activating parallel connection Channel 0 and 1 (S7-300, S7-400)

Parallel connection channel 0 and 1 (as of 6ES7138-4DD01-0AB0)

If you require a rated current of more than 2 A, you can connect the two channels of the 2PULSE in parallel. This is done by selecting the check box "Parallel connection Channel 0 and 1".

Additional information is available under "Parallel connection of both channels (as of 6ES7138-4DD01-0AB0) (Page 1139)".

Channel n (S7-300, S7-400)

Activating DO diagnostics (S7-300, S7-400)

Diagnostics DO

2PULSE will detect a short at the digital outputs if you have selected the "DO diagnostics" checkbox.

Selecting DO substitute value (S7-300, S7-400)

Substitute value DO

Select the required substitute value for the digital output:

- 0
- 1

The parameter "Substitute value DO" may only have a value not equal to zero if the parameter "Reaction to CPU/Master STOP (Page 1142)" has been set to "DO substitute a value".

Selecting an operating mode (S7-300, S7-400)

Operating mode

Select the required operating mode from the drop-down list:

- Pulse output (Page 1100)
- "Pulse width modulation" (Page 1106)
- Pulse train (Page 1116)
- On/Off delay (Page 1121)
- Frequency output (Page 1129) (as of 6ES7138-4DD01-0AB0)

Selecting the output format (S7-300, S7-400)

Output format

The output format is relevant for operating modes "Pulse width modulation(PWM)" and "Frequency output" (as of 6ES7138-4DD01-0AB0).

- You can select the following output formats from the drop-down list in "Pulse width modulation (PWM) (Page 1106)" mode:
  - Per mill
  - S7 analog output
- You can select the following output formats from the drop-down list in "Frequency output (Page 1129)" mode:
  - 1 Hz
  - S7 analog output

Selecting a time base (S7-300, S7-400)

Time base

Use the time base to select the resolution and value range for all predefined times:

- 0.1 ms
  You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
- 1 ms
  You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Selecting DI function (S7-300, S7-400)

Function DI

You can use the DI digital input as an input or as a hardware enable.

Select the required function for the digital input from the drop-down list:

- Input
- Hardware enable

Specifying on delay (S7-300, S7-400)

On-delay

The on delay is the time from the start of the output sequence to pulse output.

Enter the required on delay in the input field.

Valid value range: 0 … 65535

Specifying minimum pulse duration (S7-300, S7-400)

Minimum pulse duration

Minimum pulse duration is relevant for operating modes "Pulse width modulation (PWM) (Page 1106)" and "Frequency output (Page 1129)" (as of 6ES7138-4DD01-0AB0).

You assign the minimum pulse duration and minimum break time using the "Minimum pulse duration" parameter; they always have the same value.

You can, for example, enter the response time of the actuator connected to your DO digital output here.

Valid value range: 0 … 65535

Specifying pulse duration (S7-300, S7-400)

Pulse duration

Pulse duration is relevant for "Pulse train (Page 1116)" mode.

You can, for example, enter the response time of the actuator connected to your DO digital output here.

Valid value range:

- With time base 1 ms: from 1 ... 65535
- With time base 0.1 ms: from 2 ... 65535

Specifying period (S7-300, S7-400)

Time period

The period is relevant for "Pulse width modulation (PWM) (Page 1106)" and "Pulse train (Page 1116)" operating modes.

When specifying the time period, take into account the set minimum pulse duration / pulse duration and the response time of the actuator connected to the digital output DO. The period should always be a multiple of the response time of the actuator connected to the DO digital output.

You specify the period as a numerical value in accordance with the time base:

- With time base 1 ms: from 1 ... 65535
- With time base 0.1 ms: from 2 ... 65535

Configured period = Time base x specified numerical value

2PULSE control and feedback interface (S7-300, S7-400)

2PULSE control assignment (S7-300, S7-400)

Note

The following control interface data is consistent for 2PULSE:

- Bytes 0 ... 3
- Bytes 4 ... 7

Assignment of the short control interface (8 bytes)

The following table shows the assignment of the short control interface (outputs) of 2PULSE.

Address		Assignment
Channel 0	Channel 1	
Word 0	Word 4	Depending on the mode • Pulse output: Pulse duration • Pulse-width modulation: Output value • Pulse train: Number of pulses • On/Off-delay: Off-delay • Frequency output: Output frequency (as of 6ES7138-4DD01-0AB0)
Byte 2	Byte 6	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: Reserve = 0 Bit 2: SET_DO Bit 1: MANUAL_DO Bit 0: SW_ENABLE
Byte 3	Byte 7	Depending on mode • Pulse output: On-delay factor • Pulse width modulation: Period duration factor • Pulse train: Period duration factor • On/Off-delay: On-delay factor • Frequency output: On-delay factor (as of 6ES7138-4DD01-0AB0)

Assignment of the long control interface (12 bytes) (as of 6ES7138-4DD01-0AB0)

The following table shows the assignment of the long control interface (outputs) of 2PULSE.

Address		Assignment
Channel 0	Channel 1	
Word 0	Word 6	Depending on mode • Pulse output: Pulse duration • Pulse width modulation: Output value • Pulse train: Number of pulses • On/Off-delay: Off-delay • Frequency output: Output frequency
Word 2	Word 8	Depending on mode • Pulse output: On-delay • Pulse width modulation: Period • Pulse train: Period • On/Off-delay: On-delay • Frequency output: On-delay
Byte 4	Byte 10	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: Reserve = 0 Bit 2: SET_DO Bit 1: MANUAL_DO Bit 0: SW_ENABLE
Byte 5	Byte 11	Depending on mode • Pulse output: Not used • Pulse width modulation: On-delay factor • Pulse train: Pulse-pause ratio • On/Off-delay: Not used • Frequency output: Not used

Notes on the Control Signals

Control Signal	Explanation
"Pulse output" operating mode:	
• Pulse duration [1) 2)]	The time that is set for the DO digital output on expiration of the on-delay.
• On-delay factor [1)]	You can change the configured on-delay before the start of the output sequence: On-delay = Factor × 0.1 × configured on-delay
• On-delay [2)]	The time from the start of the output sequence to the output of the pulse train.
"Pulse width modulation" operating mode:	

Control Signal	Explanation
• Output value [1] [2]	Value that is output with pulse-width modulation at the DO digital output on expiration of the on-delay.
• Time period factor [1]	You can change the configured period duration: Period duration = Factor × 0.1 × configured period duration
• Time period [2]	When specifying the time period take into account the set minimum pulse duration and the response time of your actuator that is connected to the digital output DO.
• On-delay factor [2]	You can change the configured on-delay before the start of the output sequence: On-delay = Factor × 0.1 × configured on-delay
"Pulse train" operating mode:	
• Number of pulses [1] [2]	Number of pulses that are output at the DO digital output on expiration of the on-delay.
• Period factor [1]	You can change the configured time period before the start of the output sequence: Time period = Factor x 0.1 x configured time period
• Period [2]	When specifying the period, take into account the set minimum pulse duration and the response time of the actuator connected to the digital output DO.
• Pulse-pause ratio [2]	You can set the ratio of pulse duration to break time: Pulse/Break ratio = (Setpoint / 255) × Period
"On/Off-delay" operating mode:	
• Off-delay [1] [2]	The time between a negative edge of the digital input DI and its output on the digital output DO.
• On-delay factor [1]	You can change the configured on-delay before the start of the output sequence: On-delay = Factor × 0.1 × configured on-delay
• On-delay [2]	The time between a positive edge of the digital input DI and its output on the digital output DO.
"Frequency output" mode(as of 6ES7138-4DD01-0AB0)**:**	
• Output frequency [1] [2]	Frequency that is output at the DO digital output after expiration of the on-delay.
• On-delay factor [1]	You can change the configured on-delay before the start of the output sequence: On-delay = Factor × 0.1 × configured on-delay
• On-delay [2]	The time from the start of the output sequence to the output of the frequency.
Direct control of the digital output	
• MANUAL_DO	You use the control bit to select and deselect the "Direct control of the digital output DO" function.
• SET_DO	You use the control bit to set the status of the digital output DO.
Software enable	
• SW_ENABLE	You must always issue the software enable in your control program. If you do not use a hardware enable, the output sequence will be started by the positive edge of the software enable. If you delete the software enable, the current output sequence will be terminated.

[1] When using the short control interface

[2] When using the long control interface

2PULSE feedback interface (S7-300, S7-400)

Note

The following data of the feedback interface are consistent for the 2PULSE:

- Bytes 0 ... 3
- Bytes 4 ... 7

Assignment of the Feedback Interface

The following table shows the assignment of the feedback interface (inputs) of 2PULSE.

Address		Assignment
Channel 0	Channel 1	
Byte 0	Byte 4	Bit 7: ERR_24V Bit 6: ERR_DO Bit 5: ERR_PARA Bit 4: ERR_PULS Bit 3: ACK_SW_ENABLE Bit 2: STS_DI Bit 1: STS_DO Bit 0: STS_ENABLE
Byte 1	Byte 5	Bit 0 to Bit 3: Sequence counter SEQ_CNT (as of 6ES7138-4DD01-0AB0) Bit 4 to Bit 7: Reserve = 0
Word 2	Word 6	Current measurement (as of 6ES7138-4DD01-0AB0)

Note

In line with the IM 151 interface module used and with the configuration, a feedback value (substitute value) of either 16#7FFF7FFF or 16#00000000 will be output if the module fails.

Notes on the feedback signals

Checkback signal	Explanation
ACK_SW_ENABLE	Indicates the status of the software enable pending at the 2PULSE.
ERR_24V	Indicates a short circuit or drop of the encoder supply.
ERR_DO	Indicates a short circuit at the digital output.
ERR_PARA	Indicates a parameter assignment error.

Checkback signal	Explanation
ERR_PULS	"Pulse output" operating mode: Indicates a pulse output error. If the pulse duration is reduced after expiration of the on-delay so the time is less than the time already output, this is detected by the 2PULSE. The 2PULSE deletes the feedback bit ERR_PULS next time the output sequence starts. "Pulse train" operating mode: Indicates a pulse output error. If the number of pulses is reduced after expiration of the on-delay and the smaller number of pulses is already output, this is detected by the 2PULSE. The 2PULSE deletes the feedback bit ERR_PULS next time the output sequence starts. "On/Off-delay" mode: Indicates a pulse output error if the pulse duration or interpulse period is too short. The 2PULSE deletes the ERR_PULS feedback bit at the next positive edge of the software enable or at the next edge at the DI digital input.
STS_ DI	Indicates the signal level at the digital input DI.
STS_DO	Indicates the signal level at the digital output DO.
STS_ENABLE	"Pulse output" operating mode: Is set at the start of the output sequence until the pulse duration expires. If you delete the software enable (SW_ENABLE) or the 2PULSE detects a pulse output error (ERR_PULS), STS_ENABLE is deleted. "Pulse-width modulation" operating mode: Set at start of output sequence. If you delete the software enable (STS_ENABLE), SW_ENABLE is deleted. "Pulse train" operating mode: Is set at the start of the output sequence until the output of the last pulse. If you delete the software enable (SW_ENABLE) or the 2PULSE detects a pulse output error (ERR_PULS), STS_ENABLE will be deleted. "On/Off-delay" mode: Indicates the status of the software enable (SW_ENABLE) detected by the 2PULSE. "Frequency output" mode (as of 6ES7138-4DD01-0AB0): Set at start of output sequence. If you delete the software enable (STS_ENABLE), SW_ENABLE will be deleted.
SEQ_CNT (as of 6ES7138-4DD01-0AB0)	Counts the completed output sequences in the "Pulse output", "Pulse train" and "On/Off-delay" operating modes. No function in the "Pulse width modulation" and "Frequency output" operating modes.
Current measurement (as of 6ES7138-4DD01-0AB0)	Current at the DO as a SIMATIC S7 analog value

3.5.8.4 2PULSE diagnostics (S7-300, S7-400)

Diagnostics using the LED display (S7-300, S7-400)

LED display at the 2PULSE

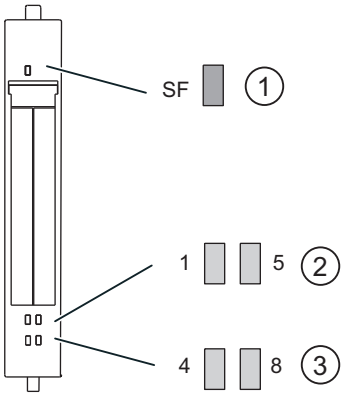

① Group error (red)
② Status display for digital input (green)
③ Status display for the digital output (green)

Status and error displays by means of LEDs on the 2PULSE

The table below shows the status and fault displays on the 2PULSE.

Event (LED)					Cause	Remedy
SF	1	5	4	8		
On					No parameter assignment. There is a diagnostic message.	Check the error bits types in the feedback interface. Evaluate the diagnostics data.
	On				Input DI0 on Channel 0 activated.	
		On			Input DI1 on Channel 1 activated.	
			On		Output DO0 on Channel 0 activated.	
				On	Output DO1 on Channel 1 activated.	

Error types (S7-300, S7-400)

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

2PULSE error types

The following table shows the 2PULSE error types.

Error class		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the sensor supply or the actuator. A drop in the 24 V supply voltage can also result in this error message.	Check the wiring to the switches and actuators. Correct the process wiring.
9_D	01001: Error	Internal module error occurred.	Replace the module.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.

A simple error evaluation is also possible through the corresponding bits in the feedback interface (Page 1149).

3.6 Auxiliary functions (S7-1200, S7-1500)

3.6.1 Polyline (S7-1200, S7-1500)

Polyline

The Polyline instruction provides a function with the characteristic curve of the polyline whose points can be used, for example, to linearize the behavior of non-linear sensors.

The Polyline instruction can be used with an S7-1500 CPU Firmware 2.0 and higher and an S7-1200 CPU Firmware 4.2 and higher.

Additional information

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

3.6.2 SplitRange (S7-1200, S7-1500)

SplitRange

The SplitRange instruction splits the output value range of the PID controller into multiple subranges. These subranges enable control of a process that is influenced by multiple actuators.

The SplitRange instruction can be used with an S7-1500 CPU Firmware 2.0 and higher and an S7-1200 CPU Firmware 4.2 and higher.

Additional information

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

3.6.3 RampFunction (S7-1200, S7-1500)

RampFunction

The RampFunction instruction limits the slew rate of a signal. A signal jump at the input is output as ramp function of the output value, to achieve a smoother response, for example, without influencing the disturbance reaction.

The RampFunction instruction can be used with an S7-1500 CPU Firmware 2.0 and higher and an S7-1200 CPU Firmware 4.2 and higher.

Additional information

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

AUTOHOTSPOT

Time-based IO (S7-1500) 4

4.1 High-precision input/output with time-based IO (S7-1500)

4.1.1 Time-based IO basics (S7-1500)

4.1.1.1 Using Time-based IO (S7-1500)

Many processes in a plant require a relatively precise reproducibility as far as time is concerned. The reproducibility can be optimized to a limited extent by shorter CPU cycle times. The use of high-precision input/output with Time-based IO provides a more accurate reproducibility than the one that can be achieved by optimizing the CPU cycle time. An additional advantage of Time-based IO is the independence from the minimum time scale of the application cycle.

Below is a selection of typical areas of application that can benefit from this technology.

Defined response times

An output signal that responds with a precise delay time that is independent of the application cycle can be output by adding a specified time to the time stamp of an edge at the digital input.

Length measurement

The length of a product passing by can be determined by the difference between two input time stamps and an associated travel speed.

Cam controller

The ongoing movement can be extrapolated from a synchronous position information (for example, from a counter module or an axis). Based on this result, the time of a switching position (cam position) is calculated and transferred to the TIO module (output). This way the switch event takes place at the required position.

Dosing

By specifying a switch-on and a switch-off edge to a time-controlled digital output, a valve can be opened for a specific time and the amount of liquid can be dosed accordingly.

4.1.1.2 Time behavior of Time-based IO (S7-1500)

Time behavior of standard technology (S7-1500)

The time behavior of the inputs/outputs depends on the following factors for standard technology:

- CPU program (program structure)
- Bus cycle times (fieldbus, backplane bus)
- Cycle time of the I/O modules
- Internal cycle time of sensors/actuators

A deterministic statement as to

- when an input event (e.g., sensor signal) has taken place
- when the output event has an effect on the input event (e.g., output switches)

will become inaccurate due to the time factors listed above.

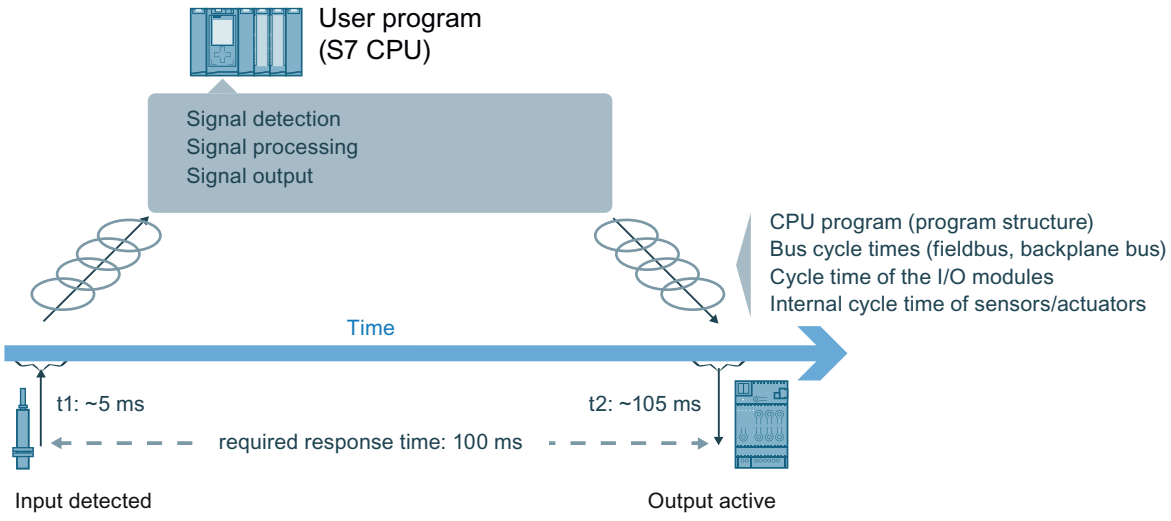

Time-based IO properties (S7-1500)

Time-controlled I/O functionality

Time-based IO stands for time-based processing of I/O signals. All input signals are referred to one time (TIO_Time). The input signals receive the time stamp t1. After signal processing, the output event can also be linked with the TIO_Time and output at the required time. Output takes place at time t2.

Here an example for Time-based IO:

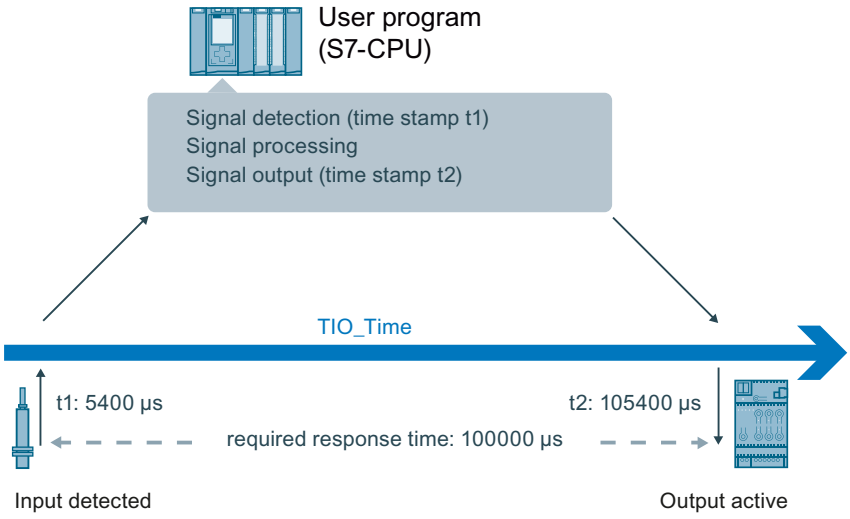

Independence and focus of Time-based IO

The shared time basis (TIO_Time) of all components involved is the basis for Time-based IO. By using TIO_Time, the accuracy of the output with Time-based IO does not depend on:

- CPU program (program structure)
- Bus cycle times (fieldbus, backplane bus)
- Cycle time of the I/O modules
- Internal cycle time of the IO-Link sensors/actuators

The focus of Time-based IO is not on the I/O response time but on the predictability (determinism) of I/O signals. With Time-based IO, it is possible to respond to an input signal with an output signal within a defined time. Keep in mind the system-dependent minimum response time when using Time-based IO.

Time-based IO stands for:

- I/O functionalities executed with high precision
- I/O processes with time stamp

Accuracy

Accuracy is crucial for the performance capability of Time-based IO.

Accuracy is a property of

- TIO module
- IO-Link sensor/ actuator

and indicates the deviation with which the required response is achieved. For Time-based IO, the accuracy and reproducibility of the response is in the millisecond range.

Response time

In addition to a very high accuracy, a minimum response time to an input event can result depending on the configuration.

The response time is the time between the input event and the required output event.

For the minimum response time:

at least 3 * application cycle (T_{APP})

4.1.1.3 System environment (S7-1500)

Introduction

Possible system configurations with PROFINET for the use of Time-based IO are shown below.

Use with TIA Portal

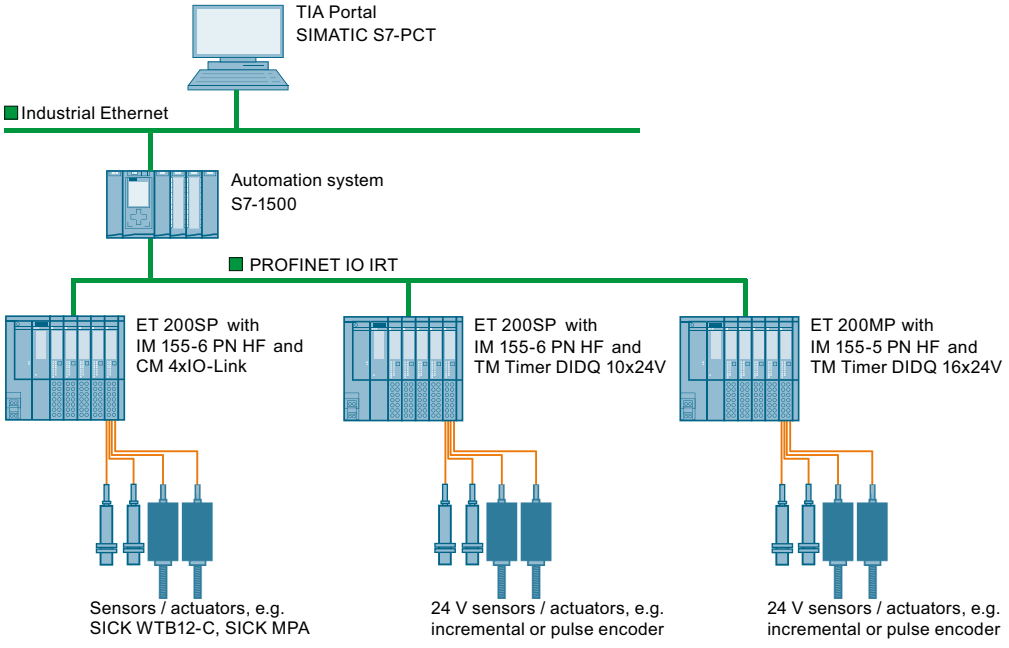

4.1.1.4 Hardware requirements (S7-1500)

Introduction

The properties of the hardware components required for Time-based IO are listed below. You will also find specific modules as an example.

Requirements

Component	Properties	Examples
CPU	• Isochronous mode/ PROFINET IO IRT: provides defines response time and high-precision plant behavior.	• CPU 1511-1 PN
ET 200 interface module	• Supports isochronous mode	• ET 200SP with IM 155-6 PN HF (as of firmware V2.1) • ET 200MP with IM 155-5 PN HF
I/O module	• TIO module	• ET 200SP: CM 4xIO-Link (as of firmware V2.0) TM Timer DIDQ 10x24V • ET 200MP: TM Timer DIDQ 16x24V
IO-Link Device for CM 4xIO-Link	• supports IO-Link time stamp function	• SICK WTB12-C • halstrup-walcher PSE, PSS, PSW

4.1.1.5 Configuration software requirements (S7-1500)

Introduction

Below please find a list of the software versions which support the "Time-based IO" function.

Requirements

Configuration software	Requirements	Supported hardware components	Additional information
STEP 7 (TIA Portal) as of V13	• PROFINET IO IRT • For CM 4xIO-Link: IO-Link technology	• S7-1500 automation system • ET 200SP and ET 200MP distributed I/O systems • CM 4xIO-Link • TM Timer DIDQ 16x24V (as of V13 Update 3) • TM Timer DIDQ 10x24V (as of V13 Update 3)	TIA Portal online help
S7-PCT as of V3.2	—	• CM 4xIO-Link	S7-PCT online help

TIO instructions

The TIO instructions are specific function blocks for use of Time-based IO. The following TIO instructions are available:

- TIO_SYNC
- TIO_IOLink_IN

- TIO_IOLink_OUT
- TIO_DI
- TIO_DQ

As of V13 Update 3, the TIO instructions are available as integrated instructions in TIA Portal.

4.1.1.6 Technical implementation (S7-1500)

Introduction

The information below will help you better understand the core aspects of Time-based IO and find out which SIMATIC functions are used to implement the described aspects.

Synchronization of involved modules in PROFINET (shared time basis)

The Time-based IO technology uses isochronous mode for all involved PROFINET stations. Isochronous communication is made possible with PROFINET IRT (Isochronous Real-Time) and isochronous mode of the PROFINET stations.

Additional information on isochronous mode is available in the PROFINET with STEP 7 (http://support.automation.siemens.com/WW/view/en/49948856) and Isochronous Mode (http://support.automation.siemens.com/WW/view/en/15218045) manuals.

Several TIO modules can be synchronized to a shared time basis by using the isochronous PROFINET. The basis of the shared time basis for the TIO modules is the relative time TIO_Time.

TIO_Time

The TIO_Time is the central time basis to which all time stamps refer.

TIO_Time has the following properties:

- Shared time basis for all TIO modules which are synchronized with the TIO instruction TIO_SYNC.
- The time starts counting again with each CPU startup.
- TIO_Time has the data type LTime (e.g., LT#14s830ms652us315ns).
- All valid time stamps refer to TIO_Time:
  - Input time stamps of the TIO modules are converted to TIO_Time in TIO instructions TIO_IOLink_IN and TIO_DI.
  - Output time stamps are converted to output time stamps of the TIO modules in TIO instructions TIO_IOLink_OUT and TIO_DQ.

TIO instructions in the "Synchronous Cycle" OB

The TIO instructions must be called in a "Synchronous Cycle" OB.

You can find additional information in the chapter Programming.

Modes for updating the process image

In isochronous mode, you can influence the order of the update of the process image partition of the input and output data. In doing so, you can select the following program execution models:

- IPO model (application cycle factor = 1)
- OIP model (application cycle factor >= 1)

The abbreviations I, P, O stand for the following processes: I = Input, P= Processing, O = Output.

Additional information on the application cycle factor is available in the PROFINET with STEP 7 (http://support.automation.siemens.com/WW/view/en/49948856) Manual.

IPO model (application cycle factor = 1)

The user program is started after the delay time. Start by updating the corresponding process image partition of the inputs in the user program by calling the SYNC_PI system instruction. Processing is started next (for example, calculation of the time stamps). The corresponding process image partition of the outputs is updated at the end of the user program in the CPU by SYNC_PO.

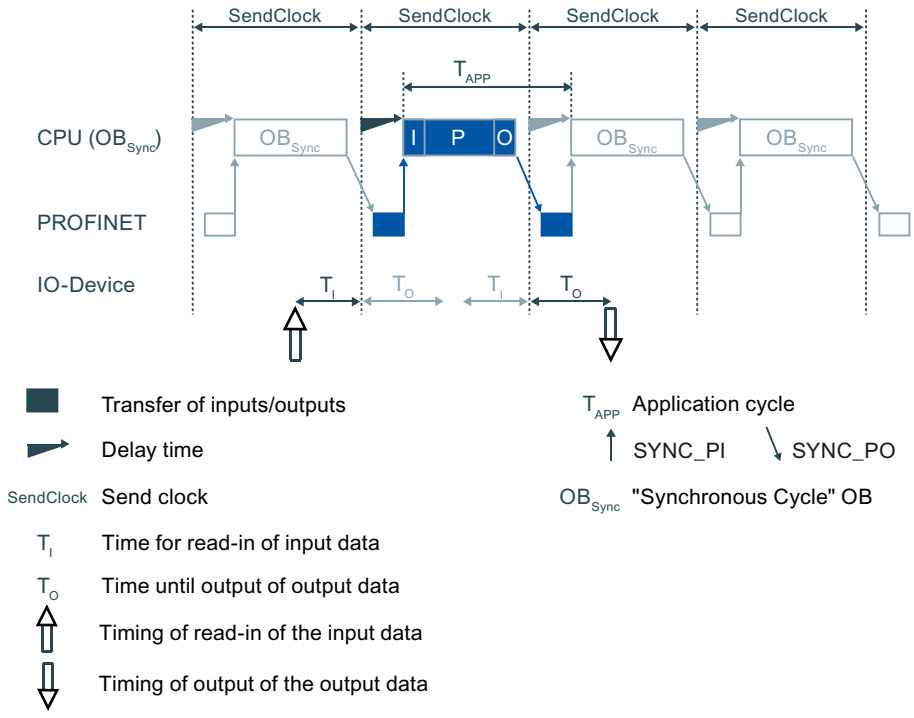

Properties of the IPO model:

- Supports shorter response times
- The application cycle may not be greater than the send clock.
  This gives the application less time than with the OIP model.

OIP model (application cycle factor >= 1)

The user program is started after the delay time. In PIP_Mode 0, the TIO_SYNC instruction updates the process image. In the other modes, you start by updating the corresponding process image partition of the outputs in the user program by calling the SYNC_PO system instruction. As a result, the output data that was calculated in the previous network cycle will become active during the next network cycle (T_O). Next the corresponding process image partition of the inputs is updated in the CPU by SYNC_PI. Processing starts after the data is transmitted (for example, calculation of the time stamps).

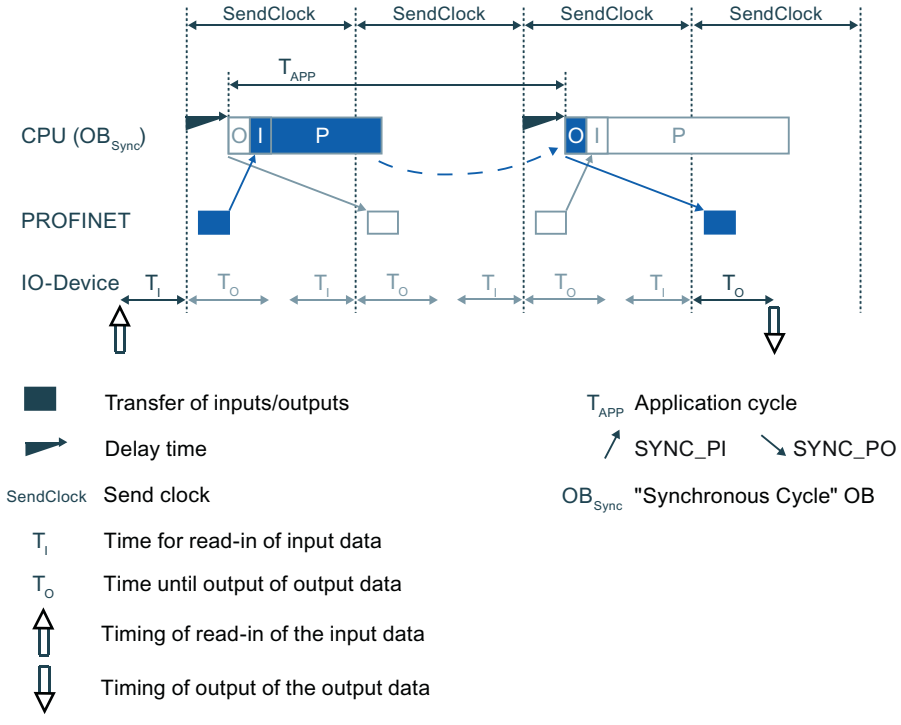

Properties of the OIP model:

- The response time is longer than with the IPO model.
- It is one application cycle longer than permitted by the send clock.
  This gives the application more time than with the IPO model.

Influence of accuracy

To estimate the accuracy you need the accuracy of

- TIO module
- Sensor/ actuator

Consult the data sheets of the respective module regarding the accuracy value.

You have to add the individual jitters of the TIO modules and the sensors/actuators. You can neglect any other influencing factors.

4.1.2 Configuring and parameter assignment (S7-1500)

4.1.2.1 Requirements (S7-1500)

Introduction

You need additional software components to use Time-based IO. You must also have created the standard configuration for your project.

Below you will learn more about the standard configuration for Time-based IO.

Requirements

In the TIA Portal:

- The project has been created.
- The CPU has been created and the parameters are assigned.
- ET 200 distributed I/O and modules have been created and the parameters are assigned.
- The connection has been created and the parameters are assigned via PROFINET.
- A "Synchronous Cycle" OB has been created.

In addition for IO-Link:

- S7-PCT is installed on the programming device/PC.
- IODD for IO-Link device (with time stamp function) are installed.

See also

System environment (Page 1158)

Configuration software requirements (Page 1159)

4.1.2.2 Settings for Time-based IO (S7-1500)

Below you will find an overview of which settings have to be made for which components of Time-based IO.

Table 4-1 Overview of settings for Time-based IO

Component	Can be set where (*)	Properties to be set	Additional information
PROFINET subnet	Properties of the PROFINET subnet > sync domain	Create sync domain or edit properties of sync domain • For example, set send clock (e.g., 2 ms)	• PROFINET with STEP 7 function manual (http://support.automation.siemens.com/WW/view/en/49948856) • Online help in the TIA Portal information system
		Specify devices of the sync domain: • Specify CPU as sync master. • Specify ET 200 interface module as sync slave with RT class "IRT".	
ET 200 station (with TIO module)	Properties of the PROFINET interface > isochronous communication	Assignment of the TIO modules to the isochronous PROFINET subnet	• PROFINET with STEP 7 function manual (http://support.automation.siemens.com/WW/view/en/49948856) • Manual Technology Module TM Timer DIDQ 16x24V (http://support.automation.siemens.com/WW/view/en/95153313) • Manual Technology Module TM Timer DIDQ 10x24V (http://support.automation.siemens.com/WW/view/en/95153951) • Online help in the TIA Portal information system
	Properties of the TIO module > isochronous mode	Activate isochronous mode	
TIO module	Properties of the TIO module > I/O addresses	A "Synchronous Cycle" OB has been assigned or created	
	Properties of the TIO module > I/O addresses	Assignment of I/O addresses to the process image partition (e.g., PIP1)	
	Properties of the TM Timer DIDQ > Basic parameters/Channel parameters	Parameter assignment for use of Timer DI and Timer DQ	
"Synchronous Cycle" OB	Properties of the "Synchronous Cycle" OB > Isochronous mode	Adjust application cycle, if necessary	
IO-Link Device (sensor/ actuator)	SIMATIC S7-PCT	Activate port of the IO-Link Device for Time-based IO	See chapter Configure IO-Link and activate Time-based IO (Page 1165)

* Does not describe the exact location but the topic in the configuration software.

Setting for Time-based IO

If you have no special response time requirements, the following setting is suitable as a starting point:

- Send clock: 2 ms
- Application cycle: 4 ms
- Assignment to the process image partition: PIP1
- PIP_Mode: 0 (OIP model)

FAQ

For more information, see the following FAQs in the Siemens Industry Online Support:

- Entry ID 109738186
- Entry ID 109736374

4.1.2.3 Configure IO-Link and activate Time-based IO (S7-1500)

Introduction

For Time-based IO, the ports of the IO-Link Master must be set to the required operating mode and suitable sensors/actuators must be connected to the ports.

Requirements

- S7-PCT is installed on the programming device/PC.
- IODD for IO-Link Device (with time stamp function) are installed.
  You can find additional information on installation in the online help for S7-PCT.
- The project is open in network view in the TIA Portal.
- The hardware catalog is open.

Procedure

1. Right-click the IO-Link Master in the TIA Portal and select "Start Device Tool".
   The Port Configuration Tool (S7-PCT) opens.
2. Select the connected IO-Link Device (sensors/actuators with time stamp function) from the device catalog for the ports of the IO-Link Master in S7-PCT.
   Use drag-and-drop to insert the IO-Link Device to the matching Port of the IO-Link Master.
3. Adapt the settings of the ports to match their respective task in the application.
   Possible port settings:
   - IO-Link Time based IN: The input signals of the sensor are given a time stamp.
   - IO-Link Time based OUT: The signal is output at the specified time.
4. Save the settings.
5. Download the Port configuration to the IO-Link Master.
6. Close S7-PCT.

Result

The IO-Link Master is configured with the connected sensor and actuator information.

4.1.3 Programming (S7-1500)

4.1.3.1 Overview of instructions (S7-1500)

Introduction

Time-based IO is used with special instructions (TIO instructions). The TIO_SYNC TIO instruction is responsible for synchronizing all involved TIO modules and creates a unique time basis (TIO_Time) to which all actions are referenced.

Depending on the type of TIO module, additional instructions undertake the reading in of process input signals with associated time stamps and/or the time-controlled output of process output signals.

Note

The TIO instructions use the time stamp functions of the TIO modules. For the other functions of the respective TIO module, no use of instructions is necessary.

TIO instructions

Instruction	Can be used with TIO module	Short description
TIO_SYNC	• CM 4xIO-Link • TM Timer DIDQ 16x24V • TM Timer DIDQ 10x24V	Synchronizes the TIO modules and provides the time basis for Time-based IO
TIO_IOLink_IN	• CM 4xIO-Link	Reads the input signals with time stamp from the IO-Link Device (sensor)
TIO_DI	• TM Timer DIDQ 16x24V • TM Timer DIDQ 10x24V	Detects the edges at the digital input (Timer DI) and supplies the associated time stamp
TIO_IOLink_OUT	• CM 4xIO-Link	Outputs time-controlled output signals at the IO-Link Device
TIO_DQ	• TM Timer DIDQ 16x24V • TM Timer DIDQ 10x24V	Outputs time-controlled edges at the digital output (Timer DQ)

4.1.3.2 Programming of Time-based IO (S7-1500)

Introduction

For use of Time-based IO, the TIO instructions must be called in a "Synchronous Cycle" OB. The application can also run in another OB. In this way, you can shorten the runtime of the "Synchronous Cycle" OB.

You need the following TIO instructions according to the required task:

TIO module	TIO instructions	
CM 4xIO-Link	• Per sensor, one TIO_IOLink_IN for read-in • Per actuator, one TIO_IOLink_OUT for output	One TIO_SYNC (for up to eight TIO modules)
TM Timer DIDQ	• Per digital input, one TIO_DI for read-in • Per digital output, one TIO_DQ for output	

This section below describes the programming of the CPU for Time-based IO.

Requirements

Hardware configuration in the TIA Portal:

- The TIO modules are assigned to an isochronous network.
- The TIO modules are assigned to a shared process image partition.
- The process image partition is assigned to a "Synchronous Cycle" OB.

Additional information on configuration of Time-based IO is available in the section Settings for Time-based IO (Page 1164).

Procedure

1. Create a TIO_SYNC TIO instruction in the "Synchronous Cycle" OB.
2. Connect all TIO modules to be synchronized at the TIO_SYNC TIO instruction using parameters HWID_1 to HWID_8.
   The HWID is located in HWCN under the module properties.
3. Configure the send clock at the TIO_SYNC TIO instruction.
   The send clock is listed in the properties of the ET 200 station.
4. Set the data update mode at the TIO_SYNC instruction at the PIP_Mode input parameter.
   The description of the modes is available in the chapter Technical implementation (Page 1160).
   The parameter assignment of the TIO_SYNC TIO instruction is complete.

5. For the following instructions, note the call sequence depending on the selected value for PIP_Mode:

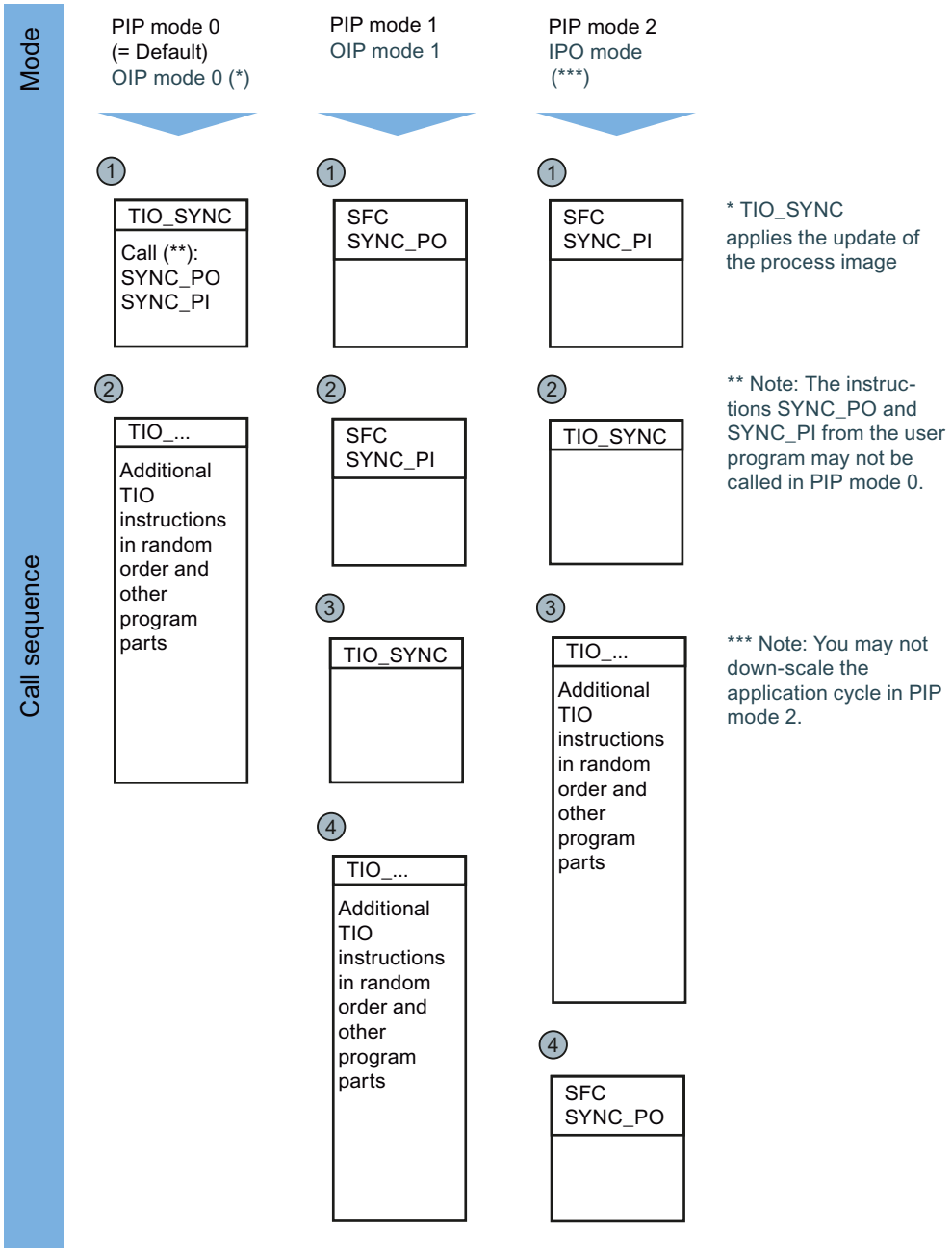

6. Add the TIO instructions for data read-in/output required for your application in the "Synchronous Cycle" OB.
7. Connect the TIO_SYNC_Data input to the output of the same name at TIO_SYNC at each TIO instruction for read-in and output.
8. Assign the input parameters TO (see properties of the ET 200 station), HWID (see module properties in HWCN), and Port/Channel" for the TIO instructions for data read-in/output. Programming for the Time-based IO functionality is complete.

9. Connect the Time-based IO functionality with your application.
10. Compile and download the entire project to the CPU.

Result

You have completed programming the use of Time-based IO.

Example

Below you see an application example with a TIO_IOLink_IN and a TIO_IOLink_OUT.

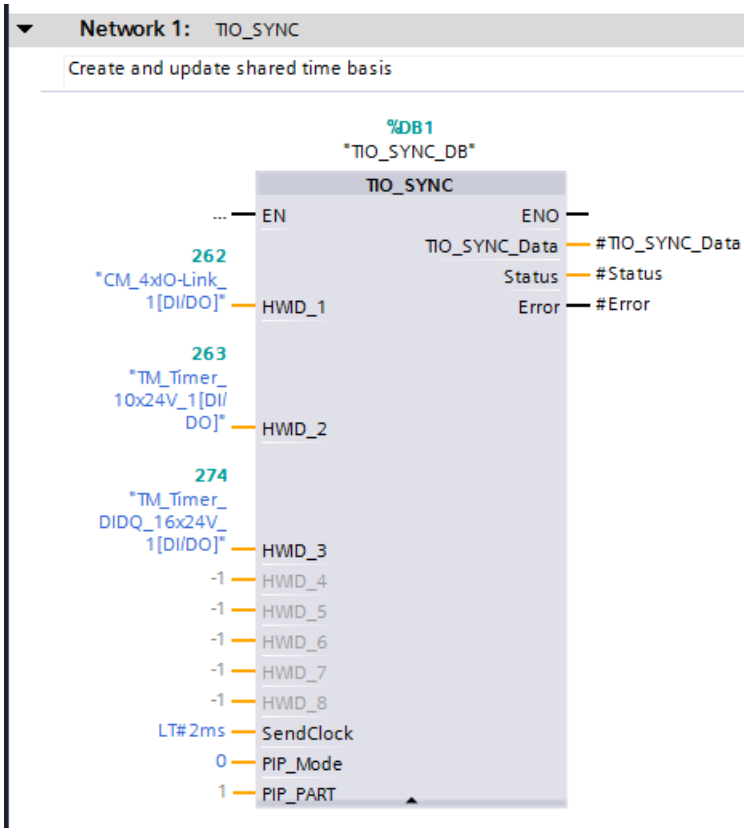

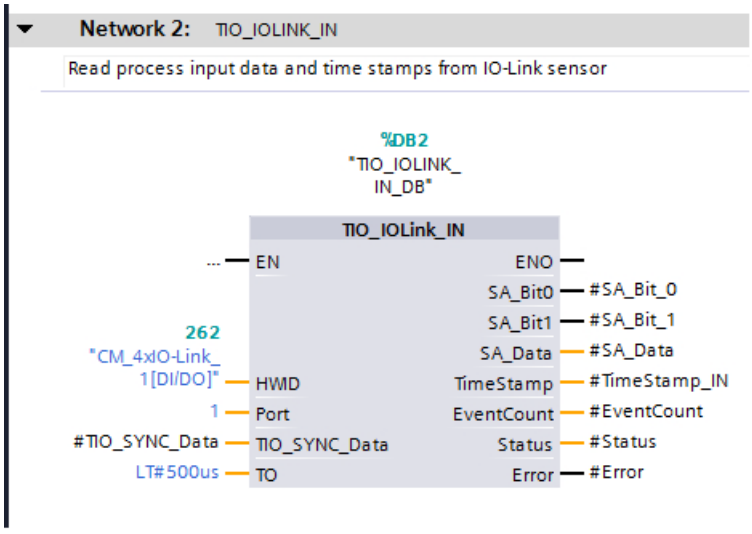
Network 2: TIO_IOLINK_IN
Read process input data and time stamps from IO-Link sensor
%DB2
"TIO_IOLINK_IN_DB"
TIO_IOLink_IN
EN
ENO
SA_Bit0
#SA_Bit_0
SA_Bit1
#SA_Bit_1
SA_Data
#SA_Data
262
"CM_4xIO-Link_1[DI/DO]"
HWID
TimeStamp
#TimeStamp_IN
1
Port
EventCount
#EventCount
#TIO_SYNC_Data
TIO_SYNC_Data
Status
#Status
LT#500us
TO
Error
#Error

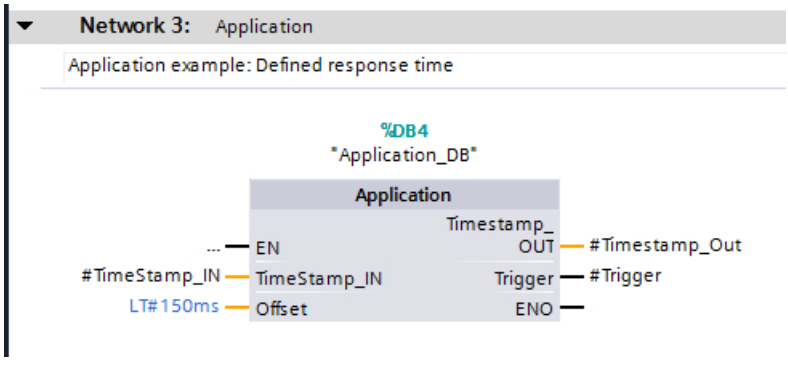
Network 3: Application
Application example: Defined response time
%DB4
"Application_DB"
Application
EN
Timestamp_OUT
#Timestamp_Out
#TimeStamp_IN
TimeStamp_IN
Trigger
#Trigger
LT#150ms
Offset
ENO

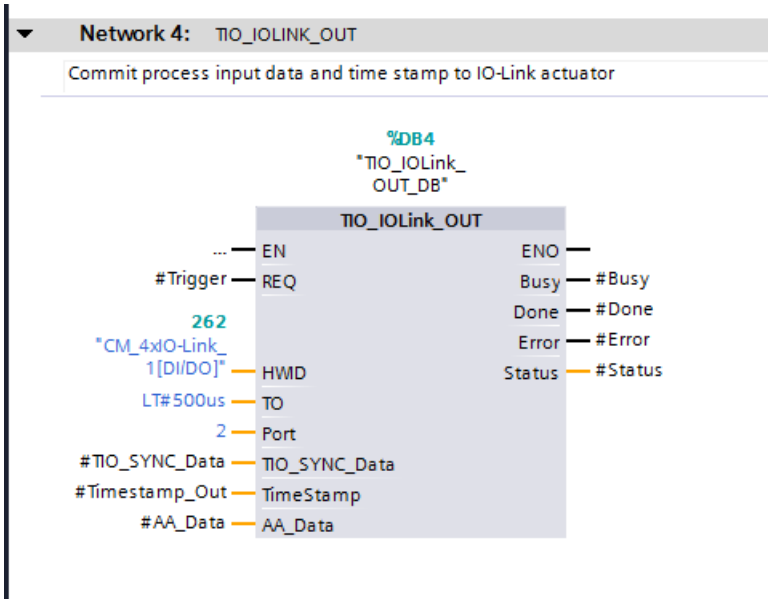
Network 4: TIO_IOLINK_OUT
Commit process input data and time stamp to IO-Link actuator
%DB4
"TIO_IOLink_OUT_DB"
TIO_IOLink_OUT
EN
ENO
#Trigger
REQ
Busy
#Busy
Done
#Done
262
"CM_4xIO-Link_1[DI/DO]"
Error
#Error
HWID
Status
#Status
LT#500us
TO
2
Port
#TIO_SYNC_Data
TIO_SYNC_Data
#Timestamp_Out
TimeStamp
#AA_Data
AA_Data

FAQ

For more information, see the following FAQs in the Siemens Industry Online Support:

- Entry ID 109738186
- Entry ID 109736374

4.1.4 Using the technology module (S7-1500)

4.1.4.1 Configuration and parameter assignment of TIO-Modul (S7-1500)

Adding a technology module to hardware configuration (S7-1500)

Requirements

- The project has been created.
- The CPU has been created.
- The ET 200 distributed I/O has been created.

Procedure

1. Open the device configuration of the CPU or IM.
2. Select a module rack.
3. Select the technology module from the module catalog:
   "Technology modules > Time-based IO > Technology module > Article number".
4. Drag the technology module to the required slot in the module rack.

Result

The new technology module is displayed under "Local modules" or "Distributed I/O" in the project tree.

Open Hardware configuration (HWCN) (S7-1500)

Opening via the project tree

Proceed as follows:

1. Open the "Local modules" or "Distributed I/O" folder in the project tree.
2. Double-click the technology module in the project tree.

Opening from the device view

Proceed as follows:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Click on the technology module.

Basic parameters (S7-1500)

Channel configuration (TM Timer DIDQ 16x24V)

You use the channel configuration to specify for TM Timer DIDQ 16x24V which combination of digital inputs and outputs you would like to use.

You can select from the following configurations:

- 0 inputs, 16 outputs (default)
- 3 inputs, 13 outputs
- 4 inputs, 12 outputs
- 8 inputs, 8 outputs

Note

Unused outputs are available as encoder supply.

Unused inputs are available as digital input without technology function. You specify the input delay of an input together with the parameters of the corresponding output channel.

PWM period

Use this parameter to specify the period duration for the signals at the digital outputs with "PWM" mode.

You can select from the following period durations:

- 10 ms (default)
- 5 ms
- 2 ms
- 1 ms
- 0.5 ms
- 0.2 ms

Note

You specify the pulse-pause ratio in the control interface. The pulse-pause ratio is a percentage and is evaluated with an accuracy of about 3 %.

Reaction to CPU STOP

Use this parameter to specify the reaction of the technology module to a CPU STOP.

You can select from the following options:

Reaction to CPU STOP	Meaning
Output substitute value (default)	The technology module outputs the configured substitute values at the digital outputs until the next CPU STOP-RUN transition. The technology module is returned to its startup state after a STOP-RUN transition: If you are using counters, the counting values are set to 0 and the digital outputs switch according to the parameter assignment and the setpoints.
Keep last value	The technology module outputs the values at the digital outputs that were valid when the transition to STOP took place until the next CPU STOP-RUN transition. The last valid period duration with the last valid pulse-pause ratio is output for a configured pulse width modulation until the next STOP-RUN transition. The technology module is returned to its startup state after a STOP-RUN transition: If you are using counters, the counting values are set to 0 and the digital outputs switch according to the parameter assignment and the setpoints.

Enable diagnostic interrupts

The technology module triggers diagnostic interrupts in the event of certain errors. The technology module can trigger additional diagnostic interrupts when you enable the diagnostic interrupts. These diagnostic interrupts are processed in an interrupt OB.

You use this parameter to specify whether or not the technology module triggers the additional diagnostic interrupts when the respective errors occur.

See the device manual for the technology module to find out which errors during operation can trigger a diagnostic interrupt. The diagnostic interrupts are not enabled in the default setting.

Channel parameters (S7-1500)

Overview (S7-1500)

The digital inputs and outputs are divided into groups with group-specific parameters in the channel parameters.

Each group is one of the following group types:

- DQ group (Page 1174)
- DQ/DI group (Page 1176)
- DI group (Page 1180)

The available group types for a TM Timer DIDQ 16x24V depend on the channel configuration in the basic parameters.

DQ group (S7-1500)

You can configure the following parameters for the digital outputs of a DQ group:

Operating mode

Use this parameter to specify the technological function of the respective digital output .

You can select from the following options:

Operating mode	Meaning
Timer DQ (default)	The technology module outputs edges at the respective digital output at defined times. This setting uses Time-based IO and does not require isochronous mode. If you are using the SIMOTION control system, use this setting for a cam application.
Oversampling	The respective digital output outputs 32 states at regular intervals for each application cycle (for example, OB61). The 32 states are specified by means of the control interface. This setting requires isochronous mode.
PWM	The technology module switches the respective digital output with the period duration specified in the basic parameters and the pulse-pause ratio from the control interface. The pulse-pause ratio is a percentage and is evaluated with an accuracy of about 3 %.

Substitute value

With this parameter you specify for the "Output substitute value" behavior which value the technology module is to output at the respective digital output in the event of a CPU STOP.

The default setting is "0".

High-speed output (0.1 A)

Use this parameter to specify whether or not the respective digital output works as fast push-pull or as sourcing output.

You can select from the following options:

Option	Meaning
Activated (default)	The digital output works as fast push-pull switch and can carry a rated load current of 0.1 A. A push-pull switch is alternately switched to DC 24 V and ground. This makes for very fast edges.
Deactivated	The digital output works as 24 V sourcing output in reference to M and can carry a rated load current of 0.5 A.

Invert DI/DQ(TM Timer DIDQ 16x24V)

You can invert the 24 V signals at the digital output and at the corresponding digital input together in order to adapt them to the process.

The two signals are not inverted in the default setting.

Invert DQ

You can invert the 24 V signal at the digital output to adapt it to the process.

The signal is not inverted in the default setting.

Input delay

By configuring the input delay, you suppress interference at the digital input, which is available as input without technology function. Signals with a pulse duration below the configured input delay are suppressed.

You can choose from the following input delays:

- No
  (input delay of 4 µs, minimum pulse width of 3 µs)
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs. To increase the accuracy of the time stamp function, we recommend the use of shielded cables even for longer input delays.

Note

The input delay is subject to a tolerance that you can find in the technical specifications of the technology module manual.

DQ/DI group (S7-1500)

A DQ/DI group consists of a digital output and one or two digital inputs. You can configure the following parameters for the group:

Configuration DQ/DI group

You use this parameter to specify if the DI is used as enable input for the DQ.

You can select from the following options:

Option	Meaning	Additional option-specific parameters
Timer DQ with enable input (default)	The technology module outputs edges time-controlled at the DQ. The DQ is enabled by the DI. The operating mode of the DQ is "Timer DQ" and cannot be changed. This setting uses Time-based IO and does not require isochronous mode. If you are using the SIMOTION control system, use this setting for a cam application.	Substitute value High-speed output (0.1 A) Invert DI/DQ(TM Timer DIDQ 16x24V) Invert DQ(TM Timer DIDQ 10x24V) Input delay HW enable
Use input/output individually	The DQ and the DI are configured and used independent of one another.	Operating mode (DQ) Substitute value High-speed output (0.1 A) Invert DI/DQ(TM Timer DIDQ 16x24V) Invert DQ(TM Timer DIDQ 10x24V) Operating mode (DI) Input delay Invert DI

Configuration DQ/DI group (DQ2/DI2/DI3 of the TM Timer DIDQ 10x24V)

This parameter is used to specify how DQ2, DI2 and DI3 of the TM Timer DIDQ 10x24V are used.

You can select from the following options:

Option	Meaning	Additional option-specific parameters
Incremental encoder (A, B phase-shifted) (default)	DQ2 is used individually. An incremental encoder with phase-shifted 24V A and B signals is connected to DI2 and DI3. The counter (DI2) always evaluates both signals four times.	Operating mode (DQ) Substitute value High-speed output (0.1 A) Invert DQ Invert counting direction
Timer-DI2 with enable input DI3	Time stamps are detected at defined edges at DI2. DI2 is enabled by DI3. The operating mode of DI2 is "Timer DI" and cannot be changed. This setting uses Time-based IO and does not require isochronous mode. If you are using the SIMOTION control system, use this setting for a probe application.	Operating mode (DQ) Substitute value High-speed output (0.1 A) Invert DQ Input delay Invert DI HW enable

Option	Meaning	Additional option-specific parameters
Timer DQ2 with enable input DI2	The technology module outputs edges time-controlled at DQ2. DQ2 is enabled by DI2. The operating mode of the DQ2 is "Timer DQ" and cannot be changed. This setting uses Time-based IO and does not require isochronous mode. If you are using the SIMOTION control system, use this setting for a cam application.	Substitute value High-speed output (0.1 A) Invert DQ Input delay HW enable Operating mode (DI) Invert DI
Use input/output individually	The DQ and the DI are configured and used independent of one another.	Operating mode (DQ) Substitute value High-speed output (0.1 A) Invert DQ Operating mode (DI) Input delay Invert DI

Operating mode (DQ)

Use this parameter to specify the technological function of the digital output.

You can select from the following options:

Operating mode	Meaning
Timer DQ (default)	The technology module outputs edges at defined times at DQ. This setting uses Time-based IO and does not require isochronous mode. If you are using the SIMOTION control system, use this setting for a cam application.
Oversampling	The technology module outputs 32 states consecutively at regular intervals at DQ for each application cycle (for example OB61) . The 32 states are specified by means of the control interface. This setting requires isochronous mode.
PWM	The technology module switches the DQ with the period duration specified in the basic parameters and the pulse-pause ratio from the control interface. The pulse-pause ratio is a percentage and is evaluated with an accuracy of about 3 %.

Substitute value

This parameter lets you specify which value the technology module is to output at the DQ in the event of a CPU STOP under "Output substitute value".

The default setting is "0".

High-speed output (0.1 A)

Use this parameter to specify whether or not the DQ works as fast push-pull or as sourcing output.

You can select from the following options:

Option	Meaning
Activated (default)	The DQ works as fast push-pull switch and can carry a rated load current of 0.1 A. A push-pull switch is alternately switched to DC 24 V and ground. This makes for very fast edges.
Deactivated	The DQ works as 24 V sourcing output in reference to M and can carry a rated load current of 0.5 A.

Invert DI/DQ(TM Timer DIDQ 16x24V)

You can invert the 24 V signals of the DQ and the corresponding DI together in order to adapt them to the process.

The two signals are not inverted in the default setting.

Invert DQ(TM Timer DIDQ 10x24V)

You can invert the 24 V signal of the DQ to adapt it to the process.

The signal is not inverted in the default setting.

Input delay

You use this parameter to suppress signal interference at the digital inputs. Changes to the signal are only detected if they remain stable for longer than the configured input delay time.

You can choose from the following input delays:

- No
  (input delay of 4 µs, minimum pulse width of 3 µs)
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms

The input delay has no effect on the function for signal evaluation set at the digital input:

Function	Influence of the input delay
Timer DI	The detected time stamp is moved by the input delay.
Incremental encoder (A, B phase-shifted)	The counter value that was valid at time T_i minus the input delay is returned.
Counters	
Oversampling	The detected states are moved together by the input delay.

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs. To increase the accuracy of the time stamp function, we recommend the use of shielded cables even for longer input delays.

Note

The input delay is subject to a tolerance that you can find in the technical specifications of the technology module manual.

HW enable

You use this parameter to specify if the Timer DQ is enabled by the level or the edges at DI.

You can select from the following options:

Enable	Meaning	Additional option-specific parameters
Level-triggered (default)	The DQ is enabled by the DI level.	Level selection for HW enable
Edge-triggered	This setting is available when you use the SIMOTION control system.	—

The figure below shows an example of the output of positive and negative edges at DQ0 with enable of DI0 by the high level of DI1:

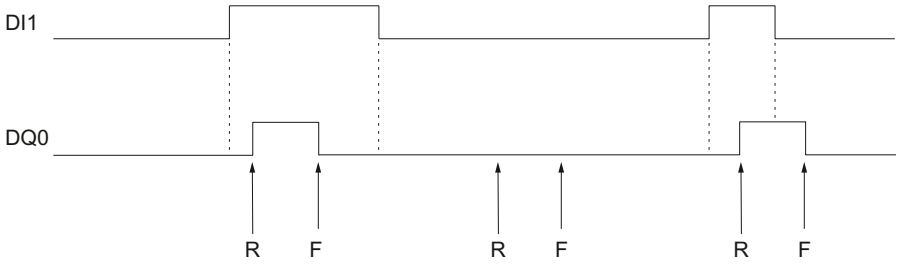

R Specified time of a positive DQ0-edge

F Specified time of a negative DQ0-edge

Level selection for HW enable

You use this parameter to specify the level at which the DQ is enabled for a level-triggered HW enable.

You can select from the following options:

Level	Meaning
Active with high level (default)	The DQ is enabled if the DI is set.
Active with low level	The DQ is enabled if the DI is not set.

Operating mode (DI)

Use this parameter to specify the technological function of the DIm+1.

You can select from the following options:

Operating mode	Meaning
Counters	An incremental encoder or sensor is connected to the DI. The counter counts the positive or negative edges at the DI. See the device manual for the technology module to find out which digital inputs can be used as counter.
Timer DI (default)	Time stamps are read in for associated time stamps at the DI at defined edges. This setting uses Time-based IO and does not require isochronous mode. If you are using the SIMOTION control system, use this setting for a probe application.
Oversampling	The technology reads the status of the DI for each application cycle (for example, OB61) at 32 points in time at regular intervals. The 32 states are returned at the same time. This setting requires isochronous mode.

Invert DI

You can invert the 24 V signal of the DI to adapt it to the process.

The signal is not inverted in the default setting.

DI group (S7-1500)

A DI group consists of two neighboring digital inputs DIm and DIm+1. You can configure the following parameters for the group:

Configuration DI group

Use this parameter to specify how the group is used.

You can select from the following options:

Option	Meaning	Additional option-specific parameters
Incremental encoder (A, B phase-shifted) (default)	An incremental encoder with phase-shifted 24V A and B signals is connected to both digital inputs. The counter of the DI group always evaluates both signals four times.	Inverting the count direction
Timer-DI with enable input	Time stamps are detected at defined edges at DIm. DIm is enabled by DIm+1. The operating mode of DIm is "Timer DI" and cannot be changed. This setting uses Time-based IO and does not require isochronous mode. If you are using the SIMOTION control system, use this setting for a probe application.	Input delay Invert HW enable
Use inputs individually	The two digital inputs are configured and used independent of one another.	Operating mode Input delay

Inverting the count direction

You can invert the count direction to adapt it to the process. The technology module inverts the B signal of the incremental encoder at DIm+1. The respective counter then interprets the edges of both signals as backward pulses.

The figure below shows an example of counter pulses in forward and backward direction:

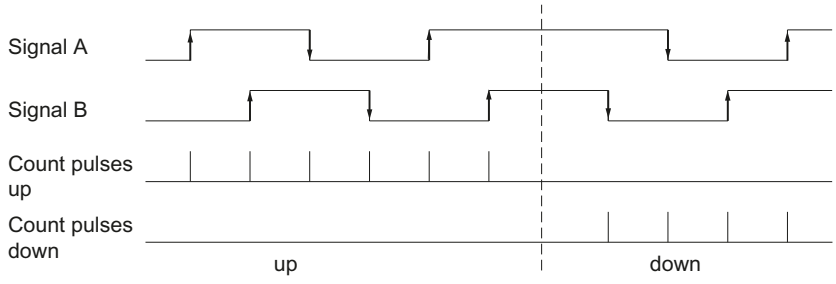

The count direction is not inverted in the default setting.

Input delay

You use this parameter to suppress signal interference at the digital inputs. Changes to the signal are only detected if they remain stable for longer than the configured input delay time.

You can choose from the following input delays:

- No
  (input delay of 4 µs, minimum pulse width of 3 µs)
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms

The input delay has no effect on the function for signal evaluation set at the digital input:

Function	Influence of the input delay
Timer DI	The detected time stamp is moved by the input delay.
Incremental encoder (A, B phase-shifted)	The counter value that was valid at time T_i minus the input delay is returned.
Counters	
Oversampling	The detected states are moved together by the input delay.

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs. To increase the accuracy of the time stamp function, we recommend the use of shielded cables even for longer input delays.

Note

The input delay is subject to a tolerance that you can find in the technical specifications of the technology module manual.

Invert

You can invert the 24 V signal of the DIm to adapt it to the process.

The signal is not inverted in the default setting.

HW enable

You use this parameter to specify whether the Timer DI is enabled by the level or the edges at DIm+1.

You can select from the following options:

Enable	Meaning	Additional option-specific parameters
Level-triggered (default)	The DIm is enabled by the DIm+1 level.	Level selection for HW enable
Edge-triggered	This setting is available when you use the SIMOTION control system.	—

The figure below shows an example for the detection of time stamps at positive and negative edges with enable of the DI0 through the high level of the DI1:

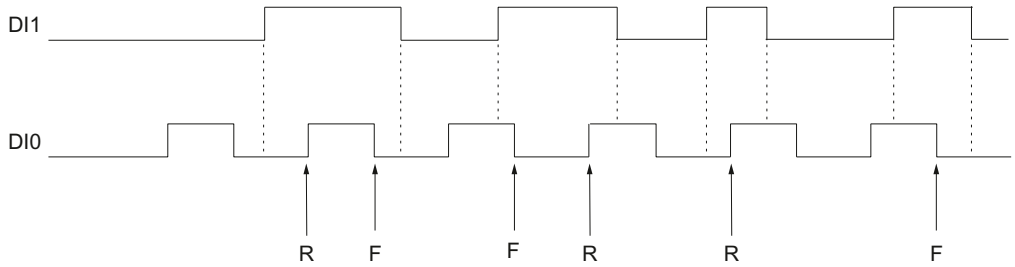

R Associated time stamp detected at positive DI0-edge

F Associated time stamp detected at negative DI0 edge

Level selection for HW enable

You use this parameter to specify the level at which DIm is enabled for a level-triggered HW enable.

You can select from the following options:

Level	Meaning
Active with high level (default)	DIm is enabled if the DIm+1 is set.
Active with low level	DIm is enabled if the DIm+1 is not set.

Operating mode

Use this parameter to specify the technological function of the respective digital input.

You can select from the following options:

Operating mode	Meaning	Additional option-specific parameters
Counters	An incremental encoder or sensor is connected to DIm. The counter counts the positive or negative edges at DIm. See the device manual for the technology module to find out which digital inputs can be used as counter.	Signal evaluation for counters
Timer DI (default)	The associated time stamps are read in at the respective digital input at defined edges. This setting uses Time-based IO and does not require isochronous mode. If you are using the SIMOTION control system, use this setting for a probe application.	Invert
Oversampling	The technology module detects the status of the respective digital input for each application cycle (for example, OB61) at 32 points in time at regular intervals. The 32 states are returned at the same time. This setting requires isochronous mode.	Invert

Signal evaluation for counters

You use this parameter to specify if the counter counts the positive or the negative edges at DIm.

The positive edges are counted in the default setting.

4.1.4.2 TIO-Modul Online & Diagnostics (S7-1500)

Displaying and evaluating diagnostics (S7-1500)

The online and diagnostics view enables hardware diagnostics. You can also

- Obtain information on the technology module (e.g., Firmware version and serial number)
- Execute a firmware update if required

Procedure

To open the display editor for the diagnostic functions, follow these steps:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Right-click on the technology module and select "Online & Diagnostics".
4. Select the required display in the diagnostics navigation.

Additional information

Additional information on the diagnostic alarms and possible remedies can be found in the technology module device manual.

Pulse output (S7-1500) 5

5.1 Basic principles of pulse output (S7-1500)

5.1.1 Overview of modules and properties (S7-1500)

Modules

The following table provides an overview of the performance features of the pulse output modules for the S7-1500 system.

Property	S7-1500		
	Technology module	Compact CPU	Technology module
	TM Timer DIDQ 16x24V	CPU 1511C-1 PN, CPU 1512C-1 PN	TM PTO 4
Number of pulse generators	16	4	4
Operating mode Oversampling	X	—	—
Operating mode Pulse width modulation PWM	X	X (Page 1256)	—
Operating mode Pulse output	—[1]	—	—
Operating mode Pulse train	—[1]	—	—
Operating mode On/Off delay	—	—	—
Operating mode Frequency output	—	X (Page 1256)	—
Operating mode PWM with DC motor	—	—	—
Operating mode Pulse Train Output (PTO)	—	X (Page 1256)	X
Sequence counter	—	—	—
High-speed output supported	X	X (Page 1266)	—
Ditheringsupported	—	—	—
Current measurement / control	—	—	—
Support for Motion Control	—	X	X
HW enable	—	—	X
Parallel operation of 2 channels	—	—	—
Control of digital outputs with the user program	X	X (Page 1267)	—
Support for isochronous mode on PROFINET	X	—	X
Diagnostic interrupt support	X	X	X

[1] You can implement a comparable function with the timer function.

The following table provides an overview of the performance features of the modules for pulse output for the ET 200SP system.

Property	ET 200SP		
	Technology module		Digital output module
	TM Timer DIDQ 10x24V	TM Pulse 2x24V	DQ 4x24VDC/ 2A HS
Number of pulse generators	6	2	4
Operating mode Oversampling	X	—	X
Operating mode Pulse width modulation PWM	X	X	X
Operating mode Pulse output	—[1]	X	—
Operating mode Pulse train	—[1]	X	—
Operating mode On/Off delay	—	X	—
Operating mode Frequency output	—	X	—
Operating mode PWM with DC motor	—	X	—
Operating mode Pulse Train Output (PTO)	—	—	—
Sequence counter	—	X	—
High-speed output supported	X	X	—
Ditheringsupported	—	X	—
Current measurement / control	—	X	—
Support for Motion Control	—	—	—
HW enable	—	X	—
Parallel operation of 2 channels	—	X	—
Control of digital outputs with the user program	X	X	X
Support for isochronous mode on PROFINET	X	X	X
Diagnostic interrupt support	X	X	X

[1] You can implement a comparable function with the timer function.

5.1.2 Basic principles of pulse output (TM Pulse 2x24V) (S7-1500)

5.1.2.1 Overview (S7-1500)

Operating modes and functions

The TM Pulse 2x24V has two channels. You configure the operating modes with STEP 7 (TIA Portal) or STEP 7.

You can select one of the following six operating modes for each channel:

- Pulse output
- Pulse width modulation PWM
- Pulse train
- On-/Off-delay
- Frequency output
- PWM with DC motor

In addition to the operating mode, the technology module has the following functions:

Function	Explanation
High-speed output	You can generate a minimum pulse duration of 1.5 µs at a current of 100 mA. Without this function, the minimum pulse duration is 10 µs with a current of max. 2 A (two channels) or 4 A (one channel).
Sequence counter	The sequence counter counts completed pulse output sequences and outputs a feedback signal.
Current measurement	For the operating modes "PWM" and "pulse train"
Current control	For the output current in PWM operating mode
Dithering	Superimposing a Dither signal on the PWM output to enhance proportional valve control
Control using the user program	You can control the digital outputs, channel-by-channel, directly with the user program.
Parallel connection of channels	Parallel connection of both channels enables a logical channel which can drive an output current of 4 A.
Reaction to CPU STOP	You can configure the reaction to a CPU STOP.
Diagnostics	The technology module enables module diagnostics and channel error detection.

The figure below shows two-channel operation.

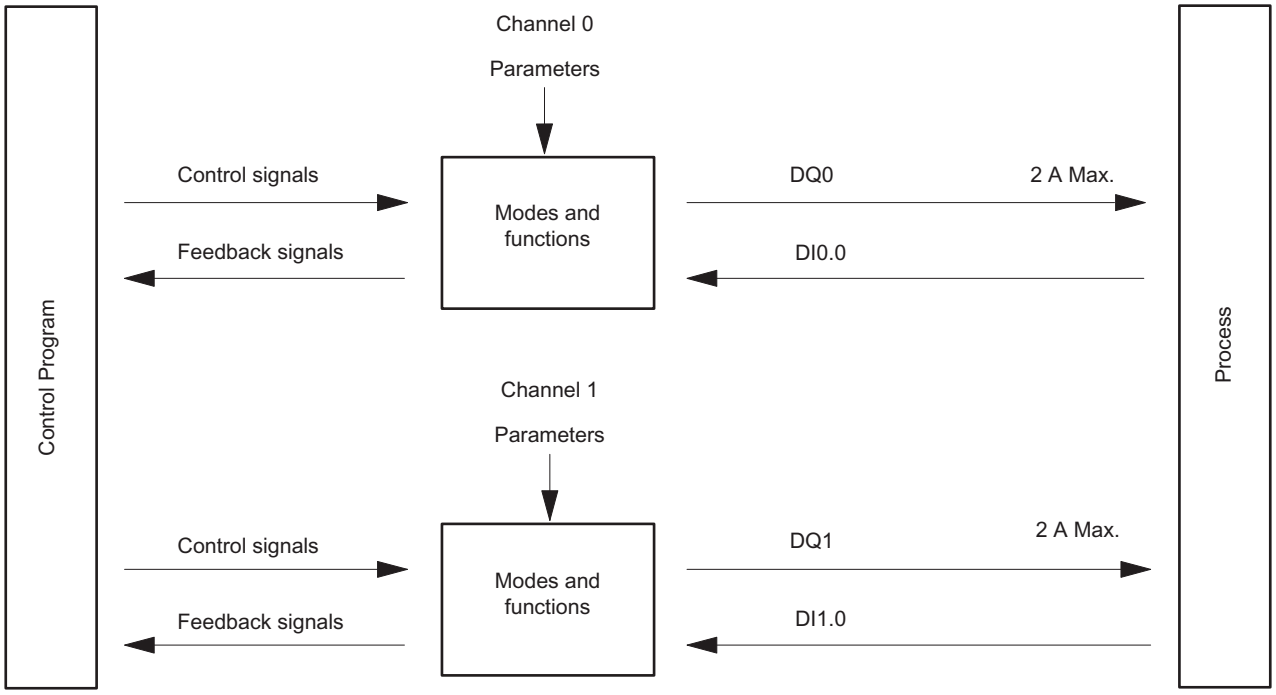

Figure 5-1 Two-channel operation

The figure below shows single-channel operation.

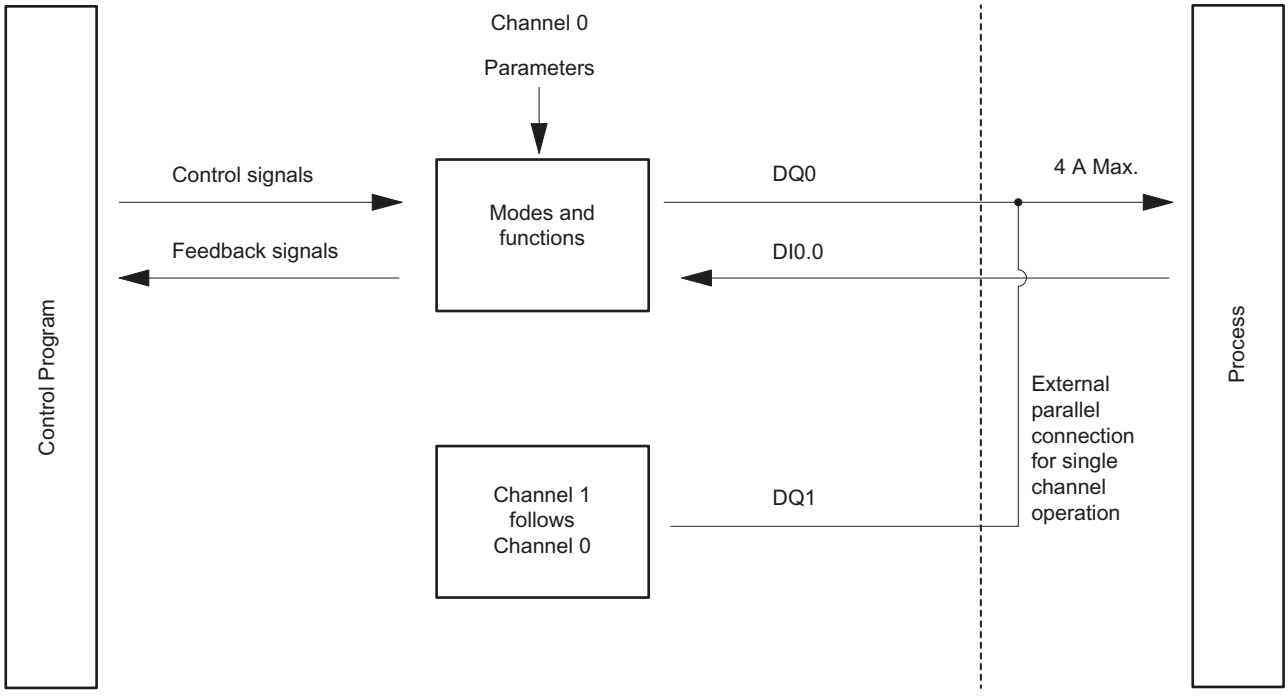

Figure 5-2 Single-channel mode

Interfaces to the control program and the controlled process

The TM Pulse 2x24V has the following I/O BaseUnit pin connections to the controlled process:

Channel 0:

- DI0.0 (digital input 0)
- DQ0.A and DQ0.B (digital output 0)
  - There are two connections for each channel output (DQ0.A and DQ0.B). Load wiring depends on the operating mode assignment of a channel.

Channel 1:

- DI1.0 (digital input 1)
- DQ1.A and DQ1.B (digital output 1)

You can find a table with configuration parameters in section "Parameter assignment and structure of parameter data records".

The operating modes and functions can be be modified and monitored using control and feedback signals in the control program. You can find these parameters in the section "Control and feedback interface".

In the following sections, you will find:

- Description of operations
- The relevant parameters
- The relevant control and feedback signals

The "description of operations" for operating modes and functions applies to both channels. The terms DIn.0, DQn.A und DQn.B are used in descriptions that apply to both channels.

5.1.2.2 Pulse output (single pulse) mode (S7-1500)

Definition

As soon as the assigned on-delay has expired, TM Pulse 2x24V issues a pulse at the digital output DQn.A (output sequence) for the pulse duration specified by you.

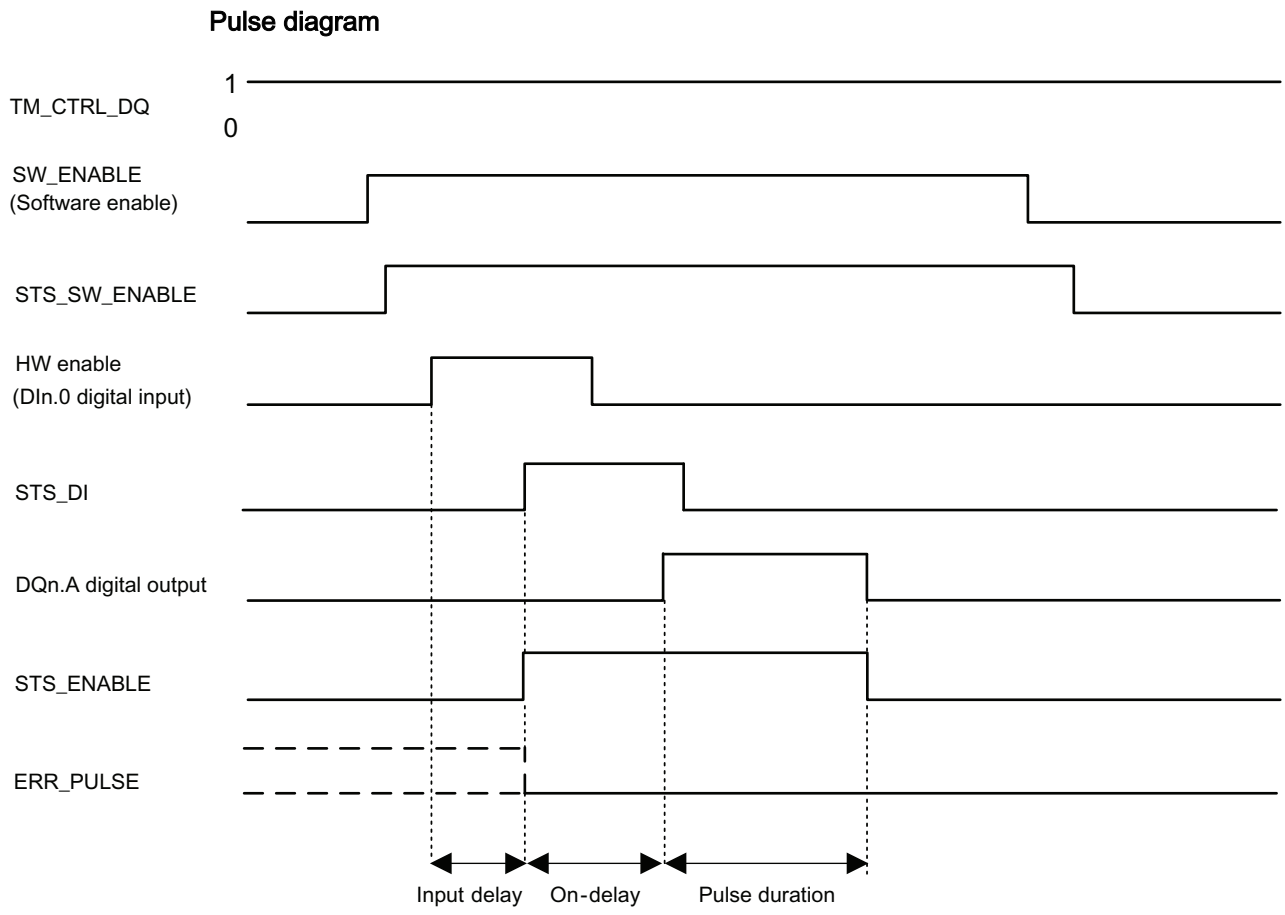

Figure 5-3 The output sequence of the pulse output operating mode uses an optional HW enable signal to start the output sequence

The pulse diagram above has the "DI function" parameter which is set to "HW enable". For the other option the "DI function" is set to "Input". If the parameter "DI function" parameter is set to "Input", the on-delay begins with the rising edge of SW_ENABLE.

Starting the output sequence

The control program must output the enable for the output sequence by using the software enable (SW_ENABLE 0 → 1.)

The STS_SW_ENABLE feedback bit indicates that a software enable in TM Pulse 2x24V is pending.

You can also set the digital input DIn.0 of a TM Pulse 2x24V channel as a hardware enable (HW enable) via the parameter "DI function". The input delay (noise filter) of the hardware enable can be configured via the "Input delay" parameter.

If you want to use the hardware enable, it must be combined with the software enable. In case of an activated software enable the output sequence starts with the first rising edge of the Hardware enable. Additional rising edges of the hardware enable are ignored during the current output sequence. If the HW enable switches to the "High" status (rising edge) and remains "High" ("On") for the duration of the input delay, the on-delay is started and STS_ENABLE is

set. As soon as the on-delay has expired, the pulse is output with the assigned pulse duration. The output sequence is terminated with the pulse end and STS_ENABLE is deleted.

If you reduce the pulse duration to an already expired duration, the ERR_PULSE signal outputs a pulse output error and the output sequence is stopped. To continue the pulse output you have to restart the output sequence after the occurrence of an ERR_PULSE error. The next time the output sequence starts, the TM Pulse 2x24V deletes the ERR_PULSE feedback bit.

Note

Output control signal TM_CTRL_DQ of the technology module

- If TM_CTRL_DQ = 1, the TM Pulse 2x24V takes over the control and generates pulse sequences at the DQn.A outputs.
- If TM_CTRL_DQ = 0, the CPU has the power of control and the program can set the DQn.A/DQn.B outputs directly with the control bits SET_DQA/SET_DQB.

Aborting the output sequence

When the software enable is deactivated (SW_ENABLE = 1 → 0), the current output sequence is aborted and the last period duration is not completed. STS_ENABLE and the digital output DQn.A are set to 0 immediately.

A renewed pulse output is only possible after a restart of the output sequence.

Truth table

Software enable SW_ENABLE	Function parameter DI	Hardware enable (digital input DIn.0)	Digital output DQn.A (if TM_CTRL_DQ = 1)	STS_ENABLE	Output sequence
1	HW_ENABLE	0 → 1 and remains 1 during the input delay Only active with the first rising edge, further rising edges are ignored, no start.	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0 → 1	Input	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	HW_ENABLE or input	Any status	0	0	Canceling
1	HW_ENABLE or input	Any status	0, if on-delay has not expired or if the pulse duration has expired. 1, if the on-delay has expired and the pulse duration has not expired.		-
0 → 1	HW_ENABLE	0	0	0	-

Setting and changing the pulse duration

The control program can set the pulse duration directly via the OUTPUT_VALUE control interface field as a DWord numerical value in microseconds.

- Rapid pulse outputs activated, from 2 µs to 85 000 000 µs
- Rapid pulse outputs deactivated, from 10 µs to 85 000 000 µs

If you change the pulse duration during a running output sequence, the already expired pulse duration is deducted from the new pulse duration and the pulse output is continued.

Reducing the pulse duration

If you have reduced the pulse duration to a time that is shorter than the time already output, the output sequence is terminated. STS_ENABLE and the DQn.A digital output are deleted, and the ERR_PULSE status bit is set. At the next output sequence, the ERR_PULSE status bit is cleared.

Setting and changing the on-delay

- Continuous update
  The on-delay can be continuously controlled via the control interface. The MODE_SLOT bit must be 1 (continuous update). LD_SLOT must be set to the value 2 (for on-delay). Set the on-delay to a value between 0 µs and 85 000 000 µs in the SLOT control interface field.
- Individual updating
  Set the on-delay to a value between 0 µs and 85 000 000 µs in the configuration parameters. Alternatively, execute an individual updating via the control interface. MODE_SLOT must be 0 (individual updating). LD_SLOT must be set to the value 2 (for on-delay). Set the on-delay to a value between 0 µs and 85 000 000 µs in the SLOT control interface field.

If you change the on-delay factor during the output sequence, then the new on-delay is activated with the next output sequence.

Additional information on the using the SLOT value is available under Handling of the SLOT parameter.

Use of the sequence counter

The sequence counter counts completed pulse output sequences and makes the SEQ_CNT counter value variable available in the feedback interface. The sequence counter can count short output sequences that are executed too fast and can therefore not be monitored by the program. You can find additional information under Sequence counter.

Isochronous mode

General information is available under Function: Isochronous mode.

In the "Pulse output" operating mode, isochronous mode does not have any influence on the functionality.

If you want to synchronize the output sequence with T_o , set the "Function DI" parameter to "Input" and the pulse output sequence starts at T_o.

Pulse output parameter

Pulse output parameter	Meaning	Value range	Default
Operating mode	0 = Setting the "Pulse output" operating mode	0 = Pulse output 1 = Pulse width modulation 2 = Pulse train 3 = On-/Off-delay 4 = Frequency output 5 = DC motor	1
High-speed pulse output [1]	In activated state the output supports higher frequencies for smaller loads.	0 = Deactivated 1 = Activated	Deactivated
Function DI	You can use the digital input DIn.0 as an input or as a hardware enable. The signal at the DIn.0 is interpreted by the TM Pulse 2x24V as the start of the output sequence after noise filtering by the input delay.	0 = input 1 = HW enable	Input
Input delay	DIn.0: A digital input has to remain stable over the course of the delay (suppression of signal noise).	0 = Off (4 µs) 1 = 0.05 ms 2 = 0.1 ms 3 = 0.4 ms 4 = 0.8 ms 5 = 1.6 ms 6 = 3.2 ms 7 = 12.8 ms 8 = 20 ms	0.1 ms
On-delay	The duration from the start of the output sequence to the pulse output. You can change the on-delay in the control program by using the SLOT parameter.	0 µs to 85 000 000 µs	0 µs

[1] Only if the module is configured as "2-channel (2 A)".

Control and feedback signals for the pulse output operating mode

Control interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Bytes 0 to 3	Bytes 12 to 15	OUTPUT_VALUE (DWord)	Pulse duration: The duration period during which the digital output DQn.A remains set, after the on-delay period has expired. If you disregard the lower or upper limit, the ERR_OUT_VAL is output in the feedback interface and the last valid value is used.	
			High-speed pulse outputs deactivated:	**High-speed pulse outputs activated:**
			10 µs to 85 000 000 µs	2 µs to 85 000 000 µs
Bytes 4 to 7	Bytes 16 to 19	SLOT (DWord)	The on-delay can be changed before the start of the output sequence. See MODE_SLOT.	
			0 µs to 85 000 000 µs	

Control interface: Offset to the start address		Parameter	Meaning				
Channel 0	Channel 1 [1]						
Byte 8	Byte 20	LD_SLOT	Interpretation of the SLOT value: All values not listed subsequently are invalid and cause the error ERR_LD (in the operating mode for single updates) or ERR_SLOT_VAL (in the operating mode for continuous updates).				
			Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	
			0	0	0	0	Idle; the value is not processed
			0	0	1	0	On-delay in µs
Byte 8: Bit 4	Byte 20: Bit 4	MODE_SLOT	**Bit 4**	Operating mode for using the SLOT field.			
			0	Operating mode for individual update			
			1	Operating mode for continuous update			
Byte 9: Bit 0	Byte 21: Bit 0	SW_ENABLE	**Bit 0**	Software enable: Start/activate and stop/deactivate the output sequence.			
			0	Output deactivated/terminated			
			0 → 1	Starts output sequence for rising edge if "Function DI" = "Input"			
			1	Activates the output sequence, if the start depends on HW enable with "DI function" = "HW enable"			
Byte 9: Bit 1	Byte 21: Bit 1	TM_CTRL_DQ	**Bit 1**	Set the DQn.A output source: Selection of the CPU program or the output sequence of the module.			
			0	DQn.A and DQn.B are controlled by the CPU (logic of your program) by means of the control bits SET_DQA and SET_DQB.			
			1	DQn.A is controlled by the pulse output sequence of the module. DQn.B is always 0.			
Byte 9: Bit 3	Byte 21: Bit 3	SET_DQA	**Bit 3**	Controls the value of the digital output DQn.A, if TM_CTRL_DQ is deleted.			
			0	0 for DQn.A			
			1	1 for DQn.A			
Byte 9: Bit 4	Byte 21: Bit 4	SET_DQB	**Bit 4**	Controls the value of the digital output DQn.B, if TM_CTRL_DQ and SET_DQA are deleted.			
			0	0 for DQn.B			
			1	1 for DQn.B			
Byte 10: Bit 0	Byte 22: Bit 0	RES_ERROR	**Bit 0**	Resetting pending errors (ERR_LD, ERR_DQA, ERR_DQB, and ERR_24V).			
			0	Reset of errors is not active			
			1	Reset of errors is active			

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)".

Note: All the bytes and bits not described in the preceding table are reserved and have to be 0.

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1[1]			
Byte 0: Bit 0	Byte 8: Bit 0	ERR_PWR	**Bit 0**	Indicates undervoltage in the power supply. The bit is not set if no current is available.
			0	PWR has no undervoltage
			1	PWR was detected, but has undervoltage
Byte 0: Bit 1	Byte 8: Bit 1	ERR_24V	**Bit 1**	Indicates a short-circuit or an overload at the 24 V DC output. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at 24 V DC
			1	Short-circuit at 24 V DC

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1[1]			
Byte 0: Bit 2	Byte 8: Bit 2	ERR_LD	**Bit 2**	Indicates an error when loading a value with the field SLOT (only in the operating mode for individual updating SLOT).
			0	No load error pending.
			1	Pending load error: To reset this error and to use SLOT again, you have to set RES_ERROR (control interface).
Byte 0: Bit 3	Byte 8: Bit 3	ERR_PULSE	**Bit 3**	Indicates a pulse output error.
			0	No pulse output error.
			1	Pulse output error
Byte 0: Bit 4	Byte 8: Bit 4	ERR_DQA	**Bit 4**	Indicates a short-circuit at the DQn.B output. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.A
			1	Short-circuit at DQn.A
Byte 0: Bit 5	Byte 8: Bit 5	ERR_DQB	**Bit 5**	Indicates a short-circuit at the DQn.B output or the attempted manual setting of both DQs by means of SET_DQA, SET_DQB und TM_CTRL_DQ. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.B
			1	Short-circuit at DQn.B or the attempted setting of DQn.A and DQn.B
Byte 0: Bit 6	Byte 8: Bit 6	ERR_OUT_VAL	**Bit 6**	Indicates that an invalid value was detected in OUTPUT_VALUE.
			0	OUTPUT_VALUE is valid.
			1	OUTPUT_VALUE is invalid. This bit is automatically reset if a valid value is read out by the module.
Byte 0: Bit 7	Byte 8: Bit 7	ERR_SLOT_VAL	**Bit 7**	Indicates that an invalid value was detected in SLOT (only in the operating mode for continuous upgrading SLOT).
			0	SLOT value is valid.
			0 → 1	SLOT value is invalid. This bit is automatically reset if a valid value is read out by the module.
Byte 1: Bit 2	Byte 9: Bit 2	STS_LD_SLOT	**Bit 2**	Switchover acknowledgement bit for each action of the SLOT in the SLOT operating mode for individual update. Every switchover of this bit means a successful LD_SLOT action.
Byte 1: Bit 4	Byte 9: Bit 4	STS_READY	**Bit 4**	Indicates that the module is ready and configured.
			0	Module is not configured
			1	Module is configured
Byte 1: Bit 5	Byte 9: Bit 5	STS_SW_ENABLE	**Bit 5**	Indicates the status of SW_ENABLE (control interface).
			0	SW_ENABLE cleared
			1	SW_ENABLE set
Byte 2: Bit 0	Byte 10: Bit 0	STS_ENABLE	**Bit 0**	Indicates a running output sequence.
			0	Output sequence not running
			1	Output sequence running
Byte 2: Bit 1	Byte 10: Bit 1	STS_DQA	**Bit 1**	Indicates the signal level at the digital output DQn.A.
			0	0 at the digital output DQn.A
			1	1 at the digital output DQn.A
Byte 2: Bit 2	Byte 10: Bit 2	STS_DQB	**Bit 2**	Indicates the signal level at the digital output DQn.B.
			0	0 at the digital output DQn.B
			1	1 at the digital output DQn.B

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1[1]			
Byte 2: Bit 3	Byte 10: Bit 3	STS_DI	**Bit 3**	Indicates the signal level at the digital input DIn.0.
			0	0 at the digital input DIn.0
			1	1 at the digital input DIn.0
Byte 3: Bit 0 … 3	Byte 11: Bit 0 … 3	SEQ_CNT	The sequence counter Is incremented after completion of an output sequence.	
			With SW_ENABLE: 0 to 1	
			For HW enable: 0 to 15	
Word 3	Word 7	Reserved	Read as 0	

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)".

Note: All the bytes and bits not described in the preceding table are reserved and are read as 0.

Input and output signals for pulse output operating mode

Input and output signals	Meaning	Value range	Channel 0 BaseUnit Number of pins	Channel 1 BaseUnit Number of pins
Input signal				
HW enable	You can select the HW enable by using the "DI function" parameter and select the input delay by using the "Input delay" parameter. The signal at the DIn.0 is interpreted by the TM Pulse 2x24V as the start of the output sequence after noise filtering by the input delay.	0 = HW enable cleared 1 = HW enable activated 0 → 1 = Start of the output sequence after the input delay dependent on the software enable (SW_ENABLE)	3	4
Output signal				
Pulse at the digital output DQn.A	A pulse is output at the digital output DQn.A for the set ON period and period duration.	0 = No pulse 1 = Pulse	9	10

See also Pin assignment und load/encoder wiring

5.1.2.3 Pulse width modulation (PWM) mode (S7-1500)

Definition

You control the duty cycle of the pulse width via the OUTPUT_VALUE field of the control interface. The TM Pulse 2x24V generates continuous pulses based on this value. The control interface field OUTPUT_VALUE determines the duty cycle (pulse-pause ratio) for PWM. The period duration is adjustable.

After the assigned on-delay has expired, the DQn.A output pulses (output sequence) begin.

In the PWM operating mode you can activate the current control by using the PID loop function to control the output load current. In the case of activated current control the TM Pulse 2x24V takes over the control of the duty cycle and the control interface field OUTPUT_VALUE is used to assign the target current as the value for the ratio of target current to reference current.

Pulse diagram

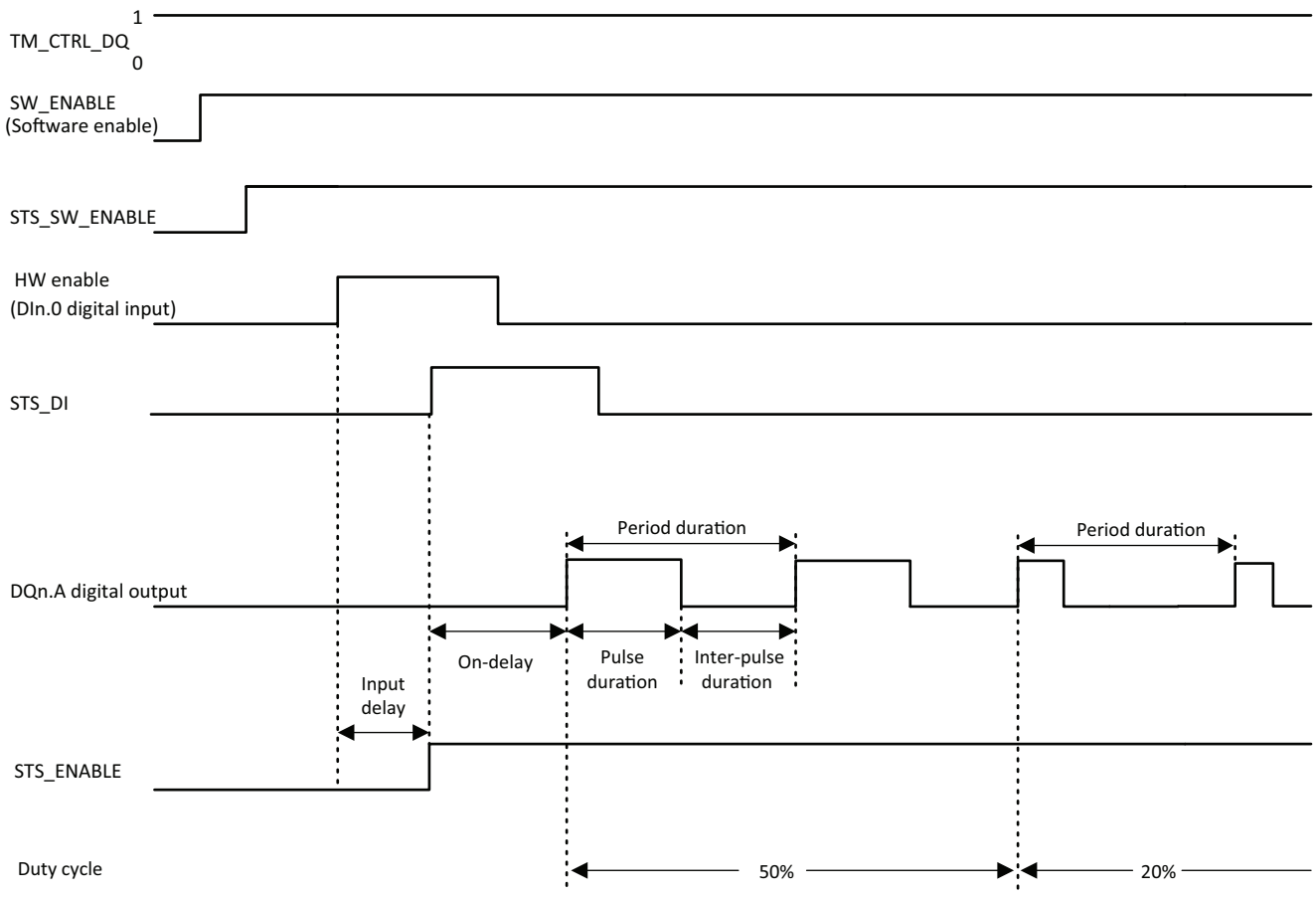

Figure 5-4 PWM output sequence

In the pulse scheme above the "DI function" parameter is set to "HW_ENABLE". For the other option the "DI function" is set to "Input". If the parameter "DI function" parameter is set to "Input", the on-delay begins with the rising edge of SW_ENABLE.

Starting the output sequence

The control program must output the enable for the output sequence with the help of the software enable (SW_ENABLE 0 → 1).

The STS_SW_ENABLE feedback bit indicates that the software enable is pending at the TM Pulse 2x24V.

You can also assign the digital input DIn.0 of the TM Pulse 2x24V as HW-enable via the parameter "Function DI". The input delay (noise filter) of the hardware enable can be configured via the "Input delay" parameter.

If you want to use the hardware enable, it must be combined with the software enable. In case of an activated software enable the output sequence starts with the first rising edge of the Hardware enable. Additional rising edges of the hardware enable are ignored during the current output sequence of the TM Pulse 2x24V. The hardware enable option is not supported in the isochronous mode.

If the enable is activated (rising edge) and remains "High" ("On") for the duration of the input delay, the on-delay is started and STS_ENABLE is set. The PWM pulse train is output when the on-delay time expires. The output sequence runs continuously, as long as SW_ENABLE is set.

Note

Output control signal TM_CTRL of the technology module

- If TM_CTRL_DQ = 1, the TM Pulse 2x24V module takes over the control and generates the pulse sequences at the DQn.A outputs.
- If TM_CTRL_DQ = 0, the CPU takes over the control and the program can directly set DQn.A/DQn.B outputs via the SET_DQA/SET_DQB

Aborting the output sequence

When the software enable is deactivated (SW_ENABLE = 1 → 0), the current output sequence is aborted and the last period duration is not completed. STS_ENABLE and the digital output DQn.A are reset to 0 immediately.

A renewed pulse output is only possible after a restart of the output sequence.

Truth table

Software enable SW_ENABLE	Function parameter DI	Hardware enable (digital input DIn.0)	Digital output DQn.A (if TM_CTRL_DQ = 1)	STS_ENABLE	Output sequence
1	HW_ENABLE	0 → 1 and remains 1 during the input delay Only active with the first rising edge, further rising edges are ignored, no start is carried out.	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0 → 1	Input	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	HW_ENABLE or input	Any status	0	0	Canceling
1	HW_ENABLE or input	Any status	0, if the on-delay has not expired or the moment lies between pulse durations 1, if the on-delay has expired and the pulse duration is still active		-
0 → 1	HW_ENABLE	0	0	0	-

Minimum pulse duration and minimum interpulse period

The minimum pulse duration and minimum interpulse period are superimposed on the proportional output characteristic.

The minimum pulse duration and the minimum interpulse period are assigned by using the "Minimum pulse duration" parameter and always have the same value.

- A pulse duration determined by TM Pulse 2x24V which is shorter than the minimum pulse duration is suppressed.
- A pulse duration determined by TM Pulse 2x24V which is longer than the period duration minus the minimum interpulse period is set to the value of the period duration (duty cycle 1 000 ‰).

Note

PWM current control and minimum interpulse period

If the current control is activated by the current control bit in the parameter record, the configured minimum pulse duration and the minimum interpulse period are ignored.

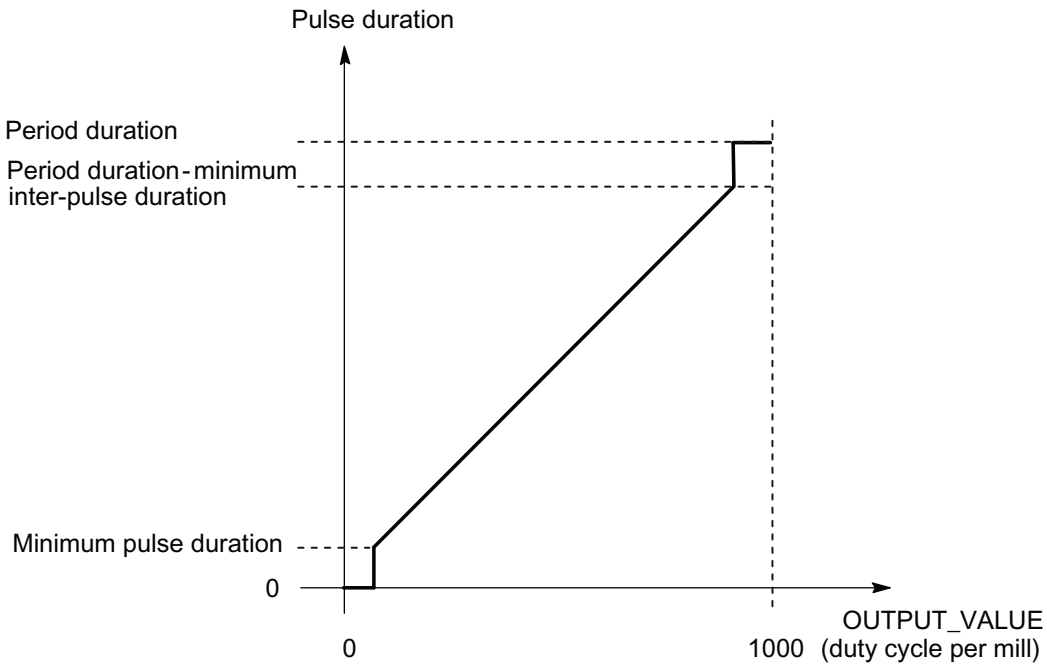

Figure 5-5 Pulse duration modulation

Note

If the dithering option without current control is activated, the minimum pulse duration and the minimum interpulse period of the module are used. In this case the overlap of the dithering is reduced so that the effective pulse duration fits into the permissible range.

Setting and changing the pulse ON period

OUTPUT_VALUE assigns the duty cycle for the current period duration. You select the range of the field OUTPUT_VALUE of the control interface with the "Output format" parameter.

- Output format "Per 100 (%)": Value range between 0 and 100
  Pulse duration = (OUTPUT_VALUE/100) x period duration.
- Output format "Per 1000 (‰)": Value range between 0 and 1000
  Pulse duration = (OUTPUT_VALUE/1000) x period duration.
- Output format "Per 10000 (‰)": Value range between 0 and 10 000
  Pulse duration = (OUTPUT_VALUE/10 000) x period duration
- Output format "S7 analog output": Value range between 0 and 27 648
  Pulse duration = (OUTPUT_VALUE/27 648) x period duration

You assign OUTPUT_VALUE directly via the control program. A new OUTPUT_VALUE is applied at the output when the next rising edge occurs.

In the case of activated current control the TM Pulse 2x24V module takes over the control of the duty cycle and the control interface field OUTPUT_VALUE is used to assign the target current as the value for the ratio of target current to reference current. You can find additional information under the Current control function.

Setting and changing the period duration

- Continuous updating
  The period duration is continuously controlled via the control interface. The MODE_SLOT bit must be set ("1" means continuous updating). LD_SLOT must be set to the value 1 ("1" means period duration).
  Set the period value in the SLOT field. The unit is always the microsecond.
  - High-speed pulse outputs **activated**: between 10 µs and 85 000 000 µs in the SLOT field.
  - High-speed pulse outputs **deactivated**: between 100 µs and 85 000 000 µs in the SLOT field.
- Individual updating
  Set the period duration in the configuration parameters. Alternatively, execute an individual updating via the control interface. MODE_SLOT must be deleted ("0" means individual updating). LD_SLOT must be set to the value 1 ("1" means period duration). Set the period duration value in the SLOT field. The unit is always the microsecond.
  - High-speed pulse outputs **activated**: Between 10 µs and 85 000 000 µs in the parameters.
  - High-speed pulse outputs **deactivated**: Between 100 µs and 85 000 000 µs in the parameters.

The new period duration is applied at the next rising edge of the output.

Additional information on the handling of SLOT parameter is available under "Handling of the SLOT parameter (control interface)".

Isochronous mode

General information is available under "Function: Isochronous mode".

In isochronous mode, the output sequence is synchronized with the T_O moment. The period duration is tailored to the application cycle (meaning the cycle as a multiple of the PROFINET cycle). The assigned period duration is possibly not configured exactly. The configured value of the period duration is adapted by TM Pulse 2x24V to the application cycle by means of the calculation algorithm. The calculation is used to minimize the difference between the assigned and calculated period duration. In the most unfavorable case, the deviation amounts to half the application cycle. The table below shows examples.

Application cycle T_{CAC} [1]	Assigned period duration $T_{Setpoint}$	T_{CAC} \ $T_{Process\ value}$	Calculated actual period duration $T_{Process}$ [1]	
10 ms (10000 µs)	5000 µs	2:1	5000 µs	
10 ms (10000 µs)	2000 µs	5:1	2000 µs	
10 ms (10000 µs)	3000 µs	3:1	3333 µs	Next possible value is used as process value
10 ms (10000 µs)	1800 µs	6:1	1666 µs	Next possible value is used as process value
10 ms (10000 µs)	6000 µs	2:1	5000 µs	Next possible value is used as process value
10 ms (10000 µs)	12000 µs	1:1	10000 µs	Next possible value is used as process value
10 ms (10000 µs)	16000 µs	1:2	20000 µs	Next possible value is used as process value
10 ms (10000 µs)	26000 µs	1:3	30000 µs	Next possible value is used as process value

[1] The determined actual period duration and the duty cycle of the application always have a ratio consisting of integers (1:1, 1:2, 1:3, ..., 2:1, 3:1, ...). This is rounded down to the next possible value.

The cycle between the digital output and the application cycle is shown below. The on/off ratio of the DQ duty cycle is displayed as 50% in the examples.

- Example 1: The period duration 10000 µs corresponds to the application cycle time 10 ms (10000 µs).

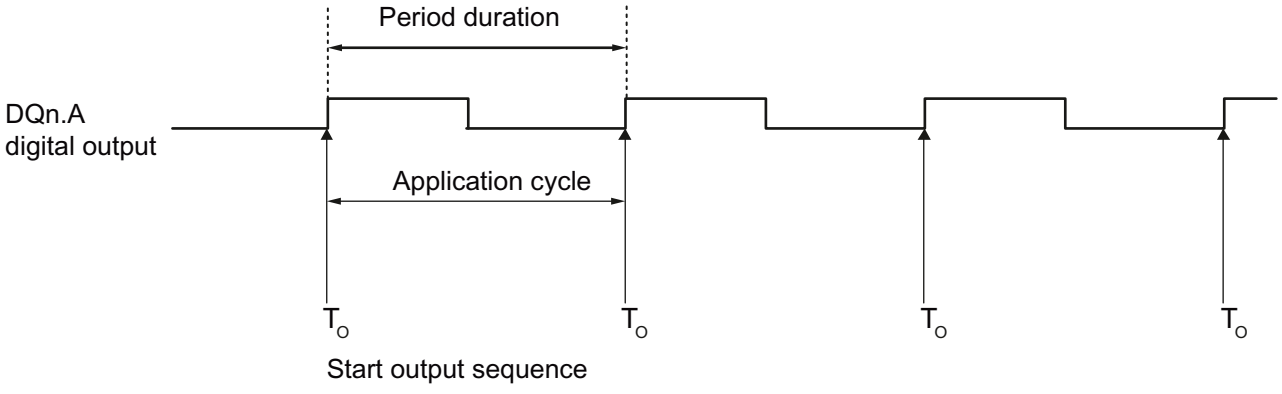

- Example 2: The 3333 µs period duration is smaller than the application cycle 10 ms (10000 µs).

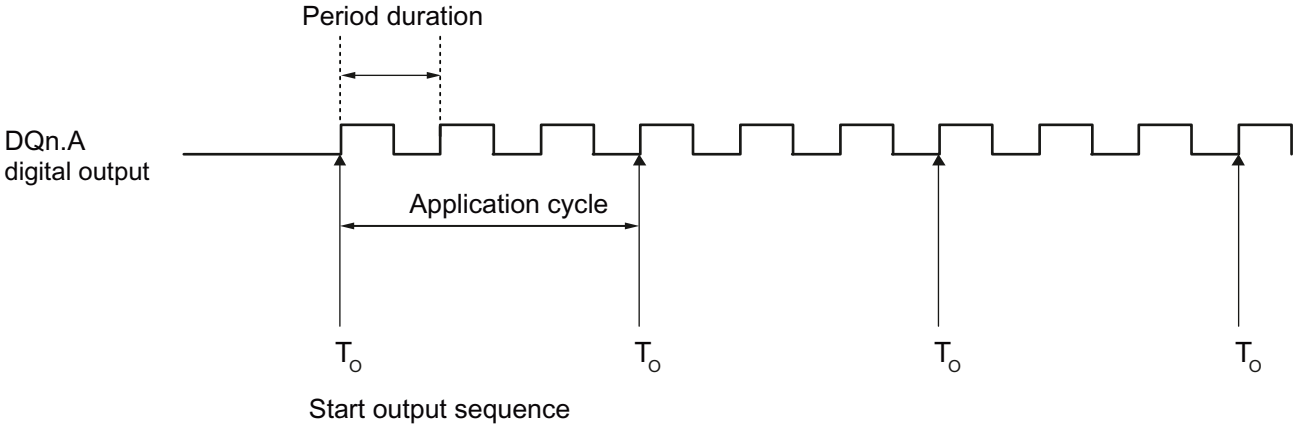

- Example 3: The 30000 µs period duration is larger than the application cycle 10 ms (10000 µs).

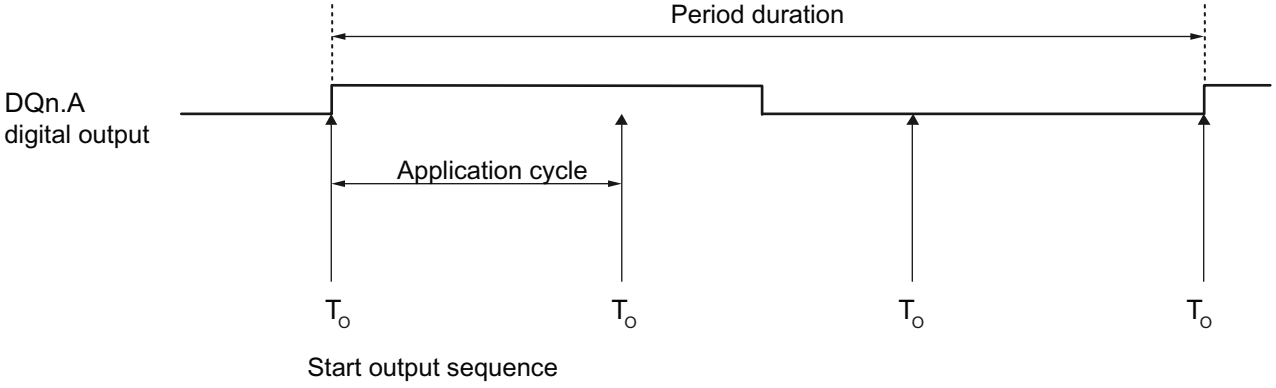

Note

Isochronous PWM mode

The on-delay is not used by the module (meaning that it is always viewed as zero) and the "DI function" parameter is always "Input". To start the output sequence only the software enable SW_ENABLE 0 → 1 is used.

Setting the minimum pulse duration and the minimum interpulse period

You assign the minimum pulse duration and the minimum pulse pause as a DWord numerical value between 0 and 85 000 000 µs using the channel parameter configuration "Minimum pulse duration".

The unit is always the microsecond. This value can only be changed via the configuration parameter record.

Setting and changing the on-delay

- Continuous updating
  The on-delay can be continuously controlled via the control interface. The MODE_SLOT bit must be set (continuous updating); LD_SLOT must have the value 2 (for on-delay). Set the on-delay to a value between 0 µs and 85 000 000 µs in the SLOT field. The unit is always the microsecond.
- Individual updating
  In the configuration parameters set the on-delay to a value between 0 µs and 85 000 000 µs. The unit is always the microsecond. Alternatively, execute an individual updating via the control interface. MODE_SLOT must be deleted (continuous updating). LD_SLOT must be set to the value 2 (for on-delay). Set the on-delay to a value between 0 µs and 85 000 000 µs in the SLOT field.

If you change the on-delay value during the output sequence, the new on-delay is activated with the next output sequence. Additional information on the handling of SLOT parameter is available under Handling of the SLOT parameter.

Current measurement

The current measurement is available in the PWM operating mode. The control program can use the current measurement for control and diagnostic purposes.

You can find additional information in the section Function: Current measurement.

Current control

The current measurement is available in the PWM operating mode. The control program can use the current measurement for control and diagnostic purposes.

You can find additional information in the section Function: Current control.

Parameters of the PWM operating mode

Parameter	Meaning	Value range	Default
Operating mode	1 = Setting the operating mode PWM	0 = Pulse output 1 = Pulse width modulation 2 = Pulse train 3 = On-/Off-delay 4 = Frequency output 5 = DC motor	1
High-speed pulse output [1]	This output supports higher frequencies.	0 = deactivated 1 = activated	Deactivated
Function DI	You can use the digital input DIn.0 as an input or as a hardware enable. The signal at the DIn. 0 is interpreted by the TM Pulse 2x24V as the start of the output sequence after noise filtering by the input delay.	0 = input 1 = HW enable	Input

Parameter	Meaning	Value range	Default
Output format	Defines the format of the relationship value (duty cycle).	0 = S7 analog format 1 = Per 100 (%) 2 = Pro 1 000 3 = Pro 10 000	Per 100 (%)
Input delay	The DIn.0 digital input has to remain stable over the course of the delay (suppression of signal noise).	0 = Off (4 µs) 1 = 0.05 ms 2 = 0.1 ms 3 = 0.4 ms 4 = 0.8 ms 5 = 1.6 ms 6 = 3.2 ms 7 = 12.8 ms 8 = 20 ms	0.1 ms
Minimum pulse duration	Minimum pulse duration and minimum inter-pulse period.	0 µs to 85 000 000 µs	0 µs
Period	Period duration of the output pulse cycle in µs. You change the period duration in the control program via the SLOT field of the control interface.	• High-speed pulse outputs deactivated 100 µs to 85 000 000 µs • High-speed pulse outputs activated 10 µs to 85 000 000 µs	2 000 000 µs
On-delay	The duration from the start of the output sequence to the pulse output. You change the on-delay in the control program via the SLOT field of the control interface.	0 µs to 85 000 000 µs	0 µs

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)".

Control and feedback signals for PWM operating mode

Control interface Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Bytes 0 to 3	Bytes 12 to 15	OUTPUT_VALUE (DWord)	The OUTPUT_VALUE determines the duty cycle (relationship pulse duration/period duration) within a period (pulse width modulation). The period duration is adjustable. The new output value is applied at the next rising edge of the output. In the case of activated current control the TM Pulse 2x24V module takes over the control of the duty cycle and the control interface field OUTPUT_VALUE is used to assign the target current as the value for the ratio of target current to reference current. You can find additional information under the Current control function. Data type UDInt: Only the 2 lowest bytes are used For Channel 0: Bytes 2 and 3 For Channel 1: Bytes 14 and 15	
			Output format	"Per 100": Value range is 0 to 1.00
				"Per 1000": Value range is 0 to 1000
				"Per 10000": Value range is 0 to 10 000
				"S7 analog output" Value range is 0 to 27 648

Control interface Offset to the start address		Parameter	Meaning				
Channel 0	Channel 1 [1]						
Bytes 4 to 7	Bytes 16 to 19	SLOT (DWord)	Your program can change the parameters listed in the Byte 8 LD_SLOT table before the start of the output sequence by using the parameters SLOT and MODE_SLOT.				
			0 µs to 85 000 000 µs				
Byte 8	Byte 20	LD_SLOT	Interpretation of the SLOT value: All values not listed subsequently are invalid and cause the error ERR_LD (in the operating mode for single updates) or ERR_SLOT_VAL (in the operating mode for continuous updates).				
			Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	**Parameter**
			0	0	0	0	Idle; the value is not processed
			0	0	0	1	Period in µs
			0	0	1	0	On-delay in µs
			0	1	0	1	Dither ramp in ms
			0	1	1	0	Dithering amplitude in per mill
			0	1	1	1	Dither period in µs
Byte 8: Bit 4	Byte 20: Bit 4	MODE_SLOT	**Bit 4**	Operating mode for using the SLOT field.			
			0	Operating mode for individual update			
			1	Operating mode for continuous update			
Byte 9: Bit 0	Byte 21: Bit 0	SW_ENABLE	**Bit 0**	Software enable: Starting and stopping of the output sequence.			
			0	Output aborted			
			0 → 1	Starts the output sequence at a rising edge if "Function DI" = "Input".			
			1	Activates the output sequence, if the start depends on HW enable with "Function DI" = "HW enable".			
Byte 9: Bit 1	Byte 21: Bit 1	TM_CTRL_DQ	**Bit 1**	Set the DQn.A output source: Selection of the CPU program or the output sequence of the module.			
			0	DQn.A and DQn.B are controlled by the CPU program (in your program) by means of the control bits SET_DQA and SET_DQB.			
			1	DQn.A is controlled by the pulse output sequence of the module. DQn.B is always 0 for TM_CTRL_DQ = 1.			
Byte 9: Bit 3	Byte 21: Bit 3	SET_DQA	**Bit 3**	Controls the value of the digital output DQn.A, if TM_CTRL_DQ = 0.			
			0	0 for DQn.A			
			1	1 for DQn.A			
Byte 9: Bit 4	Byte 21: Bit 4	SET_DQB	**Bit 4**	Controls the value for the digital output DQn.B, if TM_CTRL_DQ = 0 and if SET_DQA is deleted.			
			0	0 for DQn.B			
			1	1 for DQn.B			
Byte 10: Bit 0	Byte 22: Bit 0	RES_ERROR	**Bit 0**	Resetting pending errors (ERR_LD, ERR_DQA, ERR_DQB, and ERR_24V).			
			0	Reset of errors is not active			
			1	Reset of errors is active			

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)"

Note: All the bytes and bits not described in the preceding table are reserved and have to be 0.

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 0: Bit 0	Byte 8: Bit 0	ERR_PWR	**Bit 0**	Indicates undervoltage in the power supply. The bit is not set if no current is available.
			0	PWR has no undervoltage
			1	PWR was detected, but has undervoltage
Byte 0: Bit 1	Byte 8: Bit 1	ERR_24V	**Bit 1**	Indicates a short-circuit or an overload at the 24 V DC output. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at 24 V DC
			1	Short-circuit at 24 V DC
Byte 0: Bit 2	Byte 8: Bit 2	ERR_LD	**Bit 2**	Indicates an error when loading a value with the field SLOT (only in the operating mode for individual updating SLOT).
			0	No load error pending
			1	Pending load error: To reset this error and to use SLOT again, you have to set RES_ERROR (control interface).
Byte 0: Bit 4	Byte 8: Bit 4	ERR_DQA	**Bit 4**	Indicates a short-circuit at the DQn.A output. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.A
			1	Short-circuit at DQn.A
Byte 0: Bit 5	Byte 8: Bit 5	ERR_DQB	**Bit 5**	Indicates a short-circuit at the DQn.B output or the attempted manual setting of DQn.A and DQn.B by means of SET_DQA, SET_DB and TM_CTRL_DQ. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.B
			1	Short-circuit at DQn.B, or the attempted setting of DQn.A and DQn.B
Byte 0: Bit 7	Byte 8: Bit 7	ERR_SLOT_VAL	**Bit 7**	Indicates that an invalid value was detected in SLOT (only in the operating mode for continuous upgradingSLOT).
			0	SLOT value is valid
			0 → 1	SLOT value is invalid
Byte 1: Bit 2	Byte 9: Bit 2	STS_LD_SLOT	**Bit 2**	Switchover acknowledgement bit for each action of the SLOT in the SLOT operating mode for individual update. Every switchover of this bit means a successful LD_SLOT action.
Byte 1: Bit 4	Byte 9: Bit 4	STS_READY	**Bit 4**	Indicates that the module is ready and configured.
			0	Module is not configured
			1	Module is configured
Byte 1: Bit 5	Byte 9: Bit 5	STS_SW_ENABLE	**Bit 5**	Indicates the status of SW_ENABLE (control interface).
			0	SW_ENABLE cleared
			1	SW_ENABLE set
Byte 2: Bit 0	Byte 10: Bit 0	STS_ENABLE	**Bit 0**	Indicates a running output sequence.
			0	Output sequence not running
			1	Output sequence running
Byte 2: Bit 1	Byte 10: Bit 1	STS_DQA	**Bit 1**	Indicates the signal level at the digital output DQn.A.
			0	0 at the digital output DQn.A
			1	1 at the digital output DQn.A
Byte 2: Bit 2	Byte 10: Bit 2	STS_DQB	**Bit 2**	Indicates the signal level at the digital output DQn.B.
			0	0 at the digital output DQn.B
			1	1 at the digital output DQn.B

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 2: Bit 3	Byte 10: Bit 3	STS_DI	**Bit 3**	Indicates the signal level at the digital input DIn.0.
			0	0 at the digital input DIn.0
			1	1 at the digital input DIn.0
Byte 3: Bits 0 to 3	Byte 11: Bits 0 to 3	SEQ_CNT	Sequence counter = 0. The sequence counter is not used in the pulse width modulation operating mode.	
Word 2	Word 6	MEASURED_CURRENT	Positive value S7 analog format (0 to 32 767): • 2 ampere channel output: Measuring range final value 27 648; this means 2 000 mA • 4-ampere channel output: Measuring range final value 27 648; this means 4 000 mA • 32 767 means that no applicable current measurement is available; for example during the first PWM period.	
Word 3	Word 7	Reserved	Read as 0	

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)"

Note: All the bytes and bits not described in the preceding table are reserved and are read as 0.

Input and output signals for the PWM operating mode

Input and output signal	Meaning	Value range	Channel 0 BaseUnit Number of pins	Channel 1 BaseUnit Number of pins
Input signal				
HW enable Note: HW enable is not supported for PWM in isochronous mode	You can select the HW enable by using the "DI function" parameter and select the input delay by using the "Input delay" parameter. The signal at the DIn.0 digital input is interpreted by the TM Pulse 2x24V as the start of the output sequence after noise filtering by the input delay.	0 = HW enable cleared 1 = HW enable activated 0 → 1 = Start of the output sequence after the input delay dependent on the software enable (SW_ENABLE)	3	4
Output signal				
Pulse at the digital output DQn.A	A pulse is output at the DQn.A digital output for the set duty cycle and period duration.	0 = No pulse 1 = Pulse	9	10

See also Pin assignment und load/encoder wiring

5.1.2.4 Pulse train mode (S7-1500)

Definition

After the assigned on-delay has expired, the TM Pulse 2x24V outputs the number of pulses which you have assigned as a pulse train (output sequence). Period duration and pulse duration can be adjusted.

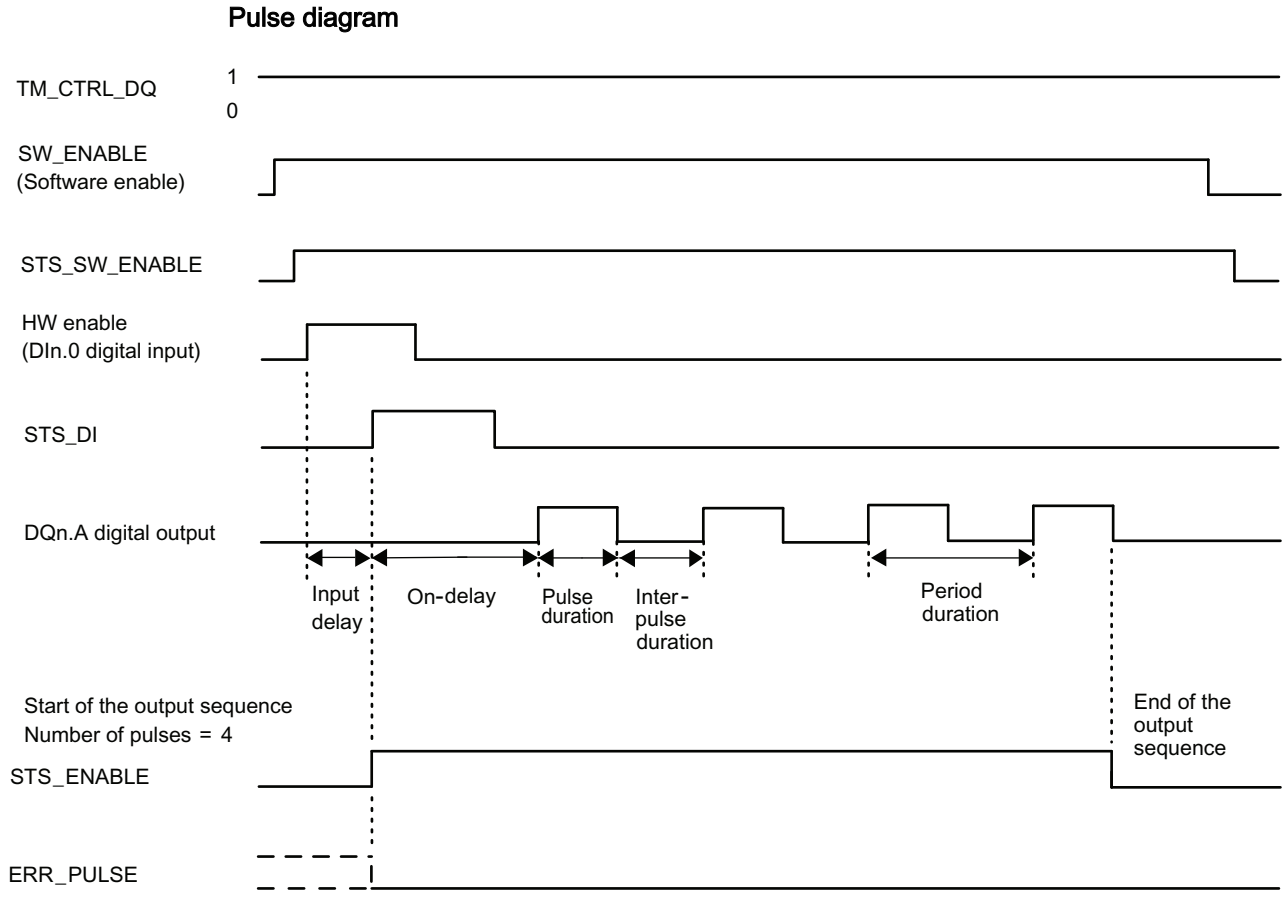

Figure 5-6 Output sequence of the pulse train

In the pulse diagram above the "DI function" parameter is set to "HW enable". For the other option the "DI function" is set to "Input". If the parameter "DI function" is set to "Input", the on-delay begins with the rising edge of SW_ENABLE.

Starting the output sequence

The control program must output the enable for the output sequence by means of the software enable (SW_ENABLE 0 → 1.)

The STS_SW_ENABLE feedback bit indicates that the software enable is pending at the TM Pulse 2x24V.

In addition you also set the digital input DIn.0 of the TM Pulse 2x24V as HW enable via the "DI function" parameter. The input delay of the hardware enable can be configured via the "Input delay" parameter.

If you want to use the hardware enable, it must be combined with the software enable. In case of activated software enable with SW_ENABLE the output sequence starts with the first rising edge of the Hardware enable. Additional rising edges of the hardware enable are ignored during the current output sequence of the TM Pulse 2x24V. As soon as the software release has been activated, a rising edge of the hardware enable (detected after the input delay) starts the next output sequence

If the enable is activated (rising edge) and the input delay remains (noise filter) "High" ("On"), the on-delay is started and STS_ENABLE is set. When the on-delay expires, the pulse train is output with the assigned pulse duration.. The output sequence is terminated with the pulse end and STS_ENABLE is deleted.

If you change the number of pulses during the output sequence to a number which was already reached in the sequence, the ERR_PULSE bit indicates an output error.

The ERR_PULSE feedback bit is deleted by TM Pulse 2x24V the next time the output sequence starts.

Note

Control signal of the technology module output TM_CTRL_DQ

- If TM_CTRL_DQ = 1, the TM Pulse 2x24V module takes over the control and generates pulse sequences at the output DQn.A.
- If TM_CTRL_DQ = 0, the CPU takes over the control and the program can set the DQn.A/DQn.B outputs directly via the SET_DQA/SET_DQB control bits.

Aborting the output sequence

When the software enable is deactivated ((SW_ENABLE = 1 → 0)) during the on-delay or the pulse train, the current output sequence is aborted and the last period duration is not completed. STS_ENABLE and the digital output are reset to 0 immediately.

A renewed pulse output is only possible after a restart of the output sequence.

Truth table

Software enable SW_ENABLE	DI function parameter	Hardware enable (Digital input DIn.0)	Digital output DQn.1 (if TM_CTRL_DQ = 1)	STS_ENABLE	Output sequence
1	HW_ENABLE	0 → 1 and remains 1 during the input delay Only active with the first rising edge, further rising edges are ignored, no start is carried out.	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0 → 1	Input	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	HW_ENABLE or input	Any status	0	0	Canceling
1	HW_ENABLE or input	Any status	0, if the on-delay has not expired or the moment lies between pulse durations or the output sequence is finished 1, if the on-delay has expired and the pulse duration is still active		-
0 → 1	HW_ENABLE	0	0	0	-

Setting and changing the number of pulses

The control program can set the number of pulses directly by means of the OUTPUT_VALUE parameter of the control interface.

Set the pulse number directly as a DWord number value between 1 and 4.294.967.295 (2^{32}-1).

If you change the number of pulses, the new value takes effect immediately, irrespective of whether the on-delay may have expired.

- If the new pulse number of the current output sequence has not been reached yet, the new pulse number of the current output sequence is used.
- If you have reduced the number of pulses to a number which is smaller than the pulse number already output, the output sequence is terminated, STS_ENABLE and the DQn.A digital output are cleared and ERR_PULSE is set. With the next start of an output sequence, ERR_PULSE is cleared.
- If number of pulses is zero, the output sequence is aborted, STS_ENABLE and the DQn.A digital output are cleared and ERR_PULSE is set. With the next start of an output sequence ERR_PULSE is cleared.

Setting and changing the period duration

- Continuous updating
  The period duration can be continuously controlled via the control interface. The MODE_SLOT bit must be set ("1" means continuous updating). LD_SLOT must be set to the value 1 (for period duration). Set the period duration as a DWord number value in microseconds in the SLOT parameter field.
  - High-speed pulse outputs **activated** (see parameter): Between 10 µs and 85 000 000 µs.
  - High-speed pulse outputs **deactivated** (see parameter): Between 100 µs and 85 000 000 µs.
- Individual updating
  Set the period duration in the configuration parameters. Alternatively, execute an individual updating via the control interface. MODE_SLOT has to be 0 (individual update). LD_SLOT must have the value 1 (for period duration) Set the period duration as a DWord number value in microseconds in the SLOT parameter field.
  - High-speed pulse outputs **activated** (see parameter): Between 10 µs and 85 000 000 µs.
  - High-speed pulse outputs **deactivated** (see parameter): Between 100 µs and 85 000 000 µs.

Setting and changing the on-delay

- Continuous updating
  The on-delay can be continuously controlled via the control interface. The MODE_SLOT bit must be set ("1" means continuous updating). LD_SLOT must be set to value 2 (for on-delay). Set the on-delay to a value between 0 µs and 85 000 000 µs in the SLOT parameter field.
- Individual updating
  In the configuration parameters set the on-delay to a value between 0 µs and 85 000 000 µs. Alternatively, execute an individual updating via the control interface. MODE_SLOT must be deleted ("0" means individual updating). LD_SLOT must be set to value 2 (for on-delay). Set the on-delay to a value between 0 µs and 85 000 000 µs in the SLOT field.
- If you change the on-delay value during the output sequence, the new on-delay is activated with the next output sequence.

Additional information on the handling of the SLOT parameter is available under Handling of the SLOT parameter.

Setting and changing the duty cycle

The range for the duty cycle parameter is selected via the "Output format" parameter. The TM Pulse 2x24V uses this assigned duty cycle value for calculating the pulse duration. If the numerical value assigned by you exceeds the high limit, a duty cycle of 100% of the period duration is used. This action does not cause an error.

- Output format "Per 100 (%)": Value range 0 to 100
  - Pulse duration = (duty cycle/100) x period duration
- Output format "Per 1000": Value range 0 to 1 000
  - Pulse duration = (duty cycle/1 000) x period duration.
- Output format "Per 10000": Value range 0 to 10 000
  - Pulse duration = (duty cycle/10 000) x period duration.
- Output format "S7 analog output": Value range 0 to 27 648
  - Pulse duration = (duty cycle/27 648) x period duration.

Updating the duty cycle

- Continuous updating
  The duty cycle can be continuously controlled via the control interface. The MODE_SLOT bit must be set ("1" means continuous updating). LD_SLOT must be set to value 4 (for the duty cycle). Determine the value for the duty cycle in the control interface field SLOT. The unit depends on the assigned output format.
- Individual updating
  Set the duty cycle in the configuration parameters. Alternatively, execute an individual updating via the control interface. MODE_SLOT must be deleted ("0" means individual updating). LD_SLOT must be set to the value 4 (for the duty cycle). Determine the value for the duty cycle in the control interface field SLOT.
- The unit depends on the assigned output format. If you change the duty cycle duration during the output sequence, the new duty cycle is activated with the next output sequence.

Additional information on the handling of SLOT parameter is available under Handling of the SLOT parameter.

Isochronous mode

General information is available under Function: Isochronous mode.

The isochronous mode does not have any influence on the function principle of the pulse train operating mode.

If you want to synchronize the output sequence with T_o , set the "Function DI" parameter to "Input". In this case the output sequence of the pulse train begins at T_o.

Current measurement

The current measurement is available in the pulse train operating mode. The control program can use the current measurement for control and diagnostic purposes.

You can find additional information in the section Function: Current measurement.

Parameters for the pulse train operating mode

Parameter	Meaning	Value range	Default
Operating mode	2 = Setting the "Pulse train" operating mode	0 = Pulse output 1 = Pulse width modulation 2 = Pulse train 3 = On-/Off-delay 4 = Frequency output 5 = Direct current	1
High-speed pulse output [1]	The output supports higher frequencies (see pulse duration range).	0 = Deactivated 1 = Activated	Deactivated
DI function	You can use the DIn.0 digital input as an input or as a hardware enable. The signal at the DIn.0 is interpreted by the TM Pulse 2x24V as the start of the output sequence after noise filtering by the input delay.	• Input • HW enable	Input
Output format	Defines the format for the ratio value of the duty cycle.	0 = S7 analog format 1 = Per 100 (%) 2 = Pro 1 000 3 = Pro 10 000	Per 100 (%)
Input delay	The DIn.0 digital input has to remain stable over the course of the delay (suppression of signal noise).	0 = Off (4 µs) 1 = 0.05 ms 2 = 0.1 ms 3 = 0.4 ms 4 = 0.8 ms 5 = 1.6 ms 6 = 3.2 ms 7 = 12.8 ms 8 = 20 ms	0.1 ms

Parameter	Meaning	Value range	Default
Period	Period duration of the output sequence: You can change the period duration in the control program via the SLOT field of the control interface.	• High-speed pulse output deactivated 100 µs to 85 000 000 µs • High-speed pulse output activated 10 µs to 85 000 000 µs	2 000 000 µs
On-delay	The time that passes from the start of the output sequence to the output of the pulse train. You can change the on-delay in the control program by using the SLOT parameter.	0 µs to 85 000 000 µs	0 µs
Duty cycle	The pulse duration results from duty cycle and period duration. The format of the duty cycle is determined by the "output format" parameter. You change the duty cycle in the control program via the SLOT field of the control interface.	Output format: Minimum value to maximum value • Per 100: 0 to 100 • Per 1000: 0 to 1 000 • Per 10000: 0 to 10 000 • S7 analog output: 0 to 27 648	50%

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)".

Control and feedback signals for the pulse train operating mode

Control interface Offset to the start address		Parameter	Meaning				
Channel 0	Channel 1 [1]						
Bytes 0 to 3	Bytes 12 to 15	OUTPUT_VALUE (DWord)	Assign the number of pulses to the OUTPUT_VALUE as a DWord numerical value between 0 and 4 294 967 295 (2^{32}-1).				
Bytes 4 to 7	Bytes 16 to 19	SLOT (DWord)	The on-delay, period duration and duty cycle can be changed before the start of the output sequence. See Handling of the SLOT parameter.				
			0 µs to 85 000 000 µs				
Byte 8	Byte 20	LD_SLOT	Interpretation of the SLOT value: All values not listed subsequently are invalid and cause the error ERR_LD (in the operating mode for single updates) or ERR_SLOT_VAL (in the operating mode for continuous updates).				
			Bit 3	Bit 2	Bit 1	Bit 0	
			0	0	0	0	Idle; the value is not processed
			0	0	0	1	Period duration in microseconds
			0	0	1	0	On-delay in microseconds
			0	1	0	0	Duty cycle in the format defined by the "Output format" parameter
Byte 8: Bit 4	Byte 20: Bit 4	MODE_SLOT	Bit 4	Operating mode for using the SLOT field.			
			0	Operating mode for individual update			
			1	Operating mode for continuous update			
Byte 9: Bit 0	Byte 21: Bit 0	SW_ENABLE	Bit 0	Software enable: Start/activate and stop/deactivate the output sequence, if "DI function" = "Input".			
			0	Output deactivated/terminated			
			0 → 1	Starts the output sequence for a rising edge if "DI function" = "Input".			
			1	Activates the output sequence, if the start depends on the HW enable with "DI function" = "HW enable".			

Control interface Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 9: Bit 1	Byte 21: Bit 1	TM_CTRL_DQ	**Bit 1**	Setting the DQn.A output source: Selection of the CPU program or the output sequence of the module.
			0	DQn.A and DQn.B are controlled by the CPU (in the program) by means of the control bits SET_DQA and SET_DQB
			1	DQn.A is controlled by the pulse output sequence of the module. DQn.B is always 0
Byte 9: Bit 3	Byte 21: Bit 3	SET_DQA	**Bit 3**	Controls the value of the digital output DQn.A, if TM_CTRL_DQ is deleted.
			0	0 at DQn.A
			1	1 at DQn.A
Byte 9: Bit 4	Byte 21: Bit 4	SET_DQB	**Bit 4**	Controls the value of the digital output DQn.B, if TM_CTRL_DQ and SET_DQA are deleted.
			0	0 at DQn.B
			1	1 at DQn.B
Byte 10: Bit 0	Byte 22: Bit 0	RES_ERROR	**Bit 0**	Reset pending error (ERR_LD, ERR_DQA, ERR_DQB and ERR_24V).
			0	Reset of errors is not active
			1	Reset of errors is active

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)"

Note: All the bytes and bits not described in the preceding table are reserved and have to be 0.

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 0: Bit 0	Byte 8: Bit 0	ERR_PWR	**Bit 0**	Indicates undervoltage in the power supply. The bit is not set if no current is available.
			0	PWR has no undervoltage
			1	PWR was detected, but has undervoltage
Byte 0: Bit 1	Byte 8: Bit 1	ERR_24V	**Bit 1**	Indicates a short-circuit or an overload at the 24 V DC output. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at 24 V DC
			1	Short-circuit at 24 V DC
Byte 0: Bit 2	Byte 8: Bit 2	ERR_LD	**Bit 2**	Indicates an error when loading a value with the SLOT field (only in the operating mode for individual updating SLOT).
			0	No load error pending
			1	Pending load error: To reset this error and be able to use SLOT again, you have to set the RES_ERROR (control interface).
Byte 0: Bit 3	Byte 8: Bit 3	ERR_PULSE	**Bit 3**	Indicates a pulse output error.
			0	No pulse output error.
			1	Pulse output error
Byte 0: Bit 4	Byte 8: Bit 4	ERR_DQA	**Bit 4**	Indicates a short-circuit at output DQn.A. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.A
			1	Short-circuit at DQn.A

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 0: Bit 5	Byte 8: Bit 5	ERR_DQB	**Bit 5**	Indicates a short-circuit at output DQn.B or the attempted manual setting of both DQs by means of SET_DQA, SET_DQB und TM_CTRL_DQ. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.B
			1	Short-circuit at DQn.B or the attempted setting of DQn.A and DQn.B
Byte 0: Bit 7	Byte 8: Bit 7	ERR_SLOT_VAL	**Bit 7**	Indicates that an invalid value was detected in SLOT (only in SLOT operation mode for continuous upgrading).
			0	SLOT value is valid
			1	SLOT value is invalid
Byte 1: Bit 2	Byte 9: Bit 2	STS_LD_SLOT	**Bit 2**	Switchover acknowledgement bit for each action of the SLOT in the SLOT operating mode for individual update. Every switchover of this bit means a successful LD_SLOT action.
Byte 1: Bit 4	Byte 9: Bit 4	STS_READY	**Bit 4**	Indicates that the module is ready and configured.
			0	Module is not configured
			1	Module is configured
Byte 1: Bit 5	Byte 9: Bit 5	STS_SW_ENABLE	**Bit 5**	Indicates the status of SW_ENABLE (control interface).
			0	SW_ENABLE deleted
			1	SW_ENABLE set
Byte 2: Bit 0	Byte 10: Bit 0	STS_ENABLE	**Bit 0**	Indicates an output sequence.
			0	Output sequence not running
			1	Output sequence running
Byte 2: Bit 1	Byte 10: Bit 1	STS_DQA	**Bit 1**	Indicates the signal level at digital output DQn.A.
			0	0 at the digital output DQn.A
			1	1 at the digital output DQn.A
Byte 2: Bit 2	Byte 10: Bit 2	STS_DQB	**Bit 2**	Indicates the signal level at digital output DQn.B.
			0	0 at the digital output DQn.B
			1	1 at the digital output DQn.B
Byte 2: Bit 3	Byte 10: Bit 3	STS_DI	**Bit 3**	Indicates the signal level at the digital input DIn.0.
			0	Signal 0 at the digital input DIn.0
			1	Signal 1 at the digital input DIn.0
Byte 3: Bits 0 to 3	Byte 11: Bits 0 to 3	SEQ_CNT	Sequence counter: Is incremented after completion of an output sequence.	
			For SW_ENABLE: 0 to 1	
			For HW_ENABLE: 0 to 15	
Word 2	Word 6	MEASURED_CUR-RENT	Positive value S7 analog format (0 to 32 767): • 2 ampere output: Measuring range final value 27 648; this means 2 000 mA • 4-ampere output: Measuring range final value 27 648; this means 4 000 mA • 32 767 means that no applicable current measurement is available; e. g. during the first period	
Word 3	Word 7	Reserved	Read as 0	

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)"

Note: All bytes and bits not described in the preceding table are reserved and are read as 0.

Input and output signals for pulse train operating mode

Input and output signal	Meaning	Value range	Channel 0 BaseUnit Pin number	Channel 1 BaseUnit Pin number
Input signal				
HW enable	You can select the HW enable with the "DI function" parameter and select the input delay with the "Input delay" parameter. The signal at the DIn.0 digital input is interpreted by the TM Pulse 2x24V as the start of the output sequence after noise filtering by the input delay.	0 = HW enable cleared 1 = HW enable issued 0 → 1 = Start of the output sequence; dependent on the software enable (SW_ENABLE)	3	4
Output signal				
Pulse train at the DQn.A digital output	Pulses are output at the DQn.A digital output for the set pulse duration.	0 = No pulse 1 = Pulse	9	10

See also Pin assignment und load/encoder wiring

5.1.2.5 On/Off-delay mode (S7-1500)

Definition

The pending signal at the digital input DIn.0 of the TM Pulse 2x24V is output with an assigned on-/off-delay at digital output DQn.A.

Pulse diagram

SW_ENABLE is set, while digital input DIn.0 = 0:

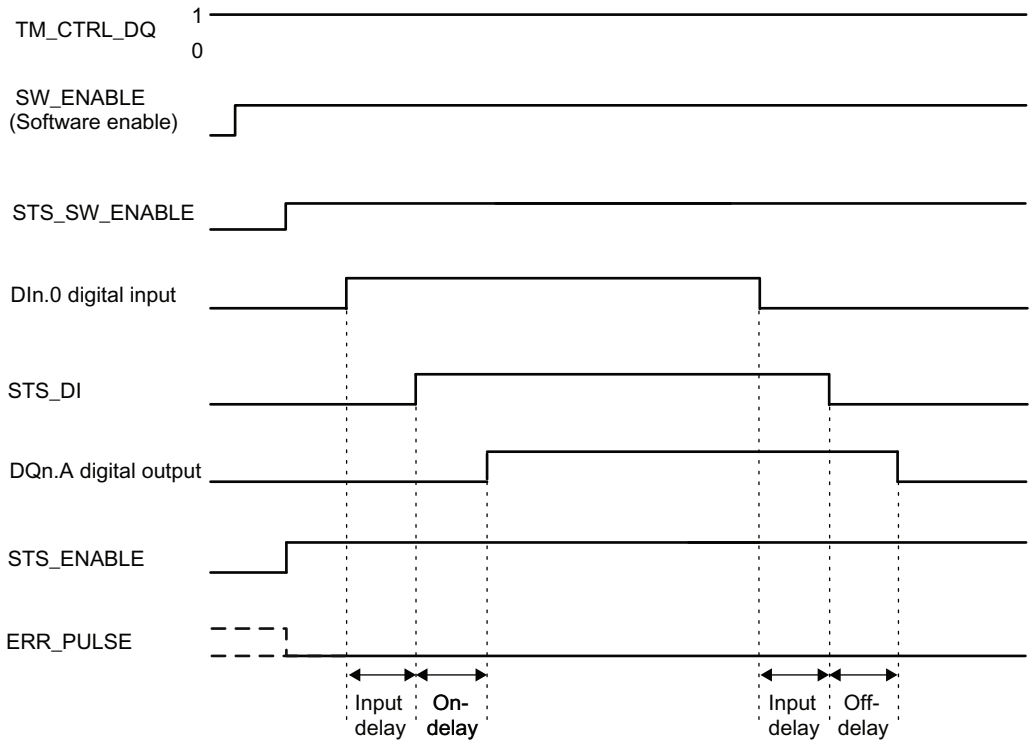

Figure 5-7 On-/Off-delay output sequence (at start digital input DIn.0 = 0)

SW_ENABLE is set, while digital input DIn.0 = 1:

If SW_ENABLE is set while digital input DIn.0 = 1, the first edge at DI (falling edge) is ignored.

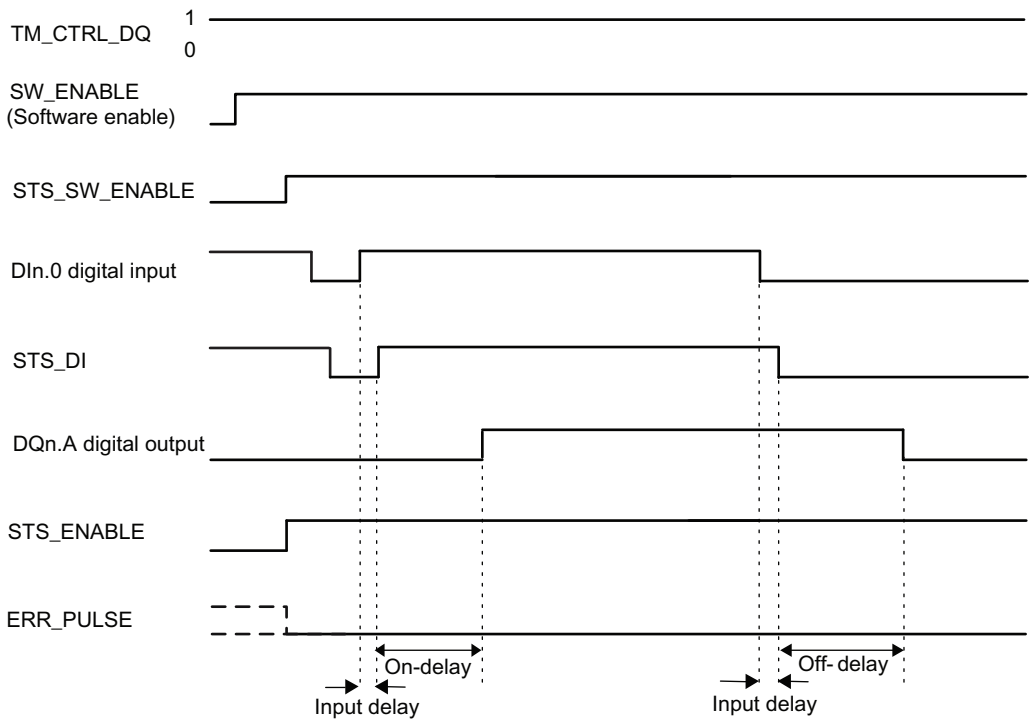

Figure 5-8 On-/Off-delay output sequence (at start digital input DIn.0 = 1)

Starting the output sequence

The control program must output the enable for the output sequence by means of the software enable (SW_ENABLE 0 → 1.)

The STS_SW_ENABLE feedback bit indicates that the software enable is pending at the TM Pulse 2x24V.

- If the DIn.0 digital input goes to the "High" status (rising edge) and remains "High" during the duration of the on-delay, the off-delay is started. After the on-delay has expired digital output DQn.A is cleared.
- If the DIn.0 digital input goes into the "Low" status ("Off") for the duration of the input delay, the off-delay is started. After the on-delay has expired digital output DQn.A is cleared.
- If the TM Pulse 2x24V detects a pulse duration at the DIn.0 digital input or an interpulse period which is shorter than the on-delay time, the input is ignored and the DQn.A digital output remains unchanged.
- If the TM Pulse 2x24V detects a pulse duration at the DIn.0 digital input or an interpulse period which is longer than the input delay time but too short to generate the configured pulse duration or interpulse period, the ERR_PULSE is set and digital output DQn.A remains unchanged.
- With the next edge of the DIn.0 digital input the TM Pulse 2x24V clears the ERR_PULSE feedback bit.

Note

Output control signal TM_CTRL_DQ of the technology module

- If TM_CTRL_DQ = 1, the TM Pulse 2x24V module takes over the control and generates pulse sequences at the output DQn.A.
- If TM_CTRL_DQ = 0, the CPU takes over the control and the program can set the DQn.A/DQn.B outputs via the SET_DQA/SET_DQB control bits.

Aborting the output sequence

When the software enable is deactivated ((SW_ENABLE = 1 → 0)) during the on-delay or the pulse train, the current output sequence is aborted and the last period duration is not completed. STS_ENABLE and the digital output are reset to 0 immediately.

A renewed pulse output is only possible after a restart of the output sequence.

Truth table

Software enable SW_ENABLE	Digital input DIn.0	Digital output DQn.A (if TM_CTRL_DQ = 1)	STS_ENABLE	Output sequence
1	0 → 1 and remains 1 during the input delay	0, if on-delay > 0 1, if on-delay = 0	1	Starting
1	1 → 0	1, if the on-delay > 0 0, if the on-delay = 0	1	Starting
0	Any status	0	0	Canceling
1	Any status	0, if the on-delay has not expired or the off-delay has expired 1, if the on-delay has expired and the off-delay has not expired	1	-
0 → 1	0	0	1	-

Minimum pulse duration/minimum interpulse period of the digital output DQn.A

The minimum pulse duration/minimum interpulse period of the digital output DQn.A amounts to 1.5 µs (for active high pulse output) and 10 µs (for inactive high pulse output). Please note that low values are possible but cannot be guaranteed because of hardware dependency.

Please consider this note when setting the input delay / output delay and the pulse duration / interpulse period for the digital input DIn.0. Otherwise the reply at digital output DQn.A cannot be guaranteed.

The pulse duration of the DIn.0 digital input is too short.

Case 1: The DIn.0 pulse duration < on-delay: The DIn.0 pulse is filtered and ignored (no error occurs).

Case 2: The TM Pulse 2x24V detects a DIn.0 pulse duration that is too short, if: DIn.0 Pulse duration + Off-delay ≤ On-delay (error is set)

Case 3: The TM Pulse 2x24V detects a DIn.0 pulse duration that is too short, if: DIn.0 pulse duration + Off-delay + Minimum DQ pulse duration < On-delay (error is set)

Reply of the TM Pulse 2x24V to a pulse duration that is too short:

- ERR_PULSE is set.
- The current on-delay is deleted.
- The off-delay is not started.
- The signal level at Digital output DQn.A remains 0.

ERR_PULSE is cleared with the next rising edge at the digital input DIn.0.

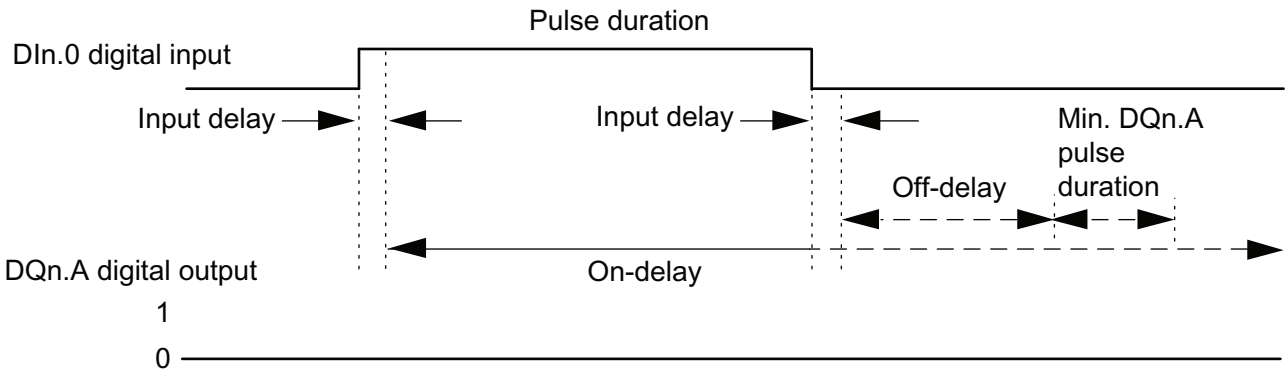

Figure 5-9 The DIn.0 pulse duration is too short

The interpulse period of the digital input DIn.0 is too short.

The TM Pulse 2x24V detects a pulse duration that is too short at the rising edge on digital output DIn.0, if:

Interpulse period + On-delay ≤ Off-delay.

- Case 1: Interpulse period < On-delay: The interpulse period at the DIn.0 input is filtered and ignored (no error occurs).
- Case 2: Interpulse period + on-delay ≤ Off-delay (error is set)
- Case 3: Interpulse period + On-delay < Off-delay + Minimum interpulse period (error is set).

Reply of the TM Pulse 2x24V to an interpulse period that is too short:

- ERR_PULSE is set.
- The current off-delay is deleted.
- The on-delay is not started.
- The signal level at Digital output DQn.A remains 1.

ERR_PULSE is cleared with the next falling edge at the digital input DIn.0.

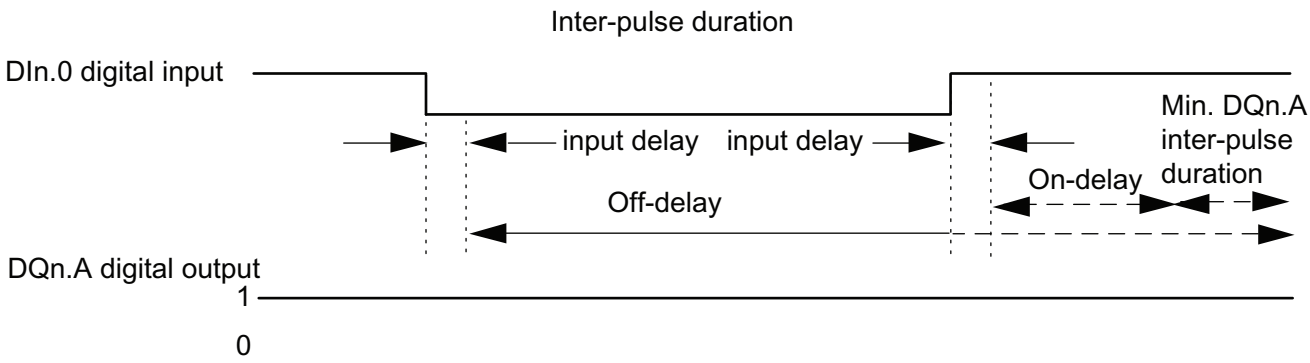

Figure 5-10 The DIn.0 interpulse period is too short

The current on-delay is triggered again

The TM Pulse 2x24V starts a new on-delay at the rising edge on the digital output DIn.0, if:

On-delay > pulse duration + interpulse period

This deletes the current off-delay.

The DQn.A digital output is only set if the signal level 1 at the digital input DIn.0 is pending longer than the on-delay. This enables you to filter rapid pulses.

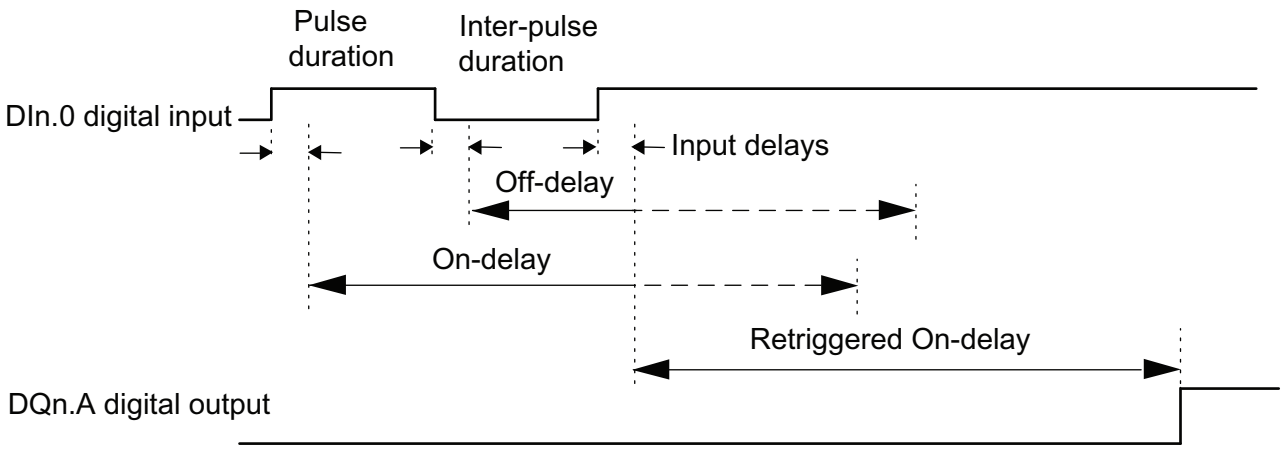

Figure 5-11 DIn.0 again triggers the current on-delay.

Retriggering the current off-delay

The TM Pulse 2x24V starts a new off-delay at the falling edge on the digital output DIn.0, if:

Off-delay > Pulse duration + Interpulse period.

The DQn.A digital output is only cleared if the signal level 0 at the digital input Dn.0I is pending longer than the off delay.

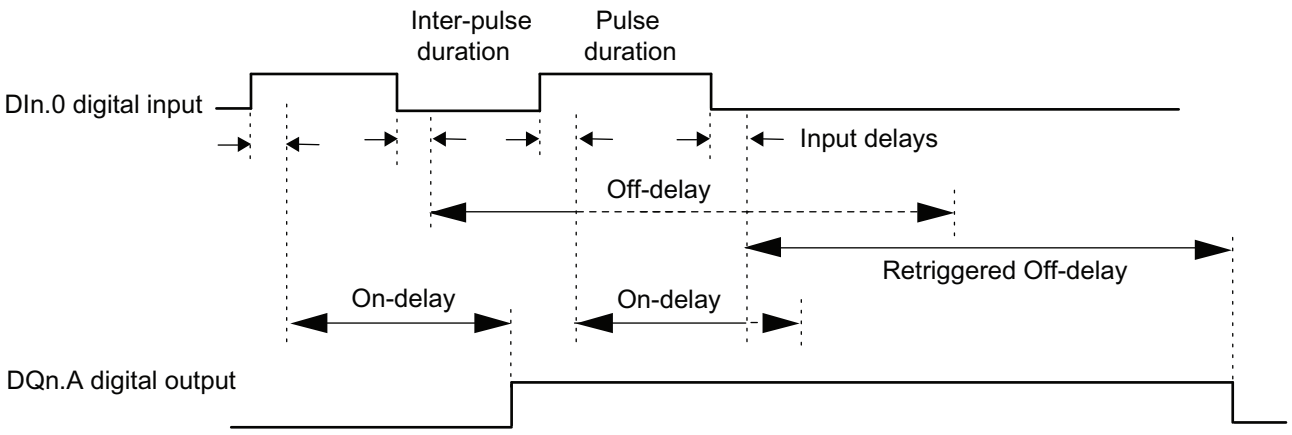

Figure 5-12 Retriggering the current off-delay

Setting and changing the OUTPUT_VALUE (on-delay)

- You set the OUTPUT_VALUE of the on-delay directly by assigning a value with the control program.
- The unit is always the microsecond. Possible range: Between 0 µs and 85 000 000 µs. If the program assigns an invalid value, the module in the feedback interface outputs the error ERR_OUT_VAL and continues to use the last valid value.
- The new on-delay value is activated with the next rising edge at the digital input DIn.0.

Setting and changing the on-delay

- Continuous updating
  The off-delay can be continuously controlled via the control interface. The MODE_SLOT bit must be set (continuous updating). LD_SLOT must have the value 3 (for off-delay). Set the off-delay to a value between 0 µs and 85 000 000 µs in the SLOT field. The unit is always the microsecond.
- Individual update
  In the configuration parameters set the off-delay to a value between 0 µs and 85 000 000 µs. Alternatively, execute an individual updating via the control interface. MODE_SLOT must be deleted (individual updating). LD_SLOT must have the value 3 (for off-delay). Set the off-delay to a value between 0 µs and 85 000 000 µs in the SLOT field.

If you change the off-delay value during the output sequence, the new off-delay is activated with the next falling edge at the Digital input DIn.0.
Additional information on the handling of the SLOT parameter is available under Handling of the SLOT parameter.

Isochronous mode

General information is available under Function: Isochronous mode.

The isochronous mode does not have any influence on the function principle of the on-/off-delay operating mode.

Parameter for the on-/off-delay operating mode

Parameter	Meaning	Value range	Default
Operating mode	3 = Setting the "On-/Off-delay" operating mode	0 = Pulse output 1 = Pulse width modulation 2 = Pulse train 3 = On-/Off-delay 4 = Frequency output 5 = Direct current	1
High-speed pulse output [1]	This output supports higher frequencies.	0 = deactivated 1 = activated	Deactivated
DI function	The digital input DIn.0 can only be used as an input.	0 = input 1 = reserved (not usable)	Input
Input delay	The DI digital input has to remain stable over the course of the delay (suppression of signal noise).	0 = Off (4 µs) 1 = 0.05 ms 2 = 0.1 ms 3 = 0.4 ms 4 = 0.8 ms 5 = 1.6 ms 6 = 3.2 ms 7 = 12.8 ms 8 = 20 ms	0.1 ms
Off-delay	The time duration from the falling edge of the DIn.0 digital input up to the falling edge of DQn.A. You change the off-delay in the control program by using the SLOT parameter.	0 µs to 85 000 000 µs	0 µs

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)".

Control and feedback signals for the on-/off-delay operating mode

Control interface Offset to the start address		Parameter	Meaning				
Channel 0	Channel 1 [1]						
Bytes 0 to 3	Bytes 12 to 15	OUTPUT_VALUE (DWord)	On-delay				
			0 µs to 85 000 000 µs				
Bytes 4 to 7	Bytes 16 to 19	SLOT (DWord)	The off-delay can be changed at any time but only takes effect with the next falling edge of the digital input DIn.0.				
			0 µs to 85 000 000 µs				
Byte 8	Byte 20	LD_SLOT	Interpretation of the SLOT value: All values not listed subsequently are invalid and cause the error ERR_LD (in the operating mode for single updates) or ERR_SLOT_VAL (in the operating mode for continuous updates).				
			Bit 3	Bit 2	Bit 1	Bit 0	
			0	0	0	0	Idle; the value is not processed
			0	0	1	1	Off-delay time in micro seconds
Byte 8: Bit 4	Byte 20: Bit 4	MODE_SLOT	Bit 4	Operating mode for using the SLOT field.			
			0	Operating mode for individual update			
			1	Operating mode for continuous update			

Control interface Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 9: Bit 0	Byte 21: Bit 0	SW_ENABLE	**Bit 0**	Software enable: Starting and stopping of the output sequence.
			0	Output aborted
			0 → 1	Starts output sequence at rising edge
Byte 9: Bit 1	Byte 21: Bit 1	TM_CTRL_DQ	**Bit 1**	Setting the DQn.A output source: Affects the selection of the CPU program or output sequence of the module.
			0	DQn.A and DQn.B are controlled by the CPU (in the program) by means of the control bits SET_DQA and SET_DQB.
			1	DQn.A is controlled by the pulse output sequence of the module. DQn.B is always 0 for TM_CTRL = 1.
Byte 9: Bit 3	Byte 21: Bit 3	SET_DQA	**Bit 3**	Controls the value of the digital output DQn.A, if TM_CTRL_DQ is deleted.
			0	0 at DQn.A
			1	1 at DQn.A
Byte 9: Bit 4	Byte 21: Bit 4	SET_DQB	**Bit 4**	Controls the value of the digital output DQn.B, if TM_CTRL_DQ and SET_DQA are deleted.
			0	0 at DQn.B
			1	1 at DQn.B
Byte 10: Bit 0	Byte 22: Bit 0	RES_ERROR	**Bit 0**	Reset pending error (ERR_LD, ERR_DQA, ERR_DQB and ERR_24V).
			0	Reset of errors is not active
			1	Reset of errors is active

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)"

Note: All the bytes and bits not described in the preceding table are reserved and have to be 0.

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 0: Bit 0	Byte 8: Bit 0	ERR_PWR	**Bit 0**	Indicates undervoltage in the power supply. The bit is not set if no current is available.
			0	PWR has no undervoltage.
			1	PWR was detected, but has undervoltage.
Byte 0: Bit 1	Byte 8: Bit 1	ERR_24V	**Bit 1**	Indicates a short-circuit or an overload at the 24 V DC output. To reset this error and be able to use SLOT again, you have to set the RES_ERROR (control interface).
			0	No short-circuit at 24 V DC
			1	Short-circuit at 24 V DC
Byte 0: Bit 2	Byte 8: Bit 2	ERR_LD	**Bit 2**	Indicates an error when loading a value with the SLOT field (only in the SLOT operating mode for individual updating).
			0	No load error pending.
			1	Pending load error: To reset this error and be able to use SLOT again, you have to set the RES_ERROR (control interface).
Byte 0: Bit 3	Byte 8: Bit 3	ERR_PULSE	**Bit 3**	Indicates a pulse output error.
			0	No pulse output error.
			1	Pulse output error

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 0: Bit 4	Byte 8: Bit 4	ERR_DQA	**Bit 4**	Indicates a short-circuit at output DQn.A. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.A
			1	Short-circuit at DQn.A
Byte 0: Bit 5	Byte 8: Bit 5	ERR_DQB	**Bit 5**	Indicates a short-circuit at output DQn.B or the attempted manual setting of DQn.A and DQn.B by means of SET_DQA, SET_DB and TMCTRL_DQ. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.B
			1	Short-circuit at DQn.B or the attempted setting of DQn.A and DQn.B
Byte 0: Bit 6	Byte 8: Bit 6	ERR_OUT_VAL	**Bit 6**	Indicates that an invalid value was detected in OUTPUT_VALUE.
			0	OUTPUT_VALUE is valid
			1	OUTPUT_VALUE is invalid
Byte 0: Bit 7	Byte 8: Bit 7	ERR_SLOT_VAL	**Bit 7**	Indicates that an invalid value was detected in SLOT (only in SLOT operation mode for continuous upgrading).
			0	SLOT value is valid
			0 → 1	SLOT value is invalid
Byte 1: Bit 2	Byte 9: Bit 2	STS_LD_SLOT	**Bit 2**	Switchover acknowledgement bit for each action of the SLOT in the SLOT operating mode for individual update. Every switchover of this bit means a successful LD_SLOT action.
Byte 1: Bit 4	Byte 9: Bit 4	STS_READY	**Bit 4**	Indicates that the module is ready and configured.
			0	Module is not configured
			1	Module is configured
Byte 1: Bit 5	Byte 9: Bit 5	STS_SW_ENABLE	**Bit 5**	Indicates the status of SW_ENABLE (control interface).
			0	SW_ENABLE deleted
			1	SW_ENABLE set
Byte 2: Bit 0	Byte 10: Bit 0	STS_ENABLE	**Bit 0**	Indicates a running output sequence.
			0	The pulse output is inactive
			1	The pulse output is active
Byte 2: Bit 1	Byte 10: Bit 1	STS_DQA	**Bit 1**	Indicates the signal level at digital output DQn.A.
			0	0 at the digital output DQn.A
			1	1 at the digital output DQn.A
Byte 2: Bit 2	Byte 10: Bit 2	STS_DQB	**Bit 2**	Indicates the signal level at digital output DQn.B.
			0	0 at the digital output DQn.B
			1	1 at the digital output DQn.B
Byte 2: Bit 3	Byte 10: Bit 3	STS_DI	**Bit 3**	Indicates the signal level at the digital input DIn.0.
			0	0 at the digital input DIn.0
			1	1 at the digital input DIn.0
Byte 3: Bits 0 to 3	Byte 11: Bits 0 to 3	SEQ_CNT	Counting of rising and falling edges at the DQn.A output.	
Word 3	Word 7	Reserved	Read as 0	

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)"

Note: All bytes and bits not described in the preceding table are reserved and are read as 0.

Input and output signals for on-/off-delay operating mode

Input and output signal	Meaning	Value range	Channel 0 BaseUnit Pin number	Channel 1 Base-Unit Pin number
Input signal				
Digital input DIn.0	The signal at the DIn.0 digital input is output with an on-/off-delay at the digital output DQn.A.	0 = No pulse 1 = Pulse	3	4
Output signal				
Pulse on the digital output DQn.A	The signal at the DIn.0 digital input is output with an on-/off-delay at the digital output DQn.A.	0 = No signal 1 = Signal	9	10

See also Pin assignment und load/encoder wiring

5.1.2.6 Frequency output mode (S7-1500)

Definition

In this operating mode you can assign a frequency value with high frequencies more precisely than by using PWM period duration and period duration.

A rectangular signal with an assigned frequency and a constant duty cycle of 50% is generated at the digital output of the TM Pulse 2x24V.

The output sequence is started after expiration of the configured on-delay on the DQn.A digital output.

Pulse diagram

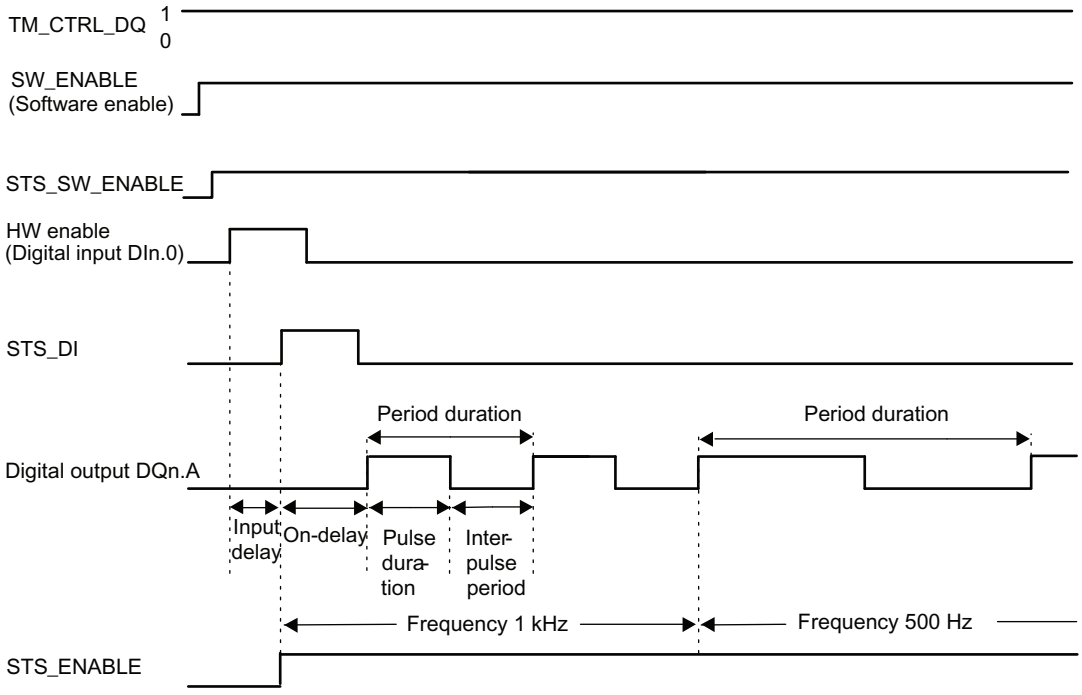

Figure 5-13 Frequency output - output sequence

In the pulse diagram above the "DI function" parameter is set to "HW enable". For the other option the "DI function" is set to "Input". If the parameter "DI function" is set to "Input", the on-delay begins with the rising edge of SW_ENABLE.

Starting the output sequence

The control program must output the enable for the output sequence by means of the software enable (SW_ENABLE 0 → 1.)

The STS_SW_ENABLE feedback bit indicates that the software enable is pending at the TM Pulse 2x24V.

In addition you also set the digital input DIn.0 of the TM Pulse 2x24V as HW enable via the "DI function" parameter.

If you want to use the hardware enable, it must be combined with the software enable. In case of activated software enable with SW_ENABLE the output sequence starts with the first rising edge of the Hardware enable. Additional rising edges of the hardware enable are ignored during the current output sequence of the TM Pulse 2x24V.

If the enable is activated (rising edge) and "High" ("On") remains (noise filter) for the duration of the input delay, the on-delay is started and STS_ENABLE is set. The frequency sequence

is output when the on-delay expires. The output sequence runs continuously, as long as SW_ENABLE is set.

Note

Output control signal TM_CTRL_DQ of the technology module

- If TM_CTRL_DQ = 1, the TM Pulse 2x24V module takes over the control and generates pulse sequences at the output DQn.A.
- If TM_CTRL_DQ = 0, the CPU takes over the control and the program can set the DQn.A/DQn.B outputs via the SET_DQA/SET_DQB control bits.

Aborting the output sequence

When the software enable is deactivated ((SW_ENABLE = 1 → 0)) during the on-delay or the frequency output, the current output sequence is aborted and the last period duration is not completed. STS_ENABLE and the digital output are reset to 0 immediately.

A renewed pulse output is only possible after a restart of the output sequence.

Truth table

Software enable SW_ENABLE	DI function parameter	Hardware enable (Digital input DIn.0)	Digital output DQn.A (if TM_CTRL_DQ = 1)	STS_ENABLE	Output sequence
1	HW_ENABLE	0 → 1 and remains 1 during the input delay Only active with the first rising edge, further rising edges are ignored, no start is carried out.	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0 → 1	Input	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	HW_ENABLE or input	Any status	0	0	Canceling
1	HW_ENABLE or input	Any status	0, if the on-delay has not expired or the moment lies between pulse durations 1, if the on-delay has expired and the pulse duration is still active		-
0 → 1	HW_ENABLE	0	0	0	-

Setting and changing of the output value (frequency)

You set the OUTPUT_VALUE directly with the control program in the control interface. The value is specified in the Real format and the unit is always "Hz". The possible range depends on the parameter "Rapid pulse output" as follows:

- High-speed pulse output deactivated
  - Frequency (OUTPUT_VALUE): 0.02 Hz to 10 000 Hz
- High-speed pulse output activated
  - Frequency (OUTPUT_VALUE): 0.02 Hz to 100 000 Hz

The new frequency value is applied at the next rising edge of the output.

Accuracy of the output sequence

The configured output sequence is output with an accuracy of +/- 100 ppm at the digital output DQn.A.

Setting and changing the on-delay

- Continuous updating
  The on-delay can be continuously controlled via the control interface. The MODE_SLOT bit must be set (continuous updating). LD_SLOT must have the value 2 (for on-delay). Set the on-delay to a value between 0 µs and 85 000 000 µs in the SLOT field. The unit is always the microsecond.
- Individual updating
  Set the on-delay to a value between 0 µs and 85 000 000 µs in the configuration parameters, the unit is always microseconds. Alternatively, execute an individual updating via the control interface. MODE_SLOT must be deleted (individual updating). LD_SLOT must have the value 2 (for on-delay). Set the on-delay to a value between 0 µs and 85 000 000 µs in the SLOT field.

If you change the on-delay value during the output sequence, the new on-delay is activated with the next output sequence.

Additional information on the handling of SLOT parameter is available under Handling of the SLOT parameter.

Isochronous mode

General information is available under "Function: Isochronous mode".

The isochronous mode does not have any influence on the functionality of the frequency output operating mode.

If you want to synchronize the output sequence with T_o , set the "DI function" parameter to "Input" and the frequency output sequence starts at T_o.

Parameters of the frequency output operating mode

Parameter	Meaning	Value range	Default
Operating mode	4 = Setting the frequency output operating mode.	0 = Pulse output 1 = Pulse width modulation 2 = Pulse train 3 = On-/Off-delay 4 = Frequency output 5 = Direct current	1
High-speed pulse output [1]	This output supports higher frequencies.	0 = deactivated 1 = activated	Deactivated
DI function	You can use the DIn.0 digital input as an input or as a hardware enable.	0 = input 1 = HW enable	Input

Parameter	Meaning	Value range	Default
Input delay	The DIn.0 digital input has to remain stable over the course of the delay (suppression of signal noise).	0 = Off (4 µs) 1 = 0.05 ms 2 = 0.1 ms 3 = 0.4 ms 4 = 0.8 ms 5 = 1.6 ms 6 = 3.2 ms 7 = 12.8 ms 8 = 20 ms	0.1 ms
On-delay	The duration from the start of the output sequence to the pulse output. You can change the on-delay in the control program by using the SLOT parameter.	0 µs to 85 000 000 µs	0 µs

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)".

Control and feedback signals for the frequency output operating mode

Control interface Offset to the start address		Parameter	Meaning				
Channel 0	Channel 1 [1]						
Bytes 0 to 3	Bytes 12 to 15	OUTPUT_VALUE (Real)	Frequency of the DQn.A digital output as a Real number. The unit is always Hz. If you do not observe the lower or upper limit of the range, TM Pulse 2x24V uses the last valid value and the ERR_OUT_VAL error is activated.				
			High-speed pulse output de-activated				**High-speed pulse output activated**
			0.02 Hz to 10 000 Hz				0.02 Hz to 100 000 Hz
Bytes 4 to 7	Bytes 16 to 19	SLOT (DWord)	The on-delay can be changed before the start of the output sequence. See MODE_SLOT.				
			0 µs to 85 000 000 µs				
Byte 8	Byte 20	LD_SLOT	Interpretation of the SLOT value: All values not listed subsequently are invalid and cause the error ERR_LD (in the operating mode for single updates) or ERR_SLOT_VAL (in the operating mode for continuous updates).				
			Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	
			0	0	0	0	Idle; the value is not processed
			0	0	1	0	On-delay in microseconds
Byte 8: Bit 4	Byte 20: Bit 4	MODE_SLOT	**Bit 4**	Operating mode for using the SLOT field.			
			0	Operating mode for individual update			
			1	Operating mode for continuous update			
Byte 9: Bit 0	Byte 21: Bit 0	SW_ENABLE	**Bit 0**	Software enable: Start/activate and stop/deactivate the output se-quence.			
			0	Output deactivated/terminated			
			0 → 1	Starts the output sequence for a rising edge if "DI function" = "Input".			
			1	Activates the output sequence, if the start depends on the HW enable with "DI function" = "HW enable".			

Control interface Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 9: Bit 1	Byte 21: Bit 1	TM_CTRL_DQ	**Bit 1**	Setting the DQn.A output source: Affects the selection of the CPU program or output sequence of the module.
			0	DQn.A and DQn.B are controlled by the CPU (in the program) by means of the control bits SET_DQA and SET_DQB.
			1	DQn.A is controlled by the pulse output sequence of the module. DQn.B is always 0.
Byte 9: Bit 3	Byte 21: Bit 3	SET_DQA	**Bit 3**	Controls the value of the digital output DQn.A, if TM_CTRL_DQ is deleted.
			0	0 at DQn.A
			1	1 at DQn.A
Byte 9: Bit 4	Byte 21: Bit 4	SET_DQB	**Bit 4**	Controls the value of the digital output DQn.B, if TM_CTRL_DQ and SET_DQA are deleted.
			0	0 at DQn.B
			1	1 at DQn.B
Byte 10: Bit 0	Byte 22: Bit 0	RES_ERROR	**Bit 0**	Reset pending error (ERR_LD, ERR_DQA, ERR_DQB and ERR_24V).
			0	Reset of errors is not active
			1	Reset of errors is active

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)"

Note: All the bytes and bits not described in the preceding table are reserved and have to be 0.

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1[1]			
Byte 0: Bit 0	Byte 8: Bit 0	ERR_PWR	**Bit 0**	Indicates undervoltage in the power supply. The bit is not set if no current is available.
			0	PWR has no undervoltage
			1	PWR was detected, but has undervoltage
Byte 0: Bit 1	Byte 8: Bit 1	ERR_24V	**Bit 1**	Indicates a short-circuit or an overload at the 24 V DC output. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at 24 V DC
			1	Short-circuit at 24 V DC
Byte 0: Bit 2	Byte 8: Bit 2	ERR_LD	**Bit 2**	Indicates an error when loading a value with the SLOT field (only in the SLOT operating mode for individual updating).
			0	No load error pending
			1	Load error pending. To reset this error and be able to use SLOT again, you have to set the RES_ERROR (control interface).
Byte 0: Bit 4	Byte 8: Bit 4	ERR_DQA	**Bit 4**	Indicates a short-circuit at output DQn.A. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.A
			1	Short-circuit at DQn.A

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1[1]			
Byte 0: Bit 5	Byte 8: Bit 5	ERR_DQB	**Bit 5**	Indicates a short-circuit at output DQn.B or the attempted manual setting of DQn.A and DQn.B by means of SET_DQA, SET_DB and TMCTRL_DQ. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.B
			1	Short-circuit at DQn.B or the attempted setting of DQn.A and DQn.B
Byte 0: Bit 6	Byte 8: Bit 6	ERR_OUT_VAL	**Bit 6**	Indicates that an invalid value was detected in OUTPUT_VALUE.
			0	OUTPUT_VALUE is valid
			1	OUTPUT_VALUE is invalid
Byte 0: Bit 7	Byte 8: Bit 7	ERR_SLOT_VAL	**Bit 7**	Indicates that an invalid value was detected in SLOT (only in SLOT operation mode for continuous upgrading).
			0	SLOT value is valid
			0 → 1	SLOT value is invalid
Byte 1: Bit 2	Byte 9: Bit 2	STS_LD_SLOT	**Bit 2**	Switchover acknowledgement bit for each action of the SLOT in the SLOT operating mode for individual update. Every switchover of this bit means a successful LD_SLOT action.
Byte 1: Bit 4	Byte 9: Bit 4	STS_READY	**Bit 4**	Indicates that the module is ready and configured.
			0	Module is not configured
			1	Module is configured
Byte 1: Bit 5	Byte 9: Bit 5	STS_SW_ENABLE	**Bit 5**	Indicates the status of SW_ENABLE (control interface).
			0	SW_ENABLE deleted
			1	SW_ENABLE set
Byte 2: Bit 0	Byte 10: Bit 0	STS_ENABLE	**Bit 0**	Indicates an output sequence.
			0	The pulse output is inactive
			1	The pulse output is active
Byte 2: Bit 1	Byte 10: Bit 1	STS_DQA	**Bit 1**	Indicates the signal level at digital output DQn.A.
			0	0 at the digital output DQn.A
			1	1 at the digital output DQn.A
Byte 2: Bit 2	Byte 10: Bit 2	STS_DQB	**Bit 2**	Indicates the signal level at digital output DQn.B.
			0	0 at the digital output DQn.B
			1	1 at the digital output DQn.B
Byte 2: Bit 3	Byte 10: Bit 3	STS_DI	**Bit 3**	Indicates the signal level at the digital input DIn.0.
			0	0 at the digital input DIn.0
			1	1 at the digital input DIn.0
Byte 3: Bits 0 to 3	Byte 11: Bits 0 to 3	SEQ_CNT	Sequence counter = 0. The sequence counter is not used in the frequency output operating mode.	
Word 3	Word 7	Reserved	Read as 0	

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)"

Note: All bytes and bits not described in the preceding table are reserved and are read as 0.

Input and output signals for frequency output operating mode

Input and output signal	Meaning	Value range	Channel 0 BaseUnit Pin number	Channel 1 BaseUnit Pin number
Input signal				
HW enable	You can select the HW enable with the "DI function" parameter and select the input delay with the "Input delay" parameter. The signal of the DIn.0 digital input is interpreted by the TM Pulse 2x24V as the start of the output sequence after noise filtering by the input delay.	0 = HW enable cleared 1 = HW enable activated 0 → 1 = Start of the output sequence after the input delay, dependent on the software enable (SW_ENABLE)	3	4
Output signal				
Pulse on the digital output DQn.A	A pulse is output at the digital output DQn.A for the assigned frequency.	0 = No pulse 1 = Pulse	9	10

See also Pin assignment und load/encoder wiring

5.1.2.7 DC motor mode (S7-1500)

Definition

Each channel has an A and a B output for the connection to the DC current motor load. The direction of rotation and the duty cycle of the output voltage are assigned via the bipolar output switch and the pulse width modulation.

You can use a two-channel circuit for a maximum of 2 A per channel or a one channel parallel circuit with a maximum of 4 A.

Two-channel motor load connection:

- Channel 0: Motor connected between DQ0.A and DQ0.B
- Channel 1: Motor connected between DQ1.A and DQ1.B

One channel motor load connection

- 4-A channel 0: Motor connected between DQ0.A and DQ0.B.
  DQ0.A and DQ1.A are connected to each other; DQ0.B und DQ1.B are connected to each other.

For additional information on the connection, refer to chapter Connection of this manual.

Motor rotation

Direction of rotation

- Forward during the high phase of the PWM signal: DQn.A is 1 and DQn.B is 0
- Reverse during the high phase of the PWM signal: DQn.A is 0 and DQn.B is 1
- Forward/reverse during the low phase of the PWM signal: DQn.A is 0 and DQn.B is 0

Motor stop

The motor can be stopped by using an external signal connected to the DIn.0 digital input or by using the SW_ENABLE signal in the control interface. Configure the "DI function" accordingly.

Setting of the DI function:

- The "HW enable" starts the motor at the rising edge of the DI and stops the motor at the falling edge of the DI.
- "External stop" starts the motor at the rising edge of SW_ENABLE and stops the motor at the rising edge at DI or the falling edge of SW_ENABLE.
- "Input" starts the motor at the rising edge of SW_ENABLE and stops the motor at the falling edge of SW_ENABLE (DI has no influence on the motor control).

You control the duty cycle of the output pulse with the OUTPUT_VALUE field of the control interface. The TM Pulse 2x24V generates continuous pulses based on this value. The OUTPUT_VALUE determines the duty cycle (relationship pulse duration/period duration) within a period for a pulse width modulation. The period duration is adjustable.

After the assigned on-delay has expired, the DQn.A and DQn.B output pulses (output sequence) begin.

Quick pulse output, current measurement and current control are not available in the DC current operating mode.

Starting the output sequence

The control program must output the enable for the output sequence by using the software enable (SW_ENABLE 0 → 1.)

The STS_SW_ENABLE feedback bit indicates that the software enable is pending at the TM Pulse 2x24V.

You can also assign the digital input DIn.0 of the TM Pulse 2x24V as a HW enable by using the parameter "DI function". The input delay (noise filter) of the hardware enable can be configured via the "Input delay" parameter.

If you want to use the hardware enable, it must be combined with the software enable. In case of activated software enable with SW_ENABLE the output sequence starts with the first rising edge of the Hardware enable. Additional rising edges of the hardware enable are ignored during the current output sequence of the TM Pulse 2x24V. The hardware enable option is not supported in the isochronous mode.

If the enable is activated (rising edge) and the input delay remains "High" ("On"), the on-delay is started and STS_ENABLE is set. The PWM pulse train is output when the on-delay expires. The output sequence runs continuously, as long as SW_ENABLE is set.

Aborting the output sequence

- Use of the SW_ENABLE signal: By deactivating the software enable (SW_ENABLE = 1→0) the current output sequence is terminated and the last period duration is not completed. STS_ENABLE and digital outputs DQn.A and DQn.B are immediately set to 1 (motor stop).

A renewed pulse output is only possible after a restart of the output sequence.

- Use of the digital input DIn.0:
  - If the "DI function" is configured as "HW_ENABLE": A falling edge at DIn.0 stops the output sequence with the same behavior as during the use of SW_ENABLE.
  - If the "DI function" is configured as "external stop": A falling edge at DIn.0 stops the sequence with the same behavior as during the use of SW_ENABLE.

Truth table

Software enable SW_ENABLE	DI function parameter	Hardware enable (Digital input DIn.0)	Digital output DQn.A and DQn.B		STS_ENABLE	Output sequence
1	HW_ENABLE	0 → 1 and remains 1 during the input delay Only active with the first rising edge, further rising edges are ignored, no start.	0, if on-delay > 0: DQn.A: 0 DQn.B: 0		0 → 1	Starting
			If on-delay = 0:			
			Up: DQn.A: 1 DQn.B: 0	Down: DQn.A: 0 DQn.B: 1		
0 → 1	Input or external stop	Any status	If on-delay > 0 DQn.A: 0 DQn.B: 0		0 → 1	Starting
			If on-delay = 0			
			Up: DQn.A: 1 DQn.B: 0	Down: DQn.A: 0 DQn.B: 1		
0	HW_ENABLE or input or external stop	Any status	DQn.A: Tri state DQn.B: Tri state		0	Canceling
1	Input	Any status	If the input delay has not expired or is located in the interpulse duration: DQn.A: 0 DQn.B: 0			-
			If the input delay has expired and is located in the pulse duration:			
			Up: DQn.A: 1 DQn.B: 0	Down: DQn.A: 0 DQn.B: 1		
1	HW_ENABLE	0	DQn.A: Tri state DQn.B: Tri state		1 → 0	Stop
1	External stop	0 → 1	DQn.A: Tri state DQn.B: Tri state		1 → 0	Stop
0 → 1	HW_ENABLE	0	DQn.A: Tri state DQn.B: Tri state		0	-

Setting and changing the pulse duty cycle and the direction of rotation

OUTPUT_VALUE assigns the duty cycle and the direction for the current period duration.

OUTPUT_VALUE is specified as an S7 analog value, the sign indicates the direction of the motor rotation ("+" means forwards, "-" means backwards). The possible range lies between -27.648 and +27.648.

Only the two lowest bytes of OUTPUT_VALUE are used, the other two bytes are ignored. A value of the two lowest bytes larger than +27.648 is interpreted as value +27.648 (100% forward). A value of the two lowest bytes smaller than +27.648 is interpreted as the value -27.648 (100% reverse).

Before changing the direction of rotation it is recommended to set OUTPUT_VALUE to 0 long enough so that the motor is initially stopped. Please take into consideration that the TM Pulse 2x24V module does not overshoot the assigned duty cycle to protect the motor. If a startup or return is required at the motor, this ramp must be implemented in the automation program and transferred by corresponding control of the value of the OUTPUT_VALUE field.

- Output format "S7 analog output": Value range between -27 648 and +27 648. The sign determines the direction of rotation of the motor.
- Pulse duration = (OUTPUT_VALUE/27 648) x period duration.

You assign OUTPUT_VALUE directly via the control program. A new OUTPUT_VALUE is applied at the output when the next rising edge occurs.

Setting and changing the period duration

- Continuous updating
  The period duration is continuously controlled via the control interface. The MODE_SLOT bit must be set ("1" means continuous updating). LD_SLOT must be set to the value 1 ("1" means period duration).
  Set the pulse duration value in the SLOT field between 100 µs and 85 000 000 µs.
- Individual updating
  Set the period duration in the configuration parameters. Alternatively, execute an individual updating via the control interface. MODE_SLOT must be deleted ("0" means individual updating). LD_SLOT must be set to value 1 ("1" means period duration).
  Set the period duration value in the SLOT field between 100 µs and 85 000 000 µs.

A new period duration is applied at the output when the next rising edge occurs.

Additional information on the handling of SLOT parameter is available under "Handling of the SLOT parameter (control interface)".

Isochronous mode

General information is available under "Function: Isochronous mode".

In isochronous mode, the output sequence is synchronized with the T_O moment. The period duration is tailored to the application cycle (meaning the cycle as a multiple of the PROFINET cycle). The behavior of the DC motor operating mode is identical with the behavior in the PWM

operating mode. Read the corresponding chapter in the manual to this purpose. Below only the differences are listed:

- During the "continuous pulse phase" (high phase) the status of the outputs is as follows:
  - In the forward direction: DQn.A is 1 and DQn.B is 0
  - In the reverse direction: DQn.A is 0 and DQn.B is 1
- During the "interpulse period phase" (high phase) the status of the outputs is as follows:
  - DQn.A is 0 and DQn.B is 0
- The "DI function" parameter can be configured as an "external stop" in isochronous mode in order to use the rising edge at DIn.0 to stop the motor. Please note that in the case of parameter assignment of "Function DI" as "HW_ENABLE" in isochronous mode, which is not supported, this parameters assignment is interpreted as an "external stop" by the TM Pulse 2x24V module.

Setting and changing the on-delay

- Continuous updating
  The on-delay can be continuously controlled via the control interface. The MODE_SLOT bit must be set (continuous updating); LD_SLOT must have the value 2 (for on-delay). Set the on-delay to a value between 0 µs and 85 000 000 µs in the SLOT field. The unit is always the microsecond.
- Individual updating
  Set the on-delay to a value between 0 µs and 85 000 000 µs in the configuration parameters. The unit is always the microsecond. Alternatively, execute an individual updating via the control interface. MODE_SLOT must be deleted (continuous updating). LD_SLOT must be set to the value 2 (for on-delay). Set the on-delay to a value between 0 µs and 85 000 000 µs in the SLOT field.

If you change the on-delay value during the output sequence, the new on-delay is activated with the next output sequence. Additional information on the handling of SLOT parameter is available under Handling of the SLOT parameter.

Parameter for the DC motor operating mode

Parameter	Meaning	Value range	Default
Operating mode	5 = Setting the direct current operating mode	0 = Pulse output 1 = Pulse width modulation 2 = Pulse train 3 = On-/Off-delay 4 = Frequency output 5 = Direct current	1
DI function	You can use the DIn.0 digital input as an input for HW_ENABLE or for the signal "External stop". The signal at DIn.0 is interpreted by the TM Pulse 2x24V as the start of the output sequence after noise filtering by the input delay. If HW_ENABLE is used in non-isochronous operation, a rising edge at DIn.0 starts the output sequence (if SW_ENABLE is set) and a falling edge at DIn.0 stops the output sequence. If HW_ENABLE is used in isochronous mode, the module is interpreted as an external stop. If the external stop is used, the output sequence starts with the rising edge of SW_ENABLE and stops with the rising edge of DIn.0.	0 = input 1 = HW enable 2 = External stop	Input
Output format	Defines the format of the relationship value (duty cycle).	0 = S7 analog format	S7 analog format
Input delay	The DIn.0 digital input has to remain stable over the course of the delay (suppression of signal noise).	0 = Off (4 µs) 1 = 0.05 ms 2 = 0.1 ms 3 = 0.4 ms 4 = 0.8 ms 5 = 1.6 ms 6 = 3.2 ms 7 = 12.8 ms 8 = 20 ms	0.1 ms
Period	Period duration of the output pulse cycle in µs. You change the period duration in the control program via the SLOT field of the control interface.	100 µs to 85 000 000 µs	1000 µs
On-delay	The duration from the start of the output sequence to the pulse output. You change the on-delay in the control program via the SLOT field of the control interface.	0 µs to 85 000 000 µs	0 µs

Control and feedback signals for the DC motor operating mode

Control interface Offset to the start address		Parameter	Meaning				
Channel 0	Channel 1 [1]						
Bytes 0 to 3	Bytes 12 to 15	OUTPUT_VALUE (DWord)	The OUTPUT_VALUE determines the duty cycle (relationship pulse duration/period duration) within a period (PWM). The period duration is adjustable. The new output value is applied at the next rising edge of the output. The OUTPUT_VALUE sign identifies the direction of rotation (positive for forward and negative for backward). Output format "S7 analog output": Value range -27 648 to +27 648 DINT data type: Only the 2 lowest bytes are used For Channel 0: Bytes 2 and 3 For Channel 1: Bytes 14 and 15				
Bytes 4 to 7	Bytes 16 to 19	SLOT (DWord)	Your program can change the on-delay and period duration before the start of the output sequence by using the SLOT and MODE_SLOT parameters.				
			0 µs to 85 000 000 µs				
Byte 8	Byte 20	LD_SLOT	Interpretation of the SLOT value: All values not listed subsequently are invalid and cause the error ERR_LD (in the operating mode for single updates) or ERR_SLOT_VAL (in the operating mode for continuous updates).				
			Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	
			0	0	0	0	Idle; the value is not processed
			0	0	0	1	Period in µs
			0	0	1	0	On-delay in µs
Byte 8: Bit 4	Byte 20: Bit 4	MODE_SLOT	**Bit 4**	Operating mode for using the SLOT field.			
			0	Operating mode for individual update			
			1	Operating mode for continuous update			
Byte 9: Bit 0	Byte 21: Bit 0	SW_ENABLE	**Bit 0**	Software enable: Starting and stopping of the output sequence.			
			0	Output aborted			
			0 → 1	Starts the output sequence at a rising edge if "DI function" = "Input" or "DI function" = "External stop".			
			1	Activates the output sequence, if the start depends on the HW enable with "DI function" = "HW enable".			
Byte 9: Bit 1	Byte 21: Bit 1	TM_CTRL_DQ	**Bit 1**	The outputs are always controlled by the module in the DC motor operating mode. This bit is ignored.			
			X	Indifferent: No effect on the outputs.			
Byte 9: Bit 3	Byte 21: Bit 3	SET_DQA	**Bit 3**	Controls the value of the digital output DQn.A, if TM_CTRL_DQ = 0.			
			X	Indifferent: No effect on the outputs.			
Byte 9: Bit 4	Byte 21: Bit 4	SET_DQB	**Bit 4**	Controls the value of the digital output DQn.B, if TM_CTRL_DQ = 0.			
			X	Indifferent: No effect on the outputs.			
Byte 10: Bit 0	Byte 22: Bit 0	RES_ERROR	**Bit 0**	Reset pending error (ERR_LD, ERR_DQA, ERR_DQB and ERR_24V).			
			0	Reset of errors is not active			
			1	Reset of errors is active			

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)"

Note: All the bytes and bits not described in the preceding table are reserved and have to be 0.

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 0: Bit 0	Byte 8: Bit 0	ERR_PWR	**Bit 0**	Indicates undervoltage in the power supply. The bit is not set if no current is available.
			0	PWR has no undervoltage
			1	PWR was detected, but has undervoltage
Byte 0: Bit 1	Byte 8: Bit 1	ERR_24V	**Bit 1**	Indicates a short-circuit or an overload at the 24 V DC output. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at 24 V DC
			1	Short-circuit at 24 V DC
Byte 0: Bit 2	Byte 8: Bit 2	ERR_LD	**Bit 2**	Indicates an error when loading a value with the SLOT field (only in the operating mode for individual updating SLOT).
			0	No load error pending
			1	Pending load error: To reset this error and be able to use SLOT again, you have to set the RES_ERROR (control interface).
Byte 0: Bit 4	Byte 8: Bit 4	ERR_DQA	**Bit 4**	Indicates a short-circuit at output DQn.A. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.A
			1	Short-circuit at DQn.A
Byte 0: Bit 5	Byte 8: Bit 5	ERR_DQB	**Bit 5**	Indicates a short-circuit at output DQn.B. To reset this error you have to set RES_ERROR (control interface).
			0	No short-circuit at DQn.B
			1	Short-circuit at DQn.B
Byte 0: Bit 7	Byte 8: Bit 7	ERR_SLOT_VAL	**Bit 7**	Indicates that an invalid value was detected in SLOT (only in the operating mode for continuous SLOT upgrading).
			0	SLOT value is valid
			0 → 1	SLOT value is invalid
Byte 1: Bit 2	Byte 9: Bit 2	STS_LD_SLOT	**Bit 2**	Switchover acknowledgement bit for each action of the SLOT in the SLOT operating mode for individual update. Every switchover of this bit means a successful LD_SLOT action.
Byte 1: Bit 4	Byte 9: Bit 4	STS_READY	**Bit 4**	Indicates that the module is ready and configured.
			0	Module is not configured
			1	Module is configured
Byte 1: Bit 5	Byte 9: Bit 5	STS_SW_ENABLE	**Bit 5**	Indicates the status of SW_ENABLE (control interface).
			0	SW_ENABLE deleted
			1	SW_ENABLE set
Byte 2: Bit 0	Byte 10: Bit 0	STS_ENABLE	**Bit 0**	Indicates a running output sequence.
			0	Output sequence not running
			1	Output sequence running
Byte 2: Bit 1	Byte 10: Bit 1	STS_DQA	**Bit 1**	Indicates the signal level at digital output DQn.A.
			0	0 at the digital output DQn.A
			1	1 at the digital output DQn.A
Byte 2: Bit 2	Byte 10: Bit 2	STS_DQB	**Bit 2**	Indicates the signal level at digital output DQn.B.
			0	0 at the digital output DQn.B
			1	1 at the digital output DQn.B

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1 [1]			
Byte 2: Bit 3	Byte 10: Bit 3	STS_DI	**Bit 3**	Indicates the signal level at the digital input DIn.0.
			0	0 at the digital input DIn.0
			1	1 at the digital input DIn.0
Byte 3: Bits 0 to 3	Byte 11: Bits 0 to 3	SEQ_CNT	Sequence counter = 0. The sequence counter is not used in the PWM operating mode.	
Word 3	Word 7	Reserved	Read as 0	

[1] Only if the module is configured as "2-channel (2 A)" and not as "1-channel (4 A)"

Note: All bytes and bits not described in the preceding table are reserved and are read as 0.

Input and output signals for DC motor operating mode

Input and output signal	Meaning	Value range	Channel 0 BaseUnit Pin number	Channel 1 BaseUnit pin number
Input signal				
HW enable/external stop Note: The HW enable is not supported for PWM in isochronous mode	You can select the interpretation of the digital input signal at the output sequence with the "DI function" parameter and select the input delay with the "Input delay" parameter. The signal at the DIn.0 digital input is interpreted by the TM Pulse 2x24V as the start of the output sequence after noise filtering by the input delay.	DI function = "HW_ENABLE": 0 = HW enable cleared 1 = HW enable output 0→1 = Start of the output sequence after input delay, dependent on the software enable (SW_ENABLE) 1→0 = Stop of the output sequence after input delay, dependent on the software enable (SW_ENABLE). DI function = External stop: 0→1 = Start of the output sequence of the input delay, depending on the software enable (SW_ENABLE)	3	4
Output signal				
Pulse at digital outputs DQn.A and DQn.B	A pulse is output at the digital outputs DQn.A and DQn.B for the assigned duty cycle and period duration.	**No pulse or interpulse period:** DQn.A is 0 V and DQn.B is 0 V **Pulse duration in forward direction:** DQn.A is 24 V and DQn.B is 0 V **Pulse duration in reverse direction:** DQn.A is 0 V and DQn.B is 24 V	9 (DQ0.A) 11 (DQ0.B)	10 (DQ1.A) 12 (DQ1.B)

See also Pin assignment und load/encoder wiring

5.1.2.8 Function: High-speed output (S7-1500)

High-speed mode improves the signal timing of the DQ digital outputs. The switching edges have less delay, variation, jitter, and smaller rise/fall times.

High-speed mode is designed to produce more precise timing for pulse signals, but provides less maximum load current.

High-speed output is only available in dual channel operation; it is not available in single channel operation.

You can use the STEP 7 (TIA Portal) or STEP 7 hardware configuration to select "High-speed output (0.1 A)" option for each channel separately. Also, you can change the parameter assignment at runtime with your program using data record 128.

STEP 7 (TIA Portal) and STEP 7 assist you during parameter assignment, by disabling keyboard input for invalid parameters and range checking your value assignments. Depending on previous parameter selections, other options will be disabled. For example, if you select the single channel operation (the parallel connection of the two output channels), then the parameter options for Channel 1 and high-speed output are disabled.

High-speed output option is available for these operation modes:

- Pulse output
- PWM
- Pulse train
- On/Off-delay
- Frequency output

High-speed output is not available in DC motor mode.

High-speed output option

Pulse timing	Minimum		Maximum	
	High-speed disabled	High-speed enabled	High-speed disabled	High-speed enabled
Pulse duration	10 µs	1.5 µs	85,000,000 µs	
Period duration	100 µs	10 µs		
On-delay	0 µs			
Off-delay				
Frequency	0.02 Hz		10 kHz	100 kHz

High-speed output load current

Parallel mode (single channel operation)	Maximum pulse output load current	
	High-speed disabled	High-speed enabled
Disabled	2 A (Two channels)	100 mA (Two channels)
Enabled	4 A (One channel)	Not allowed

5.1.2.9 Function: Sequence counter (S7-1500)

The TM Pulse 2x24V has a sequence counter for each channel that counts completed output sequences. Successfully completed and unsuccessfully completed output sequences are counted.
You can monitor the completion of an output sequence with the sequence counter SEQ_CNT variable in the feedback interface.

The counter has a width of 4 bits. After a count overflow, the counter jumps back to 0.

Function of the sequence counter

The sequence counter has the following function in the individual operating modes:

- Pulse output and pulse train
  - Function DI configuration is set to "Input": the counter is set to 1 after completion of the output sequence (range 0 to 1).
  - Function DI configuration is set to "HW enable": the counter is incremented after every completed output sequence (range 0 to 15).
- On/Off-delay
  - The counter is incremented, with each edge (positive and negative) at the DQ output (range 0 to 15).
- PWM, frequency output, and DC motor
  - The counter does not have any function.

When the software enable (SW_ENABLE = 0) occurs, the counter is reset to 0.

Application options

The sequence counter can be used for:

- Detecting (counting) very short pulse sequences
- Counting of output sequences controlled with the hardware enable

5.1.2.10 Function: Current measurement (S7-1500)

Operating principle

Your program logic can use load current measurements with a control loop to proportionally control the energy transferred at an inductive or ohmic load. The current measurements are provided in the feedback interface MEASURED_CURRENT value in the SIMATIC S7 analog value format.

Current measurement is possible:

- In the PWM and pulse train operating modes and when the quick pulse output option is inactive
  - If there is no pulse output (for STS_ENABLE = 0), 7FFFH is output as the measured value.
  - The measured value is valid at (STS_ENABLE = 1) after the first period duration. The returned current measured value is a mean value of the measured values that were acquired for the duration of at least one period.
  - In case of active dithering the mean value is captured across the complete dither period.

Current measurement is not possible and returns the MEASURED_CURRENT value 0:

- For all other operating modes or when the fast pulse output option is active.

Note

So that current measurement functions properly, do not connect any freewheeling diodes (overload protection/suppressor diode) to the output load.

Measuring range and measured value

- **The channel configuration is 2-channel (2 A)** (parallel channel circuiting is deactivated).
  - A measured current mean of 2 A corresponds to the SIMATIC S7 analog value 27 648 ($6C00_H$).
  - Measurements of current are possible up to a SIMATIC S7 analog value of 32 511 ($7EFF_H$) and corresponds to a current of 2.37 A.
  - The current may exceed 2 A only briefly.
- **The channel configuration is 1-channel (4 A)** (parallel channel circuiting is activated).
  - A measured current mean of 4 A corresponds to the SIMATIC S7 analog value 27 648.
  - Measurements of current are possible up to a SIMATIC S7 analog value of 32 511, which corresponds to a current of 4.74 A.
  - The current may exceed 4 A only briefly.

Diagnostic interrupt when the output current limit is reached.

If the diagnostics was activated and the module is in the PWM or PTO operating mode, an overcurrent diagnostics error is reported when the module measures a current above 2.37 A (4.74 A in the one channel mode).

Accuracy of the current measurement

The accuracy during current measurement amounts to ±2% of the measurement range final value (27 648)

- ±40 mA in the two-channel operating mode
- ±80 mA in the single-channel operating mode

Above a pulse frequency of 3 kHz the currency measurement accuracy drops to an accuracy of -2% to +5% when ohmic loads are approached at a frequency of 10 kHz.

Note

An accuracy of 2% for the current measurement is only possible if the period duration is not changed during the measurement process.

5.1.2.11 Function: Current control (S7-1500)

Current control (PWM mode only)

In PWM mode, the TM Pulse 2x24V module can use a proportional-integral-derivative (PID) algorithm to control output current. The target value of the current (set point) is determined by your program and the module controls the duty cycle of the PWM output to follow the set point, with a response that is based on the assigned PID parameters.

The set point and the measured value of the load current are compared by the PID controller. If the resulting difference (error) is outside a symmetrical dead band, a reaction is calculated by adding the selected proportional, integral and/or derivative parts. The manipulated value is limited by the defined high/low limits and finally the output pulse duration time (duty cycle) is modified, depending on the PWM period duration.

Assign the reference current value to the current measured in the channel's DQn.A output load when the output is continuously in the high (On) state.

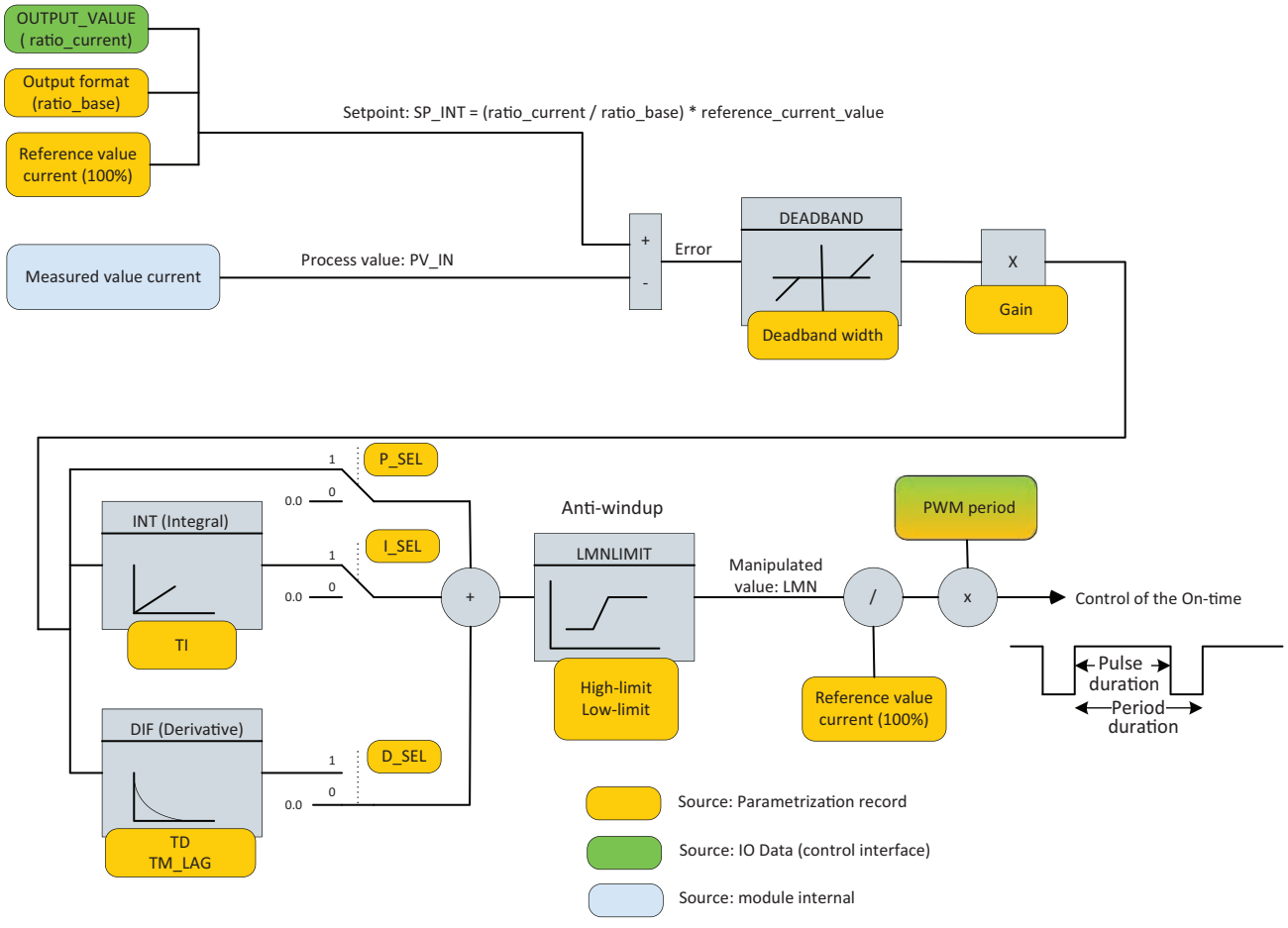

Figure 5-14 Current control PID function

For more information about this PID control method, refer to the CONT_C instruction topic in TIA Portal online help.

Setting up current control

In order to use the current control feature, the module must be parametrized correctly and the set point (target value of the current) must to be controlled by the user program. In addition, your program can also read the limit-reached flags.

PID parameters

The PID parameters can be set in TIA Portal device configuration or by your program writing parametrization record (record 128) in the module. See the parametrization record description for details.

- **Current Control**: "1" activates the PID controller. Limitations: Current control can only be activated if the selected mode of operation is PWM and the high-speed output is disabled. All the following parameters have no effect, if current control is disabled.
- **P_Sel, I_Sel, D_Sel**: defines the PID control calculation. The proportional, integral, and derivative parts can be enabled ("1") or disabled ("0") in the device configuration.
- **Reference value current (mA)**: The reference value is used to define the maximum set point and the high and low limits of the controlled current. Typically, the maximum current can be measured in PWM mode with current control disabled and a duty cycle set to 100%. The value measured can be set as a reference for the current control. The maximum value is 4000 mA for single channel operation (parallel channel connection enabled) and 2000 mA per channel for dual channel operation (parallel connection disabled).
- **Dead band width (µA)**: A dead band is applied to the output current deviation from the set point current. The "Dead band width" is half of the size of the dead band. The dead band is symmetrical: For example, if the dead band width = 1500 µA, then the dead band is from -1500 µA to +1500 µA .
- **High limit (S7 analog value)**: The manipulated value is always restricted to a high limit and low limit. The "High limit" parameter assigns the high limit in S7 analog format, relative to the reference value current. A value higher or equal than 27648 means 100% of the reference value current. The High limit value must be higher than the Low limit.
  High limit example:
  - The reference current is set to 1000 mA which should be the load current measured with 100% duty cycle.
  - If the high limit is 20000, this means the maximum current the controller will drive is 20000/27648 * 1000 mA = 723.4 mA that is equivalent to a duty cycle of 72.34%.
  - If the controller calculates a control value higher than 72.34%, it will drive 72.34% and never go higher than that value.

- **Low limit (S7 analog value):** The manipulated value is always restricted to a high limit and low limit. The "Low limit" parameter assigns the low limit in S7 analog format, relative to the reference value of the current. A value higher or equal to 27648 is not allowed. The Low limit value must be lower than the High limit.
  Low limit example:
  - The reference current is set to 1000 mA which should be the load current measured with 100% duty cycle.
  - If the low limit is 100, this means the minimum current the controller will drive is 100/27648 * 1000 mA = 3.6 mA that is equivalent to a duty cycle of 0.36%.
  - If the controller calculates a control value below 0.36%, it will drive 0.36% and never go lower than that value.
- **Gain:** The proportional gain parameter assigns the amplification factor for the P part of the PID algorithm.
- **TI (s)**: The integration time parameter determines the time interval of the integral action. If TI is smaller than the controller cycle time, TI will be set internally to the controller cycle time.
- **TD (s)**: The derivative time parameter determines the time interval of the derivative action. If TD is smaller than the controller cycle time, TI will be set internally to the controller cycle time.
- **TM_LAG (s)**: Time lag of the derivative action. The algorithm of the D-action contains a delay of TM_LAG. If TM_LAG is smaller than half the controller cycle time, TI will be set internally to half the controller cycle time.

Set point control

Your program controls the target value of the output current by setting the control interface field OUTPUT_VALUE. The output format is selected in the parametrization record (see parameter "output format"). For example, if the output format "per 100" is selected, writing the value 60 in OUTPUT_VALUE means the target current value is 60% of the reference current value (see parameter "reference value current").

Limit reached flags

The limit reached flags are available in the feedback interface.

QLMN_HLM: "1" means the manipulated value is at the high limit.

QLMN_LLM: "1" means the manipulated value is at the low limit.

For more information about the PID control parameters, refer to the CONT_C instruction topic in TIA portal online help.

Controller cycle time

The internal controller cycle time depends on the configured automation system and on the PWM period. The controller cannot control the current faster than the parametrized PWM period, as the measured current is averaged over an entire PWM period.

If Dithering is active simultaneously with current control, the PID controller uses the Dither period duration as the internal controller cycle time.

Resetting the PID current controller

The internal data of the current controller are reset in the following cases:

- SW_ENABLE (see control interface) is low.
- A new parametrization record is sent to the module.

5.1.2.12 Function: Dither PWM output (S7-1500)

Dither overview

The Dither function creates a vibration in a proportional valve when the desired valve position is controlled with current supplied from the PWM output. The vibration is induced by superimposing the dither current fluctuation around the target current, in a PWM output load (valve coil).

The dither vibration improves the accuracy and linearity of proportional control valves. You can enable and configure the Dither function to minimize valve control problems caused by static friction, stiction, and hysteresis.

Dither parameters

The dither parameters can be set in the TIA Portal device configuration or in the parameterization record (record 128) sent by the program to the module. See the Parameterization record description for details. Some dither parameters can be changed during operation by using the slot mechanism in the control interface for calibration purposes. It is recommended to restart the PWM output sequence after such a parameter change.

- Dither: "1" enables the dither feature. In addition to enabling "Dither" in the module's PWM configuration, you must set the DITHER bit in the Control interface to start the Dither signal.
- Dither amplitude (‰): Assign the amplitude ratio of the superimposed dither signal in a per mill ratio. The allowed range is 0 to 500‰ duty cycle. If a higher value is assigned, then 500‰ is used by the module.
  For example: if the dither amplitude is set to 100‰ and the duty cycle in PWM mode is set to 50%, then the effective duty cycle of the signal will vary periodically between 40% and 60%.
  The dither amplitude is adapted (reduced) dynamically by the module if the calculated effective duty cycle is higher than 100% or lower than 0%, so that the dither signal remains symmetrical.
  For example, if the dither amplitude is set to 100‰ and the duty cycle in PWM mode is set to 95%, the effective duty cycle of the signal will vary periodically between 90% and 100%. The dither amplitude is always corrected to remain symmetrical. The dither amplitude will return to the assigned value, as soon as the output signal allows enough duty cycle margin for the superimposed Dither duty cycle variation to be symmetrical.

- **Dither period (µs)**: Assign the period duration in microseconds for the superimposed dither signal. The allowed range is from (4 x PWM period) to 100,000 µs. Also, the Dither period must be greater than 2 ms. If a value less than 2 ms or less than (4 x PWM period) is parametrized, then an error occurs (see parameter validation for ERR_SLOT_VAL and ERR_LD). If a Dither period higher than 100 ms is assigned, the module will use the value 100 ms. The Dither period used by the module can only be an even multiple of the PWM period. The module will use the nearest possible value to the assigned Dither period.
- **Dither ramps**: The dither ramp parameter is one double-word made of two different words. The low word is the dither ramp-up time and the high word is the dither ramp-down time.
  - **Dither ramp-up time (ms):** Assign the time in milliseconds for the duty cycle to rise from 0% to 100%. The allowed range is 0 ms to 30,000 ms. If a higher value is assigned, then 30,000 ms is used by the module.
    The effective ramp-up time is dependent on the dither amplitude and the nominal value of the duty cycle.
    For example: if your nominal value of the duty cycle is 50% and the dither amplitude is 100‰, the effective duty cycle will vary between 40% and 60%, the ramp-up time from 50% to 60% will be 100‰ x Dither ramp-up time.
  - **Dither ramp-down time (ms):** Assign the time in milliseconds for the duty cycle to fall from 100% to 0%. The allowed range is 0 ms to 30,000 ms. If a higher value is assigned, then 30,000 ms is used by the module.
    The effective ramp-down time is dependent on the Dither amplitude and the nominal value of the duty cycle.
    For example: if your nominal value of the duty cycle is 50% and the dither amplitude is 100‰, the effective duty cycle will vary between 40% and 60%, the ramp-down time from 50% to 40% will be 100‰ x Dither ramp-down time.

Note

Changing parameters during Dither output

If the PWM period is changed during operation using the slot mechanism, so that the dither period is less than (4 x PWM period), then dither current is deactivated and feedback bit STS_DITHER = 0, until a valid combination of Dither and PWM periods is assigned.

If you change parameter values using the SLOT mechanism during dither output, then you should restart the output sequence.

Note

PWM with dithering in isochronous mode

For best results, set the dither period duration to an integral even-numbered multiple of the application cycle.

Example: If the application cycle is 1ms, then the minimal optimal dither period is 2ms. The next optimal values are 4ms, 6ms, ...

Dither starting and stopping

The ramp-up of the dither current starts as soon as the bit DITHER is set in the control interface and the current Dither period has ended and a new Dither period starts. The feedback interface provides an acknowledge bit STS_DITHER in the feedback interface that goes high when the ramp-up phase is started (by setting the DITHER bit when the output sequence is running) and goes low when the ramp-down phase is over or the output sequence is stopped. The ramp-down starts when the bit DITHER is cleared in the control interface and the current Dither period has ended and a new Dither period starts.

Example 1: No dither (Dither bit = 0)

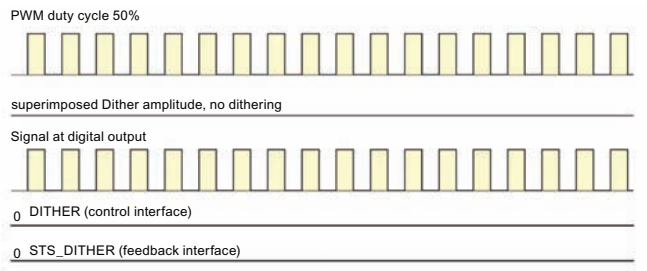

Example 2: Dither with no ramp (Dither period = 6 x PWM period)

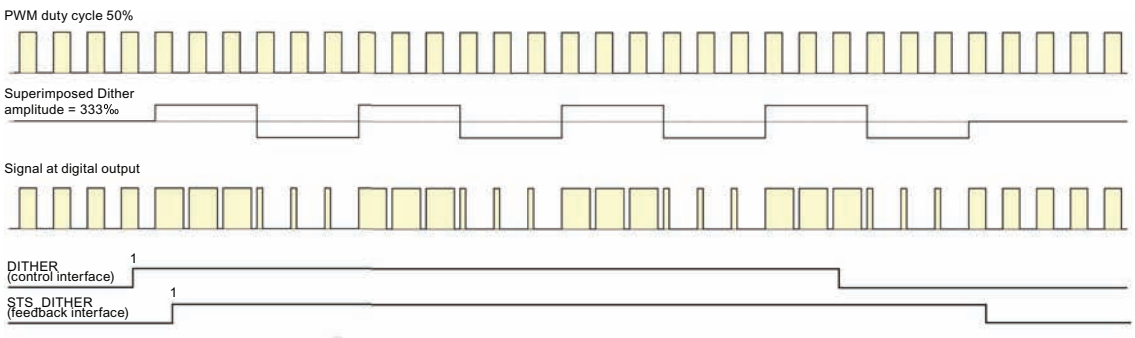

Example 3: Dither with ramp-up

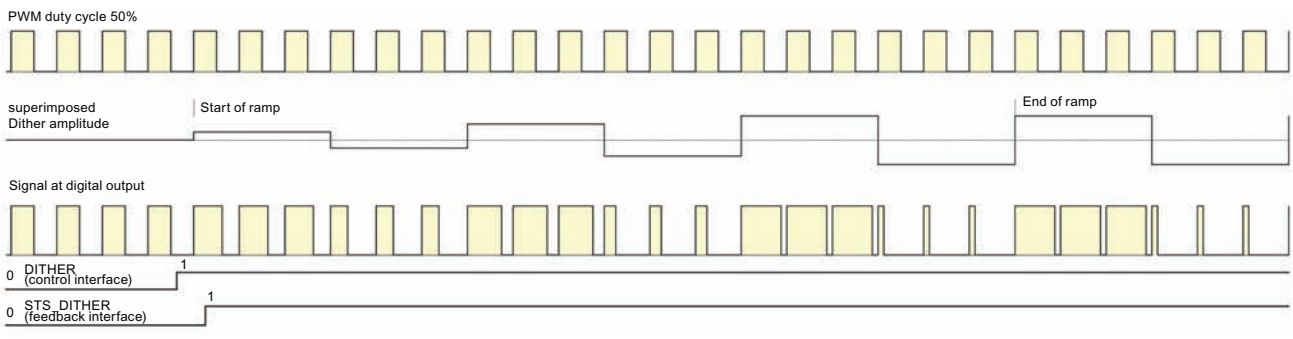

Example 4: Dither with ramp-down

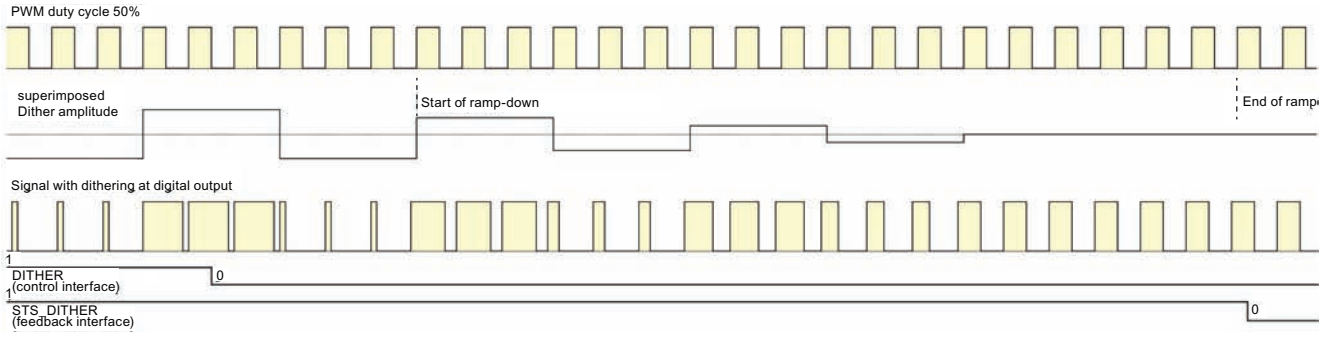

5.1.2.13 Function: Isochronous mode (S7-1500)

Note

For basic information on isochronous mode, refer to the SIMATIC PROFINET with STEP 7 manual.

Requirements

You will require the following for the TM Pulse 2x24V in isochronous mode:

- A CPU that supports isochronous mode
- An IM (interface module) that supports isochronous mode
- Engineering software, such as TIA portal or STEP 7, to parameterize isochronous mode

Response of the TM Pulse 2x24V

Depending on the system parameter assignment, the TM Pulse 2x24V works in either non-isochronous or isochronous mode.

In isochronous mode

- The output sequences are started at the moment T_O, when only the software enable is used.
- Data communication between the PROFINET controller and TM Pulse 2x24V is isochronous to the cycle.
- All 12 bytes of a channel's control interface are consistent (24 bytes for both channels).
- All 8 bytes of a channel's feedback interface are consistent (16 bytes for both channels).
- In the PWM operating mode, the period duration is synchronized to the application cycle time (PROFINET). See the PWM chapter for details.

5.1.2.14 Function: Direct control of DQ digital outputs (S7-1500)

Definition

You can directly set the TM Pulse 2x24V DQ digital outputs with your control program. Select the DQ direct control function, by clearing the Technology Module output control bit (TM_CTRL_DQ = 0), in the control interface.

Direct control of a digital output can support you when commissioning an automation control system.

If you select direct control of the DQ during a pulse output sequence, the sequence will continue to run in the background, so that when the module gets control again (by setting TM_CTRL_DQ = 1), the output sequence continues.

You assign the state of a digital output DQn.A and DQn.B with the SET_DQA and SET_DQB control bits.

You cannot use the Direct control function to set both DQn.A and DQn.B outputs high on the same channel. If an attempt to do so occurs, then error ERR_DQB is set in the feedback interface and only the DQn.A output is set high.

When you set TM_CTRL_DQ = 1, you deselect the direct control of digital output function. If the output sequence is still running (STS_ENABLE still active), then the TM Pulse 2x24V module regains control of a channel's DQn.A and DQn.B outputs.

Note

TM_CTRL_DQ technology module output control signal

- If TM_CTRL_DQ = 1, then the TM Pulse 2x24V module has control and produces pulse sequences at the DQn.A and DQn.B outputs. DQn.B is always 0 except in DC motor mode.
- If TM_CTRL_DQ = 0, then the CPU has control and your program can set DQ outputs directly with the SET_DQA/SET_DQB control bits. In DC motor mode, it is not possible to control the DQn.A and DQn.B outputs manually; TM_CTRL_DQ, SET_DQA, and SET_DQB have no effect.

Pulse diagram

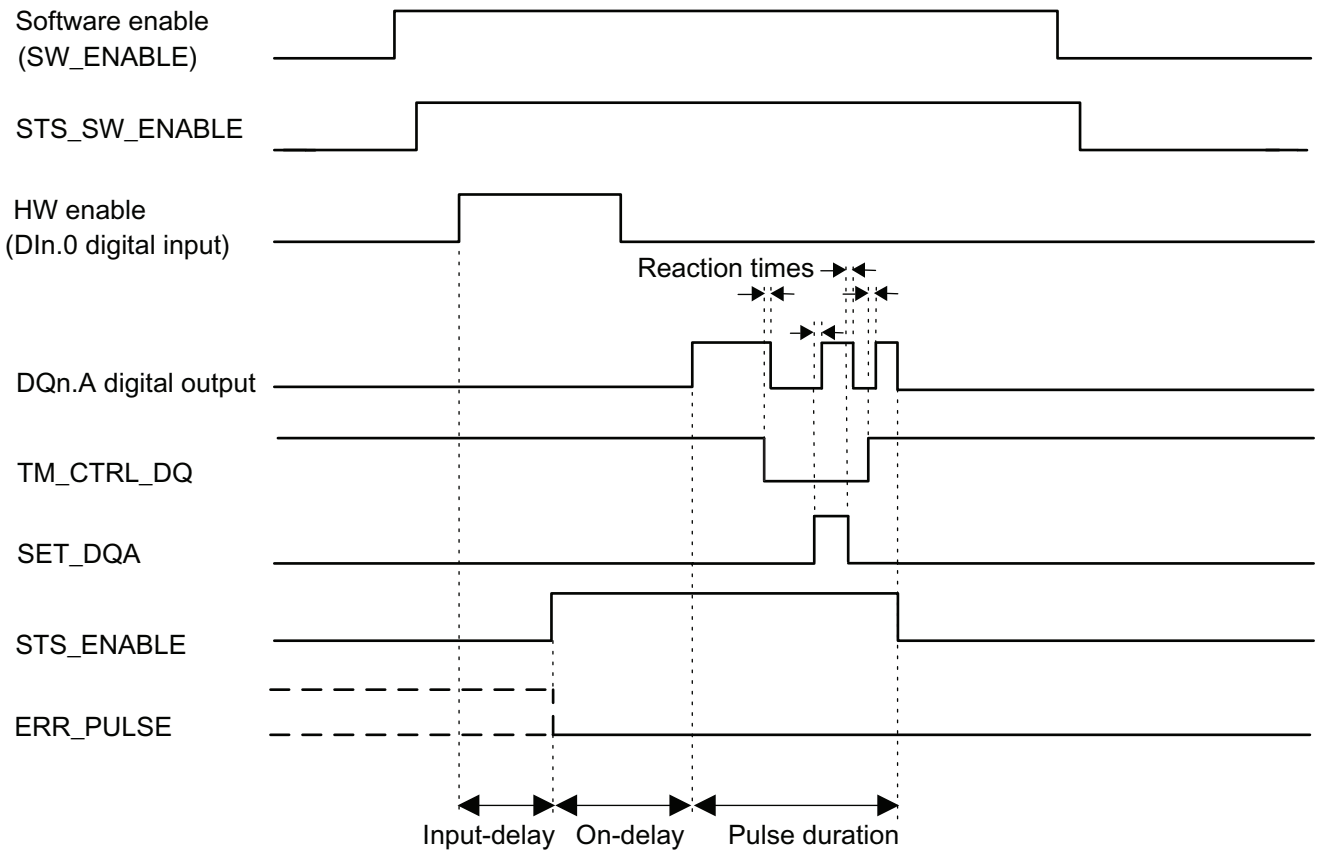

Figure 5-15 Direct control of DQ timing during Pulse output mode

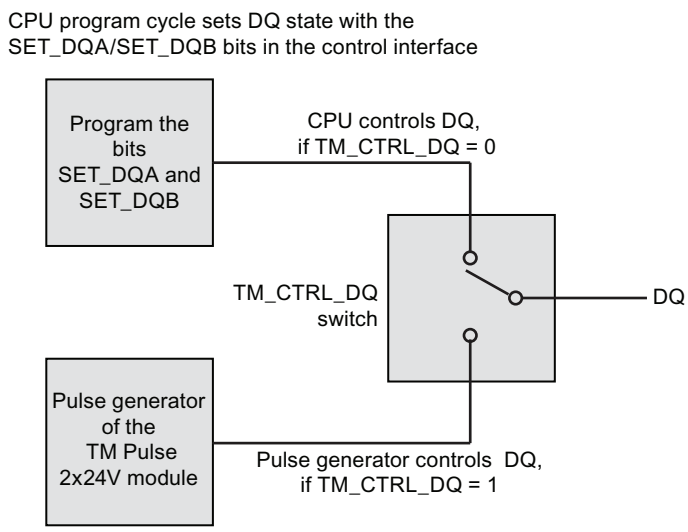

Figure 5-16 TM_CTRL_DQ output switch

Control and feedback signals

Control interface Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1			
Byte 9: Bit 0	Byte 21: Bit 0	SW_ENABLE	**Bit 0**	Software enable: Start and terminate the output sequence.
			0	Output canceled.
			0 → 1 1	Starts output sequence on positive edge; may be dependent on the hardware enable.
Byte 9: Bit 1	Byte 21: Bit 1	TM_CTRL_DQ	**Bit 1**	Set DQ output source: Selects either PLC program or module's output sequence.
			0	DQn.A and DQn.B are controlled by the PLC program using SET_DQA and SET_DQB.
			1	DQn.A is controlled by the module's pulse output sequence.
Byte 9: Bit 3	Byte 21: Bit 3	SET_DQA	**Bit 3**	Controls the value of the digital output DQn.A, if TM_CTRL_DQ is cleared.
			0	0V on DQn.A
			1	24V on DQn.A
Byte 9: Bit 4	Byte 21: Bit 4	SET_DQB	**Bit 4**	Controls the value of the digital output DQn.B, if TM_CTRL_DQ is cleared and if SET_DQA is cleared.
			0	0V on DQn.B
			1	24V on DQn.B

Feedback interface: Offset to the start address		Parameter	Meaning	
Channel 0	Channel 1			
Byte 0: Bit 4	Byte 8: Bit 4	ERR_DQA	**Bit 4**	Indicates a short-circuit on the output DQn.A. You must set RES_ERROR (control interface) to reset this error.
			0	No short- circuit on DQn.A
			1	Short-circuit on DQn.A
Byte 0: Bit 5	Byte 8: Bit 5	ERR_DQB	**Bit 5**	Indicates a short-circuit on the output DQn.B or an attempt to set both DQn.A and DQn.B manually using SET_DQA,SET_DB, and TM_CTRL_DQ. You must set RES_ERROR (control interface) to reset this error.
			0	No short-circuit on DQn.B
			1	Short-circuit on DQn.B or attempt to set both DQn.A and DQn.B
Byte 2: Bit 1	Byte 10: Bit 1	STS_DQA	**Bit 1**	Indicates the signal level at the DQn.A digital output.
			0	Signal 0 at the DQn.A digital output
			1	Signal 1 at the DQn.A digital output
Byte 2: Bit 2	Byte 10: Bit 2	STS_DQB	**Bit 2**	Indicates the signal level at the DQn.B digital output.
			0	Signal 0 at the DQn.B digital output
			1	Signal 1 at the DQn.B digital output

States of DQ bits				
TM_CTRL_DQ	SET_DQA	SET_DQB	Reaction at DQn.A	Reaction at DQn.B
0	0	0	0	0
0	1	0	1	0
0	1	1	1	0 (ERR_DQB is set)

States of DQ bits				
TM_CTRL_DQ	SET_DQA	SET_DQB	Reaction at DQn.A	Reaction at DQn.B
0	0	1	0	1
1	Don't care	Don't care	State controlled by pulse processing	State controlled by pulse processing

Note

Setting both DQn.A and DQn.B to the 1 state not allowed

You cannot set both DQn.A and DQn.B to 1 at the same time. If SET_DQA and SET_DQB are set high and TM_CTRL_DQ is low, only output DQn.A will go high and an error bit ERR_DQB is set in the feedback interface. This error must be acknowledged in the control interface by using the RES_ERROR bit.

5.1.3 Basic principles of pulse output (Compact CPU) (S7-1500)

5.1.3.1 Operating modes (S7-1500)

Overview (S7-1500)

This section lists the new features of the CPU with firmware version V2.0.

Operating mode	Applications	Customer benefits
Pulse width modulation (Page 1257) (PWM)	The PWM mode is used when an output module is to control greatest possible outputs with low power loss (heating, size). You use pulse width modulation, for example, to control: • the temperature in a heating resistor • the force of a coil in a proportional valve and thus the position of a valve from closed to completely open • the speed of a motor from standstill to full speed	With pulse width modulation, a signal with a defined period duration and variable duty cycle is output at the digital output. The duty cycle is the ratio of pulse duration to period duration. In PWM mode, you can control both the duty cycle and the period duration. With pulse width modulation, you vary the mean value of the output voltage. Depending on the connected load, you can control the load current or the power with this.
Frequency output (Page 1261)	You can implement frequencies up to 100 kHz and thus work in ranges that cannot be reached by a CPU with a simple digital output.	You can generate frequencies very precisely. The receiver can reconstruct the information exactly when transmission conditions are less than ideal. In frequency output mode, you assign a frequency value with high frequencies more precisely than by using period duration (PWM).
Pulse Train Output (PTO) (Page 1264)	Pulse Train Output is a widely used interface for drive control. It is used in many positioning applications, such as for retooling axes and feed axes.	PTO is also referred to as pulse/direction interface. The pulse/direction interface comprises two signals. The frequency of the pulse output represents the speed and the number of output pulses for the route to be traversed. The direction output defines the traversing direction. The position is thus specified precisely for the specific increment. The outputs are controlled with S7-1500 Motion Control using technology objects. PTO is a simple and universal interface between control system and drive. As a result, it is supported worldwide by many stepper and servo drives.

Operating mode: Pulse-width modulation (PWM) (S7-1500)

Properties

The pulse-width modulation (PWM) mode of the compact CPU has the following technical properties:

	Minimum			Maximum		
	Standard output	High-speed output deactivated	High-speed output activated	Standard output	High-speed output deactivated	High-speed output activated
Pulse duration	100 µs with load > 0.1 A [1] 200 µs with load ≥ 2 mA [1]	20 µs with load > 0.1 A [1] 40 µs with load ≥ 2 mA [1]	2 µs [1]	10,000,000 µs (10 s)		
Period duration	10 ms [2]	100 µs [2]	10 µs			

[1] A lower value is theoretically possible but, depending on the connected load, the output voltage can no longer be output as complete rectangular pulse

[2] Load dependent

Principle of operation

With pulse width modulation, a signal with a defined period duration and variable duty cycle is output at the digital output. The duty cycle is the ratio of pulse duration to period duration. In PWM mode, you can control both the duty cycle and the period duration.

With pulse width modulation you vary the mean value of the output voltage. Depending on the connected load, you can control the load current or the power with this.

You can specify the pulse duration as one-hundredth of the period duration (0 bis 100), as one-thousandth (0 to 1000), as one ten-thousandth (0 to 10,000) or in S7 analog format.

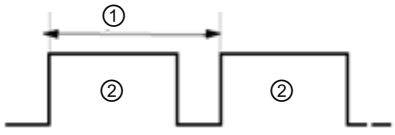

① Period duration

② Pulse duration

The pulse duration can be between 0 (no pulse, always off) and full-scale deflection (no pulse, period duration always on).

The PWM output can, for example, be used to control the speed of a motor from standstill to full speed or you can use it to control the position of a valve from closed to completely open.

Four pulse generators for PWM and Pulse Train Output (PTO) respectively are available to control high-speed pulse outputs (high-speed output). PTO is used by the technology objects positioning axis, speed axis or synchronized axis. You can assign each pulse generator to either PWM or PTO, but not to both at the same time.

You configure the pulse width modulation (PWM) mode in STEP 7 (TIA Portal).

The pulse width modulation mode has the following functions:

- If the "High-speed output (0.1 A)" option is activated, you can generate a minimum pulse duration of 2 µs at a current of 100 mA. If the "High-speed output (0.1 A)" option is not activated, you can generate a minimum pulse duration of 20 µs at a load > 0.1 A and a minimum pulse duration of 40 µs at a load of ≥ 2mA and a current of maximum 0.5 A. If you use a standard output you can generate a minimum pulse duration of 100 µs at a load > 0.1 A and a minimum pulse duration of 200 µs at a load of ≥ 2mA.
- You can control the pulse output (DQA) of the channel manually via the control and feedback interface.
- You can configure the reaction to CPU STOP. Upon change to CPU STOP, the pulse output (DQA) is set to the configured state.

Controller

For the pulse width modulation (PWM) mode, the user program directly accesses the control and feedback interface of the channel.

A reconfiguration via the instructions WRREC/RDREC and parameter assignment data record 128 is supported. You can find additional information on this in the Parameter data records (PWM) section of the respective product manner.

You control the on-load factor of the pulse width via the OUTPUT_VALUE field of the control interface. Pulse width modulation generates continuous pulses based on this value. The period duration is adjustable.

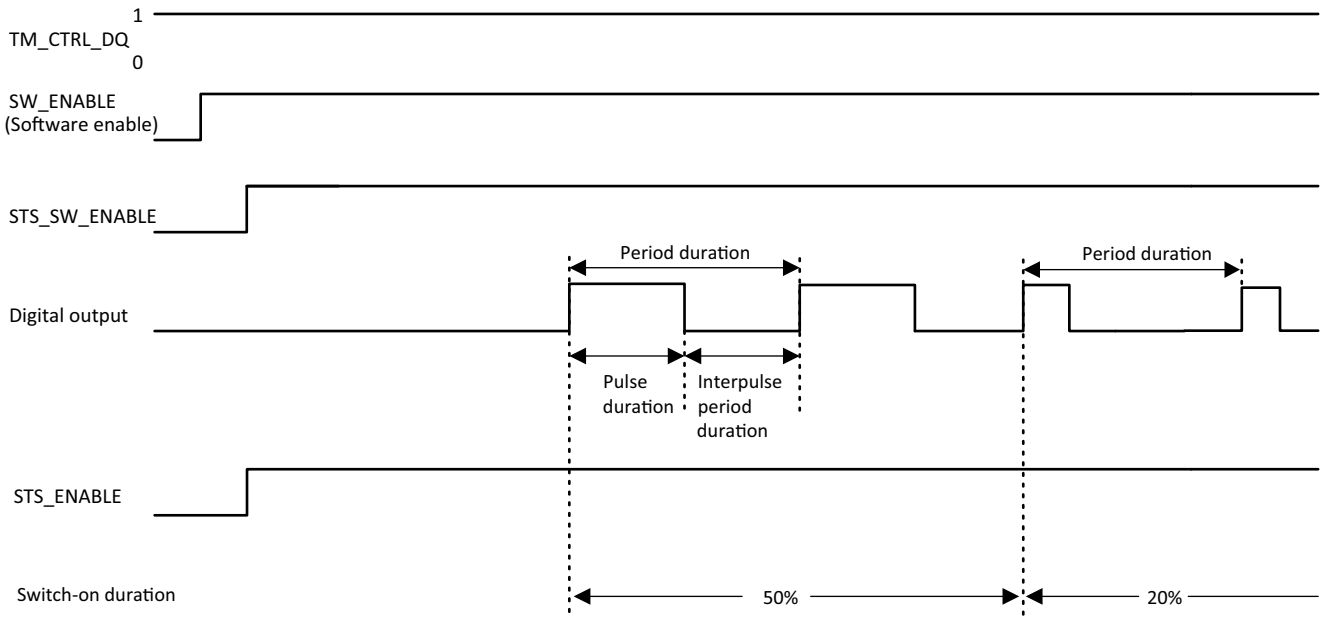

Starting the output sequence

The control program must output the enable for the output sequence with the help of the software enable (SW_ENABLE 0 → 1). The feedback bit STS_SW_ENABLE indicates that the software enable is pending at the PWM.

If the software enable is activated (positive edge), STS_ENABLE is set. The output sequence runs continuously, as long as SW_ENABLE is set.

Note

Output control signal TM_CTRL_DQ

- If TM_CTRL_DQ = 1, the technology function takes over the control and generates pulse sequences at the output PWM DQA.
- If TM_CTRL_DQ = 0, the application program takes over the control and the user can set the output PWM DQA directly with the control bit SET_DQA.

Canceling the output sequence

A deactivation of the software enable (SW_ENABLE = 1 → 0) cancels the current output sequence. The last period duration is not completed. STS_ENABLE and the digital output PWM DQA are immediately set to 0.

A renewed pulse output is only possible after a restart of the output sequence.

Minimum pulse duration and minimum interpulse period

You assign the minimum pulse duration and the minimum interpulse period with the help of the parameter "Minimum pulse duration".

- A pulse duration, determined by the technology function or the PWM channel, which is shorter than the minimum pulse duration will be suppressed.
- A pulse duration, determined by the technology function or the PWM channel, which is longer than the period duration minus the minimum pulse duration will be set to the value of the period duration (output continuously enabled).

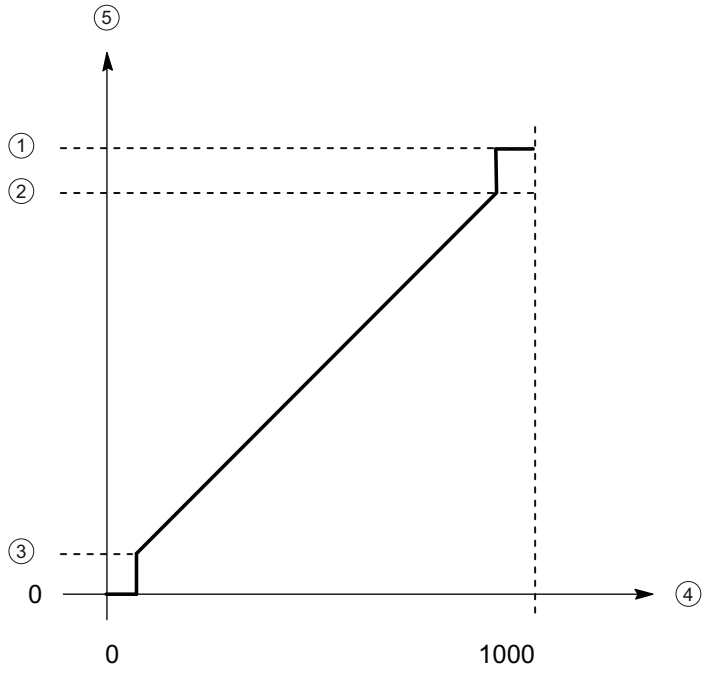

① Period duration
② Period duration minus minimum interpulse period
③ Minimum pulse duration
③ OUTPUT_VALUE (one tenth of one percent duty cycle)
③ Pulse duration

Setting and changing the pulse on-load factor

OUTPUT_VALUE assigns the on-load factor for the current period duration. You select the range of the field OUTPUT_VALUE of the control interface with the "Output format" parameter.

- Output format per 100: Value range between 0 and 100
  Pulse duration = (OUTPUT_VALUE/100) x period duration.
- Output format 1/1000: Value range between 0 and 1,000
  Pulse duration = (OUTPUT_VALUE/1,000) x period duration.
- Output format 1/10000: Value range between 0 and 10,000
  Pulse duration = (OUTPUT_VALUE/10,000) x period duration.
- Output format "S7 analog output": Value range between 0 and 27,648
  Pulse duration = (OUTPUT_VALUE/27 648) x period duration.

You assign OUTPUT_VALUE directly via the control program. A new OUTPUT_VALUE is applied at the output when the next positive edge occurs.

Setting and changing the period duration

- Permanent updating
  The period duration is permanently controlled via the control interface. The MODE_SLOT bit must be set ("1" means permanent updating); LD_SLOT must be set to value 1 ("1" means period duration). Set the period value in the field SLOT. The unit is always a microsecond.
  - High-speed output activated: Between 10 µs and 10,000,000 µs (10 s) in the field SLOT
  - High-speed output deactivated: Between 100 µs and 10,000,000 µs (10 s) in the field SLOT
  - Standard output (100 Hz output): Between 10,000 µs (10 ms) and 10,000,000 µs (10 s) in the field SLOT
- Individual updating
  Set the period duration in the configuration parameters. Alternatively, execute an individual updating via the control interface. MODE_SLOT must be deleted ("0" means individual updating); LD_SLOT must be set to value 1 ("1" means period duration). Set the period duration value in the field SLOT. The unit is always a microsecond.
  - High-speed output activated: Between 10 µs and 10,000,000 µs (10 s) in the parameters
  - High-speed output deactivated: Between 100 µs and 10,000,000 µs (10 s) in the parameters
  - Standard output (100 Hz output): Between 10,000 µs (10 ms) and 10,000,000 µs (10 s) in the parameters

The new period duration is applied at the next positive edge of the output.

Setting the minimum pulse duration and the minimum interpulse period

You assign the minimum pulse duration and the minimum pulse pause as DWord numerical value between 0 and 10,000,000 µs (10 s) using the channel parameter configuration "Minimum pulse duration".

Output signals for pulse width modulation (PWM) mode

Output signal	Meaning	Value range
Continuous pulse current at digital output PWM DQA	A pulse is output at the digital output PWM DQA for the set duty cycle and period duration.	Continuous pulse current

Operating mode: Frequency output (S7-1500)

In this operating mode you can assign a frequency value with high frequencies more precisely than by using period duration in PWM mode.

A rectangular signal with an assigned frequency and a constant on-load factor of 50 % is generated at the digital output.

The frequency output mode has the following functions:

- If the "High-speed output (0.1 A)" option is activated, you can generate a minimum pulse duration of 2 µs at a current of 100 mA. If the "High-speed output (0.1 A)" option is not activated, you can generate a minimum pulse duration of 20 µs at a load of > 0.1 A and a minimum pulse duration of 40 µs at a load of ≥ 2mA and a current of maximum 0.5 A.
  If you use a standard output you can generate a minimum pulse duration of 100 µs at a load > 0.1 A and a minimum pulse duration of 200 µs at a load of ≥ 2mA and current of maximum 0.5 A.

	Minimum			Maximum		
	Standard output	High-speed output deactivated	High-speed output activated	Standard output	High-speed output deactivated	High-speed output activated
Frequency	0.1 Hz			100 Hz [1]	10 kHz [1]	100 kHz

[1] Load dependent

- You can control the pulse output (DQA) of the channel manually via the control and feedback interface.
- You can configure the reaction to CPU STOP. Upon change to CPU STOP, the pulse output (DQA) is set to the configured state.

Controller

For the frequency output mode, the user program directly accesses the control and feedback interface of the channel.

A reconfiguration via the instructions WRREC/RDREC and parameter assignment data record 128 is supported. You can find additional information on this in the section "Parameter data records (PWM)" for the respective compact CPU.

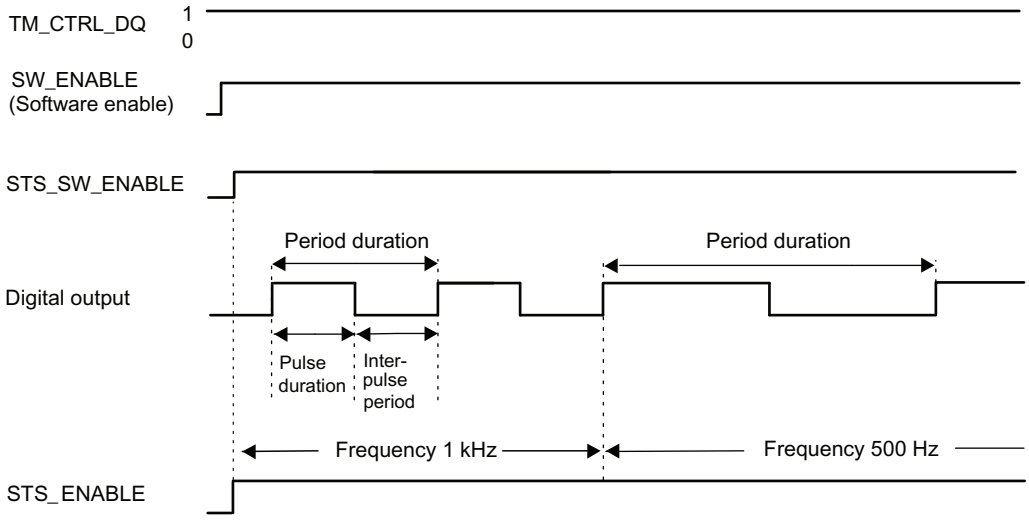

Starting the output sequence

The control program must initiate the enable for the output sequence with the help of the software enable (SW_ENABLE 0 → 1). The feedback bit STS_SW_ENABLE indicates that the software enable is pending at the pulse generator.

If the software enable is activated (positive edge), STS_ENABLE is set. The output sequence runs continuously, as long as SW_ENABLE is set.

Note

Output control signal TM_CTRL_DQ

- If TM_CTRL_DQ = 1, the technology function takes over the control and generates pulse sequences at the output PWM DQA.
- If TM_CTRL_DQ = 0, the application program takes over the control and the user can set the output PWM DQA directly with the control bit SET_DQA.

Canceling the output sequence

A deactivation of the software enable (SW_ENABLE = 1 → 0) during the frequency output cancels the current output sequence. The last period duration is not completed. STS_ENABLE and the digital output PWM DQA are immediately set to 0.

A renewed pulse output is only possible after a restart of the output sequence.

Setting and changing the output value (frequency)

You set the frequency with the OUTPUT_VALUE directly with the control program in the control interface. The value is specified in the real format and the unit is always "Hz". The possible range depends on the parameter "High-speed output (0.1 A)" as follows:

- High-speed pulse output deactivated
  - Frequency (OUTPUT_VALUE) 0.1 Hz to 10,000 Hz
- High-speed pulse output activated
  - Frequency (OUTPUT_VALUE) 0.1 Hz to 100,000 Hz
- Standard output (100 Hz output)
  - Frequency (OUTPUT_VALUE): 0.1 Hz to 100,000 Hz

The new frequency is applied at the start of the next period. The new frequency has no effect on the negative edge or the pulse-pause ratio (duty cycle). However, the application can take up to 10 s depending on the previously set frequency.

Accuracy of the output frequency

The configured output frequency is out with a frequency dependent accuracy at the digital output PWM DQA. An overview of the accuracy as a function of the used frequency can be found in section "Interconnection overview of outputs" in the product manual of the respective compact CPU.

Output signals for frequency output mode

Output signal	Meaning	Value range
Continuous pulse current at digital output PWM DQA	A pulse is output at the digital output PWM DQA for the assigned frequency.	Continuous pulse current

Operating mode: PTO (S7-1500)

Controller

For the four operating modes of the pulse generators (PTO), the pulse output channels are controlled by means of Motion Control using the technology objects TO_SpeedAxis, TO_PositioningAxis and TO_SynchronousAxis. With these operating modes the control and feedback interface of the channels is a partial implementation of the PROFIdrive interface with the standard telegram 3.

A detailed description of the use of Motion Control and its configuration can be found in the online help for Motion Control (Page 1775) and in the S7-1500 Motion Control function manual (http://support.automation.siemens.com/WW/view/en/59381279).

Signal types

PTO mode is divided into the following four signal types:

- PTO (pulse (A) and direction (B)):
  an output (A) controls the pulses and an output (B) controls the direction. B is 'High' (active) if pulses are generated in negative direction. B is 'Low' (inactive) if pulses are generated in positive direction.

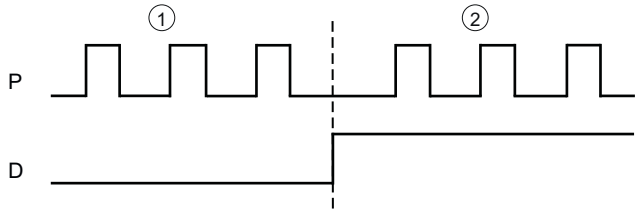

① Positive direction of rotation
② Negative direction of rotation

- PTO (Count up (A) and count down (B)):
  One output (A) outputs pulses for positive directions and another output (B) outputs pulses for negative directions.

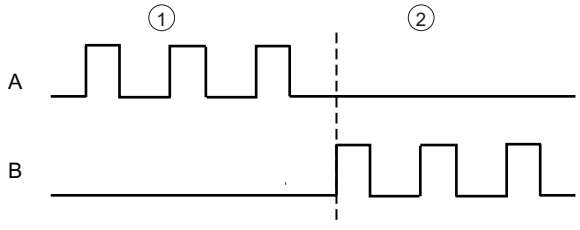

① Positive direction of rotation
② Negative direction of rotation

- PTO (A, B phase-shifted):
  Output pulses are output by both outputs at the specified velocity, but phase-shifted by 90 degrees. This involves a single pulse output in which the duration of the pulse is the time between two transitions of signal A while signal B is in low state.
  A positive direction of rotation is generated at a rising edge of signal A while signal B is in low state. A negative direction of rotation is generated at a falling edge of signal A while signal B is in low state.

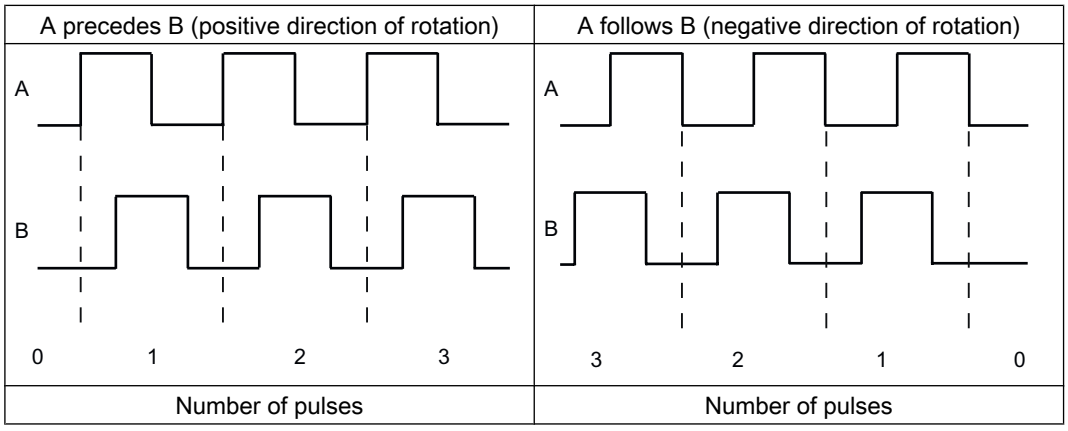

- PTO (A, B phase-shifted, quadruple):
  Output pulses are output by both outputs at the specified velocity, but phase-shifted by 90 degrees. This signal type involves a quadruple pulse output in which each edge transition corresponds to one increment. Therefore, a full period of the signal A contains four increments. In this way, it is possible, for example, to use two outputs, each with 100 kHz signal frequency, to output a control signal that supplies 400,000 increments per second. Whether count pulses are generated in the positive or negative direction of rotation depends on the edge direction of one signal and the logic state of the other signal in each case.

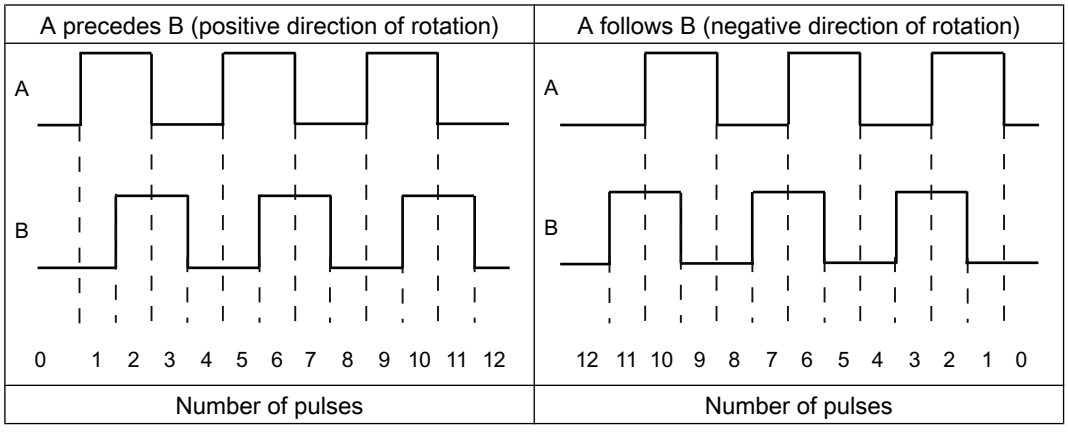

Reaction of PTO channel to a CPU STOP

The PTO channel reacts to a change to CPU STOP by canceling the drive enable (if a drive enable output is configured) and with output of the velocity setpoint 0 at the hardware outputs configured for the signal tracks A and B. The reaction of the PTO channels to a CPU STOP is not configurable.

Note

Reaction to CPU STOP

Following a CPU STOP, the hardware outputs assigned for the PTO outputs A and B can switch to signal state 'High' (1) and/or remain there. A switching/remaining of the two hardware outputs to/in signal level 'Low' (0) is not guaranteed.

5.1.3.2 Functions (S7-1500)

Function: High-speed output (S7-1500)

The function "High-speed output (0.1 A)" enhances the signal clock of the digital outputs (DQ0 to DQ7). Less delay, fluctuation, jitter, and shorter rise and fall times, occur at the switching edges.

The function "High-speed output (0.1 A)" is suited for generating pulse signals in a more precise clock, but provides a lower maximum load current.

For the operating modes PWM and frequency output, select the high-speed output of the channel in STEP 7 (TIA Portal). You can also change the parameter assignment during runtime with the help of the program via the data record.

High-speed pulse output (high-speed output) is available for the following operating modes:

- PWM
- Frequency output
- PTO (the pulse output for the PTO mode are always "High-speed output (0.1 A)")

High-speed output

	Minimum		Maximum	
	High-speed output deactivated	High-speed output activated	High-speed output deactivated	High-speed output activated
Pulse duration	20 µs with load > 0.1 A [1] 40 µs with load ≥ 2 mA [1]	2 µs [1]	10,000,000 µs (10 s)	
Period duration	100 µs [2]	10 µs		
Frequency	0.1 Hz		10 kHz [2]	100 kHz

[1] A lower value is theoretically possible but, depending on the connected load, the output voltage can no longer be output as complete rectangular pulse

[2] Load dependent

Function: Direct control of the pulse output (DQA) (S7-1500)

Direct control of the pulse output (DQA)

In the modes "Pulse width modulation PWM" and "Frequency output", you an set the pulse output (DQA) of a pulse generator directly via the control program. Select the function for the DQ direct control by resetting the output control bit of the PWM channel in the control interface (TM_CTRL_DQ = 0).

The direct control of the pulse output (DQA) can be helpful when commissioning a control system for automation.

When you select the direct control of the pulse output (DQA) during a pulse output sequence, the sequence continues to run in the background so that the output sequence is continued as soon as the channel takes control again (by setting TM_CTRL_DQ = 1).

You assign the status of the pulse output (DQA) with the control bit SET_DQA.

When you set TM_CTRL_DQ = 1, you deselect the direct control of the pulse output (DQA) and the channel takes over the processing. If the output sequence is still running (STS_ENABLE still active), the PWM channel takes over the control of the output again. If TM_CTRL_DQ = 1and STS_ENABLE is not active, the channel of the module also takes over the processing, but then outputs "0" at the output.

Note

Output signal TM_CTRL_DQ of the PWM channel

- If TM_CTRL_DQ = 1, the technology function takes over the control and generates pulse sequences at the output PWM DQA.
- If TM_CTRL_DQ = 0, the user program takes over the control and the user can set the PWM DQA directly with the control bit SET_DQA.

5.1.4 Basic principles of pulse output (TM PTO 4) (S7-1500)

5.1.4.1 Overview (S7-1500)

The technology module offers the following functions:

Operating mode	Applications	Customer benefits
Pulse Train Output (PTO) (Page 1268)	Pulse Train Output is a widely used interface for drive control. It is used in many positioning applications, such as for retooling axes and feed axes.	PTO is also referred to as pulse/direction interface. The pulse/direction interface comprises two signals. The frequency of the pulse output represents the speed and the number of output pulses for the route to be traversed. The direction output defines the traversing direction. The position is thus specified precisely for the specific increment. The outputs are controlled with S7-1500 Motion Control using technology objects. PTO is a simple and universal interface between control system and drive. As a result, it is supported worldwide by many stepper and servo drives.

5.1.4.2 Pulse Train Output (PTO) (S7-1500)

Control

The control of the pulse output channels is provided above all with S7-1500 Motion Control by means of the technology objects TO_SpeedAxis, TO_PositioningAxis and TO_SynchronousAxis . The control and feedback interface of the channels is an implementation of the PROFIdrive interface with standard frame 3.

For a detailed description of configuring the technology module with the axis technology objects, see Function Manual S7-1500T Motion Control, section "Configuring", which is available for download on theInternet (https://support.industry.siemens.com/cs/ww/en/view/109481326).

Signal types

The technology module supports the following four signal types:

- Pulse (P) and direction (D):
  One output (P) controls the pulses and one output (D) controls the direction. D is 'high' (active) when pulses are generated in the negative direction. D is 'low' (inactive) when pulses are generated in the positive direction.

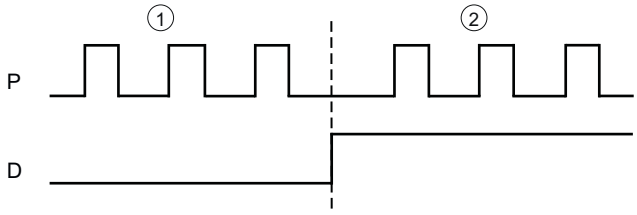

① Positive direction of rotation
② Negative direction of rotation

- Count up (A), count down (B):
  One output (A) outputs pulses for positive directions and another output (B) outputs pulses for negative directions.

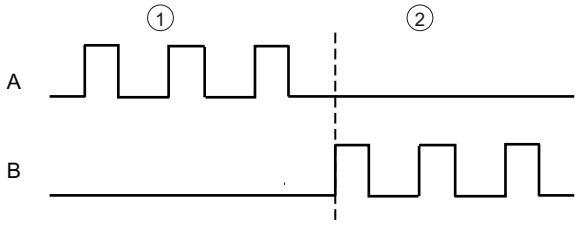

① Positive direction of rotation
② Negative direction of rotation

- Incremental encoder (A, B phase-shifted):
  Output pulses are output by both outputs at the specified velocity, but phase-shifted by 90 degrees. This involves a single pulse output in which the duration of the pulse is the time between two transitions of signal A while signal B is in low state.
  A positive direction of rotation is generated at a positive edge of signal A while signal B is in low state. A negative direction of rotation is generated at a negative edge of signal A while signal B is in low state.

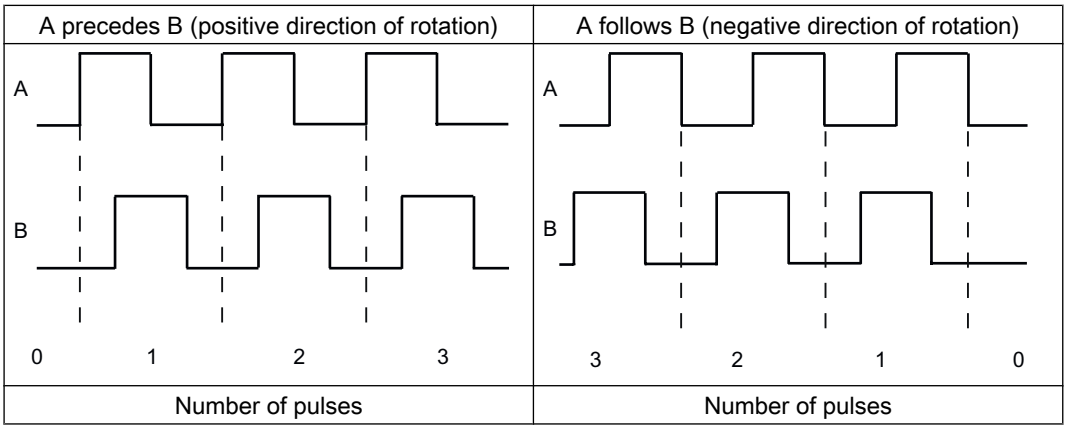

- Incremental encoder (A, B phase-shifted, quadruple):
  Output pulses are output by both outputs at the specified velocity, but phase-shifted by 90 degrees. This signal type involves a quadruple pulse output in which each edge transition corresponds to one increment. Therefore, a complete period of signal A contains four increments. In this way, it is possible, for example, to use two outputs, each with 100 kHz signal frequency, to output a control signal that supplies 400,000 increments per second. Whether count pulses are generated in the positive or negative direction of rotation depends on the edge direction of one signal and the logic state of the other signal in each case.

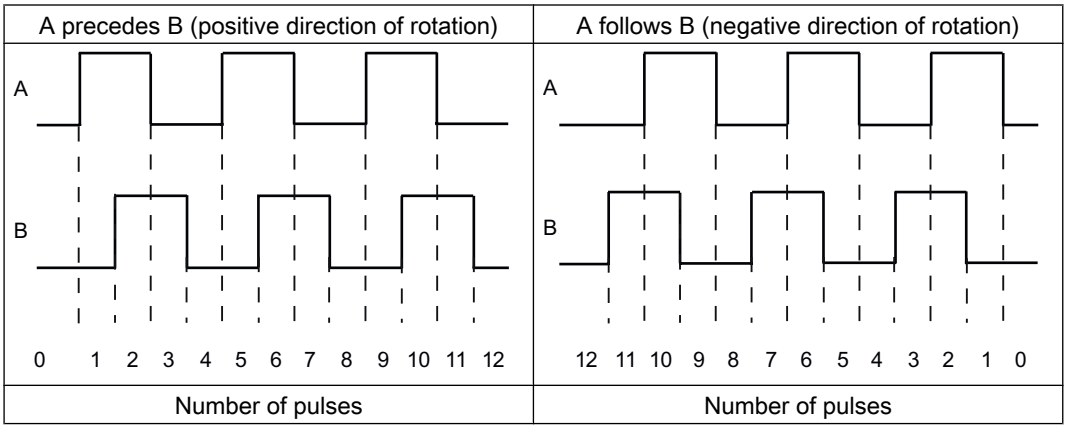

5.2 Using the module (S7-1500)

5.2.1 Using TM Pulse 2x24V (S7-1500)

5.2.1.1 Configuration and parameter assignment of technology module (S7-1500)

Adding a technology module to hardware configuration (S7-1500)

Requirements

- The project has been created.
- The CPU has been created.
- The ET 200 distributed I/O has been created.

Procedure

To add a technology module to the hardware configuration, proceed as follows:

1. Open the device configuration of the CPU or IM.
2. Select a module rack.
3. Select the technology module from the module catalog:
   "Technology modules > Pulse output > TM Pulse 2x24V > Article number"
4. Drag the technology module to the required slot in the module rack.

Parameter setting (hardware configuration) Opening (S7-1500)

Opening from the device view

Proceed as follows:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Click on the module.

Channel configuration (S7-1500)

Channel configuration

This parameter is used to specify whether both channels are to be connected in parallel.

Channel configuration	Meaning
2 channels (2 A) (default)	The two channels operate separately and each has a maximum output current of 2 A.
1 channel (4 A)	The two channels are connected in parallel and operate as a logical channel with a maximum output current of 4 A. All parameter settings and operations that use the control interface must use addresses with Channel 0. Feedback signals are only provided for Channel 0. The control and feedback interface is limited to one channel only. Channel 1 cannot be configured or operated. Current measurement adds the measured values of both channels.

Note

High-speed output

The "High-Speed Output (0.1 A)" can only be selected for the channel configuration "2 Channels (2 A)".

Operating mode (S7-1500)

Operating mode

You use this parameter to specify the function for which the channel of the technology module is used. This defines the setting options under "Parameters".

Operating mode	Meaning	Operating mode Parameter (Page 1276)
Pulse output	The channel outputs a single pulse with the configured on delay. You specify the pulse duration in the control interface.	• High-speed output (0.1 A) • Function DI • Input delay • Output format • On delay
Pulse width modulation PWM (default)	The channel outputs a pulse width modulated signal with the configured minimum pulse duration and the configured period duration. You specify the duty cycle and/or the current setpoint in the control interface.	• High-speed output (0.1 A) • Function DI[1] • Input delay • Output format • Minimum pulse duration • Period duration • On delay [1] • Dithering • Incline of dithering ramp up • Incline of dithering ramp down • Dithering amplitude • Dithering period duration • Current-controlled • Enable proportional action • Enable integral action • Enable derivative component • Reference value current • Dead band width • Manipulated value high limit • Manipulated value low limit • Proportional gain (gain) • Integral action time • Derivative action time • Delay time of derivative action

Operating mode	Meaning	Operating mode Parameter (Page 1276)
Pulse train	The channel outputs a number of pulses with specified period duration, duty cycle and on delay. You specify the number of pulses in the control interface.	• High-speed output (0.1 A) • Function DI • Input delay • Output format • Period duration • On delay • Duty cycle
On/off delay	The channel outputs pulses with specified switching delay with respect to DI0. You specify the on delay in the control interface.	• High-speed output (0.1 A) • Input delay • Output format • Off delay
Frequency output	The channel outputs a signal with a fixed duty cycle of 50%. You specify the frequency in the control interface.	• High-speed output (0.1 A) • Function DI • Input delay • Output format • On delay
PWM with DC motor	The channel outputs a pulse width modulated signal for controlling a DC motor in both directions of rotation. You specify the duty cycle in the control interface.	• Function DI[1] • Input delay • Output format • Period duration • On delay [1]

[1] Parameters cannot be assigned in isochronous mode

Reaction to CPU STOP (S7-1500)

Reaction to CPU STOP

You use this parameter to determine the response of the channel to the failure of a higher-level controller.

Reaction to CPU STOP	Meaning
Output substitute value (default)	The channel outputs the configured substitute values at the digital outputs until the next CPU STOP-RUN transition. The feedback bit STS_ENABLE is set to 0. The module is returned to its startup state after a STOP-RUN transition. Before you can start a new pulse output, you must first set the SW_ENABLE control bit to 0.
Continue operation	The channel continues to operate. The digital outputs continue to switch according to the parameter assignment.

Substitute value for DQA

This parameter lets you specify which value the technology module is to output to the digital output DQA in the event of a CPU STOP under "Output substitute value".

The default setting is "0".

Substitute value for DQB

This parameter lets you specify which value the technology module is to output to the digital output DQB in the event of a CPU STOP under "Output substitute value".

The default setting is "0".

Note

You can select the default value 1 for only one of the two digital outputs.

Diagnostics (S7-1500)

The technology module can trigger additional diagnostic interrupts when you enable the diagnostic interrupts in the basic parameters. These diagnostic interrupts are processed in an interrupt OB.

Group diagnostics

You use this parameter to specify whether a diagnostic interrupt is to be triggered for following errors:

- Supply voltage error (feedback bit ERR_PWR)
- Short-circuit or overload at encoder supply (feedback bit ERR_24V)
- Overheating

The parameter is disabled in the default setting.

Diagnostics DQA

You use this parameter to specify whether a diagnostic interrupt "Error at digital outputs" is to be triggered in the event of an error at the digital output DQA. Regardless of this, the feedback bit ERR_DQA is set when an error occurs.

The parameter is disabled in the default setting.

Diagnostics DQB

You use this parameter to specify whether a diagnostic interrupt "Error at digital outputs" is to be triggered in the event of an error at the digital output DQB. Regardless of this, the feedback bit ERR_DQB is set when an error occurs.

The parameter is disabled in the default setting.

Parameter (S7-1500)

High-speed output (0.1 A)

Use this parameter to specify whether or not the respective digital output works as a fast push-pull switch.

Option	Meaning
Enabled	The digital outputs work as fast push-pull switches and each can carry a rated load current of 0.1 A. A push-pull switch is alternately switched to DC 24 V and ground. This makes for very fast edges.
Disabled (default)	The digital outputs work as fast push-pull switches and each can carry a rated load current of 2 A.

Note

Channel configuration

You can only select this parameter with the channel configuration "2 channels (2 A)".

Note

You cannot enable this parameter in the operating modes "PWM with DC motor" and "Pulse width modulation PWM" with current control.

Function DI

By configuring a digital input, you specify which functions the digital input triggers at switching.

You can select from the following options:

Function	Meaning
Input (pre-selected)	No function is assigned to the respective digital input. The signal status of the digital input can be read by the CPU by means of the feedback interface.
HW enable	The respective digital input is used as enable input for the output sequence.
Limit switches	The respective digital input is used as external limit switch for a DC motor in the "PWM with DC motor" mode.

Input delay

By configuring the input delay, you suppress interferences at the digital inputs. Signals with a pulse duration below the configured input delay are suppressed.

You can select from the following input delays:

- None
- 0.05 ms
- 0.1 ms (default)

- 0.4 ms
- 0.8 ms
- 1.6 ms
- 3.2 ms
- 12.8 ms
- 20 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs.

Output format

Use this parameter to specify the format or the value range with which the duty cycle or the current setpoint is evaluated:

Output format	Value range
S7 analog output	Operating mode "PWM with DC motor": –27.648 ... 27.648 Other operating mode: 0 ... 27648 ("27648" corresponds to 100% of the reference value)
1/100 (default)	0 ... 100
1/1000	0 ... 1000
1/10000	0 ... 10000

Note

In "Frequency output" mode, "1 Hz" is selected as output format and cannot be changed.

In "PWM with DC motor" mode, " "S7 analog output" is selected as output format and cannot be changed.

Minimum pulse duration

This parameter is used to specify the minimum permitted pulse duration for the PWM signal.

Pulses and pauses which fall below the value are suppressed.

The default setting is "0 µs".

Note

The Minimum pulse duration is not effective if you have activated "Current-controlled".

Period duration

This parameter is used to specify the period duration of the output signal.

The default seting is "2000000 µs". This corresponds to 2 s.

Actual period duration

The actual period duration is only displayed in isochronous mode.

The displayed period duration valve is the period duration that is actually compatible with isochronous mode, which is calculated from the value which you input for the period duration.

On delay

This parameter is used to specify the delay with which the start of the output sequence effects the output of the pulse.

The default setting is "0 µs".

Note

Isochronous mode

In isochronous mode you cannot set the On delay in the operating modes "Pulse width modulation PWM" and "PWM with DC motor".

Off delay

This parameter is used to specify the delay with which a falling edge of the digital input DIO effects the digital output DQA.

The default setting is "0 µs".

Duty cycle

This parameter is used to specify the pulse-pause ratio (also referred to as pulse duty factor or duty factor) of the output signal in the selected output format.

The default setting is "50%".

Dithering

You can use this parameter to specify whether the PWM signal will be overlaid by a dither signal:

No dithering is enabled in the default setting.

Incline of dithering ramp up

You can use this parameter to specify the incline of the ramp up of the dithering amplitude for a theoretical increase of 0% to 100%.

The incline influences the effective ramp time:

(Dithering amplitude) x (Incline of dithering ramp up) = Effective ramp time

Example for calculation of the effective ramp time:

10 % x 2500 ms/100 % = 250 ms

The default setting is "0".

Incline of dithering ramp down

You can use this parameter to specify the incline of the ramp down of the dithering amplitude for a theoretical drop from 100% to 0%.

The incline influences the effective ramp time:

(Dithering amplitude) x (Incline dithering ramp down) = Effective ramp time

Example for calculation of the effective ramp time:

10% x 2500 ms/100% = 250 ms

The default setting is "0".

Dithering amplitude

You can use this parameter to specify the amplitude of the dither signal in relation to the period duration of the output signal.

The default setting is "5.0%".

Dithering period duration

This parameter is used to specify the period duration of the dither signal. The parameter can be set in a fixed grid depending on the period duration of the PWM signal.

The default setting is "50000 µs".

Dithering frequency

This value is calculated from the configured period duration of the dither signal.

Current-controlled

You can use this parameter to activate the PID algorithm for the control of the output current.

The parameter is disabled in the default setting.

Note

High-speed output

You can only use the current control with all associated parameters in the operating mode "Pulse width modulation PWM" with deactivated "high-speed output (0.1 A)".

Enable proportional action

You can use this parameter to activate the proportional action of the PID algorithm.

The parameter is enabled in the default setting.

Enable integral action

You can use this parameter to activate the integral action of the PID algorithm.

The parameter is enabled in the default setting.

Enable derivative action

You can use this parameter to activate the derivative action of the PID algorithm.

The parameter is disabled in the default setting.

Current reference value

You can use this parameter to specify the maximum current as reference value for the manipulated variables and the current setpoint.

The default setting is "0 mA".

Dead band width

You can use this parameter to specify the deviation of the output current from the current setpoint, within which there is no readjustment. The deviation refers to above and below the current setpoint.

The default setting is "0 µA".

Manipulated value high limit

This parameter is used to specify the high control limit relative to the current reference value. "27648" corresponds to 100% of the current reference value. The value must be greater than the "Manipulated variable low limit".

The default setting is "27648".

Manipulated variable low limit

This parameter is used to specify the low control limit relative to the current reference value. "0" corresponds to 0% of the current reference value. The value must be less than the "Manipulated variable high limit".

The default setting is "0".

Proportional gain (gain)

This parameter is used to define the gain factor for the proportional action of the PID algorithm.

The default setting is "2.0000".

Integral action time

This parameter defines the time used by the integral action of the PID algorithm.

The default setting is "2.0000".

Derivative action time

This value defines the time used by the derivative action of the PID algorithm.

Delay time of derivative action

This value defines the delay time of the derivative action of the PID algorithm.

5.2.1.2 Online & Diagnostics module (S7-1500)

Displaying and evaluating diagnostics (S7-1500)

The online and diagnostics view enables hardware diagnostics. You can also

- Obtain information on the module (e.g., Firmware version and serial number)
- Execute a firmware update if required

Procedure

To open the display editor for the diagnostic functions, follow these steps:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Right-click on the module and select "Online & Diagnostics".
4. Select the required display in the diagnostics navigation.

Additional information

Additional information on the diagnostic alarms and possible remedies can be found in the module's device manual.

5.2.2 Using TM PTO 4 (S7-1500)

5.2.2.1 Configuration and parameter assignment of technology module (S7-1500)

Adding a technology module to hardware configuration (S7-1500)

Requirements

- The project has been created.
- The CPU S7-1500 has been created.

Procedure

To add a technology module to the hardware configuration, proceed as follows:

1. Open the device configuration of the CPU or IM.
2. Select a module rack.
3. Select the technology module from the module catalog:
   "Technology modules > Pulse output > TM PTO 4 > Article number"
4. Drag the technology module to the required slot in the module rack.

Parameter setting (hardware configuration) Opening (S7-1500)

Opening from the device view

Proceed as follows:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Click on the module.

Channel configuration (S7-1500)

Channel configuration

This parameter defines the number of channels used. The channels are assigned in ascending order.

You can select from the following options:

- 4 channels (default)
- 3 channels

- 2 channels
- 1 channel

Operating mode (S7-1500)

Signal type

Use this parameter to specify the type of the PTO pulse output (Page 1268).

Option	Meaning
Pulse (P) and direction (D)	The channel outputs a pulse signal and a direction signal.
Count up (A), count down (B)	The channel outputs a forward signal and a backward signal.
Incremental encoder (A, B phase-shifted)	The channel outputs two phase-shifted signals with single evaluation.
Incremental encoder (A, B phase-shifted, quadruple)	The channel outputs two phase-shifted signals with fourfold evaluation.

Signal interface

You can use this parameter to define which interface is used for the pulse output.

Option	Meaning
24 V, asymmetrical	The channel outputs 24 V signals at terminals DQm.0 and DQm.1.
RS422, symmetrical / TTL (5 V), asymmetrical (default)	The channel outputs either RS422 signals at the terminals P/A and D/B and the respectively inverted terminals or 5 V-TTL signals at the terminals P/A and D/B.

Pulse pause on reversal of direction

This parameter is used to define the minimum time between a change in direction and the output of the first pulse in the new direction.

You can select from the following options:

- 0 ms (default)
- 1 ms
- 4 ms
- 10 ms

Diagnostic interrupts (S7-1500)

Enable diagnostic interrupts

You can use this parameter to specify whether a diagnostic interrupt is to be triggered when no supply voltage is available or in the event of an error at the digital outputs. The detected error is displayed for the affected channel using feedback bit Fault_Present and Sensor_Error.

Note

The hardware outputs of the channels 0 and 1 and channels 2 and 3 each have common diagnostics. Due to this, in the event of an error at an output, pulse output of each of the two pairs of channels is stopped. A short-circuit at an output can also trigger the diagnostic interrupt "Supply voltage missing".

These diagnostic interrupts are processed in an interrupt OB.

The parameter is disabled in the default setting.

Axis parameters (S7-1500)

Note

Technology object for Motion Control

Make sure that the respective parameter value matches the value of the technology object used in the configuration.

Increments per revolution

You can use this parameter to specify the number of steps (also micro steps) which corresponds to one revolution of the drive.

The range of values is 1 to 1000000. The value influences the value range of the maximum speed and reference speed. The default setting is "200".

Reference speed

This parameter is used to define the speed at which the drive rotates with a speed setpoint of 100%. The range of values is 1.0 to 20000.0 rpm. The permitted value range for the speed setpoint is -200% to 200% of the reference speed.

The reference speed influences the value range of the maximum speed and increments per revolution. The default setting is "3000.0".

Maximum speed

You can use this parameter to define the maximum permitted speed for the application.

The following table shows the calculation of the value range of the maximum speed:

Signal interface	Value range of the maximum speed			
	Low limit		High limit (the smaller of the two values applies)	
	Signal evaluation "Single"	Signal evaluation "Quadruple"	Signal evaluation "Single"	Signal evaluation "Quadruple"
24 V, asymmetrical	0.1 Hz * 60 / (increments per revolution)	0.1 Hz * 60 *4 / (increments per revolution)	• 2 * reference speed • 200000 Hz * 60 / (increments per revolution)	• 2 * reference speed • 200000 Hz * 60 *4 / (increments per revolution)
TTL (5 V), asymmetrical				
RS422, symmetrical			• 2 * reference speed • 1000000 Hz * 60 / (increments per revolution)	• 2 * reference speed • 1000000 Hz * 60 *4 / (increments per revolution)

The default setting is "3000.0".

Bits in incr. process value (G1_XIST1)

This parameter specifies the number of bits for the coding of the fine resolution within the incremental position process value of G1_XIST1.

The value is always "0" for this module.

Quick stop time (OFF3)

You can use this parameter to define the time interval to perform a fast stop from maximum speed to standstill.

The range of values is 1 to 65535 ms. The default setting is "1000 ms".

Ramp stop time (OFF1)

You can use this parameter to define the time interval to perform a stop from maximum speed to standstill.

The range of values is 1 to 65535 ms. The default setting is "5000 ms".

Hardware inputs/outputs (S7-1500)

Use drive enable

You use this parameter to define whether a hardware output will be used for the enabling signal for the drive.

The parameter is disabled in the default setting.

Drive enable output

You use this parameter to specify which hardware output will be used to enable the drive.

You can select from the following options:

- DQ0 (default)
- DIQ2

Use DI0 as reference switch

You use this parameter to define whether the hardware output DI0 will be used for the reference switch signal for the drive. You can synchronize the reference mark with the current position of the drive axis using the reference switch signal.

The parameter is disabled in the default setting.

Edge selection reference switch

This parameter is used to define the edge at DI0 that triggers detection of the reference mark.

You can select from the following options:

- At rising edge (default)
- At falling edge

Note

The parameter is only active if you use a basic-position-switch input.

This parameter is only available for selection and effective if "Use drive enable" is enabled. The default setting is "None".

Use DI1 as measuring input

You use this parameter to define whether the hardware output DI1 will be used for the measuring input signal for the drive. You can save the current position of the drive axis using the measuring input signal.

The parameter is disabled in the default setting.

Use "Drive ready"

You use this parameter to specify whether the ready signal of the drive is used at a hardware digital input.

The parameter is disabled in the default setting.

"Drive ready" input

You use this parameter to specify at which hardware digital input the ready signal of the drive is connected and read in.

You can select from the following options:

- DI0 (default)
- DI1
- DIQ2

Note

This parameter is only available for selection and effective if "Use "Drive ready"" is enabled.

Input delay

You use this parameter to suppress signal interference at the digital inputs. Changes to the signal are only detected if they remain stable for longer than the configured input delay time.

You can select from the following options:

- None
  (input delay of 4 µs, minimum pulse width of 3 µs)
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms
- 1.6 ms
- 3.2 ms
- 12.8 ms
- 20 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs.

Sign-of-life error (S7-1500)

Tolerated number of sign-of-life errors

This parameter is used to define how many Master Sign-Of-Life errors are tolerated by the module. If the number is exceeded, an error message is triggered via the return bit Sensor_Error.

The range of values is 1 to 65535. The value 65535 indicates that no monitoring for sign-of-life errors is performed. The default setting is "1".

5.2.2.2 Online & Diagnostics module (S7-1500)

Displaying and evaluating diagnostics (S7-1500)

The online and diagnostics view enables hardware diagnostics. You can also

- Obtain information on the module (e.g., Firmware version and serial number)
- Execute a firmware update if required

Procedure

To open the display editor for the diagnostic functions, follow these steps:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Right-click on the module and select "Online & Diagnostics".
4. Select the required display in the diagnostics navigation.

Additional information

Additional information on the diagnostic alarms and possible remedies can be found in the module's device manual.

5.2.3 Using Compact CPU (S7-1500)

5.2.3.1 Configuring and assigning parameters for Compact CPU (S7-1500)

Adding Compact CPU to the hardware configuration (S7-1500)

Requirements

The project has been created.

Procedure

To add a compact CPU to the hardware configuration, proceed as follows:

1. Double-click "Add new device".
   The "Add new device" dialog opens.
2. Select Controller".
3. Select the Compact CPU:
   "SIMATIC S7-1500 > CPU > Name of Compact CPU > Article number"
4. Confirm with "OK".

Result

The new Compact CPU is displayed with the following objects in the project tree. Double-click to open the required editor.

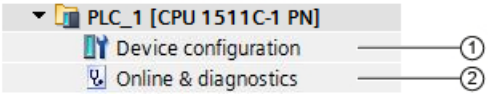

	Object	Description
①	Device configuration	In the Inspector window (per PTO/PWM channel): • Setting the operating mode (Page 1289) • Setting the reaction to CPU STOP (Page 1291) • Enable diagnostic interrupts (Page 1291) • Setting parameters for PWM or frequency output (Page 1293) • Setting axis parameters for PTO modes (Page 1294) • Assign signals to inputs and outputs (Page 1291) • Setting the module addresses
②	Online & Diagnostics (Page 1295)	• Hardware diagnostics • Obtain information about the Compact CPU • Run firmware update

Compatibility 1511C (pulse generators of the Compact CPU 1512C-1 PN) (S7-1500)

Front connector assignment same as CPU 1511C

You can use this parameter to define if the pin assignment for the front connector of the CPU 1511C-1 PN is to be used for the pulse generators of the CPU 1512C-1 PN:

Option	Meaning
Disabled (default)	CPU 1512C-1 PN uses the pin assignment of the onboard front connector. 1512C-1 PN supports the use of the connections of both front connectors of the digital onboard I/Os for the pulse generators. The assignment of hardware inputs and outputs for the PTO/PWM channels is described in the product manual of the CPU 1512C-1 PN.
Enabled	CPU 1512C-1 PN uses the pin assignment of the front connector of the CPU 1511C-1 PN. 1511C-1 PN supports the use of the connections of the first front connector of the digital onboard I/Os for the pulse generators. The assignment of hardware inputs and outputs for the PTO/PWM channels is described in the product manual of the CPU 1511C-1 PN.

Operating mode (S7-1500)

Operating mode

You use this parameter to specify the function for which the pulse generator of the compact CPU is used. You can find possible settings that are dependent on the operation mode under:

Reaction to CPU STOP (Page 1291)

Diagnostic interrupts (Page 1291)

Hardware inputs/outputs (PWN, frequency output) (Page 1291) or Hardware inputs/outputs (PTO) (Page 1292)

Parameter (PWN, frequency output) (Page 1293) or Axis parameters (PTO) (Page 1294)

Operating mode	Meaning	Mode-dependent parameters
Disabled (default)	The pulse generator is not activated. The channel does not output pulses.	—
Pulse width modulation PWM	The channel outputs a pulse width modulated signal (Page 1257) with the configured minimum pulse duration and the configured period duration. The duty cycle and the period duration are defined in the control interface.	• Reaction to CPU STOP • Substitute value for pulse output (DQA) • No supply voltage L+ • Pulse output (DQA) • High-speed output (0.1 A) • Output format • Minimum pulse duration • Period duration
Frequency output	The channel outputs a signal with a fixed duty cycle (Page 1261) of 50%. You specify the frequency in the control interface.	• Reaction to CPU STOP • Substitute value for pulse output (DQA) • No supply voltage L+ • Pulse output (DQA) • High-speed output (0.1 A) • Output format
PTO (pulse (A) and direction (B))	The channel outputs a pulse signal and a direction signal (Page 1264) for a Motion Control application.	• No supply voltage L+ • Increments per revolution • Reference speed • Maximum speed • Bits in incr. process value (G1_XIST1) • Time OFF3 • Input reference switch • Edge selection reference switch • Measurement input • "Drive ready" input • Pulse output (A) / Clock generator forward (A) / Clock generator output (A) • Direction output (B) / Clock generator backward (B) / Clock generator output (B) • Drive enable output
PTO (Count up (A), count down (B)):	The channel outputs a forward signal and a backward signal (Page 1264) for a Motion Control application.	
PTO (A, B phase-shifted)	The channel outputs two phase-shifted signals with single evaluation (Page 1264) for a Motion Control application.	
PTO (A, B phase-shifted, quadruple)	The channel outputs two phase-shifted signals with quadruple evaluation (Page 1264) for a Motion Control application.	

Reaction to CPU STOP (S7-1500)

Reaction to CPU STOP

You use this parameter to specify the reaction of the channel to the failure of the controller.

Reaction to CPU STOP	Meaning
Output substitute value (default)	The channel outputs the configured substitute value at its digital output (DQA) until the next CPU STOP-RUN transition. The feedback bit STS_ENABLE is set to 0. The channel is returned to its startup state after a STOP-RUN transition. Before you can start a new pulse output, you must first set the SW_ENABLE control bit to 0.
Continue operation	The channel continues to operate. The digital output (DQA) continue to switch according to the parameter assignment and the last setpoint.

Substitute value for pulse output (DQA)

This parameter lets you specify which value the channel is to output to the digital output DQA in the event of a CPU STOP under "Output substitute value".

The default setting is "0".

Diagnostic interrupts (S7-1500)

Missing supply voltage L+

You can use this parameter to specify whether a diagnostic interrupt is to be triggered when no supply voltage is available. These diagnostic interrupts are processed in an interrupt OB.

The parameter is disabled in the default setting.

Hardware inputs/outputs (PWN, frequency output) (S7-1500)

Pulse output (DQA)

You can use this parameter to define which hardware digital output is used for the pulse output.

High-speed output (0.1 A)

Use this parameter to specify whether or not the respective hardware digital output works as fast push-pull or as sourcing output.

Option	Meaning
Enabled	The digital output operates as very fast push-pull switch with up to 100 kHz and is capable of bearing loads up to 0.1 A. A push-pull switch is alternately switched to DC 24 V and ground. This makes for very fast edges.
Disabled (default)	The digital output works as 24 vacuum circuit-breaker/current-sourcing output with reference to M (up to 10 kHz) and can carry a rated load current of 0.5 A.

Note

The parameter is only effective if the hardware output selected as pulse output supports 100 kHz operation. The parameter is not available for hardware outputs that only support frequencies up to 100 Hz.

Hardware inputs/outputs (PTO) (S7-1500)

Input reference switch

You can use this parameter to define which hardware digital input is used as reference switch input.

The default setting is "None".

Edge selection reference switch

This parameter is used to define the edge at the reference switch input that triggers detection of the reference mark.

The default setting is "Positive edge".

Note

The parameter is only active if you use a basic-position-switch input.

Measurement input

You can use this parameter to define which hardware digital input is used as measurement input.

The default setting is "None".

"Drive ready" input

You use this parameter to specify at which hardware digital input the ready signal of the drive is connected and read in.

The default setting is "None".

Pulse output (A) / Clock generator forward (A) / Clock generator output (A)

This parameter defines which hardware output is used for the PTO signal A. The value cannot be modified.

Direction output (B) / Clock generator backward (B) / Clock generator output (B)

This parameter defines which hardware output is used for the PTO signal B. You can change the value only in operating mode "PTO (pulse (A) and direction (B))".

Drive enable output

You use these parameters to specify which hardware output will be used to enable the drive.

The default setting is "None".

Parameter (PWN, frequency output) (S7-1500)

Output format

"Pulse width modulation PWM" operating mode:

Use this parameter to specify the format or the value range with which the duty cycle is evaluated:

Output format	Value range
S7 analog output	0 ... 27648 ("27648" corresponds to 100% of the reference value)
1/100 (default)	0 ... 100
1/1000	0 ... 1000
1/10000	0 ... 10000

"Frequency output" operating mode:

"1 Hz" is selected as output format and cannot be changed.

Minimum pulse duration

This parameter is used to specify the minimum permitted pulse duration for the PWM signal.

Pulses and pauses which fall below the value are suppressed.

The default setting is "0 µs".

Period duration

This parameter is used to specify the period duration of the output signal.

The default setting is "2000000 µs". This corresponds to 2 s.

Axis parameters (PTO) (S7-1500)

Increments per revolution

You can use this parameter to specify the number of steps (also micro steps) which corresponds to one revolution of the drive.

The default setting is "200".

Note

Technology object for Motion Control

Make sure that the parameter value matches the value of the technology object used in the configuration.

Reference speed

You can use this parameter to define the speed to which the speed setpoint is to refer.

The permitted value range for the speed setpoint is -200% to 200% of the reference speed.

The default setting is "3000.0".

Note

Technology object for Motion Control

Make sure that the parameter value matches the value of the technology object used in the configuration.

Maximum speed

You can use this parameter to define the maximum permitted speed for the application.

The default setting is "3000.0".

Note

Technology object for Motion Control

Make sure that the parameter value matches the value of the technology object used in the configuration.

Bits in incr. process value (G1_XIST1)

This parameter specifies the number of bits for the coding of the fine resolution within the incremental process value of G1_XIST1.

Note

Technology object for Motion Control

Make sure that the parameter value matches the value of the technology object used in the configuration.

Time OFF3

You can use this parameter to define the time interval to perform a fast stop from maximum speed to standstill.

The default setting is "1000 ms".

5.2.3.2 Online & Diagnostics module (S7-1500)

Displaying and evaluating diagnostics (S7-1500)

The online and diagnostics view enables hardware diagnostics. You can also

- Obtain information on the compact CPU (e.g. Firmware version and serial number)
- Execute a firmware update if required

Procedure

To open the display editor for the diagnostic functions, follow these steps:

1. Open the Compact CPU folder in the project tree.
2. Double-click on the "Online & diagnostics" object.
3. Select the required display in the diagnostics navigation.

Additional information

Additional information on the diagnostic alarms and possible remedies can be found in the compact CPU device manual.

Motion Control 6

6.1 Motion Control (S7-1200, S7-1500) (S7-1200, S7-1500)

6.1.1 Using S7-1200 Motion Control (S7-1200)

6.1.1.1 Introduction (S7-1200)

Motion functionality of the CPU S7-1200 (S7-1200)

The TIA Portal, together with the motion control functionality of the CPU S7-1200, supports you in controlling stepper motors and servo motors:

- You configure the positioning axis and command table technology objects in the TIA Portal. The CPU S7-1200 uses these technology objects to control the outputs that control the drives.
- In the user program you control the axis by means of Motion Control instructions and initiate motion commands of your drive.

See also

Hardware components for motion control (Page 1298)

Integration of the positioning axis technology object (Page 1336)

Tools of the positioning axis technology object (Page 1339)

Use of the command table technology object (Page 1401)

Command table technology object tools (Page 1401)

Hardware components for motion control (S7-1200)

The representation below shows the basic hardware configuration for a motion control application with the CPU S7-1200.

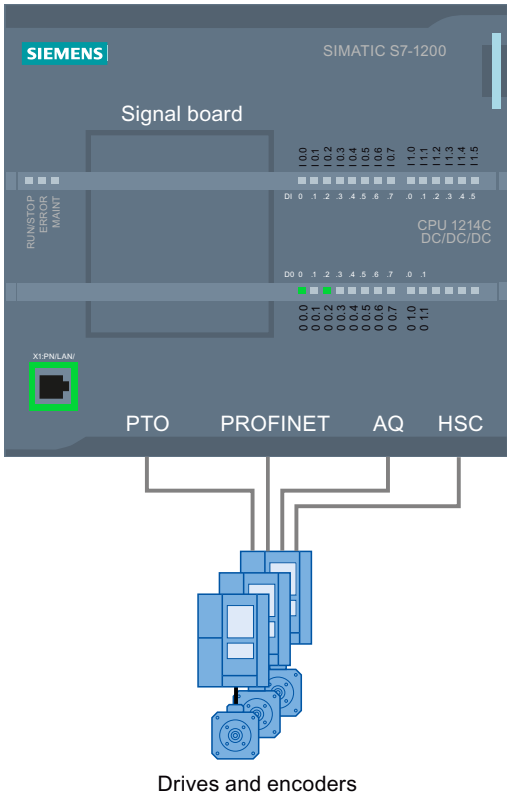

CPU S7-1200

CPU S7-1200 combines the functionality of a programmable logic controller with motion control functionality for operation of drives. The motion control functionality takes over the control and monitoring of the drives.

Signal board

You add further inputs and outputs to the CPU with the signal boards.

You can use the digital outputs as pulse generator outputs for controlling drives as required. In CPUs with relay outputs, the pulse signal cannot be output on the onboard outputs because the relays do not support the necessary switching frequencies. To be able to work with the PTO (Pulse Train Output) on these CPUs, you must use a signal board with digital outputs.

You can use the analog outputs for controlling connected analog drives as required.

PROFINET

Use the PROFINET interface to establish the online connection between the CPU S7-1200 and the programming device. In addition to the online functions of the CPU, additional commissioning and diagnostic functions are available for motion control.

PROFINET continues to support the PROFIdrive profile for connecting PROFIdrive capable drives and encoders.

Drives and encoders

Drives permit the movement of the axis. Encoders provide the actual position for the closed loop position control of the axis.

The table below shows the connection possibilities for drives and encoders:

Drive connection	Closed/open loop control of axis	Encoder connection
PTO (Pulse Train Output) (Stepper motors and servo motors with pulse interface)	Position-controlled	-
Analog output (AQ)	Position-controlled	• Encoder on high-speed counter (HSC) • Encoder on technology module (TM) • Encoder on PROFINET
PROFINET	Position-controlled	• Encoder on the drive • Encoder on high-speed counter (HSC) • Encoder on technology module (TM) • Encoder on PROFINET

Name	Article number
CPU 1211C DC/DC/DC	6ES7211-1AE40-0XB0
CPU 1211C AC/DC/RLY	6ES7211-1BE40-0XB0
CPU 1211C DC/DC/RLY	6ES7211-1HE40-0XB0
CPU 1212C DC/DC/DC	6ES7212-1AE40-0XB0
CPU 1212C AC/DC/RLY	6ES7212-1BE40-0XB0
CPU 1212C DC/DC/RLY	6ES7212-1HE40-0XB0
CPU 1214C DC/DC/DC	6ES7214-1AG40-0XB0
CPU 1214C AC/DC/RLY	6ES7214-1BG40-0XB0
CPU 1214C DC/DC/RLY	6ES7214-1HG40-0XB0
CPU 1214FC DC/DC/DC	6ES7214-1AF40-0XB0
CPU 1214FC DC/DC/RLY	6ES7214-1HF40-0XB0
CPU 1215C DC/DC/DC	6ES7215-1AG40-0XB0
CPU 1215C AC/DC/RLY	6ES7215-1BG40-0XB0
CPU 1215C DC/DC/RLY	6ES7215-1HG40-0XB0
CPU 1215FC DC/DC/DC	6ES7215-1AF40-0XB0
CPU 1215FC DC/DC/RLY	6ES7215-1HF40-0XB0

Name	Article number
CPU 1217C DC/DC/DC	6ES7217-1AG40-0XB0
Signal board DI4 x DC 24 V (200 kHz)	6ES7221-3BD30-0XB0
Signal board DI4 x DC 5 V (200 kHz)	6ES7 221-3AD30-0XB0
Signal board DQ4 x DC 24 V (200 kHz)	6ES7222-1BD30-0XB0
Signal board DQ4 x DC 5 V (200 kHz)	6ES7222-1AD30-0XB0
Signal board DI2/DQ2 x DC 24 V (20 kHz)	6ES7223-0BD30-0XB0
Signal board DI2/DQ2 x DC 24 V (200 kHz)	6ES7223-3BD30-0XB0
Signal board DI2/DQ2 x DC 5 V (200 kHz)	6ES7223-3AD30-0XB0
Signal board AQ1 x 12 bit (±10 V, 0 to 20 mA)	6ES7 232-4HA30-0XB0

Ordering information for CPU firmware V4.1

The order information listed below applies to the currently installed product phase (without any installed Hardware Support Packages) of the TIA Portal.

Use a Hardware Support Package (HSP) to install new hardware components. The hardware component will then be available in the hardware catalog.

See also

Motion functionality of the CPU S7-1200 (Page 1297)

CPU outputs relevant for motion control (Page 1300)

6.1.1.2 Basics for working with S7-1200 Motion Control (S7-1200)

Stepper motor on the PTO (S7-1200)

CPU outputs relevant for motion control (S7-1200)

The number of usable drives depends on the number of PTOs (pulse train outputs) and the number of available pulse generator outputs.

The following tables provide information about the relevant dependencies:

Maximum number of PTOs

4 PTOs are available for each CPU with technology version V4. This means a maximum of 4 drives can be controlled.

Signal type of the PTO

Depending on the signal type of the PTO, 1-2 pulse generator outputs are required per PTO (drive):

Signal type	Number of pulse generator outputs
Pulse A and direction B (direction output disabled)[1]	1
Pulse A and direction B [1]	2
Clock up A and clock down B	2
A/B phase-shifted	2
A/B phase-shifted - quadruple	2

[1] The direction output must be on-board or on a signal board.

Usable pulse generator outputs and limit frequencies

The relay variants of the CPUs can only access the pulse generator outputs of a signal board.

Depending on the CPU the pulse generator outputs Q0.0 to Q1.1 can be used with the following limit frequencies:

CPU	Q0.0	Q0.1	Q0.2	Q0.3	Q0.4	Q0.5	Q0.6	Q0.7	Q1.0	Q1.1
1211 (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	-	-	-	-	-	-
1212 (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	20 kHz	20 kHz	-	-	-	-
1214(F) (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	20 kHz	20 kHz	20 kHz	20 kHz	20 kHz	20 kHz
1215(F) (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	20 kHz	20 kHz	20 kHz	20 kHz	20 kHz	20 kHz
1217 (DC/DC/DC)	1 MHz	1 MHz	1 MHz	1 MHz	100 kHz	100 kHz	100 kHz	100 kHz	100 kHz	100 kHz

Depending on the signal board, the pulse generator outputs Qx.0 to Qx.3 can be used with the following limit frequencies:

Signal board	Qx.0	Qx.1	Qx.2	Qx.3	-	-	-	-	-	-
DI2/DQ2 x DC24V 20kHz	20 kHz	20 kHz	-	-	-	-	-	-	-	-
DI2/DQ2 x DC24V 200kHz	200 kHz	200 kHz	-	-	-	-	-	-	-	-
DQ4 x DC24V 200kHz	200 kHz	200 kHz	200 kHz	200 kHz	-	-	-	-	-	-
DI2/DQ2 x DC5V 200kHz	200 kHz	200 kHz	-	-	-	-	-	-	-	-
DQ4 x DC5V 200kHz	200 kHz	200 kHz	200 kHz	200 kHz	-	-	-	-	-	-

The low limit frequency is always 1Hz.

The pulse generator outputs can be freely assigned to the PTOs.

Note

If pulse generator outputs with different limit frequencies are used in accordance with the signal type, the low limit frequency is used in each case.

Signal type "Pulse A and direction B" is an exception. With this type of signal, the limit frequency of the pulse generator output is always used.

Note

Access to pulse generator outputs via the process image

The firmware takes control via the corresponding pulse generator and direction outputs if the PTO (Pulse Train Output) is selected and assigned to an axis.

With this takeover of the control function, the connection between the process image and I/O output is also disconnected. Although the user has the option of writing the process image of pulse generator and direction outputs via the user program or watch table, this is not transferred to the I/O output. Accordingly, it is also not possible to monitor the I/O output via the user program or watch table. The information read reflects the value of the process image and does not match the real status of the I/O output.

For all other CPU outputs that are not used permanently by the CPU firmware, the status of the I/O output can be controlled or monitored via the process image, as usual.

Outputs for drive signals

For motion control, you can optionally parameterize a drive interface for "Drive enabled" and "Drive ready".

When using the drive interface the digital output for the drive enable and the digital input for "drive ready" can be freely selected.

Acceleration/deceleration limits

The following limits apply to acceleration and deceleration:

Acceleration/deceleration	Value
Minimum acceleration/deceleration	5.0E-3 pulses/s^2
Maximum acceleration/deceleration	9.5E+9 pulses/s^2

Jerk limits

The following limits apply to the jerk:

Jerk	Value
Minimum jerk	4.0E-3 pulses/s^3
Maximum jerk	1.0E+10 pulses/s^3

See also

CPU outputs relevant for motion control (technology version V1...3) (Page 1528)
How the pulse interface works (Page 1303)
Relationship between the signal type and the direction of travel (Page 1303)
Hardware and software limit switches (Page 1322)
Jerk limit (Page 1323)
Homing (Page 1324)
Hardware components for motion control (Page 1298)
Integration of the positioning axis technology object (Page 1336)
Tools of the positioning axis technology object (Page 1339)

How the pulse interface works (S7-1200)

Depending on the settings of the stepper motor, each pulse affects the movement of the stepper motor by a specific angle. If the stepper motor is set to 1000 pulses per revolution, for example, it moves 0.36° per pulse.

The speed of the stepper motor is determined by the number of pulses per time unit.

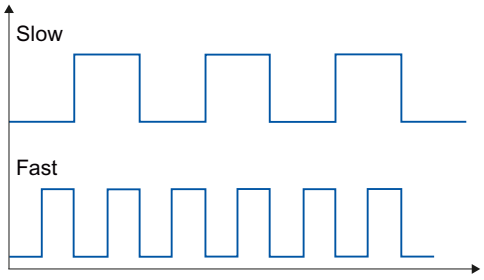

The statements made here also apply to servo motors with pulse interface.

See also

CPU outputs relevant for motion control (Page 1300)
Relationship between the signal type and the direction of travel (Page 1303)
Hardware and software limit switches (Page 1322)
Jerk limit (Page 1323)
Homing (Page 1324)
Integration of the positioning axis technology object (Page 1336)
Tools of the positioning axis technology object (Page 1339)

Relationship between the signal type and the direction of travel (S7-1200)

The CPU outputs the velocity and direction of travel via two outputs.

The relationships between the configuration and direction of travel differ depending on the selected signal type. You can configure the following signal types in the axis configuration under "Basic parameters > General":

- "PTO – pulse A and direction B"
- "PTO – clock up A and clock down B" (as of V4)
- "PTO – A/B phase-shifted" (as of V4)
- "PTO – A/B phase-shifted, quadruple" (as of V4)

You configure the direction under "Extended Parameters > Mechanics" in the axis configuration. If you select the "Invert direction" option, the direction logic described below for the respective signal type is inverted.

PTO – pulse (A) and direction (B)

The pulse output pulses and the direction output level are evaluated for this signal type.

The pulses are output via the pulse output of the CPU. The direction output of the CPU specifies the direction of rotation of the drive:

- 5 V/24 V at direction output ⇒ positive direction of rotation
- 0 V at direction output ⇒ negative direction of rotation

The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.

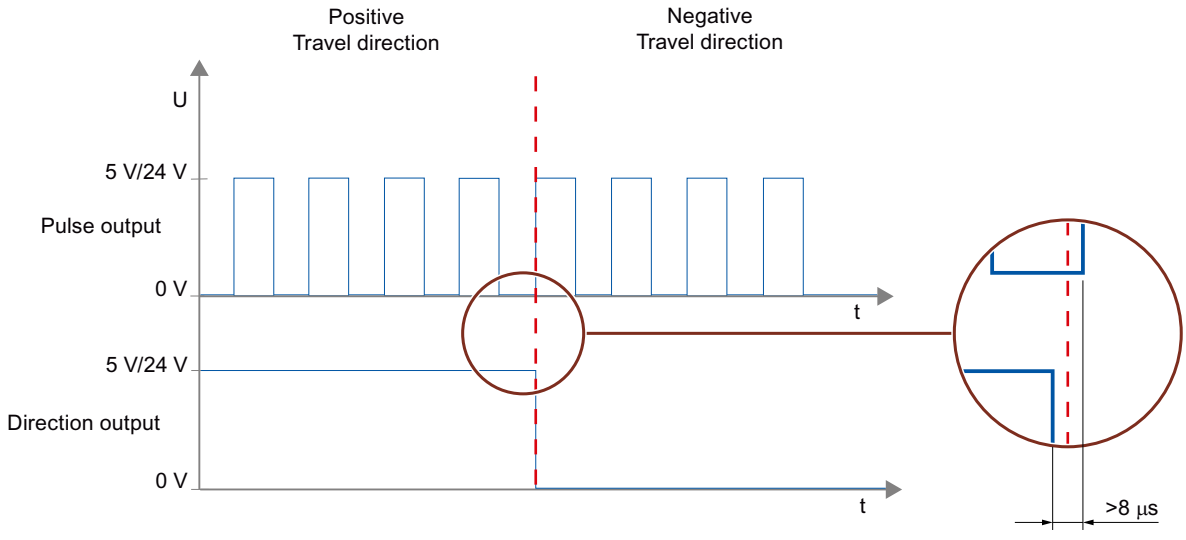

PTO – clock up A and clock down B (as of V4)

The pulses of one output are evaluated for this signal type.

The pulse for the positive direction is output via the "Pulse output up" The pulse for the negative direction is output via the "Pulse output down"

The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.

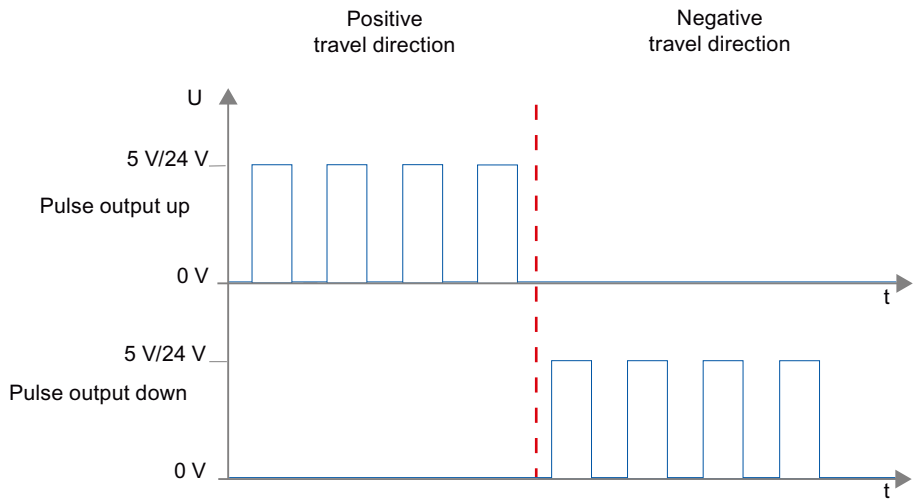

PTO – A/B phase-shifted (as of V4)

The positive edges of one output in each case are evaluated for this signal type.

The pulse is output via the "Signal A" output and phase-shifted via the "Signal B" output. The phase shifting between the outputs defines the direction of rotation:

- Signal A leads signal B by 90° ⇒ positive direction of rotation
- Signal B leads signal A by 90° ⇒ negative direction of rotation

PTO – (A/B phase-shifted - quadruple) (as of V4)

The positive and negative edges of both outputs are evaluated for this signal type. A pulse period has four edges with two phases (A and B). The pulse frequency at the output is therefore reduced to a quarter.

The pulse is output via the "Signal A" output and phase-shifted via the "Signal B" output. The phase shifting between the outputs defines the direction of rotation:

- Signal A leads signal B by 90° ⇒ positive direction of rotation
- Signal B leads signal A by 90° ⇒ negative direction of rotation

The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.

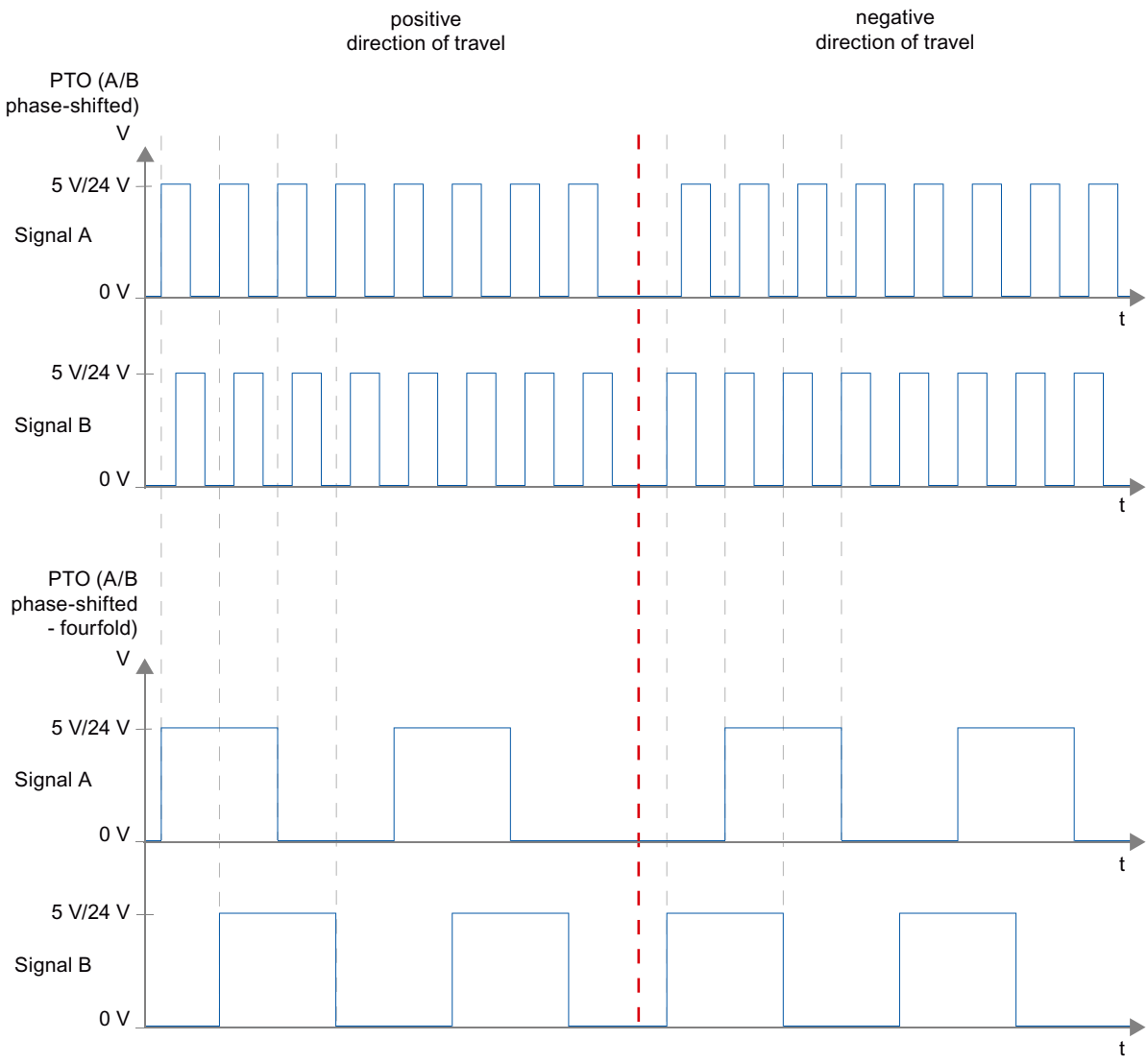

Invert direction

If you select the "Invert rotation signal" option, the direction logic is inverted:

- PTO – pulse (A) and direction (B)
  - 0 V at direction output (low level) ⇒ positive direction of rotation
  - 5 V/24 V at direction output (high level) ⇒ negative direction of rotation

  The specified voltage depends on the hardware used. The voltages indicated do not apply to the differential outputs of CPU 1217.
- PTO – clock up A and clock down B
  The outputs "Pulse output down" and "Pulse output up" are swapped.

- PTO – A/B phase-shifted
  The "Signal A" and "Signal B" outputs are swapped.
- "PTO – A/B phase-shifted, quadruple
  The "Signal A" and "Signal B" outputs are swapped.

See also

CPU outputs relevant for motion control (Page 1300)

How the pulse interface works (Page 1303)

Hardware and software limit switches (Page 1322)

Jerk limit (Page 1323)

Homing (Page 1324)

Integration of the positioning axis technology object (Page 1336)

Tools of the positioning axis technology object (Page 1339)

PROFIdrive drive /analog drive connection (S7-1200)

Drive and encoder connection (S7-1200)

A drive and an encoder are assigned to a positioning axis with drive connection via PROFIdrive/ analog drive connection.

Drives with PROFIdrive capability are connected by means of PROFIdrive telegrams. The setpoint is specified via PROFIdrive telegrams.

Drives with analog setpoint interfaces are connected using an analog output and an optional enable signal. The setpoint is specified via an analog output.

Options for connecting

Drives with PROFIdrive capability are connected via the PROFINET interface of the CPU.

Drives with analog setpoint interface are connected with the CPU via one of the following connections:

- Analog output via signal board
- Analog output on-board
- Analog output via analog output module

The following connection options are available for an encoder:

- Encoder on the PROFIdrive drive
- Encoder on technology module

- PROFIdrive encoder directly on PROFINET IO
  (With these encoders, the encoder value is always transferred via PROFIdrive telegrams per PROFIBUS or PROFINET.)
- Encoder on high speed counter (HSC)
  With this connection option, the encoder signals are connected directly to an HSC, which forms the encoder values from them. Depending on the CPU in use, up to 6 HSC encoders can be used.

Maximum number of axes

You can control up to a maximum of 8 drives via PROFIdrive or the analog drive connection (the number is independent of the simulation status of the axis).

Automatic transfer of drive and encoder parameters in the device (S7-1200)

For operation, identical reference values for the drive and encoder connections must be set in the controller and in the drive and encoder.

The speed setpoint NSET and the actual speed value NACT are transferred in the PROFIdrive telegram as a percentage of the reference speed. The reference value for the speed must be set identically in the controller and in the drive.

The resolution of the actual value in the PROFIdrive telegram must likewise be set identically in the controller and in the drive and encoder modules

Automatic transfer of parameters

For SINAMICS drives as of V4.x and PROFIdrive encoders as of product version A16, the drive and encoder parameters can be automatically transferred to the CPU.

The corresponding parameters are transferred after the (re)initialization of the technology object and (re)start of the drives and the CPU. Changes in the drive configuration are transferred after restart of the drive or technology object.

The successful transfer of the parameters can be verified in the controller in the variables of the <TO>.StatusDrive.AdaptionState = 1 and <TO>.StatusSensor[1].AdaptionState = 1 technology objects.

Parameters

The controller settings are made in the TIA Portal under "Technology object > Configuration > Basic parameters > Drive/encoder".

The drive and encoder settings are made in the configuration or the respective hardware.

The following table compares the settings in the TIA Portal and the corresponding drive/ encoder parameters:

Setting in the TIA Portal	Controller variable in the technology data block	Drive parameter	Automatic transfer
Drive			
Telegram number	Input address telegram <TO>.Actor.Interface.AddressIn	Telegram number P922	-
	Output address telegram <TO>.Actor.Interface.AddressOut		
Reference speed in [1/min]	<TO>.Actor.DriveParameter.Reference-Speed	SINAMICS drives: P2000	X
Maximum speed of motor in [1/min]	<TO>.Actor.DriveParameter.MaxSpeed	SINAMICS drives: P1082	X
Drive	<TO>.Actor.Type 0 = analog drive connection 1 = PROFIdrive 2 = PTO (pulse train output)	-	-
Encoder			
Telegram	<TO>.Sensor[n].Interface.AddressIn	P922	-
	<TO>.Sensor[n].Interface.Addressout		
Encoder type • Linear incremental • Linear absolute • Rotary incremental • Rotary absolute	<TO>.Sensor[n].System 0: Rotary 1: Linear	P979[1] Bit0 Encoder 1 P979[11] Bit0 Encoder 2	X
	<TO>.Sensor[n].Type 0: Incremental 1: Absolute	P979[5] Encoder 1 P979[15] Encoder 2	-
Resolution, linear encoder The grid spacing is specified on the name-plate of the encoder as a separation distance of the marks on the linear measuring system.	<TO>.Sensor[n].Parameter.Resolution	P979[2] Encoder 1 P979[12] Encoder 2	X
Increments per revolution, rotary encoder	<TO>.Sensor[n].Parameter.StepsPerRe-volution	P979[2] Encoder 1 P979[12] Encoder 2	X
Number of bits for fine resolution XIST1 Cyclic actual encoder value, linear or rotary encoder	<TO>.Sensor[n].Parameter.FineResolu-tionXist1	P979[3] Encoder 1 P979[13] Encoder 2	X
Number of bits for fine resolution XIST2 Absolute value of the encoder, linear or rotary encoder	<TO>.Sensor[n].Parameter.FineResolu-tionXist2	P979[4] Encoder 1 P979[14] Encoder 2	X
Differentiable encoder revolutions, rotary absolute encoder	<TO>.Sensor[n].Parameter.Determina-bleRevolutions	P979[5] Encoder 1 P979[15] Encoder 2	X

See also

Configuration - Drive - PTO (Pulse Train Output) (Page 1345)

Configuration - Drive - Analog drive connection (Page 1348)

Configuration - Drive - PROFIdrive (Page 1349)

Configuration - Encoder - Encoder on PROFINET/PROFIBUS (Page 1351)

Configuration - Encoder - Encoder on high-speed counter (HSC) (Page 1353)

PROFIdrive (S7-1200)

PROFIdrive is the standardized standard profile for drive technology in the connection of drives and encoders via PROFINET IO. Drives and encoders that support the PROFIdrive profile are connected according to the PROFIdrive standard.

You can find the current PROFIdrive specification at:

http://www.profibus.com (http://www.profibus.com)

Communication between controller and drive/encoder is by means of various PROFIdrive telegrams. Each of the telegrams has a standardized structure. Depending on the application, you can select the applicable telegram. Control words and status words as well as setpoints and actual values are transmitted in the PROFIdrive telegrams.

Telegrams for PROFIdrive

The setpoint of a positioning axis is transferred to a drive via PROFIdrive telegram 1, 2 3 or 4.

The encoder value is transmitted either in a telegram together with the setpoint (telegram 3 and 4), or in a separate encoder telegram (telegram 81 or telegram 83).

The following table shows the supported PROFIdrive telegrams for the assignment of drives and encoders:

Standard tele-grams	Brief description
1	• 16 bit speed setpoint (NSET) • 16 bit actual speed (NACT)
2	• 32 bit speed setpoint (NSET) • 32-bit actual speed (NACT) • Signs of life
3	• 32 bit speed setpoint (NSET) • 32-bit actual speed (NACT) • 1 encoder • Signs of life
4	• 32 bit speed setpoint (NSET) • 32-bit actual speed (NACT) • 2 encoders • Signs of life

Standard telegrams for encoder	Brief description
81	• 1 encoder • Signs of life
83	• 32-bit actual speed (NACT) • 1 encoder • Signs of life

Closed loop control (S7-1200)

Drives which are connected via PROFIdrive or an analog drive interface are generally operated under position control. If service is required, the axis can also be operated without position control.

The position controller is a P controller with precontrol of velocity.

Controller structure

The following figure shows the controller structure of an S7-1200 Motion Control:

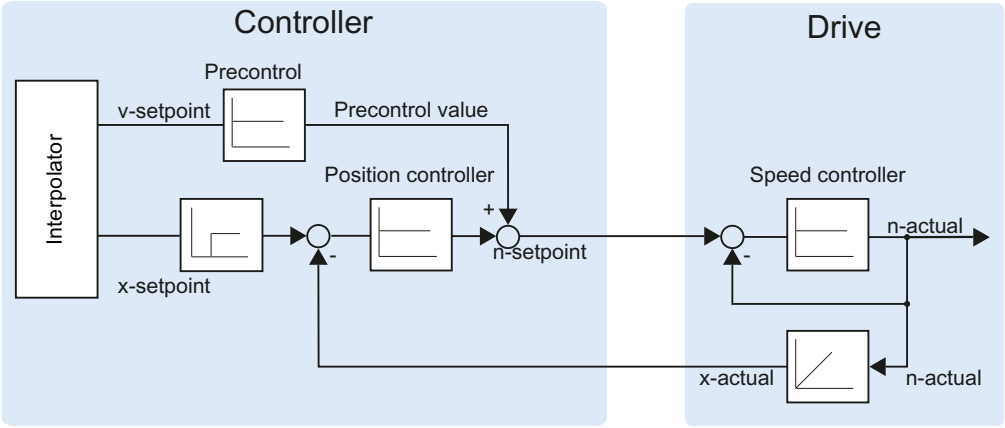

The MC-Interpolator [OB92] calculates the setpoint position for the axis. The difference between the setpoint and actual position is multiplied by the gain factor of the position controller. The resulting value is added to the precontrol value and output as setpoint speed to the drive via PROFIdrive or analog output.

The encoder records the actual position of the axis and returns it to the controller via a PROFIdrive telegram or an HSC (high speed counter).

Data connection PROFIdrive drive/PROFIdrive encoder (S7-1200)

The data connection of PROFIdrive drives and PROFIdrive encoders occurs either directly via the PROFIdrive telegram or via a data block.

Use the connection via data block if you want to influence or evaluate telegram contents in the user program for process-specific reasons.

Principle of data connection directly to the drive/encoder

The following simplified function chart shows the direct data connection to PROFIdrive drives and PROFIdrive encoders by means of telegrams:

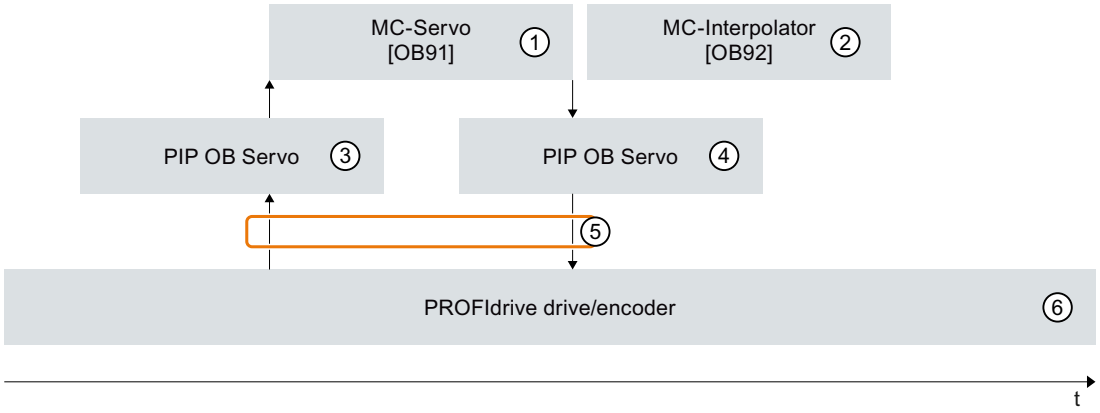

① The organization block "MC-Servo" calculates the position controller.

At the start of "MC-Servo", the input telegram of the drive or encoder is read (⑥ -> ⑤ -> ③). If an organization block "MC-PreServo" was added, the telegram is read at the beginning of "MC-PreServo".

At the end of "MC-Servo", the output telegram is written to the drive or encoder (⑥ -> ⑤ -> ③). If an organization block "MC-PostServo" was added, the telegram is written at the end of "MC-PostServo".

② In every Motion application cycle, the organization block "MC-Interpolator" is called after the "MC-Servo".

In "MC-Interpolator", the Motion Control instructions are evaluated, setpoints are generated for the next Motion application cycle and the technology object is monitored.

③ The process image partition "PIP OB servo" of the inputs is updated in the Motion application cycle.

④ The process image partition "PIP OB servo" of the outputs is updated in the Motion application cycle.

⑤ Telegram exchange via the I/O addresses of the controller and the drive or the encoder.

⑥ PROFIdrive drive or PROFIdrive encoder

Principle of data connection via data block

The following simplified function chart shows the data connection to PROFIdrive drives and PROFIdrive encoders via data block: The following sections provide details on the execution.

To influence or evaluate telegram contents for process-specific reasons, a data block must be connected in between as a data interface (see ① and ③).

To do this, use the organization blocks "MC-PreServo" and "MC-PostServo" to achieve a high quality of position control.

The organization blocks "MC-PreServo" and "MC-PostServo" can be added in the project tree with the command "Add new block".

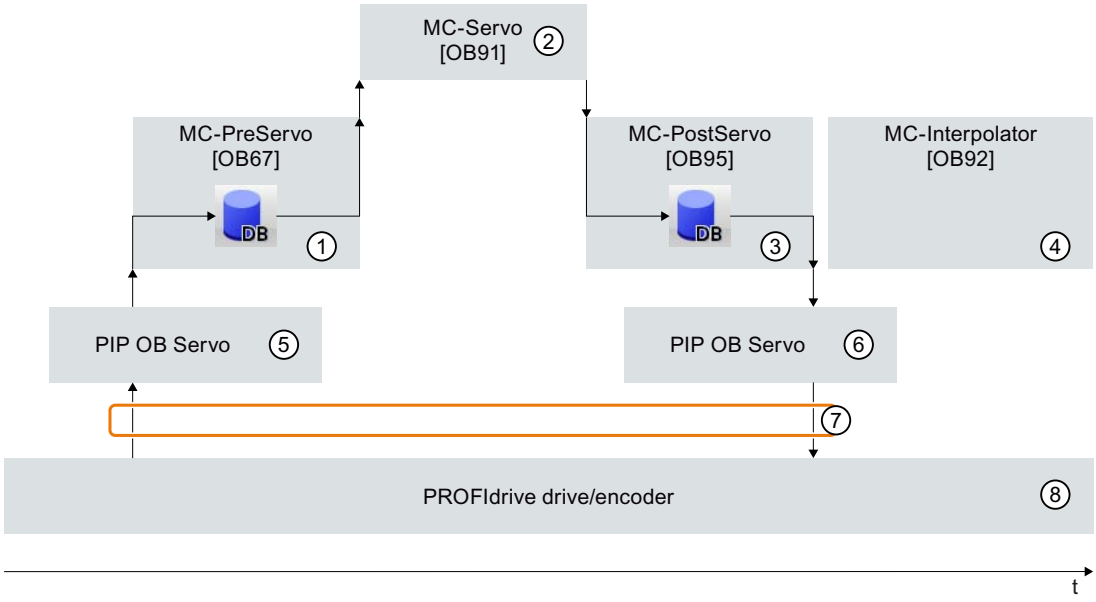

① The "MC-PreServo" is called before the "MC-Servo".

In the user program of the "MC-PreServo", transfer the content of the input telegram from the process image partition "PIP OB servo" ⑤ to the data block of the data interface.

In the other user program of the "MC-PreServo", the input area of the telegram can be processed or evaluated.

② The organization block "MC-Servo" calculates the position controller.

At the start of "MC-Servo", the input telegram of the drive or encoder is read from the data block of the data interface (① -> ②).

At the end of "MC-Servo", the output telegram of the drive or encoder is written to the data block of the data interface (② -> ③).

③ The "MC-PostServo" is called after the "MC-Servo".

In the user program of the "MC-PostServo", the output area of the telegram can be processed or evaluated.

At the end of the user program of the "MC-PostServo", transfer the content of the output telegram from the data interface of the data block to the process image partition "PIP OB servo" ⑥.

④ In every Motion application cycle, the organization block "MC-Interpolator" is called after the "MC-PostServo".

In "MC-Interpolator", the Motion Control instructions are evaluated, setpoints are generated for the next Motion application cycle and the technology object is monitored.

⑤ The process image partition "PIP OB servo" of the inputs is updated in the Motion application cycle.

⑥ The process image partition "PIP OB servo" of the outputs is updated in the Motion application cycle.

⑦ Telegram exchange via the I/O addresses of the controller and the drive or the encoder.

⑧ PROFIdrive drive or PROFIdrive encoder

Basic procedure for the data connection via a data block

To use the data connection via the data block, follow the steps described below. The data connection can be configured separately for a PROFIdrive drive and encoder.

Creating the data block for data connection

The data block for the data connection must be created by the user. The data block must contain a data structure of data type "PD_TELx" for the data connection. Here, "x" stands for the telegram number of the drive or encoder configured in the device configuration.

To create the data block, follow the steps described below:

1. Create a new data block of type "Global DB".
2. Select the data block in the project tree and select the shortcut menu command "Properties".
3. Under Attributes, disable the following attributes and confirm with OK
   "Only store in load memory"
   "Data block write-protected in the device"
   "Optimized block access"
4. Open the data block in the block editor.
5. Insert a "PD_TELx" type variable in the block editor.
6. Compile the data block for the data connection before you use it in the configuration of the axes.

This variable contains the "Input" variable structure for the input area of the telegram and the "Output" variable structure for the output area of the telegram.

Note

"Input" and "Output" relate to the view of the closed loop position control. For example, the input area contains the actual values of the drive and the output area contains the setpoints for the drive.

The data block may contain the data structures of multiple axes and encoders and other contents.

Configuring data connection via a data block

Proceed as decribed below for the configuration of the axis:

1. Open the configuration window "Hardware interface > Drive" or "Hardware interface > Encoder".
2. In the Data block drop-down list, select "Data block".
3. In the "Data block" field, select the previously created data block.
   Open this data block and select the tag name defined for the drive or encoder.

Adding a PLC variable for telegram access

To enable access to the input and output areas of the telegram, create the following PLC variable.

For the PLC variable of the input area, follow the steps described below:

1. Open the "PLC variables" folder in the project tree and have all variables displayed.
2. Add a new variable and assign a unique name, made up for example from the name of the axis or the encoder, the telegram type and the address area.
3. Enter the type "PD_TELx_IN" textually in the "Data type" column.
4. Enter the telegram input address of the drive/encoder in the "Address" column.
   You can find the address in the device configuration of the drive or encoder.

Follow the same procedure for the PLC variable of the output area and select "PD_TELx_OUT" as the data type and the telegram output address of the drive/encoder as the address.

Programming MC-PreServo and MC-PostServo

MC-PreServo

The user program of the "MC-PreServo" must read the input area of the telegram and transfer it to the data block of the data connection.

Assign the previously defined PLC tag of the input area to the tag structure "Input" of the data block in the "MC-PreServo" user program.

With further instructions, you can edit the data of the tag structure "Input" of the data block before it is then transferred to "MC-Servo" and processed in "MC_Servo" .

MC-PostServo

After it has been processed, "MC-Servo" transfers the output area of the telegram to the tag structure "Output" of the data block.

The content of the tag structure "Output" of the data block must be written to the telegram output address in the "MC-PostServo" user program.

Assign the "Output" tag structure of the data block to the previously defined PLC tag of the output area in the "MC-PostServo" user program.

If the output area is to be modified, this must be done before the assignment instruction.

NOTICE
Machine damage Improper manipulation of drive and encoder telegrams may result in unwanted movements of the drive. Check your user program in regard to consistency in the drive and encoder connection.

You can find an application example for the use of MC-PreServo and MC-PostServo at:

https://support.industry.siemens.com/cs/document/109741575 (https://support.industry.siemens.com/cs/document/109741575)

See also

PROFIdrive frame (Page 1452)

Configuration - Drive - PROFIdrive (Page 1349)

Configuration - Encoder - Encoder on PROFINET/PROFIBUS (Page 1351)

Data connection drive with analog drive connection (Page 1316)

Organization Blocks for Motion Control (Page 1318)

Data connection drive with analog drive connection (S7-1200)

The data connection of drives with analog drive interface can alternatively be made via a data block.

Use the connection via a data block if you want to adapt the analog setpoint in the user program for process-related reasons.

Principle of data connection via data block

At the end of position control by MC-Servo [OB91], the setpoint of the analog drive is written to the assigned analog output.

To adapt the analog setpoint for process-related reasons, a data interface via a data block must be connected in between.

The setpoint of the analog drive can be edited via the MC-PostServo [OB95] organization block in the data block and can then be written to the I/O address.

The MC-PostServo is called after the MC-Servo. The organization block MC-PostServo can be programmed by the user and must be added with the command "Add new block".

The procedure in principle

To use the data connection via the data block, follow the steps described below. The data connection can be configured separately for drives with analog drive interface and PROFIdrive encoder. You can find information on data connection of the PROFIdrive encoders in the section Data connection PROFIdrive drive/PROFIdrive encoder (Page 1311).

Creating the data block for data connection

The data block must be created on the user side.

To create the data block, follow the steps described below:

1. Create a new data block of type "Global DB".
2. Select the data block in the project tree and select "Properties" from the shortcut menu.
3. Under Attributes, disable the following attributes and confirm with OK
   "Only store in load memory"
   "Data block write-protected in the device"
   "Optimized block access"
4. Open the data block in the block editor.

5. Insert a variable of the "WORD" data type in the block editor.
6. Compile the data block for the data connection before you use it in the configuration of the axes.

Configuring data connection via a data block

Proceed in the configuration as described below ("Analog drive connection" must be selected in the Basic parameters > General configuration window):

1. Open the configuration window Basic parameters > Drive.
2. Select the previously defined variable of the data block in the "Analog output" box.

Set analog output address in the TPA OB Servo process image

To achieve a good level of control quality, the address area of the analog output must be within the process image "TPA OB Servo".

Proceed as described below:

1. Open the module of the analog output in the device configuration.
2. Open the "General" tab
3. Select the "I/O addresses"
4. Select the organization block "MC-Servo". "TPA OB Servo" is automatically selected as the process image.
5. Assign a variable name to the analog output in the "I/O variables" tab.

Program MC-PostServo

Assign the variable of the data block to the variable of the analog output in the MC-PostServouser program.

At the end of MC-PostServo, the output area of "TPA OB Servo" is written to the I/O.

⚠ WARNING
Improper manipulation of the drive setpoint may endanger humans and machines. Take adequate precautions to prevent danger to humans and machines.

Process response (S7-1200)

Organization Blocks for Motion Control (S7-1200)

Description

When you create a "Positioning axis" technology object with a PROFIdrive drive or with an analog drive interface, organization blocks for processing the technology objects are created automatically. The Motion Control functionality of the technology objects creates its own execution level, and is called according to the Motion Control application cycle.

The following blocks are created:

- MC-Servo [OB91]
  Calculation of the Position Controller
- MC-Interpolator [OB92]
  Evaluation of the motion control instructions, generation of setpoints and monitoring functionality

The organization blocks are protected (know-how protection). The program code cannot be viewed or changed.

The frequency relationship of the two organization blocks to one another is always 1:1. MC-Servo [OB91] is always executed before MC-Interpolator [OB92].

You can set the Motion Control application cycle and the priority of the organization blocks according to your requirements for control quality and system load.

Motion Control application cycle

You can set the Motion Control application cycle, in which the MC-Servo [OB91] is called, in the properties of the organization block in "General > Cycle Time".

The MC-Servo [OB91] is called cyclically with the specified "application cycle".

The selected Motion Control application cycle must be long enough to be able to process all technology objects in one cycle. If the processing time of the technology objects is longer than the application cycle, overflows (Page 1319) will occur.

To avoid disruptions in the program execution on the CPU, set the Motion Control application cycle depending on the number of axes used as follows:

Motion Control application cycle = (number of position-controlled axes × 2 ms) + 2 ms

The following table shows the resulting Motion Control application cycle as an example according to the number of position-controlled axes:

Number of axes	Motion Control application cycle
1	4 ms
2	6 ms
4	10 ms
8	18 ms

For SINAMICS, the following should continue to apply:

- Motion Control application cycle (MC-Servo) ≥ SINAMICS drive process image (parameter P2048) ≥ bus clock cycle

All times should be selected as integral multiples of one another.

Priority

You can configure the priority of the organization blocks as needed in their properties under "General > Properties > Priority":

- MC-Servo [OB91]
  Priority 17 to 26 (default value 25)
- MC-Interpolator [OB92]
  Priority 17 to 26 (default value 24)

The priority of MC-Servo [OB91] must be at least one higher than the priority of the MC-Interpolator [OB92].

MC-PreServo [OB67] and MC-PostServo [OB95]

Organization blocks MC-PreServo [OB67] and MC-PostServo [OB95] are programmable and are called in the configured application cycle. MC-PreServo [OB67] is called directly before MC-Servo [OB91]. MC-PostServo [OB95] is called directly after MC-Servo [OB91].

See also

Data connection PROFIdrive drive/PROFIdrive encoder (Page 1311)

Process image partition "PIP OB Servo" (S7-1200)

For optimal control, assign all I/O modules used by Motion Control (e.g. drives, technology modules, digital and analog input/output modules) to the process image partition "PIP OB servo". The assignment causes the I/O modules to be processed simultaneously with the technology object.

Operational Sequence and Timeouts (S7-1200)

When processing the Motion Control functionality, the organization blocks MC-Servo [OB91] and MC-Interpolator [OB92] are called and processed in each application cycle (processing also occurs in the STOP operating state of the CPU). The remaining cycle time is available for the processing of your user program.

For error-free program execution, keep to the following rules:

- In each application cycle, MC-Servo [OB91] must be started and executed completely.
- In every application cycle, the relevant MC-Interpolator [OB92] must at least be started.

The following figure shows an example of the error-free operational sequence for the processing of organization block OB1:

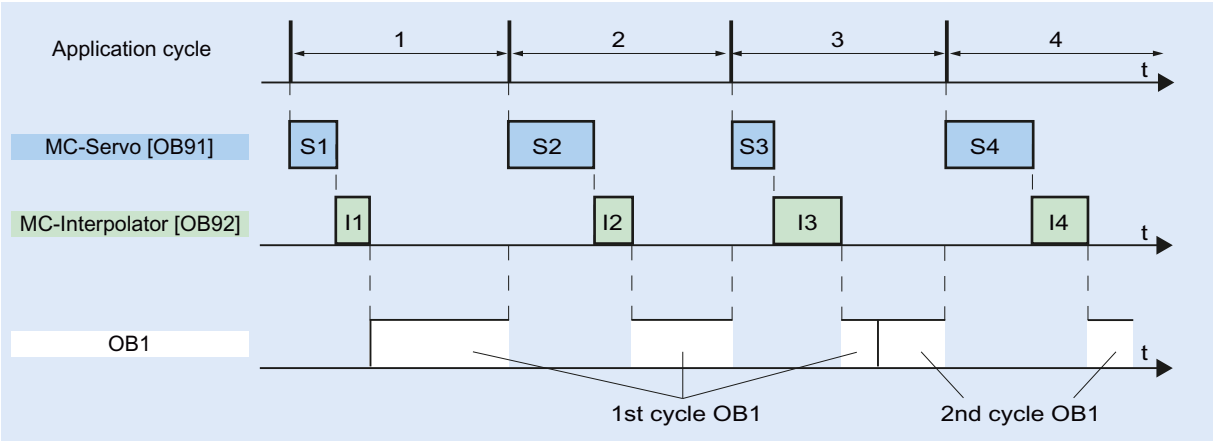

Overflows

If the set application cycle is not adhered to, for example because the application cycle is too short, overflows can occur.

An overflow of the MC-Servo [OB91], MC-Interpolator [OB92], MC_PreServo [OB67] and MC_PostServo [OB95] is entered in the diagnostic buffer of the CPU and results in setting the CPU to STOP.

MC-PreServo, MC-Servo, MC-PostServo and MC-Interpolator are stopped. If necessary, you can evaluate the entry in the diagnostic buffer via a time error OB (OB80).

Operating modes (S7-1200)

This section examines the behavior of Motion Control in each operating mode, and in the transitions between operating modes. A general description of the operating modes can be found in system manual S7-1200.

Operating modes and transitions

The CPU has three operating modes: STOP, STARTUP and RUN. The following figure shows the operating modes and the operating mode transitions:

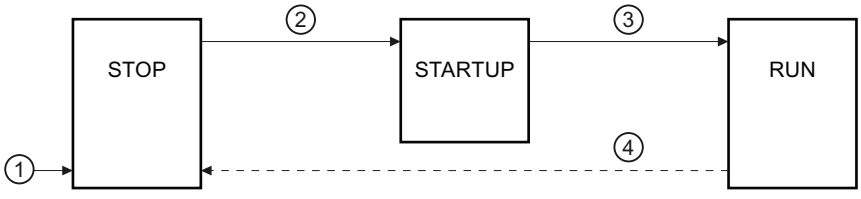

① POWER ON → STOP
② STOP → STARTUP
③ STARTUP → RUN
④ RUN → STOP

STOP mode

In STOP mode the user program is not processed and all process outputs are disabled. Thus no Motion Control jobs are executed.

The technology data blocks of the position-controlled axes are updated.

STARTUP mode

Before the CPU starts processing of the cyclical user program, the startup OBs are processed one time.

In STARTUP mode, the process outputs are disabled. Motion Control jobs are rejected.

The technology data blocks of the position-controlled axes are updated.

RUN mode

The user program is processed in RUN mode.

In RUN mode, the Motion Control commands programmed in OB1 are called and processed cyclically (other execution levels are possible).

The technology data blocks are updated.

Operating mode transitions

The following table shows the behavior of Motion Control in the transitions between the operating modes:

Operating mode transition	Behavior
POWER ON → STOP	The CPU performs a restart of the technology objects. The technology objects are reinitialized with the values from the load memory.
STOP → STARTUP	The technology objects are initialized with the start values of the CPU.
STARTUP → RUN	The process outputs are enabled.
RUN → STOP	When the CPU changes to RUN mode after STOP mode, all technology objects are disabled in accordance with the error response "remove enablement". Running Motion Control jobs are terminated.

Hardware and software limit switches (S7-1200)

Use the hardware and software limit switches to limit the "permitted traversing range" and the "working range" of your positioning axis technology object. The relationships are shown in the following diagram:

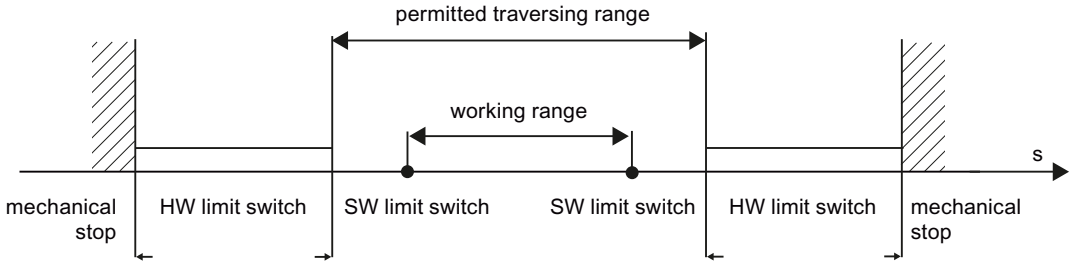

Hardware limit switches are limit switches that limit the maximum "permitted traversing range" of the axis. Hardware limit switches are physical switching elements that must be connected to interrupt-capable inputs of the CPU.

Software limit switches limit the "working range" of the axis. They should fall inside the hardware limit switches relative to the traversing range. Since the positions of the software limit switches can be flexibly set, the working range of the axis can be adapted on an individual basis, depending on the current traversing profile. In contrast to hardware limit switches, software limit switches are implemented exclusively via the software and do not require their own switching elements.

Hardware and software limit switches must be activated prior to use in the configuration or in the user program. Software limit switches are only active after homing the axis.

See also

CPU outputs relevant for motion control (Page 1300)

How the pulse interface works (Page 1303)

Relationship between the signal type and the direction of travel (Page 1303)

Jerk limit (Page 1323)

Homing (Page 1324)

Integration of the positioning axis technology object (Page 1336)

Tools of the positioning axis technology object (Page 1339)

Position limits (Page 1357)

Jerk limit (S7-1200)

With the jerk limit you can reduce the stresses on your mechanics during an acceleration and deceleration ramp. The acceleration and deceleration value is not changed abruptly when the jerk limiter is active; it is gradually increased and decreased. The figure below shows the velocity and acceleration curve without and with jerk limit.

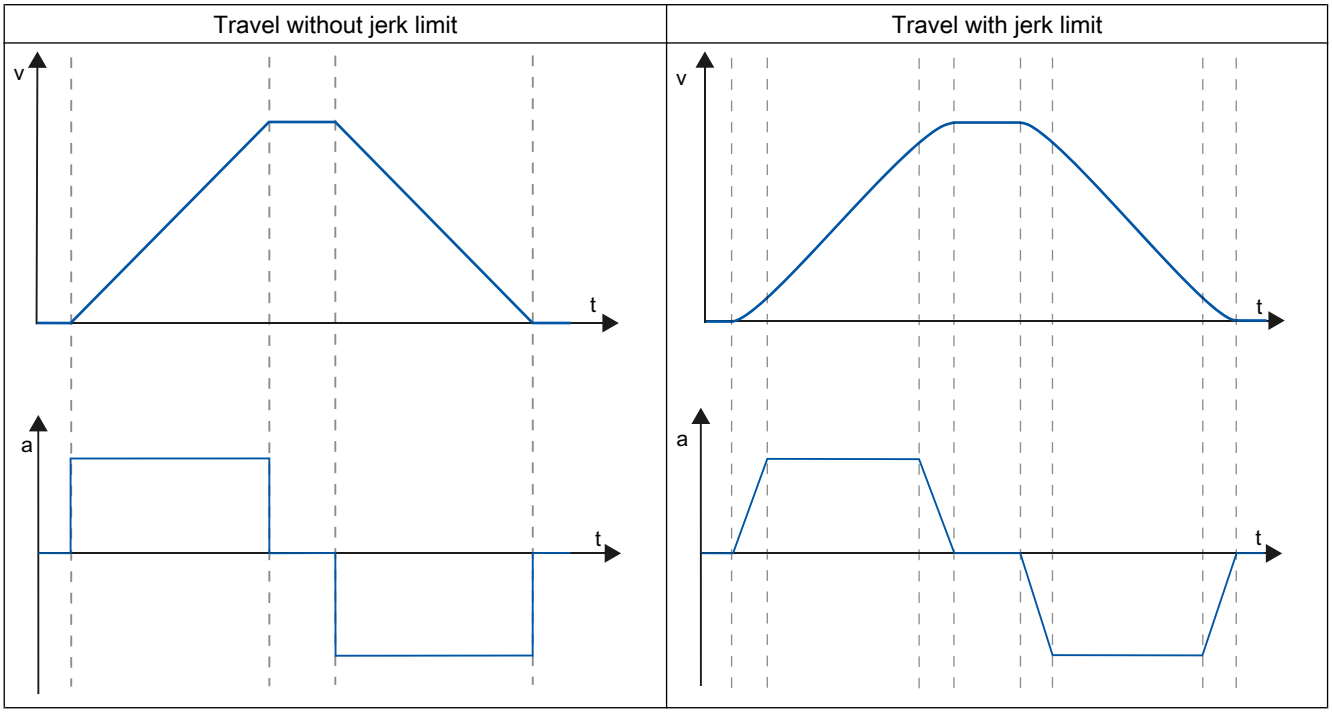

The jerk limit gives a "smoothed" velocity profile of the axis motion. This ensures, for example, soft starting and braking of a conveyor belt.

See also

Behavior of the axis when using the jerk limit (Page 1367)

CPU outputs relevant for motion control (Page 1300)

How the pulse interface works (Page 1303)

Relationship between the signal type and the direction of travel (Page 1303)

Hardware and software limit switches (Page 1322)

Homing (Page 1324)

Integration of the positioning axis technology object (Page 1336)

Tools of the positioning axis technology object (Page 1339)

Homing (S7-1200)

Homing means matching the axis coordinates of the technology object to the real, physical location of the drive. For position-controlled axes the entries and displays for the position refer exactly to these axis coordinates. Therefore, agreement between the axis coordinates and the real situation is extremely important. This step is necessary to ensure that the absolute target position of the axis is also achieved exactly with the drive.

In the S7-1200 CPU, axis homing is implemented with the motion control instruction, "MC_Home". The "Homed" status is displayed in the tags of the technology object <Axis name>.StatusBits.HomingDone . The following homing modes exist:

Homing modes

- Active homing
  In active homing mode, the motion control instruction "MC_Home" performs the required reference point approach. When the homing switch is detected, the axis is homed according to the configuration. Active traversing motions are aborted.
- Passive homing
  During passive homing, the "MC_Home" Motion Control instruction does not carry out any homing motion. The traversing motion required for this must be implemented by the user via other Motion Control instructions. When the homing switch is detected, the axis is homed according to the configuration. Active traversing motions are not aborted upon start of passive homing.
- Direct homing absolute
  The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The value of input parameter "Position" of motion control instruction "MC_Home" is set immediately as the reference point of the axis.
- Direct homing relative
  The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The following statement applies to the axis position after homing:
  New axis position = current axis position + value of parameter "Position" of instruction "MC_Home".

Resetting the "Homed" status

The "Homed" status of a technology object (<Axis name>.StatusBits.HomingDone) is reset under the following conditions:

- Drive connection via PTO (Pulse Train Output):
  - Start an "MC_Home" command for active homing
    (After successful completion of the homing operation, the "Homed" status is set again.)
  - Disabling of axis by the "MC_Power" Motion Control instruction
  - Changeover between automatic mode and manual control
  - After POWER OFF -> POWER ON of the CPU
  - After CPU restart (RUN-STOP -> STOP-RUN)
- Technology objects with incremental actual values:
  - Start an "MC_Home" command for active homing
    (After successful completion of the homing operation, the "Homed" status is set again.)
  - Error in the encoder system, or encoder failure
  - Restart of the technology object
  - After POWER OFF → POWER ON of the CPU
  - Memory reset
  - Modification of the encoder configuration
- Technology objects with absolute actual values:
  - Errors in the sensor system/encoder failure
  - Replacement of the CPU
  - Modification of the encoder configuration
  - Restoration of the CPU factory settings
  - Transfer of a different project to the controller

See also

CPU outputs relevant for motion control (Page 1300)

How the pulse interface works (Page 1303)

Relationship between the signal type and the direction of travel (Page 1303)

Hardware and software limit switches (Page 1322)

Jerk limit (Page 1323)

Integration of the positioning axis technology object (Page 1336)

Tools of the positioning axis technology object (Page 1339)

Homing (positioning axis technology object as of V2) (Page 1369)

6.1.1.3 Guidelines on use of motion control (S7-1200)

The guidelines described here present the basic procedure for using motion control with the CPU S7-1200.

Requirements

To use the positioning axis technology object, a project with a CPU S7-1200 must be created.

Procedure

Follow the steps below in the order given to use motion control with the CPU S7-1200. Use the following links for this purpose:

1. Adding a positioning axis technology object (Page 1341)
2. Working with the configuration dialog (Page 1342)
3. Download to CPU (Page 1420)
4. Function test of the axis in the commissioning window (Page 1421)
5. Programming (Page 1427)
6. Diagnostics of the axis control (Page 1448)

6.1.1.4 Using versions (S7-1200)

Overview of versions (S7-1200)

The relationship between the relevant versions for S7-1200 Motion Control can be found in the following table:

Technology version

You can check the currently selected technology version in the "Instructions" task card in the folder "Technology > Motion Control" and in the "Technology object > Add new object" dialog.

Select the technology version in the "Instructions" task card in the folder "Technology > Motion Control".

If a technology object with an alternative version is added in the "Add new object" dialog, the technology version will also be changed.

Note

The selection of an alternative technology version will also affect the Motion Control Instructions version (task card).

The technology objects and Motion Control instructions will only be converted to the selected version upon compilation or "Download to device".

Version of the technology object

The version of a technology object can be checked in the inspector window under "Properties > General > Information" in the "Version" box.

Motion Control instruction version

The Motion Control instruction version can be checked in the inspector window under "Properties > General > Information" in the "Version" box.

If the Motion Control instruction version used is not in line with the following compatibility list, the relevant Motion Control instructions will be highlighted in the program editor.

Compatibility list

Technology		CPU	Technology object	Motion Control instruction
V6.0	Innovations: • MC-PreServo • MC-PostServo • Data connection directly to the SINAMICS drive or via data block • Data connection directly to the analog output of a drive with analog drive connection or via a data block • Transfer of drive and encoder parameters in the device for PROFIdrive drives and encoders. • Move position-controlled drives without position control for servicing purposes • Simulation of position-controlled drives without configured or existing hardware • Level selection when configuring the hardware limit switch • Support PROFIdrive Telegram 4	V4.2	Positioning axis V6.0 Command table V6.0	MC_Power V6.0 MC_Reset V6.0 MC_Home V6.0 MC_Halt V6.0 MC_MoveAbsolute V6.0 MC_MoveRelative V6.0 MC_MoveVelocity V6.0 MC_MoveJog V6.0 MC_CommandTable V6.0 MC_ChangeDynamic V6.0 MC_ReadParam V6.0 MC_WriteParam V6.0

Technology		CPU	Technology object	Motion Control instruction
V5.0	Innovations: • Drive connection via PROFIdrive • Analog drive connection • Position control for PROFIdrive / analog drive connection • Position monitoring for PROFIdrive / analog drive connection • MC-Servo [OB91] • MC-Interpolator [OB92]	V4.1	Positioning axis V5.0 Command table V5.0	MC_Power V5.0 MC_Reset V5.0 MC_Home V5.0 MC_Halt V5.0 MC_MoveAbsolute V5.0 MC_MoveRelative V5.0 MC_MoveVelocity V5.0 MC_MoveJog V5.0 MC_CommandTable V5.0 MC_ChangeDynamic V5.0 MC_ReadParam V5.0 MC_WriteParam V5.0
V4.0	Innovations: • MC_ReadParam • MC_WriteParam • Standardization of S7-1200 and S7-1500 Motion Control technology data blocks.	V4.0	Positioning Axis V4.0 Command table V4.0	MC_Power V4.0 MC_Reset V4.0 MC_Home V4.0 MC_Halt V4.0 MC_MoveAbsolute V4.0 MC_MoveRelative V4.0 MC_MoveVelocity V4.0 MC_MoveJog V4.0 MC_CommandTable V4.0 MC_ChangeDynamic V4.0 MC_ReadParam V4.0 MC_WriteParam V4.0
V3.0	Innovation: Load in RUN mode	V2.2 V3.0 V4.0	Axis V3.0 Command table V3.0	MC_Power V3.0 MC_Reset V3.0 MC_Home V3.0 MC_Stop V3.0 MC_MoveAbsolute V3.0 MC_MoveRelative V3.0 MC_MoveVelocity V3.0 MC_MoveJog V3.0 MC_CommandTable V3.0 MC_ChangeDynamic V3.0

Technology		CPU	Technology object	Motion Control instruction
V2.0	Innovations: • Jerk limit • Command table • MC_ChangeDynamic	V2.1 V2.2 V3.0	Axis V2.0 Command table V2.0	MC_Power V2.0 MC_Reset V2.0 MC_Home V2.0 MC_Halt V2.0 MC_MoveAbsolute V2.0 MC_MoveRelative V2.0 MC_MoveVelocity V2.0 MC_MoveJog V2.0 MC_CommandTable V2.0 MC_ChangeDynamic V2.0
V1.0		V1.0 V2.0 V2.1 V2.2 V3.0	Axis V1.0	MC_Power V1.0 MC_Reset V1.0 MC_Home V1.0 MC_Halt V1.0 MC_MoveAbsolute V1.0 MC_MoveRelative V1.0 MC_MoveVelocity V1.0 MC_MoveJog V1.0

See also

Changing a technology version (Page 1330)

Compatibility list of variables V1...3 <-> V4...5 (Page 1331)

Status of limit switch (Page 1335)

Changing a technology version (S7-1200)

Before you can access all the benefits of a new technology version, you may need to set / modify the technology version for existing projects.

Note

Compatibility of the technology object tags

When switching between V1...3 and ≥ V4, please see the compatibility list (Page 1331) when using tags of the technology object in the user program, monitoring tables, etc .

Setting/changing a technology version

To set or change the technology version, follow these steps:

1. Open the program editor (e.g., by opening the OB1).
2. In the "Instructions" task card, select the desired technology version in the "Technology > Motion Control" folder.
3. Save and compile the project. Pay attention to any error information that is displayed during compilation. Deal with the causes of the errors indicated.
4. Check the configuration of the technology objects.
5. If necessary, adapt the tag names in the following objects in line with the compatibility list.

- User program
- Watch tables
- Force tables
- HMI configuration
- Trace configuration

See also

Overview of versions (Page 1326)

Status of limit switch (Page 1335)

Compatibility list of variables V1...3 <-> V4...5 (S7-1200)

The technology data blocks for S7-1200 Motion Control and S7-1500 Motion Control have been standardized within the framework of the V4 technology. As of V4, this has resulted in new tags and tag names for the positioning axis and command table technology objects.

Observe the information in the following tables if you have used tags of the technology objects in the user program and you want to convert the project from V1...3 to V4 or higher (or vice versa).

The tags listed in the "Automatic conversion V1...3 to ≥ V4" column are converted automatically when the project is compiled. Tag names in monitoring and force tables or the HMI or trace configuration are not converted.

The following tags are new or have been adapted and may have to be corrected in the user program, watch tables, etc.:

Config tags (positioning axis)

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Axis name>.Config.DynamicDefaults.Acceleration	<Axis name>.DynamicDefaults.Acceleration	Yes
<Axis name>.Config.DynamicDefaults.Deceleration	<Axis name>.DynamicDefaults.Deceleration	Yes
<Axis name>.Config.DynamicDefaults.EmergencyDeceleration	<Axis name>.DynamicDefaults.EmergencyDeceleration	Yes
<Axis name>.Config.DynamicDefaults.Jerk	<Axis name>.DynamicDefaults.Jerk	Yes
<Axis name>.Config.DynamicDefaults.JerkActive	Not available The jerk is activated if the configured jerk is > 0.004 pulse/s³.	No
<Axis name>.Config.DynamicLimits.MaxVelocity	<Axis name>.DynamicLimits.MaxVelocity	Yes
<Axis name>.Config.DynamicLimits.MinVelocity	<Axis name>.DynamicLimits.MinVelocity	Yes
<Axis name>.Config.General.LengthUnit	<Axis name>.Units.LengthUnit	Yes
<Axis name>.Config.Homing.AutoReversal	<Axis name>.Homing.AutoReversal	Yes
<Axis name>.Config.Homing.Direction	<Axis name>.Homing.ApproachDirection	Yes
<Axis name>.Config.Homing.FastVelocity	<Axis name>.Homing.ApproachVelocity	Yes
<Axis name>.Config.Homing.Offset	<Axis name>.Sensor[1].ActiveHoming.HomePositionOffset	Yes
<Axis name>.Config.Homing.SideActiveHoming	<Axis name>.Sensor[1].ActiveHoming.SideInput	Yes
<Axis name>.Config.Homing.SidePassiveHoming	<Axis name>.Sensor[1].PassiveHoming.SideInput	Yes
<Axis name>.Config.Homing.SlowVelocity	<Axis name>.Homing.ReferencingVelocity	Yes
<Axis name>.Config.Homing.SwitchedLevel	<Axis name>.Sensor[1].ActiveHoming.SwitchLevel <Axis name>.Sensor[1].PassiveHoming.SwitchLevel	No
<Axis name>.Config.Mechanics.InverseDirection	<Axis name>.Actor.InverseDirection	Yes
<Axis name>.Config.Mechanics.LeadScrew	<Axis name>.Mechanics.LeadScrew	Yes
<Axis name>.Config.Mechanics.PulsesPerDriveRevolution	<Axis name>.Actor.DriveParameter.PulsesPerDriveRevolution	Yes
<Axis name>.Config.PositionLimits_HW.Active	<Axis name>.PositionLimitsHW.Active	Yes
<Axis name>.Config.PositionLimits_HW.MaxSwitchedLevel	<Axis name>.PositionLimitsHW.MaxSwitchLevel	Yes
<Axis name>.Config.PositionLimits_HW.MinSwitchedLevel	<Axis name>.PositionLimitsHW.MinSwitchLevel	Yes
<Axis name>.Config.PositionLimits_SW.Active	<Axis name>.PositionLimitsSW.Active	Yes
<Axis name>.Config.PositionLimits_SW.MaxPosition	<Axis name>.PositionLimitsSW.MaxPosition	Yes
<Axis name>.Config.PositionLimits_SW.MinPosition	<Axis name>.PositionLimitsSW.MinPosition	Yes

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
Not available	<Axis name>.Actor.DirectionMode	No
Not available	<Axis name>.Actor.Type	No
Not available	<Axis name>.Sensor[1].ActiveHoming.Mode	No
Not available	<Axis name>.Sensor[1].PassiveHoming.Mode	No

ErrorBits tags (positioning axis)

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Axis name>.ErrorBits.HwLimitMax	<Axis name>.ErrorBits.HWLimit (Note the new status bits and the section Status of the limit switch (Page 1335).)	No
<Axis name>.ErrorBits.HwLimitMin		
<Axis name>.ErrorBits.SwLimitMaxExceeded	<Axis name>.ErrorBits.SWLimit (Note the new status bits and the section Status of the limit switch (Page 1335).)	No
<Axis name>.ErrorBits.SwLimitMaxReached		
<Axis name>.ErrorBits.SwLimitMinExceeded		
<Axis name>.ErrorBits.SwLimitMinReached		
Not available	<Axis name>.ErrorBits.DirectionFault	No

MotionStatus tags (positioning axis)

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Axis name>.MotionStatus.Distance	<Axis name>.StatusPositioning.Distance	Yes
<Axis name>.MotionStatus.Position	<Axis name>.Position	Yes
<Axis name>.MotionStatus.TargetPosition	<Axis name>.StatusPositioning.TargetPosition	Yes
<Axis name>.MotionStatus.Velocity	<Axis name>.Velocity	Yes

StatusBits tags (positioning axis)

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Axis name>.StatusBits.Homing	<Axis name>.StatusBits.HomingCommand	Yes
<Axis name>.StatusBits.SpeedCommand	<Axis name>.StatusBits.VelocityCommand	Yes
Not available	<Axis name>.StatusBits.HWLimitMaxActive	No
Not available	<Axis name>.StatusBits.HWLimitMinActive	No
Not available	<Axis name>.StatusBits.SWLimitMaxActive	No
Not available	<Axis name>.StatusBits.SWLimitMinActive	No

Tags (command table)

Tag name V1.0 to V3.0	Tag name as of V4.0	Automatic conversion V1..3 to ≥ V4
<Command table>.Config.Command[n].Position	<Command table>.Command[n].Position	Yes
<Command table>.Config.Command[n].Velocity	<Command table>.Command[n].Velocity	Yes
<Command table>.Config.Command[n].Duration	<Command table>.Command[n].Duration	Yes
<Command table>.Config.Command[n].NextStep	<Command table>.Command[n].NextStep	Yes
<Command table>.Config.Command[n].StepCode	<Command table>.Command[n].StepCode	Yes

See also

Overview of versions (Page 1326)

Changing a technology version (Page 1330)

Tags of the positioning axis technology object V4...5 (Page 1595)

Compatibility list of variables V4...5 <-> V6 (S7-1200)

The technology data blocks for S7-1200 Motion Control and S7-1500 Motion Control have continued to be standardized within the framework of the V6 technology. As of V6, this has resulted in new tag names for the technology object positioning axis.

Observe the information in the following tables if you have used tags of the technology objects in the user program and you want to convert the project from V4...5 to ≥ V6 or higher (or vice versa).

The tags listed in the "Automatic conversion V4...5 to ≥ V6" column are converted automatically when the project is compiled. Tag names in monitoring and force tables, HMI and trace configurations are not converted.

The following tags are new or have been adapted and may need to be corrected in the user program, monitoring tables, etc.:

Config variables (positioning axis)

Variable name V4.0 to V5.0	Variable name as of V6.0	Automatic conversion V4...5 to ≥ V6
<Axis name>.PositionLimitsSW.Active	<Axis name>.PositionLimits_SW.Active	Yes
<Axis name>.PositionLimitsSW.MinPosition	<Axis name>.PositionLimits_SW.MinPosition	Yes
<Axis name>.PositionLimitsSW.MaxPosition	<Axis name>.PositionLimits_SW.MaxPosition	Yes
<Axis name>.PositionLimitsHW.Active	<Axis name>.PositionLimits_HW.Active	Yes

Variable name V4.0 to V5.0	Variable name as of V6.0	Automatic conversion V4...5 to ≥ V6
<Axis name>.PositionLimitsHW.MinSwitchLevel	<Axis name>.PositionLimits_HW.MinSwitchLevel	Yes
<Axis name>.PositionLimitsHW.MinSwitchAddress	<Axis name>.PositionLimits_HW.MinSwitchAddress	Yes

Status of limit switch (S7-1200)

The status and error bits for the display of the reached limit switch have been adapted in version V4.

In order to replicate the behavior of the error bits of versions V1...3, use the following logical operators:

V1...3	V4 or higher
<Axis name>.ErrorBits.HwLimitMin	<Axis name>.ErrorBits.HWLimit AND <Axis name>.StatusBits.HWLimitMinActive
<Axis name>.ErrorBits.HwLimitMax	<Axis name>.ErrorBits.HWLimit AND <Axis name>.StatusBits.HWLimitMaxActive
<Axis name>.ErrorBits.SwLimitMinReached	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position = <Axis name>.PositioningLimits_SW.MinPosition)
<Axis name>.ErrorBits.SwLimitMinExceeded	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position < <Axis name>.PositioningLimits_SW.MinPosition)
<Axis name>.ErrorBits.SwLimitMaxReached	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position = <Axis name>.PositioningLimits_SW.MaxPosition)
<Axis name>.ErrorBits.SwLimitMaxExceeded	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position > <Axis name>.PositioningLimits_SW.MaxPosition)

See also

Overview of versions (Page 1326)

Changing a technology version (Page 1330)

Compatibility list of variables V1...3 <-> V4...5 (Page 1331)

6.1.1.5 Positioning axis technology object (S7-1200)

Integration of the positioning axis technology object (S7-1200)

The following representation shows the relationships between the hardware and software components that are implemented when using the positioning axis technology object:

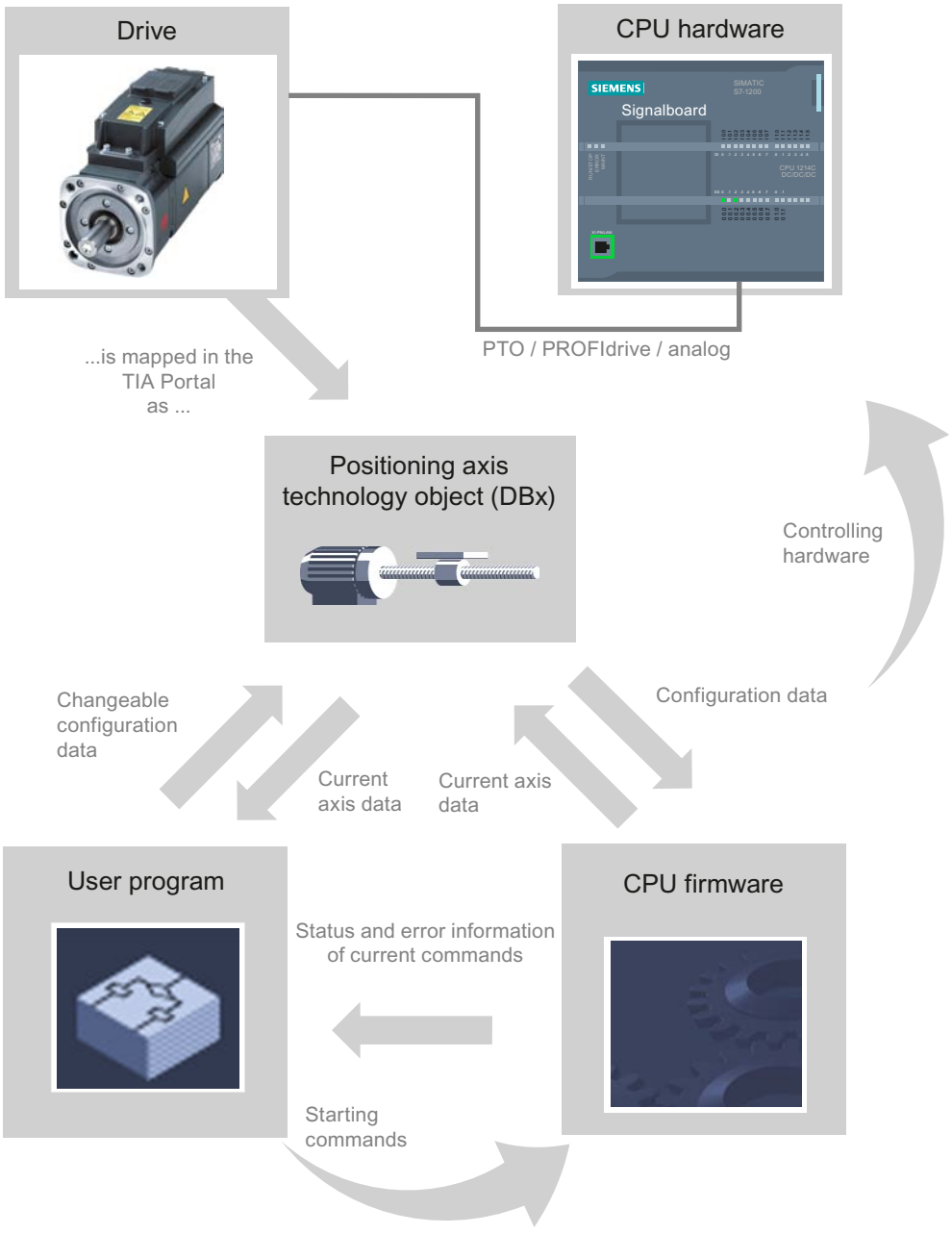

CPU hardware

The physical drive is controlled and monitored by the CPU hardware.

Drive

The drive represents the unit of power unit and motor. You can use stepper motors and servo motors with pulse, PROFIdrive or analog interfaces.

Positioning axis technology object

The physical drive including mechanics is mapped in the TIA Portal as a positioning axis technology object. To do this, configure the positioning axis technology object with the following parameters:

- Selection of the PTOs (Pulse Train Output)/PROFIdrive drives/analog outputs to be used and configuration of the drive interface
- Parameter for mechanics and gear transmission of the drive (or the machine or system)
- Parameters for position limits and position monitoring
- Parameters for dynamics and homing
- Parameters for the control loop

The configuration of the positioning axis technology object is saved in the technology object (data block). This data block also forms the interface between the user program and the CPU firmware. The current axis data is saved in the data block of the technology object at the runtime of the user program.

User program

You start Motion Control instructions jobs in the CPU firmware with the user program. The following jobs for controlling the axis are possible:

- Enable and disable axis
- Position axis absolutely
- Position axis relatively
- Move axis with velocity set point
- Run axis commands as movement sequence (technology as of V2, PTO only)
- Moving axes in jog mode
- Stop axis
- Reference axis; set reference point
- Change dynamic settings of axis
- Continuously read motion data of the axis
- Read and write variable of the axis
- Acknowledge error

You determine the command parameters with the input parameters of the Motion Control instructions and the axis configuration. The output parameters of the instruction give you up to date information about the status and any errors of the command.

Before starting a command for the axis, you must enable the axis with the Motion Control instruction "MC_Power".

You can read out configuration data and current axis data with the variables of the technology object. You can change individual, changeable variables of the technology object (e.g. the current acceleration) from the user program.

You can also change the dynamic settings of the axis with the Motion Control instruction "MC_ChangeDynamic" and write additional configuration data with "MC_WriteParam". You can read the current motion status of the axis with the Motion Control instruction "MC_ReadParam".

CPU firmware

The motion control jobs started in the user program are processed in the CPU firmware. When using the axis control panel, Motion Control jobs are triggered by operating the axis control panel. The CPU firmware performs the following jobs depending on the configuration:

- Calculate the exact motion profile for motion jobs and emergency stop situations
- Position control for drive connection via PROFIdrive/analog drive connection
- Control of the pulse and direction signal for drive connection via PTO
- Control of the drive enable
- Monitoring of the drive and the hardware and software limit switches
- Up to date feedback of status and error information to the Motion Control instructions in the user program
- Writing of current axis data into the data block of the technology object

See also

Tags of the positioning axis technology object as of V6 (Page 1489)

CPU outputs relevant for motion control (Page 1300)

Relationship between the signal type and the direction of travel (Page 1303)

Tools of the positioning axis technology object (Page 1339)

Hardware and software limit switches (Page 1322)

Homing (Page 1324)

Tools of the positioning axis technology object (S7-1200)

The TIA Portal provides the "Configuration", "Commissioning", and "Diagnostics" tools for the positioning axis technology object. The following representation shows the interaction of the three tools with the technology object and the drive:

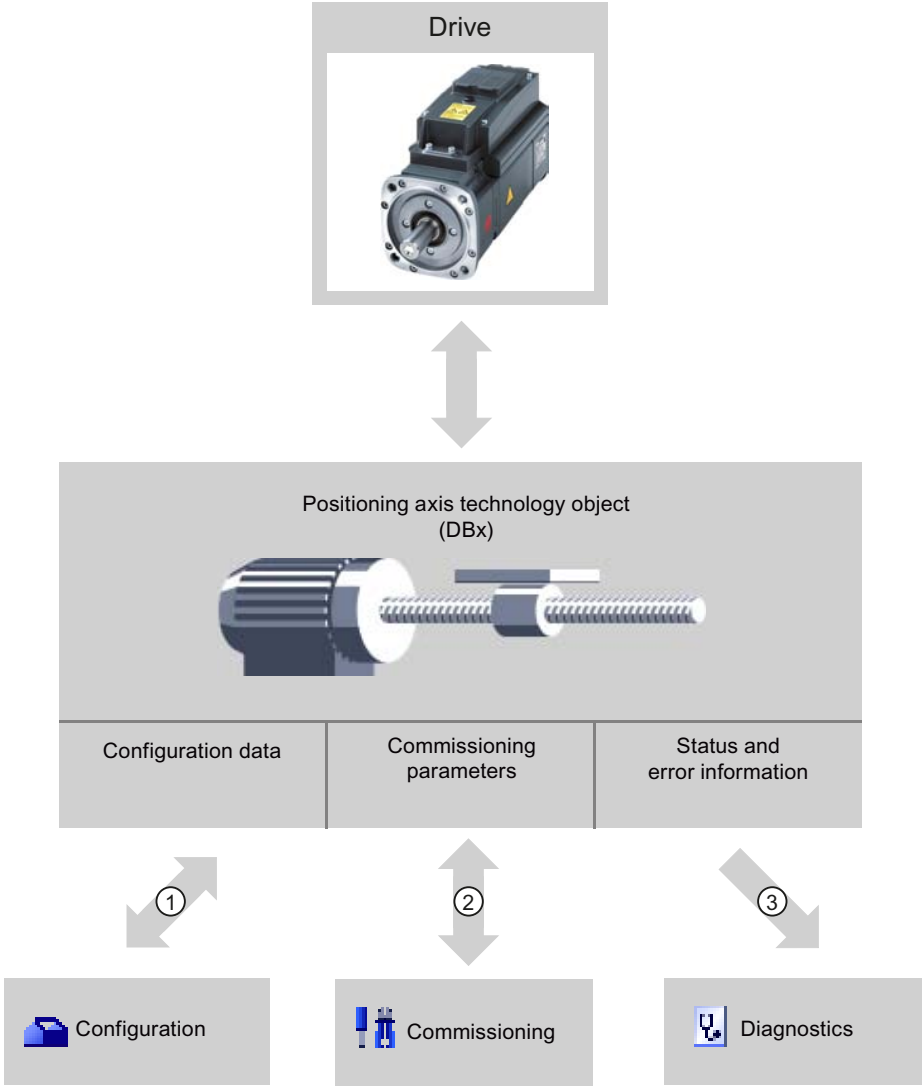

①	Reading and writing of configuration data of the technology object
②	Drive control via the technology object. Read the axis status for display in the axis control panel. Optimization of the position control
③	Readout of the current status and error information of the technology object Additional telegram information is displayed for PROFIdrive drives.

Configuration

Use the "Configuration" tool to configure the following properties of the positioning axis technology object:

- Selection of the PTOs (Pulse Train Output)/PROFIdrive drives/analog outputs to be used and configuration of the drive interface
- Properties of the mechanics and the transmission ratio of the drive (or machine/plant)
- Properties of the position limits and the position monitoring
- Properties of the dynamics and the homing
- Parameters of the control loop

Save the configuration in the data block of the technology object.

Commissioning

Use the "Commissioning" tool to test the function of your axis without having to create a user program. When the tool is started, the axis control panel will be displayed. The following commands are available on the axis control panel:

- Enabling and disabling the axis
- Move axis in jog mode
- Position axis in absolute and relative terms
- Home axis
- Acknowledge errors

The dynamic values can be adjusted accordingly for the motion commands. The axis control panel also shows the current axis status.

With drive connection via PROFIdrive / analog output, tuning supports you in determining the optimal gain for the control loop.

Diagnostics

Use the "Diagnostics" tool to keep track of the current status and error information for the axis and drive.

See also

CPU outputs relevant for motion control (Page 1300)

Relationship between the signal type and the direction of travel (Page 1303)

Integration of the positioning axis technology object (Page 1336)

Hardware and software limit switches (Page 1322)

Homing (Page 1324)

Configuring the positioning axis technology object (Page 1342)

Axis control panel (Page 1421)

Axis - Diagnostics (Page 1448)

Adding a positioning axis technology object (S7-1200)

Requirements

A project with a CPU S7-1200 has been created.

Procedure

To add a positioning axis technology object in the project tree, follow these steps:

1. Open the "CPU > Technology objects" folder in the project tree.
2. Double-click the "Add new object" command.
   The "Add new object" dialog opens.
3. Select the "Motion Control" technology.
4. Open the "Motion Control" folder.
5. Select the desired technology version in the "Version" column.
6. Select the "TO_PositioningAxis" object.
7. Enter the name of the axis in the "Name" input box.
8. To change the automatically assigned data block number, select the "Manual" option.
9. To display additional information about the technology object, click "Additional information".
10. Confirm your entry with "OK".

Result

The new technology object is created and saved to the "Technology objects" folder in the project tree.

The organization blocks MC-Servo [OB91] and MC-Interpolator [OB92] are automatically created in the "Program blocks" folder. The technology objects are processed in these organization blocks. The position controller is calculated in the MC-Servo [OB91]. The MC-Interpolator [OB92] takes over the evaluation of the Motion Control instructions, the setpoint generation and the monitoring functionality.

See also

Guidelines on use of motion control (Page 1326)

Configuring the positioning axis technology object (S7-1200)

Working with the configuration dialog (S7-1200)

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the group of the required technology object in the project tree.
2. Double-click the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters contain all the parameters which must be configured for a functioning axis.
- **Extended parameters**
  The advanced parameters include parameters to adapt to your drive or your plant.

Configuration window icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values you can use the technology object without additional changes.
	The configuration contains user-defined or automatically adapted values and is complete. All input fields of the configuration contain valid values and at least one preset value has changed.
	The configuration is incomplete or incorrect At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to display the cause of the error.
	The configuration is valid but contains warnings For example, only one hardware limit switch is configured. Depending on the application, the lacking configuration of a hardware limit switch may result in a hazard. The corresponding field or the drop-down list is displayed on a yellow background.

See also

Guidelines on use of motion control (Page 1326)

Basic parameters (Page 1343)

Extended parameters (Page 1355)

Monitor values (S7-1200)

If there is an online connection to the CPU, the icon "Monitor all" is displayed in the configuration dialog of the technology object.

The "Monitor all" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of the actual values in the project by manual transfer to the start value of the project

Icons and operator controls

If there is an online connection to the CPU, the actual values are displayed at the parameters.

In addition to the actual values of the parameters, the following symbols appear:

Icon	Description
	Start value in CPU matches the configured Start value in the project
	Start value in CPU does not match the configured Start value in the project
	A comparison of the start value in the CPU with the configured start value in the project cannot be performed because the selected CPU module does not support this comparison.
	The value is not comparable with any significance since it is not relevant in one of the configurations.
	Use the button to show the start value of the CPU and the start value of the project for the respective parameter.

The actual value and the start value in the project can be changed directly and then downloaded to the CPU. The change of the actual value is transferred directly to the CPU for directly modifiable parameters.

Basic parameters (S7-1200)

Configuration - General (S7-1200)

Configure the basic properties of the positioning axis technology object in the "General" configuration window.

Axis name

Define the name of the axis or the name of the positioning axis technology object in this field. The technology object is listed under this name in the project tree.

Drive

Select the type of drive connection:

- **PTO (Pulse Train Output)**
  The drive is connected via a pulse generator output, an optional enable output and an optional ready input.
- **Analog drive connection**
  The drive is connected via an analog output, an encoder, an optional enable output and an optional ready input.
  All movements of the axis are position-controlled.
- **PROFIdrive**
  The drive is connected via PROFINET/PROFIBUS. Communication between controller and drive is by means of PROFIdrive telegrams.
  All movements of the axis are position-controlled.

If you select the "Analog drive connection" or "PROFIdrive", additional elements are added to the navigation of the configuration:

- Encoder
- Modulo
- Position supervisions (positioning supervision, following error and standstill signal)
- Control loop

In the additional configuration windows, you configure the encoders that are to be connected and the resulting options for position control and position monitoring.

Unit of measurement position

In the drop-down list, select the desired measurement unit for the dimension system of the axis. The selected measurement unit is used for further configuration of the positioning axis technology object and for displaying the current axis data.

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this measurement unit.

Note

Select the drive connection and the measurement unit of the position at the beginning of the axis configuration.

With a subsequent change, the parameters are reset or re-initialized, which requires you to check the parameters of the configuration dialogs once again.

You may have to adapt the values of the input parameters of the Motion Control instructions to the new unit of measurement in the user program.

Simulation

In the drop-down list, select whether or not the drive and the encoder are to be simulated. The simulation can be selected for the analog drive connection or for a PROFIdrive drive. The configuration of the drive and encoder hardware is not required for simulation mode (potential errors in the drive and encoder configuration are ignored).

Application: The drive is simulated, for example, for commissioning and later operated with the hardware that may be configured.

The "Simulation" operating mode can be changed during runtime of the user program with a download and then MC_Reset with parameter "Restart" = TRUE.

In simulation mode, setpoints are not output to the drive and actual values are not read in from the drive/encoder. Hardware limit switches and homing switches have no effect.

The following table shows Motion Control instructions with adapted behavior in simulation mode.

Motion Control instruction	Behavior in simulation mode
MC_Power	The axis is enabled immediately without waiting for feedback from the drive.
MC_Home	Homing jobs are executed immediately without simulated axis motion.

PTO drives work without control loop. No separate simulation function is required in order to simulate a PTO drive when the PTO drive is not connected.

See also

CPU outputs relevant for motion control (Page 1300)

Relationship between the signal type and the direction of travel (Page 1303)

Configuration - General ("Axis" technology object V1...3) (Page 1531)

Configuration - Drive (S7-1200)

Configuration - Drive - PTO (Pulse Train Output) (S7-1200)

In the "Drive" configuration window, configure the pulse generator and the enable and feedback of the drive.

Hardware interface

The pulses are output to the power unit of the drive by fixed assigned digital outputs.

In CPUs with relay outputs, the pulse signal cannot be output at these outputs because the relays do not support the necessary switching frequencies. To be able to work with the PTO (Pulse Train Output) on these CPUs, you must use a signal board with digital outputs.

Note

The PTO requires the functionality of a high-speed counter (HSC). An internal HSC is used for this, the count of which cannot be evaluated.

Pulse generator

In the drop-down list, select the PTO (Pulse Train Output) to control the stepper motor or servo motor by means of pulse interface. If you have not used the pulse generators and high-speed counters elsewhere in the device configuration, the hardware interface can be configured automatically. In this case, the PTO selected in the drop-down list is displayed with a white background.

If PTO (PulseTrain Output) is selected, the "Device configuration" button takes you to the parameter assignment of the pulse options in the device configuration of the CPU. This may be necessary if there is a conflict because the PTO is being used at the other end or the parameters have been changed by the user.

Signal type

Select the signal type in the drop-down list. The following signal types are available:

- **PTO (pulse A and direction B)**
  A pulse output and a direction output are used for controlling the stepper motor.
- **PTO (clock up A and clock down B)**
  One pulse output each for motion in positive direction and negative direction is used for controlling the stepper motor.
- **PTO (A/B phase-shifted)**
  Both pulse outputs for Phase A and for Phase B run at the same frequency.
  The period of the pulse outputs is evaluated at the drive end as a step.
  The phase offset between Phase A and Phase B determines the direction of the motion.
- **PTO (A/B phase offset - quadruple)**
  Both pulse outputs for Phase A and for Phase B run at the same frequency.
  All positive edges and all negative edges of Phase A and Phase B are evaluated as a step at the drive end.
  The phase offset between Phase A and Phase B determines the direction of the motion.

The following table shows the parameters to be configured depending on the signal type:

Signal type/parameter		Description
PTO (pulse A and direction B)		
	Pulse output	Select the pulse output for motion in positive direction in this field. You can select the output using a symbolic address or assign it to an absolute address.
	Activate direction output	With this option, you enable or disable the direction output. The motion direction is restricted when you disable the direction output.
	Direction output	Select the output for the direction output in this field. You can select the output using a symbolic address or assign it to an absolute address.
PTO (clock up A and clock down B)		
	Pulse output forward	Select the pulse output for motion in positive direction in this field. You can select the output using a symbolic address or assign it to an absolute address.
	Pulse output backward	Select the pulse output for motion in negative direction in this field. You can select the output using a symbolic address or assign it to an absolute address.
PTO (A/B phase offset)/PTO (A/B phase offset - quadruple)		
	Signal A	Select the pulse output for Phase A signals in this field. You can select the output using a symbolic address or assign it to an absolute address.
	Signal B	Select the pulse output for Phase B signals in this field. You can select the output using a symbolic address or assign it to an absolute address.

Drive enable and feedback

In this area, you configure the output for drive enable and the input for the "Drive ready" feedback of the drive:

- Enable output
  Select the enable output for the drive enable in this field.
- Ready input
  Select the ready input for the "Drive ready" feedback of the drive in this field

Drive enable is controlled by Motion Control instruction "MC_Power" and enables power to the drive. The drive signals "Drive ready" to the CPU if it is ready to start executing movement after receiving the drive enable.

If the drive does not have any interfaces of this type, you do not have to configure the parameters. In this case, select the value TRUE for the ready input.

Configuration - Drive - Analog drive connection (S7-1200)

In the "Drive" configuration window, configure the analog output and the enable and feedback of the drive.

Hardware interface

The speed setpoint is output to the power unit of the drive by means of permanently assigned analog outputs.

Configure the inputs and outputs for the control of the drive in this area:

- Analog output
  In this field, select the PLC variable of the analog output via which the drive is controlled. When you open the autocompletion, all output addresses are displayed with 16 bits (WORD, INT, UINT). The variable of a data block with the WORD data type can be selected for data connection via a data block.
  You can also enter an address, for example QW20. If the address is valid, the name "Axis_1_AnalogOutput" is generated for this address and inserted in the variable table. For the address to be valid, it needs to be occupied by the appropriate data type and a HW module.
- Selection of enable output
  Select an available output as the enable output for the drive enable in this field.
- Selection of ready input
  Select the ready input for the "Drive ready" feedback of the drive in this field

Drive enable is controlled by Motion Control instruction "MC_Power" and enables power to the drive. The drive signals "Drive ready" to the CPU if it is ready to start executing movement after receiving the drive enable. If the drive does not have any interfaces of this type, you do not have to configure the parameters. In this case, select the value TRUE for the ready input.

Data exchange with the drive

In this area, you can configure the scaling of the setpoint speed:

- Reference speed
  The reference speed of the drive is the speed, with which the drive spins when there is an output of 100% at the analog output. The reference speed must be configured in the drive, and transferred into the configuration of the technology object.
  The analog value that is output at 100% depends on the type of the analog output. As an example, for an analog output with ±10 V, the value 10 V is output at 100%.
- Maximum speed
  In this field, specify the maximum speed of the drive.
  The maximum speed is limited by the drive and by the value range of the analog output. In the simplest situation, the reference speed and maximum speed are identical.
  Analog outputs can be overloaded by approximately 17%. If the drive permits overloading, you can use this to operate an analog output as a limit in the -117% to 117% range.
- Invert drive direction
  To invert the rotation direction of the drive, select the check box.

See also

Data connection drive with analog drive connection (Page 1316)

Configuration - Drive - PROFIdrive (S7-1200)

In the "Drive" configuration window, configure the data connection and the parameters of the PROFIdrive drive.

PROFIdrive drive (as of V6)

- **Data connection**
  In the drop-down list, select whether the data connection is to be made directly with the drive device or via an editable data block in the user program.
- **Drive** (for data connection: "Drive")
  In the "Drive" field, select an already configured PROFIdrive drive.
- **Data block** (for data connection: "Data block")
  Select a previously created data block which contains a variable structure of the data type "PD_TELx" ("x" stands for the telegram number to be used).

Data exchange with the drive

In this area, you can configure the data exchange between the drive and controller:

- **Drive telegram** (for data connection: "Data block" not selectable)
  In the drop-down list, check or select the telegram of the drive. The specification must match the device configuration of the drive.
- **Input/output address**
  The fields show the symbolic and absolute input and output address of the telegram.

- **Invert drive direction**
  To invert the rotation direction of the drive, select the check box.
- **Automatic transfer of drive parameters in the device**
  Select the check box if you want the drive parameters "Reference speed" and "Maximum speed" to be transferred as values from the drive configuration to the CPU. The drive parameters are transferred from the bus after the (re-)initialization of the technology object and (re)start of the drives and the CPU.
  Alternatively, you must synchronize the following parameters manually:
  - **Reference speed**
    Configure the reference speed to match the one in the configuration of the drive.
    On the bus, the value 16#4000 is transferred, for example, which corresponds to 100% of the reference speed.
  - **Maximum speed**
    Configure the maximum speed of the drive in this field.
    The maximum speed is obtained from the configuration of the drive. A maximum of -200% to +200% of the reference speed can be transmitted over the bus. The maximum speed can thus be twice the reference speed at maximum.

Note

Automatic transfer of drive parameters is only possible with SINAMICS drives as of V4.x. For this, "Drive" must be selected as the data connection in the configuration window.

See also

Data connection PROFIdrive drive/PROFIdrive encoder (Page 1311)

Automatic transfer of drive and encoder parameters in the device (Page 1308)

Configuration - Encoder (S7-1200)

Encoder connection (S7-1200)

Depending on the selection of the encoder connection, you configure various parameters in the "Encoder" configuration window. The following encoder connections are possible:

- Encoder on high-speed counter (HSC) (Page 1353)
- PROFIdrive encoder on PROFINET/PROFIBUS (encoder on drive, encoder on technology module, PROFIdrive encoder) (Page 1351)

Configuration - Encoder - Encoder on PROFINET/PROFIBUS (S7-1200)

Encoder selection

In the "PROFIdrive encoder" box, select a PROFIdrive encoder on PROFINET.

- **Data connection**
  In the drop-down list, select whether the data connection should be established directly with the encoder or via a data block that can be edited in the user program.
- **PROFIdrive encoder/data block**
  Select a previously configured PROFIdrive encoder in this configuration field.
  The following encoders can be selected:
  - **Connection to drive (not with analog drive connection)**
    The encoder is connected to the drive. The encoder signals are evaluated by the drive and transmitted to the controller as part of the drive telegram (telegram 3 or 4) (the encoders of the telegrams from other drives cannot be used).
    The encoder is configured using the configuration of the PROFIdrive drive.
  - **Encoder on technology module (TM)**
    Select a previously configured technology module and the channel to be used. Only technology modules set to the "Position input for Motion Control" mode are displayed for selection.
    If no technology module is available for selection, change to the device configuration and add a technology module.
    You can identify the technology modules suitable for position detection for Motion Control in the documentation for the technology module and the catalog data.
  - **PROFIdrive encoder on PROFINET/PROFIBUS (PROFIdrive)**
    In the "PROFIdrive encoder" field, select a previously configured encoder on PROFINET/PROFIBUS. Switch to the device configuration in the network view and add an encoder, in the event that no encoder is offered for selection.

  If "Data block" was selected for the data connection, a previously created data block containing a tag structure of data type "PD_TELx" must be selected here ("x" stands for the telegram number to be used by which the encoder is connected). The encoder of the selected drive telegram (Tel 3 or 4) or a separate encoder (Tel 81 or 83) can be used.

Data exchange with encoder

In this area, you can configure the data exchange between the encoder and controller:

- **Encoder telegram** (for data connection: "Data block" not selectable)
  In the drop-down list, select the telegram of the encoder. The specification must match the device configuration.
- **Input/output address**
  The fields show the symbolic and absolute input and output address of the telegram.

- **Invert encoder direction**
  To invert the actual value of the encoder, select the check box.
- **Automatic transfer of encoder parameters in the device**
  Select the check box if you want to transfer the encoder parameters from the encoder configuration to the CPU. The encoder parameters are transferred from the bus after the (re-)initialization of the technology object and (re)start of the encoder and the CPU. The encoder type must be the same in the configuration of the axis and in the configuration of the drive.

  Note

  Automatic transfer of encoder parameters is only possible with PROFIdrive encoders as of product version A16. For this, "Encoder" must be selected as the data connection in the configuration window.

  A product version > V4.x is required to use an encoder on the SINAMICS drive.

  The parameters must be adjusted manually if there is no automatic transfer of encoder parameters. You can find the parameters to be synchronized in the section Automatic transfer of drive and encoder parameters in the device (Page 1308).

Encoder type

Set the employed encoder type in the "Encoder type" box. The following encoder types can be selected:

- **Linear incremental**
- **Linear absolute**
- **Rotary incremental**
- **Rotary absolute**

Configure the various parameters depending on the selected encoder type. Depending on the selected encoder type, configure the following parameters:

Encoder type/parameter		Description
Linear incremental		
	Distance between two increments	In this field, you configure the distance between two steps of the encoder.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Linear absolute		
	Distance between two increments	In this field, you configure the distance between two steps of the encoder.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Fine resolution - Bits in abs. actual value (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).
Rotary incremental		

Encoder type/parameter		Description
	Steps per revolution	In this field, configure the number of steps that the encoder resolves per revolution.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Rotary absolute		
	Steps per revolution	In this field, configure the number of steps that the encoder resolves per revolution.
	Number of revolutions	In this field, configure the number of revolutions that the absolute value encoder can detect.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
	Fine resolution - Bits in abs. actual value (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

See also

Data connection PROFIdrive drive/PROFIdrive encoder (Page 1311)

Configuring technology modules for Motion Control (Page 1398)

Configuration - Encoder - Encoder on high-speed counter (HSC) (S7-1200)

Selection of high-speed counter (HSC)

In the high-speed counter box, select the high-speed counter to which the encoder transfers the actual value.

Check the filter times of the two high-speed counter digital inputs that are used. The filter times should be short enough to ensure reliable recording of the pulses.

HSC interface

Select the operating mode of the high-speed counter in the "Operating mode" box.

Depending on the operating mode, configure the various inputs:

Operating mode/parameter		Description
Two-phase		
	Clock generator forward	In this field, select the input for counting up. You can select the input using a symbolic address or assign it to an absolute address. The frequency and the location (on-board, signal board) of the input are displayed next to the address box.
	Clock generator backward	In this field, select the input for counting down. You can select the input using a symbolic address or assign it to an absolute address. The frequency and the location (on-board, signal board) of the input are displayed next to the address box.
A/B counter / A/B counter quadruple		
	Clock generator A	In this field, select the input for Phase A signals. You can select the input using a symbolic address or assign it to an absolute address. The frequency and the location (on-board, signal board) of the input are displayed next to the address box.
	Clock generator B	In this field, select the input for Phase B signals. You can select the input using a symbolic address or assign it to an absolute address. The frequency and the location (on-board, signal board) of the input are displayed next to the address box.

Invert encoder direction

To invert the actual value of the encoder, select the check box.

Automatic transfer of encoder parameters in the device

This selection is not possible when using encoders on the high-speed counter (HSC).

Encoder type

Select the encoder type in the "Encoder type" box. The following encoder types can be selected:

- **Linear incremental**
- **Rotary incremental**

Configure the various parameters depending on the selected encoder type. Depending on the selected encoder type, configure the following parameters:

Encoder type/parameter		Description
Linear incremental		
	Distance between two increments	In this field, you configure the distance between two steps of the encoder.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type/parameter		Description
Rotary incremental		
	Steps per revolution	In this field, configure the number of steps that the encoder resolves per revolution.
	Fine resolution - Bits in incr. actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Extended parameters (S7-1200)

Mechanics (S7-1200)

Configuration - Mechanics - PTO (Pulse Train Output) (S7-1200)

Configure the mechanical properties of the drive in the "Mechanics" configuration window.

Pulses per motor revolution

Configure the number of pulses required for one revolution of the motor in this box.

Limits (independent of the selected unit of measurement):

- 0 < Pulse per motor revolution ≤ 2147483647

Load motion per motor revolution

In this box, configure the load distance per motor revolution covered by the mechanical system of your unit.

Limits (independent of the selected unit of measurement):

- 0.0 < Distance per revolution ≤ 1.0e12

Permitted direction of rotation (technology version as of V4)

Configure this box to determine whether the mechanics of your system are to move in both directions or only in the positive or negative direction.

If you have not activated the direction output of the pulse generator in the "PTO (pulse A and direction B)" mode, the selection is limited to the positive or negative direction.

Invert direction

You can use the "Invert direction" check box to adapt the control system to the direction logic of the drive.

The direction logic is inverted according to the selected mode of the pulse generator:

- **PTO (pulse A and direction B)**
  - 0 V at direction output ⇒ positive direction of rotation
  - 5 V/24 V at direction output ⇒ negative direction of rotation

  The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.
- **PTO (clock up A, clock down B)**
  The outputs "Pulse output down" and "Pulse output up" are swapped.
- **PTO (A/B phase-shifted)**
  The "Phase A" and "Phase B" outputs are swapped.
- **"PTO (A/B phase-shifted - quadruple)**
  The "Phase A" and "Phase B" outputs are swapped.

Configuration - Mechanics - PROFIdrive/analog drive connection (S7-1200)

Configure the mechanical properties of the drive and its encoder in the "Mechanics" configuration window.

Encoder mounting type

In the drop-down list, select how the encoder is mounted on the mechanism. The following encoder installation types are possible:

- **On motor shaft**
- **External measuring system** (rotary encoders only)

Position parameters

Depending on the selected encoder installation type, configure the following position parameters:

Encoder installation type/position parameter		Description
On the motor shaft		
	Load motion per motor revolution	In this field, configure the load distance for one motor revolution.
External measuring system		
	Load motion per motor revolution	In this field, configure the load distance for one motor revolution.
	Distance per encoder revolution	In this field, configure the distance recorded by the external measuring system per encoder revolution.

Configuration - Modulo (PROFIdrive/analog drive connection only) (S7-1200)

Use the "Modulo" setting if you want to limit the traversing range to a recurring distance based on the product length / product cycle. The modulo function is only possible in position-controlled operation of the axis.

When "modulo" is enabled, the position value of the technology object is represented by means of a recurring modulo range. The modulo range is defined by the start value and the length.

For example, to limit the position value of an axis to one full rotation, the modulo range can be defined with start value = 0° and length = 360°. With an encoder resolution of 0.1°/encoder step, the position value is represented in the modulo range 0.0° to 359.9°. If the axis in this example moves to the position 400°, the actual position 40° (400° to 360°) is reached.

When "Modulo" is activated, specify the traversing direction at the Motion Control instruction "MC_MoveAbsolute" with the "Direction" input parameter. The following parameter values are available:

- 0: The sign for the velocity ("Velocity" parameter) determines the motion direction.
- 1: Target position is approached in a positive direction.
- 2: Target position is approached in a negative direction.
- 3: Starting from the current position, the technology selects the shortest distance to the target position.

Enable modulo

Select the "Enable modulo" check box to use a recurring reference system for the axis (for example, 0.0° to 359.9°).

Modulo start value

In this field, define the position at which the modulo range should begin (for example, 0°).

Modulo length

In this field, define the length of the modulo range (for example, 360°).

Position limits (S7-1200)

Requirements for hardware limit switches (S7-1200)

Use only hardware limit switches that remain permanently switched after being approached. This switching status may only be revoked after a return to the valid travel range.

See also

Configuration - Position limits (Page 1358)

Response of the axis when position limits are triggered (Page 1360)

Changing the configuration of the position limits in the user program (Page 1361)

Configuration - Position limits (S7-1200)

Configure the hardware and software limit switches of the axis in the "Position limits" configuration window.

Enable HW limit switches

Activate the function of the low and high hardware limit switch with this check box. The hardware limit switches can be used for purposes of direction reversal during a homing procedure. For details, refer to the configuration description for homing.

Enable SW limit switch

Activate the function of the low and high software limit switch with this check box.

Note

Activated software limit switches act only on a homed axis.

Input HW low / high limit switch

Select the digital input for the low or hardware high limit switch from the drop-down list.

The input of PTO axes must be interrupt-capable. You achieve the shortest response time with interrupt-capable inputs with a drive connection via PROFIdrive / analog drive connection. Alternatively, you can assign the input to the "TPA OB Servo" process image and then receive a response time in the cycle time of the "TPA OB Servo". The assignment of the standard process image of the organization block OB1 is not recommended, as the longest response times occur here.

The digital onboard CPU inputs and the digital inputs of a plugged signal board can be selected as interrupt-capable inputs for the HW limit switches.

Note

The digital inputs are set to a filter time of 6.4 ms by default. If these are used as hardware limit switches, undesired decelerations may occur. If this occurs, reduce the filter time for the relevant digital inputs.

The filter time can be set under "Input filter" in the device configuration of the digital inputs.

Select level

In the drop-down list, select the signal level available at the CPU when the hardware limit switch is approached.

- Selection of "Low level" (normally closed contact)
  0 V (FALSE) at CPU input corresponds to hardware limit switch approached
- Selection of "High level" (normally open contact)
  5 V / 24 V (TRUE) at the CPU input = hardware limit switch approached (the actual voltage depends on the hardware used)

Software high / low limit switch

Enter the position value of the low and high software limit switch in these boxes.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ software low limit switch ≤ 1.0e12
- -1.0e12 ≤ software high limit switch ≤ 1.0e12

The value of the software high limit switch must be greater than or equal to the value of the software low limit switch.

See also

Requirements for hardware limit switches (Page 1357)

Response of the axis when position limits are triggered (Page 1360)

Changing the configuration of the position limits in the user program (Page 1361)

Configuration - Homing - Active (Page 1369)

Response of the axis when position limits are triggered (S7-1200)

Behavior of axis when hardware limit switches are approached

When a hardware limit switch is approached, the axis behaves differently depending on the drive connection:

- Drive connection via PROFIdrive / analog output
  When a hardware limit switch is approached, the axis is disabled and, depending on the configuration, braked at the drive and brought to a standstill. You must select the deceleration sufficiently large in the drive so that the axis stops reliably before the mechanical stop.
- Drive connection via PTO (Pulse Train Output)
  When the hardware limit switches are approached, the axis brakes to a standstill at the configured emergency deceleration. You must select the emergency deceleration sufficiently large so that the axis stops reliably before the mechanical stop. The following diagram presents the behavior of the axis after it approaches the hardware limit switches:

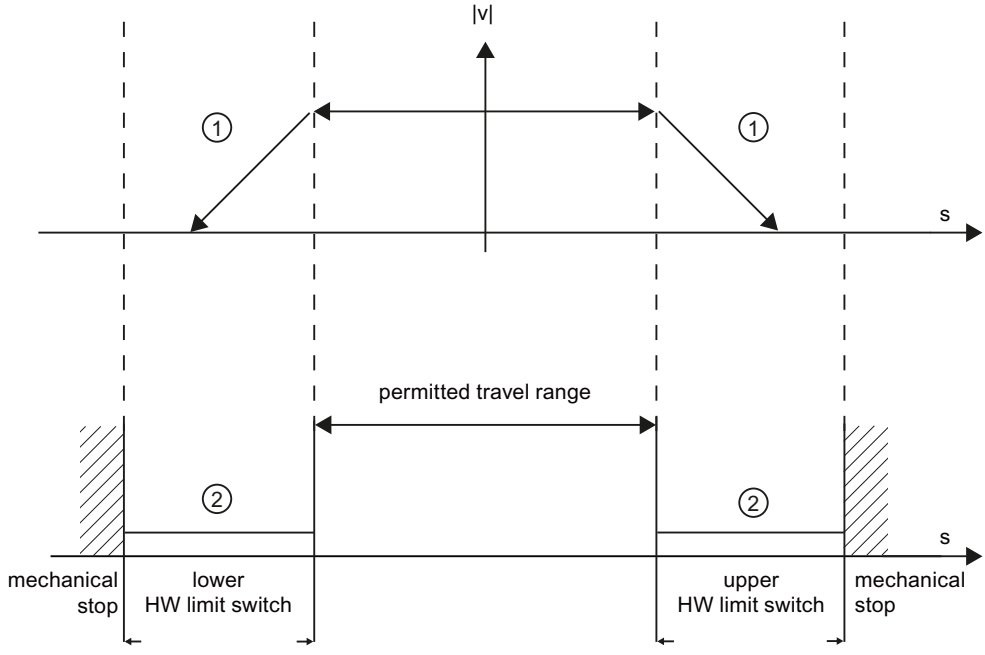

① The axis brakes at the configured emergency stop deceleration (PTO) or the deceleration configured in the drive (PROFIdrive or analog drive interface) until standstill occurs.

② Range in which the HW limit switches signal the status "approached".

The "HW limit switch approached" error is displayed in the initiating Motion Control instruction, at "MC_Power", and in the technology object variables. Instructions for eliminating errors can be found in the Appendix under "List of ErrorIDs and ErrorInfos".

Behavior of axis when software limit switches are reached

If software limit switches are activated, an active motion is stopped at the position of the software limit switch. The axis is braked at the configured deceleration.

The following diagram presents the behavior of the axis until it reaches the software limit switches:

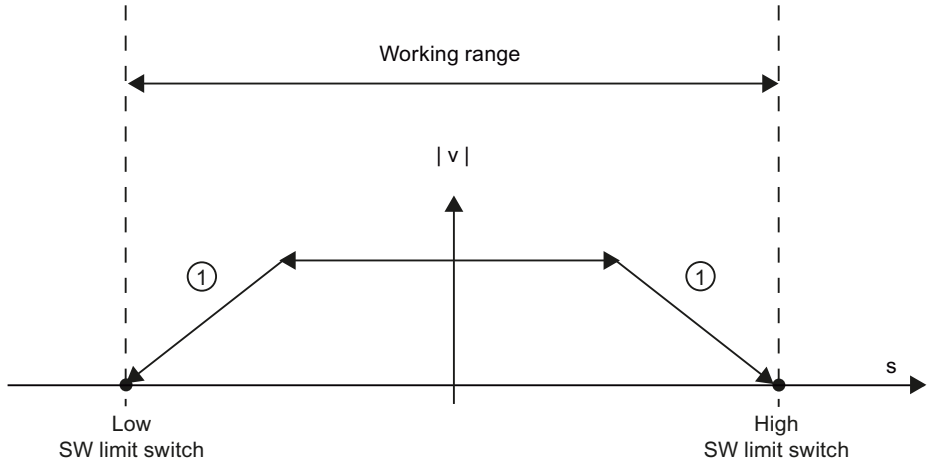

① The axis brakes to a standstill at the configured deceleration.

The "SW limit switch approached" error is displayed in the initiating Motion Control instruction, at "MC_Power", and in the technology object variables. Instructions for eliminating errors can be found in the Appendix under "List of ErrorIDs and ErrorInfos".

When a software limit switch is overtraveled, the axis behaves differently depending on the drive connection:

- Drive connection via PROFIdrive / analog output
  When a software limit switch is overtraveled, the axis is disabled and, depending on the configuration, braked at the drive and brought to a standstill.
- Drive connection via PTO (Pulse Train Output)
  You can learn about the behavior of the axis when a software limit switch is overtraveled in the sections "Software limit switches in conjunction with a homing operation (Page 1459)" and "Software limit switches in conjunction with dynamic changes (Page 1464)".

Use additional hardware limit switches if a mechanical endstop is located after the software limit switches and there is a risk of mechanical damage.

See also

Requirements for hardware limit switches (Page 1357)

Configuration - Position limits (Page 1358)

Changing the configuration of the position limits in the user program (Page 1361)

Changing the configuration of the position limits in the user program (S7-1200)

You can change the following configuration parameters during runtime of the user program in the CPU:

Hardware limit switches

You can also activate and deactivate the hardware limit switches during runtime of the user program. Use the following technology object variable for this purpose:

- <axis name>.PositionLimits_HW.Active

Refer to the description of the technology object variables (Page 1489) in the appendix for information on when changes to the configuration parameter take effect.

Software limit switches

You can also activate and deactivate the software limit switches and change their position values during runtime of the user program. Use the following technology object variables for this purpose:

- <axis name>.PositionLimits_SW.Active
  for activating and deactivating the software limit switches
- <axis name>.PositionLimits_SW.MinPosition
  for changing the position of the low software limit switch
- <axis name>.PositionLimits_SW.MaxPosition
  for changing the position of the high software limit switch

Refer to the description of the technology object variables (Page 1489) in the appendix for information on when changes to the configuration parameters take effect.

See also

Compatibility list of variables V1...3 <-> V4...5 (Page 1331)

Requirements for hardware limit switches (Page 1357)

Configuration - Position limits (Page 1358)

Response of the axis when position limits are triggered (Page 1360)

Dynamics (S7-1200)

Configuration - Dynamics - General (S7-1200)

Configure the maximum velocity, the start/stop velocity, the acceleration and deceleration, and the jerk limit (positioning axis technology object as of V2) of the axis in the "General dynamics" configuration window.

Unit of velocity limits

Select the unit of measurement with which you want to set the velocity limits in the drop-down list. The unit set here depends on the unit of measurement set under "Configuration > Basic parameters > General" and only serves to simplify input. This provides the possibility, for example, to enter the maximum velocity as a speed value of the motor in rpm.

Note

Rounding error

If you select a different unit in the "Unit of velocity limitation" drop-down list than in "Configuration > Basic parameters > General", note that a rounding error may occur.

Maximum velocity / Start/stop velocity

Define the maximum permissible velocity and the start/stop velocity of the axis in these boxes. The start/stop velocity is the minimum permissible velocity of the axis and can only be configured for drive connection via PTO (Pulse Train Output).

For drive connection via PROFIdrive or analog output, the start/stop velocity is fixed at zero. The maximum velocity when connecting via PROFdrive or analog output amounts to 1.0E12 of the selected measurement unit (e.g. mm/s, °/s, ...).

Positioning axis technology object (PTO) as of V4

Signal board	Velocity [pulse/s]
20 kHz	1 ≤ start/stop velocity ≤ 20 000 1 ≤ maximum velocity ≤ 20 000
200 kHz	1 ≤ start/stop velocity ≤ 200 000 1 ≤ maximum velocity ≤ 200 000

On-board CPU output	Velocity [pulse/s]
100 kHz	1 ≤ start/stop velocity ≤ 100 000 1 ≤ maximum velocity ≤ 100 000
20 kHz	1 ≤ start/stop velocity ≤ 20 000 1 ≤ maximum velocity ≤ 20 000
1 MHz CPU 1217	1 ≤ start/stop velocity ≤ 1000 000 1 ≤ maximum velocity ≤ 1000 000

You can learn about the limits for the technology object positioning axis < V4 in the appendix Outputs of the CPU relevant for Motion Control (technology version V1...3) (Page 1528).

The value of the maximum velocity must be greater or equal to the value of the start/stop velocity.

The limits for other units of measurement must be converted by the user to conform to the given mechanics.

Acceleration / Deceleration - Ramp-up time / Ramp-down time

Set the desired acceleration in the "Ramp-up time" or "Acceleration" boxes. The desired deceleration can be set in the "Ramp-down time" or "Deceleration" boxes.

The relation between the ramp-up time and acceleration and the ramp-down time and deceleration is shown in the following equations:

$$\text{Ramp-up time} = \frac{\text{Maximum velocity} - \text{Start/stop velocity}}{\text{Acceleration}}$$

$$\text{Ramp-down time} = \frac{\text{Maximum velocity} - \text{Start/stop velocity}}{\text{Deceleration}}$$

Motion jobs started in the user program are performed with the selected acceleration / deceleration.

The limits for acceleration and deceleration with drive connection via PTO (Pulse Train Output) can be found in section CPU outputs relevant for motion control (Page 1300).

Note

Changes to the velocity limits ("start/stop velocity" and "maximum velocity") influence the acceleration and deceleration values of the axis. The ramp-up and ramp-down times are retained.

Enable jerk limit, positioning axis technology object (as of V2)

Enable the jerk limit with this check box.

Axis acceleration and deceleration is not stopped abruptly when the jerk limit is activated; it is adjusted gently according to the set step or smoothing time.

Note

The check box is no longer displayed as a parameter in the technology data block as of V4. By disabling the check box, the jerk value is set to 0.0.

Rounding time/jerk, positioning axis technology object (as of V2)

You can input the parameters of the jerk limit in the "Smoothing time" box or alternatively in the "Jerk" box.

- Set the desired jerk for acceleration and deceleration ramp in the "Jerk" box.
- Set the desired smoothing time for the acceleration ramp in the "Smoothing time" box.

Note

Smoothing time V2...3

The set smoothing time visible in the configuration only applies to the acceleration ramp.

If the values for acceleration and deceleration differ, the smoothing time of the deceleration ramp is calculated according to the jerk of the acceleration ramp and used. (See also Behavior of the axis when using the jerk limit (Page 1367)

The smoothing time of the deceleration is adapted as follows:

- Acceleration > deceleration
  The smoothing time used for the deceleration ramp is shorter than that for the acceleration ramp.
- Acceleration < deceleration
  The smoothing time used for the deceleration ramp is greater than that for the acceleration ramp.
- Acceleration = deceleration
  The smoothing times of the acceleration and deceleration ramp are equal.

The relation between smoothing times and jerk is shown in the following equation:

$$\text{Smoothing time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Smoothing time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Motion jobs started in the user program are performed with the selected jerk.

The limits for jerk with drive connection via PTO (Pulse Train Output) can be found in section CPU outputs relevant for motion control (Page 1300).

For PROFIdrive drives and drives with analog drive interface, the limit is 1E12.

See also

Behavior of the axis when using the jerk limit (Page 1367)

Hardware components for motion control (Page 1298)

CPU outputs relevant for motion control (Page 1300)

Configuration - Dynamics - Emergency stop (Page 1366)

Changing the configuration of dynamics in the user program (Page 1368)

Configuration - Dynamics - Emergency stop (S7-1200)

Configure the emergency stop deceleration of the axis in the "Dynamics emergency stop" configuration window. In the event of an error, and when disabling the axis, the axis is brought to a standstill with this deceleration using the Motion Control instruction "MC_Power" (input parameter StopMode = 0 or 2).

Velocity

The velocity values configured in the "General dynamics" configuration window are once again displayed in this information area.

Deceleration

Set the deceleration value for emergency stop in the "Emergency deceleration" or "Emergency stop ramp-down time" field.

The relation between emergency stop ramp-down time and emergency deceleration is shown in the following equation:

$$\textit{Emergency stop ramp-down time} = \frac{\textit{Maximum velocity} - \textit{Start/stop velocity}}{\textit{Emergency deceleration}}$$

The specified emergency deceleration must be sufficient to bring the axis to a standstill in a timely manner in the event of an emergency (for example, when the hardware limit switch is approached prior to reaching the mechanical endstop).

The configured maximum velocity of the axis must be used as a basis for selecting the emergency deceleration.

Limit values:

The limits indicated below refer to the "Pulses/s^2" unit of measurement.

- As of CPU firmware V3
  0.005 ≤ emergency deceleration ≤ 9.5E9
- CPU Firmware V1...2
  0.28 ≤ emergency deceleration ≤ 9.5E9

The limits for other units of measurement must be converted to conform to the given mechanics.

The limits for jerk with drive connection via PTO (Pulse Train Output) can be found in section CPU outputs relevant for motion control (Page 1300).

For PROFIdrive drives and drives with analog drive interface, the limit is 1.0E12.

See also

Configuration - Dynamics - General (Page 1362)

Changing the configuration of dynamics in the user program (Page 1368)

Behavior of the axis when using the jerk limit (S7-1200)

Axis acceleration and deceleration is not stopped abruptly when the jerk limit is activated; it is adjusted gently according to the set step or smoothing time. The diagram below details the behavior of the axis with and without activated jerk limit:

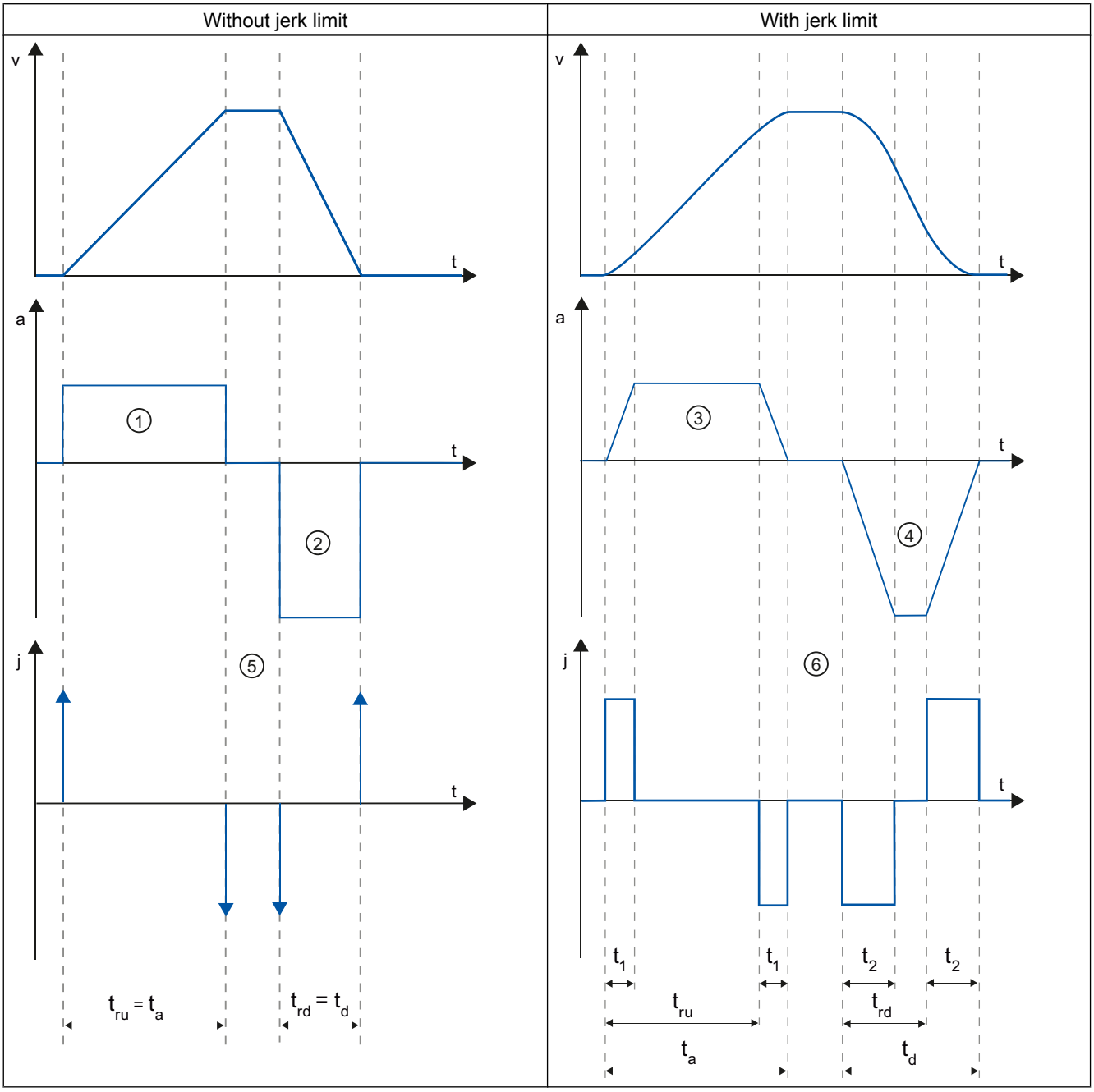

t	Time axis
v	Velocity
a	Acceleration

j	Jerk
t_{ru}	Rampup time
t_a	Time taken for the axis to accelerate
t_{rd}	Deceleration time
t_d	Time taken for the axis to decelerate
t_1	Smoothing time of the acceleration ramp
t_2	Smoothing time of the deceleration ramp

The example shows travel in which the deceleration value ② is twice the acceleration value ①. The resulting ramp-down time t_{rd} is therefore only half the length of the ramp-up time t_{ru}.

Acceleration ① and deceleration ② change abruptly without a jerk limit. Acceleration ① and deceleration ② change gradually with activated jerk limiter. As the jerk applies to entire motion, the rate is the same for the increase and decrease in acceleration and deceleration.

The step value j becomes infinitely high ⑤ as soon as the change is made without jerk limit. The step is limited to the configured value ⑥ when the jerk limit is activated.

The smoothing time t_1 given in the configuration applies to the acceleration ramp. The deceleration ramp smoothing time t_2 is calculated from the configured jerk value and the configured deceleration.

See also

Configuration - Dynamics - General (Page 1362)

Changing the configuration of dynamics in the user program (S7-1200)

You can change the following configuration parameters during runtime of the user program in the CPU:

Acceleration and deceleration

You can also change the values for acceleration and deceleration during runtime of the user program. Use the following technology object variables for this purpose:

- <Axis name>.DynamicDefaults.Acceleration
  for changing acceleration
- <Axis name>.DynamicDefaults.Deceleration
  for changing deceleration

Refer to the description of the technology object variables (Page 1489) in the appendix for information on when changes to the configuration parameters take effect.

Emergency stop deceleration

You can also change the value for the emergency stop deceleration during runtime of the user program. Use the following technology object variable for this purpose:

- <Axis name>.DynamicDefaults.EmergencyDeceleration

Refer to the description of the technology object variables in the appendix for information on when changes to the configuration parameter take effect.

Note

After changes to this parameter, it may be necessary to adapt the positions of the hardware limit switches and other safety-relevant settings.

Jerk limit

You can also activate and deactivate the jerk limit at runtime of the user program and change the value for the jerk. To do this, use the technology object tag <axis name>.DynamicDefaults.Jerk For technology objects < V4, the tag <axis name>.Config.DynamicDefaults.JerkActive must be set to TRUE in order to activate the jerk limitation and in order that a value change is visible/effective at the jerk.

The following applies to PTO axes:

- If you enter a value ≥ 0.004 pulses/s^3 for the jerk, the jerk limit is enabled with the specified value.
- If you enter a value < 0.004 pulses/s^3 for the jerk, the jerk limit is disabled.

For position-controlled axes, the jerk limit is disabled for a value of 0.0, and activated for values > 0.0.

Refer to the description of the technology object variables in the appendix for information on when changes to the configuration parameter take effect.

See also

Changing configuration of dynamic values in user program ("Axis" technology object V1...3) (Page 1539)

Compatibility list of variables V1...3 <-> V4...5 (Page 1331)

Tags of the positioning axis technology object V4...5 (Page 1595)

Configuration - Dynamics - General (Page 1362)

Configuration - Dynamics - Emergency stop (Page 1366)

Homing (positioning axis technology object as of V2) (S7-1200)

Configuration - Homing - Active (S7-1200)

Configure the necessary parameters for active homing in the "Active homing" configuration window. Active homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 3.

Select homing mode (drive connection via PROFIdrive V5 or higher only)

Select one of the following homing modes:

- Use zero mark via PROFIdrive telegram and proximity switch
- Use zero mark via PROFIdrive telegram
- Use homing mark via digital input

If you have selected drive connection via PTO (Pulse Train Output) or analog output with HSC as the encoder, the only homing mode available is "Use homing mark via digital input".

Digital inputs

In this area, you configure the homing switch:

- **Input homing switch**
  Select the digital input for the homing switch in this field.

  Note

  The digital inputs are set to a filter time of 6.4 ms by default.

  When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

  The specified filter time must be less than the duration of the input signal at the homing switch.

  For drive connection via PTO (Pulse Train Output):
  The input must be interrupt-capable. The onboard CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

- **Select level**
  In the drop-down list, select the level of the homing switch that is to be used for homing.

- **Permit auto reverse at HW limit switch**
  Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be enabled for the reversal of direction (at least the hardware limit switch in the direction of approach must be configured).
  If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The homing switch is then sensed in reverse direction.
  If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

  Note

  If possible, use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

  - Keep the approach velocity low.
  - Increase the configured acceleration/deceleration.
  - Increase the distance between the hardware limit switch and the mechanical endstop.

Approach/homing direction

With the direction selection, you determine the approach direction used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the homing switch to carry out the homing operation.

Side of homing switch

This is where you select whether the axis is to be homed on the top or bottom side of the homing switch.

Approach velocity

In this field, specify the velocity at which the homing switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity

In this field, specify the velocity at which the homing switch is to be approached for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset

If the desired home position deviates from the position of the homing switch, the home position offset can be specified in this field.

If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Configuration - Homing - Passive (S7-1200)

Configure the necessary parameters for passive homing in the "Homing - Passive" configuration window.

The movement for passive homing must be triggered by the user (e.g. using an axis motion command). Passive homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 2.

Select homing mode (drive connection via PROFIdrive V5 or higher only)

Select one of the following homing modes:

- Use zero mark via PROFIdrive telegram and proximity switch
  The system checks for when the proximity switch is reached. After the proximity switch is reached and is left again in the assigned homing direction, zero mark detection is enabled via the PROFIdrive telegram. When the zero mark is reached in the pre-selected direction, then the actual position of the technology object is set to the homing mark position.
- Use zero mark via PROFIdrive telegram
  The system enables zero mark detection as soon as the actual value of the technology object moves in the assigned homing direction. When the zero mark is reached in the specified homing direction, the actual position of the technology object is set to the homing mark position.
- Use homing mark via digital input
  The system checks the state of the digital input as soon as the actual value of the axis or encoder moves in the assigned homing direction. When the homing mark is reached (setting of the digital input) in the specified homing direction, the actual position of the technology object is set to the homing mark position.

If you have selected drive connection via PTO (Pulse Train Output), a homing mark via a digital input is used by default.

Digital inputs

In this area, you configure the homing switch:

- **Input homing switch**
  Select the digital input for the homing switch in this field. The input must be interrupt-capable. The onboard CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

  Note

  The digital inputs are set to a filter time of 6.4 ms by default.

  When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

  The specified filter time must be less than the duration of the input signal at the homing switch.

- **Select level**
  In the drop-down list, select the level of the homing switch that is to be used for homing.

Side of homing switch

This is where you select whether the axis is to be homed on the top or bottom side of the homing switch.

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Note

If passive homing is carried out without an axis motion command (axis at a standstill), homing will be executed upon the next rising or falling edge at the homing switch.

Sequence - Active homing (S7-1200)

You start active homing with motion control instruction "MC_Home" (input parameter Mode = 3). The "Position" input parameter specifies the absolute home position. Alternatively, you can start active homing on the axis control panel for test purposes.

The diagram below shows an example of a characteristic curve for an active home position approach with the following configuration parameters:

- "Homing mode" = "Use homing mark via digital input"
- "Approach/homing direction" = "Positive direction"

- "Side of homing switch" = "Top side"
- Value of "home position offset" > 0

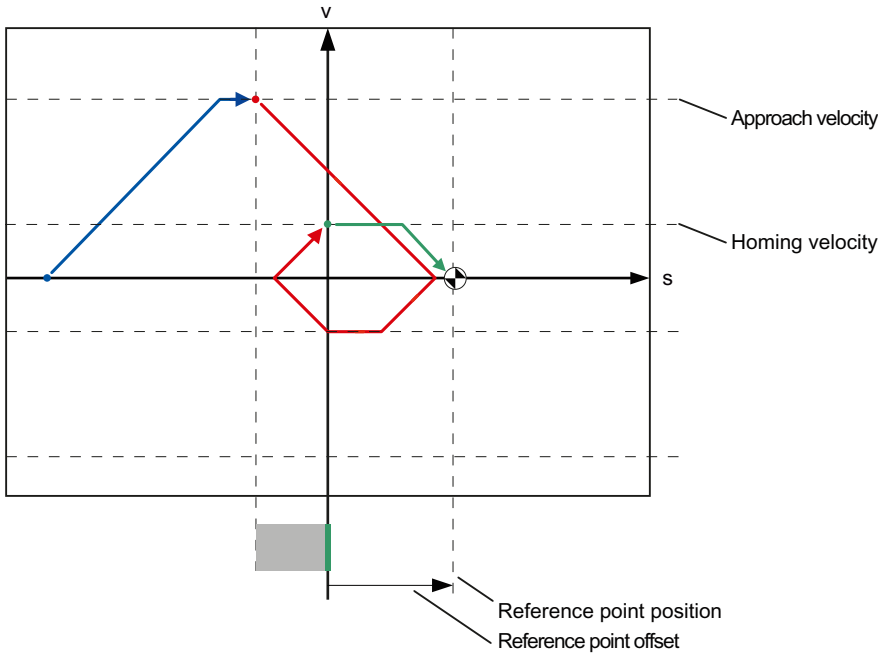

Search for homing switch (blue curve section)

When active homing starts, the axis accelerates to the configured "approach velocity" and searches at this velocity for the homing switch. The tag <axis name>.StatusBits.HomingDone is set to FALSE.

Reference point approach (red curve section)

When the homing switch is detected, the axis in this example brakes and reverses, to be homed to the configured side of the homing switch at the configured homing velocity. Homing causes the tag <axis name>.StatusBits.HomingDone to change to TRUE.

Travel to home position offset (green curve segment)

After homing, the axis moves at the homing velocity along the path to the home position offset. There the axis is at the homing point position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

See also

Configuration - Homing - General (Axis technology object V2...3) (Page 1536)

Sequence - Passive homing (S7-1200)

Passive homing is started with Motion Control instruction "MC_Home" (input parameter Mode = 2). Input parameter "Position" specifies the absolute reference point position.

The diagram below shows an example of a characteristic curve for passive homing with the following configuration parameters:

- "Side of homing switch" = "Top side"
- "Homing mode" = "Use homing mark via digital input"

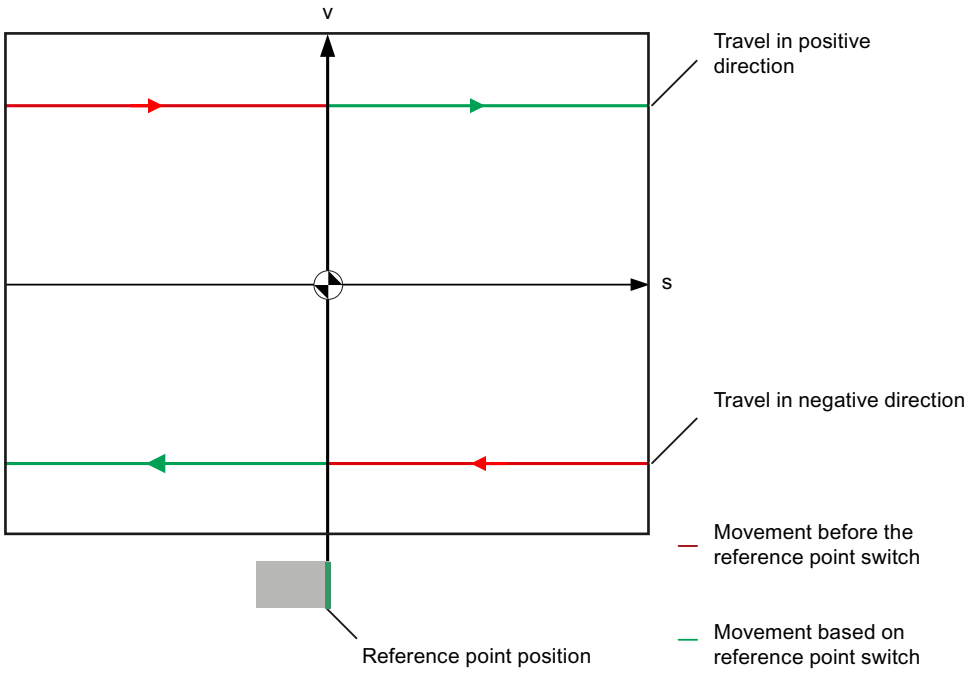

Movement towards homing switch (red section of curve)

The Motion Control instruction "MC_Home" does not itself carry out any homing motion when passive homing is started. The travel required for reaching the homing switch must be implemented by the user via other motion control instructions such as "MC_MoveRelative". The tag <axis name>.StatusBits.HomingDone remains TRUE during passive homing if the axis has already been homed.

Axis homing (transition from red to green section of curve)

The axis is homed when the configured side of the homing switch is reached. The current position of the axis is set to the home position. This is specified at the "Position" parameter of the "MC_Home" Motion Control instruction. The tag <axis name>.StatusBits.HomingDone will be set to "TRUE" if the axis has not been homed before. The travel previously started is not canceled.

Movement beyond homing switch (green section of curve)

Following homing at the homing switch, the axis continues and completes the previously started travel with the corrected axis position.

Changing the homing configuration in the user program (S7-1200)

With positioning axis technology object as of V2, you can change the following configuration parameters during runtime of the user program in the CPU:

Passive homing

You can change the end of the homing switch for passive homing during the user program runtime. Use the following technology object tag for this purpose:

- <Axis name>.Sensor[1].PassiveHoming.SideInput
  for changing the side of the homing switch
- <Axis name>.Sensor[1].PassiveHoming.Mode
  for changing the homing mode

Refer to the description of the technology object tags (Page 1595) in the appendix for information on when changes to the configuration parameter take effect.

Active homing

You can change the direction of approach, the side of the homing switch, the approach velocity, the homing velocity, and the home position offset for active homing during the program runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.Homing.AutoReversal
  for changing the auto reverse at the HW limit switch
- <Axis name>.Homing.ApproachDirection
  for changing the approach/homing direction
- <Axis name>.Sensor[1].ActiveHoming.SideInput
  for changing the side of the homing switch
- <Axis name>.Homing.ApproachVelocity
  for changing the approach velocity
- <Axis name>.Homing.ReferencingVelocity
  for changing the homing velocity
- <Axis name>.Sensor[1].ActiveHoming.HomePositionOffset
  for changing the home position offset
- <Axis name>.Sensor[1].ActiveHoming.Mode
  for changing the homing mode

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

See also

Compatibility list of variables V1...3 <-> V4...5 (Page 1331)

Positioning monitoring (S7-1200)

Configuration - Position monitoring (PROFIdrive and analog drive connection only) (S7-1200)

In the "Position monitoring" configuration window, configure the criteria for monitoring the target position.

Position monitoring monitors the behavior of the actual position at the end of the setpoint calculation. As soon as the setpoint velocity reaches the value zero, the actual position value must be located within a tolerance time in the positioning window. The actual value must not exit the positioning window during the minimum dwell time.

If the actual position reaches the positioning window within the tolerance time and remains in the positioning window for the minimum dwell time, the status bit <axis name>.StatusBits.Done is set. This completes a motion command.

Position monitoring does not make any distinction between how the setpoint interpolation was completed. The end of setpoint interpolation can, for example, be reached as follows:

- By the setpoint reaching the target position
- By position-controlled stopping during the motion through the Motion Control instruction "MC_Halt"

In the following cases, the axis is stopped by the position monitoring and a positioning error (ErrorID 16#800F) is displayed at the Motion Control instruction:

- The actual value does not reach the positioning window within the tolerance time.
- The actual value exits the positioning window during the minimum dwell time.

Positioning window

In this field, configure the size of the positioning window.

Tolerance time

In this field, configure the tolerance time within which the position value must reach the positioning window.

Minimum dwell time in positioning window

In this field, configure the minimum dwell time for which the actual position value must be located in the positioning window.

Configuration - Following error (PROFIdrive and analog drive connection only) (S7-1200)

In the "Following error" configuration window, you configure the permissible deviation of the actual position of the axis from the setpoint position.

The following error is the difference between the setpoint position and the actual position value of the axis. The transmission times of the setpoint to the drive and of the actual value to the controller are taken into account in the calculation of the following error.

The following error is monitored based on a velocity-dependent following error limit. The permissible following error depends on the setpoint velocity.

A constant permissible following error can be specified for velocities lower than an adjustable velocity low limit. Above this low velocity limit, the permissible following error increases in proportion to the setpoint velocity. The maximum following error is permitted at the maximum velocity.

If the permitted following error is exceeded, the axis is stopped and an error (ErrorID 16#800D) is displayed at the Motion Control instruction.

Enable following error monitoring

Select the check box to enable following error monitoring.

When following error monitoring is enabled, the axis is stopped in the error range (orange).

Maximum following error

In this field, configure the following error that is permissible at maximum velocity.

Following error

In this field, configure the permissible following error for low velocities (without dynamic adaptation).

Start dynamic adjustment

In this field, configure the velocity above which the following error should be dynamically adapted. Above this velocity, the following error up to the maximum velocity will be adapted to the maximum following error.

Maximum velocity

This box shows the maximum velocity configured under "Dynamics > General".

Configuration - Standstill signal (PROFIdrive and analog drive connection only) (S7-1200)

In the "Standstill signal" configuration window, configure the criteria for standstill detection.

To display the standstill (<Axis name>.StatusBits.StandStill), the velocity of the axis must remain in the standstill window for the minimum dwell time.

Standstill window

In this field, configure the size of the standstill window.

Minimum dwell time in standstill window

In this field, configure the minimum dwell time in the standstill window.

Configuration - Control loop (PROFIdrive and analog drive connection only) (S7-1200)

In the "Control loop" configuration window, configure the precontrol and the gain Kv of the position control loop.

The Kv factor affects the following parameters:

- Positioning accuracy and stop control
- Uniformity of motion
- Positioning time

The better the mechanical conditions of the axis are (high stiffness), the higher you can configure the Kv factor. This reduces the following error, and a higher dynamic response is achieved.

The "Tuning (Page 1424)" function supports you in determining the optimum gain for the position control of the axis.

Precontrol

In this field, configure the velocity precontrol of the position control loop as a percentage.

Gain (Kv factor)

In this field, you configure the gain Kv of the position control loop.

Parameter view (S7-1200)

Introduction to the parameter view (S7-1200)

The Parameter view provides you with a general overview of all relevant parameters of a technology object. You obtain an overview of the parameter settings and can easily change them in offline and online mode.

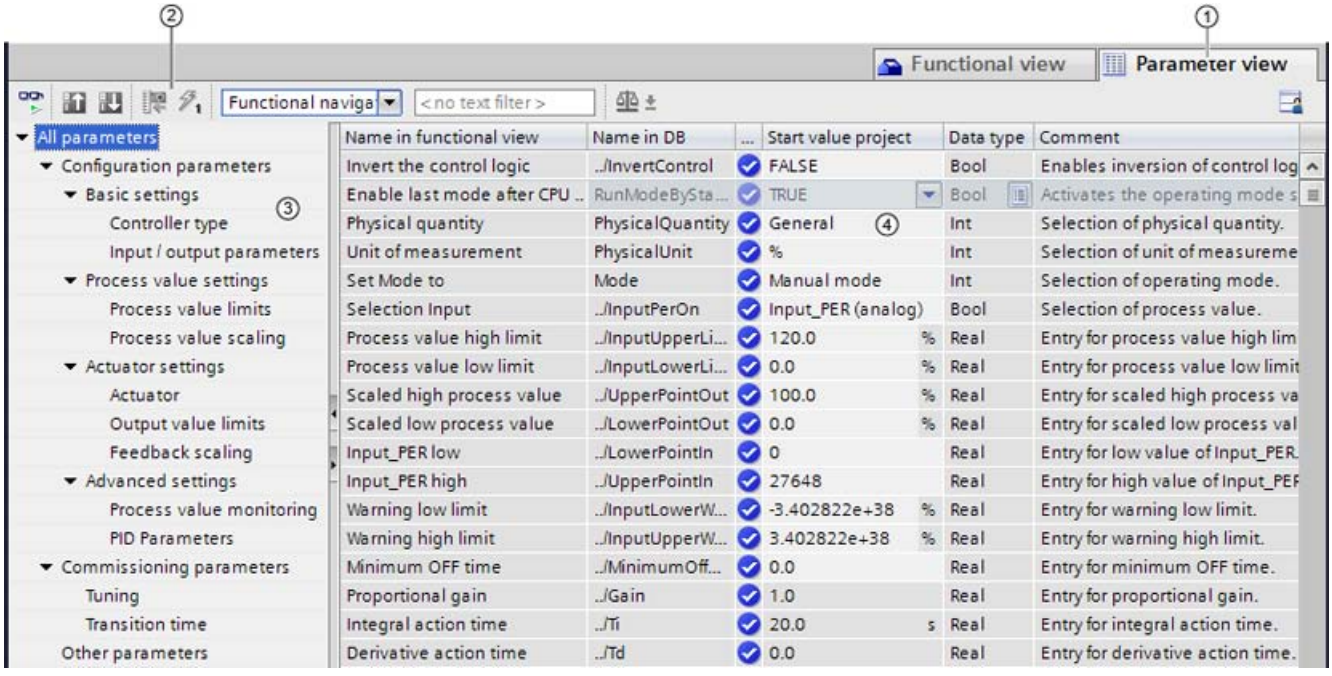

① "Parameter view" tab

② Toolbar (Page 643)

③ Navigation (Page 643)

④ Parameter table (Page 644)

Function scope

The following functions are available for analyzing the parameters of the technology objects and for enabling targeted monitoring and modification.

Display functions:

- Display of parameter values in offline and online mode
- Display of status information of the parameters
- Display of value deviations and option for direct correction
- Display of configuration errors
- Display of value changes as a result of parameter dependencies

- Display of all memory values of a parameter: Start value PLC, Start value project, Monitor value
- Display of the parameter comparison of the memory values of a parameter

Operator control functions:

- Navigation for quickly changing between the parameters and parameter structures.
- Text filter for faster searches for particular parameters.
- Sorting function for customizing the order of parameters and parameter groups to requirements.
- Memory function for backing up structural settings of the Parameter view.
- Monitoring and modifying of parameter values online.
- Change display format of value.
- Function for saving a snapshot of parameter values of the CPU in order to capture momentary situations and to respond to them.
- Function for applying a snapshot of parameter values as start values.
- Download of modified start values to the CPU.
- Comparison functions for comparing parameter values with one another.

Validity

The Parameter view described here is available for the following technology objects:

- PID_Compact
- PID_3Step
- PID_Temp
- CONT_C (S7-1500 only)
- CONT_S (S7-1500 only)
- TCONT_CP (S7-1500 only)
- TCONT_S (S7-1500 only)
- TO_Axis_PTO (S7-1200 Motion Control)
- TO_Positioning_Axis (S7-1200 Motion Control)
- TO_CommandTable_PTO (S7-1200 Motion Control)
- TO_CommandTable (S7-1200 Motion Control)

Structure of the parameter view (S7-1200)

Toolbar (S7-1200)

The following functions can be selected in the toolbar of the parameter view.

Icon	Function	Explanation
	Monitor all	Starts the monitoring of visible parameters in the active Parameter view (online mode).
	Create snapshot of monitor values and accept setpoints of this snapshot as start values	Applies the current monitor values to the “Snapshot” column and updates the start values in the project. Only in online mode for PID_Compact, PID_3Step and PID_Temp.
	Initialize setpoints	Transfers the start values updated in the project to the CPU. Only in online mode for PID_Compact, PID_3Step and PID_Temp.
	Create snapshot of monitor values	Applies the current monitor values to the “Snapshot” column. Only in online mode.
	Modify all selected parameters immediately and once	This command is executed once and as quickly as possible without reference to any particular point in the user program. Only in online mode.
Functional naviga	Select navigation structure	Toggles between functional navigation and data navigation.
< no text filter >	Text filter...	After entry of a character string: Display of all parameters containing the specified string in one of the currently visible columns.
	Selection of compare values	Selection of parameter values that are to be compared with one another in online mode (Start value project, Start value PLC, Snapshot) Only in online mode.
	Save window settings	Saves your display settings for the Parameter view (e.g., selected navigation structure, activated table columns, etc.)

Navigation (S7-1200)

Within the "Parameter view" tab, the following alternative navigation structures can be selected.

Navigation		Explanation
Functional navigation	All parameters Configuration parameters Commissioning parameters Other parameters	In the functional navigation, the structure of the parameters is based on the structure in the configuration dialog ("Functional view" tab), commissioning dialog, and diagnostics dialog. The last group "Other parameters" contains all other parameters of the technology object.
Data navigation	All parameters Input Output InOut Static Other parameters	In the data navigation, the structure of the parameters is based on the structure in the instance DB / technology DB. The last group "Other parameters" contains the parameters that are not contained in the instance DB / technology DB.

You can use the "Select navigation structure" drop-down list to toggle the navigation structure.

Parameter table (S7-1200)

The table below shows the meaning of the individual columns of the parameter table. You can show or hide the columns as required.

- Column "Offline" = X: Column is visible in offline mode.
- Column "Online" = X: Column is visible in online mode (online connection to the CPU).

Column	Explanation	Offline	Online
Name in functional view	Name of the parameter in the functional view. The display field is empty for parameters that are not configured via the technology object.	X	X
Full name in DB	Complete path of the parameter in the instance DB / technology DB. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Name in DB	Name of the parameter in the instance DB / technology DB. If the parameter is part of a structure or UDT, the prefix ". ./" is added. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Status of configuration	Display of the completeness of the configuration using status symbols. see Status of configuration (offline) (Page 652)	X	
Compare result	Result of the "Compare values" function. This column is shown if there is an online connection and the "Monitor all" button is selected.		X
Start value project	Configured start value in the project. Error indication if entered values have a syntax or process-related error.	X	X
Default value	Value that is pre-assigned to the parameter. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Snapshot	Snapshot of the current values in the CPU (monitor values). Error indication if values have a process-related error.	X	X
Start value PLC	Start value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Monitor value	Current value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Modify value	Value that is to be used to change the monitor valuet. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if entered values have a syntax or process-related error.		X
Selection for transmission	Selection of the Modify values that are to be transmitted using the "Modify all selected parameters immediately and once" button. This column is displayed together with the "Modify value" column.		X

Column	Explanation	Offline	Online
Minimum value	Minimum process-related value of the parameter. If the minimum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Maximum value	Maximum process-related value of the parameter. If the maximum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Setpoint	Designates the parameter as a setpoint. These parameters can be initialized online.	X	X
Data type	Data type of the parameter. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Retain	Designates the value as a retentive value. The values of retentive parameters are retained even after the voltage supply is switched off.	X	X
Accessible from HMI	Indicates whether the HMI can access this parameter during runtime.	X	X
Visible in HMI	Indicates whether the parameter is visible in the selection list of the HMI by default.	X	X
Comment	Brief description of the parameter.	X	X

See also

Comparing values (Page 638)

Opening the parameter view (S7-1200)

Requirement

The technology object has been added in the project tree, i.e., the associated instance DB / technology DB of the instruction has been created.

Procedure

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.
4. Select the "Parameter view" tab in the top right corner.

Result

The Parameter view opens. Each displayed parameter is represented by one row in the parameter table.

The displayable parameter properties (table columns) vary depending on whether you are working with the Parameter view in offline or online mode.

In addition, you can selectively display and hide individual table columns.

See also

Default setting of the parameter view (Page 646)

Default setting of the parameter view (S7-1200)

Default settings

To enable you to work efficiently with the Parameter view, you can customize the parameter display and save your settings.

The following customizations are possible and can be saved:

- Show and hide columns
- Change column width
- Change order of the columns
- Toggle navigation
- Select parameter group in the navigation
- Selection of compare values

Show and hide columns

To show or hide columns in the parameter table, follow these steps:

1. Position the cursor in the header of the parameter table.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the check box for the column.
4. To hide a column, clear the check box for the column.

or

1. Position the cursor in the header of the parameter table.
2. Select the "Show all columns" command in the shortcut menu if all columns of the offline or online mode are to be displayed.

Some columns can only be displayed in online mode: see Parameter table (Page 644).

Change column width

To customize the width of a column so that all texts in the rows can be read, follow these steps:

1. Position the cursor in the header of the parameter table to the right of the column to be customized until the shape of the cursor changes to a cross.
2. Then double-click this location.

or

1. Open the shortcut menu on the header of the parameter table.
2. Click
   - "Optimize column width" or
   - "Optimize width of all columns".

If the column width setting is too narrow, the complete content of individual fields are shown if you hover the cursor briefly over the relevant field.

Change order of the columns

The columns of the parameter table can be arranged in any way.

To change the order of the columns, follow these steps:

1. Click on the column header and use a drag-and-drop operation to move it to the desired location.
   When you release the mouse button, the column is anchored to the new position.

Toggle navigation

To toggle the display form of the parameters, follow these steps:

1. Select the desired navigation in the “Select navigation structure” drop-down list.
   - Data navigation
   - Functional navigation

See also Navigation (Page 643).

Select parameter group in the navigation

Within the selected navigation, you choose between the “All parameters” display or the display of a subordinate parameter group of your choice.

1. Click the desired parameter group in the navigation.
   The parameter table only displays the parameters of the parameter group.

Selection of compare values (online)

To set the compare values for the “Compare values” function, follow these steps:

1. Select the desired compare values in the “Selection of compare values” drop-down list.
   - Start value project / Start value PLC
   - Start value project / Snapshot
   - Start value PLC / Snapshot

The “Start value project / Start value PLC” option is set by default.

Saving the default setting of the Parameter view

To save the above customizations of the Parameter view, follow these steps:

1. Customize the Parameter view according to your requirements.
2. Click the “Save window settings” button at the top right of the Parameter view.

Working with the parameter view (S7-1200)

Overview (S7-1200)

The following table provides an overview of the functions of the Parameter view in online and offline mode described in the following.

- Column "Offline" = X: This function is possible in offline mode.
- Column "Online" = X: This function is possible in online mode.

Function/action	Offline	Online
Filtering the parameter table (Page 649)	X	X
Sorting the parameter table (Page 649)	X	X
Transferring parameter data to other editors (Page 650)	X	X
Indicating errors (Page 650)	X	X
Editing start values in the project (Page 651)	X	X
Status of configuration (offline) (Page 652)	X	
Monitoring values online in the parameter view (Page 652)		X
Create snapshot of monitor values (Page 654)		X
Modifying values (Page 655)		X
Comparing values (Page 656)		X
Applying values from the online program as start values (Page 657)		X
Initializing setpoints in the online program (Page 658)		X

Filtering the parameter table (S7-1200)

You can filter the parameters in the parameter table in the following ways:

- With the text filter
- With the subgroups of the navigation

Both filter methods can be used simultaneously.

With the text filter

Texts that are visible in the parameter table can be filtered. This means only texts in displayed parameter rows and columns can be filtered.

1. Enter the desired character string for filtering in the “Text filter...” input box.
   The parameter table displays only the parameters containing the character string.

The text filtering is reset.

- When another parameter group is selected in the navigation.
- When navigation is changed from data navigation to functional navigation, or vice versa.

With the subgroups of the navigation

1. Click the desired parameter group in the navigation, e.g., "Static".
   The parameter table only shows the static parameters. You can select further subgroups for some groups of the navigation.
2. Click “All parameters” in the navigation if all parameters are to be shown again.

Sorting the parameter table (S7-1200)

The values of the parameters are arranged in rows. The parameter table can be sorted by any displayed column.

- In columns containing numerical values, sorting is based on the magnitude of the numerical value.
- In text columns, sorting is alphabetical.

Sorting by column

1. Position the cursor in the header cell of the desired column.
   The background of this cell turns blue.
2. Click the column header.

Result

The entire parameter table is sorted by the selected column. A triangle with tip facing up appears in the column header.

Clicking the column header again changes the sorting as follows:

- Symbol "▲": Parameter table is sorted in ascending order.
- Symbol "▼": Parameter table is sorted in descending order.
- No symbol: The sorting is removed again. The parameter table assumes the default display.

The "../" prefix in the "Name in DB" column is ignored when sorting.

Transferring parameter data to other editors (S7-1200)

After selecting an entire parameter row of the parameter table, you can use the following:

- Drag-and-drop
- <Ctrl+C>/<Ctrl+V>
- Copy/Paste via shortcut menu

Transfer parameters to the following editors of the TIA Portal:

- Program editor
- Watch table
- Signal table for trace function

The parameter is inserted with its full name: See information in "Full name in DB" column.

Indicating errors (S7-1200)

Error indication

Parameter assignment errors that result in compilation errors (e.g. limit violation) are indicated in the Parameter view.

Every time a value is input in the Parameter view, a check is made for process-related and syntax errors and the result is indicated.

Bad values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Table field with red background
  If you click the bad field, a roll-out error message appears with information of the permissible value range or the required syntax (format)

Compilation error

From the error message of the compiler, you can directly open the Parameter view (functional navigation) containing the parameter causing the error in situations where the parameter is not displayed in the configuration dialog.

Editing start values in the project (S7-1200)

With the Parameter view, you can edit the start values in the project in offline mode and online mode.

- You make value changes in the “Start value project” column of the parameter table.
- In the “Status of configuration” column of the parameter table, the progress of the configuration is indicated by the familiar status symbols from the configuration dialog of the technology object.

Boundary conditions

- If other parameters depend on the parameter whose start value was changed, the start value of the dependent parameters are also adapted.
- If a parameter of a technology object is not editable, it is also not editable in the parameter view. The ability to edit a parameter can also depend on the values of other parameters.

Defining new start values

To define start values for parameters in the Parameter view, follow these steps:

1. Open the Parameter view of the technology object.
2. Enter the desired start values in the "Start value project" column. The value must match the data type of the parameter and must not exceed the value range of the parameter. The limits of the value range can be seen in the “Maximum value” and “Minimum value” columns.

The "Status of configuration" column indicates the progress of the configuration with colored symbols.

See also Status of configuration (offline) (Page 652)

Following adaptation of the start values and downloading of the technology object to the CPU, the parameters take the defined value at startup if they are not declared as retentive (“Retain” column).

Error indication

When a start value is input, a check is made for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Red background in the “Start value project” field
  If you click on the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Correcting bad start values

1. Correct bad start values using information from the roll-out error message.
   Red error symbol, red field background, and roll-out error message are no longer displayed.

The project cannot be successfully compiled unless the start values are error-free.

Status of configuration (offline) (S7-1200)

The status of the configuration is indicated by icons:

- In the “Status of configuration” column in the parameter table
- In the navigation structure of the functional navigation and data navigation

Symbol in “Status of configuration” column

Symbol	Meaning
	The start value of the parameter corresponds to the default value and is valid. A start value has not yet been defined by the user.
	The start value of the parameter contains a value defined by the user or an automatically adjusted value. The start value is different than the default value. The start value is error-free and valid.
	The start value of the parameter is invalid (syntax or process-related error). The input box has a red background. When clicked, the roll-out error message indicates the cause of the error.
	Only for S7-1200 Motion Control: The start value of the parameter is valid but contains warnings. The input box has a yellow background.
	The parameter is not relevant in the current configuration.

Symbol in the navigation

The symbols in the navigation indicate the progress of the configuration in the same way as in the configuration dialog of the technology object.

See also

Configure technology objects (Page 634)

Monitoring values online in the parameter view (S7-1200)

You can monitor the values currently taken by the parameters of the technology object in the CPU (monitor values) directly in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.

- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

Procedure

1. Start the monitoring by clicking .
   As soon as the Parameter view is online, the following columns are additionally displayed:
   – Compare result
   – Start value PLC
   – Monitor value
   – Modify value
   – Selection for transmission

   The "Monitor value" column shows the current parameter values on the CPU.
   Meaning of the additional columns: see Parameter table (Page 644)
2. Stop the monitoring by clicking again.

Display

All columns that are only available online have an orange background:

- Values in light-orange cells can be changed.
- Values in cells with a dark orange background cannot be changed.

Change display format of value (S7-1200)

The display format of the value can be selected via the shortcut menu of a table row in the Parameter view of the technology object.

The display format of the following values can be changed both in online mode and in offline mode:

- Start value project
- Start value PLC
- Maximum value
- Minimum value
- Snapshot
- Monitor value
- Default value
- Modify value

The set display format applies to all values of the table row.

The following display formats of the value can be changed:

- Default
- Hex
- Octal
- Bin
- Dec (+/-)
- DEC

Depending on the parameter selected in the parameter view, only the supported display formats can be selected.

Requirements

- The Parameter view of the technology object is open.

Procedure

To change the display format of the value, proceed as follows:

1. Select one or more table rows in which you want to change the display format.
2. Select the "Display format" command in the shortcut menu.
3. Select the desired display format.

Note

To change the display format of a certain data type in multiple table rows, sort the Parameter view by this data type. Then select the first and last table row with this data type while keeping the <Shift> key pressed and change the display format for the selected table rows.

Create snapshot of monitor values (S7-1200)

You can back up the current values of the technology object on the CPU (monitor values) and display them in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To show the current parameter values, follow these steps:

1. In the Parameter view, click the “Create snapshot of monitor values" icon .

Result

The current monitor values are transferred once to the "Snapshot" column of the parameter table.

You can analyze the values "frozen" in this way while the monitor values continue to be updated in the "Monitor values" column.

Modifying values (S7-1200)

With the Parameter view, you can modify values of the technology object in the CPU.

You can assign values to the parameter once (Modify value) and modify them immediately. The modify request is executed as quickly as possible without reference to any particular point in the user program.

DANGER

Danger when modifying:

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you use the "Modify" function.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.
- The parameter can be modified (associated field in the "Modify value" column has a light-orange background).

Procedure

To modify parameters immediately, follow these steps:

1. Enter the desired modify values in the “Modify values” column of the parameter table.
2. Check whether the check box for modifying is selected in the "Select for transmission" column.
   The modify values and associated check boxes of dependent parameters are automatically adapted at the same time.
3. Click the “Modify all selected parameters immediately and once” icon .

The selected parameters are modified once and immediately with the specified values and can be monitored in the "Modify values" column. The check boxes for modifying in the "Selection for transmission" column are automatically cleared after the modify request is complete.

Error indication

When a start value is input, a check is made immediately for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red background in the “Modify value” field

and

- If you click the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Bad modify values

- Modify values with process-related errors can be transmitted.
- Modify values with syntax errors **cannot** be transmitted.

Comparing values (S7-1200)

You can use comparison functions to compare the following memory values of a parameter:

- Start value project
- Start value PLC
- Snapshot

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To compare the start values on the various target systems, follow these steps:

1. Click the "Selection of compare values" icon .
   A selection list containing the comparison options opens:
   - Start value project - Start value PLC (default setting)
   - Start value project - Snapshot
   - Start value PLC - Snapshot
2. Select the desired comparison option.
   The selected comparison option is executed as follows:
   - A scales symbol appears in the header cells of the two columns selected for comparison.
   - Symbols are used in the "Compare result" column to indicate the result of the comparison of the selected columns.

Symbol in "Compare result" column

Symbol	Meaning
	The compare values are equal and error-free.
	The compare values are not equal and error-free.
	At least one of the two compare values has a process-related or syntax error.
	The comparison cannot be performed. At least one of the two comparison values is not available (e.g. snapshot).
	Comparison of the value is inappropriate since it is not relevant in one of the configurations.

Symbol in the navigation

The symbols are shown in the same way in the navigation if the comparison result applies to at least one of the parameters below the displayed navigation structure.

Applying values from the online program as start values (S7-1200)

In order to apply optimized values from the CPU to the project as start values, you create a snapshot of the monitor values. Values of the snapshot marked as a "Setpoint" are then applied to the project as start values.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").

- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To apply optimized values from the CPU, follow these steps:

1. Click the "Create snapshot of monitor values and accept setpoints of this snapshot as start values" icon .

Result

The current monitor values are applied to the "Snapshot" column and their setpoints are copied to the "Start value project" column as new start values.

Note

Applying values of individual parameters

You can also apply the values of individual parameters that are not marked as a setpoint from the "Snapshot" column to the "Start values project" column. To do so, copy the values and insert them into the "Start value project" column using the "Copy" and "Paste" commands in the shortcut menu.

Initializing setpoints in the online program (S7-1200)

You can initialize all parameters that are marked as a "Setpoint" in the Parameter view with new values in the CPU in one step. In so doing, the start values are downloaded from the project to the CPU. The CPU remains in "RUN" mode.

To avoid data loss on the CPU during a cold restart or warm restart, you must also download the technology object to the CPU.

DANGER

Danger when changing parameter values

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you reinitialize the setpoints.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").

- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.
- The parameters marked as a "Setpoint" have a "Start value project" that is free of process-related and syntax errors

Procedure

To initialize all setpoints, follow these steps:

1. Enter the desired values in the "Start value project" column.
   Ensure that the start values are free of process-related and syntax errors.
2. Click the "Initialize setpoints" icon .

Result

The setpoints in the CPU are initialized with the start values from the project.

Configuring technology modules for Motion Control (S7-1200)

Overview (S7-1200)

The following technology modules can be used as the encoder connection in S7-1200 Motion Control.

ET 200MP	ET 200 SP
TM Count 2x24V (Page 1400)	TM Count 1x24V (Page 1400)
TM PosInput 2 (Page 1398)	TM PosInput 1 (Page 1398)

Technology modules can be used centrally or distributed in the system.

The following section describes how to configure the technology modules as encoder:

See also

TM Count 1x24V / TM Count 2x24V (Page 1400)

TM PosInput 1 / TM PosInput 2 (Page 1398)

TM PosInput 1 / TM PosInput 2 (S7-1200)

For use with S7-1200 Motion Control, the following parameters must be configured:

Configuration	
Technology module	Technology object
TM PosInput 1 / TM PosInput 2	Axis
Basic parameters > Channel 0/1 > Operating mode	–
Select "Position input for Motion Control" mode	

Configuration	
Technology module **TM PosInput 1 / TM PosInput 2**	**Technology object** **Axis**
Basic parameters > Channel 0/1 > Module parameters	**Basic parameters > Encoder**
–	Select encoder and corresponding channel
Signal type • Incremental encoder • Absolute encoder	Select encoder type corresponding to configuration for technology module • Incremental • Absolute
–	**Data exchange with encoder**
	Telegram "DP_TEL83_STANDARD" is automatically selected after the selection of the encoder.
	Disable "Automatic transfer of encoder parameter values in the device"
	Select rotary or linear measuring system type
Signal evaluation • Single • Double • Quadruple	Select fine resolution corresponding to configuration for technology module • Incremental encoder: – 0 = Single – 1 = Double *or* – 2 = Quadruple • Absolute encoder: – 0 (= single)
Enter increments per revolution	• Rotary type: Enter increments per revolution corresponding to configuration at technology module (1:1) • Linear type: Enter distance per revolution
–	**Hardware interface > Data exchange with drive**
• Rotary type: Enter reference speed corresponding to configuration for technology object (1:1) • Linear type: Configuration not relevant	Enter reference speed
I/O addresses	–
Organization block: MC-Servo	
Process image: PIP OB servo	
Select "Isochronous mode"	

"–" No configuration for technology module/technology object is required for these parameters

See also

Overview (Page 1398)

TM Count 1x24V / TM Count 2x24V (Page 1400)

TM Count 1x24V / TM Count 2x24V (S7-1200)

For use with S7-1200 Motion Control, the following parameters must be configured:

Configuration	
Technology module TM Count 1x24V / TM Count 2x24V	Technology object Axis
Basic parameters > Channel 0/1 > Operating mode	–
Select "Position input for Motion Control" mode	
Basic parameters > Channel 0/1 > Module parameters	**Hardware interface > Encoder**
–	Select encoder and corresponding channel
Signal type • Incremental encoder	Select encoder type corresponding to configuration for technology module • Incremental
–	**Hardware interface > Data exchange with encoder**
	Telegram "DP_TEL83_STANDARD" is automatically selected after the selection of the encoder.
	Disable "Automatic transfer of encoder parameter values in the device"
	Select rotary or linear measuring system type
Signal evaluation • Single • Double • Quadruple	Select fine resolution corresponding to configuration for technology module • 0 = Single • 1 = Double • 2 = Quadruple
Enter increments per revolution	• Rotary type: Enter increments per revolution corresponding to configuration at technology module (1:1) • Linear type: Enter distance per increment
–	**Hardware interface > Data exchange with drive**
• Rotary type: Enter reference speed corresponding to configuration for technology object (1:1) • Linear type: Configuration not relevant	Enter reference speed
I/O addresses	–
Organization block: MC-Servo	
Process image: PIP OB servo	
Select "Isochronous mode"	

See also

Overview (Page 1398)

TM PosInput 1 / TM PosInput 2 (Page 1398)

6.1.1.6 Technology object command table (S7-1200)

Use of the command table technology object (S7-1200)

The technology object "Command table" allows you to combine multiple individual axis control commands in one movement sequence. The technology object can be used as of technology version V2 for axes with drive connection via PTO (Pulse Train Output).

You configure the movement sequence as a table in a configuration dialog.

The motion profile of the movement sequences can be checked on a graph before the project is loaded to the CPU. The command tables created are then linked to an axis and used in the user program with the "MC_CommandTable" Motion Control instruction. You can process part or all of the command table.

Command table technology object tools (S7-1200)

The "Configuration" tool is provided in the TIA Portal for the "Command Table" technology object. The representation below shows the interaction of the tool with the technology object:

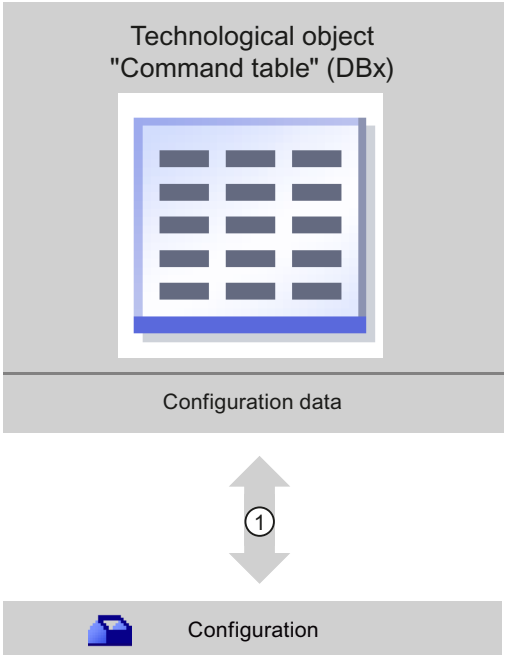

①	Writing and reading the configuration of the technology object

Configuration

Configure the following properties of the "Command Table" technology object with the "Configuration" tool:

- You can create one or more movement sequences by configuring individual jobs.
- You can configure the graphic display to check your movement sequence using an axis already configured or a configurable default axis.

The movement sequence data are saved in the data block of the technology object.

Adding the technological object command table (S7-1200)

Requirements

- A project with a CPU S7-1200 has been created.
- The CPU firmware version is V2.1 or higher

Procedure

Proceed as follows to add a "Command table" technology object in the project tree:

1. Open the "CPU > Technology objects" folder in the project tree.
2. Double-click the "Add new object" command.
   The "Add new object" dialog opens.
3. Select the "Motion Control" technology.
4. Open the "Motion Control" folder.
5. Select the desired technology version in the "Version" column.
6. Select the "TO_CommandTable" object.
7. Enter the name of the command table in the "Name" input box.
8. To change the automatically assigned data block number, select the "Manual" option.
9. To display additional information about the technology object, click "Additional information".
10. Confirm your entry with "OK".

Result

The new technology object is created and saved to the "Technology objects" folder in the project tree.

Configuring the command table technology object (S7-1200)

Working with the configuration dialog (S7-1200)

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the group of the required technology object in the project tree.
2. Double-click the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters contain all parameters which must be configured for a functional command table.
- **Extended parameters**
  The extended parameters contain the parameters of the default axis or display the parameter values of the axis selected.

Configuration window icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values you can use the technology object without additional changes.
	The configuration contains user-defined or automatically adapted values and is complete. All input fields of the configuration contain valid values and at least one preset value has changed.
	The configuration is incomplete or incorrect At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to display the cause of the error.
	The configuration contains mutually incompatible parameter values The configuration contains parameter values that contradict each other either in size or logic. The corresponding field or the drop-down list is displayed on a yellow background.

See also

Guidelines on use of motion control (Page 1326)

Basic parameters (Page 1404)

Extended parameters (Page 1417)

Monitor values (S7-1200)

If there is an online connection to the CPU, the icon "Monitor all" is displayed in the configuration dialog of the technology object.

The "Monitor all" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of the actual values in the project by manual transfer to the start value of the project

Icons and operator controls

If there is an online connection to the CPU, the actual values are displayed at the parameters.

In addition to the actual values of the parameters, the following symbols appear:

Icon	Description
	Start value in CPU matches the configured Start value in the project
	Start value in CPU does not match the configured Start value in the project
	A comparison of the start value in the CPU with the configured start value in the project cannot be performed because the selected CPU module does not support this comparison.
	The value is not comparable with any significance since it is not relevant in one of the configurations.
	Use the button to show the start value of the CPU and the start value of the project for the respective parameter.

The actual value and the start value in the project can be changed directly and then downloaded to the CPU. The change of the actual value is transferred directly to the CPU for directly modifiable parameters.

Basic parameters (S7-1200)

Configuration - General (S7-1200)

Configure the name of the technology object in the "General" configuration window.

Name

Define the name of the command table or the name of the "Command table" technology object in this field. The technology object is listed under this name in the project tree.

See also

Configuration - Command table (Page 1405)

Shortcut menu commands - Command table (Page 1408)

Working with the trend diagram (Page 1409)

Shortcut menu commands - Curve chart (Page 1413)

Transition from "Complete command" to "Blend motion" (Page 1414)

Changing the command table configuration in the user program (Page 1416)

Configuration - Command table (S7-1200)

Create the desired movement sequence in the "Command Table" configuration window and check the result against the graphic view in the trend diagram.

Note

Small deviations are possible between the time behavior and position in the trend shown and the real movement of the axis. Movements in response to software limit switches being reached are not shown.

Enable warnings

Activate the display of warnings in the command table with this check box.

Use axis parameters of

From the drop-down list, select which axis parameters are to be used for selecting the graphic view of and checking the movement sequence. Select "Default axis" if you have yet to add an axis to the "Technology object" folder or wish to use values which have not been configured in any of the available axes. You configure the properties of the default axis under "Advanced parameters".

The axis parameters of the axis selected at the "Axis" parameter are used to process the command table in the user program.

Column: Step

Shows the step number of the command.

Column: Command type

In this column, select the command types which are to be used for processing the command table. Up to 32 commands can be entered. The commands will be processed in sequence. You can choose between the following entries and command types:

- **Empty**
  The entry serves as a placeholder for any commands to be added. The empty entry is ignored when the command table is processed.
- **Halt**
  Stop axis
  (the command only takes effect after a "Velocity set point" command)
- **Positioning Relative**
  Position axis relatively
- **Positioning Absolute**
  Position axis absolutely (the axis must be homed for this)
- **Velocity set point**
  Move axis at set velocity

- **Wait**
  Waits until the given period is over. Wait does not stop active travel.
- **Separator**
  Adds a Separator line above the selected line. The Separator line acts as a range limit for the graphic display of the trend view.
  Use the Separator lines if you wish to process parts of the command table.

Column: Position/travel path

Enter the position or travel path for the selected command in this column:

- **Command "Positioning Relative"**
  The command will move the axis by the given travel path.
- **Command "Positioning Absolute"**
  The command will move the axis by the given position.
  The axis must be homed for this.
- **Separator**
  The value given specifies the start position for the graphic display.

Limit values (independent of the selected user unit):

- -1.0e12 ≤ position / distance ≤ -1.0e-12
- 1.0e-12 ≤ position / distance ≤ 1.0e12
- Position / travel path = 0.0

Column: Velocity

In this column, you enter the velocity for the selected command:

- **Command "Positioning Relative"**
  The command will move the axis at the given velocity.
  The given velocity will not be reached if the travel path selected is not large enough.
- **Command "Positioning Absolute"**
  The command will move the axis at the given velocity.
  The given velocity will not be reached if the target position is too close to the starting position.
- **Command " Velocity set point"**
  The command will move the axis at the given velocity.
  The given velocity will not be reached during the command if too short a runtime is selected.

Limit values (independent of the selected user unit):

- For the commands: "Positioning Relative" and "Positioning Absolute"
  - 1.0e-12 ≤ velocity ≤ 1.0e12
- For the command: "Velocity set point"
  - -1.0e12 ≤ velocity ≤ -1.0e-12
  - 1.0e-12 ≤ velocity ≤ 1.0e12
  - Velocity = 0.0

Column: Duration

Enter the duration of the selected command in this column:

- **Command " Velocity set point"**
  The command will move the axis for the specified duration. The duration includes both the acceleration phase and the constant travel phase. The next command will be processed once the duration is over.
- **Command "Wait"**
  Waits until the given duration is over.

Limit values (independent of the selected user unit):

- 0.001s ≤ duration ≤ 64800s

Column: Next step

Select the mode of transition to the next step from the drop-down list:

- **Complete command**
  The command will be completed. The next command will be processed immediately.
- **Blend motion**
  The motion of the current command will be blended with the motion of the following command. The transition mode "Blend motion" is available with command types "Positioning Relative" and "Positioning Absolute".
  Motion will be blended with motions of the following command types:
  - Positioning Relative
  - Positioning Absolute
  - Velocity set point

No blending occurs with other command types.

For a detailed description of the response of the axis when a command is appended or overlapped, see section Transition from "Complete command" to "Blend motion" (Page 1414)

Column: Step code

Enter a numerical value / bit pattern in this column which is to be output at the "StepCode" output parameter of the "MC_CommandTable" Motion Control instruction while the command is being processed.

Limit values:

- 0 ≤ code number ≤ 65535

See also

Configuration - General (Page 1404)

Shortcut menu commands - Command table (Page 1408)

Working with the trend diagram (Page 1409)

Shortcut menu commands - Curve chart (Page 1413)

Transition from "Complete command" to "Blend motion" (Page 1414)

Changing the command table configuration in the user program (Page 1416)

Shortcut menu commands - Command table (S7-1200)

The following shortcut menu commands are available in the command table:

Insert empty line

Adds an empty line above the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Add empty line

Adds an empty line below the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Insert separator line

Adds a separator line above the selected line.

You cannot have two consecutive separator lines.

Add separator line

Adds a separator line below the selected line.

You cannot have two consecutive separator lines, nor can you add a separator line at the end of the command table.

Cut

Removes the selected lines or content of the selected cell and saves them/it in the clipboard.

Selected lines will be deleted and the subsequent lines of the command table shifted up.

Copy

Copies the selected lines or content of the selected cell and saves them/it in the clipboard.

Paste

- Selected lines:
  Pastes the lines from the clipboard into the table above the selected line.
- Selected cell:
  Pastes the content of the clipboard into the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Replace

Replaces the selected lines with the lines in the clipboard.

Delete

Deletes the selected lines. The lines below in the command table shift up.

See also

Configuration - General (Page 1404)

Configuration - Command table (Page 1405)

Working with the trend diagram (Page 1409)

Shortcut menu commands - Curve chart (Page 1413)

Transition from "Complete command" to "Blend motion" (Page 1414)

Changing the command table configuration in the user program (Page 1416)

Working with the trend diagram (S7-1200)

The following tools and information are available in the trend view:

Trend view and components

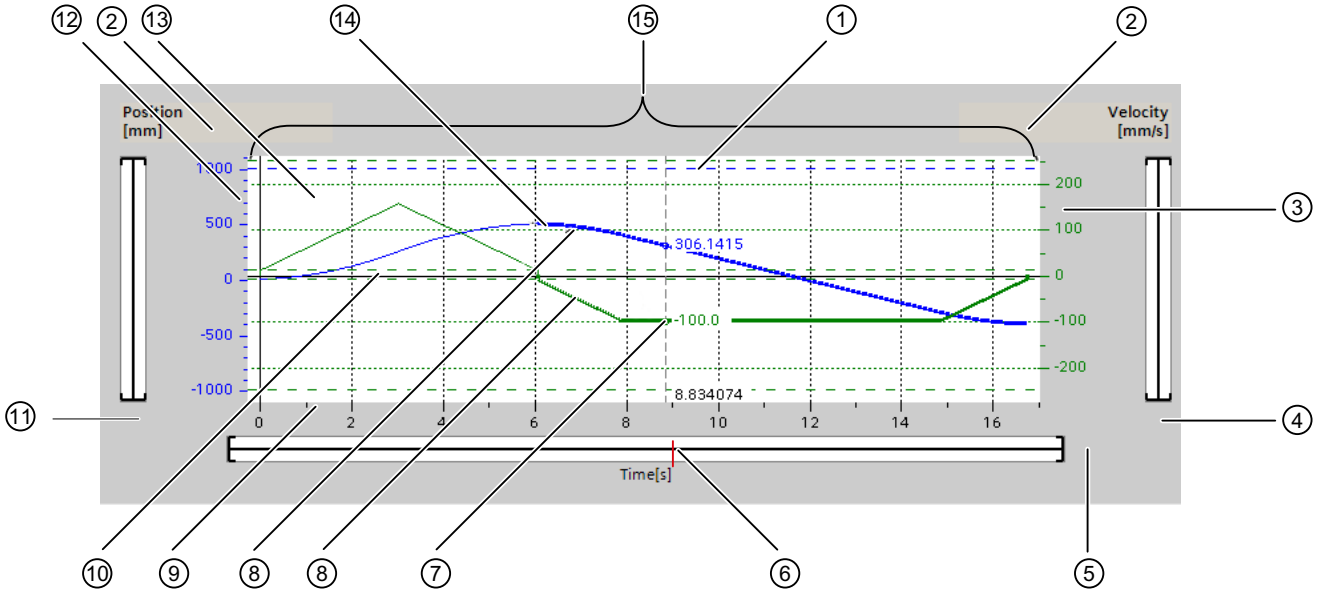

①	Ruler
②	Selecting the grid
③	Velocity axis scale range
④	Scroll bar, velocity axis
⑤	Scroll bar time axis
⑥	Ruler position marking
⑦	Velocity curve
⑧	Curve section of a selected command
⑨	Time axis scale range
⑩	Start/stop velocity
⑪	Scroll bar, position axis
⑫	Position axis scale range
⑬	Software limit switch position
⑭	Position curve
⑮	Trend view

Selecting separator sections

If the command table consists of multiple sections separated by separators, you can select these sections in the trend view by selecting a command in the section.

Selecting commands

Commands can be selected in the trend view and in the command table:

- Click on a point on the velocity or position curve in the trend view. The corresponding command will be highlighted in the command table.
- Select a command in the command table.
  The corresponding section of curve will be highlighted.

Selecting the visible range of the trend view

Follow the steps below to adjust the section of the trend view to be displayed:

Select the scaling in the shortcut menu:

- Scale to curves:
  Scales the axes so the position and velocity curves are visible.
- Scale to curves and limits:
  Scales the axes so the position and velocity curves, the positions of the activated software limit switches and the minimum and maximum velocity limits are visible.

The view selected will be marked in the shortcut menu with a tick.

Selecting the section to be shown within the range:

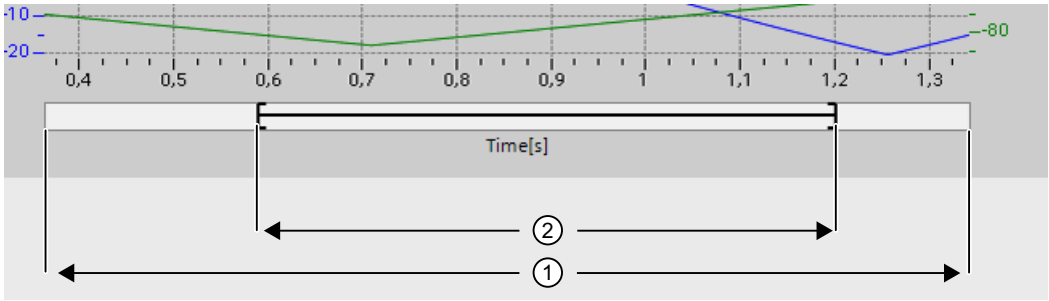

Figure 6-1 CmdTable_Scle01_new

①	Range which the curve values and / or limits are within. (see Selecting in the shortcut menu)
②	Selected range to be shown in the trend window. You set the range with the margin cursor at the right-hand and left-hand margin. You set the position within range ① with the drag cursor. You can also define the position by clicking in range ①. 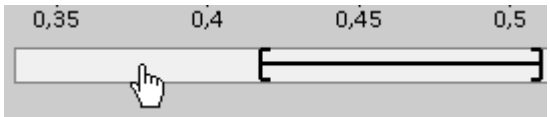

Selecting the section to be shown with the mouse:

Drag a section of the trend view by clicking and dragging with the mouse. The section of curve selected will be enlarged once you release the mouse.

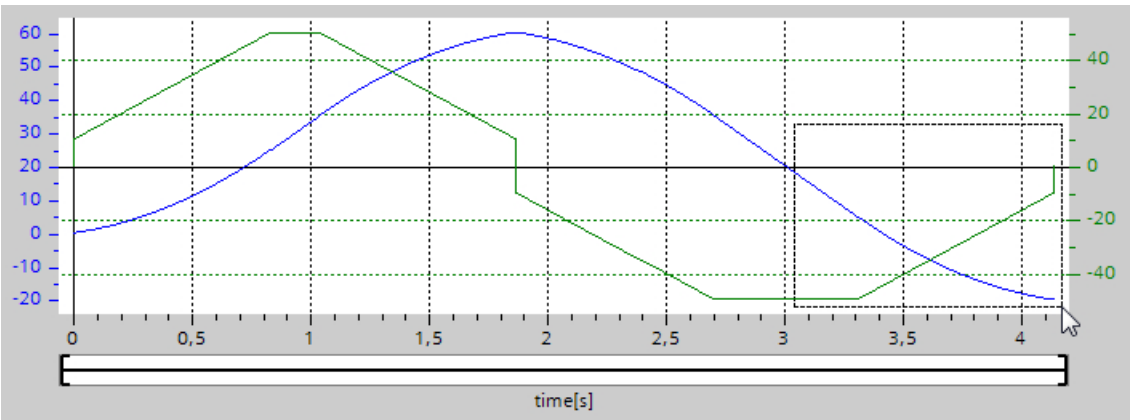

Undoing the last change to the section:

Select the shortcut command "Undo zoom" to undo the last change to the section.

Synchronizing the grid

Click on the axis scales to select whether the grid is to be synchronized with the position axis or velocity axis.

Reading off curve values from the ruler

Activate the ruler using the shortcut menu command "Show ruler".

You can move the ruler to any point on the curves using the ruler cursor.

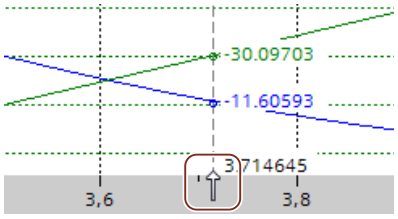

See also

Configuration - General (Page 1404)

Configuration - Command table (Page 1405)

Shortcut menu commands - Command table (Page 1408)

Shortcut menu commands - Curve chart (Page 1413)

Transition from "Complete command" to "Blend motion" (Page 1414)

Changing the command table configuration in the user program (Page 1416)

Shortcut menu commands - Curve chart (S7-1200)

The following shortcut menu commands are available in the curve window:

Zoom 100%

Selects a zoom factor which will show 100% of the curve values and / or limits.

Undo zoom

Undoes the last zoom change.

Scaling on trends

Scales the axes so the position and velocity trends are visible.

Scaling on trends and limits

Scales the axes so the position and velocity trends, the positions of the activated software limit switches and the minimum and maximum velocity limits are visible.

Show velocity limits

Shows the lines of the velocity limits.

Show software limit switches

Shows the lines of the software limit switches.

Show measuring ruler

Fades the measuring ruler in / out

Use the measuring ruler when you want to see the individual values of the trends.

See also

Configuration - General (Page 1404)

Configuration - Command table (Page 1405)

Shortcut menu commands - Command table (Page 1408)

Working with the trend diagram (Page 1409)

Transition from "Complete command" to "Blend motion" (Page 1414)

Changing the command table configuration in the user program (Page 1416)

Transition from "Complete command" to "Blend motion" (S7-1200)

The charts below show the transition between movements in various different transition modes in the "Next step" column:

Motion transition with preceding positioning jobs

The following diagrams show a command sequence with two motion tasks. The first command is for positioning (green). The second command is for velocity (red) or positioning (blue):

Complete job	Blend motion
Transition from lower to higher velocity 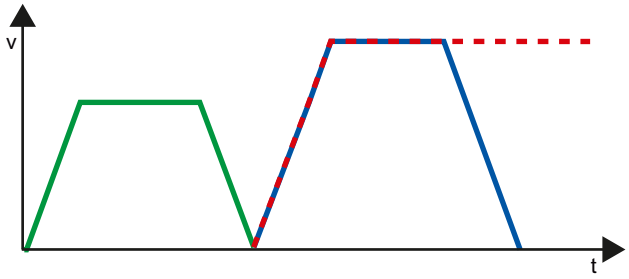 A job with high velocity is appended to a previous positioning job. The positioning job terminates at its target position at velocity "0". The second job starts from standstill.	**Transition from lower to higher velocity** 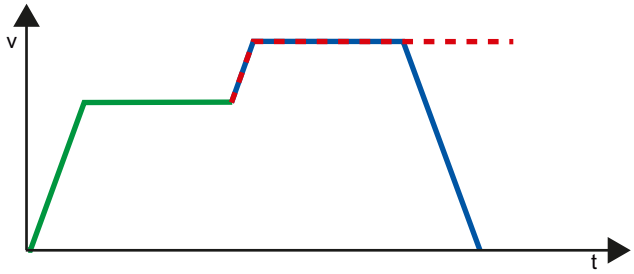 A job with high velocity is overlapped with a previous positioning job. The first positioning job terminates without standstill at its target position. The second job starts with the new velocity.
Transition from higher to lower velocity 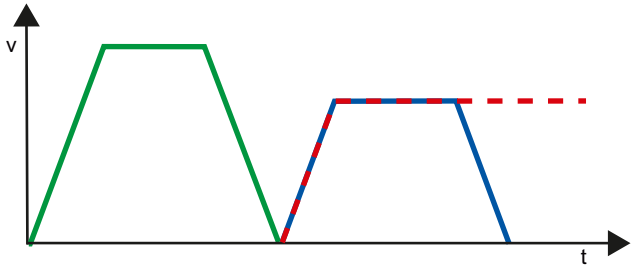 A job with low velocity is appended to a previous positioning job. The positioning job terminates at its target position at velocity "0". The second job starts from standstill.	**Transition from higher to lower velocity** 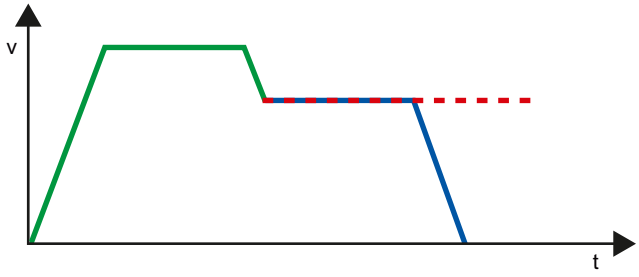 A job with low velocity is overlapped with a previous positioning job. The first positioning job terminates without standstill at its target position. The first job starts with the new velocity.

—	1. Job "Positioning Relative" or "Positioning Absolute"
- -	2. Job "Velocity set point"
—	2. Job "Positioning Relative" or "Positioning Absolute"

Motion transition with preceding velocity jobs

The following diagrams show a command sequence with two motion tasks. The first command is for velocity (violet). The second command is for velocity (red) or positioning (blue):

Transition from lower to higher velocity	Transition from higher to lower velocity
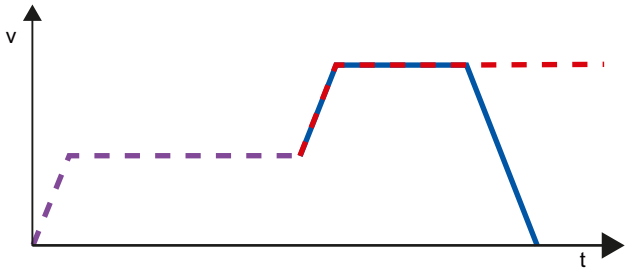	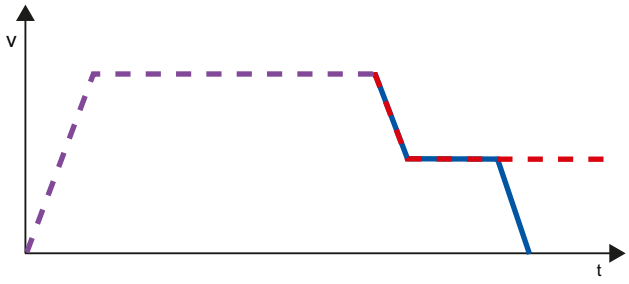
A job with a high velocity is appended to a previous velocity job. The first velocity job ends after the defined runtime. The second job starts with the new velocity.	A command with lower velocity is blended with a previous velocity command. The second job starts with the new velocity.

	1. Job "Velocity set point"
	2. Job "Velocity set point"
	2. Job "Positioning Relative" or "Positioning Absolute"

See also

Configuration - General (Page 1404)

Configuration - Command table (Page 1405)

Shortcut menu commands - Command table (Page 1408)

Working with the trend diagram (Page 1409)

Shortcut menu commands - Curve chart (Page 1412)

Changing the command table configuration in the user program (Page 1416)

Changing the command table configuration in the user program (S7-1200)

You can change the following configuration parameters during runtime of the user program in the CPU:

Commands and corresponding values

You can also change the parameters of the command table during the runtime of the user program. Use the following technology object tags for this purpose:

- <Table name>.Command[1..32].Type
  for changing the command type
- <Table name>.Command[1..32].Position
  for changing the position/travel distance

- <Table name>.Command[1..32].Velocity
  for changing the velocity
- <Table name>.Command[1..32].Duration
  for changing the duration
- <Table name>.Command[1..32].NextStep
  for changing the parameter "Next step"
- <Table name>.Command[1..32].StepCode
  for changing the step code

Refer to the description of the technology object tags (Page 1621) in the appendix for information on when changes to the configuration parameters take effect.

See also

Compatibility list of variables V1...3 <-> V4...5 (Page 1331)

Configuration - General (Page 1404)

Configuration - Command table (Page 1405)

Shortcut menu commands - Command table (Page 1408)

Working with the trend diagram (Page 1409)

Shortcut menu commands - Curve chart (Page 1412)

Transition from "Complete command" to "Blend motion" (Page 1413)

Extended parameters (S7-1200)

Configuration - Extended parameters (S7-1200)

Configure the basic properties of the chart view of the "Command table" technology object in the "Extended parameters" configuration window.

Use axis parameters of

From the drop-down list, select which axis parameters are to be used for selecting the graphic view and checking the movement sequence. Select "Default axis" in the drop-down list if you have yet to add an axis to the "Technology object" folder or wish to use values which have not been configured in any of the available axes.

The axis parameters of the axis selected at the "Axis" parameter will be used to process the command table in the user program.

Unit of measurement position

If you select a default axis in the "Use axis parameter from" drop-down list, you can set the measurement unit.

If you select a configured axis in the drop-down list, the measurement unit of the configured axis is displayed.

Copy axis parameters

Select the direction of copy and the axis for copying the axis parameters. You can copy the axis parameters of the default axis to the selected axis or accept the axis parameters of the selected axis for the default axis. Use the "Apply configuration" button to copy the axis parameters according to your configuration.

Configuration - Dynamics (S7-1200)

Configure the acceleration and deceleration and the jerk limit for the default axis in the "Dynamics" configuration window.

If you select a configured axis under "Configuration > Extended parameters > Extended parameters" in the drop-down list "Use axis parameters from", the values of the configured axis is displayed.

If you select the entry "Default axis" under "Configuration > Extended parameters > Extended parameters" in the drop-down list "Use axis parameters from", you can edit the fields described below.

Acceleration / deceleration

Set the desired acceleration of the default axis in the "Acceleration" field. The desired deceleration can be set in the "Deceleration" field.

Motion jobs configured in the command table will be calculated with the selected acceleration / deceleration.

Limit values:

- 1.0E-12 ≤ acceleration ≤ 1.0E12
- 1.0E-12 ≤ deceleration ≤ 1.0E12

Enable jerk limit

Enable the jerk limit with this check box.

Jerk

Set the desired jerk for ramping up and ramping down in the "Jerk" field.

Motion jobs configured in the command table will be calculated with the selected jerk.

Limit values:

- 1.0E-12 ≤ jerk ≤ 1.0E12

Configuration - Limit values (S7-1200)

Configure the maximum velocity, the start/stop velocity and the software limit switches of the default axis in the "Limits" configuration window.

If you select a configured axis under "Configuration > Extended parameters > Extended parameters" in the drop-down list "Use axis parameters from", the values of the configured axis is displayed.

If you select the entry "Default axis" under "Configuration > Extended parameters > Extended parameters" in the drop-down list "Use axis parameters from", you can edit the fields described below.

Maximum velocity / Start/stop velocity

Define the maximum permissible velocity and the start/stop velocity of the default axis in these boxes. The start/stop velocity is the minimum permissible velocity of the default axis.

Velocity	Limit value
Start/stop velocity	0.0
	1.0E-12 to 1.0E12
Maximum velocity	0.0
	1.0E-12 to 1.0E12

The value of the maximum velocity must be greater or equal to the value of the start/stop velocity.

Enable SW limit switch

Activate the function of the low and high software limit switch with this check box. Movements in response to software limit switches being reached are not shown in the trend view.

Software high and low limit switch

Enter the position value of the low and high software limit switch in these boxes.

Software limit switch	Limit value
Software low limit switch	-1.0E12 to -1.0E-12
	0.0
	1.0E-12 to 1.0E12
Software high limit switch	-1.0E12 to -1.0E-12
	0.0
	1.0E-12 to 1.0E12

The value of the software high limit switch must be greater than or equal to the value of the software low limit switch.

6.1.1.7 Download to CPU (S7-1200)

The data of the Motion Control technology objects are saved in the data blocks. The conditions for downloading of "blocks" therefore apply when loading a new or modified technology object.

CAUTION

Possible malfunctions of the axis when loading without hardware configuration

The hardware configuration is modified when the following modifications are made to the axis or encoder configuration:

- Modification of the pulse generator (PTO)
- Modification of the HW limit switch address
- Modification of the homing switch address
- Modification of the address of the PROFIdrive telegram
- Modification of the address of the analog output
- Modification of address of enable output or ready input

If the modified configuration of the axis or encoder is loaded with the shortcut menu commands "Software" or "Software (all blocks)" without downloading the hardware configuration, the axis may malfunction as a result.

Ensure that the current hardware configuration is downloaded to the CPU under the listed conditions.

Download in CPU S7-1200 RUN mode (from firmware version V2.2)

For CPU S7-1200 from firmware version V2.2, when loading in CPU RUN mode it is checked whether it is possible to load without stopping the CPU.

The following conditions apply when loading data blocks in RUN mode:

	Download to load memory	Download to work memory
Data block modified values	Yes	No
Data block modified structure	Yes (as of firmware V4)	Yes (as of firmware V4) • When downloading with reinitialization • For variables in system reserve for downloading without reinitialization
	No (firmware V2.2...3)	No (firmware V2.2...3)
New data block	Yes	Yes
Data block deleted	Yes	Yes

Also note the following when deleting data blocks and downloading data blocks with reinitialization:

- The axis must be disabled when downloading a positioning axis technology object.
- When downloading a command table technology object, no MC_CommandTable command with this command table must be active (parameter "Busy" = FALSE).
- When downloading an MC_Power instance data block, no MC_Power instruction must be active (parameter "Busy" = FALSE).

From technology version V3.0, Motion Control technology objects (data blocks) can also be downloaded in CPU RUN mode.

Technology objects lower than V3.0 cannot be downloaded in CPU RUN mode.

Select one of the actions described below to download the modified version of a Motion Control technology object (from version V3.0) to the work memory:

- **Technology object positioning axis and command table**
  Change the CPU operating mode from STOP to RUN.
- **Technology object positioning axis**
  Disable the axis and execute a "Restart" using the Motion Control instruction "MC_Reset".
- **Technology object command table**
  Ensure that the command table is not being used. Download the data block of the command table to the work memory using the extended instruction "READ_DBL".

See also

Guidelines on use of motion control (Page 1326)

6.1.1.8 Commissioning (S7-1200)

Axis control panel (S7-1200)

Use the axis control panel to move the axis in manual mode, to optimize the axis settings, and to test your system.

The axis control panel can only be used if an online connection to the CPU is established. It is recommended to disable any other online communication when the axis control panel and the optimization is in use in order to keep the response times as short as possible.

The axis control panel is divided into the following areas:

- Master control
- Axis
- Command
- Current values
- Axis status

Note

Response times of the axis control panel

The response time during axis control panel operation depends on the communication load of the CPU. Close all other online windows of the TIA Portal to minimize the response time.

You can adjust the timeout in the start dialog.

Master control

In this area, you can take over master control of the technology object, or return it to your user program:

- **"Activate" button**
  With the "Activate" button, you set up an online connection to the CPU and take over master control for the selected technology object. Note the following when taking over master control:
  - To take over master control, the technology object must be disabled in the user program.
  - Until master control is returned, the user program has no influence on the functions of the technology object. Motion Control jobs from the user program to the technology object are rejected with error.

  CAUTION

  Additional axes in automatic mode

  The master control is only applied for the selected technology object. If additional axes are in automatic mode, dangerous situations may arise as a result.

  In this happens, disable all other axes.

- **"Deactivate" button**
  With the "Deactivate" button, you return master control to your user program.

Axis

In this area, enable or disable the technology object for operation with the axis control panel/ optimization:

- **"Enable" button**
  With the "Enable" button, you enable the selected technology object.
- **"Disable" button**
  With the "Disable" button, you disable the selected technology object.

Command

Operation in the "command" area is only possible if the axis is enabled. You can select one of the following command inputs:

- **Jog**
  This command is equivalent to Motion Control command "MC_MoveJog" in the user program.
- **Positioning**
  This command is equivalent to the Motion Control commands "MC_MoveAbsolute" and "MC_MoveRelative" in the user program. The axis must be homed for absolute positioning.
- **Homing**
  This command is equivalent to Motion Control command "MC_Home" in the user program.
  - The "Set reference point" button corresponds to Mode = 0 (direct homing absolute)
  - The "Homing" button corresponds to Mode = 3 (active homing)

  For active homing, the homing switch must be configured in the axis configuration.
  The values for approach velocity, homing velocity, and reference position offset are taken from the axis configuration unchanged.

Depending on the selection, the relevant boxes for entry of setpoints and the buttons for starting the command are displayed.

Select the "Enable jerk limitation" check box to activate the jerk limitation. By default, the jerk is applied with 10% of the configured value. This value can be changed as required.

Current values

The following actual values of the axis are displayed in this area:

- Position
- Velocity

Axis status

The current axis status and drive status are shown in the "Axis status" area.

Status message	Description
Enabled	The axis is enabled and ready to be controlled via Motion Control commands.
Homed	The axis is homed and is capable of executing absolute positioning commands of Motion Control instruction "MC_MoveAbsolute".
Ready	The drive is ready for operation.
Axis error	An error has occurred in the positioning axis technology object. The "Error message" box displays detailed information about the cause of the error.
Encoder values valid	The encoder values are valid.
Simulation active	The axis is simulated in the CPU. Setpoints are not output to the drive.
Drive error	The drive has reported an error due to loss of its "Drive ready" signal.
Restart required	A modified configuration of the axis was downloaded to the load memory in CPU RUN mode. To download the modified configuration to the work memory, you need to restart the axis. Use the Motion Control instruction "MC_Reset" to do this.

The "Info message" box displays advanced information about the status of the axis.

The "Error message" box shows the current error.

Click "Acknowledge" to acknowledge all cleared errors.

Note

Initial values for velocity, acceleration/deceleration and jerk

For safety reasons, the "Velocity", "Acceleration/Deceleration" and "Jerk" parameters are initialized with values equivalent to only 10% of the configured values when the axis control panel is activated. The "Jerk" parameter is only used for technology object "Axis" V2.0 and higher.

The values in the configuration view displayed when you select "Extended parameters > Dynamics > General" are used for initialization.

The "Velocity" parameter on the axis control panel is derived from the "Maximum velocity" and the "Acceleration/Deceleration" parameters from "Acceleration" in the configuration.

The "Velocity", "Acceleration/deceleration" and "Jerk" parameters can be changed in the axis control panel. This does not affect the values in the configuration.

See also

Guidelines on use of motion control (Page 1326)

Tuning (S7-1200)

The movement of axes with drive connection via PROFIdrive/analog output is position-controlled.

The "Tuning" function supports you in determining the optimal gain (Kv factor) for the control loop (Page 1379) of the axis. The axis velocity profile is recorded by means of the Trace function for this purpose for the duration of a configurable positioning movement. Then you can evaluate the recording, and adapt the gain accordingly. It is recommended to disable any other online communication when the axis control panel and the optimization is in use in order to keep the response times as short as possible.

The "Tuning" function for the positioning axis technology object can be found in the project tree under "Technology object > Commissioning".

The "Tuning" dialog is divided into the following areas:

- Master control
- Axis
- Axis status
- Optimize gain setting
- Run measurement
- Trace

Note

No transfer of the parameters

The configured parameter values are discarded after master control is returned. Transfer the values as needed into your configuration.

Start of optimization

The trace is started at the same time the optimization starts. A timeout can be adapted in the start dialog for this.

Master control

In this area, you can take over master control of the technology object, or return it to your user program:

- **"Activate" button**
  With the "Activate" button, you set up an online connection to the CPU and take over master control for the selected technology object. Note the following when taking over master control:
  - To take over master control, the technology object must be disabled in the user program.
  - Until master control is returned, the user program has no influence on the functions of the technology object. Motion Control jobs from the user program to the technology object are rejected with error.

  CAUTION

  Additional axes in automatic mode

  The master control is only applied for the selected technology object. If additional axes are in automatic mode, dangerous situations may arise as a result.

  In this happens, disable all other axes.

- **"Deactivate" button**
  With the "Deactivate" button, you return master control to your user program.

Axis

In this area, enable or disable the technology object for operation with the axis control panel/ optimization:

- **"Enable" button**
  With the "Enable" button, you enable the selected technology object.
- **"Disable" button**
  With the "Disable" button, you disable the selected technology object.

Axis status

The current axis status and drive status are shown in the "Axis status" area.

Status message	Description
Enabled	The axis is enabled and ready to be controlled via Motion Control commands.
Homed	The axis is homed and is capable of executing absolute positioning commands of Motion Control instruction "MC_MoveAbsolute".
Ready	The drive is ready for operation.
Axis error	An error has occurred in the positioning axis technology object. The "Error message" box displays detailed information about the cause of the error.
Encoder values valid	The encoder values are valid.
Simulation active	The axis is simulated in the CPU. Setpoints are not output to the drive.
Drive error	The drive has reported an error due to loss of its "Drive ready" signal.
Restart required	A modified configuration of the axis was downloaded to the load memory in CPU RUN mode. To download the modified configuration to the work memory, you need to restart the axis. Use the Motion Control instruction MC_Reset to do this.

The "Info message" box displays advanced information about the status of the axis.

The "Error message" box shows the current error.

Click "Acknowledge" to acknowledge all cleared errors.

Optimize gain setting

You make the settings for optimization of the gain in this area:

- Precontrol
  In this field, configure the current velocity precontrol of the position controller as a percentage.
- Distance
  In this field, configure the load distance for one test step.
- "Customize dynamics" check box
  Select this option to adapt the acceleration and the maximum acceleration for the optimization.
- Velocity
  In this field, you configure the maximum velocity for a test step.
- Acceleration
  In this field, you configure the acceleration for a test step.
- Measurement duration
  The measurement duration is recalculated and entered depending on the selected acceleration, velocity and distance.
  You can adapt the value of the measurement duration afterwards.
- Gain
  In this field, you configure the actual gain of the position controller (Kv).
  The gain takes effect when it is entered. If the gain of the position controller (Kv) is too large, this can lead to an error on the drive.

Run measurement

Perform the test steps in this area:

- **"Forward" button**
  With the "Forward" button, you start a test step for optimization in the positive direction.
- **"Backward" button**
  With the "Backward" button, you start a test step for optimization in the negative direction.
- **"Stop" button**
  You can use the "Stop" button to end the current movement for optimization and end trace recording.

Trace

With each test step, a Trace recording of the required parameters is automatically started and displayed after completion of the test step. After master control has been returned, the Trace recording is deleted.

You will find a full description of the Trace function in the section on using the trace and logic analyzer function in the TIA Portal help.

6.1.1.9 Programming (S7-1200)

Overview of the Motion Control statements (S7-1200)

You control the axis with the user program using Motion Control instructions. The instructions start Motion Control commands that execute the desired functions.

The status of the Motion Control commands and any errors that occur during their execution can be obtained from the output parameters of the Motion Control instructions. The following Motion Control instructions are available:

- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT

See also

Creating a user program (Page 1428)

Programming notes (Page 1431)

Behavior of the Motion Control commands after POWER OFF and restart (Page 1433)

Monitoring active commands (Page 1433)

Error displays of the Motion Control statements (Page 1444)

Creating a user program (S7-1200)

In the section below you learn how to create a user program with the basic configuration for controlling your axis. All available axis functions are controlled using the Motion Control instructions to be inserted.

Requirement

- The technology object has been created and configured without errors.

Before creating and testing the user program, it is advisable to test the function of the axis and the corresponding parts of the system with the axis command table.

Procedure

Proceed as follows to create the user program in accordance with the principles described below:

1. In the project tree, double-click your code block (the code block must be called in the cyclic program).
   The code block is opened in the programming editor and all available instructions are displayed.
2. Open the "Technology" category and the "Motion Control" folder.
3. Use a drag-and-drop operation to move the "MC_Power" instruction to the desired network of the code block.
   The dialog box for defining the instance DB opens.
4. In the next dialog box, select from the following alternatives:
   Single instance
   Click "Single instance" and select whether you want to define the name and number of the instance DB automatically or manually.
   Multi-instance
   Click "Multi-instance" and select whether you want to define the name of the multi-instance automatically or manually.

5. Click "OK".
   The Motion Control instruction "MC_Power" is inserted into the network.

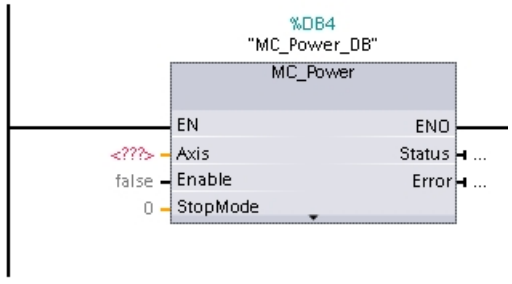

Parameters marked with "<???>" must be initialized; default values are assigned to all other parameters.
Parameters displayed in black are required for use of the Motion Control instruction.

6. Select the technology object in the project tree and drag-and-drop it on <???>.

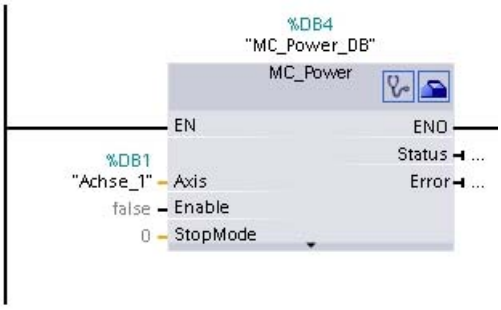

After the selection of the technology object data block, the following buttons are available:

Click the stethoscope icon if you want to open the diagnostics dialog for the technology object.

Click the toolbox icon if you want to open the configuration view of the technology object.

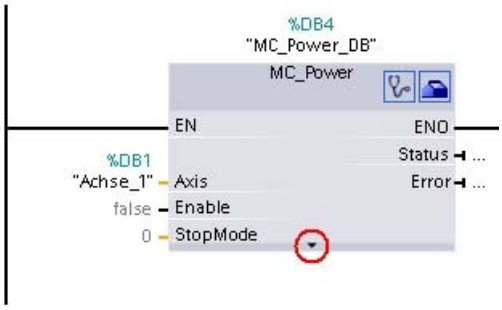

Click the arrow down icon to view additional parameters of the Motion Control instruction.

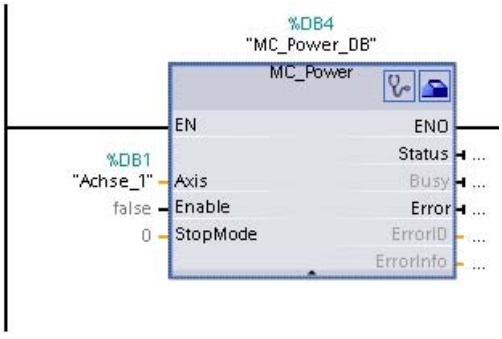

The grayed-out parameters now visible can be used optionally.

7. Add your choice of Motion Control instructions in accordance with steps 3 to 6.

Result

You have created the basic configuration for axis control in the user program.

Initialize the input parameters of Motion Control instructions in other parts of the user program to initiate the desired jobs for the "Axis" technology object.

Evaluate the output parameters of the Motion Control instructions and the tags of the data block to track the initiated jobs and the status of the axis.

Refer to the detailed description for details on the parameters of Motion Control instructions.

See also

Overview of the Motion Control statements (Page 1426)

Programming notes (Page 1431)

Behavior of the Motion Control commands after POWER OFF and restart (Page 1433)

Monitoring active commands (Page 1433)

Error displays of the Motion Control statements (Page 1444)

Programming notes (S7-1200)

When creating your user program, note the following information:

- **Cyclic call of utilized motion control instructions**
  The current status of command execution is available via the output parameters of the motion control instruction. The status is updated with every call of the motion control instruction. Therefore, make sure that the utilized motion control instructions are called cyclically.
- **Transfer of parameter values of a motion control instruction**
  The parameter values pending for the input parameters are transferred with a positive edge at input parameter "Execute" when the block is called.
  The motion control command is started with these parameter values. Parameter values that are subsequently changed for the motion control instruction are not transferred until the next start of the motion control command.
  Exceptions to this are input parameters "StopMode" of motion control instruction "MC_Power" and "Velocity" of motion control instruction "MC_MoveJog". A change in the input parameter is also applied with "Enable" = TRUE or "JogForward" and "JogBackward".

- **Programming under consideration of the status information**
  In a stepwise execution of motion control jobs, make sure to wait for the active command to finish before starting a new command. Use the status messages of the motion control instruction and the "StatusBits" tag of the technology object to check for completion of the active command.
  In the examples below, observe the indicated sequence. Failure to observe the sequence will display an axis or command error.
  - **Axis enable with motion control instruction "MC_Power"**
    You must enable the axis before it can take on motion jobs. Use an AND operation of tag <Axis name>.StatusBits.Enable = TRUE with output parameter Status = TRUE of motion control instruction "MC_Power" to verify that the axis is enabled.
  - **Acknowledge error with motion control instruction "MC_Reset"**
    Prior to starting a motion control command, errors requiring acknowledgement must be acknowledged with "MC_Reset". Eliminate the cause of the error and acknowledge the error with motion control instruction "MC_Reset". Verify that the error has been successfully acknowledged before initiating a new command. For this purpose, use an AND operation of tag <Axis name>.StatusBits.Error = FALSE with output parameter Done = TRUE of motion control instruction "MC_Reset".
  - **Home axis with motion control instruction "MC_Home"**
    Before you can start an MC_MoveAbsolute command, the axis must be homed. Use an AND operation of tag <Axis name>.StatusBits.HomingDone = TRUE with output parameter Done = TRUE of motion control instruction "MC_Home" to verify that the axis has been homed.
- **Override of motion control command processing**
  Motion control jobs for moving an axis can also be executed as overriding jobs.
  If a new motion control command is started for an axis while another motion control command is active, the active command is overridden by the new command before the existing command is completely executed. The overridden command signals this using CommandAborted = TRUE in the motion control instruction. It is possible to override an active MC_MoveRelative command with a MC_MoveAbsolute command.
- **Avoiding multiple use of the same instance**
  All relevant information of a motion control command is stored in its instance.
  Do not start a new command using this instance, if you want to track the status of the current command. Use different instances if you want to track the commands separately. If the same instance is used for multiple motion control commands, the status and error information of the individual commands will overwrite each other.
  In the user program, each axis must be called with a separate call of the Motion Control instruction "MC_Power" with a separate instance data block.
- **Call of motion control instructions in different priority classes (run levels)**
  Motion Control instructions with the same instance may not be called in different priority classes without interlocking. To learn how to call locked motion control instructions, refer to "Tracking commands from higher priority classes (run levels) (Page 1457)".

See also

Overview of the Motion Control statements (Page 1426)

Creating a user program (Page 1427)

Behavior of the Motion Control commands after POWER OFF and restart (Page 1433)

Monitoring active commands (Page 1433)

Error displays of the Motion Control statements (Page 1444)

Tracking jobs from higher priority classes (execution levels) (Page 1457)

Behavior of the Motion Control commands after POWER OFF and restart (S7-1200)

A POWER OFF or CPU-STOP aborts all active motion control jobs. All CPU outputs, including pulse and direction outputs, are reset.

After a subsequent POWER ON or CPU restart (CPU RUN), the technology objects and the motion control jobs will be reinitialized.

All actual data of the technology objects as well as all status and error information of the previously active motion control jobs are reset to their initial values.

Before the axis can be reused, it must be enabled again using the Motion Control instruction "MC_Power". If homing is required, the axis must be homed again with Motion Control instruction "MC_Home". When an absolute encoder is used, homing is retained after POWER OFF.

See also

Overview of the Motion Control statements (Page 1426)

Creating a user program (Page 1427)

Programming notes (Page 1430)

Monitoring active commands (Page 1433)

Error displays of the Motion Control statements (Page 1444)

Monitoring active commands (S7-1200)

Monitoring active commands (S7-1200)

There are three typical groups for tracking active Motion Control commands:

- **Motion control instructions with output parameter "Done"**
- **Motion control instruction "MC_MoveVelocity"**
- **Motion control instruction "MC_MoveJog"**

Motion control instructions with "Done" output parameter (S7-1200)

Motion control instructions with the output parameter "Done" are started via input parameter "Execute" and have a defined conclusion (for example, with Motion Control instruction "MC_Home": Homing was successful). The command is complete and the axis is at a standstill.

The commands of the following Motion Control instructions have a defined conclusion:

- MC_Reset
- MC_Home

- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_CommandTable (technology object as of V2)
- MC_ChangeDynamic (technology object as of V2)
- MC_WriteParam (as of technology object V4)
- MC_ReadParam (as of technology object V4)

The output parameter "Done" indicates the value TRUE, if the command has been successfully completed.

The output parameters "Busy", "CommandAborted", and "Error" signal that the command is still being processed, has been aborted or an error is pending. The Motion Control instruction "MC_Reset" cannot be aborted and thus has no "CommandAborted" output parameter. The Motion Control instruction "MC_ChangeDynamic" is completed immediately and therefore has no "Busy" or "CommandAborted" output parameters.

During execution of the Motion Control command, the output parameter "Busy" indicates the value TRUE. If the command has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE. This change occurs regardless of the signal at input parameter "Execute".

Output parameters "Done", "CommandAborted", and "Error" indicate the value TRUE for at least one cycle. These status messages are latched while input parameter "Execute" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

Complete execution of command

If the Motion Control command has been completely executed by the time of its conclusion, this is indicated by the value TRUE in output parameter "Done". The signal status of the input parameter "Execute" influences the display duration at the output parameter "Done":

"Execute" changes its value to FALSE **during processing** of the command	"Execute" changes its value to FALSE **after completion** of the command
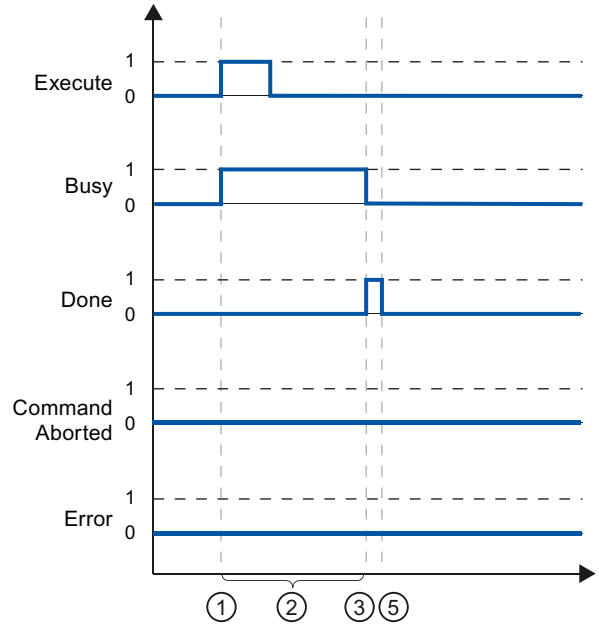	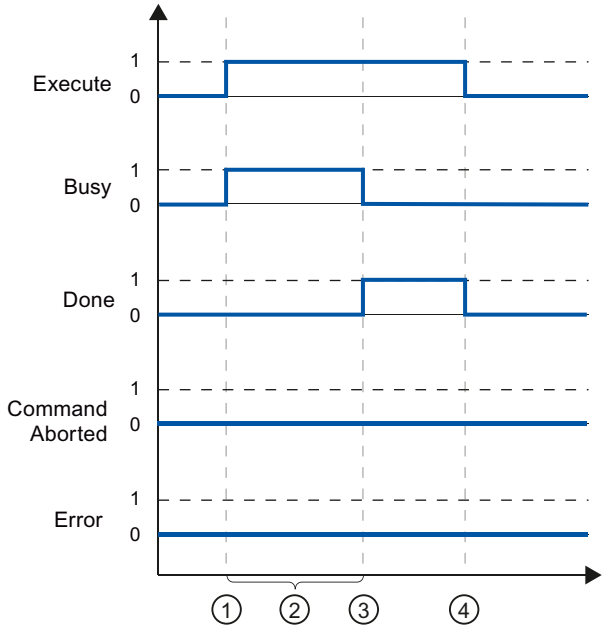

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after completion of the command.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	With conclusion of the command (for example, for Motion Control instruction "MC_Home": Homing was successful), output parameter "Busy" changes to FALSE and "Done" to TRUE.
④	If "Execute" retains the value TRUE until after completion of the command, then "Done" also remains TRUE and changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the command is complete, "Done" indicates the value TRUE for only one execution cycle.

Abort command

If the Motion Control command is aborted during execution, this is indicated by the value TRUE in output parameter "CommandAborted". The signal status of the input parameter "Execute" influences the display duration at the output parameter "CommandAborted":

"Execute" changes its value to FALSE **before** the command is aborted.	"Execute" changes its value to FALSE **after** the command is aborted.
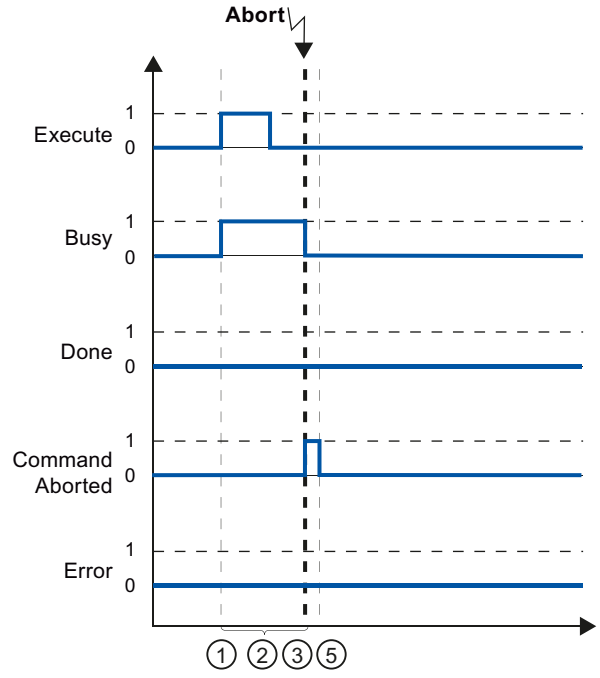	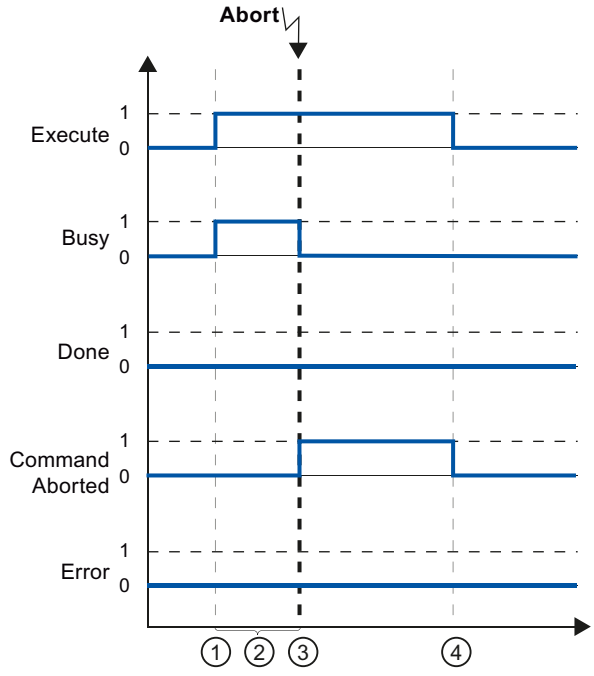

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after completion of the command.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	During command execution, the command is aborted by another Motion Control command. If the command is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	If "Execute" retains the value TRUE until after the command is aborted, then "CommandAborted" also remains TRUE and changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the command is aborted, "CommandAborted" indicates the value TRUE for only one execution cycle.

Error during command execution

If an error occurs during execution of the Motion Control command, this is indicated by the value TRUE in the output parameter "Error". The signal status of the input parameter "Execute" influences the display duration at the output parameter "Error":

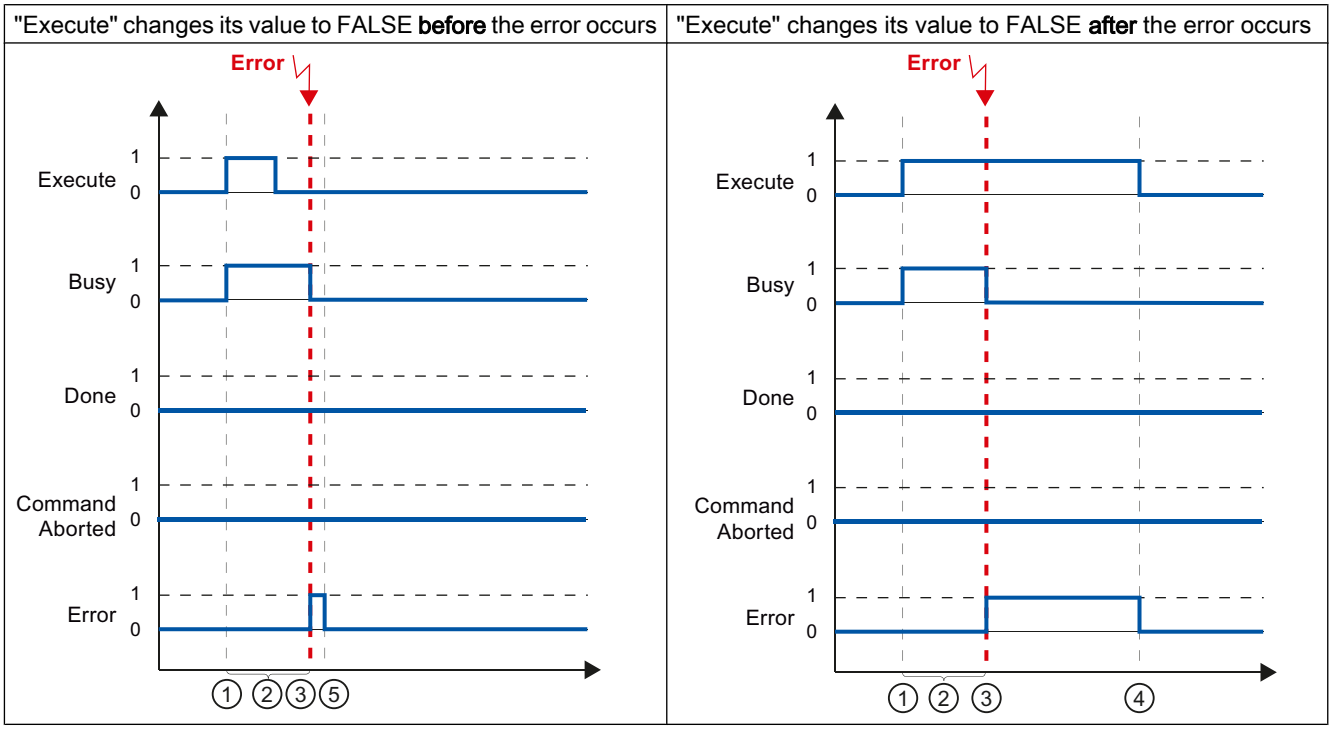

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after completion of the command.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during command execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	If "Execute" retains the value TRUE until after the error occurs, then "Error" also remains TRUE and only changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the error occurs, "Error" indicates the value TRUE for only one execution cycle.

Motion control instruction MC_MoveVelocity (S7-1200)

A "MC_MoveVelocity" command is started with a positive edge at the "Execute" parameter. The command objective is fulfilled when the assigned velocity is reached and the axis travels at constant velocity. When the assigned velocity is reached and maintained, this is indicated in the "InVelocity" parameter with the value TRUE.

The motion of the axis can, for example, be stopped with an "MC_Halt" command.

The output parameters "Busy", "CommandAborted", and "Error" signal that the command is still being processed, has been aborted or an error is pending.

During execution of the Motion Control command, the output parameter "Busy" indicates the value TRUE. If the command has been stopped by another command or by an error, the "Busy" output parameter changes its value to FALSE. This change occurs regardless of the signal at the input parameter "Execute".

The output parameters "CommandAborted" and "Error" show the value TRUE for at least one cycle. These status messages are latched while input parameter "Execute" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

The parameterized velocity is reached

If the Motion Control command has been executed by the time the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity". The parameter "Execute" has no effect on the indication duration in the "InVelocity" parameter.

①	The job is started with a positive edge at the "Execute" parameter. Depending on the programming, "Execute" can be reset to the FALSE value before or after the parameterized velocity has been reached. While the job is active, the parameter "Busy" shows the value TRUE.
②	When the assigned velocity is reached, the "InVelocity" parameter changes to TRUE. The "Busy" and "InVelocity" parameters retain the TRUE value until the "MC_MoveVelocity" command is overridden by another Motion Control command or stopped by an error.

The command is aborted prior to reaching the parameterized velocity

If the Motion Control command is aborted before the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "CommandAborted". The signal status of input parameter "Execute" influences the display duration at the output parameter "CommandAborted".

"Execute" changes its value to FALSE **before** the command is aborted.	"Execute" changes its value to FALSE **after** the command is aborted.
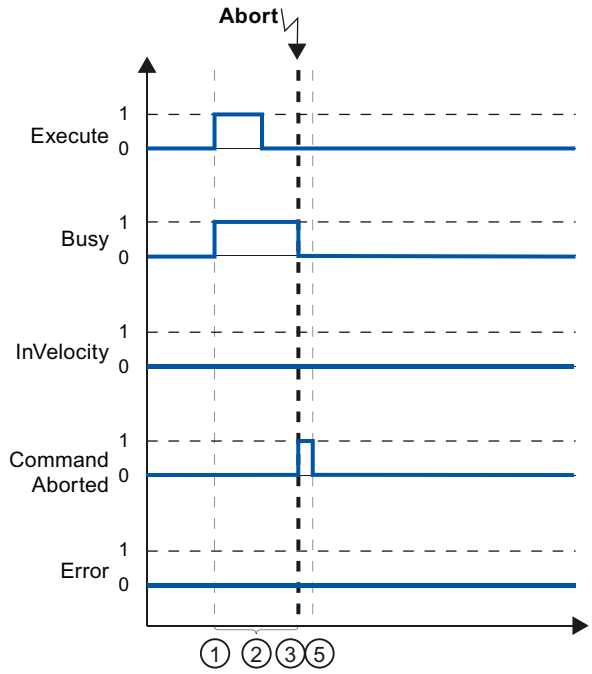	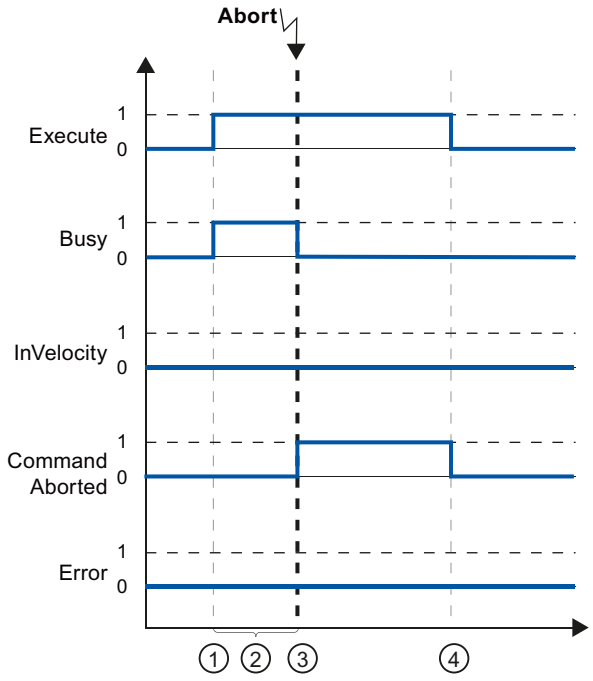

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after the command is aborted.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	During command execution, the command is aborted by another Motion Control command. If the command is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	If "Execute" retains the value TRUE until after the command is aborted, then "CommandAborted" also remains TRUE and changes its status to FALSE together with "Execute".
⑤	If "Execute" has been reset to FALSE before the command is aborted, "CommandAborted" indicates the value TRUE for only one execution cycle.

Note

Under the following conditions, an abort is not indicated in output parameter "CommandAborted":

The parameterized velocity has been reached, input parameter "Execute" has the value FALSE, and a new Motion Control command is initiated.

An error has occurred prior to reaching the parameterized velocity

If an error occurs during execution of the Motion Control command before the parameterized velocity has been reached, this is indicated by the value TRUE in the output parameter "Error". The signal status of the input parameter "Execute" influences the display duration at the output parameter "Error":

"Execute" changes its value to FALSE **before** the error occurs	"Execute" changes its value to FALSE **after** the error occurs
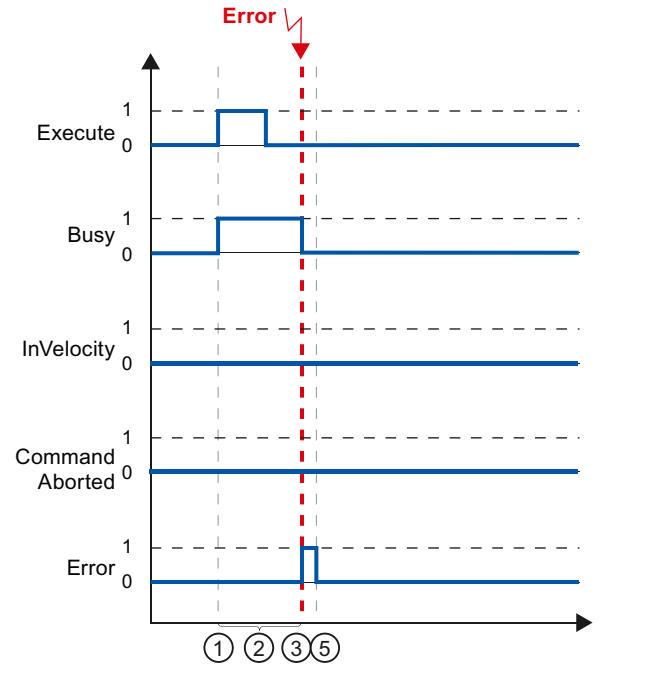	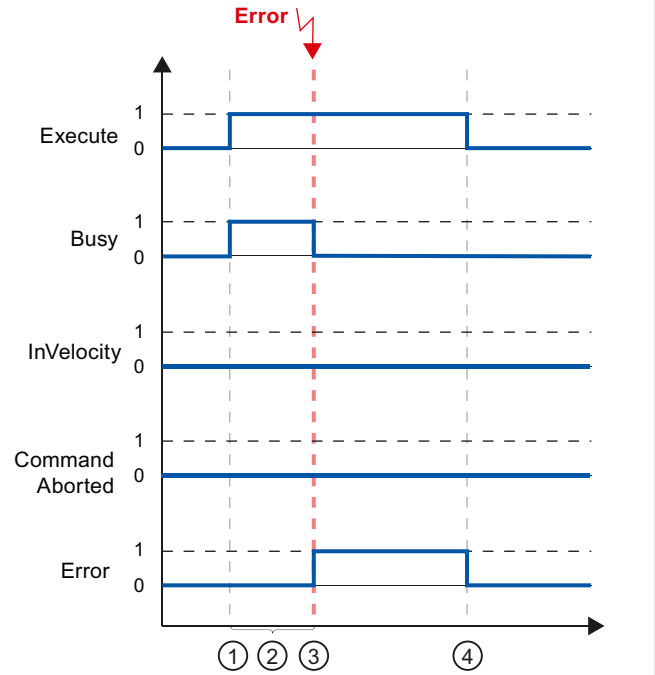

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after the error has occurred.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during command execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	If "Execute" retains the value TRUE until after the error has occurred, then "Error" also remains TRUE and only changes its status to FALSE together with "Execute".
⑤	If "Execute" has been reset to FALSE before the error occurs, "Error" indicates the value TRUE for only one execution cycle.

Note

Under the following conditions, an error is not indicated in output parameter "Error":

The parameterized velocity has been reached, input parameter "Execute" has the value FALSE, and an axis error occurs (software limit switch is approached, for example).

The error of the axis is only indicated in the "MC_Power" Motion Control instruction.

Motion control instruction MC_MoveJog (S7-1200)

The commands of Motion Control instruction "MC_MoveJog" implement a jog operation.

The motion control commands "MC_MoveJog" do not have a defined end. The command objective is fulfilled when the parameterized velocity is reached for the first time and the axis travels at constant velocity. When the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".

The order is complete when input parameter "JogForward" or "JogBackward" has been set to the value FALSE and the axis has come to a standstill.

The output parameters "Busy", "CommandAborted", and "Error" signal that the command is still being processed, has been aborted or an error is pending.

During processing of the motion control command, the output parameter "Busy" indicates the value TRUE. If the command has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE.

The output parameter "InVelocity" indicates the status TRUE, as long as the axis is moving at the parameterized velocity. The output parameters "CommandAborted" and "Error" indicate the status for at least one cycle. These status messages are latched as long as either input parameter "JogForward" or "JogBackward" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

The parameterized velocity is reached and maintained

If the motion control command has been executed by the time the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".

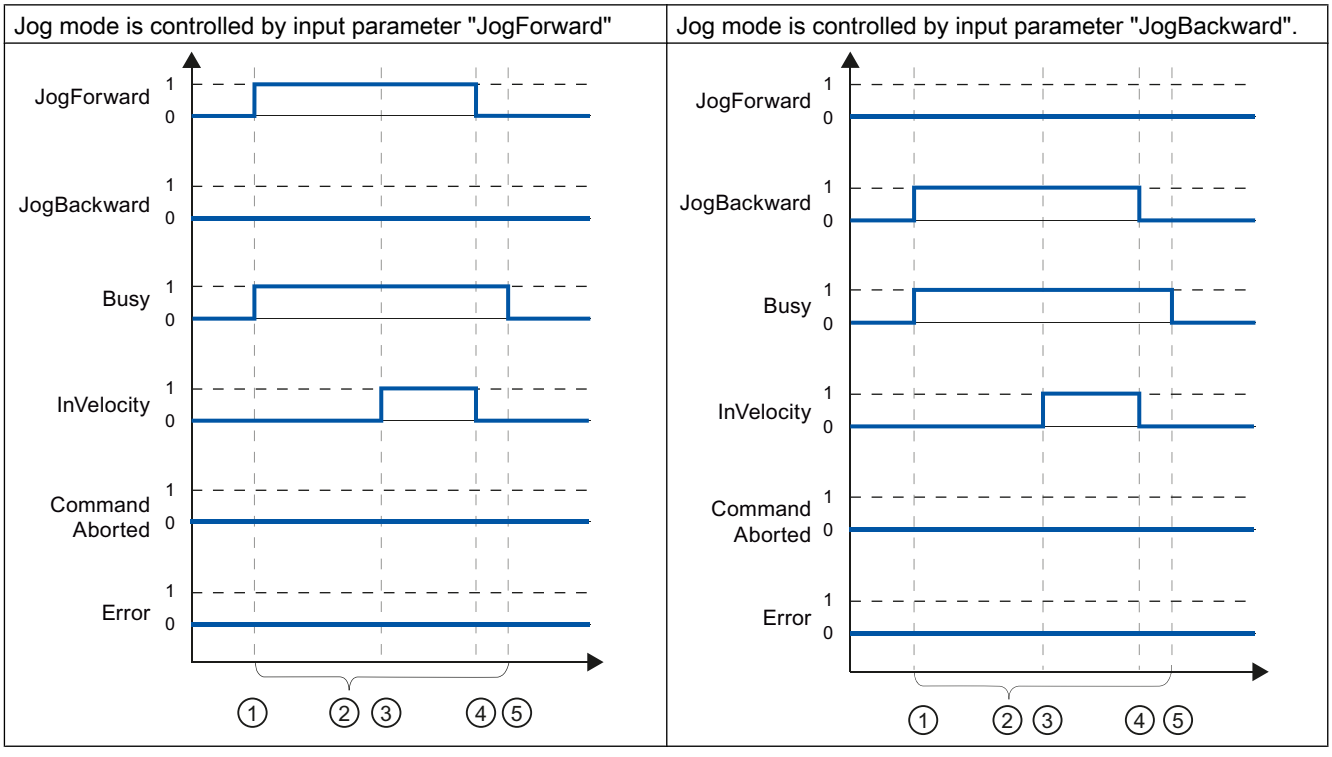

①	The command is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	When the parameterized velocity is reached, the output parameter "InVelocity" changes to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the axis motion ends. The axis starts to decelerate. As a result, the axis no longer moves at constant velocity and the output parameter "InVelocity" changes its status to FALSE.
⑤	If the axis has come to a standstill, the motion control command is complete and the output parameter "Busy" changes its value to FALSE.

The command is aborted during execution

If the motion control command is aborted during execution, this is indicated by the value TRUE in output parameter "CommandAborted". The behavior is independent of whether or not the parameterized velocity has been reached.

Jog mode is controlled by input parameter "JogForward".	Jog mode is controlled by input parameter "JogBackward".
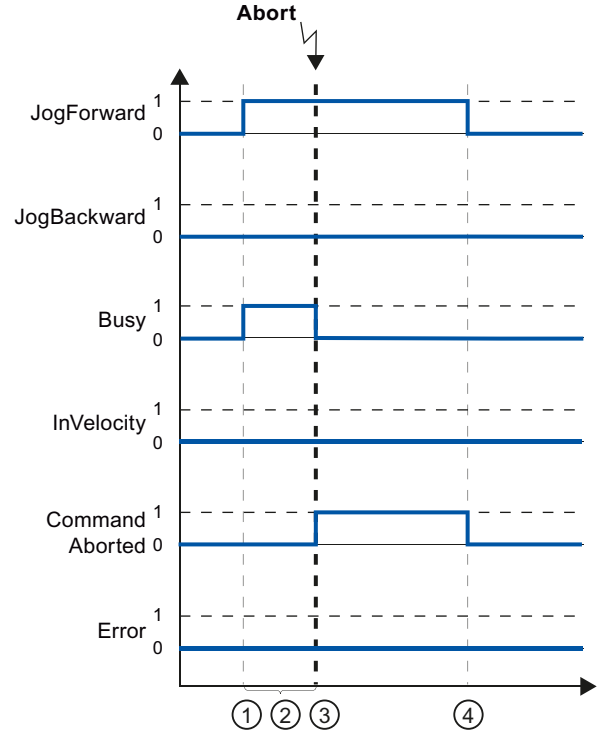	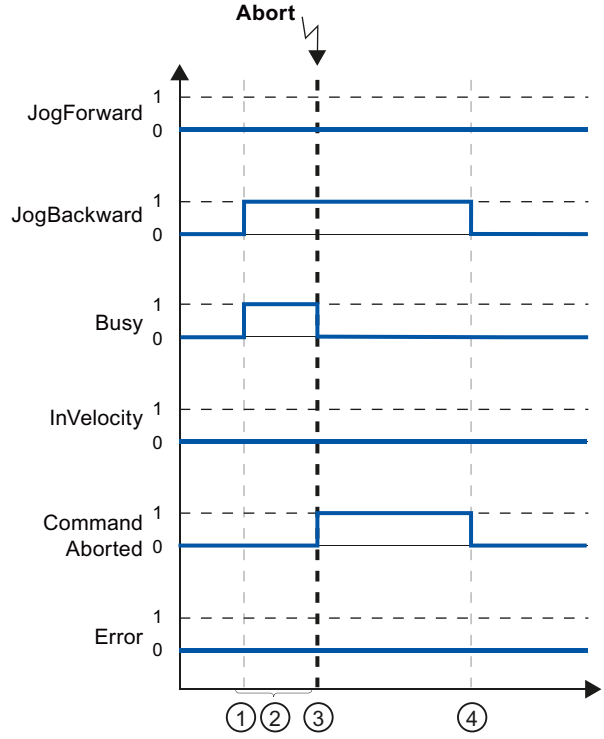

①	The command is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	During command execution, the command is aborted by another motion control command. If the command is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the output parameter "CommandAborted" changes its value to FALSE.

Note

The command abort is indicated in the output parameter "CommandAborted" for only one execution cycle, if all conditions below are met:

The input parameters "JogForward" and "JogBackward" have the value FALSE (but the axis is still decelerating) and a new motion control command is initiated.

An error has occurred during command execution

If an error occurs during execution of the motion control command, this is indicated by the value TRUE in output parameter "Error". The behavior is independent of whether or not the parameterized velocity has been reached.

Jog mode is controlled by input parameter "JogForward".	Jog mode is controlled by input parameter "JogBackward".
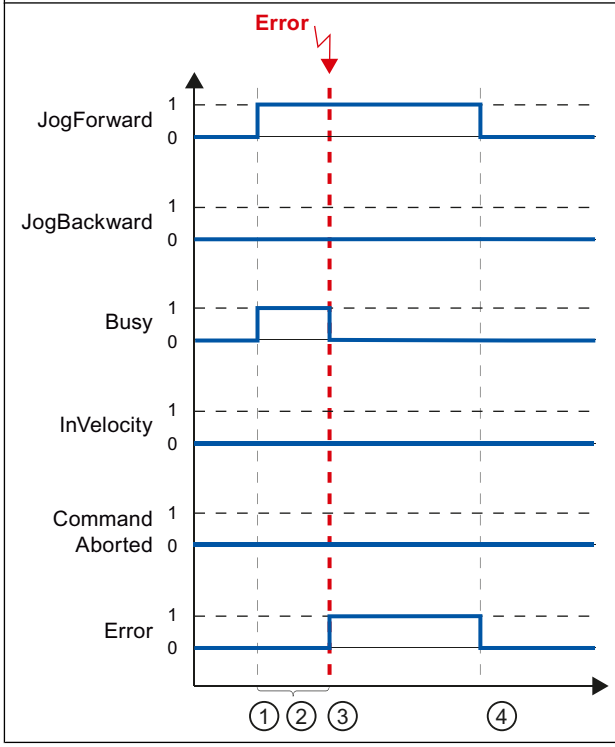	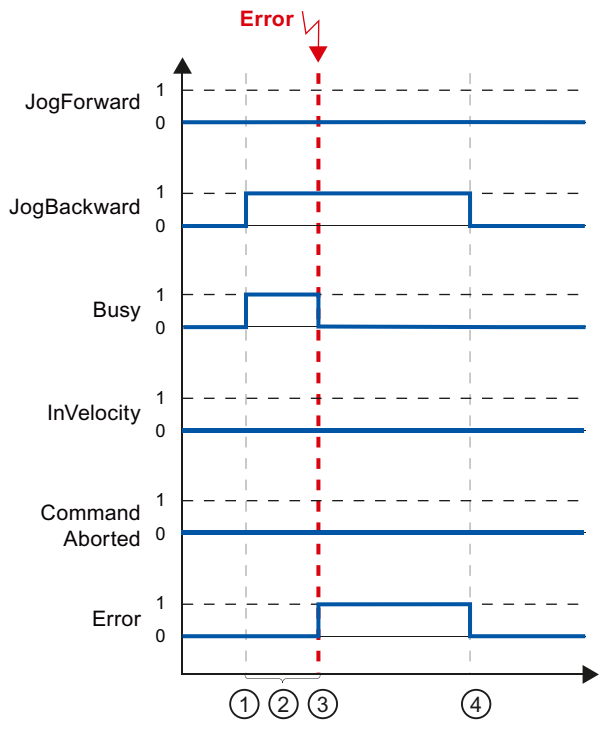

①	The command is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during command execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the output parameter "Error" changes its value to FALSE.

Note

An error occurrence is indicated in the output parameter "Error" for only one execution cycle, if all the conditions below are met:

The input parameters "JogForward" and "JogBackward" have the value FALSE (but the axis is still decelerating) and a new error occurs (software limit switch is approached, for example).

Error displays of the Motion Control statements (S7-1200)

The Motion Control instructions indicate any errors in motion control commands and the technology object at the output parameters "Error", "ErrorID" and "ErroInfo" of the Motion Control instructions.

Error display at output parameters "Error", "ErrorID" and "ErrorInfo"

If the output parameter "Error" indicates the value TRUE, the complete command, or portions thereof, could not be executed. The cause of the error is indicated by the value in output parameter "ErrorID". Detailed information about the cause of the error is returned by the value in output parameter ErrorInfo. We distinguish between the following error classes for error indication:

- **Operating error with axis stop (for example, "HW limit switch was approached")**
  Operating errors with axis stop are errors that occur during runtime of the user program. If the axis is in motion, it is stopped with the configured deceleration or emergency stop deceleration, depending on the error. The errors are indicated in the error-triggering Motion Control instruction and in the Motion Control instruction "MC_Power".
- **Operating error without axis stop (for example, "Axis is not homed")**
  Operating errors without axis stop are errors that occur during runtime of the user program. If the axis is in motion, the motion is continued. The errors are only indicated in the Motion Control instruction which triggers the error.
- **Configuration error in Motion Control instruction (for example "Incorrect value in parameter "Velocity")**
  Parameterization errors occur when incorrect information is specified in the input parameters of Motion Control instructions. If the axis is in motion, the motion is continued. The errors are only indicated in the Motion Control instruction which triggers the error.
- **Configuration error on technology object "Axis" (for example, "Value for "Acceleration" is invalid")**
  A configuration error exists if one or more parameters are incorrectly configured in the axis configuration or if editable configuration data have been modified incorrectly during runtime of the program. An axis in motion is stopped with the configured emergency stop deceleration. The error is indicated in the error-triggering Motion Control instruction and in Motion Control instruction "MC_Power".

- **Configuration error on technology object "Command table" (for example "Value for "Velocity" is invalid")**
  There is a configuration error if one or more parameters are incorrectly set in the axis command table or if programmable configuration data have been modified incorrectly during runtime of the program. If the axis is in motion, the motion is continued. The errors are only indicated in the "MC_CommandTable" Motion Control instruction.
- **Internal error**
  When an internal error occurs, the axis is stopped. The errors are indicated in the error-triggering Motion Control instruction and, in some cases, in the Motion Control instruction "MC_Power".

A detailed description of the ErrorIDs and ErrorInfos, as well as their remedies, is available in the Appendix (Page 1550).

See also

Overview of the Motion Control statements (Page 1426)

Creating a user program (Page 1427)

Programming notes (Page 1430)

Behavior of the Motion Control commands after POWER OFF and restart (Page 1432)

List of ErrorIDs and ErrorInfos (technology objects V4...5) (Page 1550)

Monitoring active commands (Page 1432)

Restart of technology objects (S7-1200)

Description

After the CPU is switched on, or after technology objects are downloaded into the CPU, the system automatically initializes the technology objects with the start values from the technology data block. If restart-relevant changes are detected during a reload into the CPU, a restart of the technology object is automatically performed.

If restart-relevant data have been changed in RUN mode by the user program, then the technology object must be reinitialized by the user in order for the changes to be used.

If changes in the technology data block should also be retained after the restart of the technology object, then you must write the changes to the start value in load memory using the extended instruction "WRIT_DBL".

Restart necessary

If a restart of the technology object is necessary, this will be indicated under "Technology object > Diagnostics > Status and error bits > Status messages > Restart required", and in the tags of the technology object <Axis name>.StatusBits.RestartRequired.

Restarting a technology object

A restart of the technology object is triggered by the user by means of the "MC_Reset" Motion Control instruction, with parameter "Restart" = TRUE.

A restart resets the "Homed" status of a technology object with incremental actual values (<Axis name>.StatusBits.HomingDone).

Parameter transfer for function blocks (S7-1200)

If you want to reuse a function block with Motion Control instructions for different technology objects, create an input parameter of the data type of the respective technology object in the block interface of the function block. You assign the data type in the block interface with direct input. The parameter is then transferred as a reference to the technology object to the "axis" parameter of the Motion Control instructions. The data types of technology objects correspond to the structure of the associated technology data block.

By specifying the data type, you can address the tags of the technology object in the function block (<parameters of the block interface>.<tag of the technology object>).

If you do not need access to the tags of the technology object, you can use the data type "DB_ANY". The data type "DB_Any" can be used to achieve more variable programming.

The following table shows the data types for reference to the technology objects:

Technology object	Data type for reference to the technology object
Positioning axis	TO_PositioningAxis
Command table	TO_CommandTable

Example 1

The following table shows the definition of the tags used:

Operand	Declaration	Data type	Description
axis	Input	TO_PositioningAxis	Reference to the technology object
on	Input	BOOL	Signal to enable the axis
actPosition	Output	Real	Query of the actual position from the technology data block
instMC_POWER	Static	MC_POWER	Multi-instance of the Motion Control instruction MC_Power

The following SCL program shows how to implement this task:

SCL	Explanation
`#instMC_POWER(Axis := #axis, Enable := #on);`	`//Call of the Motion Control instruction MC_Power with enable of the axis`
`#actPosition := #axis.ActualPosition;`	`//Query of the actual position from the technology data block`

Example 2

The data type "DB_Any" provides another option for transferring the data types of a technology object. The data type "DB_Any" can be assigned in the program during runtime.

The example shows two options for transferring technology-specific data types to a corresponding instruction, e.g. "MC_CommandTable", which has been created as a multi-instance. The first option shows the use of the data type "TO_PositioningAxis". The second option shows the simply transfer of the command table technology object as a function of the "cmdTablToUse" input. Depending on the value at the input, one of the three "cmdTablx" inputs is transferred to the "MC_CommandTable" instruction via "tempCmdTableSel".

The following table shows the declaration of the tags used:

Tag	Declaration	Data type	Description
axis	Input	TO_PositioningAxis	Positioning axis
cmdTabl1	Input	DB_ANY	1st command table
cmdTabl2	Input	DB_ANY	2nd command table
cmdTabl3	Input	DB_ANY	3rd command table
cmdTablToUse	Input	Int	Selection, command tables 1 to 3
instMC_Com-mandTable	Static	MC_CommandTable	Multi-instance of the MC_CommandTable
tempCmdTable-Sel	Temp	DB_ANY	Current command table

The example below shows the basic procedure:

```
SCL                                                  Description
CASE #cmdTablToUse OF
    1:  #tempCmdTableSel := #cmdTabl1;               //Program for scenario 1
    2:  #tempCmdTableSel := #cmdTabl2;               //Program for scenario 2
    3:  #tempCmdTableSel := #cmdTabl3;               //Program for scenario 3
    ELSE                                             //Program for all other values
        #tempCmdTableSel := #cmdTabl1;               //->Default setting 1st command table
END_CASE;
                                                     // Call of the "MC_CommandTable" instruction
                                                     //with variable transfer of the technology objects
#instMC_CommandTable(Axis:=#axis,                    //Assignment of axis
    CommandTable:=#temCmdTableSel);                  //Indirect assignment of the command table
```

Additional information

You can find more program examples using the data type "DB_Any" in the following FAQ:

https://support.industry.siemens.com/cs/ww/en/view/109750880 (https://support.industry.siemens.com/cs/ww/en/view/109750880)

6.1.1.10 Axis - Diagnostics (S7-1200)

Status and error bits (technology objects as of V4) (S7-1200)

You use the "Status and error bits" diagnostic function to monitor the most important status and error messages for the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The displayed status and error messages have the following meaning:

Status messages

Status message - Axis	Description
Enabled	The axis is enabled and ready to be controlled via Motion Control commands. (Tag of the technology object: <axis name>.StatusBits.Enable)
Homed	The axis is homed and is capable of executing absolute positioning commands of Motion Control instruction "MC_MoveAbsolute". The axis does not have to be homed for relative positioning. Special situations: • During active homing, the status is FALSE. • If a homed axis undergoes passive homing, the status is set to TRUE during passive homing. (Tag of the technology object: <axis name>.StatusBits.HomingDone)
Axis error	An error has occurred in the "Axis" technology object. Additional information about the error is available in automatic control at the ErrorID and ErrorInfo parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error. (Tag of the technology object: <axis name>.StatusBits.Error)
Control panel active	The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the "Axis" technology object. The axis cannot be controlled from the user program. (Tag of the technology object: <axis name>.StatusBits.ControlPanelActive)
Restart required	A modified configuration of the axis was downloaded to the load memory in CPU RUN mode. To download the modified configuration to the work memory, you need to restart the axis. Use the Motion Control instruction MC_Reset to do this. (Tag of the technology object: <axis name>.StatusBits.RestartRequired)

Status message - Drive	Description
Ready	The drive is ready for operation. (Tag of the technology object: <axis name>.StatusBits.DriveReady)
Drive error	The drive has reported an error due to loss of its "Drive ready" signal. (Tag of the technology object: <axis name>.ErrorBits.DriveFault)

Status message - Motion	Description
Standstill	The axis is at a standstill. (Tag of the technology object: <axis name>.StatusBits.StandStill)
Acceleration	The axis accelerates. (Tag of the technology object: <axis name>.StatusBits.Accelerating)

Status message - Motion	Description
Constant velocity	The axis travels at constant velocity. (Tag of the technology object: <axis name>.StatusBits.ConstantVelocity)
Deceleration	The axis decelerates (slows down). (Tag of the technology object: <axis name>.StatusBits.Decelerating)

Status message - Motion type	Description
Positioning	The axis executes a positioning command of the Motion Control instruction "MC_MoveAbsolute", "MC_MoveRelative" or the axis control panel. (Tag of the technology object: <axis name>.StatusBits.PositioningCommand)
Travel with velocity specification	The axis executes a command with velocity specification of the Motion Control instruction "MC_MoveVelocity", "MC_MoveJog" or the axis control panel. (Tag of the technology object: <axis name>.StatusBits.VelocityCommand)
Homing	The axis executes a homing command of the Motion Control instruction "MC_Home" or the axis control panel. (Tag of the technology object: <axis name>.StatusBits.HomingCommand)
Command table active	The axis is controlled by Motion Control instruction "MC_CommandTable". (Tag of the technology object: <axis name>.StatusBits.CommandTableActive)

Limit switch status messages

Limit switch status message	Description
SW low limit switch has been reached	A software limit switch was reached or exceeded. (Tag of the technology object: <axis name>.StatusBits.SWLimitMinActive)
SW high limit switch has been reached	A hardware limit switch was reached or exceeded. (Tag of the technology object: <axis name>.StatusBits.SWLimitMaxActive)
HW low limit switch was reached	The hardware low limit switch was reached or exceeded. (Tag of the technology object: <axis name>.StatusBits.HWLimitMinActive)
HW high limit switch was reached	The hardware high limit switch was reached or exceeded. (Tag of the technology object: <axis name>.StatusBits.HWLimitMaxActive)

Error messages

Error message	Description
SW limit switch has been reached	A software limit switch was reached or exceeded. (Tag of the technology object: <axis name>.ErrorBits.SWLimit)
HW limit switch has been reached	A hardware limit switch was reached or exceeded. (Tag of the technology object: <axis name>.ErrorBits.HWLimit)
Invalid direction of motion	The motion direction of the command does not match the configured motion direction. (Tag of the technology object: <axis name>.ErrorBits.DirectionFault)
PTO already in use	A second axis is using the same PTO (Pulse Train Output) and HSC (High Speed Counter) and is enabled with "MC_Power". (Tag of the technology object: <axis name>.ErrorBits.HWUsed)

Error message	Description
Encoder	Error in the encoder system. (Tag of the technology object: <axis name>.ErrorBits.SensorFault)
Data exchange	Error in communication with a connected device. (Tag of the technology object: <axis name>.ErrorBits.CommunicationFault)
Positioning	The axis was not correctly positioned at the end of a positioning motion. (Tag of the technology object: <axis name>.ErrorBits.PositionigFault)
Following error	The maximum permitted following error was exceeded. (Tag of the technology object: <axis name>.ErrorBits.FollowingErrorFault)
Encoder values are invalid	The encoder values are invalid. (Tag of the technology object: <axis name>.StatusSensor.State)
Configuration error	The "Axis" technology object was incorrectly configured or editable configuration data were modified incorrectly during runtime of the user program. (Tag of the technology object: <axis name>.ErrorBits.ConfigFault)
Internal error	An internal error has occurred. (Tag of the technology object: <axis name>.ErrorBits.SystemFault)

The output window below shows the first reported and still unacknowledged error.

See also

StatusBits tags V4...5 (Page 1614)

ErrorBits tags V4...5 (Page 1619)

Diagnostics - Status and error bits ("Axis" technology object V1...3) (Page 1547)

Compatibility list of variables V1...3 <-> V4...5 (Page 1331)

Motion status (Page 1450)

Motion status (S7-1200)

Use the "Motion status" diagnostic function to monitor the motion status of the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The displayed status information has the following meaning:

Status	Description
Actual position	The "Actual position" box indicates the measured position of the axis. If the axis is not homed, the value indicates the position value relative to the enable position of the axis. (Tag of the technology object: <axis name>.ActualPosition)
Actual velocity	The "Actual velocity" box indicates the measured velocity of the axis. (Tag of the technology object: <axis name>.ActualVelocity)
Position setpoint	The "Position setpoint" box indicates the measured position setpoint of the axis. If the axis is not homed, the value indicates the position value relative to the enable position of the axis. (Tag of the technology object: <axis name>.Position)
Velocity setpoint	The "Velocity setpoint:" box indicates the calculated velocity setpoint of the axis. (Tag of the technology object: <axis name>.Velocity)

Status	Description
Target position	The "Target position" box indicates the current target position of an active positioning command or of the axis command table. The value of the "Target position" is only valid during execution of a positioning command. (Tag of the technology object: <axis name>.StatusPositioning.TargetPosition)
Remaining travel distance	The "Remaining travel distance" box indicates the travel distance currently remaining for an active positioning command or the axis command table. The "Remaining travel distance" value is only valid during execution of a positioning command. (Tag of the technology object: <axis name>.StatusPositioning.Distance)

See also

Tags to position values and velocity values V4...5 (Page 1595)

StatusPositioning tags V4...5 (Page 1612)

Compatibility list of variables V1...3 <-> V4...5 (Page 1331)

Tags for position values and velocity values as of V6 (Page 1490)

MotionStatus tags V1...3 (Page 1589)

Status and error bits (technology objects as of V4) (Page 1447)

Dynamics settings (S7-1200)

Use the "Dynamics settings" diagnostic function to monitor the dynamic limits of the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The displayed status information has the following meaning:

Dynamic limit	Description
Acceleration	The "Acceleration" box indicates the currently configured acceleration of the axis. (Tag of technology object: <Axis name>.DynamicDefaults.Acceleration)
Deceleration	The "Deceleration" box indicates the currently configured deceleration of the axis. (Tag of technology object: <Axis name>.DynamicDefaults.Deceleration)
Emergency deceleration	The "Emergency deceleration" box indicates the currently configured emergency stop deceleration of the axis. (Tag of technology object: <Axis name>.DynamicDefaults.EmergencyDeceleration)
Jerk (axis technology object as of V2)	The "Velocity" box indicates the current axis step velocity configured. (Tag of technology object: <Axis name>.DynamicDefaults.Jerk)

See also

DynamicDefaults tags V4...5 (Page 1605)

Compatibility list of variables V1...3 <-> V4...5 (Page 1331)

PROFIdrive frame (S7-1200)

The "Technology object > Diagnostics > PROFIdrive telegram" diagnostics function is used in the TIA Portal to monitor the PROFIdrive telegrams returned by the drive and encoder. The display of the Diagnostics function is available in online operation.

"Drive" area

This area displays the following parameters contained in the PROFIdrive telegram from the drive to the controller:

- Status words "SW1" and "SW2"
- The setpoint speed that was output to the drive (NSET)
- The actual speed that was signaled from the drive (NACT)

"Encoder" area

This area displays the following parameters contained in the PROFIdrive telegram from the encoder to the controller:

- Status word "G1_ZSW"
- The actual position value "G1_XIST1" (cyclic actual encoder value)
- The actual position value "G1_XIST2" (absolute value of the encoder)

6.1.1.11 Appendix (S7-1200)

Using multiple axes with the same PTO (S7-1200)

Use the Motion Control functionality of the CPU S7-1200 to run multiple positioning axis technology objects with the same PTO (Pulse Train Output) and thus with the same CPU outputs. This is appropriate, for example, if different axis configurations are to be used for different production sequences via one PTO. As described below, it is possible to switch between these axis configurations as often as necessary. The following diagram presents the basic functional relationships:

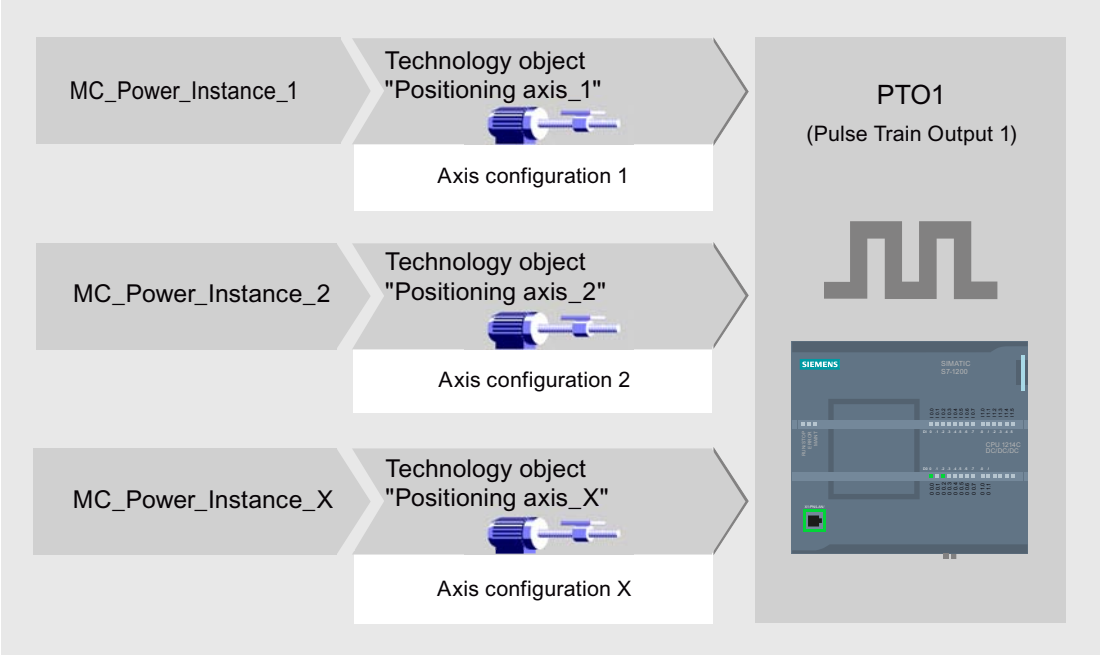

In this example, several positioning axis technology objects, each with its own axis configuration, use the same PTO. Each axis must be called in the user program with a separate call of Motion Control instruction "MC_Power" with a separate instance data block. Only one axis at a time may use the PTO. The axis that is currently using the PTO indicates this with tag <Axis name>.StatusBits.Activated = TRUE.

Switchover of the positioning axis technology object

The program scheme described below shows you how to switch between different technology objects and, thus, between different axis configurations. To use the same PTO with multiple axes without error indications, only the Motion Control instructions of the axis currently being used may be called.

The following diagram presents this principle using Motion Control instruction "MC_Power" as an example:

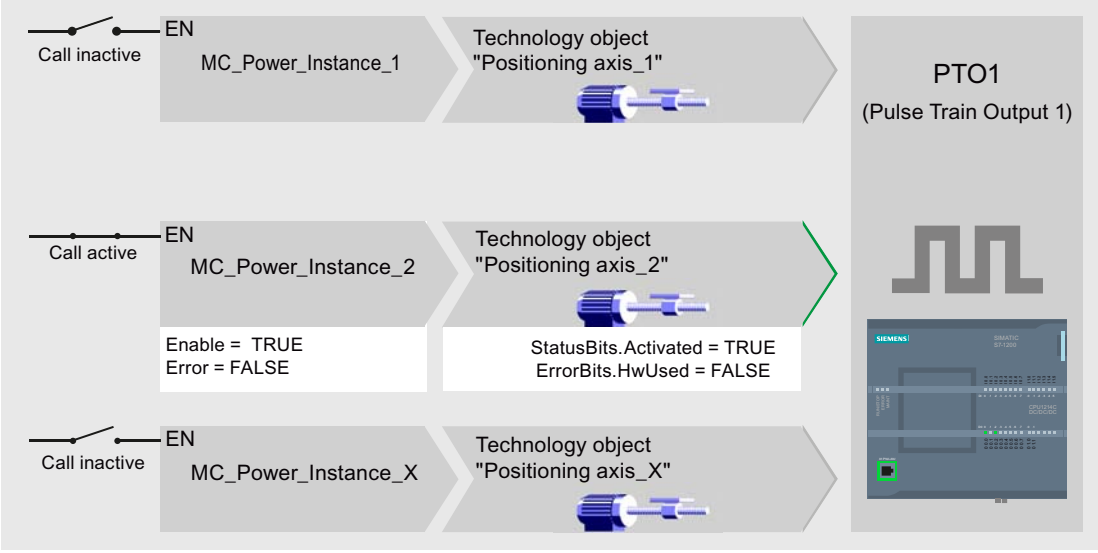

The tags of the activated axis ("Positioning axis_2" here) show the following typical indicators in the user program:

- <Axis name>.StatusBits.Activated = TRUE
- <Axis name>.ErrorBits.HWUsed = FALSE

To switch from one positioning axis technology object to another, follow the steps described below. In the example, a switch is made from "Positioning axis_2" to "Positioning axis_1":

1. End any active traversing motions of activated "Positioning axis_2"
2. Disable "Positioning axis_2" with the associated Motion Control instruction "MC_Power" using input parameter Enable = FALSE
3. To verify that "Positioning axis_2" has been disabled, use an AND operation of output parameter Status = FALSE of Motion Control instruction "MC_Power" and technology object tag <Axis name>.StatusBits.Enable = FALSE.
4. Deactivate the conditional call of the Motion Control instructions for "Positioning axis_2".
5. Activate the conditional call of the Motion Control instruction for "Positioning axis_1". At the first call of the corresponding Motion Control instruction "MC_Power", "Positioning axis_2" is deactivated and "Positioning axis_1" is activated.
6. Enable "Positioning axis_1" with the associated Motion Control instruction "MC_Power" using input parameter Enable = TRUE.
7. To verify that "Positioning axis_1" has been enabled, use an AND operation of output parameter Status = TRUE of Motion Control instruction "MC_Power" and technology object tag <Axis name>.StatusBits.Enable = TRUE.

It is also always possible to cyclically call all Motion Control instructions of all axes working with a single PTO.

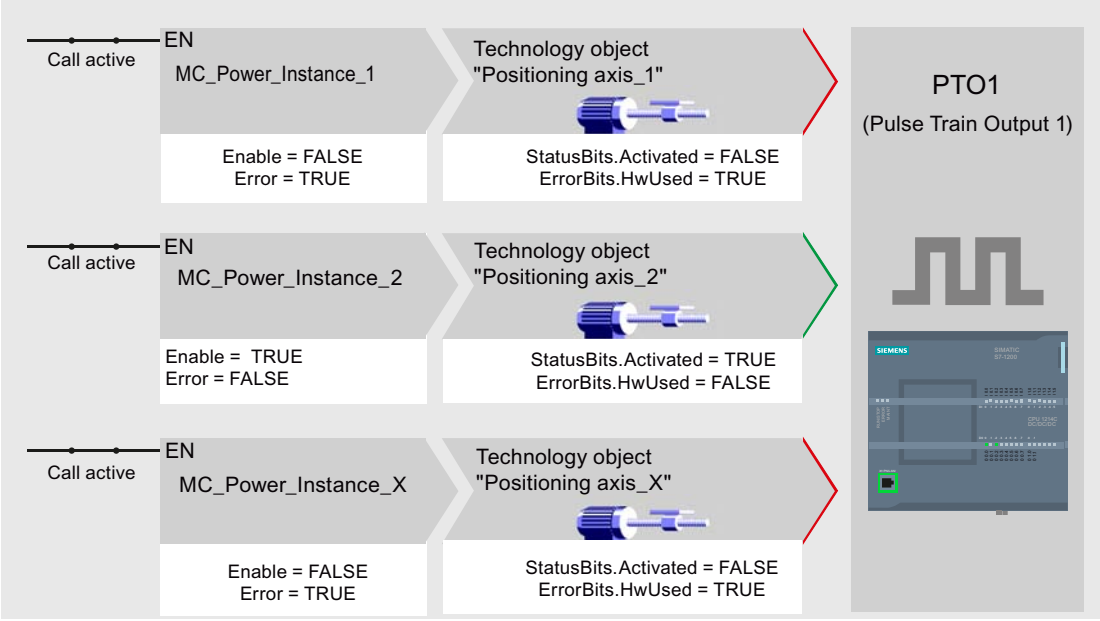

When an axis is enabled (here "Positioning axis_2"), this axis becomes active.

In contrast to the conditional call, the Motion Control instructions of the deactivated axes (here "Positioning axis_1" and "Positioning axis_x") indicate errors. The tags of these axes indicate the status <Axis name>.StatusBits.Activated = FALSE and <Axis name>.ErrorBits.HWUsed = TRUE.

Use the conditional call of the Motion Control instructions if you want to implement the user program without error indicators.

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Using multiple drives with the same PTO (Page 1456)

Tracking jobs from higher priority classes (execution levels) (Page 1457)

Special cases when using software limit switches for drive connection via PTO (Page 1459)

List of ErrorIDs and ErrorInfos (technology objects V4...5) (Page 1550)

Using multiple drives with the same PTO (S7-1200)

If multiple drives are to be used, they can be run with a common PTO (Pulse Train Output) using changeover. The following diagram represents the basic circuit design:

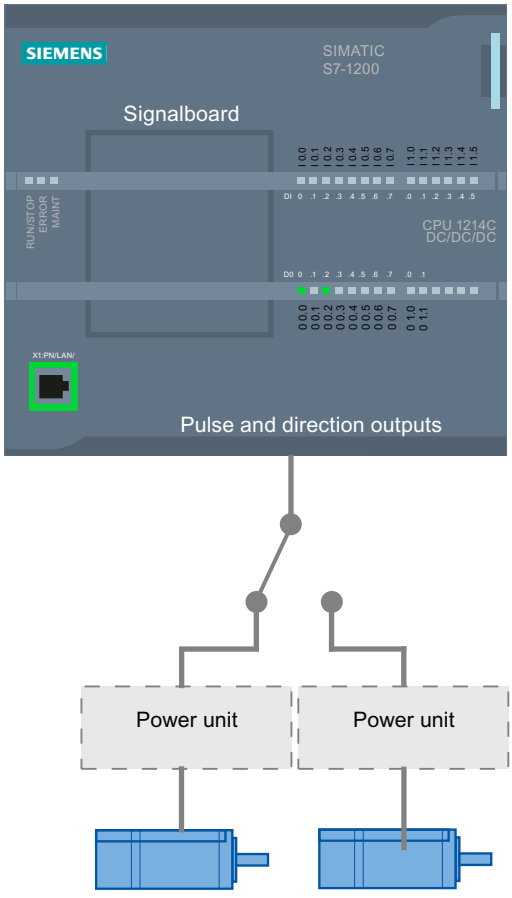

The changeover between drives can be controlled, if required, by the user program via a digital output. If different axis configurations are required for the different drives, a changeover between these configurations is required for the PTO. For additional information on this topic, refer to "Using multiple axes with the same PTO (Page 1452)".

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Using multiple axes with the same PTO (Page 1452)

Tracking jobs from higher priority classes (execution levels) (Page 1457)

Special cases when using software limit switches for drive connection via PTO (Page 1459)

List of ErrorIDs and ErrorInfos (technology objects V4...5) (Page 1550)

Tracking jobs from higher priority classes (execution levels) (S7-1200)

Depending on the application, it may be necessary to start Motion Control commands (for example, interrupt-controlled) in a higher priority class (execution level).

The Motion Control instructions must be called at short intervals for status monitoring. Motion Control commands cannot be sufficiently closely monitored if the higher priority Motion Control commands are called only once or at too great an interval. Tracking in such cases can be carried out in the cycle OB. An instance data block that is not currently being utilized must be available for each start of a higher priority Motion Control command. Refer to the following flow chart to see how you start Motion Control commands in a higher priority class (for example, hardware interrupt OB) and continue tracking in the program cycle OB:

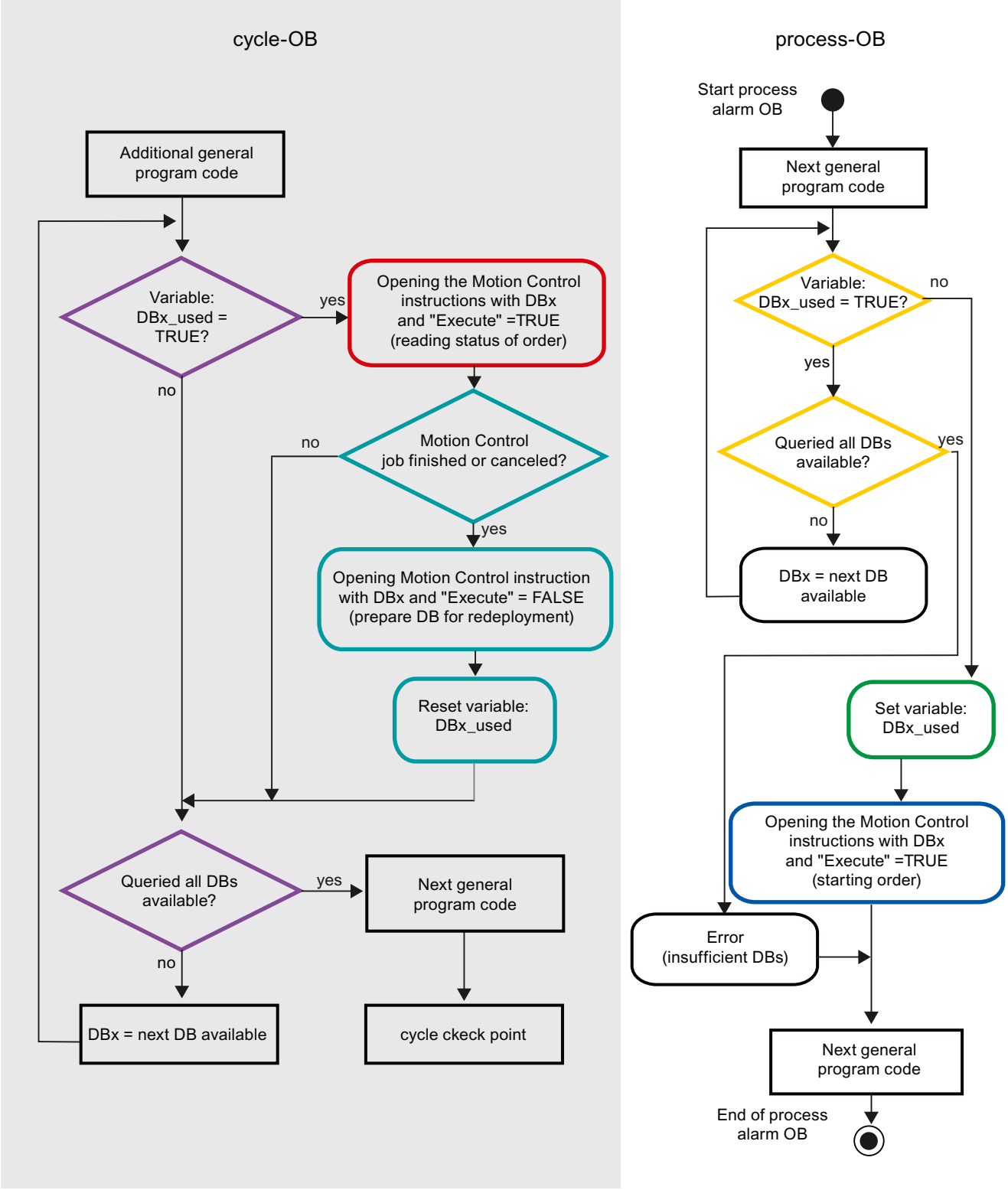

Depending on the frequency of the Motion Control commands you want to start, you will have to generate a sufficient number of instance data blocks. Users determine which instance data block is currently used in the DBx_used tags.

Start of Motion Control command in the hardware interrupt OB

Binary queries of the DBx_used tags (orange) are used to find an instance data block not currently in use. If such an instance data block is found, the utilized instance data block is marked as "used" (green) and the Motion Control command is started with this instance data block (blue).

Any other program sections of the hardware interrupt OB are then executed, followed by a return to the program cycle OB.

Tracking of started Motion Control commands in the program cycle OB

All instance data blocks available in the cycle OB are checked to determine if they are currently in use by means of the DBx_used tag (violet).

If an instance data block is in use (Motion Control command is being processed), the Motion Control instruction with this instance data block and input parameter Execute = TRUE is called to read out the status messages (red).

If the command is complete or has been aborted, the following actions are taken next (blue green):

- Call of Motion Control instruction with input parameter Execute = FALSE
- Resetting the DBx_used tag

This completes the command tracking, and the instance data block is now available for use again.

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Using multiple axes with the same PTO (Page 1452)

Using multiple drives with the same PTO (Page 1455)

Special cases when using software limit switches for drive connection via PTO (Page 1459)

List of ErrorIDs and ErrorInfos (technology objects V4...5) (Page 1550)

Special cases when using software limit switches for drive connection via PTO (S7-1200)

Software limit switches in conjunction with a homing operation (S7-1200)

Due to unfavorably parameterized homing jobs, the braking action of the axis may be influenced at the software limit switch. Take the following examples into consideration when developing your program.

Example 1:

During a travel command, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. It is still possible to bring the axis to a standstill before reaching the software limit switch:

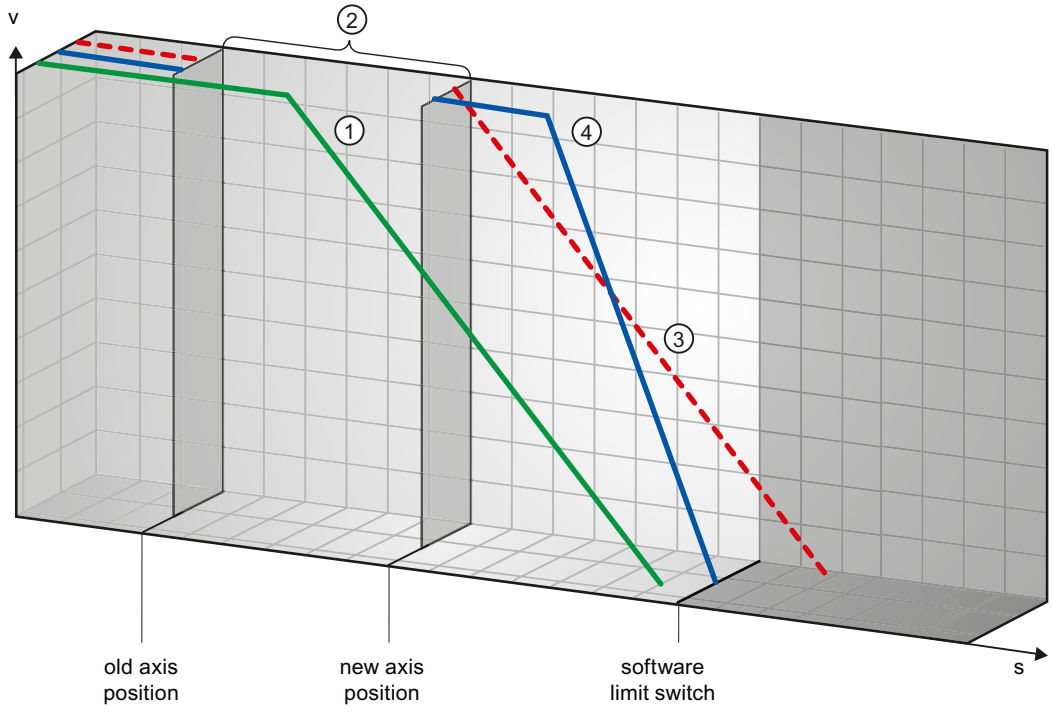

①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. Following constant motion, the axis brakes at the configured emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 2:

During a travel command, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. In contrast to example 1, it is no longer possible to bring the axis to a standstill before reaching the software limit switch. The axis overruns the position of the software limit switch.

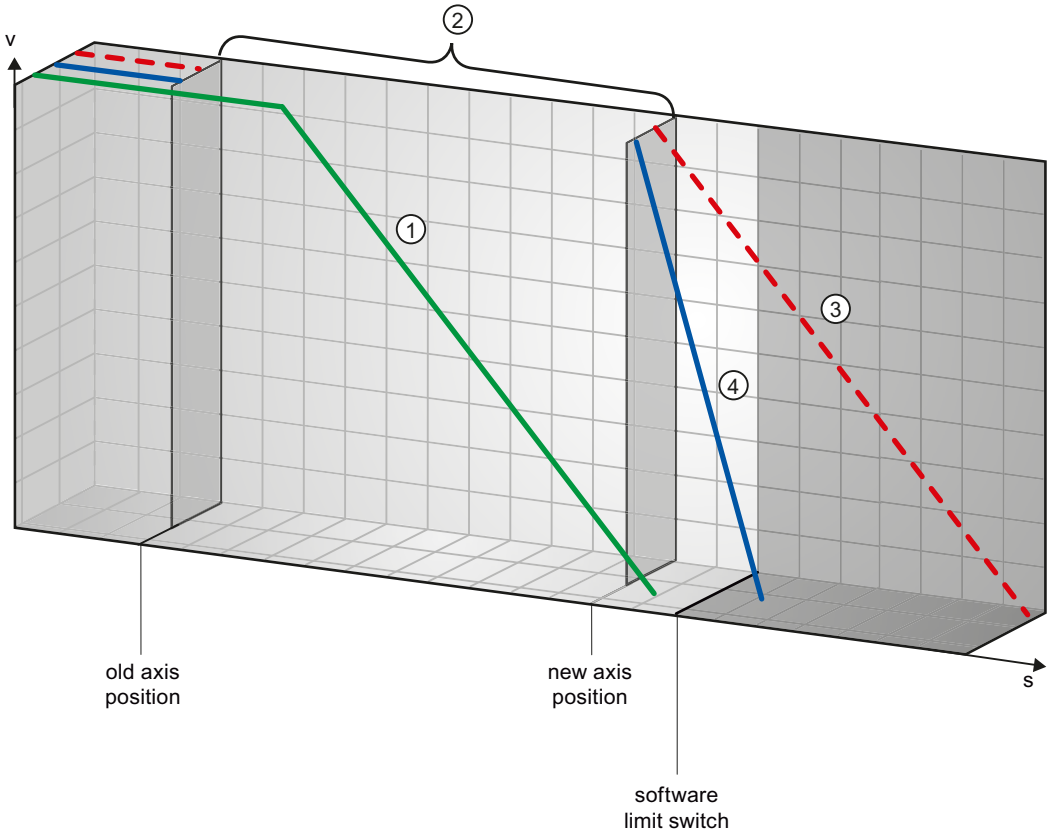

①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position well after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. The axis brakes with the configured emergency stop deceleration. However, the emergency stop deceleration is not sufficient to stop the axis at the position of the software limit switch. The position of the software limit switch is overrun.

Example 3:

During a braking operation, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. It is still possible to bring the axis to a standstill before reaching the software limit switch:

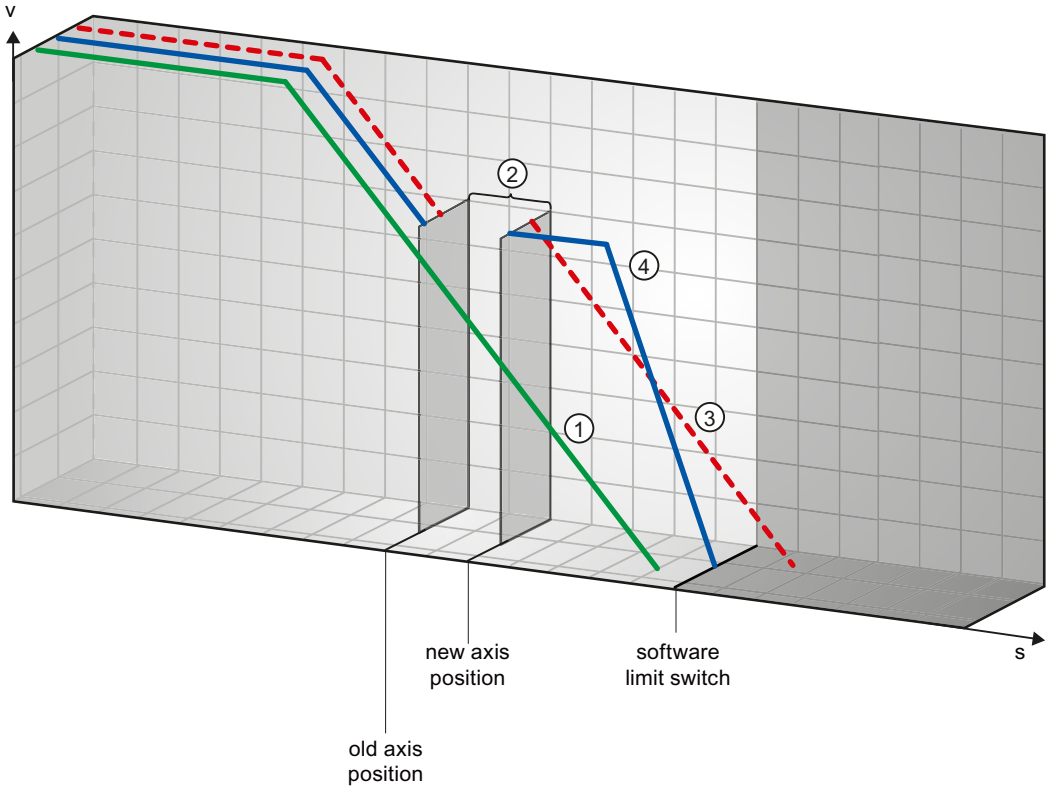

①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. Following constant motion, the axis brakes at the configured emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 4:

During a braking operation, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. In contrast to example 3, it is no longer possible to bring the axis to a standstill before reaching the software limit switch. The axis overruns the position of the software limit switch.

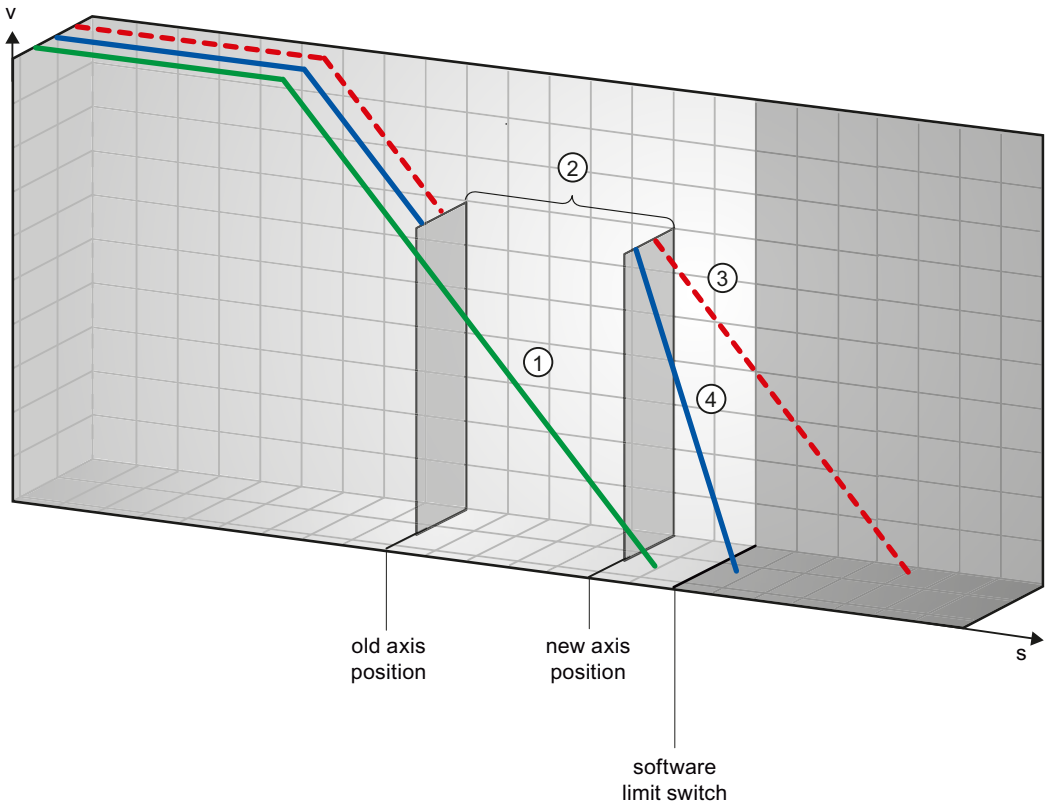

①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position well after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. The axis brakes with the configured emergency stop deceleration. However, the emergency stop deceleration is not sufficient to stop the axis at the position of the software limit switch. The position of the software limit switch is overrun.

See also

Software limit switches and software limit switch position changes. (Page 1463)

Software limit switches in conjunction with dynamic changes (Page 1464)

Response of the axis when position limits are triggered (Page 1360)

Software limit switches and software limit switch position changes. (S7-1200)

An incorrect change in the position of the software limit switch during the runtime of the user program can abruptly reduce the distance between the current axis position and the position of the software limit switch.

The axis response is similar to that described in Software limit switches in conjunction with a homing operation (Page 1458).

See also

Software limit switches in conjunction with a homing operation (Page 1458)

Software limit switches in conjunction with dynamic changes (Page 1464)

Response of the axis when position limits are triggered (Page 1360)

Software limit switches in conjunction with dynamic changes (S7-1200)

It is possible to influence the deceleration of the axis in the area of the software limit switches in conjunction with overriding motion commands. This applies when the overriding motion command is started with a lower deceleration (tag <Axis name>.DynamicDefaults.Deceleration). Take the following examples into consideration when developing your program.

Example 1:

During axis motion, an active motion command is overridden by another motion command with a lower deceleration:

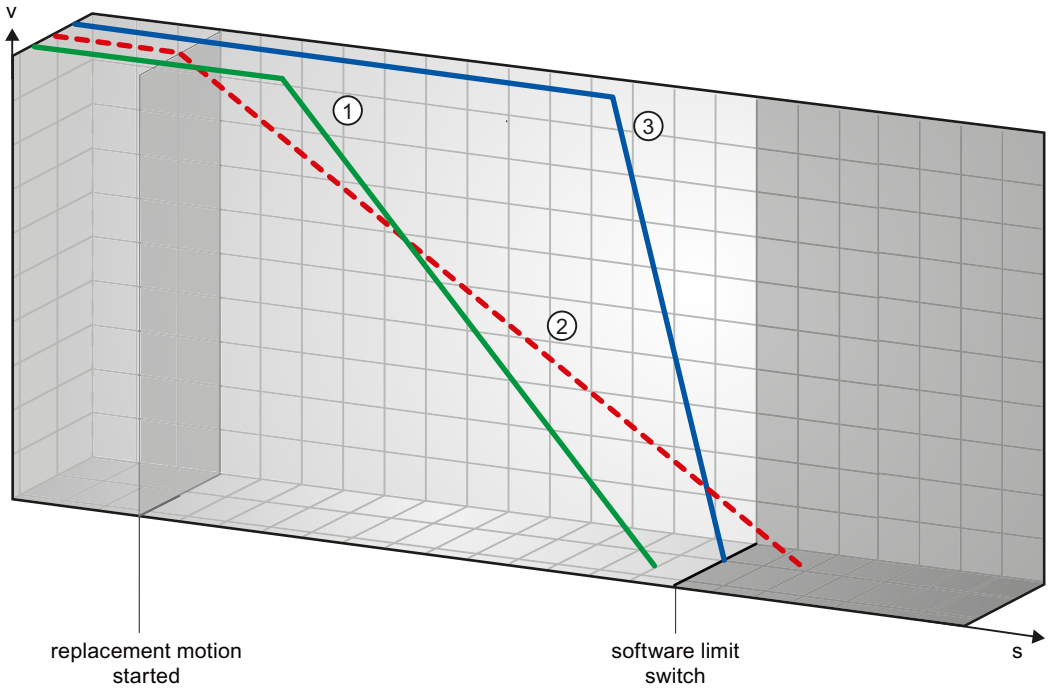

①	The green curve shows the motion of an active command **without** overriding this command. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	Based on the overriding motion command with lower deceleration, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
③	Because braking with the configured deceleration of the overriding motion command is no longer sufficient, the axis actually follows the blue curve. Following a constant motion, the axis brakes at the emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 2:

During braking of the axis, an active motion command is overridden by another motion command with a lower deceleration:

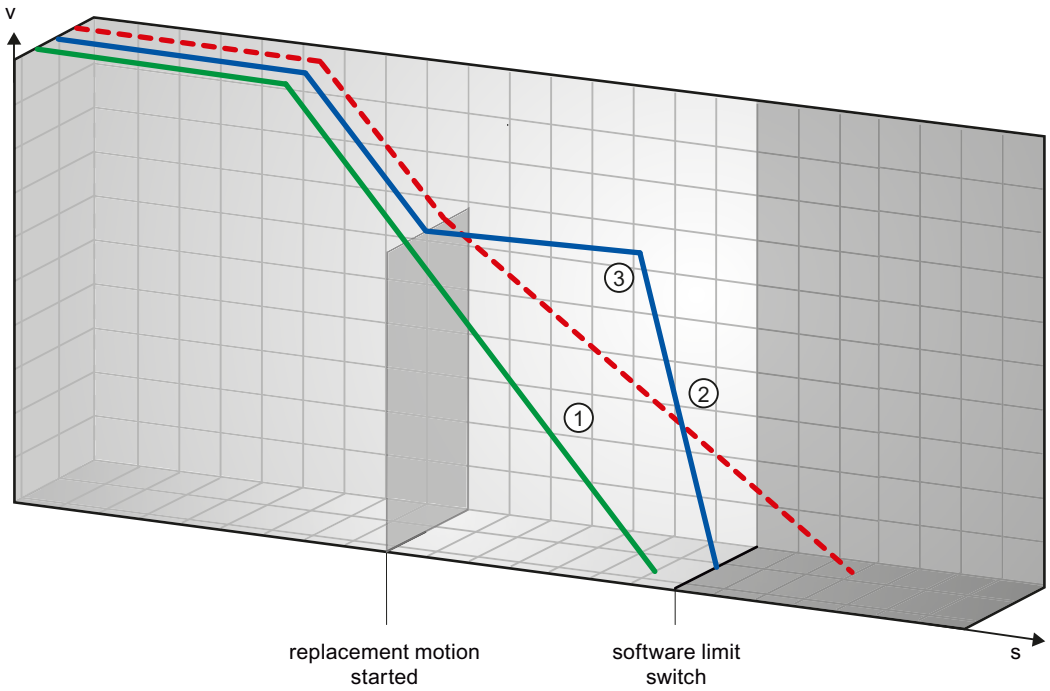

①	The green curve shows the motion of an active command **without** overriding this command. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	Based on the overriding motion command with lower deceleration, the axis would theoretically be stopped at a position well after the software limit switch (red curve).
③	Because braking with the configured deceleration of the overriding motion command is no longer sufficient, the axis actually follows the blue curve. Following a constant motion, the axis brakes at the emergency stop deceleration and comes to a standstill at the position of the software limit switch.

See also

Software limit switches in conjunction with a homing operation (Page 1458)

Software limit switches and software limit switch position changes. (Page 1462)

Response of the axis when position limits are triggered (Page 1360)

Reducing velocity for a short positioning duration (S7-1200)

The CPU can reduce the velocity of a positioning command when the planned positioning duration is < 2 ms.

The velocity of command execution will then be reduced for the entire duration. The reduced velocity (pulses per s) is calculated as follows:

- Reduced velocity = Number of pulses to be output * 500Hz

Velocity is **not** reduced if the planned positioning duration is >= 2 ms.

Dynamic adjustment of start/stop velocity (S7-1200)

The configuration of your velocity limits (start/stop velocity and maximum velocity), the dynamic values (acceleration, deceleration and jerk) and the target speed of the traversing command may under certain circumstances result in the start/stop velocity being dynamically adjusted by the CPU.

This is the case, for example, if the required time for the first pulse would be longer than required for the entire acceleration due to a configured low start/stop velocity. The first pulse is in these cases output at a greater velocity than the configured start/stop velocity. The subsequent pulses are also dynamically adjusted to ensure the acceleration process can be completed in the specified time.

If any pulse loss occurs, make sure that hardware (drive) you are using is adapted to this situation or change the dynamic settings of your axis to avoid the dynamic adjustment of the start/stop velocity.

Move the axis without position control for servicing (S7-1200)

If service is required, it may be necessary to move a PROFIdrive drive or a drive with an analog drive interface without position control.

This may be the case for example with invalid or incorrect encoder values.

The following Motion Control instructions influence the status of the position control:

Enable position-controlled drives without position control with MC_Power

Start the axis with the Motion Control instruction MC_Power StartMode = 0 in non-position-controlled operation.

The non-position-controlled mode is in effect until another Motion Control instruction changes the status of the position control.

MC_MoveVelocity

MC_MoveVelocity with PositionControlled = FALSE forces non-position-controlled operation.

MC_MoveVelocity with PositionControlled = TRUE forces position-controlled operation.

The selected position-controlled operation is retained even after the termination of MC_MoveVelocity.

MC_MoveJog

MC_MoveJog with PositionControlled = FALSE forces non-position-controlled operation.

MC_MoveJog with PositionControlled = TRUE forces position-controlled operation.

The selected position-controlled operation is retained even after the termination of MC_MoveJog.

MC_Home, MC_MoveRelative, MC_MoveAbsolute

The Motion Control instructions MC_Home, MC_MoveRelative and MC_MoveAbsolute force position-controlled operation.

The position control remains active even when the command has ended.

MC_Halt

The Motion Control instruction MC_Halt is executed in position-controlled and non-position-controlled operation.

The status of the position control is not changed.

List of ErrorIDs and ErrorInfos (technology objects as of V6) (S7-1200)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, solutions for eliminating the error are also listed:

Depending on the error reaction, the axis is stopped in the case of operating errors with stop of axis. The following error reactions are possible:

- **Remove enable**
  The setpoint zero is output and the enable is removed. The axis is braked depending on the configuration in the drive, and is brought to a standstill.
- **Stop with emergency stop ramp**
  Active motion commands are aborted. The axis is braked with the emergency stop deceleration configured under "Technology object > Extended parameters > Dynamics > Emergency stop ramp" without any jerk limit and brought to a standstill.

Operating error with axis stop

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8000		**Drive error, loss of "Drive ready"**		-
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; restart command, if necessary	
16#8001		**SW low limit switch has been tripped**		-
	16#000E	The position of the SW low limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the SW limit switch	
	16#000F	The position of the SW low limit switch was reached with the emergency stop deceleration		
	16#0010	The position of the SW low limit switch was exceeded with the emergency stop deceleration		Remove enable

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8002		SW high limit switch has been tripped		-
	16#000E	The position of the SW high limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the SW limit switch	
	16#000F 0068	The position of the SW high limit switch was reached with the emergency stop deceleration		
	16#0010	The position of the SW high limit switch was exceeded with the emergency stop deceleration		Remove enable
16#8003		HW low limit switch was reached		For drive connection via PTO (Pulse Train Output): Stop with emergency stop ramp For drive connection via PROFIdrive/analog output: Remove enable
	16#000E	The HW low limit switch was reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the HW limit switch.	
16#8004		HW high limit switch was reached		For drive connection via PTO (Pulse Train Output): Stop with emergency stop ramp For drive connection via PROFIdrive/analog output: Remove enable
	16#000E	The HW high limit switch was reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the HW limit switch.	
16#8005		PTO/HSC are already being used by another axis		-
	16#0001	-	**The axis was configured incorrectly:** Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller	
			More than one axis is to run with one PTO: Another axis is using the PTO / HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (See also Using multiple axes with the same PTO (Page 1452))	
16#8006		A communication error in the control panel has occurred		Remove enable
	16#0012	A timeout has occurred	Check the cable connection and press the "Manual control" button again	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8007		The axis cannot be enabled		-
	16#0025	Restarting	Wait until the axis restart is complete.	
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.	
16#8008		Invalid direction of motion		-
	16#002E	The selected motion direction is not allowed.	• Adjust the motion direction and restart the command. • Adjust the allowed direction of rotation in the technology object configuration under "Extended parameters > Mechanics". Restart the job.	
	16#002F	A reversing motion is not possible with the selected motion direction.		
16#8009		Reference switch/encoder zero mark not found		Stop with emergency stop ramp
	16#0033	Error in the configuration, hardware or installation of the encoder or at the homing switch.	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.	
16#800A		Alarm message from encoder		Remove enable
	16#0001	-	Check the device with regard to function, connections and I/Os.	
	16#0034	Hardware error at encoder		
	16#0035	Encoder dirty		
	16#0036	Error during reading of encoder absolute value	Compare the encoder type in the drive or encoder parameter P979 with the configuration data of the technology object.	
	16#0037	Zero mark monitoring of the encoder	Encoder reports error in zero mark monitoring (fault code 0x0002 in Gx_XIST2, see PROFIdrive profile). Check the plant for electromagnetic compatibility (EMC).	
	16#0038	Encoder is in "Parking" state	• Search for the cause of the error in the connected drive or encoder. • Check whether the error message was possibly triggered by a commissioning action at the drive or encoder.	
	16#0040	PROFIdrive: Encoder on bus failed (station failure).	Check the device with regard to function, connections and I/Os.	
	16#0041	PROFIdrive: Signs of life of encoder faulty.		
16#800B		Range violation of the position		Remove enable
	16#0039	Range violation in positive direction	Home the axis to a valid actual value range.	
	16#003A	Range violation in negative direction		
	16#003B	The change of the actual position in a position control clock cycle is greater than the modulo length.	Adjust the modulo length of the employed encoder.	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#800C		Alarm message from drive		Remove enable
	16#0001	-	Check the device with regard to function, connections and I/Os. In the "Tuning" dialog box, select a smaller gain (Kv).	
	16#003C	PROFIdrive: Drive signal "Control requested" failed.		
	16#003D	PROFIdrive/analog drive connection: Drive has shut down.		
	16#003E	PROFIdrive: Drive on bus failed (station failure)		
	16#003F	PROFIdrive: Signs of life of drive faulty.	• Check the device with regard to function, connections and I/Os. • Compare the clock parameters of HW Config (PROFIBUS line, slave OM for drive or encoder) and the execution system. Tmapc and servo must be configured with the same clock cycle time.	
16#800D		The permitted following error was exceeded		Remove enable
	16#0001	-	• Check the configuration of the control loop. • Check the direction signal of the encoder. • Check the configuration of following error monitoring.	
16#800E		Error at the hardware limit switch		Remove enable
	16#0042	Illegal free travel direction with active hardware limit switch	The programmed direction of movement is disabled due to the active hardware limit switch. Retract the axis in the opposite direction.	
	16#0043	Hardware limit switch polarity is reversed, axis cannot be freed	Check the mechanical configuration of the hardware limit switch.	
	16#0044	Both hardware limit switches are active, axis cannot be freed		
16#800F		Error in target range		Remove enable
	16#0045	Target range not reached	Target range was not reached within the positioning tolerance time. • Check the configuration of the position monitoring. • Check the configuration of the control loop.	
	16#0046	Exit target range again	The target range was exited within the minimum dwell time. • Check the configuration of the position monitoring. • Check the configuration of the control loop.	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8010		Position of the SW low limit switch is greater than that of the SW high limit switch when the axis is not a modulo axis		Remove enable
	16#0001	-	Change the position of the software limit switches.	
16#8011		Approach velocity to the homing switch / zero mark equals zero.		Remove enable
	16#000A	Value is less than or equal to 0.	Approach velocity > Select zero	
16#8012		Homing velocity for setting the home position equals zero.		Remove enable
	16#000A	Value is less than or equal to 0.	Homing velocity > Select zero	
16#8013		The axis cannot occupy the PTO because it is being used by "CTRL_PTO".		Remove enable
	16#0001	-	Select another PTO in the configuration.	

Operating error without axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8200		Axis is not enabled	
	16#0001	-	Enable the axis; restart the command
16#8201		Axis has already been enabled by another "MC_Power" instance	
	16#0001	-	Enable the axis with only one "MC_Power" instance
16#8202		The maximum number of simultaneous Motion Control commands has been exceeded (max. 200 commands for drive connection via PTO (Pulse Train Output), max. 100 commands for drive connection via PROFIdrive/analog output)	
	16#0001	-	Reduce the number of simultaneously active commands; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		Axis is currently operated in "Manual control" (axis control panel)	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		Axis is not homed	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		The axis is currently controlled by the user program (the error is only displayed in the axis control panel)	
	16#0013	The axis is enabled in the user program	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis control panel
16#8206		Technology object not activated yet	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis control panel.
16#8207		Command rejected	

ErrorID	ErrorInfo	Description	Remedy
	16#0001	-	
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt".
	16#0018	The axis cannot be moved with a command table while it is being actively or passively homed.	Wait until direct or passive homing is complete.
	16#0019	The axis cannot be actively or passively homed while a command table is being processed.	Wait for command table to finish or abort the command table with a motion command, for example, "MC_Halt".
	16#0052	The specified position exceeds the numerical limit.	Enter a valid position value at the Motion Control instruction.
	16#0053	The axis is ramping up.	Wait until the axis is ready for operation.
	16#0054	Actual value is invalid	To execute a "MC_Home" command, the actual values must be valid. Check the status of the actual values. The variable of the technology object <axis name>."StatusSensor.State" must show the value 2 (valid).
	16#0058	Command is already used in another execution level.	Call axis only via an "MC_Power" instance.
	16#006B	Call is not permitted in non-position-controlled mode.	Enable the axis with position control with the "MC_Power" with StartMode = 1.
16#8208		Difference between maximum and start/stop velocity is invalid	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#8209		Invalid acceleration for technology object "Axis"	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#820A		It is not possible to restart the axis	
	16#0013	The axis is enabled in the user program	Disable the axis with the "MC_Power" instruction; restart again
	16#0027	The axis is currently being operated in "Manual control" (axis control panel)	Exit "Manual control"; restart again
	16#002C	The axis is not disabled.	Disable the axis; restart the command
	16#0047	The technology object is not ready for restart.	Download the project again.
	16#0048	Condition for restart of the technology object is not satisfied.	Disable the technology object.
16#820B		It is not possible to execute the command table	
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.
16#820C		No configuration available	
	16#0001	-	Internal error Contact the hotline.
	16#0014	The selected hardware is used by another application	

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#0005	Value is outside the number range (greater than 1.0E12)	
	16#0006	Value is outside the number range (less than -1.0E12)	
	16#0030	Value has an incorrect number format or is outside the valid number range	
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#0005	Value is outside the number range (greater than 1.0E12)	
	16#0006	Value is outside the number range (less than -1.0E12)	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/ stop velocity	
	16#0024	Value is less than 0	
	16#0030	Value has an incorrect number format or is outside the valid number range	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	The direction reversal is activated at the hardware limit switch, despite the fact that the hardware limit switches are disabled	• Activate the HW limit switch using the variable <Axis name>.PositionLimits_HW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0055	Invalid mode at incremental encoder	Start a homing process for an incremental encoder using parameter "Mode" = 0, 1, 2, 3.
	16#0056	Invalid mode at absolute encoder	Passive and active homing ("Mode" = 2, 3) are not possible for an absolute value encoder. Start a homing process for an absolute encoder using parameter "Mode" = 0, 1.
16#8405		**Invalid value at parameter "StopMode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; enable the axis again
16#8406		**Simultaneous forward and backward jogging is not allowed**	

ErrorID	ErrorInfo	Description	Remedy
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		**Switching to another axis with instruction "MC_Power" is only permitted after disabling the active axis.**	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.
16#8408		**Invalid value at parameter "Axis" of the Motion Control instruction**	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#8409		**Invalid value at parameter "CommandTable" of the Motion Control instruction**	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#840A		**Invalid value at parameter "StartStep" of the Motion Control instruction**	
	16#000A	Value is less than or equal to 0.	Correct the value; restart the command
	16#001D	The start step is greater than the end step	
	16#001E	Value is greater than 32	
16#840B		**Invalid value at parameter "EndStep" of the Motion Control instruction**	
	16#000A	Value is less than or equal to 0.	Correct the value; restart the command
	16#001E	Value is greater than 32	
16#840C		**Invalid value at parameter "RampUpTime" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840D		**Invalid value at parameter "RampDownTime" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840E		**Invalid value at parameter "EmergencyRampTime" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840F		**Invalid value at parameter "JerkTime" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#8410		**Invalid value at parameter "Parameter" of the Motion Control instruction**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000B	Address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0028	Data type of VARIANT pointer "Parameter" and "Value" do not match.	Use a suitable data type; restart command
	16#0029	VARIANT pointer "Parameter" does not point to a data block of the technology object.	Correct the VARIANT pointer; restart the command
	16#002A	The value at the VARIANT pointer "Parameter" cannot be read.	Correct the VARIANT pointer; restart the command
	16#002B	The value at the VARIANT pointer "Parameter" cannot be written.	Correct the VARIANT pointer or value; restart the command
	16#002C	The axis is not disabled.	Disable the axis; restart the command
16#8411		Invalid value at parameter "Value" of the Motion Control instruction	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#0005	Value is outside the number range (greater than 1.0E12)	
	16#0006	Value is outside the number range (less than -1.0E12)	
16#8412		Value at "Start Mode" parameter of the Motion Control instruction invalid	
	16#0011	The selection value is invalid	Correct the selection value; enable the axis again

Configuration error of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8600		Parameter assignment of pulse generator (PTO is invalid	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8601		Parameter assignment of the high-speed counter (HSC) is invalid	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8602		Invalid parameter assignment of "Enable output"	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8603		Invalid parameter assignment of "Ready input"	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8604		Invalid "Pulses per motor revolution" value	
	16#000A	Value is less than or equal to zero	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8605		Invalid "Distance per revolution" value	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	The value has an invalid number format	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0005	Value is outside the number range (greater than 1.0E12)	
	16#000A	Value is less than or equal to zero	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8606		Invalid "Start/stop velocity" value	
	16#0002	The value has an invalid number format	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value is higher than the hardware high limit	
	16#0004	Value is lower than the hardware low limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		Invalid "maximum velocity" value	
	16#0002	The value has an invalid number format	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value is higher than the hardware high limit	
	16#0004	Value is lower than the hardware low limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8608		Invalid "Acceleration" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the hardware high limit	
	16#0004	Value is lower than the hardware low limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8609		Invalid "Deceleration" value	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the hardware high limit	
	16#0004	Value is lower than the hardware low limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#860A		Invalid "Emergency stop deceleration" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the hardware high limit	
	16#0004	Value is lower than the hardware low limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#860B		Value for position of the SW low limit switch is invalid	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1.0E12)	
	16#0006	Value is outside the number range (less than -1.0E12)	
	16#0030	The position value of the software low limit switch is greater than that of the software high limit switch	
16#860C		Value for position of the SW high limit switch is invalid	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1.0E12)	
	16#0006	Value is outside the number range (less than -1.0E12)	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#860D		Invalid address of the HW low limit switch	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#860E		Invalid address of the HW high limit switch	

ErrorID	ErrorInfo	Description	Remedy
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#860F		**Invalid "home position offset" value**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1.0E12)	
	16#0006	Value is outside the number range (less than -1.0E12)	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8610		**Invalid "approach velocity" value**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8611		**Invalid "Homing velocity" value**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8612		**Invalid address of the homing switch**	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#8613		**During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured**	

ErrorID	ErrorInfo	Description	Remedy
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8614		**Invalid "Jerk" value**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#001F	Value is greater than the maximum jerk	
	16#0020	Value is less than the minimum jerk	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8615		**Value for "Unit of measurement" is invalid**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8616		**Address of homing switch is invalid (passive homing as of V4)**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8617		**Value of variable <axis name>.Sensor.Sensor[1].ActiveHoming.Mode is invalid**	
	16#0011	The selection value is invalid (Valid value: 2 = Homing via digital input)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8618		**Value of variable <axis name>.Sensor.Sensor[1].PassiveHoming.Mode is invalid**	
	16#0011	The selection value is invalid (Valid value: 2 = Homing via digital input)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8619		**Value of variable <axis name>.Actor.Type is invalid**	

ErrorID	ErrorInfo	Description	Remedy
	16#0011	The selection value is invalid (Valid value: 2 = Connection via pulse interface)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861A		**Value for "Permitted direction of rotation" is invalid**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#002D	"Both directions" not allowed when direction output is deactivated	
16#861B		**Faulty load gear factors**	
	16#0031	Valid is invalid.	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#861C		**Illegal combination of data for homing with incremental encoder**	
	16#0031	Valid is invalid.	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861D		**The set encoder mounting type is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].MountingMode**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861E		**The configuration of the measuring wheel circumference of the encoder is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.DistancePerRevolution**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861F		**The configuration for the resolution of the linear encoder is faulty. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.Resolution**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8620		**The set fine resolution for Gn_XIST1 is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.FineResolutionXist1**	

ErrorID	ErrorInfo	Description	Remedy
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8621		The set fine resolution for Gn_XIST1 in <Axis name>.Sensor.Sensor[1].Parameter.FineResolutionXist1 is not consistent with the setting in PROFIdrive parameter P979	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8622		Value in <Axis name>.Actor.Interface.AddressIn or <Axis name>.Actor.Interface.AddressOut is invalid	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8623		The value set in the variable <axis name>.Sensor.Sensor[1].Type is invalid.	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8624		The set encoder system is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].System	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8625		Parameter of position monitoring is faulty. Invalid value in <Axis name>.PositioningMonitoring.MinDwellTime	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8626		Parameter of position monitoring is faulty. Invalid value in <Axis name>.PositioningMonitoring.Window	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8627		The configuration of the PROFIdrive interface of the actual value is faulty. Invalid value in <Axis name>.Sensor.Sensor[1].Interface.AddressIn or <Axis name>.Sensor.Sensor[1].Interface.AddressOut	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8628		Faulty controller factors	

ErrorID	ErrorInfo	Description	Remedy
	16#0030	Value has an incorrect number format or is outside the valid number range	The value for the gain or the precontrol of the control loop is faulty. • Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary (<Axis name>.PositionControl.Kv, <Axis name>.PositionControl.Kpc)
16#8629		Limit for standstill signal is faulty. Invalid value in <Axis name>.StandStillSignal.VelocityThreshold	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862A		Parameter of position monitoring is faulty. Invalid value in <Axis name>.PositioningMonitoring.Toleran-ceTime	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862B		Inconsistent PROFIBUS parameterization; the sum of Ti and To is greater than one DP cycle	
	16#0030	Value has an incorrect number format or is outside the valid number range	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#862C		Parameter of standstill monitoring is faulty. Invalid value in <Axis name>.StandStillSignal.MinDwellTime	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862D		Parameter of following error monitoring is faulty. Invalid value in <Axis name>.FollowingError.MinValue	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862E		Value in <Axis name>.Modulo.Length is invalid	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862F		Value in <Axis name>.Modulo.StartValue is invalid	

ErrorID	ErrorInfo	Description	Remedy
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8630		Value in <Axis name>.Actor.DriveParameter.ReferenceSpeed is invalid	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8631		The set fine resolution for Gn_XIST2 is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.FineResolutionXist2	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8632		The number of determinable encoder revolutions is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.DeterminableRevolutions	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8634		Parameter of the following error monitoring is faulty. Invalid value in <Axis name>.FollowingError.MaxValue	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8635		Parameter of the following error monitoring is faulty. Invalid value in <Axis name>.FollowingError.MinVelocity	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8636		Controller factor is incorrect. Invalid value of the precontrol factor <Axis name>.PositionControl.Kpc	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8637		Value in <Axis name>.Sensor.Sensor[1].Interface.Type is invalid	

ErrorID	ErrorInfo	Description	Remedy
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8638		Value in <Axis name>.Sensor.Sensor[1].Interface.HSC is invalid	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8639		Error at the drive	
	16#0049	Configuration error at device	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#004A	The technology needs a smaller servo clock.	Internal system error. Check the project for consistency and reload it into the controller.
	16#004B	Device driver not initialized during ramp-up.	To enable a technology object, the actuator driver must be initialized. Execute the command again later.
16#863A		Communication to the drive is faulty	
	16#004C	Configuration error at device	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004D	The device driver needs a smaller servo clock.	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004E	Error in internal communication with the device	Check the project for consistency and reload it into the controller.
16#863B		Error at encoder	
	16#0049	Configuration error at device	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#004A	The technology needs a smaller servo clock.	Internal system error. Check the project for consistency and reload it into the controller.
	16#004B	Device driver not initialized during ramp-up.	To enable a technology object, the actuator driver must be initialized. Execute the command again later.
16#863C		Communication with encoder is faulty	
	16#004C	Configuration error at device	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004D	The device driver needs a smaller servo clock.	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004E	Error in internal communication with the device	Check the project for consistency and reload it into the controller.

ErrorID	ErrorInfo	Description	Remedy
16#863D		**Communication to the device (drive or encoder) is faulty**	
	16#004F	The requested logical address is invalid.	• Connect a suitable device. • Check the device (I/Os). • Check the topological configuration in HW Config. • Compare the configuration of HW Config and the technology object.
	16#0050	The requested logical output address is invalid.	
	16#0051	The requested logical output address is invalid.	
16#863E		**Value of variable "ControlPanel.Input.TimeOut" is invalid (axis control panel)**	
	16#0030	Value has an incorrect number format or is outside the valid number range	Correct the value in the variables of the technology object <Axis name>.ControlPanel.Input.TimeOut. The value is specified in milliseconds (ms).
16#863F		**Value in <Axis name>.Actor.DriveParameter.MaxSpeed is invalid**	
	16#0030	Value has an incorrect number format or is outside the valid number range	Correct the reference value in the drive and in the configuration of the technology object to Actuator.MaxSpeed/2. With analog drive connection, correct the reference value in the drive and in the configuration of the technology object to Actuator.MaxSpeed/1.17.
16#8640		**Error with automatic transfer of drive parameters in the device**	

ErrorID	ErrorInfo	Description	Remedy
	16#0030	Value has an incorrect number format or is outside the valid number range	Correct the value.
	16#0059	The device is not assigned to any SINAMICS drive unit or does not support the services required for the adaptation.	Disable automatic transfer of the parameters in the device. Complete the required parameters in the axis configuration and reload the axis configuration in the device.
	16#005A	Automatic transfer of parameters canceled due to insufficient resources.	
	16#005B	The parameters can only be automatically transferred when the device is connected directly to an I/O area.	
	16#005C	Maximum speed/velocity (p1082): Either the parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	Check the causes. Disable automatic transfer of the parameters in the device if you cannot eliminate the causes. Complete the required parameters in the axis configuration and reload the axis configuration in the device.
	16#005D	Maximum torque/force (p1520): Either the parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	
	16#005E	Maximum torque/force (p1521): Either the parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	
	16#005F	Fine resolution torque/force limiting (p1544): Either the parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	
	16#0060	Nominal speed/velocity (p2000): Either the parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	
	16#0061	Nominal torque/force (p2003): Either the parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	
16#8641		Error with automatic transfer of encoder parameters in the device	

ErrorID	ErrorInfo	Description	Remedy
	16#0030	Value has an incorrect number format or is outside the valid number range	Correct the value.
	16#0059	The device is not assigned to any SINAMICS drive unit or does not support the services required for the adaptation.	Disable automatic transfer of the parameters in the device. Complete the required parameters in the axis configuration and reload the axis configuration in the device.
	16#005A	Automatic transfer of parameters canceled due to insufficient resources.	
	16#005B	The parameters can only be automatically transferred when the device is connected directly to an I/O area.	
	16#0062	Encoder system (r0979[1/11].0): Either a parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	Check the causes. Disable automatic transfer of the parameters in the device if you cannot eliminate the causes. Complete the required parameters in the axis configuration and reload the axis configuration in the device.
	16#0063	Encoder resolution (r0979[2/12]): Either a parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	
	16#0064	Encoder fine resolution Gx_XIST1 (r0979[3/13]): Either a parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	
	16#0065	Encoder fine resolution Gx_XIST2 (r0979[4/14]): Either a parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	
	16#0066	Number of determinable encoder revolutions (r0979[5/15]): Either a parameter does not exist or its value cannot be read or is outside the permitted limits. Reading of parameters has been aborted due to an error signaled by the hardware.	
16#8646		**The value in <Axis name>.Sensor.Interface.Number is invalid.**	
	16#0030	Value has an incorrect number format or is outside the valid number range.	Correct the value and load configuration in the device.
16#8647		**Simulation is not supported for PTO axes**	
	16#0001	-	Disable simulation

Configuration error of the command table

ErrorID	ErrorInfo	Description	Remedy
16#8700		**Value for "Command type" in the command table is invalid**	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
16#8701		**Value for "Position / travel path" in the command table is invalid**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1.0E12)	
	16#0006	Value is outside the number range (less than -1.0E12)	
16#8702		**Value for "Velocity" in the command table is invalid**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/ stop velocity	
16#8703		**Value for "Duration" in the command table is invalid**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0021	Value is greater than 64800 s	
	16#0022	Value is less than 0.001 s	
16#8704		**Value for "Next step" in the command table is invalid**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0023	The command transition is not permitted for this command	

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		**Internal error**	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostics buffer entries

Tags of the positioning axis technology object as of V6 (S7-1200)

Legend (S7-1200)

Tag	Name of the tag	
Data type	Data type of the tag	
Values	Start value (Value range of the tag - minimum value to maximum value) If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".	
Access	Access to the tag in the user program	
	OPR	The tag can be read by the Openness application.
	OPRW	The tag can be read and written by the Openness application.
	R	The tag can be read in the user program and in the HMI.
	RCCP	The tag can be read in the user program and in the HMI and is updated at each cycle control point.
	RP	The variable can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding variables is determined at the start of the command.
	RW	The tag can be read and written in the user program and in the HMI. The variable can be written with Motion Control instruction "MC_WriteParam".
	WD BL	For drive connection via PROFIdrive/analog output: The tag can be written to the start value in the load memory with the extended instruction "WRIT_DBL".
	WP	Independent of the drive connection: If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	WP_PD	For drive connection via PROFIdrive/analog output: If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	WP_PTO	For drive connection via PTO: If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	-	The variable cannot be used in the user program.
W	Effectiveness of changes in the technology data block	
	1	For drive connection via PTO: When axis is activated, disabled, or enabled
	2	For drive connection via PTO: When axis is enabled
	5	For drive connection via PTO: The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable, or active MC_Home command is started (Mode = 3)
	6	For drive connection via PTO: When a MC_MoveJog command is stopped
	7	For drive connection via PTO: When a passive homing command is started
	8	For drive connection via PTO: When an active homing command is started
	9	With the restart of the technology object
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
Description	Description of the tag	

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Tags for position values and velocity values as of V6 (S7-1200)

The tag structure contains the setpoint and actual values of the position and the velocity of the axis.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
Position	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP, OPR	-	Setpoint position of the axis (indicated in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis. Name in Openness: Position
Velocity	REAL	0.0	RCCP, RP, OPR	-	Velocity setpoint of the axis (indicated in the configured unit of measurement) Name in Openness: Velocity
ActualPosition	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP, OPR	-	Actual position of the axis (indicated in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis. Name in Openness: ActualPosition
ActualVelocity	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP, OPR	-	Actual velocity of the axis (indicated in the configured unit of measurement) Name in Openness: ActualVelocity

See also

Motion status (Page 1449)

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

Simulation tags as of V6 (S7-1200)

The tag structure <axis name>.Simulation.Mode contains the simulation mode.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description	
Simulation.						
Mode	UDINT	0 (0 to 1)	R WDBL OPRW	- 2, 9	Simulation mode Name in Openness: Simulation.Mode	
					0	No simulation, normal operation
					1	Simulation mode In simulation mode, you can simulate axes without a real drive in the CPU.

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

Actuator tags as of V6 (S7-1200)

The tag structure <axis name>.Actor.<tag name> contains the drive parameters.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
Actor.	STRUCT				TO_Struct_Actor

Tag	Data type	Values	Access	W	Description	
Type	DINT	2 (0 to 2)	R OPRW	-	Name in Openness: Actor.Type	
					Positioning axis technology object as of V5:	
					0	The drive is connected via an analog output. All movements of the axis are position-controlled.
					1	The drive is connected via PROFIdrive telegrams. All movements of the axis are position-controlled.
					2	The drive is connected via a pulse interface.
					Positioning axis technology object V4:	
					2	The drive is connected via a pulse interface.
InverseDirection	BOOL	FALSE	R WP_PTO OPRW	- 2	Name in Openness: Actor.InverseDirection	
					FALSE	The direction is not inverted.
					TRUE	The direction is inverted.
DirectionMode	INT	0 (0 to 2)	R WP_PTO OPRW	- 2	Permitted direction of rotation Name in Openness: Actor.DirectionMode	
					0	Both directions
					1	Positive direction
					2	Negative direction
DataAdaption	DINT	0 (0 to 1)	R OPRW	-	Name in Openness: Actor.DataAdaption	
					0	Automatic transfer of drive parameters in the device is disabled.
					1	Automatic transfer of drive parameters in the device is enabled.
Interface.	STRUCT				TO_Struct_ActorInterface	
AddressIn.	VREF	-	-	-	Input address for the PROFIdrive telegram (internal parameter)	
AREA	BYTE	-	OPR	-	Name in Openness: Actor.Interface.AddressIn.AREA	
DB_NUMBER	UINT	-	OPR	-	Name in Openness: Actor.Interface.AddressIn.DB_NUMBER	
OFFSET	UDINT	-	OPR	-	Name in Openness: Actor.Interface.AddressIn.OFFSET	
RID	DWORD	-	OPR	-	Name in Openness: Actor.Interface.AddressIn.RID	
ProfiDriveIn*	STRING	-	OPRW	-	Name in Openness: _Actor.Interface.ProfiDriveIn Valid input address, which is the part of a telegram Valid tag name	
AddressOut.	VREF	-	-	-	Output address for the PROFIdrive telegram (internal parameter)	

Tag	Data type	Values	Access	W	Description
AREA	BYTE	-	OPR	-	Name in Openness: Actor.Interface.AddressOut.AREA
DB_NUMBER	UINT	-	OPR	-	Name in Openness: Actor.Interface.AddressOut.DB_NUMBER
OFFSET	UDINT	-	OPR	-	Name in Openness: Actor.Interface.AddressOut.OFFSET
RID	DWORD	-	OPR	-	Name in Openness: Actor.Interface.AddressOut.RID
ProfiDriveOut*	STRING	-	OPRW	-	Name in Openness: _Actor.Interface.ProfiDriveOut Valid output address, which is the part of a telegram Valid tag name
DataBlock*	STRING	-	OPRW	-	Name in Openness: _Actor.Interface.DataBlock Valid data block address:
Analog*	STRING	-	OPRW	-	Name in Openness: _Actor.Interface.Analog Valid analog output, valid data block address, valid tag name
DataConnection*	INT	0 (0 to 1)	OPRW	-	Name in Openness: _Actor.Interface.DataConnection 0 Drive 1 Data block
EnableDriveOutput	VREF	-	-	-	Enable output (internal parameter)
AREA	BYTE	-	OPR	-	Name in Openness: Actor.Interface.DriveReadyOutput.AREA
DB_NUMBER	UINT	-	OPR	-	Name in Openness: Actor.Interface.DriveReadyOutput.DB_NUMBER
OFFSET	UDINT	-	OPR	-	Name in Openness: Actor.Interface.DriveReadyOutput.OFFSET
RID	DWORD	-	OPR	-	Name in Openness: Actor.Interface.DriveReadyOutput.RID
EnableDriveOutput*	STRING	-	OPRW	-	Name in Openness: _Actor.Interface.EnableDriveOutput Valid input, valid output, valid memory address, valid tag name
DriveReadyInput	VREF	-	-	-	Ready input (internal parameter)
AREA	BYTE	-	OPR	-	Name in Openness: Actor.Interface.DriveReadyInput.AREA
DB_NUMBER	UINT	-	OPR	-	Name in Openness: Actor.Interface.DriveReadyInput.DB_NUMBER
OFFSET	UDINT	-	OPR	-	Name in Openness: Actor.Interface.DriveReadyInput.OFFSET
RID	DWORD	-	OPR	-	Name in Openness: Actor.Interface.DriveReadyInput.RID

Tag	Data type	Values	Access	W	Description
DriveReadyInput*	STRING	-	OPRW	-	Name in Openness: _Actor.Interface.DriveReadyInput Valid input, valid output, valid memory address, valid tag name
PTO	DWORD	0	OPR	-	Pulse output (internal parameter)
PTO*	STRING	-	OPRW	-	Name in Openness: _Actor.Interface.PTO Name of pulse generators as in the hardware configuration
PTO_OutputA*	STRING	-	OPRW	-	Name in Openness: _Actor.Interface.PTO_OutputA Valid input address, valid tag name Only onboard CPU or Dignalboard addresses are accepted.
PTO_OutputBEnable*	BOOL		OPRW	-	Name in Openness: _Actor.Interface.PTO_OutputBEnable Only possible with PTO_SignalType = 2 FALSE: Output B is disabled. TRUE: Output B is enabled.
PTO_OutputB*	STRING	-	OPRW	-	Name in Openness: _Actor.Interface.PTO_OutputB Valid input address, valid tag name Only onboard CPU or Dignalboard addresses are accepted.
PTO_SignalType*	INT	(2 to 5)	OPRW	-	Name in Openness: _Actor.Interface.PTO_SignalType 2: Pulse A and direction B 3: Clock up A and clock down B 4: A/B phase-shifted 5: A/B phase-shifted - quadruple
DriveParameter.	STRUCT				TO_Struct_ActorDriveParameter
ReferenceSpeed	REAL	3000.0	R OPRW	-	Reference value (100 %) for the speed setpoint (NSET) of the drive The setpoint speed is transmitted in the PROFIdrive telegram as a standardized value from -200% to 200% of the "ReferenceSpeed". For setpoint specification via an analog output, the analog output can be operated in the range from -117 % to 117 %, provided the drive permits this. Name in Openness: Actor.DriveParameter.ReferenceSpeed
MaxSpeed	REAL	3000.0	R OPRW	-	Maximum value for the speed setpoint of the drive (NSET) (PROFIdrive: MaxSpeed ≤ 2 × ReferenceSpeed Analog setpoint: MaxSpeed ≤ 1.17 × ReferenceSpeed)

Tag			Data type	Values	Access	W	Description
							Name in Openness: Actor.DriveParameter.MaxSpeed
		PulsesPerDri-veRevolution	DINT	1000 (1 to 2147483648)	R WP_PTO OPRW	- 2	Pulses per motor revolution Name in Openness: Actor.DriveParameter.PulsesPerDriveRevolution

*) Available in Openness

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

Sensor[1] tags as of V6 (S7-1200)

The tag structure <axis name>.Sensor[1].<tag name> contains the encoder parameters.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
Sensor[1].	STRUCT				ARRAY[1..1] TO_Struct_Sensor

Tag	Data type	Values	Access	W	Description	
Type	DINT	0 (0 to 1)	R, OPRW	-	Encoder type (internal parameter) Name in Openness: Sensor[1].Type	
					0	Incremental
					1	Absolute
InverseDirection	BOOL	FALSE	R, OPRW	-	Inversion of the actual value Name in Openness: Sensor[1].InverseDirection	
					FALSE	Actual value is not inverted
					TRUE	Actual value is inverted
System	DINT	1 (0 to 1)	R, OPRW	-	Encoder system Name in Openness: Sensor[1].System	
					0	Linear encoder
					1	Rotary encoder
MountingMode	DINT	0 (0 to 2)	R, OPRW	-	Mounting type of encoder Name in Openness: Sensor[1].MountingMode	
					0	On the motor shaft
					2	External measuring system
DataAdaption	DINT	0 (0 to 1)	R, OPRW	-	Name in Openness: Sensor[1].DataAdaption	
					0	Automatic transfer of encoder parameters in the device is disabled.
					1	Automatic transfer of encoder parameters in the device is enabled.
Interface.	STRUCT				TO_Struct_SensorInterface	
Type	DINT	4 (0 to 4)	OPR	-	Encoder connection (internal parameter) Name in Openness: Sensor[1].Interface.Type	
					0	PROFIdrive encoder on PROFINET
					1	Encoder on technology module (TM)
					2	Encoder on the drive
					4	Encoder on high-speed counter
AddressIn.	VREF	-	-	-	Input address for the PROFIdrive telegram (internal parameter)	
AREA	BYTE	-	OPR	-	Internal parameter Name in Openness: Sensor[1].Interface.AddressIn.AREA	
DB_NUMBER	UINT	-	OPR	-	Internal parameter Name in Openness: Sensor[1].Interface.AddressIn.DB_NUMBER	
OFFSET	UDINT	-	OPR	-	Internal parameter Name in Openness: Sensor[1].Interface.AddressIn.OFFSET	

Tag	Data type	Values	Access	W	Description
RID	DWORD	-	OPR	-	Internal parameter Name in Openness: Sensor[1].Interface.AddressIn.RID
ProfiDriveIn*	STRING	-	OPRW	-	Name in Openness: _Sensor[1].Interface.ProfiDriveIn Valid input address, which is the part of a telegram Valid tag name
AddressOut.	VREF	-	-	-	Output address for the PROFIdrive telegram (internal parameter)
AREA	BYTE	-	OPR	-	Internal parameter Name in Openness: Sensor[1].Interface.AddressOut.AREA
DB_NUMBER	UINT	-	OPR	-	Internal parameter Name in Openness: Sensor[1].Interface.AddressOut.DB_NUMBER
OFFSET	UDINT	-	OPR	-	Internal parameter Name in Openness: Sensor[1].Interface.AddressOut.OFFSET
RID	DWORD	-	OPR	-	Internal parameter Name in Openness: Sensor[1].Interface.AddressOut.RID
ProfiDriveOut*	STRING	-	OPRW	-	Name in Openness: _Sensor[1].Interface.ProfiDriveOut Valid output address, which is the part of a telegram Valid tag name
DataBlock*	STRING	-	OPRW	-	Name in Openness: _Sensor[1].Interface.DataBlock Valid data block address:
DataConnec-tion*	UDINT	(0 to 1)	OPRW	-	Name in Openness: _Sensor[1].Interface.DataConnection 0 Encoder 1 Data block
EncoderCon-nection*	INT	(4 to 7)	OPRW	-	Name in Openness: _Sensor[1].Interface.EncoderConnection 4 Encoder on high-speed counter (HSC) 7 Encoder on PROFIBUS/PROFINET
Number	UDINT	1	OPRW	-	Encoder number Name in Openness: Sensor[1].Interface.Number
HSC	DWORD	0	OPR	-	High-speed counter to which the encoder transfers the actual value (internal parameter)
HSC*	STRING	-	OPRW	-	Name in Openness: _Sensor[1].Interface.HSC Name of fast counters from the hardware configuration
HSC_Operating-Mode*	INT	(1 to 3)	OPRW	-	Name in Openness: _Sensor[1].Interface.HSC_OperatingMode

Tag	Data type	Values	Access	W	Description	
					1	Two-phase
					2	A/B counter
					3	A/B counter quadruple
HSC_InputA*	STRING	-	OPRW	-	Name in Openness: _Sensor[1].Interface.HSC_InputA Valid input address, valid tag name	
HSC.InputB*	STRING	-	OPRW	-	Name in Openness: _Sensor[1].Interface.HSC.InputB Valid input address, valid tag name	
Parameter.	STRUCT				TO_Struct_SensorParameter	
Resolution	REAL	0.001 (-1.0E12 to 1.0E12)	R, OPRW	-	Resolution of a linear encoder (offset between two encoder pulses) Name in Openness: Sensor[1].Parameter.Resolution	
StepsPerRevolution	UDINT	2048 (1 to 8388608)	R, OPRW	-	Increments per rotary encoder revolution Name in Openness: Sensor[1].Parameter.StepsPerRevolution	
FineResolutionXist1	UDINT	11 (0 to 31)	R, OPRW	-	Number of bits for fine resolution Gn_XIST1 (cyclic actual encoder value) Name in Openness: Sensor[1].Parameter.FineResolutionXist1	
FineResolutionXist2	UDINT	9 (0 to 31)	R, OPRW	-	Number of bits for fine resolution Gn_XIST2 (absolute value of the encoder) Name in Openness: Sensor[1].Parameter.FineResolutionXist2	
DeterminableRevolutions	UDINT	1 (0 to 8388608)	R, OPRW	-	Number of differentiable encoder revolutions for a multiturn absolute encoder Name in Openness: Sensor[1].Parameter.DeterminableRevolutions	
					0	Incremental encoder
					1	Single return absolute encoder
DistancePerRevolution	REAL	100.0 (0.0 to 1.0E12)	R, OPRW	-	Load distance per revolution of an externally mounted encoder Name in Openness: Sensor[1].Parameter.DistancePerRevolution	
ActiveHoming.	STRUCT				TO_Struct_SensorActiveHoming	
Mode	DINT	2 (0 to 2)	R, WP_PTO, OPRW	- 2	Active homing mode Name in Openness: Sensor[1].ActiveHoming.Mode	
					Positioning axis technology object as of V5:	
					0	Zero mark via PROFIdrive telegram (not PTO)
					1	Zero mark via PROFIdrive telegram and proximity switch (not PTO)
					2	Homing via digital input
					Positioning axis technology object V4:	
					2	Homing via digital input

Tag	Data type	Values	Access	W	Description
SideInput	BOOL	FALSE	RW, WP, OPRW	1, 8, 10	End of the homing switch to which the axis is homed during active homing Name in Openness: Sensor[1].ActiveHoming.SideInput FALSE: Bottom side TRUE: Top side
DigitalInputAd-dress.	VREF	-	-	-	Symbolic input address of the homing switch (internal parameter)
AREA	BYTE	-	OPR	-	Internal parameter Name in Openness: Sensor[1].ActiveHoming.DigitalInputAddress.AREA
DB_NUMBER	UINT	-	OPR	-	Internal parameter Name in Openness: Sensor[1].ActiveHoming.DigitalInputAd-dress.DB_NUMBER
OFFSET	UDINT	-	OPR	-	Internal parameter Name in Openness: Sensor[1].ActiveHoming.DigitalInputAddress.OFFSET
RID	DWORD	-	OPR	-	Internal parameter Name in Openness: Sensor[1].ActiveHoming.DigitalInputAddress.RID
DigitalInput*	STRING	-	OPRW	-	Name in Openness: _Sensor[1].ActiveHoming.DigitalInput Valid input address, valid tag name
HomePositio-nOffset	REAL	0.0 (-1.0E12 to 1.0E12)	RW, WP, OPRW	1, 8, 10	Home position offset (active homing) (indicated in the configured unit of measurement) Name in Openness: Sensor[1].ActiveHoming.HomePositionOffset
SwitchLevel	BOOL	TRUE	RW, WP, OPRW	1, 8, 10	Selection of signal level that is present at the CPU input when the homing switch is reached Name in Openness: Sensor[1].ActiveHoming.SwitchLevel FALSE: Low level (Low active) TRUE: High level (high-enabled)
PassiveHoming.	STRUCT				TO_Struct_SensorPassiveHoming
DigitalInputAd-dress.	VREF	-	-	-	Symbolic input address of the homing switch (internal parameter)
AREA	BYTE	-	OPR	-	Internal parameter Name in Openness: Sensor[1].PassiveHoming.DigitalInputAddress.AREA
DB_NUMBER	UINT	-	OPR	-	Internal parameter Name in Openness: Sensor[1].PassiveHoming.DigitalInputAd-dress.DB_NUMBER
OFFSET	UDINT	-	OPR	-	Internal parameter

Tag	Data type	Values	Access	W	Description
					Name in Openness: Sensor[1].PassiveHoming.DigitalInputAddress.OFFSET
RID	DWORD	-	OPR	-	Internal parameter Name in Openness: Sensor[1].PassiveHoming.DigitalInputAddress.RID
Mode	DINT	2 (0 to 2)	R WP_PTO OPRW	- 2	Passive homing mode Name in Openness: Sensor[1].PassiveHoming.Mode
					Positioning axis technology object as of V5:
					0 — Zero mark via PROFIdrive telegram (not PTO)
					1 — Zero mark via PROFIdrive telegram and proximity switch (not PTO)
					2 — Homing via digital input
					Positioning axis technology object V4:
					2 — Homing via digital input
SideInput	BOOL	FALSE	RW, WP, OPRW	1, 7, 10	End of the homing switch to which the axis is homed during passive homing Name in Openness: Sensor[1].PassiveHoming.SideInput
					FALSE — Bottom side
					TRUE — Top side
DigitalInputAddress.	VREF	-	-	-	Symbolic input address of the homing switch (internal parameter)
AREA	BYTE	-	OPR	-	Internal parameter Name in Openness: Sensor[1].PassiveHoming.DigitalInputAddress.AREA
DB_NUMBER	UINT	-	OPR	-	Internal parameter Name in Openness: Sensor[1].PassiveHoming.DigitalInputAddress.DB_NUMBER
OFFSET	UDINT	-	OPR	-	Internal parameter Name in Openness: Sensor[1].PassiveHoming.DigitalInputAddress.OFFSET
RID	DWORD	-	OPR	-	Internal parameter Name in Openness: Sensor[1].PassiveHoming.DigitalInputAddress.RID
DigitalInput*	STRING		OPRW	-	Name in Openness: _Sensor[1].PassiveHoming.DigitalInput Valid input address, valid tag name
SwitchLevel	BOOL	TRUE	RW, WP, OPRW	1, 7, 10	Selection of level that is present at the CPU input when the homing switch is reached Name in Openness: Sensor[1].PassiveHoming.SwitchLevel
					FALSE — Low level (Low active)

Tag	Data type	Values	Access	W	Description	
					TRUE	High level (high-enabled)

*) Available in Openness

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

Units tag as of V6 (S7-1200)

The tag structure <axis name>.Units.LengthUnit contains the configured units of measurement of the parameters.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description	
Units.	STRUCT				TO_Struct_Units	
LengthUnit	INT	1013 (-32768 to 32767)	R WP_PTO OPRW	- 2	Configured unit of measurement of the parameter Name in Openness: Units.LengthUnit	
					-1	Pulses
					1005	°
					1010	m
					1013	mm
					1018	ft
					1019	in

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

Mechanics tag as of V6 (S7-1200)

The tag structure <axis name>.Mechanics.LeadScrew contains the distance traveled per motor revolution.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
Mechanics.	STRUCT				TO_Struct_Mechanics
LeadScrew	REAL	10.0 (-1.0E12 to 1.0E12)	R, WP_PTO, OPRW	-	Distance per revolution (indicated in the configured unit of measurement) Name in Openness: Mechanics.LeadScrew

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

Modulo tags as of V6 (S7-1200)

The tag structure <axis name>.Modulo.<tag name> contains the modulo settings.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
Modulo.	STRUCT				TO_Struct_Modulo
Enable	BOOL	FALSE	R, OPRW	-	Name in Openness: Modulo.Enable
					FALSE: Modulo conversion disabled
					TRUE: Modulo conversion enabled
					When modulo conversion is enabled, a check is made for modulo length > 0.0
Length	REAL	360.0 (0.001 to 1.0E12)	R, OPRW	-	Modulo length Name in Openness: Modulo.Length
StartValue	REAL	0.0 (-1.0E12 to 1.0E12)	R, OPRW	-	Modulo start value Name in Openness: Modulo.StartValue

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

DynamicLimits tags as of V6 (S7-1200)

The tag structure <axis name>.DynamicLimits.<tag name> contains the configuration of the dynamic limits.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
DynamicLimits.	STRUCT				TO_Struct_DynamicLimits
MaxVelocity	REAL	250.0	R WP_PTO OPRW	- 2	Maximum velocity of the axis (indicated in the configured unit of measurement) Name in Openness: DynamicLimits.MaxVelocity
MinVelocity	REAL	10.0	R WP_PTO OPRW	- 2	Start/stop velocity of the axis (indicated in the configured unit of measurement) Name in Openness: DynamicLimits.MinVelocity

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

DynamicDefaults tags as of V6 (S7-1200)

The tag structure <axis name>.DynamicDefaults.<tag name> contains the configuration of the dynamic defaults.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
DynamicDefaults.	STRUCT				TO_Struct_DynamicDefaults
Acceleration	REAL	48.0 (0.0 to 1.0E12)	RW, WP, OPRW	5, 6, 10	Default setting of the acceleration of the axis (indicated in the configured unit of measurement) Name in Openness: DynamicDefaults.Acceleration
Deceleration	REAL	48.0 (0.0 to 1.0E12)	RW, WP, OPRW	5, 6, 10	Default decelaration of the axis (indicated in the configured unit of measurement) Name in Openness: DynamicDefaults.Deceleration
Jerk	REAL	192.0 (0.0 to 1.0E12)	RW, WP, OPRW	5, 10	Setting the jerk default during acceleration and deceleration ramp of the axis (indicated in the configured unit of measurement) The jerk is activated if the configured jerk is greater than 0.00004 mm/s². Name in Openness: DynamicDefaults.Jerk
EmergencyDeceleration	REAL	120.0 (0.0 to 1.0E12)	RW, WP, OPRW	1, 5, 6, 10	Emergency stop deceleration of the axis (indicated in the configured unit of measurement) Name in Openness: DynamicDefaults.EmergencyDeceleration

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

PositionLimits_SW variables as of V6 (S7-1200)

The tag structure <axis name>.PositionLimits_SW.<tag name> contains the configuration for position monitoring with software limit switches. Software limit switches are used to limit the operating range of a positioning axis.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
PositionLimits_SW.	STRUCT				TO_Struct_PositionLimitsSW
Active	BOOL	FALSE	RW, WP, OPRW	1, 5, 6, 10	Name in Openness: PositionLimits_SW.Active FALSE: The software limit switches are deactivated. TRUE: The software limit switches are activated.
MinPosition	REAL	-10000.0 (-1.0E12 to 1.0E12)	RW, WP, OPRW	1, 5, 6, 10	Position of the software low limit switches (specification in the configured measurement unit) Name in Openness: PositionLimits_SW.MinPosition
MaxPosition	REAL	10000.0 (-1.0E12 to 1.0E12)	RW, WP, OPRW	1, 5, 6, 10	Position of the software high limit switch (indicated in the configured unit of measurement) Name in Openness: PositionLimits_SW.MaxPosition

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

PositionLimits_HW variables as of V6 (S7-1200)

The tag structure <axis name>.PositionLimits_HW.<tag name> contains the configuration for position monitoring with hardware limit switches. Hardware limit switches are used to limit the traversing range of a positioning axis.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
PositionLimits_HW.	STRUCT				TO_Struct_PositionLimitsHW

Tag	Data type	Values	Access	W	Description
Active	BOOL	FALSE	RW, WP, OPRW	1, 5, 6, 10	Name in Openness: PositionLimits_HW.Active FALSE: The hardware limit switches are deactivated. TRUE: The hardware limit switches are activated.
MinSwitchLevel	BOOL	FALSE	RW WP_PTO	- 2	Selection of signal level that is present at the CPU input when the hardware low limit switch is reached Name in Openness: PositionLimits_HW.MinSwitchLevel FALSE: Low level (Low active) TRUE: High level (high-enabled)
MinSwitchAddress.	VREF	-	-	-	Symbolic input address of the hardware low limit switch (internal parameter)
AREA	BYTE	-	OPR	-	Name in Openness: PositionLimits_HW.MinSwitchAddress.AREA
DB_NUMBER	USH-ORT	-	OPR	-	Name in Openness: PositionLimits_HW.MinSwitchAddress.DB_NUMBER
OFFSET	UINT	-	OPR	-	Name in Openness: PositionLimits_HW.MinSwitchAddress.OFFSET
RID	UINT	-	OPR	-	Name in Openness: PositionLimits_HW.MinSwitchAddress.RID
MinSwitch*	STRING	-	OPRW	-	Name in Openness: _PositionLimits_HW.MinSwitch Valid input address, valid tag name
MaxSwitchLevel	BOOL	FALSE	RW WP_PTO	- 2	Selection of signal level that is present at the CPU input when the hardware high limit switch is reached Name in Openness: PositionLimits_HW.MaxSwitchLevel FALSE: Low level (Low active) TRUE: High level (high-enabled)
MaxSwitchAd-dress.	VREF	-	-	-	Input address of the hardware high limit switch (internal parameter)
AREA	BYTE	-	OPR	-	Name in Openness: PositionLimits_HW.MaxSwitchAddress.AREA
DB_NUMBER	USH-ORT	-	OPR	-	Name in Openness: PositionLimits_HW.MaxSwitchAddress.DB_NUMBER
OFFSET	UINT	-	OPR	-	Name in Openness: PositionLimits_HW.MaxSwitchAddress.OFFSET
RID	UINT	-	OPR	-	Name in Openness: PositionLimits_HW.MaxSwitchAddress.RID
MaxSwitch*	STRING	-	OPRW	-	Name in Openness: _PositionLimits_HW.MaxSwitch Valid input address, valid tag name

*) Available in Openness

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

Homing tags as of V6 (S7-1200)

The tag structure <axis name>.Homing.<tag name> contains the configuration for homing the axis.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description	
Homing.	STRUCT				TO_Struct_Homing	
AutoReversal	BOOL	FALSE	RW, WP, OPRW	1, 8, 10	Name in Openness: Homing.AutoReversal	
					FALSE	Auto reversal at the hardware limit switch is deactivated.
					TRUE	Auto reversal at the hardware limit switch is activated.
ApproachDirection	BOOL	TRUE	RW, WP, OPRW	1, 8, 10	Name in Openness: Homing.ApproachDirection	
					FALSE	Negative approach direction for finding the homing switch and negative homing direction
					TRUE	Positive approach direction to search for reference point switch and positive homing direction
ApproachVelocity	REAL	200.0 (0.0 to 1.0E12)	RW, WP, OPRW	1, 8, 10	Approach velocity of the axis during active homing (indicated in the configured unit of measurement) Name in Openness: Homing.ApproachVelocity	
ReferencingVelocity	REAL	40.0 (0.0 to 1.0E12)	RW, WP, OPRW	1, 8, 10	Homing velocity of the axis during active homing (indicated in the configured unit of measurement) Name in Openness: Homing.ReferencingVelocity	

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

PositionControl tag as of V6 (S7-1200)

The tag structure <axis name>.PositionControl.<tag name> contains the position control settings.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
PositionControl.	STRUCT				TO_Struct_PositionControl
Kv	REAL	10.0 (0.0 to 2147480.0)	R WP_PD OPRW	- 10	Proportional gain of the closed loop position control ("Kv" > 0.0) Name in Openness: PositionControl.Kv
Kpc	REAL	100.0 (0.0 to 150.0)	R WP_PD OPRW	- 10	Velocity precontrol of the position control as a percentage Name in Openness: PositionControl.Kpc

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

FollowingError tags as of V6 (S7-1200)

The tag structure <axis name>.FollowingError.<tag name> contains the configuration of the dynamic following error monitoring.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
FollowingError.	STRUCT				TO_Struct_FollowingError
EnableMonitoring	BOOL	TRUE	R OPRW	-	Name in Openness: FollowingError.EnableMonitoring FALSE: Following error monitoring deactivated TRUE: Following error monitoring enabled
MinValue	REAL	10.0 (0.0 to 1.0E12)	R WP_PD OPRW	- 10	Permissible following error at velocities below the value of "MinVelocity". Name in Openness: FollowingError.MinValue
MaxValue	REAL	100.0 (0.0 to 1.0E12)	R WP_PD OPRW	- 10	Maximum permissible following error, which may be reached at the maximum velocity. Name in Openness: FollowingError.MaxValue
MinVelocity	REAL	10.0 (0.0 to 1.0E12)	R WP_PD OPRW	- 10	"MinValue" is permissible below this velocity and is held constant. Name in Openness: FollowingError.MinVelocity

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

PositionMonitoring tags as of V6 (S7-1200)

The tag structure <axis name>.PositionMonitoring.<tag name> contains the configuration for position monitoring at the end of a positioning motion.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
PositionMonitoring.	STRUCT				TO_Struct_PositionMonitoring
ToleranceTime	REAL	1.0 (0.0 to 1.0E12)	R WP_PD OPRW	- 10	Tolerance time Maximum permitted duration from reaching velocity set-point zero, until entrance into the positioning window. Name in Openness: PositionMonitoring.ToleranceTime
MinDwellTime	REAL	0.1 (0.0 to 1.0E12)	R WP_PD OPRW	- 10	Minimum dwell time in positioning window Name in Openness: PositionMonitoring.MinDwellTime
Window	REAL	1.0 (0.001 to 1.0E12)	R WP_PD OPRW	- 10	Positioning window Name in Openness: PositionMonitoring.Window

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

StandstillSignal tags as of V6 (S7-1200)

The tag structure <axis name>.StandstillSignal.<tag name> contains the configuration for the standstill signal.

Tags

Legend (Page 1488)

Tag		Data type	Values	Access	W	Description
StandstillSignal.		STRUCT				TO_Struct_StandstillSignal
	VelocityThreshold	REAL	5.0 (0.0 to 1.0E12)	R WP_PD OPRW	- 10	Velocity threshold If velocity is below this threshold, then the minimum dwell time begins. Name in Openness: StandStillSignal.VelocityThreshold
	MinDwellTime	REAL	0.01 (0.0 to 1.0E12)	R WP_PD OPRW	- 10	Minimum dwell time Name in Openness: StandStillSignal.MinDwellTime

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

StatusPositioning tags as of V6 (S7-1200)

The tag structure <axis name>.StatusPositioning.<tag name> indicates the status of a positioning motion.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
StatusPositioning.	STRUCT				TO_Struct_StatusPositioning
Distance	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP, OPR	-	Current distance of axis from target position (indicated in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute", "MC_MoveRelative", or the axis control panel. Name in Openness: StatusPositioning.Distance
TargetPosition	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP, OPR	-	Target position of the axis (indicated in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute", "MC_MoveRelative", or the axis control panel. Name in Openness: StatusPositioning.TargetPosition
FollowingError	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP, OPR	-	Current following error of the axis (indicated in the configured unit of measurement) FollowingError = 0.0 for drive connection via PTO (Pulse Train Output). Name in Openness: StatusPositioning.FollowingError

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

StatusDrive tags as of V6 (S7-1200)

The tag structure <axis name>.StatusDrive.<tag name> indicates the status of the drive.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description	
StatusDrive.	STRUCT				TO_Struct_StatusDrive	
InOperation	BOOL	FALSE	RCCP, RP, OPR	-	Operational status of the drive Name in Openness: StatusDrive.InOperation	
					FALSE	Drive not ready. Setpoints will not be executed.
					TRUE	Drive ready. Setpoints can be executed.
CommunicationOK	BOOL	FALSE	RCCP, RP, OPR	-	Cyclic BUS communication between controller and drive Name in Openness: StatusDrive.CommunicationOK	
					FALSE	Communication not established
					TRUE	Communication established
AdaptionState	DINT	0 (0 to 4)	R, OPR	10	Transfer status of the drive Name in Openness: StatusDrive.AdaptionState	
					0	Data not transferred
					1	Data in transfer
					2	Data transferred
					3	Transfer not possible or not selected
					4	Error during data transfer

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

StatusSensor tags as of V6 (S7-1200)

The tag structure <axis name>.StatusSensor[1].<tag name> indicates the status of the measuring system.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description	
StatusSensor[1].	STRUCT				TO_Struct_StatusSensor	
State	DINT	0 (0 to 2)	RCCP, RP, OPR	-	Status of the encoder value Name in Openness: StatusSensor.State	
					0	Invalid
					1	Waiting for valid status
					2	Valid
CommunicationOK	BOOL	FALSE	RCCP, RP, OPR	-	Cyclic BUS communication between controller and encoder Name in Openness: StatusSensor.CommunicationOK	
					FALSE	Communication not established
					TRUE	Communication established
AbsEncoderOffset	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP, OPR	-	Home point offset to the value of an absolute value encoder. The value will be retentively stored in the CPU. Name in Openness: StatusSensor.AbsEncoderOffset	
AdaptionState	DINT	0 (0 to 1)	R, OPR	10	Transfer status of the encoder Name in Openness: StatusSensor.AdaptionState	
					0	Data not transferred
					1	Data in transfer

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

StatusBits tags as of V6 (S7-1200)

The tag structure <axis name>.StatusBits.<tag name> contains the status information of the technology object.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
StatusBits.	STRUCT				TO_Struct_StatusBits

Tag	Data type	Values	Access	W	Description	
Activated	BOOL	FALSE	RCCP, RP, OPR	-	Activation of the axis Name in Openness: StatusBits.Activated	
					FALSE	The axis is not activated.
					TRUE	The axis is activated. It axis is connected to the assigned PTO (Pulse Train Output). The data of the technology data block will be updated cyclically.
Enable	BOOL	FALSE	RCCP, RP, OPR	-	Enable status of the axis Name in Openness: StatusBits.Enable	
					FALSE	The axis is not enabled.
					TRUE	The axis is enabled and ready to accept Motion Control commands.
AxisSimulation	BOOL	FALSE	RCCP, RP, OPR	-	Name in Openness: StatusBits.AxisSimulation	
					FALSE	The simulation is disabled.
					TRUE	The simulation is enabled.
NonPositionControlled	BOOL	FALSE	RCCP, RP, OPR	-	Name in Openness: StatusBits.NonPositionControlled	
					FALSE	The axis is in position-controlled mode.
					TRUE	The axis is not in position-controlled mode.
HomingDone	BOOL	FALSE	RCCP, RP, OPR	-	Homing status of the axis Name in Openness: StatusBits.HomingDone	
					FALSE	The axis is not homed.
					TRUE	The axis is homed and is capable of executing absolute positioning commands.
					The axis does not have to be homed for relative positioning. During active homing, the status is FALSE. The status remains TRUE during passive homing if the axis was already homed beforehand.	
Done	BOOL	FALSE	RCCP, RP, OPR	-	Command execution on the axis Name in Openness: StatusBits.Done	
					FALSE	A Motion Control command is active on the axis.
					TRUE	A Motion Control command is not active on the axis.
Error	BOOL	FALSE	RCCP, RP, OPR	-	Error status on the axis Name in Openness: StatusBits.Error	
					FALSE	No error is active on the axis.
					TRUE	An error has occurred on the axis.

Tag	Data type	Values	Access	W	Description	
					Additional information about the error is available in automatic mode at the "ErrorID" and "ErrorInfo" parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error.	
Standstill	BOOL	FALSE	RCCP, RP, OPR	-	Standstill of the axis Name in Openness: StatusBits.Standstill	
					FALSE	The axis is in motion.
					TRUE	The axis is at a standstill.
PositioningCommand	BOOL	FALSE	RCCP, RP, OPR	-	Execution of a positioning command Name in Openness: StatusBits.PositioningCommand	
					FALSE	A positioning command is not active on the axis.
					TRUE	The axis executes a positioning command of the "MC_MoveRelative" or "MC_MoveAbsolute" Motion Control instructions.
VelocityCommand	BOOL	FALSE	RCCP, RP, OPR	-	Execution of a command with velocity specification Name in Openness: StatusBits.VelocityCommand	
					FALSE	A command with velocity specification is not active on the axis.
					TRUE	The axis is executing a motion command with velocity specification of the "MC_MoveVelocity" or MC_MoveJog Motion Control instruction.
HomingCommand	BOOL	FALSE	RCCP, RP, OPR	-	Execution of a homing command Name in Openness: StatusBits.HomingCommand	
					FALSE	A homing command is not active on the axis.
					TRUE	The axis is executing a homing command of the "MC_Home" Motion Control instruction.
CommandTableActive	BOOL	FALSE	RCCP, RP, OPR	-	Execution of a command table Name in Openness: StatusBits.CommandTableActive	
					FALSE	A command table is not active on the axis.
					TRUE	The axis is controlled by Motion Control instruction "MC_CommandTable".
ConstantVelocity	BOOL	FALSE	RCCP, RP, OPR	-	Constant velocity Name in Openness: StatusBits.ConstantVelocity	
					FALSE	The axis is accelerating, decelerating, or at a standstill.
					TRUE	The setpoint velocity has been reached. The axis is moving at constant velocity.

Tag	Data type	Values	Access	W	Description	
Accelerating	BOOL	FALSE	RCCP, RP, OPR	-	Acceleration process Name in Openness: StatusBits.Accelerating	
					FALSE	The axis is decelerating, moving at constant velocity, or is at a standstill.
					TRUE	Axis is being accelerated.
Decelerating	BOOL	FALSE	RCCP, RP, OPR	-	Deceleration process Name in Openness: StatusBits.Decelerating	
					FALSE	The axis is accelerating, moving at constant velocity, or is at a standstill.
					TRUE	The axis is being decelerated.
ControlPanelActive	BOOL	FALSE	RCCP, RP, OPR	-	Activation status of the axis control panel Name in Openness: StatusBits.ControlPanelActive	
					FALSE	"Automatic mode" is activated. The user program has control priority over the axis.
					TRUE	The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the axis. The axis cannot be controlled from the user program.
DriveReady	BOOL	FALSE	RCCP, RP, OPR	-	Operational status of the drive Name in Openness: StatusBits.DriveReady	
					FALSE	The drive is not ready. Setpoints will not be executed.
					TRUE	The drive is ready. Setpoints can be executed.
RestartRequired	BOOL	FALSE	RCCP, RP, OPR	-	Restart of axis required Name in Openness: StatusBits.RestartRequired	
					FALSE	A restart of the axis is not required.
					TRUE	Values were modified in the load memory.
					To download the values to the work memory while the CPU is in RUN mode, the axis must be restarted. Use the MC_Reset Motion Control instruction to do this.	
SWLimitMinActive	BOOL	FALSE	RCCP, RP, OPR	-	Status of the software low limit switch Name in Openness: StatusBits.SWLimitMinActive	
					FALSE	The axis is kept within its configured work area.
					TRUE	The software low limit switch was reached or exceeded.
SWLimitMaxActive	BOOL	FALSE	RCCP, RP, OPR	-	Status of the software high limit switch Name in Openness: StatusBits.SWLimitMaxActive	

Tag		Data type	Values	Access	W	Description	
						FALSE	The axis is kept within its configured work area.
						TRUE	The software high limit switch was reached or exceeded.
	HWLimitMinActive	BOOL	FALSE	RCCP, RP, OPR	-	Status of the hardware low limit switch Name in Openness: StatusBits.HWLimitMinActive	
						FALSE	The axis is kept within its configured permitted traversing range.
						TRUE	The hardware low limit switch was reached or exceeded.
	HWLimitMaxActive	BOOL	FALSE	RCCP, RP, OPR	-	Status of the hardware high limit switch Name in Openness: StatusBits.HWLimitMaxActive	
						FALSE	The axis is kept within its configured permitted traversing range.
						TRUE	The hardware high limit switch was reached or exceeded.

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

ErrorBits tags as of V6 (S7-1200)

The tag structure <axis name>.ErrorBits.<tag name> indicates error at the technology object.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
ErrorBits.	STRUCT				TO_Struct_ErrorBits
SystemFault	BOOL	FALSE	RCCP, RP, OPR	-	Internal system error Name in Openness: ErrorBits.SystemFault
ConfigFault	BOOL	FALSE	RCCP, RP, OPR	-	Incorrect configuration of the axis Name in Openness: ErrorBits.ConfigFault
DriveFault	BOOL	FALSE	RCCP, RP, OPR	-	Error in the drive. Loss of the "Drive ready" signal. Name in Openness: ErrorBits.DriveFault
SWLimit	BOOL	FALSE	RCCP, RP, OPR	-	Software limit switch reached or exceeded Name in Openness: ErrorBits.SWLimit
HWLimit	BOOL	FALSE	RCCP, RP, OPR	-	Hardware limit switch reached or exceeded Name in Openness: ErrorBits.HWLimit
DirectionFault	BOOL	FALSE	RCCP, RP, OPR	-	Impermissible motion direction Name in Openness: ErrorBits.DirectionFault
HWUsed	BOOL	FALSE	RCCP, RP, OPR	-	Another axis is using the same PTO (Pulse Train Output) and is enabled with "MC_Power". Name in Openness: ErrorBits.HWUsed
SensorFault	BOOL	FALSE	RCCP, RP, OPR	-	Error in the encoder system Name in Openness: ErrorBits.SensorFault
Communication-Fault	BOOL	FALSE	RCCP, RP, OPR	-	Communication error Error in communication with a connected device. Name in Openness: ErrorBits.CommunicationFault
FollowingError	BOOL	FALSE	RCCP, RP, OPR	-	Maximum permitted following error exceeded Name in Openness: ErrorBits.FollowingError
PositioningFault	BOOL	FALSE	RCCP, RP, OPR	-	Positioning error The axis was not correctly positioned at the end of a positioning motion. Name in Openness: ErrorBits.PositioningFault
AdaptionError	BOOL	FALSE	RCCP, RP, OPR	-	Error during data transfer Name in Openness: ErrorBits.AdaptionError

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

ControlPanel tags as of V6 (S7-1200)

The "ControlPanel" tags do not contain any user-relevant data. These tags cannot be accessed in the user program.

Tags

The following tags "ControlPanel" are readable in Openness.

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
ControlPanel.	STRUCT				TO_Struct_ControlPanel

Tag	Data type	Values	Access	W	Description
Input	STRUCT				TO_Struct_Input
TimeOut	DINT	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Input.TimeOut
EsLifeSign	DINT	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Input.EsLifeSign
Command[1].	STRUCT				ARRAY[1..1] TO_Struct_Command
ReqCounter	DINT	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Input.Command[1].ReqCounter
Type	DINT	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Input.Command[1].Type
Position	REAL	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Input.Com-mand[1].Position
Velocity	REAL	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Input.Command[1].Velocity
Acceleration	REAL	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Input.Command[1].Acceleration
Jerk	REAL	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Input.Command[1].Jerk
Param	INT	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Input.Command[1].Param
Output.	STRUCT		-		TO_Struct_Output
RTLifeSign	INT	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Output.RTLifeSign
Command[1].	STRUCT		-		ARRAY[1..1] TO_Struct_Command
AckCounter	INT	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Output.Command[1].AckCounter
ErrorID	USH-ORT	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Output.Command[1].ErrorID
ErrorInfo	USH-ORT	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Output.Command[1].ErrorInfo
Done	BOOL	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Output.Command[1].Done

Tag	Data type	Values	Access	W	Description
Aborted	BOOL	-	OPR	-	(Internal parameter) Name in Openness: ControlPanel.Output.Command[1].Aborted

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

Internal tags as of V6 (S7-1200)

The "Internal" tags do not contain any user-relevant data. These tags cannot be accessed in the user program.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description
Internal[1..4].	STRUCT				ARRAY [1..4] TO_Struct_Internal
Id	INT	0 (-32768 to 32767)	OPRW	-	(Internal parameter) Name in Openness: Internal[1..4].Id
Value	REAL	0 (-9.0E15 to 9.0E15)	OPRW	-	(Internal parameter) Name in Openness: Internal[1..4].Value

See also

Tags of the positioning axis technology object V4...5 (Page 1595)

Tag of the axis technology object V1...3 (Page 1585)

Update of the technology object tags (S7-1200)

The status and error information of the axis indicated in the technology object tags is updated at each cycle control point.

Changes to the values of configuration tags do not take effect immediately. For information on the conditions under which a change takes effect, refer to the detailed description of the relevant tag.

Tags of the command table V6 technology object (S7-1200)

The tag structure <command table>.Command[n].<tag name> contains the configured command parameters.

Tags

Legend (Page 1488)

Tag	Data type	Values	Access	W	Description	
Command[n].	STRUCT				ARRAY[1..32] TO_Struct_Command	
Type	INT	0 (0 to 151)	RW, OPRW	-	Command type Name in Openness: Command[n].Type	
					0	"Empty" command
					2	"Stop" command
					5	"Relative positioning" command
					6	"Absolute positioning" command
					7	Command "Velocity setpoint"
					151	Command "Wait"
Position	REAL	0.0	RW, OPRW	-	Target position/traversing distance of the command Name in Openness: Command[n].Position	
Velocity	REAL	0.0	RW, OPRW	-	Command velocity Name in Openness: Command[n].Velocity	
Duration	REAL	0.0	RW, OPRW	-	Command duration Name in Openness: Command[n].Duration	
NextStep	INT	0 (0 to 1)	RW, OPRW	-	Mode for the transition to the next command Name in Openness: Command[n].NextStep	
					0	"Complete command"
					1	"Blend motion"
StepCode	WORD	0	RW, OPRW	-	Command step code Name in Openness: Command[n].StepCode	
WarningEnabled*	BOOL	FALSE	OPRW	-	Name in Openness: _WarningEnabled	
					FALSE	Warning is disabled.
					TRUE	Warning is enabled
UseAxisParametersFrom*	INT/ STRING	-	OPRW	-	Name in Openness: _UseAxisParametersFrom Axis number, "default axis", name of axis.	

*) Available in Openness

See also

Tag of the command table technology object V4...5 (Page 1621)

Tags of the command table technology object V1...3 (Page 1620)

Versions V1...5 (S7-1200)

CPU outputs relevant for motion control (technology version V1...3) (S7-1200)

The number of usable drives depends on the CPU, the number of PTOs (pulse train outputs) and the number of available pulse generator outputs.

The following tables provide information about the relevant dependencies:

Maximum number of PTOs

The maximum number of controllable PTOs (drives) depends on the article number of the CPU:

CPU article number	Number of PTOs
xxxxxxx-1xx**30**-xxxx	2
xxxxxxx-1xx**31**-xxxx	4

The maximum number of controllable PTOs (drives) applies regardless of the use of a signal board.

Usable pulse generator outputs

The CPU provides one pulse output and one direction output for controlling a stepper motor drive or a servo motor drive with pulse interface. The pulse output provides the drive with the pulses required for motor motion. The direction output controls the travel direction of the drive.

Pulse and direction outputs are permanently assigned to one another and form a pulse generator output. Onboard CPU outputs or outputs of a signal board can be used as pulse generator outputs. You select between onboard CPU outputs and outputs of the signal board during device configuration under Pulse generators (PTO/PWM) on the "Properties" tab.

The following table shows the number of usable drives per CPU or signal board:

CPU		On-board	Signal board				
			DI2/DO2 x DC24V 20kHz	DI2/DO2 x DC24V 200kHz	DO4 x DC24V 200kHz	DI2/DO2 x DC5V 200kHz	DO4 x DC5V 200kHz
CPU 1211C, CPU 1212C, CPU 1214C (MLFB - article number xxxxxxx-1xx**30**-xxxx)	DC/DC/DC	2	2	2	2	2	2
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2

CPU		On-board	Signal board				
CPU 1211C (MLFB - article number xxxxxxx-1xx**31**-xxxx)	DC/DC/DC	2	3	3	4	3	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1212C (MLFB - article number xxxxxxx-1xx**31**-xxxx)	DC/DC/DC	3	4	4	4	4	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1214C (MLFB - article number xxxxxxx-1xx**31**-xxxx)	DC/DC/DC	4	4	4	4	4	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1215C	DC/DC/DC	4	4	4	4	4	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2

The following table shows the address assignment of the pulse and direction outputs:

CPU S7-1200		PTO1 outputs [1]		PTO2 outputs [2]		PTO3 outputs [1]		PTO4 outputs [2]	
		Pls.	Dir.	Pls.	Dir.	Pls.	Dir.	Pls.	Dir.
CPU 1211C, CPU 1212C, CPU 1214C, CPU1215C (DC/DC/DC)	CPU	Ax.0	Ax.1	Ax.2	Ax.3	Ax.4	Ax.5	Ax.6	Ax.7
	Signal board	Ay.0	Ay.1	Ay.2	Ay.3	Ay.0	Ay.1	Ay.2	Ay.3
CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C (AC/DC/RLY)	CPU	-	-	-	-	-	-	-	-
	Signal board	Ay.0	Ay.1	Ay.2	Ay.3	Ay.0	Ay.1	Ay.2	Ay.3
CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C (DC/DC/RLY)	CPU	-	-	-	-	-	-	-	-
	Signal board	Ay.0	Ay.1	Ay.2	Ay.3	Ay.0	Ay.1	Ay.2	Ay.3

x = Initial byte address of onboard CPU outputs (default value = 0)

y = Initial byte address of signal board outputs (default value = 4)

[1] If a DC/DC/DC CPU variant is used together with a DI2/DO2 signal board, the signals of the PTO1/3 can be generated via the onboard CPU outputs or via the signal board.

[2] If a DC/DC/DC CPU variant is used together with a DO4 signal board, the signals for PTO1/3 or PTO2/4 can be generated via the onboard CPU outputs or via the signal board.

PTO3 and PTO4 are only available for CPUs with the article numbers xxxxxxx-1xx**31**-xxxx.

Note

Access to pulse generator outputs via the process image

The firmware takes control via the corresponding pulse generator outputs if the PTO (Pulse Train Output) is selected and assigned to an axis.

With this takeover of the control function, the connection between the process image and I/O output is also disconnected. Although the user has the option of writing the process image of pulse generator outputs via the user program or watch table, this is not transferred to the I/O output. Accordingly, it is also not possible to monitor the I/O output via the user program or watch table. The information read reflects the value of the process image and does not match the real status of the I/O output.

For all other CPU outputs that are not used permanently by the CPU firmware, the status of the I/O output can be controlled or monitored via the process image, as usual.

Limit frequencies of pulse outputs

The following limit frequencies apply to the pulse outputs:

Pulse output	Limit frequencies for technology object positioning axis V1	Limiting frequencies of the technology object positioning axis V2/V3 with CPU < V3.0	Limiting frequencies of the technology object positioning axis V2/V3 with CPU V3.0
On-board (MLFB article number xxxxxxx-1xx**30**-xxxx)	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 100 kHz	1 Hz ≤ f ≤ 100 kHz
On-board (MLFB - article number xxxxxxx-1xx**31**xxxx)	2 Hz ≤ f ≤ 100 kHz (PTO 1+2) 2 Hz ≤ f ≤ 20 kHz (PTO 3+4)	2 Hz ≤ f ≤ 100 kHz (PTO 1+2) 2 Hz ≤ f ≤ 20 kHz (PTO 3+4)	1 Hz ≤ f ≤ 100 kHz (PTO 1+2) 1 Hz ≤ f ≤ 20 kHz (PTO 3+4)
Signal board DI2/DO2 x DC24V 20kHz	2 Hz ≤ f ≤ 20 kHz	2 Hz ≤ f ≤ 20 kHz	1 Hz ≤ f ≤ 20 kHz
Signal board DI2/DO2 x DC24V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz
Signal board DO4 x DC24V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz
Signal board DI2/DO2 x DC5V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz
Signal board DO4 x DC5V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz

Drive signals

For motion control, you can optionally parameterize a drive interface for "Drive enabled" and "Drive ready". When using the drive interface the digital output for the drive enable and the digital input for "drive ready" can be freely selected.

Acceleration/deceleration limits

The following limits apply to acceleration and deceleration:

Acceleration/deceleration	Value (CPU < V3.0)	Value (CPU V3.0)
Minimum acceleration/deceleration	2.8E-1 pulses/s^2	5.0E-3 pulses/s^2
Maximum acceleration/deceleration	9.5E+9 pulses/s^2	9.5E+9 pulses/s^2

Jerk limits

The following limits apply to the jerk:

Jerk	Value (CPU < V3.0)	Value (CPU V3.0)
Minimum jerk	4.0E-2 pulses/s^3	4.0E-3 pulses/s^3
Maximum jerk	1.5E+8 pulses/s^3	1.0E+10 pulses/s^3

See also

CPU outputs relevant for motion control (Page 1300)

Configuration dialogs (S7-1200)

V1...3 (S7-1200)

Configuration - General ("Axis" technology object V1...3) (S7-1200)

Configure the basic properties of the "Axis" technology object in the "General" configuration window.

Axis name:

Define the name of the axis or the name of the "Axis" technology object in this box. The technology object is listed under this name in the project navigation.

Hardware interface

The pulses are output to the power unit of the drive by fixed assigned digital outputs.

In CPUs with relay outputs, the pulse signal cannot be output on these outputs because the relays do not support the necessary switching frequencies. A signal board with digital outputs must be used to enable you to work with the PTO (Pulse Train Output) on these CPUs.

Note

The PTO requires the functionality of a fast counter (HSC). An HSC is used for this purpose with CPU version <V3.0. The HSC is then no longer available to the user. An internal HSC is used for this with CPU version ≥ V3.0.

The count can not be evaluated from its input address.

The assignment between PTO and HSC is fixed. When the user activates PTO1, it is connected to the HSC1. If the PTO2 is activated, this is connected with the HSC2.

In the "Pulse generator selection" drop-down list, select the PTO (Pulse Train Output) which is to provide the pulses for controlling the stepper motors or servo motors with pulse interface. If the pulse generators and high-speed counters are not used elsewhere in the device configuration, the hardware interface can be configured automatically. In this case, the PTO selected in the drop-down list is displayed with a white background. The interfaces used are listed in the output boxes "Output source", "Pulse output", "Direction output" and "Assigned fast counter".

Proceed as follows if you wish to change the interfaces or if the PTO could not be automatically configured (entry in the "Pulse generator selection" drop-down list is highlighted in red):

1. Click on the "Device configuration" button.
   The pulse generator device configuration opens.
   Enlarge the property window of the device configuration if the configuration of the pulse generator is not visible.

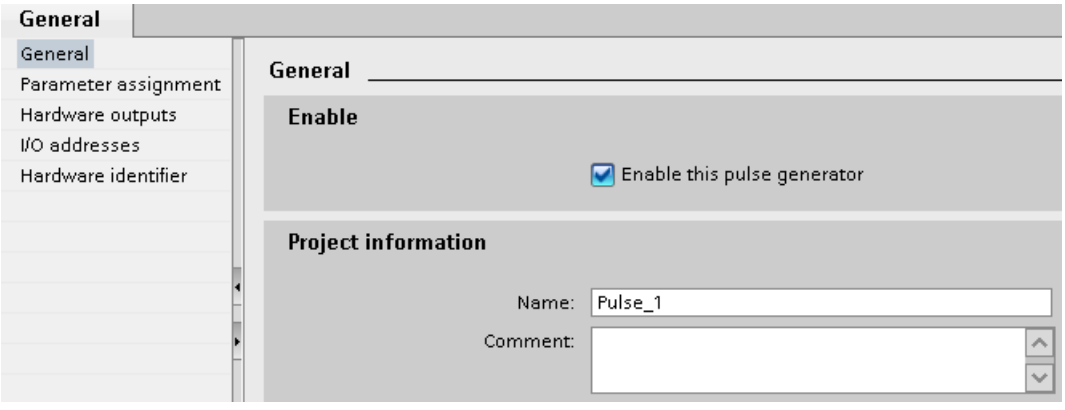

2. Select the "Enable this pulse generator" check box.

3. Select the "Parameter assignment" entry in the area navigation.
   The "Parameter assignment" opens.

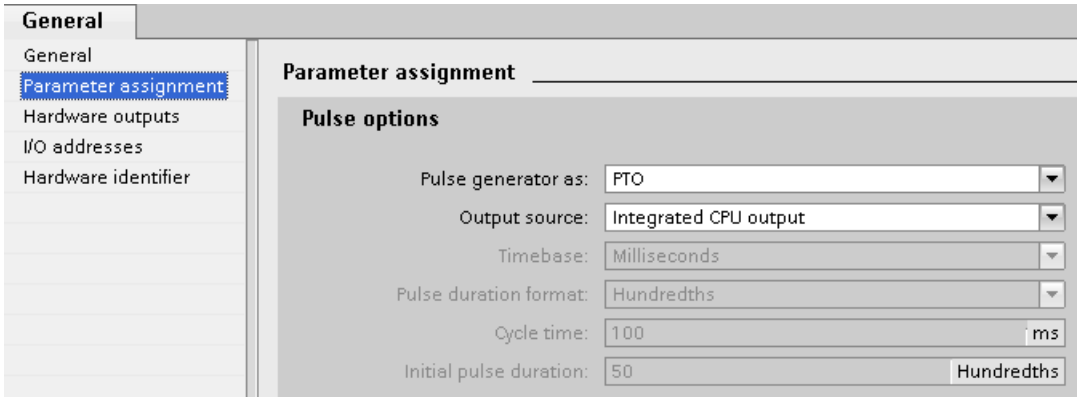

4. In the "Pulse generator as:" dropdown list select the "PTO" entry.
5. In the "Output source:" dropdown list select the "Integrated CPU output" or "Signal board output" entry. The "Signal board output" entry can only be selected for PTO1 or for PTO1 and PTO2 depending on the installed signal board. For more detailed information, see section: CPU outputs relevant for motion control (Page 1300)
6. Go back to the axis configuration.
   Unless the corresponding fast counter has already been used elsewhere, the PTO boxes of the "General" axis configuration are not shaded red. If this is not the case, correct the configuration based on the error messages.

User unit

Select the desired unit for the dimension system of the axis in the dropdown list. The selected unit is used for additional configuration of the "Axis" technology object and for displaying the current axis data.

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this unit.

Note

Later changing of the dimension system may not be converted correctly in all the configuration windows of the technology object. In this case check the configuration of all axis parameters.

The values of the input parameters of the Motion Control instructions may have to be adapted to the new unit of measurement in the user program.

See also

Configuration - General (Page 1343)

Configuration - Homing ("Axis" technology object V1) (S7-1200)

Configure the parameters for active and passive homing in the "Homing" configuration window. The homing method is set using the "Mode" input parameter of the Motion Control instruction. Here, Mode = 2 means passive homing and Mode = 3 means active homing.

Reference point switch input

Select the digital input for the reference point switch from the drop-down list. The input must be interrupt-capable. The onboard CPU inputs and the inputs of an inserted signal board can be selected as inputs for the reference point switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a reference point switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the reference point switch, the reference point may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the reference point switch.

Permitting direction reversal after reaching the HW limit switch (active homing only)

Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be activated for direction reversal. If the CPU firmware V1.0 is used, both hardware limit switches must be configured. If CPU firmware as of V2.0 is used, only the hardware limit switch in the approach direction must be configured.

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The reference point switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

Note

Use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low
- Increase the configured acceleration/deceleration
- Increase the distance between hardware limit switch and mechanical stop

Approach / homing direction (active and passive homing)

With the direction selection, you determine the "approach direction" used during active homing to search for the reference point switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the reference point switch to carry out the homing operation.

Refer to the table under "Reference point switches" for the effect of the approach direction setting on passive homing.

Side of the reference point switch (active and passive homing)

- **Active homing**
  This is where you select whether the axis is homed on the low or high end of the reference point switch.

  Note

  Depending on the start position of the axis and the configuration of the homing parameters, the homing procedure sequence can differ from the diagram in the configuration window.

- **Passive homing**
  With passive homing, the traversing motions for purposes of homing must be implemented by the user via motion commands. The end of the reference point switch on which homing occurs depends on the following factors:
  - "Approach direction" configuration
  - "Reference point switch" configuration
  - Current travel direction during passive homing

  The table below presents details on the effect of factors:

Influencing factors:			Result:
Configuration Approach direction	Configuration Reference point switch	Current travel direction	Homing on Reference point switch
Positive	"Bottom side"	Positive direction	**Top side**
		Negative direction	**Bottom side**
Positive	"Top side"	Positive direction	**Bottom side**
		Negative direction	**Top side**
Negative	"Bottom side"	Positive direction	**Bottom side**
		Negative direction	**Top side**
Negative	"Top side"	Positive direction	**Top side**
		Negative direction	**Bottom side**

Velocity (active homing only)

In this box, specify the velocity at which the reference point switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity (active homing only)

In this box, specify the velocity at which the axis approaches the reference point switch for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset (active homing only)

If the desired home position deviates from the position of the reference point switch, the home position offset can be specified in this box.

If the value does not equal 0, the axis executes the following actions following homing at the reference point switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Configuration - Homing ("Axis" technology object V2...3) (S7-1200)

Configuration - Homing - General (Axis technology object V2...3) (S7-1200)

Configure the reference point switch input for active and passive homing in the "Homing - General" configuration window.

Reference point switch input

Select the digital input for the homing switch from the drop-down list. The input must be interrupt-capable. The on-board CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the homing switch.

Active level

In the drop-list, select the level of the homing switch that is to be used for homing.

See also

Sequence - Active homing (Page 1373)

Configuration - Homing - Passive (Axis technology object V2...3) (S7-1200)

Configure the necessary parameters for passive homing in the "Homing - Passive" configuration window.

The movement for passive homing must be triggered by the user (e.g. using an axis motion command). Passive homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 2.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Note

If passive homing is carried out without an axis motion command (axis at a standstill), homing will be executed upon the next rising or falling edge at the homing switch.

Configuration - Homing - Active (Axis technology object V2...3) (S7-1200)

Configure the necessary parameters for active homing in the "Active homing" configuration window. Active homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 3.

Permit auto reverse at the hardware limit switch

Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be enabled for the reversal of direction (at least the hardware limit switch in the direction of approach must be configured).

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The homing switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

Note

If possible, use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low.
- Increase the configured acceleration/deceleration.
- Increase the distance between the hardware limit switch and the mechanical endstop.

Approach/homing direction

With the direction selection, you determine the approach direction used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the homing switch to carry out the homing operation.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Velocity

In this box, specify the velocity at which the homing switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity

In this box, specify the velocity at which the reference point switch is to be approached for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset

If the desired home position deviates from the position of the homing switch, the home position offset can be specified in this box.

If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Changing configuration of dynamic values in user program ("Axis" technology object V1...3) (S7-1200)

You can change the following configuration parameters during runtime of the user program in the CPU:

Acceleration and deceleration

You can also change the values for acceleration and deceleration during runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.Config.DynamicDefaults.Acceleration
  for changing acceleration
- <Axis name>.Config.DynamicDefaults.Deceleration
  for changing deceleration

Refer to the description of the technology object tags (Page 1585) in the appendix for information on when changes to the configuration parameters take effect.

Emergency stop deceleration

You can also change the value for the emergency stop deceleration during runtime of the user program. Use the following technology object tag for this purpose:

- <Axis name>.Config.DynamicDefaults.EmergencyDeceleration

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

NOTICE
After changes to this parameter, it may be necessary to adapt the positions of the hardware limit switches and other safety-relevant settings.

Jerk limit (as of technology object "Axis" V2.0)

You can also activate and deactivate the jerk limit at runtime of the user program and change the value for the jerk. Use the following technology object tags for this purpose:

- <Axis name>.Config.DynamicDefaults.JerkActive
  for activating and deactivating the jerk limit
- <Axis name>.Config.DynamicDefaults.Jerk
  for changing the jerk

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

See also

Changing the configuration of dynamics in the user program (Page 1368)

V4 (S7-1200)

Configuration - General (positioning axis technology object as of V4) (S7-1200)

Configure the basic properties of the positioning axis technology object in the "General" configuration window.

Axis name

Define the name of the axis or the name of the positioning axis technology object in this field. The technology object is listed under this name in the project tree.

Hardware interface

The pulses are output to the power unit of the drive by fixed assigned digital outputs.

In CPUs with relay outputs, the pulse signal cannot be output on these outputs because the relays do not support the necessary switching frequencies. A signal board with digital outputs must be used to enable you to work with the PTO (Pulse Train Output) on these CPUs.

Note

The PTO requires the functionality of a high-speed counter (HSC). An internal HSC is used for this, the count of which cannot be evaluated.

Selecting the pulse generator

In the drop-down list, select the PTO (Pulse Train Output) which is to provide the pulses for controlling the stepper motors or servo motors with pulse interface. If the pulse generators and high-speed counters are not used elsewhere in the device configuration, the hardware interface can be configured automatically. In this case, the PTO selected in the drop-down list is displayed with a white background.

"Device configuration" button

This button takes you to the settings for the pulse options in the device configuration of the CPU.

Signal type

Select the desired signal type in the drop-down list. The following signal types are available:

- **PTO (pulse A and direction B)**
  A pulse output and a direction output are used for controlling the stepper motor.
- **PTO (count up A, count down B)**
  One pulse output each for motion in positive direction and negative direction is used for controlling the stepper motor.
- **PTO (A/B phase offset)**
  Both pulse outputs for Phase A and for Phase B run at the same frequency.
  The period of the pulse outputs is evaluated at the drive end as a step.
  The phase offset between Phase A and Phase B determines the direction of the motion.
- **PTO (A/B phase offset - quadruple)**
  Both pulse outputs for Phase A and for Phase B run at the same frequency.
  All positive edges and all negative edges of Phase A and Phase B are evaluated as a step at the drive end.
  The phase offset between Phase A and Phase B determines the direction of the motion.

Pulse output (signal type "PTO (pulse A and direction B)")

Select the desired output for the pulse output in this field.

You can select the output using a symbolic address or assign it to an absolute address.

Activate direction output (signal type "PTO (pulse A and direction B)")

You can deactivate and activate the direction output in "pulse and direction" mode. This option enables you to limit the motion direction.

Direction output (signal type "PTO (pulse A and direction B)")

Select the desired output for the direction output in this field.

You can select the output using a symbolic address or assign it to an absolute address.

Pulse output up (signal type "PTO (count up A, count down B)")

Select the desired pulse output for motions in positive direction in this field.

You can select the output using a symbolic address or assign it to an absolute address.

Pulse output down (signal type "PTO (count up A, count down B)")

Select the desired pulse output for motions in negative direction in this field.

You can select the output using a symbolic address or assign it to an absolute address.

Phase A (signal types "PTO (A/B phase offset)" and "PTO (A/B phase offset - quadruple)")

Select the desired pulse output for Phase A signals in this field.

You can select the output using a symbolic address or assign it to an absolute address.

Phase B (signal types "PTO (A/B phase offset)" and "PTO (A/B phase offset - quadruple)")

Select the desired pulse output for Phase B signals in this field.

You can select the output using a symbolic address or assign it to an absolute address.

User unit

Select the desired unit for the dimension system of the axis in the drop-down list. The selected unit is used for further configuration of the positioning axis technology object and for displaying the current axis data.

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this unit.

Note

Later changing of the dimension system may not be converted correctly in all the configuration windows of the technology object. In this case check the configuration of all axis parameters.

The values of the input parameters of the Motion Control instructions may have to be adapted to the new unit of measurement in the user program.

Configuration - Drive signals (positioning axis technology object V4) (S7-1200)

Configure the output for drive enable and the input for the "Drive ready" feedback signal of the drive in the "Drive signals" configuration window.

Drive enable is controlled by Motion Control instruction "MC_Power" and enables power to the drive. The signal is provided to the drive via the output to be configured.

The drive signals "Drive ready" to the CPU if it is ready to start executing travel after receipt of drive enable. The "Drive ready" signal is reported back to the CPU via the input to be configured.

If the drive does not have any interfaces of this type, you will not have to configure the parameters. In this case, select the value TRUE for the ready input.

See also

Configuration - Mechanics (positioning axis technology object V4) (Page 1543)

Position limits (Page 1357)

Dynamics (Page 1362)

Homing (positioning axis technology object as of V2) (Page 1369)

Configuration - Mechanics (positioning axis technology object V4) (S7-1200)

Configure the mechanical properties of the drive in the "Mechanics" configuration window.

Increments per motor revolution

Configure the number of pulses required for one revolution of the motor in this box.

Limits (independent of the selected unit of measurement):

- 0 < Pulse per motor revolution ≤ 2147483647

Distance per revolution

In this box, configure the load distance per motor revolution covered by the mechanical system of your unit.

Limits (independent of the selected unit of measurement):

- 0.0 < Distance per revolution ≤ 1.0e12

Permitted direction of rotation (technology version as of V4)

Configure this box to determine whether the mechanics of your system are to move in both directions or only in the positive or negative direction.

If you have not activated the direction output of the pulse generator in the "PTO (pulse A and direction B)" mode, the selection is limited to the positive or negative direction.

Invert direction

You can use the "Invert direction" check box to adapt the control system to the direction logic of the drive.

The direction logic is inverted according to the selected mode of the pulse generator:

- **PTO (pulse A and direction B)**
  - 0 V at direction output ⇒ positive direction of rotation
  - 5 V/24 V at direction output ⇒ negative direction of rotation

  The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.
- **PTO (count up A, count down B)**
  The outputs "Pulse output down" and "Pulse output up" are swapped.
- **PTO (A/B phase offset)**
  The "Phase A" and "Phase B" outputs are swapped.
- **"PTO (A/B phase offset - quadruple)**
  The "Phase A" and "Phase B" outputs are swapped.

See also

Configuration - Drive signals (positioning axis technology object V4) (Page 1542)

Relationship between the signal type and the direction of travel (Page 1303)

Position limits (Page 1357)

Dynamics (Page 1362)

Homing (positioning axis technology object as of V2) (Page 1369)

Configuration - Homing - Passive (Positioning axis technology object V4) (S7-1200)

Configure the necessary parameters for passive homing in the "Homing - Passive" configuration window.

The movement for passive homing must be triggered by the user (e.g. using an axis motion command). Passive homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 2.

Homing switch input

Select the digital input for the homing switch from the drop-down list. The input must be interrupt-capable. The on-board CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the homing switch.

Active level

In the drop-list, select the level of the homing switch that is to be used for homing.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Note

If passive homing is carried out without an axis motion command (axis at a standstill), homing will be executed upon the next rising or falling edge at the homing switch.

Configuration - Homing - Active (Positioning axis technology object V4) (S7-1200)

Configure the necessary parameters for active homing in the "Active homing" configuration window. Active homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 3.

Homing switch input

Select the digital input for the homing switch from the drop-down list. The input must be interrupt-capable. The on-board CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the homing switch.

Active level

In the drop-list, select the level of the homing switch that is to be used for homing.

Allowing direction reversal at the hardware limit switch

Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be enabled for the reversal of direction (at least the hardware limit switch in the direction of approach must be configured).

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The homing switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

Note

If possible, use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low.
- Increase the configured acceleration/deceleration.
- Increase the distance between the hardware limit switch and the mechanical endstop.

Approach/homing direction

With the direction selection, you determine the approach direction used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the homing switch to carry out the homing operation.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Approach velocity

In this box, specify the velocity at which the homing switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity

In this box, specify the velocity at which the homing switch is to be approached for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset

If the desired home position deviates from the position of the homing switch, the home position offset can be specified in this box.

If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Diagnostics - Status and error bits ("Axis" technology object V1...3) (S7-1200)

You use the "Status and error bits" diagnostic function to monitor the most important status and error messages for the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The status error messages have the following meaning:

Status of the axis

Status	Description
Enabled	The axis is enabled and ready to be controlled via Motion Control commands. (Tag of technology object: <Axis name>.StatusBits.Enable)
Homed	The axis is homed and is capable of executing absolute positioning commands of Motion Control instruction "MC_MoveAbsolute". The axis does not have to be homed for relative positioning. Special situations: • During active homing, the status is FALSE. • If a homed axis undergoes passive homing, the status is set to TRUE during passive homing. (Tag of technology object: <Axis name>.StatusBits.HomingDone)
Axis error	An error has occurred in the "Axis" technology object. Additional information about the error is available in automatic control at the ErrorID and ErrorInfo parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error. (Tag of technology object: <Axis name>.StatusBits.Error)
Axis control panel enabled	The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the "Axis" technology object. The axis cannot be controlled from the user program. (Tag of technology object: <Axis name>.StatusBits.ControlPanelActive)
Restart necessary	A modified configuration of the axis was downloaded to the load memory in CPU RUN operating mode. To download the modified configuration to the work memory, you need to restart the axis. Use the Motion Control instruction MC_Reset to do this.

Drive status

Status	Description
Drive ready	The drive is ready for operation. (Tag of technology object: <Axis name>.StatusBits.DriveReady)
Drive error	The drive has reported an error due to loss of its "Drive ready" signal. (Tag of technology object: <Axis name>.ErrorBits.DriveFault)

Status of the axis motion

Status	Description
Standstill	The axis is at a standstill. (Tag of technology object: <Axis name>.StatusBits.StandStill)
Acceleration	The axis accelerates. (Tag of technology object: <Axis name>.StatusBits.Acceleration)
Constant velocity	The axis travels at constant velocity. (Tag of technology object: <Axis name>.StatusBits.ConstantVelocity)
Deceleration	The axis decelerates (slows down). (Tag of technology object: <Axis name>.StatusBits.Deceleration)

Status of the motion mode

Status	Description
Positioning	The axis executes a positioning command of the Motion Control instruction "MC_MoveAbsolute", "MC_MoveRelative" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.PositioningCommand)
Travel with velocity specification	The axis executes a command with velocity specification of the Motion Control instruction "MC_MoveVelocity", "MC_MoveJog" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.SpeedCommand)
Homing	The axis executes a homing command of the Motion Control instruction "MC_Home" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.Homing)
Command table active (as of technology object Axis V2.0)	The axis is controlled by Motion Control instruction "MC_CommandTable". (Tag of technology object: <Axis name>.StatusBits.CommandTableActive)

Error messages

Error	Description
Lower SW limit switch was reached	The lower software limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMinReached)
Lower SW limit switch was exceeded	The lower software limit switch has been exceeded. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMinExceeded)
Upper SW limit switch was reached	The upper software limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMaxReached)
Upper SW limit switch was exceeded	The upper software limit switch has been exceeded. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMaxExceeded)
Lower HW limit switch was reached	The lower hardware limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.HwLimitMin)
Upper HW limit switch was reached	The upper hardware limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.HwLimitMax)
PTO and HSC already in use	A second axis is using the same PTO (Pulse Train Output) and HSC (High Speed Counter) and is enabled with "MC_Power". (Tag of technology object: <Axis name>.ErrorBits.HwUsed)
Configuration error	The "Axis" technology object was incorrectly configured or editable configuration data were modified incorrectly during runtime of the user program. (Tag of technology object: <Axis name>.ErrorBits.ConfigFault)
Internal error	An internal error has occurred. (Tag of technology object: <Axis name>.ErrorBits.SystemFault)

See also

Status and error bits (technology objects as of V4) (Page 1447)

ErrorIDs and ErrorInfos (S7-1200)

List of ErrorIDs and ErrorInfos (technology objects V4...5) (S7-1200)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Depending on the error reaction, the axis is stopped in the case of operating errors with stop of axis. The following error reactions are possible:

- **Remove enable**
  The setpoint zero is output and the enable is removed. The axis is braked depending on the configuration in the drive, and is brought to a standstill.
- **Stop with emergency stop ramp**
  Active motion commands are aborted. The axis is braked with the emergency stop deceleration configured under "Technology object > Extended parameters > Dynamics > Emergency stop ramp" without any jerk limit and brought to a standstill.

Operating error with axis stop

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8000		**Drive error, loss of "Drive ready"**		-
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; restart command, if necessary	
16#8001		**SW low limit switch has been tripped**		-
	16#000E	The position of the SW low limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the SW limit switch	
	16#000F	The position of the SW low limit switch was reached with the emergency stop deceleration		
	16#0010	The position of the SW low limit switch was exceeded with the emergency stop deceleration		Remove enable
16#8002		**SW high limit switch has been tripped**		-
	16#000E	The position of the SW high limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the SW limit switch	
	16#000F	The position of the SW high limit switch was reached with the emergency stop deceleration		
	16#0010	The position of the SW high limit switch was exceeded with the emergency stop deceleration		Remove enable

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8003		HW low limit switch was reached		For drive connection via PTO (Pulse Train Output): Stop with emergency stop ramp For drive connection via PROFIdrive/analog output: Remove enable
	16#000E	The HW low limit switch was reached. The axis was stopped with the emergency stop deceleration. (The homing switch was not found during an active homing procedure)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the HW limit switch.	
16#8004		HW high limit switch was reached		For drive connection via PTO (Pulse Train Output): Stop with emergency stop ramp For drive connection via PROFIdrive/analog output: Remove enable
	16#000E	The HW high limit switch was reached. The axis was stopped with the emergency stop deceleration. (The homing switch was not found during an active homing procedure)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the HW limit switch.	
16#8005		PTO/HSC are already being used by another axis		-
	16#0001	-	The axis was configured incorrectly: Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller	
			More than one axis is to run with one PTO: Another axis is using the PTO / HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 1452))	
16#8006		A communication error has occurred in the axis control panel		Remove enable
	16#0012	A timeout has occurred.	Check the cable connection and press the "Manual control" button again	
16#8007		The axis cannot be enabled		-
	16#0025	Restarting	Wait until the axis restart is complete.	
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8008		Invalid direction of motion		-
	16#002E	The selected motion direction is not allowed.	• Adjust the motion direction and restart the command. • Adjust the allowed direction of rotation in the technology object configuration under "Extended parameters > Mechanics". Restart the command.	
	16#002F	A reversing motion is not possible with the selected motion direction.		
16#8009		Reference switch/encoder zero mark not found		Stop with emergency stop ramp
	16#0033	Error in the configuration, hardware or installation of the encoder or at the homing switch.	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.	
16#800A		Alarm message from encoder		Remove enable
	16#0001	-	Check the device with regard to function, connections and I/Os.	
	16#0034	Hardware error at encoder		
	16#0035	Encoder dirty		
	16#0036	Error during reading of encoder absolute value	Compare the encoder type in the drive or encoder parameter P979 with the configuration data of the technology object.	
	16#0037	Zero mark monitoring of the encoder	Encoder reports error in zero mark monitoring (fault code 0x0002 in Gx_XIST2, see PROFIdrive profile). Check the plant for electromagnetic compatibility (EMC).	
	16#0038	Encoder is in "Parking" state	• Search for the cause of the error in the connected drive or encoder. • Check whether the error message was possibly triggered by a commissioning action at the drive or encoder.	
	16#0040	PROFIdrive: Encoder on bus failed (station failure).	Check the device with regard to function, connections and I/Os.	
	16#0041	PROFIdrive: Signs of life of encoder faulty.		
16#800B		Range violation of the position		Remove enable
	16#0039	Range violation in positive direction	Home the axis to a valid actual value range.	
	16#003A	Range violation in negative direction		
	16#003B	The change of the actual position in a position control clock cycle is greater than the modulo length.	Adjust the modulo length of the employed encoder.	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#800C		Alarm message from drive		Remove enable
	16#0001	-	Check the device with regard to function, connections and I/Os. In the "Tuning" dialog box, select a smaller gain (Kv).	
	16#003C	PROFIdrive: Drive signal "Control requested" failed.		
	16#003D	PROFIdrive/analog drive connection: Drive has shut down.		
	16#003E	PROFIdrive: Drive on bus failed (station failure)		
	16#003F	PROFIdrive: Signs of life of drive faulty.	• Check the device with regard to function, connections and I/Os. • Compare the clock parameters of HW Config (PROFIBUS line, slave OM for drive or encoder) and the execution system. Tmapc and servo must be configured with the same clock cycle time.	
16#800D		The permitted following error was exceeded		Remove enable
	16#0001	-	• Check the configuration of the control loop. • Check the direction signal of the encoder. • Check the configuration of following error monitoring.	
16#800E		Error at the hardware limit switch		Remove enable
	16#0042	Illegal free travel direction with active hardware limit switch	The programmed direction of movement is disabled due to the active hardware limit switch. Retract the axis in the opposite direction.	
	16#0043	Hardware limit switch polarity is reversed, axis cannot be freed	Check the mechanical configuration of the hardware limit switch.	
	16#0044	Both hardware limit switches are active, axis cannot be freed		
16#800F		Error in target range		Remove enable
	16#0045	Target range not reached	Target range was not reached within the positioning tolerance time. • Check the configuration of the position monitoring. • Check the configuration of the control loop.	
	16#0046	Exit target range again	The target range was exited within the minimum dwell time. • Check the configuration of the position monitoring. • Check the configuration of the control loop.	

ErrorID	ErrorInfo	Description	Remedy	Error reaction
16#8010		Position of the SW low limit switch is greater than that of the SW high limit switch when the axis is not a modulo axis		Remove enable
	16#0001	-	Change the position of the software limit switches.	

Operating error without axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8200		Axis is not enabled	
	16#0001	-	Enable the axis; restart the command
16#8201		Axis has already been enabled by another "MC_Power" instance	
	16#0001	-	Enable the axis with only one "MC_Power" instance
16#8202		The maximum number of simultaneous motion control commands has been exceeded (max. 200 commands for drive connection via PTO (Pulse Train Output), max. 100 commands for drive connection via PROFIdrive/analog output)	
	16#0001	-	Reduce the number of simultaneously active commands; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		Axis is currently operated in "Manual control" (axis control panel)	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		Axis is not homed	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		The axis is currently controlled by the user program (the error is only displayed in the axis control panel)	
	16#0013	The axis is enabled in the user program	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis control panel
16#8206		Technology object not activated yet	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis control panel.
16#8207		Command rejected	

ErrorID	ErrorInfo	Description	Remedy
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt".
	16#0018	The axis cannot be moved with a command table while it is being actively or passively homed.	Wait until direct or passive homing is complete.
	16#0019	The axis cannot be actively or passively homed while a command table is being processed.	Wait for command table to finish or abort the command table with a motion command, for example, "MC_Halt".
	16#0052	The specified position exceeds the numerical limit.	Enter a valid position value at the Motion Control instruction.
	16#0053	The axis is ramping up.	Wait until the axis is ready for operation.
	16#0054	Actual value is invalid	To execute a "MC_Home" command, the actual values must be valid. Check the status of the actual values. The variable of the technology object <Axis name>."StatusSensor.State" must show the value 2 (valid).
16#8208		**Difference between maximum and start/stop velocity is invalid**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#8209		**Invalid acceleration for technology object "Axis"**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#820A		**It is not possible to restart the axis**	
	16#0013	The axis is enabled in the user program	Disable the axis with the "MC_Power" instruction; restart again
	16#0027	The axis is currently being operated in "Manual control" (axis control panel)	Exit "Manual control"; restart again
	16#002C	The axis is not disabled.	Disable the axis; restart the command
	16#0047	The technology object is not ready for restart.	Download the project again.
	16#0048	Condition for restart of the technology object is not satisfied.	Disable the technology object.
16#820B		**It is not possible to execute the command table**	
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.
16#820C		**No configuration available**	
	16#0001	-	Internal error Contact the hotline.

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than -1E+12)	

ErrorID	ErrorInfo	Description	Remedy
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than -1E+12)	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
	16#0024	Value is less than 0	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	The direction reversal is activated at the hardware limit switch, despite the fact that the hardware limit switches are disabled	• Activate the HW limit switch using the variable <Axis name>.PositionLimitsHW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0055	Invalid mode at incremental encoder	Start a homing process for an incremental encoder using parameter "Mode" = 0, 1, 2, 3.
	16#0056	Invalid mode at absolute encoder	Passive and active homing ("Mode" = 2, 3) are not possible for an absolute value encoder. Start a homing process for an absolute encoder using parameter "Mode" = 0, 1.
16#8405		**Invalid value at parameter "StopMode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; enable the axis again
16#8406		**Simultaneous forward and backward jogging is not allowed**	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		**Switching to another axis with instruction "MC_Power" is only permitted after disabling the active axis.**	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.
16#8408		**Invalid value at parameter "Axis" of the Motion Control instruction**	

ErrorID	ErrorInfo	Description	Remedy
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#8409		**Invalid value at parameter "CommandTable" of the Motion Control instruction**	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#840A		**Invalid value at parameter "StartStep" of the Motion Control instruction**	
	16#000A	Value is less than or equal to 0.	Correct the value; restart the command
	16#001D	The start step is greater than the end step	
	16#001E	Value is greater than 32	
16#840B		**Invalid value at parameter "EndStep" of the Motion Control instruction**	
	16#000A	Value is less than or equal to 0.	Correct the value; restart the command
	16#001E	Value is greater than 32	
16#840C		**Invalid value at parameter "RampUpTime" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840D		**Invalid value at parameter "RampDownTime" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840E		**Invalid value at parameter "EmergencyRampTime" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840F		**Invalid value at parameter "JerkTime" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#8410		**Invalid value at parameter "Parameter" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000B	Address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0028	Data type of VARIANT pointer "Parameter" and "Value" do not match.	Use a suitable data type; restart command
	16#0029	VARIANT pointer "Parameter" does not point to a data block of the technology object.	Correct the VARIANT pointer; restart the command
	16#002A	The value at the VARIANT pointer "Parameter" cannot be read.	Correct the VARIANT pointer; restart the command
	16#002B	The value at the VARIANT pointer "Parameter" cannot be written.	Correct the VARIANT pointer or value; restart the command
	16#002C	The axis is not disabled.	Disable the axis; restart the command

ErrorID	ErrorInfo	Description	Remedy
16#8411		**Invalid value at parameter "Value" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command

Configuration error of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8600		**Parameter assignment of pulse generator (PTO is invalid**	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8601		**Parameter assignment of the high-speed counter (HSC) is invalid**	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8602		**Invalid parameter assignment of "Enable output"**	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8603		**Invalid parameter assignment of "Ready input"**	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8604		**Invalid "Pulses per motor revolution" value**	
	16#000A	Value is less than or equal to zero	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8605		**Invalid "Distance per revolution" value**	
	16#0002	The value has an invalid number format	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#000A	Value is less than or equal to zero	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8606		**Invalid "Start/stop velocity" value**	
	16#0002	The value has an invalid number format	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value is higher than the hardware high limit	
	16#0004	Value is lower than the hardware low limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		**Invalid "maximum velocity" value**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	The value has an invalid number format	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value is higher than the hardware high limit	
	16#0004	Value is lower than the hardware low limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8608		Invalid "Acceleration" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the hardware high limit	
	16#0004	Value is lower than the hardware low limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8609		Invalid "Deceleration" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the hardware high limit	
	16#0004	Value is lower than the hardware low limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#860A		Invalid "Emergency stop deceleration" value	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the hardware high limit	
	16#0004	Value is lower than the hardware low limit	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#860B		Value for position of the SW low limit switch is invalid	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than -1E+12)	
	16#0030	The position value of the software low limit switch is greater than that of the software high limit switch	
16#860C		Value for position of the SW high limit switch is invalid	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than -1E+12)	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#860D		Invalid address of the HW low limit switch	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#860E		Invalid address of the HW high limit switch	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#860F		Invalid "home position offset" value	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than -1E+12)	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8610		**Invalid "approach velocity" value**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8611		**Invalid "Homing velocity" value**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8612		**Invalid address of the homing switch**	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#8613		**During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured**	

ErrorID	ErrorInfo	Description	Remedy
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8614		Invalid "Jerk" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#001F	Value is greater than the maximum jerk	
	16#0020	Value is less than the minimum jerk	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8615		Value for "Unit of measurement" is invalid	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8616		Address of homing switch is invalid (passive homing as of V4)	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8617		Value of tag <Axis name>.Sensor.Sensor[1].ActiveHoming.Mode is invalid	
	16#0011	The selection value is invalid (Valid value: 2 = Homing via digital input)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8618		Value of tag <Axis name>.Sensor.Sensor[1].PassiveHoming.Mode is invalid	

ErrorID	ErrorInfo	Description	Remedy
	16#0011	The selection value is invalid (Valid value: 2 = Homing via digital input)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8619		**Value of tag <Axis name>.Actor.Type is invalid**	
	16#0011	The selection value is invalid (Valid value: 2 = Connection via pulse interface)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861A		**Value for "Permitted direction of rotation" is invalid**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#002D	"Both directions" not permitted with direction output switched off	
16#861B		**Faulty load gear factors**	
	16#0031	Valid is invalid.	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#861C		**Illegal combination of data for homing with incremental encoder**	
	16#0031	Valid is invalid.	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861D		**The set encoder mounting type is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Mounting-Mode**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861E		**The configuration of the measuring wheel circumference of the encoder is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.DistancePerRevolution**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary

ErrorID	ErrorInfo	Description	Remedy
16#861F		**The configuration for the resolution of the linear encoder is faulty. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.Resolution**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8620		**The set fine resolution for Gn_XIST1 is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.FineResolutionXist1**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8621		**The set fine resolution for Gn_XIST1 in <Axis name>.Sensor.Sensor[1].Parameter.FineResolutionXist1 is not consistent with the setting in PROFIdrive parameter P979**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8622		**Invalid value for the configuration date <Axis name>.Actor.Interface.AddressIn or <Axis name>.Actor.Interface.AddressOut**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8623		**The value set in the tag <axis name>.Sensor.Sensor[1].Type is invalid.**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8624		**The set encoder system is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].System**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8625		**Parameter of position monitoring is faulty. Invalid value in <Axis name>.PositioningMonitoring.MinDwellTime**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary

ErrorID	ErrorInfo	Description	Remedy
16#8626		**Parameter of position monitoring is faulty. Invalid value in <Axis name>.PositioningMonitoring.Window**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8627		**The configuration of the PROFIdrive interface of the actual value is faulty. Invalid value in <Axis name>.Sensor.Sensor[1].Interface.AddressIn or <Axis name>.Sensor.Sensor[1].Interface.AddressOut**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8628		**Faulty controller factors**	
	16#0030	Value has an incorrect number format or is outside the valid number range	The value for the gain or the precontrol of the control loop is faulty. • Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with the "MC_Reset" instruction and restart the command, if necessary>.PositionControl.Kv, <axis name>.PositionControl.Kpc)
16#8629		**Limit for standstill signal is faulty. Invalid value in <Axis name>.StandStillSignal.VelocityThreshold**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862A		**Parameter of position monitoring is faulty. Invalid value in <Axis name>.PositioningMonitoring.ToleranceTime**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862B		**Inconsistent PROFIBUS parameterization; the sum of Ti and To is greater than one DP cycle**	
	16#0030	Value has an incorrect number format or is outside the valid number range	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#862C		**Parameter of standstill monitoring is faulty. Invalid value in <Axis name>.StandStillSignal.MinDwellTime**	

ErrorID	ErrorInfo	Description	Remedy
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862D		Parameter of following error monitoring is faulty. Invalid value in <Axis name>.FollowingError.MinValue	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862E		Invalid value for the configuration date <Axis name>.Modulo.Length	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#862F		Invalid value for the configuration date <Axis name>.Modulo.StartValue	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8630		Invalid value for the configuration date <Axis name>.Actor.DriveParameter.ReferenceSpeed	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8631		The set fine resolution for Gn_XIST2 is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.FineResolutionXist2	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8632		The number of determinable encoder revolutions is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].Parameter.DeterminableRevolutions	

ErrorID	ErrorInfo	Description	Remedy
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8633		**The specified approach direction of the homing switch for passive homing is invalid. Invalid value in <Axis name>.Sensor.Sensor[1].PassiveHoming.Direction**	
16#8634		**Parameter of the following error monitoring is faulty. Invalid value in <Axis name>.FollowingError.MaxValue**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8635		**Parameter of the following error monitoring is faulty. Invalid value in <Axis name>.FollowingError.MinVelocity**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8636		**Controller factor is incorrect. Invalid value of the precontrol factor <Axis name>.PositionControl.Kpc**	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8637		**Invalid value for the configuration date <Axis name>.Sensor.Sensor[1].Interface.Type**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8638		**Invalid value for the configuration date <Axis name>.Sensor.Sensor[1].Interface.HSC**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8639		**Error at the drive**	
	16#0049	Configuration error at device	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#004A	The technology needs a smaller servo clock.	Internal system error. Check the project for consistency and reload it into the controller.
	16#004B	Device driver not initialized during ramp-up.	To enable a technology object, the actuator driver must be initialized. Execute the command again later.

ErrorID	ErrorInfo	Description	Remedy
16#863A		Communication to the drive is faulty	
	16#004C	Configuration error at device	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004D	The device driver needs a smaller servo clock.	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004E	Error in internal communication with the device	Check the project for consistency and reload it into the controller.
16#863B		Error at encoder	
	16#0049	Configuration error at device	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#004A	The technology needs a smaller servo clock.	Internal system error. Check the project for consistency and reload it into the controller.
	16#004B	Device driver not initialized during ramp-up.	To enable a technology object, the actuator driver must be initialized. Execute the command again later.
16#863C		Communication with encoder is faulty	
	16#004C	Configuration error at device	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004D	The device driver needs a smaller servo clock.	• Connect a suitable device. • Check the device (I/Os). • Compare the configuration of HW Config and the technology object.
	16#004E	Error in internal communication with the device	Check the project for consistency and reload it into the controller.
16#863D		Communication to the device (drive or encoder) is faulty	
	16#0030	Value has an incorrect number format or is outside the valid number range	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0055	The requested logical address is invalid.	• Connect a suitable device. • Check the device (I/Os). • Check the topological configuration in HW Config. • Compare the configuration of HW Config and the technology object.
	16#0056	The requested logical output address is invalid.	
	16#0057	The requested logical output address is invalid.	

ErrorID	ErrorInfo	Description	Remedy
16#863E		Value of tag "ControlPanel.Input.TimeOut" is invalid (axis control panel)	
	16#0030	Value has an incorrect number format or is outside the valid number range	Correct the value in the variables of the technology object <Axis name>.ControlPanel.Input.TimeOut. The value is specified in milliseconds (ms).
16#863F		Invalid value for the configuration date <Axis name>.Actor.DriveParameter.MaxSpeed	
	16#0030	Value has an incorrect number format or is outside the valid number range	Correct the reference value in the drive and in the configuration of the technology object to Actuator.MaxSpeed/2. With analog drive connection, correct the reference value in the drive and in the configuration of the technology object to Actuator.MaxSpeed/1.17.

Configuration error of the command table

ErrorID	ErrorInfo	Description	Remedy
16#8700		Value for "Command type" in the command table is invalid	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
16#8701		Value for "Position / travel path" in the command table is invalid	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1E+12)	
	16#0006	Value is outside the number range (less than -1E+12)	
16#8702		Value for "Velocity" in the command table is invalid	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
16#8703		Value for "Duration" in the command table is invalid	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0021	Value is greater than 64800 s	
	16#0022	Value is less than 0.001 s	
16#8704		Value for "Next step" in the command table is invalid	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0023	The command transition is not permitted for this command	

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		Internal error	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

Error displays of the Motion Control statements (Page 1443)

Tags of the positioning axis technology object V4...5 (Page 1595)

ErrorIDs and ErrorInfos (Page 1549)

Using multiple axes with the same PTO (Page 1452)

Using multiple drives with the same PTO (Page 1455)

Tracking jobs from higher priority classes (execution levels) (Page 1456)

Special cases when using software limit switches for drive connection via PTO (Page 1458)

List of ErrorIDs and ErrorInfos (technology objects V2...3) (S7-1200)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Operating error with axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8000		Drive error, loss of "Drive ready"	
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; restart command, if necessary
16#8001		Low SW limit switch has been tripped	
	16#000E	The position of the low SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the SW limit switch
	16#000F	The position of the low SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the low SW limit switch was exceeded with the emergency stop deceleration	
16#8002		High SW limit switch has been tripped	

ErrorID	ErrorInfo	Description	Remedy
	16#000E	The position of the high SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the SW limit switch
	16#000F	The position of the high SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the high SW limit switch was exceeded with the emergency stop deceleration	
16#8003		**Low HW limit switch was reached**	
	16#000E	The low HW limit switch was reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the HW limit switch.
16#8004		**High HW limit switch was reached**	
	16#000E	The high HW limit switch has been reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the HW limit switch.
16#8005		**PTO/HSC are already being used by another axis**	
	16#0001	-	**The axis was configured incorrectly:** Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller
			More than one axis is to run with one PTO: Another axis is using the PTO / HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 1452))
16#8006		**A communication error in the control panel has occurred**	
	16#0012	A timeout has occurred	Check the cable connection and press the "Manual control" button again
16#8007		**The axis cannot be enabled**	
	16#0025	Restarting	Wait until the axis restart is complete.
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.

Operating error without axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8200		**Axis is not enabled**	
	16#0001	-	Enable the axis; restart the command
16#8201		**Axis has already been enabled by another "MC_Power" instance**	
	16#0001	-	Enable the axis with only one "MC_Power" instance
16#8202		**The maximum number of simultaneously active Motion Control commands has been exceeded (maximum of 200 commands for all motion control technology objects)**	

ErrorID	ErrorInfo	Description	Remedy
	16#0001	-	Reduce the number of simultaneously active commands; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		**Axis is currently operated in "Manual control" (axis control panel)**	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		**Axis is not homed**	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		**The axis is currently controlled by the user program (the error is only displayed in the axis control panel)**	
	16#0013	The axis is enabled in the user program	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis control panel
16#8206		**Technology object not activated yet**	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis control panel.
16#8207		**Command rejected**	
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt".
	16#0018	The axis cannot be moved with a command table while it is being actively or passively homed.	Wait until direct or passive homing is complete.
	16#0019	The axis cannot be actively or passively homed while a command table is being processed.	Wait for command table to finish or abort the command table with a motion command, for example, "MC_Halt".
16#8208		**Difference between maximum and start/stop velocity is invalid**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#8209		**Invalid acceleration for technology object "Axis"**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#820A		**It is not possible to restart the axis**	
	16#0013	The axis is enabled in the user program	Disable the axis with the "MC_Power" instruction; restart again
	16#0027	The axis is currently being operated in "Manual control" (axis control panel)	Exit "Manual control"; restart again
16#820B		**It is not possible to execute the command table**	
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#0005	Value is outside the number range (greater than 1.0E+12)	
	16#0006	Value is outside the number range (less than -1.0E+12)	
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#0005	Value is outside the number range (greater than 1.0E+12)	
	16#0006	Value is outside the number range (less than -1.0E+12)	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
	16#0024	Value is less than 0	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	The direction reversal is activated at the hardware limit switch, despite the fact that the hardware limit switches are disabled	• Activate the hardware limit switch using the tag <Axis>.Config.PositionLimits_HW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
16#8405		**Invalid value at parameter "StopMode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; enable the axis again
16#8406		**Simultaneous forward and backward jogging is not allowed**	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		**Switching to another axis with instruction "MC_Power" is only permitted after disabling the active axis.**	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.
16#8408		**Invalid value at parameter "Axis" of the Motion Control instruction**	

ErrorID	ErrorInfo	Description	Remedy
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#8409		Invalid value at parameter "CommandTable" of the Motion Control instruction	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#840A		Invalid value at parameter "StartStep" of the Motion Control instruction	
	16#000A	Value is less than or equal to 0.	Correct the value; restart the command
	16#001D	The start step is greater than the end step	
	16#001E	Value is greater than 32	
16#840B		Invalid value at parameter "EndStep" of the Motion Control instruction	
	16#000A	Value is less than or equal to 0.	Correct the value; restart the command
	16#001E	Value is greater than 32	
16#840C		Invalid value at parameter "RampUpTime" of the Motion Control instruction	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840D		Invalid value at parameter "RampDownTime" of the Motion Control instruction	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840E		Invalid value at parameter "EmergencyRampTime" of the Motion Control instruction	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	
16#840F		Invalid value at parameter "JerkTime" of the Motion Control instruction	
	16#0002	The value has an invalid number format	Correct the value; restart the command
	16#000A	Value is less than or equal to 0.	

Configuration error of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8600		Parameter assignment of pulse generator (PTO is invalid	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8601		Parameter assignment of the high-speed counter (HSC) is invalid	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	

ErrorID	ErrorInfo	Description	Remedy
16#8602		Invalid parameter assignment of "Enable output"	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8603		Invalid parameter assignment of "Ready input"	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8604		Invalid "Pulses per motor revolution" value	
	16#000A	Value is less than or equal to zero	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8605		Invalid "Distance per revolution" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0005	Value is outside the number range (greater than 1.0E+12)	
	16#000A	Value is less than or equal to zero	
16#8606		Invalid "Start/stop velocity" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value is higher than the high hardware limit	
	16#0004	Value is lower than the low hardware limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		Invalid "maximum velocity" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value is higher than the high hardware limit	
	16#0004	Value is lower than the low hardware limit	
16#8608		Invalid "Acceleration" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the high hardware limit	
	16#0004	Value is lower than the low hardware limit	
16#8609		Invalid "Deceleration" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the high hardware limit	
	16#0004	Value is lower than the low hardware limit	
16#860A		Invalid "Emergency stop deceleration" value	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value is higher than the high hardware limit	
	16#0004	Value is lower than the low hardware limit	
16#860B		Value for position of the low SW limit switch is invalid	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1.0E+12)	
	16#0006	Value is outside the number range (less than -1.0E+12)	
	16#0007	The position value of the low software limit switch is greater than that of the high software limit switch	
16#860C		Value for position of the high SW limit switch is invalid	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1.0E+12)	
	16#0006	Value is outside the number range (less than -1.0E+12)	
16#860D		Invalid address of the low HW limit switch	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#860E		Invalid address of the high HW limit switch	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#860F		Invalid "home position offset" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1.0E+12)	
	16#0006	Value is outside the number range (less than -1.0E+12)	
16#8610		Invalid "approach velocity" value	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
16#8611		Invalid "Homing velocity" value	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
16#8612		**Invalid address of the homing switch**	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#8613		**During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured**	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8614		**Invalid "Jerk" value**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#001F	Value is greater than the maximum jerk	
	16#0020	Value is less than the minimum jerk	
16#8615		**Value for "Unit of measurement" is invalid**	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"

Configuration error of the command table

ErrorID	ErrorInfo	Description	Remedy
16#8700		**Value for "Command type" in the command table is invalid**	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
16#8701		**Value for "Position / travel path" in the command table is invalid**	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0005	Value is outside the number range (greater than 1.0E+12)	
	16#0006	Value is outside the number range (less than -1.0E+12)	
16#8702		**Value for "Velocity" in the command table is invalid**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
16#8703		Value for "Duration" in the command table is invalid	
	16#0002	The value has an invalid number format	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0021	Value is greater than 64800 s	
	16#0022	Value is less than 0.001 s	
16#8704		Value for "Next step" in the command table is invalid	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0023	The command transition is not permitted for this command	

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		Internal error	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

List of ErrorIDs and ErrorInfos (technology objects V4...5) (Page 1549)

Using multiple axes with the same PTO (Page 1452)

List of ErrorIDs and ErrorInfos (technology objects V1) (S7-1200)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Operating error with axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8000		**Drive error, loss of "Drive ready"**	
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; restart command, if necessary
16#8001		**Lower SW limit switch has been tripped**	
	16#000E	The position of the lower SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the SW limit switch
	16#000F	The position of the lower SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the lower SW limit switch was exceeded with the emergency stop deceleration	
16#8002		**Upper SW limit switch has been tripped**	
	16#000E	The position of the upper SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the SW limit switch
	16#000F	The position of the upper SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the upper SW limit switch was exceeded with the emergency stop deceleration	
16#8003		**Lower HW limit switch was reached**	
	16#000E	The lower HW limit switch was reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the reference point switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the HW limit switch.
16#8004		**Upper HW limit switch was reached**	
	16#000E	The upper HW limit switch has been reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the reference point switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the HW limit switch.
16#8005		**PTO/HSC are already being used by another axis**	
	16#0001	-	**The axis was configured incorrectly:** Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller
			More than one axis is to run with one PTO: Another axis is using the PTO / HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 1452))

Operating error without axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8200		**Axis is not enabled**	
	16#0001	-	Enable the axis; restart the command
16#8201		**Axis has already been enabled by another "MC_Power" instance**	
	16#0001	-	Enable the axis with only one "MC_Power" instruction
16#8202		**The maximum number of simultaneously active Motion Control commands was exceeded (maximum of 200 commands for all motion control technology objects)**	
	16#0001	-	Reduce the number of simultaneously active commands; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		**Axis is currently operated in "Manual control" (axis control panel)**	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		**Axis is not homed**	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		**The axis is currently controlled by the user program (the error is only displayed in the axis control panel)**	
	16#0001	-	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis control panel
16#8206		**Technology object Axis not yet enabled**	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis control panel.
16#8207		**Command rejected**	
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt". The other homing type can then be started.

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "position" value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Number format of value is invalid	Correct the "Distance" value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "Velocity" value; restart the command
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; restart the command
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	Axis reversal is activated at the HW limit switch, despite the fact that the hardware limit switches are disabled	• Activate the hardware limit switch using the tag <axis>.Config.PositionLimits_HW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
16#8405		**Invalid value at parameter "StopMode" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; enable the axis again
16#8406		**Simultaneous forward and backward jogging is not allowed**	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		**Switching the axis with Motion Control instruction "MC_Power" is only permitted after disabling the axis.**	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.

Configuration error

ErrorID	ErrorInfo	Description	Remedy
16#8600		**Parameter assignment of pulse generator (PTO is invalid**	
	16#000B	Address is invalid	Correct the configuration of the PTO (Pulse Train Output) and download it to the controller
16#8601		**Parameter assignment of the high-speed counter (HSC) is invalid**	
	16#000B	Address is invalid	Correct the configuration of the HSC (High Speed Counter) and download it to the controller
16#8602		**Invalid parameter assignment of "Enable output"**	
	16#000D	Address is invalid	Correct the configuration and download it to the controller
16#8603		**Invalid parameter assignment of "Ready input"**	

ErrorID	ErrorInfo	Description	Remedy
	16#000D	Address is invalid	Correct the configuration and download it to the controller
16#8604		**Invalid "Pulses per motor revolution" value**	
	16#000A	Value is less than or equal to zero	Correct the configuration and download it to the controller
16#8605		**Invalid "Distance per revolution" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#000A	Value is less than or equal to zero	
16#8606		**Invalid "Start/stop velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		**Invalid "Maximum velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#8608		**Invalid "Acceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#8609		**Invalid "Deceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#860A		**Invalid "Emergency stop deceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#860B		**Value for position of the lower SW limit switch is invalid**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
	16#0007	The position value of the lower SW limit switch is greater than that of the upper SW limit switch	
16#860C		**Value for position of the upper SW limit switch is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#860D		**Invalid address of the lower HW limit switch**	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	
16#860E		**Invalid address of the upper HW limit switch**	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	
16#860F		**Invalid "home position offset" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8610		**Invalid "approach velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8611		**Invalid "Homing velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8612		**Invalid address of the reference point switch**	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	
16#8613		**During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured**	
	16#0001	-	Correct the configuration and download it to the controller

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		Internal error	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

List of ErrorIDs and ErrorInfos (technology objects V4...5) (Page 1549)

Using multiple axes with the same PTO (Page 1452)

Legend V1...5 (S7-1200)

Tag	Name of the tag	
Data type	Data type of the tag	
Values	Start value (Value range of the tag - minimum value to maximum value) If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".	
Access	Access to the tag in the user program	
	R	The tag can be read in the user program and in the HMI.
	RP	The variable can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding variables is determined at the start of the command.
	RW	The tag can be read and written in the user program and in the HMI. The variable can be written with Motion Control instruction "MC_WriteParam".
	RCCP	The tag can be read in the user program and in the HMI and is updated at each cycle control point.
	WP	Independent of the drive connection: If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	WP_PD	For drive connection via PROFIdrive/analog output: If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	WP_PTO	For drive connection via PTO: If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	WD BL	For drive connection via PROFIdrive/analog output: The tag can be written to the start value in the load memory with the extended instruction "WRIT_DBL".
	-	The variable cannot be used in the user program.

W	Effectiveness of changes in the technology data block	
	1	For drive connection via PTO: When axis is activated, disabled, or enabled
	2	For drive connection via PTO: When axis is enabled
	3	After axis enable (the axis must have previously been at a standstill). The axis standstill can be checked with tag <axis name>.StatusBits.Standstill.
	4	Upon the next start of a Motion Control command after a standstill of the axis. The axis standstill can be checked with tag <axis name>.StatusBits.Standstill.
	5	For drive connection via PTO: The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable, or active MC_Home command is started (Mode = 3)
	6	For drive connection via PTO: When a MC_MoveJog command is stopped
	7	For drive connection via PTO: When a passive homing command is started
	8	For drive connection via PTO: When an active homing command is started
	9	With the restart of the technology object
	10	For drive connection via PROFIdrive/analog output: With the next call of the MC-Servo [OB91]
Description	Description of the tag	

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Tag of the axis technology object V1...3 (S7-1200)

Config tags V1...3 (S7-1200)

The tag structure <axis name>.Config.<tag name> contains the configuration of the dynamic limits.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
Config.	STRUCT				TO_Struct_Config

Tag	Data type	Values	Access	W	Description	
General.	STRUCT				TO_Struct_General	
PTO	DWORD	DW#16#000 00000	-	-	The tag cannot be evaluated in the user program.	
HSC	DWORD	DW#16#000 00000	-	-	The tag cannot be evaluated in the user program.	
LengthUnit[2]	INT	1013 (-32768 to 32767)	R WP	- 2	Configured unit of measurement of the parameter	
					-1	Pulses
					1005	°
					1010	m
					1013	mm
					1018	ft
					1019	in
DriveInterface.	STRUCT				TO_Struct_DriveInterface	
EnableOutput	-	-	-	-	The tag cannot be evaluated in the user program.	
ReadyInput	-	-	-	-	The tag cannot be evaluated in the user program.	
Mechanics.	STRUCT				TO_Struct_Mechanics	
PulsesPerDriveRevolution	DINT	L#1000	R	-	Pulses per motor revolution	
LeadScrew	REAL	10.0	R	-	Distance per revolution (indicated in the configured unit of measurement)	
InverseDirection	BOOL	FALSE	R	-	Invert direction	
DynamicLimits.	STRUCT				TO_Struct_DynamicLimits	
MinVelocity	REAL	10.0	R	-	Start/stop velocity of the axis (indicated in the configured unit of measurement)	
MaxVelocity	REAL	250.0	R	-	Maximum velocity of the axis (indicated in the configured unit of measurement)	
DynamicDefaults.	STRUCT				TO_Struct_DynamicDefaults	
Acceleration	REAL	48.0	RW	1[2], 5, 6[2]	Acceleration of the axis (indicated in the configured unit of measurement)	
Deceleration	REAL	48.0	RW	1[2], 5, 6	Deceleration of the axis (indicated in the configured unit of measurement)	
EmergencyDeceleration	REAL	120.0	RW	1[2], 2[1], 5, 6	Emergency stop deceleration of the axis (indicated in the configured unit of measurement)	
JerkActive[2]	BOOL	FALSE	RW	1, 5	FALSE	The jerk limit is disabled.
					TRUE	The jerk limit is enabled
Jerk[2]	REAL	192.0	RW	1, 5	Jerk during acceleration and deceleration ramp of the axis	

Tag	Data type	Values	Access	W	Description	
					(indicated in the configured unit of measurement)	
PositionLimits_SW.	STRUCT				TO_Struct_PositionLimits_SW	
Active	BOOL	FALSE	RW	1[2], 4[1], 5[2], 6[2]	FALSE	The software limit switches are deactivated.
					TRUE	The software limit switches are activated.
MinPosition	REAL	-1.0E4	RW	1[2], 4[1], 5[2], 6[2]	Position of the software low limit switch (indicated in the configured unit of measurement)	
MaxPosition	REAL	1.0E4	RW	1[2], 4[1], 5[2], 6[2]	Position of the software high limit switch (indicated in the configured unit of measurement)	
PositionLimits_HW.					TO_Struct_PositionLimits_HW	
Active	BOOL	FALSE	RW	1[2], 3[1], 4[1], 5[2], 6[2]	FALSE	The hardware limit switches are deactivated.
					TRUE	The hardware limit switches are activated.
MinSwitchedLevel	BOOL	FALSE	R	-	FALSE	0 V at CPU input corresponds to hardware low limit switch approached
					TRUE	24 V at CPU input corresponds to the hardware low limit switch approached.
MinFallingEvent	DWORD	DW#16#00000000	-	-	The tag cannot be evaluated in the user program.	
MinRisingEvent	DWORD	DW#16#00000000	-	-	The tag cannot be evaluated in the user program.	
MaxSwitchedLevel	BOOL	FALSE	R	-	FALSE	0 V at CPU input corresponds to hardware high limit switch approached
					TRUE	24 V at CPU input corresponds to the hardware high limit switch approached.
MaxFallingEvent	DWORD	DW#16#00000000	-	-	The tag cannot be evaluated in the user program.	
MaxRisingEvent	DWORD	DW#16#00000000	-	-	The tag cannot be evaluated in the user program.	
Homing.	STRUCT				TO_Struct_Homing	
AutoReversal	BOOL	TRUE	R[1] RW[2]	-[1] 1[2], 8[2]	FALSE	Direction reversal at hardware limit switch disabled (active homing)
					TRUE	Direction reversal at hardware limit switch enabled (active homing)
Direction	BOOL	TRUE	R[1] RW[2]	-[1] 1[2], 8[2]	FALSE	Negative approach direction to search for reference point switch

Tag	Data type	Values	Access	W	Description	
						and positive homing direction (active homing)
					TRUE	Positive approach direction to search for reference point switch and positive homing direction (active homing)
SideActiveHoming[2]	BOOL	TRUE	RW	1, 8	FALSE	Homing on low end of the reference point switch (active homing)
					TRUE	Homing on high end of the reference point switch (active homing)
SidePassiveHoming[2]	BOOL	TRUE	RW	1, 7	FALSE	Homing on low end of the reference point switch (passive homing)
					TRUE	Homing on high end of the reference point switch (passive homing)
RisingEdge[1]	BOOL	FALSE	R	-	FALSE	Homing with positive signal edge of the reference point switch (active homing)
					TRUE	Homing with negative signal edge of the reference point switch (active homing). For information on the effect of the tag on passive homing, refer to the description in "Configuration - Homing".
Offset	REAL	0.0	R RW	-[1] 1[2], 8[2]	Homing position offset / specification in the configured unit of measurement (active homing)	
FastVelocity	REAL	200.0	R RW	-[1] 1[2], 8[2]	Approach velocity / specification in the configured unit of measurement (active homing)	
SlowVelocity	REAL	40.0	R RW	-[1], 1[2], 8[2]	Homing velocity / specification in the configured unit of measurement (active homing)	
FallingEvent	DWORD	DW#16#00000000	-	-	The tag cannot be evaluated in the user program.	
RisingEvent	DWORD	DW#16#00000000	-	-	The tag cannot be evaluated in the user program.	

1) Applies to technology version V1.0 only

2) Applies as of technology version V2.0

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tags of the positioning axis technology object V4...5 (Page 1595)

MotionStatus tags V1...3 (S7-1200)

The tag structure <axis name>.MotionStatus.<tag name> contains the status of the motion.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
MotionStatus.	STRUCT				TO_Struct_MotionStatus
Position	REAL	0.0 (-9.0E15 to 9.0E15)	R	-	Actual position of the axis (indicated in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis.
Velocity	REAL	0.0 (-9.0E15 to 9.0E15)	R	-	Actual velocity of the axis (indicated in the configured unit of measurement)
Distance	REAL	0.0 (-9.0E15 to 9.0E15)	R	-	Current distance to the target position of the axis (indicated in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute" or "MC_MoveRelative" or of the axis command table.
TargetPosition	REAL	0.0 (-9.0E15 to 9.0E15)	R	-	Target position of the axis (indicated in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute" or "MC_MoveRelative" or of the axis command table.

See also

Motion status (Page 1449)

Tags of the positioning axis technology object as of V6 (Page 1488)

Tags of the positioning axis technology object V4...5 (Page 1595)

StatusBits tags V1...3 (S7-1200)

The tag structure <axis name>.StatusBits.<tag name> contains the status information of the technology object.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
StatusBits.	STRUCT				TO_Struct_StatusBits

Tag	Data type	Values	Access	W	Description	
Activated	BOOL	FALSE	R	-	Activation of the axis	
					FALSE	The axis is not activated.
					TRUE	The axis is activated. It axis is connected to the assigned PTO (Pulse Train Output). The data of the technology data block will be updated cyclically.
Enable	BOOL	FALSE	R	-	Enable status of the axis	
					FALSE	The axis is not enabled.
					TRUE	The axis is enabled and ready to accept Motion Control commands.
HomingDone	BOOL	FALSE	R	-	Homing status of the axis	
					FALSE	The axis is not homed.
					TRUE	The axis is homed and is capable of executing absolute positioning commands.
					The axis does not have to be homed for relative positioning. During active homing, the status is FALSE. The status remains TRUE during passive homing if the axis was already homed beforehand.	
Done	BOOL	FALSE	R	-	Command execution on the axis	
					FALSE	A Motion Control command is active on the axis.
					TRUE	A Motion Control command is not active on the axis.
Error	BOOL	FALSE	R	-	Error status on the axis	
					FALSE	No error is active on the axis.
					TRUE	An error has occurred on the axis.
					Additional information about the error is available in automatic mode at the "ErrorID" and "ErrorInfo" parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error.	
Standstill	BOOL	FALSE	R	-	Standstill of the axis	
					FALSE	The axis is in motion.
					TRUE	The axis is at a standstill.
PositioningCommand	BOOL	FALSE	R	-	Execution of a positioning command	
					FALSE	A positioning command is not active on the axis.
					TRUE	The axis executes a positioning command.
SpeedCommand	BOOL	FALSE	R	-	Execution of a command with velocity specification	

Tag	Data type	Values	Access	W	Description	
					FALSE	A command with velocity specification is not active on the axis.
					TRUE	The axis executes a travel command at predefined velocity.
Homing	BOOL	FALSE	R	-	Execution of a homing command	
					FALSE	A homing command is not active on the axis.
					TRUE	The axis executes a homing command of the Motion Control instruction "MC_Home" or the axis control panel.
CommandTableActive	BOOL	FALSE	R	-	Execution of a command table	
					FALSE	A command table is not active on the axis.
					TRUE	The axis is controlled by Motion Control instruction "MC_CommandTable".
ConstantVelocity	BOOL	FALSE	R	-	Constant velocity	
					FALSE	The axis is accelerating, decelerating, or at a standstill.
					TRUE	The setpoint velocity has been reached. The axis is moving at constant velocity.
Acceleration	BOOL	FALSE	R	-	Acceleration process	
					FALSE	The axis is decelerating, moving at constant velocity, or is at a standstill.
					TRUE	Axis is being accelerated.
Deceleration	BOOL	FALSE	R	-	Deceleration process	
					FALSE	The axis is accelerating, moving at constant velocity, or is at a standstill.
					TRUE	The axis is being decelerated.
ControlPanelActive	BOOL	FALSE	R	-	Activation status of the axis control panel	
					FALSE	"Automatic mode" is activated. The user program has control priority over the axis.
					TRUE	The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the axis. The axis cannot be controlled from the user program.
DriveReady	BOOL	FALSE	R	-	Operational status of the drive	
					FALSE	The drive is not ready. Setpoints will not be executed.
					TRUE	The drive is ready. Setpoints can be executed.

Tag	Data type	Values	Access	W	Description
RestartRequired	BOOL	FALSE	R	-	Restart of axis required FALSE: A restart of the axis is not required. TRUE: Values were modified in the load memory. To download the values to the work memory while the CPU is in RUN mode, the axis must be restarted. Use the MC_Reset Motion Control instruction to do this.

See also

Status and error bits (technology objects as of V4) (Page 1447)
Tags of the positioning axis technology object as of V6 (Page 1488)
Tags of the positioning axis technology object V4...5 (Page 1595)

ErrorBits tags V1...3 (S7-1200)

The tag structure <axis name>.ErrorBits.<tag name> indicates error at the technology object.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
ErrorBits.	STRUCT				TO_Struct_ErrorBits
SystemFault	BOOL	FALSE	R	-	Internal system error
ConfigFault	BOOL	FALSE	R	-	Incorrect configuration of the axis
DriveFault	BOOL	FALSE	R	-	Error in the drive. Loss of the "Drive ready" signal.
SWLimitMinReached	BOOL	FALSE	R	-	The software low limit switch has been reached.
SWLimitMinExceeded	BOOL	FALSE	R	-	The software low limit switch has been exceeded.
SWLimitMaxReached	BOOL	FALSE	R	-	The software high limit switch has been reached.
SWLimitMaxExceeded	BOOL	FALSE	R	-	The software high limit switch has been exceeded.
HWLimitMin	BOOL	FALSE	R	-	The hardware low limit switch has been approached.
HWLimitMax	BOOL	FALSE	R	-	The hardware high limit switch has been approached.
HWUsed	BOOL	FALSE	R	-	Another axis is using the same PTO (Pulse Train Output) and is enabled with "MC_Power".

See also

Status and error bits (technology objects as of V4) (Page 1447)

Tags of the positioning axis technology object as of V6 (Page 1488)

Tags of the positioning axis technology object V4...5 (Page 1595)

Internal tags V1...3 (S7-1200)

The "Internal" tags contain no user-relevant data; these tags cannot be accessed in the user program.

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tags of the positioning axis technology object V4...5 (Page 1595)

ControlPanel tags V1...3 (S7-1200)

The "ControlPanel" tags contain no user-relevant data; these tags cannot be accessed in the user program.

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tags of the positioning axis technology object V4...5 (Page 1595)

Update of the technology object tags (S7-1200)

The status and error information of the axis indicated in the technology object tags is updated at each cycle control point.

The change in values of editable configuration tags does not take effect immediately. For information on the conditions under which a change takes effect, refer to the detailed description of the relevant tag.

Tags of the positioning axis technology object V4...5 (S7-1200)

Tags to position values and velocity values V4...5 (S7-1200)

The tag structure contains the setpoint and actual values of the position and the velocity of the axis.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
Position	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP	-	Setpoint position of the axis (indicated in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis.
Velocity	REAL	0.0	RCCP, RP	-	Velocity setpoint of the axis (indicated in the configured unit of measurement)
ActualPosition	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP	-	Actual position of the axis (indicated in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis.
ActualVelocity	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP	-	Actual velocity of the axis (indicated in the configured unit of measurement)

See also

Motion status (Page 1449)

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

Actuator tags V4...5 (S7-1200)

The tag structure <axis name>.Actor.<tag name> contains the drive parameters.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
Actor.	STRUCT				TO_Struct_Actor

Tag	Data type	Values	Access	W	Description	
Type	DINT	1 (0 to 2)	R	-	Positioning axis technology object as of V5:	
					0	The drive is connected via an analog output. All movements of the axis are position-controlled.
					1	The drive is connected via PROFIdrive telegrams. All movements of the axis are position-controlled.
					2	The drive is connected via a pulse interface.
					Positioning axis technology object V4:	
					2	The drive is connected via a pulse interface.
InverseDirection	BOOL	FALSE	R WP_PTO	- 2	FALSE	The direction is not inverted.
					TRUE	The direction is inverted.
DirectionMode	INT	0 (0 to 2)	R WP_PTO	- 2	Permitted direction of rotation	
					0	Both directions
					1	Positive direction
					2	Negative direction
Interface.	STRUCT				TO_Struct_ActorInterface	
AddressIn[3)]	VREF	-	-	-	Input address for the PROFIdrive telegram (internal parameter)	
AddressOut[3)]	VREF	-	-	-	Output address for the PROFIdrive telegram (internal parameter)	
EnableDriveOutput	VREF	-	-	-	Enable output (internal parameter)	
DriveReadyInput	VREF	-	-	-	Ready input (internal parameter)	
PTO	DWORD	0	-	-	Pulse output (internal parameter)	
DriveParameter.	STRUCT				TO_Struct_ActorDriveParameter	
ReferenceSpeed[3)]	REAL	3000.0	R	-	Reference value (100%) for the speed setpoint (N-set) of the drive The speed setpoint is transferred in the PROFIdrive telegram as a normalized value from -200% to 200% of the "ReferenceSpeed". For setpoint specification via an analog output, the analog output can be operated in the range from -117% to 117% if permitted by the drive permits.	
MaxSpeed[3)]	REAL	3000.0	R	-	Maximum value for the speed setpoint of the drive (N-set) (PROFIdrive: MaxSpeed ≤ 2 × ReferenceSpeed Analog setpoint: MaxSpeed ≤ 1.17 × ReferenceSpeed)	
PulsesPerDriveRevolution	DINT	1000 (1 to 2147483648)	R WP_PTO	- 2	Pulses per motor revolution	

3) Applies as of technology version V5.0

See also

Tag of the axis technology object V1...3 (Page 1584)

Tags of the positioning axis technology object as of V6 (Page 1488)

Sensor[1] tags V4...5 (S7-1200)

The tag structure <axis name>.Sensor.<tag name> contains the encoder parameters.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
Sensor.	STRUCT				TO_Struct_Sensor

Tag	Data type	Values	Access	W	Description	
Sensor[1].	STRUCT				ARRAY[1..1] TO_Struct_Sensor	
Type	DINT	0 (0 to 1)	R	-	Encoder type (internal parameter)	
					0	Incremental
					1	Absolute
InverseDirection	BOOL	FALSE	R	-	Inversion of the actual value	
					FALSE	Actual value is not inverted
					TRUE	Actual value is inverted
System	DINT	1 (0 to 1)	R	-	Encoder system	
					0	Linear encoder
					1	Rotary encoder
MountingMode	DINT	0 (0 to 2)	R	-	Mounting type of encoder	
					0	On the motor shaft
					2	External measuring system
Interface.	STRUCT				TO_Struct_SensorInterface	
AddressIn.	VREF	-	-	-	Input address for the PROFIdrive telegram (internal parameter)	
AREA	BYTE	-	-	-	Internal parameter	
DB_NUMBER	UINT	-	-	-	Internal parameter	
OFFSET	UDINT	-	-	-	Internal parameter	
RID	DWORD	-	-	-	Internal parameter	
AddressOut.	VREF	-	-	-	Output address for the PROFIdrive telegram (internal parameter)	
AREA	BYTE	-	-	-	Internal parameter	
DB_NUMBER	UINT	-	-	-	Internal parameter	
OFFSET	UDINT	-	-	-	Internal parameter	
RID	DWORD	-	-	-	Internal parameter	
Parameter.	STRUCT				TO_Struct_SensorParameter	
Resolution	REAL	0.001 (-1.0E12 to 1.0E12)	R	-	Resolution of a linear encoder (offset between two encoder pulses)	
StepsPerRevolution	UDINT	2048 (1 to 8388608)	R	-	Increments per rotary encoder revolution	
FineResolutionXist1	UDINT	11 (0 to 31)	R	-	Number of bits for fine resolution Gn_XIST1 (cyclic actual encoder value)	
FineResolutionXist2	UDINT	9 (0 to 31)	R	-	Number of bits for fine resolution Gn_XIST2 (absolute value of the encoder)	
DeterminableRevolutions	UDINT	1 (0 to 8388608)	R	-	Number of differentiable encoder revolutions for a multi-turn absolute encoder	
					0	Incremental encoder
					1	Single return absolute encoder
DistancePerRevolution	REAL	100.0	R	-	Load distance per revolution of an externally mounted encoder	

Tag	Data type	Values	Access	W	Description	
		(0.0 to 1.0E12)				
ActiveHoming.	STRUCT				TO_Struct_SensorActiveHoming	
Mode	DINT	2 (0 to 2)	R WP_PTO	- 2	Active homing mode	
					Positioning axis technology object as of V5:	
					0	Zero mark via PROFIdrive telegram (not PTO)
					1	Zero mark via PROFIdrive telegram and proximity switch (not PTO)
					2	Homing via digital input
					Positioning axis technology object V4:	
					2	Homing via digital input
SideInput	BOOL	FALSE	RW	1, 8, 10	End of the homing switch to which the axis is homed during active homing	
					FALSE	Bottom side
					TRUE	Top side
DigitalInputAddress.	VREF	-	-	-	Symbolic input address of the homing switch (internal parameter)	
AREA	BYTE	-	-	-	Internal parameter	
DB_NUMBER	UINT	-	-	-	Internal parameter	
OFFSET	UDINT	-	-	-	Internal parameter	
RID	DWORD	-	-	-	Internal parameter	
HomePositionOffset	REAL	0.0 (-1.0E12 to 1.0E12)	RW	1, 8, 10	Home position offset (active homing) (indicated in the configured unit of measurement)	
SwitchLevel	BOOL	TRUE	RW	1, 8, 10	Selection of signal level that is present at the CPU input when the homing switch is reached	
					FALSE	Low level (Low active)
					TRUE	High level (high-enabled)
PassiveHoming.	STRUCT				TO_Struct_SensorPassiveHoming	
Mode	DINT	2 (0 to 2)	R WP_PTO	- 2	Passive homing mode	
					Positioning axis technology object as of V5:	
					0	Zero mark via PROFIdrive telegram (not PTO)
					1	Zero mark via PROFIdrive telegram and proximity switch (not PTO)
					2	Homing via digital input
					Positioning axis technology object V4:	
					2	Homing via digital input
SideInput	BOOL	FALSE	RW	1, 7, 10	End of the homing switch to which the axis is homed during passive homing	
					FALSE	Bottom side
					TRUE	Top side
DigitalInputAddress.	VREF	-	-	-	Symbolic input address of the homing switch (internal parameter)	

Tag	Data type	Values	Access	W	Description	
AREA	BYTE	-	-	-	Internal parameter	
DB_NUMBER	UINT	-	-	-	Internal parameter	
OFFSET	UDINT	-	-	-	Internal parameter	
RID	DWORD	-	-	-	Internal parameter	
SwitchLevel	BOOL	TRUE	RW	1, 7, 10	Selection of level that is present at the CPU input when the homing switch is reached	
					FALSE	Low level (Low active)
					TRUE	High level (high-enabled)

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

Units tag V4...5 (S7-1200)

The tag structure <axis name>.Units.LengthUnit contains the configured units of measurement of the parameters.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description	
Units.	STRUCT				TO_Struct_Units	
LengthUnit	INT	1013 (-32768 to 32767)	R WP_PTO	- 2	Configured unit of measurement of the parameter	
					-1	Pulses
					1005	°
					1010	m
					1013	mm
					1018	ft
					1019	in

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

Mechanics tag V4...5 (S7-1200)

The tag structure <axis name>.Mechanics.LeadScrew contains the distance traveled per motor revolution.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
Mechanics.	STRUCT				TO_Struct_Mechanics
LeadScrew	REAL	10.0 (-1.0E12 to 1.0E12)	R, WP_PTO	-	Distance per revolution (indicated in the configured unit of measurement)

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

Modulo tags V4...5 (S7-1200)

The tag structure <axis name>.Modulo.<tag name> contains the modulo settings.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description	
Modulo.	STRUCT				TO_Struct_Modulo	
Enable	BOOL	FALSE	R	-	FALSE	Modulo conversion disabled
					TRUE	Modulo conversion enabled When modulo conversion is enabled, a check is made for modulo length > 0.0
Length	REAL	360.0 (0.001 to 1.0E12)	R	-	Modulo length	
StartValue	REAL	0.0 (-1.0E12 to 1.0E12)	R	-	Modulo start value	

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

DynamicLimits tags V4...5 (S7-1200)

The tag structure <axis name>.DynamicLimits.<tag name> contains the configuration of the dynamic limits.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
DynamicLimits.	STRUCT				TO_Struct_DynamicLimits
MaxVelocity	REAL	250.0	R WP_PTO	- 2	Maximum velocity of the axis (indicated in the configured unit of measurement)
MinVelocity	REAL	10.0	R WP_PTO	- 2	Start/stop velocity of the axis (indicated in the configured unit of measurement)

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

DynamicDefaults tags V4...5 (S7-1200)

The tag structure <axis name>.DynamicDefaults.<tag name> contains the configuration of the dynamic defaults.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
DynamicDefaults.	STRUCT				TO_Struct_DynamicDefaults
Acceleration	REAL	48.0 (0.0 to 1.0E12)	RW	1, 5, 6, 10	Default setting of the acceleration of the axis (indicated in the configured unit of measurement)
Deceleration	REAL	48.0 (0.0 to 1.0E12)	RW	1, 5, 6, 10	Default decelaration of the axis (indicated in the configured unit of measurement)
Jerk	REAL	192.0 (0.0 to 1.0E12)	RW	1, 5, 10	Setting the jerk default during acceleration and deceleration ramp of the axis (indicated in the configured unit of measurement) The jerk is activated if the configured jerk is greater than 0.00004 mm/s².
EmergencyDeceleration	REAL	120.0 (0.0 to 1.0E12)	RW	1, 5, 6, 10	Emergency stop deceleration of the axis (indicated in the configured unit of measurement)

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

PositionLimitsSW tags V4...5 (S7-1200)

The tag structure <axis name>.PositionLimits_SW.<tag name> contains the configuration for position monitoring with software limit switches. Software limit switches are used to limit the operating range of a positioning axis.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description	
PositionLimits_SW.	STRUCT				TO_Struct_PositionLimitsSW	
Active	BOOL	FALSE	RW	1, 5, 6, 10	FALSE	The software limit switches are deactivated.
					TRUE	The software limit switches are activated.
MinPosition	REAL	-10000.0 (-1.0E12 to 1.0E12)	RW	1, 5, 6, 10	Position of the software low limit switch (indicated in the configured unit of measurement)	
MaxPosition	REAL	10000.0 (-1.0E12 to 1.0E12)	RW	1, 5, 6, 10	Position of the software high limit switch (indicated in the configured unit of measurement)	

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

PositionLimitsHW tags V4...5 (S7-1200)

The tag structure <axis name>.PositionLimits_HW.<tag name> contains the configuration for position monitoring with hardware limit switches. Hardware limit switches are used to limit the traversing range of a positioning axis.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description	
PositionLimits_HW.	STRUCT				TO_Struct_PositionLimitsHW	
Active	BOOL	FALSE	RW	1, 5, 6, 10	FALSE	The hardware limit switches are deactivated.
					TRUE	The hardware limit switches are activated.
MinSwitchLevel	BOOL	FALSE	R WP_PTO	- 2	Selection of signal level that is present at the CPU input when the hardware low limit switch is reached	
					FALSE	Low level (Low active)
					TRUE	High level (high-enabled)
MinSwitchAddress	VREF	-	-	-	Symbolic input address of the hardware low limit switch (internal parameter)	
MaxSwitchLevel	BOOL	FALSE	R WP_PTO	- 2	Selection of signal level that is present at the CPU input when the hardware high limit switch is reached	
					FALSE	Low level (Low active)
					TRUE	High level (high-enabled)
MaxSwitchAddress	VREF	-	-	-	Input address of the hardware high limit switch (internal parameter)	

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

Homing tags V4...5 (S7-1200)

The tag structure <axis name>.Homing.<tag name> contains the configuration for homing the axis.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description	
Homing.	STRUCT				TO_Struct_Homing	
AutoReversal	BOOL	FALSE	RW	1, 8, 10	FALSE	Auto reversal at the hardware limit switch is deactivated.
					TRUE	Auto reversal at the hardware limit switch is activated.
ApproachDirection	BOOL	TRUE	RW	1, 8, 10	FALSE	Negative approach direction for finding the homing switch and negative homing direction
					TRUE	Positive approach direction to search for reference point switch and positive homing direction
ApproachVelocity	REAL	200.0 (0.0 to 1.0E12)	RW	1, 8, 10	Approach velocity of the axis during active homing (indicated in the configured unit of measurement)	
ReferencingVelocity	REAL	40.0 (0.0 to 1.0E12)	RW	1, 8, 10	Homing velocity of the axis during active homing (indicated in the configured unit of measurement)	

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

PositionControl tags V5 (S7-1200)

The tag structure <axis name>.PositionControl.<tag name> contains the position control settings.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
PositionControl.	STRUCT				TO_Struct_PositionControl
Kv	REAL	10.0 (0.0 to 2147480.0)	R WP	- 10	Proportional gain of the closed loop position control ("Kv" > 0.0)
Kpc	REAL	100.0 (0.0 to 150.0)	R WP	- 10	Velocity precontrol of the position control as a percentage

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

FollowingError tags V5 (S7-1200)

The tag structure <axis name>.FollowingError.<tag name> contains the configuration of the dynamic following error monitoring.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description	
FollowingError.	STRUCT				TO_Struct_FollowingError	
EnableMonitoring	BOOL	TRUE	R	-	FALSE	Following error monitoring deactivated
					TRUE	Following error monitoring enabled
MinValue	REAL	10.0 (0.0 to 1.0E12)	R WP_PD	- 10	Permissible following error at velocities below the value of "MinVelocity".	
MaxValue	REAL	100.0 (0.0 to 1.0E12)	R WP_PD	- 10	Maximum permissible following error, which may be reached at the maximum velocity.	
MinVelocity	REAL	10.0 (0.0 to 1.0E12)	R WP_PD	- 10	"MinValue" is permissible below this velocity and is held constant.	

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

PositionMonitoring tags V5 (S7-1200)

The tag structure <axis name>.PositionMonitoring.<tag name> contains the configuration for position monitoring at the end of a positioning motion.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
PositionMonitoring.	STRUCT				TO_Struct_PositionMonitoring
ToleranceTime	REAL	1.0 (0.0 to 1.0E12)	R WP_PD	- 10	Tolerance time Maximum permitted duration from reaching velocity setpoint zero, until entrance into the positioning window.
MinDwellTime	REAL	0.1 (0.0 to 1.0E12)	R WP_PD	- 10	Minimum dwell time in positioning window
Window	REAL	1.0 (0.001 to 1.0E12)	R WP_PD	- 10	Positioning window

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

StandstillSignal tags V5 (S7-1200)

The tag structure <axis name>.StandstillSignal.<tag name> contains the configuration for the standstill signal.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
StandstillSignal.	STRUCT				TO_Struct_StandstillSignal
VelocityThreshold	REAL	5.0 (0.0 to 1.0E12)	R WP_PD	- 10	Velocity threshold If velocity is below this threshold, then the minimum dwell time begins.
MinDwellTime	REAL	0.01 (0.0 to 1.0E12)	R WP_PD	- 10	Minimum dwell time

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

StatusPositioning tags V4...5 (S7-1200)

The tag structure <axis name>.StatusPositioning.<tag name> indicates the status of a positioning motion.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
StatusPositioning.	STRUCT				TO_Struct_StatusPositioning
Distance	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP	-	Current distance of axis from target position (indicated in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute", "MC_MoveRelative", or the axis control panel.
TargetPosition	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP	-	Target position of the axis (indicated in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute", "MC_MoveRelative", or the axis control panel.
FollowingError	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP	-	Current following error of the axis (indicated in the configured unit of measurement) FollowingError = 0.0 for drive connection via PTO (Pulse Train Output).

See also

Motion status (Page 1449)

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

StatusDrive tags V5 (S7-1200)

The tag structure <axis name>.StatusDrive.<tag name> indicates the status of the drive.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description	
StatusDrive.	STRUCT				TO_Struct_StatusDrive	
InOperation	BOOL	FALSE	RCCP, RP	-	Operational status of the drive	
					FALSE	Drive not ready. Setpoints will not be executed.
					TRUE	Drive ready. Setpoints can be executed.
CommunicationOK	BOOL	FALSE	RCCP, RP	-	Cyclic BUS communication between controller and drive	
					FALSE	Communication not established
					TRUE	Communication established

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

StatusSensor tags V5 (S7-1200)

The tag structure <axis name>.StatusSensor.<tag name> indicates the status of the measuring system.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description	
StatusSensor.	STRUCT				TO_Struct_StatusSensor	
State	DINT	0 (0 to 2)	RCCP, RP	-	Status of the encoder value	
					0	Invalid
					1	Waiting for valid status
					2	Valid
CommunicationOK	BOOL	FALSE	RCCP, RP	-	Cyclic BUS communication between controller and encoder	
					FALSE	Communication not established
					TRUE	Communication established
AbsEncoderOffset	REAL	0.0 (-9.0E15 to 9.0E15)	RCCP, RP	-	Home point offset to the value of an absolute value encoder. The value will be retentively stored in the CPU.	

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

StatusBits tags V4...5 (S7-1200)

The tag structure <axis name>.StatusBits.<tag name> contains the status information of the technology object.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
StatusBits.	STRUCT				TO_Struct_StatusBits

Tag	Data type	Values	Access	W	Description	
Activated	BOOL	FALSE	RCCP, RP	-	Activation of the axis	
					FALSE	The axis is not activated.
					TRUE	The axis is activated. It axis is connected to the assigned PTO (Pulse Train Output). The data of the technology data block will be updated cyclically.
Enable	BOOL	FALSE	RCCP, RP	-	Enable status of the axis	
					FALSE	The axis is not enabled.
					TRUE	The axis is enabled and ready to accept Motion Control commands.
AxisSimulation	BOOL	FALSE	RCCP, RP	-	FALSE	The simulation is disabled.
					TRUE	The simulation is enabled.
NonPositionControlled	BOOL	FALSE	RCCP, RP	-	FALSE	The axis is in position-controlled mode.
					TRUE	The axis is not in position-controlled mode.
HomingDone	BOOL	FALSE	RCCP, RP	-	Homing status of the axis	
					FALSE	The axis is not homed.
					TRUE	The axis is homed and is capable of executing absolute positioning commands.
					The axis does not have to be homed for relative positioning. During active homing, the status is FALSE. The status remains TRUE during passive homing if the axis was already homed beforehand.	
Done	BOOL	FALSE	RCCP, RP	-	Command execution on the axis	
					FALSE	A Motion Control command is active on the axis.
					TRUE	A Motion Control command is not active on the axis.
Error	BOOL	FALSE	RCCP, RP	-	Error status on the axis	
					FALSE	No error is active on the axis.
					TRUE	An error has occurred on the axis.
					Additional information about the error is available in automatic mode at the "ErrorID" and "ErrorInfo" parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error.	
Standstill	BOOL	FALSE	RCCP, RP	-	Standstill of the axis	
					FALSE	The axis is in motion.
					TRUE	The axis is at a standstill.

Tag	Data type	Values	Access	W	Description	
PositioningCommand	BOOL	FALSE	RCCP, RP	-	Execution of a positioning command	
					FALSE	A positioning command is not active on the axis.
					TRUE	The axis executes a positioning command of the "MC_MoveRelative"" or "MC_MoveAbsolute" Motion Control instructions.
VelocityCommand	BOOL	FALSE	RCCP, RP	-	Execution of a command with velocity specification	
					FALSE	A command with velocity specification is not active on the axis.
					TRUE	The axis is executing a motion command with velocity specification of the "MC_MoveVelocity" or MC_MoveJog Motion Control instruction.
HomingCommand	BOOL	FALSE	RCCP, RP	-	Execution of a homing command	
					FALSE	A homing command is not active on the axis.
					TRUE	The axis is executing a homing command of the "MC_Home" Motion Control instruction.
CommandTableActive	BOOL	FALSE	RCCP, RP	-	Execution of a command table	
					FALSE	A command table is not active on the axis.
					TRUE	The axis is controlled by Motion Control instruction "MC_CommandTable".
ConstantVelocity	BOOL	FALSE	RCCP, RP	-	Constant velocity	
					FALSE	The axis is accelerating, decelerating, or at a standstill.
					TRUE	The setpoint velocity has been reached. The axis is moving at constant velocity.
Accelerating	BOOL	FALSE	RCCP, RP	-	Acceleration process	
					FALSE	The axis is decelerating, moving at constant velocity, or is at a standstill.
					TRUE	Axis is being accelerated.
Decelerating	BOOL	FALSE	RCCP, RP	-	Deceleration process	
					FALSE	The axis is accelerating, moving at constant velocity, or is at a standstill.
					TRUE	The axis is being decelerated.
ControlPanelActive	BOOL	FALSE	RCCP, RP	-	Activation status of the axis control panel	
					FALSE	"Automatic mode" is activated. The user program has control priority over the axis.

Tag	Data type	Values	Access	W	Description	
					TRUE	The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the axis. The axis cannot be controlled from the user program.
DriveReady	BOOL	FALSE	RCCP, RP	-	Operational status of the drive	
					FALSE	The drive is not ready. Setpoints will not be executed.
					TRUE	The drive is ready. Setpoints can be executed.
RestartRequired	BOOL	FALSE	RCCP, RP	-	Restart of axis required	
					FALSE	A restart of the axis is not required.
					TRUE	Values were modified in the load memory.
					To download the values to the work memory while the CPU is in RUN mode, the axis must be restarted. Use the MC_Reset Motion Control instruction to do this.	
SWLimitMinActive	BOOL	FALSE	RCCP, RP	-	Status of the software low limit switch	
					FALSE	The axis is kept within its configured work area.
					TRUE	The software low limit switch was reached or exceeded.
SWLimitMaxActive	BOOL	FALSE	RCCP, RP	-	Status of the software high limit switch	
					FALSE	The axis is kept within its configured work area.
					TRUE	The software high limit switch was reached or exceeded.
HWLimitMinActive	BOOL	FALSE	RCCP, RP	-	Status of the hardware low limit switch	
					FALSE	The axis is kept within its configured permitted traversing range.
					TRUE	The hardware low limit switch was reached or exceeded.
HWLimitMaxActive	BOOL	FALSE	RCCP, RP	-	Status of the hardware high limit switch	
					FALSE	The axis is kept within its configured permitted traversing range.
					TRUE	The hardware high limit switch was reached or exceeded.

See also

Status and error bits (technology objects as of V4) (Page 1447)

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

ErrorBits tags V4...5 (S7-1200)

The tag structure <axis name>.ErrorBits.<tag name> indicates error at the technology object.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description
ErrorBits.	STRUCT				TO_Struct_ErrorBits
SystemFault	BOOL	FALSE	RCCP, RP	-	Internal system error
ConfigFault	BOOL	FALSE	RCCP, RP	-	Incorrect configuration of the axis
DriveFault	BOOL	FALSE	RCCP, RP	-	Error in the drive. Loss of the "Drive ready" signal.
SWLimit	BOOL	FALSE	RCCP, RP	-	Software limit switch reached or exceeded
HWLimit	BOOL	FALSE	RCCP, RP	-	Hardware limit switch reached or exceeded
DirectionFault	BOOL	FALSE	RCCP, RP	-	Impermissible motion direction
HWUsed	BOOL	FALSE	RCCP, RP	-	Another axis is using the same PTO (Pulse Train Output) and is enabled with "MC_Power".
SensorFault	BOOL	FALSE	RCCP, RP	-	Error in the encoder system
CommunicationFault	BOOL	FALSE	RCCP, RP	-	Communication error Error in communication with a connected device.
FollowingErrorFault	BOOL	FALSE	RCCP, RP	-	Maximum permitted following error exceeded
PositioningFault	BOOL	FALSE	RCCP, RP	-	Positioning error The axis was not correctly positioned at the end of a positioning motion.

See also

Status and error bits (technology objects as of V4) (Page 1447)

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

ControlPanel tags V4...5 (S7-1200)

The "ControlPanel" tags do not contain any user-relevant data. These tags cannot be accessed in the user program.

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

Internal tags V4...5 (S7-1200)

The "Internal" tags do not contain any user-relevant data. These tags cannot be accessed in the user program.

See also

Tags of the positioning axis technology object as of V6 (Page 1488)

Tag of the axis technology object V1...3 (Page 1584)

Update of the technology object tags (S7-1200)

The status and error information of the axis indicated in the technology object tags is updated at each cycle control point.

Changes to the values of configuration tags do not take effect immediately. For information on the conditions under which a change takes effect, refer to the detailed description of the relevant tag.

Tags of the command table technology object V1...3 (S7-1200)

The tag structure <command table>.Config.Command.Command[x].<tag name> contains the configured command parameters.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description	
Config.	STRUCT				TO_Struct_Config	
Command.	STRUCT				TO_Struct_Command	
Command[x].	STRUCT				ARRAY[1..32] TO_Struct_Command[x]	
Type	INT	0 (0 to 151)	RW	-	Command type	
					0	"Empty" command
					2	"Stop" command
					5	"Relative positioning" command
					6	"Absolute positioning" command
					7	Command "Velocity setpoint"
					151	Command "Wait"
Position	REAL	0.0	RW	-	Target position/traversing distance of the command	
Velocity	REAL	0.0	RW	-	Command velocity	
Duration	REAL	0.0	RW	-	Command duration	
BufferMode	INT	0 (0 to 1)	RW	-	Value for command "Next step"	
					0	"Complete command"
					1	"Blend motion"
StepCode	WORD	0	RW	-	Command step code	

See also

Tag of the command table technology object V4...5 (Page 1621)

Tags of the command table V6 technology object (Page 1526)

Tag of the command table technology object V4...5 (S7-1200)

The tag structure <command table>.Command[n].<tag name> contains the configured command parameters.

Tags

Legend (Page 1583)

Tag	Data type	Values	Access	W	Description	
Command[n].	STRUCT				ARRAY[1..32] TO_Struct_Command[n]	
Type	INT	0 (0 to 151)	RW	-	Command type	
					0	"Empty" command
					2	"Stop" command
					5	"Relative positioning" command
					6	"Absolute positioning" command
					7	Command "Velocity setpoint"
					151	Command "Wait"
Position	REAL	0.0	RW	-	Target position/traversing distance of the command	
Velocity	REAL	0.0	RW	-	Command velocity	
Duration	REAL	0.0	RW	-	Command duration	
NextStep	INT	0 (0 to 1)	RW	-	Mode for the transition to the next command	
					0	"Complete command"
					1	"Blend motion"
StepCode	WORD	0	RW	-	Command step code	

See also

Tags of the command table technology object V1...3 (Page 1619)

Tags of the command table V6 technology object (Page 1526)

6.1.2 Using S7-1500 Motion Control (S7-1500)

6.1.2.1 Introduction (S7-1500)

Interplay of the various documents (S7-1500)

For a better overview, the documentation of the Motion Control functions is divided into the following documents:

- Using S7-1500T Motion Control
- Using S7-1500T Kinematics Functions

"Using S7-1500 Motion Control" describes the Motion Control functions for the following technology objects:

- Speed axis
- Positioning axis

- Synchronous axis
- External encoder
- Measuring input
- Output cam
- Cam track
- Cam (S7-1500T)

"Using S7-1500T Kinematics Functions" describes the Motion Control functions for the kinematics technology object. This document assumes that the Motion Control functions described in "Using S7-1500 Motion Control functions" are known.

Integrated Motion Control functionality (S7-1500)

S7-1500 Motion Control supports controlled positioning and moving of axes and is an integral part of every CPU S7-1500 as well as every CPU S7-1500SP. The S7-1500T Technology CPUs provide enhanced functions.

The Motion Control functionality supports the following technology objects:

- Speed axis
- Positioning axis
- Synchronous axis
- External encoder
- Measuring input
- Output cam
- Cam track
- Cam (S7-1500T)
- Kinematics (S7-1500T)

Drives with PROFIdrive capability and drives with analog setpoint interface and stepper motors are controlled by means of standardized Motion Control instructions according to PLCopen.

The axis control panel and comprehensive online and diagnostic functions support easy commissioning and optimization of drives.

S7-1500 Motion Control is fully integrated into the system diagnostics of the S7-1500 CPU.

Principle of operation of S7-1500 Motion Control (S7-1500)

You create a project, configure technology objects, and download the configuration to the CPU with the TIA Portal. The Motion Control functionality is processed in the CPU. You control the technology objects by using the Motion Control instructions in your user program. The TIA Portal provides additional functions for commissioning, optimization (Page 1947) and diagnostics (Page 1962).

The following figure provides a schematic representation of the user interfaces and the integration of Motion Control into the S7-1500 CPU. The concepts are then briefly explained:

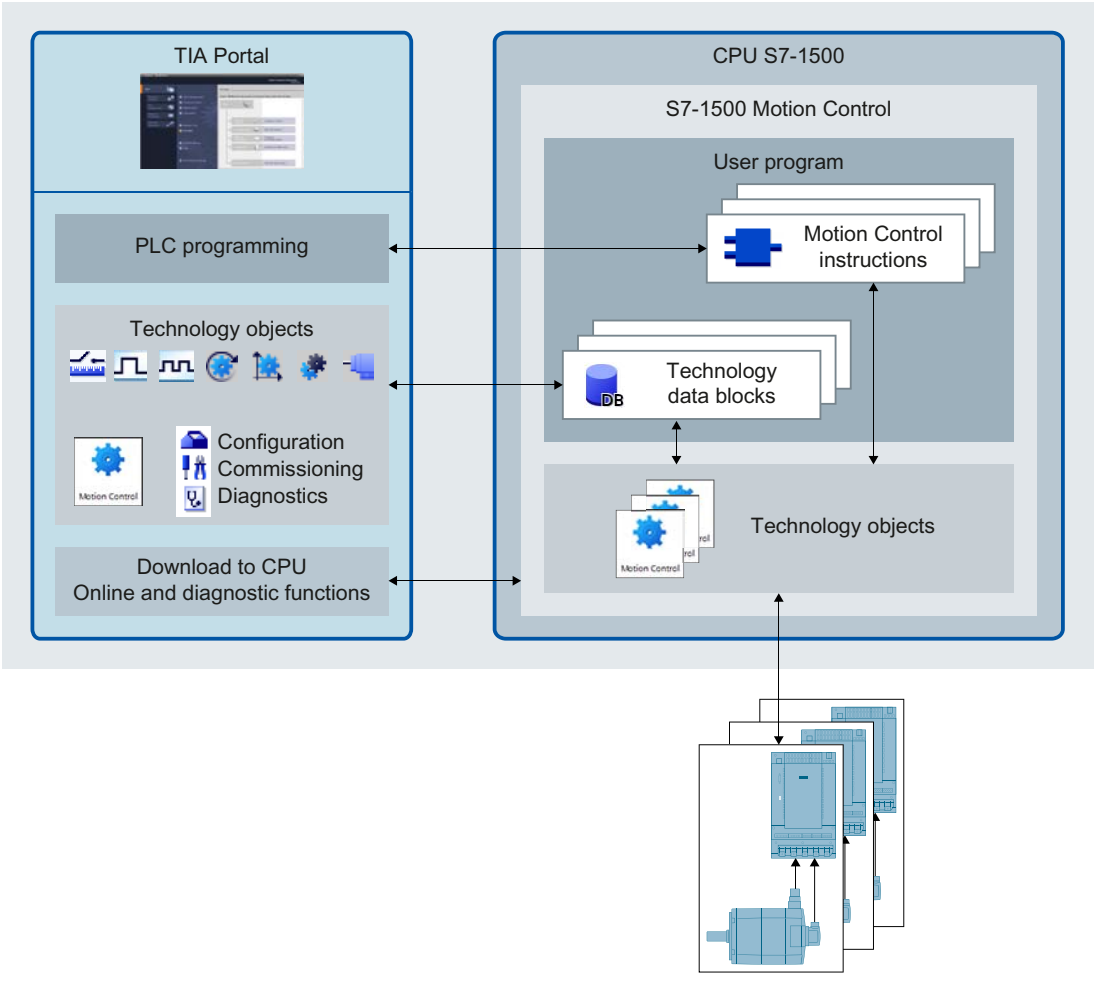

TIA Portal

The TIA Portal supports you in the planning and commissioning of Motion Control functionality:

- Integrating and configuring hardware
- Creating and configuring technology objects
- Creating the user program
- Downloading to CPU
- Commissioning of axes
- Optimization of drives
- Diagnostics

You use the TIA Portal to configure the hardware, the technology objects as well as your user program. You download the program you created to the CPU. You test your user program and diagnose the hardware with the online and diagnostic functions of the TIA Portal.

Technology objects

Technology objects represent real objects (e.g. a drive) in the controller. You call the functions of the technology objects by means of Motion Control instructions in your user program. The technology objects provide open- and closed-loop control of the movement of the real objects, and report status information (e.g. the current position).

The configuration of the technology objects represents the properties of the real object. The configuration data is stored in a technology data block.

The following technology objects are available for Motion Control:

Symbol	Technology object	Description
	Speed axis	The speed axis technology object ("TO_SpeedAxis") is used to specify the speed for a drive. You program the motion of the axis with motion control instructions.
	Positioning axis	The positioning axis technology object ("TO_PositioningAxis") is used to position a drive with closed-loop position control. You issue positioning jobs to the axis with Motion Control instructions in your user program.
	Synchronous axis	The synchronous axis technology object ("TO_SynchronousAxis") includes all functions of the positioning axis technology object. The axis can also be interconnected with a leading value so that the axis follows the position change of a leading axis in synchronous operation.
	External encoder	The external encoder technology object ("TO_ExternalEncoder") detects a position and makes it available to the controller. The detected position can be evaluated in the user program.
	Measuring input	The measuring input technology object ("TO_MeasuringInput") detects actual positions quickly, accurately and event triggered.
	Output cam	The output cam technology object ("TO_OutputCam") generates switching signals depending on the position of an axis or external encoder. You can evaluate the switching signals in the user program or feed them to digital outputs.
	Cam track	The cam track technology object ("TO_CamTrack") generates a switching signal sequence depending on the position of an axis or external encoder. In this process, up to 32 individual cams are superimposed and the switching signals are output as a track. You can evaluate the switching signals in the user program or feed them to digital outputs.
	Cam (S7-1500T)	The cam technology object ("TO_Cam") defines a function f(x) by means of interpolation points and/or segments. Missing function ranges are interpolated.
	Kinematics (S7-1500T)	The Kinematic technology object ("TO_Kinematics") is used to interconnect positioning axes to a kinematic. When you configure the kinematics technology object, you interconnect the axes in accordance with the configured kinematics type.

Technology data block

The properties of real objects are configured by means of the technology objects and saved in a technology data block (Page 1921). The technology data block contains all configuration data, setpoint and actual values, and status information of the technology object. The TIA Portal automatically creates the technology data block when the technology object is created. You access the data of the technology data block (read/write access) with your user program.

Motion Control instructions

With the Motion Control instructions you perform the desired functionality in the technology objects. The Motion Control instructions are available in the TIA Portal under "Instructions > Technology > Motion Control".

The Motion Control instructions conform to PLCopen (version 2.0).

User program

The Motion Control instructions and the technology data block represent the programming interfaces for the technology objects. You use Motion Control instructions to transfer Motion Control jobs for the technology objects in your user program. You track the status of running jobs via the output parameters of the Motion Control instructions. You access status information of the technology object and change specific configuration parameters during runtime using the technology data block.

Drives and encoders

Drives ensure the motion of the axis. They are integrated in the hardware configuration.

When you execute a Motion Control job in your user program, the technology object takes over the control of the drive and the reading in of values of encoders.

Drives and encoders with PROFIdrive capability are connected by means of PROFIdrive telegrams. The following connections are possible:

- PROFINET IO
- PROFIBUS DP
- Technology module (TM)

Drives with analog setpoint interfaces are connected using an analog output (AQ) and an optional enable signal. Analog inputs and outputs are made available by means of corresponding I/O modules.

A drive is also called an actuator, and an encoder is also called a sensor.

The figure below shows an example configuration in which all components are connected to the CPU by means of PROFINET IO:

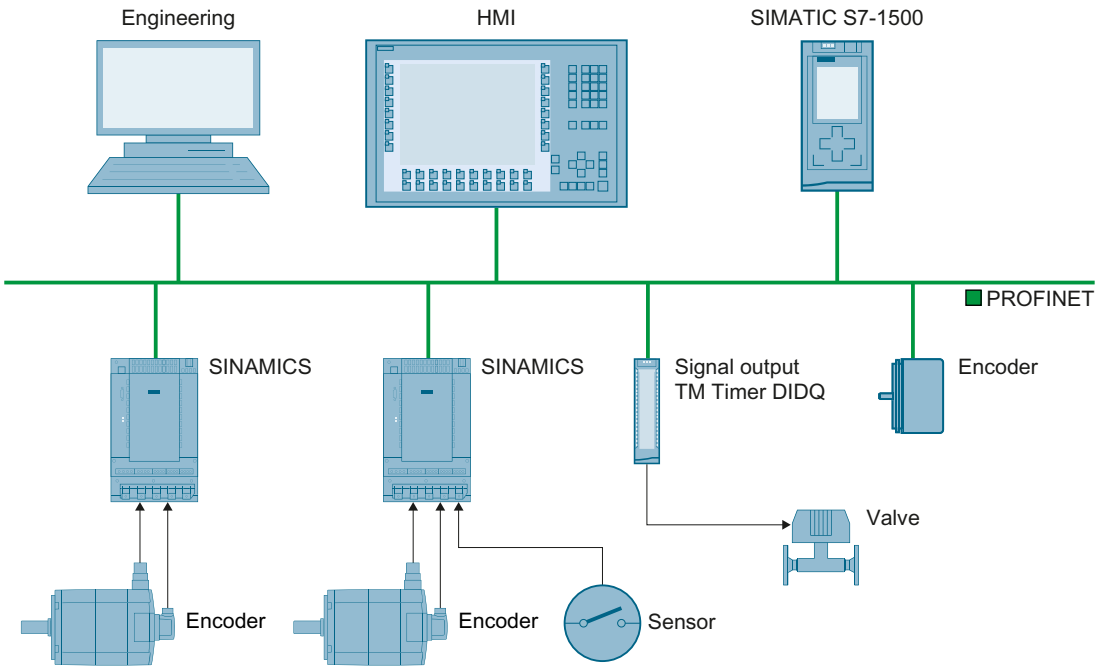

Functions (S7-1500)

Functions - Axes (S7-1500)

You execute the functions of the speed axis, positioning axis and synchronous axis technology objects using Motion Control instructions in your user program or using the TIA Portal (under "Technology object > Commissioning").

The following table shows the functions that are supported by technology objects:

Function	Technology object		
	Speed axis (Page 1704)	Positioning axis (Page 1705)	Synchronous axis (Page 1706)
Motion Control instructions (user program)			
"MC_Power" Enable, disable technology objects	X	X	X
"MC_Home" Home technology objects, set home position	-	X	X
"MC_MoveJog" Move axes in Jog mode	X	X	X
"MC_MoveVelocity" Move axes at predefined velocity	X	X	X
"MC_MoveRelative" Relative positioning of axes	-	X	X

Function	Technology object		
	Speed axis (Page 1704)	Positioning axis (Page 1705)	Synchronous axis (Page 1706)
"MC_MoveAbsolute" Absolute positioning of axes	-	X	X
"MC_MoveSuperimposed" Positioning axes overlapping	-	X	X
"MC_GearIn" Start gearing	-	-	X
"MC_Halt" Halt axes	X	X	X
"MC_TorqueLimiting" Limit torque using torque reduction	X	X	X
"MC_Reset" Acknowledge alarms, restart technology objects	X	X	X
"MC_TorqueAdditive" Specify additive torque and activate/deactivate	X	X	X
"MC_TorqueRange" Specify high and low torque limits	X	X	X
TIA Portal			
"Axis control panel (Page 1950)" Move and home axes using the TIA Portal	X	X	X
"Optimization (Page 1956)" Optimization of closed loop position control	-	X	X

Functions - Other technology objects (S7-1500)

You execute the functions of the other technology objects by means of Motion Control instructions in your user program.

The following table shows the Motion Control instructions that are supported by the technology objects:

Motion Control instruction	Technology object			
	External encoder (Page 1721)	Measuring input (Page 1722)	Output cam (Page 1734)	Cam track (Page 1748)
Motion Control instructions (user program)				
"MC_Power" Enable, disable technology objects	X	-	-	-
"MC_Home" Home technology objects, set home position	X	-	-	-

Motion Control instruction	Technology object			
	External encoder (Page 1721)	Measuring input (Page 1722)	Output cam (Page 1734)	Cam track (Page 1748)
"MC_MeasuringInput" Start measuring job once	-	X	-	-
"MC_MeasuringInputCyclic" Start cyclic measuring job	-	X	-	-
"MC_AbortMeasuringInput" End measuring job	-	X	-	-
"MC_OutputCam" Activate/deactivate output cam	-	-	X	-
"MC_CamTrack" Activate/deactivate cam track	-	-	-	X
"MC_Reset" Acknowledge alarms, restart technology objects	X	X	X	X

Extended functions of the technology CPU

In addition to the functionality of the S7-1500 CPU, the S7-1500T CPU provides additional functions and the cam technology object and kinematic:

Additional function	Description
Multiple encoders for positioning axis/synchronous axis (Page 1644)	Up to four encoders can be connected to a positioning axis/synchronous axis. The encoders can be switched over during operation. Only one encoder at a time is active for closed loop position control.
Actual value coupling (Page 1708)	As an alternative to the setpoint, the extrapolated actual value can be interconnected as a leading value for synchronous operation. As a result, an external encoder technology object can also be used as a leading value.
Gearing with "MC_GearInPos" (Page 1712)	During gearing, the leading axis and following axis are coupled, similar to a mechanical gear unit, by a linear synchronous operation function. You use the gear ratio to specify the synchronous operation function. The synchronous positions of the leading and following axes starting at which the axes move synchronously can be specified in "MC_GearInPos".
Cam technology object (Page 1761)	The cam technology object ("TO_Cam") defines a function f(x) by means of interpolation points and/or segments. Gaps between the defined interpolation points and segments of the cam are closed by interpolation during runtime of the user program.
Camming (Page 1714)	During camming, the leading axis and following axis are coupled by a synchronous operation function, which you specify using a cam.
Synchronization in advance using leading value distance (Page 1719) or dynamic parameters (Page 1718)	Gearing with "MC_GearInPos" and camming with "MC_CamIn" are synchronized in advance of the user-specified synchronous position.
"MotionIn" functions (Page 1664)	With the "MC_MotionInVelocity" and "MC_MotionInPosition" Motion Control instructions, you specify cyclically applicable calculated motion setpoints as a basic motion for the axis. No velocity profile is calculated for this, the values are directly active at the technology object.
Kinematics technology object	The Kinematic technology object ("TO_Kinematics") is used to interconnect positioning axes to a kinematic. When you configure the kinematics technology object, you interconnect the axes in accordance with the configured kinematics type.

You execute the extended functions of the technology CPU S7-1500T by means of Motion Control instructions in your user program. The following table shows the additional Motion Control instructions that are supported by the technology objects:

Motion Control instruction	Technology object		
	Positioning axis (Page 1705)	Synchronous axis (Page 1706)	Cam (Page 1761)
"MC_SetSensor" Switch active encoder	X	X	-
"MC_GearInPos" Start gearing with specified synchronous positions	-	X	-
"MC_PhasingAbsolute" Absolute shift of leading value for gearing with "MC_GearIn" and "MC_GearInPos"	-	X	-
"MC_PhasingRelative" Relative shift of leading value for gearing with "MC_GearIn" and "MC_GearInPos"	-	X	-
"MC_CamIn" Start camming	-	X	-
"MC_InterpolateCam" Interpolating a cam	-	-	X
"MC_GetCamFollowingValue" Read a following value with first and second derivative for the corresponding leading value from a cam	-	-	X
"MC_GetCamLeadingValue" Read a leading value for the corresponding following value from a cam	-	-	X
"MC_SynchronizedMotionSimulation" Start synchronous operation simulation and keep it active when following axis is disabled	-	X	-
MC_MotionInPosition Set motion setpoints for position, velocity and acceleration	X	X	-
MC_MotionInVelocity Set motion setpoints for velocity and acceleration	X[1)]	X	-

1) Also applies to the technology object speed axis.

Additional information

You can find additional information on the Motion Control instructions of the technology object kinematics in the "S7-1500T Kinematics Functions" function manual.

See also

Using S7-1500T Kinematic Functions (https://support.industry.siemens.com/cs/ww/en/view/109749264)

Guidelines on use of motion control (S7-1500)

The guidelines described here present the basic procedure for using Motion Control with the CPU S7-1500. These guidelines serve as recommendations.

Requirements

- A project with a CPU S7-1500 has been created.

Procedure

Proceed as follows to use Motion Control with the CPU S7-1500:

1. Add technology object (Page 1783)
2. Working with the configuration editor (Page 1785)
3. Programming (Page 1920)
4. Downloading to CPU (Page 1947)
5. Functional test in the Commissioning window (Page 1947)
6. Diagnostics (Page 1962)

6.1.2.2 Technology objects (S7-1500)

Configuration limits (S7-1500)

Motion Control resources

Each CPU offers a defined set of Motion Control resources. For information on the available Motion Control resources, refer to the technical specifications of the utilized CPU.

Each technology object uses Motion Control resources:

Technology object	Motion Control resources used
Speed axis	40
Positioning axis	80
Synchronous axis	160
External encoder	80
Measuring input	40
Output cam	20
Cam track	160

You can find an overview of the Motion Control resources of a CPU in the TIA Portal under "Tools > Resources".

Extended Motion Control Resources (S7-1500T)

The cam and kinematics technology objects use "Extended Motion Control resources". For information on the maximum number of cams and kinematics that can be used in addition to the Motion Control resources, refer to the technical specifications of the relevant CPU.

Technology object	Extended Motion Control resources used
Cam	2
Kinematics	30

Application cycle

As the number of technology objects used increases, the computing time needed by CPU to process the technology objects increases. The Motion Control application cycle (Page 1697) can be adapted according to the number of technology objects used.

See also

Extended Motion Control Resources (Page 2148)

Basics - Axes (S7-1500)

Axis types (S7-1500)

Axes can be configured with different axis types:

- Positioning and synchronous axes can be configured as rotary or linear axis.
- Speed axes are always rotary axes.

Depending on the execution of the mechanics, an axis is implemented as a linear axis or rotary axis:

- Linear axis

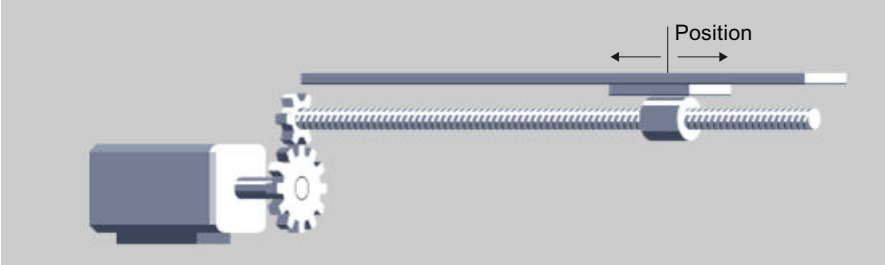

For linear axes, the position of the axis is specified as a linear measure, e.g. millimeters (mm).

- Rotary axis

For rotary axes, the position of the axis is specified as an angular measure, e.g. degrees (°).

Units of measure (S7-1500)

The supported units of measure for speed (revolutions per time unit) are 1/s, 1/min and 1/h.

The table below shows the supported units of measure for position and velocity:

Position	Velocity
nm, µm, mm, m, km	mm/s, mm/min, mm/h, m/s, m/min, m/h, km/min, km/h
in, ft, mi	in/s, in/min, ft/s, ft/min, mi/h
°, rad	°/s, °/min, rad/s, rad/min

The acceleration is set accordingly as the position/s² unit of measure.

The jerk is set accordingly as the position/s² unit of measure.

The table below shows the supported units of measure for force and torque:

Force	Torque
N, kN	Nm, kNm
lbf, ozf, pdl	lbf in, lbf ft, ozf in, ozf ft, pdl in, pdl ft

The unit of measure for time is permanently specified for the following technology objects:

Technology object	Time
Speed axis, positioning/synchronous axis, external encoder	s
Output cam, cam track, measuring input	ms

Note

When setting or changing the units of measure, take into consideration the effect on the display of parameter values and the user program:

- Display of parameter values in the technology data block
- Assignment of parameters in the user program
- Input and display of the position and velocity in the TIA Portal
- Setpoint settings by leading axes in synchronous operation

All information and displays correspond to the selected unit of measure.

The set units are displayed in the tag structure of the <TO>.Units technology object. The tag structure is described in the Appendix (Page 2012) under the tags of the respective technology object.

Modulo setting (S7-1500)

For the positioning axis, synchronous axis and external encoder technology objects, the "Modulo" setting can be activated.

When an axis moves in only one direction, the position value continually increases. To limit the position value to a recurring reference system, you can activate the "Modulo" setting.

When the "Modulo" setting is activated, the position value of the technology object is mapped onto a recurring modulo range. The modulo range is defined by the start value and the length.

For example, to limit the position value of a rotary axis to a full rotation, the modulo range can be defined with start value = 0° and length = 360°. As a result, the position value is mapped onto the modulo range 0° to 359.999°.

Long-term accuracy (S7-1500)

Long-term accuracy means that the technological position from the accumulated encoder increments can be exactly determined (without a rounding error) and are thus always accurate.

The long-term accuracy ensures the following:

- The axis position indicated by the control always corresponds to the real position.
  This means that the axis position can be exactly represented from the accumulated encoder increments.
- The calculated position in the controller does not deviate from the real position. Deviations occur without long-term accuracy, for example, due to rounding errors or conversions with finite accuracy.

Axes without modulo setting are stable over the long term until the numerical traversing range limit is reached. Axes with modulo setting are accordingly stable at least as long as they do not exceed the numerical traversing range of 9.0E12 mm at 1000 increments per unit.

You can use the following equation to estimate the time after which the traversing range is exceeded.

$$\text{Travel time} = \frac{\text{Maximum position}}{\text{Velocity}}$$

Example of the maximum traversing time

Maximum position = 9.0E12 mm with a resolution of 1000 Ink / mm

Velocity = 20.0 m/min = 2.0E4 mm/min

$$\text{Travel time} = \frac{9.0\text{E}12\ \text{mm}}{2.0\text{E}4\ \text{mm/min}} = 4.5\text{E}8\ \text{min} \quad \triangleq 856\ \text{years}$$

The traversing time restarts when one of the following conditions is fulfilled:

- You have homed the axis with "MC_Home".
- You have changed the velocity of the axis
  A change in the velocity has the consequence that the traversing time changes accordingly.

Avoiding exceeding the traversing range

To prevent exceeding the traversing range, do the following before the maximum traversing time expires:

- Incremental encoder: Home the incremental encoder again.
- Absolute encoder: Perform an absolute encoder calibration with the default of the currently known position.

Drive and encoder connection (S7-1500)

A drive is assigned to the "Speed axis" technology object. A drive and one to four encoders (only with S7-1500T) are assigned to the "Positioning axis" and "Synchronous axis" technology objects. An encoder is assigned to the "External encoder" technology object.

The setpoint for the drive is specified either with PROFIdrive telegrams or using an analog output.

The following connection options are available for an encoder:

- Encoder connected to drive
- Encoder to technology module
- PROFIdrive encoder connected directly to PROFIBUS DP/PROFINET IO

The actual encoder value is transmitted exclusively via PROFIdrive telegrams.

PROFIdrive

PROFIdrive is the standardized drive technology profile for connecting drives and encoders via PROFIBUS DP and PROFINET IO. Drives that support the PROFIdrive profile are connected according to the PROFIdrive standard.

The current PROFIdrive specification is available at:

http://www.profibus.com (http://www.profibus.com)

Communication between the controller and drive/encoder is performed using various PROFIdrive telegrams. Each of the telegrams has a standardized structure. You can select the appropriate telegram according to the application. Control words and status words as well as setpoints and actual values are transmitted in the PROFIdrive telegrams.

The PROFIdrive profile likewise supports the "Dynamic Servo Control" (DSC) control concept. DSC uses rapid closed loop position control in the drive. This can be used to solve highly dynamic Motion Control tasks.

Analog drive connection

Drives with analog setpoint interfaces are connected using an analog output and an optional enable signal. The speed setpoint is specified via an analog output signal (e.g. from -10 V to +10 V) from the CPU.

Stepper motors

Drives with a stepper motor interface are connected using telegram 3 and with the help of PTO (Pulse Train Output) pulse generators.

For functional support of stepper motor operation, quantization of the control deviation can be set.

Through the specification of a quantization, a range around the target position is defined in which no correction of the actual position is to be made. This prevents a possible oscillation of the stepper motor around the target position. Two types of quantization can be set:

- Quantization of the control deviation corresponding to the encoder resolution ("<TO>.PositionControl.ControlDifferenceQuantization.Mode" = 1)
  This prevents oscillation of the stopped motor between two increment values, for example. This mode is especially helpful when using multiple encoders. With this setting, the quantization is adapted appropriately at an encoder switchover. This mode is helpful for stepper motors with encoders in which the resolution of the encoder is lower than the step size of the stepper motor.
- Direct specification of a value for quantization of the control deviation. ("<TO>.PositionControl.ControlDifferenceQuantization.Mode" = 2, value setting in "<TO>.PositionControl.ControlDifferenceQuantization.Value")
  This mode is helpful for stepper motors with encoders in which the resolution of the encoder is greater than the step size of the stepper motor.

PROFIdrive telegrams (S7-1500)

PROFIdrive telegrams are used to transfer setpoints and actual values, control and status words and other parameters between the controller and drive/encoder.

When a PROFIdrive telegram is used for connection, the drives and encoders are handled and switched on in accordance with the PROFIdrive profile.

The following table shows the possible PROFIdrive telegrams for various technology objects.

Technology object	Possible PROFIdrive telegrams
Speed axis	• 1, 2 • 3, 4, 5, 6, 102, 103, 105, 106 (actual encoder value is not evaluated)
Positioning axis/synchronous axis	
Setpoint and actual encoder value in one drive telegram	3, 4, 5, 6, 102, 103, 105, 106
Setpoint and actual encoder value separately	
Setpoint in drive telegram	1, 2, 3, 4, 5, 6, 102, 103, 105, 106
Actual value from telegram	81, 83
External encoder	81, 83
Measuring input [1)]	391, 392, 393

1) When using SINAMICS drives (measuring using SINAMICS measuring input)

Telegram types

The following table shows the supported PROFIdrive telegram types for the assignment of drives and encoders:

Telegram	Brief description
Standard telegrams	
1[1)]	• Control word STW1, status word ZSW1 • Speed setpoint 16 bit (NSET), actual speed value 16 bit (NACT)

Telegram	Brief description
2	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT)
3	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2)
4	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) • Actual encoder value 2 (G2_XIST1, G2_XIST2)
5	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) (motor encoder) • Dynamic Servo Control (DSC)[2)] – Speed precontrol value – Position difference (XERR) – Kpc - Velocity precontrol of the closed loop position control
6	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) (motor encoder) • Actual encoder value 2 (G2_XIST1, G2_XIST2) • Dynamic Servo Control (DSC)[2)] – Speed precontrol value – Position difference (XERR) – Kpc - Velocity precontrol of the closed loop position control
Siemens telegrams (with torque limiting)	
102	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) • Torque limiting
103	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) • Actual encoder value 2 (G2_XIST1, G2_XIST2) • Torque limiting

Telegram	Brief description
105	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) (motor encoder) • Dynamic Servo Control (DSC)[2)] – Speed precontrol value – Position difference (XERR) – Kpc - Velocity precontrol of the closed loop position control • Torque limiting
106	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) (motor encoder) • Actual encoder value 2 (G2_XIST1, G2_XIST2) • Dynamic Servo Control (DSC)[2)] – Speed precontrol value – Position difference (XERR) – Kpc - Velocity precontrol of the closed loop position control • Torque limiting
SIEMENS additional telegrams (torque data)	
750[3)]	• Additive setpoint torque • High and low torque limits • Torque actual values
SIEMENS telegrams (measuring input) [4)]	
391	• Control word CU_STW1, status word CU_ZSW1 • Measuring input control word (MT_STW), measuring input status word (MT_ZSW) • Measuring input time stamp of negative (MT1…2_ZS_F) or positive edges (MT1…2_ZS_S) • Digital output 16 bit, digital input 16 bit
392	• Control word CU_STW1, status word CU_ZSW1 • Measuring input control word (MT_STW), measuring input status word (MT_ZSW) • Measuring input time stamp of negative (MT1…6_ZS_F) or positive edges (MT1…6_ZS_S) • Digital output 16 bit, digital input 16 bit
393	• Control word CU_STW1, status word CU_ZSW1 • Measuring input control word (MT_STW), measuring input status word (MT_ZSW) • Measuring input time stamp of negative (MT1…8_ZS_F) or positive edges (MT1…8_ZS_S) • Digital output 16 bit, digital input 16 bit • Analog input 16 bit
Standard telegrams - encoder	

Telegram	Brief description
81	• Control word STW2_ENC, status word ZSW2_ENC • Actual encoder value 1 (G1_XIST1, G1_XIST2)
83	• Control word STW2_ENC, status word ZSW2_ENC • Actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2)

1) Isochronous mode is not possible.

2) For use of Dynamic Servo Control (DSC), the motor encoder (first encoder in the telegram) of the drive must be used as the first encoder for the technology object.

3) Can also be used for the telegrams 1, 2, 3, 4, 5, 6, 102, 103, 105, 106

4) When using SINAMICS drives (measuring using SINAMICS measuring input)

See also

Data connection drive/encoder via data block (Page 1651)

Actual values (S7-1500)

For position-controlled motion and positioning, the controller must know the actual position value.

The actual position value is provided by a PROFIdrive telegram.

The actual values are represented as incremental or absolute values in the PROFIdrive telegram. The actual values are normalized in the controller to the technological unit taking into account the configuration of the mechanics. The reference to a physical position of the axis or external encoder is established by homing.

The controller supports the following types of actual values (encoder types):

- Incremental actual value
- Absolute actual value with the setting absolute (measuring range > traversing range of the axis)
- Absolute actual value with the setting absolute (measuring range < traversing range of the axis)

Actual value calculation for virtual axis or axis in simulation

The actual value of a virtual axis or an axis in simulation is formed from the setpoint taking time delays into account.

You calculate the time delay from actual value to the setpoint (T_t) as follows:

Calculation	
With precontrol	$T_t = T_{ipo} + T_{servo} + T_{vtc} + T_{addPtc}$
Without precontrol, without DSC	$T_t = T_{ipo} + 1/Kv + T_{addPtc}$
Without precontrol, with DSC for one axis in simulation	$T_t = T_{ipo} + T_{servo} + 1/Kv + T_{addPtc}$

T_t	Time delay from the actual value to the setpoint
T_{ipo}	CPU time of MC-Interpolator [OB92]
T_{servo}	CPU time of MC-Servo [OB91]
T_{vtc}	Speed control loop substitute time (T_{vtc} off "<TO>.DynamicAxisModel.VelocityTimeConstant")
T_{addPtc}	Additive position control loop equivalent time (T_{addPtc} from "<TO>.DynamicAxisModel.AdditionalPositionTimeConstant")
kV	Gain factor (Kv from "<TO>.PositionControl.Kv")

See also

Virtual axis (Page 1651)

Axis in simulation (Page 1650)

Incremental actual value (S7-1500)

The actual value in the PROFIdrive telegram is based on an incremental value.

After POWER ON, position zero is displayed. A transition of the CPU to RUN mode starts the actual value update. The actual value is then also updated in CPU STOP mode. The relationship between the technology object and the mechanical position must be re-established by means of homing.

Absolute actual value (S7-1500)

The actual value in the PROFIdrive telegram is based on an absolute value.

After POWER ON, position zero is displayed. The first transition of the CPU to RUN mode starts the actual value update. The actual value is then also updated in CPU STOP mode. The supplied absolute value is assigned to the associated mechanical axis position by means of the absolute encoder adjustment. The absolute encoder adjustment must be performed once. The absolute value offset is retentively saved beyond the switching on/off of the controller.

Differentiation of absolute values:

- The measuring range of the encoder is larger than the traversing range of the axis: Absolute value with setting absolute
- The measuring range of the encoder is smaller than the traversing range of the axis: Absolute value with setting cyclic absolute

Absolute actual value with setting absolute (measuring range > traversing range)

The axis position results directly from the current actual encoder value. The traversing range must be within an encoder measuring range. This means that the zero passage of the encoder must not be located in the traversing range.

When the controller is switched on, the axis position is determined from the absolute actual encoder value.

Absolute actual value with setting cyclic absolute (measuring range < traversing range)

The encoder supplies an absolute value within its measuring range. The controller includes the traversed measuring ranges and thus determines the correct axis position beyond the measuring range.

When the controller is switched off, the traversed measuring ranges are saved in the retentive memory area of the controller.
At the next power-on, the saved traversed measuring ranges are taken into account in the calculation of the actual position value.

NOTICE
Movements of the axis while the controller is switched off can skew the actual value If the axis or the encoder is moved by more than half of the encoder measuring range while the controller is switched off, then the actual value in the controller is no longer in accord with the mechanical axis position.

See also

Absolute value adjustment (Page 1688)

Tags (S7-1500)

The tags named in the Homing (Page 1689) section are relevant for adapting actual values.

Automatic transfer of drive and encoder parameters in the device (S7-1500)

Identical reference values for the drive and encoder connections must be set in the controller and in the drive and encoder for the operation.

The speed setpoint NSET and the actual speed value NACT are transferred in the PROFIdrive telegram as a percentage of the reference speed. The reference value for the speed must be set identically in the controller and in the drive.

The resolution of the actual value in the PROFIdrive telegram must likewise be set identically in the controller and in the drive and encoder modules

Automatic transfer of parameters

The drive and encoder parameters can be automatically applied in the CPU for the following drives and encoders.

- SINAMICS drives (see compatibility list (Page 2146))
- PROFIdrive encoder as of product version A16

The corresponding parameters are transferred after the (re-)initialization of the technology object or (re)start of the drive and the CPU. Changes in the drive configuration are transferred after restart of the drive or technology object.

Successful transfer of the parameters can be checked in the controller with the value of the tags "<TO>.StatusDrive.AdaptionState" = 2 and "<TO>.StatusSensor[n].AdaptionState" =2 of the technology object.

Parameters

The controller settings are made in the TIA Portal under "Technology object > Configuration > Hardware interface > Data exchange with the drive/encoder".

The drive and encoder settings are made in the configuration or the respective hardware.

The following table compares the settings in the TIA Portal, in the controller and the corresponding drive/encoder parameters:

Setting in the TIA Portal	Controller tag in the technology data block		Drive parameter	Automatic transfer
Drive				
Telegram number	Telegram input address <TO>.Actor.Interface.AddressIn		Telegram number P922	-
	Telegram output address <TO>.Actor.Interface.AddressOut			
Reference speed in [1/min]	<TO>.Actor.DriveParameter.Reference-Speed		(SINAMICS drives: P2000)	X
Maximum speed of motor in [1/min]	<TO>.Actor.DriveParameter.MaxSpeed		(SINAMICS drives: P1082)	X
Reference torque in [NM]	<TO>.Actor.DriveParameter.Reference-Torque		(SINAMICS drives: P2003)	X
Encoder				
Telegram	<TO>.Sensor[n].Interface.AddressIn		P922	-
	<TO>.Sensor[n].Interface.Addressout			
Encoder type	<TO>.Sensor[n].Type		P979[5] Encoder 1 P979[15] Encoder 2	-
	0	Incremental		
	1	Absolute		
	2	Cyclic absolute		
Measuring system	<TO>.Sensor[n].System		P979[1] Bit0 Encoder 1 P979[11] Bit0 Encoder 2	X
	0	Linear		
	1	Rotary		
Resolution (linear encoder) The grid spacing is specified on the name-plate of the encoder as a separation distance of the marks on the linear measuring system.	<TO>.Sensor[n].Parameter.Resolution		P979[2] Encoder 1 P979[12] Encoder 2	X
Increments per revolution (rotary encoder)	<TO>.Sensor[n].Parameter.StepsPerRe-volution		P979[2] Encoder 1 P979[12] Encoder 2	X
Number of bits for fine resolution XIST1 (cyclic actual encoder value, linear or rotary encoder)	<TO>.Sensor[n].Parameter.FineResolu-tionXist1		P979[3] Encoder 1 P979[13] Encoder 2	X

Setting in the TIA Portal	Controller tag in the technology data block	Drive parameter	Automatic transfer
Number of bits for fine resolution XIST2 (absolute encoder value, linear or rotary encoder)	<TO>.Sensor[n].Parameter.FineResolutionXist2	P979[4] Encoder 1 P979[14] Encoder 2	X
Differentiable encoder revolutions (rotary absolute encoder)	<TO>.Sensor[n].Parameter.DeterminableRevolutions	P979[5] Encoder 1 P979[15] Encoder 2	X

Using multiple encoders

The S7-1500T technology CPU offers the option of using up to 4 encoder or measuring systems per positioning axis and synchronous axis as the actual position for the closed loop position control

Only one encoder at a time is active for closed loop position control. You can switch between the 4 encoder or measuring systems.

However, the actual values of all configured encoders can be evaluated in the user program.

This opens up the following possible application areas, for example:

- Use of additional machine encoders (besides the motor encoder), e.g. as direct measuring systems for more accurate detection of actual positions of machining processes.
- Use of alternative encoder systems following a tool change in a flexible manufacturing process.

You configure the encoders in the axis configuration. You control the switchover of the encoders in the user program with the Motion Control instruction "MC_SetSensor".

Configuring an axis with multiple encoders

Note the following configuration windows when using multiple encoders:

- In the configuration window "Hardware interface > Encoder", configure which alternative encoders are to be used and their corresponding encoder type (incremental, absolute or cyclic absolute).
  All encoders marked as used supply continually updated actual values to the closed loop position control regardless of their use.
- In the configuration window "Hardware interface > Encoder", configure an encoder as "Encoder at power-up". This is necessary because an encoder must always be assigned to the positioning axis and synchronous axis.
- In the configuration window "Hardware interface > Data exchange with encoder", configure additional encoder details and the telegram that is to be used to connect the encoders. The configuration must be performed for each encoder used.
  Each encoder to be used or each measuring system may differ with regard to its encoder mounting type.

- In the configuration window "Extended parameters > Mechanics", configure the encoder mounting type and any gear parameters. The configuration must be performed for each encoder used.
- The axis can be homed with any configured encoder. In the configuration window "Extended parameters > Homing", configure the parameters for active and passive homing. The configuration can be performed for each encoder used.
  When the axis is homed with an encoder, the axis is homed and retains the "homed" status following encoder switchover.

Encoder switchover in the user program

For closed loop position control of the positioning and synchronous axes, an encoder must always be active. Individual encoders may fail as long as they are not involved in closed loop position control.

With the Motion Control instruction "MC_SetSensor", you switch over the encoder for closed loop position control of the axis.

The switchover can occur during an active motion job or at a standstill. The axis does not have to be enabled.

A switchover during an active homing or restart job is not possible.

Note

Homing

Homing with the Motion Control instruction "MC-Home" or the axis control panel is always performed with the encoder involved in closed loop position control.

The homing status of the axis is not changed following an encoder switchover.

Simulation

When the axis is simulated, all encoders configured as "used" are simulated.

Following the switchover to an alternative encoder or encoder system, you can select what happens if the actual positions of the encoders are different.

You define how to deal with the difference in the actual positions of the encoders using input parameter "Mode" of the Motion Control instruction "MC_SetSensor".

- **Switch over encoder and transfer actual position to the encoder to be switched ("Mode" = 0)**
  With this encoder switchover, step changes in the actual position are prevented. Bumpless switchover of the encoders is possible.
- **Switch over sensor without transferring the actual position ("Mode" = 1)**
  Following a switchover to an encoder without adjustment, a step change of the actual position may occur. This can be desirable if the new encoder is intended to compensate for possible mechanical influences (such as slip) in the positioning.
  The position difference is not implemented immediately but rather after a delay using time constant "<TO>.PositionControl.SmoothingTimeByChangeDifference" in order to prevent step changes in the actual position with active closed loop position control.

- **Transfer actual position ("Mode" = 2)**
  The actual position of the axis is transferred to the encoder specified in the "Sensor" parameter.
- **Transfer actual position of the reference encoder ("Mode" = 3)**
  The actual position of the "Reference encoder" ("ReferenceSensor" parameter) is transferred to the encoder specified in the "Sensor" parameter.

"Mode" = 2 and 3 can be used to prepare a switchover.

Safety functions in the drive (S7-1500)

Safety functions ("Safety Integrated Basic Functions") in the SINAMICS drive are safety-oriented, internal drive functions with the goal of shutting down the respective drive safely. In addition, additional safety functions ("Safety Integrated Extended Functions") are available to monitor definable limits. The goal of these safety functions is to monitor that the respective limit is maintained, to signal the violation or to subsequently shut down the drive safely. To prevent the monitoring function from being triggered, the axis must be switched to the monitored operating mode with the user program or maintained in this mode.

The cooperation of the safety functions in the drive and the SIMATIC S7-1500 and S7-1500T are necessary to ensure fault-free plant operation.

The technology objects speed axis, positioning axis and synchronous axis support the "Safety Integrated Basic Functions" of the drive. The technology object detects that the Basic Safety function is triggered and displays a corresponding warning (technology alarm 550 - alarm response: Track setpoints) or alarm (technology alarm 421 - alarm response: Remove enable).

No additional response must take place in the user program, especially for technology alarm 550, at the "Enable" input of the Motion Control instruction "MC_Power" in the sense of a "shutdown". Shutdown of the Motion Control instruction "MC_Power" by the user after technology alarm 421 is possible but not necessary.

After acknowledgment and release of the drive by the actual safety function, the technology alarm can also be acknowledged at the respective technology object with the Motion Control instruction "MC_Reset". The technology object is then enabled automatically, if "MC_Power.Enable" is still "TRUE".

The "Safety Integrated Extended Functions" are **not** supported by the technology object independently.

To prevent the extended safety functions from being triggered and thus prevent a disruption in plant operation, evaluate the status of the safety functions. This evaluation can take place in the user program by using or evaluating the status information of the "Safety Info Channels" (SIC). With the help of the corresponding Motion Control instruction, the axis can now stay within the monitored limit or reach it before a deviation is detected.

If a safety function is applied to a following axis in active synchronous operation, one of the following two responses is required:

- End synchronous operation
- Adjust velocity of the leading axis accordingly

Four status words are available in the SIC:

- S_ZSW1B
- S_ZSW2B

- S_ZSW3B
- S_V_LIMIT_B

Two predefined PROFIdrive telegrams are available for transmission:

- Tel. 700 (contains the status words S_ZSW1B & S_V_LIMIT_B)
- Tel. 701 (contains all four status words and two additional control words)

Additional information

Additional information on the safety functions in SINAMICS drives and on the SIC is available in the SINAMICS S120 Safety Integrated Function Manual.

https://support.industry.siemens.com/cs/document/99668646 (https://support.industry.siemens.com/cs/ww/en/view/109754301)

The tables below provide an overview of the four SIC status words and the required response for each to prevent faults in plant operation.

S_ZSW1B

S_ZSW1B		Meaning		Recommended response of the respective axis in the user program
Bit	Assignment			
0	STO (active)	1	Safe Torque Off active	"MC_Power" can remain enabled (waiting).
		0	Not active	None
1	SS1 (active)	1	Safe Stop 1 active	Drive brakes autonomously and goes into STO. "MC_Power" remains enabled until STO.
		0	Not active	None
2	SS2 (active)	1	Safe Stop 2 active	Drive brakes autonomously and goes into SOS. "MC_Power" remains enabled.
		0	Not active	None
3	SOS (active)	1	Safe Operating Stop active	"MC_Power" remains enabled. The drive must not move (monitoring takes place in the drive)
		0	Not active	None
4	SLS (active)	1	Safety-Limited Speed active	"MC_Power" remains enabled. Velocity must be less than active velocity limit (see "Active SLS stage" or S_V_LIMIT_B).
		0	Not active	None
5	SOS (selected)	1	Safe Operating Stop selected	"MC_Power" remains enabled. Braking with "MC_Halt" within the time permitted by SOS.
		0	Deselected	None
6	SLS (selected)	1	Safety-Limited Speed selected	Drop below velocity limit within the time specified by SLS. For example, by specifying an override or a new dynamic limit (restriction with synchronous motion).
		0	Deselected	None

S_ZSW1B		Meaning		Recommended response of the respective axis in the user program
Bit	Assignment			
7	Internal event	1	Group alarm that a safety function was selected or became active	Additional evaluation of the status words required to determine the triggering safety function. The bit indicates that a safety function is active. (See also "Safety alarm")
		0	No event	None
8	Reserved	-		-
9	Active SLS level	SLS velocity limit Display bit 0		Additional information on SLS (bit 6) – shows active velocity limit for SLS in levels (1… 4). This limit can be evaluated in the program to limit the current velocity of the axis as needed.
10		SLS velocity limit Display bit 1		
11	Reserved	-		-
12	SDI positive	1	Safe Direction positive selected	Reaching standstill or the positive velocity of the actual value of the axis (when SDI negative = 0) within the time specified by SDI.
		0	Deselected	None monitoring for positive direction.
13	SDI negative	1	Safe Direction negative selected	Reaching standstill or the negative velocity of the actual value of the axis (when SDI positive = 0) within the time specified by SDI.
		0	Deselected	None monitoring for negative direction.
14	ESR retraction	1	Extended stop and retraction required (not a Safety function)	To be considered individually. You can find additional information in the SINAMICS S120 Safety Integrated Function Manual. https://support.industry.siemens.com/cs/document/99668646 (https://support.industry.siemens.com/cs/ww/en/view/109754301)
		0	Not requested	None
15	Safety alarm	1	Effective	If necessary, evaluate bit as group alarm to see whether a Safety alarm is pending in the alarm buffer.
		0	Not effective	None

S_V_LIMIT_B

S_V_LIMIT_B		Meaning	Explanation
Bit	Assignment		
0 … 31	Velocity setpoint limit	SLS Speed limit (32-bit resolution with sign)	Additional information on SLS (S_ZSW1B bit 6) Shows the selected/active velocity limit for SLS. If necessary, the velocity limit can be evaluated in the program to limit the current velocity of the axis as needed.

S_ZSW2B

S_ZSW2B		Meaning		Recommended response of the respective axis
Bit	Assignment			
0 … 3	Reserved	-		-
4	SLP selected position area	1	SLP area 2 selected	Safe position in the area 2. Do not change the position using the user program any longer.
		0	SLP area 1 selected	Safe position in the area 1. Do not change the position using the user program any longer.
5, 6	Reserved	-		-
7	SLP selected and user approval set	1	Safety-Limited Position selected and user approval is set	Status message (if required in the user program) Application-dependent evaluation (means that the SLP is selected and the safe position was confirmed by the user – see "Safe homing")
		0	SLP not selected or user approval missing	Application-dependent evaluation
8	SDI positive	1	Safe Direction positive selected	Reaching standstill or the positive velocity of the actual value of the axis within the time specified by SDI. (when SDI negative = 0)
		0	Deselected	None monitoring for positive direction
9	SDI negative	1	Safe Direction negative selected	Reaching standstill or the negative velocity of the actual value of the axis within the time specified by SDI. (when SDI positive = 0)
		0	Deselected	None monitoring for negative direction
10, 11	Reserved	-		-
12	Test stop active	1	Test stop active	Status message (if required in the user program) Application-dependent evaluation
		0	Not active	None
13	Test stop required	1	Test stop required	Perform test stop
		0	Not required	None
14, 15	Reserved	-		-

S_ZSW3B

S_ZSW3B		Meaning		Recommended response of the respective axis
Bit	Assignment			
0	Brake test	1	Brake test selected	"MC_Power" remains enabled. Do not start travel motion with user program.
		0	Deselected	None – brake test is inactive (normal plant operation)

S_ZSW3B		Meaning		Recommended response of the respective axis
Bit	Assignment			
1	Setpoint specification drive/ external	1	Specification for drive	The speed setpoint is specified by the SBT function. Application-dependent evaluation
		0	External specification (control)	The "normal" speed setpoint is in effect. Application-dependent evaluation, setpoint value specification through user program required
2	Active brake	1	Test brake 2 active	Status message (if required in the user program) Application-dependent evaluation
		0	Test brake 1 active	Status message (if required in the user program) Application-dependent evaluation
3	Brake test active	1	Test active	Status message (if required in the user program) Application-dependent evaluation
		0	Inactive	None
4	Brake test result	1	Test required	Status message (if required in the user program) Application-dependent evaluation
		0	With error(s)	Status message (if required in the user program) Application-dependent evaluation, test must usually be successful to guarantee safety of the brake.
5	Brake test complete	1	Run test	Status message (if required in the user program) Application-dependent evaluation
		0	Incomplete	Status message (if required in the user program) Application-dependent evaluation, usually test is repeated
6	External brake request	1	Closing the brake	Closing external brake (when controlled by user program) Application-dependent evaluation
		0	Opening the brake	Opening external brake (when controlled by user program) Application-dependent evaluation
7	Current load sign	1	Negative sign	Status of the load sign if required in the user program Application-dependent evaluation
		0	Positive sign	
8 … 13	Reserved	-		-
14	Acceptance test SLP(SE) selected	1	Acceptance test SLP(SE) selected	Status message (if required in the user program) Application-dependent evaluation
		0	Deselected	None
15	Acceptance test mode selected	1	Acceptance test mode selected	Status message (if required in the user program) Application-dependent evaluation
		0	Deselected	None

Axis in simulation (S7-1500)

S7-1500 Motion Control offers the option to move real axes in simulation mode. Speed, positioning and synchronous axes can thus be simulated without a connected drive and encoder in the CPU.

When the simulation mode is activated, the drive and encoder connection does not need to be configured in the axis configuration, for example, if the drive configuration is not yet available

at this time. The "Simulation" configuration can be changed during runtime of the user program (<TO>.Simulation.Mode). A valid drive and encoder connection is required when exiting the simulation.

To use a technology object in simulation mode or with SIMATIC S7 PLCSIM, you need to use encoder 1 for closed loop position control of the axis.

Applications

- For example, an axis is simulated for programming the machine application and assigned to the configured hardware later for commissioning.
- During commissioning, for example, not all hardware components are available.
- No axis motions should take place during commissioning.

Characteristics in simulation mode

An axis in simulation does not output setpoints to the drive and does not read any actual values of the encoder. The actual values (Page 1639) are formed with a time delay from the setpoints.

Hardware limit switches and home position switches have no effect.

The technology objects measuring input (with signal detection via TM Timer DIDQ or SINAMICS measuring input), output cam and cam track can also be used for axes in simulation.

The following table shows the Motion Control instructions with adapted behavior in simulation mode:

Motion Control instruction	Characteristics in simulation mode
MC_Power	The axis is enabled immediately without waiting for feedback from the drive.
MC_Home	Homing jobs are executed immediately without simulated axis motion.
MC_TorqueLimiting	The specified torque is not output to the drive.

Virtual axis (S7-1500)

S7-1500 Motion Control offers the possibility to configure an axis as a virtual axis. A virtual axis has motion control, but, in contrast to real axes, no drive and encoder connection. The setpoints are only processed within the controller and no real drive is controlled.

Application:

A virtual axis, for example, is often used as a virtual leading axis in order to generate the setpoints for several real following axes in synchronous operation.

The "Virtual axis" configuration can only be changed by a new download to the CPU (in STOP mode) (<TO>.VirtualAxis.Mode).

The behavior of a virtual axis is identical to the behavior of an axis in simulation (Page 1649).

Data connection drive/encoder via data block (S7-1500)

The data connection of PROFIdrive drives and encoders occurs either directly via the PROFIdrive telegram or via a data block.

Use the system-generated tags of the PROFIdrive telegrams if you want to evaluate the telegram contents.

Use the connection via data block if you want to influence or evaluate telegram contents in the user program for process-specific reasons.

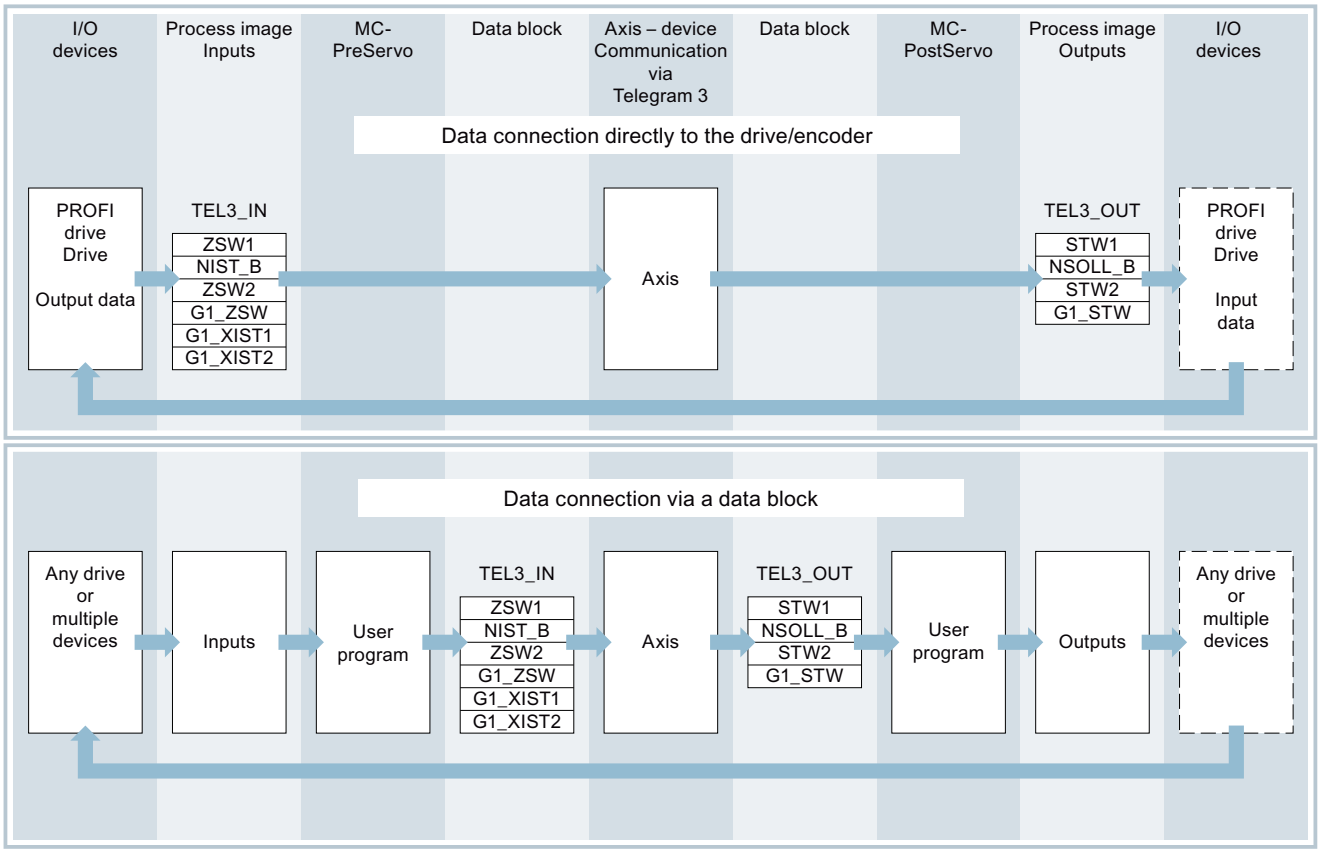

Principle of data connection via data block

Generally, at the start of closed loop position control of the axis (by MC-Servo [OB91]), the input area of the drive or encoder telegram is read.

At the end of closed loop position control, the output area of the drive or encoder telegram is written.

To influence or evaluate telegram contents for process-specific reasons, a data interface via a data block must be connected in between before and after the closed loop position control.

- The input area of the telegram can be edited using the MC-PreServo [OB67] organization block. The MC-PreServo is called before the MC-Servo.
- The input area of the telegram can be edited using the MC-PostServo [OB95] organization block. The MC-PostServo is called after the MC-Servo.

The data block must be created by the user and contain a data structure of data type "PD_TELx" for the data connection. Here, "x" stands for the telegram number of the drive or encoder configured in the device configuration.

The organization blocks MC-PreServo and MC-PostServo can be programmed by the user and must be added with the command "Add new block". The connection to the I/O via telegram must be programmed in this organization block. When you use DSC you have to edit the signs of life in the telegrams in MC-PreServo and MC-PostServo yourself according to the PROFIdrive standard.

See also

PROFIdrive telegrams (Page 1636)

Organization Blocks for Motion Control (Page 1697)

Connect drive/encoder via data block (Page 1908)

Tags (S7-1500)

The following technology object tags are relevant for the drives and encoder connections:

Drive telegram	
Tag	**Description**
<TO>.Actor.Interface.AddressIn	Input address for the PROFIdrive telegram
<TO>.Actor.Interface.AddressOut	Output address for the PROFIdrive telegram or the analog setpoint
<TO>.Actor.DriveParameter.ReferenceSpeed	Reference value (100%) for the speed setpoint (NSET) of the drive
<TO>.Actor.DriveParameter.MaxSpeed	Maximum value for the speed setpoint of the drive (NSET)
<TO>.Actor.DriveParameter.ReferenceTorque	Reference torque for the torque transferred as a percentage

Encoder telegram	
Tag	**Description**
<TO>.Sensor[n].Interface.AddressIn	Input address for the PROFIdrive telegram
<TO>.Sensor[n].Interface.AddressOut	Output address for the PROFIdrive telegram
<TO>.Sensor[n].System	Encoder system linear or rotary
<TO>.Sensor[n].Type	Encoder type, incremental, absolute or cyclic absolute
<TO>.Sensor[n].Parameter.Resolution	Resolution for linear encoder The grid spacing corresponds to the distance between two marks.
<TO>.Sensor[n].Parameter.StepsPerRevolution	Steps per revolution for rotary encoder
<TO>.Sensor[n].Parameter.DeterminableRevolutions	Number of differentiable encoder revolutions for a multi-turn absolute encoder

Fine resolution	
Tag	Description
<TO>.Sensor[n].Parameter.FineResolutionXist1	Number of bits for fine resolution XIST1 (cyclic actual encoder value)
<TO>.Sensor[n].Parameter.FineResolutionXist2	Number of bits for fine resolution XIST2 (absolute value of encoder)

Simulation mode		
Tag	Description	
<TO>.Simulation.Mode	Simulation mode	
	0	No simulation, normal operation
	1	Simulation mode

Mechanics (S7-1500)

For the display and processing of the technology object's position, the decisive factor is whether the position is represented as a unit of length (linear axis) or a unit of angle (rotary axis).

Examples of units of length: mm, m, km

Examples of units of angle: °, rad

For the determination of the physical position from an actual encoder value, the system must know the various properties and configurations of the mechanics.

Positioning axis/synchronous axis

The following configuration options for mechanics are supported:

- Load gear
- Leadscrew pitch (linear axes only)
- Encoder mounting type:
  - Motor side (before the load gear)
  - Load side (after the load gear and as applicable the leadscrew)
  - External (e.g. odometer)
- Inversion of drive direction
- Inversion of encoder direction

External encoder

The following configuration options for mechanics are supported:

- Measuring gear (for rotary encoders)
- Leadscrew pitch (only with linear system of units and rotary encoders)
- Inversion of encoder direction

Speed axis

The following configuration options for mechanics are supported:

- Load gear
- Inversion of drive direction

Tags (S7-1500)

The following technology object tags are relevant for the setting of the mechanics:

Type of motion		
Tag	**Description**	
<TO>.Properties.MotionType	Indication of linear or rotary motion	
	0	Linear motion
	1	Rotary motion

Load gear	
Tag	**Description**
<TO>.LoadGear.Numerator	Load gear numerator
<TO>.LoadGear.Denominator	Load gear denominator

Leadscrew pitch	
Tag	**Description**
<TO>.Mechanics.LeadScrew	Leadscrew pitch

Encoder mounting type	
Tag	**Description**
<TO>.Sensor[n].MountingMode	Encoder mounting type
<TO>.Sensor[n].Parameter.Distance-PerRevolution	Load distance per encoder revolution with an externally mounted encoder

Inversion	
Tag	**Description**
<TO>.Actor.InverseDirection	Setpoint inversion
<TO>.Actor.Efficiency	Efficiency of leadscrew pitch
<TO>.Sensor[n].InverseDirection	Actual value inversion

Modulo	
Tag	**Description**
<TO>.Modulo.Enable	Enable modulo
<TO>.Modulo.Length	Modulo length
<TO>.Modulo.StartValue	Modulo start value

Traversing range limitation (S7-1500)

Hardware and software limit switches limit the permissible traversing range and operating range of the positioning axis/synchronous axis. Before use, they must be enabled in the configuration or in the user program.

The following figure shows the relationship between the operating range, maximum traversing range and limit switches:

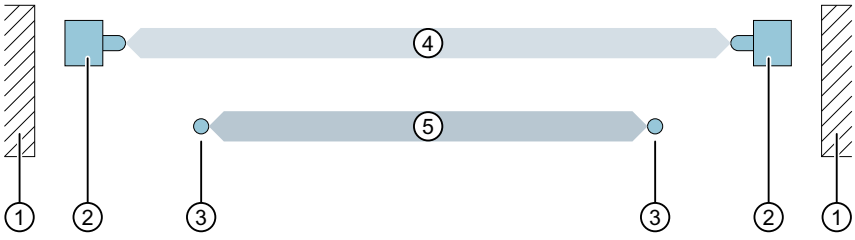

① Mechanical endstop
② Hardware limit switches
③ Software limit switches
④ Maximum traversing range
⑤ Operating range

Hardware limit switches (S7-1500)

Hardware limit switches are limit position switches that limit the maximum permissible traversing range of the axis.

Select the positions of the hardware limit switches so that there is adequate braking distance for the axis when needed. The axis should come to a standstill before a mechanical endstop.

Approaching the hardware limit switches

In the monitoring of range limitation, no distinction is made as to whether the switches are reached or crossed.

If a hardware limit switch is reached, technology alarm 531 is output, and the technology object is disabled (alarm response: remove enable).

Exception

If the hardware limit switches are used as reversing cams during homing, then the monitoring of the hardware limit switches has no effect.

When hardware limit switches are used as reversing cams, the axis is braked with the deceleration configured in the dynamic defaults.

This must be taken into account when planning the distance of the hardware limit switch to the mechanical endstop.

Retracting

The position of the axis when the hardware limit switch is detected is stored internally on the CPU. The status of the reached hardware limit switch is reset only after the hardware limit switch is left and the axis is once again in the maximum traversing range.

To retract the axis after it reaches the hardware limit switch and to reset the status of the hardware limit switch, follow the steps below:

1. To enable motion in the retraction direction, acknowledge the technology alarm.
2. Traverse the axis in the retraction direction until the hardware limit switch is left.
   The axis must then be within the maximum traversing range.
   If you move the axis opposite the retraction direction before the hardware limit switch is left, the monitoring will be triggered again.

The following chart shows the behavior of the status word when the hardware limit switch is reached and when the axis is retracted:

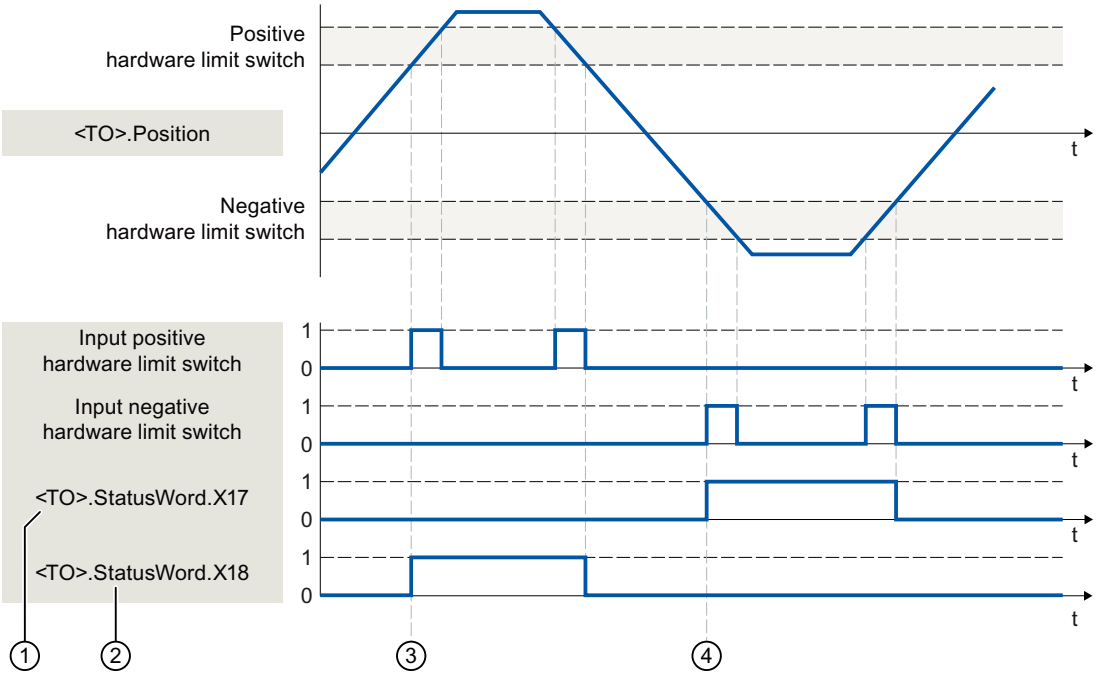

① <TO>.StatusWord.X17 (HWLimitMinActive)

0 Negative hardware limit switch not reached

1 Negative hardware limit switch reached or overtraveled

② <TO>.StatusWord.X18 (HWLimitMaxActive)

0 Positive hardware limit switch not reached

1 Positive hardware limit switch reached or overtraveled

③ The position of the axis when the **positive** hardware limit switch is detected is saved internally in the CPU. To reset the status of the hardware limit switch, the axis position must fall short of this position.

④ The position of the axis when the **negative** hardware limit switch is detected is saved internally in the CPU. To reset the status of the hardware limit switch, the axis position must go past this position.

Software limit switch (S7-1500)

The operating range of the axis is limited with software limit switches. Relative to the traversing range, always position the software limit switches within the hardware limit switches. Since the positions of the software limit switches can be flexibly configured, the operating range of the axis can be individually adapted in accordance with the current velocity profile.

Software limit switches are only in effect when there is a valid actual value after homing the technology object. The monitoring of the software limit switches is relative to the setpoint.

Modulo function is enabled

When the modulo function is enabled, the modulo position is monitored.

The software limit switches are configured and activated in the axis configuration. The software limit switches can be activated or deactivated in the user program using the

"<TO>.PositionLimits_SW.Active" tag. If the positions of both software limit switches are outside the modulo range, the monitoring has no effect. No check is made to determine whether the positions of the software limit switches are within the modulo range.

Approaching the software limit switches

The axis continually checks the position of the software limit switch during motion and brakes to exactly this position, if necessary.

If the software limit switches are reached, then technology alarm 533 is output, and the axis is stopped with the maximum dynamic values (alarm response: Stop with maximum dynamic values). The technology object remains enabled.

Overrun of the software limit switches

If a software limit switch is crossed, technology alarm 534 is output, and the technology object is disabled (alarm response: remove enable).

Retracting

To retract the axis after violation of the software limit switch, follow the steps below:

1. Acknowledge the technology alarm.
2. Move the axis in the retraction direction until the software limit switch is left.
   If you move the axis opposite the retraction direction before the software limit switch is left, the monitoring with be triggered again.

Tags (S7-1500)

The following technology object tags are relevant for software limit switches:

Status indicators	
Tag	**Description**
<TO>.StatusWord.X15 (SWLimitMinActive)	Negative software limit switch is active.
<TO>.StatusWord.X16 (SWLimitMaxActive)	Positive software limit switch is active.
<TO>.ErrorWord.X8 (SWLimit)	An alarm is pending, indicating that a software limit switch has been violated.

Control bits	
Tag	**Description**
<TO>.PositionLimits_SW.Active	Enables/disables the monitoring of the software limit switches.

Position values	
Tag	Description
<TO>.PositionLimits_SW.MinPosition	Position of the negative software limit switch
<TO>.PositionLimits_SW.MaxPosition	Position of the positive software limit switch

The following technology object tags are relevant for hardware limit switches:

Status indicators	
Tag	Description
<TO>.StatusWord.X17 (HWLimitMinActive)	Negative hardware limit switch is active.
<TO>.StatusWord.X18 (HWLimitMaxActive)	Positive hardware limit switch is active.
<TO>.ErrorWord.X9 (HWLimit)	An alarm is pending, indicating that a hardware limit switch has been violated.

Control bits	
Tag	Description
<TO>.PositionLimits_HW.Active	Enables/disables the monitoring of the hardware limit switches.

Parameters		
Tag	Description	
<TO>.PositionLimits_HW.MinSwitchLevel	Level selection for activation of the low hardware limit switch	
	FALSE	At low level, the signal is active.
	TRUE	At high level, the signal is active.
<TO>.PositionLimits_HW.MinSwitchAddress	Byte number of the I/O address of the hardware limit switch for the low or minimum position	
<TO>.PositionLimits_HW.MaxSwitchLevel	Level selection for activation of the high hardware limit switch	
	FALSE	At low level, the signal is active.
	TRUE	At high level, the signal is active.
<TO>.PositionLimits_HW.MaxSwitchAddress	Byte number of the I/O address of the hardware limit switch for the high or maximum position	

Motion control and limits for dynamics (S7-1500)

Motion control of the axis occurs by means of velocity profiles (Page 1661). The velocity profiles are calculated in accordance with the specifications for dynamics. A velocity profile defines the behavior of the axis during approach, braking and changes in velocity. During positioning a velocity profile is calculated, that moves the axis to the target point.

Maximum values for velocity, acceleration and jerk result from the properties of the drive and the mechanics. These maximum values can be configured in the limits for dynamics. The limits for dynamics are in effect as limits for every motion generated by means of the technology object. The dynamic limits have no effect on a following axis in synchronous operation.

The configurable emergency stop deceleration (Page 1662) is triggered by the Motion Control instruction MC_Power or by a technology alarm.

The jerk limit reduces the mechanical load during an acceleration ramp or deceleration ramp. A "smoothed" velocity profile results.

See also

Dynamic limits in synchronous operation (Page 1663)

Velocity profile (S7-1500)

Velocity profiles with or without jerk limitation are supported for motion control of the axis.

The dynamic values for the motion are specified in the Motion Control job. Alternatively, the values of the dynamic defaults (Page 1811) can be used. The defaults and the limits for velocity, acceleration, deceleration and jerk are set in the configuration.

To influence velocity, a velocity override can override the current traversing velocity.

Velocity profile without jerk limitation

The following figure shows velocity, acceleration and jerk:

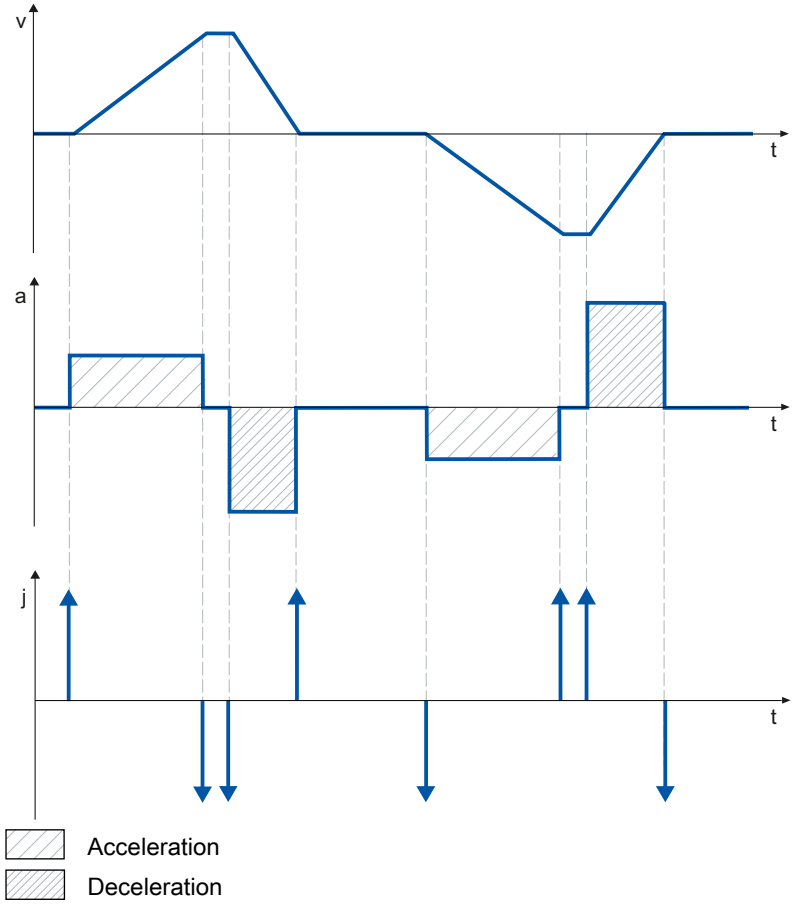

Velocity profile with jerk limitation

The following figure shows velocity, acceleration and jerk:

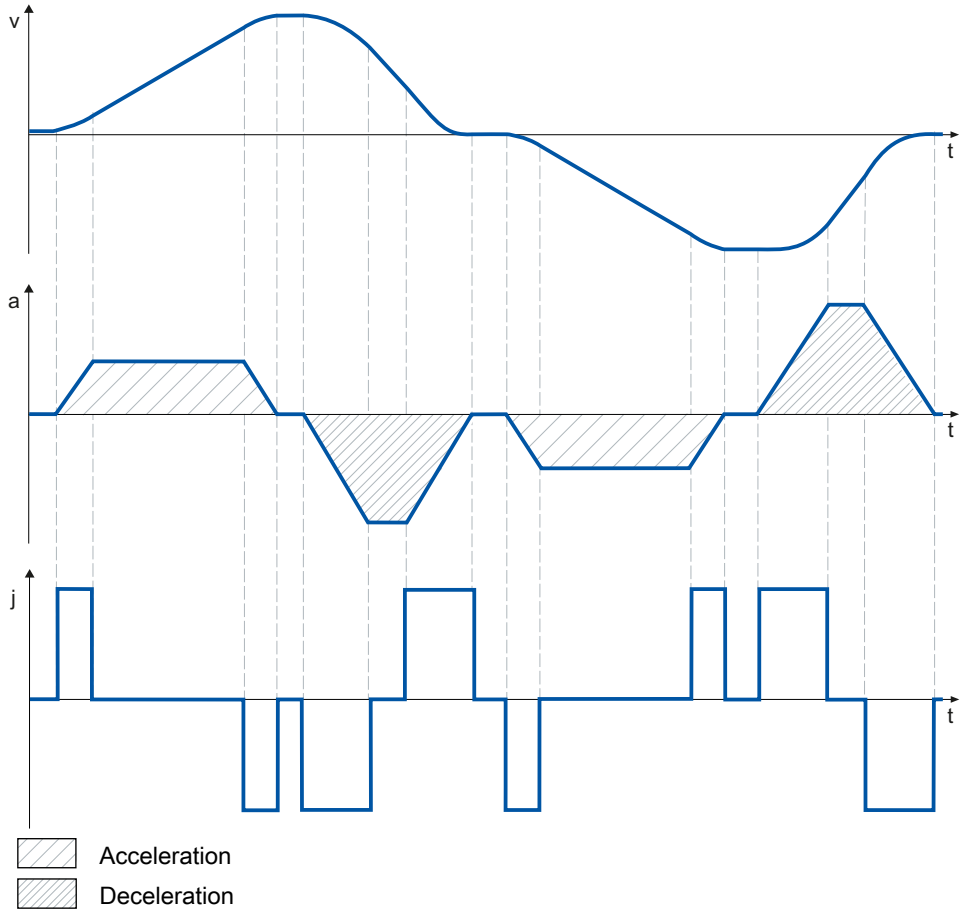

A velocity profile with jerk limitation is employed for a continuous acceleration and deceleration sequence. The jerk can be specified.

Emergency stop deceleration (S7-1500)

When stopping with the emergency stop ramp, the axis is braked to a standstill without a jerk limit, using the configured emergency stop deceleration.

In the following cases the configured emergency stop deceleration is in effect:

- For an emergency stop ramp that has been enabled via the Motion Control instruction "MC_Power" with parameter "StopMode" = 0.
- For a technology alarm with the local alarm response "Stop with emergency stop ramp".

This emergency stop deceleration can be set greater than the maximum deceleration. If the emergency stop deceleration is set lower than this, it may occur that the axis does not stop until after the limit switch in the case of "Stop at software limit switch" and the occurrence of a technology alarm with the local alarm response "Stop with emergency stop ramp".

Dynamic limits in synchronous operation (S7-1500)

Dynamic limits in synchronous operation with "MC_GearIn" (S7-1500)

Leading axis

The dynamic limits configured for the technology object are always in effect for the leading axis.

Following axis

If a synchronous axis is operated as a following axis in synchronous operation with "MC_GearIn", the following dynamic limits apply depending on the phase of the synchronous operation:

- Synchronization
  During the synchronizing phase, dynamic limits configured for the technology object apply to the following axis.
- Synchronous motion
  When the synchronous axis is moving synchronously to the leading axis as a following axis, the dynamics of the following axis is limited only to the maximum speed of the drive (<TO>.Actor.DriveParameter.MaxSpeed). The dynamics of the following axis results from the synchronous operation function.
  If the dynamic limits configured for the following axis are exceeded, this will be indicated in the "<TO.>StatusSynchronizedMotion.StatusWord" tag of the technology object.
  If the following axis cannot follow the leading value, this results in a following error, which is monitored by the following error monitoring.
- Synchronous operation override
  As soon as synchronous operation has been overridden, the dynamic limits configured for the technology object apply to the following axis again. With the start of the overriding job, the active dynamics is transitioned (smoothed) to the configured dynamic limits and the specifications for the Motion Control instruction.

See also

"StatusSynchronizedMotion" tag (synchronous axis) (Page 2054)

Dynamic limits in synchronous operation with "MC_GearInPos"/"MC_CamIn"

Leading axis

The dynamic limits configured for the technology object are always in effect for the leading axis.

Following axis

If a synchronous axis is operated as following axis in synchronous operation with the Motion Control instruction "MC_GearInPos" or "MC_CamIn", the following dynamic limits apply depending on the phase of the synchronous operation:

- **Pending synchronous operation**
  If synchronous operation is not active, the configured dynamic limits apply. If a synchronous operation is already active, these limits are overridden by the previous synchronous operation.
- **Synchronization/synchronous motion**
  During synchronization/synchronous motion, the dynamics of the following axis is limited only to the maximum speed of the drive (<TO>.Actor.DriveParameter.MaxSpeed). The dynamics of the following axis results from the synchronous operation function.
  If the dynamic limits configured for the following axis are exceeded, this will be indicated in the tag of the technology object "<TO.>StatusSynchronizedMotion.StatusWord".
  If the following axis cannot follow the leading value, this results in a following error, which is monitored by the following error monitoring.
- **Synchronous operation override**
  As soon as synchronous operation has been overridden, the dynamic limits configured for the technology object apply to the following axis again. With the start of the overriding job, the active dynamics is transitioned (smoothed) to the configured dynamic limits and the specifications for the Motion Control instruction.

See also

"StatusSynchronizedMotion" tag (synchronous axis) (Page 2054)

Motion specification via "MotionIn"

In contrast to the Motion Control instructions such as "MC_MoveAbsolute" and "MC_MoveRelative", no motion profile is calculated by the system when "MC_MotionInVelocity" and "MC_MotionInPosition" are used. Each individual setpoint of the motion profile (motion vector) must be specified with the "MotionIn" instruction in the application cycle. This allows you to calculate your own motion profile.

The setpoints are typically adapted in the processing cycle of the technology object. For this purpose, the "MotionIn" instruction is called in the organization block MC-PreServo [OB67] or

MC-PostServo [OB95]. The setpoints are then transferred directly to the axis in MC-Interpolator [OB92].

WARNING

Uncontrolled axis motions

When using the motion specification via the Motion Control instructions "MC_MotionInVelocity" and "MC_MotionInPosition", the axis can perform uncontrolled motions.

Consider the current dynamics of the axis when specifying the new motion vectors. The motion vectors must be consistent with each other.

Set up the following protective measures before operating with the Motion Control instructions "MC_MotionInVelocity" and "MC_MotionInPosition":

- Ensure that the EMERGENCY OFF switch is within the reach of the operator.
- Enable the hardware limit switches.
- Enable the software limit switches.
- Ensure that following error monitoring is enabled.

Note that a following axis that is coupled to the axis is also moved.

Overriding with "MotionIn" instructions

If a Motion Control instruction is overridden by a "MotionIn" instruction, the specified setpoints take immediate effect with the current application cycle. The dynamic results exclusively through the setpoint specifications of the user program. It is not limited and no smooth transition takes place from the current motion state. Consider the current dynamics of the axis when specifying the new motion vectors. Note that dynamic limits set on the technology object have no effect. Only limits set on the drive side are in effect.

Stopping the "MotionIn" instructions

The "MotionIn" instructions can be canceled by the following means:

- Overriding them with another Motion Control instruction
  The "MotionIn" instructions are overridden according to the behavior described in the section AUTOHOTSPOT. As a rule, the current dynamics are overridden by the new motion.
- Setting the parameter "Enable" to "FALSE"
  If you set "Enable" parameter to "FALSE", the setpoint is immediately set to zero. Note that the dynamic limits set on the technology object have no effect. Only limits set on the drive side are in effect.

"MC_MotionInVelocity"

Use the "MC_MotionInVelocity" instruction to specify the velocity and acceleration of the motion. The instruction is applicable for speed, positioning and constant axes.

To execute the instruction, you must at least specify the velocity. Acceleration is usually only required for the substituting running motions. By default, the value of the acceleration is zero.

"MC_MotionInPosition"

Use the "MC_MotionInPosition" instruction to specify the position, velocity and acceleration of the motion. The instruction is used for velocity, positioning and synchronous axes.

To execute the instruction, you must at least specify the position and velocity. Acceleration is usually only required for the substituting running motions. By default, the value of the acceleration is zero. The specified setpoints must be consistent with each other.

The position specification is position-controlled. If you use a velocity precontrol, the velocity specification is processed via the velocity precontrol.

Torque limits (S7-1500)

Force/torque limiting (S7-1500)

Adjustable force/torque limiting is available for the speed axis, positioning axis and synchronous axis technology objects. The force/torque limiting can be activated and deactivated before and during a motion job. To use force/torque limiting, the drive and the PROFIdrive telegram must support torque reduction. You can use, for example, a telegram 10x.

The limit value can be configured as a default value during configuration of the axis or it can be defined in the user program using Motion Control instruction "MC_TorqueLimiting".

The user specifies the limiting values in the configured unit of measure for force or torque. The units of measure are defined in the "Basic parameters" configuration window.

The following configuration options are available for force/torque limiting.

- "Linear" axis type
  - Torque limiting is active on motor side
  - Force limiting is active on the load side
- "Rotary" axis type
  - Torque limiting is active on load side or motor side

The force/torque limit defined by the user in accordance with the specification in the PROFIdrive telegrams 10x are transferred internally to the drive as a percentage torque reduction. The reference torque set in the "Data exchange with the drive" configuration dialog must match the reference torque set for the drive.

Linear axis type

Load-side force limitation you have defined is converted by the technology into torque reduction. If the limiting relates to the load side, the gear and leadscrew parameters defined in the "Mechanics" configuration dialog are taken into consideration. If the efficiency of the gear and leadscrew is crucial, you can set them in the "<TO>.Actor.Efficiency" tag.

Rotary axis type

The torque is reduced on the load side with the rotary axis type. The gear parameters defined in the "Mechanics" configuration window are taken into consideration. If the efficiency of the gear is crucial, you can set it in the "<TO>.Actor.Efficiency" tag.

The defined limiting values act as an absolute value and thus in the same way for positive and negative forces/torques.

Positioning and following error monitoring with active force/torque limiting

As a result of force/torque limiting, a larger setpoint-actual value difference can build up for position-controlled axes, which may cause unwanted activation of the positioning and following error monitoring.

Therefore, in the "Torque limiting" configuration window, the positioning and following error monitoring of the axis can be set as deactivated by default when force/torque limiting is active. If necessary, the positioning and following error monitoring can also be kept active even when force/torque limiting is active.

Typical behavior of a positioning or synchronous axis with active force/torque limiting

With active force/torque limiting, a larger setpoint-actual value difference can build up than during motion without force/torque limiting.

Given a constant setpoint, the axis makes repeated attempts to reduce the following error.

When the limiting values are increased or limiting is deactivated during active closed loop position control, the axis can accelerate briefly to reduce the following error. If the axis is switched to non-position-controlled operation, e.g. using "MC_MoveVelocity" with "PositionControlled" = FALSE, the following error is no longer in effect.

Stopping an axis with active force/torque limiting

When stopping an axis in position-controlled mode with "MC_Halt", the position setpoint and the velocity setpoint are used as basis. Torque limiting still remains active and any accumulated following error is reduced. The axis is in standstill when the actual velocity "0.0" is reached and the minimum dwell time in the standstill window has expired. The axis remains enabled.

When stopping an axis with "MC_Power" and an emergency stop ramp, the actual position value and the actual velocity are used as a basis. The axis is braked with the configured emergency stop deceleration without any jerk limit and brought to a standstill. The axis is then disabled when at a standstill.

When the limitation is deactivated the drive attempts to compensate for the resulting following error with maximum dynamic values. To avoid this reaction, use "MC_MoveVelocity" with deactivated closed loop position control.

See also

Fixed stop detection (Page 1668)

Configuration - Torque limits (Page 1795)

Configuration - Torque limits (Page 1818)

Fixed stop detection (S7-1500)

With the Motion Control instruction "MC_TorqueLimiting", you activate and monitor a fixed stop detection. Together with a position-controlled motion job, a "Travel to fixed stop" can be realized. The operation is also referred to as clamping. "Travel to fixed stop" can be used, for example, to move quills against the workpiece with a specified torque.

The fixed stop detection is configured in the configuration window "Extended parameters > Limits > Fixed stop detection".

The fixed stop detection is only possible in position-controlled operation of the axis. If the drive and telegram support force/torque limiting, this is active during travel to fixed stop and for clamping.

Detection of the fixed stop using following error

If the drive is stopped by a mechanical fixed stop during a motion job, the following error is increased. When the following error configured in the configuration window "Extended parameters > Limits > Fixed stop detection" is exceeded, this is regarded as the fixed stop having been reached.

When following error monitoring is activated, the configured following error must be greater than the following error for fixed stop detection.

Clamping at the mechanical endstop

When the fixed stop is reached, the active position-controlled motion job is canceled with "CommandAborted". The setpoint is no longer changed and the following error remains constant. The closed loop position control remains active and the monitoring of the configured "Positioning tolerance" is activated The drive is in "Clamping" state.

If the drive and telegram support force/torque limiting, this continues to be active with active fixed stop detection. During clamping, the clamping force or clamping torque can be changed. The value in input parameter "Limit" of the Motion Control instruction "MC_TorqueLimiting" can be changed for this.

Monitoring of the clamping

If the actual position changes by a value greater than the configured "Positioning tolerance" during active clamping, this is regarded as the break or push-back of the fixed stop. An alarm is triggered. The axis is disabled and the drive is stopped according to its configuration.

If the position setpoint is within the configured "Positioning tolerance", the breaking away or turning back of the fixed stop cannot be detected.

The configured position tolerance must be less than the configured following error for detection of clamping.

Retracting

Retracting from the fixed stop is only possible with a position-controlled motion job in the opposite direction to the fixed stop.

The "Travel to fixed stop" or "Clamping" function is ended when the "Positioning tolerance" is left in the retraction direction.

See also

Force/torque limiting (Page 1665)

Configuration - Fixed stop detection (Page 1819)

Additive setpoint torque (S7-1500)

The Motion Control instruction "MC_TorqueAdditive" allows you to apply additional torque in the drive.

The additive setpoint torque is used for example in torque feedforward control or the specification of the tensile torque for winding applications.

The following requirements must be fulfilled to set the additive torque setpoint:

- SINAMICS drive (see compatibility list (Page 2146))
- SIEMENS supplementary telegram 750 for transmitting the torque data to the drive

The additional torque can be either positive or negative. The value specified in the instruction is a technological value, not a percentage. Set the unit of measurement for the torque at the axis (default value: Nm).

Permissible torque range (S7-1500)

The Motion Control instruction "MC_TorqueRange" allows you to set torque limits for the drive.

The motion control instruction is used, for example, for winding applications in order to prevent the tearing of the material.

The following requirements must be fulfilled to set the torque data:

- SINAMICS drive (see compatibility list (Page 2146))
- SIEMENS supplementary telegram 750 for transmitting the torque data to the drive

The value specified in the instruction is a technological value, not a percentage. Set the unit of measurement for the torque at the axis (default value: Nm). If you invert the setpoints at the technology object of the axis, the values for the high and low torque limit are output inverted and reversed.

If the torque limitation is activated by specifying the high and low torque limit, the following monitorings and limits are deactivated:

- Following error monitoring
- Time limits for positioning monitoring
- Time limits for standstill monitoring

Monitoring remains in effect if you have selected the option "Leave position-related monitoring enabled" under "Technology object > Configuration > Extended parameters > Limits > Torque limiting".

Tags (S7-1500)

The following technology object tags are relevant for motion control:

Status		
Tag	**Description**	
<TO>.StatusWord	Status indicator for an active motion	
<TO>.Position	Position setpoint	
<TO>.Velocity	Velocity setpoint/speed setpoint	
<TO>.ActualPosition	Actual position	
<TO>.ActualVelocity	Actual velocity	
<TO>.ActualSpeed	Actual speed of the motor (only with PROFIdrive drive type)	
<TO>.Acceleration	Setpoint acceleration	
<TO>.ActualAcceleration	Actual acceleration	
<TO>.StatusSynchronizedMotion.StatusWord.X0 (MaxVelocityExceeded)	The tag is set to the value "TRUE" when the maximum velocity configured for the following axis is exceeded during synchronous operation.	
<TO>.StatusSynchronizedMotion.StatusWord.X1 (MaxAccelerationExceeded)	The tag is set to the value "TRUE" when the maximum acceleration configured for the following axis is exceeded during synchronous operation.	
<TO>.StatusSynchronizedMotion.StatusWord.X2 (MaxDecelerationExceeded)	The tag is set to the value "TRUE" when the maximum deceleration configured for the following axis is exceeded during synchronous operation.	
<TO>.StatusMotionIn.FunctionState	Status of the "MotionIn" function	
	0	No "MotionIn" function active
	1	"MC_MotionInVelocity" active
	2	"MC_MotionInPosition" active

Override	
Tag	**Description**
<TO>.Override.Velocity	Velocity or speed override

Dynamic limit values	
Tag	**Description**
<TO>.DynamicLimits.MaxVelocity	Dynamic limitation for maximum speed (mechanical)
<TO>.DynamicLimits.Velocity	Dynamic limitation for maximum speed (programmable)
<TO>.DynamicLimits.MaxAcceleration	Dynamic limitation for maximum acceleration
<TO>.DynamicLimits.MaxDeceleration	Dynamic limitation for maximum deceleration
<TO>.DynamicLimits.MaxJerk	Dynamic limitation for maximum jerk

Defaults for the dynamics	
Tag	**Description**
<TO>.DynamicDefaults.Velocity	Default velocity
<TO>.DynamicDefaults.Acceleration	Default acceleration
<TO>.DynamicDefaults.Deceleration	Default deceleration

Defaults for the dynamics	
Tag	**Description**
<TO>.DynamicDefaults.Jerk	Default jerk
<TO>.DynamicDefaults.EmergencyDeceleration	Emergency stop deceleration

Torque limiting		
Tag	**Description**	
<TO>.TorqueLimiting.LimitDefaults.Torque	Limiting torque	
<TO>.TorqueLimiting.LimitDefaults.Force	Limiting force	
<TO>.TorqueLimiting.LimitBase	Torque limiting motor or drive side	
	0	Motor side
	1	Load side
<TO>.TorqueLimiting.PositionBasedMonitorings	Positioning and following error monitoring	
	0	Disabled
	1	Enabled
<TO>.StatusTorqueData.CommandAdditiveTorqueActive	Additive torque setpoint function	
	0	Disabled
	1	Enabled
<TO>.StatusTorqueData.CommandTorqueRangeActive	Torque limits function	
	0	Disabled
	1	Enabled
<TO>.StatusTorqueData.ActualTorque	Actual torque of the axis	

Fixed stop detection	
Tag	**Description**
<TO>.Clamping.FollowingErrorDeviation	Value of the following error starting from which the fixed stop is detected
<TO>.Clamping.PositionTolerance	Position tolerance for the clamping monitoring

Homing (S7-1500)

With homing, you create the relationship between the position in the technology object and the mechanical position. The actual position value in the technology object is assigned to a homing mark at the same time. This homing mark represents a known mechanical position.

With incremental actual values this process is called homing; with absolute actual values it is called absolute encoder adjustment.

Homing is a requirement for display of the correct position for the technology object and for absolute positioning.

Homing is activated with the Motion Control instruction "MC_Home".

Homing status

The "<TO>.StatusWord.X5 (HomingDone)" tag of the technology object indicates whether the axis or external encoder technology object is homed.

Type of homing

Homing can occur by means of an independent homing motion (active homing), the detection of a homing mark during a motion initiated on the user side (passive homing) or a direct position assignment.

A distinction is made between the following types of homing:

- **Active homing**
  Active homing initiates a homing movement and performs the necessary homing mark approach. When the homing mark is detected, the actual position is set to the value specified in MC_Home. It is possible to specify a home position offset. Retraction to the home position offset occurs automatically during the home position approach.
  When active homing starts, current traversing movements are aborted.
- **Passive homing**
  The homing job does not perform its own homing motion. When the homing mark is detected during a motion initiated on the user side, the actual position is set to the value specified in "MC_Home".
  Passive homing is also called homing on the fly.
- **Direct homing**
  With the homing job, the actual position is set directly to the value specified in "MC_Home" or is offset by this value.
- **Absolute encoder adjustment**
  The absolute encoder adjustment compares the existing absolute actual value with the value specified in "MC_Home".

Homing mode

Depending on the type of homing mark and homing mark search, a distinction is made among the following homing modes (Page 1673):

- Homing with homing cam and zero mark via PROFIdrive telegram
- Homing with zero mark via PROFIdrive telegram
- Homing with digital input

Terms (S7-1500)

Homing mark

A homing mark is an input signal, on whose occurrence a known mechanical position can be assigned to the actual values.

A homing mark can be:

- **A zero mark**
  The zero mark of an incremental encoder or an external zero mark is used as a homing mark.
  The zero mark is detected at the drive module or encoder module and transmitted in the PROFIdrive telegram. Perform the setting and evaluation as an encoder zero mark or external zero mark at the drive module and encoder module.
- **An edge at the digital input**
  The negative or positive edge at a digital input is used as a homing mark.

Reference cam

If there are several zero marks in the traversing range, the reference cam is used to select a specific zero mark before or after the reference cam.

Homing mark position

This is the position assigned to the homing mark.

The homing mark position corresponds to the home position minus the home position offset.

Home position

At the end of the active homing motion, the axis arrives at the home position.

Home position offset

The difference between the home position and the homing mark position is the home position offset.

A home position offset only has an effect with active homing. The offset is traversed after the synchronization of the axis using the Motion Control instruction "MC_Home". For axes with modulo setting, the home position offset is always traversed with the direction setting for the shortest path.

Direction reversal at the hardware limit switch (reversing cam)

Hardware limit switches can be used as reversing cams in active homing. If the homing mark was not detected or was approached from the wrong side, the motion continues in the opposite direction after the reversing cam.

Homing mode (S7-1500)

Various homing modes are available for the positioning axis/synchronous axis and external encoder technology objects with incremental encoders. The homing mode is set in the configuration.

Homing with homing cam and zero mark via PROFIdrive telegram

The system checks for when the reference cam is reached. After the reference cam has been reached and left again in the assigned homing direction, zero mark detection is enabled via the PROFIdrive telegram.

When the zero mark is reached in the pre-selected direction, then the actual position of the technology object is set to the homing mark position.

Homing with zero mark via PROFIdrive telegram

The system enables zero mark detection, as soon as the actual value of the technology object moves in the assigned homing direction.

When the zero mark is reached in the specified homing direction, the actual position of the technology object is set to the homing mark position.

Homing with digital input

The system checks the state of the digital input, as soon as the actual value of the axis or encoder moves in the assigned homing direction.

When the homing mark is reached (setting of the digital input) in the specified homing direction, the actual position of the technology object is set to the homing mark position.

Note

The digital inputs must be placed into the process image partition "PIP OB Servo".

The filter time of the digital inputs must be set smaller than the duration of the input signal at the reference point switch.

See also

Homing SINAMICS drives with external zero marks (Page 2147)

Active homing with homing cam and zero mark (S7-1500)

The following examples show homing motions in the positive and negative directions.

Example of homing in the positive direction

The approach to the homing mark and the home position occurs in the positive direction.

The following figure shows the homing motion with the following settings:

- Active homing with homing cam and zero mark
- Approach in the positive direction
- Homing in the positive direction
- Positive home position offset

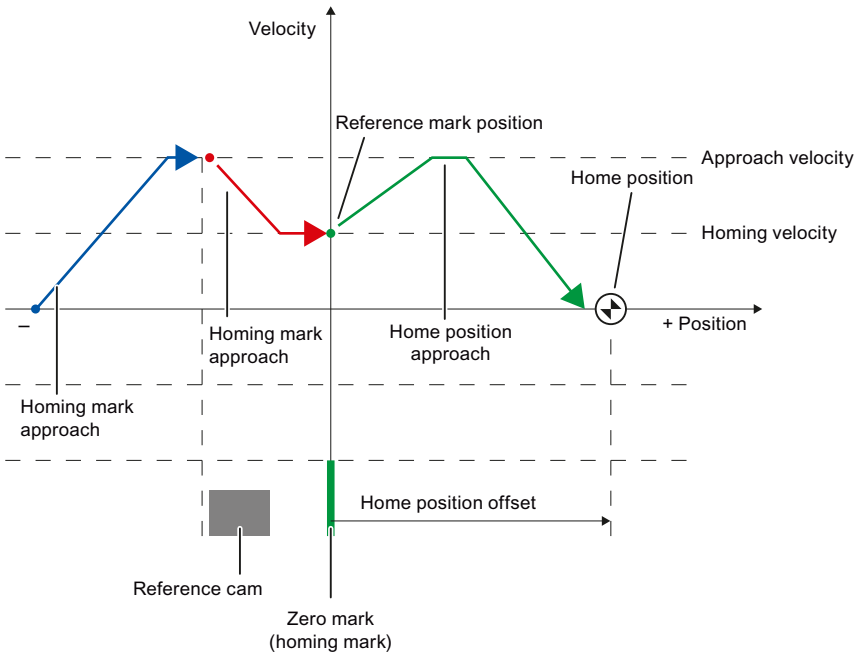

Example of homing in the negative direction

The approach to the homing mark occurs in the negative direction by means of a direction reversal during the homing process. The approach to home position causes another direction reversal and occurs in the positive direction.

The following figure shows the homing motion with the following settings:

- Active homing with homing cam and zero mark
- Approach in the positive direction
- Homing in the negative direction
- Positive home position offset

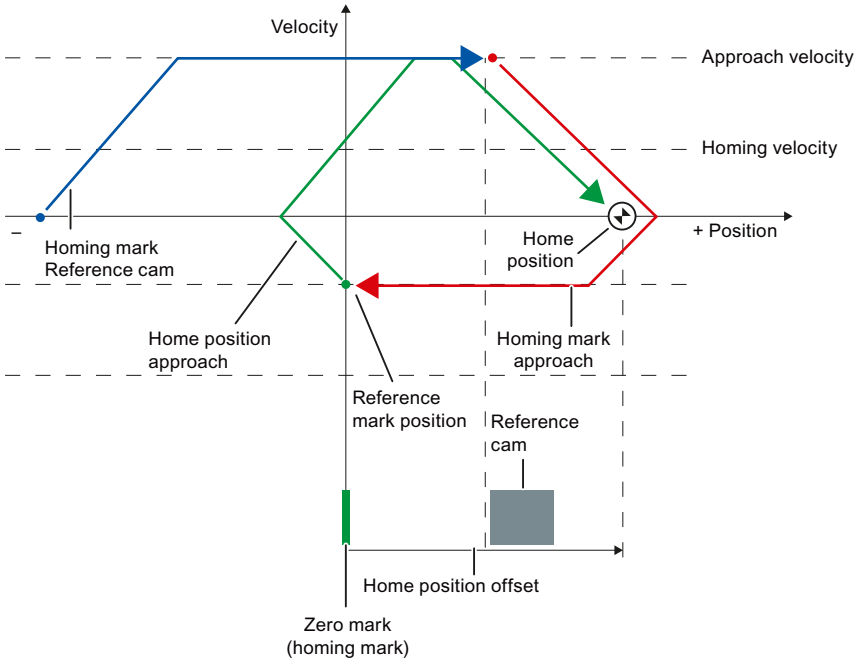

Motion sequence

The motion occurs in the following sequence:

1. Start of active homing via the "MC_Home" Motion Control instruction
2. Approach to reference cam
3. Detection of the reference cam in the homing direction and travel with homing velocity
4. Departure from the reference cam and travel to the homing mark
   With the departure from the reference cam, the detection of the homing mark is enabled.

5. Detection of the homing mark
   When the homing mark is detected, the position of the technology object is set depending on the configured mode:
   - "Mode" parameter at "MC_Home" = 3
     Position = value in parameter "Position" minus "<TO>.Sensor[n].ActiveHoming.HomePositionOffset"
   - Parameter "Mode" in "MC_Home" = 5
     Position = value in tag "<TO>.Homing.HomePosition" minus "<TO>.Sensor[n].ActiveHoming.HomePositionOffset"

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1766).

6. Approach to the home position
   - "Mode" parameter at "MC_Home" = 3
     The axis moves to the position that is specified in the "Position" parameter.
   - "Mode" parameter at "MC_Home" = 5
     The axis moves to the position that is specified in the "<TO>.Homing.HomePosition" tag.

Note

If the velocity cannot be reduced to the reference velocity between detection of the reference cam and the zero mark, homing is performed at the velocity present when the zero mark is crossed.

See also

Homing SINAMICS drives with external zero marks (Page 2147)

Active homing with zero mark (S7-1500)

The following figure shows an example of the homing motion with the following settings:

- Active homing with zero mark
- Homing in the positive direction
- Positive home position offset

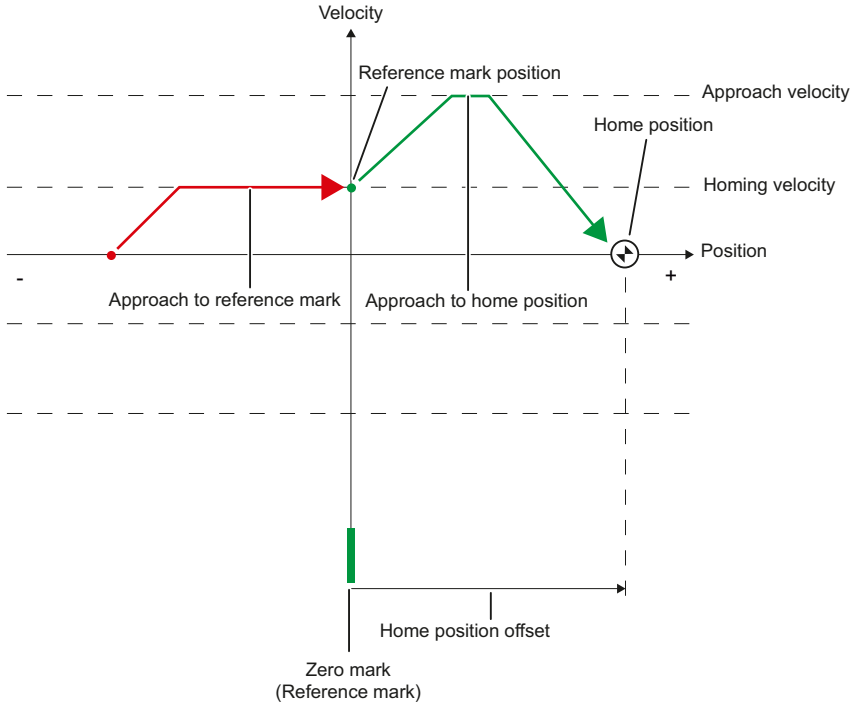

Motion sequence

The motion occurs in the following sequence:

1. Start of active homing via the Motion Control instruction "MC_Home"
2. Move to the homing mark in the homing direction with the homing velocity

3. Detection of the homing mark
   When the homing mark is detected, the position of the axis or encoder is set depending on the configured mode:
   - "Mode" parameter at "MC_Home" = 3
     Position = value in parameter "Position" minus "<TO>.Sensor[n].ActiveHoming.HomePositionOffset"
   - "Mode" parameter at "MC_Home" = 5
     Position = value in tag "<TO>.Homing.HomePosition" minus "<TO>.Sensor[n].ActiveHoming.HomePositionOffset"

   Note

   Parameter "MC_Home.Mode"

   The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1766).

4. Approach to the home position
   - "Mode" parameter at "MC_Home" = 3
     The axis moves to the position that is specified in the "Position" parameter.
   - "Mode" parameter at "MC_Home" = 5
     The axis moves to the position that is specified in the "<TO>.Homing.HomePosition" tag.

See also

Homing SINAMICS drives with external zero marks (Page 2147)

Active homing with digital input (S7-1500)

The following figure shows an example of the homing motion with the following settings:

- Active homing with digital input
- Approach in the positive direction

- Homing mark on the positive side of the digital input
- Positive home position offset

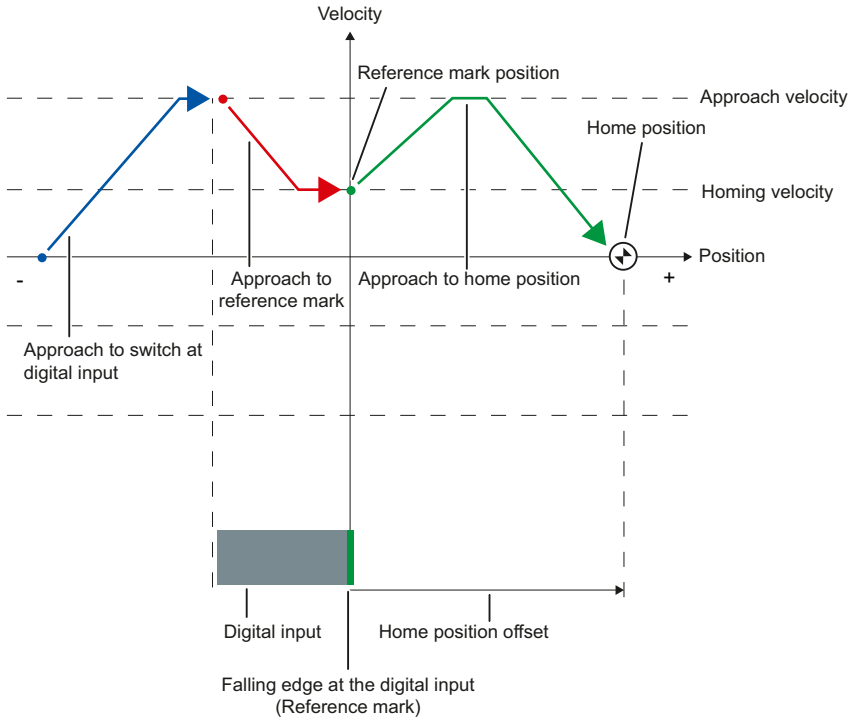

Motion sequence

The motion occurs in the following sequence:

1. Start of active homing via the "MC_Home" Motion Control instruction
2. Detection of the positive edge at the digital input, while moving with homing velocity
3. Approach to the homing mark

4. Detection of the homing mark
   In the example, the negative edge of the switch at the digital input represents the homing mark.
   When the homing mark is detected, the position of the axis or encoder is set depending on the configured mode:
   - "Mode" parameter at "MC_Home" = 3
     Position = value in parameter "Position" minus "<TO>.Sensor[n].ActiveHoming.HomePositionOffset"
   - "Mode" parameter at "MC_Home" = 5
     Position = value in tag "<TO>.Homing.HomePosition" minus "<TO>.Sensor[n].ActiveHoming.HomePositionOffset"

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1766).

5. Approach to the home position
   - "Mode" parameter at "MC_Home" = 3
     The axis moves to the position that is specified in the "Position" parameter.
   - "Mode" parameter at "MC_Home" = 5
     The axis moves to the position that is specified in the "<TO>.Homing.HomePosition" tag.

Note

If the velocity on the span from the detection of the positive edge to the negative edge cannot be reduced to the homing velocity, then homing occurs at the velocity that exists when the homing mark is traversed.

Passive homing with homing cam and zero mark (S7-1500)

The following figure shows an example of the homing motion with the following settings:

- Passive homing with homing cam and zero mark
- Homing in the positive direction

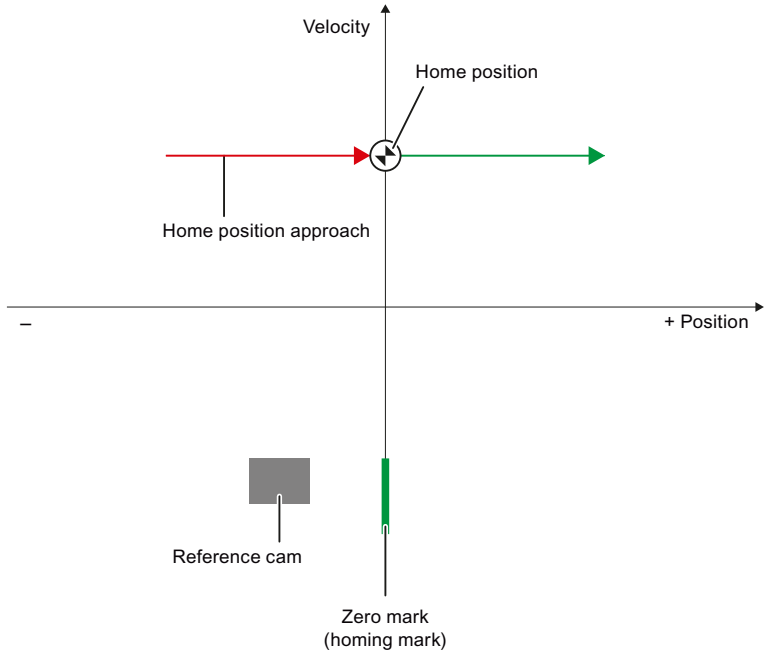

Motion sequence

The motion occurs in the following sequence:

1. Activation of passive homing using the Motion Control instruction "MC_Home"
2. Motion due to a Motion Control job from the user
   The detection of the reference cam and homing mark is enabled when the actual position value of the axis or encoder moves in the assigned homing direction.
3. Detection of the reference cam

4. Departure from the reference cam
   The departure from the reference cam enables the detection of the homing mark.
5. Detection of the homing mark
   When the homing mark is detected, the position of the axis or encoder is set depending on the configured mode:
   – "Mode" parameter at "MC_Home" = 2, 8
     Position = value in parameter "Position"
   – "Mode" parameter at "MC_Home" = 10
     Position = value in tag "<TO>.Homing.HomePosition"

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1766).

Note

If the motion direction changes after departure from the reference cam and before detection of the homing mark, the reference cam must be detected again. The Motion Control instruction "MC_Home" remains enabled.

Passive homing with zero mark (S7-1500)

The following figure shows an example of the homing motion with the following settings:

- Passive homing with zero mark
- Homing in the positive direction

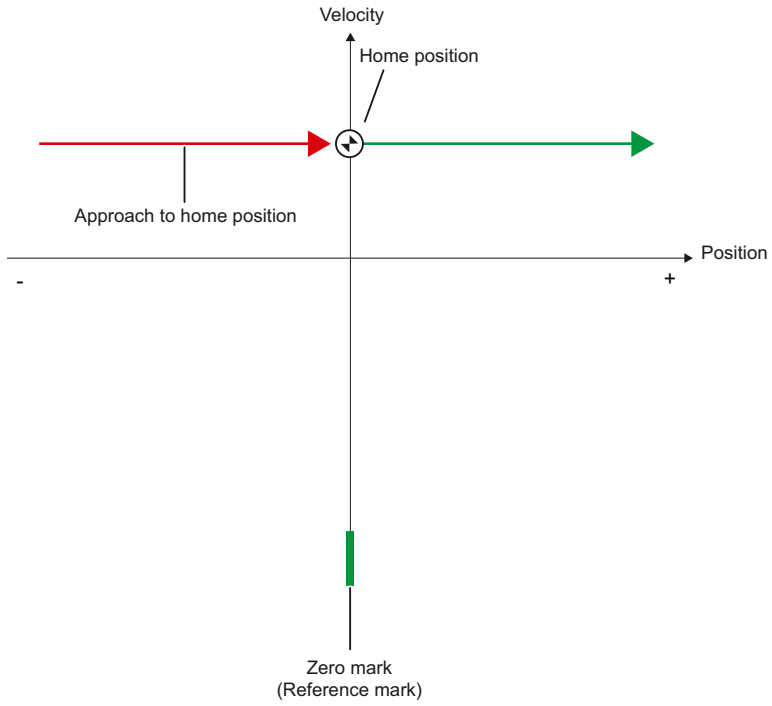

Motion sequence

The motion occurs in the following sequence:

1. Activation of passive homing using the Motion Control instruction "MC_Home"
2. Motion due to a Motion Control job from the user
   The detection of the homing mark is enabled when the actual position value of the axis or encoder moves in the assigned homing direction.
3. Detection of the homing mark
   When the homing mark is detected, the position of the axis or encoder is set depending on the configured mode:
   – "Mode" parameter at "MC_Home" = 2, 8
     Position = value in parameter "Position"
   – "Mode" parameter at "MC_Home" = 10
     Position = value in tag "<TO>.Homing.HomePosition"

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1766).

Passive homing with digital input (S7-1500)

The following figure shows an example of the homing motion with the following settings:

- Passive homing with digital input
- Homing in the positive direction
- Homing mark on the positive side of the digital input

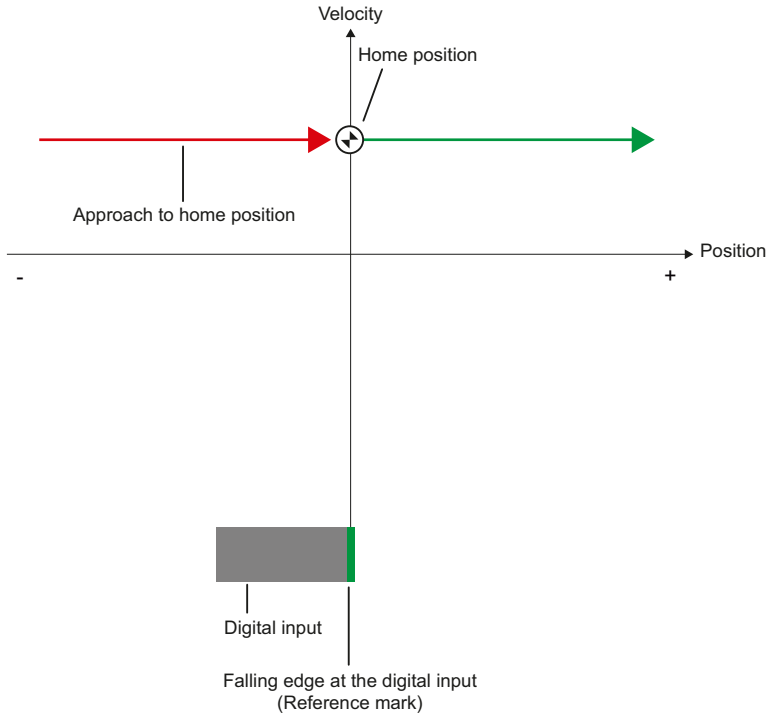

Motion sequence

The motion occurs in the following sequence:

1. Activation of passive homing using the Motion Control instruction "MC_Home"
2. Motion due to a Motion Control job from the user
   The detection of the homing mark at the digital input is enabled when the actual position value of the axis or encoder moves in the assigned homing direction.
3. Detection of the homing mark
   In the example, the falling edge of the switch at the digital input represents the homing mark. When the homing mark is detected, the position of the axis or encoder is set depending on the configured mode:
   – "Mode" parameter at "MC_Home" = 2, 8
     Position = value in parameter "Position"
   – "Mode" parameter at "MC_Home" = 10
     Position = value in tag "<TO>.Homing.HomePosition"

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1766).

Direction reversal at the hardware limit switch (reversing cam) (S7-1500)

During active homing, the hardware limit switch can optionally be used as a reversing cam. If the homing mark is not detected or the motion was not in the homing direction, the motion continues in the opposite direction with the approach velocity after the reversing cam.

When the hardware limit switch is reached, the default settings for dynamics take effect. Deceleration with the emergency deceleration does not hereby occur.

NOTICE

Avoid moving to a mechanical endstop

Ensure by one of the following measures, that in a direction reversal the machine does not move to a mechanical endstop.

- Keep the approach velocity low.
- Increase the configured acceleration/deceleration.
- Increase the distance between the hardware limit switch and the mechanical endstop.

Direct homing (S7-1500)

Depending on the configured mode, the position of the positioning axis/synchronous axis or external encoder technology objects can be absolutely or relatively set with "MC_Home".

Set actual position absolutely

Proceed as follows to set the position absolutely:

1. In the "MC_Home" Motion Control instruction, enter the absolute position in the "Position" parameter.
2. Call the "MC_Home" Motion Control instruction with the "Mode" = 0 parameter.

The position is set to the value specified in the "Position" parameter.

Set actual position relatively

Proceed as follows to set the position relatively:

1. In the "MC_Home" Motion Control instruction, enter the relative position in the "Position" parameter.
2. Call the "MC_Home" Motion Control instruction with the "Mode" = 1 parameter.

The position is set to the current position plus the value specified in the "Position" parameter.

Absolute value adjustment (S7-1500)

In absolute value adjustment, Motion Control determines an absolute value offset, that is retentively stored on the CPU.

Depending on the configured mode, the position of the axis or the encoder is absolutely or relatively set in the "MC_Home" Motion Control instruction.

- Parameter "Mode" = 7 (absolute specification of position)
  Position = value in parameter "Position"
- Parameter "Mode" = 6 (relative specification of position)
  Position = current position + value in parameter "Position"

Resetting the "Homed" status (S7-1500)

Incremental encoder

In the following cases, the "Homed" status is reset, and the technology object must be rehomed.

- Error in sensor system/encoder failure
- Initiation of active homing with the Motion Control instruction "MC_Home" with "Mode" = 3, 5 (after successful completion of the homing process, the status "Homed" is set again.)

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1766).

- Initiation of passive homing with the Motion Control instruction "MC_Home" with "Mode" = 2, 8, 10 (after successful completion of the homing process, the status "Homed" is set again.)
- Replacement of the CPU
- Replacement of the SIMATIC Memory Card
- POWER OFF
- Memory reset
- Modification of the encoder configuration
- Restart of the technology object
- Restoration of the CPU factory settings
- Transfer of a different project into the controller

Absolute encoder

In the following cases, the "Homed" status is reset, and the technology object must be rehomed.

- Replacement of the CPU
- Modification of the encoder configuration
- Restoration of the CPU factory settings
- Transfer of a different project into the controller

When you use a new absolute value encoder you need to home the absolute encoder once again.

Resetting the memory of the CPU or upgrading a project does not require another absolute value adjustment.

Tags (S7-1500)

The following technology object tags are relevant for homing:

Status indicators	
Tag	**Description**
<TO>StatusWord.X11 (HomingCommand)	Homing job running
<TO>StatusWord.X5 (HomingDone)	Technology object homed
<TO>ErrorWord.X10 (HomingFault)	Error occurred during homing

Note

Evaluation of the bits in StatusWord, ErrorWord and WarningWord

Read the information provided in section Evaluate StatusWord, ErrorWord and WarningWord (Page 1923).

Approach to reference cam	
Tag	Description
<TO>.Homing.ApproachDirection	Start direction or approach direction for the approach to the reference cam
<TO>.Homing.ApproachVelocity	Velocity for the approach to the reference cam

Approach to the homing mark	
Tag	Description
<TO>.Sensor[n].ActiveHoming.Direction	Homing direction
<TO>.Homing.ReferencingVelocity	Velocity for the approach to the homing mark

Approach to home position	
Tag	Description
<TO>.Homing.ApproachVelocity	Velocity for the move to homing point

Positions	
Tag	Description
<TO>.Homing.AutoReversal	Reversal at the hardware limit switches
<TO>.Homing.HomePosition	Home position
<TO>.StatusSensor[n].AbsEncoderOffset	Calculated offset after the absolute encoder adjustment

Parameters for active homing	
Tag	Description
<TO>.Sensor[n].ActiveHoming.Mode	Homing mode
<TO>.Sensor[n].ActiveHoming.SideInput	Side of the digital input
<TO>.Sensor[n].ActiveHoming.Direction	Homing direction or approach direction
<TO>.Sensor[n].ActiveHoming.DigitalInputAddress	Address of digital input
<TO>.Sensor[n].ActiveHoming.HomePositionOffset	Offset of the homing mark from the home position

Parameters for passive homing	
Tag	Description
<TO>.Sensor[n].PassiveHoming.Mode	Homing mode
<TO>.Sensor[n].PassiveHoming.SideInput	Side of the digital input

Parameters for passive homing	
Tag	Description
<TO>.Sensor[n].PassiveHoming.Direction	Homing direction or approach direction
<TO>.Sensor[n].PassiveHoming.DigitalInputAddress	Address of digital input

Position monitoring functions (S7-1500)

The following functions are available in the positioning axis/synchronous axis technology object for monitoring positioning and motion:

- Positioning monitoring (Page 1691)
  The actual position value must reach a positioning window within a specified time, and remain in this positioning window for a minimum dwell time.
- Following error monitoring (Page 1692)
  The following error is monitored based on a velocity-dependent following error limit. The permissible maximum following error depends on the velocity setpoint.

If monitored conditions are violated, then technology alarms are output. The technology object responds in accordance with the alarm response.

Positioning monitoring (S7-1500)

Positioning monitoring monitors the behavior of the actual position at the end of the setpoint calculation.

As soon as the velocity setpoint reaches the value zero, the actual position value must be located in the positioning window within a tolerance time. The actual value must not exit the positioning window during the minimum dwell time.

If the actual position is reached at the end of a positioning motion within the tolerance time and remains in the positioning window for the minimum dwell time, then "<TO>.StatusWord.X6 (Done)" is set in the technology data block. After expiration of the minimum dwell time, the "Done" parameter of the corresponding Motion Control instruction is also set. This completes a motion job.

The following figure shows the chronological sequence and the positioning window:

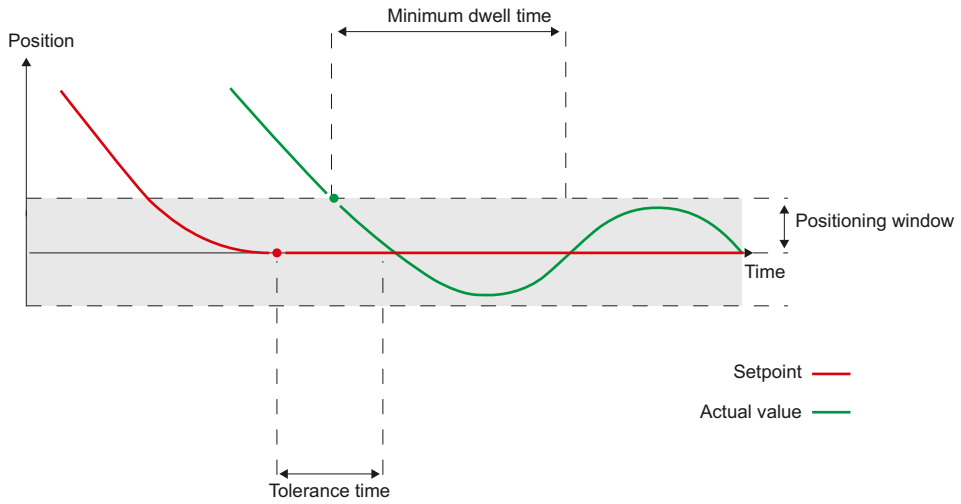

Positioning monitoring does not make any distinction between how the setpoint interpolation was completed. The end of setpoint interpolation can for example be reached as follows:

- By the setpoint reaching the target position
- By position-controlled stopping during motion by the Motion Control instruction "MC_Halt"

Violation of positioning monitoring

In the following cases, technology alarm 541 is output by the positioning monitoring, and the technology object is disabled (alarm reaction: remove enable).

- The actual value does not reach the positioning window during the tolerance time.
- The actual value exits the positioning window during the minimum dwell time.

Following error monitoring (S7-1500)

The following error in the positioning axis/synchronous axis technology object is monitored based on a velocity-dependent following error limit. The permissible following error depends on the velocity setpoint.

A constant permissible following error can be specified for velocities lower than an adjustable velocity low limit.

Above this low velocity limit, the permissible following error increases in proportion to the velocity setpoint. The configurable maximum permissible following error is the maximum velocity limit.

Calculation of the following error

The following error is the difference between the position setpoint and the actual position value. The transmission times of the setpoint to the drive, and of the actual position value to the controller, are taken into account in the calculation of the following error, i.e. subtracted out.

Warning limit

A warning limit can be specified for the following error. The warning limit is input as a percentage value and operates relative to the current permissible following error. If the warning limit of the following error is reached, then technology alarm 522 is output. This is a warning and contains no alarm response.

Violation of the permissible following error

If the permissible following error is exceeded, then technology alarm 521 is output, and the technology object is disabled (alarm response: remove enable).

When force/torque limiting is activated, the monitoring of the permissible following error can be deactivated.

Standstill signal (S7-1500)

When the actual velocity reaches the standstill window and remains there for the minimum dwell time, the standstill of the axis is indicated.

Tags (S7-1500)

The following technology object tags are relevant for position monitoring and for the standstill:

Status indicators	
Tag	**Description**
<TO>.StatusWord.X7 (Standstill)	Set to the value "TRUE" when the actual velocity reaches the standstill window and does not exit it within the minimum dwell time. The standstill signal is only present for the positioning axis/ synchronous axis.
<TO>.StatusWord.X6 (Done)	**Positioning axis/synchronous axis** Set to the value "TRUE" when the actual velocity value reaches the positioning window within the tolerance time and remains in the window for the minimum dwell time.
	Speed axis Set to "TRUE" when the motion is complete and the speed setpoint is therefore equal to zero.
<TO>.ErrorWord.X12 (Positioning-Fault)	A positioning error has occurred.

Positions and times	
Tag	**Description**
<TO>.PositioningMonitoring.Tolerance-Time	Maximum permissible time until positioning window is reached The time is started with the end of the setpoint interpolation.
<TO>.PositioningMonitoring.MinDwell-Time	Minimum dwell time in position window
<TO>.PositioningMonitoring.Window	Positioning window

Standstill signal	
Tag	Description
<TO>.StandstillSignal.VelocityThreshold	Velocity threshold for the standstill signal
<TO>.StandstillSignal.MinDwellTime	Minimum dwell time below the velocity threshold

The following technology object tags are relevant for following error monitoring:

Status indicators	
Tag	Description
<TO>.StatusPositioning.FollowingError	Current following error
<TO>.ErrorWord.X11 (FollowingErrorFault)	Status indication, that the following error is too large
<TO>.WarningWord.X11 (FollowingErrorWarning)	Status indication, that the following error warning limit has been reached

Control bits	
Tag	Description
<TO>.FollowingError.EnableMonitoring	Enable/disable following error monitoring

Limits	
Tag	Description
<TO>.FollowingError.MinVelocity	Lower velocity setpoint for the characteristic curve of the maximum following error
<TO>.FollowingError.MinValue	Permissible following error below the "<TO>.FollowingError.MinVelocity"
<TO>.FollowingError.MaxValue	Maximum permissible following error at maximum axis velocity
<TO>.FollowingError.WarningLevel	Warning limit as a percentage value relative to the maximum permissible following error (velocity-dependent in accordance with the characteristic curve)

Closed-loop control (S7-1500)

The position controller of the positioning axis / synchronizing axis is a proportional controller with velocity precontrol.

Velocity precontrol

The velocity precontrol can be used to minimize the velocity-based following error during position control. As a result, faster positioning is achieved, if needed, because the position controller only has to compensate disturbances.

When using the velocity precontrol, the velocity setpoint is additionally added to the output of the position controller. You can weight this additional setpoint by a factor.

Dynamic Servo Control (DSC)

In drives that support Dynamic Servo Control (DSC), you can optionally use the closed-loop position controller in the drive. If you use telegrams that support DSC, DSC is automatically activated. The position controller in the drive is usually implemented with a rapid speed-control cycle. This improves the control performance for digitally coupled drives.

The following requirements must be met to use DSC:

- The motor encoder (first encoder in the telegram) of the drive is used as the first encoder for the technology object.
- One of the following PROFIdrive telegrams is used for the drive:
  - Standard telegram 5 or 6
  - SIEMENS telegram 105 or 106

See also

PROFIdrive telegrams (Page 1636)

Control structure (S7-1500)

The following figure shows the effective closed loop control structure **without** DSC:

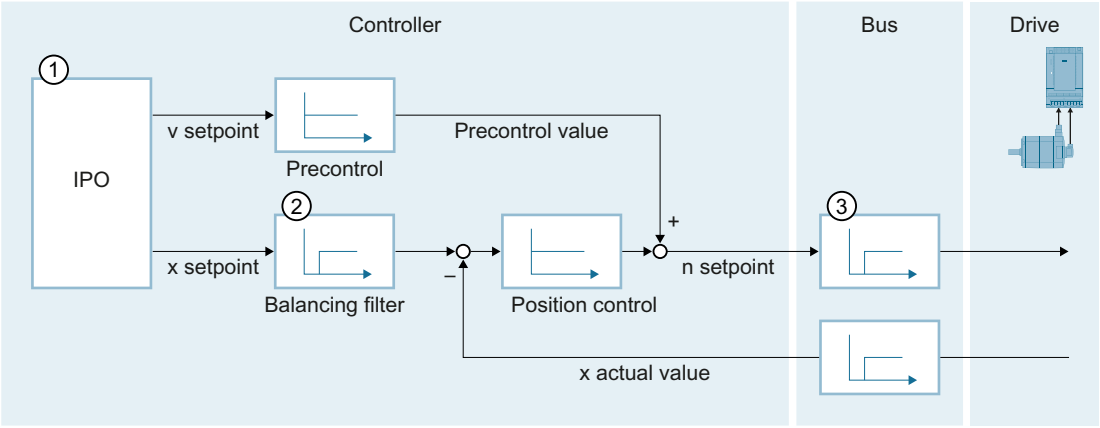

① Interpolator with motion control

② Internal consideration of the signal propagation times and the speed-control loop substitute time

③ Communication between controller and drive

The following figure shows the effective closed loop control structure **with** DSC:

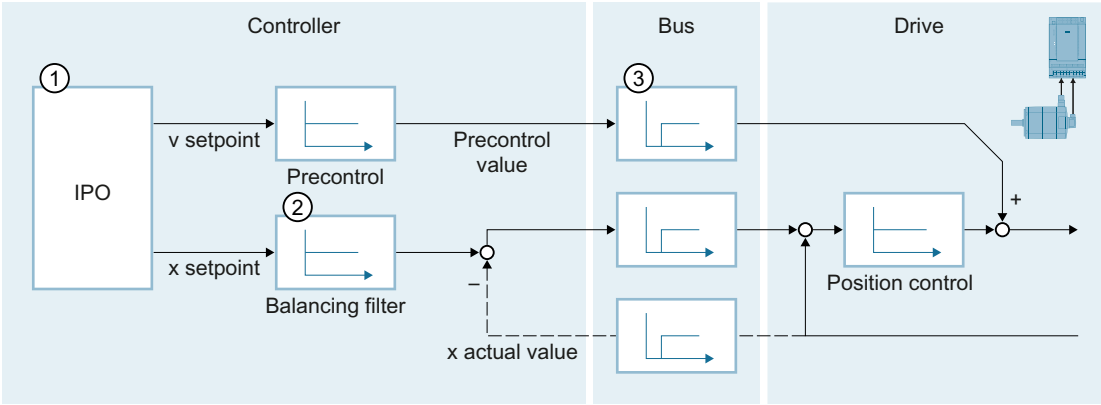

① Interpolator with motion control
② Internal consideration of speed control loop substitute time
③ Communication between controller and drive

See also

Configuration - Control loop (Page 1831)

Tags (S7-1500)

The following technology object tags are relevant for the closed-loop control:

Parameters	
Tag	**Description**
<TO>.PositionControl.Kv	Proportional gain of the closed loop position control
<TO>.PositionControl.Kpc	Velocity precontrol of the positioning system (in %)
<TO>.PositionControl.EnableDSC	Enable DSC
<TO>.DynamicAxisModel.VelocityTimeConstant	Speed control loop substitute time [s]

Parameters		
Tag	Description	
<TO>.PositionControl.ControlDifferenceQuantization.Mode	Type of quantification Configuration of a quantization when a drive with stepper motor interface is connected	
	0	No quantification
	1	Quantization corresponding to encoder resolution
	2	Quantization to a direct value (value input in "<TO>.PositionControl.ControlDifferenceQuantization.Value")
	Configuration is performed using the parameter view (data structure).	
<TO>.PositionControl.ControlDifferenceQuantization.Value	Value of quantification Configuration of a value for quantization to a direct value ("<TO>.PositionControl.ControlDifferenceQuantization.Mode" = 2) The quantization value is specified in the position unit of the axis. Configuration is performed using the parameter view (data structure).	

Process response (S7-1500)

Organization Blocks for Motion Control (S7-1500)

When you create a technology object, organization blocks are automatically created for processing the technology objects. The Motion Control functionality of the technology objects creates its own execution level, and is called according to the Motion Control application cycle.

The following blocks are created:

- MC-Servo [OB91]
  Calculation of the position controller
- MC-Interpolator [OB92]
  Evaluation of the Motion Control instructions, generation of setpoints and monitoring functionality

The organization blocks are protected (know-how protection). The program code cannot be viewed or changed.

The frequency relationship of the two organization blocks to one another is always 1:1. MC-Servo [OB91] is always executed before MC-Interpolator [OB92].

You can set the application cycle and the priority of the organization blocks according to your requirements for control quality and system load.

MC-PreServo [OB67] and MC-PostServo [OB95]

In addition, you can create the organization blocks MC-PreServo [OB67] and MC-PostServo [OB95].

The organization blocks MC-PreServo [OB67] and MC-PostServo [OB95] can be programmed. They are called in the configured application cycle. This means you can use the organization

blocks MC-PreServo [OB67] and MC-PostServo [OB95] for consistent data processing or time-critical events, such as starting of motions or homing functions for the process that may have to run synchronously with the application cycle.

Application cycle

You can set the application cycle in which the MC-Servo [OB91] is called in the properties of the organization block in "General > Cycle Time":

- **Synchronous to the bus**
  MC-Servo [OB91] is called synchronously with or at a reduced ratio to a bus system. You set the send clock in the properties of the selected bus system. In the "Distributed I/O" drop-down list, you can select the following bus systems:
  - Isochronous PROFIBUS DP
  - Isochronous PROFINET IO

  You cannot call the MC-Servo [OB91] synchronously with a bus system that is connected to the CPU via a communications processor/communications module (CP/CM).
- **Cyclic**
  The MC-Servo [OB91] is called cyclically with the specified application cycle.

The selected application cycle must be long enough to be able to process the technology objects in one cycle. If the processing time of the technology objects is longer than the application cycle, overflows (Page 1699) will occur. In order to achieve optimal control quality, the calculations of the organization blocks MC-PreServo [OB67], MC-Servo [OB91] and MC-PostServo [OB95] must be performed within a send cycle.

You can check the runtime of MC-Servo [OB91] and MC-Interpolator [OB92] with the expanded instruction "RT_INFO".

The current application cycle (information in µs) of the organization blocks MC-PreServo [OB67] and MC-PostServo [OB95] can be read using the start information.

Reduction ratio (CPU V1.5 and higher)

You can reduce the application cycle of the MC-Servo [OB91] relative to the send clock of an isochronous PROFINET IO system. You can set an integer multiple of the send clock as the factor. Values up to 14 times the send clock (maximum 32 ms) are possible for the application cycle.

If you call an isochronous mode interrupt OB and the MC-Servo [OB91] synchronously with the same PROFINET IO system, you must set the same application cycle for both organization blocks.

Priority

You can configure the priority of the organization blocks as needed in their properties under "General > Properties > Priority":

- **MC-Servo [OB91]**
  Priority 17 to 26 (default value 26)
- **MC-Interpolator [OB92]**
  Priority 16 to 25 (default value 24)

The priority of MC-Servo [OB91] must be at least one higher than the priority of MC-Interpolator [OB92].

The priority of the organization blocks MC-PreServo [OB67] and MC-PostServo [OB95] correspond to the priority of the MC-Servo [OB91]. MC-PreServo [OB67] is called immediately before MC-Servo [OB91]. MC-PostServo [OB95] is called immediately after MC-Servo [OB91].

See also

Data connection drive/encoder via data block (Page 1650)

Process image partition "OB Servo PIP" (S7-1500)

The process image partition "OB Servo PIP" is made available in isochronous mode for Motion Control when MC-Servo [OB91] is called. All drives and encoders used by Motion Control are assigned to this process image partition.

Because the organization blocks MC-PreServo [OB67] and MC-PostServo [OB95] are called automatically by the MC-Servo [OB91], the process image partition is also available automatically. If you use a MC-PreServo [OB67], the data are read in when the MC-PreServo [OB67] starts. If you use a MC-PostServo [OB95], the data are output after the MC-PostServo [OB95].

Additionally, you should assign all I/O modules used by Motion Control to this process image partition (e.g. hardware limit switches). The assignment results in chronologically synchronous processing with the technology object.

The input process image partition is also updated in STOP mode.

Process image partition in the user program

As of CPU version V1.5, you can access the process image partition "OB Servo PIP" in your user program. This makes it possible to evaluate the process image partition using the trace function.

Operational Sequence and Timeouts (S7-1500)

When processing the Motion Control functionality, the organization blocks MC-Servo [OB91] and MC-Interpolator [OB92] are called and processed in each application cycle. The remaining cycle time is available for the processing of your user program.

For error-free program execution, keep to the following rules:

- In each application cycle, MC-Servo [OB91] must be started and executed completely.
- In each application cycle, the relevant MC-Interpolator [OB92] must at least be started.

The following figure shows an example of the error-free operational sequence for the processing of organization block OB1:

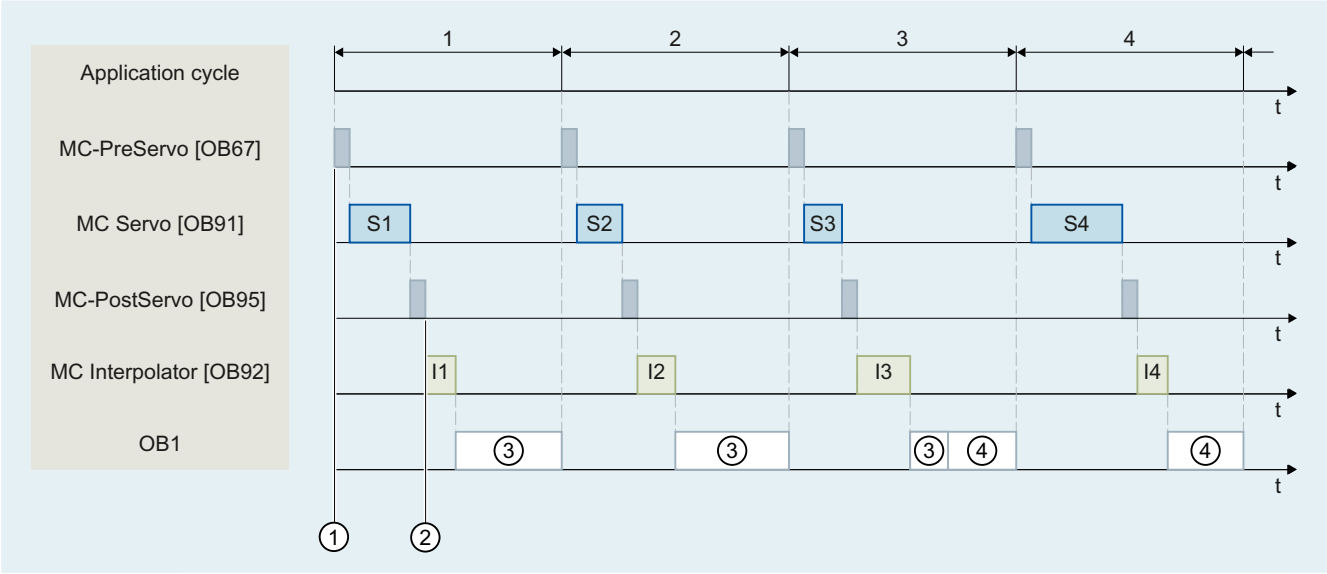

① "TPA OB Servo" input
② "TPA OB Servo" output
③ First OB1 cycle
④ Second OB1 cycle

Overflows

Overflows can occur if the configured application cycle is not adhered to, for example because additional technology objects or programs are added in the MC-PreServo [OB67] or MC-PostServo [OB95]. The application cycle must be adapted in this case to prevent overflows.

A message is generated in the diagnostic buffer of the CPU in the case of an overflow of MC-Servo [OB91] in the send clock. The controller no longer runs isochronously.

The CPU switches to STOP mode in the case of an overflow of MC-Servo [OB91] in the application cycle.

The following figure shows the behavior in the case of overflow of MC-Servo [OB91] in the application cycle and in the send clock with a reduction ratio of 2:

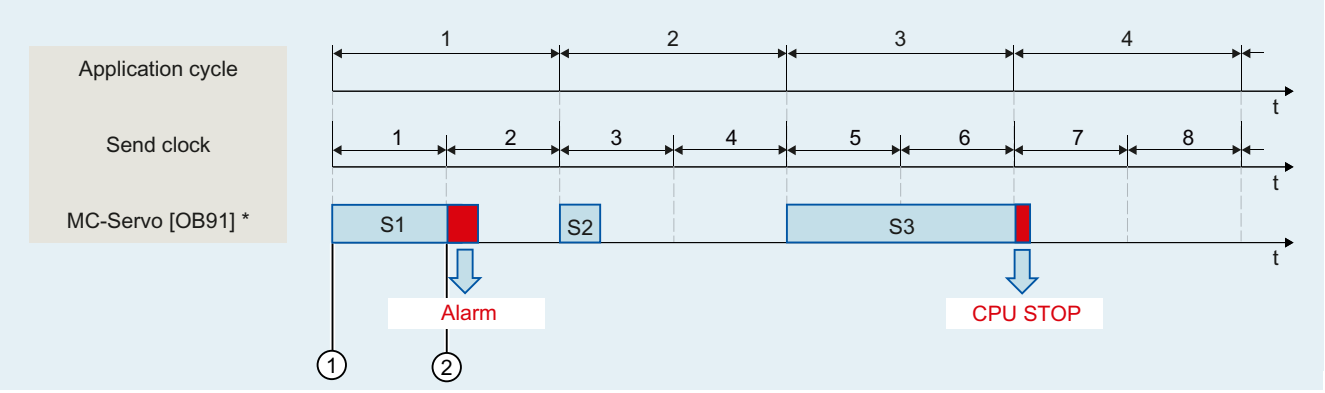

① "TPA OB Servo" input

② "TPA OB Servo" output

* Including MC-PreServo [OB67] and/or MC-PostServo [OB95], when in use

The execution of an MC-Interpolator [OB92] may only be interrupted by an MC-Servo [OB91] call. If more interruptions occur, the CPU switches to STOP mode.

The following figure shows the operational sequence when an MC-Interpolator [OB92] is interrupted over two time slices:

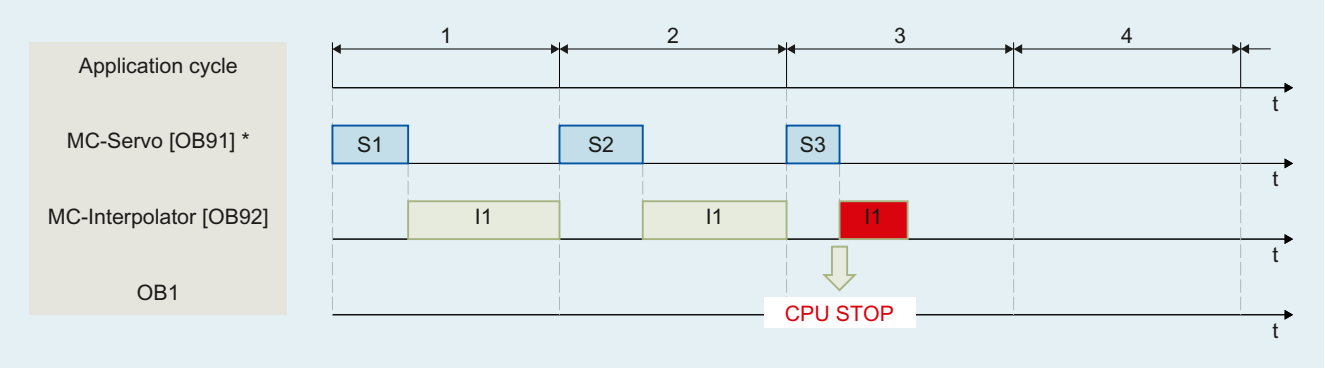

* Including MC-PreServo [OB67] and/or MC-PostServo [OB95], when in use

The CPU tolerates a maximum of three consecutive overflows of MC-Interpolator [OB92]. If more overflows occur, the CPU switches to STOP mode.

The following figure shows the operational sequence if there are four consecutive individual overflows of MC-Interpolator [OB92]:

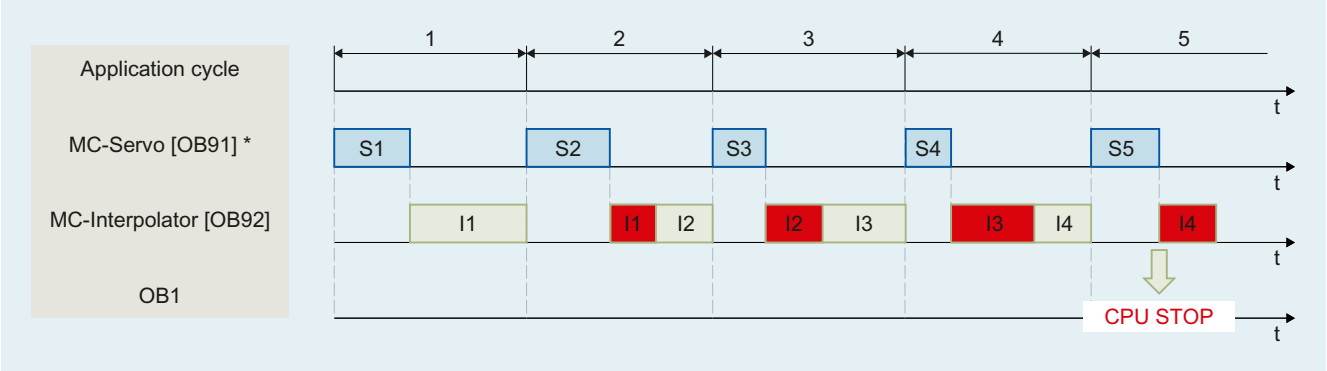

* Including MC-PreServo [OB67] and/or MC-PostServo [OB95], when in use

Operating modes (S7-1500)

This section examines the behavior of the Motion Control in the respective operating modes and in the transitions between operating modes. A general description of the operating modes can be found in the S7-1500 System Manual

Operating modes and transitions

The CPU has the following operating modes:

- STOP
- STARTUP
- RUN
- HOLD

The following figure shows the operating modes and the operating mode transitions:

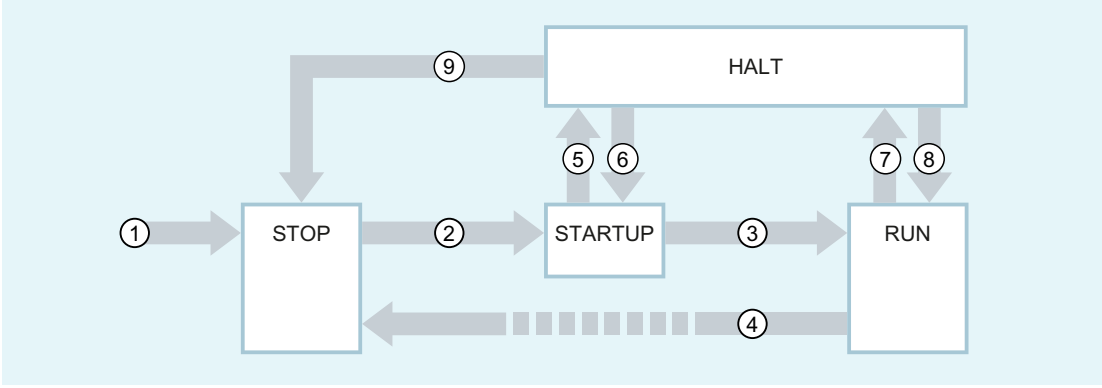

Operating mode transitions

The following table shows the behavior of the Motion Control in the transitions between the operating modes:

No.	Operating mode transition	Behavior
①	POWER ON → STOP	The CPU performs a restart of the technology objects. The technology objects are reinitialized with the values from the load memory.
②	STOP → STARTUP	Not relevant for Motion Control.
③	STARTUP → RUN	The process outputs are enabled.
④	RUN → STOP	When the CPU changes from RUN mode to STOP mode, all technology objects are disabled in accordance with the alarm response "remove enable". Active Motion Control jobs are aborted. If restart-relevant data has been changed for technology objects in RUN, the CPU performs a restart of the corresponding technology objects.
⑤	STARTUP → HOLD	Breakpoint in the start-up routine reached.
⑥	HOLD → STARTUP	Not possible when using technology objects
⑦	RUN → HOLD	Breakpoint reached
⑧	HOLD → RUN	Not possible when using technology objects
⑨	HOLD → STOP	By operation of switch/display or by setting to STOP from programming device.

STOP mode

In STOP mode the user program is not processed and all process outputs are disabled. Thus no Motion Control jobs are executed.

The technology data blocks are updated.

STARTUP mode

Before the CPU starts processing of the cyclical user program, the startup OBs are processed one time.

In STARTUP mode, the process outputs are disabled. Motion Control jobs are rejected.

The technology data blocks are updated.

RUN mode

The user program is processed in RUN mode.

In RUN mode, the programmed Motion Control jobs are cyclically called and processed.

The technology data blocks are updated.

HOLD operating state

Working with breakpoints is not supported when technology objects are used. An overflow of the MC-Servo hereby occurs. This leads to an immediate switch to STOP mode.

In HOLD operating state, events are not initiated and the user program is not executed.

All outputs are disabled or react according to the parameter settings. Outputs supply a configured substitute value or keep the last value output and bring the controlled process to a safe operating state.

When you reach a breakpoint, the CPU executes an implicit restart of the technology object. Homing the technology once again.

Speed-controlled axis technology object (S7-1500)

The speed axis technology object calculates speed setpoints, taking into account the dynamic settings, and outputs them to the drive. All motions of the speed axis take place as speed-controlled motions. An existing load gear is taken into account on the system side.

You can find an overview of the functions of the speed axis technology object in the Functions (Page 1626) section.

A drive is assigned to each speed axis by means of a PROFIdrive telegram or an analog setpoint interface.

The speed is specified in revolutions per unit of time.

The following figure shows the basic principle of operation of the speed axis technology object:

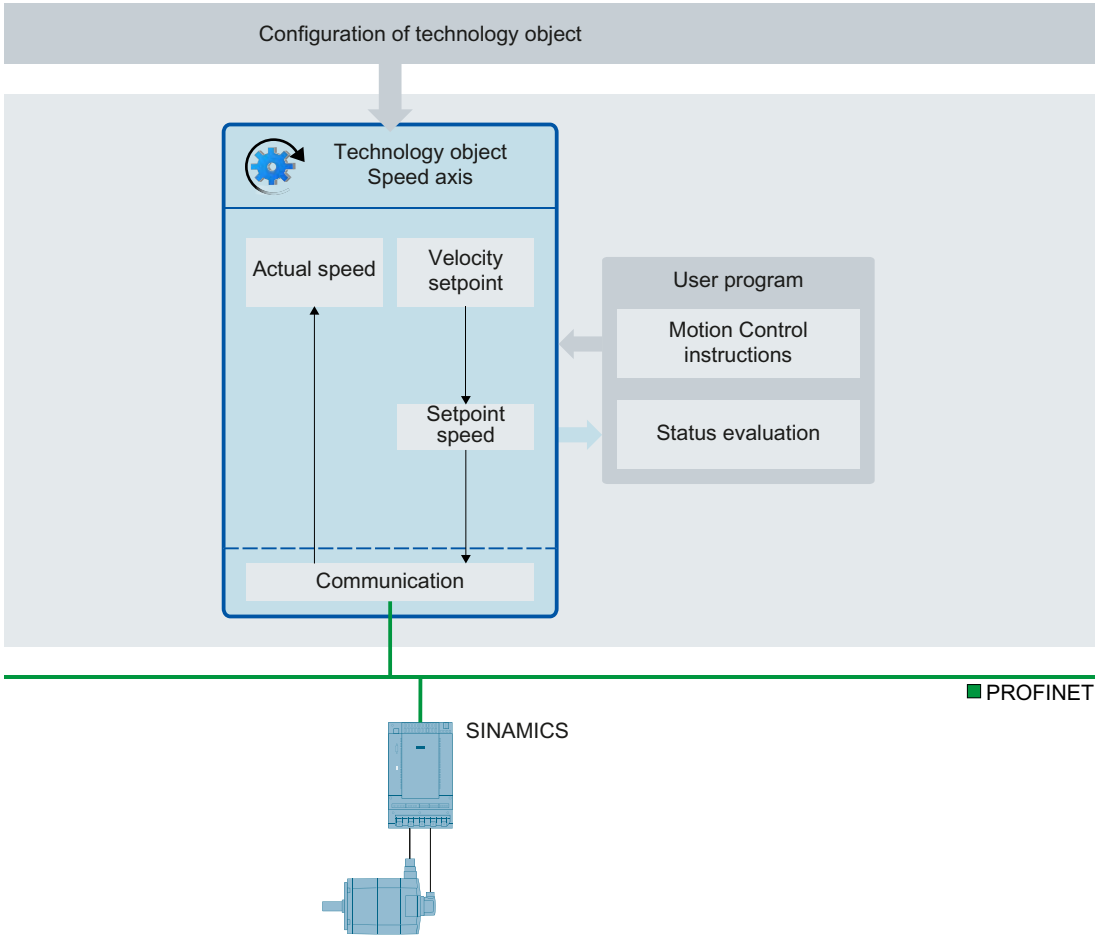

Positioning axis technology object (S7-1500)

The positioning axis technology object calculates position setpoints, taking into account the encoderc settings, and outputs corresponding speed setpoints to the drive. In position-controlled mode, all movements of the positioning axis take place as position-controlled movements. For absolute positioning, the physical position must be known to the positioning axis technology object.

You can find an overview of the functions of the positioning axis technology object in the Functions (Page 1626) section.

Each positioning axis is assigned a drive by means of a PROFIdrive telegram or an analog setpoint interface as well as an encoder by means of a PROFIdrive telegram.

The relationship between the encoder values and a defined position is established by the parameter assignment of the mechanical properties and encoder settings and by a homing operation. The technology object can also perform movements without a position relationship, and relative position movements, even without being in a homed status.

A positioning axis can be configured as a linear axis or rotary axis (Page 1631), depending on the design of the mechanics.

The figure below shows the basic principle of operation of the positioning axis technology object:

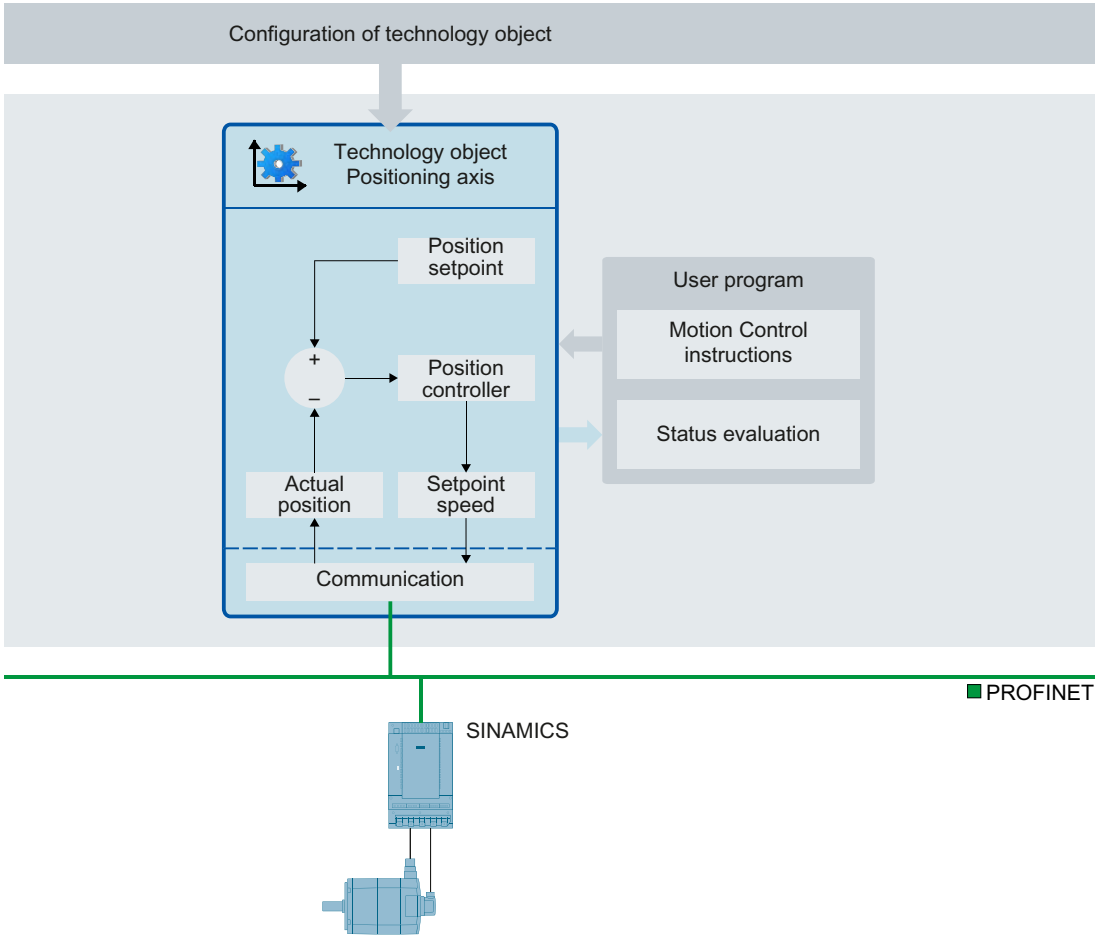

Synchronous axis technology object (S7-1500)

The synchronous axis technology object includes all functions of the positioning axis technology object.

A synchronous axis can also follow the motions of a leading axis. The synchronous operation relationship between the leading and following axes is specified by a synchronous operation function.

You can find an overview of the functions of the synchronous axis technology object in the Functions (Page 1626) section.

The figure below shows the basic principle of operation of the synchronous axis technology object:

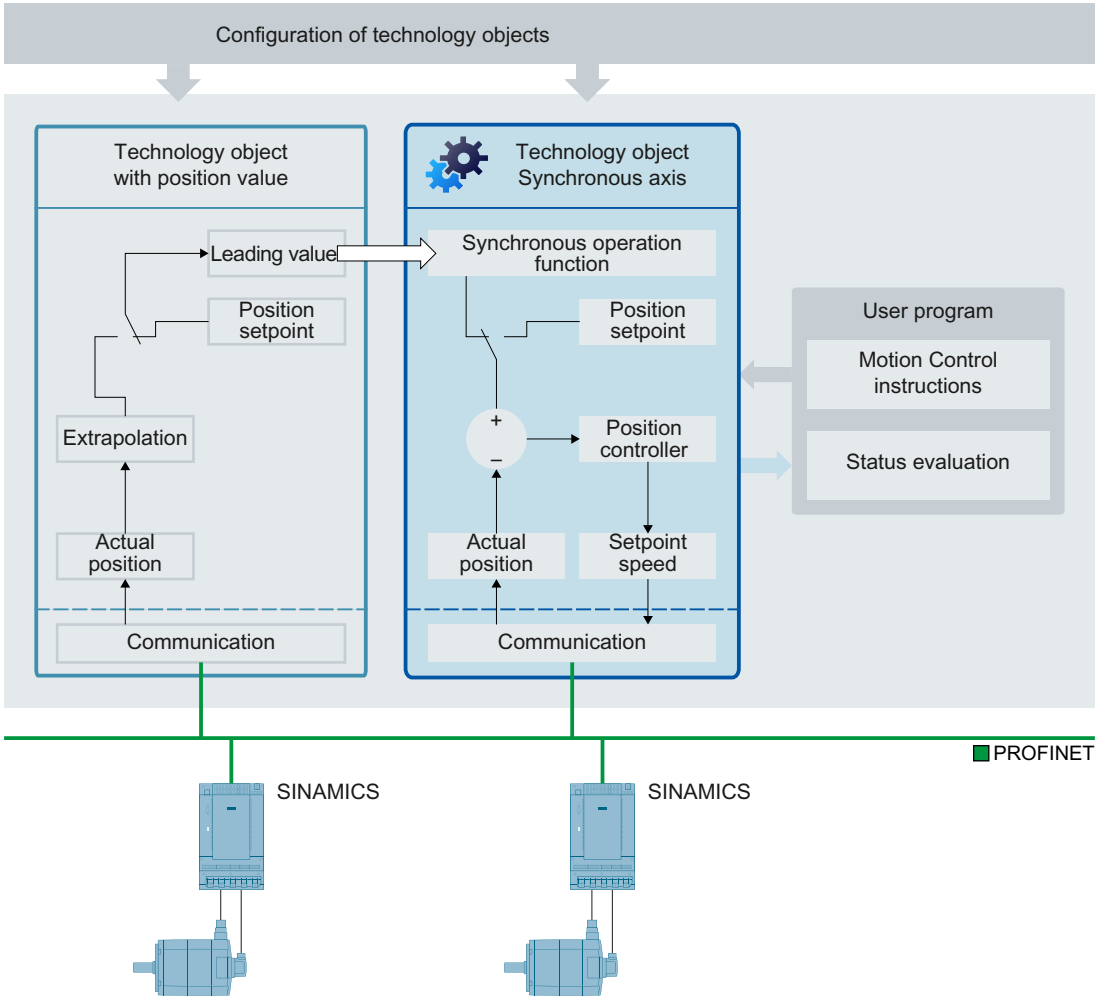

See also

Axis types (Page 1631)

Synchronous operation phases (S7-1500)

By means of synchronous operation, a following axis can be linked to a leading axis and move synchronously with it.

The synchronous operation proceeds in the following phases:

- Pending synchronous operation (S7-1500T)
  The following axis waits for the start conditions of the synchronizing motion.
- Synchronization
  The following axis is synchronized to the leading value.
- Synchronous motion
  The following axis follows the position of the leading axis according to the synchronous operation function.
- Synchronous operation override
  Active synchronous operation is overridden by motion jobs (e.g. "MC_Halt") to the following axis.

During these phases, the dynamic limits apply as described in section Dynamic limits in synchronous operation (Page 1662).

Avoid homing the leading axis during an active synchronous operation. Homing the leading axis during synchronous operation corresponds to a setpoint step change on the following axis. The following axis compensates for the step change according to the synchronous operation function and limited only to the maximum speed of the drive.

Note

The leading values and following values are coupled without conversion into the relevant configured user unit. If, for example, a linear leading axis moves by 10 mm, a rotary following axis moves by 10° with at a gear ratio of 1:1.

Leading value coupling (S7-1500)

The leading value for synchronous operation is provided by a leading axis. The leading value is specified and coupled in the user program with the call of the corresponding Motion Control instruction for synchronous operation. The leading value is switched when you call the Motion Control instruction again specifying a different leading axis.

The following rules apply to the leading value coupling:

- A leading axis can output the leading value to several following axes.
- The synchronous axis can be interconnected with different leading values. All interconnections required during operation must be set up during configuration of the technology object.
- Only one leading value at a time is coupled and evaluated.

Setpoint coupling (S7-1500)

With setpoint coupling, the position setpoint of the leading axis is used as the leading value for synchronous operation.

The position setpoint of the following technology objects can be interconnected as the leading value for synchronous operation:

- Positioning axis
- Synchronous axis

Actual value coupling

For applications in which setpoint coupling is not possible, e.g. when using an external encoder, the S7-1500T CPU additionally offers actual value coupling for synchronous operation. With actual value coupling, the extrapolated actual position (Page 1708) of a technology object is used as the leading value.

The actual position of the following technology objects can used as the leading value:

- Positioning axis
- Synchronous axis
- External encoder

Extrapolation of the leading values for actual value coupling

With actual value coupling (Page 1707), delay times result from the processing of the actual values. To compensate for these delay times, the actual value is extrapolated on the leading value side. This means that the leading value is extrapolated based on previously known values.

Actual value discrepancies at constant velocity or at constant acceleration/deceleration can be compensated for with the extrapolation. For technical reasons, changes of acceleration/deceleration (jerk) during extrapolation always cause a displacement of the following axis relative to the leading value.

The extrapolation time consists of two parts: a leading axis dependent portion and a following-axis dependent portion resulting from a user setting:

- **Leading axis-caused part**
  The part caused by the leading axis is calculated automatically and displayed at the leading axis in the "<TO>.Extrapolation.LeadingAxisDependentTime" tag of the technology object.
- **Following axis-caused part**
  The part caused by the leading axis is calculated automatically and displayed at the following axis in the "<TO>.StatusPositioning.SetpointExecutionTime" tag of the technology object. You specify the value of this tag (unchanged or compensated with user-specific runtimes) in the leading axis configuration (<TO>.Extrapolation.FollowingAxisDependentTime).

The extrapolated actual value is evaluated with a configurable hysteresis before it is output as the leading value. The hysteresis evaluation prevents an inversion of the leading value, which may result from extrapolation of a noisy value.

The following figure shows the sequence of the actual value extrapolation.

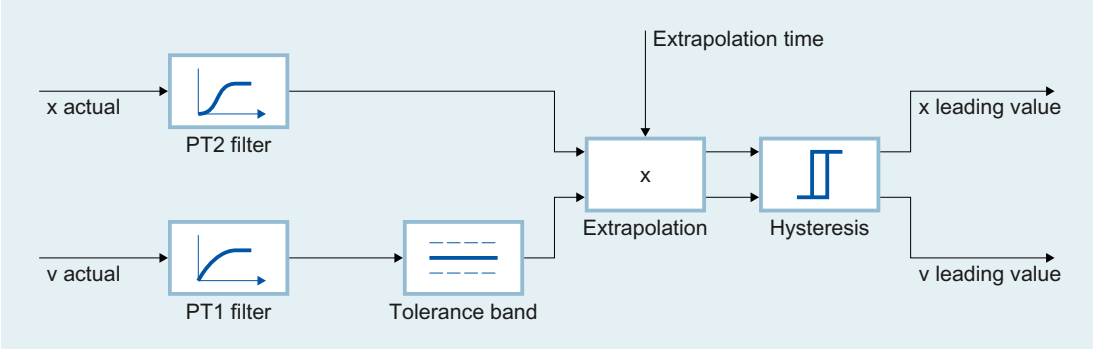

NOTICE
Machine damage If you change the extrapolation time during runtime of your user program in too large of increments, damage to the machine may occur. Change the extrapolation time only by a small amount.

Filtering the actual values

Noisy encoder signals lead to high velocity step changes, which also affect the extrapolation. These step changes can be reduced or compensated for by using suitable filter settings. The actual position filter is a PT2 filter. The velocity filter is a PT1 filter with configurable tolerance bandwidth.

The actual value is first smoothed by the actual position filter. The velocity is smoothed by the velocity filter and further "calmed" by the tolerance band. The filtered actual position is then extrapolated taking into account the filtered velocity.

Recommended settings.

Set the total of the time constants T1 and T2 the position filter much smaller than the time constant T1 of the velocity filter.

Tolerance band

The tolerance band acts on the filtered velocity value in the interpolation cycle. The position of the tolerance band is automatically shifted to the direction of the velocity value as soon as it changes in one direction by more than half of the tolerance band from the last output value. A new output value is simultaneously formed with the shift of the tolerance band. This corresponds to the filtered velocity value minus half the tolerance band. As long as the velocity value remains within the tolerance band, no new output value is formed.

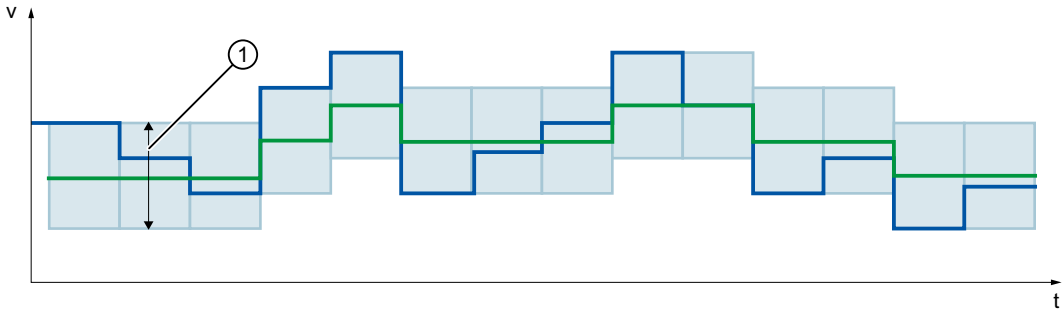

① Tolerance band

Extrapolation velocity before tolerance band

Extrapolation velocity after tolerance band

Hysteresis

The hysteresis acts on the filtered extrapolated position value in the interpolation cycle. A change of direction only takes effect when the position value changes in the direction opposite at least by the hysteresis value.

Tags

The following technology object tags are relevant for the actual value extrapolation:

Configuration	
Tag	**Description**
<TO>.Extrapolation.LeadingAxisDependent-Time	(for the leading axis) Leading axis-caused part of the extrapolation time, which results from T_i, T_{ipo}, and T_{Filter}.
<TO>.Extrapolation.FollowingAxisDependent-Time	(for the leading axis) Following-axis dependent portion of the extrapolation time Enter the value from the "<TO>.StatusPositioning.SetpointExecution-Time" tag of the following axis (unchanged or compensated with user-specific times).
<TO>.StatusPositioning.SetpointExecutionTime	(for the following axis) Setpoint execution time that results from T_{Ipo}, T_{vtc} or 1/kv, T_{Send} and T_O of the axis.
<TO>.Extrapolation.PositionFilter.T1	Position filter time constant T1
<TO>.Extrapolation.PositionFilter.T2	Position filter time constant T2
<TO>.Extrapolation.VelocityFilter.T1	Velocity filter time constant T1

Configuration	
Tag	Description
<TO>.Extrapolation.VelocityTolerance.Range	Tolerance band width for velocity
<TO>.Extrapolation.Hysteresis.Value	Hysteresis value (in the configured unit of length)

Gearing with "MC_GearIn" (S7-1500)

During gearing the position of the following axis results from the position of the leading axis multiplied by the gear ratio. You specify the gear ratio as a ratio of two integers. The result is a linear synchronous operation function.

Synchronous operation with "MC_GearIn" begins after synchronization when the following axis has reached the velocity and acceleration of the leading axis, taking into account the gear ratio.

Synchronization

Synchronization begins with the start of an "MC_GearIn" job. When a synchronous axis is synchronizing to a leading value, this is indicated in the "<TO>.StatusWord.X21 (Synchronizing)" tag of the technology object. Active motion jobs are overridden. The dynamic values (acceleration, delay, jerk) of the following axis for the synchronization are specified at "MC_GearIn".

The synchronization duration and distance are dependent on the following parameters:

- Start time of the "MC_GearIn" job
- Dynamics of the following axis at the start time
- Dynamic value settings for "MC_GearIn"
- Dynamics of the leading axis

If the following axis has reached the velocity and the acceleration of the leading axis, taking into account the gear ratio, the following axis is synchronized. The following axis travels synchronously with the leading axis.

Synchronous motion

When a synchronous axis is synchronized to a leading value, the "Synchronous" status is indicated by parameter "MC_GearIn.InGear" = TRUE as well as in the "<TO>.StatusWord.X22 (Synchronous)" tag of the technology object. The following axis follows the dynamics of the leading axis according to the gear ratio.

The response characteristic during gearing is expressed by a linear relationship between the leading value and following value.

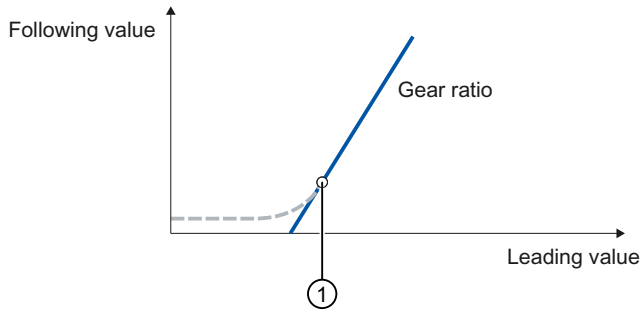

Gear ratio	Slope of line/transmission ratio Gear ratio = "MC_GearIn.RatioNumerator"/"MC_GearIn.RatioDenominator"
- - - - -	Synchronization
①	Position starting from which the leading and following axes move synchronously Position ① is dependent on the above-named parameters (see Synchronization).

The following value is calculated according to the following function:

Position of following axis (following value) = Position ① of following axis + gear ratio × (Position of leading axis - Position ① of leading axis)

Direction

The numerator of the gear ratio is specified as positive or negative. This yields the following behavior:

- Positive gear ratio:
  The leading and following axes move in the same direction.
- Negative gear ratio:
  The following axis moves in the opposite direction to the leading axis.

See also

Tags for synchronous operation (Page 1720)

Dynamic limits in synchronous operation (Page 1662)

Gearing with "MC_GearInPos" with specified synchronous positions

During gearing the position of the following axis results from the position of the leading axis multiplied by the gear ratio. You specify the gear ratio as a ratio of two integers. The result is a linear synchronous operation function.

Synchronous operation with "MC_GearInPos" starts after synchronization when the specified synchronous positions are reached.

Synchronization

You specify the synchronous positions of the leading and following axes starting from which they are to move synchronously in the Motion Control instruction "MC_GearInPos". The synchronous operation is synchronized in advance of the specified synchronous positions for the leading and following axis.

The synchronization in advance (Page 1718) is performed using the dynamic parameters (velocity, acceleration, deceleration, jerk) or the leading value distance. The dynamic parameters and the leading value distance are specified in "MC_GearInPos" and influence the start position for synchronization. After the start of a "MC_GearInPos" job, synchronization occurs in such a way that the leading and following axis move synchronously when the synchronous positions are reached.

The synchronization is indicated by parameter "MC_GearInPos.StartSync" = TRUE as well as in the "<TO>.StatusWord.X21 (Synchronizing)" tag of the technology object.

The leading value must not be reversed during synchronization.

Synchronous motion

Synchronous motion is synchronous starting from the specified synchronous positions. The "Synchronous" status is indicated by parameter "MC_GearInPos.InSync" = TRUE as well as in the "<TO>.StatusWord.X22 (Synchronous)" tag of the technology object. The following axis follows the position of the leading axis according to the gear ratio.

The response characteristic during gearing is expressed by a linear relationship between the leading value and following value.

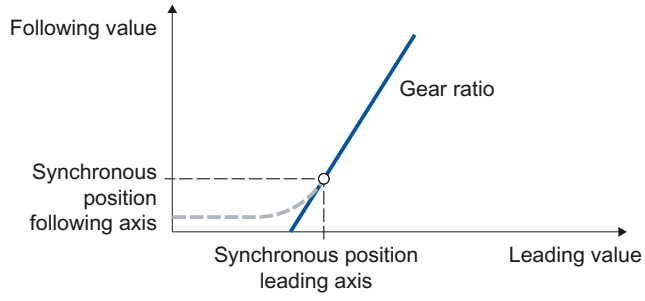

Gear ratio	Slope of line/transmission ratio Gear ratio = "MC_GearInPos.RatioNumerator"/"MC_GearInPos.RatioDenominator"
- - - -	Synchronization

The following value is calculated according to the following function:

Position of following axis (following value) = Synchronous position of following axis + gear ratio × (Position of leading axis - Synchronous position of leading axis)

Direction

The numerator of the gear ratio is specified as positive or negative. This yields the following behavior:

- Positive gear ratio:
  The leading and following axes move in the same direction.
- Negative gear ratio:
  The following axis moves in the opposite direction to the leading axis.

See also

Dynamic limits in synchronous operation (Page 1662)

Tags for gearing with "MC_GearInPos"

The following technology object tags are relevant for gearing with "MC_GearInPos":

Status indicators		
Tag	**Description**	
<TO>.StatusSynchronizedMotion.FunctionState	Indication of which synchronous operation function is active	
	0	No synchronous operation active
	1	Gearing ("MC_GearIn")
	2	Gearing with specified synchronous positions ("MC_GearIn-Pos")
	3	Camming ("MC_CamIn")
<TO>.StatusSynchronizedMotion.PhaseShift	Current absolute leading value shift in synchronous operation	

See also

Tags for synchronous operation (Page 1720)

Leading value shift in gearing

With the Motion Control instructions "MC_PhasingRelative" and "MC_PhasingAbsolute", the leading value can be shifted in gearing with "MC_GearIn" and "MC_GearInPos". The leading value shift is executed on the following axis. The leading axis is not affected.

The leading value shift only has an effect in the "Synchronous" status. If the synchronous operation is overridden, the leading value shift is reset to zero.

Camming

During camming, the leading axis and following axis are coupled by a synchronous operation function, which you specify using a cam (Page 1761).

Synchronous operation starts after synchronization when the specified synchronous position of the leading axis is reached. The corresponding synchronous position of the following axis is the result of the cam.

The utilized cam can be scaled on a job-related basis and applied shifted. To use the cam for camming, it must be interpolated. You interpolate the cam in your user program with the Motion Control instruction "MC_InterpolateCam".

Synchronization in advance

You specify the synchronous position of the leading axis relative to which the axes are to move synchronously in the Motion Control instruction "MC_CamIn". The synchronous operation is synchronized in advance of the specified synchronous position.

The synchronous position of the leading axis is determined by the following parameters:

- Start position of the cam (<TO_Cam>.StatusCam.StartLeadingValue)
- "MC_CamIn.MasterScaling" - Scaling the leading values of the cam
- "MC_CamIn.MasterOffset" - Offset/position of the cam
- "MC_CamIn.MasterSyncPosition" - Starting point within the cam

The synchronous position is calculated using the following equation:

Synchronous position = (Start position of the cam x "MasterScaling") + "MasterOffset" + "MasterSyncPosition"

The synchronization in advance (Page 1718) is performed using the dynamic parameters (velocity, acceleration, deceleration, jerk) or the leading value distance. The dynamic parameters and the leading value distance are specified in "MC_CamIn" and influence the starting position for synchronization. After the start of a "MC_CamIn" job, synchronization occurs in such a way that the leading and following axis move synchronously when the synchronous position is reached.

The synchronization is indicated by parameter "MC_CamIn.StartSync" = TRUE as well as in the "<TO>.StatusWord.X21 (Synchronizing)" tag of the technology object.

The leading value must not be reversed during synchronization.

Direct synchronization

When you set the parameter "MC_CamIn.SyncProfileReference" = 2, the status is set synchronously directly at the current leading value position and at the current following value position.

Synchronous motion

Synchronous operation is achieved when the following axis has reached the synchronous position and the dynamic values of the leading axis according to the synchronous operation function.

When a synchronous axis is synchronized to a leading value, the "Synchronous" status is indicated by parameter "MC_CamIn.InSync" = TRUE as well as in the "<TO>.StatusWord.X22 (Synchronous)" tag of the technology object. The following axis follows the position of the leading axis according to the synchronous operation function.

The response characteristic during camming is expressed by the cam curve.

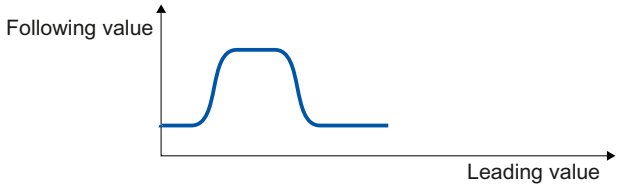

Following value = f(leading value)

See also

Dynamic limits in synchronous operation (Page 1662)

Scaling and offset of the cam

The scaling and shifting of the cam can be specified for camming in the Motion Control instruction "MC_CamIn". The configured cam is not changed by calling "MC_CamIn".

The following figure shows the basic sequence for scaling/shifting the cam:

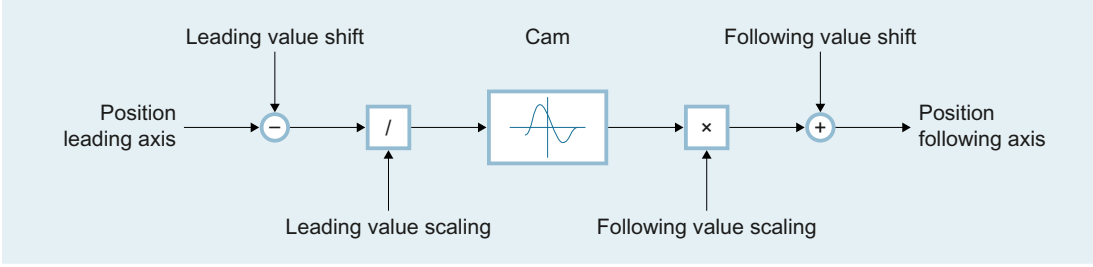

Position following axis = f[(Position leading axis - Leading value shift) / Leading value scaling] × Following value scaling + Following value shift

Cyclic and non-cyclic application of the cam

The "MC_CamIn.ApplicationMode" parameter can be used to set whether or not the cam is to be applied cyclically for synchronous operation:

- **Not cyclic**
  The cam is run exactly once. When the cam is run in the positive direction, synchronous operation is ended when the end point of the cam is reached. When the cam is run in the negative direction, synchronous operation is ended when the starting point of the cam is reached. To prevent step changes in the dynamic values, the velocity of the following axis must be zero at the starting and end points of the cam.

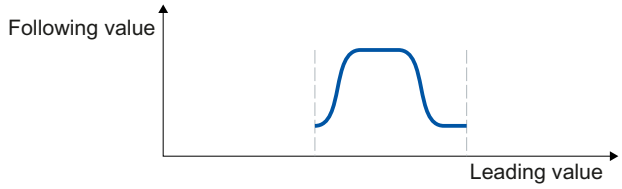

- **Cyclic**
  The cam is run cyclically. When the cam is run in the positive direction, the cam is repeated from the starting point when the end point of the cam is reached. When the cam is run in the negative direction, the cam is repeated from the end point when the starting point of the cam is reached. To prevent step changes in the dynamic values, the starting and end points of the cam must match.

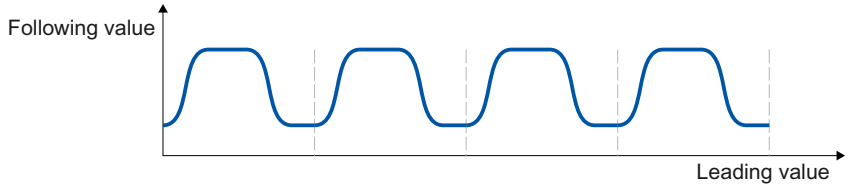

- **Cyclic appending**
  The cam is run cyclically. When the cam is run in the positive direction, the end point of the cam is used as the starting point for the next run. When the cam is run in the negative direction, the starting point of the cam is used as the starting point for the next run. The position difference between the starting and end points on the following value side is added up. To prevent step changes in the dynamic values, the velocity in the boundary points must be continuous.

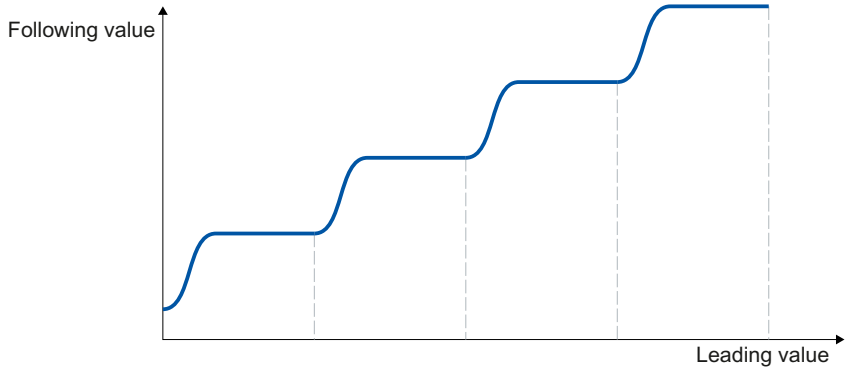

Tags for camming

The following technology object tags are relevant for camming:

Status indicators		
Tag	**Description**	
<TO>.StatusSynchronizedMotion.FunctionState	Indication of which synchronous operation function is active	
	0	No synchronous operation active
	1	Gearing ("MC_GearIn")
	2	Gearing with specified synchronous positions ("MC_GearIn-Pos")
	3	Camming ("MC_CamIn")
<TO>.StatusSynchronizedMotion.CurrentCam	Cam that is currently being used for camming	
<TO>.StatusSynchronizedMotion.MasterOffset	Current shift of the leading value range of the cam	
<TO>.StatusSynchronizedMotion.MasterScaling	Current scaling of the leading value range of the cam	
<TO>.StatusSynchronizedMotion.SlaveOffset	Current shift of the following value range of the cam	
<TO>.StatusSynchronizedMotion.SlaveScaling	Current scaling of the following value range of the cam	

See also

Tags for synchronous operation (Page 1720)

Synchronization in advance with "MC_GearInPos"/"MC_CamIn"

Synchronization using dynamic parameters

For synchronization using dynamic parameters, you specify the dynamics (velocity, acceleration, deceleration, jerk). The required travel distance (synchronization length) is calculated.

After the start of the corresponding Motion Control job, a motion profile for the following axis is calculated continuously. The motion profile is calculated based on the following parameters:

- Specified synchronous positions of the Motion Control instruction
- Specified dynamics of the Motion Control instruction
- Current position and dynamics of the leading and following axes
- Synchronous operation function

The calculation yields a synchronization length.

The calculation is ended and the following axis is moved to the specified synchronous position when the following position has been reached:

Start position = Synchronous position of leading axis - Synchronization length

The start of synchronization is indicated in the Motion Control instruction with parameter "StartSync" = TRUE.

The dynamics of the following axis during synchronization is obtained from the calculated motion profile and the current dynamics of the leading axis. Changes in the dynamics of the

leading axis during synchronization are superimposed on the calculated motion profile. This causes an increase in the dynamic value settings for the following axis, if necessary,

When the following axis has reached the synchronous position, the following axis is synchronized. The following axis travels synchronously with the leading axis. The "Synchronous" status is indicated in the Motion Control instruction with parameter "InSync" = TRUE.

When the leading axis is in its synchronous position when the corresponding Motion Control job is started, the leading axis must first cross the start position to start synchronization.

Synchronization using leading value distance

For synchronization using the leading value difference, you specify the synchronization length.

After the start of the corresponding Motion Control command, a motion profile with the required dynamics is calculated for the following-axis dependent on the specified leading value distance. For this, the leading axis must be at least the leading value distance from the synchronous position.

The following axis is moved to the specified synchronous position as soon as the leading value reaches the following position:

Start position = Synchronous position of leading axis - Synchronization length

The start of synchronization is indicated in the Motion Control instruction with parameter "StartSync" = TRUE.

The dynamics of the following axis during synchronization is obtained from the calculated motion profile and the current dynamics of the leading axis. Changes in the dynamics of the leading axis during synchronization are superimposed on the calculated motion profile in accordance with the synchronous operation function. This causes an increase in the dynamic value settings for the following axis, if necessary,

When the following axis has reached the synchronous position, the following axis is synchronized. The following axis travels synchronously with the leading axis. The "Synchronous" status is indicated in the Motion Control instruction with parameter "InSync" = TRUE.

Direct synchronous setting with "MC_CamIn"

This setting of the type of synchronization at "MC_CamIn" is suitable for synchronizing at standstill.

After the corresponding Motion Control job has started, the status is set synchronously directly at the current leading value position and at the current following value position ("InSync" = TRUE).

The point specified in the "MasterSyncPosition" parameter in the cam is assigned in the leading value range of the set position of the leading axis and in the following value range of the set position of the following axis. The cam is offset accordingly. The current offset results from the cam and is displayed at the "<TO>.StatusSynchronizedMotion.MasterOffset" and "<TO>.StatusSynchronizedMotion.SlaveOffset" tags of the technology object.

Simulate synchronous operation

An active synchronous operation connection is triggered when access enables are removed or four motion jobs on a following axis. To keep the synchronous operation active without overriding it, set the synchronous operation in simulation.

With the "MC_SynchronizedMotionSimulation" Motion Control instruction, you can simulate an active synchronous operation in simulation. The leading axis should be stopped at this time.

The synchronized motion simulation only affects the synchronized motion of the following axis. No setpoints are output from the synchronous operation to the drive. The setpoint output to the drive continues to come from the possibly superimposed motions of the following axis. The same applies to single axis commands during the synchronous operation.

Start a "MC_SynchronizedMotionSimulation" job only if the following axis is in synchronous operation. The status bit "Synchronous" is then active, "<TO>.StatusWord.X22" = TRUE. If the following axis is not or not yet in synchronous operation, the instruction is aborted with error.

The synchronous operation remains active in simulation, including the motions through single axis commands or disables of the leading and/or following axis, for example by opening a safety door. The following axis does not have to be synchronized again after the synchronized motion simulation has been completed.

To avoid the setpoint of the following axis jumping when the simulation has been completed, note the following points:

- When the simulation finishes, the leading in the following axis must both be at the starting positions that you had when the simulation was switched on.
- If the leading axis is at a different position than that where it was when the simulation was switched on, the following axis must be at the position that results from the synchronous operation relationship.
  You can calculate this position of the following axis beforehand in the user program and approach it with a motion job.

Tags for synchronous operation (S7-1500)

The following technology object tags are relevant for synchronous operation:

Status indicators		
Tag	**Description**	
<TO>.StatusSynchronizedMotion.ActualMaster	When a synchronous operation job is started, the number of the technology data block of the currently used leading axis is displayed.	
	0	Synchronous operation inactive
<TO>.StatusSynchronizedMotion.PhaseShift	Current absolute leading value shift	
<TO>.StatusWord.X21 (Synchronizing)	Set to the value "TRUE" when the synchronous axis synchronizes to a leading value.	
<TO>.StatusWord.X22 (Synchronous)	Set to the value "TRUE" when the synchronous axis is synchronized and moves synchronously to the leading axis.	
<TO>.ErrorWord.X14 (SynchronousError)	Error during synchronous operation The leading axis specified in the Motion Control instruction was not configured as a possible leading axis.	

Status indicators	
Tag	Description
<TO>.StatusSynchronizedMotion.StatusWord.X0 (MaxVelocityExceeded)	Set to the value "TRUE" when the maximum velocity configured for the following axis is exceeded during synchronous operation.
<TO>.StatusSynchronizedMotion.StatusWord.X1 (MaxAccelerationExceeded)	Set to the value "TRUE" when the maximum acceleration configured for the following axis is exceeded during synchronous operation.
<TO>.StatusSynchronizedMotion.StatusWord.X2 (MaxDecelerationExceeded)	Set to the value "TRUE" when the maximum deceleration configured for the following axis is exceeded during synchronous operation.

See also

Tags for camming (Page 1717)

Tags for gearing with "MC_GearInPos" (Page 1713)

Technology object external encoder (S7-1500)

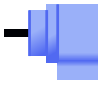

The external encoder technology object detects a position, and makes it available to the controller.

The actual position detected by the external encoder can be used for the following functions, for example:

- Measured value acquisition by a measuring input
- Position-dependent generation of switching signals and switching signal sequences by output cam and cam track with actual value reference.
- As a leading value of a synchronous axis (S7-1500T)

You can find an overview of the functions of the external encoder technology object in the Functions (Page 1626) section.

The relationship between the encoder values and a defined position is established by the parameter assignment of the mechanical properties and encoder settings and by a homing operation.

The following figure shows the basic principle of operation of the external encoder technology object:

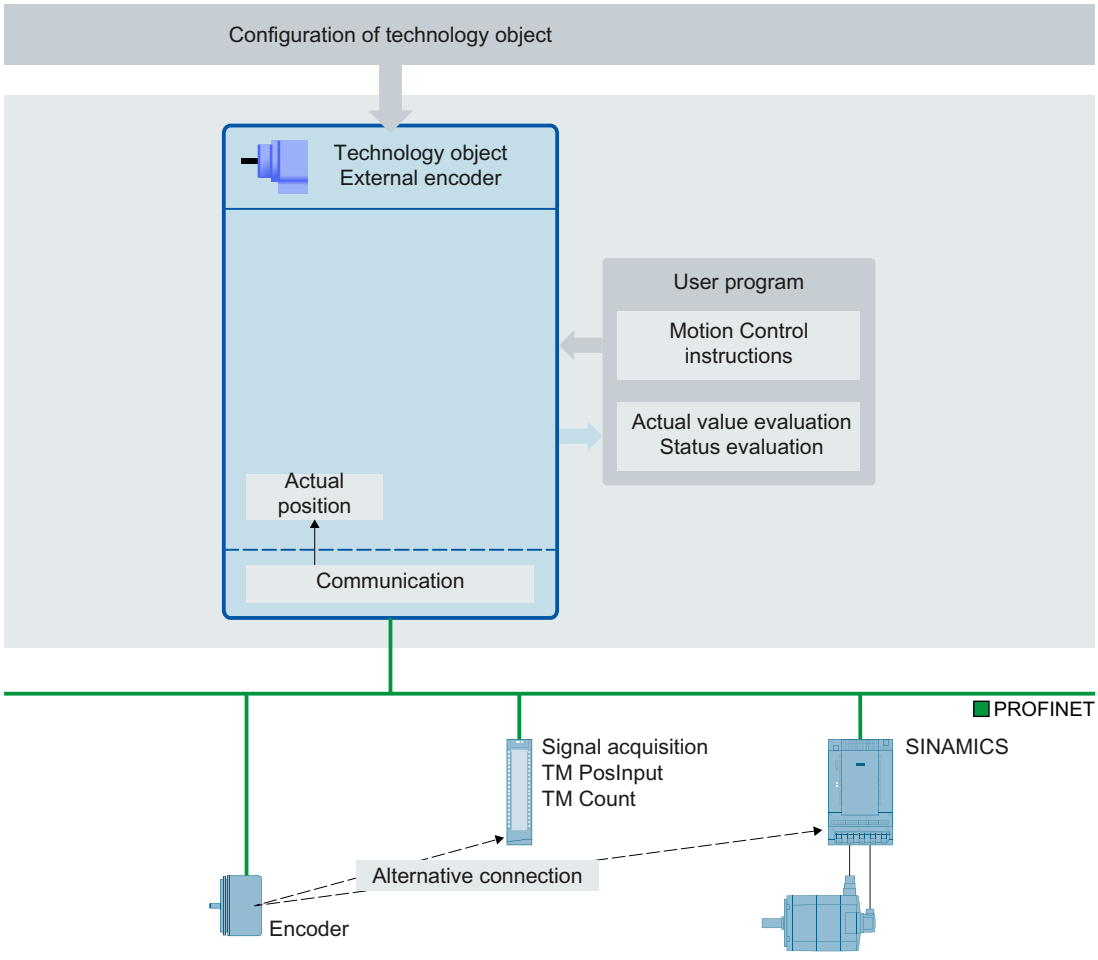

Specification of the position occurs according to the selected system of units:

- **Linear system of units**
  The position is specified as a linear measure, e.g. millimeters.
- **Rotary system of units**
  The position is specified as an angular measure, e.g. degrees.

Technology object measuring input (S7-1500)

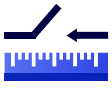

The measuring input technology object acquires the actual position of an axis or external encoder at a signal change at the measuring input.

You can find an overview of the functions of the measuring input technology object in the Functions (Page 1626) section.

The figure below shows the basic principle of operation of the measuring input technology object:

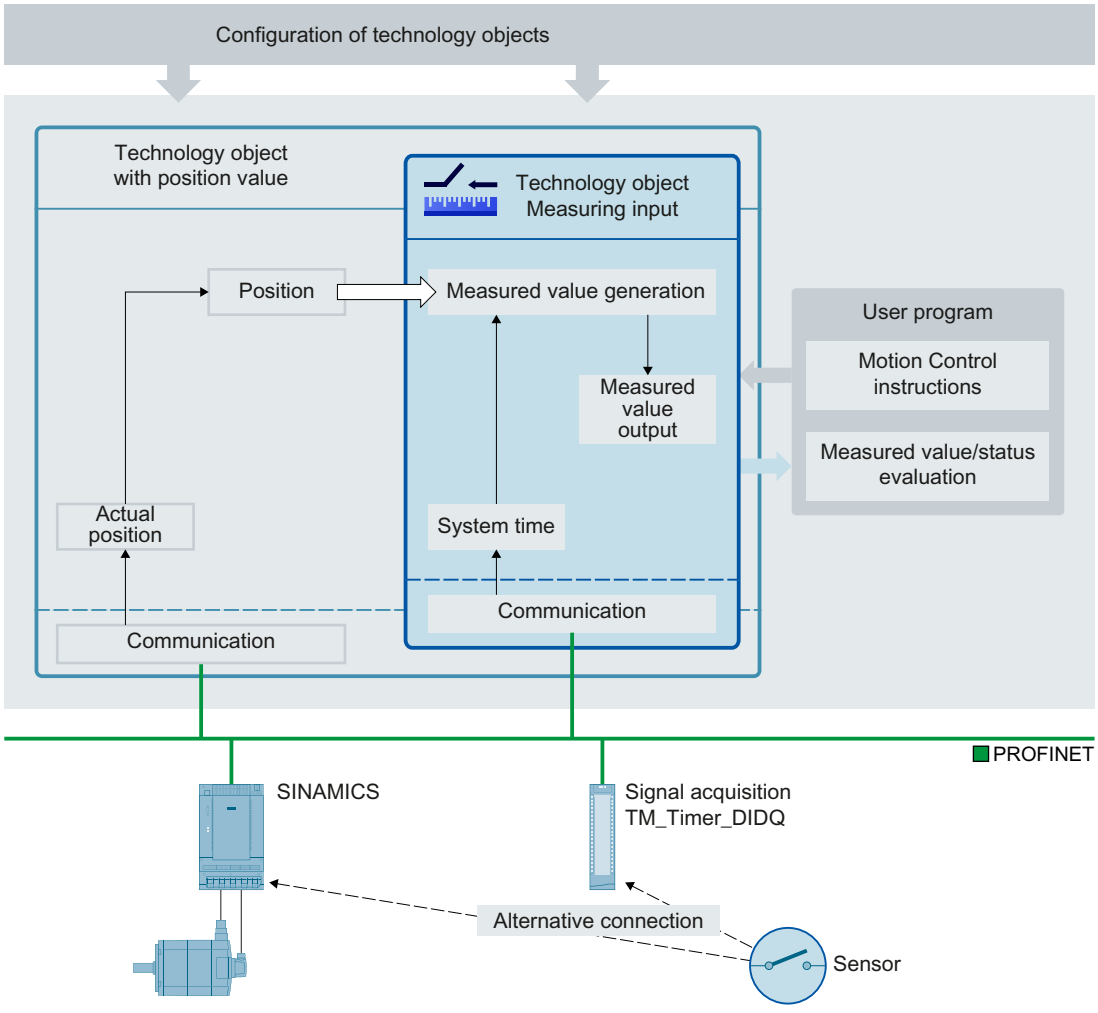

Measurement types

The following types of measurement can be performed:

- One-time measurement (Page 1725)
  Up to two measured values are acquired with edge accuracy with one measuring job. A one-time measuring job is started with "MC_MeasuringInput".
- Cyclic measuring (Page 1726)
  With cyclic measuring, up to two measured values are acquired with edge accuracy in each position control cycle.
  A cyclic measuring job is started with "MC_MeasuringInputCyclic". The measurements are continued cyclically until they are ended per command.

The edges to be detected are selected when starting the measurement using Motion Control instruction "MC_MeasuringInput" or "MC_MeasuringInputCyclic".

Assignment

The measuring input technology object must always be assigned to another technology object whose position will be evaluated by the measuring input.

The measuring input technology object can be assigned to the following technology objects:

- Synchronous axis
- Positioning axis
- External encoder

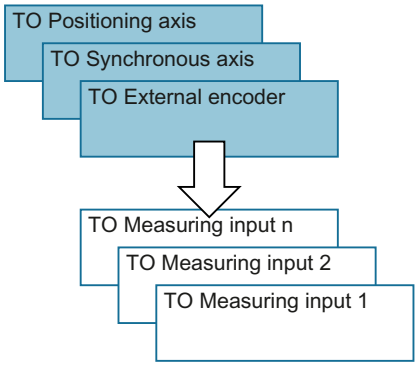

Exactly one axis or one external encoder can be assigned to a measuring input technology object.

An axis or external encoder can be assigned multiple measuring input technology objects.

Measured value determination

The position can be detected using support from the hardware in one of the following ways:

- Measuring using TM Timer DIDQ (time-based)
- Measuring with SINAMICS measuring input (time-based)
  Measurements via a measuring input are only possible for SINAMICS drives, see "Compatibility list (Page 2146)".
- Measurement using PROFIdrive telegram (drive or external encoder)

A measuring job is activated using Motion Control instruction "MC_MeasuringInput" or "MC_MeasuringInputCyclic".

With time-based measuring ("TIME_BASED"), a signal change of the external trigger signal for the measurement is detected via a corresponding module such as TM Timer DIDQ or SINAMICS Drive. The time stamp is transferred to the controller and the associated actual position is determined in the technology object.

With direct position detection ("PROFIDRIVE"), the measurement input is connected to the drive device and the measurement is implemented in the drive. The drive or encoder module directly supplies the acquired position value to the technology object via the PROFIdrive telegram.

The resulting measured value is indicated at the respective output of the Motion Control instruction "MC_MeasuringInput" or "MC_MeasuringInputCyclic".

Cyclic measuring is only possible when measuring using TM Timer DIDQ. One-time measuring is always possible.

Measuring using TM Timer DIDQ requires isochronous mode.

For measuring using PROFIdrive telegram, only one measuring input at a time can be active on an actual value or encoder in the PROFIdrive telegram. A maximum of two measuring inputs can be configured on an actual value or encoder in the PROFIdrive telegram via PROFIdrive, see section Automatic transfer of drive and encoder parameters in the device (Page 1641).

The measuring input technology object cannot be used with SIMATIC S7-PLCSIM. The measuring input technology object and the measuring input jobs used in the user program can be loaded into SIMATIC S7-PLCSIM, but have no function. Measured values are not shown.

Correction time

The time of the measurement can be corrected by setting a correction time (<TO>.Parameter.CorrectionTime) for the measuring input technology object.

Corrections may be required for the following examples:

- Times for mechanical displacement of the measuring input
- Times for the generation of the measured signal before the input at the measurement module
- Filter times at the input or filter times for the measurement inputs on the SINAMICS drive unit

The correction time is calculated by the measuring input technology object for time-based measured value acquisition ("TIME_BASED") and for direct position detection ("PROFIDRIVE").

Also note the time delay when measuring on a virtual axis, see section Actual values (Page 1639).

Measuring (S7-1500)

One-time measurement (S7-1500)

With one-time measurement, up to two edges can be detected with one measuring job. The associated actual positions are signaled back in the function block and in the technology DB and can be further processed in the user program.

Measuring job

A measuring job is started using the Motion Control instruction "MC_MeasuringInput". The "<TO>.Status" tag in the technology data block changes to "WAITING_FOR_TRIGGER". The technology object activates the measurement when the selected edge is detected.

The measurement occurs at the measurement input in the form of up to two system times. Based on the times, the associated position is determined and output, taking into consideration a correction time, if present.

With direct position detection, the detected position value is supplied directly from the drive or encoder module to the technology object via the PROFIdrive telegram.

The measurement is then finished. An additional measurement must be restarted using Motion Control instruction "MC_MeasuringInput".

Configured "Mode" parameter in Motion Control instruction "MC_MeasuringInput"	Output of Motion Control instruction "MC_MeasuringInput"	
	"MeasuredValue1"	"MeasuredValue2"
Measure positive edge only	Actual position at the time of the edge	–
Measure negative edge only		
Measure the next two edges	Actual position at the time of the first edge	Actual position at the time of the second edge
Measure the next two edges starting with the positive edge	Actual position at the time of the positive edge	Actual position at the time of the negative edge
Measure the next two edges starting with the negative edge	Actual position at the time of the negative edge	Actual position at the time of the positive edge

The last detected values are set in the technology data block. If a new job is initiated with the function block, the outputs of the function block are initialized. The technology data block is not initialized. After detecting the first valid measuring cycle, the values in the technology data block and function block are consistent with one another.

The finished measuring job is indicated in the function block in "MC_MeasuringInput.DONE" = TRUE or in the technology data block in "<TO>.Status" = "TRIGGER_OCCURRED".

Temporal requirements for measuring jobs via "MC_MeasuringInput"

The hardware-related restrictions to measurement via the PROFIdrive telegram or measuring via the SINAMICS measuring input, there are time requirements for the period until the measuring event can be captured.

When measuring once via "MC_MeasuringInput" with "Mode" = 3 (measurement at both edges, beginning with the rising edge) or "Mode" = 4 (measurement at both edges, beginning with the falling edge), a minimum interval of several servo cycles is therefore required between the first edge to be measured and the previous edge, so that the first edge to be measured can be recorded.

You can find information on the temporal boundary conditions in the section Time-related boundary conditions (Page 1731).

See also

Time-related boundary conditions (Page 1731)

Cyclic measuring (S7-1500)

With cyclic measuring, up to two measuring events can be acquired by the system in each position control cycle of the technology object and the associated measuring positions can be displayed. The measurements are continued cyclically until they are ended per command.

The determined measured values are displayed and can be read by the user program.

Cyclic measuring requires the measured value to be determined using the following type of hardware support: "Measuring using Timer DIDQ".

Measuring edges at which a measured value could not be determined are indicated in a lost edge counter in the technology DB as well as in function block "MC_MeasuringInputCyclic".

Measuring job

A cyclic measuring job is started with Motion Control instruction "MC_MeasuringInputCyclic", and the measuring job is issued to the corresponding technology module. Depending on the functionality of the technology module, up to two measuring events and thus measuring times can be captured with edge reference in each position control cycle and then forwarded to the technology object. The technology object determines the measuring positions for the measuring times taking into consideration any specified correction times.

The technology data block tag "<TO>.Status" changes from "INACTIVE" to "WAITING_FOR_TRIGGER" and remains in this status as long as additional events are awaited.

The mode set in the Motion Control instruction specifies the edges for which the measured values are to be acquired. At most, the following edges can be detected in each position control cycle:

- Two positive edges when detecting positive edges
- Two negative edges when detecting negative edges
- One positive edge and one negative edge when detecting positive and negative edges

Measured values and counters

With a positive edge at the input "MC_MeasuringInputCyclic.Execute", outputs "MeasuredValue1Counter" and "MeasuredValue2Counter" are reset to "0". As a result, new events can be tracked immediately and new measured value entries can be detected.

All measuring event occurrences of the measuring job are incremented by "1" in the corresponding event counters "<TO>.MeasuredValues.MeasuredValue1Counter" and "<TO>.MeasuredValues.MeasuredValue2Counter" of the technology data block.

The acquired measured values are continuously captured in the technology data block irrespective of individual jobs and the values are only reset to "0" at power-up or restart of the technology object.

After a completed measurement, the measured values are output in the function block. The counters of the function block are set to "0" at a new measuring job. The measured value output in the technology data block always indicates the last acquired measured value.

Lost edge counter (LEC)

If more than two edges to be detected occur within one position control cycle, a measured value cannot be evaluated for the other edges to be detected. The number of lost edges is recorded in the LEC.

The lost edges that are recorded in the LEC depend on the mode set in the Motion Control instruction. For example, if only positive edges are to be measured, the LEC records only the non-measured positive edges.

A maximum of seven lost edges can be counted and displayed in the LEC.

The number of lost edges is indicated in the function block and in the technology data block in:

- "LostEdgeCounter1"
  Lost cleared edges from the position control cycle in which "MeasuredValue1" was acquired.
  ⇒ The displayed value in "LostEdgeCounter1" is updated when counter "MeasuredValueCounter1" is incremented.
- "LostEdgeCounter2"
  Lost cleared edges from the position control cycle in which "MeasuredValue2" was acquired.
  ⇒ The displayed value in "LostEdgeCounter2" is updated when counter "MeasuredValueCounter2" is incremented.

Display of measurement results when using cyclic measuring

<table>
<tr><th rowspan="2">Edges selected in the command</th><th colspan="4">Display per position control cycle</th></tr>
<tr><th>"MeasuredValue1"</th><th>"MeasuredValue2"</th><th>"LostEdgeCounter1"</th><th>"LostEdgeCounter2"</th></tr>
<tr><td>Detect positive edges only
"MC_MeasuringInputCyclic.-Mode" = 0</td><td>Actual position at the time of the first positive edge</td><td>Actual position at the time of the second positive edge</td><td colspan="2" rowspan="2">Number of positive or negative edges in excess of two in the position control cycle of acquisition of "MeasuredValue1" and "MeasuredValue2".
The following applies here:
• If a "MeasuredValue1" and a "MeasuredValue2" are acquired, the number of acquired and lost edges indicated in "LostEdgeCounter1" and "LostEdgeCounter2" are the same.
• If only one "MeasuredValue1" is acquired, the "LostEdgeCounter1" is reset to "0". The value in "LostEdgeCounter2" remains changed.</td></tr>
<tr><td>Detect negative edges only
"MC_MeasuringInputCyclic.-Mode" = 1</td><td>Actual position at the time of the first negative edge</td><td>Actual position at the time of the second negative edge</td></tr>
<tr><td>Detect positive and negative edges
"MC_MeasuringInputCyclic.-Mode" = 2</td><td>Actual position at the time of the first positive edge in the position control cycle</td><td>Actual position at the time of the first negative edge in the position control cycle</td><td colspan="2">Number of edges in excess of two in the position control cycle of acquisition of "MeasuredValue1" and "MeasuredValue2".
The following applies here:
• If a "MeasuredValue1" and a "MeasuredValue2" are acquired, the number of acquired and lost edges indicated in "LostEdgeCounter1" and "LostEdgeCounter2" are the same.
• If only one "MeasuredValue1" is acquired, the "LostEdgeCounter1" is reset to "0". The value in "LostEdgeCounter2" remains changed.
• If only one "MeasuredValue2" is acquired, the "LostEdgeCounter2" is reset to "0". The value in "LostEdgeCounter1" remains changed.</td></tr>
</table>

Examples

The following figures show examples of divergence of "MeasuredValue1Counter" and "MeasuredValue2Counter" as a result of lost edges.

Example: Measurement at positive edges (Mode = 0)

Processing cycle clock TO Measuring input	N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10
Measuring input											
MeasuredValue1Counter	...	6	7	7	8	9	9	10	11	12	13
LostEdgeCounter1	...	0	0	0	0	0	0	1	0	1	0
MeasuredValue2Counter	...	6	7	7	7	7	7	8	8	9	9
LostEdgeCounter2	...	0	0	0	0	0	0	1	1	1	1

Example: Measurement at negative edges (Mode = 1)

Processing cycle clock TO Measuring input	N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10
Measuring input											
MeasuredValue1Counter	...	6	7	8	9	10	10	11	11	12	13
LostEdgeCounter1	...	0	0	0	1	0	0	1	1	1	0
MeasuredValue2Counter	...	6	7	7	8	8	8	9	9	10	10
LostEdgeCounter2	...	0	0	0	1	1	1	1	1	1	1

Example: Measurement at positive and negative edges (Mode = 2)

Processing cycle clock TO Measuring input	N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10
Measuring input											
MeasuredValue1Counter	...	6	7	7	8	9	9	10	11	12	13
LostEdgeCounter1	...	0	2	2	0	0	0	3	0	4	0
MeasuredValue2Counter	...	6	7	8	8	9	10	11	12	13	14
LostEdgeCounter2	...	0	2	0	0	0	0	3	0	4	0

Measuring with measuring range (S7-1500)

A measuring job can be activated directly or restricted to a defined measuring range.

The following graphic shows an example of measuring with measuring range in "Mode" = 0 (measurement of next positive edge):

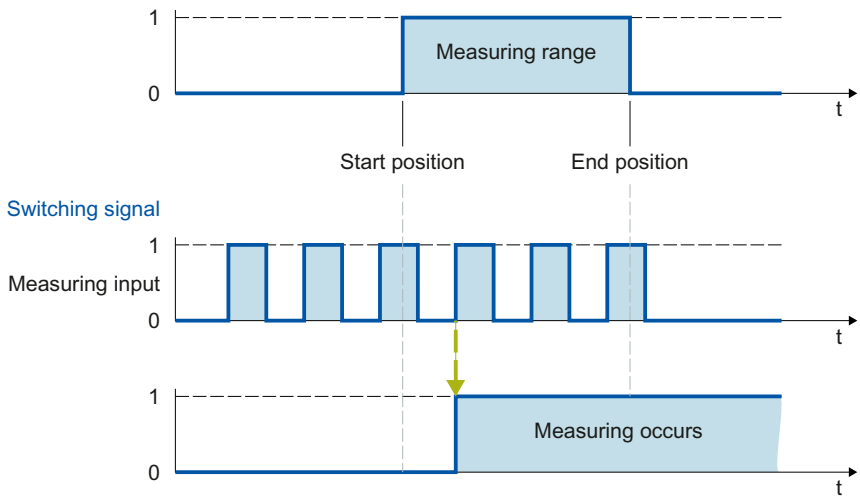

Only measured values within the measuring range are displayed for the technology object.

- If no measuring edge within the measuring range is detected during a one-time measurement, the measuring job is canceled and an alarm is triggered.
- Cyclic measuring remains active even if no measuring edge within the measuring range was detected.

For axes without modulo function, it is immaterial in which order the start and end positions are specified. If the start position is greater than the end position, the two values are interchanged in the application. If the start position for an axis with modulo function is greater than the end position, the measuring range is extended from the start position past the modulo transition of the axis to the end position.

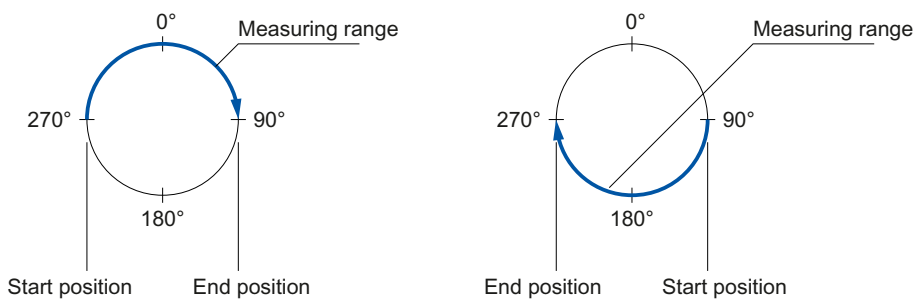

The measuring range positions are specified in the Motion Control instruction "MC_MeasuringInput" or "MC_MeasuringInputCyclic".

Activation time for the measuring range

The measuring function must be active at the measurement input when the start of the measuring range is reached. To compensate for the communication time for activation in the TM Timer DIDQ or drive, for example, the activation of the measurement in the technology object begins earlier then the measuring range start by the amount of the activation time.

The activation time for measuring with measuring range is divided up as follows:

- The activation time allocation defined and active on the system side is indicated in the "<TO>.Parameter.MeasuringRangeActivationTime" tag.
- An additional activation time can also be set by the user with the "<TO>.Parameter.MeasuringRangeAdditionalActivationTime" tag.

Time-related boundary conditions (S7-1500)

Depending on the hardware configuration and selection of edges to be detected, different system-inherent requirements apply to the time allowed after calling the Motion Control instruction "MC_MeasuringInput" or "MC_MeasuringInputCyclic" until a measurement occurs and the results are displayed.

The following times must be distinguished here:

- Time until the measuring event can be captured
- Time until the measurement result is displayed or the measurement is finished.

The times calculated taking into account the current settings are indicated in configuration window "Extended parameters" of a measuring input.

Measuring using TM Timer DIDQ/Measuring using SINAMICS measuring input

- Time from output of an "MC_MeasuringInput" job until measuring event detection becomes effective:
  - "MeasuringRangeActivationTime": 2 x T_{servo}
- Time after measuring event until measured value is available in the controller:
  - One-time measurement: 2 x T_{servo} + T_{Send}
  - Cyclic measuring: T_{servo} + T_{Send}
- Activation time for a measurement with measuring range:
  - "MeasuringRangeActivationTime" + "MeasuringRangeAdditionalActivationTime"

Measurement using PROFIdrive telegram (drive or external encoder)

- "MeasuringRangeActivationTime": 4 x T_{servo}
- Time from output of an "MC_MeasuringInput" job until measuring event detection becomes effective:
  - Measuring a positive/negative edge or two edges:
    "MeasuringRangeActivationTime" + 2 x T_{servo}
  - Measuring two dedicated edges:
    "MeasuringRangeActivationTime" + 3 x T_{servo}

- Time after measuring event until measured value is available in the controller:
  - Measuring an edge: 7 x T_{servo}
  - Measuring two edges: 13 x T_{servo}
- Activation time for a measurement with measuring range:
  - Measuring a positive/negative edge or two edges: "MeasuringRangeActivationTime" + "MeasuringRangeAdditionalActivationTime" + 2 x T_{servo}
  - Measuring two dedicated edges: "MeasuringRangeActivationTime" + "MeasuringRangeAdditionalActivationTime" + 3 x T_{servo}

Definition of tags

- T_{servo} = Call interval of the technology object in the servo cycle clock [ms]
- T_{Send} = Send clock [ms]
- "MeasuringRangeActivationTime" = See section Measuring with measuring range (Page 1728)

To prevent asynchronous processing from overwriting a measured value that was just determined, a new one-time measuring job cannot be started until the active measurement has concluded. The sum of the activation time and the time until display or conclusion must be taken into account for this.

In the case of cyclic measuring, evaluation or temporary storage of the measurement results in the synchronous MC-PostServo [OB95] is recommended from the user perspective.

Tags (S7-1500)

The following technology object tags are relevant for measuring:

Status indicator		
Tag	**Description**	
<TO>.Status	Status of the measuring input function	
	0	Measurement is not active ("INACTIVE")
	1	The measuring input is waiting for a measuring event ("WAITING_FOR_TRIGGER")
	2	The measuring input has acquired one or more measured values ("TRIGGER_OCCURRED").
	3	Error during the measurement ("MEASURING_ERROR")
<TO>.InputState	Status of measuring input	

Parameters	
Tag	**Description**
<TO>.Parameter.MeasuringInputType	Measuring input type
<TO>.Parameter.PROFIdriveProbeNumber	Number of the measuring input to be used for a measurement via PROFIdrive telegram

Parameters	
Tag	**Description**
<TO>.Parameter.MeasuringRangeActivation-Time	System-defined activation time allocation [ms]
<TO>.Parameter.MeasuringRangeAdditional-ActivationTime	Additional user-defined activation time allocation [ms]
<TO>.Parameter.CorrectionTime	User-defined correction time for the measurement result [ms]

interface	
Tag	**Description**
<TO>.Interface.Address	I/O address for the digital measuring input

Units	
Tag	**Description**
<TO>.Units.LengthUnit	Unit of the length data
<TO>.Units.TimeUnit	Unit of the time data

MeasuredValues	
Tag	**Description**
<TO>.MeasuredValues.MeasuredValue1	First measured value
<TO>.MeasuredValues.MeasuredValue2	Second measured value (when measuring two or more edges in one position control cycle)
<TO>.MeasuredValues.MeasuredValue1Counter	Count value for the first measured value
<TO>.MeasuredValues.MeasuredValue2Counter	Count value for the second measured value
<TO>.MeasuredValues.LostEdgeCounter1	Lost edges in the cycle clock of the first measured value acquisition (zero in the case of one-time measurement)
<TO>.MeasuredValues.LostEdgeCounter2	Lost edges in the cycle clock of the second measured value acquisition (zero in the case of one-time measurement)

StatusWord	
Tag	**Description**
<TO>.StatusWord.X0 (Control)	The technology object is in operation.
<TO>.StatusWord.X1 (Error)	An error occurred at the technology object.
<TO>.StatusWord.X2 (RestartActive)	The technology object is being reinitialized. The tags of the technology data block are not updated with active restart.
<TO>.StatusWord.X3 (OnlineStartValuesChanged)	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object.
<TO>.StatusWord.X5 (CommunicationOk)	The measuring input is synchronized with the measuring module and can be used.

ErrorWord	
Tag	**Description**
<TO>.ErrorWord.X0 (SystemFault)	A system-internal error has occurred.
<TO>.ErrorWord.X1 (ConfigFault)	Configuration error One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program.
<TO>.ErrorWord.X2 (UserFault)	Error in user program at a Motion Control instruction or its use
<TO>.ErrorWord.X3 (CommandNotAccepted)	Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
<TO>.ErrorWord.X13 (PeripheralError)	Error accessing a logical address

ErrorDetail		
Tag	**Description**	
<TO>.ErrorDetail.Number	Alarm number	
<TO>.ErrorDetail.Reaction	Effective alarm reaction	
	0	No reaction
	6	End measuring input processing

Technology object output cam (S7-1500)

The output cam technology object generates switching signals depending on the position of an axis or external encoder. The switching states can be evaluated in the user program and fed to digital outputs.

You can find an overview of the functions of the output cam technology object in the Functions (Page 1626) section.

The figure below shows the basic operating principle of the output cam technology object:

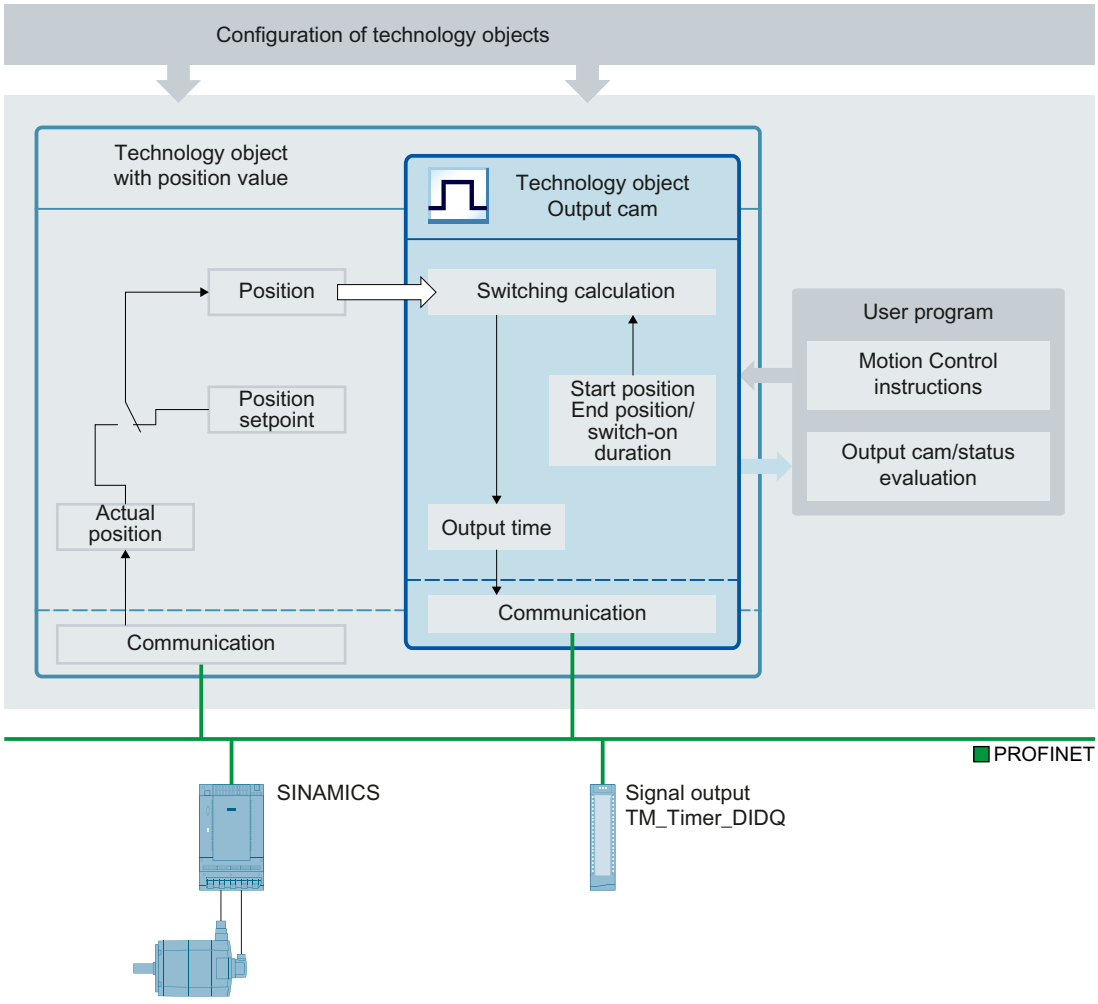

Output cam types

The following output cam types can be used:

- Distance output cam (Page 1737)
  Distance output cams switch on between the start position and end position. Outside this range, the distance output cam is switched off.
- Time-based output cam (Page 1739)
  Time-based output cams switch on for a defined time period when the start position is reached.

Assignment

The output cam technology object must always be assigned to another technology object whose position will be evaluated.

The output cam technology object can be assigned to the following technology objects:

- Synchronous axis
- Positioning axis
- External encoder

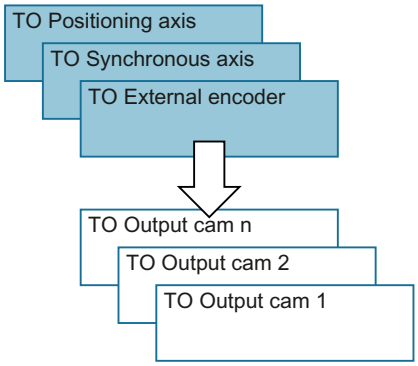

Exactly one axis or one external encoder can be assigned to an output cam.

Multiple output cams can be assigned to one axis or external encoder.

Output cam calculation and output cam output

The output cam technology object calculates the exact switching time, thereby ensuring exact adherence to the switching positions. The switching time is calculated two position control cycles before the output.

The following output options are available for the digital cam output:

- TM Timer DIDQ
  Digital output with high degree of accuracy and reproducibility in the microsecond range on ET 200MP TM Timer DIDQ 16x24V and ET 200SP TM Timer DIDQ 10x24V time-based IO modules
  Output by TM Timer DIDQ requires isochronous mode. Isochronous mode is only possible with PROFINET interface modules.
- Digital output module
  Digital output with switching accuracy depending on the output cycle of the I/O used

When output is deactivated, the output cam status is not output at the hardware output. The output cam status can be used internally in the user program by evaluating the relevant "<TO>.CamOutput" tag.

Inverted output

In the case of inverted output, the range in which the cam output is switched on and the range in which it is switched off are swapped.

The inverted output is set in Motion Control instruction "MC_OutputCam" and is active when the instruction is enabled.

The inverted output can be used for both distance output cams and time-based output cams.

Output of multiple output cams to one output

The output of multiple output cams to one output is performed with either an AND or OR logic operation of the output cam signals to the output.

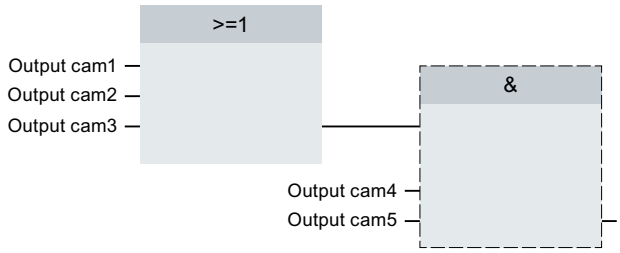

Display of the switching state

The switching state of the output cam is displayed in the associated technology DB in "<TO>.CamOutput".

Position reference

The switching points of the output cams can be referenced to the following positions, depending on the interconnected technology object.

- Actual position of a synchronous axis/positioning axis
- Position setpoint of a synchronous axis/positioning axis
- Position of an external encoder

Homing the interconnected technology object

A change to the position of an axis or external encoder using Motion Control instruction "MC_Home" is regarded as a sudden position change.

- Distance output cams are either skipped or correspondingly output.
- Time-based output cams are skipped. A time-based output cam is switched on only when the start position is overtraveled and remains switched on for the switch-on duration.
- Switched time-based output cams are not canceled by a homing operation.

Distance output cam (S7-1500)

Switch-on range

The switch-on range of distance output cams is basically defined by the start position and end position.

Start position smaller than end position

When the start position is less than the end position, the switch-on range begins with the start position and ends with the end position.

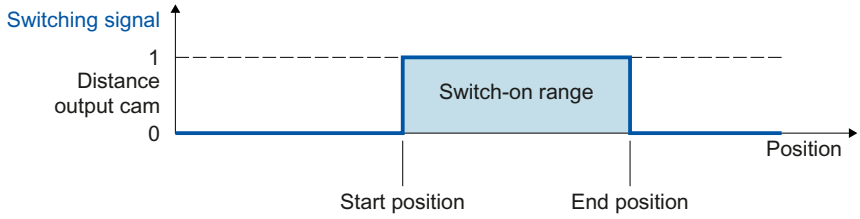

Start position greater than end position

When the start position is greater than the end position, there are two switch-on ranges as follows:

- Switch-on range beginning with the start position and ending with the positive range end (e.g. positive software limit switch, end of modulo range)
- Switch-on range beginning with the negative range end (e.g. negative software limit switch, start of modulo range) and ending with the end position

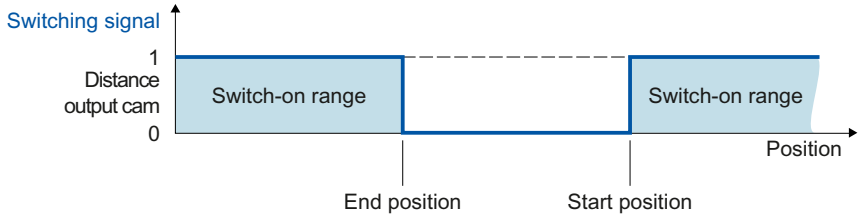

Mapping to an axis with modulo function

With active modulo function of the interconnected technology object, the start and end positions of the output cam are automatically mapped to values within the modulo range.

Example

- Modulo range = 0° to 50°
- Output cam start position = 80°
- Output cam end position = 220°

⇒ The distance output cam switches on at 30° and switches off again at 20°.

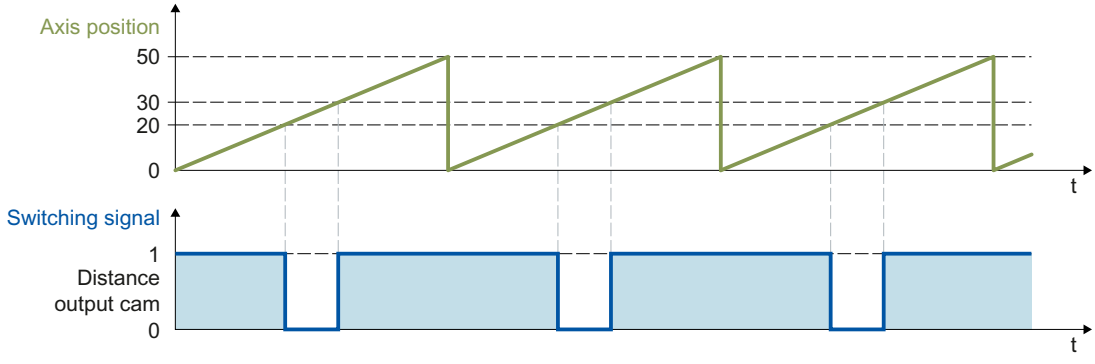

Switching characteristics

After activation, a distance output cam switches on in the following cases:

- The position of the interconnected technology object reaches the start or end position in the activation direction configured in Motion Control instruction "MC_OutputCam".
- The position of the interconnected technology object is moved into the switch-on range of the output cam (e.g. during homing) in the activation direction configured in Motion Control instruction "MC_OutputCam". If both activation directions are enabled in Motion Control instruction "MC_OutputCam", the output cam switches on even when the interconnected technology object is at a standstill.
- The output cam is switched on permanently using Motion Control instruction "MC_OutputCam" with "Mode" = 3.

An active distance output cam switches off in the following cases:

- The position is outside the switch-on range of the output cam.
- The position value is moved outside the switch-on range of the output cam.
- Motion Control instruction "MC_OutputCam" is set to "Enable" = FALSE.
- The motion direction of the interconnected technology object is reversed and no longer agrees with the enabled activation direction.

Time-based output cam (S7-1500)

A time-based output cam switches on at the start position and remains set for the on-load factor.

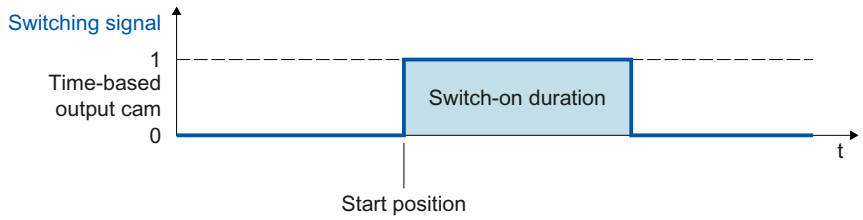

Mapping to an axis with modulo function

In the case of active modulo function of the interconnected technology object, the start position of the output cam is automatically mapped to the value within the modulo range.

Example

- Modulo range = 0° to 50°
- Output cam start positions:
  - Output cam 1 = 20°
  - Output cam 2 = 30°
  - Output cam 3 = 80°

⇒ The time-based output cam 1 switches on at 20°, the time-based output cams 2 and 3 switch on at 30°. All three output cams remain active the set on-load factor in each case.

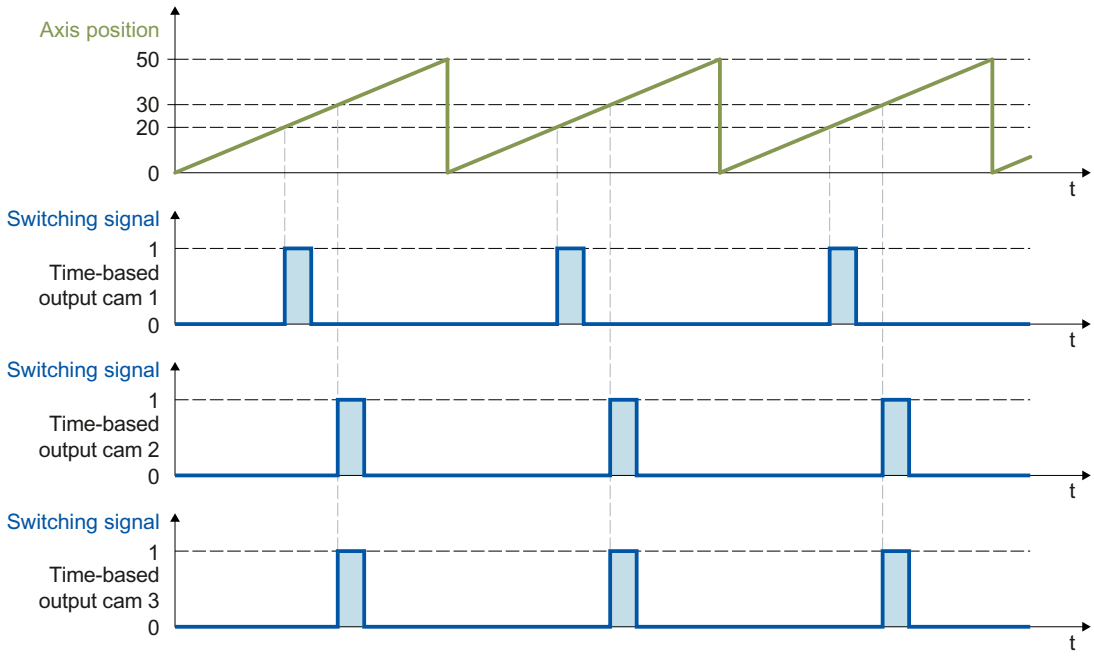

Switching characteristics

A time-based output cam switches on in the following cases:

- The start position has been reached and the motion direction of the interconnected technology object corresponds to the effective direction enabled by the instruction.

Note

- If the start position is reached again while an output cam is switched on, the on-load factor is not re-triggered.
- If due to the Motion Control instruction "MC_Home", the position value of the interconnected technology object is placed directly on or behind the start position of the output cam during the motion, the output cam does not switch on.

A time-based output cam switches off in the following cases:

- The configured on-load factor has expired.
- Motion Control instruction "MC_OutputCam" is set to "Enable" = FALSE.

Activation direction of cam (S7-1500)

An output cam can be switched depending on the motion direction of the interconnected technology object. An output of the output cam in positive or negative motion direction or independent of direction is possible.

The effective direction is set in "MC_OutputCam.Direction".

The following examples show the behavior of the output cam as a function of the effective direction setting.

Example of "positive" effective direction ("Direction" = 1)

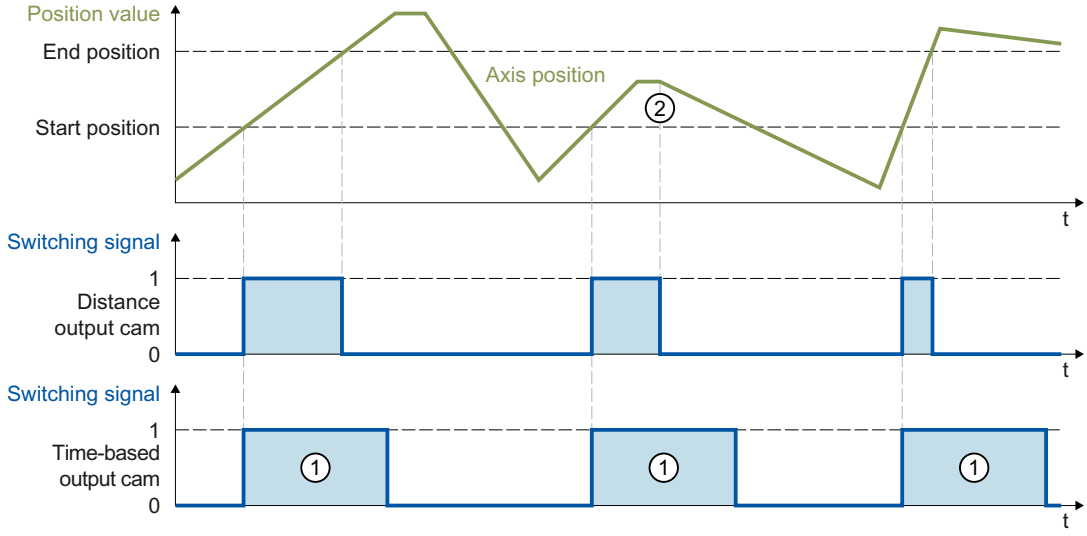

Distance output cam

The output cam switches on when the switch-on range is reached in the **positive** direction. At a direction reversal ②, the output cam switches off.

If the position value is moved into the switch-on range of the output cam, the cam switches on when there is positive motion direction of the interconnected technology object. The output cam remains switched off when the interconnected technology object is at a standstill.

Time-based output cam

The output cam switches on when the start position is reached in the **positive** direction. At a direction reversal, the output cam remains switched on for the specified on-load factor ①.

If during the homing motion, the position value of the interconnected technology object is directly on or behind the start position of the output cam, the output cam does not switch on.

Example of "negative" effective direction ("Direction" = 2)

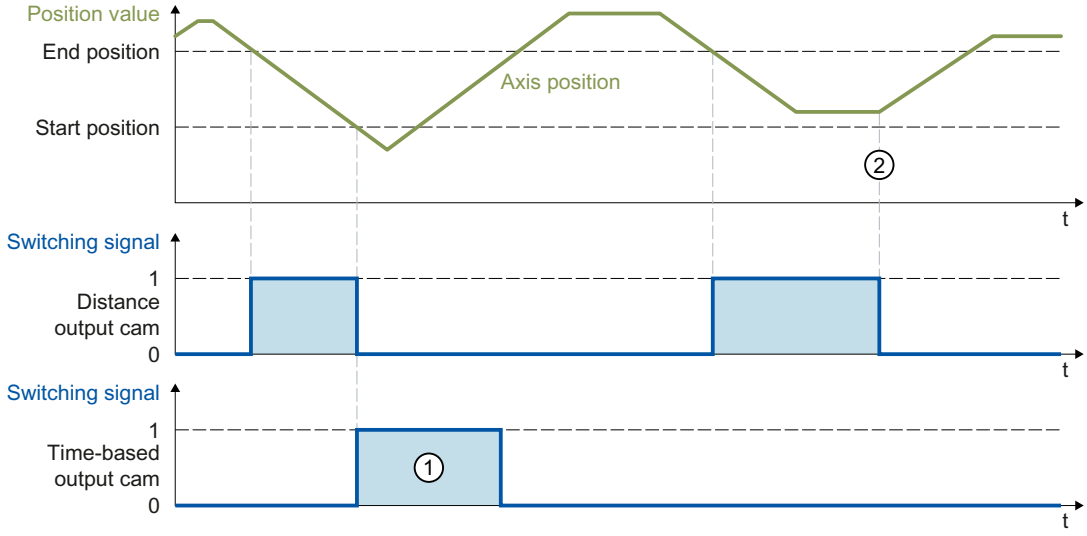

Distance output cam

The output cam switches on when the switch-on range is reached in the **negative** direction. At a direction reversal ②, the output cam switches off.

If the position value is moved into the switch-on range of the output cam, the output cam switches on when there is negative motion direction of the interconnected technology object. The output cam remains switched off when the interconnected technology object is at a standstill.

Time-based output cam

The output cam switches on when the start position is reached in the **negative** direction. At a direction reversal, the output cam remains switched on for the specified on-load factor ①.

If during the homing motion, the position value of the interconnected technology object is directly on or behind the start position of the output cam, the output cam does not switch on.

Example of "both directions" effective direction ("Direction" = 3)

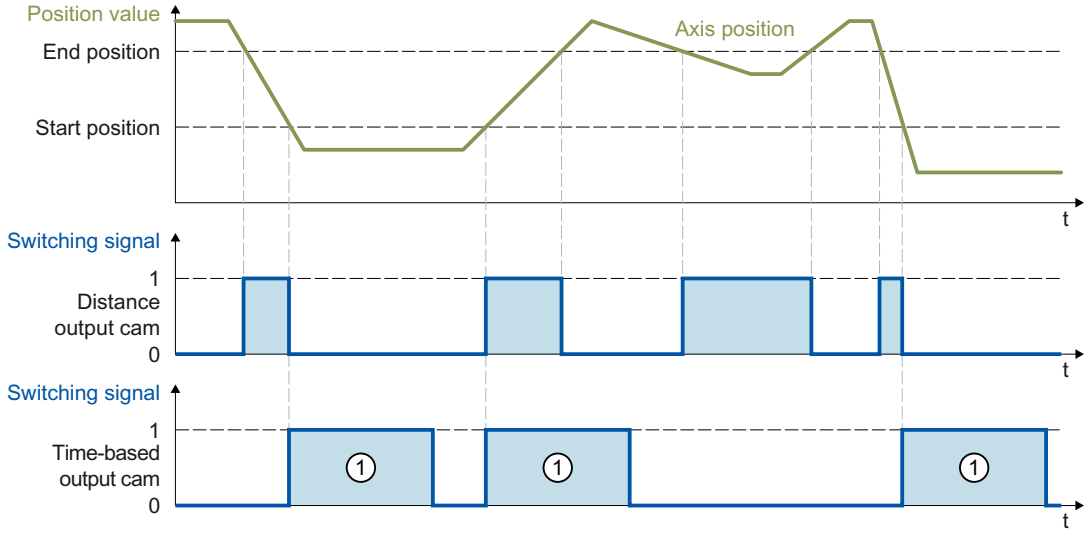

Distance output cam

The output cam switches on when the position of the interconnected technology object is within the switch-on range.

If the position value of the interconnected technology object is moved into the switch-on range of the output cam, the cam switches on even when the interconnected technology object is at a standstill.

Time-based output cam

The output cam switches on when the start position is reached. At a direction reversal, the output cam remains switched on for the specified on-load factor ①.

If during the homing motion, the position value of the interconnected technology object is directly on or behind the start position of the output cam, the output cam does not switch on.

Hysteresis (S7-1500)

Possible variations in the actual position/position setpoint can result in unwanted switch-on and switch-off of cams.

Minimal changes of the actual value of an axis in standstill can result in the switching on or off of an actual value output cam with preset positive or negative effective direction. Even minimal changes of the setpoints of a switched-off axis in follow-up mode can result in switch-on or switch-off of an actual value output cam with specified positive or negative effective direction. Such unwanted switching states can be prevented by configuring a hysteresis (> 0.0). The configuration of a hysteresis value (> 0.0) is recommended in particular with reference to the actual position.

The hysteresis is a position tolerance within which the position values may vary without changing the switching state of the output cam. Changes of direction detected within the hysteresis are ignored.

The hysteresis is set for the technology object in "<TO>.Parameter.Hysteresis".

Behavior

- The hysteresis is activated at a direction reversal.
- The following applies within the hysteresis:
  - The switching state of distance output cams is not changed.
  - The motion direction is not determined again.
  - If the start position of a time-based output cam is within the hysteresis, the time-based output cam is switched on when leaving the hysteresis with the corresponding effective direction.
  - The on-load factor of time-based output cams remains unchanged.
- After the hysteresis range is exited, distance output cams are set according to the output cam settings.

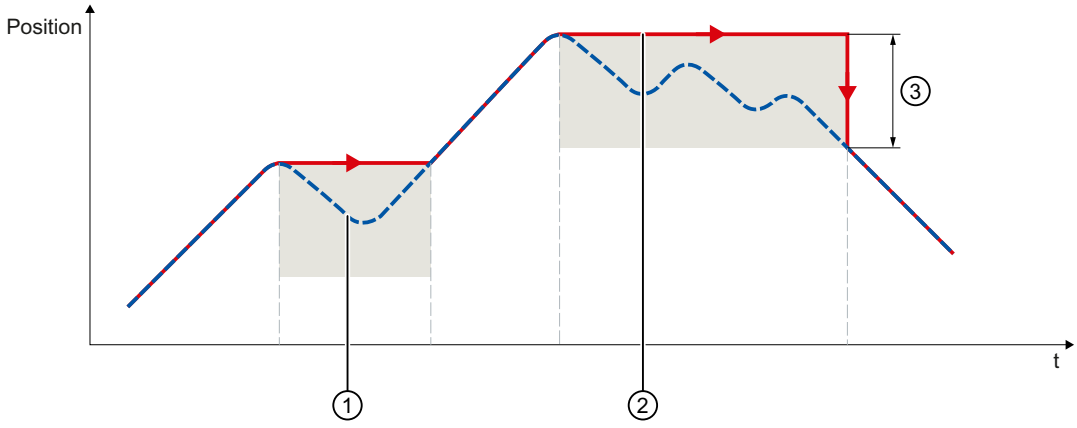

① Actual position
② Effective position
③ Hysteresis range

The following examples show the effects of the hysteresis on the switching behavior of output cams with positive activation direction.

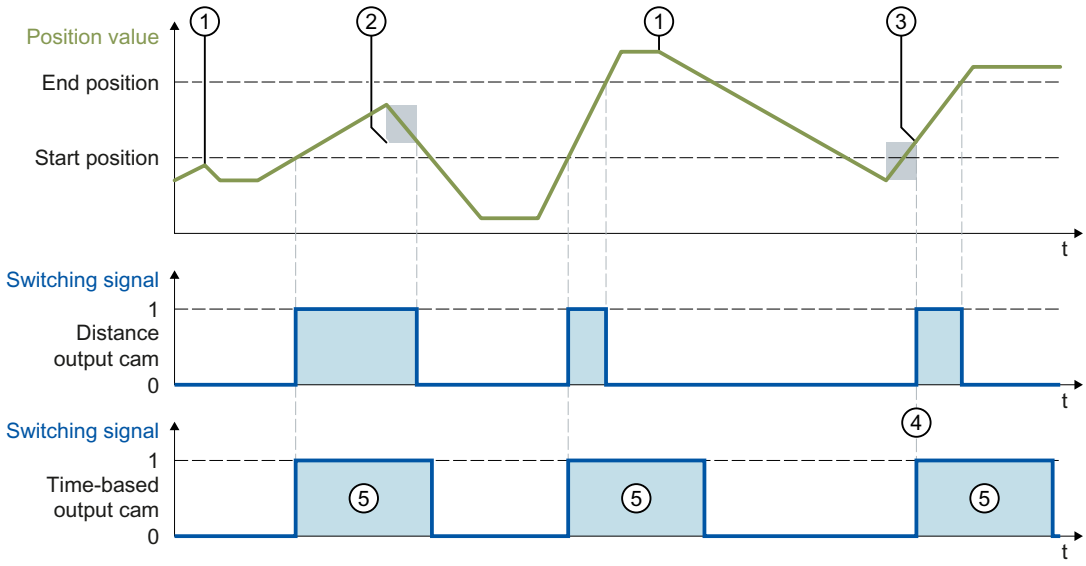

① Direction reversal without hysteresis effect

② Hysteresis in effect

③ The switch-on position of the distance output cam is influenced according to the direction reversal and hysteresis.

④ The start position of the time-based output cam is located within the hysteresis. The time-based output cam is switched on when leaving the hysteresis with the corresponding effective direction.

⑤ Switch-on duration

Hysteresis range

The maximum size of the hysteresis range in the system is as follows: for an axis with modulo function, one quarter of the modulo range; for an axis without modulo function, one quarter of the operating range.

Compensation of actuator switching times (S7-1500)

Switching times of the output and the connected actuator (e.g. valve) can be compensated for using the activation time or deactivation time of the output cam technology object.

The activation time is specified as the lead time for the switch-on edge, and the deactivation time as the lead time for the switch-off edge.

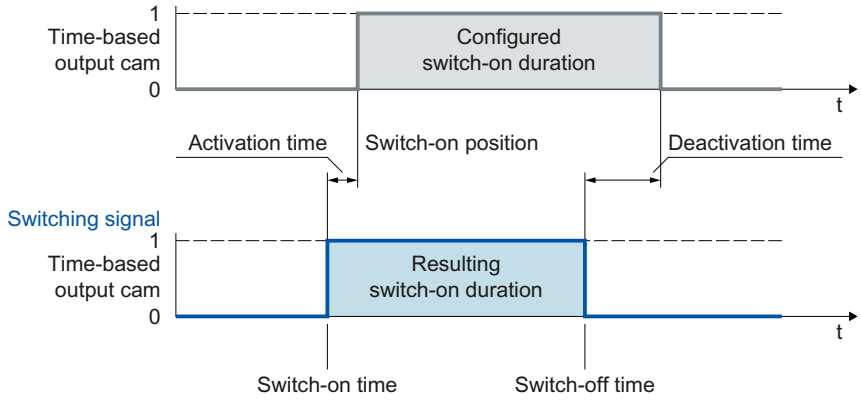

The activation time is set in the technology object via the "<TO>.Parameter.OnCompensation" tag.

The deactivation time is set at the technology object via the "<TO>.Parameter.OffCompensation" tag.

Note

- The following must be met for the times:
  On-load factor > deactivation time - activation time
  (The switch-on duration for distance output cams is determined by the switching positions and the current velocity.)
- If the output cam has been switched taking the lead times into consideration, this action has taken place from the system perspective and will not be restarted in response to any subsequent variation in the actual velocity.
- System-dependent runtimes are automatically taken into consideration when output by TM Timer DIDQ is used.
  When the output with a digital output module is used, system-dependent runtimes are not taken into consideration and must be set as correction times in the activation time or deactivation time.

Tags (S7-1500)

The following technology object tags are relevant:

Status indicator		
Tag	**Description**	
<TO>.CamOutput	The output cam is switched.	

Parameters		
Tag	**Description**	
<TO>.Parameter.OutputCamType	Output cam type	
	0	Distance output cam
	1	Time-based output cam

Parameters		
Tag	Description	
<TO>.Parameter.PositionType	Position reference	
	0	Position setpoint
	1	Actual position
<TO>.ParameterOnCompensation	Activation time (lead time for the switch-on edge)	
<TO>.Parameter.OffCompensation	Deactivation time (lead time for the switch-off edge)	
<TO>.Parameter.Hysteresis	Hysteresis value	

interface		
Tag	Description	
<TO>.Interface.EnableOutput	Activation of the output cam output	
	FALSE	No output
	TRUE	Output
<TO>.Interface.Address	I/O address of the output cam	
<TO>.Interface.LogicOperation	Logical operation of the output cam signals at the output	
	0	OR logic operation
	1	AND logic operation

Units	
Tag	Description
<TO>.Units.LengthUnit	Unit of the length data
<TO>.Units.TimeUnit	Unit of the time data

StatusWord	
Tag	Description
<TO>.StatusWord.X0 (Control)	The technology object is in operation.
<TO>.StatusWord.X1 (Error)	An error occurred at the technology object.
<TO>.StatusWord.X2 (RestartActive)	The technology object is being reinitialized. The tags of the technology data block are not updated with active restart.
<TO>.StatusWord.X3 (OnlineStartValue-sChanged)	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object.
<TO>.StatusWord.X4 (OutputInverted)	The output cam output is inverted.
<TO>.StatusWord.X5 (CommunicationOk)	The cam is synchronized with the output module and available for use.

ErrorWord	
Tag	Description
<TO>.ErrorWord.X0 (SystemFault)	A system-internal error has occurred.
<TO>.ErrorWord.X1 (ConfigFault)	Configuration error One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program.

ErrorWord	
Tag	**Description**
<TO>.ErrorWord.X2 (UserFault)	Error in user program at a Motion Control instruction or its use
<TO>.ErrorWord.X3 (CommandNotAccepted)	Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
<TO>.ErrorWord.X13 (PeripheralError)	Error accessing a logical address

ErrorDetail		
Tag	**Description**	
<TO>.ErrorDetail.Number	Alarm number You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.	
<TO>.ErrorDetail.Reaction	Effective alarm reaction	
	0	No reaction
	6	Output cam processing is complete.
	You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.	

Cam track technology object (S7-1500)

The cam track technology object generates a switching signal sequence dependent on the position of an axis or external encoder. A cam track can consist of up to 32 individual output cams and be output to one output. The switching states can be evaluated in the user program or fed to digital outputs.

You can find an overview of the functions of the cam track technology object in the Functions (Page 1626) section.

The figure below shows the basic principle of operation of the cam track technology object:

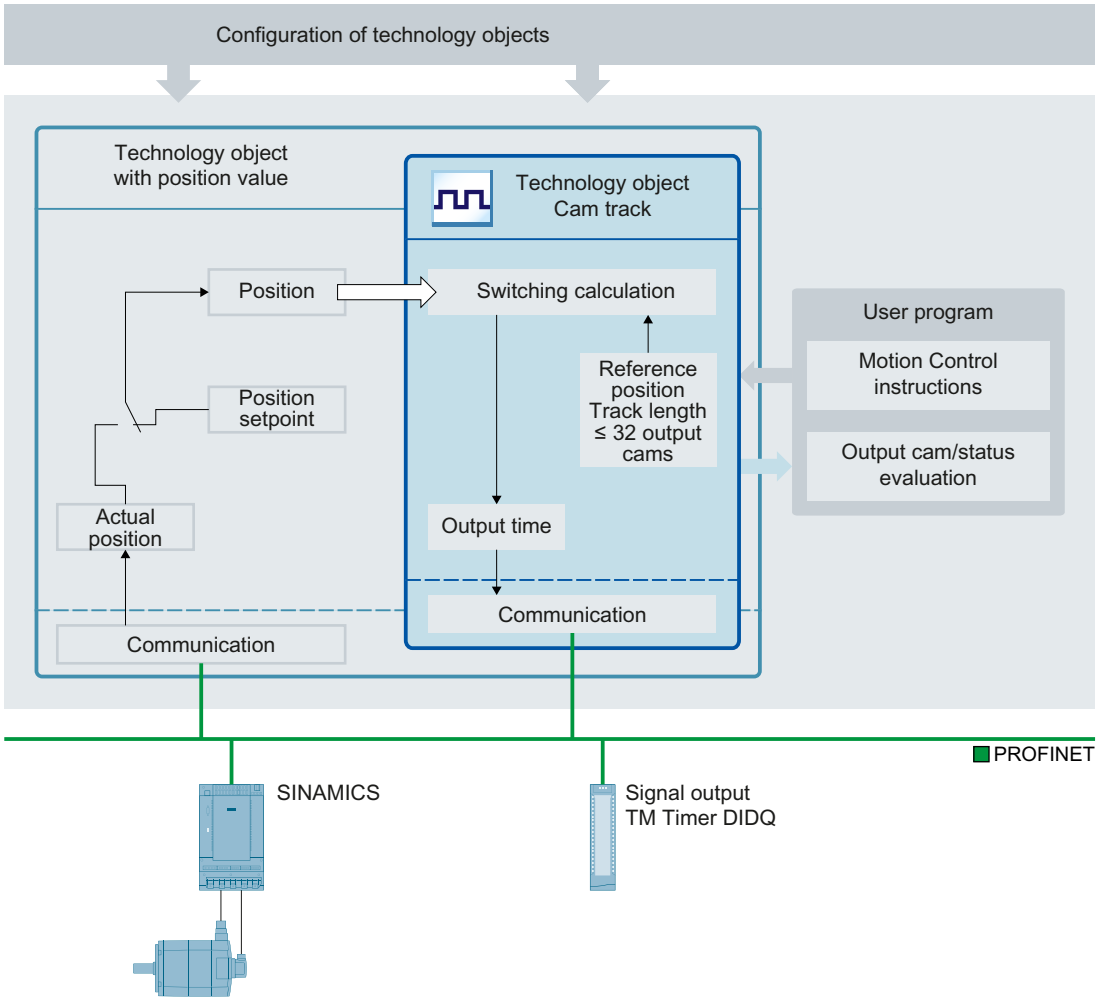

Definition of cam track

A cam track consists of up to 32 individual output cams that are specified within an adjustable track length.

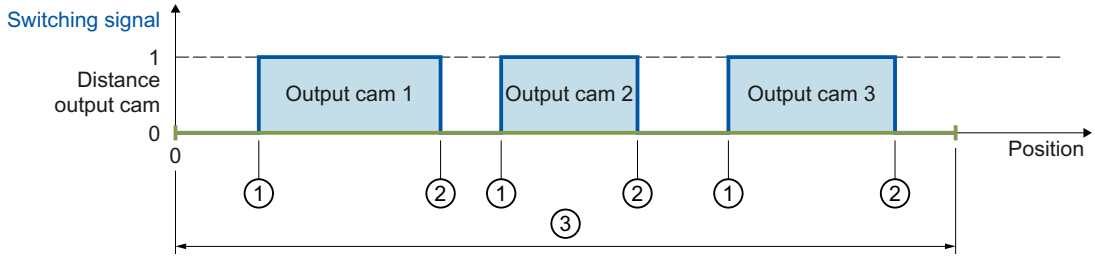

① Start position
② End position
③ Cam track length

The output cam positions are defined relative to the cam track. The start of the cam track is always 0.0. The output cam positions on the cam track are thus always positive.

When the cam track is processed, the output cams with start position within the track length are taken into consideration.

The output cams of the cam track can be set as distance output cams or time-based output cams, whereby only one of the two output cam types can be used in a cam track.

If the end of a cam track is crossed, connected output cams behave as follows within the cam track:

- Time-based output cams remain switched on for the set on-load factor.
- Distance output cams whose end position lies outside the cam track are switched off when the cam track is left.

Cams whose start and end positions lie outside the cam track length are ignored. They become active only if the cam track length is increased so that at least the respective start position of an output cam is within the new track length.

Assignment

The cam track technology object must always be assigned to another technology object whose position is evaluated.

The cam track technology object can be assigned to the following technology objects:

- Synchronous axis
- Positioning axis
- External encoder

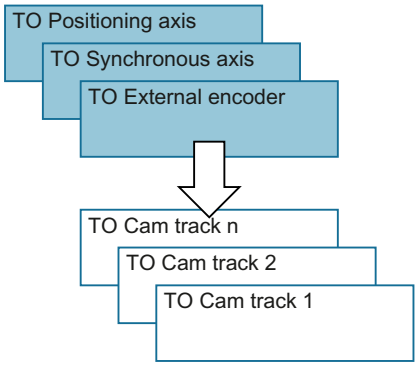

Exactly one axis or one external encoder can be assigned to a cam track.

Multiple cam tracks can be assigned to one axis or external encoder.

Position reference

The switching points of the output cams of a cam track can be referenced to the following positions, depending on the interconnected technology object.

- Actual position of a synchronous axis/positioning axis
- Position setpoint of a synchronous axis/positioning axis
- Position of an external encoder

Mapping of the cam track to the position of the technology object.

The start of the cam track is placed at the specified reference position of the interconnected technology object. Thus, the switching positions result from the cam track positions mapped onto the interconnected technology object starting from the reference position. The cam track is continued in both directions of the interconnected technology object.

The setting for the reference position can be either positive or negative.

Example

- Axis range = -1000 mm to +1000 mm
- Desired switching points of the output cam with reference to axis position:
  - Start position = -200 mm
  - End position = -100 mm
- Cam track length = 2000 mm
- Definition of output cam on the track:
  - Start position = 800 mm
  - End position = 900 mm

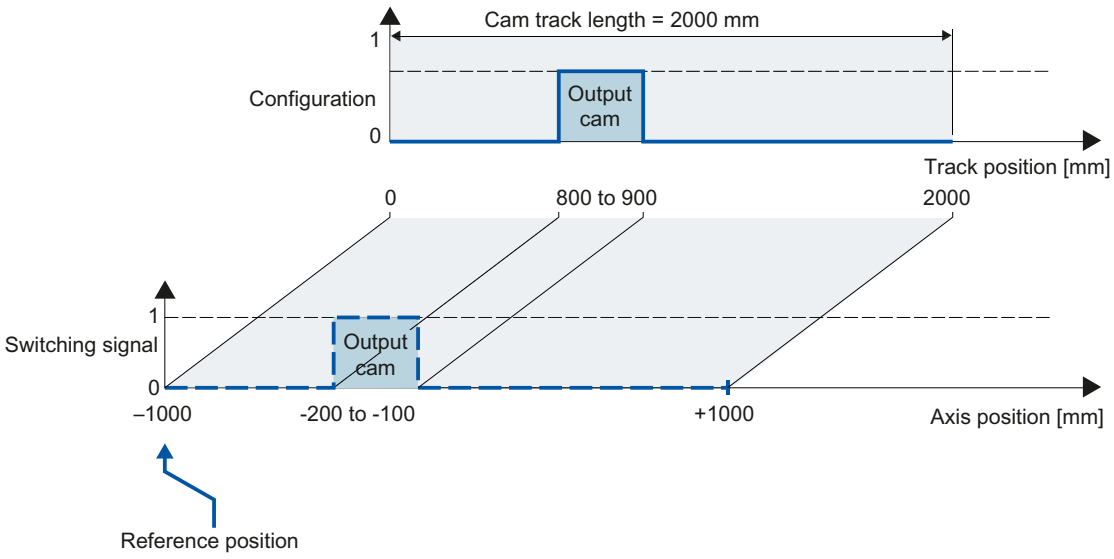

Processing of a cam track

The processing of a cam track occurs cyclically.

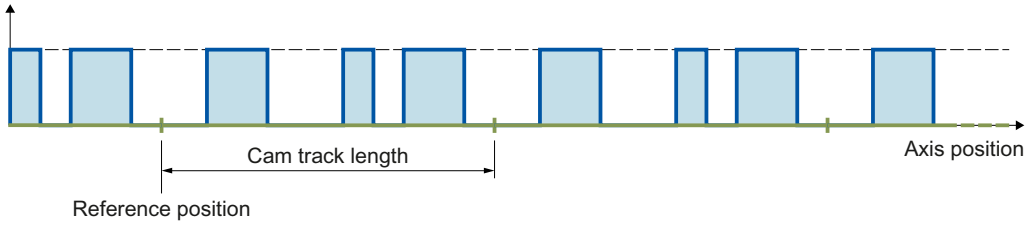

The cam track is mapped onto the position of the interconnected technology object starting from the reference position and is continued cyclically in both directions.

Output of a cam track

The following output options are available for the digital cam track output:

- TM Timer DIDQ
  Digital output with high degree of accuracy and reproducibility in the microsecond range on ET 200MP TM Timer DIDQ 16x24V and ET 200SP TM Timer DIDQ 10x24V time-based IO modules
  Output by TM Timer DIDQ requires isochronous mode. Isochronous mode is only possible with PROFINET interface modules.
- Digital output module
  Digital output with switching accuracy depending on the output cycle of the I/O used

A maximum of two edges (by TM Timer DIDQ, positive and negative) or one edge (via digital output module, positive or negative) can be output per position control cycle clock. If multiple switch-on edges or switch-off edges are transmitted in one position control cycle clock, the last written values in each case are valid.

Masking of individual output cams of a cam track

In order for output cams to processed, they must be configured as valid in the technology data block with "<TO>.Parameter.Cam[1 … 32].Existent" = TRUE. In addition, output cams of a cam track configured as valid can be defined as valid in the user program using bit masking ("<TO>.Parameter.CamMasking"). In the default setting, all valid output cams are enabled ("<TO>.Parameter.CamMasking" = 0xFFFFFFFF). The cam track itself is activated/deactivated using the Motion Control instruction "MC_CamTrack".

Modulo function (S7-1500)

Track length and mapping to an axis or encoder with modulo function

When a cam track is mapped onto an axis with modulo function, a reference position specified outside the modulo range is mapped within the modulo range.

The track length can be less than or greater than the modulo length of the axis. In order for the cam track to be mapped without offset in the modulo range and to prevent unwanted overrides, an integer ratio of modulo length to track length, and vice versa, is required.

Example Mapping without offset

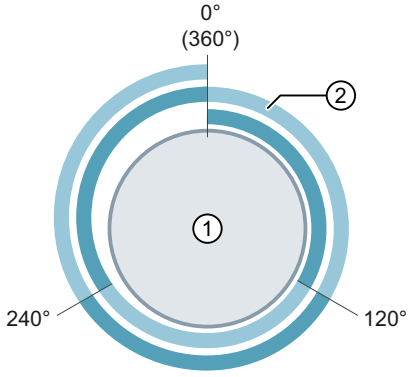

① Axis with modulo length 360°

② Cam track with track length 120°

⇒ Ratio = 360° / 120° = **3**

The cam track is output 3 times on a modulo length.

Example Mapping with offset

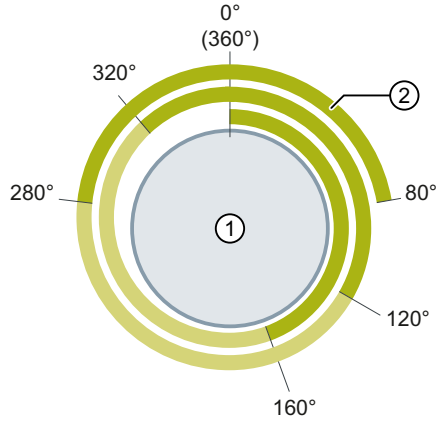

① Axis with modulo length 360°

② Cam track with track length 160°

⇒ Ratio = 360° / 160° = **2.25**

The cam track is output 2.25 times on the first modulo length and continued correspondingly in the other modulo lengths.

During cyclic processing of the cam track, the continued reference position of the current cam track is displayed in the "<TO>.MatchPosition" tag. The continued reference position is independent of direction and always the position of the left boundary of the cam track. The unique detection and output of the position is only possible when the assigned technology object is in motion. The distance to the current reference position of the current cam track (<TO>.MatchPosition) is displayed in the "<TO>.TrackPosition" tag.

When the cam track is enabled by a "MC_CamTrack" job, the position of the cam track (<TO_CamTrack>.MatchPosition) is set as if the current position had been reached by a motion in the positive direction starting from the reference position (<TO_CamTrack>.ReferencePosition).

Homing of an axis or an encoder without modulo function

A change to the position of an axis or external encoder using Motion Control instruction "MC_Home" is regarded as a sudden position change. An enabled cam track is referenced to the changed position and processed further from there.

- Homing also has an effect on the current position of the cam track (<TO>.TrackPosition). The position is formed again as quickly as possible due to the offset.
- Distance output cams are either skipped or correspondingly output.
- Time-based output cams are skipped. A time-based output cam is switched on only when the start position is overtraveled and remains switched on for the switch-on duration.
- Switched time-based output cams are not canceled by a homing operation.

Recommendation:

Disable the cam track before or during homing.

Homing of an axis or an encoder with modulo function

- Homing also has an effect on the current position of the cam track (<TO>.TrackPosition). The homing of the assigned technology object influences the position of the subsequently active cam track on a direction-dependent basis. This is dependent on the position difference, the difference between the new position minus the original position. If the position difference is negative, you add the modulo length (<TO_Axis/TO_Encoder>.Modulo.Length).
  If this position difference is less than or equal to half the modulo length, the new position of the cam track (<TO_CamTrack>.MatchPosition) is set as if the new position had been reached by a motion in the positive direction starting from the original position.
  If this position difference is greater than half the modulo length, the new position of the cam track (<TO_CamTrack>.MatchPosition) is set as if the new position had been reached by a motion in the negative direction starting from the original position.
- Distance output cams are either skipped or correspondingly output.
- Time-based output cams are skipped. A time-based output cam is switched on only when the start position is overtraveled and remains switched on for the switch-on duration.
- Switched time-based output cams are not canceled by a homing operation.

Recommendation:

Disable the cam track before or during homing.

Effective direction (S7-1500)

The cam track is always active for both directions of the position of the interconnected technology object.

Output of a cam track with distance output cams

Distance output cams are switched when the switch-on range is overtraveled.

The following graphic shows the execution of a cam track with distance output cam depending on the motion direction of the axis.

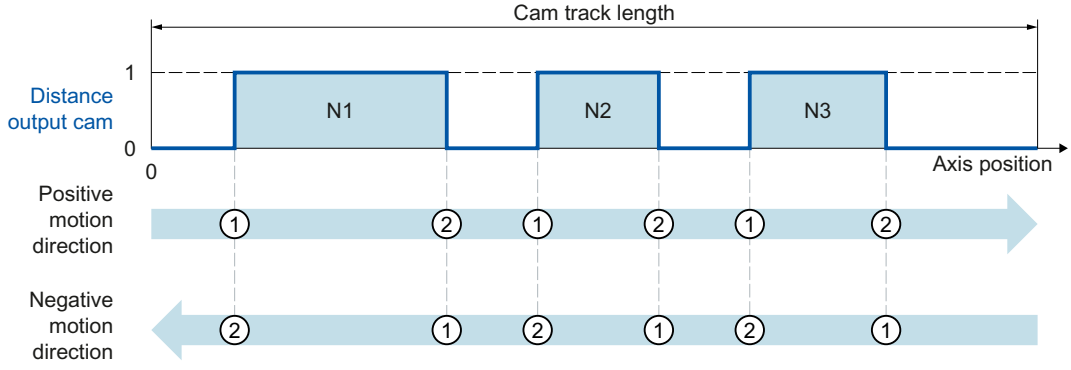

With positive motion direction, the output cams of the cam track are output in the order Output Cam N1, Output Cam N2, Output Cam N3. In the case of negative motion direction, the output cams of the cam track are output in the order Output Cam N3, Output Cam N2, Output Cam N1. The distance output cams switch on at ① and switch off at ②.

Output of a cam track with time-based output cams

The time-based output cams are switched when the start position is crossed.

The following graphic shows the execution of a cam track with time-based output cam depending on the motion direction of the axis.

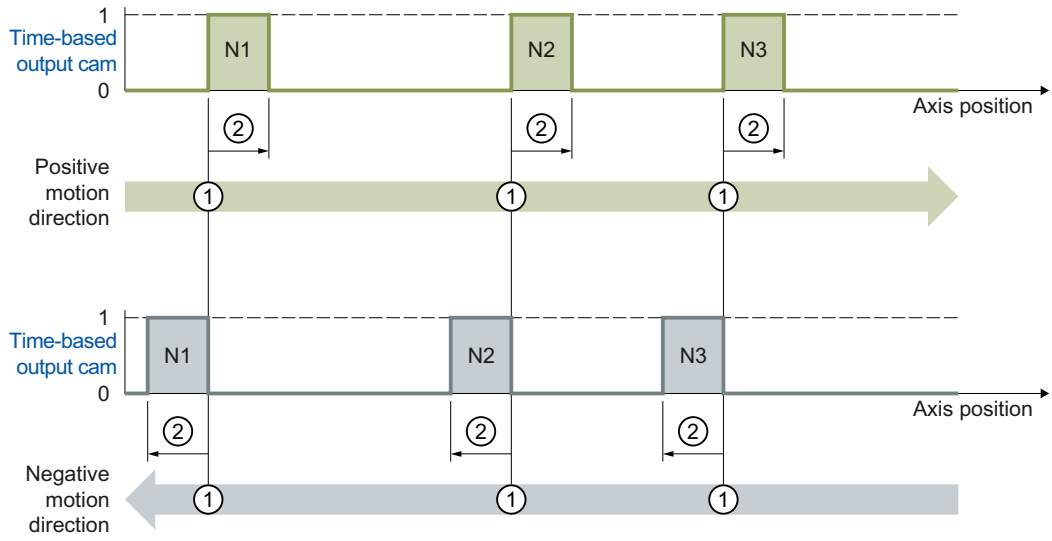

With positive motion direction, the output cams of the cam track are output in the order Output Cam N1, Output Cam N2, Output Cam N3. In the case of negative motion direction, the output cams of the cam track are output in the order Output Cam N3, Output Cam N2, Output Cam N1. The time-based output cams switch on at ① and remain switched on for the set switch-on duration ②.

Changing the cam track data during operation (S7-1500)

The data of a cam track and the parameters of the associated Motion Control instruction "MC_CamTrack" can be changed while track processing is enabled. The active Motion Control instruction "MC_CamTrack" is not aborted. The modified parameters, however, only take effect at the next call of the Motion Control instruction "MC_CamTrack".

The following parameters can be changed during operation and are in effect after another call of Motion Control instruction "MC_CamTrack".

- Cam track data in technology data block
  - Reference position (<TO>.Parameter.ReferencePosition)
  - Track length (<TO>.Parameter.CamTrackLength)
  - Bit masking of individual output cams (<TO>.Parameter.CamMasking)
  - Activation time (<TO>.Parameter.OnCompensation)
  - Deactivation time (<TO>.Parameter.OffCompensation)
  - Hysteresis value (<TO>.Parameter.Hysteresis)
  - Output cam data (<TO>.Parameter.Cam[1 … 32])
- Parameters in the function block
  - Enable (MC_CamTrack.Enable)
  - Mode (MC_CamTrack.Mode)
  - Inverted output (MC_CamTrack.InvertOutput)

Note the different activation behavior (Page 1756) when changing the cam track data.

Activation behavior (S7-1500)

A cam track is activated by the call of Motion Control instruction "MC_CamTrack" with "Enable" = TRUE. A distinction must be made here between:

- First-time activation of the cam track
- Call after a change of the cam track data during active cam track processing

The difference relates to how the cam track data is applied. Depending on the set mode ("MC_CamTrack.Mode"), the configuration (cam track data, data in the function block) is applied at different times.

- **First-time switch-on of a cam track**
  Calling the Motion Control instruction "MC_CamTrack" with "Enable" = TRUE activates the cam track immediately ("<TO>.Status" changes to 1) and configured cam track data takes effect immediately. This behavior is the same when "MC_CamTrack.Mode" = 0 and "MC_CamTrack.Mode" = 1.
- **Change of cam track data of an already activated cam track** ("<TO>.Status" = 1)
  - With the call of Motion Control instruction "MC_CamTrack" with "Enable" = TRUE and "Mode" = 0, the modified cam track data takes effect immediately.
    Previously activated distance output cams are aborted if their track signals are not still set due to the changed cam track data. Previously activated time-based output cams are always aborted.
  - With the call of Motion Control instruction "MC_CamTrack" with "Enable" = TRUE and "Mode" = 1, the cam track continues to be output with the prior configuration up to the cam track end. Modified cam track data takes effect at the end of the current track cycle.
    If you change a cam track with "MC_CamTrack.Mode" = 1 during runtime of the user program, keep in mind the lead time of the cam track as reserve for the first output cam.
    Define the first output cam position in the cam track only after the following position:
    Position of first output cam > velocity of axis x lead time of the cam track (<TO>.Parameter.OnCompensation)
    Also keep in mind the internal system time for output cam calculation, even if you set the lead time 0.0.

Changing cam track data when modulo length is not a multiple of the track length

For the switching times to be set correctly, we recommend the following procedure for changes:

- Enter changes as soon as possible after start of a new cam track
- Enter a new reference position (<TO>.Parameter.ReferencePosition) for the changed cam track. The new reference position is composed as follows depending on the direction:
  - Positive effective direction: Current reference position (<TO>.MatchPosition) + cam track length (<TO>.Parameter.CamTrackLength)
  - Negative effective direction: Current reference position (<TO>.MatchPosition) - cam track length (<TO>.Parameter.CamTrackLength)
- Output the changes of the cam track when calling the Motion Control instruction "MC_CamTrack" and "Mode" = 1.

Example

The following figure shows an example of the differences in the activation behavior.

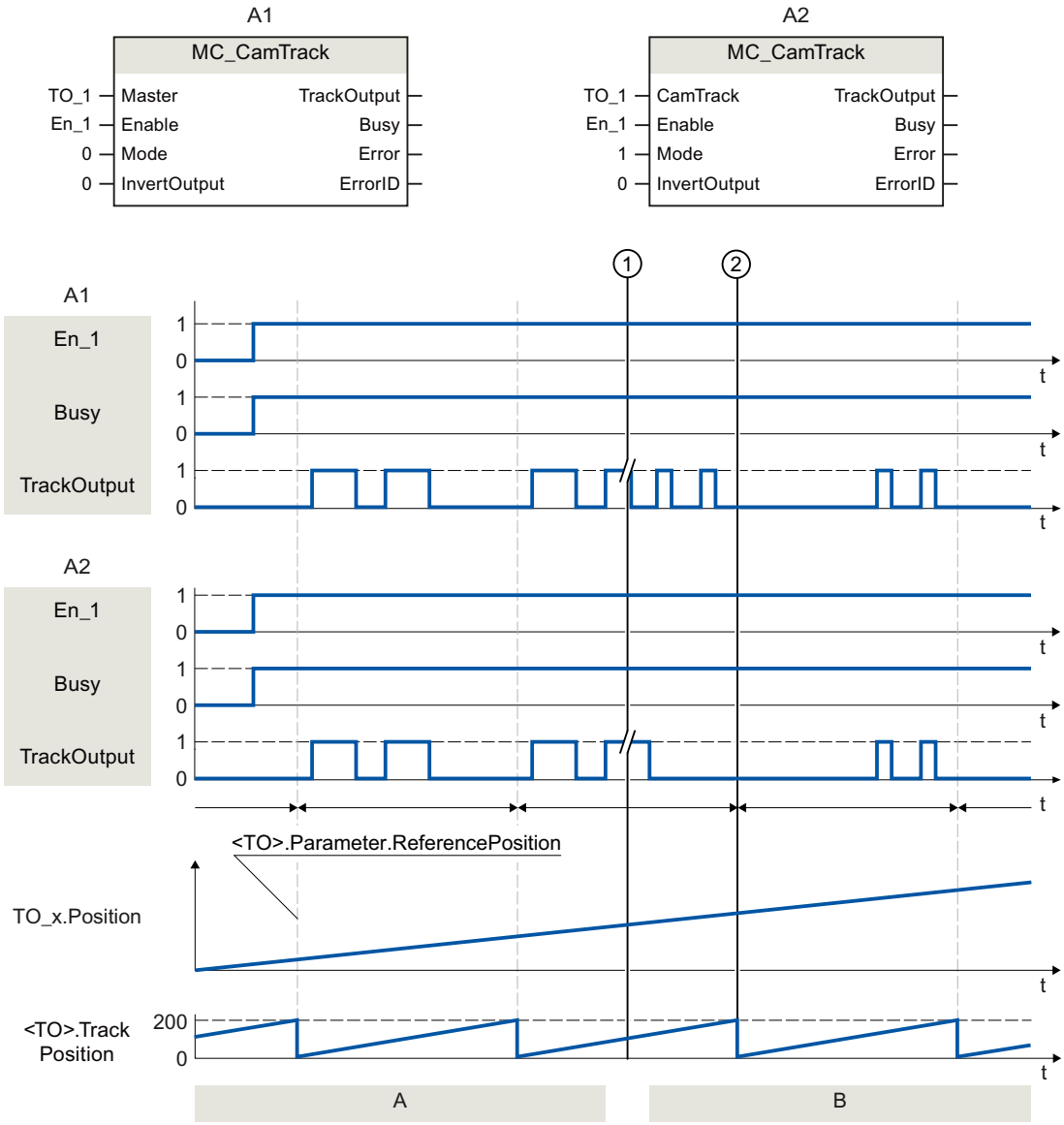

A1	A	The cam track is activated the first time with "MC_CamTrack.Enable" = TRUE and the output cams are output immediately with set "MC_CamTrack.Mode" = 0.
	B	After cam track data were changed (①), the cam track is activated by calling Motion Control instruction "MC_CamTrack" with "Enable" = TRUE and the modified data takes effect immediately (①) with set "MC_CamTrack.Mode" = 0.
A2	A	The cam track is activated the first time with "MC_CamTrack.Enable" = TRUE and the output cams are output immediately with set "MC_CamTrack.Mode" = 1.
	B	After cam track data were changed (①), the cam track is activated by calling Motion Control instruction "MC_CamTrack" with "Enable" = TRUE and the modified data takes effect at the end of the current track cycle (②) with set "MC_CamTrack.Mode" = 1.

Hysteresis (S7-1500)

The hysteresis is set in the cam track technology object. The behavior and effect of the hysteresis setting corresponds to the hysteresis (Page 1742) for the cam technology object.

Time offset of cam switching points (S7-1500)

Switching times of the output and the connected actuator (e.g. valve) can be compensated for using the activation time or deactivation time of the cam track technology object.

The time offset of cam switching points corresponds to the activation time or deactivation time (Page 1744) for the cam technology object.

Tags (S7-1500)

The following technology object tags are relevant:

Status indicator		
Tag	**Description**	
<TO>.Status	Status of the cam track function	
	0	Inactive
	1	Active
	2	Active and waiting for next track
<TO>.TrackOutput	A cam of cam track is switched.	
<TO>.SingleCamState	Switched on output cam (bit-masked)	
<TO>.TrackPosition	Display of the current position within the cam track The distance to the current reference position of the current cam track (<TO>.MatchPosition) is displayed.	
<TO>.MatchPosition	Reference position of the current cam track During cyclic processing of the cam track, the continued reference position of the current cam track is displayed. The unique detection and output of the position is only possible when the assigned technology object is in motion.	

Parameters		
Tag	**Description**	
<TO>.Parameter.CamTrackType	Output cam type	
	0	Distance output cam
	1	Time-based output cam
<TO>.Parameter.PositionType	Position reference	
	0	Position setpoint
	1	Actual position
<TO>.Parameter.ReferencePosition	Reference position	
<TO>.Parameter.CamTrackLength	Track length	
<TO>.Parameter.CamMasking	Bit masking of individual output cams	
<TO>.Parameter.OnCompensation	Activation time (lead time for the switch-on edge)	
<TO>.Parameter.OffCompensation	Deactivation time (lead time for the switch-off edge)	

Parameters		
Tag	**Description**	
<TO>.Parameter.Hysteresis	Hysteresis value	
<TO>.Parameter.Cam[1..32].OnPosition	Start position (distance output cams and time-based output cams)	
<TO>.Parameter.Cam[1..32].OffPosition	End position (distance output cam)	
<TO>.Parameter.Cam[1..32].Duration	Switch-on duration (time-based output cam)	
<TO>.Parameter.Cam[1..32].Existent	Validity of an output cam	
	FALSE	Output cam is not used.
	TRUE	Output cam is used

interface		
Tag	**Description**	
<TO>.Interface.EnableOutput	Output cam output at the bit specified under "Address"	
	FALSE	No output
	TRUE	Output
<TO>.Interface.Address	I/O address for digital cam output	

Units	
Tag	**Description**
<TO>.Units.LengthUnit	Unit of the length data
<TO>.Units.TimeUnit	Unit of the time data

StatusWord	
Tag	**Description**
<TO>.StatusWord.X0 (Control)	The technology object is in operation.
<TO>.StatusWord.X1 (Error)	An error occurred at the technology object.
<TO>.StatusWord.X2 (RestartActive)	The technology object is being reinitialized. The tags of the technology data block are not updated with active restart.
<TO>.StatusWord.X3 (OnlineStartValue-sChanged)	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object.
<TO>.StatusWord.X4 (OutputInverted)	The output cam output is inverted.
<TO>.StatusWord.X5 (CommunicationOk)	The cam track is synchronized with the output module and available for use.
<TO>.StatusWord.X6 (CamDataChanged)	The data of individual output cams has been changed but not yet taken effect with Motion Control instruction "MC_CamTrack".

ErrorWord	
Tag	**Description**
<TO>.ErrorWord.X0 (SystemFault)	A system-internal error has occurred.
<TO>.ErrorWord.X1 (ConfigFault)	Configuration error One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program.
<TO>.ErrorWord.X2 (UserFault)	Error in user program at a Motion Control instruction or its use.
<TO>.ErrorWord.X3 (CommandNotAccepted)	Command cannot be executed. A Motion Control instruction cannot be executed because necessary requirements have not been met.
<TO>.ErrorWord.X13 (PeripheralError)	Error accessing a logical address.

ErrorDetail		
Tag	**Description**	
<TO>.ErrorDetail.Number	Alarm number	
<TO>.ErrorDetail.Reaction	Effective alarm reaction	
	0	No reaction
	5	Cam track processing Is complete.

Cam technology object

The cam technology object defines a transfer function y = f(x). The dependency of an output value on an input value is described in this transfer function in a unit-neutral manner. A cam technology object can be used multiple times.

You can find an overview of the functions of the cam technology object in the Functions (Page 1626) section.

You define the function y = f(x) in the configuration of the technology object (Page 1854) using interpolation points and/or segments. Ranges between interpolation points and segments are interpolated using the Motion Control instruction "MC_InterpolateCam". The settings can be changed/redefined during runtime of the user program with the technology data block according to the appendix "Tags of the cam technology object (Page 2101)".

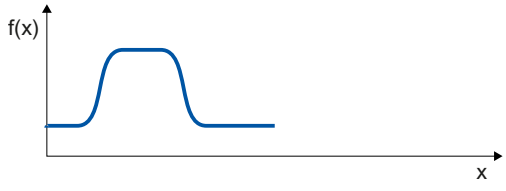

An interpolated cam can be applied as a synchronous operation function for camming (Page 1713).

The figure below shows the basic operating principle of the cam technology object:

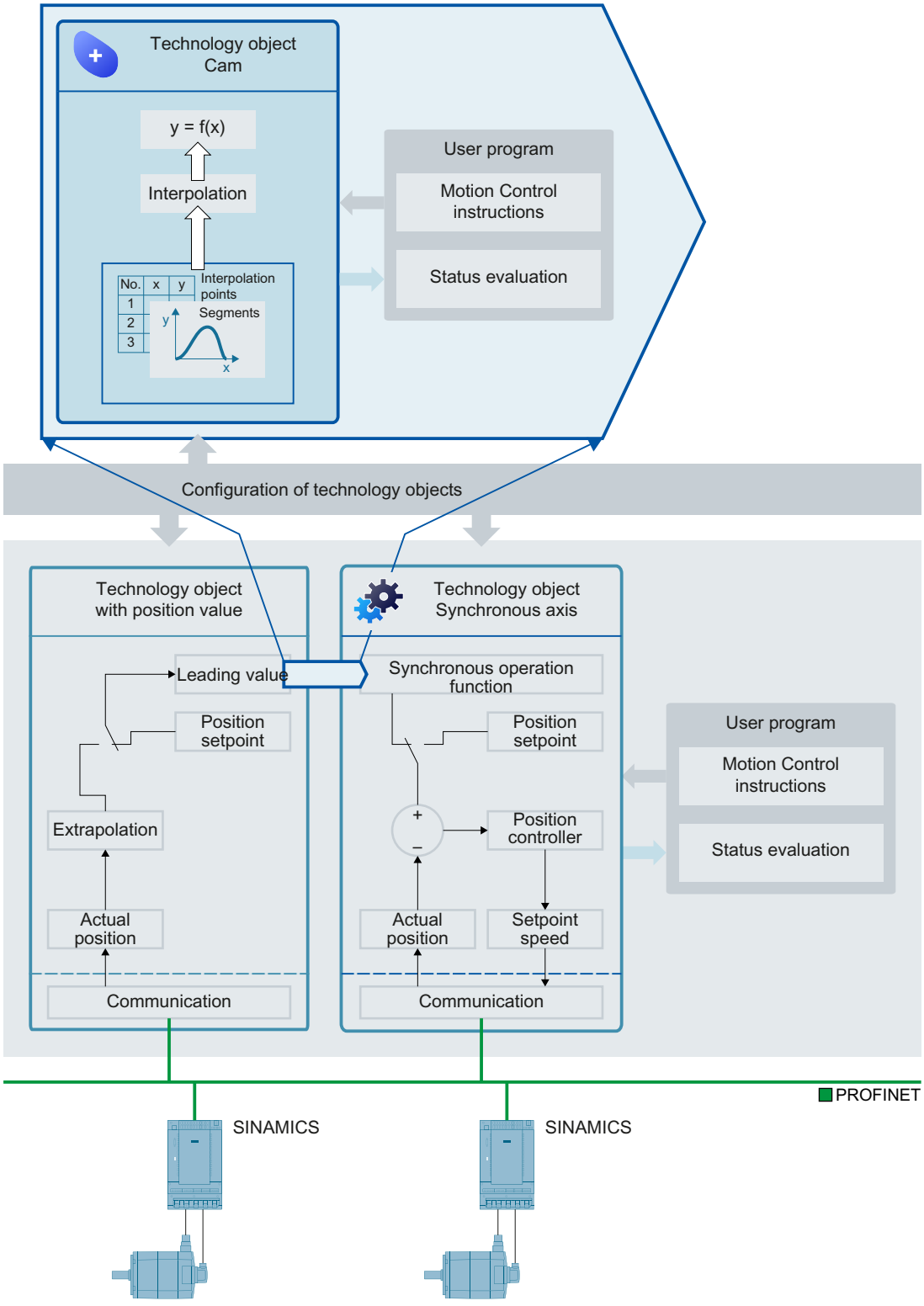
Technology object
Cam
y = f(x)
Interpolation
No.
x
y
1
2
3
Interpolation points
Segments
y
x
User program
Motion Control instructions
Status evaluation
Configuration of technology objects
Technology object with position value
Leading value
Position setpoint
Extrapolation
Actual position
Communication
Technology object Synchronous axis
Synchronous operation function
Position setpoint
+
–
Position controller
Actual position
Setpoint speed
Communication
User program
Motion Control instructions
Status evaluation
PROFINET
SINAMICS
SINAMICS

Interpolation

To use a cam in the user program, you must interpolate the cam after downloading to the CPU. The interpolation closes the gaps between the defined interpolation points and segments of the cam. The cam is interpolated from the minimum value in the leading value range to the maximum value. The minimum value in the leading value range is the first defined interpolation point/start of the first segment of the cam (<TO>.StatusCam.StartLeadingValue). The maximum value in the leading value range is the last defined interpolation point/end of the last segment of the cam (<TO>.StatusCam.EndLeadingValue).

You interpolate a cam in your user program with the Motion Control instruction "MC_InterpolateCam". When a cam is interpolated, this is indicated by the "MC_InterpolateCam.Done" parameter = TRUE and via the "<TO>.StatusWord.X5 (Interpolated)" tag = 1 in the technology data block.

You specify the interpolation in the configuration of the technology object (Page 1854). The following methods are possible:

- System interpolation
- Optimization of transitions according to VDI Guideline 2143

System interpolation

With system interpolation, the transitions are interpolated according to the interpolation type and the response in the boundary points of the transition segment.. The following interpolation methods are possible:

- **Linear interpolation**
  Gaps in the cam are closed with a straight line.

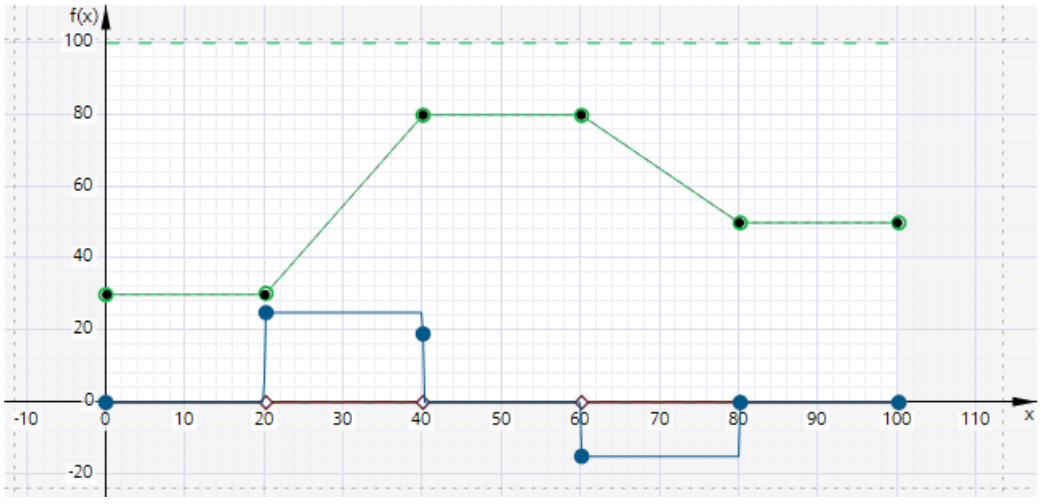

Specified position (point)
Interpolated position
Resulting velocity (scaled)
Resulting acceleration
Resulting jerk

- **Interpolation with cubic splines**
  The interpolated curve runs through the interpolation points or segments of the curve. After completion of interpolation, the range of the cam can be greater than before interpolation.

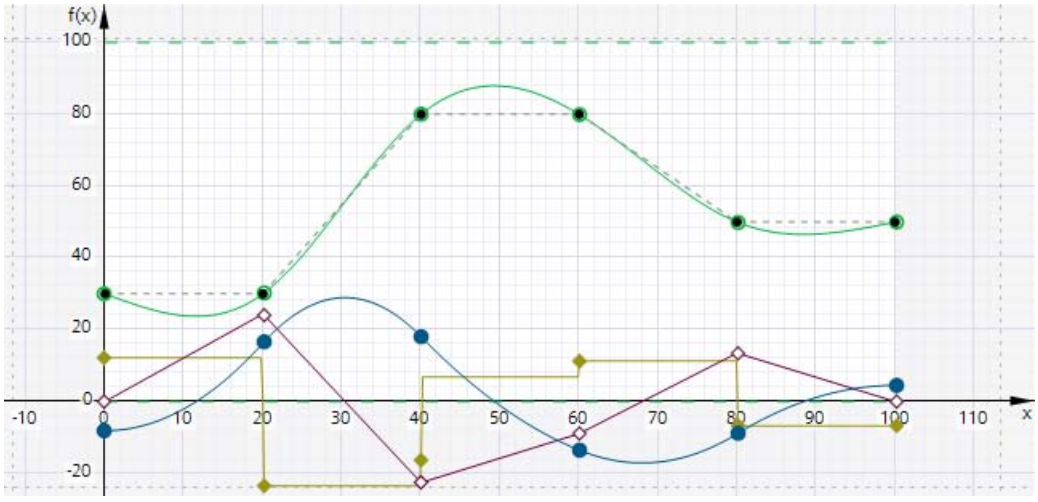

Specified position (point)
Interpolated position
Resulting velocity (scaled)
Resulting acceleration (scaled)
Resulting jerk (scaled)

- **Interpolation with Bézier splines**
  The interpolated curve runs along the interpolation points or curve segments.
  The range of the cam is not changed by interpolation.

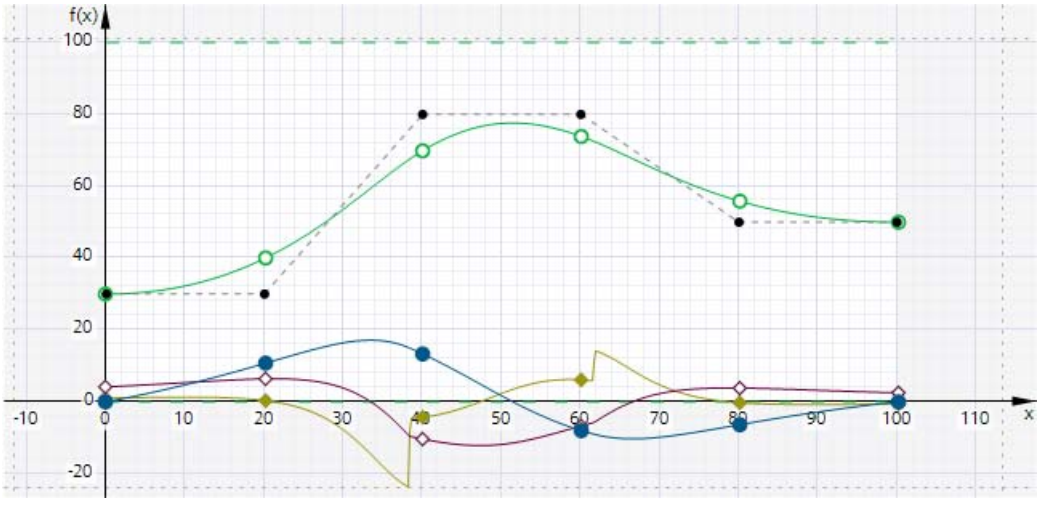

Specified position (point)
Interpolated position
Resulting velocity (scaled)
Resulting acceleration (scaled)
Resulting jerk (scaled)

Optimization of transitions according to VDI Guideline 2143

The transitions are specified according to the motion task and the optimization settings according to the VDI Guideline 2143.

Note that the optimization of transitions according to VDI guideline 2143 directly occupies segments in the technology object data block, contrary to system interpolation. This optimization type is thus not possible via "MC_InterpolateCam" during runtime.

Kinematics technology object

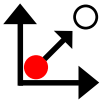

The Kinematics technology object is described in the "S7-1500T Kinematics Functions" function manual.

6.1.2.3 Using versions (S7-1500)

Overview of versions (S7-1500)

For S7-1500 Motion Control, a distinction is made between the version of the technology, the technology objects and the Motion Control instructions. The overview shown below includes S7-1500 and S7-1500T. Only one technology version can be operated on a CPU.

When changing to a CPU ≥ V1.6, you must change the technology version accordingly. Card replacement from a CPU < V1.6 to a ≥ CPU V1.6 is supported. In the TIA Portal, you can use a CPU ≥ V1.6 only to work on projects with a correspondingly higher technology version.

There are two ways of changing the technology version:

- Changing the version of the Motion Control instructions
  You change the version of the Motion Control instructions in the "Instructions" task card in the folder "Technology > Motion Control > S7-1500 Motion Control".
  If the Motion Control instruction version used does not correspond to the compatibility list, the relevant Motion Control instructions are highlighted in red in the program editor.
- Adding a technology object with an alternative version
  If a technology object with an alternative version is added in the "Add new object" dialog, the technology version is changed to the alternative version.

The technology objects and Motion Control instructions are only converted to the selected version of the technology during compilation.

The version of a technology object or a Motion Control instruction is indicated in the properties of the technology object, "General > Information" tab, "Version" field.

Compatibility list

The table below shows the compatibility of the technology version with the CPU version:

CPU	Technology	Technology object
V2.5	V4.0	Speed Axis V4.0 Positioning Axis V4.0 External Encoder V4.0 Synchronous Axis V4.0 Measuring input V4.0 Cam V4.0 Cam track V4.0 Cam V4.0 (S7-1500T) Kinematics V4.0 (S7-1500T)
V2.0, V2.1	V3.0	Speed Axis V3.0 Positioning axis V3.0 External encoder V3.0 Synchronous axis V3.0 Measuring input V3.0 Output cam V3.0 Cam track V3.0 Cam V3.0 (S7-1500T)
V1.6, V1.7, V1.8	V2.0	Speed Axis V2.0 Positioning axis V2.0 External encoder V2.0 Synchronous axis V2.0
V1.0, V1.1, V1.5[1]	V1.0	Speed Axis V1.0 Positioning axis V1.0 External encoder V1.0

1) Card replacement from a CPU < V1.6 to a CPU ≥ V1.6 is supported.

Parameter "Mode" of the Motion Control instruction "MC_Home"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter.

The table below shows a comparison of the "MC_Home.Mode" parameter for technology V1.0 and ≥ V2.0:

MC_Home.HomingMode V1.0	Parameter value	MC_Home.Mode ≥ V2.0
Direct homing (absolute)	0	Direct homing (absolute)
Direct homing (relative)	1	Direct homing (relative)
Passive homing	2	Passive homing (without reset)
Passive homing (with configured home position)	3	Active homing

MC_Home.HomingMode V1.0	Parameter value	MC_Home.Mode ≥ V2.0
Active homing	4	Reserved
Active homing (with configured home position)	5	Active homing (with configured home position)
Absolute encoder adjustment (relative)	6	Absolute encoder adjustment (relative)
Absolute encoder adjustment (absolute)	7	Absolute encoder adjustment (absolute)
Passive homing (without reset)	8	Passive homing
Abort passive homing	9	Abort passive homing
-	10	Passive homing (with configured home position)

You can find additional information about the "MC_Home.Mode" parameter in the description of the Motion Control instruction "MC_Home".

Tags of the technology object

Starting from technology version V3.0, all input and output addresses are specified using data type "VREF". This results in the following changes to the tags of the technical object:

Tag of technology object	Changes starting from V3.0
<TO>.Actor.Interface.AddressIn	Data type: VREF
<TO>.Actor.Interface.AddressOut	Data type: VREF
<TO>.Sensor[n].Interface.AddressIn	Data type: VREF
<TO>.Sensor[n].Interface.AddressOut	Data type: VREF
<TO>.Actor.Interface.EnableDriveOutputAddress	Data type: VREF
<TO>.Actor.Interface.EnableDriveOutputBitNumber	Tag eliminated
<TO>.Actor.Interface.DriveReadyInputAddress	Data type: VREF
<TO>.Actor.Interface.DriveReadyInputBitNumber	Tag eliminated
<TO>.Sensor[n].ActiveHoming.DigitalInputAddress	Data type: VREF
<TO>.Sensor[n].ActiveHoming.DigitalInputBitNumber	Tag eliminated
<TO>.Sensor[n].PassiveHoming.DigitalInputAddress	Data type: VREF
<TO>.Sensor[n].PassiveHoming.DigitalInputBitNumber	Tag eliminated
<TO>.PositionLimits_HW.MinInputAddress	Data type: VREF
<TO>.PositionLimits_HW.MinInputBitNumber	Tag eliminated
<TO>.PositionLimits_HW.MaxInputAddress	Data type: VREF
<TO>.PositionLimits_HW.MaxInputBitNumber	Tag eliminated

Version V4.0 (S7-1500)

Innovations

Technology Version V4.0 contains the following new features:

- Exchange of torque data with the drive in the technological units of the technology object.
  - Additive setpoint torque
  - Current actual torque
  - Permissible torque range
- Extension of the data structure of the positioning axis and synchronous axis for using technology object kinematics
- Use of optimized data blocks (drive/encoder connection)

Additional innovations for S7-1500T:

Technology Version V4.0 contains the following additional features:

- Kinematic technology object (S7-1500T)
- Motion specification via "MotionIn" instructions (S7-1500T)
- Direct synchronous setting with "MC_CamIn" V4.0 (S7-1500T)

Motion Control instructions

Technology Version V4.0 contains the following Motion Control instructions:

- MC_Power V4.0
- MC_Reset V4.0
- MC_Home V4.0
- MC_Halt V4.0
- MC_MoveAbsolute V4.0
- MC_MoveRelative V4.0
- MC_MoveVelocity V4.0
- MC_MoveJog V4.0
- MC_MoveSuperimposed V4.0
- MC_SetSensor V4.0 (S7-1500T)
- MC_TorqueLimit V4.0
- MC_MeasuringInput V4.0
- MC_MeasuringInputCyclic V4.0
- MC_AbortMeasuringInput V4.0
- MC_OutputCam V4.0

- MC_CamTrack V4.0
- MC_GearIn V4.0
- MC_GearInPos V4.0 (S7-1500T)
- MC_PhasingAbsolute V4.0 (S7-1500T)
- MC_PhasingRelative V4.0 (S7-1500T)
- MC_CamIn V4.0 (S7-1500T)
- MC_SynchronizedMotionSimulation V4.0 (S7-1500T)
- MC_InterpolateCam V4.0 (S7-1500T)
- MC_GetCamFollowingValue V4.0 (S7-1500T)
- MC_GetCamLeadingValue V4.0 (S7-1500T)
- MC_MotionInVelocity V4.0 (S7-1500T)
- MC_MotionInPosition V4.0 (S7-1500T)
- MC_TorqueAdditive V4.0
- MC_TorqueRange V4.0
- MC_GroupInterrupt V4.0 (S7-1500T)
- MC_GroupContinue V4.0 (S7-1500T)
- MC_GroupStop V4.0 (S7-1500T)
- MC_MoveLinearAbsolute V4.0 (S7-1500T)
- MC_MoveLinearRelative V4.0 (S7-1500T)
- MC_MoveCircularAbsolute V4.0 (S7-1500T)
- MC_MoveCircularRelative V4.0 (S7-1500T)
- MC_DefineWorkspaceZone V4.0 (S7-1500T)
- MC_DefineKinematicsZone V4.0 (S7-1500T)
- MC_SetWorkspaceZoneActive V4.0 (S7-1500T)
- MC_SetWorkspaceZoneInactive V4.0 (S7-1500T)
- MC_SetKinematicsZoneActive V4.0 (S7-1500T)
- MC_SetKinematicsZoneInactive V4.0 (S7-1500T)
- MC_DefineTool V4.0 (S7-1500T)
- MC_SetTool V4.0 (S7-1500T)
- MC_SetOcsFrame V4.0 (S7-1500T)

Version-based UDT names

The following table shows the version-based UDT names for the control words and status words of the SIEMENS telegrams 10x:

UDT name <V4.0	UDT name ≥V4.0	WORD data type
PD_STW1	PD_STW1_611Umode	Control word 1 (STW1)
PD_STW2	PD_STW2_611Umode	Control word 2 (STW2)
PD_ZSW1	PD_ZSW1_611Umode	Status word 1 (ZSW1)
PD_ZSW2	PD_ZSW2_611Umode	Status word 2 (ZSW2)

If you switch from a technology version <V4.0 to ≥V4.0 or vice versa, an error occurs during the compilation. You have to adapt the UDT names manually.

Version V3.0 (S7-1500)

Innovations

Technology Version V3.0 contains the following new features:

- Technology object measuring input
- Technology object output cam
- Cam track technology object
- Force/torque limiting
- Fixed stop detection
- Axis type virtual axis
- MC-PreServo [OB67] and MC-PostServo [OB95]
- Technology CPU S7-1500T

Additional innovations for S7-1500T:

Technology Version V3.0 contains the following additional improvements:

- Cam technology object (S7-1500T)
- Gearing with "MC_GearInPos" (S7-1500T)
- Camming with "MC_CamIn" (S7-1500T)
- Using multiple encoders (S7-1500T)

Motion Control instructions

Technology Version V3.0 contains the following Motion Control instructions:

- MC_Power V3.0
- MC_Reset V3.0
- MC_Home V3.0

- MC_Stop V3.0
- MC_MoveAbsolute V3.0
- MC_MoveRelative V3.0
- MC_MoveVelocity V3.0
- MC_MoveJog V3.0
- MC_MoveSuperimposed V3.0
- MC_SetSensor V3.0 (S7-1500T)
- MC_TorqueLimit V3.0
- MC_MeasuringInput V3.0
- MC_MeasuringInputCyclic V3.0
- MC_AbortMeasuringInput V3.0
- MC_OutputCam V3.0
- MC_CamTrack V3.0
- MC_GearIn V3.0
- MC_GearInPos V3.0 (S7-1500T)
- MC_PhasingAbsolute V3.0 (S7-1500T)
- MC_PhasingRelative V3.0 (S7-1500T)
- MC_CamIn V3.0 (S7-1500T)
- MC_SynchronizedMotionSimulation V3.0 (S7-1500T)
- MC_InterpolateCam V3.0 (S7-1500T)
- MC_GetCamFollowingValue V3.0 (S7-1500T)
- MC_GetCamLeadingValue V3.0 (S7-1500T)

Version V2.0 (S7-1500)

Innovations

Technology Version V2.0 contains the following new features:

- Synchronous axis technology object
- Gearing with "MC_GearIn"
- Superimposed positioning with "MC_MoveSuperimposed"
- Standardization of the "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control
- Simulation mode
- Support of safety functions of the drive

Motion Control instructions

Technology Version V2.0 contains the following Motion Control instructions:

- MC_Power V2.0
- MC_Reset V2.0
- MC_Home V2.0
- MC_Halt V2.0
- MC_MoveAbsolute V2.0
- MC_MoveRelative V2.0
- MC_MoveVelocity V2.0
- MC_MoveJog V2.0
- MC_MoveSuperimposed V2.0
- MC_GearIn V2.0

Version V1.0 (S7-1500)

Motion Control instructions

Technology Version V1.0 contains the following Motion Control instructions:

- MC_Power V1.0
- MC_Reset V1.0
- MC_Home V1.0
- MC_Halt V1.0
- MC_MoveAbsolute V1.0
- MC_MoveRelative V1.0
- MC_MoveVelocity V1.0
- MC_MoveJog V1.0

Changing a technology version (S7-1500)

Before you can access all the benefits of a new technology version, you need to change the technology version for existing projects.

Changing a technology version

To change the technology version, follow these steps:

1. Replace the CPU in the project with a CPU with an appropriate higher version.
2. Open the program editor (e.g. by opening the OB1).
   The technology objects and Motion Control instructions are highlighted in red after the CPU is replaced.

3. In the "Instructions" task card, select the appropriate higher technology version in the "Technology > Motion Control" folder.
4. Save and compile the project.
   The version of the technology objects and Motion Control instructions is changed to the appropriate higher technology version during compilation of the project.
   Pay attention to any error information that is displayed during compilation. Deal with the causes of the errors indicated.
5. Check the configuration of the technology objects.

Resetting the "Mode" parameter of the Motion Control instruction "MC_Home"

When the technology version is changed from V1.0 to ≥ V2.0, the "MC_Home.HomingMode" parameter (V1.0) is renamed to "MC_Home.Mode" (≥ V2.0). The assignment of the parameter values is changed as well.

To reset the "MC_Home.Mode" parameter (V2.0), follow these steps:

1. To change the technology version, follow the instructions given above.
   When compiling the project, the "MC_Home.HomingMode"parameter (V1.0) is renamed to "MC_Home.Mode" (≥ V2.0):
   - The assignment of the parameter values is changed. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and ≥ V2.0 is available in section Version overview (Page 1765).
     You can find additional information about the "MC_Home.Mode" parameter in the description of the Motion Control instruction "MC_Home".
   - The value configured at the "MC_Home.HomingMode" parameter (V1.0) is lost. As a note on renaming, the following text is entered as the parameter value in the "MC_Home.Mode" parameter (≥ V2.0).
     "The interface has changed. You can find additional information in the description of the Motion Control instruction "MC_Home".
   - There is a message in the "Info > Compile" tab in the Inspector window stating that the operand has the wrong data type.
2. Change the value of the "MC_Home.Mode" parameter (≥ V2.0) in your user program according to the new assignment.
3. Save and compile the project.

Replacing devices (S7-1500)

You can replace an S7-1500 with an S7-1500T and vice versa. The behavior with respect to functions and the existing configuration is different depending on what is being replaced.

- S7-1500 ⇒ S7-1500T
  The functions of the S7-1500 are expanded to include additional parameters for the extended functions of the S7-1500T. The additional parameters are preassigned with default values and must be configured appropriately.
- S7-1500T ⇒ S7-1500
  - Extended functions are only supported by an S7-1500T and are no longer available after a replacement with S7-1500.
  - Unsupported function blocks are marked.
  - Unsupported technology objects are displayed in an error message after compilation and must be deleted.

6.1.2.4 Configuring (S7-1500)

Adding and configuring drives in the device configuration (S7-1500)

Adding and configuring a drive is described in the following section with the example of a SINAMICS S120 drive. If you use a SINAMICS V90 PN or a SINAMICS drive with Startdrive, you can find additional information in the following documentation.

Using SINAMICS V90 PN

You need the Hardware Support Package (HSP) V90 two add and configure a SINAMICS V90 PN drive in the TIA Portal. Information on configuring a SINAMICS V90 PN drive with SIMATIC S7-1500 in the TIA Portal is available in the Getting Started "SIMATIC/SINAMICS First Steps SINAMICS V90 PN on S7-1500 Motion Control":

https://support.industry.siemens.com/cs/document/109739497 (https://support.industry.siemens.com/cs/ww/en/view/109739497)

Using Startdrive

If you use a SINAMICS drive with Startdrive, you can find additional information in the hardware catalog in the "Drives & Starter" folder. For additional information on connecting via Startdrive, refer to:

- "Getting Started SINAMICS S120 in the Startdrive V14 SP1"
  https://support.industry.siemens.com/cs/document/109747452 (https://support.industry.siemens.com/cs/ww/en/view/109747452)
- Application example "Positioning an S120 (Startdrive) with SIMATIC S7-1500 (TO)"
  https://support.industry.siemens.com/cs/document/109743270 (https://support.industry.siemens.com/cs/ww/en/view/109743270)

Drives compatibility list

In the appendix (Page 2146) you can find an overview of drives that can be interconnected with an S7-1500 CPU.

Adding and configuring PROFINET IO drives (S7-1500)

Adding and configuring a PROFINET IO drive is described below with the example of a SINAMICS S120 drive. Adding and configuring other PROFINET IO drives may differ from the description in certain respects.

When you use a S7-1500C CPU you can use the inputs/outputs of the CPU as interface to the drive.

Requirements

- The SIMATIC S7-1500 device is created in the project.
- The desired drive can be selected in the hardware catalog.

If the drive is not available in the hardware catalog, you must install the drive in the "Options" menu as a device description file (GSD).

Adding a drive and telegram in the device configuration

1. Open the device configuration and change to the network view.
2. In the hardware catalog, open the folder "Additional field devices > PROFINET IO > Drives > Siemens AG > SINAMICS".
3. Select the desired drive with the desired version, then drag it to the network view.
4. Assign the drive to the PROFINET interface of the CPU.
5. Open the drive in the device view.
6. Drag a Drive Object (DO) and a telegram from the hardware catalog and drop it onto a slot of the device overview of the drive.
7. Make sure that the order of the telegrams in the device configuration and in the drive parameter assignment are identical.

Depending on the version of the SINAMICS S120 drive, select "DO with telegram X", or "DO Servo" and a "Telegram X" for the telegram.

For more information on suitable telegrams, refer to the section PROFIdrive telegrams (Page 1636).

Repeat step 6, if you want to add another drive and another standard telegram.

Activating isochronous mode of the drive in the device configuration

PROFINET drives can always be operated in isochronous mode or clock synchronized mode. Isochronous mode, however, increases the quality of the closed loop position control of the drive and is therefore recommended for drives such as SINAMICS S120.

To control the drive in isochronous mode, follow these steps:

1. Open the device view of the drive.
2. In the properties window, select the tab "PROFINET interface [X1] > Advanced options > Isochronous mode".
3. Select the "Isochronous mode" check box in this tab.

The entry for the telegram also has to be selected for isochronous mode in the detailed view.

Interconnect the port of the CPU with the port of the drive

1. Open the topology view in the device configuration.
2. Interconnect the port of the drive as in the real configuration with the port of the CPU. At the same time, note the rules for topology configuration.

Configure the CPU as the sync master and set isochronous mode

1. Select the device view of the CPU.
2. In the Properties window, select the tab "PROFINET interface [X1] > Advanced options > Real-time settings > Synchronization".
3. Select "Sync master" from the "Synchronization role" drop-down list.
4. Click the "Domain settings" button.
5. Open the "Domain Management > Sync Domains" tab and set the desired "Send clock" (isochronous clock).

Select drive in the configuration of the technology object

1. Add a new technology object axis, or open the configuration of an existing axis.
2. Open the configuration "Hardware interface > Drive".
3. Select from the "PROFIdrive" entry in the "Drive type" drop-down list.
4. Select the Drive Object of the PROFINET drive from the "Drive" list.

For information on how to add a technology object, refer to the section Add technology object (Page 1783).

Result

The technology object is connected to the drive and the "MC-Servo" organization block can be checked / configured.

The telegram of the configured drive is assigned to the "PIP OB Servo" process image.

Checking/configuring the properties of the "MC-Servo"

1. Open the "Program blocks" folder in the project navigator.
2. Select the "MC-Servo" organization block.

3. Select the "Properties" command in the shortcut menu.
4. Select the "Cycle time" entry in the area navigation.
5. The option "Synchronous to the bus" must be selected in the dialog box.
6. A "PROFINET IO_System" must be selected in the "Distributed I/O" drop-down list.
7. The application cycle of "MC-Servo" must correspond to the send clock of the bus or be reduced by an integral factor relative to the send clock of the bus.

Result

The PROFINET IO drive is configured in such a way that it can be controlled in isochronous mode in the PROFINET IO network.

The properties of the SINAMICS drive must be configured according to the configuration of the axis with the STARTER software or SINAMICS Startdrive.

Checking isochronous mode on the drive

If the configuration sequence described above is not adhered to during configuration of the axis, and drive-specific errors occur when the project is compiled, the setting for isochronous mode on the drive must be checked.

1. Open the device view of the drive.
2. Select standard telegram in the device overview.
3. Select the properties dialog "General > I/O Addresses".
4. The following settings apply for the input and output addresses:
   - "Isochronous mode" is enabled.
   - "MC-Servo" must be selected for the "Organization block".
   - "PIP OB Servo" must be selected for the "Process image".

See also

PROFIdrive telegrams (Page 1636)

Add technology object (Page 1783)

Adding and configuring PROFIBUS DP drives (S7-1500)

Adding and configuring a PROFIBUS drive is described below with the example of a SINAMICS S120 drive. Adding and configuring other PROFIBUS drives may differ from the description in certain respects.

When you use a S7-1500C CPU you can use the inputs/outputs of the CPU as interface to the drive.

Requirements

- The SIMATIC S7-1500 device is created in the project.
- The desired drive can be selected in the hardware catalog.

If the drive is not available in the hardware catalog, you must install the drive in the "Options" menu as a device description file (GSD).

Adding a drive and telegram in the device configuration

1. Open the device configuration and change to the network view.
2. In the hardware catalog, open the folder "Additional Field Devices > PROFIBUS DP > Drives > Siemens AG > SINAMICS".
3. Select the folder of the desired drive with the desired version, then drag the drive object to the network view.
4. Assign the drive to the PROFIBUS interface of the CPU.
5. Open the drive in the device view.
6. Drag-and-drop a telegram from the hardware catalog onto a slot in the device overview of the drive.

For more information on suitable telegrams, refer to the section PROFIdrive telegrams (Page 1636).

If you want to add another drive and another telegram to the device overview, use the "Axis disconnector" in the hardware catalog.

Activating isochronous mode of the drive in the device configuration

PROFIBUS drives can be operated in cyclic mode or isochronous mode. Isochronous mode, however, increases the quality of the position control of the drive.

If you want to control the drive in isochronous mode, follow these steps:

1. Open the device view of the drive.
2. In the properties dialog, select the tab "General > Isochronous Mode".
3. Select the "Synchronize DP slave to constant DP bus cycle time" check box .

Setting isochronous mode

1. Select the network view.
2. Select the DP master system.
3. In the properties dialog, select the tab "General > Constant bus cycle time".
4. Select the desired "Constant DP bus cycle times".

Select drive in the configuration of the technology object

1. Add a new technology object axis, or open the configuration of an existing axis.
2. Open the configuration "Hardware interface > Drive".
3. Select from the "PROFIdrive" entry in the "Drive type" drop-down list.
4. Select the telegram of the PROFIBUS drive from the "Drive" list.

For information on how to add a technology object, refer to the section Add technology object (Page 1783).

Result

The technology object is connected to the drive and the "MC-Servo" organization block can be checked / configured.

The telegram of the configured drive is assigned to the "PIP OB Servo" process image.

Checking/configuring the properties of the "MC-Servo"

1. Open the "Program blocks" folder in the project navigator.
2. Select the "MC-Servo" organization block.
3. Select the "Properties" command in the shortcut menu.
   The "MC-Servo" dialog opens.
4. Select the "Synchronous to the bus" option under "General > Cycle time".
5. In the "Distributed I/O" drop-down list, select a "PROFIBUS DP-System".
   The application cycle of "MC-Servo" must correspond to the send clock of the bus or be reduced by an integral factor relative to the send clock of the bus.

You can select a drive connected to the CPU via a communications processor/communications module (CP/CM) in the configuration of the technology object. You cannot select the DP master system of the CP/CM as the source clock for MC-Servo [OB91].

Result

The PROFIBUS DP drive is configured in such a way that it can be controlled in isochronous mode in the PROFIBUS network.

The properties of the SINAMICS drive must be configured according to the configuration of the axis with the STARTER software or SINAMICS Startdrive.

Checking isochronous mode on the drive

If the configuration sequence described above is not adhered to during configuration of the axis, and drive-specific error occurs when the project is compiled, isochronous mode can be checked on the drive.

1. Open the device view of the drive.
2. Select the entry of the telegram in the device overview.

3. Select the properties dialog "General > I/O Addresses".
4. The following settings apply for the input and output addresses:
   – "MC-Servo" must be selected for the "Organization block".
   – "PIP OB Servo" must be select the "Process image".

See also

PROFIdrive telegrams (Page 1636)

Add technology object (Page 1783)

Adding and configuring drives with analog connections (S7-1500)

A description is provided below of how to add and configure a drive with an analog drive connection and an encoder. The connection is made, for example, as a positioning axis with an incremental encoder and using a technology module in the rack of the CPU.

When you use a S7-1500C CPU you can use the inputs/outputs of the CPU as interface to the drive.

Requirements

The SIMATIC S7-1500 device is created in the project.

Adding and configuring an analog output module in the device configuration

1. Open the device configuration of the CPU.
2. Select an analog output module from the hardware catalog and drag the module to the rack of the CPU.
3. Select the analog output module in the device view.
4. Open the "General" tab in the properties dialog and select there "Name of the Analog Output Module > I/O Addresses".
5. Enter the desired start address.
6. In the properties dialog, select the tab "General > IO Tags".
7. Enter the tag name for the desired analog output.

Adding and configuring a technology module

1. Switch to the device view of the CPU.
2. In the hardware catalog, open the folder "TM > Count > TM Count 2X24V".
3. Drag the counter module to a free slot in the rack. When in use in the rack of the CPU, the technology module cannot be operated in isochronous mode.
4. Select the technology module in the device view.

5. In the properties dialog, open the "General" tab and select "Count 2x24V > Basic parameters > Channel X > Operating mode" of the channel to be used.
6. Select "Position input for Motion Control" option for "Selection of the operating mode for the channel".
7. Under "Module parameters", adapt the parameters of the incremental encoder (steps per revolution = increments per revolution).
8. Under "Reaction to CPU STOP", select the item "Continue working".

Selecting the drive and encoder in the configuration of the technology object

1. Add a new positioning axis/synchronous axis technology object, or open the configuration of an existing positioning axis/synchronous axis.
2. Open the configuration "Hardware interface > Drive".
3. Select "Analog drive connection" from the "Drive type" drop-down list.
4. Select the previously defined tag name of the analog output from the "Output" list.
5. Open the configuration "Hardware interface > Encoder".
6. Under "Select encoder connection" select "Encoder on technology module (TM)".
7. Select the channel of the incremental encoder from the "Technology module" list under "local modules".
8. Open the configuration "Hardware interface > Data exchange with encoder > Fine resolution" and enter the "0" value for "bits in the incr. actual value".

For information on how to add a technology object, refer to the section Add technology object (Page 1783).

Result

The analog drive connection and the encoder connection are configured.

The analog addresses and the addresses of the TM module are assigned to the process image "PIP OB Servo".

Checking the encoder connection / drive connection

The encoder data are applied to the position control cycle clock. If in doubt, check the following settings:

1. Switch to the device view of the CPU.
2. Select the module technology.
3. Open the properties dialog "Basic Parameters > I/O Addresses".
4. The following settings apply for the input and output addresses:
   - "MC-Servo" must be selected for the "Organization block".
   - "PIP OB Servo" must be select the "Process image".
5. Select the analog module.

6. Open the properties dialog "Name of the Analog Module > I/O Addresses".
7. The settings from Step 4 apply for the input and output addresses

See also

Add technology object (Page 1783)

Configuration basics (S7-1500)

Add technology object (S7-1500)

The following describes how to add a technology object in the project tree.

Requirement

- A project with a CPU S7-1500 has been created.
- For the output cam, cam track, measuring input technology objects:
  A speed axis, positioning axis, synchronous axis or external encoder technology object is created in the project.

Procedure

To add a technology object, follow these steps:

1. Open the CPU's folder in the project navigator.
2. Open the "Technology Objects" folder.
3. Double-click "Add new object".
   The "Add new object" dialog opens.
4. Select the required technology object. You can infer the function of the technology object from the displayed description.
5. If you add output cams, cam tracks, measuring inputs to a technology object, select the higher-level technology object in the "Axis or external encoder that is to be assigned".
6. In the "Name" input field, adapt the name to your requirements.
7. To change the suggested data block number, select the "Manual" option.
8. To add your own information about the technology object, click "Additional information".
9. To open the configuration after adding the technology object, select the "Add new and open" check box.
10. To add the technology object, click "OK".

Result

The new technology object was created and created in the "Technology objects" folder in the project tree.

If the "MC-Servo" and "MC-Interpolator" organizational units were not yet available, they were added.

Copy technology object (S7-1500)

You can copy a technology object in the following ways:

- Copying a technology object within a CPU
- Copying a technology object from a CPU S7-1500 to a CPU S7-1500T
  Additional parameters for the extended functions are preset with default values. You must configure them appropriately.
- Copying a technology object from a CPU S7-1500T to a CPU S7-1500
  Additional parameters, which are not supported by the CPU S7-1500, are reset to the default values.

When you copy a technology object that has lower-level technology objects such as output cams, cam tracks or measuring inputs, the lower-level technology object are also copied.

The following describes how to copy a technology object within a CPU. The procedure also applies accordingly to the other copying methods mentioned.

Requirement

- A project with a CPU S7-1500 has been created.
- You have created an technology object in the project.

Procedure

To copy a technology object, follow these steps:

1. Open the CPU's folder in the project navigator.
2. Open the "Technology Objects" folder.
3. If necessary, open the higher-level technology object.
4. Select the technology object to be copied.
5. Select "Copy" in the shortcut menu.
6. Select the "Technology objects" folder or the higher-level technology object.
7. Select "Paste" in the shortcut menu.

Result

The selected technology object, including lower-level technology objects, has been copied and created in the "Technology objects" folder of the project tree.

Delete technology object (S7-1500)

You can delete technology objects in the project tree.

When you delete a technology object that has lower-level technology objects such as output cams, cam tracks or measuring inputs, the lower-level technology object are also deleted.

Requirement

- A project with a CPU S7-1500 has been created.
- You have created an technology object in the project.

Procedure

To delete a technology object, follow these steps:

1. Open the CPU's folder in the project navigator.
2. Open the "Technology Objects" folder.
3. Select the technology object to be deleted.
4. Select the "Delete" command in the shortcut menu.
   The "Confirm delete" dialog is opened.
5. To delete the technology object, click "Yes".

Result

The selected technology object has been deleted.

See also

Compare values (Page 1786)

Working with the configuration editor (S7-1500)

You configure the properties of a technology object in the configuration window. To open the configuration window of the technology object in the project view, follow these steps:

1. Open the device "Technology objects" group in the project navigator.
2. Select the technology object and double-click on "Configuration".

The configuration is divided into categories which depend on the object type, for example, basic parameters, hardware interface, extended parameters.

Configuration editor icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

Symbol	Description
	The configuration contains default values and is complete. The configuration contains only default values. With these default values you can use the technology object without additional changes.
	The configuration contains user-defined or automatically adapted values and is complete. All input fields of the configuration contain valid values and at least one preset value has been changed.
	The configuration is incomplete or incorrect. At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the field shows you the roll-out error message that indicates the cause of error.

Compare values (S7-1500)

If an online connection to the CPU is available, the "Monitor all" function appears in the configuration of the technology object.

The "Monitor all" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of actual values in the project
- Transfer of start values of the project to the CPU as actual values

Note

Differences between online and offline values

By adding or deleting technology objects that have a connection to other technology objects, such as cams, cam tracks, measuring inputs or synchronized axes, differences can occur when online and offline values are compared. These differences can be eliminated by re-compiling the project and then uploading it to the CPU.

Icons and operator controls

If there is an online connection to the CPU, the actual values are displayed at the parameters.

In addition to the actual values of the parameters, the following symbols appear:

Symbol	Description
	Start value in CPU matches the configured start value in project
	Start value in CPU does not match the configured start value in project

Symbol	Description
	Software error in lower-level component: The online and offline versions differ in at least one lower-level software component.
	The comparison of the Start value in CPU with the configured start value in project cannot be performed.
	Comparison of the online and offline values is not advisable.
	Use this button to show the start value of the CPU and the start value of the project for the respective parameter.

You can change the start value of the CPU directly and then download it to the CPU. For directly editable parameters, the actual value can also be changed and the change will be transferred directly to the CPU.

Configuring the Speed-Control Axis technology object (S7-1500)

Configuration - Basic Parameters (S7-1500)

Configure the basic properties of the technology object in the "Basic Parameters" configuration window.

Name

Define the name of the speed axis in this field. The technology object is listed under this name in the project tree. You can use the variables of the speed axis in the user program under this name.

Axis type

If you want to use the axis in the CPU exclusively as a virtual leading axis for synchronization, for example, select the "Virtual axis" check box. The configuration of a drive and encoder connection is not relevant.

Units of measure

In the drop-down lists, select the desired units of measurement for the speed and the torque.

Simulation

If you want to move a real axis in the simulation mode, select the "Activate simulation" check box.

In simulation mode, speed, positioning and synchronous axes can be simulated in the CPU without connected drives and encoders. Simulation mode is possible as of Technology Version V3.0 even without a configured drive and encoder connection.

For simulation mode without hardware connected to the CPU, you can influence the startup time of the CPU via the "Configuration time for central and distributed I/Os" parameter. You can find the parameter in the CPU properties in the "Startup" area navigation.

See also

Units of measure (Page 1632)

Axis in simulation (Page 1649)

Hardware interface (S7-1500)

Configuration - Drive (S7-1500)

In the "Drive" configuration window, configure which drive type and which drive you want to use.

Drive type

In the drop-down list, select whether you want to deploy a PROFIdrive drive or a drive with an analog drive connection.

PROFIdrive drives are connected to the controller by means of a digital communication system (PROFINET or PROFIBUS). The communication is performed via PROFIdrive telegrams.

Drives with an analog drive connection receive the speed setpoint via an analog output signal (e.g. from -10 V to +10 V) from the CPU.

Drive type: PROFIdrive

Data connection

In the drop-down list, select whether the data connection is to be made directly with the drive device or via an editable data block in the user program.

Drive/data block

In the "Drive" field, select an already configured PROFIdrive drive/slot. If you have selected a PROFIdrive drive, you can configure the PROFIdrive drive using the "Device configuration" button.

If no PROFIdrive drive is available for selection, switch to the device configuration, and add a PROFIdrive drive in the network view.

Note

Option "Show all modules"

If a PROFIdrive that has already been configured is not available for selection, use the option "Show all modules" to display all reachable modules.

When you select the option "Show all modules", only the address range for each of the displayed modules is checked. If the address range of the module is large enough for the selected PROFIdrive frame, you can select the module. For this reason, make sure that you select a PROFIdrive drive.

If you have selected "Data block" under the data connection, select a previously created data block which contains a tag structure of the data type "PD_TELx" ("x" stands for the telegram number to be used).

Drive type: Analog drive connection

Analog output

In the "Analog output" field, select the PLC tag of the analog output via which the drive is to be controlled.

In order to be able to select an output, you first need to add an analog output module in the device configuration and define the PLC tag name for the analog output.

Activating enable output

Select the "Activate enable output" check box if the drive supports an enable.

Select the PLC tag of the digital output for the drive enable in the corresponding field. With the enable output, the speed controller in the drive is enabled, or disabled.

In order to be able to select an enable output, a digital output module must be added in the device configuration and the PLC tag name must be defined for the digital output.

Note

If you do not use an enable output, the drive cannot be immediately disabled on the part of the system due to error reactions or monitoring functions. A controlled stop of the drive is not guaranteed.

Enable ready input

Select the "Enable ready input" check box if the drive can signal its readiness.

Select the PLC tag of the digital input via which the drive is to signal its operational readiness to the technology object in the corresponding field. The power module is switched on and the analog speed setpoint input is enabled.

In order to be able to select a ready input, you first need to add a digital input module in the device configuration and define the PLC tag name for the digital input.

Note

The enable output and the ready input can be separately enabled.

The following boundary conditions apply to the activated ready input:

- The axis is only enabled ("MC_Power Status" = TRUE) when a signal is present at the ready input.
- If a signal is not present at the ready input on an enabled axis, the axis of the alarm response "Remove enable" is disabled.
- If the axis is disabled with the instruction "MC_Power" ("Enable" = FALSE), the axis is disabled even when a signal is present at the ready input.

See also

Configuration - Data exchange with the drive (Page 1790)

Connect drive/encoder via data block (Page 1908)

Drive and encoder connection (Page 1634)

Configuration - Data exchange with the drive (S7-1500)

Configure the data exchange with the drive in the "Data exchange with the drive" configuration window.

The configuration differs according to the selected drive type:

Drive type: PROFIdrive

Drive telegram

The telegram to the drive that is set in the device configuration is preselected in the drop-down list.

Automatic data exchange for drive values (offline)

Select the check box if you want to transfer the offline values of the drive "Reference speed", "Maximum speed" and "Reference torque" to the configuration of the technology object in the project.

Automatic data exchange for drive values (online)

Select the check box if you want to transfer the effective values "Reference speed", "Maximum speed" and "Reference torque" online in the drive to the CPU during runtime. The drive parameters are transferred from the bus after the (re-)initialization of the technology object or the (re)start of the drive or the CPU.

Alternatively, you must synchronize the following parameters manually:

- Reference speed
  Configure the reference speed of the drive in accordance with the manufacturer's specifications in this field. The specification of the drive speed is a percentage of the reference speed in the range -200% to 200%.
- Maximum speed
  Configure the maximum speed of the drive in this field.
- Reference torque
  Configure the reference torque of the drive corresponding to its configuration in this field. The reference torque is needed for force/torque reduction, which is supported with telegram 10x.

Supplementary data

Select the "Torque data" check box if you want to configure the data connection of the torque data. If you have selected a drive with which the additional telegram 750 has been configured, the "Torque data" check box is preselected.

Data connection

In the drop-down list, define whether the data connection should be made via additional telegrams or data blocks:

- If you select the entry "Additional telegram" in the "Data connection" drop-down list, you can edit the "Additional telegram" drop-down list.
- If you select the "Data block" entry in the "Data connection" drop-down list, you can select the previously created data block which contains a tag structure of the "PD_TELx" data type ("x" stands for the additional telegram number that is used).

Data block / additional telegram

Select an additional telegram configured in the "Additional telegram" field.

Select the "Show all modules" check box if you want to display all submodules of the connected drive. You can also find self-defined additional telegrams with this function.

In the "Data block" field, select the data block which you want to use to integrate the torque data.

Note

Automatic transfer of drive parameters is only possible with SINAMICS drives as of V4.x. For this, "Drive" must be selected for the data connection in configuration window "Hardware interface > Drive".

Drive type: Analog drive connection

Reference speed

The reference speed of the drive is the speed with which the drive spins when there is an output of 100% at the analog output. The reference speed must be configured for the drive and transferred in the configuration of the technology object.

The analog value that is output at 100% depends on the type of the analog output. For example, for an analog output with +/- 10 V, the value 10 V is output at 100%.

Analog outputs can be overridden by approximately 17%. This means that an analog output can be operated in the range from -117% to 117%, insofar as the drive permits this.

See also

Automatic transfer of drive and encoder parameters in the device (Page 1641)

Connect drive/encoder via data block (Page 1908)

Drive and encoder connection (Page 1634)

Extended Parameters (S7-1500)

Configuration - Mechanics (S7-1500)

Configure the connection of the load to the drive in the "Mechanics" configuration window.

Drive mechanism

Select the "Invert drive direction" check box if the direction of rotation of the drive is to be inverted.

Load gear

The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions.

In this "Number of motor revolutions" configuration field, configure the integer number of motor revolutions.

In this "Number of load revolutions" configuration field, configure the integer number of load revolutions.

See also

Mechanics (Page 1653)

Configuration - Dynamic Defaults (S7-1500)

In the "Dynamic default values" configuration window, configure the default values for speed, acceleration, deceleration jerk of the axis.

The default values take effect when values < 0 are specified in Motion Control instructions for the "Velocity", "Acceleration", "Deceleration" or "Jerk" parameters. The default values can be applied separately for each of the parameters just listed.

Speed

In this field, define the default value for the speed of the axis.

Acceleration/deceleration - Ramp-up time/ramp-down time

Set the desired default value for acceleration in the "Ramp-up time" or "Acceleration" fields. The desired deceleration can be set in the "Ramp-down time" or "Deceleration" fields.

The following equations show the relationship between the ramp-up time and acceleration and the ramp-down time and deceleration:

$$\textit{Ramp-up time} = \frac{\textit{Speed}}{\textit{Acceleration}}$$

$$\textit{Ramp-down time} = \frac{\textit{Speed}}{\textit{Deceleration}}$$

Note

A change in the velocity influences the acceleration and deceleration values of the axis. The ramp-up and ramp-down times are retained.

Smoothing time/jerk

You can enter the jerk limit parameters in the "Smoothing time" box, or alternatively in the "Jerk" box:

- Set the desired jerk for the acceleration and deceleration ramp in the "Jerk" field. The value 0 means that jerk limiting is deactivated.
- Set the desired smoothing time for the acceleration ramp in the "Smoothing time" field.

Note

The jerk value is identical for the acceleration and deceleration ramp. The smoothing time in effect for the deceleration ramp results from the following relationships:

- **Acceleration > Deceleration**
  A shorter smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration < Deceleration**
  A longer smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration = Deceleration**
  The smoothing times of the acceleration and deceleration ramp are equal.

If an error occurs, the axis decelerates with the configured emergency stop deceleration. A configured jerk limit is not taken into account for this.

The following equations show the relationship between the smoothing times and the jerk:

$$\text{Smoothing time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Smoothing time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Motion jobs started in the user program are performed with the selected jerk.

See also

Velocity profile (Page 1660)

Configuration - Emergency stop (S7-1500)

In the "Emergency stop" configuration window, you can configure the emergency stop deceleration of the axis. In the event of an error, and when disabling the axis, the axis is brought to a standstill with this deceleration using the Motion Control instruction "MC_Power" (input parameter "StopMode" = 0).

Emergency deceleration/emergency stop ramp-down time

Configure the deceleration value for emergency stop in the "Emergency stop deceleration" field or the "Emergency stop ramp-down time" field.

The relationship between emergency stop ramp-down time and emergency stop deceleration can be seen in the following equation:

$$\text{Emergency stop ramp-down time} = \frac{\text{Maximum speed}}{\text{Emergency stop deceleration}}$$

The configuration of the emergency stop deceleration is related to the configured maximum speed of the axis. If the maximum speed of the axis is changed, then the value of the emergency deceleration also changes (the emergency stop ramp-down time remains unchanged).

See also

Emergency stop deceleration (Page 1661)

Limits (S7-1500)

Configuration - Dynamic limits (S7-1500)

In the "Dynamic limits" configuration window, configure the maximum values for speed, acceleration, deceleration and jerk of the axis.

Maximum speed

In this field, define the maximum permitted speed of the axis.

Maximum acceleration/maximum deceleration - ramp-up time/ramp-down time

Set the desired acceleration in the "Ramp-up time" or "Acceleration" fields. The desired deceleration can be set in the "Ramp-down time" or "Deceleration" fields.

The following equations show the relationship between the ramp-up time and acceleration and the ramp-down time and deceleration:

$$\text{Ramp-up time} = \frac{\text{Maximum velocity}}{\text{Acceleration}}$$

$$\text{Ramp-down time} = \frac{\text{Maximum speed}}{\text{Deceleration}}$$

Note

Change of maximum speed

A change in the maximum speed influences the acceleration and deceleration values of the axis. The ramp-up and ramp-down times are retained.

Smoothing time/jerk

You can enter the jerk limit parameters in the "Smoothing time" box, or alternatively in the "Jerk" box:

- Set the desired jerk for the acceleration ramp and the deceleration ramp in the "Maximum jerk" box. The value 0 means that the jerk is not limited.
- Set the desired smoothing time for the acceleration ramp in the "Smoothing time" field.

Note

Various acceleration and deceleration values

The configured smoothing time displayed in the configuration, applies only to the acceleration ramp.

If the values of the acceleration and deceleration differ, the smoothing time of the deceleration ramp is calculated and used according to the jerk of the acceleration ramp.

The smoothing time of the deceleration is adapted as follows:

- **Acceleration > Deceleration**
  A shorter smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration < Deceleration**
  A longer smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration = Deceleration**
  The smoothing times of the acceleration and deceleration ramp are equal.

If an error occurs, the axis decelerates with the configured emergency stop deceleration (Page 1660) (alarm reaction "Stop with maximum dynamic values"). A configured jerk limit is not taken into account for this.

The following equations show the relationship between the smoothing times and the jerk:

$$\text{Smoothing time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Smoothing time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Motion jobs started in the user program are performed with the selected jerk.

See also

Emergency stop deceleration (Page 1661)

Dynamic limits in synchronous operation (Page 1662)

Configuration - Torque limits (S7-1500)

Configure the torque limiting of the drive in the "Torque limits" configuration window.

The configuration is only available if a drive that supports force/torque limiting is selected and a telegram 10x is used.

Effective

In the drop-down list, select whether the limit value is to be in effect "on load side" or "on motor side".

Torque limiting

Enter a default value for the torque limit in the specified unit of measure in this field.

The default value is in effect when the torque limiting is specified using Motion Control instruction "MC_TorqueLimiting", input parameter "Limit" < 0.

If the efficiency of the gear is crucial, you can set it in the "<TO>.Actor.Efficiency" tag.

See also

Force/torque limiting (Page 1665)

Configuring the positioning axis/synchronous axis technology object (S7-1500)

Configuration - Basic Parameters (S7-1500)

Configure the basic properties of the technology object in the "Basic Parameters" configuration window.

Name

Define the name of the positioning axis/synchronous axis in this field. The technology object is listed under this name in the project tree. The variables of the technology object can be used in the user program under this name.

Axis type

If you want to use the axis in the CPU exclusively as a virtual leading axis for synchronization, for example, select the "Virtual axis" check box. The configuration of a drive and encoder connection is not relevant.

In this selection, configure whether the axis should perform linear or rotary motions.

Units of measure

In the drop-down list, select the desired units of measure for the position, velocity, torque and force of the axis.

Modulo

Select the "Enable modulo" check box if you want to use a recurring system of units for the axis (e.g. 0 to 360° for an axis of the "rotary" axis type).

- **Modulo start value**
  In this field, define the position at which the modulo range should begin (e.g. 0° for an axis of the "rotary" axis type).
- **Modulo length**
  In this field, define the length of the modulo range (e.g. 360° for an axis of the "rotary" axis type).

Simulation

If you want to move a real axis in the simulation mode, select the "Activate simulation" check box.

In simulation mode, speed, positioning and synchronous axes can be simulated in the CPU without connected drives and encoders. Simulation mode is possible as of Technology Version V3.0 even without a configured drive and encoder connection.

For simulation mode without hardware connected to the CPU, you can influence the startup time of the CPU via the "Configuration time for central and distributed I/Os" parameter. You can find the parameter in the CPU properties in the "Startup" area navigation.

See also

Axis in simulation (Page 1649)

Units of measure (Page 1632)

Modulo setting (Page 1633)

Mechanics (Page 1653)

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Hardware interface (S7-1500)

Configuration - Drive (S7-1500)

In the "Drive" configuration window, configure which drive type and which drive you want to use.

Drive type

In the drop-down list, select whether you want to deploy a PROFIdrive drive or a drive with an analog drive connection.

PROFIdrive drives are connected to the controller by means of a digital communication system (PROFINET or PROFIBUS). The communication is performed via PROFIdrive telegrams.

Drives with an analog drive connection receive the speed setpoint via an analog output signal (e.g. from -10 V to +10 V) from the CPU.

Drive type: PROFIdrive

Data connection

In the drop-down list, select whether the data connection is to be made directly with the drive device or via an editable data block in the user program.

Drive/data block

In the "Drive" field, select an already configured PROFIdrive drive/slot. If you have selected a PROFIdrive drive, you can configure the PROFIdrive drive using the "Device configuration" button.

If no PROFIdrive drive is available for selection, switch to the device configuration, and add a PROFIdrive drive in the network view.

Note

Option "Show all modules"

If a PROFIdrive that has already been configured is not available for selection, use the option "Show all modules" to display all reachable modules.

When you select the option "Show all modules", only the address range for each of the displayed modules is checked. If the address range of the module is large enough for the selected PROFIdrive frame, you can select the module. For this reason, make sure that you select a PROFIdrive drive.

If you have selected "Data block" under the data connection, select a previously created data block which contains a tag structure of the data type "PD_TELx" ("x" stands for the telegram number to be used).

Drive type: Analog drive connection

Analog output

In the "Analog output" field, select the PLC tag of the analog output via which the drive is to be controlled.

In order to be able to select an output, you first need to add an analog output module in the device configuration and define the PLC tag name for the analog output.

Activating enable output

Select the "Activate enable output" check box if the drive supports an enable.

Select the PLC tag of the digital output for the drive enable in the corresponding field. With the enable output, the speed controller in the drive is enabled, or disabled.

In order to be able to select an enable output, a digital output module must be added in the device configuration and the PLC tag name must be defined for the digital output.

Note

If you do not use an enable output, the drive cannot be immediately disabled on the part of the system due to error reactions or monitoring functions. A controlled stop of the drive is not guaranteed.

Enable ready input

Select the "Enable ready input" check box if the drive can signal its readiness.

Select the PLC tag of the digital input via which the drive is to signal its operational readiness to the technology object in the corresponding field. The power module is switched on and the analog speed setpoint input is enabled.

In order to be able to select a ready input, you first need to add a digital input module in the device configuration and define the PLC tag name for the digital input.

Note

The enable output and the ready input can be separately enabled.

The following boundary conditions apply to the activated ready input:

- The axis is only enabled ("MC_Power Status" = TRUE) when a signal is present at the ready input.
- If a signal is not present at the ready input on an enabled axis, the axis is disabled with an error.
- If the axis is disabled with the instruction "MC_Power" ("Enable" = FALSE), the axis is disabled even when a signal is present at the ready input.

See also

Connect drive/encoder via data block (Page 1908)

Data connection drive/encoder via data block (Page 1650)

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Configuration - Encoder (S7-1500)

For closed loop position control, positioning axes/synchronous axes require an actual position value in the form of an encoder position. The encoder position is transmitted to the controller by means of a PROFIdrive telegram.

Configure the encoder and encoder type in the "Encoder" configuration window.

Data connection

In the drop-down list, select whether the data connection should be established directly with the encoder or via a data block that can be edited in the user program.

The selection is only possible for encoders that are connected via PROFIdrive and support parameter P979.

Encoder/data block

Select a previously configured encoder in this configuration field.

The following encoders can be selected:

- **Connection to the drive (not with analog drive connection)**
  The encoder is configured via the configuration of the PROFIdrive drive. The drive evaluates the encoder signals and sends them to the controller in the PROFIdrive telegram.
- **Encoder on technology module (TM)**
  Select a previously configured technology module and the channel to be used. Only technology modules set to the "Position input for Motion Control" mode are displayed for selection.
  If no technology module is available for selection, change to the device configuration and add a technology module. If you have selected a technology module, you can access the configuration of the technology module using the "Device configuration" button.
  You can operate the technology module centrally on an S7-1500 CPU or decentrally on a distributed I/O. Isochronous mode is not possible with central operation in the CPU.
  You can identify the technology modules suitable for position detection for Motion Control in the documentation for the technology module and the catalog data.
- **PROFIdrive encoder on PROFINET/PROFIBUS (PROFIdrive)**
  In the "PROFIdrive encoder" field, select a configured encoder on PROFINET/PROFIBUS.
  If you have selected an encoder, you can configure this using the "Device configuration" button.
  Switch to the device configuration in the network view, and add an encoder, in the event that no encoder can be selected.

  Note

  Option "Show all modules"

  If a PROFIdrive that has already been configured is not available for selection, use the option "Show all modules" to display all reachable modules.

  When you select the option "Show all modules", only the address range for each of the displayed modules is checked. If the address range of the module is large enough for the selected PROFIdrive frame, you can select the module. For this reason, make sure that you select a PROFIdrive encoder.

If you have selected "Data block" under the data connection, select a previously created data block which contains a tag structure of the data type "PD_TELx" ("x" stands for the telegram number to be used).

Encoder type

Select the encoder type of the encoder in the drop-down list. The following encoder types are available:

- Incremental
- Absolute
- Cyclic absolute

See also

Configuration - Encoder (multiple encoders) (Page 1801)

Positioning axis technology object (Page 1704)

Data connection drive/encoder via data block (Page 1650)

Connect drive/encoder via data block (Page 1908)

Synchronous axis technology object (Page 1705)

Configuration - Encoder (multiple encoders)

With the S7-1500T technology CPU, you can switch between 4 encoder systems. You control the changeover in the user program with the Motion Control instruction "MC_SetSensor".

You configure the 4 encoders, their use and their type in the "Encoder" configuration window.

Encoder on startup

In the drop-down list, select the encoder that is to be active after startup of the CPU (STARTUP). The encoder must be configured and marked as "used".

This encoder is used after startup of the CPU and after a restart of the technology object (Page 1946). At an operating mode transition from STOP → RUN of the CPU (without restart of the technology object), the encoder that was also active before the STOP is still being used.

Use encoder

Select the "Use encoder" check box if you want to use this encoder alternatively for closed loop position control.

See also

Using multiple encoders (Page 1643)

Configuration - Encoder (Page 1798)

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Configuration - Data exchange with the drive (S7-1500)

Configure the data exchange with the drive in the "Data exchange with the drive" configuration window.

The configuration differs according to the selected drive type:

Drive type: PROFIdrive

Drive telegram

The telegram to the drive that is set in the device configuration is preselected in the drop-down list.

Automatic data exchange for drive values (offline)

Select the check box if you want to transfer the offline values of the drive "Reference speed", "Maximum speed" and "Reference torque" to the configuration of the technology object in the project.

Automatic data exchange for drive values (online)

Select the check box if you want to transfer the effective values "Reference speed", "Maximum speed" and "Reference torque" online in the drive to the CPU during runtime. The drive parameters are transferred from the bus after the (re-)initialization of the technology object or the (re)start of the drive or the CPU.

Alternatively, you must synchronize the following parameters manually:

- Reference speed
  Configure the reference speed of the drive in accordance with the manufacturer's specifications in this field. The specification of the drive speed is a percentage of the reference speed in the range -200% to 200%.
- Maximum speed
  Configure the maximum speed of the drive in this field.
- Reference torque
  Configure the reference torque of the drive corresponding to its configuration in this field. The reference torque is needed for force/torque reduction, which is supported with telegram 10x.

Supplementary data

Select the "Torque data" check box if you want to configure the data connection of the torque data. If you have selected a drive with which the supplemental telegram 750 has been configured, the "Torque data" check box is preselected.

Data connection

In the drop-down list, define whether the data connection should be made via supplemental telegrams or data blocks:

- If you select the entry "Supplemental telegram" in the "Data connection" drop-down list, you can edit the "Supplemental telegram" drop-down list.
- If you select the "Data block" entry in the "Data connection" drop-down list, you can select the previously created data block which contains a tag structure of the "PD_TELx" data type ("x" stands for the additional telegram number that is used).

Data block / supplemental telegram

Select an supplemental telegram configured in the "Supplemental telegram" field.

Select the "Show all modules" check box if you want to display all submodules of the connected drive. You can also find self-defined supplemental telegrams with this function.

In the "Data block" field, select the data block which you want to use to integrate the torque data.

Note

Automatic transfer of drive parameters is only possible with SINAMICS drives as of V4.x. To do this, set the "Drive" data connection in the configuration window "Hardware interface > Drive".

Drive type: Analog drive connection

Reference speed

The reference speed of the drive is the speed with which the drive spins when there is an output of 100% at the analog output. The reference speed must be configured for the drive and transferred in the configuration of the technology object.

The analog value that is output at 100% depends on the type of the analog output. For example, for an analog output with +/- 10 V, the value 10 V is output at 100%.

Analog outputs can be overridden by approximately 17%. This means that an analog output can be operated in the range from -117% to 117%, insofar as the drive permits this.

Maximum speed

Specify the maximum speed of the drive in this field.

See also

Positioning axis technology object (Page 1704)

Automatic transfer of drive and encoder parameters in the device (Page 1641)

Connect drive/encoder via data block (Page 1908)

Synchronous axis technology object (Page 1705)

Configuration - Data exchange with encoder (S7-1500)

Configure detailed encoder parameters and the data exchange with the encoder in the "Data exchange with encoder" configuration window.

The display and selection of the configuration parameters described here is dependent on the following parameters:

- Configuration window "Basic parameters": Drive type (linear/rotary)
- Configuration window "Hardware interface > Encoder": Encoder type (incremental/absolute/cyclic absolute)
- Configuration window - "Extended parameters > Mechanics": Encoder mounting type

Encoder telegram

The telegram to the encoder that is set in the device configuration is preselected in the drop-down list.

Automatic data exchange for encoder values (offline)

Select the check box if you want to transfer the offline values of the encoder to the configuration of the technology object in the project.

Note

Automatic transfer of encoder parameters is only possible with PROFIdrive encoders as of product version A16. For this, "Hardware interface > Encoder" must be selected as the "Encoder" data connection in the configuration window.

Automatic data exchange for encoder values (online)

Select the check box if you want to transfer the effective values online in the encoder to the CPU during runtime. The encoder parameters are transferred from the bus after the (re-)initialization of the technology object and (re)start of the encoder or the CPU.

Note

Automatic transfer of encoder parameters is only possible with PROFIdrive encoders as of product version A16. For this, "Hardware interface > Encoder" must be selected as the "Encoder" data connection in the configuration window.

Alternatively, you must manually calibrate the following parameters, depending on encoder type.

Encoder type

Depending on the selected encoder type, configure the following parameters:

Encoder type	Rotary incremental
Increments per revolution	Configure the number of increments that the encoder resolves per revolution in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.

Encoder type	Rotary absolute
Increments per revolution	Configure the number of increments that the encoder resolves per revolution in this field.
Number of revolutions	Configure the number of revolutions that the absolute encoder can detect in this field.

Encoder type	Rotary absolute
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

Encoder type	Rotary cyclic absolute
Increments per revolution	Configure the number of increments that the encoder resolves per revolution in this field.
Number of revolutions	Configure the number of revolutions that the absolute encoder can detect in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

Encoder type	Linear incremental
Distance between two increments	Configure the distance between two increments of the encoder in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.

Encoder type	Linear absolute
Distance between two increments	Configure the distance between two increments of the encoder in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

Encoder type	Linear cyclic absolute
Distance between two increments	Configure the distance between two increments of the encoder in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

See also

Configuration - Data exchange with encoder (multiple encoders) (Page 1806)
Positioning axis technology object (Page 1704)
Automatic transfer of drive and encoder parameters in the device (Page 1641)
Connect drive/encoder via data block (Page 1908)
Synchronous axis technology object (Page 1705)
Configuration - Data exchange (Page 1835)

Configuration - Data exchange with encoder (multiple encoders)

Settings for

In the drop-down list, select the encoder for which the following configurations are to apply.

See also

Using multiple encoders (Page 1643)
Configuration - Data exchange with encoder (Page 1802)
Configuration - Mechanics (Page 1807)
Positioning axis technology object (Page 1704)
Synchronous axis technology object (Page 1705)

Configuration - leading value interconnections (synchronous axis only) (S7-1500)

You can interconnect a synchronous axis with multiple leading value-capable technology objects. Leading value-capable technology objects are:

- Positioning axes
- Synchronous axes
- External encoders (only with S7-1500T CPU)

You can select only one leading value during runtime of your user program.

All interconnection required during operation must be set up during configuration of the technology object.

Possible leading values

In the "Possible leading values" column of the "Leading value interconnections" list, add all leading value-capable technology objects that you need during operation as a leading value for the synchronous axis.

You can interconnect the axes added in the list with the synchronous axis as a leading value with the corresponding Motion Control instruction.

All configured leading value interconnections for the technology object are displayed in the cross-reference list of the technology object.

Coupling type

In the "Type of coupling" column, configure whether the leading value is to be coupled via setpoint or actual value.

"Actual value" is available only for the S7-1500T CPU.

See also

Leading value coupling (Page 1706)

Synchronous axis technology object (Page 1705)

Extended Parameters (S7-1500)

Configuration - Mechanics (S7-1500)

Configuration - Mechanics (S7-1500)

In the "Mechanics" configuration window, configure the mounting type of the encoder, and the adaptation of the actual encoder value to the mechanical conditions.

Settings for (S7-1500T)

In the drop-down list, select the encoder for which the following configurations are to apply.

Encoder mounting type

In the drop-down list, select how the encoder is mounted to the mechanics.

The configuration differs depending on the axis type and the encoder mounting type selected in the "Basic parameters" configuration window.

Axis type: Linear

- Linear - On motor shaft (Page 1808)
- Linear - On load side (Page 1808)
- Linear - External Measuring System (Page 1809)

Axis type: Rotary

- Rotary - On motor shaft (Page 1810)
- Rotary - On load side (Page 1810)
- Rotary - External Measuring System (Page 1811)

Invert encoder direction

Select this check box if you must invert the direction of rotation of the encoder.

See also

Configuration - Data exchange with encoder (multiple encoders) (Page 1805)

Positioning axis technology object (Page 1704)

Using multiple encoders (Page 1643)

Synchronous axis technology object (Page 1705)

Axis type: Linear (S7-1500)

Linear - On motor shaft (S7-1500)

The encoder is connected to the motor shaft in a mechanically fixed manner. Motor and encoder form a unit.

Drive mechanism

Select the "Invert drive direction" check box if the direction of rotation of the drive is to be inverted.

Load gear

The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions.

In this "Number of motor revolutions" configuration field, configure the integer number of motor revolutions.

In this "Number of load revolutions" configuration field, configure the integer number of load revolutions.

Position parameters

In the "Leadscrew pitch" configuration field, configure the distance by which the load is moved when the leadscrew makes one revolution.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Linear - On load side (S7-1500)

The encoder is mechanically connected to the load side of the gear.

Drive mechanism

Select the "Invert drive direction" check box if the direction of rotation of the drive is to be inverted.

Load gear

The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions.

In this "Number of motor revolutions" configuration field, configure the integer number of motor revolutions.

In this "Number of load revolutions" configuration field, configure the integer number of load revolutions.

Position parameters

In the "Leadscrew pitch" configuration field, configure the distance by which the load is moved when the leadscrew makes one revolution.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Linear - External Measuring System (S7-1500)

An external measuring system provides the position values of the linear load motion.

Distance per encoder revolution

In this configuration field, configure the linear load travel per encoder revolution.

Drive mechanism

Select the "Invert drive direction" check box if the direction of rotation of the drive is to be inverted.

Load gear

The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions.

In this "Number of motor revolutions" configuration field, configure the integer number of motor revolutions.

In this "Number of load revolutions" configuration field, configure the integer number of load revolutions.

Position parameters

In the "Leadscrew pitch" configuration field, configure the distance by which the load is moved when the leadscrew makes one revolution.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Axis type: Rotary (S7-1500)

Rotary - On motor shaft (S7-1500)

The encoder is connected to the motor shaft in a mechanically fixed manner. Motor and encoder form a unit.

Drive mechanism

Select the "Invert drive direction" check box if the direction of rotation of the drive is to be inverted.

Load gear

The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions.

In this "Number of motor revolutions" configuration field, configure the integer number of motor revolutions.

In this "Number of load revolutions" configuration field, configure the integer number of load revolutions.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Rotary - On load side (S7-1500)

The encoder is mechanically connected to the load side of the gear.

Drive mechanism

Select the "Invert drive direction" check box if the direction of rotation of the drive is to be inverted.

Load gear

The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions.

In this "Number of motor revolutions" configuration field, configure the integer number of motor revolutions.

In this "Number of load revolutions" configuration field, configure the integer number of load revolutions.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Rotary - External Measuring System (S7-1500)

An external measuring system provides the position values of the rotary load motion.

Distance per encoder revolution

In this configuration field, configure the linear load travel per encoder revolution.

Drive mechanism

Select the "Invert drive direction" check box if the direction of rotation of the drive is to be inverted.

Load gear

The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions.

In this "Number of motor revolutions" configuration field, configure the integer number of motor revolutions.

In this "Number of load revolutions" configuration field, configure the integer number of load revolutions.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Configuration - Dynamic Defaults (S7-1500)

In the "Dynamic default values" configuration window, configure the default values for velocity, acceleration, deceleration and jerk of the axis.

The default values take effect when values < 0 are specified in Motion Control instructions for the "Velocity", "Acceleration", "Deceleration" or "Jerk" parameters. The default values can be applied separately for each of the parameters just listed.

The default values for acceleration and deceleration also act on the traversing motions of active homing.

Velocity

In this field, define the default value for the velocity of the axis.

Acceleration/deceleration - Ramp-up time/ramp-down time

Configure the desired default value for acceleration in the "Ramp-up time" or "Acceleration" fields. The desired deceleration can be set in the "Ramp-down time" or "Deceleration" fields.

The following equations show the relationship between the ramp-up time and acceleration and the ramp-down time and deceleration:

$$\textit{Ramp-up time} = \frac{\textit{Velocity}}{\textit{Acceleration}}$$

$$\textit{Ramp-down time} = \frac{\textit{Velocity}}{\textit{Deceleration}}$$

Note

A change in the velocity influences the acceleration and deceleration values of the axis. The ramp-up and ramp-down times are retained.

Smoothing time/jerk

You can enter the jerk limit parameters in the "Smoothing time" box, or alternatively in the "Jerk" box:

- Set the desired jerk for the acceleration and deceleration ramp in the "Jerk" field. The value 0 means that jerk limiting is deactivated.
- Set the desired smoothing time for the acceleration ramp in the "Smoothing time" field.

Note

The jerk value is identical for the acceleration and deceleration ramp. The smoothing time in effect for the deceleration ramp results from the following relationships:

- **Acceleration > Deceleration**
  A shorter smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration < Deceleration**
  A longer smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration = Deceleration**
  The smoothing times of the acceleration and deceleration ramp are equal.

If an error occurs, the axis decelerates with the configured emergency stop deceleration. A configured jerk limit is not taken into account for this.

The following equations show the relationship between the smoothing times and the jerk:

$$\text{Smoothing time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Smoothing time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Motion jobs started in the user program are performed with the selected jerk.

See also

Positioning axis technology object (Page 1704)

Velocity profile (Page 1660)

Synchronous axis technology object (Page 1705)

Configuration - Emergency stop (S7-1500)

In the "Emergency stop" configuration window, you can configure the emergency stop deceleration of the axis. In the event of an error, and when disabling the axis, the axis is brought to a standstill with this deceleration using the Motion Control instruction "MC_Power" (input parameter "StopMode" = 0).

Emergency deceleration/emergency stop ramp-down time

Configure the deceleration value for emergency stop in the "Emergency stop deceleration" field or the "Emergency stop ramp-down time" field.

The relationship between emergency stop ramp-down time and emergency stop deceleration can be seen in the following equation:

$$\textit{Emergency stop ramp-down time} = \frac{\textit{Maximum velocity}}{\textit{Emergency stop deceleration}}$$

The configuration of the emergency stop deceleration is related to the configured maximum velocity of the axis. If the maximum velocity of the axis changes, then the value of the emergency deceleration also changes (the emergency stop ramp-down time remains unchanged).

See also

Positioning axis technology object (Page 1704)

Emergency stop deceleration (Page 1661)

Synchronous axis technology object (Page 1705)

Limits (S7-1500)

Configuration - Position limits (S7-1500)

Configure the hardware and software limit switches of the axis in the "Position limits" configuration window.

Enable HW limit switches

The check box activates the function of the negative and positive hardware limit switches. The negative hardware limit switch is located on the side in the negative direction of travel, and the positive hardware limit switch on the side in the positive direction of travel.

If a hardware limit switch is reached, technology alarm 531 is output, and the technology object is disabled (alarm response: remove enable).

Exception: If a hardware limit switch is overtraveled during an active home position approach with activated direction reversal at the hardware limit switch, the axis stops with the configured maximum deceleration and continues the home position approach in the opposite direction.

Note

Only use hardware limit switches that remain permanently switched after the approach. This switching state may only be canceled after the return to the permitted traversing range.

The digital inputs of the hardware limit switches are evaluated by default in cyclic data exchange. If the hardware limit switch is to be evaluated in the position control cycle of the drive, select the entry "MC-Servo" for "Organization block" and the entry "PIP OB Servo" for "Process image" in the input module settings under "I/O addresses".

Input of negative/positive HW limit switch

In these fields, select the PLC tag of the digital input for the negative and positive hardware limit switch.

In order to be able to select an input, a digital input module must have been added in the device configuration, and the PLC tag name for the digital input must be defined.

CAUTION

During installation of hardware limit switches, attention must be paid to the filter times of the digital inputs.

Based on the time for one position control cycle clock and the filter time of the digital inputs, the resulting delay times must be taken into account.

The filter time is configurable in individual digital input modules in the device configuration.

The digital inputs are set to a filter time of 6.4 ms by default. If these are used as hardware limit switches, undesired decelerations may occur. If this occurs, reduce the filter time for the relevant digital inputs.

The filter time can be set under "Input filter" in the device configuration of the digital inputs.

Level selection of negative/positive HW limit switch

Select the triggering signal level ("low level"/"high level") of the hardware limit switch in the drop-down list. With "Low level", the input signal is "FALSE" after the axis has reached or passed the hardware limit switch. With "High level", the input signal is "TRUE" after the axis has reached or passed the hardware limit switch.

Enable SW limit switches

This check box activates the high and low software limit switches. When software switches are activated, an active motion comes to a stop at the position of the software limit switch. The technological object signals an error. After acknowledgment of the error, the axis can again be moved in the direction of its operating range.

Note

Activated software limit switches act only on a homed axis.

Position of negative/positive SW limit switch

Configure the operating range of the axis with the positions of the negative and positive software limit switches.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Configuration - Dynamic limits (S7-1500)

In the "Dynamic limits" configuration window, configure the maximum values for velocity, acceleration, deceleration and jerk of the axis.

Maximum velocity

In this field, define the maximum permitted velocity of the axis.

Maximum acceleration/maximum deceleration - ramp-up time/ramp-down time

Set the desired acceleration in the "Ramp-up time" or "Acceleration" fields. The desired deceleration can be set in the "Ramp-down time" or "Deceleration" fields.

The following equations show the relationship between the ramp-up time and acceleration and the ramp-down time and deceleration:

$$\text{Ramp-up time} = \frac{\text{Maximum velocity}}{\text{Acceleration}}$$

$$\text{Ramp-down time} = \frac{\text{Maximum velocity}}{\text{Deceleration}}$$

Note

A change in the maximum velocity influences the acceleration and deceleration values of the axis. The ramp-up and ramp-down times are retained.

The "maximum deceleration" for active homing with change of direction at the hardware limit switch must be set sufficiently large, to brake the axis before reaching the mechanical endstop.

Smoothing time/jerk

You can enter the jerk limit parameters in the "Smoothing time" box, or alternatively in the "Jerk" box:

- Set the desired jerk for the acceleration and deceleration ramp in the "Jerk" field. The value 0 means that the jerk is not limited.
- Set the desired smoothing time for the acceleration ramp in the "Smoothing time" field.

Note

The configured smoothing time displayed in the configuration, applies only to the acceleration ramp.

If the values of the acceleration and deceleration differ, the smoothing time of the deceleration ramp is calculated and used according to the jerk of the acceleration ramp.

The smoothing time of the deceleration is adapted as follows:

- **Acceleration > Deceleration**
  A shorter smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration < Deceleration**
  A longer smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration = Deceleration**
  The smoothing times of the acceleration and deceleration ramp are equal.

If an error occurs, the axis decelerates with the configured emergency stop deceleration. A configured jerk limit is not taken into account for this.

The following equations show the relationship between the smoothing times and the jerk:

$$\text{Smoothing time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Smoothing time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Motion jobs started in the user program are performed with the selected jerk.

See also

Velocity profile (Page 1660)

Dynamic limits in synchronous operation (Page 1662)

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Configuration - Torque limits (S7-1500)

Configure the force/torque limiting of the drive in the "Torque limiting" configuration window.

The configuration is only available if a drive that supports force/torque limiting is selected and a telegram 10x is used. Telegram 101 cannot be used.

Effective

In the drop-down list, select whether the limit value is to be in effect "on load side" or "on motor side".

Torque limits

Enter a default value for the torque limiting in the specified unit of measurement in this field.

The default value is in effect when the torque limiting is specified using Motion Control instruction "MC_TorqueLimiting", input parameter "Limit" < 0.

Torque limiting applies to the following axis configurations:

- Axis type is "Rotary" and limit value is in effect "On load side" or "On motor side"
- Axis type is "Linear" and limit value is in effect "On motor side"

Force limit

Enter a default value for the force limit in the specified unit of measure in this field.

The default value is in effect when the force limit is specified using Motion Control instruction "MC_TorqueLimiting", input parameter "Limit" < 0.

The force limit applies to the following axis configuration: Axis type is "Linear" and limit value is in effect "On load side"

If the efficiency of the gear and leadscrew is crucial, you can set them in the "<TO>.Actor.Efficiency" tag.

Position-related monitoring

As a result of the force/torque limiting on the drive, a larger following error may occur or the axis standstill may not be detected reliably in positioning monitoring.

To deactivate the monitoring of the following error and the positioning monitoring during force/torque limiting, select the "Deactivate position-related monitoring" option. If you want to activate the position-related monitoring, select the option "Leave position-related monitoring enabled".

Interconnection in the SINAMICS drive

The following interconnection is required in the SINAMICS drive:

- P1522 to a fixed value of +100%
- P1523 to a fixed value of -100% (e.g. through interconnection to fixed value parameter P2902[i]).

See also

Force/torque limiting (Page 1665)

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Configuration - Fixed stop detection (S7-1500)

Configure the fixed stop detection in the configuration window.

A "Travel to fixed stop" can be realized by activating fixed stop detection using the Motion Control instruction "MC_TorqueLimiting" and a position-controlled motion job. The operation is also referred to as clamping.

Positioning tolerance

In this configuration field, configure the positioning tolerance that is regarded as a breaking away or turning back of the fixed stop when exceeded. To detect the breaking away or turning back of the fixed stop, the position setpoint must be located outside the positioning tolerance. The configured position tolerance must be less than the configured following error.

Following error

If the drive is stopped by a mechanical fixed stop during a motion job, the following error is increased. The accumulating following error serves as a criterion for fixed stop detection. In the "Following error" configuration field, configure the value of the following error starting from which the fixed stop detection is to take effect. The configured following error must be greater than the configured position tolerance.

Note

If the following error monitoring was activated in the position monitoring configuration, the "Maximum following error" configured there must be greater than the "Following error" of the fixed stop detection.

See also

Fixed stop detection (Page 1667)

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Homing (S7-1500)

Homing means matching the position value of a technology object to the real, physical location of the drive. Absolute target positions of the axis can only be approached with a homed axis.

Operating modes of the Motion Control instruction "MC_Home"

In S7-1500 Motion Control, the axis is homed with the Motion Control instruction "MC_Home". The following operating modes are used in the process:

- **Active homing (incremental encoder)**
  With active homing, the Motion Control instruction "MC_Home" performs the configured home position approach. Active traversing motions are aborted. When the homing mark is detected, the position of the axis is set according to the configuration.
- **Passive homing (incremental encoder)**
  With passive homing, the Motion Control instruction "MC_Home" instruction does not carry out any homing motion. The traversing motion required for this must be implemented by the user with other Motion Control instructions. Active traversing motions are not aborted upon start of passive homing. When the homing mark is detected, the axis is set according to the configuration.
- **Direct homing absolute (incremental encoder or absolute encoder)**
  The axis position is set without taking into consideration the home position switch. Active traversing motions are not aborted. The value of input parameter "Position" of Motion Control instruction "MC_Home" is set immediately as the actual position of the axis.
- **Direct homing (relative) (incremental encoder or absolute encoder)**
  The axis position is set without taking into consideration the home position switch. Active traversing motions are not aborted. The following statement applies to the axis position after homing:
  New axis position = Current axis position + Value of parameter "Position" of instruction "MC_Home".

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Homing (Page 1670)

Active homing (S7-1500)

Configuration - Active homing (S7-1500)

In the "Active Homing" configuration window, configure the parameters for active homing. "Active homing" is executed using the Motion Control instruction "MC_Home" with "Mode" = 3 and 5.

Note

Parameter "MC_Home.Mode" (S7-1500 CPU)

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Overview of versions (Page 1765).

Settings for

In the drop-down list, select the encoder for which the homing settings are to apply (only for S7-1500T).

Select the homing mode

Select from among the following homing modes:

- Use zero mark via PROFIdrive telegram (Page 1822)
- Use reference cam and zero mark via PROFIdrive telegram (Page 1823)
- Use homing mark via digital input (Page 1824)

See also

Homing (Page 1670)

Using multiple encoders (Page 1643)

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Homing mode "Use zero mark via PROFIdrive telegram" (S7-1500)

Enable direction reversal at the hardware limit switch

Select this check box to use the hardware limit switch as a reversing cam for the home position approach. After the axis has reached the hardware limit switch during active homing, it is ramped down at the configured maximum deceleration rate and the direction is then reversed. The zero mark is then sought in the reverse direction. If this function is not enabled and the axis reaches the hardware limit switch during active homing, then the drive is disabled and braked with the ramp configured in the drive.

Homing direction

Select the direction in which the next zero mark should be approached for homing.

"Positive" is the homing direction in the direction of positive position values; "negative" in the direction of negative position values.

Approach velocity

In this field, specify the velocity which is used to traverse to the home position offset.

Homing velocity

In this field, specify the velocity at which the axis approaches the zero mark for homing.

Home position offset

In the case of a differing zero mark position and home position, enter the corresponding home position offset in this field. The axis approaches the home position at approach velocity.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect when the Motion Control instruction "MC_Home" is executed with "Mode" = 5.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Homing (Page 1670)

Homing mode "Use reference cam and zero mark via PROFIdrive telegram" (S7-1500)

Enable direction reversal at the hardware limit switch

Select this check box to use the hardware limit switch as a reversing cam for the home position approach. After the axis has reached the hardware limit switch during active homing, it is ramped down at the configured maximum deceleration rate and the direction is then reversed. The reference cam is then searched for in the reverse direction. If this function is not enabled and the axis reaches the hardware limit switch during active homing, then the drive is disabled and braked with the ramp configured in the drive.

Approach direction

Select the approach direction for the reference cam search.

"Positive" is the approach direction in the direction of positive position values; "negative" in the direction of negative position values.

Homing direction

Select the direction in which the zero mark should be approached for homing.

Approach velocity

In this field, specify the velocity at which the reference cam is searched for during the homing procedure. Any configured home position offset is traversed at the same velocity.

Homing velocity

In this field, specify the velocity at which the axis approaches the zero mark for homing. For zero mark detection, the reference cam must be exited.

Home position offset

In the case of a differing zero mark position and home position, enter the corresponding home position offset in this field. The axis approaches the home position at approach velocity.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect when the Motion Control instruction "MC_Home" is executed with "Mode" = 5.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Homing (Page 1670)

Homing mode "Use homing mark via digital input" (S7-1500)

When a digital input is used as a homing mark, the accuracy of the homing process is not as high as for hardware-supported homing using zero marks.

You can improve the accuracy by using a low homing velocity.

Pay attention to the setting of short filter times for the digital input as well.

Digital input homing mark/cam

In this configuration field, select the PLC tag of the digital input that is to act as a homing mark (reference cam). Also select the level at which the homing mark is to be detected.

In order to be able to select an input, a digital input module must have been added in the device configuration, and the PLC tag name for the digital input must be defined.

Enable direction reversal at the hardware limit switch

Select this check box to use the hardware limit switch as a reversing cam for the home position approach. After the axis has reached the hardware limit switch during active homing, it is ramped down at the configured maximum deceleration rate and the direction is then reversed. The homing mark is then sensed in reverse direction. If this function is not enabled and the axis reaches the hardware limit switch during active homing, then the drive is disabled and braked with the ramp configured in the drive.

Approach direction

Select the approach direction for the homing mark search.

"Positive" is the approach direction in the direction of positive position values; "negative" in the direction of negative position values.

Homing direction

Select the direction in which the homing mark for homing is to be approached.

Homing mark

Select the switch position of the "digital input" that is to be used as the homing mark.

When a "digital input" is crossed, two switching edges that are spatially separated from one another are generated. The selection of the positive or negative side ensures that the homing mark is always evaluated at the same mechanical position.

The positive side is the switch position with a greater position value; the negative side is the switch position with the lesser position value.

The selection of the side is independent of the approach direction, and independent of whether it causes a rising or falling edge.

Approach velocity

In this field, specify the velocity at which the axis searches for the "digital input" during the home position approach. Any configured home position offset is traversed at the same velocity.

Homing velocity

In this field, specify the velocity at which the axis approaches the home position for homing.

Home position offset

If the homing mark position is different from the home position, enter the corresponding home position offset in this field. The axis approaches the home position at approach velocity.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect when the Motion Control instruction "MC_Home" is executed with "Mode" = 5.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Homing (Page 1670)

Passive homing (S7-1500)

Configuration - Passive homing (S7-1500)

Configure the parameters for passive homing in the "Passive Homing" (homing on the fly) configuration window. The "Passive homing" homing function is executed using the Motion Control instruction "MC_Home" with "Mode" = 2, 8 and 10.

Note

Parameter "MC_Home.Mode" (S7-1500 CPU)

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. For a comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0, refer to the section Overview of versions (Page 1765).

Settings for

In the drop-down list, select the encoder for which the homing settings are to apply (only for S7-1500T).

Select the homing mode

Select from among the following homing modes:

- Use zero mark via PROFIdrive telegram (Page 1826)
- Use reference cam and zero mark via PROFIdrive telegram (Page 1827)
- Use homing mark via digital input (Page 1828)

See also

Homing (Page 1670)

Using multiple encoders (Page 1643)

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Homing mode "Use zero mark via PROFIdrive telegram" (S7-1500)

Homing direction

Select the direction in which the next zero mark should be approached for homing. The following options are available:

- **Positive**
  The axis moves in the direction of higher position values.
- **Negative**
  The axis moves in the direction of lower position values.
- **Current**
  The currently effective approach direction is used for homing.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect when the Motion Control instruction "MC_Home" is executed with "Mode" = 10.

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Overview of versions (Page 1765).

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Homing (Page 1670)

Homing mode "Use reference cam and zero mark via PROFIdrive telegram" (S7-1500)

Homing direction

Select the direction in which the zero mark should be approached for homing. The next zero mark after leaving the reference cam is used.

The following options are available:

- Positive
  The axis moves in the direction of higher position values.
- Negative
  The axis moves in the direction of lower position values.
- Current
  The currently effective approach direction is used for homing.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect when the Motion Control instruction "MC_Home" is executed with "Mode" = 10.

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Overview of versions (Page 1765).

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Homing (Page 1670)

Homing mode "Use homing mark via digital input" (S7-1500)

Digital input homing mark/cam

In this dialog field, select a digital input that is to act as a homing mark (reference cam). Also select the level at which the homing mark is to be detected.

Homing direction

Select the direction in which the homing mark for homing is to be approached.

The following options are available:

- **Positive**
  The axis moves in the direction of higher position values.
- **Negative**
  The axis moves in the direction of lower position values.
- **Current**
  The currently effective approach direction is used for homing.

Homing mark

Select which switch position of the "digital input" is to be used as the homing mark.

When a "digital input" is crossed, two switching edges that are spatially separated from one another are generated. The selection of the positive or negative side ensures that the homing mark is always evaluated at the same mechanical position.

The positive side is the switch position with a greater position value; the negative side is the switch position with the lesser position value.

The selection of the side is independent of the approach direction, and independent of whether it causes a rising or falling edge.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect when the Motion Control instruction "MC_Home" is executed with "Mode" = 10.

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. For a comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0, refer to the section Overview of versions (Page 1765).

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Homing (Page 1670)

Position monitoring functions (S7-1500)

Configuration - Positioning monitoring (S7-1500)

In the "Positioning monitoring" configuration window, configure the criteria for monitoring the target position.

Positioning window

Configure the size of the positioning window in this field. If the axis is located within this window, the position is considered to be "reached".

Tolerance time

In this field, configure the tolerance time within which the position value must reach the positioning window.

Minimum dwell time in positioning window

Configure the minimum dwell time in this field. The current position value must be located in the positioning window for at least the "minimum dwell time".

If one of the criteria is violated, then the axis is stopped and the technology alarm 541 "Position monitoring error" is displayed (alarm response: Remove enable).

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Position monitoring functions (Page 1690)

Configuration - Following error (S7-1500)

In the "Following Error" configuration window, configure the permissible deviation of the actual position of the axis from the position setpoint. The following error can be dynamically adapted to the current velocity of the axis.

Enable following error monitoring

Select this check box, if you want to enable following error monitoring. When following error monitoring is enabled, the axis is stopped in the error range (orange). The technology alarm 521 "Following error" is displayed (alarm response: Remove enable).

When following error monitoring is disabled, the configured limits have no effect.

Maximum following error

Configure the following error that is permissible at maximum velocity in this field.

Warning level

In this field, configure a percentage of the current following error limit above which a warning should be output.

Example: The current maximum following error is 100 mm. The warning level is configured at 90%. If the current following error exceeds a value of 90 mm, the technology alarm 522 "Warning following error tolerance" is output. This is a warning and contains no alarm response.

Following error

In this field, configure the permissible following error for low velocities (without dynamic adjustment of the following error).

Start of dynamic adjustment

Configure the velocity starting from which the following error is to be dynamically adjusted in this field. Starting from this velocity, the following error up to the maximum velocity will be adjusted to the maximum following error.

See also

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Position monitoring functions (Page 1690)

Configuration - Standstill signal (S7-1500)

In the "Standstill signal" configuration window, configure the criteria for standstill detection.

Standstill window

Configure the size of the standstill window in this field. For standstill to be indicated, the velocity of the axis must be within this window.

Minimum dwell time in standstill window

Configure the minimum dwell time in the standstill window in this field. The velocity of the axis must be in the standstill window for at least the specified duration.

If both criteria are met, the standstill of the axis is indicated.

See also

Positioning axis technology object (Page 1704)

Standstill signal (Page 1692)

Synchronous axis technology object (Page 1705)

Configuration - Control loop (S7-1500)

In the "Control loop" configuration window, configure the precontrol and the gain Kv of the position control loop.

The Kv factor affects the following parameters:

- Positioning accuracy and stop control
- Uniformity of motion
- Positioning time

The better the mechanical conditions of the axis are (high stiffness), the higher the Kv factor can be configured. This reduces the following error, and a higher dynamic response is achieved.

For basic information, refer to the section Closed-loop control (Page 1693).

Precontrol

Configure the percentage velocity precontrol in this field.

Speed control loop substitute time

Configure the speed control loop substitute time in this field (T_{vtc}).

When speed precontrol is activated, the setpoint is delayed by the speed control loop substitute time before the control deviation is established. This prevents an overshoot or a leading of the actual value compared with the position setpoint. The speed control loop substitute time is a simplified substitute model of the dynamic behavior of the speed control loop. The speed control loop substitute time is included in the balancing filter.

Gain (Kv factor)

In the input field, enter the gain Kv of the position control loop.

Dynamic Servo Control (DSC)

For position-controlled axes (positioning axes/synchronous axes), the closed loop position control can occur either in the CPU or in the drive, provided the drive supports Dynamic Servo Control (DSC). Select your preferred control process:

- **Position control in the drive (DSC enabled)**
  With the Dynamic Servo Control (DSC) function, the position controller is executed in the drive in the cycle clock of the speed control loop. The setting of a significantly greater position controller gain factor Kv is thus enabled. This increases the dynamics for setpoint sequence and disturbance variable correction for highly dynamic drives.
- **Position control in the PLC**

Note

Dynamic Servo Control (DSC) is only possible with one of the following PROFIdrive telegrams:

- Standard telegram 5 or 6
- SIEMENS telegram 105 or 106

See also

Closed-loop control (Page 1693)

Control structure (Page 1694)

Function and structure of the optimization (Page 1956)

Positioning axis technology object (Page 1704)

Synchronous axis technology object (Page 1705)

Configuration - Actual value extrapolation

Configure the properties of the extrapolation for an actual value coupling for synchronous operation in the "Actual value extrapolation" configuration window. The values set here only apply when the actual values of this axis are used as leading value.

Position filter T1 and T2

Enter the time constants for the filtering of the actual position.

Velocity filter T1 and tolerance band width

Enter the time constant for the smoothing of the velocity and the tolerance band width.

For optimized application of the tolerance band, enter the same width for the tolerance band as the width of the noise signal.

Following-axis dependent extrapolation time

Enter the part caused by the following axis for calculation of the extrapolation time.

Hysteresis value

Enter a value for application of the hysteresis function to the extrapolated actual value. The specification is made in the configured length unit.

See also

Positioning axis technology object (Page 1704)

Extrapolation of the leading values for actual value coupling (Page 1707)

Synchronous axis technology object (Page 1705)

Configuring the technology object external encoder (S7-1500)

Configuration - Basic Parameters (S7-1500)

Configure the basic properties of the technology object in the "Basic Parameters" configuration window.

Name

Define the name of the external encoder in this field. The technology object is listed under this name in the project tree. The tags of the external encoder can be used in the user program under this name.

External encoder type

In this selection, configure whether the external encoder records linear or rotary motions.

Units of measure

In the drop-down list, select the desired units of measure for the position and velocity of the external encoder.

Modulo

Select the check box "Enable modulo", if you want to use a recurring measuring system for the external encoder (e.g. 0-360° for an external encoder of the "rotary" type).

- **Modulo start value**
  In this field, define the position at which the modulo range should begin (e.g. 0° for an external encoder of the "rotary" type).
- **Modulo length**
  In this field, define the length of the modulo range (e.g. 360° for an external encoder of the "rotary" type).

See also

Units of measure (Page 1632)

Modulo setting (Page 1633)

Technology object external encoder (Page 1720)

Hardware interface (S7-1500)

Configuration - Encoder (S7-1500)

The external encoder records the position of an externally controlled drive. The encoder required for this purpose communicates the encoder position to the controller by means of a PROFIdrive telegram. Configure the encoder and encoder type in the "Encoder" configuration window.

Data connection

In the drop-down list, select whether the data connection should be established directly with the encoder or via a data block that can be edited in the user program.

The selection is only possible for encoders that are connected via PROFIdrive and support parameter P979.

Encoder/data block

Select the encoder you have configured in the device configuration in this area.

The following encoders can be used:

- **Encoder on technology module (TM)**
  In the "Encoder" configuration field, select a previously configured technology module and the channel to be used. Only technology modules set to the "Position input for Motion Control" mode are displayed for selection.
  You can operate the technology module centrally on an S7-1500 CPU or decentrally on a distributed I/O. Isochronous mode is not possible with central operation in the CPU.
  You can identify the technology modules suitable for position detection for Motion Control in the documentation for the technology module and the catalog data.
- **Encoder via PROFINET/PROFIBUS (PROFIdrive)**
  In the " Encoder" configuration field, select a configured encoder on the PROFINET/ PROFIBUS.

  Note

  Option "Show all modules"

  If a PROFIdrive that has already been configured is not available for selection, use the option "Show all modules" to display all reachable modules.

  When you select the option "Show all modules", only the address range for each of the displayed modules is checked. If the address range of the module is large enough for the selected PROFIdrive frame, you can select the module. For this reason, make sure that you select a PROFIdrive encoder.

If you have selected "Data block" under the data connection, select a previously created data block which contains a tag structure of the data type "PD_TELx" ("x" stands for the telegram number to be used).

Encoder type

Select the encoder type in this configuration field. The following encoder types are available for selection:

- Incremental encoder
- Absolute encoder
- Absolute encoder cyclic absolute

See also

Technology object external encoder (Page 1720)

Data connection drive/encoder via data block (Page 1650)

Connect drive/encoder via data block (Page 1908)

Configuration - Data exchange (S7-1500)

Configuration - Data exchange (S7-1500)

Configure detailed encoder parameters and the data exchange with the encoder in the "Data exchange with encoder" configuration window. The configuration varies according to the encoder connection:

- Encoder to technology module (Page 1835)
- Encoder to PROFINET/PROFIBUS (Page 1838)

See also

Technology object external encoder (Page 1720)

Encoder to technology module (S7-1500)

Data exchange with encoder

In this area, you can configure the encoder telegram and the criteria for how the encoder data are to be evaluated. The specifications must match those in the device configuration.

Encoder telegram

In the drop-down list for the technology module, select the telegram that you have configured in the technology module.

Automatic data exchange for encoder values (offline)

Select the check box if you want to transfer the offline values of the encoder to the configuration of the technology object in the project.

Note

Automatic transfer of encoder parameters is only possible with PROFIdrive encoders as of product version A16. For this, "Hardware interface > Encoder" must be selected as the "Encoder" data connection in the configuration window.

Automatic data exchange for encoder values (online)

Select the check box if you want to transfer the effective values online in the encoder to the CPU during runtime. The encoder parameters are transferred from the bus after the (re-)initialization of the technology object and (re)start of the encoder or the CPU.

Note

Automatic transfer of encoder parameters is only possible with PROFIdrive encoders as of product version A16. For this, "Hardware interface > Encoder" must be selected as the "Encoder" data connection in the configuration window.

Encoder type

Depending on the selected encoder type, configure the following parameters: The specifications must match those in the device configuration.

Encoder type	Rotary incremental
Increments per revolution	Configure the number of increments that the encoder resolves per revolution in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.

Encoder type	Rotary absolute
Increments per revolution	Configure the number of increments that the encoder resolves per revolution in this field.
Number of revolutions	Configure the number of revolutions that the absolute encoder can detect in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

Encoder type	Rotary cyclic absolute
Increments per revolution	Configure the number of increments that the encoder resolves per revolution in this field.
Number of revolutions	Configure the number of revolutions that the absolute encoder can detect in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

Encoder type	Linear incremental
Distance between two increments	Configure the distance between two increments of the encoder in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.

Encoder type	Linear absolute
Distance between two increments	Configure the distance between two increments of the encoder in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

Encoder type	Linear cyclic absolute
Distance between two increments	Configure the distance between two increments of the encoder in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

See also

Configuration - Data exchange (Page 1834)

Encoder to PROFINET/PROFIBUS (Page 1838)

Technology object external encoder (Page 1720)

Encoder to PROFINET/PROFIBUS (S7-1500)

Data exchange with encoder

In this area, you can configure the encoder telegram and the criteria for how the encoder data are to be evaluated. The specifications must match those in the device configuration.

Encoder telegram

The telegram to the encoder that is set in the device configuration is preselected in the drop-down list.

Automatic data exchange for encoder values (offline)

Select the check box if you want to transfer the offline values of the encoder to the configuration of the technology object in the project.

Note

Automatic transfer of encoder parameters is only possible with PROFIdrive encoders as of product version A16. For this, "Hardware interface > Encoder" must be selected as the "Encoder" data connection in the configuration window.

Automatic data exchange for encoder values (online)

Select the check box if you want to transfer the effective values online in the encoder to the CPU during runtime. The encoder parameters are transferred from the bus after the (re-)initialization of the technology object and (re)start of the encoder or the CPU.

Note

Automatic transfer of encoder parameters is only possible with PROFIdrive encoders as of product version A16. For this, "Hardware interface > Encoder" must be selected as the "Encoder" data connection in the configuration window.

Encoder type

Depending on the selected encoder type, configure the following parameters:

Encoder type	Rotary incremental
Increments per revolution	Configure the number of increments that the encoder resolves per revolution in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.

Encoder type	Rotary absolute
Increments per revolution	Configure the number of increments that the encoder resolves per revolution in this field.
Number of revolutions	Configure the number of revolutions that the absolute encoder can detect in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

Encoder type	Rotary cyclic absolute
Increments per revolution	Configure the number of increments that the encoder resolves per revolution in this field.
Number of revolutions	Configure the number of revolutions that the absolute encoder can detect in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

Encoder type	Linear incremental
Distance between two increments	Configure the distance between two increments of the encoder in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.

Encoder type	Linear absolute
Distance between two increments	Configure the distance between two increments of the encoder in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

Encoder type	Linear cyclic absolute
Distance between two increments	Configure the distance between two increments of the encoder in this field.
Bits for fine resolution in the incremental actual value (Gx_XIST1)	Configure the number of bits for fine resolution within the incremental actual value (Gx_XIST1) in this field.
Bits for fine resolution in the absolute actual value (Gx_XIST2)	Configure the number of bits for fine resolution within the absolute actual value (Gx_XIST2) in this field.

See also

Configuration - Data exchange (Page 1834)

Encoder to technology module (Page 1834)

Technology object external encoder (Page 1720)

Extended Parameters (S7-1500)

Configuration - Mechanics (S7-1500)

Configuration - Mechanics (S7-1500)

Configure the encoder parameters for the position of the externally controlled drive in the "Mechanics" configuration window.

The configuration varies according to the type of encoder:

- Linear (Page 1840)
- Rotary (Page 1841)

See also

Technology object external encoder (Page 1720)

Mechanics (Page 1653)

Linear (S7-1500)

Encoder

If you would like to invert the actual value of the encoder, select the check box "Invert encoder direction".

Load gear

The gear ratio of the measuring gearbox is specified as the ratio between encoder revolutions and load revolutions.

In the "Number of encoder revolutions" configuration field, configure the integer number of encoder revolutions.

In this "Number of load revolutions" configuration field, configure the integer number of load revolutions.

If no load gear is present, select the same values for the number of encoder revolutions and load revolutions.

Position parameters

In the "Leadscrew pitch" configuration field, configure the distance by which the load is moved when the leadscrew makes one revolution.

See also

Configuration - Mechanics (Page 1839)

Rotary (Page 1841)

Technology object external encoder (Page 1720)

Mechanics (Page 1653)

Rotary (S7-1500)

Encoder

If you would like to invert the actual value of the encoder, select the check box "Invert encoder direction".

Load gear

The gear ratio of the measuring gearbox is specified as the ratio between encoder revolutions and load revolutions.

In the "Number of encoder revolutions" configuration field, configure the integer number of encoder revolutions.

In this "Number of load revolutions" configuration field, configure the integer number of load revolutions.

If no load gear is present, select the same values for the number of encoder revolutions and load revolutions.

See also

Configuration - Mechanics (Page 1839)

Linear (Page 1839)

Technology object external encoder (Page 1720)

Mechanics (Page 1653)

Homing (S7-1500)

Configuration - Homing (S7-1500)

Configure the parameters for homing the external encoder in the "Homing" configuration window. Homing is executed using the Motion Control instruction "MC_Home" with "Mode" = 2, 8 and 10.

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1765).

Select the homing mode

Select from among the following homing modes:

- Reference cam and zero mark via PROFIdrive telegram (Page 1842)
- Use zero mark via PROFIdrive telegram (Page 1843)
- Homing mark via digital input (Page 1844)

See also

Technology object external encoder (Page 1720)

Reference cam and zero mark via PROFIdrive telegram (S7-1500)

Homing direction

Select the direction in which the zero mark should be approached for homing. The next zero mark after leaving the reference cam is used.

The following options are available:

- **Positive**
  Axis moves in the direction of higher position values.
- **Negative**
  Axis moves in the direction of lower position values.
- **Current**
  The currently effective approach direction is used for homing.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect when the Motion Control instruction "MC_Home" is executed with "Mode" = 10.

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1765).

See also

Configuration - Homing (Page 1841)

Use zero mark via PROFIdrive telegram (Page 1843)

Homing mark via digital input (Page 1844)

Technology object external encoder (Page 1720)

Use zero mark via PROFIdrive telegram (S7-1500)

Homing direction

Select the direction in which the next zero mark should be approached for homing. The following options are available:

- **Positive**
  Axis moves in the direction of higher position values.
- **Negative**
  Axis moves in the direction of lower position values.
- **Current**
  The currently effective approach direction is used for homing.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect when the Motion Control instruction "MC_Home" is executed with "Mode" = 10.

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1765).

See also

Configuration - Homing (Page 1841)

Reference cam and zero mark via PROFIdrive telegram (Page 1841)

Homing mark via digital input (Page 1844)

Technology object external encoder (Page 1720)

Homing mark via digital input (S7-1500)

Digital input homing mark

In this configuration field, select a digital input that is to act as a homing mark (reference cam). Also select the level at which the homing mark is to be detected.

Homing direction

Select the direction in which the homing mark for homing is to be approached.

The following options are available:

- **Positive**
  Axis moves in the direction of higher position values.
- **Negative**
  Axis moves in the direction of lower position values.
- **Current**
  The currently effective approach direction is used for homing.

Homing mark

Select which switch position of the "digital input" is to be used as the homing mark.

When a "digital input" is crossed, two switching edges that are spatially separated from one another are generated. The selection of the positive or negative side ensures that the homing mark is always evaluated at the same mechanical position.

The positive side is the switch position with a greater position value; the negative side is the switch position with the lesser position value.

The selection of the side is independent of the approach direction, and independent of whether it causes a rising or falling edge.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect when the Motion Control instruction "MC_Home" is executed with "Mode" = 10.

Note

Parameter "MC_Home.Mode"

The "MC_Home.Mode" parameter for S7-1200 Motion Control and S7-1500 Motion Control has been standardized within the framework of technology version V2.0. This results in a new assignment of the parameter values for the "MC_Home.Mode" parameter. A comparison of the "MC_Home.Mode" parameter for technology versions V1.0 and V2.0 is available in the section Version overview (Page 1765).

See also

Configuration - Homing (Page 1841)

Use zero mark via PROFIdrive telegram (Page 1842)

Reference cam and zero mark via PROFIdrive telegram (Page 1841)

Technology object external encoder (Page 1720)

Configuration - Actual value extrapolation

Configure the properties of the extrapolation for an actual value coupling for synchronous operation in the "Actual value extrapolation" configuration window. The values set here only apply when the actual values of this encoder are used as leading value.

Position filter T1 and T2

Enter the time constants for the filtering of the actual position.

Velocity filter T1 and tolerance band width

Enter the time constant for the smoothing of the velocity and the tolerance band width.

For optimized application of the tolerance band, enter the same width for the tolerance band as the width of the noise signal.

Following-axis dependent extrapolation time

Enter the part caused by the following axis for calculation of the extrapolation time.

Hysteresis value

Enter a value for application of the hysteresis function to the extrapolated actual value. The specification is made in the configured length unit.

See also

Technology object external encoder (Page 1720)

Extrapolation of the leading values for actual value coupling (Page 1707)

Configuring the technology object measuring input (S7-1500)

Configuration - Basic Parameters (S7-1500)

Configure the basic properties of the technology object in the "Basic Parameters" configuration window.

Name

Define the name of the measuring input in this field. The technology object is listed under this name in the project tree. The tags of the measuring input can be used in the user program under this name.

Assigned axis or external encoder

The axis or external encoder assigned to the measuring input is displayed. You can use the button to directly access the configuration of the basic parameters of the higher-level technology object.

Unit of measure

The indicated unit of measure for the position of the measuring input corresponds to the unit of measure of the higher-level technology object.

See also

Units of measure (Page 1632)

Technology object measuring input (Page 1721)

Configuration - Hardware interface (S7-1500)

Measuring input type

Select the measuring input type.

- Measuring using TM Timer DIDQ
- Measurement via SINAMICS measuring input
- Measurement using PROFIdrive telegram (drive or external encoder)

Measurement using TM Timer DIDQ

Select a measurement input for a measurement using a Timer DI. The selection box shows all channels that are configured correctly.

Measurement via SINAMICS measuring input

Select a measuring input for a measurement via SINAMICS measurement sensing input. The selection box shows all compatible telegram types. You are shown all terminals that can potentially be used as measuring inputs. For the assignment, note that the inputs of must be assigned to the measuring inputs (P680) in the telegram without gaps and in ascending order.

With P728.8 to P728.15, you configure all DI/DOs used as a measurement input on the control unit. Use P680 of the control unit to specify the terminals for the global measuring inputs.

Measurement using PROFIdrive telegram (drive or external encoder)

For a measurement via PROFIdrive telegram, select the number of the measuring input in the telegram in the "Number of the measuring input" drop-down list. The input field is preset with the value "1".

Two communication channels are available for the transmission of measured values in the PROFIdrive telegram. These communication channels are assigned to one measurement input/digital input each in the drive. Use the PROFIdrive parameters to configure the digital input on the drive that is to be used for the configured communication channel.

- Measurement input for the first communication channel
  ("<TO>.Parameter.PROFIdriveProbeNumber" = 1)
  If you use two encoders, you must select the associated DI in the SINAMICS for each encoder. Various results are then transferred to the technology object depending on the selected encoder. The sensors are configured using the parameters P488[0] and P488[1].
- Measurement input for the second communication channel
  ("<TO>.Parameter.PROFIdriveProbeNumber" = 2)
  If you use two encoders, you must select the associated DI in the SINAMICS for each encoder. Various results are then transferred to the technology object depending on the selected encoder. The sensors are configured using the parameters P489[0] and P489[1].

Correction time for the measuring signal

Specify a correction time if possible delay times in the measurement signal are to be compensated.

See also

Automatic transfer of drive and encoder parameters in the device (Page 1641)

Technology object measuring input (Page 1721)

Configuring technology modules for Motion Control (Page 1893)

Configuration - Extended parameters (S7-1500)

Adjustment for activation time of the measuring range

To adjust the activation time defined on the system side, enter an additional activation time here.

The configuration window also displays the following times calculated on the system side:

- Time after the measuring job request until the measuring event can be recoreded
- Time after the measuring event until the measurement result is displayed (for measuring of one or two edges)

See also

Measuring with measuring range (Page 1728)

Time-related boundary conditions (Page 1730)

Technology object measuring input (Page 1721)

Configuring the output cam technology object (S7-1500)

Configuration - Basic Parameters (S7-1500)

Configure the basic properties of the technology object in the "Basic Parameters" configuration window.

Name

Define the name of the output cam in this field. The technology object is listed under this name in the project tree. The tags of the output cam can be used in the user program under this name.

Assigned axis or external encoder

The axis or external encoder assigned to the output cam is displayed. You can use the button to directly access the configuration of the basic parameters of the higher-level technology object.

Output cam type

Select based on the desired switching behavior of an output cam type:

- Distance output cam (position-dependent switch-on/switch-off)
- Time-based output cam (position-dependent switch-on and position-independent or time-dependent switch-off)

Output cam reference

Configure in this selection whether the switching points of the output cam are to reference the actual position or the position setpoint.

Unit of measure

The indicated unit of measure for the position of the output cam corresponds to the unit of measure of the higher-level technology object.

When a time-based output cam is selected as the output cam type, the unit of measure for the switch-on duration and other times is also indicated. For output cams, this is always ms.

See also

Units of measure (Page 1632)

Technology object output cam (Page 1733)

Configuration - Hardware interface (S7-1500)

Output cam output

Select whether the generated switching signals are to be output at the digital output.

- **Activate output**
  Select one of the following two output options for the output cam output:
  - **Output by TM Timer DIDQ**
    For output by TM Timer DIDQ, select a previously configured technology module and the channel to be used in the "Output" field.
    If no technology module is available for selection, change to the device configuration and add a technology module.
  - **Output by digital output module**
    For output by a digital output module, select this in the "Cam output" field. Only the digital outputs with previously defined PLC tags are displayed for selection.

  Select the logical operation of the output cam signal at the output. The logic operation relates to the last signal to be output after the set inversion, if any.
  All output cams that use the selected output are shown graphically.
- **Output deactivated**
  When the output is deactivated, the output cam is evaluated only in the software.

See also

Technology object output cam (Page 1733)

Configuring technology modules for Motion Control (Page 1893)

Extended parameters (S7-1500)

Configuration - Activation time (S7-1500)

The specified output cam type is indicated in the upper area of the "Activation time" configuration window.

Activation time and deactivation time

For a time shift of the switch-on and switch-off times of an output cam, enter an activation time and a deactivation time.

See also

Compensation of actuator switching times (Page 1744)

Technology object output cam (Page 1733)

Configuration - Hysteresis (S7-1500)

To prevent unwanted changes in the switching state of the output cams of a cam track, enter a hysteresis value.

When using an output cam with reference to actual position, always enter a hysteresis value (> 0.0).

See also

Hysteresis (Page 1742)

Technology object output cam (Page 1733)

Configuring the cam track technology object (S7-1500)

Configuration - Basic Parameters (S7-1500)

Configure the basic properties of the technology object in the "Basic Parameters" configuration window.

Name

Define the name of the cam track in this field. The technology object is listed under this name in the project tree. The tags of the cam track can be used in the user program under this name.

Assigned axis or external encoder

The axis or external encoder assigned to the cam track is displayed. You can use the button to directly access the configuration of the basic parameters of the higher-level technology object.

Output cam type

Select based on the desired switching behavior of an output cam type for the cam track:

- Distance output cam (position-dependent switch-on/switch-off)
- Time-based output cam (position-dependent switch-on and position-independent or time-dependent switch-off)

Output cam reference

In this selection, configure whether the switching points of the cam track are to reference the actual position or the position setpoint.

Unit of measure

The indicated unit of measure for the position of the cam track corresponds to the unit of measure of the higher-level technology object.

When a time-based output cam is selected as the output cam type, the unit of measure for the switch-on duration and other times is also indicated. For output cams, this is always ms.

See also

Units of measure (Page 1632)

Cam track technology object (Page 1747)

Configuration - Hardware interface (S7-1500)

Output cam track

Select whether the generated switching signals are to be output at the digital output.

- **Activate output**
  Select one of the following two output options for the output track:
  - Output by TM Timer DIDQ
    For output by TM Timer DIDQ, select a previously configured technology module and the channel to be used in the "Output" field.
    If no technology module is available for selection, change to the device configuration and add a technology module.
  - Output by digital output module
    For output by a digital output module, select this in the "Cam output" field. Only the digital outputs with previously defined PLC tags are displayed for selection.
- **Deactivate output**
  When the output is deactivated, the cam track is evaluated only in the software.

See also

Cam track technology object (Page 1747)

Configuring technology modules for Motion Control (Page 1893)

Extended parameters (S7-1500)

Track data (S7-1500)

Configuration - Activation time (S7-1500)

The set output cam type is displayed.

Activation time and deactivation time

Enter the activation time and the deactivation time.

For a time shift of the switch-on and switch-off times of the output cam of a cam track, enter an activation time and a deactivation time.

See also

Time offset of cam switching points (Page 1758)

Cam track technology object (Page 1747)

Configuration - Hysteresis (S7-1500)

To prevent unwanted changes in the switching state of the output cams of a cam track, enter a hysteresis value.

When using an output cam with reference to actual position, always enter a hysteresis value (> 0.0).

See also

Hysteresis (Page 1758)

Cam track technology object (Page 1747)

Configuration - Track dimensions (S7-1500)

Track length

Enter the corresponding track length.

Also take into account the output cam data of the individual output cams when defining the track length. Output cams whose start position lies outside the cam track length are not included. They become active only if the cam track length is increased so that at least the respective start position of an output cam is within the new track length.

Axis reference position

Enter the position of an axis or external encoder starting from which the output of the cam track is to occur. The start of the cam track is placed at the entered position.

You can enter a negative or positive value for the reference position.

Modulo length of the axis

When an axis with modulo function is used, the modulo length of the axis is displayed.

See also

Cam track technology object (Page 1747)

Configuration - Output cam data (S7-1500)

The set output cam type is displayed.

Enter the properties for the output cams of the cam track that are to be output. You can set up to 32 individual output cams on a cam track.

Also take into account any previously defined track length when defining the output cam data. Output cams whose start position lies outside the cam track length are not included. They become active only if the cam track length is increased so that at least the respective start position of an output cam is within the new track length.

The input options described below are displayed in the Output cam data configuration window according to the configured output cam type.

- Valid
  Only output cams set as "valid" are output and have a status display.
- Start position
  - The start position may not be greater than the end position for distance output cams.
  - If the start position is equal to the end position, the distance output cam does not switch.
  - The switching ranges of individual output cams are permitted to overlap.
- End position
  - The "End position" column is only displayed when distance output cam is set for the output cam type.
  - The end position must not be less than the start position.
- Switch-on duration
  The "Switch-on duration" column is only displayed when time-based output cam is set for the cam type.

See also

Cam track technology object (Page 1747)

Configuring the cam technology object

You configure the cam technology object (Page 1760) with an editor.

You create the cam using a diagram, a table containing the elements of the curve and the properties of the elements. Transitions are calculated between the individual elements of the cam (e.g. points, lines, polynomials). The curve reflects the path-related dependency between the leading axis (leading values, abscissa in the chart) and following axis (following values, ordinate in the chart).

The following figure shows the structure of the editor.

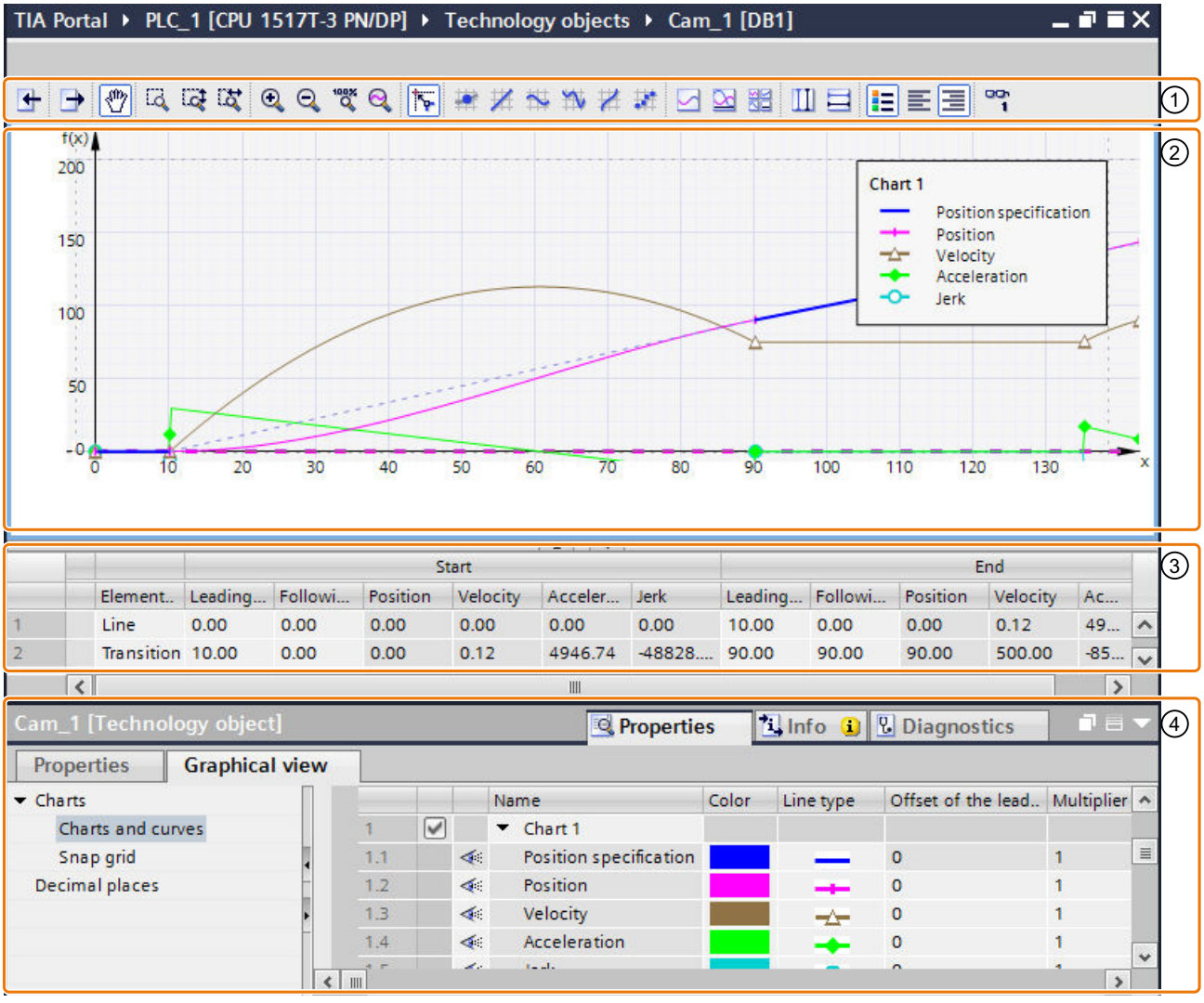

① Toolbar

② Graphical editor

The leading value range (definition range) is displayed on the abscissa (x axis).

The following value range (value range) is displayed on the ordinate (y axis).

③ Tabular editor

④ Properties (Inspector window)

Toolbar

You use the toolbar to operate the graphical editor and to import/export cams.

Graphical editor

In the graphical editor, you edit the elements of the curve graphically. The elements can be added, edited and deleted. Up to four charts can be created one above the other with synchronized abscissa. The setpoint curve as well as the curves for the effective position, velocity, acceleration and jerk can be displayed in the charts.

Tabular editor

All elements of the curve are listed in the tabular editor. Existing elements can be edited. New elements can be added.

Properties (Inspector window)

In the Inspector window, you configure the properties of the trend and of the selected element in the "Properties" tab, and the graphical view in the "Display" tab:

- Profile (e.g. leading and following value range, optimization and interpolation of the profile, number of elements used)
- Element (e.g. derivatives, polynomial coefficients, optimization of the element)
- Graphical view (e.g. line type, line color, scaling of the view)

Elements of the curve

The following table shows the elements that can be used to define the curve:

Element	Description
Point	A point assigns a following value to a leading value. The curve runs through the point with these coordinates. The velocity, acceleration and jerk can be defined in this point using the first, second and third derivative.
Point group	A point group combines two or more points into an commonly interpolated element and allows precise interpolation between these points.
Line	A line describes a motion with constant velocity from the start point of the line to the end point. The incline of the line specifies the constant velocity.
Sine	A sine element describes a motion according to the sine function. The sine function can be adjusted with the phase angle in the start point and end point, the period length, the amplitude as well as the oscillation zero point (offset).
Polynomial	A polynomial describes a motion according to a polynomial function of the 7th degree maximum. Polynomials can be defined by entering the boundary conditions or the polynomial coefficients. Optionally, you can configure a trigonometric polynomial component.
Inverse sine (approximated)	An inverse sine describes a motion according to the arcsine function. An inverse sine is approximated using interpolation points of the arcsine function.
Transition	Transitions interpolate the range between two elements. The ranges are automatically interpolated by the controller or using a configurable optimization according to VDI Guideline 2143. Transitions are added automatically.

Operating the cam editor

The procedure described here shows the basic operation of the cam editor. This procedure serves as a recommendation.

The basic operation can include the follow tasks:

- Adapting defaults
- Creating and adapting the curve
- Interpolation/optimization of the transitions

Adapting defaults

To adjust the leading and following value range of the cam profile as well as the graphical view, follow these steps:

1. In the properties (Inspector window), open the "Profile > General (Page 1870)" configuration window.
2. Configure the leading value range and the following value range of the curve definition. The graphical view is automatically adapted to the inputs.
3. In the area navigation of the Inspector window, open the "Display (Page 1888)" tab.
4. Configure the configuration windows:
   - The display of the charts and curves
   - The grid spacing for aligning inputs in the graphical editor
   - The decimal places displayed in the cam editor.

Creating and adapting the curve

To create and adapt the curve, follow these steps:

1. Use the graphical editor and/or the tabular editor to add the elements of the cam:
   - Select the tool required for inserting the respective element in the toolbar. Place the element at the required position in the graphical editor.
   - Use <Add> to insert the corresponding elements in the "Element type" column of the tabular editor. Adjust the position of the elements using the start and end values.

   Transitions between the elements are added automatically.
2. To edit an element, select it in the graphical or tabular editor.
   The element is highlighted in the graphical and in the tabular editor. The "Element > Parameter/Characteristic" configuration window is displayed in the properties (Inspector window).
3. The elements can be adjusted as follows:
   - Move the element or the drag handles of the element in the graphical editor.
   - Adjust the start and end values in the tabular editor.
   - Configure additional element-specific parameters in the properties (Inspector window) in the "Element > Parameter (Page 1876)" configuration window.
   - Set the interpolation of the transitions with the properties (Inspector window).

The number of elements used is displayed in the properties (Inspector window) in the "Profile > Statistics (Page 1874)" properties window.

Setting the interpolation of the transitions (system interpolation)

The interpolation (Page 1762) can be set separately for each transition. The default for interpolation of the transitions is the system interpolation. You configure the system interpolation for all transitions in the properties (Inspector window) in the "Profile > System interpolation (Page 1872)" configuration window.

Setting the optimization of the transitions (VDI Guideline 2143)

Each transition can also be adapted separately according to the VDI Guideline 2143. In doing so, the settings in the properties (Inspector window) in the "Profile > Default optimization settings (Page 1871)" configuration window are taken into consideration.

To adapt the optimization of a transition according to the VDI Guideline 2143, follow these steps:

1. Select the transition in the graphical or tabular editor.
2. In the properties (Inspector window), open the "Element > Characteristic (Page 1884)" configuration window.
3. Select the optimization method "VDI-based optimization" in the "Optimization method" drop-down list.
4. If necessary, change the default settings.
   The selection of the parameters is automatically limited to the settings that can be applied according to VDI Guideline 2143.

The optimization of the transitions according to VDI guidelines consumes additional points and/or segments (Page 1874) in the cam.

See also

Configuration charts - Charts and curves (Page 1888)

Graphical editor

Structure of the graphical editor

The graphical editor is divided into the following areas:

- Toolbar
- Curve diagram

Toolbar

The toolbar at the top of the graphical editor provides you with buttons for the following functions:

Button	Function	Description
	Importing cam from file	See section Importing/exporting cam (Page 1890)
	Exporting cam to file	See section Importing/exporting cam (Page 1890)
	Edit elements/Move view	• Selecting and moving of individual elements and element • Moving the view using drag-and-drop To switch from any tool to the "Edit elements/Move view" tool, press the <Esc> key.
	Activate zoom selection	Zoom into selected area
	Activate vertical zoom	Vertical zoom into selected area without horizontal scaling Alternative: <Ctrl> +drag to ordinate keeping mouse button pressed
	Activate horizontal zoom	Horizontal zoom into selected area without vertical scaling Alternative: <Ctrl> +drag to abscissa keeping mouse button pressed
	Zoom in	Enlargement of the display Alternative: <Ctrl> + mouse wheel up in curve diagram
	Zoom out	Reduction of the display Alternative: <Ctrl> + mouse wheel down in curve diagram
	Show all	Display of entire definition and value range
	Zoom into curve	Zoom to the following value range of the curve that you selected in the legend of the chart
	Activate snap grid	Inputs and element end points are aligned to the configurable snap grid and to other element end points.
	Inserting a point	Adding a point to the chart

Button	Function	Description
	Inserting a line	Adding a line to the chart
	Inserting a sine	Adding a sine element to the chart
	Inserting a polynomial	Adding a polynomial to the chart
	Inserting an inverse sine	Adding an inverse sine to the chart
	Insert point group	Add a point group to the chart
	View: A chart with positions	Display of one chart with the following curves of the cam opened in the editor: • Preset curve • Effective position
	View: A chart with all curves	Display of one chart with the following curves of the cam opened in the editor: • Preset curve • Effective position • Effective velocity • Effective acceleration • Effective jerk
	View: Four charts with all curves	Display of four charts with the following curves of the cam opened in the editor: • Chart with setpoint curve and effective position • Chart with effective velocity • Chart with effective acceleration • Chart with effective jerk
	Vertical measuring lines	Displaying and moving of vertical measuring lines Hold down the left mouse button and drag to draw a measuring range. The vertical position of the measuring lines can be moved. The function values for the measuring line positions are displayed in the chart. The difference of the measuring lines is displayed between the measuring lines.
	Horizontal measuring lines	Displaying and moving of horizontal measuring lines Hold down the left mouse button and drag to draw a measuring range. The horizontal position of the measuring lines can be moved. The function values for the measuring line positions are displayed in the chart. The difference of the measuring lines is displayed between the measuring lines.
	Show legend	Showing or hiding of the legend in the curve diagram. To display values for a specific curve on the ordinate, click on the name of the corresponding curve in the legend.
	Show legend left	Display of the legend on the left side of the curve diagram.
	Show legend right	Display of the legend on the right side of the curve diagram.
	Read out and display online curve one time	Display of the position values of the cam read back from the CPU (orange) The cam editor reads out the cam that was already loaded into the CPU. The read out "Online curve" is displayed in the graphical editor.

Curve diagram

In the curve diagram, you enter the elements of the curve and adjust the curve by selecting and moving elements.

Chart areas outside of the leading value/following value range configured in "Profile > General (Page 1870)" are grayed out. Elements outside the leading value / following value range are displayed with a warning ("Element is outside the definition range").

You can display various curves (position, velocity, acceleration and jerk) one above the other in up to four charts by configuring the graphical view accordingly. When multiple charts are displayed, you can adapt the graphs to match the separator lines.

The view can be zoomed in the manual mode by pressing <Ctrl > + Mouse wheel and <Ctrl > + while pressing the mouse button on the abscissa/ordinate.

The editor shows messages for checking the entered curve via warning triangles . The tooltip of the waring triangle shows the message text. Configure the checking of the curve in the "Check (Page 1873)" configuration window.

Display of the online curve

When you click the button, the cam editor reads the data from the technology object data block and displays the curve in the graphical editor:

Cam status	Interpolation status	
Data modified (CamDataChanged = 0)	Not interpolated (Interpolated = 0)	Only the points and segments of the cam are displayed.
	Interpolated (Interpolated = 1)	The entire cam is displayed.
Data modified (CamDataChanged = 1)	Not interpolated (Interpolated = 0)	Only the points and segments of the cam are displayed.
	Interpolated (Interpolated = 1)	Only the points and segments of the cam are displayed and interpolated.

Inserting a point

A point assigns a following value to a leading value. The curve runs through the point with these coordinates.

By means of the first, second and third derivatives, the velocity, acceleration and jerk can be defined in this point. The derivations are taken into consideration during VDI-based optimization of transitions of the points to other elements.

Inserting a point

To add a point to the curve, follow these steps:

1. Select the "Insert point" tool in the toolbar.
2. Click on the position in chart 1 where you want to insert the point.
   The point is inserted. The coordinates are displayed for the point. The tabular editor and the view of the properties (Inspector window) are updated. A transition to any element is inserted automatically.

Moving a point

To move a point in the graphical editor, follow these steps:

1. Select the "Edit elements/Move view" tool in the toolbar.
2. Select the point in chart 1.
3. Use drag-and-drop to move the point to the desired position.

Adapting parameters

The parameters of the point can be adjusted in the tabular editor as well as in the properties (Inspector window) under "Element > Parameter (Page 1876)".

Insert point group

A point group combines two or more points into an commonly interpolated element and allows precise interpolation between the points.

Insert point group

To add a point group to the trend, proceed as follows:

1. Select the "Insert point group" tool in the toolbar.
2. In Chart 1, click on the position at which you want to insert the point group.
   The point group is inserted. The coordinates of the start point and the end point are displayed at the point group. The tabular editor and the view of the properties (Inspector window) are updated. If a different element already exists, a transition to the existing element is automatically inserted.

Adapt point group

To adapt a point group in the graphical editor, proceed as follows:

1. Select the "Edit elements/Move view" tool in the toolbar.
2. Select the point group in Chart 1.
   The point group is highlighted graphically with drag handles. The following drag handles are displayed:
   – Start value of the point group
   – End value of the point group
3. Drag-and-drop the drag handles or the whole point group to the desired position.
   If further interpolation points are configured between the start point and the end point in the point group, the cam editor handles the interpolation points as follows:
   – Definition type of the leading value "Relative to the segment"
     The interpolation points are shifted relative to the start and end points.
   – Definition type of the leading value "Absolute in the profile"
     The interpolation points are not moved.

Adapting parameters

The parameters of the point group can be adapted in the graphical editor, in the tabular editor as well as in the properties (Inspector window) under "Element > Parameter (Page 1877)".

Inserting a line

A line describes a motion with constant velocity from the start point of the line to the end point. The incline of the line specifies the constant velocity.

Inserting a line

To add a line to the curve, follow these steps:

1. Select the "Insert line" tool in the toolbar.
2. Use drag-and-drop in chart 1 to draw the line from the start position to the end position. The line is inserted. The coordinates of the start point and end point are displayed for the line. The tabular editor and the view of the properties (Inspector window) are updated. A transition to any element is inserted automatically.

Moving a line

To move a line in the graphical editor, follow these steps:

1. Select the "Edit elements/Move view" tool in the toolbar.
2. Select the line in chart 1.
   The line is graphically highlighted with drag handles. The following drag handles are displayed:
   – Start point of the line
   – End point of the line
3. Use drag-and-drop to move the drag handles or the entire line to the desired position.

Adapting parameters

The parameters of the line can be adjusted in the graphical editor, in the tabular editor as well as in the properties (Inspector window) under "Element > Parameter (Page 1879)".

Inserting a sine

A sine element describes a motion according to the sine function. The sine function can be adjusted with the phase angle in the start point and end point, the period length, the amplitude as well as the oscillation zero point (offset).

Inserting a sine

To add a sine to the curve, follow these steps:

1. Select the "Insert sine" tool in the toolbar.
2. Click on the position in chart 1 where you want to insert the sine. The mouse pointer points to the start position of the sine here.
   The sine is inserted. The coordinates of the start point and end point are displayed for the sine. The tabular editor and the view of the properties (Inspector window) are updated. A transition to any element is inserted automatically.

Adjusting a sine

To adjust a sine in the graphical editor, follow these steps:

1. Select the "Edit elements/Move view" tool in the toolbar.
2. Select the sine in chart 1.
   The sine is graphically highlighted with drag handles and guide lines for the zero line and the amplitude. The following drag handles are displayed:
   – Leading value/shifting at left/right boundary
     These drag handles can also be used to adjust the inclination of an inclined sine.
   – Leading value at left/right boundary
   – Phase at left/right boundary
   – Amplitude
3. Use drag-and-drop to move the drag handles or the entire sine to the desired position.

Adapting parameters

The parameters of the sine can be adjusted in the graphical editor, in the tabular editor as well as in the properties (Inspector window) under "Element > Parameter (Page 1880)".

Inserting a polynomial

A polynomial describes a motion according to a polynomial function of the 6th degree maximum. Polynomials can be defined by entering the boundary conditions or the polynomial coefficients. Optionally, you can configure a trigonometric polynomial component.

Inserting a polynomial

To add a polynomial to the curve, follow these steps:

1. Select the "Insert polynomial" tool in the toolbar.
2. Click on the position in chart 1 where you want to insert the polynomial. In so doing, the mouse pointer points to the start position of the polynomial.
   The polynomial is inserted. The coordinates of the start point and end point are displayed for the polynomial. The tabular editor and the view of the properties (Inspector window) are updated. If a different element already exists, a transition to the existing element is automatically inserted.

Adjusting a polynomial

To adjust a polynomial in the graphical editor, follow these steps:

1. Select the "Edit elements/Move view" tool in the toolbar.
2. Select the polynomial in chart 1.
   The polynomial is graphically highlighted with drag handles. The following drag handles are displayed:
   – Leading value/following value at left/right boundary
   – Position of point of inflection
3. Use drag-and-drop to move the drag handles or the entire sine to the desired position.

Adapting parameters

The parameters of the polynomial can be adapted in the graphical editor, in the tabular editor as well as in the properties (Inspector window) under "Element > Parameter (Page 1881)".

Inserting an inverse sine

An inverse sine describes a motion according to the arcsine function. The arcsine function is the inverse function of the sine function. An inverse sine is approximated using interpolation points of the arcsine function.

Inserting an inverse sine

To add an inverse sine to the curve, follow these steps:

1. Select the "Insert inverse sine" tool in the toolbar.
2. Click on the position in chart 1 where you want to insert the inverse sine. In so doing, the mouse pointer points to the start position of the inverse sine.
   The sine is inserted. The coordinates are displayed for the point. The tabular editor and the view of the properties (Inspector window) are updated. A transition to any element is inserted automatically.

Adjusting an inverse sine

To adjust an inverse sine in the graphical editor, follow these steps:

1. Select the "Edit elements/Move view" tool in the toolbar.
2. Select the inverse sine in chart 1.
   The inverse sine is graphically highlighted with drag handles. The following drag handles are displayed:
   – Start point of the inverse sine
   – End point of the inverse sine
3. Use drag-and-drop to move the drag handles or the entire inverse sine to the desired position.

Adapting parameters

The parameters of the inverse sine can be adjusted in the graphical editor, in the tabular editor as well as in the properties (Inspector window) under "Element > Parameter (Page 1883)".

Deleting an element

To delete an element in the graphical editor, follow these steps:

1. Select the element.
2. Press the <Del> key.
   The element is deleted. The graphical editor and the view of the properties (Inspector window) are updated. A transition to any element present is also deleted.

Shortcut menu in the graphical editor

The following table shows the functions in the shortcut menu of the graphical editor:

Function	Description
Show all	Display of entire definition and value range
Zoom into curve	Displays the curve selected in the legend of the chart
Zoom in	Enlargement of the display
Zoom out	Reduction of the display
Cut	Removing the selected elements and copying them to the clipboard
Copy	Copying of the selected elements to the clipboard
Paste	Pasting of the elements from the clipboard to the last element
Delete	Deletion of the selected elements Transitions to existing elements are also deleted.
Paste special	Call of the "Paste elements (Page 1892)" dialog
Group points	Combine the selected points into a group of points The entry is displayed under the following conditions: • Only points are selected in the graphic/tabular editor. • There are no other elements between the selected points.
Dissolve point group	Ungroups the selected point group into individual points
Show/hide measuring point labels	Showing or hiding the measuring points The entry is displayed under the following conditions: • Measuring lines are displayed. • Measuring points are hidden/shown.
Move	Call of the "Move elements (Page 1892)" dialog
Scale	Call of the "Scale elements (Page 1892)" dialog

See also

Dialogs in the shortcut menu (Page 1892)

Tabular editor

Structure of the tabular editor

The tabular editor shows all elements of the curve, sorted by their leading values. The elements can be adjusted. New elements can be added.

The following properties are displayed in the corresponding column for each element of the curve:

Column/Property	Description
First column	Sequential number of the element
Second column	Display of calculation problems that might occur with warning triangle The alarm text is displayed in the tooltip of the warning triangle.
Element type	• Display/change of element type • Adding elements Possible element types: • Point • Point group • Line • Sine • Polynomial • Inverse sine • Transition
Start	Parameter values at start point of the element
Leading value	Leading values at start point of the element
Following value	Following values at start point of the element
Position[1)]	Calculated effective position at start point of the element
Velocity[1)]	Calculated effective velocity at start point of the element
Acceleration[1)]	Calculated effective acceleration at start point of the element
Jerk[1)]	Calculated effective jerk at start point of the element
End	Parameter values at end point of the element
Leading value	Leading values at end point of the element
Following value	Following values at end point of the element
Position[1)]	Calculated effective position at end point of the element
Velocity[1)]	Calculated effective velocity at end point of the element
Acceleration[1)]	Calculated effective acceleration at end point of the element
Jerk[1)]	Calculated effective jerk at end point of the element
Comment	Optional comment for element.

1) Displayed according to the configuration in "Properties (Inspector window) > Graphical view > Charts and curves".

Editing the curve

The tabular editor provides you with the following options for editing the curve:

- Pasting elements
- Deleting elements
- Changing the element type
- Adjusting the leading value and following value of the boundary points

Inserting an element

To add an element in the tabular editor, follow these steps:

1. Select the desired element type from the "Add" drop-down list in the "Element type" column. "Add" is always displayed in the line after the last added element.
   The element is inserted after the last element with suitable values. The tabular editor and the view of the properties (Inspector window) are updated. A transition to any element is inserted automatically.
   The parameters of the element can be adjusted in the graphical editor, tabular editor, and properties (Inspector window).

Deleting an element

To delete an element in the tabular editor, follow these steps:

1. Select the line of the element.
2. Press the <Del> key.
   The element is deleted. The tabular editor and the view of the properties (Inspector window) are updated. A transition to any element present is also deleted.

Converting the element type

To convert the element type of an element in the tabular editor, follow these steps:

1. Select the line of the element.
2. Select the desired element type from the drop-down list in the "Element type" column.
   The element type of the element is converted to the selected element type. The tabular editor and the view of the properties (Inspector window) are updated. A transition to any element present is adjusted automatically.

Adjusting the leading value and following value of the boundary points

To change the leading value/following value of the boundary points of an element in the tabular editor, follow these steps:

1. Select the input field of the parameter to be changed.
2. Enter the desired value.
   The tabular editor and the view of the properties (Inspector window) are updated. A transition to any element present is adjusted automatically.

Shortcut menu in the tabular editor

The following table shows the functions in the shortcut menu of the tabular editor:

Function	Description
Insert row above	Insert a table row/an element before the selected line/selected element If no transition exists before the element, the selected element and the adjoining elements are changed.
Insert row below	Insert a table row/an element after the selected line/selected element If no transition exists before the element, the selected element and the adjoining elements are changed.
Cut	Removing the selected elements and copying them to the clipboard
Copy	Copying of the selected elements to the clipboard
Paste	Pasting of the elements from the clipboard to the last element
Delete	Deletion of the selected elements Transitions to existing elements are also deleted.
Paste special	Call of the "Paste elements (Page 1892)" dialog
Group points	Combine the selected points into a group of points The entry is displayed under the following conditions: • Only points are selected in the graphic/tabular editor. • There are no other elements between the selected points.
Dissolve point group	Ungroups the selected point group into individual points
Move	Call of the "Move elements (Page 1892)" dialog
Scale	Call of the "Scale elements (Page 1892)" dialog

Properties (Inspector window)

Context-sensitive display

The parameters for the profile of the cam as well as for the elements are displayed in the properties (Inspector window). The corresponding parameters are displayed according to the selected element: If no element of the curve is selected, only the settings for the profile of the cam are displayed. If an element of the curve is selected, the parameters of the element are additionally displayed.

Configuration of profile - General

Configure the leading value and following value range (definition and value range) of the cam in the "General" configuration window.

The inputs of the leading value range and following value range only effect the display in the graphical editor. The cam is interpolated in the definition range between the following values:

- First defined interpolation point/start of the first segment of the cam (<TO>.StatusCam.StartLeadingValue)
- Last defined interpolation point/end of the last segment of the cam (<TO>.StatusCam.EndLeadingValue)

Leading value range of the curve definition

Configure the leading value range (definition range) of the cam in this area:

Parameters	Description
Start	Configure the start point of the leading value range of the cam in this field.
End	Configure the end point of the leading value range of the cam in this field.

Following value range of the curve definition

Configure the restriction of the following value range (value range) of the cam in this area:

Parameters	Description
Minimum	Configure the lowest permissible value for the following value range of the cam in this field.
Maximum	Configure the maximum permissible value for the following value range of the cam in this field.

Configuration of profile - Default optimization settings

You configure the default values for optimization of transitions according to VDI Guideline 2143 in the "Default optimization settings" configuration window. The default values are used when you use the "VDI-based optimization" optimization method for a transition (Page 1884) and when you select the setting "Default optimization setting" for the continuity or the optimization target.

The cam is interpolated with the Motion Control instruction MC_InterpolateCam according to the settings for the VDI optimization.

Defaults for the VDI optimization

Configure the default settings for continuity requirement and optimization target in this area:

Parameters	Description
Continuity	In the drop-down list, select which parameter is continuous in the boundary points and is to be taken into consideration for optimization. • Position • Velocity (bumpless) • Acceleration (jerkless) • Jerk
Optimization target	In the drop-down list, select the optimization target according the VDI guideline: • Not specified • Velocity (Cv) • Acceleration (Ca) • Jerk (Cj) • Minimum Dynamic Moment (Cmdyn)

Configuration of profile - System interpolation

In the "System interpolation" configuration window, configure the interpolation of transitions according to the system specifications. These settings are used when you use the "System interpolation" optimization method for a transition (Page 1884) (default setting).

The cam is interpolated with the Motion Control instruction MC_InterpolateCam.

System interpolation settings

Configure the interpolation type and the behavior of the boundary points in this area.

Parameters	Description
Interpolation type	In the drop-down list, select the interpolation type to be used for interpolating the curve: • Linear interpolation • Interpolation with cubic splines • Interpolation with Bézier splines
Behavior at boundary	In the drop-down list, select which behavior of the boundary points applies to the interpolation: • No restrictions • First derivative continuous (velocity continuous) The cam is interpolated in such a way that the first derivative (velocity) is equal at the start and end of the cam.

Configuration of profile - Effective runtime curves

Configure the values for the leading axis and following axis that are applied to the effective curve In the "Effective runtime curves" configuration window. The runtime emulation calculates the effective curve with these applied values and displays the curve in the graphical editor with the applied limits.

The inputs are not downloaded into the CPU. This means the cam is interpolated without these inputs. You can use these applied values to test and visualize how the cam behaves during operation, e.g. when entering a scaling at "MC_CamIn".

Settings of the leading axis

Configure the calculation and display of the curve on the leading value end in this area:

Parameters	Description
Copy from axis	Using the button and the "Copy leading value settings of axis" dialog, select an axis whose maximum velocity is applied as the velocity for the leading axis.
Scaling factor	Configure a leading value-side scaling factor in this field. This allows the acceptance that a scaling is specified for an "MC_CamIn" job.
Unit of measure	In the selection list, select the unit of measurement for the leading value.

Parameters	Description
Unit of measure of the first derivative	In the selection list, select the unit of measurement for the first derivative of the leading value.
Velocity	Configure the velocity of the leading axis applied for the runtime emulation of the curve in this field.

Settings of the following axis

Configure the calculation and display of the curve on the following value side in this area:

Parameters	Description
Copy from axis	Using the button and the "Copy following value settings of axis" dialog, select an axis whose maximum dynamic values are applied as the limits to be checked during calculation and display of the curve.
Scaling factor	Configure a following value-side scaling factor in this field. This allows the acceptance that a scaling is specified for an "MC_CamIn" job.
Unit of measure	In the selection list, select the unit of measurement for the following value.
Unit of measure of the first derivative	In the selection list, select the unit of measurement for the first derivative of the following value.
Maximum velocity	Configure the maximum velocity for the following axis in this field.
Maximum acceleration	Configure the maximum acceleration for the following axis in this field.
Maximum jerk	Configure the maximum jerk for the following axis in this field.

Configuration - Check

In the "Verification" configuration window, you configure which criteria the cam editor checks when entering the curve. When you activate a check, the graphical and the tabular editor display corresponding messages via a warning triangle on the element. Use the tooltip at the warning triangle to display the message text.

Examination of limit violations

Configure the checks for compliance with the configured limits in this area:

Check/Element	Description
Observe the curve definition of the leading and following value ranges	Select the "Check curve definition of the leading and following value ranges" check box to have the cam editor check the curve accordingly.
Check adherence to the maximum values of the derivatives of the effective runtime curve	Select the "Check adherence to the maximum values of the derivatives of the effective runtime curve" check box to have the cam editor check the curve accordingly.

Verification of VDI suitability

Select the "Check the suitability of transitions in accordance with VDI" check box to have the cam editor check the VDI suitability of the curve.

The cam editor checks the following with this:

- Support of the transition classification of the currently selected VDI transition
- Boundary value adjustment according to VDI

Verification of continuity

In the "Required continuity" list, select which parameter the cam editor checks for continuity:

- Position
- Velocity
- Acceleration
- Jerk

If a function or a derivative is discontinuous, all higher derivatives are also discontinuous.

Profile - Statistics

The "Statistics" properties window shows an overview of the number of elements of the cam, as well as the minimum and maximum values of the effective curves for the slave value and the derivatives. A cam consists of a maximum of 1000 points and a maximum of 50 segments.

Used elements

This area shows the number of used elements of the curve:

Parameters	Description
Points	This field shows the number of used points of the cam. A cam consists of a maximum of 1000 points.
Segments	This field shows the number of used segments of the cam. A cam consists of a maximum of 50 segments.

The use of points and segments depends on the compilation and configuration of the elements. The following table shows the use of points and segments per element:

Element	Number of used points	Number of used segments
Point	1	0
Point at a transition with VDI-based optimization	0	0
Point group with point approximation mapping method	Number of interpolation points configured. ("Properties (Inspector window) > Element > Parameter > Approximation > Number of interpolation points") Default setting: 32	0
Point group with segment approximation mapping method	0	Number of interpolation points configured - 1
Line	0	1
Sine	0	1

Element	Number of used points	Number of used segments
Polynomial	0	1
Inverse sine	Number of interpolation points configured. ("Properties (Inspector window) > Element > Parameter > Approximation > Number of interpolation points") Default: 32	0
Inverse sine to the right of a transition with VDI-based optimization	Number of interpolation points configured - 1	0
Transition with system interpolation	0	0
Transition with VDI-based optimization		
Motion rule		
Sine	0	1
Sine with relative Lambda ≠ 0.5	0	2
Inclined sine	0	1
Inclined sine with relative Lambda ≠ 0.5	0	2
Polynomial	0	1
Sinus with relative Lambda ≠ 0.5	0	2
Modified acceleration trapezoid		
Motion task		
Dwell-in-reverse	0	5
Reverse-in-dwell	0	5
Dwell-in-dwell	0	6
Modified sine		
Motion task		
Dwell-in-dwell	0	3
Constant velocity-in-constant velocity	0	4
Constant-velocity-in-dwell	0	4
Dwell-in-constant velocity	0	4
Sine line combination	0	3
Harmonic combination	0	3
Double-harmonic transition	Number of interpolation points configured. ("Properties (Inspector window) > Element > Parameter > Approximation > Number of interpolation points") Default setting: 32	0
Quadratic parabola	0	2

Lambda = turning point of the curve

Value ranges

This area shows the minimum and maximum values of the effective curves for the following value and the derivatives.

Boundary conditions

The following boundary conditions apply to the input and use of points and segments:

- **Points**
  With points with the same leading values, the point that you have entered last or which is listed in the tabular editor is active.
- **Segments**
  - Gaps between segments are filled with a transition segment.
  - For gaps in the leading value range of less than 1.0E-4, segment end points and segment start points are pulled together.
  - For gaps in the leading value range greater than 1.0E-4, a new transition segment is inserted.
  - For overlaps, the new segment is inserted from the start point and used completely. When the previous segment is defined in excess of the new segment, the previous segment continues to be used after the end point of the new segment.
- **Interpolation points and segments** (mixed cams)
  The segment is used when points are defined in the same range.

Configuration of elements - Parameters

In the "Parameters/Characteristic" configuration window, configure the parameters of the selected element of the curve. The inputs are applied in the tabular and graphical editors. The element-specific parameters are displayed according to the selected element:

- Point (Page 1876)
- Point group (Page 1877)
- Line (Page 1879)
- Sine (Page 1880)
- Polynomial (Page 1881)
- Inverse sine (Page 1883)
- Transition (characteristic) (Page 1884)

Configuration of elements - Parameters (Point)

Configure the parameters of the selected element in the "Parameters" configuration window.

Parameters

Configure the parameters of the selected point in this area:

Parameter/Option	Description
Leading value of the point	
Leading value	In this field, configure the leading value of the point (value in the definition area).
Following values of the point	
Following value	Configure the following value of the point (value in the range of the function) in this field.
Use first derivative	Select the check box to specify the first derivative in the selected point and to include it in the interpolation of the cam.
First derivative	Configure the value of the first derivative in the selected point in this field.
Use second derivative	Select the check box to specify the second derivative in the selected point and to include it in the interpolation of the cam.
Second derivative	Configure the value of the second derivative in the selected point in this field.
Use third derivative	Select the check box to specify the third derivative in the selected point and to include it in the interpolation of the cam.
Third derivative	Configure the value of the third derivative in the selected point in this field.

The derivations are taken into consideration during VDI-based optimization of transitions of the points to other elements.

See also

Inserting a point (Page 1860)

Configuration elements - Parameters (point group)

Configure the parameters of the selected element in the "Parameters" configuration window.

Parameters

In this area, configure the parameters of the selected point group:

Parameter/Option	Description
Leading values of the point group	
Start	In this field, configure the start point of the point group in the leading value range (definition area).
End	In this field, configure the end point of the point group in the leading value area (definition area).
Interpolation points	

Parameter/Option	Description
Definition type of the leading values	In the drop-down list, select how the leading values of the interpolation points are specified: • Relative to the segment You specify the leading values of the interpolation points relative to the group of points from 0.0 to 1.0. The value 0.0 corresponds to the beginning of the point group. The value 1.0 corresponds to the end of the point group. • Absolute in the profile You specify the leading values of the interpolation points as absolute values.
Definition type of the following values	In the drop-down list, select how the following values of the interpolation points are specified: • Relative to the segment You specify the following values of the interpolation points relative to the following value range of the point group from 0.0 to 1.0. The value 0.0 corresponds to the configured minimum following value of the point group. The value 1.0 corresponds to the configured maximum following value of the point group. • Absolute in the profile You specify the following values of the interpolation points as absolute values.
Minimum following value	In this field, configure the minimum following value for the point group in the following value range.
Maximum following value	In this field, configure the maximum following value of the point group in the following value range (value range).
	Use the "Add interpolation point" button to add an interpolation point to the point group.
Interpolation points	This table shows the configured interpolation points sorted by increasing leading value. Add breakpoints using the button. Delete interpolation points by marking a row and pressing <Delete>. If you delete all points except one, the element type is changed from "Point group" to "Point".
Leading value	In this field, configure the leading value of the interpolation point (value in the definition area).
Following value	In this field, configure the following value of the interpolation point (value in the value range).
Interpolation	
Interpolation type	In the drop-down list, select the interpolation type to be used for interpolating the point group: • Interpolation with cubic splines • Interpolation with Bézier splines
Approximation	

Parameter/Option	Description
Mapping method	Select the mapping method in the drop-down list. • Point approximation • Segment approximation
Number of interpolation points	Configure the number of breakpoints for the point approximation in this field.
Maximum following value tolerance	In this field, enter the maximum permissible deviation (absolute) of the approximation from the interpolation points. If the configured value is exceeded, a warning is displayed in the graphical editor at the point group.

See also

Insert point group (Page 1861)

Configuration of elements - Parameters (line)

Configure the parameters of the selected element in the "Parameters" configuration window.

Parameters

Configure the parameters of the selected line in this area:

Parameters	Description
Leading values of the line	
Start	Configure the start point of the line in the leading value range (definition range) in this field.
End	Configure the end point of the line in the leading value range (definition range) in this field.
Following values of the line	
Definition by	In the selection list, select the parameters to be used to define the line: • Following values at start and end • Following value at the start and incline • Incline and following value at end The corresponding parameters are displayed based on the selection.
Start	Configure the start point of the line in the following value range (value range) in this field.
End	Configure the end point of the line in the following value range (value range) in this field.
Incline	Configure the incline of the line in this field.

See also

Inserting a line (Page 1862)

Configuration of elements - Parameters (sine)

Configure the parameters of the selected element in the "Parameters" configuration window.

Parameters

Configure the parameters of the selected sine element in this area:

Parameters	Description
Leading values of the sine	
Start	Configure the start point of the sine element in the leading value range (definition range) in this field.
End	Configure the end point of the sine element in the leading value range (definition range) in this field.
Trigonometric parameters	
Amplitude	Configure the amplitude of the sine element in this field.
Definition by	In the drop-down list, select how the sine element is defined: • Phase at start and at end • Phase at start and period length • Phase at start and frequency • Period length and phase at end • Frequency and phase at end The corresponding parameters are displayed based on the selection.
Phase angle at start [°]	Configure the phase angle at the start of the sine element in this field.
Phase angle at end [°]	Configure the phase angle at the end of the sine element in this field.
Period length	Configure the period length of the sine element in this field.
Frequency	Configure the frequency of the sine element in this field.
Extended parameters	
Segment type	Select the variant of the sine element in the drop-down list. • Sine • Inclined sine The corresponding parameters are displayed based on the selection. If you have configured an inclined sine, additional orientation lines are displayed in the graphical editor for the amplitude and center position.
Shift	Configure the oscillation midpoint of the sine element in this field.
Definition of ascent via	In the drop-down list, select how the inclined sine element is defined: • Following values at start and end • Following value at the start and inclination • Inclination and following value at end The corresponding parameters are displayed based on the selection.
Offset at start	Configure the center of oscillation at the start of the sine element in this field.
Offset at end	Configure the center of oscillation at the end of the sine element in this field.
Inclination	Configure the inclination of the sine element in this field.

See also

Inserting a sine (Page 1863)

Configuration of elements - Parameters (polynomial)

Configure the parameters of the selected element in the "Parameters" configuration window.

Parameters

Configure the parameters of the selected polynomial in this area:

Parameters	Description
Leading values of the polynomial	
Start	Configure the start point of the polynomial in the leading value range (definition range) in this field.
End	Configure the end point of the polynomial in the leading value range (definition range) in this field.
Polynomial parameters	

Parameters	Description
Definition by	In the selection list, select how the polynomial is defined: • Coefficients • Boundary values The corresponding parameters are displayed based on the selection.
Coefficients	Configure the coefficients of the 6th degree polynomial function in these fields: $P(x) = a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ The coefficients are shown in scientific notation, e.g. "9.6450617283e-11".
Following value - Left boundary value	Configure the following value at the start of the polynomial in this field.
Following value - Right boundary value	Configure the following value at the end of the polynomial in this field.
Use first derivative	Select the check box to specify the first derivative in the left/right boundary value of the polynomial and to include it in the interpolation of the cam.
First derivative - left boundary value	Configure the first derivative (velocity) for the following value at the start of the polynomial in this field.
First derivative - right boundary value	Configure the first derivative (velocity) for the following value at the end of the polynomial in this field.
Use second derivative	Select the check box to specify the second derivative in the left/right boundary value of the polynomial and to include it in the interpolation of the cam.
Second derivative - left boundary value	Configure the second derivative (acceleration) for the following value at the start of the polynomial in this field.
Second derivative - right boundary value	Configure the second derivative (acceleration) for the following value at the end of the polynomial in this field.
Use third derivative	Select the check box to specify the third derivative in the left/right boundary value of the polynomial and to include it in the interpolation of the cam.
Third derivative - left boundary value	Configure the third derivative (jerk) for the following value at the start of the polynomial in this field.
Third derivation - right boundary value	Configure the third derivative (jerk) for the following value at the end of the polynomial in this field.
Lambda	In the selection list, select how the turning point of the polynomial is specified in the "Lambda position" field: • No lambda Do not enter any value. The position of the point of inflection is calculated automatically. • Relative to the element You specify the leading value of the turning point relative to the polynomial from 0.0 to 1.0. The value 0.0 corresponds to the beginning of the polynomial. The value 1.0 corresponds to the end of the polynomial. • Absolute in the profile You specify the leading value of the point of inflection as an absolute value.
	In this field, configure the leading value of the point of inflection for the polynomial according to the selection in the selection list.
Extended parameters	

Parameters	Description
Segment type	In the selection list, select whether or not the polynomial is to have a trigonometric component. When "Polynomial with trigonometric portion" is selected, the corresponding trigonometric parameters are displayed, as they are with sine. When a sine element is converted to a polynomial, the sine element is configured as a polynomial with trigonometric portion. The shape of the element is retained.
Amplitude	Configure the amplitude of the trigonometric component in this field.
Definition by	In the selection list, select how the trigonometric component is defined: • Phase at start and at end • Phase at start and period length • Phase at start and frequency • Period length and phase at end • Frequency and phase at end The corresponding parameters are displayed based on the selection.
Phase at start	Configure the phase angle at the start of the trigonometric component in this field.
Phase at end	Configure the phase angle at the end of the trigonometric component in this field.
Period length	Configure the period length of the trigonometric component in this field.
Frequency	Configure the frequency of the trigonometric component in this field.

See also

Inserting a polynomial (Page 1864)

Configuration of elements - Parameters (inverse sine)

Configure the parameters of the selected element in the "Parameters" configuration window.

The inverse sine is defined within the definition range [-1, 1]. The inverse sine can be calculated for the entire definition range or a restricted definition range of the arcsine function.

An inverse sine is approximated using interpolation points of the arcsine function.

Parameters

Configure the parameters of the selected inverse sine in this area:

Parameters	Description
Leading values of the inverse sine	
Start	Configure the start point of the inverse sine in the leading value range (definition range) in this field.
End	Configure the end point of the inverse sine in the leading value range (definition range) in this field.

Parameters	Description
Following values of the inverse sine	
Minimum	Configure the minimum value of the inverse sine in the following value range (value range) in this field.
Maximum	Configure the maximum value of the inverse sine in the following value range (value range) in this field.
Definition range	
Not mirrored/mirrored	Select whether or not the inverse sine is to be mirrored about the abscissa.
Start	Configure the start point in the definition range of the arcsine function that is to be used in this field.
End	Configure the end point in the definition range of the arcsine function that is to be used in this field.
Approximation	
Number of interpolation points	Configure the number of interpolation points for the approximation in this field.
Maximum following value tolerance	In this field, specify the maximum permitted deviation (absolute) of the approximation from the arcsine function. If the configured value is exceeded, a warning is displayed in the graphical editor for the arcsine element.

See also

Inserting an inverse sine (Page 1865)

Configuration of elements - Characteristic (transition)

Configure the parameters of the selected transition in the "Characteristics" configuration window.

Characteristics

Configure the settings for optimization of the transition in this area:

Parameters	Description
Interpolation settings of the transition	
Optimization method	Select the optimization method in the drop-down list. • System interpolation The CPU defines the optimization parameters automatically according to the settings of the system interpolation (Page 1871). • VDI-based optimization You adjust the optimization manually. The inputs are applied automatically according to the VDI Guideline 2143.
Motion task	The transition type is determined from the properties of the adjacent elements of the transition and displayed in this field.
Continuity at start/end	In the drop-down lists, select which parameter is continuous in the boundary points and is to be included for optimization. • Default optimization setting (setting under "Profile > Default optimization settings (Page 1870)") • Position • Velocity (bumpless) • Acceleration (jerkless) • Jerk (jerk continuity permitted on one side only)
Optimization target	In the drop-down list, select the optimization target: • Default optimization setting (setting under "Profile > Default optimization settings") • Not specified • Velocity (Cv) • Acceleration (Ca) • Jerk (Cj) • Minimum Dynamic Moment (Cmdyn)
Selection of motion rule	

Parameters	Description
Motion rule	In the drop-down list, select the motion rule according to which optimization is to occur. • Line • Quadratic parabola • Sine • Polynomial • Inclined sine • Modified acceleration trapezoid • Modified sine • Harmonic combination • Double-harmonic transition • Sine line combination The selection is automatically limited to the motion rules that can be applied according to the motion task and the selected boundary conditions. Additional parameters are displayed depending on the selected motion rule.
Parameter used	In the drop-down list, select the parameters to be included in the optimization: • Lambda • Maximum acceleration (Ca) • Maximum deceleration (Ca*) The selection is automatically limited to the parameters that can be applied according to the motion rule.
Lambda	In the drop-down list, select the transition point in the "Lambda position" field: • No lambda Do not enter any value. The position of the point of inflection is calculated automatically. • Relative to the segment You specify the leading value of the turning point relative to the transition from 0.0 to 1.0. The value 0.0 corresponds to the beginning of the transition. The value 1.0 corresponds to the end of the transition. • Absolute in the profile You specify the leading value of the point of inflection as an absolute value.
Lambda position	In this field, configure the leading value of the turning point for the transition according to the selection in the "Lambda" drop-down list.
Maximum acceleration (Ca)	Configure the maximum acceleration (Ca) for the transition in this field.
Maximum deceleration (Ca*)	Configure the maximum deceleration (Ca*) for the transition in this field.
Approximation	
Number of interpolation points	In this field, configure the number of interpolation points for the transition.

Parameters	Description
Maximum following value tolerance	In this field, enter the maximum permitted deviation (absolute) of the approximation from the motion law. If the configured value is exceeded, a warning is displayed in the graphical editor at the transition.
Characteristic values of the transition	The characteristic values of the transition that are relevant according to VDI 2143 are displayed in this area. The maximum value and the standardized value are displayed for the following characteristic values: • Velocity (Cv) • Acceleration (Ca) • Deceleration (Ca*) • Jerk (Cj) • Dynamic torque (Cmdyn)

Motion jobs according to VDI Guideline 2143

The VDI Guideline 2143 distinguishes between areas of usage and motion transitions:

- Areas of usage correspond to the sequences in a process, which means the inserted elements of the cam.
- Motion transitions are transitions between areas of usage that are not directly relevant to the process but must meet specific boundary conditions (e.g. velocity consistency).

The following motion tasks are defined based on VDI guideline 2143:

Motion tasks	Designation	Properties
Dwell	R	Velocity = 0 Acceleration = 0
Constant velocity	G	Velocity ≠ 0 Acceleration = 0
Reverse	U	Velocity = 0 Acceleration ≠ 0
Motion	B	Velocity ≠ 0 Acceleration ≠ 0

The following figure shows an example of the motion tasks:

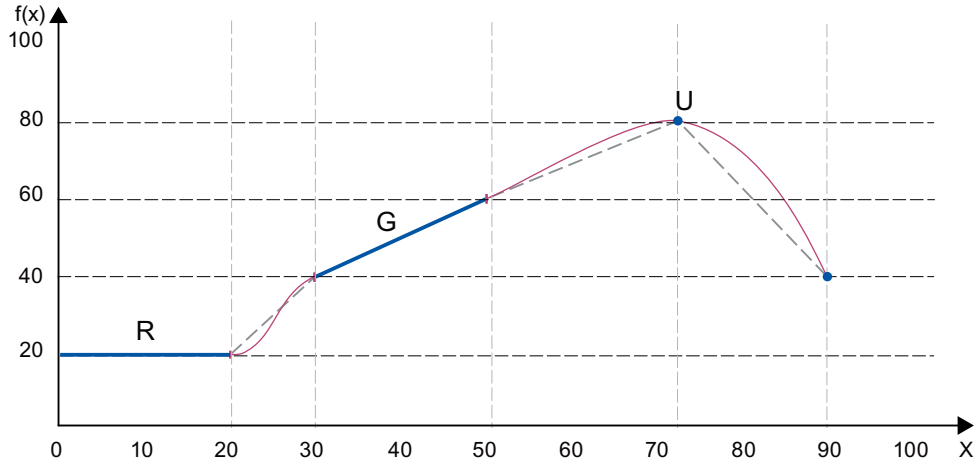

The figure below shows the possible combinations of motion tasks:

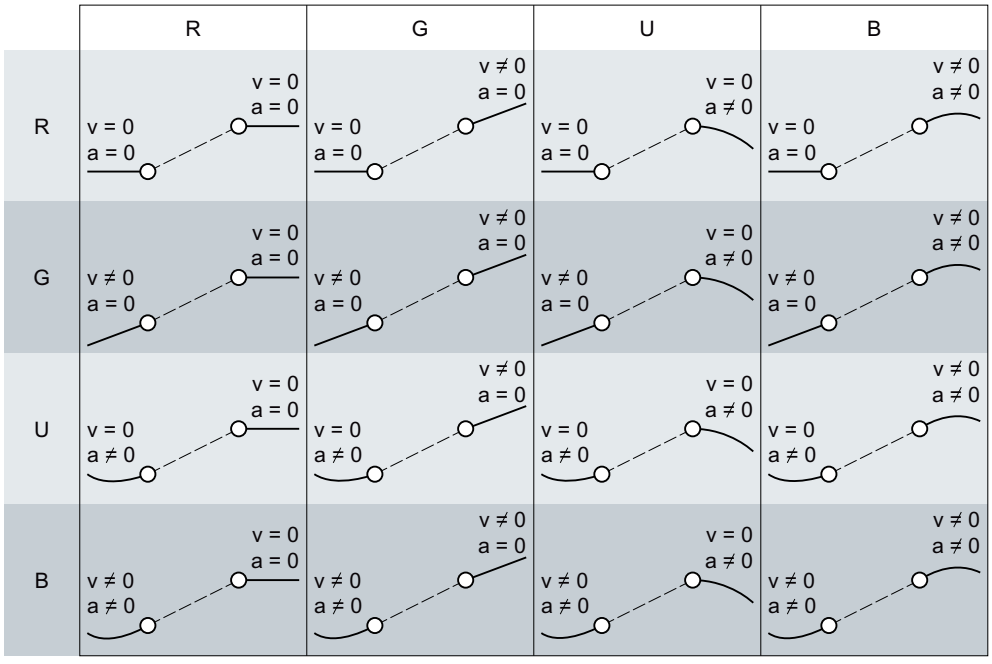

Representation (Inspector window)

Configuration charts - Charts and curves

In the "Charts and curves" configuration window, configure the display of the graphical editor.

"Reset to defaults" button

Use this button to reset all settings of the view of charts and curves to the default settings.

Configuration table

Configure the display of the graphical editor in the table:

Column	Description
Displaying	Displaying/hiding of charts 1 to 4
Visible	Displaying/hiding of curves in the chart
Name	Name of chart or curve New curves can be added. Existing curves can be removed. Curves of other cams can also be displayed. The name of the other cam is also displayed in the table and in the legend of the chart. A curve can be inserted multiple times in a chart, e.g. in order to display it with different scalings.
Color	Line color of the curve
Line type	Line type of the curve
Offset of the leading values[1)]	Movement of the curve on the abscissa
Multiplier for leading values[1)]	Scaling of abscissa
Offset of the following values[1)]	Movement of the curve on the ordinate
Multiplier for following values[1)]	Scaling of ordinate

1) Only affects the display of the curve in the chart. You specify the scaling and shifting of the cam during camming in the Motion Control instruction "MC_CamIn".

Configuration charts - Snap grid

In the "Snap grid" configuration window, you configure the grid spacing for aligning inputs to the grid in the graphical editor. When "Snap" is activated, inputs and element end points are aligned to this grid and to other element end points.

Snap grid spacing

In this area, configure the grid spacing of the snap grid:

Parameter	Description
Grid spacing leading value	Configure the grid spacing on the abscissa (leading values) in this field.
Grid spacing following value	Configure the grid spacing on the ordinate (following values) in this field.

Configuration - Decimal places

In the "Decimal places" configuration window, you configure how many decimal places are used to represent the values in the graphical and tabular editor as well as in the configuration windows. The values are rounded in the displays. The settings do not affect the calculation of the curves. The curves are calculated with higher accuracy regardless of the settings.

Displayed decimal places

In this area, configure the displayed decimal places:

Parameter	Description
Tabular editor and configuration window	In this field, configure the number of decimal places for displaying values in the tabular editor and in the configuration windows.
Graphical editor	In this field, configure the number of decimal places for displaying values in the graphical editor.

Importing / exporting cam

You can use the toolbar to export cams from the cam editor and import cams into the cam editor.

Importing cam

NOTICE
Machine damage Importing corrupt files (.txt, .csv) can result in unwanted behavior of the axes. Each time you import a cam from a file, check the integrity of the imported data.

The following table shows the supported file formats for importing/exporting a cam:

File format	Comment
Import format	
SIMOTION SCOUT format/MCD *.txt, *.csv	MCD exchange format is automatically detected, imported data: • Interpolated points • Lines • Sine elements • Inverse sine elements • Polynomials • Transitions
Proprietary binary format *.bin	The binary format is used for exchanging cams between multiple TIA Portal installations and external applications.

To import a cam, follow these steps:

1. In the toolbar, click the icon "Import cam from file".
   The "Cam import" dialog opens.
2. Select the file type of the file you want to import.
3. Select the file you want to import from the file directory.
4. Click the "Open" button.
   The cam is opened in the cam editor. All previous entries in the editor are discarded.

Exporting cam

The following table shows the structure of the "Cam export" dialog:

Parameter/Element	Description
Export format	
Export as	Select the export formate in the drop-down list: • MCD exchange format[1)] • Standard vector format • Point list • Binary format[1)]
Delimiters	In the drop-down list, select the delimiter with which the data fields are to be separated in the file: • Comma • Tab
Number of points	In this field, configure the number of points to be exported to a point list. The more points exported, the more precise the configured cam formed by the point list. Possible values: 0 to 1E5 Default setting: 360
Additional curves	Point list only
Velocity	Select the "Velocity" check box when the derivative curve of the velocity is to be exported in addition to the position.
Acceleration	Select the "Acceleration" check box if you want to export the derivative curve of the acceleration in addition to the position.
Jerk	Select the "Jerk" check box when the derivative curve of the jerk is to be exported in addition to the position.
Directory for export	
File name	Enter a file name in this field.
Directory	In this field, enter the directory into which the file is to be written.
Export	Export the file
Cancel	Cancellation of export and closing of the dialog

1) Can be imported into TIA Portal

To export a cam, follow these steps:

1. In the toolbar, click the icon "Export cam to file".
   The "Cam export" dialog opens.
2. Select the export format in the "Export as" drop-down list.
3. Optionally, configure the delimiter, the number of points, and the additional curves for the export.
4. Enter a file name in the "File name" box.
5. Select the directory to which the file is written.
6. Click "Export".

See also

Structure of the graphical editor (Page 1858)

Dialogs in the shortcut menu

The following dialogs can be called with the shortcut menu of the graphical and tabular editor:

- Pasting elements
- Moving elements
- Scaling elements

"Paste elements" dialog

The following table shows the structure of the "Paste elements" dialog:

Parameter/Element	Description	
Insert mode	Select the Insert mode from the selection list:	
	Overwrite from the end to the left	You overwrite the selected elements with the elements from the clipboard starting from the end in the direction of smaller leading values. The end of the inserted elements then lies at the end of the selected elements. Elements that are located in the leading value range of the elements in the clipboard are overwritten or truncated.
	Overwrite from the start to the right	Starting from the start, you overwrite the selected elements with the elements from the clipboard in the direction of larger leading values. The start of the inserted elements then lies at the start of the selected elements. Elements that are located in the leading value range of the elements in the clipboard are overwritten or truncated.
	Overwrite from the middle	Starting from the center, you overwrite the selected elements with the elements from the clipboard. The center of the inserted elements then lies at the center of the selected elements. Elements that are located in the leading value range of the elements in the clipboard are overwritten or truncated.
	Scale selection to the leading value range	The elements in the clipboard are scaled to the leading value range of selected elements. The start and end of the inserted elements then lie at the start and end of the selected elements.
Paste	Pasting of the elements from the clipboard with the selected mode	
Cancel	Cancellation of paste operation and closing of the dialog	

"Move elements" dialog

The following table shows the structure of the "Move elements" dialog:

Parameter/Element	Description
Horizontal distance	In this field, enter the shift of the selection on the abscissa (x axis).
Vertical distance	In this field, enter the shift of the selection on the ordinate (y axis).
Move	Move the selection by the entered distance
Cancel	Cancellation of move operation and closing of the dialog

"Scale elements" dialog

The following table shows the structure of the "Scale elements" dialog:

Parameter/Element	Description	
Adjust to leading value range	In this field, enter the scaling length (leading value side) to which you want to scale the selection.	
Anchor point	Select the direction of scaling in the selection list:	
	Left boundary	The selection is adjusted by the left boundary point to the scaling length.
	Center	The selection is adjusted by the center point to the scaling length.
	Right boundary	The selection is adjusted by the right boundary point to the scaling length.
Scale	Scaling with the selected parameter values	
Cancel	Cancellation of scaling and closing of the dialog	

See also

Shortcut menu in the tabular editor (Page 1869)

Shortcut menu in the graphical editor (Page 1866)

Configuring technology modules for Motion Control (S7-1500)

Use of technology modules with Motion Control

You can use technology modules as well as compact CPUs with Motion Control. To use the technology functions for Motion Control, the parameters of the device configuration of the technology module or the compact CPU and the parameters of the configuration of the technology object must be configured accordingly. Which parameters are relevant for the function is described below. You can set additional parameters that are not listed here. The description of the parameters is found in the documentation of the respective technology module.

The following technology modules support Motion Control functionalities:

S7-1500/ET 200MP	ET 200 SP
TM Count 2x24V (Page 1894) [1]	TM Count 1x24V (Page 1894) [1]
TM PosInput 2 (Page 1896) [1]	TM PosInput 1 (Page 1896) [1]
TM Timer DIDQ 16x24V (Page 1898) [2]	TM Timer DIDQ 10x24V (Page 1898) [2]
–	TM Pulse 2x24V (Page 1899) [1]
TM PTO 4 (Page 1900) [3]	–
CPU 1511C-1 PN/CPU 1512C-1 PN (Page 1902) [3]	–

[1] Automatic data exchange for encoder values is supported

[2] ET 200MP and ET 200SP: Isochronous mode required

[3] Automatic data exchange for drive and encoder values is supported

Isochronous mode

Technology modules can be used centrally or distributed in the system. However, isochronous mode is supported only in distributed operation with suitable PROFINET interface modules.

Automatic data exchange for drive and/or encoder values

By selecting the check box for automatic data exchange, the drive and encoder parameters are automatically applied in the CPU.

Alternatively, you can manually match, by drive and encoder type, the parameters described and identified in the following table.

The following types are available for automatic data exchange:

- **Offline**
  Select the check box if you want to transfer the offline values of the drive or encoder to the configuration of the technology object in the project.
- **Online**
  Select the check box if you want to transfer the effective values online in the drive or encoder to the CPU during runtime. The drive and encoder parameters are transferred from the bus after the (re)initialization of the technology object and (re)start of the drive, encoder or the CPU.

TM Count 1x24V/TM Count 2x24V (S7-1500)

For use with Motion Control, the following parameters must be configured:

Configuration		
Technology module	Technology object	
TM Count 1x24V/TM Count 2x24V	Axis	External encoder
TM Count 1x24V/TM Count 2x24V > Channel 0/1 > Operating mode	–	–
Select "Position detection for Motion Control technology object" mode		
TM Count 1x24V/TM Count 2x24V > Channel 0/1 > Module parameters	**Hardware interface > Encoder**	**Hardware interface > Encoder**

Configuration		
Technology module TM Count 1x24V/TM Count 2x24V	Technology object	
	Axis	External encoder
–	Select "Encoder" data connection and the channel configured for Motion Control on the technology module as encoder	Select "Encoder" data connection and the channel configured for Motion Control on the technology module as encoder
Signal type • Incremental encoder	Select encoder type corresponding to configuration for technology module • Incremental [1]	Select encoder type corresponding to configuration for technology module • Incremental [1]
–	**Hardware interface > Data exchange with encoder**	**Hardware interface > Data exchange**
	Telegram "DP_TEL83_STANDARD" is automatically selected after the selection of the encoder.	Telegram "DP_TEL83_STANDARD" is automatically selected after the selection of the encoder.
	Clear the check box "Automatic data exchange for encoder values (online)"	Clear the check box "Automatic data exchange for encoder values (online)"
	Select the check box "Automatic data exchange for encoder values (offline)" If the check box is cleared, you can manually match the parameters described and identified in this table.	Select the check box "Automatic data exchange for encoder values (offline)" If the check box is cleared, you can manually match the parameters described and identified in this table.
	Select rotary or linear measuring system type [1]	Select rotary or linear measuring system type [1]
Signal evaluation • Single • Double • Quadruple	Select fine resolution corresponding to configuration on the technology module [1] • 0 = Single • 1 = Double • 2 = Quadruple	Select fine resolution corresponding to configuration on the technology module [1] • 0 = Single • 1 = Double • 2 = Quadruple
• Rotary type: Enter increments per revolution • Linear type: Configuration not relevant	• Rotary type: Enter increments per revolution corresponding to configuration at technology module (1:1) [1] • Linear type: Enter distance between increments [1]	• Rotary type: Enter increments per revolution corresponding to configuration at technology module (1:1) [1] • Linear type: Enter distance between increments [1]
–	**Hardware interface > Data exchange with the drive**	–
• Rotary type: Enter reference speed corresponding to configuration for technology object (1:1) • Linear type: Configuration not relevant	Enter reference speed	
–	**Homing**	**Homing**

Configuration		
Technology module TM Count 1x24V/TM Count 2x24V	Technology object	
	Axis	External encoder
Select the homing signal for homing mark 0: • Signal N of the incremental encoder • DI0	Use the homing mode "Use zero mark via PROFIdrive telegram".	Use the homing mode "Use zero mark via PROFIdrive telegram".
TM Count 2x24V > I/O addresses	–	–
The organization block ("MC-Servo") and the process image ("TPA OB Servo") are selected automatically for the input and output addresses by selecting the channel in the encoder configuration at the technology object.		
Process image: PIP OB servo		

[1] Parameters are automatically applied when "Automatic data exchange for encoder values (offline)" is selected

"–" No configuration for technology module/technology object is required for these parameters

TM PosInput 1/TM PosInput 2 (S7-1500)

For use with Motion Control, the following parameters must be configured:

Configuration		
Technology module TM PosInput 1 / TM PosInput 2	Technology object	
	Axis	External encoder
TM PosInput 1/2 > Channel 0/1 > Operating mode	–	–
Select "Position detection for Motion Control technology object" mode		
TM PosInput 1/2 > Channel 0/1 > Module parameters You set the parameters for the encoder signals of the channel under "Module parameters" in the "Position input for Motion Control" mode. The parameters must be set depending on the encoder used. The configuration of the encoder is required for use with an SSI absolute encoder. Information on the configuration is available in the documentation for the respective technology module.	**Hardware interface > Encoder**	**Hardware interface > Encoder**
–	Select "Encoder" data connection and the channel activated and configured as encoder on the technology module	Select "Encoder" data connection and the channel activated and configured as encoder on the technology module

Configuration		
Technology module TM PosInput 1 / TM PosInput 2	Technology object	
	Axis	External encoder
Signal type • Incremental encoder • Absolute encoder	Select encoder type corresponding to configuration for technology module: • Incremental [1)] • Absolute/Cyclic absolute	Select encoder type corresponding to configuration for technology module: • Incremental [1)] • Absolute/Cyclic absolute
–	**Hardware interface > Data exchange with encoder**	**Hardware interface > Data exchange**
	Telegram "DP_TEL83_STANDARD" is automatically selected after the selection of the encoder.	Telegram "DP_TEL83_STANDARD" is automatically selected after the selection of the encoder.
	Clear the check box "Automatic data exchange for encoder values (online)"	Clear the check box "Automatic data exchange for encoder values (online)"
	Select the check box "Automatic data exchange for encoder values (offline)" If the check box is cleared, you can manually match the parameters described and identified in this table.	Select the check box "Automatic data exchange for encoder values (offline)" If the check box is cleared, you can manually match the parameters described and identified in this table.
	Select rotary or linear measuring system type [1)]	Select rotary or linear measuring system type [1)]
Signal evaluation • Single • Double • Quadruple	Select fine resolution corresponding to configuration on the technology module [1)] • Incremental encoder: – 0 = Single – 1 = Double – 2 = Quadruple • Absolute encoder: – 0 = Single	Select fine resolution corresponding to configuration on the technology module [1)] • Incremental encoder: – 0 = Single – 1 = Double – 2 = Quadruple • Absolute encoder: – 0 = Single
• Rotary type: Enter increments per revolution • Linear type: Configuration not relevant	• Rotary type: Enter increments per revolution corresponding to configuration at technology module (1:1) [1)] • Linear type: Enter distance between increments [1)]	• Rotary type: Enter increments per revolution corresponding to configuration at technology module (1:1) [1)] • Linear type: Enter distance between increments [1)]
–	**Hardware interface > Data exchange with the drive**	–
• Rotary type: Enter reference speed corresponding to configuration for technology object (1:1) • Linear type: Configuration not relevant	Enter reference speed	
–	**Homing**	**Homing**

Configuration		
Technology module TM PosInput 1 / TM PosInput 2	Technology object	
	Axis	External encoder
Select the homing signal for homing mark 0: • Signal N of the incremental encoder • DI0	Use the homing mode "Use zero mark via PROFIdrive telegram".	Use the homing mode "Use zero mark via PROFIdrive telegram".
TM PosInput 1/2 > I/O addresses The organization block ("MC-Servo") and the process image ("TPA OB Servo") are selected automatically for the input and output addresses by selecting the channel in the encoder configuration at the technology object.	–	–

[1] Parameters are automatically applied when "Automatic data exchange for encoder values (offline)" is selected

"–" No configuration for technology module/technology object is required for these parameters

TM Timer DIDQ 10x24V/TM Timer DIDQ 16x24V (S7-1500)

You can operate the TM Timer DIDQ technology module centrally on an S7-1500 CPU or decentrally on a distributed I/O. For use with a measuring input, output cam or cam track, the technology module must be used decentrally and with isochronous mode.

For use with Motion Control, the following parameters must be configured:

Use with output cam/cam track technology object

Configuration	
Technology module TM Timer DIDQ 10x24V/TM Timer DIDQ 16x24V	Technology object
	Output cam / Cam track
Basic parameters Select desired number of outputs under channel configuration (ET 200MP TM Timer DIDQ 16x24V only)	–
Channel parameters	Hardware interface > Output cam output/output cam track
–	Activate output
	Select output by TM Timer DIDQ
Select "Timer DQ" mode for the respective output	Select output cam output
I/O addresses Select "Isochronous mode"	–
The organization block ("MC-Servo") and the process image ("TPA OB Servo") are updated automatically for the input and output addresses by selecting the channel in the encoder configuration at the technology object.	

"–" No configuration for technology module/technology object is required for these parameters

Use with technology object measuring input

Configuration	
Technology module TM Timer DIDQ 10x24V/TM Timer DIDQ 16x24V	Technology object Measuring input
Basic parameters	–
Select desired number of inputs under channel configuration	
Channel parameters	**Hardware interface > Measuring input**
Configuration of DI group: Use inputs individually	–
Select "Timer DI" mode for the respective input	Select measuring using TM Timer DIDQ
–	Select measuring input
Select application-dependent input delay	–
I/O addresses	
Select "Isochronous mode"	
The organization block ("MC-Servo") and the process image ("TPA OB Servo") are updated automatically for the input and output addresses by selecting the channel in the input configuration at the technology object.	

"–" No configuration for technology module/technology object is required for these parameters

TM Pulse 2x24V (S7-1500)

For use with Motion Control, the parameters described below must each be configured:

Drive connection using PWM (pulse width modulation)

Configuration	
TM Pulse 2x24V	Technology object Axis
TM Pulse 2x24V > Channel configuration	–
Select if you want to use 1 or 2 channels.	
TM Pulse 2x24V > Channel > Operating mode	
Select "Pulse width modulation PWM" or "PWM with DC motor" operating mode	
TM Pulse 2x24V > Channel > Parameters	**Hardware interface > Drive**
Select "S7 analog output" output format	Select analog drive connection For the selection of the analog output, create a PLC tag of the "Int" type with corresponding address. The offset for the PLC tag to the start address is 2. To activate the output of the PWM signal, set the following two bits of the control interface of the PWM channel in the user program: • SW_ENABLE (= Bit 0 in Byte 9) • TM_CTRL_DQ (= Bit 1 in Byte 9) The offset for byte 9 to the start address of the PWM channel is 9.

Configuration	
TM Pulse 2x24V	Technology object
	Axis
TM Pulse 2x24V > Channel > I/O addresses	–
Select the organization block "MC-Servo" for the input and output addresses. The "TPA OB Servo" process image is selected automatically for the input and output addresses by selecting the organization block.	

"–" No configuration for technology object is required for these parameters

TM PTO 4 (S7-1500)

For use with Motion Control, the following parameters must be configured.

Configuration	
Technology module TM PTO 4	Technology object
	Axis
TM PTO 4 > Channel configuration	–
Configure the number of channels (1 to 4) you want to use.	
TM PTO 4 > Channel 0…3 > Operating mode	
Select signal type: • PTO (pulse (P) and direction (D)) • PTO (count up (A) and count down (B)) • PTO (A, B phase-shifted) • PTO (A, B phase-shifted - quadruple)	
Select signal interface: • RS422, symmetrical/TTL (5V), asymmetrical • 24V asymmetric	
Configure the interpulse pause for direction reversal.	
–	Hardware interface > Drive
	Select drive type "PROFIdrive" and "Drive" data connection. Select the pulse output configured at the technology module as drive.
	Hardware interface > Encoder
	The encoder of the actuator telegram (simulated encoder) is automatically selected. Alternatively, an existing encoder interface can be selected.
TM PTO 4 > Channel 0...3 > Diagnostic interrupts	-
When the "Enable diagnostic interrupts" check box is selected, diagnostic interrupts are activated if: • No supply voltage • Errors occur at digital outputs The detected error is displayed for the respective channel with feedback bit "Fault_Present" and "Sensor_Error".	
TM PTO 4 > Channel 0…3 > Axis parameters	Data exchange with the drive

Configuration	
Technology module TM PTO 4	Technology object Axis
–	Telegram "DP_TEL3_STANDARD" is automatically selected after the selection of the drive.
	Clear check box "Automatic data exchange for drive values (online)"
	Select check box "Automatic data exchange for drive values (offline)" If the check box is cleared, you can manually match the parameters described and identified in this table.
Enter reference speed corresponding to configuration for technology object (1:1)	Enter reference speed of the drive [1]
Enter maximum speed corresponding to configuration for technology object (1:1)	Enter maximum speed of the drive [1] If the maximum speed is exceeded, technology alarm 102 is triggered and displayed.
–	**Data exchange with encoder**
	Telegram "DP_TEL3_STANDARD" is automatically selected after the selection of the encoder.
	Clear the check box "Automatic data exchange for encoder values (online)"
	Select the check box "Automatic data exchange for encoder values (offline)" If the check box is cleared, you can manually match the parameters described and identified in this table.
	Select rotary measuring system type [2]
Enter increments per revolution	Enter increments per revolution corresponding to configuration at technology module (1:1) [2]
Configure fine resolution • 0 = Single • 2 = Quadruple	Select fine resolution corresponding to configuration on the technology module [2] • 0 = Single • 2 = Quadruple
Configure stop behavior • Quick stop time • Ramp stop time	–
TM PTO 4 > Channel 0…3 > Hardware inputs/outputs	–
If you want to use a hardware output to enable the drive, select the "Use drive enable" check box. Next select one of the two hardware outputs DQ0 or DIQ2.	No setting required at the technology object. The output is automatically controlled by the "MC_Power".
	Homing
Activate the hardware input (DI0) for the reference cam.	Use the homing mode "Use zero mark via PROFIdrive telegram".
Select the edge of the hardware input for triggering the reference cam function.	

Configuration	
Technology module TM PTO 4	Technology object Axis
When using a measuring input, select the "Use DI1 check box as measuring input".	**"Measuring input > Configuration > Hardware interface" technology object** Select the measuring input type "Measuring via PROFIdrive telegram (drive or external encoder)". Select the measuring input "1" under hardware connection.
Select the "Use "drive ready"" check box. In the ""Drive ready" input", select the hardware input that is to be used to display whether the drive is ready.	No setting required at the technology object. When the input is used, "MC_Power" waits until the input signal is present before it sets the drive enable.
Configuring input delay	–
TM PTO 4 > Channel 0…3 > Sign-of-life error	
Configure tolerated number of sign-of-life errors	
TM PTO 4 > I/O addresses	
The organization block ("MC-Servo") and the process image ("TPA OB Servo") are selected automatically for the input and output addresses by selecting the PTO channel for the technology object.	

1) Parameters are automatically applied when "Automatic data exchange for drive values (offline)" is selected

2) Parameters are automatically applied when "Automatic data exchange for encoder values (offline)" is selected

"–" No configuration required for these parameters at the technology module/technology object

CPU 1511C-1 PN/CPU 1512C-1 PN (S7-1500)

For use with Motion Control, the parameters described below must be configured.

Drive connection via PTO (Pulse Train Output)

Configuration	
CPU 1511C-1 PN/CPU 1512C-1 PN	Technology object Axis
Pulse generators (PTO/PWM) > PTO1…4/PWM1…4 > General	–
To activate a channel for PTO mode, select one of the following operating modes: • PTO (pulse (A) and direction (B)) • PTO (count up (A), count down (B)) • PTO (A,B phase shifted) • PTO (A,B phase shifted, quadruple)	

Configuration	
CPU 1511C-1 PN/CPU 1512C-1 PN	Technology object
	Axis
–	**Hardware interface > Drive**
	Select drive type "PROFIdrive" and "Drive" data connection. Select the pulse generator of the CPU configured for PTO mode as drive.
	Hardware interface > Encoder
	The encoder of the actuator telegram (simulated encoder) is automatically selected. Alternatively, an existing encoder interface can be selected.
Pulse generators (PTO/PWM) > PTO1…4/PWM1…4 > Axis parameters	**Hardware interface > Data exchange with the drive**
–	Telegram "DP_TEL3_STANDARD" is automatically selected after the selection of the drive.
	Clear check box "Automatic data exchange for drive values (online)"
	Select check box "Automatic data exchange for drive values (offline)" If the check box is cleared, you can manually match the parameters described and identified in this table.
Enter reference speed corresponding to configuration for technology object (1:1)	Enter reference speed of the drive [1)]
Enter reference speed corresponding to configuration for technology object (1:1)	Enter maximum speed of the drive [1)] If the maximum speed is exceeded, technology alarm 102 is triggered and displayed.
–	**Hardware interface > Data exchange with encoder**
	Telegram "DP_TEL3_STANDARD" is automatically selected after the selection of the encoder.
	Clear the check box "Automatic data exchange for encoder values (online)"
	Select the check box "Automatic data exchange for encoder values (offline)" If the check box is cleared, you can manually match the parameters described and identified in this table.
	Select rotary measuring system type [2)]
Enter increments per revolution	Enter increments per revolution corresponding to configuration for CPU (1:1) [2)]
The fine resolution has the fixed value "0 bit" (= single) and cannot be changed.	Enter the fine resolution corresponding to the configuration on the CPU [2)] Bits in incr. actual value (G1_XIST1): 0 (= single)
Pulse generators (PTO/PWM) > PTO1…4/PWM1…4 > Hardware inputs/outputs	**Homing**

Configuration	
CPU 1511C-1 PN/CPU 1512C-1 PN	Technology object
	Axis
Select the hardware input for the reference switch In addition, configure the input delay for the selected hardware input. You configure the input delay in the device configuration at the corresponding DI channel (DI 16/DQ 16 > Inputs > Channel > Input parameters > Input delay).	Use the homing mode "Use zero mark via PROFIdrive telegram" for drive connection via PTO.
Select the edge of the hardware input for triggering the reference cam function.	
When using a measuring input, select the hardware input of the measuring input. The following table includes the configuration description.	–
Select the hardware input that is used to display whether the drive is ready. In addition, configure the input delay for the selected hardware input. You configure the input delay in the device configuration at the corresponding DI channel (DI 16/DQ 16 > Inputs > Channel > Input parameters > Input delay).	
With selected "PTO (PTO (pulse (A) and direction (B))" mode, the hardware output for the PTO signal A ("Pulse output (A)") is automatically selected through the device configuration and cannot be changed. For PTO signal B ("Direction output (B)") select one of the hardware outputs offered in the selection box. The hardware outputs for the PTO signals are selected through the device configuration for the following operating modes and cannot be changed: • PTO (count up (A), count down (B)) • PTO (A,B phase shifted) • PTO (A,B phase shifted, quadruple)	
Pulse generators (PTO/PWM) > PTO1…4/PWM1…4 > I/O addresses	
The organization block ("MC-Servo") and the process image ("TPA OB Servo") are selected automatically for the input and output addresses by selecting the PTO channel for the technology object.	

1) Parameters are automatically applied when "Automatic data exchange for drive values (offline)" is selected

2) Parameters are automatically applied when "Automatic data exchange for encoder values (offline)" is selected

"–" No configuration for CPU/technology object is required for these parameters

Additional configuration for use with the technology object measuring input

Configuration	
Technology module CPU 1511C-1 PN/CPU 1512C-1 PN	Technology object Measuring input
Pulse generators (PTO/PWM) > PTO1…4/PWM1…4 > Hardware inputs/outputs	**Hardware interface > Measuring input**
Select the hardware input of the measuring input. In addition, configure the input delay for the selected hardware input. You configure the input delay in the device configuration at the corresponding DI channel (DI 16/DQ 16 > Inputs > Channel > Input parameters > Input delay).	Measuring using PROFIdrive telegram (drive or external encoder) In the "Number of the measuring input" selection box, select "1" (measuring input 1).

Drive connection using PWM (pulse width modulation)

Note that only travel in the positive direction is possible with a drive connection using the integrated PWM function of the compact CPU.

Configuration	
CPU 1511C-1 PN/CPU 1512C-1 PN	Technology object Speed axis
Pulse generators (PTO/PWM) > PTO1…4/PWM1…4 > General	–
Select "Pulse width modulation PWM" mode	
Pulse generators (PTO/PWM) > PTO1…4/PWM1…4 > Hardware inputs/outputs	
Select the hardware output to be used for pulse output.	
Select whether the set hardware output is to work as a fast push-pull switch or as P switch.	
Pulse generators (PTO/PWM) > PTO1…4/PWM1…4 > Parameters	**Hardware interface > Drive**
Select "S7 analog output" output format	Select analog drive connection For the selection of the analog output, create a PLC tag of the "Int" type with corresponding address. The offset for the PLC tag of the control interface of the PWM channel is 2. To activate the output of the PWM signal, set the following two bits of the control interface of the PWM channel in the user program: • SW_ENABLE (= Bit 0 in Byte 9) • TM_CTRL_DQ (= Bit 1 in Byte 9) The offset for byte 9 to the start address of the PWM channel is 9.

Configuration	
CPU 1511C-1 PN/CPU 1512C-1 PN	Technology object
	Speed axis
Select minimum pulse width of 0 µs	–
Select required period duration (e.g. 100 µs)	
Pulse generators (PTO/PWM) > PTO1…4/PWM1…4 > I/O addresses	
Select the organization block "MC-Servo" for the input and output addresses. The "TPA OB Servo" process image is selected automatically for the input and output addresses by selecting the organization block.	

"–" No configuration for technology object is required for these parameters

Encoder connection via HSC (High-speed counter)

Configuration		
CPU 1511C-1 PN/CPU 1512C-1 PN	Technology object	
	Axis	External encoder
High-speed counter (HSC) > HSC 1…6 > General > Enable	–	–
Enable high-speed counter		
High-speed counter (HSC) > HSC 1…6 > Basic parameters > Operating mode		
Select "Position input for Motion Control" mode		
High-speed counter (HSC) > HSC 1…6 > Basic parameters > Module parameters	**Hardware interface > Encoder**	**Hardware interface > Encoder**
–	Select "Encoder" data connection and the high-speed counter activated and configured as encoder on the CPU	Select "Encoder" data connection and the high-speed counter activated and configured as encoder on the CPU
Signal type • Incremental encoder	Select encoder type according to the device configuration of the CPU [1)] • Incremental	Select encoder type according to the device configuration of the CPU [1)] • Incremental
–	**Hardware interface > Data exchange with encoder**	**Hardware interface > Data exchange**
	Telegram "DP_TEL83_STANDARD" is automatically selected after the selection of the encoder.	Telegram "DP_TEL83_STANDARD" is automatically selected after the selection of the encoder.
	Clear the check box "Automatic data exchange for encoder values (online)"	Clear the check box "Automatic data exchange for encoder values (online)"
	Select the check box "Automatic data exchange for encoder values (offline)" If the check box is cleared, you can manually match the parameters described and identified in this table.	Select the check box "Automatic data exchange for encoder values (offline)" If the check box is cleared, you can manually match the parameters described and identified in this table.
	Select rotary measuring system type [1)]	Select rotary measuring system type [1)]

Configuration		
CPU 1511C-1 PN/CPU 1512C-1 PN	Technology object	
	Axis	External encoder
Signal evaluation • Single • Double • Quadruple	Enter fine resolution according to the configured signal evaluation set for the high-speed counter (HSC) [1)] • 0 = Single • 1 = Double • 2 = Quadruple	Enter fine resolution according to the configured signal evaluation set for the high-speed counter (HSC) [1)] • 0 = Single • 1 = Double • 2 = Quadruple
Enter increments per revolution	Enter increments per revolution corresponding to device configuration for CPU (1:1) [1)]	Enter increments per revolution corresponding to device configuration for CPU (1:1) [1)]
–	**Hardware interface > Data exchange with the drive**	–
Enter reference speed corresponding to configuration for technology object (1:1)	Enter reference speed	
–	**Homing**	**Homing**
Select the homing signal for homing mark 0: • Signal N of the incremental encoder • DI0 (can be set with the hardware inputs/outputs) In addition, configure the input delay for the selected hardware input. You configure the input delay in the device configuration at the corresponding DI channel (DI 16/DQ 16 > Inputs > Channel > Input parameters > Input delay).	Use the homing mode "Use zero mark via PROFIdrive telegram".	Use the homing mode "Use zero mark via PROFIdrive telegram".
High-speed counter (HSC) > HSC 1…6 > I/O addresses	–	–
The organization block ("MC-Servo") and the process image ("TPA OB Servo") are selected automatically for the input and output addresses by selecting the HSC channel for the technology object.		

[1)] Parameters are automatically applied when "Automatic data exchange for encoder values (offline)" is selected

"–" No configuration for CPU/technology object is required for these parameters

Additional configuration for use with the technology object measuring input

Configuration	
Technology module CPU 1511C-1 PN/CPU 1512C-1 PN	Technology object Measuring input
High-speed counter (HSC) > HSC 1…6 > Hardware inputs/ outputs	**Hardware interface > Measuring input**
Select the hardware input of the measuring input. In addition, configure the input delay for the selected hardware input. You configure the input delay in the device configuration at the corresponding DI channel (DI 16/DQ 16 > Inputs > Channel > Input parameters > Input delay).	Measuring using PROFIdrive telegram (drive or external encoder) In the "Number of the measuring input" selection box, select "1" (measuring input 1).

Connect drive/encoder via data block (S7-1500)

Creating the data block for data connection

1. Create a new data block of type "Global DB".
2. Select the data block in the project tree and select "Properties" from the shortcut menu.
3. Disable the following attributes under Attributes and accept the change with "OK":
   – "Only store in load memory"
   – "Data block write-protected in the device"
   – "Optimized block access" for technology version < V4.0
4. Open the data block in the block editor.
5. Insert a tag structure of type "PD_TELx" textually in the block editor.

This tag structure contains the "Input" tag structure for the input area of the telegram and the "Output" tag structure for the output area of the telegram.

Note

"Input" and "Output" relate to the view of the closed loop position control. For example, the input area contains the actual values of the drive and the output area contains the setpoints for the drive.

The data block may contain the data structures of multiple axes and encoders and other contents.

Configuring data connection via a data block

1. Open the configuration window "Hardware interface > Drive" or "Hardware interface > Encoder".
2. Select the entry "Data block" from the "Data block" drop-down list.
3. In the "Data block" field, select the previously created data block.
   Open this data block and select the tag name defined for the drive and encoder.

Programming MC-PreServo and MC-PostServo

1. Assign the previously defined PLC tag of the input range of the data block to MC-PreServo .
2. Assign the previously defined PLC tag of the output range of the data block to MC-PostServo.

NOTICE
Machine damage Improper manipulation of drive and encoder telegrams may result in unwanted drive motions. Check your user program for consistency in the drive and encoder connection.

An application example for the use of MC-PreServo and MC-PostServo is available at:

https://support.industry.siemens.com/cs/document/109741575 (https://support.industry.siemens.com/cs/document/109741575)

See also

PROFIdrive telegrams (Page 1636)

Data connection drive/encoder via data block (Page 1650)

Parameter view (S7-1500)

The Parameter view provides you with a general overview of all relevant parameters of a technology object. You obtain an overview of the parameter settings and can easily change them in offline and online mode.

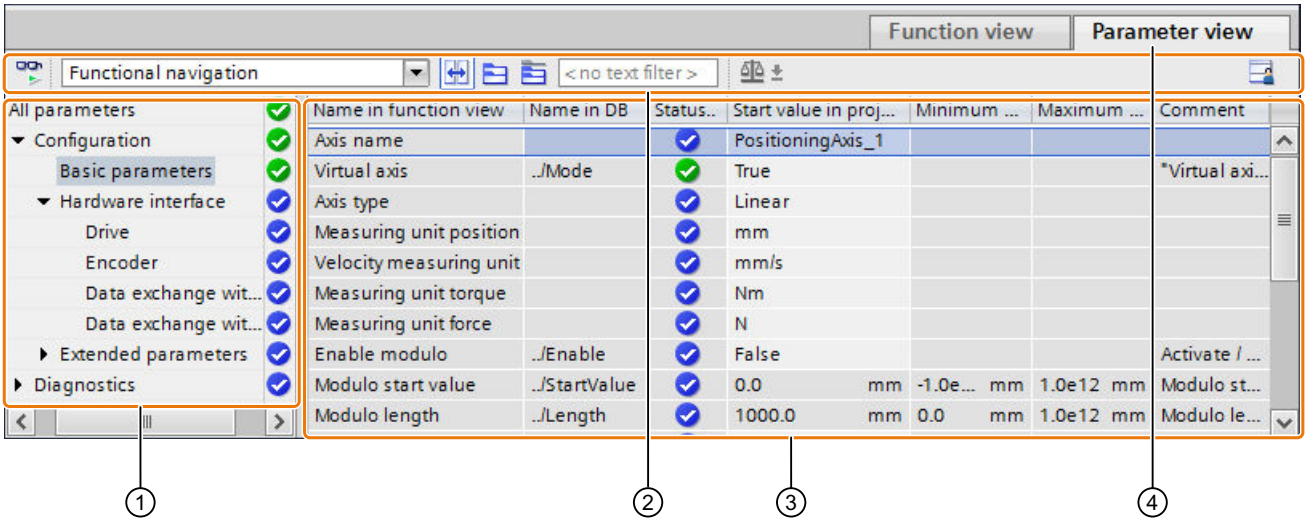

① Navigation (Page 1912)

② Toolbar (Page 1911)

③ Parameter table (Page 1912)

④ "Parameter view" tab

Function scope

The following functions are available for analyzing the parameters of the technology objects and for enabling targeted monitoring and modification.

Display functions:

- Display of parameter values in offline and online mode
- Display of status information of the parameters
- Display of value deviations and option for direct correction
- Display of configuration errors
- Display of value changes as a result of parameter dependencies
- Display of all memory values of a parameter: Start value PLC, Start value in project, Monitor value
- Display of the parameter comparison of the memory values of a parameter

Operator control functions:

- Navigation for quickly changing between the parameters and parameter structures.
- Text filter for faster searches for particular parameters.
- Sorting function for customizing the order of parameters and parameter groups to requirements.

- Memory function for backing up structural settings of the Parameter view.
- Monitoring and modifying of parameter values online.
- Function for saving a snapshot of parameter values of the CPU in order to capture momentary situations and to respond to them.
- Function for applying a snapshot of parameter values as start values.
- Download of modified start values to the CPU.
- Comparison functions for comparing parameter values with one another.

Structure of the parameter view (S7-1500)

Toolbar (S7-1500)

The following functions can be selected in the toolbar of the parameter view.

Symbol	Function	Explanation
	Monitor all	Starts monitoring of the visible tags in the active table.
Functional navigation	Select navigation structure	Toggle between function-based navigation and the view of the data structure of the technology data block.
	Couples the function view and parameter view for the objects selected in the navigation	Enables the targeted toggling between the parameter view and function-based view.
	Collapse/expand all nodes and objects	Collapses or expands all nodes and objects of the navigation or the data structure in the currently active view.
	Collapse/expand the nodes below the marked nodes	Collapses or expands the marked nodes and objects of the navigation or the data structure in the currently active view.
< no text filter >	Text filter...	After entry of a character string: Display of all parameters containing the entered string in one of the currently visible columns.
	Selection of compare values	Selection of parameter values that are to be compared with one another in online mode (Start value in project, Start value PLC) Only in online mode.
	Save window settings	Saves your display settings for the Parameter view (e.g. selected navigation structure, activated table columns, etc.)

Navigation (S7-1500)

Within the "Parameter view" tab, the following alternative navigation structures can be selected.

Navigation		Explanation
Functional navigation	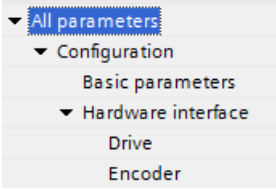	In the functional navigation, the structure of the parameters is based on the structure in the configuration window ("Function view" tab), commissioning window and diagnostics window.
Data structure	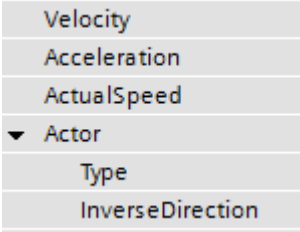	In the "Data structure" navigation, the structure of the parameters is based on the structure of the technology data block.

You can use the "Select navigation structure" drop-down list to toggle the navigation structure.

Parameter table (S7-1500)

The table below shows the meaning of the individual columns of the parameter table. You can show or hide the columns as required.

Column	Explanation	Offline	Online
Name in function view	Name of the parameter in the function view. The display field is empty for parameters that are not configured via the technology object.	X	X
Name in DB	Name of the parameter in the technology data block. If the parameter is part of a structure or UDT, the prefix "../" is added. The display field is empty for parameters that are not contained in the technology data block.	X	X
Full name in DB	Complete path of the parameter in the instance data block. The display field is empty for parameters that are not contained in the technology data block.	X	X
Status of configuration	Display of the completeness of the configuration using status symbols.	X	-
Compare result	Result of the "Compare values" function. This column is displayed when there is an online connection.	-	X
Start value in project	Configured start value in project. Error indication if entered values have a syntax or process-related error.	X	X
Default value	Value that is pre-assigned to the parameter. The display field is empty for parameters that are not contained in the technology data block.	X	X

Column	Explanation	Offline	Online
Start value PLC	Start value in the CPU. This column is displayed when there is an online connection.	-	X
Monitor value	Current value in the CPU. This column is displayed when there is an online connection.	-	X
Modify value	Value that is to be used to change the monitor value. This column is displayed when there is an online connection.	-	X
Minimum value	Minimum process-related value of the parameter. If the minimum value is dependent on other parameters, it is defined: • Offline: by the start value in the project. • Online: by the monitor values.	X	X
Maximum value	Maximum process-related value of the parameter. If the maximum value is dependent on other parameters, it is defined: • Offline: by the start value in the project. • Online: by the monitor values.	X	X
Setpoint	Designates the parameter as a setpoint. These parameters can be initialized online.	X	X
Data type	Data type of the parameter. The display field is empty for parameters that are not contained in the technology data block.	X	X
Retain	Designates the value as a retentive value. The values of retentive parameters are retained even after the voltage supply is switched off.	X	X
Accessible from HMI	Indicates whether the HMI can access this parameter during runtime.	X	X
Visible in HMI	Indicates whether the parameter is visible in the selection list of the HMI by default.	X	X
Comment	Brief description of the parameter.	X	X

X This function is visible in offline/online mode.

- This function is not visible in offline/online mode.

Opening the parameter view (S7-1500)

Requirement

The technology object was added in the project navigator.

Procedure

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.

3. Double-click the "Configuration" object.
4. Select the "Parameter view" tab in the top right corner.

Result

The Parameter view opens. Each displayed parameter is represented by one row in the parameter table.

The displayable parameter properties (table columns) vary depending on whether you are working with the Parameter view in offline or online mode.

In addition, you can selectively display and hide individual table columns.

Working with the parameter view (S7-1500)

The following table provides an overview of the functions of the Parameter view in online and offline mode described below:

Function/action	Offline	Online
Filtering the parameter table (Page 1914)	X	X
Sorting the parameter table (Page 1915)	X	X
Transferring parameter data to other editors (Page 1916)	X	X
Indicating errors (Page 1916)	X	X
Editing start values in the project (Page 1916)	X	X
Monitoring values online in the parameter view (Page 1917)	-	X
Modifying values (Page 1918)	-	X
Comparing values (Page 1919)	-	X

X This function is possible in offline/online mode.

- This function is not possible in offline/online mode.

Filtering the parameter table (S7-1500)

You can filter the parameters in the parameter table in the following ways:

- With the text filter
- With the subgroups of the navigation

Both filter methods can be used simultaneously.

With the text filter

The text filter can only be use on texts in displayed parameter lines and displayed columns.

To work with the text filter, follow these steps:

1. To filter a desired character string, enter it in the "Text filter..." input box.
   The parameter table displays only the parameters containing the character string.

To reset text filtering, the following options are available:

- Select another parameter group in the navigation.
- Switch between data navigation and functional navigation, or vice versa.

With the subgroups of the navigation

To filter the navigation with the subgroups, follow these steps:

1. Click the desired parameter group in the navigation, e.g., "Static".
   The parameter table now only shows the "static" parameters. You can select further subgroups for some groups of the navigation.
2. If all parameters are to be shown again, click "All parameters" in the navigation.

Sorting the parameter table (S7-1500)

The values of the parameters are arranged in rows. The parameter table can be sorted by any displayed column.

- In columns containing numerical values, sorting is based on the magnitude of the numerical value.
- In text columns, sorting is alphabetical.

Sort column-by-column

1. Position the cursor in the header cell of the desired column.
   The background of this cell turns blue.
2. Click the column header.

Result

The entire parameter table is sorted by the selected column. A triangle with tip facing up appears in the column header.

Clicking the column header again changes the sorting as follows:

Symbol	Description
▲	Parameter table is sorted in ascending order.
▼	Parameter table is sorted in descending order.
No symbol	The sorting is removed again. The parameter table shows the default display.

The "../" prefix in the "Name in DB" column is ignored when sorting.

Transferring parameter data to other editors (S7-1500)

You can paste parameters of the parameter view in the following editors:

- Program editor
- Watch table
- Signal table for Trace

The following options are available for pasting:

- Drag-and-drop
- <Ctrl+C>/<Ctrl+V>
- Copy/Paste via shortcut menu

Indicating errors (S7-1500)

Error display

Parameter assignment errors that result in compilation errors (e.g. limit violation) are indicated in the Parameter view.

Every time a value is entered in the Parameter view, a check is made for process-related and syntax errors and displayed with the following indicators:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns
- Table field with red background
  If you click the bad field, a roll-out error message appears with information of the permissible value range or the required syntax (format)

Compilation error

From the error message of the compiler, you can directly open the Parameter view (functional navigation) containing the parameter causing the error in situations where the parameter is not displayed in the configuration window.

Editing start values in the project (S7-1500)

With the Parameter view, you can edit the start values in the project in offline mode and online mode.

- You make value changes in the "Start value in project" column of the parameter table.
- In the "Status of configuration" column of the parameter table, the progress of the configuration is indicated by the familiar status symbols from the configuration window of the technology object.

Boundary conditions

- If other parameters depend on the parameter whose start value was changed, the start value of the dependent parameters are also adapted.
- If a parameter of a technology object is not editable, it is also not editable in the parameter view. The ability to edit a parameter can also depend on the values of other parameters.

Defining new start values

To define start values for parameters in the Parameter view, follow these steps:

1. Open the Parameter view of the technology object.
2. Enter the desired start values in the "Start value in project" column. The value must match the data type of the parameter and must not exceed the value range of the parameter. The limits of the value range can be seen in the "Maximum value" and "Minimum value" columns.

The "Status of configuration" column indicates the progress of the configuration with colored symbols.

Following adaptation of the start values and downloading of the technology object to the CPU, the parameters take the defined value at startup if they are not declared as retentive ("Retain" column).

Error display

When a start value is input, a check is made for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Red background in the "Start value in project" field
  If you click on the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Correcting bad start values

1. Correct bad start values using information from the roll-out error message.
   Red error symbol, red field background, and roll-out error message are no longer displayed.

The project cannot be successfully compiled unless the start values are error-free.

Monitoring values online in the parameter view (S7-1500)

You can monitor the values currently taken by the parameters of the technology object in the CPU (monitor values) directly in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The Parameter view of the technology object is open.

Procedure

As soon as the Parameter view is online, the following columns are additionally displayed:

- Compare result
- Start value PLC
- Monitor value
- Modify value
- Selection for transmission

The "Monitor value" column shows the current parameter values on the CPU.

Display

All columns that are only available online are displayed in color as follows:

Color	Description
	The values are modifiable.
	These values cannot be changed.

See also

Parameter table (Page 1911)

Modifying values (S7-1500)

With the Parameter view, you can modify values of the technology object in the CPU. You can assign values to a parameter once (Modify value) and modify them immediately. The modify request is executed as quickly as possible without reference to any particular point in the user program.

DANGER

Danger when modifying

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you use the "Modify" function.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The Parameter view of the technology object is open.
- The parameter is modifiable. (Associated field in the "Modify value" column has a corresponding background).

Procedure

To modify values immediately, follow these steps:

1. Enter the desired modify value in the "Modify values" column of the parameter table.
   The parameter is modified once and immediately with the specified value. You can monitor the value in the "Monitor values" column. The check box for modifying in the "Selection for transmission" column is automatically cleared after the modify request is complete.

Error display

When a modify value is input, a check is made immediately for process-related and syntax errors and the result is indicated.

You can recognize incorrect modify values as follows:

- The field "Modify value" is displayed with red background color.
- If you click in the incorrect field, a roll-out error message appears with information on the permissible value range or the necessary syntax.

Bad modify values

- Modify values with process-related errors can be transmitted.
- Modify values with syntax errors cannot be transmitted.

Comparing values (S7-1500)

You can use comparison functions to compare the following memory values of a parameter:

- Start value in project
- Start value PLC

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The Parameter view of the technology object is open.

Procedure

To compare the start values on the various target systems, follow these steps:

1. Click the "Selection of compare values" icon .
   A selection list containing the comparison options opens:
   – Start value in project - Start value in CPU (default setting)
2. Select the desired comparison option.
   The selected comparison option is executed as follows:
   – A scales symbol appears in the header cells of the two columns selected for comparison.
   – Symbols are used in the "Compare result" column to indicate the result of the comparison of the selected columns.

Symbol in "Compare result" column

Symbol	Meaning
	The compare values are equal and error-free.
	The compare values are not equal and error-free.
	At least one of the two compare values has a process-related or syntax error.
	The comparison cannot be performed. At least one of the two comparison values is not available (e.g. snapshot).

Symbol in the navigation

The symbols are shown in the same way in the navigation if the compare result applies to at least one of the parameters below the displayed navigation structure.

6.1.2.5 Programming (S7-1500)

The "Programming" section contains general information on supplying and evaluating the Motion Control instructions and on the technology data block.

You can find an overview of the Motion Control instructions in the Functions (Page 1626) section.

You can use Motion Control instructions in the user program to assign jobs to the technology object. You define the job using the input parameters of the Motion Control instructions. The current job status is indicated in the output parameters.

The technology data block is available to you as an additional interface to the technology object.

Technology data block (S7-1500)

The properties of real objects (e.g. drives) are configured by means of the technology objects and saved in a technology data block. The technology data block contains all configuration data, setpoint and actual values, and status information of the technology object. The TIA Portal automatically creates the technology data block when the technology object is created. You access the data of the technology data block (read/write access) with your user program.

A listing and description of the tags can be found in the Appendix (Page 2012).

Evaluating the technology data block (S7-1500)

Access to data in the technology data block occurs in accordance with the access to standard data blocks. Only tags with elementary data types can be accessed in the technology data block. It is not possible to access tags with composite data types (such as STRUCT, ARRAY).

Reading values from the technology data block

In your user program you can read actual values (e.g. current position) and status information, or detect error messages in the technology object. When you program a query in your user program (e.g. current velocity), the value is directly read from the technology object.

Reading values from the technology data block takes longer than for other data blocks. If you use these tags several times in a single cycle of your user program, it is recommended to copy the tag values to local tags, and use the local tags in your program.

Writing values to the technology data block

The configuration of the technology object in the TIA Portal is used to write the corresponding data to the technology data block. After they have been loaded into the CPU, these data are stored in the CPU on the SIMATIC Memory Card (load memory).

In the following cases, it may be necessary for the user program to write values to the technology data block:

- Adaptation of the configuration of the technology object (e.g. dynamic limits, software limit switches)
- Use of overrides
- Adaptation of position control (e.g. "Kv" parameter)

Changes to values in the technology data block by the user program can take effect at various points in time. The relevant property of the individual tags can be found in their descriptions in the Appendix (Page 2012):

Effectiveness of changes	Description	
Direct (DIR)	You write changes using direct assignments. The changes are applied only at the start of the next MC-Servo [OB91]. The changes are retained until the next POWER OFF of the CPU or restart of the technology object.	
	LREAL (e.g. <TO>.Override.Velocity)	The technology object performs a range check on the written value, and immediately starts using the new value. If range limits are violated when writing, the technology object automatically corrects the values. If the value is below the range, then the value is set to the low limit of the range; if the range is exceeded, then the value is set to the high limit of the range.
	DINT/BOOL (e.g. <TO>.PositionLimits_SW.Active)	Changes are only permitted in the defined value range. Value changes outside the value range are not applied. If you enter invalid values, the programming error OB (OB 121) is started.
When Motion Control instruction is called (CAL) (e.g. <TO>.Sensor[n].ActiveHoming.HomePositionOffset)	You write changes using direct assignments. The changes are applied at the start of the next MC-Servo [OB91] after the call of the corresponding Motion Control instruction in the user program. The changes are retained until the next POWER OFF of the CPU or restart of the technology object.	
Restart (RES) (e.g. <TO>.Homing.AutoReversal)	Since restart-relevant tags have dependencies on other tags, value changes cannot be applied at any arbitrary time. The changes are only used after reinitialization (restart) of the technology object. During a restart the technology object is reinitialized with the data in load memory. You therefore write changes to the start value in the load memory with the extended instruction "WRIT_DBL" (write to data block in load memory). You trigger the restart in your user program using the Motion Control instruction "MC_Reset" with parameter "Restart" = TRUE. Additional information regarding the restart can be found in the section Restarting technology objects (Page 1946).	
Read only (RON) (e.g. <TO>.Position)	The tag cannot and must not be changed during runtime of the user program.	

Note

Save changes with "WRIT_DBL"

Changes to tags immediately in effect are lost on POWER OFF of the CPU, or restart of the technology object.

If changes in the technology data block should also be retained after POWER OFF of the CPU, or restart of the technology object, you must write the changes to the start value in the load memory with the extended instruction "WRIT_DBL".

Note

Using the "READ_DBL" and "WRIT_DBL" data block functions

The "READ_DBL" and "WRIT_DBL" data block functions may only be used on individual tags in conjunction with the tags of the technology object. The "READ_DBL" and "WRIT_DBL" data block functions must not be applied to data structures of the technology object.

Isochronous evaluation of data

If you want to process data of the technology data block in isochronous mode from a Motion Control application cycle, there is the option of evaluating this data in the MC-PreServo [OB67]/ MC-PostServo [OB95] as of technology version V3.0.

See also

Organization Blocks for Motion Control (Page 1696)

Evaluate StatusWord, ErrorWord and WarningWord (S7-1500)

To be able to symbolically use individual status and error information from the "StatusWord", "ErrorWord" and "WarningWord" data double words, you can evaluate them as described below. For consistent evaluation, you should avoid using bit addressing to access these data double words in the technology data block. Access to an individual bit in the technology data block only lasts as long as the access to the entire data word.

When required, copy the required data double word to a tag of a data structure and query the individual bits of the tag.

The allocation of the individual bits in the data double words can be found in the Appendix (Page 2012) in the description of the tags of the corresponding technology object.

Requirements

The technology object has been created.

Procedure

To evaluate the individual bits in the data word "StatusWord", follow these steps:

1. Crate a global data structure. Name the data structure, e.g. as "Status".
2. Create a double word (DWORD) in the data structure "Status". Name the double word, e.g. as "Temp".
3. Create 32 Boolean tags in the "Status" data structure. You can obtain a clearer overview by giving the individual Boolean tags identical names as the bits in the technology DB (e.g. name the fifth Boolean tag "HomingDone").
4. If needed, copy the tag "<TO>.StatusWord" from the technology data block to the double word "Temp" in your data structure.

5. Copy the individual bits of double word "Temp" to the corresponding Boolean tags with bit accesses.
6. Use the Boolean tags to query the status bits.

Evaluate the data words "ErrorWord" and "WarningWord" as specified in steps 1 to 6.

Example

The following example shows how you can read out and save the fifth bit "HomingDone" of the data word "StatusWord":

SCL	Explanation
`#Status.Temp := "TO".StatusWord;`	`//Copy status word`
`#Status.HomingDone := #Status.Temp.%X5;`	`//Copy individual bits per bit access`

STL	Explanation
`L "TO".StatusWord`	`//Copy status word`
`T #Status.Temp`	
`U #Status.Temp.%X5`	`//Copy individual bits per bit access`
`= #Status.HomingDone`	

Change restart-relevant data (S7-1500)

In order to change restart-relevant data in the technology data block, write to the starting values of the tags in load memory using the extended instruction "WRIT_DBL". In order for the changes to be applied, a restart of the technology object must be performed.

Whether changes to the value of a tag are relevant for the restart can be found in the description of the tags in the Appendix (Page 2028).

Requirement

The technology object has been created.

Procedure

To change restart-relevant data, proceed as follows:

1. Create a data block and fill it with the restart-relevant values, that you want to change in the technology data block. In doing so, the data types must match the tags to be changed.
2. Write the values from your data block to the starting values of the tags of the technology data block in load memory, using the extended instruction "WRIT_DBL".
   If restart-relevant data were changed, this will be indicated in the "<TO>.StatusWord.X3 (OnlineStartValuesChanged)" tag of the technology object.
3. Perform a restart of the technology object using the Motion Control instruction "MC_Reset" with parameter "Restart" = TRUE.

After the restart of the technology object, the new value is transferred into the technology data block in work memory, and is effective.

Motion Control instructions (S7-1500)

Motion Control instruction parameters (S7-1500)

The individual Motion Control instructions are described in detail in the section AUTOHOTSPOT.

When creating your user program, take the following explanations of the Motion Control instruction parameters into account.

Reference to the technology object

The technology object is specified for the Motion Control instruction as follows:

- **Parameter "Axis"**
  The "Axis" input parameter of a Motion Control instruction contains a reference to the technology object that is to execute the corresponding job.
  The corresponding technology object is also referenced in the following parameters:
  - Parameter "Master"
  - Parameter "Slave"
  - Parameter "Cam"
  - Parameter "MeasuringInput"
  - Parameter "OutputCam"
  - Parameter "CamTrack"

  As of technology version V3.0, the reference to the technology object can also be specified with the data type "DB_ANY".

Job start and transfer of input parameters of a Motion Control instruction

For the start of jobs and the transfer of modified parameter values, a distinction is made between the following Motion Control instructions:

- **Motion Control instructions with "Execute" parameter**
  With a positive edge at the "Execute" parameter, the job is started and the existing values for the input parameters are transferred.
  Subsequently changed parameter values are not transferred until the next job start.
  Resetting the "Execute" parameter does not end the job, but it does affect the display duration of the job status. As long as "Execute" is set to "TRUE", the output parameters will be updated. If "Execute" is reset before the completion of a job, the parameters "Done", "Error" and "CommandAborted" are correspondingly set for only one call cycle.
- **Motion Control instructions with "Enable" parameter**
  The job is started by setting the "Enable" parameter.
  As long as "Enable" = TRUE, the job remains active and changed parameter values will be transferred each time the instruction is called in the user program.
  The job is ended by resetting the "Enable" parameter.
  The input parameters "JogForward" and "JogBackward" of the Motion Control instruction "MC_MoveJog" correspond in their behavior to the "Enable" parameter.

Job status

The following output parameters indicate the status of the job execution:

- **Motion Control instructions with "Done" parameter**
  The normal completion of a job is indicated with parameter "Done" = TRUE.
- **Motion Control instructions without "Done" parameter**
  The achievement of the job objective is indicated by other parameters (e.g. "Status", "InVelocity"). For more information, refer to the section Tracking running jobs (Page 1935).
- **Parameter "Busy"**
  As long as a job is in progress, the "Busy" parameter shows the value "TRUE". If a job was ended or canceled, "Busy" shows the value "FALSE".
- **Parameter "CommandAborted"**
  If a job was canceled by another job, the "CommandAborted" parameter shows the value "TRUE".
- **Parameter "Error"**
  If a Motion Control instruction error occurs, the "Error" parameter shows the value "TRUE". The ErrorID parameter indicates the cause of the error.

As long as the "Execute" or "Enable" parameter is set to "TRUE", the output parameters will be updated. Otherwise, the parameters "Done", "Error" and "CommandAborted" are correspondingly set for only one cycle.

Abort of running jobs

An active Motion Control job is aborted by the start of a new Motion Control job. In the process, the current dynamic setpoints (acceleration, deceleration, jerk, velocity) are set to the values of the overriding job.

Example of parameter behavior

The behavior of the parameters of Motion Control instructions is shown in the following chart using the example of two "MC_MoveAbsolute" jobs.

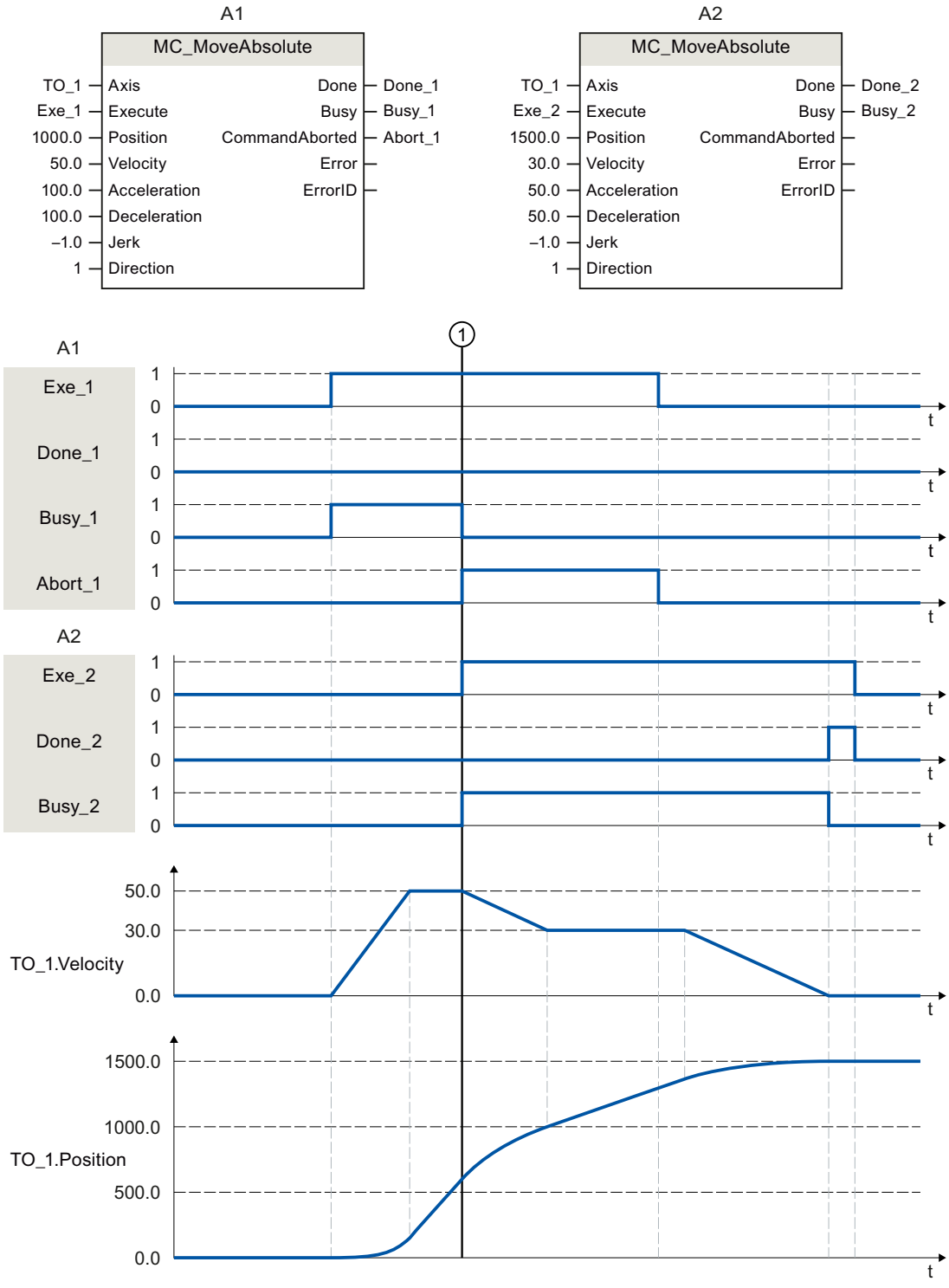

Using "Exe_1", an "MC_MoveAbsolute" job (A1) with target position 1000.0 is initiated. "Busy_1" is set to "TRUE". The axis is accelerated to the specified velocity and moved to the target position (see TO_1.Velocity and TO_1.Position). Before the target position is reached, the job is overridden at time ① by another "MC_MoveAbsolute" job (A2). The termination is signaled via "Abort_1", and "Busy_1" is set to "FALSE". The axis is braked to the velocity specified in A2 and moved to the new target position 1500.0. When the axis reaches the target position, this is signaled via "Done_2".

Non position-controlled operation

The position control of the axis can be deactivated with the following parameters:

- MC_Power.StartMode = 0
- MC_MoveVelocity.PositionControlled = FALSE
- MC_MoveJog.PositionControlled = FALSE

For more information, refer to the section Non position-controlled operation (Page 1932).

Add Motion Control instructions (S7-1500)

You add Motion Control instructions to a program block in the same way as other instructions. You control all available functions of the technology object using the Motion Control instructions.

Requirements

The technology object was created.

Procedure

To add Motion Control instructions in your user program, proceed as follows:

1. Double click your program block in the project tree (the program block must be called in the cyclical program).
   The program block is opened in the programming editor, and the available instructions are displayed.
2. In the "Instructions" task card, open the "Technology > Motion Control" folder.
3. Using drag-and-drop, move the Motion Control instruction, e.g. "MC_Power", to the desired segment of the program block.
   The "Call options" dialog opens.
4. In the dialog, specify a name and a number for the instance data block of the Motion Control instruction.

5. Click "OK".
   The Motion Control instruction "MC_Power" is inserted into the network.

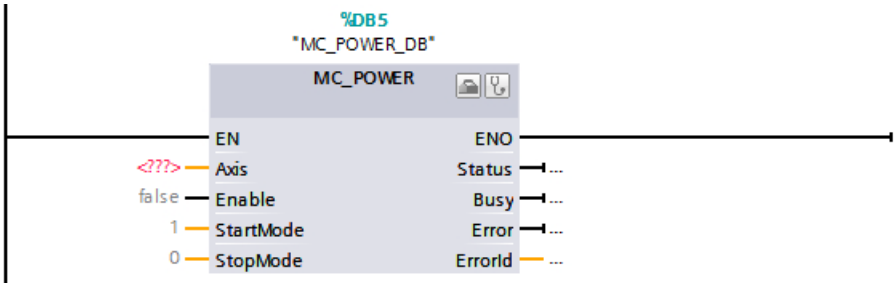

   The instance data block is automatically created under "Program Blocks > System Blocks > Program Resources".

6. Input parameters without a default value (e.g. "Axis"), must be assigned. Select the technology object in the project tree and move the technology object onto <...> at the "Axis" parameter using drag-and-drop.

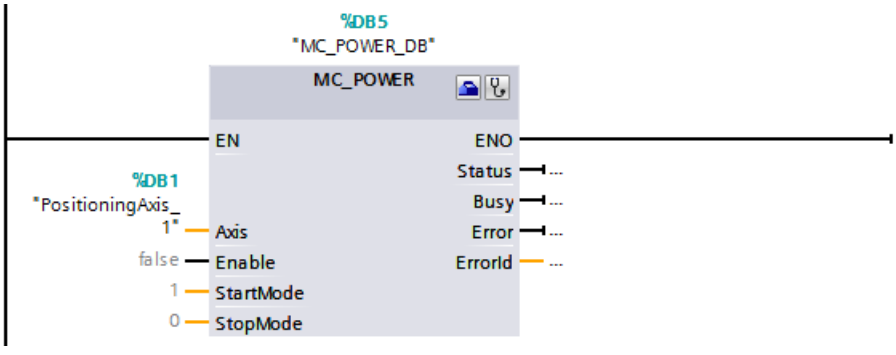

   Once the technology object is specified in the "Axis" parameter, the following buttons are available to you:

	To open the configuration of the technology object, click on the toolbox icon.
	To open the diagnostics of the technology object, click on the stethoscope icon.

7. Add additional Motion Control instructions in accordance with steps 3 through 6.

See also

Tracking active jobs (Page 1935)

Tags of the positioning axis/synchronous axis technology object (Page 2028)

Parameter transfer for function blocks (S7-1500)

If you want to reuse a function block with Motion Control instructions for different technology objects, create an input parameter of the data type of the respective technology object in the block interface of the calling function block. You assign the data type in the block interface with a direct input. The parameter is then transferred as reference to the technology object to the "Axis" parameter of the Motion Control instructions. The data types of technology objects correspond to the structure of the associated technology data block.

In contrast to standard data types, the data types for technology objects are always passed on as pointers to the function block (Call by reference). This remains true if you declare the data types of the technology objects in "Input" area of the block interface. Write access to function blocks always leads directly to modification of the referenced technology object.

Example 1: Tag transfer with specific data type

By specifying the data type, you can address the tags of the technology object in the function block (<parameters of the block interface>.<tag of the technology object>). The data types for the reference to the technology objects are available in the appendix (Page 2147).

The following table shows the declaration of the tags used:

Tag	Declaration	Data type	Description
axis	Input	TO_PositioningAxis	Reference to the technology object
on	Input	BOOL	Signal to enable the axis
actPosition	Output	LReal	Query of the actual position from the technology data block
instMC_POWER	Static	MC_POWER	Multinstance of the Motion Control instruction "MC_Power"

The following SCL program shows the tag transfer with a specific data type:

SCL	Explanation
`#instMC_POWER(Axis := #axis, Enable := #on);`	`//Call of the Motion Control instruction "MC_Power" with enable of the axis`
`#actPosition := #axis.ActualPosition;`	`//Query of the actual position from the technology data block`

Example 2: Tag transfer with "DB_Any"

The data type "DB_Any" provides a further option for the transfer of specific data types of the technology object. Unlike the data types of the technology object in the program, "DB_Any" can be assigned during runtime.

This example shows how you can program variable switching of up to four cams at "MC_CamIn".

To do this, tags of the data type "DB_Any" are first created as input parameter of the block. The cam to be used is assigned by an additional input parameter.

The following table shows the declaration of the tags used:

Tag	Declaration	Data type	Description
cam1	Input	DB_ANY	Cam 1
cam2	Input	DB_ANY	Cam 2
cam3	Input	DB_ANY	Cam 3
cam4	Input	DB_ANY	Cam 4
camToUse	Input	Int	Selection of cam 1 to 4
instMC_CAMIN	Static	MC_CAMIN	Multinstance of the Motion Control instruction "MC_CamIn"
tempCamSel	Temp	DB_ANY	Current cam

The following SCL program shows the tag transfer with "DB_Any":

```
SCL                                                  Explanation
                                                     //Selection of the desired cam 1..4
                                                     //Using an input tag of data type Int
CASE #camToUse OF
     1:  #tempCamSel := #cam1;                       //Instruction for scenario 1
     2:  #tempCamSel := #cam2;                       //Instruction for scenario 2
     3:  #tempCamSel := #cam3;                       //Instruction for scenario 3
     4:  #tempCamSel := #cam4;                       //Instruction for scenario 4
   ELSE                                              //Instruction for Int ≤ 0 or Int > 4
     #tempCamSel := #cam1;                           //Corresponds to default cam 1
END_CASE;
                                                     //Call of the Motion Control instruction "MC_Ca-
                                                     mIn" with variable transfer of the cam technology
                                                     object using the temporary tag "tempCamSel"

#instMC_CAMIN(Master := "PositioningAxis_1",         //Direct assignment of the technology object of
                                                     the leading axis
              Slave := "SynchronousAxis_1",          //Direct assignment of the technology object of
                                                     the following axis
              Cam := #tempCamSel);                   //Indirect assignment of the cam technology object
```

Additional information

You can find more program examples for the use of the data type "DB_Any" in the following FAQ entry:

https://support.industry.siemens.com/cs/document/109750880 (https://support.industry.siemens.com/cs/ww/en/view/109750880)

Non position-controlled operation (S7-1500)

The closed loop position control of an axis can be switched off/switched with the following Motion Control instructions:

- MC_Power
- MC_MoveVelocity
- MC_MoveJog
- MC_MotionInVelocity

The non-position-controlled operation is displayed with "<TO>.StatusWord.X28 (NonPositionControlled)" (TRUE) = TRUE.

MC_Power

The axis is enabled without closed loop position control with "MC_Power" and the parameter "StartMode" = 0. The closed loop position control remain switched off until a different Motion Control instruction changes the status of the closed loop position control.

MC_MoveVelocity and MC_MoveJog

A "MC_MoveVelocity" or "MC_MoveJog" job with "PositionControlled" = FALSE forces non-position-controlled operation.

A "MC_MoveVelocity" or "MC_MoveJog" job with "PositionControlled" = TRUE forces position-controlled operation.

The selected mode remains in effect after the job is completed.

MC_MotionInVelocity and MC_MotionInPosition

A "MC_MotionInVelocity" job with "PositionControlled" = FALSE forces non-position-controlled operation.

A "MC_MotionInVelocity" job with "PositionControlled" = TRUE forces position-controlled operation.

The selected mode remains in effect after the job is completed.

A "MC_MotionInPosition" job forces position-controlled operation.

Influence of additional Motion Control instructions

Starting the following Motion Control instructions forces position-controlled operation of the axis:

- MC_Home with "Mode" = 3, 5
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveSuperimposed
- MC_MotionInPosition

- MC_GearIn
- MC_GearInPos (S7-1500T)
- MC_CamIn (S7-1500T)

The closed loop position control remains active after completing the corresponding jobs.

The Motion Control instruction "MC_Halt" is executed in position-controlled and non-position-controlled operation. The status of the closed loop position control is not changed by "MC_Halt".

A torque limiting activated with "MC_TorqueLimiting" is in effect even with non-position-controlled operation.

Synchronous operation with setpoint coupling

A following axis is set into position-controlled operation with the start of a synchronous operation job. If the leading axis is in non-position-controlled operation at the start of the synchronous operation, the synchronous operation job remains waiting. Synchronization is started only after position control has been activated and the start position of the synchronization has been reached.

Note

If the leading axis is set to the non-position-controlled mode during active synchronization, your setpoint is then set to zero. A setpoint step change is obtained as a result of coupling the setpoint of following axis. The setpoint step change is compensated according to the constant function. The only limiting factor is the maximum speed of the drive.

Synchronous operation with actual value coupling (S7-1500T)

A following axis is set into position-controlled operation with the start of a synchronous operation job. If the leading axis is in non-position-controlled operation at the start of the synchronous operation and the actual values are valid, synchronization is started.

If the leading axis is set to the non-position-controlled mode during active synchronization, the synchronization remains active.

Starting Motion Control jobs (S7-1500)

Motion Control jobs are started by setting the "Execute" or "Enable" parameter of the Motion Control instruction. The call of the Motion Control instructions for a technology object should occur in an execution level.

When executing Motion Control jobs, you should also take note of the status of the technology object.

Starting Motion Control jobs should be performed in the following steps:

1. Query the status of the technology object.
2. Initiate new job for the technology object.
3. Check job status.

These steps are explained using the example of a job for absolute positioning.

1. Query the status of the technology object

Make sure that the technology object is in the appropriate status to perform the desired job:

- **Has the technology object been released?**
  To execute motion jobs, the technology object must be enabled.
  Enabling is performed using the Motion Control instruction "MC_Power".
  The "MC_Power.Status" parameter (<TO>.StatusWord.X0 (Enable)) must show the value "TRUE".
- **Is a technology alarm pending?**
  To perform motion commands, no technology alarms or alarm responses may be pending. The "<TO>.ErrorDetail.Number" and "<TO>.ErrorDetail.Reaction" tags of the technology object must show the value zero. After resolving the error, acknowledge any pending alarms using the Motion Control instruction "MC_Reset".
  You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.
- **Has the technology object been homed?**
  In order to perform a job for absolute positioning, the positioning axis/synchronous axis technology object must be homed. The referencing occurs via the Motion Control instruction "MC_Home". The "<TO>.StatusWord.X5 (HomingDone)" tag of the technology object must show the value "TRUE".

2. Initiate new command for the technology object

In the "Position" parameter of the "MC_MoveAbsolute" instruction, specify the position to which the axis should be moved. Start the job with a positive edge at the "Execute" parameter.

3. Check command status

Parameter "Done" of the Motion Control instruction indicates successful completion of a job (target reached, in this case).

If an error is detected, the "Error" parameter of the Motion Control instruction is set to "TRUE", and the job is rejected.

You can program an error handling routine for the Motion Control job. For this purpose, evaluate the error indicated in the "Error" parameter. The cause of the error is indicated in the ErrorID parameter. After resolving the cause of the error, restart the job.

If "Error" = TRUE and "ErrorID" = 16#8001 is indicated during job execution, a technology alarm has occurred.

You can find a list of the ErrorIDs in the "Error detection (Page 2133)" appendix.

Additional information

An option for evaluating the individual status bits, error bits, and warning bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tracking active jobs (S7-1500)

The current status of the job processing is made available via the output parameters of the Motion Control instruction. These parameters are updated with each call of the Motion Control instruction.

When tracking jobs, a distinction is made between three groups:

- Motion Control instructions with "Done" parameter (Page 1935)
- Motion Control instructions without "Done" parameter (Page 1938)
- Motion Control instruction "MC_MoveJog" (Page 1941)

Motion Control instructions with "Done" parameter (S7-1500)

Jobs of Motion Control instructions with the "Done" parameter are started with a positive edge at the "Execute" parameter. If the job was completed without errors or interruption by another job (e.g. "MC_MoveAbsolute": Target position reached), the "Done" parameter shows the value "TRUE".

The following Motion Control instructions have a "Done" parameter:

- MC_Home
- MC_MoveRelative
- MC_MoveAbsolute
- MC_MoveSuperimposed
- MC_Halt
- MC_Reset
- MC_MeasuringInput
- MC_AbortMeasuringInput

The behavior of the parameters is shown below by way of example for various situations.

Complete execution of the job

If the Motion Control job has been completely executed all the way to the end, this is indicated with parameter "Done" = "TRUE". The signal state of the "Execute" parameter influences the display duration for the "Done" parameter:

You set "Execute" **after completion** of the job to "FALSE".	You set "Execute" **during processing** of the job to "FALSE".
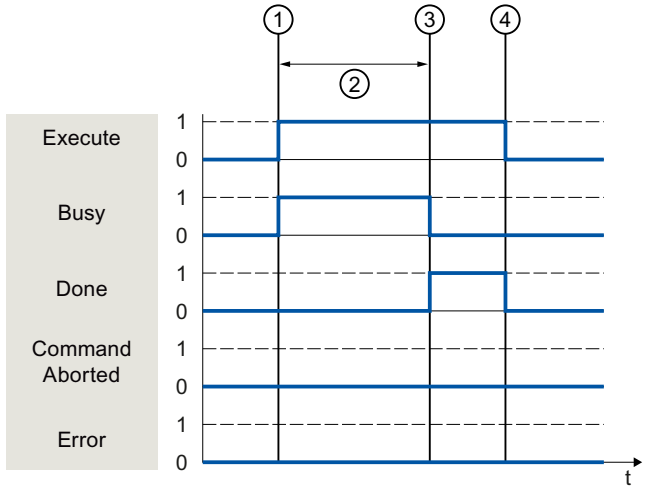	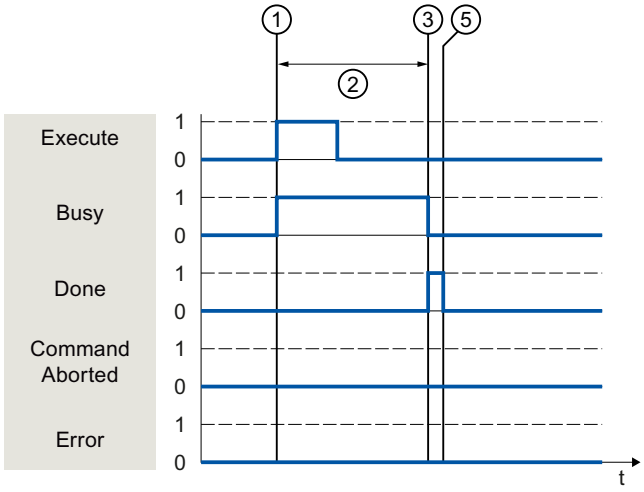

①	The job is started with a positive edge at the "Execute" parameter. Depending on the programming, "Execute" can be reset to the value "FALSE" during the job or the value "TRUE" can be retained until after completion of the job.
②	While the job is being executed, the "Busy" parameter shows the value "TRUE".
③	At the completion of the job (for example, with Motion Control instruction "MC_MoveAbsolute": Target position reached), the "Busy" parameter changes to "FALSE" and the "Done" parameter to "TRUE".
④	As long as the "Execute" parameter retains the value "TRUE" after completion of the job, the "Done" parameter also retains the value "TRUE".
⑤	If the "Execute" parameter was already set to "FALSE" before completion of the job, the "Done" parameter shows the value "TRUE" for only one execution cycle.

Job abort

If the Motion Control job is canceled during processing by another job, this is indicated in the "CommandAborted" parameter with the value "TRUE". The signal state of the "Execute" parameter influences the display duration for the "CommandAborted" parameter:

You set "Execute" **after** the job is canceled to "FALSE".	You set "Execute" **before** the job is aborted to "FALSE".
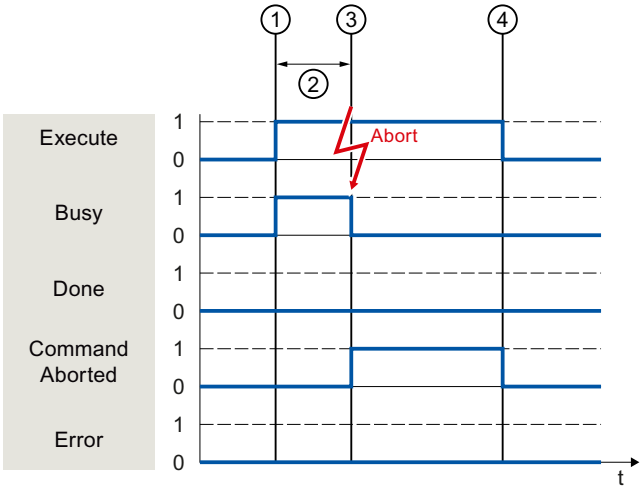	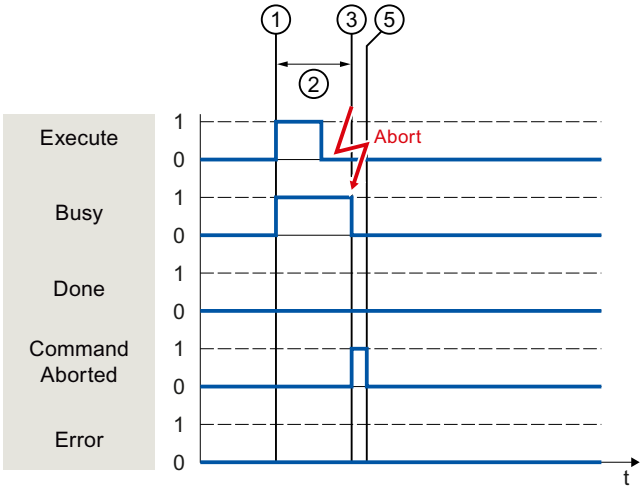

①	The job is started with a positive edge at the "Execute" parameter. Depending on the programming, "Execute" can be reset to the value "FALSE" during the job or the value "TRUE" can be retained until after completion of the job.
②	While the job is being executed, the parameter "Busy" shows the value "TRUE".
③	During job execution, the job is aborted by another Motion Control job. When the job is aborted, the "Busy" parameter changes to "FALSE" and "CommandAborted" changes to "TRUE".
④	As long as the "Execute" parameter retains the value "TRUE" after completion of the job, the "CommandAborted" parameter also retains the value "TRUE".
⑤	If the "Execute" parameter was already set to "FALSE" before the job is aborted, the "CommandAborted" parameter shows the value "TRUE" for only one execution cycle.

Error during job execution

If an error occurs during execution of the Motion Control job, this is indicated with parameter "Error" = "TRUE". The signal state of the "Execute" parameter influences the display duration for the "Error" parameter:

You set "Execute" **after** the occurrence of the error to "FALSE".	You set "Execute" **before** the occurrence of the error to "FALSE".
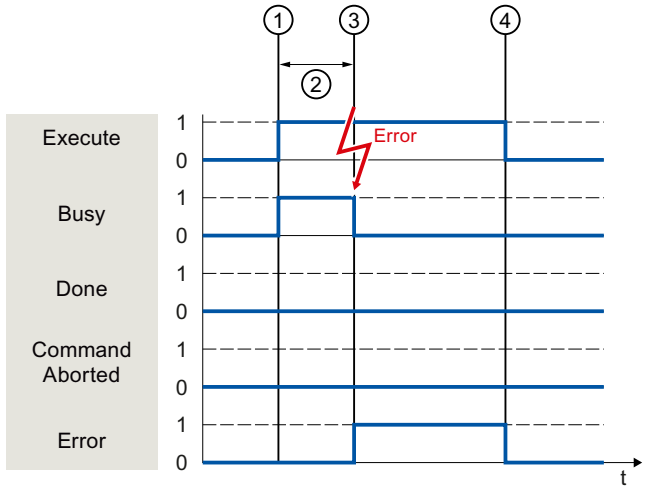	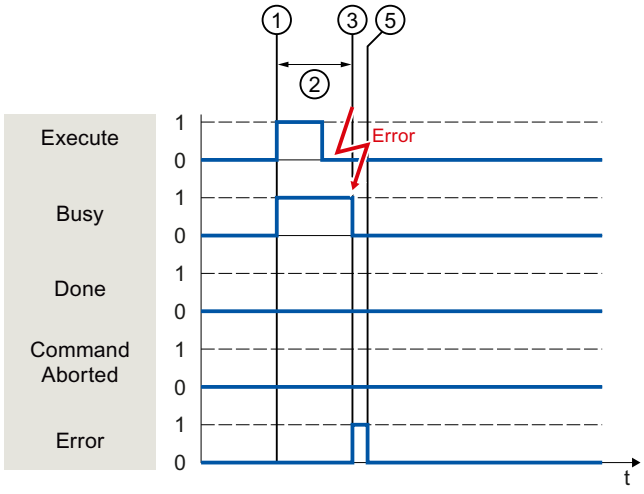

①	The job is started with a positive edge at the "Execute" parameter. Depending on the programming, "Execute" can be reset to the value "FALSE" during the job or the value "TRUE" can be retained until after completion of the job
②	While the job is being executed, the "Busy" parameter shows the value "TRUE".
③	An error occurs during the execution of the job. When the error occurs, the "Busy" parameter changes to "FALSE" and the "Error" parameter to "TRUE".
④	As long as the "Execute" parameter retains the value "TRUE" after the occurrence of the error, the "Error" parameter also retains the value "TRUE".
⑤	If the "Execute" parameter was already set to "FALSE" before the occurrence of the error, the "Error" parameter shows the value "TRUE" for only one execution cycle.

Motion Control instructions without "Done" parameter (S7-1500)

Motion Control instructions without the "Done" parameter use a special parameter to indicate that the command objective (e.g. "InVelocity", "InGear") has been achieved. The target state or motion is stopped until the job is aborted or an error occurs.

The following Motion Control instructions have a special parameter for indicating the job status:

- MC_Power ("Status" parameter)
- MC_MoveVelocity ("InVeloctity" parameter)
- MC_GearIn ("InGear" parameter)
- MC_MoveJog ("InVelocity" parameter)
  The special behavior of "MC_MoveJog" is described in the section Motion Control instruction MC_MoveJog (Page 1941).
- MC_TorqueLimiting ("InClamping" and "InLimitation") parameters

The following Motion Control instructions have no special parameter for indicating the job status: Feedback is provided via the following tags:

- MC_MeasuringInputCyclic
  The execution of a measuring job is indicated with parameter "Busy" = "TRUE". Completed measuring events are indicated in the corresponding event counters "<TO>.MeasuredValues.MeasuredValue1Counter" and "<TO>.MeasuredValues.MeasuredValue2Counter" of the technology data block.
- MC_OutputCam
  The execution of a job is indicated with parameter "Busy" = "TRUE". The CamOutput tag in the associated technology data block indicates the switching state of the output cam.
- MC_CamTrack
  The execution of a job is indicated with parameter "Busy" = "TRUE". The TrackOutput tag in the associated technology data block indicates the switching state of the output cam.

The behavior of the parameter is shown for various situations using the Motion Control instruction "MC_MoveVelocity" as an example:

Example "MC_MoveVelocity"

An "MC_MoveVelocity" job is started with a positive edge at the "Execute" parameter. The job objective is fulfilled when the assigned velocity is reached and the axis travels at constant velocity. When the assigned velocity is reached and maintained, this is indicated in the "InVelocity" parameter with the value "TRUE".

The motion of the axis can, for example, be stopped with an "MC_Halt" job.

The assigned velocity is reached and maintained

Reaching of the assigned velocity is indicated with parameter "InVelocity" = "TRUE". The "Execute" parameter has no effect on the display duration for the "InVelocity" parameter.

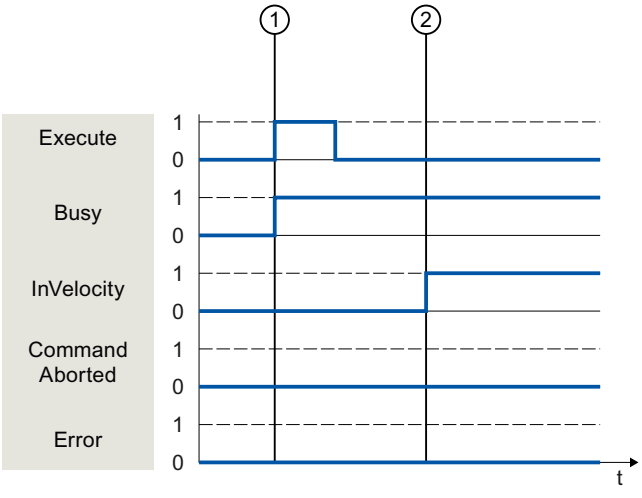

①	The job is started with a positive edge at the "Execute" parameter. Depending on the programming, "Execute" can be reset to the "FALSE" value before or after the parameterized velocity has been reached. While the job is being executed, the parameter "Busy" shows the value "TRUE".
②	When the assigned velocity is reached, the "InVelocity" parameter changes to "TRUE". The "Busy" and "InVelocity" parameters retain the value "TRUE" until another Motion Control job overrides the "MC_MoveVelocity" job.

The job is aborted before the assigned velocity is reached

If the Motion Control job is canceled by another job before the assigned velocity is reached, this is indicated with parameter "CommandAborted" = "TRUE". The signal state of the "Execute" parameter influences the display duration for the "CommandAborted" parameter.

You set "Execute" **after** the job is canceled to "FALSE".	You set "Execute" **before** the job is aborted to "FALSE".
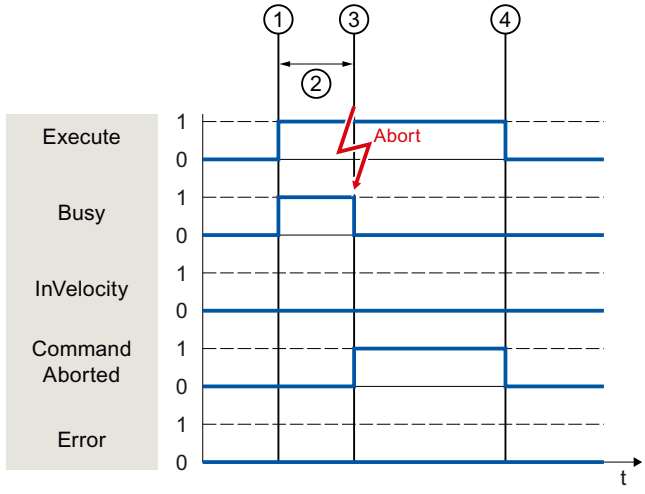	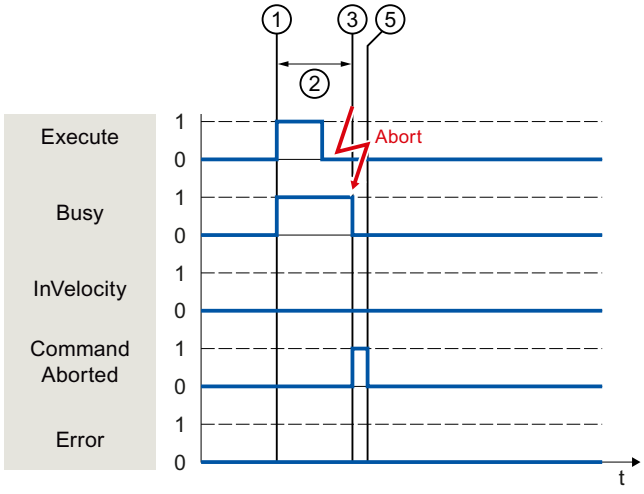

①	The job is started with a positive edge at the "Execute" parameter. Depending on the programming, "Execute" can be reset to the value "FALSE" during the job or the value "TRUE" can be retained until after the job is aborted.
②	While the job is being executed, the "Busy" parameter shows the value "TRUE".
③	During job execution, the job is aborted by another Motion Control job. When the job is aborted, the "Busy" parameter changes to "FALSE" and "CommandAborted" changes to "TRUE".
④	As long as the "Execute" parameter retains the value "TRUE" after completion of the job, the "CommandAborted" parameter also retains the value "TRUE".
⑤	If the "Execute" parameter was already set to "FALSE" before the job is aborted, the "CommandAborted" parameter shows the value "TRUE" for only one execution cycle.

An error has occurred prior to reaching the assigned velocity

If an error occurs during execution of the Motion Control job before the assigned velocity has been reached, this is indicated with parameter "Error" = "TRUE". The signal state of the "Execute" parameter influences the display duration for the "Error" parameter.

You set "Execute" **after** the occurrence of the error to "FALSE".	You set "Execute" **before** the occurrence of the error to "FALSE".
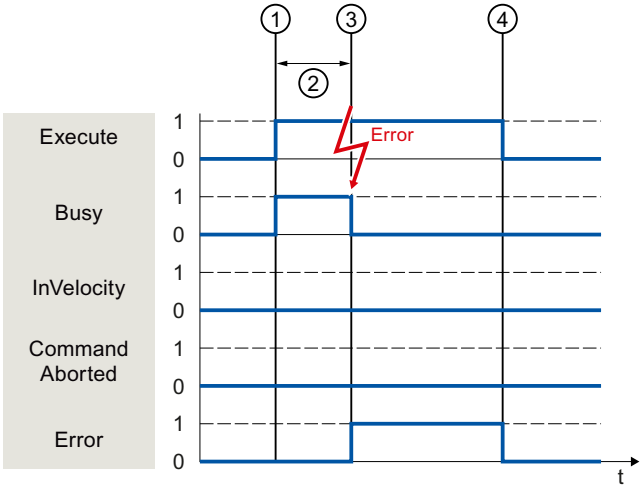	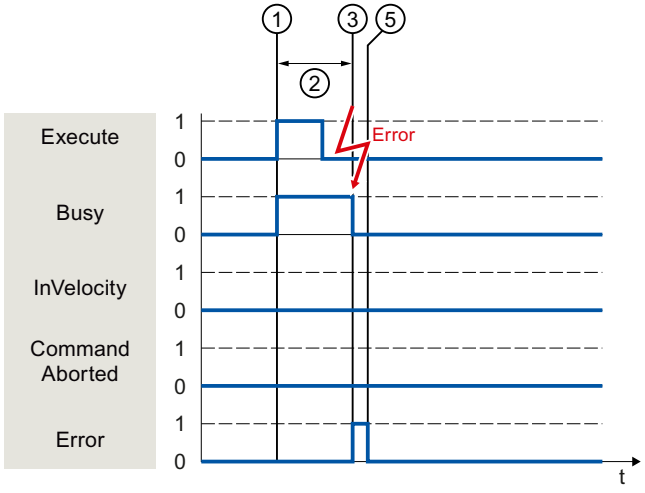

①	The job is started with a positive edge at the "Execute" parameter. Depending on the programming, "Execute" can be reset to the value "FALSE" during the job or the value "TRUE" can be retained until after the error has occurred.
②	While the job is being executed, the "Busy" parameter shows the value "TRUE".
③	An error occurs during the execution of the job. When the error occurs, the "Busy" parameter changes to "FALSE", and the "Error" parameter to "TRUE".
④	As long as the "Execute" parameter retains the value "TRUE" after completion of the job, the "Error" parameter also retains the value "TRUE".
⑤	If the "Execute" parameter was already set to "FALSE" before the job is aborted, the "Error" parameter shows the value "TRUE" for only one execution cycle.

Motion Control instruction "MC_MoveJog" (S7-1500)

An "MC_MoveJog" job is started by setting the "JogForward" or "JogBackward" parameter. The job objective is fulfilled when the assigned velocity is reached and the axis travels at constant velocity. When the assigned velocity is reached and maintained, this is indicated in the "InVelocity" parameter with the value "TRUE".

The job is complete when the "JogForward" or "JogBackward" parameter has been set to the value "FALSE" and the axis has come to a standstill.

The behavior of the parameters in various situations is shown below by way of example.

The assigned velocity is reached and maintained

If the Motion Control job has been performed up to the point of reaching the assigned velocity, then this is indicated in the "InVelocity" parameter with the value "TRUE".

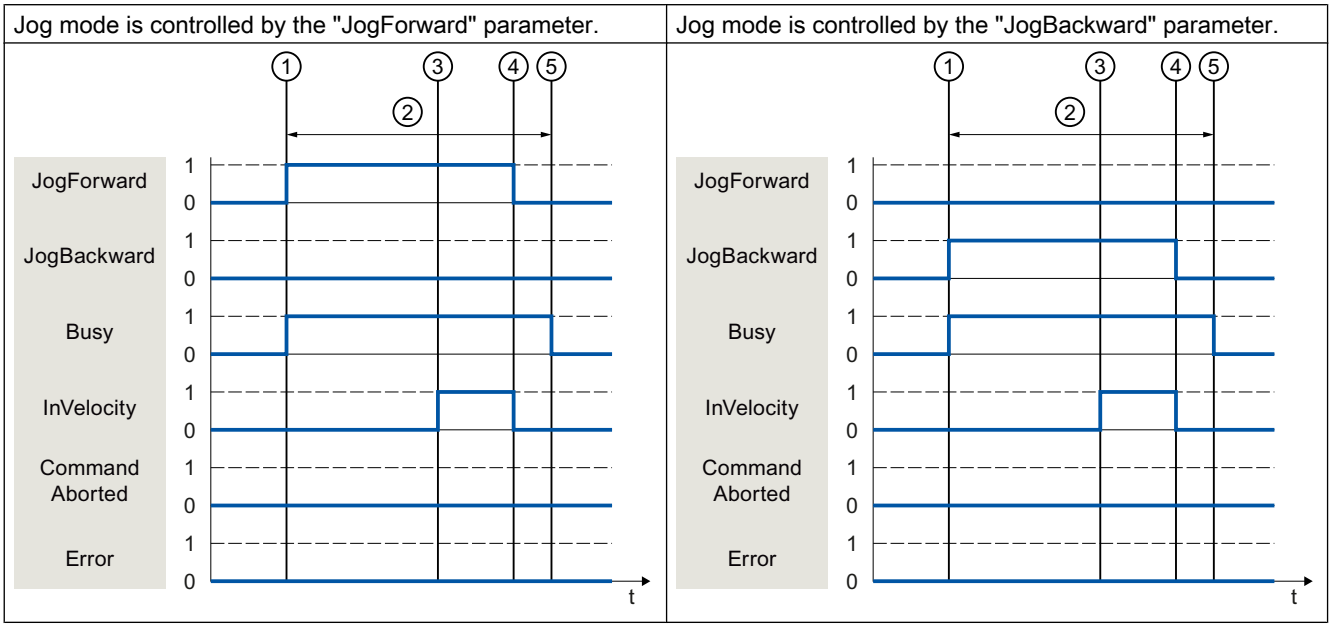

①	The job is started by setting the "JogForward" or "JogBackward" parameter.
②	While the job is being executed, the "Busy" parameter shows the value "TRUE".
③	When the assigned velocity is reached, the "InVelocity" parameter changes to "TRUE".
④	When the "JogForward" or "JogBackward" parameter is reset, the motion of the axis ends. The axis decelerates. The "InVelocity" parameter changes to "FALSE".
⑤	If the axis has come to a standstill, then the Motion Control job is complete and the "Busy" parameter changes to "FALSE".

The job is aborted during execution

If the Motion Control job is canceled during processing by another job, this is indicated in the "CommandAborted" parameter with the value "TRUE". The behavior of the "CommandAborted" parameter is independent of reaching the assigned velocity.

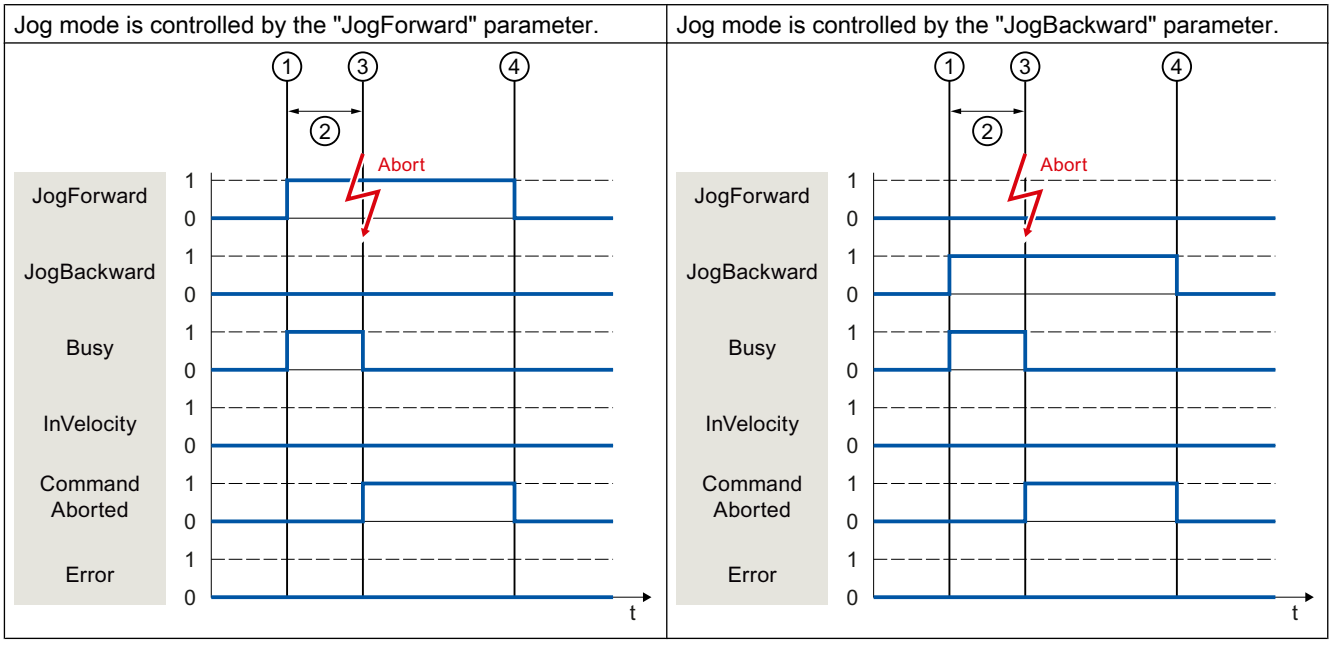

①	The job is started by setting the "JogForward" or "JogBackward" parameter.
②	While the job is processing, the "Busy" parameter shows the value "TRUE".
③	During job execution, the job is aborted by another Motion Control job. When the job is aborted, the "Busy" parameter changes to "FALSE" and "CommandAborted" changes to "TRUE".
④	When the "JogForward" or "JogBackward" parameter is reset, the "CommandAborted" parameter likewise changes to "FALSE".

An error occurs during the execution of the job

If an error occurs during execution of the Motion Control job, this is indicated in the "Error" parameter with the value "TRUE". The behavior of the "Error" parameter is independent of reaching the assigned velocity.

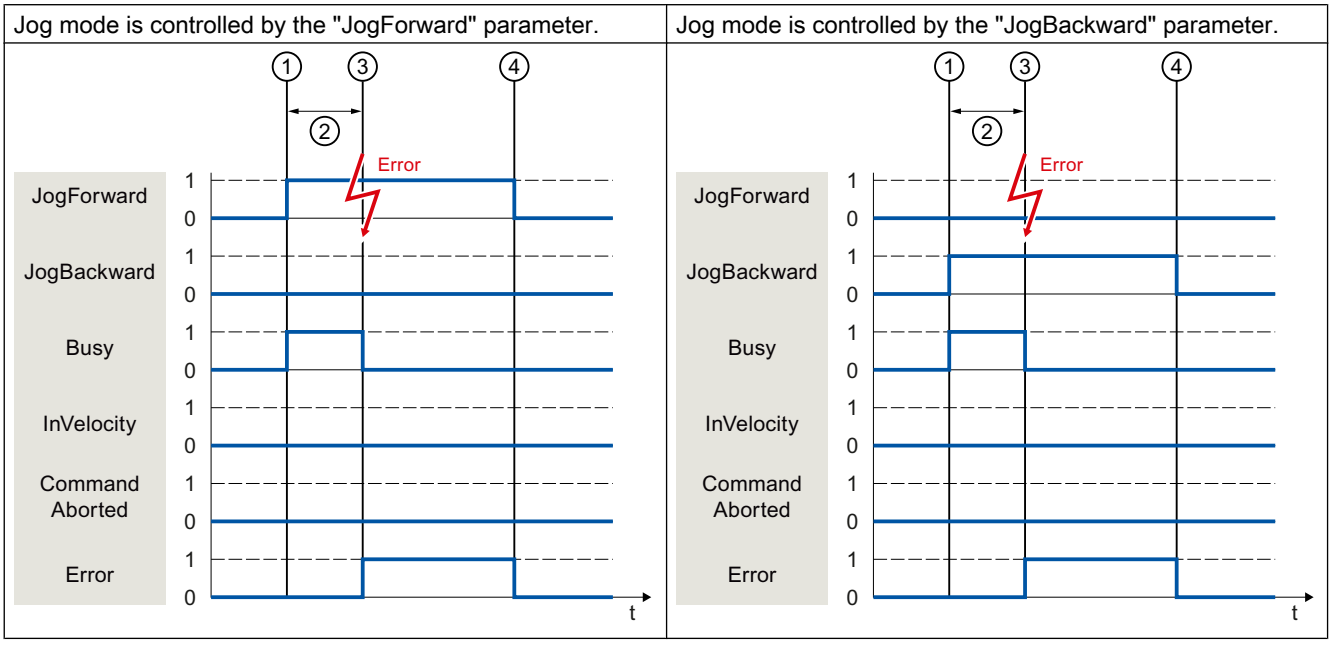

①	The job is started by setting the "JogForward" or "JogBackward" parameter.
②	While the job is being executed, the "Busy" parameter shows the value "TRUE".
③	An error occurs during the execution of the job. When the error occurs, the "Busy" parameter changes to "FALSE", and "Error" changes to "TRUE".
④	When the "JogForward" or "JogBackward" is reset to the value "FALSE", the "Error" parameter likewise changes to "FALSE".

Additional Motion Control instructions of the technology CPU

The additional Motion Control instructions of the S7-1500T CPU have the same parameter behavior as those of the S7-1500 CPU.

Motion Control instructions with "Done" parameter

The following Motion Control instructions have a "Done" parameter:

- MC_PhasingAbsolute
- MC_PhasingRelative
- MC_InterpolateCam
- MC_GetCamFollowingValue
- MC_GetCamLeadingValue

Motion Control instructions without "Done" parameter

The following Motion Control instructions have a special parameter for indicating the job status:

- MC_GearInPos ("InSync" parameter)
- MC_CamIn ("InSync" parameter)
- MC_SynchronizedMotionSimulation ("InSimulation" parameter)
- MC_MotionInVeloctiy (Parameter "Busy")
- MC_MotionInPosition (Parameter "Busy")

Motion Control instructions

You can find additional information on the Motion Control instructions of the technology object kinematics in the "S7-1500T Kinematics Functions" function manual.

See also

S7-1500T kinematics functions (https://support.industry.siemens.com/cs/ww/en/view/109749264)

Ending Motion Control jobs (S7-1500)

When a job is ended, a distinction is made between error-free completion of the job and a motion abort.

Completion of job

The completion of a Motion Control job is indicated as described in the Tracking running jobs (Page 1934) section.

Job termination

The termination and substitution behavior are described in the section AUTOHOTSPOT. Special pending jobs can be cancelled with "MC_Power".

Motion abort

If a motion must be aborted, you can perform the following measures:

- Execute "MC_Halt"
  To abort a motion and stop the axis, you can use the "MC_Halt" instruction.
- Deactivate "MC_Power"
  In an emergency, you can stop the axis via an emergency stop ramp. To do so, set the "Enable" parameter of the "MC_Power" instruction to "FALSE". The axis is decelerated according to the selected "StopMode" and all jobs for the technology object are aborted.

Measuring job abort

With the Motion Control instruction "MC_AbortMeasuringInput", an active one-time or cyclic measuring job is aborted.

Cancellation of an active output cam/cam track

- "MC_OutputCam"
  An active output cam is disabled when the "Enable" parameter of Motion Control instruction "MC_OutputCam" is set to "FALSE".
- "MC_CamTrack"
  An active cam track is disabled when the "Enable" parameter of Motion Control instruction "MC_CamTrack" is set to "FALSE".

Restart of technology objects (S7-1500)

After the CPU is switched on, or after technology objects are downloaded into the CPU, the system automatically initializes the technology objects with the start values from the technology data block. If restart-relevant changes are detected during a reload into the CPU, a restart of the technology object is automatically performed.

If restart-relevant data have been changed in RUN mode by the user program, then the technology object must be reinitialized by the user in order for the changes to be used. At a RUN → STOP transition, the CPU automatically performs a restart of technology objects with restart-relevant changes.

If changes in the technology data block should also be retained after the restart of the technology object, then you must write the changes to the start value in load memory using the extended instruction "WRIT_DBL".

Restart required

If a restart of the technology object is required, this is indicated at "Technology object > Diagnostics > Status and error bits > Axis status or Encoder status > Online start value changed", as well as in the tag "<TO>.StatusWord.X3 (OnlineStartValuesChanged)" of the technology object.

Restarting a technology object

A restart of the technology object is triggered by the user by means of the "MC_Reset" Motion Control instruction, with parameter "Restart" = TRUE.

During a restart, all configuration data of the technology object are loaded from load memory into work memory. In the process, the actual values in the technology data block are overwritten.

Note the following during a restart of the technology object:

- A restart resets the "Homed" status of a technology object with incremental actual values (<TO>.StatusWord.X5 (HomingDone)).
- While a restart is being performed, the technology object cannot perform any jobs. An active restart will be indicated under "Technology object > Diagnostics > Status and error bits > Axis status or Encoder status > Restart active", and in the "<TO>.StatusWord.X2 (RestartActive)" tag of the technology object.
- Motion Control jobs are rejected during a restart with the "Error" = TRUE and "ErrorID" = 16#800D parameters (job not executable, because a restart is active).
- While a restart is being executed, you cannot access the technology data block.

See also

Change restart-relevant data (Page 1923)

6.1.2.6 Downloading to CPU (S7-1500)

When downloading to the CPU S7-1500, it is always verified that the project files are consistent online and offline after the download.

The data of the technology objects are saved in technology data blocks. The conditions for downloading blocks thus apply when loading new or modified technology objects.

Load in RUN mode

When loading in the CPU's RUN mode, it is checked whether a load without restart of the technology objects is possible.

If restart-relevant configuration values were changed, then a restart of the technology object is automatically performed after the load into the CPU.

Loading a technology object is only possible if the technology object is disabled.

You **cannot** download the following changes to the CPU in RUN mode:

- Changes to the MC-Servo clock speeds
- Changes to the hardware interface of the technology object in "Technology object > Configuration > Hardware interface"

6.1.2.7 Commissioning (S7-1500)

The following guidelines describe the steps that you should note when commissioning the Motion Control-specific components of your equipment.

The commissioning of other components of your automation system depends on the particular equipment configuration. Commissioning (not Motion Control) is described in the Automation System S7-1500 (http://support.automation.siemens.com/WW/view/en/59191792) system manual.

Commissioning guidelines (S7-1500)

These guidelines serve as recommendations for commissioning equipment with Motion Control. The procedure is described using the example of a positioning axis technology object.

Requirement

- The configuration of the following components is complete:
  - CPU
  - BUS communication
  - Drives
  - Technology objects
- The user program has been created.
- The wiring of the CPU and of the associated I/O is complete.
- The commissioning and optimization of the drive is complete.

Procedure

Proceed as follows to commission the Motion Control-specific components of your equipment:

Step	Action to be performed	Supported by TIA Portal
Turn on CPU	Turn on the power supply and the CPU.	-
"Disable" position controller	Set the gain (Kv factor) of the position control loop to zero. (This setting avoids unwanted drive movements that may be caused by incorrect parameterization of the position control loop.)	"Technology object > Configuration > Extended parameters > Control loop"
Activate precontrol	Set the precontrol to 100 %.	"Technology object > Configuration > Extended parameters > Control loop"
Load project into the CPU	Bring the CPU to the STOP mode. Download your project to the CPU (load hardware and software).	• "Toolbar > Stop CPU" • "Toolbar > Download to device"
Create online connection to the CPU	Select the "Receive messages" check box under "Online & Diagnostics > Online Access". Configure the interface of the TIA Portal and create an online connection with the CPU.	• Device configuration • "Online & Diagnostics > Online Access"
Disable Motion Control specific user program	In order to avoid conflicts with the axis control panel, lock the enabling of technology objects in your user program (MC_Power.Enable = FALSE).	• PLC programming • Motion Control instructions
Evaluating pending messages	Evaluate the message display in the inspector window. Resolve the causes of pending technology alarms. Acknowledge the technology alarms (Page 1962).	"Inspector window > Diagnostics > Message display"
Check hardware limit switches	Click the hardware limit switches. Check for correct message display (technology alarm 531). Acknowledge the technology alarm.	"Inspector window > Diagnostics > Message display"

Step	Action to be performed	Supported by TIA Portal
Check the connection and configuration of the drive (setpoint)	Bring the CPU into the RUN mode. Open the Axis control panel (Page 1950) and take over control. Perform the following steps: • Enable the technology object. ⇒ The drive must turn itself on, and where applicable release the brake. The position is held. • Move the axis in jog mode at low velocity in the positive direction. ⇒ The drive must move. The actual position value must increase (positive direction). • Disable the technology object. ⇒ The drive must turn itself off, and where applicable apply the brake.	"Technology object > Commissioning > Axis control panel"
Check the connection and configuration of the encoder (actual value)	Check the scaling of the actual values (rotation direction, distance evaluation, and resolution of the encoder) ⇒ The change in the actual mechanical position must match the change in the actual values.	• "Technology object > Diagnostics > PROFIdrive telegram" • "Technology object > Commissioning > Axis control panel"
Checking the reference speed	Traverse the axis in jog mode at low velocity in the positive direction. ⇒ The displayed current velocity must match the velocity setpoint. If the displayed current velocity deviates significantly from the velocity setpoint, adjust the reference speed.	• "Technology object > Hardware interface > Data exchange" • "Technology object > Commissioning > Axis control panel"
Optimize position controller	Use the Optimization (Page 1956) commissioning function to optimize the gain (Kv) of the position control loop. For this purpose, adapt following error limits as needed.	"Technology object > Commissioning > Optimization"
Transfer the gain Kv to the project.	Enter the gain Kv that you determined by means of the optimization function in your configuration data. Load your project into the CPU.	"Technology object > Configuration > Extended parameters > Control loop"
Enable Motion Control specific user program	Unlock the enabling technology objects lock in your user program (MC_Power.Enable = TRUE).	• PLC programming • Motion Control instructions
Check the functioning of the user program	Check the programmed functions of your user program.	• Watch and force tables • Online and diagnostic functions
End of commissioning for a positioning axis technology object	To commission additional technology objects, perform the corresponding steps again.	See above.

Axis control panel (S7-1500)

Function and structure of the axis control panel (S7-1500)

You traverse individual axes with the axis control panel. A user program is not necessary for the operation of the axis control panel. With the axis control panel, you assume master control for a technology object and control the motions of the axis.

WARNING

Uncontrolled axis motions

During operation with the axis control panel, the axis can execute uncontrolled motions (e.g. due to erroneous configuration of the drive or the technology object). Furthermore, any synchronized following axis is moved as well when moving a leading axis with the axis control panel.

Therefore, perform the following protective measures before operation with the axis control panel:

- Ensure that the EMERGENCY OFF switch is within the reach of the operator.
- Enable the hardware limit switches.
- Enable the software limit switches.
- Ensure that following error monitoring is enabled.
- Make sure that no following axis is coupled to the axis to be moved.

The axis control panel of the speed axis, positioning axis and synchronous axis can be found in the project tree under "Technology object > Commissioning".

The axis control panel is divided into the following areas:

- Master control
- Axis
- Operating mode
- Modify
- Axis status
- Actual values

Elements of the axis control panel

The following table lists the elements of the axis control panel:

Area	Element	Description
Master control		In the "Master control" area, you assume master control of the technology object or return it to your user program.
	"Activate" button	With the "Activate" button, you set up an online connection to the CPU and take over master control for the selected technology object. • To take over master control, the technology object must be disabled in the user program. • Any synchronized following axis is moved as well when moving a leading axis with the axis control panel. • If the online connection to the CPU is lost during operation with the axis control panel, then after the sign of life monitoring has elapsed, the axis will be stopped with maximum deceleration. In this case, an error message is displayed ("ErrorID" = 16#8013) and the master control is passed back to the user program. • If the axis control panel is covered by a dialog, such as "Save as", during its operation with the axis control panel, the axis is stopped with maximum deceleration and the master control is returned to the user program. If the "Stop" button is covered during operation with the axis control panel, for example by scrolling or by another window, the master control is retained but the axis is stopped with maximum deceleration. If you change to another window within the TIA Portal, to the project tree for example, during operation with the axis control panel, the master control and motion of the axis is maintained, provided that the axis control panel is embedded in the TIA Portal. If the axis control panel is replaced by the TIA Portal and you change to another window within the TIA Portal, to the project tree for example, the master control is retained but the axis is stopped with maximum deceleration. If you change to another window outside the TIA Portal during operation with the axis control panel, the master control is retained but the axis is stopped with maximum deceleration. • When you click the "Activate" button, a warning message is displayed. In the warning, you can adapt the sign-of-life monitoring (100 to 60000 ms). If the master control of the axis control panel is lost repeatedly without a direct error message, the online connection to the CPU may be impaired because the communication load is too high. In this case, the message "Error during commissioning" is entered in the message display log. Sign-of-life failure between controller and TIA Portal" is displayed. To eliminate this error, adapt the sign-of-life monitoring in the warning. • Until master control is returned, the user program has no influence on the functions of the technology object. Motion Control jobs from the user program to the technology object are rejected with error ("ErrorID" = 16#8012: Axis control panel enabled). • When master control is taken over, the configuration of the technology object is adopted. Changes to the configuration of the technology object do not take effect until leading control has been returned. Therefore, make any necessary changes before master control is assumed again. • If master control has been taken over for the technology object, the axis control panel is blocked for access by another instance of the TIA Portal (Team Engineering as of CPU V1.5).
	"Deactivate" button	With the "Deactivate" button, you return master control to your user program.

Area	Element	Description
Axis		In the "Axis" area, you can enable or disable the technology object.
	"Enable" button	With the "Enable" button, you enable the selected technology object
	"Disable" button	With the "Disable" button, you disable the selected technology object
Operating mode		Select the required mode in the "Operating mode" drop-down list.
Modify		The "Control" area displays the parameters for traversing with the axis control panel according to the selected operating mode.
	Position	Position to which the axis is homed. ("Homing" and "Set home position" modes only)
	Distance	Distance the axis is traversed. ("Relative positioning" mode only)
	Target position	Position to which the axis is traversed. ("Absolute positioning" mode only)
	Velocity/ Velocity setpoint	Velocity or speed at which the axis is traversed. Default: 10% of default value (Velocity/speed setpoint, "Jog" and "Positioning" modes only)
	Acceleration	Acceleration with which the axis is traversed. Default: 10% of default value
	Deceleration	Deceleration with which the axis is traversed. Default: 10% of default value
	Jerk	Jerk with which the axis is traversed. Default: 100% of the default value
	"Start" button	With the "Start" button, you start a job according to the selected operating mode.
	"Forward" button	With the "Forward" button, you start a motion in the positive direction according to the selected operating mode.
	"Backward" but-ton	With the "Backward" button, you start a motion in the negative direction according to the selected operating mode.
	"Stop" button	With the "Stop" button, you cancel a job or stop the axis. If the "Stop" button is covered during operation with the axis control panel, for example by scrolling or by another window, the master control is retained but the axis is stopped with maximum deceleration.
Axis status		The "Axis status" area displays the status of the axis and the status of the drive.
	Drive ready	Drive is ready to execute setpoints.
	Error	An error occurred at the technology object.
	Enabled	The technology object has been enabled. The axis can be moved with motion jobs.
	Homed	The technology object is homed.
	More	The "More" link takes you to the window "Technology object > Diagnostics > Status and error bits".
	Active errors	The error that occurred most recently is displayed in the "Active errors" text field.
	"Confirm" button	With the "Confirm" button, you acknowledge pending errors.
	Alarm display	You can access the alarm display in the Inspector window by clicking on the "Alarm display" link.
Actual values		The "Current values" area shows the actual values of the axis.
	Position	Actual position of the axis
	Velocity	Actual velocity of the axis

Note

No transfer of the parameters

The configured parameter values are discarded when master control is returned. Transfer the values as needed into your configuration.

If you have changed configuration values during operation with the axis control panel, these changes have no effect on the operation of the axis control panel.

Operating mode

The following table shows the operating modes of the axis control panel:

Operating mode	Description
Homing	This function corresponds to active homing. The parameters for homing (Page 1670) must be configured. Homing is not possible with an absolute encoder. The technology object is not referenced when this mode is used with an absolute encoder.
Set home position	This function corresponds to direct homing (absolute). With the "Start" button, you set the actual position to the value specified in "Position" and the "Homed" status is set.
Jog	Motion commands occur by means of jogging. With the "Forward" or "Backward" button you start motion in the positive or negative direction. The motion runs for as long as you hold down the left mouse button.
Velocity specification/ speed setpoint	The axis is moved at the specified velocity or speed until you stop the movement. The motion commands are performed according to the setpoints assigned under "Controller".
Positioning relative	The positioning is executed as a controlled, relative traversing motion according to the defaults assigned under "Controller".
Positioning absolute	The positioning is executed as a controlled, absolute traversing motion according to the defaults assigned under "Controller". If you have enabled the "Modulo" setting of the technology object, the buttons "Forward" and "Backward" are shown in the "Controller" area. The axis is positioned within the modulo range. Position settings outside the modular range are recalculated to the modulo range. If you have not enabled the "Modulo" setting of the technology object, only the "Start" button is shown in the "Controller" area. You can directly approach the entered position.

Using the axis control panel (S7-1500)

You traverse individual axes with the axis control panel. You assume master control of a technology object and control the motions of the axis.

Requirement

- The project has been created and downloaded to the CPU.
- The CPU must be in the RUN mode.

- The technology object is disabled by your user program ("MC_Power.Enable" = FALSE).
- The axis control panel for the technology object is not used by another instance of the TIA Portal (Team Engineering as of CPU V1.5).

Procedure

Proceed as follows to enable control the axis using the axis control panel:

1. To assume master control of the technology object and to set up an online connection to the CPU, click "Activate" in the "Master control" area.
   A warning message is displayed.
2. If necessary, adapt the sign-of-life monitoring and click "OK".
3. To enable the technology object, click the "Enable" button in the "Axis" area.
4. In the drop-down list in the "Operation mode" area, select the desired function of the axis control panel.
5. In the "Control" area, specify the parameter values for your job.
6. Depending on the mode selected, click the "Start", "Forward" or "Backward" button to start the job.
7. Click the "Stop" button to stop the job.
8. Repeat steps 4 through 6 for additional jobs.
9. To disable the technology object, click the "Disable" button in the "Axis" area.
10. To return master control to your user program, click the "Deactivate" button in the "Master control" area.

Optimization (S7-1500)

Function and structure of the optimization (S7-1500)

The "Optimization" function supports you in determining the optimal precontrol and gain (Kv factor) for the closed loop position control of the axis. The axis velocity profile is recorded by means of the Trace function for this purpose for the duration of a configurable positioning movement. You can then evaluate the recording, and adjust the precontrol and gain accordingly.

WARNING

Uncontrolled axis motions

During operation with the optimization, the axis can execute uncontrolled motions (for example, due to erroneous configuration of the drive or the technology object). In addition, when a leading axis is moved, any synchronized following axis is also moved.

Therefore, perform the following protective measures before operation with the optimization:

- Ensure that the EMERGENCY OFF switch is within the reach of the operator.
- Enable the hardware limit switches.
- Enable the software limit switches.
- Ensure that following error monitoring is enabled.
- Make sure that no following axis is coupled to the axis to be moved.

The "Optimization" function for the positioning axis and synchronous axis technology objects can be found in the project tree under "Technology object > Commissioning".

The "Tuning" dialog is divided into the following areas:

- Master control
- Axis
- Measurement configuration
- Optimize position controller
- Run measurement
- Trace

The following table lists the elements of optimization:

Area	Element	Description
Master control		In the "Master control" area, you take over master control of the technology object, or return it to your user program.
	"Activate" button	With the "Activate" button, you set up an online connection to the CPU and take over master control for the selected technology object. • To take over master control, the technology object must be disabled in the user program. • Any synchronized following axis is moved as well when moving a leading axis with the axis control panel. • If the online connection to the CPU is lost during operation with the axis control panel, then after the sign of life monitoring has elapsed, the axis will be stopped with maximum deceleration. In this case, an error message is displayed ("ErrorID" = 16#8013) and the master control is passed back to the user program. • When you click the "Activate" button, a warning message is displayed. You can adapt the sign-of-life monitoring here (100 to 60000 ms). If the master control of the axis control panel is lost repeatedly without a direct error message, the online connection to the CPU may be impaired because the communication load is too high. In this case, the message "Error during commissioning" is entered in the message display log. Sign-of-life failure between controller and TIA Portal" is displayed. To eliminate this error, adapt the sign-of-life monitoring in the warning. • Until master control is returned, the user program has no influence on the functions of the technology object. Motion Control jobs from the user program to the technology object are rejected with error ("ErrorID" = 16#8012: Axis control panel enabled). • When master control is taken over, the configuration of the technology object is adopted. Changes to the configuration of the technology object do not take effect until master control has been returned. For this reason, make any necessary changes before master control is taken over. • If master control has been taken over for the technology object, the axis control panel is blocked for access by another installation of the TIA Portal (Team Engineering as of CPU V1.5).
	"Deactivate" button	With the "Deactivate" button, you return master control to your user program.
Axis		In the "Axis" area, you can enable or disable the technology object.
	"Enable" button	With the "Enable" button, you enable the selected technology object
	"Disable" button	With the "Disable" button, you disable the selected technology object

Area	Element	Description
Measurement configuration		In the "Measurement configuration" area, you configure the settings for a test step.
	Distance	Travel distance for a test step
	Measurement duration	Time for a test step
	"Customize dynamics" check box	The "Dynamics adapt" check box allows you to adapt the acceleration, deceleration and maximum velocity. As long as the "Commissioning" working area is open, the previously set values are displayed when the check box is selected again.
	Acceleration	Presetting of the acceleration for a test step
	Deceleration	Default setting for delaying a test step
	Maximum velocity	Presetting of the maximum velocity for a test step
Optimize position controller		In the "Optimize position controller" area, you make the settings for optimization of the controller characteristics. You open a list of values using the symbol next to a field. The list of values contains the following values of the respective parameter: • Online actual value • Online start value • Project start value Enter the new value in the text field for the actual value. The new value is applied by clicking the "Forward" or "Backward" button in the "Run measurement" area.
	Precontrol	Current percentage velocity precontrol of the position controller
	Speed control loop substitute time	Current speed control loop substitute time With velocity precontrol, a simplified speed control loop model can be generated using the speed control loop substitute time. This prevents the velocity variable being overridden by the position controller during the acceleration and deceleration phases. To accomplish this, the position setpoint of the position controller is delayed by the amount of the speed control loop substitute time in relation to the velocity precontrol.
	Gain	Current gain of the position controller (Kv)
Run measurement		In the "Run measurement" area, you perform the test steps.
	"Forward" button	With the "Forward" button, you start a test step for optimization in the positive direction.
	"Backward" button	With the "Backward" button, you start a test step for optimization in the negative direction.
	"Stop" button	You stop a test step with the "Stop" button. The axis decelerates with the configured maximum deceleration. If another window is superimposed on the "Stop" button, the axis stops.
Trace		The Trace function is displayed in the lower area of the "Tuning" dialog. With each test step, a Trace recording of the required parameters is automatically started and displayed after completion of the test step. After master control has been returned, the Trace recording is deleted. You can find a full description of the trace function in the section on using the "trace and logic analyzer function".

Note

No automatic transfer of the parameters to the technology object

The configured parameter values are discarded after master control is returned.

Transfer the values as needed into your configuration. You can apply the values for the gain, precontrol and speed control loop substitute time in your configuration using the "Project start value" value.

See also

Closed-loop control (Page 1693)

Optimize position controller (S7-1500)

Requirements

- The CPU must be in the RUN mode.
- The project has been created and downloaded to the CPU.
- The technology object is disabled by your user program (MC_Power.Enable = FALSE).
- The axis control panel for the technology object is not used by another installation of the TIA Portal (Team Engineering as of CPU V1.5).

Optimizing position control gain (Kv)

Proceed as follows to optimize the gain (Kv):

1. In the "Master control" area, click the "Activate" button to activate master control for the technology object, and to establish an online connection to the CPU.
   A warning message is displayed.
2. If necessary, adapt the sign-of-life monitoring and confirm with "OK".
3. In the "Axis" area, click the "Enable" button to enable the technology object.
4. If necessary, configure values for the distance, duration, and dynamics of a test step in the "Measurement configuration" area.
5. If necessary, configure values for the precontrol and speed control loop substitute time of a test step in the "Configure position controller" area.
6. Enter a start value for the gain. Start with a low value.

7. Click the "Forward" or "Start" button to start a test step for the optimization.
   For the specified duration, a setpoint is output according to the specified distance. The axis moves by the specified distance. A trace recording of the motion (setpoint and actual values) is created automatically.
   You can find a full description of the trace function in the section on using the "trace and logic analyzer function".

   Note

   Adapt following error limits

   If error messages from following error monitoring are repeatedly displayed during optimization, temporarily adapt the following error limits.

8. Evaluate the curve of the trace recording. Adapt the gain incrementally. Click the "Forward" or "Backward" button after each value you enter. This applies the value and start a new movement and trace recording each time.

When adjusting the gain, pay attention to the following properties of the curve:

- The curve shows a brief compensation time.
- The curve does not show any motion reversal of the actual position.
- When approaching the position setpoint, no overshoot occurs.
- The curve shows a stable overall behavior (oscillation-free curve).

The following trace recording shows a curve with a long settling time:

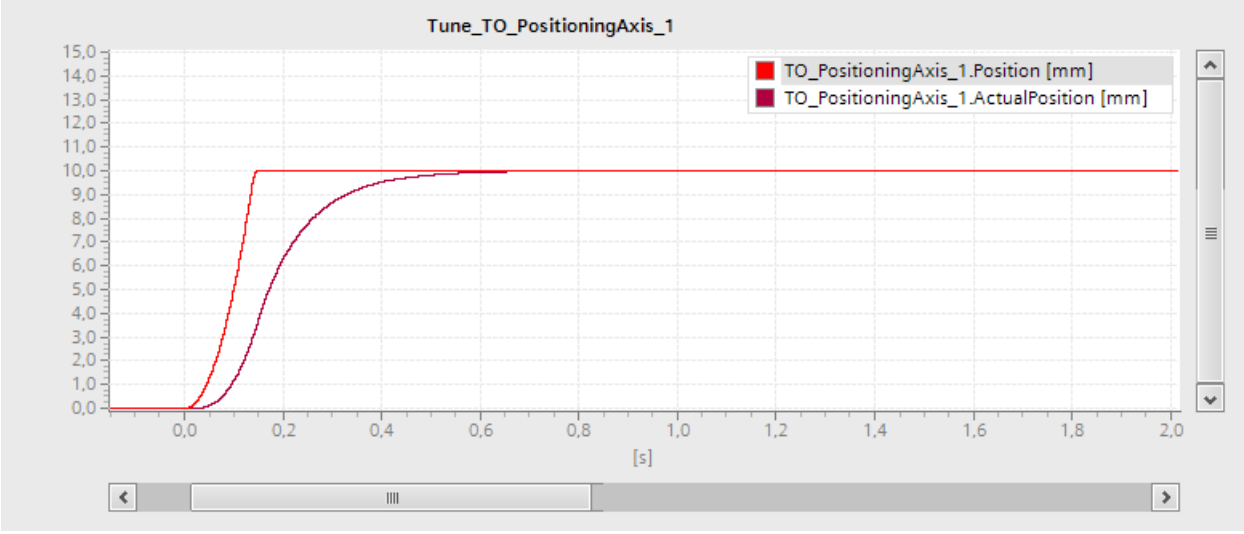

The following trace recording shows a curve with overshoot when approaching the setpoint:

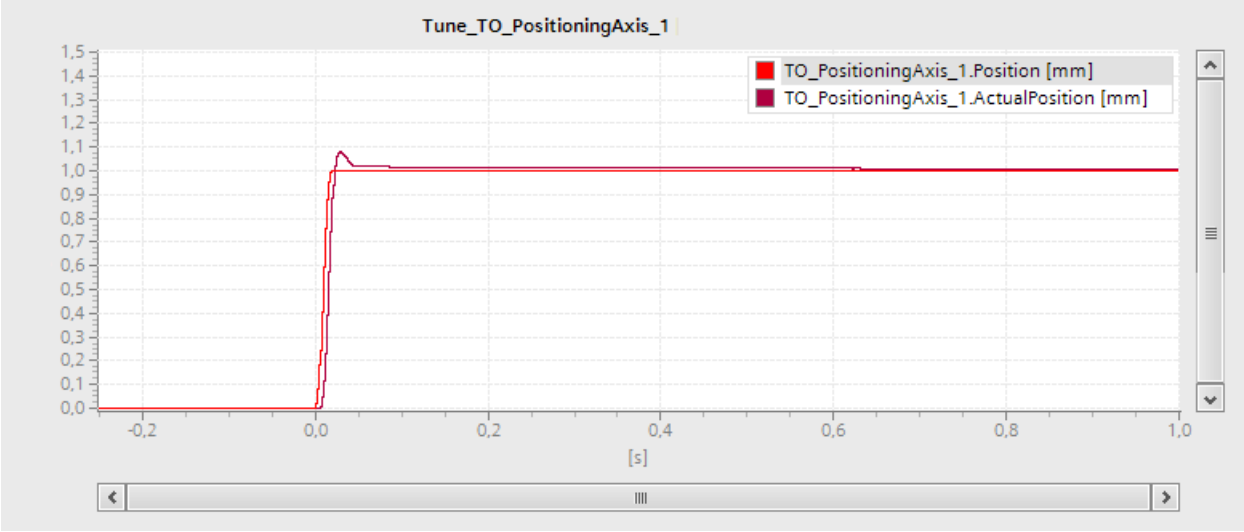

The following trace recording shows a curve in which the gain is optimal and the overall response is steady:

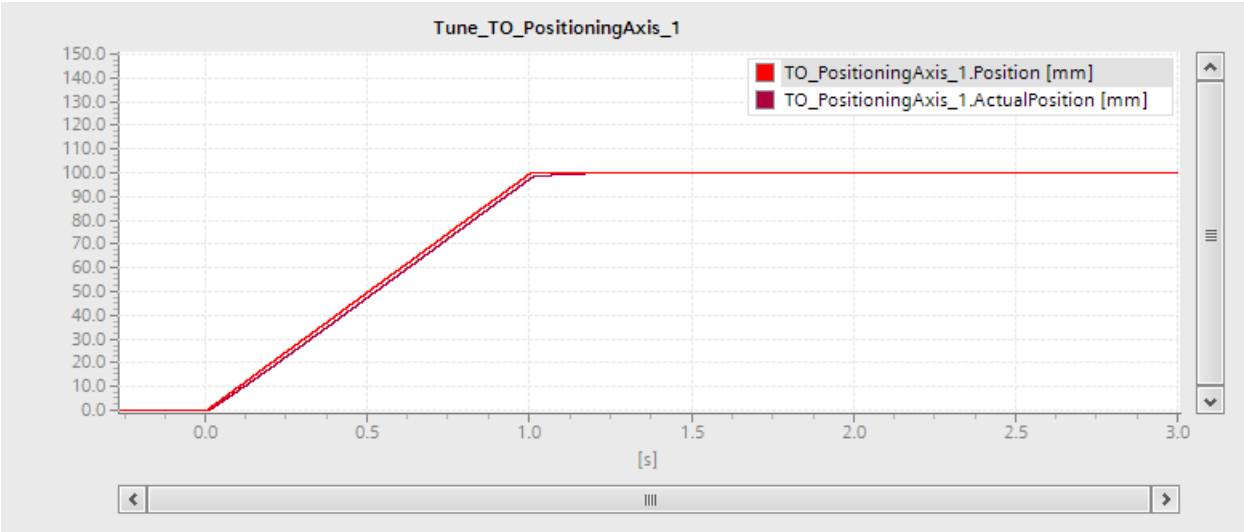

Transferring the parameter values of the position controller to the project

To transfer the determined parameter values of the position controller to your project, follow these steps:

1. Click the icon next to the field of the respective parameter.
   A list of values is displayed.
2. Enter the measured value in the "Project start value" field of the value list.
3. In the "Axis" area, click the "Disable" button to disable the technology object.
4. In the "Master control" area, click the "Deactivate" button to return master control to your user program.
5. Load your project into the CPU.

6.1.2.8 Diagnostics (S7-1500)

The Diagnostics section is limited to describing the diagnostic concept for Motion Control and describing the Diagnostics view of the individual technology objects in the TIA Portal.

For more information about system diagnostics with the S7-1500 CPU, refer to the "Diagnostics" (http://support.automation.siemens.com/WW/view/en/59192926) function manual.

Diagnostic concept (S7-1500)

The diagnostic concept encompasses alarms and associated messages, as well as error messages in the Motion Control instructions. The TIA Portal also supports you with consistency checks during configuration of the technology objects, and during the creation of your user program.

All alarms in runtime (from the CPU, technology, hardware etc.) are displayed in the Inspector window of the TIA Portal. Diagnostic information that relates to technology objects (technology alarms, status information) are additionally displayed in the Diagnostics window of the respective technology object.

During motion control, if an error occurs at a technology object (e.g. approaching a hardware limit switch), then a technology alarm (Page 1962) is triggered, and a corresponding message is displayed in the TIA Portal as well as on HMI devices.

In your user program, technology alarms are generally signaled via error bits in the technology data block. The number of the technology alarm with the highest priority is also displayed. In order to simplify error evaluation, the "Error" and "ErrorID" parameters of the Motion Control instructions also indicate that a technology alarm is pending.

Program errors (Page 1966) can occur during parameter assignment or during the processing sequence of the Motion Control instructions (e.g. invalid parameter specification when calling the instruction, initiation of a job without enable via "MC_Power"). Motion Control instruction errors are indicated at the call of instructions using the "Error" and "ErrorID" parameters.

Technology alarms (S7-1500)

If an error occurs at a technology object (e.g. approaching a hardware limit switch), a technology alarm is triggered and indicated. The impact of a technology alarm on the technology object is specified by the alarm reaction.

Alarm classes

Technology alarms are divided into three classes:

- Acknowledgeable warning
  The processing of Motion Control job is continued. The current motion of the axis can be influenced, e.g. by limiting the current dynamic values to the configured limit values.
- Alarm requiring acknowledgment
  Motion jobs are aborted in accordance with the alarm reaction. You must acknowledge the alarms in order to continue execution of new jobs after eliminating the cause of the error.
- Fatal error
  Motion jobs are aborted in accordance with the alarm reaction.
  To be able to use the technology object again after eliminating the cause of the error, you must restart the technology object (Page 1945).

Display of technology alarms

A technology alarm is displayed in the following locations:

- TIA Portal
  - "Technology object > Diagnostics > Status and error bits"
    Display of pending technology alarms for each technology object.
  - "Technology object > Commissioning > Axis control panel"
    Display of the last pending technology alarm for each technology object.
  - "Inspector window > Diagnostics > Message display"
    Select the "Receive messages" check box under "Online & Diagnostics > Online Access" in order to display technology alarms via the message display.
    With an online connection to the CPU, the pending technology alarms for all technology objects are displayed. Additionally, the archive view is available to you.
    The message display can also be activated and displayed on a connected HMI.
  - "CPU > Online & diagnostics"
    Display of the technology alarms that have been entered in the diagnostic buffer.
- User program
  - Tags "<TO>.ErrorDetail.Number" and "<TO>.ErrorDetail.Reaction"
    Indication of the number and the reaction of the technology alarm with the highest priority.
  - Tag "<TO>.StatusWord"
    A pending technology alarm is indicated with bit 1 ("Error").
  - Tag "<TO>.ErrorWord"
    Indication of alarms and fatal errors.
  - Tag "<TO>.WarningWord"
    Indication of warnings.
  - Parameter "Error" and "ErrorID"
    In a Motion Control instruction, the parameters "Error" = TRUE and "ErrorID" = 16#8001 indicate that a technology alarm is pending.
- Display of the CPU
  In order to show technology alarms on the CPU display, make the following setting when loading to the CPU:
  In the "Load preview" dialog, select the action "Consistent download" for the "Text libraries" entry.

Alarm reaction

A technology alarm always contains an alarm reaction, which describes the impact on the technology object. The alarm reaction is specified by the system.

The following table shows possible alarm reactions:

Alarm reaction	Description
Axes (speed axis, positioning axis, synchronous axis)	
No reaction (warnings only) <TO>.ErrorDetail.Reaction = 0	The processing of Motion Control job is continued. The current motion of the axis can be influenced, e.g. by limiting the current dynamic values to the configured limit values.

Alarm reaction	Description
Stop with current dynamic values <TO>.ErrorDetail.Reaction = 1	Active motion commands are aborted. The axis is braked with the dynamic values that present in the Motion Control instruction and brought to a standstill.
Stop with maximum dynamic values <TO>.ErrorDetail.Reaction = 2	Active motion commands are aborted. The axis is braked with the dynamic values configured under "Technology object > Extended parameters > Dynamic limits", and brought to a standstill. The configured maximum jerk is hereby taken into account.
Stop with emergency stop ramp <TO>.ErrorDetail.Reaction = 3	Active motion commands are aborted. The axis is braked with the emergency stop deceleration configured under "Technology object > Extended parameters > Emergency stop ramp", without any jerk limit, and brought to a standstill.
Remove enable <TO>.ErrorDetail.Reaction = 4	Active motion commands are aborted. The setpoint zero is output and the enable is removed. The axis is braked to a standstill according to the configuration in the drive.
Track setpoints <TO>.ErrorDetail.Reaction = 5	Active motion commands are aborted. The setpoint zero is output. The actual values supplied by the drive are automatically tracked as setpoints.
Other technology objects (output cam, cam track, measuring input, cam, external encoder)	
No reaction (warnings only) <TO>.ErrorDetail.Reaction = 0	The processing of Motion Control job is continued. The current motion of the axis can be influenced, e.g. by limiting the current dynamic values to the configured limit values.
Terminate processing of the technology object: • Output cam <TO>.ErrorDetail.Reaction = 6 • Cam track <TO>.ErrorDetail.Reaction = 7 • Measuring input <TO>.ErrorDetail.Reaction = 8 • Cam <TO>.ErrorDetail.Reaction = 9 • External encoder <TO>.ErrorDetail.Reaction = 10	Processing of the technology object is terminated. All running Motion Control jobs are aborted.

Acknowledging technology alarms

You can acknowledge technology alarms as follows:

- TIA Portal
  - "Technology object > Commissioning > Axis control panel"
    Click "Confirm" to acknowledge all alarms and warnings pending for the selected technology object.
  - "Inspector window > Diagnostics > Message display"
    You can acknowledge the alarms and warnings for all technology objects either individually, or all at once.
- HMI
  At an HMI with enabled message display, you can acknowledge the alarms and warnings for all technology objects either individually, or all at once.
- User program
  Acknowledge pending technology alarms for a technology object with the Motion Control instruction "MC_Reset".

Additional information

You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.

Errors in Motion Control instructions (S7-1500)

Errors in Motion Control instructions (e.g. invalid parameter value setting) are indicated by the "Error" and "ErrorID" output parameters.

Under the following conditions, "Error" = TRUE and "ErrorID" = 16#8xxx are indicated for the Motion Control instruction:

- Illegal status of the technology object, which prevents the execution of the job.
- Invalid parameter assignment of the Motion Control instruction, which prevents the execution of the job.
- As a result of the alarm reaction for a technology object error.

Error display

If there is a Motion Control instruction error, the "Error" parameter shows the value "TRUE". The cause of the error is given in the "ErrorID" parameter.

Jobs to the technology object are rejected when "Error" = TRUE. Running jobs are not influenced by rejected jobs.

If "Error" = TRUE and "ErrorID" = 16#8001 is indicated during job execution, a technology alarm has occurred. In this case, evaluate the indication of the technology alarm.

If "Error" = "TRUE " is displayed during execution of a "MC_MoveJog" job, the axis is braked and brought to a standstill. In this case, the deceleration configured for the "MC_MoveJog" instruction takes effect.

Acknowledge error

Acknowledging errors in Motion Control instructions is not required.

Restart a job after resolving the error.

Additional information

You can find a list of the ErrorIDs in the "Error detection (Page 2133)" appendix.

Speed-controlled axis technology object (S7-1500)

Status and error bits (S7-1500)

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the status and error messages for the technology object. The Diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Axis status

The following table shows the possible axis status values:

Status	Description
Simulation active	The axis is simulated in the CPU. Setpoints are not output to the drive.
Enabled	The technology object has been enabled. The axis can be moved with motion jobs. (<TO>.StatusWord.X0 (Enable))
Error	An error occurred at the technology object. Detailed information about the error is available in the "Error" area and in the "<TO>.ErrorDetail.Number" and "<TO>.ErrorDetail.Reaction" tags of the technology object. (<TO>.StatusWord.X1 (Error))
Restart active	The technology object is being reinitialized. (<TO>.StatusWord.X2 (RestartActive))
Axis control panel enabled	The axis control panel is active. The axis control panel has master control over the technology object. The axis cannot be controlled from the user program. (<TO>.StatusWord.X4 (ControlPanelActive))
Drive ready	Drive is ready to execute setpoints. (<TO>.StatusDrive.InOperation)
Restart required	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object. (<TO>.StatusWord.X3 (OnlineStartValuesChanged))

Motion status

The following table shows the possible axis motion status values:

Status	Description
Done (no job running)	No motion job is running for the technology object. (<TO>.StatusWord.X6 (Done))
Jog	The axis is being moved with a job for jog mode of Motion Control instruction "MC_MoveJog" or from the axis control panel. (<TO>.StatusWord.X9 (JogCommand))
Speed setpoint	The axis is traversed with a job with speed setpoint of the Motion Control instruction "MC_MoveVelocity" or using the axis control panel. (<TO>.StatusWord.X10 (VelocityCommand))
Constant speed	The axis is moved with constant speed or is stationary. (<TO>.StatusWord.X12 (ConstantVelocity))
Accelerating	Axis is being accelerated. (<TO>.StatusWord.X13 (Accelerating))
Decelerating	The axis is being decelerated. (<TO>.StatusWord.X14 (Decelerating))
Torque limiting enabled	The configured threshold for the force/torque affects the axis. (<TO>.StatusWord.X27 (InLimitation))

Warnings

The following table shows the possible warnings:

Warning	Description
Configuration	One or several configuration parameters are adapted internally at a given time. (<TO>.WarningWord.X1 (ConfigWarning))
Job rejected	A job cannot be executed. A Motion Control instruction cannot be executed because necessary requirements have not been met. (<TO>.WarningWord.X3 (CommandNotAccepted))
Dynamic limitation	The dynamic values are limited to the dynamic limits. (<TO>.WarningWord.X6 (DynamicError))

Error

The following table shows the possible errors:

Error	Description
System	A system-internal error has occurred. (<TO>.ErrorWord.X0 (SystemFault))
Configuration	A configuration error has occurred. One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.X1 (ConfigFault))
User program	An error occurred in the user program with a Motion Control instruction or its use. (<TO>.ErrorWord.X2 (UserFault))
Drive	An error occurred in the drive. (<TO>.ErrorWord.X4 (DriveFault))
Data exchange	Communication with a connected device is faulty. (<TO>.ErrorWord.X7 (CommunicationFault))
I/O	An error occurred accessing a logical address. (<TO>.ErrorWord.X13 (PeripheralError))
Job rejected	A job cannot be executed. A Motion Control instruction cannot be executed because necessary requirements have not been met (e.g. technology object not homed). (<TO>.ErrorWord.X3 (CommandNotAccepted))
Dynamic limitation	The dynamic values are limited to the dynamic limits. (<TO>.ErrorWord.X6 (DynamicError))

Alarm display

For additional information and to acknowledge the error, go to the Inspector window by clicking on the "Alarm display" link.

Additional information

An option for evaluating the individual status bits can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 1922).

See also

"StatusWord" tag (speed axis) (Page 2020)

"ErrorWord" tag (speed axis) (Page 2023)

"WarningWord" tag (speed axis) (Page 2025)

Motion status (S7-1500)

You use the "Technology object > Diagnostics > Motion status" diagnostic function in the TIA Portal to monitor the motion status of the axis. The Diagnostics function is available in online operation.

"Setpoints" area

The following table shows the meaning of the status data:

Status	Description
Speed setpoint	Speed setpoint of the axis (<TO>.Velocity)
Speed override	Speed setpoint correction as percentage The speed setpoint specified in motion control instructions or set by the axis control panel are superimposed with an override signal and corrected as a percentage. Valid speed correction values range from 0.0 % to 200.0 %. (<TO>.Override.Velocity)

"Current values" area

The following table shows the meaning of the status data:

Status	Description
Actual speed	Actual speed of the axis (<TO>.ActualSpeed)

"Dynamic limits" area

This area displays the limit values for the dynamic parameters.

The following table shows the meaning of the status data:

Status	Description
Speed	Configured maximum speed (<TO>.DynamicLimits.MaxVelocity)
Acceleration	Configured maximum acceleration (<TO>.DynamicLimits.MaxAcceleration)
Deceleration	Configured maximum deceleration (<TO>.DynamicLimits.MaxDeceleration)
Jerk	Configured maximum jerk (<TO>.DynamicLimits.MaxJerk)

PROFIdrive telegram (S7-1500)

The "Technology object > Diagnostics > PROFIdrive telegram" diagnostics function is used in the TIA Portal to monitor the PROFIdrive telegram that the drive returns to the controller. The Diagnostics function is available in online operation.

"Drive" area

This area displays the following parameters contained in the PROFIdrive telegram from the drive to the controller:

- Status words "ZSW1" and "ZSW2"
- The speed setpoint (NSET) that was output to the drive
- The actual speed that was signaled from the drive (NACT)

Positioning axis/synchronous axis technology object (S7-1500)

Status and error bits (S7-1500)

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the status and error messages for the technology object. The Diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Axis status

The following table shows the possible axis status values:

Status	Description
Simulation active	The axis is simulated in the CPU. Setpoints are not output to the drive. (<TO>.StatusWord.X25 (AxisSimulation))
Enabled	The technology object has been enabled. You can move the axis with motion jobs. (<TO>.StatusWord.X0 (Enable))
Position-controlled mode	The axis is in position-controlled mode. (Inversion of <TO>.StatusWord.X28 (NonPositionControlled))
Homed	The technology object is homed. The relationship between the position in the technology object and the mechanical position was successfully created. (<TO>.StatusWord.X5 (HomingDone))
Error	An error occurred at the technology object. Detailed information about the error is available in the "Error" area and in the "<TO>.ErrorDetail.Number" and "<TO>.ErrorDetail.Reaction" tags of the technology object. (<TO>.StatusWord.X1 (Error))
Restart active	The technology object is being reinitialized. (<TO>.StatusWord.X2 (RestartActive))
Axis control panel enabled	The axis control panel is active. The axis control panel has master control over the technology object. You cannot control the axis from the user program. (<TO>.StatusWord.X4 (ControlPanelActive))
Drive ready	Drive is ready to execute setpoints. (<TO>.StatusDrive.InOperation)

Status	Description
Encoder values valid	The actual encoder values are valid. (<TO>.StatusSensor[n].State)
Active encoder	Current encoder on the axis (<TO>.OperativeSensor)
Restart required	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object. (<TO>.StatusWord.X3 (OnlineStartValuesChanged))

Status limit switch

The following table shows the possibilities for enabling the software and hardware limit switches:

Status	Description
Negative SW limit switch approached.	The negative software limit switch was reached. (<TO>.StatusWord.X15 (SWLimitMinActive))
Positive SW limit switch approached.	The positive software limit switch was reached. (<TO>.StatusWord.X16 (SWLimitMaxActive))
Negative HW limit switch approached.	The negative hardware limit switch has been approached or overtraveled. (<TO>.StatusWord.X17 (HWLimitMinActive))
Positive HW limit switch approached.	The positive hardware limit switch has been approached or overtraveled. (<TO>.StatusWord.X18 (HWLimitMaxActive))

Motion status

The following table shows the possible axis motion status values:

Status	Description
Done (no job running)	No job active at technology object. (<TO>.StatusWord.X6 (Done))
Homing job	The technology object executes a homing job of the Motion Control instruction "MC_Home" or from the axis control panel. (<TO>.StatusWord.X11 (HomingCommand))
Jog	The axis is being moved with a command for jog mode of Motion Control instruction "MC_MoveJog". (<TO>.StatusWord.X9 (JogCommand))
Velocity specification	The axis is traversed with a job with velocity specification of the Motion Control instruction "MC_MoveVelocity" or from the axis control panel. (<TO>.StatusWord.X10 (VelocityCommand))
Positioning job	The axis is traversed with a positioning job of Motion Control instruction "MC_MoveAbsolute" or "MC_MoveRelative" or from the axis control panel. (<TO>.StatusWord.X8 (PositioningCommand))
Constant velocity	The axis is moved with constant velocity or is stationary. (<TO>.StatusWord.X12 (ConstantVelocity))

Status	Description
Standstill	The axis is in standstill. (<TO>.StatusWord.X7 (StandStill))
Accelerating	Axis is being accelerated. (<TO>.StatusWord.X13 (Accelerating))
Decelerating	The axis is being decelerated. (<TO>.StatusWord.X14 (Decelerating))
Torque limiting enabled	The configured threshold for the force/torque affects the axis. (<TO>.StatusWord.X27 (InLimitation))
Synchronization	Synchronous axis only The axis is synchronized to the leading value of a leading axis. (<TO>.StatusWord.X21 (Synchronizing))
Synchronous operation	Synchronous axis only The axis is synchronized and moves synchronously to the leading axis. (<TO>.StatusWord.X22 (Synchronous))

Warnings

The following table shows the possible warnings:

Warning	Description
Configuration	One or several configuration parameters are adjusted internally at a certain time. (<TO>.WarningWord.X1 (ConfigWarning))
Job rejected	Job cannot be executed. You cannot execute a Motion Control instruction because necessary requirements are not fulfilled. (<TO>.WarningWord.X3 (CommandNotAccepted))
Dynamic limitation	The dynamic values are limited to the dynamic limits. (<TO>.WarningWord.X6 (DynamicError))
Synchronization	Synchronous axis only An error occurred during synchronization. The leading axis specified for the corresponding Motion Control instruction was not configured as a possible leading axis. (<TO>.WarningWord.X14 (SynchronousError))

Error

The following table shows the possible errors:

Error	Description
System	A system-internal error has occurred. (<TO>.ErrorWord.X0 (SystemFault))
Configuration	A configuration error has occurred. One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.X1 (ConfigFault))
User program	An error occurred in the user program with a Motion Control instruction or its use. (<TO>.ErrorWord.X2 (UserFault))
Drive	An error occurred in the drive. (<TO>.ErrorWord.X4 (DriveFault))
Encoder	An error occurred in the encoder system. (<TO>.ErrorWord.X5 (SensorFault))
Data exchange	Communication with a connected device is faulty. (<TO>.ErrorWord.X7 (CommunicationFault))
I/O	An error occurred accessing a logical address. (<TO>.ErrorWord.X13 (PeripheralError))
Job rejected	A job cannot be executed. You cannot execute a Motion Control instruction because necessary requirements are not fulfilled (for example, technology object not homed). (<TO>.ErrorWord.X3 (CommandNotAccepted))
Homing	An error occurred during a homing process. (<TO>.ErrorWord.X10 HomingFault))
Positioning	The positioning axis was not positioned correctly at the end of a positioning motion. (<TO>.ErrorWord.X12 (PositioningFault))
Dynamic limitation	The dynamic values are limited to the dynamic limits. (<TO>.ErrorWord.X6 (DynamicError))
Following error	The maximum permitted following error has been exceeded. (<TO>.ErrorWord.X11 (FollowingErrorFault))
SW limit switch	A software limit switch has been reached. (<TO>.ErrorWord.X8 (SwLimit))
HW limit switch	A hardware limit switch has been reached or overtraveled. (<TO>.ErrorWord.X9 (HWLimit))
Adaptation	An error occurred during data adaption. (<TO>.ErrorWord.X15 (AdaptionError))
Synchronization	Synchronous axis only An error occurred during synchronization. The leading axis specified for the corresponding Motion Control instruction was not configured as a possible leading axis. (<TO>.ErrorWord.X14 (SynchronousError))

Alarm display

For additional information and to acknowledge the error, go to the Inspector window by clicking on the "Alarm display" link.

Additional information

An option for evaluating the individual status bits can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 1922).

See also

"StatusWord" tag (positioning axis/synchronous axis) (Page 2057)

"ErrorWord" tag (positioning axis/synchronous axis) (Page 2061)

"WarningWord" tag (positioning axis/synchronous axis) (Page 2065)

Additional displays for the technology CPU (Page 1977)

Motion status (S7-1500)

You use the "Technology object > Diagnostics > Motion status" diagnostic function in the TIA Portal to monitor the motion status of the axis. The Diagnostics function is available in online operation.

"Setpoints" area

The following table shows the meaning of the status data:

Status	Description
Target position	Current target position of an active positioning job The target position value is only valid during execution of a positioning job. (<TO>.StatusPositioning.TargetPosition)
Position setpoint	Setpoint position of the axis (<TO>.Position)
Velocity setpoint	Velocity setpoint of the axis (<TO>.Velocity)
Velocity override	Percentage correction of the velocity specification The velocity setpoint specified in Motion Control instructions or set by the axis control panel is superimposed with an override signal and corrected as a percentage. Valid velocity correction values range from 0.0 % to 200.0 %. (<TO>.Override.Velocity)

"Current values" area

The following table shows the meaning of the status data:

Status	Description
Operative encoder	Operative encoder of the axis
Actual position	Actual position of the axis If the technology object is not homed, then the value is displayed relative to the position that existed when the technology object was enabled. (<TO>.ActualPosition)
Actual velocity	Actual velocity of the axis (<TO>.ActualVelocity)
Following error	Following error of the axis (<TO>.StatusPositioning.FollowingError)

"Dynamic limits" area

This area displays the limit values for the dynamic parameters.

The following table shows the meaning of the status data:

Status	Description
Velocity	Configured maximum velocity (<TO>.DynamicLimits.MaxVelocity)
Acceleration	Configured maximum acceleration (<TO>.DynamicLimits.MaxAcceleration)
Deceleration	Configured maximum deceleration (<TO>.DynamicLimits.MaxDeceleration)
Jerk	Configured maximum jerk (<TO>.DynamicLimits.Jerk)

See also

Additional displays for the technology CPU (Page 1977)

PROFIdrive telegram (S7-1500)

The "Technology object > Diagnostics > PROFIdrive telegram" diagnostics function is used in the TIA Portal to monitor the PROFIdrive telegrams returned by the drive and encoder. The display of the Diagnostics function is available in online operation.

"Drive" area

This area displays the following parameters contained in the PROFIdrive telegram from the drive to the controller:

- Status words "ZSW1" and "ZSW2"
- The speed setpoint (NSET) that was output to the drive
- The actual speed that was signaled from the drive (NACT)

"Encoder" area

This area displays the following parameters contained in the PROFIdrive telegram from the encoder to the controller:

- Status word "Gx_ZSW"
- The actual position value "Gx_XIST1" (cyclic actual encoder value)
- The actual position value "Gx_XIST2" (absolute encoder value)

See also

Additional displays for the technology CPU (Page 1977)

Additional displays for the technology CPU (S7-1500)

The diagnostic functions contain expanded/additional displays for the S7-1500T technology CPU.

Status and error bits

The following table shows the expanded/additional displays of the status and error bits (Page 1970):

Axis status	
Encoder values valid	The actual encoder values (encoder 1, encoder 2, encoder 3 or encoder 4) are valid. (<TO>.StatusSensor[n].State)
Active encoder	The encoder in effect operationally is encoder 1, encoder 2, encoder 3 or encoder 4. (<TO>.OperativeSensor)

Motion status

The following table shows the expanded/additional displays for the status of the motion (Page 1974):

Current values	Description
Operative encoder	Operative encoder of the axis

PROFIdrive telegram

The "Encoder 1" to "Encoder 4" areas display the following parameters from the PROFIdrive telegram (Page 1975) of the corresponding encoder to the controller:

- Status word "Gx_ZSW"
- The actual position value "Gx_XIST1" (cyclic actual encoder value)
- The actual position value "Gx_XIST2" (absolute encoder value)

See also

Status and error bits (Page 1970)

Motion status (Page 1974)

PROFIdrive telegram (Page 1975)

Technology object external encoder (S7-1500)

Status and error bits (S7-1500)

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the status and error messages for the technology object. The diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Encoder status

The following table shows the possible external encoder status values:

Status	Description
Encoder enabled	The technology object has been enabled. (<TO>.StatusWord.X0 (Enable))
Homed	The technology object is homed. The relationship between the position in the technology object and the mechanical position was successfully created. (<TO>.StatusWord.X5 (HomingDone))
Error	An error occurred at the technology object. Detailed information about the error is available in the "Error" area and in the "<TO>.ErrorDetail.Number" and "<TO>.ErrorDetail.Reaction" tags of the technology object. (<TO>.StatusWord.X1 (Error))
Restart active	The technology object is being reinitialized. (<TO>.StatusWord.X2 (RestartActive))
Encoder values valid	The actual encoder values are valid. (<TO>.StatusSensor[n].State)
Restart required	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object. (<TO>.StatusWord.X3 (OnlineStartValuesChanged))

Motion status

The following table shows the possible states of the job execution:

Status	Description
Done (no job running)	No Motion Control job is running for the technology object. (Enable by "MC_Power" job excepted) (<TO>.StatusWord.X6 (Done))
Homing job	The technology object executes a homing job of the Motion Control instruction "MC_Home". (<TO>.StatusWord.X11 (HomingCommand))

Error

The following table shows the possible errors:

Error	Description
System	A system-internal error has occurred. (<TO>.ErrorWord.X0 (SystemFault))
Configuration	A configuration error has occurred. One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.X1 (ConfigFault))
User program	An error occurred in the user program with a Motion Control instruction or its use. (<TO>.ErrorWord.X2 UserFault))
Encoder	An error occurred in the encoder system. (<TO>.ErrorWord.X5 (SensorFault))
Data exchange	Missing or faulty communication. (<TO>.ErrorWord.X7 (CommunicationFault))
Adaptation	An error occurred during data adaption. (<TO>.ErrorWord.X15 (AdaptionError))

Alarm display

For additional information and to acknowledge the error, go to the Inspector window by clicking on the "Alarm display" link.

Additional information

An option for evaluating the individual status bits can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 1922).

See also

"StatusWord" tag (external encoder) (Page 2078)

"ErrorWord" tag (external encoder) (Page 2079)

"WarningWord" tag (external encoder) (Page 2081)

Motion status (S7-1500)

You use the "Technology object > Diagnostics > Motion status" diagnostic function in the TIA Portal to monitor the actual encoder values. The diagnostics function is available in online operation.

"Current values" area

The following table shows the meaning of the status data:

Status	Description
Actual position	Actual position of the axis If the technology object is not homed, then the value is displayed relative to the position that existed when the technology object was enabled. (<TO>.ActualPosition)
Actual velocity	Actual velocity of the axis (<TO>.ActualVelocity)

PROFIdrive telegram (S7-1500)

The "Technology object > Diagnostics > PROFIdrive telegram" diagnostic function is used in the TIA Portal to monitor the PROFIdrive telegram of the encoder. The display of the diagnostics function is available in technology object online mode.

"Encoder" area

This area displays the following parameters contained in the PROFIdrive telegram that the encoder returns to the controller:

- Status word "G1_ZSW"
- The actual position value "G1_XIST1" (cyclic actual encoder value)
- The actual position value "G1_XIST2" (absolute value of the encoder)

Technology object measuring input (S7-1500)

Status and error bits (S7-1500)

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the status and error messages for the technology object. The diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Measuring input status

The following table shows the possible states of the measuring input:

Status	Description
Active	The technology object is in operation. (<TO>.StatusWord.X0 (Control))
Waiting for measuring event	The measuring input is waiting for a measuring event. The technology data block tag "<TO>.Status" has the value "1" ("WAITING_FOR_TRIGGER").
Measured value present	The measuring input has acquired one or more measured values. The technology data block tag "<TO>.Status" has the value "2" ("TRIGGER_OCCURRED").
Error	An error occurred at the technology object. Detailed information about the error is available in the "Error" area and in the "<TO>.ErrorDetail.Number" and "<TO>.ErrorDetail.Reaction" tags of the technology object. (<TO>.StatusWord.X1 (Error))
Restart active	The technology object is being reinitialized. The tags of the technology data block are not updated with active restart. (<TO>.StatusWord.X2 (RestartActive))
Measuring input ready	The measuring input is synchronized with the measuring module and can be used. (<TO>.StatusWord.X5 (CommunicationOK))
Restart required	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object. (<TO>.StatusWord.X3 (OnlineStartValuesChanged))

Error

The following table shows the possible errors:

Error	Description
System	A system-internal error has occurred. (<TO>.ErrorWord.X0 (SystemFault))
Configuration	A configuration error has occurred. One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.X1 (ConfigFault))
User program	An error occurred in the user program with a Motion Control instruction or its use. (<TO>.ErrorWord.X2 (UserFault))

Error	Description
Job rejected	A job cannot be executed. A Motion Control instruction cannot be executed because the necessary conditions are not met (e.g. axis assigned to the measuring input is not homed). (<TO>.ErrorWord.X3 (CommandNotAccepted))
I/O	An error occurred accessing a logical address. (<TO>.ErrorWord.X13 (PeripheralError))

Alarm display

For additional information and to acknowledge the error, go to the Inspector window by clicking on the "Alarm display" link.

Additional information

An option for evaluating the individual status bits can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 1922).

See also

"StatusWord" tag (measuring input) (Page 2086)

"ErrorWord" tag (measuring input) (Page 2087)

"WarningWord" tag (measuring input) (Page 2088)

Technology object output cam (S7-1500)

Status and error bits (S7-1500)

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the status and error messages for the technology object. The diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Output cam status

The following table shows the possible states of the output cam:

Status	Description
Active	The technology object is in operation. (<TO>.StatusWord.X0 (Control))
Switched	The output cam is switched. (<TO>.CamOutput)

Status	Description
Inverted output cam output	The output cam output is inverted. (<TO>.StatusWord.X4 (OutputInverted))
Error	An error occurred at the technology object. Detailed information about the error is available in the "Error" area and in the "<TO>.ErrorDetail.Number" and "<TO>.ErrorDetail.Reaction" tags of the technology object. (<TO>.StatusWord.X1 (Error))
Restart active	The technology object is being reinitialized. The tags of the technology data block are not updated with active restart. (<TO>.StatusWord.X2 (RestartActive))
Output cam output ready	The cam is synchronized with the output module and available for use. (<TO>.StatusWord.X5 (CommunicationOk))
Restart required	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object. (<TO>.StatusWord.X3 (OnlineStartValuesChanged))

Error

The following table shows the possible errors:

Error	Description
System	A system-internal error has occurred. (<TO>.ErrorWord.X0 (SystemFault))
Configuration	A configuration error has occurred. One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.X1 (ConfigFault))
User program	An error occurred in the user program with a Motion Control instruction or its use. (<TO>.ErrorWord.X2 (UserFault))
Job rejected	A job cannot be executed. A Motion Control instruction cannot be executed because the necessary conditions are not met (e.g. axis assigned to the output cam is not homed). (<TO>.ErrorWord.X3 (CommandNotAccepted))
I/O	An error occurred accessing a logical address. (<TO>.ErrorWord.X13 (PeripheralError))

Alarm display

For additional information and to acknowledge the error, go to the Inspector window by clicking on the "Alarm display" link.

Additional information

An option for evaluating the individual status bits can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 1922).

See also

"StatusWord" tag (output cam) (Page 2092)
"ErrorWord" tag (output cam) (Page 2093)
"WarningWord" tag (output cam) (Page 2094)

Cam track technology object (S7-1500)

Status and error bits (S7-1500)

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the status and error messages for the technology object. The diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Cam track status

The following table shows the possible states of the cam track:

Status	Description
Active	The technology object is in operation. (<TO>.StatusWord.X0 (Control))
Switched	A cam of cam track is switched. (<TO>.TrackOutput)
Inverted output cam output	The output cam output is inverted. (<TO>.StatusWord.X4 (OutputInverted))
Error	An error occurred at the technology object. Detailed information about the error is available in the "Error" area and in the "<TO>.ErrorDetail.Number" and "<TO>.ErrorDetail.Reaction" tags of the technology object. (<TO>.StatusWord.X1 (Error))
Restart active	The technology object is being reinitialized. The tags of the technology data block are not updated with active restart. (<TO>.StatusWord.X2 (RestartActive))
Cam track output ready	The cam track is synchronized with the output module and available for use. (<TO>.StatusWord.X5 (CommunicationOk))
Restart required	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object. (<TO>.StatusWord.X3 (OnlineStartValuesChanged))

Error

The following table shows the possible errors:

Error	Description
System	A system-internal error has occurred. (<TO>.ErrorWord.X0 (SystemFault))
Configuration	A configuration error has occurred. One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.X1 (ConfigFault))
User program	An error occurred in the user program with a Motion Control instruction or its use. (<TO>.ErrorWord.X2 (UserFault))
Job rejected	A job cannot be executed. A Motion Control instruction cannot be executed because the necessary conditions are not met (e.g. axis assigned to the cam track is not homed). (<TO>.ErrorWord.X3 (CommandNotAccepted))
I/O	An error occurred accessing a logical address. (<TO>.ErrorWord.X13 (PeripheralError))

Alarm display

For additional information and to acknowledge the error, go to the Inspector window by clicking on the "Alarm display" link.

Additional information

An option for evaluating the individual status bits can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 1922).

See also

"StatusWord" tag (cam track) (Page 2098)

"ErrorWord" tag (cam track) (Page 2099)

"WarningWord" tag (cam track) (Page 2100)

Cam track status (S7-1500)

You use the "Technology object > Diagnostics > Cam track status" diagnostics function in the TIA Portal to monitor the status of the cam track. The diagnostics function is available in online operation.

"Validity and masking of the output cams" area

The individual cams of a cam track are shown in this area along with the status for the following properties:

Status	Description
Valid	Validity of the individual cams of the cam track (<TO>.Parameter.Cam[1..32].Existent)
Masked	Bit masking of the individual cams of the cam track (<TO>.Parameter.CamMasking)
Effective	Switched on output cam (bit-masked) (<TO>.SingleCamState)

"Positions" area

The following status values are displayed in this area:

Status	Description
Current position in the cam track	Position during cam track processing within a cam track cycle The distance to the current reference position of the current cam track (<TO>.MatchPosition) is displayed. (<TO>.TrackPosition)
Current cam track start	Reference position of the current cam track During cyclic processing of the cam track, the continued reference position of the current cam track is displayed. The unique detection and output of the position is only possible when the assigned technology object is in motion. (<TO>.MatchPosition)

6.1.2.9 Technology Version V3.0 (S7-1500)

Configuring (S7-1500)

Configuring the cam technology object

Graphical editor

Structure of Graphical Editor V3.0

The graphical editor is divided into the following areas:

- Toolbar
- Curve diagram

Toolbar

The toolbar at the top of the graphical editor provides you with buttons for the following functions:

Button	Function	Description
	Importing cam from file	See section Importing/exporting cam (Page 1889).
	Exporting cam to file	See section Importing/exporting cam (Page 1889).
	Edit elements/Move view	• Selecting and moving of individual elements and element • Moving the view using drag-and-drop To switch from any tool to the "Edit elements/Move view" tool, press the <Esc> key.
	Activate zoom selection	Zoom into selected area
	Activate vertical zoom	Vertical zoom into selected area without horizontal scaling Alternative: <Ctrl> +drag to ordinate keeping mouse button pressed
	Activate horizontal zoom	Horizontal zoom into selected area without vertical scaling Alternative: <Ctrl> +drag to abscissa keeping mouse button pressed
	Zoom in	Enlargement of the display Alternative: <Ctrl> + mouse wheel up in curve diagram
	Zoom out	Reduction of the display Alternative: <Ctrl> + mouse wheel down in curve diagram
	Show all	Display of entire definition and value range
	Activate snap grid	Inputs and element end points are aligned to the configurable snap grid and to other element end points.
	Inserting a point	Adding a point to the chart
	Inserting a line	Adding a line to the chart
	Inserting a sine	Adding a sine element to the chart
	Inserting a polynomial	Adding a polynomial to the chart
	Inserting an inverse sine	Adding an inverse sine to the chart
	View: One chart	Display of one chart with the following curves of the cam opened in the editor: • Preset curve • Effective position • Effective velocity • Effective acceleration • Effective jerk
	View: Four charts	Display of four charts with the following curves of the cam opened in the editor: • Chart with setpoint curve and effective position • Chart with effective velocity • Chart with effective acceleration • Chart with effective jerk

Button	Function	Description
	Vertical measuring lines	Displaying and moving of vertical measuring lines Hold down the left mouse button and drag to draw a measuring range. The vertical position of the measuring lines can be moved. The function values for the measuring line positions are displayed in the chart. The difference of the measuring lines is displayed between the measuring lines. If you position the mouse pointer on a measuring line, the function value is displayed.
	Horizontal measuring lines	Displaying and moving of horizontal measuring lines Hold down the left mouse button and drag to draw a measuring range. The horizontal position of the measuring lines can be moved. The function values for the measuring line positions are displayed in the chart. The difference of the measuring lines is displayed between the measuring lines. If you position the mouse pointer on a measuring line, the function value is displayed.
	Show legend	Showing or hiding of the legend in the curve diagram. To display values for a specific curve on the ordinate, click on the name of the corresponding curve in the legend.
	Show legend left	Display of the legend on the left side of the curve diagram.
	Show legend right	Display of the legend on the right side of the curve diagram.
	Show online curve	Display of the cam read back from the CPU (orange)

Curve diagram

In the curve diagram, you enter the elements of the curve and adjust the curve by selecting and moving elements.

Chart areas outside of the leading value/following value range configured in "Profile > General (Page 1869)" are grayed out. Elements that fall outside are displayed with a warning ("Element is outside the definition range").

Different curves (position, velocity, acceleration and jerk) can be displayed one above the other in up to four charts by configuring the graphical view accordingly. If multiple charts are to be displayed, the height of the charts can be adjusted using the separating lines.

The view can be zoomed in manual mode with <Ctrl> +mouse wheel and <Ctrl> +drag to abscissa/ordinate with mouse button pressed.

Any calculation problems that may occur are signaled by warning triangles . The alarm text is displayed in the tooltip of the warning triangle.

Shortcut menu in the Graphical Editor V3.0

The following table shows the functions in the shortcut menu of the graphical editor:

Function	Description
Show all	Display of entire definition and value range
Zoom in	Enlargement of the display
Zoom out	Reduction of the display
Cut	Removing the selected elements and copying them to the clipboard
Copy	Copying of the selected elements to the clipboard

Function	Description
Paste	Pasting of the elements from the clipboard to the last element
Delete	Deletion of the selected elements Transitions to existing elements are also deleted.
Paste special	Call of the "Paste elements (Page 1891)" dialog
Move	Call of the "Move elements (Page 1891)" dialog
Scale	Call of the "Scale elements (Page 1891)" dialog

See also

Dialogs in the shortcut menu (Page 1891)

Tabular editor

Shortcut menu in the Tabular Editor V3.0

The following table shows the functions in the shortcut menu of the tabular editor:

Function	Description
Insert row	Insertion of an element (point) in front of the selected element
Add row	Insertion of an element (point) after the selected element
Cut	Removing the selected elements and copying them to the clipboard
Copy	Copying of the selected elements to the clipboard
Paste	Pasting of the elements from the clipboard to the last element
Delete	Deletion of the selected elements Transitions to existing elements are also deleted.
Paste special	Call of the "Paste elements (Page 1891)" dialog
Move	Call of the "Move elements (Page 1891)" dialog
Scale	Call of the "Scale elements (Page 1891)" dialog

Properties (Inspector window)

Profile Statistics V3.0

The "Statistics" properties window shows an overview of the number of elements of the cam. A cam can consist of up to 1000 points and up to 50 segments.

Used elements

This area shows the number of used elements of the curve:

Parameters	Description
Points	This field shows the number of used points of the cam. A cam can consist of up to 1000 points.
Segments	This field shows the number of used segments of the cam. A cam can consist of up to 50 segments.

The use of points and segments depends on the compilation and configuration of the elements. The following table shows the use of points and segments per element:

Element	Number of used points	Number of used segments
Point	1	0
Point at a transition with VDI-based optimization	0	0
Line	0	1
Sine	0	1
Polynomial	0	1
Inverse sine	Number of interpolation points configured. (Properties (Inspector window) > Element > Parameter > Approximation > Number of interpolation points) Default: 32	0
Inverse sine to the right of a transition with VDI-based optimization	Number of interpolation points configured - 1	0
Transition with system interpolation	0	0

Element	Number of used points	Number of used segments
Transition with VDI-based optimization		
Motion rule		
Sine	0	1
Inclined sine	0	1
Polynomial	0	1
Modified acceleration trapezoid		
Motion task		
Dwell-in-reverse	0	5
Reverse-in-dwell	0	5
Dwell-in-dwell	0	6
Modified sine		
Motion task		
Dwell-in-dwell	0	3
Constant velocity-in-constant velocity	0	4
Constant-velocity-in-dwell	0	4
Dwell-in-constant velocity	0	4
Sine line combination	0	3
Harmonic combination	0	3

Boundary conditions

The following boundary conditions apply to the input and use of points and segments:

- **Points**
  For points with the same leading values, the point that was entered last or is in a lower part of the tabular editor becomes effective.
- **Segments**
  - Gaps between segments are filled with a transition segment.
  - For gaps in the leading value range of less than 1.0E-4, segment end points and segment start points are pulled together.
  - For gaps in the leading value range greater than 1.0E-4, a new transition segment is inserted.
  - For overlaps, the new segment is inserted from the start point and used completely. When the previous segment is defined in excess of the new segment, the previous segment continues to be used after the end point of the new segment.
- **Interpolation points and segments** (mixed cams)
  When segments and points are defined in the same range, the segment is used.

Configuration Elements - Parameters (polynomial) V3.0

Configure the parameters of the selected element in the "Parameters" configuration window.

Parameters

Configure the parameters of the selected polynomial in this area:

Parameters		Description
Leading values of the polynomial		
	Start	Configure the start point of the polynomial in the leading value range (definition range) in this field.
	End	Configure the end point of the polynomial in the leading value range (definition range) in this field.
Polynomial parameters		
	Definition by	In the drop-down list, select how the polynomial is defined: • Coefficients • Boundary values and point of inflection • Boundary values and left-side jerk • Boundary values and right-side jerk The corresponding parameters are displayed based on the selection.
	Coefficients	In these fields, configure the coefficients of the polynomial of degree 6: $P(x) = a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ The coefficients are shown in scientific notation, e.g. "9.6450617283e-11".
	Following value - Start	Configure the following value at the start of the polynomial in this field.
	Following value - End	Configure the following value at the end of the polynomial in this field.
	First derivative - Start	Configure the first derivative (velocity) for the following value at the start of the polynomial in this field.
	First derivative - End	Configure the first derivative (velocity) for the following value at the end of the polynomial in this field.
	Second derivative - Start	Configure the second derivative (acceleration) for the following value at the start of the polynomial in this field.
	Second derivative - End	Configure the second derivative (acceleration) for the following value at the end of the polynomial in this field.
	Third derivative - Start	Configure the third derivative (jerk) for the following value at the start of the polynomial in this field.
	Third derivative - End	Configure the third derivative (jerk) for the following value at the end of the polynomial in this field.
	Leading value of the point of inflection	Configure the leading value of the point of inflection of the polynomial in this field.
Extended parameters		

Parameters		Description
	Segment type	In the drop-down list, select whether or not the polynomial is to have a trigonometric component. When "Polynomial with trigonometric portion" is selected, the corresponding trigonometric parameters are displayed, as they are with sine. When a sine element is converted to a polynomial, the sine element is configured as a polynomial with trigonometric portion. The shape of the element is retained.
	Amplitude	Configure the amplitude of the trigonometric component in this field.
	Definition by	In the drop-down list, select how the trigonometric component is defined: • Phase at start and at end • Phase at start and period length • Phase at start and frequency • Period length and phase at end • Frequency and phase at end The corresponding parameters are displayed based on the selection.
	Phase at start	Configure the phase angle at the start of the trigonometric component in this field.
	Phase at end	Configure the phase angle at the end of the trigonometric component in this field.
	Period length	Configure the period length of the trigonometric component in this field.
	Frequency	Configure the frequency of the trigonometric component in this field.

Configuration elements - Characteristics (transition) V3.0

Configure the parameters of the selected transition in the "Characteristics" configuration window.

Parameters

Configure the settings for optimization of the transition in this area:

Parameters		Description
Interpolation settings of the transition		
	Optimization method	Select the optimization method in the drop-down list. • System interpolation The CPU defines the optimization parameters automatically according to the settings of the system interpolation (Page 1871). • VDI-based optimization You adjust the optimization manually. The inputs are applied automatically according to the VDI Guideline 2143.
	Motion task	The transition type is determined from the properties of the adjacent elements of the transition and displayed in this field.
	Continuity at start/end	In the drop-down lists, select which parameter is continuous in the boundary points and is to be included for optimization. • Default optimization setting (setting under "Profile > Default optimization settings (Page 1870)") • Position • Velocity (bumpless) • Acceleration (jerkless) • Jerk (jerk continuity permitted on one side only)
	Optimization target	In the drop-down list, select the optimization target: • Default optimization setting (setting under "Profile > Default optimization settings") • Not specified • Velocity (Cv) • Acceleration (Ca) • Jerk (Cj) • Minimum dynamic torque (Cmdyn)
Selection of motion rule		

Parameters		Description
	Motion rule	In the drop-down list, select the motion rule according to which optimization is to occur. • Line • Quadratic parabola • Sine • Polynomial • Inclined sine • Modified acceleration trapezoid • Modified sine • Harmonic combination • Sine line combination The selection is automatically limited to the motion rules that can be applied according to the motion task and the selected boundary conditions. Additional parameters are displayed depending on the selected motion rule.
	Parameter used	In the drop-down list, select the parameters to be included in the optimization: • Lambda • Maximum acceleration (Ca) • Maximum deceleration (Ca*) The selection is automatically limited to the parameters that can be applied according to the motion rule.
	Lambda	Configure the position of the point of inflection (lambda) for the transition in this field.
	Maximum acceleration (Ca)	Configure the maximum acceleration (Ca) for the transition in this field.
	Maximum deceleration (Ca*)	Configure the maximum deceleration (Ca*) for the transition in this field.
Characteristic values of the transition		The characteristic values of the transition that are relevant according to VDI 2143 are displayed in this area. The maximum value and the standardized value are displayed for the following characteristic values: • Velocity (Cv) • Acceleration (Ca) • Deceleration (Ca*) • Jerk (Cj) • Dynamic torque (Cmdyn)

Motion jobs according to VDI Guideline 2143

The VDI Guideline 2143 distinguishes between areas of usage and motion transitions:

- Areas of usage correspond to the sequences in a process, which means the inserted elements of the cam.
- Motion transitions are transitions between areas of usage that are not directly relevant to the process but must meet specific boundary conditions (e.g. velocity consistency).

The VDI Guideline 2143 defines the following areas of usage:

Area of usage	Designation	Properties
Dwell	R	Velocity = 0 Acceleration = 0
Constant velocity	G	Velocity ≠ 0 Acceleration = 0
Reverse	U	Velocity = 0 Acceleration ≠ 0
Motion	B	Velocity ≠ 0 Acceleration ≠ 0

The following figure shows an example of the areas of usage:

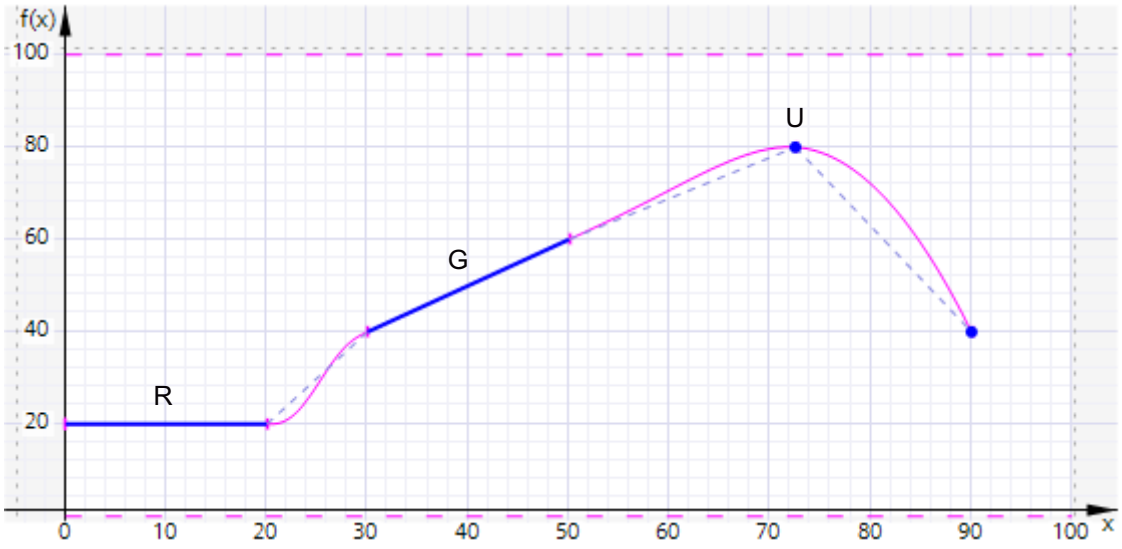

The figure below shows the possible transitions between the areas of usage:

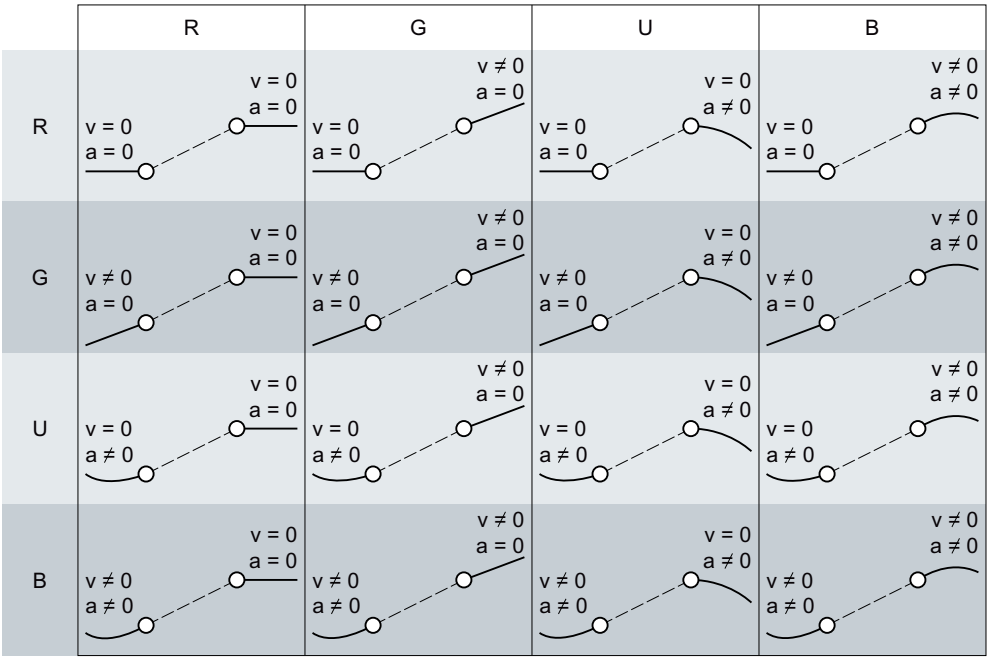

Commissioning (S7-1500)

Axis control panel (S7-1500)

Function and Structure of the Axis Control Panel V3.0 (S7-1500)

Description

You traverse individual axes with the axis control panel.

A user program is not necessary for the operation of the axis control panel. With the axis control panel, you take over master control for a technology object and control the movements of the axis.

WARNING

Uncontrolled axis motions

During operation with the axis control panel, the axis can execute uncontrolled motions (e.g. due to erroneous configuration of the drive, or of the technology object). Furthermore, any synchronized following axis is moved as well when moving a leading axis with the axis control panel.

Therefore, perform the following protective measures before operation with the axis control panel:

- Ensure that the EMERGENCY OFF switch is within the reach of the operator.
- Enable the hardware limit switches.
- Enable the software limit switches.
- Ensure that following error monitoring is enabled.
- Make sure that no following axis is coupled to the axis to be moved.

The axis control panel of the speed axis, positioning axis and synchronous axis can be found in the project tree under "Technology object > Commissioning".

The axis control panel is divided into the following areas:

- Master control
- Axis
- Operating mode
- Modify
- Axis status
- Current values

The following table lists the elements of the axis control panel:

Area	Element	Description
Master control		In the "Master control" area, you take over master control of the technology object, or return it to your user program.
	"Activate" button	With the "Activate" button, you set up an online connection to the CPU and take over master control for the selected technology object. • To take over master control, the technology object must be disabled in the user program. • Any synchronized following axis is moved as well when moving a leading axis with the axis control panel. • If the online connection to the CPU is lost during operation with the axis control panel, then after the sign of life monitoring has elapsed, the axis will be stopped with maximum deceleration. In this case, an error message is displayed ("ErrorID" = 16#8013) and the master control is passed back to the user program. • When you click the "Activate" button, a warning message is displayed. In the warning, you can adapt the sign-of-life monitoring (100 to 60000 ms). If the master control of the axis control panel is lost repeatedly without a direct error message, the online connection to the CPU may be impaired because the communication load is too high. In this case, the message "Error during commissioning" is entered in the message display log. Sign-of-life failure between controller and TIA Portal" is displayed. To eliminate this error, adapt the sign-of-life monitoring in the warning. • Until master control is returned, the user program has no influence on the functions of the technology object. Motion Control jobs from the user program to the technology object are rejected with error ("ErrorID" = 16#8012: axis control panel active). • When master control is taken over, the configuration of the technology object is adopted. Changes to the configuration of the technology object do not take effect until master control has been returned. For this reason, make any necessary changes before master control is taken over. • If master control has been taken over for the technology object, the axis control panel is blocked for access by another installation of the TIA Portal (Team Engineering as of CPU V1.5).
	"Deactivate" button	With the "Deactivate" button, you return master control to your user program.
Axis		In the "Axis" area, enable or disable the technology object.
	"Enable" button	With the "Enable" button, you enable the selected technology object
	"Disable" button	With the "Disable" button, you disable the selected technology object
Operating mode		Select the required mode in the "Operating mode" drop-down list.
Modify		The "Control" area displays the parameters for traversing with the axis control panel according to the selected operating mode.

Area	Element	Description
	Position	Position to which the axis is homed. (Homing and set home position modes only)
	Distance	Distance the axis is traversed. (Relative positioning mode only)
	Target position	Position to which the axis is traversed. (Absolute positioning mode only)
	Velocity/ Velocity setpoint	Velocity or speed at which the axis is traversed. Default: 10% of the default value (Velocity/speed setpoint, Jog and Positioning modes only)
	Acceleration	Acceleration with which the axis is traversed. Default: 10% of the default value
	Deceleration	Deceleration with which the axis is traversed. Default: 10% of the default value
	Jerk	Jerk with which the axis is traversed. Default: 100% of default value
	"Start" button	With the "Start" button, you start a job according to the selected operating mode.
	"Forward" button	With the "Forward" button, you start a motion in the positive direction according to the selected operating mode.
	"Backward" button	With the "Backward" button, you start a motion in the negative direction according to the selected operating mode.
	"Stop" button	With the "Stop" button, you cancel a job or stop the axis.
Axis status		The "Axis status" area displays the status of the axis and the status of the drive.
	Drive ready	Drive is ready to execute setpoints.
	Error	An error occurred at the technology object.
	Enabled	The technology object has been enabled. The axis can be moved with motion jobs.
	Homed	The technology object is homed.
	More	The "More" link takes you to the window "Technology object > Diagnostics > Status and error bits".
	Active errors	The error that occurred most recently is displayed in the "Active errors" text field.
	"Confirm" button	With the "Confirm" button, you acknowledge pending errors.
Current values		The "Current values" area shows the actual values of the axis.
	Position	Actual position of the axis
	Velocity	Actual velocity of the axis

Note

No transfer of the parameters

The configured parameter values are discarded when master control is returned. Transfer the values as needed into your configuration.

If you have changed configuration values with the axis control panel during operation, they have no effect on the operation of the axis control panel.

Operating mode

The following table shows the operating modes of the axis control panel:

Operating mode	Description
Homing	This function corresponds to active homing. The parameters for homing (Page 1670) must be configured. Homing is not possible with an absolute encoder. The technology object is not referenced when this mode is used with an absolute encoder.
Set home position	This function corresponds to direct homing (absolute). With the "Start" button, you set the actual position to the value specified in "Position" and the "Homed" status is set.
Jog	Motion commands occur by means of jogging. With the "Forward" or "Backward" button you start motion in the positive or negative direction. The motion runs for as long as you hold down the left mouse button.
Velocity specification/ speed setpoint	The axis is moved at the specified velocity or speed until you stop the movement. The motion commands are performed according to the setpoints assigned under "Controller".
Positioning relative	The positioning is executed as a controlled, relative traversing motion according to the defaults assigned under "Controller".
Positioning absolute	The positioning is executed as a controlled, absolute traversing motion according to the defaults assigned under "Controller". If you have enabled the "Modulo" setting of the technology object, the buttons "Forward" and "Backward" are shown in the "Controller" area. The axis is positioned within the modulo range. Position settings outside the modular range are recalculated to the modulo range. If you have not enabled the "Modulo" setting of the technology object, only the "Start" button is shown in the "Controller" area. You can directly approach the entered position.

Optimization (S7-1500)

Function and Structure of Optimization V3.0 (S7-1500)

Description

The "Optimization" function supports you in determining the optimal precontrol and gain (Kv factor) for the closed loop position control of the axis. The axis velocity profile is recorded by means of the Trace function for this purpose for the duration of a configurable positioning movement. You can then evaluate the recording, and adjust the precontrol and gain accordingly.

The "Optimization" function for the positioning axis and synchronous axis technology objects can be found in the project tree under "Technology object > Commissioning".

The "Tuning" dialog is divided into the following areas:

- Master control
- Axis
- Measurement configuration
- Optimize position controller

- Run measurement
- Trace

The following table lists the elements of optimization:

Area	Element	Description
Master control		In the "Master control" area, you take over master control of the technology object, or return it to your user program.
	"Activate" button	With the "Activate" button, you set up an online connection to the CPU and take over master control for the selected technology object. • To take over master control, the technology object must be disabled in the user program. • Any synchronized following axis is moved as well when moving a leading axis with the axis control panel. • If the online connection to the CPU is lost during operation with the axis control panel, then after the sign of life monitoring has elapsed, the axis will be stopped with maximum deceleration. In this case, an error message is displayed ("ErrorID" = 16#8013) and the master control is passed back to the user program. • When you click the "Activate" button, a warning message is displayed. You can adapt the sign-of-life monitoring here (100 to 60000 ms). If the master control of the axis control panel is lost repeatedly without a direct error message, the online connection to the CPU may be impaired because the communication load is too high. In this case, the message "Error during commissioning" is entered in the message display log. Sign-of-life failure between controller and TIA Portal" is displayed. To eliminate this error, adapt the sign-of-life monitoring in the warning. • Until master control is returned, the user program has no influence on the functions of the technology object. Motion Control jobs from the user program to the technology object are rejected with error ("ErrorID" = 16#8012: axis control panel active). • When master control is taken over, the configuration of the technology object is adopted. Changes to the configuration of the technology object do not take effect until master control has been returned. For this reason, make any necessary changes before master control is taken over. • If master control has been taken over for the technology object, the axis control panel is blocked for access by another installation of the TIA Portal (Team Engineering as of CPU V1.5).
	"Deactivate" button	With the "Deactivate" button, you return master control to your user program.
Axis		In the "Axis" area, enable or disable the technology object.
	"Enable" button	With the "Enable" button, you enable the selected technology object
	"Disable" button	With the "Disable" button, you disable the selected technology object

Area	Element	Description
Measurement configuration		In the "Measurement configuration" area, you configure the settings for a test step.
	Distance	Travel distance for a test step
	Measurement duration	Time for a test step
	"Adapt dynamic values" check box	With the "Adapt dynamic values" check box, you enable adaptation of the acceleration and the maximum velocity.
	Acceleration	Presetting of the acceleration for a test step
	Maximum velocity	Presetting of the maximum velocity for a test step
Optimize position controller		In the "Optimize position controller" area, you make the settings for optimization of the controller characteristics. You open a list of values using the symbol next to the field. The list of values contains the following values of the respective parameter: • Online actual value • Online start value • Project start value Enter the new value in the text field for the actual value. The new value is applied by clicking the "Forward" or "Backward" button in the "Run measurement" area.
	Precontrol	Current percentage velocity precontrol of the position controller
	Speed control loop substitute time	Current speed control loop substitute time With velocity precontrol, a simplified speed control loop model can be generated using the speed control loop substitute time. This prevents overmodulation by the position controller of the manipulated variable for velocity during the acceleration and deceleration phases. To accomplish this, the position setpoint of the position controller is delayed by the amount of the speed control loop substitute time in relation to the velocity precontrol.
	Gain	Current gain of the position controller (Kv)
Run measurement		In the "Run measurement" area, you perform the test steps.
	"Forward" button	With the "Forward" button, you start a test step for optimization in the positive direction.
	"Backward" button	With the "Backward" button, you start a test step for optimization in the negative direction.
	"Stop" button	You stop a test step with the "Stop" button.
Trace		The Trace function is displayed in the lower area of the "Tuning" dialog. With each test step, a Trace recording of the required parameters is automatically started and displayed after completion of the test step. After master control has been returned, the Trace recording is deleted. You can find a full description of the trace function in the section on using the trace and logic analyzer function.

Note

No transfer of the parameters

The configured parameter values are discarded after master control is returned.

Transfer the values as needed into your configuration. You can apply the values for the gain, precontrol and speed control loop substitute time in your configuration using the "Project start value" value.

Diagnostics (S7-1500)

Speed-controlled axis technology object (S7-1500)

Status and Error Bits V3.0 (S7-1500)

Description

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the status and error messages for the technology object. The Diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Axis status

The following table shows the possible axis status values:

Status	Description
Simulation active	The axis is simulated in the CPU. Setpoints are not output to the drive.
Enabled	The technology object has been enabled. The axis can be moved with motion jobs. (<TO>.StatusWord.X0 (Enable))
Error	An error occurred at the technology object. Detailed information about the error is available in the "Error" area, and in the <TO>.ErrorDetail.Number and <TO>.ErrorDetail.Reaction tags of the technology object. (<TO>.StatusWord.X1 (Error))
Restart active	The technology object is being reinitialized. (<TO>.StatusWord.X2 (RestartActive))
Axis control panel active	The axis control panel is active. The axis control panel has master control over the technology object. The axis cannot be controlled from the user program. (<TO>.StatusWord.X4 (ControlPanelActive))
Drive ready	Drive is ready to execute setpoints. (<TO>.StatusDrive.InOperation)
Restart required	Data relevant for the restart has been changed. The changes will not be applied until the technology object is restarted. (<TO>.StatusWord.X3 (OnlineStartValuesChanged))

Motion status

The following table shows the possible axis motion status values:

Status	Description
Done (no job running)	No motion job is running for the technology object. (<TO>.StatusWord.X6 (Done))
Jog	The axis is being moved with a job for jog mode of Motion Control instruction "MC_MoveJog" or from the axis control panel. (<TO>.StatusWord.X9 (JogCommand))
Speed setpoint	The axis is traversed with a job with speed setpoint of the Motion Control instruction "MC_MoveVelocity" or using the axis control panel. (<TO>.StatusWord.X10 (VelocityCommand))
Constant speed	The axis is moved with constant speed or is stationary. (<TO>.StatusWord.X12 (ConstantVelocity))
Accelerating	Axis is being accelerated. (<TO>.StatusWord.X13 (Accelerating))
Decelerating	The axis is being decelerated. (<TO>.StatusWord.X14 (Decelerating))
Torque limit active	The configured threshold for the force/torque affects the axis. (<TO>.StatusWord.X27 (InLimitation))

Warnings

The following table shows the possible warnings:

Warning	Description
Configuration	One or several configuration parameters are adjusted internally at a certain time. (<TO>.WarningWord.X1 (ConfigWarning))
Job rejected	Command cannot be executed. A Motion Control instruction cannot be executed because necessary requirements have not been met. (<TO>.WarningWord.X3 (CommandNotAccepted))
Dynamic limitation	Dynamic values are restricted to the dynamic limits. (<TO>.WarningWord.X6 (DynamicError))

Error

The following table shows the possible errors:

Error	Description
System	A system-internal error has occurred. (<TO>.ErrorWord.X0 (SystemFault))
Configuration	Configuration error One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.X1 (ConfigFault))
User program	Error in user program at a Motion Control instruction or its use. (<TO>.ErrorWord.X2 (UserFault))
Drive	Error in the drive. (<TO>.ErrorWord.X4 (DriveFault))
Data exchange	Communication with a connected device is faulty. (<TO>.ErrorWord.X7 (CommunicationFault))
I/O	Error accessing a logical address. (<TO>.ErrorWord.X13 (PeripheralError))
Job rejected	Command cannot be executed. A Motion Control instruction cannot be executed because necessary requirements have not been met (e.g. technology object not homed). (<TO>.ErrorWord.X3 (CommandNotAccepted))
Dynamic limitation	Dynamic values are restricted to the dynamic limits. (<TO>.ErrorWord.X6 (DynamicError))

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Motion Status V3.0 (S7-1500)

Description

You use the "Technology object > Diagnostics > Motion status" diagnostic function in the TIA Portal to monitor the motion status of the axis. The Diagnostics function is available in online operation.

"Current values" area

The following table shows the meaning of the status data:

Status	Description
Speed setpoint	Speed setpoint of the axis (<TO>.Velocity)
Speed override	Speed setpoint correction as percentage The speed setpoint specified in motion control instructions or set by the axis control panel are superimposed with an override signal and corrected as a percentage. Valid speed correction values range from 0.0 % to 200.0 %. (<TO>.Override.Velocity)

"Dynamic limits" area

This area displays the limit values for the dynamic parameters.

The following table shows the meaning of the status data:

Status	Description
Speed	Configured maximum speed (<TO>.DynamicLimits.MaxVelocity)
Acceleration	Configured maximum acceleration (<TO>.DynamicLimits.MaxAcceleration)
Deceleration	Configured maximum deceleration (<TO>.DynamicLimits.MaxDeceleration)

Positioning axis/synchronous axis technology object (S7-1500)

Status and Error Bits V3.0 (S7-1500)

Description

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the status and error messages for the technology object. The Diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Axis status

The following table shows the possible axis status values:

Status	Description
Simulation active	The axis is simulated in the CPU. Setpoints are not output to the drive. (<TO>.StatusWord.X25 (AxisSimulation))
Enabled	The technology object has been enabled. The axis can be moved with motion jobs. (<TO>.StatusWord.X0 (Enable))
Position-controlled mode	The axis is in position-controlled mode. (Inversion of <TO>.StatusWord.X28 (NonPositionControlled))
Homed	The technology object is homed. The relationship between the position in the technology object and the mechanical position was successfully created. (<TO>.StatusWord.X5 (HomingDone))
Error	An error occurred at the technology object. Detailed information about the error is available in the "Error" area, and in the <TO>.ErrorDetail.Number and <TO>.ErrorDetail.Reaction tags of the technology object. (<TO>.StatusWord.X1 (Error))
Restart active	The technology object is being reinitialized. (<TO>.StatusWord.X2 (RestartActive))
Axis control panel active	The axis control panel is active. The axis control panel has master control over the technology object. The axis cannot be controlled from the user program. (<TO>.StatusWord.X4 (ControlPanelActive))
Drive ready	Drive is ready to execute setpoints. (<TO>.StatusDrive.InOperation)
Encoder values valid	The actual encoder values are valid. (<TO>.StatusSensor[n].State)
Restart required	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object. (<TO>.StatusWord.X3 (OnlineStartValuesChanged))

Status limit switch

The following table shows the possibilities for enabling the software and hardware limit switches:

Status	Description
Negative SW limit switch approached.	The negative software limit switch was reached. (<TO>.StatusWord.X15 (SWLimitMinActive))
Positive SW limit switch approached.	The positive software limit switch was reached. (<TO>.StatusWord.X16 (SWLimitMaxActive))
Negative HW limit switch approached.	The negative hardware limit switch has been approached or overtraveled. (<TO>.StatusWord.X17 (HWLimitMinActive))
Positive HW limit switch approached.	The positive hardware limit switch has been approached or overtraveled. (<TO>.StatusWord.X18 (HWLimitMaxActive))

Motion status

The following table shows the possible axis motion status values:

Status	Description
Done (no job running)	No job active at technology object. (<TO>.StatusWord.X6 (Done))
Homing job	The technology object executes a homing job of the Motion Control instruction "MC_Home" or from the axis control panel. (<TO>.StatusWord.X11 (HomingCommand))
Jog	The axis is being moved with a command for jog mode of Motion Control instruction "MC_Move-Jog". (<TO>.StatusWord.X9 (JogCommand))
Velocity specification	The axis is traversed with a job with velocity specification of the Motion Control instruction "MC_MoveVelocity" or from the axis control panel. (<TO>.StatusWord.X10 (VelocityCommand))
Positioning job	The axis is traversed with a positioning job of Motion Control instruction "MC_MoveAbsolute" or "MC_MoveRelative" or from the axis control panel. (<TO>.StatusWord.X8 (PositioningCommand))
Constant velocity	The axis is moved with constant velocity or is stationary. (<TO>.StatusWord.X12 (ConstantVelocity))
Standstill	The axis is in standstill. (<TO>.StatusWord.X7 (StandStill))
Accelerating	Axis is being accelerated. (<TO>.StatusWord.X13 (Accelerating))
Decelerating	The axis is being decelerated. (<TO>.StatusWord.X14 (Decelerating))
Torque limit active	The configured threshold for the force/torque affects the axis. (<TO>.StatusWord.X27 (InLimitation))
Synchronization	Synchronous axis only The axis is synchronized to the leading value of a leading axis. (<TO>.StatusWord.X21 (Synchronizing))
Synchronous operation	Synchronous axis only The axis is synchronized and moves synchronously to the leading axis. (<TO>.StatusWord.X22 (Synchronous))

Warnings

The following table shows the possible warnings:

Warning	Description
Configuration	One or several configuration parameters are adjusted internally at a certain time. (<TO>.WarningWord.X1 (ConfigWarning))
Job rejected	Command cannot be executed. A Motion Control instruction cannot be executed because necessary requirements have not been met. (<TO>.WarningWord.X3 (CommandNotAccepted))

Warning	Description
Dynamic limitation	Dynamic values are restricted to the dynamic limits. (<TO>.WarningWord.X6 (DynamicError))
Synchronization	Synchronous axis only Error during synchronous operation. The leading axis specified for the corresponding Motion Control instruction was not configured as a possible leading axis. (<TO>.WarningWord.X14 (SynchronousError))

Error

The following table shows the possible errors:

Error	Description
System	A system-internal error has occurred. (<TO>.ErrorWord.X0 (SystemFault))
Configuration	Configuration error One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.X1 (ConfigFault))
User program	Error in user program at a Motion Control instruction or its use. (<TO>.ErrorWord.X2 (UserFault))
Drive	Error in the drive. (<TO>.ErrorWord.X4 (DriveFault))
Encoder	Error in encoder system. (<TO>.ErrorWord.X5 (SensorFault))
Data exchange	Communication with a connected device is faulty. (<TO>.ErrorWord.X7 (CommunicationFault))
I/O	Error accessing a logical address. (<TO>.ErrorWord.X13 (PeripheralError))
Job rejected	Command cannot be executed. A Motion Control instruction cannot be executed because necessary requirements have not been met (e.g. technology object not homed). (<TO>.ErrorWord.X3 (CommandNotAccepted))
Homing	Error during homing process. (<TO>.ErrorWord.X10 HomingFault))
Positioning	The positioning axis was not positioned correctly at the end of a positioning motion. (<TO>.ErrorWord.X12 (PositioningFault))
Dynamic limitation	Dynamic values are restricted to the dynamic limits. (<TO>.ErrorWord.X6 (DynamicError))
Following error	The maximum permitted following error has been exceeded. (<TO>.ErrorWord.X11 (FollowingErrorFault))
SW limit switch	A software limit switch has been reached. (<TO>.ErrorWord.X8 (SwLimit))

Error	Description
HW limit switch	A hardware limit switch has been reached or overtraveled. (<TO>.ErrorWord.X9 (HWLimit))
Synchronization	Synchronous axis only Error during synchronous operation. The leading axis specified for the corresponding Motion Control instruction was not configured as a possible leading axis. (<TO>.ErrorWord.X14 (SynchronousError))

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Motion Status V3.0 (S7-1500)

Description

You use the "Technology object > Diagnostics > Motion status" diagnostic function in the TIA Portal to monitor the motion status of the axis. The Diagnostics function is available in online operation.

"Current values" area

The following table shows the meaning of the status data:

Status	Description
Operative encoder	Operative encoder of the axis
Target position	Current target position of an active positioning job The target position value is only valid during execution of a positioning job. (<TO>.StatusPositioning.TargetPosition)
Actual position	Actual position of the axis If the technology object is not homed, then the value is displayed relative to the position that existed when the technology object was enabled. (<TO>.ActualPosition)
Actual velocity	Actual velocity of the axis (<TO>.ActualVelocity)
Velocity override	Percentage correction of the velocity specification The velocity setpoint specified in Motion Control instructions or set by the axis control panel is superimposed with an override signal and corrected as a percentage. Valid velocity correction values range from 0.0 % to 200.0 %. (<TO>.Override.Velocity)

"Dynamic limits" area

This area displays the limit values for the dynamic parameters.

The following table shows the meaning of the status data:

Status	Description
Velocity	Configured maximum velocity (<TO>.DynamicLimits.MaxVelocity)
Acceleration	Configured maximum acceleration (<TO>.DynamicLimits.MaxAcceleration)
Deceleration	Configured maximum deceleration (<TO>.DynamicLimits.MaxDeceleration)

6.1.2.10 Appendix (S7-1500)

Tags of the speed axis technology object (S7-1500)

Legend (S7-1500)

Tag	Name of the tag	
Data type	Data type of the tag	
Values	Value range of the tag - minimum value to maximum value If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".	
W	Effectiveness of changes in the technology data block	
	DIR	Direct: Values are changed directly and take effect at the start of the next MC-Servo [OB91].
	CAL	At call of Motion Control instruction: Values are changed directly and take effect at the start of the next MC-Servo [OB91] after the call of the corresponding Motion Control instruction in the user program.
	RES	Restart: Changes to the start value in the load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed during runtime of the user program.
Description	Description of the tag	

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Actual values and setpoints (speed axis) (S7-1500)

The following tags indicate the setpoint and actual values of the technology object.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description
Velocity	LREAL	-	RON	Velocity setpoint/speed setpoint
ActualSpeed	LREAL	-	RON	Actual speed of the motor (with analog setpoint = 0.0)
Acceleration	LREAL	-	RON	Setpoint acceleration

"Simulation" tag (speed axis) (S7-1500)

The tag structure "<TO>.Simulation.<tag name>" contains the configuration of the simulation mode. In simulation mode, you can simulate axes without a real drive in the CPU.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description	
Simulation.	STRUCT				
Mode	UDINT	-	RES[1)]	Simulation mode	
				0	No simulation, normal operation
				1	Simulation mode

1) Technology version V2.0: RON

"VirtualAxis" tag (speed axis) (S7-1500)

The tag structure "<TO>.VirtualAxis.<tag name>" contains the configuration of the simulation mode. In simulation mode, you can simulate axes without a real drive in the CPU.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description	
VirtualAxis.	STRUCT				
Mode	UDINT	-	RON	Virtual axis	
				0	No virtual axis
				1	Axis is always and exclusively operated as virtual axis

"Actor" tag (speed axis) (S7-1500)

The tag structure "<TO>.Actor.<tag name>" contains the controller-side configuration of the drive.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description
Actor.	STRUCT			

Tag	Data type	Values	W	Description	
Type	DINT	-	RON	Drive connection	
				0	Analog output
				1	PROFIdrive telegram
InverseDirection	BOOL	-	RES	Inversion of the setpoint	
				FALSE	No
				TRUE	Yes
DataAdaption	DINT	-	RES	Automatic transfer of the drive values reference speed, maximum speed and reference torque in the device	
				0	No automatic transfer, manual configuration of values
				1	Automatic transfer of values configured in the drive to the configuration of the technology object
Efficiency	LREAL	0.0 to 1.0	RES	Efficiency of gear	
Interface.	STRUCT				
AddressIn	VREF	0 to 65535	RON	Input address for the PROFIdrive telegram	
AddressOut	VREF	0 to 65535	RON	Output address for the PROFIdrive telegram or the analog setpoint	
EnableDriveOutput	BOOL	-	RES	"Enable output" for analog drives	
				FALSE	Disabled
				TRUE	Enabled
EnableDriveOutputAddress	VREF	0 to 65535	RON	Address for the "Enable output" for analog setpoint	
DriveReadyInput	BOOL	-	RES	"Ready input" for analog drives The analog drive signals its readiness to receive speed setpoints.	
				FALSE	Disabled
				TRUE	Enabled
DriveReadyInputAddress	VREF	0 to 65535	RON	Address for the "Enable input" for analog setpoint	
EnableTorqueData	BOOL	-	RES	Torque data	
				FALSE	Disabled
				TRUE	Enabled
TorqueDataAddressIn	VREF	0 to 65535	RON	Input Address of the Telegram 750	
TorqueDataAddressOut	VREF	0 to 65535	RON	Output address of the telegram 750	
DriveParameter.					
ReferenceSpeed	LREAL	0.0 to 1.0E12	RES	Reference value (100%) for the speed setpoint (N-set) of the drive The speed setpoint is transferred in the PROFIdrive telegram as a normalized value from -200% to 200% of the "ReferenceSpeed". For setpoint specification via an analog output, the analog output can be operated in the range from -117% to 117%, provided the drive permits this.	

Tag	Data type	Values	W	Description
MaxSpeed	LREAL	0.0 to 1.0E12	RES	Maximum value for the speed setpoint of the drive (N-set) (PROFIdrive: MaxSpeed ≤ 2 × ReferenceSpeed Analog setpoint: MaxSpeed ≤ 1.17 × ReferenceSpeed)
ReferenceTorque	LREAL	0.0 to 1.0E12	RES	Reference torque of drive (p2003). Valid for standard motor setting.

"TorqueLimiting" tag (speed axis) (S7-1500)

The tag structure "<TO>.TorqueLimiting.<tag name>" contains the configuration of the torque limiting.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description	
TorqueLimiting.	STRUCT				
LimitBase	DINT	-	RES	Torque limiting	
				0	Motor side
				1	Load side
PositionBasedMonitor-ings	DINT	-	RES	Positioning and following error monitoring	
				0	Monitoring deactivated
				1	Monitoring activated
LimitDefaults.	STRUCT				
Torque	LREAL	0.0 to 1.0E12	CAL	Limiting torque	
Force	LREAL	0.0 to 1.0E12	CAL	Limiting force	

"LoadGear" tag (speed axis) (S7-1500)

The tag structure "<TO>.LoadGear.<tag name>" contains the configuration of the load gear.

Tags

Legend (Page 2011)

Tag	Data type	Value range	W	Description
LoadGear.	STRUCT			
Numerator	UDINT	1 to 4294967295	RES	Load gear numerator
Denominator	UDINT	1 to 4294967295	RES	Load gear denominator

"Units" tag (speed axis) (S7-1500)

The tag structure "<TO>.Units.<tag name>" shows the set technological units.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description	
Units.	STRUCT				
VelocityUnit	UDINT	-	RON	Unit for velocity	
				1082	1/s
				1083	1/min
				1528	1/h
TimeUnit	UDINT	-	RON	Unit for time	
				1054	s
TorqueUnit	UDINT	-	RON	Unit for torque	
				1126	Nm
				1128	kNm
				1529	lbf in (pound-force-inch)
				1530	lbf ft
				1531	ozf in (ounce-force-inch)
				1532	ozf ft
				1533	pdl in (poundal-inch)
				1534	pdl ft
ForceUnit	UDINT	-	RON	Unit for force	
				1120	N
				1122	kN
				1094	lbf (pound-force)
				1093	ozf (ounce-force)
				1535	pdl (poundals)

"DynamicLimits" tag (speed axis) (S7-1500)

The tag structure "<TO>.DynamicLimits.<tag name>" contains the configuration of the dynamic limits. During Motion Control, no dynamic values greater than the dynamic limits are permitted. If you have specified greater values in a Motion Control instruction, then motion is performed using the dynamic limits, and a warning is indicated (alarm 501 to 503 - Dynamic values were limited).

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description
DynamicLimits.	STRUCT			
MaxVelocity	LREAL	0.0 to 1.0E12	RES	Maximum permissible velocity of the axis
MaxAcceleration	LREAL	0.0 to 1.0E12	DIR	Maximum permissible acceleration of the axis
MaxDeceleration	LREAL	0.0 to 1.0E12	DIR	Maximum permissible deceleration of the axis
MaxJerk	LREAL	0.0 to 1.0E12	DIR	Maximum permissible jerk on the axis

"DynamicDefaults" tag (speed axis) (S7-1500)

The tag structure "<TO>.DynamicDefaults.<tag name>" contains the configuration of the dynamic defaults. These settings will be used when you specify a dynamic value less than 0.0 in a Motion Control instruction (exceptions: "MC_MoveJog.Velocity", "MC_MoveVelocity.Velocity"). Changes to the default dynamic values will be applied at the next positive edge at the "Execute" parameter of a Motion Control instruction.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description
DynamicDefaults.	STRUCT			
Velocity	LREAL	0.0 to 1.0E12	CAL	Default velocity
Acceleration	LREAL	0.0 to 1.0E12	CAL	Default acceleration
Deceleration	LREAL	0.0 to 1.0E12	CAL	Default deceleration
Jerk	LREAL	0.0 to 1.0E12	CAL	Default jerk
EmergencyDecelera-tion	LREAL	0.0 to 1.0E12	DIR	Emergency stop deceleration

"Override" tag (speed axis) (S7-1500)

The tag structure "<TO>.Override.<tag name>" contains the configuration of override parameters. The override parameters are used to apply a correction percentage to default values. An override change takes effect immediately, and is performed with the dynamic settings in effect in the Motion Control instruction.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description
Override.	STRUCT			
Velocity	LREAL	0.0 to 200.0%	DIR	Velocity or speed override Percentage correction of the velocity/speed

"StatusDrive" tag (speed axis) (S7-1500)

The tag structure "<TO>.StatusDrive.<tag name>" indicates the status of the drive.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description	
StatusDrive.	STRUCT				
InOperation	BOOL	-	RON	Operational status of the drive	
				FALSE	Drive not ready. Setpoints will not be executed.
				TRUE	Drive ready. Setpoints can be executed.
CommunicationOK	BOOL	-	RON	Cyclic BUS communication between controller and drive	
				FALSE	Not established
				TRUE	Established
Error	BOOL	-	RON	FALSE	No drive error
				TRUE	Drive error
AdaptionState	DINT	-	RON	Status of automatic data transfer of drive parameters	
				0	"NOT_ADAPTED" Data not transferred
				1	"IN_ADAPTION" Data transfer in progress
				2	"ADAPTED" Data transfer complete
				3	"NOT_APPLICABLE" Data transfer not selected, not possible
				4	"ADAPTION_ERROR" Error during data transfer

"StatusTorqueData" tag (speed axis) (S7-1500)

The tag structure "<TO>.StatusTorqueData.<tag name>" indicates the status of the torque.

Tags

Legend (Page 2011)

Tag	Data type	Value range	W	Description	
StatusTorqueData.	STRUCT				
CommandAdditiveTorqueActive	DINT	-	RON	Additive torque setpoint function	
				0	Disabled
				1	Enabled
CommandTorqueRangeActive	DINT	-	RON	Torque range above high and low limit of the torque function	
				0	Disabled
				1	Enabled
ActualTorque	LREAL	-1.0E12 to 1.0E12	RON	Actual torque of the axis in the technological unit of the technology object for torque	

"StatusMotionIn" tag (speed axis) (S7-1500)

The tag structure "<TO>.StatusMotionIn.<tag name>" indicates the motion status.

Tags

Legend (Page 2011)

Tag	Data type	Value range	W	Description	
StatusMotionIn.	STRUCT				
FunctionState	DINT	-	RON	0	No "MotionIn" function active
				1	"MotionInVelocity" function active

"StatusWord" tag (speed axis) (S7-1500)

The "<TO>.StatusWord" tag contains the status information of the technology object.

Information on the evaluation of the individual bits (e.g. bit 0 "Enable") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2011)

Tag	Data type	Values	W	Description
StatusWord	DWORD	-	RON	Status information of the technology object

Tag	Data type	Values	W	Description	
Bit 0	-	-	-	"Enable" Enable status	
				0	Technology object disabled
				1	Technology object enabled
Bit 1	-	-	-	"Error"	
				0	No error present
				1	Error present
Bit 2	-	-	-	"RestartActive"	
				0	No restart active
				1	Restart active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged"	
				0	Restart tags unchanged
				1	Change to Restart tags. For the changes to be applied, the technology object must be reinitialized.
Bit 4	-	-	-	"ControlPanelActive" Axis control panel	
				0	Disabled
				1	Enabled
Bit 5	-	-	-	Reserved	
Bit 6	-	-	-	"Done"	
				0	Motion job is active or axis control panel enabled
				1	No motion job in process and axis control panel disabled
Bit 7	-	-	-	Reserved	
Bit 8	-	-	-	Reserved	
Bit 9	-	-	-	"JogCommand"	
				0	No "MC_MoveJog" job active
				1	"MC_MoveJog" job active
Bit 10	-	-	-	"VelocityCommand"	
				0	No "MC_MoveVelocity" job active
				1	"MC_MoveVelocity" job active
Bit 11	-	-	-	Reserved	
Bit 12	-	-	-	"ConstantVelocity"	
				0	The axis is accelerated or decelerated.
				1	The setpoint velocity is reached. A constant velocity setpoint is output.
Bit 13	-	-	-	"Accelerating"	
				0	No acceleration process active
				1	Acceleration process active
Bit 14	-	-	-	"Decelerating"	
				0	No deceleration process active
				1	Deceleration process active

Tag	Data type	Values	W	Description	
Bit 15... Bit 24	-	-	-	Reserved	
Bit 25	-	-	-	"AxisSimulation"	
				0	No simulation
				1	Simulation is active
Bit 26	-	-	-	"TorqueLimitingCommand"	
				0	No "MC_TorqueLimiting" job is active
				1	"MC_TorqueLimiting" job is active
Bit 27	-	-	-	"InLimitation"	
				0	Drive is not operating at the torque limit
				1	Drive is operating at the torque limit
Bit 28... Bit 31	-	-	-	Reserved	

"ErrorWord" tag (speed axis) (S7-1500)

The "<TO>.ErrorWord" tag indicates technology object errors (technology alarms).

Information on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2011)

Tag	Data type	Values	W	Description
ErrorWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" System error
Bit 1	-	-	-	"ConfigFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4	-	-	-	"DriveFault" Error in drive
Bit 5	-	-	-	Reserved
Bit 6	-	-	-	"DynamicError" Specified dynamic values are limited to permissible values.
Bit 7	-	-	-	"CommunicationFault" Communication error Missing or faulty communication.
Bit 8... Bit 12	-	-	-	Reserved
Bit 13	-	-	-	"PeripheralError" Error accessing a logical address
Bit 14	-	-	-	Reserved
Bit 15	-	-	-	"AdaptionError" Error during data transfer
Bit 16 ... Bit 31	-	-	-	Reserved

"ErrorDetail" tag (speed axis) (S7-1500)

The tag structure "<TO>.ErrorDetail.<tag name>" contains the alarm number and the effective local alarm reaction for the technology alarm that is currently pending on the technology object.

You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description	
ErrorDetail.	STRUCT				
Number	UDINT	-	RON	Alarm number	
Reaction	DINT	0 to 5	RON	Effective alarm reaction	
				0	No reaction
				1	Stop with current dynamic values
				2	Stop with maximum dynamic values
				3	Stop with emergency stop ramp
				4	Remove enable
				5	Track setpoints

"WarningWord" tag (speed axis) (S7-1500)

The "<TO>.WarningWord" tag indicates pending warnings at the technology object.

Information on the evaluation of the individual bits (e.g. bit 13 "PeripheralWarning") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2011)

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemWarning" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigWarning" Configuration error One or several configuration parameters are adjusted internally.
Bit 2	-	-	-	"UserWarning" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4	-	-	-	"DriveWarning" Error in drive
Bit 5	-	-	-	Reserved
Bit 6	-	-	-	"DynamicWarning" Specified dynamic values are limited to permissible values.
Bit 7	-	-	-	"CommunicationWarning" Communication error Missing or faulty communication.
Bit 8... Bit 12	-	-	-	Reserved
Bit 13	-	-	-	"PeripheralWarning" Error accessing a logical address
Bit 14	-	-	-	Reserved
Bit 15	-	-	-	"AdaptionWarning" Error in automatic data transfer
Bit 16... Bit 31	-	-	-	Reserved

"ControlPanel" tag (speed axis) (S7-1500)

The tag structure "<TO>.ControlPanel.<tag name>" contains no user-relevant data. This tag structure is internally used.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description
ControlPanel.	STRUCT			
Input.	STRUCT			
TimeOut	LREAL	100 to 60000	DIR	-
EsLifeSign	UDINT	-	DIR	-
Command.	ARRAY [1..2] OF STRUCT			
ReqCounter	UDINT	-	DIR	-
Type	UDINT	-	DIR	-
Position	LREAL	-	DIR	-
Velocity	LREAL	-	DIR	-
Acceleration	LREAL	-	DIR	-
Deceleration	LREAL	-	DIR	-
Jerk	LREAL	-	DIR	-
Param	LREAL	-	DIR	-
Output.	STRUCT			
RTLifeSign	UDINT	-	RON	-
Command.	ARRAY [1..2] OF STRUCT			
AckCounter	UDINT	-	RON	-
Error	BOOL	-	RON	-
ErrorID	UDINT	-	RON	-
Done	BOOL	-	RON	-
Aborted	BOOL	-	RON	-

"InternalToTrace" tag (speed axis) (S7-1500)

The tag structure "<TO>.InternalToTrace.<tag name>" contains no user-relevant data. This tag structure is internally used.

Tags

Legend (Page 2011)

Tag	Data type	Values	W	Description
InternalToTrace.	ARRAY [1..4] OF STRUCT			
Id	DINT	-	DIR	-
Value	LREAL	-	DIR	-

Tags of the positioning axis/synchronous axis technology object (S7-1500)

Legend (S7-1500)

Tag	Name of the tag	
Data type	Data type of the tag	
Values	Value range of the tag - minimum value to maximum value (L - linear specification R - rotary specification) If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".	
W	Effectiveness of changes in the technology data block	
	DIR	Direct: Values are changed directly and take effect at the start of the next MC-Servo [OB91].
	CAL	At call of Motion Control instruction: Values are changed directly and take effect at the start of the next MC-Servo [OB91] after the call of the corresponding Motion Control instruction in the user program.
	RES	Restart: Changes to the start value in the load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed during runtime of the user program.
Description	Description of the tag	

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Actual values and setpoints (positioning axis/synchronous axis) (S7-1500)

The following tags indicate the setpoint and actual values of the technology object.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
Position	LREAL	-	RON	Position setpoint
Velocity	LREAL	-	RON	Velocity setpoint/speed setpoint
ActualPosition	LREAL	-	RON	Actual position
ActualVelocity	LREAL	-	RON	Actual velocity
ActualSpeed	LREAL	-	RON	Actual speed of the motor (with analog setpoint = 0.0)
Acceleration	LREAL	-	RON	Setpoint acceleration
ActualAcceleration	LREAL	-	RON	Actual acceleration
OperativeSensor	UDINT	1 to 4	RON	Operative encoder

"Simulation" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Simulation.<tag name>" contains the configuration of the simulation mode. In simulation mode, you can simulate axes without a real drive in the CPU.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
Simulation.	STRUCT				
Mode	UDINT	-	RES[1]	Simulation mode	
				0	No simulation, normal operation
				1	Simulation mode

1) Technology version V2.0: RON

"VirtualAxis" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.VirtualAxis.<tag name>" contains the configuration of the simulation mode. In simulation mode, you can simulate axes without a real drive in the CPU.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
VirtualAxis.	STRUCT				
Mode	UDINT	-	RON	Virtual axis	
				0	No virtual axis
				1	Axis is always and exclusively operated as virtual axis

"Actor" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Actor.<tag name>" contains the controller-side configuration of the drive.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
Actor.	STRUCT			

Tag	Data type	Values	W	Description	
Type	DINT	-	RON	Drive connection	
				0	Analog output
				1	PROFIdrive telegram
InverseDirection	BOOL	-	RES	Inversion of the setpoint	
				FALSE	No
				TRUE	Yes
DataAdaption	DINT	-	RES	Automatic transfer of the drive values reference speed, maximum speed and reference torque	
				0	No automatic transfer, manual configuration of values
				1	Automatic transfer of values configured in the drive to the configuration of the technology object
Efficiency	LREAL	0.0 to 1.0	RES	Efficiency of mechanics (gear and leadscrew)	
Interface.	STRUCT				
AddressIn	VREF	0 to 65535	RON	Input address for the PROFIdrive telegram	
AddressOut	VREF	0 to 65535	RON	Output address for the PROFIdrive telegram or the analog setpoint	
EnableDriveOutput	BOOL	-	RES	"Enable output" for analog drives	
				FALSE	Disabled
				TRUE	Enabled
EnableDriveOutputAddress	VREF	0 to 65535	RON	Address for the "Enable output" for analog setpoint	
DriveReadyInput	BOOL	-	RES	"Ready input" for analog drives The analog drive signals its readiness to receive speed setpoints.	
				FALSE	Disabled
				TRUE	Enabled
DriveReadyInputAddress	VREF	0 to 65535	RON	Address for the "Enable input" for analog setpoint	
EnableTorqueData	BOOL	-	RES	Torque data	
				FALSE	Disabled
				TRUE	Enabled
TorqueDataAddressIn	VREF	0 to 65535	RON	Input address of the supplemental telegram	
TorqueDataAddressOut	VREF	0 to 65535	RON	Output address of the supplemental telegram	
DriveParameter.					
ReferenceSpeed	LREAL	0.0 to 1.0E12	RES	Reference value (100%) for the speed setpoint (N-set) of the drive The speed setpoint is transferred in the PROFIdrive telegram as a normalized value from -200% to 200% of the "ReferenceSpeed". For setpoint specification via an analog output, the analog output can be operated in the range from -117% to 117%, provided the drive permits this.	

Tag	Data type	Values	W	Description
MaxSpeed	LREAL	0.0 to 1.0E12	RES	Maximum value for the speed setpoint of the drive (N-set) (PROFIdrive: MaxSpeed ≤ 2 × ReferenceSpeed Analog setpoint: MaxSpeed ≤ 1.17 × ReferenceSpeed)
ReferenceTorque	LREAL	0.0 to 1.0E12	RES	Reference value (100%) for the drive torque

See also

Evaluating the technology data block (Page 1920)

"TorqueLimiting" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.TorqueLimiting.<tag name>" contains the configuration of the torque limiting.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
TorqueLimiting.	STRUCT				
LimitBase	DINT	-	RES	Torque limiting	
				0	Motor side
				1	Load side
PositionBasedMonitor-ings	DINT	-	RES	Positioning and following error monitoring	
				0	Monitoring deactivated
				1	Monitoring activated
LimitDefaults.	STRUCT				
Torque	LREAL	0.0 to 1.0E12	CAL	Limiting torque	
Force	LREAL	0.0 to 1.0E12	CAL	Limiting force	

"Clamping" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Clamping.<tag name>" contains the configuration of the fixed stop detection.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
Clamping.	STRUCT			
FollowingErrorDevia-tion	LREAL	0.001 to 1.0E12	DIR	Value of the following error starting from which the fixed stop is detected.
PositionTolerance	LREAL	0.001 to 1.0E12	DIR	Position tolerance for the clamping monitoring

"Sensor[n]" tags (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Sensor[n].<tag name>" contains the controller-end configuration for the encoder, and the configuration for active and passive homing.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
Sensor[n].	ARRAY [1..4] OF STRUCT			

Tag	Data type	Values	W	Description	
Existent	BOOL	-	RON	Displaying created encoders	
Type	DINT	-	RON	Encoder type	
				0	"INCREMENTAL" (incremental)
				1	"ABSOLUTE" (absolute)
				2	"CYCLIC_ABSOLUTE" (cyclic absolute)
InverseDirection	BOOL	-	RES	Inversion of the actual value	
				FALSE	No
				TRUE	Yes
System	DINT	-	RES	Encoder system	
				0	"LINEAR" (linear encoder)
				1	ROTATORY (rotary encoder)
MountingMode	DINT	-	RES	Mounting type of encoder	
				0	On motor shaft
				1	On load side
				2	External measuring system
DataAdaption	DINT	-	RES	Automatic transfer of the drive values reference speed, maximum speed and reference torque in the device	
				0	No automatic transfer, manual configuration of values
				1	Automatic transfer of values configured in the drive to the configuration of the technology object
Interface.					
AddressIn	VREF	0 to 65535	RON	Input address for the PROFIdrive telegram	
AddressOut	VREF	0 to 65535	RON	Output address for the PROFIdrive telegram	
Number	UDINT	1 to 2	RON	Number of the encoder in the telegram	
Parameter.					
Resolution	LREAL	1.0E-12 to 1.0E12	RES	Resolution of a linear encoder (offset between two encoder pulses)	
StepsPerRevolution	UDINT	1 to 8388608	RES	Increments per rotary encoder revolution	
FineResolutionXist1	UDINT	0 to 31	RES	Number of bits for fine resolution Gx_XIST1 (cyclic actual encoder value)	
FineResolutionXist2	UDINT	0 to 31	RES	Number of bits for fine resolution Gx_XIST2 (absolute value of encoder)	
DeterminableRevolutions	UDINT	0 to 8388608	RES	Number of differentiable encoder revolutions for a multi-turn absolute encoder (For a single-turn absolute encoder = 1; for an incremental encoder = 0)	
DistancePerRevolution	LREAL	0.0 to 1.0E12	RES	Load distance per revolution of an externally mounted encoder	
ActiveHoming.	STRUCT				
Mode	DINT	-	RES	Homing mode	
				0	Use zero mark via PROFIdrive telegram
				1	Zero mark via PROFIdrive telegram and reference cam

Tag	Data type	Values	W	Description	
				2	Use homing mark via digital input
SideInput	BOOL	-	CAL	Side of the digital input for active homing	
				FALSE	Negative side
				TRUE	Positive side
Direction	DINT	-	CAL	Homing direction/approach direction on the homing mark	
				0	Positive homing direction
				1	Negative homing direction
DigitalInputAddress	VREF	0 to 65535	RON	Address of digital input	
HomePositionOffset	LREAL	-1.0E12 to 1.0E12	CAL	Home position offset	
SwitchLevel	BOOL	-	RES	Signal level that is present at the digital input when homing mark is approached	
				FALSE	Low level
				TRUE	High level
PassiveHoming.	STRUCT				
Mode	DINT	-	RES	Homing mode	
				0	Use zero mark via PROFIdrive telegram
				1	Zero mark via PROFIdrive telegram and reference cam
				2	Use homing mark via digital input
SideInput	BOOL	-	CAL	Side of the digital input for passive homing	
				FALSE	Negative side
				TRUE	Positive side
Direction	DINT	-	CAL	Homing direction/approach direction on the homing mark	
				0	Positive homing direction
				1	Negative homing direction
				2	Current homing direction
DigitalInputAddress	VREF	0 to 65535	RON	Address of digital input	
SwitchLevel	BOOL	-	RES	Signal level that is present at the digital input when homing mark is approached	
				FALSE	Low level
				TRUE	High level

See also

Evaluating the technology data block (Page 1920)

"Extrapolation" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Extrapolation.<tag name>" contains the configuration of the actual value extrapolation.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
Extrapolation.	STRUCT			
LeadingAxisDependentTime	LREAL	-	RON	Extrapolation time component (caused by leading axis) Results from the following times: • Time of actual value acquisition for the leading axis • Interpolator cycle clock • Time of actual position filter of leading axis (T1 + T2)
FollowingAxisDependentTime	LREAL	0.0 to 1.0E12	DIR	Extrapolation time component (caused by following axis) Results from the following times: • For a following axis with set velocity precontrol: – Communication cycle – Interpolator cycle clock – Speed control loop substitute time for the following axis – Output time of the setpoint for the following axis • For a following axis without velocity precontrol: – Communication cycle – Interpolator cycle clock – Position control loop equivalent time (1/Kv from "<TO>.PositionControl.Kv") – Output time of the setpoint for the following axis
PositionFilter.	STRUCT			
T1	LREAL	0.0 to 1.0E12	DIR	Position filter time constant T1
T2	LREAL	0.0 to 1.0E12	DIR	Position filter time constant T2
VelocityFilter.	STRUCT			
T1	LREAL	0.0 to 1.0E12	DIR	Velocity filter time constant T1
VelocityTolerance.	STRUCT			
Range	LREAL	0.0 to 1.0E12	DIR	Tolerance band width for velocity
Hysteresis.	STRUCT			
Value	LREAL	0.0 to 1.0E12	DIR	Hysteresis value

"LoadGear" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.LoadGear.<tag name>" contains the configuration of the load gear.

Tags

Legend (Page 2027)

Tag	Data type	Value range	W	Description
LoadGear.	STRUCT			
Numerator	UDINT	1 to 4294967295	RES	Load gear counter
Denominator	UDINT	1 to 4294967295	RES	Load gear denominator

"Properties" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Properties.<tag name>" contains the configuration of the type of axis or motion.

Tags

Legend (Page 2027)

Tag	Data type	Value range	W	Description	
Properties.	STRUCT				
MotionType	DINT	-	RON	Indication of axis type or motion type	
				0	Linear axis or motion
				1	Rotary axis or motion

"Units" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Units.<tag name>" shows the set technological units.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
Units.	STRUCT			

Tag	Data type	Values	W	Description	
LengthUnit	UDINT	-	RON	Unit for position	
				1010	m
				1013	mm
				1011	km
				1014	µm
				1015	nm
				1019	in
				1018	ft
				1021	mi
				1004	rad
				1005	°
VelocityUnit	UDINT	-	RON	Unit for velocity	
				1521	°/s
				1522	°/min
				1086	rad/s
				1523	rad/min
				1062	mm/s
				1061	m/s
				1524	mm/min
				1525	m/min
				1526	mm/h
				1063	m/h
				1527	km/min
				1064	km/h
				1066	in/s
				1069	in/min
				1067	ft/s
				1070	ft/min
				1075	mi/h
TimeUnit	UDINT	-	RON	Unit for time	
				1054	s
TorqueUnit	UDINT	-	RON	Unit for torque	
				1126	Nm
				1128	kNm
				1529	lbf in (pound-force-inch)
				1530	lbf ft
				1531	ozf in (ounce-force-inch)
				1532	ozf ft
				1533	pdl in (poundal-inch)
				1534	pdl ft
ForceUnit	UDINT	-	RON	Unit for force	
				1120	N
				1122	kN

Tag	Data type	Values	W	Description	
				1094	lbf (pound-force)
				1093	ozf (ounce-force)
				1535	pdl (poundals)

"Mechanics" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Mechanics.<tag name>" contains the configuration of the mechanics.

Tags

Legend (Page 2027)

Tag	Data type	Value range	W	Description
Mechanics.	STRUCT			
LeadScrew	LREAL	0.0 to 1.0E12	RES	Leadscrew pitch

"Modulo" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Modulo.<tag name>" contains the configuration of the modulo function.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
Modulo.	STRUCT				
Enable	BOOL	-	RES	FALSE	Modulo conversion disabled
				TRUE	Modulo conversion enabled
				When modulo conversion is enabled, a check is made for modulo length > 0.0	
Length	LREAL	0.001 to 1.0E12	RES	Modulo length	
StartValue	LREAL	-1.0E12 to 1.0E12	RES	Modulo start value	

See also

Evaluating the technology data block (Page 1920)

"DynamicLimits" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.DynamicLimits.<tag name>" contains the configuration of the dynamic limits. During Motion Control, no dynamic values greater than the dynamic limits are permitted. If you have specified greater values in a Motion Control instruction, then motion is performed using the dynamic limits, and a warning is indicated (alarm 501 to 503 - Dynamic values are limited).

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
DynamicLimits.	STRUCT			
MaxVelocity	LREAL	0.0 to 1.0E12	RES	Maximum permissible velocity of the axis
Velocity	LREAL	0.0 to 1.0E12	DIR	Current maximum velocity of the axis
MaxAcceleration	LREAL	0.0 to 1.0E12	DIR	Maximum permissible acceleration of the axis
MaxDeceleration	LREAL	0.0 to 1.0E12	DIR	Maximum permissible deceleration of the axis
MaxJerk	LREAL	0.0 to 1.0E12	DIR	Maximum permissible jerk on the axis

See also

Evaluating the technology data block (Page 1920)

"DynamicDefaults" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.DynamicDefaults.<tag name>" contains the configuration of the dynamic defaults. These settings will be used when you specify a dynamic value less than 0.0 in a Motion Control instruction (exceptions: "MC_MoveJog.Velocity", "MC_MoveVelocity.Velocity"). Changes to the default dynamic values will be applied at the next positive edge at the "Execute" parameter of a Motion Control instruction.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
DynamicDefaults.	STRUCT			
Velocity	LREAL	0.0 to 1.0E12	CAL	Default velocity
Acceleration	LREAL	0.0 to 1.0E12	CAL	Default acceleration
Deceleration	LREAL	0.0 to 1.0E12	CAL	Default deceleration
Jerk	LREAL	0.0 to 1.0E12	CAL	Default jerk
EmergencyDecelera-tion	LREAL	0.0 to 1.0E12	DIR	Emergency stop deceleration

See also

Evaluating the technology data block (Page 1920)

"PositionLimits_SW" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.PositionLimits_SW.<tag name>" contains the configuration of position monitoring with software limit switches. Software limit switches are used to limit the operating range of a positioning axis.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
PositionLimits_SW.	STRUCT				
Active	BOOL	-	DIR	FALSE	Monitoring deactivated
				TRUE	Monitoring enabled
MinPosition	LREAL	-1.0E12 to 1.0E12	DIR	Position of negative software limit switches	
MaxPosition	LREAL	-1.0E12 to 1.0E12	DIR	Position of positive software limit switches ("MaxPosition" > "MinPosition")	

See also

Evaluating the technology data block (Page 1920)

"PositionLimits_HW" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.PositionLimits_HW.<tag name>" contains the configuration of position monitoring with hardware limit switches. Hardware limit switches are used to limit the traversing range of a positioning axis.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
PositionLimits_HW.	STRUCT				
Active	BOOL	-	RES	FALSE	Monitoring deactivated
				TRUE	Monitoring enabled
				With "Active", both (negative and positive) hardware limit switches are activated or deactivated.	
MinSwitchLevel	BOOL	-	RES	Level selection for activation of the negative hardware limit switch	
				FALSE	Low level (Low active)
				TRUE	High level (high-enabled)
MinSwitchAddress	VREF	0 to 65535	RON	Address for the negative hardware limit switch	
MaxSwitchLevel	BOOL	-	RES	Level selection for activation of the positive hardware limit switch	
				FALSE	Low level (Low active)
				TRUE	High level (high-enabled)
MaxSwitchAddress	VREF	0 to 65535	RON	Address for the positive hardware limit switch	

See also

Evaluating the technology data block (Page 1920)

"Homing" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Homing.<tag name>" contains the configuration for homing the TO.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
Homing.	STRUCT				
AutoReversal	BOOL	-	RES	Reversal at the hardware limit switches	
				FALSE	No
				TRUE	Yes
ApproachDirection	BOOL	-	CAL	Direction of approach to the homing position switch	
				FALSE	Positive direction
				TRUE	Negative direction
ApproachVelocity	LREAL	Linear: 0.0 to 10000.0 mm/s	CAL	Approach velocity Velocity during active homing at which the reference cam and home position are approached.	
		Rotary: 0.0 to 360000.0 °/s			
ReferencingVelocity	LREAL	Linear: 0.0 to 1000.0 mm/s	CAL	Homing velocity Velocity during active homing at which the home position is approached.	
		Rotary: 0.0 to 36000.0 °/s			
HomePosition	LREAL	-1.0E12 to 1.0E12	CAL	Home position	

See also

Evaluating the technology data block (Page 1920)

"Override" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.Override.<tag name>" contains the configuration of override parameters. The override parameters are used to apply a correction percentage to default values. An override change takes effect immediately, and is performed with the dynamic settings in effect in the Motion Control instruction.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
Override.	STRUCT			
Velocity	LREAL	0.0 to 200.0%	DIR	Velocity or speed override Percentage correction of the velocity/speed

See also

Evaluating the technology data block (Page 1920)

"PositionControl" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.PositionControl.<tag name>" contains the settings of position control.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
PositionControl.	STRUCT			

Tag	Data type	Values	W	Description
Kv	LREAL	0.0 to 2147480.0	DIR	Proportional gain of the closed loop position control ("Kv" > 0.0)
Kpc	LREAL	0.0 to 150.0%	DIR	Velocity precontrol of the position control Recommended setting: • Isochronous drive connection via PROFIdrive: 100.0% • Non-isochronous drive connection via PROFIdrive: 0.0 to 100.0% • Analog drive connection: 0.0 to 100.0%
EnableDSC	BOOL	-	RES	Dynamic Servo Control (DSC) FALSE: DSC disabled TRUE: DSC activated DSC is only possible with one of the following PROFIdrive telegrams: • Standard telegram 5 or 6 • SIEMENS telegram 105 or 106
SmoothingTimeBy-ChangeDifference	LREAL	0.0 to 1.0E+12	DIR	Manipulated variable ramp at step change of control deviation due to encoder switchover.
InitialOperativeSensor	UDINT	-	RES	Active encoder after initialization of the axis. (encoder number 1 to 4) This encoder is used after startup of the CPU and after a restart of the technology object (Page 1945). At an operating mode transition from STOP → RUN of the CPU (without restart of the technology object), the encoder that was also active before the STOP is still being used.
ControlDifference-Quantization.	STRUCT			
Mode	DINT	-	RES	Type of quantification Configuration of a quantization when a drive with stepper motor interface is connected 0: No quantification 1: Quantization corresponding to encoder resolution 2: Quantization to a direct value (configuration is performed using the parameter view (data structure))
Value	LREAL	0.001 to 1.0E+12	RES	Value of quantification Configuration of a value for quantization to a direct value (<TO>.PositionControl.ControlDifferenceQuantization.Mode = 2) (configuration is performed using the parameter view (data structure))

See also

Evaluating the technology data block (Page 1920)

"DynamicAxisModel" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.DynamicAxisModel.<tag name>" contains the settings of the balancing filter.

Tags

Legend (Page 2027)

Tag		Data type	Values	W	Description
DynamicAxisModel.		STRUCT			
	VelocityTimeConstant	LREAL	0.0 to 1.0E12	DIR	Speed control loop substitute time [s]
	AdditionalPositionTi-meConstant	LREAL	0.0 to 1.0E12	DIR	Additive position control loop substitute time [s]

"FollowingError" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.FollowingError.<tag name>" contains the configuration of the dynamic following error monitoring.

If the permissible following error is exceeded, then technology alarm 521 is output, and the technology object is disabled (alarm reaction: remove enable).

When the warning level is reached, a warning is output (technology alarm 522).

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
FollowingError.	STRUCT				
EnableMonitoring	BOOL	-	RES	FALSE	Following error monitoring deactivated
				TRUE	Following error monitoring enabled
MinValue	LREAL	Linear: 0.0 to 1.0E12	DIR	Permissible following error at velocities below the value of "MinVelocity"	
		Rotary: 0.001 to 1.0E12			
MaxValue	LREAL	Linear: 0.0 to 1.0E12	DIR	Maximum permissible following error, which may be reached at the maximum velocity.	
		Rotary: 0.002 to 1.0E12			
MinVelocity	LREAL	0.0 to 1.0E12	DIR	"MinValue" is permissible below this velocity and is held constant.	
WarningLevel	LREAL	0.0 to 100.0	DIR	Warning level Percentage value relative to the valid maximum following error	

See also

Evaluating the technology data block (Page 1920)

"PositioningMonitoring" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.PositioningMonitoring.<tag name>" contains the configuration of position monitoring at the end of a positioning motion.

If the actual position value at the end of a positioning motion is reached within the tolerance time and remains in the positioning window for the minimum dwell time, then "<TO>.StatusWord.X5 (Done)" is set in the technology data block. This completes a Motion Control job.

If the tolerance time is exceeded, then technology alarm 541 "Positioning monitoring" with supplemental value 1: "Target range not reached" is displayed.

If the minimum dwell time is not met, then technology alarm 541 "Positioning monitoring" with supplemental value 2: "Exit target range again" is displayed.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
PositioningMonitoring.	STRUCT			
ToleranceTime	LREAL	0.0 to 1.0E12	DIR	Tolerance time Maximum permitted duration from reaching of velocity set-point zero until entrance into the positioning window
MinDwellTime	LREAL	0.0 to 1.0E12	DIR	Minimum dwell time in positioning window
Window	LREAL	0.0 to 1.0E12	DIR	Positioning window

See also

Evaluating the technology data block (Page 1920)

"StandstillSignal" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.StandstillSignal.<tag name>" contains the configuration of the standstill signal.

If the actual velocity value is below the velocity threshold, and does not exceed it during the minimum dwell time, then the standstill signal "<TO>.StatusWord.X7 (Standstill)" is set.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
StandstillSignal.	STRUCT			Configuration for the standstill signal
VelocityThreshold	LREAL	0.0 to 1.0E12	DIR	Velocity threshold If velocity is below this threshold, the minimum dwell time begins.
MinDwellTime	LREAL	0.0 to 1.0E12	DIR	Minimum dwell time

See also

Evaluating the technology data block (Page 1920)

"StatusPositioning" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.StatusPositioning.<tag name>" indicates the status of a positioning motion.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
StatusPositioning.	STRUCT			
Distance	LREAL	-1.0E12 to 1.0E12	RON	Distance to the target position
TargetPosition	LREAL	-1.0E12 to 1.0E12	RON	Target position
FollowingError	LREAL	-1.0E12 to 1.0E12	RON	Current following error
SetpointExecutionTime	LREAL	-1.0E12 to 1.0E12	RON	Setpoint execution time of the axis (Results from T_{lpo}, T_{vtc} or 1/kv, T_{Send} and T_O of the axis)

See also

Evaluating the technology data block (Page 1920)

"StatusDrive" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.StatusDrive.<tag name>" indicates the status of the drive.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
StatusDrive.	STRUCT				
InOperation	BOOL	-	RON	Operational status of the drive	
				FALSE	Drive not ready. Setpoints will not be executed.
				TRUE	Drive ready. Setpoints can be executed.
CommunicationOK	BOOL	-	RON	Cyclic BUS communication between controller and drive	
				FALSE	Not established
				TRUE	Established
Error	BOOL	-	RON	FALSE	No drive error
				TRUE	Drive error
AdaptionState	DINT	-	RON	Status of automatic data transfer of drive parameters	
				0	"NOT_ADAPTED" Data not transferred
				1	"IN_ADAPTION" Data transfer in progress
				2	"ADAPTED" Data transfer complete
				3	"NOT_APPLICABLE" Data transfer not selected, not possible
				4	"ADAPTION_ERROR" Error during data transfer

See also

Evaluating the technology data block (Page 1920)

"StatusServo" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.StatusServo.<tag name>" indicates the status for the balancing filter.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
StatusServo.	STRUCT			
BalancedPosition	LREAL	-	RON	Position after the balancing filter
ControlDifference	LREAL	-	RON	Control error

"StatusSensor[n]" tags (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.StatusSensor[n].<tag name>" indicates the status of the measuring system.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
StatusSensor[n].	ARRAY [1..4] OF STRUCT				
State	DINT	-	RON	Status of the actual encoder value	
				0	"NOT_VALID" Invalid
				1	"WAITING_FOR_VALID" Waiting for "Valid" status
				2	"VALID" Valid
CommunicationOK	BOOL	-	RON	Cyclic BUS communication between controller and encoder	
				FALSE	Not established
				TRUE	Established
Error	BOOL	-	RON	FALSE	No error in the measuring system
				TRUE	Error in the measuring system.
AbsEncoderOffset	LREAL	-	RON	Home point offset to the value of an absolute value encoder. The value will be retentively stored in the CPU.	
Control	BOOL	-	RON	FALSE	Encoder is not active
				TRUE	Encoder is active
Position	LREAL	-	RON	Encoder position	
Velocity	LREAL	-	RON	Encoder velocity	
AdaptionState	DINT	-	RON	Status of automatic data transfer of encoder parameters	
				0	"NOT_ADAPTED" Data not transferred
				1	"IN_ADAPTION" Data transfer in progress
				2	"ADAPTED" Data transfer complete
				3	"NOT_APPLICABLE" Data transfer not selected, not possible
				4	"ADAPTION_ERROR" Error during data transfer

See also

Evaluating the technology data block (Page 1920)

"StatusExtrapolation" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.StatusExtrapolation.<tag name>" indicates the status of the actual value extrapolation.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description
Extrapolation.	STRUCT			
FilteredPosition	LREAL	-1.0E12 to 1.0E12	RON	Position after position filter
FilteredVelocity	LREAL	-1.0E12 to 1.0E12	RON	Velocity according to tolerance band
ExtrapolatedPosition	LREAL	-1.0E12 to 1.0E12	RON	Extrapolated position
ExtrapolatedVelocity	LREAL	-1.0E12 to 1.0E12	RON	Extrapolated velocity

"StatusSynchronizedMotion" tag (synchronous axis) (S7-1500)

The tag structure "<TO>.StatusSynchronizedMotion.<tag name>" indicates the status of the synchronous operation.

Tags

Legend (Page 2027)

Tag	Data type	Value range	W	Description	
StatusSynchronizedMotion.	STRUCT				
FunctionState	DINT	-	RON	Indication of which synchronous operation function is active	
				0	No synchronous operation active
				1	Gearing (MC_GearIn)
				2	Gearing with specified synchronous positions (MC_GearInPos)
				3	Camming (MC_CamIn)
PhaseShift	LREAL	-1.0E12 to 1.0E12	RON	Current absolute leading value shift	
ActualMaster	DB_ANY	0 to 65535	RON	At the start of a "MC_GearIn" job, the number of the technology data block of the currently used leading axis is displayed. "ActualMaster" = 0 when synchronous operation is inactive	
ActualCam	DB_ANY	0 to 65535	RON	Cam that is currently being used for camming	
MasterOffset	LREAL	-1.0E12 to 1.0E12	RON	Current shift of the leading value range of the cam	
MasterScaling	LREAL	-1.0E12 to 1.0E12	RON	Current scaling of the leading value range of the cam	
SlaveOffset	LREAL	-1.0E12 to 1.0E12	RON	Current shift of the following value range of the cam	
SlaveScaling	LREAL	-1.0E12 to 1.0E12	RON	Current scaling of the following value range of the cam	
StatusWord.	STRUCT				
Bit 0	BOOL	-	RON	"MaxVelocityExceeded" Configured maximum velocity is exceeded during synchronous operation.	
Bit 1	BOOL	-	RON	"MaxAccelerationExceeded" Configured maximum acceleration is exceeded during synchronous operation.	
Bit 2	BOOL	-	RON	"MaxDecelerationExceeded" Configured maximum deceleration is exceeded during synchronous operation.	
Bit 3	BOOL	-	RON	"InSimulation"	
				FALSE	Synchronous operation not simulated
				TRUE	Synchronous operation is being simulated
Bit 4... Bit 31	BOOL	-	RON	Reserved	

See also

Synchronous axis technology object (Page 1705)

"StatusKinematicsMotion" tag (positioning axis/synchronous axis) (S7-1500)

The "<TO>.StatusKinematicsMotion" tag contains the status information of the technology object.

Information on the evaluation of the individual bits (e.g. bit 2 "MaxDecelerationExceeded") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2027)

Tag	Data type	Values	W	Description	
StatusKinematicsMotion	DWORD	-	RON	Status information of the technology object	
Bit 0	-	-	-	"MaxVelocityExceeded"	
				0	The kinematics technology object calculated a lower velocity setpoint than the maximum velocity on the axis.
				1	The kinematics technology object calculated a higher velocity setpoint than the maximum velocity on the axis.
Bit 1	-	-	-	"MaxAccelerationExceeded"	
				0	The kinematics technology object calculated a lower setpoint acceleration calculated than the maximum acceleration of the axis.
				1	The kinematics technology object calculated a higher setpoint acceleration than the maximum acceleration of the axis.
Bit 2	-	-	-	"MaxDecelerationExceeded"	
				0	The kinematics technology object calculated a lower setpoint deceleration than the maximum deceleration of the axis.
				1	The kinematics technology object calculated a lower setpoint deceleration than the maximum deceleration of the axis.

"StatusTorqueData" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.StatusTorqueData.<tag name>" indicates the status of the torque data.

Tags

Legend (Page 2027)

Tag	Data type	Value range	W	Description	
StatusTorqueData.	STRUCT				
CommandAdditiveTorqueActive	DINT	-	RON	Additive torque setpoint active function	
				0	No
				1	Yes
CommandTorqueRangeActive	DINT	-	RON	Torque limits B +, B- active function	
				0	No
				1	Yes
ActualTorque	LREAL	-1.0E12 to 1.0E12	RON	Actual torque of the axis	

"StatusMotionIn" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.StatusMotionIn.<tag name>" indicates the status of the "MotionIn" function.

Tags

Legend (Page 2027)

Tag	Data type	Value range	W	Description	
StatusMotionIn.	STRUCT				
FunctionState	DINT	-	RON	0	No "MotionIn" function active
				1	"MC_MotionInVelocity" active
				2	"MC_MotionInPosition" active

"StatusWord" tag (positioning axis/synchronous axis) (S7-1500)

The "<TO>.StatusWord" tag contains the status information of the technology object.

Information on the evaluation of the individual bits (e.g. bit 5 "HomingDone") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2027)

Tag	Data type	Values	W	Description
StatusWord	DWORD	-	RON	Status information of the technology object

Tag	Data type	Values	W	Description	
Bit 0	-	-	-	"Enable" Enable status	
				0	The technology object is disabled.
				1	The technology object is enabled.
Bit 1	-	-	-	"Error"	
				0	No error is present.
				1	An error is present.
Bit 2	-	-	-	"RestartActive"	
				0	No restart is active.
				1	A restart is active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged"	
				0	The restart tags are unchanged.
				1	The restart tags have been changed. For the changes to be applied, the technology object must be reinitialized.
Bit 4	-	-	-	"ControlPanelActive"	
				0	The axis control panel is deactivated.
				1	The axis control panel is active.
Bit 5	-	-	-	"HomingDone" Homing status	
				0	The technology object is not homed.
				1	The technology object is homed.
Bit 6	-	-	-	"Done"	
				0	A motion job is in progress or the axis control panel is activated.
				1	No motion job is in progress and the axis control panel is deactivated.
Bit 7	-	-	-	"Standstill" Standstill signal	
				0	The axis is in motion.
				1	The axis is at a standstill.
Bit 8	-	-	-	"PositioningCommand"	
				0	No positioning job is active.
				1	A positioning job is active ("MC_MoveRelative","MC_MoveAbsolute").
Bit 9	-	-	-	"JogCommand"	
				0	No "MC_MoveJog" job is active.
				1	An "MC_MoveJog" job is running
Bit 10	-	-	-	"VelocityCommand"	
				0	No "MC_MoveVelocity" job is active.
				1	An "MC_MoveVelocity" job is running
Bit 11	-	-	-	"HomingCommand"	
				0	No "MC_Home" job is in progress.

Tag	Data type	Values	W	Description	
				1	An "MC_Home" job is being processed.
Bit 12	-	-	-	"ConstantVelocity"	
				0	The axis is accelerated or decelerated.
				1	The setpoint velocity is reached. A constant velocity setpoint is output.
Bit 13	-	-	-	"Accelerating"	
				0	No acceleration operation is active.
				1	An acceleration operation is active.
Bit 14	-	-	-	"Decelerating"	
				0	No deceleration process is active.
				1	A deceleration operation is active.
Bit 15	-	-	-	"SWLimitMinActive"	
				0	No negative software limit switch was approached.
				1	A negative software limit switch was reached or exceeded.
Bit 16	-	-	-	"SWLimitMaxActive"	
				0	No positive software limit switch was approached.
				1	A positive software limit switch was reached or exceeded.
Bit 17	-	-	-	"HWLimitMinActive"	
				0	No negative hardware limit switch was approached.
				1	A negative hardware limit switch was reached or exceeded.
Bit 18	-	-	-	"HWLimitMaxActive"	
				0	No positive hardware limit switch was approached.
				1	A positive hardware limit switch was reached or exceeded.
Bit 19	-	-	-	Reserved	
Bit 20	-	-	-	Reserved	
Bit 21	-	-	-	Positioning axis technology object: Reserved	
				Synchronous axis technology object: "Synchronizing"	
				0	The axis does not synchronize to a leading value.
				1	The axis synchronizes to a leading value.
Bit 22	-	-	-	Positioning axis technology object: Reserved	
				Synchronous axis technology object: "Synchronous"	

Tag	Data type	Values	W	Description	
				0	The axis moves asynchronous to a leading value.
				1	The axis moves synchronously to a leading value.
Bit 23	-	-	-	"SuperimposedMotionCommand"	
				0	No overlaid movement is active.
				1	No overlaid movement is running.
Bit 24	-	-	-	Positioning axis technology object: Reserved	
				Synchronous axis technology object: "PhasingCommand"	
				0	No Motion Control instruction for leading value shift is active.
				1	A Motion Control instruction for leading value shift is active.
Bit 25	-	-	-	"AxisSimulation"	
				0	The simulation is not running.
				1	The simulation is active.
Bit 26	-	-	-	"TorqueLimitingCommand"	
				0	No "MC_TorqueLimiting" job is active.
				1	An "MC_TorqueLimiting" job is running
Bit 27	-	-	-	"InLimitation"	
				0	The drive does not operate at the torque limit.
				1	The drive is operating at the torque limit.
Bit 28	-	-	-	"NonPositionControlled"	
				0	The axis is in position-controlled mode.
				1	The axis is not in position-controlled mode.
Bit 29	-	-	-	"KinematicsMotionCommand"	
				0	The axis is not used for a kinematics job.
				1	The axis is used for a kinematics job.
Bit 30	-	-	-	"InClamping"	
				0	The axis is not clamped at a fixed stop.
				1	The axis is clamped at a fixed stop.
Bit 31	-	-	-	"MotionInCommand"	
				0	No "MotionIn" job is active.
				1	A "MotionIn" job is running.

See also

Evaluating the technology data block (Page 1920)

"ErrorWord" tag (positioning axis/synchronous axis) (S7-1500)

The "<TO>.ErrorWord" tag indicates technology object errors (technology alarms).

Information on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2027)

Tag	Data type	Values	W	Description
ErrorWord	DWORD	-	RON	

Tag	Data type	Values	W	Description
Bit 0	-	-	-	"SystemFault" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4	-	-	-	"DriveFault" Error in drive
Bit 5	-	-	-	"SensorFault" Error in encoder system
Bit 6	-	-	-	"DynamicError" Specified dynamic values are limited to permissible values.
Bit 7	-	-	-	"CommunicationFault" Communication error Missing or faulty communication.
Bit 8	-	-	-	"SWLimit" Software limit switch reached or overtraveled.
Bit 9	-	-	-	"HWLimit" Hardware limit switch reached or overtraveled.
Bit 10	-	-	-	"HomingError" Error during homing operation The homing cannot be completed.
Bit 11	-	-	-	"FollowingErrorFault" Following error limits exceeded
Bit 12	-	-	-	"PositioningFault" Positioning error
Bit 13	-	-	-	"PeripheralError" Error accessing a logical address
Bit 14	-	-	-	Positioning axis technology object: Reserved
				Synchronous axis technology object: "SynchronousError" Error during synchronous operation The leading axis specified for the Motion Control instruction "MC_GearIn" was not configured as a possible leading axis.
Bit 15	-	-	-	"AdaptionError"

Tag	Data type	Values	W	Description
				Error in automatic data transfer
Bit 16 ... Bit 31	-	-	-	Reserved

See also

Evaluating the technology data block (Page 1920)

"ErrorDetail" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.ErrorDetail.<tag name>" contains the alarm number and the effective local alarm reaction for the technology alarm that is currently pending on the technology object.

You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.

Tags

Legend (Page 2027)

Tag	Data type	Values	W	Description	
ErrorDetail.	STRUCT				
Number	UDINT	-	RON	Alarm number	
Reaction	DINT	0 to 5	RON	Effective alarm reaction	
				0	No reaction
				1	Stop with current dynamic values
				2	Stop with maximum dynamic values
				3	Stop with emergency stop ramp
				4	Remove enable
				5	Track setpoints

See also

Evaluating the technology data block (Page 1920)

Technology alarms (Page 1961)

"WarningWord" tag (positioning axis/synchronous axis) (S7-1500)

The "<TO>.WarningWord" tag indicates pending warnings at the technology object.

Information on the evaluation of the individual bits (e.g. bit 13 "PeripheralWarning") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2027)

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	

Tag	Data type	Values	W	Description
Bit 0	-	-	-	"SystemWarning" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigWarning" Configuration error One or several configuration parameters are adjusted internally.
Bit 2	-	-	-	"UserWarning" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4	-	-	-	"DriveWarning" Error in drive
Bit 5	-	-	-	"SensorWarning" Error in encoder system
Bit 6	-	-	-	"DynamicError" Specified dynamic values are limited to permissible values.
Bit 7	-	-	-	"CommunicationWarning" Communication error Missing or faulty communication.
Bit 8	-	-	-	"SWLimitMin"
Bit 9	-	-	-	"SWLimitMax"
Bit 10	-	-	-	"HomingWarning" Error during homing operation The homing cannot be completed.
Bit 11	-	-	-	"FollowingErrorWarning" Warning limit of following error monitoring reached/exceeded
Bit 12	-	-	-	"PositioningWarning" Positioning error
Bit 13	-	-	-	"PeripheralWarning" Error accessing a logical address
Bit 14	-	-	-	Positioning axis technology object: Reserved
				Synchronous axis technology object: "SynchronousWarning" Error during synchronous operation The leading axis specified for the Motion Control instruction "MC_GearIn" was not configured as a possible leading axis.
Bit 15	-	-	-	"AdaptionWarning" Error in automatic data transfer

Tag		Data type	Values	W	Description
	Bit 16... Bit 31	-	-	-	Reserved

See also

Evaluating the technology data block (Page 1920)

"ControlPanel" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.ControlPanel.<tag name>" contains no user-relevant data. This tag structure is internally used.

See also

Evaluating the technology data block (Page 1920)

"InternalToTrace" tag (positioning axis/synchronous axis) (S7-1500)

The tag structure "<TO>.InternalToTrace.<tag name>" contains no user-relevant data. This tag structure is internally used.

See also

Evaluating the technology data block (Page 1920)

Tags of the technology object external encoder (S7-1500)

Legend (S7-1500)

Tag	Name of the tag
Data type	Data type of the tag
Values	Value range of the tag - minimum value to maximum value If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".

W	Effectiveness of changes in the technology data block	
	DIR	Direct: Values are changed directly and take effect at the start of the next MC-Servo [OB91].
	CAL	At call of Motion Control instruction: Values are changed directly and take effect at the start of the next MC-Servo [OB91] after the call of the corresponding Motion Control instruction in the user program.
	RES	Restart: Changes to the start value in the load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed during runtime of the user program.
Description	Description of the tag	

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Actual values and setpoints (external encoder) (S7-1500)

The following tags indicate the setpoint and actual values of the technology object.

Tags

Legend (Page 2067)

Tag	Data type	Values	W	Description
ActualPosition	LREAL	-	RON	Actual position
ActualVelocity	LREAL	-	RON	Actual velocity
ActualAcceleration	LREAL	-	RON	Actual acceleration

"Sensor" tag (external encoder) (S7-1500)

The tag structure "<TO>.Sensor.<tag name>" contains the controller-end configuration for the encoder, and the configuration for active and passive homing.

Tags

Legend (Page 2067)

Tag	Data type	Values	W	Description
Sensor.	STRUCT			

Tag	Data type	Values	W	Description	
Type	DINT	-	RES	Encoder type	
				0	"INCREMENTAL" Incremental
				1	"ABSOLUTE" Absolute
				2	"CYCLIC_ABSOLUTE" Cyclic absolute
InverseDirection	BOOL	-	RES	Inversion of the actual value	
				FALSE	No
				TRUE	Yes
System	DINT	-	RES	Encoder system	
				0	"LINEAR" Linear encoder
				1	"ROTATORY" Rotary encoder
MountingMode	DINT	-	RES	Mounting type of encoder	
				0	On motor shaft
				1	On load side
				2	External measuring system
DataAdaption	DINT	-	RES	Automatic transfer of the drive values reference speed, maximum speed and reference torque in the device	
				0	No automatic transfer, manual configuration of values
				1	Automatic transfer of values configured in the drive to the configuration of the technology object
Interface.					
AddressIn	VREF	0 to 65535	RON	Input address for the PROFIdrive telegram	
AddressOut	VREF	0 to 65535	RON	Output address for the PROFIdrive telegram	
Number	UDINT	1 to 2	RON	Number of the encoder in the telegram	
Parameter.					
Resolution	LREAL	-1.0E12 to 1.0E12	RES	Resolution of a linear encoder (offset between two encoder pulses)	
StepsPerRevolution	UDINT	1 to 8388608	RES	Increments per rotary encoder revolution	
FineResolutionXist1	UDINT	0 to 31	RES	Number of bits for fine resolution Gx_XIST1 (cyclic actual encoder value)	
FineResolutionXist2	UDINT	0 to 31	RES	Number of bits for fine resolution Gx_XIST2 (absolute value of encoder)	
DeterminableRevolutions	UDINT	0 to 8388608	RES	Number of differentiable encoder revolutions for a multi-turn absolute encoder (For a single-turn absolute encoder = 1; for an incremental encoder = 0)	
DistancePerRevolution	LREAL	0.0 to 1.0E12	RES	Load distance per revolution of an externally mounted encoder	
PassiveHoming.	STRUCT				

Tag	Data type	Values	W	Description	
Mode	DINT	-	RES	Homing mode	
				0	Use zero mark via PROFIdrive telegram
				1	Zero mark via PROFIdrive telegram and reference cam
				2	Use homing mark via digital input
SideInput	BOOL	-	CAL	Side of the digital input for passive homing	
				FALSE	Negative side
				TRUE	Positive side
Direction	DINT	-	CAL	Homing direction/approach direction on the homing mark	
				0	Positive homing direction
				1	Negative homing direction
				2	Current homing direction
DigitalInputAddress	VREF	0 to 65535	RES	Address of the digital input	
SwitchLevel	BOOL	-	RON	Signal level that is present at the digital input when homing mark is approached	
				FALSE	Low level
				TRUE	High level

"Extrapolation" tag (external encoder) (S7-1500)

The tag structure "<TO>.Extrapolation.<tag name>" contains the configuration of the actual value extrapolation.

Tags

Legend (Page 2067)

Tag	Data type	Values	W	Description
Extrapolation.	STRUCT			
LeadingAxisDependentTime	LREAL	-	RON	Extrapolation time component (caused by leading axis) Results from the following times: • Time of actual value acquisition for the leading axis • Interpolator cycle clock • Time of actual position filter of leading axis (T1 + T2)
FollowingAxisDependentTime	LREAL	0.001 to 1.0E12	DIR	Extrapolation time component (caused by following axis) Results from the following times: • For a following axis with set velocity precontrol: – Communication cycle – Interpolator cycle clock – Speed control loop substitute time for the following axis – Output time of the setpoint for the following axis • For a following axis without velocity precontrol: – Communication cycle – Interpolator cycle clock – Position control loop equivalent time (1/Kv from <TO>.PositionControl.Kv) – Output time of the setpoint for the following axis
PositionFilter.	STRUCT			
T1	LREAL	0.001 to 1.0E12	DIR	Position filter time constant T1
T2	LREAL	0.001 to 1.0E12	DIR	Position filter time constant T2
VelocityFilter.	STRUCT			
T1	LREAL	0.001 to 1.0E12	DIR	Velocity filter time constant T1
VelocityTolerance.	STRUCT			
Range	LREAL	0.001 to 1.0E12	DIR	Tolerance band width for velocity
Hysteresis.	STRUCT			
Value	LREAL	0.001 to 1.0E12	DIR	Hysteresis value

"LoadGear" tag (external encoder) (S7-1500)

The tag structure "<TO>.LoadGear.<tag name>" contains the configuration of the load gear.

Tags

Legend (Page 2067)

Tag	Data type	Value range	W	Description
LoadGear.	STRUCT			
Numerator	UDINT	1 to 4294967295	RES	Load gear counter
Denominator	UDINT	1 to 4294967295	RES	Load gear denominator

"Properties" tag (external encoder) (S7-1500)

The tag structure "<TO>.Properties.<tag name>" contains the configuration of the type of axis or motion.

Tags

Legend (Page 2067)

Tag	Data type	Value range	W	Description	
Properties.	STRUCT				
MotionType	DINT	-	RON	Display of axis type or motion type	
				0	Linear axis or motion
				1	Rotary axis or motion

"Units" tag (external encoder) (S7-1500)

The tag structure "<TO>.Units.<tag name>" shows the set technological units.

Tags

Legend (Page 2067)

Tag	Data type	Values	W	Description	
Units.	STRUCT				
LengthUnit	UDINT	-	RON	Unit for position	
				1010	m
				1013	mm
				1011	km
				1014	µm
				1015	nm
				1019	in
				1018	ft
				1021	mi
				1004	rad
				1005	°
VelocityUnit	UDINT	-	RON	Unit for velocity	
				1521	°/s
				1522	°/min
				1086	rad/s
				1523	rad/min
				1062	mm/s
				1061	m/s
				1524	mm/min
				1525	m/min
				1526	mm/h
				1063	m/h
				1527	km/min
				1064	km/h
				1066	in/s
				1069	in/min
				1067	ft/s
				1070	ft/min
				1075	mi/h
TimeUnit	UDINT	-	RON	Unit for time	
				1054	s

"Mechanics" tag (external encoder) (S7-1500)

The tag structure "<TO>.Mechanics.<tag name>" contains the configuration of the mechanics.

Tags

Legend (Page 2067)

Tag	Data type	Value range	W	Description
Mechanics.	STRUCT			
LeadScrew	LREAL	0.0 to 1.0E12	RES	Leadscrew pitch

"Modulo" tag (external encoder) (S7-1500)

The tag structure "<TO>.Modulo.<tag name>" contains the configuration of the modulo function.

Tags

Legend (Page 2067)

Tag	Data type	Values	W	Description	
Modulo.	STRUCT				
Enable	BOOL	-	RES	FALSE	Modulo conversion disabled
				TRUE	Modulo conversion enabled
				When modulo conversion is enabled, a check is made for modulo length > 0.0	
Length	LREAL	0.001 to 1.0E12	RES	Modulo length	
StartValue	LREAL	-1.0E12 to 1.0E12	RES	Modulo start value	

"Homing" tag (external encoder) (S7-1500)

The tag structure "<TO>.Homing.<tag name>" contains the configuration for homing the TO.

Tags

Legend (Page 2067)

Tag	Data type	Values	W	Description
Homing.	STRUCT			
HomePosition	LREAL	-1.0E12 to 1.0E12	CAL	Home position

"StatusSensor" tag (external encoder) (S7-1500)

The tag structure "<TO>.StatusSensor.<tag name>" indicates the status of the measuring system.

Tags

Legend (Page 2067)

Tag	Data type	Values	W	Description	
StatusSensor.	STRUCT				
State	DINT	-	RON	Status of the actual encoder value	
				0	"NOT_VALID" Invalid
				1	"WAITING_FOR_VALID" Waiting for "Valid" status
				2	"VALID" Valid
CommunicationOK	BOOL	-	RON	Cyclic BUS communication between controller and encoder	
				FALSE	Not established
				TRUE	Established
Error	BOOL	-	RON	FALSE	No error in the measuring system
				TRUE	Error in the measuring system.
AbsEncoderOffset	LREAL	-	RON	Home point offset to the value of an absolute value encoder. The value will be retentively stored in the CPU.	
Control	BOOL	-	RON	FALSE	Encoder is not active
				TRUE	Encoder is active
Position	LREAL	-	RON	Encoder position	
Velocity	LREAL	-	RON	Encoder velocity	
AdaptionState	DINT	-	RON	Status of automatic data transfer of encoder parameters	
				0	"NOT_ADAPTED" Data not transferred
				1	"IN_ADAPTION" Data transfer in progress
				2	"ADAPTED" Data transfer complete
				3	"NOT_APPLICABLE" Data transfer not selected, not possible
				4	"ADAPTION_ERROR" Error during data transfer

"StatusExtrapolation" tag (external encoder) (S7-1500)

The tag structure "<TO>.StatusExtrapolation.<tag name>" indicates the status of the actual value extrapolation.

Tags

Legend (Page 2067)

Tag	Data type	Values	W	Description
Extrapolation.	STRUCT			
FilteredPosition	LREAL	-1.0E12 to 1.0E12	RON	Position after position filter
FilteredVelocity	LREAL	-1.0E12 to 1.0E12	RON	Velocity according to tolerance band
ExtrapolatedPosition	LREAL	-1.0E12 to 1.0E12	RON	Extrapolated position
ExtrapolatedVelocity	LREAL	-1.0E12 to 1.0E12	RON	Extrapolated velocity

"StatusWord" tag (external encoder) (S7-1500)

The "<TO>.StatusWord" tag contains the status information of the technology object.

Information on the evaluation of the individual bits (e.g. bit 5 "HomingDone") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2067)

Tag	Data type	Values	W	Description	
StatusWord	DWORD	-	RON	Status information of the technology object	
Bit 0	-	-	-	"Enable" Enable status	
				0	Technology object disabled
				1	Technology object enabled
Bit 1	-	-	-	"Error"	
				0	No error present
				1	Error present
Bit 2	-	-	-	"RestartActive"	
				0	No restart active
				1	Restart active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged"	
				0	Restart tags unchanged
				1	Change to Restart tags. For the changes to be applied, the technology object must be reinitialized.
Bit 4	-	-	-	Reserved	
Bit 5	-	-	-	"HomingDone" Homing status	
				0	Technology object not homed
				1	Technology object homed
Bit 6	-	-	-	"Done"	
				0	Motion job is active or axis control panel enabled
				1	No motion job in process and axis control panel disabled
Bit 7	-	-	-	Reserved	
Bit 8	-		-	Reserved	
Bit 9	-	-	-	Reserved	
Bit 10	-	-	-	Reserved	
Bit 11	-	-	-	"HomingCommand"	
				0	No "MC_Home" job in progress
				1	"MC_Home" job in progress
Bit 12... Bit 31	-	-	-	Reserved	

"ErrorWord" tag (external encoder) (S7-1500)

The "<TO>.ErrorWord" tag indicates technology object errors (technology alarms).

Information on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2067)

Tag	Data type	Values	W	Description
ErrorWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4	-	-	-	Reserved
Bit 5	-	-	-	"SensorFault" Error in encoder system
Bit 6	-	-	-	Reserved
Bit 7	-	-	-	"CommunicationFault" Communication error Missing or faulty communication.
Bit 8	-	-	-	Reserved
Bit 9	-	-	-	Reserved
Bit 10	-	-	-	"HomingError" Error during homing operation The homing cannot be completed.
Bit 11	-	-	-	Reserved
Bit 12	-	-	-	Reserved
Bit 13	-	-	-	"PeripheralError" Error accessing a logical address
Bit 14 … Bit 31	-	-	-	Reserved

"ErrorDetail" tag (external encoder) (S7-1500)

The tag structure "<TO>.ErrorDetail.<tag name>" contains the alarm number and the effective local alarm reaction for the technology alarm that is currently pending on the technology object.

You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.

Tags

Legend (Page 2067)

Tag	Data type	Values	W	Description	
ErrorDetail.	STRUCT				
Number	UDINT	-	RON	Alarm number	
Reaction	DINT	0, 10	RON	Effective alarm reaction	
				0	No reaction
				10	Remove enable

"WarningWord" tag (external encoder) (S7-1500)

The "<TO>.WarningWord" tag indicates pending warnings at the technology object.

Information on the evaluation of the individual bits (e.g. bit 13 "PeripheralWarning") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2067)

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemWarning" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigWarning" Configuration error One or several configuration parameters are adjusted internally.
Bit 2	-	-	-	"UserWarning" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4	-	-	-	Reserved
Bit 5	-	-	-	"SensorWarning" Error in encoder system
Bit 6	-	-	-	Reserved
Bit 7	-	-	-	"CommunicationWarning" Communication error Missing or faulty communication.
Bit 8	-	-	-	Reserved
Bit 9	-	-	-	Reserved
Bit 10	-	-	-	"HomingWarning" Error during homing operation The homing cannot be completed.
Bit 11	-	-	-	Reserved
Bit 12	-	-	-	Reserved
Bit 13	-	-	-	"PeripheralWarning" Error accessing a logical address
Bit 14... Bit 31	-	-	-	Reserved

"InternalToTrace" tag (external encoder) (S7-1500)

The tag structure "<TO>.InternalToTrace.<tag name>" contains no user-relevant data. This tag structure is internally used.

Tags

Legend (Page 2067)

Tag		Data type	Values	W	Description
InternalToTrace.		ARRAY [1..4] OF STRUCT			
	Id	DINT	-	DIR	-
	Value	LREAL	-	RON	-

Tags of the measuring input technology object (S7-1500)

Legend (S7-1500)

Tag	Name of the tag	
Data type	Data type of the tag	
Values	Value range of the tag - minimum value to maximum value If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".	
W	Effectiveness of changes in the technology data block	
	DIR	Direct: Values are changed directly and take effect at the start of the next MC-Servo [OB91].
	CAL	At call of Motion Control instruction: Values are changed directly and take effect at the start of the next MC-Servo [OB91] after the call of the corresponding Motion Control instruction in the user program.
	RES	Restart: Changes to the start value in the load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed during runtime of the user program.
Description	Description of the tag	

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Display data (measuring input) (S7-1500)

The "<TO>.Status" and "<TO>.InputState" tags show the status of the measuring function and the measuring input.

Tags

Legend (Page 2082)

Tag	Data type	Values	W	Description	
Status	DINT	-	RON	Measuring input function status	
				0	"INACTIVE" Measurement is not active.
				1	"WAITING_FOR_TRIGGER" Measuring input is waiting for measuring event.
				2	"TRIGGER_OCCURRED" One or more measured values have been captured.
				3	"MEASURING_ERROR" Error during the measurement
InputState	BOOL	-	RON	Status of measuring input	
				FALSE	Input of measuring input not active
				TRUE	Input of measuring input active

"Parameter" tag (measuring input) (S7-1500)

The tag structure "<TO>.Parameter.<tag name>" contains the configuration of the basic parameters of the measuring input technology object.

Tags

Legend (Page 2082)

Tag	Data type	Values	W	Description	
Parameter.	STRUCT				
MeasuringInputType	DINT	-	RON	Measuring input type	
				0	"TIME_BASED" Measurement using time stamp
				1	"PROFIDRIVE" Measuring event using PROFIdrive telegram
PROFIdriveProbeNumber	UDINT	1, 2	RES	Number of the measuring input to be used for a measurement via PROFIdrive telegram	
MeasuringRangeActivationTime	LREAL	0.0 to 1.0E12	RON	System share for activation time of measuring range	
MeasuringRangeAdditionalActivationTime	LREAL	0.0 to 1.0E12	RES	Additional activation time when using measuring range limits [ms]	
CorrectionTime	LREAL	0.0 to 1.0E12	RES	Correction time for the measurement result [ms]	

"Interface" tag (measuring input) (S7-1500)

The tag structure "<TO>.Interface.<tag name>" contains the configuration of the input properties for the measuring input technology object.

Tags

Legend (Page 2082)

Tag	Data type	Values	W	Description
Interface.	STRUCT			
Address	VREF	-	RON	I/O address for the digital measuring input

"Units" tag (measuring input) (S7-1500)

The tag structure "<TO>.Units.<tag name>" shows the set technological units.

Tags

Legend (Page 2082)

Tag	Data type	Values	W	Description	
Units.	STRUCT				
LengthUnit	UDINT	-	RON	Unit for position	
				1010	m
				1013	mm
				1011	km
				1014	µm
				1015	nm
				1019	in
				1018	ft
				1021	mi
				1004	rad
				1005	°
TimeUnit	UDINT	-	RON	Unit for time	
				1056	ms

"MeasuredValues" tag (measuring input) (S7-1500)

The tag structure "<TO>.MeasuredValues.<tag name>" displays the measurement results.

Tags

Legend (Page 2082)

Tag	Data type	Values	W	Description
MeasuredValues.	STRUCT			
MeasuredValue1	LREAL	-1.0E12 to 1.0E12	RON	First measured value
MeasuredValue2	LREAL	-1.0E12 to 1.0E12	RON	Second measured value
MeasuredValue1Counter	UDINT	0 to 2147483647	RON	Count value for the first measured value
MeasuredValue2Counter	UDINT	0 to 2147483647	RON	Count value for the second measured value
LostEdgeCounter1	UDINT	0 to 7	RON	LEC for measured value 1 (Zero in the case of one-time measurement)
LostEdgeCounter2	UDINT	0 to 7	RON	LEC for measured value 2 (Zero in the case of one-time measurement)

"StatusWord" tag (measuring input) (S7-1500)

The "<TO>.StatusWord" tag contains the status information of the technology object.

Information on the evaluation of the individual bits (e.g. bit 2 "RestartActive") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tags

Legend (Page 2082)

Tag	Data type	Values	W	Description	
StatusWord	DWORD	-	RON	Status information of the technology object	
Bit 0	-	-	-	"Control"	
				0	Technology object not in operation
				1	Technology object in operation
Bit 1	-	-	-	"Error"	
				0	No error present
				1	Error present
Bit 2	-	-	-	"RestartActive"	
				0	No restart active
				1	Restart active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged"	
				0	Restart tags unchanged
				1	Change to Restart tags. For the changes to be applied, the technology object must be reinitialized.
Bit 4	-	-	-	Reserved	
Bit 5	-	-	-	"CommunicationOK" Communication between measuring input and measuring module	
				0	Not established
				1	Established
Bit 6... Bit 31	-	-	-	Reserved	

"ErrorWord" tag (measuring input) (S7-1500)

The "<TO>.ErrorWord" tag indicates technology object errors (technology alarms).

Information on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tags

Legend (Page 2082)

Tag	Data type	Values	W	Description
ErrorWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4... Bit 12	-	-	-	Reserved
Bit 13	-	-	-	"PeripheralError" Error accessing a logical address
Bit 14... Bit 31	-	-	-	Reserved

"ErrorDetail" tag (measuring input) (S7-1500)

The tag structure "<TO>.ErrorDetail.<tag name>" contains the alarm number and the effective local alarm reaction for the technology alarm that is currently pending on the technology object.

You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.

Tags

Legend (Page 2082)

Tag	Data type	Values	W	Description	
ErrorDetail.	STRUCT				
Number	UDINT	-	RON	Alarm number	
Reaction	DINT	0, 8	RON	Effective alarm reaction	
				0	No reaction
				8	End measuring input processing

"WarningWord" tag (measuring input) (S7-1500)

The "<TO>.WarningWord" tag indicates pending warnings at the technology object.

Information on the evaluation of the individual bits (e.g. bit 1 "ConfigWarning") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2082)

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemWarning" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigWarning" Configuration error One or several configuration parameters are adjusted internally.
Bit 2	-	-	-	"UserWarning" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4... Bit 12	-	-	-	Reserved
Bit 13				"PeripheralWarning" Error accessing a logical address
Bit 14... Bit 31	-	-	-	Reserved

Tags of the cam technology object (S7-1500)

Legend (S7-1500)

Tag	Name of the tag
Data type	Data type of the tag
Values	Value range of the tag - minimum value to maximum value If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".

W	Effectiveness of changes in the technology data block	
	DIR	Direct: Values are changed directly and take effect at the start of the next MC-Servo [OB91].
	CAL	At call of Motion Control instruction: Values are changed directly and take effect at the start of the next MC-Servo [OB91] after the call of the corresponding Motion Control instruction in the user program.
	RES	Restart: Changes to the start value in the load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed during runtime of the user program.
Description	Description of the tag	

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Display data (cam) (S7-1500)

The "<TO>.CamOutput" tag indicates the switching state of the cam.

Tags

Legend (Page 2088)

Tag	Data type	Values	W	Description	
CamOutput	BOOL	-	RON	Switching state of output cam	
				FALSE	Not switched
				TRUE	Switched

"Parameter" tag (output cam) (S7-1500)

The tag structure "<TO>.Parameter.<tag name>" contains the configuration of the basic parameters of the output cam technology object.

Tags

Legend (Page 2088)

Tag	Data type	Values	W	Description	
Parameter.	STRUCT				
OutputCamType	DINT	-	RES	Output cam type	
				0	Distance output cam
				1	Time-based output cam
PositionType	DINT	-	RES	Position reference	
				0	Position setpoint
				1	Actual position
OnCompensation	LREAL	0.0 to 1.0E12	DIR	Activation time Lead time for the switch-on edge	
OffCompensation	LREAL	0.0 to 1.0E12	DIR	Deactivation time Lead time for the switch-off edge	
Hysteresis	LREAL	0.0 to 1.0E12	DIR	Hysteresis value For output cams with reference to actual position, always enter a hysteresis value (> 0.0).	

"Interface" tag (output cam) (S7-1500)

The tag structure "<TO>.Interface.<tag name>" contains the configuration of the output properties for the output cam technology object.

Tags

Legend (Page 2088)

Tag	Data type	Values	W	Description	
Interface.	STRUCT				
EnableOutput	BOOL	-	RES	Activation of the output cam output	
				FALSE	Output is deactivated
				TRUE	Output is activated
Address	VREF	-	RON	I/O address of the output cam	
LogicOperation	DINT	-	RON	Logical operation of the output cam signals at the output	
				0	OR logic operation
				1	AND logic operation

"Units" tag (output cam) (S7-1500)

The tag structure "<TO>.Units.<tag name>" shows the set technological units.

Tags

Legend (Page 2088)

Tag	Data type	Values	W	Description	
Units.	STRUCT				
LengthUnit	UDINT	-	RON	Unit for position	
				1010	m
				1013	mm
				1011	km
				1014	µm
				1015	nm
				1019	in
				1018	ft
				1021	mi
				1004	rad
				1005	°
TimeUnit	UDINT	-	RON	Unit for time	
				1056	ms

"StatusWord" tag (output cam) (S7-1500)

The "<TO>.StatusWord" tag contains the status information of the technology object.

Information on the evaluation of the individual bits (e.g. bit 2 "RestartActive") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tags

Legend (Page 2088)

Tag	Data type	Values	W	Description	
StatusWord	DWORD	-	RON	Status information of the technology object	
Bit 0	-	-	-	"Control"	
				0	Technology object not in operation
				1	Technology object in operation
Bit 1	-	-	-	"Error"	
				0	No error present
				1	Error present
Bit 2	-	-	-	"RestartActive"	
				0	No restart active
				1	Restart active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged"	
				0	Restart tags unchanged
				1	Change to Restart tags. For the changes to be applied, the technology object must be reinitialized.
Bit 4	-	-	-	"OutputInverted"	
				0	Output cam output not inverted
				1	Output cam output inverted
Bit 5	-	-	-	"CommunicationOK" Communication between output cam and output module	
				0	Not established
				1	Established
Bit 6... Bit 31	-	-	-	Reserved	

"ErrorWord" tag (output cam) (S7-1500)

The "<TO>.ErrorWord" tag indicates technology object errors (technology alarms).

Information on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tags

Legend (Page 2088)

Tag	Data type	Values	W	Description
ErrorWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4... Bit 12	-	-	-	Reserved
Bit 13	-	-	-	"PeripheralError" Error accessing a logical address
Bit 14... Bit 31	-	-	-	Reserved

"ErrorDetail" tag (output cam) (S7-1500)

The tag structure "<TO>.ErrorDetail.<tag name>" contains the alarm number and the effective local alarm reaction for the technology alarm that is currently pending on the technology object.

You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.

Tags

Legend (Page 2088)

Tag	Data type	Values	W	Description	
ErrorDetail.	STRUCT				
Number	UDINT	-	RON	Alarm number	
Reaction	DINT	0, 6	RON	Effective alarm reaction	
				0	No reaction
				6	Output cam processing is complete.

"WarningWord" tag (output cam) (S7-1500)

The "<TO>.WarningWord" tag indicates pending warnings at the technology object.

Information on the evaluation of the individual bits (e.g. bit 1 "ConfigWarning") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2088)

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemWarning" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigWarning" Configuration error One or several configuration parameters are adjusted internally.
Bit 2	-	-	-	"UserWarning" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4... Bit 12	-	-	-	Reserved
Bit 13				"PeripheralWarning" Error accessing a logical address
Bit 14... Bit 31	-	-	-	Reserved

Tags of the cam track technology object (S7-1500)

Legend (S7-1500)

Tag	Name of the tag
Data type	Data type of the tag
Values	Value range of the tag - minimum value to maximum value If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".

W	Effectiveness of changes in the technology data block	
	DIR	Direct: Values are changed directly and take effect at the start of the next MC-Servo [OB91].
	CAL	At call of Motion Control instruction: Values are changed directly and take effect at the start of the next MC-Servo [OB91] after the call of the corresponding Motion Control instruction in the user program.
	RES	Restart: Changes to the start value in the load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed during runtime of the user program.
Description	Description of the tag	

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Display data (cam track) (S7-1500)

The following tags indicate the status of the cam track:

Tags

Legend (Page 2094)

Tag	Data type	Values	W	Description	
Status	DINT	-	RON	0	"INACTIVE" (inactive)
				1	"ACTIVE" (active)
				2	"ACTIVE_WAITING_FOR_NEXT_CYCLE" (active and waiting for next track)
TrackOutput	BOOL	-	RON	FALSE	Cam track is not output.
				TRUE	Cam track is output.
SingleCamState	DWORD	16#0 to 16#FFFF_FFFF	RON	Switched on output cam (bit-masked)	
				0	Output cam is not switched on
				1	Output cam is switched on
TrackPosition	LREAL	-1.0E12 to 1.0E12	RON	Display of the current position within the cam track	
MatchPosition	LREAL	-1.0E12 to 1.0E12	RON	Reference position of the current cam track During cyclic processing of the cam track, the continued reference position of the current cam track is displayed. The unique detection and output of the position is only possible when the assigned technology object is in motion.	

"Parameter" tag (cam track) (S7-1500)

The tag structure "<TO>.Parameter.<tag name>" contains the configuration of the basic parameters of the cam track technology object.

Tags

Legend (Page 2094)

Tag	Data type	Values	W	Description	
Parameter.	STRUCT				
CamTrackType	DINT	-	RES	Output cam type	
				0	Distance output cam
				1	Time-based output cam
PositionType	DINT	-	RES	Position reference	
				0	Position setpoint
				1	Actual position
ReferencePosition	LREAL	-1.0E12 to 1.0E12	DIR	Reference position	
CamTrackLength	LREAL	0.001 to 1.0E12	DIR	Track length	
CamMasking	DWORD	16#0 to 16#FFFF_FFFF	DIR	Bit masking of individual output cams	
OnCompensation	LREAL	0.0 to 1.0E12	DIR	Activation time Lead time for the switch-on edge	
OffCompensation	LREAL	0.0 to 1.0E12	DIR	Deactivation time Lead time for the switch-off edge	
Hysteresis	LREAL	0.0 to 1.0E12	DIR	Hysteresis value For output cams with reference to actual position, always enter a hysteresis value (> 0.0).	
Cam[1..32].	STUCT				
OnPosition	LREAL	0.0 to 1.0E12	CAL	Start position (distance output cams and time-based output cams)	
Offposition	LREAL	0.0 to 1.0E12	CAL	End position (distance output cam)	
Duration	LREAL	0.001 to 1.0E12	CAL	Switch-on duration (time-based output cam)	
Existent	BOOL	-	CAL	FALSE	Output cam is not used.
				TRUE	Output cam is used

"Interface" tag (cam track) (S7-1500)

The tag structure "<TO>.Interface.<tag name>" contains the configuration of the output properties for the cam track technology object.

Tags

Legend (Page 2094)

Tag	Data type	Values	W	Description	
Interface.	STRUCT				
EnableOutput	BOOL	-	RES	Output cam output at the bit specified under "<TO>.Interface.Address"	
				FALSE	No output
				TRUE	Output
Address	VREF	-	RON	I/O address for digital cam output	

"Units" tag (cam track) (S7-1500)

The tag structure "<TO>.Units.<tag name>" shows the set technological units.

Tags

Legend (Page 2094)

Tag	Data type	Values	W	Description	
Units.	STRUCT				
LengthUnit	UDINT	-	RON	Unit for position	
				1010	m
				1013	mm
				1011	km
				1014	µm
				1015	nm
				1019	in
				1018	ft
				1021	mi
				1004	rad
				1005	°
TimeUnit	UDINT	-	RON	Unit for time	
				1056	ms

"StatusWord" tag (cam track) (S7-1500)

The "<TO>.StatusWord" tag contains the status information of the technology object.

Information on the evaluation of the individual bits (e.g. bit 2 "RestartActive") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tags

Legend (Page 2094)

Tag	Data type	Values	W	Description	
StatusWord	DWORD	-	RON	Status information of the technology object	
Bit 0	-	-	-	"Control"	
				0	Technology object not in operation
				1	Technology object in operation
Bit 1	-	-	-	"Error"	
				0	No error present
				1	Error present
Bit 2	-	-	-	"RestartActive"	
				0	No restart active
				1	Restart active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged"	
				0	Restart tags unchanged
				1	Change to Restart tags. For the changes to be applied, the technology object must be reinitialized.
Bit 4	-	-	-	"OutputInverted"	
				0	Output cam output not inverted
				1	Output cam output inverted
Bit 5	-	-	-	"CommunicationOK" Communication between cam track and output module	
				0	Not established
				1	Established
Bit 6... Bit 31	-	-	-	Reserved	

"ErrorWord" tag (cam track) (S7-1500)

The "<TO>.ErrorWord" tag indicates technology object errors (technology alarms).

Information on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tags

Legend (Page 2094)

Tag	Data type	Values	W	Description
ErrorWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4... Bit 12	-	-	-	Reserved
Bit 13	-	-	-	"PeripheralError" Error accessing a logical address
Bit 14... Bit 31	-	-	-	Reserved

"ErrorDetail" tag (cam track) (S7-1500)

The tag structure "<TO>.ErrorDetail.<tag name>" contains the alarm number and the effective local alarm reaction for the technology alarm that is currently pending on the technology object.

You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.

Tags

Legend (Page 2094)

Tag	Data type	Values	W	Description	
ErrorDetail.	STRUCT				
Number	UDINT	-	RON	Alarm number	
Reaction	DINT	0, 7	RON	Effective alarm reaction	
				0	No reaction
				7	Cam track processing Is complete.

"WarningWord" tag (cam track) (S7-1500)

The "<TO>.WarningWord" tag indicates pending warnings at the technology object.

Information on the evaluation of the individual bits (e.g. bit 1 "ConfigWarning") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tag

Legend (Page 2082)

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemWarning" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigWarning" Configuration error One or several configuration parameters are adjusted internally.
Bit 2	-	-	-	"UserWarning" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4... Bit 12	-	-	-	Reserved
Bit 13	-	-	-	"PeripheralWarning" Error accessing a logical address
Bit 14... Bit 31	-	-	-	Reserved

Tags of the cam technology object

Legend

Tag	Name of the tag
Data type	Data type of the tag
Values	Value range of the tag - minimum value to maximum value If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".

W	Effectiveness of changes in the technology data block	
	DIR	Direct: Values are changed directly and take effect at the start of the next MC-Servo [OB91].
	CAL	At call of Motion Control instruction: Values are changed directly and take effect at the start of the next MC-Servo [OB91] after the call of the corresponding Motion Control instruction in the user program.
	RES	Restart: Changes to the start value in the load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed during runtime of the user program.
Description	Description of the tag	

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

"Point" tag (cam)

The tag structure "<TO>.Point.<tag name>" contains the defined points of the cam.

Tags

Legend (Page 2100)

Tag		Data type	Values	W	Description
Point.		ARRAY [1..1000] OF STRUCT			
	x	LREAL	-1.0E12 to 1.0E12	CAL	Value of the point in the definition range
	y	LREAL	-1.0E12 to 1.0E12	CAL	Value of the point in the range of the function

"ValidPoints" tag (cam)

The tag structure "<TO>.ValidPoint.<tag name>" shows the validity of the defined points of the cam.

Tags

Legend (Page 2100)

Tag	Data type	Values	W	Description	
ValidPoint.	ARRAY [1..1000] OF BOOL				
ValidPoint	BOOL	-	CAL	Indicates whether the defined point is valid.	
				FALSE	Invalid
				TRUE	Valid

"Segment" tag (cam)

The tag structure "<TO>.Segment.<tag name>" contains the defined segments of the cam.

Tags

Legend (Page 2100)

Tag	Data type	Values	W	Description
Segment.	ARRAY [1..50] OF STRUCT			
xmin	LREAL	-1.0E12 to 1.0E12	CAL	Start coordinates of the segment
xmax	LREAL	-1.0E12 to 1.0E12	CAL	End coordinates of the segment
a0	LREAL	-1.0E12 to 1.0E12	CAL	Coefficient A0 for x^0 of the polynomial for the segment
a1	LREAL	-1.0E12 to 1.0E12	CAL	Coefficient A1 for x^1 of the polynomial for the segment
a2	LREAL	-1.0E12 to 1.0E12	CAL	Coefficient A2 for x^2 of the polynomial for the segment
a3	LREAL	-1.0E12 to 1.0E12	CAL	Coefficient A3 for x^3 of the polynomial for the segment
a4	LREAL	-1.0E12 to 1.0E12	CAL	Coefficient A4 for x^4 of the polynomial for the segment
a5	LREAL	-1.0E12 to 1.0E12	CAL	Coefficient A5 for x^5 of the polynomial for the segment
a6	LREAL	-1.0E12 to 1.0E12	CAL	Coefficient A6 for x^6 of the polynomial for the segment
sineAmplitude	LREAL	-1.0E12 to 1.0E12	CAL	Amplitude of the sine element
sinePeriod	LREAL	-1.0E12 to 1.0E12	CAL	Period length of the sine element [rad]
sinePhase	LREAL	-1.0E12 to 1.0E12	CAL	Phase offset of the sine element [rad]

"ValidSegments" tag (cam)

The tag structure "<TO>.ValidSegment.<tag name>" shows the validity of the defined segments of the cam.

Tags

Legend (Page 2100)

Tag	Data type	Values	W	Description	
ValidSegment.	ARRAY [1..50] OF BOOL				
ValidSegment	BOOL	-	CAL	Indicates whether the defined segment is valid.	
				FALSE	Invalid
				TRUE	Valid

"InterpolationSettings" tag (cam)

The tag structure "<TO>.InterpolationSettings.<tag name>" contains the configuration for the interpolation of the cam.

Tags

Legend (Page 2100)

Tag	Data type	Values	W	Description	
InterpolationSettings.	STRUCT				
InterpolationMode	DINT	-1.0E12 to 1.0E12	CAL	Interpolation type	
				0	Linear
				1	C splines
				2	B splines
BoundaryConditions	DINT	-1.0E12 to 1.0E12	CAL	Characteristics of the boundary points	
				0	No profile start or profile end conditions
				1	First derivative equal at profile start and end

"StatusCam" tag (cam)

The tag structure "<TO>.StatusCam.<tag name>" indicates the status of the cam.

Tags

Legend (Page 2100)

Tag	Data type	Values	W	Description
StatusCam.	STRUCT			
StartLeadingValue	LREAL	-1.0E12 to 1.0E12	RON	First defined interpolation point/start of the first segment of the cam (start value of the cam definition range)
EndLeadingValue	LREAL	-1.0E12 to 1.0E12	RON	Last defined interpolation point/end of the last segment of the cam (End value of the definition range of the cam)

"StatusWord" tag (cam)

The "<TO>.StatusWord" tag contains the status information of the technology object.

Information on the evaluation of the individual bits (e.g. bit 4 "CamDataChanged") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tags

Legend (Page 2100)

Tag	Data type	Values	W	Description	
StatusWord	DWORD	-	RON	Status information of the technology object	
Bit 0	-	-	-	"Control" Use status	
				0	Cam not in use
				1	Cam in use
Bit 1	-	-	-	"Error"	
				0	No error present
				1	Error present
Bit 2	-	-	-	"RestartActive"	
				0	No restart active
				1	Restart active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged"	
				0	Restart tags unchanged
				1	Change to Restart tags. For the changes to be applied, the technology object must be reinitialized.
Bit 4	-	-	-	"CamDataChanged"	
				0	No change
				1	The definition range of the cam has changed in the technology data block.
Bit 5	-	-	-	"Interpolated"	
				0	Cam is not interpolated
				1	Cam is interpolated
Bit 6	-	-	-	"InInterpolation"	
				0	Cam not undergoing interpolation
				1	Cam undergoing interpolation
Bit 7... Bit 31	-	-	-	Reserved	

"ErrorWord" tag (cam)

The "<TO>.ErrorWord" tag indicates technology object errors (technology alarms).

Information on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tags

Legend (Page 2100)

Tag	Data type	Values	W	Description
ErrorWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4 … Bit 31	-	-	-	Reserved

"ErrorDetail" tag (cam)

The tag structure "<TO>.ErrorDetail.<tag name>" contains the alarm number and the effective local alarm reaction for the technology alarm that is currently pending on the technology object.

You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2108)" appendix.

Tags

Legend (Page 2100)

Tag	Data type	Values	W	Description	
ErrorDetail.	STRUCT				
Number	UDINT	-	RON	Alarm number	
Reaction	DINT	0, 9	RON	Effective alarm reaction	
				0	No reaction
				9	Terminate processing of the technology object

"WarningWord" tag (cam)

The "<TO>.WarningWord" tag indicates pending warnings at the technology object.

Information on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 1922) section.

Tags

Legend (Page 2100)

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemWarning" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigWarning" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserWarning" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed A Motion Control instruction cannot be executed because necessary requirements have not been met.
Bit 4 … Bit 31	-	-	-	Reserved

Technology alarms (S7-1500)

Overview of the technology alarms (S7-1500)

The following table shows an overview of the technology alarms and the corresponding alarm reactions. When a technology alarm occurs, evaluate the entire indicated alarm text, in order to find the precise cause.

Legend

No.	Number of the technology alarm (corresponds to <TO>.ErrorDetail.Number)
Reaction	Effective alarm reaction (corresponds to <TO>.ErrorDetail.Reaction)
Error bit	Bit that is set in ""<TO>.ErrorWord" when the technology alarm occurs A description of the bits can be found in the appendix (Page 2060).
Warning bit	Bit that is set in "<TO>.WarningWord" when the technology alarm occurs A description of the bits can be found in the appendix (Page 2064).
Restart	To acknowledge the technology alarm, the technology object must be reinitialized (Restart).
Diagnostic buffer	The alarm is entered in the diagnostics buffer.
Alarm text	Displayed alarm test (limited)

List of the technology alarms

No.	Reaction	Error bit	Warning bit	Restart	Diagnostic buffer	Alarm text
101	Remove enable	X1	-	X	X	Configuration error.
102	Remove enable	X15	-	X	X	Drive configuration adaptation error.
103	Remove enable	X15	-	X	X	Encoder configuration adaptation error.
104	Stop with maximum dynamic values	X1	-	-	-	SW limit switch specification error.
105	Remove enable	X1	-	X	X	Drive configuration error.
106	Remove enable	X1	-	-	X	Drive connection configuration error.
107	Remove enable	X1	-	X	X	Encoder configuration error.
108	Remove enable	X1	-	-	X	Encoder connection configuration error.
109	Remove enable	X1	-	X	-	Configuration error.
110	No reaction	-	X1	-	-	Configuration is adjusted internally.
111	No reaction	-	X15	-	X	TO and drive configuration inconsistent.
112	No reaction	-	X15	-	X	TO and encoder configuration inconsistent.
113	Remove enable	X2	-	X	-	Isochronous mode not possible.
201	Remove enable	X0	-	X	X	Internal error.
202	No reaction	X0	-	X	-	Internal configuration error.
203	Remove enable	X0	-	X	-	Internal error.
204	Remove enable	X0	-	-	-	Commissioning error.
304	Stop with emergency stop ramp	X2	-	-	-	Velocity limit is zero.
305	Stop with emergency stop ramp	X2	-	-	-	• Limit value of the acceleration is zero. • Limit value of the deceleration is zero.
306	Stop with emergency stop ramp	X2	-	-	-	Jerk limit is zero.
307	Stop with maximum dynamic values	X2	-	-	X	• Negative numerical value range of the position reached. • Positive numerical value range of the position reached.
308	Remove enable	X2	-	-	X	• Negative numerical value range of the position exceeded. • Positive numerical value range of the position exceeded.
321	Stop with emergency stop ramp	X3	-	-	-	Axis not homed.
322	No reaction	-	X3	-	-	Restart not executed.
323	Remove enable	X3	-	-	-	MC_Home could not be performed.
341	Stop with maximum dynamic values	X10	-	-	-	Error in homing data.
342	Stop with emergency stop ramp	X10	-	-	-	Reference cam/encoder zero mark not found.
343	Remove enable	X1	-	-	-	Homing function not supported by device.
401	Remove enable	X13	-	-	X	Error accessing logical address.

No.	Reaction	Error bit	Warning bit	Restart	Diagnostic buffer	Alarm text
411	Remove enable	X5	-	-	X	Faulty encoder at the logical address.
412	Remove enable	X5	-	-	-	Permitted actual value range exceeded.
421	Remove enable	X4	-	-	X	Faulty drive at the logical address.
431	Remove enable	X7	-	-	X	Faulty communication with device at logical address.
501	No reaction	-	X6	-	-	Programmed velocity is limited.
502	No reaction	-	X6	-	-	• Programmed acceleration is being limited. • Programmed deceleration is being limited.
503	No reaction	-	X6	-	-	Programmed jerk is limited.
504	No reaction	-	X6	-	-	Speed setpoint monitoring active.
511	No reaction	-	X6	-	-	Dynamic limits are violated by the kinematics motion.
521	Remove enable	X11	-	-	-	Following error.
522	No reaction	-	X11	-	-	Warning following error tolerance.
531	Remove enable	X9	-	-	-	• Positive HW limit switch approached. • Negative HW limit switch approached. • Illegal free travel direction of active hardware limit switch. • HW limit switch polarity reversed, free travel not possible. • Both hardware limit switches active, retraction not possible.
533	Stop with maximum dynamic values	X8	-	-	-	• Negative SW limit switch approached. • Positive SW limit switch approached.
534	Remove enable	X8	-	-	-	• Negative SW limit switch is crossed. • Positive SW limit switch is crossed.
541	Remove enable	X12	-	-	-	Position monitoring error.
542	Remove enable	X2	-	-	-	Clamping monitoring error: Axis leaving clamping tolerance window.
550	Track setpoints	X4	-	-	-	Drive-autonomous motion is being executed.
551	No reaction	X2	X6	-	-	Maximum velocity cannot be reached with drive/axis parameters.
552	Remove enable	X15	-	-	-	Encoder adaptation error during ramp-up.
601	Stop with maximum dynamic values	X14	-	-	-	Leading axis is not assigned or defective.
603	Remove enable	X14	-	-	-	Leading axis is not in position-controlled mode.
608	Stop with maximum dynamic values	X14	-	-	-	Error during synchronization.
611	Remove enable	X2	-	-	-	The cam specified in the MC_CamIn.Cam parameter has not been configured or is not available or is not interpolated.
612	Remove enable	X2	-	-	-	Specified cam has not been interpolated.
700	Remove enable	X2	-	-	-	Output cam limiting error.

No.	Reaction	Error bit	Warning bit	Restart	Diagnostic buffer	Alarm text
701	Remove enable	X13	-	-	-	I/O output error.
702	Remove enable	X2	-	-	-	Position value valid.
703	Remove enable	X2	-	-	-	Cam track data faulty.
704	Remove enable	X2	-	-	-	Output cam data faulty.
750	Remove enable	X2	-	-	-	Measuring job not possible during homing of assigned axis.
752	Remove enable	X2	-	-	-	Validity range of measuring job not recognized.
753	Remove enable	X2	-	-	-	Only one measuring input can access an encoder at a time.
754	Remove enable	X2	-	-	-	Measuring input configuration in external device is not correct.
755	Remove enable	X13	-	-	-	Measuring job not possible.
758	No reaction	X2	-	-	-	A measuring edge was not evaluated.

See also

Technology alarms (Page 1961)

Technology alarms 101-113 (S7-1500)

Technology alarm 101

Alarm reaction: Remove enable

Restart: Required

Alarm text	Remedy
Configuration error.	
Value in <tag> not allowed.	Adjust the specified value.
Faulty load gear factors.	Adjust the load gear factors in the "<TO>.LoadGear.Numerator" and/or "<TO>.LoadGear.Denominator" parameters.
At least one encoder required. Sensor[].existent	Configure at least one encoder.
Sensor[1] must be configured for DSC.	Configure Sensor[1]
Values in Sensor.Parameter.FineResolutionXist1 and P979 are not identical.	Set the identical fine resolution on the technology as on the drive.
Controller parameter incorrect.	Adjust the value of the "<TO>.PositionController.Kv" parameter.
PROFIBUS parameter assignment is inconsistent; sum Ti and To greater than send clock.	Adjust the send clock in the hardware configuration.
Drive or drive telegram type or encoder not suitable for DSC.	Check whether the drive can be operated with DSC and adjust the drive telegram if required.
TimeOut parameter outside of limits.	Set the monitoring time of the axis control panel to a valid value.
Simulation.Mode parameter outside of limits.	Set the parameter to a valid value.
Telegram in Actor.Interface.AddressIn and AddressOut are not identical.	Set the identical drive telegram type for sending and receiving direction.
Illegal combination for referencing data incremental. encoder.	Check the active and passive homing settings.
Telegram in Sensor.Interface.AddressIn and AddressOut are not identical.	Set the identical encoder telegram type for sending and receiving direction.

Technology alarm 102

Alarm reaction: Remove enable

Restart: Required

Alarm text	Remedy
Drive configuration adaptation error.	
Drive is not assigned to a SINAMICS device.	The drive adaptation is only available for SINAMICS drives.
Drive is not interconnected directly to I/O area.	During configuration of the axis, the logical addresses were set to a data block or bit memory, for example. The adaptation is only possible when the encoder has been directly interconnected to an I/O area.
Adaptation canceled due to insufficient resources.	Check whether your device supports acyclic data communication according to PROFIdrive.
Parameter does not exist, value unreadable or invalid.	
Maximum speed	
Maximum torque/force (P1520)	
Maximum torque/force (P1521)	
Torque resolution	
Rated speed	
Rated torque	
Motor type	

Technology alarm 103

Alarm reaction: Remove enable

Restart: Required

Alarm text	Remedy
Encoder configuration adaptation error.	
Encoder is not assigned to a SINAMICS device.	The encoder adaptation is only available for SINAMICS devices and external Siemens encoders.
Encoder is not interconnected directly to I/O area.	During configuration of the axis, the logical addresses were set to a data block or bit memory address area, for example. The adaptation is only possible when the encoder has been directly interconnected to an I/O area.
Adaptation canceled due to insufficient resources.	Check whether your device supports acyclic data communication according to PROFIdrive.
Parameter does not exist, value unreadable or invalid.	
Encoder system	
Encoder resolution	
Encoder fine resolution Gx_XIST1	
Encoder fine resolution Gx_XIST2	
Encoder revolutions	

Technology alarm 104

Alarm reaction: Stop with maximum dynamic values

Restart: Not required

Alarm text	Remedy
SW limit switch specification error.	
Neg. SW limit switch greater than pos. SW limit switch.	Change the position of the software limit switches.

Technology alarm 105

Alarm reaction: Remove enable

Restart: Required

Alarm text	Remedy
Drive configuration error.	
HW Configuration.	• Connect a suitable device. • Check the device (I/Os). • Check the topology of the project. • Compare the device configuration and the configuration of the technology object. • Contact customer service.
The TO needs a smaller servo cycle clock.	
Error in internal communication.	• Check the project for consistency and reload the project into the controller. • Contact customer service.
Address for drive data does not exist in project.	Check the project for consistency and reload the project into the controller.

Technology alarm 106

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Drive connection configuration error.	
System has no communication with drive.	Internal system error. • Check the project for consistency and reload the project into the controller. • Contact customer service.
Drive not initialized during ramp-up.	• Ensure that the communication between the controller and drive is established. To do this, evaluate the "<TO>.StatusDrive.CommunicationOK" parameter before enabling the axis. • To enable a technology object, the drive initialization must be complete. Trigger the job again later.

Technology alarm 107

Alarm reaction: Remove enable

Restart: Required

Alarm text	Remedy
Encoder configuration error.	
HW Configuration	• Connect a suitable device. • Check the device (I/Os). • Check the topology of the project. • Compare the device configuration and the configuration of the technology object. • Contact customer service.
The TO needs a smaller servo cycle clock.	
Error internal communication.	• Check the project for consistency and reload the project into the controller. • Contact customer service.

Technology alarm 108

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Encoder connection configuration error.	
System without communication to encoder.	Internal system error. • Check the project for consistency and reload the project into the controller. • Contact customer service.
Encoder not initialized during ramp-up.	• Ensure that the communication between the controller and encoder is established. To do this, evaluate the "<TO>.StatusSensor[1..4].CommunicationOK" parameter before enabling the axis and also check if the status of the encoder actual value is "<TO>.StatusSensor[1..4].State" = VALID (2). • To enable a technology object, the encoder initialization must be complete. Trigger the job again later.
Encoder data address missing in project.	Check the project for consistency and reload the project into the controller.

Technology alarm 109

Alarm reaction: Remove enable

Restart: Required

Alarm text	Remedy
Configuration error.	
Neg. HW limit switch.	• Connect a suitable device. • Check the device (I/Os). • Check the topology of the project. • Compare the device configuration and the configuration of the technology object. • Contact customer service.
Pos. HW limit switch	
Reference cam "Active homing".	
Reference cam "Passive homing".	
Enable bit for the analog drive interface.	
DriveReady bit of the analog drive interface.	
Measurement sensing input is faulty.	
Output cam output faulty.	

Technology alarm 110

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
Configuration is adjusted internally.	
Actor.DriveParameter.MaxSpeed is limited.	• Correct the reference value in the drive and in the configuration of the technology object to "<TO>.Actor.MaxSpeed" / 2. • With analog drive connection, correct the reference value in the drive and in the configuration of the technology object to "<TO>.Actor.MaxSpeed" / 1.17. • The value can be set in the drive, for example, in p2000 = p1082.
PositioningMonitoring.ToleranceTime is limited.	Change the configuration data.
DynamicDefaults.EmergencyDeceleration is limited.	
DriveParameter.ReferenceTorque too small.	

Technology alarm 111

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
TO and drive configuration inconsistent.	
Different telegram.	Match the telegram configuration for the technology object with the telegram configuration in the drive. (P922 in drive)
Incompatible torque resolution.	Adjust the high torque resolution for the drive.
Application cycle of the drive and servo cycle clock are not the same.	Adjust the application cycle of the drive in the device configuration for the PROFIBUS drive.
Application cycle of the drive and processing cycle of the TO are not the same.	
Linear motor configured.	Set round-frame motor (P300) in the drive.

Technology alarm 112

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
TO and encoder configuration inconsistent.	
Different telegram type.	Match the telegram configuration for the technology object with the telegram configuration in the encoder.
Encoder is not an absolute encoder.	Configure the encoder for the technology object as an incremental encoder.
Application cycle of the encoder and servo cycle clock are not the same.	Adjust the application cycle of the encoder in the device configuration for the PROFIBUS encoder.
Application cycle of the encoder and processing cycle of the TO are not the same.	

Technology alarm 113

Alarm reaction: Remove enable

Restart: Required

Alarm text	Remedy
Isochronous mode not possible.	• The configured output for the cam or cam track technology object or the input for the technology object measuring input cannot be used in isochronous mode. Configure the I/O in the device configuration as isochronous I/O. • Make sure that the organization block MC_Servo [OB91] is called synchronously with the bus system.

Technology alarms 201-204 (S7-1500)

Technology alarm 201

Alarm reaction: Remove enable

Restart: Required

Alarm text	Solution
Internal error.	Contact customer service.

Technology alarm 202

Alarm reaction: No reaction

Restart: Required

Alarm text	Solution
Internal configuration error.	Contact customer service.

Technology alarm 203

Alarm reaction: Remove enable

Restart: Required

Alarm text	Solution
Internal error.	Contact customer service.

Technology alarm 204

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Solution
Commissioning error.	
Connection to the TIA Portal interrupted.	Check the connection properties.

Technology alarms 304-343 (S7-1500)

Technology alarm 304

Alarm reaction: Stop with emergency stop ramp

Restart: Not required

Alarm text	Remedy
Velocity limit is zero.	Enter a non-zero value for the maximum velocity (DynamicLimits.MaxVelocity) in the dynamic limits.

Technology alarm 305

Alarm reaction: Stop with emergency stop ramp

Restart: Not required

Alarm text	Remedy
Acceleration/deceleration limit is zero.	
Acceleration	Enter a non-zero value for the maximum acceleration (DynamicLimits.MaxAcceleration) in the dynamic limits.
Deceleration	Enter a non-zero value for the maximum deceleration (DynamicLimits.MaxDeceleration) in the dynamic limits.

Technology alarm 306

Alarm reaction: Stop with emergency stop ramp

Restart: Not required

Alarm text	Remedy
Jerk limit is zero.	Enter a non-zero value for the maximum jerk (DynamicLimits.MaxJerk) in the dynamic limits.

Technology alarm 307

Alarm reaction: Stop with maximum dynamic values

Restart: Not required

Alarm text	Remedy
Negative/positive numerical value range of the position reached.	
Negative	Enable the "Modulo" setting for the technology object.
Positive	

Technology alarm 308

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Negative/positive numerical value range of the position exceeded.	
Negative	Enable the "Modulo" setting for the technology object.
Positive	

Technology alarm 321

Alarm reaction: Stop with emergency stop ramp

Restart: Not required

Alarm text	Remedy
Axis not homed.	To perform an absolute positioning motion, you must home the technology object.

Technology alarm 322

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
Restart not executed.	
The technology object is not ready for restart.	Download the project again.
The condition for restart of the technology object is not satisfied.	Disable the technology object.

Technology alarm 323

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
MC_Home could not be performed.	• Enable the "Modulo" setting for the technology object. • Adjust the position value for use of the Motion Control instruction "MC_Home".

Technology alarm 341

Alarm reaction: Stop with maximum dynamic values

Restart: Not required

Alarm text	Remedy
Error in homing data.	
Approach velocity is zero.	Check the configuration for homing (Homing.Approach-Velocity).
Homing velocity is zero.	Check the configuration for homing (Homing.Referencing-Velocity).

Technology alarm 342

Alarm reaction: Stop with emergency stop ramp

Restart: Not required

Alarm text	Remedy
Reference cam/encoder zero mark not found.	The reference cam configured for homing was not found in the traversing range of the axis.

Technology alarm 343

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Homing function not supported by device.	Configure a reference switch input for the pulse generator output used in the properties of the C-CPU. ("Pulse generators (PTO/PWM) > PTO[n]/PWN[n] > Hardware inputs/outputs") When homing across a zero mark, the CPU transfers the reference switch input as zero mark.

Technology alarms 401-431 (S7-1500)

Technology alarm 401

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Solution
Error accessing logical address.	
Invalid address.	• Connect a suitable device. • Check the device (I/Os). • Check the topology of the project. • Compare the device configuration and the configuration of the technology object. • Contact customer service.
Input address is invalid.	
Output address is invalid.	

Technology alarm 411

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Solution
Faulty encoder at the logical address.	
Alarm message from encoder.	Check the function, connections and I/Os of the encoder.
HW error encoder.	
Encoder dirty.	
Read error encoder absolute value.	Compare the encoder type in the drive or encoder parameter P979 with the configuration data of the technology object.
Zero mark monitoring encoder.	Encoder signals error in zero mark monitoring (fault code 0x0002 in Gx_XIST2, see PROFIdrive profile).
Encoder in Parking state.	• Search for the cause of the error in the connected drive or encoder. • Check whether the alarm was possibly triggered by a commissioning action involving the drive or encoder.

Technology alarm 412

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Permitted actual value range exceeded.	
Positive.	Home the axis/encoder in a valid actual value range.
Negative.	
Modulo length.	Adjust the modulo length to the utilized encoder.

Technology alarm 421

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Solution
Faulty encoder at the logical address.	
Alarm message from drive.	• Check the functions and connections of the drive. • Enable and acknowledge safety function in the drive. • In the case of analog connected axes, check if the "<TO>.StatusDrive.InOperation" tag = TRUE.
No drive control required.	
Drive has shut down.	
Drive enable not possible.	

Technology alarm 431

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Solution
Faulty communication with device at logical address.	
Drive failed.	Check the function, connections and I/Os of the drive .
Signs of life of drive faulty.	• Check the function, connections and I/Os of the drive . • Compare the clock parameters in the device configuration (PROFIBUS line, slave OM for drive or encoder) and the execution system. Tmapc and servo must be set to the same cycle time. (Incorrect parameter assignment is indicated by reason 0x0080.) • If you call the application cycle of the MC-Servo [OB91] reduced to the send clock of a PROFINET IO system and the technology alarm 431 (Signs of life of drive faulty) is repeatedly shown, increase the update time of the send clock.
Encoder failed.	Check the function, connections and I/Os of the encoder.
Signs of life of encoder faulty.	• Check the function, connections and I/Os of the encoder. • Compare the clock parameters in the device configuration (PROFIBUS line, slave OM for drive or encoder) and the execution system. Tmapc and servo must be set to the same cycle time.

Technology alarms 501-552 (S7-1500)

Technology alarm 501

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
Programmed velocity is limited.	• Check the value for the velocity of the Motion Control instruction. • Check the configuration of the dynamic limits.

Technology alarm 502

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
Programmed acceleration/deceleration is being limited.	
Acceleration	• Check the value for the acceleration of the Motion Control instruction. • Check the configuration of the dynamic limits.
Deceleration	• Check the value for the deceleration of the Motion Control instruction. • Check the configuration of the dynamic limits.

Technology alarm 503

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
Programmed jerk is limited.	• Check the value for the jerk of the Motion Control instruction. • Check the configuration of the dynamic limits.

Technology alarm 504

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
Speed setpoint monitoring active.	• Check the mechanical configuration. • Check the encoder connection. • Check the configuration of the speed setpoint interface. • Check the configuration of the control loop. • Check the value for the maximum velocity (<TO>.DynamicLimits.MaxVelocity).

Technology alarm 511

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
Dynamic limits are violated by the kinematics motion.	
Velocity	Reduce the velocity of the kinematics motion.
Acceleration	Reduce the acceleration of the kinematics motion.
Deceleration	Reduce the deceleration of the kinematics motion.

Technology alarm 521

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Following error.	• Check the configuration of the control loop. • Check the direction signal of the encoder. • Check the configuration of the following error monitoring.

Technology alarm 522

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
Warning following error tolerance.	• Check the configuration of the control loop. • Check the direction signal of the encoder. • Check the configuration of the following error monitoring.

Technology alarm 531

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Pos. HW limit switch reached.	Acknowledge the alarm. After the acknowledgment, motions in the negative direction are allowed.
Neg. HW limit switch reached.	Acknowledge the alarm. After the acknowledgment, motions in the positive direction are allowed.
Illegal free travel direction of active hardware limit switch.	The programmed direction of movement is disabled due to the active hardware limit switch. Retract the axis in the opposite direction.
HW limit switch polarity reversed, free travel not possible.	• Check the mechanical configuration of the hardware limit switch. • Check the limit switches. • The error can be acknowledged by switching the controller off and on or using "MC_Reset" with "Restart" = TRUE.
Both hardware limit switches active, retraction not possible.	

Technology alarm 533

Alarm reaction: Stop with maximum dynamic values

Restart: Not required

Alarm text	Remedy
Negative/positive SW limit switch approached.	
Negative	The software limit switch was approached from an active motion. Move the axis away from the software limit switch in the positive direction.
Positive	The software limit switch was approached from an active motion. Move the axis away from the software limit switch in the negative direction.

Technology alarm 534

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Negative/positive SW limit switch is crossed.	
Negative	The software limit switch was overtraveled. Acknowledge the alarm. After the acknowledgment, motions in the positive direction are allowed.
Positive	The software limit switch was overtraveled. Acknowledge the alarm. After the acknowledgment, motions in the negative direction are allowed.

Technology alarm 541

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Position monitoring error.	
Target range not reached.	The target range was not reached within the tolerance time. • Check the configuration of the position monitoring. • Check the configuration of the control loop.
Exit target range again.	The target range was exited within the minimum dwell time. • Check the configuration of the position monitoring. • Check the configuration of the control loop.

Technology alarm 542

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Clamping monitoring error: Axis leaving clamping tolerance window.	The axis has executed a motion greater than the permissible tolerance at the fixed stop. Check whether the fixed stop has broken away.

Technology alarm 550

Alarm reaction: Track setpoints

Restart: Not required

Alarm text	Remedy
Drive-autonomous motion is being executed.	The drive is performing a motion that was not specified by the technology object. Check if a safety function is active in the drive.

Technology alarm 551

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
Maximum velocity cannot be reached with drive/axis parameters.	The configured maximum velocity cannot be reached with the configured mechanics of the axis. Check the configuration of the mechanics and the set reference speed.

Technology alarm 552

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Encoder adaptation error during ramp-up.	
Encoder is not assigned to a SINAMICS device.	• The operationally active encoder could not be adapted. Other encoders that can be used are configured. Use the encoder switch (MC_SetSensor). • The encoder set as the operationally active encoder could not be adapted. • Specify a different sensor for the initialization of the technology object.
Encoder is not interconnected directly to I/O area.	During configuration of the axis, the logical addresses were set to a data block or bit memory address area, for example. The adaptation is only possible when the encoder has been directly interconnected to an I/O area.
Adaptation canceled due to insufficient resources.	Check whether your device supports acyclic data communication according to PROFIdrive.
Parameter does not exist, value unreadable or invalid.	
Encoder system	
Encoder resolution	
Encoder fine resolution	
Encoder revolutions	

Technology alarms 601-618 (S7-1500)

Technology alarm 601

Alarm reaction: Stop with maximum dynamic values

Restart: Not required

Alarm text	Solution
Leading axis is not assigned or defective.	Configure the possible leading value axes for the following axis under "Configuration > Leading value interconnections".

Technology alarm 603

Alarm reaction: No reaction

Restart: Not required

Alarm text	Solution
Leading axis is not in position-controlled mode.	The following axis must be operated in position-controlled mode for synchronous operation functionality.

Technology alarm 608

Alarm reaction: Stop with maximum dynamic values

Restart: Not required

Alarm text	Solution
Error during synchronization.	Prevent a reversing leading value motion during the synchronization operation.

Technology alarm 611

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Solution
The cam specified in the MC_CamIn.Cam parameter has not been configured or is not available.	Configure and interpolate the cam. Restart the job.

Technology alarm 612

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Solution
Specified cam has not been interpolated.	Interpolate the cam used for camming with the Motion Control instruction "MC_InterpolateCam".

Technology alarms 700-758 (S7-1500)

Technology alarm 700

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Output cam limiting error.	
Cam position: OnPosition	The position for the "OnPosition" parameter could not be calculated. Invalid positions (e.g. "OnPosition" > "OffPosition") were calculated due to lead times. The output cam cannot be switched due to the axis dynamics and compensation times.
Cam position: OffPosition	The position for the "OffPosition" parameter could not be calculated. Invalid positions (e.g. "OffPosition" > "OnPosition") were calculated due to lead times. The output cam cannot be switched due to the axis dynamics and compensation times.

Technology alarm 701

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
I/O output error.	The digital output for the output cam or cam track technology object cannot be addressed. Download the device configuration again.

Technology alarm 702

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Position value valid.	• A Motion Control job "MC_Reset" is being executed on the axis. Wait until the technology object restart is complete. • The encoder values are invalid due to an encoder error. Check the encoder and adjust the configuration if necessary.

Technology alarm 703

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Output cam data faulty.	
Output cam: Output cam number	Check the configuration of the relevant output cam in the cam track and adjust the values if necessary. Examples of a correct configuration: • "<TO>.Parameter.Cam[].OnPosition" < "<TO>.Parameter.Cam[].OffPosition" • "<TO>.Parameter.Cam[].Duration" > "<TO>.Parameter.OffCompensation" - "<TO>.Parameter.OnCompensation"

Technology alarm 704

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Output cam data faulty.	Check the configuration of the output cam and adjust the values if necessary. Examples of a correct configuration: • "MC_OutputCam.OnPosition" < "MC_OutputCam.OffPosition" • "MC_OutputCam.Duration" > "<TO>.Parameter.OffCompensation" - "<TO>.Parameter.OnCompensation"

Technology alarm 750

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Measuring job not possible during homing of assigned axis.	Do not use the motion instructions "MC_Home" and "MC_MeasuringInput" simultaneously.

Technology alarm 752

Alarm reaction: No reaction

Restart: Not required

Alarm text	Remedy
Validity range of measuring job not recognized.	The measuring range specified in Motion Control instruction "MC_MeasuringInput" was not recognized. Adjust the measuring range.

Technology alarm 753

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Only one measuring input can access an encoder at a time.	Use only one Motion Control instruction "MC_MeasurinInput" for an encoder.

Technology alarm 754

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Measuring input configuration in external device is not correct.	Check the configuration of the measuring inputs on the external device.

Technology alarm 755

Alarm reaction: Remove enable

Restart: Not required

Alarm text	Remedy
Measuring job not possible.	
Device has reported an error.	The measurement was aborted with error. Check the measuring input functionality in the utilized device
Cyclic measuring is not possible with telegram 39x.	• Use the Motion Control instruction "MC_MeasuringInput" for starting a one-time measurement. • Cyclic measuring is only possible when measuring using TM Timer DIDQ. Change the configuration of the measuring input type to "TM Timer DIDQ".

Technology alarm 758

Alarm reaction: None

Restart: Not required

Alarm text	Remedy
A measuring edge was not evaluated.	An edge was already detected at the input of the measuring input even though the module was not yet ready. The measured value is provided at the next edge.

Error ID for Motion Control instructions (S7-1500)

Errors in Motion Control instructions are signaled using the parameters "Error" and "ErrorID".

Under the following conditions, "Error" = TRUE and "ErrorID" = 16#8xxx are indicated for the Motion Control instruction:

- Illegal status of the technology object, which prevents the execution of the job.
- Illegal parameter assignment of the Motion Control instruction, which prevents the execution of the job.
- As a result of the alarm reaction for a technology object error.

The following tables list all "ErrorIDs" that can be indicated for Motion Control instructions. In addition to the cause of the error, solutions for eliminating the error are also listed:

16#0000 - 16#800F

ErrorID	Description	Remedy
16#0000	No error	-
16#8001	A technology alarm (technology object error) occurred while processing the Motion Control instruction.	In the technology data block, an error message is output at the "ErrorDetail.Number" tag. You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2107)" appendix.

ErrorID	Description	Remedy
16#8002	Illegal specification of the technology object	• Check the specification of the technology object for the "Axis", "Master", "SlaveOutputCamCamTrackMeasuringInput" or "Cam" parameter. • You can use a kinematics technology object only for the "AxesGroup" parameter. • With "MC_MeasuringInputCyclic": Specify a valid measuring input type for parameter "MeasuringInputType".
16#8003	Illegal velocity specification	Specify a permissible value for the velocity for parameter "Velocity".
16#8004	Illegal acceleration specification	Specify a permissible value for the acceleration for parameter "Acceleration".
16#8005	Illegal deceleration specification	Specify a permissible value for the deceleration for parameter "Deceleration".
16#8006	Illegal jerk specification	Specify a permissible value for the jerk for parameter "Jerk".
16#8007	Illegal direction specification	Specify a permissible value for the rotation direction for parameter "Direction" or "SyncDirection".
	Invalid entry Both the "JogForward" and "JogBackward" parameters are set to TRUE at the same time. The axis is braked at the last valid deceleration.	Reset both the "JogForward" parameter and the "JogBackward" parameter.
16#8008	Invalid distance specification	Set a valid distance value at parameter "Distance".
16#8009	Invalid position specification	Set a valid position value at parameter "Position".
16#800A	Illegal operating mode	Specify a permissible operating mode for parameter "Mode".
16#800B	Illegal stop mode specifications	Specify a permissible value for the stop mode at the "StopMode" parameter.
16#800C	Only one instance of the instruction per technology object is allowed.	The instruction is called at multiple points in the user program with identical value for parameter "Axis", "Master", "Slave" or "Cam". Ensure that only one instruction with the value for parameter "Axis", "Master", "Slave" or "Cam" is called.
16#800D	The job is not permitted in the current state. "Restart" is executed.	While a "Restart" is being performed, the technology object cannot perform any jobs. Wait until the "Restart" of the technology object is complete.
16#800E	If the technology object is enabled, a "Restart" is not possible.	Before a "Restart", deactivate the technology object with "MC_Power.Enable"FALSE.
16#800F	The job cannot be executed because the technology object is locked.	Enable the technology object with "MC_Power.Enable" = TRUE. Restart the job.

16#8010 - 16#802F

ErrorID	Description	Remedy
16#8010	Invalid homing mode for incremental encoder	Absolute value adjustment is not possible with an incremental encoder ("Mode" = 6, 7). Start a homing process for an incremental encoder using parameter "Mode" = 0, 1, 2, 3, 5, 8, 10.
16#8011	Invalid homing mode for absolute encoder	Passive and active homing ("Mode" = 2, 3, 5, 8, 10) are not possible for an absolute value encoder. Start a homing process for an absolute encoder using parameter "Mode" = 0, 1, 6, 7.
16#8012	The job cannot be executed because the axis control panel is active.	Return master control to your user program. Restart the job.
16#8013	The online connection between the CPU and the TIA Portal is down.	Check the online connection to the CPU.
16#8014	No internal job memory available.	The maximum possible number of Motion Control job has been reached. Reduce the number of jobs to be executed (parameter "Execute" = FALSE).
16#8015	Error acknowledgment with "MC_Reset" is not possible. Error in the configuration of the technology object.	Check the configuration of the technology object.
16#8016	The actual values are not valid.	To execute a "MC_Home" or positioning job, the actual values must be valid. Check the status of the actual values. The "<TO>.StatusSensor[n].State" tag of the technology object must show the value 2 (valid).
16#8017	Illegal value for gear ratio numerator	Specify a permissible value for the gear ratio numerator for parameter "RatioNumerator". Permitted integer values: -2147483648 to 2147483647 (value 0 not permitted)
16#8018	Illegal value for gear ratio denominator	Specify a permissible value for the gear ratio denominator for parameter "RatioDenominator". Permitted integer values: 1 to 2147483647
16#8019	Job cannot be executed. The specified following axis is the original leading value for the synchronous operation chain.	Recursive interconnections are not possible. A leading axis cannot be interconnected as a following axis to its own leading value. Specify a permissible following axis for parameter "Slave".
16#8021	Illegal value for shift of the leading value range	Specify a permissible value for the shift of the leading value range for parameter "MasterOffset".
16#8022	Illegal value for shift of the following value range	Specify a permissible value for the shift of the leading value range for parameter "SlaveOffset".
16#8023	Illegal value for scaling of the leading value range	Specify a permissible value for the scaling of the leading value range for parameter "MasterScaling".
16#8024	Illegal value for scaling of the following value range	Specify a permissible value for the scaling of the following value range for parameter "SlaveScaling".

ErrorID	Description	Remedy
16#8026	Illegal value for leading value distance	Specify a permissible value for the leading value distance for parameter "MasterStartDistance".
16#8027	Illegal value for use of cam	Specify a permissible value for cyclic/acyclic use of the cam for parameter "ApplicationMode".

16#8030 - 16#807F

ErrorID	Description	Remedy
16#8034	Illegal value for synchronous position of the leading axis	Specify a permissible value for the synchronous position of the leading axis for parameter "MasterSyncPosition".
16#8035	Illegal value for synchronous position of the following axis	Specify a permissible value for the synchronous position of the following axis for parameter "SlaveSyncPosition".
16#8036	Illegal value for type of synchronization	Specify a permissible value for the type of synchronization for parameter "SyncProfileReference".
16#8040	Illegal value for start position of output cam	Specify a permissible value for the start position of the output cam for parameter "OnPosition".
16#8041	Illegal value for end position of distance output cam	Specify a permissible value for the end position of the distance output cam for parameter "OffPosition".
16#8042	Illegal value for switch-on duration of time-based output cam	Specify a permissible value for the switch-on duration of the time-based output cam for parameter "Duration".
16#8043	Illegal value for force/torque limiting	Specify a value within the permissible range at the "Limit" parameter. Permitted integer values: -2147483648 to 2147483648
16#8044	The axis is not configured for torque reduction.	Select drive telegram 102, 103, 105 or 106
16#8045	The job cannot be executed because a job for traveling to fixed stop is active.	Switchover to non-position-controlled mode is not possible during active travel to fixed stop.
16#8046	The "MC_TorqueLimiting" job cannot be deactivated in the "InClamping" state.	Retract the axis and deactivate "MC_TorqueLimiting".
16#8047	The motion results in a fixed stop.	Only motions away from the fixed stop are permitted.
16#804A	Illegal value for additive torque setpoint	Specify a permissible value for the additive torque setpoint at the "Value" parameter.
16#804B	Illegal value for torque high limit	Specify a permissible value for the high limit of the torque at the "UpperLimit" parameter.
16#804C	Illegal value for torque low limit	Specify a permissible value for the low limit of the torque at the "LowerLimit" parameter.
16#804D	The value of the high limit of the torque is less than or equal to the value of the low limit of the torque.	Adapt the values of the "UpperLimit" and "LowerLimit" parameters so that the high limit of the torque is greater than the value of the low limit of the torque.
16#804E	The job cannot be executed because a "MC_TorqueLimiting" job is active.	Stop the force / torque limit or fixed stop detection. Restart the "MC_TorqueRange" job.
	The job cannot be executed because a "MC_TorqueRange" job is active.	Exit the setting of the high and low torque limits. Restart the "MC_TorqueLimiting" job.
16#804F	The axis is not configured for additional torque values.	Use supplemental telegram 750.

ErrorID	Description	Remedy
16#8050	Illegal encoder number	Specify a permissible number of the new encoder (1 to 4) for parameter "MC_SetSensor.Sensor".
16#8051	Illegal number of the reference encoder	Specify a permissible number of the reference encoder for parameter "MC_SetSensor.ReferenceSensor".
16#8062	Illegal approach value	Specify a permissible approach value for the searched for leading value for parameter "ApproachLeadingValue".
16#8063	A valid mapping to the definition range (leading values) does not exist for the specified following value.	Specify a permissible following value for parameter "FollowingValue".
16#8064	A valid mapping to the range of the function (following values) does not exist for the specified leading value.	Specify a permissible leading value for parameter "LeadingValue".
16#8070	Illegal value for leading value shift	Specify a permissible value for the leading value shift for parameter "PhaseShift".
16#8071	The job cannot be executed because the axis is not in position-controlled mode.	Activate position-controlled mode.
16#8074	The job cannot be executed because a "MC_Home" job is active.	During active or passive homing, an encoder switchover is rejected. Wait until the "MC_Home" job is complete. Restart the job.
16#8075	The job cannot be executed because no synchronization operation is active on the axis.	Switch on the synchronous operation function. Restart the job.
16#8076	The job cannot be executed because synchronization is being simulated at the specified axis.	End the simulation of the synchronous operation. Restart the job.

16#80A0 - 16#8FFF

ErrorID	Description	Remedy
16#80A1	The order cannot be executed because a synchronous operation job is active.	A "MC_Home" job on a following axis is not executed when a "MC_CamIn" or "MC_GearInPos" job is active. Exit the synchronous operation job. Restart the job.
16#80A2	The measuring range is invalid with the configured modulo axis settings.	Check and adjust the measuring input and adjust the measuring range positions, if necessary.
16#80A3	The measuring input job via PROFIdrive telegram could not be started because a homing job is active.	Simultaneous execution of a homing job and a measuring input job via PROFIdrive telegram is not possible. Wait until the homing job has ended. Restart the measuring job via PROFIdrive telegram.
16#80A5	Illegal value for start position of measuring range	Specify a permissible value for the start position of the measuring range for parameter "MC_MeasuringInput.StartPosition" or MC_MeasuringInputCyclic.StartPosition.
16#80A6	Illegal value for end position of measuring range	Specify a permissible value for the end position of the measuring range for parameter "MC_MeasuringInput.EndPosition" or MC_MeasuringInputCyclic.EndPosition.
16#80A7	A measurement is performed when measuring with the measuring range, but the calculated position is outside the specified measuring range. The measured value is discarded.	Check and adjust the measuring input and adjust the measuring range positions, if necessary.

ErrorID	Description	Remedy
16#80A8	The job cannot be executed because camming is active on the axis.	The Motion Control instructions "MC_PhasingRelative" and "MC_PhasingAbsolute" can only be applied to active gearing with "MC_GearIn" or "MC_GearInPos" ("MC_GearIn.InGear" = TRUE or "MC_GearInPos.InSync" = TRUE).
16#80A9	The job cannot be executed because the following axis is synchronized ("MC_GearInPos.StartSync" = TRUE) or a kinematics motion is active.	The Motion Control instructions "MC_PhasingRelative" and "MC_PhasingAbsolute" can only be applied to active gearing with "MC_GearIn" or "MC_GearInPos" ("MC_GearIn.InGear" = TRUE or "MC_GearInPos.InSync" = TRUE).
16#80AA	The cam contains no points or segments and cannot be interpolated.	Fill the cam with points/segments. Restart the job.
16#80AB	The cam is currently being used and cannot be interpolated.	End the current use of the cam. Restart the job.
16#80AC	The cam contains incorrect points or segments and cannot be interpolated. (for example, the cam contains only one point.)	Fill the cam with permissible points/segments. Restart the job.
16#80AD	The specified synchronous position is outside the definition range of the cam.	Specify a permissible synchronous position for parameter "MasterSyncPosition". Restart the job.
16#80AE	The job cannot be executed because a kinematic motion is active.	End the current kinematic motion. Restart the job.
16#8FFF	Unspecified error	Contact your local Siemens representative or support center. You can find your contact at Industry Automation and Drive Technologies at: http://www.siemens.com/automation/partner (http://www.siemens.com/automation/partner)

See also

Errors in Motion Control instructions (Page 1965)

"MC_Power" function diagrams (S7-1500)

Drive connection via PROFIdrive (S7-1500)

"StopMode" = 0, 2 (S7-1500)

Function chart: Enabling a technology object and disabling with "StopMode" = 0, 2

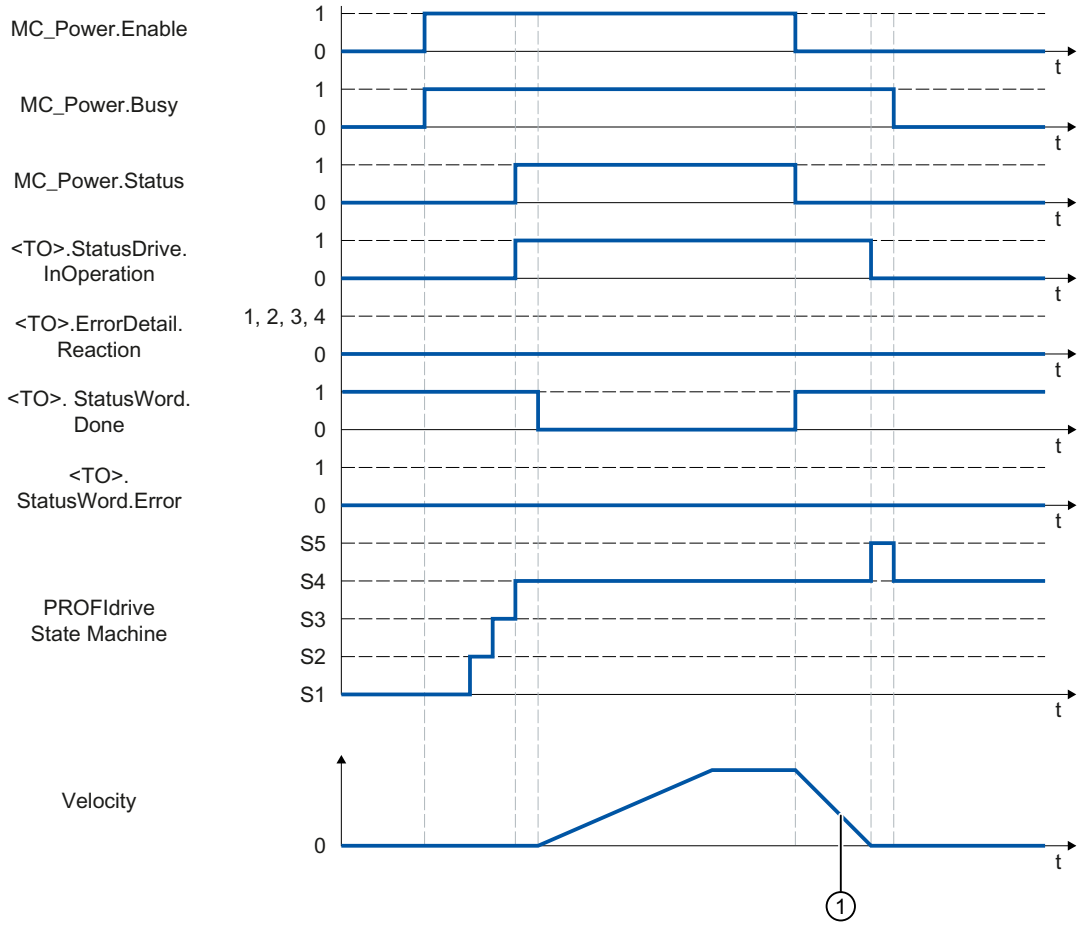

①
- "StopMode" = 0
  The axis is braked with the configured emergency stop deceleration.
- "StopMode" = 2
  The axis decelerates with the configured maximum deceleration.

"StopMode" = 1 (S7-1500)

Function chart: Enabling a technology object and disabling with "StopMode" = 1

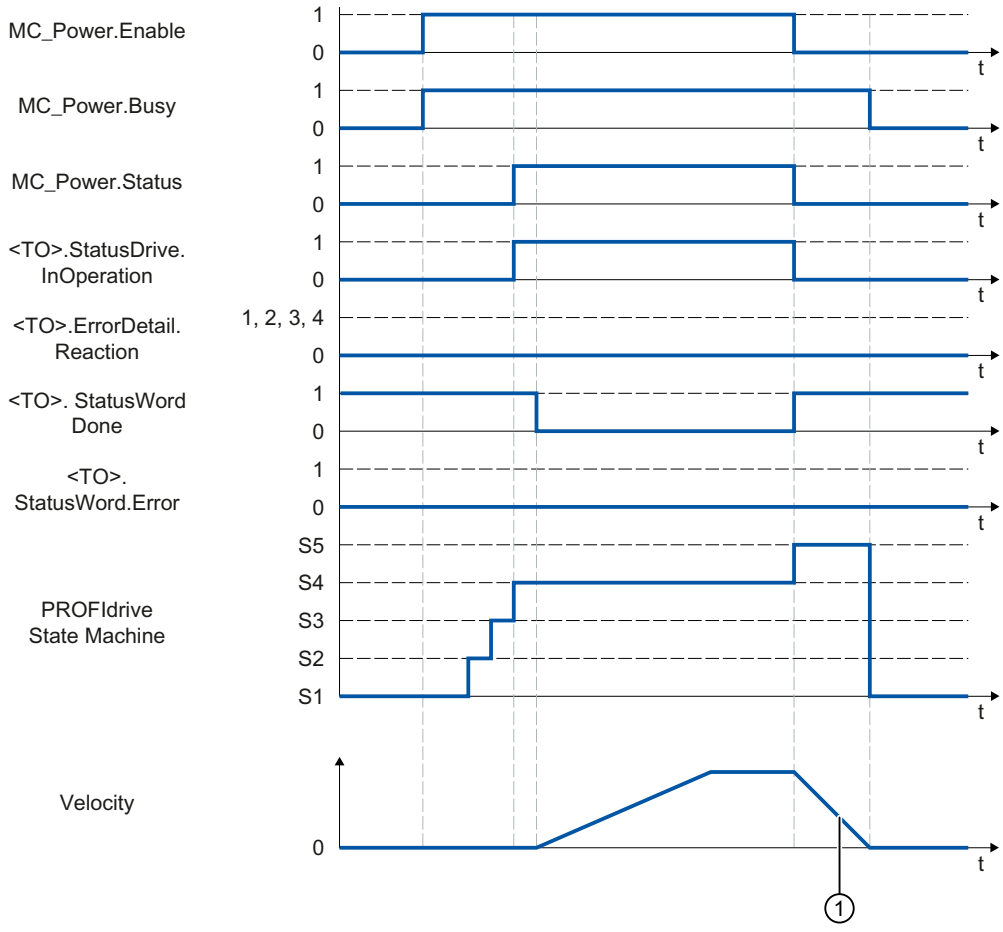

① The deceleration ramp depends on the configuration in the drive.

Alarm reactions with braking ramp via the technology object (S7-1500)

Function chart: Enabling a technology object and occurrence of a technology alarm with braking ramp via the technology object

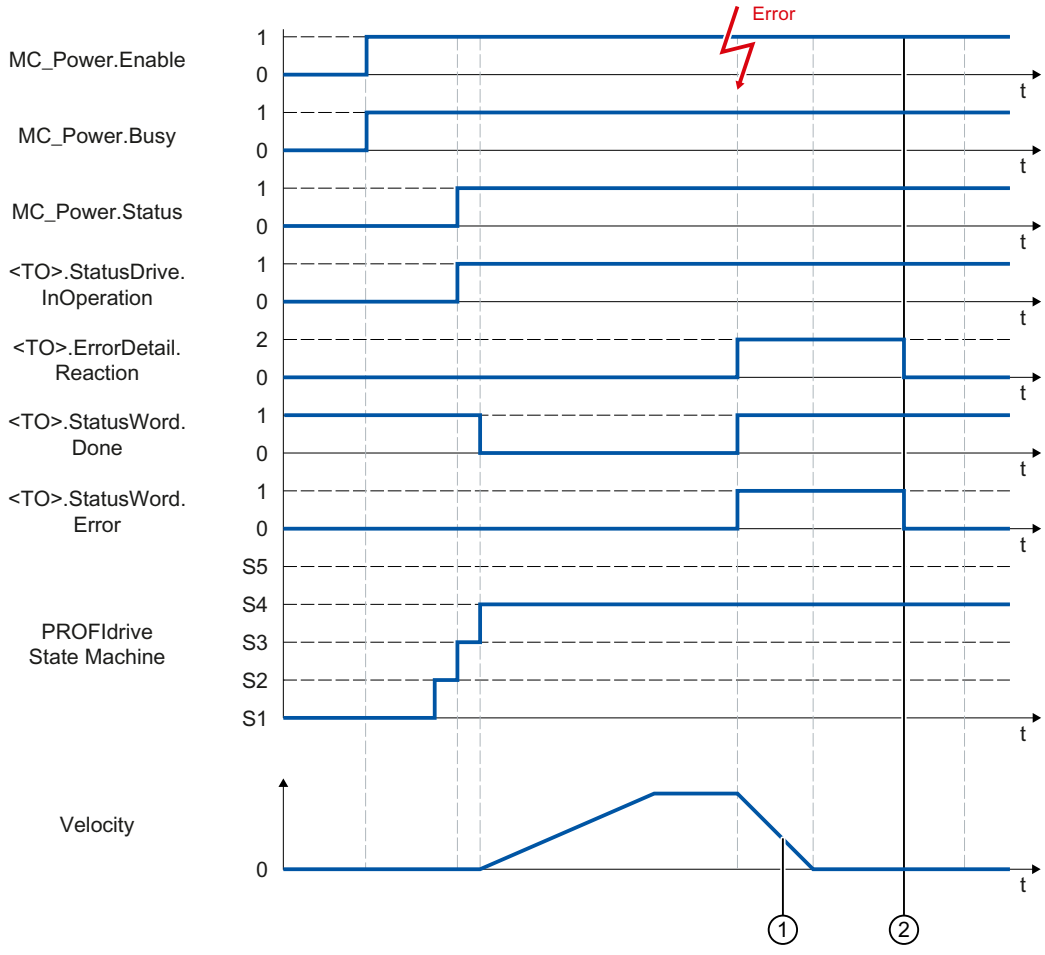

① The axis is braked based on the alarm reaction:

- Stop with current dynamic values (<TO>.ErrorDetail.Reaction = 1)
  The axis is braked with the deceleration in the Motion Control instruction.
- Stop with maximum dynamic values (<TO>.ErrorDetail.Reaction = 2)
  The axis decelerates with the configured maximum deceleration.
- Stop with emergency stop ramp (<TO>.ErrorDetail.Reaction = 3)
  The axis is braked with the configured emergency stop deceleration.

② The technology alarm is acknowledged.

Alarm response "Remove enable" (S7-1500)

Function chart: Enabling a technology object and occurrence of a technology alarm with alarm reaction "Remove enable"

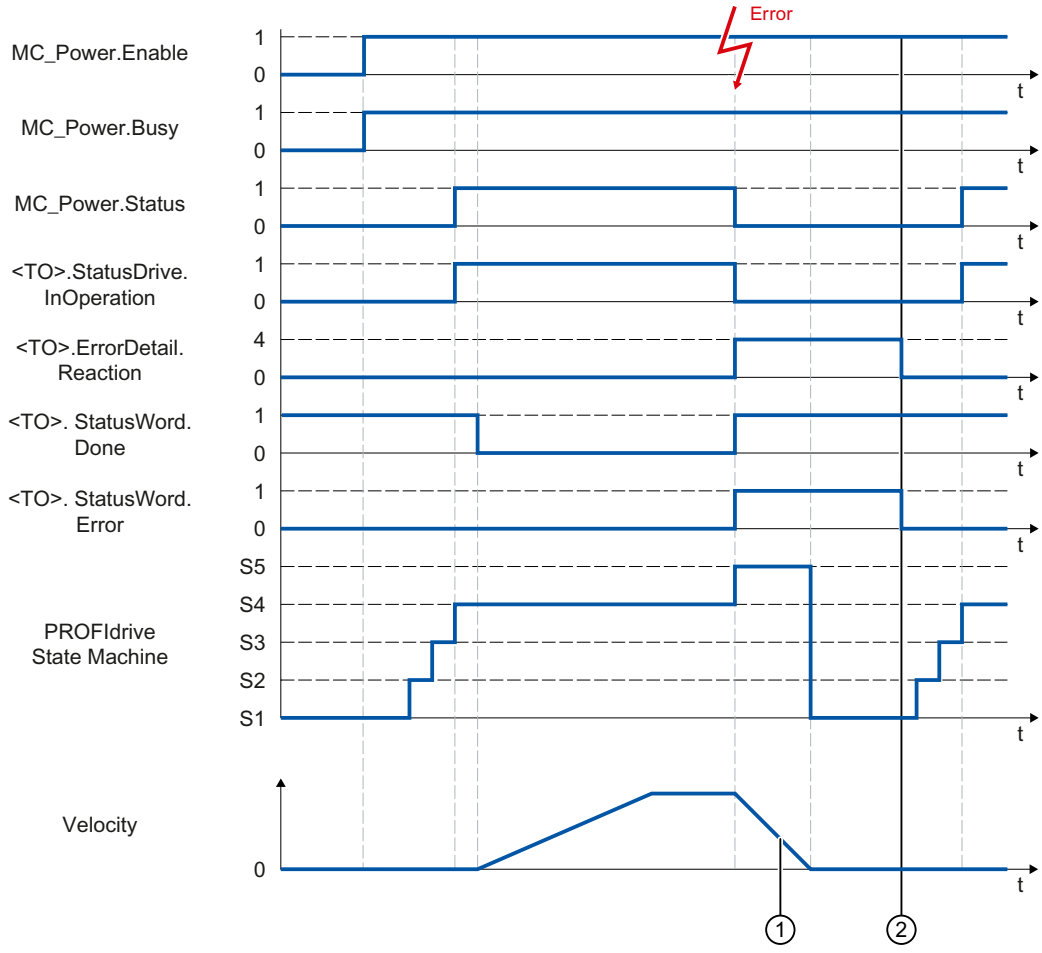

① The deceleration ramp depends on the configuration in the drive.

② The technology alarm is acknowledged at time ②.

Analog drive connection (S7-1500)

"StopMode" = 0, 2 (S7-1500)

Function chart: Enabling a technology object and disabling with "StopMode" = 0, 2

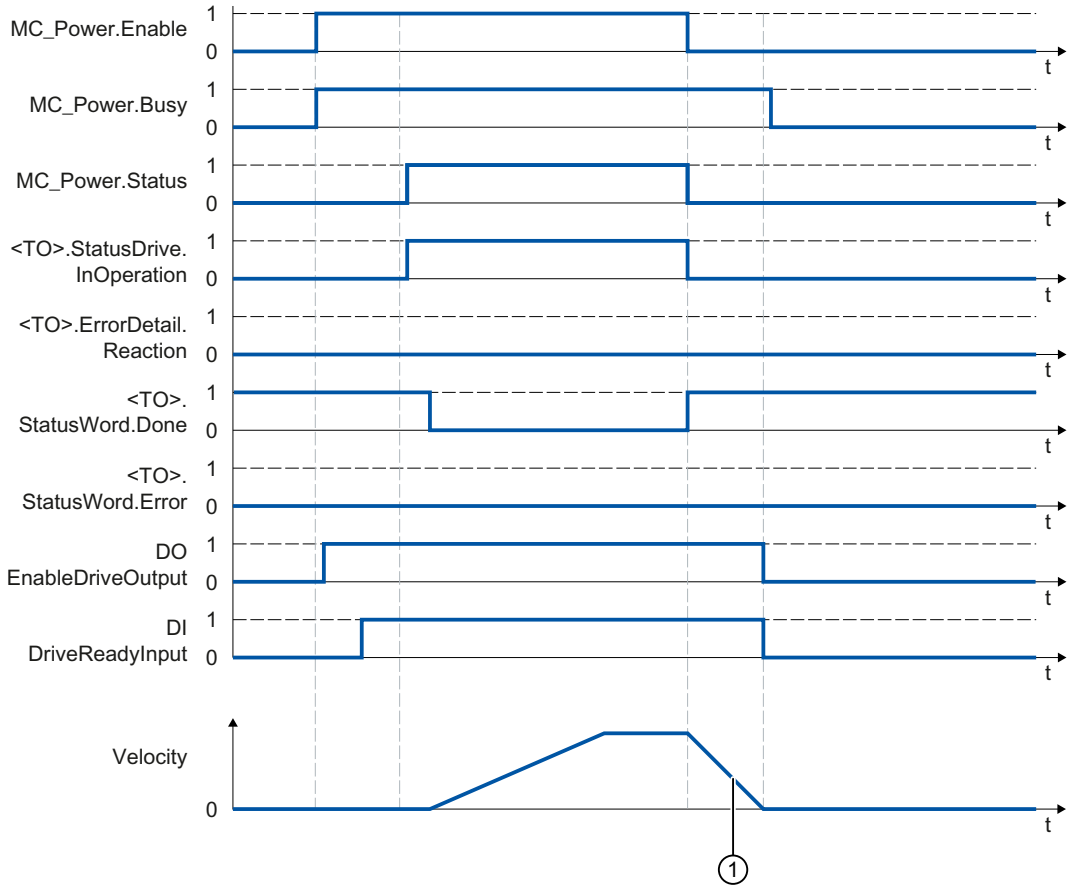

①
- "StopMode" = 0
  The axis is braked with the configured emergency stop deceleration.
- "StopMode" = 2
  The axis decelerates with the configured maximum deceleration.

"StopMode" = 1 (S7-1500)

Function chart: Enabling a technology object and disabling with "StopMode" = 1

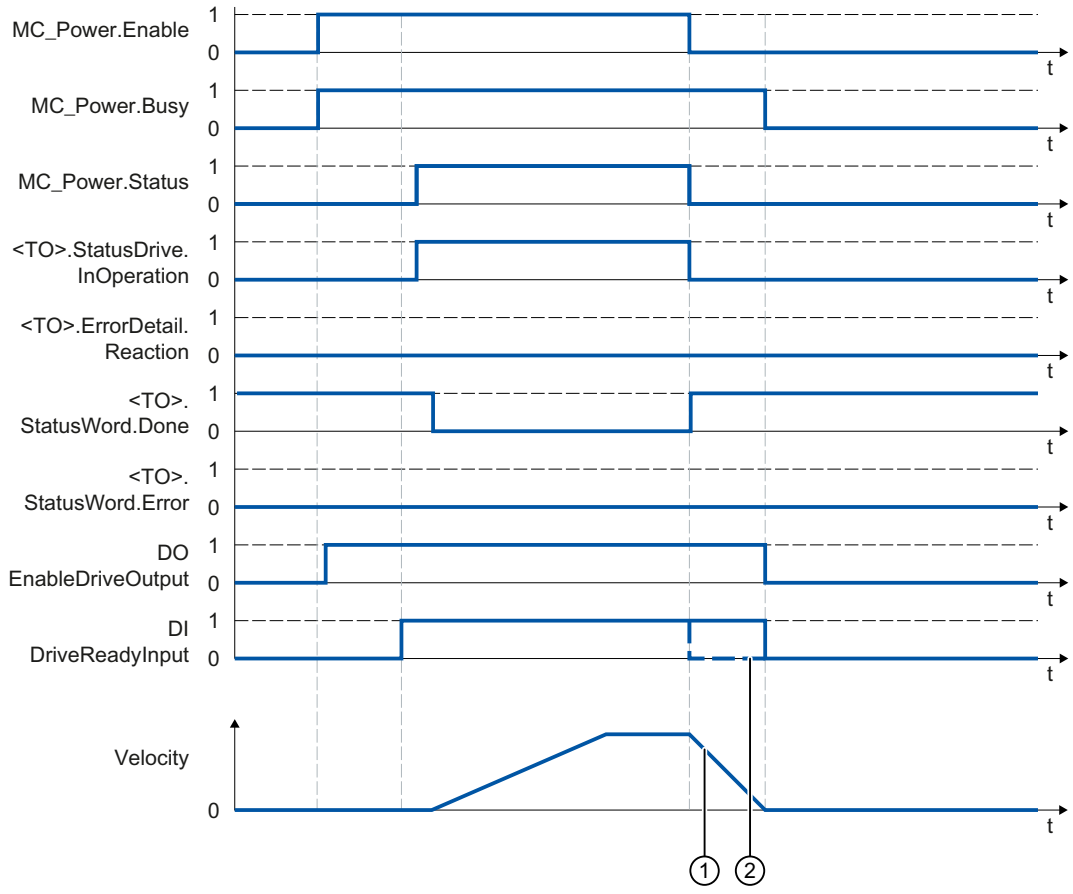

① The deceleration ramp depends on the configuration in the drive.

② The behavior of the ready signal of the drive "DI DriveReadyInput" is manufacturer-specific.

Alarm reactions with braking ramp via the technology object (S7-1500)

Function chart: Enabling a technology object and occurrence of a technology alarm with braking ramp via the technology object

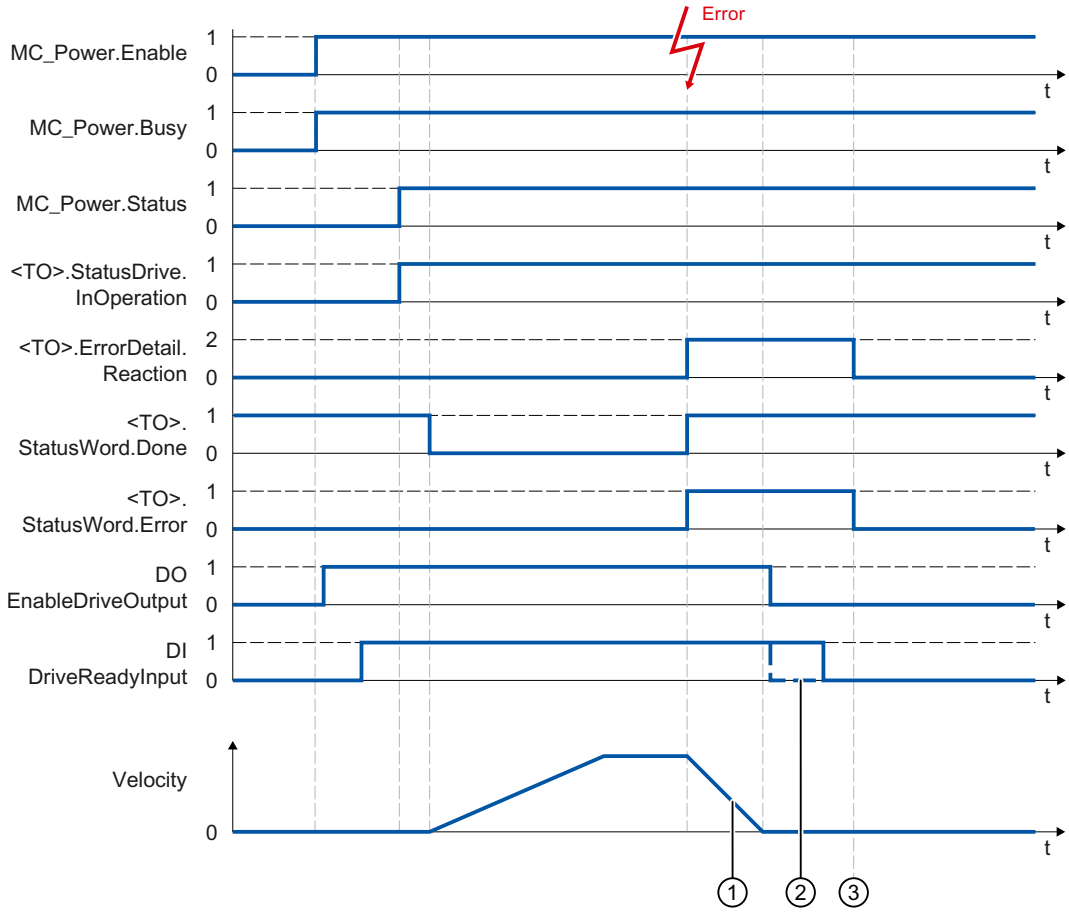

① The axis is braked based on the alarm reaction:

- Stop with current dynamic values (<TO>.ErrorDetail.Reaction = 1)
  The axis is braked with the deceleration in the Motion Control instruction.
- Stop with maximum dynamic values (<TO>.ErrorDetail.Reaction = 2)
  The axis decelerates with the configured maximum deceleration.
- Stop with emergency stop ramp (<TO>.ErrorDetail.Reaction = 3)
  The axis is braked with the configured emergency stop deceleration.

② The behavior of the ready signal of the drive "DI DriveReadyInput" is manufacturer-specific.

③ The technology alarm is acknowledged at time ③.

Alarm response "Remove enable" (S7-1500)

Function chart: Enabling a technology object and occurrence of a technology alarm with alarm reaction "Remove enable"

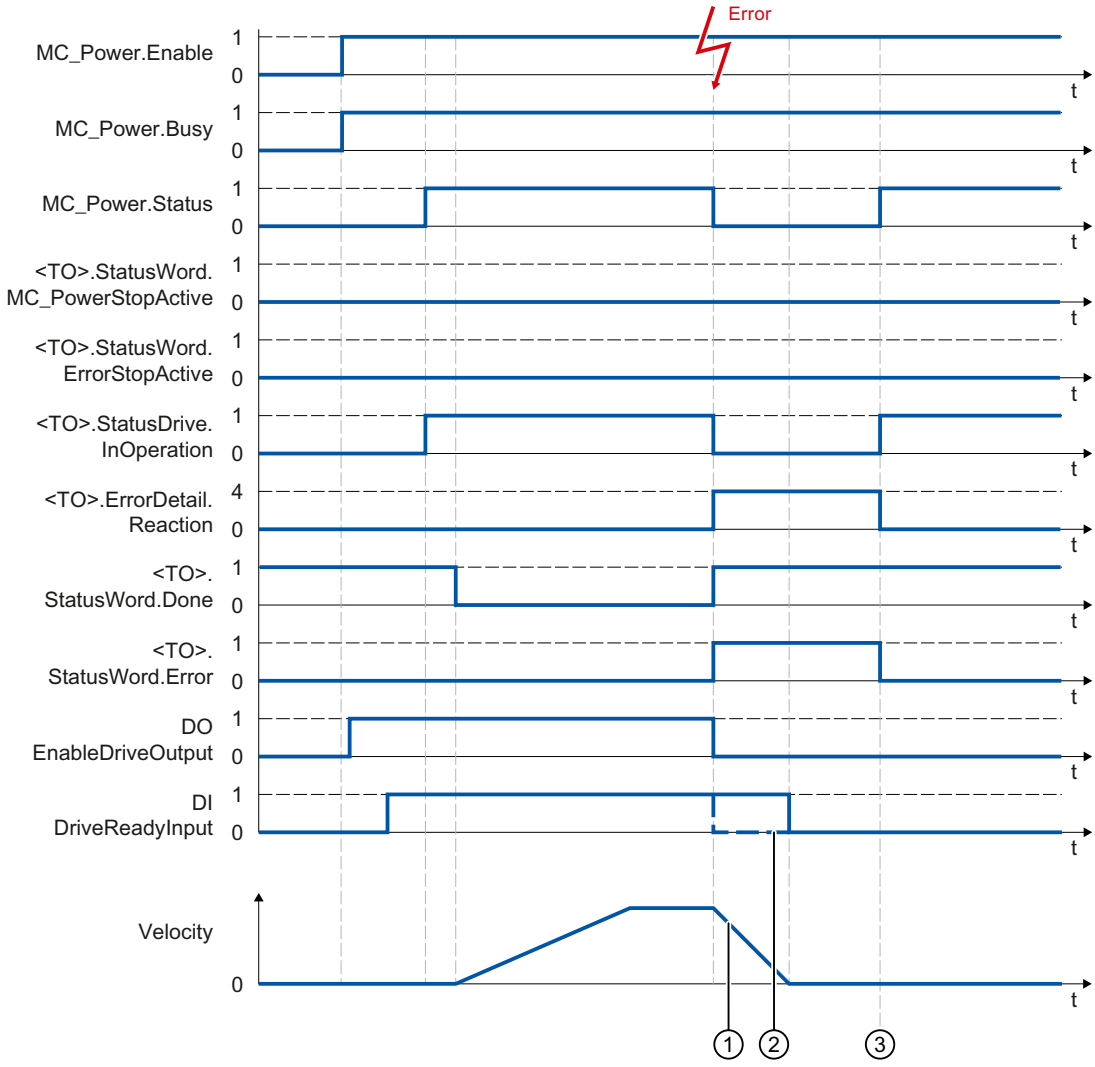

① The deceleration ramp depends on the configuration in the drive.

② The behavior of the ready signal of the drive "DI DriveReadyInput" is manufacturer-specific.

③ The technology alarm is acknowledged at time ③.

SINAMICS drives (S7-1500)

Compatibility list (S7-1500)

An overview of drives that can be interconnected with an S7-1500 CPU is available at:

https://support.industry.siemens.com/cs/document/109750431 (https://support.industry.siemens.com/cs/ww/en/view/109750431)

Homing SINAMICS drives with external zero marks (S7-1500)

For SINAMICS drives with external zero mark, synchronization during homing must always occur on the left side of the external zero mark's signal. That is to say, with a positive direction of travel synchronization is done on a positive edge, and with a negative direction of travel synchronization is done on a negative edge.

By inverting the signal, synchronization can also be done on the right sight of the signal of the external zero mark. The inversion can be configured in the drive using SINAMICS parameter P490.

Homing to an encoder zero mark or an external zero mark is configured in SINAMICS parameter P495.

Data types (S7-1500)

Data type for use with technology as of V4.0

The table below contains the data types for reference to the respective technology object:

Data type	Structure
TO_SpeedAxis	Speed axis
TO_PositioningAxis	Positioning axis
TO_SynchronousAxis	Synchronous axis
TO_Encoder	External encoder
TO_OutputCam	Output cam
TO_CamTrack	Cam track
TO_MeasuringInput	Measuring input
TO_Cam	Cam (S7-1500T)
TO_Kinematics	Kinematics (S7-1500T)
PD_TELx	Telegram no. "x"
PD_STW1_611Umode	Control word 1 (STW1)
PD_STW2_611Umode	Control word 2 (STW2)
PD_ZSW1_611Umode	Status word 1 (ZSW1)
PD_ZSW2_611Umode	Status word 2 (ZSW2)

See also

PROFIdrive telegrams (Page 1636)

Extended Motion Control Resources

If you use multiple kinematics technology objects and multiple cam technology objects, you cannot completely use up the "Extended Motion Control Resources".

CPU	Extended Motion Control Resources	Number of Kinematics Technology Objects	Maximum Number of Cam Technology Objects
CPU 1511T(F)	40	0	20
		1	0
CPU 1515T(F)	120	0	60
		1	45
		2	30
		3	15
		4	0
CPU 1516T(F)	192	0	96
		1	80
		2	64
		3	48
		4	32
		5	16
		6	0
CPU 1517T(F)	256	0	128
		1	112
		2	96
		3	80
		4	64
		5	48
		6	32
		7	16
		8	0

6.1.3 Using S7-1500T Kinematics Functions

6.1.3.1 Introduction

Interplay of the various documents

For a better overview, the documentation of the Motion Control functions is divided into the following documents:

- Using S7-1500T Motion Control
- Using S7-1500T Kinematics Functions

"Using S7-1500 Motion Control" describes the Motion Control functions for the following technology objects:

- Speed axis
- Positioning axis
- Synchronous axis
- External encoder
- Measuring input
- Output cam
- Cam track
- Cam (S7-1500T)

"Using S7-1500T Kinematics Functions" describes the Motion Control functions for the kinematics technology object. This document assumes that the Motion Control functions described in "Using S7-1500 Motion Control functions" are known.

See also

Function Manual "S7-1500T Motion Control V4.0 in the TIA Portal V15" (https://support.industry.siemens.com/cs/ww/en/view/109749263)

Kinematics systems for handling tasks

Kinematics are user-programmable mechanical systems in which multiple mechanically coupled axes produce the motion of a working point. The S7-1500T technology CPUs provide functions for controlling kinematics systems, e.g. for handling tasks, with the kinematics technology object. Typical applications include:

- Pick & Place
- Installation
- Palletizing

The kinematics control panel and extensive online and diagnostic functions support straightforward commissioning of kinematics systems. The kinematics technology object is fully integrated in the system diagnostics of the S7-1500 CPU.

Term definition

Kinematics

Kinematics are user-programmable mechanical systems in which multiple mechanically coupled axes produce the motion of a working point.

Kinematics axes

Kinematics axes are the axes of the kinematics motion. You connect each kinematics axis with a positioning axis/synchronous axis technology object.

Kinematics zero point (KZP)

The coordinate origin of the kinematics coordinate system (KCS) is the KZP. You configure the geometry parameters of the kinematics starting from the KZP.

Zero point of the flange coordinate system (FNP)

The coordinate origin of the flange coordinate system (FCS) is the FNP. Starting from the FNP, you define, for example, the flange zones of the kinematics.

Tool center point (TCP)

The coordinate origin of the tool coordinate system (TCS) is the tool center point or TCP. The TCP is the operating point of the kinematics.

Degrees of freedom of kinematics

The degrees of freedom of kinematics are the dimensions in which the tool can move. 2D kinematics systems move the tool in the xz plane and thus have two translational degrees of freedom. 3D kinematics systems move the tool in xyz space and thus have three translational degrees of freedom. The optional orientation of the tool is a further degree of freedom (rotation of the tool around the z-axis).

Machine coordinate system (MCS)

The MCS contains the position data of the interconnected kinematics axes and thus combines up to four one-dimensional systems in one system.

Job sequence

The job sequence of the kinematics technology object is the memory to which motion-related Motion Control jobs are entered as pending, inactive jobs. All jobs in the job sequence are taken into account during the motion preparation.

AxesGroup

Kinematics-related Motion Control instructions have the input parameter "AxesGroup". The kinematics technology object groups the interconnected kinematics axes. Therefore, you can assign the kinematics technology object directly to the input parameter "AxesGroup".

Functions

You execute the functions of the kinematics technology object using the Motion Control instructions in your user program or the TIA Portal (under "Technology object > Commissioning").

The following table shows the functions that are supported by the technology object:

Function	Brief description
Motion Control instructions (user program)	
"MC_GroupInterrupt"	Interrupt motion execution
"MC_GroupContinue"	Continue motion execution
"MC_GroupStop"	Stop motion
"MC_MoveLinearAbsolute"	Position kinematics with linear path motion
"MC_MoveLinearRelative"	Relative positioning of kinematics with linear path motion
"MC_MoveCircularAbsolute"	Position kinematics with circular path motion
"MC_MoveCircularRelative"	Relative positioning of kinematics with circular path motion
"MC_DefineWorkspaceZone"	Define workspace zone
"MC_DefineKinematicsZone"	Define kinematics zone
"MC_SetWorkspaceZoneActive"	Activate workspace zone
"MC_SetWorkspaceZoneInactive"	Deactivate workspace zone
"MC_SetKinematicsZoneActive"	Activate kinematics zone
"MC_SetKinematicsZoneInactive"	Deactivate kinematics zone
"MC_DefineTool"	Re-define tool
"MC_SetTool"	Change active tool
"MC_SetOcsFrame"	Redefine object coordinate systems
TIA Portal	
"Kinematics control panel (Page 2290)"	Homing of kinematics axes and traversing of kinematics systems or individual kinematics axes via the TIA Portal

6.1.3.2 Basics

Kinematics technology object

The kinematics technology object calculates motion setpoints for the tool center point (TCP) of the kinematics taking into account the dynamic settings. The kinematics technology object calculates the motion setpoints for the individual axes of the kinematics and vice versa from the current values of the axes using the kinematics transformation. The kinematics technology object outputs the axis-specific motion setpoints to the interconnected positioning axes.

The kinematics technology object provides the kinematics transformation (Page 2238) for the predefined kinematics types on the system level. In the case of user-defined kinematics systems, you must provide the user transformation (Page 2243) in a separate program.

You create the individual axes of the kinematics in the TIA Portal as "Positioning axis" or "Synchronous axis" technology objects. When you configure the kinematics technology object, you interconnect the axes in accordance with the configured kinematics type.

You can find an overview of the functions of the kinematics technology object in the "Functions" (Page 2149) section.

The graphic below shows the basic principle of operation of the kinematics technology object:

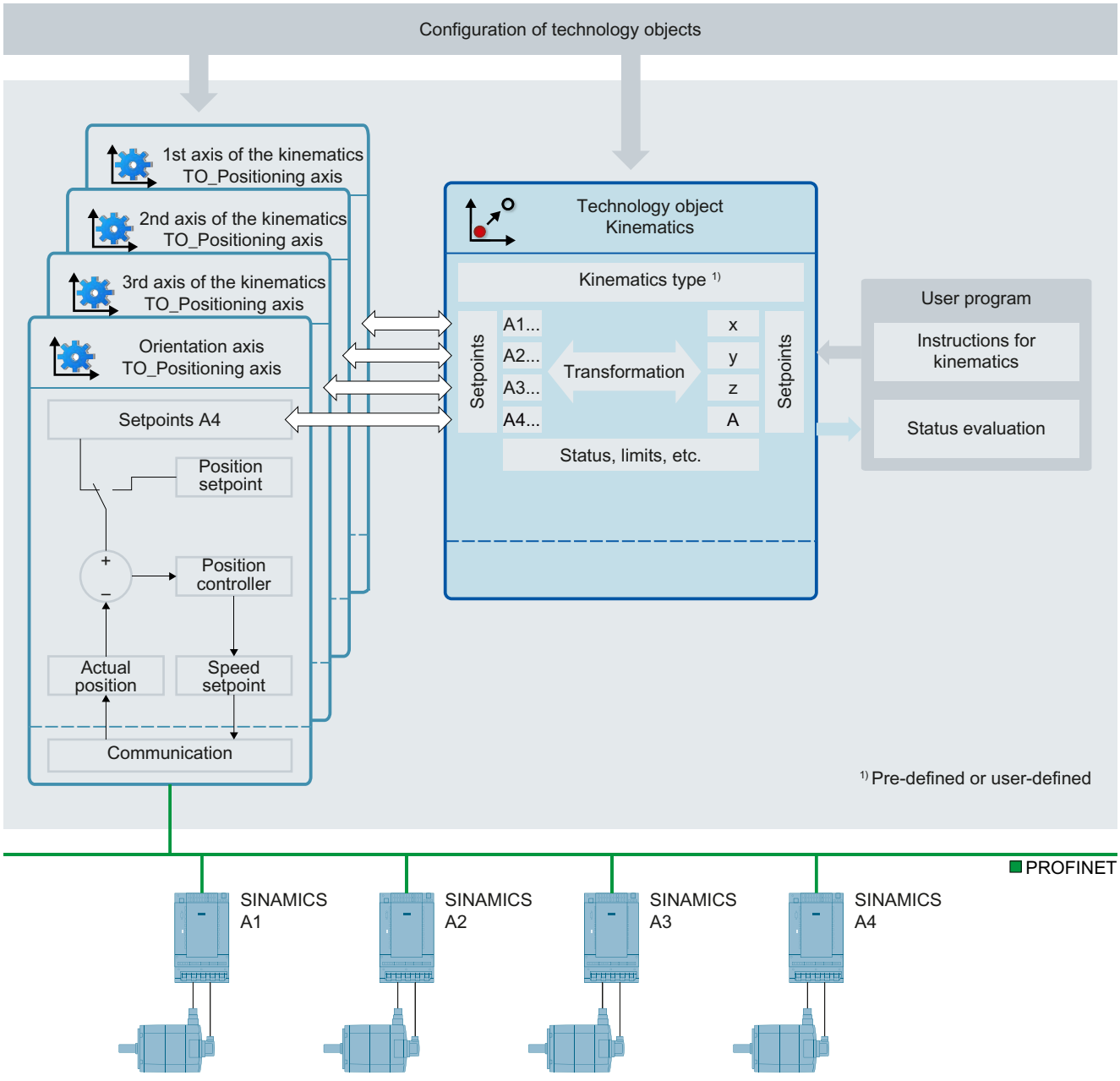

Interconnection rules

You can interconnect a kinematics technology object with positioning axes and synchronous axes. There must be a clear reference between the kinematics technology object and the interconnected axes. You cannot use a second kinematics technology object with already interconnected axes.

No provision is made for changing the interconnection of the axes during operation.

Virtual axis/simulation

You can also interconnect the kinematics technology object with axes in simulation and with virtual axes.

Configuration limits for kinematics systems

Motion Control resources

Each CPU offers a defined set of "Motion Control resources". For information on the total Motion Control resources available, refer to the technical specifications of the utilized CPU.

You can find an overview of the Motion Control resources of a CPU in the TIA Portal under "Tools > Resources".

Extended Motion Control resources (S7-1500T)

In addition to the Motion Control resources of the interconnected axes, a kinematics technology object utilizes 30 "Extended Motion Control resources". For information on the maximum number of usable kinematics systems, refer to the technical specifications of the utilized CPU.

You can find the technical specifications of the S7-15xxT CPUs in the respective manual.

Application cycle

As the number of technology objects used increases, the computing time needed by the CPU to process the technology objects increases. The Motion Control application cycle can be adapted according to the number of technology objects used.

Units of measure

The kinematics technology object supports the following units of measure for position and velocity of linear axes:

Position	Velocity
nm, µm, mm, m, km	mm/s, mm/min, mm/h, m/s, m/min, m/h, km/min, km/h
in, ft, mi	in/s, in/min, ft/s, ft/min, mi/h

The kinematics technology object supports the following units of measure for angle and angular velocity of rotary axes:

Angle	Angular velocity
°, rad	°/s, °/min, rad/s, rad/min

The acceleration is set accordingly as the position/s² (angle/s²) unit of measure.

The jerk is set accordingly as the position/s³ (angle/s³) unit of measure.

Note

When setting or changing the units of measure, take into consideration the effect on the display of parameter values and the user program:

- Display of parameter values in the technology data block
- Assignment of parameters in the user program
- Input and display of the position and velocity in the TIA Portal
- Setpoint settings by leading axes in synchronous operation

All information and displays are shown according to the selected unit of measure.

The set units are displayed in the tag structure of the <TO>.Units technology object. The tag structure is described in the Appendix (Page 2330).

Units of measure of the axes and the kinematics technology object

The technology objects always transfer values without units of measure.

For example, if you set [mm] for an axis and [m] for the kinematics technology object, the kinematics technology object miscalculates the position values of the linear axis in [m]. If, in this example, the kinematics technology object outputs a setpoint for a one-meter motion, the axis only moves by one millimeter.

The kinematics technology object outputs linear and rotary setpoints to the interconnected axes according to the kinematics type. The kinematics technology object does not check the axis type of the interconnected axis (linear or rotary).

When configuring the units of measure, take into consideration the following specifications:

- Configure the interconnected technology objects as linear or rotary axes according to the kinematics type.
- Configure the same linear/rotary units of measure for the axes interconnected according to the kinematics type as for the kinematics technology object.

Modulo setting

The kinematics technology object itself has no modulo setting. When you interconnect axes with active modulo setting to the kinematics technology object, the module range of the axes must cover at least the traversing range of the kinematics. The zero position of the axis must match the zero position of the kinematics axis. With the exception of the orientation axis, the module range of the axes cannot be changed during a kinematics motion.

The modulo setting is typically used for the orientation axis. In the case of the orientation axis (axis A4 in the kinematics), the kinematics transformation covers the entire traversing range of the axis. The orientation axis traverses without limitations through an activated modulo setting.

An angle greater than 360° can be defined for the Cartesian orientation. A relative motion traverses this angle. An absolute motion maps this angle in the range from 0° to 360°. The range -180° to +180° is generally defined for the coordinate A of the tool center point (TCP).

Coordinate systems and frames

A handling task involves many objects, e.g. kinematics systems, tools, pallets and products. You describe these objects and their relative positions with coordinate systems and frames. The kinematics technology object calculates all motions for the tool center point (TCP).

Frames

Frames specify the shift and rotation of one coordinate system relative to another coordinate system.

Coordinate systems

The kinematics technology object uses the following right-handed Cartesian coordinate systems according to DIN 66217:

- World coordinate system (WCS)
- Kinematics coordinate system (KCS)
- Flange coordinate system (FCS)
- Tool coordinate system (TCS)
- Object coordinate systems (OCS)

The following graphic shows the relative position of the coordinate systems using a workspace example:

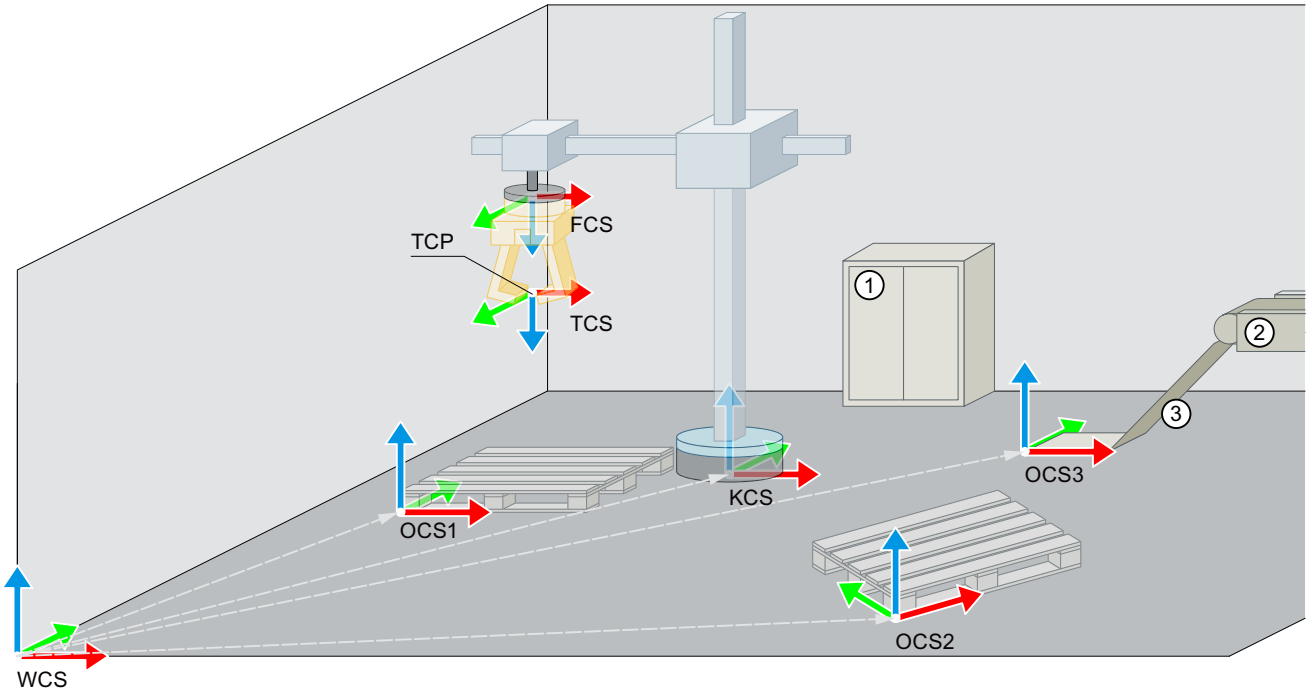

①	Control cabinet
②	Conveyor belt
③	Slide
WCS	World coordinate system
KCS	Kinematics coordinate system
FCS	Flange coordinate system (FCS)
TCS	Tool coordinate system
TCP	Tool center point
OCS	Object coordinate system

World coordinate system (WCS)

The WCS is the fixed coordinate system of the environment or workspace of the kinematics. The zero point of the WCS is the reference point for objects and motions on the kinematics technology object. Starting from the zero point of the WCS (e.g. corner of a workspace), you define the position of the objects using frames.

Kinematics coordinate system (KCS)

The KCS is connected to the kinematics. The position of the KCS within the kinematics is specified for each predefined kinematics type. The coordinate origin of the KCS is the kinematics zero point (KZP). You configure the geometry parameters of the kinematics starting from the KZP.

You configure the position of the KCS in the WCS using the KCS frame.

Flange coordinate system (FCS)

The FCS is attached to the tool adapter (flange) of the kinematics. As a result, the position of the FCS changes with kinematics motions.

The position of the FCS in the zero position of the kinematics results from the configuration of the geometry parameters of the kinematics. The kinematics technology object calculates the transformation frame from the geometry parameters. The transformation frame describes the position of the FCS in the KCS. The z axis of the FCS always points in the negative z direction of the KCS.

The following graphic shows the positions of the FCS and KCS and the transformation frame using the "Cylindrical robot" kinematics example:

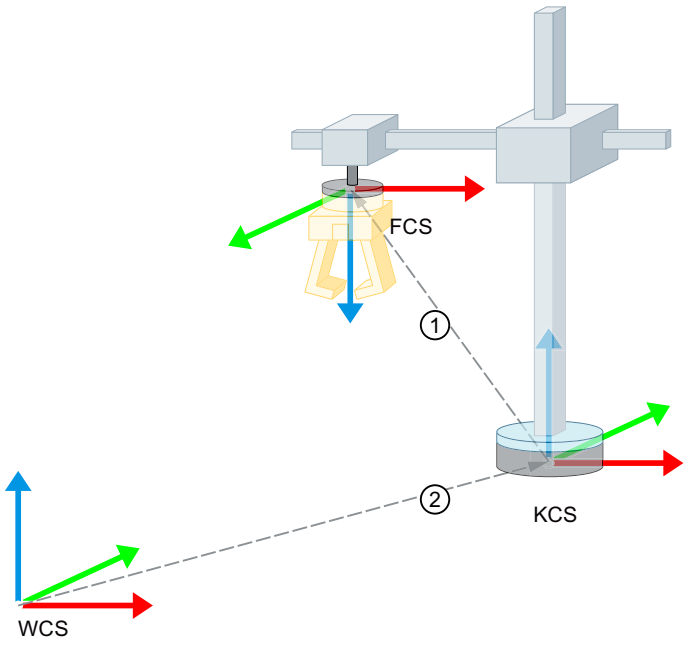

① Transformation frame

② KCS frame

Tool coordinate system (TCS - Tool Coordinate System) and tool center point (TCP - Tool Center Point)

The TCS is attached to the FCS and defines the tool center point (TCP) in the coordinate origin. The TCP is the operating point of the tool. The kinematics motions always refer to the TCP (with reference to WCS/OCS). You define the position of the TCS in the FCS using a tool frame. The z axis of the TCS always points in the negative z direction of the KCS. You can define tool frames for up to three tools, of which only one tool and therefore one tool frame is active at the same time.

The following graphic shows the position of the TCS and the TCP in the workspace:

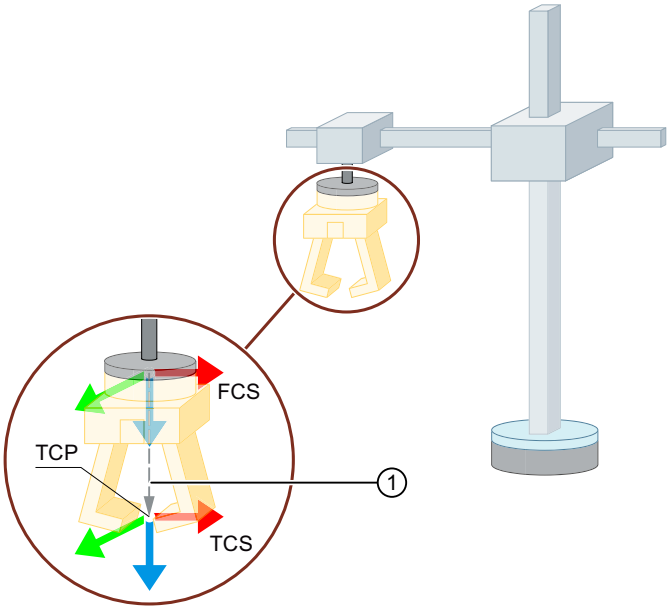

① Tool frame

Object coordinate system (OCS)

The OCS is a user-defined coordinate system. With an OCS, for example, you define the position of a pallet in the workspace. You define the position of the OCS in the WCS with an OCS frame. You can define up to three OCS frames which are active at the same time.

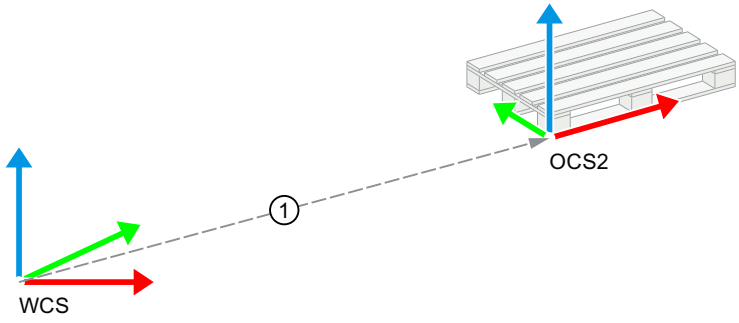

① OCS frame

Frames

The following table shows the frames for the kinematics technology object:

Frame	Description
KCS frame	Position of the kinematics coordinate system (KCS) in the world coordinate system (WCS)
Transformation frame	Position of the flange coordinate system (FCS) in the KCS The transformation frame results from the kinematics transformation and is displayed in the "<TO>.FlangeInKcs" tag of the technology object.
Tool frame	Position of the tool coordinate system (TCS) in the FCS

Frame	Description
OCS[1..3]-frame	Position of the object coordinate systems 1 to 3 (OCS[1..3]) in the WCS
Target position	Target position for a kinematics motion

Frame definition

Frames define the shift and rotation of one coordinate system relative to another coordinate system with the following values:

Value in the frame	Description
x	Shift in the x direction in the reference coordinate system
y	Shift in the y direction in the reference coordinate system
z	Shift in the z direction in the reference coordinate system
A	Rotation around the z-axis
B	Rotation around the y-axis
C	Rotation around the x-axis

The following table shows the restrictions for frames depending on the kinematics type. The information "x", "y" and "z" means that a shift in the respective direction is possible. The information "A", "B" and "C" means that a rotation in the respective direction is possible. The information "= 0.0" means that a shift or rotation in the respective direction is not permitted or not relevant for the kinematics type.

Kinematics type	KCS frame/OCS frame		Tool frame		Target position	
2D	Shift	Rotation	Shift	Rotation	Shift	Rotation
	x	A = 0.0	x	A = 0.0	x	A = 0.0
	y = 0.0	B	y = 0.0	B = 0.0	y = 0.0	B = 0.0
	z	C = 0.0	z	C = 0.0	z	C = 0.0
2D with orientation	Shift	Rotation	Shift	Rotation	Shift	Rotation
	x	A = 0.0	x = 0.0	A	x	A
	y = 0.0	B = 0.0	y = 0.0	B = 0.0	y = 0.0	B = 0.0
	z	C = 0.0	z	C = 0.0	z	C = 0.0
3D	Shift	Rotation	Shift	Rotation	Shift	Rotation
	x	A	x	A = 0.0	x	A = 0.0
	y	B	y	B = 0.0	y	B = 0.0
	z	C	z	C = 0.0	z	C = 0.0
3D with orientation	Shift	Rotation	Shift	Rotation	Shift	Rotation
	x	A	x	A	x	A
	y	B = 0.0	y	B = 0.0	y	B = 0.0
	z	C = 0.0	z	C = 0.0	z	C = 0.0

x, y, z, A, B, C: Shift/rotation possible

Value = 0.0: Shift/rotation not permitted or not relevant

The following table shows the value ranges for the rotations of KCS, OCS and tool frames depending on the kinematics type:

Kinematics type	Value ranges			
	KCS frame/OCS frame		Tool frame	
2D	A	0.0	A	0.0
	B	-180.0° to 179.999°	-	-
	C	0.0	-	-
2D with orientation	A	0.0	A	-180.0° to 179.999°
	B	0.0	-	-
	C	0.0	-	-
3D	A	-180.0° to 179.999°	A	0.0
	B	-90.0° to 90.0°	-	-
	C	-180.0° to 179.999°	-	-
3D with orientation	A	-180.0° to 179.999°	A	-180.0° to 179.999°
	B	0.0	-	-
	C	0.0	-	-

Value = 0.0: Rotation not permitted

No information (-): Parameter not available

Tags of coordinate systems and frames

The following tags of the kinematics technology object are relevant for coordinate systems and frames:

Tag	Description
Configuration	
<TO>.KcsFrame	KCS frame x, y, z, A, B, C
<TO>.OcsFrame[1..3]	OCS frame x, y, z, A, B, C
<TO>.Tool[1..3]	Tool frame x, y, z, A
Status values	
<TO>.Tcp	Position of the tool center point (TCP), TCP frame in the world coordinate system (WCS). x, y, z, A
<TO>.TcpInWcs	Parameter for tool center point in the world coordinate system x, y, z, A
<TO>.TcpInOcs[1..3]	Parameter for the tool center point (TCP) in the Object Coordinate Systems 1 to 3 (OCS) x, y, z, A
<TO>.FlangeInKcs	Parameter for the flange coordinate system (FCS) in the kinematics coordinate system (KCS) x, y, z, A
<TO>.StatusOcsFrame	Display of the OCS frames x, y, z, A, B, C

Kinematics types

The type of the mechanical system and the number of the axes determine the kinematics type. The mechanically coupled axes produce the motion of the tool center point (TCP). Depending on the kinematics type, you configure the kinematics using appropriate geometry parameters.

The kinematics technology object supports the following kinematics types:

Category	Kinematics type
Predefined kinematics systems	
Cartesian portal (Page 2162)	Cartesian portal 2D
	Cartesian portal 2D with orientation
	Cartesian portal 3D
	Cartesian portal 3D with orientation
Roller picker (Page 2173)	Roller picker 2D
	Roller picker 2D with orientation
	Roller picker 3D (vertical)
	Roller picker 3D with orientation (vertical)
	Roller picker 3D with orientation (horizontal)
SCARA (Page 2187)	SCARA 3D with orientation
Articulated arm (Page 2192)	Articulated arm 2D
	Articulated arm 2D with orientation
	Articulated arm 3D
	Articulated arm 3D with orientation
Delta picker (Page 2208)	Delta picker 2D
	Delta picker 2D with orientation
	Delta picker 3D
	Delta picker 3D with orientation
Cylindrical robot (Page 2221)	Cylindrical robot 3D
	Cylindrical robot 3D with orientation
Tripod (Page 2230)	Tripod 3D
	Tripod 3D with orientation
User-defined kinematics systems	
User-defined kinematics systems (Page 2237)	User-defined 2D
	User-defined 2D with orientation
	User-defined 3D
	User-defined 3D with orientation

Legend for representation of the kinematics

The following table shows the graphic elements and symbols which are used to display the kinematics:

Graphic element	Meaning
	Basis of kinematics
	Kinematics arm

Graphic element	Meaning
	Kinematics deflected from zero position
	Active rotary axis
	• Passive joint • Axis guide
	Active linear axis
	Rotary axis on the tool adapter (orientation axis)
	Tool adapter
	Tool (gripper)
	Coordinate axis set up out of the mapping plane
	Coordinate axis set up into the mapping plane
	Color x axis
	Color y axis
	Color z axis

Cartesian portal

Portal 2D

The kinematics "Portal 2D" supports two axes and two degrees of freedom. The following graphic shows the principal configuration and the typical working area of the kinematics:

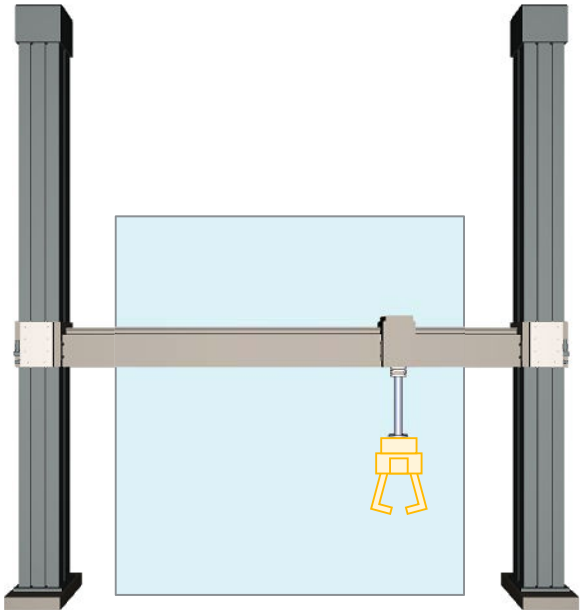

The kinematics consists of two orthogonal, linear axes A1 and A2. The axes enclose a rectangular working range.

Coordinate systems and zero position

The graphic below shows the following in the front view:

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive deflection of the kinematics is indicated (dashed)

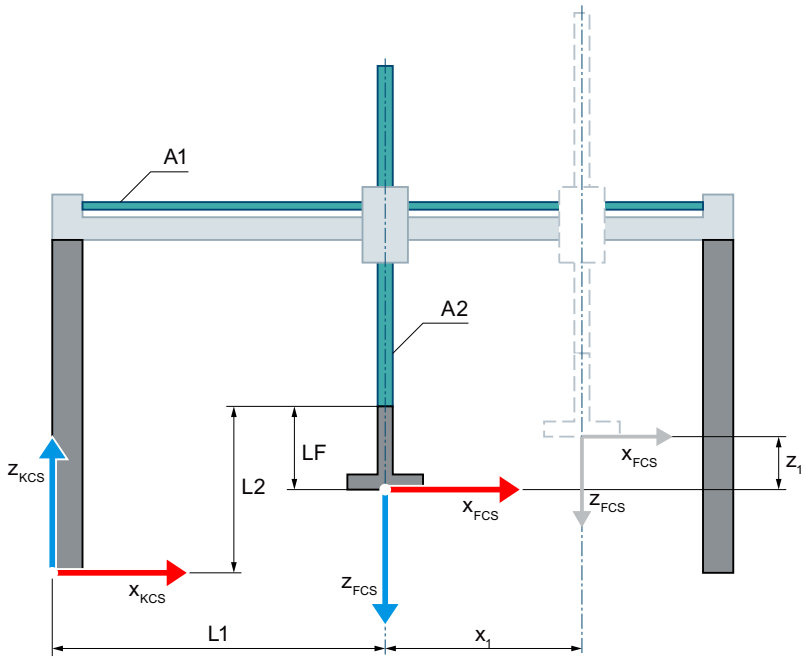

Zero position of the kinematics

L1	At zero position of the axis A1: Distance of the FCS to the KZP in x direction of the KCS
L2	At zero position of the axis A2: Distance of the FCS to the KZP and flange length LF in z direction of the KCS
LF	Flange length before the FCS in the z direction of the KCS

Deflection of the kinematics

x_1	Deflection of the axis A1 in the positive x direction
z_1	Deflection of the axis A2 in the positive z direction

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. The flange coordinate system (FCS) is located at distance LF from the zero position of the axis A2.

The position 0.0 on the respective interconnected technology object defines the zero positions of the axes A1 and A2 in the KCS. You define the distances of the zero positions of the axes in relation to the kinematics zero point using lengths L1 and L2.

Transformation area

The kinematics transformation covers the entire traversing range (Page 2239) of the axes.

Portal 2D with orientation

The kinematics "Portal 2D with orientation" supports three axes and three degrees of freedom. The following graphic shows the principal configuration and the typical working area of the kinematics:

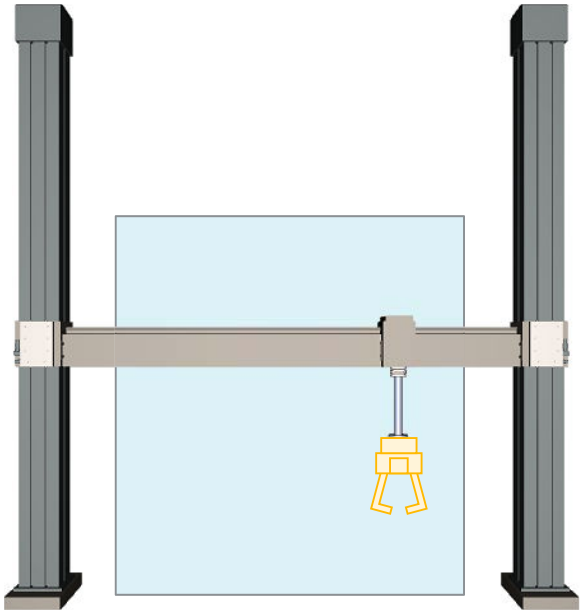

The kinematics consists of the following axes:

- Two orthogonal linear axes A1 and A2
- One rotary axis A4 (orientation axis)

The linear axes enclose a rectangular working area. The orientation axis A4 enables rotation of the tool.

Coordinate systems and zero position

The graphic below shows the following in the front view:

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive deflection of the kinematics is indicated (dashed)

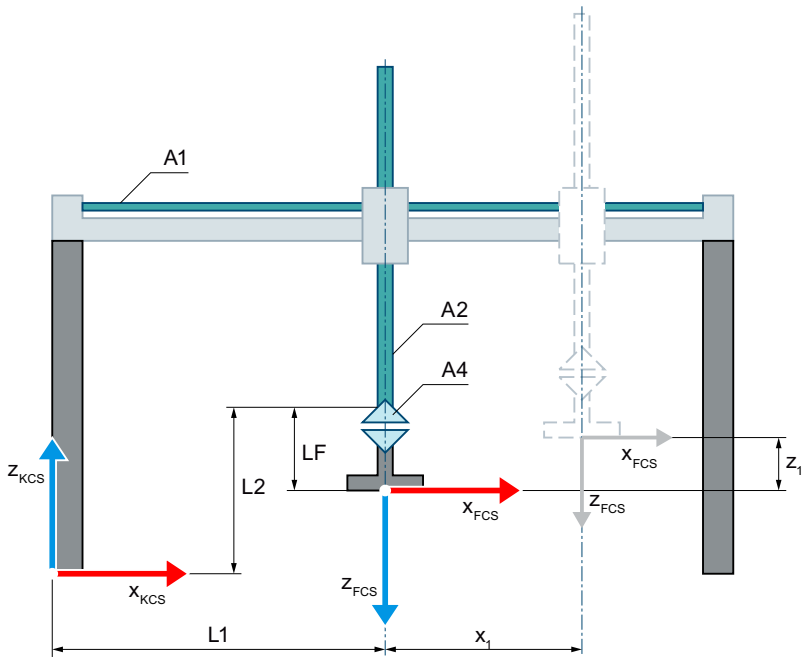

Zero position of the kinematics

L1 At zero position of the axis A1:
Distance of the FCS to the kinematics zero point (KZP) in x direction of the KCS

L2 At zero position of the axis A2:
Distance of the FCS to the KZP and flange length LF in z direction of the KCS

LF Flange length before the FCS in the z direction of the KCS

Deflection of the kinematics

x_1 Deflection of the axis A1 in the positive x direction

z_1 Deflection of the axis A2 in the positive z direction

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. The flange coordinate system (FCS) is located at distance LF from the zero position of the axis A2.

The position 0.0 on the respective interconnected technology object defines the zero positions of the axes A1 and A2 in the KCS. You define the distances of the zero positions of the axes A1 and A2 to the kinematics zero point with lengths L1 and L2. At the zero position of axis A4, the x axis of the FCS points in the direction of the x axis of the KCS.

Transformation area

The kinematics transformation covers the entire traversing range (Page 2239) of the axes.

Portal 3D

The kinematics "Portal 3D with orientation" supports three axes and three degrees of freedom. The following graphic shows the principal configuration and the typical working area of the kinematics:

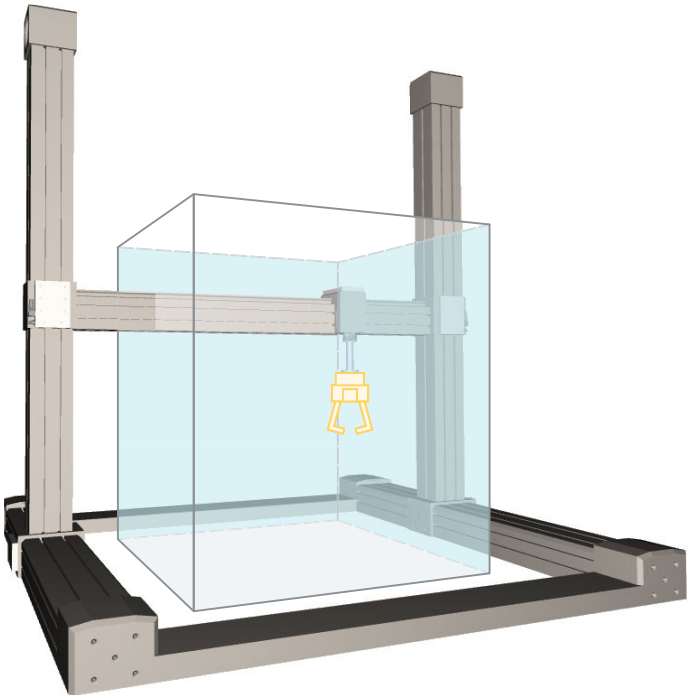

The kinematics consists of three orthogonal, linear axes A1, A2 and A3. The linear axes enclose a rectangular working area.

Coordinate systems and zero position

The graphic below shows the following in the front view (xz plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the axes A1 and A3 is indicated (dashed)

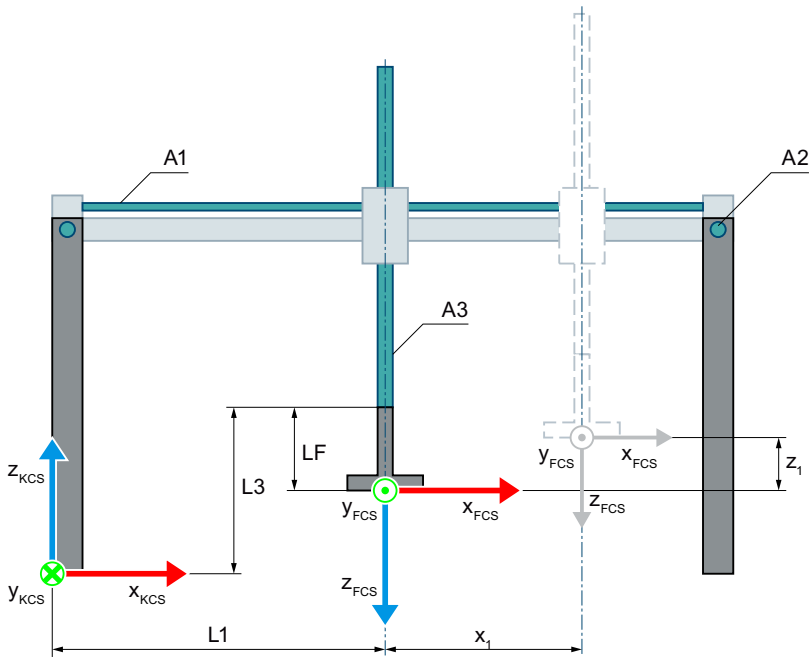

Zero position of the kinematics

L1 At zero position of the axis A1:
Distance of the FCS to the kinematics zero point (KZP) in x direction of the KCS

L3 At zero position of the axis A3:
Distance of the FCS to the KZP and flange length LF in z direction of the KCS

LF Flange length before the FCS in the z direction of the KCS

Deflection of the kinematics

x_1 Deflection of the axis A1 in the positive x direction

z_1 Deflection of the axis A3 in the positive z direction

Legend for representation of the kinematics (Page 2160)

The graphic below shows the following in the top view (xy plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the axes A1 and A2 is indicated (dashed)

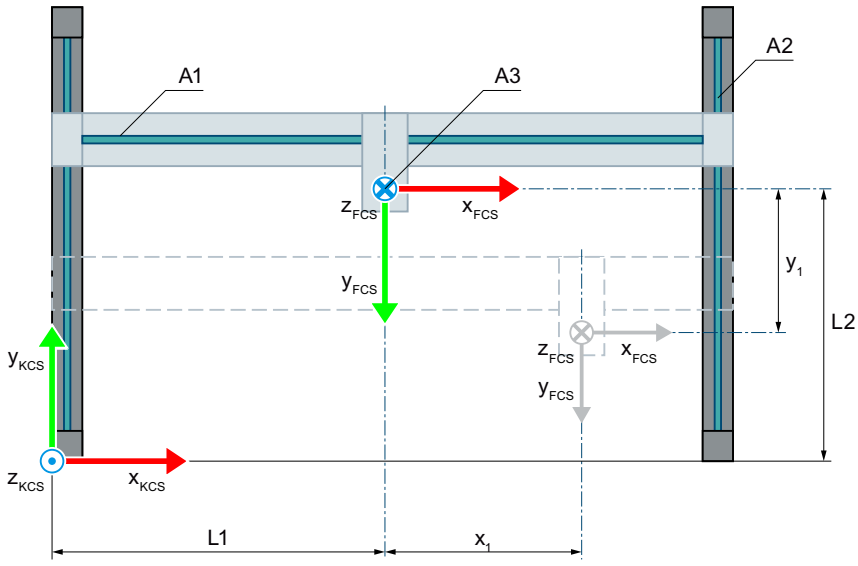

Zero position of the kinematics

L1 At zero position of the axis A1:
Distance of the FCS to the KZP in x direction of the KCS

L2 At zero position of the axis A2:
Distance of the FCS to the KZP in y direction of the KCS

Deflection of the kinematics

x_1 Deflection of the axis A1 in the positive x direction

y_1 Deflection of the axis A2 in negative y direction

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. The flange coordinate system (FCS) is located at distance LF from the zero position of the axis A2.

The position 0.0 on the respective interconnected technology object defines the zero positions of the axes A1, A2 and A3 in the KCS. You define the distances of the zero positions of the axes to the kinematics zero point with lengths L1, L2 and L3.

Transformation area

The kinematics transformation covers the entire traversing range (Page 2239) of the axes.

Portal 3D with orientation

The kinematics "Portal 3D with orientation" supports four axes and four degrees of freedom. The following graphic shows the principal configuration and the typical working area of the kinematics:

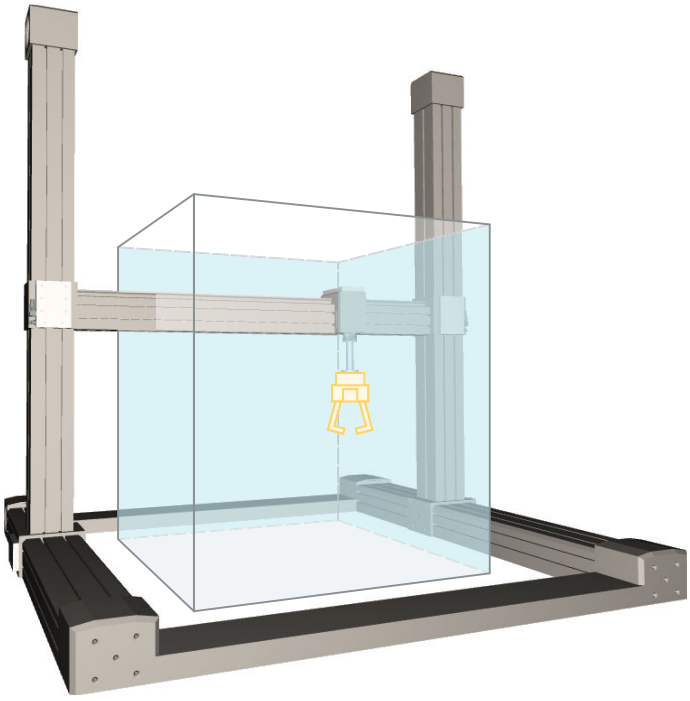

The kinematics consists of the following axes:

- Three orthogonal linear axes A1, A2 and A3
- One rotary axis A4 (orientation axis)

The linear axes enclose a rectangular working area. The orientation axis A4 enables rotation of the tool.

Coordinate systems and zero position

The graphic below shows the following in the front view (xz plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the axes A1 and A3 is indicated (dashed)

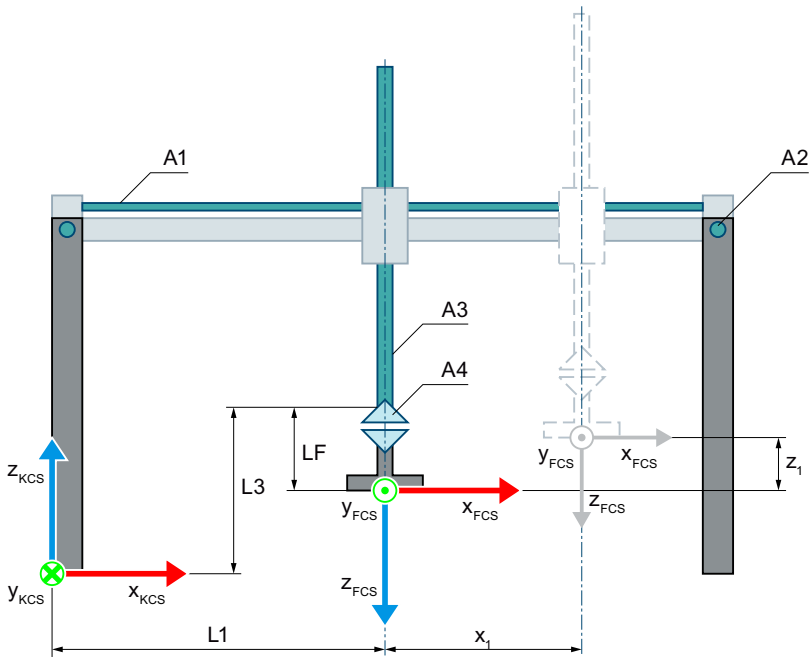

Zero position of the kinematics

L1 At zero position of the axis A1:
Distance of the FCS to the kinematics zero point (KZP) in x direction of the KCS

L3 At zero position of the axis A3:
Distance of the FCS to the KZP and flange length LF in z direction of the KCS

LF Flange length before the FCS in the z direction of the KCS

Deflection of the kinematics

x_1 Deflection of the axis A1 in the positive x direction

z_1 Deflection of the axis A3 in the positive z direction

Legend for representation of the kinematics (Page 2160)

The graphic below shows the following in the top view (xy plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the axes A1 and A2 is indicated (dashed)

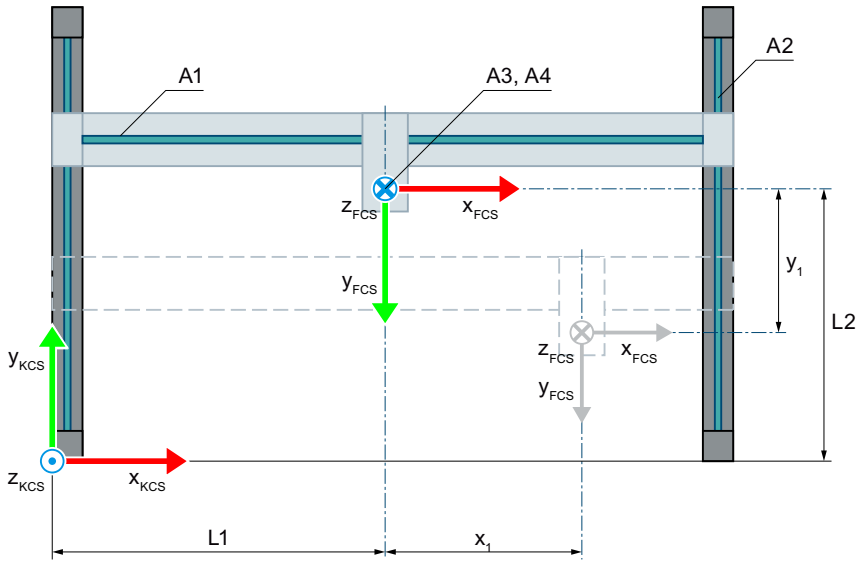

Zero position of the kinematics

L1 At zero position of the axis A1:
Distance of the FCS to the KZP in x direction of the KCS

L2 At zero position of the axis A2:
Distance of the FCS to the KZP in y direction of the KCS

Deflection of the kinematics

x_1 Deflection of the axis A1 in the positive x direction

y_1 Deflection of the axis A2 in negative y direction

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. The flange coordinate system (FCS) is located at distance LF from the zero position of the axis A2.

The position 0.0 on the respective interconnected technology object defines the zero positions of the axes A1, A2 and A3 in the KCS. You define the distances of the zero positions of the axes A1, A2 and A3 to the kinematics zero point with lengths L1, L2 and L3. At the zero position of axis A4, the x axis of the FCS points in the direction of the x axis of the KCS.

Transformation area

The kinematics transformation covers the entire traversing range (Page 2239) of the axes.

Portal tags

Portal 2D

You define the 2D portal kinematics using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	1	Portal 2D
	2	Portal 2D with orientation
<TO>.Kinematics.Parameter[1]	-1.0E12 to 1.0E12	Distance L1 of the zero position of the axis A1 to the kinematics zero point (KZP) in x direction of the kinematics coordinate system (KCS)
<TO>.Kinematics.Parameter[2]	-1.0E12 to 1.0E12	Distance of the flange coordinate system from the axis A2 in the negative z direction of the KCS
<TO>.Kinematics.Parameter[3]	-1.0E12 to 1.0E12	Distance L2 of the zero position of the axis A2 to the KZP in z direction of the KCS

Portal 3D

You define the 3D portal kinematics using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	3	Portal 3D
	4	Portal 3D with orientation
<TO>.Kinematics.Parameter[1]	-1.0E12 to 1.0E12	Distance L1 of the zero position of the axis A1 to the KZP in x direction of the KCS
<TO>.Kinematics.Parameter[2]	-1.0E12 to 1.0E12	Distance L2 of the zero position of the axis A2 to the KZP in y direction of the KCS
<TO>.Kinematics.Parameter[3]	-1.0E12 to 1.0E12	Distance of the flange coordinate system from the axis A3 in the negative z direction of the KCS
<TO>.Kinematics.Parameter[4]	-1.0E12 to 1.0E12	Distance L3 of the zero position of the axis A3 to the KZP in z direction of the KCS

See also

Tags of the kinematics technology object (Page 2308)

Roller picker

Roller picker 2D

The kinematics "Roller picker 2D" supports two axes and two degrees of freedom. The following graphic shows the principal configuration and the typical working area of the kinematics:

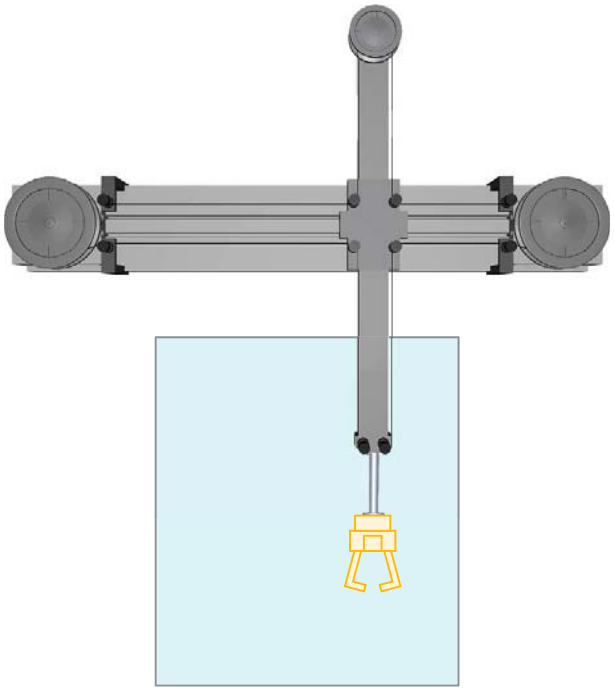

The kinematics consists of two rotary axes A1 and A2 and a system of guide rollers. If both axes A1 and A2 rotate in the same direction, the flange moves horizontally in the x direction of the KCS. If both axes A1 and A2 rotate in opposite directions, the flange moves vertically in the z direction of the KCS. The kinematics enables a rectangular working area.

Coordinate systems and zero position

The graphic below shows the following in the front view:

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The deflection of the kinematics is indicated (dashed)

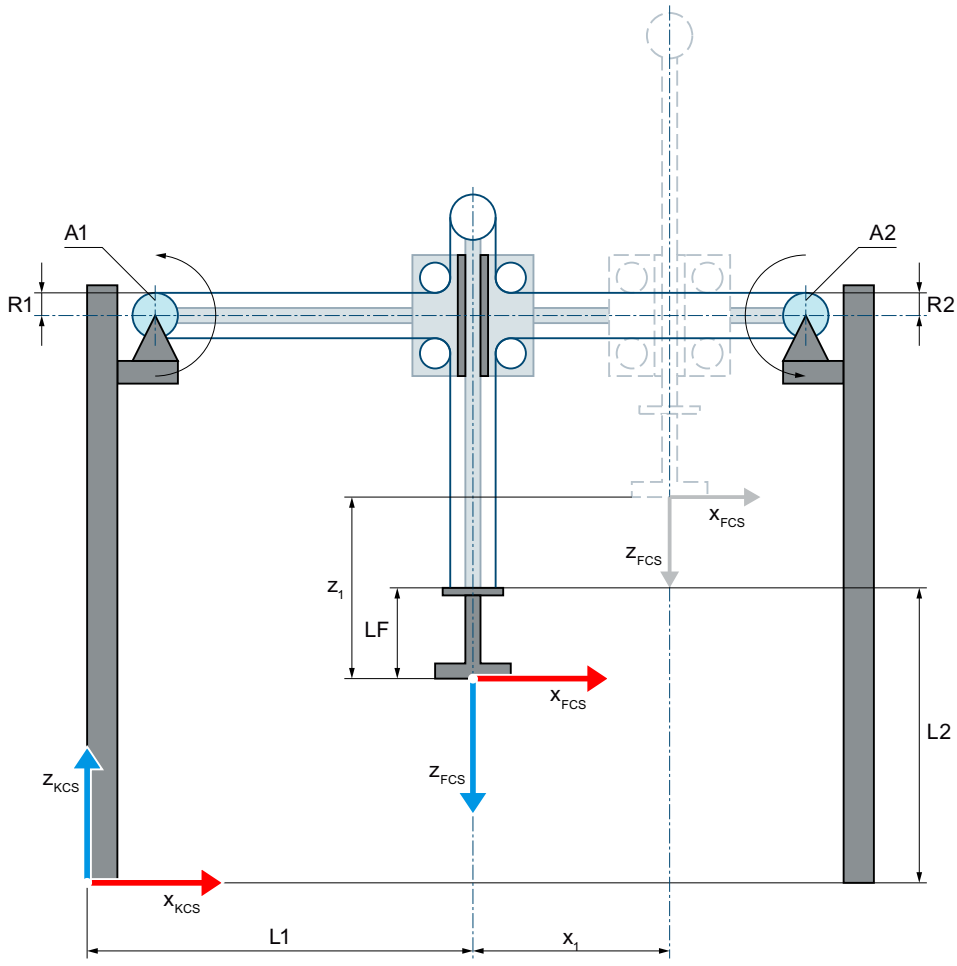

Zero position of the kinematics

L1	With zero position of the axes A1 and A2: Distance of the FCS to the KZP in x direction of the KCS
L2	With zero position of the axes A1 and A2: Distance of the FCS to the KZP and flange length LF in z direction of the KCS
LF	Flange length before the FCS in the z direction of the KCS
R1	Cam radius for axis A1
R2	Cam radius for axis A2

Deflection of the kinematics

x_1	Deflection of the kinematics in the positive x direction
z_1	Deflection of the kinematics in the positive z direction

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. The flange coordinate system (FCS) is located between axes A1 and A2.

The position 0.0 on the respective interconnected technology object defines the zero position of the axes A1 and A2. You define the position of the FCS for zero position of the axes A1 and A2 using distances L1 and L2. You shift the FCS in the negative z direction of the KCS using length LF.

Transformation area

The kinematics transformation covers the entire traversing range (Page 2239) of the axes.

Roller picker 2D with orientation

The kinematics "Roller picker 2D with orientation" supports three axes and three degrees of freedom. The following graphic shows the principal configuration and the typical working area of the kinematics:

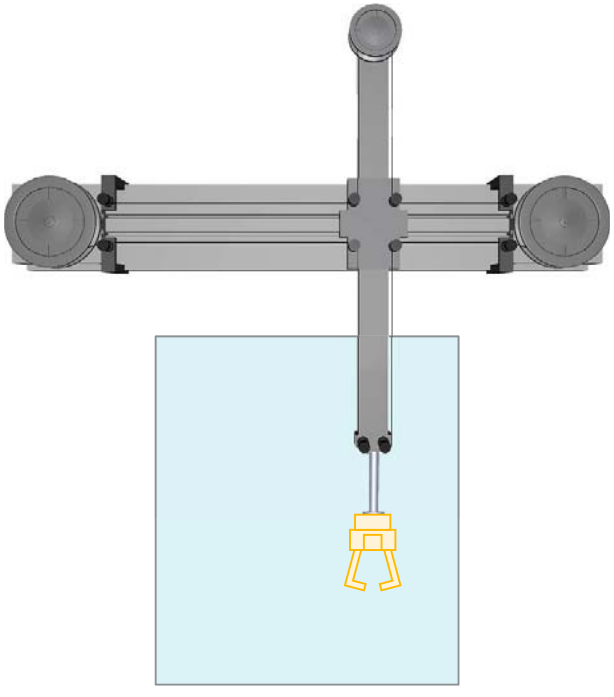

The kinematics consists of a system made up of guide rollers and the following axes:

- Two rotary axes A1 and A2
- A rotational axis A4 (orientation axis) with rotation around z in the KCS

If both axes A1 and A2 rotate in the same direction, the flange moves horizontally in the x direction of the KCS. If both axes A1 and A2 rotate in opposite directions, the flange moves vertically in the z direction of the KCS. The kinematics enables a rectangular working area. The orientation axis A4 enables rotation of the tool.

Coordinate systems and zero position

The graphic below shows the following in the front view:

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The deflection of the kinematics is indicated (dashed)

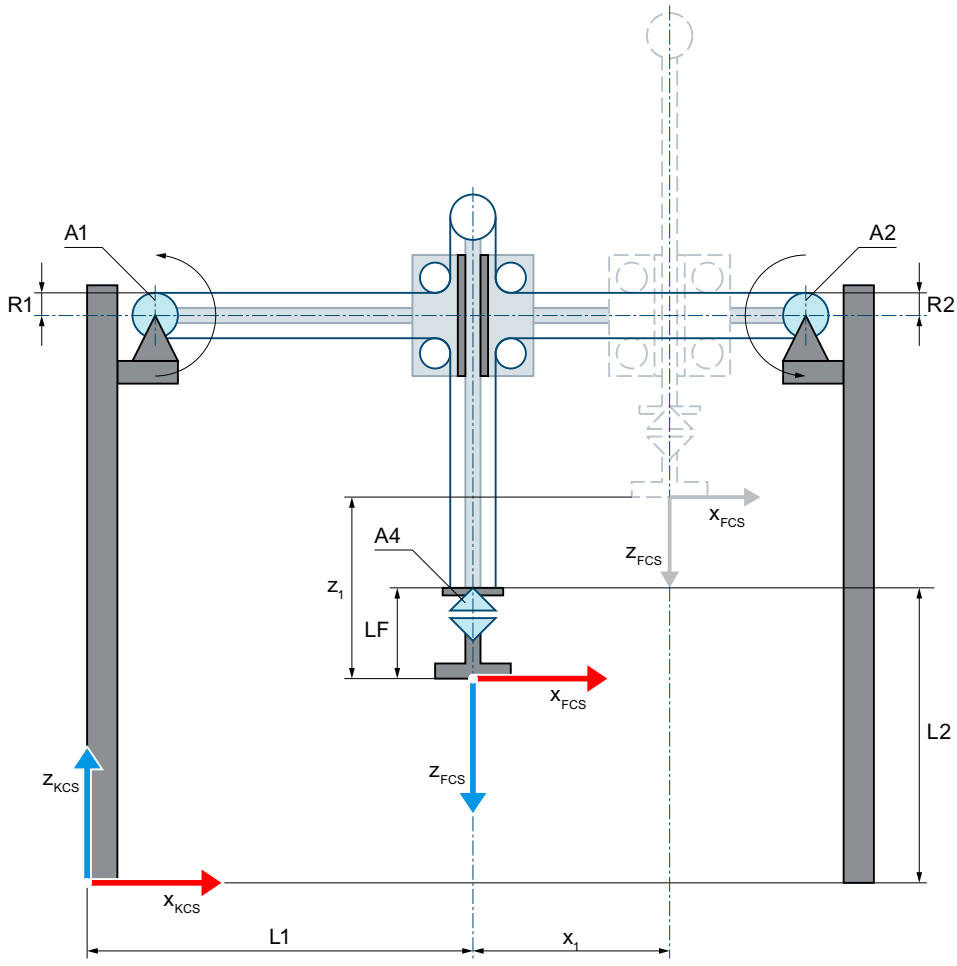

Zero position of the kinematics

L1	With zero position of the axes A1 and A2: Distance of the FCS to the KZP in x direction of the KCS
L2	With zero position of the axes A1 and A2: Distance of the FCS to the KZP and flange length LF in z direction of the KCS
LF	Flange length before the FCS in the z direction of the KCS
R1	Cam radius for axis A1
R2	Cam radius for axis A2

Deflection of the kinematics

x_1	Deflection of the kinematics in the positive x direction
z_1	Deflection of the kinematics in the positive z direction

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. The flange coordinate system (FCS) is located between axes A1 and A2.

The position 0.0 on the respective interconnected technology object defines the zero position of the axes A1 and A2. You define the position of the FCS for zero position of the axes A1 and A2 using lengths L1 and L2. You shift the FCS in the negative z direction of the KCS using length LF. At the zero position of axis A4, the x axis of the FCS points in the direction of the x axis of the KCS.

Transformation area

The kinematics transformation covers the entire traversing range (Page 2239) of the axes.

Roller picker 3D (vertical)

The kinematics "Roller picker 3D (vertical)" supports three axes and three degrees of freedom. The following graphic shows the principal configuration and the typical working area of the kinematics:

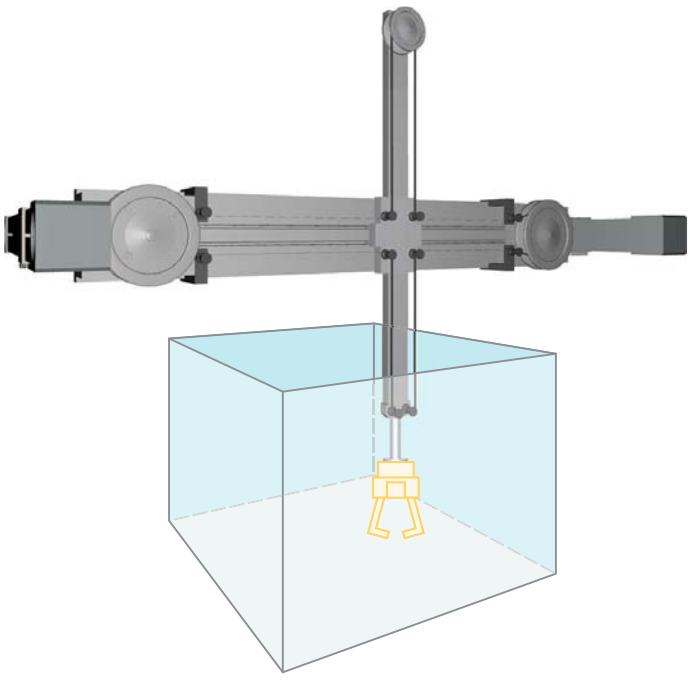

The kinematics consists of a system made up of guide rollers and the following axes:

- Two rotary axes A1 and A2
- A linear axis A3 in y direction of the KCS

If both axes A1 and A2 rotate in the same direction, the flange moves horizontally in the x direction of the KCS. If both axes A1 and A2 rotate in opposite directions, the flange moves vertically in the z direction of the KCS. The linear portal axis A3 moves the system of guide rollers horizontally in y direction of the KCS. The kinematics enables a cuboid working area.

Coordinate systems and zero position

The graphic below shows the following in the front view (xz plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The deflection of the kinematics is indicated (dashed)

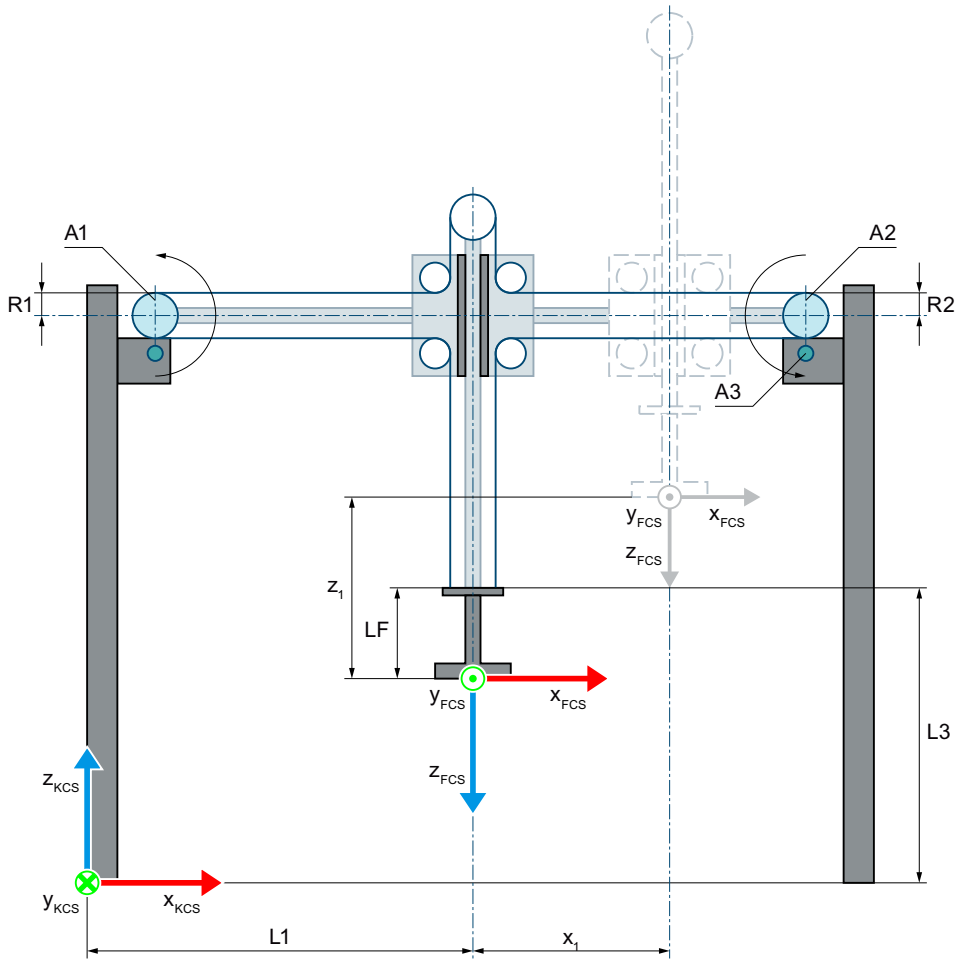

Zero position of the kinematics

L1	With zero position of the axes A1 and A2: Distance of the FCS to the KZP in x direction of the KCS
L3	With zero position of the axes A1 and A2: Distance of the FCS to the KZP and flange length LF in z direction of the KCS
LF	Flange length before the FCS in the z direction of the KCS
R1	Cam radius for axis A1
R2	Cam radius for axis A2

Deflection of the kinematics

x_1	Deflection of the kinematics in the positive x direction
z_1	Deflection of the kinematics in the positive z direction

Legend for representation of the kinematics (Page 2160)

The graphic below shows the following in the top view (xy plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The deflection of the kinematics is indicated (dashed)

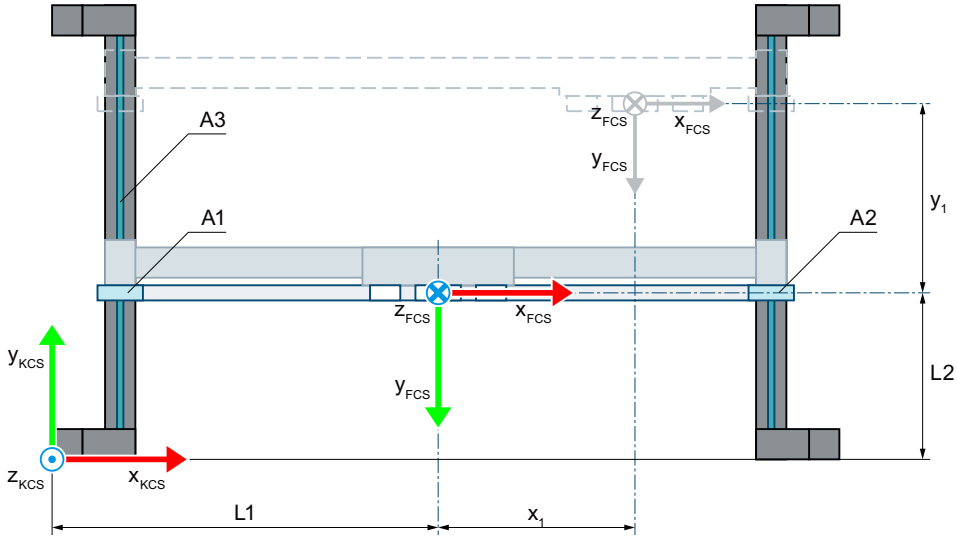

Zero position of the kinematics

L1	With zero position of the axes A1 and A2: Distance of the FCS to the KZP in x direction of the KCS
L2	At zero position of the axis A3: Distance of the FCS to the KZP in y direction of the KCS
R1	Cam radius for axis A1
R2	Cam radius for axis A2

Deflection of the kinematics

x_1	Deflection of the kinematics in the positive x direction
y_1	Deflection of the kinematics in the positive y direction

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. The flange coordinate system (FCS) is located between axes A1 and A2.

The position 0.0 on the respective interconnected technology object defines the zero position of the axes A1 and A2 and the zero position of the axis A3 in the KCS. You define the distance of the zero position of the axis A3 to the KZP in y direction of the KCS using length L2.

You define the position of the FCS for zero position of the axes A1 and A2 using lengths L1 and L3. You shift the FCS in the negative z direction of the KCS using length LF.

Transformation area

The kinematics transformation covers the entire traversing range (Page 2239) of the axes.

Roller picker 3D with orientation (vertical)

The kinematics "Roller picker 3D with orientation (vertical)" supports four axes and four degrees of freedom. The following graphic shows the principal configuration and the typical working area of the kinematics:

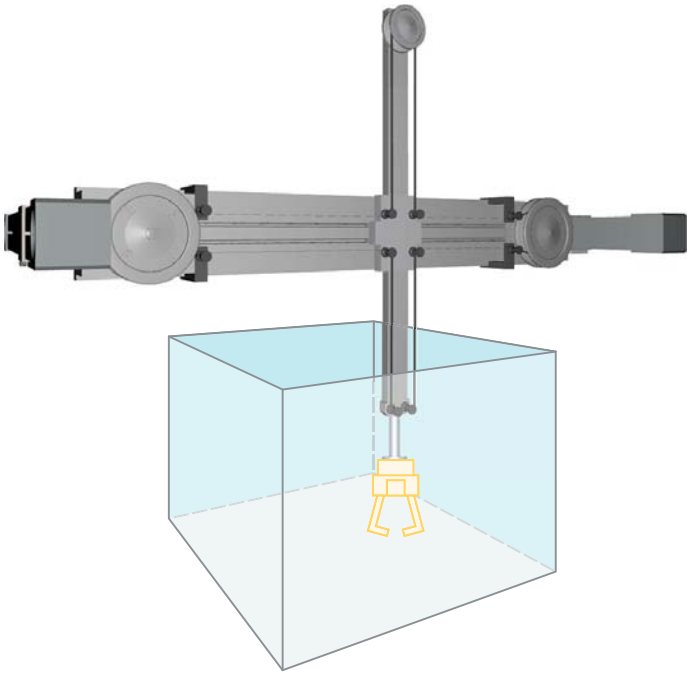

The kinematics consists of a system made up of guide rollers and the following axes:

- Two rotary axes A1 and A2
- A linear axis A3 in y direction of the KCS
- A rotational axis A4 (orientation axis) with rotation around z in the KCS

If both axes A1 and A2 rotate in the same direction, the flange moves horizontally in the x direction of the KCS. If both axes A1 and A2 rotate in opposite directions, the flange moves vertically in the z direction of the KCS. The linear portal axis A3 moves the system of guide rollers horizontally in y direction of the KCS. The kinematics enables a cuboid working area. The orientation axis A4 enables rotation of the tool.

Coordinate systems and zero position

The graphic below shows the following in the front view (xz plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The deflection of the kinematics is indicated (dashed)

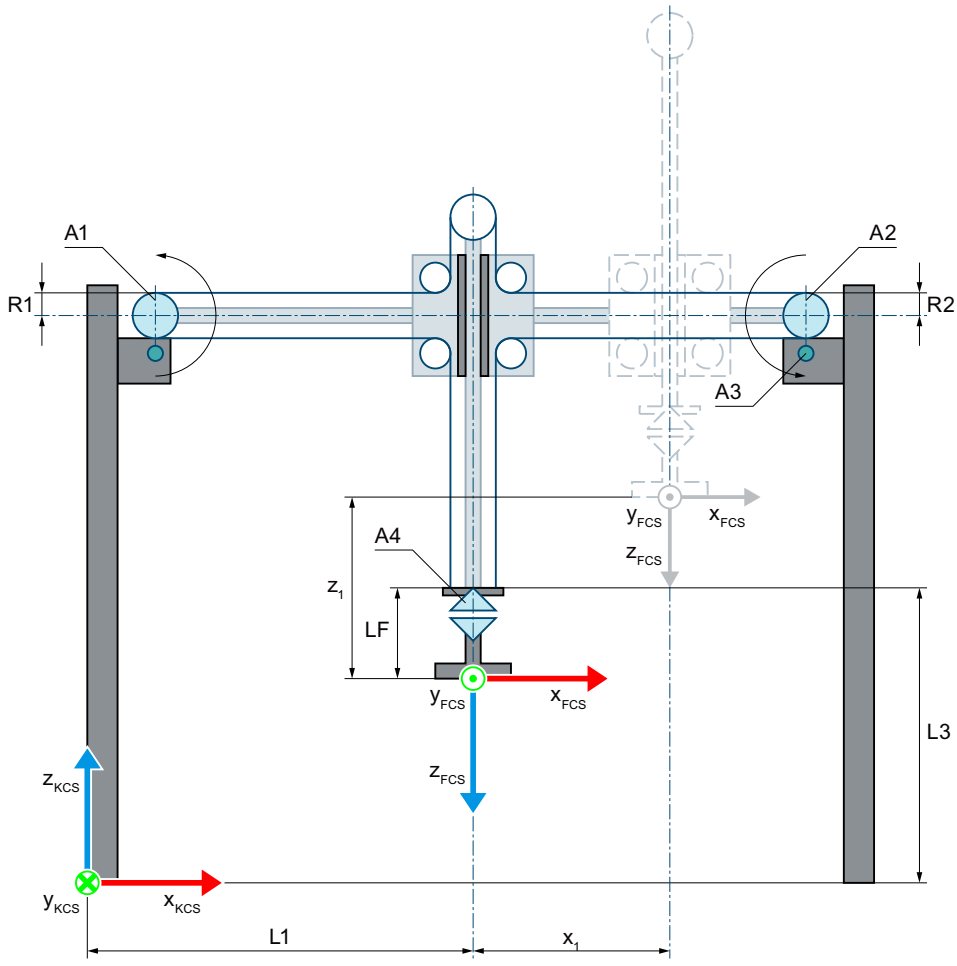

Zero position of the kinematics

L1	With zero position of the axes A1 and A2: Distance of the FCS to the KZP in x direction of the KCS
L3	With zero position of the axes A1 and A2: Distance of the FCS to the KZP and flange length LF in z direction of the KCS
LF	Flange length before the FCS in the z direction of the KCS
R1	Cam radius for axis A1
R2	Cam radius for axis A2

Deflection of the kinematics

x_1	Deflection of the kinematics in the positive x direction
z_1	Deflection of the kinematics in the positive z direction

Legend for representation of the kinematics (Page 2160)

The graphic below shows the following in the top view (xy plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The deflection of the kinematics is indicated (dashed)

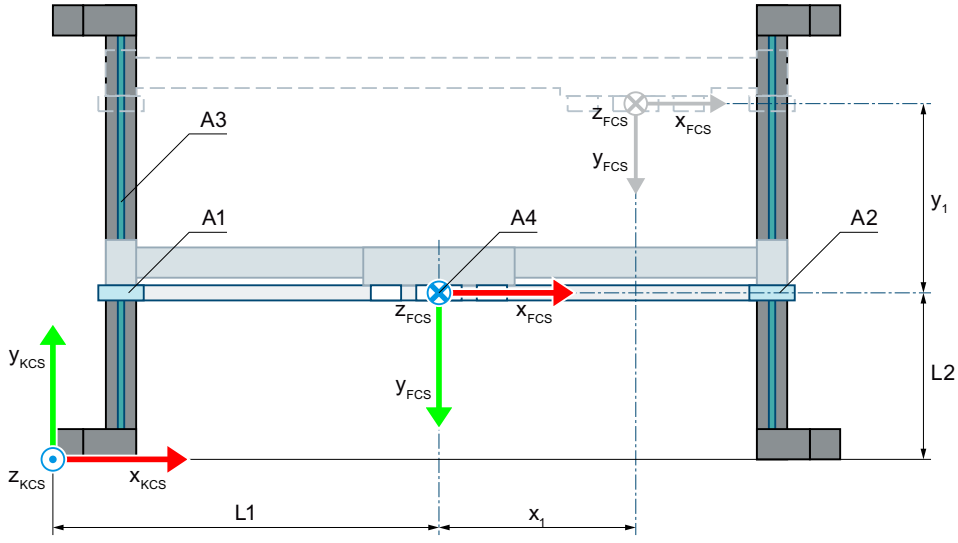

Zero position of the kinematics

L1	With zero position of the axes A1 and A2: Distance of the FCS to the KZP in x direction of the KCS
L2	At zero position of the axis A3: Distance of the FCS to the KZP in y direction of the KCS
R1	Cam radius for axis A1
R2	Cam radius for axis A2

Deflection of the kinematics

x_1	Deflection of the kinematics in the positive x direction
y_1	Deflection of the kinematics in the positive y direction

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. The flange coordinate system (FCS) is located between axes A1 and A2.

The position 0.0 on the respective interconnected technology object defines the zero position of the axes A1 and A2 and the zero position of the axis A3 in the KCS. You define the distance of the zero position of the axis A3 to the KZP in y direction of the KCS using length L2.

You define the position of the FCS for zero position of the axes A1 and A2 using lengths L1 and L3. You shift the FCS in the negative z direction of the KCS using length LF. At the zero position of axis A4, the x axis of the FCS points in the direction of the x axis of the KCS.

Transformation area

The kinematics transformation covers the entire traversing range (Page 2239) of the axes.

Roller picker 3D with orientation (horizontal)

The kinematics "Roller picker 3D with orientation (horizontal)" supports four axes and four degrees of freedom. The following graphic shows the principal configuration and the typical working area of the kinematics:

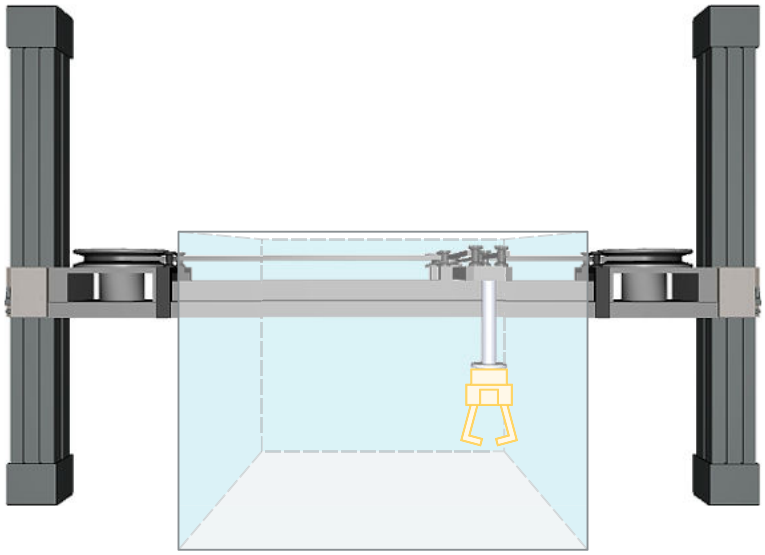

The kinematics consists of a system made up of guide rollers and the following axes:

- Two rotary axes A1 and A2
- A linear axis A3 in z direction of the KCS
- A rotational axis A4 (orientation axis) with rotation around z in the KCS

If both axes A1 and A2 rotate in the same direction, the flange moves horizontally in the x direction of the KCS. If both axes A1 and A2 rotate in opposite directions, the flange moves horizontally in the y direction of the KCS. The linear portal axis A3 moves the system of guide rollers vertically in z direction of the KCS. The kinematics enables a cuboid working area. The orientation axis A4 enables rotation of the tool.

Coordinate systems and zero position

The graphic below shows the following in the front view (xz plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The deflection of the kinematics is indicated (dashed)

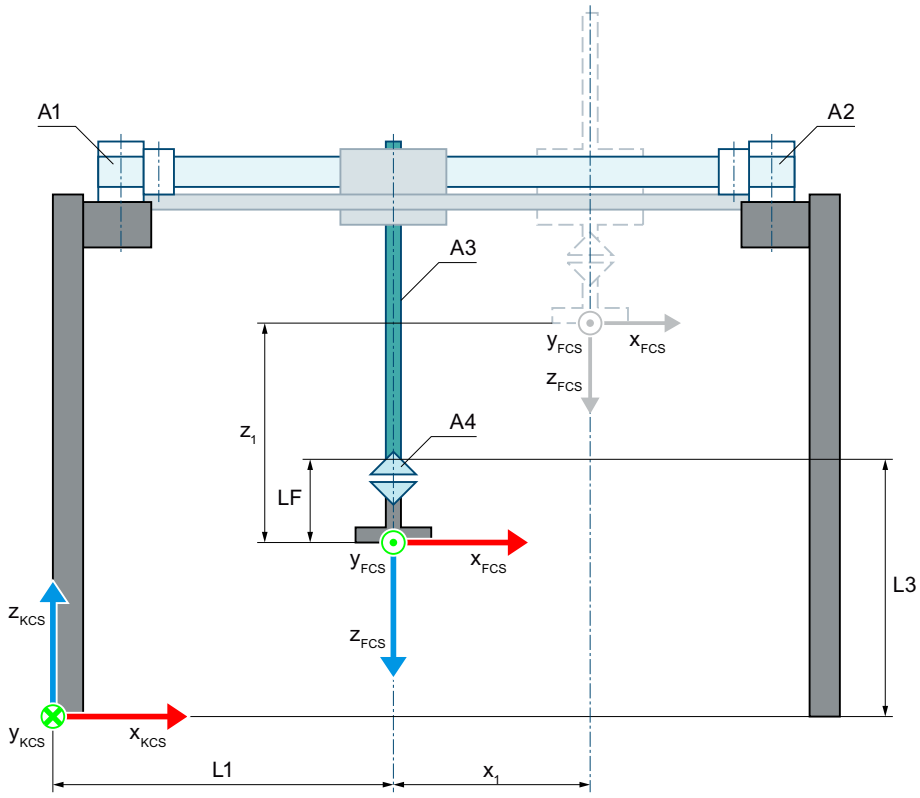

Zero position of the kinematics

L1	With zero position of the axes A1 and A2: Distance of the FCS to the KZP in x direction of the KCS
L3	At zero position of the axis A3: Distance of the FCS to the KZP and flange length LF in z direction of the KCS
LF	Flange length before the FCS in the z direction of the KCS
R1	Cam radius for axis A1
R2	Cam radius for axis A2

Deflection of the kinematics

x_1	Deflection of the kinematics in the positive x direction
z_1	Deflection of the kinematics in the positive z direction

Legend for representation of the kinematics (Page 2160)

The graphic below shows the following in the top view (xy plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The deflection of the kinematics is indicated (dashed)

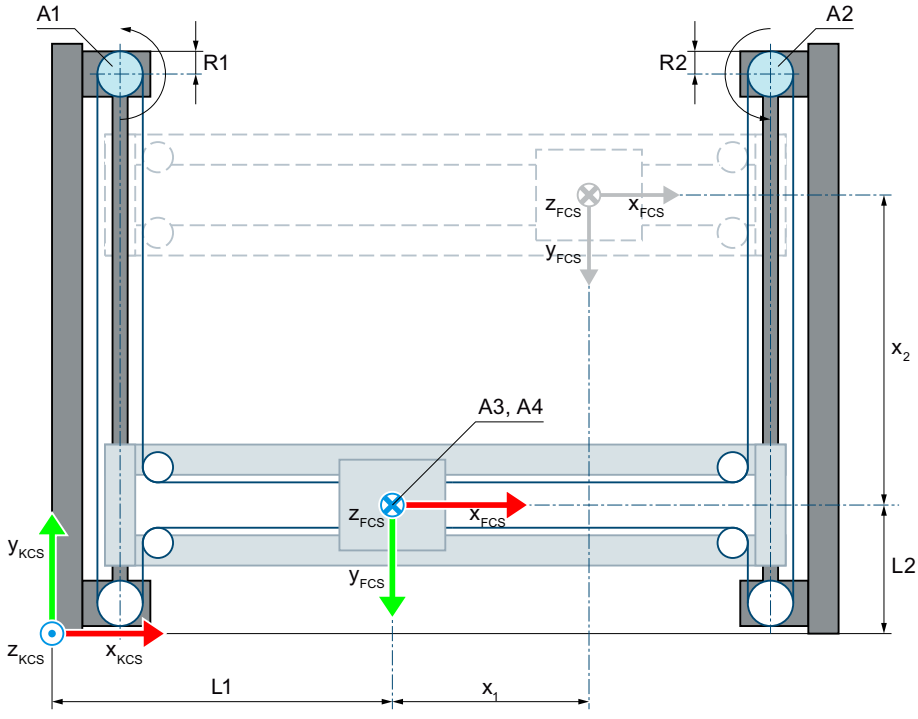

Zero position of the kinematics

L1	With zero position of the axes A1 and A2: Distance of the FCS to the KZP in x direction of the KCS
L2	With zero position of the axes A1 and A2: Distance of the FCS to the KZP in y direction of the KCS
R1	Cam radius for axis A1
R2	Cam radius for axis A2

Deflection of the kinematics

x_1	Deflection of the kinematics in the positive x direction
y_1	Deflection of the kinematics in the positive y direction

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. The flange coordinate system (FCS) is located between axes A1 and A2.

The position 0.0 on the respective interconnected technology object defines the zero position of the axes A1 and A2 and the zero position of the axis A3 in the KCS. You define the distance of the zero position of the axis A3 to the KZP in y direction of the KCS using length L2.

You define the position of the FCS for zero position of the axes A1 and A2 using lengths L1 and L3. You shift the FCS in the negative z direction of the KCS using length LF.

Transformation area

The kinematics transformation covers the entire traversing range (Page 2239) of the axes.

Roller picker tags

Roller picker 2D

You define the 2D delta picker kinematics systems using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	5	Roller picker 2D
	6	Roller picker 2D with orientation
<TO>.Kinematics.Parameter[1]	-1.0E12 to 1.0E12	With zero position of the axes A1 and A2: Distance L1 of the FCS to the KZP in x direction of the kinematics coordinate system (KCS)
<TO>.Kinematics.Parameter[2]	0.001 to 1.0E12	Cam radius R1 for axis 1
<TO>.Kinematics.Parameter[3]	0.001 to 1.0E12	Cam radius R2 for axis 2
<TO>.Kinematics.Parameter[4]	-1.0E12 to 1.0E12	Flange length LF before the flange coordinate system (FCS) in the negative z direction of the KCS
<TO>.Kinematics.Parameter[5]	-1.0E12 to 1.0E12	With zero position of the axes A1 and A2: Distance L2 of the FCS to the KZP in z direction of the KCS

Roller picker 3D

You define the 3D roller picker kinematics systems using the following tags of the technology object:

Tags	Values	Description	
<TO>.Kinematics.TypeOfKinematics	7	Roller picker 3D (vertical)	
	8	Roller picker 3D with orientation (vertical)	
	9	Roller picker 3D with orientation (horizontal)	
<TO>.Kinematics.Parameter[1]	-1.0E12 to 1.0E12	With zero position of the axes A1 and A2: Distance L1 of the FCS to the KZP in x direction of the KCS	
<TO>.Kinematics.Parameter[2]	0.001 to 1.0E12	Cam radius R1 for axis 1	
<TO>.Kinematics.Parameter[3]	0.001 to 1.0E12	Cam radius R2 for axis 2	
<TO>.Kinematics.Parameter[4]	-1.0E12 to 1.0E12	Flange length LF before the FCS in the negative z direction of the KCS	
<TO>.Kinematics.Parameter[5]	-1.0E12 to 1.0E12	Roller picker vertical	Distance L2 of the zero position of the axis A3 to the KZP in y direction of the KCS
		Roller picker horizontal	With zero position of the axes A1 and A2: Distance L2 of the FCS to the kinematics zero point (KZP) in y direction of the KCS

Tags	Values	Description	
<TO>.Kinematics.Parameter[6]	-1.0E12 to 1.0E12	Roller picker vertical	With zero position of the axes A1 and A2: Distance L3 of the FCS to the KZP in z direction of the KCS
		Roller picker horizontal	Distance L3 of the zero position of the axis A3 to the KZP in z direction of the KCS

SCARA

SCARA 3D with orientation

The kinematics "SCARA (Selective Compliance Assembly Robot Arm) 3D with orientation" supports four axes and four degrees of freedom. The axes are configured as serial kinematics.

The following graphic shows the principal configuration and the typical working area of the kinematics:

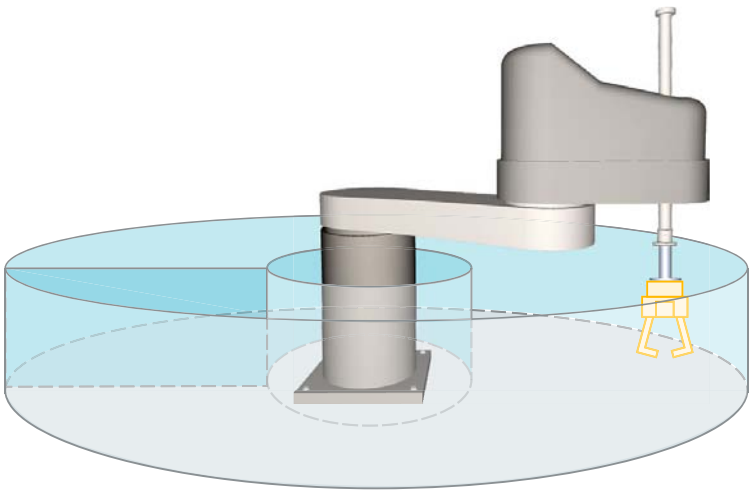

The kinematics consists of the following axes:

- A rotary axis A1 with rotation around the z axis of the kinematics coordinate system (KCS)
- A rotary axis A2 at distance L2 to A1 with rotation around z of the KCS
- A linear axis A3 at distance L3 to A2 with motion in z direction of the KCS
- A rotational axis A4 (orientation axis) with rotation around z in the KCS

The kinematics consists of a base and two levers for horizontal alignment, which are connected by revolute joints (axis A1 and A2). A linear axis (axis A3) is fastened to the end of the articulated arm for the vertical alignment. The tool is fastened to the end of the linear axis. The orientation axis A4 enables the rotary motion of the tool.

Coordinate systems and zero position

The graphic below shows the following in the side view (xz plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics

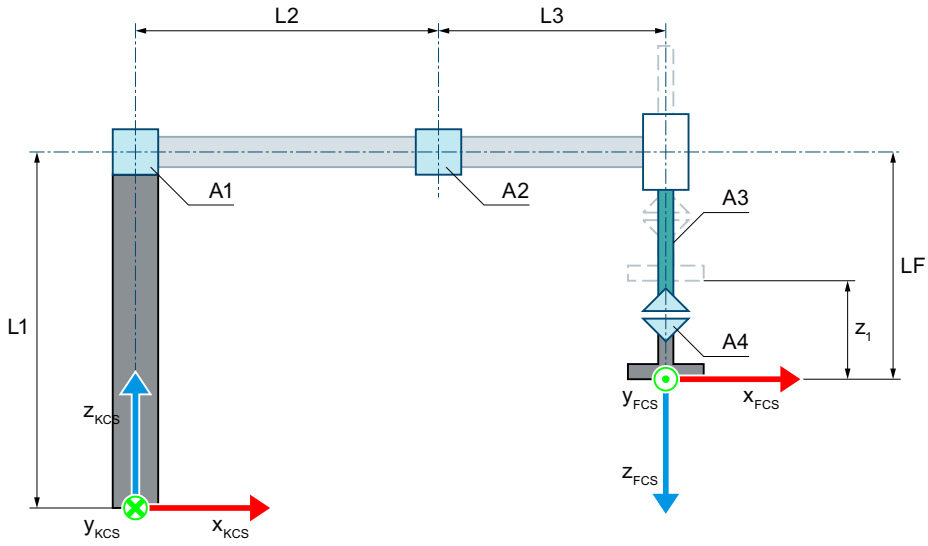

Zero position of the kinematics

L1	Distance of the axis A1 to the KZP in z direction of the KCS
L2	Distance of the axis A2 to the axis A1 in x direction of the KCS
L3	Distance of the axis A3 to the axis A2 in x direction of the KCS
LF	Distance of the FCS to the axis A2 in z direction of the FCS

Deflection of the kinematics

z_1	Deflection of the axis A3 in the positive direction

Legend for representation of the kinematics (Page 2160)

The graphic below shows the following in the top view (xy plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the kinematics is indicated (dashed)

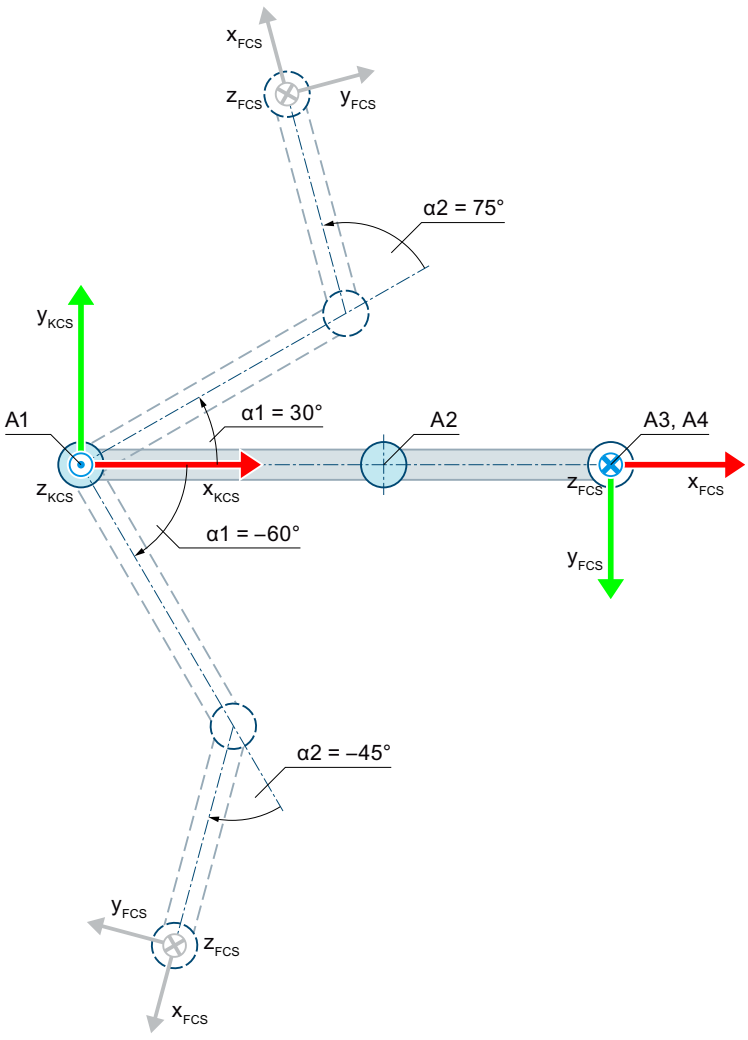

Zero position of the kinematics

Deflection of the kinematics in the positive direction when α1 = 30.0° with positive joint position when α2 = 75.0°

Deflection of the kinematics in the negative direction when α1 = -60.0° with negative joint position when α2 = -45.0°

α1 Deflection of the axis A1 in positive direction when α1 = 30.0°

Deflection of the axis A1 in negative direction when α1 = -60.0°

α2 The deflection of the axis A2 in the positive direction when α2 = 75.0° produces a positive joint position.

The deflection of the axis A2 in negative direction when α2 = -45.0° produces a negative joint position.

Legend for representation of the kinematics (Page 2160)

The KCS with the kinematics zero point (KZP) is located at the base of the kinematics. The flange coordinate system (FCS) is located at the end of the axis A3.

The following table shows the zero position of the axes:

Axis	Zero position
A1 and A2	The kinematics is elongated in the x_{KCS} direction.
A3	The FCS is located at distance L1-LF from the KCS in z direction.
A4	At the zero position of axis A1 and A2, the x axis of the FCS points in the direction of the x axis of the KCS.

Compensation of mechanical axis couplings

You can configure the following mechanical coupled axes for the kinematics:

- Mechanical coupling of axis A1 to axis A2
- Mechanical coupling of axis A4 to axis A3

The kinematics transformation compensates for the configured mechanical axis couplings. With a coupling factor > 0.0, the kinematics transformation assumes that a positive motion of the axis A1 leads to a negative motion on the axis A2. The axis coupling between axis A4 and axis A3 is implemented as a leadscrew pitch. With a coupling factor of 1.0, 360.0° on axis A4 corresponds to a distance of -1.0 mm on axis A3.

Transformation area

The kinematics transformation covers the following traversing range (Page 2239) of the axes:

- Axis A1: -180.0° ≤ α1 ≤ 180.0°
- Axis A2: -180.0° ≤ α2 ≤ 180.0°
- Axis A3: No limiting
- Axis A4: No limiting
  An angle greater than 360° can be defined for the orientation. But coordinate A of the tool center point (TCP) is mapped in the range -180° to +180°.

Note

Singular positions

The kinematics have singular positions (Page 2241).

A singular position occurs when the zero point of the flange coordinate system (FCS) is located on the z-axis of the kinematics coordinate system (KCS). Inverse transformation is not possible in this area. This position may result, e.g. in the event of suspended installation if the lengths L2 and L3 are the same size.

The graphic below shows examples of a movement in the direction of the singular joint position:

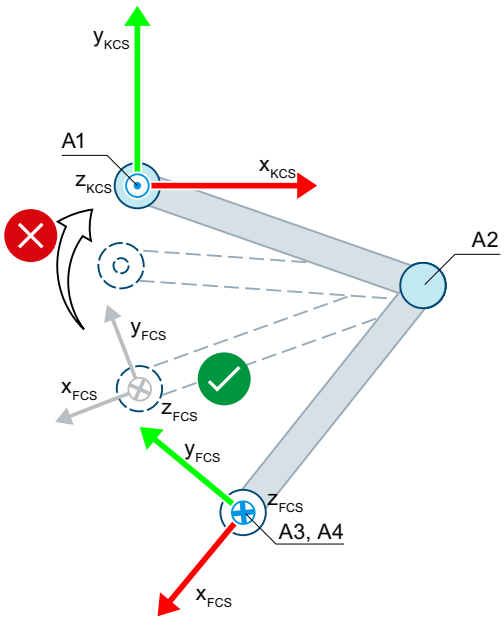

Permissible joint position

Invalid joint position for transformation with L2 = L3

SCARA tags

You define the SCARA kinematics using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	10	SCARA 3D with orientation
<TO>.Kinematics.Parameter[1]	-1.0E12 to 1.0E12	Distance of the axis A1 from the kinematics zero point in z direction of the kinematics coordinate system (KCS)
<TO>.Kinematics.Parameter[2]	0.001 to 1.0E12	Distance L2 of the axis A2 from the axis A1 in x direction of the KCS
<TO>.Kinematics.Parameter[3]	-	Mechanical axis coupling of axis A1 to axis A2 present/not present
	0	Not present
	1	Present
<TO>.Kinematics.Parameter[4]	-1.0E12 to 1.0E12	Mechanical axis coupling factor of axis A1 to axis A2
<TO>.Kinematics.Parameter[5]	0.001 to 1.0E12	Distance L3 of the axis A3 from the axis A2 in x direction of the KCS
<TO>.Kinematics.Parameter[6]	-	Mechanical axis coupling of axis A4 to axis A3 present/not present
	0	Not present
	1	Present
<TO>.Kinematics.Parameter[7]	-1.0E12 to 1.0E12	Mechanical axis coupling factor of axis A4 to axis A3
<TO>.Kinematics.Parameter[8]	-1.0E12 to 1.0E12	Distance of the flange coordinate system from the axis A2 in the negative z direction of the KCS

Articulated arm

Articulated arm 2D

The kinematics "Articulated arm 2D" supports two axes and two degrees of freedom. The axes are configured as serial kinematics with forced coupling of the flange system.

The following graphic shows the principal configuration and the typical working area of the kinematics:

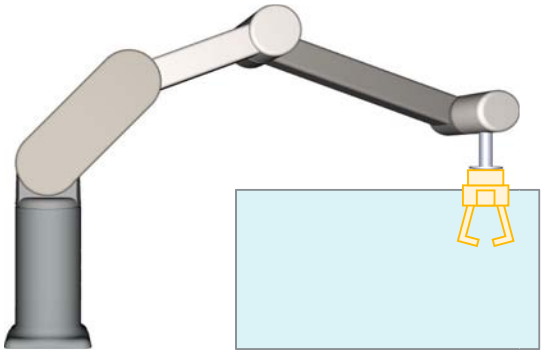

The kinematics consists of the following axes:

- A rotary axis A1 with the distances L1 in z direction of the KCS and L2 in x direction of the KCS to the kinematics zero point
- A rotary axis A2 at distance L3 to axis A1

The kinematics consists of a base and articulated arms, which are connected by revolute joints (axes A1, A2). The axes A1 and A2 move the articulated arm vertically. Through a forced coupling between the axis A2 and the flange system, the z axis of the FCS always points in the negative z direction of the KCS.

Coordinate systems and zero position

The graphic below shows the following in the side view:

- The position of the axes and of the forced coupler point
- The position of coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the kinematics is indicated (dashed)

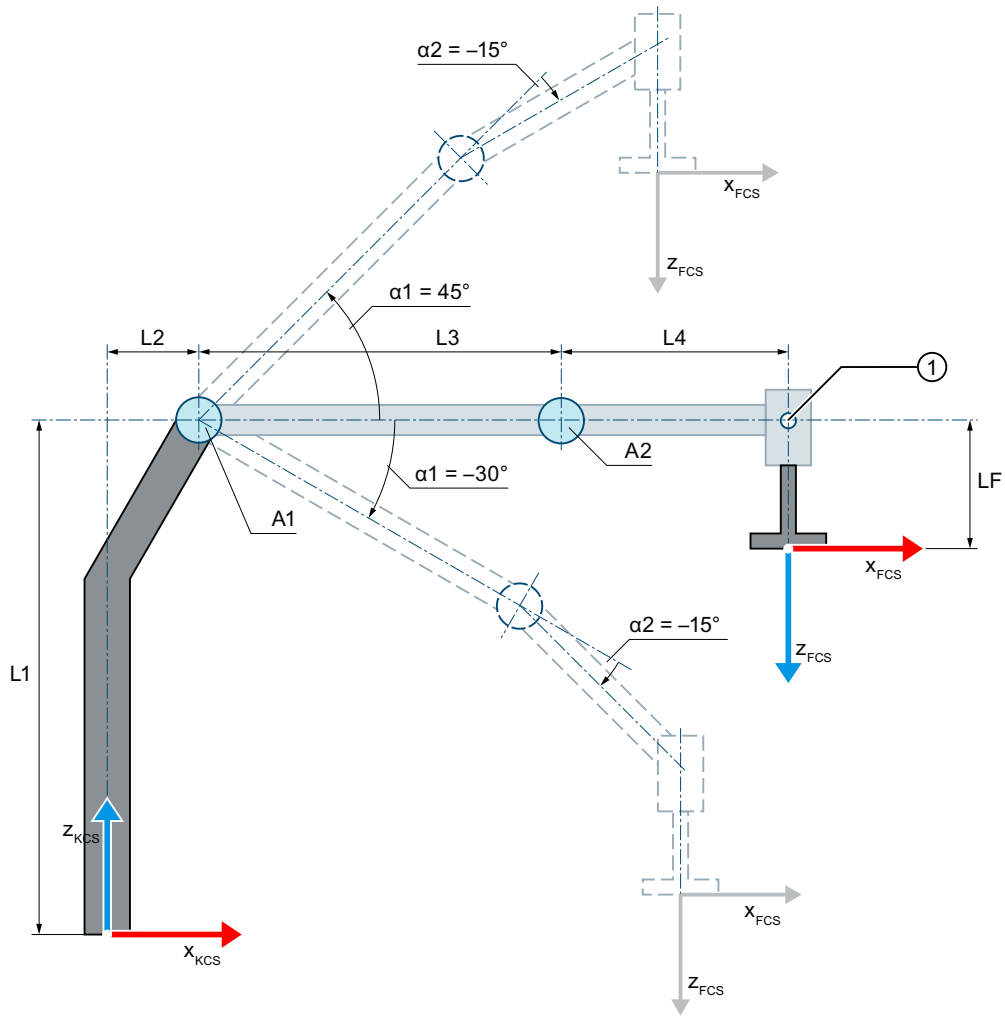

① Forced coupler point

Zero position of the kinematics

- L1 Distance of the axis A1 to the kinematics zero point (KZP) in z direction of the KCS
- L2 Distance of the axis A1 to the KZP in x direction of the KCS
- L3 Distance of the axis A2 to the axis A1 in x direction of the KCS
- L4 Distance of the forced coupler point to the axis A2 in x direction of the KCS
- LF Distance of the FCS to the forced coupler point in z direction of the FCS

Deflection of the kinematics

- α1 Positive deflection of the axis A1 when α1 = 45.0°
  Negative deflection of the axis A1 when α1 = -30.0°
- α2 Negative deflection of the axis A2 when α2 = -15.0°

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. You define the position of the axis A1 relative to the KZP using lengths L1 and L2. The axis A2 is located at distance L3 in x direction of the KCS from the axis A1.

The flange coordinate system (FCS) is located at the following distances from the axis A2 and the forced coupler point:

- Distance L4 to the axis A2 in x direction of the KCS
- Distance LF to the forced coupler point in the negative z direction of the KCS

The axis A2 and the flange system are force-coupled. With the force coupling, the z axis of the FCS always points in negative z direction of the KCS. The forced coupler point is located at distance L4 in x direction of the KCS from the axis A2.

The following table shows the zero position of the axes:

Axis	Zero position
A1	Length L3 points in x direction of the KCS.
A2	At zero position of the axis A1, the length L4 points in x direction of the KCS.

Compensation of mechanical axis couplings

For the kinematics, you can configure a mechanical axis coupling of axis A1 to axis A2. The kinematics transformation compensates for the configured mechanical axis coupling. With a coupling factor > 0.0, the kinematics transformation assumes that a positive motion of the axis A1 leads to a negative motion on the axis A2.

Transformation area

The kinematics transformation covers the following traversing range (Page 2239) of the axes:

- Axis A1: -180.0° ≤ α1 < 180.0°
- Axis A2: -180.0° ≤ α2 < 180.0°

Articulated arm 2D with orientation

The kinematics "Articulated arm 2D with orientation" supports three axes and three degrees of freedom. The axes are configured as serial kinematics with forced coupling of the flange system.

The following graphic shows the principal configuration and the typical working area of the kinematics:

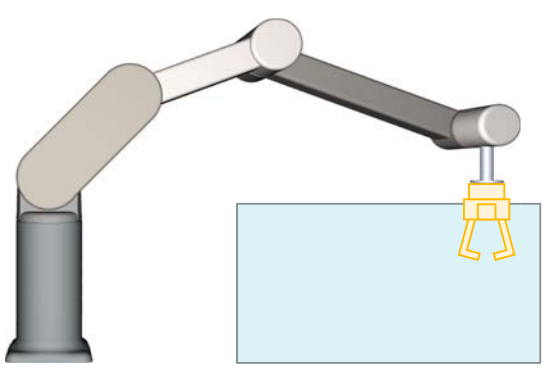

The kinematics consists of the following axes:

- A rotary axis A1 with the distances L1 in z direction of the KCS and L2 in x direction of the KCS to the kinematics zero point
- A rotary axis A2 at distance L3 to axis A1
- A rotary axis A4 (orientation axis) at distance L4 in x direction of the KCS from the axis A2

The kinematics consists of a base and articulated arms, which are connected by revolute joints (axes A1, A2). The axes A1 and A2 move the articulated arm vertically. Through a forced coupling between the axis A2 and the flange system, the z axis of the FCS always points in the negative z direction of the KCS. The orientation axis A4 enables rotation of the tool.

Coordinate systems and zero position

The graphic below shows the following in the side view:

- The position of the axes and of the forced coupler point
- The position of coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the kinematics is indicated (dashed)

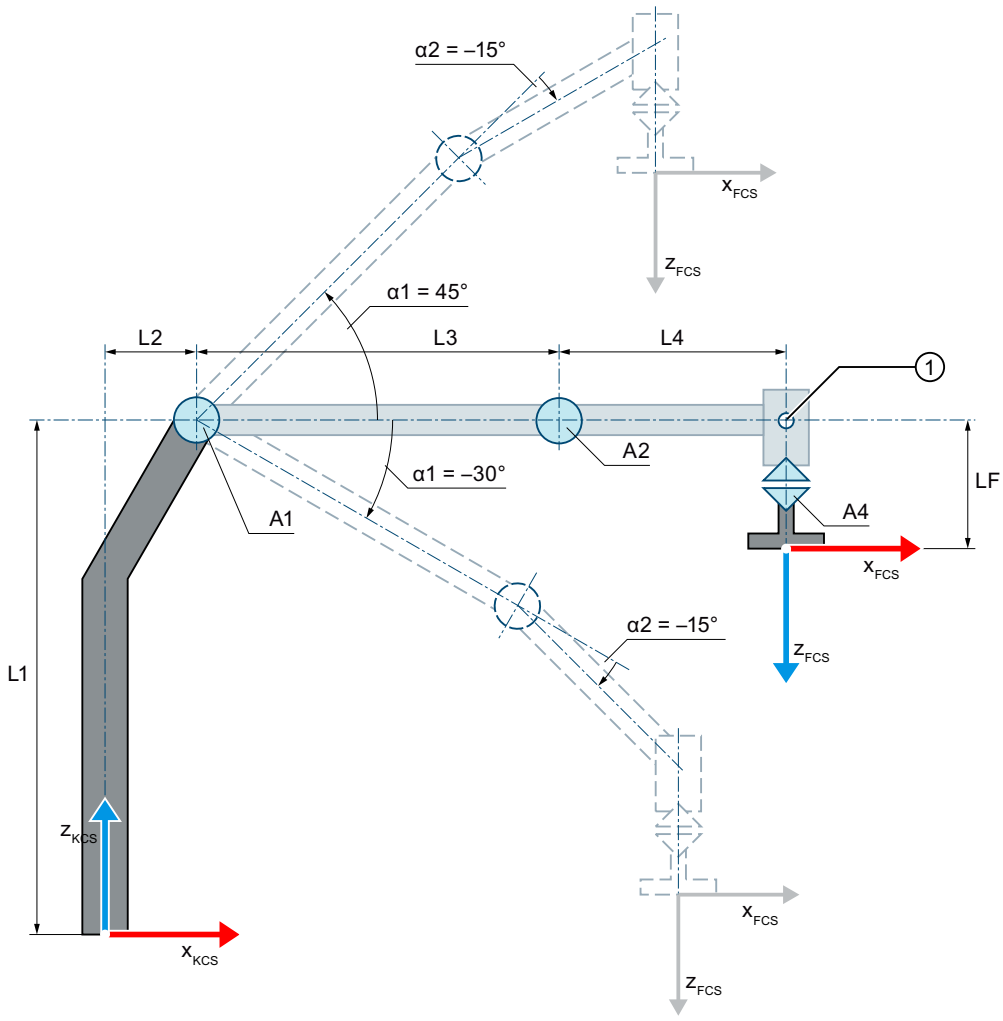

① Forced coupler point

Zero position of the kinematics

- L1 Distance of the axis A1 to the kinematics zero point (KZP) in z direction of the KCS
- L2 Distance of the axis A1 to the KZP in x direction of the KCS
- L3 Distance of the axis A2 to the axis A1 in x direction of the KCS
- L4 Distance of the forced coupler point to the axis A2 in x direction of the KCS
- LF Distance of the FCS to the forced coupler point in z direction of the FCS

Deflection of the kinematics

- α1 Positive deflection of the axis A1 when α1 = 45.0°
  Negative deflection of the axis A1 when α1 = -30.0°
- α2 Negative deflection of the axis A2 when α2 = -15.0°

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the base of the kinematics. You define the position of the axis A1 relative to the KZP using lengths L1 and L2. The axis A2 is located at distance L3 in x direction of the KCS from the axis A1.

The flange coordinate system (FCS) is located at the following distances from the axis A2 and the forced coupler point:

- Distance L4 to the axis A2 in x direction of the KCS
- Distance LF to the forced coupler point in the negative z direction of the KCS

The axis A2 and the flange system are force-coupled. With the force coupling, the z axis of the FCS always points in negative z direction of the KCS. The forced coupler point is located at distance L4 in x direction of the KCS from the axis A2.

The following table shows the zero position of the axes:

Axis	Zero position
A1	Length L3 points in x direction of the KCS.
A2	At zero position of the axis A1, the length L4 points in x direction of the KCS.
A4	At the zero position of axis A1 and A2, the x axis of the FCS points in the direction of the x axis of the KCS.

Compensation of mechanical axis couplings

For the kinematics, you can configure a mechanical axis coupling of axis A1 to axis A2. The kinematics transformation compensates for the configured mechanical axis coupling. With a coupling factor > 0.0, the kinematics transformation assumes that a positive motion of the axis A1 leads to a negative motion on the axis A2.

Transformation area

The kinematics transformation covers the following traversing range (Page 2239) of the axes:

- Axis A1: -180.0° ≤ α1 < 180.0°
- Axis A2: -180.0° ≤ α2 < 180.0°
- Axis A4: No limiting
  An angle greater than 360° can be defined for the orientation. But coordinate A of the tool center point (TCP) is mapped in the range -180° to +180°.

Articulated arm 3D

The kinematics "Articulated arm 3D" supports three axes and three degrees of freedom. The axes are configured as serial kinematics with forced coupling of the flange system.

The following graphic shows the principal configuration and the typical working area of the kinematics:

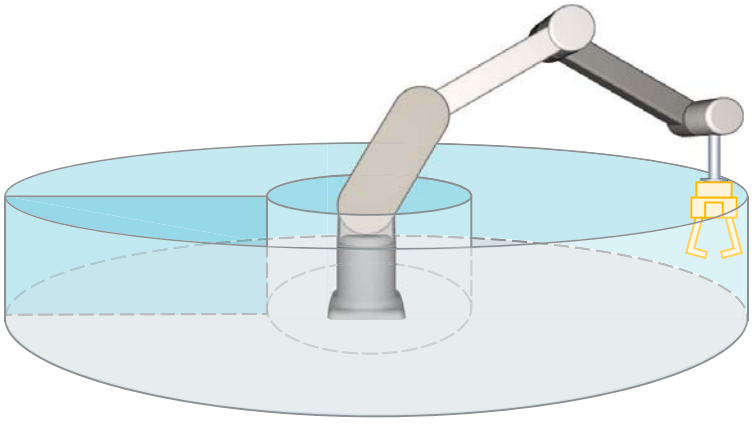

The kinematics consists of the following axes:

- A rotary axis A1 with rotation around the z axis of the kinematics coordinate system (KCS)
- A rotary axis A2 with the distances L1 in z direction of the KCS and L2 in x direction of the KCS to the kinematics zero point
- A rotary axis A3 at distance L3 to axis A2

The kinematics consists of a base and articulated arms, which are connected by revolute joints (axes A1, A2 and A3). Axis A1 rotates the kinematics horizontally around the base. Axes A2 and A3 move the articulated arm vertically. The kinematics enables an approximately spherical working area. Through a forced coupling between the axis A2 and the flange system, the z axis of the FCS always points in the negative z direction of the KCS.

Coordinate systems and zero position

The graphic below shows the following in the side view (xz plane):

- The position of the axes and of the forced coupler point
- The position of coordinate systems KCS and FCS
- The zero position of the axes
- The positive/negative deflection of the axes A2 and A3 is indicated (dashed)

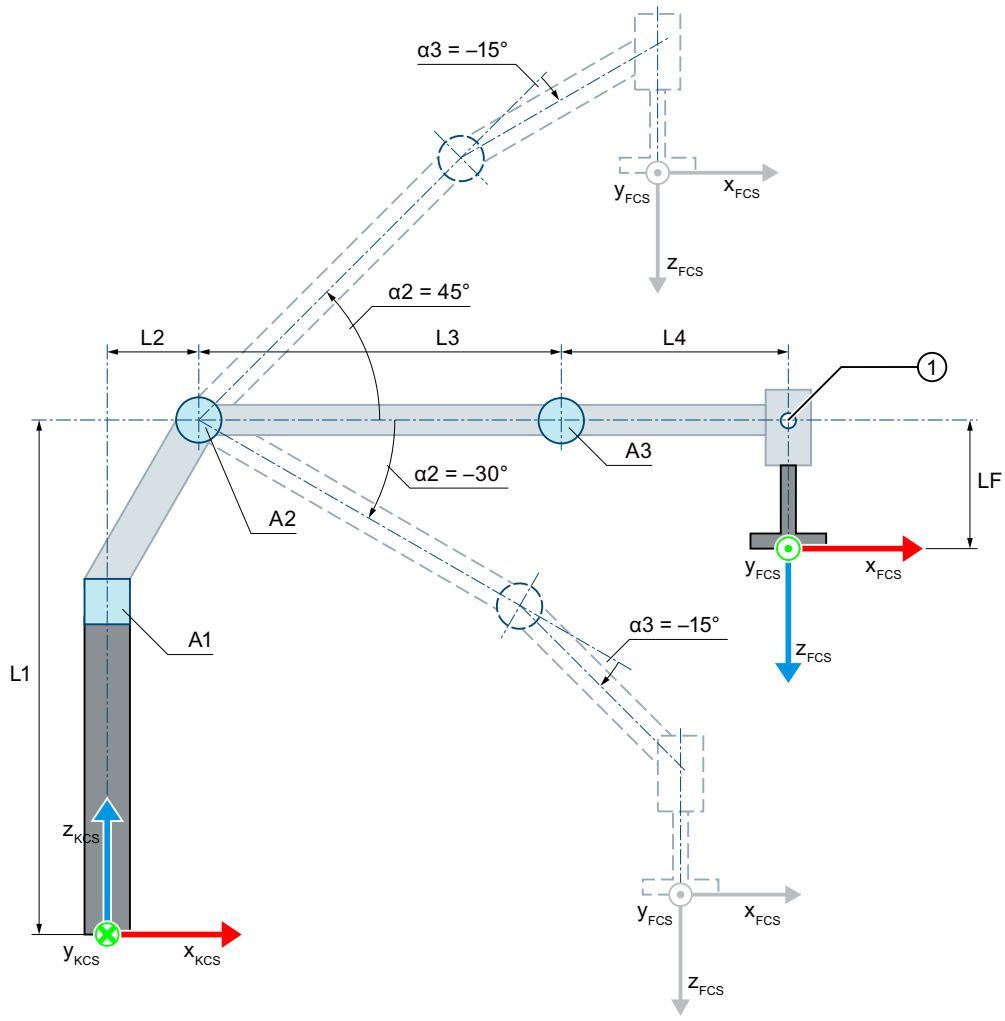

① Forced coupler point

Zero position of the kinematics

- L1 Distance of the axis A2 to the kinematics zero point (KZP) in z direction of the KCS
- L2 Distance of the axis A2 to the KZP in x direction of the KCS
- L3 Distance of the axis A3 to the axis A2 in x direction of the KCS
- L4 Distance of the forced coupler point to the axis A3 in x direction of the KCS
- LF Distance of the FCS to the forced coupler point in z direction of the FCS

Deflection of the kinematics

- α2 Positive deflection of the axis A2 when α2 = 45.0°
  Negative deflection of the axis A2 when α2 = -30.0°
- α3 Negative deflection of the axis A3 when α3 = -15.0°

Legend for representation of the kinematics (Page 2160)

The graphic below shows the following in the top view (xy plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the kinematics is indicated (dashed)

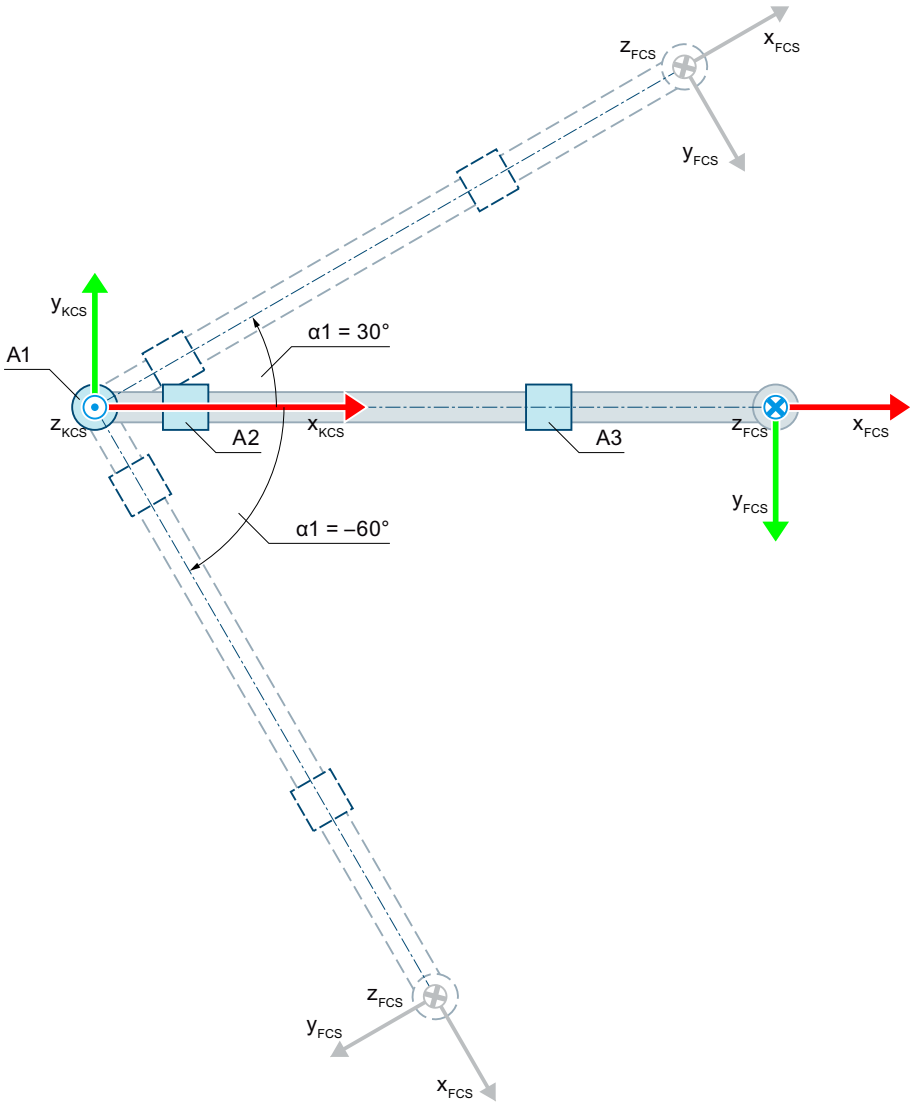

Zero position of the kinematics

Deflection of the kinematics

α1 Positive deflection of the axis A1 when α1 = 30.0°

Negative deflection of the axis A1 when α1 = -60.0°

Legend for representation of the kinematics (Page 2160)

The KCS with the kinematics zero point (KZP) is located at the base of the kinematics. You define the position of the axis A2 relative to the KZP using lengths L1 and L2. The axis A3 is located at distance L3 in x direction of the KCS from the axis A2.

The flange coordinate system (FCS) is located at the following distances from the axis A3 and the forced coupler point:

- Distance L4 to the axis A3 in x direction of the KCS
- Distance LF to the forced coupler point in the negative z direction of the KCS

The axis A3 and the flange system are force-coupled. With the force coupling, the z axis of the FCS always points in negative z direction of the KCS. The forced coupler point is located at distance L4 in x direction of the KCS from the axis A3.

The following table shows the zero position of the axes:

Axis	Zero position
A1	The articulated arms of the kinematics point in the x direction of the KCS.
A2	At zero position of the axis A1, the length L3 points in x direction of the KCS.
A3	At zero position of the axes A1 and A2, the length L4 points in x direction of the KCS.

Compensation of mechanical axis couplings

You can configure a mechanical axis coupling of axis A2 to axis A3 for the kinematics. The kinematics transformation compensates for the configured mechanical axis coupling. With a coupling factor > 0.0, the kinematics transformation assumes that a positive motion of the axis A2 leads to a negative motion on the axis A3.

Transformation area

The kinematics transformation covers the following traversing range (Page 2239) of the axes:

- Axis A1: -180.0° ≤ α1 < 180.0°
- Axis A2: -180.0° ≤ α2 < 180.0°
- Axis A3: -180.0° ≤ α3 < 180.0°

Note

Singular positions

The kinematics have singular positions (Page 2241).

A singular position occurs when the zero point of the flange coordinate system (FCS) is located on the z-axis of the kinematics coordinate system (KCS). Inverse transformation is not possible in this area.

The graphic below shows examples of permissible and impermissible joint positions for the transformation:

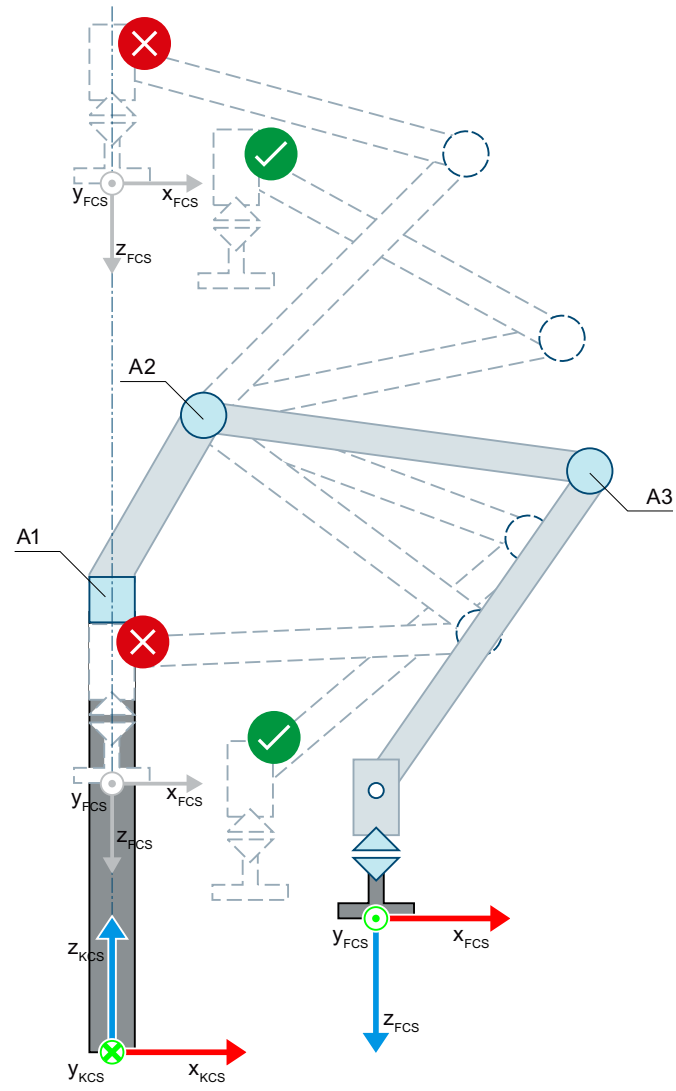

Permissible joint position

Invalid joint position for the transformation

Articulated arm 3D with orientation

The kinematics "Articulated arm 3D with orientation" supports four axes and four degrees of freedom. The axes are configured as serial kinematics with forced coupling of the flange system.

The following graphic shows the principal configuration and the typical working area of the kinematics:

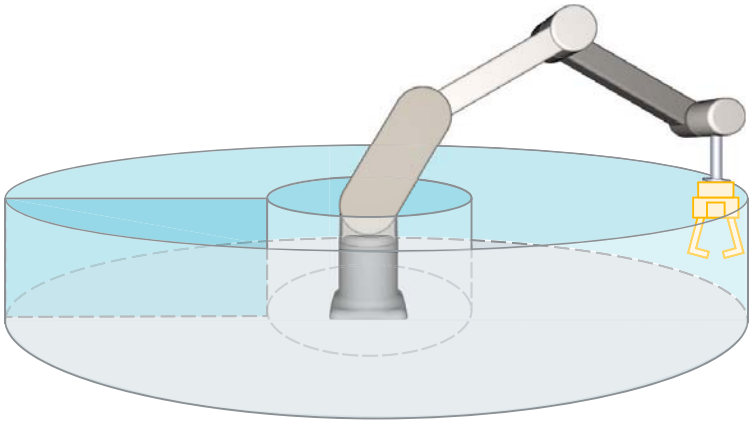

The kinematics consists of the following axes:

- A rotary axis A1 with rotation around the z axis of the kinematics coordinate system (KCS)
- A rotary axis A2 with the distances L1 in z direction of the KCS and L2 in x direction of the KCS to the kinematics zero point
- A rotary axis A3 at distance L3 to axis A2
- A rotary axis A4 (orientation axis) at distance L4 in x direction of the KCS from the axis A3

The kinematics consists of a base and articulated arms, which are connected by revolute joints (axes A1, A2 and A3). Axis A1 rotates the kinematics horizontally around the base. Axes A2 and A3 move the articulated arm vertically. The kinematics enables an approximately spherical working area. Through a forced coupling between the axis A2 and the flange system, the z axis of the FCS always points in the negative z direction of the KCS. The orientation axis A4 enables rotation of the tool.

Coordinate systems and zero position

The graphic below shows the following in the side view (xz plane):

- The position of the axes and of the forced coupler point
- The position of coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the kinematics is indicated (dashed)

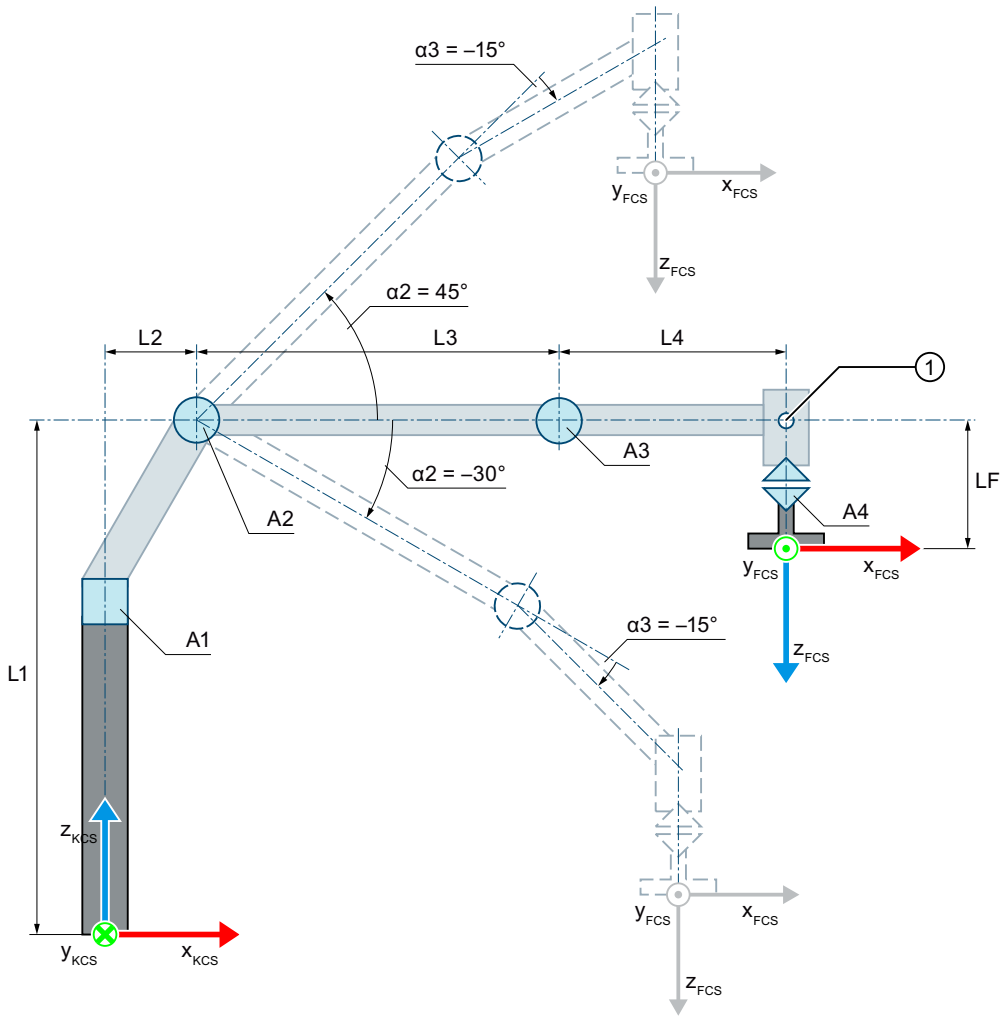

① Forced coupler point

Zero position of the kinematics

- L1 Distance of the axis A2 to the kinematics zero point (KZP) in z direction of the KCS
- L2 Distance of the axis A2 to the KZP in x direction of the KCS
- L3 Distance of the axis A3 to the axis A2 in x direction of the KCS
- L4 Distance of the forced coupler point to the axis A3 in x direction of the KCS
- LF Distance of the FCS to the forced coupler point in z direction of the FCS

Deflection of the kinematics

- α2 Positive deflection of the axis A2 when α2 = 45.0°
  Negative deflection of the axis A2 when α2 = -30.0°
- α3 Negative deflection of the axis A3 when α3 = -15.0°

Legend for representation of the kinematics (Page 2160)

The graphic below shows the following in the top view (xy plane):

- The position of the axes
- The position of coordinate systems KCS and FCS

- The zero position of the kinematics
- The positive/negative deflection of the kinematics is indicated (dashed)

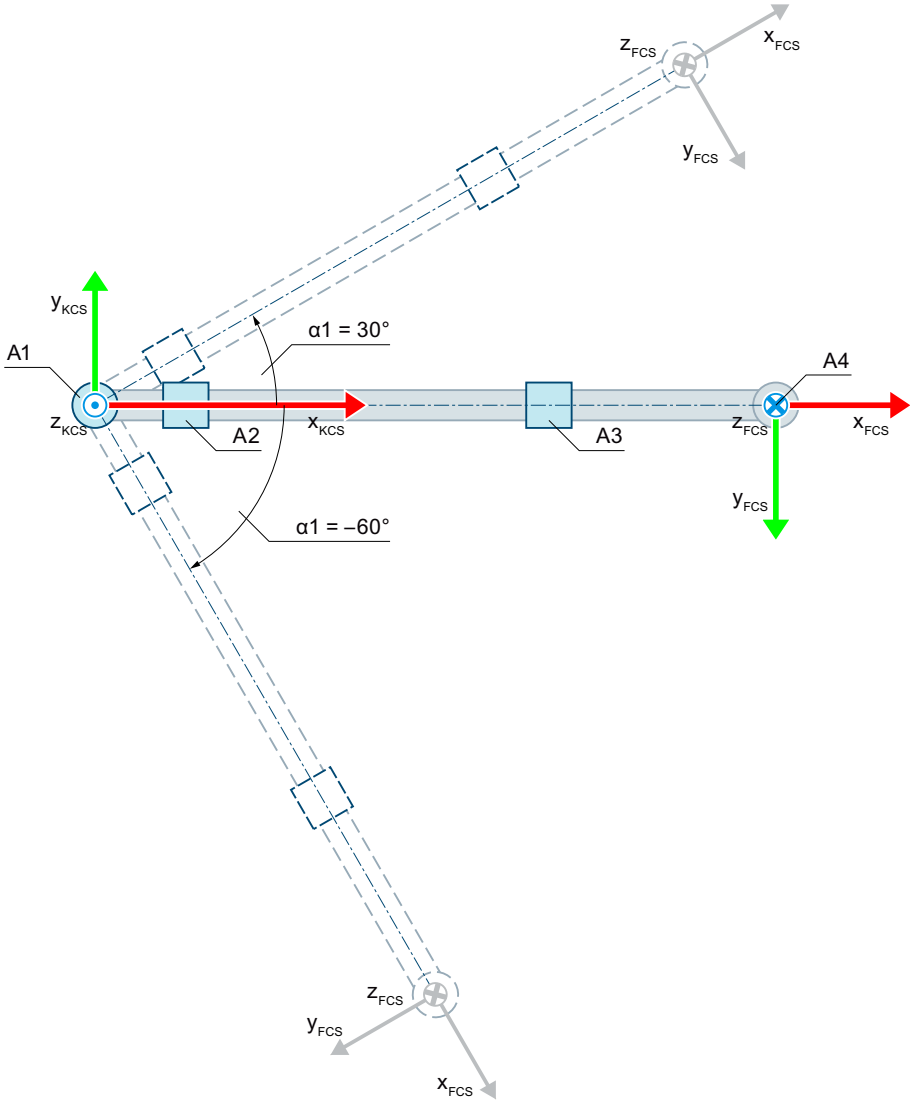

Zero position of the kinematics

Deflection of the kinematics

α1 Positive deflection of the axis A1 when α1 = 30.0°

Negative deflection of the axis A1 when α1 = -60.0°

Legend for representation of the kinematics (Page 2160)

The KCS with the kinematics zero point (KZP) is located at the base of the kinematics. You define the position of the axis A2 relative to the KZP using lengths L1 and L2. The axis A3 is located at distance L3 in x direction of the KCS from the axis A2.

The flange coordinate system (FCS) is located at the following distances from the axis A3 and the forced coupler point:

- Distance L4 to the axis A3 in x direction of the KCS
- Distance LF to the forced coupler point in the negative z direction of the KCS

The axis A3 and the flange system are force-coupled. With the force coupling, the z axis of the FCS always points in negative z direction of the KCS. The forced coupler point is located at distance L4 in x direction of the KCS from the axis A3.

The following table shows the zero position of the axes:

Axis	Zero position
A1	The articulated arms of the kinematics point in the x direction of the KCS.
A2	At zero position of the axis A1, the length L3 points in x direction of the KCS.
A3	At zero position of the axes A1 and A2, the length L4 points in x direction of the KCS.
A4	At zero position of the axes A1, A2 and A3, the x axis of the FCS points in the direction of the x axis of the KCS.

Compensation of mechanical axis couplings

You can configure a mechanical axis coupling of axis A2 to axis A3 for the kinematics. The kinematics transformation compensates for the configured mechanical axis coupling. With a coupling factor > 0.0, the kinematics transformation assumes that a positive motion of the axis A2 leads to a negative motion on the axis A3.

Transformation area

The kinematics transformation covers the following traversing range (Page 2239) of the axes:

- Axis A1: -180.0° ≤ α1 < 180.0°
- Axis A2: -180.0° ≤ α2 < 180.0°
- Axis A3: -180.0° ≤ α3 < 180.0°
- Axis A4: No limiting
  An angle greater than 360° can be defined for the orientation. But coordinate A of the tool center point (TCP) is mapped in the range -180° to +180°.

Note

Singular positions

The kinematics have singular positions (Page 2241).

A singular position occurs when the zero point of the flange coordinate system (FCS) is located on the z-axis of the kinematics coordinate system (KCS). Inverse transformation is not possible in this area.

The graphic below shows examples of permissible and impermissible joint positions for the transformation:

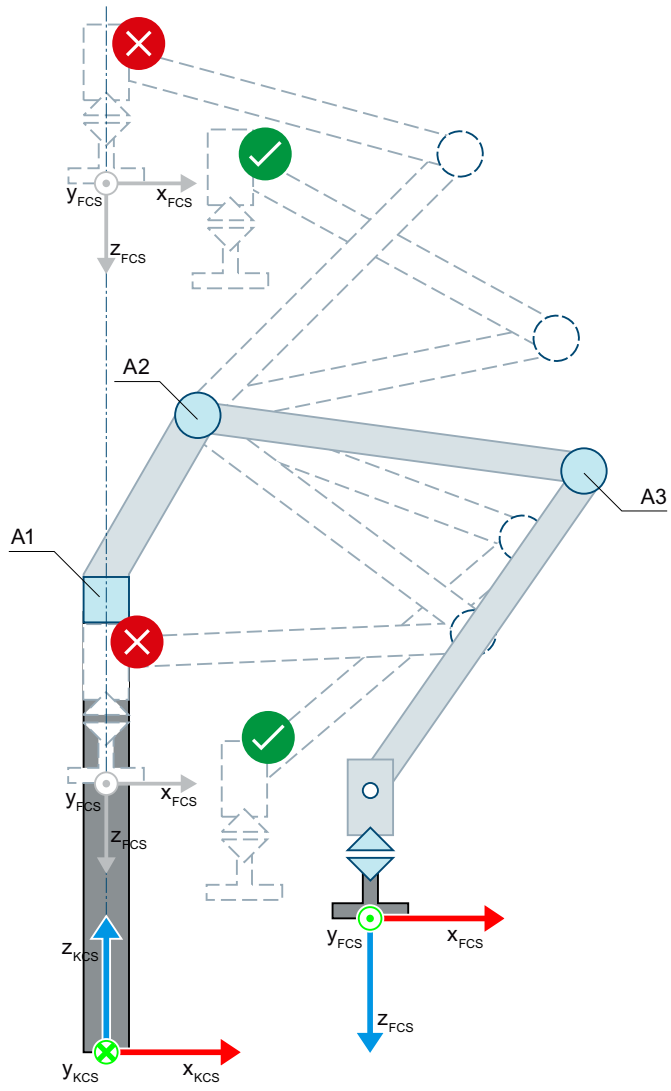

Permissible joint position

Invalid joint position for the transformation

Articulated arm tags

Articulated arm 2D

You define the 2D articulated arm kinematics systems using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	11	Articulated arm 2D
	12	Articulated arm 2D with orientation
<TO>.Kinematics.Parameter[1]	-1.0E12 to 1.0E12	Distance L1 of the axis A1 to the kinematics zero point in z direction of the kinematics coordinate system (KCS)

Tags	Values	Description
<TO>.Kinematics.Parameter[2]	-1.0E12 to 1.0E12	Distance L2 of the axis A1 to the kinematics zero point in x direction of the KCS
<TO>.Kinematics.Parameter[3]	0.001 to 1.0E12	Arm length L3 between the axes A1 and A2
<TO>.Kinematics.Parameter[4]	-	Mechanical axis coupling of axis A1 to axis A2 present/not present
	0	Not present
	1	Present
<TO>.Kinematics.Parameter[5]	-1.0E12 to 1.0E12	Mechanical axis coupling factor of axis A1 to axis A2
<TO>.Kinematics.Parameter[6]	0.001 to 1.0E12	Side length L4 between A2 and positive coupling point
<TO>.Kinematics.Parameter[7]	-1.0E12 to 1.0E12	Distance LF of the flange coordinate system (FCS) from the forced coupler point in the negative z direction of the KCS

Articulated arm 3D

You define the 3D articulated arm kinematics systems using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	13	Articulated arm 3D
	14	Articulated arm 3D with orientation
<TO>.Kinematics.Parameter[1]	-1.0E12 to 1.0E12	Distance L1 of the axis A2 to the kinematics zero point in z direction of the KCS
<TO>.Kinematics.Parameter[2]	0.0 to 1.0E12	Distance L2 of the axis A2 to the kinematics zero point in x direction of the KCS
<TO>.Kinematics.Parameter[3]	0.001 to 1.0E12	Arm length L3 between the axes A2 and A3
<TO>.Kinematics.Parameter[4]	-	Mechanical axis coupling of axis A2 to A3 present/not present
	0	Not present
	1	Present
<TO>.Kinematics.Parameter[5]	-1.0E12 to 1.0E12	Mechanical axis coupling factor from axis A2 to axis A3
<TO>.Kinematics.Parameter[6]	0.001 to 1.0E12	Arm length L4 between the axis A3 and positive coupler point
<TO>.Kinematics.Parameter[7]	-1.0E12 to 1.0E12	Distance LF of the FCS from the forced coupler point in the negative z direction of the KCS

Delta picker

Delta picker 2D

The kinematics "Delta picker 2D" supports two axes and two degrees of freedom The axes are configured as parallel kinematics.

The following graphic shows the principal configuration and the typical working area of the kinematics:

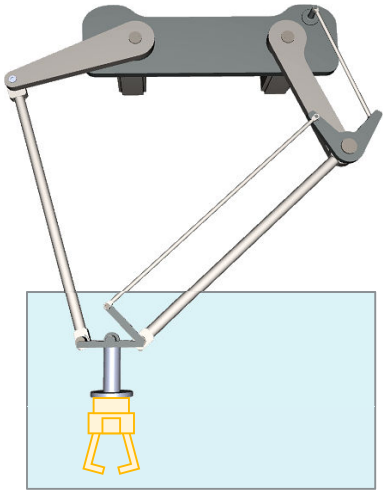

The kinematics consists of two rotary axes A1 and A2.

The kinematics is modeled suspended and consists of an upper connecting plate, two upper arms and a lower connecting plate. The axes for moving the arms (axes A1, A2) are fastened to the upper connecting plate. The upper arms and the connecting rods connect the upper and lower connecting plates. The tool is suspended from the lower connecting plate. The parallelogram structures of the connecting rods keep the lower connecting plate parallel to the xy plane of the KCS.

Coordinate systems and zero position

The graphic below shows the following in the front view:

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The deflection of the kinematics is indicated (dashed)

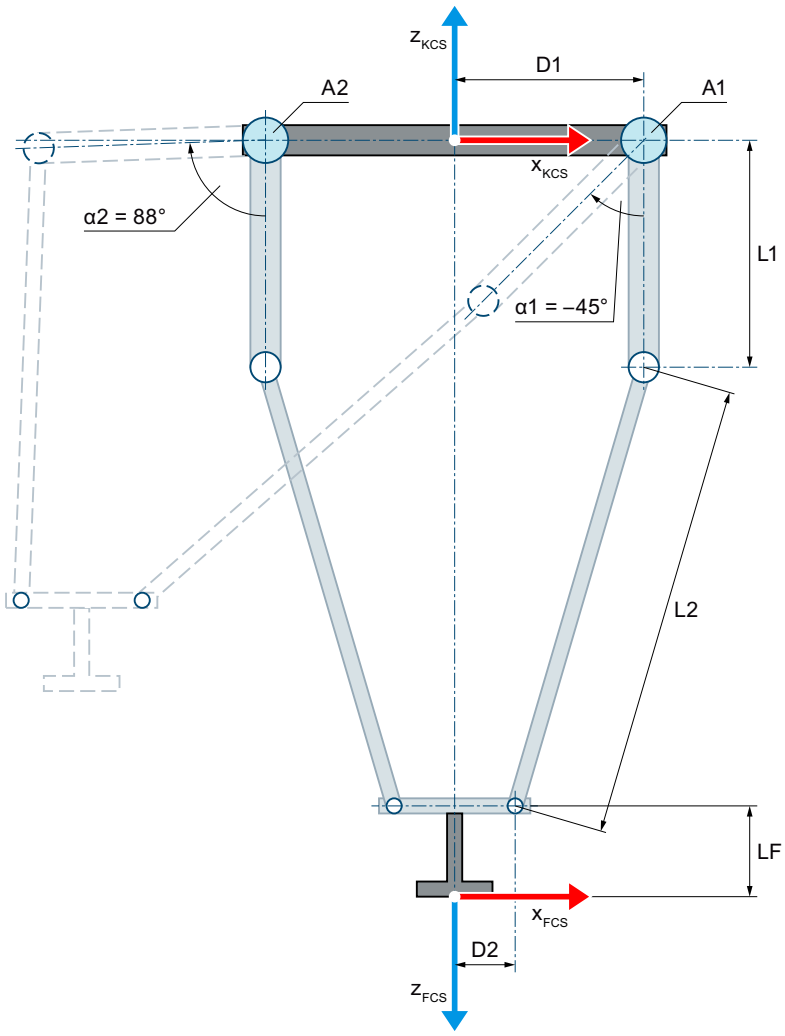

Zero position of the kinematics

D1 Distance of the axes from the center of the upper connecting plate (radius of the upper connecting plate)

D2 Distance of the hinge points of the connecting rods from the lower connecting plate (radius of the lower connecting plate)

L1 Length of the upper arms

L2 Length of the connecting rods

D1, D2, L1 and L2 are identical for the two arms of the kinematics.

LF Flange length before the FCS in the z direction of the KCS

Deflection of the kinematics

The motion of the axes in the positive direction is the outward rotation of the upper arms.

α1 Deflection of the axis A1 in the negative direction when α1 = -45.0°

α2 Deflection of the axis A2 in the positive direction when α2 = 88.0°

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the center point of the upper connecting plate. The axes A1 and A2 are at distance D1 from the common center point (kinematics zero point).

The flange coordinate system (FCS) is located on the bottom of the lower connecting plate with equal distance D2 to the hinge points of each arm. You shift the FCS in the negative z direction of the KCS using length LF.

In the zero position of the axes A1 and A2, the upper arms point in the negative z direction of the KCS.

Transformation area

Only the outwardly bent joint position (Page 2220) is permitted for the arms of the kinematics. You cannot traverse the axes beyond the extended position of the arms.

Delta picker 2D with orientation

The kinematics "Delta picker 2D with orientation" supports three axes and three degrees of freedom. The axes are configured as parallel kinematics.

The following graphic shows the principal configuration and the typical working area of the kinematics:

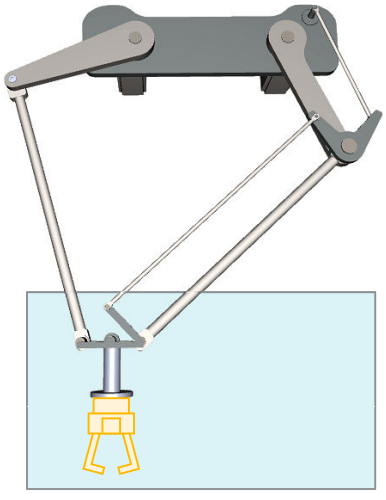

The kinematics consists of the following axes:

- Two rotary axes A1, A2
- One rotary axis A4 (orientation axis)

The kinematics is modeled suspended and consists of an upper connecting plate, two upper arms and a lower connecting plate. The axes for moving the arms (axes A1, A2) are fastened to the upper connecting plate. The upper arms and the connecting rods connect the upper and lower connecting plates. The tool is suspended from the lower connecting plate. The parallelogram structures of the connecting rods keep the lower connecting plate parallel to the xy plane of the KCS. The orientation axis A4 enables rotation of the tool.

Coordinate systems and zero position

The graphic below shows the following in the front view:

- The position of the axes and the coordinate systems KCS and FCS
- The zero positions of the axes A1 and A4
- The deflection of the kinematics is indicated (dashed)

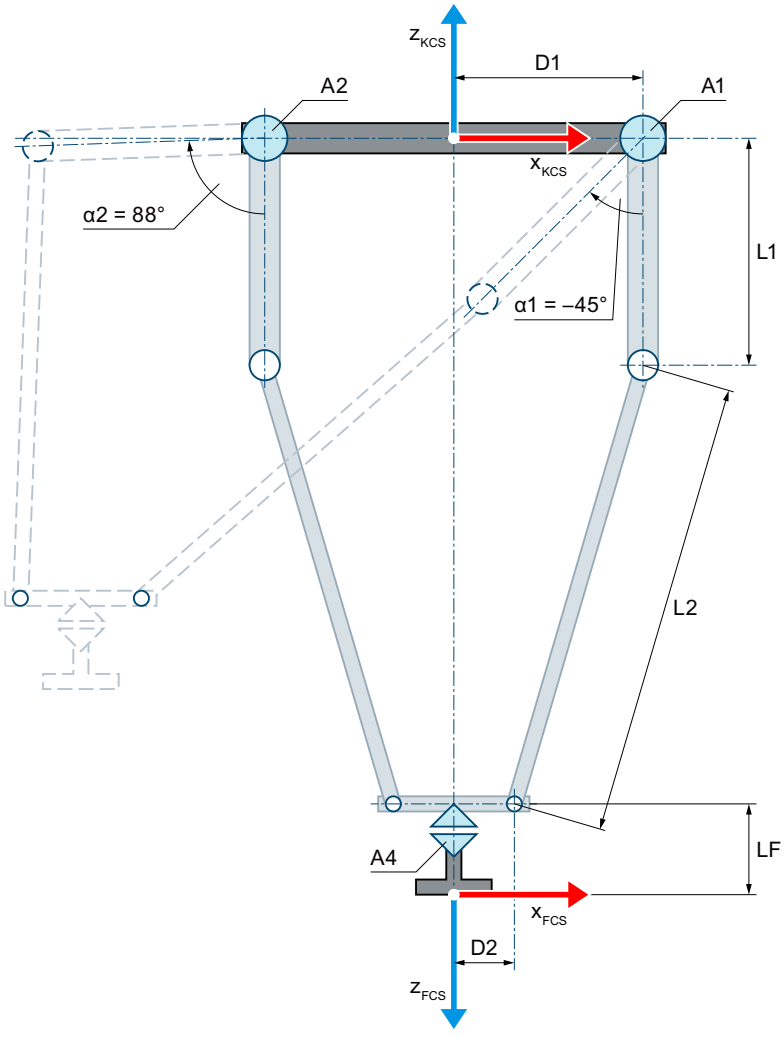

Zero position of the kinematics

D1 Distance of the axes from the center of the upper connecting plate (radius of the upper connecting plate)

D2 Distance of the hinge points of the connecting rods from the lower connecting plate (radius of the lower connecting plate)

L1 Length of the upper arms

L2 Length of the connecting rods

D1, D2, L1 and L2 are identical for the two arms of the kinematics.

LF Flange length before the FCS in the z direction of the KCS

Deflection of the kinematics

The motion of the axes in the positive direction is the outward rotation of the upper arms.

α1 Deflection of the axis A1 in the negative direction when α1 = -45.0°

α2 Deflection of the axis A2 in the positive direction when α2 = 88.0°

Legend for representation of the kinematics (Page 2160)

The kinematics coordinate system (KCS) with the kinematics zero point (KZP) is located at the center point of the upper connecting plate. The axes A1 and A2 are at distance D1 from the common center point (kinematics zero point).

The flange coordinate system (FCS) is located on the bottom of the lower connecting plate with equal distance D2 to the hinge points of each arm. You shift the FCS in the negative z direction of the KCS using length LF.

In the zero position of the axes A1 and A2, the upper arms point in the negative z direction of the KCS. At the zero position of axis A4, the x axis of the FCS points in the direction of the x axis of the KCS.

Transformation area

Only the outwardly bent joint position (Page 2220) is permitted for the arms of the kinematics. You cannot traverse the axes beyond the extended position of the arms.

Delta picker 3D

The kinematics "Delta picker 3D" supports three axes and three degrees of freedom. The axes are configured as parallel kinematics.

The following graphic shows the principal configuration and the typical working area of the kinematics:

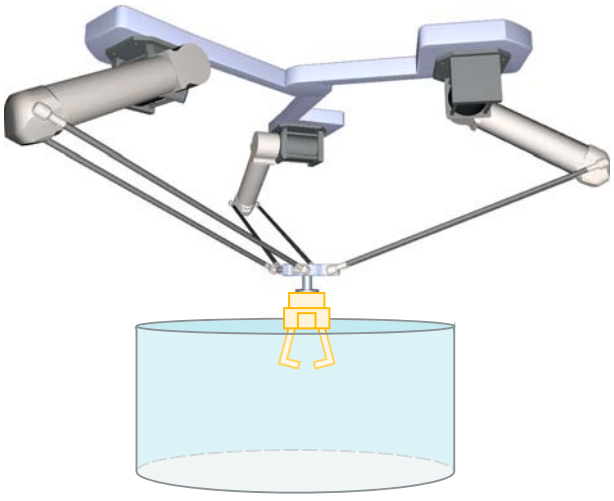

The kinematics consists of three rotary axes A1, A2 and A3.

The kinematics is modeled suspended and consists of an upper connecting plate, three upper arms and a lower connecting plate. The axes for moving the arms (axes A1, A2 and A3) are fastened to the upper connecting plate. The upper arms and the connecting rods connect the upper and lower connecting plates. The tool is suspended from the lower connecting plate. The parallelogram structures of the connecting rods keep the lower connecting plate parallel to the xy plane of the KCS.

Coordinate systems and zero position

The graphic below shows the following in the top view (xy plane):

- The position of the kinematics coordinate system (KCS)
- The angles of the axes A1, A2 and A3 to one another

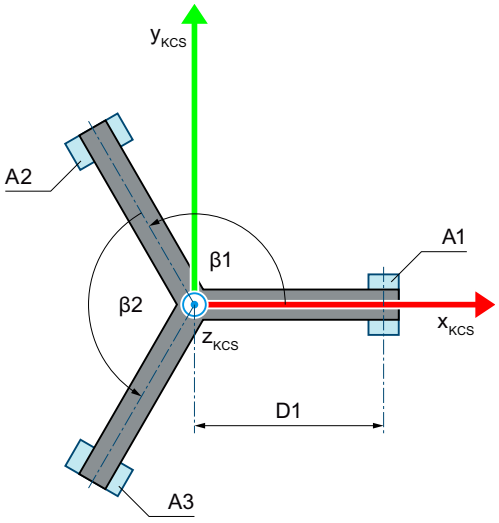

β1 Angle between the axes A1 and A2

β2 Angle between the axes A2 and A3

Legend for representation of the kinematics (Page 2160)

The following graphic shows the top view of the position of the flange coordinate system (FCS) in the xy plane of the lower connecting plate:

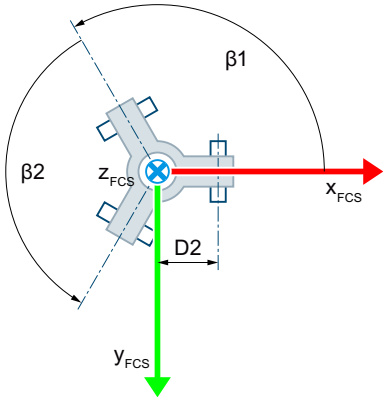

The graphic below shows the following in the front view (xz plane):

- The position of the axis A1 and the coordinate systems KCS and FCS
- The zero position of the axis A1
- The positive/negative deflection of the axis A1 is indicated (dashed)

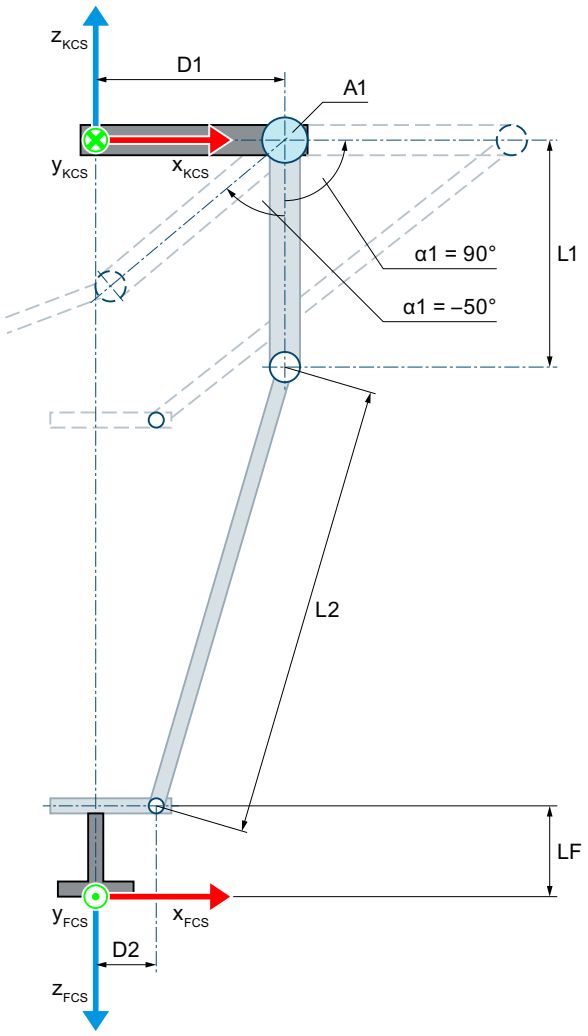

Zero position of the kinematics

D1 Distance of the axes from the center of the upper connecting plate (radius of the upper connecting plate)

D2 Distance of the hinge points of the connecting rods from the lower connecting plate (radius of the lower connecting plate)

L1 Length of the upper arms

L2 Length of the connecting rods

D1, D2, L1 and L2 are identical for the three arms of the kinematics.

LF Flange length before the FCS in the z direction of the KCS

Deflection of the kinematics

The motion of the axes in the positive direction is the outward rotation of the upper arms.

α1 Deflection of the axis A1 in negative direction when α1 = -50.0°

Deflection of the axis A1 in positive direction when α1 = 90.0°

Legend for representation of the kinematics (Page 2160)

The KCS with the kinematics zero point (KZP) is located at the center point of the upper connecting plate. The axes A1, A2 and A3 are at distance D1 from the common center point (kinematics zero point).

The FCS is located in the center on the bottom of the lower connecting plate with equal distance D2 to the hinge points of each arm. You shift the FCS in the negative z direction of the KCS using length LF.

In the zero position of the axes A1, A2 and A3, the upper arms point in the negative z direction of the KCS.

Transformation area

Only the outwardly bent joint position (Page 2220) is permitted for the arms of the kinematics. You cannot traverse the axes beyond the extended position of the arms.

Delta picker 3D with orientation

The kinematics "Delta picker 3D with orientation" supports four axes and four degrees of freedom. The axes are configured as parallel kinematics.

The following graphic shows the principal configuration and the typical working area of the kinematics:

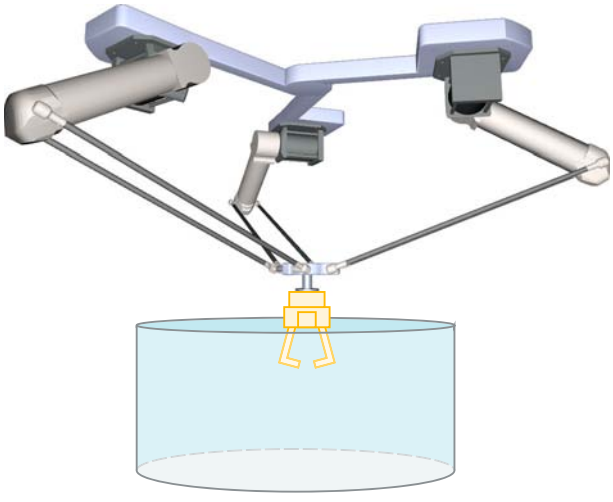

The kinematics consists of the following axes:

- Three rotary axes A1, A2 and A3
- One rotary axis A4 (orientation axis)

The kinematics is modeled suspended and consists of an upper connecting plate, three upper arms and a lower connecting plate. The axes for moving the arms (axes A1, A2 and A3) are fastened to the upper connecting plate. The upper arms and the connecting rods connect the upper and lower connecting plates. The tool is attached to the lower connecting plate. The parallelogram structures of the connecting rods keep the lower connecting plate parallel to the xy plane of the KCS. The orientation axis A4 enables rotation of the tool.

Coordinate systems and zero position

The graphic below shows the following in the top view (xy plane):

- The position of the kinematics coordinate system (KCS)
- The angles of the axes A1, A2 and A3 to one another

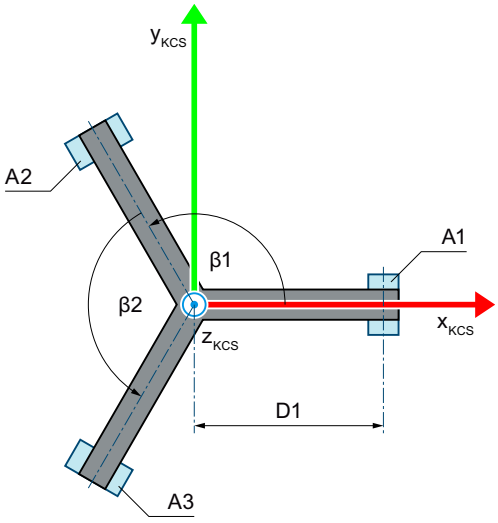

β1 Angle between the axes A1 and A2

β2 Angle between the axes A2 and A3

Legend for representation of the kinematics (Page 2160)

The following graphic shows the top view of the position of the flange coordinate system (FCS) in the xy plane of the lower connecting plate:

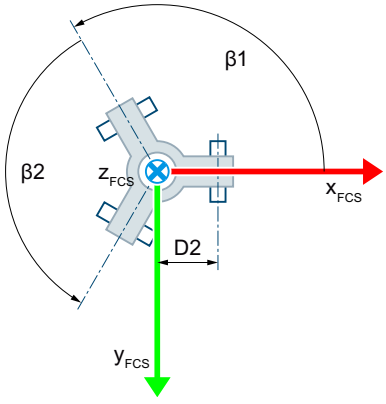

The graphic below shows the following in the front view (xz plane):

- The position of the axis A1 and the coordinate systems KCS and FCS
- The zero positions of the axes A1 and A4
- The positive/negative deflection of the axis A1 is indicated (dashed)

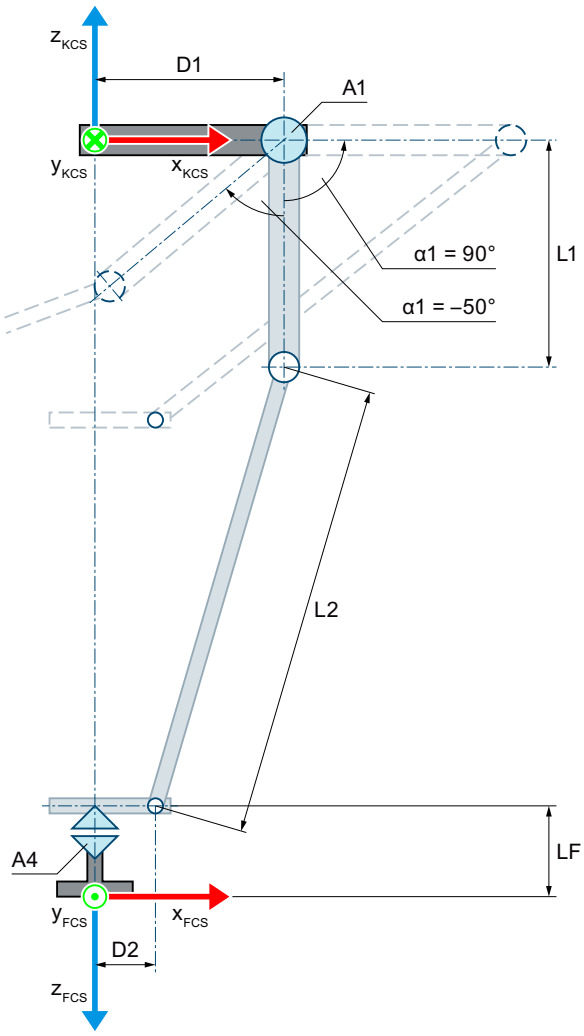

Zero position of the kinematics

D1 Distance of the axes from the center of the upper connecting plate (radius of the upper connecting plate)

D2 Distance of the hinge points of the connecting rods from the lower connecting plate (radius of the lower connecting plate)

L1 Length of the upper arms

L2 Length of the connecting rods

D1, D2, L1 and L2 are identical for the three arms of the kinematics.

LF Flange length before the FCS in the z direction of the KCS

Deflection of the kinematics

The motion of the axes in the positive direction is the outward rotation of the upper arms.

α1 Deflection of the axis A1 in negative direction when α1 = -50.0°
Deflection of the axis A1 in positive direction when α1 = 90.0°

Legend for representation of the kinematics (Page 2160)

The KCS with the kinematics zero point (KZP) is located at the center point of the upper connecting plate. The axes A1, A2 and A3 are at distance D1 from the common center point (kinematics zero point).

The FCS is located in the center on the bottom of the lower connecting plate with equal distance D2 to the hinge points of each arm. You shift the FCS in the negative z direction of the KCS using length LF.

In the zero position of the axes A1, A2 and A3, the upper arms point in the negative z direction of the KCS. At the zero position of axis A4, the x axis of the FCS points in the direction of the x axis of the KCS.

Transformation area

Only the outwardly bent joint position (Page 2220) is permitted for the arms of the kinematics. You cannot traverse the axes beyond the extended position of the arms.

Permissible joint position for delta picker

Only the outwardly bent joint position is permitted for the arms of the delta picker kinematics. The graphic below shows examples of permissible and impermissible joint positions for the transformation:

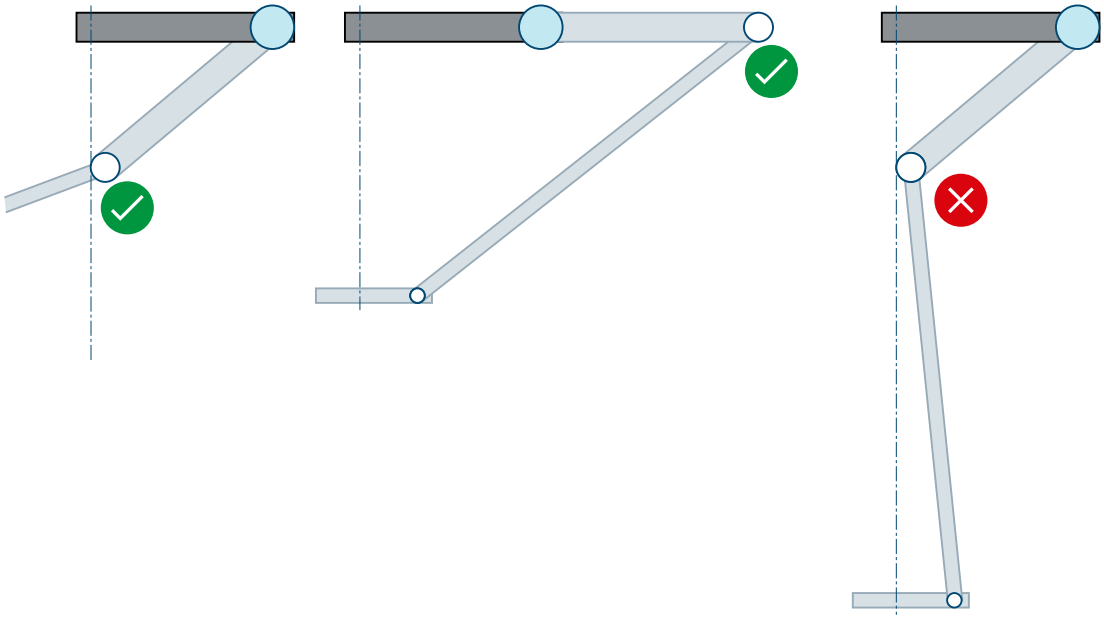

Permissible joint position

Invalid joint position for the transformation

Delta picker tags

Delta picker 2D

You define the 2D delta picker kinematics using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	15	Delta picker 2D
	16	Delta picker 2D with orientation
<TO>.Kinematics.Parameter[1]	0.0 to 1.0E12	Distance D1 (radius of the upper connecting plate)
<TO>.Kinematics.Parameter[2]	0.001 to 1.0E12	Length L1 of the upper arms
<TO>.Kinematics.Parameter[3]	0.001 to 1.0E12	Length L2 of connecting rods
<TO>.Kinematics.Parameter[4]	0.0 to 1.0E12	Distance D2 (radius of the lower connecting plate)
<TO>.Kinematics.Parameter[5]	-1.0E12 to 1.0E12	Distance LF of the FCS from the lower connecting plate in the negative z direction of the KCS

Delta picker 3D

You define the 3D delta picker kinematics using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	17	Delta picker 3D
	18	Delta picker 3D with orientation
<TO>.Kinematics.Parameter[1]	0.0 to 1.0E12	Distance D1 (radius of the upper connecting plate)
<TO>.Kinematics.Parameter[2]	0.001 to 1.0E12	Length L1 of the upper arms
<TO>.Kinematics.Parameter[3]	0.001 to 1.0E12	Length L2 of connecting rods
<TO>.Kinematics.Parameter[4]	0.0 to 1.0E12	Distance D2 (radius of the lower connecting plate)
<TO>.Kinematics.Parameter[5]	90.001° to 179.998°	Angle β1 between the axes A1 and A2
<TO>.Kinematics.Parameter[6]	90.001° to 179.998°	Angle β2 between the axes A2 and A3
<TO>.Kinematics.Parameter[7]	-1.0E12 to 1.0E12	Distance LF of the FCS from the lower connecting plate in the negative z direction of the KCS

Cylindrical robot

Cylindrical robot 3D

The kinematics "Cylindrical robot 3D" supports three axes and three degrees of freedom. The axes are configured as serial kinematics.

The following graphic shows the principal configuration and the typical working area of the kinematics:

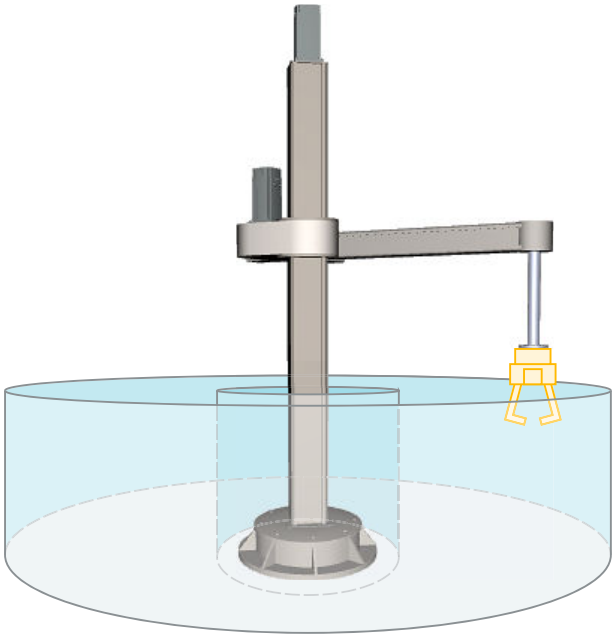

The kinematics consists of the following axes:

- A rotary axis A1 with rotation around the z axis of the kinematics coordinate system (KCS)
- A linear axis A2 in z direction of the KCS
- A linear axis A3 in x direction of the KCS

The kinematics consists of a base, a supporting column and a jib. Axis A1 rotates the supporting column with jib around the base. Axis A2 moves the jib vertically. Axis A3 moves the flange system horizontally on the jib. The kinematics enables a cylindrical working area.

Coordinate systems and zero position

The graphic below shows the following in the side view (xz plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero positions of the axes A1 and A2
- The deflection of the kinematics is indicated (dashed)

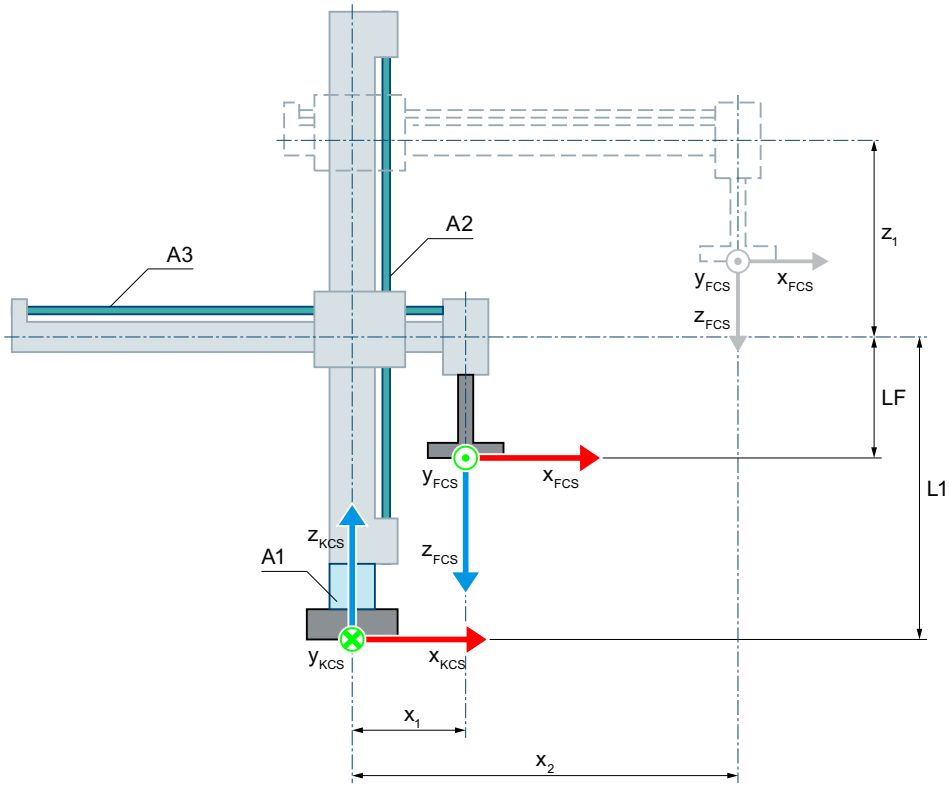

Zero position of the axes A1 and A2

L1 At zero position of the axis A2:

Distance of the FCS to the kinematics zero point (KZP) and flange length LF in z direction of the KCS

LF Flange length before the FCS in the z direction of the KCS

x_1 Positive deflection of the axis A3

At the zero position of axis A3, the z axis of the FCS is located on the z axis of the KCS. For mechanical reasons, the kinematics shown cannot approach the zero position of the axis A3.

Deflection of the kinematics

x_2 Positive deflection of the axis A3

z_1 Positive deflection of the axis A2

Legend for representation of the kinematics (Page 2160)

The graphic below shows the following in the top view (xy plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the kinematics is indicated (dashed)

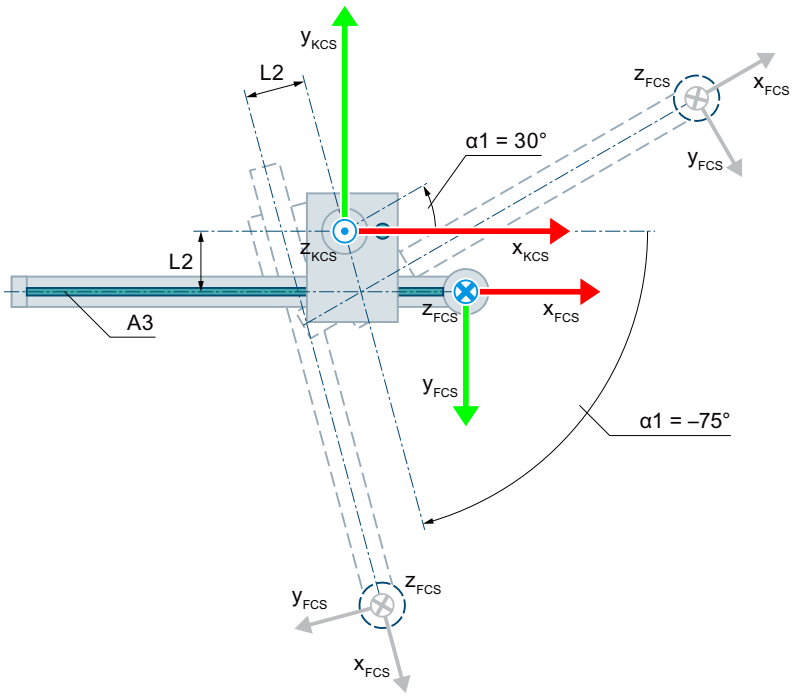

Zero position of the kinematics

L2 Distance of the axis A3 from the KZP in y direction of the KCS (negative value here)

Deflection of the kinematics

α1 Positive deflection of the axis A1 when α1 = 30°

Negative deflection of the axis A1 when α1 = -75°

Legend for representation of the kinematics (Page 2160)

The KCS with the kinematics zero point (KZP) is located at the base of the kinematics. You define the distance of the zero position of axis A2 in z direction of the KCS from the KZP using length L1. You define the distance of the axis A3 from the KNP in y direction of the KCS using length L2.

The flange coordinate system (FCS) is located on the axis A3, shifted by the length LF in negative z direction of the KCS.

The following table shows the zero position of the axes:

Axis	Zero position
A1	The jib with the axis A3 points in x_{KCS} direction.
A2	The axis A2 is at the position 0.0 of the interconnected technology object.
A3	The axis A3 is at the position 0.0 of the interconnected technology object.

Transformation area

The kinematics transformation covers the following traversing range (Page 2239) of the axes:

- Axis A1: -180.0° ≤ α1 < 180.0°
- Axis A2: No limiting
- Axis A3: No limiting

Note

Singular positions

The kinematics have singular positions (Page 2241).

A singular position occurs when the zero point of the flange coordinate system (FCS) is located on the z-axis of the kinematics coordinate system (KCS). Inverse transformation is not possible in this area. This position may result, e.g. if the length L2 is 0.0 due to the design.

The graphic below shows examples of permissible and impermissible joint positions for the transformation:

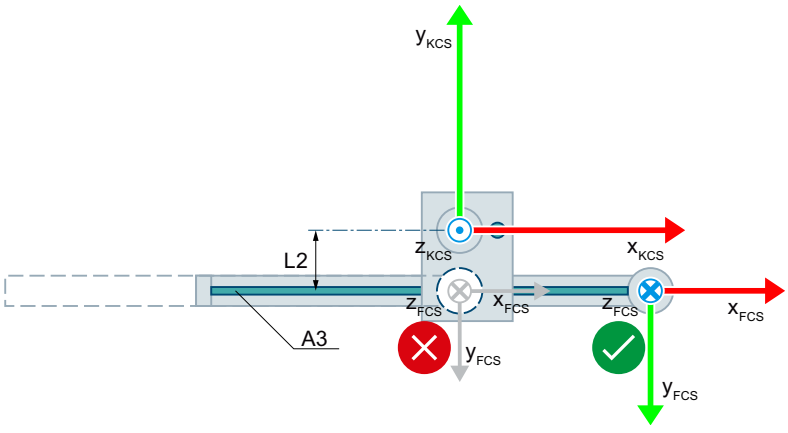

Permissible joint position

Invalid joint position for transformation with L2 = 0.0

Cylindrical robot 3D with orientation

The kinematics "Cylindrical robot 3D with orientation" supports four axes and four degrees of freedom. The axes are configured as serial kinematics.

The following graphic shows the principal configuration and the typical working area of the kinematics:

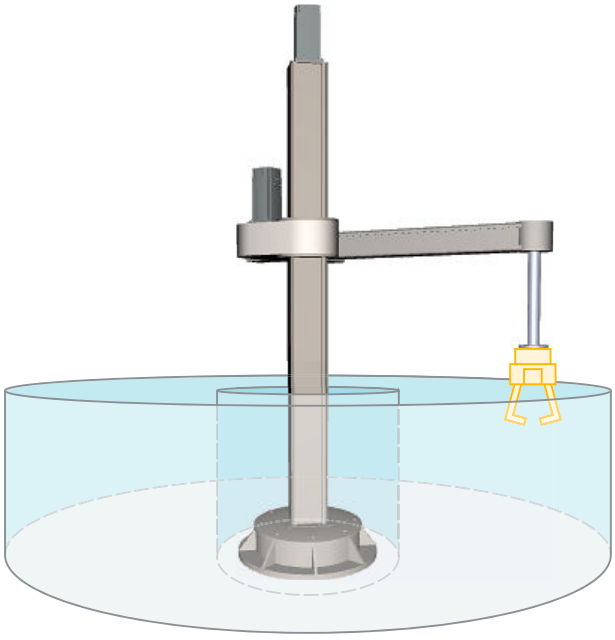

The kinematics consists of the following axes:

- A rotary axis A1 with rotation around the z axis of the kinematics coordinate system (KCS)
- A linear axis A2 in z direction of the KCS
- A linear axis A3 in x direction of the KCS
- One rotary axis A4 (orientation axis)

The kinematics consists of a base, a supporting column and a jib. Axis A1 rotates the supporting column with jib around the base. Axis A2 moves the jib vertically. Axis A3 moves the flange system horizontally on the jib. The kinematics enables a cylindrical working area. The orientation axis A4 enables the rotary motion of the tool.

Coordinate systems and zero position

The graphic below shows the following in the side view (xz plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero positions of the axes A1 and A2
- The deflection of the kinematics is indicated (dashed)

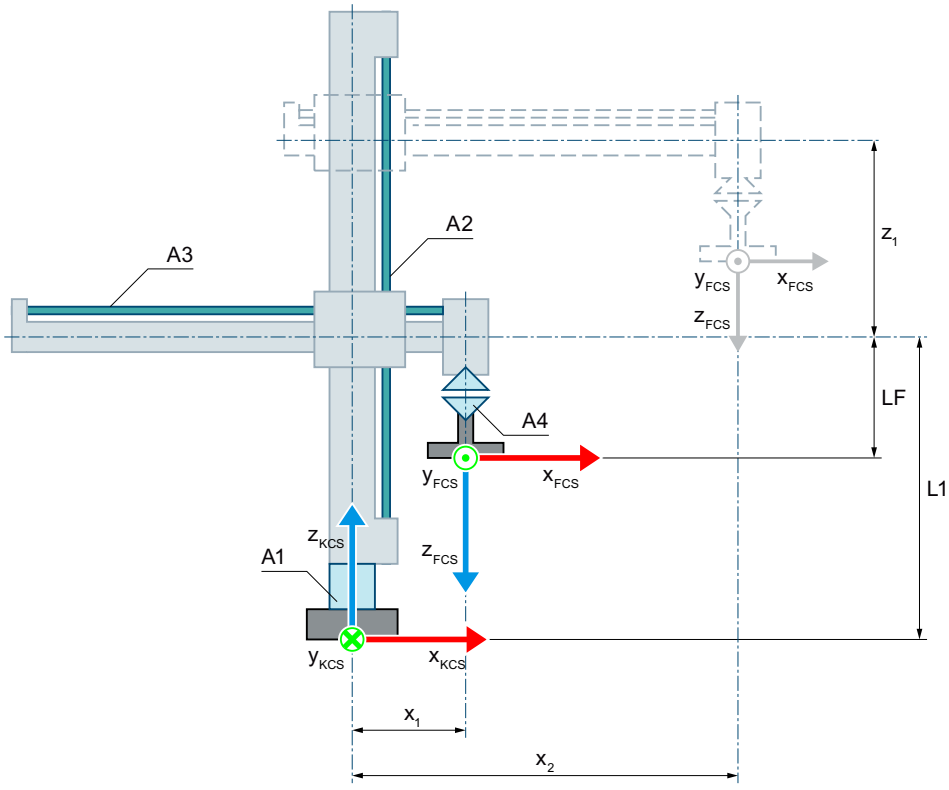

Zero position of the kinematics

L1 At zero position of the axis A2:
Distance of the FCS to the kinematics zero point (KZP) and flange length LF in z direction of the KCS

LF Flange length before the FCS in the z direction of the KCS

x_1 Positive deflection of the axis A3
At the zero position of axis A3, the z axis of the FCS is located on the z axis of the KCS. For mechanical reasons, the kinematics shown cannot approach the zero position of the axis A3.

Deflection of the kinematics

x_2 Positive deflection of the axis A3

z_1 Positive deflection of the axis A2

Legend for representation of the kinematics (Page 2160)

The graphic below shows the following in the top view (xy plane):

- The position of the axes and the coordinate systems KCS and FCS
- The zero position of the kinematics
- The positive/negative deflection of the kinematics is indicated (dashed)

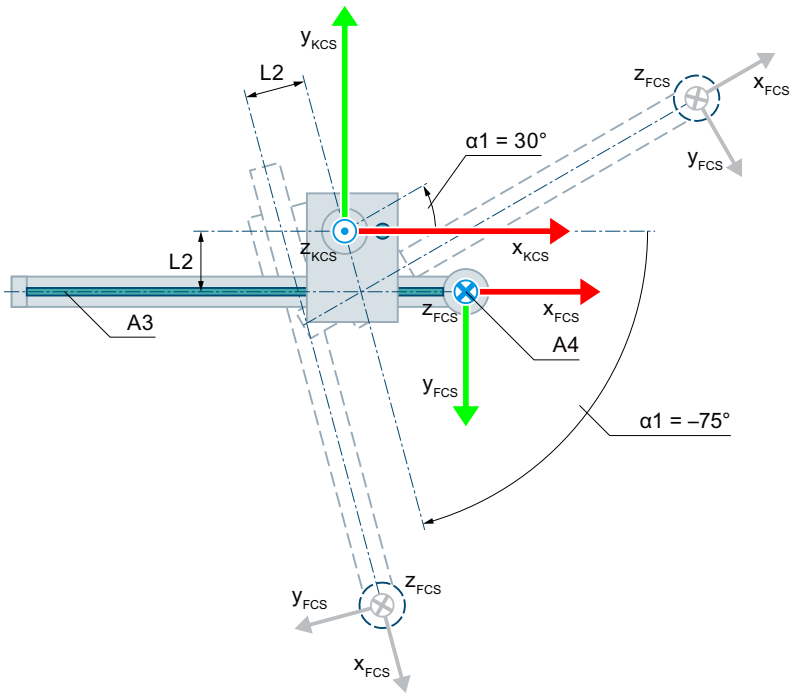

Zero position of the kinematics

L2 Distance of the axis A3 to the KZP in y direction of the KCS (negative value in this case)

Deflection of the kinematics

α1 Positive deflection of the axis A1 when α1 = 30°

Negative deflection of the axis A1 when α1 = -75°

Legend for representation of the kinematics (Page 2160)

The KCS with the kinematics zero point (KZP) is located at the base of the kinematics. You define the distance of the zero position of axis A2 in z direction of the KCS from the KZP using length L1. You define the distance of the axis A3 from the KNP in y direction of the KCS using length L2.

The flange coordinate system (FCS) is located on the axis A3, shifted by the length LF in negative z direction of the KCS.

The following table shows the zero position of the axes:

Axis	Zero position
A1	The jib with the axis A3 points in x_{KCS} direction.
A2	The axis A2 is at the position 0.0 of the interconnected technology object.
A3	The axis A3 is at the position 0.0 of the interconnected technology object.
A4	At the zero position of axis A1, the x axis of the FCS points in the direction of the x axis of the KCS.

Compensation of mechanical axis couplings

You can configure a mechanical axis coupling of axis A4 to axis A2 for the kinematics. The kinematics transformation compensates for the configured mechanical axis coupling. The axis coupling between axis A4 and axis A2 is implemented as a leadscrew pitch. With a coupling factor of 1.0, 360.0° on axis A4 corresponds to a distance of -1.0 mm on axis A2.

Transformation area

The kinematics transformation covers the following traversing range (Page 2239) of the axes:

- Axis A1: -180.0° ≤ α1 < 180.0°
- Axis A2: No limiting
- Axis A3: No limiting
- Axis A4: No limiting
  An angle greater than 360° can be defined for the orientation. But coordinate A of the tool center point (TCP) is mapped in the range -180° to +180°.

Note

Singular positions

The kinematics have singular positions (Page 2241).

A singular position occurs when the zero point of the flange coordinate system (FCS) is located on the z-axis of the kinematics coordinate system (KCS). Inverse transformation is not possible in this area. This position may result, e.g. if the length L2 is 0.0 due to the design.

The graphic below shows examples of permissible and impermissible joint positions for the transformation:

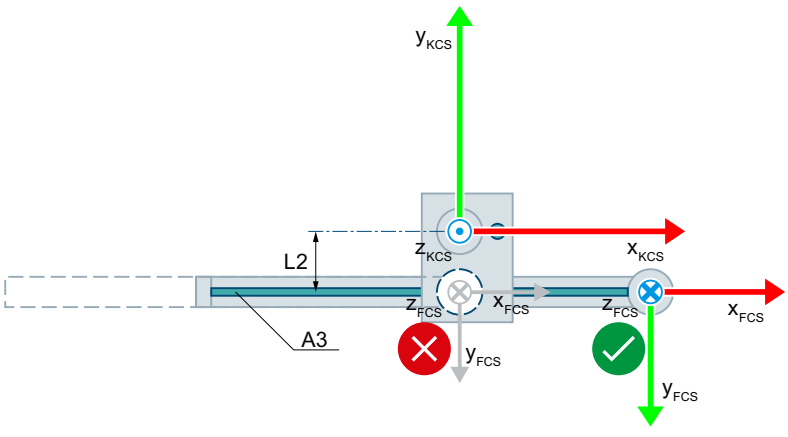

Permissible joint position

Invalid joint position for transformation with L2 = 0.0

Cyclic robot tags

Cylindrical robot 3D

You define the "Cylindrical robot 3D" kinematics using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	21	Cylindrical robot 3D
<TO>.Kinematics.Parameter[1]	-1.0E12 to 1.0E12	Distance L1 of the zero position of the axis A2 to the kinematics zero point in z direction of the kinematics coordinate system (KCS)
<TO>.Kinematics.Parameter[2]	-1.0E12 to 1.0E12	Distance L2 between the axes A2 and A3 in y direction of the KCS
<TO>.Kinematics.Parameter[3]	-1.0E12 to 1.0E12	Distance of the flange coordinate system from the axis A3 in the negative z direction of the KCS

Cylindrical robot 3D with orientation

You define the "Cylindrical robot 3D with orientation" kinematics using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	22	Cylindrical robot 3D with orientation
<TO>.Kinematics.Parameter[1]	-1.0E12 to 1.0E12	Distance L1 of the zero position of the axis A2 to the kinematics zero point in z direction of the KCS
<TO>.Kinematics.Parameter[2]	-1.0E12 to 1.0E12	Distance L2 between the axes A2 and A3 in y direction of the KCS
<TO>.Kinematics.Parameter[3]	-1.0E12 to 1.0E12	Distance of the flange coordinate system from the axis A3 in the negative z direction of the KCS
<TO>.Kinematics.Parameter[4]	-	Mechanical axis coupling of axis A4 to A2 present/not present
	0	Not present
	1	Present
<TO>.Kinematics.Parameter[5]	-1.0E12 to 1.0E12	Mechanical axis coupling factor of axis A4 to axis A2

Tripod

Tripod 3D

The kinematics "Tripod 3D" supports three axes and three degrees of freedom. The axes are configured as parallel kinematics.

The following graphic shows the principal configuration and the typical working area of the kinematics:

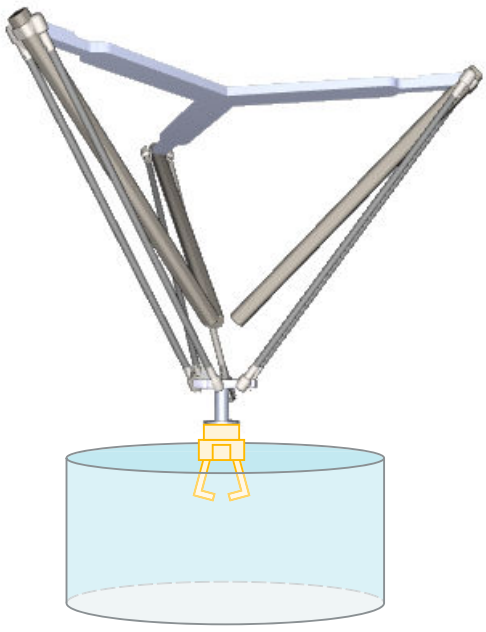

The kinematics consists of three linear axes A1, A2 and A3.

The kinematics is modeled suspended and consists of an upper connecting plate, three arms and a lower connecting plate. The axes for the motion of the arms consist of rails with sliding carriages. The rails with the sliding carriages are fastened to the upper connecting plate. Connecting rods connect the sliding carriages to the lower connecting plate. The tool is suspended from the lower connecting plate. The parallelogram structures of the connecting rods keep the lower connecting plate parallel to the xy plane of the KCS.

Coordinate systems and zero position

The graphic below shows the following in the top view (xy plane):

- The position of the kinematics coordinate system (KCS)
- The angles of the axes A1, A2 and A3 to one another

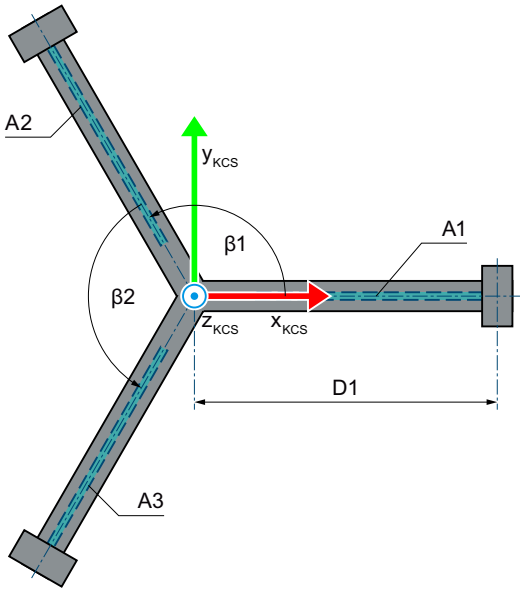

β1 Angle between the axes A1 and A2

β2 Angle between the axes A2 and A3

Legend for representation of the kinematics (Page 2160)

The following graphic shows the top view of the position of the flange coordinate system (FCS) in the xy plane of the lower connecting plate:

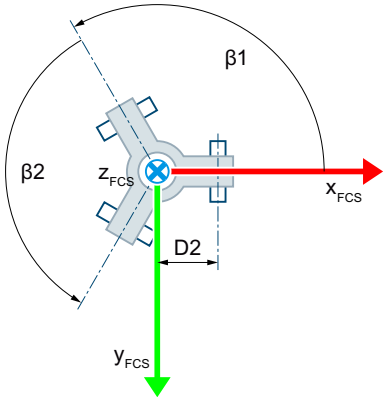

The graphic below shows the following in the front view (xz plane):

- The position of the axis A1 and the coordinate systems KCS and FCS
- The zero position of the axis A1
- The positive deflection of the axis A1 is indicated (dashed)

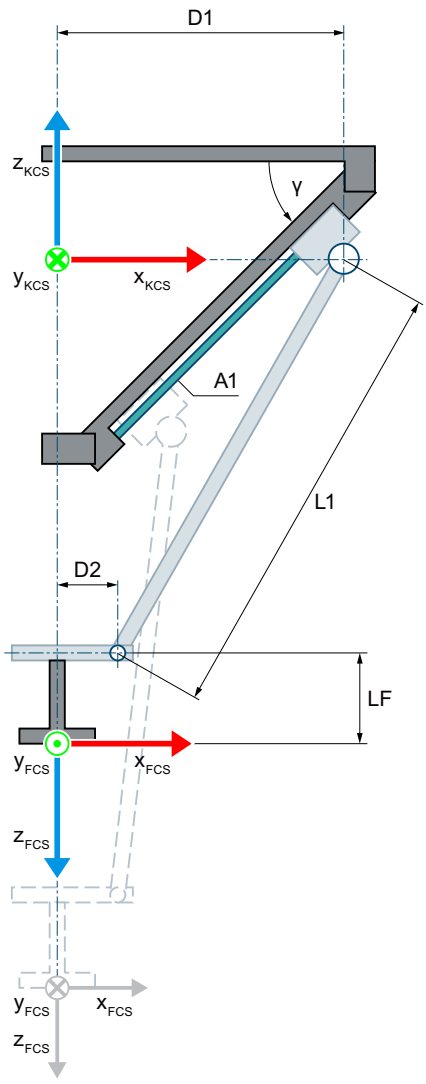

Zero position of the kinematics

D1	Distance of the upper hinge points of the connecting rods to the center of the upper connecting plate
D2	Distance of the lower hinge points of the connecting rods to the center of the lower connecting plate
L1	Length of the connecting rods
LF	Flange length before the FCS in the z direction of the KCS
γ	Angle between the upper connecting plate (xy plane of the KCS) and the rail of the axis A1 ($0.0° \leq \gamma < 90.0°$)

D1, D2, L1 and γ are identical for the three arms of the kinematics.

Deflection of the kinematics with deflection of the axis A1 in positive direction

Legend for representation of the kinematics (Page 2160)

The KCS with the kinematics zero point (KZP) is located at the center point of the upper connecting plate. The kinematics zero point is centered relative to the zero positions of the axes A1, A2 and A3.

The FCS is located in the center point of the lower connecting plate with identical distance D2 to the joint points of each connecting rod. You shift the FCS in the negative z direction of the KCS using length LF.

At the zero position, axes A1, A2 and A3 are in the x-y plane of the KCS.

Transformation area

The kinematics transformation covers the following traversing range (Page 2239) of the axes:

- Axes A1, A2 and A3: 0.0 ≤ Travel distance

Tripod 3D with orientation

The kinematics "Tripod 3D with orientation" supports four axes and four degrees of freedom. The axes are configured as parallel kinematics.

The following graphic shows the principal configuration and the typical working area of the kinematics:

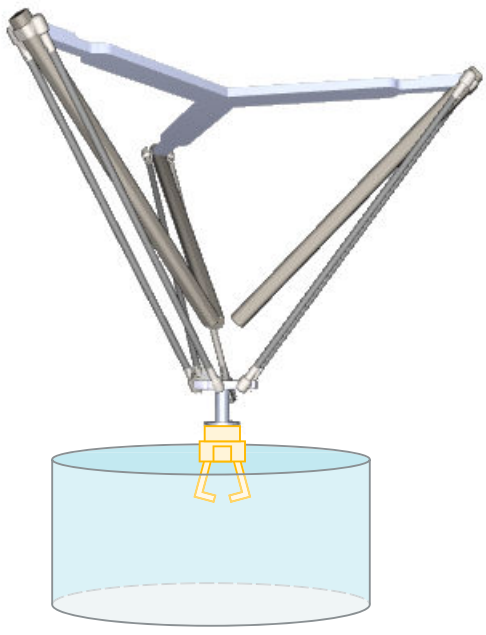

The kinematics consists of the following axes:

- Three linear axes A1, A2 and A3
- One rotary axis A4 (orientation axis)

The kinematics is modeled suspended and consists of an upper connecting plate, three arms and a lower connecting plate. The axes for the motion of the arms consist of rails with sliding carriages. The rails with the sliding carriages are fastened to the upper connecting plate. Connecting rods connect the sliding carriages to the lower connecting plate. The tool is suspended from the lower connecting plate. The parallelogram structures of the connecting rods keep the lower connecting plate parallel to the xy plane of the KCS. The orientation axis A4 enables the rotary motion of the tool.

Coordinate systems and zero position

The graphic below shows the following in the top view (xy plane):

- The position of the kinematics coordinate system (KCS)
- The angles of the axes A1, A2 and A3 to one another

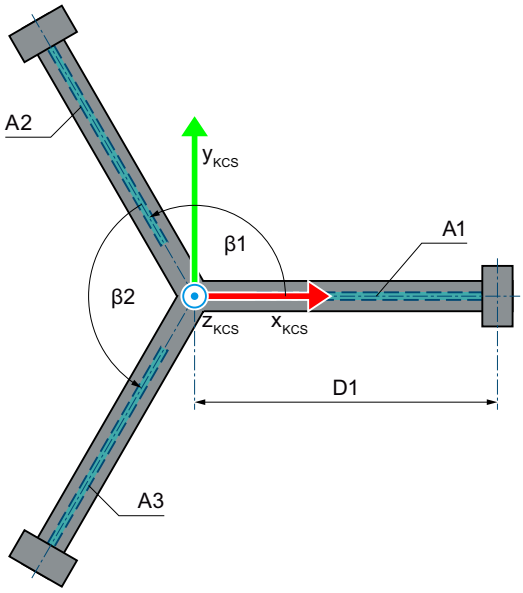

β1 Angle between the axes A1 and A2

β2 Angle between the axes A2 and A3

Legend for representation of the kinematics (Page 2160)

The following graphic shows the top view of the position of the flange coordinate system (FCS) in the xy plane of the lower connecting plate:

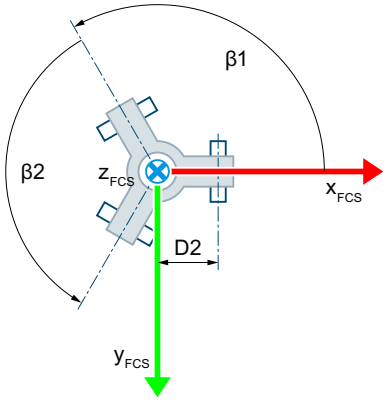

The graphic below shows the following in the side view:

- The position of the axis A1 and the coordinate systems KCS and FCS
- The zero position of the axis A1
- The positive deflection of the axis A1 is indicated (dashed)

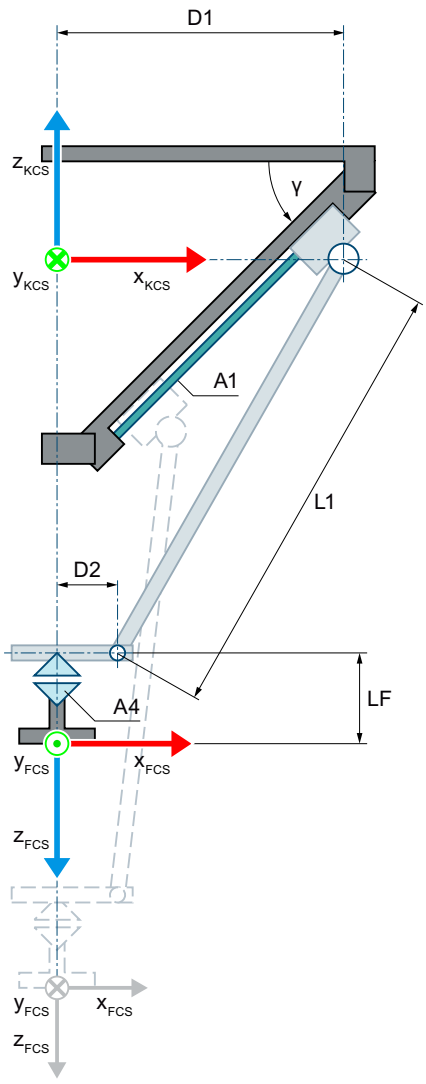

Zero position of the kinematics

D1	Distance of the upper hinge points of the connecting rods to the center of the upper connecting plate
D2	Distance of the lower hinge points of the connecting rods to the center of the lower connecting plate
L1	Length of the connecting rods
LF	Flange length before the FCS in the z direction of the KCS
γ	Angle between the upper connecting plate (xy plane of the KCS) and the rail of the axis A1 (0.0° ≤ γ < 90.0°)

D1, D2, L1 and γ are identical for the three arms of the kinematics.

Deflection of the kinematics with deflection of the axis A1 in positive direction

Legend for representation of the kinematics (Page 2160)

The KCS with the kinematics zero point (KZP) is located at the center point of the upper connecting plate. The kinematics zero point is centered relative to the zero positions of the axes A1, A2 and A3.

The FCS is located in the center point of the lower connecting plate with identical distance D2 to the joint points of each connecting rod. You shift the FCS in the negative z direction of the KCS using length LF.

At the zero position, axes A1, A2 and A3 are in the x-y plane of the KCS. At the zero position of axis A4, the x axis of the FCS points in the direction of the x axis of the KCS.

Transformation area

The kinematics transformation covers the following traversing range (Page 2239) of the axes:

- Axes A1, A2 and A3: 0.0 ≤ Travel distance
- Axis A4: No limiting
  An angle greater than 360° can be defined for the orientation. But coordinate A of the tool center point (TCP) is mapped in the range -180° to +180°.

Tags of tripod

You define the tripod kinematics using the following tags of the technology object:

Tags	Values	Description
<TO>.Kinematics.TypeOfKinematics	23	Tripod 3D
	24	Tripod 3D with orientation
<TO>.Kinematics.Parameter[1]	0.0 to 1.0E12	Distance D1 (radius of the upper connecting plate)
<TO>.Kinematics.Parameter[2]	0.001 to 1.0E12	Length L1 of connecting rods
<TO>.Kinematics.Parameter[3]	0.0° to 89.999°	Angle γ between the linear axes and the x-y plane of the kinematics coordinate system
<TO>.Kinematics.Parameter[4]	0.0 to 1.0E12	Distance D2 (radius of the lower connecting plate)
<TO>.Kinematics.Parameter[5]	90.001° to 179.998°	Angle β1 between the axes A1 and A2
<TO>.Kinematics.Parameter[6]	90.001° to 179.998°	Angle β2 between the axes A2 and A3
<TO>.Kinematics.Parameter[7]	-1.0E12 to 1.0E12	Distance of the flange coordinate system from the lower connecting plate in the negative z direction of the kinematics coordinate system

User-defined kinematics systems

You can configure user-defined kinematics systems with corresponding axis interconnections:

- User-defined 2D
- User-defined 2D with orientation
- User-defined 3D
- User-defined 3D with orientation

The configuration supports you in the interconnection of the positioning axes in a user-defined kinematics. In addition, up to 32 tags for defining the geometry of your kinematics area available on the system level.

You have to program the user transformation (Page 2243) of the Cartesian positions and the axis positions and axis dynamics. Predefined interfaces are available on the system level.

See also

User transformation (Page 2243)

Tags of user-defined kinematics systems

You configure the user-defined kinematics systems using the following tags of the technology object:

Tag	Values	Description
<TO>.Kinematics.TypeOfKinematics	31	User-defined 2D
	32	User-defined 2D with orientation
	33	User-defined 3D
	34	User-defined 3D with orientation
<TO>.Kinematics.Parameter[1..32]	-	Up to 32 user-specific parameters

Kinematics transformation

The kinematics transformation is the conversion between the Cartesian coordinates of the kinematics motion and the setpoints for the individual kinematics axes:

- Forward transformation
  Calculation of the Cartesian coordinates from the axis positions of the kinematics axes
- Inverse transformation
  Calculation of the axis positions of the kinematics axes from the Cartesian coordinates

The kinematics transformation converts the position values and the dynamic values (velocity, acceleration).

The kinematics technology object provides the kinematics transformation for the predefined kinematics types on the system level. In the case of user-defined kinematics systems, you must calculate the user transformation (Page 2243) in its own program.

Transformation for predefined kinematics systems

Reference points

The kinematics transformation uses the following reference points:

- Kinematics zero point (KZP)
- Zero positions of the kinematics axes
- Tool center point (TCP)

The positive direction of the axes for the kinematics transformation is dependent on the kinematics type (Page 2160). Configure the positive direction on the positioning axis/ synchronous axis technology object corresponding to the positive direction of the axis in the kinematics.

Kinematics zero point (KZP)

The coordinate origin of the kinematics coordinate system (KCS) is the KZP. You configure the geometry parameters of the kinematics starting from the KZP.

Zero positions of the kinematics axes

The position 0.0 on the positioning axis/synchronous axis technology object defines the zero position of the kinematics axis. Reference the axes in such a way that the axes indicate the position 0.0 in the zero position of the kinematics. The zero position of the kinematics depends on the kinematics type (Page 2160).

Tool center point (TCP)

The position of the TCP results from the geometry parameters and the tool frame.

Traversing range and transformation area

The transformation area is the area of the axis positions that the kinematics transformation covers. The kinematics type defines the transformation area for the individual kinematics axes. You will find information on the transformation area in the description of the individual kinematics systems (Page 2160).

The hardware and software limit switches of an axis define the maximum traversing range and the working area. The working area of a kinematics axis can be greater than or less than the transformation area depending on the axis configuration:

- Work area > Transformation area
  When a kinematics axis exits the transformation area during a kinematics motion, the kinematics technology object outputs the technology alarm 803. The kinematics motion is aborted and the axes stop with the maximum dynamic values configured for the axes (alarm reaction: Stop with maximum dynamic values of the axes).
- Work area ≤ Transformation area
  When a kinematics axis runs into the software limit switch, the positioning axis/synchronous axis technology object outputs the technology alarm 533. The axis stops with the maximum dynamic values configured for the axis (alarm reaction: Stop with maximum dynamic values). When the axis is stopped, the kinematics technology object outputs the technology alarm 801. The kinematics motion is aborted and the axes stop with the maximum dynamic values configured for the axes (alarm reaction: Stop with maximum dynamic values of the axes).

The following graphic shows the relationship between the working area of the axis and the transformation area:

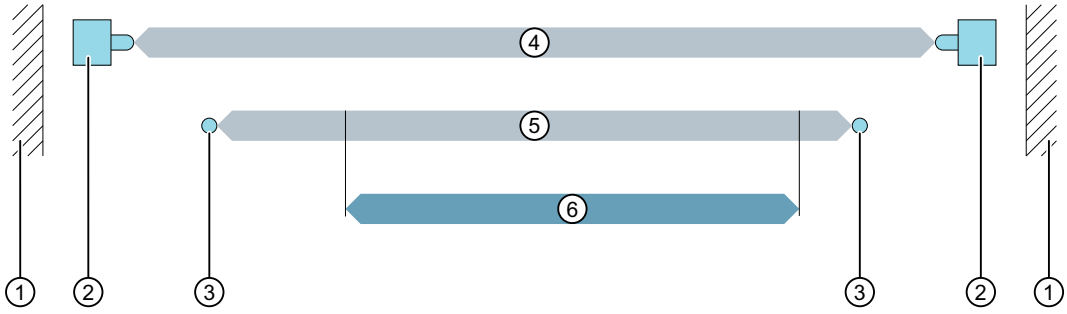

① Mechanical end stop
② Hardware limit switch for positioning axis/synchronous axis technology object
③ Software limit switch for positioning axis/synchronous axis technology object
④ Maximum traversing range of the axis
⑤ Work area of the axis
⑥ Transformation area
(here working area > transformation area)

Joint position spaces (kinematics-dependent)

Depending on the kinematics type, a kinematics system can reach Cartesian coordinates via various joint positions. The kinematics type (Page 2160) defines the possible joint positions and the positive and negative joint position space. The joint positioning spaces are limited by the respective transformation areas. In addition, with the kinematics type "Delta picker" there are further limitations due to invalid joint positions and singular positions (Page 2241) with the kinematics types "articulated arm", "SCARA" and "cylindrical robot". Also note the resulting constructional limitations due to the installation location of the kinematics.

The kinematics technology object indicates the current joint position in the "<TO>.StatusKinematics.LinkConstellation" tag.

The kinematics system cannot exit the joint position space during a kinematics motion. You can change the joint position space using single axis motions.

Example: "SCARA" kinematics type

A "SCARA" kinematics is to relocate an object one pallet to another pallet. Due to a wall, the kinematics system cannot reach the second pallet without the axis A2 changing the joint position space.

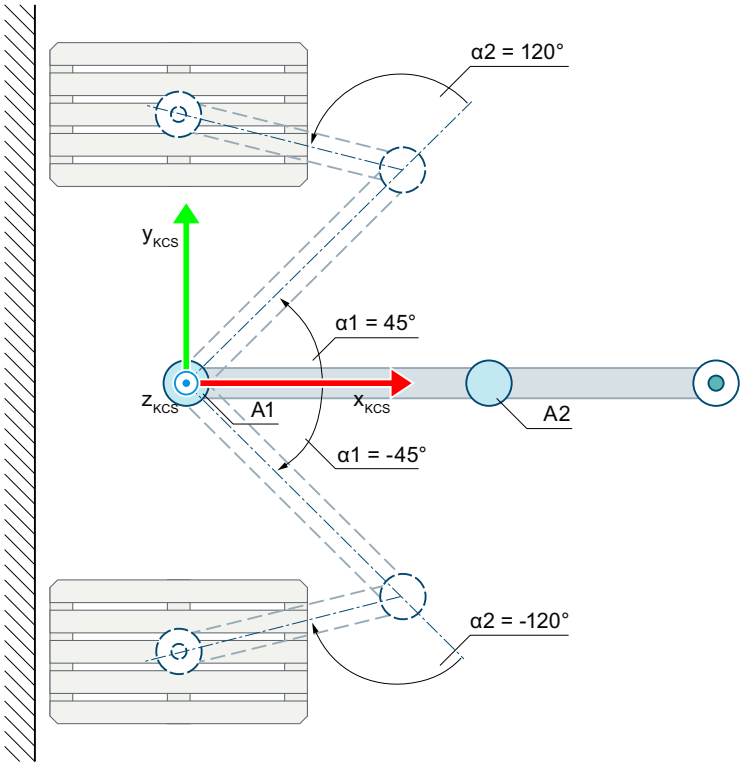

Zero position of the kinematics

Deflection of the kinematics in the positive direction when α1 = 45.0° with positive joint position when α2 = 120.0°

Deflection of the kinematics in the negative direction when α1 = -45.0° with negative joint position when α2 = -120.0°

α1 Deflection of the axis A1 in positive direction when α1 = 45.0°

Deflection of the axis A1 in the negative direction when α1 = -45.0°

α2 The deflection of the axis A2 in the positive direction when α2 = 120.0° produces a positive joint position.

The deflection of the axis A2 in negative direction when α2 = -120.0° produces a negative joint position.

Legend for representation of the kinematics (Page 2160)

Singular positions

Depending on the kinematics type, inverse transformation Cartesian coordinates are possible, which cannot be clearly converted into axis positions of the kinematics axes. This occurs when the zero point of the flange coordinate system (FCS) is located on the z-axis of the kinematics coordinate system (KCS). The Cartesian coordinates at which this behavior occurs are referred to as singular positions.

The following kinematics types of the kinematics technology object have singular positions.

- Articulated arm 3D
- Articulated arm 3D with orientation
- SCARA 3D with orientation
- Cylindrical robot 3D
- Cylindrical robot 3D with orientation

Behavior in singular positions

A path motion on or through a singular position is not possible. When the singular position is reached, the technology alarm 803 "Error during calculation of the transformation" is output (alarm reaction: Stop with maximum dynamic values of the axes). The kinematics system reduces the overruns of the dynamics caused by setpoints to the maximum dynamics of the axis. The reduction of the overruns of the dynamics can result in unexpected axis motion.

Dynamic increase in the vicinity of singular positions

If the path motion runs in the vicinity of singular positions, kinematics axes may accelerate or decelerate considerably and travel at very high speed. The dynamic limits of the axes may be exceeded because of this. If the dynamic limits of the axes with regard to speed, acceleration or deceleration are exceeded, this is displayed on the technology object data block of the corresponding axis and the technology alarm 511 "Dynamic limit is violated by the kinematics motion" is triggered. This is a warning and contains no alarm response. The kinematics motion is not stopped.

The size of the area in which this behavior occurs depends on the kinematics used.

Possible measures

To prevent travel in or in the vicinity of singular positions, take the following measures:

1. Plan the path motions in such a way that the kinematics do not travel in or in the vicinity of singular positions.
2. Check whether travel with your kinematics in or in the vicinity of singular positions can be prevented by blocked zones or software limit switches.
3. Therefore, use a kinematics system with singular positions within or at the threshold of the transformation range with enabled dynamics adaptation. By doing this you avoid unexpected dynamics overruns and resulting axis motions.

Mechanical couplings (kinematics-dependent)

If the position of a kinematics axis changes due to the motion of another kinematics axis, these two axes are mechanically coupled. Mechanical couplings between two kinematics axes can arise for reasons of construction. For example, if the orientation axis of the "SCARA" kinematics is coupled to the spindle of a linear axis, the orientation changes due to the motion of the linear axis.

The kinematics transformation compensates for the mechanical couplings with a coupling factor. You specify the mechanical couplings and coupling factors depending on the kinematics type during configuration of the kinematics.

Transformation for user-defined kinematics systems

User transformation

Unlike for predefined kinematics types, you must calculate the transformation for user-defined kinematics systems in your own program. Like for predefined kinematics types, the kinematics technology object performs the following tasks:

- Processing of Motion Control instructions
- Monitoring functions
- Communication with the interconnected axes

You program the user transformation of the Cartesian coordinates and the axis-specific setpoints in the MC-Transformation [OB98] organization block. This programming includes the transformation of the positions and the dynamic values (velocity, acceleration, jerk). You freely define the parameters of a user-defined kinematics system in the tags of the kinematics technology object "<TO>.Kinematics.Parameter[1..32]" or under "Technology object > Configuration > Geometry".

When you add the MC-Transformation [OB98] in the TIA Portal, the system data block "TransformationParameter" is automatically created under "Program blocks > System blocks > Program resources". In the properties of the organization block under "General > Transformation", the MC-Transformation [OB98] indicates the number of the system data block "TransformationParameter". You write and read the axis-specific data or the Cartesian data of the kinematics to be transformed in the system data block "TransformationParameter".

Programming

The graphic below shows the interfaces and the interaction of system performance and user transformation:

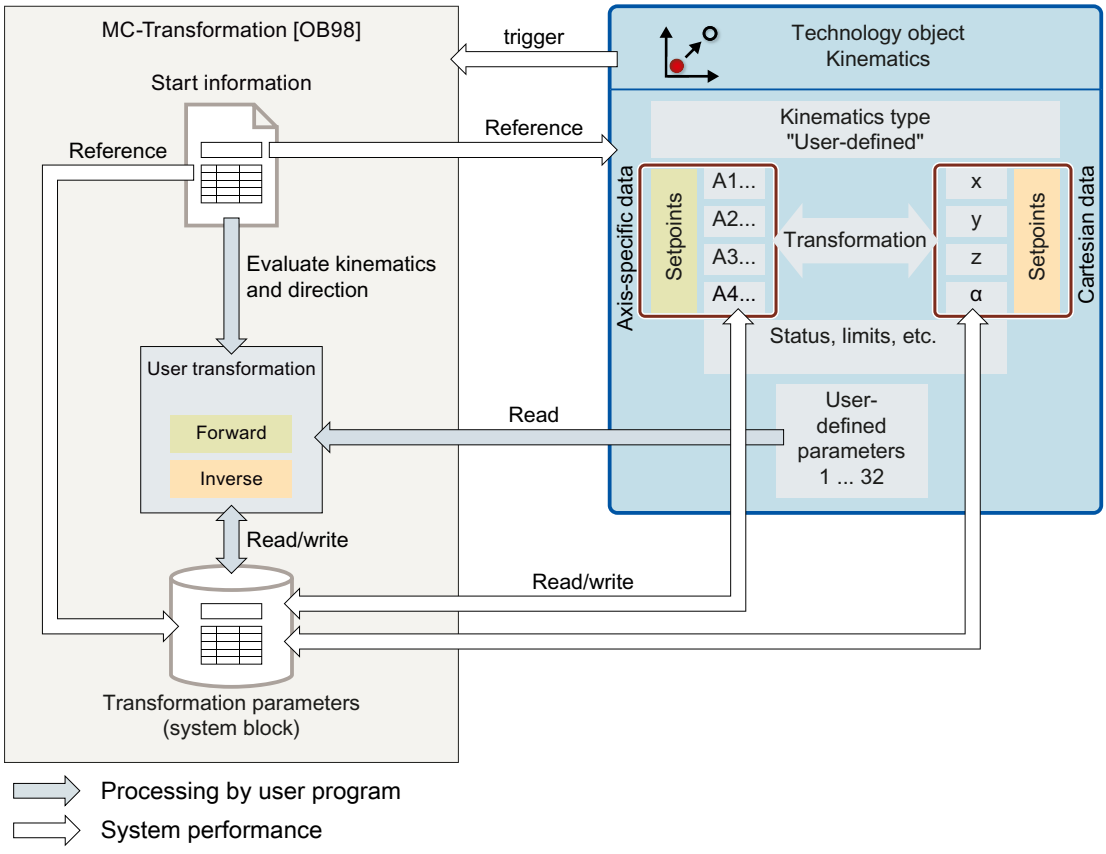

The kinematics technology object automatically calls the MC-Transformation [OB98]. The MC-Transformation [OB98] contains the following start information:

- Kinematics technology object that calls the MC-Transformation [OB98]
- Required direction of the transformation (forward or inverse transformation)
- Processing context of the transformation (current motion or motion planning)
- Pointer to the system data block "TransformationParameter" (VARIANT)

You evaluate this status information in your user program. In MC-Transformation [OB98]. you program the algorithms for the calculation of the axis-specific data or the Cartesian data of all user-defined kinematics systems. You read the kinematics parameters needed for this from the tags of the "<TO>.Kinematics.Parameter[1..32]" technology object. You write the result of the calculation to the "TransformationParameter" interface.

The transformation parameters are then applied automatically to the kinematics technology object and processed further. The kinematics technology object outputs the setpoints to the kinematics axes.

Restrictions during transformation of multiple kinematics

Each kinematics technology object calls the MC-Transformation [OB98]. Only one kinematics can be transformed in an MC-Transformation [OB98]. Only one MC-Transformation [OB98] can be called in each servo clock.

MC-Transformation [OB98]

Reference declaration for system data block "TransformationParameter"

You must specify a reference to the data type for the system data block "TransformationParameter" in the "MC-Transformation [OB98]". For this purpose, you specify a tag with the following data type in the "Temp" area of the block interface:

"REF_TO TO_Struct_TransformationParameter_V1"

To enable access to the system data block "TransformationParameter", assign the data type "TO_Struct_TransformationParameter_V1" using the following casting command:
`#P ?= #TransformationParameters;`

The declaration is described in a program example (Page 2247).

Block call

The MC-Transformation [OB98] is called in the Motion Control application cycle according to the configured priority. When the MC-Transformation [OB98] is called, the kinematics technology object assigns its parameters:

Parameter	Declaration	Data type	Default value	Description	
KinematicsObject	INPUT	DB_ANY	-	Kinematics technology object for which the MC-Transformation [OB98] calculates the transformation when called.	
ExecutionContext	INPUT	DINT	-	Processing context of the MC-Transformation OB	
				0	MOTION_EXECUTION Calculation of the axis setpoints in the motion execution in the MC-Interpolator [OB92]. The calculated values are necessary for the current motion control.
				1	NON_MOTION_EXECUTION The transformation is necessary for the motion planning (currently no motion).
TransformationType	INPUT	DINT	-	Calculation called for	
				0	Forward transformation Calculation of the Cartesian parameters from the axis positions
				1	Inverse transformation Calculation of the axis-specific parameters from the Cartesian parameters
TransformationParameters	InOut	VARIANT	-	Pointer to the system data block "TransformationParameter"	

Parameter	Declaration	Data type	Default value	Description	
FunctionResult	OUTPUT	DINT	-	Return value of the MC-Transformation [OB98] to the kinematics technology object	
				0	Calculation performed and parameters output
				≠ 0	Error during calculation (user-defined) If an error occurs during the calculation, the kinematics technology object stops the motion. The kinematics technology object outputs a technology alarm with the error ID as an accompany value and deletes the job sequence. The value can be greater than or less than zero.

Priority

You configure the priority of the MC-Transformation [OB98] in the properties of the organization block under "General > Attributes > Priority". For the priority you can set values from 17 to 25 (default setting 25):

- The priority of MC-Transformation [OB98] must be at least one level lower than the priority of MC-Servo [OB91].
- The priority of MC-Transformation [OB98] must be at least one level higher than the priority of MC-Interpolator [OB92].

Tags of the system data block "TransformationParameter"

The following table shows the tags in the system data block "TransformationParameter":

Tag	Data type	Description
AxisData.	STRUCT_ Transformation AxisData_V1	Axis-specific parameters
a1Position	LREAL	Position setpoint of the axis A1
a1Velocity	LREAL	Velocity setpoint of the axis A1
a1Acceleration	LREAL	Acceleration setpoint of the axis A1
a2Position	LREAL	Position setpoint of the axis A2
a2Velocity	LREAL	Velocity setpoint of the axis A2
a2Acceleration	LREAL	Acceleration setpoint of the axis A2
a3Position	LREAL	Position setpoint of the axis A3
a3Velocity	LREAL	Velocity setpoint of the axis A3
a3Acceleration	LREAL	Acceleration setpoint of the axis A3
a4Position	LREAL	Position setpoint of the axis A4
a4Velocity	LREAL	Velocity setpoint of the axis A4
a4Acceleration	LREAL	Acceleration setpoint of the axis A4

Tag	Data type	Description
CartesianData	STRUCT_ Transformation CartesianDa-ta_V1	Cartesian parameters and joint position
xPosition	LREAL	x position
xVelocity	LREAL	x velocity
xAcceleration	LREAL	x acceleration
yPosition	LREAL	y position
yVelocity	LREAL	y velocity
yAcceleration	LREAL	y acceleration
zPosition	LREAL	z position
zVelocity	LREAL	z velocity
zAcceleration	LREAL	z acceleration
aPosition	LREAL	A-position (orientation)
aVelocity	LREAL	A velocity (orientation)
aAcceleration	LREAL	A acceleration (orientation)
LinkConstellation	DWORD	Joint position area

Program example for a user-defined kinematics systems

The following describes a example for the user transformation in the MC-Transformation [OB98] of a 2D kinematics with the name "KinematicsUserDefined2D". For these kinematics, two transformation parameters were defined under "Technology object > Configuration > Geometry".

The following table shows the declaration of the tags used:

Tag	Declaration	Data type	Description
KinematicsOb-ject	Input	DB_ANY	Reference to the technology object
Transformation-Type	Input	DInt	Transformation direction 0: Forward transformation 1: Inverse transformation
FunctionResult	Output	DInt	Transformation result 0: Successful < 0 Error
Transformation-Parameters	InOut	Variant	Pointer to the system data block "TransformationParameter"
P	Temp	REF_TO TO_Struct_Tran sformationPara-meter_V1	Temporary tag for the casting command
GearRatioA1	Temp	LReal	Temporary tag for read the defined transformation parameters
GearRatioA2	Temp	LReal	Temporary tag for read the defined transformation parameters
InvalidCast	Constant	DInt	Return value for unsuccessful casting

The program example is structured as follows:

- Casting command for access to the system data block "TransformationParameter"
- Evaluation of the technology object
- Read the defined transformation parameters
- Evaluation of the transformation direction
- Calculation of the Cartesian coordinates from the axis positions of the kinematics axes (forward transformation)
- Calculation of the axis positions of the kinematics axes from the Cartesian coordinates (backward transformation)

SCL

```
//Caste of the variant "TransformationParameters" to the referenced datatype
//"TO_Struct_TransformationParameter_V1".
//This has to be done in order to access the variant pointer, which references
//the "TransformationParameters" where the "AxisData" and "CartesianData" for
//the calculation of user transformation are stored.
#P ?= #TransformationParameters;

//Check if cast of "TransformationParameters" was successfull. Otherwise abort calculation.
IF #P = NULL THEN
    #FunctionResult := #InvalidCast;
    RETURN;
END_IF;

//Check if "KinematicsUserDefined2D" needs transformation.
IF #KinematicsObject = "KinematicsUserDefined2D" THEN

    //Read the user defined cartesian parameters.
    #GearRatioA1 := "KinematicsUserDefined2D".Kinematics.Parameter[1];
    #GearRatioA2 := "KinematicsUserDefined2D".Kinematics.Parameter[2];

    //Calculate the forward transformation "AxisData" -> "CartesianData".
    //The system fills the "AxisData" of "TransformationParameters" with values.
    //To calculate the "CartesianData" evaluate "AxisData".
    IF #TransformationType = 0 THEN

        //Calculate the position, velocity and acceleration component for the x-vector.
        #P^.CartesianData.xPosition := #P^.AxisData.a1Position * #GearRatioA1;
        #P^.CartesianData.xVelocity := #P^.AxisData.a1Velocity * #GearRatioA1;
        #P^.CartesianData.xAcceleration := #P^.AxisData.a1Acceleration * #GearRatioA1;

        //Calculate the position, velocity and acceleration component for the z-vector.
        #P^.CartesianData.zPosition := #P^.AxisData.a2Position * #GearRatioA2;
        #P^.CartesianData.zVelocity := #P^.AxisData.a2Velocity * #GearRatioA2;
        #P^.CartesianData.zAcceleration := #P^.AxisData.a2Acceleration * #GearRatioA2;
        //Link constellation can be set to 0 here, hence it is not needed.
        #P^.CartesianData.LinkConstellation := 16#0000;
```

SCL

```
        //Transformation was successfull.
        #FunctionResult := 0;

    //Calculate the backward transformation "CartesianData" -> "AxisData".
    //The system fills the "CartesianData" of "TransformationParameters" with values.
    //To calculate the "AxisData" evaluate "CartesianData".
    ELSIF #TransformationType = 1 THEN

        //Calculate the position, velocity and acceleration component for the first axis.
        #P^.AxisData.a1Position := #P^.CartesianData.xPosition / #GearRatioA1;
        #P^.AxisData.a1Velocity := #P^.CartesianData.xVelocity / #GearRatioA1;
        #P^.AxisData.a1Acceleration := #P^.CartesianData.xAcceleration / #GearRatioA1;

        //Calculate the position, velocity and acceleration component for the second axis.
        #P^.AxisData.a2Position := #P^.CartesianData.zPosition / #GearRatioA2;
        #P^.AxisData.a2Velocity := #P^.CartesianData.zVelocity / #GearRatioA2;
        #P^.AxisData.a2Acceleration := #P^.CartesianData.zAcceleration / #GearRatioA2;

        //Transformation was successfull.
        #FunctionResult := 0;

    END_IF;
END_IF;
```

See also

MC-Transformation [OB98] (Page 2244)

Tags of kinematics transformation

The following tags of the kinematics technology object are relevant for the kinematics transformation:

Tag	Description	
Status values		
<TO>.StatusKinematics.Valid	TRUE	Transformation/Cartesian values valid
	FALSE	Transformation/Cartesian values invalid
<TO>.StatusKinematics.LinkConstellation	Joint position	
<TO>.FlangeInKcs	Current transformation frame (with dynamics, setpoint reference)	

Kinematics motions

With kinematics motions you move the kinematics through the three-dimensional space. Plan the kinematics motion in advance. Take into consideration the following:

- Reachable points of the kinematics
- Zones
- Transformation areas

- Joint position spaces
- Software limit switches of axes

The orientation motion is the motion of the Cartesian orientation and is performed at the same time for the kinematics motion. When motions are smoothed, the orientation motion is also smoothed. When the kinematics motion stops, the orientation motion also stops.

Reference system

The target position and the target orientation you specify for a kinematics motion can relate to the world coordinate system (WCS) or an object coordinate system (OCS).

Motion types

Linear motion

You can move a kinematics system with a linear motion. You define the linear motion using the Motion Control instructions "MC_MoveLinearAbsolute" and "MC_MoveLinearRelative". While the kinematics system is moved to an absolute position with a "MC_MoveLinearAbsolute" job, it is moved relative to the current position with a "MC_MoveLinearRelative" job. The kinematics system moves from the current position to the defined target position with a linear motion.

Circular motion

You can move a kinematics system with a circular motion. You define the circular motion using the Motion Control instructions "MC_MoveCircularAbsolute" and "MC_MoveCircularRelative". While the kinematics system is moved to an absolute position with a "MC_MoveCircularAbsolute" job, it is moved relative to the current position with a "MC_MoveCircularRelative" job.

Definition of circular path ("CircMode")

With the "CircMode" parameter, you specify the definition of the circular path. Depending on this parameter value, the circular path is calculated as follows:

- Via an intermediate point and the end point ("CircMode" = 0)
  With the intermediate point, you specify a point on the circular path, via which the end point is to be approached. The circular path is calculated from the starting, intermediate and end points. Only circular paths less than 360° can be traversed here.
- Via the circle center and the angle in a main plane ("CircMode" = 1)
  The end point of the circular path is calculated with the circle center and the angle. With the "PathChoice" parameter, you specify whether the circular path is to be traversed in positive or negative direction of rotation. With the "CirclePlane" parameter, you specify the main plane in which the circular path is to be traversed.
- Via the circle radius and the end point in a main plane ("CircMode" = 2)
  The circular path is calculated with the circle radius and the end point. This can yield up to four possible circular paths. With the "PathChoice" parameter, you specify which of the four possible circular paths is to be traversed. With the "CirclePlane" parameter, you specify the main plane in which the circular path is to be traversed.

When defining the intermediate and end points, circle center and angle or circle radius and end point, ensure that the information is consistent.

Orientation direction of the circular path ("PathChoice")

When the circular path is to be calculated using circle center and the angle, you use the "PathChoice" parameter to define whether the circular path is to be traversed in positive or negative direction of rotation.

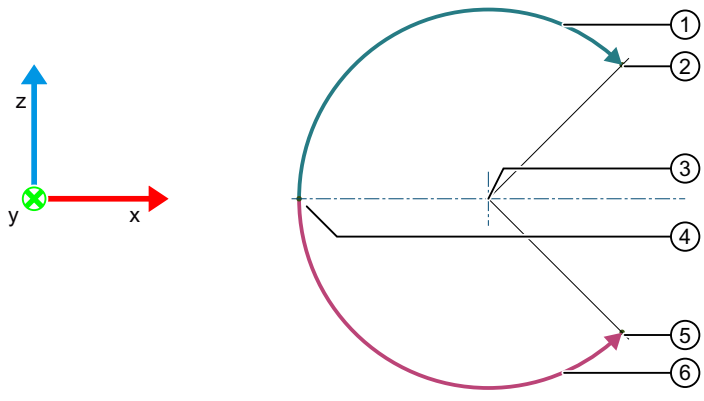

① Positive direction of rotation ("PathChoice" = 0)
② End point
③ Circle center
④ Starting point
⑤ End point
⑥ Positive direction of rotation ("PathChoice" = 1)

When the circular path is to be calculated using the circle radius and the end point, you use the "PathChoice" parameter to define which of the four possible circular paths is to be

traversed. A distinction is made here between the positive and negative direction of rotation as well as between the longer and shorter circle segment.

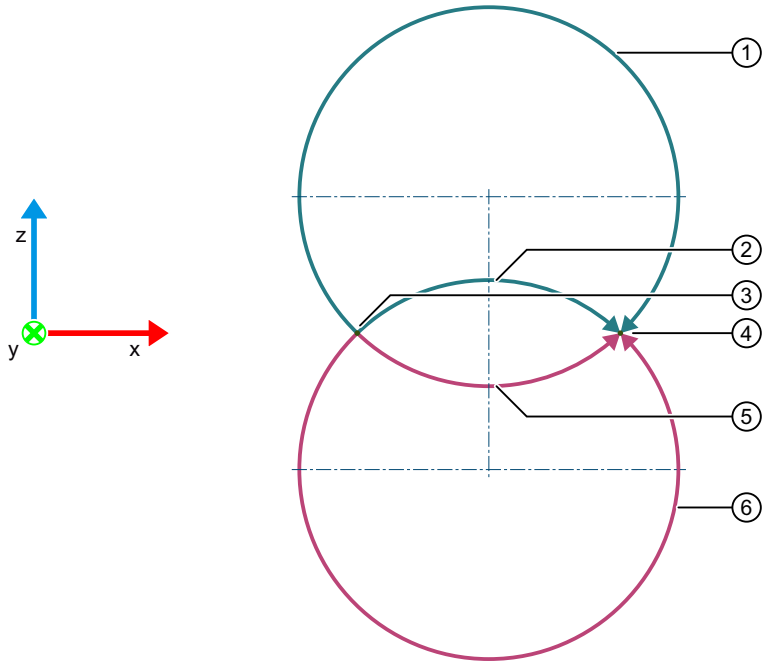

① Longer positive circle segment ("PathChoice" = 2)
② Shorter positive circle segment ("PathChoice" = 0)
③ Starting point
④ End point
⑤ Shorter negative circle segment ("PathChoice" = 1)
⑥ Longer negative circle segment ("PathChoice" = 3)

Motion dynamics

Dynamics of kinematics motion and orientation motion

You specify the dynamic values (velocity, acceleration, jerk) of a kinematics motion for the corresponding Motion Control instruction.

Dynamic defaults

If you do not specify any dynamic values for motion jobs (default value "-1.0"), the dynamic defaults configured under "Technology object > Configuration > Extended parameters > Dynamics" are used for the kinematics motion. For the orientation motion you can only specify the dynamic values using the dynamic defaults.

If you change the dynamic defaults during an active motion, the changed values take effect only in the next motion job.

Dynamic limits of the kinematics

The configured dynamic limits of the kinematics configured under "Technology object > Configuration > Extended parameters > Dynamics" are taken into account during the motion execution. The dynamics of a motion can be restricted so that the dynamic limits of the kinematics are not exceeded. When you change the dynamic limits of the kinematics, the changes take effect immediately for the kinematics motion and orientation motion.

Dynamic limits of the axes

When a motion job is transmitted, the dynamic limits of the axes configured under "Technology object > Configuration > Extended parameters > Limits > Dynamic limits" are only taken into account if the Dynamic adaptation is active. The dynamics of the motion can be restricted so that the dynamic limits of the axes are not exceeded. When you change the dynamic limits of the axes during an active motion, the changed values only take effect with the next motion job.

Dynamic adaptation

You set the dynamic adaptation under "Technology object > Configuration > Extended parameters > Dynamics". When dynamic adaptation is active, a velocity profile is calculated for the entire motion which takes into account the dynamic limits of the axes and the kinematics (<TO>.StatusPath.DynamicAdaption). Velocity and acceleration are included in the dynamic adaptation. Tangential and radial acceleration of the path is taken into account in acceleration. The jerk is not limited for dynamic adaptation.

The graphic below shows examples of a velocity curve with and without dynamic adaptation:

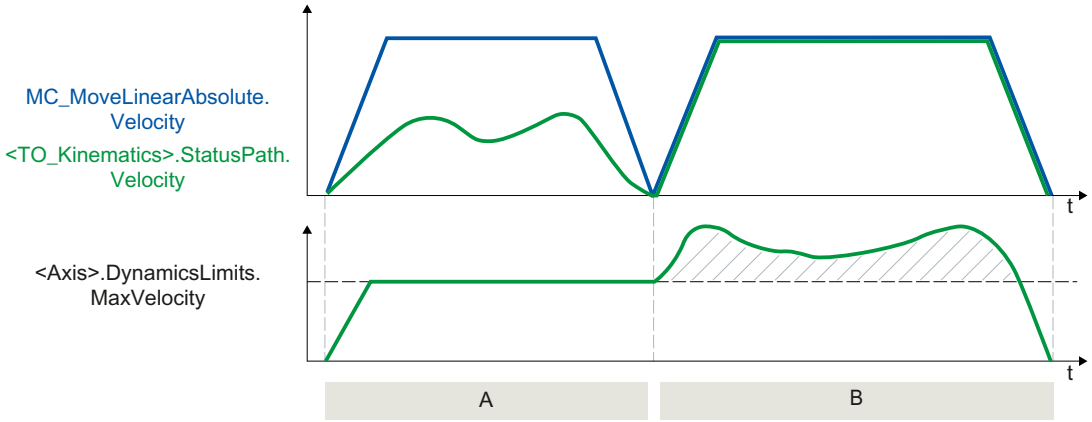

A	With dynamic adaptation
B	Without dynamic adaption

With dynamic adaptation without segmentation of the path, the velocity profile is calculated taking into consideration the dynamic limits of the axes which apply for the entire motion.

For dynamic adaptation with segmentation, the path is divided into individual equidistant segments. For each of these segments, the velocity profile is calculated taking into consideration the dynamic limits of the axes which apply for individual sections of the motion. The dynamic response is therefore adapted for individual sections of a motion.

Override

You can specify a velocity override for the kinematics (<TO>.Override.Velocity) using the technology object data block. You can specify a value between 0% and 200%. The velocity override acts on the velocity of the tool center point (TCP) along the path. If you change the velocity override of the kinematics, the change takes effect immediately for the kinematics motion and orientation motion.

The setpoint velocity of the motion is the velocity specified for the Motion Control instruction multiplied by the percentage value of the velocity override.

The axis-specific velocity override values do not take effect for kinematics motions.

Tags of motion control and dynamics

The following technology object tags are relevant for motion control:

Tag	Description	
Status values		
<TO>.StatusWord	Status indicator for an active motion	
<TO>.StatusPath.CoordSystem	Coordinate system of the active motion job	
	0	World coordinate system
	1, 2, 3	Object coordinate system 1, 2, 3
<TO>.Tcp	Target coordinates of the kinematics motion in the world coordinate system x, y, z, A	
<TO>.StatusPath.Velocity	Current path velocity (setpoint reference)	
<TO>.StatusPath.Acceleration	Current path acceleration (setpoint reference)	
<TO>.StatusPath.DynamicAdaption	Dynamic adaptation	
	0	No dynamic adaptation
	1	Dynamic adaptation with segmentation of the path
	2	Dynamic adaptation without segmentation of the path
<TO>.StatusMotionQueue.NumberOfCommands	Number of jobs in the job sequence	
Override		
<TO>.Override.Velocity	Velocity override	
Dynamic limits		
<TO>.DynamicLimits.Path.Velocity	Dynamic limitation for the maximum velocity of the path	
<TO>.DynamicLimits.Path.Acceleration	Dynamic limitation for the maximum acceleration of the path	
<TO>.DynamicLimits.Path.Deceleration	Dynamic limitation for the maximum deceleration of the path	
<TO>.DynamicLimits.Path.Jerk	Dynamic limitation for the maximum jerk of the path	
<TO>.DynamicLimits.Orientation.Velocity	Dynamic limitation for the maximum velocity of the Cartesian orientation	
<TO>.DynamicLimits.Orientation.Acceleration	Dynamic limitation for the maximum acceleration of the Cartesian orientation	
<TO>.DynamicLimits.Orientation.Deceleration	Dynamic limitation for the maximum deceleration of the Cartesian orientation	
<TO>.DynamicLimits.Orientation.Jerk	Dynamic limitation for the maximum jerk of the Cartesian orientation	
Dynamic defaults		
<TO>.DynamicDefaults.Path.Velocity	Default setting of the velocity of the path	

Tag	Description	
<TO>.DynamicDefaults.Path.Acceleration	Default setting of the acceleration of the path	
<TO>.DynamicDefaults.Path.Deceleration	Default setting of the deceleration of the path	
<TO>.DynamicDefaults.Path.Jerk	Default setting of the jerk of the path	
<TO>.DynamicDefaults.Orientation.Velocity	Default setting of the velocity of the Cartesian orientation	
<TO>.DynamicDefaults.Orientation.Acceleration	Default setting of the acceleration of the Cartesian orientation	
<TO>.DynamicDefaults.Orientation.Deceleration	Default setting of the deceleration of the Cartesian orientation	
<TO>.DynamicDefaults.Orientation.Jerk	Default setting of the jerk of the Cartesian orientation	
<TO>.DynamicDefaults.DynamicAdaption	Default setting of the dynamic adaptation	
	0	No dynamic adaptation
	1	Dynamic adaptation with segmentation of the path
	2	Dynamic adaptation without segmentation of the path

Zone monitoring

The zone monitoring has the following purposes:

- Protection from collisions with mechanical installations
- Triggering of process-related actions (signal zones)

WARNING

No protection of personnel

The zone monitoring is not suitable for protection of personnel.

Implement suitable protective measures for protecting personnel, e.g. set up protective fences, install safety doors, etc.

Zones are geometric bodies you can use to describe and subdivide the workspace of a kinematics system. You can configure workspace zones and kinematics zones on the kinematics technology object. Workspace zones describe the environment of a kinematics system. Kinematics zones envelope the end point of a kinematics system (flange or tool).

The following graphic shows the zones of a kinematics system:

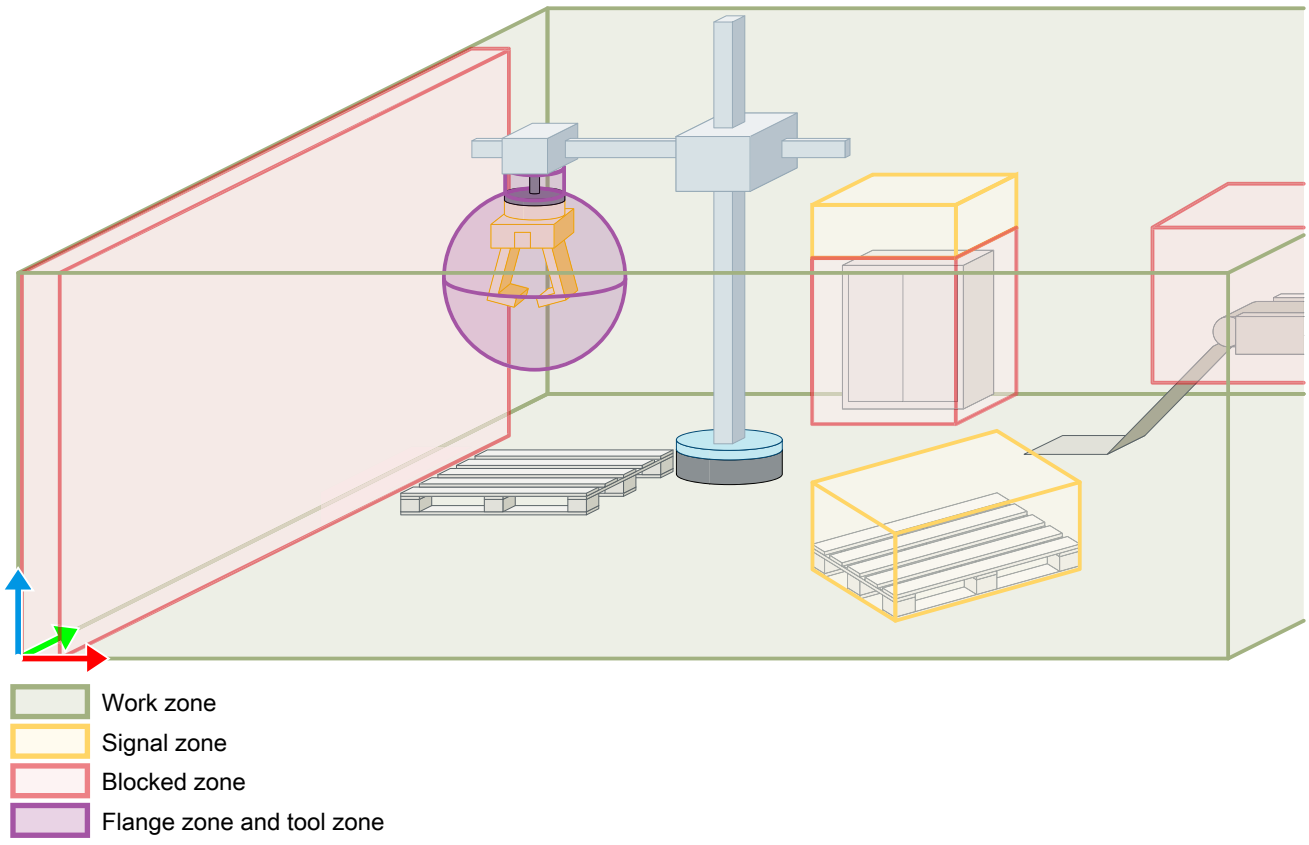

Zone configuration

You can specify and activate/deactivate zones via the configuration of the kinematics technology object or in your user program using Motion Control instructions.

Zone monitoring

The zone monitoring checks all activated workspace zones (work zones, signal zones, blocked zones) for collision with all activated kinematics zones (flange zones, tool zones). The zone monitoring monitors the zones for all motions of the kinematics system:

- Kinematics motions via the user program or kinematics control panel
- Single axis motions via the user program or axis control panel

The status of the zone monitoring is indicated in the diagnostics (Page 2307) and in the tags (Page 2262) of the kinematics technology object.

If the zone monitoring detects a zone violation by a kinematics motion, the following reactions occur:

Zone violation	Reaction	Description
Exiting the work zone	Alarm with stop	The kinematics technology object outputs a technology alarm. The kinematics motion is stopped.
Entering an signal zone	Alarm without stop	The kinematics technology object outputs a technology alarm. The kinematics motion will be continued.
Entering an blocked zone	Alarm with stop	The kinematics technology object outputs a technology alarm. The kinematics motion is stopped. The kinematics violates the zone by the length of the brake path at a minimum.

The kinematics technology object outputs a technology alarm following zone violations by single axis motions. The positioning axis/synchronous axis technology object outputs a technology alarm. The single axis motion is not aborted. You can abort the single axis motion in the application.

In addition to the zones of the kinematics technology object, you limit the travel space of the kinematics by the software limit switches of the axes.

Retracting after a zone violation

Once you have acknowledged the technology alarm on the kinematics technology object, you can move the kinematics system again.

NOTICE
Zone monitoring for violated zone deactivated after acknowledgment After you have acknowledged the technology alarm on the kinematics technology object, the zone monitoring is deactivated for the violated zone until the kinematics exits the violated blocked zone / signal zone or enters the violated work zone again. You can move the kinematics in all directions including again into the violated blocked zone / signal zone or from the work zone. Take into consideration the travel direction when retracting the kinematics.

Monitor retraction in the application. The zone monitoring status is still displayed in the data block technology object.

After the kinematics has exited the violated blocked zone / signal zone again or entered the violated work zone again, zone monitoring is activated again for this zone. A new technology alarm is therefore triggered when the zone is violated again.

See also

Tags for zone monitoring (Page 2262)

Workspace zones

Workspace zones describe the environment of a kinematics system. You define workspace zones in the world coordinate system (WCS) or in the object coordinate system (OCS). You can configure and activate/deactivate up to ten workspace zones. The following table shows the workspace zones of the kinematics technology object:

Workspace zone	Description
Work zone	Work zones define areas in which kinematics zones may move.
Signal zone	Signal zones indicate the following: • Kinematics zone is entering the signal zones • Kinematics zone is located in the signal zone
Blocked zone	Blocked zones define areas in which a kinematics zone must not enter.

Work zone

With work zones you limit the possible travel space of the kinematics or define several possible work areas. You can specify several work zones. Only one work zone can be activated at a given time. If no work zone is activated, the entire traversing space of the kinematics is regarded as the work area.

Kinematics zones must be located within work zones. When a kinematics zone exits a work zone, the kinematics technology object outputs the technology alarm 806 (alarm reaction: Stop with maximum dynamic values of the kinematics). The axes involved in the kinematics motion stop with the maximum dynamic values configured for the kinematics technology object. All jobs in the job sequence are canceled.

Signal zone

Signal zones are areas within the traversing space of the kinematics. Signal zones indicate a zone violation by a kinematics zone but do not trigger a stop of the kinematics motion. Signal zones can be located outside the work zone to some extent.

When a kinematics zone violates an signal zone, the kinematics technology object outputs the technology alarm 807 (no alarm reaction).

Blocked zone

Blocked zones are areas within the traversing space of the kinematics in which a kinematics zone must not enter (e.g. control cabinet, protective wall). Blocked zones can be located outside the work zone to some extent.

When a kinematics zone violates an blocked zone, the kinematics technology object outputs the technology alarm 806 (alarm reaction: Stop with maximum dynamic values of the kinematics). The axes involved in the kinematics motion stop with the maximum dynamic values configured for the kinematics technology object. All jobs in the job sequence are canceled.

Kinematics zones

Kinematics zones are related to the working point / flange of a kinematics and move with the kinematics. The zone monitoring checks the kinematics zones for penetration with workspace zones. With kinematics zones you expand the monitored area beyond the tool center point (TCP). You can configure and activate/deactivate up to nine kinematics zones. The following table shows the kinematics zones of the kinematics technology object:

Kinematics zone	Reference system	Description
Tool zone	TCS	Tool zones envelope the tool or parts of the tool.
Flange zone	FCS	Flange zones envelope the flange or parts of the flange.

Tool zone

You define tool zones in the tool coordinate system (TCS). The following graphic shows a spherical tool zone:

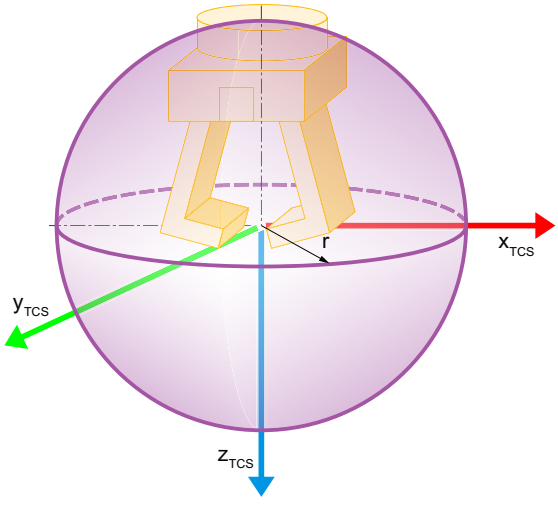

Flange zone

You define flange zones in the flange coordinate system (FCS). The following graphic shows a cylindrical flange zone:

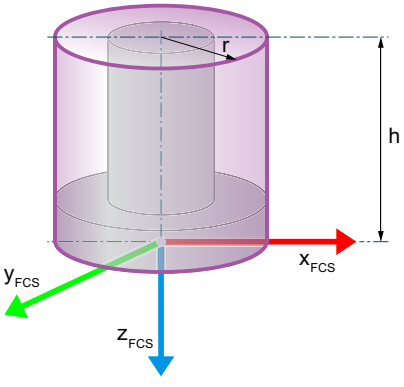

In this example, a shift by the height of the flange zone in negative z direction of the FCS has been defined.

Zone geometry

You can configure zones with the following geometric bodies:

- Sphere
- Cuboid
- Cylinder

You specify the position of the zero point of the zone coordinate system in the reference coordinate system. You specify the dimensions and rotation of the body starting from this zero point.

Sphere

You define a sphere starting from the zero point using the radius:

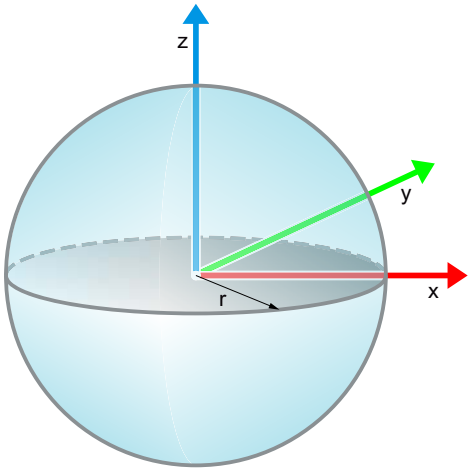

r Radius of the sphere

Cuboid

You define a cuboid starting from the zero point using the edge lengths in x, y and z direction:

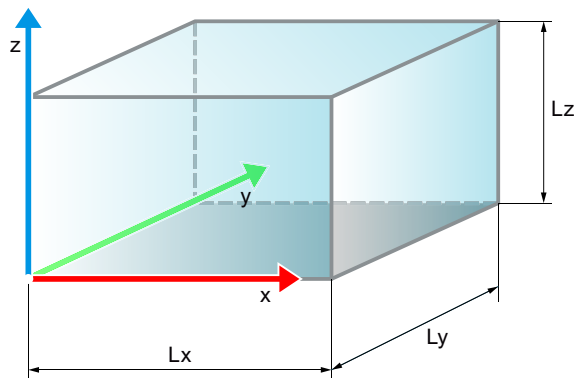

Lx	Edge length in x direction of the zone coordinate system
Ly	Edge length in y direction of the zone coordinate system
Lz	Edge length in z direction of the zone coordinate system

Cylinder

You define a cylinder starting from the zero point using the radius of the base and the cylinder height:

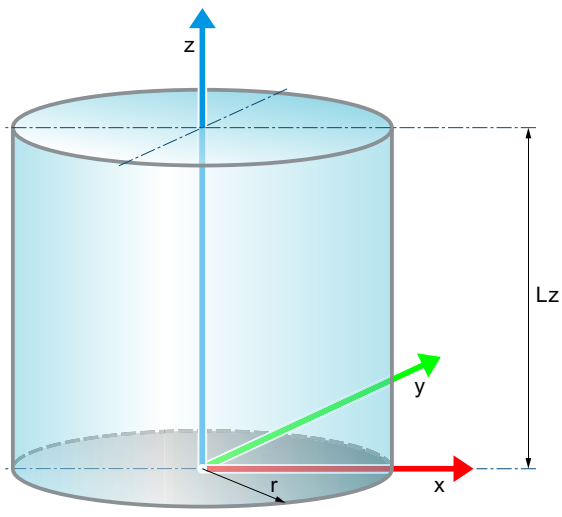

r	Radius of the base of the cylinder
Lz	Height of the cylinder in z direction of the zone coordinate system

Tags for zone monitoring

The following tags of the kinematics technology object are relevant for the zone monitoring:

Tag	Description
Zone configuration	
<TO>.WorkspaceZone[1..10]	Configuration of the workspace zones
<TO>.KinematicsZone[2..10]	Configuration of the kinematics zones The <TO>.KinematicsZone[1] zone is the tool center point (TCP) and is always activated.
Status values	
<TO>.StatusWorkspaceZone[1..10]	Status of the workspace zones
<TO>.StatusKinematicsZone[2..10]	Status of the kinematics zones
<TO>.StatusZoneMonitoring.WorkingZones	Display of violated work zones The bit numbers 1 to 10 correspond to the configured zone numbers.
<TO>.StatusZoneMonitoring.BlockedZones	Display of violated blocked zones The bit numbers 1 to 10 correspond to the configured zone numbers.
<TO>.StatusZoneMonitoring.SignalizingZones	Display of approached signal zones The bit numbers 1 to 10 correspond to the configured zone numbers.
<TO>.StatusZoneMonitoring.KinematicsZones	Display of kinematics zones that violate workspace zones The bit number 1 indicates the monitoring status of the TCP. The bit numbers 2 to 10 correspond to the configured zone numbers.

6.1.3.3 Version overview

For S7-1500T Motion Control, a distinction is made between the version of the technology, the technology objects and the Motion Control instructions.

The version of a technology object or a Motion Control instruction is indicated in the properties of the technology object in the "General > Information" tab, "Version" field.

Compatibility list

The table below shows the compatibility of the technology version with the CPU version:

CPU	Technology	Technology object	Motion Control instruction
V2.5	V4.0	Kinematics V4.0	MC_GroupInterrupt V4.0 MC_GroupContinue V4.0 MC_GroupStop V4.0 MC_MoveLinearAbsolute V4.0 MC_MoveLinearRelative V4.0 MC_MoveCircularAbsolute V4.0 MC_MoveCircularAbsolute V4.0 MC_DefineWorkspaceZone V4.0 MC_DefineKinematicsZone V4.0 MC_SetWorkspaceZoneActive V4.0 MC_SetWorkspaceZoneInactive V4.0 MC_SetKinematicsZoneActive V4.0 MC_SetKinematicsZoneInactive V4.0 MC_DefineTool V4.0 MC_SetTool V4.0 MC_SetOcsFrame V4.0

6.1.3.4 Configuring

Adding a kinematics technology object

The following describes how to add a kinematics technology object in the project tree.

Requirement

A project with a CPU S7-1500T is created.

Procedure

To add a kinematics technology object, follow these steps:

1. Open the CPU's folder in the project navigator.
2. Open the "Technology Objects" folder.
3. Double-click "Add new object".
   The "Add new object" dialog opens.
4. Select "TO_Kinematics". You can infer the function of the technology object from the displayed description.
5. In the "Name" input field, adapt the name of the kinematics to your requirements.
6. To change the suggested data block number, select the "Manual" option.
7. To add your own information about the technology object, click "Additional information".

8. To open the configuration after adding the technology object, select the "Add new and open" check box.
9. To add the technology object, click "OK".

Result

The new kinematics technology object was created and placed in the "Technology objects" folder of the project tree.

Configuring the kinematics technology object

Configuration - Basic Parameters

Configure the basic properties of the kinematics technology object in the "Basic Parameters" configuration window.

Kinematics name

Define the name of the kinematics in this field. The technology object is listed under this name in the project tree. The variables of the technology object can be used in the user program under this name.

Kinematics type

Select the desired kinematics type (Page 2160) in this drop-down list.

Units of measure

In the drop-down list, select the desired units of measure (Page 2152) for the position, velocity, angle and angular velocity of the kinematics.

Configuration - Interconnections

Configure the axes of kinematics in the "Interconnections" configuration window.

Kinematics axes

You can interconnect a kinematics technology object with positioning axes and synchronous axes which are already created in the project. In the drop-down list, select the desired axes depending on the kinematics type (Page 2160). You can directly call the configuration of the selected technology object using the button. Configure the interconnected technology objects as linear or rotary axes according to the kinematics type.

Depending on the kinematics type, the following kinematics axes are relevant:

Kinematics type	Kinematics axis A1	Kinematics axis A2	Kinematics axis A3	Orientation axis A4
2D	x	x	-	-
2D with orientation	x	x	-	x
3D	x	x	x	-
3D with orientation	x	x	x	x

x relevant

- Not relevant

See also

Interconnection rules (Page 2151)

Configuration - Geometry

Configuration - Geometry (Cartesian portal)

Configure the geometric parameters of kinematics in the "Geometry" configuration window.

Transformation parameters

In these fields, define the transformation parameters of the kinematics in the kinematics coordinate system (KCS) depending on the kinematics type:

- Kinematics type "Cartesian portal 2D" and "Cartesian portal 2D with orientation"

Field	Description
Length L1	Define the distance of the zero position of the A1 axis to the kinematics zero point (KZP) in x direction of the KCS
Length L2	Define the distance of the zero position of the A2 axis to the kinematics zero point in z direction of the KCS
Flange length LF	Define the distance of the flange coordinate system (FCS) from the axis A2 in the negative z direction of the KCS in this field.

- Kinematics type "Cartesian portal 3D" and "Cartesian portal 3D with orientation"

Field	Description
Length L1	Define the distance of the zero position of the A1 axis to the kinematics zero point in x direction of the KCS in this field.
Length L2	Define the distance of the zero position of the A2 axis to the kinematics zero point in y direction of the KCS in this field.
Length L3	Define the distance of the zero position of the A3 axis to the kinematics zero point in z direction of the KCS in this field.
Flange length LF	Define the distance of the flange coordinate system from the axis A3 in the negative z direction of the KCS in this field.

Representation in kinematics trace

In these fields, define the scaling in which the kinematics is displayed in the kinematics trace, depending on the kinematics type:

- Kinematics type "Cartesian portal 2D" and "Cartesian portal 2D with orientation"

Field	Description
x minimum	Define the dimensioning of the kinematics in the negative x direction in this field.
x maximum	Define the dimensioning of the kinematics in the positive x direction in this field.
z minimum	Define the dimensioning of the kinematics in the negative z direction in this field.
z maximum	Define the dimensioning of the kinematics in the positive z direction in this field.

- Kinematics type "Cartesian portal 3D" and "Cartesian portal 3D with orientation"

Field	Description
x minimum	Define the dimensioning of the kinematics in the negative x direction in this field.
x maximum	Define the dimensioning of the kinematics in the positive x direction in this field.
y minimum	Define the dimensioning of the kinematics in the negative y direction in this field.
y maximum	Define the dimensioning of the kinematics in the positive y direction in this field.
z minimum	Define the dimensioning of the kinematics in the negative z direction in this field.
z maximum	Define the dimensioning of the kinematics in the positive z direction in this field.

See also

Cartesian portal (Page 2161)

Configuration - Geometry (roller picker)

Configure the geometric parameters of kinematics in the "Geometry" configuration window.

Transformation parameters

In these fields, define the transformation parameters of the kinematics in the kinematics coordinate system (KCS) depending on the kinematics type:

- Kinematics type "Roller picker 2D" and "Roller picker 2D with orientation"

Field	Description
Radius R1	Define the cam radius for the axis A1 in this field.
Radius R2	Define the cam radius for the axis A2 in this field.
Length L1	Define the distance of the flange coordinate system (FCS) to the kinematics zero point (KZP) in x direction of the KCS on zero position of the axes A1 and A2 in this field.
Length L2	Define the distance of the FCS to the kinematics zero point including the flange length LF in z direction of the KCS on zero position of the axes A1 and A2 in this field.
Flange length LF	Define the flange length before the FCS in the negative z direction of the KCS in this field.

- Kinematics type "Roller picker 3D (vertical)", and "Roller picker 3D with orientation (vertical)" and "Roller picker 3D with orientation (horizontal)"

Field	Description
Radius R1	Define the cam radius for the axis A1 in this field.
Radius R2	Define the cam radius for the axis A2 in this field.
Length L1	Define the distance of the FCS to the kinematics zero point in x direction of the KCS in this field.
Length L2	For kinematics type "Roller picker 3D (vertical)" and "Roller picker 3D with orientation (vertical)": Define the distance of the A3 axis to the kinematics zero point in y direction of the KCS in this field.
	For kinematics type "Roller picker 3D with orientation (horizontal)": Define the distance of the FCS to the kinematics zero point in y direction of the KCS in this field.
Length L3	For kinematics type "Roller picker 3D (vertical)" and "Roller picker 3D with orientation (vertical)": Define the distance of the FCS to the kinematics zero point in z direction of the KCS in this field.
	For kinematics type "Roller picker 3D with orientation (horizontal)": Define the distance of the A3 axis to the kinematics zero point in z direction of the KCS in this field.
Flange length LF	Define the flange length before the FCS in the negative z direction of the KCS in this field.

Representation in kinematics trace

In these fields, define the scaling in which the kinematics is displayed in the kinematics trace, depending on the kinematics type:

- Kinematics type "Roller picker 2D" and "Roller picker 2D with orientation"

Field	Description
x minimum	Define the dimensioning of the kinematics in the negative x direction in this field.
x maximum	Define the dimensioning of the kinematics in the positive x direction in this field.
z minimum	Define the dimensioning of the kinematics in the negative z direction in this field.
z maximum	Define the dimensioning of the kinematics in the positive z direction in this field.

- Kinematics type "Roller picker 3D (vertical)", and "Roller picker 3D with orientation (vertical)" and "Roller picker 3D with orientation (horizontal)"

Field	Description
x minimum	Define the dimensioning of the kinematics in the negative x direction in this field.
x maximum	Define the dimensioning of the kinematics in the positive x direction in this field.
y minimum	Define the dimensioning of the kinematics in the negative y direction in this field.
y maximum	Define the dimensioning of the kinematics in the positive y direction in this field.
z minimum	Define the dimensioning of the kinematics in the negative z direction in this field.
z maximum	Define the dimensioning of the kinematics in the positive z direction in this field.

See also

Roller picker (Page 2172)

Configuration - Geometry (SCARA)

Configure the geometric parameters of kinematics in the "Geometry" configuration window.

Transformation parameters

In these fields, define the transformation parameters of the kinematics in the kinematics coordinate system (KCS):

Field	Description
Length L1	Define the distance of the A1 axis to the kinematics zero point (KZP) in z direction of the KCS
Length L2	Define the distance of the A1 axis to the A2 axis in x direction of the KCS in this field.
Length L3	Define the distance of the A2 axis to the A3 axis in x direction of the KCS in this field.
Flange length LF	Define the distance of the flange coordinate system (FCS) from the axis A3 in the negative z direction of the KCS in this field.

Mechanical axis coupling

The kinematics transformation compensates for the configured mechanical axis couplings. If you set a mechanical axis coupling for a first axis to a second axis, the setting "Modulo" must not be activated for the first axis.

You can configure the following mechanical coupled axes for the kinematics:

- Mechanical coupling of axis A1 to axis A2
- Mechanical coupling of axis A4 to axis A3

Enter the desired coupling factor in the "Compensation factor" fields.

Representation in kinematics trace

In these fields, define the scaling in which the kinematics is displayed in the kinematics trace:

Field	Description
z minimum	Define the dimensioning of the kinematics in the negative z direction in this field.
z maximum	Define the dimensioning of the kinematics in the positive z direction in this field.

See also

SCARA (Page 2186)

Configuration - Geometry (articulated arm)

Configure the geometric parameters of kinematics in the "Geometry" configuration window.

Transformation parameters

In these fields, define the transformation parameters of the kinematics in the kinematics coordinate system (KCS) depending on the kinematics type:

- Kinematics type "Articulated arm 2D" and "Articulated arm 2D with orientation"

Field	Description
Length L1	Define the distance of the A1 axis from the kinematics zero point (KZP) in z direction of the KCS
Length L2	Define the distance of the A1 axis from the kinematics zero point in x direction of the KCS
Length L3	Define the distance of the A2 axis from the A1 axis in this field.
Length L4	Define the distance of the forced coupler point from the A2 axis in this field.
Flange length LF	Define the distance of the flange coordinate system (FCS) from the forced coupler point in the negative z direction of the KCS in this field.

- Kinematics type "Articulated arm 3D" and "Articulated arm 3D with orientation"

Field	Description
Length L1	Define the distance of the A2 axis from the kinematics zero point in z direction of the KCS in this field.
Length L2	Define the distance of the A2 axis from the kinematics zero point in x direction of the KCS
Length L3	Define the distance of the A3 axis from the A2 axis in this field.
Length L4	Define the distance of the forced coupler point from the A3 axis in this field.
Flange length LF	Define the distance of the FCS from the forced coupler point in the negative z direction of the KCS in this field.

Mechanical axis coupling

The kinematics transformation compensates for the configured mechanical axis couplings. If you set a mechanical axis coupling for a first axis to a second axis, the setting "Modulo" must not be activated for the first axis.

You can configure the following mechanical coupled axes, depending on the kinematics type:

- Kinematics type "Articulated arm 2D" and "Articulated arm 2D with orientation": Mechanical coupling of axis A1 to axis A2
- Kinematics type "Articulated arm 3D" and "Articulated arm 3D with orientation": Mechanical coupling of axis A2 to axis A3

Enter the required coupling factor in the "Compensation factor" field.

See also

Articulated arm (Page 2191)

Configuration - Geometry (delta picker)

Configure the geometric parameters of kinematics in the "Geometry" configuration window.

Transformation parameters

In these fields, define the transformation parameters of the kinematics in the kinematics coordinate system (KCS) depending on the kinematics type:

- Kinematics type "Delta picker 2D" and "Delta picker 2D with orientation"

Field	Description
Length L1	Define the length of the upper arms in this field.
Length L2	Define the length of the connecting rods in this field.
Flange length LF	Define the distance of the flange coordinate system (FCS) from the lower connecting plate in the negative z direction of the KCS in this field.
Distance D1	Define the distance of the axes to the middle of the upper connecting plate (radius of the upper connecting plate) in this field.
Distance D2	In this field, you can define the distance of the hinge points of the connecting rods to the lower connecting plate (radius of the lower connecting plate) in this field.

- Kinematics type "Delta picker 3D" and "Delta picker 3D with orientation"

Field	Description
Length L1	Define the length of the upper arms in this field.
Length L2	Define the length of the connecting rods in this field.
Flange length LF	Define the distance of the FCS from the lower connecting plate in the negative z direction of the KCS in this field.
Distance D1	Define the distance of the axes to the middle of the upper connecting plate (radius of the upper connecting plate) in this field.
Distance D2	In this field, you can define the distance of the hinge points of the connecting rods to the lower connecting plate (radius of the lower connecting plate) in this field.
Angle A1 to A2	Define the angle between the axes A1 and A2 in this field.
Angle A2 to A3	Define the angle between the axes A2 and A3 in this field.

See also

Delta picker (Page 2207)

Configuration - Geometry (cylindrical robot)

Configure the geometric parameters of kinematics in the "Geometry" configuration window.

Transformation parameters

In these fields, define the transformation parameters of the kinematics in the kinematics coordinate system (KCS):

Field	Description
Length L1	Define the distance of the zero position of the A2 axis from the kinematics zero point (KZP) in z direction of the KCS
Length L2	Define the distance of the A3 axis from the kinematics zero point in y direction of the KCS
Flange length LF	Define the distance of the flange coordinate system (FCS) from the axis A3 in the negative z direction of the KCS in this field.

Mechanical axis coupling

The kinematics transformation compensates for the configured mechanical axis couplings. If you set a mechanical axis coupling for a first axis to a second axis, the setting "Modulo" must not be activated for the first axis.

You can configure the following mechanical axis coupling for the kinematics type "Cylindrical robot 3D with orientation":

- Mechanical coupling of axis A4 to axis A2

Enter the required coupling factor in the "Compensation factor" field.

Representation in kinematics trace

In these fields, define the scaling in which the kinematics is displayed in the kinematics trace:

Field	Description
z minimum	Define the dimensioning of the kinematics in the negative z direction in this field.
z maximum	Define the dimensioning of the kinematics in the positive z direction in this field.
A3 maximum	Define the maximum traversing length of the A3 axis in this field.

See also

Cylindrical robot (Page 2220)

Configuration - Geometry (tripod)

Configure the geometric parameters of kinematics in the "Geometry" configuration window.

Transformation parameters

In these fields, define the transformation parameters of the kinematics in the kinematics coordinate system (KCS):

Field	Description
Length L1	Define the length of the connecting rods in this field.
Flange length LF	Define the distance of the flange coordinate system (FCS) from the lower connecting plate in the negative z direction of the KCS in this field.
Distance D1	Define the distance of the upper hinge points of the connecting rods to the center of the upper connecting plate in this field.
Distance D2	Define the distance of the lower hinge points of the connecting rods to the center of the lower connecting plate in this field.
Angle for A1 to xy plane of the KCS	Define the angle between the upper connecting plate (xy plane of the KCS) and the rail of the axis A1 ($0.0° \leq \gamma < 90.0°$) in this field.
Angle A1 to A2	Define the angle between the axes A1 and A2 in this field.
Angle A2 to A3	Define the angle between the axes A2 and A3 in this field.

See also

Tripod (Page 2229)

Configuration - Geometry (user-defined)

Configure the geometric parameters of kinematics in the "Geometry" configuration window.

Transformation parameters

In this table, define the start values of the parameters 1 to 32 of the kinematics (<TO>.Kinematics.Parameter[1..32]).

Representation in kinematics trace

In these fields, define the scaling in which the kinematics is displayed in the kinematics trace, depending on the kinematics type:

- Kinematics type "User-defined 2D" and "User-defined 2D with orientation"

Field	Description
x minimum	Define the dimensioning of the kinematics in the negative x direction in this field.
x maximum	Define the dimensioning of the kinematics in the positive x direction in this field.
z minimum	Define the dimensioning of the kinematics in the negative z direction in this field.
z maximum	Define the dimensioning of the kinematics in the positive z direction in this field.

- Kinematics type "User-defined 3D" and "User-defined 3D with orientation"

Field	Description
x minimum	Define the dimensioning of the kinematics in the negative x direction in this field.
x maximum	Define the dimensioning of the kinematics in the positive x direction in this field.
y minimum	Define the dimensioning of the kinematics in the negative y direction in this field.
y maximum	Define the dimensioning of the kinematics in the positive y direction in this field.
z minimum	Define the dimensioning of the kinematics in the negative z direction in this field.
z maximum	Define the dimensioning of the kinematics in the positive z direction in this field.

See also

User transformation (Page 2242)

User-defined kinematics systems (Page 2236)

Extended parameters

Configuration - Dynamics

In the "Dynamics" configuration window, configure the default values for the dynamics, the dynamic limits and the dynamic adaptation of the kinematics motion and the orientation motion.

Presets and limits

In order to define the default values for the kinematics motion, select the entry "Path" in the "Settings" drop-down list. In order to define the default values for the orientation motion, select the entry "Orientation motion" in the "Settings" drop-down list.

Define the default values of the dynamics in the "Velocity", "Acceleration", "Deceleration" and "Jerk" fields. Kinematics motion jobs triggered in the user program are executed with these default values if no separate dynamics values are defined for the jobs.

Define the default values of the dynamic limits in the "Maximum velocity", "Maximum acceleration", "Maximum deceleration" and "Maximum jerk" fields.

Dynamic adaptation

Select the default for the dynamics adaptation in the drop-down list. When dynamic adaptation is active, a velocity profile is calculated for the entire motion which takes into account the dynamic limits of the axes and the kinematics .

Mode	Description
Do not limit	The dynamic limits of the axes are not taken into consideration.
Limit with segmentation of the path	The path is divided into segments. For each of these segments, the dynamic is adapted in such a way that the dynamic limits of the axes are not exceeded.
Limit without segmentation of the path	The dynamic is adapted in such a way that the dynamic limits of the axes are not exceeded over the entire path.

See also

Dynamics of kinematics motion and orientation motion (Page 2251)

Configuration - Kinematics coordinate system

In the "Kinematics coordinate system" configuration window, configure the KCS frame (Page 2157) and therefore the position of the kinematics coordinate system (KCS) in the world coordinate system (WCS).

Kinematics zero point in the WCS

In these fields, define the position of the kinematics coordinate system:

Field	Description
x position	Define the shift of the KCS in x direction of the WCS in this field.
y position	Define the shift of the KCS in y direction of the WCS in this field.
z position	Define the shift of the KCS in z direction of the WCS in this field.
Rotation A	Define the rotation of the KCS around the z axis in this field.
Rotation B	Define the rotation of the KCS around the y axis in this field.
Rotation C	Define the rotation of the KCS around the x axis in this field.

See also

Coordinate systems and frames (Page 2154)

Configuration - Object coordinate systems

In the "Object coordinate system" configuration window, configure the OCS frames (Page 2157) and therefore the position of the object coordinate system (OCS) in the world coordinate system (WCS).

Object coordinate system (OCS)

Select the object coordinate system to be defined in this drop-down list. You can define up to three tool object coordinate systems.

OCS in the world coordinate system (WCS)

In these fields, define the position of the selected object coordinate system:

Field	Description
x position	Define the shift of the OCS in x direction of the WCS in this field.
y position	Define the shift of the OCS in y direction of the WCS in this field.
z position	Define the shift of the OCS in z direction of the WCS in this field.
Rotation A	Define the rotation of the OCS around the z axis in this field.
Rotation B	Define the rotation of the OCS around the y axis in this field.
Rotation C	Define the rotation of the OCS around the x axis in this field.

See also

Coordinate systems and frames (Page 2154)

Configuration- Tools

In the "Tools" configuration window, configure the tool frames (Page 2157) and also the position of the tool center point (TCP) of the tools in the flange coordinate system (FCS).

Tools

Select the tool to be defined in this drop-down list. You can define up to three tools.

Tool center point in the FCS

In these fields, define the position of the tool center point of the selected tool:

Field	Description
x position	Define the shift of the TCP in x direction of the FCS in this field.
y position	Define the shift of the TCP in y direction of the FCS in this field.
z position	Define the shift of the TCP in z direction of the FCS in this field.
Rotation A	Define the rotation of the TCP around the z axis in this field.

See also

Coordinate systems and frames (Page 2154)

Configuration - Zones

Configure the workspace zones and kinematics zones of the technology object in the "Zones" configuration window.

The configuration window is divided into the following areas:

- Graphic view
- Tabular editor
  - Workspace zones
  - Kinematics zones

Graphic view

The workspace zones or the kinematics zones, which you define in the corresponding tabular editor, are displayed in the graphic view. You can rotate the view and zoom in and out using the mouse.

The toolbar at the top of the graphical editor provides you with buttons for the following functions, depending on the respective tabular editor:

Button	Function	Description
	Fit to screen size	The view is displayed adapted to the size of the window.
	Show / hide grid	The grid lines of the coordinate system are hidden/displayed.
WCS	Select the coordinate system.	Select a coordinate system.
Tool 1	Select tool	Select a tool.
	Show 2D view	The 2D view is shown.
	Show 3D view	The 3D view is shown.
	Show xy plane	The xy plane is displayed.
	Show xy plane rotated	The xy plane is displayed rotated around the x-axis.
	Show xz plane	The xz plane is displayed.
	Show xz plane rotated	The xz plane is displayed rotated around the z-axis.
	Show xz plane rotated	The xz plane is displayed rotated around the x-axis.
	Show xz plane rotated	The xz plane is displayed rotated around the x-axis and the z-axis.
	Show yz plane	The yz plane is displayed.
	Show yz plane rotated	The yz plane is displayed rotated around the z-axis.
	Show yz plane rotated	The yz plane is displayed rotated around the y-axis.
	Show yz plane rotated	The yz plane is displayed rotated around the y-axis and the z-axis.

Workspace zones

Workspace zones (Page 2257) describe the environment of a kinematics system. You can configure up to ten workspace zones in the table.

Column	Description	
Visible	You show and hide the zone in the top view using the symbol in this column.	
Number	This column displays the zone number.	
Status	Select the activation status of the zone in this column.	
	Active	Zone monitoring is activated for the zone. You can deactivate the zone in your user program via an "MC_SetWorkspaceZoneInactive" job.
	Inactive	Zone monitoring is deactivated for the zone. You can activate the zone in your user program via an "MC_SetWorkspaceZoneActive" job.
	Invalid	The zone is not defined. You can define the zone in your user program via an "MC_DefineWorkspaceZone" job.
Zone type	Select the type of the zone in this column.	
	Work zone	Work zones define areas in which kinematics zones may move. You can specify several work zones. Only one work zone can be activated at a given time, however. If no work zone is activated, the entire traversing space of the kinematics is regarded as the work area.
	Blocked zone	Blocked zones define areas in which a kinematics zone must not enter.
	Signal zone	Signal zones are areas within the traversing space of the kinematics. Signal zones indicate a zone violation by a kinematics zone but do not trigger a stop of the kinematics motion.
Geometry	Select the geometry of the zone (Page 2259) in this column.	
		Sphere
		Cuboid
		Cylinder
Length x	With a cuboid-shaped zone, define the length of the zone in x direction in this column.	
Length y	With a cuboid-shaped zone, define the width of the zone in y direction in this column.	
Length z	With a cuboid-shaped zone, define the height of the zone in z direction in this column.	
	With a cylindrical zone, define the height of the zone in z direction in this column.	
Radius	With a spherical zone, define the radius of the zone in this column.	
	With a cylindrical zone, define the radius of the zone in this column.	

Column	Description	
CS	Select the reference coordinate system in this column.	
	WCS	World coordinate system
	OCS 1	Object coordinate system 1
	OCS 2	Object coordinate system 2
	OCS 3	Object coordinate system 3
x	Define the position of the zone in x direction in this column.	
y	Define the position of the zone in y direction in this column.	
z	Define the position of the zone in z direction in this column.	
A	Define the rotation of the zone around the z-axis in this column (not relevant for a spherical zone). Enter a value between -180.0° and 179.999°.	
B	Define the rotation of the zone around the y-axis in this column (not relevant for a spherical zone). Enter a value between -90.0° and 90.0°.	
C	Define the rotation of the zone around the x-axis in this column (not relevant for a spherical zone). Enter a value between -180.0° and 179.999°.	

Kinematics zones

Kinematics zones (Page 2258) are related to the working point / flange of a kinematics system and move with the kinematics. The zone monitoring checks the kinematics zones for penetration with workspace zones. You can configure up to ten kinematics areas in the table.

Column	Description	
Visible	You show and hide the zone in the top view using the symbol in this column.	
Number	This column displays the zone number.	
Status	Select the activation status of the zone in this column.	
	Active	Zone monitoring is activated for the zone. You can deactivate the zone in your user program via an "MC_SetKinematicsZoneInactive" job.
	Inactive	Zone monitoring is deactivated for the zone. You can activate the zone in your user program via an "MC_SetKinematicsZoneActive" job.
	Invalid	The zone is not defined. You can define the zone in your user program via an "MC_DefineKinematicsZone" job.
Zone type	Select the type of the zone in this column.	
	Flange zone	Flange zones envelope the flange or parts of the flange.
	Tool zone	Tool zones envelope the tool or parts of the tool.
Geometry	Select the geometry of the zone (Page 2259) in this column.	
		Sphere
		Cuboid
		Cylinder
Length x	With a cuboid-shaped zone, define the length of the zone in x direction in this column.	

Column	Description	
Length y	With a cuboid-shaped zone, define the width of the zone in y direction in this column.	
Length z	With a cuboid-shaped zone, define the height of the zone in z direction in this column.	
	With a cylindrical zone, define the height of the zone in z direction in this column.	
Radius	With a spherical zone, define the radius of the zone in this column.	
	With a cylindrical zone, define the radius of the zone in this column.	
CS	Select the reference coordinate system in this column.	
	FCS	Flange coordinate system (FCS)
	TCS	Tool coordinate system
x	Define the position of the zone in x direction in this column.	
y	Define the position of the zone in y direction in this column.	
z	Define the position of the zone in z direction in this column.	
A	Define the rotation of the zone around the z-axis in this column (not relevant for a spherical zone). Enter a value between -180.0° and 179.999°.	
B	Define the rotation of the zone around the y-axis in this column (not relevant for a spherical zone). Enter a value between -90.0° and 90.0°.	
C	Define the rotation of the zone around the x-axis in this column (not relevant for a spherical zone). Enter a value between -180.0° and 179.999°.	

Online view

When you click on the symbol, values of the workspace zones and the kinematics zones at the current point in time are read back from the CPU. You can display two additional rows for each zone in the table using the symbol in the "Number" column. These three lines contain the following values:

Line	Identifier in the "Number" column.	Description
1	Number of the zone	This line contains the current values of the zone.
2		This line contains the start values of the CPU.
3		This line contains the start values in the project. You can edit the values.

See also

Zone monitoring (Page 2254)

Copying a kinematics technology object

The following describes how to copy a kinematics technology object in the project tree.

Requirement

- A project with a CPU S7-1500T is created.
- A kinematics technology object is created in the project.

Procedure

To copy a kinematics technology object, follow these steps:

1. Open the CPU's folder in the project navigator.
2. Open the "Technology Objects" folder.
3. Select the kinematics technology object to be copied.
4. To also copy the interconnected axes, select them as well. To select multiple axes, press and hold down the <Ctrl> key.
5. Select "Copy" in the shortcut menu.
6. Select the "Technology objects" folder.
7. Select "Paste" in the shortcut menu.

Result

The selected kinematics technology object, together with any selected interconnected axes, are copied and created in the "Technology objects" folder of the project tree.

Deleting a kinematics technology object

The following describes how to delete a kinematics technology object in the project tree.

Requirement

- A project with a CPU S7-1500T is created.
- A kinematics technology object is created in the project.

Procedure

To delete a kinematics technology object, follow these steps:

1. Open the CPU's folder in the project navigator.
2. Open the "Technology Objects" folder.
3. Select the kinematics technology object to be deleted.
4. Select the "Delete" command in the shortcut menu.
   The "Confirm delete" dialog is opened.
5. To delete the technology object, click "Yes".

Result

The selected kinematics technology object has been deleted. The axes interconnected to the kinematics technology object are retained.

Toolbar of the configuration

The following functions are available in the toolbar of the function view:

Symbol	Function	Explanation
	Show online values	Displays the current values read back from the CPU.
	Couples the function view and parameter view for the objects selected in the navigation	Enables the targeted toggling between the parameter view and function-based view.
	Collapse/expand all nodes and objects	Collapses or expands all nodes and objects of the navigation or the data structure in the currently active view.
	Collapse/expand the nodes below the marked node	Collapses or expands the marked nodes and objects of the navigation or the data structure in the currently active view.

6.1.3.5 Programming

The "Programming" section contains general information on supplying and evaluating the Motion Control instructions.

You can find an overview of the Motion Control instructions for the kinematics technology object in the "Functions" section (Page 2149).

You can use Motion Control instructions in the user program to transmit jobs to the technology object. You define the job with the input parameters of the Motion Control instructions. The current job status is indicated in the output parameters.

Because the kinematics technology object groups the kinematics axes, you can assign the kinematics technology object directly to the input parameter "AxesGroup".

You cannot enable the kinematics technology object itself using an "MC_Power" command or home it using an "MC_Home" job. For kinematics motions, the interconnected axes must be enabled ("MC_Power.Enable" = TRUE) or referenced instead.

You can acknowledge errors of the kinematics technology object with an "MC_Reset" job or by restarting the technology object.

Job sequence

Motion-relevant jobs are entered in the job sequence of the kinematics technology object.

The following jobs enter the job sequence:

Job	Brief description
Kinematics motions	
"MC_MoveLinearAbsolute"	Position kinematics with linear path motion

Job	Brief description
"MC_MoveLinearRelative"	Relative positioning of kinematics with linear path motion
"MC_MoveCircularAbsolute"	Position kinematics with circular path motion
"MC_MoveCircularRelative"	Relative positioning of kinematics with circular path motion
Zones	
"MC_DefineWorkspaceZone"	Define workspace zone
"MC_DefineKinematicsZone"	Define kinematics zone
"MC_SetWorkspaceZoneActive"	Activate workspace zone
"MC_SetWorkspaceZoneInactive"	Deactivate workspace zone
"MC_SetKinematicsZoneActive"	Activate kinematics zone
"MC_SetKinematicsZoneInactive"	Deactivate kinematics zone
Coordinate systems	
"MC_SetOcsFrame"	Redefine object coordinate systems

The jobs are processed in the same order in which they were entered in the job sequence. The order of the jobs cannot be changed subsequently. If another motion job is added to the job sequence, all jobs in the job sequence will be recalculated. The kinematics motion jobs do not cancel each other. Because all jobs in the job sequence are taken into account for preparing the motion and calculation of the velocity profile, even jobs with short traversing lengths and blend motions with higher velocities can be traversed than what is possible for individual motions. The current job is also included in the new calculation so that the current job can be blended with the next job.

You can also interrupt execution of the jobs with an "MC_GroupInterrupt" job, fill the job sequence and then continue execution with an "MC_GroupContinue" job.

By default, the job sequence can contain up to five jobs. You can change the maximum number of jobs (Page 2315) using the parameter view. The job sequence can contain a maximum of ten jobs.

See also

"MotionQueue" tag (kinematics) (Page 2315)

Interrupting, continuing and stopping kinematics motions (Page 2284)

Motion status and remaining distance

You can obtain the status and the remaining distance of a motion job from the parameters of the corresponding Motion Control instruction.

Status of a motion job

You can identify the status of a motion job based on the "Busy" and "Active" parameters. When the job is transmitted, the "Busy" parameter is set to TRUE and the job is added to the job sequence. As long as the job is still in the job sequence, the "Active" parameter is set to FALSE. As soon as the job is active in the motion control, the "Active" parameter is set to TRUE. If the motion job is completed, the parameters "Busy" and "Active"are set to FALSE and the parameter "Done" to TRUE.

All inactive jobs in the job sequence are recalculated if another motion job is added to the job sequence. The current job is also included in the new calculation so that the current job can be blended with the next job. If a motion control is interrupted by a "MC_GroupInterrupt" job, the jobs in the job sequence are only calculated when the motion control is continued with an "MC_GroupContinue" job.

Remaining distance of a motion job

You can obtain the remaining distance of a motion job from the "RemainingDistance" parameter. If the motion is not being blended, the "RemainingDistance" parameter contains the distance to the target position on the path. If the active motion is being blended with the next motion, the "RemainingDistance" parameter contains the distance to the beginning of the blending segment on the path. If only the orientation axis is being moved in a motion job (reorientation), the "RemainingDistance" parameter contains the value "-1.0".

Interrupting, continuing and stopping kinematics motions

You can interrupt and continue active kinematics motions or stop them and thus also cancel queued motion jobs.

Interruption of kinematics motions

With the Motion Control instruction "MC_GroupInterrupt", you interrupt the execution of the motion for the kinematics technology object. With the "Mode" parameter, you specify the dynamic behavior. The kinematics can be stopped either with the dynamics of the motion job to be interrupted or with the maximum dynamics. The current path is not exited when the kinematics is stopped. If the kinematics is already at a standstill, the motion control is also interrupted for subsequent motion jobs.

The kinematics technology object is in "Interrupted" status (<TO>.StatusWord.X17).

For path planning, you can interrupt execution of the jobs, fill the job sequence (Page 2281) and then continue execution of the commands.

Continuation of kinematics motions

With the Motion Control instruction "MC_GroupContinue", you continue a kinematics motion that was previously interrupted with a "MC_GroupInterrupt" job. The kinematics motion can also be continued if the kinematics has not yet come to a standstill following the "MC_GroupInterrupt" job.

The "MC_GroupContinue" job only has an effect if the kinematics technology object is in "Interrupted" status (<TO>.StatusWord.X17).

Stopping of kinematics motions

With the Motion Control instruction "MC_GroupStop", you stop the motion control of the kinematics technology object. In so doing, both the active motion job as well as all queued jobs in the job sequence are canceled and the job sequence is emptied. If the kinematics motion was already interrupted with an "MC_GroupInterrupt" job, this job is also canceled. As long as the "Execute" parameter is set to TRUE, the following kinematics jobs are rejected ("ErrorID" = 16#80CD).

With the "Mode" parameter, you specify the dynamic behavior. The kinematics can be stopped either with the dynamics of the motion job to be stopped or with the maximum dynamics. The current path is not exited when the kinematics is stopped.

Motion preparation using multiple jobs

Connection of multiple kinematics motions with geometric transitions

Multiple motions can be appended to one another, in which case the kinematics comes to a standstill between the individual motions. To achieve an uninterrupted motion control, the individual motions can be blended with geometric transitions. You define the corresponding parameters at the new motion job (A2), into which the previous job (A1) is to be blended.

Transitions of linear motions

With the Motion Control instructions "MC_MoveLinearAbsolute" and "MC_MoveLinearRelative", you move a kinematics system with a linear motion. You define the motion transition mode with the "BufferMode" parameter and the rounding clearance with the "TransitionParameter[1]" parameter. The following table shows how these parameters act on the motion transition based on two linear motions:

Rounding clearance ("TransitionParameter[1]")	Motion transition ("BufferMode")	Description
Not relevant	"BufferMode" = 1 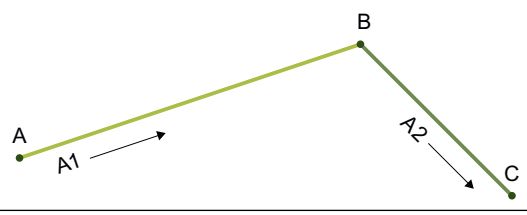 	Append motion The active linear motion is completed and the kinematics comes to a standstill. The next linear motion is then executed.
d > 0.0	"BufferMode" = 2, 5 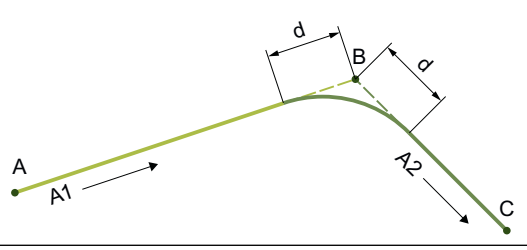 	Blend motion When the rounding clearance distance from the target position is reached, the active linear motion is blended with the next linear motion. Both motion jobs are blended at the lower velocity when "BufferMode" = 2 and at the higher velocity when "BufferMode" = 5.

Rounding clearance ("TransitionParameter[1]")	Motion transition ("BufferMode")	Description
d = 0.0	"BufferMode" = 2, 5 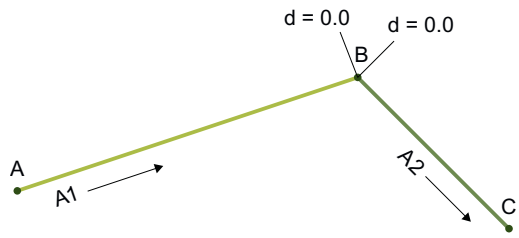 	Blend motion Because the rounding clearance is 0.0, traversing is the same as when "BufferMode" = 1. The active linear motion is completed and the kinematics comes to a standstill. The next linear motion is then executed.
d < 0.0	"BufferMode" = 2, 5 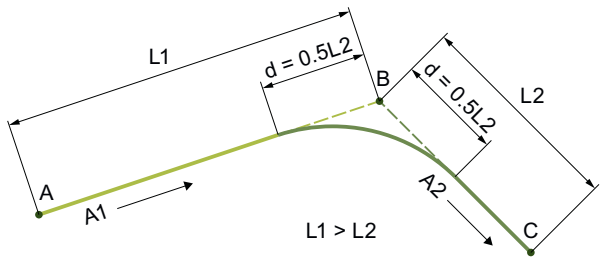 	Blend motion Because the rounding clearance is negative, the maximum rounding clearance is used. When the rounding clearance distance from the target position is reached, the active linear motion is blended with the next linear motion. Both motion jobs are blended at the lower velocity when "BufferMode" = 2 and at the higher velocity when "BufferMode" = 5.

Transitions of circular motions

With the Motion Control instructions "MC_MoveCircularAbsolute" and "MC_MoveCircularRelative", you move a kinematics system with a circular motion. You define the motion transition mode with the "BufferMode" parameter and the rounding clearance with the "TransitionParameter[1]" parameter. The following table shows how these parameters act on the motion transition based on a linear motion and a circular motion:

Rounding clearance ("TransitionParameter[1]")	Motion transition ("BufferMode")	Description
Not relevant	"BufferMode" = 1 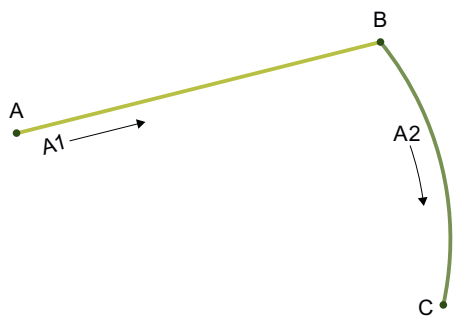 	Append motion The active linear motion is completed and the kinematics comes to a standstill. The circular motion is then executed.
d > 0.0	"BufferMode" = 2, 5 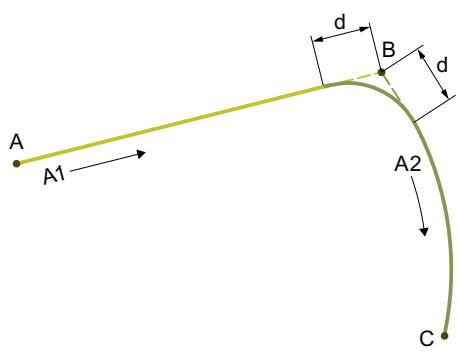 	Blend motion When the rounding clearance distance from the target position is reached, the active linear motion is blended with the circular motion. Both motion jobs are blended at the lower velocity when "BufferMode" = 2 and at the higher velocity when "BufferMode" = 5.

Rounding clearance ("TransitionParameter[1]")	Motion transition ("BufferMode")	Description
d = 0.0	"BufferMode" = 2, 5 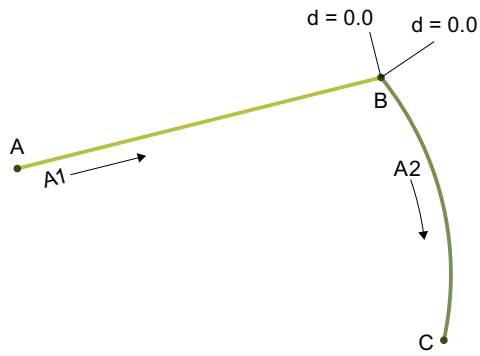	Blend motion Because the rounding clearance is 0.0, traversing is the same as when "BufferMode" = 1. The active linear motion is completed and the kinematics comes to a standstill. The circular motion is then executed.
d < 0.0	"BufferMode" = 2, 5 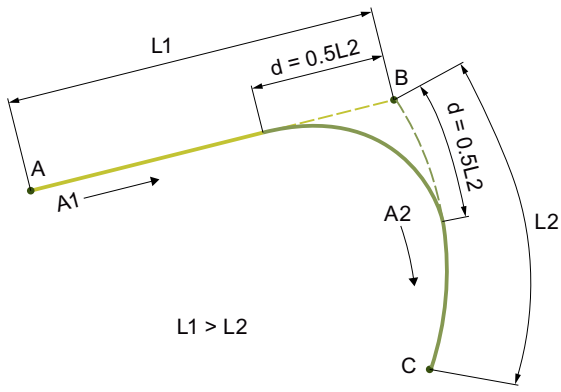	Blend motion Because the rounding clearance is negative, the maximum rounding clearance is used. When the rounding clearance distance from the target position is reached, the active linear motion is blended with the circular motion. Both motion jobs are blended at the lower velocity when "BufferMode" = 2 and at the higher velocity when "BufferMode" = 5.

Maximum rounding clearance

The maximum rounding clearance is used if the value of the "TransitionParameter[1]" parameter is < 0.0. The maximum rounding clearance is calculated as half the shorter path distance of the two motions.

Requirement	Maximum rounding clearance
L1 > L2	d_{max} = ½ · L2
L1 < L2	d_{max} = ½ · L1

L1 Path length of the first job

L2 Path length of the second job

Dynamic behavior when motions are appended/blended

You define the dynamic behavior for the transition of kinematics motions with the "BufferMode" and "DynamicAdaption" parameters.

Multiple motions can be appended to one another, in which case the kinematics comes to a standstill between the individual motions ("BufferMode" = 1). To achieve an uninterrupted motion, the individual motions can be blended with a blending segment. The consecutive

motions can be blended at the lower velocity ("BufferMode" = 2) or at the higher velocity ("BufferMode" = 5).

Dynamic adaptation

For active dynamic adaptation with segmentation, the path including the blending segment is subdivided into additional segments ("DynamicAdaption" = 1). For each of these segments, the velocity profile is calculated taking into consideration the dynamic limits of the axes which apply for individual sections of the motion. The dynamic response is therefore adapted for individual sections of a motion.

With active dynamic adaptation without segmentation of the path, the velocity profile is calculated taking into consideration the dynamic limits of the axes which apply for the entire motion ("DynamicAdaption" = 2).

Velocity and acceleration are included in the dynamic adaptation. Tangential and radial acceleration of the path is taken into account in acceleration. The jerk is not limited for dynamic adaptation.

If the dynamic adaptation is deactivated, the dynamic limits of the axes are not taken into account ("DynamicAdaption" = 0).

Interaction of kinematics motions and single axis motions

Kinematics motions are only possible if no single axis motions are active on the kinematics axes. Single axis motions have an overriding effect on kinematics motions. The motion of the corresponding axis is overridden by the single-axis motion and the job sequence is cleared. The other kinematics axes stop with the maximum dynamics.

The following functions are permitted during an active kinematics motion:

- Torque reduction on the axes/Travel to fixed stop ("MC_TorqueLimiting")
  When the fixed stop is reached, the kinematics motion is aborted.
- Setting of an additive torque ("MC_TorqueAdditive")
- Setting of the upper and lower torque limit ("MC_TorqueRange")
- Sensor switchover ("MC_SetSensor")

The following functions are rejected during an active kinematics motion:

- Overlaid motion on the axes ("MC_MoveSuperimposed")
- Homing of the axes ("MC_Home")

6.1.3.6 Commissioning

Function and structure of the kinematics control panel

With the kinematics control panel, you assume master control for a kinematics technology object and control the motions of the kinematics or the individual axes.

WARNING

Uncontrolled axis motions

During operation with the kinematics control panel, the kinematics can execute uncontrolled motions (e.g. due to incorrect configuration of the drive or technology object). In addition, when a leading axis is moved with the kinematics control panel, any synchronized following axis is also moved.

Therefore, take the following protective measures before operation with the kinematics control panel:

- Ensure that the EMERGENCY OFF switch is within the reach of the operator.
- Enable the hardware limit switches.
- Enable the software limit switches.
- Ensure that following error monitoring is enabled.
- Make sure that no following axis is coupled to the axis to be moved.

You can find kinematics control panel of the kinematics technology object in the project tree under "Technology object > Commissioning".

The kinematics control panel is divided into the following areas:

- Master control
- Kinematics
- Operating mode
- Control
- Status
- Current position values

Elements of the kinematics control panel

The following table shows the elements of the kinematics control panel:

Area	Element	Description
Master control		In the "Master control" area, you assume master control for the technology object or return it to your user program.
	"Activate" button	With the "Activate" button, you establish an online connection to the CPU and take over master control for the selected technology object. • To take over master control, the technology object must be disabled in the user program. • With the takeover of the master control of the kinematics, the master control of all axes interconnected with the kinematics is taken over. The master control can only be assumed with a kinematics control panel if axis control panel of the interconnected axes is active. • Any synchronized following axis is moved as well when a leading axis is moved with the kinematics control panel. • When you click the "Activate" button, a warning message is displayed. In the warning, you can adapt the sign-of-life monitoring (100 to 60000 ms). If the master control of the kinematics control panel is lost repeatedly without a direct error message, the online connection to the CPU may be impaired because the communication load is too high. In this case, the following message is entered in the alarm display log: "Commissioning error. Sign-of-life failure between controller and TIA Portal". To eliminate this error, adapt the sign-of-life monitoring in the warning. • Until master control is returned, the user program has no influence on the functions of the technology object. Motion Control jobs from the user program to the technology object are rejected with error ("ErrorID" = 16#8012: Kinematics control panel enabled). • When master control is taken over, the configuration of the technology object is taken over. Changes to the configuration of the technology object do not take effect until master control has been returned. Therefore, make any necessary changes before the takeover of master control. • If master control has been taken over for the technology object, the kinematics control panel and the axis control panels of the interconnected axes are blocked for access by another instance of the TIA Portal (Team Engineering as of CPU V1.5). • If the online connection to the CPU is lost during operation with the kinematics control panel, the kinematics or axis will be stopped at maximum deceleration after expiration of the sign of life monitoring. In this case, an error message is displayed ("ErrorID" = 16#8013) and the master control is passed back to the user program. • If the kinematics control panel is covered by a dialog, e.g. "Save As", during its operation, the kinematics or axis is stopped at maximum deceleration and the master control is returned to the user program. If you change to another window within the TIA Portal, e.g. to the project tree, during operation with the kinematics control panel, the master control and motion of the kinematics or axis is maintained, provided that the kinematics control panel is embedded in the TIA Portal. If the kinematics control panel is replaced by the TIA Portal and you change to another window within the TIA Portal, e.g. to the project tree, the master control is retained but the kinematics or axis is stopped at maximum deceleration.

Area	Element	Description
		If you change to another window outside the TIA Portal during operation with the kinematics control panel, the master control is retained but the kinematics or axis is stopped at maximum deceleration.
	"Deactivate" button	With the "Deactivate" button, you return master control to your user program.
Kinematics		In the "Kinematics" area, you enable or disable the technology object.
	"Enable" button	With the "Enable" button, you release the interconnected axes of the selected kinematics technology object.
	"Disable" button	With the "Disable" button, you disable the interconnected axes of the selected kinematics technology object.
Operating mode		Select the desired operating mode of the kinematics control panel in the "Operating mode" drop-down list.

Area	Element	Description
Control		The "Control" area displays the parameters for traversing with the kinematics control panel according to the selected operating mode.
	Coordinate system	In the "Coordinate system" drop-down list, you select the desired coordinate system in which the kinematics will be moved. ("Jog" and "Jog to target position" modes only)
	Active tool	Select the required tool from the "Active tool" drop-down list.
	"Customize dynamics" check box	If you select the check box, you can edit the values for acceleration, deceleration and jerk. ("Jog" and "Jog to target position" modes only)
	Acceleration	Acceleration at which the kinematics is moved dependent on the kinematics type in x, y and z direction and orientation. Preassignment: 10% of the default value You can only edit the values if the "Customize dynamics" check box is selected. ("Jog" and "Jog to target position" modes only)
	Deceleration	Deceleration at which the kinematics is moved dependent on the kinematics type in x, y and z direction and orientation. Preassignment: 100% of default value You can only edit the values if the "Customize dynamics" check box is selected. ("Jog" and "Jog to target position" modes only)
	Jerk	Jerk at which the kinematics is moved dependent on the kinematics type in x, y and z direction and orientation. Preassignment: 100% of default value You can only edit the values if the "Customize dynamics" check box is selected. ("Jog" and "Jog to target position" modes only)
	Velocity	Velocity at which the kinematics is moved dependent on the kinematics type in x, y and z direction and orientation. Preassignment: 10% of the default value You use the slider to adjust the velocity as a percentage between 0% and 200% of the set velocity values (default value 100%). ("Jog" and "Jog to target position" modes only)
	Target position	Position to which the kinematics or axis is moved. ("Jog to target position" and "Single axes: Set home position" mode only)
		Position at which the home position is set. ("Single axes: Set home position" mode only)
	"Set" button	With the "Set" button, you set a home position. ("Single axes: Set home position" mode only)
	"Start" button	With the "Start" button, you start a motion according to the selected operating mode. ("Jog to target position" and "Single axes: Homing" mode only)
	"Forward" button	With the "Forward" button, you start a motion in the positive direction according to the selected operating mode.
	"Backward" button	With the "Backward" button, you start a motion in the negative direction according to the selected operating mode.

Area	Element	Description
Status		The "Axis status" area displays the status of the axis and the status of the drive.
	Enabled	The technology object has been enabled. The axis can be moved with motion jobs.
	Homed	The technology object is homed.
	Error	An error occurred at the technology object. Error messages are displayed in the Inspector window under "Diagnostics > Alarm display".
Current position values		The "Current position values" area shows the actual values of the axis.
	Coordinate system	Coordinate system in which the kinematics or an axis is currently being moved. In the drop-down list on the right, you can select an additional coordinate system in order to display the actual position of the active tool in this coordinate system.
	x position	The current position and rotation of the tool center point in the set coordinate system.
	y position	
	z position	
	Rotation A	

Note

No transfer of the parameters

The configured parameter values are discarded when master control is returned. If necessary, transfer the values to your configuration.

If you have changed configuration values during operation with the kinematics control panel, these changes have no effect on the operation of the kinematics control panel.

Operating mode

The following table shows the operating modes of the kinematics control panel:

Operating mode	Description
Jog	With the "Forward" button, you move an axis by jogging in the positive direction. With the "Backward" button, you move an axis by jogging in the negative direction. The respective axis moves as long as you keep the "Forward" or "Backward" button pressed.
Jog to target position	With the "Start" button, you move the kinematics or an axis by jogging to the position specified under "Target position". The motion direction of the Cartesian orientation is not limited to the positive direction here. The orientation axis is moved with the "Shortest distance" mode. The kinematics moves as long as you keep the "Start" button pressed. When the target position is reached, the kinematics stops automatically. The specified position relates to the coordinate system that is selected in the "Coordinate system" drop-down list.

Operating mode	Description
Single axes: Set home position	With the "Set" button, you set the home position of the respective axis to the value specified under "Target position". The "Homed" status is set for the corresponding axis. The specified position relates to the machine coordinate system (MCS) that is preset in the "Coordinate system" drop-down list with this operating mode. This function corresponds to direct homing (absolute). Homing is not possible with an absolute encoder. The technology object is not referenced when this mode is used with an absolute encoder.
Single axes: Homing	With the "Start" button, you move an axis by jogging to the predefined home position. The respective axis moves as long as you keep the "Start" button pressed. When the home position is reached, the axis stops automatically.

Using the kinematics control panel

With the kinematics control panel, you assume master control for a kinematics technology object and control the motions of the kinematics or the individual axes.

Requirement

- The project is created and downloaded to the CPU.
- The CPU is in RUN operating state.
- The interconnected axes of the kinematics are disabled by your user program ("MC_Power.Enable" = FALSE).
- The kinematics control panel for the technology object is not being used by another instance of the TIA Portal (Team Engineering as of CPU V1.5).
- The drives are ready.

Procedure

To control the kinematics or the kinematics axes with the kinematics control panel, follow these steps:

1. To assume master control for the technology object and to establish an online connection to the CPU, click the "Activate" button in the "Master control" area.
   A warning message is displayed.
2. If necessary, adapt the sign-of-life monitoring and click "OK".
3. To enable the technology object, click the "Enable" button in the "Kinematics" area.
4. In the drop-down list in the "Operating mode" area, select the desired function of the kinematics control panel.
5. Select the reference coordinate system depending on the set operating mode in the "Control" area in the "Coordinate system" drop-down list.
6. In the "Control" area, specify the appropriate parameter values for your job.
7. Depending on the set operating mode, click the "Set", "Start", "Forward" or "Backward" button to start the job.
8. Repeat steps 4 through 7 for additional jobs.

9. To disable the technology object, click the "Disable" button in the "Axis" area.
10. To return master control to your user program, click the "Deactivate" button in the "Master control" area.

Kinematics trace

The kinematics trace offers the following functions:

- 3D visualization of the current motion of the tool center point (TCP)
- Recording and playing back kinematics motions
- Exporting and importing the recordings

You can find the kinematics trace of the kinematics technology object in the project tree under "Technology object > Kinematics trace".

3D visualization

The "3D visualization" view of the kinematics trace is divided into the following areas:

- Graphic display
  The graphic display shows the kinematics and the movement of the tool center point (TCP) as a tracer.

  Note

  You set the scaling of the kinematics in the kinematics trace under "Technology object > Configuration > Geometry (Page 2264)".

- Tabular editor
  The recordings are listed in the tabular editor.

Toolbar

The toolbar of the kinematics trace provides you with buttons for the following functions:

Button	Function	Description
	Monitor all	Establishing/closing an online connection The kinematics trace establishes an online connection to the device. If the online/offline trace configurations differ, the trace configuration is loaded to the device.
	Start recording	See section "Record and play traces (Page 2299)"
	Stop recording	

Button	Function	Description
	Import recording from file	See section "Importing and exporting recordings (Page 2302)"
	Export selected recording to file	
	Add selected recording to measurements	Recordings are added to measurements under Trace See section "Importing and exporting recordings (Page 2302)"

The toolbar at the top of the graphical display provides you with buttons for the following functions:

Button	Function	Description
	Switch to live display	Switch live display of the kinematics on/off. With the live display you can view how the kinematics is currently moving. This function is only available in online mode.
	Darker	Set the brightness of the graphical view
100	Select brightness	
	Lighter	
	Show 2D view	Switch to 2D view
	Show 3D view	Switch to 3D view
	Focus on TCP	Switch view to "Focus TCP" During the path motion of the kinematics, the focus is always on the TCP. In standstill, the view can be moved or rotated using the mouse.
	Fit to screen size	The view is centered and the entire kinematics is displayed.
	Show coordinate system	Hide/show selected coordinate system including grid lines
WCS	Select coordinate system	Select the coordinate system.
	Show xz plane	The xz plane is displayed.
	Show xz plane rotated	The xz plane is displayed rotated around the z-axis.
	Show yz plane	The yz plane is displayed.
	Show yz plane rotated	The yz plane is displayed rotated around the z-axis.
	Show xy plane	The xy plane is displayed.
	Show xy plane rotated	The xy plane is displayed rotated around the x-axis.
	Display simplified model	Simplified representation of the kinematics or show/hide kinematics
	Highlight TCS	Show/hide tool coordinate system

The toolbar at the bottom of the graphical display provides the following functions via buttons:

Button	Function	Description
	Jump to start	Jump to start of the recording
	Step backward	Jump backwards step-by-step in the recording
	Play	Playing the recording
	Pause	Pause recording
	Stop	Stop recording
	Step forward	Jump forward step-by-step in the recording
	Jump to end	Jump to the end of the recording

Operation of the graphical display

You have the following operating options using the mouse, within the graphical view:

- Rotate and zoom the coordinate system
- If you position the mouse pointer on the tracer, the following values are displayed for the corresponding position:
  - Measuring point number
  - Coordinates x, y, z and A

Record and play traces

Record path motion

Click the button to record a path motion of the kinematics. If no online connection is available at this point, it is established automatically.

The recording takes place with defaults that are stored under "Configuration" (Page 2301).

During the recording, the graphical view shows a live image of the kinematics. In case of user transformations, the graphical view shows the moved tool center point (TCP) instead.

The "current recording" and the configuration are retained until the "kinematics trace" user interface is closed.

When you start a new recording, the "Current recording" is overwritten.

Saving current recording

1. In "Current recording", edit the entries for name and color of the trace and comment according to your requirements.
2. To save the "Current recording", use the icon in the table or the shortcut menu.
   The recording is inserted in the tabular editor. A saved recording contains the following kinematics data:
   – Kinematics coordinates
   – Kinematics type
   – Valid online geometry at the time of the recording

Note

The tabular editor can save a maximum of 20 recordings. If you want to save additional recordings, you have to delete recordings which are not needed.

Toolbar for recordings

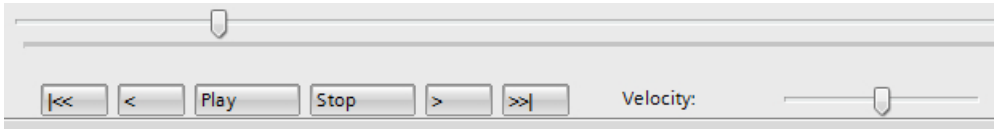

The toolbar for recordings offers the following operating options:

- Timeline with slider
  To stop the playback at a specific position, click directly on the desired position in the timeline.
- Buttons for playing and stopping the recording
- One measuring point forwards or back
- Jump to beginning / end of the recording
- Slider to set the playback speed

Show and play path motion as a trace

1. When the live monitor of the kinematics is activated, switch it off using the button in the toolbar.
2. To switch on the trace, click on the button of the recording that you want to play.
   This selects the recording. The graphical viewer displays the complete path motion of the recording.
3. Click on the "Play" button in the toolbar.
   The selected recording is played. The recording shows:
   – The recorded path in the graphic view as a trace
   – The kinematics with the online geometry that was valid at the time of the recording
     If multiple traces are visible, only the kinematics of the selected recording are shown in the graphical view.

Configuration

You specify the parameter values for the recording under "Configuration".

You can start the recording of path motions using the button in the toolbar.

Sampling

Parameter	Description
Time of recording	Choice of the following OBs: • MC-Servo • MC-Interpolator
Record every	Specify the value for the recording interval. With MC-Servo, you can select the following in a drop-down list: • Specification in "Cycles" • Specification in "Seconds"
Max. recording duration	Displays the calculated maximum recording duration. With a defined maximum number of measuring points, the maximum recording duration depends on the specified recording interval.
Use maximum recording duration	When the check box is selected, the recording duration is set to the maximum possible recording duration.
Recording duration (a)	Specification of recording duration: • In seconds • Number of measuring points

Trigger

Parameter	Description
Trigger mode	• Record now The recording starts immediately after downloading the configuration. • Trigger on tag The system is waiting for a trigger event that triggers the recording.
Trigger tag	You need a linked tag of the BOOL type for the "Trigger on tag" mode.
Event	Select the event that you want to use as trigger: • Positive edge • Falling edge
Pre-trigger	"Pre-trigger" defines the measuring points that are already recorded before the actual trigger condition is fulfilled. The information is given: • In seconds • Number of measuring points

Importing and exporting recordings

There are various export and import options for the current or saved recordings.

Export or import the recording as a file

The export contains the TCP position values as well as the configuration of the kinematics object.

To export a record, proceed as follows:

1. Select the record to be exported.
2. Click the button.
3. Select the desired file format "*.csv" or "*.ltr".
4. Select the desired folder.
5. Click "Export file".

An exported *.ltr file can be re-imported into the kinematics technology object.

To import a recording, proceed as follows:

1. Open the "Kinematics technology object" > Kinematics trace in the project navigation.
2. Click the button.

3. Select the desired folder.
4. Click "Import file".
   After import, the following is displayed under 3D visualization:
   – The imported kinematics
   – The path motions

Save recording as measurement

To save the recording as a measurement, proceed as follows:

1. Select the "Current recording" or a "Saved recording".
2. Save the recording using the button in the toolbar.
   The recording is filed under "Traces > Measurements".

The file contains the TCP coordinates (x, y, z, A) as well as the positions of the interconnected axes.

The recordings saved under "Measurements" can be downloaded and displayed in the trace. Trace offers you extended evaluation options to analyze path motions in detail.

Note

The recordings saved under measurements cannot be imported into the kinematics trace again.

6.1.3.7 Diagnostics

The "Diagnostics" section is limited to describing the diagnostics view of the individual technology objects in the TIA Portal.

A comprehensive description of the system diagnostics of the S7-1500 CPU can be found in the "Diagnostics" function manual (http://support.automation.siemens.com/WW/view/en/59192926).

The description of the diagnostics concept for Motion Control can be found in the "Diagnostics concept" section.

See also

Function manual "S7-1500T Motion Control", section "Diagnostics concept" (https://support.industry.siemens.com/cs/ww/en/view/109749263)

Kinematics technology object

Status and error bits

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the status and error messages for the technology object. The diagnostic function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Kinematics status

The following table shows the possible statuses of the kinematics:

Status	Description
Error	An error occurred at the technology object. Detailed information about the error is available in the "Error" area and in the "<TO>.ErrorDetail.Number" and "<TO>.ErrorDetail.Reaction" tags of the technology object. (<TO>.StatusWord.X1 (Error))
Restart active	The technology object is being reinitialized. (<TO>.StatusWord.X2 (RestartActive))
Kinematics control panel active	The kinematics control panel is activated. The kinematics control panel has master control over the technology object. The kinematics cannot be controlled from the user program. (<TO>.StatusWord.X4 (ControlPanelActive))
Restart required	Data relevant for the restart has been changed. The changes are applied only after a restart of the technology object. (<TO>.StatusWord.X3 (OnlineStartValuesChanged))

Status of the motion

The following table shows the possible statuses of the kinematics motion:

Status	Description
Done (no job running)	No motion job is active for the technology object. (<TO>.StatusWord.X6 (Done))
Linear motion active	A linear motion job is active for the technology object. (<TO>.StatusWord.X8 (LinearCommand))
Circular motion active	A circular motion job is active for the technology object. (<TO>.StatusWord.X9 (CircularCommand))
Constant velocity	The kinematics is being moved at constant velocity or is at a standstill. (<TO>.StatusWord.X12 (ConstantVelocity))
Accelerating	The kinematics is being accelerated. (<TO>.StatusWord.X13 (Accelerating))
Decelerating	The kinematics is being decelerated. (<TO>.StatusWord.X14 (Decelerating))
Motion interrupted	The active kinematics motion is interrupted by an "MC_GroupInterrupt" command . (<TO>.StatusWord.X17 (Interrupted))
Orientation movement active	An orientation motion is active for the technology object. (<TO>.StatusWord.X15 (OrientationMotion))

Error

The following table shows the possible errors:

Error	Description
System	A system-internal error has occurred. (<TO>.ErrorWord.X0 (SystemFault))
Configuration	A configuration error has occurred. One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data was incorrectly modified during runtime of the user program. (<TO>.ErrorWord.X1 (ConfigFault))
Transformation	A transformation error has occurred. (<TO>.ErrorWord.X4 (TransformationFault))
User program	An error in user program at a Motion Control instruction or its use occurred. (<TO>.ErrorWord.X2 (UserFault))
Job rejected	A job cannot be executed. A Motion Control instruction cannot be executed because necessary requirements have not been met (e.g. technology object not homed). (<TO>.ErrorWord.X3 (CommandNotAccepted))
Dynamic limitation	The dynamic values are restricted to the dynamic limits. (<TO>.ErrorWord.X6 (DynamicError))

Warnings

The following table shows the possible warnings:

Warning	Description
Configuration	One or more configuration parameters are being internally adapted temporarily. (<TO>.WarningWord.X1 (ConfigWarning))
Job rejected	A job cannot be executed. A Motion Control instruction cannot be executed because the necessary requirements are not met. (<TO>.WarningWord.X3 (CommandNotAccepted))
Dynamic limitation	The dynamic values are restricted to the dynamic limits. (<TO>.WarningWord.X6 (DynamicWarning))

See also

"StatusWord" tag (kinematics) (Page 2331)

"ErrorWord" tag (kinematics) (Page 2334)

"WarningWord" tag (kinematics) (Page 2336)

Status of the motion

You use the "Technology object > Diagnostics > Motion status" diagnostic function in the TIA Portal to monitor the motion status of the kinematics. The diagnostic function is available in online operation.

"Tool center point (TCP)" area

The following table describes the meaning of the position information of the tool center point:

Status	Description
Coordinate system	Reference coordinate system The world coordinate system (WCS) is preset in this field.
x position	x coordinate of the TCP in the WCS (<TO>.TcpInWcs.x.Position)
y position	y coordinate of the TCP in the WCS (<TO>.TcpInWcs.y.Position)
z position	z coordinate of the TCP in the WCS (<TO>.TcpInWcs.z.Position)
Rotation A	A coordinate of the TCP in the WCS (<TO>.TcpInWcs.a.Position)
Coordinate system	Reference coordinate system In the drop-down list you can select an additional coordinate system in order to display the actual position of the active tool in this coordinate system.
x position	x coordinate of the active tool in the set coordinate system.
y position	y coordinate of the active tool in the set coordinate system.
z position	z coordinate of the active tool in the set coordinate system.
Rotation A	A coordinate of the active tool in the set coordinate system.

"Dynamic values of the kinematics" area

The following table describes the meaning of the dynamic information:

Status	Description
Limit path dynamics to axis dynamics	Display of dynamic values with or without configured dynamic adaptation
Velocity	Path velocity (<TO>.StatusPath.Velocity)
Acceleration	Path acceleration (<TO>.StatusPath.Acceleration)
Override	Percentage correction of the velocity specification The setpoint velocity set in Motion Control instructions or from the kinematics control panel is superimposed with an override signal and corrected as a percentage. Values from 0.0% to 200.0% are permissible for the velocity correction. (<TO>.Override.Velocity)

"Job sequence" area

The following table describes the meaning of the job information:

Status	Description
Job in the job sequence	Current number of jobs for the kinematics technology object in the job sequence (<TO>.StatusMotionQueue.NumberOfCommands)

Zones and tools

You use the "Technology object > Diagnostics > Zones and tools" diagnostic function to monitor the zone and tool status of the kinematics in the TIA Portal. The diagnostic function is available in online operation.

"Zones" area

The "Workspace zones" and "Kinematics zones" tables show the status of the individual zones. The following symbols are displayed for this purpose:

Symbol	Description
	The zone is invalid.
	The zone is inactive.
	The zone is active.
	The zone was violated.

"Active tool parameters" area

The following table describes the meaning of the tool information:

Status	Description
Active tool	Currently active tool (<TO>.StatusTool.ActiveTool)
Tool center point in the FCS	In the "Tool center point in the FCS", the values of the current tool frames are displayed in the flange coordinate system (FCS).
x position	x coordinate (<TO>.StatusTool.Frame[1].x)
y position	y coordinate (<TO>.StatusTool.Frame[1].y)
z position	z coordinate (<TO>.StatusTool.Frame[1].z)
Rotation A	A coordinate (<TO>.StatusTool.Frame[1].a)

6.1.3.8 Appendix

Tags of the kinematics technology object

Legend

Tag	Name of the tag	
Data type	Data type of the tag	
Values	Value range of the tag - minimum value to maximum value (L - linear specification R - rotary specification) If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".	
W	Effectiveness of changes in the technology data block	
	DIR	Direct: Values are changed directly and take effect at the start of the next MC-Servo [OB91].
	CAL	At call of Motion Control instruction: Values are changed directly and take effect at the start of the next MC-Servo [OB91] after the call of the corresponding Motion Control instruction in the user program.
	RES	Restart: Changes to the start value in the load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed during runtime of the user program.
Description	Description of the tag	

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

"Tcp" tag (kinematics)

The tag structure "<TO>.Tcp.<Tag name>" contains the position of the tool center point (TCP), the TCP frame in the World Coordinate System (WCS)

Tags

Tag	Data type	Values	W	Description
Tcp.	STRUCT			
x	LREAL	-1.79769E308 to 1.79769E308	RON	x coordinate
y	LREAL	-1.79769E308 to 1.79769E308	RON	y coordinate
z	LREAL	-1.79769E308 to 1.79769E308	RON	z coordinate
a	LREAL	-180 to 180	RON	A coordinate

See also

Tags of coordinate systems and frames (Page 2159)

"Kinematics" tag (kinematics)

The tag structure "<TO>.Kinematics.<Tag name>" contains the defined kinematics type.

Tags

Tag	Data type	Values	W	Description	
Kinematics.	STRUCT				
TypeOfKinematics	DINT	1 to 34	RON	Kinematics type The configuration is performed using the function view.	
				1	Cartesian portal 2D
				2	Cartesian portal 2D with orientation
				3	Cartesian portal 3D
				4	Cartesian portal 3D with orientation
				5	Roller picker 2D
				6	Roller picker 2D with orientation
				7	Roller picker 3D (vertical)
				8	Roller picker 3D with orientation (vertical)
				9	Roller picker 3D with orientation (horizontal)
				10	SCARA 3D with orientation
				11	Articulated arm 2D
				12	Articulated arm 2D with orientation
				13	Articulated arm 3D
				14	Articulated arm 3D with orientation
				15	Delta picker 2D
				16	Delta picker 2D with orientation
				17	Delta picker 3D
				18	Delta picker 3D with orientation
				19 … 20	Reserved
				21	Cylindrical robot 3D
				22	Cylindrical robot 3D with orientation
				23	Tripod 3D
				24	Tripod 3D with orientation
				25 … 30	Reserved
				31	User-defined 2D
				32	User-defined 2D with orientation
				33	User-defined 3D
				34	User-defined 3D with orientation
Parameter[1..32]	ARRAY [1..32] OF LREAL	-1.0E12 to 1.0E12	RES	Kinematics-specific parameters	

See also

Portal tags (Page 2171)
Delta picker tags (Page 2220)
Roller picker tags (Page 2185)
Articulated arm tags (Page 2206)

Cyclic robot tags (Page 2229)

Tags of tripod (Page 2236)

SCARA tags (Page 2190)

Tags of user-defined kinematics systems (Page 2237)

"KcsFrame" tag (kinematics)

The tag structure "<TO>.KcsFrame.<Tag name>" contains the frame of the Kinematic Coordinate System (KCS) in the World Coordinate System (WCS).

Tags

Tag	Data type	Value range	W	Description
KcsFrame.	STRUCT			
x	LREAL	-1.0E12 to 1.0E12	RES	x coordinate
y	LREAL	-1.0E12 to 1.0E12	RES	y coordinate
z	LREAL	-1.0E12 to 1.0E12	RES	z coordinate
a	LREAL		RES	A coordinate
		0.0		With kinematics type "2D"
		0.0		With kinematics type "2D with orientation"
		-180.0 to 179.999		With kinematics type "3D"
		-180.0 to 179.999		With kinematics type "3D with orientation"
b	LREAL		RES	B coordinate
		-180.0 to 179.999		With kinematics type "2D"
		0.0		With kinematics type "2D with orientation"
		-90.0 to 90.0		With kinematics type "3D"
		0.0		With kinematics type "3D with orientation"
c	LREAL		RES	C coordinate
		0.0		With kinematics type "2D"
		0.0		With kinematics type "2D with orientation"
		-180.0 to 179.999		With kinematics type "3D"
		0.0		With kinematics type "3D with orientation"

See also

Tags of coordinate systems and frames (Page 2159)

"OcsFrame" tag (kinematics)

The tag structure "<TO>.OcsFrame[1..3].<Tag name>" contains the frames of the Object Coordinate Systems 1 to 3 (OCS) in the World Coordinate System (WCS).

Tags

Tag	Data type	Value range	W	Description
OcsFrame[1..3].	ARRAY [1..3] OF STRUCT			
x	LREAL	-1.0E12 to 1.0E12	RES	x coordinate
y	LREAL	-1.0E12 to 1.0E12	RES	y coordinate
z	LREAL	-1.0E12 to 1.0E12	RES	z coordinate
a	LREAL		RES	A coordinate
		0.0		With kinematics type "2D"
		0.0		With kinematics type "2D with orientation"
		-180.0 to 179.999		With kinematics type "3D"
		-180.0 to 179.999		With kinematics type "3D with orientation"
b	LREAL		RES	B coordinate
		-180.0 to 179.999		With kinematics type "2D"
		0.0		With kinematics type "2D with orientation"
		-90.0 to 90.0		With kinematics type "3D"
		0.0		With kinematics type "3D with orientation"
c	LREAL		RES	C coordinate
		0.0		With kinematics type "2D"
		0.0		With kinematics type "2D with orientation"
		-180.0 to 179.999		With kinematics type "3D"
		0.0		With kinematics type "3D with orientation"

See also

Tags of coordinate systems and frames (Page 2159)

"Tool" tag (kinematics)

The tag structure "<TO>.Tool[1..3].<Tag name>" contains the tool frame in the Flange Coordinate System (FCS).

Tags

Tag	Data type	Value range	W	Description
Tool[1..3].	ARRAY [1..3] OF STRUCT			
Frame.	STRUCT	-1.0E12 to 1.0E12	RES	
x	LREAL	-1.0E12 to 1.0E12	RES	x coordinate in the FCS
y	LREAL	-1.0E12 to 1.0E12	RES	y coordinate in the FCS
z	LREAL	-1.0E12 to 1.0E12	RES	z coordinate in the FCS
a	LREAL		RES	A coordinate
		0.0		With kinematics type "2D"
		-180.0 to 179.999		With kinematics type "2D with orientation"
		0.0		With kinematics type "3D"
		-180.0 to 179.999		With kinematics type "3D with orientation"

See also

Tags of coordinate systems and frames (Page 2159)

"DynamicDefaults" tag (kinematics)

The tag structure "<TO>.DynamicDefaults.<Tag name>" contains the configuration of the dynamic presets. These settings will be used when you specify a dynamic value less than 0.0 for a Motion Control instruction. Changes to the default dynamics will be applied at the next positive edge at the "Execute" parameter of a Motion Control instruction.

Tags

Tag	Data type	Values	W	Description	
DynamicDefaults.	STRUCT				
Path.	STRUCT				
Velocity	LREAL	0.0 to 1.0E12	DIR	Default setting of the velocity of the path	
Acceleration	LREAL	0.0 to 1.0E12	DIR	Default setting of the acceleration of the path	
Deceleration	LREAL	0.0 to 1.0E12	DIR	Default setting of the deceleration of the path	
Jerk	LREAL	0.0 to 1.0E12	DIR	Default setting of the jerk of the path	
Orientation.	STRUCT				
Velocity	LREAL	0.0 to 1.0E12	DIR	Default setting of the velocity of the Cartesian orientation	
Acceleration	LREAL	0.0 to 1.0E12	DIR	Default setting of the acceleration of the Cartesian orientation	
Deceleration	LREAL	0.0 to 1.0E12	DIR	Default setting of the deceleration of the Cartesian orientation	
Jerk	LREAL	0.0 to 1.0E12	DIR	Default setting of the jerk of the Cartesian orientation	
DynamicAdaption	DINT	0 to 2	CAL	Default setting of the dynamic adaptation	
				0	No dynamic adaptation
				1	Dynamic adaptation with segmentation of the path
				2	Dynamic adaptation without segmentation of the path

See also

Tags of motion control and dynamics (Page 2253)

"DynamicLimits" tag (kinematics)

The tag structure "<TO>.DynamicLimits.<Tag name>" contains the configuration of the dynamic limits. During Motion Control, no dynamic values greater than the dynamic limits are permitted. If you have specified greater values in a Motion Control instruction, then motion is performed using the dynamic limits, and a warning is indicated (alarm 501 to 503 - Dynamic values are limited).

Tags

Tag	Data type	Values	W	Description
DynamicLimits.	STRUCT			
Path.	STRUCT			
Velocity	LREAL	0.0 to 1.0E12	DIR	Dynamic limitation for the maximum velocity of the path
Acceleration	LREAL	0.0 to 1.0E12	DIR	Dynamic limitation for the maximum velocity of the path
Deceleration	LREAL	0.0 to 1.0E12	DIR	Dynamic limitation for the maximum deceleration of the path
Jerk	LREAL	0.0 to 1.0E12	DIR	Dynamic limitation for the maximum jerk of the path
Orientation.	STRUCT			
Velocity	LREAL	0.0 to 1.0E12	DIR	Dynamic limitation for the maximum velocity of the Cartesian orientation
Acceleration	LREAL	0.0 to 1.0E12	DIR	Dynamic limitation for the maximum velocity of the Cartesian orientation
Deceleration	LREAL	0.0 to 1.0E12	DIR	Dynamic limitation for the maximum deceleration of the Cartesian orientation
Jerk	LREAL	0.0 to 1.0E12	DIR	Dynamic limitation for the maximum jerk of the Cartesian orientation

See also

Tags of motion control and dynamics (Page 2253)

"MotionQueue" tag (kinematics)

The tag structure "<TO>.MotionQueue.<Tag name>" contains the configuration of parameters of the job sequence.

Tags

Tag	Data type	Values	W	Description
MotionQueue.	STRUCT			
MaxNumberOfCommands	DINT	1 to 10	RON	Maximum number of commands in job sequence The configuration is performed using the parameter view (data structure).

See also

Tags of motion control and dynamics (Page 2253)

"Override" tag (kinematics)

The tag structure "<TO>.Override.<Tag name>" contains the configuration of override parameters.

Tags

Tag	Data type	Values	W	Description
Override.	STRUCT			
Velocity	LREAL	0.0 to 200.0	DIR	Velocity override

See also

Tags of motion control and dynamics (Page 2253)

"WorkspaceZone" tag (kinematics)

The tag structure "<TO>.WorkspaceZone[1..10].<Tag name>" contains the parameter for the workspace zones.

Tags

Tag	Data type	Values	W	Description	
WorkspaceZone[1..10].	ARRAY [1..10] OF STRUCT				
Active	BOOL	-	RES	FALSE	Workspace zone deactivated
				TRUE	Workspace zone activated
Valid	BOOL	-	RES	FALSE	Zone is not defined
				TRUE	Zone is defined
Type	DINT	0 to 2	RES	Type of the workspace zone	
				0	Blocked zone
				1	Work zone
				2	Signal zone
ReferenceSystem	DINT	0 to 3	RES	Reference coordinate system for the workspace zone	
				0	WCS
				1	OCS1
				2	OCS2
				3	OCS3
Frame.	STRUCT				
x	LREAL	-1.0E12 to 1.0E12	RES	x coordinate	
y	LREAL	-1.0E12 to 1.0E12	RES	y coordinate	
z	LREAL	-1.0E12 to 1.0E12	RES	z coordinate	
a	LREAL	-180.0 to 179.999	RES	A coordinate	
b	LREAL	-90.0 to 90.0	RES	B coordinate	
c	LREAL	-180.0 to 179.999	RES	C coordinate	
Geometry.	STRUCT				
Type	DINT	0 to 2	RES	Zone geometry	
				0	Cuboid
				1	Sphere
				2	Cylinder
Parameter[1..3]	ARRAY [1..3] OF LREAL	0.0 to 1.0E12	RES	1	Length x (cuboid) or radius (sphere, cylinder)
				2	Length y (cuboid) or height (cylinder)
				3	Length z (cuboid)

See also

Tags for zone monitoring (Page 2261)

"KinematicsZone" tag (kinematics)

The tag structure "<TO>.KinematicsZone[2..10].<Tag name>" contains the parameters for kinematic zones.

Tags

Tag	Data type	Values	W	Description	
KinematicsZone[2..10].	ARRAY [2..10] OF STRUCT				
Active	BOOL	-	RES	FALSE	Kinematics zone deactivated
				TRUE	Kinematics zone activated
Valid	BOOL	-	RES	FALSE	Zone is not defined
				TRUE	Zone is defined
ReferenceSystem	DINT	0 to 1	RES	Reference coordinate system for the kinematics zone	
				0	FCS
				1	TCS
Frame.	STRUCT				
x	LREAL	-1.0E12 to 1.0E12	RES	x coordinate	
y	LREAL	-1.0E12 to 1.0E12	RES	y coordinate	
z	LREAL	-1.0E12 to 1.0E12	RES	z coordinate	
a	LREAL	-180.0 to 179.999	RES	A coordinate	
b	LREAL	-90.0 to 90.0	RES	B coordinate	
c	LREAL	-180.0 to 179.999	RES	C coordinate	
Geometry.	STRUCT				
Type	DINT	0 to 2	RES	Zone geometry	
				0	Cuboid
				1	Sphere
				2	Cylinder
Parameter[1..3]	ARRAY [1..3] OF LREAL	0.0 to 1.0E12	RES	1	Length x (cuboid) or radius (sphere, cylinder)
				2	Length y (cuboid) or height (cylinder)
				3	Length z (cuboid)

See also

Tags for zone monitoring (Page 2261)

"StatusPath" tag (kinematics)

The tag structure "<TO>.StatusPath.<Tag name>" contains the parameters of the current kinematics movement.

Tags

Tag	Data type	Values	W	Description	
StatusPath.	STRUCT				
CoordSystem	DINT	0 to 3	RON	Coordinate system of the active motion job	
				0	World coordinate system
				1, 2, 3	Object coordinate system 1, 2, 3
Velocity	LREAL	-1.0E12 to 1.0E12	RON	Current path velocity (setpoint reference)	
Acceleration	LREAL	-1.0E12 to 1.0E12	RON	Current path acceleration (setpoint reference)	
DynamicAdaption	DINT	0 to 2	RON	Dynamic adaptation	
				0	No dynamic adaptation
				1	Dynamic adaptation with segmentation of the path
				2	Dynamic adaptation without segmentation of the path

See also

Tags of motion control and dynamics (Page 2253)

"TcpInWcs" tag (kinematics)

The tag structure "<TO>.TcpInWcs.<Tag name>" contains the parameters for the tool center point (TCP) in the World Coordinate System (WCS).

Tags

Tag	Data type	Values	W	Description
TcpInWcs.	STRUCT			
x.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of path coordinate x
Position	LREAL	-1.79769E308 to 1.79769E308	RON	Position of path coordinate x
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of path coordinate x
y.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of path coordinate y
Position	LREAL	-1.79769E308 to 1.79769E308	RON	Position of path coordinate y
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of path coordinate y
z.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of path coordinate z
Position	LREAL	-1.79769E308 to 1.79769E308	RON	Position of path coordinate z
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of path coordinate z
a.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of rotation A
Position	LREAL	-1.79769E308 to 1.79769E308	RON	Position of rotation A
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of rotation A

See also

Tags of coordinate systems and frames (Page 2159)

"TcpInOcs" tag (kinematics)

The tag structure "<TO>.TcpInOcs.<Tag name>" contains the parameters for the tool center point (TCP) in the Object Coordinate Systems 1 to 3 (OCS).

Tags

Tag	Data type	Values	W	Description
TcpInOcs[1..3].	ARRAY [1..3] OF STRUCT			
x.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of the x coordinate of the tool center point in the object coordinate system 1 to 3
Position	LREAL	-1.79769E308 to 1.79769E308	RON	x coordinate of the tool center point in the object coordinate system 1 to 3
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of the x coordinate of the tool center point in the object coordinate system 1 to 3
y.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of the y coordinate of the tool center point in the object coordinate system 1 to 3
Position	LREAL	-1.79769E308 to 1.79769E308	RON	y coordinate of the tool center point in the object coordinate system 1 to 3
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of the y coordinate of the tool center point in the object coordinate system 1 to 3
z.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of the z coordinate of the tool center point in the object coordinate system 1 to 3
Position	LREAL	-1.79769E308 to 1.79769E308	RON	z coordinate of the tool center point in the object coordinate system 1 to 3
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of the z coordinate of the tool center point in the object coordinate system 1 to 3

Tag	Data type	Values	W	Description
a.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of the A coordinate of the tool center point in the object coordinate system 1 to 3
Position	LREAL	-1.79769E308 to 1.79769E308	RON	A coordinate of the tool center point in the object coordinate system 1 to 3
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of the A coordinate of the tool center point in the object coordinate system 1 to 3

See also

Tags of coordinate systems and frames (Page 2159)

"StatusOcsFrame" tag (kinematics)

The tag structure "<TO>.StatusOcsFrame.<Tag name>" contains the frames of the Object Coordinate Systems 1 of the 3 (OCS) in the World Coordinate System (WCS).

Tags

Tag	Data type	Value range	W	Description
StatusOcsFrame[1..3].	ARRAY [1..3] OF STRUCT			
x	LREAL	-1.79769E308 to 1.79769E308	RON	x coordinate in the WCS
y	LREAL	-1.79769E308 to 1.79769E308	RON	y coordinate in the WCS
z	LREAL	-1.79769E308 to 1.79769E308	RON	z coordinate in the WCS
a	LREAL		RON	A coordinate in the WCS
		0.0		With kinematics type "2D"
		0.0		With kinematics type "2D with orientation"
		-180.0 to 179.999		With kinematics type "3D"
		-180.0 to 179.999		With kinematics type "3D with orientation"
b	LREAL		RON	B coordinate in the WCS
		-180.0 to 179.999		With kinematics type "2D"
		0.0		With kinematics type "2D with orientation"
		-90.0 to 90.0		With kinematics type "3D"
		0.0		With kinematics type "3D with orientation"
c	LREAL		RON	C coordinate in the WCS
		0.0		With kinematics type "2D"
		0.0		With kinematics type "2D with orientation"
		-180.0 to 179.999		With kinematics type "3D"
		0.0		With kinematics type "3D with orientation"

See also

Tags of coordinate systems and frames (Page 2159)

"StatusKinematics" tag (kinematics)

The tag structure "<TO>.StatusKinematics.<Tag name>" contains the status of the kinematics.

Tags

Tag	Data type	Values	W	Description	
StatusKinematics.	STRUCT				
Valid	BOOL	-	RON	Validity of the transformation values	
				FALSE	Invalid
				TRUE	Valid
LinkConstellation	DWORD	0 to n	RON	Joint position	
Bit 0	-	-	-	Angle α1 of axis A1 in the front/rear area (standard area/ overhead area)	
				0	The zero point of the FCS is located in the front area (standard area) of the joint position lines for the axis A1. $\alpha 1 = \arctan(y_{FCS}/x_{FCS})$
				1	The zero point of the FCS is located in the rear area (overhead area) of the joint position lines for the axis A1. $\alpha 1 = -\arctan(y_{FCS}/x_{FCS})$
Bit 1	-	-	-	Angle α2 of axis A2 positive/negative taking into consideration the mechanical axis coupling	
				0	α2 positive
				1	α2 negative
Bit 2	-	-	-	Angle α3 of axis A3 positive/negative taking into consideration the mechanical axis coupling	
				0	α3 positive
				1	α3 negative

See also

Tags of kinematics transformation (Page 2248)

"FlangeInKcs" tag (kinematics)

The tag structure "<TO>.FlangeInKcs.<Tag name>" contains the parameters for the Flange Coordinate System (FCS) in the Kinematics Coordinate System (KCS).

Tags

Tag	Data type	Values	W	Description
FlangeInKcs.	STRUCT			
x.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of the x coordinate of the flange coordinate system (FCS) in the kinematics coordinate system (KCS)
Position	LREAL	-1.79769E308 to 1.79769E308	RON	x coordinate of the flange coordinate system in the kinematics coordinate system
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of the x coordinate of the flange coordinate system in the kinematics coordinate system
y.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of the y coordinate of the flange coordinate system in the kinematics coordinate system
Position	LREAL	-1.79769E308 to 1.79769E308	RON	y coordinate of the flange coordinate system in the kinematics coordinate system
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of the y coordinate of the flange coordinate system in the kinematics coordinate system
z.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of the z coordinate of the flange coordinate system in the kinematics coordinate system
Position	LREAL	-1.79769E308 to 1.79769E308	RON	z coordinate of the flange coordinate system in the kinematics coordinate system
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of the z coordinate of the flange coordinate system in the kinematics coordinate system
a.	STRUCT			
Acceleration	LREAL	-1.79769E308 to 1.79769E308	RON	Acceleration of the rotation A of the flange coordinate system in the kinematics coordinate system
Position	LREAL	-1.79769E308 to 1.79769E308	RON	Position of the rotation A of the flange coordinate system in the kinematics coordinate system
Velocity	LREAL	-1.79769E308 to 1.79769E308	RON	Velocity of the rotation A of the flange coordinate system in the kinematics coordinate system

See also

Tags of coordinate systems and frames (Page 2159)

"StatusTool" tag (kinematics)

The tag structure "<TO>.StatusTool.<Tag name>" contains the parameters for the tool.

Tags

Tag	Data type	Values	W	Description
StatusTool.	STRUCT			
ActiveTool	DINT	1 to 3	RON	Currently active tool
Frame[1..1].	ARRAY [1..1] OF STRUCT			
x	LREAL	-1.79769E308 to 1.79769E308	RON	x coordinate of tool 1
y	LREAL	-1.79769E308 to 1.79769E308	RON	y coordinate of tool 1
z	LREAL	-1.79769E308 to 1.79769E308	RON	z coordinate of tool 1
a	LREAL	-1.79769E308 to 1.79769E308	RON	A coordinate of tool 1

See also

Tags of coordinate systems and frames (Page 2159)

"StatusWorkspaceZone" tag (kinematics)

The tag structure "<TO>.StatusWorkspaceZone.<Tag name>" contains the status of the workspace zones.

Tags

Tag	Data type	Values	W	Description	
StatusWorkspace-Zone[1..10].	ARRAY [1..10] OF STRUCT				
Active	BOOL	-	RON	FALSE	Workspace zone deactivated
				TRUE	Workspace zone activated
Valid	BOOL	-	RON	FALSE	Zone is not defined
				TRUE	Zone is defined
Type	DINT	0 to 2	RON	Type of the workspace zone	
				0	Blocked zone
				1	Work zone
				2	Signal zone
ReferenceSystem	DINT	0 to 3	RON	Reference coordinate system for the workspace zone	
				0	WCS
				1	OCS1
				2	OCS2
				3	OCS3
Frame.	STRUCT				
x	LREAL	-1.79769E308 to 1.79769E308	RON	x coordinate	
y	LREAL	-1.79769E308 to 1.79769E308	RON	y coordinate	
z	LREAL	-1.79769E308 to 1.79769E308	RON	z coordinate	
a	LREAL	-180.0 to 179.999	RON	A coordinate	
b	LREAL	-90.0 to 90.0		B coordinate	
c	LREAL	-180.0 to 179.999	RON	C coordinate	
Geometry.	STRUCT		RON		
Type	DINT	0 to 2	RON	Zone geometry	
				0	Cuboid
				1	Sphere
				2	Cylinder
Parameter[1..3]	ARRAY [1..3] OF LREAL	0.0 to 1.0E12	RON	1	Length x (cuboid) or radius (sphere, cylinder)
				2	Length y (cuboid) or height (cylinder)
				3	Length z (cuboid)

See also

Tags for zone monitoring (Page 2261)

"StatusKinematicsZone" tag (kinematics)

The tag structure "<TO>.StatusKinematicsZone.<Tag name>" contains the status of the kinematics zones.

Tags

Tag	Data type	Values	W	Description	
StatusKinematics-Zone[2..10].	ARRAY [2..10] OF STRUCT				
Active	BOOL	-	RON	FALSE	Kinematics zone deactivated
				TRUE	Kinematics zone activated
Valid	BOOL	-	RON	FALSE	Zone is not present
				TRUE	Zone is present
ReferenceSystem	DINT	0 to 1	RON	Reference coordinate system for the kinematics zone	
				0	FCS
				1	TCS
Frame.	STRUCT				
x	LREAL	-1.79769E308 to 1.79769E308	RON	x coordinate	
y	LREAL	-1.79769E308 to 1.79769E308	RON	y coordinate	
z	LREAL	-1.79769E308 to 1.79769E308	RON	z coordinate	
a	LREAL	-180.0 to 179.999	RON	A coordinate	
b	LREAL	-90.0 to 90.0	RON	B coordinate	
c	LREAL	-180.0 to 179.999	RON	C coordinate	
Geometry.	STRUCT				
Type	DINT	0 to 2	RON	Zone geometry	
				0	Cuboid
				1	Sphere
				2	Cylinder
Parameter[1..3]	ARRAY [1..3] OF LREAL	0.0 to 1.0E12	RON	1	Length x (cuboid) or radius (sphere, cylinder)
				2	Length y (cuboid) or height (cylinder)
				3	Length z (cuboid)

See also

Tags for zone monitoring (Page 2261)

"StatusZoneMonitoring" tag (kinematics)

The tag structure "<TO>.StatusZoneMonitoring.<Tag name>" contains the status of the violated zones.

Tags

Tag	Data type	Values	W	Description
StatusZoneMonitoring.	STRUCT			
WorkingZones	DWORD	-	RON	Display of violated work zones The bit numbers 1 to 10 correspond to the configured zone numbers.
BlockedZones	DWORD	-	RON	Display of violated blocked zones The bit numbers 1 to 10 correspond to the configured zone numbers.
SignalizingZones	DWORD	-	RON	Display of approached signal zones The bit numbers 1 to 10 correspond to the configured zone numbers.
KinematicsZones	DWORD	-	RON	Display of kinematics zones that violate workspace zones The bit number 1 indicates the monitoring status of the tool center point (TCP). The bit numbers 2 to 10 correspond to the configured zone numbers.

See also

Tags for zone monitoring (Page 2261)

"StatusMotionQueue" tag (kinematics)

The tag structure "<TO>.StatusMotionQueue.<Tag name>" contains the status of the job sequence.

Tags

Tag	Data type	Values	W	Description
StatusMotionQueue.	STRUCT			
NumberOfCommands	DINT	-	RON	Number of queued jobs in the job sequence

See also

Tags of motion control and dynamics (Page 2253)

"KinematicsAxis" tag (kinematics)

The tag structure "<TO>.KinematicsAxis.<Tag name>" contains the defined kinematics axes.

Tags

Tag	Data type	Values	W	Description
KinematicsAxis.	STRUCT			
A1	DB_ANY	-	RON	Technology object data block of the kinematics axis A1
A2	DB_ANY	-	RON	Technology object data block of the kinematics axis A2
A3	DB_ANY	-	RON	Technology object data block of the kinematics axis A3
A4	DB_ANY	-	RON	Technology object data block of the kinematics axis A4

"Units" tag (kinematics)

The tag structure "<TO>.Units.<Tag name>" contains the configured technological units.

Tags

Tag	Data type	Values	W	Description	
Units.	STRUCT				
LengthUnit	UDINT	0 to 4294967295	RON	Unit for the position	
				1010	m
				1013	mm
				1011	km
				1014	µm
				1015	nm
				1019	in
				1018	ft
				1021	mi
LengthVelocityUnit	UDINT	0 to 4294967295	RON	Unit for the velocity	
				1062	mm/s
				1061	m/s
				1524	mm/min
				1525	m/min
				1526	mm/h
				1063	m/h
				1527	km/min
				1064	km/h
				1066	in/s
				1069	in/min
				1067	ft/s
				1070	ft/min
				1075	mi/h
AngleUnit	UDINT	0 to 4294967295	RON	Unit for the position of the orientation axis	
				1004	rad
				1005	°
AngleVelocityUnit	UDINT	0 to 4294967295	RON	Unit for the velocity of the orientation axis	
				1521	°/s
				1522	°/min
				1086	rad/s
				1523	rad/min

"StatusWord" tag (kinematics)

The tag "<TO>.StatusWord" contains the status information of the technology object.

Information on the evaluation of the individual bits (e.g. bit 2 "RestartActive") can be found in the "Evaluating StatusWord, ErrorWord and WarningWord" section.

Tags

Tag	Data type	Values	W	Description
StatusWord	DWORD	-	RON	Status information of the technology object

Tag	Data type	Values	W	Description	
Bit 0	-	-	-	Reserved	
Bit 1	-	-	-	"Error"	
				0	No error is present.
				1	An error is present.
Bit 2	-	-	-	"RestartActive"	
				0	No "Restart" is active.
				1	A "Restart" is active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged"	
				0	"Restart" tags unchanged
				1	Change to "Restart" tags For the changes to be applied, the technology object must be reinitialized.
Bit 4	-	-	-	"ControlPanelActive"	
				0	The kinematics control panel is deactivated.
				1	The kinematics control panel is activated.
Bit 5	-	-	-	Reserved	
Bit 6	-	-	-	"Done"	
				0	A motion job is in progress or the kinematics control panel is activated.
				1	No motion job is in progress and the kinematics control panel is deactivated.
Bit 7	-	-	-	Reserved	
Bit 8	-	-	-	"LinearCommand"	
				0	No linear motion is active.
				1	A linear motion is active.
Bit 9	-	-	-	"CircularCommand"	
				0	No circular motion is active.
				1	A circular motion is active.
Bit 10	-	-	-	Reserved	
Bit 11	-	-	-	Reserved	
Bit 12	-	-	-	"ConstantVelocity"	
				0	The kinematics is being accelerated or decelerated.
				1	The setpoint velocity is reached. The kinematics is moving at this constant velocity or is at a standstill.
Bit 13	-	-	-	"Accelerating"	
				0	No acceleration operation is active.
				1	An acceleration operation is active.
Bit 14	-	-	-	"Decelerating"	
				0	No deceleration operation is active.
				1	A deceleration operation is active.
Bit 15	-	-	-	"OrientationMotion"	

Tag	Data type	Values	W	Description	
				Orientation movement active	
Bit 16	-	-	-	"Stopping"	
				0	No "MC_GroupStop" job is active.
				1	An "MC_GroupStop" job is active. The motion of the kinematics technology object is aborted
Bit 17	-	-	-	"Interrupted"	
				0	The motion of the kinematics technology object is not interrupted.
				1	The motion of the kinematics technology object is interrupted with an "MC_GroupInterrupt" job. The motion can be continued with an "MC_GroupContinue" job.
Bit 18	-	-	-	"Blending"	
				0	No blending segment is active.
				1	A blending segment is active.
Bit 19 … Bit 31	-	-	-	Reserved	

See also

Tags of motion control and dynamics (Page 2253)

Function Manual "S7-1500T Motion Control V4.0 in the TIA Portal V15" section "Evaluating StatusWord, WarningWord and ErrorWord" (https://support.industry.siemens.com/cs/ww/en/view/109749263)

"ErrorWord" tag (kinematics)

The tag "<TO>.ErrorWord" shows errors at the technology object (technology alarms).

Information on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the "Evaluating StatusWord, ErrorWord and WarningWord" section.

Tags

Tag	Data type	Values	W	Description
ErrorWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed. A Motion Control instruction cannot be executed because the necessary requirements are not met.
Bit 4	-	-	-	"TransformationFault" Error in the kinematics transformation
Bit 5	-	-	-	Reserved
Bit 6	-	-	-	"DynamicError" Specified dynamic values are limited to permissible values.
Bit 7 … Bit 31	-	-	-	Reserved

See also

Function Manual "S7-1500T Motion Control V4.0 in the TIA Portal V15" section "Evaluating StatusWord, WarningWord and ErrorWord" (https://support.industry.siemens.com/cs/ww/en/view/109749263)

"ErrorDetail" tag (kinematics)

The tag structure "<TO>.ErrorDetail.<Tag name>" contains the alarm number and effective local alarm reaction to the technology alarm currently pending at the technology object.

You can find a list of the technology alarms and alarm reactions in the "Technology alarms (Page 2339)" appendix.

Tags

Tag	Data type	Values	W	Description	
ErrorDetail.	STRUCT				
Number	UDINT	-	RON	Alarm number	
Reaction	DINT	0, 11, 12	RON	Effective alarm reaction	
				0	No reaction (warnings only)
				11	Stop with maximum dynamic values of the kinematics
				12	Stop with maximum dynamic values of the axes

"WarningWord" tag (kinematics)

The tag "<TO>.WarningWord" shows pending warnings at the technology object.

Information on the evaluation of the individual bits (e.g. bit 2 "UserFault") can be found in the "Evaluating StatusWord, ErrorWord and WarningWord" section.

Tags

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" A system-internal error has occurred.
Bit 1	-	-	-	"ConfigFault" Configuration error One or more configuration parameters are being internally adapted temporarily.
Bit 2	-	-	-	"UserFault" Error in user program at a Motion Control instruction or its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed. A Motion Control instruction cannot be executed because the necessary requirements are not met.
Bit 4	-	-	-	Reserved
Bit 5	-	-	-	Reserved
Bit 6	-	-	-	"DynamicWarning" Specified dynamic values are limited to permissible values.
Bit 7 … Bit 31	-	-	-	Reserved

See also

Function Manual "S7-1500T Motion Control V4.0 in the TIA Portal V15" section "Evaluating StatusWord, WarningWord and ErrorWord" (https://support.industry.siemens.com/cs/ww/en/view/109749263)

"ControlPanel" tag (kinematics)

The tag structure "<TO>.ControlPanel.<Tag name>" contains no data relevant to the user. This tag structure is internally used.

Tags

Tag	Data type	Values	W	Description
ControlPanel.	STRUCT			
Input.	STRUCT			
TimeOut	LREAL	100 to 60000	DIR	-
EsLifeSign	UDINT	-	DIR	-
Command[1..2].	ARRAY [1..2] OF STRUCT			
ReqCounter	UDINT	-	DIR	-
Type	UDINT	-	DIR	-
Position[1..4]	ARRAY [1..4] OF LREAL	-	DIR	-
Velocity[1..4]	ARRAY [1..4] OF LREAL	-	DIR	-
Acceleration[1..4]	ARRAY [1..4] OF LREAL	-	DIR	-
Deceleration[1..4]	ARRAY [1..4] OF LREAL	-	DIR	-
Jerk[1..4]	ARRAY [1..4] OF LREAL	-	DIR	-
Param[1..9]	ARRAY [1..9] OF LREAL	-	DIR	-
CoordinateSystem	UDINT	-	DIR	-
ToolNumber	UDINT	-	DIR	-
Output.	STRUCT			
RtLifeSign	UDINT	-	RON	-
Command[1..2].	ARRAY [1..2] OF STRUCT	-		
AckCounter	UDINT	-	RON	-
Error	BOOL	-	RON	-
ErrorID	WORD	-	RON	-
Done	BOOL	-	RON	-
Aborted	BOOL	-	RON	-

Technology alarms

Overview

The following table shows an overview of the technology alarms and the corresponding alarm reactions. When a technology alarm occurs, evaluate the entire indicated alarm text, in order to find the precise cause.

Legend

No.	Number of the technology alarm (corresponds to <TO>.ErrorDetail.Number)
Reaction	Effective alarm reaction (corresponds to <TO>.ErrorDetail.Reaction)
Error bit	Bit that is set in <TO>.ErrorWord when the technology alarm occurs A description of the bits can be found in the appendix (Page 2333).
Warning bit	Bit that is set in <TO>.WarningWord when the technology alarm occurs A description of the bits can be found in the appendix (Page 2335).
Restart	To acknowledge the technology alarm, the technology object must be reinitialized (Restart).
Diagnostic buffer	The alarm is entered in the diagnostics buffer.
Alarm text	Displayed alarm test (limited)

List of the technology alarms

No.	Reaction	Error bit	Warning bit	Restart	Diagnostic buffer	Alarm text
101	Stop with maximum dynamic values of the axes	X1	-	X	X	Configuration error.
201	Stop with maximum dynamic values of the axes	X0	-	X	X	Internal error.
202	Stop with maximum dynamic values of the axes	X0	-	X	-	Internal configuration error.
203	Stop with maximum dynamic values of the axes	X0	-	X	-	Internal error.
204	Stop with maximum dynamic values of the axes	X0	-	-	-	Commissioning error.
304	Stop with maximum dynamic values of the axes	X2	-	-	-	Velocity limit is zero.
305	Stop with maximum dynamic values of the axes	X2	-	-	-	• Limit value of the acceleration is zero. • Limit value of the deceleration is zero.

No.	Reaction	Error bit	Warning bit	Restart	Diagnostic buffer	Alarm text
306	Stop with maximum dynamic values of the axes	X2	-	-	-	Jerk limit is zero.
501	No reaction (warnings only)	-	X6	-	-	Programmed velocity is limited.
502	No reaction (warnings only)	-	X6	-	-	• Programmed acceleration is being limited. • Programmed deceleration is being limited.
503	No reaction (warnings only)	-	X6	-	-	Programmed jerk is limited.
561	No reaction (warnings only)	-	X6	-	-	Programmed velocity of the orientation motion is limited.
562	No reaction (warnings only)	-	X6	-	-	• Programmed acceleration of the orientation motion is limited. • Programmed deceleration of the orientation motion is limited.
563	No reaction (warnings only)	-	X6	-	-	Programmed jerk of the orientation motion is limited.
801	Stop with maximum dynamic values of the axes	X2	-	-	-	Kinematics axis not ready.
802	Stop with maximum dynamic values of the axes	X3	-	-	-	Cannot calculate the geometry element.
803	Stop with maximum dynamic values of the axes	X4	-	-	-	Error in the calculation of the transformation.
804	Stop with maximum dynamic values of the axes	X2	-	-	-	The kinematics motion cannot be stopped at the end.
805	Stop with maximum dynamic values of the axes	X2	-	-	-	Limitation of the path dynamics by the dynamics of the kinematics axes.
806	Stop with maximum dynamic values of the kinematics	X2	-	-	-	Collision with work or blocked zones occurred.
807	No reaction (warnings only)	-	X2	-	-	Collision with work or blocked zones occurred.
808	Stop with maximum dynamic values of the axes	X2	-	-	-	Ambiguity due to multiple active work zones.
809	Stop with maximum dynamic values of the axes	X2	-	-	-	Path dynamic limit through dynamic of the orientation motion faulty.

Alarm reaction

A technology alarm always contains an alarm reaction, which describes the effect on the technology object. The alarm reaction is specified by the system.

The following table shows possible alarm reactions:

Alarm reaction	Description
No reaction (warnings only) <TO>.ErrorDetail.Reaction = 0	The processing of the Motion Control jobs will be continued. The active motion of the kinematics can be influenced, e.g. by limiting the current dynamic values to the configured limit values.
Stop with maximum dynamic values of the kinematics <TO>.ErrorDetail.Reaction = 11	Active and queued motion jobs are canceled. The kinematics is decelerated with the maximum dynamic values configured under "Technology object > Configuration > Extended parameters > Dynamics" and brought to a standstill. The configured maximum jerk is hereby taken into account.
Stop with maximum dynamic values of the axes <TO>.ErrorDetail.Reaction = 12	Active and queued motion jobs are canceled. The axes are decelerated with the maximum dynamic values configured under "Technology object > Configuration > Extended parameters > Limits > Dynamic limits", and brought to a standstill. The configured maximum jerk is hereby taken into account.

Technology alarms 101

Technology alarm 101

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Required

Alarm text	Remedy
Configuration error.	
Value in <tag> not allowed.	Adjust the specified value.
Interconnection of axis <no.> missing.	Interconnect the axis.
Interconnection of orientation axis missing.	
Delta 2D picker: No formation of a closed parallel structure.	Adjust the geometry data of the mechanics.
Delta 3D picker: No formation of a closed parallel structure.	
Delta 3D picker: Angular offset does not permit a third arm.	
Roll picker: Radius incorrect.	
Invalid arm distances.	
Limits for specifying orientations violated.	Adjust the corresponding value.

Technology alarms 201 - 204

Technology alarm 201

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Required

Alarm text	Remedy
Internal error.	Contact customer service.

Technology alarm 202

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Required

Alarm text	Remedy
Internal configuration error.	Contact customer service.

Technology alarm 203

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Required

Alarm text	Remedy
Internal algorithm error.	Contact customer service.

Technology alarm 204

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Required

Alarm text	Remedy
Commissioning error.	
Connection to the TIA Portal interrupted.	Check the connection properties.

Technology alarms 304 - 306

Technology alarm 304

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Not required

Alarm text	Remedy
Velocity limit is zero.	Enter a value for the velocity (DynamicLimits.Path.Velocity) that does not equal zero in the limits for dynamics.

Technology alarm 305

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Not required

Alarm text	Remedy
Acceleration/deceleration limit is zero.	
Acceleration	Enter a value for the acceleration (DynamicLimits.Path.Acceleration) that does not equal zero in the limits for dynamics.
Deceleration	Enter a value for the deceleration (DynamicLimits.Path.Deceleration) that does not equal zero in the limits for dynamics.

Technology alarm 306

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Not required

Alarm text	Remedy
Jerk limit is zero.	Enter a value for the JERK (DynamicLimits.Path.Jerk) that does not equal zero in the limits for dynamics.

Technology alarms 501 - 563

Technology alarm 501

Alarm reaction: No reaction (warnings only)

Restart: Not required

Alarm text	Remedy
Programmed velocity is limited.	• Check the value for the velocity of the Motion Control instruction. • Check the configuration of the dynamic limits.

Technology alarm 502

Alarm reaction: No reaction (warnings only)

Restart: Not required

Alarm text	Remedy
Programmed acceleration/deceleration is limited.	
Acceleration	• Check the value for the acceleration of the Motion Control instruction. • Check the configuration of the dynamic limits.
Deceleration	• Check the value for the deceleration of the Motion Control instruction. • Check the configuration of the dynamic limits.

Technology alarm 503

Alarm reaction: No reaction (warnings only)

Restart: Not required

Alarm text	Remedy
Programmed jerk is limited.	• Check the value for the jerk of the Motion Control instruction. • Check the configuration of the dynamic limits.

Technology alarm 561

Alarm reaction: No reaction (warnings only)

Restart: Not required

Alarm text	Remedy
Programmed velocity of the orientation motion is limited.	• Check the configuration of the velocity of the orientation motion.

Technology alarm 562

Alarm reaction: No reaction (warnings only)

Restart: Not required

Alarm text	Remedy
Programmed acceleration/deceleration of the orientation motion is limited.	
Acceleration	• Check the configuration of the acceleration of the orientation motion.
Deceleration	• Check the configuration of the deceleration of the orientation motion.

Technology alarm 563

Alarm reaction: No reaction (warnings only)

Restart: Not required

Alarm text	Remedy
Programmed jerk of the orientation motion is limited.	• Check the configuration of the jerk of the orientation motion.

Technology alarms 801 - 809

Technology alarm 801

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Not required

Alarm text	Remedy
Kinematics axis <no.> not ready.	
Axis not enabled.	Enable the technology object.
Axis command programmed.	To transmit another kinematics command, put the specified axis in a standstill.
Axis alarm.	Check and acknowledge the technology alarms of the specified kinematics axis.

Technology alarm 802

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Not required

Alarm text	Remedy
Cannot calculate the geometry element.	
Radius for "CircMode" = 2 less than half the path distance.	Adjust the radius.
Starting point, intermediate point and/or end point are identical when "CircMode" = 0.	Specify different values for starting point, intermediate point and end point.
Intermediate point cannot be reached when "CircMode" = 0.	Adjust the intermediate point.
Starting point and end point are identical when "CircMode" = 2 as well as "PathChoice" = 2 or 3 is specified.	Define different start and end points.
Unable to execute dynamic adaptation.	Switch off the dynamic adaptation.

Technology alarm 803

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Not required

Alarm text	Remedy
Error calculating the transformation.	
Transformation of the axis coordinates to the kinematics coordinates resulted in an error.	• Correct your specified motion with regard to the joint positioning space and the transformation areas: • Position the kinematics axes with single-axis motions in a permitted transformation area. • For user transformation: Check the calculation in the MC-Transformation [OB98].
With user-defined kinematics systems: "FunctionResult" of the MC-Transformation [OB98]	
Transformation of the kinematics coordinates to the axis coordinates resulted in an error.	
With predefined kinematics systems: Additional info:	
0 Cartesian position cannot be reached	
1 Singular position	
With user-defined kinematics systems: "FunctionResult" of the MC-Transformation [OB98]	

Technology alarm 804

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Not required

Alarm text	Remedy
Kinematics motion cannot be stopped at end.	Ensure that the path is sufficiently long.

Technology alarm 805

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Not required

Alarm text	Remedy
Limitation of the path dynamics by the axis dynamics faulty.	
Path velocity is limited to zero.	Configure a larger path velocity for the kinematics axis.
Path acceleration or deceleration is limited to zero.	Configure a larger path acceleration or deceleration for the kinematics axis.

Technology alarm 806

Alarm reaction: Stop with maximum dynamic values of the kinematics

Restart: Not required

Alarm text	Remedy
Collision with work or blocked zones occurred.	Move the kinematics object from the work zone or blocked zone.

Technology alarm 807

Alarm reaction: No reaction (warnings only)

Restart: Not required

Alarm text	Remedy
Collision with work or blocked zones occurred.	-

Technology alarm 808

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Not required

Alarm text	Remedy
Ambiguity due to multiple active work zones.	
<Number of the currently active zone>	Activate only one work zone.

Technology alarm 809

Alarm reaction: Stop with maximum dynamic values of the axes

Restart: Not required

Alarm text	Remedy
Path dynamic limit through dynamic of the orientation motion faulty.	
Velocity is limited to zero.	Configure a higher maximum velocity for the axes involved in the orientation motion.
Acceleration/deceleration is limited to zero.	Configure a higher maximum acceleration or deceleration for the axes involved in the orientation motion.

Error ID (kinematics)

Errors in Motion Control instructions are signaled using the "Error" and "ErrorID" parameters.

Under the following conditions, "Error" = TRUE and "ErrorID" = 16#8xxx are indicated for the Motion Control instruction:

- Invalid status of the technology object that prevents execution of the job.
- Invalid parameter assignment of the Motion Control instruction that prevents execution of the job.
- As a result of the alarm reaction to a technology object error.

The following tables show a list of all "ErrorIDs" that can be indicated for Motion Control instructions. Besides the cause of the error, remedies for eliminating the error are also listed:

16#0000 - 16#800F

ErrorID	Description	Remedy
16#0000	No error	-
16#8001	A technology alarm (technology object error) occurred while processing the Motion Control instruction.	In the technology data block, an error message is output at the "ErrorDetail.Number" tag. You can find a list of the technology alarms and alarm reactions in the Technology alarms (Page 2340) appendix.
16#8002	Illegal specification of the technology object	Check the specification of the technology object for the "Axis" or "AxesGroup" parameter. You can use a kinematics technology object only for the "AxesGroup" parameter.
16#8003	Illegal velocity specification	Specify a permissible value for the velocity for the "Velocity" parameter.
16#8004	Illegal acceleration specification	Specify a permissible value for the acceleration for the "Acceleration" parameter.
16#8005	Illegal deceleration specification	Specify a permissible value for the deceleration for the "Deceleration" parameter.
16#8006	Illegal jerk specification	Specify a permissible value for the jerk for the "Jerk" parameter.
16#8007	Illegal direction specification	Specify a permissible value for the motion direction for the "DirectionA" parameter.
16#8008	Illegal specification of the relative target coordinate	Specify permissible values for the relative target coordinate for the "Distance" parameter.
16#8009	Illegal specification of the absolute target coordinate	Specify a permissible value for the absolute target coordinate in the "Position" parameter.
16#800A	Illegal mode specification	Specify a permissible value for the mode for the "Mode" parameter.
16#800D	The job is not permitted in the current state. Restart is being executed.	While a restart is being performed, the technology object cannot perform any jobs. During a restart of the kinematics technology object, all interconnected axes are reinitialized first. Then the kinematics technology object is reinitialized. This may take up to one second. Wait until the technology object restart is complete.
16#800F	The job cannot be executed because the technology object is disabled.	Enable the technology object with "MC_Power.Enable" = TRUE. Restart the job.

16#8010 - 16#807F

ErrorID	Description	Remedy
16#8012	The job cannot be executed because the kinematics control panel is active.	Return the master control to your user program. Restart the job.
16#8014	No internal job memory available.	The maximum possible number of Motion Control jobs has been reached. Reduce the number of jobs to be executed (parameter "Execute" = FALSE).
16#8015	Error acknowledgment with "MC_Reset" is not possible. Error in the configuration of the technology object.	Check the configuration of the technology object.

16#80A0 - 16#8FFF

ErrorID	Description	Remedy
16#80B1	Illegal specification of the coordinate system	Specify a permissible value for the coordinate system for the "CoordSystem" parameter.
16#80B2	Illegal specification of the motion transition	Specify a permissible value for the motion transition for the "BufferMode" parameter.
16#80B3	Illegal specification of the rounding clearance	Specify a permissible value for the rounding clearance for the "TransitionParameter" parameter.
16#80B5	Illegal specification of the dynamic adaptation	Specify a permissible value for the dynamic adaptation for the "DynamicAdaption" parameter.
16#80B6	Illegal specification for the definition of the circular path	Specify a permissible value for the definition of the circular path for the "CircMode" parameter.
16#80B7	Illegal specification for the circular path auxiliary point	Specify a permissible value for the circular path auxiliary point for the "AuxPoint" parameter.
16#80B8	Illegal specification of the target position	Specify a permissible value for the target position for the "EndPoint" parameter.
16#80B9	Illegal specification of the orientation of the circular path	Specify a permissible value for the orientation of the circular path for the "PathChoice" parameter.
16#80BA	Illegal specification for the main plane of the circular path	Specify a permissible value for the main plane of the circular path for the "CirclePlane" parameter.
16#80BB	Illegal radius specification	Specify a permissible value for the radius of the circular path for the "Radius" parameter.
16#80BC	Illegal angle specification	Specify a permissible value for the angle of the circular path for the "Arc" parameter.
16#80C1	Illegal specification of the zone type	Specify a permissible value for the zone type for the "ZoneType" parameter.
16#80C2	Illegal specification of the zone position	Specify a permissible value for the zone number for the "ZoneNumber" parameter.
16#80C3	Illegal specification of the reference system	Specify a permissible value for the reference system for the "ReferenceSystem" parameter.
16#80C4	Illegal coordinate specification	Specify permissible values for the coordinates for the "Frame" parameter.
16#80C5	Illegal specification of the zone geometry	Specify a permissible value for the zone geometry for the "GeometryType" parameter.
16#80C6	Illegal specification of the geometric parameters	Specify permissible values for the geometric parameters for the "GeometryParameter" parameter.
16#80C7	The zone is not defined.	Define a workspace zone using the "MC_DefineWorkspaceZone" job or a kinematics zone using the "MC_DefineKinematicsZone" job.
16#80C8	A tool cannot be redefined during a motion.	End the active motion. Restart the "MC_DefineTool" job.
	An active tool cannot be changed during a motion.	End the active motion. Restart the "MC_SetTool" job.
16#80CA	Illegal specification of the tool number	Specify a permissible value for the tool number for the "ToolNumber" parameter.
16#80CB	Illegal specification of the object coordinate system	Specify a permissible value for the object coordinate system for the "OcsNumber" parameter.
16#80CC	The job cannot be executed because a single-axis motion is active at a kinematics axis.	End the current single-axis motion. Restart the job.

ErrorID	Description	Remedy
16#80CD	The job cannot be executed because an "MC_GroupStop" job is active.	Set the "MC_GroupStop.Execute" parameter to FALSE. Restart the job.
16#80CE	The job sequence is used to capacity.	The maximum possible Motion Control jobs have been transmitted.
16#8FFF	Unspecified error	Contact your local Siemens representative or support center. You can find your contact for Industry Automation and Drive Technologies at: http://www.siemens.com/automation/partner (http://www.siemens.com/automation/partner)

6.2 Motion Control (S7-300, S7-400) (S7-300, S7-400)

6.2.1 Position feedback basics (S7-300, S7-400)

6.2.1.1 What is position feedback? (S7-300, S7-400)

Position feedback

Position feedback is the detection of actual values scaled to units of length.

It differs depending on the encoder used:

- Incremental encoder
  The module counts the incremental encoder signals. A reference point approach is required before position feedback to create a reference point for the encoder position. The counter can, for example, be set to "0" at a reference mark.
  Following the reference point approach (with synchronized axis), the counter value is a measurement of the current position.
- Absolute encoder
  No reference point approach is required. The signal output represents a fixed position.

The values determined can be processed directly in the user program. A direct response to encoder values from motion systems is therefore possible.

6.2.1.2 Encoder value acquisition (S7-300, S7-400)

Description

The absolute encoder transfers its encoder values to the module in frames. The transmission of frames is initiated by the module.

- In non-isochrone mode, the encoder values are acquired freewheeling.
- In isochrone mode, the encoder values are acquired in synchronism with the bus cycle at each T_i.

Freewheeling encoder value acquisition

The module initiates the transmission of a frame each time the parameterized monoflop time elapses.

The module processes the acquired encoder value asynchronously to these freewheeling frames in the cycle of the update rate.

Freewheeling acquisition therefore returns encoder values of different ages. The difference between the maximum and minimum age is the jitter.

Isochrone encoder value acquisition

Isochrone encoder value acquisition is automatically set if the bus system and I/O modules are synchronized.

The module initiates a frame transfer in each bus cycle at T_i.

The module processes the transmitted encoder value isochronously to the bus cycle.

6.2.1.3 Gray/dual converter (S7-300, S7-400)

Gray code/dual code converter

The "Gray code/dual code converter allows you to define how the codes returned by the encoder are processed:

- When "Gray code" is set, the encoder value returned by the absolute encoder in gray code will be converted to dual code.
- When "Dual code" is set, the values returned by the encoder will remain unchanged.

Note

If you have selected the "Gray code" setting, the module will always convert the total encoder value (13 to 25 bits). Any leading special bits will therefore influence the encoder value and the appended bits may be corrupted.

6.2.1.4 Encoder value and scaling (S7-300, S7-400)

Encoder value and scaling

The encoder value transferred contains the encoder position of the absolute encoder. In addition to the encoder position, the encoder transfers additional bits located before and after the encoder position, depending on the encoder used.

The module requires the following information if it is to determine the encoder position:

- Encoder type
- Number of adjusted bits
- Increments per encoder revolution

Scaling specifies the position of the encoder value at the feedback interface.

- You specify that trailing, irrelevant bits in the encoder value are to be discounted by entering either "Number of adjusted bits = 1, 2 ... 12" (for SM 338) or "Scaling on" (for ET 200S 1SSI). The encoder value is then positioned in the address area, right-aligned (see example below).
- Entering "Number of adjusted bits = 0" (for SM 338) or "Scaling of" (for ET 200S 1SSI) will mean adjusted bits are retained and available for evaluation.
  This can be useful is you are using an absolute encoder which transfers information in the adjusted bits (see manufacturer's specifications) and you wish to evaluate this information. Please also read the Gray code/dual code converter (Page 2351) information.

Example of encoder value scaling

Requirements:

You are using a single-turn encoder with 2^9 steps (corresponds to 9 bits) = 512 increments/ encoder revolution (resolution/360°) with the following parameter assignment:

- Encoder type: SSI-13 bit
- Number of adjusted bits: 4 places
- Increments per encoder revolution: 512

The following image shows the location of the relevant bits before and after scaling.

Prior to normalizing: cyclically measured encoder value 100

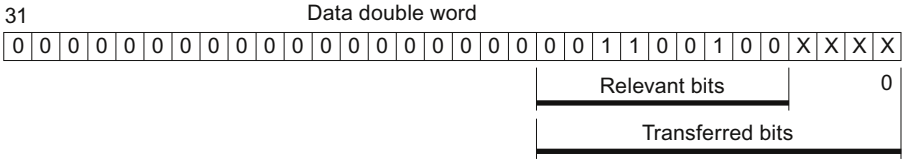

After normalizing: Encoder value 100

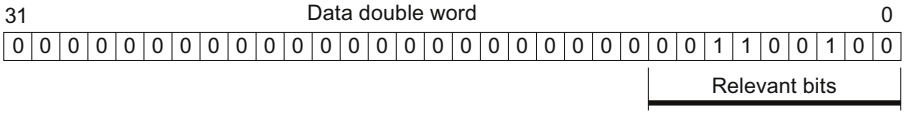

Result: bits 0 to 3 (4 digits, marked with "x") were omitted.

6.2.2 Using SM 338 (S7-300, S7-400)

6.2.2.1 Terminal assignment of the front connector of SM 338 (S7-300, S7-400)

Wiring rules

Important rules to observe when wiring the module:

- Ground of the encoder supply is connected to CPU ground potential. Thus, establish a low impedance connection between pin 2 of SM 338 (M) and CPU ground.
- Always use shielded twisted-pair cable to wire encoder signals (pins 3 to 14.) Connect both ends of the shielding.
  You use the shield contact element for the shield contact at the SM 338 (article number: 6ES7390-5AA00-0AA0).
- Connect an external power supply if the maximum output current (900 mA) of the encoder supply is exceeded.

Terminal assignment of the front connector and block diagram

The figure below shows the terminal assignment of the front connector of the SM 338 and the block diagram.

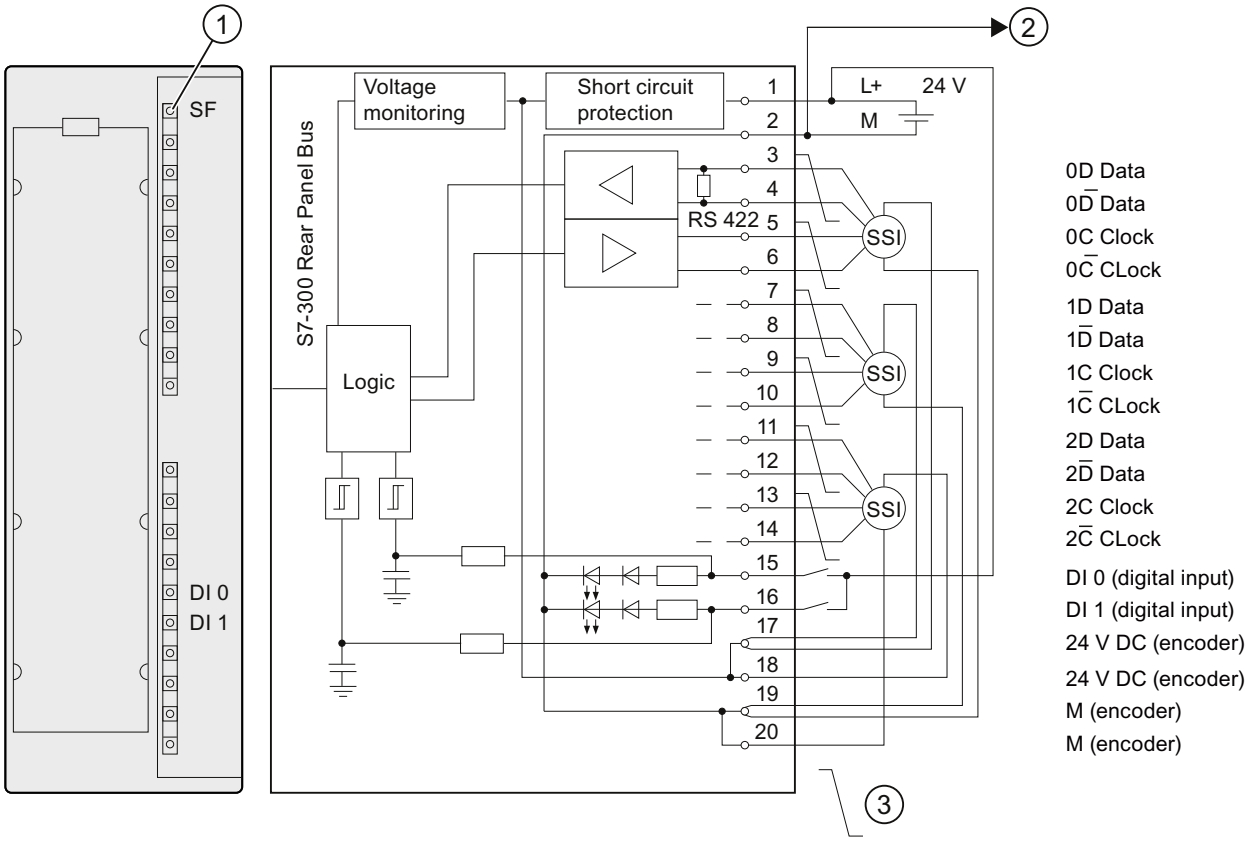

① Error LED - red

② Connection to CPU ground

③ Twisted-pair cables

6.2.2.2 Configuring SM 338 (S7-300, S7-400)

6.2.2.3 Assigning parameters for SM 338 (S7-300, S7-400)

Operating mode (S7-300, S7-400)

Enabling Fast Mode (S7-300, S7-400)

Fast Mode

Fast Mode is enabled in the respective check box.

In Fast Mode, the module operates with faster encoder value detection and compressed feedback interface. The Freeze function is not possible.

Isochronous mode (S7-300, S7-400)

Hardware requirements

To operate SM 338 in isochronous mode, you need:

- a CPU which supports isochronous mode
- a DP master which supports the constant bus cycle
- Interface module IM 153-x, which supports isochronous mode

Properties of the SM 338 in isochronous mode

SM 338 operates in non-isochronous or isochronous mode, depending on system parameters.

In isochronous mode, data are exchanged between the DP master and SM 338 in synchronism to the PROFIBUS DP cycle.

In isochronous mode, all 16 bytes of the feedback interface are consistent.

If synchronism is lost due to disturbance or failure/delay of Global Control (GC), the SM 338 resumes isochronous mode at the next cycle without error reaction.

If synchronism is lost, the feedback interface is not updated.

Note

For basic information on isochronous mode, refer to the Isochronous mode function manual (http://support.automation.siemens.com/WW/view/en/15218045).

Enable (S7-300, S7-400)

Enabling diagnostic interrupt (S7-300, S7-400)

Diagnostic interrupt

The diagnostic interrupts are not preset - in other words, they are inhibited without appropriate enabling.

Diagnostic interrupts are enabled in the respective check box.

If you have enabled diagnostic interrupts, the incoming error events (initial occurrence of the error) and outgoing error events (message after troubleshooting) are reported by means of interrupts (Page 2362).

The CPU interrupts user program execution, and executes diagnostic interrupt OB.

The diagnostic information is consistent until such time as diagnostic interrupt OB is exited. When diagnostic interrupt OB is exited, the diagnostic interrupt is acknowledged on the module.

Encoder input n (S7-300, S7-400)

Selecting the encoder type (S7-300, S7-400)

Encoder type

The SM 338 supports the following encoder types:

- Absolute value encoder (SSI), frame length 13 bit
- Absolute value encoder (SSI), frame length 21 bit
- Absolute value encoder (SSI), frame length 25 bit

The encoder type can be found in the technical data of the absolute value encoder used.

The selection "No encoder" switches off the encoder input.

Selecting the code type (S7-300, S7-400)

Code type

The parameter "Code type" is used to determine the handling of the code supplied by the encoder:

- When "Gray" is set, the encoder value supplied by the absolute encoder in gray code is converted to dual code.
- When "Binary" is set, the supplied encoder value is not converted.

Selecting the transmission rate (S7-300, S7-400)

Transmission rate

With the "Transmission rate", you can determine the data transmission rate from the SSI encoder to the SM 338.

Select the desired transmission rate in the drop-down list.

Observe the relationship between line length and transmission rate:

- up to 320 m → 125 kHz
- up to 160 m → 250 kHz
- up to 60 m → 500 kHz
- up to 20 m → 1 MHz

Selecting monoflop time (S7-300, S7-400)

Monoflop time

The monoflop time is the break interval between 2 SSI message frames.

The configured monoflop time must be greater than the monoflop time of the absolute value encoder used (see manufacturer's technical data).

The time 2 × (1 / transmission rate) is added to the configured value. With a transmission rate of 125 kHz and a configured monoflop time of 16 µs, the actual effective monoflop time is 32 µs.

The following limitation applies to the monoflop time of the absolute value encoder:

(1 / transmission rate) < monoflop time of absolute value encoder < 64 µs + 2 × (1 / transmission rate)

Select the corresponding monoflop time in the drop-down list.

The specification of the monoflop time is relevant to freewheeling encoder value acquistion.

Entering number of adjusted bits (S7-300, S7-400)

Number of adjusted bits [0..12]

SM 338 requires the number of adjusted bits to determine the encoder position and to correctly detect the direction of movement of the encoder.

This value can be found in the technical data of the absolute value encoder used.

Selecting steps/revolution (S7-300, S7-400)

Steps/revolution

Up to 13 bits are available for the steps/revolution. The resulting number of steps/revolutions is automatically calculated according to the information "Number of adjusted bits (Page 2357)".

Selecting the digital input for the FREEZE function (S7-300, S7-400)

FREEZE function

The FREEZE function "freezes" the actual encoder values of SM 338. The FREEZE function is coupled to the digital inputs DI 0 and DI 1 of SM 338.

The freeze function is triggered by a signal transition (positive edge) at digital input DI 0 or DI 1. Bit 31 = 1 (output address) identifies a frozen encoder value. You can freeze one, two or three encoder values using one digital input.

To allow their event-triggered evaluation, the encoder values are retained until the FREEZE function is terminated.

Enabling the FREEZE function

The FREEZE function must be enabled during parameter assignment.

To do so, select the digital input at which a rising edge freezes the encoder value:

- off (encoder value is not frozen)
- DI 0
- DI 1

Canceling the FREEZE function

The FREEZE function must be terminated separately at each encoder input. You acknowledge the function in the user program by using the operation T POB "xyz" to set bit 0, 1 or 2 according to the encoder input (see the program example (Page 2360)).

The acknowledgement resets bit 31 of the corresponding encoder value, and initiates a refresh of the encoder values. You can freeze the encoder values again after you cleared the ACK bit at the output address of the module.

In isochronous mode, the acknowledgement is processed at the time T_o. From this point on, you can freeze the encoder values again by setting the digital inputs.

Note

The FREEZE function is automatically acknowledged if you reconfigure the corresponding encoder input with different parameters.
I The freeze function will be unaffected if the parameters are identical.

Addressing SM 338 (S7-300, S7-400)

Data areas for encoder values

The SM 338 inputs and outputs are addressed beginning at the module start address. The input and output address is determined during the configuration of the SM 338.

Input addresses

The table below shows the input addresses of the SM 338.

Encoder input	Input address (derived from configuration) + address offset
0	"Module start address"
1	"Module start address" + 4 byte address offset
2	"Module start address" + 8 byte address offset

Structure of double data word in Standard Mode

The figure below shows the structure of the double data word for each encoder input in standard mode.

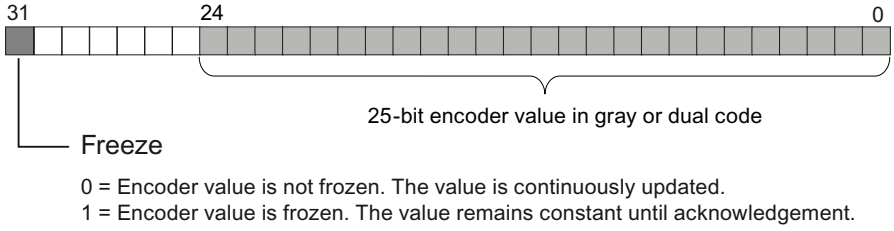

Structure of double data word in Fast Mode

The figure below shows the structure of the double data word for each encoder input in fast mode.

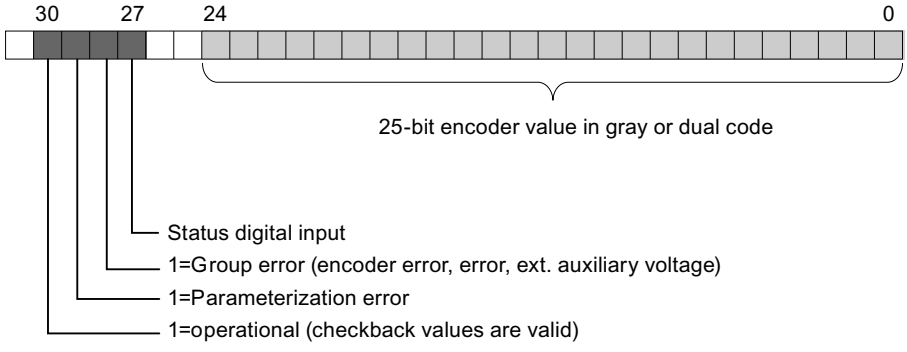

The status of digital input DI0 is reported in bit 27 (digital input status) in the double data word from encoder input 0, and the status of digital input DI1 in the double data word from encoder input 1.

In the double data word from encoder input 2, bit 27 is always = 0.

Output data in standard mode

The figure below shows the structure of the output data in standard mode.

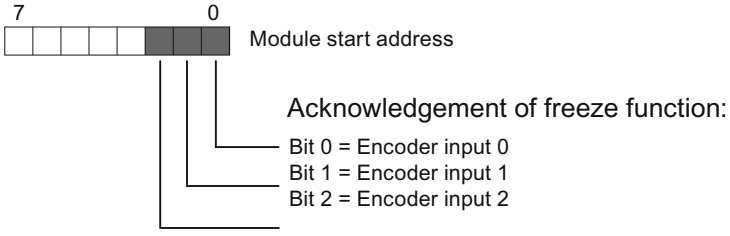

no output data are supported in Fast Mode.

Reading data areas

You can read the data areas in your user program using the operation L PID "xyz."

Program example of access to encoder values and use of the FREEZE function

You want to read and evaluate the values at the encoder inputs. The module start address is 256.

```
STL                 Explanation
L PED256            // Read encoder value in the address area for encoder input 0

T MD 100            // Save encoder value to memory double word
UD W#16#1FFFFFF     // smooth the lower 25 bits to completely hide the upper 7 bits
T MD 120            // Encoder value without FREEZE or other status bits can also
                    be used in fast mode, since only 25 bits are displayed
U M 100.7           // FREEZE state for subsequent acknowledgement
= M 99.0            // determine and save

L PED 260           // Read encoder value in the address area for encoder input 1

T MD 104            // Save encoder value to memory double word
UD W#16#1FFFFFF     // smooth the lower 25 bits to completely hide the upper 7 bits
T MD 124            // Encoder value without FREEZE or other status bits can also
                    be used in fast mode, since only 25 bits are displayed
U M 104.7           // FREEZE state for subsequent acknowledgement
= M 99.1            // determine and save

L PED 264           // Read encoder value in the address area for encoder input 2

T MD 108            // Save encoder value to memory double word
```

```
STL                 Explanation
UD W#16#1FFFFFF     // smooth the lower 25 bits to completely hide the upper 7 bits
T MD 128            // Encoder value without FREEZE or other status bits can also
                    be used in fast mode, since only 25 bits are displayed
U M 108.7           // FREEZE state for subsequent acknowledgement
= M 99.2            // determine and save

L MB 99             // Load FREEZE state and
T PAB 256           // acknowledge (SM 338: output address 256)
```

You can then process the encoder values from the bit memory address areas MD 100, MD 104 and MD 108. The encoder value is set in bits 0 to 30 of the memory double word.

6.2.2.4 SM 338: Online & Diagnostics (S7-300, S7-400)

Diagnostics of the SM 338 (S7-300, S7-400)

The SM 338 provides diagnostics messages, i.e. it always provides all diagnostics messages without user intervention.

Measures after a diagnostic message

Actions initiated by diagnostic messages:

- The diagnostic message is entered in the diagnosis of the module and forwarded to the CPU.
- The SF LED on the module is lit.
- If you configured "Enable Diagnostic Interrupt (Page 2356)", the system triggers a diagnostic interrupt and calls the diagnostic interrupt OB.

Reading diagnostic messages

You can read detailed diagnostic messages using instructions in the user program.

The error cause can be displayed in the module diagnostics.

Diagnostic message using the SF LED

The SM 338 indicates errors at its SF LED (group error LED.) The SF LED lights up when the SM 338 generates a diagnostic message. It goes dark after all error states are cleared.

The SF LED also lights up to indicate external errors (short circuit at the encoder supply), regardless of the CPU operating state (at POWER ON.)

The SF LED lights up temporarily at startup during the self-test of SM 338.

Diagnostic messages of SM 338

The table below shows the diagnostic messages of SM 338.

Diagnostics message	LED	Scope of diagnostics
Module error	SF	Module
Internal error	SF	Module
External error	SF	Module
Channel error	SF	Module
External auxiliary voltage missing	SF	Module
Module parameters not assigned	SF	Module
Incorrect parameters	SF	Module
Channel information available	SF	Module
Watchdog time-out	SF	Module
Channel error	SF	Channel (encoder input)
Configuration / programming error	SF	Channel (encoder input)
External channel error (encoder error)	SF	Channel (encoder input)

Causes of error and troubleshooting

The table below shows error causes and troubleshooting for diagnostic messages of SM 338.

Diagnostics message	Possible cause of error	To correct or avoid error
Module error	Any, the module has detected an error.	
Internal error	Module has detected an error within the automation system.	
External error	Module has detected an error outside of the automation system.	
Channel error	Indicates that only specific channels are faulty.	
External auxiliary voltage missing	Supply voltage L+ of the module is missing	Connect supply L+
Module parameters not assigned	Module requires information whether it should operate with default system parameters or user parameters.	Message present after POWER ON, until the CPU has completed the transfer of parameters; configure the module as required.
Incorrect parameters	One parameter, or the combination of parameters, is not plausible	Program the module
Channel information available	Channel error; module can provide additional channel information.	
Watchdog time-out	Infrequent high electromagnetic interference	Eliminate interference
Channel error	Any, the module has detected an error at the encoder input.	
Configuration / programming error	Illegal parameter transferred to module	Program the module
External channel error (encoder error)	Wire-break at encoder cable, encoder cable not connected, or encoder defective.	Check connected encoder

6.2.3 Using ET 200S 1SSI (S7-300, S7-400)

6.2.3.1 Terminal Assignment of the 1SSI (S7-300, S7-400)

Wiring Rules

The cables (terminals 1 and 5 and terminals 4 and 8) must be shielded, twisted-pair cables. The shield must be supported at both ends. To do this, use the shield contact (see the appendix of the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal assignment of the 1SSI

The following table shows the terminal assignment of 1SSI.

View	Terminal assignment	Remarks	
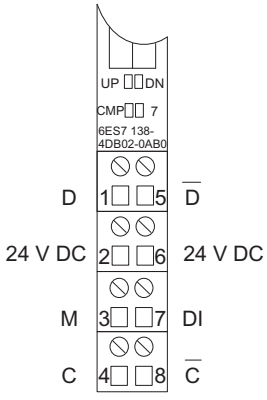	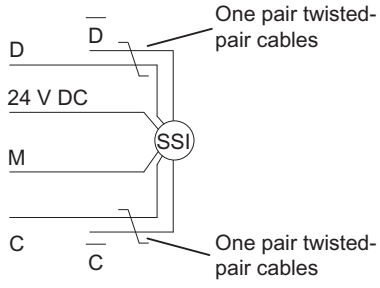	Terminals 1 to 8:	
		1/5:	Data from the SSI encoder [1]
		2/6:	Power supply for absolute value encoder and switch [2]
		3:	Chassis ground
		7:	Digital input latch function
		4/8:	SSI clock (clock wire) [1]

View	Terminal assignment	Remarks
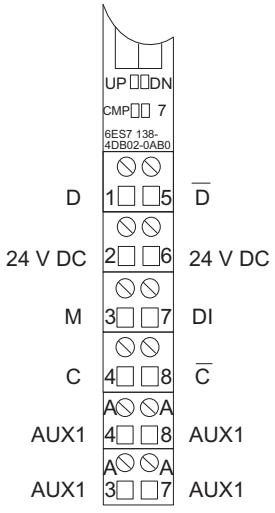	TM-E15S26-A1 and EM 1SSI 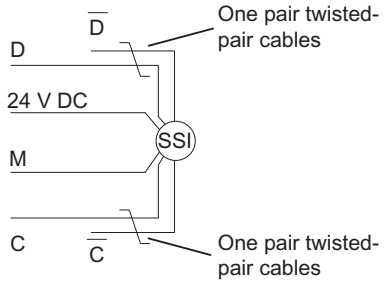	

[1] It is essential that you maintain the correct polarity when wiring the encoder. If you do not, an absolute encoder error is reported. Signals to RS 422

[2] Short circuit-proof, maximum 0.5 A.

6.2.3.2 Configuring and assigning parameters to 1SSI (S7-300, S7-400)

Operation (S7-300, S7-400)

Selecting an operating mode (S7-300, S7-400)

Operating mode

You define 1SSI functionality by specifying an operating mode.

Operating mode	Description
Standard Mode (Page 2366)	1SSI standard operating mode with complete functional scope: • Scaling the encoder value • Direction reversal to adapt the direction of motion of the absolute encoder to the axis. • Encoder value acquisition modes: – freewheeling – synchronous to the update rate • Maximum encoder sampling rate (e.g. for ultrasonic encoders) is taken into account in isochrone mode • Lifebeat in isochrone mode • Parity check of encoder value can be performed • Gray code/dual code converter • Latch function for freezing the current encoder value • Compare function for comparing the current encoder value and loadable comparison values This operating mode is suitable for monitoring or detecting position points and for length measurement, edge detection and synchronization with workpieces.
Fast mode (Page 2378)	The 1SSI has compressed functionality in this operating mode. The latch and compare functions are not available. This operating mode is suitable for rapid encoder value acquisition and controller applications such as position controllers with the position as the actual value.

Standard mode parameters (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting encoder value acquisition (S7-300, S7-400)

Description

The following alternatives are available for encoder value acquisition:

- freewheeling encoder value acquisition
- synchronous encoder value acquisition
- Isochrone encoder value acquisition

You can set cyclic or synchronous encoder value acquisition with the "Detection" parameter in parameter assignment. This parameter only works in non-isochrone mode.

Encoder value acquisition will be isochrone when the 1SSI is operating in isochrones mode. In this case, the "Detection" parameter is not evaluated.

The following table sets out these encoder value acquisition conditions.

Operation	"Detection" parameter	Encoder value acquisition
Non-isochrone mode	freewheeling	freewheeling encoder value acquisition
	Synchronous	synchronous encoder value acquisition
Isochrone mode	- (irrelevant)	Isochrone encoder value acquisition

Cyclic encoder value acquisition

You obtain maximum accuracy when latching the encoder value (Page 2371) during freewheeling encoder value acquisition.

The 1SSI initiates the transmission of a frame each time the parameterized monoflop time (Page 2369) elapses.

The 1SSI processes the acquired encoder value asynchronously to these freewheeling frames in the update rate cycle.

Freewheeling acquisition therefore returns encoder values of different ages. The difference between the maximum and minimum age is the jitter.

Synchronous encoder value acquisition

Synchronous encoder value acquisition gives you maximum accuracy.

The 1SSI initiates the transmission of a frame in the cycle of its update rate.

The 1SSI processes the transferred encoder value synchronously with its update rate.

Selecting encoder type (S7-300, S7-400)

Encoder type

The 1SSI supports the following encoder types:

- Absolute encoder (SSI) with 13 bits
  to
- Absolute encoder (SSI) with 25 bits

You will find the encoder type in the technical specifications for your absolute encoder.

Selecting "No encoder" deactivates the encoder input.

Selecting Gray code/dual code converter (S7-300, S7-400)

Gray code/dual code converter

The "Gray code/dual code converter" parameter defines how the codes returned by the encoder are processed:

- When "Gray code" is set, the encoder value returned by the absolute encoder in gray code will be converted to dual code.
- When "Dual code" is set, the values returned by the encoder will remain unchanged.

Selecting the transmission rate (S7-300, S7-400)

Transmission rate

You define the data transmission rate between the SSI encoder and the 1SSI with the "transmission rate".

Select the desired transmission rate in the drop-down box.

Note that the transmission rate affects how accurate and up-to-date the encoder values are.

The maximum transmission rate depends on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 60 m → 500 kHz
- 20 m → 1 MHz
- 10 m → 1.5 MHz
- 8 m → 2 MHz

See also

Selecting monoflop time (Page 2369)

Selecting parity (S7-300, S7-400)

Parity

With "Parity", you specify whether no parity, even parity or odd parity should be assigned for an absolute encoder.

A configured parity bit is not included in the "Encoder Type (Page 2367)" parameter. If a 25-bit encoder with parity is parameterized, the encoder will read 26 bits.

The bit following the LSB (least significant bit) of the encoder is evaluated as a parity bit. A parity error is reported in the feedback interface (Page 2377) via the bit ERR_SSI.

Selecting monoflop time (S7-300, S7-400)

Monoflop time

The monoflop time represents the interval between two SSI frames.

The parameterized monoflop time must be greater than the monoflop time of the absolute encoder used (refer to the technical specifications of the manufacturer).

The following restriction applies to the monoflop time of the absolute encoder:

(1/transmission rate) < monoflop time of absolute encoder < 64 µs

Select the desired monoflop time in the drop-down box.

The specification of the monoflop time is relevant for freewheeling encoder value acquisition (Page 2366).

See also

Selecting the transmission rate (Page 2367)

Activating scaling (S7-300, S7-400)

Encoder value and scaling

The encoder value transferred contains the encoder position of the absolute encoder. In addition to the encoder position, the encoder transfers additional bits located before and after the encoder position, depending on the encoder used.

The 1SSI requires the following information to allow it to detect the encoder position:

- Encoder type (Page 2367)
- Number of adjusted bits (Page 2370)
- Increments per encoder revolution (Page 2371)

Scaling specifies presentation of the encoder value at the feedback interface (Page 2377).

- "Scaling **on**" specifies that trailing, irrelevant bits in the encoder value are to be discounted.
- "Scaling **off**" specifies that adjusted bits are retained and are available for evaluation.

Entering number of adjusted bits (S7-300, S7-400)

Number of adjusted bits

The 1SSI requires the number of adjusted bits in order to detect the encoder position and correctly detect the encoder's direction of motion.

You will find this value in the technical specifications for your absolute encoder.

Enabling direction reversal (S7-300, S7-400)

Direction detection and direction reversal

The 1SSI needs the following information to detect the direction of movement of the encoder correctly:

- Encoder type (Page 2367)
- Number of adjusted bits (Page 2369)
- Increments per encoder revolution (Page 2371)

The direction of motion determined is displayed in the feedback interface (Page 2377) and by the LEDs.

- LED UP: Encoder position change from lower to higher value
- LED DN: Encoder position change from higher to lower value

Direction reversal adapts the encoder direction of motion to the axis.

Two settings are possible:

- **Off:** The direction of the transferred encoder position is maintained.
- **On:** The direction of the transferred encoder position is reversed. This means that although the encoder sends ascending values, descending values are displayed.
  The reversal relates to the increments/encoder revolution set in the parameters.

Example of Direction Reversal

Presettings:

You are using a single-turn encoder with 2^{10} (corresponds to 10 bits) = 1024 increments/ encoder revolution (resolution/360°) with the following parameter assignment:

- Encoder type: SSI-13 bit
- Number of adjusted bits: 3 places

- Direction reversal: On
- Increments per encoder revolution: 1024

Encoder value before direction reversal: cyclically recorded encoder position 1023

Encoder value after direction reversal: displayed encoder position 0

Entering increments per encoder revolution (S7-300, S7-400)

Increments per encoder revolution

The "Increments per encoder revolution" parameter defines the number of increments an encoder returns per revolution.

You will find this value in the technical specifications for your absolute encoder.

Valid value ranges

The limits depend on the individual encoder models. Only values to the power of 2 are permitted.

- 13-bit encoder type: 16 to 8192
- 14-bit encoder type: 16 to 16384
- 15-bit encoder type: 16 to 32768
- 16-bit encoder type: 16 to 65536
- 17-bit encoder type: 16 to 131072
- 18-bit encoder type: 16 to 262144
- 19-bit encoder type: 16 to 524288
- 20-bit encoder type: 16 to 1048576
- 21-bit encoder type: 16 to 2097152
- 22-bit encoder type: 16 to 4194304
- 23-bit encoder type: 16 to 8388608
- 24-bit encoder type: 16 to 16777216
- 25-bit encoder type: 16 to 33554432

Selecting encoder value latching (S7-300, S7-400)

Latching the encoder value

You use the latch function to "freeze" the current encoder value of the 1SSI with an edge at the digital input (DI).

The encoder value can thus be evaluated on an event-dependent basis.

A frozen encoder value is identified by the set bit 31 and is retained until the latch function is exited.

The frozen encoder value is entered at the feedback interface (Page 2377) at the position of the otherwise cyclically recorded value and assigned the identifier "Bit 31 set".

Note

Direction determination, comparison, and error monitoring also take place when the encoder value is frozen.

Prerequisites for Using the Latch Function

You specify which edge at the digital input is to freeze the encoder value using parameter assignment.

- Falling edge at DI
- Rising edge at DI
- Both edges at DI

Selecting an edge activates the latch function coupled to the digital input.

The encoder value cannot be frozen if "Inactive" is selected.

Terminating the Latch Function

The latch function must be acknowledged. When the controller program acknowledges the acceptance of the encoder value, bit 31 is deleted and the encoder value is updated again. Freezing is then possible again.

The figure below shows the latch function process.

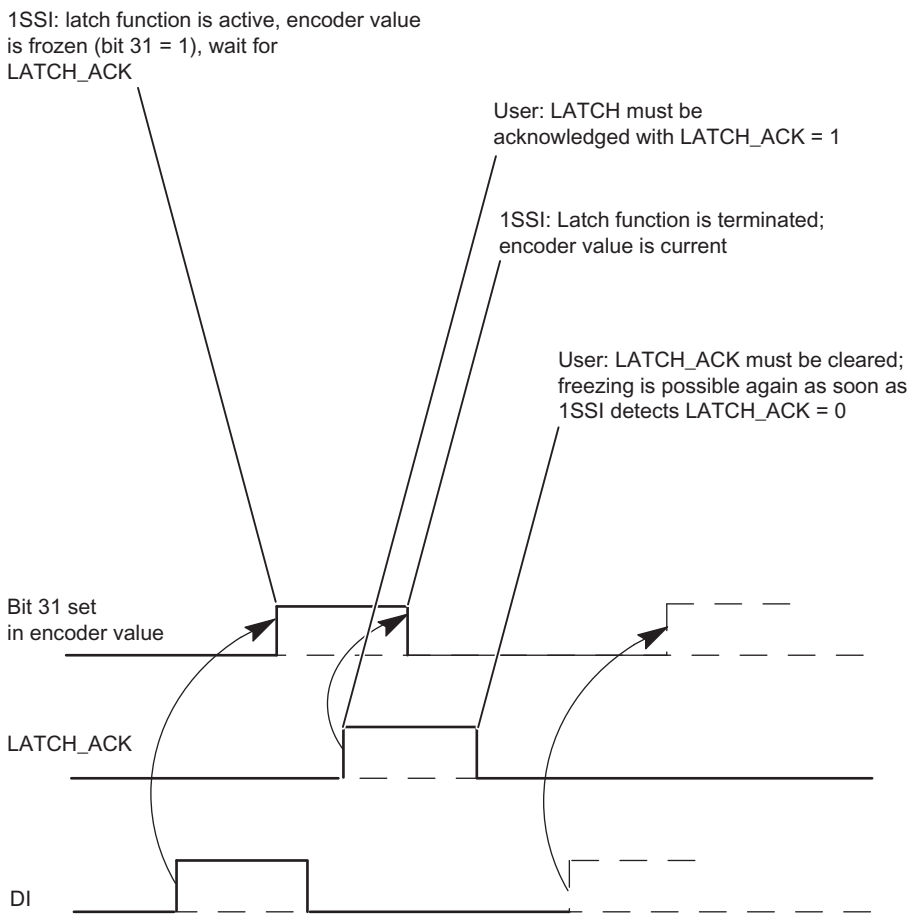

Selecting comparator behavior (S7-300, S7-400)

Setting comparators

The encoder position detected can be compared with up to two loadable values. Both comparison results are stored in the feedback interface (Page 2377). The appropriate comparator becomes active only after the comparison value is loaded.

You set the two comparators in the "Comparator 1" and "Comparator 2" parameters:

Setting	Effect on the result of comparison (CMPx)
inactive	The encoder value is not compared. The feedback bit CMPx = 0
UP	The encoder value is compared in the up direction (UP). • If the encoder value ≥ comparison value, feedback bit CMPx = 1. • If the encoder value < comparison value, feedback bit CMPx = 0. • If the direction is down, the feedback bit CMPx remains unchanged. • If no change is detected in the encoder value, the feedback bit CMPx will remain unchanged.

Setting	Effect on the result of comparison (CMPx)
Down	The encoder value is compared in the down direction (DN). • If the encoder value ≤ comparison value, feedback bit CMPx = 1. • If the encoder value > comparison value, feedback bit CMPx = 0. • If the direction is up, the feedback bit CMPx will remain unchanged. • If no change is detected in the encoder value, the feedback bit CMPx will remain unchanged.
In both directions	The encoder value is compared in both directions. If the direction is up, the following conditions apply: • If the encoder value ≥ comparison value, feedback bit CMPx = 1. • If the encoder value < comparison value, feedback bit CMPx = 0. • If no change is detected in the encoder value, the feedback bit CMPx will remain unchanged. If the direction is down, the following conditions apply: • If the encoder value ≤ comparison value, feedback bit CMPx = 1. • If encoder value > comparison value, feedback bit CMPx = 0. • If no change is detected in the encoder value, the feedback bit CMPx will remain unchanged.

The comparison result is deleted and entered in accordance with the directional setting as soon as you load a comparison value.

Note

Only one control bit can be set at a particular time:

CMP_VAL1 or CMP_VAL2).

Otherwise, the ERR_LOAD error will be reported until both control bits are deleted.

Loading the Comparison Value

The figure below illustrates the value transfer process:

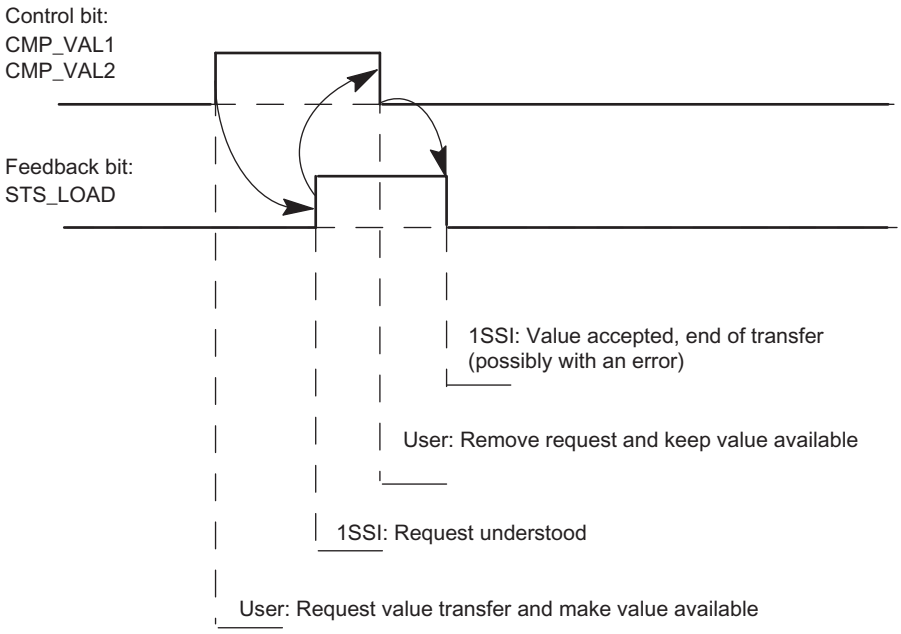

Comparator in Isochronous Mode

In isochronous mode, the comparison values are loaded at time T_o and are effective as of time T_i in the same bus cycle.

Enabling lifebeat (default mode) (S7-300, S7-400)

Lifebeat

This parameter is only active in isochrone mode.

When lifebeat is activated, the lifebeat bit is toggled each time an encoder value is read in in isochronous mode, i.e. the last value sent is inverted. If a reduction is assigned in the "Encoder Sampling Rate (Page 2375)" parameter, the value is only toggled if an encoder value has actually been read in.

The lifebeat bit can be found in the feedback interface (Page 2377) in byte 4 / bit 7.

Setting the encoder sampling rate (S7-300, S7-400)

Encoder sampling rate

This parameter is only active in isochrone mode.

Any encoder sampling rate to be taken into account is set here. This allows slower encoders (such as ultrasound encoders) to be used, even with a fast processing cycle.

An integer reduction n is calculated using the set frequency. The encoder is then only read in again every nth clock cycle.

Example

- Processing cycle 500 µs
- Encoder sampling rate: 1.2 kHz (approximately every 833 µs)

This produces a reduction n = 2.

The encoder is only read in again every second processing cycle, in other words each millisecond.

Control and feedback interface for standard mode (S7-300, S7-400)

Standard mode feedback interface (S7-300, S7-400)

Note

The following control interface data is consistent for 1SSI:

- Bytes 0 to 3
- Bytes 4 to 7

Control interface assignment

The table below shows the assignment of the 1SSI control interface (outputs) in standard mode.

Address	Assignment		Name
Bytes 0 to 3	Comparison value 1 or 2 (double word)		
Byte 4	Bit 7	Error Acknowledgment	EXTF_ACK
	Bit 6	Acknowledgement latch function	LATCH_ACK
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	Load comparison value 2	CMP_VAL2
	Bit 0	Load comparison value 1	CMP_VAL1
Byte 5	Reserved = 0		
Bytes 6 to 7	Reserved = 0		

Notes on the control bits

Bit	Explanation
CMP_VAL1	Load comparison value 1
CMP_VAL2	Load comparison value 2

Bit	Explanation
EXTF_ACK	Error acknowledgement for the "Absolute encoder error" ERR_SSI and "Encoder supply short circuit" ERR_24V.
LATCH_ACK	Acknowledgement for latch function

Standard mode feedback interface (S7-300, S7-400)

Note

The following feedback interface data is consistent for 1SSI:

- Bytes 0 to 3
- Bytes 4 to 7

Feedback interface assignment

The table below shows the assignment of the 1SSI feedback interface (inputs) in standard mode.

Address	Assignment		Name
Bytes 0 to 3	Encoder value (double word) (bit 31 set, encoder value frozen)		
Byte 4	Bit 7	Reserved = 0 or lifebeat	Lifebeat
	Bit 6	Ready	RDY
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Absolute encoder error	ERR_SSI
	Bit 3	Encoder supply short circuit	ERR_24V
	Bit 2	DI status	STS_DI
	Bit 1	Status DN	STS_DN
	Bit 0	Status UP	STS_UP
Byte 5	Bit 7	Reserved = 0	
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Comparison value 2 reached	CMP2
	Bit 2	Comparison value 1 reached	CMP
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Bytes 6 to 7	Reserved = 0		

Notes on the feedback bits

Bit	Explanation
CMP	Comparison result of comparator 1
CMP2	Comparison result of comparator 2

Bit	Explanation
ERR_24V	There is a short circuit in the encoder supply. ERR_24V is reset when the short circuit is eliminated and acknowledged with the EXTF_ACK control bit.
ERR_LOAD	Error while loading the comparison values because both control bits CMP_VAL1 and CMP_VAL2 are set.
ERR_PARA	Incorrect parameter assignment for the ET 200S station. Cause: Increments per encoder revolution not in the valid value range for the encoder type. The parameter bit is cleared when correct parameter assignment is transmitted.
ERR_SSI	The 1SSI detects an "Absolute encoder error" if the frames at the SSI interface are interrupted. Causes: No encoder connected; wire break in the encoder cable; encoder supply undervoltage; encoder type, transmission rate, monoflop time do not correspond to the connected encoder; programmable encoders do not correspond to the settings on the 1SSI; encoder is defective or faults or parity errors exist. ERR_SSI is reset when the cause of the error is eliminated and acknowledged with the EXTF_ACK control bit.
Lifebeat	This parameter is only active in isochrone mode. When lifebeat is on, the lifebeat bit is toggled each time an encoder value is read in in isochronous mode, i.e. the last value sent is inverted. If a reduction is assigned in the "Encoder Sampling Rate" parameter, the value is only toggled if an encoder value has actually been read in.
STS_DI	The bit displays the status of the digital input DI.
STS_DN	Status direction down; for encoder value change from larger to smaller encoder positions (including zero pass)
STS_LOAD	Feedback bit for CMP_VAL1 and CMP_VAL2. The 1SSI uses this bit to indicate that a comparison value is loaded.
STS_UP	Status direction down; for encoder value change from smaller to larger encoder positions (including zero pass)
RDY	1SSI parameter assignment is correct, and the module is executing its functions. The feedback displayed is valid. For absolute value encoder errors, ERR_SSI is also set.

Fast mode parameters (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting encoder value acquisition (S7-300, S7-400)

Description

The absolute encoder transfers its encoder values in frames to the 1SSI. The transmission of frames is initiated by the 1SSI. The following alternatives are available for encoder value acquisition:

- freewheeling encoder value acquisition
- synchronous encoder value acquisition
- Isochrone encoder value acquisition

You can set freewheeling or synchronous encoder value acquisition with the "Detection" parameter in parameter assignment. This parameter only works in non-isochrone mode.

Encoder value acquisition will be isochrone when the 1SSI is operating in isochrones mode. In this case, the "Detection" parameter is not evaluated.

The following table sets out these encoder value acquisition conditions.

Operation	"Detection" parameter	Encoder value acquisition
Non-isochrone mode	freewheeling	freewheeling encoder value acquisition
	Synchronous	synchronous encoder value acquisition
Isochrone mode	- (irrelevant)	Isochrone encoder value acquisition

Cyclic encoder value acquisition

The 1SSI initiates the transmission of a frame each time the parameterized monoflop time (Page 2381) elapses.

The 1SSI processes the acquired encoder value asynchronously to these freewheeling frames in the update rate cycle.

Freewheeling acquisition therefore returns encoder values of different ages. The difference between the maximum and minimum age is the jitter.

Synchronous encoder value acquisition

Synchronous encoder value acquisition gives you maximum accuracy.

The 1SSI initiates the transmission of a frame in the cycle of its update rate.

The 1SSI processes the transferred encoder value synchronously with its update rate.

Isochrone encoder value acquisition

Isochrone encoder value acquisition is automatically set if the bus system and I/O modules are synchronized.

The 1SSI initiates the transmission of a frame in each bus cycle at time T_i as long as the configured maximum encoder sampling rate does not result in a reduction.

The 1SSI processes the transmitted encoder value isochronously to the bus cycle.

Selecting encoder type (S7-300, S7-400)

Encoder type

The 1SSI supports the following encoder types:

- Absolute encoder (SSI) with 13 bits
  to
- Absolute encoder (SSI) with 25 bits

You will find the encoder type in the technical specifications for your absolute encoder.

Selecting "No encoder" deactivates the encoder input.

Selecting Gray code/dual code converter (S7-300, S7-400)

Gray code/dual code converter

The "Gray code/dual code converter" parameter defines how the codes returned by the encoder are processed:

- When "Gray code" is set, the encoder value returned by the absolute encoder in gray code will be converted to dual code.
- When "Dual code" is set, the values returned by the encoder will remain unchanged.

Selecting the transmission rate (S7-300, S7-400)

Transmission rate

You define the data transmission rate between the SSI encoder and the 1SSI with the "transmission rate".

Select the desired transmission rate in the drop-down box.

Note that the transmission rate affects how accurate and up-to-date the encoder values are.

The maximum transmission rate depends on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 60 m → 500 kHz
- 20 m → 1 MHz
- 10 m → 1.5 MHz
- 8 m → 2 MHz

See also

Selecting monoflop time (Page 2381)

Selecting parity (S7-300, S7-400)

Parity

"Parity" allows you to specify whether no parity, even parity or odd parity is to be assigned for an absolute encoder.

A configured parity bit is not included in the "Encoder Type (Page 2379)" parameter. If a 25-bit encoder with parity is parameterized, the encoder will read 26 bits.

The bit following the LSB (least significant bit) of the encoder is evaluated as a parity bit. A parity error is reported in the feedback interface (Page 2384) via the bit EXTF.

Selecting monoflop time (S7-300, S7-400)

Monoflop time

The monoflop time represents the interval between two SSI frames.

The parameterized monoflop time must be greater than the monoflop time of the absolute encoder used (refer to the technical specifications of the manufacturer).

The following restriction applies to the monoflop time of the absolute encoder:

(1/transmission rate) < monoflop time of absolute encoder < 64 µs

Select the desired monoflop time in the drop-down box.

The specification of the monoflop time is relevant for freewheeling encoder value acquisition (Page 2378).

See also

Selecting the transmission rate (Page 2379)

Activating scaling (S7-300, S7-400)

Encoder value and scaling

The encoder value transferred contains the encoder position of the absolute encoder. In addition to the encoder position, the encoder transfers additional bits located before and after the encoder position, depending on the encoder used.

The 1SSI requires the following information to allow it to detect the encoder position:

- Encoder type (Page 2379)
- Number of adjusted bits (Page 2382)
- Increments per encoder revolution (Page 2383)

Scaling specifies presentation of the encoder value at the feedback interface (Page 2384).

- "Scaling **on**" specifies that trailing, irrelevant bits in the encoder value are to be discounted.
- "Scaling **off**" specifies that adjusted bits are retained and are available for evaluation.

Entering number of adjusted bits (S7-300, S7-400)

Number of adjusted bits

The 1SSI requires the number of adjusted bits in order to detect the encoder position and correctly detect the encoder's direction of motion.

You will find this value in the technical specifications for your absolute encoder.

Enabling direction reversal (S7-300, S7-400)

Direction detection and direction reversal

The 1SSI needs the following information to detect the direction of movement of the encoder correctly:

- Encoder type (Page 2379)
- Number of adjusted bits (Page 2381)
- Increments per encoder revolution (Page 2383)

The direction of motion determined is displayed in the feedback interface (Page 2384) and by the LEDs.

- LED UP: Encoder position change from lower to higher value
- LED DN: Encoder position change from higher to lower value

Direction reversal adapts the encoder direction of motion to the axis.

Two settings are possible:

- **Off:** The direction of the transferred encoder position is maintained.
- **On:** The direction of the transferred encoder position is reversed. This means that although the encoder sends ascending values, descending values are displayed.
  The reversal relates to the increments/encoder revolution set in the parameters.

Example of Direction Reversal

Presettings:

You are using a single-turn encoder with 2^{10} (corresponds to 10 bits) = 1024 increments/encoder revolution (resolution/360°) with the following parameter assignment:

- Encoder type: SSI-13 bit
- Number of adjusted bits: 3 places

- Direction reversal: On
- Increments per encoder revolution: 1024

Encoder value before direction reversal: cyclically recorded encoder position 1023

Encoder value after direction reversal: displayed encoder position 0

Entering increments per encoder revolution (S7-300, S7-400)

Increments per encoder revolution

The "Increments per encoder revolution" parameter defines the number of increments an encoder returns per revolution.

You will find this value in the technical specifications for your absolute encoder.

Valid value ranges

The limits depend on the individual encoder models. Only values to the power of 2 are permitted.

- 13-bit encoder type: 16 to 8192
- 14-bit encoder type: 16 to 16384
- 15-bit encoder type: 16 to 32768
- 16-bit encoder type: 16 to 65536
- 17-bit encoder type: 16 to 131072
- 18-bit encoder type: 16 to 262144
- 19-bit encoder type: 16 to 524288
- 20-bit encoder type: 16 to 1048576
- 21-bit encoder type: 16 to 2097152
- 22-bit encoder type: 16 to 4194304
- 23-bit encoder type: 16 to 8388608
- 24-bit encoder type: 16 to 16777216
- 25-bit encoder type: 16 to 33554432

Enabling lifebeat (Fast mode) (S7-300, S7-400)

Lifebeat

This parameter is only active in isochrone mode.

When lifebeat is activated, the lifebeat bit is toggled each time an encoder value is read in in isochronous mode, i.e. the last value sent is inverted. If a reduction is assigned in the "Encoder sampling rate (Page 2384)" parameter, the value is only toggled if an encoder value has actually been read in.

The lifebeat bit can be found in the feedback interface (Page 2384) in bytes 0 to 3 / bit 31.

Setting the encoder sampling rate (S7-300, S7-400)

Encoder sampling rate

This parameter is only active in isochrone mode.

Any encoder sampling rate to be taken into account is set here. This allows slower encoders (such as ultrasound encoders) to be used, even with a fast processing cycle.

An integer reduction n is calculated using the set frequency. The encoder is then only read in again every nth clock cycle.

Example

- Processing cycle 500 µs
- Encoder sampling rate: 1.2 kHz (approximately every 833 µs)

This produces a reduction n = 2.

The encoder is only read in again every second processing cycle, in other words each millisecond.

Fast mode feedback interface (S7-300, S7-400)

Feedback interface assignment

The table below shows the assignment of the 1SSI feedback interface (inputs) in fast mode.

Address	Assignment		Name
Bytes 0 to 3	Bit 31	Reserved = 0 or lifebeat	Lifebeat
	Bit 30	Ready for operation (feedback is valid)	RDY
	Bit 29	Parameter assignment error	ERR_PARA
	Bit 28	Group error , absolute encoder or encoder supply short circuit	EXTF
	Bit 27	DI status	STS_DI
	Bit 26	Status DN	STS_DN
	Bit 25	Status UP	STS_UP
	Bits 0 to 24	Encoder value	

Notes on the feedback bits

Bit	Explanation
ERR_PARA	Incorrect parameter assignment for the ET 200S station. Cause: Increments per encoder revolution not in the valid value range for the encoder type. The parameter bit is cleared when correct parameter assignment is transmitted.
EXTF	Group error absolute encoder or encoder supply short circuit Causes: • The encoder supply has short-circuited • No encoder connected; wire break in the encoder cable; encoder type, transmission rate, monoflop time do not correspond to the connected encoder; programmable encoders do not correspond to the settings on the 1SSI; encoder is defective or faults or parity errors exist. EXTF is reset when the causes of the errors have been eliminated.
Lifebeat	This parameter is only active in isochrone mode. When lifebeat is on, the lifebeat bit is toggled each time an encoder value is read in in isochronous mode, i.e. the last value sent is inverted. If a reduction is assigned in the "Encoder Sampling Rate" parameter, the value is only toggled if an encoder value has actually been read in.
STS_DI	The bit displays the status of the digital input DI.
STS_DN	Status direction down; for encoder value change from larger to smaller encoder positions (including zero pass)
STS_UP	Status direction down; for encoder value change from smaller to larger encoder positions (including zero pass)
RDY	1SSI parameter assignment is correct, and the module is executing its functions. The feedback displayed is valid. For absolute value encoder errors, EXTF is also set.

6.2.3.3 Diagnostics of the 1SSI (S7-300, S7-400)

Diagnostics using the LED display (S7-300, S7-400)

LED display on the 1SSI

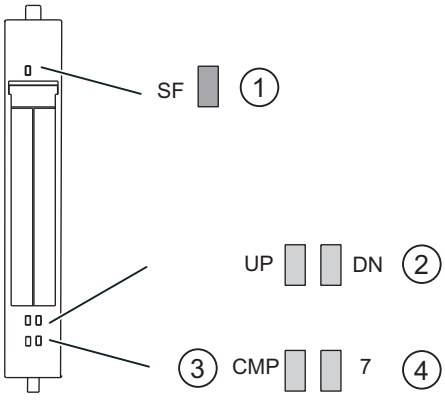

① Group error (red)
② Status display for change in sensor value (green)
③ Status display for result of comparison
④ Status display for digital input (green)

Status and error displays by means of LEDs on the 1SSI

The table below shows the status and error displays on the 1SSI.

Event (LED)					Cause	Remedy
SF	UP	DN	CMP	7		
On					No parameter assignment. There is a diagnostic message.	Check the parameter assignment. Evaluate the diagnostics data.
	On				At value change from smaller to larger sensor values (including zero-crossing)	
		On			At value change from larger to smaller sensor values (including zero-crossing)	
			On		Set for comparison result CMP 1	
				On	DI (latch) is activated.	

Error types (S7-300, S7-400)

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

1SSI error types

The following table shows the error types on the 1SSI.

Error class		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the supply to the absolute encoder.	Correct the process wiring.
9_D	01001: Error	Internal module error occurred.	Replace the module.
		Load voltage from the power module is too low.	Correct the process wiring. Check the load voltage.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.
26_D	11010: External error	Start/Stop bit error (error absolute value encoder) Wire break in the encoder cable or encoder cable is not connected. Encoder type, transmission rate, monoflop time do not correspond to the connected encoder Programmable encoders do not correspond to the settings on the 1SSI Sensor is defective or there are faults.	Replace the encoder Correct the process wiring Correct the parameter assignment.

6.2.4 Basics of positioning with function modules (S7-300, S7-400)

6.2.4.1 What is positioning? (S7-300, S7-400)

Positioning

Positioning is taking up a specific spatial position, in particular a position defined by coordinates.

In SIMATIC technology, positioning means adjusting and positioning mechanical axes. A range of difference procedures are used:

- Controlled positioning with rapid traverse/creep speed
- Controlled positioning
- Positioning with stepper motor

Positioning can control linear axes with a limited traversing range and continuous rotary axes.

6.2.4.2 Controlled positioning with rapid traverse/creep speed (S7-300, S7-400)

Controlled positioning

Each positioning process is characterized by:

- A starting position
- The target of the positioning
- Parameters that determine the sequence of the positioning

The target is initially approached at a higher velocity (rapid traverse). At a specified distance from the target, the velocity switches to creep speed. The drive is switched off shortly before the axis reaches the target - also at a specified distance from the target. In doing so, the module monitors the target approach.

The drive is controlled via digital outputs with rapid traverse or creep speed and in the appropriate direction.

6.2.4.3 Ranges and switching points of the positioning module (S7-300, S7-400)

Target

The target is the absolute or relative position on the axis that is approached during a positioning process.

Definition of the switching points and switching ranges

The following ranges and positions can be parameterized for controlled positioning:

Range / Position	Description
Operating range	defines the range you set for your job by means of the software limit switches or the end of the rotary axis.
Switch-over difference	defines the distance from the target at which the drive changes from rapid traverse to creep traverse.
Switch-over point	Defines the position at which the drive changes from rapid traverse to creep speed.
Switch-off difference	defines the distance from the target at which the drive is switched off.
Switch-off point	Defines the position at which the drive is switched off. The positioning module then adopts the monitoring functions from this point on.
Target range	defines the positioning precision for your application and surrounds the target symmetrically.
Standstill range	Defines a symmetrical range around the target to be monitored by the positioning module.

Switching points and switching differences

The following figure shows how the switching points and switching differences can be arranged for positioning. In the interests of simplicity, it is assumed that the actual speed changes linearly over the distance traversed. The emerging ramps can be explained by mechanical inertia or by the parameterization possibilities of the power section.

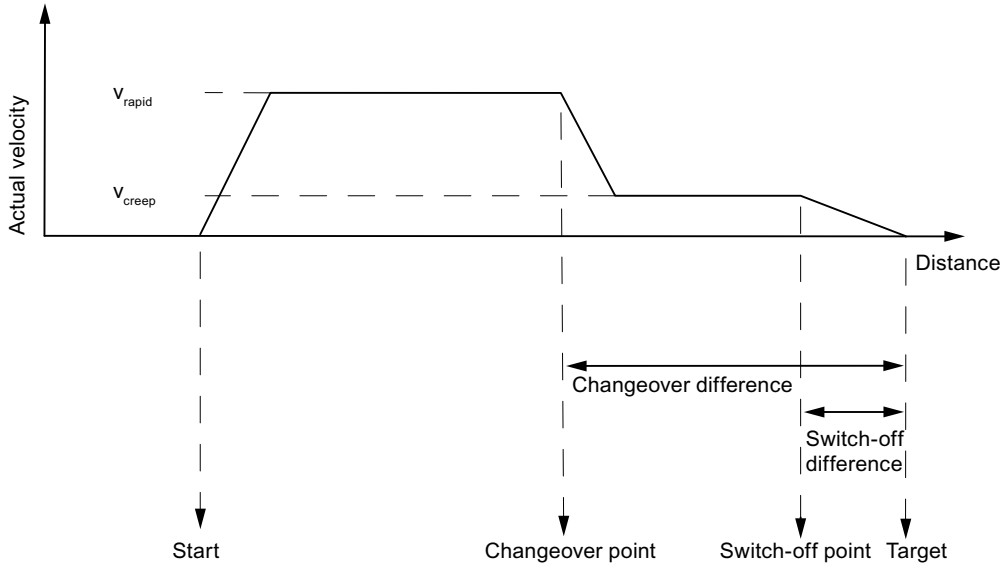

Switching ranges

The figure below shows how the switching ranges can be arranged around the target.

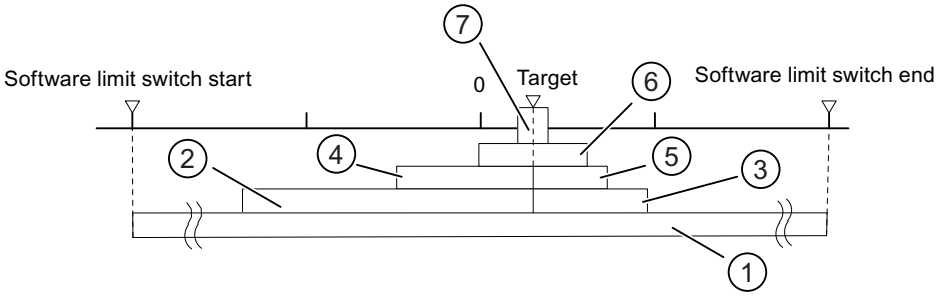

① Operating range
② Changeover difference in direction plus
③ Changeover difference in direction minus
④ Switch-off difference in plus traverse direction
⑤ Switch-off difference in minus traverse direction
⑥ Standstill range
⑦ Target range

6.2.4.4 Encoder signals (S7-300, S7-400)

Supported encoders (S7-300, S7-400)

Incremental encoders supported

Incremental encoders (Page 2391) with two pulses 90° out of phase with or without zero marks are supported:

- Encoders with asymmetrical output signals with 24 V level
  - Limit frequency = 50 kHz
  - Cable up to 100 m long, shielded
- Encoders with symmetrical output signals with 5 V differential interface in accordance with RS 422
  - Limit frequency = 400 kHz
  - With 5 V voltage supply: Cable up to 32 m long, shielded
  - With 24 V voltage supply: Cable up to 100 m long, shielded

Note

If the 5 V encoder does not output a zero mark signal and wire-break monitoring is enabled, you must interconnect the zero mark signal inputs N and /N externally so that the inputs have different signal levels (for example, N to 5 V, /N to ground).

Absolute encoders supported

Absolute encoders (Page 2392) with serial interface are supported. Position data are transferred synchronously using the SSI protocol (**S**ynchronous**S**erial**I**nterface). FM x51 only supports GRAY code. The data formats 25-bit (fir tree) and 13-bit (half fir tree) result from the arrangement of the data bits in the frames.

Absolute encoders are divided into the categories:

- Single-turn encoder
  The total range of single-turn encoders is scaled to one revolution.
- Multiturn encoder
  The total range of multiturn encoders is scaled to several revolutions.

Encoder type	Frame length
Single-turn encoder	13 bits
Single-turn encoder	25 bits
Multiturn encoder	13 bits … 25 bits

The transmission rate for data transmission depends on the cable length:

- Max. 188 kHz for 200 m of cable, shielded
- Max. 375 kHz for 100 m of cable, shielded
- Max. 750 kHz for 40 m of cable, shielded
- Max. 1.5 MHz for 12 m of cable, shielded

Incremental encoder (S7-300, S7-400)

Signal forms

The table below illustrates the signal forms of encoders with asymmetrical and symmetrical output signals.

asymmetrical	symmetrical
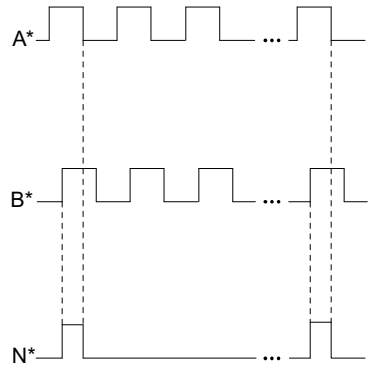	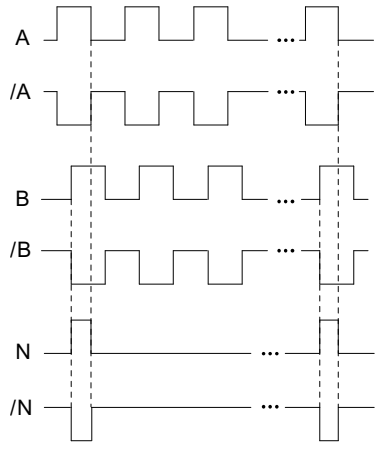

Signal evaluation

Increments

An increment identifies a signal period of the two encoder signals A and B. This value is specified on the rating plate of the encoder and/or in the encoder specifications.

The figure below shows the ratio of increments to pulses.

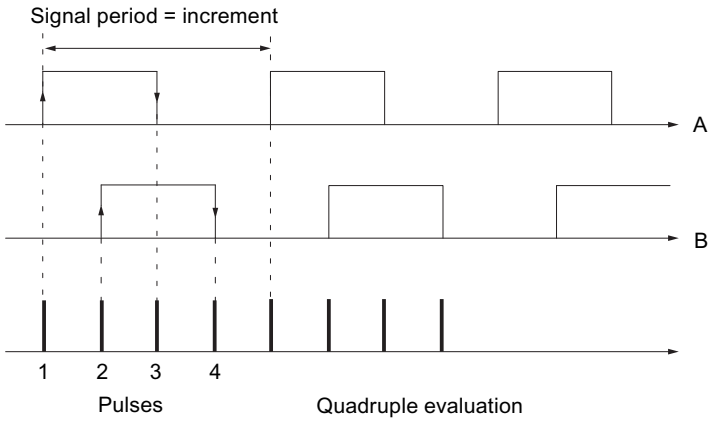

Pulses

The positioning module evaluates all 4 edges of signals A and B in each increment (quadruple evaluation).

Pulses
1 increment (encoder specification) = 4 pulses (evaluation in the module)

Response times

The positioning module has the following response times for connected incremental encoders:

Response times
Response time = switching cycle of the connected switching elements

Note

You can compensate for the minimum response time by assigning the changeover difference and switch-off difference accordingly.

Indecision

The indecision influences the precision of the positioning. In the case of incremental encoders the indecision is negligible.

Absolute encoder (S7-300, S7-400)

Pulse evaluation of absolute encoders

Pulse evaluation of absolute encoders
1 increment (encoder specification) = 1 pulse (FM evaluation)

Response times

The FM x51 has the following response times for absolute encoders:

Response times
Minimum response time = message frame time + switching cycle of the connected switching elements
Maximum reaction time = 2 x frame runtime + monoflop time + switching time of the connected switching elements
With programmable absolute encoders: Maximum reaction time = message frame time + monoflop time + switching cycle of the connected switching elements +1/ max. step sequence frequency

Monoflop time

The monoflop time is 64 µs.

Encoders with values greater than the limits specified here are not permitted.

Message frame times

The message frame times depend on the transmission rate:

Transmission rate	Message frame time with 13 bit	Message frame time with 25 bit
0.188 MHz	75 µs	139 µs
0.375 MHz	38 µs	70 µs
0.750 MHz	19 µs	35 µs
1.500 MHz	10 µs	18 µs

Example of response times

This example shows you how to calculate the minimum and maximum reaction time. The example does not use a programmable encoder.

- Hardware switching cycle: approx. 150 µs
- Message frame time: 18 µs at 1.5 MHz transmission rate (25 Bit message frames)
- Monoflop time: 64 µs

Minimum response time = 18 µs + 150 µs = 168 µs

Maximum reaction time = 2 x 18 µs + 64 µs + 150 µs = 250 µs

Note

You can compensate for the minimum response time by assigning the changeover and switch-off differences accordingly.

Indecision

The indecision is the difference between the maximum and the minimum response time. With an absolute encoder this amounts to

Indecision
Indecision = message frame time + monoflop time
With programmable absolute encoders: Indecision = message frame time + monoflop time + 1/max. step sequence frequency

6.2.5 Using FM 351 (S7-300, S7-400)

6.2.5.1 Terminal assignment of FM 351 front connector (S7-300, S7-400)

Description of the encoder interface (S7-300, S7-400)

Location of the sub D sockets

The figure shows the position and designation of the connectors on the module. You can connect an incremental encoder or absolute encoder (SSI) to the two sub D sockets.

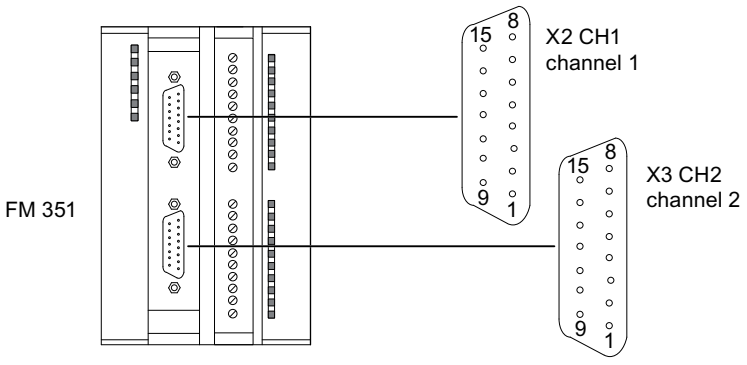

Assignment of encoder interfaces (X2 and X3)

Pin	Name	Incremental encoder (24 V)	Incremental encoder (5 V)	Absolute encoder
1	A*	Encoder signal A	---	---
2	CLS	---	---	SSI shift clock
3	/CLS	---	---	SSI shift clock inverse
4	B*	Encoder signal B	---	---
5	24 VDC	Encoder supply	Sensor supply	Sensor supply
6	5.2 VDC	---	Sensor supply	Sensor supply
7	M	Chassis ground	Chassis ground	Chassis ground

Pin	Name	Incremental encoder (24 V)	Incremental encoder (5 V)	Absolute encoder
8	N*	Zero mark signal	---	---
9	RE	Current sourcing/current sinking[1)]	---	---
10	N	---	Zero mark signal	---
11	/N	---	Zero mark signal, inverse	---
12	/B	---	Encoder signal B, inverse	---
13	B	---	Encoder signal B	---
14	/A / /DAT	---	Encoder signal A, inverse	SSI data inverse
15	A / DAT	---	Encoder signal A	SSI data

[1)]**Note:** If you are connecting an incremental encoder from a third party in series-mode switching (current sourcing/current sinking), the following must be observed:

- Sourcing output: Connect RE (9) with ground (7).
- Sinking output: Connect RE (9) with +24 V (5).

Terminal assignment of the front connector (S7-300, S7-400)

Front connector

Connect the following to the 20-pin front connector:

- the digital inputs and outputs assigned to the channels
- the supply voltage of the encoders and digital outputs

Terminal assignment of front connector (X1)

The table below shows the terminal assignment of the front connector of FM 351.

Wiring	Name	Function	Incremental encoder	Absolute encoder
1	1L+	24 V DC auxiliary voltage for the encoder supply		
2	1M	Ground for sensor power supply		
3	1I0	Channel 1: Digital input 0	Reference point switch	Not used
4	1I1	Channel 1: Digital input 1	Reversing switch	Not used
5	1I2	Channel 1: Digital input 2	Enable input	
6	1I3	Channel 1: Digital input 3	Not used	
7	2I0	Channel 2: Digital input 0	Reference point switch	Not used
8	2I1	Channel 2: Digital input 1	Reversing switch	Not used
9	2I2	Channel 2: Digital input 2	Enable input	
10	2I3	Channel 2: Digital input 3	Not used	
11	1Q0	Channel 1: Digital output 0		
12	1Q1	Channel 1: Digital output 1		
13	1Q2	Channel 1: Digital output 2		
14	1Q3	Channel 1: Digital output 3		
15	2Q0	Channel 2: Digital output 0		

Wiring	Name	Function	Incremental encoder	Absolute encoder
16	2Q1	Channel 2: Digital output 1		
17	2Q2	Channel 2: Digital output 2		
18	2Q3	Channel 2: Digital output 3		
19	2L+	24 V DC auxiliary voltage for the load current supply		
20	2M	Mass load power supply		

Additional information about design and wiring can be found in the Operating Instructions (http://support.automation.siemens.com/WW/view/en/2103043) of the positioning module FM 351.

Auxiliary voltage for the encoder supply (1L+, 1M)

Here, you connect a 24 V DC auxiliary voltage for the encoder supply. The reference potential of this supply (1M) is not connected to the ground of the load power supply (2M) in the FM 351.

The 24 V DC auxiliary voltage for the encoder supply is monitored for undervoltage and ground wire break.

This supplies 24 V DC and 5.2 V DC at the encoder interface (sub D sockets X2 and X3) to the various types of encoders.

CAUTION

Ensure the correct polarity of the 24 V DC auxiliary voltage for the encoder supply (1L+, 1M).

If you reverse the polarity of the 24 V DC auxiliary voltage for the encoder supply, the module will be damaged and will have to be replaced.

Auxiliary voltage for the load power supply (2L+, 2M)

You must connect a 24 V DC auxiliary voltage for the load power supply of the digital outputs to the terminals 2L+ and 2M.

CAUTION

Ensure the correct polarity of the 24 V DC auxiliary voltage for the load power supply (2L+, 2M).

If you reverse the polarity of the 24 V DC auxiliary voltage for the load current supply, the module will be damaged and will have to be replaced.

Wiring information for 24 V DC

When wiring, note that the terminals 1L+, 1M and 2L+, 2M must be interconnected so that the module will operate error-free.

If you connect the 1L+, 1M and 2L+, 2M to separate voltage supplies, after a voltage loss on the auxiliary voltage for the load power supply (2L+, 2M) the synchronization of the axes is maintained.

Load power supplies

The DC load current supply must meet the following requirements:

Only a safe, isolated extra-low voltage of ≤ 60 V DC may be used as the load current supply. Safe isolation can be implemented, for example, by adhering to the specifications

- VDE 0100 Part 410 / HD 384-4-41 / IEC 364-4-41
  (as protective extra low voltage) and/or
- VDE 0805 / EN 60950 / IEC 950
  (as safety low voltage SELV) or
- VDE 0106 Part 101.

8 digital inputs (1I0 to 2I3)

The FM 351 has 4 digital inputs per channel.

You can connect bounce-free switches (24 V current sourcing) or non-contact sensors (2- or 3-wire proximity switches) to the 8 digital inputs.

The digital inputs are not monitored for short circuit or wire break and are electrically isolated from the encoder supply ground and the CPU ground.

The state of each input can be read off from the respective LED.

8 digital outputs (1Q0 to 2Q3)

The FM 351 has 4 digital outputs per channel.

The power unit is controlled via the digital outputs. The function of the digital outputs depends on the control mode (Page 2436).

The digital outputs are not monitored for short circuit or wire break and are electrically isolated from the encoder supply ground and the CPU ground.

The state of each output can be read off from the respective LED.

6.2.5.2 Functions of the FM 351 (S7-300, S7-400)

Overview of functions (S7-300, S7-400)

You define module functionality by setting the mode, job or function switch. The table below gives an overview of the functionality of the module.

Functionality	Description
Writing and reading machine data (Page 2398)	You can use the machine data to adapt the positioning module to the axis and the encoder.
Writing and reading incremental dimension tables (Page 2400)	The incremental dimensions define the targets controlled by the positioning module with "Relative / absolute incremental approach" mode.

Functionality	Description
End of a positioning operation (Page 2402)	The end of a positioning operation is achieved in three different ways: • Target approach • Shutdown • Cancel
"Jog" mode (Page 2409)	In "Jog" mode you can move the drive in one direction by pressing a button.
"Reference point approach" mode (Page 2412)	The "Reference point approach" mode can be used to synchronize the axis based on a recurring external event.
"Incremental approach" mode (Page 2419)	In "Incremental approach" mode, the positioning module can move the drive • toward absolute targets, • relatively by an incremental distance in a specified direction.
"Set actual value" / "Undo set actual value" job (Page 2424)	With the "Set actual value" job, you assign a new coordinate to the actual encoder value. With the "Undo set actual value" job, you reset all the operating range shifts that were generated by means of the "Set actual value" job.
"Set reference point" job (Page 2426)	With the "Set reference point" job, you synchronize the axis.
"Loop traverse" function switch (Page 2428)	You use "Loop traverse" to specify the direction in which a target is approached with a frictional connection.
"Enable input" function switch (Page 2431)	The enable input is an external input with which positioning can be enabled by an external event.
"Read position data" job (Page 2432)	With the "read position data" job you can read the incremental dimension, residual distance and velocity at the current time.
"Read encoder data" job (Page 2432)	With the "read encoder data" job you can read the current data of the encoder as well as the value for the absolute encoder adjustment.
Checkback signals for positioning (Page 2433)	The "Checkback signals for positioning" inform you of the current status of the positioning.
Checkback signals for diagnostics (Page 2433)	The "Checkback signals for diagnostics" informs you of occurring diagnostic events.

Writing and reading machine data (S7-300, S7-400)

Writing, enabling and reading machine data

You can use the machine data to adapt the positioning module to the axis and the encoder.

You can change and read parameters during operation using the user program.

All parameters are stored in the parameter DB:

- Machine data is located in the parameter DB at the addresses 4.0 to 116.0.

Enter the parameter DB number in the corresponding channel DB.

You can enter the parameters with the DB Editor or, simply and easily, enter them in the "Drive", "Axis", "Encoder" and "Increments" dialogs before writing them to the parameter DB with the "Export" function.

You can use the "Import" function to import parameters from an existing parameter DB to the dialogs.

Initial parameter assignment

If the channel does not yet contain any machine data, proceed as follows for initial parameter assignment without the parameter assignment dialogs:

1. Enter the new values in the parameter DB and then save them.
2. Download the parameter DB to the CPU.
3. Set the following trigger bit in the channel DB for the job:
   – Write machine data (MDWR_EN = 1).
4. Call the ABS_CTRL instruction in the cyclic user program.

Modifying machine data

Proceed as follows to change machine data using the user program:

1. Enter the new values in the parameter DB.
2. Set the trigger bits in the channel DB for the jobs:
   – Write machine data (MDWR_EN = 1)
   – Activate machine data (MD_EN = 1)
3. Call the ABS_CTRL instruction in the cyclic user program.

If you set the trigger bits for these jobs all at once, the ABS_CTRL instruction will ensure the jobs are processed in the correct order.

Otherwise change the machine data always in the following sequence:

- Write machine data
- Enable machine data

Reading machine data

In order to read the current machine data from a channel, proceed as follows:

1. Set the following trigger bit in the channel DB:
   – Reading machine data (MDRD_EN = 1)
2. Call the ABS_CTRL instruction in the cyclic user program.

The current machine data is then stored in the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.0	MDWR_EN	BOOL	FALSE	1 = Write machine data
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
36.5	MDRD_EN	BOOL	FALSE	1 = read machine data

Note

If synchronization-relevant parameters have been changed, the module will carry out the following actions for the respective channel when the machine data is enabled:

- Synchronization will be deleted.
- The function switches and zero offset will be reset.
- All previous machine data and incremental dimension tables will become invalid.

Parameters relevant to synchronization:

- Axis type
- End of rotary axis
- Signal type
- Distance per encoder revolution
- Increments per encoder revolution
- Number of revolutions
- Reference point coordinate
- Absolute encoder adjustment
- Type of reference point approach
- Count direction

Writing and reading incremental dimension tables (S7-300, S7-400)

Writing and reading incremental dimension tables

The incremental dimensions define the targets controlled by the positioning module with "Relative / absolute incremental approach" mode.

You can change and read parameters during operation using the user program.

All parameters are stored in the parameter DB:

- Increment tables are located in the parameter DB at the addresses 120.0 to 516.0.

The number of the parameter DB must be entered in the respective associated channel DB.

You can enter the parameters with the DB Editor or, simply and easily, enter them in the "Incremental dimensions" dialog before writing them to the parameter DB with the "Export" function.

You can use the function "Import" to import parameters in an existing parameter DB to the dialogs.

Initial parameter assignment

If the channel does not yet contain any increment tables, proceed as follows for initial parameter assignment without the parameter assignment dialog:

1. Enter the new values in the parameter DB and then save them.
2. Download the parameter DB to the CPU.
3. Set the trigger bits in the channel DB for the jobs:
   – Writing incremental dimension table 1 (TRGL1WR_EN = 1) and / or writing incremental dimension table 2 (TRGL2WR_EN = 1).
4. Call the ABS_CTRL instruction in the cyclic user program.

Changing increment tables

Proceed as follows to change increment tables using the user program:

1. Enter the new values in the parameter DB.
2. Set the trigger bits in the channel DB for the jobs:
   – Writing incremental dimension table 1 (TRGL1WR_EN = 1) and / or writing incremental dimension table 2 (TRGL2WR_EN = 1).
3. Call the ABS_CTRL instruction in the cyclic user program.

Reading increment tables

In order to read the increment tables from a channel, proceed as follows:

1. Set the trigger bits in the channel DB for the jobs:
   – Reading incremental dimension table 1 (TRGL1RD_EN = 1) and / or writing incremental dimension table 2 (TRGL2RD_EN = 1).
2. Call the ABS_CTRL instruction in the cyclic user program.

The increment tables are then stored in the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.4	TRGL1WR_EN	BOOL	FALSE	1 = write increment table 1 (1 ... 50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = write increment table 2 (51 ... 100)
36.6	TRGL1RD_EN	BOOL	FALSE	1 = read increment table 1 (1 ... 50)
36.7	TRGL2RD_EN	BOOL	FALSE	1 = read increment table 2 (51 ... 100)

Note

If synchronization-relevant parameters have been changed, the module will carry out the following actions for the respective channel when the machine data is enabled:

- Synchronization will be deleted.
- The function switches and zero offset will be reset.
- All previous machine data and incremental dimension tables will become invalid.

Parameters relevant to synchronization:

- Axis type
- End of rotary axis
- Signal type
- Distance per encoder revolution
- Increments per encoder revolution
- Number of revolutions
- Reference point coordinate
- Absolute encoder adjustment
- Type of reference point approach
- Count direction

End of a positioning operation (S7-300, S7-400)

Definition

The end of a positioning operation is displayed by the checkback signal WORKING = 0. It can be reached in three different ways:

- Target approach
- End
- Cancel

Monitoring

The following monitoring functions are active towards the end of a positioning operation:

- Monitoring time
  The monitoring time is retriggered at the switch-off point for the last time and then loses its validity with the end of positioning.
  The end of the positioning operation must be reached within this time. If it is not, the outputs will be switched off and the operating error "Error during target approach" (error number 5) issued.
- Monitoring the target range
  The FM x51 sets a symmetrical range around each target, thereby defining the positioning precision of your application. The axis must come to a standstill within this range during a target approach. A value 0 switches off the tolerance during the target approach.
- Monitoring the standstill velocity
  The standstill velocity is used determine whether the drive comes to a standstill within the target range. The FM x51 checks whether the actual drive speed is less than the stationary drive speed when the switch-off point has been reached.
  It must be less than the standstill velocity within the target range. The FM x51 will otherwise report the operating error "Target range overrun" (error number 10).
  Falling short of the standstill velocity is monitored only once per target approach.
  Note that the velocity can briefly fall below the standstill velocity for the velocity determination of the module if the axis moves at a very slow positioning velocity (less than 2 pulses per 8 ms).
- Monitoring the stationary range
  Once a positioning operation ends, the drive is monitored to see whether it remains at an approached target position or drifts off.
  The standstill range is monitored
  - after the FM x51 has reported the checkback signal "PR"
  - if the monitoring time is exceeded
  - if the velocity falls below the standstill velocity.

  If the standstill range is exited without a valid traversing job, the FM x51 reports the "Stationary range exited" error (error number 6).

Target approach

The target approach in the "Absolute incremental approach" and "Incremental approach" modes starts when the switch-off point has been reached. As of this point the drive is switched off and the FM x51 adopts the monitoring functions.

Depending on the monitoring you have parameterized, there are a number of different scenarios in which the checkback signal "PR (POS_RCD)" is generated. The positioning operation is cancelled if no "PR (POS_RCD)" checkback signal is generated.

Scenario 1: You have assigned parameters as follows:

- Target range (TRG_RANGE) > 0
- Standstill velocity (ZSPEED_L) > 0
- Monitoring time (MON_TIME) > 0

PR is generated when the velocity falls below the standstill velocity and the target range has been reached. It does not matter which condition is fulfilled first.

PR is not generated if the actual value has not reached the target range within the monitoring time or the target range has been overrun without the velocity falling below the standstill velocity.

The figure below shows the target approach in incremental approach mode with a target range > 0 and standstill velocity > 0.

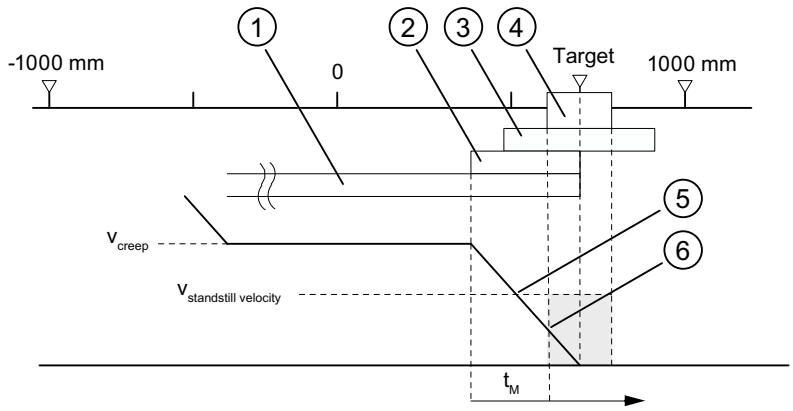

①	Changeover difference plus
②	Switch-off difference plus
③	Stationary range
④	Target range
⑤	Standstill velocity reached
⑥	Target range with V_{still} reached: PR is set
t_M	Monitoring time

Scenario 2: You have assigned parameters as follows:

- Target range (TRG_RANGE) > 0
- Standstill velocity (ZSPEED_L) = 0
- Monitoring time (MON_TIME) > 0

PR is generated when the target range has been reached.

PR is not generated if the actual value does not reach the target range within the monitoring time.

The figure below shows the target approach in incremental approach mode with a target range > 0 and standstill velocity = 0.

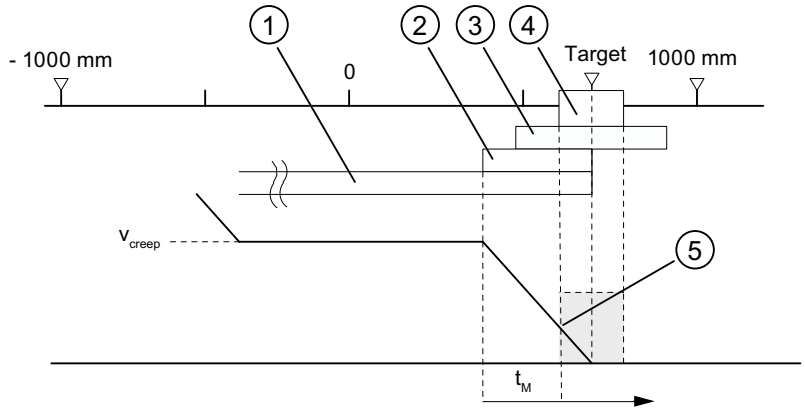

① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Target range
⑤ Target range with V_{still} reached: PR is set
t_M Monitoring time

Scenario 3: You have assigned parameters as follows:

- Target range (TRG_RANGE) = 0
- Standstill velocity (ZSPEED_L) > 0
- Monitoring time (MON_TIME) > 0

PR is generated if the velocity falls below the standstill velocity and the target is then reached.

PR is not generated if the actual value has not reached the target within the monitoring time or the target range has been overrun without the velocity falling below the standstill velocity.

The figure below shows the target approach in incremental approach mode with a target range = 0 and standstill velocity > 0.

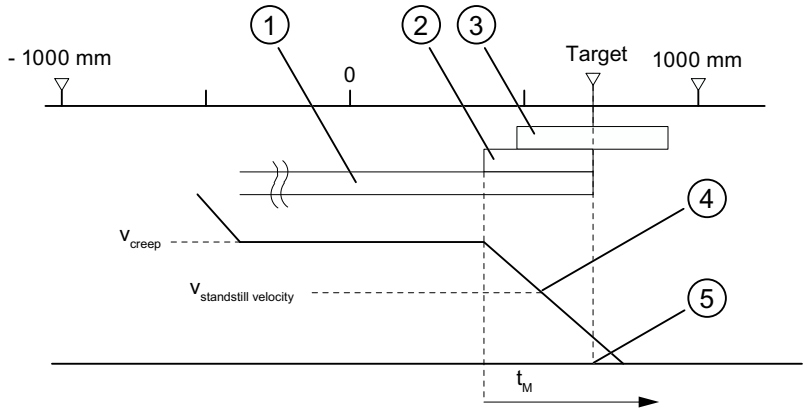

① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Standstill velocity reached
⑤ Target reached: PR is set
t_M Monitoring time

Scenario 4: You have assigned parameters as follows:

- Target range (TRG_RANGE) = 0
- Standstill velocity (ZSPEED_L) = 0
- Monitoring time (MON_TIME) > 0

PR is generated when the target has been reached.

PR is not generated if the actual value does not reach the target within the monitoring time.

The figure below shows the target approach in incremental approach mode with a target range = 0 and standstill velocity = 0.

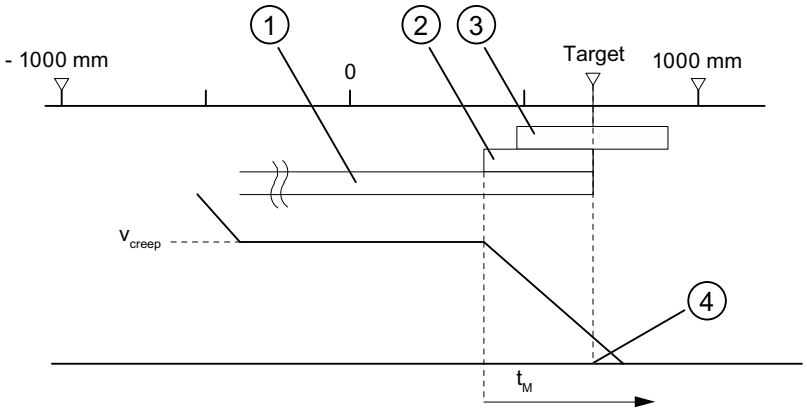

① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Target reached: PR is set
t_M Monitoring time

Scenario 5: You have assigned parameters as follows:

- Target range (TRG_RANGE) ≥ 0
- Standstill velocity (ZSPEED_L) ≥ 0
- Monitoring time (MON_TIME) = 0

If, in this case, the drive comes to a standstill before the target range, the end of the positioning operation will not be detected. PR will not be generated and the checkback signal WORKING will remain set. You can only cancel the positioning operation only by deleting the drive enable (DRV_EN = 0).

Shutdown without specified target

Shutdown means: The positioning operation is purposefully ended whilst complying with the differences between rapid traverse and creep speed.

The positioning operation is ended if

- The FM x51 receives a STOP signal (STOP = 1)
- The operating modes "Jog" and "Reference point approach" are ended.
- Operating errors or the operating error "Target overrun" (error number 9) occur.

The checkback signal "PR (POS_RCD)" is not set. The order of events are analogous to those of the target approach.

The figure below shows the end of a positioning operation.

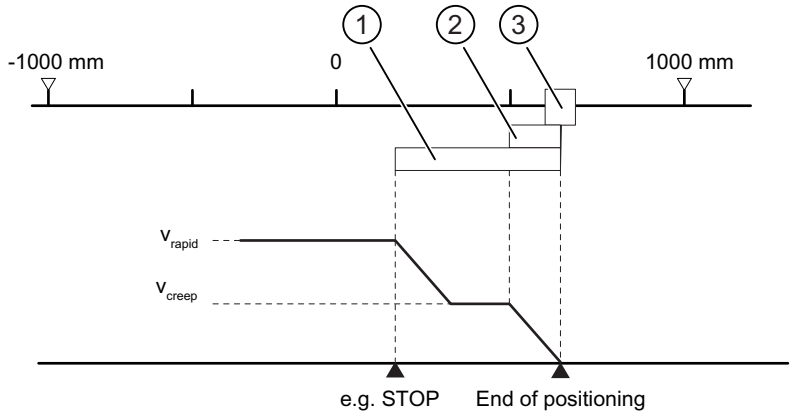

① Changeover difference plus
② Switch-off difference plus
③ Target range

Cancel

Shutdown means: The positioning process is ended immediately after standstill without the use of the changeover and switch-off differences between rapid traverse and creep speed. In addition, all the relevant outputs of the respective control mode are immediately switched off and also:

- Increment = actual value
- Distance-to-go = zero

The positioning operation is cancelled if

- The drive enable signal is deleted (DRV_EN = 0).
- The CPU goes into STOP mode.
- Diagnostic errors or all operating errors apart from the "Target overrun" operating error (error number 9) occur.

The checkback signal "PR (POS_RCD)" is not be set in "Incremental approach" mode.

If the standstill velocity is assigned, the standstill monitoring will become active as soon as the velocity falls below the standstill velocity. If the standstill velocity is not assigned, the standstill monitoring will be active when the outputs have been switched off.

The figure below shows the cancellation of a positioning operation.

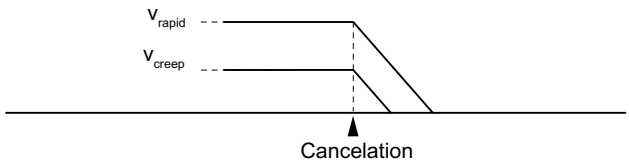

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	TRG_RANGE	DINT	L#1000	Target range
80.0	MON_TIME	DINT	L#2000	Monitoring time
84.0	ZSPEED_R	DINT	L#1000	Stationary range
88.0	ZSPEED_L	DINT	L#30000	Standstill velocity

Checkback signals in the channel DB

Address	Name	Type	Initial value	Comment
23.1	WORKING	BOOL	FALSE	1 = Positioning operation is running
25.7	POS_RCD	BOOL	FALSE	1 = position reached

"Jog" mode (S7-300, S7-400)

Definition

In "Jog" mode you can move the drive in one direction by pushing a button. You must install a button for both directions (plus and minus) respectively. You can use the "Jog" mode both for a synchronized as well as for an unsynchronized axis.

In the case of an unsynchronized axis, jogging is a positioning in the specified direction.

For a synchronized axis (linear axis), jogging involves positioning to the software limit switches.

Requirement

The axis parameters must be assigned.

Sequence of events for "Jog" mode

1. Set the control signal for the "Jog" mode (MODE_IN = 1).
2. Set the control signal for the drive enable (DRV_EN = 1).
3. Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.
4. Enter the start speed.
   - Rapid traverse (MODE_TYPE = 1)
   - Creep speed (MODE_TYPE = 0)
5. Set the control signal for traversing direction plus or minus (DIR_P = 1 or DIR_M = 1).
6. Call the ABS_CTRL instruction.

The following diagram shows an example of "Jog" mode.

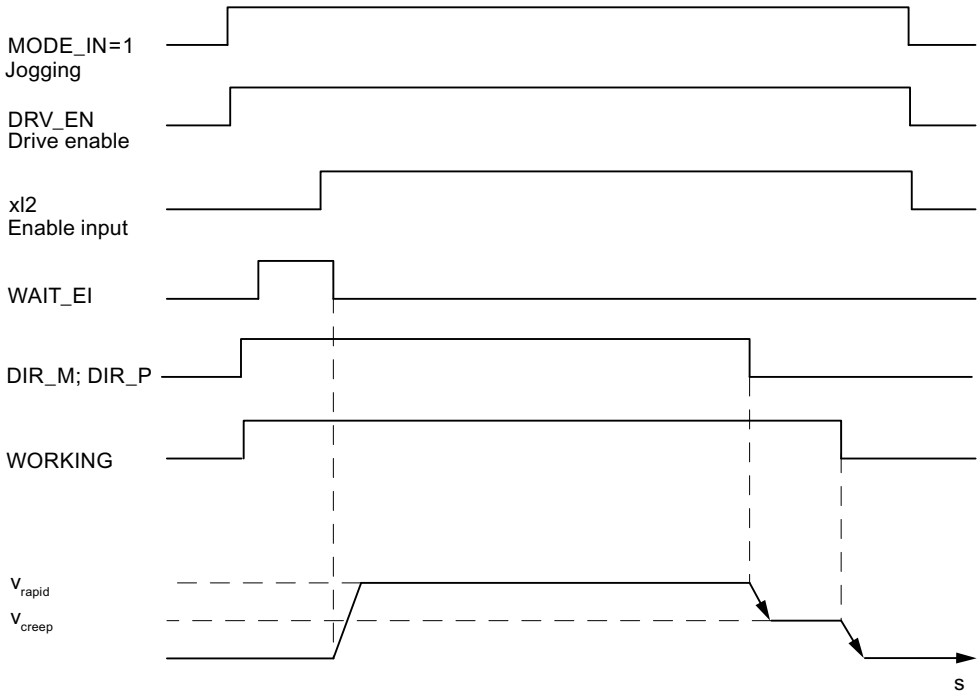

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	1 = jog
17.0	MODE_TYPE	BYTE	B#16#0	1 = rapid traverse 0 = creep speed
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = Positioning operation is running
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Shutting down jogging

The "Jog" mode is shut down if

- You reopen the switch used for jogging (DIR_M = 0 or DIR_P = 0),
- The FM 351 receives a STOP signal (STOP = 1)
- the process value with a synchronized linear axis reaches the limit of the operating range. The axis can only continue moving in the opposite direction.

After traversing is shut down, it can continue in any direction.

Canceling jogging

"Jog" mode is cancelled if

- The drive enable signal is deleted (DRV_EN = 0)
- A traversing range limit has been overrun for a linear axis

Monitoring functions

The following monitoring functions are not active at the end of positioning in "Jog" mode:

- Monitoring the stationary range
- Monitoring the target range
- Monitoring the standstill velocity

Jogging does not result in proper positioning at the end of which the "Position reached" signal is set.

Operating range limit for a linear axis

The limits for the "Jog" mode differ between a synchronized and an unsynchronized axis.

The following table shows "Jog" with synchronized and non-synchronized axes.

Axis is unsynchronized	Axis is synchronized
If, when jogging, the traversing range limit is overrun, • the process value display is no longer valid. • the positioning process is cancelled. 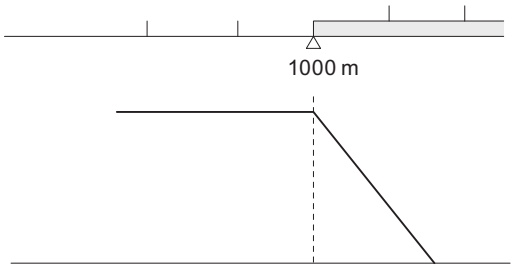 The process value indicated is no longer valid	Jog involves positioning to targets that are removed from the software limit switches by a distance equal to the whole target range. The operating range limits are determined from • **SLE - ½ target range** for the end of the linear axis in the plus direction • **SLE + ½ target range** for the end of the linear axis in the plus direction If you do not open the switch beforehand, the FM 351 shuts down at a target point that is positioned half the target range away from the respective software limit switch. All ranges that are necessary for proper shutdown are set by the FM 351 relative to this target point. 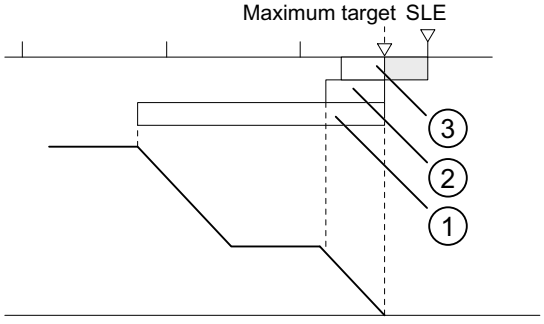 = Part of operating range in which a target position cannot be set ① = Changeover difference plus ② = Switch-off difference plus ③ = ½ target range

"Reference point approach" mode (S7-300, S7-400)

Definition

The "Reference point approach" mode can be used to synchronize the axis based on a recurring external event.

Requirements

- An incremental encoder with zero mark.
- The axis parameters must be assigned:

Connections	Channel 1	Channel 2
Reference point switch	Digital input 1I0	Digital input 2I0
	The reference point switch should be such that the drive can safely brake from rapid traverse to creep speed in the range of the switch.	
Reversing switch	Digital input 1I1	Digital input 2I1
	When assigning parameters, ensure that the start of the reference point approach is assigned in the direction of the reversing switch. Only in this way can you guarantee that the reference point switch will always be found.	
Enable input	Digital input 1I2	Digital input 2I2

Sequence of events for "Reference point approach" mode

The keys for the positive and negative traversing direction must be installed for each channel.

1. Set the value for the reference point coordinate in the parameter DB REFPT.
2. Enter the type of "Reference point approach" in the parameter DB.
   The following options are available:

Start in direction ...	That leads to synchronization ...	
plus	the first zero mark after leaving the reference point switch in direction **plus**	REFPT_TYPE = 0
plus	the first zero mark after leaving the reference point switch in direction **minus**	REFPT_TYPE = 1
minus	the first zero mark after leaving the reference point switch in direction **plus**	REFPT_TYPE = 2
minus	the first zero mark after leaving the reference point switch in direction **minus**	REFPT_TYPE = 3

3. Enter the start speed:
   - Rapid traverse (REFPT_SPD = 0)
   - Creep speed (REFPT_SPD = 1)
4. Write and enable the machine data.
5. Set the control signal for the "Reference point approach" mode (MODE_IN = 3).
6. Set the control signal for the drive enable (DRV_EN = 1).
7. Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.
8. Set the control signal for traversing direction plus or minus or the start signal (DIR_P = 1, DIR_M = 1 or START = 1)
9. Call the ABS_CTRL instruction.

The following table shows the start commands for a reference point approach.

Start command	Job	Comment
DIR_P	The drive starts in the direction with positive values. It therefore moves in the direction of the traversing range end.	The FM 351 will signal an operating error if a negative direction is entered in the machine data. No reference point approach is executed.
DIR_M	The drive starts in the direction with negative values. It therefore moves in the direction of the traversing range start.	If a positive direction is entered in the machine data, the FM 351 will signal an operating error. No reference point approach is executed.
START	The drive starts in the direction that is entered in the machine data.	

Note

The following applies to the rotary axis: Reproducibility of the reference point is only ensured if the ratio of the **End of rotary axis** value and the **Distance per encoder revolution** value is an integer.

The following figure shows an example of "Reference point approach" mode.

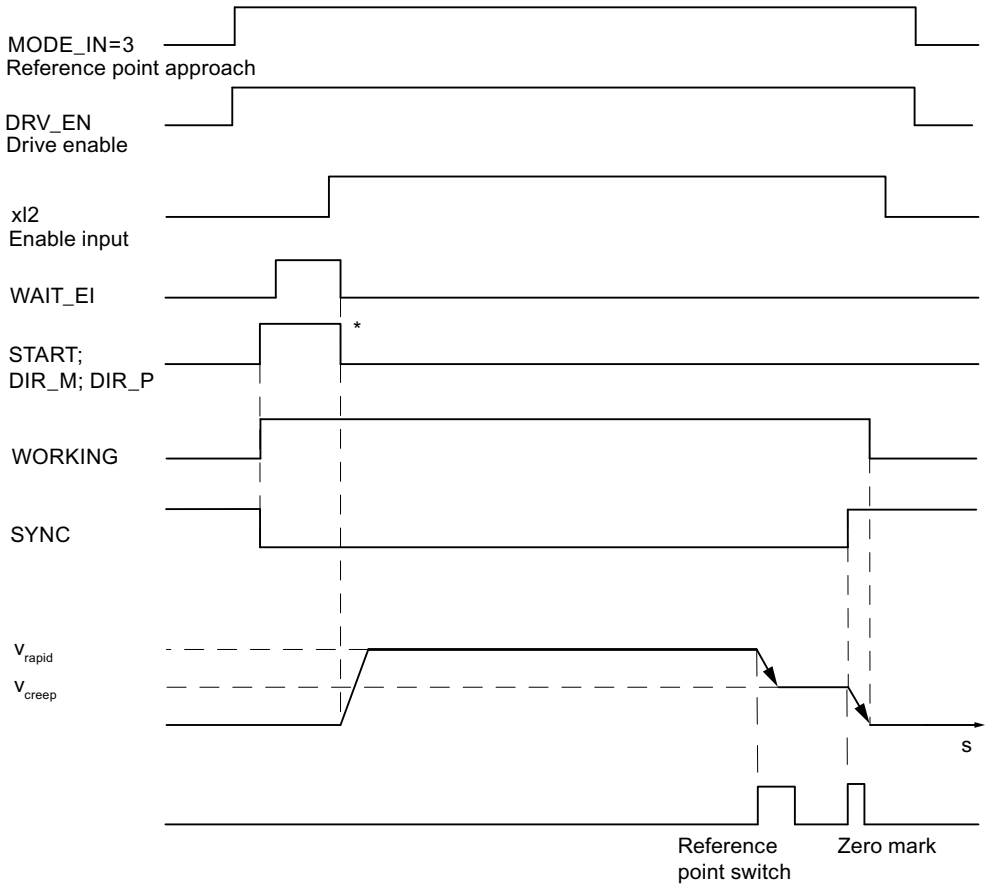

* The start signals are reset by the ABS_CTRL instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = Start positioning operation
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	3 = reference point approach
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = Positioning operation is running
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	Reference point coordinate
52.0	REFPT_TYPE	DINT	L#0	Type of reference point approach
99.0	REFPT_SPD	BOOL	TRUE	Start velocity for the reference point approach 0 = rapid traverse 1 = creep speed

Effects of the mode

- Synchronization is canceled when traversing starts.
- The actual position is set to the value of the reference point coordinate when the SYNC feedback signal is set.
- The operating range is established on the axis.
- The individual points within the operating range maintain their original value, but are located at different positions.

Canceling the reference point approach

The "Reference point approach" mode is cancelled if

- The drive enable signal is deleted (DRV_EN = 0)
- A traversing range limit has been overrun for a linear axis

Reference point approach in dependence on the start position

In the case of a reference point approach, you must distinguish between different cases that are dependent on

- the location of the drive at the time a reference point approach is started,
- the assigned start direction,
- the assigned location of the zero mark relative to the reference point switch.

The following table shows the REFPT_TYPE = 0 and REFPT_TYPE = 1 cases. The figures apply accordingly to REFPT_TYPE = 2 and REFPT_TYPE = 3.

Conditions of the reference point approach	Sequence of events for the reference point approach
Example of reference point approach (REFPT_TYPE = 0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction.	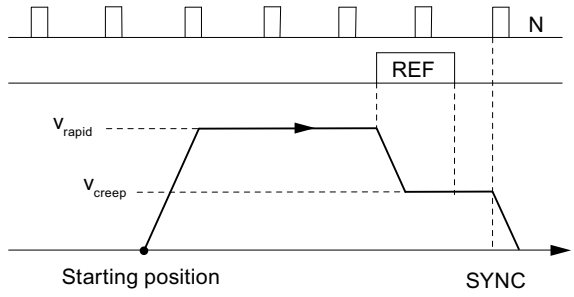
Example of reference point approach (REFPT_TYPE = 1): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the minus direction.	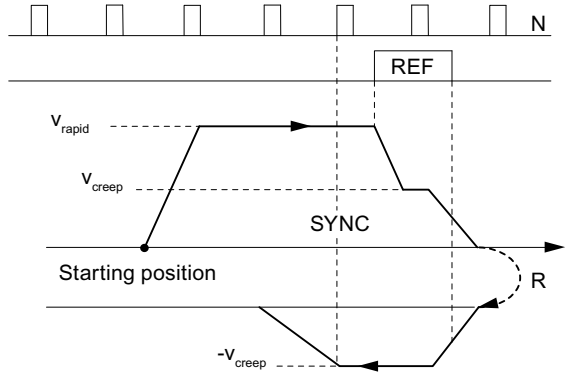
Example of reference point approach (REFPT_TYPE = 0): • The start direction must be assigned in the plus direction. • The location of the zero mark of the reference point switch is assigned in the plus direction. • The reversing switch is more positive than the reference point switch.	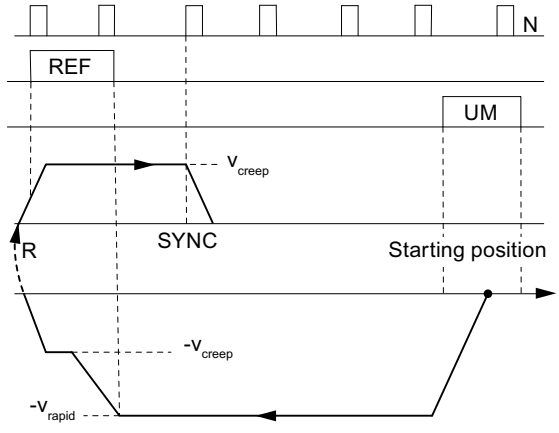

Conditions of the reference point approach	Sequence of events for the reference point approach
Example of reference point approach (REFPT_TYPE = 1): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the minus direction. • The start position of the reference point approach is on the reference point switch.	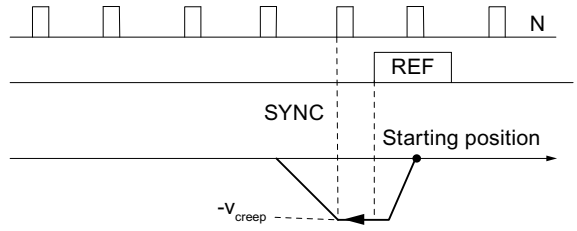
Example of reference point approach (REFPT_TYPE = 0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction. • The reversing switch is more positive than the reference point switch.	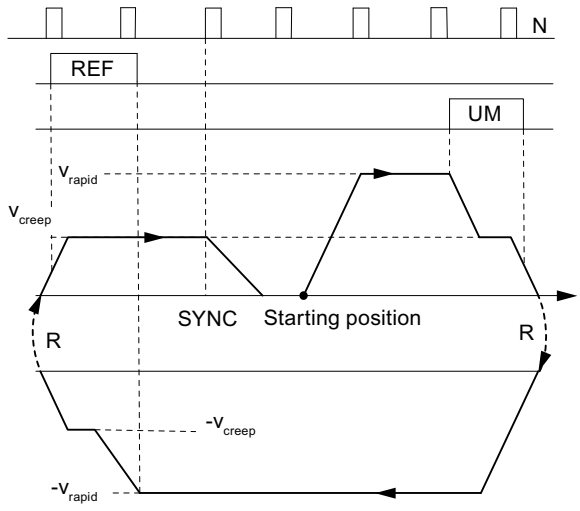
Example of reference point approach (REFPT_TYPE=0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction. • Start velocity = creep speed	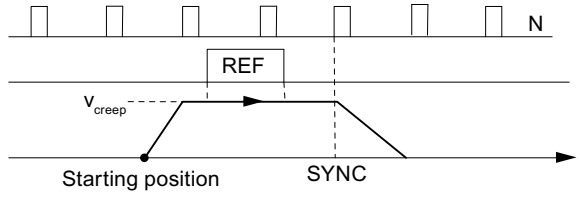
R = Direction reversal REF = Reference point switch UM = Reversing switch N = Zero mark of the encoder SYNC = Synchronization has been achieved	

"Incremental approach" mode (S7-300, S7-400)

Definition

In "Incremental approach" mode, the FM 351 can move the drive.

- toward **absolute** targets,
- **relatively** by an incremental distance in a specified direction.

You specify the target or the relative distances as incremental dimensions of the FM 351. You can enter up to 100 incremental dimensions in a table; these are valid for both "Relative incremental approach" mode and "Absolute incremental approach" mode. Independently of the incremental dimension table, you can specify the distance using incremental dimension 254 (Page 2462) and incremental dimension 255 (Page 2463).

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.
- The increments must be available on the module.

Interpretation of the increments

The FM 351 interprets the specifications differently depending on the incremental approach you select.

- Absolute incremental approach: The increments will be interpreted as an absolute target position.
- Relative incremental approach: The increments will be interpreted as a relative distance from the start position.

Note

Only positive increments are permissible for the "Relative incremental approach" mode.

The incremental dimensions result from the DIR_P or DIR_M direction specifications.

"Incremental approach" mode sequence with incremental dimension number 1 to 100

Step	Absolute incremental approach	Relative incremental approach
	Incremental dimension number 1 to 100	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Enter the incremental dimensions in the tables (TRGL1; TRGL2).	
3	Write the incremental dimension table (TRGL1/2WR_EN = 1).	
4	Set the control signal for the drive enable (DRV_EN = 1).	

Step	Absolute incremental approach	Relative incremental approach
	Incremental dimension number 1 to 100	
5	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
6	Enter the incremental dimension number (MODE_TYPE = 1 … 100).	
7	Set the control signal: • Linear axis: – START; the direction is determined exactly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
8	Call the ABS_CTRL instruction.	
Steps 2 and 3 are only required if no increments are given or the existing increments are to be changed.		

Sequence of events for the "incremental approach" mode with increment number 254

Step	Absolute incremental approach	Relative incremental approach
	Increment number 254	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Set the control signal for the drive enable (DRV_EN = 1).	
3	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
4	Enter the incremental dimension number (MODE_TYPE = 254).	
5	Enter the incremental dimension for the incremental dimension number 254 (TRG252_254).	
6	Set the trigger bit for writing the incremental dimension (TRG252_254_EN = 1).	
7	Set the control signal: • Linear axis: – START; the direction is determined exactly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
8	Call the ABS_CTRL instruction.	

Sequence of events for the "incremental approach" mode with increment number 255

Step	Absolute incremental approach	Relative incremental approach
	Increment number 255	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Set the control signal for the drive enable (DRV_EN = 1).	
3	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
4	Enter the incremental dimension number (MODE_TYPE = 255).	
5	Enter the incremental dimension for the incremental dimension number 255 (TRG255).	
6	Enter the value for the changeover difference of the incremental dimension number 255 (CHGDIF255).	
7	Enter the value for the changeover difference of the incremental dimension number 255 (CUTDIF255).	
8	Set the trigger bit for writing the incremental dimension, switch-off difference, and changeover difference (TRG255_EN = 1).	
9	Set the control signal: • Linear axis: – START; the direction is determined exactly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
10	Call the ABS_CTRL instruction.	

The following figure shows an example of "Incremental approach" mode.

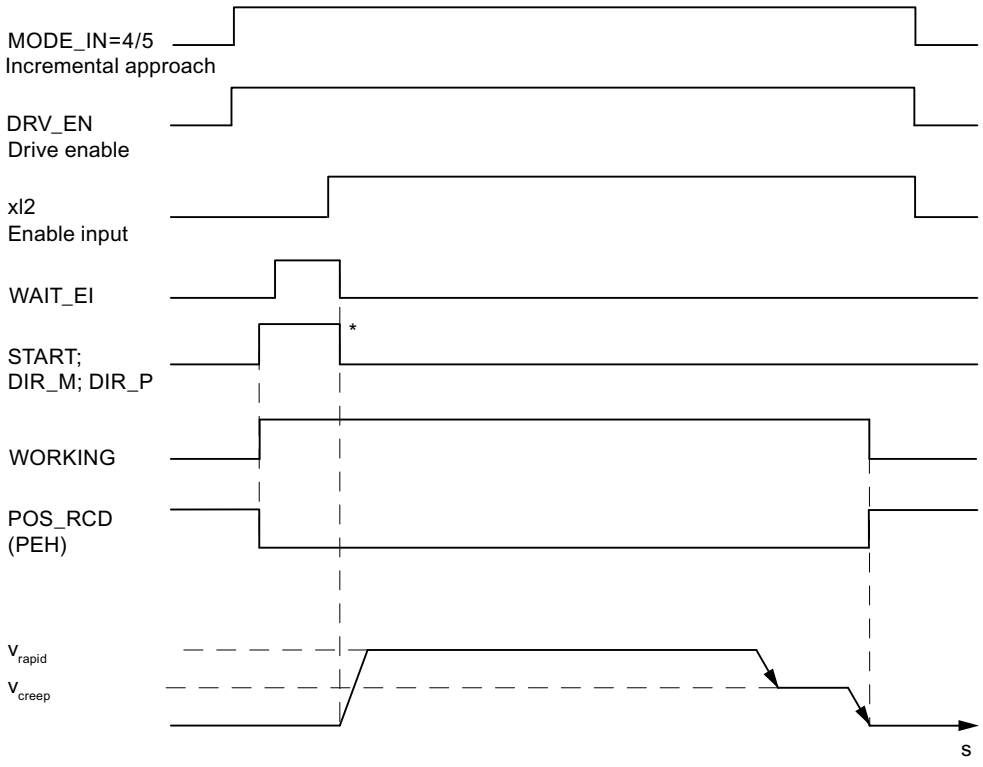

* The start signals are reset by the ABS_CTRL instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = start positioning
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	4 = relative incremental approach 5 = absolute incremental approach
17.0	MODE_TYPE	BYTE	B#16#0	Incremental dimension number 1 … 100, 254 or 255
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = positioning is active
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
25.7	POS_RCD	BOOL	FALSE	1 = Position reached
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input
36.2	TRG252_254_EN	BOOL	FALSE	1 = write increment for increment number 254
36.3	TRG255_EN	BOOL	FALSE	1 = write increment for increment number 255
35.4	TRGL1WR_EN	BOOL	FALSE	1 = Write incremental dimension table 1 (incremental dimension number 1...50)

Address	Name	Type	Initial value	Comment
35.5	TRGL2WR_EN	BOOL	FALSE	1 = Write incremental dimension table 2 (incremental dimension number 51...100)
96.0	TRG252_254	DINT	L#0	Increment for increment number 254
100.0	TRG255	DINT	L#0	Increment for increment number 255
104.0	CHGDIF_255	DINT	L#0	Changeover difference for increment number 255
108.0	CUTDIF_255	DINT	L#0	Switch-off difference for increment number 255

Data used in the parameter DB

Address	Name	Type	Initial value	Comment	
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus	
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus	
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus	
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus	
120.0	TRGL1.TRG[1]	DINT	L#0	Increment number 1	Increment table 1
. . .	. . .	. . .	. . .	. . .	
316.0	TRGL1.TRG[50]	DINT	L#0	Increment number 50	
320.0	TRGL2.TRG[51]	DINT	L#0	Increment number 51	Increment table 2
. . .	. . .	. . .	. . .	. . .	
516.0	TRGL2.TRG[100]	DINT	L#0	Increment number 100	

Distance-to-go

The residual distance is the difference between the target (increment) and process value. The residual distance is

- positive if the target has not yet been reached
- negative if the target has already been overrun.

In the case of a rotary axis, the displayed distance-to-go cannot be used.

Shutting down the Incremental approach

The "Incremental approach" mode is shut down when the FM 351 receives a STOP signal (STOP = 1).

After traversing is shut down, the distance-to-go remains.

The remaining distance-to-go with a "relative incremental approach" can be traveled to the end if

- the mode is unchanged, and
- the increment number is unchanged, and

- the direction is unchanged, and
- the remaining distance-to-go is greater than the assigned switch-off difference.

You travel the distance-to-go by starting the "relative incremental approach" without any changes.

Canceling the incremental approach

The "Incremental approach" mode is cancelled when the "Drive enable" signal is deleted (DRV_EN = 0).

Delete residual distance

With the job "Delete residual distance" you can delete any awaiting residual distance.

You also delete the awaiting residual distance by calling a different mode or by starting the mode in the other direction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.2	DELDIST_EN	BOOL	FALSE	1 = Delete residual distance

"Set actual value" / "Undo set actual value" job (S7-300, S7-400)

Definition

The job "Set actual value" assigns a new coordinate to the current encoder state. The operating range is projected on a different area of the axis.

You determine the shift in the operating range with (ACT_{new}- $ACT_{current}$).

- $ACTUAL_{new}$ is the default value
- $ACTUAL_{current}$ is the actual value at the time of the execution

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.

Job sequence

1. Enter the coordinate AVAL for the actual value (ACT_{new}) again.
   - Linear axis:
     You must select the specified actual value so that the software limit switches are still within the permissible traversing range after the job is called.
     The amount of the shift resulting from (ACT_{new} - $ACT_{current}$) must be less than or equal to the amount of the permissible traversing range (max. 100 m or 1000 m).
   - Rotary axis:
     Rule for the defined actual value:
     0 ≤ actual value < end of rotary axis
2. Set the relevant trigger bit (AVAL_EN = 1).

If issued during a positioning, the "Set actual value" job is held until the end of positioning and only carried out after the subsequent block call.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.7	AVAL_EN	BOOL	FALSE	1 = set actual value
84.0	AVAL	DINT	L#0	Coordinate for "Set actual value"

Effects of the job

In the example "set actual value " to 300 mm you can see how this job projects the operating range onto a specified position of the axis. This has the following results:

- The actual position is set on the value of the actual value coordinate.
- The operating range is shifted on the axis.
- The individual points (e.g., software limit switch end) within the operating range maintain their original value, but have new positions.

The following table shows the shift in the axis operating range by "set reference point".

Set actual value	SLS	AC-TUAL	SLE
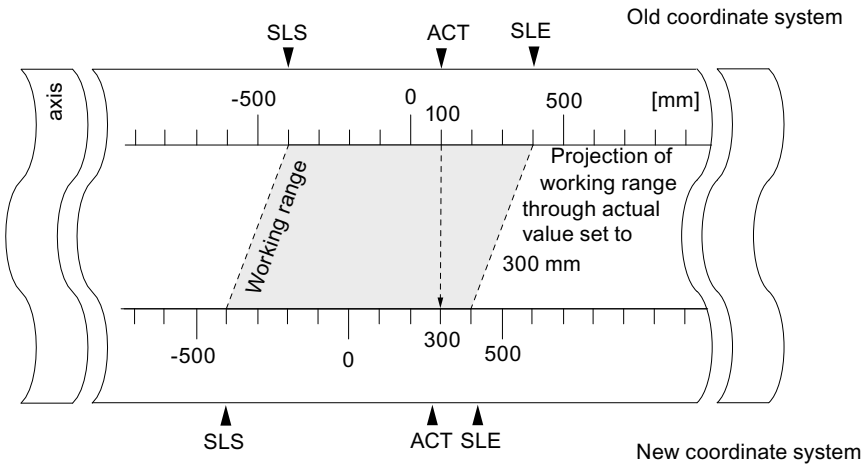	-400	100	400
	-400	300	400

Withdrawing the job ("Undo set actual value")

The job "Undo set actual value" resets all the operating range shifts that were created by means of "Set actual value".

The total of all operating range shifts must not exceed the traversing range, so that this job can be correctly executed.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.3	AVALREM_EN	BOOL	FALSE	1 = cancel set actual value

"Set reference point" job (S7-300, S7-400)

Definition

You use the "set reference point" job to synchronize the axes. The job shifts the operating range. All the shifts that were created with "Set actual value" are retained.

The setting projects the operating range on the axis. Therefore, by entering different values, the operating range can be moved to any position within the physical area of the axis.

Prerequisites

- The positioning must have ended.
- The axis parameters must be assigned.

Job sequence

1. Enter the value REFPT for the reference point coordinate.
   - Linear axis:

     The reference point coordinates must not lie outside the software limit switches. This also applies to the reference point coordinate in a shifted coordinate system.
   - Rotary axis:
     Rule for the reference point coordinate:
     0 ≤ reference point coordinate < end of rotary axis
2. Set the relevant trigger bit (REFPT_EN = 1).

Data used in the channel DB

Address	Name	Type	Initial value	Comment
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
35.6	REFPT_EN	BOOL	FALSE	1 = set reference point
92.0	REFPT	DINT	L#0	Reference point coordinate

Effects of the job

In the example "set reference point" to 400 mm you can see how this job projects the operating range onto a specific, physical position of the axis. This has the following results:

- The actual position is set on the value of the reference point coordinate.
- The operating range is shifted on the axis.
- The individual points (e.g. software limit switch end) maintain their original value, but are located at new positions.
- The SYNC bit in the checkback signals is set.

The following table shows how "Set reference point" shifts the operating range on the axis.

Set reference point	SLS	ACTUAL	SLE
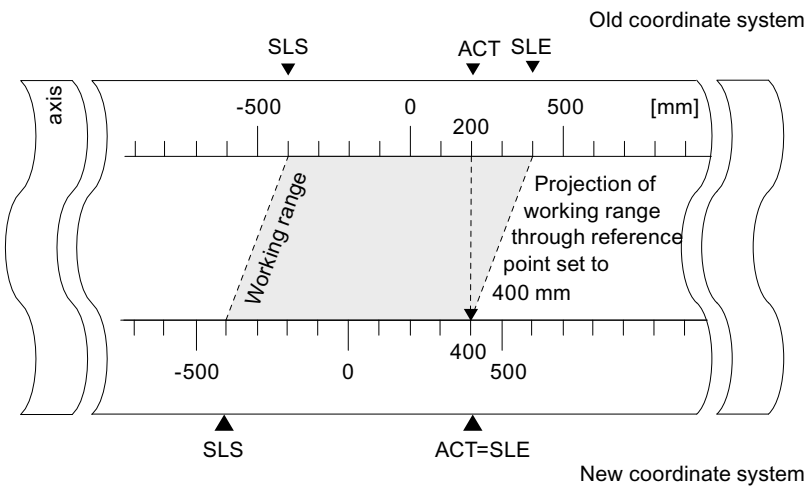	-400	200	400
	-400	400	400

Special features of the absolute encoder

This job is necessary for an absolute encoder adjustment (Page 2450).

"Loop traverse" function switch (S7-300, S7-400)

Definition

With "loop approach" you specify the direction in which a target is approached with a frictional connection. You can use loop approach if a frictional connection between the motor and axis can only be ensured in one direction.

A target that is approached in the direction opposite the specified direction is initially overrun. Then, the module performs a direction reversal and approaches the target in the specified direction.

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.

- In the case of a loop approach in the opposite direction to that of the traversing direction of the target the maximum target position is:
  - traversing direction plus
    Target < SLE - ½ Target range - Switch-off difference plus - Changeover difference minus
  - traversing direction minus
    Target < SLE - ½ Target range + Switch-off difference minus + Changeover difference minus
- A loop approach will not be carried out if the target is being approached in the direction of the loop approach. In this case, an incremental approach without an about-turn in direction will be carried out.
- The sequence of the "incremental approach" (Page 2418) operating mode must be known.

Sequence of events for the loop approach

1. Set the control signal for the "absolute/relative incremental approach" operating mode (MODE_IN = 5/4).
2. Set the control signal for the drive enable (DRV_EN = 1).
3. Set the "Enable input" (EI_OFF = 1) function switch or wire the enable input for the corresponding channel.
4. Enter the incremental dimension number (MODE_TYPE = 1 … 100, 254, 255).
5. Set the "Loop traverse" function switch (PLOOP_ON / MLOOP_ON = 1).
6. Start the incremental approach.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = start positioning
15.2	DIR_M	BOOL	FALSE	1 = minus direction
15.3	DIR_P	BOOL	FALSE	1 = plus direction
15.7	DRV_EN	BOOL	FALSE	1 = switch on drive enable
16.0	MODE_IN	BYTE	B#16#0	4/5 = relative / absolute incremental approach
17.0	MODE_TYPE	BYTE	B#16#0	Incremental dimension number 1 … 100, 254 or 255
34.0	PLOOP_ON	BOOL	FALSE	1 = loop approach in plus direction
34.1	MLOOP_ON	BOOL	FALSE	1 = loop approach in minus direction
34.2	EI_OFF	BOOL	FALSE	1 = do not evaluate enable input

Fictitious target

If you start a positioning to a target that is located in the direction opposite to that of the assigned loop traverse, the module determines a fictitious target for this target. The module performs a direction reversal at the fictitious target and then approaches the target in the proper direction.

This fictitious target **must** be located by at least half the target range before the respective software limit switch.

The distance of the fictitious target from the parameterized target is established depending on the direction: The following table shows the calculation of the location of the fictitious target with a loop traverse.

Guidelines	Location of fictitious target
Parameter assignment: Loop + (frictional connection plus) and approach in minus direction.	The fictitious target (**$target_f$**) has the value: **$target_f$ = target - switch-off difference minus - changeover difference plus** 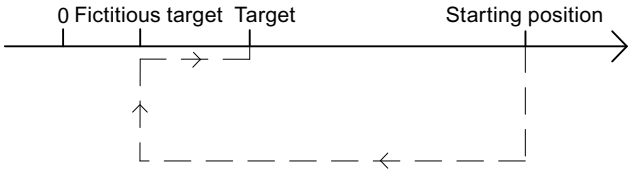
Parameter assignment: Loop - (frictional connection minus) and approach in plus direction.	The fictitious target (**$target_f$**) has the value: **$target_f$ = target + switch-off difference plus + changeover difference minus** 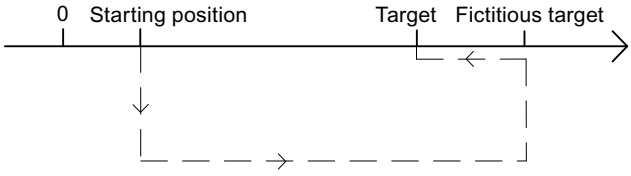

Example

Using a positioning with loop approach minus to a maximum target we can illustrate the location of the fictitious target.

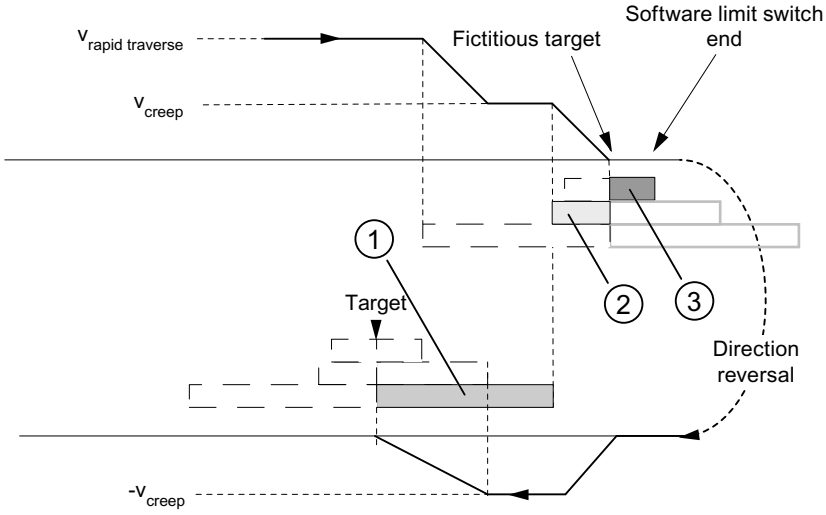

① Changeover difference negative
② Switch-off difference positive
③ ½ target range

"Enable input" function switch (S7-300, S7-400)

Definition

The enable input is an external input with which a positioning can be enabled by an external event.

Evaluate enable input (EI_OFF = 0)

The respective enable input (xI2) must be wired for the channel.

With this you have the possibility of preparing the start of a positioning. Start the positioning independently of the program flow of your user program by setting a "1" signal on the enable input.

Traversing begins when you create a "1" signal at the enable input and is ended when you create a "0" signal at the enable input.

Do not evaluate enable input (EI_OFF = 1)

If you switch off the evaluation of the enable input, an operating mode starts immediately after the start signal is detected. Then it is not possible to prepare an operating mode and start it at a defined later time.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.2	EI_OFF	BOOL	FALSE	1 = do not evaluate enable input

"Read position data" job (S7-300, S7-400)

Definition

With the "read position data" job you can read the increment, distance-to-go and speed at the current time.

Job sequence

1. Set the trigger bit at the DB channel (ACTSPD_EN = 1).
2. The data are then stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.1	ACTSPD_EN	BOOL	FALSE	1 = read position data
112.0	ACTSPD	DINT	L#0	Current speed
116.0	DIST_TO_GO	DINT	L#0	Distance-to-go
120.0	ACT_TRG	DINT	L#0	Current increment

"Read encoder data" job (S7-300, S7-400)

Definition

With the "read encoder data" job you can read the current data of the encoder as well as the value for the absolute encoder adjustment.

Requirements

You can read the value for the absolute encoder adjustment (Page 2450) after you have carried out the "set reference point" (Page 2425) job.

Job sequence

1. Set the trigger bit at the DB channel (ENCVAL_EN = 1).
2. The data is stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.2	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
124.0	ENCVAL	DINT	L#0	Actual encoder value (internal representation)
128.0	ZEROVAL	DINT	L#0	Last zero mark value (internal representation)
132.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment

Querying checkback signals for the positioning (S7-300, S7-400)

Definition

The "checkback signals for the positioning" inform you of the current status of the positioning.

Sequence

1. The data is stored in the channel DB each time the ABS_CTRL instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = positioning is active
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
23.4	SPEED_OUT	BOOL	FALSE	0 = creep speed 1 = rapid traverse
23.5	ZSPEED	BOOL	FALSE	1 = axis is located in the standstill range
23.6	CUTOFF	BOOL	FALSE	1 = axis is located in the switch-off range
23.7	CHGOVER	BOOL	FALSE	1 = axis is located in the changeover range
24.0	MODE_OUT	BYTE	B#16#0	Active operating mode
25.2	GO_M	BOOL	FALSE	1 = axis moves in minus direction
25.3	GO_P	BOOL	FALSE	1 = axis moves in plus direction
25.7	POS_RCD	BOOL	FALSE	1 = Position reached
26.0	ACT_POS	DINT	L#0	Current process value (current position of the axis)

Querying checkback signals for diagnostics (S7-300, S7-400)

Definition

The "checkback signals for the diagnostics" informs you of occurring diagnostic events.

Sequence

1. If the module enters a new event in the diagnostic buffer, it sets the DIAG bit in all the channels in the feedback interface. Error (Page 2475) events are logged in the diagnostics buffer. The DIAG bit is also set when the diagnostic buffer is deleted.
2. If it is not possible to call an operating mode or control an active operating mode, or if this has been carried out incorrectly, the module sets an OT_ERR operator error in the feedback interface. The cause of error is then entered in the diagnostic buffer. While the operator error exists, you can neither start a new operating mode nor continue with the halted one. You acknowledge an existing operating error with OT_ERR_A = 1.
3. The module sets the DATA_ERR bit in the feedback interface when it detects incorrect data in a write job. The cause of error is then entered in the diagnostic buffer.
4. The checkback signals are stored in the channel DB.
5. When the diagnostic buffer is read, the module sets the DIAG bit back to 0 in all channels.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
22.2	DIAG	BOOL	FALSE	1 = delete diagnostic buffer
22.3	OT_ERR	BOOL	FALSE	1 = operator error
22.4	DATA_ERR	BOOL	FALSE	1 = data error

6.2.5.3 Configuring FM 351 (S7-300, S7-400)

6.2.5.4 Parameter assignment FM 351 (S7-300, S7-400)

Specifying channels (S7-300, S7-400)

Selecting system of units (S7-300, S7-400)

Selecting system of units

For the input and output of the data, you can select from the following systems of units:

- Millimeter (default setting)
- Inches
- Degrees

Note

If you change the system of units in the "Specify channels" screen, the values will be converted into the new system. This can cause round-off errors.

If you change the system of units via the machine date, the values are not automatically converted.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
8.0	UNITS	DINT	L#1	**System of units** Valid values: • 1 = 10^{-3} mm • 2 = 10^{-4} inch • 3 = 10^{-4} degrees • 4 = 10^{-2} degrees • 6 = 10^{-3} degrees

Standard system of units in the online help

To specify the minimum and maximum values we use the mm system of units. To define the limits in other systems of units, convert the values as shown below:

To convert from		calculate
mm → inch		Limit value(inches) = limit value (mm) × 0.1 [1]
mm → degrees	10^{-4} (4 post-decimal places)	Limit value (degrees) = limit value (mm) x 0.1
	10^{-3} (3 post-decimal places)	Limit value (degrees) = limit value (mm) x 1
	10^{-2} (2 post-decimal places)	Limit value (degrees) = limit value (mm) x 10
inches → degrees		Limit (degrees) = limit (inches) .

[1] When set to the maximum value, the number of decimal places affects the number of digits before the decimal point. In the inch measurement system, 4 decimal places are used; you can therefore enter a maximum of 100,000.0000 inches. The millimeter measurement system uses 3 decimal places; a maximum of 1,000,000.000 mm can be entered.

Connection between increments and system of units

The encoder signals of a connected encoder will be evaluated by the module and converted into the current system of units. The resolution (Page 2456) is used for the conversion.

If the positioning module

- has counted 10 increments and
- a resolution of 100 µm per increment is assigned in the encoder data,

this means that the axis will be moved by a distance of 1 mm.

Copying channels (S7-300, S7-400)

Copying channels

You can copy the parameters of one channel to another.

1. Select the channel whose parameters are to be copied (source channel).
2. Select the channel to which the parameters are to be copied (target channel).
3. Press the "Copy" button.

Channel n drive (S7-300, S7-400)

Selecting control mode (S7-300, S7-400)

Overview of control modes (S7-300, S7-400)

Overview

The control mode specifies how a motor is operated at the four digital outputs (Q) of a channel by means of the power module. There are 4 control modes per channel for the positioning module:

- Control mode 1 (Page 2437)
- Control mode 2 (Page 2438)
- Control mode 3 (Page 2438)
- Control mode 4 (Page 2439)

The following table shows the states of the 4 digital outputs for each control mode, x stands for channel 1 and 2.

	Rapid traverse		Creep speed		PR Position reached stop
	Direction +	Direction -	Direction +	Direction -	
Control mode 1					
xQ0	1	1	0	0	-
xQ1	0	0	1	1	-
xQ2	1	0	1	0	-
xQ3	0	1	0	1	-
Control mode 2					
xQ0	1	1	0	0	0
xQ1	0	0	0	0	1
xQ2	1	0	1	0	0
xQ3	0	1	0	1	0
Control mode 3					
xQ0	1	1	0	0	-

	Rapid traverse		Creep speed		PR Position reached stop
	Direction +	Direction -	Direction +	Direction -	
xQ1	1	1	1	1	-
xQ2	1	0	1	0	-
xQ3	0	1	0	1	-
Control mode 4					
xQ0	1	0	0	0	-
xQ1	1	0	1	0	-
xQ2	0	1	0	0	-
xQ3	0	1	0	1	-

Control mode 1 (S7-300, S7-400)

Control mode 1

The following settings apply for control mode 1 (x stands for channel n):

- xQ0: Rapid traverse
- xQ1 Creep speed
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 1.

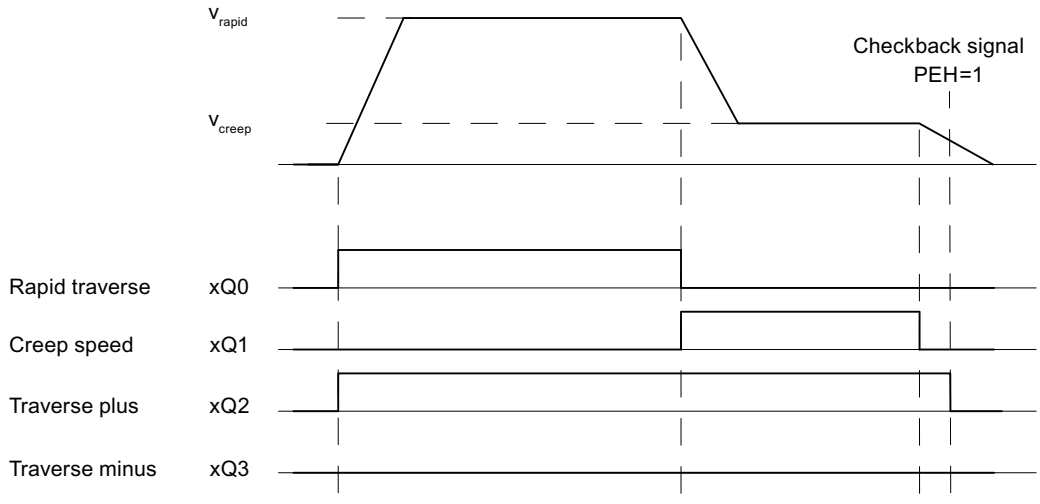

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	1 = Control mode 1

Control mode 2 (S7-300, S7-400)

Control mode 2

The following settings apply for control mode 2 (x stands for channel n):

- xQ0: Rapid traverse/creep speed
- xQ1 Position reached
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 2.

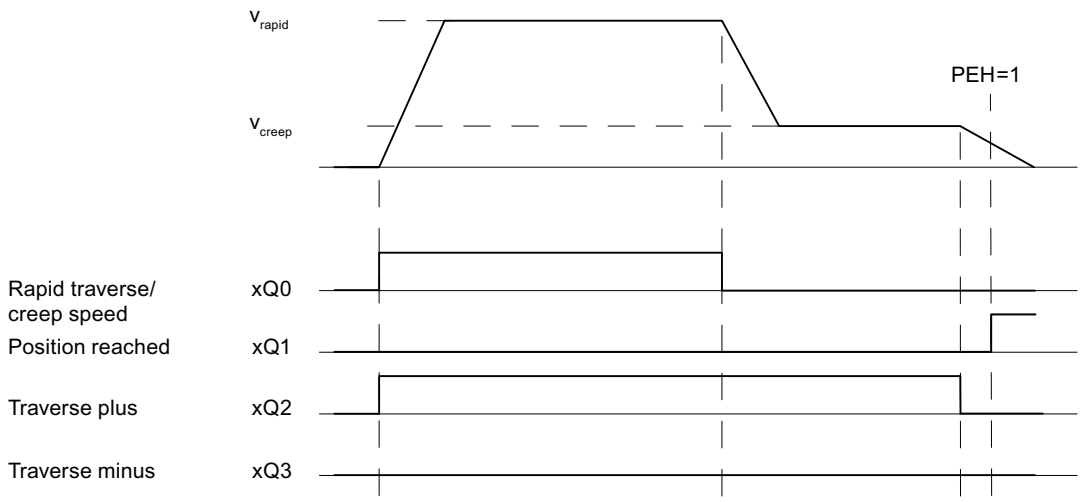

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	2 = Control mode 2

Control mode 3 (S7-300, S7-400)

Control mode 3

The following settings apply for control mode 3 (x stands for channel n):

- xQ0: Rapid traverse
- xQ1 Creep speed
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 3.

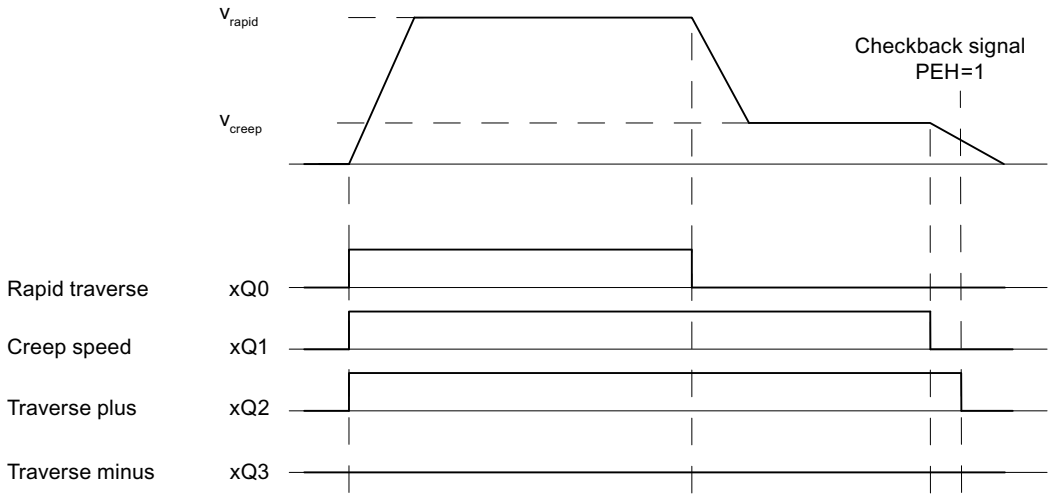

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	3 = Control mode 3

Control mode 4 (S7-300, S7-400)

Control mode 4

The following settings apply for control mode 4 (x stands for channel n):

- xQ0: Rapid traverse plus
- xQ1 Creep speed plus
- xQ2 Rapid traverse minus
- xQ3 Creep speed minus

The following image shows the function of the digital outputs with control mode 4.

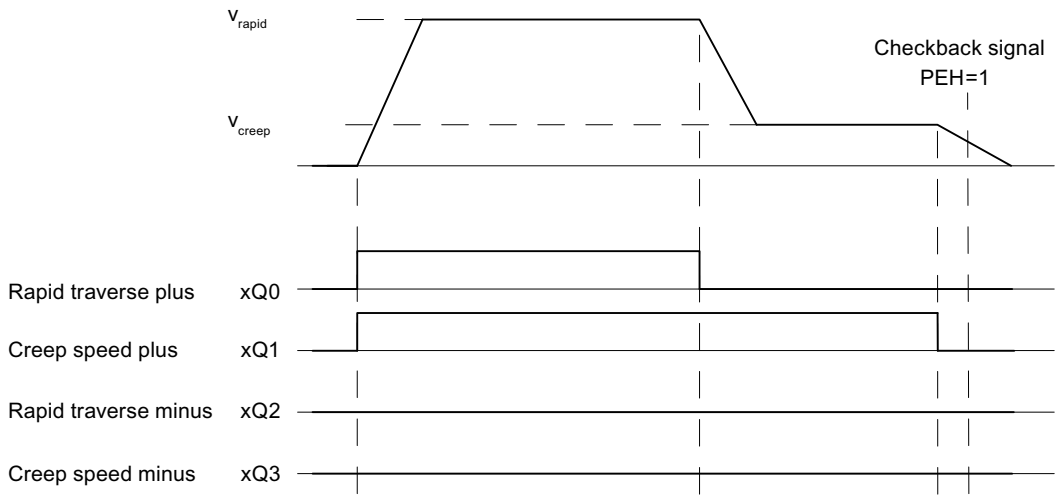

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	4 = Control mode 4

Specifying changeover difference (S7-300, S7-400)

Changeover difference

The changeover difference defines the distance from the destination at which the drive is switched over from rapid feed to creep feed.

The specified values apply to all targets approached by the positioning module.. An incremental approach with the incremental dimension 255 is an exception. A special changeover difference is specified for this.

Rules for changeover difference

- The changeover difference plus and the changeover difference minus do not have to be arranged symmetrically around the target. The values can differ.
- The changeover difference must be greater than the switch-off difference.
- The changeover difference must lie within the operating range.
- The changeover difference must be less than the end of rotary axis.
- Select a sufficient distance between the changeover point and the switch-off point to ensure that the drive can safely switch from rapid traverse to creep speed.
- The changeover point and the switch-off point must be separated from each other by a time interval of at least 8 ms.

The following image shows the location of the switch areas around the target.

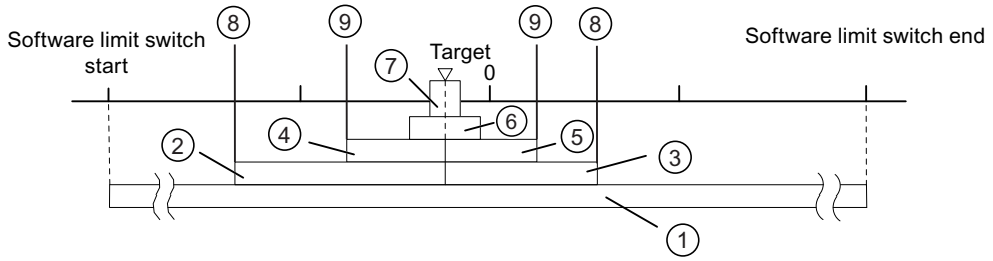

① Operating range
② Changeover difference plus / minus
③
④ Switch-off difference plus / minus
⑤
⑥ Standstill range
⑦ Target range
⑧ Changeover point
⑨ Switch-off point

For more information regarding the arrangement of the ranges, please see "Ranges and switching points of the positioning modules".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CHGDIF_P	DINT	L#5000	**Changeover difference plus**
104.0	CHGDIF_M	DINT	L#5000	**Changeover difference minus** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying switch-off difference (S7-300, S7-400)

Switch-off difference

The "switch-off difference" defines the distance to the destination at which the drive (creep speed) is switched off. The positioning module then adopts the monitoring functions from this point on.

The specified values apply to all targets approached by the positioning module.. An incremental approach with the incremental dimension 255 is an exception. A special switch-off difference is specified for this.

Rules for switch-off difference

- The switch-off difference plus and the switch-off difference minus do not have to be arranged symmetrically around the target. The values can differ.
- The switch-off difference must be greater than half the target range.

- Select the distance between the switch-off point and the target to ensure that the drive comes to a standstill within the target range.
- The switch-off point and the start of the target range must be separated from each other by a time interval of at least 8 ms.

The following image shows the location of the switch areas around the target.

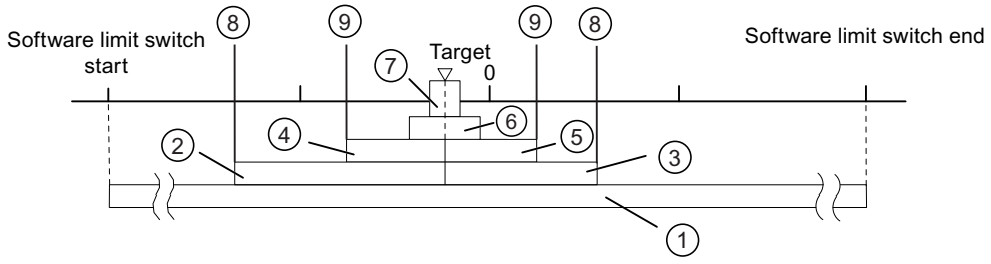

①	Operating range	⑦	Target range
②	Changeover difference plus / minus	⑧	Changeover point
③			
④	Switch-off difference plus / minus	⑨	Switch-off point
⑤			
⑥	Standstill range		

For more information regarding the arrangement of the ranges, please see "Ranges and switching points of the positioning modules".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
108.0	CUTDIF_P	DINT	L#2000	**Switch-off difference plus**
112.0	CUTDIF_M	DINT	L#2000	**Switch-off difference minus** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying target approach (S7-300, S7-400)

Specifying target range (S7-300, S7-400)

Target range

The positioning module sets a symmetrical range around each target, the target range. The actual value must lie within this range with a target approach.

A value specification of "0" switches off the tolerance during the target approach. The speed when reaching the target must then be less than the assigned standstill velocity (Page 2445) parameter.

Rules for the target range

- The target range lies symmetrically around the target.
- A value specification of "0" switches off the monitoring of the target range.

The following image shows the location of the ranges around the target.

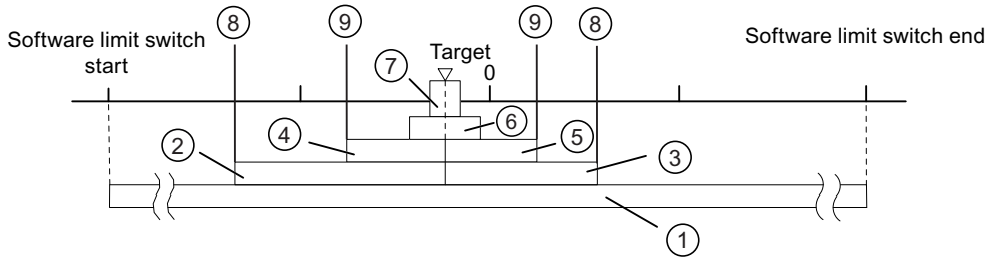

① Operating range
② Changeover difference plus / minus
③
④ Switch-off difference plus / minus
⑤
⑥ Standstill range
⑦ Target range
⑧ Changeover point
⑨ Switch-off point

Additional information on target approach can be found under "Ranges and switching points of the positioning modules" as well as under "End of a positioning operation (Page 2401)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	TRG_RANGE	DINT	L#1000	**Target range** Valid value range: • 0 = no monitoring • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying monitoring time (S7-300, S7-400)

Monitoring time

The module monitors the following during the monitoring time:

- the movement of the axis up to the switch-off point.
  The monitoring time starts with the start of a positioning and is retriggered with each actual value change in the traversing direction.
  Within the monitoring time, the actual value must have changed by at least one increment (the distance of the resolution).
- the target approach.
  The monitoring time is retriggered when the switch-off difference is reached for the last time.
  The target range must be reached within the monitoring time.

Additional information on target approach can be found under "End of a positioning operation (Page 2401)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
80.0	MON_TIME	DINT	L#2000	**Monitoring time** Valid value range: • 0 = no monitoring • 1 to 100 000 ms
Actual monitoring time You can specify all values from the range of values for the monitoring time. The module rounds the specified time to a multiple of 8 ms (module cycle).				

Specifying standstill range (S7-300, S7-400)

Standstill range

The standstill range lies symmetrically around the respective target. In this range, monitoring is performed to determine whether the drive remains on an approached target position or drifts away from it. The monitoring begins when the positioning module has reported "Position reached".

If the standstill range is exited without a valid traversing job, the positioning module signals an error.

Rules for standstill range

- The standstill range lies symmetrically around the respective target.
  (Target – ½ × standstill range) ≤ target ≤ (target + ½ × standstill range)
- The standstill range must be greater than or equal to the target range.

- The coordinate values for the standstill range must lie in the operating range.
- A setting of "0" deactivates the standstill monitoring function.

Additional information on target approach can be found under "Ranges and switching points of the positioning modules" as well as under "End of a positioning operation (Page 2401)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	ZSPEED_R	DINT	L#1000	**Standstill range** Valid value range: • 0 = no monitoring • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying standstill velocity (S7-300, S7-400)

Standstill velocity

The standstill velocity serves as a reference speed. If, during a positioning operation, the actual speed reaches the standstill velocity and the drive is located in the target range, then "Position reached" (PR) is set. The positioning operation is therefore successfully completed.

The precision of the positioning increases when the value of the standstill velocity is reduced. A setting of "0" deactivates the standstill monitoring function.

Additional information on target approach can be found under "End of a positioning operation (Page 2401)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
88.0	ZSPEED_L	DINT	L#30000	**Standstill velocity** Valid value range: • 0 = no monitoring • 1 µm/min to 100 000 µm/min

Channel n axis (S7-300, S7-400)

Specifying axis type (S7-300, S7-400)

Selecting axis type (S7-300, S7-400)

Axis types

The positioning module can control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.

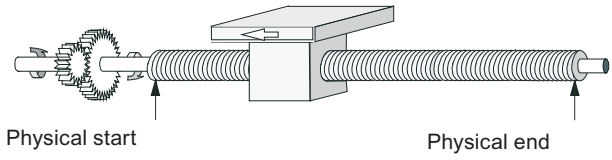

- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.

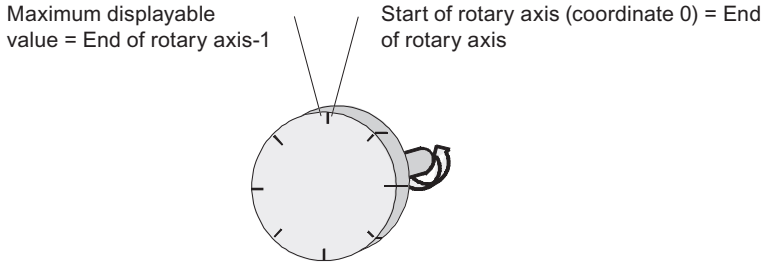

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
12.0	AXIS_TYPE	DINT	L#0	**Axis type** Valid values: • Linear axis = 0 • Rotary axis = 1

Specifying software limit switch (S7-300, S7-400)

Software limit switch

Software limit switches only exist for the linear axis.

The operating range is the range limited by the software limit switches.

The software limit switch start must always be smaller than the software limit switch end.

The software limit switches are monitored after the axis is synchronized.

- When using an incremental encoder:
  The synchronization of the axis is achieve by a reference point approach or by setting a reference point.
- When using an absolute encoder (SSI):
  The axis is synchronized when the module has received a complete, error-free frame for the associated channel.
  The absolute encoder must cover at least the operating range including the software limit switches.

Relationship between the operating range, encoder range and traversing range

- The operating range is the range which you specify for your job using the software limit switches.
- The encoder range represents the range explicitly covered by the encoder. For a linear axis, the module assigns the encoder range symmetrically across across the operating range. In other words, the module shifts the encoder range in order to equalize the distances between the software limit switches and the ends of the encoder range (see figure below).
- This traversing range represents the value range the positioning module can process. The traversing range is dependent on the resolution.

The following figure shows the relationship between software limit switches, the operating range, the encoder range and the traversing range.

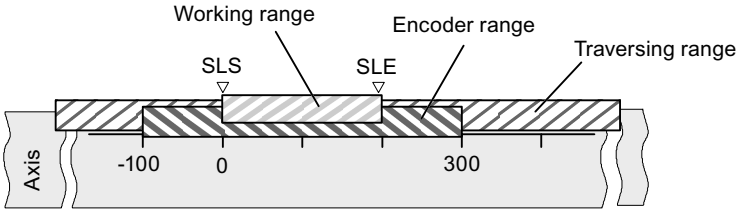

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L#-100000000	**Software limit switch start**
68.0	SSW_END	DINT	L#100000000	**Software limit switch end** Valid value range: • -1,000,000,000 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • -100,000,000 µm to 100,000,000 µm at resolution <1 µm/pulse

Specifying end of rotary axis (S7-300, S7-400)

End of rotary axis

The value "end of rotary axis" is the theoretical maximum value that the actual value can reach. The theoretically highest value is however never displayed, because it physically identifies the same position as the start of the rotary axis (0).

The maximum value that will be displayed for a rotary axis has the value:

End of rotary axis [µm] - resolution [µm / pulse] x 1 [pulse]

Example: End of rotary axis 1000 mm, resolution 1000 µm / pulse

The display jumps:

- with a positive rotational direction from 999 mm to 0 mm
- with a negative rotational direction from 0 mm to 999 mm.

Rotary axis with absolute encoders

The rotary axis range (0 to end of rotary axis) of a rotary axis with an absolute encoder must correspond exactly to the encoder range of the absolute encoder:
End of the rotary axis [µm] = number of encoder revolutions x distance/encoder revolutions [µm]

Data used in the DB parameter

Address	Name	Type	Initial value	Comment
16.0	ENDROTAX	DINT	L#100000	**End of rotary axis** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying reference point approach (S7-300, S7-400)

Selecting type of reference point approach (S7-300, S7-400)

Type of reference point approach

With "Type of reference point approach", you determine the conditions for the axis synchronization.

- The first specification defines the start direction in which the reference point approach is starting.
- The second specification defines the location of the zero mark, which leads to synchronization, as far as the reference point switch is concerned.

Detailed information can be found under "Reference point approach" operating mode (Page 2411).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
52.0	REFPT_TYPE	DINT	L#0	**Type of reference point approach** Valid values: • 0 = plus, reference point switch in plus direction • 1 = plus, reference point switch in minus direction • 2 = minus, reference point switch in plus direction • 3 = minus, reference point switch in minus direction

Specifying reference point coordinate (S7-300, S7-400)

Reference point coordinate

- **Incremental encoders:**
  You require the reference point coordinate for the "Reference point approach" mode. If the axis is not synchronized following writing and activating of machine data, the actual value is set to the value of the reference point coordinate.
- **Absolute encoder (SSI)**
  You require the reference point coordinates for the mechanical adjustment of the encoder. For more information, refer to the description of absolute encoder adjustment (Page 2450) which explains the interaction of the absolute encoder adjustment with other data.

The value of the reference point coordinate must lie within the operating range:

- **Linear axis**
  Including the software limit switches.
- **Rotary axis**
  Greater than or equal to 0 and less than the "end of rotary axis" value (0 ≤ reference point coordinate < end of rotary axis).

Data used in the DB parameter

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1 000 000 000 µm to 1 000 000 000 µm at resolution ≥1 µm/pulse • -100 000 000 µm to 100 000 000 µm at resolution < 1 µm/pulse

Specifying absolute encoder adjustment (S7-300, S7-400)

Determining the absolute encoder adjustment (S7-300, S7-400)

Absolute encoder adjustment

Absolute encoder adjustment is only necessary for an absolute encoder.

The absolute encoder adjustment and reference point coordinate map the encoder's range of values explicitly to the axis coordinate system.

Additional information can be found under "Absolute encoder adjustment procedure (Page 2450)" and in an example (Page 2451).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1 000 000 000 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • -100,000,000 µm to 100 000 000 µm at resolution <1 µm/pulse
48.0	ENC_ADJ	DINT	L#0	**Absolute encoder adjustment** Valid value range: • 0 to (2^{25} - 1)

Absolute encoder adjustment procedure (S7-300, S7-400)

Determining the correct absolute encoder adjustment

After initial parameter assignment, further steps must be taken to ensure that a correct relationship is created between the encoder and coordinate system. The procedure shown in the next section is based on the use of the parameter assignment dialogs.

1. Traverse the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch end".
2. Call the job "Set reference point (Page 2425)" with the coordinate of the point defined under 1.
   The positioning module now determines an encoder value for the reference point coordinate entered in the DB channel (REFPT in channel DB), namely the absolute encoder adjustment. You can read this value with the "Read encoder data (Page 2431)" job.
3. Enter the value read in the "absolute encoder adjustment" field.
4. Save your parameter assignment to the corresponding parameter DB using the export function.

5. Download the data in the HWCN to the CPU.
6. Restart the CPU (cold restart) to apply the data.

Note

You perform this adjustment just once during the commissioning. After a configuration, the position module is synchronized during start-up as soon as a complete, faultless message frame of the encoder is received after start-up.

Otherwise: Mechanical adjustment of the encoder

A correct relationship between the coordinate system and the encoder can be achieved as follows:

1. Move the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch start".
2. Enter this coordinate value in the machine data as "reference point coordinate".
3. Read the encoder value displayed at this position with the "Read encoder data (Page 2431)" job.
4. Enter this value in the machine data as the "absolute encoder adjustment".

After the parameter assignment a correct actual value will always be shown.

Instead of steps 3 and 4, you can also set the encoder to zero using "Reset", if available. Then enter the value "0" in the machine data as the absolute encoder adjustment.

Example of absolute encoder adjustment (S7-300, S7-400)

Example of an absolute encoder adjustment

The following assumptions apply to the example:

- Reference point coordinate = -125 mm
- Operating range from SSW_STRT = -1000 mm to SSW_END = 1000 mm
- Absolute encoder adjustment = 0
- Encoder range = 2048 increments with a resolution of 1 mm/pulse
- The absolute encoder used cannot be adjusted mechanically with precision and it also is not possible to set the encoder value specifically.

The following figure shows an example of absolute encoder adjustment.

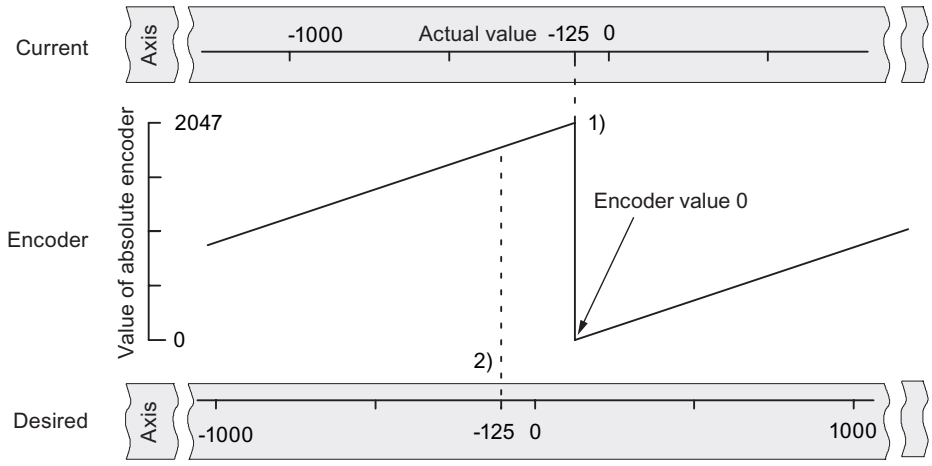

1) Assignment of the coordinate system to encoder values, based on the absolute encoder adjustment. The encoder value 0 corresponds to the actual value -125.

2) Required assignment of the coordinate system to the encoder. At this position the coordinate should be -125.

Result in accordance with "Set reference point"

After "Set reference point" the relationship looks as follows:

The reference point coordinate on the axis (-125) is assigned to the encoder value (1798) determined from absolute encoder adjustment. This assignment is set out in the figure below.

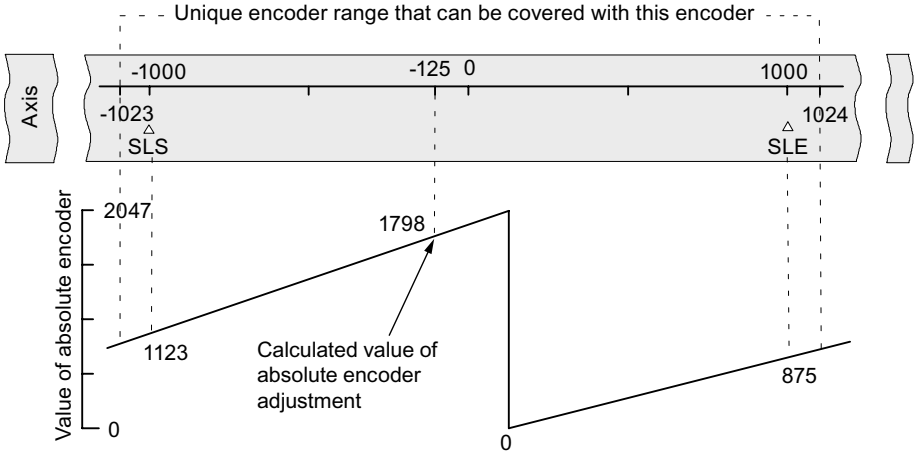

The encoder supplies 2048 unique values. The operating range is defined by the software limit switches. The selected resolution of 1 mm per pulse means the encoder can, however, cover a greater operating range than the one set by the software limit switches.

With the set resolution, the operating range is already covered with 2001 values. Therefore, in the example, 47 pulses "remain" and these are symmetrically positioned around the operating range.

Selecting start velocity (S7-300, S7-400)

Start speed in the case of a reference point approach

With this data you select the speed for the start of the reference point approach.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
99.0	REFPT_SPD	BOOL	TRUE	**Start speed for reference point approach** Valid values: • 0 = rapid traverse • 1 = creep speed

Channel n encoder (S7-300, S7-400)

Selecting the signal type (S7-300, S7-400)

Signal type

The encoder provides distance information to the module. The module then evaluates this information and converts this with the resolution into an actual value.

You can select the signal type from one of the following encoders:

- 5 V incremental encoder
  Select the signal type "5 V differential signal" to use a 5 V incremental encoder with differential signal A, /A, B, /B and N, /N.
- 24 V incremental encoder
  Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.
- Absolute encoder (SSI)
  Select the "SSI absolute" signal type if you use an absolute encoder with a serial interface.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
20.0	ENC_TYPE	DINT	L#1	**Signal type** Valid values: • 1 = 5 V differential signal • 2 = 24 V track A+B phase shifted • 3 = SSI 13-bit frame length [1)] • 4 = SSI 25-bit frame length [1)] • 5 = SSI 14-bit frame length [1)] • 6 = SSI 15-bit frame length [1)] • 7 = SSI 16-bit frame length [1)] • : • : • 14 = SSI 23-bit frame length [1)] • 15 = SSI 24-bit frame length [1)]

[1)] The frame length is selected in the "SSI" dialog box.

Selecting the count direction (S7-300, S7-400)

Count direction

With the machine data "Count direction" you can adapt the direction of the position measuring to the direction of movement of the axis. You must also take into account the rotational direction of all transmission elements (for example, couplings and gears).

- Normal = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent increasing actual position values
- Inverted = incremental count pulses (incremental encoder) or encoder values (absolute encoder) correspond to decreasing actual position values

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
59.0	CNT_DIR	BOOL	FALSE	**Count direction** Valid values: • 0 = normal • 1 = inverted

Specifying resolution (S7-300, S7-400)

Specify distance per encoder revolution (S7-300, S7-400)

Distance per encoder revolution

With the machine data "Distance per encoder revolution" you can inform the positioning module which distance the drive system should cover per encoder revolution.

The "Distance per encoder revolution" value depends on the structure of the axis and the way in which the encoder is mounted. You must take into account all the transfer elements like clutches or gearboxes. The following figure shows an example of two differently-mounted encoders.

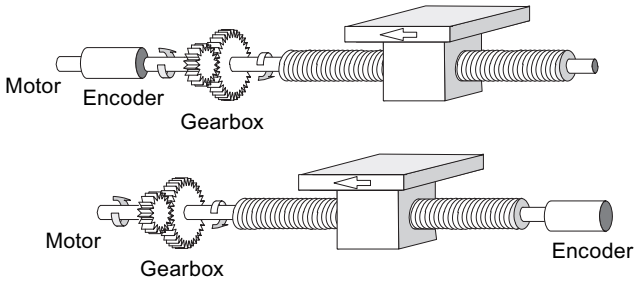

The relationship between the "Distance per encoder revolution" and "Increments per encoder revolution" machine data is described under resolution (Page 2456).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
24.0	DISP_REV	DINT	L#80000	**Distance per encoder revolution** Valid value range: • 1 µm to 1 000 000 000 µm

Specifying increments per encoder revolution (S7-300, S7-400)

Increments per encoder revolution

The machine data "Increments per encoder revolution" indicates the number of increments that an encoder makes for each revolution. The positioning module calculates the resolution (Page 2456) based on this value and the "Distance per encoder revolution" machine date.

Incremental encoder

Any value within the range of values can be entered. The module evaluates the increments four times.

Absolute encoder

For the limits you have to differentiate between the individual encoder models. Only values to the power of 2 are permitted as the entered value.

- Single-turn encoder (number of revolutions = 1) with 13-bit frame length:
  - minimum value = 4
  - maximum value = 8192
- Multiturn encoder (number of revolutions > 1) with 25-bit message frame length:
  - Minimum value = 4
  - Maximum value = 8192
- Single-turn encoder (number of revolutions = 1) with 25-bit frame length:
  - Minimum value = 4
  - maximum value = 2^{25}

Linear scales will be configured as multiturn encoders as follows:

- Increments per encoder revolution = 8192
- Number of revolutions x 8192 ≥ number of steps in the linear scale

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
32.0	INC_REV	DINT	L#500	**Increments per encoder revolution** Valid value range: • 1 to 2^{25}

Resolution (S7-300, S7-400)

Resolution

The resolution indicates the traversing distance corresponding to one pulse. It is a measurement for the precision of the positioning and also determines the possible maximum traversing range of the positioning module.

The resolution (RES) can be calculated as in the following table:

	Incremental encoder	Absolute encoder
Input values	• Distance per encoder revolution • Increments per encoder revolution: – Quadruple evaluation – 1 increment = 4 pulses	• Distance per encoder revolution • Increments per encoder revolution: – 1 increment = 1 pulse
Calculation	$RES = \frac{\frac{Distance}{Encoder\ revolution}}{\frac{Pulses}{Encoder\ revolution}}$	

Note

All position information is rounded up to the integral multiple of the resolution. This way the entered values differentiate from the used values.

Range of values of the resolution

The chosen system of units determines the range of values of the resolution:

System of units	Information in ...	Range of values of the resolution
mm	10^{-3} mm	$0{,}1 \times 10^{-3} \ldots 1000 \times 10^{-3}$ mm/pulse
inch	10^{-4} inch	$0{,}1 \times 10^{-4} \ldots 1000 \times 10^{-4}$ inch/pulse
degrees	10^{-4} degrees 10^{-3} degrees 10^{-2} degrees	$0{,}1 \times 10^{-4} \ldots 1000 \times 10^{-4}$ degrees/pulse $0{,}1 \times 10^{-3} \ldots 1000 \times 10^{-3}$ degrees/pulse $0{,}1 \times 10^{-2} \ldots 1000 \times 10^{-2}$ degrees/pulse

Example

- An incremental encoder has the following data:
  - Increments per encoder revolution: 5000
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 4 pulses

  This results in the resolution (4 x evaluation):

$$\text{Resolution} = \frac{1000\ \text{mm}}{5000\ \text{increments}} = 0.2000\ \frac{\text{mm}}{\text{Increment}} = 0.2000\ \frac{\text{mm}}{4\ \text{pulses}}$$

$$= 0.0500\ \frac{\text{mm}}{\text{Pulse}}$$

- An SSI encoder has the following data:
  - Increments per encoder revolution: 4096
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 1 pulse

  This results in the resolution:

$$\text{Resolution} = \frac{1000\ \text{mm}}{4096\ \text{increments}} = 0.2441\ \frac{\text{mm}}{\text{Increment}} = 0.2441\ \frac{\text{mm}}{\text{Pulse}}$$

Relationship between the traversing range and resolution

The traversing range is restricted by the numerical representation in the position module. This numerical representation varies depending on the resolution. You should thus make sure that your specifications are always within the permitted limits.

The maximum traversing range is illustrated in the following table:

Resolution (RES) lies in the range	Maximum traversing range
0.1 µm/pulse ≤ RES < 1 µm/pulse	-10^8 µm to 10^8 µm (-100 m to +100 m)
1 µm/pulse ≤ RES ≤ 1000 µm/pulse	-10^9 µm to 10^9 µm (-1000 m to +1000 m)

Selecting data for absolute encoder (SSI) (S7-300, S7-400)

Number of encoder revolutions (S7-300, S7-400)

Number of encoder revolutions

The machine date "Number of encoder revolutions" only applies to absolute encoders. As a result, enter the number of revolutions that are possible with this encoder.

The **overall number of steps of the encoder** is not machine data. It is calculated as follows:

Overall number of steps = increments/encoder revolution x number of revolutions

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
36.0	NO_REV	DINT	L#1	**Number of encoder revolutions** Valid value range: • 1 (single-turn encoder) • 2 to 4096 in powers of 2 (multiturn encoder)

Transmission rate (S7-300, S7-400)

Transmission rate

With the machine data "Transmission rate" you can determine the data transmission rate from the SSI encoder to the positioning module.

The maximum Transmission rate is dependent on the cable length:

- 200 m → 188 kHz
- 100 m → 375 kHz
- 40 m → 750 kHz
- 12 m → 1500 kHz

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
40.0	BAUDRATE	DINT	L#0	**Transmission rate** Valid values: • 0 = 188 kHz • 1 = 375 kHz • 2 = 750 kHz • 3 = 1500 kHz

Frame length (S7-300, S7-400)

Frame length

With "frame" you define the clock frame output by the positioning module for absolute encoder.

A frame between 13 bits and 25 bits is permitted.

Data used in the parameter DB

The frame length selected here is stored in the parameter DB under the ENC_TYPE (Page 2452) date in the parameter DB.

Parity (S7-300, S7-400)

Parity

With "Parity", you specify whether no parity, even parity or odd parity should be assigned for an absolute encoder.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
28.0	PARITY	DINT	L#0	**Parity** Valid values: • 0 = none • 1 = even parity • 2 = odd parity

Selecting monitoring (S7-300, S7-400)

Monitoring

Wire break

When monitoring is activated, the positioning module monitors all cables for a 5 V incremental encoder and an absolute encoder. The monitoring detects:

- Wire break
- Short circuit on individual cables
- Edge distance of counting pulses (also for 24 V incremental encoder)

For monitoring for a 24 V incremental encoder, you must set MON_TIME > 0 as the monitoring time (Page 2443).

With 5 V incremental encoders with zero marks, you must disable the wire break monitoring or interconnect the N and /N signals externally.

Frame error

The module monitors the message frame of an absolute encoder (SSI) for:

- Start and stop bit errors

You cannot disable the message frame error monitoring with absolute encoders (SSI).

Missing pulses (incremental encoder)

An incremental encoder must always return the same number of increments between two successive zero marks.

The positioning module checks to see if the zero mark of an incremental encoder is occurring at the correct encoder state.

For encoders without zero marks you must switch off the monitoring for missing pulses.

You must also disable the wire-break monitoring or interconnect the N and /N zero mark inputs externally.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
				Monitoring:
63.0	MON_WIRE	BOOL	TRUE	1 = wire break
63.1	MON_FRAME	BOOL	TRUE	1 = message frame error (must always be 1)
63.2	MON_PULSE	BOOL	TRUE	1 = missing pulses

Channel n incremental dimensions (S7-300, S7-400)

Specifying incremental dimensions (S7-300, S7-400)

Definition

Incremental dimensions are target specifications that can be controlled by the positioning module with the "relative incremental approach" or rather "absolute incremental approach" mode.

Possible range for incremental dimensions

The target that is to be approached must be at a distance of at least one-half the target range in front of the respective software limit switch. The following figure shows the limits for specifying incremental dimensions.

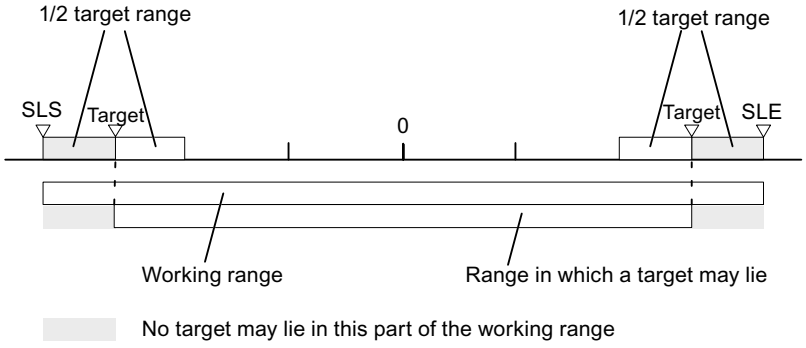

Increment number 1 to 100 (S7-300, S7-400)

Incremental dimension number 1 to 100

You have the possibility of entering up to 100 incremental dimensions in a table that are valid both for the "Relative incremental approach" mode as well as for the "Absolute incremental approach" mode.

Note that the positioning module does not permit negative values for the "relative incremental approach". The values will be interpreted by the positioning module - depending on the direction of movement - as either a positive or a negative difference.

Note

The entry is made in the unit of the system of units set. Note the decimal places in particular here.

Numerical example:

- Incremental dimension: 800 mm
- System of units: 10^{-3} mm
- Entry in the DB parameter: 800000

Tip: Define separate areas for relative and absolute incremental dimensions in the incremental dimension table.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.4	TRGL1WR_EN	BOOL	FALSE	1 = write increment table 1 (increments 1 ... 50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = write increment table 2 (increments 51 ... 100)

Address	Name	Type	Initial value	Comment
36.6	TRGL1RD_EN	BOOL	FALSE	1 = read increment table 1 (increments 1 ... 50)
36.7	TRGL2RD_EN	BOOL	FALSE	1 = read increment table 2 (increments 51 ... 100)

Data used in the parameter DB

Address	Name	Type	Initial value	Comment	
120.0	TRGL1.TRG[1]	DINT	L#0	Incremental dimension number 1	Increment table 1
. . .	. . .	. .	. . .	. . .	
316.0	TRGL1.TRG[50]	DINT	L#0	Incremental dimension number 50	
320.0	TRGL2.TRG[51]	DINT	L#0	Incremental dimension number 51	Increment table 2
. . .	. . .	. . .	. . .	. . .	
516.0	TRGL2.TRG[100]	DINT	L#0	Incremental dimension number 100	

Increment number 254 (S7-300, S7-400)

Incremental dimension number 254

Independently of the increment table you can use the increment number 254 as an additional distance specification. For the changeover and switch-off differences, the entries from the parameter DB are valid for this increment.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.2	TRG252_254_EN	BOOL	FALSE	1 = write increment for increment number 254
96.0	TRG252_254	DINT	L#0	Increment for increment number 254

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus

Increment number 255 (S7-300, S7-400)

Incremental dimension number 255

The incremental dimension number 255 gives you an additional guideline for the path.

The switch-off differences and the changeover differences are presented together with the increment. Unlike the situation with the other increments, the increment 255 uses the values established in the channel DB for the switch-off and changeover differences. The entries from the machine data have no validity for this increment.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.3	TRG255_EN	BOOL	FALSE	1 = write increment for increment number 255
100.0	TRG255	DINT	L#0	Increment for increment number 255
104.0	CHGDIF255	DINT	L#0	Changeover differences for increment number 255
108.0	CUTDIF255	DINT	L#0	Switch-off differences for increment number 255

Control and feedback interface of the FM 351 (S7-300, S7-400)

Application

In special applications or in an alarm level, a particularly fast access to checkback and control signals could be required. You can reach this data directly via the input and output areas of the module.

To coordinate startup following each module startup, for example, after plugging a module or after a STOP-RUN transition of the CPU, you must call the ABS_CTRL instruction until the end of the startup is indicated by RETVAL = 0. Then, do not use the ABS_CTRL anymore.

Note

It is not possible to use the ABS_CTRL together with a write access.

Reading checkback signals by means of direct access

The byte addresses must be specified relative to the start address of the outputs of the respective channel. The names of the parameters correspond to the names in the channel DB.

Start address channel 1 = start address of the module

Start address channel 2 = start address of the module + 8

In STL, you access the data using the commands PEB (read 1 byte), PEW (read 2 bytes) and PED (read 4 bytes).

The following table shows the assignment of the feedback interface (inputs) of FM 351.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	PARA	Internal	Internal	DATA_ERR	OT_ERR	DIAG	Internal	Internal
Byte 1	CHGOVER	CUTOFF	ZSPEED	SPEED_OUT	0	WAIT_EI	WORKING	ST_ENBLD
Byte 2	MODE_OUT							
Byte 3	POS_RCD	0	0	0	GO_P	GO_M	0	SYNC
Byte 4	ACT_POS							
Byte 5								
Byte 6								
Byte 7								

Example: Position process value ACT_POS

The start address of the module is 512.

STL	Description

L PED 516	//Read the current position process value (ACT_POS) from channel 1
	//with direct access:
	//Start address of the channel + 4

Writing control signals using direct access

The byte addresses must be specified relative to the start address of the inputs of the respective channel. The names of the parameters correspond to the names in the channel DB.

Start address channel 1 = start address of the module

Start address channel 2 = start address of the module + 8

In STL, you access the data using the commands PAB (write 1 byte), PAW (write 2 bytes) and PAD (write 4 bytes).

The following table shows the assignment of the control interface (outputs) of FM 351.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	OT_ERR_A	0	0	0
Byte 1	DRV_EN	0	0	0	DIR_P	DIR_M	STOP	START
Byte 2	MODE_IN							
Byte 3	MODE_TYPE							
Byte 4	Reserved							
Byte 5								
Byte 6								
Byte 7								

Example: START signals of channel 2

The start address of the module is 512.

STL	Description
L 2#10001000	//Set DRV_EN and DIR_P to 1
T PAB 521	//Write signals with direct access for channel 2:
	//Start address of the module + 8 + 1

Parameter transmission paths (S7-300, S7-400)

Parameters include the following machine data and increments.

The following figure shows the parameter transmission paths of the FM x51.

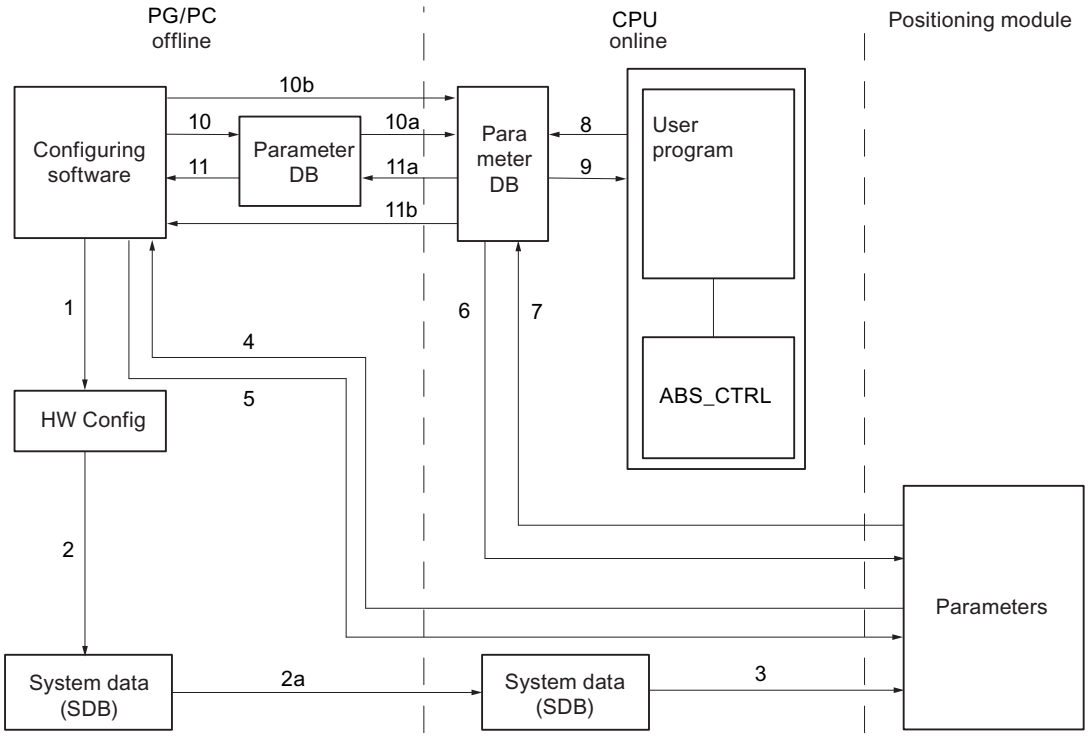

1 Saving parameters in the configuration software.

2 Save and compile the HW configuration.

2a Download the HW configuration to the CPU. The CPU automatically carries out step 3.

3 The CPU writes the parameters when system parameter assignments are carried out for the module.

4 With the command "download target system to PG" download the parameters of a channel of the module to the PG.

5 With the command "download target system" download the parameters from the configuration software into a channel of the module.

6 Write parameters using jobs of the user program in a channel of the module.

7 Read parameters using jobs of the user program from a channel of the module.

8 Store parameters from the user program into the online DB.

9 Read in parameters from the online DB into the user program.

10 Export parameters from the configuration software to the offline DB.

10a Download the offline DB to the CPU.

10b Export parameters from the configuration software to the online DB.

11 Import parameters from the offline DB into the configuration software.

11a Load parameters from the online DB to the PG.

11b Import parameters from the online DB into the configuration software.

Some use cases for the transfer of parameters

Use case	Steps
You process the parameters using the configuration software. The channels of the module should then be automatically assigned upon start-up.	Carry out steps 1, 2, and 2a.
You change the parameters during commissioning in test mode in the configuration software.	Carry out steps 4 and 5.
The parameters edited during commissioning should be downloaded automatically during startup.	Carry out steps 1, 2, and 2a.
You create the parameters using the configuration software. The channels of the module should be assigned upon start-up only by the user program via the data blocks.	Carry out steps 10, 10a and 6 or 10b and 6.
You would like to create conveniently stored data for recipes.	Carry out steps 10 and 10a.
You create the parameters using the configuration software. These should be made available to the user program for temporary changes.	Carry out steps 1, 2 and 2a for the automatic parameter assignment.
	Carry out the steps 10, 10a, 9, 8 and 6 for access by the user program.
You change existing parameters solely via the user program.	Carry out steps 7, 9, 8 and 6.
You would like to use the configuration software to view the data that has been changed with the user program.	Carry out steps 11a and 11 or just 11b.
Parameters modified by the user program should also be loaded automatically during startup.	Carry out steps 11b or 11a, 11 and then 1, 2, 2a.

6.2.5.5 Using FM 351 from the user program (S7-300, S7-400)

Integrating the FM 351 into a user program

Instructions are available for integrating the FM 351 into a user program. These enable you to handle the desired functions with ease.

The instructions ABS_INIT and ABS_CTRL can be called in the CPU user program. They ensure communication between the CPU and the FM 351. The channel DB is created automatically as the instance DB for the ABS_CTRL instruction when you insert the ABS_CTRL instruction into the project.

All the parameter data associated with the channel of the module (machine data, incremental dimension table) is located in the parameter DB for the module. Use the command "Program blocks > Add new block" to add a ABS_PARATYPE type data block to your project.

The instruction ABS_DIAG is also available for FM 351. This operation allows you to transfer the diagnostic information to the diagnostic DB. The diagnostic DB is created automatically as the instance DB for the ABS_DIAG instruction when you insert the ABS_DIAG instruction into the project.

Commissioning the module

The module can be initially tested using the "Commissioning" dialog, even when no user program is available. Note the following warning in particular in this case.

WARNING

Injury and material damage could occur.

To prevent personal injury and material damage, observe the following points:

- Install an emergency stop switch in the vicinity of the computer.
  This is the only way you can ensure that the plant can be safely shut down in the event of computer or software failure.
- Install hardware limit switches, which act directly on the power units of all drives.
- Make sure that no one has access to the area of the plant in which moving parts are present.
- When the FM 351 is controlled and monitored simultaneously by your program and the "Commissioning" dialog, conflicts can occur, the effects of which are not clear. Therefore, always set the CPU to STOP mode or disable your program when you work with the "Commissioning" dialog.

6.2.5.6 Commissioning the FM 351 (S7-300, S7-400)

6.2.5.7 FM 351: Online & Diagnostics (S7-300, S7-400)

Current module information (S7-300, S7-400)

Default diagnostics (S7-300, S7-400)

Error display through error LEDs of FM 351 (S7-300, S7-400)

Where is the error displayed?

If the red group error LED is lit, an error occurred either at the module (internal error) or at the cable ports (external error).

The red LEDs CH 1 and CH 2 display a channel error.

The following figure shows the location of the error LEDs on the FM 351.

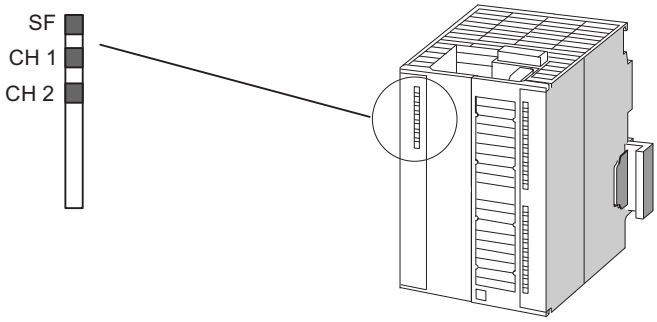

Which errors are displayed?

The table below shows the error types indicated by the error LEDs.

LED	Meaning	Explanations
SF	Group errors	This LED indicates the following error state of the FM 351: Diagnostic interrupt (internal or external (channel) errors) To remove the error, see Error classes (Page 2475).
CH 1 CH 2	Channel error 1 Channel error 2	These LEDs show a channel error on channel 1 or on channel 2. • Encoder wire break • Absolute encoder error • Missing pulses of Incremental encoder • Operating error • Programming error

Channel-specific diagnostics (S7-300, S7-400)

Triggering diagnostic interrupts (S7-300, S7-400)

Processing interrupts

The positioning module can trigger a diagnostic interrupt. These interrupts are processed in an interrupt OB. If an interrupt is triggered without the corresponding OB being loaded, the CPU switches to STOP.

Enable the processing of the diagnostic interrupts as follows:

1. Select the module in HWCN.
2. Select "Edit" > Object Properties > Basic Parameters" to enable the diagnostic interrupt.
3. Save and compile the configuration.
4. Download the configuration to the CPU.

Overview of diagnostic interrupts

The following events and errors trigger a diagnostic interrupt:

- Operating error
- Machine data error
- Diagnostics errors

These errors are explained in detail in "Error classes (Page 2475)".

Reaction of the positioning module to an error with diagnostic interrupt

- The positioning is canceled.
- The synchronization is deleted in the case of the following diagnostic interrupts:
  - Front connector missing, external auxiliary voltage for encoder supply missing
  - Zero mark error detected, cable fault (5 V encoder signals)
  - Traversing range exceeded (indicated by an operating error)
  - Set process value cannot be executed (indicated by an operating error).
- With one exception, control signals START, DIR_P und DIR_M are no longer processed.
  Exception:
  In the case of an operating error it is still possible to jog in the direction of the operating range.
- Function switches and jobs will no longer be processed.

Positioning module detects an error ("incoming")

A diagnostic interrupt is "incoming", if at least one error is given. If not all errors are remedied, the remaining existing errors will be reported again as "incoming".

Sequence:

1. The positioning module detects one or more errors and generates a diagnostic interrupt. The "SF" LED and, depending on error, the "CH1" / "CH2" LEDs illuminate. The error is then entered in the diagnostic buffer.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. You can evaluate the start info of OB 82.
4. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
5. For additional information, call the ABS_DIAG instruction.

The positioning module detects the transition to the error-free state ("outgoing")

A diagnostic interrupt is then only "outgoing" if the last error on the module has been remedied.

Sequence:

1. The positioning module detects that all errors have been eliminated and triggers a diagnostic interrupt. The "SF" LED is not lit. The diagnostic buffer is not changed.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
4. Evaluate the OB82_MDL_DEFECT bit.
   If this bit is "0", there are no errors on the module. You can end your evaluation here.

Diagnostic interrupts in dependence on the CPU status

- Diagnostic interrupts are blocked by the positioning module when the CPU is in STOP mode.
- If not all of the pending errors are eliminated while the CPU is in STOP mode, the positioning module reports the remaining errors as "incoming" events again at the next CPU transition to RUN.
- If all pending errors are eliminated while the CPU is in STOP mode, the module does **not** report its error-free state with a diagnostic interrupt at the next CPU transition to RUN.

Error evaluation in the user program (S7-300, S7-400)

Error response in the user program

In the user program you can respond selectively to errors. The following methods can be used:

- The RETVAL return values of the integrated standard FB:
  This value is determined again each time the block is called.
  RETVAL = -1 is a group display for a synchronous error in a job or when communicating with the module.
- An _ERR error bit belongs to each job. This is a group display for an error in the job or one of its predecessors in the job chain:
  The error bit is set for a write job and its subsequent jobs when a data error is reported by the module or a communication error occurs.
  With reading jobs the error bit is set for the job in question when a communication error occurs.
  The ABS_CTRL instruction resets the error bits once a job has been processed. These should however be cancelled by the user program in the case of an error assessment.
- The DATA_ERR checkback signal as a group display for an error that the module has detected during a write job. The signal will be established again with the next write job.
- The OT_ERR checkback signal (operator error) as a group display for an error that the module has detected when starting a traverse. The error must be acknowledged after the cause has been acknowledged with OT_ERR_A = 1.
- The DIAG checkback signal is set when the contents of the diagnostic buffer have changed. This signal can come along later than the signals DATA_ERR und OT_ERR.

- The communication error JOB_ERR includes the error code (Page 2474) in the case of a communication problem between the instruction and the module. The value is determined again after a job has been processed and is stored in the channel DBI for the ABS_CTRL operation and in the diagnostic DB for the ABS_DIAG operation.
- The ABS_DIAG instruction for reading the diagnostic buffer of the module. Here you can discover the error causes for synchronous and asynchronous events.
- Diagnostic interrupts for the fast response to events in the diagnostic interrupt OB (e.g. OB 82).

Evaluation of a diagnostic interrupt in the user program

The positioning module sets the following entries in the local data of the diagnostic interrupt OB (e.g. OB 82). The errors will also be entered in the diagnostic buffer (error class 128 (Page 2482)).

Address	Name	Type	Comment
0.0	OB82_EV_CLASS	BYTE	Event class and identifiers: B#16#38: Outgoing event B#16#39: Incoming event
1.0	OB82_FLT_ID	BYTE	Error code (B#16#42)
2.0	OB82_PRIORITY	BYTE	Priority class: B#16#1A in RUN B#16#1C in START-UP
3.0	OB82_OB_NUMBR	BYTE	OB number (82)
4.0	OB82_RESERVED_1	BYTE	Reserved
5.0	OB82_IO_FLAG	BYTE	Input module: B#16#54
6.0	OB82_MDL_ADDR	INT	Logical base address of the module in which the error has occurred
8.0	OB82_MDL_DEFECT	BOOL	Module fault
8.1	OB82_INT_FAULT	BOOL	Internal error
8.2	OB82_EXT_FAULT	BOOL	External error
8.3	OB82_PNT_INFO	BOOL	Channel error
8.4	OB82_EXT_VOLTAGE	BOOL	Missing external auxiliary supply
...	Not used		
10.3	OB82_WTCH_DOG_FLT	BOOL	Time monitoring has responded
...	Not used		
12.0	OB82_DATE_TIME	DATE_AND_TIME	Date and time at which the OB was requested

Diagnostics buffer of the module (S7-300, S7-400)

Diagnostic events

The module´s diagnostic buffer includes maximum 9 diagnostic events and is organized as a ring buffer.

A diagnostic event is written in the buffer if an "incoming" (error) message is detected. This can be a message, a synchronous error (data error, operator error) or even an asynchronous error (operational error and diagnostics error). From a cause of error several entries can occur as follow-up errors. Outgoing messages do not create any entries in the diagnostic buffer.

For each diagnostic event the following is given:

- Status (always incoming)
- Internal error
- External error
- Error class
- Error number
- Channel number
- Incremental dimension number (in the case of increment table errors)

When a diagnostic event is written in the diagnostic buffer, the DIAG = 1 checkback signal is set in all assigned channels.

Using the ABS_DIAG instruction, the diagnostic buffer can be transferred as a whole to a data block (diagnostic DB) or displayed by the error evaluation dialog of the configuration software. If the diagnostic buffer is read, the module sets the checkback signal DIAG = 0.

Note

If the diagnostic buffer is read simultaneously from the ABS_DIAG operation and the error evaluation dialog, it is possible that a new diagnostic event not be detected by the program.

Data and structure of the diagnostic DB (S7-300, S7-400)

Data and structure of the diagnostic DB

Note

You must not change any data that is not listed in this table.

The following table shows the structure of the diagnostic DB:

Address	Name	Type	Initial value	Comment
0.0	MOD_ADDR	INT	0	Module address
256.0	JOB_ERR	INT	0	Error number of the communication error
258.0	JOBBUSY	BOOL	FALSE	1 = job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = definitely read diagnostics buffer
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostics data - latest entry
272.0	DIAG[2]	STRUCT		Diagnostics data - second entry
282.0	DIAG[3]	STRUCT		Diagnostics data - third entry

Address	Name	Type	Initial value	Comment
292.0	DIAG[4]	STRUCT		Diagnostics data - fourth entry
302.0	DIAG[5]	STRUCT		Diagnostics data - fifth entry
312.0	DIAG[6]	STRUCT		Diagnostics data - sixth entry
322.0	DIAG[7]	STRUCT		Diagnostics data - seventh entry
332.0	DIAG[8]	STRUCT		Diagnostics data - eighth entry
342.0	DIAG[9]	STRUCT		Diagnostics data - ninth entry

The diagram below shows the structure of a DIAG[n] diagnostic entry.

Address	Name	Type	Initial value	Comment
+0.0	STATE	BOOL	FALSE	0 = incoming event 1 = outgoing event
+0.1	INTF	BOOL	FALSE	1 = internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 2: Operator errors 4: Data error 5: Machine data error 6: Incremental dimension table error 15: Alarms 128: Diagnostics errors
+4.0	FNO	INT	0	Error number
+6.0	CH_NO	INT	0	Channel number
+8.0	TRG_NO	INT	0	Incremental dimension number

List of JOB_ERR messages (S7-300, S7-400)

JOB_ERR alarms

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgement when reading from the module. The module has been removed during the read operation - or the module is faulty.
80A1	32929	-32607	Negative acknowledge when writing to the module. The module has been removed during the write operation - or the module is faulty.
80A2 [1)]	32930	-32606	Protocol error at layer 2
80A3 [1)]	32931	-32605	Protocol error involving user interface/user
80A4 [1)]	32932	-32604	Communication on C bus interrupted
80B0	32944	-32592	Data record / job unknown.
80B1	32945	-32591	Incorrect length specification. FM_TYPE parameter in the channel DB is not correctly set for the module being used.
80B2	32946	-32590	The configured slot is not assigned.
80B3	32947	-32589	Actual module type dissimilar to the set module type.

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80C0 [1]	32960	-32576	The module does not yet have ready the data to be read.
80C1 [1]	32961	-32575	The data of a same-named write job is not yet ready on the module.
80C2 [1]	32962	-32574	The module is currently processing the possible maximum of jobs.
80C3 [1]	32963	-32573	Required equipment (memory etc.) is currently in use.
80C4 [1]	32964	-32572	Communication error
80C5 [1]	32965	-32571	Distributed I/O unavailable.
80C6 [1]	32966	-32570	Priority class cancellation (restart or background)
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read from the DB. (write job)
8532	34098	-31438	DB number of the parameter DB too high (write job)
853A	34106	-31430	Parameter DB does not exist. (write job)
8544	34116	-31420	Error with the n-th (n > 1) read access on a DB after an error has occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written in the DB. (read job)
8730	34608	-30928	Parameter DB in the CPU is write-protected. The data cannot be written in the DB (read job).
8732	34610	-30926	DB number of the parameter DB too high (read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error with the n-th (n > 1) write access on a DB after an error has occurred. (read job)

[1] Errors 80A2 to 80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time.

Error classes (S7-300, S7-400)

Operating error (S7-300, S7-400)

Error class 1: Operating error

Operating errors are detected asynchronously to an operation / control. The operating errors bring about a cancellation of the positioning, apart from with error number 9, which causes the positioning to shut down.

The following table shows the possible operating errors and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Software limit switch start overrun**		Yes
	Cause	The actual value lies outside of the operating range.	
2	**Software limit switch end overrun**		Yes
	Cause	The actual value lies outside of the operating range.	
3	**Traversing range start overrun**		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits belong with the traversing range).	

No.	Meaning		Diagnostic interrupt
4	**Traversing range end overrun**		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits also belong to the traversing range)	
5	**Error on target approach**		Yes
	Cause	Target range has not been reached within the monitoring time.	
6	**Standstill range exited**		Yes
	Cause	The actual value lies outside of the standstill range.	
7	**Positive feedback**		Yes
	Cause	Actual value change > ½ standstill range in the wrong direction.	
8	**Missing or too slight actual value change**		Yes
	Cause	No actual value change or an actual value change against the set direction within the monitoring time.	
9	**Target overrun (FM 451)**		Yes
	Cause	The target has been overrun with "Set actual value on-the-fly".	
10	**Target range overrun**		Yes
	Cause	Target range has been overrun after target approach.	
11	**Changeover point switched incorrectly**		Yes
	Cause	Axis swinging in changeover point.	
12	**Switch-off point switched incorrectly**		Yes
	Cause	Axis swinging between switch-off / return point.	
13	**Start of target range switched incorrectly**		Yes
	Cause	Axis oscillating in the target range.	
14	**Change greater than half the rotary axis range**		Yes
	Cause	The velocity/frequency is too high or there are faulty actual value jumps.	
15	**Change greater than the rotary axis range**		Yes
	Cause	The velocity/frequency is too high or there are faulty actual value jumps.	
16	**Increment for increment number 252 cannot be transferred (FM 451)**		Yes
	Cause	The increment cannot be transferred.	
17	**Increment for increment number 252 cannot be approached (FM 451)**		Yes
	Cause	The distance between the current actual position and the specified increment is less than the changeover difference or switch-off difference.	
18	**Incorrect increment for increment number 252 (FM 451)**		Yes
	Cause	The increment lies outside of the operating range.	

Operator errors (S7-300, S7-400)

Error class 2: Operator errors

Operator errors have been detected when changing the control signals in the user data area. The operator errors cause the positioning to shut down.

The following table shows the possible operator errors and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Impermissible operating mode**		No
	Cause	The chosen operating mode is inadmissible.	
3	**Inadmissible interface job**		No
	Cause	The selected signal is inadmissible in this operating mode.	
4	**Incorrect operating mode parameter**		No
	Cause	• In "jog" mode the velocity specification is not equal to the rapid traverse or creep speed. • In "incremental approach" operating mode, the incremental dimension is a value other than 1 - 100, 252, 254, or 255	
5	**Start enable not available**		No
	Cause	Start enable not available upon start.	
7	**Target / target range lies outside of the operating range**		No
	Cause	Specified or calculated target is outside the software limit switches.	
8	**Axis parameters not assigned**		No
	Cause	Incorrect machine data or no machine data has been assigned for the axis.	
9	**Axis not synchronized**		No
	Cause	The "incremental approach" mode is only possible with an already synchronized axis.	
10	**Target/incremental distance cannot be positioned**		No
	Cause	The distance between the current actual position and the specified target is less than the switch-off difference.	
17	**Reference point approach not possible**		No
	Cause	An SSI encoder is connected.	
18	**Relative or absolute incremental approach is not possible**		No
	Cause	The increment is invalid.	
19	**Switch-off difference is less than ½ the target range with increment number 255**		No
	Cause	The switch-off difference for the increment number 255 is less than half the target range.	
20	**Inadmissible travel in the specified direction**		No
	Cause	The distance to the software limit switch is insufficient	

Data error (S7-300, S7-400)

Error class 4: Data error

Operating errors are detected synchronously to an operation / control. Data errors do not result in an error response.

The following table shows the possible data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
6	Specified increment too great		No
	Cause	The value lies outside of ±100 m or ±1000 m. The increment/target must not be greater than the traversing range. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	
10	Faulty zero offset (FM 451)		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. The software limit switches are positioned after the zero offset outside of the traversing range (100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	Incorrect actual value		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
12	Incorrect reference point		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
20	Not permissible to activate machine data		No
	Cause	There are no new, error-free machine data on the module.	
27	Unauthorized bit-coded setting		No
	Cause	Bits that are not used and not described here are not equal to 0.	
29	Inadmissible bit coding		No
	Cause	Unused and, in this case, unwritten bits are not 0.	
34	Not possible to cancel set actual value		No
	Cause	With an SSI encoder and a linear axis, the location of the actual value would lie outside the operating range after making the setting.	
36	Incorrect changeover difference with the increment number 255		No
	Cause	The value is outside the permitted numeric range of ±100 m or ±1000 m. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	
37	Incorrect switch-off difference with the increment number 255		No
	Cause	The value is outside the permitted numeric range of ±100 m or ±1000 m. The switch-off difference must be less than the changeover difference.	
107	Axis parameters not assigned		No
	Cause	On the axis there is either no machine data or it has not been enabled.	
108	Axis not synchronized		No
	Cause	One of the "set actual value" and "set actual value on-the-fly" jobs has been triggered even though the axis is not synchronized.	

Machine data error (S7-300, S7-400)

Error class 5: Machine data error

The diagnostic interrupt is only triggered in the event of a faulty system data block (SDB). Machine data errors do not result in an error response.

The following table shows the possible machine data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
5	**Error in hardware interrupt setting**		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	**Incorrect minimum edge distance (FM 451)**		Yes
	Cause	You have entered a value < 0 or > 10^9 µm as the minimum edge-to-edge distance.	
7	**Incorrect system of units**		Yes
	Cause	The value for the system of units lies outside of the permissible range of 1 to 4 and 6.	
8	**Incorrect axis type**		Yes
	Cause	You have entered neither 0 nor 1 as an axis type.	
9	**Incorrect end of rotary axis**		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or 1 to 10^8 µm (depending on the resolution).	
10	**Incorrect encoder type**		Yes
	Cause	The value for the encoder type lies outside of the permissible range of 1 to 4.	
11	**Incorrect distance per encoder revolution**		Yes
	Cause	The value for distance per encoder revolution lies outside the permissible range of 1 to 10^9 µm (irrespective of the resolution).	
13	**Incorrect number of increments per encoder revolution** (see "Increments per encoder revolution (Page 2454)")		Yes
14	**Incorrect number of revolutions** (see "Number of encoder revolutions (Page 2457)")		Yes
15	**Incorrect transmission rate**		Yes
	Cause	You have specified a value for the transmission rate that lies outside the permissible range of 0 to 3.	
16	**Incorrect reference point coordinate**		Yes
	Cause	The coordinate lies outside the range from -100 m to +100 m or -1000 m to +1000 m (depending on the resolution). Linear axis: The coordinate lies outside of the operating range. Rotary axis: The coordinate is greater than the end of the rotary axis or < 0.	
17	**Incorrect absolute encoder adjustment**		Yes
	Cause	SSI navigator: The value of the absolute encoder adjustment is not in the encoder range: (increments per encoder revolution x number of revolutions - 1).	
18	**Incorrect type of reference point approach**		Yes
	Cause	You have specified a value outside of the permissible value quantities of 0, 1, 2 and 3.	
19	**Incorrect counting direction**		Yes
	Cause	You have specified a value outside of the permissible value quantities of 0 and 1.	

No.	Meaning		Diagnostic interrupt
20	It is not possible to monitor the hardware		Yes
	Cause	You have set frame error monitoring to "FALSE" in the parameter DB. The monitoring of missing pulses is not possible with encoders being used. Deactivate the MON_PULSE parameter.	
21	Incorrect software limit switch start		Yes
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or 1,000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any existing zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	Incorrect software limit switch end		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or 1,000 m to +1000 m, depending on the resolution) or is less than the software limit switch start. Linear axis: The software limit switch end (including any existing zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
23	Incorrect maximum velocity		Yes
	Cause	The data that is not listed in the parameter DB must be 0.	
24	Incorrect target range		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m depending on resolution. Rotary axis: Range greater than the end of rotary axis.	
25	Incorrect monitoring time		Yes
	Cause	The value for the monitoring time lies outside the permissible range from 0 to 100 000 ms.	
26	Incorrect standstill range		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m, depending on the resolution. Rotary axis: Range greater than the end of rotary axis.	
127	Incorrect standstill velocity		Yes
	Cause	The value for the standstill velocity lies outside the permissible range from 0 to 100 000 µm/min.	
128	Incorrect control mode		Yes
	Cause	You have specified a value for the control mode that lies outside the permissible range of 1 to 4.	
129	Incorrect start velocity for the reference point approach		Yes
	Cause	You have entered neither 0 nor 1 as a start velocity.	
130	Incorrect changeover difference in the direction +		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m (depending on resolution). Rotary axis: Range greater than the end of rotary axis and smaller than ½ of the target range.	
131	Incorrect changeover difference in the - direction		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m (depending on the resolution). Rotary axis: Range greater than the end of rotary axis and smaller than ½ of the target range.	

No.	Meaning		Diagnostic interrupt
132	**Incorrect switch-off difference in the direction +**		Yes
	Cause	The switch-off difference is greater than the switch-over difference plus, less than ½ of the target range, or is outside the permissible range between 0 to 100 m or 1000 m (depending on the resolution)	
133	**Incorrect switch-off difference in the - direction**		Yes
	Cause	The switch-off difference is greater than the switch-over difference minus, less than ½ of the target range, or is outside the permissible range between 0 to 100 m or 1000 m (depending on the resolution)	
200	**Incorrect resolution**		Yes
	Cause	You have specified a resolution < 0.1 µm/pulse or >1000 µm/pulse. You have specified a distance per encoder revolution and a number of pulses per encoder revolution that results in a resolution of < 0.1 or > 1000.	
201	**Encoder does not fit the operating range / rotary axis range**		Yes
	Cause	SSI encoders and rotary axes: The encoder does not precisely cover the rotary axis range. Linear axis: The encoder does not cover at least the operating range (including software limit switches).	

Incremental dimension table error (S7-300, S7-400)

Error class 6: Incremental dimension table error

The incremental dimension table errors do not result in an error response.

The following table shows the possible incremental dimension errors and their respective causes:

No.	Meaning		Diagnostic interrupt
6	**Specified increment in the increment table too great**		No
	Cause	The value lies outside of ±100 m or ±1000 m. The increment/target must not be greater than the traversing range. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	

Alarms (S7-300, S7-400)

Error class 15: Alarms

Alarms do not result in an error response.

The following table shows the possible alarms and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Start of parameter assignment**		No
	Cause	The module has detected a parameter assignment via a system data block.	
2	**End of parameter assignment**		No
	Cause	The module has processed the parameter assignment via a system data block without errors.	
11	**Distance to change-over point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	
12	**Distance to return point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	
14	**Distance to cut-off point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	
15	**Distance to start of target range too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	

Diagnostic errors (S7-300, S7-400)

Error class 128: Diagnostic errors

The following table shows the possible diagnostic errors and their respective causes, effects and remedies.

No.	Meaning		Diagnostic interrupt
4	**Missing external auxiliary supply**		Yes
	Cause	• External 24 V auxiliary voltage is not connected or has failed • The fuse on the module is defective • Undervoltage • Ground wire break • Short circuit (e.g. at the encoder connected)	
	Effect	• The positioning is cancelled on all channels • Output switch-off • Synchronization is cancelled for incremental encoders when the auxiliary voltage for the encoder supply is missing • Parameters have not been assigned to the positioning module • Start enable is deleted	
	Remedy	Make sure that the 24 V connection is correct (if the 24 V connection is correct, then the module is defective)	

No.	Meaning		Diagnostic interrupt
5	**Front connector is missing (FM 451)**		Yes
	Cause	Front connector of the positioning module is not inserted	
	Effect	• No external 24 V auxiliary supply • Module is not ready	
	Correction	Plug the front connector into the positioning module	
51	**Time monitoring activated (watchdog)**		Yes
	Cause	• Strong interference on the positioning module • Error in the positioning module	
	Effect	• Module is reset • Turn off all outputs • If, after resetting the module, no module fault is detected, the module is ready for operation again • The module reports the expired watchdog with "incoming" and "outgoing"	
	Correction	• Remedying interferences • Contact the relevant sales department which will require details of the exact circumstances leading to the error • Replace the positioning module	
144	**Encoder wire break**		Yes
	Cause	• The encoder cable has sheared off or is not plugged in • Encoders without cross signals • Incorrect pin assignment • Cable length too long • Encoder signals short-circuited • Encoder signal edge fault • Maximum input frequency of the encoder input has been exceeded • Sensor voltage supply failure	
	Effect	• The positioning is canceled • Output switch-off • Delete the synchronization with incremental encoders • Start enable is deleted	
	Correction	• Check encoder cable • Comply with encoder specification • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog • Comply with the technical specifications of the module	

No.	Meaning		Diagnostic interrupt
145	**Absolute encoder frame error**		Yes
	Cause	The frame traffic between the positioning module and the absolute encoder (SSI) is incorrect or interrupted: • Encoder cable cut or not plugged in • Incorrect encoder type • The encoder is incorrectly set (programmable encoder) • The message length has been incorrectly specified • The encoder is supplying incorrect values (encoder is faulty) • Interference on measuring system cable • Chosen transmission rate is too high • Monoflop time of the encoder is greater than 64 µs	
	Effect	• The positioning is canceled • Output switch-off • Start enable is deleted	
	Correction	• Check encoder cable • Check encoder • Check the frame traffic between the encoder and positioning module	
146	**Missing pulses of incremental encoder**		Yes
	Cause	• Encoder monitoring has established a missing pulse • Incorrect entry of the number of increments per encoder revolution • Encoder faulty: does not return the specified number of pulses • incorrect or no zero mark • Crosstalk on the encoder cable	
	Effect	• The positioning is canceled • Output switch-off • Start enable is deleted	
	Correction	• Enter the correct number of increments per encoder revolution (programming dialog) • Check the encoder and encoder cable • Comply with the shielding and grounding directives • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog	

6.2.6 Using FM 451 (S7-300, S7-400)

6.2.6.1 Terminal assignment of FM 451 front connector (S7-300, S7-400)

Description of the encoder interface (S7-300, S7-400)

Location of the sub D sockets

The figure shows the position and designation of the connectors on the module. You can connect an incremental encoder or absolute encoder (SSI) to the sub D sockets.

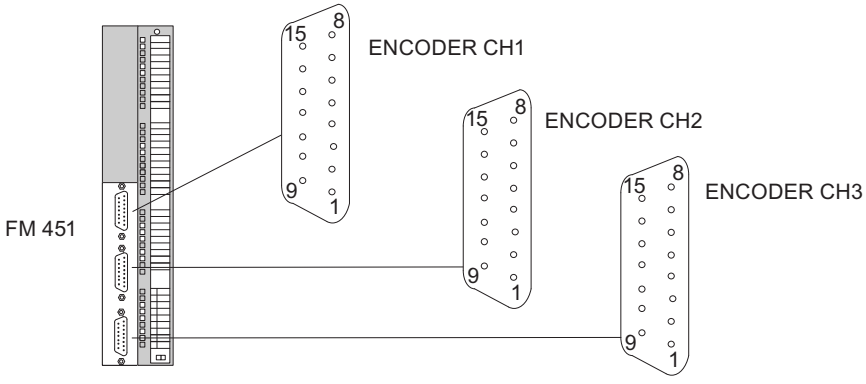

CH = Channel

Assignment of the encoder interfaces (CH1, CH2, and CH3 encoders)

Pin	Name	Incremental encoder (24 V)	Incremental encoder (5 V)	Absolute encoder
1	A*	Encoder signal A	---	---
2	CLS	---	---	SSI shift clock
3	/CLS	---	---	SSI shift clock inverse
4	B*	Encoder signal B	---	---
5	24 VDC	Sensor supply	Sensor supply	Sensor supply
6	5.2 VDC	---	Sensor supply	Sensor supply
7	M	Chassis ground	Chassis ground	Chassis ground
8	N*	Zero mark signal	---	---
9	RE	Current sourcing/current sinking[1)]	---	---
10	N	---	Zero mark signal	---
11	/N	---	Zero mark signal, inverse	---
12	/B	---	Encoder signal B, inverse	---
13	B	---	Encoder signal B	---
14	/A / /DAT	---	Encoder signal A, inverse	SSI data inverse

Pin	Name	Incremental encoder (24 V)	Incremental encoder (5 V)	Absolute encoder
15	A / DAT	---	Encoder signal A	SSI data
[1]**Note:** If you are connecting an incremental encoder from a third party in series-mode switching (current sourcing/current sinking), the following must be observed: • Sourcing output: Connect RE (9) with ground (7). • Sinking output: Connect RE (9) with +24 V (5).				

Terminal assignment of the front connector (S7-300, S7-400)

Front connector

Connect the following to the 48-pin front connector:

- the digital inputs and outputs assigned to the channels
- the supply voltage of the encoders and digital outputs

Terminal assignment of front connector (X1)

The table below shows the terminal assignment of the front connector of FM 451.

Wiring	Name	Function	Incremental encoder	Absolute encoder
1		Assigned; contains cable jumpers for detection of inserted front connector.		
2				
3	1L+	24 V DC auxiliary voltage for the encoder supply		
4 to 7		Not used		
8	1I0	Channel 1: Digital input 0	Reference point switch	Not used
9	1I1	Channel 1: Digital input 1	Reversing switch	Not used
10	1I2	Channel 1: Digital input 2	Enable input	
11	1I3	Channel 1: Digital input 3	Input for "Set process value on-the-fly"/ length measurement/edge detection/ external event for increment number 252	
12		Not used		
13	2L+	24 V DC auxiliary voltage for the load current supply		
14	2L+	Both terminals are internally connected on the module.		
15	2I0	Channel 2: Digital input 0	Reference point switch	Not used
16	2I1	Channel 2: Digital input 1	Reversing switch	Not used
17	2I2	Channel 2: Digital input 2	Enable input	
18	2I3	Channel 2: Digital input 3	Input for "Set process value on-the-fly"/ length measurement/edge detection/ external event for increment number 252	
19	3I0	Channel 3: Digital input 0	Reference point switch	Not used
20	3I1	Channel 3: Digital input 1	Reversing switch	Not used
21	3I2	Channel 3: Digital input 2	Enable input	

Wiring	Name	Function	Incremental encoder	Absolute encoder
22	3I3	Channel 3: Digital input 3	Input for "Set process value on-the-fly"/ length measurement/edge detection/ external event for increment number 252	
23 to 24		Not used		
25	3L+	24 V DC auxiliary voltage for the load current supply		
26	3L+	Both terminals are internally connected on the module.		
27	1Q0	Channel 1: Digital output 0		
28	1Q1	Channel 1: Digital output 1		
29	1Q2	Channel 1: Digital output 2		
30	1Q3	Channel 1: Digital output 3		
31	2Q0	Channel 2: Digital output 0		
32	2Q1	Channel 2: Digital output 1		
33	2Q2	Channel 2: Digital output 2		
34	2Q3	Channel 2: Digital output 3		
35 to 36		Not used		
37	4L+	24 V DC auxiliary voltage for the load current supply		
38	4L+	Both terminals are internally connected on the module.		
39	3Q0	Channel 3: Digital output 0		
40	3Q1	Channel 3: Digital output 1		
41	3Q2	Channel 3: Digital output 2		
42	3Q3	Channel 3: Digital output 3		
43 to 47		Not used		
48	M	Ground for auxiliary voltages 1L+, 2L+, 3L+, and 4L+		

Additional information about design and wiring can be found in the Operating Instructions (http://support.automation.siemens.com/WW/view/en/1118070) of the positioning module FM 451.

Auxiliary voltage for the encoder supply (1L+, M)

Here, you connect a 24 V DC auxiliary voltage for the encoder supply. The reference potential of this supply (M) is connected to the ground of the load current supply (M) in the FM 451.

The 24 V DC auxiliary voltage for the encoder supply is monitored for undervoltage and wire break.

This supplies 24 V DC and 5.2 V DC at the encoder interface (sub D sockets encoder CH1, CH2 and CH3) to the various types of encoders.

CAUTION

Ensure the correct polarity of the 24 V DC auxiliary voltage for the encoder supply (1L+, M).

If you reverse the polarity of the 24 V DC auxiliary voltage for the encoder supply, the module will be damaged and will have to be replaced.

Auxiliary voltage for the load current supply (2L+, 3L+, 4L+, M)

You must connect the 24 V DC auxiliary voltages for the load current supply of the digital outputs to the 2L+, 3L+, 4L+, and M terminals.

⚠ CAUTION
Verify the correct polarity of the 24 V DC auxiliary voltage for the load current supply (2L+, 3L+, 4L+, M). If you reverse the polarity of the 24 V DC auxiliary voltage for the load current supply, the module will be damaged and will have to be replaced.

Wiring information for 24 V DC

When wiring, note that the terminals 1L+ to 4L+ must be interconnected so that the module will operate error-free.

Start by wiring the 24 V DC to terminal 38 and connect the auxiliary voltage of terminal 37 to terminal 26. Use the same procedure for the other terminals.

Alternatively, you can also connect separate voltage supplies to the 1L+ (encoder), 2L+ (channel 1), 3L+ (channel 2), and 4L+ (channel 3) connections. Make sure that all voltage supplies have a common ground potential. In this case, synchronization of the axes will be retained following a power failure in the auxiliary voltages for the load current supply.

Load current supplies

The DC load current supply must meet the following requirements:

Only a safe, isolated extra-low voltage of ≤ 60 V DC may be used as the load current supply. Safe isolation can be implemented, for example, by adhering to the specifications

- VDE 0100 Part 410 / HD 384-4-41 / IEC 364-4-41
  (as protective extra low voltage) and/or
- VDE 0805 / EN 60950 / IEC 950
  (as safety low voltage SELV) or
- VDE 0106 Part 101.

12 digital inputs (1I0 to 3I3)

The FM 451 has 4 digital inputs per channel.

You can connect bounce-free switches (24 V current sourcing) or non-contact sensors (2- or 3-wire proximity switches) to the 12 digital inputs.

The digital inputs are not monitored for short circuits or wire break and have a non-isolated connection to the encoder supply ground.

A separate LED indicates the state of each input.

12 digital outputs (1Q0 to 3Q3)

The FM 451 has 4 digital outputs per channel.

The power unit is controlled via the digital outputs. The function of the digital outputs depends on the control mode (Page 2534).

The digital outputs are not monitored for short circuits or wire break and have a non-isolated connection to the encoder supply ground.

The associated LED indicates the state of each output.

6.2.6.2 Functions of the FM 451 (S7-300, S7-400)

Overview of functions (S7-300, S7-400)

You define module functionality by setting the mode, job or function switch. The table below gives an overview of the functionality of the module.

Functionality	Description
Writing and reading machine data (Page 2490)	You can use the machine data to adapt the positioning module to the axis and the encoder.
Writing and reading increment tables (Page 2492)	The incremental dimensions define the targets which are controlled by the positioning module with the "Relative / absolute incremental approach" mode.
End of a positioning operation (Page 2494)	The end of a positioning operation is achieved in three different ways: • Target approach • Shutdown • Cancel
"Inching" mode (Page 2501)	In "Inching" mode you can move the drive in one direction at the press of a button.
"Reference point approach" mode (Page 2504)	The "Reference point approach" mode can be used to synchronize the axis based on a recurring external event.
"Incremental approach" mode (Page 2511)	With the "Incremental approach" mode, the positioning module can move the drive • toward absolute targets, • relatively by an incremental distance in a specified direction.
The "Set actual value" / Set actual value on-the-fly / "Undo Set Actual Value" job (Page 2517)	With the "Set actual value" or "Set actual value on-the-fly", you assign a new coordinate to the actual encoder value. With the "Undo set value" job, you reset all the operating range shifts that were generated by means of the "Set actual value" job.
"Zero offset" job (Page 2519)	The "zero offset" job lets you shift the zero point in the coordinate system by a defined value.
"Set reference point" job (Page 2522)	With the "Set reference point" job you synchronize the axis.
"Length measurement" / "Edge detection" function switches (Page 2523)	"Length measurement" and "Edge detection" let you determine the length of a workpiece.

Functionality	Description
"Loop traverse" function switch (Page 2525)	You use "Loop traverse" to specify the direction in which a target is approached with a frictional connection.
"Enable input" function switch (Page 2528)	The enable input is an external input with which positioning can be enabled by an external event.
"Read position data" job (Page 2529)	With the "read position data" job you can read the incremental dimension, residual distance and velocity at the current time.
"Read encoder data" job (Page 2529)	With the "read encoder data" job you can read the current data of the encoder as well as the value for the absolute encoder adjustment.
Checkback signals for positioning (Page 2530)	The "Feedback signals for positioning" inform you of the current status of the positioning.
Checkback signals for diagnostics (Page 2531)	The "Feedback signals for diagnostics" informs you of occurring diagnostic events.

Writing and reading machine data (S7-300, S7-400)

Writing, enabling and reading machine data

You can use the machine data to adapt the positioning module to the axis and the encoder.

You can change and read parameters during operation using the user program.

All parameters are stored in the parameter DB:

- Machine data is located in the parameter DB at the addresses 4.0 to 116.0.

Enter the parameter DB number in the corresponding channel DB.

You can enter the parameters with the DB Editor or, simply and easily, enter them in the "Drive", "Axis", "Encoder" and "Increments" dialogs before writing them to the parameter DB with the "Export" function.

You can use the "Import" function to import parameters from an existing parameter DB to the dialogs.

Initial parameter assignment

If the channel does not yet contain any machine data, proceed as follows for initial parameter assignment without the parameter assignment dialogs:

1. Enter the new values in the parameter DB and then save them.
2. Download the parameter DB to the CPU.
3. Set the following trigger bit in the channel DB for the job:
   – Write machine data (MDWR_EN = 1).
4. Call the ABS_CTRL or ABS_CTRL_451 instruction in the cyclic user program.

Modifying machine data

Proceed as follows to change machine data using the user program:

1. Enter the new values in the parameter DB.
2. Set the trigger bits in the channel DB for the jobs:
   – Write machine data (MDWR_EN = 1)
   – Activate machine data (MD_EN = 1)
3. Call the ABS_CTRL or ABS_CTRL_451 instruction in the cyclic user program.

If you set the trigger bits for these jobs all at once, the ABS_CTRL instruction (or ABS_CTRL_451) makes sure the jobs are processed in the correct order.

Otherwise change the machine data always in the following sequence:

- Write machine data
- Enable machine data

Reading machine data

In order to read the current machine data from a channel, proceed as follows:

1. Set the following trigger bit in the channel DB:
   – Reading machine data (MDRD_EN = 1)
2. Call the ABS_CTRL or ABS_CTRL_451 instruction in the cyclic user program.

The current machine data is then stored in the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.0	MDWR_EN	BOOL	FALSE	1 = Write machine data
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
36.5	MDRD_EN	BOOL	FALSE	1 = read machine data

Note

If synchronization-relevant parameters have been changed, the module will carry out the following actions for the respective channel when the machine data is enabled:

- Synchronization will be deleted.
- The function switches and zero offset will be reset.
- All previous machine data and incremental dimension tables will become invalid.

Parameters relevant to synchronization:

- Axis type
- End of rotary axis
- Signal type
- Distance per encoder revolution
- Increments per encoder revolution
- Number of revolutions
- Reference point coordinate
- Absolute encoder adjustment
- Type of reference point approach
- Count direction

Writing and reading incremental dimension tables (S7-300, S7-400)

Writing and reading incremental dimension tables

The incremental dimensions define the targets which are controlled by the positioning module with the "Relative / absolute incremental approach" mode.

You can change and read parameters during operation using the user program.

All parameters are stored in the parameter DB:

- Increment tables are located in the parameter DB at the addresses 120.0 to 516.0.

The number of the parameter DB must be entered in the respective associated channel DB.

You can enter the parameters with the DB Editor or also easily using the "Incremental dimensions" dialog and then write in the parameter DB with the "Export" function.

You can use the function "Import" to import parameters that are in an already available parameter DB into the dialogs.

First parameter assignment

If the channel still contains no increment tables, proceed as with a first parameter assignment without the parameter assignment dialog in the following way:

1. Enter the new values in the parameter DB and then save these.
2. Download the parameter DB to the CPU.

3. Set the trigger bits in the channel DB for the jobs:
   – Writing incremental dimension table 1 (TRGL1WR_EN = 1) and / or writing incremental dimension table 2 (TRGL2WR_EN = 1).
4. Call the ABS_CTRL or ABS_CTRL_451 instruction in the cyclic user program.

Changing increment tables

Proceed as follows to change increment tables using the user program:

1. Enter the new values in the parameter DB.
2. Set the trigger bits in the channel DB for the jobs:
   – Writing incremental dimension table 1 (TRGL1WR_EN = 1) and / or writing incremental dimension table 2 (TRGL2WR_EN = 1).
3. Call the ABS_CTRL or ABS_CTRL_451 instruction in the cyclic user program.

Reading increment tables

In order to read the increment tables from a channel, proceed as follows:

1. Set the trigger bits in the channel DB for the jobs:
   – Reading incremental dimension table 1 (TRGL1RD_EN = 1) and / or writing incremental dimension table 2 (TRGL2RD_EN = 1).
2. Call the ABS_CTRL or ABS_CTRL_451 instruction in the cyclic user program.

The increment tables are then stored in the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.4	TRGL1WR_EN	BOOL	FALSE	1 = write increment table 1 (1 ... 50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = write increment table 2 (51 ... 100)
36.6	TRGL1RD_EN	BOOL	FALSE	1 = read increment table 1 (1 ... 50)
36.7	TRGL2RD_EN	BOOL	FALSE	1 = read increment table 2 (51 ... 100)

Note

If synchronization-relevant parameters have been changed, when the machine data is enabled the module will carry out the following actions for the respective channel:

- Synchronization will be deleted.
- The function switches and zero offset will be reset.
- All previous machine data and incremental dimension tables will become invalid.

Parameters relevant to synchronization:

- Axis type
- End of rotary axis
- Signal type
- Distance per encoder revolution
- Increments per encoder revolution
- Number of revolutions
- Reference point coordinate
- Absolute encoder adjustment
- Type of reference point approach
- Count direction

End of a positioning operation (S7-300, S7-400)

Definition

The end of a positioning operation is displayed by the checkback signal WORKING = 0. It can be reached in three different ways:

- Target approach
- End
- Cancel

Monitoring

The following monitoring functions are active towards the end of a positioning operation:

- Monitoring time
  The monitoring time is retriggered at the switch-off point for the last time and then loses its validity with the end of positioning.
  The end of the positioning operation must be reached within this time. If it is not, the outputs will be switched off and the operating error "Error during target approach" (error number 5) issued.
- Monitoring the target range
  The FM x51 sets a symmetrical range around each target, thereby defining the positioning precision of your application. The axis must come to a standstill within this range during a target approach. A value 0 switches off the tolerance during the target approach.
- Monitoring the standstill velocity
  The standstill velocity is used determine whether the drive comes to a standstill within the target range. The FM x51 checks whether the actual drive speed is less than the stationary drive speed when the switch-off point has been reached.
  It must be less than the standstill velocity within the target range. The FM x51 will otherwise report the operating error "Target range overrun" (error number 10).
  Falling short of the standstill velocity is monitored only once per target approach.
  Note that the velocity can briefly fall below the standstill velocity for the velocity determination of the module if the axis moves at a very slow positioning velocity (less than 2 pulses per 8 ms).
- Monitoring the stationary range
  Once a positioning operation ends, the drive is monitored to see whether it remains at an approached target position or drifts off.
  The standstill range is monitored
  - after the FM x51 has reported the checkback signal "PR"
  - if the monitoring time is exceeded
  - if the velocity falls below the standstill velocity.

  If the standstill range is exited without a valid traversing job, the FM x51 reports the "Stationary range exited" error (error number 6).

Target approach

The target approach in the "Absolute incremental approach" and "Incremental approach" modes starts when the switch-off point has been reached. As of this point the drive is switched off and the FM x51 adopts the monitoring functions.

Depending on the monitoring you have parameterized, there are a number of different scenarios in which the checkback signal "PR (POS_RCD)" is generated. The positioning operation is cancelled if no "PR (POS_RCD)" checkback signal is generated.

Scenario 1: You have assigned parameters as follows:

- Target range (TRG_RANGE) > 0
- Standstill velocity (ZSPEED_L) > 0
- Monitoring time (MON_TIME) > 0

PR is generated when the velocity falls below the standstill velocity and the target range has been reached. It does not matter which condition is fulfilled first.

PR is not generated if the actual value has not reached the target range within the monitoring time or the target range has been overrun without the velocity falling below the standstill velocity.

The figure below shows the target approach in incremental approach mode with a target range > 0 and standstill velocity > 0.

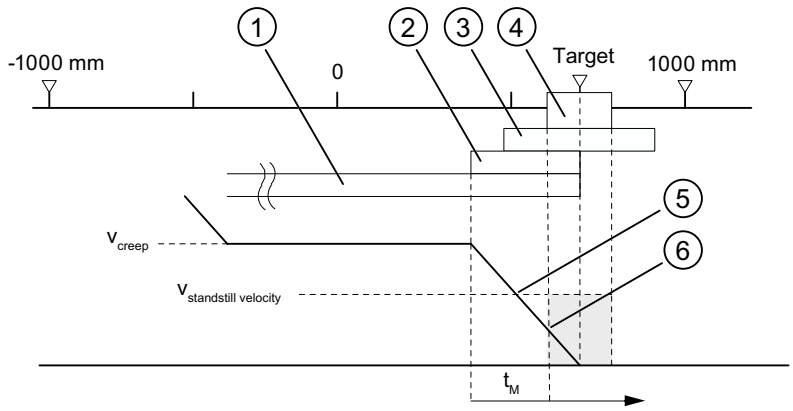

①	Changeover difference plus
②	Switch-off difference plus
③	Stationary range
④	Target range
⑤	Standstill velocity reached
⑥	Target range with V_{still} reached: PR is set
t_M	Monitoring time

Scenario 2: You have assigned parameters as follows:

- Target range (TRG_RANGE) > 0
- Standstill velocity (ZSPEED_L) = 0
- Monitoring time (MON_TIME) > 0

PR is generated when the target range has been reached.

PR is not generated if the actual value does not reach the target range within the monitoring time.

The figure below shows the target approach in incremental approach mode with a target range > 0 and standstill velocity = 0.

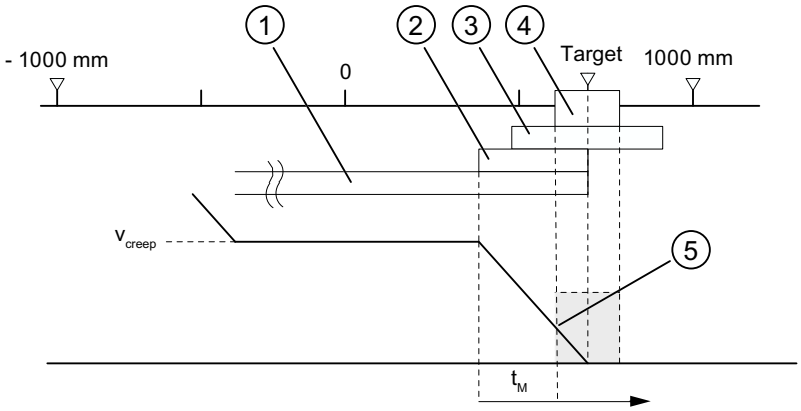

① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Target range
⑤ Target range with V_{still} reached: PR is set
t_M Monitoring time

Scenario 3: You have assigned parameters as follows:

- Target range (TRG_RANGE) = 0
- Standstill velocity (ZSPEED_L) > 0
- Monitoring time (MON_TIME) > 0

PR is generated if the velocity falls below the standstill velocity and the target is then reached.

PR is not generated if the actual value has not reached the target within the monitoring time or the target range has been overrun without the velocity falling below the standstill velocity.

The figure below shows the target approach in incremental approach mode with a target range = 0 and standstill velocity > 0.

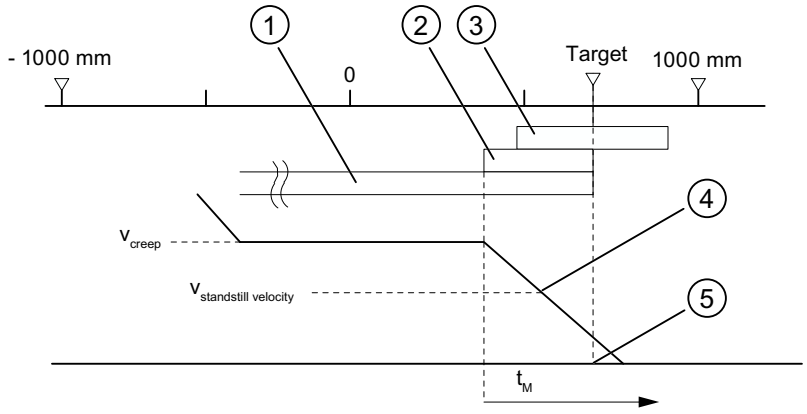

① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Standstill velocity reached
⑤ Target reached: PR is set
t_M Monitoring time

Scenario 4: You have assigned parameters as follows:

- Target range (TRG_RANGE) = 0
- Standstill velocity (ZSPEED_L) = 0
- Monitoring time (MON_TIME) > 0

PR is generated when the target has been reached.

PR is not generated if the actual value does not reach the target within the monitoring time.

The figure below shows the target approach in incremental approach mode with a target range = 0 and standstill velocity = 0.

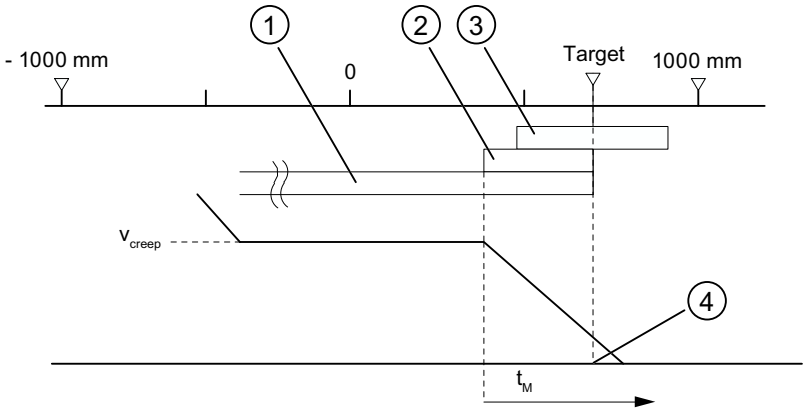

①	Changeover difference plus
②	Switch-off difference plus
③	Stationary range
④	Target reached: PR is set
t_M	Monitoring time

Scenario 5: You have assigned parameters as follows:

- Target range (TRG_RANGE) ≥ 0
- Standstill velocity (ZSPEED_L) ≥ 0
- Monitoring time (MON_TIME) = 0

If, in this case, the drive comes to a standstill before the target range, the end of the positioning operation will not be detected. PR will not be generated and the checkback signal WORKING will remain set. You can only cancel the positioning operation only by deleting the drive enable (DRV_EN = 0).

Shutdown without specified target

Shutdown means: The positioning operation is purposefully ended whilst complying with the differences between rapid traverse and creep speed.

The positioning operation is ended if

- The FM x51 receives a STOP signal (STOP = 1)
- The operating modes "Jog" and "Reference point approach" are ended.
- Operating errors or the operating error "Target overrun" (error number 9) occur.

The checkback signal "PR (POS_RCD)" is not set. The order of events are analogous to those of the target approach.

The figure below shows the end of a positioning operation.

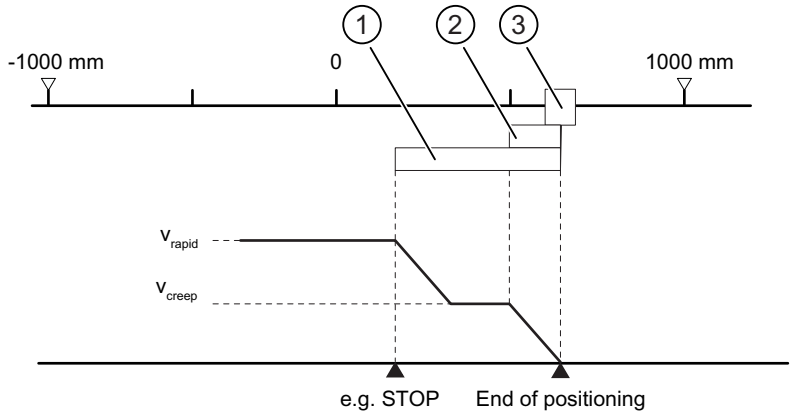

① Changeover difference plus
② Switch-off difference plus
③ Target range

Cancel

Shutdown means: The positioning process is ended immediately after standstill without the use of the changeover and switch-off differences between rapid traverse and creep speed. In addition, all the relevant outputs of the respective control mode are immediately switched off and also:

- Increment = actual value
- Distance-to-go = zero

The positioning operation is cancelled if

- The drive enable signal is deleted (DRV_EN = 0).
- The CPU goes into STOP mode.
- Diagnostic errors or all operating errors apart from the "Target overrun" operating error (error number 9) occur.

The checkback signal "PR (POS_RCD)" is not be set in "Incremental approach" mode.

If the standstill velocity is assigned, the standstill monitoring will become active as soon as the velocity falls below the standstill velocity. If the standstill velocity is not assigned, the standstill monitoring will be active when the outputs have been switched off.

The figure below shows the cancellation of a positioning operation.

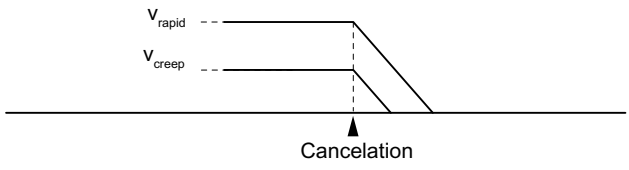

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	TRG_RANGE	DINT	L#1000	Target range
80.0	MON_TIME	DINT	L#2000	Monitoring time
84.0	ZSPEED_R	DINT	L#1000	Stationary range
88.0	ZSPEED_L	DINT	L#30000	Standstill velocity

Checkback signals in the channel DB

Address	Name	Type	Initial value	Comment
23.1	WORKING	BOOL	FALSE	1 = Positioning operation is running
25.7	POS_RCD	BOOL	FALSE	1 = position reached

"Jog" mode (S7-300, S7-400)

Definition

In "Jog" mode, you can move the drive in one direction by pushing a button. You must install a button for both directions (plus and minus) respectively. You can use the "Jog" mode both for a synchronized as well as for an unsynchronized axis.

In the case of an unsynchronized axis, jogging is a positioning in the specified direction.

For a synchronized axis (linear axis), jogging involves a positioning to the software limit switches.

Requirement

The axis parameters must be assigned.

Sequence of events for "Jog" mode

1. Set the control signal for the "Jogging" mode (MODE_IN = 1).
2. Set the control signal for the drive enable (DRV_EN = 1).
3. Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.
4. Enter the start velocity.
   - Rapid traverse (MODE_TYPE = 1)
   - Creep speed (MODE_TYPE = 0)
5. Set the control signal for traversing direction plus or minus (DIR_P = 1 or DIR_M = 1).
6. Call the ABS_CTRL or ABS_CTRL_451 instruction.

The following diagram shows an example for the "Jogging" mode.

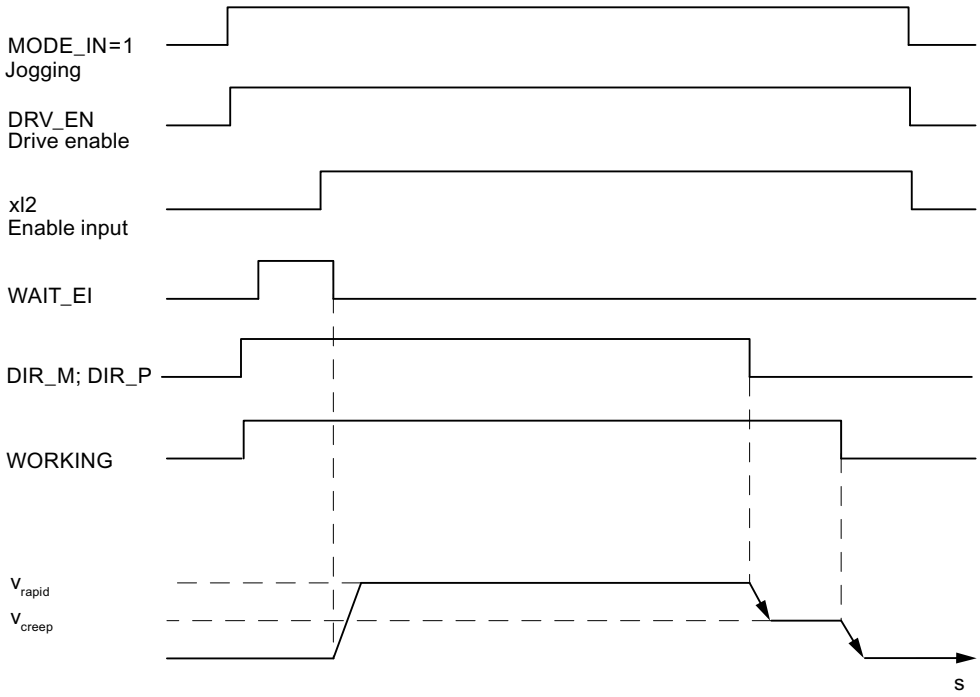

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	1 = jog
17.0	MODE_TYPE	BYTE	B#16#0	1 = Rapid traverse 0 = Creep speed
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = Positioning process is running
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Shutting down jogging

The "Jog" mode is shut down if

- You open the button used for jogging (DIR_M = 0 or DIR_P = 0),
- The FM 451 receives a STOP signal (STOP = 1)
- the process value with a synchronized linear axis reaches the limit of the operating range. The axis can only continue moving in the opposite direction.

After traversing is shut down, it can continue in any direction.

Canceling jogging

"Jog" mode is cancelled if

- The drive enable signal is deleted (DRV_EN = 0)
- A traversing range limit has been overrun for a linear axis

Monitoring functions

In the "Jogging" mode, the following monitoring functions are not active at the end of the positioning:

- Monitoring of standstill range
- Monitoring of target range
- Monitoring of standstill velocity

Jogging does not result in proper positioning for which the "Position reached" signal is set at the end.

Operating range limit for a linear axis

The limits for the "Jogging" mode differ between a synchronized and an unsynchronized axis.

The following table shows "Jogging" with synchronized and non-synchronized axes.

Axis is unsynchronized	Axis is synchronized
If, when jogging, the traversing range limit is overrun, • the process value display is no longer valid. • the positioning process is aborted. 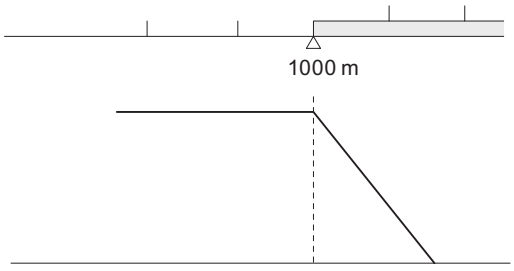 The process value indicated is no longer valid	Jogging involves a positioning to targets that are removed from the software limit switches by a distance equal to the whole target range. The operating range limits are determined from • **SEE - ½ target range** for the end of the linear axis in the plus direction • **SEE + ½ target range** for the end of the linear axis in the plus direction If you do not open the button beforehand, the FM 451 shuts down at a target point that is positioned half the target range away from the respective software limit switch. All ranges that are necessary for proper shutdown are set by the FM 451 relative to this target point. 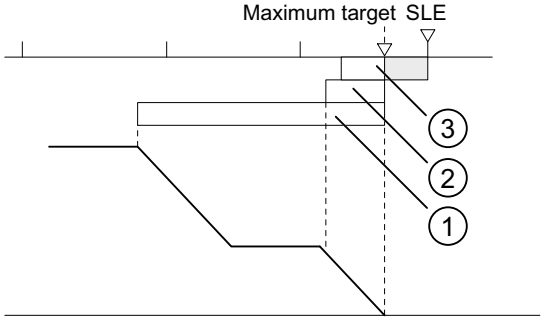 = Part of operating range in which a target position cannot be set ① = Changeover difference plus ② = Switch-off difference plus ③ = ½ target range

"Reference point approach" mode (S7-300, S7-400)

Definition

The "Reference point approach" mode can be used to synchronize the axis based on a recurring external event.

Requirements

- An incremental encoder with zero mark.
- The axis parameters must be assigned:

Connections	Channel 1	Channel 2	Channel 3
Reference point switch	Digital input 1I0	Digital input 2I0	Digital input 3I0
	The reference point switch should be furnished such that the drive can safely brake from rapid traverse to creep speed in the range of the switch.		
Reversing switch	Digital input 1I1	Digital input 2I1	Digital input 3I1
	When assigning parameters, ensure that the start of the reference point approach is assigned in the direction of the reversing switch. Only in this way can you guarantee that the reference point switch will always be found.		
Enable input	Digital input 1I2	Digital input 2I2	Digital input 3I2

Sequence of events for "Reference point approach" mode

The keys for the positive and negative traversing direction must be installed for each channel.

1. Set the value for the reference point coordinate at the parameter DB REFPT.
2. Enter the type of "Reference point approach" in the parameter DB.
   The following options are available:

Start in direction ...	That leads to synchronization ...	
plus	the first zero mark after leaving the reference point switch in direction **plus**	REFPT_TYPE = 0
plus	the first zero mark after leaving the reference point switch in direction **minus**	REFPT_TYPE = 1
minus	the first zero mark after leaving the reference point switch in direction **plus**	REFPT_TYPE = 2
minus	the first zero mark after leaving the reference point switch in direction **minus**	REFPT_TYPE = 3

3. Enter the start velocity.
   - Rapid traverse (REFPT_SPD = 0)
   - Creep speed (REFPT_SPD = 1)
4. Write and enable the machine data.
5. Set the control signal for the "Reference point approach" mode (MODE_IN = 3).
6. Set the control signal for the drive enable (DRV_EN = 1).
7. Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.
8. Set the control signal for traversing direction plus or minus or the start signal (DIR_P = 1, DIR_M = 1 or START = 1)
9. Call the ABS_CTRL or ABS_CTRL_451 instruction.

The following table shows the start commands for a reference point approach.

Start command	Job	Comment
DIR_P	The drive starts in the direction with positive values. It therefore moves in the direction of the traversing range end.	If a negative direction is entered in the machine data, the FM 451 signals an operator error. No reference point approach is executed.
DIR_M	The drive starts in the direction with negative values. It therefore moves in the direction of the traversing range start.	If a positive direction is entered in the machine data, the FM 451 signals an operator error. No reference point approach is executed.
START	The drive starts in the direction that is entered in the machine data.	

Note

The following applies to the rotary axis: Reproducibility of the reference point is only ensured if the ratio of the **End of rotary axis** value and the **Distance per encoder revolution** value is an integer.

The following figure shows an example for the "Reference point approach" mode.

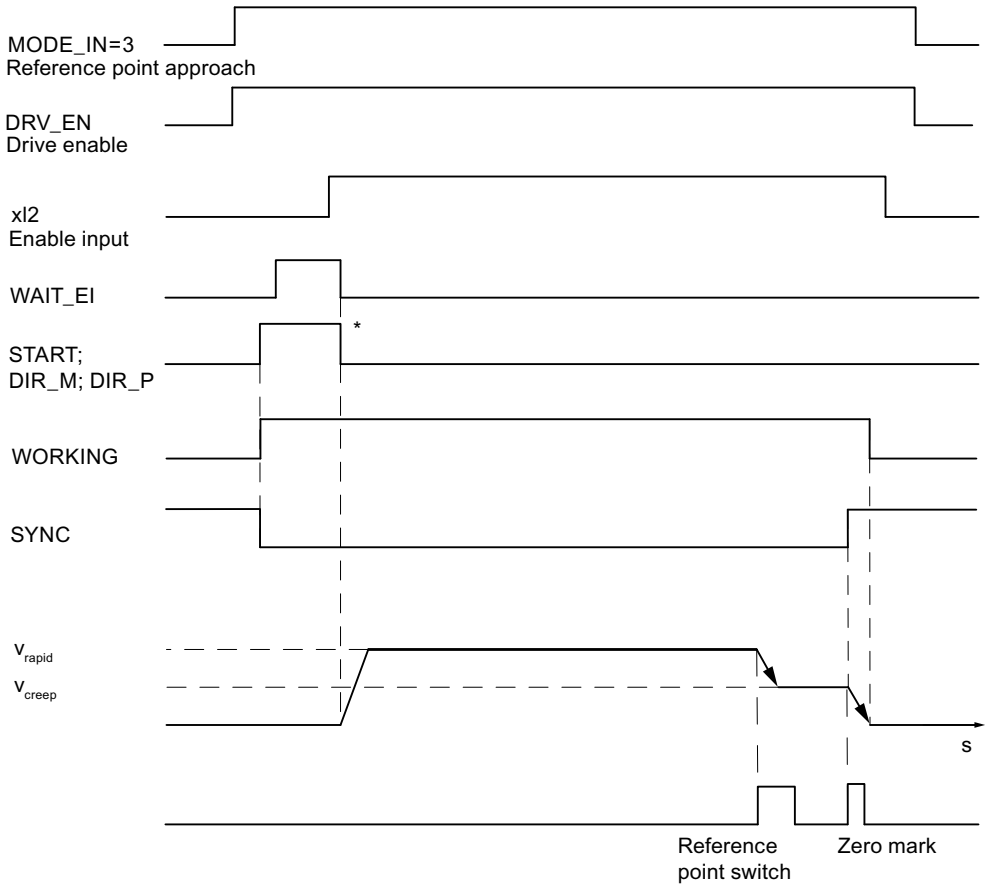

* The start signals are reset by the ABS_CTRL or ABS_CTRL_451 instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = Start positioning process
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	3 = reference point approach
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = Positioning process is running
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	Reference point coordinate
52.0	REFPT_TYPE	DINT	L#0	Type of reference point approach
99.0	REFPT_SPD	BOOL	TRUE	Start velocity for the reference point approach 0 = rapid traverse 1 = creep speed

Effects of the mode

- Synchronization is canceled when traversing starts.
- The actual position is set to the value of the reference point coordinate when the SYNC feedback signal is set.
- The operating range is established on the axis.
- The individual points within the operating range maintain their original value, but are located at different positions.

Canceling the reference point approach

The "Reference point approach" mode is cancelled if

- The drive enable signal is deleted (DRV_EN = 0)
- A traversing range limit has been overrun for a linear axis

Reference point approach in dependence on the start position

In the case of a reference point approach, you must distinguish between different cases that are dependent on

- the location of the drive at the time a reference point approach is started,
- the assigned start direction,
- the assigned location of the zero mark relative to the reference point switch.

The following table shows the cases for REFPT_TYPE = 0 and REFPT_TYPE = 1. The figures apply correspondingly to REFPT_TYPE = 2 and REFPT_TYPE = 3.

Conditions of the reference point approach	Sequence of events for the reference point approach
Example of reference point approach (REFPT_TYPE = 0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction.	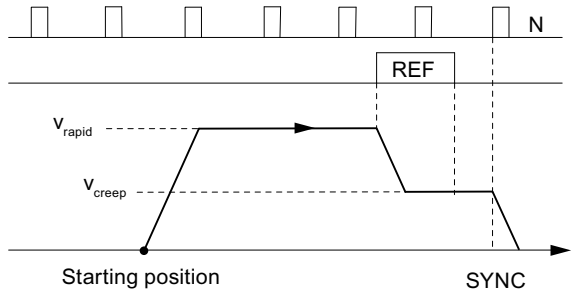
Example of reference point approach (REFPT_TYPE = 1): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the minus direction.	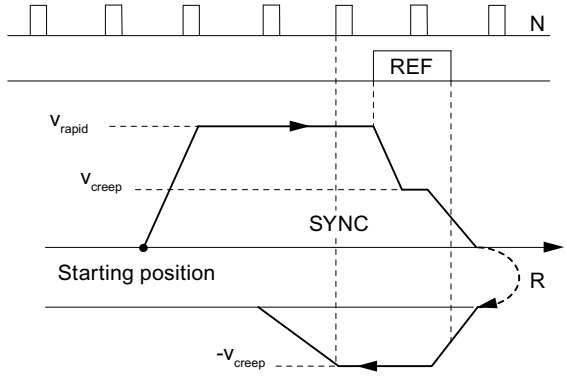
Example of reference point approach (REFPT_TYPE = 0): • The start direction must be assigned in the plus direction. • The location of the zero mark of the reference point switch is assigned in the plus direction. • The reversing switch is more positive than the reference point switch.	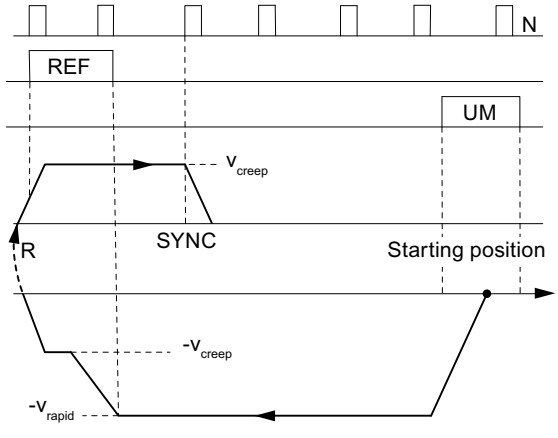

Conditions of the reference point approach	Sequence of events for the reference point approach
Example of reference point approach (REFPT_TYPE = 1): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the minus direction. • The start position of the reference point approach is on the reference point switch.	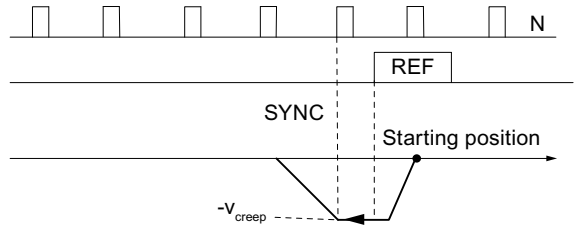
Example of reference point approach (REFPT_TYPE = 0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction. • The reversing switch is more positive than the reference point switch.	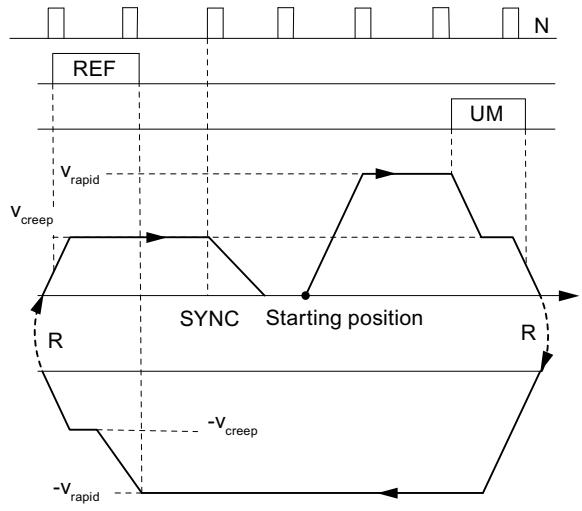
Example of reference point approach (REFPT_TYPE=0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction. • Start velocity = creep speed	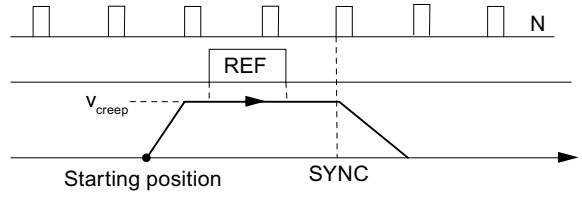
R = Direction reversal REF = Reference point switch UM = Reversing switch N = Zero mark of the encoder SYNC = Synchronization has been achieved	

"Incremental approach" mode (S7-300, S7-400)

Definition

With the "Incremental approach" mode, the FM 451 can move the drive.

- toward **absolute** targets,
- **relatively** by an incremental distance in a specified direction.

You specify the target or the relative distances as increment dimensions of the FM 451. You have the possibility of entering up to 100 incremental dimensions in a table that are valid both for the "Relative incremental approach" mode as well as for the "Absolute incremental approach" mode. Independently of the incremental dimension table, you can specify the distance using increment dimension 254 (Page 2561) and incremental dimension 255 (Page 2562)

The incremental dimension 252 (Page 2561) gives you the option of moving the drive initially in a specified direction without a target (continuous traversing). When an external event occurs (xI3), the programmed target is then positioned upon.

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.
- The increments must be available on the module.

Interpretation of the increments

The FM 451 interprets the specifications differently, depending on the incremental approach you select.

- Absolute incremental approach: The incremental dimensions are interpreted as absolute target position.
  With a rotary axis, the end of the axis may not be passed.
- Relative incremental approach: The incremental dimensions are interpreted as distance difference from the start position.
  With a rotary axis, the end of the axis may not be passed (multiple revolutions are permitted).

Note

Only positive increments are permissible for the "Relative incremental approach" mode.

The incremental dimensions result from the DIR_P or DIR_M direction specifications.

"Incremental approach" mode sequence with incremental dimension number 1 to 100

Step	Absolute incremental approach	Relative incremental approach
	Incremental dimension number 1 to 100	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Enter the incremental dimensions in the tables (TRGL1; TRGL2).	
3	Write the incremental dimension table (TRGL1/2WR_EN = 1).	
4	Set the control signal for the drive enable (DRV_EN = 1).	
5	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
6	Enter the incremental dimension number (MODE_TYPE = 1 … 100).	
7	Set the control signal: • Linear axis: – START; the direction is determined explicitly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
8	Call the ABS_CTRL or ABS_CTRL_451 instruction.	
Steps 2 and 3 are only required if no increments are given or the existing increments are to be changed.		

"Incremental approach" mode with incremental dimension number 252

Step	Absolute incremental approach	Relative incremental approach
	Incremental dimension number 252	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Set the control signal for the drive enable (DRV_EN = 1).	
3	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
4	Enter the incremental dimension number (MODE_TYPE = 252).	
5	Enter the start velocity. • Rapid traverse (SPEED252 = 1) • Creep speed (SPEED252 = 0)	
6	Enter the incremental dimension for the incremental dimension number 252 (TRG252_254).	
7	Set the trigger bit for writing the incremental dimension (TRG252_254_EN = 1).	

Step	Absolute incremental approach	Relative incremental approach
	Incremental dimension number 252	
8	Set the control signal: • Linear axis / rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction	
9	Call the ABS_CTRL or ABS_CTRL_451 instruction.	
10	When an external event occurs (digital input xI3), the positioning to the assigned target (increment) is performed.	
The actions listed in steps 6 and 7 can also only be performed during traversing. In this case, the FM 451 module controls a target up to the incremental dimension specification. This target is located a distance in front of the respective software limit switch equal to half the target range.		

Sequence of events for the "incremental approach" mode with increment number 254

Step	Absolute incremental approach	Relative incremental approach
	Increment number 254	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Set the control signal for the drive enable (DRV_EN = 1).	
3	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
4	Enter the incremental dimension number (MODE_TYPE = 254).	
5	Enter the incremental dimension for the incremental dimension number 254 (TRG252_254).	
6	Set the trigger bit for writing the incremental dimension (TRG252_254_EN = 1).	
7	Set the control signal: • Linear axis: – START; the direction is determined explicitly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
8	Call the ABS_CTRL or ABS_CTRL_451 instruction.	

Sequence of events for the "incremental approach" mode with increment number 255

Step	Absolute incremental approach	Relative incremental approach
	Increment number 255	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Set the control signal for the drive enable (DRV_EN = 1).	
3	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
4	Enter the incremental dimension number (MODE_TYPE = 255).	
5	Enter the incremental dimension for the incremental dimension number 255 (TRG255).	
6	Enter the value for the switch-over difference of the incremental dimension number 255 (CHGDIF255).	
7	Enter the value for the switch-over difference of the incremental dimension number 255 (CUTDIF255).	
8	Set the trigger bit for writing the incremental dimension, switch-off difference, and switch-over difference (TRG255_EN = 1).	
9	Set the control signal: • Linear axis: – START; the direction is determined explicitly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
10	Call the ABS_CTRL or ABS_CTRL_451 instruction.	

The following figure shows an example for the "Incremental approach" mode.

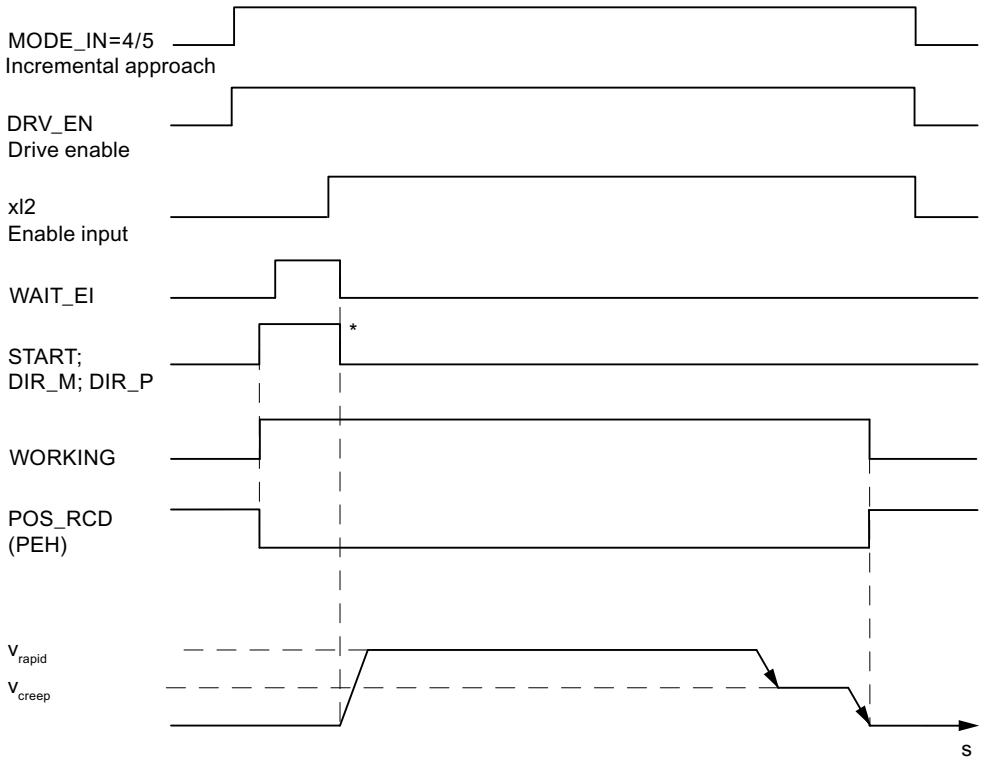

* The start signals are reset by the ABS_CTRL or ABS_CTRL_451 instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = start positioning
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.6	SPEED252	BOOL	FALSE	Start velocity for incremental approach with increment number 252 • 0 = Creep speed • 1 = Rapid traverse
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	4 = relative incremental approach 5 = absolute incremental approach
17.0	MODE_TYPE	BYTE	B#16#0	Incremental dimension 1 … 100, 252, 254 or 255
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = positioning is active
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
25.7	POS_RCD	BOOL	FALSE	1 = in position
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Address	Name	Type	Initial value	Comment
36.2	TRG252_254_EN	BOOL	FALSE	1 = Write incremental dimension for increment dimension number 252 or 254
36.3	TRG255_EN	BOOL	FALSE	1 = write increment for increment number 255
35.4	TRGL1WR_EN	BOOL	FALSE	1 = Write incremental dimension table 1 (incremental dimension number 1...50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = Write incremental dimension table 2 (incremental dimension number 51...100)
96.0	TRG252_254	DINT	L#0	Write incremental dimension for incremental dimension number 252 or 254
100.0	TRG255	DINT	L#0	Increment for increment number 255
104.0	CHGDIF_255	DINT	L#0	Changeover difference for increment number 255
108.0	CUTDIF_255	DINT	L#0	Switch-off difference for increment number 255

Data used in the parameter DB

Address	Name	Type	Initial value	Comment	
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus	
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus	
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus	
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus	
120.0	TRGL1.TRG[1]	DINT	L#0	Increment number 1	Increment table 1
. . .	. . .	. . .	. . .	. . .	
316.0	TRGL1.TRG[50]	DINT	L#0	Increment number 50	
320.0	TRGL2.TRG[51]	DINT	L#0	Increment number 51	Increment table 2
. . .	. . .	. . .	. . .	. . .	
516.0	TRGL2.TRG[100]	DINT	L#0	Increment number 100	

Distance-to-go

The distance-to-go is the difference between the target (increment) and process value. The distance-to-go is

- positive, if the target has not yet been reached
- negative, if the target has already been passed.

In the case of a rotary axis, the displayed distance-to-go cannot be used.

Shutting down the Incremental approach

The "Incremental approach" mode is shut down when the FM 451 receives a STOP signal (STOP = 1).

After traversing is shut down, the distance-to-go remains.

The distance-to-go for "Relative incremental approach" and "Absolute incremental approach with incremental dimension number 252" can be traversed to the end if

- the mode is unchanged, and
- the increment number is unchanged, and
- the direction is unchanged, and
- the remaining distance-to-go is greater than the assigned switch-off difference.

You traverse the residual distance by starting "Relative incremental approach" or "Absolute incremental approach with incremental dimension number 252" again without making any changes.

Canceling the incremental approach

The "Incremental approach" mode is cancelled when the "Drive enable" signal is deleted (DRV_EN = 0).

Delete distance-to-go

With the job "Delete distance-to-go" you can delete any awaiting distance-to-go.

With "Incremental approach with incremental dimension number 252", traversing restarts with continuous traversing after "Delete residual distance".

You also delete the awaiting distance-to-go by calling a different mode or by starting the mode in the other direction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.2	DELDIST_EN	BOOL	FALSE	1 = delete distance-to-go

The "Set actual value" / Set actual value on-the-fly / "Undo Set Actual Value" job (S7-300, S7-400)

Definition

You use the "Set actual value" or "Set actual value on-the-fly" jobs in order to assign a new coordinate to the current encoder value. The operating range is projected on a different area of the axis.

You determine the shift in the operating range with (ACT_{new}- $ACT_{current}$).

- $ACTUAL_{new}$ is the default value
- $ACTUAL_{current}$ is the actual value at the time of the execution

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.

For "Set actual value", the positioning process must be finished.

Job sequence

1. Enter the coordinate AVAL or FVAL for the actual value (**ACT**$_{new}$).
   - Linear axis:
     You must select the specified actual value so that the software limit switches are still within the permissible traversing range after the job is called.
     The amount of the shift resulting from (**ACT**$_{new}$ **- ACT**$_{current}$) must be less than or equal to the amount of the permissible traversing range (max. 100 m or 1000 m).
   - Rotary axis:
     Rule for the defined actual value:
     0 ≤ actual value < end of rotary axis
2. Set the relevant trigger bit (AVAL_EN = 1 or FVAL_EN = 1).

"Set actual value" is executed immediately.

"Set actual value on-the-fly" is executed at the next positive edge at digital input xI3.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.7	AVAL_EN	BOOL	FALSE	1 = set actual value
36.0	FVAL_EN	BOOL	FALSE	1 = Set actual value on-the-fly
84.0	AVAL	DINT	L#0	Coordinate for "Set actual value"
88.0	FVAL	DINT	L#0	Coordinate for "Set actual value on-the-fly"

Effects of the job

In the example "set actual value " to 300 mm you can see how this job projects the operating range onto a specified position of the axis. This produces the following effects:

- The actual position is set on the value of the actual value coordinate.
- The operating range is shifted on the axis.
- The individual points (e.g., software limit switch end) within the operating range maintain their original value, but have new positions.

The following table shows the shift of the axis operating range by "set reference point".

Set actual value	SLS	AC-TUAL	SLE
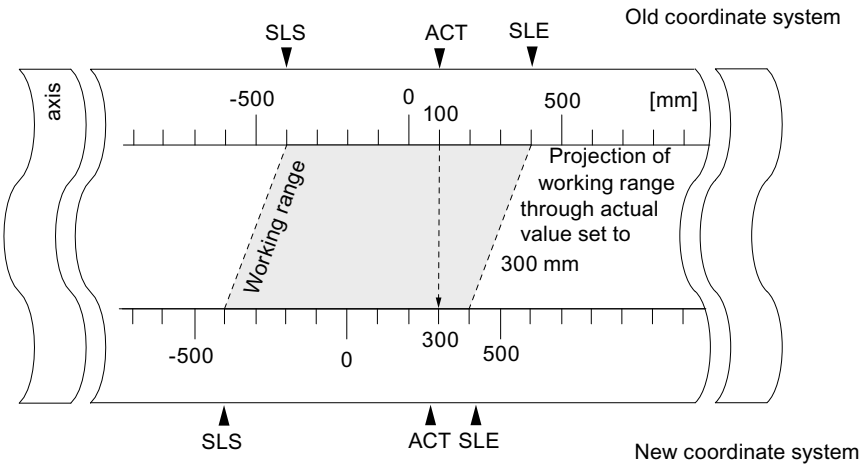	-400	100	400
	-400	300	400

Withdrawing the job ("Undo set actual value")

The "Undo set actual value" job can be used to reset the **most recent** operating range shift caused by "Set actual value".

Once "Set actual value on-the-fly" has been initiated, it can no longer be deleted before execution by a positive edge at digital input xI3. However, it can be overwritten by a new "Set actual value on-the-fly". These jobs are reset the next time the module starts up.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.3	AVALREM_EN	BOOL	FALSE	1 = cancel set actual value

"Zero offset" job (S7-300, S7-400)

Definition

The "zero offset" job lets you shift the zero point in the coordinate system by a defined value.

The sign determines the offset direction in the coordinate system.

Determining a new coordinate

All values of the shifted coordinate system can be calculated according to the following formula:

$\text{Coordinate}_{new} = \text{Coordinate}_{old} - (\text{ZPO}_{new} - \text{ZPO}_{old})$

ZPO_{old} designates an existing zero offset.

If no zero offset was active prior to the call, set the value 0 for **ZPO_{old}**.

You can use this to determine the coordinate values for the software limit switches, for example.

Requirements

- The positioning must be ended.
- The axis parameters must be assigned.

Job sequence

1. Enter the value ZOFF for the zero offset.
   - Linear axis:
     The zero offset must be selected so that the software limit switches remain within the permissible numerical range after the job is called.
   - Rotary axis:
     Rule for the zero offset:
     value of zero offset ≤ end of rotary axis
2. Set the relevant trigger bit (ZOFF_EN = 1).

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.1	ZOFF_EN	BOOL	FALSE	1 = Set zero offset
80.0	ZOFF	DINT	L#0	Zero offset

Effects of the job on a linear axis

Based on the example of a zero offset of -200 mm, you can see that this job shifts the coordinate system positively. This produces the following effects:

- The operating range is **not** physically shifted.
- The individual points (such as the software limit switches) are assigned new coordinate values.

The following table shows the offset of the coordinate system by "Zero offset".

Zero offset	SLS	ACTUAL	SLE
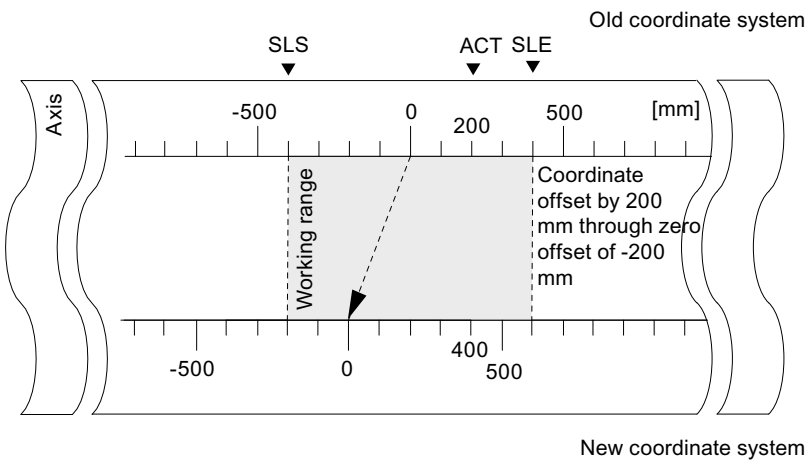	-400	200	400
	-200	400	600

Effects of the job on a rotary axis

Based on the example of a zero offset by -45°, you can see how this job **rotates** the coordinate system:

The following figure shows the revolution of the coordinate system by a zero offset.

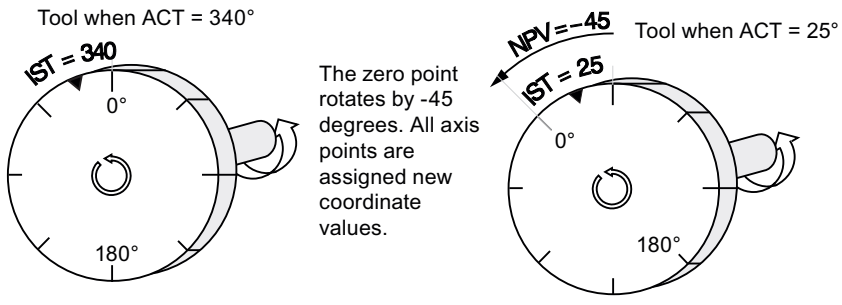

Including a **ZPO_{old} = 0**, the result is a new value of 385°.

Because the actual value starts over at 0 at the end of the rotary axis with a positive direction of rotation, the actual value of 25° is calculated from:

$Coordinate_{new}$ = $Coordinate_{old}$ - (ZPO_{new} - ZPO_{old}) - end of rotary axis

Loss of synchronization

If synchronization is lost due to an error or is reset due to "reference point approach", a zero offset **remains** active.

Canceling the job

A zero offset of 0 mm resets any existing zero offset.

"Set reference point" job (S7-300, S7-400)

Definition

You use the "set reference point" job to synchronize the axes. The job shifts the operating range. All the shifts that were created with "Set actual value" are retained.

The setting projects the operating range on the axis. Therefore, by entering different values, the operating range can be moved to any position within the physical area of the axis.

Prerequisites

- The positioning must have ended.
- The axis parameters must be assigned.

Job sequence

1. Enter the value REFPT for the reference point coordinate.
   - Linear axis:

     The reference point coordinates must not lie outside the software limit switches. This also applies to the reference point coordinate in a shifted coordinate system.
   - Rotary axis:
     Rule for the reference point coordinate:
     0 ≤ reference point coordinate < end of rotary axis
2. Set the relevant trigger bit (REFPT_EN = 1).

Data used in the channel DB

Address	Name	Type	Initial value	Comment
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
35.6	REFPT_EN	BOOL	FALSE	1 = set reference point
92.0	REFPT	DINT	L#0	Reference point coordinate

Effects of the job

In the example "set reference point" to 400 mm you can see how this job projects the operating range onto a specific, physical position of the axis. This has the following results:

- The actual position is set on the value of the reference point coordinate.
- The operating range is shifted on the axis.
- The individual points (e.g. software limit switch end) maintain their original value, but are located at new positions.
- The SYNC bit in the checkback signals is set.

The following table shows how "Set reference point" shifts the operating range on the axis.

Set reference point	SLS	ACTUAL	SLE
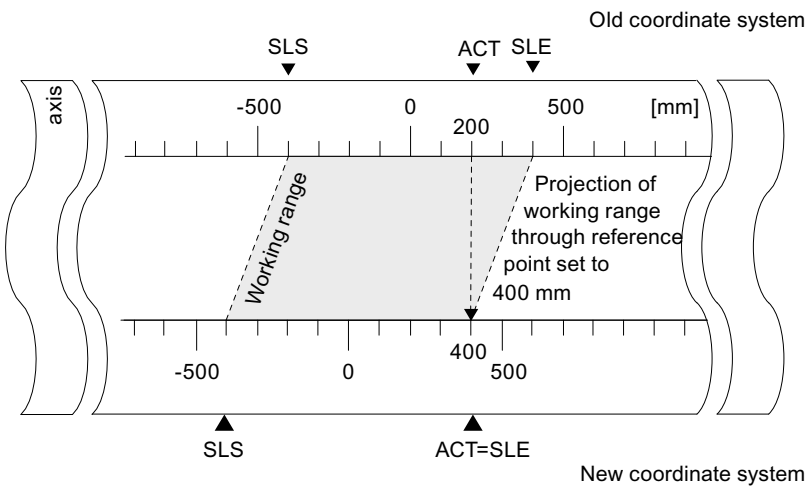	-400	200	400
	-400	400	400

Special features of the absolute encoder

This job is necessary for an absolute encoder adjustment (Page 2548).

"Length measurement" / "Edge detection" function switches (S7-300, S7-400)

Definition

The "length measurement" and "edge detection" let you determine the length of a workpiece.

The "length measurement" or "edge detection" function switch is active and remains active until you disable it or select the other measuring method. If you select both measuring methods in parallel, ABS_CTRL or ABS_CTRL_451 enables length measurement.

Requirements

- The positioning must be ended.
- The axis parameters must be assigned.
- The axis must be synchronized.
- A **bounce-free** switch must be connected to input xI3.
- If these function switches are selected during a positioning, "Length measurement" and "Edge detection" can only be executed with the next positioning.

Measuring sequence

Depending on the type of measurement, the FM 451 updates the data on the module at different times. FM 451 reports each update at a parameter on the feedback interface.

The following table shows the measuring sequence.

Sequence	Length measurement	Edge detection
	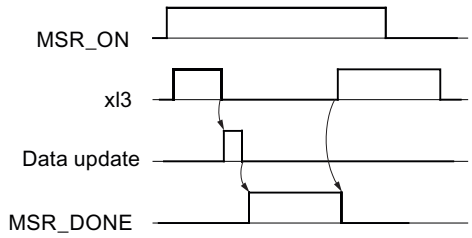	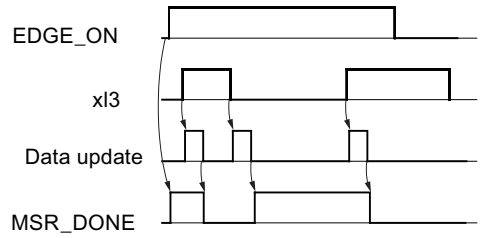
1	Set the function switch for "Length measurement" (MSR_ON = 1).	Enter a value EDGEDIST for the minimum edge-to-edge distance in the parameter DB. Write and enable machine data.
2	Start a positioning.	Set the "Edge detection" function switch (EDGE_ON = 1). This sets the MSR_DONE parameter.
3	A positive edge at input xI3 starts the length measurement.	Start a positioning.
4	A negative edge at input xI3 stops the current measurement. The FM 451 updates the initial value, end value, and length data.	A positive edge at input xI3 starts the measurement. After the "minimum edge distance" is traversed, the measurement is valid. This is signaled by MSR_DONE = 0. The results of the measurement are updated and can be read out, and the start value of the measurement is entered; the end value and length assume the value -1.
5	When parameter MSR_DONE is set, FM 451 reports the data update. The parameter indicates completion of the measurement. The results of the measurement can be read out.	After the update, FM 451 reports the change by resetting the MSR_DONE parameter.
6	The start of the next measurement at a positive edge at xI3 resets the MSR_DONE parameter.	A negative edge at input xI3 stops the current measurement. The FM 451 updates the data for the end measurement and length values.
7	-	After the update, FM 451 reports the change by setting the MSR_DONE parameter. The results of the measurement can be read out.
8	-	The start of the next measurement at a positive edge at xI3 resets the MSR_DONE parameter.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
25.1	MSR_DONE	BOOL	FALSE	1 = Length measurement/edge detection completed
34.3	EDGE_ON	BOOL	FALSE	1 = Edge detection ON
34.4	MSR_ON	BOOL	FALSE	1 = Length measurement ON
35.0	MDWR_EN	BOOL	FALSE	1 = Write machine data

Address	Name	Type	Initial value	Comment
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
37.0	MSRRD_EN	BOOL	FALSE	1 = Read measured values
136.0	BEG_VAL	DINT	L#0	Start value of length measurement/edge detection
140.0	END_VAL	DINT	L#0	End value of length measurement/edge detection
144.0	LEN_VAL	DINT	L#0	Length

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	Minimum edge distance for edge detection
The measurement will be discarded if the measuring operation ends within this range (BEG_VAL, END_VAL and LEN_VAL = -1).				

Boundary conditions for a length measurement

- The CPU program requires an interval of sufficient length between the off and on edges at input xI3 in order to correctly evaluate the result of the measurement before a new measurement starts.
- The minimum interval between the positive and negative edges at input xI3 and between the negative edge and the next positive edge at input xI3 must be greater than 8 ms.

Faulty measurement

The FM 451 returns the length value -1 in the event of faulty length measurement / edge detection. A length measurement is faulty if:

- The length measured for a rotary axis is greater than 2^{31}.
- the on and off edges are detected simultaneously by the FM 451 (for example, due to switch bounce).

"Loop traverse" function switch (S7-300, S7-400)

Definition

With "loop approach" you specify the direction in which a target is approached with a frictional connection. You can use loop approach if a frictional connection between the motor and axis can only be ensured in one direction.

A target that is approached in the direction opposite the specified direction is initially overrun. Then, the module performs a direction reversal and approaches the target in the specified direction.

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.
- In the case of a loop approach in the opposite direction to that of the traversing direction of the target the maximum target position is:
  - traversing direction plus
    Target < SLE - ½ Target range - Switch-off difference plus - Changeover difference minus
  - traversing direction minus
    Target < SLE - ½ Target range + Switch-off difference minus + Changeover difference minus
- A loop approach will not be carried out if the target is being approached in the direction of the loop approach. In this case, an incremental approach without an about-turn in direction will be carried out.
- The sequence of the "incremental approach" (Page 2510) operating mode must be known.

Sequence of events for the loop approach

1. Set the control signal for the "absolute/relative incremental approach" operating mode (MODE_IN = 5/4).
2. Set the control signal for the drive enable (DRV_EN = 1).
3. Set the "Enable input" (EI_OFF = 1) function switch or wire the enable input for the corresponding channel.
4. Enter the incremental dimension number (MODE_TYPE = 1 … 100, 254, 255).
5. Set the "Loop traverse" function switch (PLOOP_ON / MLOOP_ON = 1).
6. Start the incremental approach.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = start positioning
15.2	DIR_M	BOOL	FALSE	1 = minus direction
15.3	DIR_P	BOOL	FALSE	1 = plus direction
15.7	DRV_EN	BOOL	FALSE	1 = switch on drive enable
16.0	MODE_IN	BYTE	B#16#0	4/5 = relative / absolute incremental approach
17.0	MODE_TYPE	BYTE	B#16#0	Incremental dimension number 1 … 100, 254 or 255
34.0	PLOOP_ON	BOOL	FALSE	1 = loop approach in plus direction
34.1	MLOOP_ON	BOOL	FALSE	1 = loop approach in minus direction
34.2	EI_OFF	BOOL	FALSE	1 = do not evaluate enable input

Fictitious target

If you start a positioning to a target that is located in the direction opposite to that of the assigned loop traverse, the module determines a fictitious target for this target. The module performs a direction reversal at the fictitious target and then approaches the target in the proper direction.

This fictitious target **must** be located by at least half the target range before the respective software limit switch.

The distance of the fictitious target from the parameterized target is established depending on the direction: The following table shows the calculation of the location of the fictitious target with a loop traverse.

Guidelines	Location of fictitious target
Parameter assignment: Loop + (frictional connection plus) and approach in minus direction.	The fictitious target (**$target_f$**) has the value: **$target_f$ = target - switch-off difference minus - changeover difference plus** 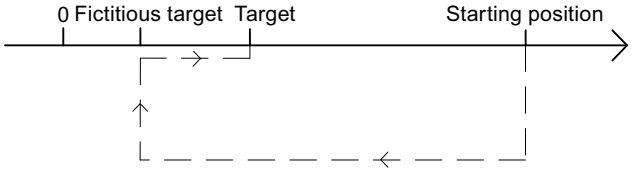
Parameter assignment: Loop - (frictional connection minus) and approach in plus direction.	The fictitious target (**$target_f$**) has the value: **$target_f$ = target + switch-off difference plus + changeover difference minus** 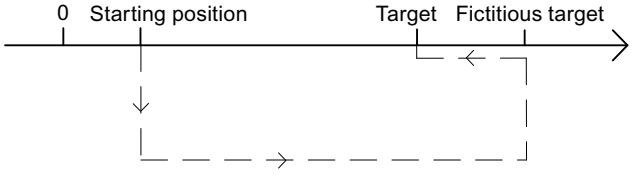

Example

Using a positioning with loop approach minus to a maximum target we can illustrate the location of the fictitious target.

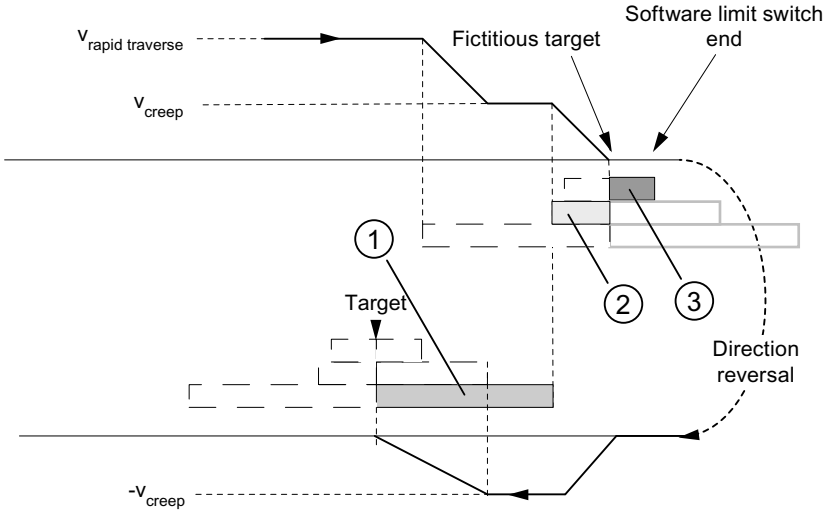

① Changeover difference negative
② Switch-off difference positive
③ ½ target range

"Enable input" function switch (S7-300, S7-400)

Definition

The enable input is an external input with which positioning can be enabled by an external event.

Evaluate enable input (EI_OFF = 0)

The relevant enable input (xI2) must be wired for the channel.

You have the option to prepare for the start of a positioning. You start the positioning independent of the program sequence of your user program by applying a "1" signal at the enable input.

You have the following options for evaluating the enable inputs:

- level-controlled (EI_TYPE = 0)
  The traverse begins when you set a "1" signal on the enable input and is shut down when you set a "0" signal on the enable input.
- edge-controlled (EI_TYPE = 1)
  Traversing begins with detection of a rising edge at the enable input. The additional signal characteristic at this input no longer affects the course of the motion that was started.

Do not evaluate enable input (EI_OFF = 1)

If you disable the evaluation of the enable input, an operating mode starts immediately after the start signal is detected. In this case, you cannot prepare an operating mode and start it at a future specified time.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
99.1	EI_TYPE	BOOL	FALSE	0 = Level-controlled 1 = Edge-controlled

"Read position data" job (S7-300, S7-400)

Definition

With the "read position data" job you can read the increment, distance-to-go and speed at the current time.

Job sequence

1. Set the trigger bit at the DB channel (ACTSPD_EN = 1).
2. The data are then stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.1	ACTSPD_EN	BOOL	FALSE	1 = read position data
112.0	ACTSPD	DINT	L#0	Current speed
116.0	DIST_TO_GO	DINT	L#0	Distance-to-go
120.0	ACT_TRG	DINT	L#0	Current increment

"Read encoder data" job (S7-300, S7-400)

Definition

With the "read encoder data" job you can read the current data of the encoder as well as the value for the absolute encoder adjustment.

Requirements

You can read the value for the absolute encoder adjustment (Page 2548) after you have carried out the "set reference point" (Page 2521) job.

Job sequence

1. Set the trigger bit at the DB channel (ENCVAL_EN = 1).
2. The data is stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.2	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
124.0	ENCVAL	DINT	L#0	Actual encoder value (internal representation)
128.0	ZEROVAL	DINT	L#0	Last zero mark value (internal representation)
132.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment

Checkback signals for the positioning (S7-300, S7-400)

Definition

The "checkback signals for the positioning" inform you of the current status of the positioning.

Sequence

The data is stored in the channel DB each time the ABS_CTRL or ABS_CTRL_451 instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = positioning is active
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
23.4	SPEED_OUT	BOOL	FALSE	0 = Creep speed 1 = Rapid traverse
23.5	ZSPEED	BOOL	FALSE	1 = axis is located in the standstill range
23.6	CUTOFF	BOOL	FALSE	1 = axis is located in the switch-off range
23.7	CHGOVER	BOOL	FALSE	1 = axis is located in the changeover range
24.0	MODE_OUT	BYTE	B#16#0	Active operating mode
25.2	GO_M	BOOL	FALSE	1 = axis moves in minus direction
25.3	GO_P	BOOL	FALSE	1 = axis moves in plus direction
25.5	FVAL_DONE	BOOL	FALSE	1 = Set process value on-the-fly completed

Address	Name	Type	Initial value	Comment
25.7	POS_RCD	BOOL	FALSE	1 = in position
26.0	ACT_POS	DINT	L#0	Current process value (current position of the axis)

Querying checkback signals for diagnostics (S7-300, S7-400)

Definition

The "checkback signals for the diagnostics" informs you of occurring diagnostic events.

Sequence

1. If the module enters a new event in the diagnostic buffer, it sets the DIAG bit in all the channels in the feedback interface. Error (Page 2574) events are logged in the diagnostics buffer. The DIAG bit is also set when the diagnostic buffer is deleted.
2. If it is not possible to call an operating mode or control an active operating mode, or if this has been carried out incorrectly, the module sets an OT_ERR operator error in the feedback interface. The cause of error is then entered in the diagnostic buffer. While the operator error exists, you can neither start a new operating mode nor continue with the halted one. You acknowledge an existing operating error with OT_ERR_A = 1.
3. The module sets the DATA_ERR bit in the feedback interface when it detects incorrect data in a write job. The cause of error is then entered in the diagnostic buffer.
4. The checkback signals are stored in the channel DB.
5. When the diagnostic buffer is read, the module sets the DIAG bit back to 0 in all channels.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
22.2	DIAG	BOOL	FALSE	1 = delete diagnostic buffer
22.3	OT_ERR	BOOL	FALSE	1 = operator error
22.4	DATA_ERR	BOOL	FALSE	1 = data error

6.2.6.3 Configuring FM 451 (S7-300, S7-400)

6.2.6.4 Parameter assignment FM 451 (S7-300, S7-400)

Specifying channels (S7-300, S7-400)

Selecting system of units (S7-300, S7-400)

Selecting system of units

For the input and output of the data, you can select from the following systems of units:

- Millimeter (default setting)
- Inches
- Degrees

Note

If you change the system of units in the "Specify channels" screen, the values will be converted into the new system. This can cause round-off errors.

If you change the system of units via the machine date, the values are not automatically converted.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
8.0	UNITS	DINT	L#1	**System of units** Valid values: • 1 = 10^{-3} mm • 2 = 10^{-4} inch • 3 = 10^{-4} degrees • 4 = 10^{-2} degrees • 6 = 10^{-3} degrees

Standard system of units in the online help

To specify the minimum and maximum values we use the mm system of units. To define the limits in other systems of units, convert the values as shown below:

To convert from		calculate
mm → inch		Limit value(inches) = limit value (mm) × 0.1 [1]
mm → degrees	10^{-4} (4 post-decimal places)	Limit value (degrees) = limit value (mm) x 0.1
	10^{-3} (3 post-decimal places)	Limit value (degrees) = limit value (mm) x 1
	10^{-2} (2 post-decimal places)	Limit value (degrees) = limit value (mm) x 10
inches → degrees		Limit (degrees) = limit (inches) .

[1] When set to the maximum value, the number of decimal places affects the number of digits before the decimal point. In the inch measurement system, 4 decimal places are used; you can therefore enter a maximum of 100,000.0000 inches. The millimeter measurement system uses 3 decimal places; a maximum of 1,000,000.000 mm can be entered.

Connection between increments and system of units

The encoder signals of a connected encoder will be evaluated by the module and converted into the current system of units. The resolution (Page 2554) is used for the conversion.

If the positioning module

- has counted 10 increments and
- a resolution of 100 µm per increment is assigned in the encoder data,

this means that the axis will be moved by a distance of 1 mm.

Copying channels (S7-300, S7-400)

Copying channels

You can copy the parameters of one channel to another.

1. Select the channel whose parameters are to be copied (source channel).
2. Select the channel to which the parameters are to be copied (target channel).
3. Press the "Copy" button.

Channel n drive (S7-300, S7-400)

Selecting control mode (S7-300, S7-400)

Overview of control modes (S7-300, S7-400)

Overview

The control mode specifies how a motor is operated at the four digital outputs (Q) of a channel by means of the power module. There are 4 control modes per channel for the positioning module:

- Control mode 1 (Page 2535)
- Control mode 2 (Page 2535)
- Control mode 3 (Page 2536)
- Control mode 4 (Page 2537)

The following table shows the states of the 4 digital outputs for each control mode, x stands for channel 1, 2 and 3.

	Rapid traverse		Creep speed		PR Position reached stop
	Direction +	Direction -	Direction +	Direction -	
Control mode 1					
xQ0	1	1	0	0	-
xQ1	0	0	1	1	-
xQ2	1	0	1	0	-
xQ3	0	1	0	1	-
Control mode 2					
xQ0	1	1	0	0	0
xQ1	0	0	0	0	1
xQ2	1	0	1	0	0
xQ3	0	1	0	1	0
Control mode 3					
xQ0	1	1	0	0	-
xQ1	1	1	1	1	-
xQ2	1	0	1	0	-
xQ3	0	1	0	1	-
Control mode 4					
xQ0	1	0	0	0	-
xQ1	1	0	1	0	-
xQ2	0	1	0	0	-
xQ3	0	1	0	1	-

Control mode 1 (S7-300, S7-400)

Control mode 1

The following settings apply for control mode 1 (x stands for channel n):

- xQ0: Rapid traverse
- xQ1 Creep speed
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 1.

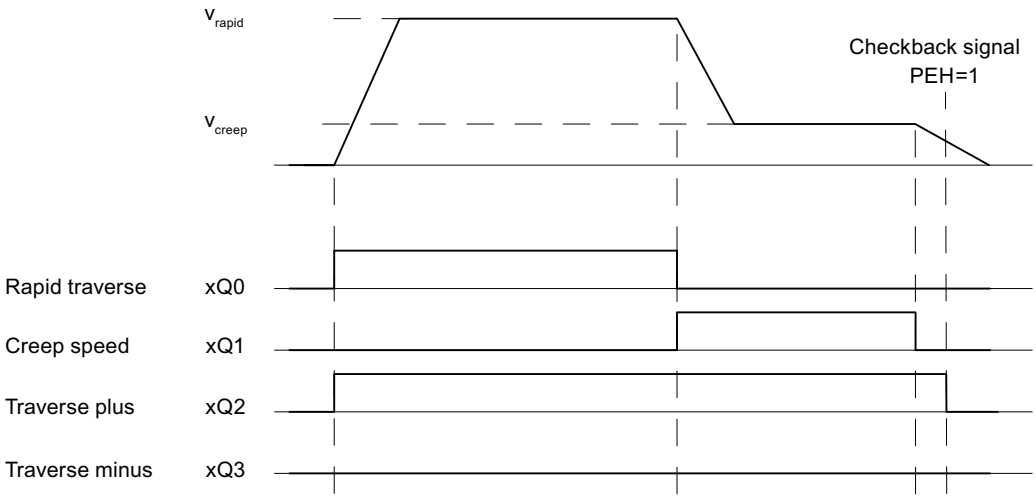

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	1 = Control mode 1

Control mode 2 (S7-300, S7-400)

Control mode 2

The following settings apply for control mode 2 (x stands for channel n):

- xQ0: Rapid traverse/creep speed
- xQ1 Position reached
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 2.

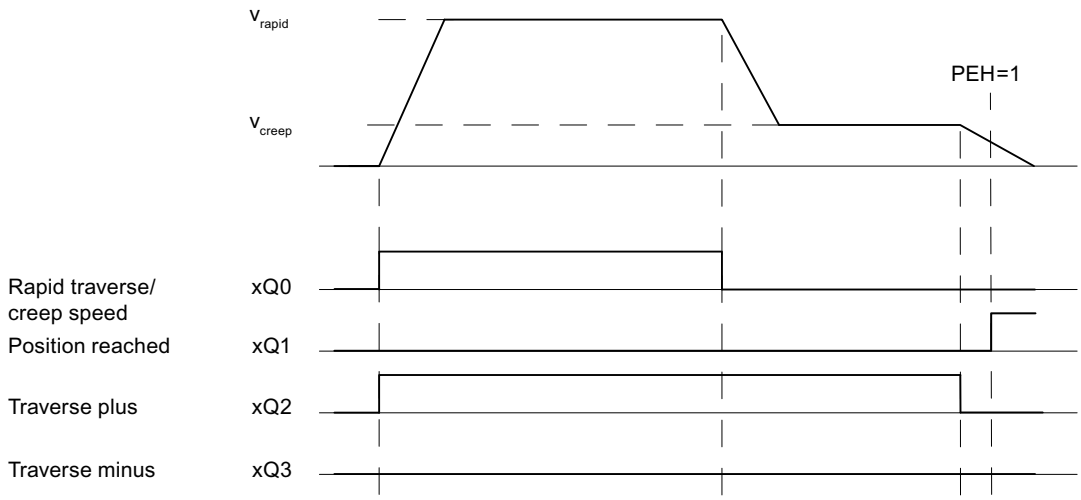

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	2 = Control mode 2

Control mode 3 (S7-300, S7-400)

Control mode 3

The following settings apply for control mode 3 (x stands for channel n):

- xQ0: Rapid traverse
- xQ1 Creep speed
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 3.

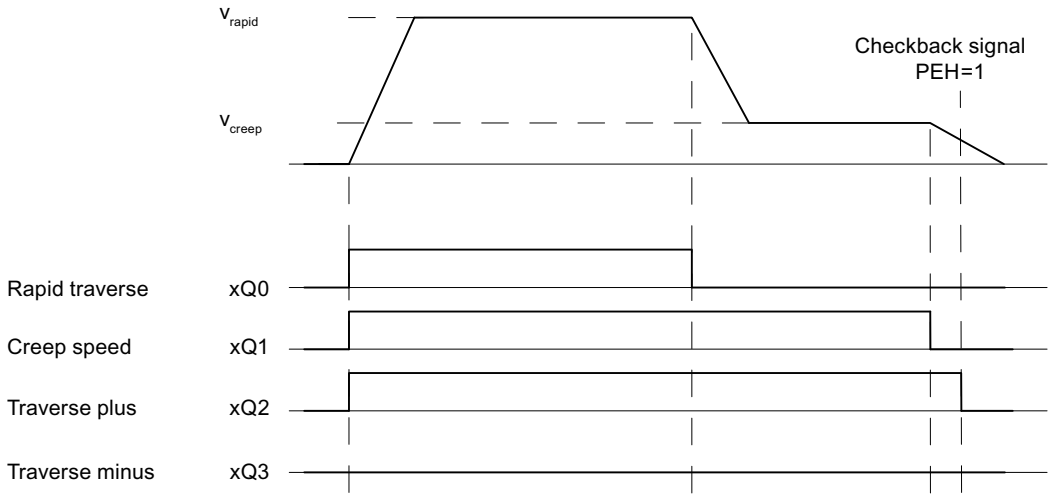

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	3 = Control mode 3

Control mode 4 (S7-300, S7-400)

Control mode 4

The following settings apply for control mode 4 (x stands for channel n):

- xQ0: Rapid traverse plus
- xQ1 Creep speed plus
- xQ2 Rapid traverse minus
- xQ3 Creep speed minus

The following image shows the function of the digital outputs with control mode 4.

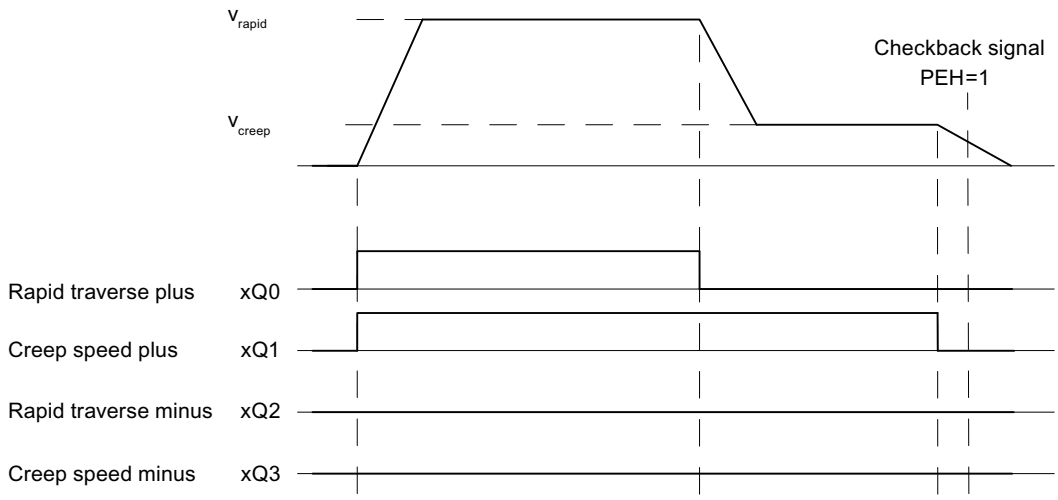

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	4 = Control mode 4

Specifying changeover difference (S7-300, S7-400)

Changeover difference

The changeover difference defines the distance from the destination at which the drive is switched over from rapid feed to creep feed.

The specified values apply to all targets approached by the positioning module.. An incremental approach with the incremental dimension 255 is an exception. A special changeover difference is specified for this.

Rules for changeover difference

- The changeover difference plus and the changeover difference minus do not have to be arranged symmetrically around the target. The values can differ.
- The changeover difference must be greater than the switch-off difference.
- The changeover difference must lie within the operating range.
- The changeover difference must be less than the end of rotary axis.
- Select a sufficient distance between the changeover point and the switch-off point to ensure that the drive can safely switch from rapid traverse to creep speed.
- The changeover point and the switch-off point must be separated from each other by a time interval of at least 8 ms.

The following image shows the location of the switch areas around the target.

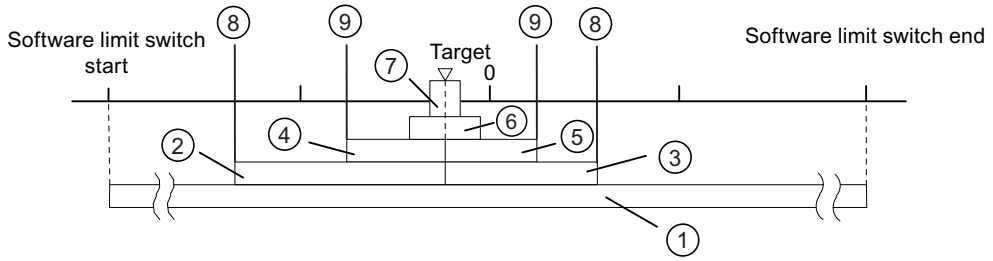

① Operating range
② Changeover difference plus / minus
③
④ Switch-off difference plus / minus
⑤
⑥ Standstill range
⑦ Target range
⑧ Changeover point
⑨ Switch-off point

For more information regarding the arrangement of the ranges, please see "Ranges and switching points of the positioning modules".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CHGDIF_P	DINT	L#5000	**Changeover difference plus**
104.0	CHGDIF_M	DINT	L#5000	**Changeover difference minus** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying switch-off difference (S7-300, S7-400)

Switch-off difference

The "switch-off difference" defines the distance to the destination at which the drive (creep speed) is switched off. The positioning module then adopts the monitoring functions from this point on.

The specified values apply to all targets approached by the positioning module.. An incremental approach with the incremental dimension 255 is an exception. A special switch-off difference is specified for this.

Rules for switch-off difference

- The switch-off difference plus and the switch-off difference minus do not have to be arranged symmetrically around the target. The values can differ.
- The switch-off difference must be greater than half the target range.

- Select the distance between the switch-off point and the target to ensure that the drive comes to a standstill within the target range.
- The switch-off point and the start of the target range must be separated from each other by a time interval of at least 8 ms.

The following image shows the location of the switch areas around the target.

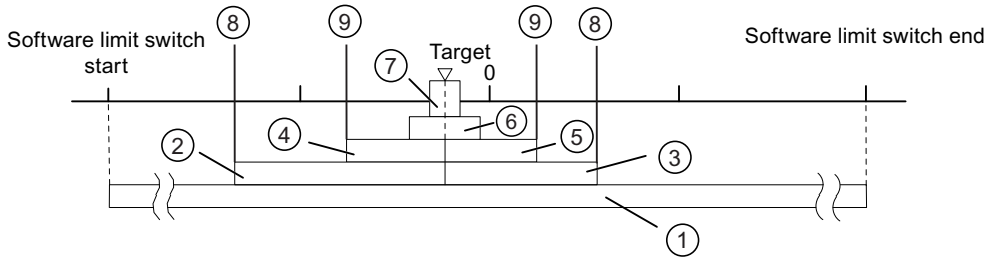

①	Operating range	⑦	Target range
②	Changeover difference plus / minus	⑧	Changeover point
③			
④	Switch-off difference plus / minus	⑨	Switch-off point
⑤			
⑥	Standstill range		

For more information regarding the arrangement of the ranges, please see "Ranges and switching points of the positioning modules".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
108.0	CUTDIF_P	DINT	L#2000	**Switch-off difference plus**
112.0	CUTDIF_M	DINT	L#2000	**Switch-off difference minus** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying target approach (S7-300, S7-400)

Specifying target range (S7-300, S7-400)

Target range

The positioning module sets a symmetrical range around each target, the target range. The actual value must lie within this range with a target approach.

A value specification of "0" switches off the tolerance during the target approach. The speed when reaching the target must then be less than the assigned standstill velocity (Page 2543) parameter.

Rules for the target range

- The target range lies symmetrically around the target.
- A value specification of "0" switches off the monitoring of the target range.

The following image shows the location of the ranges around the target.

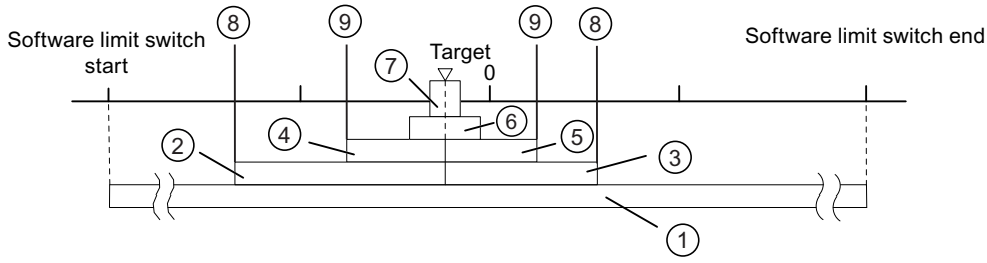

① Operating range
② Changeover difference plus / minus
③
④ Switch-off difference plus / minus
⑤
⑥ Standstill range
⑦ Target range
⑧ Changeover point
⑨ Switch-off point

Additional information on target approach can be found under "Ranges and switching points of the positioning modules" as well as under "End of a positioning operation (Page 2493)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	TRG_RANGE	DINT	L#1000	**Target range** Valid value range: • 0 = no monitoring • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying monitoring time (S7-300, S7-400)

Monitoring time

The module monitors the following during the monitoring time:

- the movement of the axis up to the switch-off point.
  The monitoring time starts with the start of a positioning and is retriggered with each actual value change in the traversing direction.
  Within the monitoring time, the actual value must have changed by at least one increment (the distance of the resolution).
- the target approach.
  The monitoring time is retriggered when the switch-off difference is reached for the last time.
  The target range must be reached within the monitoring time.

Additional information on target approach can be found under "End of a positioning operation (Page 2493)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
80.0	MON_TIME	DINT	L#2000	**Monitoring time** Valid value range: • 0 = no monitoring • 1 to 100 000 ms
Actual monitoring time You can specify all values from the range of values for the monitoring time. The module rounds the specified time to a multiple of 8 ms (module cycle).				

Specifying standstill range (S7-300, S7-400)

Standstill range

The standstill range lies symmetrically around the respective target. In this range, monitoring is performed to determine whether the drive remains on an approached target position or drifts away from it. The monitoring begins when the positioning module has reported "Position reached".

If the standstill range is exited without a valid traversing job, the positioning module signals an error.

Rules for standstill range

- The standstill range lies symmetrically around the respective target.
  (Target – ½ × standstill range) ≤ target ≤ (target + ½ × standstill range)
- The standstill range must be greater than or equal to the target range.

- The coordinate values for the standstill range must lie in the operating range.
- A setting of "0" deactivates the standstill monitoring function.

Additional information on target approach can be found under "Ranges and switching points of the positioning modules" as well as under "End of a positioning operation (Page 2493)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	ZSPEED_R	DINT	L#1000	**Standstill range** Valid value range: • 0 = no monitoring • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying standstill velocity (S7-300, S7-400)

Standstill velocity

The standstill velocity serves as a reference speed. If, during a positioning operation, the actual speed reaches the standstill velocity and the drive is located in the target range, then "Position reached" (PR) is set. The positioning operation is therefore successfully completed.

The precision of the positioning increases when the value of the standstill velocity is reduced. A setting of "0" deactivates the standstill monitoring function.

Additional information on target approach can be found under "End of a positioning operation (Page 2493)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
88.0	ZSPEED_L	DINT	L#30000	**Standstill velocity** Valid value range: • 0 = no monitoring • 1 µm/min to 100 000 µm/min

Channel n axis (S7-300, S7-400)

Specifying axis type (S7-300, S7-400)

Selecting axis type (S7-300, S7-400)

Axis types

The positioning module can control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.

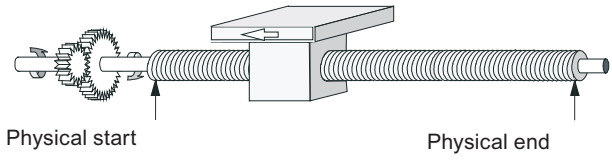

- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.

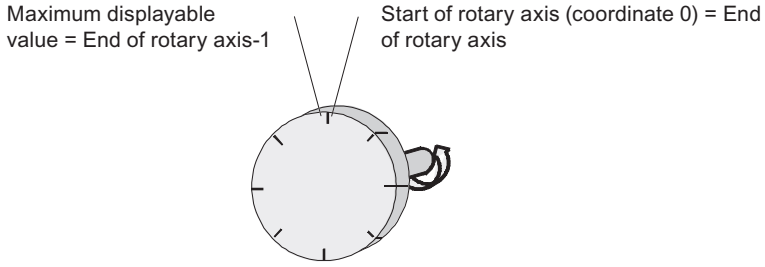

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
12.0	AXIS_TYPE	DINT	L#0	**Axis type** Valid values: • Linear axis = 0 • Rotary axis = 1

Specifying software limit switch (S7-300, S7-400)

Software limit switch

Software limit switches only exist for the linear axis.

The operating range is the range limited by the software limit switches.

The software limit switch start must always be smaller than the software limit switch end.

The software limit switches are monitored after the axis is synchronized.

- When using an incremental encoder:
  The synchronization of the axis is achieve by a reference point approach or by setting a reference point.
- When using an absolute encoder (SSI):
  The axis is synchronized when the module has received a complete, error-free frame for the associated channel.
  The absolute encoder must cover at least the operating range including the software limit switches.

Relationship between the operating range, encoder range and traversing range

- The operating range is the range which you specify for your job using the software limit switches.
- The encoder range represents the range explicitly covered by the encoder. For a linear axis, the module assigns the encoder range symmetrically across across the operating range. In other words, the module shifts the encoder range in order to equalize the distances between the software limit switches and the ends of the encoder range (see figure below).
- This traversing range represents the value range the positioning module can process. The traversing range is dependent on the resolution.

The following figure shows the relationship between software limit switches, the operating range, the encoder range and the traversing range.

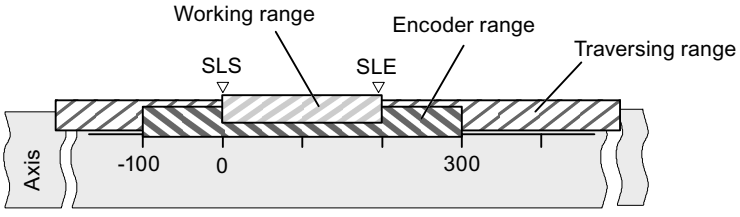

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L#-100000000	**Software limit switch start**
68.0	SSW_END	DINT	L#100000000	**Software limit switch end** Valid value range: • -1,000,000,000 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • -100,000,000 µm to 100,000,000 µm at resolution <1 µm/pulse

Specifying end of rotary axis (S7-300, S7-400)

End of rotary axis

The value "end of rotary axis" is the theoretical maximum value that the actual value can reach. The theoretically highest value is however never displayed, because it physically identifies the same position as the start of the rotary axis (0).

The maximum value that will be displayed for a rotary axis has the value:

End of rotary axis [µm] - resolution [µm / pulse] x 1 [pulse]

Example: End of rotary axis 1000 mm, resolution 1000 µm / pulse

The display jumps:

- with a positive rotational direction from 999 mm to 0 mm
- with a negative rotational direction from 0 mm to 999 mm.

Rotary axis with absolute encoders

The rotary axis range (0 to end of rotary axis) of a rotary axis with an absolute encoder must correspond exactly to the encoder range of the absolute encoder:
End of the rotary axis [µm] = number of encoder revolutions x distance/encoder revolutions [µm]

Data used in the DB parameter

Address	Name	Type	Initial value	Comment
16.0	ENDROTAX	DINT	L#100000	**End of rotary axis** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying reference point approach (S7-300, S7-400)

Selecting type of reference point approach (S7-300, S7-400)

Type of reference point approach

With "Type of reference point approach", you determine the conditions for the axis synchronization.

- The first specification defines the start direction in which the reference point approach is starting.
- The second specification defines the location of the zero mark, which leads to synchronization, as far as the reference point switch is concerned.

Detailed information can be found under "Reference point approach" operating mode (Page 2503).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
52.0	REFPT_TYPE	DINT	L#0	**Type of reference point approach** Valid values: • 0 = plus, reference point switch in plus direction • 1 = plus, reference point switch in minus direction • 2 = minus, reference point switch in plus direction • 3 = minus, reference point switch in minus direction

Specifying reference point coordinate (S7-300, S7-400)

Reference point coordinate

- **Incremental encoders:**
  You require the reference point coordinate for the "Reference point approach" mode. If the axis is not synchronized following writing and activating of machine data, the actual value is set to the value of the reference point coordinate.
- **Absolute encoder (SSI)**
  You require the reference point coordinates for the mechanical adjustment of the encoder. For more information, refer to the description of absolute encoder adjustment (Page 2548) which explains the interaction of the absolute encoder adjustment with other data.

The value of the reference point coordinate must lie within the operating range:

- **Linear axis**
  Including the software limit switches.
- **Rotary axis**
  Greater than or equal to 0 and less than the "end of rotary axis" value (0 ≤ reference point coordinate < end of rotary axis).

Data used in the DB parameter

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1 000 000 000 µm to 1 000 000 000 µm at resolution ≥1 µm/pulse • -100 000 000 µm to 100 000 000 µm at resolution < 1 µm/pulse

Specifying absolute encoder adjustment (S7-300, S7-400)

Determining the absolute encoder adjustment (S7-300, S7-400)

Absolute encoder adjustment

Absolute encoder adjustment is only necessary for an absolute encoder.

The absolute encoder adjustment and reference point coordinate map the encoder's range of values explicitly to the axis coordinate system.

Additional information can be found under "Absolute encoder adjustment procedure (Page 2548)" and in an example (Page 2549).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1 000 000 000 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • -100,000,000 µm to 100 000 000 µm at resolution <1 µm/pulse
48.0	ENC_ADJ	DINT	L#0	**Absolute encoder adjustment** Valid value range: • 0 to (2^{25} - 1)

Absolute encoder adjustment procedure (S7-300, S7-400)

Determining the correct absolute encoder adjustment

After initial parameter assignment, further steps must be taken to ensure that a correct relationship is created between the encoder and coordinate system. The procedure shown in the next section is based on the use of the parameter assignment dialogs.

1. Traverse the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch end".
2. Call the job "Set reference point (Page 2521)" with the coordinate of the point defined under 1.
   The positioning module now determines an encoder value for the reference point coordinate entered in the DB channel (REFPT in channel DB), namely the absolute encoder adjustment. You can read this value with the "Read encoder data (Page 2528)" job.
3. Enter the value read in the "absolute encoder adjustment" field.
4. Save your parameter assignment to the corresponding parameter DB using the export function.

5. Download the data in the HWCN to the CPU.
6. Restart the CPU (cold restart) to apply the data.

Note

You perform this adjustment just once during the commissioning. After a configuration, the position module is synchronized during start-up as soon as a complete, faultless message frame of the encoder is received after start-up.

Otherwise: Mechanical adjustment of the encoder

A correct relationship between the coordinate system and the encoder can be achieved as follows:

1. Move the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch start".
2. Enter this coordinate value in the machine data as "reference point coordinate".
3. Read the encoder value displayed at this position with the "Read encoder data (Page 2528)" job.
4. Enter this value in the machine data as the "absolute encoder adjustment".

After the parameter assignment a correct actual value will always be shown.

Instead of steps 3 and 4, you can also set the encoder to zero using "Reset", if available. Then enter the value "0" in the machine data as the absolute encoder adjustment.

Example of absolute encoder adjustment (S7-300, S7-400)

Example of an absolute encoder adjustment

The following assumptions apply to the example:

- Reference point coordinate = -125 mm
- Operating range from SSW_STRT = -1000 mm to SSW_END = 1000 mm
- Absolute encoder adjustment = 0
- Encoder range = 2048 increments with a resolution of 1 mm/pulse
- The absolute encoder used cannot be adjusted mechanically with precision and it also is not possible to set the encoder value specifically.

The following figure shows an example of absolute encoder adjustment.

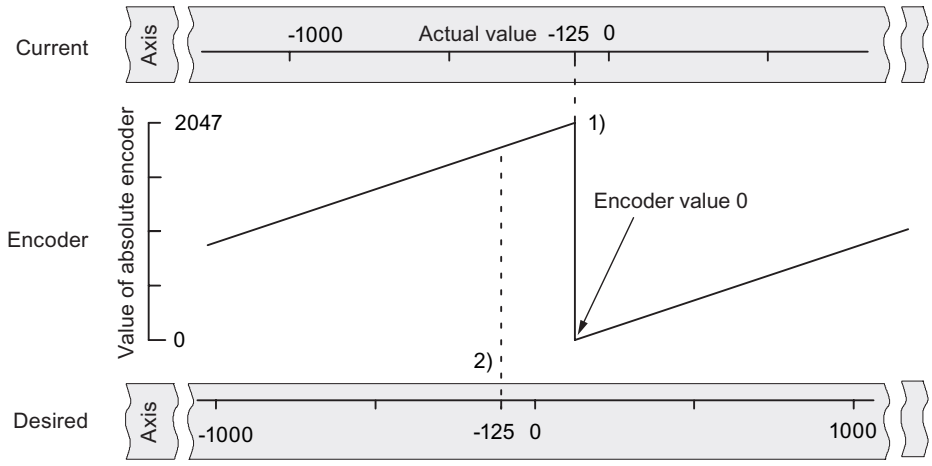

1) Assignment of the coordinate system to encoder values, based on the absolute encoder adjustment. The encoder value 0 corresponds to the actual value -125.

2) Required assignment of the coordinate system to the encoder. At this position the coordinate should be -125.

Result in accordance with "Set reference point"

After "Set reference point" the relationship looks as follows:

The reference point coordinate on the axis (-125) is assigned to the encoder value (1798) determined from absolute encoder adjustment. This assignment is set out in the figure below.

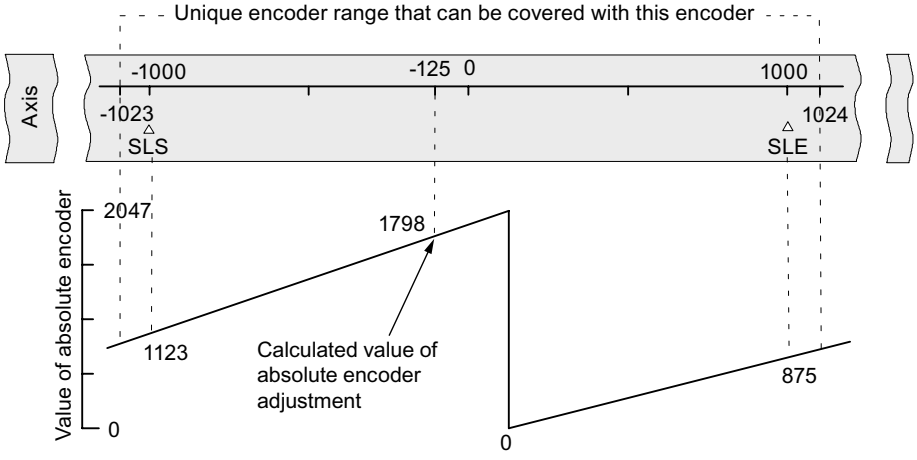

The encoder supplies 2048 unique values. The operating range is defined by the software limit switches. The selected resolution of 1 mm per pulse means the encoder can, however, cover a greater operating range than the one set by the software limit switches.

With the set resolution, the operating range is already covered with 2001 values. Therefore, in the example, 47 pulses "remain" and these are symmetrically positioned around the operating range.

Selecting start velocity (S7-300, S7-400)

Start speed in the case of a reference point approach

With this data you select the speed for the start of the reference point approach.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
99.0	REFPT_SPD	BOOL	TRUE	**Start speed for reference point approach** Valid values: • 0 = rapid traverse • 1 = creep speed

Specifying minimum edge-to-edge distance (S7-300, S7-400)

Minimum edge distance

With the machine data "Minimum edge-to-edge distance", you define a range after the start of a measurement is detected with an edge detection. The measurement will be discarded if the measuring operation ends within this range.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	**Minimum edge distance** Valid value range: • 0 to 1 000 000 000 µm at resolution ≥1 µm/pulse • 0 to 100 000 000 µm at resolution <1 µm/pulse

Channel n encoder (S7-300, S7-400)

Selecting the signal type (S7-300, S7-400)

Signal type

The encoder provides distance information to the module. The module then evaluates this information and converts this with the resolution into an actual value.

You can select the signal type from one of the following encoders:

- 5 V incremental encoder
  Select the signal type "5 V differential signal" to use a 5 V incremental encoder with differential signal A, /A, B, /B and N, /N.
- 24 V incremental encoder
  Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.
- Absolute encoder (SSI)
  Select the "SSI absolute" signal type if you use an absolute encoder with a serial interface.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
20.0	ENC_TYPE	DINT	L#1	**Signal type** Valid values: • 1 = 5 V differential signal • 2 = 24 V track A+B phase shifted • 3 = SSI 13-bit frame length [1)] • 4 = SSI 25-bit frame length [1)] • 5 = SSI 14-bit frame length [1)] • 6 = SSI 15-bit frame length [1)] • 7 = SSI 16-bit frame length [1)] • : • : • 14 = SSI 23-bit frame length [1)] • 15 = SSI 24-bit frame length [1)]

[1)] The frame length is selected in the "SSI" dialog box.

Selecting the count direction (S7-300, S7-400)

Count direction

With the machine data "Count direction" you can adapt the direction of the position measuring to the direction of movement of the axis. You must also take into account the rotational direction of all transmission elements (for example, couplings and gears).

- Normal = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent increasing actual position values
- Inverted = incremental count pulses (incremental encoder) or encoder values (absolute encoder) correspond to decreasing actual position values

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
59.0	CNT_DIR	BOOL	FALSE	**Count direction** Valid values: • 0 = normal • 1 = inverted

Specifying resolution (S7-300, S7-400)

Specify distance per encoder revolution (S7-300, S7-400)

Distance per encoder revolution

With the machine data "Distance per encoder revolution" you can inform the positioning module which distance the drive system should cover per encoder revolution.

The "Distance per encoder revolution" value depends on the structure of the axis and the way in which the encoder is mounted. You must take into account all the transfer elements like clutches or gearboxes. The following figure shows an example of two differently-mounted encoders.

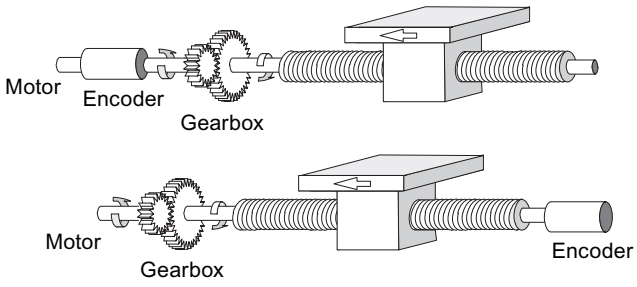

The relationship between the "Distance per encoder revolution" and "Increments per encoder revolution" machine data is described under resolution (Page 2554).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
24.0	DISP_REV	DINT	L#80000	**Distance per encoder revolution** Valid value range: • 1 µm to 1 000 000 000 µm

Specifying increments per encoder revolution (S7-300, S7-400)

Increments per encoder revolution

The machine data "Increments per encoder revolution" indicates the number of increments that an encoder makes for each revolution. The positioning module calculates the resolution (Page 2554) based on this value and the "Distance per encoder revolution" machine date.

Incremental encoder

Any value within the range of values can be entered. The module evaluates the increments four times.

Absolute encoder

For the limits you have to differentiate between the individual encoder models. Only values to the power of 2 are permitted as the entered value.

- Single-turn encoder (number of revolutions = 1) with 13-bit frame length:
  - minimum value = 4
  - maximum value = 8192
- Multiturn encoder (number of revolutions > 1) with 25-bit message frame length:
  - Minimum value = 4
  - Maximum value = 8192
- Single-turn encoder (number of revolutions = 1) with 25-bit frame length:
  - Minimum value = 4
  - maximum value = 2^{25}

Linear scales will be configured as multiturn encoders as follows:

- Increments per encoder revolution = 8192
- Number of revolutions x 8192 ≥ number of steps in the linear scale

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
32.0	INC_REV	DINT	L#500	**Increments per encoder revolution** Valid value range: • 1 to 2^{25}

Resolution (S7-300, S7-400)

Resolution

The resolution indicates the traversing distance corresponding to one pulse. It is a measurement for the precision of the positioning and also determines the possible maximum traversing range of the positioning module.

The resolution (RES) can be calculated as in the following table:

	Incremental encoder	Absolute encoder
Input values	• Distance per encoder revolution • Increments per encoder revolution: – Quadruple evaluation – 1 increment = 4 pulses	• Distance per encoder revolution • Increments per encoder revolution: – 1 increment = 1 pulse
Calculation	$\text{RES} = \dfrac{\dfrac{\text{Distance}}{\text{Encoder revolution}}}{\dfrac{\text{Pulses}}{\text{Encoder revolution}}}$	

Note

All position information is rounded up to the integral multiple of the resolution. This way the entered values differentiate from the used values.

Range of values of the resolution

The chosen system of units determines the range of values of the resolution:

System of units	Information in ...	Range of values of the resolution
mm	10^{-3} mm	$0{,}1 \times 10^{-3} \ldots 1000 \times 10^{-3}$ mm/pulse
inch	10^{-4} inch	$0{,}1 \times 10^{-4} \ldots 1000 \times 10^{-4}$ inch/pulse
degrees	10^{-4} degrees 10^{-3} degrees 10^{-2} degrees	$0{,}1 \times 10^{-4} \ldots 1000 \times 10^{-4}$ degrees/pulse $0{,}1 \times 10^{-3} \ldots 1000 \times 10^{-3}$ degrees/pulse $0{,}1 \times 10^{-2} \ldots 1000 \times 10^{-2}$ degrees/pulse

Example

- An incremental encoder has the following data:
  - Increments per encoder revolution: 5000
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 4 pulses

  This results in the resolution (4 x evaluation):

$$\text{Resolution} = \frac{1000\ \text{mm}}{5000\ \text{increments}} = 0.2000\ \frac{\text{mm}}{\text{Increment}} = 0.2000\ \frac{\text{mm}}{4\ \text{pulses}}$$

$$= 0.0500\ \frac{\text{mm}}{\text{Pulse}}$$

- An SSI encoder has the following data:
  - Increments per encoder revolution: 4096
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 1 pulse

  This results in the resolution:

$$\text{Resolution} = \frac{1000\ \text{mm}}{4096\ \text{increments}} = 0.2441\ \frac{\text{mm}}{\text{Increment}} = 0.2441\ \frac{\text{mm}}{\text{Pulse}}$$

Relationship between the traversing range and resolution

The traversing range is restricted by the numerical representation in the position module. This numerical representation varies depending on the resolution. You should thus make sure that your specifications are always within the permitted limits.

The maximum traversing range is illustrated in the following table:

Resolution (RES) lies in the range	Maximum traversing range
0.1 µm/pulse ≤ RES < 1 µm/pulse	-10^8 µm to 10^8 µm (-100 m to +100 m)
1 µm/pulse ≤ RES ≤ 1000 µm/pulse	-10^9 µm to 10^9 µm (-1000 m to +1000 m)

Date for absolute encoder (S7-300, S7-400)

Number of encoder revolutions (S7-300, S7-400)

Number of encoder revolutions

The machine date "Number of encoder revolutions" only applies to absolute encoders. As a result, enter the number of revolutions that are possible with this encoder.

The **overall number of steps of the encoder** is not machine data. It is calculated as follows:

Overall number of steps = increments/encoder revolution x number of revolutions

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
36.0	NO_REV	DINT	L#1	**Number of encoder revolutions** Valid value range: • 1 (single-turn encoder) • 2 to 4096 in powers of 2 (multiturn encoder)

Transmission rate (S7-300, S7-400)

Transmission rate

With the machine data "Transmission rate" you can determine the data transmission rate from the SSI encoder to the positioning module.

The maximum Transmission rate is dependent on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 63 m → 500 kHz
- 20 m → 1,000 kHz

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
40.0	BAUDRATE	DINT	L#0	**Transmission rate** Valid values: • 0 = 125 kHz • 1 = 250 kHz • 2 = 500 kHz • 3 = 1,000 kHz

Frame length (S7-300, S7-400)

Frame length

With "frame" you define the clock frame output by the positioning module for absolute encoder.

A frame between 13 bits and 25 bits is permitted.

Data used in the parameter DB

The frame length selected here is stored in the parameter DB under the ENC_TYPE (Page 2550) date in the parameter DB.

Parity (S7-300, S7-400)

Parity

With "Parity", you specify whether no parity, even parity or odd parity should be assigned for an absolute encoder.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
28.0	PARITY	DINT	L#0	**Parity** Valid values: • 0 = none • 1 = even parity • 2 = odd parity

Selecting monitoring (S7-300, S7-400)

Monitoring

Wire break

When monitoring is activated, the positioning module monitors all cables for a 5 V incremental encoder and an absolute encoder. The monitoring detects:

- Wire break
- Short circuit on individual cables
- Edge distance of counting pulses (also for 24 V incremental encoder)

For monitoring for a 24 V incremental encoder, you must set MON_TIME > 0 as the monitoring time (Page 2541).

With 5 V incremental encoders with zero marks, you must disable the wire break monitoring or interconnect the N and /N signals externally.

Frame error

The module monitors the message frame of an absolute encoder (SSI) for:

- Start and stop bit errors

You cannot disable the message frame error monitoring with absolute encoders (SSI).

Missing pulses (incremental encoder)

An incremental encoder must always return the same number of increments between two successive zero marks.

The positioning module checks to see if the zero mark of an incremental encoder is occurring at the correct encoder state.

For encoders without zero marks you must switch off the monitoring for missing pulses.

You must also disable the wire-break monitoring or interconnect the N and /N zero mark inputs externally.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
				Monitoring:
63.0	MON_WIRE	BOOL	TRUE	1 = wire break
63.1	MON_FRAME	BOOL	TRUE	1 = message frame error (must always be 1)
63.2	MON_PULSE	BOOL	TRUE	1 = missing pulses

Channel n incremental dimensions (S7-300, S7-400)

Specifying incremental dimensions (S7-300, S7-400)

Definition

Incremental dimensions are target specifications that can be controlled by the positioning module with the "relative incremental approach" or rather "absolute incremental approach" mode.

Possible range for incremental dimensions

The target that is to be approached must be at a distance of at least one-half the target range in front of the respective software limit switch. The following figure shows the limits for specifying incremental dimensions.

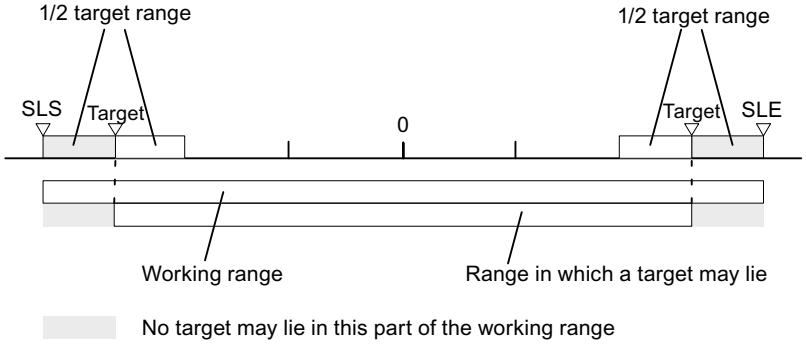

Increment number 1 to 100 (S7-300, S7-400)

Incremental dimension number 1 to 100

You have the possibility of entering up to 100 incremental dimensions in a table that are valid both for the "Relative incremental approach" mode as well as for the "Absolute incremental approach" mode.

Note that the positioning module does not permit negative values for the "relative incremental approach". The values will be interpreted by the positioning module - depending on the direction of movement - as either a positive or a negative difference.

Note

The entry is made in the unit of the system of units set. Note the decimal places in particular here.

Numerical example:

- Incremental dimension: 800 mm
- System of units: 10^{-3} mm
- Entry in the DB parameter: 800000

Tip: Define separate areas for relative and absolute incremental dimensions in the incremental dimension table.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.4	TRGL1WR_EN	BOOL	FALSE	1 = write increment table 1 (increments 1 ... 50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = write increment table 2 (increments 51 ... 100)
36.6	TRGL1RD_EN	BOOL	FALSE	1 = read increment table 1 (increments 1 ... 50)
36.7	TRGL2RD_EN	BOOL	FALSE	1 = read increment table 2 (increments 51 ... 100)

Data used in the parameter DB

Address	Name	Type	Initial value	Comment	
120.0	TRGL1.TRG[1]	DINT	L#0	Incremental dimension number 1	Increment table 1
. . .	. . .	. .	. . .	. . .	
316.0	TRGL1.TRG[50]	DINT	L#0	Incremental dimension number 50	
320.0	TRGL2.TRG[51]	DINT	L#0	Incremental dimension number 51	Increment table 2
. . .	. . .	. . .	. . .	. . .	
516.0	TRGL2.TRG[100]	DINT	L#0	Incremental dimension number 100	

Increment number 252 (S7-300, S7-400)

Increment number 252

The increment number 252 gives you the option of moving the drive initially in a specified direction without a target (continuous traversing) and positioning the drive at the assigned target (increment) on occurrence of an external event at digital input xI3.

The entries from the parameter DB for this increment are applicable to the changeover and switch-off differences.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.2	TRG252_254_EN	BOOL	FALSE	1 = Write incremental dimension for incremental dimension number 252 or 254
96.0	TRG252_254	DINT	L#0	Incremental dimension for incremental dimension number 252 or 254

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus

Increment number 254 (S7-300, S7-400)

Increment number 254

Independently of the increment table you can use the increment number 254 as an additional distance specification. For the changeover and switch-off differences, the entries from the parameter DB are valid for this increment.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.2	TRG252_254_EN	BOOL	FALSE	1 = Write incremental dimension for incremental dimension number 252 or 254
96.0	TRG252_254	DINT	L#0	Write incremental dimension for incremental dimension number 252 or 254

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus

Increment number 255 (S7-300, S7-400)

Incremental dimension number 255

The incremental dimension number 255 gives you an additional guideline for the path.

The switch-off differences and the changeover differences are presented together with the increment. Unlike the situation with the other increments, the increment 255 uses the values established in the channel DB for the switch-off and changeover differences. The entries from the machine data have no validity for this increment.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.3	TRG255_EN	BOOL	FALSE	1 = write increment for increment number 255
100.0	TRG255	DINT	L#0	Increment for increment number 255
104.0	CHGDIF255	DINT	L#0	Changeover differences for increment number 255
108.0	CUTDIF255	DINT	L#0	Switch-off differences for increment number 255

Control and feedback interface of the FM 451 (S7-300, S7-400)

Application

In special applications or in an alarm level, a particularly fast access to checkback and control signals could be required. You can reach this data directly via the input and output areas of the module.

To coordinate startup following each module startup, for example, after plugging a module or after a STOP-RUN transition of the CPU, you must call the ABS_CTRL instruction until the end of the startup is indicated by RETVAL = 0. Then, do not use the ABS_CTRL anymore.

Note

It is not possible to use the ABS_CTRL together with a write access.

Reading checkback signals by means of direct access

The byte addresses must be specified relative to the start address of the outputs of the respective channel. The names of the parameters correspond to the names in the channel DB.

Start address channel 1 = start address of the module

Start address channel 2 = start address of the module + 8

Start address channel 3 = start address of the module + 16

In STL, you access the data using the commands PEB (read 1 byte), PEW (read 2 bytes) and PED (read 4 bytes).

The following table shows the assignment of the feedback interface (inputs) of FM 451.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	PARA	Internal	Internal	DATA_ERR	OT_ERR	DIAG	Internal	Internal
Byte 1	CHGOVER	CUTOFF	ZSPEED	SPEED_OUT	0	WAIT_EI	WORKING	ST_ENBLD
Byte 2	MODE_OUT							
Byte 3	POS_RCD	0	FVAL_DONE	0	GO_P	GO_M	MSR_DONE	SYNC
Byte 4	ACT_POS							
Byte 5								
Byte 6								
Byte 7								

Example: Position process value ACT_POS

The start address of the module is 512.

STL	Description

L PED 516	//Read the current position process value (ACT_POS) from channel 1
	//with direct access:
	//Start address of the channel + 4

Writing control signals using direct access

The byte addresses must be specified relative to the start address of the inputs of the respective channel. The names of the parameters correspond to the names in the channel DB.

Start address channel 1 = start address of the module

Start address channel 2 = start address of the module + 8

Start address channel 3 = start address of the module + 16

In STL, you access the data using the commands PAB (write 1 byte), PAW (write 2 bytes) and PAD (write 4 bytes).

The following table shows the assignment of the control interface (outputs) of FM 451.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	OT_ERR_A	0	0	0
Byte 1	DRV_EN	SPEED252	0	0	DIR_P	DIR_M	STOP	START
Byte 2	MODE_IN							
Byte 3	MODE_TYPE							
Byte 4	Reserved							
Byte 5								
Byte 6								
Byte 7								

Example: START signals of channel 2

The start address of the module is 512.

```
STL                 Description
L 2#10001000        //Set DRV_EN and DIR_P to 1
T PAB 521           //Write signals with direct access for channel 2:
                    //Start address of the module + 8 + 1
```

Parameter transmission paths (S7-300, S7-400)

Parameters include the following machine data and increments.

The following figure shows the parameter transmission paths of the FM x51.

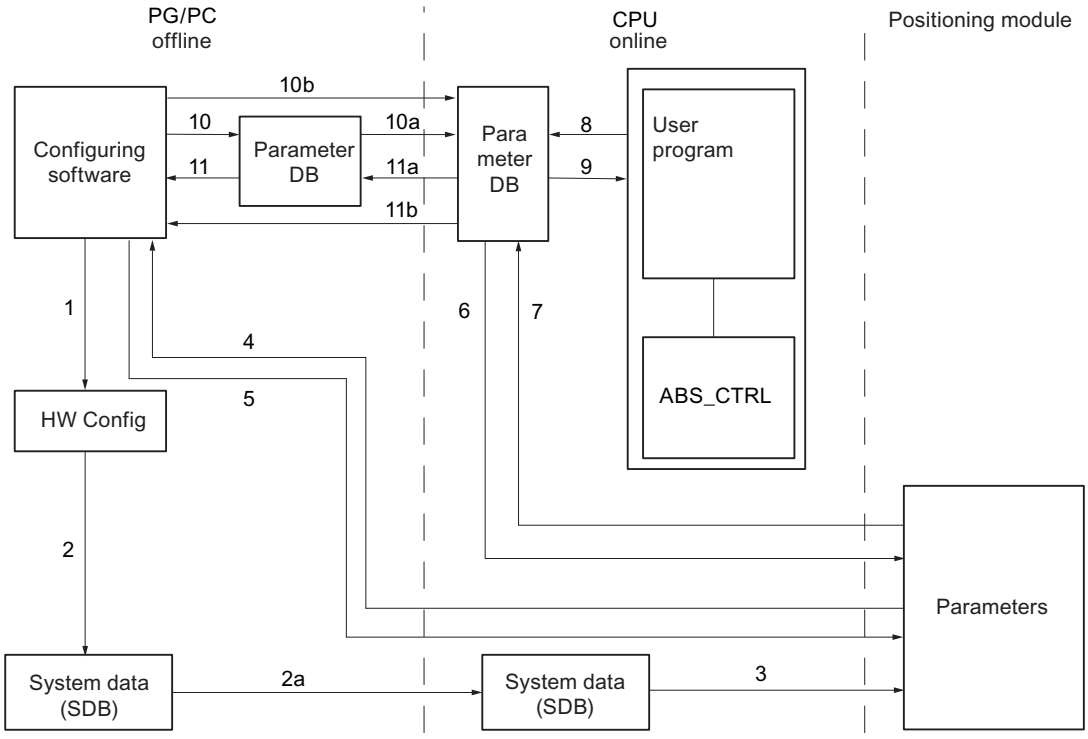

1 Saving parameters in the configuration software.

2 Save and compile the HW configuration.

2a Download the HW configuration to the CPU. The CPU automatically carries out step 3.

3 The CPU writes the parameters when system parameter assignments are carried out for the module.

4 With the command "download target system to PG" download the parameters of a channel of the module to the PG.

5 With the command "download target system" download the parameters from the configuration software into a channel of the module.

6 Write parameters using jobs of the user program in a channel of the module.

7 Read parameters using jobs of the user program from a channel of the module.

8 Store parameters from the user program into the online DB.

9 Read in parameters from the online DB into the user program.

10 Export parameters from the configuration software to the offline DB.

10a Download the offline DB to the CPU.

10b Export parameters from the configuration software to the online DB.

11 Import parameters from the offline DB into the configuration software.

11a Load parameters from the online DB to the PG.

11b Import parameters from the online DB into the configuration software.

Some use cases for the transfer of parameters

Use case	Steps
You process the parameters using the configuration software. The channels of the module should then be automatically assigned upon start-up.	Carry out steps 1, 2, and 2a.
You change the parameters during commissioning in test mode in the configuration software.	Carry out steps 4 and 5.
The parameters edited during commissioning should be downloaded automatically during startup.	Carry out steps 1, 2, and 2a.
You create the parameters using the configuration software. The channels of the module should be assigned upon start-up only by the user program via the data blocks.	Carry out steps 10, 10a and 6 or 10b and 6.
You would like to create conveniently stored data for recipes.	Carry out steps 10 and 10a.
You create the parameters using the configuration software. These should be made available to the user program for temporary changes.	Carry out steps 1, 2 and 2a for the automatic parameter assignment.
	Carry out the steps 10, 10a, 9, 8 and 6 for access by the user program.
You change existing parameters solely via the user program.	Carry out steps 7, 9, 8 and 6.
You would like to use the configuration software to view the data that has been changed with the user program.	Carry out steps 11a and 11 or just 11b.
Parameters modified by the user program should also be loaded automatically during startup.	Carry out steps 11b or 11a, 11 and then 1, 2, 2a.

6.2.6.5 Using FM 451 from the user program (S7-300, S7-400)

Integrating the FM 451 into a user program

Instructions are available for integrating the FM 451 into a user program. These enable you to handle the desired functions with ease.

You call the ABS_INIT and ABS_CTRL or ABS_CTRL_451 instructions in the CPU user program. They ensure communication between the CPU and the FM 451.

The channel DB is created automatically as the instance DB for the ABS_CTRL instruction when you insert the ABS_CTRL instruction into the project.
When you insert the ABS_CTRL_451 instruction into the project, use the command "Program blocks > Add new block" to add a ABS_CHANTYPE type data block to your project as the channel DB.

All the parameter data associated with the channel of the module (machine data, incremental dimension table) is located in the parameter DB for the module. Use the command "Program blocks > Add new block" to add a ABS_PARATYPE type data block to your project.

The ABS_DIAG or ABS_DIAG_451 instruction is also available for FM 451. This instruction lets you transfer the diagnostic information to the diagnostic DB.

The diagnostic DB is created automatically as the instance DB for the ABS_DIAG instruction when you insert the ABS_DIAG instruction into the project.
When you insert the ABS_DIAG_451 instruction into the project, use the command "Program

blocks > Add new block" to add a ABS_DIAGTYPE type data block to your project as the diagnostic DB.

Commissioning the module

The module can be initially tested using the "Commissioning" dialog, even when no user program is available. Note the following warning in particular in this case.

WARNING

Injury and material damage could occur.

To prevent personal injury and material damage, observe the following points:

- Install an emergency stop switch in the vicinity of the computer.
  This is the only way you can ensure that the plant can be safely shut down in the event of computer or software failure.
- Install hardware limit switches, which act directly on the power units of all drives.
- Make sure that no one has access to the area of the plant in which moving parts are present.
- When the FM 451 is controlled and monitored simultaneously by your program and the "Commissioning" dialog, conflicts can occur, the effects of which are not clear. Therefore, always set the CPU to STOP mode or disable your program when you work with the "Commissioning" dialog.

6.2.6.6 Commissioning the FM 451 (S7-300, S7-400)

6.2.6.7 FM 451: Online & Diagnostics (S7-300, S7-400)

Current module information (S7-300, S7-400)

Default diagnostics (S7-300, S7-400)

Error display through error LEDs of FM 451 (S7-300, S7-400)

Where is the error displayed?

The red error LEDs on the module display different error states. The LEDs are lit, even with errors that occur briefly, for at least 3 seconds.

The following figure shows the location of the error LEDs on FM 451.

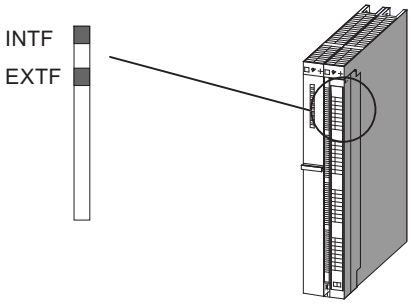

Which errors are displayed?

The table below shows the error types indicated by the error LEDs.

LED	Meaning	Explanations
INTF	Group error for internal error	This LED indicates the following error states of FM 451: • Module defect • Internal (channel) error • The FM 451 is not configured. Configure the module. To remove the error, see Error classes (Page 2574).
EXTF	Group error for external error	This LED indicates an external (channel) error.

Channel-specific diagnostics (S7-300, S7-400)

Triggering diagnostic interrupts (S7-300, S7-400)

Processing interrupts

The positioning module can trigger a diagnostic interrupt. These interrupts are processed in an interrupt OB. If an interrupt is triggered without the corresponding OB being loaded, the CPU switches to STOP.

Enable the processing of the diagnostic interrupts as follows:

1. Select the module in HWCN.
2. Select "Edit" > Object Properties > Basic Parameters" to enable the diagnostic interrupt.
3. Save and compile the configuration.
4. Download the configuration to the CPU.

Overview of diagnostic interrupts

The following events and errors trigger a diagnostic interrupt:

- Operating error
- Machine data error
- Diagnostics errors

These errors are explained in detail in "Error classes (Page 2574)".

Reaction of the positioning module to an error with diagnostic interrupt

- The positioning is canceled.
- The synchronization is deleted in the case of the following diagnostic interrupts:
  - Front connector missing, external auxiliary voltage for encoder supply missing
  - Zero mark error detected, cable fault (5 V encoder signals)
  - Traversing range exceeded (indicated by an operating error)
  - Set process value cannot be executed (indicated by an operating error).
- With one exception, control signals START, DIR_P und DIR_M are no longer processed.
  Exception:
  In the case of an operating error it is still possible to jog in the direction of the operating range.
- Function switches and jobs will no longer be processed.

Positioning module detects an error ("incoming")

A diagnostic interrupt is "incoming", if at least one error is given. If not all errors are remedied, the remaining existing errors will be reported again as "incoming".

Sequence:

1. The positioning module detects one or more errors and generates a diagnostic interrupt. Either the "INTF" or "EXTF" LED is illuminated. The error is then entered in the diagnostic buffer.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. You can evaluate the start info of OB 82.
4. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
5. For additional information, call the ABS_DIAG or ABS_DIAG_451 instruction.

The positioning module detects the transition to the error-free state ("outgoing")

A diagnostic interrupt is then only "outgoing" if the last error on the module has been remedied.

Sequence:

1. The positioning module detects that all errors have been eliminated and triggers a diagnostic interrupt. The "INTF" or "EXTF" LED is no longer illuminated. The diagnostic buffer is not changed.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
4. Evaluate the OB82_MDL_DEFECT bit.
   If this bit is "0", there are no errors on the module. You can end your evaluation here.

Diagnostic interrupts in dependence on the CPU status

- Diagnostic interrupts are blocked by the positioning module when the CPU is in STOP mode.
- If not all of the pending errors are eliminated while the CPU is in STOP mode, the positioning module reports the remaining errors as "incoming" events again at the next CPU transition to RUN.
- If all pending errors are eliminated while the CPU is in STOP mode, the module does **not** report its error-free state with a diagnostic interrupt at the next CPU transition to RUN.

Error evaluation in the user program (S7-300, S7-400)

Error response in the user program

In the user program you can respond selectively to errors. The following methods can be used:

- The RET_VAL return values of the integrated operations:
  This value is determined again each time the instruction is called.
  RET_VAL = -1 is a group display for a synchronous error in a job or when communicating with the module.
- An _ERR error bit belongs to each job. This is a group display for an error in the job or one of its predecessors in the job chain:
  The error bit is set for a write job and its subsequent jobs when a data error is reported by the module or a communication error occurs.
  With reading jobs the error bit is set for the job in question when a communication error occurs.
  The ABS_CTRL or ABS_CTRL_451 instruction resets the error bits once a job has been executed. These should however be cancelled by the user program in the case of an error assessment.
- The DATA_ERR checkback signal as a group display for an error that the module has detected during a write job. The signal will be established again with the next write job.
- The OT_ERR checkback signal (operator error) as a group display for an error that the module has detected when starting a traverse. The error must be acknowledged after the cause has been acknowledged with OT_ERR_A = 1.
- The DIAG checkback signal is set when the contents of the diagnostic buffer have changed. This signal can come along later than the signals DATA_ERR und OT_ERR.

- The communication error JOB_ERR includes the error code (Page 2573) in the case of a communication problem between the instruction and the module. The value is determined again after a job has been processed and is stored in the channel DB for the ABS_CTRL or ABS_CTRL_451 instruction in the channel DB and for the ABS_DIAG or ABS_DIAG_451 instruction in the diagnostics DB.
- The ABS_DIAG or ABS_DIAG_451 instruction for reading the diagnostic buffer of the module. Here you can discover the error causes for synchronous and asynchronous events.
- Diagnostic interrupts for the fast response to events in the diagnostic interrupt OB (e.g. OB 82).

Evaluation of a diagnostic interrupt in the user program

The positioning module sets the following entries in the local data of the diagnostic interrupt OB (e.g. OB 82). The errors will also be entered in the diagnostic buffer (error class 128 (Page 2581)).

Address	Name	Type	Comment
0.0	OB82_EV_CLASS	BYTE	Event class and identifiers: B#16#38: Outgoing event B#16#39: Incoming event
1.0	OB82_FLT_ID	BYTE	Error code (B#16#42)
2.0	OB82_PRIORITY	BYTE	Priority class: B#16#1A in RUN B#16#1C in START-UP
3.0	OB82_OB_NUMBR	BYTE	OB number (82)
4.0	OB82_RESERVED_1	BYTE	Reserved
5.0	OB82_IO_FLAG	BYTE	Input module: B#16#54
6.0	OB82_MDL_ADDR	INT	Logical base address of the module in which the error has occurred
8.0	OB82_MDL_DEFECT	BOOL	Module fault
8.1	OB82_INT_FAULT	BOOL	Internal error
8.2	OB82_EXT_FAULT	BOOL	External error
8.3	OB82_PNT_INFO	BOOL	Channel error
8.4	OB82_EXT_VOLTAGE	BOOL	Missing external auxiliary supply
...	Not used		
10.3	OB82_WTCH_DOG_FLT	BOOL	Time monitoring has responded
...	Not used		
12.0	OB82_DATE_TIME	DATE_AND_TIME	Date and time at which the OB was requested

Diagnostics buffer of the module (S7-300, S7-400)

Diagnostic events

The module´s diagnostic buffer includes maximum 9 diagnostic events and is organized as a ring buffer.

A diagnostic event is written in the buffer if an "incoming" (error) message is detected. This can be a message, a synchronous error (data error, operator error) or even an asynchronous error (operational error and diagnostics error). From a cause of error several entries can occur as follow-up errors. Outgoing messages do not create any entries in the diagnostic buffer.

For each diagnostic event the following is given:

- Status (always incoming)
- Internal error
- External error
- Error class
- Error number
- Channel number
- Increment number (in the case of increment table errors)

When a diagnostic event is written in the diagnostic buffer, the DIAG = 1 checkback signal is set in all assigned channels.

With the ABS_DIAG or ABS_DIAG_451 instruction, the diagnostic buffer can be transferred as a whole into a data block (diagnostic DB) or displayed via the evaluation dialog of the configuration software. If the diagnostic buffer is read, the module sets the checkback signal DIAG = 0.

Note

If the diagnostic buffer is read simultaneously from the ABS_DIAG or ABS_DIAG_451 instruction and the error evaluation dialog, it is possible that a new diagnostic event is not detected by the program.

Data and structure of the diagnostic DB (S7-300, S7-400)

Data and structure of the diagnostic DB

Note

You must not change any data that is not listed in this table.

The following table shows the structure of the diagnostic DB:

Address	Name	Type	Initial value	Comment
0.0	MOD_ADDR	INT	0	Module address
256.0	JOB_ERR	INT	0	Error number of the communication error
258.0	JOBBUSY	BOOL	FALSE	1 = job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = definitely read diagnostics buffer
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostics data - latest entry
272.0	DIAG[2]	STRUCT		Diagnostics data - second entry

Address	Name	Type	Initial value	Comment
282.0	DIAG[3]	STRUCT		Diagnostics data - third entry
292.0	DIAG[4]	STRUCT		Diagnostics data - fourth entry
302.0	DIAG[5]	STRUCT		Diagnostics data - fifth entry
312.0	DIAG[6]	STRUCT		Diagnostics data - sixth entry
322.0	DIAG[7]	STRUCT		Diagnostics data - seventh entry
332.0	DIAG[8]	STRUCT		Diagnostics data - eighth entry
342.0	DIAG[9]	STRUCT		Diagnostics data - ninth entry

The diagram below shows the structure of a DIAG[n] diagnostic entry.

Address	Name	Type	Initial value	Comment
+0.0	STATE	BOOL	FALSE	0 = incoming event 1 = outgoing event
+0.1	INTF	BOOL	FALSE	1 = internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 2: Operator errors 4: Data error 5: Machine data error 6: Incremental dimension table error 15: Alarms 128: Diagnostics errors
+4.0	FNO	INT	0	Error number
+6.0	CH_NO	INT	0	Channel number
+8.0	TRG_NO	INT	0	Incremental dimension number

List of JOB_ERR messages (S7-300, S7-400)

JOB_ERR alarms

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgement when reading from the module. The module has been removed during the read operation - or the module is faulty.
80A1	32929	-32607	Negative acknowledge when writing to the module. The module has been removed during the write operation - or the module is faulty.
80A2 [1)]	32930	-32606	Protocol error at layer 2
80A3 [1)]	32931	-32605	Protocol error involving user interface/user
80A4 [1)]	32932	-32604	Communication on C bus interrupted
80B0	32944	-32592	Data record / job unknown.
80B1	32945	-32591	Incorrect length specification. FM_TYPE parameter in the channel DB is not correctly set for the module being used.
80B2	32946	-32590	The configured slot is not assigned.

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80B3	32947	-32589	Actual module type dissimilar to the set module type.
80C0 [1]	32960	-32576	The module does not yet have ready the data to be read.
80C1 [1]	32961	-32575	The data of a same-named write job is not yet ready on the module.
80C2 [1]	32962	-32574	The module is currently processing the possible maximum of jobs.
80C3 [1]	32963	-32573	Required equipment (memory etc.) is currently in use.
80C4 [1]	32964	-32572	Communication error
80C5 [1]	32965	-32571	Distributed I/O unavailable.
80C6 [1]	32966	-32570	Priority class cancellation (restart or background)
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read from the DB. (write job)
8532	34098	-31438	DB number of the parameter DB too high (write job)
853A	34106	-31430	Parameter DB does not exist. (write job)
8544	34116	-31420	Error with the n-th (n > 1) read access on a DB after an error has occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written in the DB. (read job)
8730	34608	-30928	Parameter DB in the CPU is write-protected. The data cannot be written in the DB (read job).
8732	34610	-30926	DB number of the parameter DB too high (read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error with the n-th (n > 1) write access on a DB after an error has occurred. (read job)

[1] Errors 80A2 to 80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time.

Error classes (S7-300, S7-400)

Operating error (S7-300, S7-400)

Error class 1: Operating error

Operating errors are detected asynchronously to an operation / control. The operating errors bring about a cancellation of the positioning, apart from with error number 9, which causes the positioning to shut down.

The following table shows the possible operating errors and their respective causes:

No.	Meaning		Diagnostic interrupt
1	Software limit switch start overrun		Yes
	Cause	The actual value lies outside of the operating range.	
2	Software limit switch end overrun		Yes
	Cause	The actual value lies outside of the operating range.	

No.	Meaning		Diagnostic interrupt
3	**Traversing range start overrun**		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits belong with the traversing range).	
4	**Traversing range end overrun**		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits also belong to the traversing range)	
5	**Error on target approach**		Yes
	Cause	Target range has not been reached within the monitoring time.	
6	**Standstill range exited**		Yes
	Cause	The actual value lies outside of the standstill range.	
7	**Positive feedback**		Yes
	Cause	Actual value change > ½ standstill range in the wrong direction.	
8	**Missing or too slight actual value change**		Yes
	Cause	No actual value change or an actual value change against the set direction within the monitoring time.	
9	**Target overrun (FM 451)**		Yes
	Cause	The target has been overrun with "Set actual value on-the-fly".	
10	**Target range overrun**		Yes
	Cause	Target range has been overrun after target approach.	
11	**Changeover point switched incorrectly**		Yes
	Cause	Axis swinging in changeover point.	
12	**Switch-off point switched incorrectly**		Yes
	Cause	Axis swinging between switch-off / return point.	
13	**Start of target range switched incorrectly**		Yes
	Cause	Axis oscillating in the target range.	
14	**Change greater than half the rotary axis range**		Yes
	Cause	The velocity/frequency is too high or there are faulty actual value jumps.	
15	**Change greater than the rotary axis range**		Yes
	Cause	The velocity/frequency is too high or there are faulty actual value jumps.	
16	**Increment for increment number 252 cannot be transferred (FM 451)**		Yes
	Cause	The increment cannot be transferred.	
17	**Increment for increment number 252 cannot be approached (FM 451)**		Yes
	Cause	The distance between the current actual position and the specified increment is less than the changeover difference or switch-off difference.	
18	**Incorrect increment for increment number 252 (FM 451)**		Yes
	Cause	The increment lies outside of the operating range.	

Operator errors (S7-300, S7-400)

Error class 2: Operator errors

Operator errors have been detected when changing the control signals in the user data area. The operator errors cause the positioning to shut down.

The following table shows the possible operator errors and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Impermissible operating mode**		No
	Cause	The chosen operating mode is inadmissible.	
3	**Inadmissible interface job**		No
	Cause	The selected signal is inadmissible in this operating mode.	
4	**Incorrect operating mode parameter**		No
	Cause	• In "jog" mode the velocity specification is not equal to the rapid traverse or creep speed. • In "incremental approach" operating mode, the incremental dimension is a value other than 1 - 100, 252, 254, or 255	
5	**Start enable not available**		No
	Cause	Start enable not available upon start.	
7	**Target / target range lies outside of the operating range**		No
	Cause	Specified or calculated target is outside the software limit switches.	
8	**Axis parameters not assigned**		No
	Cause	Incorrect machine data or no machine data has been assigned for the axis.	
9	**Axis not synchronized**		No
	Cause	The "incremental approach" mode is only possible with an already synchronized axis.	
10	**Target/incremental distance cannot be positioned**		No
	Cause	The distance between the current actual position and the specified target is less than the switch-off difference.	
17	**Reference point approach not possible**		No
	Cause	An SSI encoder is connected.	
18	**Relative or absolute incremental approach is not possible**		No
	Cause	The increment is invalid.	
19	**Switch-off difference is less than ½ the target range with increment number 255**		No
	Cause	The switch-off difference for the increment number 255 is less than half the target range.	
20	**Inadmissible travel in the specified direction**		No
	Cause	The distance to the software limit switch is insufficient	

Data error (S7-300, S7-400)

Error class 4: Data error

Operating errors are detected synchronously to an operation / control. Data errors do not result in an error response.

The following table shows the possible data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
6	**Specified increment too great**		No
	Cause	The value lies outside of ±100 m or ±1000 m. The increment/target must not be greater than the traversing range. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	
10	**Faulty zero offset (FM 451)**		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. The software limit switches are positioned after the zero offset outside of the traversing range (100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	**Incorrect actual value**		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
12	**Incorrect reference point**		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
20	**Not permissible to activate machine data**		No
	Cause	There are no new, error-free machine data on the module.	
27	**Unauthorized bit-coded setting**		No
	Cause	Bits that are not used and not described here are not equal to 0.	
29	**Inadmissible bit coding**		No
	Cause	Unused and, in this case, unwritten bits are not 0.	
34	**Not possible to cancel set actual value**		No
	Cause	With an SSI encoder and a linear axis, the location of the actual value would lie outside the operating range after making the setting.	
36	**Incorrect changeover difference with the increment number 255**		No
	Cause	The value is outside the permitted numeric range of ±100 m or ±1000 m. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	
37	**Incorrect switch-off difference with the increment number 255**		No
	Cause	The value is outside the permitted numeric range of ±100 m or ±1000 m. The switch-off difference must be less than the changeover difference.	
107	**Axis parameters not assigned**		No
	Cause	On the axis there is either no machine data or it has not been enabled.	
108	**Axis not synchronized**		No
	Cause	One of the "set actual value" and "set actual value on-the-fly" jobs has been triggered even though the axis is not synchronized.	

Machine data error (S7-300, S7-400)

Error class 5: Machine data error

The diagnostic interrupt is only triggered in the event of a faulty system data block (SDB). Machine data errors do not result in an error response.

The following table shows the possible machine data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
5	**Error in hardware interrupt setting**		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	**Incorrect minimum edge distance (FM 451)**		Yes
	Cause	You have entered a value < 0 or > 10^9 µm as the minimum edge-to-edge distance.	
7	**Incorrect system of units**		Yes
	Cause	The value for the system of units lies outside of the permissible range of 1 to 4 and 6.	
8	**Incorrect axis type**		Yes
	Cause	You have entered neither 0 nor 1 as an axis type.	
9	**Incorrect end of rotary axis**		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or 1 to 10^8 µm (depending on the resolution).	
10	**Incorrect encoder type**		Yes
	Cause	The value for the encoder type lies outside of the permissible range of 1 to 4.	
11	**Incorrect distance per encoder revolution**		Yes
	Cause	The value for distance per encoder revolution lies outside the permissible range of 1 to 10^9 µm (irrespective of the resolution).	
13	**Incorrect number of increments per encoder revolution** (see "Increments per encoder revolution (Page 2553)")		Yes
14	**Incorrect number of revolutions** (see "Number of encoder revolutions (Page 2555)")		Yes
15	**Incorrect transmission rate**		Yes
	Cause	You have specified a value for the transmission rate that lies outside the permissible range of 0 to 3.	
16	**Incorrect reference point coordinate**		Yes
	Cause	The coordinate lies outside the range from -100 m to +100 m or -1000 m to +1000 m (depending on the resolution). Linear axis: The coordinate lies outside of the operating range. Rotary axis: The coordinate is greater than the end of the rotary axis or < 0.	
17	**Incorrect absolute encoder adjustment**		Yes
	Cause	SSI navigator: The value of the absolute encoder adjustment is not in the encoder range: (increments per encoder revolution x number of revolutions - 1).	
18	**Incorrect type of reference point approach**		Yes
	Cause	You have specified a value outside of the permissible value quantities of 0, 1, 2 and 3.	
19	**Incorrect counting direction**		Yes
	Cause	You have specified a value outside of the permissible value quantities of 0 and 1.	

No.	Meaning		Diagnostic interrupt
20	**It is not possible to monitor the hardware**		Yes
	Cause	You have set frame error monitoring to "FALSE" in the parameter DB. The monitoring of missing pulses is not possible with encoders being used. Deactivate the MON_PULSE parameter.	
21	**Incorrect software limit switch start**		Yes
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or 1,000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any existing zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	**Incorrect software limit switch end**		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or 1,000 m to +1000 m, depending on the resolution) or is less than the software limit switch start. Linear axis: The software limit switch end (including any existing zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
23	**Incorrect maximum velocity**		Yes
	Cause	The data that is not listed in the parameter DB must be 0.	
24	**Incorrect target range**		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m depending on resolution. Rotary axis: Range greater than the end of rotary axis.	
25	**Incorrect monitoring time**		Yes
	Cause	The value for the monitoring time lies outside the permissible range from 0 to 100 000 ms.	
26	**Incorrect standstill range**		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m, depending on the resolution. Rotary axis: Range greater than the end of rotary axis.	
127	**Incorrect standstill velocity**		Yes
	Cause	The value for the standstill velocity lies outside the permissible range from 0 to 100 000 µm/min.	
128	**Incorrect control mode**		Yes
	Cause	You have specified a value for the control mode that lies outside the permissible range of 1 to 4.	
129	**Incorrect start velocity for the reference point approach**		Yes
	Cause	You have entered neither 0 nor 1 as a start velocity.	
130	**Incorrect changeover difference in the direction +**		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m (depending on resolution). Rotary axis: Range greater than the end of rotary axis and smaller than ½ of the target range.	
131	**Incorrect changeover difference in the - direction**		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m (depending on the resolution). Rotary axis: Range greater than the end of rotary axis and smaller than ½ of the target range.	

No.	Meaning		Diagnostic interrupt
132	**Incorrect switch-off difference in the direction +**		Yes
	Cause	The switch-off difference is greater than the switch-over difference plus, less than ½ of the target range, or is outside the permissible range between 0 to 100 m or 1000 m (depending on the resolution)	
133	**Incorrect switch-off difference in the - direction**		Yes
	Cause	The switch-off difference is greater than the switch-over difference minus, less than ½ of the target range, or is outside the permissible range between 0 to 100 m or 1000 m (depending on the resolution)	
200	**Incorrect resolution**		Yes
	Cause	You have specified a resolution < 0.1 µm/pulse or >1000 µm/pulse. You have specified a distance per encoder revolution and a number of pulses per encoder revolution that results in a resolution of < 0.1 or > 1000.	
201	**Encoder does not fit the operating range / rotary axis range**		Yes
	Cause	SSI encoders and rotary axes: The encoder does not precisely cover the rotary axis range. Linear axis: The encoder does not cover at least the operating range (including software limit switches).	

Incremental dimension table error (S7-300, S7-400)

Error class 6: Incremental dimension table error

The incremental dimension table errors do not result in an error response.

The following table shows the possible incremental dimension errors and their respective causes:

No.	Meaning		Diagnostic interrupt
6	**Specified increment in the increment table too great**		No
	Cause	The value lies outside of ±100 m or ±1000 m. The increment/target must not be greater than the traversing range. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	

Alarms (S7-300, S7-400)

Error class 15: Alarms

Alarms do not result in an error response.

The following table shows the possible alarms and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Start of parameter assignment**		No
	Cause	The module has detected a parameter assignment via a system data block.	
2	**End of parameter assignment**		No
	Cause	The module has processed the parameter assignment via a system data block without errors.	
11	**Distance to change-over point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	
12	**Distance to return point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	
14	**Distance to cut-off point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	
15	**Distance to start of target range too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	

Diagnostic errors (S7-300, S7-400)

Error class 128: Diagnostic errors

The following table shows the possible diagnostic errors and their respective causes, effects and remedies.

No.	Meaning		Diagnostic interrupt
4	**Missing external auxiliary supply**		Yes
	Cause	• External 24 V auxiliary voltage is not connected or has failed • The fuse on the module is defective • Undervoltage • Ground wire break • Short circuit (e.g. at the encoder connected)	
	Effect	• The positioning is cancelled on all channels • Output switch-off • Synchronization is cancelled for incremental encoders when the auxiliary voltage for the encoder supply is missing • Parameters have not been assigned to the positioning module • Start enable is deleted	
	Remedy	Make sure that the 24 V connection is correct (if the 24 V connection is correct, then the module is defective)	

No.	Meaning		Diagnostic interrupt
5	**Front connector is missing (FM 451)**		Yes
	Cause	Front connector of the positioning module is not inserted	
	Effect	• No external 24 V auxiliary supply • Module is not ready	
	Correction	Plug the front connector into the positioning module	
51	**Time monitoring activated (watchdog)**		Yes
	Cause	• Strong interference on the positioning module • Error in the positioning module	
	Effect	• Module is reset • Turn off all outputs • If, after resetting the module, no module fault is detected, the module is ready for operation again • The module reports the expired watchdog with "incoming" and "outgoing"	
	Correction	• Remedying interferences • Contact the relevant sales department which will require details of the exact circumstances leading to the error • Replace the positioning module	
144	**Encoder wire break**		Yes
	Cause	• The encoder cable has sheared off or is not plugged in • Encoders without cross signals • Incorrect pin assignment • Cable length too long • Encoder signals short-circuited • Encoder signal edge fault • Maximum input frequency of the encoder input has been exceeded • Sensor voltage supply failure	
	Effect	• The positioning is canceled • Output switch-off • Delete the synchronization with incremental encoders • Start enable is deleted	
	Correction	• Check encoder cable • Comply with encoder specification • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog • Comply with the technical specifications of the module	

No.	Meaning		Diagnostic interrupt
145	**Absolute encoder frame error**		Yes
	Cause	The frame traffic between the positioning module and the absolute encoder (SSI) is incorrect or interrupted: • Encoder cable cut or not plugged in • Incorrect encoder type • The encoder is incorrectly set (programmable encoder) • The message length has been incorrectly specified • The encoder is supplying incorrect values (encoder is faulty) • Interference on measuring system cable • Chosen transmission rate is too high • Monoflop time of the encoder is greater than 64 µs	
	Effect	• The positioning is canceled • Output switch-off • Start enable is deleted	
	Correction	• Check encoder cable • Check encoder • Check the frame traffic between the encoder and positioning module	
146	**Missing pulses of incremental encoder**		Yes
	Cause	• Encoder monitoring has established a missing pulse • Incorrect entry of the number of increments per encoder revolution • Encoder faulty: does not return the specified number of pulses • incorrect or no zero mark • Crosstalk on the encoder cable	
	Effect	• The positioning is canceled • Output switch-off • Start enable is deleted	
	Correction	• Enter the correct number of increments per encoder revolution (programming dialog) • Check the encoder and encoder cable • Comply with the shielding and grounding directives • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog	

6.2.7 Using ET 200S 1PosU (S7-300, S7-400)

6.2.7.1 Terminal Assignment of the 1PosU (S7-300, S7-400)

Wiring rules

If a position encoder with 5 V differential signals is used, the wires to the terminals 9 and 13, the terminals 12 and 16, as well as at incremental encoders the wires to the terminals 11 and 15 have to be in twisted pairs and shielded.

If an incremental encoder with 24 V signals is used, the wires to the terminals 9, 11 and 12 have to be shielded.

The shield must be supported at both ends. To do this, use the shield contact (see the appendix of the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal assignment

The following table shows the terminal assignment of 1PosU.

View	Terminal assignment	Remarks			
		Connection of the Switches and the Drive: Terminals 1 to 8		Connection of the position encoder with 5 V differential signals or 24 V signals: Terminals 9 to 16	
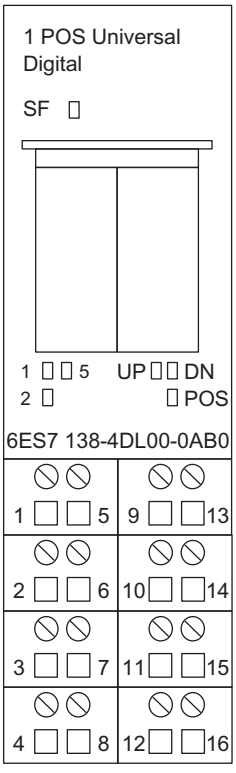	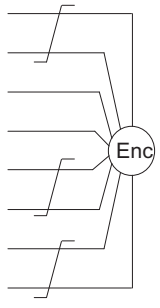	1: IN0	Limit switch minus	9: A / D	Track A / Data from the SSI encoder
		5: IN1	Limit switch plus	13: /A / /D	
		2: IN2	Reducing cam; latch signal	10: 24 V DC	Power supply for the position encoder
		6: 24 V DC	Supply for the switches	14: M	
		3: OUT0	Travel minus or rapid feed	11: B	Track B
		7: 2L+	Load voltage in-feed for OUT0, OUT1 and OUT2	15: /B	
		4: OUT1	Travel plus or creep feed	12: N / C	Track N / SSI clock (clock line)
		8: OUT2	Rapid/creep feed and travel plus/minus	16: /N / /C	

Connection of Relays and Contactors to the Digital Outputs

Note

Direct connection of inductivities (such as relays and contactors) is possible without external circuiting. If SIMATIC output circuits can be deactivated by additionally installed contacts (for example relay contacts), you have to provide additional overvoltage protection devices at inductivities (see the following example for overvoltage protection).

Overvoltage Protection Example

The following figure shows an output circuit that requires additional overvoltage protection devices. Direct-current coils are wired with diodes or Zener diodes.

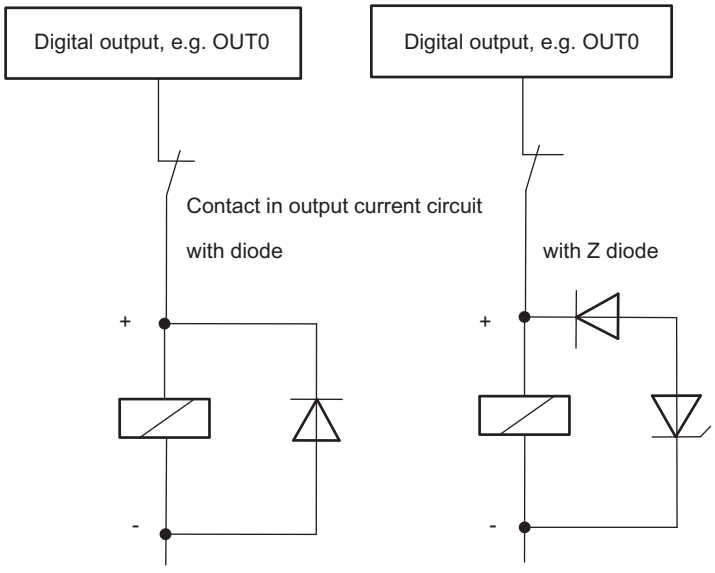

6.2.7.2 Basics of controlled positioning by means of rapid/creep feed (S7-300, S7-400)

Positioning process

From the start position, the target is approached at high velocity (rapid traverse). At a preset distance from the target (changeover point), there is a change to a lower velocity (creep speed). Shortly before the axis reaches the target, again at a preset distance from the target, the drive is switched off (switch-off point).

The drive is controlled via digital outputs for rapid traverse or creep speed and the appropriate direction.

The following figure shows the switch points and switch differences. To facilitate understanding, the change in velocity is illustrated over the path traversed.

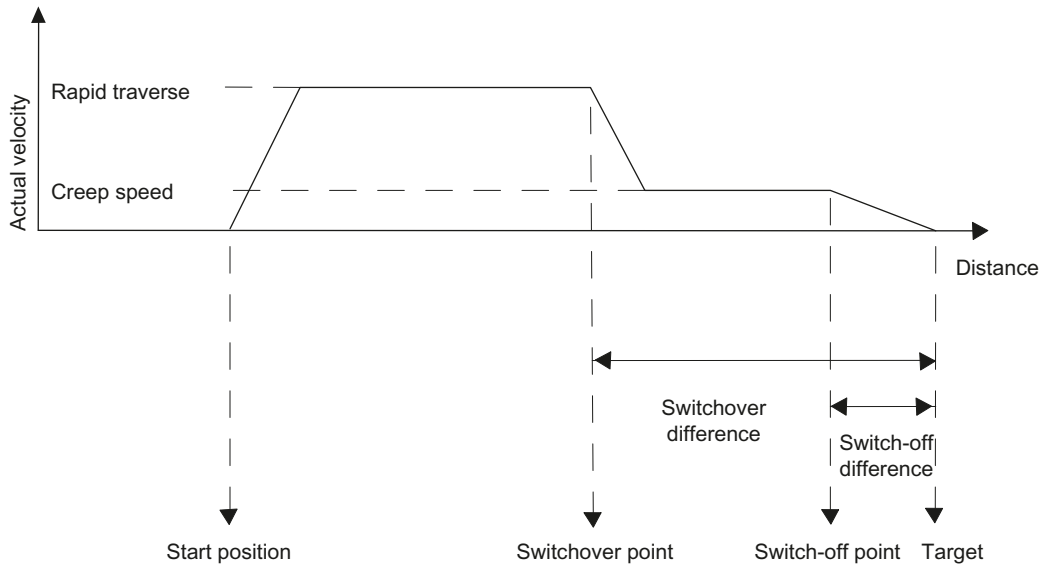

Definitions

The following tables contains the basic terms for controlled positioning using rapid traverse / creep speed.

Term	Description
Working range	defines the range which you set for a particular job by means of the hardware limit switches. At an SSI encoder, the operating range is also limited by the range covered by the SSI encoder. You enter the encoder range in the parameters for: • Pulses per encoder revolution and • revolutions. Encoder range = pulses per encoder revolution × revolutions Maximum operating range: • Linear axis max. 0 to (encoder range – 1) • Rotary axis from 0 to (encoder range - 1) At an incremental encoder the operating range is limited to: • Max. 0 to 16,777,215 increments at a linear axis • 0 to the programmed end of the rotary axis
Changeover difference	defines the distance from the destination at which the drive is switched over from rapid traverse to creep speed.
Changeover point	defines the position at which the drive is switched over from rapid traverse to creep speed.
Switch-off difference	defines the distance from the destination at which the drive is switched off. If the switch-off difference ≥ the changeover difference, there is no changeover point. There is no change from rapid traverse to creep speed.
Switch-off point	defines the position at which the drive is switched off. The 1PosU reports the end of the traverse at this point.

Term	Description
Start position	defines the position of the drive within the operating range from which the run is started. If the start position is within the switch-off difference, the drive is not triggered. The 1PosU reports the end of the traverse at this point. If the start position is within the changeover difference, the traverse is only executed at creep speed.
Target	defines the absolute or relative position of the axis approached during positioning. The target is the position to be reached on an axis during a traverse. In the case of an absolute traverse, you specify the target directly by means of your control program. In the case of a relative traverse, the destination is calculated from the start position and the distance specified in the control program. If you want to find out how accurately you have reached the destination, you have to compare the actual value with the position specified.
Linear axis	defines the axis type with a limited operating range. It is limited by the following: • the encoder range • the displayable numeric range (0 to 16 777 215 increments) • hardware limit switches
Rotary axis	defines the axis type with an infinite operating range. This includes resetting the axis position to 0 after one rotation (assigned parameter end of a rotary axis at an incremental encoder or parameterized encoder range at an SSI encoder).
Minus direction	If the drive moves in the minus direction, the actual value displayed is decreased.
Plus direction	If the drive moves in the plus direction, the actual value displayed is increased.

6.2.7.3 Functions of 1PosU (S7-300, S7-400)

Overview of the Functions (S7-300, S7-400)

Overview

The 1PosU offers you the following functions for moving your axis:

- Stop (Page 2593)
- Inching (Page 2593)
- Referance point approach (Page 2596)
- Relative Positioning (Page 2601)
- Absolute Positioning (Page 2603)

In addition to the different types of motion, the 1PosU also offers functions for:

- Cancelling JOB processing (Page 2607)
- Setting of Actual Value (Page 2608)
- Move Encoder Range (Page 2610)
- Change Switch-Off Difference (Page 2611)

- Change Changeover Difference (Page 2612)
- Reference Signal Evaluation (Page 2613)
- Latch Function (Page 2615)
- Setting the Monitoring of the Direction of Rotation (Page 2616)
- Display Current Values (Page 2618)
- Error detection / diagnostics (Page 2620)
- Behavior in CPU/Master-STOP (Page 2627)

Parameters

Define the variables that depend on the drive, axis, and encoder uniquely in the parameters.

Dosing mode

With incremental encoders, you can use the 1PosU for dosing. You set the dosing mode in the parameters. In dosing mode the 1PosU only evaluates the encoder signal A (/A). The actual value is incremented at each positive edge.

In dosing mode, only the "Inching" and "Relative positioning" functions are available for controlling the digital outputs.

The dosing function itself is triggered using the "Relative positioning" function. The dosing quantity is specified during starting by means of the control interface (distance).

At every start the actual value is set to 0 and the digital outputs are controlled as a function of the changeover and switch-off difference.

How the 1PosU works

The following figure shows how the 1PosU works.

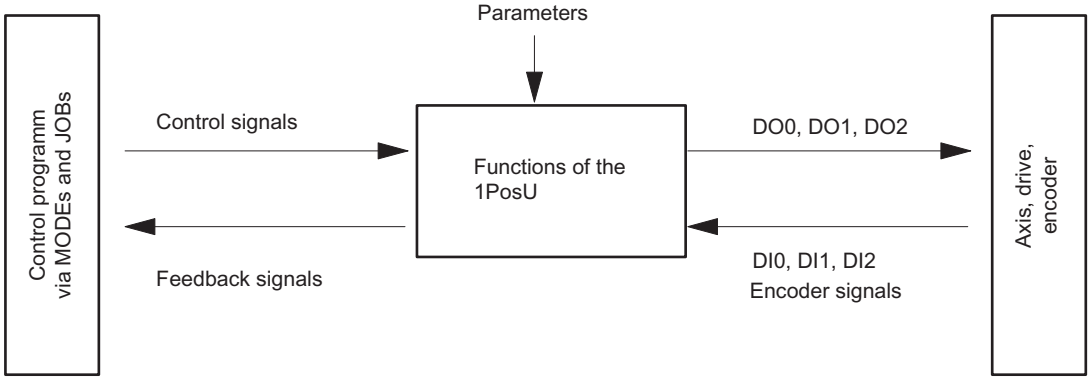

Interfaces to the Control Program and the Axis

To execute the function, the 1PosU has digital inputs as an interface to the axis, encoder signals for the connection to an encoder and digital outputs to control the drive.

Starting motion types (MODEs) (S7-300, S7-400)

You can control and monitor the types of motion (MODEs) with your control program using control signals and checkback signals.

Starting MODEs

The following table shows how you start MODEs and how the 1PosU reacts to them.

What You Do	1PosU reactions
Provide the control interface (Page 2646) with the data required depending on the MODE. Check whether the feedback bit (Page 2648) is POS_ACK = 0.	
Switch the START control bit from 0 to 1.	The 1PosU sets the POS_ACK = 1 and POS_DONE = 0 feedback bit. You can tell by this that the start has been detected by 1PosU and when POS_ERR = 0, the MODE is executed. When POS_ERR = 1, the MODE is not executed.
Switch the START control bit from 1 to 0.	The 1PosU sets the POS_ACK = 0 feedback bit.
	In the case of stopping, the reference point traverse and absolute and relative positioning, the 1PosU sets the POS_DONE = 1 feedback bit when the MODE is terminated without errors. When POS_ERR = 1, the MODE is terminated with errors.
You can start a new MODE only if POS_ACK = 0. If you start when a MODE is running, the 1PosU takes on the new motion and executes a change of direction, if necessary.	

The following figure shows the interaction of the control signals and checkback signals when starting the MODEs.

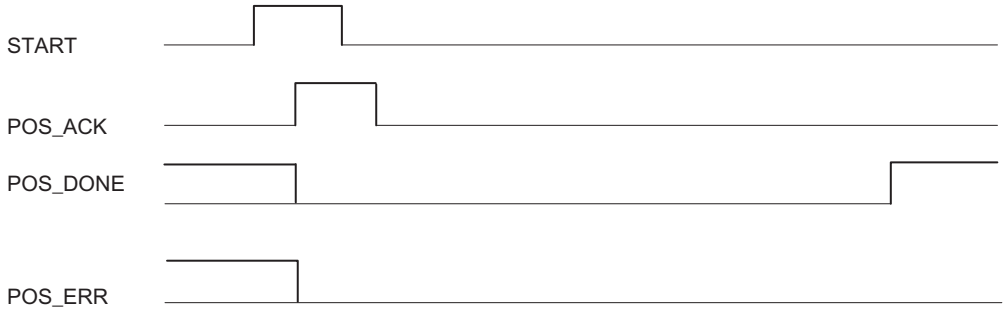

Activating further functions JOBs) (S7-300, S7-400)

You can control and monitor further functions (JOBs) with your control program using control signals and checkback signals.

Activating JOBs

The following table shows how you start JOBs and how the 1PosU reacts to them.

What You Do	1PosU reactions
Provide the control interface (Page 2646) with the data required depending on the JOB. Check whether the feedback bit (Page 2648) is JOB_ACK = 0.	
Switch the JOB_REQ control bit from 0 to 1.	The 1PosU sets the JOB_ACK = 1 feedback bit. You can tell by this that the start has been detected by 1PosU and when JOB_ERR = 0, the JOB is executed. • With "Evaluate reference signal", the 1PosU sets the SYNC = 0 feedback bit simultaneously. • In the case of the latch function, the 1PosU sets the LATCH_DONE = 0 feedback bit simultaneously. • All the other JOBs are then executed. When JOB_ERR = 1, the JOB is not executed.
Switch the JOB_REQ control bit from 1 to 0.	The 1PosU sets the JOB_ACK = 0 feedback bit.
	When a reference signal is evaluated, the 1PosU sets the SYNC = 1 feedback bit when the function has been executed. With "Latch", the 1PosU sets the LATCH_DONE = 1 feedback bit when the function has been executed.
You can only activate a new JOB if JOB_ACK = 0.	

The following figure shows the interaction of the control signals and checkback signals when starting the JOBs.

Axis, Drive and Encoder (S7-300, S7-400)

Evaluation of the Encoder Signals

The 1PosU evaluates the signals supplied by the encoder differently depending on the encoder type:

- SSI encoder:
  The 1PosU evaluates the encoder value supplied by the SSI encoder directly in increments and forms the actual value in increments (actual value = encoder value).
  The actual value lies in the encoder range from 0 ... (number of rotations * number of increments) – 1. The 1PosU generates an high limit violated / low limit violated of the actual value at the limits of the encoder operating range.
- Incremental encoder:
  The 1PosU evaluates the pulses supplied by the position encoder four times and adds them up direction-specifically to form the actual value. You must take the quadruple evaluation into account when you make settings for paths in the parameters and in the control and feedback interfaces:
  1 pulse of the incremental encoder corresponds to 4 increments of the 1PosU.
  The actual value lies in the operating range of 0 - 16 777 215 increments. The 1PosU generates an high limit violated or low limit violated of the actual value at the limits of the operating range.
- Incremental encoder in dosing mode:
  The 1PosU only evaluates the positive edges of the encoder A signal and adds them up to form the actual value.
  The actual value lies in the operating range of 0 - 16 777 215 increments. The 1PosU creates an high limit violated of the actual value at the high limit of the operating range.

Controlling the Drive

The drive is controlled using the 3 digital outputs of the 1PosU.

You can select the velocity with the SPEED control bit (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse). You can also change the speed during the run.

You can bring about a change in direction (Page 2634) with the "T_{min} direction change" parameter.

You can read the status of the respective output from the feedback interface (Page 2648) (DO 0, DO 1 and DO 2).

The function of the digital outputs depends on the control mode:

The following figure shows the states of the digital outputs with **control mode 0**.

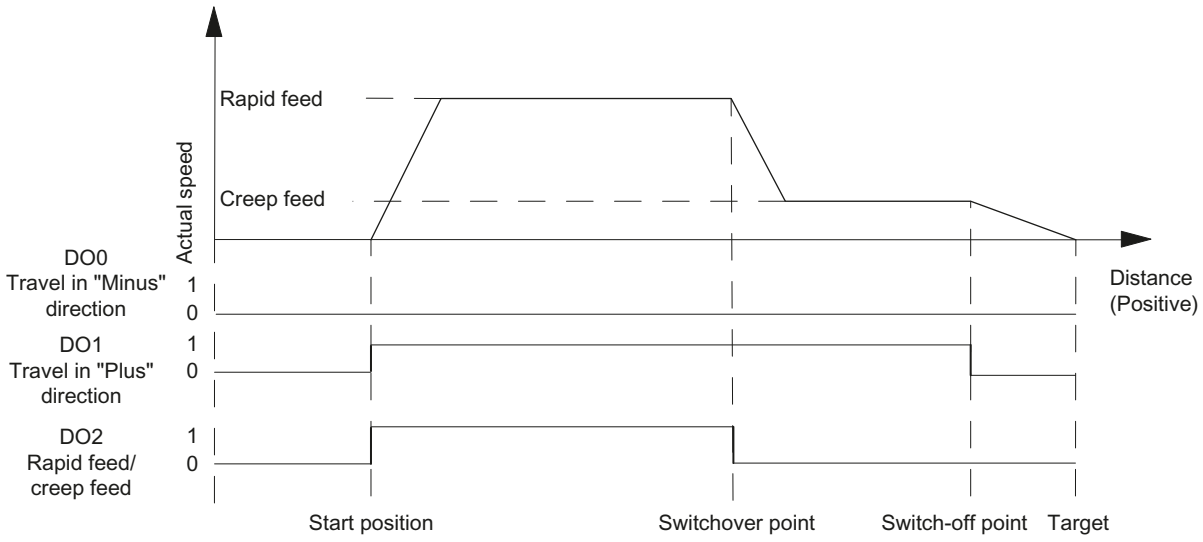

The following figure shows the states of the digital outputs with **control mode 1**.

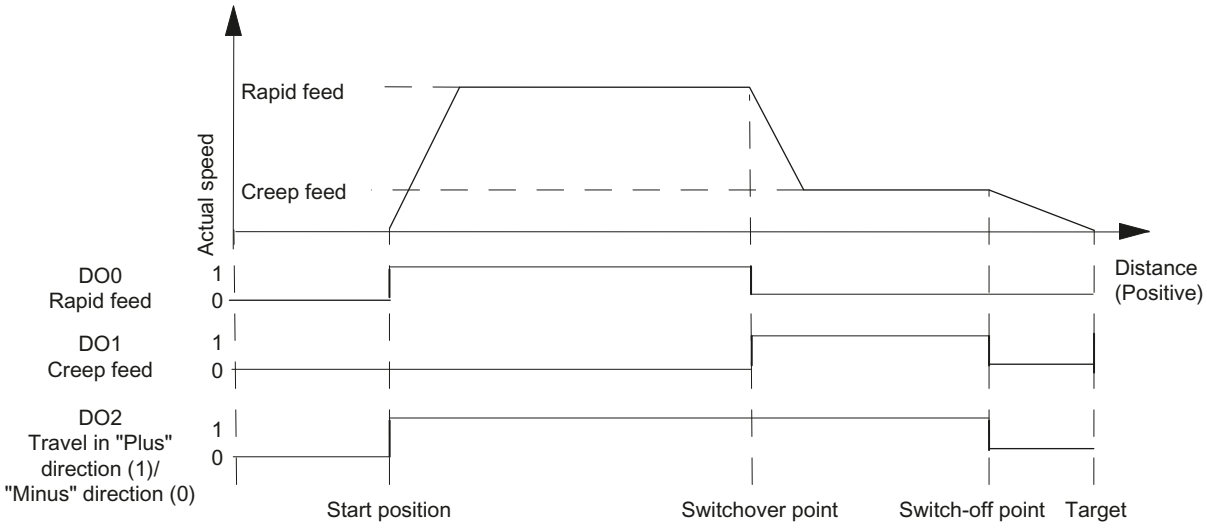

Effect of the Directional Enables (S7-300, S7-400)

Description

You enable the digital outputs directionally using the DIR_M and DIR_P control bits.

- With DIR_M = 1, you can traverse in a minus direction.
- With DIR_P = 1, you can traverse in a plus direction.

Interrupting and Continuing the Run

If you reset the relevant directional enable during a run, the motion of the axis is halted, all 3 digital outputs are set to 0, and the run is interrupted.

If you set the relevant directional enable again, the run is continued.

Stop (MODE 0) (S7-300, S7-400)

Stop

If you start MODE 0, then the 1PosU stops the current traverse, all 3 digital outputs are set to "0" and the traverse is terminated (POS_ERR = 0, POS_DONE = 1).

A traverse terminated with MODE 0 cannot be continued. To put the axis into motion again, you start a new MODE.

Control signals

The following table shows the data in the control interface which is relevant for "Stop".

Address	Assignment					
Byte 0	Bit 0.7 … 0.4:					
	Bit	7	6	5	4	
		0	0	0	0	MODE 0 = Stop
	Bit 0: START					

Checkback signals

The following table shows the data in the control interface (Page 2648) relevant for "Stop".

Address	Assignment
Byte 0	Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK

See also

1PosU control interface (Page 2646)

Inching (MODE 1) (S7-300, S7-400)

Inching

You use "Inching" to control the drive directly in a particular direction using the DIR_M or DIR_Pcontrol bits (Page 2591).

If you start MODE 1 then the 1PosU moves the drive with the specified velocity (SPEEDcontrol bit (Page 2646)) in the specified direction (DIR_M or DIR_Pcontrol bits (Page 2646)).

You stop the drive by setting the DIR_M = 0 and DIR_P = 0 control bits.

A change of direction (Page 2634) is executed after the time T_{min} elapses.

You can also activate jogging on an unsynchronized axis (feedback bit SYNC = 0) or when there is a pending encoder error (feedback bit ERR_ENCODER = 1) or without an encoder connected.

The figure below shows the sequence of the "Inching" motion type.

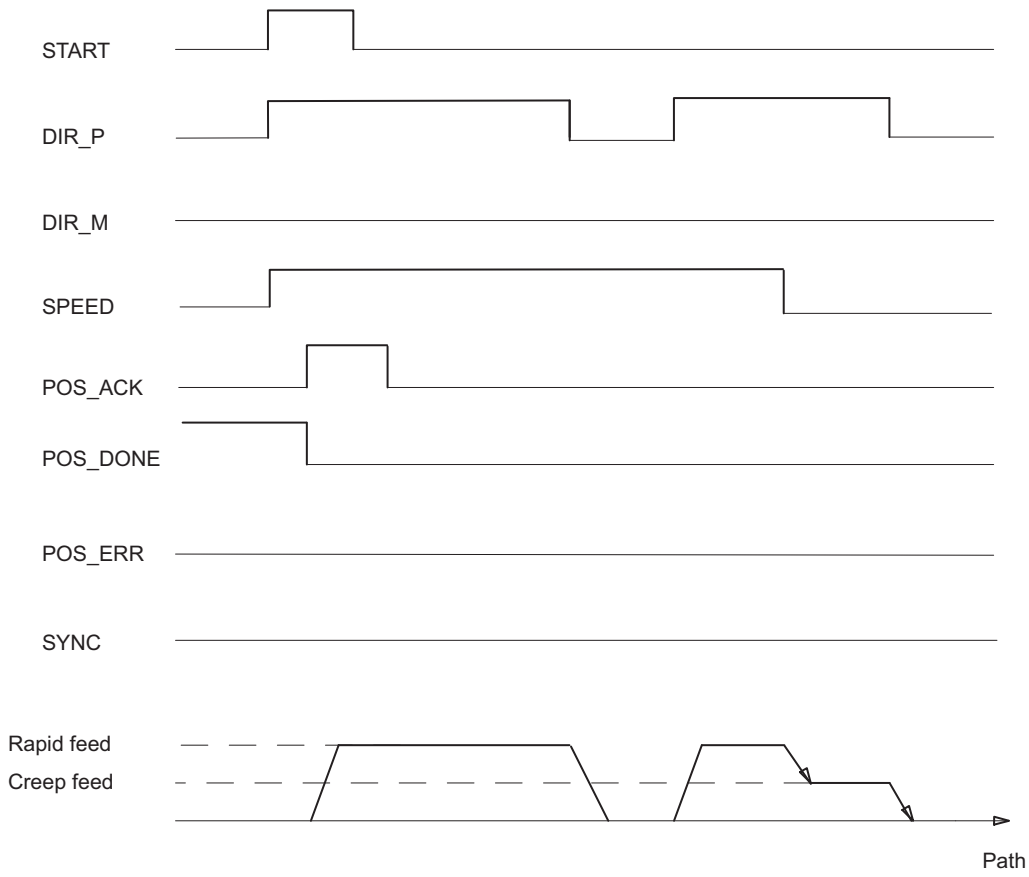

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Inching".

Address	Assignment					
Byte 0	Bit 0.7 … 0.4:					
	Bit	7	6	5	4	
		0	0	0	1	MODE 1 = Inching
	Bit 3: SPEED (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse) Bit 2: DIR_M Bit 1: DIR_P Bit 0: START					

Checkback signals

The following table shows the data in the control interface (Page 2648) which is relevant for "Inching".

Address	Assignment
Byte 0	Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK
Bytes 1 to 3	Actual value: • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1

Causes of the POS_ERR error

You must find out the causes of errors with JOB 15 (Display current values (Page 2618)).

The following table shows the error numbers relevant for "Inching" with the causes of the errors and possible remedies.

Error Number	Cause	Remedy
2	ERR_2L+ is displayed	Check the load voltage (2L+) at terminal 7.
5	The limit switch that lies in the direction in which the drive is moved is active	Check your switches and the wiring as well as the "DI0 limit switch minus (Page 2631)" and "DI1 limit switch plus (Page 2631)" parameters.
7	Inching: DIR_P = 1 and DIR_M = 1	
13	Direction of rotation of the drive and the encoder varies	Check the wiring of the drive and the encoder as well as the "Reversal of the direction of rotation (Page 2630)" parameter.
15	In dosing mode: DIR_M = 1	

Reference point approach (MODE 3) (S7-300, S7-400)

Reference point approach (MODE 3) (S7-300, S7-400)

Reference point approach

A reference point run can only be started at incremental encoders and non-activated dosing mode.

With reference point approach, you synchronies the axis by using external referencing signals. You can use either the 3 digital inputs or the zero mark (Page 2598) as a reference signal.

You can assign parameters to the digital inputs DI0 (minus limit switch), DI1 (plus limit switch) and DI2 (reducing cam) as NC or NO contacts.

Provide the control interface (Page 2646) with the reference point coordinates and start MODE 3. The 1PosU sets the SYNC = 0 checkback signal, moves the drive at the set velocity (SPEED control bit) in the programmed start direction and searches for the reference signal. The 1PosU automatically executes the required change of direction at the limit switches or the reducing cam.

Set the necessary directional enables (Page 2591) (DIR_M, DIR_P) to ensure that the drive is controlled.

When the 1PosU detects the assigned parameter reference signal, it then controls the drive at creep speed in the referencing direction. This is controlled by the "Reference signal" and "Reference switch" parameters.

	Reference switch: Reduction cam towards minus	Reference switch: Reduction cam towards plus	Reference switch: Limit switch minus	Reference switch: Limit switch plus
Reference signal: Reference switch and zero mark	Minus referencing direction	Plus referencing direction	Plus referencing direction	Minus referencing direction
Reference signal: Reference switch				
Reference signal: Zero mark	The referencing direction is not defined. The axis is synchronized at the next zero mark.			

After the reference signal has been traversed, the axis is synchronized. The 1PosU sets the checkback signal SYNC = 1 and assigns the reference point coordinates to the actual value.

The following figure shows the sequence of the "Inching" motion type.

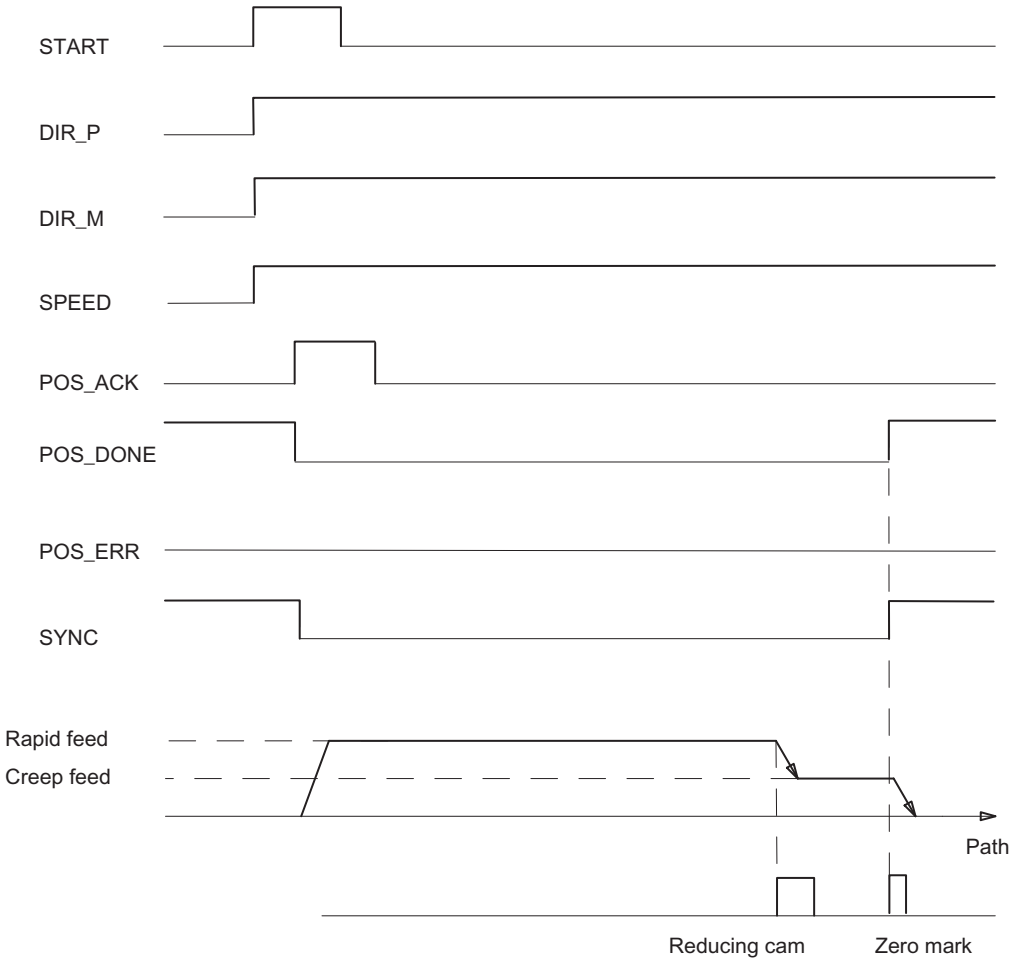

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Reference point approach".

Address	Assignment					
Byte 0	Bit 0.7 … 0.4:					
	Bit	7	6	5	4	
		0	0	1	1	MODE 3 = Reference point approach
	Bit 3: SPEED (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse) Bit 2: DIR_M Bit 1: DIR_P Bit 0: START					
Bytes 1 to 3	Reference point coordinates: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … end of the rotary axis – 1					

Checkback signals

The following table shows the data in the control interface (Page 2648) which is relevant for "Reference point approach".

Address	Assignment
Byte 0	Bit 3: SYNC Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK
Bytes 1 to 3	Actual value: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … end of the rotary axis – 1

Causes of the POS_ERR error

You must find out the causes of errors with JOB 15 (Display current values (Page 2618)).

The following table shows the error numbers relevant for "Reference point approach" with the causes of the errors and possible remedies.

Error number	Cause	Remedy
1	Impermissible MODE in dosing operation	
2	ERR_2L+ is displayed	Check the load voltage (2L+) at terminal 7.
3	ERR_ENCODER is displayed	Check the encoder wiring
10	Reference point approach: Reference point coordinate ≥ end of the rotary axis	
11	No reference signal found up to the limit switch or between the limit switches	Check your switches, the encoder and the wiring
13	Direction of rotation of the drive and the encoder varies	Check the wiring of the drive and the encoder as well as the "Reversal of the direction of rotation (Page 2630)" parameter.

Profile of a reference point approach (S7-300, S7-400)

Execution of a Reference Point Run Depending on Parameterization and Start Position

The profile of a reference point approach depends on the assigned parameters and the start position.

Therefore, for a reference point approach, there are different scenarios which are dependent on

- The start position of the drive at the start of the reference point run
- The assigned parameter start direction

- The assigned parameter reference signal
- The assigned parameter reference switch.

Example 1: Reference point approach with reduction cam and zero mark

Start position and parameters assigned:

- Start position: between the minus limit switch and the reducing cam
- Start direction: Plus
- Reference signal: Reference switch and zero mark
- Reference switch: Reduction cam towards plus

The following figure shows a reference point approach with reducing cams and zero mark.

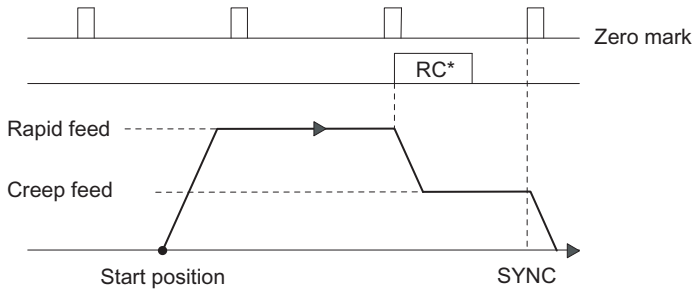

*RC = reducing cam

You can also carry out synchronization using the reducing cam without a zero mark.

If the start position is on the reducing cam, the 1PosU controls the drive directly at creep speed in the referencing direction.

Example 2: Reference Point Run with Minus Limit Switch

Start position and parameters assigned:

- Start position: between the minus limit switch and the plus limit switch
- Start direction: Minus
- Reference signal: Reference switch
- Reference switch: Limit switch minus

The following figure shows a reference point approach with minus limit switch.

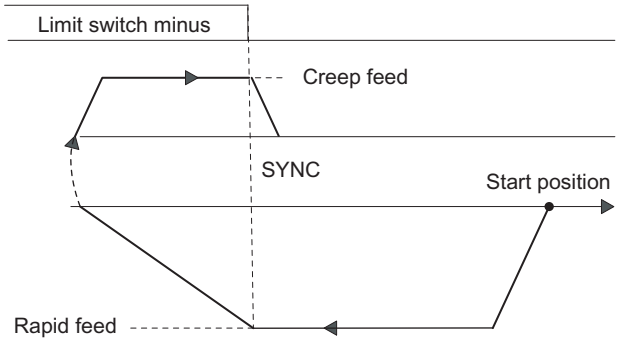

You can also carry out synchronization at the limit switch with the following zero mark.

If the start position is on the limit switch, the 1PosU controls the drive directly at creep speed in the referencing direction.

Example 3: Reference Point Run with Reversal of Direction at the Plus Limit Switch

Start position and parameters assigned:

- Start position: between the minus limit switch and the reducing cam
- Start direction: Plus
- Reference signal: Reference switch and zero mark
- Reference switch: Reduction cam towards plus

The following figure shows a reference point approach with a direction reversal at the plus limit switch.

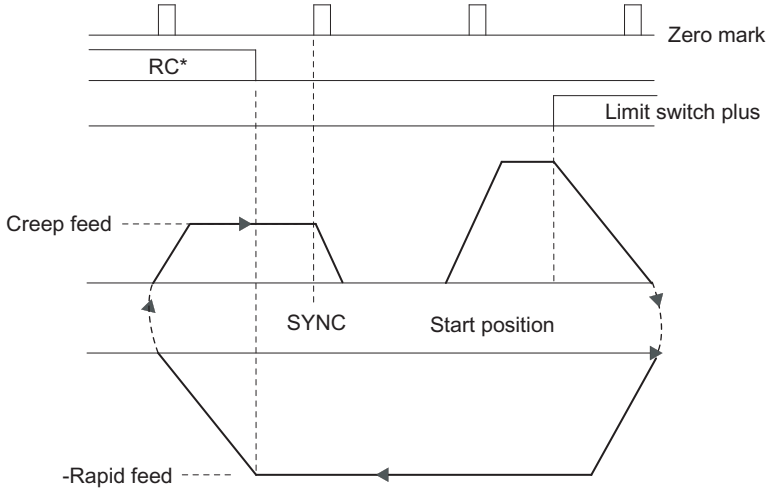

*RC = Reducing cam

If the start position is at the plus limit switch, the 1PosU controls the drive directly at rapid traverse in the opposite direction to the assigned parameter start direction.

Example 4: Reference Point Run Only with Zero Mark

Start position and parameters assigned:

- Start position: between the minus limit switch and the plus limit switch
- Start direction: Minus
- Reference signal: Zero mark
- Reference switch: irrelevant

The following figure shows a reference point approach just with zero mark.

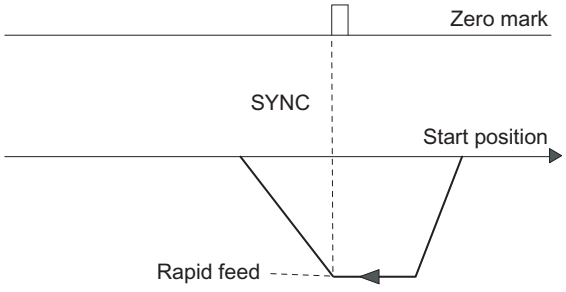

Relative positioning (MODE 4) (S7-300, S7-400)

Relative Positioning

In relative positioning the 1PosU moves the drive from the start position in a specified direction for a certain preset distance.

Provide the control interface (Page 2646) with the distance, and start MODE 4, specifying the direction (DIR_M and DIR_P). The 1PosU moves the drive at the preset velocity (SPEED control bit) for that distance. At the switch-over point the 1PosU switches from rapid traverse to creep speed, and at the switch-off point the 1PosU terminates the traverse.

If you start during an active traverse, the 1PosU executes the necessary change in direction (Page 2634) after the time T_{min} has elapsed.

The preset distance is not checked by the 1PosU. This means that more than one revolution may be involved with rotary axes.

In dosing mode (Page 2586), "Relative positioning" is only possible in the plus direction. The actual value is set to 0 at every start.

The figure below shows the sequence of the "Relative positioning" motion type.

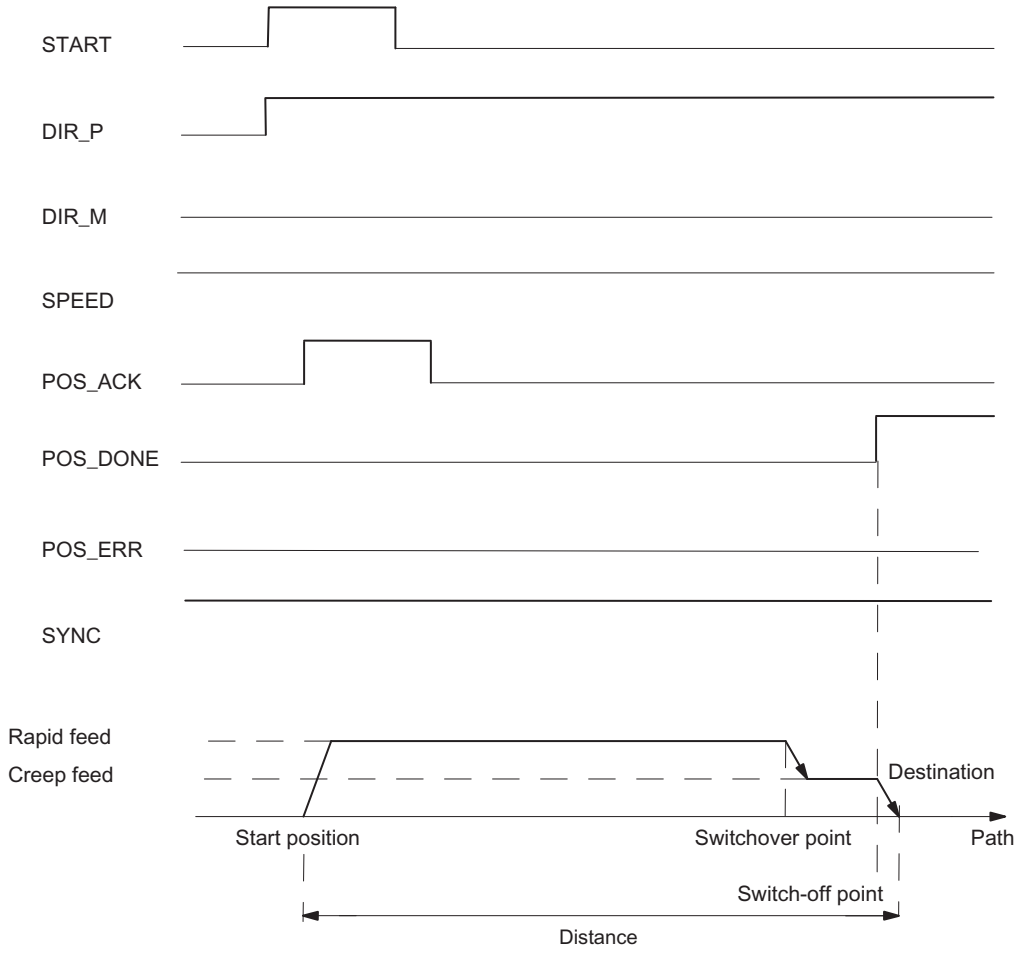

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Relative positioning".

Address	Assignment					
Byte 0	Bit 0.7 … 0.4:					
	Bit	7	6	5	4	
		0	1	0	0	MODE 4 = Relative positioning
	Bit 3: SPEED (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse) Bit 2: DIR_M Bit 1: DIR_P Bit 0: START					
Bytes 1 to 3	distance: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … 16 777 215					

Checkback signals

The following table shows the data in the feedback interface (Page 2648) which is relevant for "Relative positioning".

Address	Assignment
Byte 0	Bit 3: SYNC Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK
Bytes 1 to 3	Actual value: • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1

Causes of the POS_ERR error

You must find out the causes of errors with dem JOB 15 (Display current values (Page 2618)).

The following table shows the error numbers relevant for "Relative positioning" with the causes of the errors and possible remedies.

Error number	Cause	Remedy
2	ERR_2L+ is displayed	Check the load voltage (2L+) at terminal 7.
3	ERR_ENCODER is displayed	Check the encoder wiring
5	The limit switch that lies in the direction in which the drive is moved is active	Check your switches and the wiring as well as the "DI0 limit switch minus (Page 2631)" and "DI1 limit switch plus (Page 2631)" parameters.
7	Relative positioning: Start withDIR_P = 0 and DIR_M = 0 or DIR_P = 1 and DIR_M = 1	
13	Direction of rotation of the drive and the encoder varies	Check the wiring of the drive and the encoder as well as the "Reversal of the direction of rotation (Page 2630)" parameter.
15	In dosing mode: DIR_M = 1	

Absolute positioning (MODE 5) (S7-300, S7-400)

Absolute Positioning

With absolute positioning, the 1PosU moves the drive toward absolute targets. To do this, the axis must be synchronized.

"Absolute positioning" is not possible in activated dosing mode.

Provide the control interface (Page 2646) with the target, and start MODE 5 with the necessary directional enable (Page 2591) (DIR_M, DIR_P). The 1PosU moves the drive at the preset velocity (control bit SPEED) toward the target. At the switch-over point the 1PosU switches from rapid traverse to creep speed, and at the switch-off point the 1PosU terminates the traverse.

If you start during an active traverse, the 1PosU executes the necessary change in direction (Page 2634) after the time T_{min} has elapsed.

Linear axis

The 1PosU determines the direction the destination is to be approached from. You must set the necessary directional enable (DIR_M, DIR_P) to start. You can also set both enables.

Rotary axis

You determine the direction in which the target is approached by selecting the directional enable (DIR_M, DIR_P):

DIR_P and DIR_M control bits	Direction
DIR_P = 1 DIR_M = 0	The destination is approached in the plus direction.
DIR_P = 0 DIR_M = 1	The destination is approached in the minus direction.
DIR_P = 1 DIR_M = 1	The destination is approached by the shortest route. The 1PosU determines the direction the destination is to be approached from. If the resulting distance to be traversed is smaller than the switch-off difference, no traverse is started (POS_DONE = 1).

The figure below shows the sequence of the "Absolute positioning" motion type.

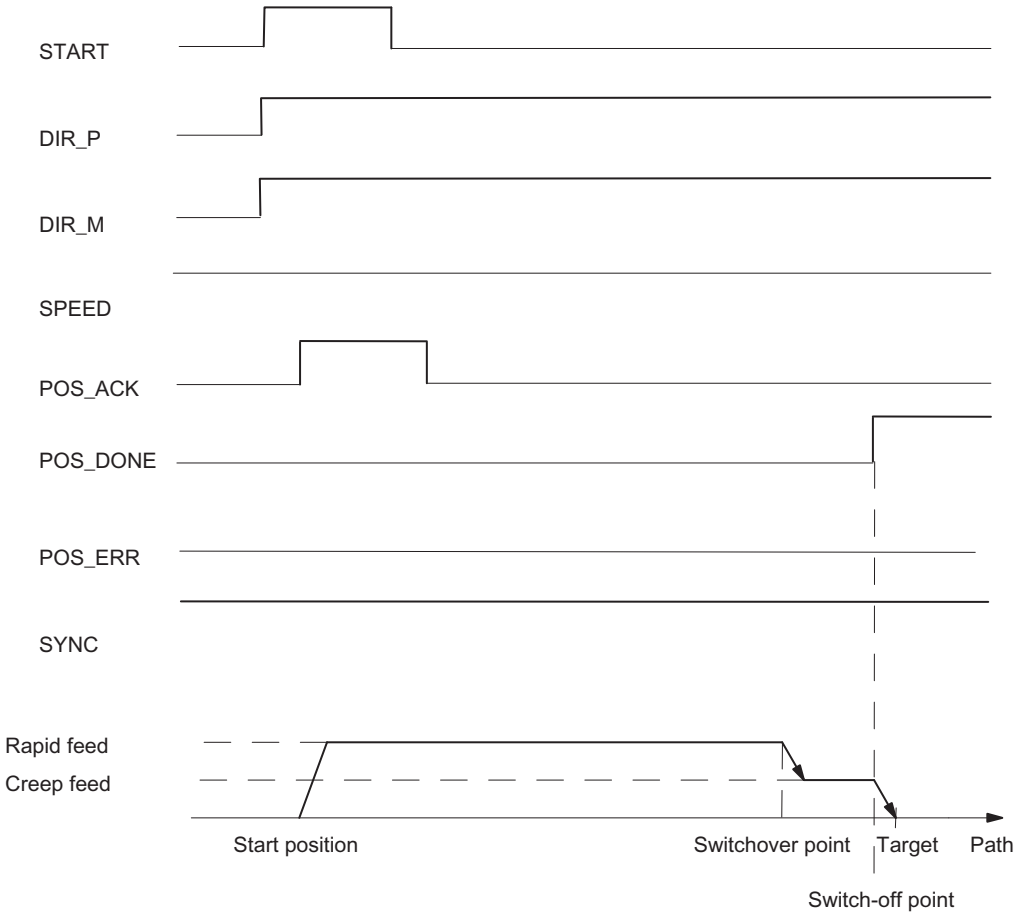

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Absolute positioning".

Address	Assignment					
Byte 0	Bit 0.7 … 0.4:					
	Bit	7	6	5	4	
		0	1	0	1	MODE 5 = Absolute positioning
	Bit 3: SPEED (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse) Bit 2: DIR_M Bit 1: DIR_P Bit 0: START					
Bytes 1 to 3	Target: • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1					

Checkback signals

The following table shows the data in the feedback interface (Page 2648) which is relevant for "Absolute positioning".

Address	Assignment
Byte 0	Bit 3: SYNC Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK
Bytes 1 to 3	Actual value: • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1

Causes of the POS_ERR error

You must find out the causes of errors with JOB 15 (Display current values (Page 2618)).

The following table shows the error numbers relevant for "Absolute positioning" with the causes of the errors and possible remedies.

Error number	Cause	Remedy
1	Impermissible MODE in dosing operation	
2	ERR_2L+ is displayed	Check the load voltage (2L+) at terminal 7.
3	ERR_ENCODER is displayed	Check the encoder wiring
4	Axis is not synchronized (SYNC = 0)	You can synchronize the axis with: • Reference point run • Reference Signal Evaluation • Setting Actual Value
5	The limit switch in the direction of which the drive is controlled is active	Check your switches and the wiring as well as the "DI0 limit switch minus (Page 2631)" and "DI1 limit switch plus (Page 2631)" parameters.
7	Start with DIR_P = 0 and DIR_M = 0 or relevant DIR_P = 0 or DIR_M = 0 control bit	
8	Target ≥ end of rotary axis (at incremental encoders) or target ≥ encoder range (at SSI encoders)	

Error number	Cause	Remedy
9	Absolute positioning was terminated because JOB 9 was initiated (only at incremental encoders)	
13	Direction of rotation of the drive and the encoder varies	Check the wiring of the drive and the encoder as well as the "Reversal of the direction of rotation (Page 2630)" parameter.

Canceling JOB Processing (JOB 0) (S7-300, S7-400)

Cancelling JOB processing

When you activate JOB 0, then the 1PosU reacts as follows:

- it cancels the current JOB 9 "Evaluate referencing signal",
- it cancels the current JOB 10 "Latch function",
- it sets a pending JOB_ERR = 0.

You can activate JOB 0 whatever the state of the axis.

Effect on the MODEs

MODEs are not affected by the JOB 0.

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Cancel JOB processing".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		0	0	0	0	JOB 0 = JOB-Cancel processing
	Bit 0: JOB_REQ					

Checkback signals

The following table shows the data in the feedback interface (Page 2648) which is relevant for "Cancel JOB processing".

Address	Assignment
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK

Setting the Actual Value (JOB 1) (S7-300, S7-400)

Setting actual value

"Set actual value" assigns new coordinates to the actual value displayed. This moves the operating range to another part of the axis.

The axis is synchronized at incremental encoders and non-activated dosing mode.

Provide the control interface (Page 2646) with the new actual value coordinates and activate JOB 1.

The 1PosU sets the preset actual value coordinates to the actual value displayed in the feedback interface (Page 2648) and sets the feedback bit SYNC = 1.

Effect on the MODEs

The following table shows the effects of JOB 1 on the various MODEs.

MODE	What happens?
Reference point approach	At incremental encoders and in non-activated dosing mode, ensure that, when the reference point run is evaluated, the SYNC = 1 feedback bit is set immediately. The reference point run still continues to run.
Inching	-
Absolute positioning	The following reactions are possible: • Distance to the target ≤ switch-off difference The switch-off point is reached or overshot; positioning is switched off immediately, and the run is terminated with POS_DONE = 1. In this case, the destination is sometimes overshot. • Distance to the destination ≤ the changeover difference The changeover point is reached or overshot; there is an immediate reduction from rapid feed to creep feed. In this case, the distance covered at creep speed is less than (switch-over difference - switch-off difference). • Distance to the destination > the changeover difference The drive is moved using rapid traverse, even if it was switched over to creep speed beforehand.
Relative Positioning	The preset distance continues to be traversed.

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Set actual value".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		0	0	0	1	JOB 1 = Setting actual value
	Bit 0: JOB_REQ					
Bytes 5 to 7	Actual value coordinates: • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1					

Checkback signals

The following table shows the data in the feedback interface (Page 2648) which is relevant for "Set actual value".

Address	Assignment
Byte 0	Bit 3: SYNC
Bytes 1 to 3	Actual value: • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Set actual value" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
23	ERR_ENCODER is displayed	Check the encoder wiring
34	Actual value coordinates ≥ end of rotary axis (at incremental encoders) or actual value coordinate ≥ encoder range (at SSI encoders)	

Shifting the encoder range (JOB 2) (S7-300, S7-400)

Move Encoder Range

The "Shift encoder range" function can only be executed at SSI encoders.

"Shift encoder range" adjusts the encoder value so that the actual value displayed corresponds to the real actual value. Before this can be done, any active run must be terminated.

Provide the control interface (Page 2646) with the offset and activate JOB 2.

You calculate the offset as follows:

- Offset = displayed actual value – real actual value

If the offset is negative, proceed as follows:

- Offset = displayed actual value - real actual value +
  (number of rotations × number of increments)

The 1PosU accepts the preset offset and displays the real actual value coordinates at the feedback interface (Page 2648).

Effect on the MODEs

MODEs are not affected by the JOB 2.

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Shift encoder range"

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		0	0	1	0	JOB 2 = Shift encoder range
	Bit 0: JOB_REQ					
Bytes 5 to 7	Offset (0 to encoder range)					

Checkback signals

The following table shows the data in the feedback interface (Page 2648) which is relevant for "Shift encoder range".

Address	Assignment
Bytes 1 to 3	Actual value: • 0 … encoder range – 1
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Shift encoder range" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
21	Impermissible JOB at the incremental encoder:	
23	ERR_ENCODER is displayed	Check the encoder wiring
26	JOB 2 "Shift encoder range" cannot be initiated because there is an active traverse	
33	with JOB 2: Offset not in encoder range	

Changing the switch-off difference (JOB 3) (S7-300, S7-400)

Change Switch-Off Difference

"Change switch-off difference" allows you to adjust the drive control to adapt to any changes in the load and mechanical conditions.

Provide the control interface (Page 2646) with the new switch-off difference and activate JOB 3.

The 1PosU accepts the specified switch-off difference.

The switch-off difference remains valid until the parameter assignment (Page 2627) of the 1PosU is changed.

Effect on the MODEs

The following table shows the effects of JOB 3 on the various MODEs.

MODE	What happens?
Reference point approach	-
Inching	
Absolute positioning	Distance to the target ≤ switch-off difference The switch-off point is reached or overshot; positioning is switched off immediately, and the run is terminated with POS_DONE = 1. In this case, the destination is sometimes overshot.
Relative Positioning	

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Change switch-off difference".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		0	0	1	1	JOB 3 = Change switch-off difference
	Bit 0: JOB_REQ					
Bytes 5 to 7	Switch-off difference • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … 16 777 215					

Checkback signals

The following table shows the data in the feedback interface (Page 2648) which is relevant for "Change switch-off difference".

Address	Assignment
Byte 4	Bit 0: JOB_ACK

Changing the changeover difference (JOB 4) (S7-300, S7-400)

Change Changeover Difference

"Change switch-over difference" allows you to adjust the drive control to adapt to any changes in the load and mechanical conditions.

Provide the control interface (Page 2646) with the new switch-over difference and activate JOB 4.

The 1PosU accepts the specified switch-over difference.

The switch-over difference remains valid until the parameter assignment (Page 2627) of the 1PosU is changed.

Effect on the MODEs

The following table shows the effects of JOB 4 on the various MODEs.

MODE	What happens?
Reference point approach	-
Inching	

MODE	What happens?
Absolute positioning	The following reactions are possible: • Distance to the destination ≤ the changeover difference. The changeover point is reached or overshot; there is an immediate reduction from rapid feed to creep feed. In this case, the distance covered at creep speed is less than (switch-over difference - switch-off difference). • Distance to the destination > the changeover difference The drive is moved using rapid feed, even if it was switched over to creep feed beforehand.
Relative Positioning	

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Change switch-over difference".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		0	1	0	0	JOB 4 = Change switch-over difference
	Bit 0: JOB_REQ					
Bytes 5 to 7	Switch-over difference: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … 16 777 215					

Checkback signals

The following table shows the data in the feedback interface (Page 2648) which is relevant for "Change changeover difference".

Address	Assignment
Byte 4	Bit 0: JOB_ACK

Evaluating the reference signal (JOB 9) (S7-300, S7-400)

Evaluating the reference signal

The "Evaluate reference signal" function is only available at incremental encoders and in non-active dosing mode.

"With "Evaluate reference signal" you can synchronize the axis using an external reference signal during a current traverse in "Jogging" and "Relative positioning" MODEs. You can use either the 3 digital inputs or the zero mark (Page 2597) as a reference signal.

You can assign parameters to the digital inputs DI0 (minus limit switch), DI1 (plus limit switch) and DI2 (reducing cam) as NC or NO contacts.

Provide the control interface (Page 2646) with the reference point coordinates and activate JOB 9. The 1PosU sets the SYNC = 0 checkback signal.

If the 1PosU detects the passing of the assigned parameter reference signal in the reference direction, the axis is synchronized. The 1PosU sets the checkback signal SYNC = 1 and assigns the reference point coordinates to the actual value.

The reference direction is controlled by the "Reference signal" and "Reference switch" parameters.

	Reference switch: Reduction cam towards minus	Reference switch: Reduction cam towards plus	Reference switch: Limit switch minus	Reference switch: Limit switch plus
Reference signal: Reference switch and zero mark	Minus referencing direction	Plus referencing direction	Plus referencing direction	Minus referencing direction
Reference signal: Reference switch				
Reference signal: Zero mark	The referencing direction is not defined. The axis is synchronized at the next zero mark.			

Effect on the MODEs

The following table shows the effects of JOB 9 on the various MODEs.

MODE	What happens?
Reference point approach	The reference coordinates transferred with JOB 9 are valid.
Inching	-
Absolute positioning	Traverse is cancelled with POS_ERR = 1, as SYNC us deleted
Relative positioning	-

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Evaluate reference signal".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		1	0	0	1	JOB 9 = Evaluating the reference signal
	Bit 0: JOB_REQ					
Bytes 5 to 7	Reference point coordinates: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … end of the rotary axis – 1					

Checkback signals

The following table shows the data in the control interface (Page 2648) which is relevant for "Evaluate reference signal".

Address	Assignment
Byte 0	Bit 3: SYNC
Bytes 1 to 3	Actual value: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … end of the rotary axis – 1
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Evaluate the reference signal" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
21	Impermissible JOB at SSI encoders or in dosing operation	
23	ERR_ENCODER is displayed	Check the encoder wiring
30	Reference point coordinate ≥ end of the rotary axis	

Latch Function (JOB 10) (S7-300, S7-400)

Latch Function

The "Latch function" allows you to store the actual value once at an edge at the DI2 digital input. You can use this function, for example, to detect edges or measure lengths.

Provide the control interface (Page 2646) with the desired edge and activate JOB 10.

If the 1PosU detects the preset edge at the DI2 digital input, it stores the associated actual value, displays it as a feedback value (Page 2648) and sets the LATCH_DONE = 1 feedback bit.

You can then activate the "Latch function" again.

Latch function and reference point approach or "Evaluate the reference signal"

If the 1PosU synchronizes at the same edge, it stores the actual value before it assigns the reference point coordinates.

Effect on the MODEs

MODEs are not affected by the JOB 10.

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Latch function".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		1	0	1	0	JOB 10 = Latch function
	Bit 0: JOB_REQ					
Byte 5	Bit 1: Latch at negative edge at DI2 Bit 0: Latch at positive edge at DI2					

Checkback signals

The following table shows the data in the feedback interface (Page 2648) which is relevant for "Latch function".

Address	Assignment
Byte 4	Bit 2: LATCH_DONE Bit 1: JOB_ERR Bit 0: JOB_ACK
Bytes 5 to 7	Feedback value: Actual value at the edge at the DI2 • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Latch function" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
23	ERR_ENCODER is displayed	Check the encoder wiring
36	Edge selection unknown	

Defining the settings for monitoring the direction of rotation (JOB 11) (S7-300, S7-400)

Setting the Monitoring of the Direction of Rotation

The "Define the settings for monitoring the direction of rotation" function is not available in dosing mode.

With "Define the settings for monitoring the direction of rotation" you can adjust the monitoring of the 1PosU's direction of rotation to suit your load and mechanical conditions.

Monitoring of the direction of rotation is always active. The 1PosU detects whether the direction of rotation of the drive and the encoder is the same. Direction of rotation monitoring will tolerate different directions for the drive and the encoder up to the preset path difference. If the specified distance is exceeded then the 1PosU reports POS_ERR = 1.

Unless you have activated JOB 11, double the switch-off difference is used from the parameters as the path difference. JOB 3 "Change switch-off difference" does not affect the distance difference for the purpose of monitoring the direction of rotation.

Provide the control interface (Page 2646) with the new distance difference and activate JOB 11.

The 1PosU accepts the preset distance difference for the monitoring of the direction of rotation.

The distance difference for the monitoring of the direction of rotation remains valid until the parameter assignment (Page 2627) of the 1PosU is changed.

Disabling the Monitoring of the Direction of Rotation

Monitoring of the direction of rotation is disabled when the path difference is 0.

Effect on the MODEs

MODEs are not affected by the JOB 11.

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Define the settings for monitoring the direction of rotation".

<table>
<tr><th>Address</th><th colspan="6">Assignment</th></tr>
<tr><td rowspan="3">Byte 4</td><td colspan="6">Bit 4.7 … 4.4:</td></tr>
<tr><td>Bit</td><td>7</td><td>6</td><td>5</td><td>4</td><td rowspan="2">JOB 11 = Define the settings for monitoring the direction of rotation</td></tr>
<tr><td></td><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td colspan="6">Bit 0: JOB_REQ</td></tr>
<tr><td>Byte 5</td><td colspan="6">0</td></tr>
<tr><td>Bytes 6, 7</td><td colspan="6">Distance difference for monitoring the direction of rotation
• 0 … 65 535</td></tr>
</table>

Checkback signals

The following table shows the data in the feedback interface (Page 2648) which is relevant for "Define the settings for monitoring the direction of rotation".

Address	Assignment
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Define the settings for monitoring the direction of rotation" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
21	Impermissible JOB in dosing mode	
38	Monitoring of the direction of rotation path difference > 65 535	

Displaying current values (JOB 15) (S7-300, S7-400)

Display Current Values

You can display the following values in the feedback interface as feedback values:

- Residual distance
- Actual speed
- Causes of errors POS_ERR and JOB_ERR

The residual distance is set by the 1PosU as the default for the feedback value.

The 1PosU continuously displays the actual value in the feedback interface irrespective of the selected feedback value.

Provide the control interface (Page 2646) with the desired feedback value and activate JOB 15.

The selected feedback value remains valid until the parameter assignment (Page 2627) of the 1PosU is changed.

"Display current values and latch function

If you activate the latch function, the 1PosU sets a feedback value of 0 and displays the actual value at the edge at the DI2 digital input.

You can only activate JOB 15 again after the latch function has terminated.

Residual distance

The 1PosU calculates the distance to the target as the residual distance in the "Absolute positioning" and "Relative positioning" MODEs. As long as the actual value is before the destination, the residual distance remains positive. It becomes negative when the destination is overshot. The residual distance is 0 in the other MODEs.

The 1PosU displays the residual distance with a sign between -8 388 608 and 8 388 607 increments. Negative values are displayed in twos complement. If the actual residual distance is beyond these limits, the limit value is displayed.

Actual speed

The 1PosU calculates the actual velocity as an encoder value change in increments per 10 ms. It displays these between 0 and 16 777 215.

Causes of errors POS_ERR and JOB_ERR

The 1PosU displays the causes of errors for POS_ERR and JOB_ERR as well as the MODE and JOB entered in the control interface (Page 2646).

Effect on the MODEs

MODEs are not affected by the JOB 15.

Control signals

The following table shows the data in the control interface (Page 2646) which is relevant for "Display current values".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		1	1	1	1	JOB 15 = Display current values
	Bit 0: JOB_REQ					
Byte 5	0: Residual distance 1: Actual speed 2: Causes of errors POS_ERR and JOB_ERR					

Checkback signals

The following table shows the data in the feedback interface (Page 2648) which is relevant for "Display current values".

Address	Assignment
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK
Bytes 5 to 7	In accordance with the selected feedback value: • With a residual distance of: –8 388 608 … 8 388 607 • With an actual speed of: 0 … 16 777 215 • with causes of POS_ERR and JOB_ERR errors: – Byte 5: Causes of (Page 2625)POS_ERR error – Byte 6: Causes of (Page 2626)JOB_ERR error – Bit 7.3 … 7.0: MODE (= bits 0.7 to 0.4 from the control signals) – Bit 7.7 … 7.4: JOB (= bits 4.7 to 4.4 from the control signals)

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Display current values" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
35	Display current values: Selection unknown	
37	Display current values: JOB 15 cannot be activated with the latch function running.	

Error detection / diagnostics (S7-300, S7-400)

Overview of errors with 1PosU (S7-300, S7-400)

Various errors can occur when working with the 1PosU. The following types of error are differentiated between:

- Parameter assignment error (Page 2621)
- External Errors (Page 2621)
- Errors in the Control of MODEs and JOBs (Page 2624)

You must acknowledge (Page 2624) the "Load voltage missing", "Short circuit of the encoder supply" and "Wire break/short circuit of the encoder signals" errors.

If errors occur when starting MODEs or activating JOBs, you can find more information here:

- Causes of POS_ERR error (Page 2625)
- Causes of JOB_ERR error (Page 2626)

Parameter assignment error (S7-300, S7-400)

Parameter assignment error

The following table shows 1PosU parameter assignment errors.

Parameter assignment error	1PosU reactions
Causes: • The 1PosU cannot identify existing parameters as its own. • The 1PosU slot you have configured does not match the structure. Only at SSI encoder: • Impermissible value for "Pulses per encoder revolution" parameter. • Impermissible value for "Revolutions" parameter. • Pulses per encoder revolution × revolutions is greater than 4096 × 4096. Only for dosing operation: • Activated "Encoder signal diagnostics" • Activated "Reversal of the direction of rotation" What to Do: • Check the configuration and structure.	• The 1PosU is not configured and cannot execute its functions. • Generating channel-specific diagnostics

External errors (S7-300, S7-400)

External errors

The following table shows 1PosU external errors.

Load Voltage 2L+ Missing	Reactions of the 1PosU
Causes: • Load voltage 2L+ not present or too low at terminal 7 Remedy: • Check the wiring. Eliminate a possible short-circuit. • Acknowledge (Page 2624) the error with the EXTF_ACK control bit.	• The current traverse is halted; it is not possible to start a new traverse. – All 3 digital outputs are set to 0. – POS_ERR = 1 feedback bit – POS_DONE = 0 feedback bit • ERR_2L+ = 1 feedback bit • generate channel-specific diagnostics • waits for EXTF_ACK error acknowledgment
Short circuit of the encoder supply	**Reactions of the 1PosU**

Causes: • Short circuit of the encoder supply made available at terminals 6 and 10 Remedy: • Check the wiring and correct the short circuit. • Acknowledge (Page 2624) the error with the EXTF_ACK control bit.	• The current "Reference point approach", "Relative positioning" and "Absolute positioning" MODEs are stopped; it is not possible to start a new traverse in these MODEs. – All 3 digital outputs are set to 0. – POS_ERR = 1 feedback bit – POS_DONE = 0 feedback bit • ERR_ENCODER = 1 feedback bit • SYNC = 0 feedback bit • generate channel-specific diagnostics • waits for EXTF_ACK error acknowledgment • Jogging MODE is not affected by this error. • The current "Evaluate referencing signal" JOB is cancelled.

Wire break/short circuit of the encoder signals	Reactions of the 1PosU
If an SSI encoder is used: Requirements: • To detect errors at the encoder signals, you must enable the "Encoder signal diagnostics" parameter. Causes: • Wire break or short circuit of the encoder signals at terminals 9 and 13 or 12 and 16. • The parameters for the SSI encoder do not correspond to the encoder connected. Remedy: • Check the wiring and correct the short circuit. • Compare the configuration with the technical specifications of the encoder. • Acknowledge (Page 2624) the error with the EXTF_ACK control bit. **The following applies when using an incremental encoder:** Requirement: • You must enable the "Encoder signal diagnostics" parameter for error detection at the A, /A and B, /B signals with 5V differential signal or at the A* and B* with 24V differential signal. • The "Zero marker diagnostics" parameter must be enabled in order to allow error recognition at the N, /N signals with 5V differential signal. If you use an encoder without a zero mark, switch off error detection. If dosing mode is activated, "Zero marker signal diagnostics" is not possible. Causes: • For 5V differential signal: Wire break or short circuit of the encoder signals at terminals 9 and 13 or 11 and 15 or 12 and 16. • Edge error of the encoder signals recognized so that 1PosU cannot carry out clear direction recognition. Remedy: • Check the wiring and correct the short circuit. • Acknowledge (Page 2624) the error with the EXTF_ACK control bit.	• The current "Reference point approach", "Relative positioning" and "Absolute positioning" MODEs are stopped; it is not possible to start a new traverse in these MODEs. – All 3 digital outputs are set to 0. – POS_ERR = 1 feedback bit – POS_DONE = 0 feedback bit • ERR_ENCODER = 1 feedback bit • SYNC = 0 feedback bit • generate channel-specific diagnostics • waits for EXTF_ACK error acknowledgment • Jogging MODE is not affected by this error. • The current "Evaluate referencing signal" JOB is cancelled.

Errors in the control of MODEs and JOBs (S7-300, S7-400)

Errors in the control of MODEs and JOBs

The following tables shows errors in the control of MODEs and JOBs with the 1PosU.

POS_ERR	Reactions of the 1PosU
Causes: • Certain requirements or conditions have not been met at the start of a MODEs	• The MODE started is not executed. • The current traverse is stopped. – All 3 digital outputs are set to 0. – POS_ERR = 1 feedback bit – POS_DONE = 0 feedback bit
JOB_ERR	**Reactions of the 1PosU**
Causes: • Certain requirements or conditions have not been met at the start of a JOBs	• The activated JOB is not executed. – JOB_ERR = 1 feedback bit

Error acknowledgment EXTF_ACK (S7-300, S7-400)

EXTF_ACK error acknowledgment

You must acknowledge (Page 2620) the "Load voltage missing", "Short circuit of the encoder supply" and "Wire break/short circuit of the encoder signals" corrected external errors. The following table shows the sequence necessary for this.

What You Do	1PosU reactions
	• ERR_2L+ = 1 feedback bit and/or • ERR_ENCODER = 1 feedback bit
Your control program detects the set ERR_2L+ or ERR_ENCODER feedback bit. Execute your application-specific error response. Eliminate the cause of the error.	
Switch the EXTF_ACKcontrol bit (Page 2646) from 0 to 1	The 1PosU sets the ERR_2L+ = 0 and ERR_ENCODER = 0feedback bits (Page 2648). This tells you that the cause has been eliminated and acknowledged. If ERR_2L+ = 1 and/or ERR_ENCODER = 1, the cause of the error has not yet been eliminated.
Switch the EXTF_ACKcontrol bit (Page 2646) from 1 to 0.	
In the case of constant error acknowledgment (EXTF_ACK = 1) or at CPU/Master Stop, the 1PosU reports the errors as soon as they are detected and deletes them as soon as they have been eliminated.	

Causes of POS_ERR error (S7-300, S7-400)

Causes of the POS_ERR error

The following table shows the causes and possible remedies for the POS_ERR error.

Error number	Cause	Remedy
1	MODE unknown or impermissible	Permissible MODEs are: • MODE 0 • MODE 1 • MODE 3 (not possible at SSI encoders or in dosing mode) • MODE 4 • MODE 5 (not possible in dosing mode)
2	ERR_2L+ is displayed	Check the load voltage (2L+) at terminal 7.
3	ERR_ENCODER is displayed	Check the encoder wiring
4	Axis is not synchronized (SYNC = 0)	SSI encoder: • Eliminate the encoder error. The axis can be synchronized at incremental encoders and non-activated dosing mode with: • Reference point approach • Evaluating the reference signal • Set actual value
5	The limit switch in the direction of which the drive is controlled is active	Check your switches and the wiring as well as the "DI0 limit switch minus (Page 2631)" and "DI1 limit switch plus (Page 2631)" parameters.
7	Inching: DIR_P = 1 and DIR_M = 1 Absolute positioning: Start with DIR_P = 0 and DIR_M = 0 or relevant DIR_P = 0 or DIR_M = 0 control bit Relative positioning: Start withDIR_P = 0 and DIR_M = 0 or DIR_P = 1 and DIR_M = 1	
8	Absolute positioning : Target ≥ end of rotary axis at incremental encoders or target ≥ encoder range at SSI encoders	
9	Absolute positioning was terminated because JOB 9 was initiated	
10	Reference point approach: Reference point coordinate ≥ end of the rotary axis	
11	Reference point approach: No reference signal found up to the limit switch or between the limit switches	Check your switches, the encoder and the wiring

Error number	Cause	Remedy
13	Direction of rotation of the drive and the encoder varies	Check the wiring of the drive and the encoder as well as the "Reversal of the direction of rotation (Page 2630)" parameter.
15	Jogging, relative positioning: In dosing mode DIR_M = 1	

Causes of JOB_ERR error (S7-300, S7-400)

Causes of the JOB_ERR error

The following table shows the causes and possible remedies for the POS_ERR error.

Error number	Meaning	Remedy
21	JOB unknown or impermissible	Permissible JOBs are: • JOB 0 • JOB 1 • JOB 2 (only possible at SSI encoders) • JOB 3 • JOB 4 • JOB 9 (not possible at SSI encoders or in dosing mode (Page 2586)) • JOB 10 • JOB 11 (not possible in dosing mode) • JOB 15
23	ERR_ENCODER is displayed	Check the encoder wiring
26	JOB 2 "Shift encoder range" cannot be initiated because there is an active traverse	
29	Evaluating the Reference Signal: Reference point coordinate ≥ end of the rotary axis	
34	Setting an Actual Value: Actual value coordinates ≥ encoder range	
35	Display current values: Selection unknown	
36	Latch Function: Edge selection unknown	
37	Display current values: JOB 15 cannot be activated with the latch function running.	
38	Monitoring of the direction of rotation path difference > 65 535	

CPU/Master STOP and RESET state (S7-300, S7-400)

Behavior in CPU/Master-STOP

The following table shows the behavior of 1PosU in CPU/Master-STOP.

Behavior in CPU/Master-STOP	1PosU reactions
• due to POWER-OFF of the CPU / bus master or • due to POWER-OFF of the IM 151 / IM 151 FO or • due to failure of bus transmission or • Due to change from RUN to STOP	• The current traverse is stopped. • All 3 digital outputs are set to 0. • POS_ERR = 0feedback bit (Page 2648) • POS_DONE = 1feedback bit (Page 2648)

Exiting the CPU/Master STOP status

The following table shows the behavior of 1PosU after exiting the CPU/Master-STOP status.

Exiting the CPU/Master STOP status	1PosU reactions
• with POWER-ON of the CPU / bus master or • with POWER-OFF of the IM 151 / IM 151 FO or • after failure of the bus transmission or • After a change from STOP to RUN	• The feedback interface (Page 2648) of the 1PosU remains current. • The axis remains synchronized, and the actual value is current. • The moved encoder range remains valid. • The changed switch-off and changeover differences and the path difference for the monitoring of the direction of rotation remain valid. • An initiated "Evaluate referencing signal (Page 2612)" JOB 9 and "Latch function (Page 2614)" JOB 10 remain active. • The feedback value (Page 2617) selected with JOB 15is current.

RESET state of the 1PosU

The following table shows the behavior of the 1PosU in RESET state.

RESET state of the 1PosU and changing the 1PosU parameters	1PosU reactions
• due to changing the parameters of the 1PosU and downloading the parameter assignment or configuration of the ET 200S station to the CPU / bus master or • due to POWER-ON at the power module of the 1PosU or • inserting the 1PosU in an energized state	• The axis is synchronized, and the actual value corresponds to the current encoder value. • The encoder range has not been moved. • The switch-off and changeover difference is accepted from the parameters. • The path difference for the monitoring of the direction of rotation is set at double the switch-off difference. • "Evaluate referencing signal (Page 2612)" JOB 9 and "Latch function (Page 2614)" JOB 10 are not active. • The residual distance is displayed as a feedback value (Page 2617).

6.2.7.4 Configuring and assigning parameters to 1PosU (S7-300, S7-400)

Operation (S7-300, S7-400)

Selecting the encoder interface (S7-300, S7-400)

Encoder interface

You define the 1PosU signal type you want to use by setting the encoder interface.

Encoder interface	Description
5V differential signal (Page 2629)	The 1PosU receives the signals from the incremental encoders with 5V differential signal.
24 V differential signal (Page 2634)	The 1PosU receives the signals from the incremental encoders with 24V signals.
SSI absolute (Page 2640)	The 1PosU receives the signals from the absolute encoders (SSI).

Parameters for incremental encoders with 5V differential signal (S7-300, S7-400)

Enables (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

You enable group diagnostics by ticking the corresponding options box.

When group diagnostics is enabled, an encoder error (ERR_ENCODER), no load voltage (ERR_2L+) or a parameter assignment error will result in a channel-specific diagnostics.

Enabling encoder signal diagnostics (S7-300, S7-400)

Encoder signal diagnostics

You enable encoder signal diagnostics by ticking the corresponding options box.

With "Encoder signal diagnostics" enabled, encoder signals A, /A and B, /B are monitored for short circuit and wire break. With non-enabled dosing mode, the signal sequence is also monitored. A simultaneous edge change at signals A and B causes an error.

Enable zero mark signal diagnostics (S7-300, S7-400)

Zero mark signal diagnostics

You enable zero mark signal diagnostics by ticking the corresponding options box.

With "Zero mark signal diagnostics" enabled, zero mark signals N, /N are monitored for short circuit and wire break.

Encoder and axes (S7-300, S7-400)

Enabling dosing mode (S7-300, S7-400)

Dosing mode

You enable dosing mode by ticking the corresponding options box.

In dosing mode (Page 2586), 1PosU only evaluates encoder signals A and /A. As a result, the distance can only be changed in the positive direction. The actual value is incremented at each positive edge.

Enabling reversal of direction of rotation (S7-300, S7-400)

Reversal of direction of rotation

You enable reversal of direction of rotation by ticking the corresponding options box.

You can use the "Reversal of direction of rotation" parameter to adapt the direction of rotation of the encoder to that of the drive and the axis.

In dosing mode (Page 2586), reversal of the direction of rotation is not possible.

Selecting axis type (S7-300, S7-400)

Axis types

The 1PosU positioning module can control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.

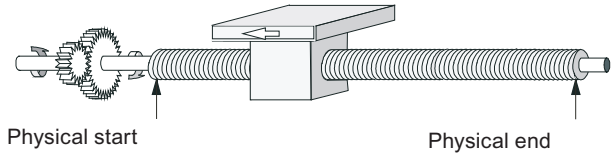

- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.

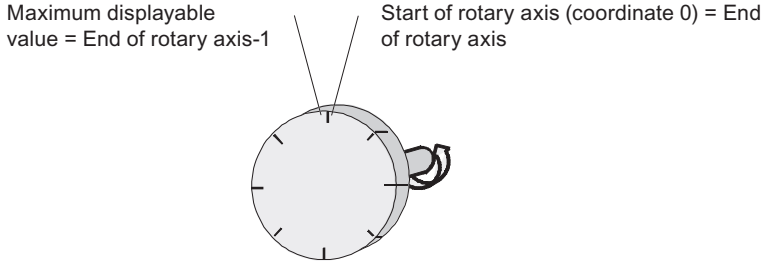

Specifying end of rotary axis (S7-300, S7-400)

End of rotary axis

This parameter is only relevant for the "Rotary axis" axis type.

The value "end of rotary axis" is the theoretical maximum value that the actual value can reach. The theoretically highest value is however never displayed, because it physically identifies the same position as the start of the rotary axis (0).

The displayed value jumps at

- high limit violated of (end of rotary axis – 1) to 0
- low limit violated of 0 to (end of rotary axis – 1)

Permitted value range for the end of the rotary axis: 1 … 16 777 215
Specifying "0" causes a configuration error.

Digital inputs (S7-300, S7-400)

Selecting minus limit switch and plus limit switch (S7-300, S7-400)

Function of the hardware limit switches

The two digital inputs (DI0 and DI1) are evaluated by the 1PosU as limit switches:

- DI0 is the minus limit switch and limits the operating range in the minus direction.
- DI1 is the plus limit switch and limits the operating range in the plus direction.

You can assign parameters to the hardware limit switches **separately as NC or NO contacts**.

The hardware limit switches are evaluated with linear axes and rotary axes.

Only the hardware limit switch that lies in the direction in which the drive is being moved is evaluated.

This enables you to move away from a hardware limit switch without additional error acknowledgment by moving in the other direction if you reach or overrun a hardware limit switch.

The current signal level of the digital inputs is displayed in the feedback interface, delayed by the update rate.

The following table shows how the hardware limit switches work in the individual MODEs.

MODE	Function of the hardware limit switches
Reference point approach	The 1PosU executes an automatic direction reversal at the hardware limit switch.
Inching	The motion of the axis is halted on the hardware limit switch, all 3 digital outputs are set to 0, and the POS_ERR feedback bit is reported.
Absolute Positioning	
Relative Positioning	

Starting on the hardware limit switch

The following table shows the reaction of the 1PosU when starting MODEs on the hardware limit switch.

Direction	1PosU reactions
Starting into the operating range	The 1PosU starts the specified MODE.
Starting away from the operating range	The POS_ERR = 1 feedback bit is set.

Selecting reduction cam (S7-300, S7-400)

Function of the reduction cam

The digital input DI2 is evaluated as a reduction cam or latch input by 1PosU:

- as a reduction cam with MODE 3 "Reference point approach (Page 2595)" and with JOB 9 "Evaluate referencing signal (Page 2612)"
- as a latch input with JOB 10 "Latch function (Page 2614)"

You can assign parameters to the hardware limit switches separately as **NC or NO contacts**.

Reference point approach and evaluation of the reference signal (S7-300, S7-400)

Selecting reference signal (S7-300, S7-400)

Reference signal

The "Reference signal" parameter defines the relevant switch or combination of switches and zero marks for MODE 3 "Reference point approach (Page 2595)" and JOB 9 "Evaluate reference signal (Page 2612)". The following are selectable:

- Reference switch and zero mark
- Reference switch
- Zero mark

You select the type of reference switch with the "Reference switch (Page 2632)" parameter.

Selecting reference switch (S7-300, S7-400)

Reference switch

The "Reference switch" parameter defines the relevant switch and the reference direction in which the switch must be overrun. The following are selectable:

- Reduction cam towards minus
- Reduction cam towards plus
- Limit switch minus
- Limit switch plus

These parameters are only relevant if you have selected it with "reference signal (Page 2631)"

- Reference switch and zero mark
  or
- Reference switch

Selecting start direction (S7-300, S7-400)

Start direction

The "Start direction" parameter defines the start direction for the MODE 3 "Reference point approach (Page 2595)". The following are selectable:

- Plus
- Minus

Drive (S7-300, S7-400)

Selecting control mode (S7-300, S7-400)

Control mode

The"Control mode" parameter defines the function of the digital outputs for the control of the drive (Page 2590). The following are selectable:

- Control mode 0:
  - DO0 travel minus
  - DO1 travel plus
  - DO2 rapid/creep feed
- Control mode 1:
  - DO0 rapid feed
  - DO1 creep feed (rapid feed is 0)
  - DO2 traverse plus (1)/ traverse minus (0)

Entering the switch-off difference (S7-300, S7-400)

Switch-off difference

Switch-off difference (Page 2584) defines the distance from the target at which the drive is slowed down from creep speed to 0.

If the switch-off difference ≥ the changeover difference, there is no changeover point. There is no deceleration from rapid traverse to creep speed; the response is executed directly at the switch-off point.

Valid value range:

- 0 to 65,535

You can use JOB 3 (Page 2610) to change the switch-off difference.

Entering the changeover difference (S7-300, S7-400)

Changeover difference

Changeover difference (Page 2584) defines the distance from the target at which the drive is slowed down from rapid traverse to creep speed.

Valid value range:

- 0 to 65,535

You can use JOB 4 (Page 2611) to change the changeover difference.

Entering the delay time for the direction change (S7-300, S7-400)

T_{min} direction change

The "T_{min} direction change" parameter defines the delay time for the direction change. The digital outputs are switched off, and a change of direction by T_{min} is executed with a delay. T_{min} is effective at each change of direction during a run.

T_{min} does not work when starting after POS_DONE = 1 or POS_ERR = 1.

Valid value range:

- 0 to 255

Your input value is multiplied by 10. You thus specify T_{min} in increments of 10 ms (for example, 0 ms, 10 ms or 2550 ms).

Parameters for incremental encoders with 24V signals (S7-300, S7-400)

Enables (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

You enable group diagnostics by ticking the corresponding options box.

When group diagnostics is enabled, an encoder error (ERR_ENCODER), no load voltage (ERR_2L+) or a parameter assignment error will result in a channel-specific diagnostics.

Enabling encoder signal diagnostics (S7-300, S7-400)

Encoder signal diagnostics

You enable encoder signal diagnostics by ticking the corresponding options box.

A simultaneous edge change of the signals A* and B* with "Encoder signal diagnostics" enabled leads to an error.

Encoder and axes (S7-300, S7-400)

Enabling dosing mode (S7-300, S7-400)

Dosing mode

You enable dosing mode by ticking the corresponding options box.

In dosing mode (Page 2586) the 1PosU only evaluates the encoder signal A*. As a result, the distance can only be changed in the positive direction. The actual value is incremented at each positive edge.

Selecting encoder inputs (S7-300, S7-400)

Encoder inputs

Different sensors can be connected to the encoder inputs:

- Sourcing output/push pull
- Sinking output

Note

If you have selected the "Sinking output" setting with for the "Encoder inputs" parameter, you must use the M-switching sensors.

Select the output circuit of the encoder in use for the signals A*, B* and N* from the drop-down list box.

Enabling reversal of direction of rotation (S7-300, S7-400)

Reversal of direction of rotation

You enable reversal of direction of rotation by ticking the corresponding options box.

You can use the "Reversal of direction of rotation" parameter to adapt the direction of rotation of the encoder to that of the drive and the axis.

In dosing mode (Page 2586), reversal of the direction of rotation is not possible.

Selecting axis type (S7-300, S7-400)

Axis types

The 1PosU positioning module can control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.

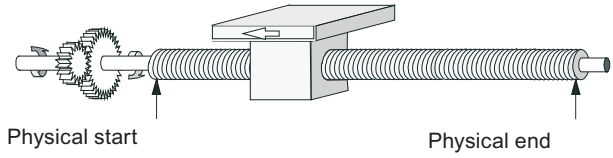

- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.

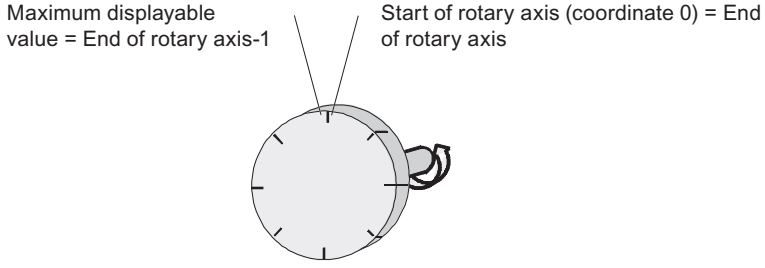

Specifying end of rotary axis (S7-300, S7-400)

End of rotary axis

This parameter is only relevant for the "Rotary axis" axis type.

The value "end of rotary axis" is the theoretical maximum value that the actual value can reach. The theoretically highest value is however never displayed, because it physically identifies the same position as the start of the rotary axis (0).

The displayed value jumps at

- high limit violated of (end of rotary axis – 1) to 0
- low limit violated of 0 to (end of rotary axis – 1)

Permitted value range for the end of the rotary axis: 1 … 16 777 215
Specifying "0" causes a configuration error.

Digital inputs (S7-300, S7-400)

Selecting minus limit switch and plus limit switch (S7-300, S7-400)

Function of the hardware limit switches

The two digital inputs (DI0 and DI1) are evaluated by the 1PosU as limit switches:

- DI0 is the minus limit switch and limits the operating range in the minus direction.
- DI1 is the plus limit switch and limits the operating range in the plus direction.

You can assign parameters to the hardware limit switches **separately as NC or NO contacts**.

The hardware limit switches are evaluated with linear axes and rotary axes.

Only the hardware limit switch that lies in the direction in which the drive is being moved is evaluated.

This enables you to move away from a hardware limit switch without additional error acknowledgment by moving in the other direction if you reach or overrun a hardware limit switch.

The current signal level of the digital inputs is displayed in the feedback interface, delayed by the update rate.

The following table shows how the hardware limit switches work in the individual MODEs.

MODE	Function of the hardware limit switches
Reference point approach	The 1PosU executes an automatic direction reversal at the hardware limit switch.
Inching	The motion of the axis is halted on the hardware limit switch, all 3 digital outputs are set to 0, and the POS_ERR feedback bit is reported.
Absolute Positioning	
Relative Positioning	

Starting on the hardware limit switch

The following table shows the reaction of the 1PosU when starting MODEs on the hardware limit switch.

Direction	1PosU reactions
Starting into the operating range	The 1PosU starts the specified MODE.
Starting away from the operating range	The POS_ERR = 1 feedback bit is set.

Selecting reduction cam (S7-300, S7-400)

Function of the reduction cam

The digital input DI2 is evaluated as a reduction cam or latch input by 1PosU:

- as a reduction cam with MODE 3 "Reference point approach (Page 2595)" and with JOB 9 "Evaluate referencing signal (Page 2612)"
- as a latch input with JOB 10 "Latch function (Page 2614)"

You can assign parameters to the hardware limit switches separately as **NC or NO contacts**.

Reference point approach and evaluation of the reference signal (S7-300, S7-400)

Selecting reference signal (S7-300, S7-400)

Reference signal

The "Reference signal" parameter defines the relevant switch or combination of switches and zero marks for MODE 3 "Reference point approach (Page 2595)" and JOB 9 "Evaluate reference signal (Page 2612)". The following are selectable:

- Reference switch and zero mark
- Reference switch
- Zero mark

You select the type of reference switch with the "Reference switch (Page 2638)" parameter.

Selecting reference switch (S7-300, S7-400)

Reference switch

The "Reference switch" parameter defines the relevant switch and the reference direction in which the switch must be overrun. The following are selectable:

- Reduction cam towards minus
- Reduction cam towards plus
- Limit switch minus
- Limit switch plus

These parameters are only relevant if you have selected it with "reference signal (Page 2637)"

- Reference switch and zero mark
  or
- Reference switch

Selecting start direction (S7-300, S7-400)

Start direction

The "Start direction" parameter defines the start direction for the MODE 3 "Reference point approach (Page 2595)". The following are selectable:

- Plus
- Minus

Drive (S7-300, S7-400)

Selecting control mode (S7-300, S7-400)

Control mode

The"Control mode" parameter defines the function of the digital outputs for the control of the drive (Page 2590). The following are selectable:

- Control mode 0:
  - DO0 travel minus
  - DO1 travel plus
  - DO2 rapid/creep feed
- Control mode 1:
  - DO0 rapid feed
  - DO1 creep feed (rapid feed is 0)
  - DO2 traverse plus (1)/ traverse minus (0)

Entering the switch-off difference (S7-300, S7-400)

Switch-off difference

Switch-off difference (Page 2584) defines the distance from the target at which the drive is slowed down from creep speed to 0.

If the switch-off difference ≥ the changeover difference, there is no changeover point. There is no deceleration from rapid traverse to creep speed; the response is executed directly at the switch-off point.

Valid value range:

- 0 to 65,535

You can use JOB 3 (Page 2610) to change the switch-off difference.

Entering the changeover difference (S7-300, S7-400)

Changeover difference

Changeover difference (Page 2584) defines the distance from the target at which the drive is slowed down from rapid traverse to creep speed.

Valid value range:

- 0 to 65,535

You can use JOB 4 (Page 2611) to change the changeover difference.

Entering the delay time for the direction change (S7-300, S7-400)

T_{min} direction change

The "T_{min} direction change" parameter defines the delay time for the direction change. The digital outputs are switched off, and a change of direction by T_{min} is executed with a delay. T_{min} is effective at each change of direction during a run.

T_{min} does not work when starting after POS_DONE = 1 or POS_ERR = 1.

Valid value range:

- 0 to 255

Your input value is multiplied by 10. You thus specify T_{min} in increments of 10 ms (for example, 0 ms, 10 ms or 2550 ms).

Parameters for SSI encoders (S7-300, S7-400)

Enables (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

You enable group diagnostics by ticking the corresponding options box.

When group diagnostics is enabled, an encoder error (ERR_ENCODER), no load voltage (ERR_2L+) or a parameter assignment error will result in a channel-specific diagnostics.

Enabling encoder signal diagnostics (S7-300, S7-400)

Encoder signal diagnostics

You enable encoder signal diagnostics by ticking the corresponding options box.

With "Encoder signal diagnostics" enabled, encoder signals D, /D are monitored for short circuit and wire break. Monitoring of the frame is carried out additionally (start bit and stop bit).

Encoder and axes (S7-300, S7-400)

Selecting encoder type (S7-300, S7-400)

Encoder type

Select the encoder in use from the drop-down list:

- Single-turn encoder (SSI 13 bit)
- Multi-turn encoder (SSI 25 bit)

Selecting the transmission rate (S7-300, S7-400)

Transmission rate

With the "transmission rate" parameter, you define the data transmission rate between the SSI encoder and the positioning module.

The maximum transmission rate is dependent on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 60 m → 500 kHz
- 20 m → 1 MHz

Selecting increments per encoder revolution (S7-300, S7-400)

Increments per encoder revolution

The "Increments per encoder revolution" parameter defines the number of increments an absolute encoder returns per revolution.

For the limits you have to differentiate between the individual encoder models. Only values to the power of 2 are selectable.

- Single-turn encoder (number of revolutions = 1) with 13-bit frame length:
  - minimum value = 4
  - maximum value = 8192
- Multiturn encoder (number of revolutions > 1) with 25-bit message frame length:
  - Minimum value = 4
  - Maximum value = 8192

Selecting the number of encoder revolutions (S7-300, S7-400)

Number of encoder revolutions

This parameter is used to define the maximum number of revolutions of this encoder.

The entry is only relevant for multi-turn encoders. For single-turn encoders, the 1PosU sets the number of revolutions to 1.

Valid value range:

- 2 to 4096 in powers of 2

Enabling reversal of direction of rotation (S7-300, S7-400)

Reversal of direction of rotation

You enable reversal of direction of rotation by ticking the corresponding options box.

You can use the "Reversal of direction of rotation" parameter to adapt the direction of rotation of the encoder to that of the drive and the axis.

Selecting axis type (S7-300, S7-400)

Axis types

The 1PosU positioning module can control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.

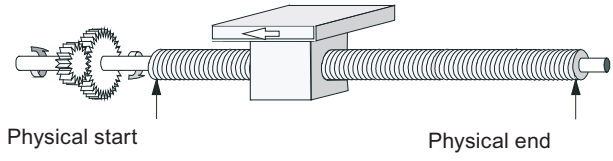

- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.

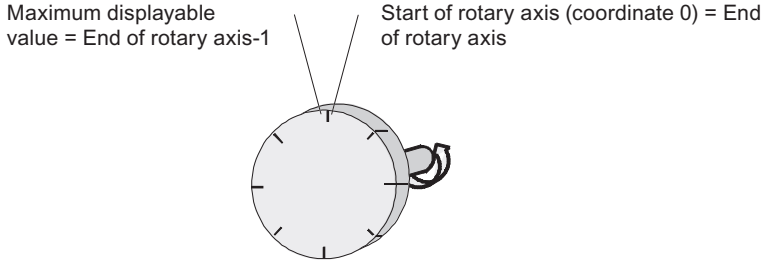

Digital inputs (S7-300, S7-400)

Selecting minus limit switch and plus limit switch (S7-300, S7-400)

Function of the hardware limit switches

The two digital inputs (DI0 and DI1) are evaluated by the 1PosU as limit switches:

- DI0 is the minus limit switch and limits the operating range in the minus direction.
- DI1 is the plus limit switch and limits the operating range in the plus direction.

You can assign parameters to the hardware limit switches **separately as NC or NO contacts**.

The hardware limit switches are evaluated with linear axes and rotary axes.

Only the hardware limit switch that lies in the direction in which the drive is being moved is evaluated.

This enables you to move away from a hardware limit switch without additional error acknowledgment by moving in the other direction if you reach or overrun a hardware limit switch.

The current signal level of the digital inputs is displayed in the feedback interface, delayed by the update rate.

The following table shows how the hardware limit switches work in the individual MODEs.

MODE	Function of the hardware limit switches
Reference point approach	The 1PosU executes an automatic direction reversal at the hardware limit switch.
Inching	The motion of the axis is halted on the hardware limit switch, all 3 digital outputs are set to 0, and the POS_ERR feedback bit is reported.
Absolute Positioning	
Relative Positioning	

Starting on the hardware limit switch

The following table shows the reaction of the 1PosU when starting MODEs on the hardware limit switch.

Direction	1PosU reactions
Starting into the operating range	The 1PosU starts the specified MODE.
Starting away from the operating range	The POS_ERR = 1 feedback bit is set.

Selecting reduction cam (S7-300, S7-400)

Function of the reduction cam

The digital input DI2 is evaluated as a reduction cam or latch input by 1PosU:

- as a reduction cam with MODE 3 "Reference point approach (Page 2595)" and with JOB 9 "Evaluate referencing signal (Page 2612)"
- as a latch input with JOB 10 "Latch function (Page 2614)"

You can assign parameters to the hardware limit switches separately as **NC or NO contacts**.

Drive (S7-300, S7-400)

Selecting control mode (S7-300, S7-400)

Control mode

The"Control mode" parameter defines the function of the digital outputs for the control of the drive (Page 2590). The following are selectable:

- Control mode 0:
  - DO0 travel minus
  - DO1 travel plus
  - DO2 rapid/creep feed
- Control mode 1:
  - DO0 rapid feed
  - DO1 creep feed (rapid feed is 0)
  - DO2 traverse plus (1)/ traverse minus (0)

Entering the switch-off difference (S7-300, S7-400)

Switch-off difference

Switch-off difference (Page 2584) defines the distance from the target at which the drive is slowed down from creep speed to 0.

If the switch-off difference ≥ the changeover difference, there is no changeover point. There is no deceleration from rapid traverse to creep speed; the response is executed directly at the switch-off point.

Valid value range:

- 0 to 65,535

You can use JOB 3 (Page 2610) to change the switch-off difference.

Entering the changeover difference (S7-300, S7-400)

Changeover difference

Changeover difference (Page 2584) defines the distance from the target at which the drive is slowed down from rapid traverse to creep speed.

Valid value range:

- 0 to 65,535

You can use JOB 4 (Page 2611) to change the changeover difference.

Entering the delay time for the direction change (S7-300, S7-400)

T_{min} direction change

The "T_{min} direction change" parameter defines the delay time for the direction change. The digital outputs are switched off, and a change of direction by T_{min} is executed with a delay. T_{min} is effective at each change of direction during a run.

T_{min} does not work when starting after POS_DONE = 1 or POS_ERR = 1.

Valid value range:

- 0 to 255

Your input value is multiplied by 10. You thus specify T_{min} in increments of 10 ms (for example, 0 ms, 10 ms or 2550 ms).

1PosU control and feedback interface (S7-300, S7-400)

1PosU control interface (S7-300, S7-400)

Assignment of the Control Interface

The table below shows the assignment of the 1PosU control interface (outputs).

Address	Assignment					
Byte 0	Bits 0.7 to 0.4 stand for the MODEs					
	Bit	7	6	5	4	
		0	0	0	0	MODE 0 = Stop
		0	0	0	1	MODE 1 = Inching
		0	0	1	1	MODE 3 = Reference point approach
		0	1	0	0	MODE 4 = Relative positioning
		0	1	0	1	MODE 5 = Absolute positioning
	Bit 3: SPEED (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse) Bit 2: DIR_M Bit 1: DIR_P Bit 0: START					
Bytes 1 to 3	with MODE 3 = Reference point approach: reference point coordinates					
	with MODE 4 = Relative positioning: distance					
	with MODE 5 = Absolute positioning: target					

<table>
<tr><th>Address</th><th>Assignment</th></tr>
<tr><td>Byte 4</td><td>Bits 4.7 to 4.4 stand for the JOBs
<table>
<tr><td>Bit</td><td>7</td><td>6</td><td>5</td><td>4</td><td></td></tr>
<tr><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>JOB 0 = Cancel JOB processing</td></tr>
<tr><td></td><td>0</td><td>0</td><td>0</td><td>1</td><td>JOB 1 = Set actual value</td></tr>
<tr><td></td><td>0</td><td>0</td><td>1</td><td>0</td><td>JOB 2 = Shift encoder range (only at SSI encoders)</td></tr>
<tr><td></td><td>0</td><td>0</td><td>1</td><td>1</td><td>JOB 3 = Change switch-off difference</td></tr>
<tr><td></td><td>0</td><td>1</td><td>0</td><td>0</td><td>JOB 4 = Change changeover difference</td></tr>
<tr><td></td><td>1</td><td>0</td><td>0</td><td>1</td><td>JOB 9 = Evaluate the reference signal</td></tr>
<tr><td></td><td>1</td><td>0</td><td>1</td><td>0</td><td>JOB 10 = Latch function</td></tr>
<tr><td></td><td>1</td><td>0</td><td>1</td><td>1</td><td>JOB 11 = Define the settings for monitoring the direction of rotation</td></tr>
<tr><td></td><td>1</td><td>1</td><td>1</td><td>1</td><td>JOB 15 = Display current values</td></tr>
</table>
Bit 3: EXTF_ACK
Bit 2: Reserved = 0
Bit 1: Reserved = 0
Bit 0: JOB_REQ</td></tr>
<tr><td>Bytes 5 to 7</td><td>In accordance with the selected JOBs:
• with JOB 1 = Actual value coordinates:
• with JOB 2 = Offset:
• with JOB 3 = Switch-off difference
• with JOB 4 = Changeover difference
• withJOB 9 = Reference point coordinates
• with JOB 10
– Byte 5: Bit 0 = latch at positive edge at DI2
– Byte 5: Bit 1 = latch at negative edge at DI2
• with JOB 11 = Distance difference for monitoring of the direction of rotation
• with JOB 15
– Byte 5: 0 = Residual distance
– Byte 5: 1 = Actual speed
– Byte 5: 2 = Error information</td></tr>
</table>

1PosU feedback interface (S7-300, S7-400)

Assignment of the Feedback Interface

The following table shows the assignment of the 1PosU feedback interface (inputs).

Address	Assignment
Byte 0	Bit 7: ERR_ENCODER Bit 6: STATUS DO 2 Bit 5: STATUS DO 1 Bit 4: STATUS DO 0 Bit 3: SYNC Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK
Bytes 1 to 3	Actual value
Byte 4	Bit 7: ERR_2L+ Bit 6: STATUS DI 2 Reduction cam Bit 5: STATUS DI 1 Limit switch plus Bit 4: STATUS DI 0 Limit switch minus Bit 3: Reserved = 0 Bit 2: LATCH_DONE Bit 1: JOB_ERR Bit 0: JOB_ACK
Bytes 5 to 7	Feedback value

6.2.7.5 Diagnostics of 1PosU (S7-300, S7-400)

Diagnostics using the LED display (S7-300, S7-400)

LED display on the 1PosU

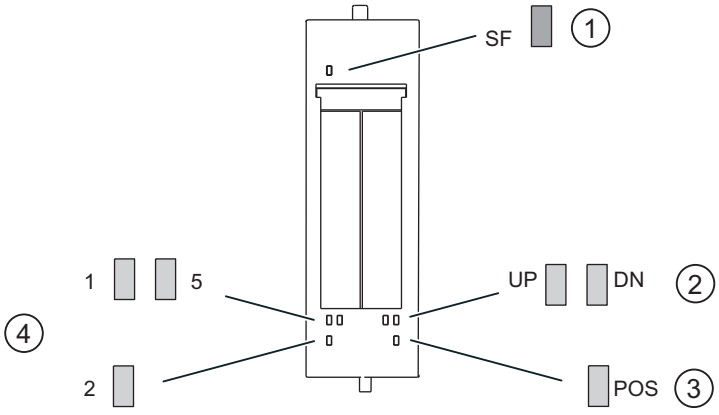

① Group error (red)
② Status display for a change in an actual value (green)
③ Positioning underway (green)
④ Status displays for digital inputs (green)

Status and error displays by means of LEDs on the 1PosU

The table below shows the status and error displays on the 1PosU.

Event (LED)							Cause	Remedy
SF	1	5	2	UP	DN	POS		
On							No parameter assignment. Diagnostic message.	Check the parameter settings. Analyze the diagnostic data.
	On						DI 0 is activated.	
		On					DI 1 is activated.	
			On				DI 2 is activated.	
				On			In the case of actual value change from lower to higher values	
					On		In the case of actual value change from higher to lower values	
						On	Positioning is running and one of the 3 digital outputs is set.	

Error types (S7-300, S7-400)

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

1PosU error types

The following table shows the error types on the 1PosU.

Error type		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the sensor supply.	Check the wiring to the sensor. Correct the process wiring.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.
17_D	10001: Load voltage 2L + missing	Power supply voltage not present or too low.	Correct the process wiring. Check the supply voltage.
26_D	11010: External error	Wire break/short circuit of the sensor signals. Wire break in the sensor cable, or sensor cable is not connected. Sensor is defective or there are faults. Sensor type, transmission rate, and monoflop time do not correspond to the sensor connected; programmable sensors do not correspond to the settings on the module.	Correct the process wiring. Correct the parameter assignment. Replace the sensor.

6.2.8 Using ET 200S 1STEP 5V (S7-300, S7-400)

6.2.8.1 Terminal assignment of the 1STEP 5V (S7-300, S7-400)

Wiring rules

The cables (terminals 1 and 5 and terminals 4 and 8) to the power unit must be shielded, twisted-pair cables. The shield must be supported at both ends. To do this, use the shield contact (see the appendix of the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal Assignment

The following table shows the terminal assignment of 1STEP 5V.

View	Terminal Assignment	Remarks
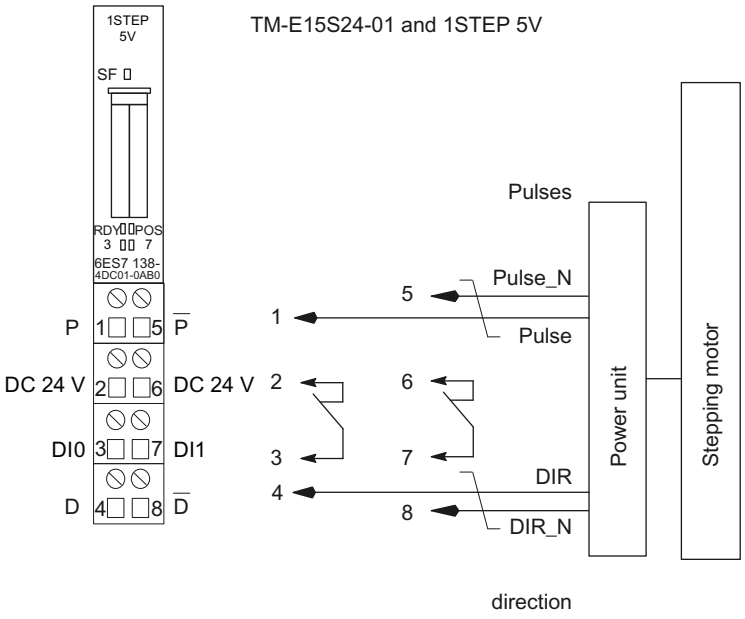		24 V DC: Load voltage DI0: external enable pulse, external STOP or limit switch in forward/reverse direction DI1: Reference switch or reference switch and limit switch in forward/reverse direction P, /P and D, /D are signals to RS 422.
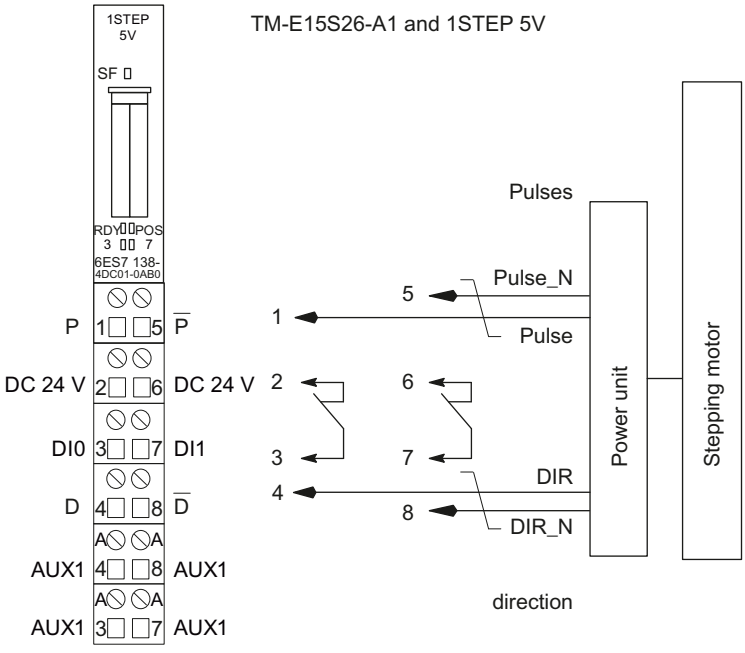		24 V DC: Load voltage DI0: external enable pulse, external STOP or limit switch in forward/reverse direction DI1: Reference switch or reference switch and limit switch in forward/reverse direction P, /P and D, /D are signals to RS 422.

6.2.8.2 Fundamentals of positioning with stepper motor (S7-300, S7-400)

Interaction of components (S7-300, S7-400)

Below is a description of how individual components for positioning with stepping motor affect each other.

1STEP 5V

The 1STEP 5V generates pulses and a directional signal for the power units of stepping motors. The number of pulses emitted determines the distance traversed. The pulse frequency determines the velocity. The change in the pulse frequency per time unit (second) is a measure for the acceleration or deceleration. 1STEP 5V is influenced by its parameters and settings.

Power unit for stepping motors

The power unit is the link between the 1STEP 5V and the stepping motor. The 1STEP 5V sends RS 422 differential signals for frequency and direction. These signals are converted in the power unit into motor currents that control the movements of the motor with a very high degree of precision.

Stepping motors

Stepping motors are used to position axes. They represent the simple and cost-effective solution for precision positioning jobs in wide performance ranges.

A stepping motor shaft turns by a certain angle with every pulse. During rapid pulse sequences, this stepping movement becomes a continuous turning motion.

Parameters and Settings (S7-300, S7-400)

Required Information

To ensure optimum interplay between the individual components, you must provide the 1STEP 5V with information:

- **One time: during parameter configuration using your configuration software**
  - Base frequency (Page 2656) F_b
  - Multiplier n (Page 2654) for setting the start-stop frequency F_{ss}
  - Multiplier i (Page 2654) for setting the acceleration / delay
  - Function and behavior of the digital inputs (Page 2675)
  - Traversing range (Page 2671)
- **In operation: movement of the motor by means of a traversing job in your user program**
  - Multiplier G (Page 2654) for the velocity/output frequency F_a
  - Reduction factor R (Page 2656)
  - Distance, position or frequency
  - Operating mode
  - Direction specification (traversing job) to the start
- **In operation: to adjust to different load conditions as a parameter assignment request in your user program**
  - Base frequency (Page 2656) F_b
  - Multiplier n (Page 2654) for setting the start-stop frequency F_{ss}
  - Multiplier i (Page 2654) for setting the acceleration / delay

Traversal curve of the 1STEP 5V (S7-300, S7-400)

Traversal curve

An incremental mode is carried out by the 1STEP 5V in accordance with the following traversal curve. The 1STEP 5V forms the fundamental parameters (start-stop frequency, output frequency, and acceleration/delay) of the traversal curve with a base frequency (Page 2656) that you select.

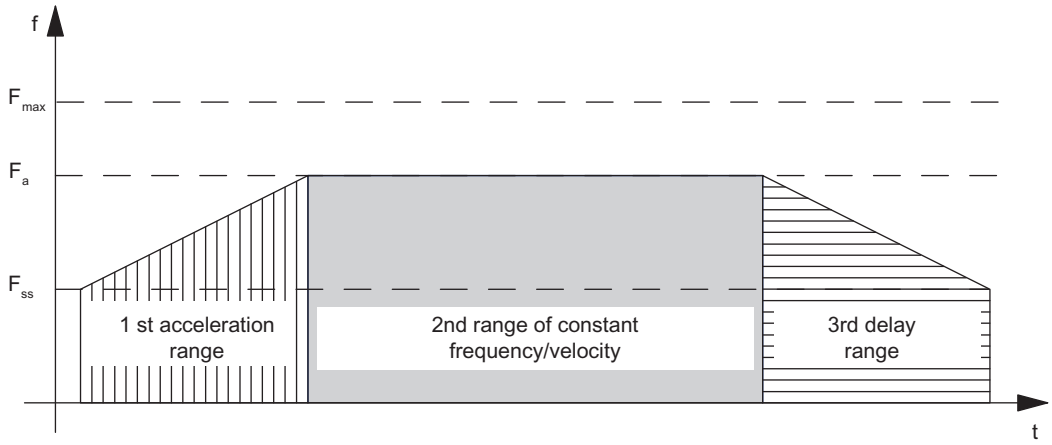

F_{ss} = start-stop frequency
F_a = output frequency
F_{max} = maximum frequency of the stepping motor

Start-Stop Frequency F_{ss}

The start-stop frequency F_{ss} is the frequency to which the motor can be accelerated under load from a standstill.

The size of F_{ss} depends on the load inertia. The best way to work out the load inertia is by trial and error.

The start-stop frequency F_{ss} is simultaneously the minimum output frequency F_a needed to move the stepping motor.

Setting the Start-Stop Frequency F_{ss}

Through parameter assignment, the 1STEP 5V permits the start-stop frequency F_{ss} to be set in steps. To do so, select the multiplier n between 1 and 255, which is multiplied with the base frequency F_b. You can lower the start-stop frequency F_{ss} again with the reduction factor R (1 or 0.1) in the traversing job.

The start-stop frequency is calculated according to equation:

$F_{ss} = F_b \times n \times R$

Additional information is available in the following table (Page 2656) with the ranges for start-stop frequency, starting frequency and acceleration.

Maximum frequency / velocity of the Axis F_{max}

When you choose a stepping motor, remember the following:

The maximum frequency/velocity is determined by your application. At this frequency, the motor must reach a torque high enough to move its load.

Note this does not mean the highest possible frequency that the motor or the power unit can tolerate.

You can work out the maximum frequency F_{max} with the corresponding characteristic curve. The following figure shows the torque characteristic curve of a stepping motor.

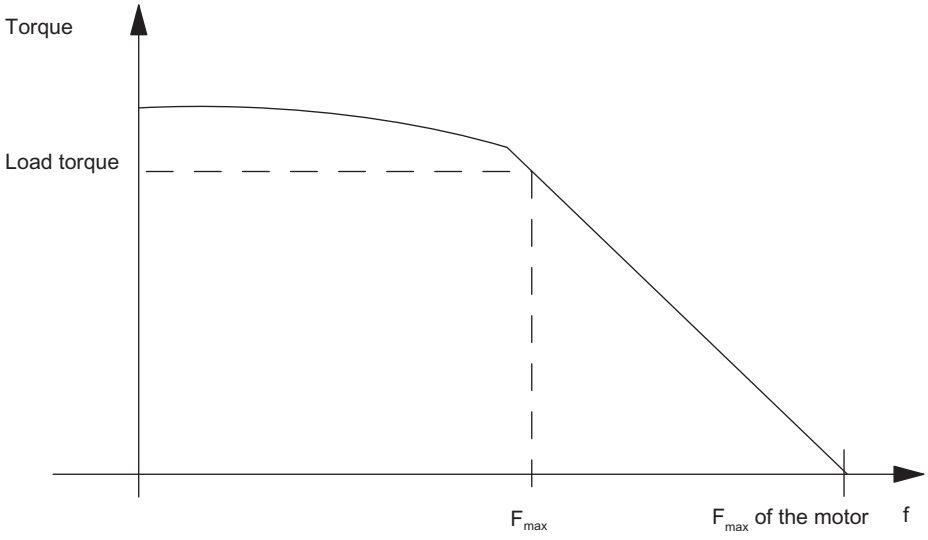

Output frequency / velocity F_a

The output frequency can be set differently for each run.

When selecting the output frequency, also observe the minimum pulse duration of your power unit (see Formula and table (Page 2656) with the ranges for start-stop frequency, output frequency and acceleration).

If the selected output frequency is lower than the set start-stop frequency F_{ss}, the 1STEP 5V output frequency is set to the start-stop frequency F_{ss}.

F_a must always be smaller than F_{max}.

Setting of the output frequency / Velocity F_a

The 1STEP 5V permits the output frequency F_a to be set in steps. To do so, select the multiplier G between 1 and 255, which is multiplied with the base frequency F_b. You can lower the output frequency F_a again with the reduction factor R (1 or 0.1) in the traversing job.

The output frequency is calculated according to the equation:

$F_a = F_b \times G \times R$

Additional information is available in the following table (Page 2656) with the ranges for start-stop frequency, starting frequency and acceleration.

Acceleration / delay a

The maximum permitted acceleration / delay depends on the load to be moved.

The motor must reach a torque high enough to accelerate or delay the load without loss of step.

Depending on the application, you must also take into account additional criteria for setting the acceleration/delay, such as smooth starting and stopping.

Setting the acceleration / delay a

Through parameter assignment, the 1STEP 5V permits the acceleration / delay to be set in steps by means of the multiplier i.

During the acceleration phase, the frequency is increased continuously starting from the start-stop frequency F_{ss} until the output frequency F_a has been reached.

The time interval for the continuous increase in frequency can be set in steps. To do so you select a multiplier i between 1 and 255.

In the delay phase, the output frequency is reduced in the same way.

You can lower the acceleration / delay a again with the reduction factor R (1 or 0.1) in the traversing job.

The acceleration / deceleration is calculated according to the equation:

$a = F_b \times R / (i \times 0.128\ ms)$

Additional information is available in the following table (Page 2656) with the ranges for start-stop frequency, starting frequency and acceleration.

Setting the Base Frequency (S7-300, S7-400)

Introduction

Through parameter assignment, the 1STEP 5V permits the base frequency to be set in steps.

The base frequency sets the range for the start-stop frequency, the output frequency, and the acceleration.

Procedure

1. Depending on the priority of your requirements select a suitable range either of the start-stop frequency F_{ss} and of the starting frequency F_a or of the acceleration a from the following table in accordance with the following criteria:
   - Range for the start-stop frequency F_{ss},
     for example, for starting and stopping as soon as possible
   - Range for the output frequency F_a,
     for example, for a velocity setting that is as precise as possible
   - Range for the acceleration a,
     for example, for the fastest possible positioning operations
2. Use the table to determine the base frequency F_b.
   To optimize the base frequency F_b, proceed as follows:
3. Check whether the other corresponding values meet your requirements. If necessary, select another base frequency F_b, which meets your requirements better.
4. Define the multipliers required to set the output frequency F_a, the acceleration/delay a, and the start-stop frequency F_{ss}.
5. Determine the corresponding reduction factor R from the table.

The following table shows the ranges for the start-stop frequency, output frequency, and acceleration.

Base frequency F_b: in Hz	Reduction factor R	Range Start-stop frequency F_{ss} Starting frequency F_a in Hz	Range Acceleration a in Hz/ms
		Equation: $F_{ss} = F_b \times n \times R$ $F_a = F_b \times G \times R$	Equation: $a = F_b \times R / (i \times 0.128\ ms)$
4	0,1	0,4 … 102	0,01 … 3,13
8	0,1	0,8 … 204	0,02 … 6,25
20	0,1	2 … 510	0,06 … 15,6
4	1	4 … 1020	0,12 … 31,3
8	1	8 … 2040	0,25 … 62,5
20	1	20 … 5100	0,61 … 156
40	1	40 … 10200	1,23 … 313
80	1	80 … 20400	2,45 … 625
200	1	200 … 51000	6,13 … 1563
400	1	400 … 102000	12,25 … 3125
800	1	800 … 204000	24,51 … 6250

Base frequency F_b: in Hz	Reduction factor R	Range Start-stop frequency F_{ss} Starting frequency F_a in Hz	Range Acceleration a in Hz/ms
		Equation: $F_{ss} = F_b \times n \times R$ $F_a = F_b \times G \times R$	Equation: $a = F_b \times R / (i \times 0.128\ ms)$
2000 [1]	1	2000 … 510000	61,27 … 15625
F_b = Base frequency F_{ss} = Start-Stop frequency F_a = Output frequency a = Acceleration / delay R = Reduction factor n = Multiplier for setting the start-stop frequency in steps G = Multiplier for setting the output frequency in steps i = Multiplier for setting the acceleration / delay in steps			

[1] as of 6ES7138-4DD01-0AB0

The minimum pulse duration results from the set starting frequency F_a and is calculated using the equation $T_{Pulse} = 1 / (2 \times F_a)$.

6.2.8.3 Functions of 1STEP 5V (S7-300, S7-400)

Overview of functions (S7-300, S7-400)

Overview

The job of the 1STEP 5V is to position a drive on certain predefined targets (incremental modes) and to travel continuously with specifiable frequencies (speed-control mode).

The following functions are available to you to this purpose:

- Reference point approach (Page 2660): The axis is synchronized to a reference point
- Set reference point (Page 2665) (as of 6ES7138-4DC01-0AB0): A value is assigned to the current position.
- Relative incremental approach (Page 2666) (relative positioning): The axis is moved by a predefined distance.
- Absolute incremental approach (Page 2667) (absolute positioning) (as of 6ES7138-4DC01-0AB0): The axis is traveled to a predefined position.
- Speed mode (Page 2668) (as of 6ES7138-4DC01-0AB0): The drive is moved with a speed that can be specified flexibly (pulse frequency).
- Hold traversing job (Page 2669)
- Changing Parameters during Operation (Page 2674)

The description of the control interface (Page 2683) provides information about how to control the functions.

Starting a traversing job (S7-300, S7-400)

Starting the traversing job

The figure below shows the process from start request for a traversing job up to the end of the traversing.

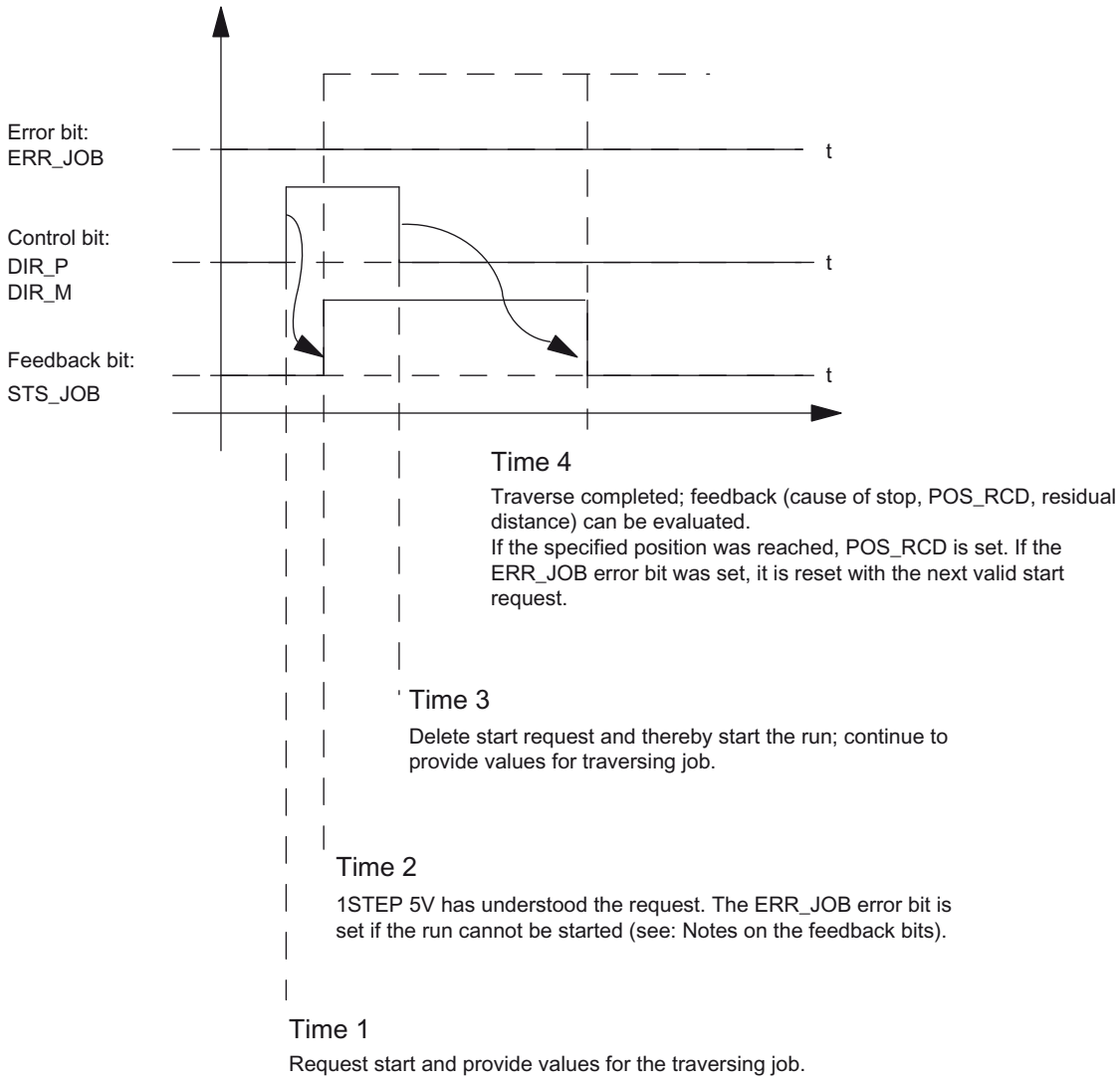

Evaluating the ERR_JOB error bit

As soon as the feedback bit (Page 2687)STS_JOB is cleared at time stamp 4, evaluate the ERR_JOB error bit. Note that the feedback bit STS_JOB is only cleared if the control bits (Page 2683)DIR_P, DIR_M and C_PAR are cleared.

Search for Reference (S7-300, S7-400)

Description of the function

The home position marks the point of reference of your drive system (reference cam) for the following traversing jobs. You can determine the home position by, for example, installing an initiator on the reference cam and connecting it to the DI1 digital input.

The 1STEP 5V ensures the reference point can be reproduced accurately in that it is always approached from the same direction. You can specify this direction by always starting the search for reference in the same direction.

Traversing job for reference point approach

The traversing job contains the following information:

- Multiplier G for the velocity/output frequency F_a
- Reduction factor R for the assigned parameters base frequency F_b
- Reference point position
- Mode = 1 for reference point approach
- Stop at the reference cam (Page 2669)
- Direction specification as start (Page 2683)

Note

The 1STEP 5V checks the set position for limits (minimum 0 and maximum 16777215). The full-scale value can be configured.

Note

If you have configured the behavior of the digital input (Page 2675) DI1 (7) as a "Reference switch and limit switch", the 1STEP 5V automatically selects the starting direction toward the limit switch, irrespective of the direction specified in the traversing job.

Note

Status of the limit switch

When you use limit switches it is possible that these may not supply the correct status at the start of the reference point approach. Such cases can occur, for example, if the limit switches are configured as normally closed contacts (active at 0-level) and the power supply of the limit switches was not yet available before the start of the reference point approach.

This can lead to the reference point approach starting in the wrong direction.

Make sure that each of the limit switches returns the correct status before you start an approach.

Status bit SYNC

The status bit (Page 2687) SYNC informs you that the axis has been synchronized, that is, after the correct homing procedure, this status bit is set and deleted during the run.

The SYNC status bit is deleted

- After parameter assignment of your ET 200S station
- After deletion of the pulse enable
- After a CPU/Master STOP

In these cases it is advisable to carry out a search for reference.

POS and POS_RCD status bits

While reference point approach is active, it is indicated by the set feedback bit (Page 2687) POS.

On completion of a reference point approach, the set POS_RCD feedback bit indicates that the position has been reached.

If the reference point approach is interrupted, the POS_RCD feedback bit remains reset.

Residual distance, position, frequency

The residual distance reported (Page 2687) is irrelevant during the reference point approach.

Note

In order for the 1STEP 5V to approach the home position with repeated precision, the period duration of the start-stop frequency has to be greater than the runtime of a single step from the 1STEP 5V to the stepping motor and via the reference cam back to 1STEP 5V (input delay of digital inputs: 4 ms).

When stopping at the reference cam or at one of the limit switches during the acceleration phase, the 1STEP 5V continues to send pulses for a maximum of 50 ms at the frequency already reached before it starts braking. This avoids abrupt changes in frequency, which can lead to step losses.

Sequence of Execution of the Search for Reference (S7-300, S7-400)

Steps of the Search for Reference

A search for reference consists of a maximum of three sections.

In the **first section** (1) and **second section** (2), the system ensures that the reference cam is found.

Both these sections are traversed at the defined output frequency F_a.

In the **third section** (3), the reference cam is approached with start-stop frequency F_{ss} in the selected direction up to the reference point ⊕ with reproducible accuracy.

Note

The maximum number of output pulses of a section is the configured length of the traversing range minus 1.

Different Execution Sequences

Depending on the position (P) at the start of the reference point approach, there are different execution patterns for the run (REF is the reference cam, which is wired to the DI1 digital input). The illustration applies to the forward starting direction (DIR_P).

- **Start before REF or at limit switch LIMIT_M**

The figure below shows a reference point approach with start before the REF.

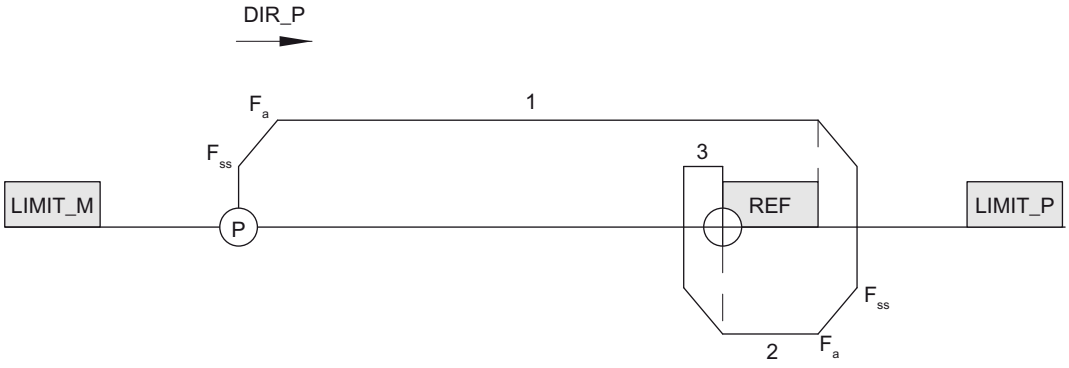

- **Start after the REF**

The figure below shows a reference point approach with start after the REF.

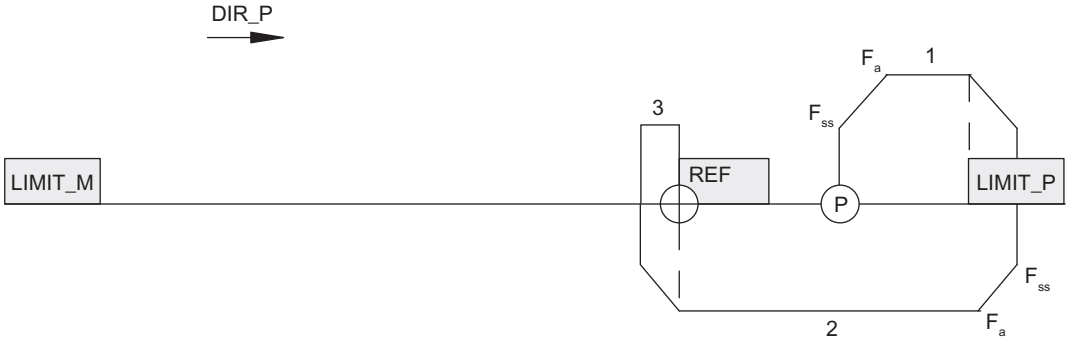

- **Start on the REF**

The figure below shows a reference point approach with start on the REF.

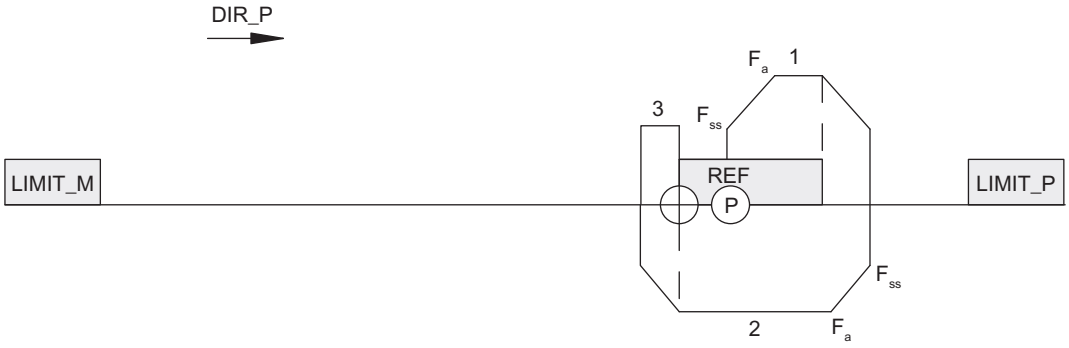

- **Start at the limit switch in start direction**

The figure below shows a reference point approach with start on the limit switch in start direction.

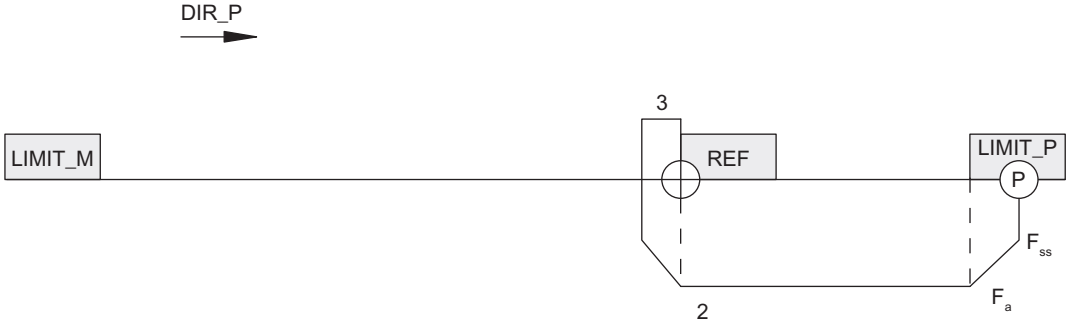

Reference switch is simultaneously a limit switch

Note

When the reference switch REF is simultaneously a limit switch, the 1STEP 5V automatically chooses the starting direction toward the limit switch, regardless of the direction specified in the traversing job.

- **Start within the traversing range**

The figure below shows a reference point approach with start within the traversing range, in which REF is simultaneously the limit switch LIMIT_M.

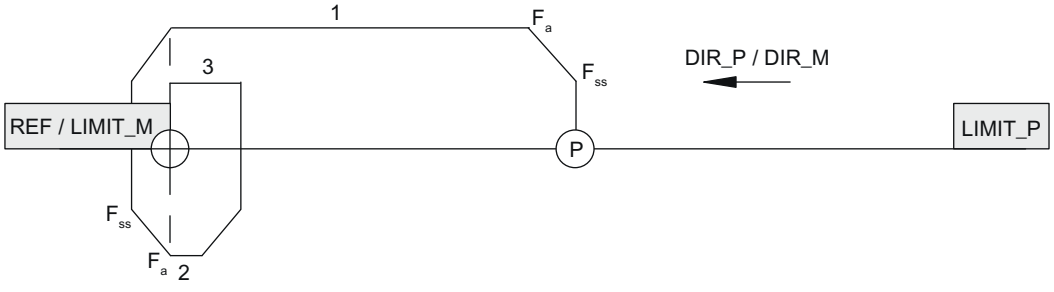

- **Start on the limit switch**

The figure below shows a reference point approach with start on the limit switch LIMIT_P, in which REF is simultaneously LIMIT_P.

Response to defective reference cam with limit switch (interruption of traversing)

The figure below shows a reference point approach with defective reference cam and start before the REF.

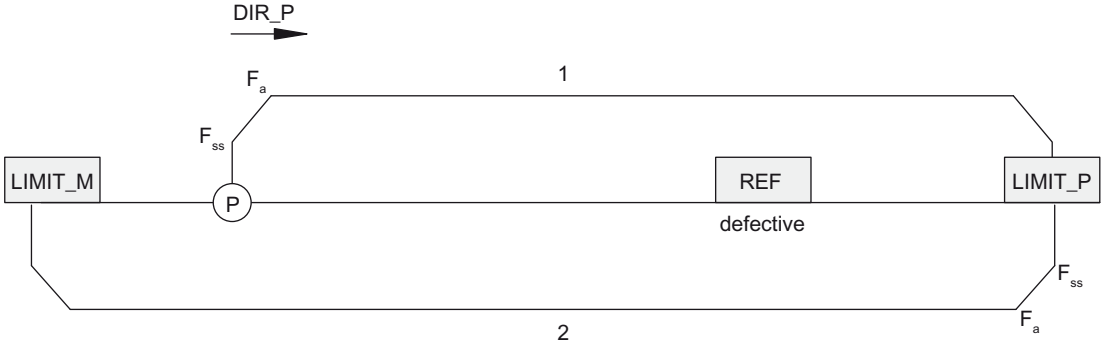

The figure below shows a reference point approach with defective reference cam and start on the LIMIT_P.

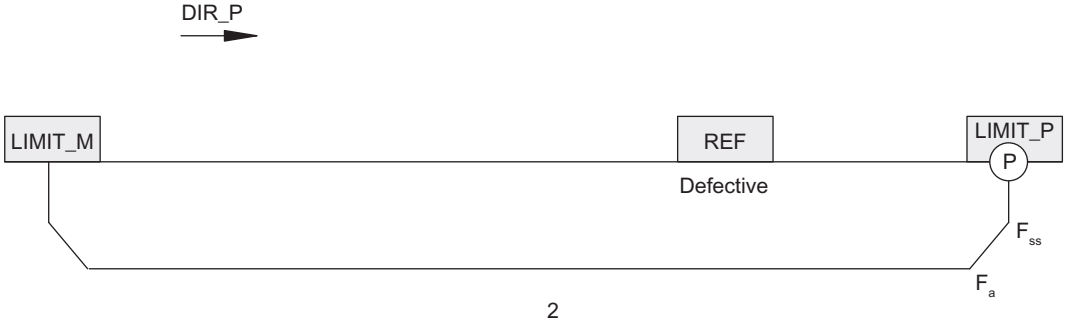

Behavior in the case of a constantly set reference cam without limit switch

At the end of the first section, after 16777215 pulses have been output, traversing is terminated with cleared SYNC and POS_RCD status bits.

Response to failure of the reference cam without limit switch

All three sections of traversing are executed, each with output of 16777215 pulses. Afterwards, the search is interrupted with cleared SYNC and POS_RCD status bits.

Set reference point (as of 6ES7138-4DC01-0AB0) (S7-300, S7-400)

Description of functions

The home position marks the reference point of your drive system which the subsequent absolute incremental modes and the position value in the feedback interface reference. You define the home position by specifying the absolute position value for the current position of the stepping motor.

Job for setting the home position

A job for setting the home position is a virtual job without traversing movement. It contains the following information:

- Position of the home position
- Mode = 4 for setting home position
- Any direction specification as start (Page 2683)

Note

The 1STEP 5V checks the set position for limits (minimum 0 and maximum 16777215). The full-scale value can be configured.

Note

Because you can set the reference point within the traversing range to values from 0 to 16777215, situations are possible in which a limit switch is no longer reachable.

Examples:

- If you set the reference point within the traversing range to "0", the limit switch in backward direction is no longer reachable by a traversing job, because you cannot approach negative positions.
  Ensure that all positions to be approached are in the positive value range.
- In non-synchronized state, the axis cannot traverse a distance that is greater than the configured traversing range. If the configured traversing range is less than the distance between the two limit switches and the axis is at the limit switch in the forward direction, the axis cannot transverse to the limit switch in the backward direction with a relative approach.

Feedback messages

After correct execution of the job, the feedback bits (Page 2687) SYNC and POS_RCD are set.

If the feedback bit SYNC is already set prior to the execution of the job, it remains set. You can also recognize the correct execution from the change of the position value in the feedback interface.

Relative incremental mode (relative positioning) (S7-300, S7-400)

Description of functions

You can use the relative incremental mode to move the stepping motor a defined distance and thus approach a specified position.

You can determine the direction of traversing and the velocity at the start.

Traversing job for relative incremental mode

The traversing job contains the following information:

- Distance (number of pulses to be emitted)
- Multiplier G for the velocity/output frequency F_a
- Reduction factor R for the assigned parameters base frequency F_b
- Mode = 0 for incremental mode, relative
- Stop at the reference cam (Page 2669)
- Direction specification as start (Page 2683)

Note

The 1STEP 5V checks the specified distance for limits (minimum 1 and maximum 16777215 pulses). The distance to the limit switch is not checked by the 1STEP 5V. Traversing is stopped at the latest when the limit switch is reached.

Feedback messages

The feedback bit (Page 2687)POS_RCD is reset at the beginning of incremental mode.

While the incremental mode is active, it is indicated by the set POS feedback bit.

After incremental mode has been correctly executed, the set POS_RCD feedback bit indicates that the position has been reached.

If the incremental mode is interrupted, the POS_RCD feedback bit remains reset. After incremental mode has been stopped, the distance still to be traversed is displayed if the feedback value (Page 2687) is set to "Residual distance".

Absolute incremental approach (absolute positioning) (as of 6ES7138-4DC01-0AB0) (S7-300, S7-400)

Description of functions

You can use the absolute mode to move the stepping motor to a defined position and thus approach a specified position.

The velocity is specified at the start. The direction and the distance of traversing are determined automatically by the 1STEP 5V on the basis of the starting position (actual position value). You can also specify the direction for a modulo axis.

Note

If you set Forward start and Backward start (DIR_P and DIR_M) simultaneously at a modulo axis, the 1STEP 5V then automatically selects the shortest distance to reach (Page 2671) the target position.

Traversing job for absolute incremental mode

The traversing job contains the following information:

- Target position
- Multiplier G for the velocity/output frequency F_a
- Reduction factor R for the assigned parameters base frequency F_b
- Mode = 2 for incremental mode
- Any direction specification as start (Page 2683)

Note

The 1STEP 5V checks the set position for limits (minimum 0 and maximum 16777215). The full-scale value can be configured.

The traversing job is only executed if you have determined or specified the position of the home position beforehand (the SYNC bit has to be set, see "Reference point approach (Page 2659)" or "Set reference point (Page 2664)").

The control signal (Page 2683) "Stop at reference cam" is not taken into consideration.

Feedback messages

The feedback bit (Page 2687)POS_RCD is reset at the beginning of incremental mode.

While the incremental mode is active, it is indicated by the set POS feedback bit.

After incremental mode has been correctly executed, the set POS_RCD feedback bit indicates that the position has been reached.

If the incremental mode is interrupted, the POS_RCD feedback bit remains reset. After incremental mode has been stopped, the distance still to be traversed is displayed if the feedback value (Page 2687) is set to "Residual distance".

Speed mode (as of 6ES7138-4DC01-0AB0) (S7-300, S7-400)

Description of functions

In this operating mode you specify the frequency with which the pulses (steps) are output. When you change the frequency, the pulses are output with the new frequency after an acceleration or deceleration phase. The output is carried out continuously until you stop the traversing job or a traversing range is reached at a linear axis.

The figure below shows the speed-control mode with a modulo axis.

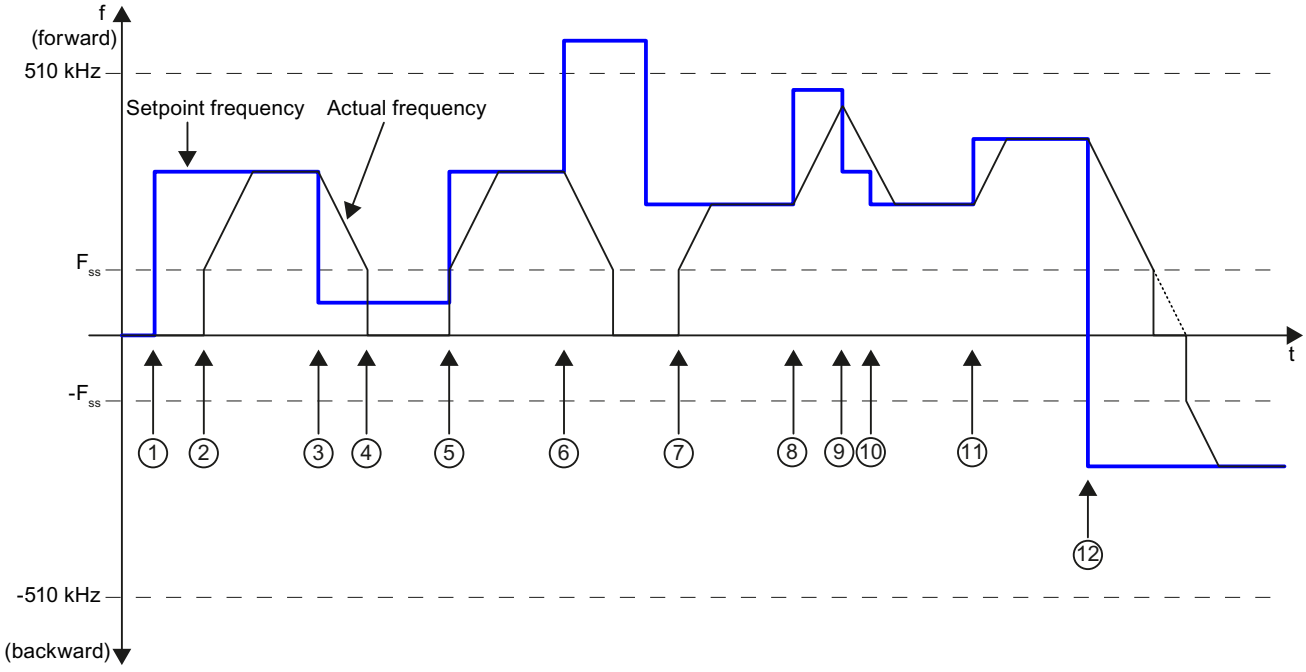

① Specification for setpoint frequency

② Start of the speed-control mode with frequency F_{SS}, acceleration ramp up to the setpoint frequency

③ New setpoint frequency (< F_{SS})

④ Deceleration ramp up to F_{SS}, then stop of pulse output (no stop of speed-control mode)

⑤ Specification for setpoint frequency: Continuation of speed-control mode with F_{SS}, acceleration ramp up to the setpoint frequency

⑥ Specification for invalid setpoint frequency: Stop of speed-control mode

⑦ Start of the speed-control mode with frequency F_{SS}, acceleration ramp up to the setpoint frequency

⑧ Specification for setpoint frequency

⑨ Specification for setpoint frequency: Change from acceleration ramp to deceleration ramp

⑩ Specification for setpoint frequency: Continuation of the deceleration ramp

⑪ Specification for setpoint frequency

⑫ Specification for negative setpoint frequency: Direction change

Traversing job for speed-control mode

The traversing job contains the following information:

- Setpoint frequency as 32-bit value (STEP 7 data type REAL)
- Direction specification by the sign of the setpoint frequency (positive: forward)
- Mode = 3 for speed-control mode
- Any direction specification as start (Page 2683)

Note

The 1STEP 5V checks the set position for limits (minimum -510,0 kHz and maximum +510,0 kHz).

The specified frequency is approached with the configured acceleration a under consideration of the start-stop frequency F_{ss}. No pulse output is emitted at frequencies that are less than F_{ss}.

The continuous output of the frequency is terminated at the following events:

- Reaching of the limits of the configured traversing range (0 in the direction backward) unless a modulo axis is configured
- Other aborting conditions for traversing jobs (Page 2669)

Feedback messages

While the traversing job is active, it is indicated by the set feedback bit (Page 2687) POS.

When a new frequency is specified, the POS_RCD feedback bit is cleared. When the new frequency has been reached after the acceleration or deceleration phase, POS_RCD is set again.

The current frequency is displayed in the feedback interface as a 32-bit value (STEP 7 data type REAL) if the feedback value (Page 2687) is set to "Frequency".

Note

If a traversing job is being executed with a frequency of 0.0 Hz, the job is stopped if a frequency is unequal to 0.0 Hz and the STOP control bit requests the job stop. The status of the feedback bit POS_RCD is then undefined.

Hold traversing job (S7-300, S7-400)

Specific holding of the traversing job

- Caused by	Displayed by Feedback Bit
STOP by control bit	-
External STOP at digital input	STOP_EXT

- Caused by	Displayed by Feedback Bit
Limit switch in the forward direction reached (LIMIT_P or digital input)	STOP_LIMIT_P
Limit switch in the backward direction reached (LIMIT_M or digital input)	STOP_LIMIT_M
STOP at the reference cam	STOP_REF

Note

Remember that the limit switches are used in the reference point approach mode also to search for the reference cam.

If a traversing job to LIMIT_P was stopped, then you can move the axis away from the limit switch using a new traversing job with DIR_M. When stopping at LIMIT_M, you can move the axis away from the limit switch with DIR_P.

Stop at the reference cam

If the "Hold at reference cam" function is selected (the control bit STOP_REF_EN is set) at the start of traversing and the reference cam is detected during traversing, the stepping motor is halted and traversing is terminated.

Holding the traversing job in exceptional circumstances

In the following cases the traversing job is halted with loss of the synchronization:

- Incorrect operation in the control interface during an active traversing job
- External error ERR_24V through overload of the encoder supply (e.g. short circuit)
- CPU/Master STOP
- At linear axis: Reaching of the limit of the traversing range

Effects

If one of the above reasons for holding the current positioning operation occurs, it is terminated by a deceleration ramp.

The return value continues to be updated even when the traversing job is halted in exceptional cases. This enables you to traverse the residual distance after holding by means of a new traversing job in the "Relative incremental mode".

Limit Switches and External STOP

By assigning parameters, you can choose to wire normally open or normally closed contacts for the external STOP and the limit switches.

- **Normally closed contact means:**The external STOP and the effect of the limit switches are triggered by a 0 signal. When the limit switches are reached, delete the associated control bit.
- **Normally open contact means:**The external STOP and the effect of the limit switches are triggered by a 1 signal. When the limit switches are reached, set the associated control bit.

Note

In case of holding during the acceleration phase the 1STEP 5V continues to send pulses for a maximum of 50 ms at the frequency already reached before it starts braking. This avoids abrupt changes in frequency, which can lead to step losses.

Axis type and traversing range (as of 6ES7138-4DC01-0AB0) (S7-300, S7-400)

Overview

During configuration you specify the axis type to be controlled by the stepping motor controlled by 1STEP 5V. You have the choice of the following axis types:

- Linear axis
- Modulo axis

Description of the function

Linear axis

The traversing range of a linear axis can be set. The low limit is always 0, the high limit is configured and has a value range from 1 to 16777215.
The traversing range can be restricted further using limit switches (operating range).

The following figure shows a linear axis.

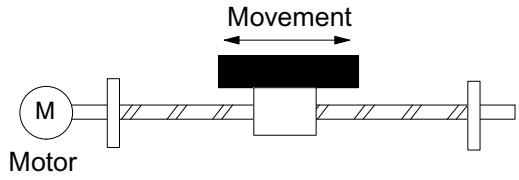

Modulo axis

A modulo axis is a particular form of the rotary axis. The following figure shows a modulo axis.

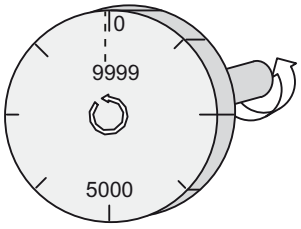

End of the modulo axis

The "Traversing range" parameter is used to specify the end of the modulo axis.

The actual position value cannot reach the traversing range value, because this highest value lies physically at the same position as the start of the modulo axis (0).

Example:

You specify the value 10000 as the traversing range, see figure above.

During a forward movement the position value jumps in the feedback interface from 9999 to 0, during a backward movement from 0 to 9999.

Reference point approach

If you have selected the modulo axis during the configuration and have assigned a reference cam to your drive system, you can carry out a reference point approach (Page 2659).

Traversing is aborted unsuccessfully if the reference cam is not found after the output of a number of pulses that corresponds to the configured traversing range. The status bits (Page 2687)SYNC and POS_RCD will then remain deleted.

Set home position

You may only specify values from 0 to the end of the configured end of the traversing range – 1 for the position of the home position.

Relative positioning

The end of the traversing range (end of the modulo axis) may be exceeded in both directions.

Absolute positioning

If you have selected the modulo axis during the configuration, you may only specify values from 0 to the configured end of the traversing range – 1 for the target position.

In contrast to the linear axis, specify on which path the target position is to be reached by means of the direction specification (Page 2683) when the traversing job is started:

- Backward start (DIR_M): The 1STEP 5V approaches the target position in the direction of lower actual position values (Option 1 in the following figure).
- Forward start (DIR_P): The 1STEP 5V approaches the target position in the direction of higher actual position values (Option 2 in the following figure).
- Forward start and backward start simultaneously (DIR_P and DIR_M): The 1STEP 5V automatically selects the shortest path for reaching the target position (Option 1 in the following figure).

The figure below shows absolute positioning with a modulo axis.

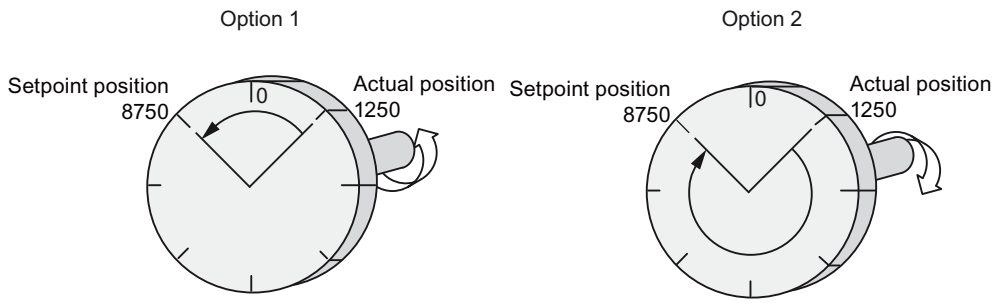

Pulse Enable (S7-300, S7-400)

Description of the function

Pulse enable permits the output of pulses from the 1STEP 5V to the power unit. A run is not possible without pulse enable.

Activating Pulse Enable

You can activate pulse enable by one of the following methods:

- Through the digital input DI0 when "Function DI0" is configured as an external enable pulse (Page 2675)

or

- Through the control bit DRV_EN when the "Function DI0" is configured as an external STOP (Page 2675) or limit switch forward or backward (Page 2675)

You can recognize the assigned pulse enable through the fact that

- The LED (Page 2690)RDY at the 1STEP 5V lights up in the case of correct configuration.
- The feedback bit (Page 2687)STS_DRV_EN is set.

Deleting the Pulse Enable

Deleting the pulse enable during a run terminates the run immediately because no more pulses are emitted to the power unit. The residual distance and actual position value are no longer valid. The synchronization of the axis by means of the reference point is lost. The SYNC feedback bit and the RDY LED are deleted.

Deleting the pulse enable when the motor is at standstill deletes the SYNC feedback bit and the RDY LED.

In this case, it may be necessary to carry out a reference point approach (Page 2659).

Changing Parameters during Operation (S7-300, S7-400)

Introduction

You can change several of the 1STEP 5V parameters during operation without having to reassign the parameters of the whole ET 200S station.

Parameters That Can Be Changed

The following parameters can be changed:

- Base Frequency $F_{b:}$
- Multiplier n for start-stop frequency F_{ss}
- Multiplier i for acceleration / delay
- Feedback value in the feedback interface

When you start changing parameters by means of the control bit (Page 2683)C_PAR, the parameters are checked for permitted values. If you have set invalid values, the feedback bit (Page 2687)ERR_JOB is set.

Only the feedback bits for the ERR_JOB and STS_JOB job processing are affected by the configuration job.

Carrying Out a Parameter Change

The following figure shows the process for a parameter change.

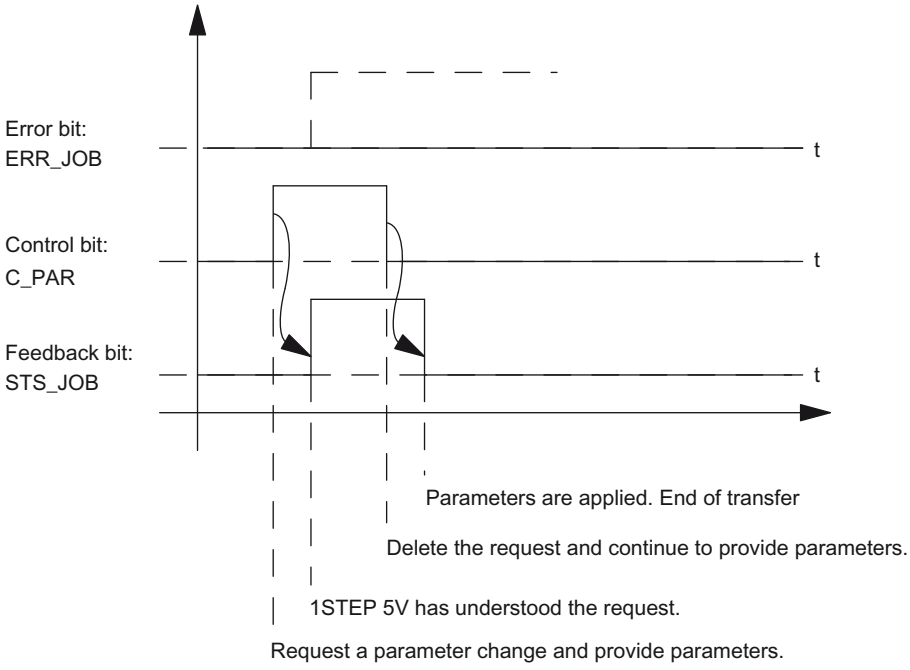

Note

Only one of the following control bits can be set at a particular time:

DIR_Por DIR_Mor C_PAR.

Otherwise, the ERR_JOB error is reported. The job error message is deleted by the start of the next job.

Behavior of the Digital Inputs (S7-300, S7-400)

Introduction

You can configure the function and the behavior (active level) of the digital inputs DI0 (3) and DI1 (7) . These parameters cannot be changed using the user program.

Digital input DI0 (3)

You can configure the function of the digital input DI0 (3) as:

- External enable pulse
- External STOP

- Limit switch in the forward direction
- Limit switch in the backward direction

You can also configure the behavior of the digital input DI0 (3) as:

- Normally closed contact
- Normally open contact

Digital input DI0 (3) as an external enable pulse

The input must be set (Page 2672) (activated) during operation. If the input is set and the configuration correct, the 1STEP 5V is ready for operation.

Digital input DI0 (3) as an external STOP

With this input function, you can halt (Page 2668) a current traversing job using an external signal.

Digital input DI0 (3) as a limit switch in the forward or backward direction

With these input functions, you can limit the traversing range in the forward or backward direction through an external signal. The signal has the same effect as one of the two control bits LIMIT_P or LIMIT_M in the control interface (Page 2683).

Digital input DI1 (7)

You can configure the function of the digital input DI1 (7) as:

- Reference switch: Reference cam
- Reference switch and limit switch in the forward direction
  This parameter selection is only possible if "Function DI0" is not configured as a "Limit switch in forward direction".
- Reference switch and limit switch in the backward direction
  This parameter selection is only possible if "Function DI0" is not configured as a "Limit switch in backward direction".

You can also configure the behavior of the digital input DI1 (7) as:

- Normally closed contact
- Normally open contact

Digital input DI1 (7) as a reference switch

You can wire a switch to this input for the reference cam.

You need a reference cam for the following:

- For a reference point approach
- For an incremental mode with hold on the reference cam.

Digital input DI1 (7) as a reference switch and limit switch in the forward or backward direction

With these input functions, you additionally limit the traversing range in the forward or backward direction through the reference cam. The signal also has the same effect as one of the two control bits LIMIT_P or LIMIT_M in the control interface (Page 2683).

If you have configured the behavior of the digital input DI1 (7) as a "Reference and limit switch", then 1STEP 5V automatically selects the starting direction toward the limit switch, regardless of the direction specified in the traversing job.

Reaction to CPU/Master STOP (S7-300, S7-400)

Introduction

The 1STEP 5V detects the CPU/Master STOP. It responds to it by stopping the active traversing job (Page 2668).

Exiting the CPU/Master STOP status

ET 200S station	1STEP 5V
Without reconfiguration of the ET 200S station	• The feedback interface of the 1STEP 5V remains current. • The values changed by means of parameter assignment job are maintained. • If a control bit was set (DIR_P, DIR_M, C_PAR) when the CPU/Master STOP occurred, the bits STS_JOB and ERR_JOB are set when the CPU/Master STOP status is exited. Delete the control bit. Traversing / the parameter assignment job is not executed. You can then start a new traversing by means of the control bit. • After the delay ramp, the pulse enable, the RDY LED, and the SYNC status bit are deleted.
With reconfiguration of the ET 200S station	• Information on previous searches and parameter assignment jobs is reset. • If pulse enable was activated by means of the control bit DRV_EN at the time of the CPU/Master STOP, the pulse enable, the RDY LED, and the SYNC status bit are deleted after the delay ramp.

Reconfiguration of the ET 200S station

Reconfiguration of the ET 200S station is carried out by your CPU/ DP master at:

- POWER ON of the CPU / DP master
- POWER ON of the IM 151 / IM 151 FO
- After failure of the DP transmission

- Upon loading changed parameters or configuration of the ET 200S station into the CPU / DP master
- When the 1STEP 5V is plugged
- Upon power on or inserting of the appropriate power module

See also

Pulse Enable (Page 2672)

Behavior of the digital inputs (-4DC00-) (S7-300, S7-400)

Digital input DI 3

You can configure the function of the digital input DI 3 as:

- External enable pulse
- External STOP

You can also configure the behavior of the digital input DI 3 as:

- Normally closed contact
- Normally open contact

Digital input DI 3 as an external enable pulse

The input must be set (Page 2678) (activated) during operation. If the input is set and the configuration correct, the 1STEP 5V/204kHz is ready for operation.

Digital input DI 3 as an external STOP

With this input function, you can hold an active traversing job using an external signal.

Digital input REF

You can wire a switch to this input for the reference cam.

You need a reference cam for the following:

- For a reference point approach
- For an incremental mode with hold on the reference cam.

Pulse enable (-4DC00-) (S7-300, S7-400)

Description of the function

Pulse enable permits the output of pulses from the 1STEP 5V/204kHz to the power unit. A run is not possible without pulse enable.

Activating Pulse Enable

You can activate pulse enable by one of the following methods:

- Through the digital input DI 3 when "Function DI" is configured as an external enable pulse (Page 2677)

or

- Through the control bit DRV_EN when the "Function DI" is configured as an external STOP (Page 2677)

You can recognize the assigned pulse enable through the fact that

- The LED (Page 2690)RDY at the 1STEP 5V/204kHz lights up in the case of correct configuration.
- The STS_DRV_EN feedback bit is set.

Deleting the Pulse Enable

Deleting the pulse enable during a run terminates the run immediately because no more pulses are emitted to the power unit. The stepper motor runs without pulse control. This means the residual distance is no longer valid. The synchronization of the axis by means of the reference point is lost. The SYNC feedback bit and the RDY LED are deleted.

Deleting the pulse enable when the motor is at standstill deletes the SYNC feedback bit and the RDY LED.

In this case, it may be necessary to perform a reference point approach.

6.2.8.4 1STEP 5V configuration and parameter assignment (S7-300, S7-400)

Parameters (S7-300, S7-400)

Enabling group diagnostics (S7-300, S7-400)

Group diagnostics

Group diagnostics are enabled in the respective check box.

A "Short circuit encoder supply" ERR_24V or a "Parameter assignment error" ERR_PARA lead to channel-based diagnostics if group diagnostics is enabled.

Selecting the base frequency (S7-300, S7-400)

Base frequency F_b:

The base frequency F_b is the base value for setting the start-stop frequency, output frequency and acceleration/delay.

Select the base frequency suitable for your application.

Detailed information can be found under Setting the base frequency (Page 2655).

Specifying multiplier n (S7-300, S7-400)

Multiplier n

With multiplier n, you can set the start-stop frequency in increments:

$F_{ss} = F_b \times n$

Permitted range of values: 1 … 255

See also

Setting the Base Frequency (Page 2655)

Specifying time interval i (S7-300, S7-400)

Time interval i

With multiplier i, you can set the acceleration/delay in increments:

$a = F_b / (i \times 0.128\ ms)$

Permitted range of values: 1 … 255

See also

Setting the Base Frequency (Page 2655)

Selecting the feedback value (S7-300, S7-400)

Feedback value

Select the feedback value that is supplied in the feedback interface (Page 2687):

- Residual distance
- Position
- Frequency

Selecting the function DI0 (S7-300, S7-400)

Function DI0

Select which function the digital input DI0 (3) (Page 2674) should have:

- An external enable pulse
- An external STOP
- Limit switch in the forward direction
- Limit switch in the backward direction

Selecting the function DI1 (S7-300, S7-400)

Function DI1

Select which function the digital input DI1 (7) (Page 2674) should have:

- A reference switch (reference cam)
- Reference switch and limit switch in the forward direction
  This parameter selection is only possible if "Function DI0" is not configured as a "Limit switch forward".
- Reference switch and limit switch in the backward direction
  This parameter selection is only possible if "Function DI0" is not configured as a "Limit switch backward".

Selecting the behavior of inputs DI0 and DI1 (S7-300, S7-400)

Input DI0, input DI1

Select the behavior of digital inputs DI0 (3) and DI1 (7):

- Normally closed contact
- Normally open contact

See also

Behavior of the Digital Inputs (Page 2674)

Hold traversing job (Page 2668)

Selecting the behavior of the limit switches in the control interface (S7-300, S7-400)

Limit switches in the control interface

Select the behavior of the limit switches in the control interface (Page 2683):

- Normally closed contact
- Normally open contact

See also

Hold traversing job (Page 2668)

Enabling the modulo axis (S7-300, S7-400)

Modulo axis

The axis type "linear axis" is preconfigured at the 1STEP 5V. If you want to control a "modulo axis", you have to enable the "Modulo axis" check box.

See also

Axis type and traversing range (as of 6ES7138-4DC01-0AB0) (Page 2670)

Specifying the traversing range (S7-300, S7-400)

Traversing range

The "Traversing range" parameter is used to specify the end of the modulo axis.

Permitted range of values: 1 … 16777216

See also

Axis type and traversing range (as of 6ES7138-4DC01-0AB0) (Page 2670)

1STEP 5V control and feedback interface (S7-300, S7-400)

Control interface of 1STEP 5V (S7-300, S7-400)

Note

For the 1STEP 5V, the following data of the control interface are consistent:

- Bytes 0 to 3
- Bytes 4 to 7

Assignment of the control interface

The following table shows the assignment of the control interface (outputs) of 1STEP 5V.

Address	Assignment	
Bytes 0 to 3	**Relative incremental mode (Byte 4 Bit 2 to Bit 0 is 0), Absolute incremental mode (Byte 4 Bit 2 to Bit 0 is 2)**	
	Byte 0	Multiplier G: $F_a = F_b \times R \times G$ (value range 1 to 255)
	Byte 1	Distance or position bit 23 to bit 16
	Byte 2	Distance or position bit 15 to bit 8
	Byte 3	Distance or position bit 7 to bit 0
	Reference point approach (Byte 4 Bit 2 to Bit 0 is 1)	
	Byte 0	Multiplier G: $F_a = F_b \times R \times G$ (value range 1 to 255)
	Byte 1	Position bit 23 to bit 16
	Byte 2	Position bit 15 to bit 8
	Byte 3	Position bit 7 to bit 0
	Set reference point approach (Byte 4 Bit 2 to Bit 0 is 4)	
	Byte 0	Reserve = 0
	Byte 1	Position bit 23 to bit 16
	Byte 2	Position bit 15 to bit 8
	Byte 3	Position bit 7 to bit 0
	Speed-control mode (Byte 4 Bit 2 to Bit 0 is 3)	
	Bytes 0 to 3	Frequency as STEP 7 data type REAL
	Parameter assignment request (Byte 5 Bit 6 is 1)	
	Byte 0	Reserve = 0
	Byte 1	Multiplier i: $a = F_b \times R / (i \times 0.128\ ms)$ (value range 1 to 255)
	Byte 2	Multiplier n: $F_{ss} = F_b \times n \times R$ (value range 1 to 255)
	Byte 3	Base frequency F_b: • 0 = 800 Hz • 1 = 400 Hz • 2 = 200 Hz • 3 = 80 Hz • 4 = 40 Hz • 5 = 20 Hz • 6 = 8 Hz • 7 = 4 Hz • 8 = 2000 Hz

Address	Assignment		
Byte 4	**Traversing job**		
	Bit 7	Reduction factor • 0 = Factor 1.0 (no reduction) • 1 = Factor 0.1	R
	Bit 6	Hold traversing job	STOP
	Bit 5	Backward start	DIR_M
	Bit 4	Forward start	DIR_P
	Bit 3	Firmware Version V1.0.0: Reserve = 0 Firmware-Version V1.0.1 and higher: Release of frequencies below F_{SS} in speed-control mode	EN_LF
	Bit 2 to Bit 0	Operating mode • 0 = Relative incremental mode (relative positioning) • 1 = Reference point approach • 2 = Absolute incremental mode (absolute positioning) • 3 = Speed-control mode • 4 = Set home position	MODE
	Parameter assignment request		
	Bit 7 to Bit 0	Value is not relevant and is ignored	
Byte 5	**Traversing job**		
	Bit 7	Diagnostics error acknowledgment	EXTF_ACK
	Bit 6	Must be set to 0	C_PAR
	Bit 5 to Bit 4	Value is not relevant and is ignored	FEEDBACK
	Bit 3	Stop at the reference cam	STOP_REF_EN
	Bit 2	Pulse enable	DRV_EN
	Bit 1	Limit switch in the forward direction	LIMIT_P
	Bit 0	Limit switch in the backward direction	LIMIT_M
	Parameter assignment request		
	Bit 7	Diagnostics error acknowledgment	EXTF_ACK
	Bit 6	Change parameter; must be set to 1	C_PAR
	Bit 5 to Bit 4	Feedback value in the feedback interface • 00 = Residual distance • 01 = Position • 10 = Frequency • 11 = Reserved	FEEDBACK
	Bit 3	Value is not relevant and is ignored	STOP_REF_EN
	Bit 2	Pulse enable	DRV_EN
	Bit 1	Value is not relevant and is ignored	LIMIT_P
	Bit 0	Value is not relevant and is ignored	LIMIT_M
Byte 6	Reserve = 0		
Byte 7	Reserve = 0		

Notes on the Control Bits

Bits	Explanation
Base frequency $F_{b:}$	Coding for setting the base frequency in steps: • 0 = 800 Hz • 1 = 400 Hz • 2 = 200 Hz • 3 = 80 Hz • 4 = 40 Hz • 5 = 20 Hz • 6 = 8 Hz • 7 = 4 Hz • 8 = 2000 Hz You can change this parameter during operation using a Parameter assignment request (Page 2673) with the control bit C_PAR.
Operating mode	Coding for operating mode: • 0 = Relative incremental mode (relative positioning) • 1 = Reference point approach • 2 = Absolute incremental mode (absolute positioning) • 3 = Speed-control mode • 4 = Set home position
C_PAR	A parameter change is requested with this bit.
DIR_M	This bit requests and starts a traversing job in the backward direction.
DIR_P	This bit requests and starts a traversing job in the forward direction.
DRV_EN	If you are not using the digital input DI0 (3) as an external pulse enable, this bit is interpreted as a pulse enable. If no external pulse enable is configured and DRV_EN goes to 0, the referencing (SYNC) may be lost.
EN_LF	In Firmware Version V1.0.1 and higher, you use this bit in speed-control mode to also permit frequencies that are less, according to amount, than the configured start-stop frequency. If the specified frequency, according to amount, is below the start-stop frequency, it is output without an acceleration or deceleration ramp. Frequencies above the start-stop frequency are output with acceleration or deceleration ramp. The lowest frequency, according to amount, that can be output is 0.1 Hz. The bit is only evaluated when speed-control mode is started. A change of the bit during a traversing job in speed-control mode has no effect.
Limit switches LIMIT_M	This limit switch limits the travel range in the backward direction. You set or delete this bit in your user program.
Limit switches LIMIT_P	This limit switch limits the travel range in the forward direction. You set or delete this bit in your user program.
EXTF_ACK	Acknowledgment bit for diagnostic message

Bits	Explanation
FEEDBACK	Coding for the feedback value in the feedback interface: • 00 = Residual distance • 01 = Position • 10 = Frequency • 11 = Reserved You can change this parameter during operation using a Parameter assignment request (Page 2673) with the control bit C_PAR.
Frequency	A 32-bit value (STEP 7 data type REAL) that contains the pulse frequency to be output.
Multiplier G	Factor for setting the velocity / output frequency in steps
Multiplier i	Factor for setting the acceleration / deceleration in steps You can change this parameter during operation using a Parameter assignment request (Page 2673) with the control bit C_PAR.
Multiplier n	Factor for setting the start-stop frequency in steps You can change this parameter during operation using a Parameter assignment request (Page 2673) with the control bit C_PAR.
Position	24-bit value that contains the target position to be approached
Reduction factor R	The base frequency F_b is multiplied by 0.1 if the bit is set. This reduces the output frequency F_a, the start-stop frequency F_{ss}, and the acceleration / deceleration a by the same amount.
STOP	With this bit, you can stop a traversing job with a delay ramp at any time (see also "Stop traversing job (Page 2668)").
STOP_REF_EN	When the bit is set, the "Stop at the reference cam" function is active. When the reference cam is recognized, the traversing job is stopped with a delay ramp (see also "Stop traversing job (Page 2668)").
Distance	A 24-bit value that contains (without signs) the number of pulses that have to be traversed.

Feedback interface of 1STEP 5V (S7-300, S7-400)

Note

For the 1STEP 5V, the following data of the feedback interface are consistent:

- Bytes 0 to 3
- Bytes 4 to 7

Assignment of the feedback interface

The following table shows the assignment of the feedback interface (inputs) of 1STEP 5V.

Address	Assignment		
Bytes 0 to 3	Residual distance (bit 23 to bit 0 of 32 bits) / Position (bit 23 to bit 0 of 32 bits) / Frequency (32 bits, STEP 7 data type REAL)		
Byte 4	Bit 7	Short-circuit at encoder supply	ERR_24V
	Bit 6	Reserve = 0	
	Bit 5	Programming error	ERR_PARA
	Bit 4	Determining the home position	SYNC
	Bit 3	Reserve = 0	
	Bit 2	Position reached	POS_RCD
	Bit 1	Error during job transfer	ERR_JOB
	Bit 0	Job transfer running	STS_JOB
Byte 5	Bit 7	Traversing job running	POS
	Bit 6	Limit switch forward is cause for stop	STOP_LIMIT_P
	Bit 5	Limit switch backward is cause for stop	STOP_LIMIT_M
	Bit 4	External STOP is cause for stop	STOP_EXT
	Bit 3	Reference cam is cause for stop	STOP_REF
	Bit 2	Status DI0	STS_DI0
	Bit 1	Status DI1	STS_DI1
	Bit 0	Status pulse enable active	STS_DRV_EN
Byte 6	Error number at an error during job transfer		
Byte 7	Reserve = 0		

Notes on the Feedback Bits

Bits	Explanation
Frequency	A 32-bit value (STEP 7 data type REAL) that contains the current pulse frequency. The pulse frequency can take on positive (forward direction) and negative (backward direction) values.
ERR_JOB	This bit is set if the job is not clear or not possible. The error cause is specified in more detail by the returned error number (see the following table "Error number in the feedback interface").
ERR_PARA	Incorrect parameter assignment for the ET 200S station. The error cause is specified in more detail by the returned error number (see the following table "Error number in the feedback interface"). The parameter error bit is deleted when a correct parameter assignment is transmitted.
ERR_24V	The encoder supply has been overloaded (e.g. by a short circuit) and is now switched off. ERR_24V is reset if it has been acknowledged with the EXTF_ACK control bit. After the overload has been eliminated, the encoder supply is switched on again and ERR_24V remains cleared.
Error number	Specifies the error cause if ERR_JOB or ERR_PARA is set (see table below "Error numbers in the feedback interface").
POS	Traversing: This bit is set as long as the traversing job is running.

Bits	Explanation
POS_RCD	POS_RCD is cleared at the start of an incremental mode or at specification of a new setpoint frequency in speed-control mode. POS_RCD is set after a correctly executed incremental mode or when the setpoint frequency has been reached in speed-control mode. If traversing was interrupted (if the traversing job has stopped (Page 2672) or the pulse enable is deleted (Page 2668)), POS_RCD remains cleared.
Position	A 24-bit value that contains the current absolute position (without signs). Byte 0 of the feedback interface is 0.
Residual distance	A 24-bit value that contains the number of pulses that still have to be traversed (without signs). Byte 0 of the feedback interface is 0.
STOP_EXT	Cause for stop: This bit is set if the traversing job has been stopped by an external STOP.
STOP_LIMIT_M	Cause for stop: This bit is set if the traversing job has been stopped by reaching of the limit switch backward.
STOP_LIMIT_P	Cause for stop: This bit is set if the traversing job has been stopped by reaching of the limit switch forward.
STOP_REF	Cause for stop: This bit is set if the traversing job has been stopped by reaching of the reference cam.
STS_DI0	The bit displays the status of the DI0 (3) digital input.
STS_DI1	The bit displays the status of the DI1 (7) digital input.
STS_DRV_EN	This bit is set when one of the following occurs, depending on the assigned parameter function of the digital input DI0: • The external enable pulse is set. or • The DRV_EN control bit is set for the pulse enable.
STS_JOB	This bit is set as feedback when a job request for a traversing or parameter assignment job is detected and then reset when the job has been executed.
SYNC	This bit is set after a correct reference point approach or after manual specification of the home position has been set. The SYNC bit is cleared after parameter assignment with new ET 200S station parameters or after deletion of the pulse enable.

Error number

If an error with the job transfer (ERR_JOB is set) or an error in the basic parameter assignment (ERR_PARA is set) is displayed in the feedback interface, the error cause is specified in more detail by means of an error number.

The table below shows the error numbers of the feedback interface.

Error number	Meaning
General error causes	
0_D	No error (then ERR_JOB or ERR_PARA is also not set)
1_D	Combination of the control bits (DIR_P, DIR_M, C_PAR) is invalid
2_D	Another job is still running.
Causes of errors with a traversing job	
16_D	Start forward (DIR_P) at limit switch forward (LIMIT_P) active
17_D	Start backward (DIR_M) at limit switch backward (LIMIT_M) active
18_D	Start with set control bit STOP

Error number	Meaning
19_D	Start at external STOP active
20_D	Start at a missing pulse enable (internal or external)
21_D	Start with set STOP_REF_EN with active reference cam
22_D	Start without reference (at absolute incremental mode)
23_D	Start with diagnostic error present
24_D	Start was interrupted by CPU/Master STOP
25_D	Start with incorrect operating mode (not identical with requirement)
26_D	Distance or position specification is invalid
27_D	Multiplier G for the velocity is zero
28_D	Frequency is invalid at speed-control mode
Error causes at a parameter assignment job or for the basic parameter assignment	
32_D	Specification for the base frequency is invalid
33_D	Multiplier n for start-stop frequency is zero
34_D	Multiplier i for acceleration / delay is zero
35_D	Feedback value for the feedback interface is invalid
36_D	Combination of the functions of DI0 and DI1 is invalid (limit switches)
37_D	Specification for the end of the traversing range is invalid

6.2.8.5 1STEP 5V diagnostics (S7-300, S7-400)

Diagnostics using the LED display (S7-300, S7-400)

LED display on the 1STEP 5V

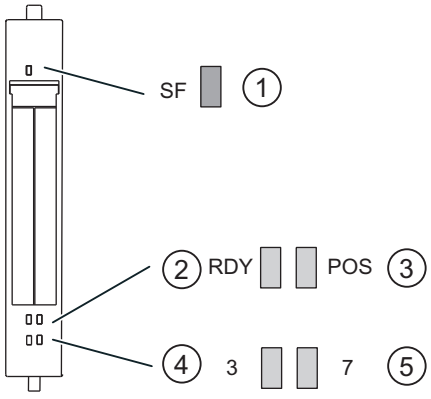

① Group error (red)
② Ready for traversing job (green)
③ Positioning underway (green)
④ Status display for Digital input 0 (green)
⑤ Status display for Digital input 1 (green)

Status and error displays by means of LEDs on the 1STEP 5V

The table below shows the status and error displays on the 1STEP 5V.

Event (LED)					Cause	Remedy
SF	RDY	POS	3	7		
On					No configuration or incorrect module plugged in. There is a diagnostic message.	Check the parameter assignment. Evaluate the diagnostics data.
	On				If parameters were assigned to the module correctly and pulse enable has been activated	
		On			When traversing job is running	
			On		DI 0 is activated.	
				On	DI 1 is activated.	

Error types (S7-300, S7-400)

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

1STEP 5V error types

The following table shows the error types on the 1STEP 5V.

Error type		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the sensor supply.	Check the connection to the switches. Correct the process wiring.
9_D	01001: Error	Internal module error occurred.	Replace the module.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.

6.2.9 Using the IM 174 (S7-300, S7-400)

6.2.9.1 Overview (S7-300, S7-400)

Interface module for analog drives and stepper motors

An IM 174 module (interface module) is an interface module suitable for running up to 4 drives with an analog setpoint interface and with one TTL or SSI encoder per axis on the isochronous PROFIBUS.

Communication between the controller and the IM 174 is performed exclusively using PROFIBUS, via an IM 174-specific message frame type which, in addition to digital input/output data, also contains a message frame type (standard message frame 3 and 81) for each drive

specified according to a PROFIDrive profile. As part of cyclic DP communication, the actual drive values (encoder values) are transferred from the IM 174 module to the controller via the isochronous PROFIBUS, and the speed setpoints calculated by the controller are transferred to the IM 174 module.

The transferred speed setpoints are then output from the IM 174 module to the drives as analog values or pulses.

Field of application

IM 174 can be used for the following control jobs:

- controlling analog drives
- controlling stepper motors
- reading in encoder data

6.2.9.2 Configuring and programming IM174 (S7-300, S7-400)

Configuring IM174 (S7-300, S7-400)

Steps required - Overview

You configure the IM 174 in the Inspector window under "Properties". You must carry out the following steps:

1. Add a PROFIBUS module to the hardware configuration
2. Assign parameters for isochronous PROFIBUS
3. Specify drive and encoder parameters for the axes

Assign parameters for isochronous PROFIBUS

1. Select the PROFIBUS subnetwork in the "PROFIBUS address" section and assign a PROFIBUS address.

   Note

   PROFIBUS address at the DIP switch

   The PROFIBUS address is set via DISourcing outputs on the PROFIBUS module IM 174. A power on/off / is necessary on the IM 174 to activate a PROFIBUS address set on the DIP switch. The assigned PROFIBUS address must be identical to the PROFIBUS address set at the DIP switch.

2. Select the check box "Synchronize DP slave to equidistant DP cycle" in the "Isochronous mode" section and set time Ti/To.
3. Select the frame type (Page 2702) in the "Encoder and drives" section.
4. Specify the address ranges for the input and output parameters of the axes and assign a process display in the "Addresses" section.

Specify drive and encoder parameters for the axes

1. Select the drive type (Page 2693) for each axis and set the drive parameters (Page 2693) in the "Encoder and drives" section.
2. Select the encoder type (Page 2693) for each axis and if necessary set the required encoder parameters (Page 2698) in the "Encoder and drives" section.
3. You can set the parameters (Page 2700) for switch-off behavior in the "Parameters" section.

Encoders and Drives (S7-300, S7-400)

You can use the "Encoders and Drives" property in the area navigation to configure the drives and encoders that are connected to the IM 174.

Drive types

You can select the following drive types:

- Servo (analog drive)
- Stepper (stepper drive)

Encoder types for analog drives

The following encoder types can be selected for analog drives

- Encoder type not available
- Encoder type TTL
- Encoder type SSI

Encoder types for stepper drives

The following encoder types can be selected for stepper drives

- Encoder type not available
- Encoder type TTL
- Encoder type SSI
- Encoder type stepper

Drive parameters (S7-300, S7-400)

Drive types

You can select the following drive types:

- Servo (analog drive)
- Stepper (stepper drive)

Servo drive type

If you have selected the servo drive type, you can switch the voltage range of the analog output voltage via the "Unipolar" check box.

Unipolar not selected

If the "Unipolar" option is **not** selected, an analog voltage in the range of **-10 V** to **+10 V** will be output as the setpoint. The drive can then be traversed in two directions.

Unipolar selected

If the "Unipolar" option **selected**, an analog voltage in the range of **0V** to **+10 V** will be output as the setpoint. The direction of rotation is then output from the IM 174, depending on the current speed setpoint via a digital output of the IM 174:

- Direction of rotation signal for Axis 1 -> Digital output X11, Pin 13
- Direction of rotation signal for Axis 2 -> Digital output X11, Pin 15
- Direction of rotation signal for Axis 3 -> Digital output X11, Pin 17
- Direction of rotation signal for Axis 4 -> Digital output X11, Pin 19

Alt.DrvRdy

Depending on the drive you use, the Drive Ready signal returns different status messages:

- The drive signals "Ready for switching on" with Signal Drive Ready. An enable from the controller is still required to switch the drive on.
- The drive signals "Ready" with the Signal Drive Ready signal. The drive is ready for control in this state and directly follows a setpoint.

You should **not**activate the "Alt.DrvRdy" function, if your drive reports a "ready to switch on" state upon a Drive-Ready signal (mostly in analogue drives).

Activate the "Alt.DrvRdy" function when the drive signals "Ready" with a Drive Ready signal (mostly with servo drives).

Information on which status message the drive you use returns can be found in the product information for your drive.

Stepper drive type

You must enter drive-specific or process-specific frequency values [Hz] in the fields "Max. frequency [Hz]" and "Stand. frequency [Hz]" for the "Stepper" drive type. The necessary power electronics (e.g. FM Stepdrive) and the stepper motor are required for the operation of a stepper motor on the IM 174.

The basic structure of an IM 174 with stepper drives and without encoders is shown in the following figure:

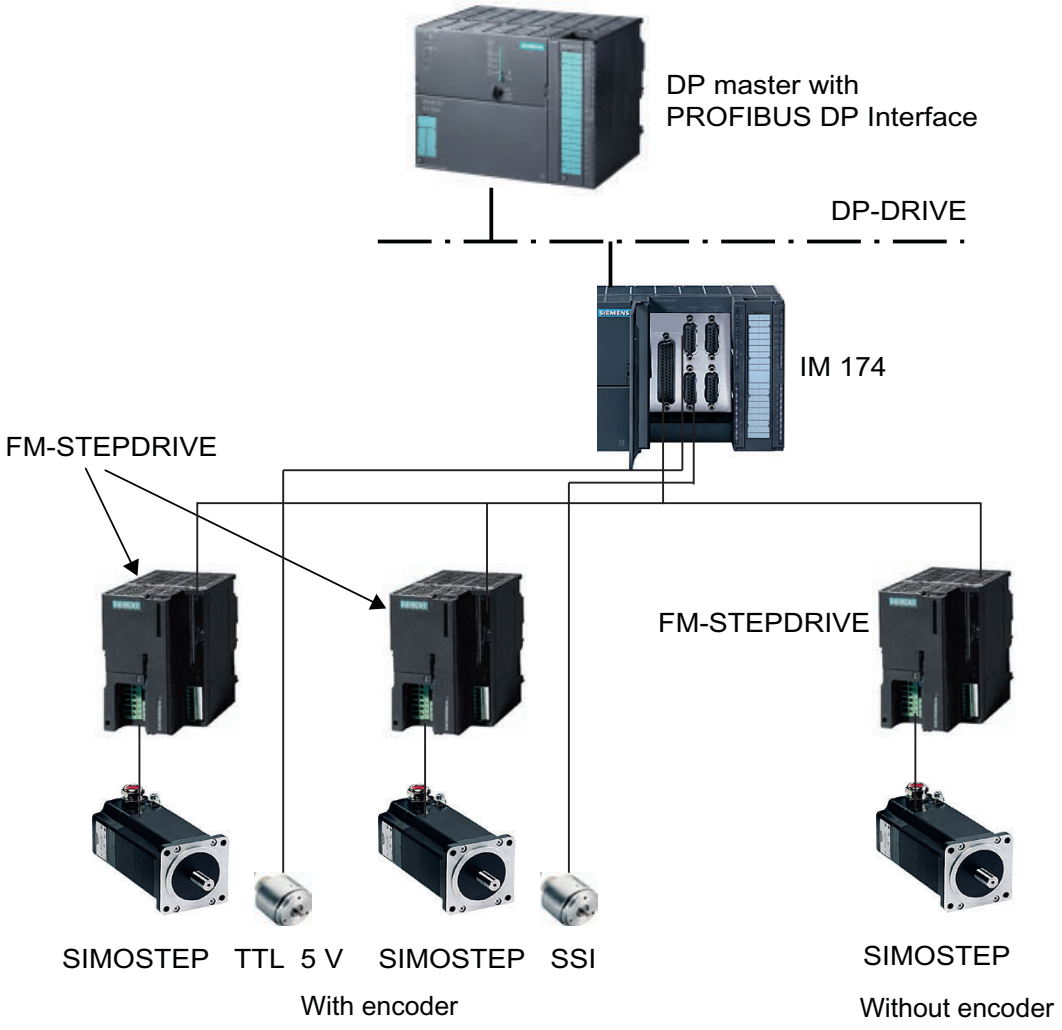

Figure 6-2 Basic structure of an IM 174 with FM STEPDRIVE and SIMOSTEP motor

Alt.DrvRdy

Depending on the drive you use, the Drive Ready signal returns different status messages:

- The drive signals "Ready" with the Drive Ready signal. An enable from the controller is still required to switch the drive on.
- The drive signals "Ready" with the Signal Drive Ready signal. The drive is ready for control in this state and directly follows a setpoint.

You should **not**activate the "Alt.DrvRdy" function, if your drive reports a "ready to switch on" state upon a Drive Ready signal (mostly in analogue drives).

Activate the "Alt.DrvRdy" function when the drive signals "Ready" with a Drive Ready signal (mostly with servo drives).

Information on which status message the drive you use returns can be found in the product information for your drive.

Max. frequency [Hz] and stand. frequency [Hz]

You must enter drive-specific or process-specific frequency values [Hz] in the fields "Max. frequency [Hz]" (maximum frequency) and "Stand. frequency [Hz]" (rated frequency) for the "Stepper" drive type.

Calculation of the stand. frequency

The stand. frequency is calculated according to the following formula:

Stand. frequency [Hz] = n [rpm] / 60 * resolution on the stepper motor

n [rpm]:	Speed of the stepper motor (characteristic values lie between 500 and 1000 rpm)
Resolution on the stepper motor:	Number of increments on the stepper motor (characteristic values: 500, 1000, 5000)

Calculation of the max. frequency

The max. frequency is calculated according to the following formula:

Maximum frequency [Hz] = n_{max} [rpm] / 60 * resolution on stepping motor

n_{max} [rpm]:	Maximum speed of the stepper motor (characteristic values lie between 500 and 1000 rpm)
Resolution on the stepper motor:	Number of increments on the stepper motor (characteristic values: 500, 1000, 5000)

The maximum speed n_{max} is determined by the operating characteristics of

stepper motors. Hence, a maximum motor speed nmax,

corresponding to a maximum axis speed,

is determined from the technology required torque M_t.

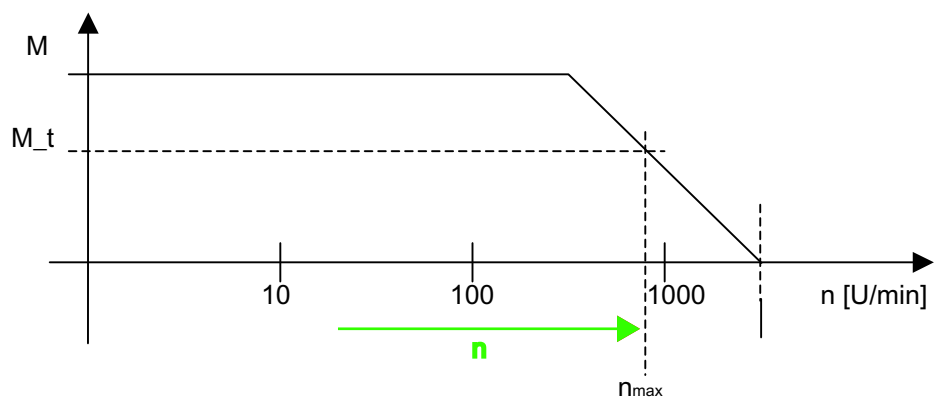

Figure 6-3 Characteristic operational features of a stepper motor

Note

Overload range

To avoid operating in the overload range, use the drive only at normal frequency!

Description of the behavior of a stepper motor

The positioning accuracy, the speed n as well as the torque M produced by the motor are very important for the process-specific requirements. For the optimum determination of these values, the behavior of the stepper motor must be considered. From a given speed of rotation of the stepper motor (around 500 rpm), the torque produced by the motor drops logarithmically, and approaches 0 at a maximum speed (around 3,000 rpm). The specific data can be found in the data sheet for the motor used.

Consequences of operation in the overload range

If the stepper motor is ever unable to produce the requested torque, it loses synchronization with the predefined frequency and its speed drops suddenly. This can lead to standstill. Motion can only be resumed in this state once a setpoint of 0 has been entered. For a position axis without additional encoders, the traversing position and thus synchronization of the axis are lost.

Note

Stepper drive with "Stepper" encoder type (PULSE REFEED operation)

During operation with a "Stepper" encoder type (PULSE REFEED operation), the stepper motor may vibrate at standstill.

This occurs with the default parameterization of a project when the stepper is traversed in a preceding traversing motion to a position value which is exactly between two natural step divisions of the stepper.

Stepper drive with TTL/SSI encoders

When using TTL/SSI encoders, the ratio of the encoder resolution (incl. fine resolution) to the step resolution of the stepper must be taken into account.

To prevent vibration between two position values at standstill, the stepper increment distance must be less than the distance representation in the encoder feedback (i.e. the increment of the stepper must be greater than the encoder resolution). If necessary, encoders with a lower resolution must be used or a higher step resolution set on the stepper.

Encoder parameters (S7-300, S7-400)

Encoder types for analog drives

The following encoder types can be selected for analog drives:

- Encoder type not available
- Encoder type TTL
- Encoder type SSI

Encoder types for stepper drives

The following encoder types can be selected for stepper drives:

- Encoder type not available
- Encoder type TTL
- Encoder type SSI
- Encoder type stepper

Encoder type "not available"

The following applies to **"Servo" drive types**: Axis x is not available or should not be operated. Useful data transmitted for this axis in the PROFIBUS message frame are empty.

The following applies to**"Stepper" drive types**: The stepper is operated without an encoder.

If the stepper is configured for operation with encoder, it may only be used as a speed-controlled axis.

If no encoder (SSI/TTL) is used, the encoder type "Stepper" can be selected to enable positioning even if no encoder is connected.

Encoder type TTL

In the case of encoder type "TTL" you can set the following encoder parameters:

- **Resolution**: Encoder resolution in encoder pulses per encoder revolution
- **Speed calculation enabled**: The IM 174 calculates the speed if the check box is enabled.

- **Normalization speed**: This input box is only visible if the "Speed calculation enabled" option is enabled. Enter the rated speed of the motor in revolutions per minute. The speed with the least noise is obtained when the speed is calculated by the IM174.
- **Reserved bits for fine resolution**: Setting 0 - 15
  The "Reserved bits for fine resolution" parameter specifies the desired pulse multiplication of the encoder increments transmitted in actual encoder values G1_XIST1 and G1_XIST2. The setting corresponds to a pulse multiplication between 2^0 = 1to 2^{15} = 32,768.

  Note

  Minimum fine resolution for TTL encoders

  TTL encoders always require considering and setting a minimum fine resolution of 2^2 (= 4).

Transmitted actual encoder value G1_XIST1 or G1_XIST2

31		0
Overflows	Current encoder increments	Reserved bits for fine resolution

Encoder type SSI

You can set the following parameters for encoder type "SSI":

- **Parity**: Select this option if the encoder data are to be transmitted from the encoder to the IM 174 with a parity bit.
- **MsgLength**: Setting 0 - 25
  Number of useful data bits transmitted by the encoder
- **Encoding**: The following encoder codes are supported:
  - Binary
  - Gray
- **Transmission rate**: The transmission rate setting must be identical for all SSI encoders. If transmission rate settings are different, the transmission rate of the SSI encoder with the highest encoder number is assigned.
  The following transmission rates are supported:
  - 187.5 Kbps
  - 375 Kbps
  - 750 Kbps
  - 1.5 Mbps
  - 3.0 Mbps

  Note

  Configure a maximum of 187.5 Kbps. It is not practical to set a quicker transmission rate for this module.

- **Reserved bits for fine resolution**: Setting 0 - 15
  The "Reserved bits for fine resolution" parameter specifies the desired pulse multiplication of the encoder increments transmitted in actual encoder values G1_XIST1 and G1_XIST2. The setting corresponds to a pulse multiplication between 2^0 = 1 to 2^{15} = 32,768.

Encoder type stepper

Stepper drive in PULSE REFEED mode

An Axis x with the "Stepper" drive type can be operated with a "Stepper" encoder type. In this mode, the setpoints are signaled back as process values from the IM 174 to the controller.

You can set the following encoder parameters for encoder type "Stepper":

- **Motor monitoring**: If the "Motor monitoring" check box is activated, the number of steps of a stepper motor must be constantly monitored for a preset reference point distance (see "BERO distance"). In the case of encoderless operation, the stepper drive used simultaneously signals the process values (pulse refeed).
- **Bero distance**: In the case of "BERO distance" the number of steps (for the stepper drive used) between two BERO signals is entered. The distance between the two BERO signals is the reference point distance.
- **Bero tolerance**: Setting 0 - 65535
  In the case of "BERO tolerance" the permissible deviation of the steps is entered. The resulting step range lies within the following tolerance:
  resulting step range = Bero distance ± 1 / 2 * Bero tolerance
- **Reserved bits for fine resolution**: Setting 0 - 15
  The "Reserved bits for fine resolution" parameter specifies the desired pulse multiplication of the encoder increments transmitted in actual encoder values G1_XIST1 and G1_XIST2. The setting corresponds to a pulse multiplication between 2^0 = 1to 2^{15} = 32,768.

Parameters (S7-300, S7-400)

Important parameters for the IM 174 are specified under the "Parameters" property in the area navigation.

shutdown ramp [ms]

The value range is 0 - 65535 ms.

The "Shutdown ramp" parameter specifies a function that is linear with respect to time. If an error is detected, all drives that are connected to the IM 174 are decelerated down to setpoint 0 in accordance with this function.

The shutdown ramp takes effect for the following errors:

- Temperature alarm of the module (approx. 90 °C ON, approx. 85 °C OFF)
- Lifebeat error
- Synchronization error between master and slave

Note

Parameter value 0

A parameter value of 0 causes the drive to coast to a standstill when an error occurs.

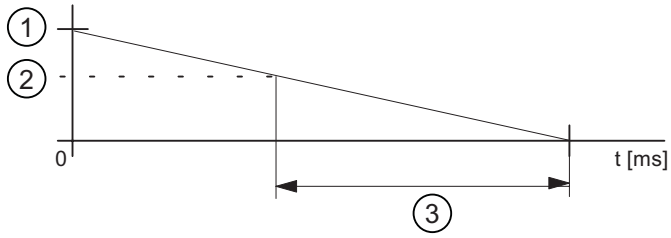

1) Maximum setpoint

2) Current setpoint

3) Parameter value: Shutdown ramp

Figure 6-4 Parameters: Shutdown ramp

Shutdown delay

The value range is 0 - 65535 s.

The "Shutdown delay" parameter can be used to specify a time after which, after a **temperature alarm** has occurred, the shutdown ramp is activated.

The shutdown delay is started when a temperature of 90 °C is exceeded in the module. If the temperature drops to approx. 85 °C while the delay is active, the shutdown ramp is not activated and the shutdown delay time is reset.

Tolerable lifebeat errors

The value range is 0 - 15.

The "Tolerable lifebeat error" parameter specifies the number of lifebeat errors tolerated for the DP master. The shutdown ramp is activated when the parameterized number is exceeded. The value range is

611U-compliant mode

In 611U-compliant mode, the signal source for homing of axes is no longer specified using the PROFIDrive standard message frame (STD3, encoder control word G1_STW), but rather using the additional digital output word in the PROFIBUS message frame of the IM 174 (see table "Message frame structure" in the "Message frame type" section).

611U-compliant mode:

- Not selected
  The signal source for homing is specified via the encoder control word Gx_STW in the PROFIDrive standard message frame.
- Selected
  The signal source for homing is specified via the additional digital output word in the PROFIBUS message frame.

Selecting the message frame type (S7-300, S7-400)

Message frame type

The IM 174 DP slave is operated with two specific message frame types: Four axes, each with one encoder (standard message frame 3 and 81) and I/O data

Note

Standard message frame 81

Standard message frame 81 can only be used for encoders.

Message frame structure (S7-300, S7-400)

Message frame structure

The overview below explains the frame structure.

Table 6-1

Message frame type	Description
4 axes, each with one encoder, standard message frame 3 + IO, PZD-5/9 O/I 1/1 or message frame 81 + IO, PZD-2/6 + I/O 1/1	4 x Standard message frame 3 or 81 and 1 PZD word each for digital I/O data
PZD x/y Number of process data words, x: Setpoint, y: Actual value, for example, PZD-5 / 9: 5 process data words to desired values 9 process data words for actual values	

IM174 message frame structure

	Axis1	Axis2	Axis3	Axis4	I/O	
	STD 3 or STD 81	STD 3	STD 3	STD 3	O word	Setpoints (master -> slave)
Low	Axis1	Axis2	Axis3	Axis4	I/O	High
	STD 3 or STD 81	STD 3	STD 3	STD 3	I word	Actual values (slave -> master)

STD 3: standard message frame 3 per PROFIDrive V3.0
Q word: digital output data (16 bits)
I word: digital input data (16 bits)

Message frame type	Description

Standard message frame 3: speed setpoint interface 32 bits with 1 encoder

PZD1	PZD1	PZD3	PZD4	PZD5	
STW1	NSOLL_B		STW2	G1_STW	Setpoint (master -> slave)

Low High

PZD1	PZD2	PZD3	PZD4	PZD5	PZD6	PZD7
ZSW1	NIST_B		ZSW2	G1_ZSW	G1_XIST1	

PZD8	PZD9	
	G1_XIST2	Actual value (slave -> master)

Standard message frame 81: speed setpoint interface 32 bits with 1 encoder

PZD1	PZD2	
STW2	G1_STW1	Setpoint (master -> slave)

PZD1	PZD2	PZD3	PZD4	PZD5	PZD6	
ZSW2	G1_ZSTW1	G1_XIST1		G1_XIST2		Actual value (slave -> master)

O word (dig. output data 16 bits)

High-Byte		Low-Byte	
15 12	11 8	7 4	3 0

- 3 0: Dig. outputs 1–4 –> X11: Pin 2,4,6,8
- 7 4: Dig. outputs 5 to 8 / direction signal 1 to 4 for unipolar spindle –> X11: Pin 13,15,17,19
- 11 8: Selection: Homing using external zero mark signals 1 to 4
  611U conformant mode
- 15 12: Not used

I word (dig. input data 16 bits)

High-Byte		Low-Byte			
15 10	9 8	7 6	5 4	3 0	

- 3 0: Dig. inputs 1–4 / ext. zero marks 1 to 4
  ->X11: Pins 22-25
- 5 4: Dig. inputs 5–6 / measuring inputs 1–2
  ->X11: Pins 26-27
- 7 6: Dig. inputs 7–8 / Drv_Rdy 1–2
  ->X11: Pin 31, 32
- 9 8: Dig. inputs 9–10 / Drv_Rdy 3–4
  ->X11: Pin 33, 34
- 15 10: Not used

Note

The message frame type setting for the IM 174 DP slave must agree with the message frame type setting in the controller. There is no automatic adjustment.

Encoder control word Gx_STW

The overview below explains the encoder control word (extract). The overview is based on the following requirements:

- Find reference mark
- On-the-fly measurement
- Encoder error

Bit	Identifier		Signal state/description		
0	Find reference mark or On-the-fly measurement	Functions	Bit 7 = 0 => Request: Find reference mark		
			Bit	Meaning	Homing using:
			0	Function 1:	Encoder zero mark (except in "611 U conform mode")
			1	Function 2:	Not used
			2	Function 3:	Not used
			3	Function 4:	Not used
1			Bit 7 = 1 => Request: On-the-fly measurement		
			Bit	Meaning	
			0	Function 1:	Measuring input 1 rising edge
			1	Function 2:	Measuring input 1 falling edge
			2	Function 3:	Measuring input 2 rising edge
			3	Function 4:	Measuring input 2 falling edge
2			Note • Bit x = 1 Function requested Bit x = 0 Function not requested • If more than one function is enabled, the values for all functions cannot be read until all functions have ended and this has been signaled via the relevant status bit (G1_ZSW, Bit 0 - Bit 3 = 0). • On-the-fly measurement The rising and falling edges of the measuring input can be enabled simultaneously. The measuring input signal is detected according to the direction of the signal change. The measured values are read out consecutively.		
3					
4		Command	Bits 6, 5, 4	Meaning	
5			000	--	
6			001	Enable function x	
			010	Read value x	
			011	Cancel function x	
7		Mode	0	Find reference mark	
			1	On-the-fly measurement	
15	Acknowledge encoder error		0 1	-- Acknowledge encoder error	

Note

Measurement on rising or falling edge

IM 174 only supports measurement on a rising or falling edge.

Additional encoder actual value Gx_XIST2

The overview below sets out the error codes in Gx_XIST2 where G1_ZSW, Bit 15 == 1:

Table 6-2 Error codes in Gx_XIST2

G1_XIST2	Meaning	Possible causes / description
1_{Hex}	Encoder sum error	The encoder signal levels are too low, faulty (inadequate shielding) or open-circuit monitoring has been tripped.
2_{Hex}	Zero mark monitoring	A fluctuation in the measured rotor position has arisen between 2 encoder zero marks (encoder pulses may be lost).

Status word ZSW_1 - Bit 11 to Bit 14

The following overview describes the status word ZSW_1 with regard to the module-specific messages Bit 11 to Bit 14:

Table 6-3 Status word ZSW_1

Bit	Name	Description
11	Temperature error	A specific temperature in the housing of module IM 174 has been exceeded.
12	PLL synchronization error	The IM 174 cannot be synchronized with the global check-back signal.
13	Master life sign error	The IM 174 cannot be synchronized with the master life sign signal.
14	Drive error	A drive error exists, for example the Drive Ready signal is missing.

Zero mark monitoring (S7-300, S7-400)

Method of functioning of the zero reference mark monitoring

Zero mark monitoring compares the recognized encoder increments between two zero marks with the configured encoder resolution. If a difference is found, an encoder error is reported.

Track A and Track B as well as Track Z of the encoder have to be on the high level simultaneously so that zero mark monitoring is carried out correctly.

Restrictions:

- The failure of a zero mark in itself does not inevitably result in an error, since checking is not started until a zero mark is reached.
- The result of the counted encoder increments MOD 16 has to be nonzero so that an error is reported.
- The result of the counted encoder increments MOD 10 has to be nonzero so that an error is reported.

Example:

Encoder configuration: TTL 2048 pulses

Counted encoder increments 2049 -> encoder error

Counted encoder increments 1024 -> no encoder error

Programming the IM 174 (S7-300, S7-400)

We recommend the "Easy Motion Control" add-on package for programming the IM 174.

Instructions for IM 174

The following instructions are available for IM 174 in the Easy Motion Control instruction list:

- EncoderIM174 - Use of IM 174 as position feedback module for incremental and absolute encoders
- OutputIM174 - Use of a channel of module IM 174 as output module

6.2.10 Using Easy Motion Control (S7-300, S7-400)

6.2.10.1 Positioning with Easy Motion Control (S7-300, S7-400)

Product overview (S7-300, S7-400)

Overview

Easy Motion Control allows you to control the positioning of drives. The movement is controlled solely by software (instructions) in the CPU. **No special hardware** is required for connection to the drives, although corresponding I/O must of course be available in order to record actual positions and output speed setpoint values.

The instructions of Easy Motion Control were created based on the "Technical Specification Function blocks for motion control V1.0" of **PLCopen**.

You assign parameters using configuration software that also supports commissioning using test functions. All parameters of an axis are stored in a single data block, namely the axis DB.

You can connect the drives to central I/O, or to distributed I/O via a DP master / IO controller that is integrated in the CPU.

You execute the movement by interconnecting the corresponding instructions and running them in the user program.

Components for Positioning (S7-300, S7-400)

The following figure shows a configuration model with Easy Motion Control.

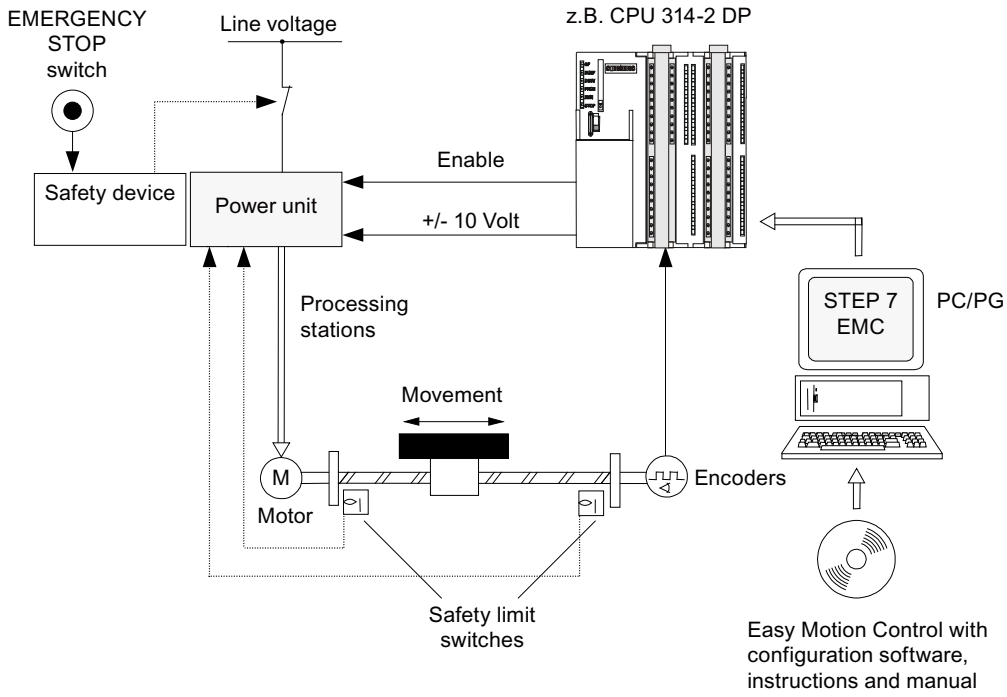

Safety device

When the safety device is actuated, the emergency STOP button disconnects the power supply.

Power unit

The power unit controls the motor. The safety limit switches disconnect the power unit.

Motor

The motor is controlled by the power unit and drives the axis.

Encoder

The encoder provides information on distance and direction.

Easy Motion Control

Easy Motion Control provides the following functions:

- Jogging
- Reference point approach
- Absolute/relative positioning
- Electronic gear
- Set reference point
- All common monitoring functions for positioning
- Velocity override
- Closed-loop position control
- Simulation
- Preconfigured input drivers for recording encoder values, or output drivers for analog output modules

Modules for position detection and setpoint output

These modules form the interface between the CPU and the machine. They read axis positions and output setpoint values to the power unit.

CPU

The CPU executes the user program that contains the embedded instructions of Easy Motion Control.

PC/PG

A PC/PG is used for the following functions:

- Parameter assignment: You assign parameters to Easy Motion Control using the configuration software, or directly in the axis DB (see Configuring the axis TO (Page 2719)).
- Programming: You integrate the Easy Motion Control instructions directly in your program.

Supported hardware configurations (S7-300, S7-400)

The following hardware configurations are conveniently supported by driver instructions. These are supplied with Easy Motion Control.

You can adapt configurations that are not directly supported using the open driver blocks that are also provided in your package.

The following figure shows **distributed** configurations for Easy Motion Control.

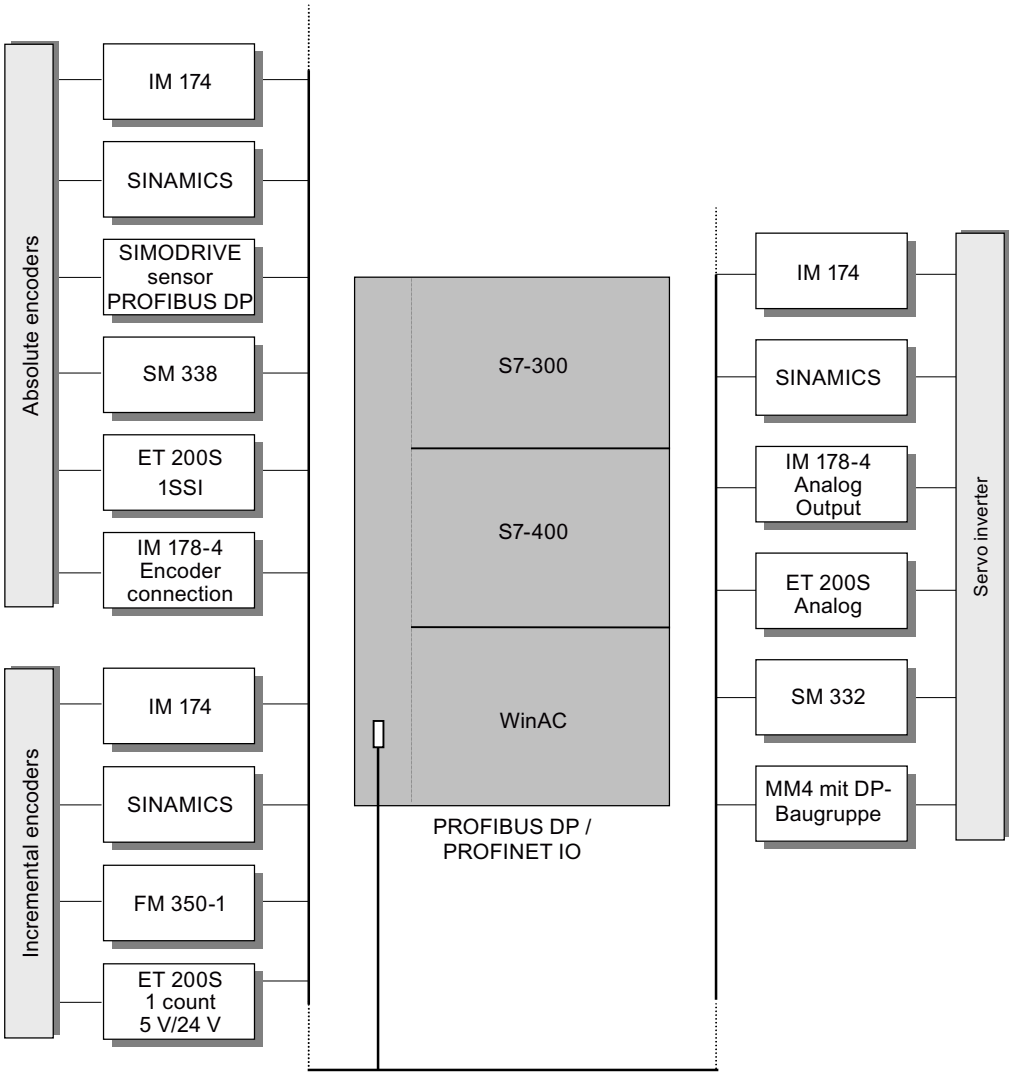

The following figure shows **centralized** configurations for Easy Motion Control.

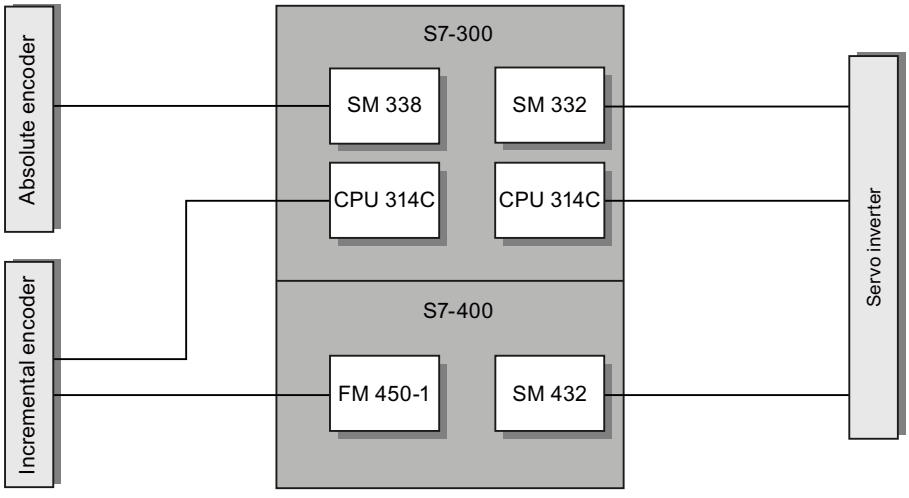

Principles of Positioning with Easy Motion Control (S7-300, S7-400)

Setting up closed loop position control

With controlled positioning, the current position of the axis is compared with the setpoint position. Any deviations are compensated using the position controller both during a travel movement as well as at a standstill.

The following figure shows the schematic representation of controlled positioning with Easy Motion Control. A travel instruction (Page 2743) specifies the position setpoints. The input driver records the actual value of the axis position. The position controller (Page 2730) compares the actual position value with the position setpoint and generates a velocity setpoint value from the difference. The output driver outputs the scaled velocity setpoint.

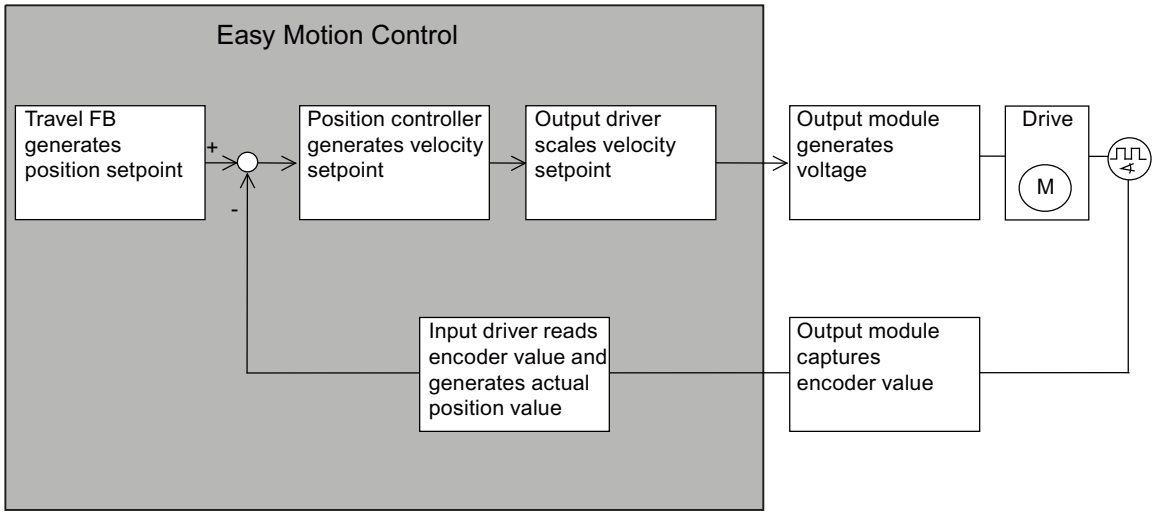

The following figure shows the relationship between the axis velocity and the distance covered during a traversing movement.

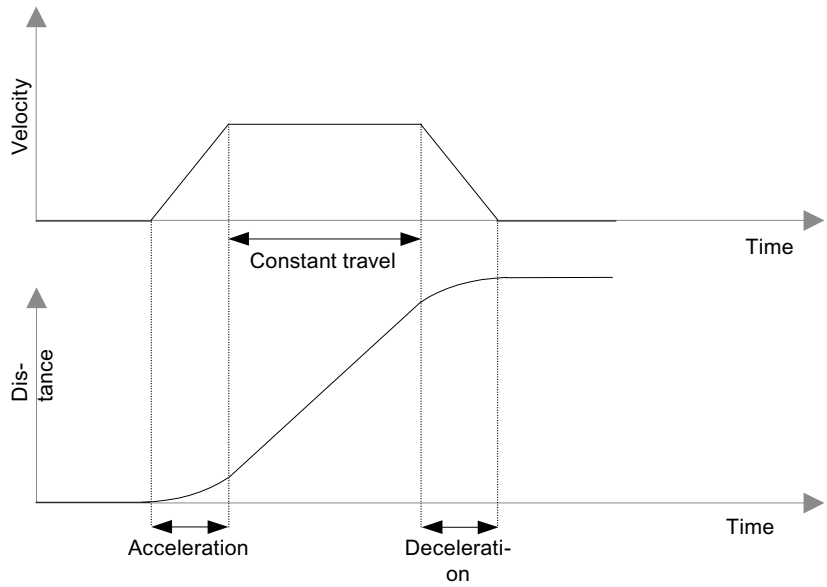

Concept Definitions (S7-300, S7-400)

Target

The target is the absolute or relative position of the axis that is approached during positioning.

Traversing movement

Traversing movements are triggered by calling instructions, which are also referred to as travel instructions in the following.

Residual distance

Residual distance is the difference between target position and current position setpoint.

Following distance / following error

During a travel movement, a difference builds up between the current position setpoint and the actual position value due to axial inertia. This is called the following error. If the following distance exceeds an assignable value, a following error occurs.

Monitoring ranges

- Work range:
  Defines the permitted traversing range of your axis, which you determine for your task using the software limit switches (Page 2713), or the end of the rotary axis (Page 2714).
- Standstill range:
  Defines a configurable range that the axis must not leave when at a standstill. The standstill range lies to the left and right of the target.
- Target range:
  If the setpoint has reached the target at the end of a motion, the approach of the axis into the target range is monitored. The target range is used to monitor the positioning accuracy of your application and lies to the left and right of the target.
  Standstill range monitoring begins as soon as the axis reaches the target range. For this reason, the standstill range should be greater than the target range.

The following figure shows the position of the ranges in the target area.

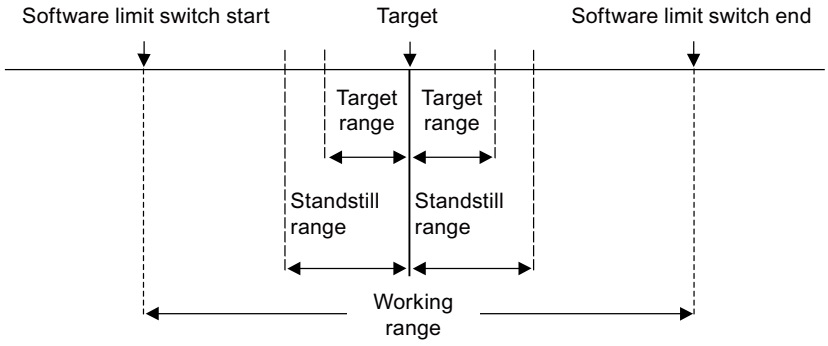

Linear axis (S7-300, S7-400)

Description

The **working range** extends from the start of the software limit switch to the end of the software limit switch and is limited to 1 × 2^{24} length units (Cartesian coordinates between -2^{24}+6 and +2^{24}-6).

The **physical axis length** is limited by the mechanical stops of your axis.

The **numerical range** of Easy Motion Control is only relevant if work range monitoring is switched off. It is limited to 2 × 2^{24}-12 length units (from -2^{24}+6 to +2^{24}-6).

The **encoder range** is:

- The range covered by an absolute encoder
- The distance from -2^{31} to +2^{31} steps covered by an incremental encoder

With linear axes, the **work range** (= travel range) is limited as follows:

Working range < physical axis length < numerical range of Easy Motion Control < encoder range

The following figure shows the various ranges of a linear axis.

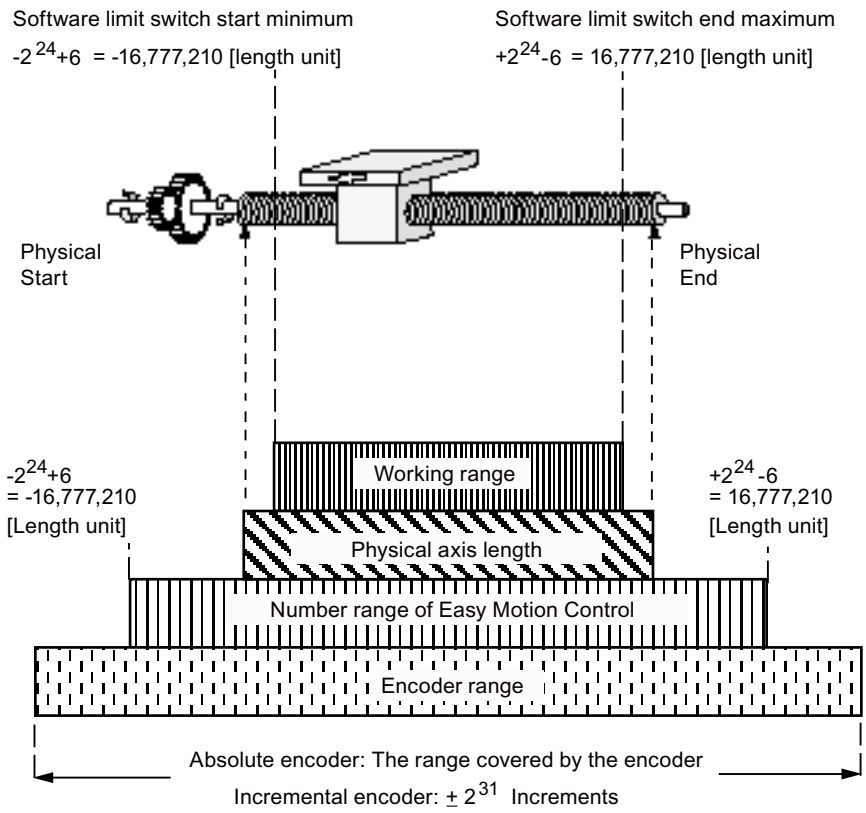

Rotary axis (S7-300, S7-400)

Description

The **working range** extends from the start of the rotary axis to the end of the rotary axis and is limited to 1 × 2^{24} length units (Cartesian coordinates between $-2^{24}+6$ and $+2^{24}-6$).

The **encoder range** is:

- Arbitrary for an incremental encoder
- Determined for an absolute encoder based on the rule:
  Encoder range = working range × n
  where:
  - n = 1 with single-turn encoders
  - n > 1 and integer with multi-turn encoders

The actual value of a rotary axis is reset to the start of the rotary axis after every revolution. Rotary axes therefore have an infinite traversing range that extends beyond the start or end of the rotary axis. The axis position is always displayed as a value between the start and end of the rotary axis.

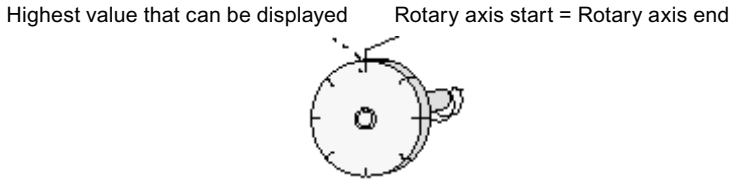

6.2.10.2 Functions with Easy Motion Control (S7-300, S7-400)

Overview (S7-300, S7-400)

Easy Motion Control is used to position an axis on a target with programmable acceleration, velocity, and deceleration or move it at constant velocity. The necessary traversing movements are triggered by calling travel instructions.

At the start of a traversing job, the target or distance of the traversing movement and possibly the required acceleration, velocity, or deceleration, are configured at the travel instruction. The system of units (length unit) can be selected freely for these values (for example, mm).

Jogging (S7-300, S7-400)

The jogging function allows you to move the drive in the positive or negative direction. No target is specified.

The jogging function is realized using the MC_MoveJog instruction.

Synchronization (S7-300, S7-400)

In order to assign a reproducible encoder value to the real position, a reference (synchronization) must be established between the axis position and the encoder value.

The synchronization function is realized using the MC_Home instruction.

Synchronization with an Incremental Encoder

An incremental encoder does not give any information about the immediate position of the axis when the system is switched on. Instead, it starts counting at the current position of the axis.

Synchronization is always necessary when the system is switched on again after occurrence of errors or after replacing the encoder.

An incremental encoder can be synchronized by:

- Reference point approach
- Set reference point

Synchronization with absolute encoder

When commissioning absolute encoders, you must synchronize the value indicated by the encoder with the actual axis position.

Resynchronization is not necessary when the system is switched back on or after errors have occurred. After the encoder has been replaced, however, resynchronization is necessary.

An absolute encoder can only be synchronized by means of reference point setting (also called absolute encoder adjustment).

Reference point approach

The axis is moved in the direction of the reference point. The axis is synchronized after it has passed the reference point, i.e. a position value is assigned to the numerical value of the incremental encoder. A new coordinate value can be assigned to the reference point with every reference point approach.

A reference point approach is only possible with incremental encoders.

Set reference point

With reference setting, a new coordinate value is assigned to the current position.

Reference point setting is supported for incremental and absolute encoders.

Absolute or Relative Positioning (S7-300, S7-400)

Easy Motion Control can move the axis

- To **absolute** targets
- By a given distance in a specific direction with **relative** positioning

The absolute and relative positioning functions are realized using MC_MoveAbsolute and MC_MoveRelative instructions.

Gear (S7-300, S7-400)

Easy Motion Control realizes an electronic gear between two axes using the MC_GearIn instruction.

Stopping Travel (S7-300, S7-400)

You can stop current travel at any time.

The Stop function is realized using the MC_StopMotion instruction.

Overriding Travel (S7-300, S7-400)

A currently active travel instruction can always

- Be overridden by starting a different travel instruction
- Be canceled by calling instruction MC_StopMotion (see Sequence of traversing movements (Page 2749)).

Tracking (S7-300, S7-400)

With tracking, the position setpoint is continuously adjusted (tracked) in the position controller to the actual position value. During tracking, the axis is not monitored for leaving the standstill range.

This function is activated by clearing the "EnableDrive" signal at instruction MC_Control, or after an error has occurred.

Velocity override (S7-300, S7-400)

Velocity override enables dynamic changes to the velocity that is configured in a travel instruction. Acceleration and deceleration values are not affected.

For more information on velocity override, refer to Specifying velocity override (Page 2735).

Simulation (S7-300, S7-400)

In simulation mode, you can test your travel program without moving your axis.

Simulation is realized using the MC_Simulation instruction.

Error Responses and Diagnosis (S7-300, S7-400)

All errors detected by the instructions are displayed in the axis DB by

- Setting a group error and
- Error-specific display

With errors requiring acknowledgment, an error acknowledgment must be set after rectifying the error.

If a travel instruction (for example, MC_MoveJog, or MC_MoveAbsolute) has caused or detected the error, the "Error" output is set at the corresponding block.

However, if the error was detected by a driver block, the exact cause of the error is indicated in the static data of the block (see Input drivers).

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB (Page 2764).

6.2.10.3 Configuring Easy Motion Control (S7-300, S7-400)

Guideline for using Easy Motion Control (S7-300, S7-400)

This guideline shows you the basic procedures for using Easy Motion Control in combination with an S7-300/400 CPU.

Requirement

To use the "Axis" technology object (TO), you need to create a project with a CPU.

Procedure

Proceed based on the sequence recommended below to use Easy Motion Control with the S7-300/400 CPU. Use the following links for this purpose:

1. Adding an axis technology object (Page 2718)
2. Working with the configuration dialog (Page 2719)
3. Download to CPU (Page 2741)
4. Function test of the axis in the commissioning window
5. Programming Easy Motion Control (Page 2743)
6. Diagnostics of the axis control

Adding an axis technology object (S7-300, S7-400)

Inserting an axis TO in the project navigator

When you add a technology object, an instance DB of this TO will be generated. The axis configuration is stored in this instance DB.

Requirement

A project with a CPU has been created.

Procedure

Proceed as follows to insert an axis TO in the project navigator:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click "Add new object".
   The "Add new object" dialog opens.
4. Select the "Motion" technology.
5. Open the "Motion Control" folder.
6. Open the "Easy Motion Control" folder.
7. Select the instruction for the technology object: in this case, AXIS_REF.
8. Type in an individual name for the axis in the "Name" text box.
9. Select the "Manual" option if you want to change the proposed data block number of the instance DB.
10. Click "More information" if you want to supplement user information on the TO.
11. Click "OK" to add the TO.
    Click "Cancel" if you want to discard your entries.

Result

The new technology object has been created and stored in the project tree in the "Technology objects" folder. The technology object is used if the instructions for this TO are called in a cyclic interrupt or synchronous cycle interrupt OB.

For more information, refer to "User Program Setup (Page 2743)".

Note

You can select the "Add new and open" check box at the bottom of the dialog box. This opens the configuration of the technology object after adding has been completed.

Configuring the axis TO (S7-300, S7-400)

Working with the configuration dialog (S7-300, S7-400)

You configure the properties of the technology object in the configuration window. To open the configuration window of the technology object, follow these steps:

1. Open the group of the selected technology object in the project tree.
2. Double-click the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters contain all the parameters which must be configured for a functioning axis.
- **Advanced parameters**
  The advanced parameters include parameters for adapting to your drive or plant.

Icons of the configuration window

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values you can use the technology object without additional changes.
	The configuration contains values set by the user and is complete. All input fields of the configuration contain valid values and at least one preset value has changed.
	The configuration is incomplete or incorrect At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to indicate the cause of error.
	The configuration is valid but contains warnings Only one hardware limit switch is configured. Depending on the plant, the lacking configuration of a hardware limit switch may result in a hazard. The corresponding field or the drop-down list is displayed on a yellow background.

Basic parameters (S7-300, S7-400)

Naming the axis (S7-300, S7-400)

Axis name

Define the name of the axis, or the name of the "Axis" technology object, in this field. The technology object is listed under this name in the project navigator.

Selecting axis type (S7-300, S7-400)

Axis types

Easy Motion Control is able to control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.

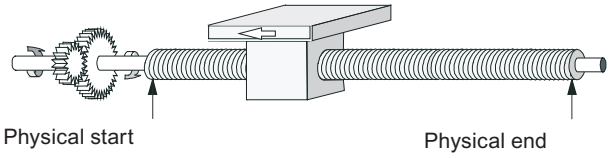

- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.

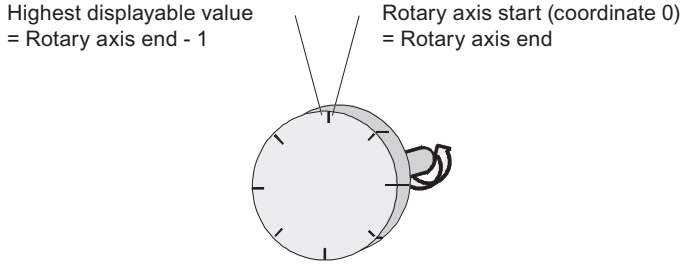

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
4.0	AxisType	BOOL	FALSE	**Axis type** Valid values: • FALSE = Linear axis • TRUE = Rotary axis	I

Selecting length units (S7-300, S7-400)

Length unit

Select the length unit for the axis unit system from the drop-down list. The selected unit will be used for further configuration of the "Axis" TO and for visualization of the current axis data.

You can select a length unit for data input and output from the following:

- Millimeter (default setting)
- Degree
- Inch

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this unit.

Note

A change of the length unit will not affect the values of the parameters. There, is no conversion.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
196.0	LengthUnit	STRING [4]	mm	**Length unit** for configuration software Valid values: • mm (default) • Degree • Inch

Activating simulation mode (S7-300, S7-400)

Simulation mode

Select the check box if you want to test your traversing program without traversing the axis.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
4.2	Sim	BOOL	FALSE	**Simulation mode** Valid values: • FALSE = Deactivated • TRUE = Activated

Advanced parameters (S7-300, S7-400)

Driver interfaces (S7-300, S7-400)

Input driver (S7-300, S7-400)

Selecting an input module (S7-300, S7-400)

Input module

Select the input module to which your encoder is connected from the drop-down list.

Based on your selection, the "Instruction" field displays the name of the instruction that you have to call for actual value acquisition.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
202.0	InputModuleType	STRING [24]	IM 174	Input driver **for the module**. It is not allowed to overwrite this parameter in manual mode.

Selecting the encoder type (S7-300, S7-400)

Encoder type

Select the check box for the encoder used.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
4.1	EncoderType	BOOL	FALSE	**Encoder type** Valid values: • FALSE = Incremental encoder • TRUE = Absolute encoder	

Input module: Module address inputs (S7-300, S7-400)

Module address inputs

Enter the value specified for the input module in your hardware configuration in the text box.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
58.0	InputModuleInAddr	INT	0	**Start address of the inputs of the position decoder module** Range: • Determined by the hardware configuration	I

Input module: Module address outputs (S7-300, S7-400)

Module address outputs

The parameter is used only for modules for which it is possible to set the "Module address outputs" separately from the "Module address inputs" in the hardware configuration.

Enter the value specified for the input module in your hardware configuration in the text box.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
60.0	InputModuleOutAddr	INT	0	**Start address of the outputs of the position decoder module** Range: • Determined by the hardware configuration	I

Input module: Channel number (S7-300, S7-400)

Channel number

- Single-channel modules: The default value "0" that is set in the text box cannot be edited.
- Multi-channel modules: Enter the number of the channel used in this text box. Enter "0" for the first channel, etc.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
62.0	InputChannelNo	INT	0	**Channel number of the position input module** Range: • Depends on the module	I

Output drivers (S7-300, S7-400)

Selecting output modules (S7-300, S7-400)

Output module

Select the output module to which your drive is connected from the drop-down list.

Based on your selection, the "Instruction" field displays the name of the instruction that you have to call for speed setpoint output.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
228.0	OutputModuleType	STRING [24]	IM 174	Output driver **for the module**. It is not allowed to overwrite this parameter in manual mode.

Output module: Module address inputs (S7-300, S7-400)

Module address inputs

The parameter is used only for modules for which it is possible to set the "Module address inputs" separately from the "Module address outputs" in the hardware configuration.

Enter the value specified for the output module in your hardware configuration in the text box.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
82.0	OutputModuleInAddr	INT	0	**Start address of the inputs of the output module** Range: • Determined by the hardware configuration	I

Output module: Module address outputs (S7-300, S7-400)

Module address outputs

Enter the value specified for the output module in your hardware configuration in the text box.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
80.0	OutputModuleOutAddr	INT	0	**Start address of the outputs of the module outputs** Range: • Determined by the hardware configuration	I

Output module: Channel number (S7-300, S7-400)

Channel number

- Single-channel modules: The default value "0" that is set in the text box cannot be edited.
- Multi-channel modules: Enter the number of the channel used in this text box. Enter "0" for the first channel, etc.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
84.0	OutputChannelNo	INT	0	**Channel number of the output module** Range: • Depends on the module	I

Mechanics (S7-300, S7-400)

Encoders (S7-300, S7-400)

Specifying the steps per encoder revolution (S7-300, S7-400)

Steps per encoder revolution

In the text box, enter the number of steps that your position decoder module outputs per encoder revolution.

The number of steps is converted to the length unit based on this value and the "Axis distance per encoder revolution" parameter (axis DB address 68.0 (Page 2728)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
64.0	StepsPerRev	DINT	L#1	**Steps per encoder revolution** Valid range of values: • ≥ 1 [steps/encoder revolution]	I

Incremental encoder

If the incremental encoder is read using a module that evaluates the pulses in a two-step or four-step operation, this has to be taken into account when entering the parameters.

Example:

- Encoder pulses per revolution: 500
- 4-step evaluation

In the "Steps per encoder revolution" text box, enter: 2000

Absolute encoder

The number of steps per encoder revolution is usually a power of two for absolute encoders.

Common values are:

- With single-turn encoders: 4096 and 8192
- With multi-turn encoders (12 × 12 bits in the 25-bit frame): 4096

If there, is any doubt, the correct value can be determined by moving the encoder one revolution and observing the change of the encoder value in the configuration software.

With a linear scale, enter the total number of steps for "Steps per encoder revolution" corresponding to the length of your linear scale.

Specifying the number of encoder revolutions (S7-300, S7-400)

Number of encoder revolutions

In the text box, enter the number of encoder revolutions for an absolute encoder.

Enter "1" for a single-turn encoder.

With a **linear scale**, enter "1" for "number of encoder revolutions".

This parameter is irrelevant for incremental encoders.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
72.0	NumberRevs	INT	1	**Number of revolutions** of an absolute encoder Valid range of values: • > 0	I

Specifying the axis distance per encoder revolution (S7-300, S7-400)

Axis distance per encoder revolution

In the text box, enter the distance traversed by the axis that is equivalent to one encoder revolution. This value depends on the structure of the axis and on the encoder mounting position. You must make allowances for all transmission elements, such as couplings or gears.

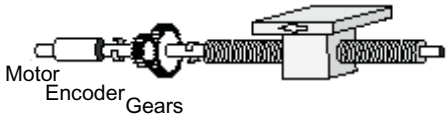

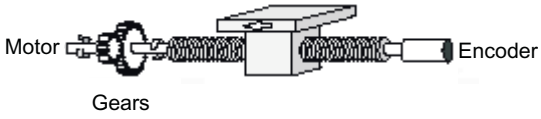

For a **linear scale**, enter the length of your linear scale for "Axis travel per encoder revolution".

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
68.0	DisplacementPerRev	REAL	1.0	**Axis distance per encoder revolution** Valid range of values: • > 0.0 [length unit/encoder revolution]	I

Special features with rotary axes

The encoder must perform an integer number of steps per rotary axis revolution:

$$\frac{\text{(Rotary axis end - rotary axis start)}}{\text{Axis distance per encoder revolution}} \times \text{Steps per encoder revolution} = \text{integer}$$

Special features for rotary axes with absolute encoders

To ensure the reproducibility of all positions, the axis traversing distance covered by the encoder must be an integer multiple of the distance that the rotary axis covers in one rotation:

$$\frac{\text{(Number of encoder revolutions) x (Axis distance per encoder revolution)}}{\text{(Rotary axis end - Rotary axis start)}} = \text{Integer}$$

Example:

24-bit absolute encoder

=> Number of encoder revolutions = 4096

Rotary axis start = 0.0°

Rotary axis end = 360.0°

=> Rotary axis end - rotary axis start = 360.0°

Axis distance per encoder revolution = 45°

$$\frac{4096 \text{ x } 45°}{360°} = 512 \text{ (integer)}$$

Selecting the encoder polarity (S7-300, S7-400)

Set encoder polarity

Here, you adjust the position feedback direction to the movement direction of the axis:

- Positive = Incremental count pulses (incremental encoder), or encoder values (absolute encoder) represent increasing actual position values
- Negative = Incremental count pulses (incremental encoder), or encoder values (absolute encoder) represent descending actual position values.

Select the corresponding option from the drop-down list.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
74.0	PolarityEncoder	INT	1	**Set encoder polarity** Valid values: • +1 or -1	I

Controller (S7-300, S7-400)

Specifying controller gain (S7-300, S7-400)

Controller gain

The optimum controller gain can be determined empirically on the axis.

Increase controller gain in increments of 1.0 until the axis begins to oscillate while traversing, or while it is at a standstill. If this is the case, reduce controller gain until this tendency to oscillate is no longer perceptible.

Enter the corresponding value in the text box.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
44.0	FactorP	REAL	1.0	**Controller gain** Valid range of values: • > 0.0 [1/s]

Specifying the velocity setpoint in manual mode (S7-300, S7-400)

Velocity setpoint in manual mode

Enter a velocity setpoint for manual mode in this text box.

When manual mode is activate, the controller outputs the value in "Manual setpoint velocity" as a manipulated variable and limits it to "MaxVelocity". The position setpoint is tracked to the actual position value.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
48.0	ManVelocity	REAL	0.0	**Velocity setpoint in manual mode** Valid range of values: • ≤ ± MaxVelocity (Page 2735) [length unit/s]
52.0	ManEnable	BOOL	FALSE	**Activating manual mode** Valid values: • FALSE = Deactivated • TRUE = Activated

Motor (S7-300, S7-400)

Reference value for 100 % speed (S7-300, S7-400)

Reference value for 100 % speed

This parameter is irrelevant when you are operating power units with a control voltage of ±10 V.

When you use the OutputMM4_DP output driver (bus-controlled MICROMASTER DP module), set the frequency that is needed to accelerate the drive to 100% of its speed at this parameter.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
96.0	DriveInputAt100	REAL	10.0	**Reference value for 100 % speed** Valid range of values: • -10.0 to +10.0 [V]	I

Example

- Rated speed (100 %) of the motor: = 1380 rpm
- Motor is synchronized to: = 1380 rpm at 50 Hz

=> Enter "Reference value for 100% speed = 50 Hz

Specifying the reference value for maximum axis velocity (S7-300, S7-400)

Reference value for maximum axis velocity

If using power unit with a control voltage of **±10 V**, enter the voltage value at which the drive reaches "Maximum velocity" in the text box (see Axis DB address 14.0 (Page 2735)).

Example:

- Rated speed of the motor: = 3000 rpm
- Motor is synchronized to: = 9 V at 3000 rpm
- Maximum speed to reach: = 1500 rpm

=> Enter the following value for "Reference value for maximum axis speed": = 4.5 V

If using the **OutputMM4_DP** output driver, enter the voltage value at which the drive reaches "Maximum speed" in the text box (see Axis DB address 14.0 (Page 2735)).

Example:

- Rated speed of the motor: = 1380 rpm
- Motor is synchronized to: = 1380 rpm at 50 Hz
- Maximum speed to reach: = 690 rpm

=> Enter the following value for "Reference value for maximum axis speed": = 25 Hz

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
88.0	DriveInputAtMaxVel	REAL	9.0	**Reference value for max. axis velocity [V]** Valid range of values: • Depends on the power unit	I

Specifying zero offset compensation (S7-300, S7-400)

Zero offset compensation

Use this parameter to implement zero offset compensation if your drive does not provide this option. The value configured here is added to the output value.

The OutputMM4_DP output driver ignores this parameter.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
92.0	OffsetCompensation	REAL	0.0	**Zero offset compensation** Valid range of values: • -10.0 to +10.0 [V]

Selecting the drive polarity (S7-300, S7-400)

Set drive polarity

Here, you select the direction of drive movement:

- Positive = Positive velocity setpoints initiate movements towards the positive direction.
- Negative = positive velocity setpoints initiate movements towards the negative direction.

Select the corresponding option from the drop-down list.

If you change "Set drive polarity", you must also change "Set encoder polarity" (Axis DB address 74.0 (Page 2728)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
86.0	PolarityDrive	INT	1	**Steps per encoder revolution** Valid values: • +1 or -1	

Limit switches (S7-300, S7-400)

Activating software limit switches (S7-300, S7-400)

Software limit switches

Software limit switches are only available for linear axes.

Monitoring begins as soon as the axis has been synchronized and software limit switch monitoring has been activated.

When software limit switch monitoring is deactivated, the "Software limit switch start" and "Software limit switch end" parameters (Axis DB addresses 6.0 and 10.0 (Page 2733)) are of no significance.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
42.1	SWLimitEnable	BOOL	TRUE	**Monitoring software limit switches** Valid values: • TRUE = Monitoring activated • FALSE = Monitoring deactivated

Specifying software limit switches (S7-300, S7-400)

Software limit switches

Software limit switches are only available for linear axes.

Linear axis: Software limit switch start/end

Monitoring begins as soon as the axis has been synchronized and software limit switch monitoring has been activated (see "Activating software limit switches (Page 2732)"). The working range is limited by means of software limit switches.

The value in "Software limit switch start" must always be less than the "Software limit switch end" value.

Incremental encoder

Initially, the axis is not synchronized after CPU startup. The axis is not synchronized and the software limit switches are not monitored unless a reference point approach or reference point setting has been completed.

Absolute encoder

Once reference point setting has been completed, the setting will not be lost at the next CPU restart. The software limit switches can be monitored.
The encoder range of the absolute encoder must cover at least the working range.

Rotary axis: Rotary axis start / Rotary axis end

"Rotary axis start" and "Rotary axis end" are used to define the range of axis position values to be displayed during one rotation of the axis.

The "Rotary axis end" value is theoretically the maximum an actual value can reach:

$$\text{Highest displayable value} = \text{Rotary axis end} - \frac{\text{Distance per encoder revolution}}{\text{Increments per encoder revolution}}$$

However, the theoretical maximum value is never displayed, as it is physically identical to the rotary axis start position (= zero).

Example:

- AxisLimitMin = 0.0°
- AxisLimitMax = 360.0°

With positive direction, the actual value counts ...359.8, 359.9, 0.0, 0.1, etc.

Pay attention to the "Axis distance per encoder revolution (Page 2727)" parameter when using absolute encoders.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
6.0	AxisLimitMax	REAL	$1.0e^{+6}$	Linear axis: **Software limit switch end** Rotary axis: **Rotary axis end** Valid range of values: • $-2^{24}+6 \leq$ AxisLimitMin < AxisLimitMax $\leq +2^{24}-6$ [length unit]	I
10.0	AxisLimitMin	REAL	$-1.0e^{+6}$	Linear axis: **Software limit switch start** Rotary axis: **Rotary axis start** Valid range of values: • $-2^{24}+6 \leq$ AxisLimitMin < AxisLimitMax $\leq +2^{24}-6$ [length unit]	I

Dynamic response (S7-300, S7-400)

General (S7-300, S7-400)

Specifying the maximum axis velocity (S7-300, S7-400)

Maximum axis velocity

Here, you configure the maximum axis velocity if the motor is controlled with the value entered in the "Reference value for maximum axis velocity" parameter (Axis DB address 88.0 (Page 2730)).

The "Maximum axis velocity" parameter is also used as a factor for calculating the value of the signal output to the drive, as well as for limiting the axis velocity.

Special features for rotary axes with absolute encoders

To enable the recording of all the rotary axis rotations, the maximum axis velocity must be limited so that the distance traversed in each scan cycle is less than half of the encoder range of the absolute encoder.

Example for single-turn encoders:

Condition => one axis rotation is equivalent to one encoder rotation.

Traversing distance / encoder range = 360.0°

Traversing distance / half of encoder range = 180.0°

Scan time = 0.01 s

$$\Rightarrow \text{"Maximum velocity"} \quad - \frac{\text{Travel distance/half the encoder range}}{\text{Scan time}}$$

=> "Maximum velocity" < 180.0° / 0.01s = 18000°/s is equivalent to 50 axis rev/s

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
14.0	MaxVelocity	REAL	0.1	**Maximum axis velocity** Valid range of values: • > 0.0 [unit of length/s]	I

Specifying velocity override (S7-300, S7-400)

Velocity override

"Velocity override" has a sustained effect on the configured velocity of a travel instruction. Acceleration and deceleration values are not affected.

Example: Travel trends with 50 % and 100 % override

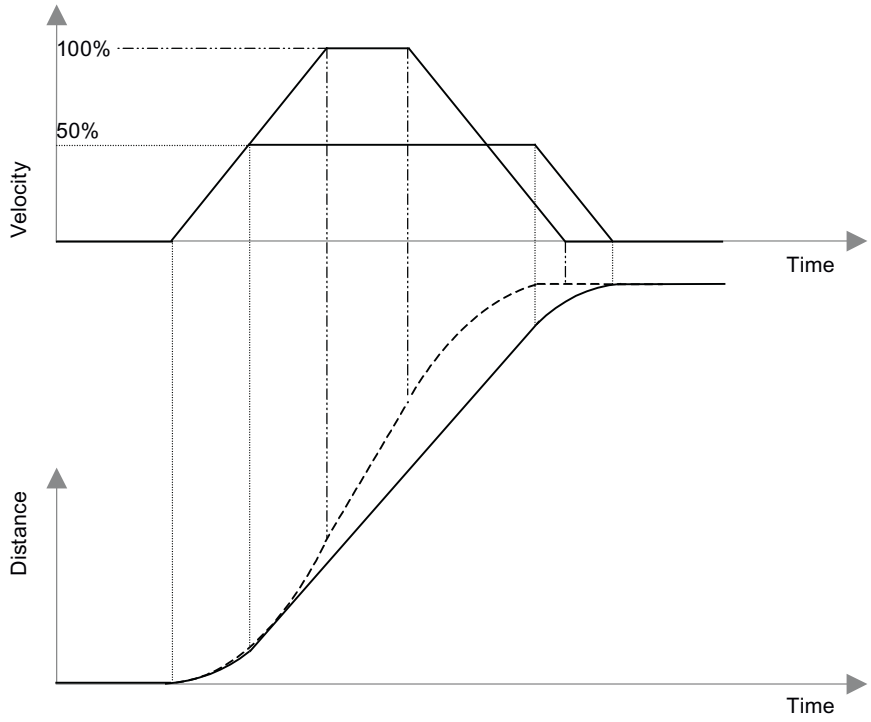

The velocity is doubled if you switch from 50 % to 100 %. As acceleration and deceleration are not affected by the "Velocity override" parameter, however, the positioning time is **not** halved.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
100.0	Override	REAL	100.0	**Velocity override** Valid range of values: • 0.0 to +100.0 [%]

Specifying the maximum axis velocity (S7-300, S7-400)

Maximum axis acceleration

The gradient of the acceleration ramp of the axis is limited to this value in controlled mode.

The maximum possible acceleration can be determined empirically.

It should only be high enough so that, when accelerating to maximum velocity (under load), it is just below the current limit of the drive.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
18.0	MaxAcceleration	REAL	0.1	**Maximum axis acceleration** Valid range of values: • > 0.0 [length unit/s^2]	I

Specifying maximum axis deceleration (S7-300, S7-400)

Maximum axis deceleration

The slope of the deceleration ramp of the axis is restricted to this value in controlled mode.

The maximum possible deceleration can be determined empirically.

It should only be high enough so that, when slowing down from maximum velocity (under load), it is just below the current limit of the drive.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
22.0	MaxDeceleration	REAL	0.1	**Maximum axis deceleration** Valid range of values: • > 0.0 [length unit/s^2]	I

Specifying the scan time (S7-300, S7-400)

Scan time

To ensure correct functioning of position control, it is important that all functions of one axis are processed in a fixed time frame. This time pattern is determined by calling all instructions of an axis in the same cyclic interrupt or synchronous cycle interrupt OB. Select the time pattern of this OB in the hardware configuration of the CPU.

For the "Scan time", enter the time pattern of the OB in which you call all instructions of an axis.

Example:

Call the program using the Easy Motion Control instructions in OB 35. You have set the cyclic interrupt time pattern of OB 35 to 10 ms in the hardware configuration.

Enter 0.010 as the "scan time".

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
0.0	Sample_T	REAL	0.01	**Scan time** of the axis instructions Recommended: • S7 300 to CPU 316 ≥ 0.020 [s] • S7 400 and as of CPU 317 ≥ 0.004 [s]	I

Displaying the ramp-up time (S7-300, S7-400)

Rampup time

The ramp-up time denotes the time an axis needs to accelerate from a standstill to maximum velocity.

The ramp-up time is calculated based on the following parameters:

Rampup time = Maximum axis velocity × velocity override / maximum acceleration

The output field displays the calculated ramp-up time.

Displaying the ramp-down time (S7-300, S7-400)

Ramp-down time

The ramp-down time denotes the time required to decelerate an axis without jerk limitation from maximum velocity to a standstill.

The ramp-down time is calculated based on the following parameters:

Ramp-down time = Maximum axis velocity × velocity override / maximum deceleration

The output field displays the calculated ramp-down time.

Emergency stop (S7-300, S7-400)

Specifying emergency stop deceleration (S7-300, S7-400)

Deceleration for emergency stop (hard stop)

At this parameter you define the slope of the time-controlled deceleration ramp used to decelerate the drive when an error with hard stop occurs.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
54.0	EmergencyDec	REAL	1.0	**Emergency stop deceleration** Valid range of values: • > 0.0 [length unit/s^2]	I

Displaying the ramp-down time (S7-300, S7-400)

Ramp-down time

The ramp-down time denotes the time required to decelerate an axis in emergency stop mode from maximum velocity to a standstill.

The ramp-down time is calculated based on the following parameters:

Ramp-down time = Maximum axis velocity / emergency stop deceleration

The output field displays the calculated ramp-down time.

Monitoring (S7-300, S7-400)

Specifying the target range (S7-300, S7-400)

Target range

The "target range" lies to the left and right of the target (see Concept Definitions (Page 2711)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
30.0	TargetRange	REAL	1.0	**Target range** Valid range of values: • > 0.0 [length unit]	I

Specifying the standstill range (S7-300, S7-400)

Standstill range

When at a standstill, the axis is monitored to determine whether it is remaining at a given position or drifting away from it. The "Outside standstill range" error bit (Axis DB address 130.3 (Page 2769)) is set as soon as the axis leaves the "standstill range" without an active traversing job. The "standstill range" lies to the left and right of the target (see Concept Definitions (Page 2711)).

Recommended: The "standstill range" should be greater than the "target range" (Axis DB address 30.0 (Page 2738)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
34.0	StandstillRange	REAL	1.5	**Standstill range** Valid range of values: • > 0.0 [length unit]	I

Target approach monitoring (S7-300, S7-400)

Monitoring the target approach

When target range monitoring is activated, a movement is terminated as soon as the actual position value has reached the "target range" (axis DB address 30.0) within the "Monitoring time for target approach" (axis DB address 26.0).

If target range monitoring is deactivated, axis movement is terminated as soon as the position setpoint has reached the target (see Terminology (Page 2711)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment
42.0	MonitorTargetAppr	BOOL	TRUE	**Target approach monitoring** Valid values: • TRUE = Monitoring activated • FALSE = Monitoring deactivated

Specifying the monitoring time for target approach (S7-300, S7-400)

Monitoring time for target approach

The "Monitoring time for target approach" starts as soon as the traversing axis has reached the **position setpoint** in the target area, provided the "Monitor target approach" bit has been set (Axis DB address 42.0 (Page 2739)). If the **actual position value** has not reached the range defined in the "target area" parameter (Axis DB address 30.0 (Page 2738)) on expiration of this time, the "Target approach error" bit is set in the axis DB (Axis DB address 130.4 (Page 2769)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
26.0	MonTimeTargetAppr	REAL	1.0	**Monitoring time for target approach** Valid range of values: • > 2 × Sample_T [s] (scan time (Page 2736))	I

Specifying the valid maximum following error (S7-300, S7-400)

Maximum permitted following distance

The following error monitoring function is used to check whether the actual position during a motion remains inadmissibly far behind the position setpoint. If the maximum following error has been exceeded, the "Following error exceeded" error bit is set in the axis DB" (Axis DB address 130.2 (Page 2769)).

Recommended setting: ("maximum velocity" / "controller gain") × 1.1 (see axis DB addresses 14.0 (Page 2734) and 44.0 (Page 2729))

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
38.0	MaxFollowingDist	REAL	5.0	**Maximum permitted following distance** Valid range of values: • > 0.0 [length unit]	I

6.2.10.4 Download to CPU (S7-300, S7-400)

New or modified project data must be downloaded to the CPU.

Downloading the program blocks and the TO configuration to the CPU

Proceed as follows for downloading:

1. Select the "Program blocks" object, or the "Technology objects" object in the project navigator.
2. Select the "Online > Download to device" command.

Note

Whenever you download blocks to the CPU, the TIA Portal always verifies consistency between the offline project blocks and the online blocks in the CPU.

Regardless of the marking in the project tree, all new and modified offline blocks and technology objects are downloaded to the CPU.

On each loading operation, all process values of the data blocks are reset to their initial values.

Downloading the hardware configuration to the CPU

Proceed as follows when downloading the hardware configuration:

1. Select the CPU object in the project navigator.
2. Select the "Download > Hardware configuration" command from the shortcut menu.

Downloading the program blocks and the TO configuration to the CPU

Proceed as follows for downloading:

1. Select the CPU object in the project navigator.
2. Select the "Online > Download to device" command.

6.2.10.5 Programming Easy Motion Control (S7-300, S7-400)

User Program Setup (S7-300, S7-400)

Interaction of the instructions

The following figure demonstrates how instructions interact in Easy Motion Control.

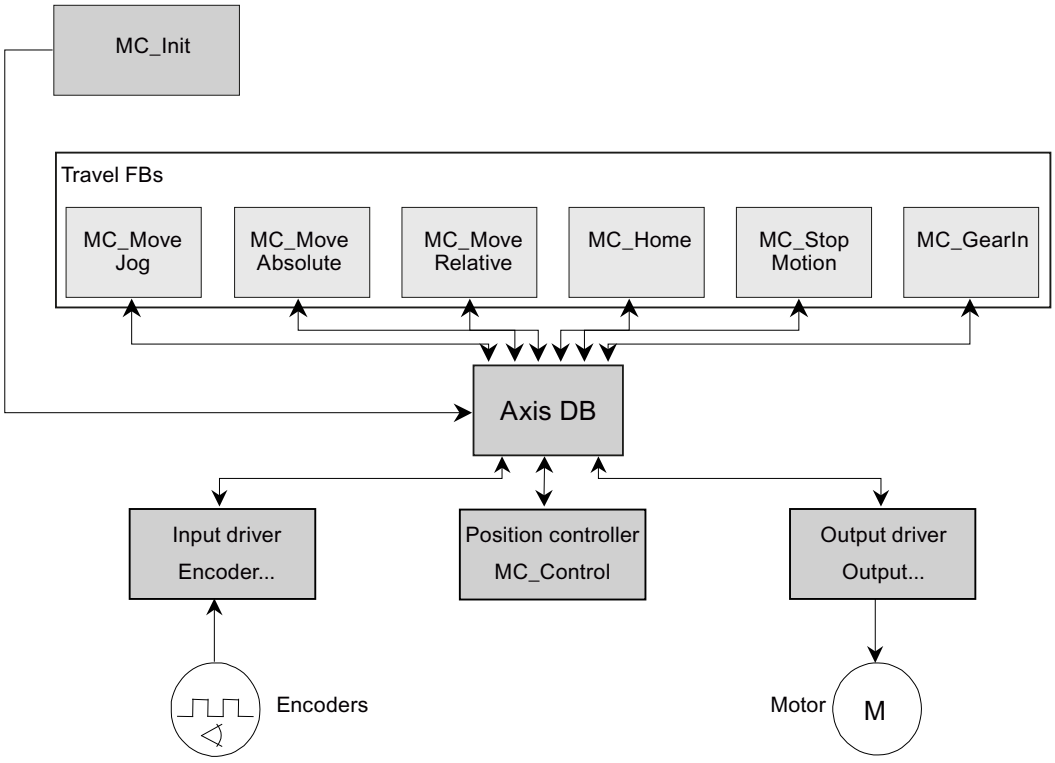

instruction MC_Init serves to **initialize** the instructions.

The **input driver** (Encoder… instruction) reads the actual position value of an axis from the I/O module and scales this value in accordance with the length unit that you have selected.

One of the **travel instructions** generates the position setpoint.

The **position controller** (MC_Control instruction) compares the actual position value with the position setpoint and generates a velocity setpoint value from the difference.

The **output driver** (Output… instruction) scales the velocity setpoint to the suitable format and writes it to the connected I/O module.

The program for a **closed-loop control circuit** consists of the input driver (Encoder…), at least one travel instruction, the controller (MC_Control), and the output driver (Output…).

All instructions of an axis work with the same **Axis DB**. This axis DB contains all axis data needed to operate the axis and is available for each axis. Read the information on Loading the axis DB (Page 2753).

Call environment

To ensure correct functioning of position control, it is important that all functions of one axis are processed in a fixed **time frame**. This time pattern is determined by calling the instructions of Easy Motion Control in the **same** execution level. The following two methods can be applied, depending on the CPU and type of I/O connection used:

Cyclic interrupt OB (OB 30 to OB 38)

Used for central integration of the I/O, or for distributed integration via DP bus system without constant bus cycle time. The cyclic interrupt time corresponds to the position-control cycle.

Synchronous cycle interrupt OBs (OB 61 to OB 64)

The position control cycle on a continuously isochronous DP bus for I/O integration corresponds to the value you have selected in the bus configuration for "Constant bus cycle time".

This value must be entered at the axis DB in both cases.

For general information on the isochronous mode, refer to the TIA Portal online help and to the Isochronous Mode Function Manual (http://support.automation.siemens.com/WW/view/en/15218045).

Please note the following special features for Easy Motion Control:

- When configuring the I/O bus and the DP slaves / IO devices (including their modules, if applicable), you must select and configure the "constant bus cycle time".
- The I/O addresses of an axis used for position detection or setpoint output must be placed in the same process image partition (PIP). When using gear axes, both gear axes must be placed into the same PIP.
- When configuring the CPU, this PIP is assigned to the synchronous cycle interrupt used.
- You need to update this PIP in the interrupt OB before and after each call of the EMC blocks using the system instructions "SYNC_PI" / "SYNC_PO".
- Evaluate the feedback from these system instructions in your program and respond accordingly. (If an error has occurred, the PIP update has possibly failed and the actual value read was therefore incorrect.)

Recommended Call Sequence

The following figure shows the call sequence recommended for the instructions of Easy Motion Control.

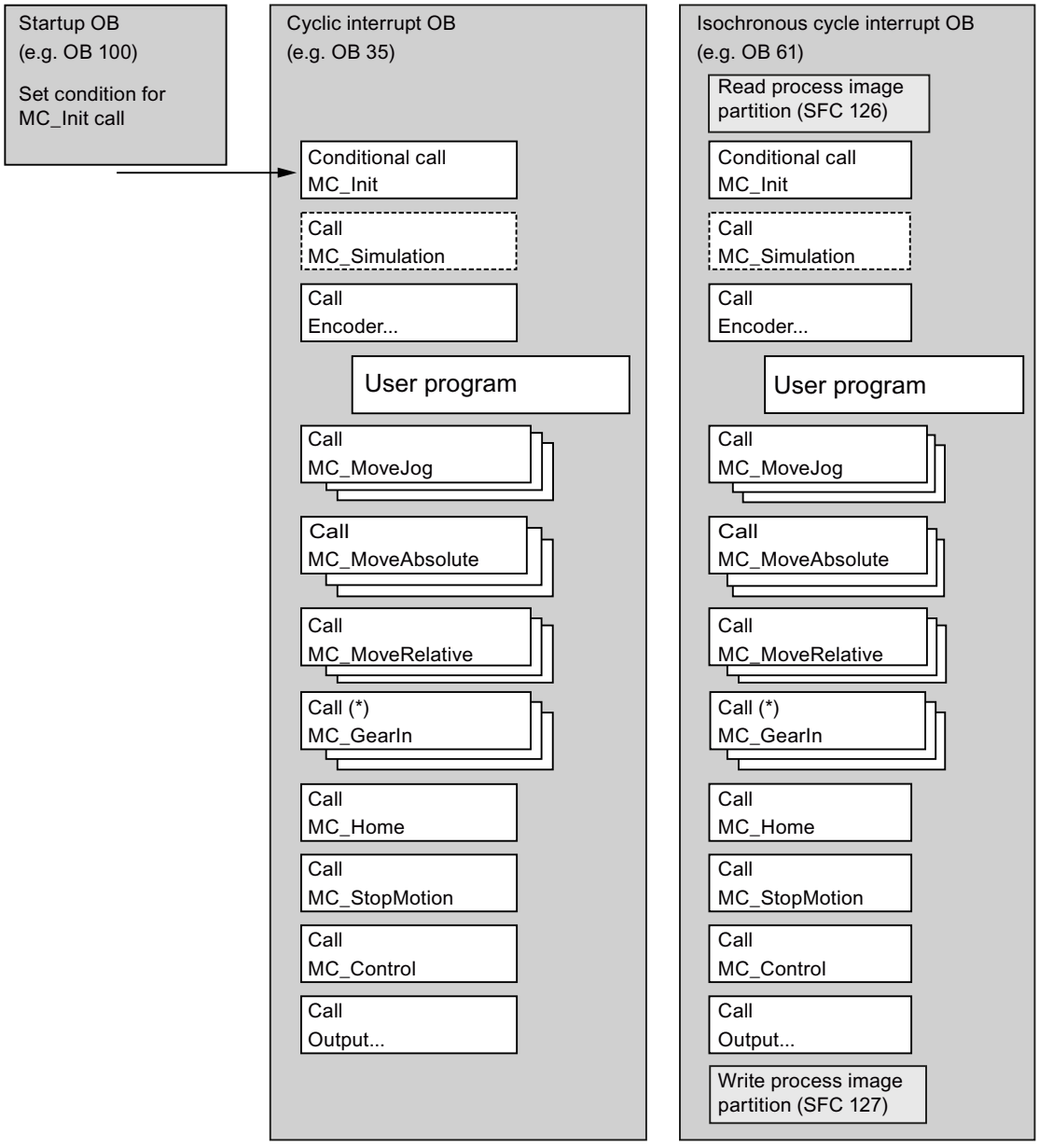

(*) A call of the gear instruction MC_GearIn is only appropriate for following axes.

Multiple Axes

We recommend the following user program structure for controlling multiple axes in one CPU:

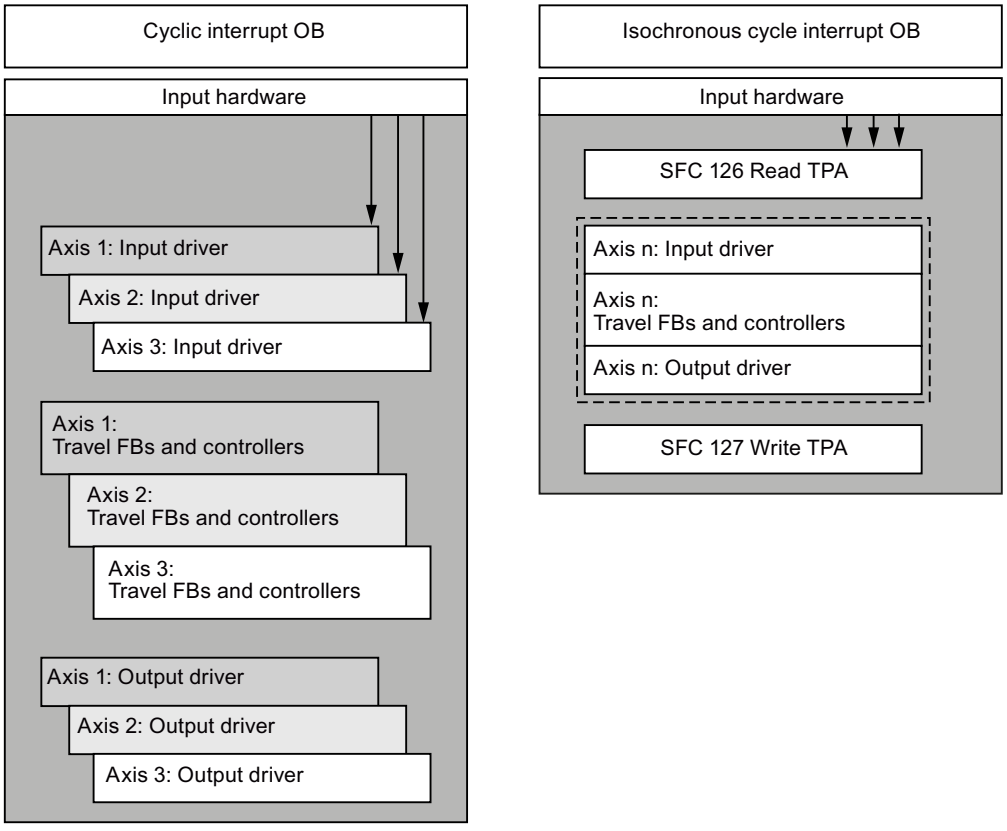

Recommended for the cyclic interrupt OB.	Recommended for the synchronous cycle OB.
Advantage: Calling the input drivers at the start of the OB produces a minimum time offset and minimum jitter when reading the I/O.	**Advantage:** It is possible to group *all* instructions of an axis in a *single* multi-instance instruction. Due to the PIP, which is required for the synchronous cycle interrupt OB, the read I/O data are consistent irrespective of the driver call.
Disadvantage: It is not possible to group *all* instructions of an axis in a *single* multi-instance instruction.	

All interacting axes must be called in the same OB.

All axes of an isochronous application must be placed into the same process image partition.

Handling Instance Data

You can store the **instance data** for each instruction in its own DB, or in a single multi-instance DB for all instructions.

On completion of the download of the instance data of a instruction to the module, it is necessary to initialize the instruction. Set the "Init" input of the instruction accordingly. See also Initialization and Parameter Changes (Page 2748) and Saving and Loading the Axis Data Block (Page 2753).

Interconnecting the instructions

The **connections** between the instructions and the axis DB displayed in the above figure under Interaction of the instructions are handled using the following parameters:

- Axis
- Init

(See Significance of the common parameters (Page 2748))

The following interconnections are also necessary:

- The "EnableDrive" input of instruction MC_Control must be interconnected with an enable switch. instruction MC_Control will only output a speed setpoint if input "EnableDrive" = TRUE.
- Input "EnableDrive" of instruction Output… must be interconnected with output "DriveEnabled" of instruction MC_Control. "EnableDrive" = TRUE must be set in order to output a value not equal to zero at instruction Output….
- You should interconnect the "DriveEnabled" output of instruction MC_Control with the enable input of your drive.

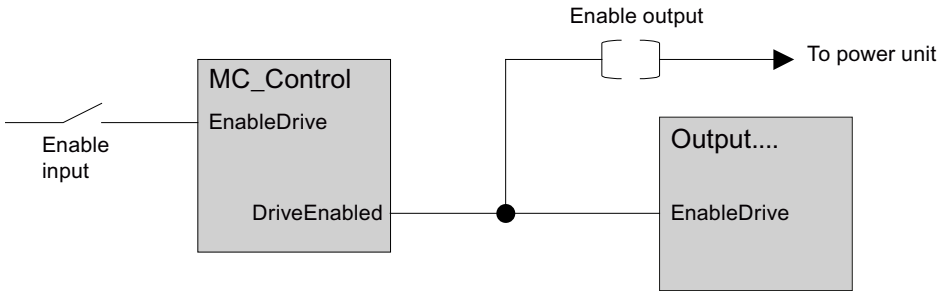

Influences on Position Control in the CPU (S7-300, S7-400)

Recommended for optimal position control:

- Scan time must be adhered to as closely as possible.
  - In the hardware configuration parameters for the CPU, assign the highest possible priority to the cyclic interrupt OB that calls the instructions.
  - Minimize the communication load on the cycle as far as possible with appropriate parameter settings in the hardware configuration of the CPU.
  - Use an isochronous environment to avoid having to take additional measures.
- When integrating encoder value acquisition via distributed I/O (PROFIBUS DP / PROFINET IO), set the maximum possible transmission rate for the I/O bus (for example, 12 Mbps).

Importance of Shared Parameters (S7-300, S7-400)

The following table lists the parameters that are used in all instructions.

Parameter	Declaration	Data type	Description
EN	INPUT	BOOL	Enable input of LAD/instructionD box
ENO	OUTPUT	BOOL	Enable output of LAD/instructionD box
Axis	IN-OUT	STRUCT	Interconnect this parameter with the axis DB. For a DB of the type AXIS_REF, the interconnection is: • Axis := DBname.Ax For a DB that contains a variable of the type AXIS_REF, the interconnection is: • Axis := DBname.Variable name.Ax
Init	IN-OUT	BOOL	Interconnect these parameter with an initialization bit in the axis DB that is not used by any other instruction ("Init.I0" … "Init.I31"; refer to Bit array for the initialization function (Page 2756)). With Init = TRUE, the instruction completes its initialization and then resets the bit. For a DB of the type AXIS_REF, the interconnection is: • Init := DBname.Ax.Init.Ix For a DB that contains a variable of the type AXIS_REF, the interconnection is: • Init := DBname.Variablename.Ax.Init.Ix when Ix = I0 … I31

Initialization and Parameter Changes (S7-300, S7-400)

The functionality of Easy Motion Control instructions is based on the data stored in the axis DB. This is where parameters describing the basic axis itself can also be found. If these parameters are changed, **all** Easy Motion Control instructions called in your user program must run their initialization **once** following the change. These parameters are identified in configuration sections (Page 2718) by the following symbol:

Bit array "Init.Ix" is defined for initialization in the axis DB and all instructions are assigned the "Init" parameter. The instructions run their initialization routine if their "Init" input is set and then reset the interconnected initialization bit.

The bits in the "Init.Ix" bit array are set in the axis DB:

- At every call of instruction MC_Init.
- By the configuration software in the event of any change in corresponding parameters of the axis DB

Interconnect the "Init" parameter with an initialization bit in the axis DB that is not in use by any other instruction for all Easy Motion Control instructions called in your user program. This interconnection ensures that the instructions run their initialization routine whenever this becomes necessary as a result of parameter changes or CPU restart ("Init.I0" … "Init.I31"; refer to Bit array for the initialization function (Page 2756)).

Call instruction MC_Init once in the following situations:

- After every startup of the CPU
- If you have changed one of the parameters identified by the following symbol in the configuration sections (Page 2718) without using the configuration software:

For additional information on the initialization instruction, refer to MC_Init.

WARNING

Parameters identified by the following symbol in the configuration section (Page 2718) may only be changed **when the axis is at a standstill**:

If you do not observe this rule, controlled axis movement cannot be guaranteed.

WARNING

During initialization, specific auxiliary values are calculated and saved to the static parameters of the instructions and to the axis DB. For this reason, you are only permitted to overwrite the instance data blocks and the axis DB when the axis is at a standstill.

If you do not observe this rule, controlled axis movement cannot be guaranteed.

Sequence of Travel Movements (S7-300, S7-400)

Start of a travel order

The input parameters of travel instructions are interpreted once at the start of a travel order. Changing these values while an order is in progress does not affect current travel.

Maximum Travel Distance

The travel distance per movement is limited to the distance that is equivalent to 2^{24} steps of the encoder. It is calculated as follows:

$$\text{Maximum traversing distance} = 2^{24} \times \frac{\text{"Axis distance per encoder revolution"}}{\text{"Steps per encoder revolution"}}$$

Example:

Axis distance per encoder revolution	= 2.5 mm
Steps per encoder revolution	= 4096
Maximum distance = 2^{24} x 2.5 mm / 4096	= 10240 mm

Exception:

The MC_MoveJog and MC_GearIn instructions do not limit the traversing distance of a rotary axis.

Overriding travel

A currently active travel instruction can always be

- Overridden by starting a different travel instruction
- Cancelled by the start of instruction MC_StopMotion
- Aborted by setting manual mode

When overridden by another travel instruction, the axis is decelerated or accelerated to the velocity specified in the overriding instruction and then approaches the new target without intermediate stop. The overridden travel instruction sets the "CommandAborted" output.

If the new motion requires a change in direction of the axis, the axis is first stopped and then moved in the opposite direction (reversal).

For safety reasons, the deceleration of the two travel instructions involved in the override must coincide. This ensures that on reversing direction, the **original** target of the movement is not overtraveled.

Multiple use of a travel instruction

A travel instruction can be called several times for an axis in your program if necessary.

Error Response

All errors that occur are indicated in the axis DB, where they must be acknowledged by the user (see Error Displays and Error Acknowledgment in the axis DB (Page 2764)). Each error will abort any travel order that is currently active. A new travel order can only start if there is no longer any error pending.

Signal Flow Diagrams

The following diagram shows the signal flow of a movement without error.

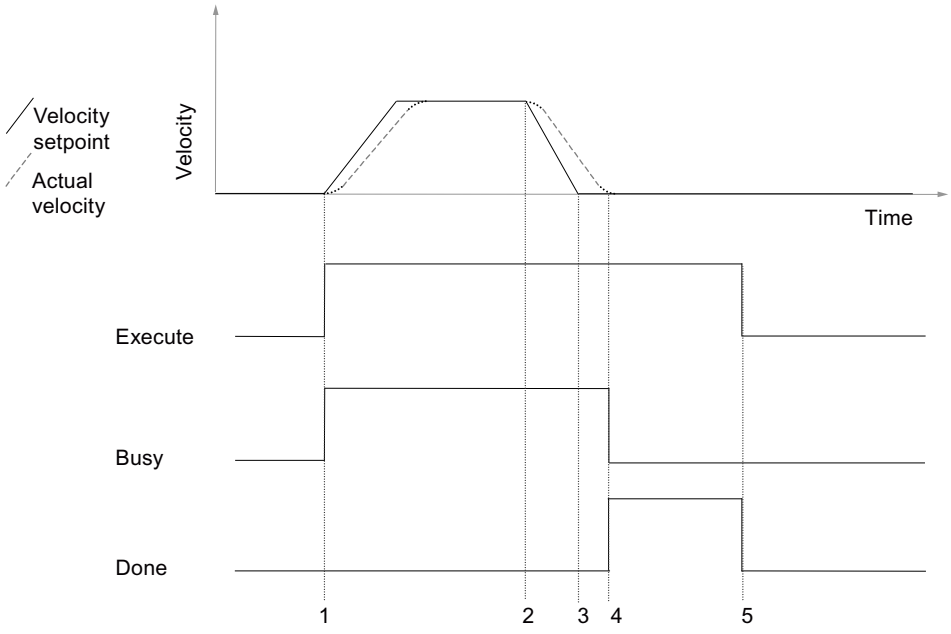

1 Start of the travel FB

2 Brake application point

3 Setpoint has reach target position

4 Axis has reached the target range

5 Reset of "Execute" resets "Done". The process is completed. If "Execute" is reset before the target position has been reached, "Done" is set for one cycle after the target position has been reached and is then reset again.

The following diagram shows the signal flow of a movement with error.

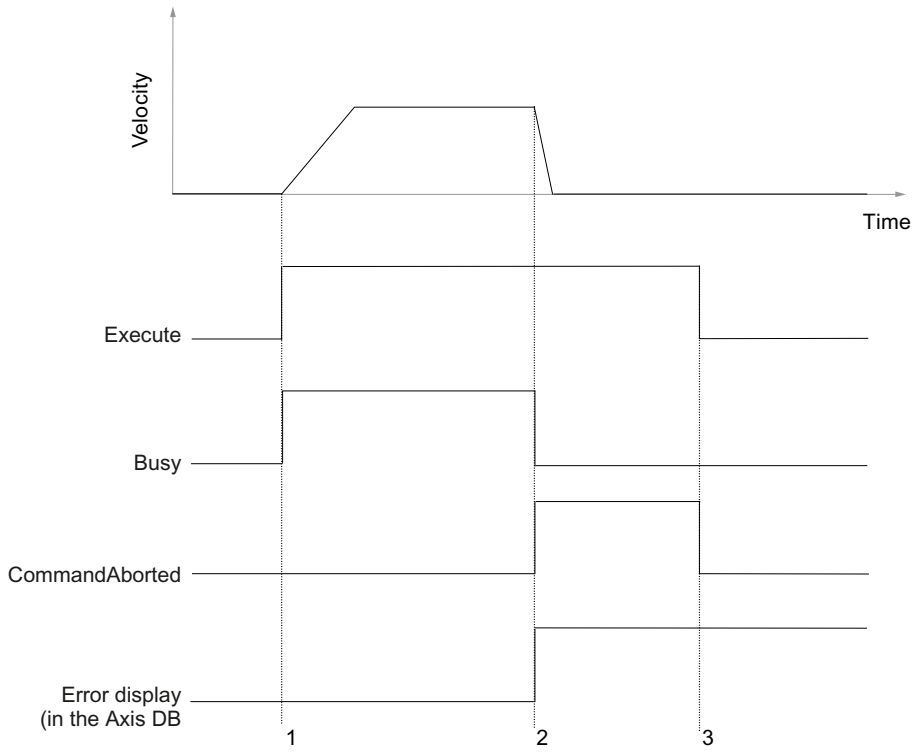

1 Start of the travel FB

2 Error occurred

3 Reset of "Execute" resets "CommandAborted" feedback. If "Execute" is reset before the error has occurred, "MotionAborted" is set for one cycle after the error has occurred and is then reset again.

Axis data block (S7-300, S7-400)

Function (S7-300, S7-400)

The axis DB is the central database of Easy Motion Control. It contains:

- Parameters that describe the basic axis (for example, I/O addresses)
- Position control parameters that are calculated based on these basic parameters during initialization of the instructions of Easy Motion Control.
- Actual values of the position control (for example, position, error states, etc.).
- Initialization bits for coordinating the start of the instructions

Interaction of these parameter types is ensured if you take one of the following measures:

- Edit and transfer the axis DB using the configuration software.
- After having changed specific parameters, ensure that instruction MC_Init is called.

In both cases, the plausibility of the entered parameters is checked wherever possible, and the values internally required by Easy Motion Control are computed.

The configuration software also analyzes the type of changes and uses this information to control the loading processes.

Saving and Loading the Axis Data Block (S7-300, S7-400)

WARNING

The entire axis DB can only be overwritten when the axis is at a standstill.

You may **NOT** download the axis DB in RUN mode.

Correct movement of the axis cannot be ensured if you download the axis DB to the module while ignoring the notes in the following sections.

Offline

After specific parameters have been changed, all instructions of Easy Motion Control must complete an initialization routine.

These parameters are identified in the description of the axis DB by the following symbol:

Changing and loading these parameters:

- Is only possible when the axis is at a standstill
- Must include setting the initialization bits (see Initialization and Parameter Changes (Page 2747)).

Online

Transferring parameters that were changed online (for example, the coordinate of the reference point for absolute encoders) to the offline management is not supported.

Axis parameters (S7-300, S7-400)

All information on length, velocity, and deceleration provided in the "Configuring Easy Motion Control (Page 2717)" section refers to an arbitrary, yet uniform length unit. All times are specified in seconds.

In the tables of that section, the parameter name from the axis DB is specified in the "Name" column, while the parameter name used in the configuration software is specified in the "Comments" column.

Note

Parameters not listed are reserved for internal use by the Easy Motion Control instructions and must not be altered.

Actual values (S7-300, S7-400)

Overview

The following parameters in the axis DB show information on the current status of the moving axis. You must not change the contents of these parameters.

Actual position value

Displays the actual position value.

Address	Name	Type	Initial value	Comment
104.0	ActPosition	REAL	0.0	**Actual position value** Range of values: Floating point number [length unit]

Following error

Following error = current position setpoint - actual position value

Address	Name	Type	Initial value	Comment
108.0	FollowingDistance	INT	0	**Following error** Range of values: Floating point number [length unit]

Residual distance

Indicates the distance remaining to the target position during movements with defined target (MC_MoveAbsolute, MC_MoveRelative).

Travel without target (MC_MoveJog, MC_Home) and errors with hard stop reset the remaining distance to 0.

Address	Name	Type	Initial value	Comment
112.0	RemainingDistance	REAL	0.0	**Distance remaining** to target Range of values: Floating point number [length unit]

Setpoint velocity

Displays the velocity setpoint of the axis calculated by the travel instruction during a movement.

Address	Name	Type	Initial value	Comment
116.0	NomVelocity	REAL	0.0	**Setpoint velocity** Range of values: Floating point number [length unit/s]

Actual velocity

The actual velocity of the axis is displayed here.

Address	Name	Type	Initial value	Comment
120.0	ActVelocity	REAL	0.0	**Actual velocity** Range of values: Floating point number [length unit/s]

Axis synchronized

When using an incremental encoder:

- FALSE: After CPU startup or detection of an encoder error
- TRUE: After a reference point approach or reference setting

When using an absolute encoder:

- FALSE: Before initial reference setting
- TRUE: After reference setting

Address	Name	Type	Initial value	Comment
124.0	Sync	BOOL	FALSE	**Axis synchronized** FALSE = Not synchronized TRUE = Synchronized

Current encoder value

The current encoder value is displayed here.

Address	Name	Type	Initial value	Comment
136.0	EncoderValue	DINT	L#0	**Current encoder value** Range of values: Decimal [steps]

Bit Field for the Initialization Function (S7-300, S7-400)

Bit array function

All Easy Motion Control instructions must be initialized after every CPU startup and after specific parameters have been changed.

For this purpose, the "Init.Ix" bit array is defined for initialization in the axis DB and all instructions are assigned the "Init" parameter. The instructions run their initialization routine if their "Init" input is set. On completion, the instructions reset the interconnected initialization bit.

You must interconnect the "Init" parameter with a bit of the bit array at each call of a instruction. This bit may not be used by another instruction (refer to Initialization and parameter changes (Page 2747)).

Whenever the MC_Init instruction is called, or if you change a parameter in the configuration software, which is not activated unless the instructions have completed their initialization, all bits of this array is set simultaneously to TRUE. This allows all instructions to carry out an initialization with their next call.

Address	Name	Type	Initial value	Comment
278.0	Init	STRUCT	TRUE	Bit array for initialization of the instructions

User Program as a Multi-instance FB (S7-300, S7-400)

If you want to save all parameters of your travel instructions to a single DB, or use the same program for several axes, create the user program in a multi-instance FB.

Place the axis data in this multi-instance DB, as well. To do this, declare a static variable of the type AXIS_REF from the "Easy Motion Control" library in your multi-instance FB.

Interconnect the Axis and Init parameters of the instructions of Easy Motion Control with this variable.

Call instruction MC_Init conditionally as the first instruction in your multi-instance block. This condition must be set in the startup OB (for example, OB100) and reset after instruction MC_Init has been called.

New multi-instance DBs can only be created using the TIA Portal, or the LAD/STL/instructionD Editor.

6.2.10.6 Commissioning Easy Motion Control (S7-300, S7-400)

Safety Information (S7-300, S7-400)

Introduction

Read the following notes before you start commissioning.

> **WARNING**
>
> **The commissioning functions result in a movement of the axis.**
>
> Uncontrolled travel motions of the axis during commissioning and operation can cause serious injury and material damage.
>
> To avoid injury to persons and damage to objects, take the following precautions:
>
> - Install an EMERGENCY STOP switch in the vicinity of the computer. This is the only way to ensure that the system is reliably switched off in the event of a hardware or software failure.
> - Install safety limit switches for direct control of the power units of all drives.
> - Ensure that nobody has access to the area of the system where there are moving parts.

Commissioning the axis technology object (S7-300, S7-400)

The commissioning editor helps you with commissioning and the function test for the axis technology object. You can run the following tests and measurements:

- Wiring test (Page 2758)
- Velocity measurement (Page 2758)
- Homing the axis (Page 2759)
- Distance measurement (Page 2760)

Procedure

To open the commissioning editor of the axis technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the corresponding instance of the AXIS_REF technology object (e.g. EMC_AXIS_1) in the project tree.
3. Double-click on the "Commissioning" object.

Remedy in the case of errors

If errors occur during commissioning, you will find information about possible remedies in section Error profiles and remedies during commissioning of the user program (Page 2761).

Additional information

The corresponding tooltips include detailed information on the parameters and buttons of the commissioning user interfaces.

Testing the wiring (S7-300, S7-400)

The Wiring test is used to determine the direction of rotation of the axis.

Procedure

To perform the Wiring test, follow these steps:

1. Click on "Wiring test" in the configuration view.
2. Click the "Activate" button.
   A query regarding activation of manual control is displayed.
3. Click "OK".
   An online connection is created between STEP 7 (TIA Portal) and the CPU.
4. Enter a velocity as a percentage of the maximum velocity in the "Specified velocity" entry field.
5. Enter a value in the "Ramp-up time" entry field.
6. Make sure that the start position permits a movement of the axis in both directions.
7. Click the "Jog backwards" or "Jog forwards" button to move the axis.
8. In "Result", specify whether the axis has moved in the requested direction.
9. If you have selected "Yes" as the result, click "OK" in the window that appears.
10. If you have selected "No" as the result, click "Yes" in the window that appears.
    The parameters "Encoder polarity" and "Drive polarity" are changed.

Measuring the axis velocity (S7-300, S7-400)

The velocity measurement determines the maximum axis velocity.

Procedure

To perform velocity measurement, follow these steps:

1. Click on "Velocity measurement" in the configuration view.
2. Click the "Activate" button.
   A query regarding activation of manual control is displayed.

3. Click "OK".
   An online connection is created between STEP 7 (TIA Portal) and the CPU.
4. Enter a velocity as a percentage of the maximum velocity in the "Specified velocity" entry field.
5. Enter a value in the "Ramp-up time" entry field.
6. Make sure that the start position permits a sufficiently long travel distance in at least one direction.
7. Click the "Jog backwards" or "Jog forwards" button and hold it until the value in the "Actual output" output field has reached the "Setpoint output".
   Easy Motion Control calculates the maximum axis velocity and displays it in the "Determined value" output field.
8. If you want to save the calculated axis velocity in the project, click the "Apply" button.
   The calculated value overwrites the "Maximum velocity" in the online/offline data management of the technology object.

Set reference point (S7-300, S7-400)

A reference point is set for absolute encoders, and this is referred to as absolute encoder adjustment in this case.

Procedure

To set a reference point for an absolute encoder, proceed as follows:

1. Click on "Homing the axis" in the configuration view.
2. Click the "Activate" button.
   A query regarding activation of manual control is displayed.
3. Click "OK".
   An online connection is created between STEP 7 (TIA Portal) and the CPU.
4. Enter a velocity as a percentage of the maximum velocity in the "Specified velocity" entry field.
5. Enter a value in the "Ramp-up time" entry field.
6. Click the "Jog backwards" or "Jog forwards" button until the axis has traveled to the reference point.
   Alternatively:
   Move the axis to the reference point manually.
7. In the "Input of reference position" input field, enter a position value as reference value.
   The offset between the reference position and the physical axis position is determined and saved in the axis DB.
8. In the window that appears, confirm whether you wish to save the offset for the absolute encoder adjustment in the offline data management of the technology object.

Note

If you replace the absolute encoder, you will have to set the reference point again.

Measuring the axis distance (S7-300, S7-400)

The distance measurement determines the axis distance per encoder revolution.

Procedure

Proceed as follows to perform the distance measurement:

1. Click on "Distance measurement" in the configuration view.
2. Click the "Activate" button.
   A query regarding activation of manual control is displayed.
3. Click "OK".
   An online connection is created between STEP 7 (TIA Portal) and the CPU.
4. Enter a velocity as a percentage of the maximum velocity in the "Specified velocity" entry field.
5. Enter a value in the "Ramp-up time" entry field.
6. Click the "Jog backwards" or "Jog forwards" button until the axis has traveled to the start position.
   Alternatively:
   Move the axis to the start position manually.
7. Enter a position value as the start position in the "Input of start position" entry field.
8. Click the "Jog backwards" or "Jog forwards" button until the axis has traveled to the end position.
   Alternatively:
   Move the axis to the end position manually.
9. Enter a position value as the end position in the "Input of end position" entry field.
   Easy Motion Control calculates the axis distance per encoder revolution and displays it in the output field "Determined value".
10. If you want to save the calculated axis distance per encoder revolution in the project, click "Apply".
    The calculated value is entered in the online/offline data management of the technology object.

Error profiles and remedies during commissioning of the user program (S7-300, S7-400)

Axis does not Move Despite Travel Order (S7-300, S7-400)

If the axis is not moving irrespective of a pending travel command, check the following conditions:

Error	Remedies
Wiring of the drive is faulty	• Check whether the "DriveEnabled" output of the MC_Control instruction is continuously interconnected with the enable input of the power unit. • Check whether the "DriveEnabled" output of the MC_Control instruction is interconnected with the "EnableDrive" input of the output driver.
I/O addresses or channel numbers are incorrect	Set the correct I/O addresses and channel numbers: In the axis DB, the **start** addresses of the modules are expected. From the start address and channel number (counting method: from channel 0), the driver blocks calculate the module-specific I/O addresses for setpoint and actual values.
Simulation mode switched on	Switch off simulation mode ("Sim" = FALSE). If the simulation mode is switched on ("Sim" = TRUE), **no** I/O values are read or written.
Manual mode switched on with missing setpoint velocity	If manual mode is switched on ("ManEnable" = TRUE), specify a "manual setpoint velocity" ("ManVelocity" <> 0.0 [length unit/s]).
Drive enable not set	Set the "EnableDrive" input of the MC_Control instruction to TRUE.
Axis error pending	If the "group error" bit ("Error") is set in the axis DB, the type of axis error is visible in the bit arrays for the error display: • After the MC_Init instruction is called, the "Stop status requiring acknowledgment" error is always set ("Err.StoppedMotion"). Acknowledge the error by setting the "Group acknowledgment" "ErrorAck" = TRUE. • After you have rectified incorrect values in the case of a parameter assignment error, you need to restart MC_Init. • In the case of an encoder error, check whether the position input module is fully configured (with the help of the module configuration software, if necessary). • Other errors: Remedy and acknowledge the error by setting the "Group acknowledgment" "ErrorAck" = TRUE.
"Group acknowledgment" is not reset	If a "Group acknowledgment" ("ErrorAck") you have set is not detected and reset by Easy Motion Control , then: • Prevent the Easy Motion Control instructions from remaining in their initialization branch due to the constant call of the MC_Init instruction. • Check whether the position decoder module responds to the error acknowledgment.
"Velocity override" not 0%	Configure the "Velocity override" = 0% ("Override").
Travel command not processed by program	Set the parameter "Busy" = TRUE in the travel instruction.
"Parameter of travel instruction inadmissible" ("Err.DataErr") or "Target outside travel range" ("Err.TargetErr")	If you download the axis DB to the CPU without using the Easy Motion Control configuration software, you must subsequently call the MC_Init instruction. MC_Init stores the auxiliary values in the axis DB.

Axis Moves without Travel Order (S7-300, S7-400)

Possible causes of axis movement without a travel command are:

Error	Causes	Remedies
The axis starts moving very quickly.	The direction of control action has been set incorrectly.	Adjust the "Encoder polarity" ("PolarityEncoder") and "Drive polarity" ("PolarityDrive") parameters with the help of the "Wiring test" wizard. In this way, you adapt the instructions to your hardware installation.
The axis is drifting slowly.	The power unit of the drive is enabled and the axis is **not** in control (MC_Control instruction outputs "DriveEnabled" = FALSE).	—
	The analog output is not correctly wired with the power unit of the drive.	Correct the wiring between the analog output and the power unit of the drive.
	Either the input driver or the output driver is not accessing the modules belonging to the drive because the module address or the channel number in the axis DB is set incorrectly.	Adjust the parameters with the help of the "wiring test" wizard.

Axis Velocity Other than Expected or Axis Oscillates (S7-300, S7-400)

Are the parameters describing the axis correct?

There are certain relations between these parameters as shown in the following example.

- "Controller gain" ("FactorP"):
  The axis may oscillate if the selected "Controller gain" ("FactorP") factor is too high.
- "Maximum axis velocity" ("MaxVelocity")
- "Reference value for maximum axis velocity" ("DriveInputAtMaxVel"):
  Example of a ±10 V control:

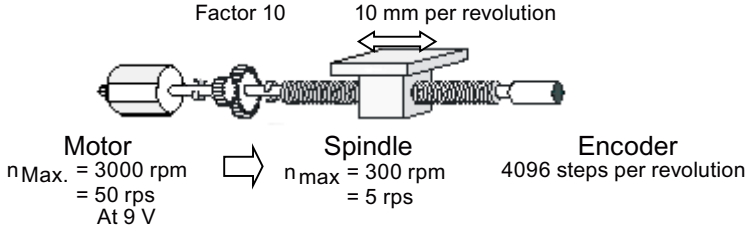

The following relationship holds:
With an output voltage of 9 V at the output module, the drive rotates at a speed of 3000 rpm. This speed is proportional to a spindle speed of 300 rpm or 5 rps and the slide moves by a distance of 10 mm per spindle revolution. The maximum velocity of the slide is thus 10 mm/s • 300 U/min = 50 mm/s.
In other words, set:

– "Maximum axis velocity" = 50 mm/s and "Reference value for maximum axis velocity" = 9 V.

- "Steps per encoder revolution" ("StepsPerRev"):
  You can check whether the number of steps produced by the encoder coincides with the configured value by observing the parameter "Encoder value" with the configuration software, while you turn the **encoder once** by hand.
  Example: The "encoder value" changes by 4096 steps per spindle revolution.
- "Axis distance per encoder revolution" ("DisplacementPerRev"):
  This parameter assigns a specific distance to one revolution of the encoder. You can check whether this is correct by observing the parameter "Actual position" with the configuration software, while you turn the **encoder once** by hand.
  Example: The "actual position" changes by 10 mm per spindle revolution.
- "Scan time" ("Sample_T"):
  Scan time is used together with the distance covered since the last scan in order to calculate axis velocity The time must correspond to the call time of the OB in which the instructions of Easy Motion Control are called.
  "Sample_T" must be entered with the second unit in the axis DB.

Positioning Not Accurate Enough (S7-300, S7-400)

If positioning lacks precision:

- Check the settings in the "Steps per encoder revolution" parameter ("StepsPerRev").
- Calculate the "Axis distance per encoder revolution" parameter ("DisplacementPerRev") with maximum possible precision ("Distance measurement" wizard).

See also

Measuring the axis distance (Page 2759)

Maximum Following Distance Exceeded (S7-300, S7-400)

Overshoot of the maximum following error has different causes:

- In order for the "maximum axis velocity" ("MaxVelocity") to be reached at all, the following rules must apply:

$$„MaxFollowingDistance“ \quad >= \frac{„MaxVelocity“}{„FactorP“}$$

- The configured acceleration cannot be reached by the real axis.
- Incorrect settings for the "Maximum axis velocity" ("MaxVelocity") and "Reference value for maximum axis velocity" ("DriveInputAtMaxVel") parameters, or of the "Reference value for 100 % speed" ("DriveInputAt100") (refer to Axis velocity not as expected, or axis oscillates (Page 2761)).

The MC_GearIn instruction displays neither "InGear" nor "Coupled" (S7-300, S7-400)

The following causes prevent the MC_GearIn instruction from displaying "InGear" or "Coupled":

- The slave axis is about to couple to a moving master axis without having (yet) reached the gear speed.
- The slave axis has released the coupling due to an error.

MC_GearIn instruction displays "InGear", but not "Coupled" (S7-300, S7-400)

instruction MC_GearIn displays "InGear", but not "Coupled" due to the following causes:

- The slave axis may have reached the gear speed for (possibly only for one cycle), but has lost this speed:
  - The acceleration or deceleration of the master axis exceeds the capability of the slave axis based on its configuration and the gear ratio.
  - The slave axis cannot reach the velocity that is determined by the velocity setpoint of the master axis and by the gear ratio.

6.2.10.7 Diagnostics (S7-300, S7-400)

Error Displays and Error Acknowledgement in the Axis DB (S7-300, S7-400)

Error displays and error acknowledgment

All errors detected by the instructions are displayed in the axis DB as follows:

- By setting the group error "Error"
- By means of error-specific display "Err.xxxx"

With errors requiring acknowledgment, an error acknowledgment must be set after the error has been eliminated (parameter "ErrorAck" in the axis DB; see below).

There are three types of errors with different error responses:

- Errors where perfect functioning of position control is **no longer** guaranteed lead to a **hard stop**:
  instruction MC_Control brings the axis to a standstill along the deceleration ramp stored in "EmergencyDec". instruction MC_Control will then reset its "DriveEnabled" to "0".
  Once the error has been eliminated, set the error acknowledgment in the axis DB.
  "DriveEnabled" is then set again.
- Errors where position control still functions lead to a **soft stop**:
  The axis is slowed down at the deceleration that is configured in the active travel instruction.
  The axis retains controlled operation and "DriveEnabled" is **not** reset.
  In this situation, the movement can only be overridden by means of MC_StopMotion.
  Once the error has been eliminated, set the error acknowledgment in the axis DB.
- Error in the axis parameters (**parameter error**):
  instruction MC_Init checks the parameter entered in the Axis Db as far as possible. In the event of an error, it sets "Err.ConfigErr" bit and enters the precise cause of error in "Config.Err_xxx".
  Parameter errors **cannot** be rectified by acknowledgment. Approach of the axis is prevented. After having corrected the invalid parameter entry, you must once again call instruction MC_Init. Parameter error displays are then reset once there is no longer any error.

Note

The following applies to errors with hard and soft stop:

No travel order is possible as long as one or several unacknowledged errors are pending.

Exception:

If the axis only overshoots a software limit switch, a travel order for reversing the movement is possible. This error is automatically acknowledged once the axis has cleared the software limit switch.

Note

If a travel block (for example, MC_MoveJog, or MC_MoveAbsolute) has detected the error, the "Error" output is set at the corresponding instruction.

If a driver instruction has detected the error, the precise cause of error is indicated at this instruction (see Input drivers).

Error display in the axis DB

The group error is TRUE as long as there is at least one error.

Address	Name	Type	Initial value	Comment
124.1	Error	BOOL	TRUE	**Group error** FALSE = No error TRUE = Group error

Error acknowledgment in the axis DB

When group acknowledgment is set, all acknowledgeable error displays are deleted.

Group acknowledgment is only effective when the axis is at a standstill and after initialization.

On completion of the evaluation of error acknowledgment, "ErrorAck" is reset automatically.

Address	Name	Type	Initial value	Comment
124.2	ErrorAck	BOOL	FALSE	**Group acknowledgment** for all errors FALSE = No acknowledgment TRUE = Acknowledge error

Axis error (S7-300, S7-400)

Error with soft stop (S7-300, S7-400)

Description

The following rules apply to the errors described below: TRUE = Error is pending.

Errors requiring acknowledgment are identified by the following symbol:

The following table lists all errors with soft stop.

Address	Name: Err.xxx	Description		
128.0	SWLimitMinExceeded	**Software limit switch start exceeded**		—
		Cause:	• The actual position value is outside the working range. • Replacement of the absolute encoder.	
		Remedy:	• Move the axis out of the range of the limit switch. • Deactivate the software limit switch ("SWLimitEnable" = FALSE), synchronize the axis and then reactivate the software limit switch.	

Address	Name: Err.xxx	Description		Q
128.1	SWLimitMaxExceeded	**Software limit switch end exceeded**		—
		Cause:	• The actual position value is outside the working range. • Replacement of the absolute encoder.	
		Remedy:	• Move the axis out of the range of the limit switch. • Deactivate the software limit switch ("SWLimitEnable" = FALSE), synchronize the axis and the reactivate the software limit switch.	
128.2	TargetErr	**Target outside of travel range**		Q
		Cause:	• **MC_MoveAbsolute**: – The target is in the range of a software limit switch. – The target is < "Rotary axis start", or ≥ "Rotary axis end". – The calculated traversing distance is greater than the distance that is equivalent to 2^{24} steps of your encoder. – Auxiliary variables were not calculated in the axis DB, because instruction MC_Init was not called. • **MC_MoveRelative**: – The travel distance entered leads to a target on a software limit switch. – The specified travel distance is greater than the distance that is equivalent to 2^{24} steps of your encoder. – Auxiliary variables were not calculated in the axis DB, because instruction MC_Init was not called. • **MC_Home**: – The reference point coordinate is in the range of a software limit switch. – The reference point coordinate is < "Rotary axis start", or ≥ "Rotary axis end". – Auxiliary variables were not calculated in the axis DB, because instruction MC_Init was not called.	
		Remedy:	• Correct the parameters. • Call the MC_Init instruction.	
128.3	NoSync	**Axis not synchronized**		Q
		Cause:	instruction MC_MoveAbsolute was started with unsynchronized axis.	
		Remedy:	Synchronize the axis (reference point approach, reference setting).	
128.4	DirectionErr	**Invalid travel direction entered**		Q
		Cause:	Further movement into the range of a software limit switch is not allowed.	
		Remedy:	Start a travel order that moves the axis out of the range of the software limit switch.	

Address	Name: Err.xxx	Description		Q
128.5	DataErr	**Invalid parameter in travel instruction**		Q
		Cause:	**MC_MoveHome**: • "Velocity" < 0 • "Velocity" > 0 and "Acceleration" ≤ 0 • "Velocity" > 0 and "Deceleration" ≤ 0 • "Velocity" > 0 with absolute encoder • "Position" lies outside the numerical range on a linear axis. • Auxiliary variables were not calculated in the axis DB, because instruction MC_Init was not called. **MC_MoveAbsolute**, **MC_MoveRelative** and **MC_MoveJog**: • "Velocity" ≤ 0 • "Acceleration" ≤ 0 • "Deceleration" ≤ 0 • The entered or calculated target coordinate is outside the numerical range (for MC_MoveAbsolute and MC_MoveRelative). • Gear ratio denominator (RatioDenominator") ≤ 0 (MC_GearIn) • Gear ratio numerator ("RatioNumerator") = 0 (MC_GearIn) • A travel instruction was canceled and "Deceleration" of the overriding instruction is inconsistent with the deceleration configured at the canceled instruction (see Overriding a movement (Page 2716)). • Auxiliary variables were not calculated in the axis DB, because instruction MC_Init was not called.	
		Remedy:	• Specify admissible values. • Initialize the instructions. • Call the MC_Init instruction.	
128.6	StartErr	**Start from current axis status not possible**		Q
		Cause:	• In a travel order, or at MC_StopMotion, the Execute input was set while the axis was in error state or manual mode ("ManEnable" = TRUE), or while a MC_StopMotion instruction was active. • instruction MC_Home was not started while the axis was at a standstill. • Both directions are set in instruction MC_MoveJog. • Invalid status at the leading axis for instruction MC_GearIn: – Error with hard stop – MC_Home is active – Manual mode ("ManEnable" = TRUE)	
		Remedy:	• Eliminate any errors. • Make sure that the master axis is in a permitted state.	

Address	Name: Err.xxx	Description		Q
128.7	DistanceErr	**Overtravel**		Q
		Cause:	A movement without target (MC_Home, MC_MoveJog, MC_GearIn) has covered a distance that is equivalent to more than 2^{24} encoder steps. The axis is braked at about this point. A MC_MoveJog or MC_GearIn instruction has decelerated a synchronized linear axis onto a software limit switch. Moreover, "SWLimitMinExceeded" or "SWLimitMaxExceeded" is set as well in most cases.	
		Remedy:	Make sure the travel distance does not exceed the limit specified above (see Maximum Travel Distance (Page 2748)).	
129.0	MasterErr	**Master axis in invalid state**		Q
		Cause:	The master axis was set to an invalid state during gear coupling: • Error with hard stop • MC_Home is active • Manual mode ("ManEnable" = TRUE)	
		Remedy:	Correct the invalid state of the master axis.	

Error with hard stop (S7-300, S7-400)

Description

The following rules apply to the errors described below: TRUE = Error is pending.

Errors requiring acknowledgment are identified by the following symbol:

Q

The following table lists all errors with hard stop.

Address	Name: Err.xxx	Description		Q
130.0	StoppedMotion	**Axis is in stop state requiring acknowledgment**		Q
		Cause:	The axis is in a stop state requiring acknowledgment: • After initialization by calling instruction MC_Init • As the result of an error listed in this table	
		Remedy:	Rectify any errors that might possibly be present.	
130.1	EnableDriveErr	**Drive enable missing**		Q
		Cause:	"EnableDrive" is not set at instruction MC_Control.	
		Remedy:	Set "EnableDrive".	

Address	Name: Err.xxx	Description		Q
130.2	FollowingDistErr	**Following error exceeded**		Q
		Cause:	The axis is not following the setpoints or is too slow in following.	
		Remedy:	• Switch on the drive. • Optimize "Controller gain" (Axis DB address 44.0 (Page 2729)). • Adjust "Set encoder polarity" (Axis DB address 74.0 (Page 2728)) to "Set drive polarity" (Axis DB address 86.0 (Page 2731)). • Increase the valid maximum following error (Axis DB address 38.0 (Page 2740)).	
130.3	StandstillErr	**Outside standstill range**		Q
		Cause:	• The axis has moved out of "Standstill range" without a travel order. • "Standstill range" is less than "Target range". • "Set encoder polarity" or "Set drive polarity" is incorrectly configured.	
		Remedy:	• Check the connection to the setpoint input of the power unit. • Adjust the parameters "Standstill range"/"Target range" parameters (Axis DB address 34.0 (Page 2739) / 30.0 (Page 2738)) • During first commissioning: Test the wiring (Axis DB address 74.0 (Page 2728) / 86.0 (Page 2731)).	
130.4	TargetApproachErr	**Error on target approach**		Q
		Cause:	The actual value has not reached the "Target range" within the "Monitoring time for target approach".	
		Remedy:	• Optimize the "target range" and "Monitoring time for target approach" (Axis DB address 30.0 (Page 2738) / 26.0 (Page 2740)). • Check the drive and axis.	
130.5	EncoderErr	**Encoder error**		Q
		Cause:	• The input driver has detected an encoder error. The instance DB of the driver DB indicates the precise cause of error. See Input drivers. • Input "EncErr" was set at the input driver (see Input drivers, error handling section).	
		Remedy:	Eliminate the cause of the error.	
130.6	OutputErr	**Error at output driver**		Q
		Cause:	Input "OutErr" was set at the output driver (see Output drivers, error handling section).	
		Remedy:	Eliminate the cause of the error.	

Address	Name: Err.xxx	Description		Q
130.7	ConfigErr	**Group error: Axis data incorrectly configured**		**Acknowledgment not possible**
		Cause:	The parameters in the axis DB are defective. For exact cause of error, see Parameter errors (Page 2771).	
		Remedy:	Adjust the faulty parameter and make sure that instruction MC_Init is called.	
131.0	DriveErr	**Drive error**		**Acknowledgment not possible**
		Cause:	The output driver has detected an error at the power unit / drive. The instance DB of the driver DB indicates the precise cause of error. See Output drivers.	
		Remedy:	Eliminate the cause of the error.	

Parameter errors (S7-300, S7-400)

Parameter error

The following table shows the parameter errors that are likely to occur in Easy Motion Control.

The "Address" column contains the address of the error display in the axis DB. The "Parameter address" column points to the address of the faulty parameter in the axis DB (see Structure of the axis data block).

Address	Name Config.xxx	Parameter error	Parameter address
132.0	Err_AxisLimit	"Software limit switch start" < $-2^{24}+6$ [length unit] "Software limit switch end" > $+2^{24}-6$ [length unit] "Software limit switch end" ≤ "Software limit switch start"	6.0 (Page 2732) , 10.0 (Page 2732)
132.1	Err_MaxVelocity	"Maximum axis velocity" ≤ 0.0 [length unit/s]	14.0 (Page 2734)
132.2	Err_MaxAcceleration	"Maximum acceleration" ≤ 0.0 [length unit/s^2]	18.0 (Page 2735)
132.3	Err_MaxDeceleration	"Maximum deceleration" ≤ 0.0 [length unit/s^2]	22.0 (Page 2736)
132.4	Err_MonTimeTargetAppr	"Monitoring time for target approach" ≤ 0.0 [s]	26.0 (Page 2740)
132.5	Err_TargetRange	"Target range" ≤ 0.0 [length unit]	30.0 (Page 2738)
132.6	Err_StandstillRange	"Standstill range" ≤ 0.0 [length unit]	34.0 (Page 2739)
132.7	Err_MaxFollowingDist	"Maximum permitted following distance" ≤ 0.0 [length unit]	38.0 (Page 2740)
133.0	Err_EmergencyDec	"Deceleration for hard stop" ≤ 0.0 [length unit/s^2]	54.0 (Page 2737)
133.1	Err_StepsPerRev	"Steps per encoder revolution" < 1	64.0 (Page 2725)
133.2	Err_DisplacementPerRev	"Axis distance per encoder revolution" ≤ 0.0 [length unit]	68.0 (Page 2727)
133.3	Err_NumberRevs	"Number of encoder revolutions" < 1 for absolute encoder	72.0 (Page 2726)
133.4	Err_PolarityEncoder	"Encoder polarity" not equal to ±1	74.0 (Page 2728)
133.5	Err_PolarityDrive	"Drive polarity" not equal to ±1	86.0 (Page 2731)
133.6	Err_DriveInputAtMaxVel	"Reference value for maximum axis velocity" ≤ 0.0, or > "DriveInputAt100"	88.0 (Page 2730) , 96.0 (Page 2730)
133.7	Err_AxisLength	Axis length > 2^{24} [length unit], or axis length > (2^{31} -1) [steps]	6.0 (Page 2732) , 10.0 (Page 2732)

Address	Name Config.xxx	Parameter error	Parameter address
134.0	Err_EncoderRange	Encoder range does not match axis length (see "Axis Distance per Encoder Revolution", Axis DB address 68.0 (Page 2727))	6.0 (Page 2732) , 10.0 (Page 2732) , 68.0 (Page 2727) , 72.0 (Page 2726)
134.1	Err_MaxVelRotaryAxis	Maximum velocity and scan time do not match rotary axis length (see "Maximum axis velocity" Axis DB address 14.0 (Page 2734))	14.0 (Page 2734) , 0.0 (Page 2736) , 68.0 (Page 2727) , 72.0 (Page 2726)
134.2	Err_DriveInputAt100	"Reference value for 100 % speed" ≤ 0	96.0 (Page 2730)

JOB_ERR interrupts (S7-300, S7-400)

JOB_ERR interrupts

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgment on reading from module. Module removed during reading process or module defective
80A1	32929	-32607	Negative acknowledgment on writing to module. Module removed during write process or module defective
80A2 1)	32930	-32606	Protocol error at layer 2
80A3 1)	32931	-32605	Protocol error at user interface / user
80A4 1)	32932	-32604	Communication faulty at K-bus
80B0	32944	-32592	Data record / order unknown
80B1	32945	-32591	Length data incorrect. FM_TYPE parameter in the channel DB is not correctly set for the module being used.
80B2	32946	-32590	Configured slot is not assigned.
80B3	32947	-32589	Actual module type not same as setpoint module type.
80C0 1)	32960	-32576	Module has not yet prepared the data to be read.
80C1 1)	32961	-32575	Data of a similar write order has not yet been processed on the module.
80C2 1)	32962	-32574	Module is currently processing the maximum possible number of orders.
80C3 1)	32963	-32573	Required resources (memory, etc.) are currently occupied.
80C4 1)	32964	-32572	Communication error
80C5 1)	32965	-32571	Distributed I/O not available.
80C6 1)	32966	-32570	Priority class aborted (restart or background)
8522	34082	-31454	Channel DB or parameter DB too short. Data cannot be read from the DB (write order).
8532	34098	-31438	DB number of parameter DB too high (write order)
853A	34106	-31430	Parameter DB not available (write order).
8544	34116	-31420	Error at nth (n > 1) read access to a DB after an error has occurred (write order).
8723	34595	-30941	Channel DB or parameter DB too short. Data cannot be written to the DB (read order).

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
8730	34608	-30928	Parameter DB in CPU is write-protected. Data cannot be written to the DB (read order).
8732	34610	-30926	DB number of parameter DB too high (read order).
873A	34618	-30918	Parameter DB not available (read order).
8745	34629	-30907	Error at nth (n > 1) write access to a DB after an error has occurred (read order).

1) Errors 80A2 to 80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time.

SIMATIC Ident 7

7.1 Configuring Ident systems

7.1.1 Technology object "SIMATIC Ident"

Introduction

The technology object for SIMATIC Ident products allows simple parameter assignment of an Ident system in the TIA Portal. It provides an interface with simple parameter assignment between the configured hardware and the function blocks in the "SIMATIC Ident" library.

With the aid of the "Configuration" object you can select a configured device, decide on the channel / the read point and if necessary specify the connected Ident system. The technology object therefore automatically determines all the parameters required for operation.

The "Diagnostics" object displays the last 5 error messages signaled by the Ident device to the function blocks. In addition to this for every error message an error description and detailed information about the executed command are displayed.

The following overview shows the basic procedure for adding and configuring a "SIMATIC Ident" technology object.

Procedure

Follow the steps below to configure the "SIMATIC Ident" technology object:

Step	Description
1	Adding a technology object (Page 2775)
2	AUTOHOTSPOT
3	Calling the Ident instruction in the user program (Page 2783)
4	Load technology object in CPU
5	Diagnosing devices (Page 2785)

7.1.2 Adding a technology object

Adding a technology object in the project navigator

When you add a technology object, a data block is generated. The configuration of the technology object is stored in this data block.

Requirements

To use the "SIMATIC Ident" technology object the following requirements must be met:

- A project (V14 SP1 or higher) was created.
- According to the hardware used, the connected devices were added to the hardware configuration ("Devices & Networks"), networked with each other and configured.
- The project contains at least one S7 controller and at least one Ident device (e.g. communications module, reader, optical reader).
  The following controllers are compatible:
  - S7-1200 as of V4.0
  - S7-1500 as of V1.8
  - ET200pro (CPU 1516pro) as of V1.8
  - ET200SP as of V1.8

Procedure

Follow the steps below to add a technology object:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click on ""Add new object".
   The "Add new object" dialog box opens.
4. Click the "SIMATIC Ident" button.
5. In the "Version" column select the newest version (as of "V5.0").
6. Select the object "TO_Ident".
7. Type in an individual name for the technological object in the "Name" input field.
8. Click "More information" if you want to store your own information for the technology object.
9. Confirm with "OK".

Result

The new technology object is created and saved to the "Technological objects" folder in the project tree.

7.1.3 Configuring devices with the aid of the technology object

7.1.3.1 Working with the configuration dialog

You configure the properties of the technology object in the configuration window. To open the configuration window of the technology object, follow these steps:

1. In the project tree, open the "Technological objects" folder.
2. Open the technology object the project tree.
3. Double-click on the "Configuration" object.

The configuration is divided into the following categories:

- Basic parameters
- Ident device parameters
- Reader parameters

Symbols of the configuration window

Symbols in the area navigation of the configuration and Inspector window show additional information about the completeness of the configuration:

	The configuration contains default values and is complete The configuration only contains default values. With these preset values you can use the technology object without additional changes.
	The configuration contains values defined by the user or automatically adapted values and is complete All input fields of the configuration contain valid values and at least one default value was modified.
	The configuration is valid but contains unexpected values At least one input field or a drop down list contains an unexpected value. The corresponding field, or the drop-down list is displayed on a yellow background. When you click on it, the roll-out warning message shows you the cause of the warning.
	The configuration is incomplete or incorrect At least one input field or a drop down list contains no or an invalid value. The corresponding field, or the drop-down list, is displayed on a red background. Click the roll-out error message to identify the cause of the error.

7.1.3.2 Configuring basic parameters

In this parameter group, you can configure the module-specific parameters of your system.

Table 7-1 Parameters in the "Basic parameters" parameter group

Parameter	Description
Ident device	Click "..." and select the Ident device to be configured in the folder "Distributed I/O" or "Local modules". The following devices can be selected: • ASM 456 • RF120C, RF170C, RF180C • RF18xC • RF615R • RF680R, RF685R • MV400, MV500
Channel	Open the drop-down list and, if needed, select, the channel of the communications module to which the device to be configured is connected.
Reader parameter assignment	Open the drop-down list and select the device/system connected to the Ident device that needs parameter assignment.
Read point	Requirement: An RF600 reader was selected in the "Reader parameter assignment" parameter. Open the drop-down list and select the read point to be used.

7.1.3.3 Configuring Ident device parameters

Depending on the selected Ident device, the following Ident device parameters are displayed in this parameter group. The displayed Ident device parameters are adopted automatically from the device configuration of the connected devices.

Note that not all parameter values adopted and displayed from the device configuration of the connected devices are permitted in conjunction with the technology object. If non-permitted

values are set in the device configuration you can change these either automatically with the "Autoconfiguration" button or manually using the device configuration.

Table 7-2 Parameters in the "Ident device parameters" parameter group

Parameter	Possible displayed parameter value	Permitted parameter values	Description
User Mode	Ident profile/RFID standard profile FB 45 / FC 45 FB 55 / FC 55	Ident profile/RFID standard profile	With this parameter, you select the block. Note that only the following parameter values are permitted in conjunction with the technology object. • Ident profile/RFID standard profile: The program block for the Ident profile/RFID standard profile is used on the controller.
MOBY Mode	RF200/RF300/RF600; MV400/MV500; MOBY U/D RF680R/RF685R MOBY I/E Normal addressing RF300 file handler MOBY U file handler MOBY I file handler	RF200/RF300/RF600; MV400/MV500; MOBY U/D RF680R/RF685R	With this parameter, you set the mode of the communications module. Note that only the following parameter values are permitted in conjunction with the technology object. • RF200/RF300/RF600; MV400/MV500; MOBY U/D • RF680R/RF685R
Connected device	Parameters via FB / optical readers	Parameters via FB / optical readers	This parameter is only displayed in conjunction with the communications module RF120C. Note that only the following parameter values are permitted in conjunction with the technology object. • Parameters via FB / optical readers

"Autoconfiguration" button

The displayed HWCN parameters are adopted automatically from the device configuration of the connected devices. Note that not all parameter values are permitted in conjunction with the technology object. Non-permitted parameter values are highlighted in color.

Click the "Autoconfiguration", button to set the suitable parameter values automatically (e.g. User Mode). The parameter values of the connected device are prepared for the use of the technology object.

Note

Behavior with MV400/MV500

Note that the "Autoconfiguration" button does not work with the optical readers MV400 and MV500. With these devices you need to enter the HWCN parameters in the device configuration of the connected devices manually.

7.1.3.4 Configuring reader parameters

Depending on the set basic parameters you can configure the following reader parameters in this parameter group.

Table 7-3 Overview of the "Reader parameters"

Reader parameter assignment in "Basic parameters"	Transmission speed	Presence check	Reset error LED	HF power	Max. number of transponders	Transponder type			Program selection with reset	Byte sequence of the reset parameter [hex]
						ISO Mode	RF300	MOBYE		
RF200 general	✓	✓	✓	-	✓	✓	-	-	-	-
RF290R	✓	✓	✓	✓	✓	✓	-	-	-	-
RF300 general	✓	✓	✓	-	✓	✓	✓	-	-	-
RF380R	✓	✓	✓	✓	✓	✓	✓	-	-	-
RF300 Gen 2 general	✓	✓	✓	-	✓	✓	✓	✓	-	-
RF380R Gen 2	✓	✓	✓	✓	✓	✓	✓	✓	-	-
RF615R [1]	-	-	-	-	-	-	-	-	-	-
RF680R [1]	-	-	-	-	-	-	-	-	-	-
RF685R [1]	-	-	-	-	-	-	-	-	-	-
MV400 [1]	-	-	-	-	-	-	-	-	✓	-
MV500 [1]	-	-	-	-	-	-	-	-	✓	-
General reader	✓	-	-	-	-	-	-	-	-	✓

[1] The device-specific parameters of these devices are set in the WBM of the relevant devices.

Table 7-4 Parameters in the "Reader parameters" parameter group

Parameter	Parameter value	Default value	Description
Transmission speed	19.2 kBd 57.6 kBd 115.2 kBd	115.2 kBd	The selection depends on the communications module and Ident system being used. Note that the value specified here is adopted automatically from the device configuration of the connected devices. With this parameter, you set the transmission speed between the communications module and reader. When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power). When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.
Presence check	On Off (RF field on) Off (RF field off)	On	On = As soon as there is a transponder in the antenna field of the reader, its presence is reported. Off (RF field on) = the presence check in the FB is suppressed. The antenna on the reader is nevertheless turned on as long as it has not been turned off by a command. Off (RF field off) = the antenna is turned on only when a command is sent and it then turns itself off again.
Reset error LED	On Off	On	On = the flashing of the error LED on the communications module is reset by each FB reset. Off = the error LED always indicates the last error. The display can only be reset by turning off the communications module.
HF power [1]	0.50 ... 2.00 0.50 ... 5.00	1.00 1.25	Setting for the output power of the reader. The selectable values depend on the connected device.
Max. number of transponders	1 ... 4	1	Number of transponders expected in the antenna field. The selection depends on the connected device.
Reader mode	Normal operation P2P master P2P slave	Normal operation	Selection of the required reader mode.
Transponder type	[1]	[1]	Selection of the transponder types used. The selection depends on the connected device.
Program selection with reset	No program change 1 ... 15	No program change	Selection of the program that will be used after restarting the optical reader. If the parameter "No program change" is set, after restarting the device the last active program is selected.
Byte sequence of the reset parameter [hex]	[1]	[1]	Hexadecimal representation of the byte sequence of the reset parameter of the reader.

[1] You will find a detailed description of this parameter in the following paragraphs:

The "HF power" parameter

The selectable values depend on the value specified in the parameter "Reader parameter assignment".

- RF290R
  - Range of values: 0.50 ... 5.00 W
  - Default value: 1.00 W
- RF380R; RF380R Gen 2
  - Range of values: 0.50 ... 2.00 W
  - Default value: 1.25 W

The "Transponder type" parameter

Selection of the transponders used. The selection depends on the value specified in the parameter "Reader parameter assignment". The following transponder types can be selected:

- ISO Mode
- RF300
- MOBY E

If you select "ISO mode", the following selection options are displayed depending on the reader parameter assignment:

Reader parameter assignment	Values
RF200 general	ISO 15693 general
RF290R	ISO 15693 general ISO 15693 (Infineon, MDS D3xx)
RF300 general RF380R	ISO 15693 general ISO 15693 (Infineon, MDS D3xx) ISO 15693 (Fujitsu - 4 kB, MDS D4xx) ISO 15693 (NXP, MDS D1xx) ISO 15693 (TI, MDS D2xx) ISO 15693 (STM, MDS D261)
RF300 Gen 2 general RF380R Gen 2	ISO 15693 general ISO 15693 (Infineon, MDS D3xx) ISO 15693 (Fujitsu - 4 kB, MDS D4xx) ISO 15693 (NXP, MDS D1xx) ISO 15693 (TI, MDS D2xx) ISO 15693 (STM, MDS D261) ISO 15693 (Fujitsu - 8 kB, MDS D5xx)

The parameter "Byte sequence of the reset parameter [hex]"

With this parameter, you can specify the Reset parameter in hexadecimal format. This setting is intended only for trained users.

The following Reset parameters are available:

Byte	1	2...5 [1]	6	7...8	9	10	11	12	13...14	15	16
Value	0x04	0x00	0x0A	0x00	scan-ning_ time	param	op-tion_1	distance_ limiting	multitag	field_on_ control	field_on_ time

[1] In the communications module RF180C as of V2.2 in conjunction with MOBY U byte 4 is preset with the calendar week and 5 with the year.

7.1.4 Calling the Ident instruction in the user program

Procedure

Proceed as follows to call the instruction in the user program:

1. Open the CPU folder in the project tree.
2. Open the "Program blocks" folder.
3. Double-click on the OB for cyclic program execution.
   The block is opened in the working area.
4. In the "Instructions" window open the group "Optional packages" and the "SIMATIC Ident" folder.
   The folder and the subfolders contain the instructions.
5. Select an instruction and drag it to your OB.
   The "Call options" dialog is opened.
6. From the "Name" list select an instance data block or enter the name for a new instance data block.
7. Confirm with "OK".
8. Select the configured technology object and drag it to the parameter "HW_CONNECT".

Reset the readers using the "Ident profile" or the "Reset_Reader" block

There are various methods for resetting readers:

- If you work with the Ident blocks, reset all reader types with the "Reset_Reader" block. This transfers the values set in the technology object to the readers.
- If you work with the Ident profile, when you set the "INIT" bit, the values set in the technology object are transferred to the readers.
- If you want to send your own "WRITE CONFIG" command, instead of the "INIT" bit you need to use the "EXECUTE" bit.

If a Reset parameter of the technology object needs to be changed at runtime, the relevant value can be overwritten in the data block of the technology object using the user program. The Reset parameters are located in the data block of the technology object in "Static.CONFIGURATION.param_buffer[]".

You will find detailed information on the Ident profile in the manual "Ident Profile and Ident Blocks, Standard Function for Ident Systems" on the pages of "Industry Online Support (https://support.industry.siemens.com/cs/ww/en/ps/14971/man)".

7.1.5 Loading the configuration on the CPU

After configuration you need to load the changes from the technology object or the S7 controller on the connected devices.according to

Load to device

When loading to the devices note that a distinction is made between "Software (changes only)" and "Hardware and software (changes only)". If only changes to the software were made, loading is via the technology object. If changes were made to the hardware and software, loading is via the S7 controller.

NOTICE
Load to device when using the "Autoconfiguration" button If the "Autoconfiguration" button was used during configuration, you need to load the changes on the connected devices via the S7 controller ("Hardware and software (changes only)").

7.2 Diagnosing Ident systems

After configuration of the technology object is complete, you can monitor the devices configured in the technology object using the diagnostics function.

Requirements

- There is an online connection between STEP 7 (TIA Portal) and the S7 controller.
- The Ident instruction is called in the user program.
- A technology object was created and loaded on the S7 controller-
- The S7 controller is in "RUN" mode.

Procedure

To open the display editor for the diagnostics function, follow these steps:

1. In the project tree, open the "Technological objects" folder.
2. Open the relevant technology object in the project tree.
3. Double-click on the "Diagnostics" object.
4. Click on the symbol."Monitor all" to activate the monitoring mode.

7.2.1 Block diagnostics

In Block diagnostics you can display the last error messages to occur during command execution.

The following values are read by the technology object and displaed:

Table 7-5 Displayed values of "Block diagnostics"

Displayed values	Description
Date and time	Shows the date and time when the error occurred.
Command	Shows the command with which the error occurred.
Error description	Shows a brief description of the error that occurred.
	Note: The last five error messages are displayed and described in the table. If a new error message is displayed, the previously displayed error messages are moved one line down. A maximum of five error messages are displayed.
Error message	Shows the error message of the error that occurred
Error description	Shows a detailed description of the error that occurred
Date and time	Shows the date and time when the error occurred.
Command code	Show the command code of the command with which the error occurred.
Command	Shows the command with which the error occurred.
Attribute	Shows the attribute of the command with which the error occurred.

7.2.2 Hardware diagnostics

In Hardware diagnostics you can display the status parameters of the selected reader and of the transponder currently in the antenna field.

Monitoring status

In this area you can update the reader and transponder parameters and reset error counters of the reader.

Reader status

The following parameters are read by the technology object and displayed:

Table 7-6 Displayed parameters of the "Reader status" area

Displayed parameters	Description	
Hardware type	Hardware type of the reader	
Hardware version	Hardware version of the reader	
Bootloader version	Bootloader version of the reader	
Firmware version	Firmware version of the reader	
Driver type	Driver type of the serial interface of the reader	
Driver version	Driver version of the serial interface of the reader	
Interface	Used serial interface of the reader Possible values: RS232, RS422	
Transmission speed	Used transmission speed of the reader Possible values: 19.2; 57.6; 115.2 kBd	
Distance limiting	Set transmit power of the reader (RF380R) The following values are possible:	
	Value	Meaning
	02	0.5 W
	03	0.75 W
	04	1.0 W
	05	1.25 W
	06	1.5 W
	07	1.75 W
	08	2.0 W
Max. number of transponders	Maximum number of transponders to be expected that may be located in the antenna field at the same time.	

Displayed parameters	Description	
Transponder type	Set transponder type profile The following values are possible:	
	Value	Meaning
	00	RF300 (RF3xxT)
	01	ISO 15693 general
	03	ISO 15693 (Infineon, MDS D3xx)
	04	ISO 15693 (Fujitsu - 2 kB, MDS D4xx)
	05	ISO 15693 (NXP, MDS D1xx)
	06	ISO 15693 (TI, MDS D2xx)
	07	ISO 15693 (STM, MDS D261)
	08	ISO 15693 (Fujitsu - 8 kB, MDS D5xx)
	10	RF300 (RF3xxT)
	20	ISO 14443 (MOBY E, E6xx)
	31	General Mode
	40	P2P master
	41	P2P master & mixed operation with ISO transponder (MDS Dxxx)
	50	P2P master & mixed operation with RF300 transponder
	4F	P2P slave
Antenna status	Status of the antenna	
Presence check	Set presence check profile	

Transponder status

The following parameters are read by the technology object and displayed:

Table 7-7 Displayed parameters of the "Transponder status" area

Displayed parameters	Description
UID	Unique identifier of the transponder
Transponder type	Transponder type (vendor, identification)
Version	Version of the transponder chip
Lock status of the OTP area	Disabled blocks of the OTP areaon the chip
Size of the user memory	Memory size of the user memory in bytes
Size of a memory block	Size of the memory blocks of the transponder chip
Number of memory blocks	Number of memory blocks of the transponder chip
Measured value power flux density	Radiant power that arrives at the transponder. The lower the value, the more power the transponder receives.
Passive error counter	Number of passive errors which occurred
Active error counter	Number of errors which occurred Sum of the signature and CRC errors
Presence counter	Time in [ms] that the transponder spent in the antenna field.

Error counter

The following error types are read by the technology object and displayed:

Table 7-8 Error types displayed in the "Error counter" area

Displayed error types	Description
FZP	Passive error counter This error counter is an indicator for a disturbed environment (interferences). Only relevant for the transponder type "RF300".
ABZ	Abort counter Counter for protocol errors on the air interface for which the transponder aborted communication.
CFZ	Code error counter Counter for disruptions or collisions on the air interface that caused communication to be disturbed.
SFZ	Signature error counter Counter for failed signature encryptions of written blocks of data
CRCFZ	CRC error counter Counter for failed CRC checks
ASMFZ	Error counter for problems on the interface to the host (CM/PC) Counter for errors on the serial interface
Signal strength	
AMLI	AM power indicator Received signal strength [dB] for which amplitude modulation with the transponder was detected.
PMLI	PM power indicator Received signal strength [dB] for which phase modulation with the transponder was detected.

Note

Error counter readings

The displayed error counter readings are counted from the last restart of the reader or the last manual reset of the error counter readings.

Index

"

"Interrupt" gate function, 202

1

1Count24V, 316
1Count5V, 376
1PosU
  Absolute positioning (MODE 5), 2602
  Activating JOBs, 2589
  Axis type, 2629, 2635, 2641
  Cancel JOB processing (JOB 0), 2606
  Causes of JOB_ERR error, 2625
  Causes of POS_ERR error, 2624
  Changeover difference, 2633, 2639, 2644
  Changing the changeover difference (JOB 4), 2611
  Changing the switch-off difference (JOB 3), 2610
  Control interface, 2645
  Controlling the Drive, 2632, 2638, 2643
  CPU/Master STOP, 2626
  Defining the settings for monitoring the direction of rotation (JOB 11), 2615
  directional enables, 2591
  Displaying current values (JOB 15), 2617
  Dosing mode, 2587, 2628, 2634
  Encoder inputs, 2634
  Encoder type, 2640
  End of rotary axis, 2629, 2635
  Errors in the control of MODEs and JOBs, 2623
  Evaluating the reference signal (JOB 9), 2612
  Evaluation of the encoder signals, 2590
  External errors, 2620
  EXTF_ACK error acknowledgment, 2623
  Feedback interface, 2647
  Functions, 2586
  Hardware limit switches, 2630, 2636, 2642
  Inching (MODE 1), 2592
  Increments per encoder revolution, 2640
  Number of encoder revolutions, 2641
  Parameter assignment error, 2620
  Reduction cam, 2631, 2637, 2643
  Reference point approach (MODE 3), 2595
  Reference signal, 2631, 2637
  Reference switch, 2631, 2637
  Relative positioning (MODE 4), 2600
  RESET state, 2627
  Reversal of direction of rotation, 2629, 2634, 2641
  Setting actual value (JOB 1), 2607
  Shifting the encoder range (JOB 2), 2609
  Start direction for reference point approach, 2632, 2638
  Starting MODEs, 2588
  Stop (MODE 0), 2592
  Switch-off difference, 2632, 2638, 2643
  Tmin direction change, 2633, 2639, 2644
  Transmission rate, 2640
1SSI
  Encoder value acquisition, 2350, 2366, 2378
1STEP 5V
  Absolute incremental mode, 2666
  Acceleration / delay a, 2655
  Axis type, 2670
  Base frequency, 2655
  Changing Parameters during Operation, 2673
  Digital input DI0 (3), 2674
  Digital input DI1 (7), 2675
  Hold traversing job, 2668
  Maximum frequency/velocity of the axis Fmax, 2654
  Modulo axis, 2670
  Output frequency/velocity Fa, 2654
  Pulse enable, 2672
  Reaction to CPU/Master STOP, 2676
  Reference point approach, 2659
  Reference point approach, process, 2660
  Relative incremental mode, 2665
  Set home position, 2664
  Speed-control mode, 2667
  Starting a traversing job, 2658
  Start-stop frequency Fss, 2653
  Traversal curve, 2653

2

24 V encoder signals, 205
  Input filter, 222, 223, 297, 298
2PULSE
  Direct control of the DO, 1136
  Frequency output, 1129
  Measurement of current, 1135
  On/Off-delay, 1121
  Parallel connection of both channels, 1139
  Pulse output, 1100

Pulse train, 1116
Pulse width modulation, 1106
Sequence counter, 1135

5

5 V encoder signals, 204
Monitoring, 205

6

611U-compliant mode, 2700

A

Absolute actual value, 1639, 1640
Absolute encoder, 22, 448, 2389
Frame run time, 452
Increments per encoder revolution, 2454, 2553
Listen-in, 451
Message frame times, 2392
Monoflop time, 452, 2392
Pulse evaluation, 451, 2391
Reaction times, 451
Response times, 2392
Absolute encoder adjustment, 488, 564, 1671, 2449, 2547
Determining, 489, 565, 2449, 2547
Example, 490, 566, 2450, 2548
Mechanical adjustment, 490, 566, 2450, 2548
Absolute positioning
Easy Motion Control, 2715
Absolute value adjustment, 1687
Accuracy, 1157
Calculating TIO_Time, 1162
Active homing, 1671, 1673, 1677, 1678, 1820
Actual encoder value
Additional, 2705
Actual value change, 460, 535
Actual values
Easy Motion Control, 2753
Actuator, 1625
Add empty line, 1408
Add separator line, 1408
Additional encoder actual value, 2705
Additive setpoint torque, 1668
adjustment
Dynamic, 445
Application cycle, 1161, 1166
Application examples Time-based IO, 1155
Articulated arm
2D, 2191
2D with orientation, 2193
3D, 2196
3D with orientation, 2201
Assignment
Diagnostic data record DS0, 1059
Diagnostic data record DS1 of the FM 355-2, 1060
AxesGroup, 2149
Axis
Absolute encoder adjustment, 488, 564, 2449, 2547
Axis type,
Axis types, 485, 561
End of rotary axis, 2447, 2545
Hysteresis, 492, 568
Minimum edge distance, 2550
Minimum edge-to-edge distance, 494, 570
Reference point coordinate, 488, 564, 2448, 2546
Simulation speed, 493, 569
Software limit switch, 485, 561, 2445, 2543
Start speed for reference point approach, 2452, 2550
Type of reference point approach, 2447, 2545
Axis and command table technology object
List of ErrorIDs and ErrorInfo, 1466, 1549
Axis control panel, 1949, 1953, 1996
Axis data block
Loading, 2752
Saving, 2752
Axis DB
Function, 2751
Axis does not move
Easy Motion Control, 2760
Axis moves without travel command
Easy Motion Control, 2761
Axis type, 1631, 2445, 2543
Axis types, 485, 561

B

Basic parameters
High_Speed_Counter, 84
SSI_Absolute_Encoder, 108
Technology module, 129
Behavior of a stepper motor, 2696
Behavior of the digital outputs
Active on reaching the comparison value for pulse duration down, 233, 307

Active on reaching the comparison value for pulse duration up, 232, 307
Active on reaching the comparison value for pulse duration up/down, 232, 306
Basic conditions, 235
Boundary conditions, 308
Default setting, 230
in counting modes, 228
in isochrone mode, 230
in measuring modes, 238
No comparison, 231, 305
On when counter value <= comparison value, 231, 306
On when counter value >= comparison value, 231, 306
Switch at comparison values, 233
Blocked zones, 2255, 2257
Bounce-free switch, 2522
Brake cam track, 443

C

Call sequence, 1167
Cam, 1760, 1853
Basics, 1760
Configuration, 1853, 1856, 1889
Effective direction, 511, 587
Functions, 1628
Hardware interrupt, 513, 589
Import/export, 1889
Interpolation, 1762
Switching characteristics, 512, 588
Tags, 2101
Cam controller, (Application examples Time-based IO)
Cam cycle time, 505, 581
Cam data
Reading, 456
Writing, 455
Cam editor, 1713, 1853, 1856, 1889
Cam track
Basics, 1747
Configuration, 1850
Diagnostics, 1983, 1984
Functions, 1627
Tags, 2095
Cam tracks, 441
Camming, 1713
Cams
Activation time, 508, 584
Cam "valid", 506, 582
Cam data, 505, 581
Cam end for position-based cams, 507, 583
Cam end for time-based cams, 508, 584
Cam start for position-based cams, 507, 583
Cam start for time-based cams, 508, 584
Cam type, 507, 583
Directional reversal, 445
Dynamic adjustment, 510, 586
Lead time, 509, 585
Number of cams, 505, 581
Track number, 506, 582
Cancel, 2407, 2499
Incremental approach, 2423, 2516
Jog, 2410
Jogging, 2502
Reference point approach, 2415, 2507
Cancel gate function, 202
Cancel set actual value, 2425, 2518
Data in the channel DB, 2518
Canceling
Zero offset, 2520
Cancelling the function
Undo set process value, 462, 538
Capture, 21, 22, 28, 31, 91, 112, 146, 158
Change cam edges, 467, 543
Canceling, 468, 544
Data used in the channel DB, 467, 543
Definition, 467, 543
Effects, 467, 543
Requirements, 467, 543
Steps, 467, 543
Changeover difference, 2439, 2537
Changeover difference minus, 2440, 2538
Changeover difference plus, 2440, 2538
Changing
Incremental dimension tables, 2400, 2492
Machine data, 2398, 2490
Channel configuration, 1272, 1282
Technology module, 1172
Channel setting
FM 350-2, 264
Checkback signals for the diagnostics, 2432, 2530
Data in the channel DB, 2433, 2530
Sequence, 2433, 2530
Checkback signals for the positioning, 2432, 2529
Data in the channel DB, 2432, 2529
Sequence, 2432, 2529
Checkback signals in the channel DB
End of a positioning operation, 2408, 2500
CircMode, 2250
Circular motion, 2249
Circular path
Definition, 2250

Closed loop control, 1379
Closed loop position control, 1693, 1694, 1695, 1830, 1831
Closed-loop control, 1693, 1694, 1695, 1830, 1831
CM 4xIO-Link, (IO-Link Device)
Command table technology object
    Add new object, 1402
    Basic parameters, 1404
    Command table configuration, 1405
    Configuration window icons, 1403
    Configuring activate warnings, 1405
    Configuring duration, 1407
    Configuring position / travel path, 1406
    Configuring the command type, 1405
    Configuring the next step, 1407
    Configuring the step code, 1407
    Configuring use axis parameters of, 1405
    Configuring velocity, 1406
    Extended parameters, 1416
    General configuration, 1404
    Tag command[1...32], 1526, 1620
    Tag Config.Command.Command[1...32], 1619
    Tools, 1401
    Usage, 1401
Commissioning
    High_Speed_Counter, 103
    SSI_Absolute_Encoder, 123
Compact CPU
    Performance features, 11, 13, 1185
Compact CPU S7-1500, (Technology module)
Comparison value, 39, 95, 114, 149, 159, 229, 304
    Counting, 95, 149
    Digital input module, 76
    Position input, 114, 159
Compatibility list, 2145
Configuration, (System environment)
    Standard configuration, 1163
Configuration software, 1159
Control interface, 166
    1Count24V for counting modes, 337
    1Count24V for measuring modes, 354
    1Count24V for position feedback, 366
    1Count5V for counting modes, 398
    1Count5V for measuring modes, 413
    1Count5V for position feedback, 427
    1SSI in standard mode, 2375, 2376
    1STEP 5V, 2683
    2PULSE (long), 1147
    2PULSE (short), 1146
    FM 350-1 for counter modes, 243
    FM 350-1 for measuring modes, 249
    FM 351, 2463
    FM 352, 515
    FM 451, 2562
    FM 452, 591
Control jobs, 2691
Control mode 1, 2436, 2534
Control mode 2, 2437, 2534
Control mode 3, 2437, 2535
Control mode 4, 2438, 2536
Correction distance, 445
Count continuously, 182
Count once, 183
Count periodically, 184
Counter cam track, 443
Counter limits, 20, 23, 73, 176
Counting
    Comparison value, 95, 149
    Continuous, 182
    Periodic, 184
    Single, 183
Counting functions, 20, 69, 70, 73
Counting limits, 69, 70, 90, 144
Counting range, 217, 294
Coupling factor, 2241
CPU cycle time, 1155
CPU-STOP, 129, 1274, 1291
Current control function, 1245
    dithering and cycle time,
    limit reached flags, 1247
    PID function diagram, 1245
    PID parameters, 1246
    resetting the PID controller, 1248
    set point control, 1247
Current measurement function, 1243
    Accuracy, 1244
Cycle time, 1156
Cylindrical robot
    3D, 2220
    3D with orientation, 2224

D

Data check, 470, 546
    by the module, 470, 546
Data error, 258, 315
Data in the channel DB
    Cancel set actual value, 2425, 2518
    Checkback signals for the diagnostics, 2433, 2530
    Checkback signals for the positioning, 2432, 2529
    Edge detection, 2523
    Enable input, 2431, 2528

Encoder data, 2432, 2529
Incremental approach, 2421, 2423, 2514, 2516
Incremental dimension tables, 2400, 2492
Jog, 2409
Jogging, 2501
Length measurement, 2523
Loop traverse, 2428, 2525
Machine data, 2399, 2490
Position data, 2431, 2528
Reference point approach, 2414, 2506
Set actual value, 2424, 2517
Set reference point, 2426, 2521
Setting actual value on the fly, 2517
Zero offset, 2519
Data in the parameter DB
Edge detection, 2524
Enable input, 2528
End of a positioning operation, 2408, 2500
Incremental approach, 2422, 2515
Reference point approach, 2415, 2507
Data type
DB_Any, 1929
Data update, (PIP_Mode)
DB_Any, 1929
Default setting, 1164
Diagnostic interrupt, 1058
Definition
Diagnostic interrupt, 1057
Delay time, 1161
Delete residual distance, 2423, 2516
Delta picker
2D, 2207
2D with orientation, 2210
3D, 2213
3D with orientation, 2216
Determining a coordinate
Zero offset, 463, 539
Determinism, (Predictability)
Diagnostic data record, 1009, 1010, 1058, 1090
Diagnostic data record DS0
Assignment, 1011, 1059, 1092
Diagnostic data record DS1 of the FM 355-2
Assignment, 1060
Diagnostic DB
Configuration, 522, 599
Data, 522, 599, 2472, 2571
Structure, 2472, 2571
Diagnostic events, 521, 599, 2471, 2570
Diagnostic interrupt, 59, 255, 281, 312
Default setting, 1009, 1058, 1090
Definition, 1009, 1057
Enable, 131, 1173
In event of error, 1089
OB 82, 1009, 1058, 1090
trigger, 1008, 1009, 1057, 1089
triggered, 1057
What is a diagnostic interrupt?, 1057
Diagnostic interrupts, 519, 596, 2468, 2567
Evaluation, 2471, 2570
Diagnostic record DS1
Assignment, 1012, 1060, 1093
Diagnostics
Easy Motion Control, 2716, 2763
High_Speed_Counter, 104
S7-1500 Motion Control, 1961, 1965
SSI_Absolute_Encoder, 124
Technology module, 165, 178, 1184, 1281, 1288
Diagnostics text, 1010, 1058, 1091
Digital input
Configuration, 1180
Functions, 91, 112, 146, 158
Operating mode, 1180
Digital input module
Applications, 72
Hardware configuration, 173
Performance features, 11, 13
Digital input/digital output module
Performance features, 14
Digital inputs
Input filter, 223
Input filters, 298
Digital output
Comparison value, 39, 95, 114, 149, 159
Functions, 95, 114, 149, 159
High-speed output, 1174
Operating mode, 1174, 1177, 1183
Digital output module
Performance features, 1185
Digital outputs
Behavior in counting modes, 228, 304
Behavior in measuring modes, 238
Enable, 229, 238
Enabling, 305
Setting and resetting, 229
Direct control of a digital output, 1252
Direct homing, 1671, 1686
Direction of rotation signal, 2693
Direction output and travel direction relation, 1304
Direction reversal, 2428, 2526
Direction reversal at the hardware limit switch, 1672, 1686, 1822
Distance measurement
Easy Motion Control, 2759

Distance-to-go, 2422, 2515
Dither PWM output function, 1248
amplitude, 1248
example diagrams, 1250
period, 1249
ramp-up and ramp-down, 1249
Dosing, 197, (Application examples Time-based IO)
Double evaluation, 207
Drive
Changeover difference, 2439, 2537
Compatibility list, 2145
Control mode 1, 2436, 2534
Control mode 2, 2437, 2534
Control mode 3, 2437, 2535
Control mode 4, 2438, 2536
Machine data, 2435, 2533
Monitoring time, 2443, 2541
Standstill range, 2443, 2541
Standstill velocity, 2444, 2542
Switch-off difference, 2440, 2538
Target range, 2441, 2539
Drive connection S7-1200 Motion Control, 1308
Drive type, 2692, 2693, 2699
Drives compatibility list, 1774
DSC (Dynamic Servo Control), 1694, 1695, 1830
Dynamic adjustment, 445, 510, 586
Dynamic default values, 1660, 1811
Dynamic Servo Control (DSC), 1694, 1695, 1830

E

Easy Motion Control
Absolute encoder adjustment, 2715
Absolute positioning, 2715
Actual values, 2753
Axis DB, 2751
Axis does not move, 2760
Axis moves without travel command, 2761
Axis oscillates, 2761
Axis parameters, 2753
Bit array for initialization, 2755
Call environment, 2743
Call sequence of the instructions, 2744
Common parameters for all instructions, 2747
Diagnostics, 2716, 2763
Distance measurement, 2759
EMERGENCY STOP switch, 2756
Error with hard stop, 2768
Error with soft stop, 2765
Fault reactions, 2716
Following error, 2711, 2762
Gear, 2716
Hardware configurations, 2708
Interaction of the instructions, 2742
Jogging, 2714
Linear axis, 2712
Monitoring ranges, 2712
Multi-instance FB, 2755
Overriding travel, 2716, 2749
Parameter error, 2764, 2770
Positioning, 2710
Reference point approach, 2715
Relative positioning, 2715
Residual distance, 2711
Rotary axis, 2713
Set reference point, 2715
Simulation, 2716
Standstill range, 2712
Start of a travel order, 2748
Stopping, 2716
Synchronization, 2714
Target, 2711
Target range, 2712
Tracking, 2716
Traversing range, 2712
User program configuration, 2742
Velocity measurement, 2757
Velocity of axis, 2761
Velocity override, 2716
Wiring test, 2757
Working range, 2712
Edge detection, 2522
Data in the channel DB, 2523
Data in the parameter DB, 2524
faulty, 2524
Requirement, 2522
Sequence, 2523
Effects
Reference point approach, 2415, 2507
Set actual value, 2424, 2517
Set reference point, 2426, 2521
Zero offset, 2519
Emergency stop deceleration, 1661, 1812
EMERGENCY STOP switch
Easy Motion Control, 2756
Enable
Diagnostic interrupt, 59, 1058
Hardware interrupt, 59, 78
Enable diagnostic interrupts, 1284
Enable input, 2412, 2430, 2504, 2527
Data in the channel DB, 2431, 2528
Data in the parameter DB, 2528
Do not evaluate, 2430, 2528
Evaluate, 2430, 2527

Enabling
Machine data, 453, 2398, 2490
Encoder
Count direction, 500, 576, 2453, 2551
Distance per encoder revolution, 496, 572, 2454, 2552
Frame length, 502, 578, 2458, 2556
Increments per encoder revolution, 496, 572, 2454, 2553
Initiator direction, 503, 579
Monitoring, 500, 576, 2458, 2557
Number of encoder revolutions, 501, 577, 2457, 2555
Parity, 2458, 2557
Selecting the signal type, 219, 269, 294
Signal type, 495, 571, 2452, 2550
Transmission rate, 502, 578, 2457, 2556
Encoder connection S7-1200 Motion Control, 1308
Encoder control word, 2704
Encoder data, 2431, 2528
Data in the channel DB, 2432, 2529
Process when reading, 2431, 2529
Requirement, 2431, 2529
Encoder machine data
Data in the parameter DB, 497, 573
Encoder mounting type, 1653, 1654, 1806
Encoder type, 2692, 2697
SSI, 2698
TTL, 2697
Encoder type not available, 2692, 2697
Encoder value acquisition
freewheeling, 2366, 2378
isochrone, 2351, 2378
Synchronous, 2366, 2378
Encoders
Connectable, 204
Incremental, 204
Pulse, 204
End, 2406, 2498
Positioning operation, 2401, 2493
End of a positioning operation
Checkback signals in the channel DB, 2408, 2500
Data in the parameter DB, 2408, 2500
End of rotary axis, 2447, 2545
ENSET_DN, 224, 299
ENSET_UP, 224, 299
Entering
Message frame type, 2701
Error
external, 1008, 1057, 1089
Internal, 1008, 1057, 1089
Error classes
Alarms, 529, 607, 2480, 2579
Cam data error, 528, 606
Data error, 524, 602, 2476, 2575
Diagnostic errors, 530, 607, 2481, 2580
Incremental dimension table error, 2480, 2579
Machine data error, 526, 604, 2478, 2577
Operating error, 524, 601, 2474, 2573
Operator errors, 2475, 2574
Error codes
Gx_XIST2, 2705
Error display, 1008, 1056, 1088
Error evaluation
User program, 521, 598, 2470, 2569
Error ID, 1965, 2132, 2346
Error LEDs, 1088
Error with hard stop
Easy Motion Control, 2768
Error with soft stop
Easy Motion Control, 2765
ErrorID
Basics, 1961, 1965
List of ErrorIDs, 2132, 2346
Errors in Motion Control instructions, 1961, 1965, 2132, 2346
ET 200, 1159, 1164
TM Timer DIDQ, 1159
Evaluation
Diagnostic interrupts, 2471, 2570
External encoder
Adding, 1782
Basics, 1720
Configuration, 1832
Deleting, 1784
Diagnostics, 1977, 1979
Functions, 1627
Moving, 1783
Tags, 2068
External errors, 1008, 1057, 1089

F

Fast cam parameter change
Canceling, 470, 546
Data check, 470, 546
Data used in the channel DB, 469, 545
Definition, 468, 544
Effects, 470, 546
Requirements, 468, 544
Steps, 468, 544
Fault reactions
Easy Motion Control, 2716

FCS, 2156
Feedback interface, 168
1Count24V for counting modes, 339
1Count24V for fast mode, 374
1Count24V for measuring modes, 356
1Count24V for position feedback, 367
1Count5V for counting modes, 400
1Count5V for fast mode, 435
1Count5V for measuring modes, 415
1Count5V for position feedback, 428
1SSI in fast mode, 2383
1STEP 5V, 2687
2PULSE, 1149
FM 350-1 for counter modes, 246
FM 350-1 for measuring modes, 251
FM 351, 2462
FM 352, 514
FM 451, 2562
FM 452, 591
Fictitious target, 2428, 2526
Location, 2429, 2526
Field of application, 2691
Fields of application, 2691
Filter frequency, 86, 141
Find reference mark, 2704
Flange coordinate system (FCS), 2156
Flange zones, 2258
FM 350-1, 208
FM 350-2, 260
Channel setting, 264
FM 351, 2393
FM 352, 457
FM 355, 983
FM 355-2, 1015
FM 450-1, 287
FM 451, 2484
FM 452, 533
FM 455, 1063
FNP, 2149
Following error monitoring, 1377, 1690, 1691, 1692, 1828
Frame definition, 2158
Frame error, 500, 576, 2459, 2557
Frequency measurement, 50, 191
Function
Change cam edges, 467, 543
Set process value, 460, 536
Set process value on-the-fly, 460, 536
Set reference point, 465, 541
Undo set process value, 460, 536
Zero offset, 463, 539
Functions
current control, 1245
Current measurement, 1243
direct control of a digital output, 1252
dither PWM output, 1248
high-speed output, 1242
isochronous mode, 1251
Overview, 1187
sequence counter, 1242

G

Gate
Hardware, 199
Internal, 199
Software, 199
Gate control, 21, 25, 50, 73, 75, 91, 146
in isochrone mode, 202
gate function
cancel, 202
interrupt, 202
Gate stop function, 200
Gear
Easy Motion Control, 2716
Gear ratio, 1710, 1711
Gearing, 1710, 1711
Group error LED, 1008, 1056
FM 350-1, 254
FM 351, 2467
FM 352, 518
SM 338, 2361
Gx_STW, 2704, 2705
Gx_XIST2, 2705

H

Hardware and software limit switches
Function, 1322
Hardware configuration for Motion Control S7-1200, 1298
Hardware gate, 21, 25, 73, 75, 91, 146, 199
edge-triggered opening and closing, 201
level-triggered opening and closing, 201
Hardware interrupt, 59, 78, 241, 277, 308
Enable, 138
enabling, 241, 277
lost, 136
hardware interrupts
Enable, 513, 589
Hardware limit switches, 1655, 1813
Hardware requirements, 1158

High count limit, 213, 268, 336, 397
High limit, 237, 353, 412
High_Speed_Counter, 80
    Basic parameters, 84
    Commissioning, 103
    Configuring, 82
    Diagnostics, 104
    Programming, 102
High-speed output, 1174
High-speed output function, 1242
Home position, 1672
Homing
    Homing modes, 1324
Homing mark, 1671
Hysteresis, 22, 23, 56, 58, 99, 117, 153, 162, 240, 271, 492, 568
    Position-based cams, 492, 568
    Time-based cams, 493, 569
Hysteresis range, 444

I

IM 174
    Configuring, 2691
Increment number 1-100, 2418, 2511
Increment number 254, 2419, 2512
Increment number 255, 2420, 2513
Incremental actual value, 1639, 1640
Incremental approach, 2418, 2510
    Absolute, 2418, 2510
    Cancel, 2423, 2516
    Data in the channel DB, 2421, 2423, 2514, 2516
    Data in the parameter DB, 2422, 2515
    Increment number 1-100:, 2418, 2511
    Increment number 254, 2419, 2512
    Increment number 255, 2420, 2513
    Incremental dimension number 252, 2511
    Relative, 2418, 2510
    Requirement, 2418, 2510
    Shutdown, 2422, 2515
Incremental approach with increment number 254
    Sequence, 2419, 2512
Incremental approach with increment number 255
    Sequence, 2420, 2513
Incremental approach with incremental dimension number 252
    Sequence, 2511
Incremental dimension number 1 to 100, 2460, 2558
Incremental dimension number 252, 2511, 2560
Incremental dimension number 254, 2461, 2560
Incremental dimension number 255, 2462, 2561
Incremental dimension tables, 2399, 2491
    Changing, 2400
    Data in the channel DB, 2400, 2492
    Reading, 2400
    Writing, 2400, 2491
Incremental encoder, 447, 2389
    Increment, 449, 2390
    Increments per encoder revolution, 2454, 2553
    Pulse, 449, 2391
    Reaction times, 450
    Response times, 2391
    Signal forms, 2390
    Signal shapes, 449
Increments, 2459, 2558
Initiator, 448, 451
Input delay
    Technology module, 1175, 1178, 1181
Input filter, 223, 298
Insert empty line, 1408
Insert separator line, 1408
Integration time, 187
Internal errors, 1008, 1057, 1089
Internal Gate, 199
Interpolator OB, 1318, 1319, 1696, 1698
IODD, (IO-Link Device)
IO-Link, 1159
    Setting ports, 1165
IO-Link Device, 1159, 1163, 1164, 1165
IPO model
    Understanding IPO, 1161
Isochronous communication, 1160
Isochronous mode, 67, 71, 79
    Operating mode DC motor, 1236
    Operating mode frequency output, 1229
    Operating mode On-/Off-delay, 1222
    Operating mode pulse output (individual pulse), 1192
    Operating mode pulse width modulation (PWM), 1200
    overview, 1251

J

Jerk limit, 1660, 1811, 1815
    Function, 1323
Jitter, 1162
Job
    Cancel set actual value, 2425, 2518
    Set actual value, 2423, 2516
    Set reference point, 2425, 2521
    Setting actual value on the fly, 2516
    Zero offset, 2518

Job sequence, 2149, 2281
Jog, 2408
  Cancel, 2410
  Data in the channel DB, 2409
  Monitoring functions, 2410
  Operating range limit, 2410
  Requirement, 2408
  Sequence, 2408
  Shutdown, 2409
Jogging, 2500
  Cancel, 2502
  Data in the channel DB, 2501
  Easy Motion Control, 2714
  Monitoring functions, 2502
  Operating range limit, 2502
  Requirement, 2500
  Sequence, 2500
  Shutdown, 2501
Joint position space, 2239

K

KCS, 2155
Kinematics, 2148, 2160
  Adding, 2262
  Circular motion, 2249
  Configuration, 2263
  Coordinate systems, 2154
  Copying, 2280
  Degrees of freedom, 2149
  Deleting, 2280
  Diagnostics, 2302, 2305, 2306
  Dynamic adaptation, 2252
  Dynamic defaults, 2251
  Dynamic limits, 2252
  Dynamic settings, 2253
  Frames, 2154
  Functions, 2149
  Interconnection rules, 2151
  Joint position space, 2239
  Kinematics control panel, 2290
  Linear motion, 2249
  Mechanical couplings, 2241
  Motion, 2248
  Simulation, 2152
  Singular positions, 2240
  Tags, 2307
  Term definition, 2148
  User transformation, 2242
  Zone monitoring, 2254
Kinematics axes, 2148
Kinematics control panel, 2290, 2295
Kinematics coordinate system, 2155
Kinematics motion, 2248
  Circular motion, 2286
  continue, 2283
  Dynamics, 2251
  Interrupting, 2283
  Linear motion, 2284
  Motion transition, 2284
  Motion transition dynamics, 2287
  Remaining distance, 2283
  Status, 2282
  stop, 2284
Kinematics transformation, 2237, 2238
Kinematics type
  Articulated arm 2D, 2191
  Articulated arm 2D with orientation, 2193
  Articulated arm 3D, 2196
  Articulated arm 3D with orientation, 2201
  Cylindrical robot 3D, 2220
  Cylindrical robot 3D with orientation, 2224
  Delta picker 2D, 2207
  Delta picker 2D with orientation, 2210
  Delta picker 3D, 2213
  Delta picker 3D with orientation, 2216
  Portal 2D, 2161
  Portal 2D with orientation, 2163
  Portal 3D, 2165
  Portal 3D with orientation, 2168
  Roller picker 2D, 2172
  Roller picker 2D with orientation, 2174
  Roller picker 3D (vertical), 2176
  Roller picker 3D with orientation (horizontal), 2182
  Roller picker 3D with orientation (vertical), 2179
  SCARA 3D with orientation, 2186
  Tripod 3D, 2229
  Tripod 3D with orientation, 2233
  User-defined, 2236
Kinematics types, 2160
Kinematics zero point, 2149
Kinematics zones, 2255, 2258
KZP, 2149

L

Latch, (Capture)
Latch/Retrigger, 215
Latching, 213
Lead time, 445
Leading value coupling, 1706
Leadscrew pitch, 1653, 1654, 1806

Length measurement, 2522, (Application examples Time-based IO)
Data in the channel DB, 2523
Edge distance, 2524
faulty, 2524
Requirement, 2522
Sequence, 2523
Lifebeat failures, 2700
Limit monitoring, 189
Limit switches, 1655, 1657, 1813
Limits for dynamics, 1659, 1815
Linear axis, 1632
Easy Motion Control, 2712
Linear motion, 2249
Load gear, 1653, 1654, 1806
Load value, 198, 199, 223, 298
directly, 198
in preparation, 198
Loading
Axis data block, 2752
Loop traverse, 2427, 2524
Data in the channel DB, 2428, 2525
Requirements, 2427, 2525
Sequence, 2428, 2525
Loss of
Synchronization, 2520
Loss of synchronization, 465, 541
Low limit, 237, 353, 412

M

Machine coordinate system, 2149
Machine data, 2397, 2489
Data in the channel DB, 2399, 2490
Drive, 2435, 2533
Enable input, 504, 580
Enabling, 453
Reading, 455
Writing, 453
writing, activating, changing, reading, 2398, 2489
Main count direction, 185, 218, 269, 330, 389
Master control, 1949, 1996
Max. frequency, 2695
MC_TorqueAdditive, 1668
MC_TorqueRange, 1668
MC-Interpolator OB, 1318, 1319, 1696, 1698
MCS, 2149
MC-Servo OB, 1318, 1319, 1696, 1698
MC-Transformation-OB, 2244
Measure
Frequency, 191
Rotary speed, 193
Measure times
between two edges, 227
Measuring
Period, 194
Measuring functions, 50
Measuring input
Basics, 1721
Configuration, 1845
Diagnostics, 1979
Functions, 1627
Tags, 2082
Measuring interval, 51, 54
Measuring range, 50, 51, 54
Measuring range violation (overrange), 1013, 1061, 1095
Measuring range violation (underrange), 1013, 1061, 1095
Measuring system
Selecting, 483, 559
Measuring time, 187
Measuring times, 227
Measuring transducer fault, 1013, 1061, 1095
Mechanical couplings, 2241
Message frame structure, 2701
Message frame type
Entering, 2701
Minimum edge distance, 2550
Minimum edge-to-edge distance, 494, 570
Minimum pulse width, 223, 298
Missing pulses incremental encoder, 501, 577
Missing pulses of incremental encoder, 2459, 2557
Modes, (PIP_Mode)
Selecting the model of program editing, 1161
Modulo, 1633, 1795
Monitoring
Standstill velocity, 2402, 2494
Target range, 2402, 2494
Monitoring 5 V encoder signals, 205
Monitoring functions, 2402, 2494
Jog, 2410
Jogging, 2502
Monitoring time, 2402, 2403, 2443, 2494, 2495, 2541
Motion Control CPU S7-1200
Guidelines, 1326
Motion Control S7-1200
Drive and encoder connections, 1308
PROFIdrive, 1308
Telegram, 1308
Motion transition, 2284
Multi-instance FB
Easy Motion Control, 2755

N

NAMUR encoder, 264

O

OB 82
Diagnostic interrupt, 1009, 1058, 1090
OB servo, 1319
Object coordinate system, 2157
OCS, 2157
OIP model
Understanding OIP, 1162
On-the-fly measurement, 2704
Operating mode, 132, 1273, 1283, 1289
Count continuously, 182
Count once, 183
Count periodically, 184
Digital input, 1180
Digital output, 1174, 1177, 1183
Dosing, 197
Frequency measurement, 191
High_Speed_Counter, 95
Incremental approach, 2418, 2510
Jog, 2408
Jogging, 2500
Period measurement, 194
Reference point approach, 2411, 2503
Rotational speed measurement, 193
Operating mode DC motor, 1233
Control and feedback signals of the PWM operating mode, 1239
Input and output signals of the PWM operating mode, 1241
isochronous mode, 1236
Parameters of the DC motor operating mode, 1238
Setting and changing the period duration, 1236
Setting and changing the pulse duty cycle and the direction of rotation, 1235
Truth table, 1235
Operating mode frequency output, 1226
Control and feedback signals, 1230
Input and output signals, 1233
isochronous mode, 1229
On-delay, 1229
Output value (frequency), 1228
Parameter, 1229
Truth table, 1228
Operating mode On-/Off-delay, 1216
Control and feedback signals, 1223
Input and output signals, 1226
Isochronous mode, 1222
Minimum pulse duration, 1219
Off-delay, 1222
On-delay, 1222
Parameter, 1223
Retriggering of the off-delay, 1221
Retriggering the on-delay, 1221
Truth table, 1219
Operating mode pulse output (individual pulse), 1189
Control and feedback signals, 1193
Input and output signals, 1196
On-delay, 1192
Parameter, 1193
Pulse duration, 1191
Sequence counter, 1192
Truth table, 1191
Operating mode pulse train, 1207
Control and feedback signals, 1213
Current measurement, 1212
Duty cycle, 1211
Input and output signals, 1216
Number of pulses, 1210
On-delay, 1211
Period, 1210
Truth table, 1209
Operating mode pulse width modulation (PWM), 1196
Control and feedback signals, 1204
Current control, 1203
Current measurement, 1203
Duty cycle, 1200
Input and output signals, 1207
Isochronous mode, 1200
Minimum pulse duration, 1198
On-delay, 1203
Parameter, 1203
Period, 1200
Setting minimum pulse duration, 1202
Truth table, 1198
Operating modes
DC motor, 1233
Frequency output, 1226
On-/Off-delay, 1216
Overview, 1186
Pulse output (individual pulse), 1189
Pulse train, 1207
Pulse width modulation (PWM), 1196
Operating range, 2387

Operating range limit
Jog, 2410
Jogging, 2502
Operator error, 260, 316
Optimization, 1423
Optimize position controller, 1423, 1955, 1958, 2000
Orientation movement, 2249
Output cam
Basics, 1733
Configuration, 1847
Diagnostics, 1981
Functions, 1627
Tags, 2089
Overload range, 2696
Overriding travel
Easy Motion Control, 2716, 2749

P

Parameter assignment
Compact CPU, 19
Technology module, 19, 129
Parameter error
Easy Motion Control, 2764, 2770
Passive homing, 1671, 1681, 1683, 1685, 1824
PathChoice, 2250
Period
Resolution, 218, 351, 410
Period measurement, 50, 194
Period resolution, 218, 351, 410
PID parameters, 1246
PID_Temp
Cascading, 763
Multi-zone applications, 770
Pin assignment
Front connector of the FM 350-1, 208
PIP_Mode
Setting PIP_Mode, 1167
Port Configuration Tool, 1159, 1163, 1165
Portal
2D, 2161
2D with orientation, 2163
3D, 2165
3D with orientation, 2168
Ports
Configuring ports, 1165
Position control, 1379
Position data, 2431, 2528
Data in the channel DB, 2431, 2528
Process when reading, 2431, 2528
Position input
Comparison value, 114, 159
With Motion Control, 60
with SSI absolute encoder, 22
Position limits, 1655, 1657, 1658, 1813
Position monitoring, 1377
Position-based cams, 439, 440
Positioning axis
Adding, 1782
Basics, 1704
Configuration, 1795
Deleting, 1784
Diagnostics, 1970, 1974, 1975, 2006, 2010
Functions, 1626
Moving, 1783
Tags, 1488, 1584, 2027
Positioning axis technology object
Active homing, 1373
Actor tags, 1490, 1595
Add new object, 1341
Axis name configuration, 1343, 1539
Basic parameters, 1343, 1539
Changing the configuration parameters for dynamics in the user program, 1368
Changing the configuration parameters for homing in the user program, 1376
Commissioning overview, 1340
Configuration overview, 1340
Configuration window icons, 1342
Configuring acceleration, 1364
Configuring active homing, 1369, 1544
Configuring approach/homing direction, 1371, 1545
Configuring deceleration, 1364
Configuring distance per motor revolution, 1355, 1542
Configuring end of homing switch, 1371, 1373
Configuring end of reference point switch, 1544, 1546
Configuring home position offset, 1371, 1546
Configuring homing switch input, 1370, 1373
Configuring invert direction signal, 1355, 1543
Configuring jerk limiter, 1364
Configuring maximum velocity / start/stop velocity, 1363
Configuring passive homing, 1372, 1543
Configuring permit auto reverse at the hardware limit switch, 1545
Configuring pulses per motor revolution, 1355, 1542
Configuring ramp-down time, 1364
Configuring ramp-up time, 1364
Configuring reference point position, 1371, 1373, 1544, 1546

Configuring reference point switch input, 1544, 1545
Configuring smoothing time, 1365
Configuring the homing speed, 1371, 1546
Configuring the velocity limiting unit configuration, 1363
Diagnostics overview, 1340
Drive enable configuration, 1542
Drive ready configuration, 1542
Drive signal configuration, 1347, 1542
DynamicDefaults tag, 1504, 1604
DynamicLimits tag, 1504, 1604
Emergency stop deceleration configuration, 1366
ErrorBits tag, 1520, 1618
Extended parameters, 1542
FollowingError tag, 1510, 1609
General dynamics configuration, 1362
Hardware and software components, 1336
Hardware interface configuration, 1345, 1539
Homing tag, 1509, 1607
Mechanics configuration, 1355, 1356, 1542
Mechanics tag, 1502, 1602
Modulo tag, 1503, 1603
Passive homing, 1375
Position tag, 1489, 1594
PositionControl tag, 1509, 1608
PositionLimitsHW tag, 1506, 1606
PositionLimitsSW tag, 1505, 1605
PositionMonitoring tag, 1511, 1609
PTO and HSC configuration, 1346, 1540
Response when jerk limiter is activated, 1367
Sensor tag, 1495, 1598
Simulation tag, 1489
StandstillSignal tag, 1511, 1610
StatusBits tag, 1515, 1613
StatusDrive tag, 1513, 1611
StatusPositioning tag, 1512, 1611
StatusSensor tag, 1514, 1612
Tag ActualPosition, 1489
Tag ActualVelocity, 1489
Tools, 1339
Unit of measurement configuration, 1344
Units tag, 1502, 1602
Updating the tags, 1525, 1619
User unit configuration, 1541
Velocity tag, 1489
Positioning monitoring, 1690, 1692, 1828
Positioning operation
End, 2401, 2493
Predictability, 1157
Process image partition
PIP, 1166
PIP1, 1164
Process image partition "OB Servo PIP", 1698
PROFIdrive, 1308, 1635, 1641
PROFINET, 1159, 1163, 1164
Using PROFINET IRT, 1160
Programming, 1166
Programming example, 1169
Pulse duration, 239, 303
Default setting, 239, 303
Range of values, 239, 303
Pulse interface
Principle, 1303
PULSE REFEED, 2696
PWM
Technology module, 1172

Q

Quadruple evaluation, 207

R

Ramp-down time, 1811, 1815
Ramp-up time, 1811, 1815
Reaction to CPU STOP, 129, 1274, 1291
Technology module, 1173
Reaction to CPU/Master STOP
1STEP 5V, 2676
Reading
Cam data, 456
Incremental dimension tables, 2400, 2492
Machine data, 455, 2398, 2490
Reduction ratio, 1696
Reference cam, 1672
Reference point approach, 2411, 2503
Cancel, 2415, 2507
Data in the channel DB, 2414, 2506
Data in the parameter DB, 2415, 2507
Effects, 2415, 2507
Requirement, 2412, 2504
Start speed, 2452, 2550
Synchronization, 2411, 2503
Types, 2415, 2507
Reference point coordinate, 488, 564, 2448, 2546
Reference point switch, 2412, 2504
Reinitialization of technology objects, 1444, 1945
Relative positioning
Easy Motion Control, 2715
Reproducibility, 1155

Requirement
Edge detection, 2522
Encoder data, 2431, 2529
Incremental approach, 2418, 2510
Jog, 2408
Jogging, 2500
Length measurement, 2522
Loop traverse, 2427, 2525
Reference point approach, 2412, 2504
Set actual value, 2423, 2517
Set reference point, 2426, 2521
Setting actual value on the fly, 2517
Zero offset, 2519
Resolution, 497, 573, 2455, 2553
Range of values, 2456, 2554
Value range, 498, 574
Response time, 1158
Defined response times, (Application examples Time-based IO)
Minimum response time, 1158
Restart of technology objects, 1444, 1945
RET_VALU messages, 1014, 1015, 1062, 1063, 1096, 1097
Retrigger reference point, 487, 563
Reversing cam, 1672, 1686, 1822
Reversing switch, 2412, 2504
Roller picker
2D, 2172
2D with orientation, 2174
3D (vertical), 2176
3D with orientation (horizontal), 2182
3D with orientation (vertical), 2179
Rotary axis, 1632
Easy Motion Control, 2713
Rotation conventions, 2158
Rotational speed measurement, 193

S

S7-1500 Motion Control, 1628
Actual value, 1639, 1640
Axis type, 1631
Closed-loop control, 1693, 1694, 1695, 1830, 1831
Commissioning, 1946, 1947, 1949, 1953, 1955, 1958, 1996, 2000
Configuration, 1784, 1785, 1795, 1853
Configuration limits, 1630
Diagnostics, 1961, 1965
Downloading to CPU, 1946
Drive and encoder connection, 1625, 1634, 1636, 1641, 1652, 1774, 1775, 1777, 1780, 1796, 1798, 1800
Dynamic defaults, 1661
Dynamic settings, 1659, 1660, 1669, 1811, 1815
Guidelines for operation, 1630
Homing, 1670, 1671, 1672, 1673, 1681, 1686, 1687, 1688, 1819, 1820, 1824
How it works, 1622
Introduction, 1622
Mechanics, 1653, 1654, 1806
Modulo, 1633, 1795
Motion Control instruction, 1625, 1626, 1627, 1628, 1924, 1927
Position limits, 1655, 1657, 1658, 1813
Position monitoring, 1690, 1691, 1692, 1828, 1829
Process response, 1696, 1698, 1701
PROFIdrive, 1635, 1641
Programming, 1919, 1920, 1924, 1932, 1945
Synchronous operation, 1706, 1710, 1711, 1713, 1719, 1805
Technology alarms, 1961, 2107, 2338
Technology data block, 1624, 1920, 1922, 1923
Technology object, 1624, 1626, 1627, 1628, 1703, 1704, 1705, 1720, 1721, 1733, 1747, 1760, 1784, 1795, 1853, 1945
Telegram, 1636, 1641, 1652
Unit of measure, 1632, 2152
Versions, 1765, 1772, 2261
S7-1500 Motion Control actual value, 1639, 1640
S7-1500 Motion Control commissioning, 1946, 1947, 1949, 1953, 1955, 1958, 1996, 2000
S7-1500 Motion Control drive connection, 1625, 1634, 1636, 1641, 1652, 1774, 1775, 1777, 1780, 1796
S7-1500 Motion Control encoder connection, 1625, 1634, 1636, 1641, 1652, 1774, 1775, 1777, 1780, 1798, 1800
S7-1500 Motion Control homing
Absolute encoder adjustment, 1671
Absolute value adjustment, 1687
Active, 1671, 1673, 1820
Basics, 1670
Configuration, 1819
direct, 1686
Home position, 1672
Homing mark, 1671
Homing mode, 1671, 1672
On the fly, 1671, 1681, 1824
Passive, 1671, 1681, 1824
Reference cam, 1672

Reversing cam, 1672, 1686, 1822
Zero mark, 1672
S7-1500 Motion Control instruction, 1625
Ending a Motion Control job, 1944
Errors in Motion Control instructions, 1961, 1965, 2132
Inserting, 1927
Overview, 1626, 1627, 1628
Parameters, 1924
Starting Motion Control job, 1932
Tracking Motion Control job, 1934
S7-1500 Motion Control mechanics, 1653, 1654, 1806
S7-1500 Motion Control operating mode, 1701
S7-1500 Motion Control optimization, 1955, 1958, 2000
S7-1500 Motion Control telegram, 1636, 1641, 1652
S7-1500 Motion Control
Configuration, 1782, 1783
Technology object, 1782, 1783
S7-1500T, 1628
S7-1500T Motion Control
Commissioning, 2295
S7-1500T Motion Control instruction
Errors in Motion Control instructions, 2346
S7-PCT, (Port Configuration Tool)
Saving
Axis data block, 2752
SCARA
3D with orientation, 2186
Selecting gate control, 183, 184, 191, 193, 195, 197
Selecting the Gate Contrtol, 182
Sensor, 1625
Sensor type, 86, 141
Sequence
Checkback signals for the diagnostics, 2433, 2530
Checkback signals for the positioning, 2432, 2529
Edge detection, 2523
Incremental approach with increment number 1-100, 2418, 2511
Incremental approach with increment number 254, 2419, 2512
Incremental approach with increment number 255, 2420, 2513
Incremental approach with incremental dimension number 252, 2511
Jog, 2408
Jogging, 2500
Length measurement, 2523
Loop traverse, 2428, 2525
Read encoder data, 2431, 2529
Read position data, 2431, 2528
Set actual value, 2424, 2517
Set reference point, 2426, 2521
Setting actual value on the fly, 2517
Zero offset, 2519
Sequence counter, 1242
Servo drive type, 2693
Servo motor, 1298
Servo OB, 1318, 1319, 1696, 1698
SET, 223
Set actual value, 2423, 2516
Data in the channel DB, 2424, 2517
Effects, 2424, 2517
Requirement, 2423, 2517
Sequence, 2424, 2517
Set process value, 460, 536
Data in the channel DB, 461, 537
Effect, 462, 537
Requirement, 461, 537
Sequence, 461, 537
Set process value on-the-fly, 460, 536
Data in the channel DB, 461, 537
Requirement, 461, 537
Sequence, 461, 537
Set reference point, 465, 541, 2425, 2521
Data in the channel DB, 2426, 2521
Data used in the channel DB, 466, 542
Definition, 465, 541
Effects, 466, 542, 2426, 2521
Requirement, 2426, 2521
Requirements, 465, 541
Sequence, 2426, 2521
Special features of the absolute encoder, 467, 543
Steps, 466, 542
Synchronization, 2425, 2521
Setting actual value on the fly, 2516
Data in the channel DB, 2517
Requirement, 2517
Sequence, 2517
Setting the counter, 223, 298
With an external signal, 224, 299
With digital input 1I2, 299
With digital input I2, 224
With the zero mark, 226, 301
Shutdown
Incremental approach, 2422, 2515
Jog, 2409
Jogging, 2501
Shutdown delay, 2700
Signal evaluation, 86, 141, 206

Signal N, 86, 141
Signal type, 85, 140, 155, 1283
Signal zones, 2255, 2257
Sign-of-life error, 1287
SIMATIC functions, 1160
Simulation
    Easy Motion Control, 2716
Simulation speed, 493, 569
SINAMICS V90 PN, 1774
Single evaluation, 206
Singular positions, 2240
Sinking output, 86, 141
SM 338
    Addressing, 2359
    Diagnostic interrupt, 2356
    Diagnostic messages, 2361
    Isochronous mode, 2355
    Transmission rate, 2357
Software controller
    Configuring, 631
Software gate, 21, 25, 73, 75, 199
    Opening and closing, 200
Software limit switch, 485, 561, 2445, 2543
Software limit switches, 1655, 1657, 1813
Sourcing output, 86, 141
Special tracks, 442
Speed axis
    Adding, 1782
    Basics, 1703
    Configuration, 1786
    Deleting, 1784
    Diagnostics, 1966, 1969, 2003, 2005
    Functions, 1626
    Moving, 1783
    Tags, 2011
SSI absolute encoder, 109, 155
SSI_Absolute_Encoder, 105
    Basic parameters, 108
    Commissioning, 123
    Configuring, 108
    Diagnostics, 124
    Programming, 123
Stand. frequency, 2695
Standstill range, 2387, 2443, 2541
Standstill signal, 1378
Standstill velocity, 2444, 2542
    Monitoring, 2402, 2494
Start of a travel order
    Easy Motion Control, 2748
Start speed for reference point approach, 2452, 2550
Start value, 21, 73, 90, 145, 176
Startdrive, 1774
STEP 7, 1159
Stepper drive, 2693
Stepper drive type, 2693
Stepper drive with Stepper encoder type PULSE REFEED operation, 2696
Stepper motor, 1298
Stopping
    Easy Motion Control, 2716
Substitute value, 230, 276
SW_GATE, 25, 75
Switch points, 2387
Switching characteristics
    of time-based cams, 460, 535
Switching ranges, 2387
Switch-off difference, 2387, 2440, 2538
Switch-off difference minus, 2441, 2539
Switch-off difference plus, 2441, 2539
Switch-off point, 2387
Switch-over difference, 2387
Switch-over point, 2387
Symbol
    For value comparison, 638
Symmetrical encoders, 204
SYNC_PI, 1161
SYNC_PO, 1161
Synchronization, 21, 32, 91, 146
    at signal N, 37, 91, 146
    by digital input, 35
    Easy Motion Control, 2714
    loss of, 465, 541
    Reference point approach, 2411, 2503
    Set reference point, 2425, 2521
Synchronous axis
    Adding, 1782
    Basics, 1705
    Deleting, 1784
    Diagnostics, 1970, 1974, 1975, 2006, 2010
    Functions, 1626
    Moving, 1783
    Tags, 2027
Synchronous Cycle, 1167
    Understanding TIO instructions, 1160
Synchronous operation, 1706, 1710, 1711, 1713, 1805
System environment, 1158
System of units
    Selecting, 2433, 2531

T

Tags of the cam technology object, 2089, 2101

Tags of the cam track technology object, 2095
Tags of the measuring input technology object, 2082
Tags of the positioning axis technology object, 1488, 1584, 2027
Tags of the speed axis technology object, 2011
Tags of the synchronous axis technology object, 2027
Tags of the technology object external encoder, 2068
TAPP, (Application cycle)
Target, 2387
Target approach, 2402, 2494
Target range, 2387, 2441, 2539
    Monitoring, 2402, 2494
TCP, 2149, 2156
T-CPU, 1628
TCS, 2156
Technology alarms
    Basics, 1961
    List of the technology alarms, 2107, 2338
Technology data block
    Analyzing, 1920
    Basics, 1624, 1920
    Change restart-relevant data, 1923
    Evaluating StatusWord, ErrorWord and WarningWord, 1922
    Tags of the cam technology object, 2089, 2101
    Tags of the cam track technology object, 2095
    Tags of the kinematics technology object, 2307
    Tags of the measuring input technology object, 2082
    Tags of the positioning axis technology object, 2027
    Tags of the speed axis technology object, 2011
    Tags of the synchronous axis technology object, 2027
    Tags of the technology object external encoder, 2068
Technology module
    Application scenarios, 15
    Applications, 68
    Basic parameters, 129
    Channel configuration, 1172
    Control interface, 166
    Feedback interface, 168
    Hardware configuration, 127, 1171, 1271, 1282, 1288
    Input delay, 1175, 1178, 1181
    Online & Diagnostics, 165, 178, 1184, 1281, 1288
    Parameter assignment, 129
    Performance features, 11, 13, 1185
    Project tree, 127
    PWM, 1172
    Reaction to CPU STOP, 1173
Technology object
    Cam, 1628, 1760, 1762, 1853, 1889
    Cam track, 1627, 1747, 1850, 1983, 1984
    Data types, 1445, 1929
    External encoder, 1627, 1720, 1782, 1783, 1784, 1832, 1977, 1979
    High_Speed_Counter, 80
    Kinematics, 2149, 2150, 2262
    Measuring input, 1627, 1721, 1845, 1979
    Output cam, 1627, 1733, 1847, 1981
    Positioning axis, 1626, 1704, 1782, 1783, 1784, 1795, 1970, 1974, 1975, 2006, 2010
    Speed axis, 1626, 1703, 1782, 1783, 1784, 1786, 1966, 1969, 2003, 2005
    SSI_Absolute_Encoder, 105
    Synchronous axis, 1626, 1705, 1782, 1783, 1784, 1795, 1970, 1974, 1975, 2006, 2010
Technology object axis
    ErrorBits tag, 1593
    MotionStatus tag, 1588
    StatusBits tag, 1589
    Tag config., 1584
Technology object command table: Shortcut menu commands, 1408
Technology objects, 1624
    CONT_C, 777
    CONT_S, 780
    PID_3Step, 697
    PID_Compact, 660
    PID_CP, 806
    PID_ES, 853
    PID_Temp, 734
    TCONT_CP, 783
    TCONT_S, 803
Telegram S7-1200 Motion Control, 1308
Terminal assignment
    Encoder interface of FM 351, 2393
    Encoder interface of FM 352, 457
    Encoder interface of FM 451, 2484
    Front connector of FM 351, 2394
    Front connector of FM 352, 458
    Front connector of FM 451, 2485
    Front connector of FM 452, 533
    Front connector of the FM 350-2, 260
    Front connector of the FM 450-1, 287
    Front connector of the SM 338, 2354
TIA environment, 2691

Time behavior
Standard technology, 1156
Time-based IO, 1156
Time-based cam
Activating, 460, 536
Influence of functions, 460, 535
Switching characteristics, 460, 535
Time-based cams, 439, 440
Time-based IO
Setting Time-based IO, 1164
Understanding SIMATIC functions, 1160
Using the default setting, 1164
Using Time-based IO, 1169
TIO instructions, 1159
Listing TIO instructions, 1166
TM Timer DIDQ, 1159
TIO module, 1159, 1164
TIO_SYNC
Synchronized TIO modules, 1160
TIO_Time, 1156, 1166
Understanding TIO_Time, 1160
TM_CTRL_DQ signal, 1252
TO_PositioningAxis, 1341
Tolerable lifebeat failures, 2700
Tool center point, 2149, 2156
Tool coordinate system, 2156
Tool zones, 2258
Torque limits, 1668
Torque setpoint, 1668
Tracking
Easy Motion Control, 2716
Tracks
Brake cam track, 504, 579
Counter cam track, 504, 579
enabling, 442
External enable, 442
High count value for counter cam track, 504, 580
Setting a track signal, 442
Special tracks, 504, 579
Track output control, 503, 579
Track result, 441
Transformation area, 2238
Transformation-OB, 2244
Transmission rate, 2367, 2379
Traversing range, 2238
Traversing range limitation, 1655, 1657, 1658, 1813
Triggering
Diagnostic interrupt, 1057
Tripod
3D, 2229
3D with orientation, 2233
Truth table
Operating mode DC motor, 1235
Operating mode frequency output, 1228
Operating mode On-/Off-delay, 1219
Operating mode pulse output (individual pulse), 1191
Operating mode pulse train, 1209
Operating mode pulse width modulation (PWM), 1198
Type of reference point approach, 2447, 2545
Types
Reference point approach, 2415, 2507

U

Undo set actual value
Data in the channel DB, 2425
Undo set process value, 460, 536
AVALREM_EN, 462, 538
Unit of measure, 1632, 2152
Update time, 50
User program, (Synchronous Cycle)
Error evaluation, 521, 598, 2470, 2569
User transformation, 2242
Example, 2246
User-defined kinematics systems, 2236

V

Values
Comparing, 638
Velocity measurement, 50
Easy Motion Control, 2757
Velocity override
Easy Motion Control, 2716
Velocity precontrol, 1693
Velocity profile, 1660

W

WCS, 2155
Wire break, 131, 500, 576, 1061, 2458, 2557
Wire breakage, 1013
Wire-break, 1095
Wiring test
Easy Motion Control, 2757
Work zones, 2255, 2257
Workspace zones, 2255, 2257
World coordinate system, 2155
Write
Incremental dimension tables, 2491

Writing
Cam data, 455
Incremental dimension tables, 2400
Machine data, 453, 2398, 2489

Z

Zero mark, 1672
Zero offset, 2518
Canceling, 465, 541, 2520
Data in the channel DB, 2519
Data used in the channel DB, 463, 539
Definition, 463, 539
Effects, 2519
on a linear axis, 464, 539
on a rotary axis, 464, 540
Requirement, 2519
Sequence, 463, 539, 2519
ZOFF_EN, 463, 539
Zero pass, 217, 294
Zone geometry, 2259
Zone monitoring, 2254
Zones
Flange zones, 2258
Kinematics zones, 2258
Signal zones, 2257
Subentry, 2254, 2257
Tool zones, 2258
Work zones, 2257
Workspace zones, 2257
Zone geometry, 2259

SIEMENS

Using online and diagnostics functions

Programming and Operating Manual

Displaying accessible devices 1

Changing the device configuration online 2

Connecting devices online 3

Creating a backup of an S7 CPU 4

Configuring the PG/PC interface 5

Using the trace and logic analyzer function 6

Establishing a remote connection with TeleService 7

Simulating devices with S7-PLCSIM 8

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Displaying accessible devices....9

2 Changing the device configuration online....11

3 Connecting devices online....13

3.1 General information about online mode....13

3.2 View in online mode....15

3.3 Default setting online connection data....17

3.4 Establishing or changing an online connection....18

3.5 Canceling an online connection....21

3.6 Connecting online with several devices....22

3.7 Disconnecting online connections of multiple devices....23

4 Creating a backup of an S7 CPU....25

4.1 Backup options for S7 CPUs....25

4.2 Backing up S7-300 and S7-400 CPUs (S7-300, S7-400, PC)....27
4.2.1 Creating a backup of a device (S7-300, S7-400, PC)....27
4.2.2 Restoring the software and hardware configuration of a device (S7-300, S7-400, PC)....28
4.2.3 Backing up a device configuration (S7-300, S7-400, PC)....28

4.3 Backing up S7-1200 and S7-1500 CPUs (S7-1200, S7-1500, S7-1200, S7-1500)....30
4.3.1 Creating a backup of a device (S7-1200, S7-1500, S7-1200, S7-1500)....30
4.3.2 Backing up a device configuration (S7-1200, S7-1500, S7-1200, S7-1500)....31
4.3.3 Restoring the configuration of a device (S7-1200, S7-1500, S7-1200, S7-1500)....32

5 Configuring the PG/PC interface....35

5.1 Online access....35

5.2 Basics of assigning parameters for the PG/PC interface....37

5.3 Showing or hiding interfaces....38

5.4 Displaying and modifying interface properties....39

5.5 Adding interfaces....40

5.6 Setting parameters for the Ethernet interface....41
5.6.1 Setting parameters for the Industrial Ethernet interface....41
5.6.2 Displaying operating system parameters....42
5.6.3 Connecting the PG/PC interface to a subnet....42
5.6.4 Setting parameters for the Ethernet interface....43
5.6.5 Utilization of temporary IP addresses....44
5.6.6 Managing temporary IP addresses....44
5.6.7 Resetting the TCP/IP configuration....45

5.7 Setting parameters for the MPI and PROFIBUS interfaces....46
5.7.1 Setting parameters for the MPI and PROFIBUS interfaces....46

5.7.2 Setting MPI or PROFIBUS interface parameters automatically....47
5.7.3 Setting parameters for the MPI interface....48
5.7.4 Setting parameters for the PROFIBUS interface....49
5.7.5 Overview of the bus parameters for PROFIBUS....52
5.7.6 Resetting the MPI or PROFIBUS configuration....53

6 Using the trace and logic analyzer function....55

6.1 Preface....55
6.1.1 Security information....56

6.2 Description....58
6.2.1 Supported hardware....58
6.2.2 Recording of measured values with the trace function....58
6.2.3 Trace configuration, recording, installed trace and measurement....60
6.2.4 Data storage....61

6.3 Software user interface....63
6.3.1 Project navigator....64
6.3.1.1 User interface - “Traces” project tree folder....64
6.3.1.2 User interface - “Measurements” project tree folder....66
6.3.1.3 User interface - “Installed measurements (memory card)” project tree folder....67
6.3.1.4 User interface - “Overlay measurements” project tree folder....68
6.3.2 Working area....69
6.3.2.1 User interface - trace toolbar....69
6.3.2.2 User interface - Configuration tab....70
6.3.2.3 User interface - Diagram tab....71
6.3.2.4 User interface - Signal selection tab (overlay measurements)....83
6.3.3 Trace task card....84
6.3.3.1 User interface - Measurement cursor pane....84
6.3.3.2 User interface - Snapshots pane....86
6.3.4 Inspector window....87
6.3.4.1 Interface - Inspector window....87

6.4 Operation....89
6.4.1 Quick start....89
6.4.2 Using the trace function - overview....93
6.4.3 Project tree....94
6.4.3.1 Creating a trace....94
6.4.3.2 Displaying a configuration....94
6.4.3.3 Displaying a diagram....95
6.4.3.4 Apply overlay measurement....95
6.4.3.5 Configuring objects in groups....95
6.4.4 Working area - general....96
6.4.4.1 Transferring the trace configuration to the device....96
6.4.4.2 Activating/deactivating an installed trace....97
6.4.4.3 Displaying the recording....97
6.4.4.4 Analyze an ongoing recording....98
6.4.4.5 Saving measurements in the project....99
6.4.4.6 Exporting and importing measurements....99
6.4.4.7 Transferring the trace configuration from the device to the project....100
6.4.4.8 Deleting installed traces....100
6.4.5 Working area - Configuration tab....101
6.4.5.1 Configuring the trace....101
6.4.6 Working area - Diagram tab....102

6.4.6.1 Use of the curve diagram....102
6.4.6.2 Use of the signal table....103
6.4.6.3 Using the signal group in the signal table....104
6.4.6.4 Observation of fast signals....106
6.4.6.5 Compare records (overlay measurements)....106
6.4.6.6 Align measurements precisely (overlay measurements)....108
6.4.6.7 Printing a recording....108

6.5 Devices....109
6.5.1 S7-1200/1500 CPUs....109
6.5.1.1 Recordable variables....109
6.5.1.2 Lifetime of the installed trace configuration and recorded values....110
6.5.1.3 Recording levels....110
6.5.1.4 Quantity structure....111
6.5.1.5 CPU load through trace recording....111
6.5.1.6 Software user interface of the configuration....111
6.5.1.7 Configuration....121

7 Establishing a remote connection with TeleService....125

7.1 Basics of working with TeleService....125
7.1.1 Introduction to TeleService....125
7.1.2 TeleService functionality....126
7.1.3 Telephone book at TeleService....126

7.2 Working with the phone book....128
7.2.1 Basics on working with the phone book....128
7.2.2 Structure of the phone book....128
7.2.3 Symbols in the phone book....129
7.2.4 Manage phone book....130
7.2.4.1 Open phone book....130
7.2.4.2 Inserting rows in the phone book....130
7.2.4.3 Showing and hiding columns in the phone book....131
7.2.4.4 Saving phone book....131
7.2.4.5 Importing phone book data....131
7.2.4.6 Exporting phone book data....133
7.2.4.7 Printing the phone book....134
7.2.4.8 Defining dialing rules....134

7.3 Remote connections as dial-up connections....136
7.3.1 Basics for establishing a dial-up connection....136
7.3.2 Telephone networks and modems....137
7.3.2.1 Supported telephone networks and modems....137
7.3.2.2 Installing the local modem....138
7.3.2.3 Connecting and configuring the remote modem....139
7.3.3 Access protection for dial-up connections....140
7.3.3.1 Access protection information....140
7.3.3.2 TeleService callback options....141
7.3.3.3 Levels of protection....141
7.3.3.4 Setting up access protection and callback number for the TS adapter....142
7.3.3.5 Complete a callback in TeleService....143
7.3.4 TS adapter MPI....145
7.3.4.1 Short description of the TS adapter MPI....145
7.3.4.2 How the TS adapter MPI works....145
7.3.4.3 Operating a TS adapter MPI in direct connection mode....146

7.3.4.4 Operating a TS adapter MPI in modem connection mode....147
7.3.4.5 TS adapter MPI configuration options....148
7.3.4.6 Configuring TS adapter MPI....149
7.3.4.7 Restoring default parameter assignment for TS adapter MPI....150
7.3.4.8 Exporting adapter parameters....150
7.3.4.9 Importing adapter parameters....151
7.3.5 TS adapter IE....152
7.3.5.1 Short description of the TS adapter IE....152
7.3.5.2 How the TS adapter IE works....153
7.3.5.3 Connection Types....153
7.3.5.4 TS Adapter IE parameter assignment options....155
7.3.5.5 Parameter assignment for TS Adapter IE....156
7.3.6 Establishing a dial-up connection to a remote system....158
7.3.6.1 Establishing a dial-up connection....158
7.3.6.2 Terminating a dial-up connection....159

7.4 Remote VPN connections....160
7.4.1 Basics for establishing a VPN connection....160
7.4.2 Basics of CA certificates....161
7.4.3 Installing CA certificates for VPN connections....163
7.4.4 Deleting CA certificates for VPN connections....165
7.4.5 Establishing a VPN connection to a remote system....166
7.4.5.1 Establishing VPN connections....166
7.4.5.2 Terminating VPN connections....167
7.4.6 TS Adapter IE Advanced....168
7.4.6.1 Short description of the TS Adapter IE Advanced....168
7.4.6.2 Connection types....168
7.4.6.3 TS Adapter IE Advanced parameter assignment options....170
7.4.6.4 Parameter assignment for TS Adapter IE Advanced....170

7.5 CPU controlled TeleService remote connections 173
7.5.1 Overview of CPU controlled remote connections....173
7.5.2 Establishing a connection from and to remote systens (PG-AS-remote coupling)....174
7.5.2.1 Remote plant access to a programming device/personal computer....174
7.5.2.2 Requirements for establishing a connection....174
7.5.3 Data exchange between remote systems (AS-AS-remote coupling)....175
7.5.3.1 AS-AS remote link basics....175
7.5.3.2 Hardware and software requirements for AS-AS remote link....176
7.5.4 Send SMS from a system....177
7.5.4.1 Requirements for sending an SMS....177
7.5.5 Send an email from a system....178
7.5.5.1 Requirements for sending e-mails....178

7.6 Notes on troubleshooting....181
7.6.1 General information on troubleshooting for modem problems....181
7.6.2 Recording a log file for the modem....181
7.6.3 Dial-up connection to the TS Adapter is not established....182
7.6.4 Dial-up connection from the TS Adapter is not established....184
7.6.5 Modem connection is interrupted....185
7.6.6 Checklist for troubleshooting the modem....185
7.6.7 Modem alarms....186
7.6.8 Possible error messages with VPN connections....187

8 Simulating devices with S7-PLCSIM....189

8.1 Simulation of devices....189

8.2 Starting the simulation....190

Index....191

Displaying accessible devices 1

Accessible devices are all devices connected to an interface of the programming device / PC that are turned on. Devices that allow only restricted configuration using the currently installed products or that cannot be configured at all can also be displayed.

To speed up the display in large networks with many network devices, the PROFINET device names are displayed in a slightly adapted form for PROFINET devices. Unicode-specific characters in the PROFINET device name are converted into ASCII-compatible characters using the Punycode method (according to RFC 3492).

Displaying accessible devices in the project tree

To display accessible devices at an interface of the programming device / PC, follow these steps:

1. Open the "Online access" folder in the project tree.
2. Click on the arrow to the left of the interface to show all the objects arranged below the interface.
3. Double-click "Update accessible devices" below the interface.
   You can see the progress of the search in the status bar. If you have found the desired device before the search is completed, you can cancel the search. To do this, click on the cross to the right of the progress bar.
   The network stations that have been found through the search are displayed below the interface.
4. Optional: Select the interface in the project tree. Then open the overview window in the detail view to display more details for the respective devices. More details include, for example, the device type, the MAC address or the configured device name.
   The detailed display of the devices in the overview window can take some time in networks with many devices because each individual device must submit the details.

Displaying accessible devices in a table

To display accessible devices in an easy to read table outside the project tree, follow these steps:

1. Select the "Accessible devices" command in the "Online" menu.
   The "Accessible devices" dialog is displayed.
2. Select the type of interface from the "Type of the PG/PC interface" drop-down list.
3. Select the required interface of the programming device / PC from the "PG/PC interface" drop-down list.

4. Click the "Start Search" button.
   The accessible devices at the selected interface of the programming device/PC are displayed in the table. The connection line between PG/PC and the network is closed and displayed in green in the graphic. If no devices are accessible at an interface, the connection line is displayed as a dashed line.
5. To go to a device in the project tree, select the device from the list of accessible devices and click the "Show" button.
   The selected device is highlighted in the project tree.

Displaying additional information about the devices in the project tree

To display additional information on the individual accessible devices in the project tree, follow these steps:

1. Click on the arrow to the left of one of the accessible devices in the project tree.
   All data available online, for example blocks and system data, is displayed for known devices. If additional editing options are available for a device with the shortcut menu, the device is shown in black text. You have only read access to the objects with a gray background, such as blocks.

Display additional information about the devices in the detailed view

Follow these steps to display additional information about the devices in the detailed view of the overview window:

1. Double-click the entry "Update accessible devices" in the "Online access" folder in the node of the network card.
   All the devices accessible in the network are listed below the interface.
2. Double-click the entry "Display more information".
   More information is displayed in the detailed view of the overview window.

See also

Changing the device configuration online (Page 11)

Default setting online connection data (Page 17)

Changing the device configuration online 2

You can assign parameters to some devices, preferably in small hardware setups, directly online. You do not need to create a project or have offline data to do this. In this way, you can quickly and easily change the device configuration. You do not need to compile the hardware configuration or perform a download. Depending on the device, either all changes become active immediately or they are written to the device only after confirmation.

Requirement

- The device must support online parameter assignment. You can learn whether or not your specific devices support this function in the device manual.
- The device must be connected to the PG/PC and available in the list of accessible devices.

Procedure

To change the device configuration online, follow these steps:

1. Show the accessible devices on the interface via which the device is connected. To learn how to show the accessible devices, see the chapter "Showing accessible devices (Page 9)".
2. Expand the device to display the lower-level elements.
3. Double-click the "Parametrize device" item.
   A configuration page for the device opens in the work area.
4. Make all required settings.
   With some devices, the new settings take effect immediately.
5. Optionally, depending on the device: Click on the "Upload to device" button.
   The settings are transferred to the device.

Connecting devices online 3

3.1 General information about online mode

Online mode

In online mode, there is an online connection between your programming device / PC and one or more devices.

An online connection between the programming device/PC and the device is required, for example, for the following tasks:

- Testing user programs
- Displaying and changing the operating mode of the CPU
- Displaying and setting the date and time of day of the CPU
- Displaying module information
- Comparing blocks
- Hardware diagnostics

Before you can establish an online connection, the programming device/PC and the device must be physically or remotely connected. As an alternative, some devices support a simulation mode. In this case, a connection to the device is simulated via the PLCSIM virtual interface.

After establishing a connection, you can use the Online and Diagnostics view or the "Online tools" task card to access the data on the device. The current online status of a device is indicated by an icon to the right of the device in the project tree. You will find the meaning of the individual status icons in the relevant tooltip.

Note

Some online functions depend on the scope of the installed software or whether a project is open.

Standby or hibernation of the programming device / PC

If the programming device / PC is changed to the standby or hibernation mode when there is an online connection, all online connections are terminated. When the programming device / PC wakes up from hibernation, the online connections are not automatically re-established.

Note that suddenly terminating an online connection can lead to loss of data or a connected device may interrupt program execution.

Performing an LED flash test

In many online dialogs you can perform an LED flash test, if the device connected online supports this feature. If you select the "Flash LED" check box, an LED flashes on the currently selected device. This feature is useful, for example, when you are not sure which device in the hardware configuration corresponds to the station currently selected in the software.

Read any additional information and learn about the possible limitations to the LED flash test in the respective device documentation.

See also

View in online mode (Page 15)

3.2 View in online mode

Online displays

After the online connection has been established successfully, the user interface changes. If a device is unavailable, this is indicated by a symbol. The following figure shows a device connected online and the corresponding user interface:

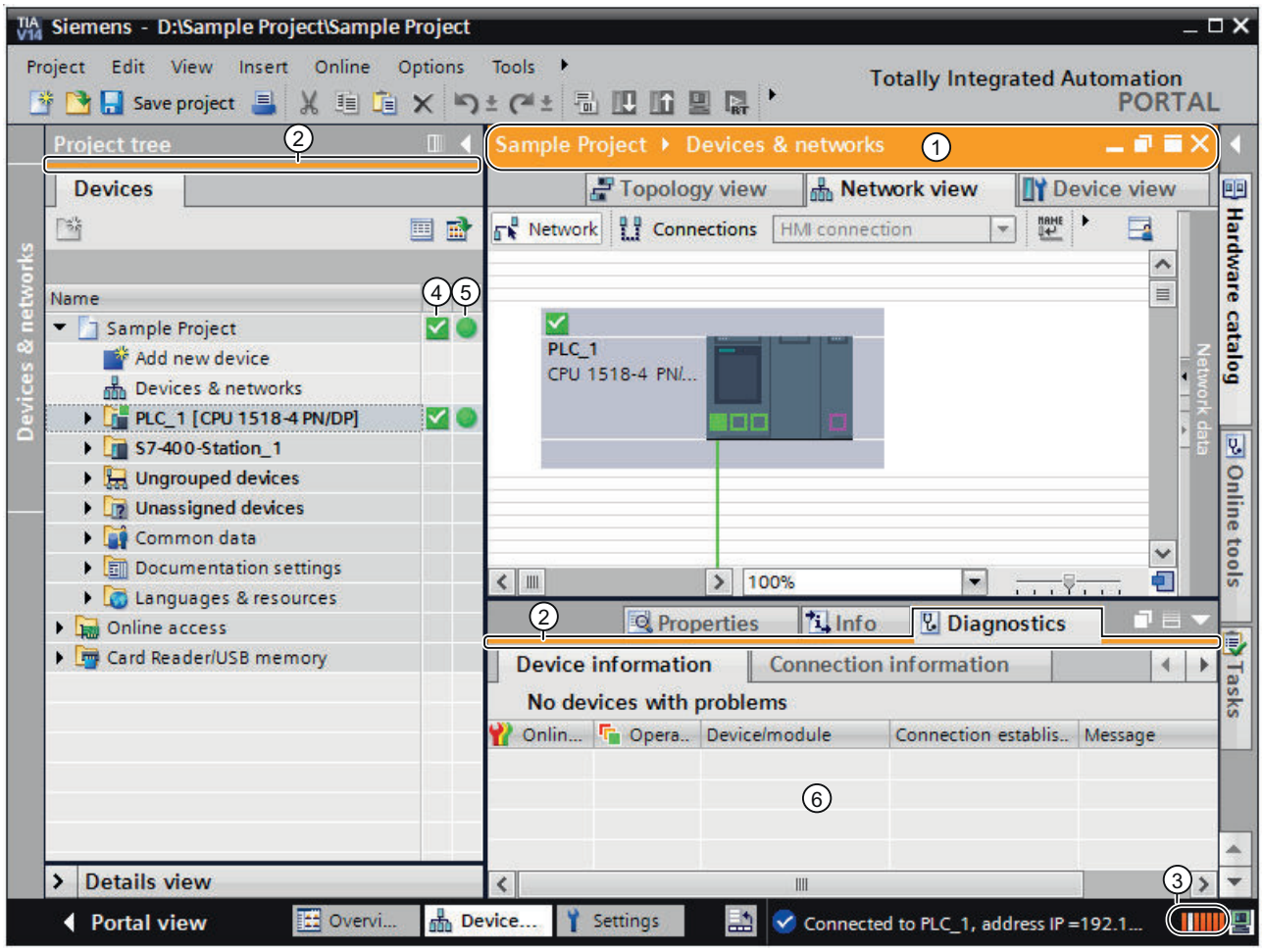

① The title bar of the active window gets an orange background as soon as at least one of the devices currently displayed in the editor has been successfully connected online. If one or more devices are unavailable, a symbol for a broken connection appears in the title bar of the editor.

② The title bars of inactive windows for the relevant station now have an orange line below them.

③ An orange, pulsing bar appears at the right-hand edge of the status bar. If the connection has been established but is functioning incorrectly, an icon for an interrupted connection is displayed instead of the bar. You will find more information on the error in "Diagnostics" in the Inspector window. In addition, the status line indicates if there is an online connection to the SIMATIC hardware via the TIA Portal Cloud Connector.

④ The left column in the project tree indicates the diagnostics status for hardware objects in online mode through symbols.

⑤ The right column in the project tree indicates the comparison status for software objects in online mode through symbols. A comparison of the online and offline state is performed automatically. Differences between online and offline objects are displayed in the form of symbols.

⑥ The "Diagnostics > Device information" area is brought to the foreground in the Inspector window.

Online connection abort

The online mode and its display are retained as long as at least one device is connected online. If the online connection of one or more devices breaks, the TIA Portal remains in online mode. The display of the TIA Portal changes to offline mode only when there is no longer an online connection for any device.

Online connection via the TIA Portal Cloud Connector

If you use the TIA Portal in a virtual environment, you have the opportunity to create an online connection via the TIA Portal Cloud Connector to SIMATIC hardware that is connected to a user device. If the online connection is made via the TIA Portal Cloud Connector, the following symbol is displayed in the status bar:

You can find instructions on how to use the TIA Portal in a virtual environment (private cloud) on the installation disk in the directory "Documents\Readme\<language directory>". You can open the PDF document "TIAPortalCloudConnectorHowTo<Sprachkennung>.pdf" here.

See also

General information about online mode (Page 13)

3.3 Default setting online connection data

If you prefer to use a specific network interface of your programming device/PC to establish an online connection, you can preset this interface. The specified connection path is then already selected in the dialogs for online connection.

Procedure

To specify a default connection path, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are opened.
2. Select the "Online & Diagnostics" entry in the area navigation.
3. In the "Preset connection path for online access" area, specify the type of the PG/PC interface as well as the interface itself.
4. Select the "Use preset connection path for online access" check box.

See also

Establishing or changing an online connection (Page 18)

Displaying accessible devices (Page 9)

Online access (Page 35)

3.4 Establishing or changing an online connection

To use the online functions of a device, specify the connection path the first time you establish the connection.

Requirement

At least one PG/PC interface is installed and is physically connected to a device, for example, with an Ethernet cable. As an alternative, it is also possible to establish virtual online connections using PLCSIM.

Procedure

To establish or change an online connection, follow these steps:

1. Select one or more devices for the online connection in the project tree.
2. Select the "Go online" command in the "Online" menu.
   - If the device has already been connected online, the online connection is automatically established using the previously specified connection path.
   - If there was no previous connection or if the device address is not located in the same subnet of the programming device/PC, the "Go online" dialog opens.

   If you want to change a specified connection path, select the "Extended go online" command instead in the "Online" menu. The "Go online" dialog box is opened again in this case.
3. Select the connection path:
   - Select the type of interface from the "Type of the PG/PC interface" drop-down list.
   - Select the interface of your programming device/PC from the "PG/PC interface" drop-down list.
     No selection is necessary if you have already defined a programming device/PC interface in the settings.
   - Select the interface or the subnet for the connection from the "Connection to interface/subnet" drop-down list. Select a direct connection if no S7 gateway is connected in between. Select the matching subnet for the connection to the programming device or PC if the device can be accessed via an S7 gateway. Select the item "Try all interfaces" if the device has several interfaces and you do not know by which of the interfaces the device is connected to the programming device/PC.
     If you selected an MPI or PROFIBUS subnet, the bus parameters configured in the programming device/PC interface are applied at this point.
   - If the device is accessible via a gateway, select the gateway that connects the two subnets involved in the "1st gateway" drop-down list.
4. Optional: If no devices can be accessed by the selected connection path, a dashed connecting line is displayed between the programming device/PC and the device. This is the case, for example, when the device is connected with the network by a NAT router. To display all devices that are switched on and connected to an interface of the programming device/PC, select the entry "Show accessible devices" from the drop-down list and click "Start search" again.

5. Optional: You have two options to access the device despite a different subnet: Use the two options in administered networks only after consultation with the administrator.
   - Assign a static address to the programming device/PC interface under Windows.
   - During the creation of the online connection, an additional IP address in the same subnet as the device is temporarily assigned to the interface of the programming device/PC. Click "Go online" and confirm the next prompt.
6. Click the "Start search" button.
   Devices which can be reached by the set connection path are displayed in the table of the target devices. The connection line in the graphic on the left is displayed as solid.
7. Optional: Run a flash test. Select the "Flash LED" check box on the left in the graphic. You use the flash test to check that you have selected the correct device. The flash test is not supported by all devices.
8. Select the target device in the table and confirm the selection with "Go online".
   The online connection to the selected target device is established.

Result

After the online connection has been established, the title bars of the editors change to orange. An orange progress bar is also shown in the title bar of an editor and in the status bar. In the project tree, status symbols show the difference between online and offline objects.

The set connection path is stored for future connection attempts.

With automatic assignment of the programming device/PC address the dialog reappears when the computer is restarted. The connection must be re-established in this case.

Note

Undoing actions

You cannot undo actions once you have established an online connection.

Note

Using direct connections or subnets

An online connection between programming device/PC and connected devices is possible either via direct connection or via subnet.

Simultaneous connection of the devices over different subnets is not possible if these are physically connected through the same interface. Simultaneous connections to different subnets are only possible through routed connections.

Simultaneous connection of devices through a subnet and through direct connections using the same interface is not possible.

See also

Default setting online connection data (Page 17)

Canceling an online connection (Page 21)

Connecting online with several devices (Page 22)

View in online mode (Page 15)
Utilization of temporary IP addresses (Page 44)

3.5 Canceling an online connection

Procedure

To terminate the existing online connection, follow these steps:

1. Select the device for which you want to disconnect the online connection in the project tree.
2. Select the "Go offline" command in the "Online" menu.
   The online connection is disconnected.

Note

Undoing actions

You cannot undo actions once you have disconnected an online connection.

See also

Establishing or changing an online connection (Page 18)

3.6 Connecting online with several devices

You can establish an online connection to several devices at the same time without needing to select individual devices previously in the network view.

Requirement

- No device must be selected
- At least one PG/PC interface is installed and is physically connected to a device, for example with an Ethernet cable. As an alternative, it is also possible to establish a virtual online connection using PLCSIM or a remote connection.

Procedure

To establish an online connection to several devices at the same time, follow these steps:

1. Select the project in the project tree.
2. Select the "Go online" command in the "Online" menu.
   The "Select device for opening the online connection" dialog opens with a table of all available devices.
3. Select the devices to which you want to establish an online connection in the "Go online" column.
4. Click the "Go online" button.

Result

Without any further prompt for confirmation, a connection is established to all selected devices if a connection was already established to the selected devices at least once. If there was no previous online connection, the "Go online" dialog opens. In this case, first configure the online connection as described in the section "Go online and Go offline (Page 18)".

See also

Establishing or changing an online connection (Page 18)

Utilization of temporary IP addresses (Page 44)

3.7 Disconnecting online connections of multiple devices

You can disconnect the online connections to multiple devices at one time without needing to select individual devices beforehand in the network view.

Requirement

- No device is selected.
- There is currently an online connection to at least one device.

Procedure

To terminate the online connections to multiple devices at one time, follow these steps:

1. Select the "Go offline" command in the "Online" menu.
   The "Select devices" dialog opens with a table of all available devices.
2. Select the device for which you want to terminate the online connection in the "Go offline" column.
3. Click the "Go offline" button.

Result

The online connection to the all the selected devices is terminated.

Creating a backup of an S7 CPU 4

4.1 Backup options for S7 CPUs

You will make a number of changes to your automation system over time, for example, add new devices, replace existing devices or adapt the user program. If these changes were to result in undesirable behavior, you can restore the automation plant to an earlier version. To continue working without interruptions after replacing individual devices, you can accept existing programs and values. The CPUs offer different options for backup and restoration of the hardware configuration and software.

Backup options

The table below provides an overview of the backup and restoration options of S7 CPUs:

	Snapshot of the monitored values	Upload from device (software)	Upload device as new station (hardware and software)	Download backup from online device
Use case	Restoring a specific status of a data block. The actual values of data blocks including time stamp are accepted in the project.	Upload blocks on a CPU to the project.	Upload of hardware configuration and software from a device to the project.	Create a complete backup of a CPU as restore point. The backup copy is consistent and cannot be changed or opened.
Requirement	The CPU has already been created in a project. The data blocks must be identical online and offline.	The CPU is created in the project.	The device is available in the hardware catalog of TIA Portal. Any necessary HSPs or GSD files are installed.	-
Possible in mode	RUN, STOP	RUN, STOP	RUN, STOP	STOP
Possible for F-CPUs	Yes	Yes	Yes	Yes
Backup can be edited	Yes	Yes	Yes	No

Backup contents

The table below shows which data you can upload and back up with which options:

	Snapshot of the monitored values	Upload from device (software)	Upload device as new station (hardware and software)	Download backup from online device
Actual values of the data blocks	Snapshot is possible	Upload is possible	Upload is possible	Backup is possible
Software blocks	-	Upload is possible	Upload is possible	Backup is possible
PLC tags (tags and constants names)	-	Upload possible for S7-1200 and S7-1500 CPUs	Upload possible for S7-1200 and S7-1500 CPUs	Backup is possible
Technology objects	-	Upload is possible	Upload is possible	Backup is possible

	Snapshot of the monitored values	Upload from device (software)	Upload device as new station (hardware and software)	Download backup from online device
Hardware configuration	-	-	Upload is possible	Backup is possible
Monitoring tables (web server)	-	-	Upload is not possible	Backup is possible
Local data, bit memories, timers, counters and process image	Snapshot is not possible	Upload is not possible	Upload is not possible	Backup is possible
Archives and recipes (PLC)	-	-	-	Backup is possible
General data on the SIMATIC Memory Card, for example, help for program blocks or GSD files	-	-	-	Backup is possible

Special considerations during backup of actual values

The "Backup from online device" type of backup backs up the actual values of the tags that are set as retentive. To ensure consistency of the retentive data, disable all write access to retentive data during the backup.

Actual values of the non-retentive data are reset to their start values during a transition from STOP to RUN mode. When the CPU is backed up only the start values of non-retentive data are backed up.

See also

Creating a backup of a device (Page 27)

Creating a backup of a device (Page 30)

4.2 Backing up S7-300 and S7-400 CPUs (S7-300, S7-400, PC)

4.2.1 Creating a backup of a device (S7-300, S7-400, PC)

Backing up the software and hardware configuration of an S7-300/400 CPU

If you have already downloaded a configuration to an S7-300/400 CPU, it is advisable to make a backup. You may have modified the configuration and want to test the new configuration. Before you download the new configuration to the CPU, you can create a backup of the current device state and then restore the current configuration at a later date. The backup is performed with the current values of the CPU. In the case of S7-400 CPUs with fail-safe function, the initial values are backed up.

You can create as many backups as you want and store a variety of configurations for a CPU. The backups are named with the name of the CPU and the time and date of the backup. You can find the backup in the project tree under the CPU in the "Online backups" folder.

The following figure shows an S7-319 CPU for which two backups were created:

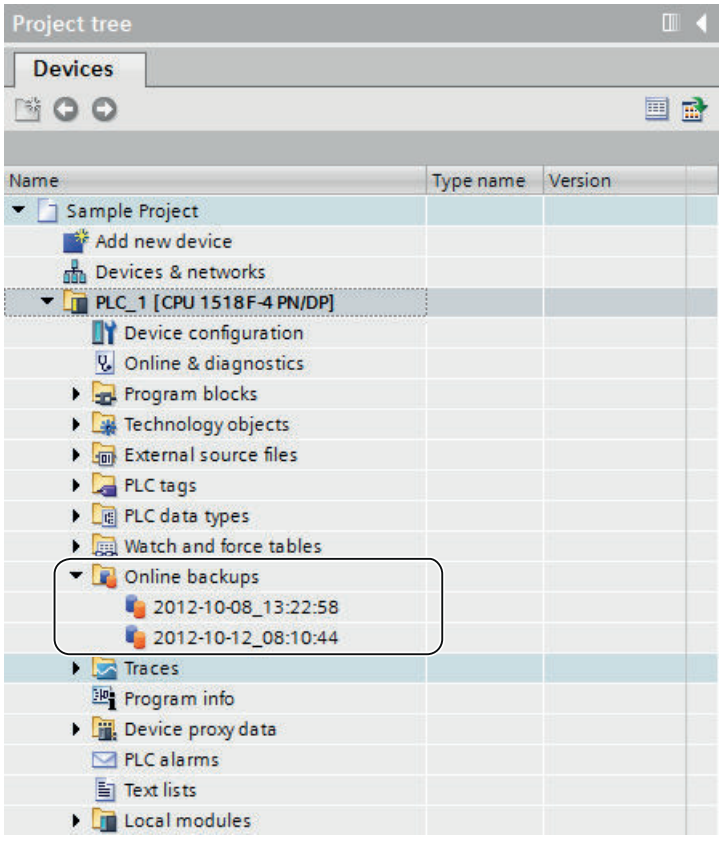

See also

Backup options for S7 CPUs (Page 25)

4.2.2 Restoring the software and hardware configuration of a device (S7-300, S7-400, PC)

If you have backed up the configuration of a device at an earlier point in time, you can transfer the backup back to the device. The saved configuration is then restored on the device.

Requirement

You must have previously configure the device and stored a backup of the device in the project.

Procedure

To restore older software and hardware state on a device, follow these steps:

1. Open up the folder of the device in the project tree to display the lower-level objects.
2. Open the "Online backups" folder.
3. Select the backup you want to restore.
4. In the "Online" menu, select the "Download to device" command.
   - If you had previously established an online connection, the "Load preview" dialog opens. This dialog displays alarms and recommends actions needed for the loading operation.
   - If you had not previously established an online connection, the "Extended download to device" dialog opens, and you must first select the interfaces via which you want to establish the online connection to the device.
     See also: Establishing and terminating an online connection
5. Check the alarms in the "Load preview" dialog, and select the actions in the "Action" column, if necessary.

   Note

   Performing the proposed actions during operation of the plant can cause serious damage to property or injury to persons if there are functional disturbances or program errors.

6. As soon as downloading becomes possible, the "Load" button is enabled.
7. Click the "Load" button.
   The backup is transferred to the device and device is restored. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
8. Click the "Finish" button.

4.2.3 Backing up a device configuration (S7-300, S7-400, PC)

You can back up the configuration of an S7-300/400 CPU in the TIA Portal. So you can download and test a new configuration to a device without any risk. If needed, you can restore the initial configuration of the CPU.

Requirement

- The CPU has already been created in the project.
- The CPU is online. If there is no online connection yet, an online connection is established during the backup.

Procedure

To create a backup of the current configuration of a CPU, follow these steps:

1. Select the CPU in the project tree.
2. Select the "Backup from online device" command in the "Online" menu.

Result:

A backup of the entire hardware configuration and software is created. The backup is stored in the project tree in the "Name of the CPU > Online backups" folder. The backup is assigned the name of the CPU with the time and date of the backup. You can rename the backup, but you cannot make any changes to the contents of the backup.

4.3 Backing up S7-1200 and S7-1500 CPUs (S7-1200, S7-1500, S7-1200, S7-1500)

4.3.1 Creating a backup of a device (S7-1200, S7-1500, S7-1200, S7-1500)

Backing up the software and hardware configuration of a CPU

If you have already downloaded a configuration to an S7-1200/S7-1500 CPU, it may be useful to make a backup. You may have modified the configuration and want to test the new configuration. Before you download the new configuration to the CPU, create a backup of the current device version. You can restore the current configuration at a later time.

You can create as many backups as you want and store a variety of configurations for a CPU. The backups are named with the name of the CPU and the time and date of the backup. You can find the backup in the project tree under the CPU in the "Online backups" folder.

The following figure shows an S7-1500 CPU for which two backups have been created:

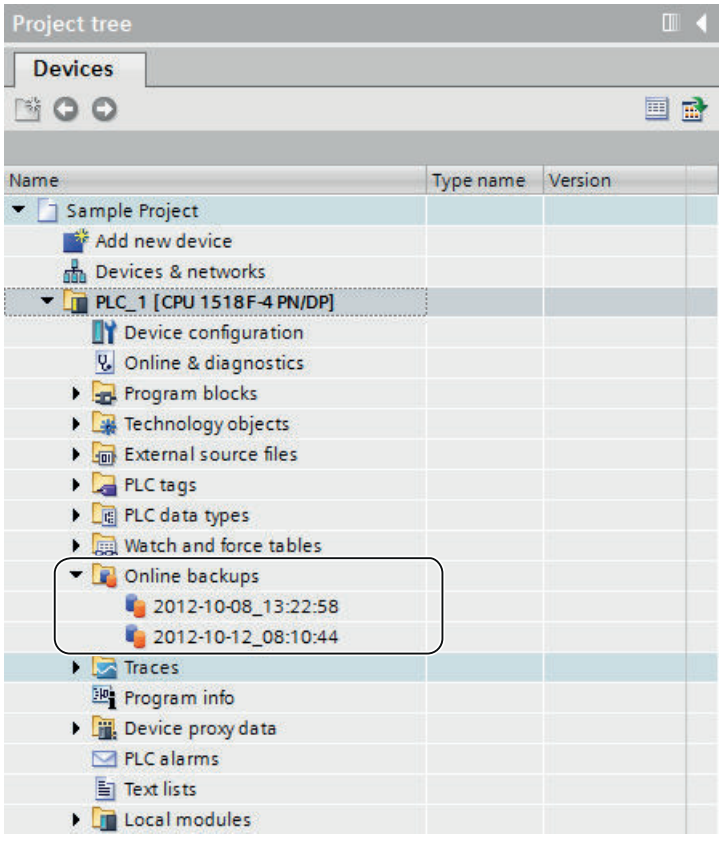

Scope of the backup

The backup includes all data that are needed to restore a particular configuration version of a CPU. The following data is backed up, for example:

- Contents of the memory card
- Retentive memory areas for example of data blocks, counters, and bit memory
- Other retentive memory contents, such as IP address parameters

The backup is performed with the current values of the CPU. Entries in the diagnostic buffer are not included in the backup. The current time of day is not backed up for an S7-1500 CPU.

See also

Backup options for S7 CPUs (Page 25)

4.3.2 Backing up a device configuration (S7-1200, S7-1500, S7-1200, S7-1500)

You can back up the configuration of a CPU in the TIA Portal. So you can download and test a new configuration to a device without any risk. If needed, you can restore the initial configuration of the CPU.

The CPU goes to STOP while a backup is being created. If an access level is configured for the CPU, you need the password for read access to the CPU.

Requirement

- The CPU has already been created in the project.
- The device is connected to the programming device/PC directly via the PROFINET interface of the CPU. PROFIBUS interfaces of the CPU and interfaces of CMs/CPs are not supported.
- The CPU is online. If there is no online connection yet, an online connection is established during the backup.
- The CPU is in "STOP" mode.

Procedure

To create a backup of the current configuration of a CPU, follow these steps:

1. Select the CPU in the project tree.
2. Select the "Backup from online device" command in the "Online" menu.
   If necessary, you must enter the password for read access to the CPU and confirm that the CPU should enter "STOP" mode.

Result:

A backup of the entire hardware configuration and software is created. The backup is stored in the project tree in the "Name of the CPU > Online backups" folder. The backup is assigned the name of the CPU with the time and date of the backup. You can rename the backup, but you cannot make any changes to the contents of the backup.

An entry is created for each backup operation in the diagnostic buffer of the CPU.

See also

Restoring the configuration of a device (Page 32)

4.3.3 Restoring the configuration of a device (S7-1200, S7-1500, S7-1200, S7-1500)

If you have backed up the configuration of a device at an earlier point in time, you can transfer the backup back to the device. The saved configuration is then restored on the device.

The CPU goes to STOP while a backup is being loaded to the CPU. If an access level is configured for the CPU, you need the password for write access to the CPU.

WARNING

Download backups with unknown content

If you activate the recommended actions during download during plant operation, you can cause serious damages or injuries in case of malfunctions or program errors.

Make sure that the backup contents do not include a configuration which results in unexpected plant behavior.

Requirement

- The device is configured and a backup of the device is stored in the project.
- The device is connected to the programming device/PC directly via the PROFINET interface of the CPU.
- The CPU is in "STOP" mode.
- You have the password for read access to the CPU, if an access level was configured.

Procedure

To restore older software and hardware state on a device, follow these steps:

1. Open up the folder of the device in the project tree to display the lower-level objects.
2. Open the "Online backups" folder.
3. Select the backup you want to restore.

4. In the "Online" menu, select the "Download to device" command.
   - If you had previously established an online connection, the "Load preview" dialog opens. This dialog displays alarms and recommends actions needed for the loading operation.
   - If you had not previously established an online connection, the "Extended download to device" dialog opens, and you must first select the interfaces via which you want to establish the online connection to the device.
     See also: AUTOHOTSPOT
5. Check the alarms in the "Load preview" dialog, and select the actions in the "Action" column, if necessary.
6. As soon as downloading becomes possible, the "Load" button is enabled.
7. Click the "Load" button.
   The backup is transferred to the device and device is restored. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
8. Click the "Finish" button.
   If necessary, you must enter the password for read access to the CPU and confirm that the CPU should enter "STOP" mode.
   The contents of the backup are restored on the CPU. The CPU is then restarted.

Configuring the PG/PC interface 5

5.1 Online access

Online access of the project

In the "Online access" folder of the project tree, you will find all active interfaces of your programming device/PC. Each interface icon provides you with information on the status. You can also display the accessible devices and display and edit the properties of an interface using the shortcut menu.

The following figure shows the "Online access" folder in the project tree:

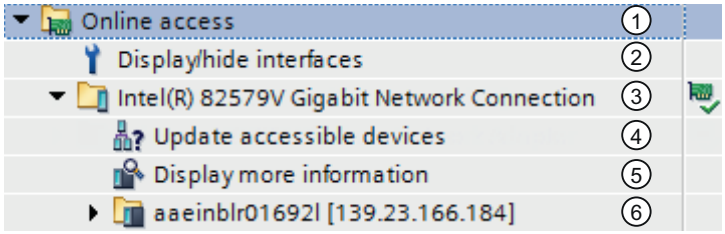

① "Online access" folder in the project tree

All interfaces installed in the programming device/PC are displayed in the "Online access" folder.

② Show/hide interfaces

Use the function "Show/hide interfaces" to show or hide individual interfaces.

③ Status display for the interfaces

The current status of an interface is indicated by an icon to the right of the name. You can see the meaning of the icon in the tooltip.

④ Updating the list of accessible devices.

This function is available for each hardware interface of the programming device/PC. Software interfaces, such as a remote connection, do not offer this function.

⑤ Display more information

This function is available for each hardware interface of the programming device/PC after you have executed the function "Update accessible devices". You can find additional information in the detailed view of the overview window.

⑥ Devices connected via the respective interface with the programming device/PC

The type of the device and its status are displayed by the preceding icon.

Displaying or updating accessible devices

You have the following options if you want to display all devices accessible online on your programming device/PC:

- Display of the accessible devices on a single interface of the programming device / PC in the project tree. In the project tree, you can also display additional information about the individual accessible devices.
- Display of the accessible devices of all interfaces in a list.

See also:AUTOHOTSPOT

Overview of icons for accessible devices

The accessible devices are identified with an icon according to their type and status. The following is an overview of all icons and their meaning.

	Icon for unidentified modules This icon is displayed whenever the identification of a module is not yet complete or when the identification of a module was not successful, for example, because the required online data could not be read.
	Icon for the following device types: • PLCs • SIMOCODE pro devices • IE/PB links • CPs of PC systems • SCALANCE head modules • S7-300 and S7-400 CPs • PROFINET IO devices and PROFINET CPs • SCALANCE modules and gateways that could not be identified
	PROFINET IO devices, encoders, switchgear, sensors and identification systems that were replaced by similar devices because these could not be identified
	Icon for the following device types: • HMI devices • PROFINET IO devices of the HMI type if these could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the drive type that could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the type development kit and network components that could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the type TeleService adapter that could not be identified and were therefore replaced by a similar device

See also

Displaying and modifying interface properties (Page 39)

5.2 Basics of assigning parameters for the PG/PC interface

Options for connecting to target systems

If the devices of the project are connected via different subnets, you assign a suitable network access to each PG/PC interface to be able to establish online connections to the target systems. The following interfaces are automatically supported:

- MPI
- PROFIBUS
- Industrial Ethernet (ISO and TCP/IP)

You can make various settings for the interfaces. The following sections explain the parameter settings you can make.

Note

Note that changes to interface parameters have a direct influence on the operating system and the programming device / PC. Remember that some parameter settings can only be changed if you have adequate user rights.

See also

Setting parameters for the Industrial Ethernet interface (Page 41)

Setting parameters for the MPI and PROFIBUS interfaces (Page 46)

5.3 Showing or hiding interfaces

To improve clarity, it is possible to hide individual interfaces of the PG/PC in the project tree and to show them again when necessary.

Procedure

To show or hide individual interfaces, follow the steps below:

1. Open the "Online access" folder in the project tree.
2. Double click on the "Display/hide interfaces" icon.
   The "Display/hide interfaces" dialog opens.
3. Enable or disable the required interfaces in the "Display in the project tree" column.
4. Click the "Apply" button.
   The changes are applied. The view of the interfaces in the "Online access" folder is refreshed.

5.4 Displaying and modifying interface properties

Introduction

For each interface, you can display and, in some cases, modify properties, for example the network type, address, and status.

Procedure

To open the properties, follow these steps:

1. Right-click on the required interface below "Online access" in the project tree.
2. Select the "Properties" command from the shortcut menu.
   A dialog containing the properties of the interface opens. On the left of the dialog, you will see the area navigation. You can view the current parameter settings in the individual entries in the area navigation and, if necessary, change them.

5.5 Adding interfaces

You have the option of installing additional interfaces after installation of the TIA Portal.

Procedure

To install an interface at a later time and add it to the TIA Portal, follow these steps:

1. Install or update the drivers in the operating system once you have installed the interface hardware.
2. Close the TIA Portal if it is still open.
3. Open the Windows control panel.
4. Open the entry "Setting the PG/PC Interface" in the Control Panel.
   The "Setting the PG/PC Interface" dialog opens.
5. Make any necessary changes to the interface configuration and confirm them with "OK".
   You have to click "OK", even if you have not made any changes.
6. Restart the TIA Portal.

Result

The newly installed interface is now displayed in the project tree under the "Online access" folder.

5.6 Setting parameters for the Ethernet interface

5.6.1 Setting parameters for the Industrial Ethernet interface

Options in the parameter settings for the Industrial Ethernet interface

When setting parameters for the Industrial Ethernet interface, you have the following options:

- Parameters dependent on the operating system
  The Industrial Ethernet interface has parameters that are set in the operating system and are valid for all connected devices. These parameter settings are only displayed here, they can, however, be changed in the network settings of the operating system.
- Parameters that can be set in the software

Note

Note that changes to interface parameters have a direct influence on the operating system and the programming device / PC. Remember that some parameter settings can only be changed if you have adequate user rights.

Parameters for the Industrial Ethernet interface

The following table contains an overview of the parameters of the Industrial Ethernet interface that are set by the operating system and can be changed by the user.

Parameter settings that cannot be changed	Parameters that can be set
MAC address	Fast acknowledge at the IE-PG access and for TCP/IP
DHCP server activated/deactivated	Timeout at the IE-PG access and for TCP/IP
APIPA activated/deactivated	LLDP
IP address	Additional, dynamic IP addresses for the network adapter
Subnet mask	-
DNS addresses	-
DHCP addresses	-

See also

Basics of assigning parameters for the PG/PC interface (Page 37)

Displaying operating system parameters (Page 42)

Connecting the PG/PC interface to a subnet (Page 42)

Setting parameters for the Ethernet interface (Page 43)

Utilization of temporary IP addresses (Page 44)

Managing temporary IP addresses (Page 44)

5.6.2 Displaying operating system parameters

The Ethernet interface is part of the operating system. All parameters of the network adapter can therefore be adapted in the network settings of the operating system.

You can display the following parameters in the software:

- Physical address of the network adapter
- Assignment of the IP address by a DHCP server activated or deactivated
- Assignment of a private IP address by the operating system activated or deactivated
- Current static IP address
- Assigned subnet mask
- DNS addresses
- DHCP addresses

If you want to modify the parameter settings, please refer to the documentation of the operating system or the network adapter.

Displaying current parameters of the Ethernet interface

To display the current parameters of the Ethernet interface, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > Industrial Ethernet" in the area navigation.

See also

Setting parameters for the Ethernet interface (Page 43)

5.6.3 Connecting the PG/PC interface to a subnet

If you have created several subnets, you can specify the subnet to which the Ethernet interface is connected.

Procedure

To select the subnet to which the Ethernet interface is connected, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.

3. Go to "General > Assignment" and select the subnet to which you want to connect the Ethernet interface of the programming device / PC in the "Connection to subnet" drop-down list.
4. Close the dialog with "OK".

5.6.4 Setting parameters for the Ethernet interface

You can adapt some parameter settings relating to the network protocol directly in the software.

Requirement

You must have adequate user rights.

See also: Influence of user rights.

Procedure

To change parameter settings relating to the network protocol, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > IE-PG access" to adapt the protocol settings relevant for network management.
   - Select the "Fast acknowledge" check box to achieve faster reaction times with smaller network packets.
   - From the "Timeout" drop-down list, select the maximum time that can elapse before a network node is detected.
4. To activate the LLDP protocol and discover the network topology more accurately, set the "LLDP active" check box in "Configurations > LLDP".
5. Select "Configurations > TCP/IP" to adapt the TCP/IP protocol for network traffic during runtime.
   - Select the "Fast acknowledge" check box to achieve faster reaction times with smaller network packets.
   - From the "Timeout" drop-down list, select the maximum time that can elapse before there is a timeout during communication with a network node.

See also

Displaying operating system parameters (Page 42)

5.6.5 Utilization of temporary IP addresses

Adding a temporary IP address

If the IP address of a device is located in a different subnet from the IP address of the network adapter, you will first need to assign an additional IP address with the same subnet address as the device. Only then is communication between the device and the programming device / PC possible.

The assignment of an additional temporary IP address is also proposed automatically if you want to perform an online action and the current IP address of the programming device/PC is not yet in the correct subnet.

A temporarily assigned IP address remains valid until the next time the programming device/ PC is restarted or until you delete it manually. See also: Managing temporary IP addresses (Page 44)

Note

You require adequate permissions to be able to assign a temporary IP address.

See also: Influence of user rights

5.6.6 Managing temporary IP addresses

If the IP address of a device is located in a different subnet from the current permanently assigned IP address of the network adapter, a suitable IP address from the subnet of the device is temporarily assigned to the network adapter.

You can display all temporarily assigned addresses and delete them. Note that IP addresses that you manually assigned in the operating system are not displayed in the TIA Portal.

Requirement

To delete, you require adequate permissions.

Procedure

To display and delete temporarily assigned addresses, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > IE-PG access".
   A table with the assigned IP addresses is displayed.
4. Click the "Delete project-specific IP addresses" button to delete all the IP addresses at one time.

5.6.7 Resetting the TCP/IP configuration

If you have changed the TCP/IP protocol settings, you can reset them to the defaults.

Procedure

To restore the TCP/IP configuration to the default settings, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > TCP/IP".
4. Click the "Standard" button to reset all the settings.

5.7 Setting parameters for the MPI and PROFIBUS interfaces

5.7.1 Setting parameters for the MPI and PROFIBUS interfaces

Possible parameter settings for the MPI and PROFIBUS interfaces

The following parameter settings can be made for the MPI and PROFIBUS interfaces:

- Automatic configuration: You can use automatic detection functions to find out whether a device is connected to the PG/PC interface over PROFIBUS or MPI.
- Selecting a default configuration for PROFIBUS or MPI that can be adapted later.

Device- and network-related settings for MPI and PROFIBUS

You can set device- and network-related parameters for MPI and PROFIBUS interfaces. Device-related parameters are local settings for the interface. Network-related parameters, on the other hand, must match up on all devices.

MPI interface parameters you can modify

You can adapt the following default parameters for the MPI interface:

Device-related parameters	Network-related parameters
Is only master	Highest address
Own address	Transmission rate
Timeout	

PROFIBUS interface parameters you can modify

You can adapt the following default parameters for the PROFIBUS interface:

Device-related parameters	Network-related parameters
Is only master	Highest address
Own address	Transmission rate
Timeout	Profile
	Bus parameters
	Number of masters on bus
	Number of slaves on bus

Note

Behavior when a subnet is assigned

If you permanently assign a subnet from the project to the interface, the subnet settings of the project take precedence. In this case, the settings for the respective interface cannot be changed.

See also

Basics of assigning parameters for the PG/PC interface (Page 37)

5.7.2 Setting MPI or PROFIBUS interface parameters automatically

Setting up automatic bus parameter detection

If you select an interface with automatic detection of the bus parameters (for example CP 5611 (Auto)), you can connect the programming device or PC to MPI or PROFIBUS without needing to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute.

Requirement

- Masters that distribute bus parameters cyclically are connected to the bus.
- In PROFIBUS networks, the cyclic distribution of bus parameters is enabled.
- The interface is not assigned to a subnet. If you assign a subnet in the properties of the interface, the settings for the subnet in the project take precedence. In this case, the settings for the automatic interface configuration cannot be changed.

Procedure

To enable automatic bus parameter detection, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Go to "General > Configurations > Active configuration" and select the setting "Automatic protocol detection".
4. Go to "Configurations > Auto configuration > Local settings" and select the address of the PG/PC interface in the "Own address" drop-down list.
5. If you then want to display the current bus settings, click the "Network detection" button.

See also

Setting parameters for the MPI interface (Page 48)

Setting parameters for the PROFIBUS interface (Page 49)

5.7.3 Setting parameters for the MPI interface

Changing the parameter settings of the MPI interface

The network-related parameters and bus parameters for the MPI network can be adapted. You should first select a default setting and then adapt this to the specific situation.

Requirement

The interface is not assigned to a subnet. If you assign a subnet in the properties of the interface, the settings for the subnet in the project take precedence. In this case, the interface configuration cannot be changed.

Setting defaults for the MPI interface

To adapt the parameters of the MPI interface, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Go to "General > Assignment" and select the subnet with which you want to connect the interface in the "Connection to subnet" drop-down list.
4. Under "General > Configuration", select a default for the device and network-related parameters. The defaults are suitable for most configurations. Select one of the following settings:
   - Automatic protocol detection
     You can connect the programming device to MPI or PROFIBUS without having to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute. Prerequisite for the automatic detection is a connection to the bus master, which distributes the bus parameters cyclically. With PROFIBUS subnets, cyclic distribution of bus parameters may not be deactivated (default PROFIBUS network setting).
   - MPI
     The "MPI" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.
   - PROFIBUS
     The "PROFIBUS" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.

Changing the default parameter settings

To adapt the default settings to your requirements, change the parameter setting where necessary in "Configurations > MPI".

You can set the following device-related parameters:

- Is only master
  An additional verification function to prevent bus disruptions when connecting the PG/PC to the network is disabled because the programming device or PC is the only master on the bus.
  - Do not enable this option unless you have only connected slaves to your programming device or PC.
  - If the "Is only master" check box is enabled, it is not possible to identify the directly connected device in the "Accessible devices" window.
- Own address
  This setting relates to the programming device or PC on which you call up the parameter settings of the interface. Set the local device address of your programming device or PC here.
  - This address must be unique throughout the network.
  - The programming device or PC is addressed using this address in the MPI network.
- Check
  This enables an additional safety function to prevent bus disruptions when connecting the PG/PC to the network. The driver checks whether the local address is already being used by another station. Active as well as passive stations are taken into consideration in this case. The driver monitors this on the PROFIBUS. The connection of the PG/PC to the network will take longer with the automatic check. To use the check, the driver must support the function. Furthermore, the "Is only master" option must not be selected.
- Timeout
  Set a higher timeout value if, for example, you have problems with long response times on the network.

You can set the following network-related parameters:

- Highest address:
  Select the configured highest device address. Make sure that the same highest device address is set for all devices of a PROFIBUS or MPI network.
- Transfer rate:
  Here, you select the transmission speed to be used on the MPI network.

See also

Setting MPI or PROFIBUS interface parameters automatically (Page 47)

5.7.4 Setting parameters for the PROFIBUS interface

Changing the parameter settings of the PROFIBUS interface

The network-related parameters and bus parameters for the PROFIBUS network can be adapted more precisely. You should first select a default setting and then adapt this to the specific situation.

Requirement

The interface is not assigned to a subnet. If you assign a subnet in the properties of the interface, the settings for the subnet in the project take precedence. In this case, the interface configuration cannot be changed.

Setting defaults for the PROFIBUS interface

To adapt the parameters of the PROFIBUS interface, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Go to "General > Assignment" and select the subnet with which you want to connect the interface in the "Connection to subnet" drop-down list.
4. Under "General > Configuration", select a default for the device and network-related parameters. The defaults are suitable for most configurations. Select one of the following settings:
   - Automatic protocol detection
     You can connect the programming device to MPI or PROFIBUS without having to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute. Prerequisite for the automatic detection is a connection to the bus master, which distributes the bus parameters cyclically. With PROFIBUS subnets, cyclic distribution of bus parameters may not be deactivated (default PROFIBUS network setting).
   - MPI
     The "MPI" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.
   - PROFIBUS
     The "PROFIBUS" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.

Changing the default parameter settings

To adapt the default settings to your requirements, change the parameter setting where necessary in "Configurations > PROFIBUS".

You can set the following device-related parameters:

- Is only master
  An additional verification function to prevent bus disruptions when connecting the PG/PC to the network is disabled because the programming device or PC is the only master on the bus.
  - Do not enable this option unless you have only connected slaves to your programming device or PC.
  - If the "Is only master" check box is enabled, it is not possible to identify the directly connected device in the "Accessible devices" window.
- Own address
  This setting relates to the programming device or PC on which you call up the parameter settings of the interface. Set the local device address of your programming device or PC here.
  - This address must be unique throughout the network.
  - The programming device or PC is addressed using this address in the PROFIBUS network.
- Check
  This enables an additional safety function to prevent bus disruptions when connecting the PG/PC to the network. The driver checks whether the local address is already being used by another station. Active as well as passive stations are taken into consideration in this case. The driver monitors this on the PROFIBUS. The connection of the PG/PC to the network will take longer with the automatic check. To use the check, the driver must support the function. Furthermore, the "Is only master" option must not be selected.
- Timeout
  Set a higher timeout value if, for example, you have problems with long response times on the network.

You can set the following network-related parameters:

- Highest address:
  Select the configured highest device address. Make sure that the same highest station address is set for all devices of a PROFIBUS network.
- Transfer rate:
  Here, you select the transmission speed to be used on the PROFIBUS network.
- Profile:
  You have a choice of four alternatives for the PROFIBUS settings. "DP", "Standard" and "Universal (DP/FMS)" are predefined settings that you cannot change. If you select "User-defined", you can adapt the bus parameters yourself.
  - If you have selected "User-defined", go to "Configurations > PROFIBUS > Bus parameters" in area navigation.
  - If you have selected one of the defaults (DP, Standard or Universal (DP/FMS)), you should select the "Include" check box in "Configurations > PROFIBUS > Bus parameters > Additional parameters". You can then set the number of masters and slaves on the bus. This allows a more precise calculation of the bus parameters and potential bus disruptions can be prevented. The option cannot be selected with a user-defined profile.

See also

Overview of the bus parameters for PROFIBUS (Page 52)

Setting MPI or PROFIBUS interface parameters automatically (Page 47)

5.7.5 Overview of the bus parameters for PROFIBUS

Introduction

The PROFIBUS subnet will only function problem-free if the parameters for the bus profile are matched to one another. You should therefore only change the default values if you are familiar with how to configure the bus profile for PROFIBUS.

It may be possible for the bus parameters to be adjusted depending on the bus profile. The offline values of the bus parameters are always shown even if you are online and linked to the target system.

The displayed parameters are valid for the entire PROFIBUS subnet.

Meaning of the individual parameters

- Tslot: Wait-to-receive time (slot time)
  The wait-to-receive time (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner.
- Max. Tsdr: maximum protocol processing time (max. station delay responder)
  The maximum protocol processing time defines the time after which the responding device must have processed the protocol.
- Min. Tsdr: minimum protocol processing time (min. station delay responder)
  The minimum protocol processing time specifies the minimum time required by the responding device to process the protocol.
- Tset: Trigger time (setup time)
  The trigger time is the time that may lapse between the reception of a data frame frame and the reaction to it.
- Tqui: Quiet time for modulator
  The quiet time for modulator specifies the time required to change from sending to receiving.
- GAP factor: GAP update factor (GAP factor)
  The GAP update factor specifies the number of token rotations before a new device is included in the token ring.
- Retry limit: maximum number repeated calls (retry limit)
  This parameter defines the maximum number of attempts (message frame repeats) made to reach a node.
- Trdy: Ready time
  The ready time is the time for an acknowledgment or response.
- Tid1: Idle time 1
  Idle time 1 specifies the delay time after receiving a response.

- Tid2: Idle time 2
  Idle time 2 specifies the delay time after sending a call without a response.
- Ttr: Target rotation time
  The target rotation time is the maximum time made available for a token rotation. During this time, all active devices (masters) receive the token once. The difference between the desired token round-trip time and the actual token round-trip time decides how much time is left for masters to send data message frames to the slaves.
  As the minimum target rotation time (Ttr), select a value = 5000 times the HSA (Highest Station Address).
- Watchdog: Watchdog
  The watchdog time specifies the time after which a device must be addressed.
  As the minimum watchdog time, select a value = 6250 times the HSA.

Note

If you want to create a user-defined bus profile, note that the minimum target rotation time (Ttr) should be 5000 times the HSA (highest PROFIBUS address). The minimum watchdog time should also be 6250 times the HSA.

See also

Setting parameters for the PROFIBUS interface (Page 49)

5.7.6 Resetting the MPI or PROFIBUS configuration

If you have changed the MPI or PROFIBUS protocol settings, you can reset them to the defaults.

Requirement

The interface is not assigned to a subnet. If you assign a subnet in the properties of the interface, the settings for the subnet in the project take precedence. In this case, the interface configuration cannot be reset to the default values.

Procedure

To restore the MPI or PROFIBUS configuration to the default settings, follow these steps:

1. Select the MPI/PROFIBUS interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > MPI" or "Configurations > PROFIBUS", depending on the interface properties you want to reset.
4. Click the "Standard" button to reset all the settings.

Using the trace and logic analyzer function 6

6.1 Preface

Purpose of the documentation

The diagnostics options available with the trace and logic analyzer function are described in this documentation. Depending on the device used, the recording options can vary.

Required basic knowledge

In order to understand this documentation, the following knowledge is required:

- General knowledge in the field of automation
- Knowledge about the use of Windows-based computers
- S7-1200/1500 CPUs, ET 200SP, ET 200Pro
  - Knowledge of the SIMATIC industrial automation system
  - Knowledge of working with STEP 7
- SINAMICS Drives
  - Knowledge of working with the drive
- SIRIUS SIMOCODE pro, SIRIUS Soft Starter 3RW
  - Proficiency in using these systems

Validity of the documentation

This documentation applies to all products of the product series S7-1200, S7-1500, S7-1500 Software Controller, ET 200SP, CPU 1516(F)pro-2 PN, SINAMICS drives, SIRIUS SIMOCODE pro and SIRIUS Soft Starter 3RW as of TIA Portal V15.

Conventions

This documentation contains pictures of the devices described. The pictures may differ slightly from the devices supplied.

Please also observe notes marked as follows:

Note

A note contains important information on the product described in the documentation, on the handling of the product and on the section of the documentation to which particular attention should be paid.

"mySupport"

With "mySupport", your personal working area, you make the most of your Industry Online Support.

In "mySupport" you can store filters, favorites and tags, request CAx data and put together your personal library in the Documentation area. Furthermore, your data is automatically filled into support requests and you always have an overview of your current requests.

You need to register once to use the full functionality of "mySupport".

You can find "mySupport" on the Internet (https://support.industry.siemens.com/My/ww/en/).

"mySupport" - Documentation

In the Documentation area of "mySupport", you have the possibility to combine complete manuals or parts of them to make your own manual.
You can export the manual in PDF format or in an editable format.

You can find "mySupport" - Documentation on the Internet (http://support.industry.siemens.com/My/ww/en/documentation).

Further support

- The range of technical documentation for the individual SIMATIC products and automation systems can be found on the Internet (http://www.siemens.com/simatic-tech-doku-portal).
- The online catalog and the online ordering system is available on the Internet (https://mall.industry.siemens.com).

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under (https://www.siemens.com/industrialsecurity).

6.2 Description

6.2.1 Supported hardware

If a device supports the trace and logic analyzer function, "Traces" is offered for selection in the project tree below the device.

The following devices (Page 109) support the trace and logic analyzer function:

- SIMATIC S7-1200 CPUs (as of firmware version V4.0)
- SIMATIC S7-1500, ET 200SP and CPU 1516pro-2 PN CPUs
- SIMATIC S7-1500 Software Controller
- SINAMICS drives that are supported in Startdrive
- SIRIUS SIMOCODE pro (with Simocode ES)
- SIRIUS Soft Starter 3RW (with Soft Starter ES)

6.2.2 Recording of measured values with the trace function

Introduction

The trace and logic analyzer function can be called in the project navigator (Page 63) by double-clicking an entry in the "Traces" system folder. The measurements on the memory card can also be read and displayed via the diagnostic interface of the Web server.

You record device tags and evaluate the recordings with the trace and logic analyzer function. Tags are, for example, drive parameters or system and user tags of a CPU. The number of installed traces is hardware-dependent.

The recordings are saved on the device and, when required, can be read out with the engineering system (ES) and saved permanently. The trace and logic analyzer function is therefore suitable for monitoring highly dynamic processes. The recorded values are overwritten when the recording is activated again.

The trace and logic analyzer functions are also used in the commissioning editors of technology objects (for example, axle control panels). Active recordings from the axis control panel are displayed in the "Traces" system folder as installed traces. Recordings can be added to the measurements in the curve diagram of the axis control panel or the PID via a shortcut menu command.

Depending on the device (Page 109) used, the recording options can vary.

A quick start (Page 89) for working with the trace and logic analyzer function can be found in the Operation section.

The following figure shows the method of operation of the "Traces":

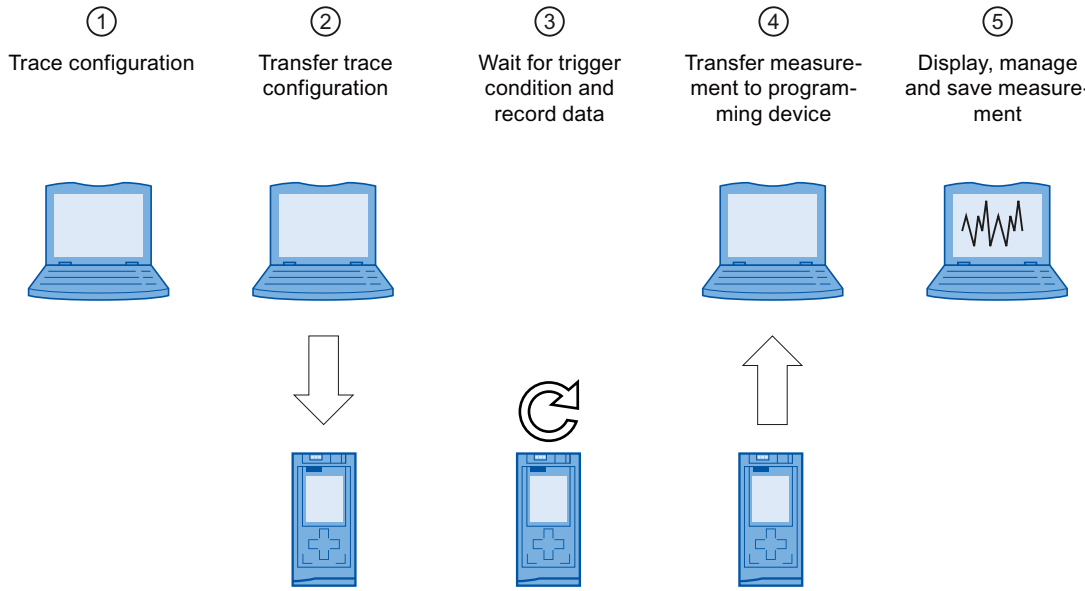

① Trace configuration on the programming device (PG) in the TIA Portal

You can specify the signals to be recorded, the duration of the recording and the trigger condition in the trace configuration. The trace configuration depends on the device and is described at the respective device (Page 109).

② Transferring the trace configuration from the PG to the device

You can transfer the complete trace configuration (Page 96) to the device when an online connection is established.

③ Waiting for the recording

If the installed trace configuration is activated (Page 97), then the recording is performed independently of the PG. The recording is started as soon as the trigger condition is satisfied.

④ Transferring the measurement from the device to the PG

The saving of the measurement in the project (Page 99) stores the measurement in the opened project of the TIA Portal. The measurement can be saved at any time after completing the recording, irrespective of the time of the measurement.

⑤ Evaluating, managing and saving the measurement

Numerous options are available for the evaluation of the measurement in the curve diagram and in the signal table (Page 97). Various display types are possible, for example, a bit representation for binary signals.
Signal waves from different measurements can be put together as an overlay measurement and compared with each other.
Measurements can also be exported and imported as a file.
With the saving of the project (Page 99) in the TIA Portal, the measurements transferred to the project are also saved.

Additional features

The following features are supported as of TIA Portal V14:

- Saving of trace configurations and measurements in libraries
- Multiuser support for trace configurations and measurements (device-dependent)
- Global search in the project

6.2.3 Trace configuration, recording, installed trace and measurement

This section explains the meaning and relationships of the terms: trace configuration, recording, installed trace and measurement.

Trace configuration

Implement the following settings in the trace configuration:

- Signals to be recorded
- Recording conditions
  - Sampling
  - Trigger
  - Installed measurements (memory card)

Recording

A recording is performed in the device. There is only one recording for each installed trace configuration. When a new recording is started, the old recording is overwritten.

An installed recording is not retentive (it is lost when the device is switched off/on) but can be saved permanently in the project as a measurement.

Installed trace

An installed trace consists of a trace configuration and optionally a recording. The maximum number of installed traces depends on the device.

The trace configuration is stored retentively on the device. The retentivity of the trace configuration may also be configurable depending on the device, e.g. with the S120.

Measurement

A measurement consists of a trace configuration and a recording, provided that recorded data is present. Each installed trace can be saved as a measurement in the project.

The recording of a measurement can be viewed offline.

The configuration for a measurement can be transferred to the “Traces” folder using drag & drop.

Installed measurements (memory card)

The folder contains measurements that are saved on the device (e.g. on the memory card). These measurements are retentive and can only be deleted by the user.

The installed measurements can be transferred to the Measurements folder using drag & drop and are then saved as measurements in the project.

Trace configuration with an installed trace of the same name

Usually, there is a trace configuration in the project with the same name for an installed trace. When there is an online connection, this trace is displayed with the icon in the project tree.

See also User interface - “Traces” project tree folder (Page 64).

Overlay measurement

The overlay measurement allows a comparison and analysis of signals from different measurements with each other.

The measurements can be synchronized with each other and displayed as overlay measurements.

6.2.4 Data storage

The trace toolbar and the curve diagram also enable the transfer of the trace configuration and the viewing of the recording.

The following figure is a schematic diagram of the data storage:

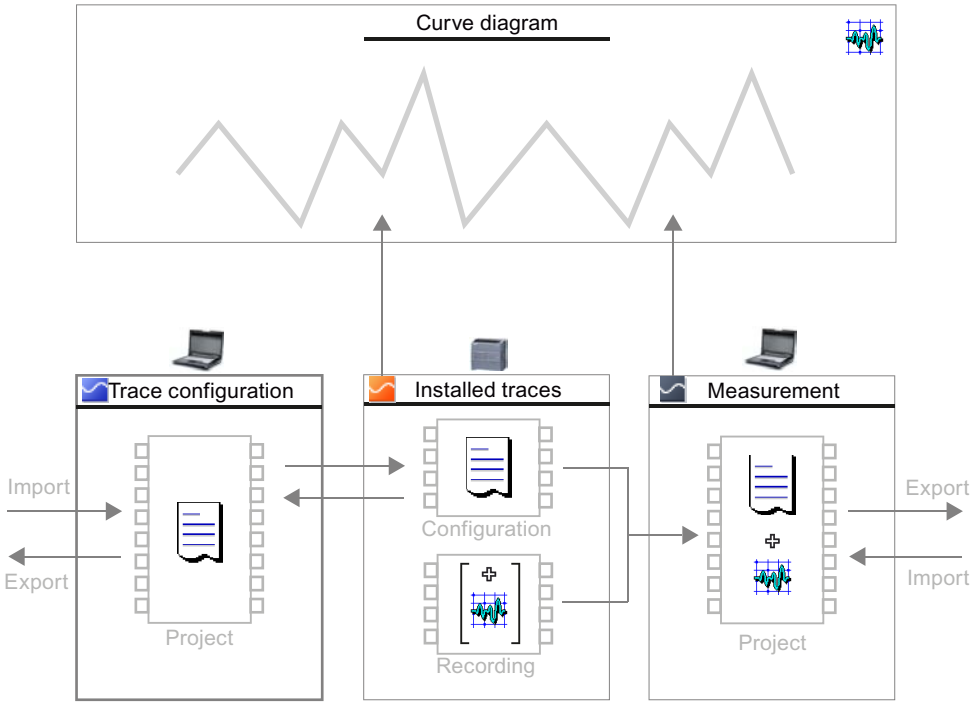

Note

Saving the trace configuration and measurement

You save the trace configuration and measurement with the project in the TIA Portal.

If you close the project without saving, the trace configurations and the measurements transferred to the project are discarded. The trace editor can be closed and reopened without loss of data until the project is closed.

6.3 Software user interface

Display areas

The user interface of the trace and logic analyzer function consists of several areas.

The example in the figure below shows the division of the user interface in the TIA Portal:

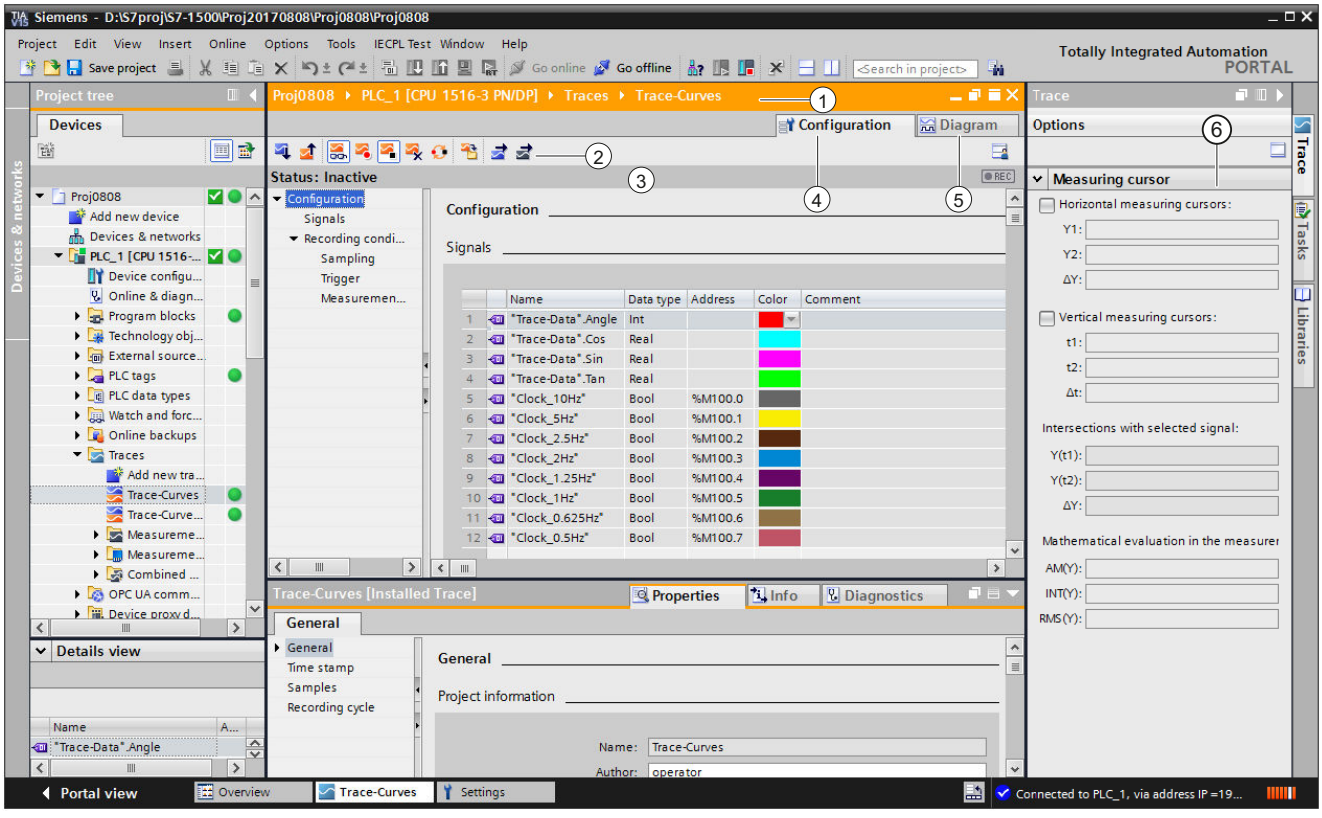

Project navigator Management and creation of the trace and measurements directly in the project tree and via context menu commands.
Working area

	① Title bar of the working area Shows the device to which the current display belongs.
	① Trace toolbar Buttons for managing the trace in the project and device: • Activation/deactivation of installed traces • Deletion of installed traces • Transfer of trace configurations and measurements between the device and the project • Export of trace configurations and measurements • Switchover between offline and online display
	③ Status display of the trace Display of the current status of the recording.
	④ Configuration tab Device-specific configuration of the recording duration, trigger condition and signal selection. See Device-specific descriptions (Page 109).
	⑤ Diagram tab Display of the recorded values as a curve diagram and the signals from the displayed measurement.
	Signal selection tab Display of all signals that are contained in the overlay measurements.
"Trace" task card Display of the measurement cursor data with mathematical evaluation ⑥ and snapshots.	
Inspector window Display of general information about the trace configuration	

6.3.1 Project navigator

6.3.1.1 User interface - “Traces” project tree folder

Trace configurations and installed traces are displayed in the "Traces" folder.

More information about the “Traces” sub-folder is provided in the following sections.

Double-click a trace to open the corresponding "Diagram" or "Configuration" tab in the working area.

Icons in the "Traces" folder

The following table explains the icons in the "Traces" folder:

Icon	Description
	Add trace configuration Double-click the icon to add a new trace configuration.
	Trace configuration (offline) Double-click the icon to open the "Configuration" tab.

Icon	Description
	Installed trace (online) The icon is only displayed when there is no offline trace configuration of the same name for the installed trace. Double-click the icon to open the "Diagram" tab.
	Trace configuration with an installed trace of the same name If the button is deactivated, the trace configuration from the project is displayed. The trace corresponds to a trace configuration. If the button is activated, the trace configuration from the device is displayed. The trace corresponds to an installed trace. Double-click on the symbol to open the "Diagram" tab of the trace in the device.

Status

When there is an online connection, the status is displayed in the right-hand column of the project tree. The status is also displayed as tooltip above the respective icon.

The following table shows the meaning of the icons:

Icon	Description
	Online and offline configuration are identical
	Online and offline configuration are different
	Configuration only exists online

Shortcut menu commands

The following table shows the shortcut menu commands for the "Traces" system folder:

Shortcut menu command	Description
“Add new group”	Inserts a new folder.
“Add new trace”	Inserts a new trace configuration and opens the configuration tab.
"Import trace configuration"	Imports a trace configuration from a file.

The following table shows the shortcut menu commands for trace configurations and installed traces / :

Shortcut menu command	Trace configuration	Installed trace	Description
"Copy"	x	-	Copies the trace configuration or trace to the clipboard.
"Paste"	x	-	Inserts a trace configuration or measurement from the clipboard.
"Delete"	x	x	Deletes the selected objects from the project tree or from the device.

Shortcut menu command	Trace configuration	Installed trace	Description
"Rename"	x	-	Switches the selected object to the editing mode.
"Export trace configuration"	x	-	Export a trace configuration as a file with the file extension "* .ttcfgx" or a trace in the device with the file extension "* .ttrecx". For reasons of compatibility, the "* .ttcfg" and "* .ttrec" file extensions are supported by TIA Portal V12, although they do not contain any information about the device family.

The trace configuration can also be copied across devices within the same device family.

Several objects can be selected.

6.3.1.2 User interface - “Measurements” project tree folder

The “Measurements" folder shows the saved measurements.

Icons in the “Measurements" folder

The following table explains the icons in the “Measurements" folder:

Icon	Description
	Measurement (offline) Double-click the icon to open the "Diagram" tab. The configuration for a measurement can be transferred to the “Traces” folder using drag & drop.

Shortcut menu commands

The following table shows the shortcut menu commands for the "Measurements" system folder:

Shortcut menu command	Description
“Add new group”	Inserts a new folder.
"Import measurement"	Imports a measurement from a file with the "*.ttrecx" file extension. The import is device-independent. For reasons of compatibility, the "*.ttrec" file extension is supported in V12, although it does not contain any information about the device family.

The following table shows the shortcut menu commands for measurements :

Shortcut menu command	Description
"Copy"	Copies the trace configuration of the selected objects to the clipboard.
"Paste"	Inserts a measurement from the clipboard.
"Delete"	Deletes the selected objects from the project tree or from the device.

Shortcut menu command	Description
"Rename"	Switches the selected object to the editing mode.
“Generate new overlay measure-ment”	Generates a new overlay measurement with the selected measurements.
"Export measure-ment"	Exports a measurement with the last saved standard view The measurement is saved with the extension "*.ttrecx" or "*.csv". For reasons of compatibility, the "*.ttrec" file extension is supported in V12, although it does not contain any information about the device family.

The measurements can also be copied independent of the device family.

Several objects can be selected.

6.3.1.3 User interface - “Installed measurements (memory card)” project tree folder

The folder shows all measurements present on the memory card. The folder is only displayed when there is an online connection to the device.

Drag folders or measurements contained here to the “Measurements” system folder using drag & drop. This transfers the measurements to the project.

Icons in the "Traces" folder

The following table explains the icons in the system folder :

Icon	Description
	Folders generated automatically with information on the recording activation time: The name of the folder cannot be changed.
	Installed measurement Double-click the icon to open the "Diagram" tab. The time stamp in the name shows the occurrence of the trigger event.

Shortcut menu commands

The following table shows the shortcut menu commands for the folder :

Shortcut menu command	Description
"Copy"	Copies the selected objects to the clipboard.
"Delete"	Deletes the selected objects from the project tree and from the device.

The following table shows the shortcut menu commands for measurements :

Shortcut menu command	Description
“Open”	Opens the measurement in the “Diagram” tab.
"Copy"	Copies the selected objects to the clipboard.
"Delete"	Deletes the selected objects from the project tree and from the device.

Shortcut menu command	Description
"Export measurement"	Exports a measurement as a file with the extension "*.ttrecx" or "*.csv". For reasons of compatibility, the "*.ttrec" file extension is supported in V12, although it does not contain any information about the device family.
“Properties”	Displays the general properties of the measurement (Page 87).

Several objects can be selected.

6.3.1.4 User interface - “Overlay measurements” project tree folder

The system folder “Overlay measurements" shows the configured overlay measurements.

Icons in the “Overlay measurements" folder

The following table explains the icons in the system folder “Overlay measurements”:

Icon	Description
	Add new overlay measurements Double-click the icon to add a new overlay measurement and open the “Diagram” tab.
	Overlay measurement Double-click the icon to open the "Diagram" tab.

Shortcut menu commands

The following table shows the shortcut menu commands for the system folder “Overlay measurements" or for a group folder contained within this:

Shortcut menu command	Description
“Add new group”	Inserts a new folder.
“Add new overlay measurement”	Inserts a new overlay measurement and opens the “Diagram” tab.
"Import overlay measurement"	Imports an overlay measurement from a file with the file extension "*.ttcbmx"

The following table shows the shortcut menu commands for overlay measurements:

Shortcut menu command	Description
“Open”	Opens the selected overlay measurements in the working area.
"Import measurement"	Imports a measurement from a file with the file extension "*.ttrecx" For reasons of compatibility, the "*.ttrec" file extension is supported in V12, although it does not contain any information about the device family.
"Export overlay measurement"	Exports an overlay measurement The overlay measurement is saved with the extension "*.ttcbmx" or "*.csv". The "*.ttcbmx" format can also be imported again.
"Copy"	Copies the selected objects to the clipboard.

Shortcut menu command	Description
"Paste"	Pastes measurements, measurements from traces in the device or from an overlay measurement from the clipboard. Multiple objects can be inserted from the clipboard if they are all of the same type.
"Delete"	Deletes the selected objects from the project tree or from the device.
"Rename"	Switches the selected object to the editing mode.
“Properties”	Displays the general properties for the overlay measurements.

The overlay measurements can also be copied device-wide.

Several objects can be selected.

6.3.2 Working area

6.3.2.1 User interface - trace toolbar

Tools are available for handling the trace via buttons.

The following table shows the functions of the buttons:

Icon	Description
	Transfer the selected trace configuration to the device The selected trace configuration is transferred to the device.
	Transfer the selected trace configuration from the device The selected trace configuration is transferred from the device to the project.
	Observe on/off Change of the display between online and offline. **Note** Once monitor and automatic scaling are activated at the same time, no more actions can be undone using the “Undo” button. **Note** When an installed trace is first started the display in the curve diagram is set to automatic scaling by default. Make sure when the recording is restarted that any changes to the scaling settings are retained. Reactivate automatic scaling manually if necessary in order to monitor the recording.
	Activate recording If the recording of an installed trace is repeated, then the settings relevant for the display (curve diagram and signal table) are also retained for the new recording. **Note** When a recording is restarted, the previously recorded values are lost. To save the recorded values, save the measurement in the project (Page 99) before you activate the recording again.
	Deactivate recording
	Delete installed trace Deletes the selected trace from the device.

Icon	Description
	Automatically repeat recording After a recording, the recording is automatically activated again. The display of the curve is refreshed when the recording is completed. This type of refresh is particularly suitable for monitoring fast signals (Page 106).
	Transfer the selected measurement from the device to the project The measurement is added to the "Measurements" system folder. **Note** In order to be able to save the installed trace data as a measurement, it must first have been displayed once in the curve diagram. The recording data is loaded from the device when displayed. If necessary wait until the display is updated, as only the displayed data is transferred to the project in all cases.
	Export trace configuration Exports a trace configuration as a file with the file extension "*.ttcfgx". For reasons of compatibility, the "*.ttcfg" file extension is supported by V12, although it does not contain information about the device family.
	Generate a trace configuration Generates a new trace configuration from the measurement.
	Export measurement with the settings from the current view Exports a measurement as a file with the file extension "*.ttrecx" or "*.csv". For reasons of compatibility, the "*.ttrec" file extension is supported in V12, although it does not contain any information about the device family.
	Import measurement (only with overlay measurements) Imports a measurement from a file with the file extension "*.ttrecx". For reasons of compatibility, the "*.ttrec" file extension is supported in V12, although it does not contain any information about the device family.
	Export overlay measurement (only with overlay measurements) The overlay measurement is saved with the extension "*.ttcbmx" or "*.csv". The "*.ttcbmx" format can also be imported again.
	Select a measurement (only with overlay measurements) The drop down list box contains the imported measurements. Select the desired measurement to display the configuration.

6.3.2.2 User interface - Configuration tab

User interface - Configuration

The trace configuration depends on the device and is described at the respective device (Page 109).

6.3.2.3 User interface - Diagram tab

User interface - curve diagram

The curve diagram displays the selected signals of a recording. Analog signals are displayed in the upper curve diagram. Binary signals are displayed as bit track in the lower diagram. You can adjust the display of the signals in the signal table (Page 76) and with the toolbar of the curve diagram.

Setting options and displays in the curve diagram

The following figure shows an example of the display in the TIA Portal:

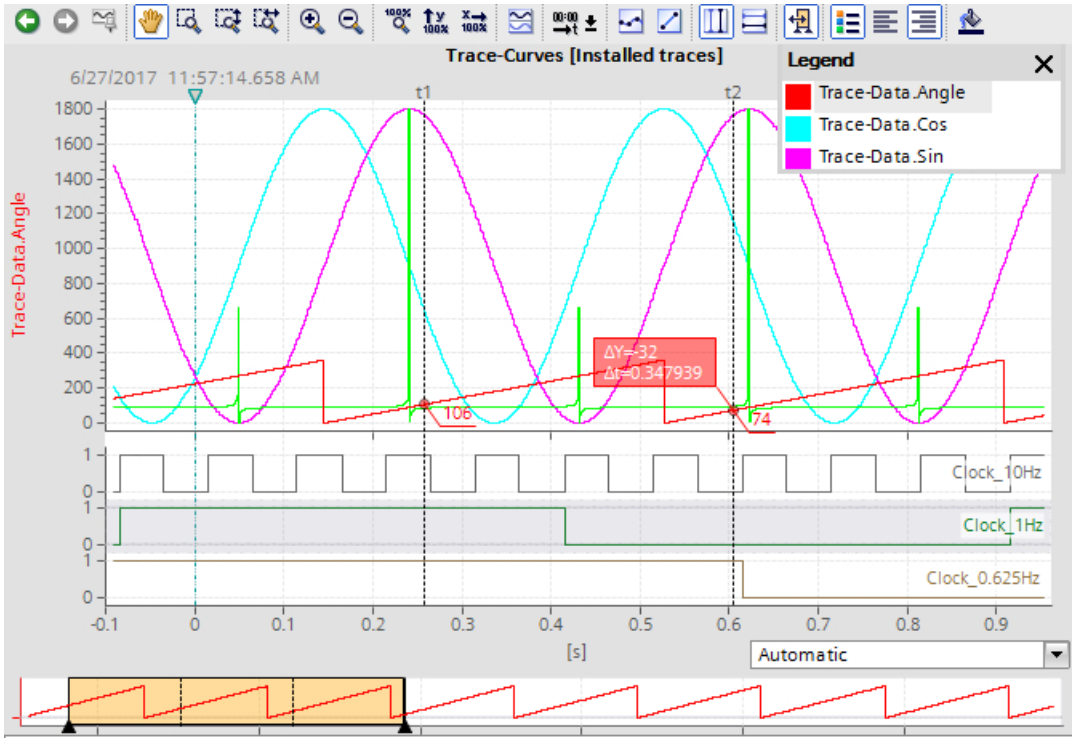

The scale in the diagram applies to the selected (highlighted in gray) signal in the legend. The legend can be moved and its size can be adjusted with the mouse.

The ▽ icon shows the device trigger time with a vertical line.

A drop-down list for selecting the unit is available below the curve for the "Time" setting for the time axis. The "Automatic" setting automatically adjusts the unit based on the displayed time range.

Note

Non-interpretable data types

Some data types require a defined format, e.g. the S7 data type `LTime_of_Day`. If this format is not available, the data type is interpreted as INT.

Functions using the mouse wheel

The following table shows which functions are possible in the curve diagram using the mouse wheel:

Function of the mouse wheel	Description
Move the curve diagram vertically	Turning the mouse wheel moves the display in the upper curve diagram up or down. If the signals are arranged in traces the display for the group is shifted that is located below the cursor. The mouse pointer must be positioned above the curve with the analog signals.
Move the curve diagram horizontally	Turning the mouse wheel with the <Shift> button pressed down moves the display in the curve diagram to the left or the right. The cursor must be positioned above the curve diagram.
Zoom in and zoom out	Turning the mouse wheel with the <Ctrl> button pressed down zooms in or out of the display in the curve diagram. The cursor position is the starting point for zooming in or out. The value axis of the lower curve diagram (bit tracks) is not affected. The cursor must be positioned above the curve diagram.

Functions using the keyboard

The following table shows which keyboard commands are possible with a focus on the curve diagram:

Shortcut key	Description
Selecting a measurement cursor	
<Ctrl+Shift+1>	The vertical measurement cursor t1 is selected or deselected.
<Ctrl+Shift+2>	The vertical measurement cursor t2 is selected or deselected.
<Ctrl+Shift+3>	The horizontal measurement cursor Y1 is selected or deselected.
<Ctrl+Shift+4>	The horizontal measurement cursor Y2 is selected or deselected.
<Tab>	The next measurement cursor is selected.
Positioning a vertical measurement cursor	
<Left>, <Right>	With the unit "Samples", the selected measurement cursor is moved by one sample by the signal in the foreground. With the unit "Time", it is moved by one pixel.
<Shift+Left>, <Shift+Right>	With the unit "Samples", the selected measurement cursor is moved by 10 samples by the signal in the foreground. With the unit "Time", it is moved by 10 pixels.
Positioning a horizontal measurement cursor	
<Up>, <Down>	The selected measurement cursor is moved by one pixel along the value axis.
<Shift+Up>, <Shift+Down>	The selected measurement cursor is moved by 10 pixels along the value axis.
Display of vertical measurement cursors	
<Ctrl+Space>	The vertical measurement cursors are shown or hidden.

Shortcut key	Description
<Ctrl+Shift+Space>	The vertical measurement cursors are shown and centered for the current view.
Changing the view	
<Space>	Move view
<Ctrl+0>	Set 100% view in open editor
<Ctrl++>	Apply zoom in with 10%
<Ctrl+->	Apply zoom out with 10%

Shortcut menu commands

The following table shows the shortcut menu commands in the curve diagram:

Shortcut menu command	Description
"Save diagram as image"	Exports the current display in graphic format, e.g. as a bitmap.
"Copy image to clipboard"	Copies the current display to the clipboard.
"Center measurement cursors"	Positions the activated measurement cursors at a central point in the current display.
"Add to measurements" (only axis control panel and PID)	Adds the displayed recording to the "Measurements" system folder.
“Automatic bit track height"	Automatically adjusts the height of the bit tracks and thereby determines the size of the lower curve diagram. The setting is automatically deactivated once you change the space allocation between the curve diagrams manually. **Note** You can change the vertical space allocation between the upper and lower curve diagram. To do this drag the time axis of the upper curve diagram up or down with the mouse.

Toolbar of the curve diagram

Tools are available for adapting the display via buttons.

The following table shows the functions of the buttons:

Icon	Function	Description
	Undo zoom	Undoes the zoom function executed last. If several zoom functions have been executed, they can be undone step-by-step.
	Redo zoom	Redoes the last undone zoom function. If several zoom functions have been undone, they can be redone step-by-step.
	Standard view	Uses the current view as standard for this recording. If the trace recording is shown again later, the standard view is restored.
	Move view	Moves the display with the mouse button pressed.
	Zoom selection	Selection of an arbitrary range with the mouse button pressed. The display is scaled to the range selection.

Icon	Function	Description
	Vertical zoom selection	Selection of a vertical range with the mouse button pressed. The display is scaled to the range selection.
	Horizontal zoom selection	Selection of a horizontal range with the mouse button pressed. The display is scaled to the range selection.
	Zoom in	Enlargement of the display. The ranges of the time axis and value axis are reduced every time the button is clicked. The curves are displayed larger.
	Zoom out	Reduction of the display. The ranges of the time axis and value axis are increased every time the button is clicked. The curves are displayed smaller.
	Display all	Scales the display of the available data so that the entire time range and all values are displayed.
	Automatic scaling of the value axis	Scaling of the display so that all values are displayed for the currently displayed time range. The relative scaling ratio between the signals may change. **Note** The automatic scaling of the value axis is stopped when the zoom function is activated for the value axis. This button reactivates the automatic adjustments to the minimum/maximum values.
	Show the overall time range	Scaling of the display so that the values in the value range currently displayed are displayed for the overall time range. The value range displayed only then changes if “Display all values” is activated. **Note** The automatic scaling of the time axis is stopped when a zoom function is activated for the time axis. This button reactivates the automatic adjustments for the time range.
	Arrange in tracks	Activate or deactivate the trace arrangement. When the trace arrangement is activated the signals are arranged among themselves with the relevant value axes. Signal groups are displayed in the same trace. This setting does not affect the display for the bit tracks.
	Unit changeover of the time axis	Switching the unit of the time axis The following units are adjustable: • Measurement points • Time (relative time related to the trigger time) • Trigger stamp of the measurement points
	Display samples	The samples are displayed as small circles on the curves.
	Interpolated representation	Linear interpolation between two consecutive measuring points for floating point numbers If linear interpolation is not enabled (default), the connection between the measuring points is drawn in steps.

Icon	Function	Description
	Display vertical measurement cursors	Display of the vertical measurement cursors. The vertical position of the two measurement cursors can be moved with the mouse. The associated measured values and the difference of the measurement cursors corresponding to the position are shown in the signal table. Display the “Measurement cursor” pane (Page 84) in the Trace task card in order to display further information. Also use the cursor keys. The following actions are possible for vertical measurement cursors with the cursor keys: • Select • Positioning • Show or hide measurement cursor • Center measurement cursors
	Display horizontal measurement cursors	Display of the horizontal measurement cursors. The horizontal position of the two measurement cursors can be moved with the mouse. Display the “Measurement cursor” pane (Page 84) in the Trace task card in order to display the values or to reposition the measurement cursor through entering the position. Also use the cursor keys. The following actions are possible for horizontal measurement cursors with the cursor keys: • Select • Positioning
	Time range display	Show or hide the time range display below the curve The time range display shows the area in the curve in yellow based on a selected signal. The yellow area can be moved with the mouse and the borders can be changed horizontally. 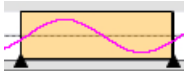
	Display chart legend	Showing or hiding of the legend in the curve diagram and the bit track labels.
	Align the chart legend to the left	Display of the legend and the bit track labels on the left side of the curve diagram.
	Align the chart legend to the right	Display of the legend and the bit track labels on the right side of the curve diagram.
	Change background color	Changeover between various background colors.

User interface - signal table

The signal table lists the signals of the selected measurement and provides setting options for some properties. If recording data of installed traces is displayed and the settings are changed in the signal table, these settings are retained until there is a change to the offline mode. If the installed trace is added to the measurements, the current settings of the signal table are saved in the measurement.

The signals can be sorted using drag-and-drop. The bits of a signal can be resorted within a signal.

Setting options and displays in the signal table

The following figure shows an example of the display in the TIA Portal:

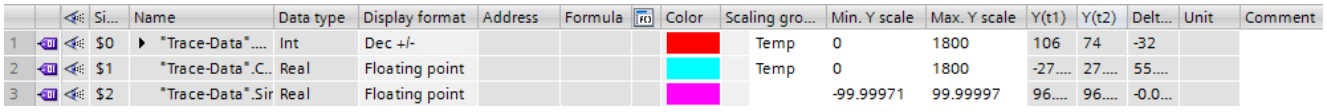

		Si...	Name	Data type	Display format	Address	Formula		Color	Scaling gro...	Min. Y scale	Max. Y scale	Y(t1)	Y(t2)	Delt...	Unit	Comment
1		$0	"Trace-Data"....	Int	Dec +/-					Temp	0	1800	106	74	-32		
2		$1	"Trace-Data".C..	Real	Floating point					Temp	0	1800	-27....	27....	55....		
3		$2	"Trace-Data".Sir	Real	Floating point						-99.99971	99.99997	96....	96....	-0.0...		

The following table shows the settings and displays of the recorded signals:

Column	Description
Signal or error symbol	
	Signal
	Failsafe signal
	Signal from a data block
	Signal from a failsafe data block
	Calculated signal (formula)
	Error in the formula of the calculated signal
	Selection for the display in the curve diagram - a maximum of 16 signals can be selected. The point indicates that at least one bit has been selected for display as bit track for the signal in the bit selection.
"Signal reference"	Automatically generated number of the signal The signal can be accessed via the signal reference in the formulas.
"Name"	Display of the signal name A click on the name of a displayed signal updates the scale in the curve diagram. You can enter a name for a calculated signal in the last line without a signal symbol. The calculated signal is entered with its name.
“Measurement”	Display of the measurement (in the “Diagram” tab only with overlay measurements) Shows the name of the measurement to which the signal belongs.

Column		Description
		Open bit selection Individual bits can also be selected for the following data types for display as a bit track in the lower curve diagram. • Byte, Word, DWord, LWord • SInt, USInt, Int, UInt, DInt, UDInt, LInt, ULInt Example of an opened bit selection for the DWORD data type: $0 "Tag_1" Byte Dez %MB10 $1 "Tag_2" DWord Dez %MD20 $1.x0 Bool Bin %M23.0 $1.x1 Bool Bin %M23.1 $1.x2 Bool Bin %M23.2 Select or deselect the relevant bit for display by clicking the icon.
"Data type"		Display of the data type
"Display format"		Display format of the signal The display formats supported for the signal are offered for selection. A display format suitable for the data type is set with "Default".
"Address"		Display of the address of the signal The field remains empty with optimized / type correct tags.
"Formula"		Display or entry of a formula A formula can contain mathematical functions with numbers and signals. Use the formula editor to conveniently create formulas.
		Call of the formula editor for calculated signals Click on the icon to open the formula editor.
"Color"		Display and setting option for the color of the signal
"Signal group"		Display or input of the signal group name for one signal group The Y-scales are scaled identically for all signals of one signal group. Enter an identical signal group name for those signals that are to be scaled identically. Signals can be removed from the signal group by deleting the signal group name. The signal groups can be saved via the function "Use current view as standard" (button). **Notes** Binary signals cannot be grouped. In hex display format, group only the signals with a format compatible to the sign for the display.
	Gray field for chain icon	Move the cursor over the gray field or the chain icon (or) to add the signal to a signal group or delete the signal from the signal group. Clicking the chain icon adds the signal to a signal group or creates a new signal group. Clicking the chain icon removes the signal from the signal group. For a selected signal with signal group, the chain icon displays all signals of the same signal group.
	Input field	The input field displays the signal group name. As an alternative to the chain icon, you can assign or delete a group name via text input in this field.
"Min. Y-scale"		Display or input of the minimum value for the scaling of the signal

Column	Description
"Max. Y-scale"	Display or input of the maximum value for the scaling of the signal
"Y(t1)"	Display of the value at the position of the first measurement cursor
"Y(t2)"	Display of the value at the position of the second measurement cursor
"ΔY"	Display of the value difference between the first and the second measurement cursor
"Unit"	Display of the unit (e.g. for used-based values from technology objects)
"Comment"	Display and input option for a comment about the signal

Shortcut menu commands

The following table shows the shortcut menu commands of the signal table:

Shortcut menu command	Description
"Insert calculated signal"	Inserts a new calculated signal at the top of the table
"Edit formula"	Opens the formula editor for the calculated signal
"Cut"	Cannot be selected.
"Copy"	Copies the contents of the selected lines to the clipboard.
"Paste"	Cannot be selected.
"Delete"	Cannot be selected.
"Rename"	Cannot be selected
"Display format"	Allows you to switch the display format The display formats supported for the signal are offered for selection.
“Display signal(s)”	Displays the selected signals in the curve diagram.
“Hide signal(s)”	Hides the selected signals in the curve diagram.

See also

Use of the signal table (Page 103)

Using the signal group in the signal table (Page 104)

Interface - Formula editor

The formula editor provides various mathematical functions for analyzing signals. Open the editor in the signal table by clicking the button.

Configuration options and displays in the formula editor

The following figure shows an example of the display in the TIA Portal:

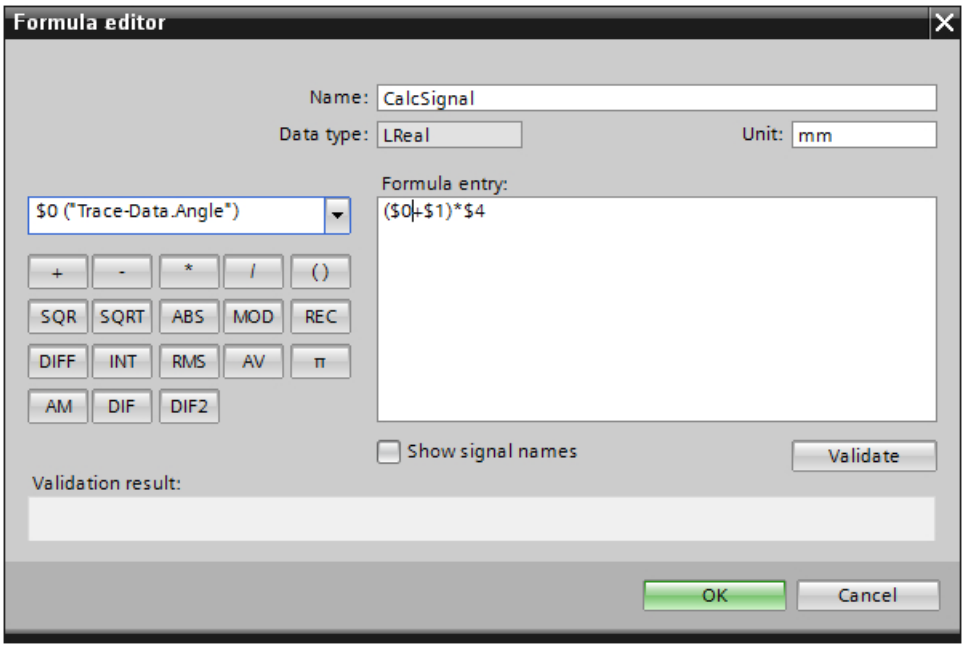

Figure 6-1 Formula editor

The following table shows the configuration options and displays of the formula editor:

Field/Button		Description
"Name"		Display and input of the name for the created formula The name must be unique and only contain characters that are allowed in Windows file names.
"Data type"		Display of formula data type The data type is pre-assigned with a floating-point number of LREAL type and cannot be changed.
"Unit"		Display and input of a unit Freely specified user-defined unit.
Drop-down list with signals		Selection of the signals The drop-down list contains the signals from the signal table and inserts a selected signal into the formula.
"Formula entry"		Text field to display and enter the formula Create a formula by typing into this text box or by using the buttons for the mathematical functions. Signals can be referenced in the text box using the signal reference with a prefixed $ character or the name in double quotes in the formula. Mixed input is possible. Bits from a bit selection (e.g. below the INT data type) are not allowed in the formula.
Mathematical functions		
	+	Addition
	-	Subtraction

Field/Button		Description
	*	Multiplication
	/	Division
	()	Brackets Grouping expressions
	SQR	Square
	SQRT	Square root
	ABS	Absolute value Calculates the size of a number. **Examples** `ABS(5)` → 5 `ABS(-3)` → 3 `ABS(-3.14)` → 3.14
	MOD	Modulo Calculates the residual value of a division **Examples** `MOD(5,3)` → 2 `MOD(3.14,3)` → 0.14
	REC	Reciprocal value (1/x)
	DIFF [1)]	Numerical differentiation **Examples** Formula: `DIFF($0,SAMPLETIME)`
	INT [1)]	Numerical integration **Examples** Formula: `INT($0,SAMPLETIME)`
	RMS [1)]	Quadratic mean The quadratic mean is given by first adding the squares of all the measured values and dividing them by the number of measured values. The quadratic mean is the square root of this value. **Examples** Formula: `RMS($0,SAMPLETIME)`
	AV	Mean value filter from 1st to 5th order If the specification of an order is missing, the mean filter of the 1st order is used. **Examples** `AV($0,1)` → Mean filter 1st order `AV($0,5)` → Mean filter 5th order
	π	Mathematical constant Pi
	AM	Arithmetic mean The arithmetic mean is a moving average over five measuring points.

Field/Button		Description
	DIF	Simple subtraction with mean filter from 1st to 5th order If the specification of an order is missing, simple subtraction is performed with a 1st order filter. **Examples** `DIF($0,1)` → Single subtraction with 1st order filter `DIF($0,5)` → Single subtraction with 5th order filter `DIF($0)` → Single subtraction with 1st order filter **Example: Calculate an acceleration curve from a velocity signal** `$0`: Velocity signal in meters per second Cycle time of the constant cycle velocity recording: 1 ms Formula: `DIF($0,1)/0.001` Unit: m/s^2
	DIF2	Double subtraction with mean filter from 1st to 5th order If the specification of an order is missing, then double subtraction is executed with a 1st order filter. **Examples** `DIF2($0,1)` → Double subtraction with 1st order filter `DIF2($0,5)` → Double subtraction with 5th order filter `DIF2($0)` → Double subtraction with 1st order filter **Example: Calculate an acceleration curve from a position sequence** `$0`: Position sequence in meters Cycle time of the constant cycle position recording: 1 ms Formula: `DIF2($0,1)/SQR(0.001)` Unit: m/s^2
"Show signal name"		Display of the signal names If the check box is selected, the signal names in the formula are displayed instead of the signal references.
"Validate"		Check the validity of the formula
"Result of validation"		Result of validation Displays the result of the validation and indicates errors and error locations.
"OK"		Transfer the entries in the formula editor
"Cancel"		Discard the entries in the formula editor

[1] The constant `SAMPLETIME` is only available for equidistant recording cycles. Time unit for `SAMPLETIME` is always µs.

User interface - Measurements (overlay measurements)

The Measurements tab displays the individual measurements and among other things provides the setting options for synchronization.

Setting options and displays in the Measurements tab

The following figure shows an example of the display in the TIA Portal:

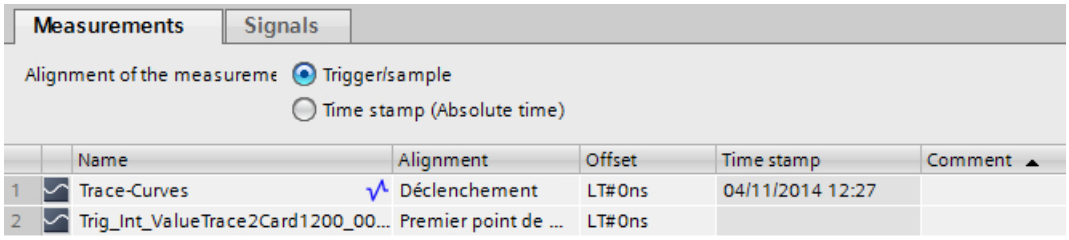

The following table shows the settings and displays for the measurements:

Column		Description
Alignment of the measurements		
	“Trigger/ measure-ment point"	Alignment of the measurements in accordance with the trigger or measurement point The individual zero point for the measurement is predefined in the table under the “Alignment” column.
	“Time stamp (absolute time)”	Alignment of the measurements in accordance with their time stamp The signals are aligned in accordance with the time from the absolute time stamp.
Table columns		
		Static display of the measurement icon
	"Name"	Display and change options for the measurement name The name must be a unique one and can be changed.
	“Alignment”	Alignment of the measurement (only adjustable with the “Trigger/measurement point” check box activated) Determines the individual zero point for a measurement. All signals for the measurement are displayed in relation to this zero point. The following settings are possible: • Trigger • First measurement point following the trigger event • First measurement point • Last measurement point
	"Offset"	Offset related to the time axis Moves the measurement left or right by the offset stated on the time axis. The offset can also be transferred via the clipboard to the cell from the ΔX value of the measurement cursor. See Align measurements precisely (overlay measurements) (Page 108).
	“Time stamp"	Display of the trigger time
	"Comment"	Display and input option for a comment about the signal

Shortcut menu commands

The following table shows the shortcut menu commands of the signal table:

Shortcut menu command	Description
"Cut"	Cannot be selected.
"Copy"	Copies the contents of the selected lines to the clipboard.
"Paste"	Cannot be selected.
"Delete"	Cannot be selected.
"Rename"	Switches the selected cell to the editing mode.
"Import measurement"	Imports a measurement from a file, e.g. with the "*.ttrecx" file extension. The import is device-independent and also suitable, for example, for comparing measurements of a PLC with the measurements of a drive device. For reasons of compatibility, the "*.ttrec" file extension is supported in V12, although it does not contain any information about the device family.
"Export measurement"	Exports a measurement as a file with the extension "*.ttrecx" or "*.csv". For reasons of compatibility, the "*.ttrec" file extension is supported in V12, although it does not contain any information about the device family.

6.3.2.4 User interface - Signal selection tab (overlay measurements)

User interface - Signal selection (overlay measurements)

The Signal selection tab shows the signals for all measurements and allows signals that are presented in the signal table of the diagram to be preselected.

Setting options and displays in the Signal selection tab.

The following figure shows an example of the display in the TIA Portal:

		Availa...	Measurement	Name	Data type	Address	Comment
1		☑	Trace-Curves	"Trace-Data".Angle	Int		
2		☑	Trace-Curves	"Trace-Data".Cos	Real		
3		☑	Trace-Curves	"Trace-Data".Sin	Real		
4		☐	Trace-Curves	"Trace-Data".Tan	Real		
5		☑	Trace-Curves	"Clock_10Hz"	Bool	%M100.0	
6		☑	Trace-Curves	"Clock_5Hz"	Bool	%M100.1	

The following table shows the settings and displays for the table:

Column	Description
	Static display of the signal icon
"Available in the diagram"	Selection for the display in the curve diagram When the selection is activated the signal is transferred to the signal table for the curve diagram.
“Measurement”	Display of the measurement to which the signal belongs
“Name”	Display of the signal name
“Data type “	Display of the data type
"Address"	Display of the address (not for symbolic tags)
"Comment"	Display of a comment on the signal

You will find further information on the specific settings in User interface - signal table (Page 76).

Shortcut menu commands

The following table shows the shortcut menu commands for the signal selection:

Shortcut menu command	Description
"Copy"	Copies the content of the selected lines to the clipboard.
“Display selection in the signal table”	The selected signals are displayed in the signal table and are available in the diagram.
“Remove selection from the signal table”	The removed signals are not available in the diagram.

Several objects can be selected.

6.3.3 Trace task card

6.3.3.1 User interface - Measurement cursor pane

The “Measurement cursor” pane shows the position of the measurement cursor in the curve diagram and the values at the intersection points.

Setting options and displays of the "Measurement cursor" pane

The figure below shows the "Measurement cursor" pane:

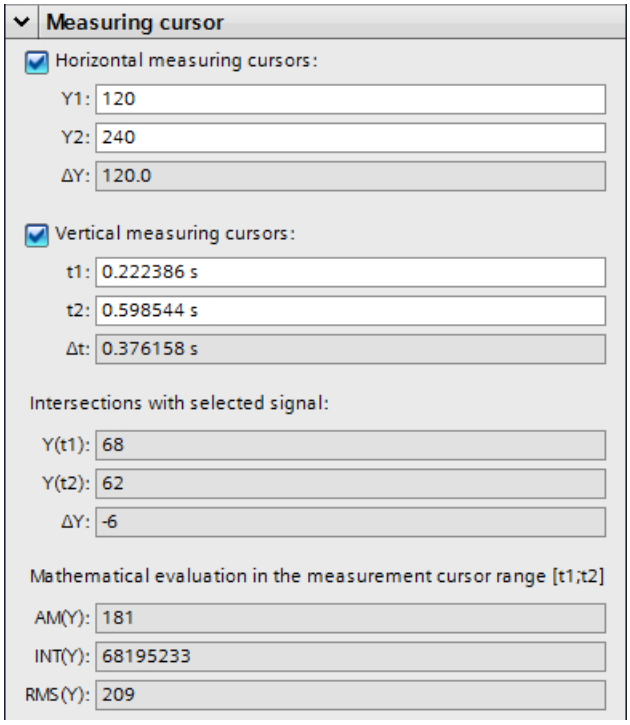

The following table describes the settings and displays:

Setting/display		Description
Horizontal measurement cursor		
	Y1	Position of first measurement cursor The value states the position in relation to the scale of the signal currently selected. You also have the option of specifying a new position for the measurement cursor in this entry field for moving with the mouse.
	Y2	Position of the second measurement cursor The value states the position in relation to the scale of the signal currently selected. You also have the option of specifying a new position for the measurement cursor in this entry field for moving with the mouse.
	ΔY	Display of the position difference between the first and the second measurement cursor
Vertical measurement cursor		

Setting/display		Description
	t1	Position of first measurement cursor You also have the option of specifying a new position for the measurement cursor in this entry field for moving with the mouse.
	t2	Position of the second measurement cursor You also have the option of specifying a new position for the measurement cursor in this entry field for moving with the mouse.
	Δt	Display of the position difference between the first and the second measurement cursor
Intersection points with selected signal		
	Y(t1)	Display of the value at the position of the first measurement cursor
	Y(t2)	Display of the value at the position of the second measurement cursor
	ΔY	Display of the value difference between the first and the second measurement cursor
Mathematical analysis in the range of the measurement cursor·[t1;t2] for the selected signal		
	AM(Y)	Mean The arithmetic mean is calculated for the range between the vertical measurement cursors.
	INT(Y)	Integral The integral is calculated for the range between the vertical measurement cursors.
	RMS(Y)	RMS value The root-mean square (RMS value) is calculated for the range between the vertical measurement cursors.

See also

User interface - curve diagram (Page 71)

6.3.3.2 User interface - Snapshots pane

The “Snapshots” pane allows the user to save and restore different views for a measurement.

A snapshot is taken of the current view in the “Diagram” tab. The snapshots are saved in the measurement with the project.

Setting options and displays of the “Snapshots” pane

The figure below shows the “Snapshots” pane:

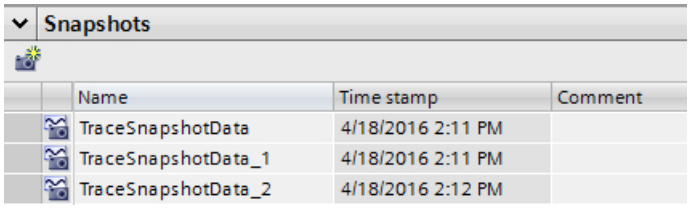

The following table shows the functions of the buttons:

Icon	Description
	Generate snapshot of the current view Saves the current view as a snapshot in the “Diagram” tab.

The following table shows the settings and displays:

Column	Description
	Static display of the snapshot symbol
“Name”	Display and change options for the name
“Time stamp"	Display of the snapshot generation time
"Comment"	Display and input option for a comment

Several rows can be selected and deleted.

Double-clicking on a row opens the measurement with the saved view in the “Diagram” tab.

Shortcut menu commands

The following table shows the shortcut menu commands of the table:

Shortcut menu command	Description
“Restore snapshot”	Shows the measurement with the saved view in the “Diagram” tab.
"Delete"	Deletes the snapshot
"Rename"	Switches the cell to the editing mode

Several rows can be selected and deleted.

6.3.4 Inspector window

6.3.4.1 Interface - Inspector window

The Inspector window displays general information about the trace.

Additional information is available for measurements:

- Time stamp range
  The availability of the time stamps depends on the configured recording conditions.
- Measuring point range
- Cycle time range
  For equidistant cycle recordings, the time duration between two measurement points is displayed.
  This time, for example, can be used in the formula editor.

Sample time stamp

The following figure shows the time stamps for a measurement:

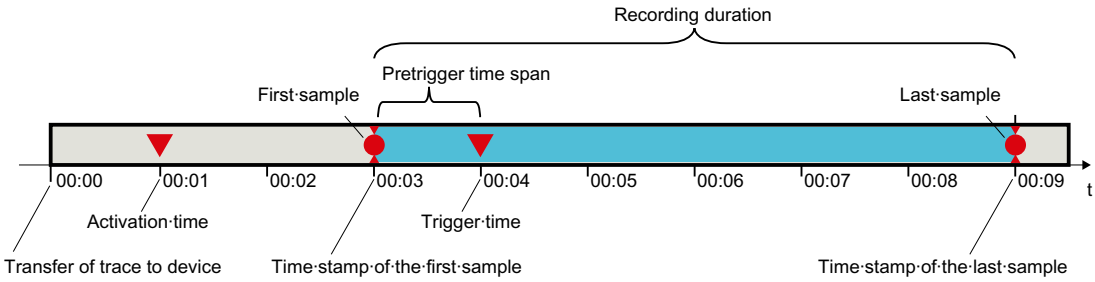

Note

Analysis of measurements with sporadically occurring recording condition

When evaluating your measurements, keep in mind that the recording condition between the activation time and the trigger time may not have been fulfilled.

6.4 Operation

6.4.1 Quick start

This description shows the steps for a recording of the S7-1500 CPU as an example. The displayed settings can differ depending on the device.

Requirement

A device is configured that supports the trace and logic analyzer function.

Creating a trace

The following figure shows the project tree with the "Traces" system folder below the device:

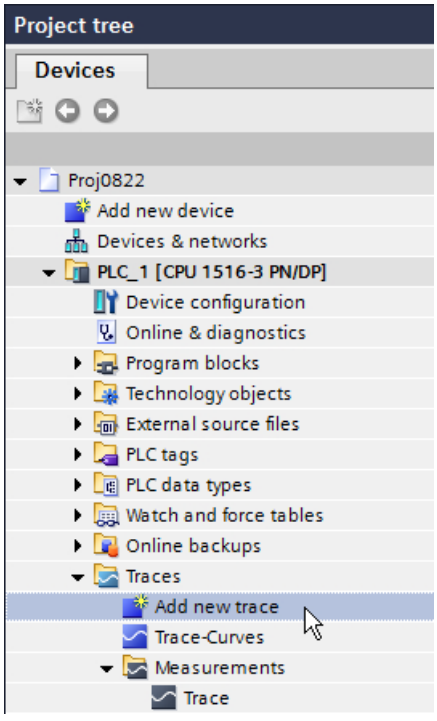

Procedure:

1. Double-click the "Add new trace" entry.
   A new trace configuration is created and the “Configuration" tab opens in the working area.
2. Adapt the name of the trace configuration by clicking the text.

Selecting signals

The following figure shows the configuration of the signals:

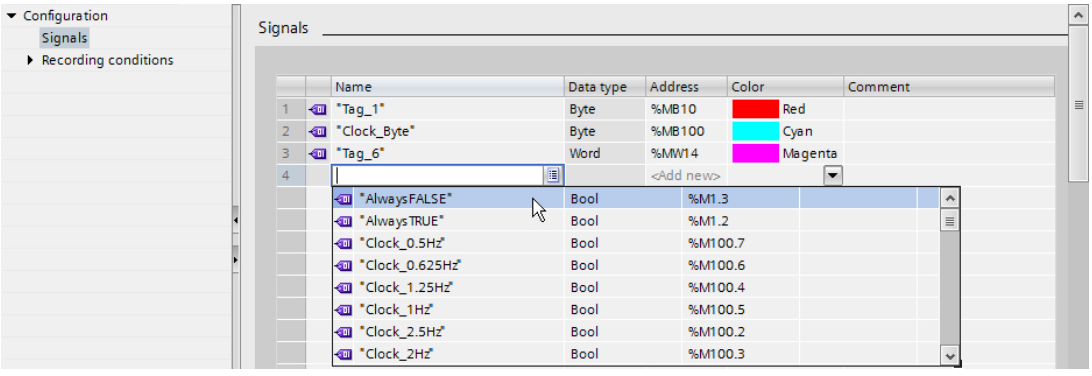

Procedure:

1. Select the signals to be recorded in the "Signals" area.
   Or:
2. Drag one or more signals, e.g. from a tag table, and drop them in the signal table.

Configuring the recording cycle

The following figure shows the configuration of the sampling:

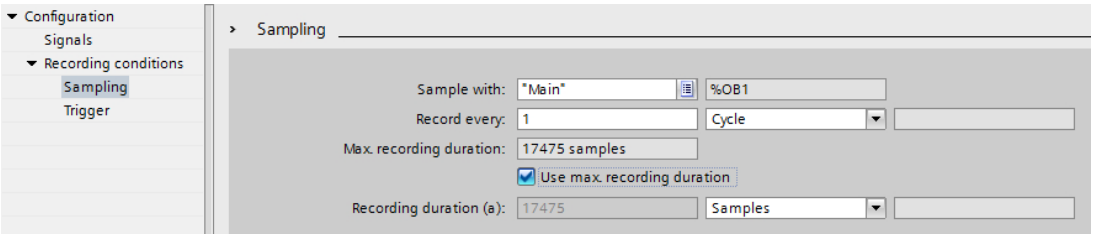

Procedure:

1. Configure the sampling.

Configuring the trigger

The following figure shows the configuration of the trigger:

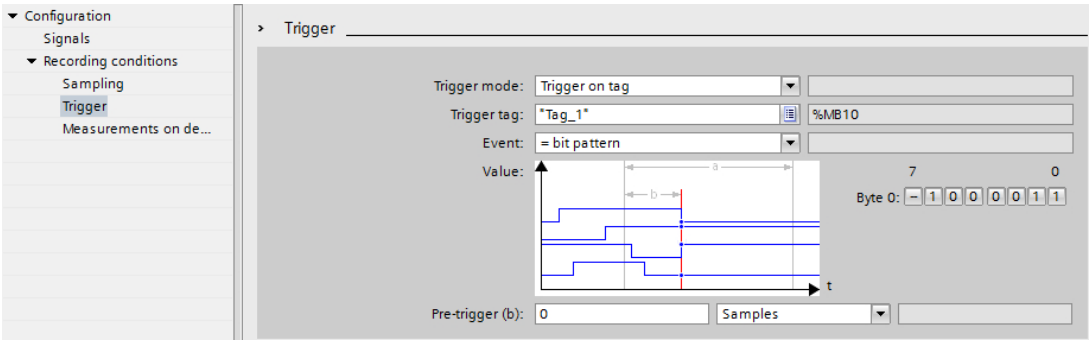

Procedure:

1. Configure the trigger mode and the condition for the selected trigger.

Transferring the trace configuration to the device

Procedure:

1. Transfer the trace configuration to the device with the button.
   The following functions are executed:
   - An online connection is established to the device.
   - The trace configuration is transferred to the device.
   - The monitoring is activated.
   - The display switches to the "Diagram" tab.

Activating a recording

Procedure:

1. Click the button.

Displaying the recording

The following figure shows the curve diagram with a recording:

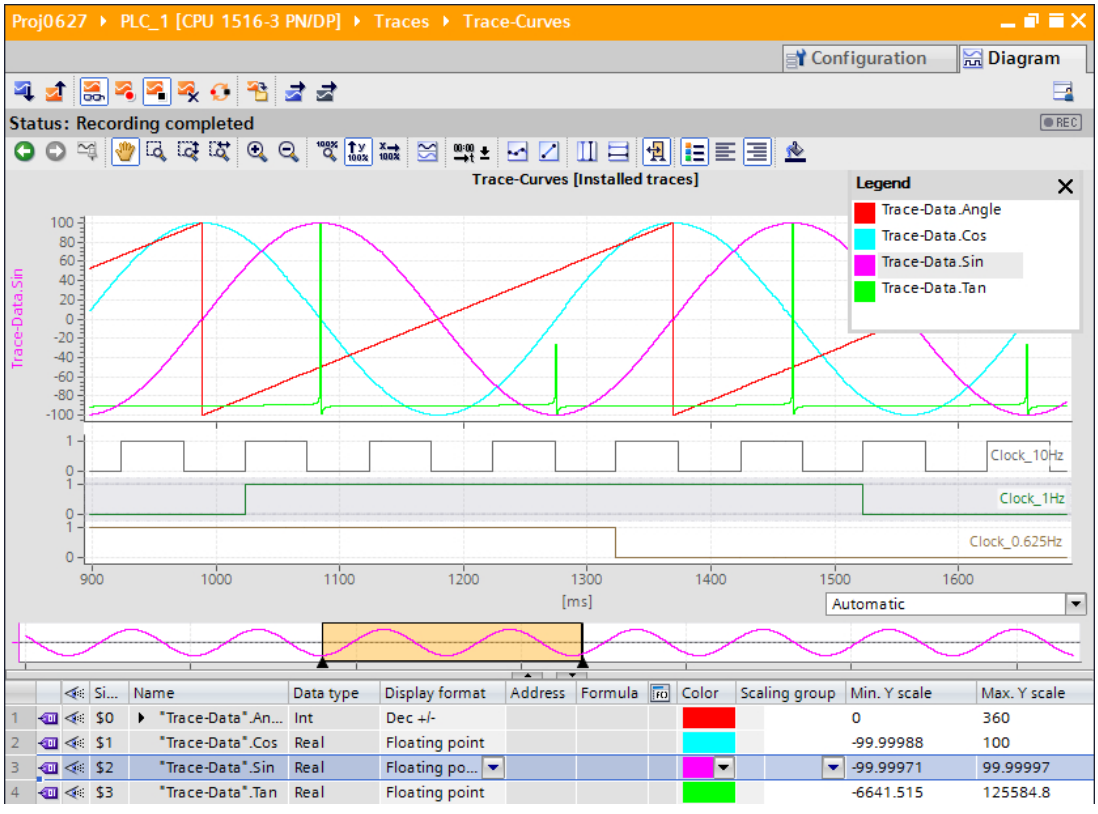

Procedure:

1. Wait until the "Recording" or "Recording completed" status is displayed in the status display of the trace.
2. Switch to the "Diagram" tab.
3. Click the ▸ icon of a signal in the signal table.
   The individual bits of the signal are offered for display as a bit track.
4. In the signal table, select or deselect the individual signals and bits for display with the icon.

Saving the measurement in the project

Procedure:

1. Transfer the measurement to the project with the button.
   The measurement is displayed in the project tree under the "Measurements" system folder.

See also

User interface - trace toolbar (Page 69)

6.4.2 Using the trace function - overview

Requirement

A device is configured in the TIA Portal that supports the trace and logic analyzer function and to which an online connection has been established.

Procedure

The following table shows a procedural overview with typical steps when working with the trace and logic analyzer function.

Step	Description
1	Creating a trace (Page 94)
2	Configuring the trace (Page 101)
3	Transferring the trace configuration to the device (Page 96)
4	Activating/deactivating an installed trace (Page 97)
5	Monitoring the recording (Page 97)
6	Saving measurements in the project (Page 99)
7	Displaying the recording (Page 97)
8	Analyze an ongoing recording (Page 98)
9	Compare records (overlay measurements) (Page 106)

See also

Displaying a configuration (Page 94)

6.4.3 Project tree

6.4.3.1 Creating a trace

Traces can be created in the form of trace configurations in the project navigator.

The following instructions describe how you can create a trace configuration under the "Traces" system folder

Procedure

To create a trace configuration, proceed as follows:

1. Double-click the "Add new trace" entry.

A new trace configuration is created.

6.4.3.2 Displaying a configuration

Requirement

A trace configuration, an installed trace, a measurement or an overlay measurement is available in the "Traces" system folder.

Procedure

To display a trace configuration, proceed as follows:

1. Double-click the appropriate icon of a trace configuration, an installed trace, a measurement or an overlay measurement in the project tree.
   The "Configuration" or "Diagram" tab opens in the working area.
2. If required, click the "Configuration" tab for the display.

Note

Write protection

The configuration data of an installed trace and in all measurements is displayed with write protection.

See also

User interface - “Traces” project tree folder (Page 64)

6.4.3.3 Displaying a diagram

Requirement

An installed trace, a measurement or an overlay measurement is available in the "Traces" system folder.

Procedure

To display a diagram, proceed as follows:

1. Double-click the appropriate icon of an installed trace, a measurement or an overlay measurement in the project tree.
   The "Configuration" or "Diagram" tab opens in the working area.
2. If required, click the "Diagram" tab for the display.

See also

User interface - “Traces” project tree folder (Page 64)

6.4.3.4 Apply overlay measurement

Overlay measurements can be applied in the project tree with a comparison function for different measurements.

The following instructions describe how you can create an overlay measurement under the “Overlay measurements" system folder

Requirement

A device is configured that supports the trace and logic analyzer function.

Procedure

To apply an overlay measurement, proceed as follows:

1. Select one or more measurements in the “Measurements" system folder.
2. Drag the measurements to the “Overlay measurements” system folder.

A new overlay measurement is created. This contains copies of the selected measurements.

6.4.3.5 Configuring objects in groups

You can set up groups in system folders in the project navigator. Use this option to configure the view for multiple objects.

The following instructions use measurements as an example in order to describe how to consolidate the measurements into groups. The same functionality is also available for the Traces and Overlay measurements system folders.

Note

Traces in the device can also be displayed in groups

A trace configuration in a group with a trace of the same name in the device is displayed under the group. Therefore, all traces in the device on the first level in the "Traces" folder are not necessarily displayed.

Requirement

Measurements are available in the “Measurements" system folder.

Procedure

Proceed as follows to configure measurements into groups:

1. Select the shortcut menu command “Add new group” by right-clicking on the “Measurements" system folder.
   A new group folder is created.
2. Assign a meaningful name to the new group.
3. Repeat step 1 until all required groups have been created.
   (Sub-groups (groups within groups) can also be created.)
4. Drag & drop the corresponding measurements to the group folder that has been created.

6.4.4 Working area - general

6.4.4.1 Transferring the trace configuration to the device

Requirement

- A valid trace configuration is in the "Traces" system folder.
- The maximum number of installed traces has not been reached yet.

Procedure

To transfer a trace configuration to the device, proceed as follows:

1. Open a valid trace configuration in the working area.
2. Click the button.

Result

The trace configuration is transferred to the device.

6.4.4.2 Activating/deactivating an installed trace

Requirement

- There is an online connection to the device.
- There is a trace in the device.
- The installed trace is displayed in the working area.
- The button is activated for viewing the displayed trace.

Activating an installed trace

To activate the recording for an installed trace, proceed as follows:

1. Click the button.
   The installed trace is activated and starts the recording according to the configured trigger condition. The trigger condition is device-specific and described in Section "Configuration" below the respective device (Page 109).
   The current status of the recording is displayed in the status display of the trace.

   Note

   When a recording is restarted, the previously recorded values are lost.

   To save the recorded values, save the measurement in the project (Page 99) before you activate the recording again.

Deactivating an installed trace

To deactivate an activated installed trace, proceed as follows:

1. Click the button.
   The installed trace is deactivated.
   The status display of the trace changes to "Inactive".

6.4.4.3 Displaying the recording

Requirement

- There is an online connection to the device.
- There is a trace with recording in the device.

Or:

- A measurement is in the "Measurements" system folder.

Procedure

To display the recording, proceed as follows:

1. Select an installed trace.
2. Double-click the selected trace.
3. If required, activate the button for viewing.

Or:

1. Select a measurement in the "Measurements" system folder.
2. Double-click the selected measurement.

Result

The recording is displayed in the "Diagram" tab.

See also

User interface - “Traces” project tree folder (Page 64)

6.4.4.4 Analyze an ongoing recording

Requirements

- An ongoing recording is displayed in the "Diagram" tab.

Save the data currently recorded as a measurement

To analyze a certain time range for an ongoing recording, follow these steps:

1. Click the button.
   The data recorded up to now is added to the measurements.
   The current recording is not affected by this and continues running uninterrupted.

Analyze the measurement

To display the saved measurement, follow these steps:

1. In the “Measurements" system folder double-click the measurement that has just been saved in order to open it
   The “Diagram” tab for the measurement opens in the working area.

See also

Displaying the recording (Page 97)

Working area - Diagram tab (Page 102)

6.4.4.5 Saving measurements in the project

Requirement

- There is an online connection to the device.
- There is a trace with recording in the device.
- The installed trace data must have been displayed at least once in the curve diagram. The recording data is loaded from the device for the display.

Procedure

To save a recording in the project, proceed as follows:

1. Open the installed trace with the recorded data.
2. If required, make sure that the current data is loaded from the device by activating the button.
3. After activating the button wait until all data has been loaded and displayed.
4. Click the button.
   The measurement is added to the "Measurements" system folder.
5. Save the project in the TIA Portal.

Note

Generate measurements

A measurement of an installed trace can be generated at any time.

Use this functionality e.g. to save the data recorded up until this point in a recording and to analyze it as a static measurement.

6.4.4.6 Exporting and importing measurements

Requirement

At least one measurement is in the "Measurements" system folder for export.

Exporting measurements

To export a measurement, proceed as follows:

1. Right-click a measurement in the "Measurements" system folder and select the shortcut menu command “Export measurement".
2. Select a folder, a file name and a data type to save the measurement.
3. Click the "Save" button.

Importing measurements

To import a measurement, proceed as follows:

1. Right-click in the "Measurements" system folder and select the shortcut menu command "Import measurement".
2. Select the file e.g. of the "*.ttrecx" file type with the measurement to be imported.
3. Click the "Open" button.
   The imported measurement is displayed with the file name in the "Measurements" system folder.

Note

Export and import trace configurations

The same functionality is available for exporting and importing trace configurations.

6.4.4.7 Transferring the trace configuration from the device to the project

Requirement

- There is an online connection to the device.
- There is a trace in the device.

Procedure

To transfer a trace configuration to the project, proceed as follows:

1. Open an installed trace.
2. If required, activate the button for viewing.
3. Click the button to transfer the trace configuration from the device.

Result

The configuration is taken over as new trace configuration in the "Traces" system folder.

A trace configuration of the same name is overwritten in the system folder.

6.4.4.8 Deleting installed traces

Requirement

- There is an online connection to the device.
- There is a trace in the device.

Procedure

To delete an installed trace, proceed as follows:

1. Open an installed trace.
2. If required, activate the button for viewing.
3. Click the button.
   A confirmation prompt opens.
4. Confirm the prompt for deletion.

Or

1. Select one or more installed traces / in the project tree.
2. Press <Delete> to delete the installed traces.
   A confirmation prompt opens.
3. If required, select an option for deletion and confirm the prompt.

6.4.5 Working area - Configuration tab

6.4.5.1 Configuring the trace

Requirement

The "Configuration" tab is open in the working area.

Configuring the trace

In the configuration, you specify the recording and trigger conditions and select the signals to be recorded.

See Section "Configuration" below the respective device (Page 109).

Note

Saving the trace configuration

You save the trace configuration with the project in the TIA Portal.

If you close the project without saving, the configuration is discarded.

See also

Displaying a configuration (Page 94)

6.4.6 Working area - Diagram tab

6.4.6.1 Use of the curve diagram

The curve diagram shows the signals of a recording selected in the signal table.

The display area can be zoomed as required. Measurement cursors can be used to select individual values for display in the signal table.

The following operating instructions describe the use of the curve diagram and of the measurement cursors as examples.

Requirements

- An installed trace or a measurement has been selected for display.
- The button is activated to monitor an installed trace.
- The "Diagram" tab is open in the working area.

Monitor an ongoing recording.

To display all of the data in an ongoing recording, proceed as follows:

1. Activate “Display all" via the button.

The entire time range and all values for the ongoing recording are displayed.

To display a consistent time window in an ongoing recording, proceed as follows:

1. Activate “Display all" via the button.
2. Select the desired time range via the button.

The trend view is updated while the scaling of the time range remains the same.

Evaluation of a certain instant of a recording

To display the values for a specific sample, proceed as follows:

1. Display the vertical measurement cursors via the button.
2. Move a measurement cursor with the mouse to the required position in the recording. The values of the signals are displayed in the signal table and in the “Measurement cursor” pane of the “Trace” task card.

Evaluation of the difference between two samples

To display the difference, proceed as follows:

1. Display the vertical measurement cursors via the button.
2. Move both measurement cursors with the mouse to the required samples in the recording. The values of the signals and the difference are displayed in the signal table and in the “Measurement cursor” pane of the “Trace” task card.

Using horizontal measurement cursors

To check whether a certain value has been reached, proceed as follows:

1. Display the horizontal measurement cursors via the button.
2. Move a measurement cursor with the mouse to the required value of the recording.
   The values of the measurement cursors for the selected signal are displayed in the "Measurement cursor" pane of the "Trace" task card.

Moving the time range displayed

Proceed as follows to move the time range displayed:

1. Select a time range via the button.
2. Move the curve to the desired time range by turning the mouse wheel with the <Shift> key pressed down.

Bringing a signal into the foreground

1. Display the legend via the button.
2. Click a signal in the legend.

Or:

1. Click a signal in the curve diagram.

The signal is displayed in the foreground and is highlighted/selected in the signal table. The value axis is updated for the selected signal.

See also

Displaying a diagram (Page 95)

User interface - curve diagram (Page 71)

User interface - signal table (Page 76)

6.4.6.2 Use of the signal table

The signal table shows the signals of an installed trace or a measurement. The preselected signals in the signal selection are displayed with an overlay measurement. You can show or hide individual signals for the display in the table and adapt the properties for the display. Individual bits can be selected for some data types and displayed as a bit track.

The following operating instructions describe the operation of the signal table.

Requirements

- An installed trace or a measurement has been opened in the "Diagram" tab.
- The button is activated to monitor an installed trace.
- For the display of individual bits as a bit track:
  at least one recorded signal supports the display as a bit track.

Display or hide individual signals and change the color

To adapt the display to suit your requirements, proceed as follows:

1. Click the icon of the respective signal in the column to select or deselect it for the display.
2. Click in the "Color" column for the respective signal and select a color.
   The default color for the signal changes.

Bringing a signal into the foreground

1. In the signal table, select the line of the signal.
   The Y-scale of the signal is displayed.
   The signal curve is brought into the foreground in the curve diagram.

Selecting individual bits for display as a bit track

To display individual bits as a bit track in the lower curve diagram, proceed as follows:

1. Click the icon of a signal in the signal table.
2. Click the icon in the open bit selection of the signal.
   The bits are selected or deselected for display.

See also

Displaying the recording (Page 97)

6.4.6.3 Using the signal group in the signal table

Individual signals can be scaled identically in a signal group, which makes it easier to compare the curve characteristics.

Binary signals cannot be grouped.

The following operating instructions describe how to work with the signal group.

Note

Saving signal groups

The signal groups can be saved individually for each measurement via the "Use current view as standard" function (button).

If the signal groups and the project are not saved then the signal groups created will be lost when the "Diagram" tab is closed.

Requirements

- An installed trace or a measurement is displayed.
- The button is activated to monitor an installed trace.

- The "Diagram" tab is open in the working area.
- There are at least two signals in the signal table that are not of the BOOL type.

Assigning signals to a signal group

To apply a signal group and assign signals to this group, proceed as follows:

1. In the signal table, select the line or cell of the required signal.
2. Click the gray field in the "Signal group" column.
   The sequence icon is displayed in the gray field and the name of the signal group is pre-assigned: Group
3. Click the gray fields of further signals that are to be assigned to this signal group.

Or:

1. Click in the text field of the "Signal group" column for a signal to be grouped.
2. Enter a name for the group.
3. Enter the same group name in the respective text fields for further signals or select the group name via the drop-down list.

The Y-scales of the grouped signals are scaled with the values of the signal that was selected first. Changes to a scale value always affect the entire group.

Removing signals from a signal group

To delete the assignment of a signal to a signal group, proceed as follows:

1. Click the sequence icon for the required signal in the "Signal group" column.

Or:

1. Click the text field for the required signal in the "Signal group" column.
2. Press the <Del> key.

Or:

1. Select the respective text field in the "Signal group" column for several signals using the <Shift> and <Ctrl> keys.
2. Press the <Del> key.

The signals are removed from the signal group or the signal group is deleted.

6.4.6.4 Observation of fast signals

Requirements

- There is an online connection to the device.
- There is a trace in the device.

Procedure

To monitor the progress of a fast signal, proceed as follows:

1. Select a trace in the device.
2. Double-click the selected trace.
3. Click on the button for monitoring.
4. Click on the button to automatically repeat the recording.

Result

The recording is automatically re-activated at the end of each recording. The display in the curve is similar to the display of an oscilloscope.

6.4.6.5 Compare records (overlay measurements)

Requirement

- An overlay measurement is created or is created implicitly by dragging the measurements to the system folder “Overlay measurements”.
  See also Apply overlay measurement (Page 95).

Adding measurements for comparison

To compare measurements, insert the measurements to be compared to the overlay measurements. Proceed as follows for this:

1. In the project tree drag one or more measurements from the system folder “Measurements" to the icon for the overlay measurement .

Or:

1. Import saved measurements via the “Import measurement” shortcut menu command.

A copy of the measurements is added to the overlay measurement.

Note

Changes to the settings for measurements within the overlay measurement have no impact on the original measurements. The original measurements remain unchanged.

Select signals of the measurements for the signal table

Proceed as follows to select the signals for the signal table in the “Diagram” tab:

1. Double-click on the icon for the overlay measurement in the project tree.
   The tabs for the overlay measurement will be displayed in the working area.
2. Click the “Signal selection" tab in the working area.
   The signals for all measurements are displayed in the table.
3. Activate or deactivate the signal check box for those signals that should be visible or should not be visible in the signal table.

The activated signals are displayed in the signal table of the “Diagram” tab.

Use of the signal table

Proceed as follows to open and use the signal tables:

1. Click the “Diagram" tab in the working area.
2. Click on the “Signals” tab within the “Diagram” tab.
3. Use the signal tables as described under Use of the signal table (Page 103).

Align measurements

Proceed as follows to align the time axis for the measurements for the comparison:

1. Click on the “Measurements” tab within the “Diagram” tab.
2. Select the alignment for the measurements via the check box.
3. Adjust the alignment and if necessary set an offset for the alignment of the individual measurements.

The measurements are aligned with each other accordingly on the time axis.

(The precise alignment of two measurements is described in the next section.)

Use of the curve diagram

Proceed as follows to open and use the curve diagram:

1. Click the “Diagram" tab in the working area.
2. Use the curve diagram as described under Use of the curve diagram (Page 102).

See also

Align measurements precisely (overlay measurements) (Page 108)

6.4.6.6 Align measurements precisely (overlay measurements)

Requirement

- An overlay measurement is applied.
- Measurements for comparison are added to the overlay measurement.
- Signals of the measurements for the signal table are selected.
- The “Diagram” tab for the overlay measurement opens in the working area.

Align measurements precisely with position difference ΔX

Proceed as follows to align the time axis for two measurements precisely:

1. Display the vertical measurement cursors via the button.
2. Zoom into the time range, e.g. with the button until you are able to position the first measurement cursor precisely on the desired reference point for the first measurement.
3. Move the first measurement cursor with the mouse to the required position.
4. Search for the reference point for the second measurement, e.g. by switching to “Display all” with the button.
5. Zoom into the time range, e.g. with the button until you are able to position the second measurement cursor precisely on the desired reference point for the second measurement.
6. Move the second measurement cursor with the mouse to the required position.
7. Open the “Trace" task card.
8. In the “Measurement cursor” pane select the position difference value ΔX.
9. Copy the value to the clipboard.
10. Insert the value from the clipboard into the Offset cell of the first or second measurement.

Both measurements are precisely aligned with each other at the desired measurement points.

Note

When inserting the position difference as the offset make sure that you also adjust the leading character as necessary.

6.4.6.7 Printing a recording

The curve diagram supports the saving of the display as a graphic and the copying of the display to the clipboard. Also use these functions (Page 71) for printing.

6.5 Devices

6.5.1 S7-1200/1500 CPUs

6.5.1.1 Recordable variables

Device-dependent recording of tags

The following list shows the operand areas from which tags can be recorded:

- Process image input
- Process image output
- Bit memory
- Data blocks

Data types

A selection of elementary and composite data types can be recorded. The availability of the individual data types depends on the device used:

For more information, see the help under "Overview of valid data types".

The following table lists the supported data types:

Data types	Note
Binary numbers	
BOOL	-
Bit strings	
BYTE	-
WORD	-
DWORD	-
LWORD [1)]	Symbolic name required
Integers	
SINT	-
USINT	-
INT	-
UINT	-
DINT	-
UDINT	-
LINT [1)]	Symbolic name required
ULINT [1)]	Symbolic name required
Floating-point numbers	
REAL	-
LREAL	Symbolic name required

Data types	Note
Timers	
TIME	-
LTIME [1)]	-
Date and time	
DATE	-
TOD	-
LTOD [1)]	-
LDT [1)]	-

1) Not supported by S7-1200.

6.5.1.2 Lifetime of the installed trace configuration and recorded values

Installed trace configurations are retained after POWER OFF. The recording is activated again after the restart of the CPU.

Recorded values are lost during the restart.

Note

Downloading a configuration to the device in the "STOP" operating state

Note that after downloading a configuration in the "STOP" operating state, you must check the installed traces and, if required, reactivate them or transfer them again.

Note

If trigger tags that affect the address are changed, the trace configuration must also be transferred to the device again.

This is the case for example, when a data block is shortened or extended or the data type is changed.

6.5.1.3 Recording levels

All runtime levels can be used for the recording cycle. The cyclic execution levels are offered for selection via the button . In non-periodic recording levels, the recording time is undefined.

Note

The measured values are recorded at the end of the OB after the processing of the user program.

6.5.1.4 Quantity structure

The following table shows the maximum quantity structure that can be recorded with the trace and logic analyzer function:

Device	Maximum number of installed traces	Maximum number of signals per trace configuration
S7-1200 (as of firmware version V4.0)	2	16
S7-1500, ET 200SP, CPU 1516pro-2 PN, S7-1500 Software Controller	At least 4 (depending on the CPU type)	16

Example CPU 1516-3 PN/DP

- Maximum of 3854 samples for 16 signals from PLC tags of the DWORD data type
- Maximum of 21844 samples for 16 signals from PLC tags of the BOOL data type
- Maximum of 58250 samples for 1 signal from a PLC tag of the BOOL data type

Further information can also be found in the form of FAQs under the ID 102781176 (http://support.automation.siemens.com/WW/view/en/102781176).

6.5.1.5 CPU load through trace recording

A trace recording increases the runtime of the respective recording level that can result in an execution level overflow with high utilization of the CPU.

Remedy for execution level overflow:

- **Change the trace configuration**
  1) Configure fewer tags and signals.
  2) Then increase the number of tags and signals up to the maximum number of signals step-by-step without an execution level overflow.
- **Select a slower recording level**

6.5.1.6 Software user interface of the configuration

Structure of the user interface

Display areas in the "Configuration" tab of the working area

The settings options differ depending on the configured device.

The following figure shows an example of the display in the TIA Portal:

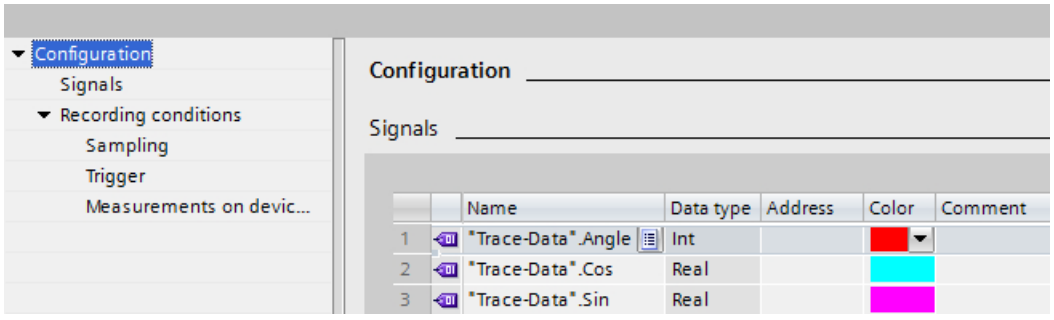

The area navigation provides the following entries for selection:

- Configuration
  - Signals (Page 112)
  - Recording conditions (Page 113)

Displaying and changing properties of a trace configuration

A trace is selected in the project tree and displayed in the "Configuration" tab.

The trace configuration can only be changed offline and is displayed online with the write protection.

User interface - Signals

The "Signals" area shows a table in which the signals to be recorded are configured for the selected trace configuration.

Signals can also be inserted in the table using drag-and-drop.
The signals can be sorted using drag-and-drop.

Setting options and displays in "Signals"

The following figure shows an example of the display in the TIA Portal:

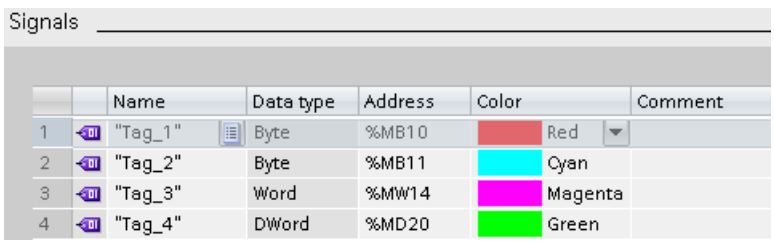

The following table shows the settings and displays:

Column	Icon	Description
		Display of the signal icon for a selected signal.
"Name"	-	Input field for the name or address of the signal. **Examples:** • “Data_block_1”.pressure • M0.0 • DB1.DBW3
-		Button to open the signal selection table. The button is displayed when the table line is selected. Clicking the icon opens a table which offers possible signals for selection. The selected signal is displayed in the input field.
"Data type"	-	Text field with display of the data type for the signal.
"Address"	-	Input field for the address of the signal. The field remains empty with optimized / type correct tags.
"Color"	-	Text field for display and selection of the color. Click the signal color to open the color selection dialog.
"Comment"		Input field for a comment on the signal.

Shortcut menu commands

The following table shows the shortcut menu commands of the table:

Shortcut menu command	Description
"Cut"	Cannot be selected.
"Copy"	Copies the contents of the selected lines to the clipboard.
"Paste"	Pastes the contents of the clipboard to the selected line. The existing contents are overwritten.
"Delete"	Deletes the selected lines from the table or deletes the content of the selected cell.
"Rename"	Switches the selected cell to the editing mode.

Recording conditions

User interface - Recording conditions

The "Recording conditions" area shows the trigger condition for the selected trace configuration and in which cycle, how fast and how long the recording is made. Configuration is possible when the trace configuration is displayed in offline mode or online mode with deactivated viewing .

Setting options and displays in "Recording conditions"

The following figure shows an example of the settings for the sampling in the TIA Portal:

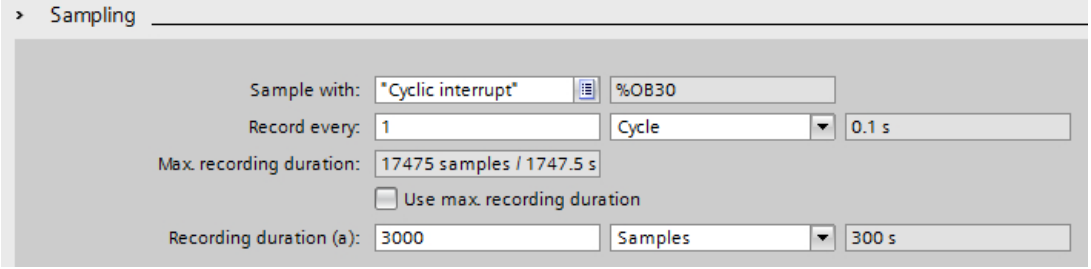

The following table explains the settings and displays:

Setting/display		Description
"Recording time"		
	Recording level entry field	Selection of the recording time. See Recording levels (Page 110)
	Address of the OB text field	Detailed information on the selected recording time.
"Record every"		
	Reduction entry field	Input of the reduction in relation to the reduction ratio and the unit.
	Reduction ratio drop-down list	Selection of the reduction ratio unit The following settings are possible: • "Cycle" • "s" (setting option depends on the recording level selected under “recording time”)
	Sampling time text field	Display of the sampling time, taking into account the configured reduction and the selected unit (only for constant bus cycle time OBs).
"Max. recording duration"		
	Max. recording duration text field	Displays the calculated maximum recording duration. The "Max. recording duration" depends on how many signals are recorded and the data type of these signals.
	"Use max. recording duration"	Set the recording duration to the maximum value. When the checkbox is activated, the recording duration is set to the maximum possible recording duration. The set reduction in the "Record every" input field is taken into account. The recording duration is also adapted when additional signals are added. Further information can also be found in the form of FAQs under the ID 102781176 (http://support.automation.siemens.com/WW/view/en/102781176).
"Recording duration"		
	Recording duration entry field	Input of the recording duration in relation to the selected unit. If the "Recording duration = max. recording duration" checkbox is activated, entries are overwritten by the value displayed in "Max. recording duration".

Setting/display		Description
	Unit drop-down list	Selection of the unit for the recording duration. The following settings are possible: • "Samples" The maximum number of samples recorded is the number for which parameters are assigned under recording duration. • "s" (setting option depends on the recording level selected under “recording time”)
	Calculated recording duration text field	Display of the calculated recording duration (only for constant bus cycle time OBs)

The following figure shows an example of the settings for the trigger in the TIA Portal:

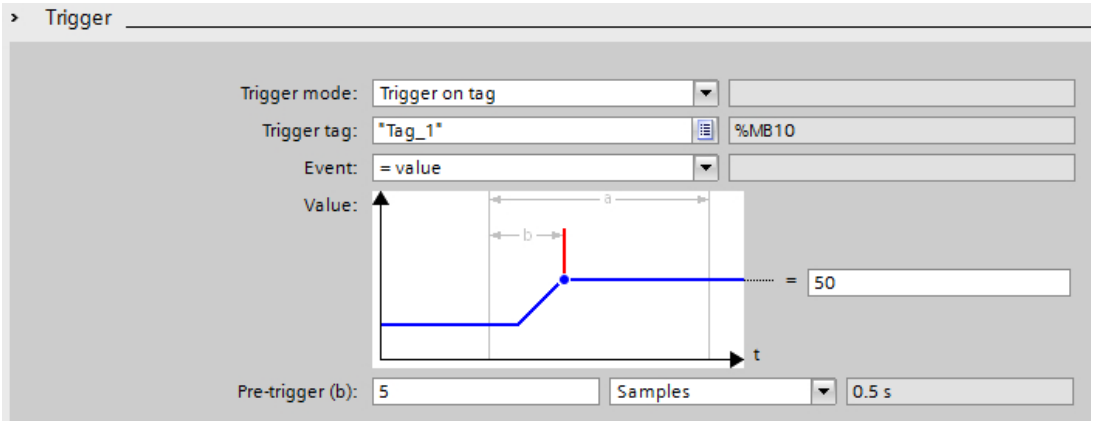

The following table explains the settings and displays:

Setting/display		Description
"Trigger mode"		Selection of the trigger mode.
	Trigger mode drop-down list	The following settings are possible: • "Record immediately" The recording is made immediately after the device is activated. • "Trigger on tag" The recording is made as soon as the installed trace is activated and the configured trigger condition is fulfilled.
	Text field	-
"Trigger tag"		The "Trigger tag" specifies a signal that triggers the recording.
	Trigger tag entry field	Enter a signal. **Examples:** • "DataBlock_1".Temperature • M0.0 • DB1.DBW3 See also Data types for trigger tags (Page 118).

Setting/display			Description
			Opens the signal selection table. Clicking the icon opens a table which offers possible signals for selection as trigger tag. The selected signal is displayed in the input field.
	Trigger tag address text field		Display of the trigger tag address. With purely symbolic signals the field remains empty.
"Event"			The events that can be used on this trigger tag are offered for selection according to the data type of the trigger tag. The event can only be configured when a valid signal has been entered as trigger tag.
	Trigger events drop-down list		Event selection for which the trigger tag is checked. The entries in the drop-down list are described in Section Trigger event (Page 118).
	Text field		-
"Value"			Configuration of the selected event. The configuration options differ depending on the format of the trigger tag and the selected event. SeeTrigger event (Page 118).
"Pre-trigger"			"Pre-trigger" defines the number of samples that are already recorded before the actual trigger condition is fulfilled. If the trigger event occurs immediately or shortly after the activation of the recording, this may result in a shorter recording duration. Examples of "Recording duration (a)" = 20 samples and "Pre-trigger (b)" = 5 samples: • Case 1: Trigger event occurs 50 samples after activation of the recording Actual recording duration (a) = 20 samples • Case 2: Trigger event occurs 2 samples after activation of the recording Actual recording duration (a) = 17 samples
	Duration entry field		Input of the duration in relation to the selection in the drop-down list.
	Unit drop-down list		Selection of the unit The following settings are possible: • "Samples" • "s" (setting option depends on the recording level selected under “recording time”)
	Resulting pre-trigger duration text field		Display of the calculated "Pre-trigger" duration (only for recording in constant bus cycle time OBs)

The following figure shows an example of the settings for the saving of installed measurements:

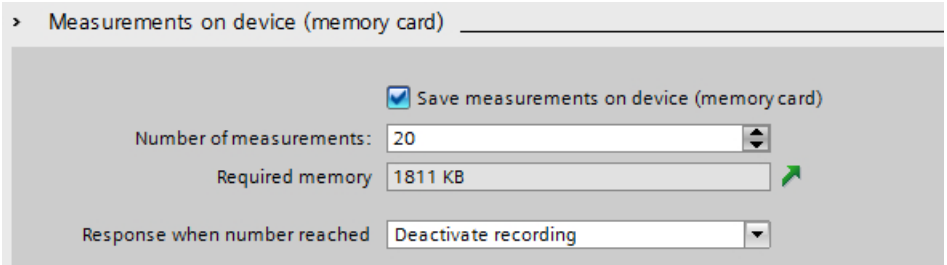

Note

Available memory in the device (memory card)

The memory in the device (memory card) is partly used by system-relevant functions or reserved for that purpose.

Thus it is not possible for the entire memory to be used for saving measurements.

For further information please refer to the Function Manual Structure and Use of the CPU Memory (https://support.industry.siemens.com/cs/us/en/view/59193101).

Note

Memory requirements upon restart

Following a device reboot the maximum number of measurements saved in the device is the number configured under “Number of measurements”.

With repeated restarts note that the measurements already saved are not overwritten and the “Number of measurements” configured in the device is saved once again.

The following table explains the settings and displays:

Setting/display	Description
“Saving of installed measurements (memory card)”	Repeat measurement automatically and store in the device retentively This setting is only possible with “Trigger on tag” trigger mode. The measurements are stored on the “primary” memory card. For traces which have been saved in the device (memory card), the function for automatically repeating the recording is not available. **Note** Only completed measurements are saved on the device. Any recording deactivated by the user is not saved on the device. **This function is available with the following firmware versions:** • S7-1200 as of V4.2 • S7-1500 as of V2.0 The function is not supported by CPU S7-1500 R/H.
“Number of measurements”	Input of the number of measurements to be saved on the card.
“Memory requirements”	Display of the expected memory requirement for all measurements

Setting/display	Description
	Displaying memory usage Shows the tab with the memory usage
“Behavior if number reached”	Selection for the behavior once “Number of measurements” is reached The following settings are possible: • “Deactivating a recording” The measurements are repeated until the “Number of measurements on the card” is reached. • “Overwrite oldest recording” The measurements are saved in a ring buffer and repeated until the user deactivates the recording. Once the number of measurements exceeds the “Number of measurements on the card” the oldest measurement on the card is overwritten in each case. **Note** Note that write processes that are repeated frequently may damage the card.

Data types for trigger tags

The following table shows the supported data types for the trigger tag:

Memory requirement and format of the number	Data type
1 byte	BOOL
8-bit integers	SINT, USINT, BYTE
16-bit integers	INT, UINT, WORD, DATE
32-bit integers	DINT, UDINT, DWORD, TIME, TOD
64-bit integers [1)]	LINT, ULINT, LWORD, LTIME, LTOD, LDT
32-bit floating-point numbers	REAL
64-bit floating-point numbers	LREAL

1) Not supported by S7-1200.

Trigger event

Depending on the selection in the drop-down list, the further settings differ for the "event".

The individual events are described below.

"=TRUE"

Supported data types: Bit (Page 118)

The recording starts when the state of the trigger is TRUE.

"=FALSE"

Supported data types: Bit (Page 118)

The recording starts when the state of the trigger is FALSE.

"Rising edge"

Supported data types: Bit (Page 118)

The recording is started when the trigger state changes from FALSE to TRUE.
After activation of the installed trace, at least two cycles are required to identify the edge.

"Rising signal"

Supported data types: Integers and floating-point numbers (Page 118) (no times, date and time of day)

The recording is started when the rising value of the trigger reaches or exceeds the value configured for this event.
After activation of the installed trace, at least two cycles are required to identify the edge.

"Falling edge"

Supported data types: Bit (Page 118)

The recording is started when the trigger state changes from TRUE to FALSE.
After activation of the installed trace, at least two cycles are required to identify the edge.

"Falling signal"

Supported data types: Integers and floating-point numbers (Page 118) (no times, date and time of day)

The recording is started when the falling value of the trigger reaches or falls below the value configured for this event.
After activation of the installed trace, at least two cycles are required to identify the edge.

"In the range"

Supported data types: Integers and floating-point numbers (Page 118)

The recording starts as soon as the value of the trigger is in the value range configured for this event.

"Outside of the range"

Supported data types: Integers and floating-point numbers (Page 118)

The recording starts as soon as the value of the trigger is outside the value range configured for this event.

"Value change"

All data types are supported.

The value is checked for change when the recording is activated. The recording starts when the value of the trigger changes.

This trigger event is supported as of V13 SP1. Older versions of the TIA Portal cannot interpret the trigger. Note that no explicit information is output in this case. This can occur, for example, when the trace is transferred from a CPU to a TIA Portal less than V13 SP1 or a trace configuration is imported.

"= value"

Supported data types: Integers (Page 118)

The recording starts when the value of the trigger is equal to the value configured for this event.

"<> value"

Supported data types: Integers (Page 118)

The recording starts when the value of the trigger is not equal to the value configured for this event.

"= bit pattern"

Supported data types: Integers and floating-point numbers (Page 118) (no times, date and time of day)

The recording starts when the value of the trigger matches the bit pattern configured for this event.

The following figure shows the setting options for a "bit pattern":

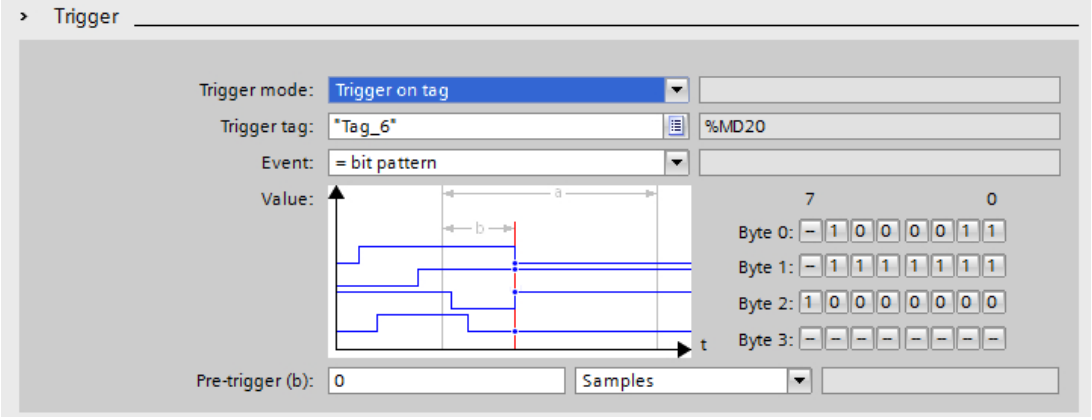

It is possible to switch between the icons by clicking the respective button.

The following table shows the icons:

Icon	Description
–	Bit is not evaluated
0	Bit is checked for FALSE
1	Bit is checked for TRUE

"<> bit pattern"

Supported data types: Integers and floating-point numbers (Page 118) (no times, date and time of day)

The recording starts when the value of the trigger does not match the bit pattern configured for this event.

See also

Configuring the trigger conditions (Page 123)

Recordable variables (Page 109)

6.5.1.7 Configuration

Trace configuration - overview

The configuration of the recording conditions and the signals to be recorded is device-specific.

Requirement

A trace configuration has been created and opened in the working area of the "Configuration" tab.

Procedure

The following table shows the procedure for configuring.

Step	Description
1	Documentation of the configuration (optional) Enter a comment and an author for the configuration in the Inspector window.
2	Selecting signals (Page 122) Select the signals to be recorded in the "Signals" area.
3	AUTOHOTSPOT Select a recording time, a cycle and the duration in the "Recording conditions" area.
4	Configuring the trigger conditions (Page 123) In the "Recording conditions" area, select whether the recording is to be performed immediately or depending on a trigger condition.
5	Configure installed measurements (memory card) (Page 123) In the "Recording conditions" area, select whether the recording is to be saved on the device (memory card).

Selecting signals

Requirement

- A trace configuration has been created and opened.
- The "Signals" area is open in the "Configuration" tab.

Procedure

To configure the signals to be recorded, proceed as follows:

1. Select a signal. The following options are available:
   – In the "Name" column, click the button and select a tag.
   – Enter the symbolic tag name in the cell in the "Name" column.
   – Enter the address directly in the "Address" column.
   – Drag a signal to the table using drag-and-drop.
2. Click in the "Color" column and select a color for the display of the signal.
3. Click in the "Comment" column and enter a comment for the signal.
4. Repeat the procedure from step 1 until all the signals to be recorded have been entered in the table.

Configuring the recording cycle and duration

Requirement

- A trace configuration has been created and opened.
- The "Recording conditions" area is open in the "Configuration" tab.

Procedure

To configure the cycle and the duration of a recording, proceed as follows:

1. Click the button for the recording time.
2. Select an OB for the recording time (Page 110).
3. Select a unit for the reduction factor in the drop-down list for "Record every".
4. Enter the factor for the reduction in the input field for "Record every".
5. Select a unit in the drop-down list for "Recording duration".
6. Specify the recording duration.
   The following options are available:
   – Enter a value for the duration in the input field for "Recording duration".
   – Activate the "Use max. recording duration" checkbox.

Configuring the trigger conditions

Requirement

- A trace configuration has been created and opened.
- The "Recording conditions" area is open in the "Configuration" tab.

"Record immediately" trigger condition

To start the recording immediately, proceed as follows:

1. Select the "Record immediately" entry in the drop-down list for "Trigger mode".
   The input fields for the trigger tag are hidden.

"Trigger on tag" trigger condition

To start the recording depending on a condition, proceed as follows:

1. Select the "Trigger on tag" entry in the drop-down list for "Trigger mode".
2. Select a trigger tag. The following options are available:
   – Click the button for the trigger tag and select a tag.
   – Enter the address or the symbolic name of the tag directly in the input field for the trigger tag.

   A drop-down list with events and input fields is displayed. The display depends on the data type of the tag.
3. Configure the event.
4. Select a unit for the pre-trigger in the drop-down list for "Pre-trigger".
5. In order to record a period before the trigger event, enter a value greater than 0 in the input field for the pre-trigger.

Note

Cyclic test of the trigger condition

The trigger condition is checked in every cycle irrespective of the setting in "Record every". To reliably identify the trigger, the trigger signal must be present for at least one full cycle.

Configure installed measurements (memory card)

Requirement

- A trace configuration has been created and opened.
- The "Recording conditions" area is open in the "Configuration" tab.

- The “Trigger on tag” trigger mode is set.
- The firmware on the device supports the recording of an installed measurement.

Procedure

Proceed as follows to save the installed measurement (on the memory card):

1. Select the "Save measurements on device (memory card)" check box.
2. Enter the number of measurements that ought to be saved on the card in the “Number of measurements” entry field.
3. Set the desired behavior once the “Number of measurements” has been reached in the “Behavior if number reached” drop-down list.

Note

No evaluation of the trigger during saving

No new trigger can be evaluated as long as the recording is saved.

Establishing a remote connection with TeleService 7

7.1 Basics of working with TeleService

7.1.1 Introduction to TeleService

Introduction

TeleService gives your controller telecommunication capability. You can manage, control and monitor distributed plants centrally by means of remote connections.

Scope of functions

TeleService allows you to use the range of TIA portal functions via a telephone network or an Internet connection by establishing a remote connection to a remote system. The online connection allows you to edit a remote system in the usual way with the TIA Portal.

Advantages

Using TeleService offers the following advantages:

- You can easily access even remote sections of plants and include them in a complete system.
- You can offer rapid help and support in the event of faults in a remote system without having to go there yourself.
- You can employ your resources effectively.
- It significantly reduces costs.
- It can significantly reduce plant downtimes.
- It improves the efficiency of your plant.

See also

TeleService functionality (Page 126)

7.1.2 TeleService functionality

TeleService fields of application

TeleService offers the following the fields of application:

- **Access to remote systems (remote maintenance):**
  You can manage, control, and monitor remote systems centrally by means of remote connections.
  This is possible with an S7-300/400 CPU, an S7-1200 CPU and an S7-1500 CPU and a TS Adapter MPI or a TS Adapter IE.
- **Establishing connections from and to remote systems (PG-AS remote link):**
  You can use PRODAVE MPI V5.0 and newer to establish a remote connection to a remote system, and the communications instruction "PG_DIAL" to establish a remote connection from a remote system.
  This is possible with an S7-300/400 CPU and a TS Adapter MPI.
- **Data exchange between systems (AS-AS remote link):**
  The communications instruction "AS_DIAL" allows two automation systems to exchange process data over the telephone network.
  This is possible with an S7-300/400 CPU and a TS Adapter MPI.
- **Sending a text message from a system:**
  An automation system can send a message (SMS) via a GSM wireless modem using the communications instruction "SMS_SEND" ".
  This is possible with an S7-300/400 CPU and a TS Adapter MPI.
- **Sending an email from a system**
  An automation system can also send an email with the following communications instructions and a TS Adapter IE .
  - S7-300/400 CPUs (CPU S7-31x-2PN/DP or CPU 41x-3PN/D) use the instruction "AS_MAIL"
  - S7-1200 CPUs use the instruction "TM_MAIL"
  - S7-1500 CPUs use the instruction "TMAIL_C"

7.1.3 Telephone book at TeleService

Introduction

By double clicking the "phone book" folder in the project tree you open the phone book editor displaying the TeleService phone book.

Each version of the TIA Portal has its own "global phone book". If a global phone book from an earlier version of the TIA Portal is found in a more recent version of the TIA Portal, you will be asked once if you want to import this phone book.

This gives you the advantage of being able to import the system data from the previous version into the more recent version of the TIA Portal.

Global phone book properties

The global phone book is used in TeleService to manage specific system data that are required to establish a remote connection.

When you open the phone book for the first time, you will see an empty phone book with all available columns; in all other cases, the last phone book edited is displayed.

You can enter any number of systems in a phone book. Systems contain the data required for establishing a remote connection, for example, the name and location of the device and the phone number to be dialed, along with all country-specific details. With VPN connections, you can enter an IP address or a DNS name instead of the phone number.

The TS adapters used for establishing the connection are distinguished by color, depending on whether a TS Adapter MPI or a TS Adapter IE is used for establishing the connection.

7.2 Working with the phone book

7.2.1 Basics on working with the phone book

Working with the phone book

You have the following options when working with a phone book:

- Open phone book
- Save phone book
- Import phone book data
- Export phone book data
- Printing the phone book
- Use phone book data to establish a remote connection

You can implement these functions simply and easily by using the buttons in the phone book toolbar.

Note

Access to phone books

The phonebook is user specific in TeleService. However, it is not possible to access the global phone book with more than one instance of the TIA portal at the same time.

See also

Open phone book (Page 130)

Saving phone book (Page 131)

Exporting phone book data (Page 133)

Printing the phone book (Page 134)

Structure of the phone book (Page 128)

7.2.2 Structure of the phone book

Introduction

A global phone book in TeleService is used to manage the data you require for establishing a remote connection. Once you have created the connection data and saved it in the phone book, you can access it each time you want to establish a remote connection.

Structure of the phone book

The integrated global phone book in TeleService contains the following columns:

Column name	Description
Plant name	Enter a name for your system.
Adapter type	Select the TS Adapter type used in the drop-down list: TS Adapter MPI or TS Adapter IE.
Connection type	Select the required connection type: Dial-up connection or VPN connection.
Area code	Enter the required area code. This column is only active with dial-up connections. This column cannot be edited for VPN connections.
Phone number/remote address	Enter the required connection data for establishing the remote connection. For dial-up connections, this can be a phone number, and for VPN connections a DNS name or an IP address.
Fingerprint	Enter the corresponding fingerprint for establishing a VPN connection to the TS Adapter IE Advanced.
Country	Enter the country code. This column is only active with dial-up connections. This column cannot be edited for VPN connections.
User name	Enter the user name you have logged on under.
Password	Enter the password for this user name.
Group	Enter the group if you have carried out grouping.
Company	Enter the company to be contacted.
Department	Enter the relevant department.
Street	Enter the street.
Town/City	Enter the town or city to which the remote connection is to be established.
Comment	Enter a comment if required.

Showing or hiding columns

You can show or hide the individual columns. To do so, select the required column header and open the shortcut menu with the right mouse button.

7.2.3 Symbols in the phone book

Meaning of the TeleService icons

The following table shows the meaning of the TeleService icons:

Symbol	Meaning
	Open the global phone book
	Imports a phone book
	Exports a phone book

Symbol	Meaning
	Establishes a remote connection
	Terminates the active remote connection
	Establishes a remote connection or terminates it
	Displays the connection to a TS Adapter IE in the phone book
	Displays the connection to a TS Adapter MPI in the phone book
	Adds a new row to the phone book
	Inserts a new row in the phone book

7.2.4 Manage phone book

7.2.4.1 Open phone book

Opening phone books

To open the phone book, follow these steps:

1. In the project tree, double-click on the "Phone book" folder under "Online access" > "TeleService".
2. The phone book opens so that you can enter or edit the required system data.

7.2.4.2 Inserting rows in the phone book

Inserting rows in the phone book

To insert a new row in the phone book, follow these steps:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button in the toolbar.

Result

A new row is inserted in the phone book above the selected row.

7.2.4.3 Showing and hiding columns in the phone book

Showing and hiding columns

To show or hide columns in the phone book, follow these steps:

1. Click a column header.
2. In the shortcut menu, select the "Show/hide columns" command.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

Result

The respective columns are shown or hidden when displaying the phone book.

7.2.4.4 Saving phone book

Saving phone books

When you exit the phone book editor or the TIA Portal, you are asked whether you want to save the global phone book.

Click "Yes" to save the phone book.

7.2.4.5 Importing phone book data

Introduction

It is possible to import the phone book data from an external file or from an older version of the TIA Portal.

Requirement

You have already created an import-capable phone book file.

You have already created a phone book with an older version of the TIA Portal.

Importing phone book data from a phone book file

To import phone book data from a phone book file, follow these steps:

1. Open the "TeleService" folder under "Online access" in the project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Import" button in the toolbar.

4. Confirm the query as to whether the current version of the work phone book should be saved, if appropriate with "Yes" and in the next dialog select where the phone book will be saved.
5. If you do not want to save the current state of the phone book, answer the prompt with "No". In the subsequent dialog, select the phone book file to which the current phone book should be stored.
6. Close the dialog box with "OK".

Result

The imported phone book data is displayed in the global phone book.

Note

Defining dialing rules

Dialing rules must be defined before the "area code" and "country" columns in the imported phone book can be edited for dial-up connections.

To define dialing rules, follow the link in the history.

Importing phone book data from an older version of the TIA Portal

To import the phone book data from an older version of the TIA Portal follow the steps below:

1. Open the "TeleService" folder under "Online access" in the project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Import" button in the toolbar.
4. Confirm the query as to whether the current version of the work phone book should be saved, if appropriate with "Yes" and in the next dialog select where the phone book will be saved.
5. If you do not want to save the current state of the phone book, answer the prompt with "No".
6. In the following dialog enter the following path information "%appdata%\siemens \automation\TeleService\GlobalTeleServicePhoneBook.tel".
7. Close the dialog box with "OK".

Result

The phone book data imported from an older version of the TIA Portal is displayed in the global phone book.

Note

Defining dialing rules

Dialing rules must be defined before the "area code" and "country" columns in the imported phone book can be edited for dial-up connections.

To define dialing rules, follow the link in the history.

See also

Defining dialing rules (Page 134)

7.2.4.6 Exporting phone book data

Introduction

It is possible to export phone book data to an external file.

Requirement

You have already created a phone book under TeleService with the corresponding system data.

Procedure

To export the phone book data, follow these steps:

1. Open the "TeleService" folder in project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Export" button in the toolbar.
4. In the next dialog box, select where the current phone book is to be exported.
5. Close the dialog box with "OK".

Result

The exported phone book data are saved in the specified export file.

7.2.4.7 Printing the phone book

Printing phone books

You can print out all or some of the data in a phone book.

Proceed as follows:

1. Open the phone book.
2. Select the **Project > Print** menu command or click on the appropriate button in the toolbar. The "Print" dialog will open.
3. Specify whether you wish to print the complete phone book or just part of it and set all other options.
4. Start the print job with "OK".

Result

The phone book data is printed on the default printer. If the printout is more than one page long, an identifier is printed after the page number at the bottom right corner of the page to indicate that there is another page. The last page does not have this symbol, indicating that no more pages are to follow.

7.2.4.8 Defining dialing rules

Procedure

To define location-specific dialing rules, follow these steps:

1. Open the "Online access" folder in the TIA Portal and select the "TeleService" folder.
2. Use the shortcut menu to access the "TeleService" properties.
3. In the dialog that follows, open the "Advanced settings" tab and activate the "Use dialing rules" option.
4. Click the "Adapt" button.
5. In the dialog that follows, enter the required location information for connection establishment.
   - For new locations, enter the country or region and the corresponding area code.
   - If required, set the correct destination code and pre-dial line for local/long-distance calls.
   - Select your chosen dialing mode for the location.
6. Exit the dialog by clicking "OK".
7. In the next dialog, select the location from which you wish to dial, and click "OK".

8. In the TeleService dialog that follows, check that the correct location appears under "Location".
9. Confirm your entries by clicking "OK".

Note

Modem operation on a local loop or extension

You do not need to specify a pre-dial line if you operate your modem on a local loop (main telephone line). The fields for the pre-dial lines for local calls and long-distance calls must be empty.
If you operate your modem on an extension, you should enter the pre-dial lines which need to be called to access a local loop.

Example of the use of dialing rules

The following example illustrates the use of location-specific dialing rules for the city of Karlsruhe in Germany.

- In the TeleService phone book, you enter the phone number "1234567", without the area code and without the country code.
  Result:
  The phone number "1234567" is always dialed, irrespective of the location of the caller.
- In addition to the phone number "1234567", you enter the location-specific dial parameters for the Karlsruhe area code "0721" and the country code "+49" for Germany in the TeleService phone book.
  Result:
  "1234567" is dialed from Karlsruhe.
  "07211234567" is dialed from other towns in Germany.
  "00497211234567" is dialed from other countries.

Note

Display in the phone book

The "area code" and "country" columns in the TeleService phone book can only be edited once you have defined the dialing rules as detailed above.

7.3 Remote connections as dial-up connections

7.3.1 Basics for establishing a dial-up connection

Using a TS Adapter for dial-up connections

A TS Adapter is needed for establishing a remote dial-up connection using TeleService.

The TS Adapter is used to prepare an automation system for use with TeleService by connecting it to a telephone network via a modem. The TS Adapter has an integrated parameter memory in which a parameter set for TeleService operation is stored.

With the function "export adapter parameter", different parameter sets can be saved in external files, and reloaded in the TS Adapter via the function "import adapter parameter".

Establishing a remote dial-up connection

You can choose between a number of different TS Adapters, each of which offers a different functionality and different connection options.

The figure below shows two possible configurations for establishing a dial-up connection to a system with the TS Adapter IE.

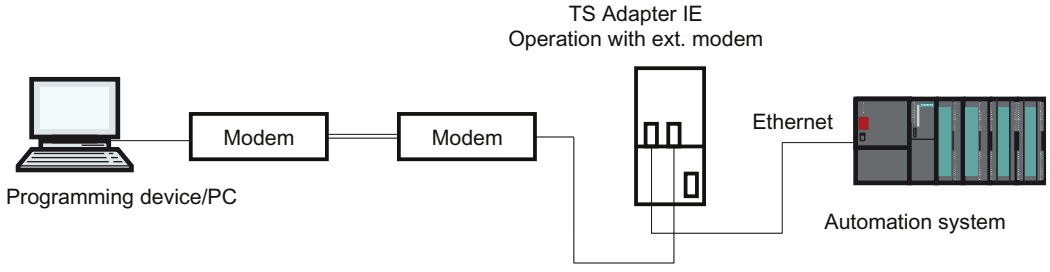

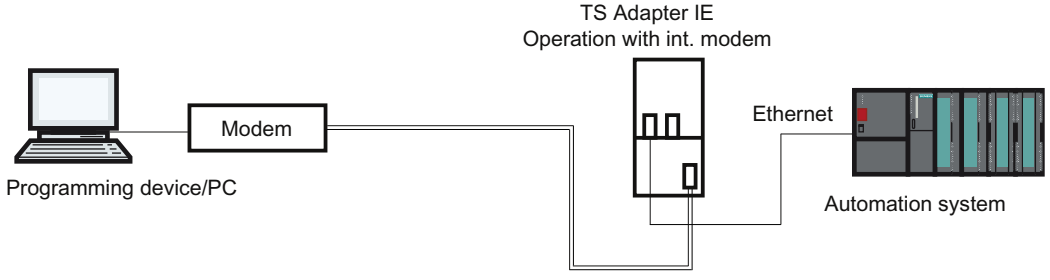

Overview of possible TS Adapter:

The TS Adapter comes in the following versions:

- TS Adapter II (also designated "TS Adapter MPI")
- TS Adapter IE Standard (also designated "TS Adapter IE")
- TS Adapter IE Basic (also designated "TS Adapter IE")

Designation "TS Adapter"

In the following pages, the designation "TS Adapter" stands for all versions. The relevant product designation is listed beside information which only applies to a specific version, for example, "TS Adapter II" or "TS Adapter IE Standard".

Note

For more detailed information on your TS Adapter, please refer to the documentation supplied with the TS Adapter.

See also

Short description of the TS adapter MPI (Page 145)

Short description of the TS adapter IE (Page 152)

Exporting adapter parameters (Page 150)

Importing adapter parameters (Page 151)

7.3.2 Telephone networks and modems

7.3.2.1 Supported telephone networks and modems

Telephone networks which can be used

TeleService can be used with digital networks (ISDN), analog networks and wireless networks (with GSM technology). This version supports a remote connection to a TS Adapter.

Modem support

TeleService has been implemented to be independent of the modem. This means that all standard modems which can be installed in the Windows Control Panel and are visible as modems can also be used by TeleService.

The choice of modem type is determined primarily by the existing hardware of the programming device/PC and the telephone network to be used.

The following modem types/media are supported:

- Modems (external modems at the COM interface, internal modems and PCMCIA cards)
- External ISDN adapter at the COM interface or USB interface
- Internal ISDN adapter with virtual COM interface (for example AVM CAPI port)
- External ISDN modems (ISDN adapter with integrated analog modem functionality) at the COM interface or USB interface
- Radio network modems with GSM technology, PCMCIA adapter card or data cable and mobile phone

Gateways

Gateways between the various telephone networks are in principle possible. Remote connections from an ISDN adapter to an analog modem and vice versa only function with special ISDN telephone adapters.

Performance in telephone networks

The data throughput of a remote connection depends on the modem and telephone network used and the quality of the telephone line.

7.3.2.2 Installing the local modem

Introduction

If you have already installed a modem for data transfer in your operating system, this modem can also be used for TeleService.

If a modem has not yet been installed for your operating system, one must be installed before you can establish a remote connection with TeleService.

Procedure

Proceed as follows:

1. Make sure your programming device/PC and the modem are switched off.
2. Physically connect the external modem to a COM or USB interface on your programming device/PC. You can also install an internal modem or a PCMCIA card in accordance with the manufacturer's specifications.
3. Now switch on the modem and then the programming device or the PC.

Result

Plug-and-play modems are recognized and installed automatically by the operating system. Dialogs will take you through the installation procedure.

Note

Modems without plug-and-play

If your modem is not recognized automatically when switched on, you will have to install it yourself using the Control Panel.

Please refer to the information in the documentation supplied with your modem.

7.3.2.3 Connecting and configuring the remote modem

Introduction

A modem must also be connected to the remote system before you can work with TeleService. This modem is designated the "remote modem".

Configuring the remote modem

The modem receives all parameters required for operation from the TS Adapter connected. These include data for initializing the modem and settings for serial transmission between the TS Adapter and the modem.

The data required for the remote modem is specified during the configuration of the TS Adapter.

The modem may be internal or external depending on the TS Adapter used.

How to connect a TS Adapter with an internal modem

1. Switch off the TS Adapter.
2. Connect the TS Adapter to the automation system.
3. Connect the TS Adapter to the telephone line.
4. Switch on the TS Adapter.

How to connect a TS Adapter with an external modem

1. Switch off the modem.
2. Connect the TS Adapter to the automation system.
3. Connect the TS Adapter to the modem using a modem cable.
4. Connect the modem to the telephone line.
5. Switch on the modem.
6. Switch on the TS Adapter.

Note

Please note the following information on configuring the remote modem:

- The default parameters for the modem and the serial port set in the TS Adapter should in most cases ensure successful operation; changes in the parameter assignment will only be needed in rare cases.
- You need only change the parameter assignment of the TS Adapter if a modem connection is not established or if factory settings are to be adapted or optimized.
- TS Adapter parameter assignment can be changed via either a direct or a remote connection.

7.3.3 Access protection for dial-up connections

7.3.3.1 Access protection information

Introduction

When you assign the parameters for your TS adapter, you can restrict access to the parameters of the TS adapter and access to remote systems.

Scope of access protection

Access protection only exists for remote connections; TS adapter parameter assignment can be accessed at any time in direct connection mode.

Access protection also exists in direct connection for the TS adapter IE.

Access protection information

The TS Adapter MPI is not delivered with access protection activated. There is a default password for the TS adapter IE.

The first user who assigns the parameters for this adapter can therefore activate access protection by defining the password for a user and/or a callback number.

This is a multi-level access protection with several users, each with or without administrator rights. For the TS adapter MPI there is only one adminstrator and no more than two users.

For a modem connection, only an administrator can define the two users and change and, if necessary, delete their settings. Those logged in as users can only change their own passwords and their own callback numbers. However, with the TS adapter MP you can access the process of assigning parameters of the TS adapter in direct connection, without restriction.

Advantages

Access protection offers the following advantages:

- Unauthorized access by persons outside the system is almost impossible.
- The plant operator bears most of the telephone costs.

7.3.3.2 TeleService callback options

Callback variants

The costs of a telephone connection are normally borne by the caller who establishes the dial-up connection.

TeleService can, however, be used so that after a short initial connection the modem connection is established again in the opposite direction, in other words initiated by the TS Adapter (callback). In this case, the plant operator bears the costs of the callback.

There are two callback variants in TeleService:

1. Callback to a number specified during connection establishment.
2. Callback to a number stored on the TS Adapter.

7.3.3.3 Levels of protection

Introduction

You can set up one of two possible levels of access protection for TeleService access to the TS Adapter. Different options are available with each protection level.

Access protection options

Access protection level 1:

The TS Adapter is protected by the user name and password. You can access the TS Adapter via any telephone line and specify any callback number during connection establishment.

Access protection level 2:

The TS Adapter is protected by the user name, password, and the callback number. You can only access the TS Adapter from one telephone connection per user.

The table below sets out the above conditions for the various protection levels:

Level of access protection	Administrator/User password	Callback number
1	enter	do not enter
2	enter	enter

Logging on to TS Adapter

When you log on to the TS Adapter and after you have set up access protection, enter your user name, the corresponding password and, if desired, a callback number:

Level of access protection	Administrator/User password	Callback number
1	enter	do not enter or enter any callback number
2	enter	do not enter

If you have entered a callback number during connection establishment (access protection level 1) or have stored a callback number in the TS Adapter (access protection level 2), the modem connection is disconnected and the TS Adapter calls the specified number back.

7.3.3.4 Setting up access protection and callback number for the TS adapter

Introduction

During the parameter assignment for the TS adapter MPI in TeleService, you can set up access protection and a callback number for the parameter assignment of the adapter and connection to the remote system. The following describes the parameter assignment for a TS adapter MPI. The parameter assignment of a TS adapter IE is carried out in analog. The specific method is described in the web help of this adapter.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Accessible nodes".

Procedure

To set up access protection for the TS adapter, proceed as follows:

1. Click on the command "Assign TS Adapter MPI parameters" in the project tree.
2. Open the "Access security" tab.
3. Enter a password for your user name and/or number that you want the modem to call back following logon.
   - If you are an administrator, you can change all the settings for administrators and users, and delete or create users.
   - If you are logged on as a user, you can only change your own settings (password and callback number).
4. Confirm all entries before exiting the dialog with "OK".
5. Click the "Yes" button to confirm the following query.

Result

The parameter assignment for the access protection and the callback number is saved in the non-volatile memory of the TS adapter MPI.

Note

Important points to note when setting up access protection:

- The settings in the "Modem" tab must correspond to the conditions at the plant if callback functionality is to be guaranteed.
- Entering an incorrect callback number in the role of "ADMIN" user will mean you are no longer able to access the TS Adapter MPI over a remote connection!
- Test the callback number before you enter it as the "ADMIN" user by calling the given callback number during connection establishment (access protection level 1).

7.3.3.5 Complete a callback in TeleService

Callback options

Two different callback variants can be set up in TeleService.

The following callback options are available:

- Callback to a number specified during connection establishment.
- Callback to a number stored on the TS Adapter

Callback to a number specified during connection establishment

1. In the project tree of the TIA Portal, click the "Online access" folder.
2. Click on the "TeleService" folder it contains.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. In the "Adapter type" drop-down list, select the adapter type used.
5. Select the "dial-up connection" under "Connection type" if it is not already selected.
6. Select the modem you are using under "Local Settings".
7. Enter the phone number to be dialed in the appropriate box or open the phone book by clicking on the button behind it and take the desired phone number from the phone book.
8. Enter your user name and associated password of the TS adapter.
9. If you want a "Connection setup with callback", select the appropriate option button.

10. Click the "Connect" button to establish the required remote connection. This button only becomes active when you have entered all the parameters needed to establish a remote connection. Any remote connection is displayed under "Status".
11. Enter the desired callback number in the dialog that follows.

Result

The remote connection to the desired system is made with callback.

The connected system is shown with the corresponding icon in the project tree.

Note

This procedure is useful if the costs of the modem connection are to be borne by the plant operator and if the actual callback number is not fixed, which means callback is not always to the same receiver. It is particularly useful for mobile users.

Callback to a number stored on the TS Adapter

1. Assign the parameters for the desired callback number in the TS adapter.
2. Establish a connection to the TS adapter as described above, and observe the following features:
   - Enter the user name and password for which the callback number parameters are assigned in the TS adapter.
   - The check box "Establish a connection with callback" does not have to be selected, since the callback number is already known by the TS adapter.

Result

Callback to a number stored on the TS adapter has been established. If a remote connection is established, the callback occurs from the remote system.

Note

This procedure offers the highest level of access protection. However, it does pose a risk: if the callback number stored on the TS Adapter is not correct, it will no longer be possible to access the TS Adapter over a modem connection. The device can in such a case only be put back into operation by changing the parameter settings on site.

7.3.4 TS adapter MPI

7.3.4.1 Short description of the TS adapter MPI

TS Adapter MPI:

The designation "TS Adapter MPI" is a collective term for allTS Adapter with an MPI/DP interface.

The TS Adapter MPI comes in the following versions:

- As TS Adapter I (parameters cannot be assigned via the TIA Portal)
- as TS Adapter II

The table below provides a short description of the functionalities. For detailed information on your TS Adapter, please refer to the documentation supplied with your TS Adapter.

TS Adapter II:	
Direct connection	Connection via the Universal Serial Bus (USB). The firmware can be replaced. The modem is integrated or can also be connected as external modem. The TS Adapter II switches automatically between the modems. As long as no external modem is connected, the adapter will use the internal modem.
There are two variants	• With internal analog modem. An external modem can also be connected to the RS232 port. • With internal ISDN adapter. An external modem can also be connected to the RS232 port.

Use of the designation "TS Adapter"

For TeleService the designation "TS Adapter" is the generalization for all versions. The relevant product designation is listed beside information which only applies to a specific version of a TS Adapter, for example, "TS Adapter II", "TS Adapter IE Standard" or "TS Adapter IE Basic".

7.3.4.2 How the TS adapter MPI works

How the TS Adapter MPI Works

In line with the configuration, the TS Adapter MPI connects the serial port or USB port of your programming device/personal computer (direct connection) or the serial port of a modem (modem connection) to the MPI/PROFIBUS network of your automation system.

The TS Adapter MPI has a non-volatile memory. Parameters for the following functions are stored in this memory:

- The MPI/PROFIBUS network (network parameters)
- The mode of the modem used

- The serial port to the modem
- Access protection

Default parameter assignment

The TS Adapter comes with default parameter assignment. The parameters can be set and saved to the non-volatile memory of the TS adapter in a parameter assignment session.

When "Direct connection" is configured, the TS Adapter will only use the network parameters for access to the MPI/PROFIBUS network.

In the "Modem connection" configuration, all the parameters stored on the TS Adapter will be activated.

Note

For more detailed information on the configuration of your TS Adapter, please refer to the documentation supplied with it.

7.3.4.3 Operating a TS adapter MPI in direct connection mode

Direct connection with TS Adapter MPI

The direct connection is used to assign the parameters of the TS Adapter MPI. The same configuration also allows you to go online in the TIA Portal and thereby check the assigned MPI/PROFIBUS parameters for bus compatibility. This means that (as with a PC adapter) SIMATIC S7/C7 systems can be accessed via the MPI/DP interface without an MPI/PROFIBUS module occupying a slot for a programming device/PC.

Access protection for the TS Adapter is not active in direct connection configuration. This means that the parameter assignment of the TS Adapter can be changed without any problems, for example by importing adapter parameters.

Note

Display of the TS Adapter MPI in the TIA Portal

As soon as you have connected a TS Adapter MPI to the programming device/PC via the USB interface, the "TS Adapter" folder is displayed in the project tree in the TIA Portal.

When you open the folder, you can assign the parameters of the connected TS Adapter MPI via the following dialog.

Establishing the direct connection for TS Adapter MPI

Direct connection mode means there is a direct connection via the TS Adapter MPI between the programming device/personal computer on which TeleService is installed and the automation system. No modem is required.

The figure below shows the configuration of the TS Adapter MPI with a direct connection.

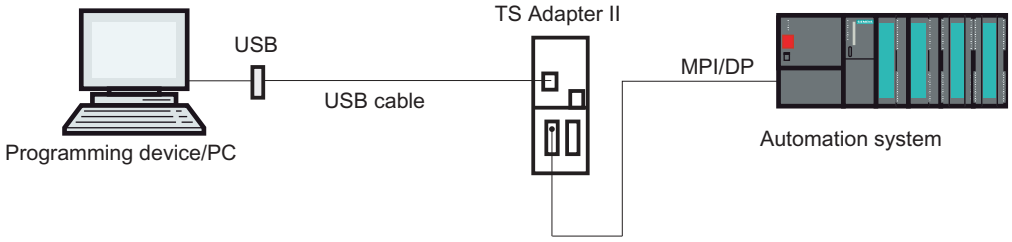

7.3.4.4 Operating a TS adapter MPI in modem connection mode

Introduction to the modem connection with TS Adapter MPI

This configuration allows you to dial into a remote system. To do this, you establish a remote connection to a remote system using TeleService on a telephone network. Via the established modem connection you can then process the selected plant as usual with the TIA Portal.

Establishing a modem connection with TS Adapter MPI

This connection between the programming device or PC on which TeleService is installed and the automation system in which the TS Adapter MPI is inserted in the MPI/DP interface is made through a modem route.

The configuration therefore includes the programming device or PC via the telephone network and the TS Adapter MPI on the MPI/DP interface of the automation system.

The figure below shows the structure of the modem connection.

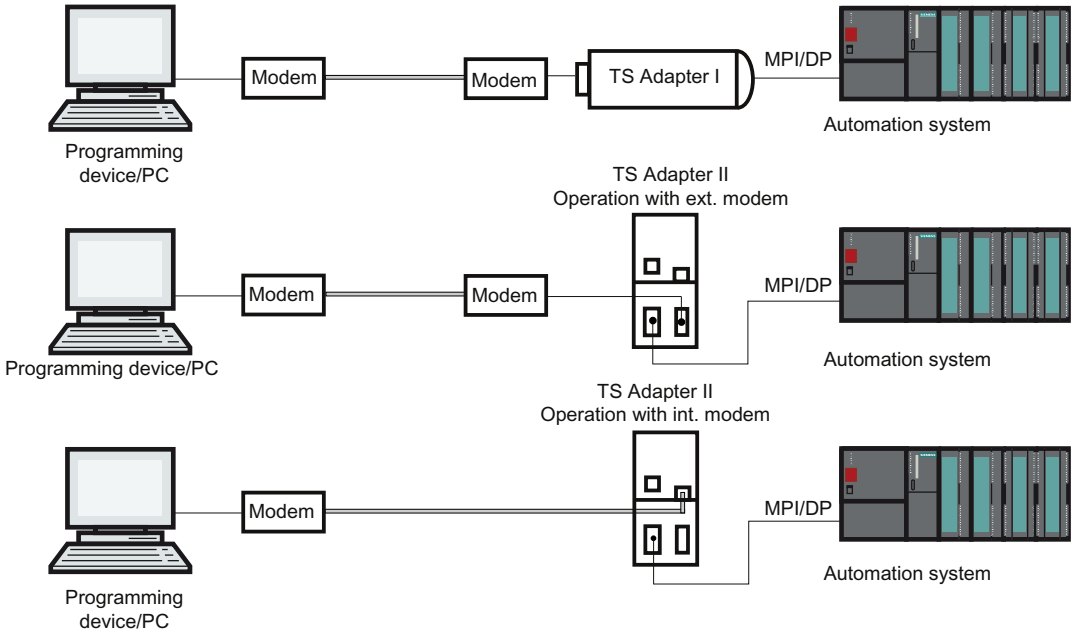

Note

Parallel operation between direct and modem connection

The TS Adapter II has two connections for communication with PG/PC, both of which can be connected at the same time. At the same time, connect the USB interface with the PG/PC and the modem interface with the telephone network.

In this configuration you can either use the direct or the modem connection.

A parallel operation is **not** possible!

7.3.4.5 TS adapter MPI configuration options

Useful information on configuring the TS Adapter MPI

The TS Adapter MPI can be configured in both direct connection mode and via an existing remote connection.

The following parameter assignment options are available:

- Reconfiguration (Page 149)
- Restoring default parameter assignment (Page 150)
- Importing adapter parameters (Page 151)
- Exporting adapter parameters (Page 150)
- Setting up access protection (Page 142)

Parameter assignment

Configure your TS Adapter in accordance with the documentation supplied with the TS Adapter. It will detail the exact procedure for parameter assignment.

Note

Please note the following when configuring the TS Adapter MPI

- If you change the current parameter settings when there is an established remote connection, there is a risk it will not subsequently be possible to establish a modem connection with the modified parameters. The TS Adapter MPI can in this case only be configured in direct connection mode.
- This means that either parameter assignment must be carried out with a programming device/personal computer at the plant location or the TS Adapter MPI must be brought to the location of the local programming device/personal computer in order to be configured.

Positive acknowledgement

During parameter assignment, the data is written to the non-volatile memory of the TS Adapter MPI. The parameter assignment process is not acknowledged positively until all precautions have been taken to ensure that parameter changes have been carried out correctly and will thus survive a power failure.

Changes become effective for the TS Adapter MPI as follows:

- The serial parameters, the modem parameters and the parameters for access protection are activated once the remote connection has been terminated.
- The modified network parameters are activated immediately.

7.3.4.6 Configuring TS adapter MPI

Introduction

The TS Adapter MPI can be configured in both direct connection mode and via an existing remote connection in modem connection mode.

The following describes the method for assigning parameters.

Requirement

A TS Adapter MPI is connected to your computer and the folder "TS Adapter" is displayed in the project tree under "Online access".

Procedure

To assign the parameters for the TS Adapter MPI im Direktanschluss , follow these steps:

1. Double-click the "Online access" folder in the project tree.
2. Open the required folder:
   - The "TS Adapter" folder for a direct connection.
   - For an existing remote connection, the "TeleService" folder followed by the folder with the required system name.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Set the required parameters in the individual tabs of the dialog.
5. Confirm your entries with "OK".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter MPI. Parameter assignment is then complete.

7.3.4.7 Restoring default parameter assignment for TS adapter MPI

Introduction

You can restore the default, factory state parameters of the TS Adapter MPI.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

To restore the default parameter assignment for the TS Adapter MPI, follow the steps below:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Reset" button under "General".
5. Confirm your entries with "OK".

Result

The TS Adapter MPI default parameters set on delivery are restored.

See also

TS adapter MPI configuration options (Page 148)

7.3.4.8 Exporting adapter parameters

Introduction

You can export the configuration of a TS Adapter MPI to an external file. The configuration saved in this file can be imported in turn into any number of TS Adapter MPI.

This can for example be useful if you want to assign identical parameters to multipleTS Adapter MPI or if you want save, document or distribute the parameter set.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

To export the adapter parameters of a TS adapter MPI, follow the steps below:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Export" button.
5. A window will open in which you can select the file to which you wish to export the configuration of the TS Adapter MPI.
6. Confirm with "Save".

Result

The parameters of the TS Adapter MPI are saved in the specified file (*.tap). The export of the adapter parameters is now complete.

7.3.4.9 Importing adapter parameters

Introduction

You can import the configuration of a TS Adapter MPI from a previously created export file (*.tap).

The configuration saved in this file can be imported into any number of TS Adapter. This can for example be useful if you want to assign identical parameters to multiple TS Adapter MPI.

You can import parameters locally in direct connection mode or via an existing remote connection in modem connection mode.

The TS Adapter takes on the parameters from the selected export file and continues working with these settings. All settings made up to this point that have not been saved in an export file are lost.

Before importing the file, back up the current configuration in an export file. You can restore the configuration if necessary.

Requirement

- A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

To import the adapter parameters of a TS Adapter MPI:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.

3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Import" button.
5. A dialog will open in which you can select the file to which you wish to import the configuration of the TS Adapter MPI.
6. Confirm the next dialog with "Yes".

Result

The parameters selected are saved in the non-volatile memory of the TS Adapter MPI. Adapter parameter import is then complete.

7.3.5 TS adapter IE

7.3.5.1 Short description of the TS adapter IE

TS Adapter IE

The designation "TS Adapter IE" is a collective term for allTS Adapter with an Ethernet port.

The TS Adapter IE comes in the following versions:

- as TS Adapter IE Standard
- as TS Adapter IE Basic

The tables below provide a short description of the functionalities. For detailed information on your TS Adapter, please refer to the documentation supplied with it.

TS Adapter IE Standard:
Direct connection by means of Industrial Ethernet (IE). Firmware update possible. Modem integrated or external. The TS Adapter IE cannot automatically switch between modems like the TS Adapter II. Parameters are assigned via a Web interface.
There are 2 variants:
• With internal analog modem. An external modem can also be connected to the RS232 port.
• With internal ISDN adapter. An external modem can also be connected to the RS232 port.

TS Adapter IE Basic:
Direct connection by means of Industrial Ethernet (IE). Firmware update possible. Plug-in modules. Parameters are assigned via a Web interface.
There are 4 variants:
• TS Adapter IE Basic MODEM: Basic device TS Adapter IE Basic with TS Module MODEM for operation on the analog telephone network.

TS Adapter IE Basic:
• TS Adapter IE Basic ISDN: Basic device TS Adapter IE Basic with TS Module ISDN for operation on ISDN telephone systems.
• TS Adapter IE Basic GSM: Basic device TS Adapter IE Basic with TS Module GSM for operation on the GSM radio network.
• TS Adapter IE Basic RS232: Basic device TS Adapter IE Basic with TS Module RS232 for connecting an external modem.

Use of the designation "TS Adapter"

"TS Adapter" is used in the TeleService online help as a general designation for all versions. The relevant product designation is listed beside information which only applies to a specific version of a TS Adapter, e.g. "TS Adapter I", "TS Adapter II", "TS Adapter IE Standard" or "TS Adapter IE Basic".

7.3.5.2 How the TS adapter IE works

How the TS Adapter IE works

The TS Adapter IE connects the telephone network or the serial port of a modem with the Industrial Ethernet of your automation system.

The TS Adapter IE has a non-volatile memory. Parameters for the following functions are stored in this memory:

- The mode of the modem used
- The serial port to the modem
- Access protection

Default parameter assignment

The TS Adapter IE comes with default parameter assignment. The parameters can be set and saved to the non-volatile memory of the TS adapter in a parameter assignment session.

Note

For more detailed information on the configuration of your TS Adapter, please refer to the documentation supplied with it.

7.3.5.3 Connection Types

Connection types of the TS Adapter IE Basic

The following diagrams show the connection types possible with the TS Adapter IE Basic.

Direct connection

In the direct connection to the PG/PC, you can set the TS Adapter IE Basic through Ethernet.

Note

The operation of the TS Adapter IE Basic without a TS module is not permitted.

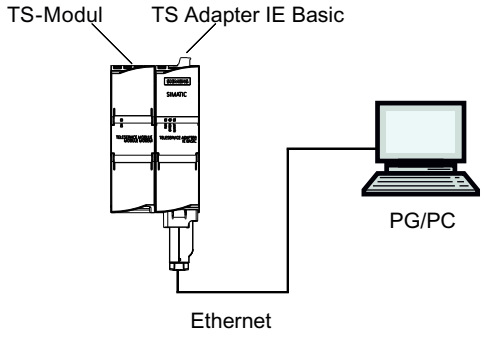

Figure 7-1 Direct connection

Connection to the telephone network

In order to have a direct connection to the telephone network, you must connect the TS Adapter IE Basic together with one of the following TS modules:

- TS Module Modem
- TS Module ISDN

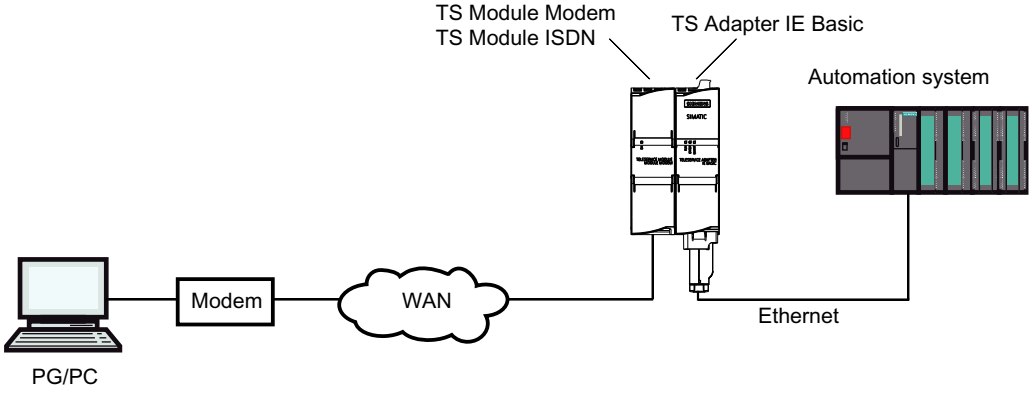

Figure 7-2 Direct connection to the telephone network

More information about the TS modules can be found in the *TS Adapter modular* manual.

Connection to the GSM network

In order to connect to the GSM network, you must operate the TS Adapter IE Basic together with this TS module:

- TS Module GSM

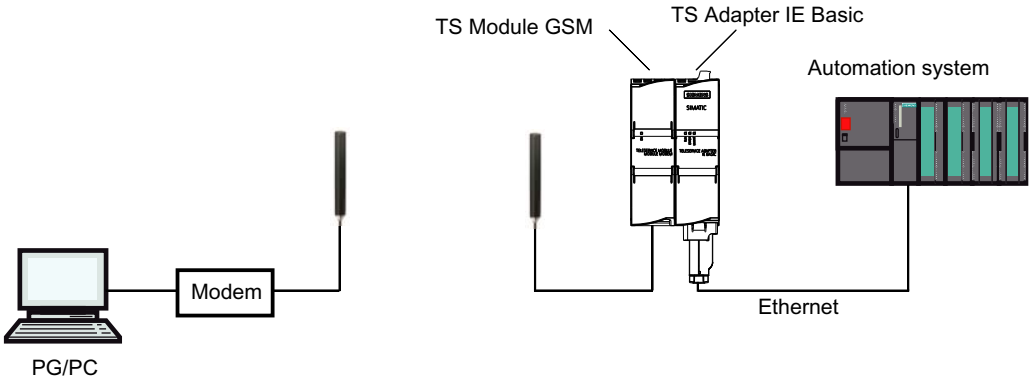

Figure 7-3 Connection to the GSM network

More information about the TS modules can be found in the *TS Adapter modular* manual.

Connection to the telephone network through an external modem

For the connection to an external modem, you must operate the TS Adapter IE Basic together with this module:

- TS Module RS232

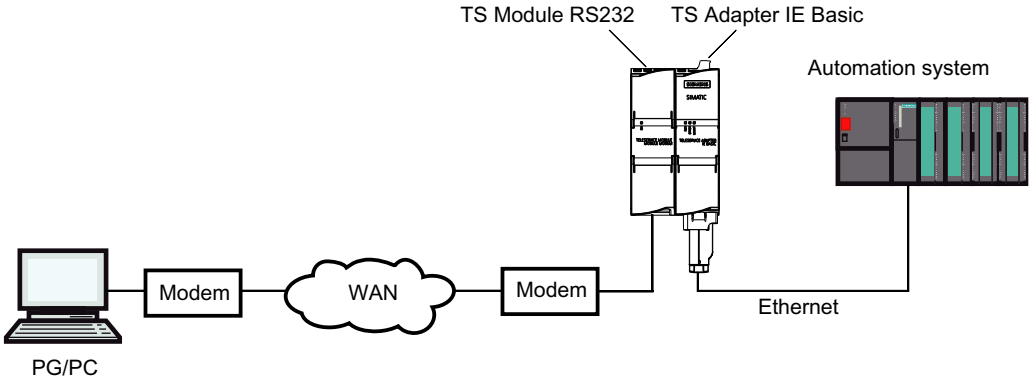

Figure 7-4 Connection to an external modem

More information about the TS modules can be found in the *TS Adapter modular* manual.

7.3.5.4 TS Adapter IE parameter assignment options

Useful information for configuring the TS Adapter IE

The TS Adapter IE is configured via a Web interface.

Web help associated with the parameter assignment interface is made available for the configuring the TS Adapter IE.

The following parameter assignment options are available:

- Reconfiguration
- Restoring default parameter assignment
- Importing adapter parameters
- Exporting adapter parameters

Note

Parameter assignment

Configure your TS Adapter in accordance with the documentation supplied with the TS Adapter. It will detail the exact procedure for parameter assignments.

7.3.5.5 Parameter assignment for TS Adapter IE

Introduction

The TS Adapter IE can be configured in both direct connection mode and via an existing remote connection in modem connection mode.

Both parameter assignment options are described below.

Specific details for assigning parameters of the TS Adapter IE can be obtained from the TS Adapter IE documentation.

Parameter assignment of the TS Adapter IE in direct connection

Requirement

There is a LAN connection to your TS Adapter IE.

The TS Adapter IE Basic is connected to the power supply.

Procedure

To assign the parameters for the TS Adapter IE , follow these steps:

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Double-click on the Ethernet port of your computer.
3. Double-click on the "Display accessible nodes" command. The TS Adapter IE is then displayed.
4. Double-click on the <TS Adapter IE> folder and then on "Online and diagnostic", and assign the desired IP address to the TS Adapter IE in the following dialogs. Please note that the IP address of the PG/PC interface card is located in the same subnet as the IP address that you issue for the TS Adapter IE.

5. Update the view in the project tree for the "Accessible nodes", so that the TS Adapter IE is displayed with the newly allocated IP address.
6. Open the folder <TS Adapter IE> in the device list.
7. Double-click the command "Assign TS Adapter IE parameters". The allocated web interface opens for assigning the parameters of the TS Adapter IE.
8. Complete the "logon" for the web interface.
9. Set the required parameters in the individual tabs of the dialog.
10. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE. Parameter assignment is then complete.

Parameter assignment of the TS Adapter IE using a remote connection

Requirement

There is an established remote connection to a TS Adapter IE .

Procedure

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Open the "TeleService" folder followed by the required system folder.
3. Double-click the command "Assign TS Adapter IE parameters". The associated web interface for assignment of the TS Adapter IE parameters opens. The "logon" for the web interface takes place automatically with the login data of the remote connection.
4. Set the required parameters in the individual tabs of the dialog.
5. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE. Parameter assignment is then complete.

7.3.6 Establishing a dial-up connection to a remote system

7.3.6.1 Establishing a dial-up connection

Introduction to establishing a remote dial-up connection

A dial-up connection is established when you use TeleService to dial into a remote system via a telephone network. The programming device/personal computer is connected to the telephone network with TeleService via a modem. At the other end, the automation system is connected to the telephone line via a configured TS Adapter and a modem.

Requirements

A local modem is installed and configured.

The TS Adapter is located in the remote system.

A remote modem is installed and parameters have been assigned.

Proceed as follows:

1. In the project tree of the TIA Portal, click the "Online access" folder.
2. Click on the "TeleService" folder it contains.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. In the "Adapter type" drop-down list, select the adapter type used.
5. Under "Connection type", select "Dial-up connection".
6. Select the modem you are using under "Local settings".
7. Enter the phone number to be dialed in the appropriate box or open the phone book by clicking on the button behind it and take the desired phone number from the phone book.
8. Enter your user name and the corresponding password.
9. If you want a "Connection setup with callback", select the appropriate option button.
10. Click the "Connect" button to establish the required remote connection. This button only becomes active when you have entered all the parameters needed to establish a remote connection. Any remote connection is displayed under "Status".

Result

The dial-up connection to the desired system is established.

The dialog closes as soon as the dial-up connection is established. The following message appears in the status bar of the TIA Portal: "Remote connection is up". You can now use the remote connection with the TIA Portal and communicate with the automation system.

Connection cannot be established

If the connection cannot be established, try to find the cause using the "Notes on troubleshooting".

Terminating the connection

Once you have finished editing the remote system, exit the remote connection in the project tree by double-clicking on the entry "Establish/disconnect remote connection".

By exiting the TIA Portal, you are also terminating the remote connection.

7.3.6.2 Terminating a dial-up connection

Terminating an active dial-up connection

Note

Terminating the dial-up connection

You should go offline in the TIA Portal before you terminate the remote connection.

Proceed as follows:

1. Double-click the "Set up/close remote connection" entry in the TIA Portal.
2. Confirm the prompt in the dialog that follows with "Yes".

Result

The connection is terminated.

7.4 Remote VPN connections

7.4.1 Basics for establishing a VPN connection

Using a TS Adapter IE Advanced for VPN connections

A TS Adapter IE Advanced is required to establish a VPN connection using TeleService.

VPN definition

VPN stands for "Virtual Private Network". This is a private network of computers based on a public network infrastructure.

VPN is used to create a connection that is as secure as possible between devices in a private network at different locations and the gateway of another network.

A VPN device has direct access to the gateway in the other network over an encrypted connection.

VPN connections

When a VPN connection is established with TeleService, a CA certificate and a unique fingerprint generated from it are used for the unique identification of the TS Adapter IE Advanced. The VPN connection allows a communication that is secure from tapping and manipulation between the remote PC and the TS Adapter.

If you want to establish a VPN connection to multiple remote networks, you will require a separate TS Adapter IE Advanced as an interface for each network. The networks connected by the TS Adapter can used identical IP addresses internally. Only the external IP addresses (WAN) of the TS Adapters need to be different.

Only one VPN connection to a TS Adapter is possible at any one time.

Note

Authentication

Please not that the CA certificate exclusively authenticates the TS Adapter IE Advanced.

As in the case of dial-up connections, users are authenticated via the login (user name and password).

Therefore, use only a secure password for the login.

See also

- Establishing VPN connections (Page 166)
- Terminating VPN connections (Page 167)

7.4.2 Basics of CA certificates

Introduction

To establish a secure VPN connection with TeleService, you need to generate a CA certificate with a unique fingerprint when you configure the TS Adapter IE Advanced.

You can install this CA certificate on every PC that should have access to the TS adapter IE advanced.

- Using the automatic download in the TeleService connection dialog, by entering the fingerprint for the CA certificate
- Manually by using the Microsoft® Management Console.

Definition of CA certificate

A CA certificate is a digital certificate issued by a "certificate authority" or "certification authority", hereinafter referred to as "CA". In the case of the TS Adapter IE Advanced, self-signed certificates are used; the certification authority in this case is the TS Adapter IE Advanced itself.

Certificates for "SSTP" (Secure Socket Tunneling Protocol) and "HTTPS" (Hypertext Transfer Protocol Secure) are derived from the CA certificate.

CA certificates contain a "key" and additional information used for authentication and for encrypting and decrypting confidential data. Additional information includes the validity period, references to certificate revocation lists, etc. which are included in the certificate by the CA.

Using CA certificates with TeleService

A CA certificate with a unique fingerprint is generated by the TS Adapter to uniquely identify the TS Adapter IE Advanced as connection partner for the remote PC.

This CA certificate must be saved in the Windows certificate store of your programming device/PC before you can establish a VPN connection. When you access the Web server using a

direct connection, a security warning will appear if no CA certificate exists. In such cases, however, you can choose to ignore the warning and allow the connection.

Note

Handling CA certificates

Specialist knowledge of the operating system is required for handling CA certificates. CA certificates should only be managed by trained personnel!

You require administrator rights to manage CA certificates.

Definition of fingerprint

The fingerprint is a 20-byte hexadecimal expression. It provides a unique value for a CA certificate and is used to identify a specific CA certificate.

A fingerprint is calculated dynamically using the SHA-1 algorithm and is not physically contained in the CA certificate.

Use of fingerprints with TeleService

The CA certificate is used to uniquely identify the TS Adapter IE Advanced as connection partner. A unique 20-byte fingerprint of this certificate is generated each time a CA certificate is generated by the TS Adapter IE Advanced. It is calculated automatically by the TS Adapter IE Advanced when the certificate is generated. Each certificate has a specific, unique fingerprint. This fingerprint must be transferred securely to your computer, for example by phone or in an encrypted e-mail. You need to enter this fingerprint in the connection dialog when establishing a VPN connection with TeleService, unless the CA certificate is already saved on your PC in the Windows certificate store.

You will find the fingerprint for your TS Adapter IE Advanced in the web interface for the TS Adapter IE Advanced. To open the web interface, double-click the command "Assign TS Adapter IE parameters" in the device list in the TIA Portal. Perform the Web login to view the fingerprint in the "Security > Certificates" tab.

CA certificate download during connection establishment

When the connection is established, TeleService checks whether a suitable CA certificate is installed in the Windows certificate store of your programming device/PC. If an appropriate certificate is found, the VPN connection is established as an SSTP connection (Secure Socket Tunneling Protocol).

If no appropriate CA certificate is found, the CA certificate is first loaded by the corresponding TS Adapter IE Advanced. The TS Adapter IE Advanced is called by the remote address entered in the connection dialog. If this download of the CA certificate is successful, the fingerprint of the downloaded CA certificate is calculated and compared with the fingerprint entered in the connection dialog. If the two fingerprints match, a dialog opens and you are prompted to save the CA certificate in the Windows certificate store of your programming device/PC. You require administrator rights to save the CA certificate.

The VPN connection is then established.

See also

- Installing CA certificates for VPN connections (Page 163)
- Deleting CA certificates for VPN connections (Page 165)

7.4.3 Installing CA certificates for VPN connections

Installing CA certificates

To establish a secure VPN connection between your programming device/PC and a remote system with TeleService, you require a valid CA certificate generated by the TS Adapter IE Advanced. This must be saved in the Windows certificate store of your programming device/PC.

A CA certificate can be installed manually or using an automatic download.

CA certificates are managed with the Microsoft® Management Console.

Note

Handling CA certificates

Specialist knowledge of the operating system is required for handling CA certificates. CA certificates should only be managed by trained personnel!

You require administrator rights to manage CA certificates!

Requirement

A CA certificate has yet to be installed in the Windows certificate store on your computer.

Installing CA certificates using the automatic download

Proceed as follows:

1. Log on to the system as administrator.
2. Transfer the fingerprint for the CA certificate from the TS Adapter IE Advanced to your computer using a "secure route", for example by phone or in an encrypted e-mail.
   You will find the fingerprint for your TS Adapter IE Advanced in the web interface for the TS Adapter IE Advanced. To open the web interface, double-click the command "Assign TS Adapter IE parameters" in the device list in the TIA Portal. Perform the Web login to view the fingerprint in the "Security > Certificates" tab.
3. In the project tree of the TIA Portal, click the "Online access" folder.
4. Click on the "TeleService" folder it contains.

5. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
6. Select the "TS Adapter IE" as Adapter type from the drop-down list.
7. Under "Connection type", select "VPN".
8. Enter the IP address / DNS name for the TS Adapter IE Advanced to be contacted in the relevant field. Alternatively, you can apply any existing data from the phone book by clicking the corresponding button.
9. Enter your user name and the corresponding password.
10. Copy the fingerprint displayed in the web interface for the TS Adapter IE Advanced to the "Fingerprint" column.
11. Click the "Connect" button to establish the required remote connection. This button only becomes active when you have entered all the parameters needed to establish a remote connection.
12. No valid CA certificate is found, because no CA certificate has been installed on your PC yet.
    A "normal" connection (not a VPN connection) is therefore established and the certificate required is downloaded from the TS Adapter IE Advanced to the working memory of your computer. The fingerprint is then calculated (SHA-1 algorithm) and compared with the fingerprint entered in the connection dialog. If the two fingerprints match, a dialog opens and you are prompted to save the CA certificate in the Windows certificate store of your programming device/PC.
13. Confirm that you want to save the CA certificate.

Note

Restart of STEP 7 required after renewal of CA certificate

STEP 7 must be restarted after renewal of the CA certificate on the TS Adapter IE Advanced before the remote connection can be established with the help of the new fingerprint. Otherwise, it may not be possible to store the CA certificate in the Windows certificate memory and the remote connection cannot be established.

Result

The VPN connection to the required TS Adapter IE Advanced is established. The dialog closes once the connection is established.

Manual installation of CA certificates

Proceed as follows:

1. Log on to the system as administrator.
2. Open the Windows Certificate Manager on your programming device/PC with the Microsoft® Management Console.
   Click "Start", enter "mmc" in the search field and press ENTER.
   The console will open.

3. Open the "File" menu and click "Add/remove snap-in...".
   The "Add/remove snap-in..." dialog will open.
4. Double-click "Certificates" in the "Snap-in" list and select "Computer account" in the dialog that opens.
5. Select "Local computer" in the next dialog and click "Finish" and "OK".
   The console root opens and the "Certificates (local computer)" folder appears.
6. Open the displayed "Certificates (local computer)" folder and click "Trusted certification authorities".
7. Click the "Certificates " folder and use the shortcut menu to access the command "All tasks" > "Import...".
8. Note the information that appears in the "Certificate - Import wizard" dialog and click "Next".
9. In the dialog that follows, click "Browse ..." and select the required CA certificate.
10. Then click "Next" twice and subsequently "Finish" to install the CA certificate.

Result

The selected CA certificate is installed at the specified location in the Windows certificate memory.

Note

Additional information ...

... on installing CA certificates can be found using the "F1" button in the online help of your operating system.

7.4.4 Deleting CA certificates for VPN connections

Deleting CA certificates

Proceed as follows:

1. Log on to the system as administrator.
2. Open the Windows Certificate Manager on your programming device/PC with the Microsoft® Management Console.
   Click "Start", enter "mmc" in the search field and press the ENTER KEY.
   The console will open.
3. Open the "File" menu and click "Add/remove snap-in...".
   The snap-in selection dialog will open.
4. Double-click "Certificates" in the "Snap-in" list and select "Computer account" in the dialog that opens.

5. Select "Local computer" in the next dialog and click "Finish" and "OK".
   The console root opens and the "Certificates (local computer)" folder appears.
6. Open the displayed "Certificates (local computer)" folder and click "Trusted certification authorities".
7. Open the "Certificates" folder, select the required CA certificate and click "Delete" in the shortcut menu.
8. Confirm the next prompt with "Yes".

Result

The selected CA certificate is deleted from the list of available certificates.

7.4.5 Establishing a VPN connection to a remote system

7.4.5.1 Establishing VPN connections

Introduction to establishing VPN connections

A VPN connection is established when you use TeleService to connect to a remote system over the Internet.

Your programming device/PC with installed TIA Portal is connected to the Internet at one end for this. At the other end, the automation system is connected to the Internet via the WAN (Wide Area Network) interface of the configured TS Adapter IE Advanced.

Requirement

Your programming device/PC is connected to the Internet.

There is a TS Adapter IE Advanced in the remote system.

The TS Adapter IE Advanced has been configured and is connected to the Internet.

The CA certificate required for identifying the TS Adapter has been generated and installed in the Windows certificate store of your programming device/PC.

Proceed as follows:

1. In the project tree of the TIA Portal, click the "Online access" folder.
2. Click on the "TeleService" folder it contains.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. Select the "TS Adapter IE Advanced" as "Adapter type" from the drop-down list.
5. Enter "VPN" as "connection type".

6. Enter the IP address / DNS name for the TS Adapter IE Advanced to be contacted in the relevant field. Alternatively, you can apply any existing data from the phone book by clicking the corresponding button.
7. Enter your user name and the corresponding password.
8. Click the "Establish" button to establish the required VPN connection. This button only becomes active once you have entered all the parameters needed for establishing the remote connection.

Result

The VPN connection to the required system is established. "Status" indicates the progress in establishing the connection. The dialog closes once the VPN connection is established. The following message appears in the status bar of the TIA Portal: "Remote connection is up". You can now use the remote connection with the TIA Portal and communicate with the automation system.

Connection cannot be established

If the connection cannot be established, try to find the cause using the "Notes on troubleshooting".

Note

Rules for IP addresses

- Only use IP addresses which have not yet been assigned in the plant network.
- If the IP address configured for the TS Adapter IE Advanced has already been assigned in the system network, the TS Adapter IE Advanced can only be addressed by its MAC address.

See also

Installing CA certificates for VPN connections (Page 163)

7.4.5.2 Terminating VPN connections

Terminating an active VPN connection

Note

Terminating the VPN connection

You should go offline in the TIA Portal before you terminate the VPN connection.

Proceed as follows:

1. Double-click the "Set up/close remote connection" entry in the TIA Portal.
2. Confirm the prompt in the dialog that follows with "Yes".

Result

The VPN connection is terminated.

7.4.6 TS Adapter IE Advanced

7.4.6.1 Short description of the TS Adapter IE Advanced

TS Adapter IE Advanced

The TS Adapter IE Advanced has the following properties:

- Direct connection over Industrial Ethernet (IE), 2 ports
- WAN (Wide Area Network) interface for VPN connections
- Firmware update supported
- Plug-in modules
- Parameters are assigned via a Web interface.

Note

Further information on the TS Adapter IE Advanced

For detailed information on your TS Adapter, please refer to the documentation supplied with your TS Adapter.

7.4.6.2 Connection types

TS Adapter IE Advanced connection types

The following diagrams show the types of connection possible with the TS Adapter IE Advanced.

Direct connection

In the direct connection to the programming device/PC, you can set the TS Adapter IE Advanced parameters over the Ethernet.

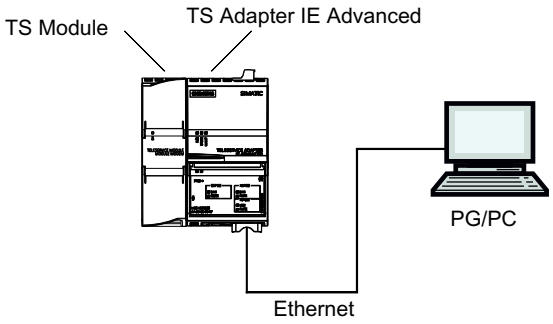

Figure 7-5 TS Adapter IE Advanced - direct connection

Connection to the Internet (DSL modem/router)

In order to connect to the Internet, you must operate the DSL modem/router at the WAN connection of the TS Adapter IE Advanced.

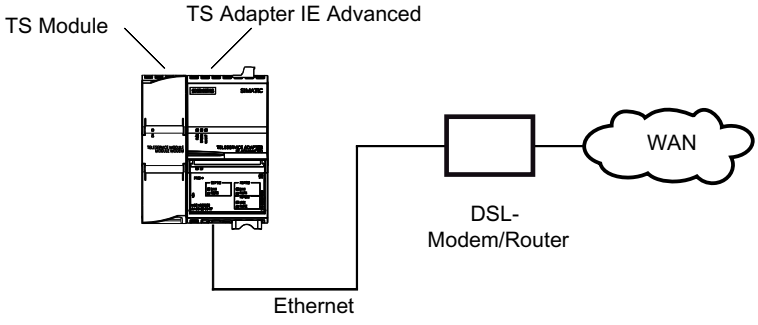

Figure 7-6 TS Adapter IE Advanced - connection to the Internet

Connection to the company network (Intranet)

In order to connect to the Intranet, you must operate plant network (Ethernet) at one of the two LAN connections of the TS Adapter IE Advanced.

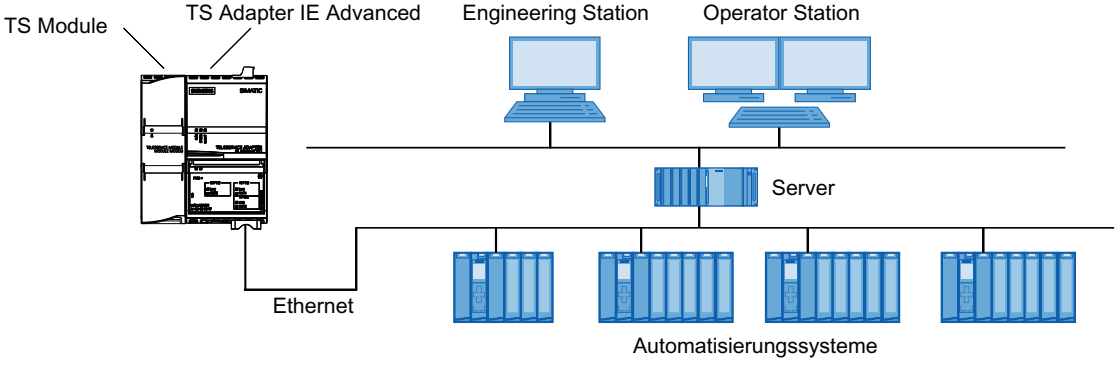

Figure 7-7 TS Adapter IE Advanced - connection to company network (Intranet)

Additional information

For more detailed information on the TS modules, please refer to the documentation supplied with your TS Adapter IE Advanced.

7.4.6.3 TS Adapter IE Advanced parameter assignment options

Basics on configuring the TS Adapter IE

The TS Adapter IE Advanced is configured via a Web interface.

Web help in the Web interface is available for assigning TS Adapter IE Advanced parameters.

Parameter assignment options include:

- Reconfiguration
- Restoring default parameter assignment
- Importing adapter parameters
- Exporting adapter parameters

Note

Parameter assignment

Configure your TS Adapter IE Advanced in accordance with the documentation supplied. This details the exact procedure for parameter assignment.

7.4.6.4 Parameter assignment for TS Adapter IE Advanced

Introduction

The TS Adapter IE Advanced can be configured in both direct connection mode and via an existing remote connection.

Both parameter assignment options are described below.

Specific details on assigning parameters for the TS Adapter IE Advanced are available in the TS Adapter IE Advanced documentation.

Assigning parameters for the TS Adapter IE Advanced in direct connection

Requirement

There is a LAN connection to your TS Adapter IE Advanced.

Procedure

To assign parameters for the TS Adapter IE Advanced, follow the steps below:

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Double-click on the Ethernet port of your computer.
3. Double-click on the "Display accessible nodes" command. The TS Adapter IE Advanced is then displayed.
4. Double-click on the "TS Adapter IE Advanced" folder and then on "Online and diagnostics", and assign the desired IP address to the TS Adapter in the dialogs that follow. Please note that the IP address of the programming device/PC interface card must be located in the same subnet as the IP address that you assign for the TS Adapter IE Advanced.
5. Update the view in the project tree for the "Accessible nodes", so that the TS Adapter IE Advanced is displayed with the newly assigned IP address.
6. Open the folder "TS Adapter IE Advanced" in the device list.
7. Double-click the command "Assign TS Adapter IE parameters". The corresponding web interface opens for assigning TS Adapter parameters.
8. Complete the "logon" for the web interface.
9. Set the required parameters in the individual tabs of the dialog.
10. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE Advanced. Parameter assignment is then complete.

Assigning parameters for the TS Adapter IE Advanced using a remote connection

Requirement

There is an established remote connection to a TS Adapter IE Advanced..

Procedure

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Open the "TeleService" folder followed by the required plant folder.
3. Double-click the command "Assign TS Adapter IE parameters". The associated web interface for assignment of the TS Adapter IE parameters opens. The "logon" for the web interface takes place automatically with the login data of the remote connection.
4. Set the required parameters in the individual tabs of the dialog.
5. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE Advanced. Parameter assignment is then complete.

7.5 CPU controlled TeleService remote connections

7.5.1 Overview of CPU controlled remote connections

Introduction

TeleService offers a range of options for establishing remote connections; these differ according to the CPU used. The initiative for establishing a connection starts from the CPU. The communication instructions given below are used for the individual connection options.

Connection establishment with S7-300/400 CPUs

The following communication instructions are available:

- Communication instruction "PG_DIAL": Establish remote connection to programming device/PC
- Communication instruction "SMS_SEND": Send text message (SMS)
- Communication instruction "AS_DIAL": Establish remote connection to AS
- Communication instruction "AS_MAIL": Transfer email

Connection establishment with S7-1200 CPUs

The following communication instruction is available:

- Communication instruction "TM_MAIL": Transfer email

Connection establishment with S7-1500 CPUs

The following communication instruction is available:

- Communication instruction "TMAIL_C": Transfer email

Note

Description of individual communication instructions

More detailed information on the available communication instructions can be found in the information system of the TIA Portal in the "References > Communication > TeleService" directory.

See also

TS Adapter IE parameter assignment options (Page 155)

7.5.2 Establishing a connection from and to remote systens (PG-AS-remote coupling)

7.5.2.1 Remote plant access to a programming device/personal computer

Introduction

You can establish a remote connection to and communicate with a remote system using the application TeleService and a TS Adapter MPI. The initiative for establishing the remote connection comes from the programming device/personal computer.

However, events which require rapid intervention often occur at a remote system. In such cases, the automation system can initiate a remote connection to a programming device/ personal computer if an asynchronous event occurs.

The graphic below shows the components which are required for establishing a connection from a plant to a programming device/personal computer.

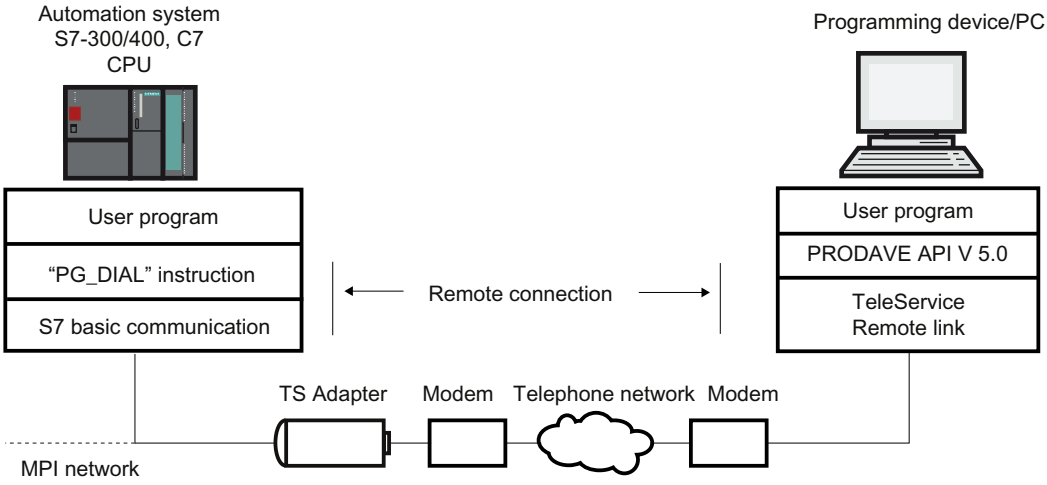

Figure 7-8 How the "PG_DIAL" communications instruction works

7.5.2.2 Requirements for establishing a connection

Introduction

Certain hardware and software requirements must be fulfilled if a remote system is to establish a remote connection to a programming device/personal computer. These requirements are detailed below.

Hardware requirements:

The only hardware required for establishing a remote connection from a remote system to a programming device/personal computer is that needed for accessing the remote system from the programming device/personal computer.

Your user program calls the "PG_DIAL" communication instruction to establish the connection. This can only be executed on an S7-300 or S7-400 CPU on which S7 basic communication is implemented.

A TS Adapter I , version 5.0 or later, or a TS Adapter II must be used.

Software requirements on the system side:

The "PG_DIAL" communication instruction is included in the scope of delivery of TeleService and is installed when the TIA Portal is installed. You will find the installed communications instructions in the task card of the block editor in the folder "Communication > TeleService".

If a remote system is to establish a remote connection to a programming device/PC, the user program of the system must call the "PG_DIAL" communication instruction.

Software requirements on the programming device/PC side:

You require a software product on the programming device/PC which, with TeleService, waits for a call from a remote system, recognizes this call and informs your user program.

7.5.3 Data exchange between remote systems (AS-AS-remote coupling)

7.5.3.1 AS-AS remote link basics

Introduction

The AS-AS remote link allows two automation systems to exchange process data via the telephone network.

Requirement

Communication instruction "AS_DIAL" is available if you use a CPU from the S7-300/400 family.

Definition: Local and remote automation system

- The automation system from which the initiative to establish the remote connection originates is described as **local**.
- The automation system to which the remote connection is to be established is described as **remote**.

Data exchange over the AS-AS remote link

Data exchange is carried out using specific communication instructions for non-configured S7 connections. Use the communication instruction "AS_DIAL" to establish a remote connection to the automation system.

More detailed information on establishing the connection can be found in the information system in the directory "References > Communication > TeleService".

The following graphic shows the components required for establishing a connection from a local to a remote automation system.

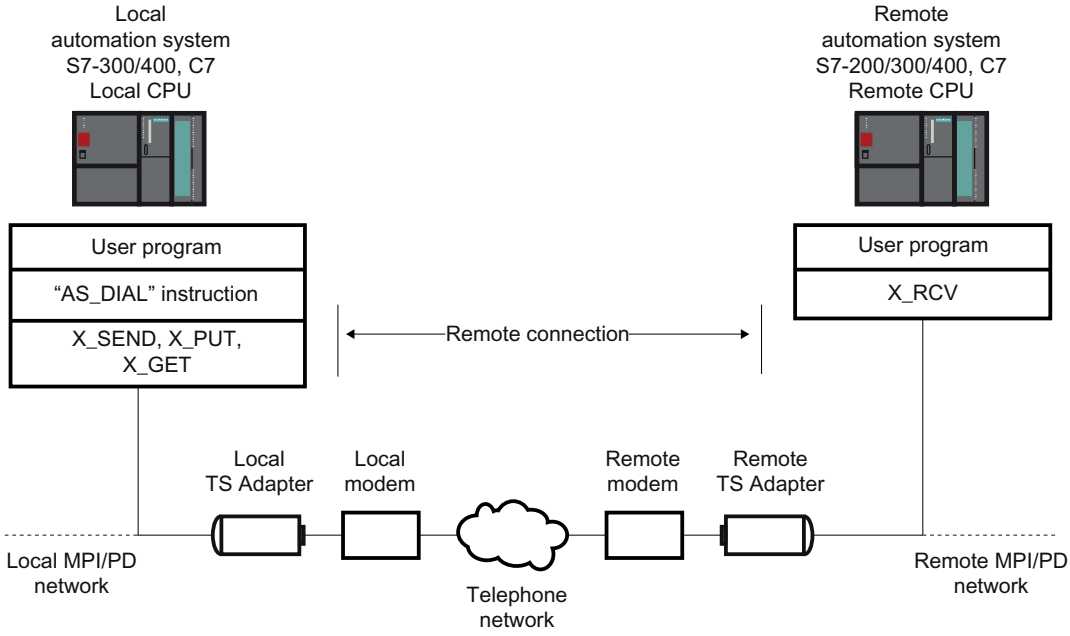

Figure 7-9 Data exchange over the AS-AS remote link

7.5.3.2 Hardware and software requirements for AS-AS remote link

Introduction

Certain hardware and software requirements must be fulfilled before a local automation system can establish a remote connection to a remote automation system. These requirements are detailed below.

Hardware requirements

The only hardware you need for transferring process data from a local to a remote automation system is that also needed for accessing the respective automation system from the programming device/personal computer.

To establish and terminate the remote connection, the user program of the TIA Portal of the local CPU calls a communication instruction. This communication instruction can be executed on an S7-300/400 CPU or a C7 CPU. The communication instruction requires S7 basic communication to be implemented on the CPU. The remote CPU must also support S7 basic communication.

A TS Adapter I, version V5.1 or later, or a TS Adapter II must be used.

Software requirements

The "AS_DIAL" communication instruction is included in the scope of delivery of TeleService. When installed, the instruction is integrated in the library of the TIA Portal in the task card of the Communication instructions folder under TeleService. In order to establish and terminate a remote connection to a remote automation system from a local automation system, the "AS_DIAL" communication instruction must be called in the user program of the TIA Portal on the local CPU.

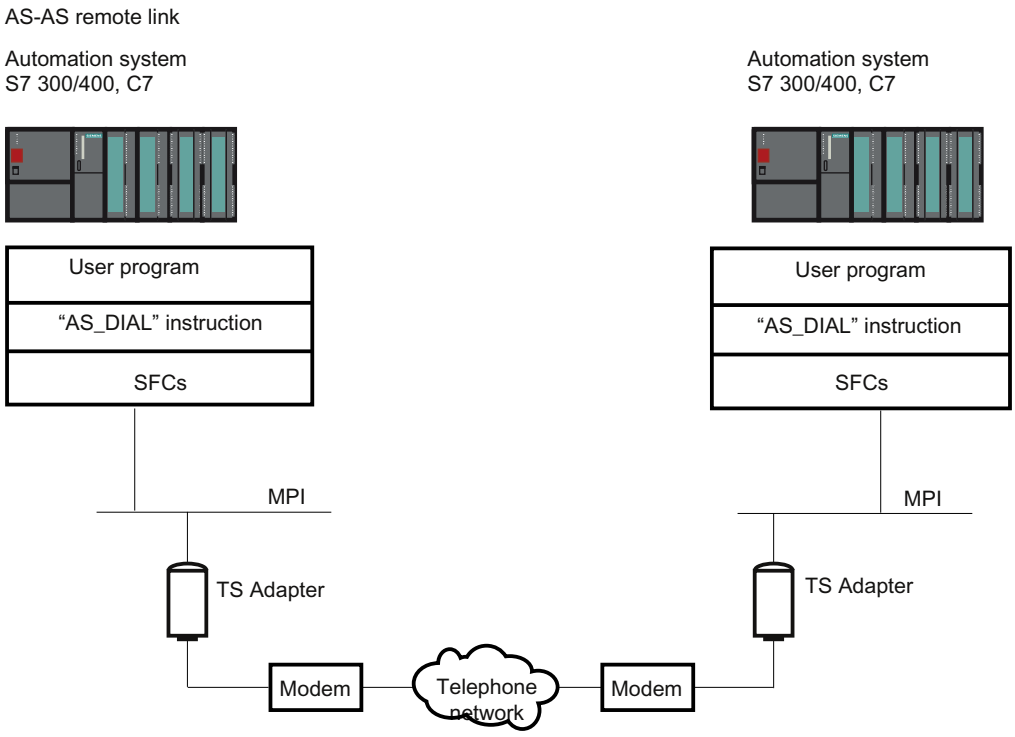

Figure 7-10 Hardware and software requirements for AS-AS remote link

7.5.4 Send SMS from a system

7.5.4.1 Requirements for sending an SMS

Introduction

Certain hardware and software requirements must be fulfilled before a system can send an SMS. These requirements are detailed below.

Hardware requirements

To send an SMS from a system, you will require a GSM wireless modem and a TS Adapter MPI.

A TS Adapter I, version V5.2 or later, or a TS Adapter II must be used.

Software requirements on the system side

The "SMS_SEND" communication instruction is included in the scope of delivery of TeleService. When installed, the instruction is integrated in the library of the TIA Portal in the task card of the Communication instructions folder under TeleService. If a system is to send an SMS, the user program of the system must call the "SMS_SEND" communication instruction.

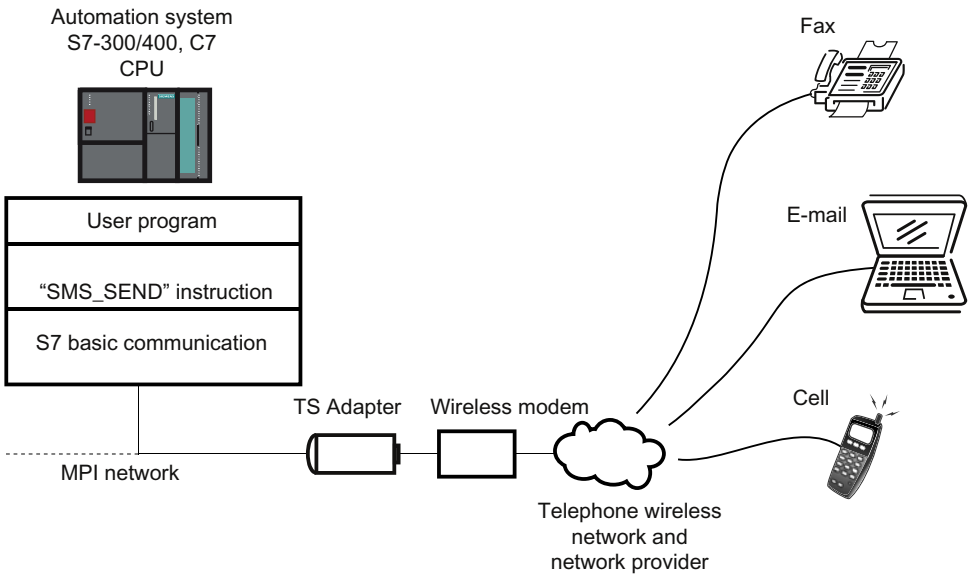

Figure 7-11 How the "SMS_SEND" communication instruction works

Note

You may be able to send an SMS not only to a cell phone but also to an email address or a fax device by using additional services offered by the cell phone service provider.

7.5.5 Send an email from a system

7.5.5.1 Requirements for sending e-mails

Introduction

The following hardware and software requirements must be fulfilled if a system is to send an email:

Hardware requirements

You need a TS Adapter IE to send an email from a system and one of the CPUs listed below:

- a CPU 31x2 PN/DP as of firmware version V2.5
- a CPU 41x-3 PN/DP
- a CPU of the S7-1200 series
- a CPU of the S7-1500 series

Software requirements on the system side

Various communication instructions are included in the scope of delivery of TeleService depending on the CPU. When installed, they are integrated in the library of the TIA Portal in the task card of the Communication instructions folder under TeleService.

If a system is to send an email, the user program of the system must call the corresponding CPU-dependent communication instruction.

The following communication instructions are available for sending an email:

- S7-300/400 CPUs: Uses the communication instruction "AS_MAIL": Transfer email
- S7-1200 CPU V2.x and V3.x: Uses the communication instruction "TM_MAIL": Transfer email
- S7-1200 CPU >V4.0: Uses the communication instruction "TMAIL_C": Transfer email
- S7-1500 CPUs: Uses the communication instruction "TMAIL_C": Transfer email

The respective communication instruction transfers an email from a CPU to a mail server by means of the Simple Mail Transfer Protocol (SMTP) with the "LOGIN" authentication method. The data is transferred unencrypted with this SMTP process.

The figure below shows an example with the "AS_MAIL" communication instruction:

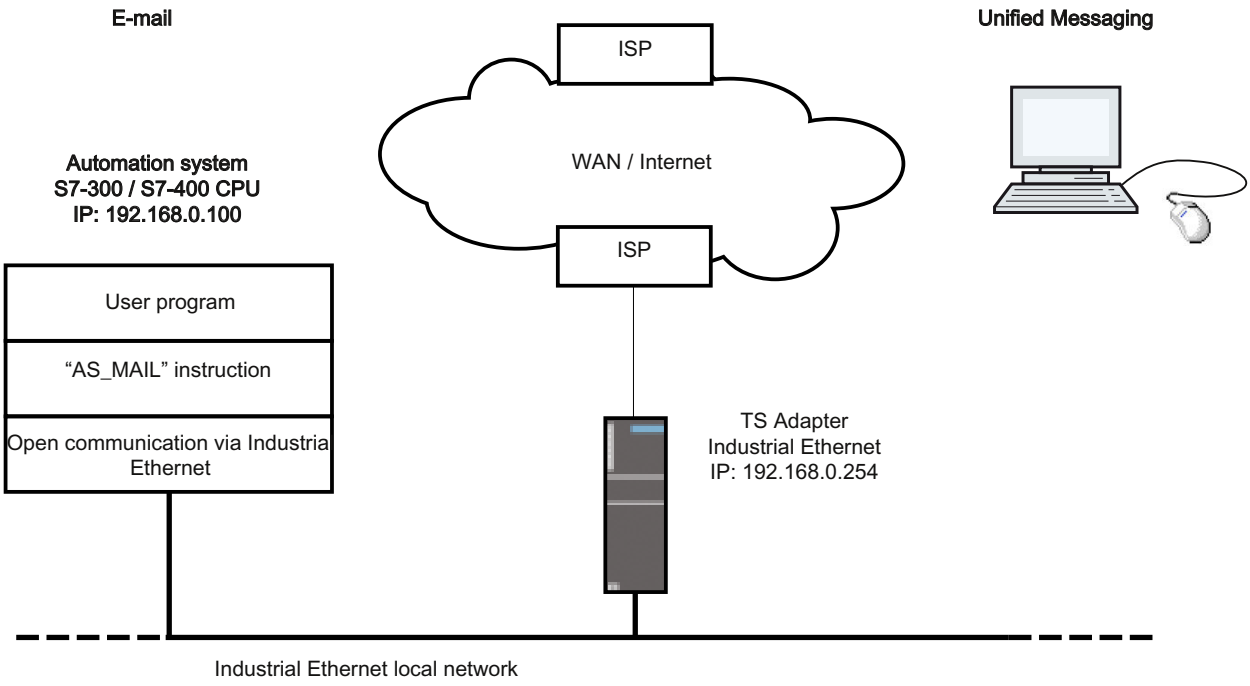

The "Use gateway/router" property must also be set for the Ethernet interface in the configuration of the CPU on which the "AS_MAIL" communication instruction runs. (available in the device configuration under Ethernet addresses and there under IP protocol)The IP address of the Ethernet interface of the TS Adapter IE is to be specified as "Address".

Note

You can find further information in the task card in the "Communication instructions" folder under TeleService.

7.6 Notes on troubleshooting

7.6.1 General information on troubleshooting for modem problems

Introduction

The information below should help you establish and eliminate the causes of any modem problems.

1. Enable "Record log file" for data traffic between PG/PC and modem. The entries in this file can provide valuable information for determining the cause of errors.
2. Switch on the loudspeaker on your local modem. Select a volume loud enough to be clearly heard.
   You can then hear whether:
   – there is a dial tone at the connection
   – the modem called is busy
   – the modem called accepts the call.

Common modem problems

Modem connection problems are among the most common modem problems:

- Modem connection is not established
- Modem connection is interrupted

The topics below contain tables detailing possible causes and providing information on eliminating the error in question.

See also

Dial-up connection to the TS Adapter is not established (Page 182)

Dial-up connection from the TS Adapter is not established (Page 184)

Modem connection is interrupted (Page 185)

Modem alarms (Page 186)

Recording a log file for the modem (Page 181)

7.6.2 Recording a log file for the modem

Introduction

It is advisable to record a log file as this makes it easier to find the causes of faults in a modem.

Procedure:

Proceed as follows:

1. Activate the properties dialog of the modem used via the "Phone and modem options" option in the Control Panel.
2. Check the settings of the "Log" option in the "Diagnostics" tab and, if necessary, change the log file settings so that the file is recorded.

Result:

Activity between the programming device/personal computer and the modem are entered in the log file. If there are problems establishing the connection, you can evaluate the data recorded in the log file to determine the cause of the error.

7.6.3 Dial-up connection to the TS Adapter is not established

Dial-up connection to TS Adapter is not established

The table below sets out possible causes and how to eliminate them if no remote connection to the TS Adapter can be established.

Possible cause	Check/Remedy
Cabling faulty	• Are all the connecting cables connected correctly? • Are the connectors loose?
Dial parameters for main telephone line and extension incorrectly set	• Do the properties and dial parameters of your modem match the phone connection to main phone line or extension? • Do not specify a dial-out code in the "Dial parameters" dialog if you operate your modem on a local loop (main telephone line). The fields for the dial-out code for local calls and long-distance calls must be empty.
Dialing mode incorrectly set	• Is the correct dialing mode (tone/pulse) set in the dialog for the dial parameters of your modem? • Use a connected telephone to check the connection on which you want to operate the modem. You should hear crackling noises on the telephone during pulse dialing and tones of varying pitches during tone dialing. Set the corresponding dialing mode in the modem dial parameters.

Possible cause	Check/Remedy
Dial disable active	• The dial disable function is a country-specific modem property which, depending on the modem, becomes effective after one or more unsuccessful attempts to establish a connection. • If your modem still does not respond after several attempts to dial, the dial disable function may be active. Characters are still sent to the modem after the dial command but the modem does not start the dialing process. The driver receives a general error message. • Refer to the modem documentation for information on how the dial disable function is implemented for your modem. • Create a log file (Page 181) (modemlog.txt) in which the activities between the programming device/personal computer and modem are recorded. Then check whether the file contains an entry caused by dial disable (e.g. DELAYED).
Phone connection defective or busy	• Connect a phone and check whether a dial tone can be heard on this connection. • Any analog phone connected on the same connection must be hung up. You cannot establish an additional modem connection on this connection if there is an existing phone connection.
Serial parameters set incorrectly	• Are the correct values entered in the "Settings" tab for modem properties (8 data bits, no parity, 1 stop bit)? Is the correct COM interface set in the "General" tab for the modem properties?
Initialization string of the TS Adapter is not suitable for the modem.	• Familiarize yourself with the modem initialization string requirements and set the string accordingly. Procedure for configuring the TS Adapter IE (Page 153)
Settings for error correction between the modem at the TS Adapter and the modem at the PC/programming device are not compatible.	• Adapt the modem settings. Useful information on configuring the TS Adapter MPI (Page 148) Restoring the default parameter assignment of a TS Adapter MPI (Page 150) Procedure for configuring the TS Adapter IE (Page 149)

7.6.4 Dial-up connection from the TS Adapter is not established

No callback from TS Adapter

The table below sets out possible causes and how to eliminate them if there is no callback from the TS Adapter.

Possible cause	Check/Remedy
Errors in the location or call settings in the TS Adapter	Check the TS Adapter parameter assignment: • Are the dialing mode and dial-out code set correctly for your phone connection? • Does the modem at the TS Adapter support the characters configured for the dial-out code? • Is "Wait for dial tone before dialing" deactivated for an extension?
Initialization of modem insufficient	Check the string for modem initialization: • The modem may require a further initialization in order to establish a remote connection. • Properties of the modem initialization string for the TS Adapter MPI
Callback number is incorrect	Check the configuration of the callback number you assigned.

No call from TS Adapter MPI

The table below sets out possible causes and and how to remedy them if there is no call from the TS Adapter MPI.

Possible cause	Check/Remedy
Phone number is incorrect	Is the required number being transferred to the communication instruction "PG_DIAL" ?
TS Adapter MPI parameter assignment incorrect	Check the TS Adapter MPI parameter assignment: • Are the dialing mode and dial-out code set correctly for your phone connection? • Does the modem at the TS Adapter MPI support the characters configured for the dial-out code? • Is "Wait for dial tone before dialing" deactivated for an extension?

7.6.5 Modem connection is interrupted

Modem connection is interrupted

The table below sets out possible causes and and how to remedy them if the modem connection is interrupted.

Possible cause:	Check / Remedy:
Metering pulses in the line	Metering pulses will be generated if you have applied to the phone company for a metering clock. This may mean that the modem no longer recognizes the carrier signal and switches off. • Set a longer waiting or switch-off time at the modem. • Have the metering pulse deactivated by the phone company.
Shielding	• Are the connection cables used shielded sufficiently? • Make sure that the modem cables do not run next to power cables and that they are as far as possible from power supply units and monitors.
Protocol timeout	• Set fixed monitoring times.
Automatic connection termination	• Deactivate the option that terminates an existing connection automatically after a specified time without data transfer ("Terminate after idle of ...").
Data flow control deactivated	• Click on the "Extended" button in the "Settings" tab of the modem properties and activate the following options in the dialog displayed (if available and not yet set): – Data flow control – Hardware (RTS/CTS) – Data compression – Error control
Initialization string of the TS Adapter is not suitable for the modem	• Set the modem initialization string in accordance with the following requirements. See also: TS Adapter IE parameter assignment options (Page 155)

See also

TS adapter MPI configuration options (Page 148)

7.6.6 Checklist for troubleshooting the modem

Introduction

The following list should help you establish the potential cause of any problems with the modem. The help topics below set out how and in which dialogs you define the relevant settings.

Modem connection cannot be established:

- Check the cabling and the connections.
- Check whether the correct dialing mode (tone/pulse) is set.

- If your modem does not react after several attempts to dial, a dial disable function may be active. Familiarize yourself with dial disable on your modem.
- Are you operating your modem on a main telephone line or on an extension line? Configure the properties and dialing parameters of the modem accordingly.
- Enable the log file option in the advanced properties. The next attempt to establish a connection will then be recorded in a file in the Windows directory.
- Ensure that the ISDN TAs used work with the same B and D channel protocol.

The modem connection is terminated:

- Metering pulses can have a negative affect on a connection. Have the pulses deactivated by your telephone company.
- Set fixed monitoring times.
- Deactivate the option that terminates an existing connection automatically after a specified time without data transfer (idle).
- Make sure that you have activated RTS/CTS for data flow control.

7.6.7 Modem alarms

Information in the log file

The modem alarms are entered in a log file if you have activated the recording function.

The log file contains the following information:

Alarm:	Possible cause:	Remedy:
NO DIALTONE	A phone call may currently be being carried out on this line.	• Repeat the process once the phone call is over.
NO CARRIER	The device dialed is not ready, is not a modem or cannot establish a connection in the set operating mode.	• Check the numbers and the settings.
BUSY	The device dialed is busy.	• Try again later.
DELAYED: ...	Dial disable	• Refer to the modem documentation for information on how the dial disable function is implemented for your modem and if necessary remove it.

7.6.8 Possible error messages with VPN connections

VPN connection to TS Adapter IE Advanced is not established

The table below sets out possible errors and how to eliminate them if no VPN connection to the TS Adapter IE Advanced can be established.

Possible errors	Check/Remedy
The error message "The webpage cannot be displayed" appears when the Web interface is accessed.	• Note: This problem occurs when the Windows certificate store contains a CA certificate with the same name as the CA certificate which has just been generated. • If a CA certificate for this TS adapter is already installed in the Windows certificate store and you have generated a new CA certificate on the TS adapter, remove the old CA certificate from the Windows certificate store and install the newly generated CA certificate. • See also: Installing CA certificates for VPN connections (Page 163) • See also: Deleting CA certificates for VPN connections (Page 165)
Error message: "Unexpected error in application".	• Check whether other users have been saved in the TS Adapter IE Advanced user database. • The user "Administrator" cannot establish a remote connection.
Error message: "No corresponding CA certificate was found in the Windows certificate store".	• Use the automatic certificate download. Enter the fingerprint in the corresponding field in the connection dialog. • See also: Establishing VPN connections (Page 166) Or • Export the CA certificate from the Web interface of the TS Adapter IE Advanced and install it manually in the Windows certificate store. • See also: Installing CA certificates for VPN connections (Page 163)
Error message: "The remote address specified does not correspond to the remote address of the TS Adapter".	• You must use the remote address which you specified in the TS Adapter IE Advanced to establish the connection. • You cannot use the IP address if you have entered the DNS name (and vice versa).
Error message: "The signature of the certificate could not be verified".	• If you have generated a new CA certificate on the TS Adapter IE Advanced, you need to delete the old CA certificate from the Windows certificate store and install the new one. • See also: Deleting CA certificates for VPN connections (Page 165) • See also: Installing CA certificates for VPN connections (Page 163)
Error message: "Protocol error"	• Check that the IP address of the service PC is set in the configuration of the TS Adapter IE Advanced (Parameters > Plant network)
Error message: "Connection setup error."	• Check to make sure that the LAN IP-address of the TS Adapter IE Advanced and the IP address of your network cards are in the same subnet. If this is the case, deactivate this network card before you establish the remote connection.
An e-mail cannot be sent.	• Check whether there is a network/Internet connection at the WAN port. • Check whether the mail server is accessible. • Check whether the outgoing connections in the TS Adapter IE Advanced are valid. • Check whether the SMTP port in the firewall is open for outgoing connections.

Simulating devices with S7-PLCSIM 8

8.1 Simulation of devices

Introduction

You can use the TIA Portal to run and test the hardware and software of the project in a simulated environment. The simulation is performed directly on the programming device or PC. No additional hardware is required.

The simulation software provides a graphical user interface for monitoring and changing the configuration. It differs according to the currently selected device.

Integration in the TIA Portal

The simulation software is fully integrated in the TIA Portal but is only supported by certain devices. Therefore, the button for calling the simulation software is only active if the selected device supports simulation.

The simulation software for some devices requires its own virtual interface to communicate with the simulated devices. The virtual interface can be found in the project tree under the "Online access" entry next to the physical interfaces of the programming device/PC.

Once you have opened the software, additional help on the simulation software is available via a separate link.

See also

Starting the simulation (Page 190)

8.2 Starting the simulation

Some devices can be simulated with additional software. You therefore do not have to have the actual devices to perform comprehensive testing of your project.

Procedure

To start the simulation software, follow these steps:

1. Select the device you want to simulate, for example, in the project tree.
2. Select the "Simulation > Start" command in the "Online" menu.
   This calls the simulation software.

See also

Simulation of devices (Page 189)

Index

A

Access protection
    Access protection level 1, 141
    Access protection level 2, 141
    Advantages, 141
    Factory state, 140
    Validity, 140
Automation system
    Local, 175
    Remote, 175

B

Backup from online device, 29, 31
Benefits of using TeleService, 125

C

CA certificate, 161
Callback to a number specified during connection establishment, 143
Changing the configuration online, 11
Changing the device configuration online, 11
Checklist for troubleshooting the modem, 185
Communication instruction
    "PG_DIAL", 175
    "SMS_SEND", 177
Communications instruction
    "AS_DIAL", 175, 176
    "AS_MAIL", 178
    "PG_DIAL", 174
Configuring the remote modem, 139
Connecting a TS Adapter with an external modem, 139
Connecting a TS Adapter with an internal modem, 139
CPU load through trace, 111

D

Data exchange over the AS-AS remote link
    Communication instruction "AS_DIAL", 175
Deleting CA certificates, 165
Designation
    TS Adapter, 137, 145, 153
Dialing rules in phone books, 134
Dial-up connection
    Establishing, 158
    Terminating, 159
Displaying the online status, 19

E

Establishing a connection from a remote system ("PG_DIAL"), 174
Establishing a remote connection, 158
    Procedure, 158
Ethernet interface
    Displaying parameters, 42

G

General information on troubleshooting, 181
Go online, 18
    connecting several devices, 22, 23

H

Hardware requirements
    Communication instruction "AS_MAIL", 179
    Communication instruction "PG_DIAL", 174
    Communication instruction "SMS_SEND", 177
Hardware requirements for AS-AS remote link
    Communication instruction "AS_DIAL", 176

I

Icons
    in TeleService, 129
Installing CA certificate, 163
Installing the local modem, 138
Instructions for configuring a remote modem, 140
Interface
    Adding, 40

L

Log file
    Modem alarms, 186
Logic analyzer function, 58

M

Measurements
Project navigator, 66
Modem
Local, 138
Problem, 181
Remote, 139
Modem alarms
Log file, 186
Modem connection cannot be established, 185
Modem connection is closed, 186
Modem connection is interrupted, 185
Causes and solutions, 185
Modems without plug-and-play, 139

N

Network access
Opening the properties, 39

O

O, 13
Online
Change connection path, 18
Display of the online mode, 19
Establish connection, 18
Go offline, 21
Show accessible devices, 9
Online access, 35
Online connection, 13
Specifying a default connection path, 17
Online mode, 13
Display of the online mode, 15
Go online, 18, 21
Standby or hibernation of the programming device / PC, 13
Opening phone books, 130
Overview of TS Adapters that can be used, 137

P

PG/PC interface
Go online, 21
Modifiable MPI interface parameters, 46
Modifiable PROFIBUS interface parameters, 46
Prerequisites for establishing a VPN connection, 166

R

Recording a log file for the modem, 181
Remote connection
as CPU-controlled remote connection, 173
as dial-up connection, 136
As VPN connection, 160
Remote connection cannot be established, 159
Remote connection from the TS Adapter is not established, 184
Remote connection to the TS Adapter is not established, 182
Remote system access to a programming device/ personal computer, 174
Requirements for establishing a remote connection, 158
Requirements for sending an SMS, 177
Requirements for sending emails, 178
Resetting the MPI/PROFIBUS settings, 53
Resetting the TCP/IP settings, 45
Restoring a backup, 28, 32
Restoring a device, 28, 32

S

Saving phone books, 131
Setting parameters for the Ethernet interface, 43
Adding a temporary IP address, 44
Connecting to a subnet, 42
Deleting dynamic IP addresses, 44
Modifiable parameters, 41
Options in the parameter settings, 41
Settings for the PG/PC interface, 37
Automatic bus parameter detection, 47
Setting parameters for the MPI interface, 48
Setting parameters for the PROFIBUS interface, 49
Show/hide interfaces, 38
Simulate hardware, 189
Simulate modules, 189
Simulate software, 189
Simulating devices, 189
Software requirements for AS-AS remote link
Communication instruction "AS_DIAL", 177
Software requirements on the programming device/ PC side
Communication instruction "PG_DIAL", 175
Software requirements on the system side
Communication instruction "AS_MAIL", 179

Communication instruction "PG_DIAL", 175
Communication instruction "SMS_SEND", 178
Start the simulation, 190

T

TeleService
Access to phone books, 128
AS_DIAL, 175
Callback variants, 141
Communications instruction:"AS_DIAL", 126
Communications instruction:"AS_MAIL", 126
Communications instruction:"PG_DIAL", 126
Communications instruction:"SMS_SEND", 126
Communications instruction:"TM_MAIL", 126
Communications instruction:"TMAIL_C", 126
Connection establishment options:S7-1200 CPUs, 173
Connection establishment options:S7-1500 CPUs, 173
Connection establishment options:S7-300/400 CPUs, 173
Defining dialing rules, 134
Establish connection to AS, 175
Export phone book, 133
Functionality, 126
Gateways, 138
Import phone book, 131
Inserting rows in the phone book, 130
Meaning of the icons, 129
Modem support, 137
Modem types / media, 138
Open phone book, 130
Performance in telephone networks, 138
Phone book, 127
Phone book properties, 127
Printing the phone book, 134
Save phone book, 131
Showing or hiding columns in the phone book, 131
Structure of the phone book, 129
Working with the phone book, 128
TeleService callback options, 143
TeleService phone book, 127
Terminating the remote connection, 159
Trace, 58, (Trigger time)
Align overlay measurements, 108
Analyze an ongoing recording, 98
Apply overlay measurement, 95
Bit track, 71
Configure project navigator into groups, 96
CPU load, 111
Creating a trace configuration, 94
Curve diagram, 71, 102
Data storage, 61
Displaying a diagram, 95
Displaying a trace configuration, 94
Inspector window, 87
Installed measurements (memory card), 61, 67
Installed trace, 60
Keyboard operation, 72
Lifetime of the values, 110
Mathematical functions, 78
Measurement, 60, 61, 99
Measurement cursor, 102
Measurement cursor pane, 84
Monitor fast signals, 106
Mouse wheel, 72
Overlay measurement, 61, 68, 81, 106
Pre-trigger, 116
Printing, 108
Project navigator, 64
Quantity structure, 111
Quick start, 89
Recordable tags, 109
Recording, 60, 97, 98
Recording conditions, 113, 118
Recording cycle, 90, 122
Recording duration, 122
Recording levels, 110
Reduction, 114
Sampling, 90
Save installed measurements, 117, 124
Saving the trace configuration, 101
Signal group, 105
Signal table, 76, 104
Signals, 112, 122
Snapshots pane, 86
Status, 65
Supported devices, 58
Time stamp, 87
Trace configuration, 60, 61, 96, 100, 101, 121
Trigger, 123
Trigger mode, 115
Trigger tag, 115, 118
User interface, 63, 87, 111
Trace function, 58
Trace S7-1200/1500, 109
TS Adapter IE Advanced
Brief description, 168
Connection types, 168
Parameter assignment, 170
Parameter assignment options, 170

TS Adapter MPI
Brief description, 145
Default parameter assignment, 146
Direct connection, 146
Establishing a modem connection, 147
Establishing the direct connection, 146
Exporting adapter parameters, 150
Importing adapter parameters, 151
Modem connection, 147
Parameter assignment, 148, 149
Parameter assignment options, 148
Principle of operation, 145
Restoring the default parameter assignments, 150
Setting up access protection, 142
TS Adapter IE
Brief description, 152
Default parameter assignment, 153
Parameter assignment, 156, 170
Principle of operation, 153
TS Adapter IE Basic
Connection to the GSM network, 155
Connection to the telephone network, 154
Connection to the telephone network through an external modem, 155
Connection types, 153

U

Useful information on configuring the TS Adapter MPI, 148
Using a TS Adapter for TeleService, 136, 160

V

VPN connection
Deleting CA certificates, 165
Establishing, 166
Installing CA certificate, 163
Terminating, 167
VPN connection is not established
Check and remedy, 187

SIEMENS

Processing energy data

Programming Manual

Basics of the SIMATIC Energy Suite 1

Supply energy data 2

Process energy data 3

Buffering and archiving energy data 4

Generate energy program 5

Visualizing and evaluating energy data 6

Energy Suite blocks A

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Basics of the SIMATIC Energy Suite....9
1.1 Energy Management with SIMATIC Energy Suite....9
1.2 SIMATIC Energy Suite operations and components....11
1.3 Procedural overview....15
1.4 Performance features and conventions....19
2 Supply energy data....21
2.1 Important information on energy data....22
2.2 Energy data from measurement hardware....24
2.3 Energy data from the process image or bit memory area of the S7-CPU....27
2.4 Energy data from data blocks....29
3 Process energy data....31
3.1 Energy objects and energy program....31
3.2 Creating an energy object table....33
3.3 Adding energy objects for energy object tables....34
3.4 Setting the properties of the energy objects....36
3.4.1 Configuring an energy object....36
3.4.2 Basic settings....37
3.4.3 Project information....37
3.4.4 Identification & Maintenance....38
3.4.5 Change the enable for energy objects for the energy program....38
3.4.6 Energy data source....39
3.4.6.1 Assign energy data source....39
3.4.6.2 Energy data source....40
3.4.6.3 Energy data type....41
3.4.7 Periodic energy values....44
3.4.7.1 Setting energy values for periodic archiving....44
3.4.7.2 Periodic energy data....45
3.4.8 Processing....47
3.4.8.1 General processing settings....47
4 Buffering and archiving energy data....49
4.1 Defining archiving periods....49
4.2 Archiving periods....51
4.3 Setting the archiving on a S7-CPU (SIMATIC Memory Card)....53
4.4 Archiving and buffering (S7-CPU)....54
4.5 Example for archiving (S7 CPU)....56
4.6 Setting the archiving on a PC station with WinCC Professional (Data Log)....57

4.7 Archiving and buffering (WinCC Professional)....59
4.8 Examples of archiving and buffering (WinCC Professional)....63
5 Generate energy program....65
5.1 Making general settings for the energy program....65
5.2 General settings....66
5.3 Runtime licenses for Energy Suite....67
5.4 Generate energy program....68
5.5 Structure of the energy program....70
5.5.1 Structure of the energy program....70
5.5.2 EnS - block types....71
5.5.3 EnS - energy objects....73
5.5.4 EnS - general program blocks....74
5.5.5 Energy Suite data types....77
5.5.6 Driver blocks for measurement hardware....79
5.5.6.1 Structure of the energy program for measurement hardware....79
5.5.6.2 Energy data processing based on the "IPO" principle....79
5.5.6.3 Area: Input....81
5.5.6.4 Area: Processing....81
5.5.6.5 Area: Output....83
5.6 Compiling and downloading the PLC program....85
5.7 Testing the energy program....86
5.8 Editing the PLC program without installation of Energy Suite....87
5.9 PLC program without measurement hardware library with driver blocks....89
6 Visualizing and evaluating energy data....91
6.1 Procedural overview of archiving and visualization with WinCC Professional....91
6.2 Energy values on PC station with WinCC Professional....93
6.3 Displaying energy values for archiving in WinCC Professional....94
6.4 Overview of energy data....96
6.5 Create archiving structure for energy values....97
6.6 Data log and segmentation for Energy Suite....100
6.7 Archiving energy data....101
6.8 Visualize energy data with WinCC Online Trend View....102
6.9 Evaluate energy data with Energy Suite Export Tool....103
A Energy Suite blocks....109
A.1 Overview of Energy Suite blocks....109
A.2 Function blocks (FBs)....113
A.2.1 EnS_ArchiveDataLog: Write energy values to data logs....113
A.2.1.1 Description of EnS_ArchiveDataLog....113
A.2.1.2 Parameters of EnS_ArchiveDataLog....114
A.2.1.3 Parameter status....116

A.2.1.4 Alarms of EnS_ArchiveDataLog....118
A.2.2 EnS_ArchiveHMI: Write energy values to data logs....121
A.2.2.1 Description of EnS_ArchiveHMI....121
A.2.2.2 WinCC Professional component....123
A.2.2.3 Parameters of EnS_ArchiveHMI....124
A.2.2.4 Parameter status....125
A.2.2.5 Alarms of EnS_ArchiveHMI....126
A.2.3 EnS_BufferRead: Reading energy values from data blocks....128
A.2.3.1 Description of EnS_BufferRead....128
A.2.3.2 Parameters of EnS_BufferRead....129
A.2.3.3 Parameter status....129
A.2.4 EnS_BufferWrite: Write energy values to data blocks....130
A.2.4.1 Description of EnS_BufferWrite....130
A.2.4.2 Parameters of EnS_BufferWrite....131
A.2.4.3 Parameter status....132
A.2.5 EnS_CalcEnergyData: Calculating energy values....133
A.2.5.1 Description of EnS_CalcEnergyData....133
A.2.5.2 Parameters of EnS_CalcEnergyData....136
A.2.5.3 Parameter status....137
A.2.6 EnS_DrvAnalogValue: Acquire power values of the REAL type....139
A.2.6.1 Description of EnS_DrvAnalogValue....139
A.2.6.2 Parameters of EnS_DrvAnalogValue....139
A.2.6.3 Parameter status....140
A.2.6.4 Alarms of EnS_DrvAnalogValue....140
A.2.7 EnS_DrvCounterLInt: Acquire energy counter values of the LINT type....141
A.2.7.1 Description of EnS_DrvCounterLInt....141
A.2.7.2 Parameters of EnS_DrvCounterLInt....141
A.2.7.3 Parameter status....142
A.2.7.4 Alarms of EnS_DrvCounterLInt....142
A.2.8 EnS_DrvCounterLReal: Acquire energy counter values of the LREAL type....143
A.2.8.1 Description of EnS_DrvCounterLReal....143
A.2.8.2 Parameters of EnS_DrvCounterLReal....143
A.2.8.3 Parameter status....144
A.2.8.4 Alarms of EnS_DrvCounterLReal....144
A.2.9 EnS_DrvPulse: Acquiring pulse signals....145
A.2.9.1 Description of EnS_DrvPulse....145
A.2.9.2 Parameters of EnS_DrvPulse....145
A.2.9.3 Parameter status....146
A.2.9.4 Alarms of EnS_DrvPulse....146
A.2.10 EnS_EnergyDataAdv: Providing advanced energy values....147
A.2.10.1 Description of EnS_EnergyDataAdv....147
A.2.10.2 Parameters of EnS_EnergyDataAdv....147
A.2.10.3 Parameter status....147
A.2.11 EnS_EnergyDataAdvMinMax: Providing advanced minimum and maximum values....148
A.2.11.1 Description of EnS_EnergyDataAdvMinMax....148
A.2.11.2 Parameters of EnS_EnergyDataAdvMinMax....148
A.2.11.3 Parameter status....149
A.2.12 EnS_EnergyDataBasic: Providing metadata and energy values....149
A.2.12.1 Description of EnS_EnergyDataBasic....149
A.2.12.2 Parameters of EnS_EnergyDataBasic....149
A.2.12.3 Parameter status....150
A.2.12.4 Alarms of EnS_EnergyDataBasic....151

A.2.13 EnS_Organization: Providing organizational data....153
A.2.13.1 Description of EnS_Organization....153
A.2.13.2 Parameters of EnS_Organization....153
A.2.14 EnS_TimeSync: Synchronize timers....154
A.2.14.1 Description of EnS_TimeSync....154
A.2.14.2 Parameters of EnS_TimeSync....156
A.2.14.3 Parameter status....157

A.3 Data blocks (DBs)....160
A.3.1 EnS_BufferDB....160
A.3.1.1 Description of EnS_BufferDB....160
A.3.1.2 Structure of EnS_BufferDB....160
A.3.2 EnS_BufferWorkDB....161
A.3.2.1 Description of EnS_BufferWorkDB....161
A.3.2.2 Structure of EnS_BufferWorkDB....161

A.4 User-defined data types (UDTs)....162
A.4.1 EnS_typeAnalogValue....162
A.4.1.1 Description of EnS_typeAnalogValue....162
A.4.1.2 Structure of EnS_typeAnalogValue....162
A.4.2 EnS_typeBufferReadCtrl....162
A.4.2.1 Description of EnS_typeBufferReadCtrl....162
A.4.2.2 Structure of EnS_typeBufferReadCtrl....162
A.4.3 EnS_typeBufferReadStat....163
A.4.3.1 Description of EnS_typeBufferReadStat....163
A.4.3.2 Structure of EnS_typeBufferReadStat....163
A.4.4 EnS_typeBufferWork....163
A.4.4.1 Description of EnS_typeBufferWork....163
A.4.4.2 Structure of EnS_typeBufferWork....163
A.4.5 EnS_typeBufferWriteCtrl....164
A.4.5.1 Description of EnS_typeBufferWriteCtrl....164
A.4.5.2 Structure of EnS_typeBufferWriteCtrl....165
A.4.6 EnS_typeBufferWriteStat....165
A.4.6.1 Description of EnS_typeBufferWriteStat....165
A.4.6.2 Structure of EnS_typeBufferWriteStat....165
A.4.7 EnS_typeEnergyAdv....165
A.4.7.1 Description of EnS_typeEnergyAdv....165
A.4.7.2 Structure of EnS_typeEnergyAdv....166
A.4.8 EnS_typeEnergyBasic....167
A.4.8.1 Description of EnS_typeEnergyBasic....167
A.4.8.2 Structure of EnS_typeEnergyBasic....167
A.4.9 EnS_typeEnergyCounter....167
A.4.9.1 Description of EnS_typeEnergyCounter....167
A.4.9.2 Structure of EnS_typeEnergyCounter....168
A.4.10 EnS_typeEnergyMeta....168
A.4.10.1 Description of EnS_typeEnergyMeta....168
A.4.10.2 Structure of EnS_typeEnergyMeta....168
A.4.11 EnS_typeHMIWriteCtrl....171
A.4.11.1 Description of EnS_typeHMIWriteCtrl....171
A.4.11.2 Structure of EnS_typeHMIWriteCtrl....171
A.4.12 EnS_typeHMIWriteStat....172
A.4.12.1 Description of EnS_typeHMIWriteStat....172
A.4.12.2 Structure of EnS_typeHMIWriteStat....172

A.4.13 EnS_typeOrgData....172
A.4.13.1 Description of EnS_typeOrgData....172
A.4.13.2 Structure of EnS_typeOrgData....173
A.4.14 EnS_typeSync....173
A.4.14.1 Description of EnS_typeSync....173
A.4.14.2 Structure of EnS_typeSync....173
A.4.15 EnS_typeValue....174
A.4.15.1 Description of EnS_typeValue....174
A.4.15.2 Structure of EnS_typeValue....174

A.5 Definition of enumerations....175
A.5.1 Enumeration for the connection type of the measuring device....175
A.5.2 Enumeration for units....175

Basics of the SIMATIC Energy Suite 1

1.1 Energy Management with SIMATIC Energy Suite

Energy management

Globally rising energy costs as well as legal requirements for achieving global climate goals increasingly motivate industrial enterprises to deal with energy data acquisition and energy management. The first step to sustainable energy management is to identify the main energy flows and energy consumers of a company or a production facility and to analyze their consumption characteristics. This is referred to as "energy transparency".

Measures to improve energy efficiency are specified based on the acquired energy data. The measures are subsequently implemented in the most effective sequence, technically and economically. The order of implementation is mainly determined by the factors "savings potential" and "implementation cost". This means measures with medium savings potential and low cost may be preferred to measures with high potential and high costs.

Another aspect is legal requirements. These require regular monitoring and reporting of emissions. The acquired energy data can be used to create company-wide energy transparency and to form complete energy and material balancing.

In addition to acquisition, visualization, archiving, performance indicator calculation and energy reporting of energy data are also important factors in energy management.

SIMATIC Energy Suite in STEP 7 and WinCC (TIA Portal)

The components of SIMATIC Energy Suite, which we simply refer to as the Energy Suite in the following, enable you to integrate central functions of the energy management even down to the level of your automation system. The configuration is made simply and conveniently directly in the TIA Portal (STEP 7 and WinCC).

Energy Suite records, archives and provides the energy data in the CPU in standardized form. You can then visualize the energy data and use the data in your automation program (e.g. for monitoring).

Template pictures and faceplates for visualizing the acquired energy data in WinCC Professional or WinCC Basic, Comfort bzw. Advanced are provided by Siemens Industry Online Support .

In addition, you can process the energy data in SIMATIC Energy Manager PRO:

- For performance indicator calculation, for example, energy costs per produced piece, CO_2 emissions per year
- Convenient generation of energy reports, for example, to optimize energy purchasing

Further processing energy data with SIMATIC Energy Manager PRO

You can directly transfer configuration information from Energy Suite into SIMATIC Energy Manager PRO and save the additional configuration work involved. SIMATIC Energy Manager PRO, which we simply refer to as Energy Manager PRO in the following, primarily offers functions for commercial and enterprise-wide assessment of energy consumption. To optimize the energy consumption, the relevant energy flows of all the company's energy media must be transparent.

You use the Energy Manager PRO to create energy and material balances or calculations of performance indicators, for example. The performance indicators (Key Performance Indicator (KPI)) enable you to evaluate and compare the energy used in various processes and plants. The evaluation and comparison is also referred to as "benchmarking". The assignment of energy costs to the actual consumers results in cost transparency and raising awareness of employees in regard to the energy consumption of machinery, processes and plants.

You can find additional information about the Energy Manager PRO at Siemens Industry Online Support (https://support.industry.siemens.com/cs/ww/en/ps/14711).

1.2 SIMATIC Energy Suite operations and components

IPO principle

Energy data processing according to the "IPO" principle can be divided into "Input" "Processing" and "Output":

- Input: Provision of the input data for processing, for example, by measurement hardware or calculations within the S7 CPU
- Processing: Acquisition, calculation and provision of energy values for other systems by energy objects in the S7 CPU
- Output: Visualization and archiving of energy values on connected HMI or SCADA systems

Overview

The following figure shows the relationship of the individual Energy Suitecomponents and the processing steps:

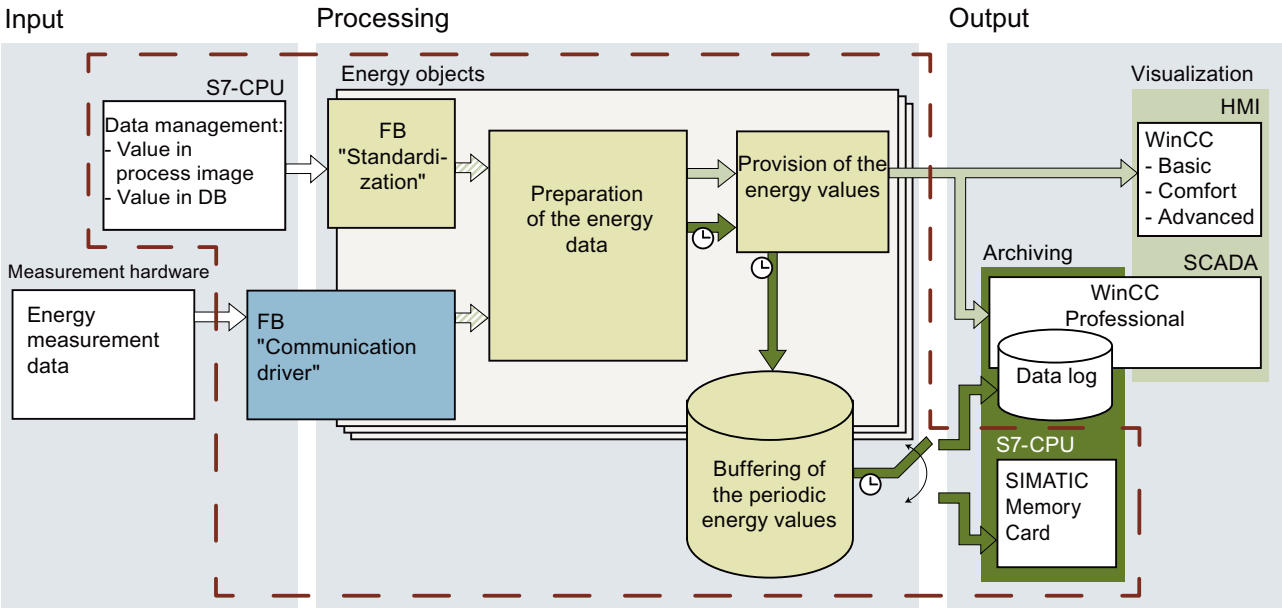

Input measurement data

Raw energy data

Processed instantaneous energy values

Prepared periodic energy values with time stamp

Energy Suite blocks (prefix EnS)

Driver blocks for measurement hardware (prefix EnSL)

FB Function block

Process on an S7 CPU

Components

The following table shows the components for energy data management with the Energy Suite:

Area	Component / processing step	Description
Input - Supply energy data (Page 21)	S7-CPU	S7-1500 CPU configured in STEP 7 project (TIA Portal) Provides the input measurement data to the "Normalization" function block (FB)
	Measurement hardware	Measurement capable hardware of Energy Suite Provides the input measurement data to the "Communication driver" function block (FB)

Area	Component / processing step	Description
Processing - Process energy data (Page 31)	FB "Normalization"	Includes the Energy Suite blocks with driver functionality for acquiring, pre-processing and harmonization of input measurement data on raw energy data Provides the raw energy data for the "Preparation of the energy data" processing step The following Energy Suite blocks are used: • EnS_DrvAnalogValue (Page 139) • EnS_DrvCounterLInt (Page 141) • EnS_DrvCounterLReal (Page 143) • EnS_DrvPulse (Page 145)
	FB "Communication driver"	Includes the function blocks with driver functionality (driver blocks) for the supported measurement hardware The driver blocks for the measurement hardware are provided in the form of an Energy Support Library (EnSL) (Page 24) on the SIMATIC Energy Suite product DVD and must be installed separately (not part of the Energy Suite library). Provides the raw energy data for the "Preparation of the energy data" processing step
	Energy objects	An energy object in the Energy Suite is typically a measuring point (for example, a measuring device or an analog value). The energy object represents the central engineering component where all settings are made (for example, the assignment of the archiving period and the energy source data as well as settings for processing of raw energy data).
	Preparation of energy data	Used for calculation, time synchronization and organization of raw energy data (for example, compression and periodic time stamping of the energy values) Passes the prepared instantaneous energy values and prepared periodic energy values with a time stamp to the "Provision of the energy values" processing step The following Energy Suite blocks are used: • EnS_CalcEnergyData (Page 133) • EnS_TimeSync (Page 154) • EnS_Organization (Page 153)
	Provision of the energy values	Provision of: • Processed instantaneous energy values on "Visualization - HMI" and "Visualization - SCADA" • Prepared periodic energy values with time stamp to "Buffering periodic energy values" The following Energy Suite blocks are used: • EnS_EnergyDataBasic (Page 149) • Ens_EnergyDataAdv (Page 147) • Ens_EnergyDataAdvMinMax (Page 148)
	Buffering of the periodic energy values	Buffering of the processed periodic energy values with a time stamp from several energy objects Provides the energy values to: • Archiving - WinCC Professional (Data Log) or • Archiving - S7-CPUs (SIMATIC Memory Card) The following data blocks (DBs) are used by the Energy Suite: • EnS_BufferDB (Page 160)

Area	Component / processing step	Description
		• EnS_BufferWorkDB (Page 161) • EnS_BufferRead (Page 128) • EnS_BufferWrite (Page 130)
Output - Visualizing and evaluating energy data (Page 91)	Visualization - HMI	Visualization of instantaneous energy values on a SIMATIC Comfort Panel or PC with WinCC applications (Basic, Comfort or Advanced)
	Visualization - SCADA	Visualization of instantaneous energy values or the buffered periodic energy values on the WinCC Professional SCADA system
	Archiving - WinCC Professional (Data Log)	Archiving of buffered periodic energy values in the data log of the WinCC Professional SCADA system The following Energy Suite block is used: • EnS_ArchiveHMI (Page 121)
	Archiving - S7-CPUs (SIMATIC Memory Card)	Archiving of buffered periodic energy values as data logs (CSV files) on the SIMATIC memory card inserted in the S7-1500 CPU The following Energy Suite block is used: • EnS_ArchiveDataLog (Page 113)

1.3 Procedural overview

Requirement

To perform the following configuration steps, the following requirements must be met in STEP 7 (TIA Portal):

- The "SIMATIC Energy Suite" add-on package has been installed.
  You can find additional information on this in the installation instructions for Energy Suite on the DVD.
- The STEP 7 project has been created and configured with a CPU S7-1500.

Additionally, when using measurement hardware as an energy data source:

- The Energy Support Library for using future measurement hardware has been installed.
  You can find additional information on this in the product documentation for the measurement hardware.
- The measurement hardware has been created and configured in the STEP 7 project.
  You can find additional information on this in the product documentation for the measurement hardware.

Additionally, for visualization or archiving to data logs:

- The PC station has been created and configured in the STEP 7 project with WinCC Professional or WinCC Basic, Comfort or Advanced.

Configuration steps

The following table provides an overview of the most important configuration steps for Energy Data Management with the Energy Suite components:

Topic	Step		Description
Supply energy data	1	Supply energy data (Page 21)	Create and configure connected measurement hardware and/or PLC tags (for example, values of data blocks and/or from the process image) so that they can be later configured as an energy source for the energy objects.
Process energy data	2.1	Creating an energy object table (Page 33)	Energy object tables are used to create and manage the energy objects. Several energy object tables enable the mapping and structuring of the energy objects, for example, based on the measurement hierarchy of your production plant.
	2.2	Adding energy objects for energy object tables (Page 34)	In order to acquire and process the energy data from a data source of energy, you need to create energy objects in the energy object tables.
	2.3	Setting the properties of the energy objects (Page 36)	Each energy object represents a measuring point of your production plant and must be correspondingly configured. • Assign energy data source • Set processing of the input measurement data • Activate the energy values for archiving and assign the archive period • Check the enable of energy objects for the energy program
Buffering and archiving energy data	3.1	Defining archiving periods (Page 49)	From a central location, you manage the periodic time intervals (archiving periods) in which the raw energy data should be calculated to a prepared periodic energy value with time stamp and archived. Each energy object can be assigned a desired archiving period.
	3.2	Defining the buffering of periodic energy values (Page 59)	For buffering the periodic energy values, you specify the size of the data blocks (buffer DBs). The number of buffer DBs corresponds to the number of created energy object tables (one buffer DB for each energy object table).
	3.3	Setting the archiving on a S7-CPU (SIMATIC Memory Card) (Page 53)	Define the name, the number and size of data logs for archiving the periodic energy values on a SIMATIC memory card inserted in the S7 CPU. The energy values are entered in the data logs.
	3.4	Setting the archiving on a PC station with WinCC Professional (Data Log) (Page 57)	For archiving the periodic energy values in the data log of WinCC Professional, select the appropriate check box in the energy program settings of Energy Suite. The data logs of WinCC Professional must be set up for this in the settings of WinCC Professional.

Topic	Step		Description
Generate energy program	4.1	Making general settings for the energy program (Page 65)	Before generating the energy program, set the number of acyclic parallel communication channels and the starting time of the blocks in the energy program. You also define the alarm suppression.
	4.2	Check the enable of energy objects for the energy program and adapt it if necessary (Page 38)	At the latest before generating the energy program, you need to decide which energy objects (and thus which measuring points) are to be included in the energy program. You can easily enable/disable the energy objects in the energy object tables.
	4.3	Generate energy program (Page 68)	You start the generation of the PLC program for energy data acquisition and processing (or energy program for short) from the project tree. During the generating process, the required blocks and data types from the Energy Suite Library and the measurement hardware library are automatically used in the energy program and interconnected according to the configured energy objects. The generated energy program is created below the S7 CPU in the "Energy Suite - Program" subfolder of the "Program blocks" folder.
	4.4	Compiling and downloading the PLC program (Page 85)	Once the energy program is generated, compile the complete PLC program and download it to the S7-1500 CPU target system.
	4.5	Testing the energy program (Page 86)	To check your settings, open the relevant data block of the respective energy object in the energy program and watch the online values with the "Monitor all" function.

Topic	Step		Description
Visualizing and evaluating energy data (Page 91)	5.1	Configuring a PC station with WinCC Professional	A PC station with the SIMATIC "WinCC RT Professional" HMI application is required for the SCADA system.
	5.2	Creating a connection to the SCADA system	To enable data transfer between the S7-1500 CPU and the SCADA system, configure an HMI connection to the PC station with WinCC Professional in STEP 7 (TIA Portal).
	5.3	Reading and displaying energy values configured for archiving on the engineering system on the PC station (Page 94)	On the PC station with WinCC Professional, Energy Suite offers an overview of all energy objects and the periodic energy values that you have configured in the engineering system for archiving in the data log of WinCC Professional. With the "Refresh" function, synchronization is performed with the engineering data of the S7-1500 CPUs that can be reached by the PC station with WinCC Professional.
	5.4	Configuring the SCADA system (Page 97)	Configuring the PC station with WinCC Professional: • For data transfer to the SCADA system, generate the HMI tags for the data blocks for buffering ("HMI tags" folder). • For archiving, generate the data log with the logging tags (energy values) and set up the archive segments for archiving (Page 100) of the periodic energy values ("Archive" folder). • For visualization of the collected energy data in runtime, you can integrate the template screens and screen elements from the Siemens Industry Online Support into the WinCC application ("Screens" folder).
	5.5	Compiling and downloading the PC station with WinCC Professional	Next you compile the configuration for the entire PC station and download it to the SCADA system.
	5.6	Starting the PLC program and SCADA system (Runtime)	Once WinCC RT Professional is started on the SCADA system and the PLC programs are started on the S7-1500 CPUs, archiving is ready to begin. The selected periodic energy values are archived automatically. The archiving is performed continuously while Runtime is active.

1.4 Performance features and conventions

Compatibility of Energy Suite with CPU and FW

The following table shows the compatibility of CPUs to respective Energy Suite versions.

CPU / ET 200SP CPU / ET 200SP Open Controller	FW	Energy Suite version
S7-1500 CPU	≥ V2.0	V14.0 V14.0 SP1 V15.0 V15.1

Performance features of Energy Suite

The following table shows the performance features of the Energy Suite:

Characteristic	Value
Maximum number of energy objects that can be licensed per S7-1500 CPU	500
Maximum number of energy object tables per S7-1500 CPU	10
Maximum number of S7-1500 CPUs for simultaneous archiving	10
Maximum number of energy objects that can be archived simultaneously (WinCC Professional) Recommendation: distributed to a maximum of 10 S7-1500 CPUs	2000
Minimal adjustable archiving period	1 min
Value range of an archiving period	Integer part of one hour/one day, for example: • 1 minute • 5 minutes • 15 minutes • 12 hours The following are invalid, for example: • 7 minutes • 11 minutes • 5 hours
Maximum adjustable archiving period	24 h
Typical archiving period for electrical variables	15 min
Typical archiving period for non-electrical variables	60 min

Naming conventions

The following table shows the naming conventions for engineering with the Energy Suite:

Object	Value range for name	Range of validity
PLC	• Prohibited characters: – : ? " ' \ * % – <space> – <line feed> – <line break>	• The naming convention of the PLC applies to the archiving of energy data of energy objects on PC station with WinCC Professional.
Energy object	• 1 - 32 characters • Prohibited characters: – : ? ' * " \ $ % § ° – <Space> – <Line feed> – <Line break>	• Unique project-wide for archiving type"WinCC Professional (data log)"
Energy object table	• 1 - 32 characters • Prohibited characters: – <Line feed> – <Line break>	• Unique project-wide for archiving type"WinCC Professional (data log)"
Archiving period	• 1 - 32 characters • Prohibited characters: – : ? ' * " \ % § ° – <Space> – <Line feed> – <Line break>	-
Data log	• 1 - 29 characters • Permitted characters: – 0 ... 9 – a ... z – A ... Z – - – _	-

Note

Unique names

The object names must be unique at least CPU-wide. You cannot use a name that has already been assigned to a block, a PLC tag or a constant within the CPU for a new object. No distinction is made between upper and lower case when checking the uniqueness of the name.

Supply energy data

2

Energy data sources provide the energy data for processing (acquisition, processing, provision) to the energy object (Page 31). The following data sources can be configured as energy sources of an energy object:

- Input measurement data from connected measurement hardware (Page 24)
- Energy data from the process image or the bit memory area of the CPU (PLC tags) (Page 27)
- Energy data from data blocks (DB tags) (Page 29)

Data exchange of periodic energy data energy counter value, (pulse signal, energy counter value, power value), from which the basic energy data are calculated, occurs cyclically via the process image. The measurement hardware provides energy data to energy objects using cyclic and acyclic communication. Advanced energy data, for example, voltage, current or frequency value, are passed in a data record from the measurement hardware to the energy object. The maximum number of parallel acyclic communication channels is limited to 10.

You assign energy data sources for the energy program when you configure the energy objects. Each energy object is assigned an energy data source.

2.1 Important information on energy data

The following different energy data exists:

- Basic energy data
- Advanced energy data

Basic energy data

Basic energy data includes the following energy data:

- Periodically calculated energy consumption
- Average output per archive period
- Current values for energy and power consumption

Basic energy data is invoice-relevant energy data which is required for visualizing the energy flows in a plant.

Basic energy data is primarily transmitted by cyclical communication via the process image.

Advanced energy data

Advanced energy data are only read from the measurement hardware which can make this data available. Advanced energy data includes the following energy data:

- Phase-specific current, voltage, power and power factor values
- Total power, total energy and total power factor values
- Frequency values

Whether and to what extent the advanced energy data is made available depends on the respective measurement hardware. In addition to supplying the input measured value, the measurement hardware or the driver block also supply the **minimum and maximum values** of the advanced energy data, for example for the output in a faceplate in WinCC Professional.

Advanced energy data is primarily transmitted by acyclic communication in a data record.

Cyclic communication

You use cyclic communication for invoice-relevant energy data (for example, power, energy counters) which is processed for monitoring and calculation functions in the CPU. This data is read cyclically, quickly and reliably via the process image.

For additional information on the process image, refer to the online help for the TIA Portal information system under the keyword "Process image".

Acyclic communication

You use acyclic communication for energy data relevant for visualization (e.g. current, voltage), which is displayed on a SIMATIC Comfort Panel or PC. For the visualization of the screen opening, for example, this data is communicated for the duration of the screen opening.

The energy data is read in a data record with the "RDREC" instruction from the corresponding module. For additional information on the "RDREC" instruction, refer to the online help for the TIA Portal information system under the keyword "RDREC".

The driver blocks for the measurement hardware call the "RDREC" instruction for acyclic communication internally.

Note

Use both types of communication depending on the application. Make sure that the communication channels are not overloaded.

2.2 Energy data from measurement hardware

A variety of devices can provide energy data. In addition to the measuring instruments (e.g. ET 200SP AI EnergyMeter 480VAC ST, SENTRON PAC), the sole purpose of which is to measure energy, many automation devices (for example, frequency converters or protection devices and switchgear) provide energy data in addition to their actual automation functions (for example, speed control). In the following, these devices are referred to as measurement hardware.

Overview of measurement hardware for Energy Suite

The following table shows some examples of devices as measurement hardware that are supported by SIMATIC Energy Suite. For this purpose, specific function blocks (FBs) with driver functionality are available for the measurement hardware from the Energy Support Library (EnSL).

Measurement hardware	Description
AI EnergyMeter 480VAC ST	Analog input module for the SIMATIC ET 200SP distributed I/O system for measuring relevant network parameters in the voltage range up to 480V AC phase-to-phase voltage
AI EnergyMeter 480VAC HF	Analog input module for the SIMATIC ET 200SP distributed I/O system for measuring relevant network parameters in the voltage range up to 480 V AC phase-to-phase voltage Extended applications and functionality; current measurement using Rogowski coils or current and voltage transformers The AI EnergyMeter 480 V AC HF is available in two versions: CT and RC.
SENTRON PAC3200/PAC4200	Multifunctional measuring instrument for measuring and displaying relevant power system parameters in low-voltage power distribution
SENTRON 3VA (8x ETU)	Molded case circuit breakers for low-voltage power distribution equipment, for example, for protecting production equipment against electrical damage
SINAMICS AC/AC	Converters for efficient and economical operation of pumps, fans and compressors, for example
SIMOCODE pro V PN	Motor protection and control device for motor management tasks such as comprehensive protection, monitoring, safety and control functions between the motor feeder and automation system
ET 200SP/M200D Motor Starter	Motor starter for switching and protecting three-phase induction motors and single-phase AC motors
Soft starter 3RW44	Soft starter for limiting the starting current and the starting torque. Both mechanical loads and supply voltage failures can be reliably avoided with this starter.
Soft starter 3RW55	New generation of soft starters for limiting the starting current and the starting torque. Both mechanical loads and supply voltage failures can be reliably avoided with this starter.

Device configuration of measurement hardware in STEP 7 (TIA Portal)

The device configuration of the measurement hardware is performed in STEP 7 (TIA Portal). The following options are available to add the measurement hardware to the hardware configuration of your project:

- Transfer the device directly from the hardware catalog into the network or topology view
- Add a device not contained in the hardware catalog via a **GSD file** (device master data file) or **HSP** (Hardware Support Packages)

If you want to configure measurement hardware that is not contained in the hardware catalog, you need to first install the vendor-supplied GSD file or HSP. Using GSD files or HSP, installed measurement hardware is displayed in the hardware catalog and can then be selected and configured in the project.

You can find additional information about configuring a device using the GSD file or HSP in the online help of the TIA Portal information system.

Driver blocks of the measurement hardware for STEP 7 (TIA Portal)

The energy data (input measurement data) of the measurement hardware are made available in different ways (e.g. differences in the data format or in the data record address).

In addition to the Energy Suite blocks, function blocks (FBs) specific to the measurement hardware with driver functionality are made available through the **Energy Support Library (EnSL)**. These hardware-specific function blocks (referred to as driver blocks in the following), form the interface between the special measurement hardware and your automation system.

The energy data from the measurement hardware is read and uniformly provided to the automation system using driver blocks. In this way, the raw energy data is provided easier and faster to your PLC program and Energy Suite. The Energie Suite blocks are then used to prepare and process raw energy data.

Energy Support Library (EnSL)

Perform the installation of the Energy Support Library according to the installation instructions for the measurement hardware.

After the successful installation of the Energy Support Library:

- The measurement hardware library with the driver blocks is available in STEP 7 (TIA Portal) as a global library. The driver blocks can be used in the PLC program.
- The measurement hardware can be assigned an energy object for selecting the energy data source.

You can find additional information about using global libraries in the online help of the TIA Portal information system.

Type of communication

The CPU can read the energy data from the measurement hardware in two ways, with cyclic or acyclic communication. The cyclic energy data is transferred to Energy Suite via the process image and the acyclic energy data is transferred in a data record of the measurement hardware.

The type of the communication between the CPU and the measurement hardware depends on the measurement hardware and the driver blocks used. Normally, the basic energy data is transferred via cyclic communication, advanced energy data is transferred via acyclic communication:

- You use cyclic communication for invoice-relevant energy data (for example, power, energy counters) which is processed for monitoring and calculation functions in the CPU.
- You use acyclic communication for visualization-relevant energy data (e.g. current, voltage) which is displayed on a SIMATIC Comfort Panel or PC.

You can find additional information on communication between the CPU and the measurement hardware used in the documentation of the respective measurement hardware.

Note

If you use multiple pieces of measurement hardware, pay attention to the load on the communication channels that are supported by the CPU used.

See also

Assign energy data source (Page 39)

2.3 Energy data from the process image or bit memory area of the S7-CPU

PLC tags

PLC tags can be used as an energy source for energy objects. To do this, create new PLC tags or use existing PLC tags in the PLC program. Depending on the desired energy data type (pulse signal, power value or energy counter value) you assign the created PLC tag to an appropriate data type. You assign the data type of the PLC tag with the declaration in the PLC tag table. You assign the energy data types to energy data sources with the configuration of the energy object.

Examples

- A measuring instrument that is not supported by the Energy Support Library provides an power value and a energy counter value. You assign these values to the PLC tags.
- 10 fan motors each have a rated output of 1.5 kW. The motors are controlled by a PLC. In the PLC, the theoretical consumption is derived from the switching states and rated power. For example, if five motors are switched on, the resulting power consumption is 7.5 kW. This calculation is performed in the PLC and can be assigned to the PLC tags.

Overview of the relevant data types for PLC tags

The following table shows the relevant data types for PLC tags for energy data acquisition:

Data type	Description
BOOL	Used to acquire a pulse signal From an acquired binary pulse signal, the resulting energy consumption is calculated using the normalization factor. Example: With a normalization factor of 2 and the physical unit "Wh", 3 pulse signals result in an energy consumption of 6 Wh.
REAL, LREAL	Used to acquire a power value The acquired power value is standardized and the resulting energy consumption calculated. Example: Average power consumption of 180 kW over 10 seconds corresponds to energy consumption of 0.5 kWh or 500 Wh.
SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT	Used to acquire an energy counter value The power consumption and the instantaneous value of the power consumption is calculated from the change of the absolute energy counter value using the normalization factor. Example: The counter states of an energy counter change in 10 seconds from 6,734,392 Wh to 6,734,414 Wh. This corresponds to an energy consumption of 22 Wh and an instantaneous value of the power consumption of 7.92 kW.
REAL, LREAL	

Declaring PLC tags for energy data acquisition

To declare the PLC tags for energy data acquisition, create the PLC tags in a PLC tag table according to the standard procedure. Depending on the energy data type, you assign the PLC tags the appropriate data type.

You can find additional information about the procedure in the online help of the TIA Portal information system under "Declaring tags in the PLC tag table".

See also

Assign energy data source (Page 39)

2.4 Energy data from data blocks

DB tags

DB tags can be used as an energy source for energy objects. To do this, create new DB tags in data blocks or use existing DB tags in the PLC program.

Depending on the desired energy data type (pulse signal, power value or energy counter value) you assign the created DB tag to an appropriate data type. You assign the data type of the DB tag or an element in the declaration table for the data blocks. You assign the energy data types to energy data sources with the configuration of the energy object.

Overview of data types for DB tags

The following chart shows the different data types for DB tags for energy data acquisition:

Data type	Description
BOOL	Used to acquire a pulse signal From an acquired binary pulse signal, the resulting analog power value is calculated using the normalization factor. Example: With a normalization factor of 2 and the physical unit "Wh", 3 pulse signals result in an energy consumption of 6 Wh.
REAL, LREAL	Used to acquire a power value The acquired power value is standardized and the resulting energy value calculated. Example: Power consumption of 180 kW over 10 seconds corresponds to energy consumption of 0.5 kWh or 500 Wh.
SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT	Used to acquire an energy counter value The power consumption and the instantaneous mean power consumption is calculated from the change of the absolute energy counter value using the normalization factor. Example: The counter states of an energy counter change in 10 seconds from 6,734,392 Wh to 6,734,414 Wh. This corresponds to an energy consumption of 22 Wh and an average power consumption of 7.92 kW.
REAL, LREAL	

Declaring DB tags for energy data acquisition

To declare the DB tags for energy data acquisition, create corresponding data blocks and define their structure according to the standard procedure.

You can find additional information about the procedure in the online help of the TIA Portal information system under "Editing properties of tags in data blocks".

See also

Assign energy data source (Page 39)

Process energy data

3

3.1 Energy objects and energy program

For processing raw energy, define energy objects in STEP 7 (TIA Portal) in the project tree below the S7-1500 CPU. The energy objects are used for process-specific configuration of raw energy data and the subsequent generation of an energy program.

There is the new "Energy objects" folder in the project tree for this after the successful installation of SIMATIC Energy Suite. The following figure shows the "Energy objects" folder in the project tree.

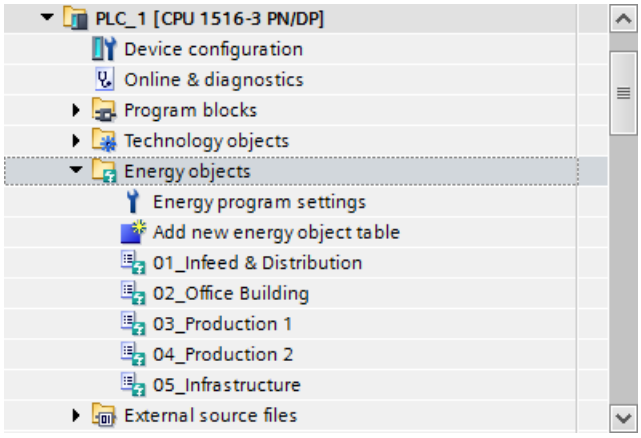

Energy objects

Each energy object represents a measuring point of your production plant. You use the parameter setting of energy object to define the processing of the input measurement data to energy values that can be subjected to analysis.

To define the periodic calculation of energy values, assign each energy object an archiving period. This enables you to define the intervals at which the acquired raw energy data for a periodic energy value are to be compressed and archived with their time stamp. Often, an electric power value is archived in 15 min. periods and a non-electric energy value (e.g., gas) is archived in 60 min. periods. You can centrally create manage multiple archiving periods independent of the energy objects.

You create and manage energy objects in energy object tables. Several energy object tables enable the structuring of the energy objects, for example, based on the hierarchy of the measuring points in your production plant.

You can find additional information about the possible number of energy objects and energy object tables in the section "Performance features and conventions (Page 19)".

Energy program

Based on the configured energy objects and the archiving periods, a PLC program is generated for the acquisition, processing and archiving of energy data. This program is referred to as the "energy program" in the following.

When the energy program is generated, blocks from the supplied Energy Suite library and from the measurement hardware library (Energy Support Library) are automatically created and interconnected according to the configuration of the energy objects. Before the start of the generation of the energy program, you have the option to select those measuring points that should be taken into account in the energy program by enabling the energy objects.

Once the energy program is generated, it is available in the project tree below the CPU in the "Program blocks" > "Energy Suite - program" folder together with all the required components.

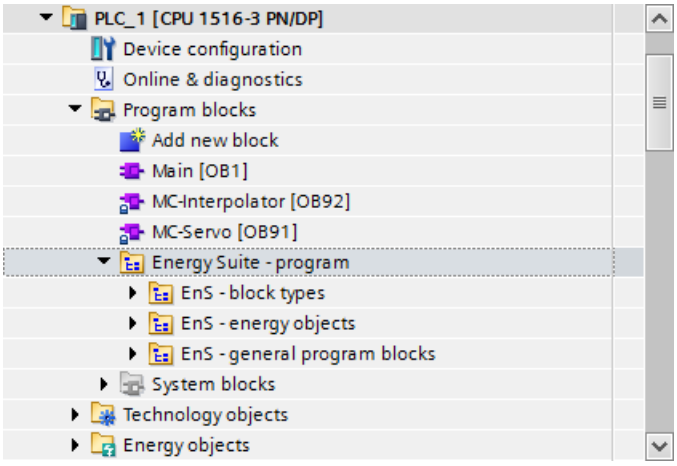

Programming rules

Note

The "EnS_" prefix is reserved for the Energy Suite.

Before generating the energy program, make sure that the prefix "EnS_" is not used in your existing PLC program (user blocks, tags, UDTs or technology objects). In this way, you avoid unintentional overwriting when generating the energy program.

Note

The "Energy Suite - Program" folder is a reserved area for the Energy Suite.

Before generating the energy program, make sure that no parts of your existing PLC program are located in the "Energy Suite program" folder. In this way, you avoid unintentional overwriting when generating the energy program.

3.2 Creating an energy object table

First, create energy object tables below the CPU in the "Energy objects" folder. Consider in advance how you want to structure the energy objects over multiple energy object tables. The number of energy object tables affects the maximum buffer duration (Page 59).

You can find additional information about the possible number of energy object tables and naming conventions in the section "Performance features and conventions (Page 19)".

Requirement

- An S7-1500 CPU is created in the project.

Procedure

To create an energy object table, proceed as follows:

1. Open the S7-1500 CPU in the project tree.
2. Open the "Energy objects" folder below the S7-1500 CPU.
3. Double-click on the "Add new energy object table" object.
   A new energy object table is created directly in the "Energy objects" folder.
4. In the shortcut menu of the new energy object table, select the "Rename" command and enter the desired name in adherence with the naming conventions (Page 19).

If needed, create additional energy object tables in the same way. Continue creating energy object tables below any other S7-1500 CPUs.

3.3 Adding energy objects for energy object tables

In order to process the energy data, create energy objects in the energy object tables for the measuring points of your production plant. Consider how you want to structure the energy objects over multiple energy object tables.

You can find additional information about the possible number of energy objects and naming conventions in the section "Performance features and conventions (Page 19)".

Requirement

- An energy object table has been created in the "Energy objects" folder below the S7-1500 CPU.

Procedure

To create an energy object, follow these steps:

1. Open the "Energy objects" folder below the S7-1500 CPU in the project tree.
2. Double-click on the energy object table in which you want to enter the energy object.
   The energy object table is open in the working area.
3. Double-click "<Add>" in the "Name" column of the energy object table.
   A new energy object is created in the energy object table.

Create additional energy objects for all energy object tables in the same way. Continue creating energy objects below any other S7-1500 CPUs.

Result

Each row of an energy object table represents an energy object. You can clearly view the most important properties features of the energy objects directly in the energy object table and change them necessary. This includes:

- Name of the energy object
- Energy data source of the energy object
- Enable status of the energy object for the energy program
- Configuration of specific properties of the energy object (processing properties)
- Archiving period of the energy object

The following figure shows an energy object table in the working area of STEP 7 (TIA Portal):

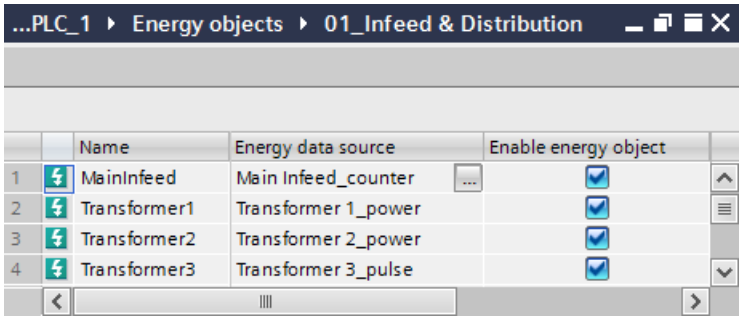

When an energy object is selected in the energy object table in the working area of the Inspector window of STEP 7 (TIA Portal), the specific properties of the energy object are displayed. Setting for these properties is described in the following sections.

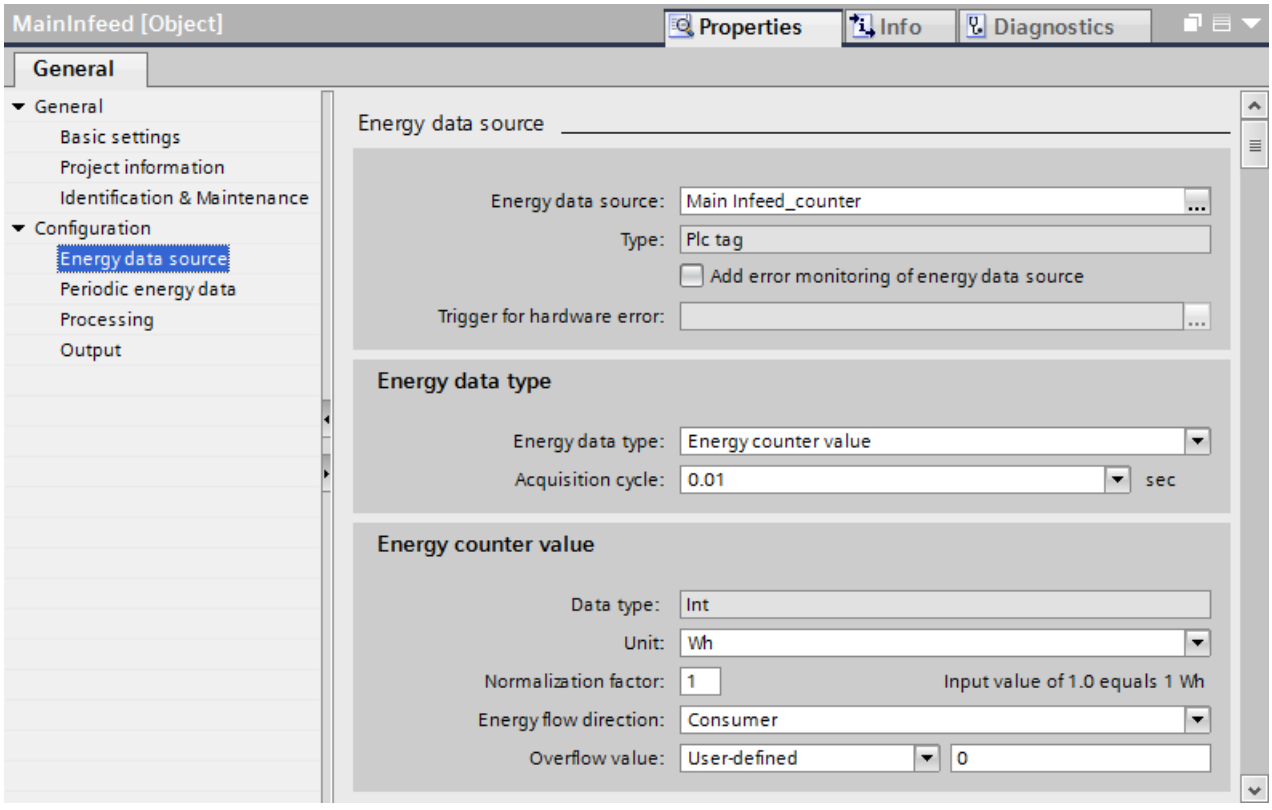

Copying energy objects

Energy objects can be copied between the energy object tables via the shortcut menu. After copying, check the properties of the energy object and adapt the copied settings as needed.

See also

Configuring an energy object (Page 36)

Change the enable for energy objects for the energy program (Page 38)

3.4 Setting the properties of the energy objects

3.4.1 Configuring an energy object

Each energy object represents a measuring point of your production plant and is correspondingly configured by you. You configure an energy object in the "Properties" Inspector window.

Note

Alternatively, you can display or hide the most important parameters of the energy object in the energy object table or (via the shortcut menu of the table header). The settings of the energy objects can be clearly displayed and also changed in the energy object table individually assembled in this way.

For each energy object, you define:

- Data source of raw energy data
- Parameters for the acquisition and processing of energy data
- Parameters for archiving energy data (e.g. archiving period)
- Enable of the energy object for the energy program

The energy program is generated based on the configured properties.

Requirement

- An energy object has been created in the energy object table.

Procedure

To open the "Properties" Inspector window of an energy object, follow these steps:

1. Open the "Energy objects" folder in the project tree.
2. Double-click on the energy object table in which the energy object has been created.
   The energy object table with the energy objects opens.
3. Click on the desired energy object in the energy object table.
   All parameters of the energy object are displayed in the "Properties" Inspector window.

The configuration of the energy object is described in detail in the following sections.

See also

Basic settings (Page 37)

Energy data source (Page 39)

Periodic energy values (Page 44)

Processing (Page 47)

3.4.2 Basic settings

The following table shows the basic settings of an energy object:

Setting	Description
Name	CPU-wide unique name of the energy object. The name appears in the energy object table and can be changed according to individual requirements. Valid name length: 1 - 32 characters The following characters are prohibited: ? ' * " \ % § ° You can find additional information about naming conventions in the section "Performance features and conventions (Page 19)".
Comment	Optional short description of energy objects, for example, about the purpose or the data source of energy object
Enable energy object	• When this check box is selected, the energy object is enabled for generating the energy program. • When the check box is not selected, the energy object is not used in the energy program. If the number of Runtime licenses for the Energy Suite (Page 67) is fewer than the number of enabled energy objects, the generation of the energy program is canceled and an error alarm is displayed. Energy objects that are not enabled are ignored by the license count.
Date created	The creation date corresponds to the date on which the energy object was added to the energy object table. The creation date cannot be changed.
Last modified	The date of modification corresponds to the date on which the energy object was last changed. The date of modification is updated each time a change is made to the properties of the energy object. The date of modification cannot be changed.

See also

Configuring an energy object (Page 36)

Change the enable for energy objects for the energy program (Page 38)

3.4.3 Project information

The following table shows the settings of the energy object under "Project information":

Setting	Description
Energy object detection	The energy object identifier is assigned automatically when you create the energy object. The energy object identifier is used internally for addressing and identification of the energy object. The energy object identifier cannot be changed.
Author	Name of the editor of the energy object. The name can be changed according to individual requirements. The user logged onto the operating system is set by default.

See also

Configuring an energy object (Page 36)

3.4.4 Identification & Maintenance

The following table shows the settings of the energy object under "Identification & Maintenance":

Setting	Description
Plant designation	The plant designation defines the measuring point to which the energy object is assigned within the process plant.
Location identifier	The location ID defines the exact location of the measuring point to which the energy object is assigned within the process plant.
Installation date	Date on which the measuring point assigned to the energy object was installed in the plant. The date can be selected from the calendar dialog box.
Additional information	Additional information about the measuring point assigned to the energy object, for example, the intended use of the measurement hardware.

See also

Configuring an energy object (Page 36)

3.4.5 Change the enable for energy objects for the energy program

You can change the inclusion of the individual energy objects for generating the energy program. After adding a new energy object, the enable is active (default setting). It may be useful to disable this setting for some applications (e.g. for testing purposes or to make maximum use of runtime licenses).

Note

Energy objects that are not enabled are ignored by the Runtime licensing (Page 67).

You have the following options for this:

- In the Inspector window: when creating the energy object and assigning parameters to it
- In the energy object table: central and clear for all associated energy objects

Procedure

To change the enable of the energy object for the energy program in the Inspector window, follow these steps:

1. Open the "Energy objects" folder in the project tree.
2. Double-click on the energy object table in which the energy object has been created. The energy object table with the created energy objects opens.

3. Click on the desired energy object in the energy object table.
   All parameters of the energy object are displayed in the "Properties" Inspector window.
4. Select/clear the check box "Enable energy object" for the basic settings.
   – Activated: The energy object is enabled for generating the energy program.
   – Deactivated: The energy object is not used for generating the energy program.

This setting for the energy object is also updated within the "Enable energy object" column of the energy object table. Alternatively, you can select/clear the check box in the energy object table.

3.4.6 Energy data source

3.4.6.1 Assign energy data source

In order to create an energy program, each energy object requires an energy data source that provides energy data. The following sources are available as energy data sources for the energy objects:

- Input measurement data from connected measurement hardware (Page 24)
- Energy data from the process image or the bit memory area of the CPU (PLC tags) (Page 27)
- Energy data from data blocks (DB tags) (Page 29)

You have the following options for assigning the energy data source:

- In the Inspector window: when creating the energy object and assigning parameters to it
- In the energy object table: central and clear for all associated energy objects

Requirement

- The PLC tags and DB tags are declared for energy data acquisition.
- The measurement hardware for power data acquisition is configured below the S7-1500 CPU.
- The Energy Support Library with the driver blocks for the measurement hardware is installed.

In the Inspector window

To assign an energy data source to a energy object, follow these steps:

1. Open the "Energy objects" folder in the project tree.
2. Double-click on the energy object table in which the energy object was entered.
   The energy object table with the created energy objects opens.
3. Click on the desired energy object in the energy object table.
   The properties of the energy object are displayed in the "Properties" Inspector window.

4. Select the "Configuration > Energy data source" item in the navigation area.
5. Right-click on the "Energy data source" field.
   The selection dialog with PLC/DB tags declared CPU-wide or with the modules of the available measurement hardware opens.
6. Navigate to the desired energy data source in the selection dialog.
7. Apply the select by clicking on the green check mark in the selection dialog.

The energy data types that can be selected in the "Energy data type" area are updated based on the selected energy data source. You can make additional settings for the energy data type (Page 41) in the Inspector window.

The assigned energy data source is also updated within the "Energy data source" column of the energy object table.

In energy object table

To assign or change the energy data source for multiple energy objects when needed, follow these steps:

1. Open the "Energy objects" folder in the project tree.
2. Double-click on the energy object table in which the energy objects were entered.
   The energy object table with the created energy objects opens.
3. Click on the table field in the row of the energy object in the "Energy data source" column.
   The selection dialog with PLC/DB tags declared CPU-wide or with the modules of the available measurement hardware opens.
4. Navigate to the desired energy data source in the selection dialog.
5. Apply the select by clicking on the green check mark in the selection dialog.

The assigned energy data source is also updated in the "Properties" Inspector window. You can make additional settings for the energy data type (Page 41) in the Inspector window.

3.4.6.2 Energy data source

The following table shows the settings of the energy object under "Energy data source":

Setting	Description
Energy data source	The energy data sources available for selection are: • Measurement hardware module with the input measured data of the measuring point or • PLC tag with the values from the process image or DB The energy data sources must be created and configured before parameter assignment of the energy objects. Afterwards, they are available in the selection dialog for the energy data source.
Type	Depending on the selected energy source for the energy object, the corresponding type of energy data source is displayed. The type of energy data source cannot be changed.

Setting	Description
Add error monitoring of energy data source	Error monitoring for energy data sources that do not have integrated error monitoring (e.g. for analog energy values). You program the evaluation of the hardware error in the PLC program for this purpose. If a fault occurs at the input signal of the energy data source, the trigger for hardware error is set. Conventions for the signal: • Minimum duration of 2 times the value at the "Acquisition cycle" setting • Rising edge: incoming error • High-level: Error is present • Falling edge: outgoing error • Low-level: No error The only error events recorded are those that have been pending at least for the duration of the acquisition cycle. You can find additional information on the acquisition cycle in the section Energy data type (Page 41). When this check box is selected, the tag for the trigger for hardware error can be selected.
Trigger for hardware error	Selection of the tag for the trigger for hardware error If an invalid energy value is detected from an energy data source that has no integrated error monitoring, then the selected tag of the "BOOL" data type is set and the "Hardware error" error message is generated. The evaluation of the tag for the hardware error trigger must be programmed accordingly to the conventions for the signal in order to diagnose an incoming error at a falling edge. You can find the conventions for the signal in the description of the setting "Add error monitoring of energy data source". Example: • A measuring device that is not supported by the Energy Support Library provides an analog power value. • In addition, the measuring device provides a binary output which generates a falling edge in case of a fault. Can only be selected when the "Add error monitoring of energy data source" check box is selected.

See also

Assign energy data source (Page 39)

3.4.6.3 Energy data type

Depending on the energy data source of the energy object, the following energy data types are processed.

Energy data type

The following table shows the available energy data types with corresponding units and examples:

Setting	Unit	Example
Pulse signal	Wh l hl m³ m³ [n] kg t pcs	With a normalization factor of 0.5, six pulses correspond to 3 Wh.
Energy counter value	Wh l hl m³ m³ [n] kg t pcs	The counter states of an energy counter change in ten seconds from 6,734,392 Wh to 6,734,414 Wh. The difference corresponds to an energy consumption of 22 Wh and an average power consumption of 7.92 kW. Resetting the energy count value during an active archiving period results in the calculation of incorrect energy values for this archiving period. The following applies for a valid energy counter value: • Energy counter value < Overflow value
Power value	W l/h hl/h m³/h m³/ h [n]kg/h t/h pcs/h	Average power consumption of 180 kW over ten seconds corresponds to energy consumption of 0.5 kWh or 500 Wh.

Acquisition cycle

The driver blocks for the acquisition of energy data are called once per acquisition cycle.

The following acquisition cycles can be selected:

- 0.01 s
- 0.02 s
- 0.05 s
- 0.1 s
- 0.2 s
- 0.3 s
- 0.5 s

- 1 s
- 2 s

The following default values are set:

- Pulse signal: 0.01 s
- Energy counter value: 1 s
- Power value: 0.1 s

Settings for selected energy data type

The following table shows the additional settings for the selected energy data type:

Setting	Description
Data type	Only visible with energy data type "Energy counter value" The data type of the energy counter value is displayed depending on the selected energy data source that provides the energy counter value. The data type cannot be changed.
Unit	Physical unit of the energy value • Units for pulse signal: Wh, l, hl, m³, m³ [n], kg, t, pcs • Units for energy counter value: Wh, l, hl, m³, m³ [n], kg, t, pcs • Units for power value: W, l/h, hl/h, m³/h, m³/h [n], kg/h, t/h, pcs/h
Normalization factor	Specifies the value by which the acquired energy value is multiplied to convert this energy value into the selected unit (e.g. "W" or "Wh" or "m³" or "m³/h"). Preset to 1.0 by default. Permissible value range: 0 – 9999999 Example: A measuring instrument provides an output value of 3.7 with the "kWh" unit. Settings for this in the Energy Suite: • Unit: Wh • Normalization factor: 1000 Energy Suite processed the value to 3700 Wh.
Energy flow direction	Configuration of the energy flow direction • Consumer The energy is drawn from the power grid by the consumer (for example, an electrical machine in motor mode). • Producer The energy is fed into the power grid by the producer (for example, an electric machine in producer mode).

Setting	Description
Overflow value	Only for "energy counter value" energy data type Specifies the value at which the energy counter value is reset to 0. 1000000 (1*10^6) is set by default. Assignable values: • 1000000 (1*10^6) • 1000000000 (1*10^9) • 1000000000000 (1*10^12) • User-defined
User-defined overflow value	Configurable only with overflow value = "User-defined" Manual specification of the overflow value at which the energy counter value is reset to 0. Permissible value range: 1 - 1.0e+15

See also

Assign energy data source (Page 39)

3.4.7 Periodic energy values

3.4.7.1 Setting energy values for periodic archiving

The acquired energy data are archived after processing as periodic energy values together with their time stamps. Depending on the selected energy data type, different energy values can be set for archiving.

During the configuration of the energy object, you select the periodic energy values and the archiving period in the Inspector window for this.

Requirement

- The archiving periods with different values are created at a central location in the settings for the energy program.
  You can find additional information on this in the section "Defining archiving periods (Page 49)".

Procedure

To define the periodic energy values for an energy object for archiving, follow these steps:

1. Open the "Energy objects" folder in the project tree.
2. Double-click on the energy object table in which the energy object has been created.
   The energy object table with the created energy objects opens.
3. Click on the desired energy object in the energy object table.
   The properties of the energy object are displayed in the "Properties" Inspector window.
4. Select the "Configuration > Periodic energy data" item in the navigation area.

5. Select the check box for the desired energy values (Page 45):
   - Power consumption
   - Energy consumption
   - Energy counter value (can only be enabled for the energy counter value energy data type)
6. Right click on the "Archiving period" field.
   The table with the centrally created archiving periods (Page 49) for the energy program opens. If no custom archiving periods have been created previously, there are only two archiving periods (15 min., 60 min.) to choose from by default.
7. Click on the desired archiving period and apply the selection by clicking the green check mark in the selection dialog.

3.4.7.2 Periodic energy data

The acquired energy data are archived after processing as prepared periodic energy values together with their time stamps. Depending on the selected energy data type, different energy values are available for archiving.

Periodic energy values

The following table shows the periodic energy values for archiving:

Setting	Description
Power consumption	Average power consumption per archiving period. Archiving for the following selected energy data types possible: • Pulse signal • Energy counter value • Power value
Energy consumption	Energy consumption per archiving period. Archiving for the following selected energy data types possible: • Pulse signal • Energy counter value • Power value
Energy counter value	Energy counter value per archiving period. Archiving for the following selected energy data types possible: • Energy counter value

Assignment of the archive period

Selecting the archiving period assigns the energy object and thus the selected energy value of the periodic time interval for calculating and archiving as well. You create new archiving periods with the central settings for the energy program (Page 49). You assign exactly one archiving period to each energy object.

Note

Avoid using the same measurement hardware multiple times as energy data source

Make sure that the same measurement hardware is used only once as energy data source in the energy program to prevent overload of the communication channels.

Example for assignment of multiple archiving periods

To archive the energy consumption, for example, with a 15-minute and a 60-minute archiving period, create an energy object for each archiving period.

1. For the energy object with the 15-minute archiving period, select the measurement hardware as energy data source and activate archiving of the energy consumption. Then generate the energy program.
2. For the energy object with the 60-minute archiving period, select the "energy" output parameter of the driver block of the measurement hardware as energy data source. Next, activate archiving of the energy consumption for the energy object and generate the energy program again.

In this way, the energy consumption of a measuring point is archived in two archiving periods, and the measurement hardware is used only once as energy data source in the energy program.

See also

Setting energy values for periodic archiving (Page 44)

3.4.8 Processing

3.4.8.1 General processing settings

The following table shows the settings of the energy object under "Processing":

Setting	Description
General processing settings	
Internal processing cycle	For each internal processing cycle, the energy data activated for archiving is calculated and refreshed once. The following duration can be selected for the internal processing cycle: • 0.1 s • 0.2 s • 0.5 s • 1 s The internal processing cycle can be changed for faster updating of the energy data, for example for: • Visualizing energy data • Further processing of the energy data by the user program (for example, load management) Set to 1 s by default.
Calculation of current value of power consumption	
Calculation interval	Used for calculation of current value of power consumption. The current value of power consumption is calculated from the energy consumption for each calculation interval. The calculated instantaneous value corresponds to the average over the duration of the calculation interval. Can only be edited when energy data type "Energy counter value" or "Pulse signal" is selected. The following duration can be selected for the calculation interval: • 1 s • 2 s • 3 s • 4 s • 5 s • 6 s • 10 s • 15 s • 20 s • 30 s • 60 s 10 s is set by default.

Setting	Description
Extend calculation interval	The dynamic extension of calculation interval allows the calculation of current value of power consumption with additional acquisition time. Example: A pulse is output every 4 s with constant power consumption. When a default calculation interval of 10 s is configured • 2 pulses are counted in the first calculation interval instead of 2 1/2 • 3 pulses are counted in the second calculation interval With an extension of the calculation interval of 5 s • 3 pulses are counted in the first calculation interval • 3 pulses are counted in the second calculation interval The second calculation interval starts after 12 s beginning with the first calculation interval.
Extension of calculation interval	Calculation of current value of power consumption starts when one of the following requirements is met: • The energy counter value has changed. • At least the duration of the extension of calculation interval has expired. • The consumer has been switched off. The following duration can be selected for the extension of calculation interval: • 1 s • 2 s • 3 s • 4 s • 5 s • 6 s • 10 s • 15 s • 20 s • 30 s • 60 s

Buffering and archiving energy data 4

4.1 Defining archiving periods

Archiving periods

Archiving periods are periodic time intervals in which the raw energy data of the respective energy object in the CPU are processed to a periodic energy value with a time stamp and then - depending on the setting - archived in the CPU (SIMATIC Memory Card) or in WinCC Professional (Data Log) (for example, every 15 minutes).

To ensure plant-wide energy values from all energy objects are archived at the same time, the configured duration of the archiving period can be synchronized to an internal or external timer:

- By default, the internal CPU clock is used as a timer for all blocks of the energy program. For this, you synchronize the CPU time with an external time server (for example, NTP). You can find additional information on this in the online help of the TIA Portal information system under the key "Time-of-day synchronization (S7-1500)".
- To synchronize with an external clock (for example, synchronization signal of a utility company), enable the external pulse and configure its pulse source.

You create the various archiving periods at a central location in the settings for the energy program. When you configure the energy object, you then assign the respective archiving period to each individual energy object.

You can find additional information about the naming conventions of archiving periods in the section "Performance features and conventions (Page 19)".

UTC

Note

Set the time of the CPU to UTC, for example, to avoid a time difference when switching to daylight saving time.

Setting the time of day for the CPU corresponds to the standard procedure and therefore occurs outside the Energy Suite in the online and diagnostics view of the CPU.

You can find additional information on this in the online help of the TIA Portal information system in the section "Determining and setting the time of a CPU".

Procedure

To create new archiving periods in the central administration dialog, follow these steps:

1. Open the "Energy objects" folder in the project tree.
2. Double-click on the "Energy program settings" object.
   The dialog with the settings for the energy program opens in the working area.

3. Select "Archiving periods" in the navigation area.
   The table with the central management of the archiving periods opens. Each row of the table represents an archiving period. If no custom archiving periods have been created previously, the table only contains two archiving periods (15 min., 60 min.) to choose from by default.
4. Only for 24h archiving periods: Use the "Apply time zone settings from the regional settings of the local engineering PC" check box to set the time zone (difference to UTC time) for the synchronization:
   - Activated: The time zone of the local engineering PC is used.
   - Deactivated: The time zone from the following drop-down list is used. Select the desired time zone from the drop-down list.
5. To create another archiving period, double-click "<Add>" in the "Name" column of the table. A new archiving period is created in the table.
6. Create additional archiving periods in the same way, if required.
7. Make the settings for each archiving period:
   - Name of the archiving period
   - Duration of archiving period
   - Unit for duration of archiving period
   - Synchronization with external pulse (timer) and selection of the pulse source.

The meaning of the settings is described in detail in the section "Archiving periods (Page 51)".

4.2 Archiving periods

You create the various archiving periods at a central location in the settings for the energy program. When you configure the energy object, you then assign the respective archiving period to each individual energy object.

Two archiving periods (15 min., 60 min.) with an internal clock for the synchronization are preset by default.

Time zone

The following table shows the time zone settings for synchronization under "Archiving periods". The time zone settings are only valid for the synchronization of the archiving periods with a duration of 24 hours.

Setting	Description
Apply time zone settings from the regional settings of the local engineering PC	• Activated: The 24 hour archiving periods are synchronized based on the time zone settings from the regional settings of the local engineering PC. • Deactivated: Selection of the required time zone for synchronization of the 24 hour archiving periods from the drop-down list. Deactivated by default.
Time zone	Time zone (difference to UTC time) Archiving periods with a duration of 24 h are synchronized at 0:00:00:000 UTC time. In case of an offset between UTC time and regional time, synchronization of the 24 h archiving period can be adjusted based on the selected time zone. The time zone can only be selected when the "Apply time zone settings from the regional settings of the local engineering PC" check box is cleared.

Archiving periods

The following table shows the settings for new archiving periods in central administration:

Setting	Description
Name	Two archiving periods (15 min., 60 min.) with an internal clock for the synchronization are preset by default. If different values are needed for the duration, create additional archiving periods. The name of the archiving period is unique CPU-wide and can be changed according to individual requirements. Recommendation: Use in the name of the duration of the archiving period (for example, UserPeriod_15min) You can find additional information about naming conventions in the section "Performance features and conventions (Page 19)".
Duration	Period at the end of which the energy values are periodically calculated and archived. The time stamps are generated based on the time period. Example: With an archiving period of 15 min., time stamps are generated in 15 min. increments (starting at the top of the hour). The prepared periodic energy values are archived together with the respective time stamps: • As a CSV file internal to the CPU on the SIMATIC memory card or • In the data log of the WinCC Professional SCADA system 15 min is set by default for newly added archiving periods.
Unit	Unit for the time period in which a periodic energy value is calculated and stored. The following units can be set: • min = minutes • h = hours
External pulse	An external pulse triggers the clock for synchronization of the archiving period. • With selected check box, you assign the synchronization signal (e.g. from a utilities company) to the external pulse in the "Pulse source" column. • When the check box is not selected, the internal CPU clock triggers the synchronization of the archiving period. The internal CPU time is synchronized with an external time server to UTC time (for example, with NTP) to form the time stamp. Deactivated by default.
Pulse source	A tag of data type "BOOL" can be selected as the pulse source. The tag must be assigned to the synchronization signal, for example, of the utilities company. Can only be edited when the external pulse is enabled.

See also

Defining archiving periods (Page 49)

4.3 Setting the archiving on a S7-CPU (SIMATIC Memory Card)

The prepared periodic energy values with time stamp are cached in the data blocks for buffering before they are archived on the S7-1500 CPU (on an inserted SIMATIC memory card).

The number of data blocks for buffering corresponds to the number of energy object tables.

Then the energy values in are stored in data logs (CSV files) on the S7-1500 CPU (SIMATIC memory card). One data record is written to the data log per archiving period. A data record contains an energy value with time stamp and status.

You can find general information on data logs in the online help of the TIA Portal information system under "Data Logging - Overview".

You define the number and size of the data log files for archiving on the memory card.

Requirement

- The prepared periodic energy values with time stamp (power and energy consumption, energy counter value) are enabled for archiving.
  You can find additional information on this in the section "Setting energy values for periodic archiving (Page 44)".

Procedure

To make the required settings for archiving on the S7-1500 CPU, follow these steps:

1. Open the "Energy objects" folder in the project tree.
2. Double-click on the "Energy program settings" object.
   The dialog with the settings for the energy program opens in the working area.
3. Select the entry "Archiving and buffering" in the area tree.
4. In the "Archiving" area, select the option button "PLC (SIMATIC memory card)".
   - The parameters for the data log are displayed below the option button.
   - The number and size of the data blocks for buffering are shown in the "Buffering" area.
5. In the "Archiving" area, make the settings for the data logs (data archives in CSV format) for the transfer to the memory card:
   - Name of the data log
   - Number of data logs
   - Size of the data log
6. In the "Buffering" area, check the parameters displayed for the size and the memory area of the data blocks.

The meaning of the settings is described in detail in the section "Archiving and buffering (S7-CPU) (Page 54)".

4.4 Archiving and buffering (S7-CPU)

Archiving

The prepared periodic energy values with time stamp are archived in data logs (CSV files) on the S7-1500 CPU (SIMATIC memory card). You can find general information on data logs and the structure of CSV files in the online help of the TIA Portal information system under "Data Logging - Overview".

You define the number and size of the data log files for archiving on the SIMATIC memory card.

The following table shows the settings in the area "Archiving" for the "PLC (SIMATIC Memory Card)" archiving type:

Setting	Description
Name of the data log	The name of the data log is used for file name of the CSV file (comma separated values). The file name consists of the name (as prefix) and the time stamp of the first energy value in the data log. The following restrictions apply to the name of the data log with the S7-1500 CPUs: • Permissible name length: 1 - 29 characters • The following characters are allowed: 0 ... 9, a ... z, A ... Z, "_", "-" You can find additional information about naming conventions in the section "Performance features and conventions (Page 19)".
Number of data logs	The number of data logs is limited by the capacity of the SIMATIC memory card. If the selected number of data logs is reached, the oldest log data is deleted and a new data log created. Valid number of data logs: 2 - 10 Set to 5 by default.
Size of the data log	If a data log approaches the specified size, a new data log is created. The maximum size of the data log is limited by the system to 1024 MB. The required space on the SIMATIC memory card for a data log is calculated using the following formula: 45 bytes + (number of data records * 190 bytes) A data record contains an energy value with time stamp and status. Recommendation: Set the value for the "Size of data log" setting no larger than 180 MB in order to avoid data loss if the CSV file is re-used, for example, with "Microsoft Excel". Set to 10 MB by default.

Buffering

The periodic energy values are cached along with their time stamps in the data blocks for buffering before they are archived on the S7-1500 CPU (SIMATIC memory card). The caching avoids loss of data - for example in the event that the SIMATIC memory card is full because the configured number of data logs has been reached.

The following table shows the settings in the area "Buffering" for the "PLC (SIMATIC Memory Card)" archiving type:

Setting	Description
Number of data blocks	The number of data blocks corresponds to the number of created energy object tables. One data block for buffering is required for every energy object table. The data blocks are automatically created with the generation of the energy program.
Size of the data blocks	By using larger data blocks, you have the option to extend the buffer period of the energy values on the CPU, for example. Set to 10 KB by default.
Allocated memory	Product of number multiplied by the size of the data blocks

See also

Example for archiving (S7 CPU) (Page 56)

Setting the archiving on a S7-CPU (SIMATIC Memory Card) (Page 53)

4.5 Example for archiving (S7 CPU)

The required space on the SIMATIC memory card for a data log is calculated using the following formula:

45 bytes + (number of data records * 190 bytes)

The maximum size of the data log is limited by the system to 1GB. In this way, you can archive 5,500,000 data records of an energy object in which archiving of the power consumption, energy consumption and energy counter value is enabled to a data log.

This corresponds to an archiving over 19097 days (52 years) for an energy object, or 95 days for 200 energy objects with the same configuration.

4.6 Setting the archiving on a PC station with WinCC Professional (Data Log)

The prepared periodic energy values with time stamp are cached in the data blocks for buffering before they are archived on the WinCC Professional SCADA system Then the energy values are transferred to the data log on the WinCC Professional SCADA system. This type of buffered communication enables secure archiving and thus a significant increase in the quality and consistency of invoice-related energy data.

The number of data blocks for buffering corresponds to the number of energy object tables.

Note

WinCC Professional (data log)

You select a corresponding check box in the Energy Suite to archive the buffered energy values in data logs of WinCC Professional.

The data logs of WinCC Professional are set up in the settings of WinCC Professional.

Requirement

- The desired periodic energy values (power and energy consumption, energy counter value) are enabled for archiving.
  You can find additional information on this in the section "Setting energy values for periodic archiving (Page 44)".

Procedure

To make the required settings for archiving to the data log of WinCC Professional on the PC station, follow these steps:

1. Open the "Energy objects" folder in the project tree.
2. Double-click on the "Energy program settings" object.
   The dialog with the settings for the energy program opens in the working area.
3. Select the entry "Archiving and buffering" in the area tree.

4. In the "Archiving" area, select the option button "WinCC Professional (data log)".
   - Below the option button, a notice appears telling you that the setting of the data log for this archiving type must be made in WinCC Professional.
     You can find additional information on this in the section "Create archiving structure for energy values (Page 97)".
   - You can use the "Advanced communication settings (between PLC and WinCC)" check box to make additional optional settings for checking the data transmission and the connection between the PLC and WinCC Professional.
   - The number and size of the data blocks for buffering are shown in the "Buffering" area. The "Number of data blocks" output field indicates the number of data blocks. This number corresponds to the number of energy object tables created and cannot be changed.
5. In the "Buffering" area, specify the size of the data blocks for buffering.
   The "Occupied memory" output field shows the product of the defined number multiplied by the size of the data blocks.

The meaning of the settings is described in detail in the section "Archiving and buffering (WinCC Professional) (Page 59)".

4.7 Archiving and buffering (WinCC Professional)

Archiving

The prepared periodic energy values with time stamp are archived in data logs on the WinCC Professional SCADA system.

Note

WinCC Professional (data log)

The data logs of WinCC Professional are configured in the settings of WinCC Professional.

You can find additional information on this in the section "Create archiving structure for energy values (Page 97)".

The following table shows the advanced communication settings (between PLC and WinCC) for the selected archiving type "WinCC Professional (Data Log)":

CAUTION

Advanced communication settings

You do not need to change the advanced communication settings configured by default.

If you have any questions about archiving, contact Service & Support.

Setting	Description
Advanced communication settings (between PLC and WinCC)	Can only be activated for the archiving type "WinCC Professional (data log)" When the check box is selected, the settings under the following areas can be configured: • Monitoring of connection • Handshake for data transfer Deactivated by default.
Monitoring of connection	

Setting	Description
Monitoring time of connection	Monitoring time of connection for watchdog The connection between the CPU and WinCC Professional is monitored by the watchdog counter as follows: • When the connection is established, the Energy Suite component in WinCC Professional acknowledges the request for comparison of the watchdog counter. • If no acknowledgment is made within the monitoring time for the watchdog, the connection is interrupted and an error message is generated. The following duration can be selected for the monitoring time of connection: • 1 s • 2 s • 5 s • 10 s • 20 s 5 s are set by default.
Handshake for data transfer	

Setting	Description
Monitoring time	Monitoring time for writing the energy data to the data log Handshake for data transfer takes place under the following conditions: • The Energy Suite component acknowledges the request for writing the energy data to the data log of WinCC Professional within the monitoring time. • No monitoring time error has occurred for the watchdog. The following duration can be selected for the monitoring time: • 10 s • 20 s • 30 s • 60 s • 120 s • 180 s • 240 s • 300 s • 600 s • 900 s The default setting is 30 s.
Wait time	Wait time for repeating the request to write the energy data to the data log The request is repeated until the following conditions are met: • Wait time has expired. • Energy Suite component acknowledges the request. The following duration can be selected for the wait time: • 10 s • 20 s • 30 s • 60 s • 90 s • 120 s 60 s is set by default.

Note

Sufficient watchdog time

Adapt the watchdog time to the number of energy objects. The more energy objects are configured, the longer the watchdog time needs to be.

Buffering

The periodic energy values are cached along with the associated time stamp in the data blocks (DBs) for buffering before they are archived in the data log of WinCC Professional. Caching avoids loss of data - for example, in the event that the connection to WinCC Professional is interrupted.

You specify the size of the data blocks (buffer DBs) for the duration of the buffering of the periodic energy values. The number of archived energy values and the configured archiving period are relevant for the maximum duration of buffering.

The following table shows the settings in the area "Buffering" for the "WinCC Professional (Data Log)" archiving type:

Setting	Description
Number of data blocks	The number of data blocks corresponds to the number of created energy object tables. One data block for buffering is required for every energy object table. The data blocks are automatically created with the generation of the energy program.
Size of the data blocks	By using larger data blocks, you have the option to extend the buffer period of the energy values on the CPU, for example, in the event of failure of the communication to the WinCC Professional SCADA system. Set to 10 KB by default.
Allocated memory	Product of number multiplied by the size of the data blocks

See also

Examples of archiving and buffering (WinCC Professional) (Page 63)

Setting the archiving on a PC station with WinCC Professional (Data Log) (Page 57)

4.8 Examples of archiving and buffering (WinCC Professional)

Examples for buffering

The following examples show the relationship between the number of energy object tables and the duration of the buffering, depending on the CPU. The examples show the calculation of one or two energy object tables.

- The duration of the buffering depends on the number of energy object tables, because exactly one data block for buffering is created for each energy object table.
- The assignment of individual energy objects to the energy object table also has an effect on the maximum buffer duration.

The following table shows the maximum size of the data block for buffering and the default values for the duration of the buffering for a data record (e.g. power consumption) with a 15 min. archiving period in various S7-1500 CPUs: The size of an archive value including the time stamp is 150 bytes.

CPU	Maximum size of the DB [MB]	Duration of buffering [d]
CPU 1511	1	72
CPU 1516	5	375
CPU 1518	10	750
CPU 1510SP	0.75	52
CPU 1512SP	1	72

Example with one energy object table

The duration of the buffering is calculated based on the following constraints:

- CPU used: CPU 1518
- Maximum size of the data block for buffering: 10 MB
- Number of energy object tables: 1
- Number of energy objects: 50
- Energy data to archive per energy object:
  - Power consumption
  - Energy consumption
- Duration of the archiving period: 15 min
- Duration of buffering: 7 1/2 days

The duration of the buffering is calculated as follows:

750 days / 2 energy data items / 50 energy objects = 7 1/2 days

Example with two energy object tables

The duration of the buffering is calculated based on the following constraints:

- CPU used: CPU 1518
- Maximum size of the data block for buffering: 10 MB
- Number of energy object tables: 2
  - Per energy object table: 25 energy objects
- Number of energy objects: 50
- Energy data to archive per energy object:
  - Power consumption
  - Energy consumption
- Duration of the archiving period: 15 min
- Duration of buffering: 15 days

The duration of the buffering is calculated as follows:

1500 days / 2 energy data items / 50 energy objects = 15 days

The duration of the buffering directly depends on the number of energy object tables, because exactly one data block for buffering is created for each energy object table.

Generate energy program 5

5.1 Making general settings for the energy program

Once the energy objects have been configured and settings for buffering and archiving have been made, define the following general properties for the energy program before the generation if required:

- Enable for alarm suppression
- Startup time of the Energy Suite blocks

Procedure

To open the dialog for general settings of the energy program, follow these steps:

1. Open the "Energy objects" folder in the project tree.
2. Double-click on the "Energy program settings" object.
   The "General" dialog with the settings for the energy program opens in the working area.
3. Make the desired settings in the "General" dialog. The meaning of the parameters is described in detail in the section "General settings (Page 66)".

See also

Defining archiving periods (Page 49)

Setting the archiving on a S7-CPU (SIMATIC Memory Card) (Page 53)

Setting the archiving on a PC station with WinCC Professional (Data Log) (Page 57)

5.2 General settings

The following table shows the settings of the energy program under "General":

Setting	Description
Alarm suppression	Selection of the tag (data type "BOOL") for higher-level alarm suppression for the energy program. This allows block alarms to be suppressed, for example, at restart of the CPU after a voltage failure.
Startup time	Duration for which the Energy Suite blocks remain disabled after restarting the CPU. 10000 ms is set by default. Example: The startup time is used to prevent field devices that are still starting up from causing errors or the reporting of errors due to the startup. When the initialization of the field devices is complete (startup finished), the field devices are ready for operation. The setting of the startup time depends on the number of devices connected to the CPU. The more devices on the PROFINET line there are, the longer the startup time must be set.

See also

Making general settings for the energy program (Page 65)

Archiving periods (Page 51)

Archiving and buffering (S7-CPU) (Page 54)

Archiving and buffering (WinCC Professional) (Page 59)

5.3 Runtime licenses for Energy Suite

Energy Suite requires Runtime licenses on the CPU based on the number of configured energy objects.

The following runtime licenses can be set for the Energy Suite:

- "5 energy objects" license type
- "10 energy objects" license type

The available runtime licenses for the Energy Suite are displayed for the CPU configuration in the "Properties" Inspector window under:

Runtime licenses > Energy Suite > Runtime licenses > Number of purchased licenses

---

Note

Runtime licenses

Before generating the energy program, ensure that sufficient runtime licenses are available on your engineering system. Disabled energy objects are not included in a runtime license.

Lacking or insufficient runtime licenses result in the termination of the program generation.

Example

The following figure shows the configuration of runtime licenses for the following example:

If you use 32 energy objects in the energy program, you need:

- 1 * "5 energy objects" license type and 3 * "10 energy objects" license type

or

- 7 * "5 energy objects" license type

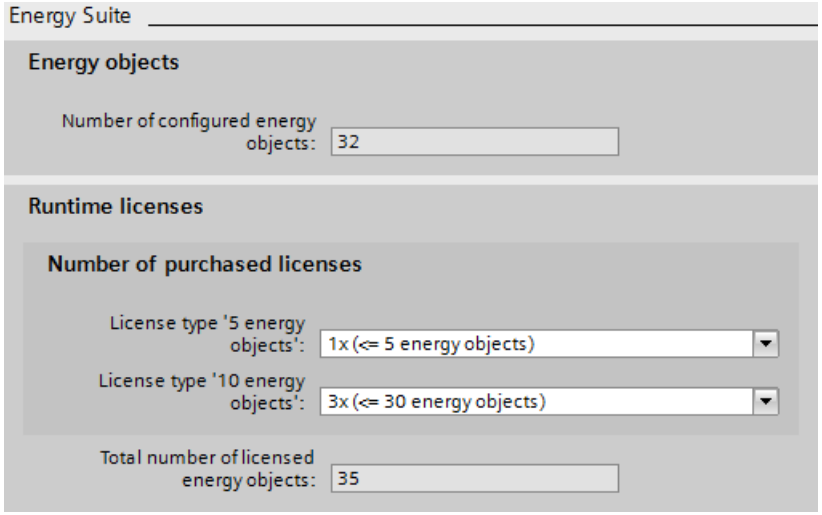

5.4 Generate energy program

Once the energy objects have been fully configured and settings for the energy program have been made, you generate the energy program based on these guidelines.

Requirement

- All energy objects that should be included are enabled for the energy program.
- The energy objects enabled for the energy program are fully configured.
- The settings for the energy program are defined.
- The number of required runtime licenses for Energy Suite is sufficient.
  You can find additional information on this in the section "Runtime licenses for Energy Suite (Page 67)".
- The name structures in the SCADA system do not exceed the maximum permissible length for the CPU name of 88 characters.

Procedure

To generate the energy program, follow these steps:

1. Open the S7-1500 CPU in the project tree.
2. Click on the "Energy objects" object below the S7-1500 CPU and select the command for the generation from the shortcut menu. You can select three options:
   - Generate energy program > Basic energy data
   - Generate energy program > Basic and advanced energy data
   - Generate energy program > Basic and advanced energy data (with minimum/maximum values)

   The generator checks the completeness and the plausibility of the configuration of the energy objects and starts the generation of the energy program. Information and events during the generation of the energy program are displayed in the Inspector window under ""Info > Energy Suite".
3. You can use the "Go to" function to jump to the position in the configuration that is relevant for the respective event.
4. Adapt the configuration if necessary and start the generation of the energy program again.

Result

After error-free and full configuration of energy objects, instances of the required blocks are created from the Energy Suite Library and the measurement hardware library in the "Program blocks" folder below the S7-1500 CPU. In addition, other required components, such as organization blocks and data blocks, are added to the "Program blocks" folder and the required interconnections are made.

Instances of data types from the Energy Suite Library and the measurement hardware library can be created in the "PLC data types" folder below the S7-1500 CPU.

The energy program is generated for all enabled energy objects (regardless of the division in the energy object tables).

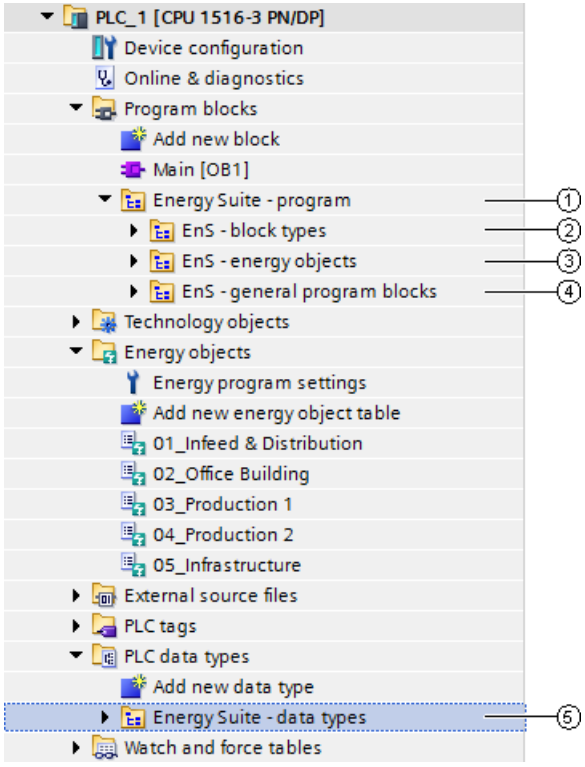

	Object	Description
①	Energy Suite program	New folder for energy program
②	EnS - block types (Page 71)	Folder with the Energy Suite blocks and measurement hardware driver blocks (function blocks)
③	EnS - energy objects (Page 73)	Blocks for energy objects (data blocks)
④	EnS - general program blocks (Page 74)	Folder for additional blocks (organization blocks, functions, buffer DBs, etc.)
⑤	Energy Suite data types (Page 77)	Folder with the Energy Suite data types and measurement hardware data types

See also

Change the enable for energy objects for the energy program (Page 38)

Making general settings for the energy program (Page 65)

Important information on energy data (Page 22)

5.5 Structure of the energy program

5.5.1 Structure of the energy program

When the energy program is generated, the following folders are automatically created below the S7-1500 CPU:

- "Energy Suite Program" folder under "Program blocks" with subfolders
  - EnS - block types (Page 71)
  - EnS - energy objects (Page 73)
  - EnS - general program blocks (Page 74)
- Folder "Energy Suite data types (Page 77)" under "PLC data types"

Note

Blocks of the energy program

SIMATIC Energy Suite automatically generates for you all the required blocks and data types and their interconnection based on the configured energy objects and the energy program settings.

You do not need to further adapt these blocks for the standard application with Energy Suite.

CAUTION

Manual change to blocks of the energy program

If you make manual changes to the blocks after generating the energy program, you yourself are responsible for the error-free operation of the devices and the energy program.

CAUTION

Using Energy Suite blocks in other PLC programs

If you use the Energy Suite blocks from the global library in other PLC programs, you yourself are responsible for error-free use of Energy Suite blocks.

See also

Energy objects and energy program (Page 31)

5.5.2 EnS - block types

The structure of your energy program depends on the following configurations:

- Energy data sources (Page 21)
- Enabled energy objects (Page 36)
- Energy program settings (Page 65)

The following shows how the Energy Suite blocks generally work in the energy program for energy data from the process image of the CPU (PLC tags) or from data blocks (DB tags).

When the input measurement data is received from the connected measurement hardware, the hardware-specific function blocks (FBs) are added with driver functionality. You can find additional information on this in the section "Structure of the energy program for measurement hardware (Page 79)".

How Energy Suite blocks work

The placeholder "xxx" in the picture means that there are different types of this block, for example, for "EnS_Drvxxx":

- EnS_DrvPulse
- EnS_DrvCounterLInt

You can find more types in the "EnS - block types" table after the picture.

The following picture show how the Energy Suite blocks work:

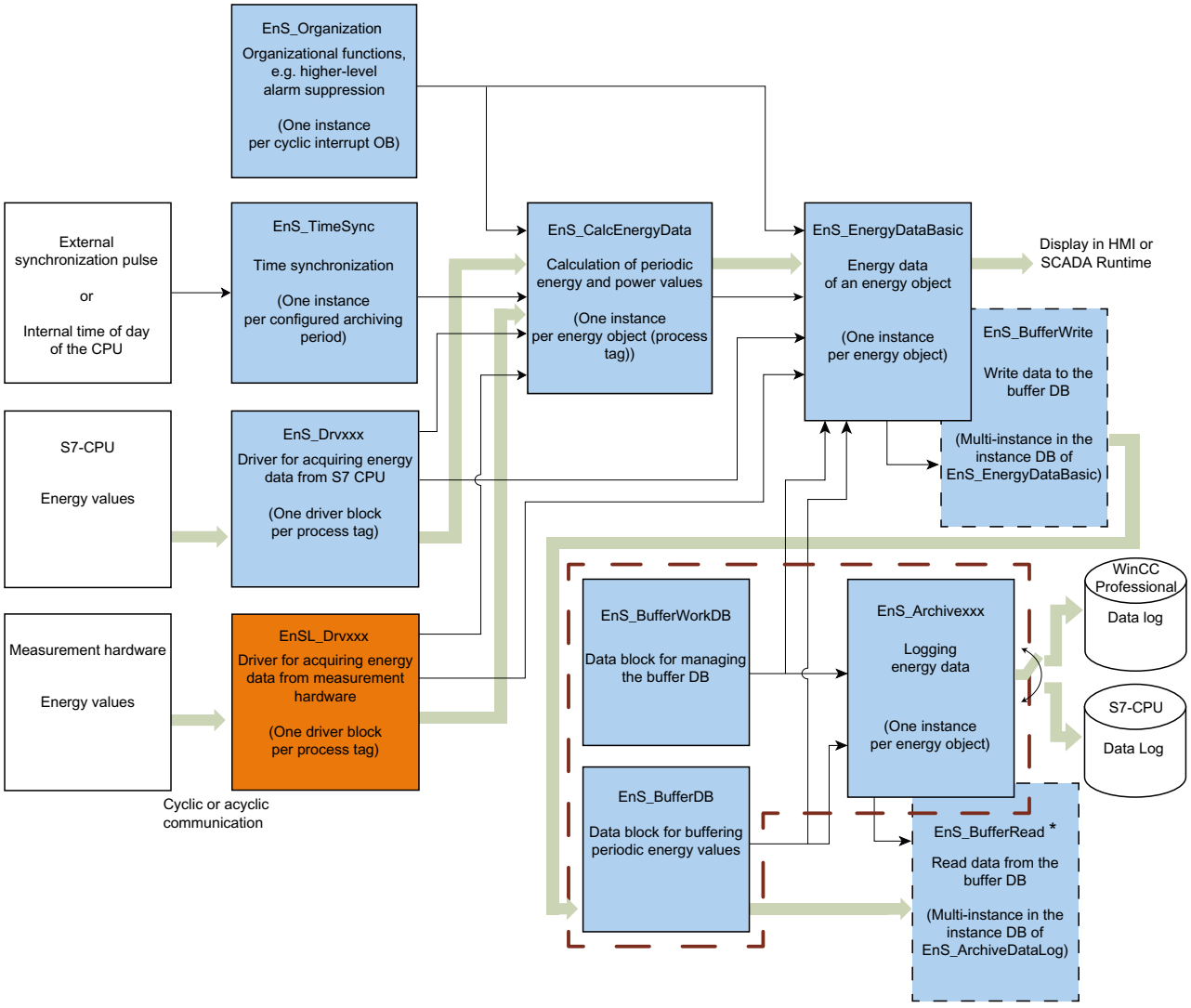

* EnS_BufferRead is not called for archiving in WinCC Professional

Energy data flow

Parameter assignment data

Blocks used internally

Blocks associated with an energy object table

EnS - block types

The "EnS - block types" folder contains the instances of all function blocks from the Energy Suite library once the program is generated. When the measurement hardware acts as the energy data source, the "EnS - block types" folder also contains a subfolder "S7-1500 blocks" with the instances of the associated Driver block from the measurement hardware library (Page 79).

Function block	Description	Example in STEP 7 (TIA Portal)
EnS_ArchiveDataLog (Page 113)	Archived energy data in data logs (CSV file on the SIMATIC memory card of the CPU)	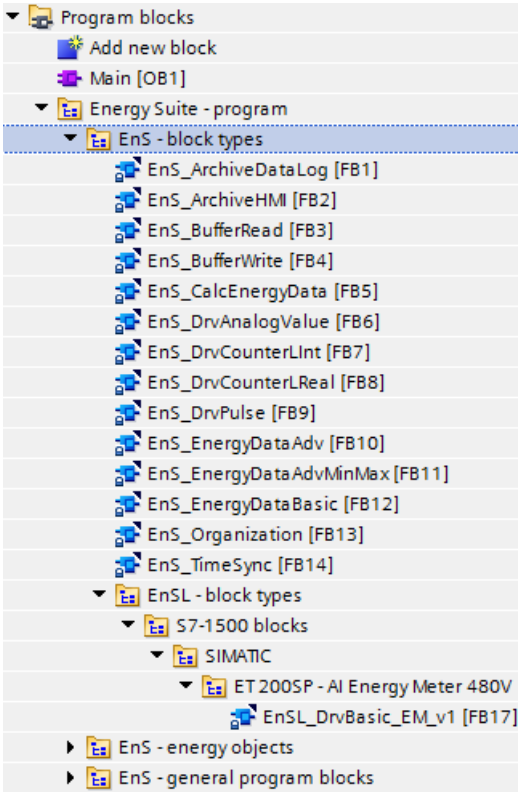
EnS_ArchiveHMI (Page 121)	Archived energy data in data logs of WinCC Professional	
EnS_BufferRead (Page 128)	Reads the data from the buffer DB (EnS_BufferDB (Page 160)). Used internally	
EnS_BufferWrite (Page 130)	Writes data to the buffer DB (EnS_BufferDB (Page 160)). Used internally	
EnS_CalcEnergyData (Page 133)	Periodically calculates the energy used and average power consumption	
EnS_DrvAnalogValue (Page 139)	Driver block for analog input (e.g. power value, flow value)	
EnS_DrvCounterLInt (Page 141)	Driver block for counter values of the LINT type	
EnS_DrvCounterLReal (Page 143)	Driver block for counter values of the LReal type	
EnS_DrvPulse (Page 145)	Driver block for a pulse input	
EnS_EnergyDataAdv (Page 147)	Advanced energy data from a device (for example, phase-specific current and voltage values)	
EnS_EnergyDataAdvMinMax (Page 148)	Minimum and maximum values of the advanced energy data of a device	
EnS_EnergyDataBasic (Page 149)	Basic energy data from a device (for example, power, energy counter value)	
EnS_Organization (Page 153)	Contains general data and parameters for central program organization	
EnS_TimeSync (Page 154)	Synchronizes the times for calculating the energy data	

5.5.3 EnS - energy objects

Once the program is generated, the "EnS - energy objects" folder contains a subfolder with the name of the energy object "EO - <name of the energy object>" for each energy object.

The "EO - <name of the energy object>" folder contains the instance data blocks

- On the function blocks that are required for the energy object in the energy program
- With the periodic energy values of the energy object, which are provided for buffering and archiving

EO - <name of the energy object>

Depending on the configuration of the energy object, the "EnS_<name of the energy object>" folder may contain the following data blocks:

Data block	Description	Example in STEP 7 (TIA Portal)
<Name of the energy object>	Data block with information for the measuring point and the energy values that are provided for archiving	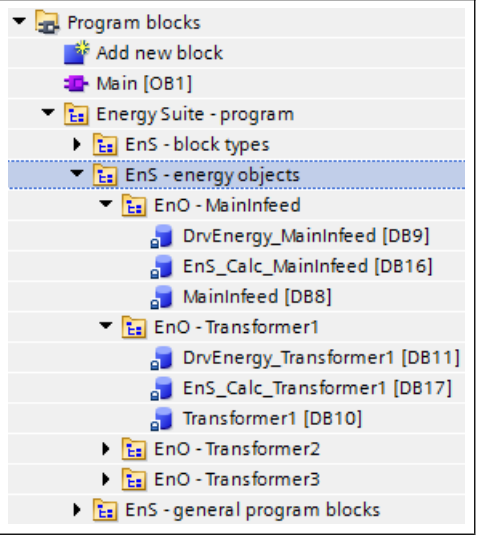
DrvEnergy_<name of the energy object>	Instance DB of the FB "EnS_DrvPulse" Example: The instance DB "DrvEnergy_PulseGenerator" is created by an energy object with the name "PulseGenerator".	
EnS_Calc_<Name of the energy object>	Instance DB of FB "EnS_CalcEnergyData" Example: The instance DB "EnS_Calc_PulseGenerator" is created by an energy object with the name "PulseGenerator".	

5.5.4 EnS - general program blocks

The "EnS - general program blocks" folder contains additional blocks, such as organization blocks, data blocks and functions (FCs) required for the energy program, once the program is generated. The data blocks for buffering the energy data are also created as instances from the Energy Suite library in this folder.

EnS - general program blocks

The "EnS - general program blocks" folder contains the following blocks once the program is generated:

Block	Description	Example in STEP 7 (TIA Portal)
Organization block		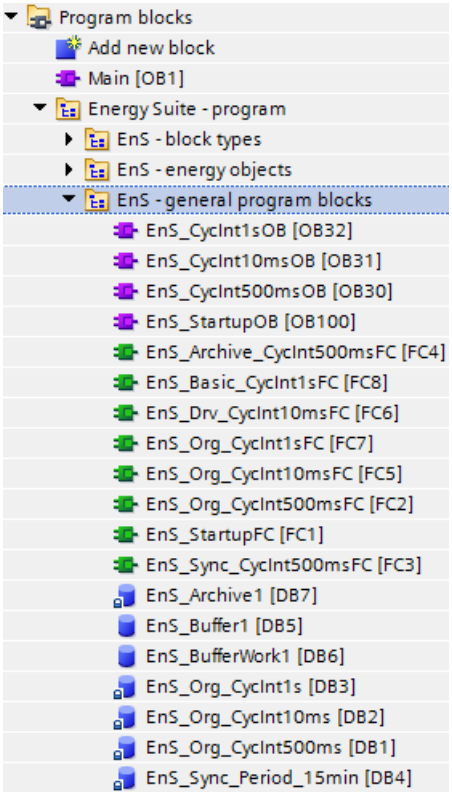
EnS_CycInt<NNNcc>OB	Three cyclic interrupt OBs with different timings. "NNN" and "cc" are placeholders for the clock and the unit. Example: For three clock speeds of 100 ms, 1000 ms and 10000 ms, the following cyclic interrupt OBs are created: • "EnS_CycInt100msOB" • "EnS_CycInt1sOB" • "EnS_CycInt10sOB"	
EnS_StartupOB	Startup OB 100	
Function		
EnS_Archive_CycInt<NNNcc>	An FC for archiving energy data in the cyclic interrupt OB. "NNN" and "cc" are place holders for the duration and unit. Example: With a cycle time of 100 ms, the "EnS_Archive_CycInt100ms" FC is created.	
EnS_Basic_CycInt<NNNcc>	An FC for the basic energy data of energy objects in the cyclic interrupt OB. "NNN" and "cc" are place holders for the duration and unit. Example: With a cycle time of 100 ms, "EnS_Basic_CycInt100ms" is created.	
EnS_Drv_CycInt<NNNcc>	An FC for the driver blocks of energy objects in the cyclic interrupt OB. "NNN" and "cc" are place holders for the duration and unit. Example: With a cycle time of 100 ms, "EnS_Drv_CycInt100ms" is created.	
EnS_Org_CycInt<NNNcc>FC	Three FCs for the central organization of the energy program that are called in the cyclic interrupt OB and startup OB. "NNN" and "cc" are place holders for the cycle duration and unit. Example: For three cycle times of 1 s, 100 ms and 10 s, the following FCs are created: • "EnS_Org_CycInt1sFC" • "EnS_ Org_CycInt100msFC" • "EnS_ Org_CycInt10sFC"	
EnS_StartupFC	FC called in the startup OB.	
EnS_Sync_CycInt<NNNcc>	An FC for the time synchronization. "NNN" and "cc" are place holders for the duration and unit. Example:	

Block	Description	Example in STEP 7 (TIA Portal)
	With a cycle time of 100 ms, the "EnS_Sync_CycInt100ms" FC is created.	
Data block		
EnS_Archive<N>	Instance DB of FB "EnS_ArchiveDataLog". "N" is a placeholder for the number of the buffer. Example: For buffer number "2", the "EnS_Archive2" instance DB is created.	
EnS_Buffer<N> (Page 160)	Data block for buffering energy data records. "N" is a placeholder for the number of the buffer. Example: For buffer number "2", the "EnS_Buffer2" buffer DB is created.	
EnS_BufferWork<N> (Page 161)	Data block for control and status information of the buffer "N" is a placeholder for the number of the buffer. Example: For buffer number "2", the "EnS_BufferWork2" buffer DB is created.	
EnS_Org_CycInt<NNNcc>	Instance DB of FB "EnS_Organization". "NNN" and "cc" are place holders for the cycle duration and unit. Example: With a cycle time of 10 ms, an "EnS_Org_CycInt10ms" instance DB is created.	
EnS_Sync_Period_<NNNcc>	Instance DB of FB "EnS_TimeSync". "NNN" and "cc" are placeholders for the synchronization period and the unit. Example: For a synchronization period of 30 minutes, the "EnS_Time30m" instance DB is created.	

5.5.5 Energy Suite data types

The Energy Suite library contains the information required data types for the energy program.

Energy Suite - data types

The "Energy Suite - data types" folder contains the instances of the following data types blocks from the Energy Suite library once the program is generated:

Data type	Description	Example in STEP 7 (TIA Portal)
EnS_typeAnalogValue (Page 162)	Data type for analog process values	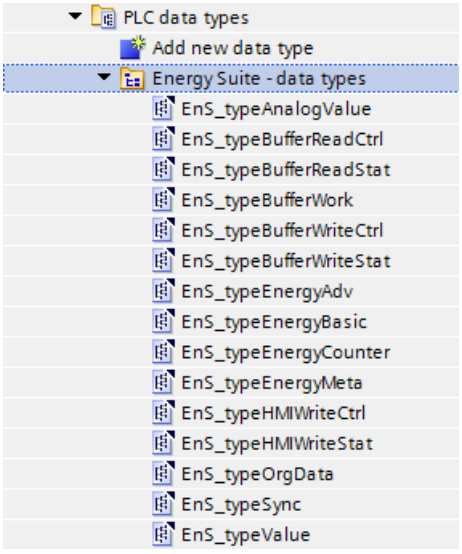
EnS_typeBufferReadCtrl (Page 162)	Data type for control information of the buffer for reading data	
EnS_typeBufferReadStat (Page 163)	Data type for data and status information of the buffer for reading data	
EnS_typeBufferWork (Page 163)	Data type for control and status information of the buffer	
EnS_typeBufferWriteCtrl (Page 164)	Data type for control information of the buffer for writing data	
EnS_typeBufferWriteStat (Page 165)	Data type for status information of the buffer for writing data	
EnS_typeEnergyAdv (Page 165)	Data type for advanced energy data	
EnS_typeEnergyBasic (Page 167)	Data type for basic energy data	
EnS_typeEnergyCounter (Page 167)	Data type for energy counter value	
EnS_typeEnergyMeta (Page 168)	Data type for metadata of the measuring point (information for configuration of the energy object)	
EnS_typeHMIWriteCtrl (Page 171)	Data type for control information of the buffer for writing energy data to the data log of WinCC Professional	
EnS_typeHMIWriteStat (Page 172)	Data type for status information of the buffer for writing energy data to the data log of WinCC Professional	
EnS_typeOrgData (Page 172)	Data type for organizational data	
EnS_typeSync (Page 173)	Data type for time synchronization	
EnS_typeValue (Page 174)	Data type for output values	

5.5.6 Driver blocks for measurement hardware

5.5.6.1 Structure of the energy program for measurement hardware

Driver blocks of the measurement hardware for STEP 7 (TIA Portal)

The energy data (input measurement data) of the measurement hardware are made available in different ways (e.g. differences in the data format or in the data record address). These differences make it difficult to access and further process the energy data from the measurement hardware in the PLC program.

For this, **in addition** to the Energy Suite blocks, hardware-specific function blocks (FBs) with driver functionality are available for the measurement hardware from the Energy Support Library (EnSL). This hardware-specific function blocks (referred to as driver blocks in the following), form the interface between the respective measurement hardware and your automation system.

The energy data from the measurement hardware is read and uniformly provided to the automation system using driver blocks. In this way, the raw energy data is provided easier and faster to your PLC program and Energy Suite. The components of the Energy Suite are then used to prepare and process raw energy data.

Interaction of Energy Suite blocks and driver blocks for measurement hardware

When the energy program is generated, instances of specific driver blocks for the measurement hardware are automatically created and interconnected (depending on the selected energy data sources) in the "EnS block types" folder of the energy program.

The following sections of this document describe the **basic interaction** of the Energy Suite blocks and the specific driver blocks for the measurement hardware in the energy program.

Documentation of the specific driver blocks for the measurement hardware

You can find a **detailed description** about how driver blocks operate and their block parameters in the documentation of the measurement hardware. The documentation for the measurement hardware is part of the Energy Support Library (EnSL).

5.5.6.2 Energy data processing based on the "IPO" principle

Energy data processing according to the "IPO" principle can be divided into "Input" "Processing" and "Output":

How function blocks work

The following figure shows the general operation of the driver blocks for the measurement hardware according to the "IPO" principle:

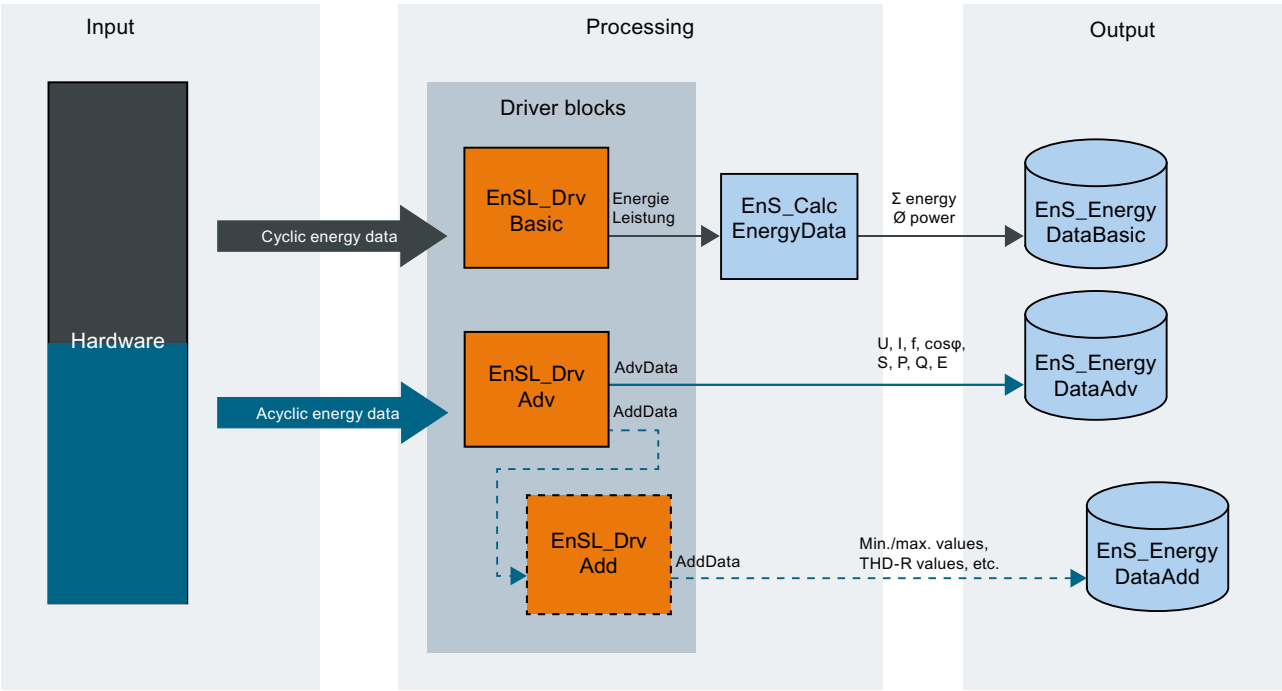

Energy data via cyclic communication (process image)

Energy data via acyclic communication (read data record)

Driver blocks for measurement hardware

Blocks of the SIMATIC Energy Suite

Required blocks

Optional blocks

The following table describes the different areas of the picture:

Area	Description
Input	Contains the measurement hardware (for example, measuring instrument) which provides the input measurement data for processing
Processing	Acquisition, calculation, time stamping and provision of energy values for other systems by the respective function blocks
Output	Provision of periodic energy values for different systems for visualizing, archiving or further evaluation (for example, SIMATIC Energy Manager PRO)

5.5.6.3 Area: Input

The measurement hardware is entered in the "Input" area. The driver blocks read the energy measurement data from the measurement hardware. This is done using cyclical communication (process image) and/or acyclic communication (read data record).

The following figure shows "input" according to the "IPO" principle:

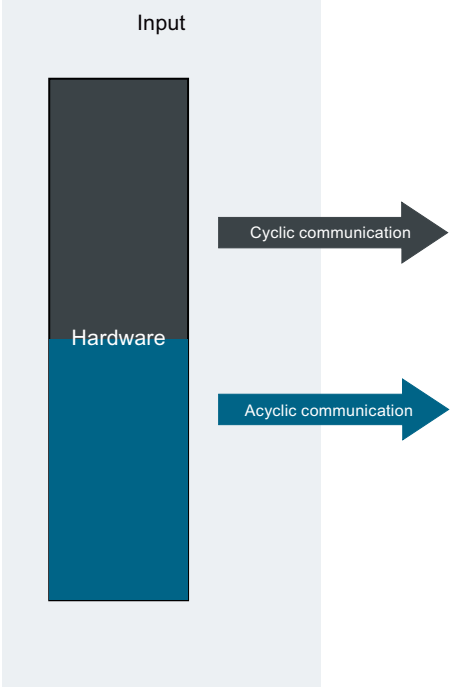

Energy data using cyclic communication (process image)

Energy data using acyclic communication (read data record)

You can find a detailed description of the measurement hardware in the documentation for the respective measuring instrument.

5.5.6.4 Area: Processing

In the "processing" area, the energy data from the respective driver blocks are acquired, normalized and prepared as raw energy data that can be easily further processed.

The "Basic", "Adv" and "Add" blocks are located in the "Driver blocks" area. The "EnSL_DrvBasic" driver block passes the current basic energy data to the Energy Suite function block "EnS_CalcEnergyData (Page 133)". The "EnS_CalcEnergyData" function block calculates the energy consumption and the average power over the individually configured archiving period (e.g. 15 min.).

You can find a detailed description of the driver blocks in the documentation for the respective measurement hardware.

The following figure shows an example of "processing" according to the "IPO" principle. The interconnection of the parameters may vary depending on the measurement hardware used.

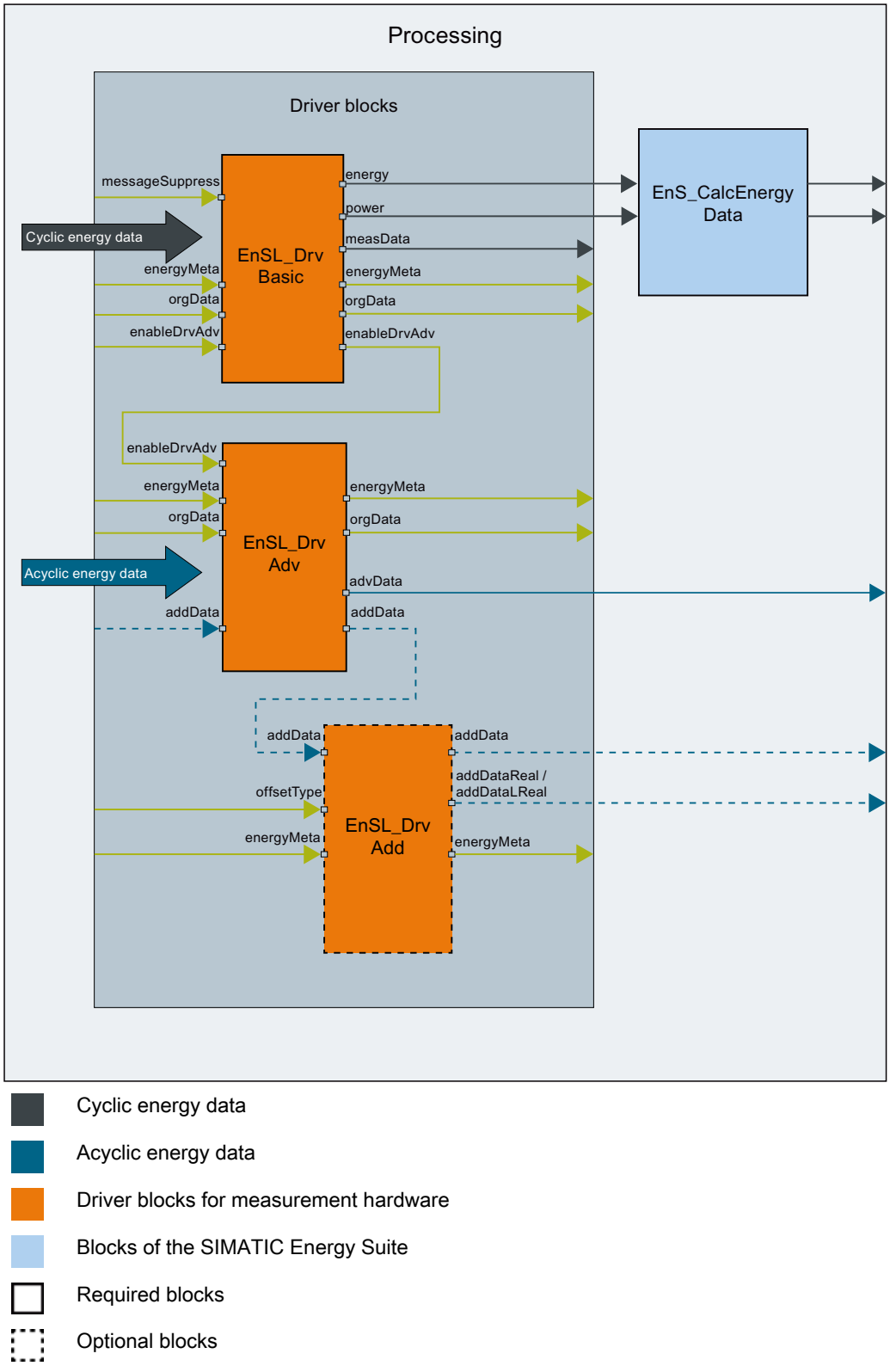

Cyclic energy data

Acyclic energy data

Driver blocks for measurement hardware

Blocks of the SIMATIC Energy Suite

Required blocks

Optional blocks

The following table describes the components of a picture:

Component	Description
EnSL_DrvBasic	Driver block for basic energy data Reads the basic energy data via cyclic communication from the measurement hardware, for example, energy counters, power
EnSL_DrvAdv	Driver block for advanced energy data Reads a data record with advanced energy data via acyclic communication from the measurement of hardware, for example, voltage, current
EnSL_DrvAdd	Driver block for additional energy data Reads a record with additional energy data via acyclic communication from the measurement hardware, for example, harmonics
EnS_CalcEnergyData	Energy Suite block for further processing (periodic calculation, time stamping, etc.) the energy data. You can find additional information on this in the section "EnS_CalcEnergyData (Page 133)".
Cyclic energy data	Predominantly basic energy data that are read by the "EnS_DrvBasic" driver block from the measurement hardware using cyclic communication (process image).
Acyclic energy data	Predominantly advanced and additional energy data that is read by the "EnSL_DrvAdv" and "EnSL_DrvAdd" driver blocks using acyclic communication (read data record).
messageSuppress	Alarm suppression
energyMeta	Process tag specific data You can find additional information on this in the section "EnS_typeEnergyMeta (Page 168)".
orgData	Organization data You can find additional information on this in the section "EnS_typeOrgData (Page 172)".
enableDrvAdv	Processing enable of the "EnSL_DrvAdv" driver block
energy	Current value of the active energy counter
power	Current value of the active power
measValues	Cyclic measurement data that is passed to the "EnSL_DrvAdv" driver block
advData	Extended measured data record
addData	Additional energy data
addDataLReal / addDataReal	Current LREAL/REAL data record

See also

Important information on energy data (Page 22)

5.5.6.5 Area: Output

After acquisition and processing, the energy data are collected in the "Output". The "Output" serves as an interface for visualization with SIMATIC WinCC or other systems (for example, SIMATIC Energy Manager PRO).

The following figure shows "output" according to the "IPO" principle:

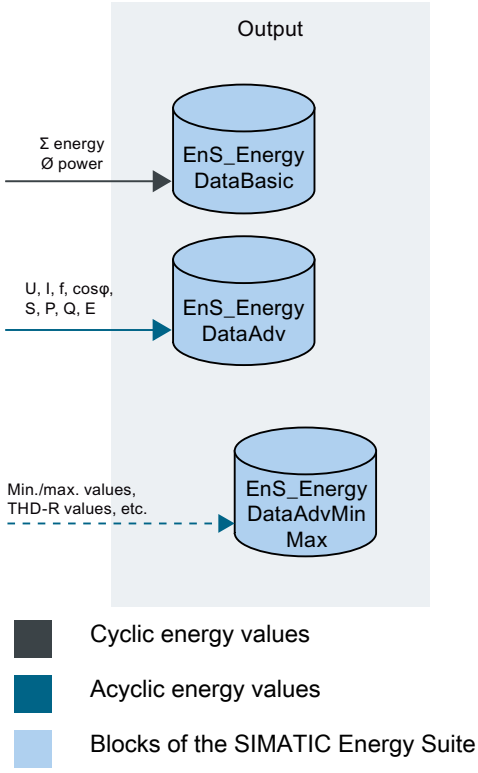

The following table describes the components of a picture:

Function block	Description
EnS_EnergyDataBasic	Energy Suite block for providing the prepared basic energy data and metadata. The cyclic basic energy data are calculated to the following values by the "EnS_CalcEnergyData" Energy Suite block: • Energy consumption over a period (for example, 15 min.) • Average power over a period (for example, 15 min.) You can find additional information under "EnS_EnergyDataBasic: Providing metadata and energy values (Page 149)".
EnS_EnergyDataAdv	Energy Suite block for providing a data record with advanced energy values Advanced energy values (acyclic): • Voltage, current, cos φ • Apparent, active and reactive power etc. You can find additional information under "EnS_EnergyDataAdv: Providing advanced energy values (Page 147)".
EnS_EnergyDataAdvMinMax	Energy Suite block for providing a data record with minimum and maximum values of the advanced energy values Minimum and maximum values of the advanced energy values: • Voltage, current, cos φ • Apparent, active and reactive power You can find additional information under "EnS_EnergyDataAdvMinMax: Providing advanced minimum and maximum values (Page 148)".

5.6 Compiling and downloading the PLC program

Once the energy program is generated, compile the complete PLC program and download it to the S7-1500 CPU target system.

There are several ways to compile and download project data to the CPU. One version is described below. You can find additional information on this in the online help of the TIA Portal information system.

Requirement

- The energy program has been fully generated without error.
- The PLC program is also complete and error free.

Compile

To compile the entire PLC program (including the energy program), follow these steps:

1. Open the S7-1500 CPU in the project tree.
2. Click "Program blocks" object below the S7-1500 CPU and select the command "Compile > Software (rebuild all blocks)" from the shortcut menu.
   The PLC program is compiled. You can check to see if the compiling has been successfully executed in the Inspector window under "Info > Compile".
3. If needed, correct the causes of possible compiling errors and start the compiling again.

Download

To compile the entire PLC program on the CPU, follow these steps:

1. Open the S7-1500 CPU in the project tree.
2. Click "Program blocks" object below the S7-1500 CPU and select the command "Download to device > Software (only changes)" from the shortcut menu.
3. Follow the instructions of the wizard for charging. If required, select the interface for the online connection to the target system.
   The online connection to the CPU is established.
4. Click on the "Download" button and then the "Finish" button.

If the download was successful, the entire PLC program (including the energy program) is loaded into the CPU. If the download was not successful, consult the online help of the TIA Portal information system to learn about the further actions required.

5.7 Testing the energy program

The standard functions of STEP 7 (TIA Portal) enable you to test the acquisition and processing of the energy values.

Requirement

- The energy program has been fully generated without error.
- The PLC program has been successfully compiled and downloaded to the S7-1500 CPU.
- An online connection to S7-1500 CPU is available.
- Program execution is active (CPU in "RUN").

Procedure

To test the energy program, follow these steps:

1. In the "EO - <name of the energy object>" folder, open the data block for the energy object: <name of the energy object> [DBx].
   The contents of the data block with the metadata and energy data of the energy object opens in the working area.
2. Start the monitoring by clicking the "Monitor all" icon.
   The additional "Monitor value" column is displayed in the table. It displays the current data values. You can track and, in needed, control the current periodic energy values under the "energyData" structure

Behavior after regeneration of the energy program

The Energy Suite has the following behavior when the energy program is regenerated:

1. The entire energy program is deleted.
2. The energy program including changes is newly created.
   All unchanged energy objects are created exactly as before.
3. The addresses (symbolic names) of the DBs are changed when the following settings are changed:
   - Name of the energy object (instance DB of EnS_Drvxxx, EnS_CalcEnergyData, EnS_EnergyDataBasic)
   - Name of the measurement hardware used in the hardware configuration (instance DB of EnSL_DrvBasicxxx)
   - Name of the archiving period (instance DB of EnS_TimeSync)

Note

The DB numbers are assigned automatically by STEP 7 (TIA Portal).

5.8 Editing the PLC program without installation of Energy Suite

STEP 7 (TIA Portal) projects that have configured energy objects and parts of an energy program can be opened and edited on STEP 7 (TIA Portal) computers on which there is no installation of Energy Suite.

This allows you to further edit the device configuration or the rest of the PLC program, for example. The existing configuration of energy objects and the generated energy program remain unchanged in the project.

Response of the engineering system without an Energy Suite installation

When you open such a STEP 7 (TIA Portal) project on an engineering system without an Energy Suite installation, a dialog appears informing you that no installation of the Energy Suite is available. After confirming the dialog, you can change the project (excluding the Energy Suite objects), compile it, and download it to the target systems (S7-1500 CPU, PC station).

Possible application examples

The following figure and the following table show the applications that are possible for the project execution without installing Energy Suite:

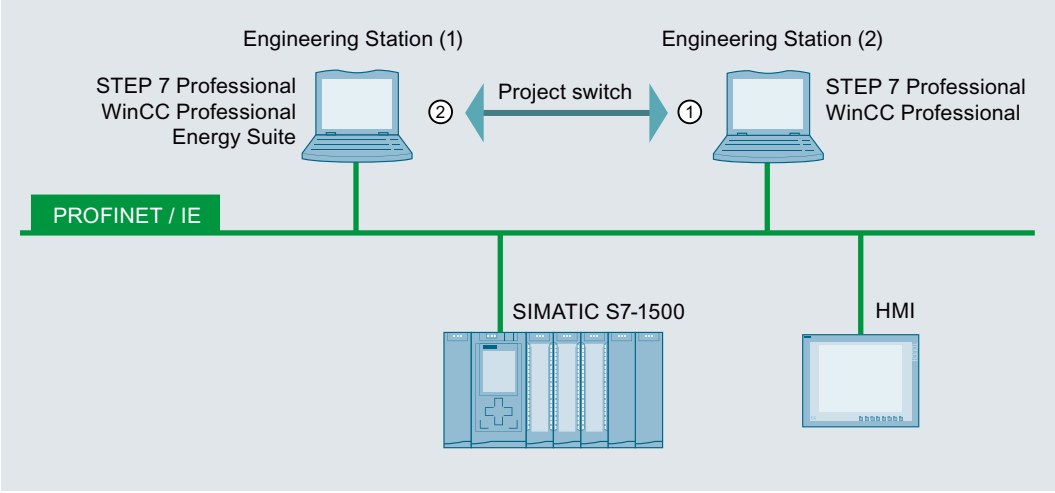

Application example	Newly created/changed project on engineering station (1) with	...can be further processed on the engineering station (2) with	Supported processing
①	• STEP 7 Professional • WinCC Professional • Energy Suite with at least one energy object	• STEP 7 Professional	Opening and editing a STEP 7 (TIA Portal) project on the engineering station (2) containing an Energy Suite configuration but no Energy Suite installation. The Energy Suite configuration remains unchanged.
		• STEP 7 Professional • WinCC Professional	
Application example	**Project that can be further processed on engineering station (2) with**	**... can be further processed on the engineering station (1) with**	**Supported processing**

②	• STEP 7 Professional with unchanged Energy Suite configuration • STEP 7 Professional • WinCC Professional with unchanged Energy Suite configuration	• STEP 7 Professional • WinCC Professional • Energy Suite	Opening and editing a STEP 7 (TIA Portal) project on an engineering station (1) containing an Energy Suite installation after it was previously changed on an engineering station (2) without an Energy Suite installation. Note: If a PLC (S7-1500 CPU) with configured energy objects has been entirely deleted beforehand, the Energy Suite objects for this S7-1500 CPU are no longer available in the project.

5.9 PLC program without measurement hardware library with driver blocks

STEP 7 (TIA Portal) projects that have configured energy objects and parts of an energy program for measurement hardware can also be opened and edited on STEP 7 (TIA Portal) computers on which there is no installation of the Energy Support Library .

This allows you to further edit the device configuration or the rest of the PLC program, for example. The existing configuration of energy objects and the generated energy program for measurement hardware remain unchanged in the project.

Response of the engineering system without Energy Support Library

When configuring the energy objects and assigning the energy data source, the measurement hardware for an Energy Support Library that no longer exists (for example, because the Energy Support Library was uninstalled) is no longer displayed in the list of possible energy data sources.

Measurement hardware already assigned as an energy data source for an Energy Support Library that is no longer available prevents regeneration of the energy program. The missing reference is displayed in the selection field of energy data source. Regeneration of the energy program is canceled as long as the missing reference is not corrected (for example, by changing the energy data source or by installing the Energy Support Library).

The remainder of the PLC program that is unaffected by the energy program can still be modified, compiled and downloaded to the target systems (S7-1500 CPU, PC station).

Visualizing and evaluating energy data 6

6.1 Procedural overview of archiving and visualization with WinCC Professional

Requirement

To perform the following configuration steps, the following requirements must be met in STEP 7 (TIA Portal) in the project tree below the S7-1500 CPU:

- For each CPU that provides energy data, the "WinCC Professional (data log)" archiving type is enabled.
  You can find additional information on this in the section "Setting the archiving on a PC station with WinCC Professional (Data Log) (Page 57)".
- The energy program has been fully generated without error.
- The PLC program has been successfully compiled and downloaded to the S7-1500 CPU.

Configuration steps

The following table provides an overview of the most important configuration steps for visualization and archiving of the periodic energy data with the WinCC Professional SCADA system.

These steps are performed in STEP 7 (TIA Portal) in the project tree below the PC station.

Topic	Step		Description
Visualizing and evaluating energy data	1.1	Configuring a PC station with WinCC Professional	A PC station with the SIMATIC "WinCC RT Professional" HMI application is required for the SCADA system.
	1.2	Creating a connection to the SCADA system	To enable data transfer between the S7-1500 CPU and the SCADA system, configure an HMI connection to the PC station with WinCC Professional in STEP 7 (TIA Portal).
	1.3	Reading and displaying energy values configured for archiving on the engineering system on the PC station (Page 94)	On the PC station with WinCC Professional, Energy Suite offers an overview of all energy objects and the periodic energy values that you have configured in the engineering system for archiving in the data log of WinCC Professional. With the "Refresh" function, synchronization is performed with the engineering data of the S7-1500 CPUs that can be reached by the PC station with WinCC Professional.
	1.4	Configuring the SCADA system (Page 97)	Configuring the PC station with WinCC Professional: • For data transfer to the SCADA system, generate the HMI tags for the data blocks for buffering ("HMI tags" folder). • For archiving, generate the data log with the logging tags (energy values) and set up the archive segments for archiving (Page 100) of the periodic energy values ("Archive" folder). • For visualization of the collected energy data in runtime, you can integrate the template screens and screen elements from the Siemens Industry Online Support into the WinCC application ("Screens" folder). • For the visualization of the collected energy data in runtime, you can also use the automatically installed faceplates from the Energy Suite library.
	1.5	Compiling and downloading the PC station with WinCC Professional	Next you compile the configuration for the entire PC station and download it to the SCADA system.
	1.6	Starting the PLC program and SCADA system (Runtime)	Once WinCC RT Professional is started on the SCADA system and the PLC programs are started on the S7-1500 CPUs, archiving is ready to begin. The selected periodic energy values are archived automatically. The archiving is performed continuously while Runtime is active.

6.2 Energy values on PC station with WinCC Professional

The periodic energy values prepared on the S7-1500 CPU can be transferred to the data log of WinCC Professional for long-term archiving. For example, this allows you to:

- Generate reports from energy values
- Visualize the energy values within WinCC Professional environment
- Transfer energy values to SIMATIC Energy Manager PRO for further processing on the management level

On the PC station with WinCC Professional, the Energy Suite offers an overview of all energy objects and the periodic energy values that you have enabled for archiving in the data log of WinCC Professional.

Once the SIMATIC Energy Suite has been successfully installed, there is this new "Energy data" folder in the project tree below the PC station with WinCC Professional.

The following figure shows the "Energy data" folder in the project tree.

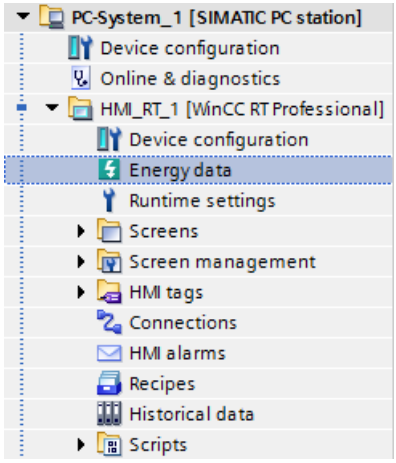

See also

Displaying energy values for archiving in WinCC Professional (Page 94)

6.3 Displaying energy values for archiving in WinCC Professional

You would like an overview of all energy objects and their periodic energy values that you have configured for archiving in the data log of WinCC Professional.

Requirement

- For each CPU that provides energy data, the "WinCC Professional (data log)" archiving type is enabled.
  You can find additional information on this in the section "Setting the archiving on a PC station with WinCC Professional (Data Log) (Page 57)".
- A PC station with the SIMATIC "WinCC RT Professional" HMI application is configured in your project.
- A valid HMI connection between the S7-1500 CPU and the PC station with WinCC Professional exists.
  You can find additional information on this in the online help of the TIA Portal information system under "Configuring HMI connections > Creating a new connection graphically".

Procedure

To display the energy values for the logging and archiving, follow these steps:

1. Open the PC station in the project tree.
2. Below the PC station, open the application "WinCC RT Professional".
   The "Energy data" object is visible below the application.
3. Double-click on the "energy data" object.
   A table with all the energy objects of S7-1500 CPUs of the entire project opens in the working area. This gives you an overview of all available energy objects and the periodic energy values to be archived in data log of WinCC Professional.
4. Click on the "Refresh" icon.
   The data in the table are read in again. A comparison is performed between the engineering data from this PC station and the S7-1500 CPUs that can be reached by WinCC Professional.

Result

For each energy object, there is an enable for the energy program and the configuration of the following energy values for archiving:

- Power consumption
- Energy consumption
- Energy counter value

The meaning of the displays is described in detail in the section "Overview of energy data (Page 96)".

Note

Quantity structure

- A maximum of 2000 energy objects can be archived with WinCC Professional.
- The energy objects of a maximum of 10 S7-1500 CPUs can be archived.

6.4 Overview of energy data

The "energy data" table in the "WinCC RT Professional" application provides an overview of the configured energy objects and the periodic energy values to be archived in data log of WinCC Professional.

Note

Adaptation of periodic energy values

The periodic energy values are only displayed in WinCC Professional. To change the settings of the periodic energy values, you have to switch to the configuration of the energy objects and the energy program settings below the S7-1500 CPUs.

The following table shows the settings for acquisition and archiving of the energy values in the data log of WinCC Professional:

Setting	Description
Name of the energy object	Displays the name of the energy object. The name can only be changed when configuring the energy object below the S7-1500 CPU.
Name of the CPU	Displays the name of the S7-1500 CPU to which the energy object is assigned
Enable energy object	Displays the enable status of the energy object for the generation of the energy program. You activate the enable when configuring the energy object below the S7-1500 CPU.
Power consumption	Indicates whether the periodic energy values of the power consumption are enabled for archiving
Energy consumption	Indicates whether the periodic energy values of the energy consumption are enabled for archiving
Energy counter value	Indicates whether the periodic energy values of the energy counter value are enabled for archiving
Duration of archiving period	Shows the duration of the archiving period. The duration can only be changed in the energy program settings below the S7-1500 CPU.
Name of the archiving period	Displays the name of the archive period. The name can only be changed in the energy program settings below the S7-1500 CPU.

Note

Quantity structure

- A maximum of 2000 energy objects can be archived with WinCC Professional.
- The energy objects of a maximum of 10 S7-1500 CPUs can be archived.

See also

Displaying energy values for archiving in WinCC Professional (Page 94)

Energy values on PC station with WinCC Professional (Page 93)

6.5 Create archiving structure for energy values

After synchronization of the energy data of the engineering system with the energy data of the PC station with WinCC Professional, generate the HMI tags and the data log with the logging tags for the energy values to be archived. This generation takes place automatically by pressing a button; manual adjustment afterwards is not required.

The generated data can then be compiled and downloaded to the SCADA system or is available for visualization in Runtime.

Requirement

- For each CPU that provides energy data, the "WinCC Professional (data log)" archiving type is enabled.
  You can find additional information on this in the section "Setting the archiving on a PC station with WinCC Professional (Data Log) (Page 57)".
- A PC station with the SIMATIC "WinCC RT Professional" HMI application is configured in your project.
- A valid HMI connection exists between the S7-1500 CPU and the PC station with WinCC Professional.
  You can find additional information on this in the online help of the TIA Portal information system under "Configuring HMI connections > Creating a new connection graphically".

Procedure

To generate the archiving structure for the energy values, follow these steps:

1. Open the PC station in the project tree.
2. Below the PC station, open the application "WinCC RT Professional".
   The "Energy data" object is visible below the application.
3. Double-click on the "energy data" object.
   The table with all energy objects and their archived periodic energy values is opened in the working area structured according to the S7-1500 CPUs of the entire project.
4. Click the icon "Create data log".

Result: HMI tags created

A new folder "Energy Suite - HMI tags" is created automatically in the "HMI tags" folder in the project tree below the PC station with all tags required for the data blocks for buffering.

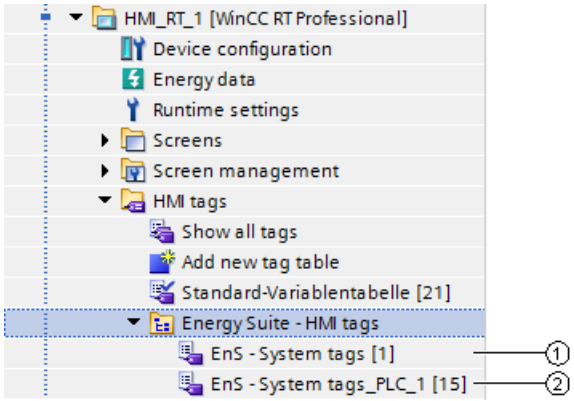

	Object	Description
①	EnS - System tags	Internal tag @EnS_SystemTag for the connection to the data log
②	EnS - System tags_<name of the CPU>	The following HMI tags are created for each data block n (n = 1, 2, 3, ...) for buffering (and therefore for each energy object table): • @EnS_Archive<n>_request_<name of the CPU> with the data type EnS_typeHMIWriteCtrl • @EnS_Archive<n>_statusWrite_<name of the CPU> with the data type EnS_typeHMIWriteStat • @EnS_Buffer<n>_<name of the CPU> with the data type EnS_typeValue

The associated table in the working area shows the created HMI tags and the HMI connection to the S7-1500 CPU. The HMI tags cannot be changed.

Result: Data log with logging tags

A new data log called "EnS_EnergyArchive" with default settings is automatically created in the working area for the "Archive" folder. The settings "Locked", "Manual input permitted" can be checked and changed, if necessary.

An archive tag with the name of the energy object and the extensions _energy, _power or _energyCounter is created in the "Logging tags" area for each energy value of all configured energy objects that is to be archived. The generated logging tags are write-protected and connected to the internal tag "@EnS_SystemTag".

Parameter values are created in the "Comment for Runtime" column; these can be interpreted by the SIMATIC Energy Manager PRO for further processing.

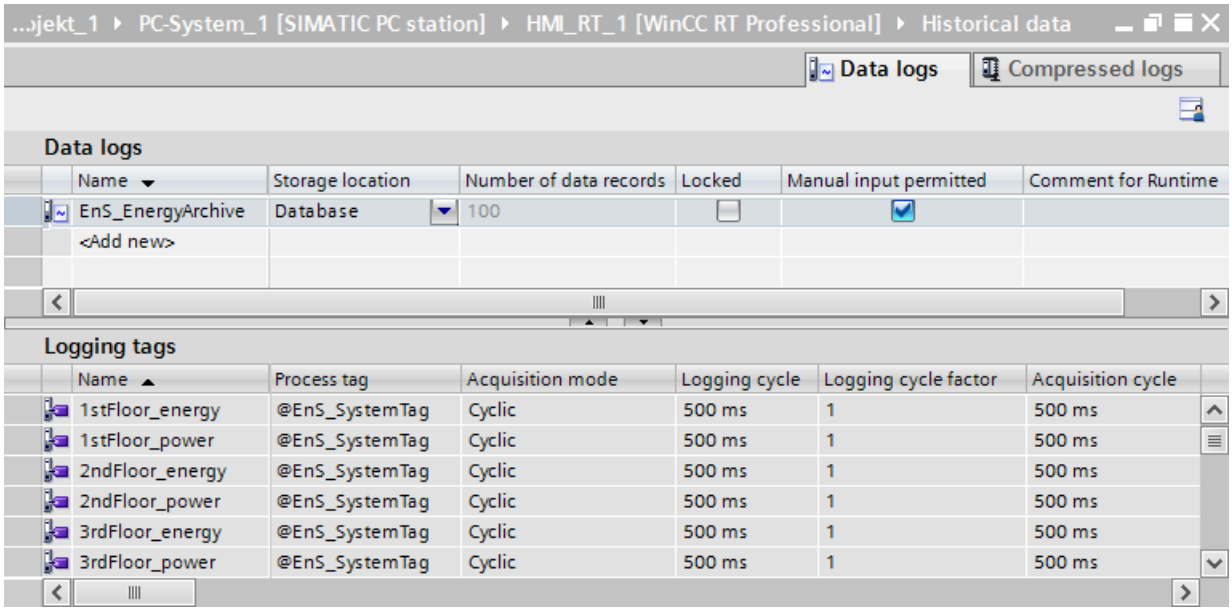

In addition, the Energy Suite component is entered in the startup list of WinCC Runtime under:

Runtime settings > Service > Additional tasks/applications

The generated data can be compiled and downloaded to the SCADA system or is available for visualization in Runtime.

Prior to compiling, check the Settings of the archive segments of the data log (Page 100) in the Runtime settings of the PC station.

6.6 Data log and segmentation for Energy Suite

The automatically created data log on the PC station with WinCC Professional enables long-term archiving of the energy values with time stamp so that the archived historical data can be used for empirical statistical analyses.

In the Runtime settings below the PC station with WinCC Professional, you also have the option of segmenting the data log and thereby defining its size.

Setup of the archive segments for archiving the periodic energy values is made in the working area under:

Runtime settings > Archiving > Archive segments

Recommendation for segmenting the data log

Several archive segments of equal size are created in a segmented circular log and successively filled with periodic energy data (process values). When all segments are completely filled, the oldest segment is overwritten and the energy data it contains is lost.

Only the energy data within the period that contains all segments is accessed for long-term archiving. The time period is set to 1 week by default.

The following Energy Suite recommendations apply to continuous long-term archiving of energy data.

- Time period of all segments: 2 years
- Maximum size of all segments: 24 GB
- Time period contained in a single segment: 1 month
- Maximum size of a segment: 1 GB
- Backup: Activated

Also set the path for the backup on a backup server.

6.7 Archiving energy data

Once both WinCC RT Professional and the PLC program have started, archiving is ready to begin. The periodic energy values are archived automatically. The archiving is performed continuously while Runtime is active.

Requirement

- The project has been compiled and downloaded to the target systems (CPUs and PC station with WinCC Professional).
- There are online connections to the S7-1500 CPUs.
- The program processing on the S7-1500 CPUs is active (CPU in "RUN").
- PC station with WinCC Professional is in Runtime ("Start runtime").

Result

The archived data include:

- Time stamps generated by the CPU
- Parameters of the energy objects, for example:
  - Name of the energy object
  - Energy object detection
  - High/low limit
  - Name of the data log
- Periodic energy values over an archiving period:
  - Energy counter value
  - Energy consumption
  - Power consumption
- Tag attributes (e.g. Quality Code)

Table view

Time column	EO1_Power	EO1_Energy	EO2_Power	EO2_Energy
6/16/2016 1:02:00 PM	111.11100769043 m.	1.8518500328064 m.	222.222015380859 m.	3.70370006561279 m.
6/16/2016 1:03:00 PM	111.11100769043 m.	1.8518500328064 m.	222.222015380859 m.	3.70370006561279 m.
6/16/2016 1:04:00 PM	111.11100769043 m.	1.8518500328064 m.	222.222015380859 m.	3.70370006561279 m.
6/16/2016 1:05:00 PM	111.11100769043 m.	1.8518500328064 m.	222.222015380859 m.	3.70370006561279 m.
6/16/2016 1:06:00 PM	111.11100769043 m.	1.8518500328064 m.	222.222015380859 m.	3.70370006561279 m.

6.8 Visualize energy data with WinCC Online Trend View

You can track the course of the periodic energy values

- Graphically via WinCC Online Trend Control (with connection to the data log)
- Numerically via the Energy Suite Export Tool (Page 103) (included in the installation scope of the Energy Suite)

Visualization with WinCC Online Trend View

The WinCC Online Trend View is used to display periodic energy values archived in the data logs as a trend in Runtime. You can configure the graphical representation of the energy values yourself. To do this, create trend windows and connect the values in the trend to the logging tags of the required energy values.

You can find additional information on this in the WinCC Information System under the keyword "WinCC Online Trend View".

The following figure shows an example for displaying energy values in WinCC Online Trend View:

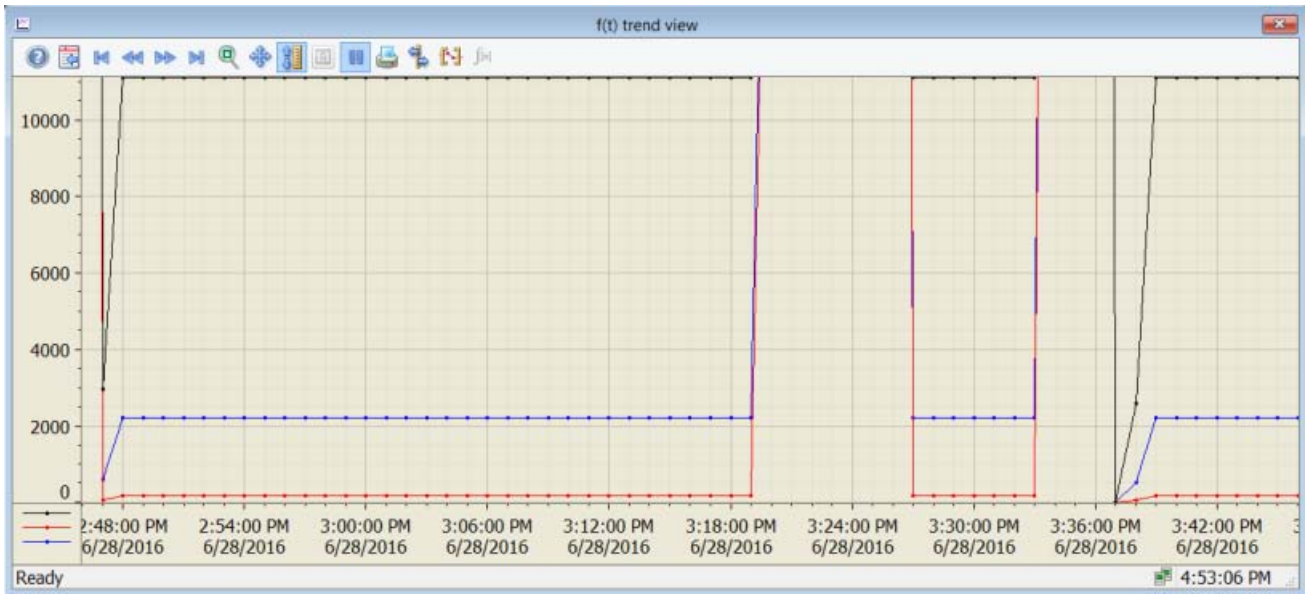

6.9 Evaluate energy data with Energy Suite Export Tool

The Energy Suite Export Tool supports you in evaluating the acquired energy values in Microsoft Excel format.

The Energy Suite Export Tool is part of SIMATIC Energy Suite Runtime and is installed together with it on the PC station with WinCC Professional. You can find additional information on this in the installation instructions for Energy Suite on the DVD.

You can use the Energy Suite Export Tool to export the periodic energy values archived in the data logs to an external file in Runtime. The data exported in this way can then be opened in Microsoft Excel and evaluated.

In addition, you can generate cost center reports with the tariff set you have created with the Energy Suite Export Tool. You assign these cost center reports to your cost centers.

Note

To generate a cost center report, you must have created at least one tariff set and one cost center.

Based on the defined tariff sets and cost centers, the cost center report shows you the energy consumption and the resulting energy costs. You can open and analyze the cost center report in Microsoft Excel.

Possible export format:

- .xlsx
- Compatible with Microsoft Excel Version 2010 and higher

Possible creation of report:

- Manual report
- Automatic report

Start Energy Suite Export Tool and create manual report

1. Select the menu command Start > Siemens Automation > Options and Tools > SIMATIC Energy Suite Runtime Toolbox > SIMATIC Energy Suite Export Tool.
   The Energy Suite Export Tool opens.
2. Check the name of the WinCC server:
   - When WinCC Professional is in Runtime, the server name (name of the local Runtime PC) is entered automatically.
   - Otherwise, you can manually set the server connection using the menu command Settings > WinCC Server.
3. Select the interface language (English, German) with the menu command Settings > Languages.

4. Configure the template for the reports using the Report menu. The following settings are possible, for example:
   - Report type: Export of archived measured values or Cost center report
   - Execution as "Manual"
   - Report name
   - File name (made up of the name of the report, date and time of report creation)
   - Storage path
   - Tariff rate (for Cost center report)
   - Unit (is specified automatically according to the tariff set)
   - Data log and logging tags for evaluation (for Export of archived measured values)
   - Cost center (for Cost center report)
   - Time period that the report evaluates (reporting period)

   Note

   The units for cubic meters m^3 is shown in the Energy Suite Export Tool as "m3".

5. Start the export to Microsoft Excel with the menu command Report > Create or click on the icon.
   This step requires a connection to the WinCC server with WinCC Professional in Runtime.

Result

The following figure shows an example for displaying energy values in the Energy Suite Export Tool:

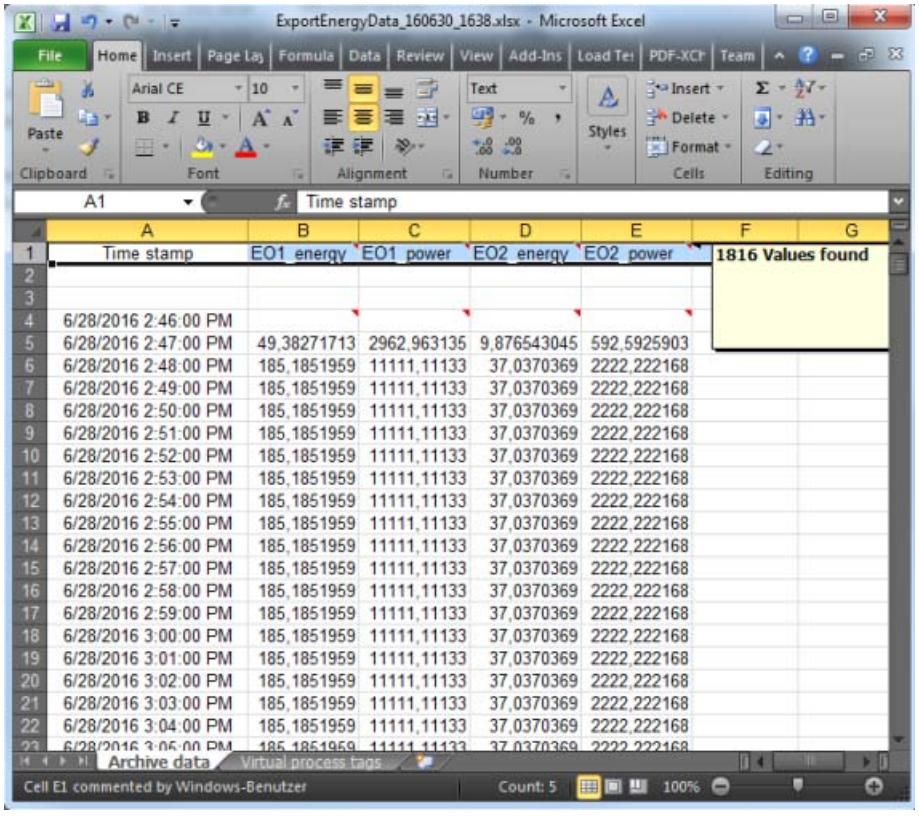

Time stamp	EO1_energy	EO1_power	EO2_energy	EO2_power
6/28/2016 2:46:00 PM				
6/28/2016 2:47:00 PM	49,38271713	2962,963135	9,876543045	592,5925903
6/28/2016 2:48:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:49:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:50:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:51:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:52:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:53:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:54:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:55:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:56:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:57:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:58:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 2:59:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 3:00:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 3:01:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 3:02:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 3:03:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 3:04:00 PM	185,1851959	11111,11133	37,0370369	2222,222168
6/28/2016 3:05:00 PM	185,1851959	11111,11133	37,0370369	2222,222168

Start Energy Suite Export Tool and create automatic report

1. Select the menu command Start > Siemens Automation > Options and Tools > SIMATIC Energy Suite Runtime Toolbox > SIMATIC Energy Suite Export Tool.
   The Energy Suite Export Tool opens.
2. Check the name of the WinCC server:
   - When WinCC Professional is in Runtime, the server name (name of the local Runtime PC) is entered automatically.
   - Otherwise, you can manually set the server connection using the menu command Settings > WinCC Server.
3. Select the user interface language (English, German) with the menu command Settings > Languages.
4. Save the language settings with the menu command File > Save settings.

5. Configure the report template via the Report menu. The following table shows all possible settings. The last table column shows the concrete values for the following examples:
   – Example 1: Automatic archived measured value report is to be created each Monday at 12:15 AM for the elapsed week, for the period of the vacation from 17/07/2017 to 18/08/2017.
   – Example 2: Automatic archived measured value report is to be created continuously on the 2nd of each month at 09:00 AM for the last month.

Parameter	Description	Value range	Value for example
Type (Type)	Type of the report	Archived measured values export Cost center report	• Archived measured values export
Trigger (Execution)	Execution of the report	• Manually • Automatically	• Automatically
Name (Name)	Name of the report	-	For example 1: • WeeklyReport For example 2: • MonthlyReport
File name (File name)	File name of the report, made up of the name of the report, date and time of report creation.	-	For example 1: • WeeklyReport_170717_1215.xls For example 2: • MonthlyReport_170702_0900.xls
Path (Path)	Storage location of the created report	-	• C:\Users\Administrator\Desktop\
Tariff set (Tariff set)	Selection of the tariff set	-	For example 1: - For example 2: -
Unit (Unit)	Unit of energy consumption (automatic from tariff set)	-	For example 1: - For example 2: -
Cost centers (cost centers)	Selection of cost center	-	For example 1: - For example 2: -
Report cycle (Report cycle)	Cycle in which the report is created	• Daily • Weekly • Monthly • Yearly	For example 1: • Weekly For example 2: • Monthly

Parameter	Description	Value range	Value for example
Time of day (Time of day)	Time at which the report is created	• 12-hour format • 24-hour format	For example 1: • 12:15:00 AM For example 2: • 09:00:00 AM
Week day (Weekday)	Weekday on which the report is created. Can only be selected if report cycle = weekly.	Monday - Sunday	For example 1: • Monday For example 2: -
Day (Day)	Time at which the report is created. Can only be selected if report cycle = monthly or annually.	1 - 28	For example 1: - For example 2: • 2
Month (Month)	Month in which the report is created. Can only be selected if report cycle = annually.	January - December	For example 1: - For example 2: -
Time period (Time period)	Period that the report evaluates	For "daily" report cycle: • Start, End • Last day • Elapsed day (00:00 - 24:00) For "daily" report cycle: • Last week • Elapsed week (Monday - Sunday) For "monthly" report cycle: • Last month • Elapsed month (1st - end of month) For "annual" report cycle: • Last year • Elapsed year (January - December)	For example 1: • Elapsed week (Monday - Sunday) For example 2: • Last month
Valid period (Valid period)	Entire period for which the reporting is permitted and valid	• Start: Current date (set by default) • End: 12/31/2099 12:00:00 AM (set by default)	For example 1: • Start: 07/15/2017 12:15:00 AM End: 08/15/2017 12:15:00 AM For example 2: • Values are set by default

Note

The units for cubic meters m^3 is shown in the Energy Suite Export Tool as "m3".

6. Once the report template has been fully configured, click on "Finish".
   A task will automatically be added to the Windows Task Scheduler and executed in the configured report cycle . The reports are stored in the configured storage path.

Creating tariff sets

To create tariff sets, follow these steps:

1. Select the menu command "Settings > Tariff set".
   The "Manage tariff sets" dialog opens.
2. To create a new tariff set, click on the "Add tariff set" icon in the toolbar.
   The "Add tariff set" dialog opens.
3. Configure the template for the tariff sets. The following settings are possible, for example:
   - Template name of the tariff set
   - Unit price per energy consumption
   - Assignment of tariffs to time ranges and weekdays
   - Price per energy consumption
   - Hiding holidays

   The following entries are required:
   - Template name
   - One tariff selection per weekday
   - Tariff must cover complete day (time range) per weekday
4. Click on the "Add" button.
   The tariff set is added to the list in the "Manage tariff sets" dialog.

Reaction to error

Note

If an unexpected response occurs (for example, WinCC Professional database does not start), the error messages are stored in the file "Siemens.Simatic.EnergyMgmt.WinCCprof.DataExportTool.txt" under "C:\Program Files (x86)\Siemens\Automation\SCADA-RT_V11\WinCC\bin\config".

Energy Suite blocks

A

A.1 Overview of Energy Suite blocks

Note

Description of Energy Suite blocks for experts

You can find a detailed description about how each block operates and its block parameters in the following sections.

You do not need this information for the standard application with Energy Suite. SIMATIC Energy Suite automatically generates for you all the required blocks and their interconnection based on the configured energy objects and the energy program settings.

CAUTION

Manual change to blocks of the energy program

If you make manual changes to the blocks and their interconnection after generating the energy program, you yourself are responsible for the error-free operation of the devices and the energy program.

CAUTION

Using Energy Suite blocks in other PLC programs

If you use the Energy Suite blocks from the global library in other PLC programs, you yourself are responsible for error-free use of Energy Suite blocks.

This section provides an overview of all available Energy Suite blocks. Following the installation of SIMATIC Energy Suite, the blocks can be seen under "Global Libraries" in STEP 7 (TIA Portal):

- Function blocks (Page 113)
- Data blocks (Page 160)
- Data types (Page 162) including enumerations (Page 175)

How Energy Suite blocks work

The following shows how the Energy Suite blocks generally work in the energy program for energy data from the process image of the CPU (PLC tags) or from data blocks (DB tags).

The placeholder "xxx" in the picture means that there are different types of this block, for example, for "EnS_Drvxxx":

- EnS_DrvPulse
- EnS_DrvCounterLInt

You can find more types in the "Function Blocks" table after the picture below.

The following picture show how the Energy Suite blocks work:

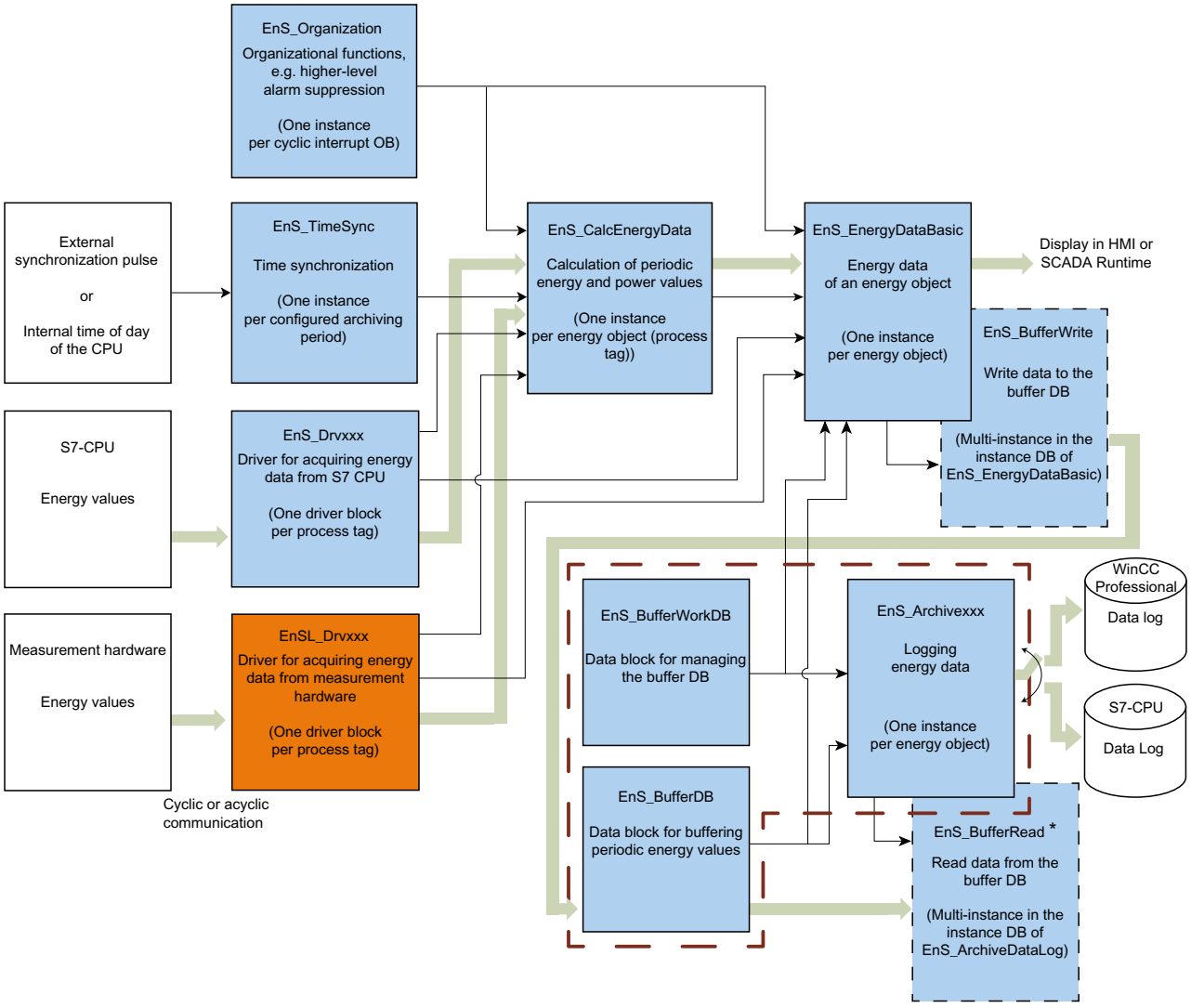

* EnS_BufferRead is not called for archiving in WinCC Professional

Energy data flow

Parameter assignment data

Blocks used internally

Blocks associated with an energy object table

Function blocks

Name	Function
EnS_ArchiveDataLog (Page 113)	Archived energy data in data logs (CSV file on the SIMATIC memory card of the CPU)
EnS_ArchiveHMI (Page 121)	Archived energy data in data logs of WinCC Professional
EnS_BufferRead (Page 128)	Reads the data from the buffer DB (EnS_BufferDB (Page 160)). Used internally
EnS_BufferWrite (Page 130)	Writes data to the buffer DB (EnS_BufferDB (Page 160)). Used internally
EnS_CalcEnergyData (Page 133)	Periodic calculation of the consumed energy and average power consumption
EnS_DrvAnalogValue (Page 139)	Driver block for analog input (e.g. power value, flow value)
EnS_DrvCounterLInt (Page 141)	Driver block for counter values of the LINT type
EnS_DrvCounterLReal (Page 143)	Driver block for counter values of the LREAL type
EnS_DrvPulse (Page 145)	Driver block for a pulse input
EnS_EnergyDataAdv (Page 147)	Advanced energy data from a device (for example, phase-specific current and voltage values)
EnS_EnergyDataAdvMinMax (Page 148)	Minimum and maximum values of the advanced energy data of a device
EnS_EnergyDataBasic (Page 149)	Basic energy data from a device (for example, power, energy counter value)
EnS_Organization (Page 153)	Contains general data and parameters for central program organization
EnS_TimeSync (Page 154)	Synchronizes the times for calculating the energy data

Data blocks

Name	Function
EnS_BufferDB (Page 160) *	Data block for buffering of energy data records
EnS_BufferWorkDB (Page 161) *	Data block for monitoring and status information of the buffer

* The name can be changed.

User-defined data types

Name	Function	Used by
EnS_typeAnalogValue (Page 162)	Data type for analog process values	EnS_CalcEnergyData EnS_DrvAnalogValue
EnS_typeBufferReadCtrl (Page 162)	Data type for control information of the buffer for reading data	EnS_BufferRead EnS_typeBufferWork
EnS_typeBufferReadStat (Page 163)	Data type for data and status information of the buffer for reading data	EnS_typeBufferWork
EnS_typeBufferWork (Page 163)	Data type for monitoring and status information of the buffer	EnS_BufferRead EnS_BufferWrite EnS_ArchiveDataLog EnS_EnergyDataBasic EnS_BufferWorkDB EnS_ArchiveHMI

Name	Function	Used by
EnS_typeBufferWriteCtrl (Page 164)	Data type for control information of the buffer for writing data	EnS_BufferWrite EnS_typeBufferWork
EnS_typeBufferWriteStat (Page 165)	Data type for status information of the buffer for writing data	EnS_typeBufferWork
EnS_typeEnergyAdv (Page 165)	Data type for advanced energy data	EnS_EnergyDataAdv
EnS_typeEnergyBasic (Page 167)	Data type for basic energy data	EnS_CalcEnergyData EnS_EnergyDataBasic
EnS_typeEnergyCounter (Page 167)	Data type for energy count value	EnS_CalcEnergyData EnS_DrvCounterLInt EnS_DrvCounterLReal EnS_DrvPulse
EnS_typeEnergyMeta (Page 168)	Data type for metadata of the measuring point (information for configuration of the energy object)	EnS_CalcEnergyData EnS_BufferWrite EnS_DrvAnalogValue EnS_DrvCounterLInt EnS_DrvCounterLReal EnS_DrvPulse EnS_EnergyDataAdv EnS_EnergyDataBasic
EnS_typeHMIWriteCtrl (Page 171)	Data type for control information of the buffer for writing energy data to the data log of WinCC Professional	EnS_ArchiveHMI
EnS_typeHMIWriteStat (Page 172)	Data type for status information of the buffer for writing energy data to the data log of WinCC Professional	EnS_ArchiveHMI
EnS_typeOrgData (Page 172)	Data type for organization data	EnS_Organization EnS_TimeSync EnS_DrvAnalogValue EnS_DrvCounterLInt EnS_DrvCounterLReal EnS_DrvPulse EnS_ArchiveDataLog EnS_EnergyDataAdv EnS_EnergyDataBasic EnS_ArchiveHMI
EnS_typeSync (Page 173)	Data type for time synchronization	EnS_TimeSync EnS_CalcEnergyData
EnS_typeValue (Page 174)	Data type for output values	EnS_BufferDB EnS_BufferRead EnS_ArchiveDataLog

A.2 Function blocks (FBs)

A.2.1 EnS_ArchiveDataLog: Write energy values to data logs

A.2.1.1 Description of EnS_ArchiveDataLog

Description

The "EnS_ArchiveDataLog" function block writes periodic energy data from the data block for buffering to buffer EnS_BufferDB (Page 160)" ("bufferDB" parameter) in "Data Logs to the SIMATIC memory card of the CPU. Writing is performed only if the "archiveEnergy", "archivePower", "archiveEnergyCounter" parameters have been set to "TRUE" in the corresponding metadata (Page 168). The data are stored as a CSV file on the SIMATIC memory card.

To avoid access conflicts on the SIMATIC memory card of the CPU, use only one instance of the function block per CPU.

The following table shows the structure of a CSV file.

Column	Name	Description
1	SeqNo	Sequential number (specified by the system)
2	Tagname	Tag name
3	Value	Archived value
4	TimeStamp	Time stamp of the archived value
5	Status	Status of the archived value

The name of the archived data is composed of <DataLogName>+<YYYYMMDD_hhmm (time stamp of the creation time stamp in UTC format)>

Example: DataLog_20141117_2315

The size of the data log is limited by the system to 1 GB. The number of data records must be set accordingly at the "noRecords" parameter (see the following calculation example).

Example: The space required on the memory card for a data log is calculated using the following formula: 45 bytes + (number of data records * 190 bytes).

The number of data logs is defined by the "noDataLogs" parameter. The capacity of the memory card limits the number of available data logs (valid number: 1 - 100).

Writing data logs

The archiving of energy data is performed when the buffer DB contains new data (bufferWork.noItems > 0). The function block then reads the new data with the "EnS_BufferRead (Page 128)" function block and writes it to the data log.

Note

The writing speed may vary depending on the CPU used, the employed memory card and the location of the data on the memory card. The number of cycles that are required to write an energy data record varies. No more than 1 data record can be written per cycle.

Number of data logs

If a data log is full, the "full" output is set for one cycle and a alarm is generated.

When the maximum number of data logs ("noDataLogs", valid number: 2-10) has been reached and the last data log is full, the oldest data log is deleted and a new data log is created. A message is generated when a data log is deleted or a new one is created. You can view the alarm on a SCADA system, for example.

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 116)" parameter contains additional error information. The error information of the lower-level function block "EnS_BufferRead (Page 128)" is added to the "status" parameter. The error information is displayed in the "status" parameter until the error is cleared. Only then is the information about the next pending error displayed.

A.2.1.2 Parameters of EnS_ArchiveDataLog

The following table shows the function block "EnS_ArchiveDataLog":

Parameter	Declaration	Data type	Default	Description	R *
dataLogName	Input	STRING[32]	-	Name of the data log	-
noRecords	Input	UDINT	10000	Number of data records in the data log	-
noDataLogs	Input	UINT	5	Number of data logs	-
bufferDB	Input	DB_ANY	-	Pointer to buffer data block	-
sD	Input	VARIANT	-	Associated value for alarm messages	-
messageSup-press	Input	BOOL	FALSE	TRUE = Alarm suppression activated	-
full	Output	BOOL	FALSE	TRUE = Data log is full	-
busy	Output	BOOL	FALSE	TRUE = Processing is active	-
done	Output	BOOL	FALSE	TRUE = Data successfully processed	-

Parameter	Declaration	Data type	Default	Description	R *
error	Output	BOOL	FALSE	TRUE = Error	-
status	Output	WORD	-	Error status information (Page 116)	-
statusCreateFct	Output	WORD	-	Error status information of the DataLog-Create instruction	-
statusOpenFct	Output	WORD	-	Error status information of the DataLo-gOpen instruction	-
statusWriteFct	Output	WORD	-	Error status information of the DataLog-Write instruction	-
statusDeleteFct	Output	WORD	-	Error status information of the DataLog-Delete instruction	-
statusCloseFct	Output	WORD	-	Error status information of the DataLog-Close instruction	-
dataLogNa-meAct	Output	STRING[48]	-	Name of the active data log	-
dataLogName-Last	Output	STRING[48]	-	Name of the last data log	-
dataLogName-Del	Output	STRING[48]	-	Name of the deleted data log	-
errorAlarm1	Output	BOOL	FALSE	TRUE = Error Program_Alarm1	-
statusAlarm1	Output	WORD	-	Error status information Pro-gram_Alarm1	-
alarmState1	Output	BYTE	-	Alarm status of Program_Alarm1	-
errorAlarm2	Output	BOOL	FALSE	TRUE = Error Program_Alarm2	-
statusAlarm2	Output	WORD	-	Error status information Pro-gram_Alarm2	-
alarmState2	Output	BYTE	-	Alarm status of Program_Alarm2	-
errorAlarm3	Output	BOOL	FALSE	TRUE = Error Program_Alarm3	-
statusAlarm3	Output	WORD	-	Error status information Pro-gram_Alarm3	-
alarmState3	Output	BYTE	-	Alarm status of Program_Alarm3	-
errorAlarm4	Output	BOOL	FALSE	TRUE = Error Program_Alarm4	-
statusAlarm4	Output	WORD	-	Error status information Pro-gram_Alarm4	-
alarmState4	Output	BYTE	-	Alarm status of Program_Alarm4	-
errorAlarm5	Output	BOOL	FALSE	TRUE = Error Program_Alarm5	-
statusAlarm5	Output	WORD	-	Error status information Pro-gram_Alarm5	-
alarmState5	Output	BYTE	-	Alarm status of Program_Alarm5	-
errorAlarm6	Output	BOOL	FALSE	TRUE = Error Program_Alarm6	-
statusAlarm6	Output	WORD	-	Error status information Pro-gram_Alarm6	-
alarmState6	Output	BYTE	-	Alarm status of Program_Alarm6	-
id	InOut	DWORD	0	ID of the current data log	X

Parameter	Declaration	Data type	Default	Description	R *
orgData	InOut	EnS_typeOrgData (Page 172)	-	Organization data	-
bufferWork	InOut	EnS_typeBuffer-Work (Page 163)	-	Buffer monitoring and status information	-

* The marked parameters are given the "Retain" system attribute and are stored in the retentive memory area of the CPU.

A.2.1.3 Parameter status

The following table shows the error codes that are generated at the "status" output parameter of the "EnS_ArchiveDataLog" function block when errors occur:

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8511	Error reading from buffer	Archiving error	buffer-Work.read-Stat.status-ReadFct	The reading continues until it is successful. Cause of error: Wrong data block is connected or no data block is present. Procedure: Correct the program.
8601	Error creating data log	Error creating	statusCreateFct	The errors of the "DataLogCreate" SFC are copied to the "statusCreateFct" parameter. The creation is repeated until it is successful. Causes of error: • Name of the data log is invalid • No memory space Procedure: • Assign a valid name to the Data Log. • You can find additional information about valid names in the section "Archiving and buffering (S7-CPU) (Page 54)". • Adapt the storage capacity of the SIMATIC Memory Card used. • Release the lock on the used SIMATIC Memory Card.
8602	Error opening data log	Error opening	statusOpenFct	Opening the data log is repeated until it is successful. Causes of error: • Data Log is not available • Data Log is already open Procedure: • Close the data log before opening. • Ensure that the Data Log has already been created. • Release the lock on the used SMATIC Memory Card.

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8603	Error writing data log	Error writing	statusWriteFct	Writing to the data log is repeated until it is successful. **Causes of error:** • Data Log is not available • No memory space **Procedure:** • Ensure that the Data Log has already been created. • Release the lock on the SIMATIC Memory Card used.
8604	Error deleting data log	Error deleting	statusDeleteFct	The Data Log file is not deleted **Causes of error**: • Data Log is not available • Data Log is write protected • No memory space **Procedure**: • Close the data log file. • Adapt the storage capacity of the SIMATIC Memory Card used. • Release the lock on the used SIMATIC Memory Card.
8605	Error closing data log	Error during closing	statusCloseFct	Closing the data log is repeated until it is successful. **Causes of error**: • Data Log is not available • Data Log is already closed **Procedure:** • Open the Data Log before closing. • Ensure that the Data Log has already been created. • Release the lock on the SIMATIC Memory Card used.

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8611	Maximum number of data records too high	Parameter assignment error	noRecords	The size of the data log is limited by the system to 1 GB. The capacity of the SIMATIC Memory Card limits the number of available data logs. Causes of error: • The maximum number of data records is exceeded. Procedure: • Adapt the storage capacity of the SIMATIC memory card used (for example, by deleting the old data logs).
8612	Number of data logs invalid (<2 or >10)	Parameter assignment error	noDataLogs	The number of data logs corresponds to the "Number of data logs" setting in the "Archiving" area for the "PLC (SIMATIC Memory Card)" archiving type. Causes of error: • The number of data logs is out of range. Procedure: • Specify a valid number of data logs under "CPU > Energy Objects > Energy Program Settings > Archiving and buffering > Number of data logs".

The additional alarm texts ("Description" column) are written in the "EnS_Alarms" text list of the "Program_Alarm2" alarm block and are output with the error codes in the alarm.

See also

Description of EnS_BufferRead (Page 128)

A.2.1.4 Alarms of EnS_ArchiveDataLog

The alarms are generated with the instruction "Program_Alarm" A multi-instance of the "Program_Alarm" data type is created under the "Static" tag of the "EnS_ArchiveDataLog" function block for each alarm. You can find additional information on "Program_Alarm" in the online help of the TIA Portal information system under "Extended instructions > Alarms".

The following table shows the alarms generated by the "EnS_ArchiveDataLog" function block:

Alarm block	Alarm text	Alarm class
Program_Alarm1	Parameter assignment error	With acknowledgment
Program_Alarm2	Logging error: <text list: EnS_Alarms:#status>	With acknowledgment
Program_Alarm3	Energy log full: <Tag:#dataLogNameLast >	With acknowledgment
Program_Alarm4	Create energy Log: <Tag:#dataLogNameAct >	With acknowledgment

Alarm block	Alarm text	Alarm class
Program_Alarm5	Delete energy log: <Tag:#datalogNameDel >	With acknowledgment
Program_Alarm6	Data loss of <Tag:#bufferWork.noDeletedItems" data records*	With acknowledgment

* The "bufferWork.noDeletedItems" parameter contains the number of records that will be overwritten with a buffer overflow.

The additional alarm texts for "Program_Alarm2" are described in the "EnS_Alarms" text list. The error codes (status) are output in the alarm text and contain additional alarm text of the respective error codes.

The following table shows the error codes with additional alarm texts of the "Program_Alarm2" alarm blocks:

Alarm block	Error code (W#16#...)	Alarm text	Parameter	Function block
Program_Alarm2	8511	Error reading from buffer	status	EnS_BufferRead (Page 128)
	8601	Error creating data log		EnS_ArchiveDataLog
	8602	Error opening data log		
	8603	Error writing data log		
	8604	Error deleting data log		
	8605	Error closing data log		

The associated values for the alarms are assigned as follows:

Alarm block	Associated value	Parameter	Description
Program_Alarm1	1	sD	Free
	2	-	Not used
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used

Alarm block	Associated value	Parameter	Description
Program_Alarm2	1	status	Error status information
	2	sD	Free
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used
Program_Alarm3	1	dataLogNameLast	Name of the full data log
	2	sD	Free
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used
Program_Alarm4	1	dataLogNameAct	Name of the generated data log
	2	sD	Free
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used
Program_Alarm5	1	dataLogNameDel	Name of the deleted data log
	2	sD	Free
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used

Alarm block	Associated value	Parameter	Description
Program_Alarm6	1	bufferWork.noDeletedI-tems	Number of deleted data records on buffer overflow
	2	sD	Free
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used

The alarms can be changed.

The "SD" input parameter is used as an associated value. You can connect the parameters with additional information that is displayed in the alarm.

The alarms can be suppressed with "messageSuppress" or overridden with "orgData.messageSuppress" of the function block "EnS_Organization (Page 153)".

A.2.2 EnS_ArchiveHMI: Write energy values to data logs

A.2.2.1 Description of EnS_ArchiveHMI

Description

The "EnS_ArchiveHMI" function block (FB) periodic writes energy data from the data block (DB) for buffering "EnS_BufferDB (Page 160)" to the data log of WinCC Professional. To do this, the "EnS_ArchiveHMI" FB cyclically requests the DB for buffering "EnS_BufferDB" for available data records with energy data. For existing data records, the "EnS_ArchiveHMI" function block (FB) sends a request to the Energy Suite WinCC Professional component to retrieve the energy data from the DB for buffering and write to the data log.

Writing the data logs

The archiving of energy data to data logs is only performed when the following requirements are met:

- The DB for buffering contains new data records (bufferWork.noItems > 0).
- The DB is available for buffering (bufferWork.busy = FALSE).
- The HMI connection to WinCC Professional has been established.

If the requirements are met, the reading of data from the DB is activated for buffering. The "EnS_ArchiveHMI" FB sends a request to transfer data to WinCC Professional. The request

at the "request" parameter with the "EnS_typeHMIWriteCtrl (Page 171)" data type contains read and write pointers, and the number of data records to be archived.

Then the Energy Suite WinCC Professional component reads the data from the DB for buffering and writes it into the data log. The "statusWrite" parameter of the "EnS_typeHMIWriteStat (Page 172)" data type contains the number of written data records and the updated read pointer. During the writing process, the WinCC Professional component, Energy Suite, updates the number of written data records and the write pointer after every written data record.

Note

Individual instances of "EnS_BufferWrite (Page 130)", "EnS_BufferRead (Page 128)" or WinCC script can access the data block for buffering. Only read access to the data block is possible at the same time.
The buffer control and status information of the global DB "EnS_BufferWorkDB (Page 161)" are provided at the "bufferWork" parameter.

Watchdog

The connection between the CPU and WinCC Professional is monitored using the watchdog counter. The "EnS_ArchiveHMI" FB sets the watchdog counter at the "request.watchdog" parameter. The content of the "request.watchdog" parameter is copied to the "statusWrite.watchdog" parameter.

- If the WinCC Professional component, Energy Suite, acknowledges the time monitoring, the "EnS_ArchiveHMI" FB increments the watchdog counter.
- If no acknowledgment is made within the monitoring time, the error code "16#8606" is output at the "status" parameter. In addition, an active request is reset and no further request is sent to the watchdog until the error is eliminated.
  An alarm with information about the data loss is generated once the error is eliminated.

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 125)" parameter contains additional error information.

If the WinCC Professional component, Energy Suite, does not respond to the request of the "EnS_ArchiveHMI" FB during the monitoring period and no watchdog error occurs, the error code "16#8605" is output at the "status (Page 125)" parameter. Once the waiting period has expired and the request has been successfully fulfilled by the Energy Suite, WinCC Professional, the request is sent again.

A.2.2.2 WinCC Professional component

Description

The Energy Suite WinCC Professional component writes energy data to the data log of WinCC Professional.

Alarms

If errors occur, the WinCC Professional component generates system events in WinCC Professional.

The following table shows the alarms:

Alarm text	Alarm class
Archiving error <text list: EnS_Alarms:#status>	System
Archiving continues	System
Possible data loss	System

Reaction to error

If an error occurs, the error code is output with the additional alarm text in the system event by WinCC Professional. The additional alarm text is displayed in the following table under "Description".

Furthermore, the error codes with additional alarm texts at the "statusWrite.status"/"statusHMI" parameters of the instance DB of the "EnS_ArchiveHMI" function block are output to the CPU.

The following table shows the error codes of the WinCC Professional component:

Error code (W#16#...)	Description	Error type	Copied to parameter
8621	Connection interrupted	Time monitoring error	statusWrite.status
8622	Access to the data log denied	Error writing	statusWrite.status
8623	Writing of the energy data to the data log canceled	Error writing	statusWrite.status
8624	Possible data loss	Data error	statusWrite.status
8625	Access to the data block for buffering interrupted	Data error	statusWrite.status
8631	Tag not configured	Parameter assignment error	statusWrite.status

A.2.2.3 Parameters of EnS_ArchiveHMI

The following table shows the function block "EnS_ArchiveHMI":

Parameter	Declaration	Data type	Default	Description	HMI *
statusWrite	Input	EnS_typeHMIWrite-Stat (Page 172)	-	Status information for writing the energy values to the data log	-
monitoringTime	Input	TIME	T#5s	Monitoring time for writing the energy values to the data log	-
monitoringTime-Watchdog	Input	TIME	T#10s	Monitoring time for watchdog	-
waitTime	Input	TIME	T#1m	Waiting time for repeating the request to write the energy data to the data log	-
bufferDB	Input	DB_ANY	-	Pointer to data block for buffering	-
sD	Input	VARIANT	-	Associated value for alarms	-
messageSup-press	Input	BOOL	FALSE	TRUE = Alarm suppression activated	-
request	Output	EnS_typeHMIWri-teCtrl (Page 171)	-	Request for writing the energy data to data log	X
busy	Output	BOOL	FALSE	TRUE = Writing the energy data to data log is enabled	-
done	Output	BOOL	FALSE	TRUE = Writing the energy data to data log was successful	-
error	Output	BOOL	FALSE	TRUE = Error	-
status	Output	WORD	-	Error status information (Page 125)	-
statusHMI	Output	WORD	-	Error status information	-
errorAlarm1	Output	BOOL	FALSE	TRUE = Error "Program_Alarm1"	-
statusAlarm1	Output	WORD	-	Error status information of "Pro-gram_Alarm1"	-
alarmState1	Output	BYTE	-	Alarm status of "Program_Alarm1"	-
errorAlarm2	Output	BOOL	FALSE	TRUE = Error "Program_Alarm2"	-
statusAlarm2	Output	WORD	-	Error status information of "Pro-gram_Alarm2"	-
alarmState2	Output	BYTE	-	Alarm status of "Program_Alarm2"	-
errorAlarm3	Output	BOOL	FALSE	TRUE = Error "Program_Alarm3"	-
statusAlarm3	Output	WORD	-	Error status information of "Pro-gram_Alarm3"	-
alarmState3	Output	BYTE	-	Alarm status of "Program_Alarm3"	-
orgData	Output	EnS_typeOrgData (Page 172)	-	Organizational data	-
bufferWork	Output	EnS_typeBuffer-Work (Page 163)	-	buffer control and status information	-

A.2.2.4 Parameter status

The following table shows the error codes that are generated at the "status" output parameter when errors occur:

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8603	Error writing energy data to the data log	Error writing	statusHMI	Internal error **Causes of error:** • HMI connection to the S7-CPU has closed • Data Log is not available • Name of the data log is invalid **Procedure:** • No action needs to be taken when an internal error occurs.
8606	Timeout writing the energy data to the data log	Error writing	status-Write.busy / status-Write.done / statusWrite-error	The period within which the energy data from the S7 CPU are transferred to the data log has expired. **Causes of error:** • Energy Suite SCADA component does not respond to the request to write the energy data within the monitoring time. • HMI connection to the S7-CPU has closed • Data loss
8607	Watchdog error	Error writing	statusWrite.error	Internal error **Causes of error**: The Energy Suite SCADA component was not acknowledged within the monitoring time. **Procedure:** • No action needs to be taken when an internal error occurs.
8611	Monitoring time < 0 or > 1 hour	Parameter assignment error	monitoringTime	Internal error **Causes of error:** • Monitoring time is invalid • Watchdog error has not occurred **Procedure:** • No action needs to be taken when an internal error occurs.

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8612	Monitoring time for watchdog < 0 or > 1 hour	Parameter assignment error	monitoringTime-Watchdog	Internal error Causes of error: • The monitoring time for Watchdog is invalid Procedure: • No action needs to be taken when an internal error occurs.
8613	Wait time < 0 or > 1 hour	Parameter assignment error	waitTime	Internal error Causes of error: • Waiting time for repeating the request to write the energy data to the data log is invalid Procedure: • No action needs to be taken when an internal error occurs.

The additional alarm texts ("Description" column) are written in the "EnS_Alarms" text list of the "Program_Alarm2" alarm block and are output with the error codes in the alarm.

A.2.2.5 Alarms of EnS_ArchiveHMI

The alarms are generated with the instruction "Program_Alarm" A multi-instance of the "Program_Alarm" data type is created under the "Static" tag of the "EnS_ArchiveHMI" function block for each alarm. You can find additional information on "Program_Alarm" in the online help of the TIA Portal information system under "Extended instructions > Alarms".

The following table shows the alarms of the "EnS_ArchiveHMI" function block:

Alarm block	Alarm text	Alarm class
Program_Alarm1	Parameter assignment error	With acknowledgment
Program_Alarm2	Archiving error <text list: EnS_Alarms:#status>	With acknowledgment
Program_Alarm3	Data loss of <Tag:#bufferWork.noDeletedItems" data records *	With acknowledgment

* The "bufferWork.noDeletedItems" parameter contains the number of records that have been overwritten during the buffer overflow. The data block for buffering is emptied (bufferWork.noItems = 0) for each buffer overflow.

The additional alarm texts of the "Program_Alarm2" alarm block are written in the "EnS_Alarms" text list. The error codes (status) are output in the alarm text and contain additional alarm text of the respective error codes.

The following table shows the error codes with additional alarm texts of the "Program_Alarm2" alarm blocks:

Alarm block	Error code (W#16#...)*	Alarm text	Parameter	Function block
Program_Alarm2	8606	Timeout writing the energy data to the data log	status	EnS_ArchiveHMI
	8607	Connection to the data log lost		EnS_ArchiveHMI

* The error code in the program editor can be displayed as integer or hexadecimal value. You can find additional information about switching the display format in the online help of the TIA Portal information system under "Changing display formats in the program status".

The following table shows the alarm blocks with assigned associated values:

Alarm block	Associated value	Parameter	Description
Program_Alarm1	1	sD	Free
	2	-	Not used
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used
Program_Alarm2	1	status	Error status information
	2	sD	Free
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used
Program_Alarm3	1	bufferWork.noDeletedI-tems	Number of data records overwritten by a buffer overflow
	2	sD	Not used
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used

Alarm block	Associated value	Parameter	Description
	9	-	Not used
	10	-	Not used

The alarms can be changed.

The "sD" input parameter as an associated value. You can connect the parameters with additional information that is displayed in the alarm.

The alarms can be suppressed with "messageSuppress" or overridden with "orgData.messageSuppress" of the function block "EnS_Organization (Page 153)".

A.2.3 EnS_BufferRead: Reading energy values from data blocks

A.2.3.1 Description of EnS_BufferRead

Description

The "EnS_BufferRead" function block reads data from the "EnS_BufferDB (Page 160)" data block.

The "EnS_BufferRead" function block is called internally by the "EnS_ArchiveDataLog (Page 113)" function block.

The following picture show the circuitry of the parameter:

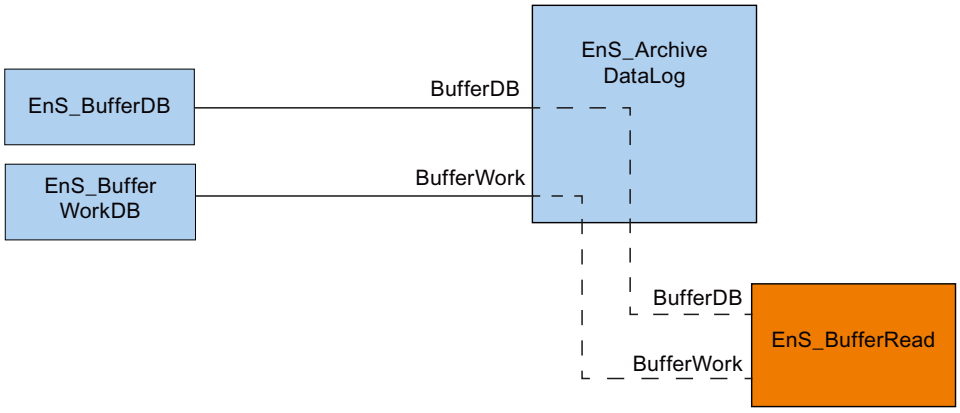

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 129)" parameter contains additional error information.

Alarms

The block has no signaling characteristics. Alarms are created by calling function block.

See also

EnS_BufferWorkDB (Page 161)

A.2.3.2 Parameters of EnS_BufferRead

The following table shows the function block "EnS_BufferRead":

Parameter	Declaration	Data type	Default	Description
bufferDB	Input	DB_ANY	-	Reference to the data block for buffering
data	Output	EnS_typeValue (Page 174)	-	Read data from the buffer
busy	Output	BOOL	FALSE	TRUE = Processing is active
done	Output	BOOL	FALSE	TRUE = Data successfully processed
error	Output	BOOL	FALSE	TRUE = Error
status	Output	WORD	-	Error status information (Page 129)
bufferWork	InOut	EnS_typeBuffer-Work (Page 163)	-	Buffer monitoring and status information
request	InOut	EnS_typeBuffer-ReadCtrl (Page 162)	-	Buffer monitoring information

A.2.3.3 Parameter status

The following table shows the error codes that are generated at the "status" output parameter when errors occur:

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8511	Error reading from buffer	Error reading	buffer-Work.read-Stat.status-ReadFct	The reading continues until it is successful. Causes of error: • Wrong data block is connected or no data block is present. Procedure: • Correct the program.
8512	Data loss *		bufferWork.no-DeletedItems	Causes of error: • Data loss after buffer overflow A buffer overflow occurs when the buffer cannot be cleared, for example, because the connection to WinCC (for archiving in WinCC) is interrupted. Procedure: • Establish the connection to WinCC. • Adjust the size of the data block for buffering.

* The "bufferWork.noDeletedItems" parameter contains data records that are overwritten with a buffer overflow. The number of deleted data records is part of the alarm that is output by the "EnS_ArchiveDataLog (Page 113)" FB.

A.2.4 EnS_BufferWrite: Write energy values to data blocks

A.2.4.1 Description of EnS_BufferWrite

Description

The "EnS_BufferWrite" function block writes data (periodic energy data) to the data block "EnS_BufferDB (Page 160)".

The "EnS_BufferWrite" is called internally by the "EnS_EnergyDataBasic (Page 149)" block.

"EnS_BufferWrite" is linked to "EnS_BufferWorkDB (Page 161)" ("bufferWork" parameter) for access control.

The following picture show the circuitry of the parameter:

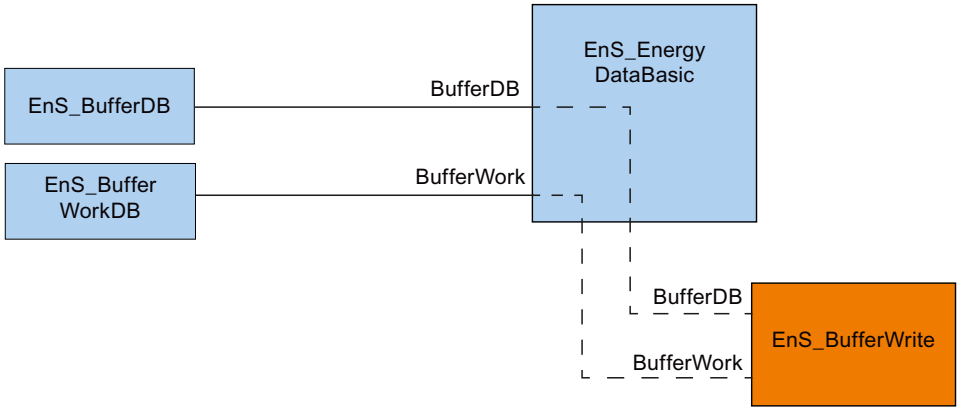

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 132)" parameter contains additional error information.

Alarms

The block has no signaling characteristics. Alarms are created by calling function block.

A.2.4.2 Parameters of EnS_BufferWrite

The following table shows the function block "EnS_BufferWrite":

Parameter	Declaration	Data type	Default	Description
bufferDB	Input	DB_ANY	-	Reference to the data block for buffering
busy	Output	BOOL	FALSE	TRUE = Processing is active
done	Output	BOOL	FALSE	TRUE = Data successfully processed
error	Ouput	BOOL	FALSE	TRUE = Error
status	Output	WORD	-	Error status information (Page 132)
bufferWork	InOut	EnS_typeBuffer-Work (Page 163)	-	Buffer monitoring and status information
request	InOut	EnS_typeBufferWri-teCtrl (Page 164)	-	Buffer monitoring information
energyMeta	InOut	EnS_typeEnergy-Meta (Page 168)	-	Measuring point-specific data

A.2.4.3 Parameter status

The following table shows the error codes that are generated at the "status" output parameter when errors occur:

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8501	Error writing to the buffer	Error writing	buffer-Work.write-Stat.statusWri-teFct	The writing is repeated until it is successful. Cause of error: • Wrong data block is connected or no data block is present. Procedure: • Correct the program.
8502	Alarm: Buffer at 100%, loss of data *		bufferWork.noI-tems / bufferWork.max-Item	Cause of error: • Data loss after buffer overflow • A buffer overflow occurs when the buffer cannot be cleared, for example, because the connection to WinCC (for archiving in WinCC) is interrupted. Procedure: • Make a connection to WinCC (archiving with WinCC) or eliminate the error (for example, memory full) on SIMATIC memory card.
8503	Warning: Buffer level above 80%, imminent loss of data		bufferWork.noI-tems / bufferWork.max-Item	Cause of error: • The number of ARRAY elements in the data block is >= 0.8 * maximum number of ARRAY elements. Procedure: • Increase the ARRAY limits in the properties of the data block.
8504	Warning: Buffer level above x%, imminent loss of data **		bufferWork.noI-tems / bufferWork.max-Items	Cause of error: • The number of ARRAY elements in the data block is < 0.8 * maximum number of ARRAY elements. Procedure: • Increase the ARRAY limits in the properties of the data block.

* The "bufferWork.noDeletedItems" parameter contains data records that are overwritten with a buffer overflow. The number of deleted data records is part of the alarm that is output by the "EnS_ArchiveDataLog (Page 113)" FB.

** The warning range is configured under "bufferWork.bufferLimit".

A.2.5 EnS_CalcEnergyData: Calculating energy values

A.2.5.1 Description of EnS_CalcEnergyData

Description

The "EnS_CalcenergyBasic" function block calculates the energy consumption and the average power over the individually configured archiving period (e.g. 15 min).

You can connect the block as follows:

- Power input
- Energy input
- Energy and power input

Block structure with connected power input

The graphic below shows the block structure with a connected power input:

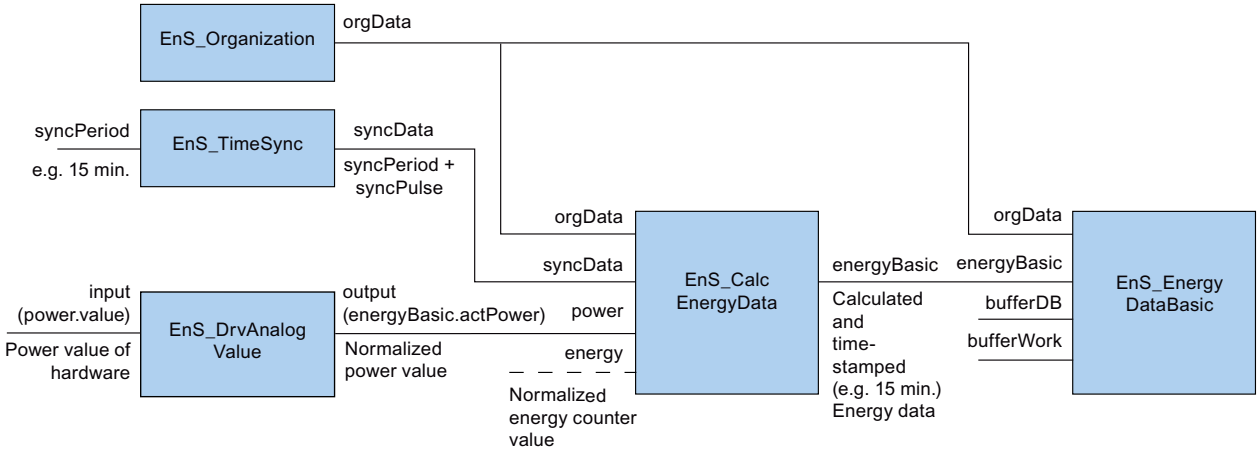

Current power

"power.value" is passed to the power output "energyBasic.actPower".

A dead range function can be configured for the calculated power value ("zeroCut" parameter). If the power value amounts to less than the configured value, the calculated power value is set to 0. Disable the dead range function by assigning "zeroCut" the value 0.

Current energy

At the start of the new archiving period, the current energy value "actEnergy" and the elapsed time of archiving period "syncData.syncPeriod" are reset.

"energyBasic.actEnergy" is calculated during the archiving period using the current power value of the current energy consumption.

The calculated energy value is cyclically added to the current energy value "energyBasic.actEnergy".

Energy counter value

"energyBasic.actEnergyCounter" and "energyBasic.energyCounter" are set to 0.

Block structure with connected energy input

The following graphic shows the block structure at a connected energy input.

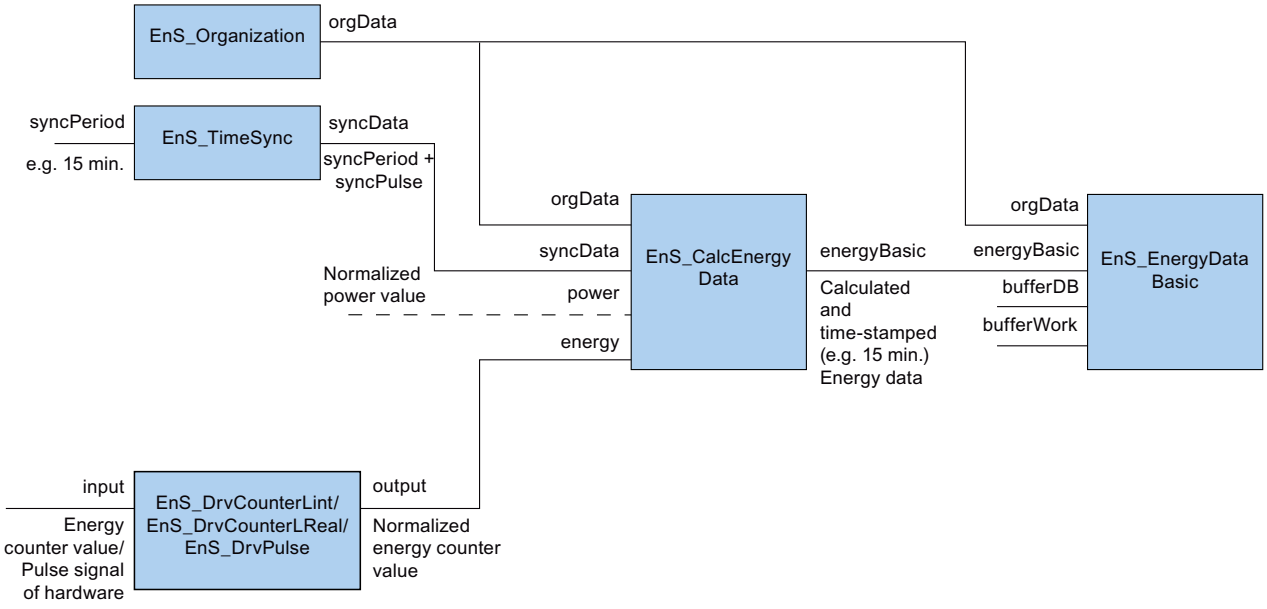

Current energy

The current power consumption "energyBasic.actEnergy" is calculated by subtracting the current energy counter value "energy.value" from the previous counter value.

At the start of the new archiving period, the current energy value "energyBasic.ectEnergy" and the elapsed time of the archiving period "syncData.syncPeriod" are reset.

Current power

The current power "energyBasic.actPower" is calculated at the end of the acquisition interval ("baseAcqTime" + "addAcqTime") based on the energy consumption.

Acquisition interval:

The acquisition interval for the power calculation is defined by the "baseAcqTime" and "addAcqTime" parameters. "baseAcqTime" specifies the minimum acquisition interval. "addAcqTime" specifies the additional acquisition interval.

The difference of the count is determined between the beginning and end of the acquisition interval.

To calculate the current power, the block waits at least the default acquisition interval "baseAcqTime". Calculation of the current power "energyBasic.actPower" starts when one of the requirements is met:

- The count value has changed.
- The additional acquisition interval "addAcqTime" has expired.
- The consumer has been switched off. In this case, the "energy.consumerStatus" parameter is set to FALSE.
  This condition is evaluated only when the consumer status is available ("energyMeta.consumerStatusEnable" = TRUE)

Energy counter value

The current energy counter value "energy.value" is copied to "energyBasic.actEnergyCounter" and "energyBasic.energyCounter" at the end of the archiving period.

Block structure with connected energy and power input

The graphic below shows the block structure with a connected energy and power input:

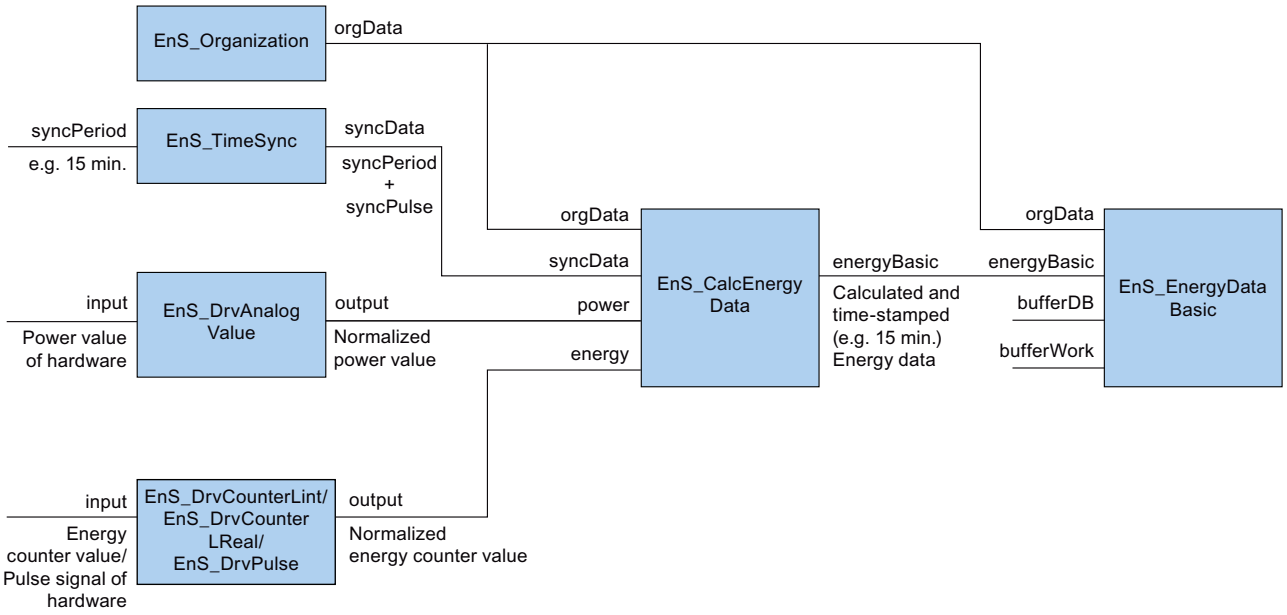

Current power

"power.value" is passed to the power output "energyBasic.actPower".

A dead range function can be configured for the calculated power value ("zeroCut" parameter). If the power value amounts to less than the configured value, the calculated power value is set to 0. Disable the dead range function by assigning "zeroCut" the value 0.

Current energy

The current energy consumption "energyBasic.actEnergy" is calculated by subtracting the current counter value "energy.value" from the initial counter value.

At the start of the new archiving period, the current energy value "energyBasic.actEnergy" and the elapsed time of the archiving period "syncData.syncPeriod" are reset.

Energy counter value

The current energy counter value "energy.value" is copied to "energyBasic.actEnergyCounter" and "energyBasic.energyCounter" at the end of the archiving period.

Average power / energy consumed

At the end of the archiving period "syncData.syncPeriod", the average power consumption "energyBasic.power" is calculated from the energy used "energyBasic.energy".

Status

The calculations are only performed if the input value is valid.

The status of the input value is written to the data structures "energy" ("energy.status") and "power" ("power.status").

If the input value is invalid, the corresponding output parameters are set to 16#FFFF_FFFF . The status is available for downstream blocks for further logging and evaluations.

Momentary values with the status 16#8301 are valid. However, the values are not suitable for calculating the consumed energy and the average power consumption in the current archiving period (for example, 15 minute period).. The values for the average momentary power consumption and the consumed energy are marked with the status 18#8001 .

Startup

During startup, all values are reset and the timers are restarted.

Note

During an incomplete archiving period, the values are set to the status "16#8301" (bad input values during the archiving period).

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 137)" parameter contains additional error information.

Alarms

The block has no signaling characteristics.

A.2.5.2 Parameters of EnS_CalcEnergyData

The following table shows the function block "EnS_CalcEnergyData":

Parameter	Declaration	Data type	Default	Description
power	Input	EnS_typeAnalog-Value (Page 162)	-	Input data power
energy	Input	EnS_typeEnergy-Counter (Page 167)	-	Input data energy

Parameter	Declaration	Data type	Default	Description
energyBasic	Output	EnS_typeEnergyBasic (Page 167)	-	Basic energy data
error	Output	BOOL	FALSE	TRUE = Error
status	Output	WORD	-	Error status information (Page 137)
orgData	InOut	EnS_typeOrgData (Page 172)	-	Organization data
syncData	InOut	EnS_typeSync (Page 173)	-	Synchronization data
energyMeta	InOut	EnS_typeEnergyMeta (Page 168)	-	Measuring point-specific data

A.2.5.3 Parameter status

The following table shows the error codes that are generated at the "status" output parameter when errors occur:

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8301	Incorrect input values during the archiving period	Hardware fault	energy.status / power.status	Internal error **Cause of error:** • Error occurred during archiving period and has already been corrected. **Procedure:** • No action needs to be taken when an internal error occurs.
8311	Linking error calculation	Connection error	energy.status / power.status	**Cause of error:** • The "energy" and "power" parameters are not interconnected. **Procedure:** • Program the interconnection.

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8312	Calculation interval "energyMeta.baseAcqTime" < 0	Parameter assignment error	energyMeta.baseAcqTime	**Cause of error:** • The duration of the calculation interval is invalid. **Procedure:** • Use valid values for the duration of the calculation interval: 0 - 3600
8313	Extension of calculation interval "energyMeta.addAcqTime" < 0		energyMeta.baseAcqTime / energyMeta.addAcqTime	**Cause of error:** • The duration of the extension of calculation interval is invalid. **Procedure:** • Use valid values for the duration of the extension of calculation interval: 0 - 3600
8314	Limit for zero point "energyMeta.zeroCut" < 0.0		energyMeta.zeroCut	**Cause of error**: • The value of the limit for the zero point is invalid. **Procedure:** • Use the valid values for the zero point limit: 0.0 - 3.402823e+38.
8315	"energy.value" >= "energyMeta.overflowCntValue"		energy.value / energyMeta.overflowCntValue	**Cause of error:** • The energy counter value on the "energy.value" parameter is greater than or equal to the overflow value on the "energyMeta.overflowCntValue" parameter. • The "overflowCntValue" parameter is not configured correctly (too small). **Procedure:** • Correct the configuration. • Increase the overflow value.
8316	Normalization factor "energyMeta.normFactorOut" ≤ 0.0 or > 9999.0	Parameter assignment error	energyMeta.normFactorOut	**Cause of error**: • The normalization factor at the "energyMeta.normFactorOut" parameter is invalid. **Procedure**: • Use valid values for the normalization factor: 0.0 - 9999.0.

A.2.6 EnS_DrvAnalogValue: Acquire power values of the REAL type

A.2.6.1 Description of EnS_DrvAnalogValue

Description

You use the "EnS_DrvAnalogValue" function block to acquire a power value. The input signal is normalized.

The output value in the "Output" structure is passed to the "EnS_CalcEnergyData (Page 133)" function block.

When electrical energy is measured, the output value has the "Wh" unit. If no electrical measured quantity is measured, for example, the flow rate, the output value has the unit "m³/h", for example.

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 140)" parameter contains additional error information.

A.2.6.2 Parameters of EnS_DrvAnalogValue

The following table shows the function block "EnS_DrvAnalogValue":

Parameter	Declaration	Data type	Default	Description
input	Input	REAL	0.0	Analog input signal
normFactor	Input	REAL	1.0	Normalization factor
hWFault	Input	BOOL	FALSE	TRUE = Hardware error of the input signal
sD	Input	VARIANT	-	Associated value for alarm message
messageSup-press	Input	BOOL	FALSE	TRUE = Alarm suppression activated
output	Output	EnS_typeAnalog-Value (Page 162)	-	Normalized output signal
error	Output	BOOL	FALSE	TRUE = Error
status	Output	WORD	-	Error status information (Page 140)
errorAlarm	Output	BOOL	FALSE	TRUE = Error of alarm block "Program_Alarm"
statusAlarm	Output	WORD	-	Error status information of the alarm block "Pro-gram_Alarm"
alarmState	Output	BYTE	-	Alarm status of the alarm block "Program_Alarm"

Parameter	Declaration	Data type	Default	Description
orgData	InOut	EnS_typeOrgData (Page 172)	-	Organization data
energyMeta	InOut	EnS_typeEnergy-Meta (Page 168)	-	Measuring point-specific data

A.2.6.3 Parameter status

The following table shows the error codes that are generated at the "status" output parameter when errors occur:

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8001	Hardware fault on measuring device	Hardware fault	hWFault	Causes of error: • Bad terminal on measuring device • Bad address Procedure: • Check the connection of the measuring device or correct the address.
8011	Normalization factor <= 0.0 or > 9999999.0	Parameter assignment error	normFactor	Causes of error: • Invalid value at "Normalization factor" setting Procedure: • Enter a valid value (0.0 - 9999999.0) for the "Normalization factor" setting.

A.2.6.4 Alarms of EnS_DrvAnalogValue

The alarms are generated with the instruction "Program_Alarm" You can find additional information on "Program_Alarm" in the online help of the TIA Portal information system under "Extended instructions > Alarms".

The following table shows the alarms of the "EnS_DrvAnalogValue" function block:

Alarm block	Alarm text	Alarm class
Program_Alarm	Hardware fault	With acknowledgment

The alarm can be changed.

The "sD" input parameter as an associated value. You can connect the parameters with additional information that is to be displayed in the alarm.

The alarm can be suppressed with "messageSuppress" or overridden with "orgData.messageSuppress" of the function block "EnS_Organization (Page 153)".

A.2.7 EnS_DrvCounterLInt: Acquire energy counter values of the LINT type

A.2.7.1 Description of EnS_DrvCounterLInt

Description

You use the "EnS_DrvCounterLInt" function block to acquire an integral energy meter value of the type LInt (64-bit). The input signal is converted to a standardized energy counter value.

The output value from the "EnS_CalcEnergyData (Page 133)" function block is processed as an energy counter value.

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 142)" parameter contains additional error information.

A.2.7.2 Parameters of EnS_DrvCounterLInt

The following table shows the function block "EnS_DrvCounterLInt":

Parameter	Declaration	Data type	Default	Description
input	Input	LINT	0.0	Input counter value
consumerStatus	Input	BOOL	FALSE	Consumer status bit: TRUE = On, FALSE = Off
hWFault	Input	BOOL	FALSE	TRUE = Hardware error of the input signal
sD	Input	VARIANT	-	Associated value for alarm message
messageSup-press	Input	BOOL	FALSE	TRUE = Alarm suppression activated
output	Output	EnS_typeEnergy-Counter (Page 167)	-	Output counter value
error	Output	BOOL	FALSE	TRUE = Error
status	Output	WORD	-	Error status information (Page 142)
errorAlarm	Output	BOOL	FALSE	TRUE = Error of alarm block "Program_Alarm"
statusAlarm	Output	WORD	-	Error status information of the alarm block "Pro-gram_Alarm"
alarmState	Output	BYTE	-	Alarm status of the alarm block "Program_Alarm"
orgData	InOut	EnS_typeOrgData (Page 172)	-	Organization data
energyMeta	InOut	EnS_typeEnergy-Meta (Page 168)	-	Measuring point-specific data

A.2.7.3 Parameter status

The following table shows the error codes that are generated at the "status" output parameter when errors occur:

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8001	Hardware error of the input device	Hardware fault	hWFault	**Causes of error:** • Bad terminal on measuring device • Bad address **Procedure:** • Check the connection of the measuring device
8011	Normalization factor <= 0.0 or > 9999999.0	Parameter assignment error	energyMeta.normFactorIn	**Causes of error:** • Invalid value at "Normalization factor" setting **Procedure:** • Enter a valid value (0.0 - 9999999.0) for the "Normalization factor" setting.
8012	Counter overflow value < 1.0 or > 1.0e+21		energyMeta.overflowCntValue	**Causes of error:** • Invalid value at "Overflow value" setting **Procedure:** • Enter a valid value (1.0 - 1.0e+21) for the user-defined "Overflow value" setting.

A.2.7.4 Alarms of EnS_DrvCounterLInt

The alarms are generated with the instruction "Program_Alarm" You can find additional information on "Program_Alarm" in the online help of the TIA Portal information system under "Extended instructions > Alarms".

The following table shows the alarms of the "EnS_DrvCounterLInt" function block:

Alarm block	Alarm text	Alarm class
Program_Alarm	Hardware fault	With acknowledgment

The alarm can be changed.

The "sD" input parameter as an associated value. You can connect the parameters with additional information that is to be displayed in the alarm.

The alarm can be suppressed with "messageSuppress" or overridden with "orgData.messageSuppress" of the function block "EnS_Organization (Page 153)".

A.2.8 EnS_DrvCounterLReal: Acquire energy counter values of the LREAL type

A.2.8.1 Description of EnS_DrvCounterLReal

Description

You use the "EnS_DrvCounterLInt" function block to acquire a floating point energy meter value of the type LREAL (64-bit). The input signal is converted into a normalized energy value.

The output value from the "EnS_CalcEnergyData (Page 133)" function block is processed as an energy counter value.

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 144)" parameter contains additional error information.

A.2.8.2 Parameters of EnS_DrvCounterLReal

The following table shows the function block "EnS_DrvCounterLReal":

Parameter	Declaration	Data type	Default	Description
input	Input	LREAL	0.0	Input counter value
consumerStatus	Input	BOOL	FALSE	User status bit: 1 = On, 0 = Off
hWFault	Input	BOOL	FALSE	FALSE = Hardware error of the input signal
sD	Input	VARIANT	-	Associated value for alarm message
messageSup-press	Input	BOOL	FALSE	TRUE = Alarm suppression activated
output	Output	EnS_typeEnergy-Counter (Page 167)	-	Output count values
error	Output	BOOL	FALSE	TRUE = Error
status	Output	WORD	-	Error status information (Page 144)
errorAlarm	Output	BOOL	FALSE	TRUE = Error of alarm block "Program_Alarm"
statusAlarm	Output	WORD	-	Error status information of the alarm block "Pro-gram_Alarm"
alarmState	Output	BYTE	-	Alarm status of the alarm block "Program_Alarm"
orgData	InOut	EnS_typeOrgData (Page 172)	-	Organization data
energyMeta	InOut	EnS_typeEnergy-Meta (Page 168)	-	Measuring point-specific data

A.2.8.3 Parameter status

The following table shows the error codes that are generated at the "status" output parameter when errors occur:

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8001	Hardware fault on measuring device	Hardware fault	hWFault	Causes of error: • Bad terminal on measuring device • Bad address Procedure: • Check the connection of the measuring device
8011	Normalization factor <= 0.0 or > 9999.0	Parameter assignment error	energyMeta.normFactorIn	Causes of error: • Invalid value at "Normalization factor" setting Procedure: • Enter a valid value (0.0 - 9999.0) for the "Normalization factor" setting.
8012	Counter overflow value < 1.0 or > 1.0e+21		energyMeta.overflowCntValue	Causes of error: • Invalid value at "Overflow value" setting Procedure: • Enter a valid value (1.0 - 1.0e+21) for the user-defined "Overflow value" setting.

A.2.8.4 Alarms of EnS_DrvCounterLReal

The alarms are generated with the instruction "Program_Alarm" You can find additional information on "Program_Alarm" in the online help of the TIA Portal information system under "Extended instructions > Alarms".

The following table shows the alarms of the "EnS_DrvCounterLReal" function block:

Alarm block	Alarm text	Alarm class
Program_Alarm	Hardware fault	With acknowledgment

The alarm can be changed.

The "sD" input parameter as an associated value. You can connect the parameters with additional information that is displayed in the alarm.

The alarm can be suppressed with "messageSuppress" or overridden with "orgData.messageSuppress" of the function block "EnS_Oganization (Page 153)".

A.2.9 EnS_DrvPulse: Acquiring pulse signals

A.2.9.1 Description of EnS_DrvPulse

Description

The "EnS_DrvPulse" function block adds the acquired pulses, converts them with the configured normalization factor and calculates an energy counter value of the LREAL type. The output value from "EnS_CalcEnergyData (Page 133)" is processed as a normalized energy counter value.

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 146)" parameter contains additional error information.

A.2.9.2 Parameters of EnS_DrvPulse

The following table shows the function block "EnS_DrvPulse":

Parameter	Declaration	Data type	Default	Description
input	Input	BOOL	FALSE	Pulse input
consumerStatus	Input	BOOL	FALSE	User status bit: TRUE = On, FALSE = Off
hWFault	Input	BOOL	FALSE	TRUE = Hardware error of the input signal
sD	Input	VARIANT	-	Associated value for alarm message
messageSup-press	Input	BOOL	FALSE	TRUE = Alarm suppression activated
output	Output	EnS_typeEnergy-Counter (Page 167)	-	Output count values
error	Output	BOOL	FALSE	TRUE = Error
status	Output	WORD	-	Error status information (Page 146)
errorAlarm	Output	BOOL	FALSE	TRUE = Error Program_Alarm
statusAlarm	Output	WORD	-	Error status information of the "Program_Alarm" alarm block
alarmState	Output	BYTE	-	Alarm status of the "Program_Alarm" alarm block
orgData	InOut	EnS_typeOrgData (Page 172)	-	Organization data
energyMeta	InOut	EnS_typeEnergy-Meta (Page 168)	-	Measuring point-specific data

A.2.9.3 Parameter status

The following table shows the error codes that are generated at the "status" output parameter when errors occur:

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8001	Hardware fault on measuring device	Hardware fault	hWFault	Causes of error: • Bad terminal on measuring device • Bad address Procedure: • Check the connection of the measuring device or correct the address.
8011	Normalization factor <= 0.0 or > 9999999.0	Parameter assignment error	energyMeta.normFactorIn	Causes of error: • Invalid value at "Normalization factor" setting Procedure: • Enter a valid value (0.0 - 9999999.0) for the "Normalization factor" setting.
8012	Counter overflow value < 1.0 or > 1.0e+21		energyMeta.overflowCntValue	Causes of error: • Invalid value at "Overflow value" setting Procedure: • Enter a valid value (1.0 - 1.0e+21) for the user-defined "Overflow value" setting.

A.2.9.4 Alarms of EnS_DrvPulse

The alarms are generated with the instruction "Program_Alarm" You can find additional information on "Program_Alarm" in the online help of the TIA Portal information system under "Extended instructions > Alarms".

The following table shows the alarms of the "EnS_DrvPulse" function block:

Alarm block	Alarm text	Alarm class
Program_Alarm	Hardware fault	With acknowledgment

The alarm can be changed by the user.

The "sD" input parameter as an associated value. You can connect the parameters with additional information that is to be displayed in the alarm.

The alarm can be suppressed with "messageSuppress" or overridden with "orgData.messageSuppress" of the function block "EnS_Organization (Page 153)".

A.2.10 EnS_EnergyDataAdv: Providing advanced energy values

A.2.10.1 Description of EnS_EnergyDataAdv

Description

The "EnS_EnergyDataAdv" function block provides advanced energy data record collected from a measuring point. The function block is used as a central interface to WinCC Professional or other programs (for example, SIMATIC Energy Manager Pro).

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 147)" parameter contains additional error information.

Alarms

The block has no signaling characteristics.

A.2.10.2 Parameters of EnS_EnergyDataAdv

The following table shows the block parameters of the "EnS_EnergyDataAdv" block:

Parameter	Declaration	Data type	Default	Description	HMI *
advData	Input	EnS_typeEnergyAdv (Page 165)	-	Advanced measured data record	X
error	Output	BOOL	-	TRUE = Error	-
status	Output	WORD	-	Error status information (Page 147)	-
orgData	InOut	EnS_typeOrgData (Page 172)	-	Organization data	-
energyMeta	InOut	EnS_typeEnergyMeta (Page 168)	-	Measuring point-specific data	-

* The highlighted parameters are assigned the system attribute "Accessible for HMI" and "Visible for HMI". They are used for operator control and monitoring functions on the HMI device, for example, for displaying a current value in a faceplate.

A.2.10.3 Parameter status

The "error" output parameter is set when an error occurs in a lower-level function block.

The "status" parameter contains additional error information. The error information is displayed in the "status" parameter until the error is cleared. Only then is the information about the next pending error displayed.

A.2.11 EnS_EnergyDataAdvMinMax: Providing advanced minimum and maximum values

A.2.11.1 Description of EnS_EnergyDataAdvMinMax

Description

The "EnS_EnergyDataAdvMinMax" function block provides minimum and maximum values of the advanced energy data for the output, for example, in WinCC Professional.

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The error information of the lower-level function block (from measurement hardware, for example) are output at the "status (Page 149)" output. The error information is displayed in the "status" parameter until the error is cleared. Only then is the information about the next pending error displayed.

Alarms

The block has no signaling characteristics.

A.2.11.2 Parameters of EnS_EnergyDataAdvMinMax

The following table shows the function block "EnS_EnergyDataAdvMinMax":

Parameter	Declaration	Data type	Default	Description	HMI *
advDataMin	Input	EnS_typeEnergyAdv (Page 165)	-	Data record with minimum values of the advanced energy values	X
advDataMax	Input	EnS_typeEnergyAdv (Page 165)	-	Data record with maximum values of the advanced energy values	X
error	Output	BOOL	-	TRUE = Error	
status	Output	WORD	-	Error status information (Page 149)	
orgData	InOut	EnS_typeOrgData (Page 172)	-	Organizational data	
energyMeta	InOut	EnS_typeEnergyMeta (Page 168)	-	Process tag specific data	

* The selected parameters receive the system attribute "Available for HMI" and "Visible in HMI". They are used for operator control and monitoring functions on the HMI device, for example, for displaying a current value in the faceplate.

A.2.11.3 Parameter status

The "error" output parameter is set when an error occurs in a lower-level function block.

The "status" parameter contains additional error information. The error information is displayed in the "status" parameter until the error is cleared. Only then is the information on the next pending error displayed.

A.2.12 EnS_EnergyDataBasic: Providing metadata and energy values

A.2.12.1 Description of EnS_EnergyDataBasic

Description

The "EnS_EnergyDataBasic" function block provides basic metadata and energy data record collected from a measuring point. The function block is used as a central interface to WinCC Professional or other programs (for example, SIMATIC Energy Manager Pro).

Metadata

The metadata at the "energyMeta" parameter with the "EnS_typeEnergyMeta (Page 168)" data type contains general information about the energy object or the measurement hardware.

Startup

The block has no startup characteristics.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 150)" parameter contains additional error information.

A.2.12.2 Parameters of EnS_EnergyDataBasic

The following table shows the function block "EnS_EnergyDataBasic":

Parameter	Declaration	Data type	Default	Description	HMI *
energyMeta	Input	EnS_typeEnergy-Meta (Page 168)	-	Measuring point-specific data	X
energyBasic	Input	EnS_typeEnergyBa-sic (Page 167)	-	Basic energy data	X
bufferDB	Input	DB_ANY	-	Pointer to the data block for buffering	-
sD	Input	VARIANT	-	Associated value for alarm messages	-
messageSup-press	Input	BOOL	FALSE	TRUE = Alarm suppression activated	X
error	Output	BOOL	FALSE	TRUE = Error	-

Parameter	Declaration	Data type	Default	Description	HMI *
status	Output	WORD	-	Error status information (Page 150)	-
status1	Output	WORD	-	Error status information 1	-
status2	Output	WORD	-	Error status information 2	-
status3	Output	WORD	-	Error status information 3	-
status4	Output	WORD	-	Error status information 4	-
status5	Output	WORD	-	Error status information 5	-
errorAlarm1	Output	BOOL	FALSE	TRUE = Error Program_Alarm1	-
statusAlarm1	Output	WORD	-	Error status information Program_Alarm1	-
alarmState1	Output	BYTE	-	Alarm status Program_Alarm1	-
errorAlarm2	Output	BOOL	FALSE	TRUE = Error Program_Alarm2	-
statusAlarm2	Output	WORD	-	Error status information Program_Alarm2	-
alarmState2	Output	BYTE	-	Alarm status Program_Alarm2	-
errorAlarm3	Output	BOOL	FALSE	TRUE = Error Program_Alarm3	-
statusAlarm3	Output	WORD	-	Error status information Program_Alarm3	-
alarmState3	Output	BYTE	-	Alarm status Program_Alarm3	-
errorAlarm4	Output	BOOL	FALSE	TRUE = Error Program_Alarm4	-
statusAlarm4	Output	WORD	-	Error status information Program_Alarm4	-
alarmState4	Output	BYTE	-	Alarm status Program_Alarm4	-
errorAlarm5	Output	BOOL	FALSE	TRUE = Error Program_Alarm5	-
statusAlarm5	Output	WORD	-	Error status information Program_Alarm5	-
alarmState5	Output	BYTE	-	Alarm status Program_Alarm5	-
orgData	InOut	EnS_typeOrgData (Page 172)	-	Organization data	-
bufferWork	InOut	EnS_typeBufferWork (Page 163)	-	Buffer monitoring and status information	-

* The selected parameters receive the system attribute "Available for HMI" and "Visible in HMI". They are used for operator control and monitoring functions on the HMI device, for example, for displaying a current value in the faceplate.

A.2.12.3 Parameter status

The error information of the lower-level function blocks (for example, "EnS_TimeSync") are added to the "status1", "status5" output parameter. The error information is displayed in the "status" parameter until the error is cleared. Only then is the information of the next pending error displayed.

A.2.12.4 Alarms of EnS_EnergyDataBasic

The alarms are generated with the instruction "Program_Alarm" You can find additional information on "Program_Alarm" in the online help of the TIA Portal information system under "Extended instructions > Alarms".

The following table shows the alarms of the "EnS_EnergyDataBasic" function block:

Alarm block	Alarm text	Alarm class
Program_Alarm1	Parameter assignment error <text list: EnS_Alarms:#status1>	With acknowledgment
Program_Alarm2	<Text list: EnS_Alarms:#status2>	With acknowledgment
Program_Alarm3	Archiving error <text list: EnS_Alarms:#status3>.	With acknowledgment
Program_Alarm4	Reserved (warning for limit violation)	With acknowledgment
Program_Alarm5	Reserved (warning for limit violation)	With acknowledgment

The additional alarm texts are written in the "EnS_Alarms" text list. The output parameters <"status1" to "status5" contain the value for the error information of the "EnS_EnergyDataBasic" FB and its lower-level FBs.

Alarm block	Error code (W#16#...)	Alarm text	Parameter	Function block
Program_Alarm1	801x	Hardware	status1	EnS_Drvxxx
	821x	Time synchronization		EnS_TimeSync (Page 154)
	831x	Calculation		EnS_CalcEnergyData (Page 133)
	8311	Linking error calculation		
Program_Alarm2	800x	Hardware fault	status2	EnS_Drvxxx
	8201	External synchronization signal lost		EnS_TimeSync (Page 154)
	8202	External synchronization signal not within valid time range		
	8203	Time leap synchronization time		
ProgramAlarm3	8501	Error writing to the buffer	status3	EnS_BufferWrite (Page 130)
	8502	Alarm: Buffer at 100%, loss of data		
	8503	Warning: Buffer level above 80%, imminent loss of data		
	8504	Warning: Buffer level above x%, imminent loss of data		
	8512	Possible data loss		EnS_BufferRead (Page 128)

The associated values are assigned as follows:

Alarm block	Associated value	Parameter	Meaning
Program_Alarm1	1	status1	Error status information 1
	2	sD	Free
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used
Program_Alarm2	1	status2	Error status information 2
	2	sD	Free
	3	-	Not used
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used
Program_Alarm3	1	status3	Error status information 3
	2	Reserved*	Reserved
	3	sD	Free
	4	-	Not used
	5	-	Not used
	6	-	Not used
	7	-	Not used
	8	-	Not used
	9	-	Not used
	10	-	Not used

* The associated values are reserved for future use.

The alarm can be changed.

The "sD" input parameter as an associated value. You can connect the parameters with additional information that is displayed in the alarm.

The alarm can be suppressed with "messageSuppress" or overridden with "orgData.messageSuppress" of the function block "EnS_Organization (Page 153)".

A.2.13 EnS_Organization: Providing organizational data

A.2.13.1 Description of EnS_Organization

Description

The "EnS_Organization" function block provides central organizational data that is accessed by other function blocks. The block must be called once by each cyclic interrupt OB.

The block manages the following data:

- Start-up flag ("orgData.startup")
  The startup flag is a parameter of the BOOL type. The parameter is at the "orgData.startup" output parameter of the "EnS_Organization" function block. The function block uses this parameter to signal if the program is still in the startup phase. You set the number of cycles for the startup phase at the input parameter "runUpCyc" of the function block. The default is a startup phase of ten cycles.
- Alarm suppression ("orgData.messageSuppress")
  You can use the parameter to specify if messages from the user program of the CPU are to be suppressed.
- Momentary time stamp (UTC) of the CPU
  The momentary time stamp corresponds to the CPU system time.

Startup

The block sets the "orgData.startup" output parameter for the number of configured startup cycles "runUpCyc" and suppresses the alarms using the "orgData.messageSuppress" parameter.

Reaction to error

When an error occurs, the output parameter "error" is set. The "status" parameter contains additional error information.

Alarms

The block has no signaling characteristics.

A.2.13.2 Parameters of EnS_Organization

The following table shows the function block "EnS_Organization":

Parameter	Declaration	Data type	Default	Description
runUpCyc	Input	UINT	10	Number of start-up cycles
messageSup-press	Input	BOOL	FALSE	TRUE = Alarm suppression activated
orgData	Output	EnS_typeOrgData (Page 172)	-	Organization data

Parameter	Declaration	Data type	Default	Description
error	Output	BOOL	FALSE	TRUE = Error
status	Output	WORD	-	Error status information

A.2.14 EnS_TimeSync: Synchronize timers

A.2.14.1 Description of EnS_TimeSync

Description

The block is used as a central clock to synchronize the block for energy acquisition "EnS_CalcEnergyData (Page 133)".

The clock pulse can be specified as follows:

The "syncData.syncPulse" clock generator is triggered once per archiving period by an external synchronization pulse "extSyncPulse", for example, from a power supply company, or the internal CPU time. The block evaluates the from rising edge of the synchronization pulse. The time stamp for the synchronization pulse at the "syncData.syncTimeStamp" parameter is rounded up or down to the nearest whole time value, for example, 15 minutes.

The synchronization pulse is only acquired within a specified range of validity. You define the range of validity with the "validTimeRange" block parameter. Tolerance can be set for a premature synchronization pulse. Premature synchronization pulses are still considered valid if they are acquired up to one CPU cycle too early. Premature synchronization pulses depend on the cycle time of the cyclic interrupt OB in which the "EnS_TimeSync" block is called.

If the cyclic interrupt OB is called once per second, the external synchronization pulse must be set no earlier than the 59th second.

If the cyclic interrupt OB is called twice per second, the external synchronization pulse must be set no earlier than the 58th second.

If the external synchronization pulse is set too early by a power supply company, the synchronization pulse of the block is not set and the error status 16#8202 is set.

Invalid pulses are identified by the status "16#8202" and acknowledged with a message.

- External synchronization "extEnable" = TRUE
  Examples:
  - External synchronization pulse "extSyncPulse" set to early:
    The period between the momentary CPU time stamp and the next period is less than the calculation interval of the function block. In this case, the synchronization time stamp is rounded up to the nearest whole value of the period.
    Calculation interval: 1 s
    Period: 15 min
    Current CPU time: 6:59:59:500
    Synchronization time stamp: 7:00:00:000
  - External synchronization pulse "extSyncPulse" set to late:
    The time period between the current CPU time stamp and the last period is less than the "validTimeRange". In this case, the synchronization time stamp is rounded down to the last whole value of the period.
    Calculation interval: 1 s
    Period: 15 min
    Scope: 10 s
    Current CPU time 7:00:00:001
    Synchronization time stamp: 7:00:00:000
  - External synchronization pulse "extSyncPulse" is out of range:
    The time period between the current CPU time stamp and the last period is greater than the "validTimeRange". In this case, the synchronization time stamp is discarded and an error message (status = 16#8202) appears.
    Calculation interval: 1 s
    Period: 15 min
    Scope: 10 s
    Current CPU time: 7:00:20:000
    Synchronization time stamp: Error message (status = 16#8202)
- Set the "syncPeriod" parameter
  If external synchronization is disabled "extEnable" = FALSE, the "syncPeriod" parameter contains the period of time for the synchronization, for example, 15 minutes.
- The archiving of energy data behaves as follows when a synchronization pulse is missing:
  - The energy data for archiving period is calculated until the synchronization pulse is set.
  - The power values are averaged over the complete archiving period beginning with the last synchronization pulse.
  - The energy count values are accumulated over the complete archiving period beginning with the last synchronization pulse.

Time stamps are output as UTC (Coordinated Universal Time).

You can create multiple instances of this block, and thereby produce more archiving periods. The usual periods are 15 minutes for electrical energy and 60 minutes for non-electrical media.

Startup

During startup, the internally running timers are restarted.

Reaction to error

When an error occurs, the output parameter "error" is set. The "Status (Page 157)" parameter contains additional error information.

Alarms

The block has no signaling characteristics.

A.2.14.2 Parameters of EnS_TimeSync

The following table shows the function block "EnS_TimeSync":

Parameter	Declaration	Data type	Default	Description
extEnable	Input	BOOL	FALSE	TRUE = Enable for external synchronization
extSyncPulse	Input	BOOL	FALSE	External synchronization pulse
syncPeriod	Input	TIME	T#15m	Archiving period
pulseTime	Input	TIME	T#2s	Duration of the synchronization pulse
validTimeRange	Input	TIME	T#10s	Scope for the external synchronization pulse
monitoringTi-meExtSync	Input	TIME	T#2s	Reserved
modeExtSync	Input	BOOL	FALSE	Reserved
offsetUTCToLo-calTime	Input	TIME	T#0s	Offset of UTC Coordinated Universal Time from the local time
syncData	Output	EnS_typeSync (Page 173)	-	Synchronization data
curTimeStamp	Output	DT	-	Momentary time stamp at block call
error	Output	BOOL	FALSE	TRUE = Error
status	Output	WORD	-	Error status information (Page 157)
orgData	InOut	EnS_typeOrgData (Page 172)	-	Organization data
resetErrorExt-Sync	InOut	BOOL	FALSE	Reserved

A.2.14.3 Parameter status

The following table shows the error codes that are generated at the "status" output parameter when errors occur:

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8201	The external synchronization pulse extSyncPulse was lost during the syncPeriod archiving period.	Hardware fault	extSyncPulse	**Causes of error:** • External synchronization pulse lost **Procedure:** • Check the hardware address of the pulse. • If the external pulse is unreliable, use the internal time synchronization (CPU time as clock).
8202	The external synchronization pulse extSyncPulse was not set within the validTimeRange range of validity.		extSyncPulse / validTimeRange	**Causes of error:** • External synchronization pulse was set outside the range of validity. The default setting for the range of validity is 10 s . **Procedure:** • If the external pulse is unreliable, use the internal time synchronization (CPU time as clock).
8203	The time interval of the current time stamp is longer than the duration of the syncPeriod archiving period.	Time error	curTimeStamp / syncPeriod	Internal error **Causes of error**: • The time interval of the current time stamp is longer than the duration of the archiving period. **Procedure:** • If the external pulse is unreliable, use the internal time synchronization (CPU time as clock).

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
8211	The duration of the syncPeriod archiving period is shorter than: • Duration of the synchronization pulse pulseTime or • 1 minute	Parameter assignment error	syncPeriod / pulseTime	Causes of error: • The duration of the synchronization pulse is longer than the duration of the archiving period or longer than 1 minute. Procedure: • If the external pulse is unreliable, use the internal time synchronization (CPU time as clock).
8212	The duration of the syncPeriod archiving period is longer than one day (24 hours).		syncPeriod	Causes of error: • The duration of the syncPeriod archiving period is longer than one day (24 hours). Procedure: • Enter a valid value (1 min - 24 h) at the setting for the duration of the archiving period.
8213	The duration of the syncPeriod archiving period is not an integer value.		syncPeriod	Causes of error: • The duration of the syncPeriod archiving period is not an integer value. Procedure: • Enter a valid value (integer) at the setting for the duration of the archiving period.
8214	The duration of the syncPeriod archiving period is not an integer part of one hour/one day. The following are valid, for example: • 1 minute • 5 minutes • 15 minutes • 12 hours The following are invalid, for example: • 7 minutes • 11 minutes • 5 hours		syncPeriod	Causes of error: • The duration of the syncPeriod archiving period is not an integer part of one hour/one day. Procedure: • Enter a valid value at the setting for the duration of the archiving period.
8215	The offset of the standard UTC time to the local time is not a multiple of 15 minutes or shorter than 15 minutes. The following are valid, for example: • 1 hour		extSyncPulse	Causes of error: • The offset of the UTC to the local time is not a multiple of 15 minutes or shorter than 15 minutes. Procedure: • Enter a valid value for the offset of the standard UTC time to the local time.

Error code (W#16#...)	Description	Error type	Copied to parameter	Solution
	• 4 hours • 30 minutes The following are invalid, for example: • 10 minutes • 25 minutes			

A.3 Data blocks (DBs)

A.3.1 EnS_BufferDB

A.3.1.1 Description of EnS_BufferDB

Description

The global data block "EnS_BufferDB" contains periodically acquired energy data in an array. The energy data is written by the "EnS_EnergyDataBasic (Page 149)" function block to the "EnS_BufferDB" data block. For this, the "EnS_BufferWrite (Page 130)" function block is called internally by the "EnS_EnergyDataBasic" function block.

The energy data is read from the "EnS_BufferDB" data block by the "EnS_ArchiveDataLog (Page 113)" function block or the Energy Suite SCADA component. For this, the "EnS_BufferRead (Page 128)" function block is called internally by the "EnS_ArchiveDataLog" function block.

You can adjust the size of arrays according to your requirements and the memory capacity of the CPU. To do this, set the number of desire ARRAY elements in the properties of the "EnS_BufferDB" data block. In the "EnS_BufferWorkDB (Page 161)" block, you also set a number of other parameters, for example, a fill level warning.

Each array element corresponds to the value of the type "EnS_typeValue (Page 174)". Four more values are contained in it

- Tag name
- Value
- Time stamp
- Status

You can add several data blocks to a CPU as long as the CPU has enough free memory available for buffering. A pair always includes two data blocks, "EnS_BufferDB" and "EnS_BufferWorkDB".

A.3.1.2 Structure of EnS_BufferDB

The following table shows the structure of the data block "EnS_BufferDB":

Element	Data type	Description
EnS_BufferDB	ARRAY[0..n] of "EnS_typeValue"	n+1 elements of calculated data values

A.3.2 EnS_BufferWorkDB

A.3.2.1 Description of EnS_BufferWorkDB

Description

The "EnS_BufferWorkDB" data block is a global data block of the type "EnS_typeBufferWork (Page 163)". The data block contains the control and status information of the buffer DB "EnS_BufferDB (Page 160)". The values of the "EnS_BufferWorkDB" data block are evaluated and described internally by the "EnS_BufferRead (Page 128)" and "EnS_BufferWrite (Page 130)" function blocks for the coordination of the read and write access to the buffer DB "EnS_BufferDB".

A.3.2.2 Structure of EnS_BufferWorkDB

The following table shows the structure of the data block "EnS_BufferWorkDB":

Element	Data type	Description
EnS_BufferWorkDB	EnS_typeBufferWork (Page 163)	Buffer monitoring and status information

A.4 User-defined data types (UDTs)

A.4.1 EnS_typeAnalogValue

A.4.1.1 Description of EnS_typeAnalogValue

Description

The "EnS_typeAnalogValue" PLC data type describes the interface between the function block for analog inputs ".EnS_DrvAnalogValue (Page 139)" and the function block for periodic calculation of energy data "EnS_CalcEnergyData (Page 133)".

A.4.1.2 Structure of EnS_typeAnalogValue

The following table shows the parameters of the "EnS_typeAnalogValue" PLC data type:

Parameter	Data type	Default	Description
value	REAL	0.0	Process value
status	WORD	16#FFFF	Status of the process value

A.4.2 EnS_typeBufferReadCtrl

A.4.2.1 Description of EnS_typeBufferReadCtrl

Description

The "EnS_typeBufferReadCtrl" PLC data type contains the required control information for reading data from the data block for buffering "EnS_BufferDB (Page 160)".

A.4.2.2 Structure of EnS_typeBufferReadCtrl

The following table shows the parameters of the "EnS_typeBufferReadCtrl" PLC type data:

Parameter	Data type	Default	Description
req	BOOL	FALSE	Request to read data from the data block for buffering "EnS_BufferDB (Page 160)"

A.4.3 EnS_typeBufferReadStat

A.4.3.1 Description of EnS_typeBufferReadStat

Description

The "EnS_typeBufferReadStat" PLC data type contains the data to be read and the status information for reading from the data block for buffering "EnS_BufferDB (Page 160)".

A.4.3.2 Structure of EnS_typeBufferReadStat

The following table shows the parameters of the "EnS_typeBufferReadStat" PLC data type:

Parameter	Data type	Default	Description
data	EnS_typeValue (Page 174)	-	Data read from the buffer
busy	BOOL	FALSE	TRUE = Processing is active
done	BOOL	FALSE	TRUE = Data successfully processed
error	BOOL	FALSE	TRUE = Error reading
status	WORD	16#0000	Status of the buffer
statusReadFct	WORD	16#0000	Status of "ReadFromArrayDB"

A.4.4 EnS_typeBufferWork

A.4.4.1 Description of EnS_typeBufferWork

Description

The "EnS_typeBufferWork" PLC data type contains all the control and status information for writing and reading data in the data block for buffering "EnS_BufferDB (Page 160)".

A.4.4.2 Structure of EnS_typeBufferWork

The following table shows the parameters of the "EnS_typeBufferWork" PLC data type:

Parameter	Data type	Default	Description
readCtrl	EnS_typeBuffer-ReadCtrl (Page 162)	-	Control information of the buffer for reading data
readStat	EnS_typeBufferRead-Stat (Page 163)	-	Data and status information of the buffer for reading data
writeCtrl	EnS_typeBufferWri-teCtrl (Page 164)	-	Control information of the buffer for writing data

Parameter	Data type	Default	Description
writeStat	EnS_typeBufferWrite-Stat (Page 165)	-	Status information of the buffer for writing data
busy	BOOL	FALSE	TRUE = Processing is active
done	BOOL	FALSE	TRUE = Data successfully processed
error	BOOL	FALSE	TRUE = Error in data processing
status	WORD	16#0000	Status of the buffer
maxItem	UDINT	0	Maximum number of data records of "EnS_BufferDB"
maxItemRead	UDINT	0	Reserved
maxCycleWrite	UDINT	30	Maximum number of cycles for writing data
cycleWrite	UINT	0	Number of cycles for writing data
noItems	UDINT	0	Number of written data records from "EnS_BufferDB"
noDeletedItems	UDINT	0	Number of deleted data records from "EnS_BufferDB" with a buffer overflow
writeItem	DINT	0	Write cursor on the data record
readItem	DINT	0	Read cursor on the data record
bufferLimit	UINT	20	Configurable buffer limit for warnings in [%] (validity range 1-79). Starting at an 80% fill level of the buffer, an error status is displayed by default (see also "EnS_Buffer-Write (Page 130)").

A.4.5 EnS_typeBufferWriteCtrl

A.4.5.1 Description of EnS_typeBufferWriteCtrl

Description

The "EnS_typeBufferWriteCtrl" PLC data type contains the periodic energy data to be written to the data block for buffering "EnS_BufferDB (Page 160)" and the required control information (for example, query) for writing data.

A.4.5.2 Structure of EnS_typeBufferWriteCtrl

The following table shows the parameters of the "EnS_typeBufferWriteCtrl" PLC data type:

Parameter	Data type	Default	Description
req	BOOL	FALSE	Request to write data to the buffer.
data	EnS_typeValue (Page 174)	-	Periodic energy data that are written to the buffer.

A.4.6 EnS_typeBufferWriteStat

A.4.6.1 Description of EnS_typeBufferWriteStat

Description

The "EnS_typeBufferWriteStat" PLC data type contains status information for writing periodic energy data to the data block for buffering "EnS_BufferDB (Page 160)".

A.4.6.2 Structure of EnS_typeBufferWriteStat

The following table shows the parameters of the "EnS_typeBufferWriteStat" PLC data type:

Parameter	Data type	Default	Description
busy	BOOL	FALSE	TRUE = Processing is active
done	BOOL	FALSE	TRUE = Data successfully processed
error	BOOL	FALSE	TRUE = Error in data processing
status	WORD	16#0000	Status of the buffer
statusWriteFct	WORD	16#0000	Write status

A.4.7 EnS_typeEnergyAdv

A.4.7.1 Description of EnS_typeEnergyAdv

Description

The "EnS_typeEnergyAdv" PLC data type contains advanced energy data which are read from measuring instruments and provided to a CPU or HMI system.

A.4.7.2 Structure of EnS_typeEnergyAdv

The following table shows the parameters of the "EnS_typeEnergyAdv" PLC data type:

Parameter	Data type	Default	Description
voltage1N	REAL	0.0	Momentary voltage between phase 1 and N
voltage2N	REAL	0.0	Momentary voltage between phase 2 and N
voltage3N	REAL	0.0	Momentary voltage between phase 3 and N
voltage12	REAL	0.0	Momentary voltage between phase 1 and 2
voltage23	REAL	0.0	Momentary voltage between phase 2 and 3
voltage31	REAL	0.0	Momentary voltage between phase 3 and 1
current1	REAL	0.0	Momentary current at phase 1
current2	REAL	0.0	Momentary current at phase 2
current3	REAL	0.0	Momentary current at phase 3
frequency	REAL	0.0	Momentary frequency
powerFactor1	REAL	0.0	Current power factor of Phase 1
powerFactor2	REAL	0.0	Current power factor of Phase 2
powerFactor3	REAL	0.0	Current power factor of Phase 3
totalPowerFactor	REAL	0.0	Momentary total power factor
appPower1	REAL	0.0	Current apparent power of Phase 1
appPower2	REAL	0.0	Current apparent power of Phase 2
appPower3	REAL	0.0	Current apparent power of Phase 3
totalAppPower	REAL	0.0	Momentary total apparent power
actPower1	REAL	0.0	Current active power of Phase 1
actPower2	REAL	0.0	Current active power of Phase 2
actPower3	REAL	0.0	Current active power of Phase 3
totalActPower	REAL	0.0	Momentary total active power
reactPower1	REAL	0.0	Current reactive power of Phase 1
reactPower2	REAL	0.0	Current reactive power of Phase 2
reactPower3	REAL	0.0	Current reactive power of Phase 3
totalReactPower	REAL	0.0	Momentary total reactive power
totalAppEnergy	LREAL	0.0	Momentary total apparent energy
totalActEnergy	LREAL	0.0	Momentary total active energy
totalReactEnergy	LREAL	0.0	Momentary total reactive energy
status	WORD	16#FFFF	Status of the extended data

A.4.8 EnS_typeEnergyBasic

A.4.8.1 Description of EnS_typeEnergyBasic

Description

The "EnS_typeEnergyBasic" PLC data type contains basic energy data that are provided for a CPU, an HMI system and archiving, for example:

- Periodically calculated energy consumption
- Average power over the period
- Current values for energy and power consumption

A.4.8.2 Structure of EnS_typeEnergyBasic

The following table shows the parameters of the "EnS_typeEnergyBasic" PLC data type:

Parameter	Data type	Default	Description
actEnergy	REAL	0.0	Momentary energy value
actPower	REAL	0.0	Momentary power value
actEnergyCounter	LREAL	0.0	Current energy counter value
actStatus	WORD	16#FFFF	Status of actual values
energy	REAL	0.0	Cumulative energy value at the end of the archiving period
power	REAL	0.0	Average power value at the end of the archiving period
energyCounter	LREAL	0.0	Energy counter value at the end of the archiving period
status	WORD	16#FFFF	Status of the cumulative energy and average power
syncData	EnS_typeSync (Page 173)	-	Synchronization data

A.4.9 EnS_typeEnergyCounter

A.4.9.1 Description of EnS_typeEnergyCounter

Description

The "EnS_typeEnergyCounter" PLC data type contains an interface between the driver function block for energy counter values "EnS_DrvCounterLInt (Page 141)", "EnS_DrvCounterLReal (Page 143)" and the function block for the periodic calculation of energy data "EnS_CalcEnergyData (Page 133)".

A.4.9.2 Structure of EnS_typeEnergyCounter

The following table shows the parameters of the "EnS_typeEnergyCounter" PLC data type:

Parameter	Data type	Default	Description
value	LREAL	0.0	Energy counter value
normValue	LREAL	0.0	Normalized energy count value
consumerStatus	BOOL	FALSE	Consumer status: TRUE = On, FALSE = Off
status	WORD	16#FFFF	Status of the energy value

A.4.10 EnS_typeEnergyMeta

A.4.10.1 Description of EnS_typeEnergyMeta

Description

The "EnS_typeEnergyMeta" PLC data type contains information about the measuring point. The information is used for display in the CPU or in an HMI system.

A.4.10.2 Structure of EnS_typeEnergyMeta

The following table shows the parameters of the "EnS_typeEnergyMeta" PLC data type:

Parameter	Data type	Default	Value range	Description	HMI [1)]
name	STRING[32]	" "	-	Energy object name	X
plantIdentCode	STRING[128]	" "	-	Plant designation	-
localIdentCode	STRING[128]	" "	-	Location identifier	-
techUnitId	DWORD	0	0 = Not configured	ID for technological unit	-
deviceId	DWORD	0	0 = Invalid 16#0000_0001 - 16#FFFF_FFFF	Device ID for the measuring device	X
parentDeviceId	DWORD	0	0 = Not configured	Device ID of the higher-level measuring device (hierarchy relationship)	X
id	ARRAY[0..9] of HW_IO	0	0 = Not configured	Hardware IDs of the measuring device in the hardware configuration	-
systemParam	ARRAY[0..6] of LREAL	0	-	System parameters for internal use	-
connection-Type[2)]	USINT	0	0 - 16 (see [2)])	Connection type of the measuring device	-

Parameter	Data type	Default	Value range	Description	HMI [1)]
inputType	USINT	0	0 = Invalid 1 = Pulse signal 2 = Energy counter value 3 = Power value 4 = Energy counter value and power value	Type of input signal	X
mode	USINT	0	0 = Invalid 1 = Consumer 2 = Producer	Configuration of the energy flow direction	X
accuracyVoltConv	REAL	0.0	0.0 = Precision setting is disabled Range of validity: 0.0 - 100.0	Conversion error of the voltage transformer in [%]	-
accuracyCurrConv	REAL	0.0	0.0 = Precision setting is disabled Range of validity: 0.0 - 100.0	Conversion error of the momentary current transformer in \ [%]	-
accuracyDevice	REAL	0.0	0.0 = Precision setting is disabled Range of validity: 0.0 - 100.0	Measuring error of the measuring device in [%]	-
accuracyAdd	REAL	0.0	0.0 = Precision setting is disabled Range of validity: 0.0 - 100.0	Additional error factor in [%]	-
correctionFactor	REAL	0.0	0.0 = Invalid Range of validity: 0.001 - 1d	Correction factor	-
syncPeriod	REAL	900.0	60.0 - 86400.0 (= 1min -1d)	Archiving period in [s]	X
overflowCntValue	LREAL	0.0	1.0 - 1.0e+21	Overflow value of the meter	X
normFactorIn	REAL	1.0	0.0 - 9999.0	Scale factor for input values (same scale factor for energy and power values)	X
normFactorOut	REAL	0.001	0.0 - 9999.0 (0.001 = Output corresponds to kWh)	Scale factor for output values	X
zeroCut	REAL	0.1	0.0 - 3.402823e+38	Limit for zero point during the energy consumption calculation	X
baseAcqTime	REAL	10.0	0.0 - 3600.0 (1 h) See also EnS_CalcEnergyData (Page 133)	Calculation interval for the instantaneous value of the power consumption in [s]	X

Parameter	Data type	Default	Value range	Description	HMI [1]
addAcqTime	REAL	10.0	0.0 - 3600.0 (=1h) See also EnS_CalcEnergyData (Page 133)	Dynamic extension of the calculation interval for the instantaneous value of the power consumption in [s]	X
consumerStatusEnable	BOOL	FALSE	TRUE = on FALSE = off	Consumer/generator status is active	-
coordinationEnable	BOOL	FALSE	-	TRUE = Coordination of acyclic communication services is enabled	-
cyclicRd	BOOL	FALSE	-	TRUE = Cyclic reading of acyclic energy data is enabled	-
cyclicTime	REAL	60.0	0.0 - 3600.0 (1 h)	Period for the cyclic reading of acyclic energy data in [s]	-
unitActEnergy[3]	UINT	0	see [3]	Unit for active energy	-
unitActPower[3]	UINT	0	see [3]	Unit for active power	-
unitAppEnergy[3]	UINT	0	see [3]	Unit for apparent energy	-
unitAppPower[3]	UINT	0	see [3]	Unit for apparent power	-
unitReactEnergy[3]	UINT	0	see [3]	Unit for reactive energy	-
unitReactPower[3]	UINT	0	see [3]	Unit for reactive power	-
unitPowerFactor[3]	UINT	0	see [3]	Unit for power factor	-
unitCurrent[3]	UINT	0	see [3]	Unit for current	-
unitVoltage[3]	UINT	0	see [3]	Unit for voltage	-
unitFrequency[3]	UINT	0	see [3]	Unit of frequency	-
unitAmpUnbalCurrent[3]	UINT	0	see [3]	Unit for amplitude balance of the current	-
unitAmpUnbalVoltage[3]	UINT	0	see [3]	Unit for amplitude symmetry of voltage	-
unitAddValue[3]	ARRAY[0..19]	0	see [3]	Reserved	-
typeAddValue	ARRAY[0..19]	0	0 - 16 (see InputType)	Reserved	-
archiveEnergy	BOOL	FALSE	-	FALSE = Archiving of the energy value deactivated, TRUE = Archiving of the energy value activated	-
archivePower	BOOL	FALSE	-	FALSE = Archiving of the power value deactivated, TRUE = Archiving of the power value activated	-
archiveEnergyCounter	BOOL	FALSE	-	FALSE = Archiving of the energy counter value deactivated, TRUE = Archiving of the energy counter value activated	-
archiveAddValue	ARRAY[0..19]	0	-	Reserved	-

[1] The selected parameters are assigned the system attribute "Available for HMI" and "Visible in HMI". They are used for operator control and monitoring functions on the HMI device, for example, displaying a current value in a faceplate.

[2] Enumeration for the connection type of the measuring device (Page 175)

[3] Enumeration for units (Page 175)

See also

Parameter status (Page 132)

A.4.11 EnS_typeHMIWriteCtrl

A.4.11.1 Description of EnS_typeHMIWriteCtrl

The EnS_typeHMIWriteCtrl PLC data type contains control information for writing the energy data to the SCADA system.

A.4.11.2 Structure of EnS_typeHMIWriteCtrl

The following table shows the parameters of the "EnS_typeHMIWriteCtrl" PLC data type:

Parameter	Data type	Default	Description	HMI *
req	BOOL	FALSE	Request for writing the energy data to the SCADA system	X
maxItem	UDINT	0	Maximum number of data records in the buffer	X
noItems	UDINT	0	Number of data records written in the SCADA system	X
noDeletedItems	UDINT	0	Number of deleted data records with a buffer overflow	X
writeItems	DINT	0	Write cursor on the data record in the buffer	X
readItems	DINT	0	Read cursor on the data record in the buffer	X
watchdog	UINT	0	Watchdog counter	X

* The highlighted parameters are assigned the system parameters "Accessible for HMI" and "Visible for HMI". The parameters are used for data exchange with the Energy Suite SCADA component.

A.4.12 EnS_typeHMIWriteStat

A.4.12.1 Description of EnS_typeHMIWriteStat

The EnS_typeHMIWriteStat PLC data type contains status information for the writing the energy data to the data log of the SCADA system.

A.4.12.2 Structure of EnS_typeHMIWriteStat

The following table shows the parameters of the "EnS_typeHMIWriteStat" PLC data type:

Parameter	Data type	Default	Description	HMI *
busy	BOOL	FALSE	TRUE = WinCC script: in progress	X
done	BOOL	FALSE	TRUE = Energy data is successfully written to the SCADA system	X
error	BOOL	FALSE	TRUE = Error occurred writing the energy data	X
status	WORD	16#0000	Status of the WinCC script	X
noItems	UDINT	0	Number of written data records	X
readItem	DINT	0	Read cursor on the data record in the buffer	X
writeHMIItem	DINT	0	Write cursor on the data record in the data log	X
watchdog	UINT	0	Watchdog counter	X

* The highlighted parameters are assigned the system parameters "Accessible for HMI" and "Visible for HMI". The parameters are used for data exchange with the Energy Suite SCADA component.

A.4.13 EnS_typeOrgData

A.4.13.1 Description of EnS_typeOrgData

Description

The "EnS_typeOrgData" PLC data type contains higher-level data and settings that all blocks access. In this way, the information can be forwarded with priority to the blocks, for example, an alarm lock.

A.4.13.2 Structure of EnS_typeOrgData

The following table shows the parameters of the "EnS_typeOrgData" PLC data type:

Parameter	Data type	Default	Description
startup	BOOL	TRUE	TRUE = Startup is active
messageSuppress	BOOL	FALSE	TRUE = Alarm suppression
currTimeStamp	DT	-	Current time stamp (UTC format)

A.4.14 EnS_typeSync

A.4.14.1 Description of EnS_typeSync

Description

The "EnS_typeSync" PLC data type contains all the information about the archiving period including the synchronization pulse.

A.4.14.2 Structure of EnS_typeSync

The following table shows the parameters of the "EnS_typeSync" PLC data type:

Parameter	Data type	Default	Description	HMI *
syncPeriod	REAL	900.0	Archiving period in [s]	X
syncTime	REAL	0.0	Remaining time of the archiving in [s]	X
syncPulse	BOOL	FALSE	Synchronization pulse	-
syncTimeStamp	DT	DT#1990-01-01-00:00:00.0	Time stamp of the synchronization pulse	X
error	BOOL	FALSE	1 = Error	-
status	WORD	16#0000	Error status information	-

* The selected parameters become the system parameters "Available for HMI" and "Visible for HMI". They can be used for operator control and monitoring functions on the HMI.

A.4.15 EnS_typeValue

A.4.15.1 Description of EnS_typeValue

Description

The "EnS_typeValue" PLC data type contains all the information of periodically acquired energy data to be archived, for example, to the data log. The global data block EnS_BufferDB (Page 160)" consists of an ARRAY of the "EnS_typeValue" PLC data type.

A.4.15.2 Structure of EnS_typeValue

The following table shows the parameters of the "EnS_typeValue" PLC data type:

Parameter	Data type	Default	Description	HMI *
tagname	STRING[64]	" "	Tag name	X
value	LREAL	0.0	Periodic energy value	X
timeStamp	DT	DTL#1990-01-01-00:00:00.0	Time stamp of the periodic energy value	X
status	WORD	16#0000	Status of the periodic energy value	X

A.5 Definition of enumerations

A.5.1 Enumeration for the connection type of the measuring device

The following table shows the enumeration for the connection type of the measuring instrument:

Number	Connection type
0	Not defined
1	1 phase
2	1 phase - 3 phase (symmetrical)
3	3 phases (symmetrical)
4	3 phases + N (asymmetrical) without voltage transformer / with 3 current transformers 3P4W
5	3 phases + N (asymmetrical) with voltage transformer / with 3 current transformers 3P4W
6	3 phases + N (symmetrical) without voltage transformer / with 1 current transformer 3P4WB
7	3 phases + N (symmetrical) with voltage transformer / with 1 current transformer 3P4WB
8	3 phases (asymmetrical) without voltage transformer / with 3 current transformers 3P3W
9	3 phases (asymmetrical) with voltage transformer / with 3 current transformers 3P3W
10	3 phases (asymmetrical) without voltage transformer / with 3 current transformers 3P3W
11	3 phases (asymmetrical) with voltage transformer / with 2 current transformers 3P3W
12	3 phases (symmetrical) without voltage transformer / with 1 current transformer 3P3WB
13	3 phases (symmetrical) with voltage transformer / with 1 current transformer 3P3WB
14	2 phases + N (asymmetrical) without voltage transformer / with 2 current transformers 3P4W
15	1 phase + N without voltage transformer / with 1 current transformer 1P2W
16	3 phases + N (asymmetrical) with voltage transformer / with 3 current transformers 3P3W

A.5.2 Enumeration for units

The following table shows the enumeration for the units:

Number	Unit
Electrical units	
0	Not defined
1209	A

Number	Unit
1210	kA
1240	V
1242	kV
1241	MV
1186	W
1190	kW
1189	MW
1188	GW
1175	Wh
1179	kWh
1178	MWh
1177	GWh
1342	%
1077	Hz
Non-electrical units	
1034	m^3
1573	Nm^3
1038	l
1041	hl
1088	kg
1092	t
32800	pcs
1598	m^3/h
1590	Nm^3/h
1353	l/h
1635	hl/h
1324	kg/h
1328	t/h
32801	pcs/h

SIEMENS

Overview of working with Team Engineering 1

Using Multiuser Engineering 2

Using Multiuser Commissioning 3

Commissioning projects with PLC synchronization 4

Exchanging data with Inter Project Engineering (IPE) 5

Using Team Engineering

Programming and Operating Manual

10/2018

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 10/2018 Subject to change

Copyright © Siemens AG 2018.
All rights reserved

Table of contents

1 Overview of working with Team Engineering....7

2 Using Multiuser Engineering....13

2.1 Introduction to Multiuser Engineering....13

2.2 Overview of the mode of operation with Multiuser Engineering....16

2.3 Basics of the multiuser server concept....21

2.4 Requirements for working with Multiuser Engineering....24

2.5 Supported objects for Multiuser Engineering....27

2.6 Settings for working with Multiuser Engineering....29

2.7 Improvement of the Multiuser Engineering functionality....32

2.8 Requirements for multiuser projects....34

2.9 Upgrading multiuser projects....36

2.10 Working offline with Multiuser Engineering....37

2.11 Operator controls in Multiuser Engineering....39
2.11.1 Multiuser icons in the user interface....39
2.11.2 Multiuser icons in the user interface....40
2.11.3 Multiuser banner in the user interface....42
2.11.4 Functionality of the multiuser editor....43

2.12 Installing and uninstalling the Multiuser server....47
2.12.1 Notes on the installation of the multiuser server....47
2.12.2 Notes on the compatibility of the multiuser server....48
2.12.3 Installing the multiuser server....50
2.12.4 Installing the multiuser server again without loss of data....52
2.12.5 Uninstalling the multiuser server....53

2.13 Configuring and managing the multiuser server....55
2.13.1 Options for configuring and administering the multiuser server....55
2.13.2 Using the graphic tool....56
2.13.2.1 Introduction to the graphic multiuser tools....56
2.13.2.2 Configuring a multiuser server with the graphic Configuration Tool....57
2.13.2.3 Managing a multiuser server with the graphic Administration Tool....62
2.13.2.4 Adding and deleting a new server connection....64
2.13.2.5 Define "User administration" for the multiuser servers and projects....67
2.13.2.6 Managing alerts....70
2.13.2.7 Managing multiuser projects and local sessions....71
2.13.2.8 Showing the history for projects and managing revisions....74
2.13.2.9 Switching back and forth between engineering and commissioning mode....78
2.13.2.10 Selecting a user interface language....80
2.13.2.11 Displaying confirmations....80
2.13.2.12 Managing certificates....82
2.13.3 Using the command line tools....83

2.13.3.1 Introduction to the multiuser command line tools....83
2.13.3.2 Configuring the multiuser server with the administrative tools....85
2.13.3.3 Commands for configuring the multiuser server....87
2.13.3.4 Managing the multiuser server with the power tools....89
2.13.3.5 Commands for managing the multiuser server....90
2.13.4 Confirm the certificate for a multiuser server....97
2.13.5 Enter access data for the multiuser server....97
2.13.6 Transferring server data to other multiuser servers....98
2.13.7 Reading out error log for multiuser server....99
2.13.8 Error log for multiuser server packed as Zip file....101
2.13.9 Information on response of the multiuser server to errors....101
2.13.10 Supported network configurations for the multiuser server....102
2.13.10.1 Requirements for successful client-server communication....102
2.13.10.2 Network with dedicated multiuser server....104
2.13.10.3 Network with temporary multiuser server....106
2.13.10.4 Network in a Windows domain....108
2.13.10.5 Network in a Windows domain with Internet access via VPN....110
2.13.10.6 Network in a Windows domain with Internet access via proxy....113

2.14 Working with the multiuser server....117
2.14.1 Introduction to the multiuser server....117
2.14.2 Setting the multiuser network profiles....118
2.14.3 Adding a new multiuser server connection....120
2.14.4 Displaying the multiuser server connection....121
2.14.5 Editing the multiuser server connection....121
2.14.6 Restore multiuser server connection....122
2.14.7 Deleting the multiuser server connection....123

2.15 Working with the local multiuser server....125
2.15.1 Introduction to the local multiuser server....125
2.15.2 Configuring the local multiuser server....126
2.15.3 Displaying the local multiuser server....127
2.15.4 Starting the local multiuser server....128
2.15.5 Exiting the local multiuser server....129

2.16 Creating and managing multiuser server projects....130
2.16.1 Introduction to working in multiuser server projects....130
2.16.2 Adding a new multiuser server project....131

2.17 Working in the local session....133
2.17.1 Creating and managing a local session....133
2.17.1.1 Creating a local session....133
2.17.1.2 Opening a local session....134
2.17.1.3 Saving a local session....137
2.17.1.4 Closing a local session....138
2.17.1.5 Deleting a local session....138
2.17.1.6 Displaying all local sessions associated with a server project....140
2.17.1.7 Exporting a local session as single-user project....141
2.17.2 Editing objects in the local session....142
2.17.2.1 Introduction to working in local sessions....142
2.17.2.2 Working offline in the local session....145
2.17.2.3 Rules for selecting objects....146
2.17.2.4 Marking objects automatically or manually for editing....148
2.17.2.5 Possible conflicts during autoselect....151

2.17.2.6 Unmarking objects....152
2.17.2.7 Displaying usage info for marked objects....152
2.17.2.8 Refresh local session....153
2.17.2.9 Restoring the local session after the refresh....155
2.17.2.10 Check-in objects to the server project....156
2.17.2.11 Possible conflicts during check-in....159
2.17.2.12 Possible error messages during check-in....161
2.17.2.13 Deleting objects in the local session....163

2.18 Working in the server project view....166
2.18.1 Introduction to working in the server project view....166
2.18.2 Opening and closing the server project view....168
2.18.3 Editing objects in the server project view....169
2.18.4 Renaming objects in the server project view....171

3 Using Multiuser Commissioning....173

3.1 Introduction to Multiuser Commissioning....173

3.2 Overview of the mode of operation with Multiuser Commissioning....175

3.3 Settings for working with Multiuser Commissioning....178

3.4 Requirements for working with Multiuser Commissioning....180

3.5 Enabling and disabling commissioning mode....182

4 Commissioning projects with PLC synchronization....185

4.1 Basics of PLC synchronization....185

4.2 Requirements for PLC synchronization....189

4.3 Procedure for PLC synchronization....190

4.4 Rules for PLC synchronization....194

5 Exchanging data with Inter Project Engineering (IPE)....201

5.1 Basics of Inter Project Engineering (IPE)....201

5.2 Requirements for Inter Project Engineering (IPE)....204

5.3 Overview for working with Inter Project Engineering (IPE)....205

5.4 Creating "Device proxy data" in the source project....208

5.5 Creating an IPE file by means of the "Device proxy data"....209

5.6 Using controller data from other projects with IPE....210
5.6.1 Using controller data from other projects in an HMI device 210
5.6.1.1 Basics on controller data....210
5.6.1.2 Initializing a device proxy via an IPE file....210
5.6.1.3 Initializing a device proxy via a project file....216
5.6.2 Communication with device proxies....221
5.6.2.1 Basics of communication with device proxies....221
5.6.2.2 Configuring a direct connection....223
5.6.2.3 Configuring a routed connection....225
5.6.3 Integrated configuring with WinCC and SIMATIC Manager....228
5.6.3.1 Basic principles for configuring with WinCC and SIMATIC Manager....228
5.6.3.2 Initializing a device proxy via a STEP 7 project file....229
5.6.3.3 Adapting network parameters....230

5.6.3.4 Checking inconsistent data....235
5.6.3.5 Adapting tags and connections....236
5.6.3.6 Updating a device proxy via a STEP 7 project file....236
5.6.3.7 Including tags from the SIMATIC Manager....237
5.6.3.8 Configuring the "Direct keys" system function....238

Index....243

Overview of working with Team Engineering 1

Introduction

You have a choice of different functionalities for editing your automation tasks in the TIA Portal; all of these functionalities are grouped under the collective term Team Engineering .

Team Engineering supports convenient working in a team.

The advantage of this is that you can work with multiple users simultaneously in projects. This approach significantly reduces the configuring times for your projects. You can commission your plant faster and thus speed up the start of production.

Team Engineering offers different functions for the following development phases:

- For the engineering phase
- For the commissioning phase

In the engineering phase you develop your projects as part of a team. A suitable and useful project structure is created for working together on the project that takes into consideration all the requirements of the project. This includes, for example, a completely configured hardware configuration and an expedient distribution of the projects into program parts or software units that can be edited individually.

In the commissioning phase you take your project into operation together and at the same time. You download your project versions to the CPU with the support of the corresponding Team Engineering functionalities.

Functionalities of Team Engineering

TIA Portal offers different functions and options for working together on a project in a team and at the same time:

- Multiuser Engineering and Multiuser Commissioning
- Software Units
- PLC synchronization
- Inter Project Engineering (IPE)

Below you will find a list of the different options for working with all functionalities of Team Engineering .

Multiuser Engineering and Multiuser Commissioning

With Multiuser Engineering you can edit multiuser projects together and at the same time in multiple local sessions. You create your projects offline and initially download them to the multiuser server. The multiuser server supports the convenient synchronization of projects and allows you to conduct tests locally.

With Multiuser Commissioning you can conveniently commission your completely configured projects within the team. The multiuser server project is also updated after each download to the CPU during the commissioning phase, and a new revision is created for the server project.

This has the following advantages:

Synchronization of the multiuser project takes place over the server project when working in the local sessions.

The project versions in the CPU and on the multiuser server are always consistent after download to the CPU.

See also:

Introduction to Multiuser Engineering (Page 13)

Introduction to Multiuser Commissioning (Page 173)

Combined application options:

The functionality of software units and IPE can also be used as part of Multiuser Engineering and Multiuser Commissioning.

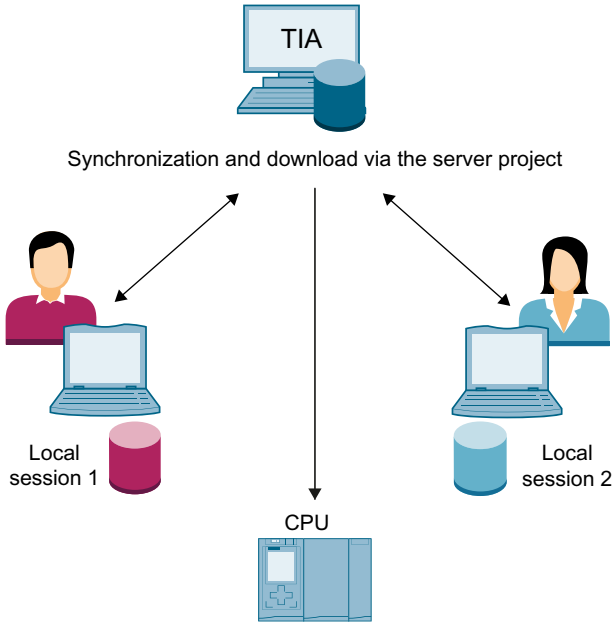

PLC synchronization

A "master project" that already contains the completely configured hardware configuration with all required program objects is used as the basis for PLC synchronization. This project is downloaded to the jointly used CPU and then distributed by means of project copies to up to five participating engineering systems. The project copies of the master project can be edited independently and offline at the same time. Depending on the firmware version of the CPU in use, certain online functions can also be executed at the same time from several engineering systems on the shared CPU, for example: Monitoring and operator control of blocks on the CPU and trace functions. Only one ES can download data to the CPU at any given time. The changes made by the other editors are displayed in the "Software synchronization" dialog during downloading and are synchronized automatically, if possible. A manual synchronization is required for concurrent changes. You are supported by the compare editor during manual synchronization of your changes. See also: Basics of PLC synchronization (Page 185)	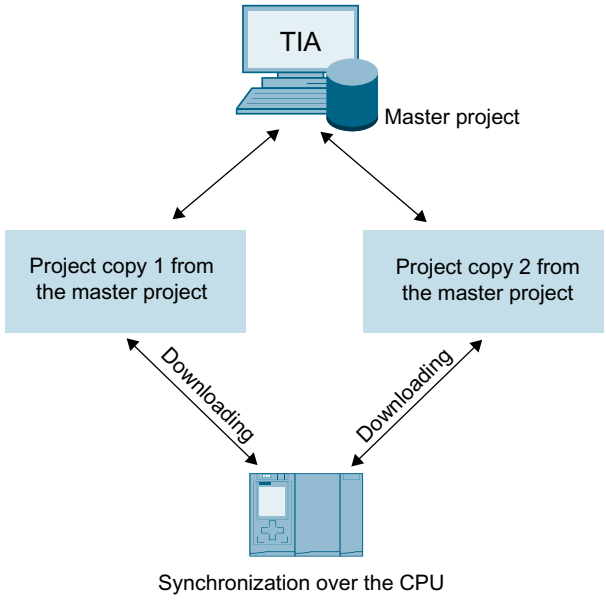
Software units	

You structure your user program with the help of software units. This means you can, for example, combine all the programming elements required for "Motor_1" into one software unit.

Here, too, a "master project" that already contains the completely configured hardware configuration with all required software units serves as the basis.

This project is downloaded in the jointly used CPU and then distributed by means of project copies to up to five participating engineering systems, each of which will then work exclusively on the software units it has been assigned.

Each software unit comes with its own subordinate programming elements such as tag tables, for example, which guarantee independent working in the individual units.

The software units can be edited offline independent of one another and at the same time, plus they can be downloaded to the CPU without the need for synchronization when observing specific requirements.

The software units are integrated back into the master project at the end of the commissioning phase with each ES copying the software units it has edited back into the master project.

See also:

AUTOHOTSPOT

Combined application options:

The functionality of software units can also be used with IPE as well as Multiuser Engineering and Commissioning.

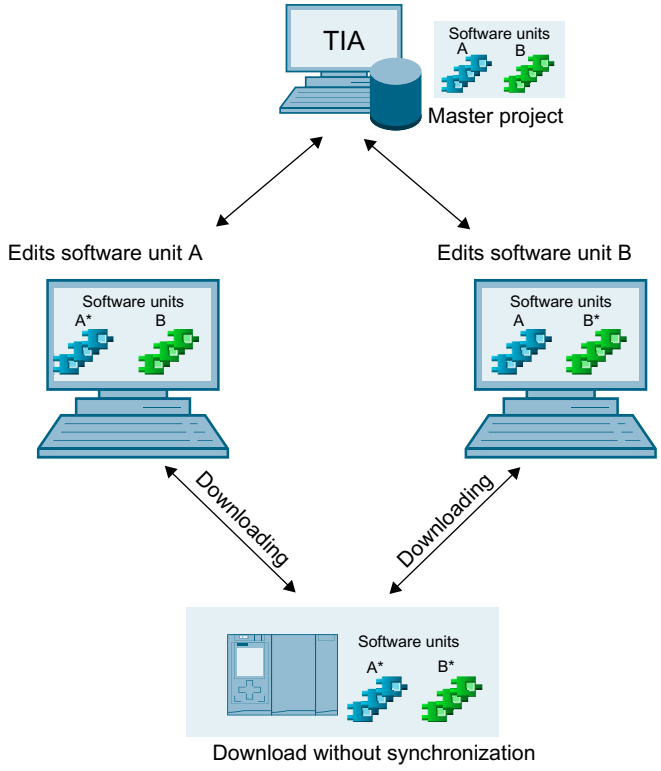

Inter Project Engineering (IPE)	
The functionality of Inter Project Engineering, hereinafter referred to as IPE, is used to exchange the controller data in a source project between different projects with the help of a device proxy. You can then transfer this data to other projects, for visualization in HMI, for example, and use it there for further configuration. Inter Project Engineering gives you a simple way of providing controller data from a PLC programming across multiple projects for an HMI configuration. See also: AUTOHOTSPOT **Combined application options:** The IPE functionality can also be used with PLC synchronization as well as with software units and with Multiuser Engineering and Commissioning.	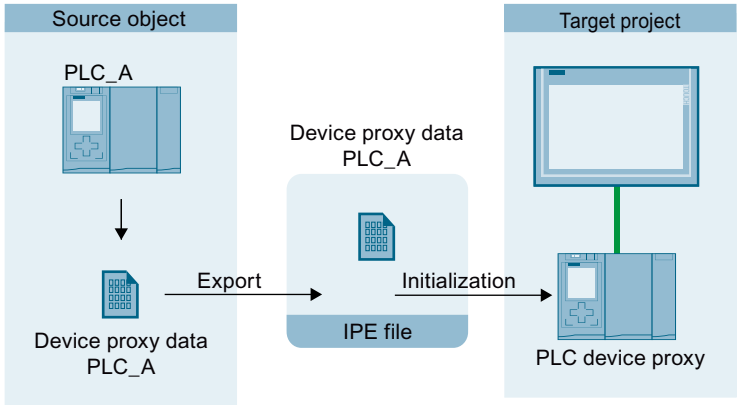

See also

Introduction to Multiuser Engineering (Page 13)

Introduction to Multiuser Commissioning (Page 173)

Basics of PLC synchronization (Page 185)

Using Multiuser Engineering 2

2.1 Introduction to Multiuser Engineering

Introduction

In the TIA Portal , you have the option of working with Multiuser Engineering .

The advantage of this is that you can work with multiple users simultaneously in multiuser projects. This approach significantly reduces the configuring times for your projects. You can commission your plant faster and start production sooner.

In Multiuser Engineering, you can also use other functionalities of Team Engineering , such as working together on a CPU and exchanging data with Inter Project Engineering (IPE).

This is how Multiuser Engineering works in TIA Portal:

1. Project management is located either on a local or external server.
2. Different users work in local sessions, based on the projects managed by the server.
3. Users work independently from each other in local sessions.
4. The changes from the local sessions are transferred to the server project on check-in.
5. Checked-in changes from other editors are displayed and can be conveniently applied.

Note

Working efficiently with Multiuser Engineering

When working in parallel in multiple local sessions, you should correct errors that occur at an early stage so that the distributed work on a shared project stays efficient.

Therefore, it is recommended to first compile pending changes before check-in in the multiuser editor. Several compiling options are available for this. If compiling errors are indicated, you should correct them remove before check-in.

Compiling pending changed before check-in ensures that you are checking in and transferring an error-free project to other editors.

Different options are available for working simultaneously on a multiuser project, as can be seen in the table below:

Application scenarios for Multiuser Engineering	
Scenario 1: Several configuration engineers work *across multiple devices* on different objects. In this scenario, each configuration engineer works on the objects assigned to them on different devices.	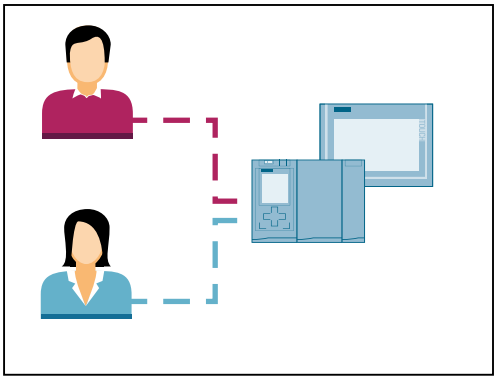
Scenario 2: Several configuration engineers work *device-oriented* on different objects. In this scenario, each configuration engineer works on the objects assigned to them on only one device.	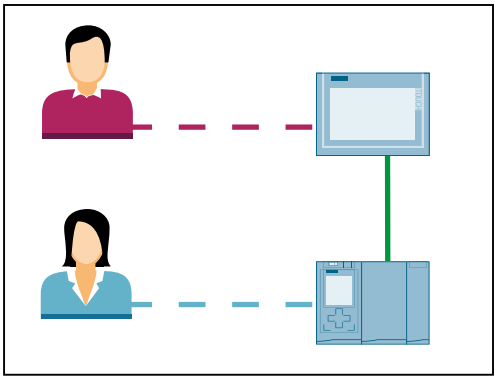
Scenario 3: Several configuration engineers work *technology-based* on different objects. In this scenario, each configuration engineer works on defined technological objects (e.g. Motor_1 or Motor_2) on all existing devices.	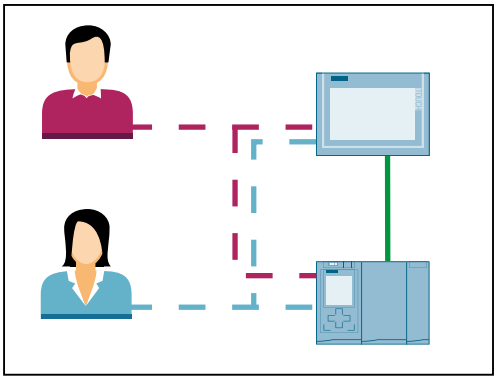

By working on different objects, such as different tags, blocks, PLC tags, interrupts, screens, text and graphic lists, technology objects etc., on one or multiple devices within a multiuser project at the same time, you can make your configuration times more effective and reduce them significantly.

See also

Overview of the mode of operation with Multiuser Engineering (Page 16)
Basics of the multiuser server concept (Page 21)
Requirements for working with Multiuser Engineering (Page 24)
Supported objects for Multiuser Engineering (Page 27)
Introduction to Multiuser Commissioning (Page 173)

2.2 Overview of the mode of operation with Multiuser Engineering

Introduction

If you want to work with multiple users simultaneously in multiuser projects, specific procedures should be observed.

Below you will find an overview of the general mode of operation with Multiuser Engineering.

Application example

You can find an application example for working with Multiuser Engineering at:

https://support.industry.siemens.com/cs/ww/en/view/109740141 (https://support.industry.siemens.com/cs/ww/en/view/109740141)

Requirements for working with Multiuser Engineering

Please observe the following rules for working in multiuser projects:

- Before they can be used as multiuser projects, the projects to be edited should already contain the entire hardware configuration, the required blocks, connections and global settings.
  For more information on this topic, refer to: Requirements for multiuser projects (Page 34)
- Avoid working on the same objects simultaneously by consulting beforehand with the other editors.
- All global settings, such as changes within the device configuration, must be made in the server project view (i.e. in the central server project).
- All objects of a multiuser project can be edited in the server project view. You can also edit objects that do not support the Multiuser Engineering functionality.

Overview of the mode of operation with Multiuser Engineering

To use the Multiuser Engineering functionality, carry out the following steps:

Action	Example
1. Create multiuser project on the server You install a multiuser server and upload a single-user project to the multiuser server that is being used. • The project should be structured so that the requirements for multiuser projects are fulfilled. **Result:** By uploading a project to the multiuser server, you generate a multiuser server project out of the single-user project.	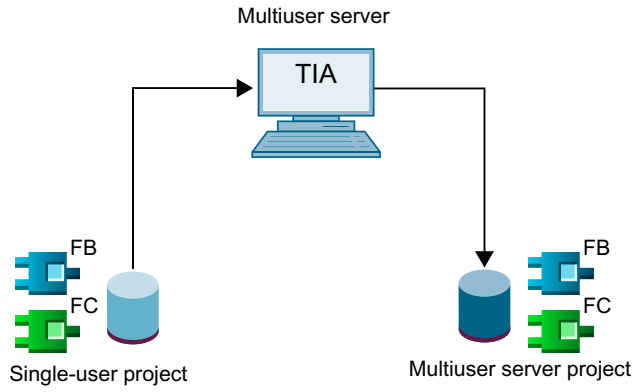
2. Create local session for "Editor 1" You create a "local session" as copy of the server project on your computer. • For each operator, the objects they are editing in their local session is automatically identified with a blue flag. • Various objects can be edited at the same time by several operators in the various local sessions. **Result:** Objects edited in their own local session are identified with a blue flag. The flags are also displayed in the project tree and the editors of the TIA Portal.	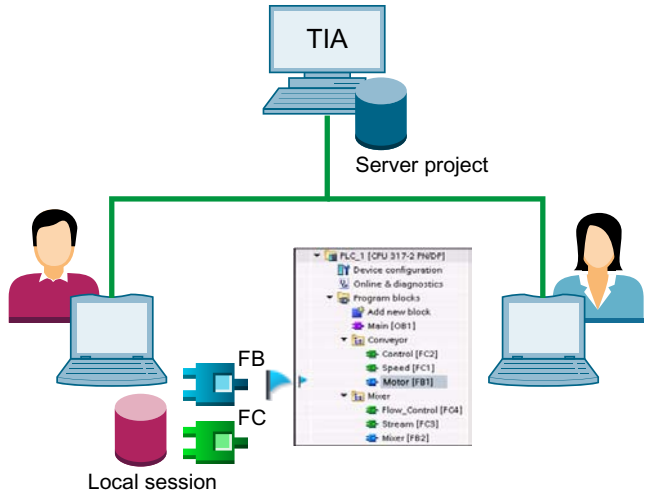

Action	Example
3. Create local session for "Editor 2" You create a second local session so that the second editor can edit their task in their own local session. • For the second operator, too, the objects they are editing in their local session is automatically identified with a blue flag. **Result:** Objects marked in their own local session are identified with a blue flag. Objects selected in another local session are automatically identified with a yellow flag in their own local session. Therefore, each editor also sees which objects they are working on or which other editors are working on.	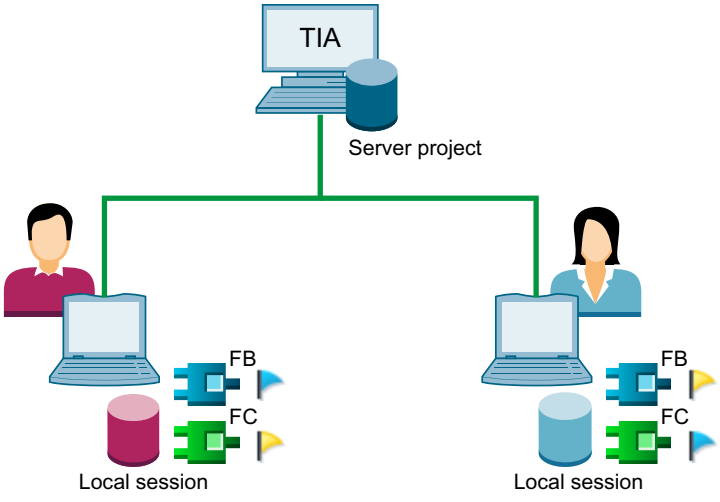
4. Checking in the changes by "Editor 1". As soon as the work in the local session has reached a certain working version, check the changes into the multiuser server project. • The changes to marked objects are applied in the multiuser server project by the check-in. • After the check-in, "Editor 2" is shown in their local session that a object is "outdated". **Result:** The changes of "Editor 1" are transferred to the server project. "Editor 2" can refresh their local session with the new server data, if required.	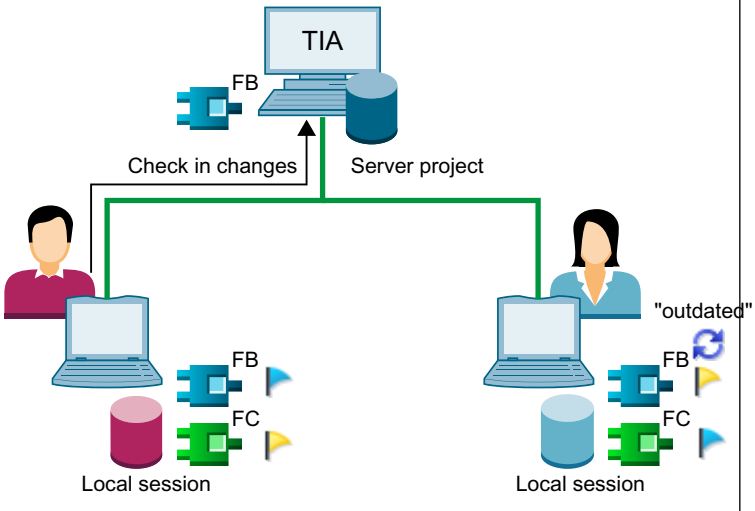

Action	Example
5. "Editor 2" updates their local session "Editor 2" is shown in their local session that an object is available in a newer version in the server project and updates their local session. • The newer version is applied by the refresh and inserted in their own local session. • All unmarked objects are overwritten with the current server version, all marked objects are retained. **Result:** The changes from the server project are transferred to the local session of "Editor 2".	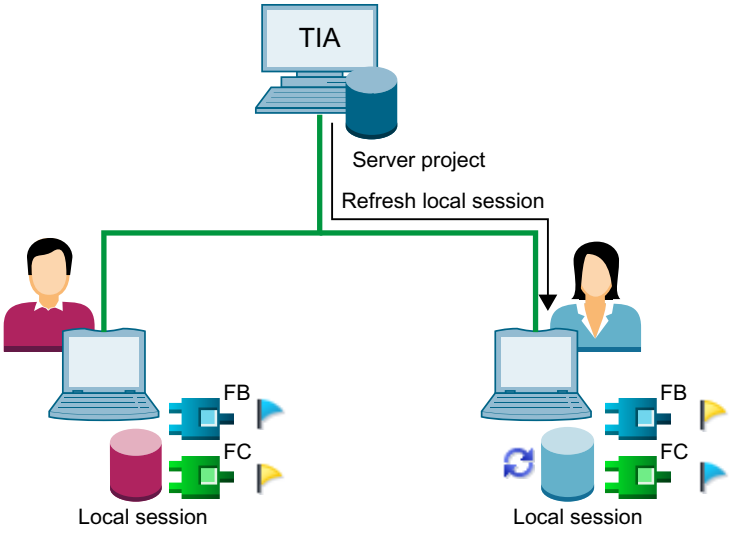
6. "Editor 2" checks in changes to server project "Editor 2" has completed their changes in the local session and checks their blue-marked objects into the multiuser server project. • The changes to marked objects are applied in the multiuser server project by the check-in. • After the check-in, "Editor 1" is shown in their local session that an object is "outdated". **Result:** The changes of "Editor 2" are transferred to the server project. "Editor 1" can refresh their local session with the new server data, if required.	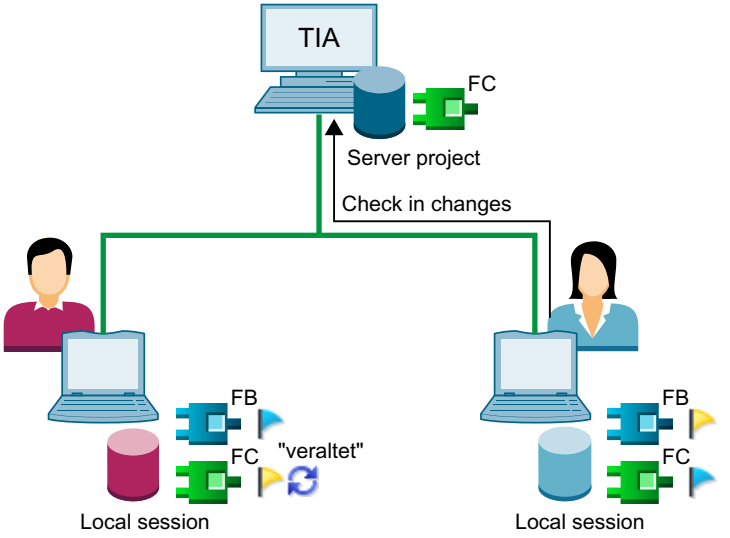

Action	Example
7. Processing is completed The processing of the various configuration tasks in the local sessions has been completed. **Result:** All changes are checked in to server project. The local sessions and the server project are synchronized. There are no more markings in the local sessions. You can save the multiuser server project as single-user project and continue to use it in TIA Portal as usual. **Note**: If required, you can export the local sessions as single-user project and continue to work on them as usual in the TIA Portal.	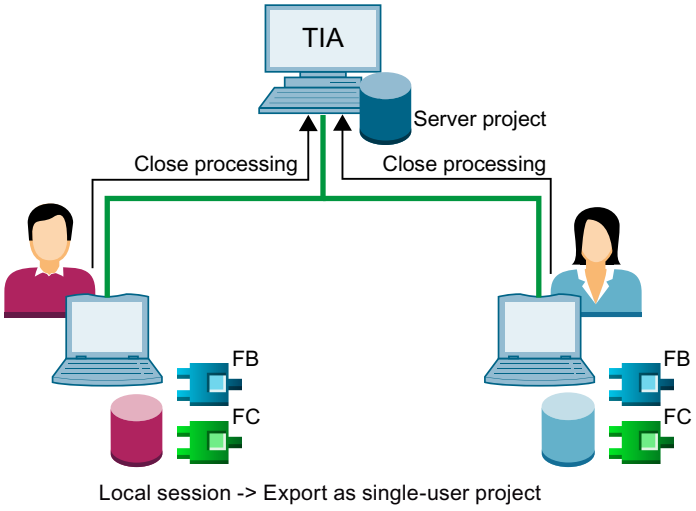

See also

Requirements for working with Multiuser Engineering (Page 24)
Basics of the multiuser server concept (Page 21)
Supported objects for Multiuser Engineering (Page 27)
Requirements for multiuser projects (Page 34)
Exporting a local session as single-user project (Page 141)

2.3 Basics of the multiuser server concept

Introduction

To work with Multiuser Engineering , you need a multiuser server that manages your multiuser projects and the local sessions.

Each version of the TIA Portal contains a version-specific multiuser server with corresponding functionality.

With the help of the multiuser server you can:

- Create new multiuser server projects in which multiple users can work at the same time.
- Display and manage existing server projects.
- Create and manage local sessions.

Compatibility of the multiuser server with the TIA Portal

Different versions of the multiuser server can be installed side-by-side on a device and run at the same time.

Communication between the multiuser server and the TIA Portal is supported for various applications.

For more details, see: Notes on the compatibility of the multiuser server (Page 48)

Requirement

To use the functionality of the multiuser server, you need to install the appropriate software for the multiuser server or use the local multiuser server that is installed by default.

Possible server constellations

Different server constellations for the Multiuser Engineering functionality are available in the TIA Portal.

1. The "local" multiuser server that is automatically installed in the TIA Portal.
2. The so-called "temporary multiuser server" in which the server functionality is implemented on a workstation.
3. The multiuser server that is implemented as dedicated external server for long-term collaboration.

The table below provides an overview of working with the possible server constellations.

Overview of possible server constellations for Multiuser Engineering	
1. Multiuser server as dedicated server • The multiuser server can be installed with the TIA Portal or as a standalone product. • This configuration is usually the one that is used as "office network configuration". • The multiuser server runs on a separate computer. • This server constellation can be reached at all times and supports integrated working "around the clock" by default. • The multiuser server must be installed and configured.	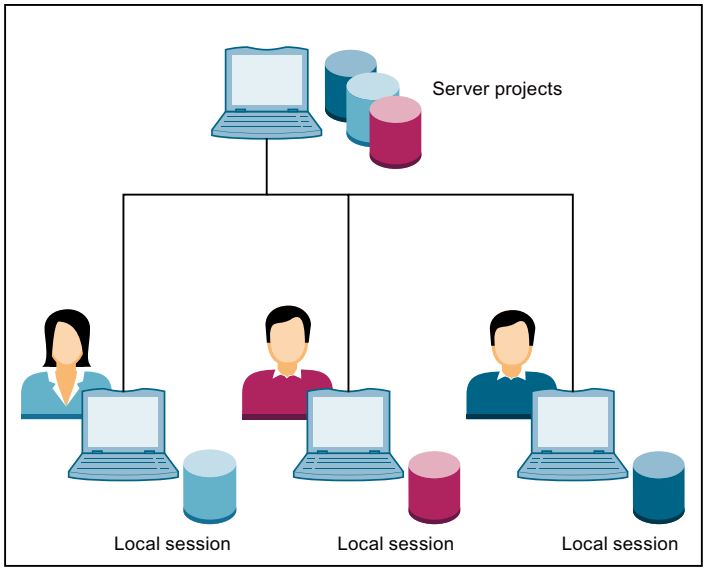
2. Temporary multiuser server: • Editors of the multiuser project can apply the server function to their computer. • This configuration is also a possible "office configuration". • The temporary multiuser server must be installed and configured.	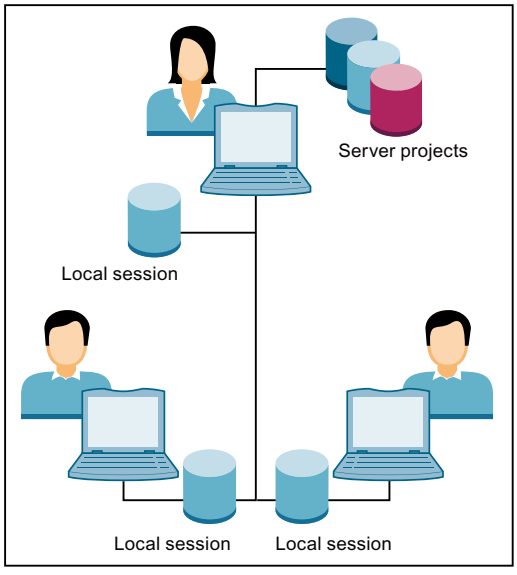

Overview of possible server constellations for Multiuser Engineering	
3. Local multiuser server: • The local multiuser server runs on the computer of the respective editor. • It is available immediately without additional installation and configuration because it is installed with the TIA Portal. • There is no multiuser license request for the local multiuser server. • The local multiuser server cannot be accessed from other computers over a network. • The local multiuser server is suitable for learning and testing the Multiuser Engineering functionality because it does not require server installation or configuration. • The local multiuser server can also be used for separate project management with "rollback" function.	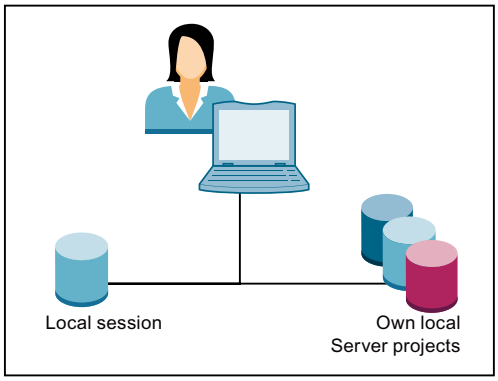

See also

Overview of the mode of operation with Multiuser Engineering (Page 16)

Requirements for working with Multiuser Engineering (Page 24)

Supported objects for Multiuser Engineering (Page 27)

Notes on the compatibility of the multiuser server (Page 48)

2.4 Requirements for working with Multiuser Engineering

System requirements

The same hardware and software requirements as for working with TIA Portal apply to working with Multiuser Engineering .

For information about the regulations in place for the products you have installed, go to "Installation" > "System requirements for installation".

Note

Identical software installation as requirement for working with Multiuser Engineering

To work with Multiuser Engineering without any restrictions, it is important that the same software products of the TIA Portal are installed with identical product versions on all Engineering Systems used.

Note

Access to the multiuser server

Users can only receive access to the multiuser server if they are **not** included in the local administrator group of the multiuser server.

Note

Access to the multiuser server project

Users can only receive access to the multiuser server project if they are included at least as a "Member" in the user management of the multiuser server.

Multiuser server

To work with the full functionality of Multiuser Engineering, you need an installed multiuser server.

You need administrator rights in part for installing, configuring and administering the multiuser server.

You will find information on the compatibility of the multiuser server under:Notes on the compatibility of the multiuser server (Page 48)

The multiuser server is configured and administered with the supplied multiuser tools.

Two alternative procedures are available for this purpose:

- Using the graphical user interface
- Using the multiuser administrative tools and power tools which are executed with the command line.

You can configure and administer the multiuser server using a convenient user interface or with command line commands. Both procedures lead to an identical result.

A detailed description of the procedure is available under "Installing the multiuser server (Page 47)" and under "Configuring and administering the multiuser server (Page 55)".

Multiuser network profiles

You have the option of setting the required performance capability for the network profile you are using in the settings for the TIA Portal.

To do so, select the required profile from the drop-down list in the TIA Portal under "Options" > "Settings" > "Multiuser" > "Multiuser network profiles".

You will find a detailed description of the network profiles under: Setting the multiuser network profiles (Page 118)

Licensing for Multiuser Engineering

For working with Multiuser Engineering , you only need a license for a specific version (Floating License) and an associated License Key when you are working with a multiuser server.

As long as you are only using the local multiuser server with limited functionality, you do not need an additional license for the installed TIA Portal.

You can either include the License Key in the installation of the TIA Portal or transfer it using the Automation License Manager after the installation. Multiuser Engineering checks whether a valid license key exists as soon as you open a local session to edit objects.

If a valid license key is found, the local session is opened and you can work in the local session without restrictions.

If a valid license key is not found, you will receive an error message.

You can temporarily activate a "Trial" license to work in a local session.

Once the "Trial" license has expired, you need a valid multiuser license to work in local sessions.

A license is not required for the following actions in the context of Multiuser Engineering:

- Configuring and starting the multiuser server
- Opening the multiuser project management
- Adding projects to the multiuser server
- Creating local sessions
- Deleting local sessions
- Exporting local sessions as single-user projects

Note

Handling licenses

Please observe the general information on handling licenses in the TIA Portal. This information is available in the "Installation" section of the respective products in the information system.

Archiving and retrieving projects

The archiving and dearchiving of local sessions is not supported by Multiuser Engineering.

Archiving and retrieving of multiuser projects results in invalid local sessions, because the connection to the multiuser server is interrupted by these actions.

Opening a local session as reference project

It is not possible to open a local session as reference project when working with Multiuser Engineering. This prevents a local session from being opened multiple times simultaneously and therefore becoming invalid.

Length of the path and file names in Multiuser Engineering

If the path and file names have more than 259 characters in total, an error message is displayed when you are working with Multiuser Engineering. You are notified that the used names are too long and that the requested action cannot be performed for this reason. Shorten the path names or file names in this case so that the required action can be performed.

See also

Setting the multiuser network profiles (Page 118)

Notes on the compatibility of the multiuser server (Page 48)

2.5 Supported objects for Multiuser Engineering

Introduction

The Multiuser Engineering functionality is available in the TIA Portal for the objects listed below.

Objects that do not support this functionality yet can be edited in the local session but cannot be marked.

This means they are **not** applied to the server project during check-in. The changes to such objects are again overwritten with the contents of the server project and lost after check-in and after a refresh in the local session.

You can edit these objects that cannot be marked in the server project view.

You should make version changes to instructions in the server project view.

In Multiuser Engineering, it is also possible to work with the Team Engineering functionality, such as working on a shared CPU and data exchange via Inter Project Engineering (IPE).

Objects supported by STEP 7 for Multiuser Engineering

The objects listed below for the STEP 7 product are supported:

- Blocks:
  OB, FB, FC, global DBs and instance DBs
- Software-Units:
  Contained blocks, such as OB, FB, FC, global DBs and instance DBs.
  Relations can only be edited in the server project view.
- PLC data types and constants:
  With the exception of system data types generated automatically by the system.
- Tags:
  With the exception of system tags that are created and deleted by the system.
- Interrupts:
  With the exception of system interrupts.
  Program interrupts can only be marked for editing together with the associated FB or DB.
- Interrupt classes:
  With the exception of system interrupt classes
- Text lists:
  With the exception of system text lists
- Watch and force tables
- Library elements of the project libraries
- Technology objects:
  With the exception of technology objects for Standard Motion Control on S7-1500 CPUs.
- Trace functionality:
  With the exception of online configurations

Objects supported by WinCC for Multiuser Engineering

The objects listed below are supported for the WinCC product:

- Screens:
  With the exception of global screen, overview screen, slide-in screens and global filtering of the alarm view.
- Scripts:
  With the exception of the header file "GlobalDefinitions.h".
- Tags:
  With the exception of system tags, recipe tags, archive tags and raw data tags generated by the system.
- Alarms:
  With the exception of system alarms, controller alarms, predefined system alarm classes (including "Safety Warnings" for KTP700F, KTP900F), system class groups and system alarm groups as well as message blocks.
- Text lists, graphic lists and C text list entries

Objects for Multiuser Engineering supported by OPC UA

The objects listed below are supported for OPC UA communication:

- Server interfaces and client interfaces

NOTICE
Editing unsupported objects in the local session Objects that are not supported can be edited in a local session but cannot be marked. Changes to unsupported objects are **not** applied to the server project during check-in. Changes to such objects are lost after check-in and after a refresh in the local session, because they are once again overwritten with the contents of the server project. You can, however, edit such objects that are not supported in the usual way in the server project view.

2.6 Settings for working with Multiuser Engineering

Introduction

For working with Multiuser Engineering you can make the following multiuser-specific settings in the TIA Portal under "Options > Settings" in the "Multiuser" tab:

- General
- Multiuser server connection
- Multiuser network profiles and compression
- Searching in the project
- Storage settings
- Compilation settings

The individual setting options are described below.

General

In the "General" settings, you can select whether the editors opened in the local session are to be hidden when the server project view is opened.

This option is disabled by default.

Multiuser server connection

This dialog is used to add new server connections for a selected multiuser server.

- To create this connection, click the "Add server connection" button.
- In the following dialog box, enter the server name and the associated host and port under "Server Name".
- Select the required transmission protocol for the server connection. "http" is selected by default as transmission protocol.
- Click "Add".

The new multiuser server connection is added and displayed.

See also Introduction to the multiuser server (Page 117).

Multiuser network profiles and compression

You have the option to select from the drop-down list the required network profile that matches the performance capability of the network you are using

The selected multiuser network profile determines the time until a timeout message is generated.

You will find a detailed description of the individual network profiles at Setting the multiuser network profiles (Page 118).

Setting the network profile to "Middle" or "Slow" activates compression for the data transfer. With WLAN or with medium data transfer speeds (Fast Ethernet), an improved transfer time is achieved through compression.

No compression is used for the network profile "Fast".

Note

A change to the network profile only takes effect after a restart of the Multiuser client or the Multiuser server.

Searching in the project

The data for the search must be indexed in order for the global search to also work in local sessions. This option is enabled by default so that the indexing for the search is always executed during check-in.

This is the recommended setting.

If you disable this option, the indexing is not executed during check-in but instead at a different time.

The indexing data is then transferred first as file to the multiuser server. However, a search in the local session is only possible after indexing has been successfully completed. If no indexing was performed during check-in, this can result in somewhat longer waiting times when searching and prolong the check-in times of other editors that have activated this option.

Storage settings

Here you can define where your local sessions and your multiuser server projects will be saved.

- Storage location currently used
  This option is enabled by default.
- Defining a storage location by default
  Select the respective required storage location here by navigating to the required location. Click "Browse" for this purpose. Your settings are applied as default.

Compilation settings

Define here the required compilation settings for your local sessions. The settings made here are applied as defaults in the multiuser editor and can no longer be changed there.

Select the required settings.

- Compile before check-in
  This option is enabled by default and means that a compiling process is executed before the check-in.
- Compilation settings:
  This setting defines the extent to which the software changes are compiled. You have the following options.
  - Marked objects (changes only)
    The changes made to the marked objects for the selected device are compiled together with the device-specific objects.
  - Device software (changes only)
    The changes made to the marked objects are compiled together with the software version on the assigned devices.
    Example: You have marked and changed several objects that belong to more than one device (for example to CPU_1 and to CPU_2). These marked objects are compiled together with the software versions of both CPUs during compiling.
- Disable compilation settings in the multiuser editor
  If you activate the option, you can no longer change the compilation settings in the multiuser editor.
  The settings you made here then apply as default.

See also

Setting the multiuser network profiles (Page 118)

Introduction to the multiuser server (Page 117)

2.7 Improvement of the Multiuser Engineering functionality

Introduction

In TIA Portal , you have the option of working with multiple users simultaneously in multiuser projects within the framework of Multiuser Engineering .

Multiuser Engineering functionality has been further improved as described below.

"Autoselect" function

- As of TIA Portal V15, objects edited with Multiuser Engineering are selected automatically. The "Autoselect" function identifies newly created or changed objects in the local sessions for check-in by setting the corresponding multiuser selections. It is irrelevant here if the changes were made by the user or by the system.
- Deleted objects, too, are selected automatically so that these objects can also be deleted in the server project after check-in.
- In addition to autoselect, you still have the option of selecting objects manually during editing.
- When you work with Multiuser Engineering, there are no functional differences between objects selected automatically and those selected manually.
- The "Autoselect" function recognizes all changes made by the operator to objects that can be selected and supports you in preventing inconsistencies while sharing work with Multiuser Engineering by automatically selecting these objects.
  For more details, see: Marking objects automatically or manually for editing (Page 148)

Working offline with Multiuser Engineering

- As of TIA Portal V15, you can switch between "online mode and offline mode" when working with Multiuser Engineering. This means you can continue working in a local session even without an active server connection to a multiuser server.
- Regardless of whether the server connection is interrupted while working in the local session or if you decided to "Work offline" from the very beginning, you can now continue working – with some restrictions – on your multiuser projects even without an active server connection.
  For more details, see: Working offline with Multiuser Engineering (Page 37)

Improvements in the multiuser editors

- Comment functions for documentation of the type and scope of the changes made have been added to the multiuser editors.
- When you check in, a filter is available for viewing objects with "conflicts".
  For more details, see: Functionality of the multiuser editor (Page 43)

Compatibility of the multiuser server

- Various versions of the multiuser server can be used side-by-side.
- The current multiuser server also supports older TIA Portal projects, with corresponding range of functions.
- In each case, the new server functionality added for a new version is only available if the latest TIA Portal version is installed and the latest project version is used.
- If you are still working with an older TIA Portal or with an older multiuser server, only the functionality included in the installed version is available.
- For more details, see: Notes on the compatibility of the multiuser server (Page 48)

Improvements for Multiuser "Configuration Tool" and "Administration Tool"

- As of multiuser server V15, the external multiuser tools for the configuration and administration of the multiuser server are now available in all languages of the TIA Portal.
- The multiuser server provides extended project history with a restore function. This enables traceability of the project process on the multiuser server. The milestones of a project can be commented on and saved. The project history can be exported for extended evaluation. For more details, see: Using the graphic tool (Page 56)

User administration with Multiuser Engineering in TIA Portal

- If you upload a single-user project that is protected with user administration to a multiuser server, the resulting multiuser project is also protected. The users and user groups contained in the project continue to exist with their passwords and can therefore continue to log into the local sessions of the multiuser project, provided they have the corresponding user rights.
- If you protect a previously unprotected local session using the user administration, this protect is removed once again after checking-in and updating the local session, as the multiuser project on the server is not protected as a result. This also applies when several operators are protecting your local sessions.
- If you protect the multiuser project on the server using the user administration, the local sessions created by this multiuser project are also protected as soon as these are updated.
- If during the processing of your unprotected local session, the multiuser project on the server is protected using the user administration and your user account was not assigned the "Open protect read/write" user right, you cannot check in any changes to your session and cannot update your local session as well.
  You can find additional information on user administration in TIA Portal in the section "Using user administration".

See also

Marking objects automatically or manually for editing (Page 148)

Notes on the compatibility of the multiuser server (Page 48)

Functionality of the multiuser editor (Page 43)

Using the graphic tool (Page 56)

2.8 Requirements for multiuser projects

Introduction

To work with Multiuser Engineering simultaneously and in parallel, it is very important that the multiuser server project that serves as the work basis is clearly structured.

Requirements for multiuser server projects

To create multiuser capable server projects, start with a single-user project that fulfills the following requirements:

- The project should already include the hardware configuration with all required connections.
- Divide the user program into program parts that are independent of each other.
- Create a useful, technology-oriented project structure with folders and groups for the objects to be edited by the individual users.
- When structuring the project, make sure that multiple users can work with the different project directories simultaneously and exclusively.
  Example: One user edits the blocks in group 1 while another user simultaneously edits the blocks in group 2.
- Use a main OB and a central FB or FC for each program part which calls the functions of the subprogram.
- If possible, create a separate PLC tag table for each group.
- You should define in advance all needed project languages that are to be used by the engineering systems involved.
- If possible, create all connections that are to be edited in the multiuser project.
- Define the screen resolution and specify the size of the HMI screens.
- Use global data blocks to save the data for the individual program parts, no bit memories.
- When you load a single-user project created according to these rules to a multiuser server, it becomes a multiuser project.
- Edit the defined objects of the multiuser project simultaneously with multiple users in the local sessions. When you create new objects in the local sessions, make sure that you use different symbolic names.

Example of a server project for multiple users

The figure below shows an example of a server project for multiple users with a clear structure that permits parallel editing of individual objects in the context of Multiuser Engineering .

Each program part contains a "main FB", which calls the specific lower-lever functions for this program part.

The program parts edited by the individual Engineering Systems are:

- Program part 1: "Conveyer"
- Program part 2: "Drill"
- Program part 3: "FurtherProgramPart"

In the "Conveyer" program part, the "ConveyerMainFB" calls the "ConveyerStartupFC" function.

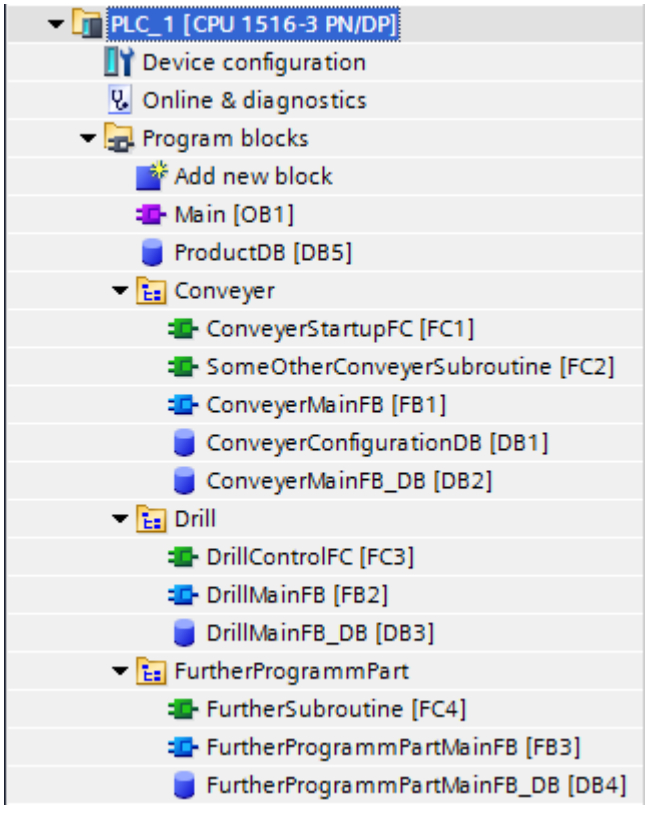

2.9 Upgrading multiuser projects

Introduction

As soon as you try to open a local session that was created with an older version of the TIA Portal with a more recent version of the TIA Portal, you receive a corresponding message. To use older multiuser projects and local sessions with the current version of the TIA Portal, these projects must first be updated.

Upgrading multiuser projects

To use older multiuser projects in the current TIA Portal version, follow these steps:

1. Open the local session with the older version of the TIA Portal with which this local session was created.
2. Make sure that all changes from the local sessions are saved in the associated multiuser server project.
3. Export the multiuser server project with the created version as single-user project.
4. Open this project with the new TIA Portal version and run the suggested project upgrade.
5. Add the updated project as new multiuser server project to the current multiuser server.
6. Create a new local session for the current multiuser server project.

Result

The updated multiuser server project can be edited as usual with Multiuser Engineering.

2.10 Working offline with Multiuser Engineering

Introduction

As of TIA Portal V15, you can also work "offline" with Multiuser Engineering.

This means you can continue working in a local session even without an active server connection to a multiuser server with the restrictions listed below.

Application cases for "Work offline"

The following application cases may occur when working without server connection:

- **A server connection cannot be established when opening the local session.**
  - A dialog with a corresponding message is displayed.
  - When you click "OK", the local session is opened without server connection and you can continue working offline.
- **An existing server connection is interrupted while you are working in a local session.**
  - A dialog with a corresponding message is displayed.
  - When you click "Restore", you can continue working offline.
- **An existing server connection is interrupted while the server project view is open.**
  - A dialog with a corresponding message is displayed which offers you various options.
  - When you click "Restore", an attempt is made to reestablish the server connection.
  - When you click "Discard", the changes you have made in the server project view are discarded and the local session opens.
  - Because there is no server connection when the local session opens, a dialog with a corresponding message is displayed.
  - When you click "Restore", you can continue working offline.
- **You have decided to "work offline" from the very beginning and have selected this option in the TIA Portal under "Project > Multiuser > Work offline".**

Restrictions for "Work offline"

The following restrictions apply for working in offline mode:

- Only your own selections are displayed in your local session.
- Selections made by other users, i.e. those marked with yellow and red flags, and selections for outdated objects cannot be displayed without a server connection.
- The icon for the server status in the multiuser toolbar shows that you are working offline and that there is no server connection.
- You cannot check in your changes to the server project.
- You cannot update your local session.

Switching between "Working online and offline"

You can use the command "Project > Multiuser > Work offline" to switch between online and offline mode when working with Multiuser Engineering .

The default is "Work online". If you click on the check box in front of the "Work offline" command, the system switches to offline mode. In the local session, the symbol for the active server connection changes from a green circle to a light-gray circle. When you clear the "Work offline" check box, the server connection is established once again. Once the server connection has been successfully established, the server status is again indicated with a green circle.

The respective setting is saved together with the local session.

If you have worked in "offline mode" in your local session and are switching to "online mode", the open local session is checked for possible selecting conflicts. The result of the check is displayed with a corresponding message.

If conflicts are found, you are prompted to open the change editor and use the filter setting "Show all conflicts" to display all existing conflicts.

Resolve the existing conflicts and continue working online as usual in your local session.

2.11 Operator controls in Multiuser Engineering

2.11.1 Multiuser icons in the user interface

Introduction

When you are working with Multiuser Engineering and a local session is open, you can see multiuser-specific icons in the TIA Portal user interface.

Multiuser icons in the toolbar

The Multiuser Engineering user interface shows the following icons in the toolbar when a local session is open:

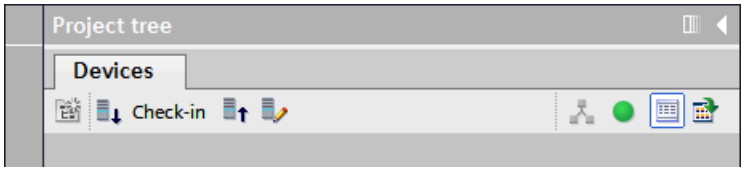

Icon	Meaning
	Check-in When you click this icon, the multiuser editor opens in the change view. In the modification view, all objects marked for check-in are displayed and you can select and check in the desired objects.
	Refresh When you click this icon, the multiuser editor opens in the "Refresh" view. In this view, you can, if you wish, refresh your local session with the new contents of the server project.
	Open/close server project When you click this icon, the multiuser server project associated with your local session is opened for editing. You can then make your changes directly in the server project.
	Commissioning mode is not active This symbol indicates that commissioning mode is **not** activated and that you are working in engineering mode. "Commissioning mode" is enabled and disabled in the Administration Tool . You always set the mode for all local sessions of a server project.
	Commissioning mode is active This symbol indicates that you are working in commissioning mode**.** "Commissioning mode" is enabled and disabled in the Administration Tool . You always set the mode for all local sessions of a server project.
	Server status "connected" This icon indicates that the associated multiuser server is available.

Icon	Meaning
	Server status "locked" This icon indicates that the associated multiuser server is currently being used by a user and is therefore locked. The server is also displayed as locked when a user is working in the server project view. If you place the mouse pointer on this icon, you will receive additional information about the lock.
	Server status "not connected" This icon indicates that the associated multiuser server is not connected.
	"Work offline" server status This symbol indicates that no connection is established to the multiuser server, since you have enabled the "Work offline" function. Clear the check box under "Project > Multiuser > Work offline", to re-establish a connection to the multiuser server.
	This overlay icon shows that a newer version of the server project is available on the multiuser server.

Icons for display of multiuser projects

The following icons are displayed for multiuser projects and local sessions in the user interface:

Icon	Meaning
	Multiuser server This icon identifies a multiuser server.
	Multiuser project This icon identifies a multiuser project associated with a multiuser server.
	Local session This icon identifies a local session that is associated with a multiuser project that is assigned to a multiuser server.

2.11.2 Multiuser icons in the user interface

Introduction

When working with Multiuser Engineering , multiuser-specific icons are displayed in the TIA Portal user interface which make it easier for multiple users to work at the same time. A unique, user-specific visualization is intended to make collaboration within multiuser projects and in local sessions more effective and transparent.

Marking objects

To edit an object using Multiuser Engineering, it must first be "selected". Only objects marked for check-in can be transferred into the server project after editing.

Marking can only be added for "marked objects". Objects that cannot be edited in the local session also cannot be marked.

Such objects can, however, be edited in the server project view.

The icons described below show you which objects were marked by you or other editors.

You can select objects manually by clicking the gray flag in front of the respective object, or automatically. If an object not yet selected manually is opened in an editor in a local session, it is automatically selected with the "Autoselect" function.

Icons for marking in Multiuser Engineering

The following icons are used for marking in Multiuser Engineering :

Icon	Meaning
	Markable object This object can be marked for check-in by clicking it.
	Marked object The object was marked in your own local session for check-in.
	Marked object The object was marked for check-in in a different session or in the server project view.
	Marked object with conflict The object was marked for check-in in multiple local sessions or in a local session and in the server project view and causes a conflict. An object should only be marked for check-in once within the entire multiuser working area (with the local session and within the server project view). An object marked in red can still be checked in. However, the check-in may cause unwanted overwriting in the server project because only the last checked in version is applied in the server project.
	Object is outdated If one of the icons mentioned above Is also identified with this overlay, the object does no longer correspond to the latest version of the server project and should definitely be updated. If the object is checked in without refresh, the editing of the other user is overwritten and thus cancelled. The refresh ensures that all editors of a multiuser project have a consistent version.

Note

Objects without flags

Note that objects without flag can only be edited in the server project view and not in the local session.

Display of multiuser markings in the TIA Portal

When you are working with Multiuser Engineering in the local session or in the server project, the multiuser-specific markings are displayed in the following views of the TIA Portal in all editors:

- In the project tree
- In the detail view

- In the overview window under "Details"
- In the library views

An additional first table column is available in all table displays within the project tree when working with Multiuser Engineering. The first table column is reserved for the multiuser-specific markings described above.

The following figure shows an example of multiple selected objects in the project tree:

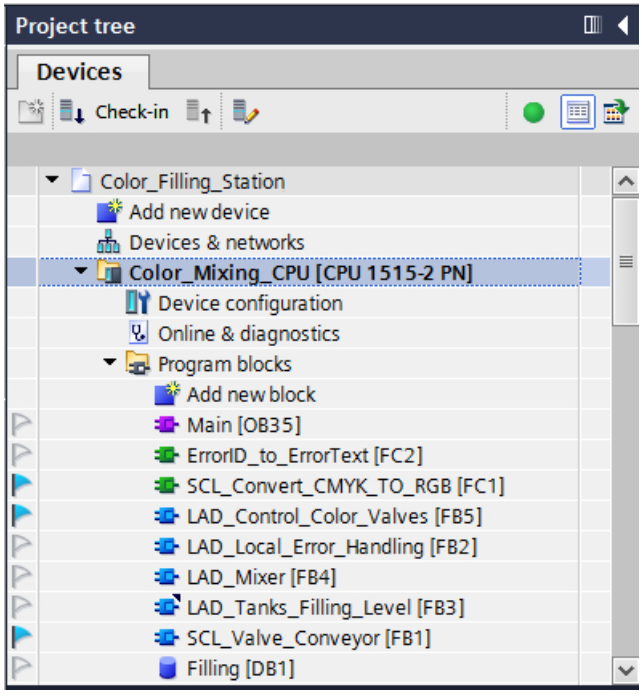

2.11.3 Multiuser banner in the user interface

Introduction

When working with Multiuser Engineering , you are supported by the display of multiuser-specific banner texts in editors.

Banner texts include notes and warnings on the further procedure which make your work easier.

Banner texts can be expanded and collapsed by clicking the yellow circle with the exclamation point.

Example of a multiuser banner text

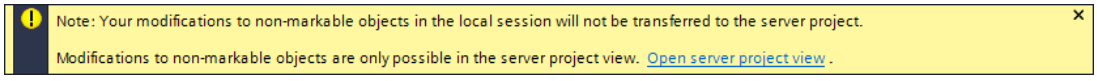

Functionality of the multiuser banner texts

Banner texts are displayed both when working into the local session and when working in the server project view.

As soon as you need additional information to work with Multiuser Engineering , a yellow banner text appears.

The multiuser banner texts have the following function:

- Information about how to edit the opened object:
  - Banner texts point out, for example, that the opened object has not been selected for check-in yet and your changes therefore cannot be applied to the server project.
  - Or they make you aware that this specific object cannot be edited in the local session but only in the server project view.
- Links make for easier execution of the further sequences required.
  - You can use the links, for example, to conveniently mark the open object or open the server project view in which you will continue editing.

As soon as you have executed the required sequences in the individual editors, the banner texts are partially deleted in the editor once again and you can continue working as usual.

2.11.4 Functionality of the multiuser editor

Introduction

You can call the multiuser editor with the "Check-in" and "Refresh local session" buttons in the Multiuser Engineering toolbar.

The multiuser editor has two different views, depending on the button with which the editor was started in the multiuser toolbar:

- "Check-in" view: Opened by clicking the "Check-in" button in the toolbar of the project tree.
- "Refresh" view: Opened by clicking the "Refresh local session" button in the toolbar of the project tree.
  The "Update" view can always be opened even if the local session does not contain any objects that can be updated.

The structure of the multiuser editor is the same in both views, but each view offers different buttons and options.

Note

Displaying in the multiuser editor

If you have opened the multiuser editor and simultaneously make changes in your local session, these changes are only visible in the open editor after a "Refresh". To do this, click the "Refresh" button (circular arrow) in the open editor.

Multiuser editor in the "Check-in" view

The figure below shows an example of the multiuser editor in the "Check-in" view:

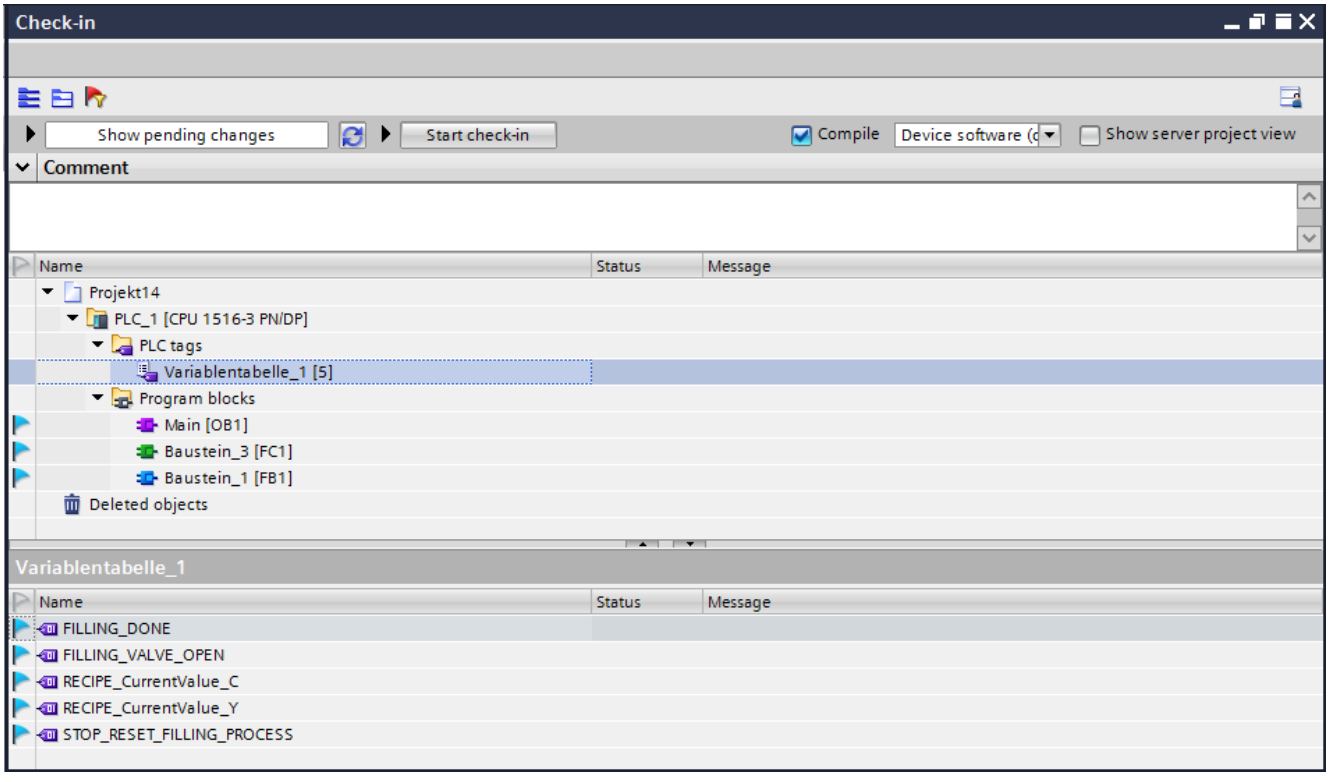

Multiuser editor in the "Refresh" view

The figure below shows an example of the multiuser editor in the "Refresh" view:

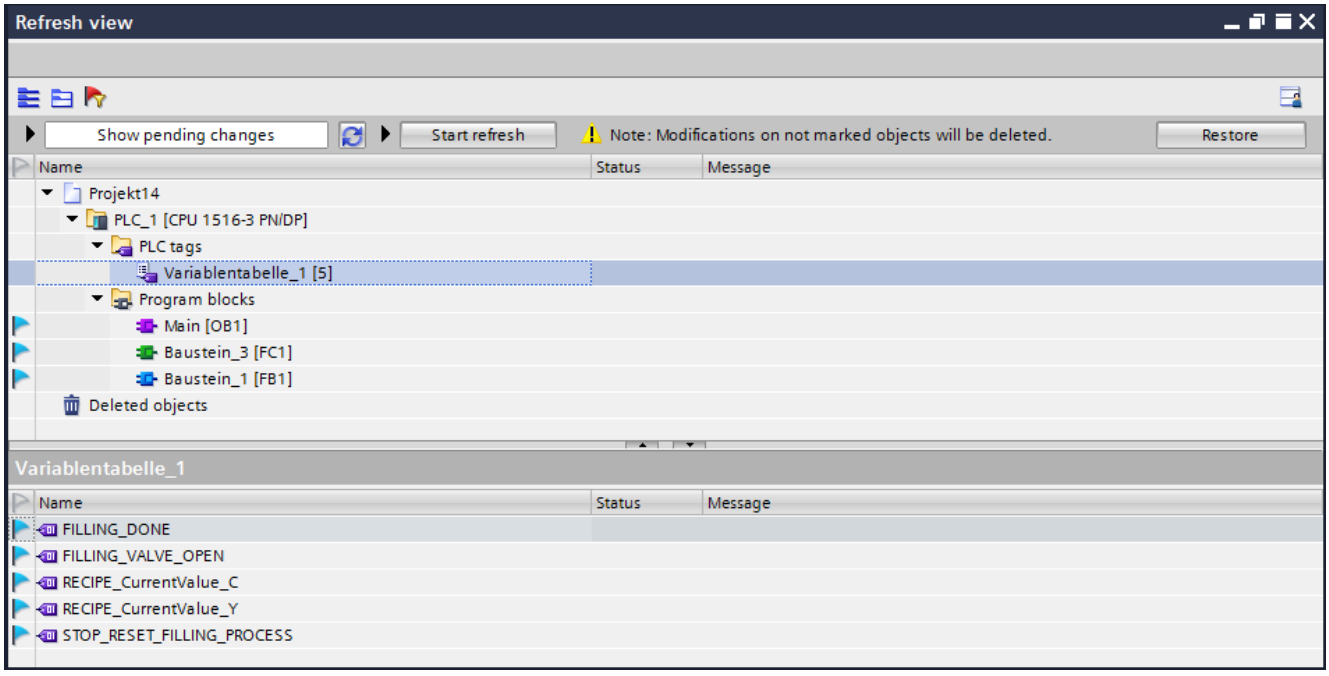

Structure of the multiuser editor

The multiuser editor has the following structure:

1. Toolbar:

Displays following buttons:

- On the right side, the button for saving the layout settings.
- On the left side, the button "Collapse all", "Expand all" and "Show conflicts".
  When you click "Show conflicts", all selections with "Conflict" (red flag) are displayed in the editor.

2. Button area:

Shows the multiuser-specific buttons for the check-in and refresh.

3. Comment line:

The comment line is only available in the "Check-in" view. Here you can enter a comment to document the changes made in the local session.

4. Working area:

Shows the change view prior to check-in or refresh.

- Shows all objects marked in the local session for check-in and refresh as these are also displayed in the project tree.
- Shows the status messages with the results for the check-in and the refresh.

- The "PLC_Tags" folder can be opened.
  Clicking on the tag table in the bottom window in the detail view shows the tags selected for check-in.
- The "Library objects" folder is only displayed when library objects were marked for check-in.
  If the folder is selected, the selected library objects are displayed in the detailed view.
- The "Deleted objects" folder is only displayed if objects were deleted in the local session.
  If the folder is selected, all objects that were deleted in the local session are displayed in the detailed view.

5. Detailed view:

Shows information on lower-level elements (in this case, the selected PLC tags) and status messages.

Note

Procedure for working with the multiuser editor

The procedures for working with the multiuser editor are described in detail in the sections below.

2.12 Installing and uninstalling the Multiuser server

2.12.1 Notes on the installation of the multiuser server

Introduction

Installation of the multiuser server enables you to use the Multiuser Engineering functionality.

You can install the multiuser server as follows:

- Together with the installation of the TIA Portal.
- As standalone installation by means of a separate installation process without TIA Portal.

Note

Administrator rights are required for the server installation.

Installation of the multiuser server with the TIA Portal

The multiuser server is installed together with the following TIA Portal products:

- SIMATIC STEP 7 Basic
- SIMATIC STEP 7 Professional
- SIMATIC WinCC Basic
- SIMATIC WinCC Comfort/Advanced
- SIMATIC WinCC Professional

The multiuser server is usually activated in the product selection by default and is then installed with the TIA Portal. Check the settings during installation of the TIA Portal. You can find the specific default settings for the Multiuser Server during setup when selecting the product in the "Options" directory. By selecting or clearing the options, you can specify if the multiuser server is to be installed along with the TIA Portal or not.

Installation of the multiuser server as standalone installation

The installation of the multiuser server can also be a standalone installation.

A description of automated installation is also available on the product DVD in the directory "Documents\Readme\<language directory>".

Identical TIA Portal versions during installation

When installing different TIA Portal products on various clients, make sure that all participants use the same versions of software products for the installation.

For example, if you have installed STEP 7 Professional, WinCC Advanced and Safety as well as the multiuser server with version Vx.y, all other Team Engineering devices must also have installed this software in identical versions.

The service packs and updates must be installed for all products at the same time.

Installation path

Do not use any UNICODE characters (for example, Chinese characters) in the installation path.

Installation of a newer multiuser server

Make sure that you **install the new multiuser server first** and then uninstall the older version of the multiuser server.

In this way, you ensure that the user authorizations for the server can be taken over for the new server. If all existing multiuser servers are completely deleted before the new installation, the user settings are lost and must be created again.

Antivirus programs

During the installation, read and write access to installed files is necessary. Some antivirus programs block this access. We therefore recommend that you disable antivirus programs during the installation of the multiuser server and enable them again afterwards.

Supported virtualization platforms

You can also install the multiuser server as well as the TIA Portal in a virtual machine.

For this purpose, use one of the following virtualization platforms in the specified version or a newer version:

- VMware vSphere Hypervisor (ESXi) 6.5
- VMware Workstation 12.5.7
- VMware Player 12.5.7
- Microsoft Hyper-V Server 2016

Installation of a visualization platform has the same requirements as the installation of the TIA Portal.

2.12.2 Notes on the compatibility of the multiuser server

Introduction

To work with Multiuser Engineering , you need a multiuser server that manages your multiuser projects and the local sessions.

Different versions of the multiuser server can be installed side-by-side on a device and run at the same time in TIA Portal.

Compatibility of the multiuser server with the TIA Portal

Communication between the multiuser server and the TIA Portal is supported for the following applications.

Multiuser server version	TIA Portal project version
V15.1	V15.1, V15 and V14
V15	V15 and V14
V14	V14

Note

Notes on the compatibility of the multiuser server

The newly added server functionality for V15.1 is only available if TIA Portal V15.1 is also installed.

If you are still working with TIA Portal <V15.1 or a multiuser server <V15.1, only the functionality included in the respective product version is available.

Storage directory of the multiuser server

Each TIA Portal version is delivered with a corresponding multiuser server version. You can define a new storage directory for the new multiuser server or continue using an existing storage directory of an older multiuser server for the new server version V15 in which you have saved the V14 projects, for example.

The following rules apply:

- To use different multiuser servers at the same time and independent of each other, you must configure different ports and storage directories for the different multiuser servers during installation of the new server.
- To continue using an older server directory as storage for the newly installed multiuser server, enter the existing directory during configuration of the new multiuser server. The storage path is checked during installation of the multiuser server, and you receive a corresponding message that this storage directory is already being used for a different multiuser server.
  You then have the following options:
  – You can cancel the installation and enter a new storage directory for the multiuser server.
  – Or you can upgrade the storage directory for the multiuser server.

  However, the upgraded directory becomes invalid for older server versions and can no longer be started with the older multiuser server.
  The upgrade of the storage directory does not impact the TIA Portal projects saved in the directory.
  An internal backup is executed prior to the upgrade so that you can undo any erroneous upgrade.
  If there are no errors during the upgrade, the "Undo" function is not offered.

Note

Storage directory of the multiuser server

The storage directory of the multiuser server is locked during installation of the server. This is intended to prevent changes to a storage directory while it is actively being accessed.

See also

Configuring the multiuser server with the administrative tools (Page 85)

Configuring a multiuser server with the graphic Configuration Tool (Page 57)

Using the graphic tool (Page 56)

Using the command line tools (Page 83)

2.12.3 Installing the multiuser server

Introduction

The multiuser server can be installed as part of the installation of the TIA Portal or standalone. If the multiuser server is installed with the TIA Portal, the version of the multiuser server installed will always correspond to that of the TIA Portal.

A description of automated installation is also available on the product DVD in the directory "Documents\Readme\<language directory>".

Note

New installation before uninstallation

Make sure that you **install the new multiuser server first** and then uninstall the older version of the multiuser server.

In this way, you ensure that the user authorizations for the server can be taken over for the new server. If all existing multiuser servers are completely deleted before the new installation, the user settings are lost and must be created again.

Installation rules

Multiple versions of the multiuser server can be installed on a computer and can use different project versions of the TIA Portal.

Usable project versions:

Multiuser server version	TIA Portal project version
Server V15.1	V15.1, V15 and V14
Server V15	V15 and V14
Server V14	V14

Note, however, that each version of the TIA Portal and the multiuser server only has the functionality contained in this version.

Multiuser Engineering does not upgrade the project versions.

Requirements for installation

Installation of the multiuser server has the same requirements as the installation of the TIA Portal.

Installing the multiuser server with the TIA Portal

Follow these steps for installation:

1. Note the requirements for the installation of the selected software package.
2. Start the setup program of the required product by inserting the installation medium into the respective drive.
3. Select the preferred settings and click the "Install" button.
4. Make sure that the multiuser server is selected in the product selection and observe the notes in the setup dialogs.

Result

The multiuser server is installed on your computer together with the TIA Portal.

Installing the multiuser server as standalone product

Follow these steps for installation:

1. Note the requirements for the installation of the multiuser server.
2. Start setup for the standalone installation of the multiuser server with a double-click on the self-extracting .exe file "TIA_Portal_Multiuser_Server_V<x.y>.exe". You can find these files in the directory "Support" on the corresponding product DVD.
3. Select the preferred settings and click the "Install" button.
4. Follow the instructions in the setup dialogs.

Result

The multiuser server is installed on your computer.

2.12.4 Installing the multiuser server again without loss of data

Introduction

After deleting an installed multiuser server, you can install it again without the loss of data as long as the directory structure with the previously used data still exists and can be accessed for the newly installed multiuser server.

Installation rules

Multiple versions of the multiuser server and the TIA Portal can be installed on a computer, but only one version of the multiuser server can run with the corresponding version of the TIA Portal.

For more details, see: Notes on the compatibility of the multiuser server (Page 48)

Installation of a newer multiuser server

Make sure that you **install the new multiuser server first** and then uninstall the older version of the multiuser server.

In this way, you ensure that the user authorizations for the server can be taken over for the new server. If all existing multiuser servers are completely deleted before the new installation, the user settings are lost and must be created again.

Requirements for installation

Installation of the multiuser server V<x.y> has the same requirements as the installation of the TIA Portal V<x.y>.

Installing a standalone multiuser server again

Follow these steps for installation:

1. Note the requirements for the installation of the selected software package.
2. Start setup for the standalone installation of the multiuser server with a double-click on the self-extracting .exe file "TIA_Portal_Multiuser_Server_V<x.y>.exe" that is available in the "Support" directory of the respective product DVD.
3. Select the preferred settings and click the "Install" button.
4. Follow the instructions in the setup dialogs.

Result

The multiuser server was installed on your computer.

See also

Notes on the compatibility of the multiuser server (Page 48)

2.12.5 Uninstalling the multiuser server

Introduction

You can use the Control Panel to remove the multiuser server.

Note

New installation before uninstallation

Make sure that you **install the new multiuser server first** and then uninstall the older version of the multiuser server.

In this way, you ensure that the user authorizations for the server can be taken over for the new server. If all existing multiuser servers are completely deleted before the new installation, the user settings are lost and must be created again.

Requirements for removal

There is a multiuser server installed on your computer.

Uninstalling the multiuser server

Proceed as follows for removal:

1. Open the Control Panel.
2. Click on "Uninstall a program" in the Control Panel.
   The "Uninstall or change a program" dialog opens.

3. Right-click the entry for the "Multiuser server" in the "Uninstall or change a program" dialog, and click "Uninstall".
   A security prompt appears.
4. Click "Yes" to confirm this prompt.

Result

The multiuser server is removed from your computer.

2.13 Configuring and managing the multiuser server

2.13.1 Options for configuring and administering the multiuser server

Introduction

The following options are also available for configuring and administering the multiuser server:

- Using the graphic tools (Configuration Tool and Administration Tool).
- Using the command line tools (Multiuser Administrative Tools and Multiuser Power Tools).

You can configure and manage multiuser server with both methods.

The graphical tools are easier to operate using the graphical user interface. The command line tools have a somewhat wider range of functions with respect to the error log. You can compress and send the error logs as a zip file.

You have the option of using the procedure you prefer.

Administrator rights are required for configuring the multiuser server irrespective of the selected procedure.

You need administrator rights (for user administration) and user rights for some for administrative tasks.

Note

Available languages

- The tools for configuring and administering the multiuser server are available in English only for Multiuser Server V14.
- As of Multiuser Server V15, the tools are available in the user interface languages supported by TIA Portal.

Configuration and administration with the graphic tools

The graphic tool provides you with a very convenient way of configuring and administering the multiuser server.

You see at a glance all setting options and can activate these by selecting the corresponding options.

You can find the exact procedure in: Using the graphic tool (Page 56)

Configuration and administration with the command line tools

The configuration of the multiuser server is performed with the administrative tools.

The power tools are used for administering the multiuser server.

The power tools for performing administration tasks on the multiuser server are also available to you within the TIA Portal.

You can find the exact procedure in: Using the command line tools (Page 83)

Note

Working with the command line tools

For working with the command line tools, you should have prior experience handling inputs in the command line.

If you lack this experience, you should use the more user-friendly graphic user interface for configuring and administering the multiuser server.

See also

Using the command line tools (Page 83)

Using the graphic tool (Page 56)

2.13.2 Using the graphic tool

2.13.2.1 Introduction to the graphic multiuser tools

Introduction

The multiuser server can be easily configured and administered with the help of the graphic tools.

You use the "ConfigurationTool" to configure the multiuser server.

You use the "AdministrationTool" to administer the multiuser server.

You need administrator rights in some areas to configure the multiuser server.

Note

Available languages

- The tools for configuring and administering the multiuser server are available in English only for Multiuser Server V14.
- As of Multiuser Server V15, the tools are available in the user interface languages supported by TIA Portal.

Requirement

The multiuser server is installed.

Steps for configuring the multiuser server with the "Configuration Tool"

To configure the multiuser server, the following steps are required:

- Open the graphic tool for the multiuser server configuration with administrator rights.
- To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
- Click on the "TIA Portal Multiuser" > Multiuser Server V<x.y> Configuration" entry to open the Configuration Tool .
- Select the language you want from the black highlighted footer.
- Enter the required configuration data and click "Install service".

You can find the exact procedure in: Configuring a multiuser server with the graphic Configuration Tool (Page 57)

Steps for administering the multiuser server with the "Administration Tool"

To administer the multiuser server, the following steps are required:

- Open the graphic tool for the multiuser server administration.
- To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
- Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
- On the "Administration" tab, insert a connection to the desired multiuser server and enter the data for the administration of the server.
- On the "Settings" tab, select the language settings and manage confirmations and certificates.

You can find the exact procedure in: Managing a multiuser server with the graphic Administration Tool (Page 62)

See also

Configuring a multiuser server with the graphic Configuration Tool (Page 57)

Managing a multiuser server with the graphic Administration Tool (Page 62)

2.13.2.2 Configuring a multiuser server with the graphic Configuration Tool

Introduction

You use the graphic "Configuration Tool" to configure the multiuser server.

Requirement

The multiuser server is installed.

Functionality of the Configuration tool

The following functions can be executed with the "Configuration Tool":

"Server status" area

- When the Configuration tool is opened for the first time, the area contains the contents displayed in the graphic. This area is not editable.
- Once the configuration has been successfully completed, the server can be uninstalled, started and stopped in this area.

"Connection settings" area

- Allows you to select the connection data.

"Security" area

- Offers the option to create a new certificate or to use an existing certificate for a selected https connection.

"Memory" area

- Allows you to select the storage location for multiuser projects.
- Allows you to select how many project versions are available as "rollback".

Note

Memory space on the multiuser server

For the project management, you need to have sufficient free space on the multiuser server, depending on the number of saved projects, their sizes and the number of retained project revisions.

Basically, the provided space should be at least five times the expected size of the project.

When a greater number of project revisions is used, the available memory space also needs to be increased accordingly. You set the number of project revisions when configuring the multiuser server.

Generally, you should provide sufficient storage capacity and also take into account or evaluate the messages of the operating system as the memory space decreases.

Note

Working with Multiuser Commissioning

When you work with Multiuser Commissioning, a new revision of the server project is created automatically after each download to the CPU.

To reduce the memory requirements, you should reduce the number of revisions to a daily maximum and only keep the important revisions, e.g. when a project milestone has been reached.

Please also note that the oldest revisions are automatically deleted after the maximum set number of revisions has been reached.

You set the number of project revisions to be saved when configuring the multiuser server in the Configuration Tool. The maximum number is 100 revisions.

User interface of the Configuration tool

The user interface of the Configuration tool contains the following the first time it is opened:

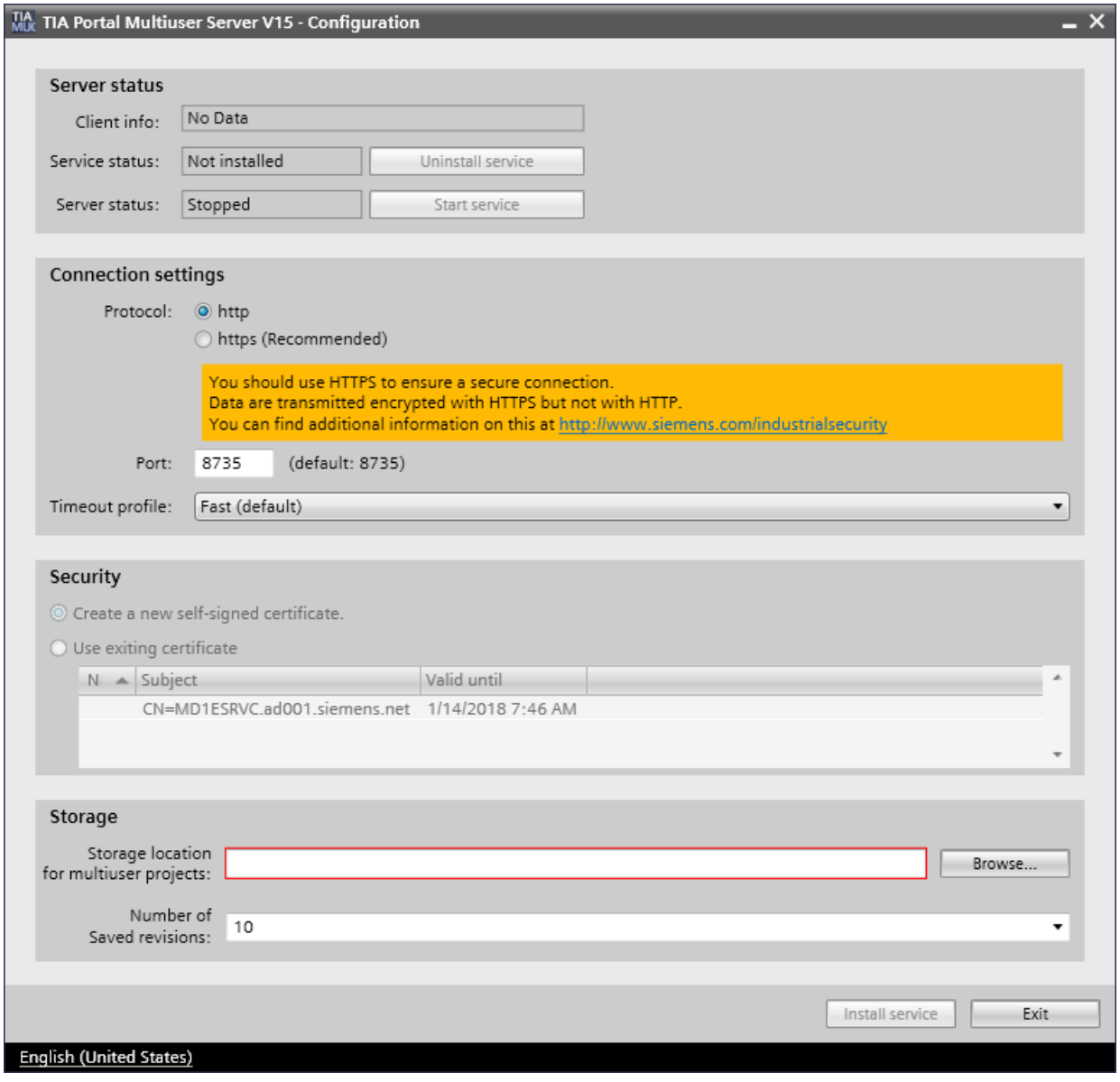

Configuring the multiuser server with the "Configuration Tool"

To configure the multiuser server, follow these steps:

1. Open the graphic tool for the multiuser server configuration with administrator rights.
   To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Configuration" entry to open the Configuration Tool.
   The user interface of the Configuration Tool then opens.
2. In the "Connection settings" area, select the desired protocol.
   Default selection is "http". The text with yellow background is hidden as soon as you click on "https".
   With "https", communication is encrypted and a certificate is requested. This means communication has a higher security level than the unencrypted "http" communication.
3. Enter the desired port for the server connection if you do not want to use the default setting.
4. Select the desired "Timeout profile" for the server from the drop-down list.
   This setting defines the timeout behavior for the server.
   You have the option of selecting between three profiles: "Fast", "Medium" and "Slow"; the default is "Fast".
5. "Security" area If you have selected "https" as the protocol, the "Security" area is activated.
   Select the desired option for the certificate.
   You have the option of creating a new certificate or using an existing certificate on your computer.
   If you activate the first option, a new certificate is created after successful completion of the configuration.
   If you activate the second option, you can click on the desired certificate in the displayed list of certificates to select it.
6. Under "Memory", specify the desired storage location for multiuser projects. You can enter the path directly or navigate to the desired location via "Browse".
7. Select the number of project versions to be stored from the subsequent drop-down list.
   Ten project versions are saved by default.
   You can choose to store between "1" and "Unlimited" project versions.
8. Click "Install service" to complete the configuration.

Result

A message informs you that the configuration of the multiuser server was successful.

In the "Server status" area, you will find the URL for the multiuser server and the fingerprint for the generated certificate under "Client info".

Example: https://XY12345.ww00X.siemens.net:8735/

C35F81B179062BA92A3EAE4DBF18B0DEA9F50D8A

If you access the server from the client side, you should always check the certificate in order to ensure secure communication with the multiuser server. The configuration of the multiuser server is now complete.

Error correction in the Configuration tool

If warnings or errors occur, you will find specific information about these in ServerConfiguration.log.

Proceed as follows:

1. Click on "Open log" in the yellow highlighted area in order to open ServerConfiguration.log.
2. Correct the errors and warnings displayed in the log file.
3. Restart the Configuration tool if it is no longer open.
4. Click "Install service".

Result

A message informs you that the configuration of the multiuser server was successful.

The configuration of the multiuser server is now complete.

Note
Changes to existing inputs

To apply changes to existing inputs in the Configuration Tool, you must click the "Apply" button.

2.13.2.3 Managing a multiuser server with the graphic Administration Tool

Introduction

You use the graphic "Administration Tool" to manage the multiuser server.

Note
Administration Tool

- You need administrator rights to work with the Administration Tool.
- The Administration Tool is also available outside the multiuser server.

Requirement

The multiuser server is installed and configured.

You have already created projects as multiuser projects and have created at least one local session in the TIA Portal.

Note

Multiuser projects and local sessions

If you have not yet created multiuser projects and local sessions on the multiuser server in the TIA Portal, no information regarding these will be displayed when you administer the multiuser server.

User interface of the Administration tool

Open the graphic tool for the multiuser server administration.

- To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
- Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.

The user interface of the Administration tool has the following appearance:

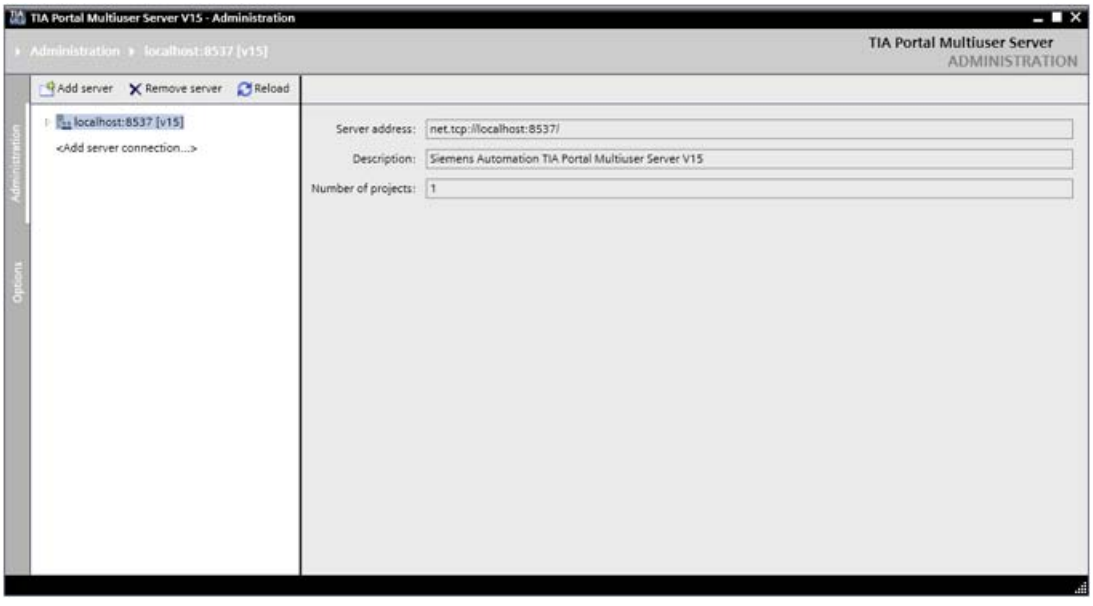

Functionality of the Administration Tool

You can see two tabs on the left-hand side of the tool that let you switch between "Settings" and "Administration".

The following functions can be performed with the "Administration Tool":

Under "Administration", you can find a list of the configured server connections as soon as you have set up a connection to the multiuser server with "Add server connection". On the right side, you receive additional information about the objects selected on the left in a detail view.

Both pages contain their own toolbar for executing the displayed administration functions.

The view can be switched by clicking on the "Administration" button or on the "Settings" button.

On the "Administration" tab you can:

- Add and delete servers.
- Add and delete server connections.
- Update data
- Define user management for the server connections.
- Manage existing multiuser projects and sessions.
  - Specify user roles for projects and sessions: ("Manager", "Contributor", "Member")
  - Projects: Lock, unlock, and delete
  - Sessions: Display and delete
- Enabling and disabling commissioning mode for projects
- Display "History": Project history with all data and display of the saved project revisions
- Export "History": The project history with the selected data is exported as an XML file.
- Manage revisions.
- Add revision information.

In the "Settings" tab you can:

- Select user interface language.
- Select confirmations
- Managing certificates

Note

"Reload" button

Clicking this button updates the display for the selected object in the Administration Tool .

The exact procedures for working with the Administration Tool are described below.

See also

Configuring a multiuser server with the graphic Configuration Tool (Page 57)

2.13.2.4 Adding and deleting a new server connection

Introduction

You can add and delete new multiuser server connections with the graphic "Administration" tool.

These commands can be executed using the commands in the toolbar or from the shortcut menu.

Note

Administration Tool

You need administrator rights to work with the Administration tool.

Requirement

The multiuser server is installed.

The multiuser server has already been configured.

You have already created projects as multiuser projects and created local sessions in the TIA Portal.

Note

Multiuser projects and local sessions

If you have not yet created multiuser projects and local sessions on the multiuser server in the TIA Portal, no information regarding these will be displayed when you administer the multiuser server.

Adding a new server connection

Proceed as follows:

1. Open the graphic tool for the multiuser server administration.
   To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
2. Click "Add server connection".

3. In the dialog that follows, enter the desired connection data for your server connection as shown by way of example in the graphic:

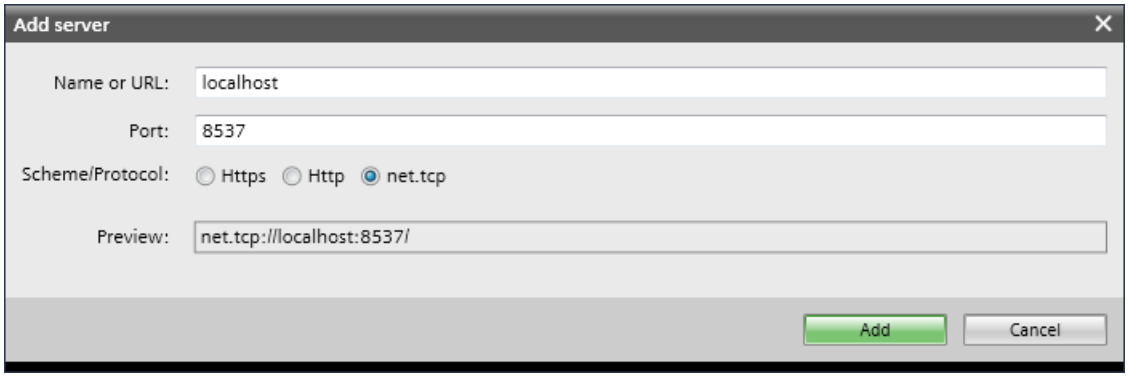

Note

Connection information for the local multiuser server

If you want to connect to the local multiuser server, you must start it first.

The local multiuser server is started automatically when you open a local session in the TIA Portal.

Alternatively, you can restart the local multiuser server by double-clicking on the file "Siemens.Automation.Portal.LocalServer.exe" under "bin" in the installation directory of the TIA Portal.

You can see the connection information for the local multiuser server after opening the displayed command window.

4. Click "Add" to add the server connection.
5. Click on the triangle of the new server connection to activate the connection and view the project list.
6. If you use an "https" connection, you are prompted to confirm the validity of the displayed certificate.
   You have already specified the utilized connection protocol when configuring the multiuser server under "Connection settings".
   Compare the fingerprint of the displayed certificate with the certificate generated during configuration.
   Click "OK" when both fingerprints match.
7. If the multiuser server does not recognize the login data you used, you are prompted to enter alternative connection data for access to the server.
   In the displayed dialog, enter alternative login data that the multiuser server recognizes.

Result

The new server connection is displayed in the left part of the Administration tool.

You see the number of existing multiuser projects on the right.

If you expand the new server connection in the left area, you see a list of all existing projects and other setting options.

Delete an existing server connection

Proceed as follows:

1. Select the desired server connection on the left side of the Administration tool.
2. Click on the "Remove server" button or select the command "Remove" from the shortcut menu.
3. Confirm the next prompt with "Yes".

Result

The selected server connection is deleted.

Accessibility of the multiuser server

If the multiuser server is not accessible, you receive a corresponding error message:

If this happens, proceed as follows:

1. Check the displayed connection data.
2. Ensure that the selected server is started.
3. Delete the invalid server connection by clicking on it and then clicking ""Remove server.
4. Confirm the next prompt with "Yes".
5. Click "Add server connection" and create a new server connection to a started multiuser server with valid connection data.

Result

The new server connection is displayed with valid connection information.

2.13.2.5 Define "User administration" for the multiuser servers and projects

Introduction

With the graphical "Administration Tool", you specify under the project in the "User administration" the rights for working with multiuser server connections, multiuser projects and local sessions. The rights concept is based on the Windows access rights for folders and files.

You will find the "User administration" entry in the Administration Tool below each server connection as well as below each project and each local session. Depending on where you call the "User administration", you specify the corresponding authorizations.

You use the "User administration" to specify explicit personal authorizations for the following roles:

- Manager (full access)
- Contributor (partial access, no deleting)
- Member (read-only authorization)

Note

User administration

Please note that the assignment of users' rights for one project does **not** automatically set up users' rights for the entire multiuser server.

The following functions can be executed for each role using the buttons:

- "Adding": Adds new users with the corresponding authorizations.
- "Remove selected user": Removes the selected user
- "Upgrades the selected user": Moves the selected user up to the next higher authorization group.
- "Downgrades the selected user": Moves the selected user down to the next lower authorization group.

The rights that are specified for the server connections are passed on to the authorizations for projects and local sessions.

Note

Administration Tool

- You need administrator rights to work with the "User administration".
- After changing the permissions, the project must be restarted.

Requirement

The multiuser server is installed.

The multiuser server has already been configured.

You have already created projects as multiuser projects and created local sessions in the TIA Portal.

Note

Multiuser projects and local sessions

If you have not yet created multiuser projects and local sessions on the multiuser server in the TIA Portal, no information regarding these will be displayed when you administer the multiuser server.

Assigning access rights

Proceed as follows:

1. Open the graphic tool for the multiuser server administration.
   - To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
2. Double-click "User administration" below the desired entry on the left side.
   - When you click on "User administration" below a server, you assign the rights for working with the server.
   - When you double-click "User administration" below a project, you assign the rights for working with the project.
3. "User administration" opens and displays the possible settings on the right side. You already see which rights have been assigned for the associated server connection. These rights are inherited for the projects.

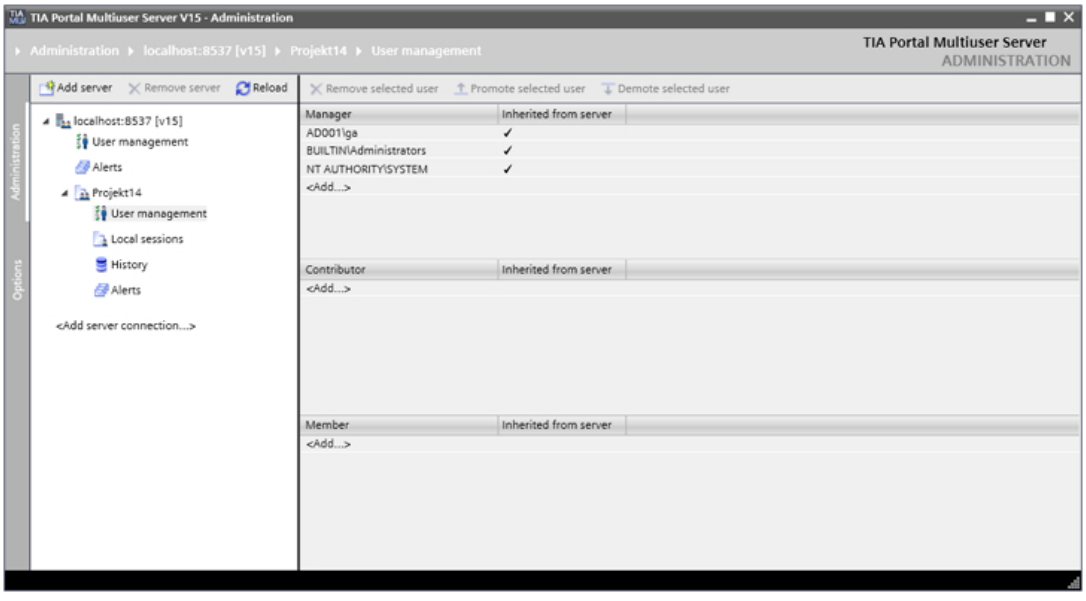

4. Click on "Add" to add new users for the displayed roles.
5. Enter the required data in the subsequent Windows dialog and click "OK".
6. You can use the "Upgrade selected user" and "Downgrade selected user" buttons to move users up and down to other permission groups.
7. You can use the "Remove selected user" button to remove the selected user again.
8. Assign the desired rights and roles via "User administration" for both the server as well as for the lower-level projects and sessions.

2.13.2.6 Managing alerts

Introduction

You can use the graphical "Administration Tool" to view and manage existing alerts on the "Administration" > "Alert" tab. Alerts allow information to be transferred from the server to the administrator.

A number in brackets beside the "Alert" entry means there is that number of alerts.

The entry "Alert" appears both below the server and below the individual projects.

The following alerts are displayed:

- A server alarm as soon as there is less than 3 GB of free memory on the drive containing the directory.
- A project alarm for each local session that is more than 5 revisions old.

Requirement

The multiuser server is installed and configured.

You have already created projects as multiuser projects and have created at least one local session in the TIA Portal.

Note

Multiuser projects and local sessions

If you have not yet created multiuser projects and local sessions on the multiuser server in the TIA Portal, no information regarding these will be displayed when you administer the multiuser server.

Managing certificates

Proceed as follows:

1. Open the graphic tool for the multiuser server administration.
   - To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
2. On the left of the "Administration" tab, click on "Alert". There are alerts for both the multiuser server and the individual projects.
   All alerts for the selected multiuser server are displayed.
3. If you no longer need an alert and wish to delete it, select the alert and click on the "Delete selected alert" button.

Result: The selected alert is deleted from the list.

2.13.2.7 Managing multiuser projects and local sessions

Introduction

You can manage the sessions created for a project with the "Administration Tool".

Under "Sessions", you can see the local sessions for the selected multiuser project.

You can find the "Sessions" entry in the "Administration Tool" below each project.

Requirement

The multiuser server is installed and configured.

You have already created projects as multiuser projects and have created at least one local session in the TIA Portal.

Note

Multiuser projects and local sessions

If you have not yet created multiuser projects and local sessions on the multiuser server in the TIA Portal, no information regarding these will be displayed when you administer the multiuser server.

Display project details and manage projects

Proceed as follows:

1. Call the "Administration Tool".
   - To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
2. Click on the desired project so that the project details are displayed on the right side.
   The following information about the projects is displayed:
   "Project name": Displays the project name
   Status Displays the project status ("locked" or "unlocked").
   "Operating mode": Indicates whether engineering mode or commissioning mode has been selected.
   "Created by / date": Editor who created the project and the creation date for the project.
   "Last edited by / Date": Editor who last amended the project and the change date for the project.
   "Number of local sessions": Number of local sessions that belong to the project.
   "Comment": Display of project comments, if available.

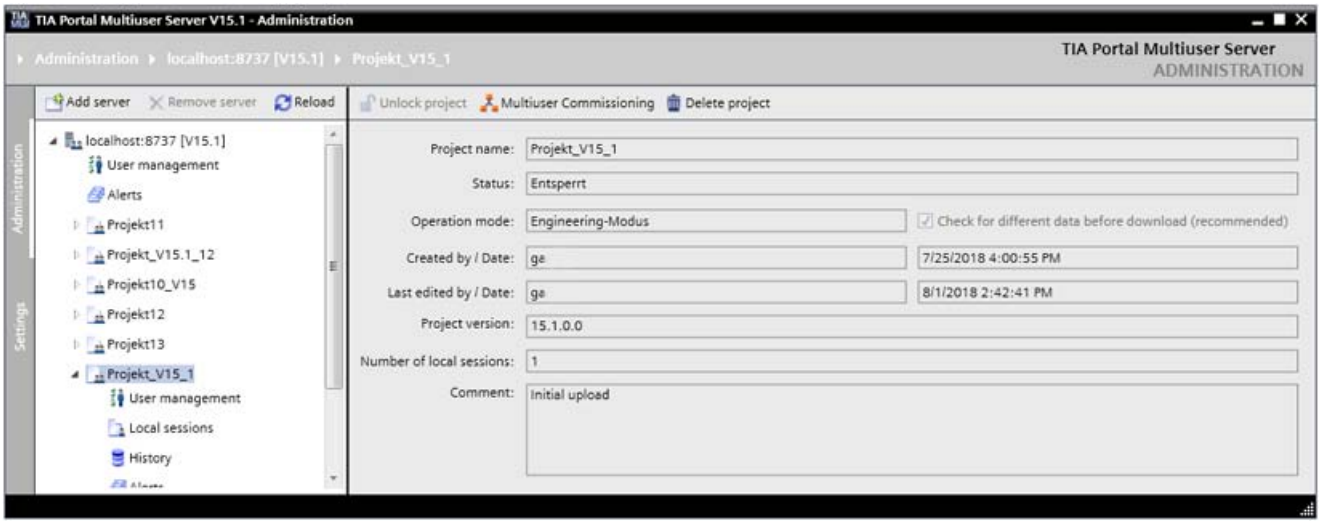

3. To delete a project, select the respective project and then click on the "Delete project" button.
   Projects can only be deleted if there are no local sessions for them.
4. Confirm the next prompt to delete the project.
5. To unlock a locked project, select the respective project and then click on the "Unlock project" button.
6. Confirm the next prompt to unlock the project.
7. Click the "Multiuser Commissioning" button to switch to "Commissioning mode". The button becomes active and has a blue background.
   When commissioning mode is activated, the "Check for different data before download (recommended)" option is simultaneously enabled.
8. Click the "Multiuser Commissioning" button again to exit "Commissioning mode" and switch back to "Engineering mode".

Display and delete local sessions

Proceed as follows:

1. Open the graphic tool for the multiuser server administration.
   - To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "TIA Portal Multiuser Server V<x.y> - Administration" entry to open the Administration Tool.
2. Click on "Local sessions" on the left below the desired multiuser project.
3. The following information on the existing sessions is displayed on the right in the "Administration Tool".
   "ID": Increments the existing sessions.
   "Local path": Displays the set local disk space for the sessions.
   "Computer name": Displays the name of the computer where the session is located.
   "Last used by": Displays the name of the last editor.
   "Project revision": Displays the project version to which each session belongs.

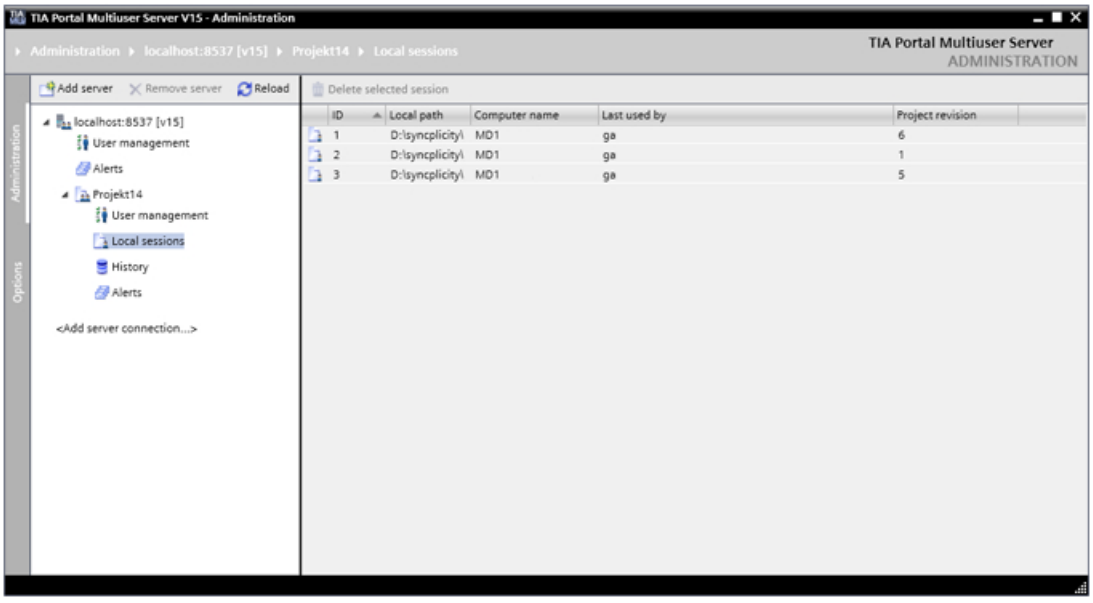

4. Select the desired session and click on the "Delete selected session" button to delete it.
5. In the following dialog, click "OK" to delete the selected session.

Note

Number of project revisions

You set the number of project revisions when configuring the multiuser server.

For more details, see: Configuring a multiuser server with the graphic Configuration Tool (Page 57)

See also

Configuring a multiuser server with the graphic Configuration Tool (Page 57)

2.13.2.8 Showing the history for projects and managing revisions

Introduction

You can view the history of the created projects and manage existing project revisions with the graphical "Administration Tool".

The following functions are available:

- Roll back to the selected revision
- Export selected revision
- Export History
- Save revision
- Show details
- Delete revision data (via shortcut menu)

You can find the "History" entry in the "Administration Tool" below each project.

Requirement

The multiuser server is installed and configured.

You have already created projects as multiuser projects and have created at least one local session in the TIA Portal.

Note

Multiuser projects and local sessions

If you have not yet created multiuser projects and local sessions on the multiuser server in the TIA Portal, no information regarding these will be displayed when you administer the multiuser server.

Displaying history

Proceed as follows:

1. Open the graphic tool for the multiuser server administration.
   - To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
2. Click on "History" on the left below the desired multiuser project.
   The history displays all existing project revisions.
   The following information on the existing revisions is displayed on the right in the "Administration Tool".
   "Availability": Indicates whether the session on the server is still available.
   "Revision number": Displays the revision number.
   "Computer name": Displays the name of the computer on which the revision was created.
   "Created by": Displays the name of the creator.
   "Creation date": Displays the creation date of the revision.
   "Comment": Displays the comment, if any.
   "Notes": Displays the number of notes on the revision, if any. You can display the notes in the details view.

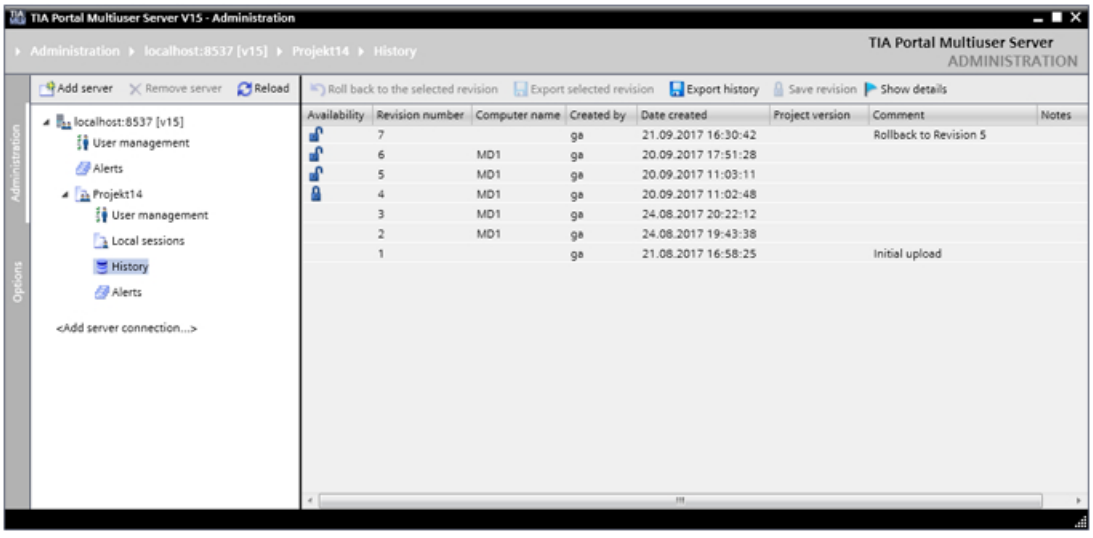

3. You can use the "To roll back selected session" button roll back a selected session to the selected revision.
   A new line is then created showing the new revision with the comment "Rolled back to the revision <number>".
4. To export the revision as a single-user project, click on the "Export selected revision" button. Select the desired storage path in the subsequent dialog and click "OK".
5. To export the project history as an XML file, select the required revisions and click the "Export history" button.
   Enter the required data in the subsequent dialog and click "Save".
6. To save the revision, click on the "Save revision" button.
   In the following dialog, enter a note on the revision to be saved and click on "Save revision".

Deleting revision data

To save memory space, you can delete the data for revisions that are no longer required.

Proceed as follows:

1. Select the desired revision or revisions and click the command "Delete revision data" in the shortcut menu.
2. Confirm the prompt in the dialog that follows with "OK".

Result:

The data of the selected revision is deleted. All relevant information on the change history of the revision is retained.

The revision is no longer displayed under "Availability" and rollback to this revision is no longer possible.

Note

The data for the following revisions cannot be deleted:

- Revisions displayed in the "Availability" column without a symbol. Access to these revisions is no longer possible.
- Revisions with the "Retain" label. You have already labeled these revisions as "secured".
- The last revision of a project.

If you attempt to delete the data for the revisions listed above, you will receive an error message.

Display details about a revision

You can use the button "Display details" to show more information about a selected revised in a detail window within the history for a project versions.

- You can add notes in the "Details" tab.
  To do so, click on the link and enter the required note in the following dialog.
  Click "Add note" to apply the note.
- In the "Changed objects" tab you see all changes made in the selected project.

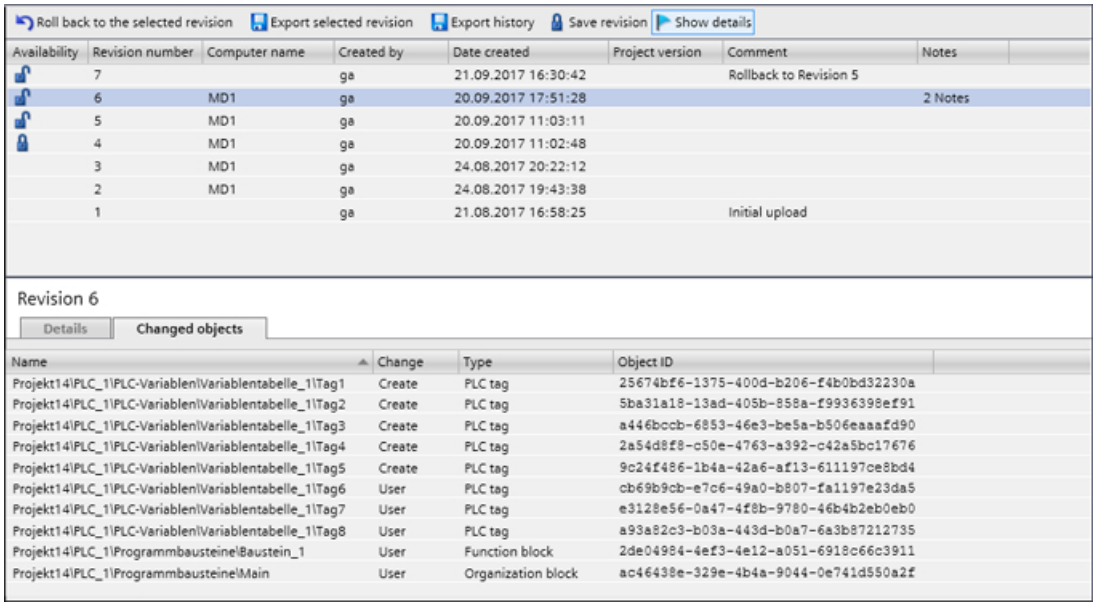

Availability	Revision number	Computer name	Created by	Date created	Project version	Comment	Notes
	7		ga	21.09.2017 16:30:42		Rollback to Revision 5	
	6	MD1	ga	20.09.2017 17:51:28			2 Notes
	5	MD1	ga	20.09.2017 11:03:11			
	4	MD1	ga	20.09.2017 11:02:48			
	3	MD1	ga	24.08.2017 20:22:12			
	2	MD1	ga	24.08.2017 19:43:38			
	1		ga	21.08.2017 16:58:25		Initial upload	

Name	Change	Type	Object ID
Projekt14\PLC_1\PLC-Variablen\Variablentabelle_1\Tag1	Create	PLC tag	25674bf6-1375-400d-b206-f4b0bd32230a
Projekt14\PLC_1\PLC-Variablen\Variablentabelle_1\Tag2	Create	PLC tag	5ba31a18-13ad-405b-858a-f9936398ef91
Projekt14\PLC_1\PLC-Variablen\Variablentabelle_1\Tag3	Create	PLC tag	a446bccb-6853-46e3-be5a-b506eaaafd90
Projekt14\PLC_1\PLC-Variablen\Variablentabelle_1\Tag4	Create	PLC tag	2a54d8f8-c50e-4763-a392-c42a5bc17676
Projekt14\PLC_1\PLC-Variablen\Variablentabelle_1\Tag5	Create	PLC tag	9c24f486-1b4a-42a6-af13-611197ce8bd4
Projekt14\PLC_1\PLC-Variablen\Variablentabelle_1\Tag6	User	PLC tag	cb69b9cb-e7c6-49a0-b807-fa1197e23da5
Projekt14\PLC_1\PLC-Variablen\Variablentabelle_1\Tag7	User	PLC tag	e3128e56-0a47-4f8b-9780-46b4b2eb0eb0
Projekt14\PLC_1\PLC-Variablen\Variablentabelle_1\Tag8	User	PLC tag	a93a82c3-b03a-443d-b0a7-6a3b87212735
Projekt14\PLC_1\Programmbausteine\Baustein_1	User	Function block	2de04984-4ef3-4e12-a051-6918c66c3911
Projekt14\PLC_1\Programmbausteine\Main	User	Organization block	ac46438e-329e-4b4a-9044-0e741d550a2f

Save revision

Revisions can assume the following states:

- Available
  This means that the data from this revision is currently available on the server. However, this data can automatically be deleted as soon as the maximum number of revisions on the server is reached.
  An open lock symbol is displayed in the "Availability" column.
- Saved
  This means that the data from this revision has been saved and the revision will be retained on the server. This revision does not count towards the maximum number of available revisions.
  A closed lock symbol is displayed in the "Availability" column.
- No longer available
  This means that the data from this revision has already been deleted and the revision is no longer available on the server.
  In this case, the revision is still listed but no symbol is displayed in the "Availability" column.

You can secure useful revisions to prevent them from being automatically deleted.

Proceed as follows:

1. Select the revision to be saved that you want to retain on the server and click the "Save revision" button.
2. If required, enter a comment for the revision and confirm the prompt to save the selected revision.

2.13.2.9 Switching back and forth between engineering and commissioning mode

Introduction

You use the graphical "Administration Tool" to switch back and forth between engineering and commissioning mode. You always set the mode for all local sessions of a server project. All local sessions can either be in engineering mode or in commissioning mode. The default setting is engineering mode.

When you switch to commissioning mode, the "Check for different data before download (recommended)" button is also selected automatically. This is a recommended setting for working with Multiuser Commissioning. The check set in this way ensures that no data deviating from the server project is loaded into the CPU. This can be the case, for example, if check-in to a server project from a previous local session took place without a download to the CPU.

If there are deviations between the data available in the server project and the data loaded to the CPU, a dialog is displayed that informs you how to proceed.

Deviating data can be cleaned up in the server project view.

Requirement

The multiuser server is installed and configured.

You have already created projects as multiuser projects and have created at least one local session in the TIA Portal.

Note

Multiuser projects and local sessions

If you have not yet created multiuser projects and local sessions on the multiuser server in the TIA Portal, no information regarding these will be displayed when you administer the multiuser server.

Enabling and disabling commissioning mode

Proceed as follows:

1. Open the graphic tool for the multiuser server administration.
   - To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
2. Under "Administration", you can find a list of the configured server connections as soon as you have set up a connection to the multiuser server with "Add server connection". On the right side, you receive additional information about the objects selected on the left in a detail view.
3. On the left side, select the server and the project for which you want to set up commissioning mode.
4. On the right side, click the "Multiuser Commissioning" button in the header to activate commissioning mode.
   - As soon as the button is active, it is given a blue background and the project selected on the left side receives the commissioning symbol as overlay.
   - With commissioning mode, the "Check for different data before download (recommended)" option is also activated.

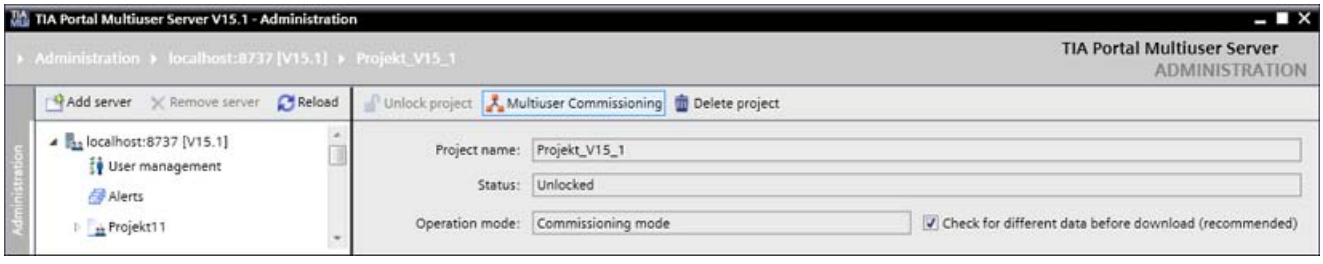

Click the "Multiuser Commissioning" button in the toolbar again to exit "commissioning mode".

- As soon as the button is deactivated, it has no color and no overlay symbol is displayed for the project selected on the left side.
- With commissioning mode, the "Check for different data before download (recommended)" button is also deactivated.

Result

The "commissioning mode" is switched on by clicking the "Multiuser Commissioning" button and switched off by clicking it again.

2.13.2.10 Selecting a user interface language

Introduction

You can use the graphical "Administration Tool" to select the required user interface language on the "Settings" > "General" tab.

All six TIA Portal languages are available.

Requirement

The multiuser server is installed and configured.

You have already created projects as multiuser projects and have created at least one local session in the TIA Portal.

Note

Multiuser projects and local sessions

If you have not yet created multiuser projects and local sessions on the multiuser server in the TIA Portal, no information regarding these will be displayed when you administer the multiuser server.

Changing the language

Proceed as follows:

1. Open the graphic tool for the multiuser server administration.
   - To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
2. Click on the left on the "Settings" > "General" tab.
   In the "Language" section, select the required user interface language from the drop-down list.

Result: The interface texts are displayed in the selected language.

2.13.2.11 Displaying confirmations

Introduction

With the graphical "Administration Tool", you can go to the "Settings" > "General" tab and select the actions for which confirmations are to be enabled when you work with Multiuser Engineering .

If one of the selected actions is subsequently to be implemented, it must always first be confirmed in a prompt dialog.

Confirmations for working in the following areas can be selected:

- User administration
- Deleting a server connection
- Deleting a local session
- Unlocking a project
- Exporting a revision to a folder that is not empty
- Deleting a notification
- Publishing a revision

Requirement

The multiuser server is installed and configured.

You have already created projects as multiuser projects and have created at least one local session in the TIA Portal.

Note

Multiuser projects and local sessions

If you have not yet created multiuser projects and local sessions on the multiuser server in the TIA Portal, no information regarding these will be displayed when you administer the multiuser server.

Selecting a confirmation

Proceed as follows:

1. Open the graphic tool for the multiuser server administration.
   - To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
2. Click on the left on the "Settings" > "General" tab.
3. Go to "Confirmations" and click on the required actions.
   - You can enable confirmations for all entries with the "All" button.
   - You can disable confirmations for all entries with the "None" button.

Result: As soon as one of the selected actions is implemented, it must be confirmed in a prompt dialog before execution.

2.13.2.12 Managing certificates

Introduction

You can use the graphical "Administration Tool" to view and manage existing certificates for the multiuser server on the "Settings" > "Certificate" tab. A certificate is created as soon as you select "https" for the server connection.

Certificates that are no longer required can be deleted if not in use. If you try to delete a certificate that is currently in use in an active server connection, a corresponding message is output.

Requirement

The multiuser server is installed and configured.

You have already created projects as multiuser projects and have created at least one local session in the TIA Portal.

Note

Multiuser projects and local sessions

If you have not yet created multiuser projects and local sessions on the multiuser server in the TIA Portal, no information regarding these will be displayed when you administer the multiuser server.

Managing certificates

Proceed as follows:

1. Open the graphic tool for the multiuser server administration.
   - To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
2. Click on the left on the "Settings" > "Certificates" tab.
   All certificates for the selected multiuser server are displayed with the corresponding data.
3. If you no longer need a certificate and wish to delete it, select the certificate and click on the "Remove selected certificate confirmation" button.

Result: The selected certificate is deleted from the list. If the certificate is currently in use in an active server connection, you receive a message stating that the certificate will not be deleted until the system is restarted.

2.13.3 Using the command line tools

2.13.3.1 Introduction to the multiuser command line tools

Introduction

The multiuser server can be configured and managed with the help of the Multiuser Administrative Tools and the Multiuser Power Tools.

Administrator rights are required for the configuration of the multiuser server with the Administrative Tools.

User rights are required for administrative tasks with the Power Tools.

The Multiuser Power Tools are also available in the TIA Portal for administrative tasks on the multiuser server.

Requirement

The multiuser server is installed.

Steps for the configuration of the multiuser server with the Administrative Tools

The server is installed and managed using a command line command.

To configure the multiuser server, the following steps are required:

- Call the multiuser server Administrative Tools with administrator rights.
- The complete installation is executed with the command:
  `musrv install -r d:\TIA\SRV -p 5080 -b https`
- The project directory (repository) is created automatically.
- The definition of the protocol and the port address is already contained in this command, together with the integrated certificate administration for secure HTTPS communication.
- Windows tools are used to create new certificates and select existing certificates.
- You can find additional commands for configuring the multiuser server here: Commands for configuring the multiuser server (Page 87)

Note

Enabling the firewall port

The specified port is automatically enabled in the firewall for access to the multiuser server.

Managing the multiuser server with the power tools

To manage the multiuser server, the following steps are required after installation of the server:

- To access the multiuser server, a port may have to be enabled in the firewall.
- To manage the multiuser server, the multiuser management or the multiuser server power tools must be called.
- You can find additional commands for managing the multiuser server here: Commands for managing the multiuser server (Page 90)

User and rights management

The user and rights management for the multiuser server is based on the user management for the Windows user accounts.

- Project-specific rights assignment is possible.
- The use of group guidelines enables simple, fast and clear assignment of rights.

Project management

Project management enables a fast overview of existing projects and associated local sessions.

Multiuser projects and local servers can be conveniently managed.

The project management can be automated by scripts (batch files).

Error messages

If an action cannot be executed successfully, a corresponding error message is displayed.

You can find the error descriptions in the corresponding tools by calling the help for the tools:

Calling help in the administrative tools:

- Open the Administrative Tools as follows:
- Copy the command "musrv" to the command line and press "Enter".

Calling help in the power tools:

- Open the Power Tools as follows:
- Copy the command "mupt" to the command line and press "Enter".

See also

Commands for configuring the multiuser server (Page 87)

Commands for managing the multiuser server (Page 90)

2.13.3.2 Configuring the multiuser server with the administrative tools

Introduction

Before you can use the installed external multiuser server as part of Multiuser Engineering , the server must be configured.

When you configure the multiuser server using the Multiuser Administrative Tools , you use the commands described below, which you execute in the command line.

The following actions are automatically executed when you input the "Install" command described below:

- The Windows service for the multiuser service is installed.
- A firewall activation for the specified port is added via a TCP connection.
- Performance monitoring installed to monitor the multiuser server.
- The storage directory, called "Repository", is created if it does not already exist.
- The "binding" is defined, whether you use "https" (recommended) or "http" as transfer protocol.
- The "fingerprint" to confirm a valid certificate is transferred to the server.

The multiuser server can then be used.

Note

You need administrator rights to configure the multiuser server.

Requirement

The multiuser server is installed.

Note

The commands as well as the associated help and information texts within the command line are only available in English.

Procedure

To configure the multiuser server, follow these steps:

1. Open the tool for the multiuser server configuration with administrator rights. Under "bin" in the installation directory of the TIA Portal, double-click on the file "start-musrv.bat" to do this.
2. Copy the following command for server configuration with the required parameters to the command line and press "Enter".
   Command:
   ```
   => musrv install -r <repository-path> -p <port> -b <binding> -h <certificate-hash>
   ```
   Example of the entry:
   ```
   musrv install -r c:\TIA\SERVER -p 5080 -b https -h <certificate-hash>
   ```
3. If the command "musrv install" could not be executed successfully, you receive an error message for which you can view an explanation.
   Once these commands have been executed successfully, the multiuser server is configured.

Configuration example

If you do not enter any parameters, the system prompts you for the required parameters.

Command	Description
Define "Repository-path" Used in the above example: `-r c:\TIA\SERVER`	Enter the name and the path of the directory (Repository) in which the server is to save the files. If you enter a new directory that does not exist yet, the directory is automatically created by the system. **Important note:** This directory and the data storage is retained even after deleting the server. If you reinstall the multiuser server after deleting it and enter an existing directory as "Repository", the data is also available on the new server without any restrictions.
Define "Port" Used in the above example: `-p 5080`	Enter the port you want to use for communication with the multiuser server.
Define "Binding" Used in the above example: `-b https`	Define whether you want to use "https" (recommended) or "http" as transfer protocol. With "https", communication is encrypted and a certificate is requested. This means communication has a higher security level than the unencrypted "http" communication.
"Certificate hash" = fingerprint for the certificate used	For secure server communication, enter the fingerprint for the certificate you created or the certificate used. For more information on creating a certificate, refer to the Help for the respective Windows operating system.
"Run service" = start service Example: `musrv start`	Use it to start the Windows service for the multiuser server.

You can find additional commands for configuring the multiuser server here: Commands for configuring the multiuser server (Page 87)

See also

Commands for configuring the multiuser server (Page 87)

2.13.3.3 Commands for configuring the multiuser server

Introduction

The commands for configuring the multiuser server are available in the Multiuser Administrative Tools .

Observe the basic procedures for configuring the multiuser server: Configuring the multiuser server with the administrative tools (Page 85)

Note

The commands as well as the associated help and information texts within the command line are only available in English.

Commands for configuring the multiuser server

The commands listed below are available:

Command	Description	Possible entries
"Install"	Installs the multiuser server and enables communication with the server. A configuration example is available at: Configuring the multiuser server with the administrative tools (Page 85)	**Parameters:** • -r, --repository: Directory in which the server data is to be stored. • -p, --port: Port number for the server access • -b, --binding: Specifies the protocol for the communication (http or https) • -d, --debug: (optional): Defines whether runtime debug information is displayed by the server. Note: If you use the option "-d", no debug logs are displayed. • -h, --certhash: Fingerprint of the certificate which is to be used for secure data transfer (is only required for https). • -t, --timeout: (optional): Definition of the utilized timeout profile ("fast", "middle" or "slow"). Default: "fast". • -f, --force: (optional): Activates the command without further query • -c, --count: Maximum number (1 to 10) of the saved project versions. **Example**: `musrv install -r <repository-path> -p <port> -b <binding> -h <certificate-hash>`
"Config"	Displays the current configuration of the multiuser server and offers optional dynamic configuration parameters.	**Parameters:** • -t, --timeout: (optional): Definition of the utilized timeout profile ("fast", "middle" or "slow"). • -c, --count: Maximum number (1 to 10) of the saved project versions. • -d, --debug: (optional): Defines whether runtime debug information is displayed by the server. Note: If you use the option "-d", no debug logs are displayed. **Example:** `musrv config -d`
"Start"	Starts the multiuser server.	**Parameters:** • No parameters available **Example:** `musrv start`

Command	Description	Possible entries
"Stop"	Stops the multiuser server.	**Parameters:** • -f, --force: (optional): Activates the command without further query **Example:** `musrv stop`
"Uninstall"	Uninstalls the multiuser server. Uninstallation can also take place with Windows mechanisms.	**Parameters:** • -f, --force: (optional): Activates the command without further query **Example:** `musrv uninstall`

See also

Configuring the multiuser server with the administrative tools (Page 85)

2.13.3.4 Managing the multiuser server with the power tools

Introduction

To manage server projects and local sessions on the multiuser server as part of Team Engineering , use the commands described below that can be executed with the help of the Power Tools in the command line.

The multiuser server can be managed via the Power Tools from any computer on which TIA Portal and Power Tools are installed.

Your entries are stored in a "Session memory" and are then available the next time the Power Tools are opened.

Note

You need user rights to manage the multiuser server.

Requirement

The multiuser server is installed and a port is enabled in the Windows firewall so that the multiuser server is accessible.

Procedure

To manage the multiuser server, follow these steps:

1. Open the Power Tool for the multiuser server administration with administrator rights. Under "bin" in the installation directory of the TIA Portal, double-click on the file "start-mupt.bat" to do this.
2. Copy the command "mupt" to the command line and press "Enter".
   The help for the multiuser power tools is displayed.
3. Enter the required command for server management in the command line and press "Enter".
   The available commands are described under: Commands for managing the multiuser server (Page 90)
4. If the command could not be executed successfully, you receive an error message.

See also

Commands for managing the multiuser server (Page 90)

2.13.3.5 Commands for managing the multiuser server

Introduction

The commands for managing the multiuser server are available in the Multiuser Power Tools .

Note

The commands as well as the associated help and information texts within the command line are only available in English.

"User names" and "Role names"

"User names":

All "User names" that are used must take the following format into account:

- <domain-or-machine-name>\<account>, e.g. ww004\adgrfl0 or MD17E4MC\adgrfl0.
- If the <domain-or-machine-name> is not assigned, Windows will automatically propose a corresponding domain for the account.

"Role names":

The following "Role names" with the corresponding access rights are available:

- Member (only read access)
- Contributor (read & write access)
- Manager (read and write access and role management).

Commands for managing the multiuser server

The commands listed below are available:

Command	Description	Possible entries
"Display project"	Lists all projects on the current multiuser server.	**Parameters:** • -s, --server: Multiuser server URI address **Example:** `%InstallDirectory% > mupt project show -s http://192.168.0.100:8057/` **Help:** • "show-projects -?"
"Delete project"	Deletes specific projects on the multiuser server	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Definition of the project to be deleted • -f --force: (optional): Activates the command without further query **Example:** `%InstallDirectory% > mupt project remove -s http://192.168.0.100:8057/ -p MyProject` **Help:** • "delete-project -?"
"Display Sessions"	Lists all local sessions that are part of the current server project.	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Selection of the project **Example:** `%InstallDirectory% > mupt session show -s http://192.168.0.100:8057/ -p MyProject` **Help:** • "show-sessions -?"
"Delete Session"	Deletes the selected local session on the multiuser server	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Selection of the project • -i, --id: Session identification number • -f, --force: (optional): Activates the command without further query **Example:** `%InstallDirectory% > mupt session remove -s http://192.168.0.100:8057/ -p MyProject -i 15` **Help:** • "delete-session -?"

Command	Description	Possible entries
"Remove project lock"	Removes the lock for the multiuser project	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Selection of the project • -f, --force: (optional): Activates the command without further query **Example:** `%InstallDirectory% > mupt project unlock -s http://192.168.0.100:8057/ -p MyProject` **Help:** • "remove-projectlock -?"
"Show check-in history"	Displays the check-in history for the selected server and the selected local session.	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Selection of the project • -i, --id: Session identification number • -l, --page: Page by page result display • -d, --display: Result display in different form: USAGE: -d [cascade/short/tab] **Example:** `%InstallDirectory% > mupt checkin show -s http://192.168.0.100:8057/ -p MyProject` **Help:** • "show-checkinhistory -?"
"Show server access rights"	Displays the defined user rights for the selected server.	**Parameters:** • -s, --server: Multiuser server URI address **Example**: `%InstallDirectory% > mupt user show -s http://192.168.0.100:8057/` **Help:** • "show-serveraccessrights -?"
"Modify server access rights"	Modifies the access rights for the selected server.	**Parameters:** • -s, --server: Multiuser server URI address • -r, --role: Selecting the role • -u, --user: User name by which to filter. • -f, --force: (optional): Activates the command without further query **Example:** `%InstallDirectory% > mupt user add -u user1 -r Manager -s http://192.168.0.100:8057/` **Help:** • "modify-serveraccessrights -?"

Command	Description	Possible entries
"Remove server access rights"	Removes the access rights for the selected server.	**Parameters:** • -s, --server: Multiuser server URI address • -u, --user: User name by which to filter. • -f, --force: (optional): Activates the command without further query **Example:** `%InstallDirectory% > mupt user remove -u user1 -s http://192.168.0.100:8057/` **Help:** • "modify-serveraccessrights -?"
"Show project access rights"	Displays the defined user rights for the selected project.	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Project name for which the user rights are to be listed. **Example**: `%InstallDirectory% > mupt user show -s http://192.169.0.100:8057/ -p MYProject` **Help:** • "show-projectaccessrights -?"
"Modify project access rights"	Modifies the user rights for the selected project.	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Project name for which the user rights are to be listed. • -r, --role: Selecting the role • -u, --user: User name by which to filter. • -f, --force: (optional): Activates the command without further query **Example**: `%InstallDirectory% > mupt user add -u user2, user3 -r Member -p MyProject -s http://192.168.0.100:8057/` **Help:** • "modify-projectaccessrights -?"

Command	Description	Possible entries
"Remove project access rights"	Removes the access rights of the selected user for the selected project.	**Parameters:** • -s, --server: Multiuser server URI address • -u, --user: User name by which to filter. • -p, --project: Project name for which the user rights are to be listed. • -f, --force: (optional): Activates the command without further query **Example**: `%InstallDirectory% > mupt user remove -u user3 -p MyProject -s http://192.168.0.100:8057/` **Help:** • "remove-serveraccessrights -?"
"Export project to single user"	Exports the multiuser project as single-user project to the specified path.	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Project name for which the user rights are to be listed. • -t, --path: Selection of the storage path **Example**: `%InstallDirectory% > mupt project export -s http://192.168.0.100:8057/ -p MyProject -t D:\MyDirectory\` **Help:** • "export-projectassingleuser -?"

Command	Description	Possible entries
"Export of server project"	Exports an existing multiuser server project to the destination folder in the specified path.	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Project name of the project which is to be exported. If no project name is specified, all projects in the specified server repository are exported. • -t, --path: Selection of the destination folder to which the project is to be exported. The folder must be empty. • -m,--metadata: (Optional) The multi-user server project is exported completely with the entire history and all existing revisions. The default setting is: "metadata" = false. If metadata is not specified, the selected revision (-v, --revision) is exported. • -v, --revision: (Optional) Defines the revision number which is to be exported. When "metadata" is active, this value is ignored. By default, the last revision is always exported. • -f, --force: (optional): Activates the command without further query **Example**: `%InstallDirectory% > mupt project export --server net.tcp://localhost:8735 --project (ProjectName) --path (targetPath) --metadata (true/false) --revision(revisionNumber) --force` **Help:** • "export project -?"
"Import of server project"	Imports an existing multiuser server project from the specified path to the specified multiuser server repository. **Requirement:** Only users with the role "Contributor" or "Manager" are permitted to execute this action. **Rules:** • Only one server project can be imported at a time. • The name of the server project to be imported is also assigned to the destination folder.	**Parameters:** • -s, --server: Multiuser server URI address • -t, --path: Selection of the storage path from which the project is to be imported. **Example**: `%InstallDirectory% > mupt project import --server net.tcp://localhost:8735 --path (sourcePath)` **Help:** • "import project -?"

Command	Description	Possible entries
"Export of server project sessions"	Exports the local sessions of a multiuser server project to the specified path.	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Project name of the server project to which the associated local sessions are to be exported. If no project name is specified, all projects in the specified server repository are exported. • -t, --path: Selection of the destination folder to which the local sessions of the server project are to be exported. The folder must be empty. • -i--id: (Optional) ID of the local session which is to be exported. **Example:** `%InstallDirectory% > mupt session export --server net.tcp://localhost:8735 --path (targetPath)--id (idNum)` **Help:** • "session export -?"
"Import of server project sessions"	Imports the exported local sessions to a selected multiuser server project. **Requirement:** Only users with the role "Contributor" or "Manager" are permitted to execute this action. **Rule**: Only the local sessions known to the associated server project are imported.	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Project name of the server project to which the associated local sessions are to be imported. • -t, --path: Selection of the storage path from which the local sessions are to be imported. • -a, --alias: (Optional) Alias name of the destination server, if available. **Example:** `%InstallDirectory% > mupt session import --server net.tcp://localhost:8735 --path (sourcePath)--alias (serverAlias)`

Command to reset the server to an older revision

The command listed below can only be executed if the server is not locked.

"Project rollback"	Resets the server to an existing older revision.	**Parameters:** • -s, --server: Multiuser server URI address • -p, --project: Project name for which the user rights are to be listed. • -v, --revision: Revision number of the project to which it is to be reset. • -f, --force: (optional): Activates the command without further query **Example**: `%InstallDirectory% > mupt project rollback -s http://192.168.0.100:8057/ -p MyProject -v 132`

2.13.4 Confirm the certificate for a multiuser server

Introduction

When working with Multiuser Engineering , you must first specify the SSL Server certificate in order to establish a secure connection to an installed multiuser server. As soon as you start working in a local session for the first time after installation of the server, you are prompted to confirm the displayed certificate.

When an existing certificate has expired, you are prompted to confirm it again.

The mechanisms provided by Windows are used for managing the SSL server certificates.

If you do not confirm the SSL server certificate, a connection to the multiuser server cannot be established.

Requirement

You have installed TIA Portal and the required multiuser server software.

The multiuser server must be configured and started with the help of the multiuser tools.

You are working in the local session for the first time.

Confirm the certificate for a multiuser server

To confirm the certificate for the multiuser server, follow these steps:

1. When you are working with a connected multiuser server for the first time, for example, when opening a local session, the operating system automatically prompts you to confirm the server certificate for the multiuser server.
2. Check whether the displayed certificate is trustworthy. If so, click "OK" in the dialog that appears to confirm the certificate for the "TIA_Multiuser_Server".

Result

The dialog is closed. The certificate for the multiuser server is considered trustworthy and saved permanently.

2.13.5 Enter access data for the multiuser server

Introduction

When you are working with Multiuser Engineering and have installed a multiuser server, you must log on to the multiuser server if an authentication is required.

In this case the multiuser server prompts you to log on. A dialog is displayed in which you can enter the access data.

You have two options for logging on:

- Logon with your current Windows-user name und password
- Logon with the access data of a different user

Requirement

You have installed TIA Portal and a multiuser server.

The multiuser server must be configured and started with the help of the Power Tools.

Enter access data for the multiuser server

To enter the user data, follow these steps:

1. Select the required option in the dialog by clicking it.
   - For logon with your current login, the user name is already entered and cannot be edited.
   - Alternatively, you can log on with the access data of a different user.
2. To do so, select the second option and enter the respective data for "User name" and "Password".
3. Click "OK".

Result

If authentication was successful, the dialog is closed and the connection to the multiuser server is being established.

2.13.6 Transferring server data to other multiuser servers

Introduction

If you work with Multiuser Engineering and have installed a multiuser server, you can transfer existing server data to another multiuser server without data loss.

The data can be transferred to the same computer or to a another computer.

Requirement

You have installed TIA Portal and at least one multiuser server.

The multiuser server must be configured and started.

Transfer multiuser server data to other multiuser servers

To transfer the server data of a multiuser server to another multiuser server, follow these steps:

1. Stop the active multiuser server.
   - Open the graphic ConfigurationTool for the multiuser server configuration with administrator rights.
   - Open the start menu by clicking the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "Multiuser Server V<x.y> - Configuration" entry. This will open the user interface of the Configuration Tool.
   - In the "Server status" area, click on the "Stop service" button.
2. Uninstall the "old" multiuser server.
   For more details, see: Installing and uninstalling the Multiuser server (Page 47)
3. Copy the entire repository with the existing server data to the new storage location, for example, to another computer.
4. Install the multiuser server on the new computer.
   For more details, see: Installing and uninstalling the Multiuser server (Page 47)
5. Configure the new storage directory for the new multiuser server.
   For more details, see: Configuring a multiuser server with the graphic Configuration Tool (Page 57)
6. Start the new multiuser server. In the Configuration Tool, click on "Start service" in the "Server status" area.

Result

The server data were transferred without data loss to the new multiuser server.

See also

Configuring a multiuser server with the graphic Configuration Tool (Page 57)

Installing and uninstalling the Multiuser server (Page 47)

2.13.7 Reading out error log for multiuser server

Introduction

To check the multiuser server functionality, you can read out the error logs. When server errors occur, the error logs provide you with additional information on the error messages displayed in the user interface.

A separate log file is created by the server for each day. The log files are not deleted automatically.

Reading out the error log of a multiuser server

To read out the error log, follow these steps:

1. Open the directory for the respective multiuser server in your archiving.
2. Select the desired log file under the path <Repository>\logs.
3. Open the requested log file to check the content

Reading out error logs for a multiuser project

To read out the error log, follow these steps:

1. Open the directory for the respective multiuser server in your archiving.
2. Select the desired log file under the path <Repository>\proj\<Project name>\logs.
3. Open the requested log file to check the content.

Note

Reading out the error log

Keep in mind that the error log cannot be read out when you are working in debug mode.

Check and/or change the server configuration in the command line via the administrative tools using the command "musrv config".

Make sure that the option "-d" is not selected.

Reading out Windows event view

The event view is a tool with which detailed information about important events can be displayed on the computer. The event view can be helpful for dealing with problems and errors.

The following security relevant events relating to the multiuser server also receive an entry in the Windows event view:

- If a user wants to the start the multiuser server and the multiuser server does not "know" this user; i.e. the user is not logged in on the server.
- If a user with an incorrect password want to log onto the multiuser server.

To read out the event view, follow these steps:

1. Click "Start" > "Control Panel" > "System and Security" > "Administration". Alternatively, click "Start" and enter "Event view" in the search field.
2. Double-click to open the Event view.
3. Under "Windows protocols" click "Security" to display the logged security relevant events related to the multiuser server.

Result

The logged security relevant events for the multiuser server are shown in a list.

2.13.8 Error log for multiuser server packed as Zip file

Introduction

To check the multiuser server functionality, you can read out the error logs. When server errors occur, the error logs provide you with additional information on the error messages displayed in the user interface.

You have the option of packing all existing error logs into a Zip file to forward these, if necessary, to Customer Support for example.

To do so you must log yourself in as Administrator directly on the server. It is not possible to execute this action via the TIA Portal or via the administrative and power tools.

Packing error logs as Zip file

To create a Zip file for error logs, follow these steps:

1. Log onto the multiuser server as administrator.
2. Execute the command "musrv ziplogs".
3. Alternatively, you can execute the command with the following options:
   - -n, --name: Entering the storage path for the Zip file
   - -f, --force: Overwrites any already existing files without prompt.

Result

The Zip file with all existing error logs for the multiuser server is created.

2.13.9 Information on response of the multiuser server to errors

Introduction

The response of the multiuser server after the occurrence of an error is described below.

If an error occurs, you receive a corresponding error message.

Behavior of the server after the occurrence of an error

The multiuser server exhibits the following behavior after the occurrence of an error:

Project lock:

- Information on which project is locked may get lost after a server error.
- This information is no longer available after a restart.

Errors that occur during "Upload from device" or "Download to device":

- The refresh of the local session cannot be performed. The action is aborted with an error message.
- The check-in cannot be executed. The action is aborted with an error message.
- Any already loaded data are removed again from the server, the server has the same status as before the start of the faulty action.

Note

Manual restart of the server

The server has to be manually restarted after the occurrence of an error.

An automatic start of the server only occurs after the next restart of the computer on which the server is running.

2.13.10 Supported network configurations for the multiuser server

2.13.10.1 Requirements for successful client-server communication

Introduction

Below is an overview of the requirements for successful client-server communication in the context of Multiuser Engineering with:

- Network communication
- Certificate management
- Authentication

Requirements for network communication

The network used must allow HTTPS communication between client and server.

There are two ways of doing this:

- Direct TCP communication between client and multiuser server via the enabled server port.
  - No outbound firewall settings blocks communication on the workstations.
  - No incoming firewall settings blocks communication on the server.
  - Both the multiuser server and all workstations are in the same IP network (without routing).
  - If a router is installed in between, the router should not block communication via the selected server port.
- Indirect communication via an HTTPS proxy server.
  - An HTTPS proxy server is available.
  - The HTTPS proxy server is configured at the client end in the Windows system and enabled for communication.

Requirements for certificate management

To establish secure communication, you should use "HTTPS". With "HTTPS", communication is encrypted and a certificate is requested. This means communication has a higher security level than the unencrypted "http" communication.

To establish HTTPS communication, the multiuser server must be configured in such a way that it uses a valid HTTPS certificate for data encryption.

- The certificate in the Windows certificate store is used on the multiuser server.
- The certificate must be currently valid and registered for use with "http.sys".
- The fingerprint of the certificate must be handed over to the client.

Requirements for authentication

Windows mechanisms are used for the authentication of the communication with the multiuser server.

Multiuser Engineering supports both authentication through domain accounts and authentication through local Windows accounts.

- Authentication with domain account
  - Both the client and the multiuser server are members of the same Windows domain.
  - A valid user account for this domain is used.
  - Both the client and the multiuser server can establish network communication to the domain server.
  - The requirements for using a secure https communication must be met.

- Authentication with local account
  - A local account on the multiuser server must be used for authentication.
  - The requirements for using a secure HTTPS communication must be met.

Note

NTLM authentication on the multiuser server

NTLM is used for authentication on the multiuser server. For security reasons, we recommend you activate NTLM V2, which has been used as standard since Windows 2000, and not NTLM V1.

NTLM V2 offers a logon method more secure than NTLM V1.

Note

In the following sections, you can find more information about supported network configurations for the multiuser server.

See also

Network with dedicated multiuser server (Page 104)

Network with temporary multiuser server (Page 106)

Network in a Windows domain (Page 108)

Network in a Windows domain with Internet access via VPN (Page 110)

Network in a Windows domain with Internet access via proxy (Page 113)

2.13.10.2 Network with dedicated multiuser server

Introduction

Below, you can find information on a server configuration with a dedicated multiuser server in an office network.

The graph shows the possible structure of such a configuration:

Office network with dedicated multiuser server

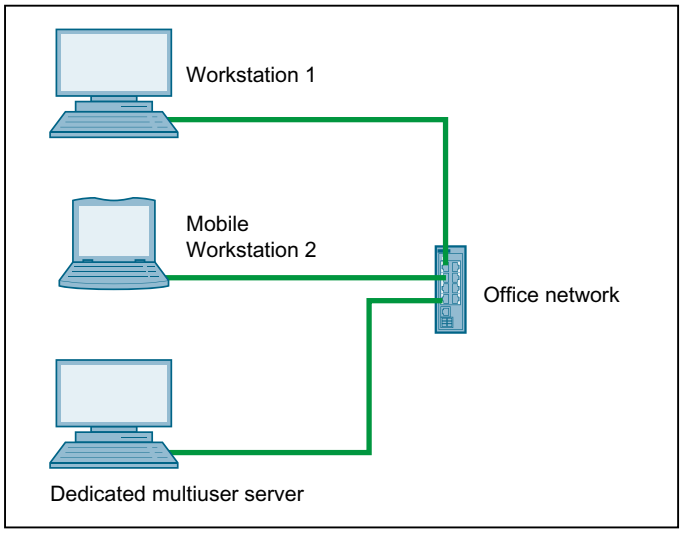

Description of the network configuration with a dedicated multiuser server

The network configuration should meet the following requirements.

Configuration	Description
Network environment	Configuration: • Various Windows workstations on which the TIA Portal is installed. • A dedicated Windows server, which is installed as a multiuser server. • All workstations and servers are located in the same network.
Network	Requirements: • There is a direct TCP connection between the workstations and the multiuser server via the selected port. • No outbound firewall settings blocks communication on the workstations. • No incoming firewall settings blocks communication on the server. • Both the multiuser server and all workstations are in the same IP network (without routing). • If a router is installed in between, the router should not block communication via the selected server port. • Either no proxy server is configured, or there is a local bypass. The "Bypass proxy server for local addresses" option should be selected in the Windows settings for a local area network.
Authentication	User accounts and Windows workgroups: • These must be created locally on the multiuser server. • This must be configured in accordance with the multiuser rights and roles for servers and projects. • Name and password for authentication are queried on all workstations. • Only local accounts of the server system can be used for authentication. Note: The Windows authentication can be used to query name and password. You have to create local accounts, which are identical to the accounts that exist on the multiuser server. This procedure allows the server to automatically authenticate a local account with identical logon information for an existing server account. Restriction: User accounts and groups cannot be managed centrally; local users and groups are required for this. This configuration is therefore unsuitable for enterprise networks.

2.13.10.3 Network with temporary multiuser server

Introduction

Below, you can find information on a server configuration with a temporary multiuser server in an office network.

The graph shows the possible structure of such a configuration:

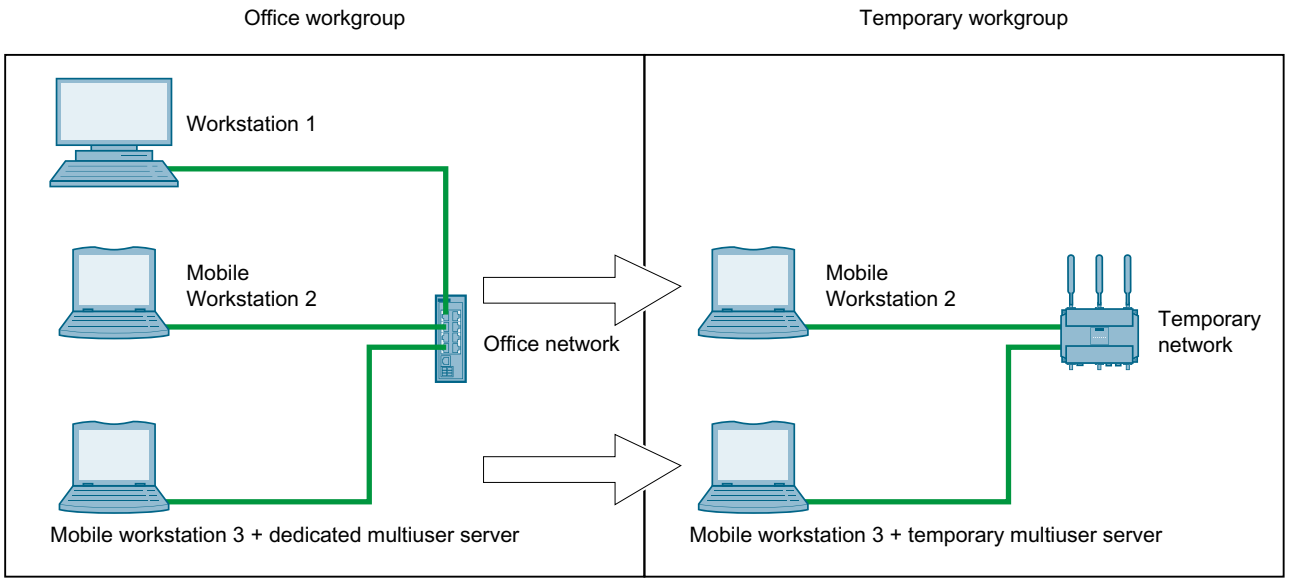
Office workgroup
Temporary workgroup
Workstation 1
Mobile
Workstation 2
Office network
Mobile
Workstation 2
Temporary
network
Mobile workstation 3 + dedicated multiuser server
Mobile workstation 3 + temporary multiuser server

Description of the network configuration with a temporary multiuser server

The network configuration should meet the following requirements.

Configuration	Description
Network environment	Configuration: • Various Windows workstations on which the TIA Portal is installed. • A temporary Windows server, which is installed as a multiuser server at one of the workstations. The office environment may be left, for example, to perform commissioning in a factory. • All workstations and temporary multiuser servers are located in the same network. • With the temporary multiuser server, the office environment can be left and work can still continue.
Network	Requirements: • There is a direct TCP connection between the workstations and the multiuser server via the selected port. • No outbound firewall settings blocks communication on the workstations. • No incoming firewall settings blocks communication on the server. • Both the multiuser server and all workstations are in the same IP network (without routing). • If a router is installed in between, the router should not block communication via the selected server port. • Either no proxy server is configured, or there is a local bypass. The "Bypass proxy server for local addresses" option should be selected in the Windows settings for a local area network.
Authentication	User accounts and Windows workgroups: • These must be created locally on the multiuser server. • This must be configured in accordance with the multiuser rights and roles for servers and projects. • Name and password for authentication are queried on all workstations. • Only local accounts of the server system can be used for authentication. Note: The Windows authentication can be used to query name and password. You have to create local accounts, which are identical to the accounts that exist on the multiuser server. This procedure allows the server to automatically authenticate a local account with identical logon information for an existing server account. Restriction: User accounts and groups cannot be managed centrally; local groups are required for this. This configuration is therefore unsuitable for enterprise networks.

2.13.10.4 Network in a Windows domain

Introduction

Below, you can find information on a server configuration with a dedicated multiuser server in a Windows domain.

The graph shows the possible structure of such a configuration:

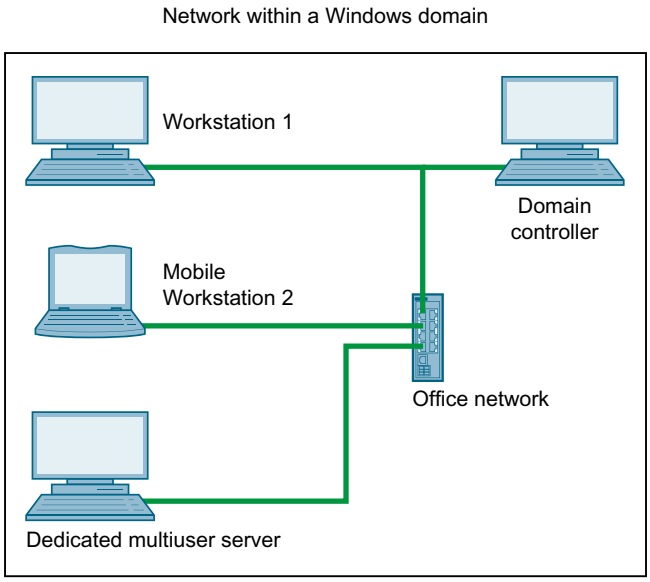
Network within a Windows domain
Workstation 1
Domain controller
Mobile Workstation 2
Office network
Dedicated multiuser server

Description of the network configuration in a Windows domain

The network configuration should meet the following requirements.

Configuration	Description
Network environment	Configuration: • All Windows workstations on which the TIA Portal is installed are members of the same domain. • A dedicated Windows server as a multiuser server is also located in the same domain. • All workstations and multiuser servers use the same LAN (Local Area Network). • All workstations and multiuser servers can connect to the domain controller.
Network	Requirements: • There is a direct TCP connection between the workstations and the multiuser server via the selected port. • No outbound firewall settings blocks communication on the workstations. • No incoming firewall settings blocks communication on the server. • Both the multiuser server and all workstations are in the same IP network (without routing). • If a router is installed in between, the router should not block communication via the selected server port. • Either no proxy server is configured, or there is a local bypass. The "Bypass proxy server for local addresses" option should be selected in the Windows settings for a local area network.
Authentication	User accounts and Windows workgroups: • These must be created on the domain controller. • This must be configured in accordance with the multiuser rights and roles for servers and projects. • Users at workstations on which the TIA Portal is running are automatically authenticated with their domain logon information. Note: You can also log on with other valid logon data on the multiuser server. Restriction: This configuration is not suitable for temporary server configurations, since they also need access to the domain controller.

2.13.10.5 Network in a Windows domain with Internet access via VPN

Introduction

Below, you can find information on a server configuration with a dedicated multiuser server in a Windows domain with VPN (Virtual Private Network).

The graph shows the possible structure of such a configuration:

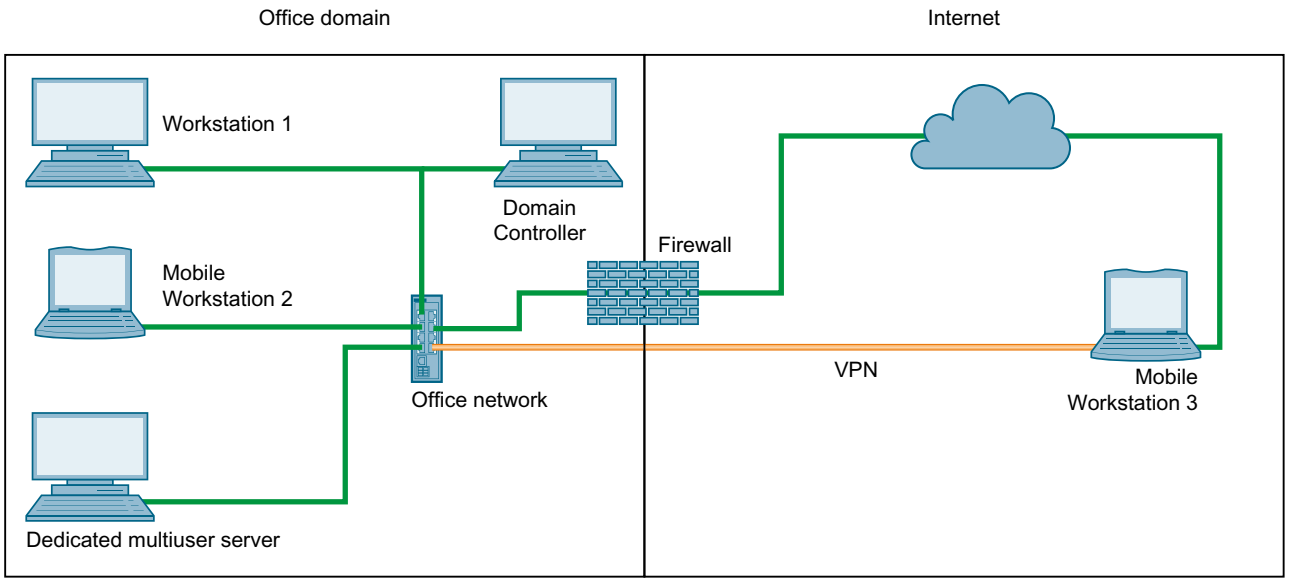
Office domain
Internet
Workstation 1
Domain Controller
Firewall
Mobile Workstation 2
Office network
VPN
Mobile Workstation 3
Dedicated multiuser server

Description of the network configuration in a Windows domain with VPN

The network configuration should meet the following requirements.

Configuration	Description
Network environment	Configuration: • All Windows workstations on which the TIA Portal is installed are members of the same domain. • A dedicated Windows server as a multiuser server is also located in the same domain. • All workstations and multiuser servers use the same LAN (Local Area Network). • Both workstations and the multiuser server have access to the domain controller. • Workstations can also be connected to the LAN via VPN.
Network	Requirements: • There is a direct TCP connection via the selected port between the workstations and the multiuser server. • No outbound firewall settings blocks communication on the workstations. • No incoming firewall settings blocks communication on the server. • Both the multiuser server and all workstations are in the same IP network (without routing). • If a router is installed in between, the router should not block communication via the selected server port. • Either no proxy server is configured, or there is a local bypass. The "Bypass proxy server for local addresses" option should be selected in the Windows settings for a local area network. • For security reasons, we recommend using VPN solutions, such as "Routing and Remote Access Service". • For security reasons, it is strongly recommended to set up a firewall. It is not necessary to open a port in the firewall for multiuser server when using VPN.

Configuration	Description
Authentication	Authentication is performed based on the current configuration selected. **User accounts and Windows workgroups when configured as an office workgroup:** • These must be created locally on the multiuser server. • This must be configured in accordance with the multiuser rights and roles for servers and projects. • Name and password for authentication are queried on all workstations. • Only local accounts of the server system can be used for authentication. **Note**: The Windows authentication can be used to query name and password. You have to create local accounts, which are identical to the accounts that exist on the multiuser server. This procedure allows the server to automatically authenticate a local account with identical logon information for an existing server account. **Restriction:** User accounts and groups cannot be managed centrally; local users and groups are required for this. This configuration is therefore unsuitable for enterprise networks. **User accounts and Windows workgroups with configuration as Windows domain:** • These must be created on the domain controller. • This must be configured in accordance with the multiuser rights and roles for servers and projects. • Users at workstations on which the TIA Portal is running are automatically authenticated with their domain logon information. **Note**: You can also log on with other valid logon data on the multiuser server. **Restriction:** This configuration is not suitable for temporary server configurations, since they also need access to the domain controller. Communication with a remote computer through a VPN demonstrates typical performance limitations for all client-server applications. The multiuser server is also affected.

2.13.10.6 Network in a Windows domain with Internet access via proxy

Introduction

Below, you can find information on a server configuration with a dedicated multiuser server in a Windows domain with intranet/Internet access via a proxy.

The graph shows the possible structure of such a configuration:

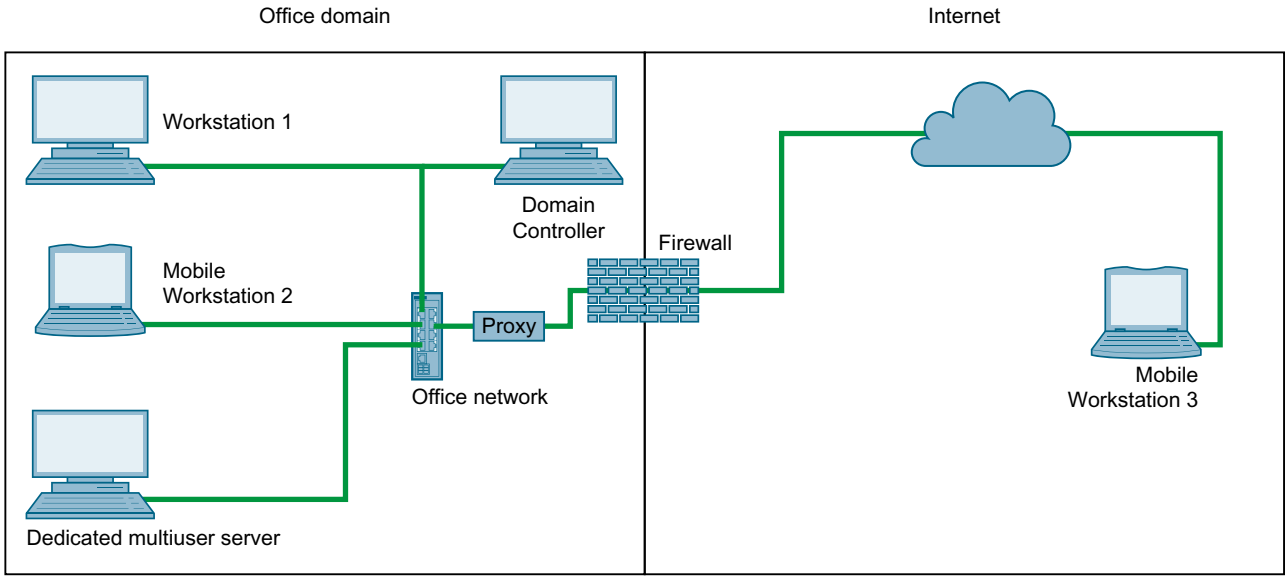
Office domain
Internet
Workstation 1
Domain Controller
Firewall
Mobile Workstation 2
Proxy
Office network
Mobile Workstation 3
Dedicated multiuser server

Description of the network configuration in a Windows domain with proxy

The network configuration should meet the following requirements.

Network environment	Configuration: • All Windows workstations on which the TIA Portal is installed are members of the same domain. • A dedicated Windows server as a multiuser server is also located in the same domain. • All workstations and multiuser servers use the same LAN (Local Area Network). • Both workstations and the multiuser server have access to the domain controller. • Workstations outside the LAN can also be connected to the LAN via a proxy server.
Network	Requirements: • There is a direct TCP connection between the workstations and the multiuser server via the selected port. • No outbound firewall settings blocks communication on the workstations. • No incoming firewall settings blocks communication on the server. • Both the multiuser server and all workstations are in the same IP network (without routing). • If a router is installed in between, the router should not block communication via the selected server port. • Either no proxy server is configured, or there is a local bypass. The "Bypass proxy server for local addresses" option should be selected in the Windows settings for a local area network. • For security reasons, it is not recommended to make the multiuser server available on the Internet. • For security reasons, it is strongly recommended to set up a firewall to accept the incoming connections to the selected port. • The proxy server must accept HTTPS communication and forward it to the multiuser server on the selected port.

Authentication	Authentication is performed based on the current configuration selected. **User accounts and Windows workgroups when configured as an office workgroup:** • These must be created locally on the multiuser server. • This must be configured in accordance with the multiuser rights and roles for servers and projects. • Name and password for authentication are queried on all workstations. • Only local accounts of the server system can be used for authentication. **Note**: The Windows authentication can be used to query name and password. You have to create local accounts, which are identical to the accounts that exist on the multiuser server. This procedure allows the server to automatically authenticate a local account with identical logon information for an existing server account. **Restriction:** User accounts and groups cannot be managed centrally; local users and groups are required for this. This configuration is therefore unsuitable for enterprise networks. **User accounts and Windows workgroups with configuration as Windows domain:** • These must be created on the domain controller. • This must be configured in accordance with the multiuser rights and roles for servers and projects. • Users at workstations on which the TIA Portal is running are automatically authenticated with their domain logon information. **Note**: You can also log on with other valid logon data on the multiuser server. **Restriction:** This configuration is not suitable for temporary server configurations, since they also need access to the domain controller. Communication with a remote computer through a proxy demonstrates typical performance limitations for all client-server applications. The multiuser server is also affected.
Port	**Note**: Sometimes it is only possible to use port "443" for an https connection on a proxy server. A port other than "443" for an https connection cannot be set on some proxy servers. In this case, you need to install the multiuser server on port "443" as well.

2.14 Working with the multiuser server

2.14.1 Introduction to the multiuser server

Introduction

To be able to work with the full functionality of Multiuser Engineering , you need a multiuser server, which manages your multiuser projects.

The multiuser server must be installed first and then configured and started with the help of the multiuser tools.

The connections to the multiuser server in the TIA Portal can then be configured and administered using the "Options" > "Settings" > "Multiuser" command.

In the TIA Portal, you can execute the following actions for the server connections:

- Add new multiuser server connections.
- Create, edit, and delete existing multiuser server connections

Properties of the multiuser server

The multiuser server has the following properties:

- You need administrator rights to install and configure the multiuser server.
- The installation of the server requires settings in the Windows firewall and may require a certificate.
- The multiuser server is configured and administered by an administrator with the supplied multiuser tools.
- The server is available to all editors of the multiuser project and it can also be accessed from the outside.
- The multiuser server can be started and administered outside the TIA Portal.
- The "Alias" assigned as the name for the multiuser server is used as an identifier and may not be changed at a later point in time.

Microsoft system account for multiuser server

When the multiuser server is installed, a Microsoft system account is created for the multiuser server. Microsoft changes the password cyclically for this account.

If the password for this account has expired, the password needs to be changed and then the multiuser server needs to be restarted.

Quantity structure for the multiuser server

For efficient work with Multiuser Engineering, you can create up to 100 multiuser server connections.

When you have reached this number, you receive a message informing you that you cannot create any new server connections. Once you have deleted any server connections that are no longer required, you can once again create new connections up to the maximum number.

The following configuration limits are available for a multiuser server depending on the respective hardware:

Existing projects and revisions on the multiuser server	Standard multiuser server (application as a temporary multi-user server) Hardware: Field PG M5 Advanced (or equivalent) Intel® Core™ i5-6440EQ up to 3.4 GHz·,16 GB RAM (32 GB for large projects) SSD with 50 GB of available disk space Network 1 GB 15.6" Full HD Display (1920 x 1080)
Maximum number of server projects per server	1000
Maximum number of active projects per server	100
Maximum number of active work areas per project	64
Maximum number of parallel load processes per server	10
Maximum number of saved revisions per server	Unlimited, number can be configured Default setting = 10 Make sure that sufficient memory space is available to save the revisions.

Note

Server hardware

When more powerful server hardware is used, even higher quantities are possible.

See also

Displaying the multiuser server connection (Page 121)

Editing the multiuser server connection (Page 121)

Deleting the multiuser server connection (Page 123)

Setting the multiuser network profiles (Page 118)

2.14.2 Setting the multiuser network profiles

Introduction

When working with Multiuser Engineering you have the option of selecting between three different network profiles, depending on the performance capability of your own network.

The selected network profile determines the time until a timeout message is generated.

Available multiuser network profiles

The following network profiles are available for working with the Multiuser server:

Network profile	Area of application
"Fast"	• Office environment • Gigabit-Ethernet • Stable and mostly interference-free network environment
"Medium"	• Office environment, small workgroup with fixed location • Fast Ethernet, WLAN, VPN connections • Good network environment, rare interference
"Slow"	• Commissioning, non-stationary workstations • WLAN connection • Interference possible (machines, electrical installations)

Multiuser network profiles and compression

You have the option to select from the drop-down list the required network profile that matches the performance capability of the network you are using.

The selected Multiuser network profile determines the time until a timeout message is generated.

Setting the network profile to "Middle" or "Slow" activates compression for the data transfer.

No compression is used for the network profile "Fast".

Note

A change to the network profile only takes effect after a restart of the Multiuser client or the Multiuser server.

Setting the multiuser network profiles

To set the network profiles in the TIA Portal, follow these steps:

1. In the TIA Portal, select the command "Options" > "Settings" > "Multiuser".
   The settings displayed below contain the available options for the network profiles.
2. From the drop-down list for "Multiuser network profiles", select the desired profile.
3. Restart either the Multiuser client or the Multiuser server so that the changes to the network profile can be applied.

Result

The selected connection profile is activated.

The compression is activated depending on the profile selected.

Note

The timeout behavior of the current network profile is not affected at the changeover.

2.14.3 Adding a new multiuser server connection

Introduction

If you work with Multiuser Engineering and have installed a multiuser server, you can add up to 100 server connections to the multiuser server in the TIA Portal.

Requirement

You have installed TIA Portal and a multiuser server.

The multiuser server must be configured and started with the help of the Configuration and Administration Tools.

Add new multiuser server connection

To add a new multiuser server connection, follow these steps:

1. In the TIA Portal, select the command "Options > Settings > Multiuser".
   The "Multiuser server connections" tab is open and all existing server connections are displayed.
   If no multiuser server connection has been configured yet, only the entry for the "Local multiuser server" is available.
2. In the following editor, click the "Add server connection" command in the "Server name" column.
   If the maximum number of 100 server connections has already been reached, you receive a message informing you that you first have to delete one of the existing server connections before you can add a new one.
   If it is possible to add a new server connection, the "Add new Multiuser server connection" dialog box opens.
3. Enter the name for the multiuser server in the "Server name" field.
   You can use a maximum of 256 characters for the name and the path. The name can include letters, characters, numbers and commas.
   You will be notified if the entered name is already being used or if the path is longer than 256 characters in total.
4. Select the required transmission protocol (https or http) for the multiuser server. The "http" protocol is selected by default.

5. Enter the required host and port for the server connection.
   You will be notified if the entered host and/or port is already in use.
6. Click "Add".

Result

The new multiuser server connection is added and displayed under "Options > Settings > Multiuser".

2.14.4 Displaying the multiuser server connection

Introduction

When you are working with Multiuser Engineering you can display the existing server connections to the multiuser server in the settings of the TIA Portal.

Requirement

You have installed TIA Portal and a multiuser server.

The multiuser server must be configured and started with the help of the multiuser tools.

Displaying the multiuser server connection

To display the existing multiuser server connections, follow these steps:

1. In the TIA Portal, select the command "Options > Settings > Multiuser".
   The existing multiuser server connections are displayed in the "Server name" column in the next editor.
2. If you are looking for a specific server connection, click the "Server name" or "Host" column headers to sort the content.

Result

The existing multiuser servers are displayed in the editor under "Options > Settings > Multiuser"

2.14.5 Editing the multiuser server connection

Introduction

When you are working with Multiuser Engineering you can edit the existing server connections to the multiuser server and add new server connections in the settings of the TIA Portal.

Requirement

You have installed TIA Portal and a multiuser server.

The multiuser server must be configured and started with the help of the multiuser tools.

Editing the multiuser server connection

1. In the TIA Portal, select the command "Options > Settings > Multiuser".
   All existing multiuser server connections are listed in the next dialog in the "Server name" column.

   Note

   It is not permitted to edit a multiuser server connection that is used in a currently open local session.

2. If necessary, click the "Server name" or "Host" column headers to sort the contents.
3. Double-click the server connection that you wish to edit.
   Result: The "Edit server connection" dialog opens.
4. You can enter the desired changes here.
   – The "Server name" field cannot be edited.
   – The entry in the "Host" field can be edited.
   – The entry in the "Port" field can be edited.
5. If possible, select "https" as the transmission protocol to ensure secure data exchange with encryption and certificate.
6. Click "OK" for the changes to become effective.

Result

Your changes for the selected multiuser server connection are applied and displayed under "Options > Settings > Multiuser".

2.14.6 Restore multiuser server connection

Introduction

When you are working with Multiuser Engineering the existing server connection to the multiuser server can be interrupted.

If this is the case, you will receive a corresponding error message after about 30 seconds.

You have different options, depending on whether you are working in the local session or in the server project view.

You have the following options when you are working in the local session:

- Restore the server connection
- Save and close the local session
- Close the local session without saving

You have the following options when you are working in the server project view:

- Restore the server connection
- Discard changes

Possible procedure in the local session

To restore a multiuser server connection, follow these steps:

1. Click "Restore" in the displayed dialog.
   If the server connection can be restored, the dialog closes and you can continue working without restrictions.
2. Click "Save and close the local session", if you are not continuing to work at the moment but want to save your changes.
3. Click "Close local session", if you want to exit the local session without storing it.

Possible procedure in the server project view

To restore a multiuser server connection, follow these steps:

1. Click "Restore" in the displayed dialog.
   If the server connection can be restored, the dialog closes and you can continue working without restrictions.
2. Click "Discard changes", if you exit the server project view without saving it and want to discard any changes you may have made.
   The server project view closes.

2.14.7 Deleting the multiuser server connection

Introduction

When you are working with Multiuser Engineering you can delete the existing server connections to the multiuser server in the settings of the TIA Portal.

You can create, edit and delete up to 100 multiuser server connections in the TIA Portal.

Note

Keep in mind that you can only delete a server connection if all existing local sessions for this server connection were closed beforehand.

Keep in mind that a connection from a local session to a deleted multiuser server connection is no longer possible.

Requirement

You have installed TIA Portal and a multiuser server.

The multiuser server must be configured and started with the help of the multiuser tools.

Deleting the multiuser server connection

To delete an existing multiuser server connection, follow these steps:

1. In the TIA Portal, select the command "Options > Settings > Multiuser".
   All existing multiuser server connections are listed in the "Server name" column in the next editor.
2. If necessary, click the "Server name" or "Host" column headers to sort the contents.
3. Select one or more server connections that you wish to delete and click "Delete" in the shortcut menu.
   The "Delete" dialog box opens.
4. Click "Yes" to confirm the next message and to delete the selected server connection permanently.

Result

The selected multiuser server connection is deleted and is no longer displayed under "Options > Settings > Multiuser".

2.15 Working with the local multiuser server

2.15.1 Introduction to the local multiuser server

Introduction

To work with Multiuser Engineering , you need a multiuser server that manages your multiuser projects.

A local multiuser server is always installed automatically together with the installation of the TIA Portal and cannot be deleted. It does not require a separate software installation.

The local multiuser server is displayed in the list of Multiuser servers as the first entry under the non-editable "Local Multiuser Server" name and can be called with the " Options" > "Settings" > "Multiuser" command.

Advantages of the local multiuser server

The local multiuser server offers the following advantages:

- You can try out Multiuser Engineering without having to install a dedicated multiuser server first.
- You can buffer your work progress on the local multiuser server.

Requirement

TIA Portal must be installed before you can use the functionality of the local multiuser server.

Properties of the local multiuser server

Unlike a dedicated multiuser server, the local multiuser server offers only limited functionality.

The local multiuser server has the following properties:

- It is only available to one user on their computer and cannot be deleted.
- It can only be started within the TIA Portal.
- It can only be called from the computer on which the local session is running that is assigned to this local multiuser server.
- It does not support management tasks.
- Its settings do not have to be adapted for the firewall and it does not need certificates.
- Only the port can be edited; the server name and host cannot be changed.

See also

Configuring the local multiuser server (Page 126)

Displaying the local multiuser server (Page 127)

Starting the local multiuser server (Page 128)

Exiting the local multiuser server (Page 129)

2.15.2 Configuring the local multiuser server

Introduction

A local multiuser server is installed by default during installation of the TIA Portal by default.

It is displayed in the list of multiuser servers as the first entry and can be called with the command "Options > Settings > Multiuser" under "Multiuser server connection".

Note

It is not permitted to configure the local multiuser server while it is connected to an open local session.

If a local session has already been created and the local multiuser server is only reconfigured afterwards, the server must first be stopped and then restarted to prevent the existing local session from becoming invalid.

Requirement

You have installed the TIA Portal.

Configuring the local multiuser server

To configure the local multiuser server, follow these steps:

1. In the TIA Portal, select the command "Options > Settings > Multiuser".
   All existing multiuser servers are displayed under "Multiuser server connection" in the "Server Name" column.
2. If necessary, click the "Server name" or "Host" column headers to sort the contents.
3. Position the cursor in the corresponding line and select the command "Edit server connection" from the shortcut menu.
   - The "Server name" and "Host" fields cannot be edited for the local multiuser server.
   - The entry in the "Port" field can be edited. Enter the required port here.
4. Click "OK" for the changes to become effective.

Result: Your changes to the local multiuser server are applied and displayed under "Options > Settings > Multiuser" under "Multiuser server connection".

Note

The local multiuser server cannot be deleted.

The local multiuser server is displayed with the uneditable server name "Local Multiuser Server" in all languages.

See also

Exiting the local multiuser server (Page 129)

2.15.3 Displaying the local multiuser server

Introduction

A local multiuser server is installed by default during the installation of the TIA Portal.

Requirement

You have installed the TIA Portal.

Displaying the multiuser server

To display the local multiuser server, follow these steps:

1. In the TIA Portal, select the command "Options > Settings > Multiuser".
   The local multiuser server is displayed under "Multiuser server connection" as the first entry in the "Server name" column.
   The dedicated multiuser servers you have created are listed below this. If the local multiuser server is not displayed as the first entry, click on the "Server name" column to sort the displayed entries.

Note

The local multiuser server cannot be deleted.

The local multiuser server is displayed with the uneditable server name "Local Multiuser Server" in all languages.

2.15.4 Starting the local multiuser server

Introduction

A local multiuser server is installed by default during the installation of the TIA Portal.

It is displayed in the list of multiuser servers as the first entry and can be called with the command ""Options > Settings > Multiuser" under "Multiuser server connection".

Starting the local multiuser server

To start the local multiuser server, follow these steps:

1. In the TIA Portal, select the command "Project > Multiuser > Manage multiuser server projects".
   The "Manage multiuser server projects" dialog box opens.
2. In the "Select server" box, select the local multiuser server from the drop-down list. It is displayed under the name "Local Multiuser Server".
   - Any existing server projects are listed here. The individual columns include detailed information on the respective project relating to path and modification date. If required, you can change the arrangement of the columns.
   - If the local multiuser server has not yet been started, you will be asked if you want to start it now.
   - You will be informed if a connection to the selected server is not possible.
3. Confirm the next query as to whether the local multiuser server is to be started on your computer with "Yes".

Result

The connection to the local multiuser server is being established. A command window opens and shows that the server is now activated.

Note

Although the connection to the local multiuser server is terminated by closing the TIA Portal, this server is not stopped automatically.

To stop the local multiuser server, you must close the displayed command line.

Opening the local session assigned to the local multiuser server

As soon as you open a local session that is assigned to the local multiuser server and when the local multiuser server has not been started, you are asked if you want to start the local multiuser server.

If you answer this query with "Yes", the local multiuser server is started and the local session is opened for editing.

You cancel the process with "No".

Note

Keep in mind that you can only delete a server connection if all existing local sessions for this server connection were deleted beforehand.

Keep in mind that a connection from an existing local session to a deleted multiuser server project is no longer possible.

2.15.5 Exiting the local multiuser server

Introduction

When you open a local session that is assigned to the local multiuser server , a prompt appears asking whether the local multiuser server is to be started. When the local session is ended or the TIA Portal is closed, the local multiuser server is not automatically ended.

The local multiuser server must be explicitly ended.

Requirement

The local multiuser server was started.

Procedure

To end the local multiuser server, follow these steps:

1. Terminate the local multiuser server by closing the displayed command line.

2.16 Creating and managing multiuser server projects

2.16.1 Introduction to working in multiuser server projects

Introduction

Once the multiuser server has been installed and then configured and started with the help of the multiuser tools , you can manage the multiuser server projects in the TIA Portal.

Managing the multiuser server projects

When you run the command "Project > Multiuser > Manage multiuser server projects", all available server projects and local sessions for the existing server connections are displayed in the next dialog.

If you have not added a multiuser server project yet, the list is empty.

If server projects already exist, they are displayed. From the drop-down list in the top part of the dialog, select the required server connection for which all existing server projects and local sessions are then listed.

You can execute the following actions with the individual buttons:

- "Select server": Adds a new server connection.
- "Update": Updates the display in the dialog.
- "Manage server projects": Opens the display for the multiuser server projects in the settings.
- "Add project to the server": Adds a new server project to the selected server connection.
- "Create new local session": Creates a new local session for the selected server project.
- "Open": Opens the selected local session.
- "Close": Closes the dialog.

Note

Displaying server projects

Server projects can only be displayed when a connection to the multiuser server exists and it has been started.

You will be notified if a connection could not be established.

Note

Deleting server projects

To prevent inconsistencies in the project management, server projects must not be deleted "manually" with the Explorer in the storage directory.

In the context of Multiuser Engineering, server projects must always be deleted with the Multiuser Tools.

All associated local sessions must be deleted before a server project can be deleted.

Commands in the shortcut menu

The following actions can be executed in the "Manage multiuser server projects" dialog from the shortcut menu:

- Adding projects to the multiuser server.
- Creating a new local session.
- Deleting a local session.
- Exporting a local session as single-user project.

See also

Adding a new multiuser server project (Page 131)

2.16.2 Adding a new multiuser server project

Introduction

When working with Multiuser Engineering, you must first create a multiuser server project. A multiuser server project is created by adding a standard project (single-user project) to the previously installed multiuser server and then be able to use this project as a multiuser server project.

Note

Multiuser server projects

Observe the requirements described under Requirements for multiuser projects (Page 34), which a multiuser project has to fulfill.

Requirement

You have installed TIA Portal and a multiuser server.

Procedure

To create a multiuser server project, follow these steps:

1. In the TIA Portal, click the menu command "Project > Multiuser > Manage multiuser server projects".
2. Select the required sever connection from the drop-down list in the next dialog box.
   If you select the "Local Multiuser Server", a prompt appears asking if you want to start the local multiuser server.
   In this case, click "Yes".
3. Click the "Add project to the server" button.
4. If required, set the required source path to the single-user project in the following dialog or navigate to the required directory using the "..." button.
5. Select the project which you want to add as multiuser server project.
6. If required, enter a comment on the multiuser server project under "Details".
   The information under "Server project name" and "Published by" are always preassigned and cannot be edited.
7. The "Create local session" button is disabled by default. Enable this button if you do want to create a local session for the newly added multiuser server project.
8. Click "Add".
   If the selected project already exists as multiuser server project or is currently opened by another user, you will be notified.
9. The "Create local session" dialog box opens. You can enter details on the local session here. To create a local session, select the "Open local session" check box and click "Create".

Result

The selected single-user project is added as new multiuser server project and displayed under "Project > Multiuser > Manage multiuser server projects".

If the option "Create local multiuser session" was selected, a new local session is created and opened.

See also

Requirements for multiuser projects (Page 34)

2.17 Working in the local session

2.17.1 Creating and managing a local session

2.17.1.1 Creating a local session

Introduction

To work with Multiuser Engineering , you must create a local session. The local session is always assigned to a specific multiuser server project.

Requirement

You have installed TIA Portal and a multiuser server.

You have created an executable multiuser server project.

Procedure

To create a local session, follow these steps:

1. In the TIA Portal, select the command "Project > Multiuser > Manage multiuser server projects".
   The "Manage multiuser server projects" dialog box opens.
2. Select the required multiuser server in the drop-down list.
3. Select the required project from the displayed list and double-click "Create new local session".
   The "Create local session" dialog box opens.
4. In the "Create local session" dialog box, check your selection regarding the multiuser server and the multiuser server project.
   The displayed selection cannot be modified in this dialog box.
   If the displayed selection is not what you require, click "Cancel" and start again with step 1.

5. Add the detailed information to the local session:
   - Enter a valid path for storage of the local session or use the corresponding button to navigate to the required location.
   - Assign a name for the directory in which the local session is to be saved. The directory is created implicitly if it does not exist yet.

     Note

     Keep in mind that the author for the local session was already entered by default in this dialog box and cannot be edited.

   - The "Open local session" button is selected by default and causes the new local session to open as soon as it has been created. If desired, deactivate this button.
6. Click the "Create" button to execute this action.
   A message indicates that the local session is being created.

Result

The newly created local session is opened and displayed in the project tree.

See also

Opening a local session (Page 134)

Saving a local session (Page 137)

Closing a local session (Page 138)

Deleting a local session (Page 138)

Displaying all local sessions associated with a server project (Page 140)

Exporting a local session as single-user project (Page 141)

2.17.1.2 Opening a local session

Introduction

You have different options for opening a local session:

- In the TIA Portal with "Recently used projects"
- In the TIA Portal, select the command "Project > Multiuser > Manage multiuser server projects".
- With the Windows Explorer when the TIA Portal is closed

As soon as you open a local session that is assigned to the local multiuser server, a prompt appears asking if you want to start the local multiuser server, if it has not yet been started. Click "Yes" to start the local multiuser server and to open the selected local session.

Note

Instructions for opening local sessions:

- A renamed or deleted local session can no longer be opened.
- A local session cannot be opened as a reference project.

When opening a local session that meets one of the criteria listed above, you will see a note informing you that the session cannot be opened. In this case a prompt asks whether you want to export this local session as single-user project.

Requirement

You have installed TIA Portal and a multiuser server.

You have created an executable multiuser server project and a valid local session.

Opening a local session

To open a recently used local session, follow these steps:

1. In the TIA Portal, select the command "Project > Open".
   The "Open project" dialog box opens with a list of recently used projects and sessions.
   The icons in the first column help you distinguish between single-user projects and multiuser projects.
2. Select the preferred local session and click "Open".

Result

The selected local session is opened.

Opening a local session by means of the associated server project

To open a local session via the associated server project, follow these steps:

1. In the TIA Portal, select the command "Project > Multiuser > Manage multiuser server projects".
   The "Manage multiuser server projects" dialog box opens.
2. Select the required multiuser server in the drop-down list.
3. Select the required local session from the displayed list and click "Open".

Result

All opened dialog boxes are closed and the selected local session is opened.

Opening a local session with the Windows Explorer

To open a local session with the Windows Explorer, follow these steps:

1. While the TIA Portal is closed, navigate to the directory in which your local sessions are stored in Windows Explorer.
2. Select the desired local session with the file extension "*.als <version>" and double-click to open it.

Result

The TIA Portal is started and the selected local session is opened.

Note

You close a local session as you would a single-user project with the command "Project > Close".

Opening a local session from a different storage location

To open a local session that has been moved to another storage location in the meantime, follow these steps:

1. In the TIA Portal, select the command "Project > Open".
   The "Open project" dialog box opens with a list of recently used projects and sessions.
2. Select the preferred local session and click "Open".
   You are notified that you are trying to open a local session from a different storage location.
3. Confirm this query with "OK" if you want to execute the action.

Result

The selected local session is opened from the new storage location.

The information about the new storage location is saved in the associated server.

Opening a local session without a server connection

A local session can also be opened when the associated server is currently not available. In this case, when you open the local session, you receive a message which informs you that you can continue working offline in the local session with the following restrictions:

- You cannot see the changes made by other operators in your local session. No yellow or red flags and no "outdated" objects are displayed.
- You cannot check in your changes to the server project until the server connection has been restored.

If you confirm this message with "OK", you can continue working in your local session even without a server connection.

See also

Creating a local session (Page 133)
Saving a local session (Page 137)
Closing a local session (Page 138)
Deleting a local session (Page 138)
Displaying all local sessions associated with a server project (Page 140)
Exporting a local session as single-user project (Page 141)

2.17.1.3 Saving a local session

Introduction

You have different options for saving a local session:

- As "local session" with the file extension "*.als"
- After an export as "single-user project" with the file extension "*.ap".

When you export and save a local session as single-user project, you can continue to edit this project in the familiar way without multiuser functionality.

While working in the local session, you save this as usual as local session in order to continue to work in the context of Multiuser Engineering .

Requirement

You have installed TIA Portal and a multiuser server.

You have created an executable multiuser server project and a valid local session.

Procedure

To save a local session, follow these steps:

1. Open the local session:
2. Make your required changes.
3. In the TIA Portal, select the command "Project > Save".

Result

The local session is saved on the associated multiuser server with the file extension "*.als".

See also

Creating a local session (Page 133)
Opening a local session (Page 134)
Closing a local session (Page 138)

Deleting a local session (Page 138)

Displaying all local sessions associated with a server project (Page 140)

Exporting a local session as single-user project (Page 141)

2.17.1.4 Closing a local session

Introduction

You close a local session as you would a single-user project with the command "Project > Close".

Requirement

You have installed TIA Portal and a multiuser server.

You have created an executable multiuser server project and a valid local session that is open.

Procedure

To close a local session, follow these steps:

1. In the TIA Portal, select the command "Project > Close".
2. Confirm the next prompt with "Yes" if you want to save your changes.

Result

The local session is closed.

See also

Creating a local session (Page 133)

Opening a local session (Page 134)

Saving a local session (Page 137)

Displaying all local sessions associated with a server project (Page 140)

Exporting a local session as single-user project (Page 141)

2.17.1.5 Deleting a local session

Introduction

To avoid inconsistencies in multiuser projects, you **cannot** delete a local session in the usual way with the command "Project > Delete project...".

Local sessions can only be deleted in the context of the multiuser functionality and depending on the associated multiuser server project.

Note

If you have manually deleted a local session with the Windows Explorer from the storage directory, it is still displayed in the "Local sessions" dialog box but can no longer be opened.

Delete this local session again explicitly in the multiuser context as described in the procedure below.

Local sessions that existing in the multiuser server can also be deleted with the Multiuser Tools in the server management. For more details, see:Commands for managing the multiuser server (Page 90)

Requirement

You have installed TIA Portal and a multiuser server.

You have created an executable multiuser server project and a valid local session.

The local session you wish to delete must not be open.

Procedure

To delete a local session in the TIA Portal, follow these steps:

1. In the TIA Portal, select the command "Project > Multiuser > Manage multiuser server projects".
   The "Manage multiuser projects" dialog box opens with a list of sessions available for the selected server.
2. Select the respective server from the drop-down list.
3. Select the required server project from the list and click on the relevant local session that you want to delete.
4. Select the "Delete" command in the shortcut menu.

   Note

   A local session cannot be deleted if it is open; the delete command is inactive in this case.

   In this case, close the local session first and click the "Delete" button once again.

5. Confirm the next query regarding deletion with "Yes".

Result

The selected local session is deleted and also removed from the list of existing local sessions.

See also

Creating a local session (Page 133)

Opening a local session (Page 134)

Saving a local session (Page 137)

Closing a local session (Page 138)

Displaying all local sessions associated with a server project (Page 140)

Exporting a local session as single-user project (Page 141)

Commands for managing the multiuser server (Page 90)

2.17.1.6 Displaying all local sessions associated with a server project

Introduction

You have the option of displaying all local sessions associated with a multiuser server project in the TIA Portal.

Requirement

You have installed TIA Portal and a multiuser server.

You have created an executable multiuser server project and a valid local session.

Displaying all local sessions

To display all local sessions belonging to a server project, follow these steps:

1. In the TIA Portal, select the command "Project > Multiuser > Manage multiuser server projects".
   The "Manage multiuser server projects" dialog box opens and displays all available server connections for the multiuser server.
2. Select the required server connection from the drop-down list.
   All server projects associated with this server connection are listed.
3. Select the required server project and open it in order to show all associated local sessions.
4. If you want to open a local session, select the respective session and click "Open".

Result

All existing local sessions for the selected server project are displayed.

See also

Creating a local session (Page 133)

Opening a local session (Page 134)

Saving a local session (Page 137)

Closing a local session (Page 138)

Deleting a local session (Page 138)

Exporting a local session as single-user project (Page 141)

2.17.1.7 Exporting a local session as single-user project

Introduction

When working with Multiuser Engineering, you have the option of exporting a local session as single-user project.

This is beneficial when a local session can currently not be opened, for example, because the multiuser server is not available.

However, a local session may also become invalid and can therefore no longer be opened.

A local session becomes invalid for the following reasons:

- When the local session was renamed.
- When the local session has been moved to a different storage location.
- When the settings for the associated server have been changed and the server is no longer available for this reason.

If a local session is detected to be invalid when opened, you will receive a query as to whether you want to export the session as a single-user project.

This means you can save the work results from the local session in a single-user project and still use them.

You can apply the changes from the exported local session into your multiuser server project once again by copying them.

You also have the option of exporting a local session with the multiuser Power Tools.

For more details, see: Commands for managing the multiuser server (Page 90)

Exporting a local session as single-user project

To export a local session as single-user project, follow these steps:

1. In the TIA Portal, select the command "Project > Multiuser > Manage multiuser server projects".
   The "Manage multiuser server projects" dialog box opens and displays all available server connections for the multiuser server.
2. Select the required server connection from the drop-down list.
   All server projects associated with this server connection are listed.
3. Select the required server project and open it in order to show all associated local sessions.

4. Select the local session and click "Export as single-user project" in the shortcut menu. The "Export as single-user project" dialog box opens. The name of the local session and the name of the export file are preset and cannot be edited.
5. Enter the desired export path or navigate to the desired storage location and click "Export". If the target directory is not empty, a corresponding message is output to this effect. Select a new, empty target directory and click "Export" again.

Result

The local session is exported as single-user project and saved under the set storage path.

You will receive a message that the export has been successfully completed.

If you click "OK", the export dialog box closes and the server project list is displayed once again.

See also

Creating a local session (Page 133)

Opening a local session (Page 134)

Saving a local session (Page 137)

Closing a local session (Page 138)

Deleting a local session (Page 138)

Displaying all local sessions associated with a server project (Page 140)

Commands for managing the multiuser server (Page 90)

2.17.2 Editing objects in the local session

2.17.2.1 Introduction to working in local sessions

Introduction

Within the context of Multiuser Engineering , to work simultaneously with multiple users in a multiuser project, each editor must create their own local session.

The respective editor can make changes in one or more local sessions, check them in to the multiuser server project and thus publish them.

After check-in, the changes from the local session are once again available to all editors in the server project. They can be reused and modified once again in subsequent local sessions.

Procedure when working in local sessions

The following steps are necessary to work as multiuser in the engineering tasks that have been assigned to you in a local session:

- You have installed TIA Portal and a multiuser server.
- Alternatively, you can use the local multiuser server installed by default.
- You have created an executable project for working with Multiuser Engineering.
- You have loaded the prepared project to the multiuser server and thus created a multiuser project.
- You have coordinated the task and the configuration limits for working in the local sessions with the other editors.
- You have created a local session for each engineering task.

Rules for working in local sessions

The following rules apply to working in local sessions:

- A local session is always assigned to a defined multiuser server and a defined multiuser server project.
- Each local session includes different tasks for different editors who are assigned to a multiuser server projects.
- A local session is given the file extension "*.als" when it is saved and must not be renamed in Windows Explorer.
- While a local session is open, the associated multiuser server connection and the associated multiuser server project must not be renamed, moved or deleted.
- Local sessions can no longer be opened and edited when the associated multiuser server or the associated multiuser server project have been deleted.
- As of TIA Portal V15, a local session can also be opened without a server connection. After confirming a corresponding message, you can continue working offline, that is without a server connection.
- A local session cannot be opened as reference project.
- A local session can be exported as single-user project and edited further without multiuser functionality.
- A local session cannot be archived or retrieved. These actions are possible as soon as the local session is once again edited as single-user project.
- Working with external tools is not possible in the local session. Changes by external tools in local sessions are lost after the local session is checked in and refreshed. External tools that make changes to the file structure of a multiuser project may only be used in the server project view.
- Changing of user-defined documentation is not possible in the local session. Changes to existing user-defined documentation are lost after the local session is checked in and refreshed. Changing of user-defined documentation is only possible in the server project view.

Note

Renaming objects in local sessions

Objects should be renamed only in the server project view, since this does **not** cause any inconsistencies in the server project.

When renaming objects in the local session, you must ensure that all objects affected by the renaming and their places of use are also marked and checked in the server project. Otherwise, you get compiling errors and an inconsistent server project due to partial renaming.

Before checking the places of use of the renamed object, you should be careful not to create conflicts (red flags) when marking and you should not mark any outdated objects; otherwise, you will overwrite the work of another user when checking in.

Note

Editing library objects in local sessions

Library objects should be edited only in the server project view, since this does **not** cause any inconsistencies in the server project.

With version changes to library objects in the local session, for example, when you create new versions of type instance objects, you must ensure that all relevant objects have been marked with a version change and checked into the server project.

Otherwise, you get compiling errors and an inconsistent server project from creating new versions.

Note

You can find detailed instructions on working with local sessions in the sections below.

See also

Displaying usage info for marked objects (Page 152)

Unmarking objects (Page 152)

Check-in objects to the server project (Page 156)

Possible conflicts during check-in (Page 159)

Possible error messages during check-in (Page 161)

2.17.2.2 Working offline in the local session

Introduction

When you open a local session , the connection to the associated multiuser server is automatically established. In the past, you could only work with an active server connection in the local session when working with Multiuser Engineering .

As of TIA Portal V15, you have the option to work in a local session "offline" without a server connection.

If a server connection cannot be established when you open the local session, a dialog appears with a corresponding error message.

You are first be asked whether you want to export this local session as single-user project. If you confirm this option with "OK", a follow-up dialog appears informing you that a connection to the server could not be established.

If you also confirm this dialog with "OK", you can work in the local session even without a server connection with the following restrictions:

- Selections made by other users are not visible in your local offline session; no yellow or red flags and no outdated objects are displayed.
- Changes you are making in the local offline session cannot be checked in until a server connection is available once again.

Note

Instructions for opening local sessions:

- A renamed or deleted local session can no longer be opened.
- A local session cannot be opened as a reference project.

When opening a local session that meets one of the criteria listed above, you will see a note informing you that the session cannot be opened. In this case a prompt asks whether you want to export this local session as single-user project.

Requirement

You have installed TIA Portal V15 or higher and a multiuser server.

You have created an executable multiuser server project and a valid local session.

Opening a local session and working offline

To open a local session, follow these steps:

1. In the TIA Portal, select the command "Project > Open".
   The "Open project" dialog box opens with a list of recently used projects and sessions.
   The icons in the first column help you distinguish between single-user projects and multiuser projects.
2. Select the preferred local session and click "Open".

3. If a server connection cannot be established to the associated multiuser server, you receive a corresponding message. You are asked whether you want to export this local session as single-user project.
4. Click "OK".
5. A second dialog opens which informs you that a server connection could not be established. If you confirm this dialog with "OK", you can continue working in your local session even without a server connection with the restrictions listed above.

Result

The selected local session is opened offline.

You do not see any selections made by other users.

You cannot check in your changes until a server connection is available once again.

2.17.2.3 Rules for selecting objects

Introduction

For objects to be edited in Multiuser Engineering at the same time, the objects to be edited must be selected in the local session.

There are two equivalent options for selecting objects:

- As of TIA Portal V15 the "Autoselect" function automatically selects all objects approved for editing. This means each permitted "selectable" object in the local session that you create, edit or delete is automatically selected.
- In addition, you can also select objects manually in the local session:
  - With the shortcut menu command "Mark objects for check-in".
  - By clicking the displayed gray flags in the project tree and in the individual editors of the TIA Portal.

  For manual selection of objects, multiple selections are also possible via the shortcut menu. A prerequisite for the marking is that the selection includes at least one object that can be marked. If the multiple selection does not include an object that can be marked, the shortcut menu commands are possibly not active.

There are some editing steps (e.g. changes to the HW configuration) that can only be executed in the server project view.

Rules for marking objects

The following rules apply to marking:

- The markings and the marking column in the project tree and in the individual editors are only displayed in the context of Multiuser Engineering.
- Objects that can be marked are identified by a gray flag in the project tree and in the editors.
- You can only mark objects that can be edited with Multiuser Engineering.

- If an object does **not** have a gray flag in front of it, this object cannot be marked and it should not be edited in the local session as the changes cannot be checked into the server project. Such objects that cannot be marked should be edited in the server project view.
- Some higher-level directories cannot be marked themselves, but the individual objects in the folder can be marked.
  Example: "PLC tag table": The PLC tag table itself cannot be marked, but the individual tags in the open tag table can be marked.
- Autoselect is done by the system when you edit an object that can be selected.
- Manual selection is done by clicking the gray flag or with the shortcut menu. The flag is shown in color as soon as an object has been marked.
- The color of the flags provides additional information about the marking:

Icon	Meaning
	Object can be marked for check-in.
	Object is marked in own local session.
	Object is marked in a different local session that is part of the same multiuser server project.
	Conflict: The object was marked in multiple sessions simultaneously.

Special considerations for autoselect of objects

The following special considerations have to be noted for autoselect:

When working in the different editors of the TIA Portal and changes have been made, the effected objects in the lower-level or higher-level folders are also selected automatically. Some of the actions are listed as an example below.

- Autoselect of objects in Multiuser Engineering works in open editors as well as when working in the project tree.
  - If an object is renamed in the project tree, the new, renamed object is selected automatically.
  - If an object is moved to a different folder within the same device in the project tree, the moved object is selected automatically.
  - If, however, a folder that cannot be selected (e.g. the PLC tag table) is moved to a different location in the project tree, the lower-level objects are **not** selected automatically.
- When working in the project library, all newly created objects are selected automatically. If, however, a complete library folder is moved to a different location in the project tree, the types it contains are **not** selected automatically.
- When working with project texts, the selectable objects to which the project texts belong are selected automatically.
  - Example: In case of changes to a block comment, the associated block is selected automatically so that the changes are applied to the server project during the next check-in.

- When working with PLC message text lists, all changes to texts are selected automatically.
  - If a new entry is created in a text list, the associated text list is selected automatically.
  - If an existing text is edited in a text list, the associated text list is selected automatically.
- When editing PLC supervisions & alarms all changes are selected automatically.
  - If a new entry is added or if existing entries are changed, the changed entry as well as the associated block are selected automatically.
- When editing HMI text and graphic lists all changes are selected automatically. However, the selections are only shown in the higher-level editor.
- When editing HMI tags, only the higher-level tags can be selected.
  - In case of changes to the lower-level tags, the associated higher-level tags are selected automatically.
  - In case of changes in the alarm editor, the alarms associated with these tags are automatically selected but not the tags themselves.
- When editing HMI screens and HMI scripts, there are special considerations regarding autoselect because not all actions performed here result in objects being selected automatically.
  - In case of changes in the script editor, autoselect is only performed when the executed action can be reversed with "Undo".
  - In changes are made to individual elements of a screen, the associated screen is automatically selected.
  - In changes are made to the global filter configuration in the screen properties, these are **not** automatically selected because this configuration is not part of a screen.
- When editing trace and logic analyzer functions, the following actions are **not** automatically selected:
  - Deleting the example OB (OB30) from the programming folder.
  - Changing the cycle time in OB 30.

2.17.2.4 Marking objects automatically or manually for editing

Introduction

For objects to be edited in Multiuser Engineering at the same time, the objects to be edited must be marked in the local session.

There are two equivalent options for selecting objects:

- The "Automark" function which automatically marks the objects edited in the local session. As soon as you open, copy, add or delete an object in an editor of the TIA Portal in the local session, this object is automatically marked for check-in.
  This is also true for objects that are created by the system at a lower level during editing by the user (e.g. system blocks, etc.).
- Objects can still be selected manually:
  - With the shortcut menu command "Mark objects for check-in".
  - By clicking the displayed gray flags in the project tree and in the individual editors of the TIA Portal.

Note

Changes to objects that cannot be marked

Changes to objects that cannot be marked - these are objects that do not have a gray flag - in the local session are possible but **cannot** be applied to the server project.

These changes are lost after update of the local session, because they are once again overwritten with the contents of the server project.

As soon as you want to edit an object that cannot be marked, a banner text is displayed in the open editor. This banner text contains information on the further procedure.

Procedure for automatic selection of objects

To mark objects in a local session automatically, follow these steps:

1. Open the required local session.
2. Edit the required object, for example, by opening, copying, adding or deleting it in an editor.

Result

The object is automatically marked during editing and deleting, and the corresponding blue or red flag is set in the own local session.

If a blue flag is displayed, the object can be edited in the local session.

If a red flag is displayed, you have a conflict that has to be resolved through coordination with the other editors.

If an automark conflict occurs when copying or adding objects, you receive an error message. The further procedure is partly controlled by a dialog in which you can select the required option.

Procedure for manual selection of objects

To mark objects in a local session, follow these steps:

1. Open the required local session and click on the displayed gray flag to mark an object for check-in.
2. Alternatively, you can select the required object and mark it with the shortcut menu command "Mark object for check-in".
3. Mark all objects in your local session that you currently want to edit. You can make multiple selections for marking. After marking, you can edit and save the objects as usual.

Note

Multiple selections

Once > = 50 objects are selected in the local session, both shortcut menu commands are active simultaneously: the command for marking the objects as well as the command for unmarking.

Result

The marked objects are displayed with a blue or red flag.

If a blue flag is displayed, the object can be edited in the local session.

If a red flag is displayed, you have a conflict that has to be resolved through coordination with the other editors.

Note

Markings with conflict (red flag)

A marking conflict must be resolved by agreement of the editors, and a sequence for editing the marked objects must be specified. To solve the conflict, one editor must unmark the objects.

See also

Introduction to working in local sessions (Page 142)

Unmarking objects (Page 152)

Displaying usage info for marked objects (Page 152)

2.17.2.5 Possible conflicts during autoselect

Introduction

For objects to be edited in Multiuser Engineering at the same time, the objects to be edited must be selected in the local session. When automatically selecting objects in the local session, the following conflicts can occur:

- An automatically selected object can be "outdated".
  In this case, it is mandatory that you update the object prior to check-in so that the changes on the multiuser server can be applied.
- An automatically selected object can already have been selected by other users. A red flag is displayed during autoselect in this case and the conflict has to be resolved through coordination with the other editors.

Note

Markings with conflict (red flag)

A marking conflict must be resolved by agreement of the editors, and a sequence for editing the marked objects must be specified. To solve the conflict, one editor must unmark the objects.

Displaying conflicts during autoselect

If an autoselect conflict occurs, you receive an error message in the "Info" tab of the Inspector window.

This error message provides information on the action for which the conflict has occurred and which type of conflict you are dealing with.

The further procedure is partly controlled by a dialog in which you can select the required option.

Example of an error message for autoselect:

The "Add new network" operation results in an autoselect conflict because one of the automatically selected objects is outdated.

Resolving conflicts during autoselect

All conflicts that might occur during autoselect must be resolved before check-in because there is a risk of loss of data otherwise. Changes are lost or overwritten when you check in outdated objects or objects "selected with conflict". Follow these steps:

1. If an object is shown as "outdated" during autoselect, you must update this object in the local session prior to check-in and before you edit it further so that the changes on the multiuser server can be applied.
2. If an object is selected with a red flag during autoselect, this object is already being edited by another operator in a different local session. Resolve this conflict by having one of the editors manually remove their selection.

2.17.2.6 Unmarking objects

Introduction

For objects to be edited in Multiuser Engineering, these objects must either be selected automatically or manually in the local sessions. The objects marked for check-in can also be unmarked.

This is necessary primarily in a conflict (indicated by a red flag) in which multiple editors have marked the same object for check-in in their respective local session.

Note

Markings with conflict (red flag)

A marking conflict must be resolved by agreement of the editors, and a sequence for editing the marked objects must be specified. To solve the conflict, one editor must unmark the objects.

Procedure

To unmark objects in a local session, follow these steps:

1. Open the local session and click on the displayed blue or red flag to unmark an object.
2. Alternatively, you can also select the required object and delete the marking with the shortcut menu command "Unmark".

Result

The objects that are no longer marked and objects marked as conflict are displayed as follows:

Icon before	Action	Icon after
	Unmark	
	Unmark	

See also

Introduction to working in local sessions (Page 142)

Marking objects automatically or manually for editing (Page 148)

Displaying usage info for marked objects (Page 152)

2.17.2.7 Displaying usage info for marked objects

Introduction

For objects to be edited in Multiuser Engineering at the same time, each editor must mark the objects they want to edit in their local session.

To avoid multiple marking of objects and any marking conflicts caused by it, you receive the following information for marked objects:

- User: Indicates who has marked the object
- Computer: Indicates on which computer the object was marked.
- Storage location: Indicates where the object was marked: in a local session or in the server project view.

This information helps you coordinate the further procedure for marked objects or objects that are marked with a conflict with the other users.

Show usage info

To display the usage info for marked objects, follow these steps:

1. Select a yellow or red marked object in your local session or in the server project view.
2. Call the "Usage info" command from the shortcut menu.

Result

The "Usage info" dialog box opens and displays the above-mentioned information for the marked object.

Note

Markings with conflict (red flag)

A marking conflict must be resolved by agreement of the editors, and a sequence for editing the marked objects must be specified. To solve the conflict, one editor must unmark the objects.

See also

Multiuser icons in the user interface (Page 40)

Introduction to working in local sessions (Page 142)

Marking objects automatically or manually for editing (Page 148)

Unmarking objects (Page 152)

2.17.2.8 Refresh local session

Introduction

For objects to be edited in Multiuser Engineering at the same time, the objects edited in the local session must be selected automatically or manually. If objects are more recent in the server project than in the local session, these objects are selected as "outdated".

You can refresh these objects in the local session by clicking the "Refresh local session" icon in the multiuser toolbar.

Other users may have created new objects or deleted objects in the server project that are not yet displayed in your local session.

These new objects are also displayed in your local session after a refresh.

The contents of "outdated" objects in the local session are overwritten with the more recent contents of the server project during the refresh.

Note

Refresh local session

- The button to refresh the local session opens the view "Refresh", even if there are no objects to be refreshed in your local session. When you click the "Start refresh" button, the local session is refreshed.
- During the refresh, all changes to **unmarked objects** in your local session are overwritten with the contents of the server project. The marked objects remain unchanged.
- You can undo the refresh of the local session if required by clicking the "Restore" button. Your local session will then be restored with the contents prior to the refresh.

Refresh procedure

To refresh objects in a local session, follow these steps:

1. Open the local session that contains the objects that are to be updated.
2. Click the "Refresh local session" button in the multiuser toolbar.
   The multiuser editor opens in the refresh view and shows all objects marked in your local session. The objects you have marked are not overwritten during the refresh. Only unmarked objects are overwritten with the contents of the server project.
3. Click the "Start refresh" button to start the refresh.
   You are kept up-to-date on the progress of the refresh. You will receive an error message if an error occurs.
4. You receive the message that the refresh was completed successfully. Click "OK".

Result

All unmarked objects in your local session were refreshed.

The objects marked in your local session were **not** synchronized with the server version.

Note

Online/offline differences after the refresh

After downloading to the device, the online information on the loaded objects is stored in the local session in the TIA Portal. This enables a comparison between the online and offline version. However, because this online information is not checked in on the server, this information is no longer available in the local session after a refresh. If you then work online on the device after the refresh, the TIA Portal cannot perform a comparison because of missing information. The objects are displayed with online/offline differences although the contents of the objects are identical.

To correct this, perform a new complete download to the device after a refresh of the local session.

2.17.2.9 Restoring the local session after the refresh

Introduction

Unmarked objects which have a newer version in the server project can be updated in the local session. The unmarked objects in the local session are overwritten with the more recent contents of the server project during the refresh.

You can undo the refresh of the local session if required. To do this, click the "Restore" button in the multiuser editor after the refresh. Your local session will then be restored with the contents prior to the refresh.

Note

"Restore" button

The "Restore" button is only active after a refresh has been performed as long as a restore of the contents of the local session with the last version is possible.

The "Restore" button is inactive if no restore of the local session can be performed, for example because no refresh was performed.

Closing and opening the editor has no impact on the "Restore" button. If a refresh is available that can be undone, the button is still active even if you close the editor and open it again.

Procedure to restore the local session

To restore the local session after the refresh, follow these steps:

1. Open the local session for which you have performed a refresh, if you have closed this in the meantime.
   Start with step 3 if the refresh view is still open.
2. Click the "Refresh local session" button in the multiuser toolbar to open the multiuser editor in the refresh view.
3. Click the "Restore" button to restore the version of the local session before the last refresh.
4. In the next dialog box, click "Yes" to answer the prompt.
5. When errors occur, you receive an error message that you have to acknowledge.
6. When the restore is successfully completed, you receive a message. Click "OK".

Result

Your local session was successfully restored with the version before the last refresh.

See also

Introduction to working in local sessions (Page 142)

2.17.2.10 Check-in objects to the server project

Introduction

After editing marked objects in the local session, you can check the changes into the server project.

The following rules apply to check-in:

- Only changes to selected objects are applied during check-in to the server project.
- If you have edited objects that cannot be selected in the local session, these changes are **not** applied during check-in.
- During the refresh of your local session after check-in, changes to objects that cannot be selected are overwritten with the contents of the server project.
- You will receive an error message with additional information if an error occurs during check-in. Correct the errors that occurred as specified, either in your local session or in the server project view. Then start the check-in process again.
- During check-in and editing of objects in the server project view, the respective user works exclusively on the multiuser server. The server is therefore locked for all other users. The server status is shown as "not available" in the toolbar during exclusive editing. You can nevertheless create a new local session and refresh local sessions.

Options for the check-in

Compile

You have the option to compile the changes made in the local session before check-in.

Note

Options displayed for the compilation during check-in

The options displayed for the compilation during the check-in depend on the settings made in the TIA Portal under "Options > Settings > Multiuser" for "Compilation settings" .

The compiling options may be deactivated in the multiuser editor based on these settings.

Compile mode

The compile mode defines the extent to which the software changes are compiled. You can select the following options from the drop-down list:

- Marked objects (changes only)
  The changes made to the marked objects for the selected device are compiled together with the device-specific objects.
- Device software (changes only)
  The changes made to the marked objects are compiled together with the software version on the assigned devices.
  Example: You have marked and changed several objects that belong to more than one device (for example to CPU_1 and to CPU_2). These marked objects are compiled together with the software versions of both CPUs during compiling.

Display server project view

When you enable this option, you have the possibility before check-in to again display your changes to the marked objects in the server project view together with the current contents of the server project.

You can mark and edit all objects in the server project view and unmark them. After this, you can either save your changes in the server project or discard your changes and exit the server project view without further action.

Requirement

The connection to the multiuser server exists and the server is available.

You have marked the objects edited in the local session for check-in.

Procedure for check-in of changes

To check-in the changes from the local session to the server project, follow these steps:

1. Open the local session that you want to check in.
2. Click the "Check-in" button in the multiuser toolbar.
   The multiuser editor opens in the "check-in" view and shows all objects marked in the local session that are applied to the server project during check-in.
3. Open the displayed folders and check the objects displayed and marked for check-in.

4. In the toolbar, click on the button with the red flag to display any conflicts.
   Existing conflicts are displayed. To prevent data loss or unintended overwriting, resolve the displayed conflicts before checking in the project.
   If there are no conflicts, click on this button again to return to the previous "Check-in" view.
5. Select the preferred options for check-in.
   – "Compile": Select the required setting from the drop-down list.
   – "Open server project view": When you click this options, the server project view is opened in the project tree prior to check-in.
6. Enter a comment on the change history to document the changes you have made in the local session.
7. Click "Start check-in" after reviewing to check the displayed objects into the multiuser server project.
   – If you have opened the server project view, you can check in your changes with the "Save changes" button. If you click "Discard changes", your changes are not applied in the server project.
   – If you want to keep the marking in the local session even after the check-in, you must select the option "Keep markings for objects".
   – If you have marked and checked in only some of your changes, you should perform **no refresh** to retain your changes to the unmarked objects.
8. If check-in was successful, you receive a final message. If you confirm this message with "OK", your local session is overwritten with the contents of the server project. If you want to keep your changes to the selected objects, click on the "Keep local session" button in the message.
9. If you want to retain your selections in the local session, select the "Keep markings" check box.
10. If check-in was unsuccessful and aborted due to errors, you receive a corresponding error message. Correct the specified errors in your local session or in the server project view. Then start the check-in process again.

Result

You receive a message that the check-in was completed successfully.

Your local session is refreshed and opened with the contents of the server project after successful check-in.

- The local session will include all changes after the refresh that have been checked in in the server project in the meantime.
- If new objects were added to the server project, these objects are now also visible in your local session.
- If objects were deleted in the server project, these objects now no longer exist in your local session.

- Your markings in the local session are automatically deleted if you have not selected the option "Keep marking for objects".
- The server connection is released once again after check-in. The server status changes in the display within the toolbar from "busy" to "available".

See also

Possible conflicts during check-in (Page 159)

Possible error messages during check-in (Page 161)

Introduction to working in local sessions (Page 142)

Introduction to working in the server project view (Page 166)

Editing objects in the server project view (Page 169)

2.17.2.11 Possible conflicts during check-in

Introduction

Errors can occur during check-in of changes from the local session to the associated multiuser server project that may prevent successful check-in.

You have the option of displaying existing errors in the multiuser editor prior to check-in.

To do this, click the red flag in the header of the multiuser editor and all conflicts in the local session are displayed. If there are no conflicts, a message to this effect is displayed.

If you want to check in your changes from the local session to the server project despite existing conflicts, you must confirm this action explicitly in a dialog because this procedure can result in a loss of data.

The error constellations listed below can be easily avoided through coordination between editors and by observing the recommended procedures.

Possible naming conflicts during check-in

Naming conflicts during check-in arise when identical names are used for different objects in different local sessions.

Example:

- Editor 1 creates the tag "Motor_1" in their local session.
- Editor 2 also creates a tag "Motor_1" in their local session.

Result:

Editor 1 can check in their changes without errors.

Editor 2 **cannot** check in their changes. When they want to check in their changes they receive an error message due to a naming conflict and the check-in is aborted. The error must be removed in the local session.

Possible conflicts when creating devices in the local session

If an editor adds new devices with marked objects in a local session, for example, a new CPU with blocks, the result will be a conflict during check-in.

Example:

- Editor 1 creates a new CPU with two function blocks in their local session.
- Editor 1 marks the blocks appropriately before editing them.

Result:

Editor 1 receives an error message during check-in because the newly created CPU cannot be found in the server project. The transfer of devices is currently not supported in Multiuser Engineering. Check-in is aborted in this case and the server project view opens. Editor 1 can copy their newly created objects directly to the server project. This means the editing work is not lost.

Possible conflicts when renaming objects in the local session

Objects should be renamed only in the server project view, since this does **not** cause any inconsistencies in the server project.

When renaming objects in the local session, you must ensure that all objects affected by the renaming and their places of use are also marked and checked in the server project. Otherwise, you get compiling errors and an inconsistent server project due to partial renaming.

Before checking the places of use of the renamed object, you should be careful not to create conflicts (red flags) when marking and you should not mark any outdated objects; otherwise, you will overwrite the work of another user when checking in.

Possible conflicts when changing the device type in the local session

If a device type is changed in a local session, e.g. a "CPU_1" of the type S7-15xx is replaced or renamed, the objects marked in the "old" CPU may no longer be displayed in the local session.

Possible conflicts when deleting devices in the server project

If devices with included objects are deleted in the server project, it may result in a transfer error.

Example:

- Editor 1 works in their local session on marked objects that are part of "CPU_1".
- In the meantime, editor 2 deletes "CPU_1" in their server project view and saves the changes in the server project.

Result:

Editor 1 receives an error message during check-in because "CPU_1" no longer exists in the server project. In this case, editor 1 can add their edited objects in the server project view. The editor either copies "CPU_1" completely to the server project or they add the edited blocks to an existing CPU in the server project.

Possible scenarios during check-in of objects that are deleted in the server project

When you check in objects that are already deleted in the server project, these objects are recreated.

Example:

- User 1 edits the marked tag "Tag_1" in their local session that is part of a group that includes a tag table with multiple tags.
- User 2 deletes this group with its entire tag table in the server project view and saves the changes in the server project.

Result:

User 1 ignores the display that "Tag_1" is outdated, continues working on it and checks in the changes. The group with the tag table is created once again in the server project, but it now only includes the tag "Tag_1".

Renaming objects

You should avoid renaming objects in the local session as it results in naming conflicts during check-in. If objects must be renamed, they should be renamed directly in the server project view. Any local session with marked objects that is associated with this server project must not be present at the same time.

Different project languages in server project and local session

If a project language is changed in a local session, these changes are lost in the local session after check-in and after refresh.

If a project language is subsequently deleted in the server project view, this project language no longer exists in the server project.

Changes to the project languages should always be made only directly in the server project view. Note that no local sessions with marking to this server project were edited or opened.

See also

Introduction to working in local sessions (Page 142)

Opening and closing the server project view (Page 168)

2.17.2.12 Possible error messages during check-in

Introduction

The following described error messages can occur during check-in of marked changes from the local session to the multiuser server project :

- Transfer error
- Compilation error

The procedure for dealing with these error messages is described below.

Transfer error

The following actions are automatically executed when transfer errors occur during check-in:

- An error message will inform you about the type of error that has occurred and the next steps. After acknowledgment, this message is closed.
- In the case of a critical transfer error, the status prior to check-in is restored in the local session as well as in the server project. Changes are not lost. Correct the error in your local session.
- If the transfer error can be corrected in the server project view, the server project view is opened. Correct the error in the server project view.
- The faulty objects are displayed and identified in the multiuser editor in the "Check-in" view.
- The multiuser server locked for check-in is released once again if the errors are corrected in the local session.
- The multiuser server remains locked in the server project view during error correction. Note that no other user can check in their changes during this time period.

Correcting transfer errors

After occurrence of a transfer error, you have the following options for the action to take:

1. Correct the errors in your local session or in the server project view.
2. Then start the check-in process again so that an error-free project can be checked in.

Compilation error

If the option "Compile" is selected during check-in, a compilation process takes place after transfer of the marked changes to the server project with the selected option. The following compile modes are available:

- Marked objects (changes only)
  The changes made to the marked objects for the selected device are compiled together with the device-local objects.
- Device software (changes only)
  The changes made to the marked objects are compiled together with the software version on the assigned devices.

Note

Options displayed for the compilation during check-in

The options for compiling that are displayed during check-in are dependent on the settings made in the TIA Portal under "Options > Settings > Multiuser" in "Compilation settings" .

The compiling options may be deactivated in the multiuser editor due to these settings.

If the compiling process is completed successfully, the changes from the local session are automatically transferred to the server project.

The following actions are taken when a compilation error occurs:

- The progress indicator for check-in is closed, and the check-in is stopped.
- The faulty objects are displayed and identified in the multiuser editor in the "Check-in" view.
- The multiuser server remains locked in the server project view during error correction. Note that no other user can check in their changes during this time period.

Correcting compilation errors

After occurrence of a compilation error, you have the following options for the action to take:

1. Correct the indicated errors in your local session or in the server project view.
2. Then start the check-in with compilation again so that an error-free project can be checked in.

Ignoring compilation errors

You also have the option of ignoring the indicated compilation errors and checking in a project with errors.

NOTICE
Check-in with compilation errors This action is possible but not recommended! To enable efficient work to continue for all editors involved, you should check in only error-free projects on the multiuser server whenever possible.

See also

Introduction to working in local sessions (Page 142)

2.17.2.13 Deleting objects in the local session

Introduction

When working with Multiuser Engineering , you can delete objects in the local session.

The following rules apply to deleting objects:

- Deleted objects are no longer displayed in the local session.
- The "Undo" function can be use to undo the "Delete" action in the local session.
- Deleted objects are automatically marked by the system and displayed in the "Deleted objects" folder.

- If you unmark a deleted an object in the multiuser editor, the deleted object is **not** checked-in in the server project. This means that the deleted object is **not** deleted in the server project. The now deleted but not checked-in object is only displayed after the next refresh.
- When deleting unmarked objects in the local session, these objects are once again present in your local session after the refresh. Only marked objects are transferred during check-in. Since the deleted objects could not be marked, the deletion is not applied in the server project. During the next refresh of the local session, these objects are once again added to the local session.

Requirement

The connection to the multiuser server exists and the server is available.

Procedure for deleting objects

To delete object in the local session, follow these steps:

1. Open the required local session.
2. Delete the requested marked object or delete an unmarked but markable object.
3. Click the "Check-in" button in the multiuser toolbar.
   The multiuser editor opens in the "check-in" view and shows all objects marked in the local session that are applied to the server project during check-in. The objects deleted in the local session are displayed in the "Deleted objects" folder.
4. Check the objects displayed and marked for deletion. You can unmark any objects that are already marked if you do not want to delete them after all.
5. Select the preferred options for check-in:
   – "Compile": When you click this option, your changes are compiled with the option "Software, changes only" prior to check-in. This means that the software changes you have made are compiled with all their dependent objects in the server project.
   – "Open server project view": When you click this options, the server project view is opened in the project tree prior to check-in. The server project view shows you in advance the content of the current server project with your changes marked for check-in. You can mark and edit all objects in the server project view and unmark them.
6. Click the "Check-in" button to check in the displayed objects into the multiuser server project. If you want to keep the markings in the local session even after check-in, you must select the option "Keep markings".
7. If check-in was unsuccessful and aborted due to errors, you will receive the corresponding error messages. Correct the errors in your local session or in the server project view. Then start the check-in process again.

Result

Your local session is refreshed and opened with the contents of the server project after successful check-in.

- The local session will include all changes after the refresh that have been checked in in the server project in the meantime.
- If new objects were added to the server project, these objects are now also visible in your local session.
- The objects deleted in the server project, now no longer exist in your local session.
- Your markings in the local session are automatically deleted if you have not selected the option "Keep objects marked".
- The server connection is released once again after check-in. The server status changes in the display within the toolbar from "busy" to "available".

2.18 Working in the server project view

2.18.1 Introduction to working in the server project view

Introduction

In Multiuser Engineering , you have the option of working directly in the server project view in addition to working in local sessions.

The server project view can be opened for editing with the icon in the multiuser toolbar.

Structure of the server project view

The server project view shows the current content of the server project in the project tree below the local session against a yellow background.

The figure below shows an example of a multiuser project opened in the server project view:

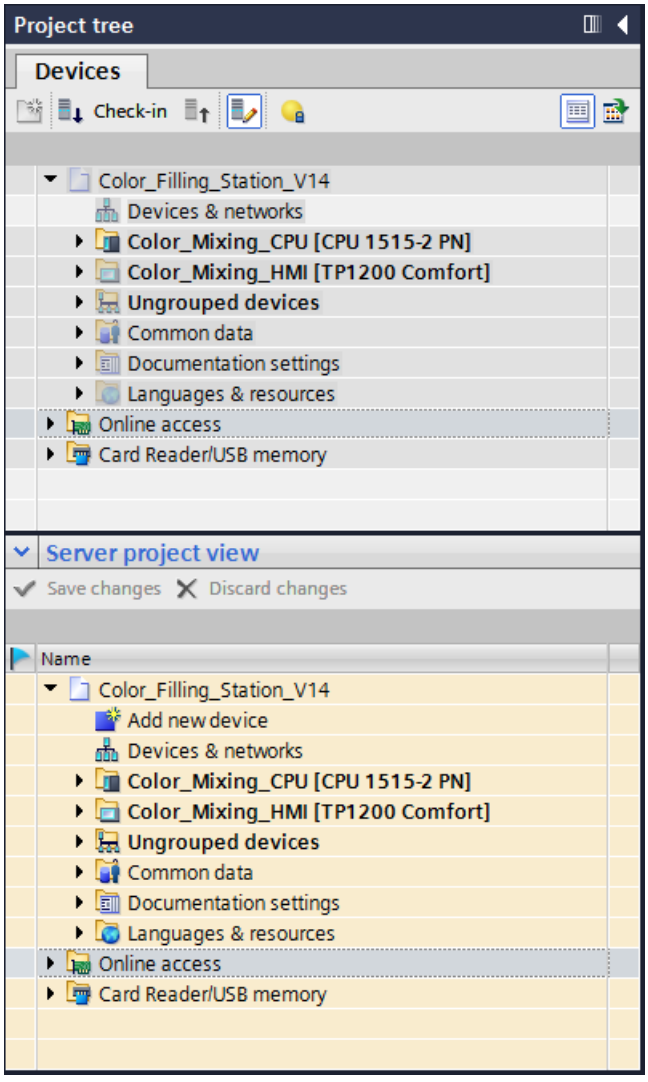

Actions in the server project view

You can perform the following actions in the server project view:

- The server project view lets you work directly in the server project.
- All existing objects can be edited in the server project view, even those that could not be edited in the local session.
- In the server project view you can work with the functionality of Inter Project Engineering (IPE) and exchange data between CPU and HMI via a PLC proxy.
- Shared working on one CPU is also supported during shared commissioning.
- Blocks or larger program parts from a CPU should only be uploaded in the server project view.
- You can also make changes to the configuration with the help of external tools in the server project view. This function is not supported when working in the local sessions.

- Changes to user-defined documentation, such as adding new user texts or new HMI scripts, should only be made in the server project view.
- The objects that are to be edited must also be marked for editing in the server project view so that other editors see that these objects are being changed.
- You can save or discard the changes made in the server project view with the "Save changes" and "Discard changes" buttons.
- Any changes you have made will be saved directly in the server project with the save step.

Note

Restrictions when working in the server project view

The following conditions apply as soon as the server project view is open:

- The local session is locked to prevent editing.
- The server is locked for the other editors.
- The icon for the server status is shown in yellow in the toolbar.
- If the changes you have made cannot be saved when working in the server project view, you should additionally select and deselect another object so that the "Save changes" button is activated and saving is possible.

See also

Opening and closing the server project view (Page 168)

Editing objects in the server project view (Page 169)

2.18.2 Opening and closing the server project view

Introduction

The server project view lets you edit all objects that are available in the server project.

You have the following options to open the server project view:

- Using the icon in the multiuser toolbar
- Using the link displayed in the banner texts
- Using the button in the multiuser editor in the "Check-in" view

You will receive a message if the server project view cannot be opened because the server is currently in use by another editor and locked.

Displayed contents in the server project view

The server project view always shows the latest version of the multiuser server project.

If the server project view is opened in the "Check-in" view in the multiuser editor, it also shows the changes made in the local session in addition to the current content of the server project.

This means it offers a preview of how the server project will look after check-in of the changes made in the local session.

Opening the server project view

To open the server project view, follow these steps:

1. Open the required local session.
2. In the multiuser toolbar, click the "Open/close server project view" icon.
   Or you can click "Check-in" in the multiuser toolbar and then click "Open server project view before check-in" in the multiuser editor that opens.
   Alternatively, you can also click on the link in a displayed banner text.

Closing the server project view

To close the server project view, follow these steps:

1. In the multiuser toolbar, click the "Open/close server project view" icon once again.
2. If you have made changes in the server project view, you will be prompted as to whether you want to save or discard the changes.
3. Click "Yes" to save your changes or "No" to discard them.
   Alternatively, you can click on the corresponding button in the server project view before you close it to save or discard your changes.

Result

The server project view is closed.

The local session is opened once again for editing.

The server is released again; the server status is displayed in green.

See also

Introduction to working in the server project view (Page 166)

Editing objects in the server project view (Page 169)

2.18.3 Editing objects in the server project view

Introduction

In the context of Multiuser Engineering you can also edit objects directly in the server project view .

The server project view lets you edit all objects in the server project view, even those that must not be edited in the local session.

Working in the server project view is an exclusive working directly in the multiuser server project. The multiuser server is locked for other editors while you are working in the server project view. The associated local session cannot be edited during the working in the server project view.

The objects you want to check in also have to be selected in the server project view. This enables you to ensure that other editors in your local session can see that you are currently editing these objects. Marking conflicts can hereby be avoided.

You can also use the "Autoselect" function in the server project view or alternatively select the objects to be edited manually.

Note

Marking of identical objects in the local session and in the server project view

Keep in mind that a marking conflict is displayed with red flag if you mark an object more than once, even if both markings belong to the same editor.

Editing objects in the server project view

You can, for example, perform the following actions in the server project view:

- Adding new objects that did not exist previously in the server project.
- Add or change hardware configuration and connections.
- Perform version changes on the instructions used.
- Working with Inter Project Engineering (IPE).
- Editing all objects regardless of whether these are marked or not.
- Renaming, moving and deleting objects.
- Establishing an online connection to a CPU and downloading or uploading objects.
- Executing all functions of the TIA Portal supported by Multiuser Engineering.

Procedure for editing objects in the server project view

To edit objects in the server project view, follow these steps:

1. Open the required local session.
2. In the multiuser toolbar, click the "Open/close server project view" icon.
   The server project view is then shown against a yellow background below the local session in the project tree.
   The server project view is closed again the next time you click this icon.
3. Select the objects you want to edit in the server project view by clicking on the flag in front of them, or use the "Autoselect" function by opening and editing the objects directly in the editor.
4. Make your required changes.
5. Save your changes by clicking the "Save changes" button.
   Or click "Discard changes" if you do not want to apply your changes to the server project.

6. Enter a comment on the change history to document the changes you have made in the server project view.
7. If you want to apply the changes from the server project view to the local session, click the "Refresh local session" button after saving the changes in the local session.

Result

The changes made in the server project view are saved in the server project.

- The server project view is closed and the local session is once again released for editing.
- The markings are removed in the server project view.
- The local session is **not** automatically refreshed with the new contents of the server project. The symbol for the server shows that a more recent version of the server project is available.

Server project view is open and the server connection is interrupted

If you have already opened the server project view and the connection to the multiuser server is subsequently interrupted, you receive a corresponding error message.

You then have the following options:

- Click the "Reconnect" button to establish a new connection to the associated multiuser server.
- Or click the "Discard" button to discard the changes made in the server project view.

If you cannot reestablish the interrupted server connection and you discard your changes, the local session is opened offline with an error message. The local session can also be opened when the associated server is currently not available.

You can continue working offline in the local session with the following restrictions:

- Selections made by other users are not visible in your local offline session; no yellow or red flags and no outdated objects are displayed.
- Changes you are making in the local offline session cannot be checked in until a server connection is available once again.

See also

Introduction to working in the server project view (Page 166)

Opening and closing the server project view (Page 168)

2.18.4 Renaming objects in the server project view

Introduction

When you work with Multiuser Engineering , object can be renamed.

Rules for renaming objects

The following rules apply to renaming objects:

- You should avoid renaming objects in the local session as it can lead to naming conflicts during check-in.
- The renaming of objects should always be performed in the server project view.
- When objects are renamed in the server project view, the objects to be renamed must have no markings within local sessions.
- Name conflicts may occur due to the renaming. During renaming in the project, make sure that you do not again use an already used name.
- To avoid unintended references of objects, conflicts that occur during renaming are not automatically corrected.
- The renaming of devices does not create any conflicts for objects within a device, as devices are not identified by their name but by their ID.
- However the renaming of devices may create conflicts with any existing references between devices (PLC and HMI).
- After objects have been renamed, the other editors should refresh their local sessions.

Requirement

The connection to the multiuser server exists and the server is available.

No other local session to this server project is open.

Renaming objects

To rename objects in the server project view:

1. Open the required local session.
2. In the multiuser toolbar, click the "Open/close server project view" icon.
   The server project view is then shown against a yellow background below the local session in the project tree.
   The server project view is closed again the next time you click this icon.
3. Mark the required objects in the server project view and rename these.
4. Click the "Save changes" button to apply the changes in the multiuser server project.

Result

The changes made in the server project view are saved in the server project.

- The server project view is closed and the local session is once again released for editing.
- The markings are removed in the server project view.
- The local session is **not** automatically refreshed with the new contents of the server project.

Using Multiuser Commissioning 3

3.1 Introduction to Multiuser Commissioning

Introduction

In TIA Portal as of V15.1 you have the option of working with Multiuser Commissioning as part of the "Multiuser Engineering" product.

After you have created and edited multiuser projects with multiple users in Multiuser Engineering , commissioning can be executed just as conveniently and also in the team with Multiuser Commissioning .

This approach significantly reduces the commissioning times for your projects. You can use your plant faster and start production sooner.

Working with Multiuser Commissioning

This is how Multiuser Commissioning works in TIA Portal:

1. First you create and edit your projects in the engineering phase with Multiuser Engineering. for more details, see: Overview of the mode of operation with Multiuser Engineering (Page 16)
2. As soon as your projects are ready for commissioning, you continue working with Multiuser Commissioning .
   - As with commissioning, project management is located either on a local or external server.
   - Different users work in local sessions, based on the projects managed by the server.
   - Users can also work independently from each other in local sessions also during commissioning.
   - The changes from the local sessions are first always downloaded to the common CPU and then after successful download, transferred to the server project.
   - Downloaded changes from other editors are displayed and can be conveniently applied.
   - The server project is already synchronized on the multiuser server. This means consistent versions of the server project are always downloaded to the CPU. Time-consuming synchronization of different project versions on the CPU therefore becomes obsolete.

Note

Working efficiently with Multiuser Commissioning

When working in parallel in multiple local sessions, you should correct errors that occur at an early stage so that the distributed work on a shared project stays efficient.

Therefore, it is recommended to first compile pending changes before download and check-in in the multiuser editor. Several compiling options are available for this. If compiling errors are indicated, you should correct them before download and check-in.

Compiling pending changed before check-in ensures that you are checking in and transferring an error-free project and download it to the CPU.

By downloading a project version that is always synchronized in the server project as well as on the CPU and is therefore consistent, fewer errors are being forwarded to other editors.

By working on different objects, such as different tags, blocks, PLC tags, interrupts, screens, text and graphic lists, technology objects etc., on one or multiple devices within a multiuser project at the same time, you can make your commissioning times more effective and reduce them significantly.

See also

Overview of the mode of operation with Multiuser Engineering (Page 16)

3.2 Overview of the mode of operation with Multiuser Commissioning

Introduction

If you want to work with multiple users simultaneously in multiuser projects during the commissioning phase, specific procedures should be observed.

Below you will find an overview of the general mode of operation with Multiuser Commissioning.

Requirements for working with Multiuser Commissioning

You have already executed the following steps with Multiuser Engineering in the engineering phase:

1. Installing the multiuser server.
2. Create suitable multiuser project on the server.
3. Create local sessions for all editors.
4. Gradual creation of the automation task in the different local versions and checking in all changes to the server project.
5. Updating the local sessions with the server project.
6. Continue operation until the multiuser project is ready for commissioning and the engineering phase can be completed.

Working with Multiuser Commissioning

Please observe the following notes for working in multiuser projects:

- Before they can be used in the commissioning phase with Multiuser Commissioning , the projects should have already been edited with Multiuser Engineering and contain the entire hardware configuration, the required blocks, connections and global settings.
  For more information on this topic, refer to: Requirements for multiuser projects (Page 34)
- Avoid working on the same objects simultaneously by consulting beforehand with the other editors.
- All global settings, such as changes within the device configuration, must be made in the server project view (i.e. in the central server project).
- All objects of a multiuser project can be edited in the server project view. You can also edit objects that do not support the Multiuser Engineering functionality.
- All changes checked into the server project from the local sessions must be compiled to always keep a consistent version on the server project.
- After download to the CPU, the multiuser server project is automatically updated. A new revision of the server project is created after each download.

Overview of the mode of operation with Multiuser Commissioning

To use the Multiuser Commissioning functionality, carry out the following steps:

Action	Example
1. Processing in the engineering phase is completed The processing of the various configuration tasks in the local sessions has been completed. **Result:** All changes have been compiled and are checked in to the server project. The local sessions and the server project are synchronized. There are no more markings in the local sessions. The multiuser server project was downloaded to the CPU. This means the version of the server project and the version of the multiuser project on the CPU are consistent. **Note**: This completes the engineering phase.	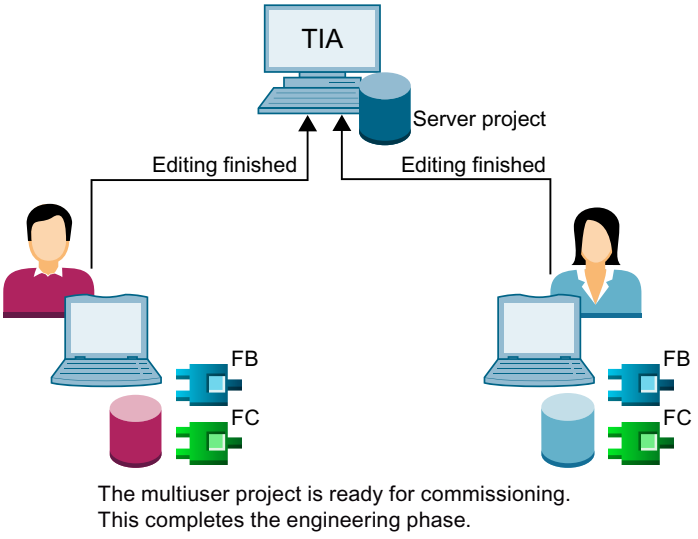

Action	Example
2. Working in the commissioning phase You now switch to "commissioning mode" in the TIA Portal and start with the commissioning phase. The switch is always made in the Administration Tool. The procedure is as follows: • You edit the local session in commissioning mode as usual. • All changes from the local sessions are compiled and checked in to the server project. • The server project and the local session are updated automatically after each download to the CPU. Changes made by other users will then become visible in the local session. • A new revision of the server project is created after each download to the CPU. • Continue working in commissioning mode until commissioning is complete. • If required, you can export the multiuser server project as single-user project and continue working and managing as usual in the TIA Portal. **Result:** It is ensured in commissioning mode that the online project in the CPU and the offline project in the multiuser server are consistent. This makes for easier commissioning because there is no need for synchronization. However, if desired, inconsistencies can be permitted during the download, for example, when you download to the CPU directly from the local session without prior check-in.	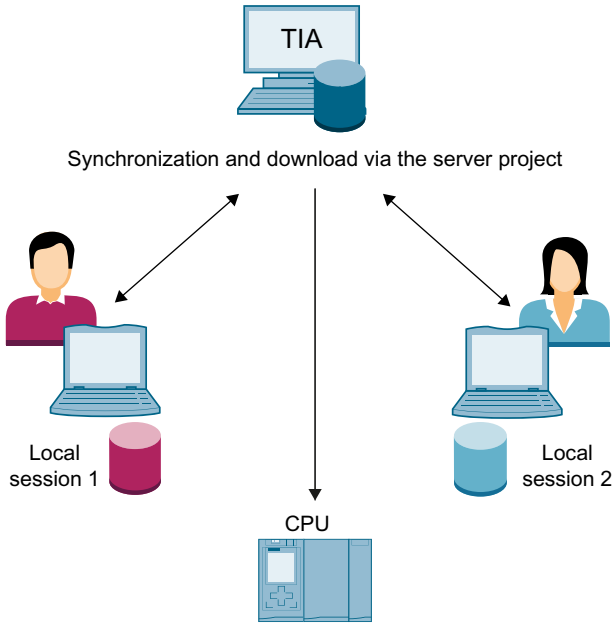

See also

Requirements for multiuser projects (Page 34)

3.3 Settings for working with Multiuser Commissioning

Introduction

The general settings for Multiuser Engineering also apply to working with Multiuser Commissioning . You can make the following multiuser-specific settings in the TIA Portal under "Options > Settings" in the "Multiuser" tab:

- General
- Multiuser server connection
- Multiuser network profiles and compression
- Searching in the project
- Storage settings
- Compilation settings

For more details, see: Settings for working with Multiuser Engineering (Page 29)

Settings for Multiuser Commissioning

As of TIA Portal V15.1, you can switch between engineering mode and commissioning mode when working with Multiuser Engineering .

The switch is always made in the Administration Tool.

For more details, see: Managing a multiuser server with the graphic Administration Tool (Page 62)

Open the Administration Tool and add the server you are using.

Select the desired multiuser project.

The "Multiuser Commissioning" button has been added to the Administration Tool toolbar.

As soon as this button is pressed, you switch to commissioning mode. The symbol for commissioning mode is also displayed in the TIA Portal project tree as overlay symbol in front of the project name.

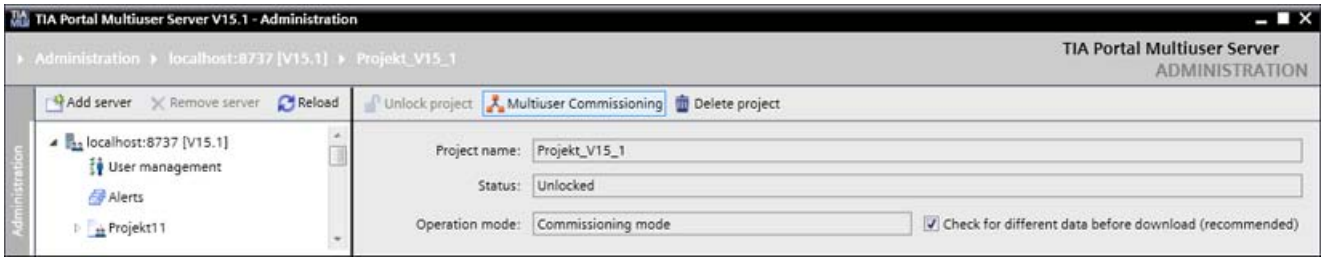

In commissioning mode, the server project is always updated after download to the CPU.

The changes from the local sessions are compiled before they are downloaded to the CPU. If the download process is successful, the changes are transferred to the server project. A new revision is then created on the multiuser server and the local session is updated each time.

If the multiuser server is not reachable, you receive a corresponding error message and commissioning mode is exited.

In this case you can download directly to the CPU from the local session in engineering mode and transfer your changes to the server project once the connection to the server has been established again.

Exiting commissioning mode

You exit commissioning mode again by deactivating the "Multiuser Commissioning" button.

Click on the Commissioning mode button again to deactivate it. The deactivated Commissioning mode button means the engineering mode is active.

The small overlay symbol in front of the project name is no longer displayed in engineering mode.

Automatic interruption of commissioning mode

Commissioning mode is automatically interrupted in the following cases:

- Whenever errors occur during compilation of the changes.
- Whenever errors occur during subsequent download to the CPU.
- Whenever errors occur during the transfer of changes downloaded to the CPU to the server project.
- Whenever network interrupts occur.
- When working offline, when you are working without being connected to the server project in the local session.
- When the server project is locked by another editor.

Whenever errors occur you receive an error message with information on the type of error and possible remedy.

3.4 Requirements for working with Multiuser Commissioning

Introduction

If you are commissioning a project with Multiuser Commissioning as part of a team, it is very important for successful cooperation that all editors of the project are familiar with the following rules.

Setting the commissioning mode

The commissioning mode is set in the Server Administration Tool . The setting is usually made by the administrator and applies to all existing local sessions of a server project. This means that all local sessions are either working in engineering mode or in commissioning mode .

For more details, see: Enabling and disabling commissioning mode (Page 182)

Working with the multiuser server in commissioning mode

As soon as you start downloading to the CPU in commissioning mode , an attempt is made to establish a connection to the multiuser server to update the local session in the multiuser server project as well. If the multiuser server cannot be reached at this time, a dialog opens in which you will see a message to that effect.

You can then switch to engineering mode by clicking the corresponding option in the dialog and download the local session to the CPU in engineering mode.

Multiuser server not reachable in commissioning mode

When you are working in commissioning mode and the multiuser server cannot be reached any longer, you will receive a message to that effect. The dialog that opens offers the following options:

- Attempt to reestablish the connection to the multiuser server.
- Switch to engineering mode and download from the local session to the CPU without updating the server project.
- Canceling the action.

Each option opens an additional dialog with more options for selection and cancellation.

Working "offline" in commissioning mode

Even when you are "working offline" with Multiuser Commissioning, which means without being connected to the multiuser server, you can download from the local session to the CPU.

In this case, a dialog opens in which you can switch to engineering mode and download the local session to the CPU without updating the server project. You can also cancel the process.

Automatic interruption of commissioning mode

Commissioning mode is automatically interrupted by the system in the following cases:

- Whenever errors occur during the transfer of changes to the server project.
- Whenever errors occur during compilation of the changes.
- Whenever network interrupts occur.
- Whenever errors occur during download to the CPU.
- When working offline, when you are working without being connected to the server project in the local session.
- When the server project is locked by another editor.

Whenever errors occur you receive an error message with information on the type of error and possible remedy.

3.5 Enabling and disabling commissioning mode

Introduction

As of TIA Portal V15.1, you can switch between "Engineering mode" and "Commissioning mode" when working with Multiuser Engineering .

The default setting is engineering mode.

The same requirements apply for working in "commissioning mode" as for working in "engineering mode".

In "engineering mode" , you edit your multiuser projects as usual and configure your automation tasks at the same time in the different local sessions.

As soon as your multiuser project is ready for commissioning, you switch to "commissioning mode" .

The following functions are available in "commissioning mode":

- The changes from the local sessions are compiled and downloaded to the CPU.
- The changes are then automatically transferred to the server project and the local session is updated.
- A new revision is created on the server after each download.
- This ensures that the project versions in the CPU and on the multiuser server are consistent.

Note

If the multiuser server is not reachable or other errors have occurred during downloading, you receive an error message and "commissioning mode" is exited.

In this case you can download directly to the CPU from the local session in "engineering mode" and transfer your changes to the server project once the connection to the server has been established again.

Enabling and disabling "commissioning mode"

"Commissioning mode" is enabled and disabled in the Administration Tool. You always set the mode for all local sessions of a server project. All local sessions can either be in engineering mode or in commissioning mode.

See also: Managing multiuser projects and local sessions (Page 71)

See also: Managing a multiuser server with the graphic Administration Tool (Page 62)

Proceed as follows:

1. Open the graphic tool for the multiuser server administration.
   - To open the start menu, click the command "Start" > "All Programs" > "Siemens Automation".
   - Click on the "TIA Portal Multiuser" > "Multiuser Server V<x.y> Administration" entry to open the Administration Tool.
2. Under "Administration", you can find a list of the configured server connections as soon as you have set up a connection to the multiuser server with "Add server connection". On the right side, you receive additional information about the objects selected on the left in a detail view.
3. On the left side, select the server and the project for which you want to set up commissioning mode.
4. On the right side, click the "Multiuser Commissioning" button in the toolbar.
   - As soon as the button is active, it is given a blue background and the project selected on the left side receives the commissioning symbol as overlay.
   - With commissioning mode, the "Check for different data before download (recommended)" option is also activated.

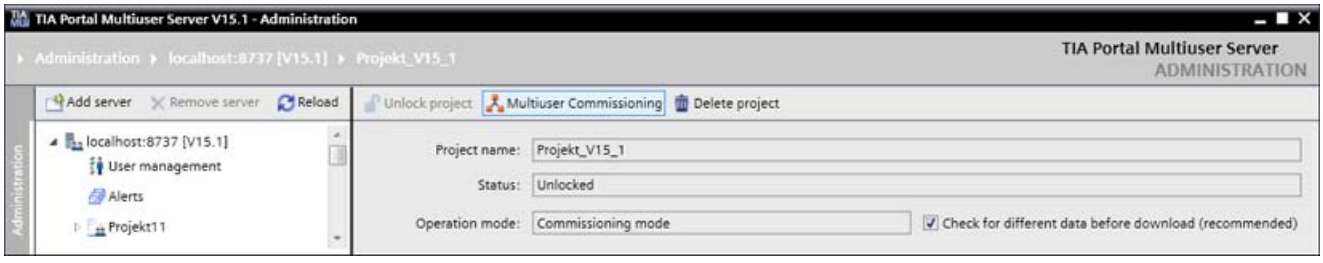

5. Click the "commissioning mode" button in the toolbar again to exit "commissioning mode".
   - As soon as the button is deactivated, it has no color and no overlay symbol is displayed for the project selected on the left side.

Result

The "commissioning mode" is switched on by clicking the "Multiuser Commissioning" button and switched off by clicking it again.

See also

Managing a multiuser server with the graphic Administration Tool (Page 62)

Managing multiuser projects and local sessions (Page 71)

Commissioning projects with PLC synchronization 4

4.1 Basics of PLC synchronization

Introduction

You have the option to perform PLC synchronization of projects as part of Team Engineering . Several editors can access one CPU at the same time with up to five Engineering Systems (ES) in this process.

A special advantage is that parts of a master project can be edited independently and offline at the same time during the commissioning phase. The changes of the other editors are displayed in the dialog "Software synchronization" during loading and synchronized automatically, if possible.

Depending on the firmware version of the CPU in use, certain online functions can also be executed at the same time from several Engineering Systems on the shared CPU, for example:

- Monitoring blocks on the CPU
- Modifying and forcing blocks on the CPU
- Trace functions

The following online functions cannot be executed at the same time:

- Loading: Only one ES can load to the CPU.

You can also find additional information on the topic of Team Engineering at Service and support in Siemens Industry online support (http://support.automation.siemens.com/WW/view/en/82142829).

Creating a master project

A "master project" structured according to the specified rules (Page 194), which already contains the complete configured hardware configuration with all required tags and blocks, is used as the basis for PLC synchronization during shared commissioning. This project is loaded in the jointly used CPU and then distributed as "master project" by means of project copies to up to five participating Engineering Systems.

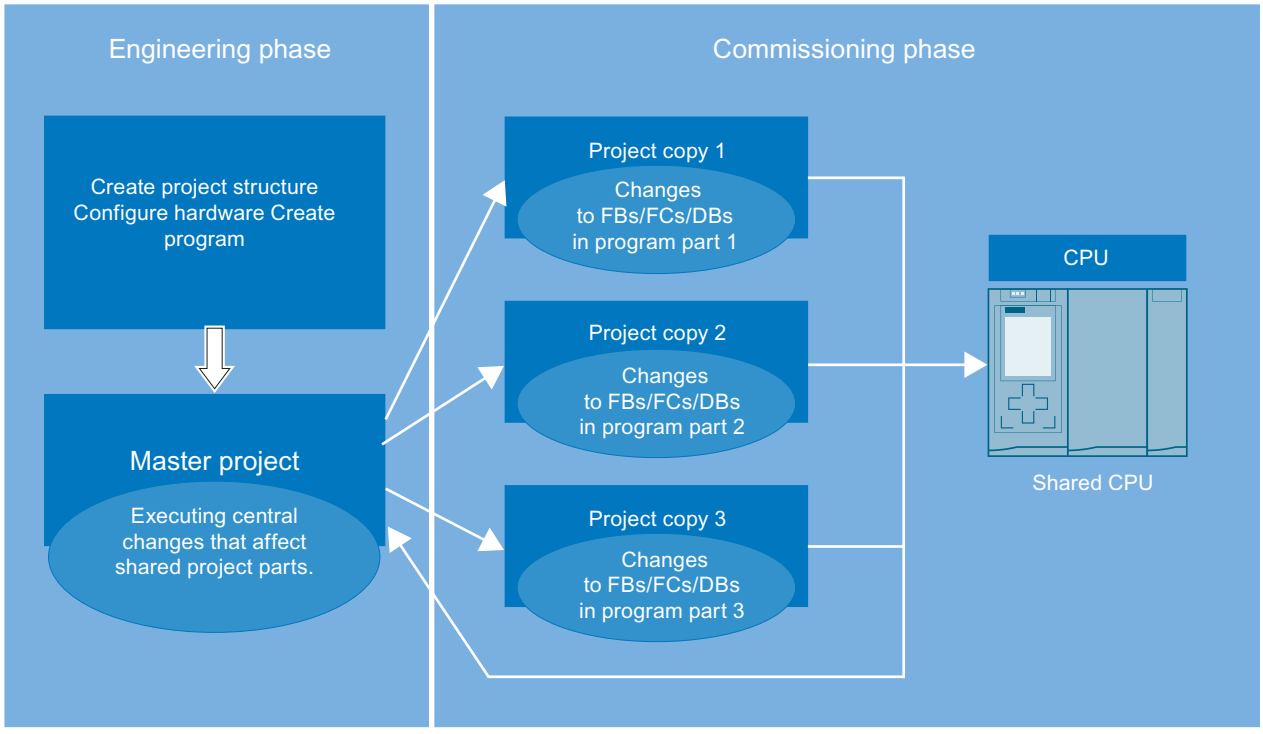

Working with project copies

Each Engineering System only processes the parts it was assigned within the specific project copy. This is especially important to avoid conflicts during loading and unintentional overwriting of blocks that were already modified by other Engineering Systems.

Each ES loads its parts to the shared CPU after editing. The TIA Portal helps you during download to the CPU. The changes made since the last download are displayed in the dialog "Software synchronization before loading to a device" (hereafter referred to as "synchronization dialog"), by means of an online-offline comparison. You are provided recommendations during download to the CPU about how you can synchronize the offline data you have edited with the existing online data.

If there are "competing changes", you may have to synchronize them manually.

Competing changes are:

- when different editors edit the same block in different project copies at the same time.
- when PLC tags (I, Q, M, T, C) are added, changed or deleted in the project copies.

- when blocks that have relations to the hardware configuration are edited in the project copies.
- when F blocks are edited in the project copies.

Each editor loads their changes only after all conflicts have been removed.

The procedure for editing and loading of project copies can be repeated as often as necessary by the individual engineering systems until PLC synchronization during shared commissioning is complete.

Integrating project copies into the master project once again

The individual project copies must once again be integrated into the master project at the end of PLC synchronization. This way you also back up the data that only exist in the offline projects and have not been downloaded to the CPU. These include, for example, the project languages and text lists created in the project copies.

To do this, open the master project and one project copy as reference object. You determine the differences between the two projects by performing an offline/offline comparison using the comparison editor. Copy the edited objects from the respective project copy to the master project. Repeat this step for all project copies.

When the shared commissioning with PLC synchronization is completed, you therefore have an executable master project with all created project data.

The integration of the project data into the master project is required even if you determine during the commissioning phase that the shared central elements need to be expanded or corrected, for example, the PLC tag table needs to be expanded, or a new hardware component needs to be added.

Notes on compatibility

The following compatibility rules apply to PLC synchronization performed during shared commissioning:

- If an Engineering System with TIA Portal V13 or later is online on the CPU and performs a download, a second ES cannot establish an online connection, regardless of the version.
- If an Engineering System with TIA Portal < V13 is online on the CPU, a second ES cannot establish an online connection, regardless of the version.

Team Engineering

As part of Team Engineering , you have the option to perform shared commissioning of projects. Several editors can access one CPU in parallel and at the same time with multiple Engineering Systems (ES) in this process.

PLC synchronization

In PLC synchronization, multiple engineering systems work together and in parallel on one CPU as part of team engineering.

Master project

The master project is the basic project for PLC synchronization. It is a project structured according to the specified rules which already contains the fully configured hardware configuration with all required tags and blocks. This project is loaded into the jointly used CPU and then distributed as "master project" by means of project copies to the participating Engineering Systems. Each Engineering System only processes the parts it was assigned within the specific project copy. The project copies are then integrated into the master project once again.

Project copy

Project copies are created from structured master projects when working with team engineering and distributed to the participating Engineering Systems for editing. Each engineering system only processes the parts it was assigned within the specific project copy as part of PLC synchronization. The individual project copies are once again integrated into the master project at the end of PLC synchronization.

See also

Requirements for PLC synchronization (Page 189)

Procedure for PLC synchronization (Page 190)

4.2 Requirements for PLC synchronization

Software and hardware requirements

For shared commissioning of projects, the minimum software and hardware requirements for the installation of TIA Portal V13 or later must be met.

The following requirements also apply:

Software:

- You have installed TIA Portal V13 or later and the "SIMATIC STEP 7 Professional" software package on the involved Engineering Systems.
- The same software version must be installed on all Engineering Systems.
- The master project must have been created with a project of V13 or later, which has been updated to the version of the TIA Portal used in the engineering system.
- After switching from a lower to a higher TIA Portal version, all involved engineering systems must use this version.
- After switching from a lower to a higher TIA Portal version, the master project must centrally adapted and new project copies must be transferred to the engineering systems involved.

Hardware:

- You have access to a fully configured CPU S7-1500 as of firmware version V1.5.
- The involved Engineering Systems can establish an online connection to this CPU.

Requirements for shared, parallel working on the project

The following requirements are in effect for shared commissioning of projects:

- You have created a project with complete hardware configuration and fully programmed user program that can be commissioned.
- The project was downloaded to the CPU and defined as "master project".
- Multiple Engineering Systems have access to this CPU and can establish an online connection to this CPU.
- You are familiar with the defined rules and procedures for working together on one CPU.

Note on compatibility mode

The functions for PLC synchronization are not available in compatibility mode.

See also

Basics of PLC synchronization (Page 185)

Procedure for PLC synchronization (Page 190)

Rules for PLC synchronization (Page 194)

4.3 Procedure for PLC synchronization

Introduction

If you are performing shared commissioning of a project as part of team engineering, it is very important that all editors of the project observe a defined procedure.

Only when the specified procedure is observed are changes and corrections in the project automatically synchronized and applied, and changes of individual editors not unintentionally overwritten during loading or even lost in case of competing changes.

Procedure for creating the master project

The master project is created in the following steps:

1. Create a master project that includes the entire project structure.
2. Fully configure the hardware for the master project.
3. Define a language as project language, which must also be used exclusively by all participating engineering systems.
4. Create all tags and blocks that are required in the master project.
5. Create your own folders and groups for the blocks to be edited by the individual engineering systems.
6. Create a fully programmed and executable user program.
7. Download the master project to the shared CPU.
8. Save the master project following every download.

You should also note the Rules for PLC synchronization (Page 194).

Procedure for PLC synchronization

PLC synchronization is performed with the following steps:

1. Download the master project to the shared CPU.
2. Create copies of the master project and distribute them to the editors involved in the project on the associated engineering systems.
3. Inform all editors who is to edit the specific parts of the project copies and who may download to the CPU.
4. Have the project copies edited in the individual engineering systems.
5. After editing, the individual engineering systems download the changes one after the other to the CPU.

6. All changes are automatically detected during loading by means of an online-offline comparison. The displayed synchronization dialog offer you recommendations for synchronizing the modified data when possible. You may first have to download blocks that were edited by other editors to your ES before you can download your changes to the CPU. This may be needed to update the corrections of the other editors that have already been downloaded to the CPU in your project copy.
   The following synchronization options are available:
   - Download block: There are edited blocks on the CPU that must be updated in your project copy.
   - Download block: There are new blocks on the CPU that must be downloaded to your project copy.
   - Blocks with competing changes: System-supported synchronization is not possible in this case; the conflict must be resolved manually.
7. Manually resolve any conflicts that may occur due to competing changes or changes to central objects.
8. Download your project copy to the CPU when all conflicts have been removed.
9. Save your project copy following every download.
10. Repeat editing of project copies and download to the CPU as often as necessary until PLC synchronization is complete.
11. Integrate the completely edited project copies into the master project once again in order to also back up the project data that only exist offline.

You should also observe the Rules for PLC synchronization (Page 194).

Procedure for manual synchronization of competing changes

Manual synchronization of competing changes is performed in the following steps:

1. Start the comparison editor to manually resolve a conflict displayed in the synchronization dialog during download to the CPU. Select the shared CPU in the project tree and select the "Compare > Offline/Online" command in the shortcut menu. You can define certain actions depending on the status of the objects. Note, however, that you can only perform actions in one direction during synchronizing.
2. First, select the action "Upload from device" for all blocks that were changed by other editors on the CPU and which you want to apply.
3. If necessary, perform a detailed comparison of the blocks to identify differences between your offline blocks and the online blocks loaded on the CPU.
4. Manually correct the competing changes to the blocks.
5. Then download the affected blocks to the CPU using the "Continue without synchronization" command.
6. Save your project or project copy following every download.

You should also note the Rules for PLC synchronization (Page 194).

Note

To prevent online changes occurring again while an engineering system is being manually synchronized, no downloading to the shared CPU should be performed by another participating engineering system during the manual synchronizing.

Procedure for integrating project copies into the master project

You integrate project copies into the master project with the following steps:

1. Open the master project and the project copy to be integrated as a reference object.
2. Copy the program parts you have edited from the respective project copy to the master project and confirm overwriting the existing objects. You can also use the comparison editor to apply your program parts to the master project.
3. Save the master project and download it to the shared CPU with the "Continue without synchronization" command.
4. Save the master project following every download.

You should also note the Rules for PLC synchronization (Page 194).

Procedure for modifying central objects within the master project

Modification of central objects that have an effect on all program parts is performed with the following steps:

1. Stop further processing of the project copies.
2. Integrate all existing project copies one after the other into the master project as described above.
3. Download the master project to the shared CPU with the "Continue without synchronization" command.
4. Make the required modification on the shared objects such as in the hardware configuration, for example, or in the PLC tag table. The procedure is the same for modification of other central objects such as technology objects, F blocks, OBs, etc.
5. Download the master project into the CPU once again when modification is complete.
6. Save the master project following every download.
7. Distribute the updated project copies to the respective engineering systems for further processing.

You should also note the Rules for PLC synchronization (Page 194).

See also

Basics of PLC synchronization (Page 185)
Requirements for PLC synchronization (Page 189)
Rules for PLC synchronization (Page 194)

4.4 Rules for PLC synchronization

Introduction

If you are performing PLC synchronization of a project as part of team engineering, it is very important that all editors of the project observe predefined rules for successful cooperation.

Rules for the master project

The following rules are to be observed:

- Create a master project that is suitable for shared editing.
- Divide the user program into program parts that are independent of each other.
- Use "Groups" to visually separate the program parts from each other.
- Use a main OB and a central FC for each program part which calls the functions of the part program.
- If possible, create a separate PLC tag table for each part.
- In the master project, specify a project language which cannot be changed in the project copies.
- If you want to exchange data between the program parts, use the FC and FB interface parameters (IN, OUT, INOUT) or global data blocks.
- Use global data blocks to save the data for the individual program parts, no bit memories.
- Do not assign different names for blocks with identical addresses.
- Do not assign identical names for blocks with different addresses.

Example of the program structure in the master project

Here is an example of a team-capable, structured master project that is suitable for parallel editing during PLC synchronization.

The program parts to be edited by the individual Engineering Systems are:

- Program part 1: "Conveyer"
- Program part 2: "Drill"
- Program part 3: "FurtherProgramPart"

Each program part contains a "main FB", which calls the specific lower-lever functions for this program part.

Example: In the "Conveyer" program part, the "ConveyerMainFB" calls the "ConveyerStartupFC" function.

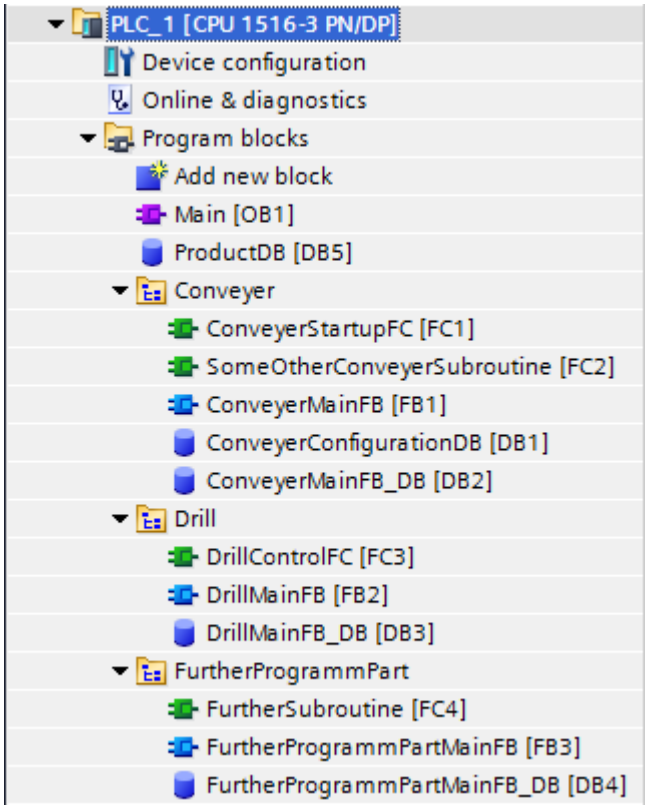

In team engineering, this subdivision within the project structure allows the parallel editing of individual program parts and the automatic synchronization of changes during the shared commissioning phase.

Rules for shared working on one CPU

Observe the following rules for shared working:

- Editor only edits the blocks they have been assigned in the project copy in the assigned groups.
- Only OBs, FBs, DBs, FCs and UDTs may be changed in the project copies, which means the following program elements must **not** be edited in the project copies:
  - Hardware configurations
  - PLC tag table
  - Technology objects
  - Text lists and project language
  - F blocks
- Global data blocks should be used in programming instead of bit memories.
- IEC timers and counters should be used during programming instead of SIMATIC timers and counters.
- The project language specified in the master project may not be changed in the project copies.

Rules for editing shared, central objects

Observe the following rules for shared central objects:

- Changes to shared central objects always have to be made by means of the master project. These objects include:
  - Hardware configurations
  - PLC tag table
  - Technology objects
  - Text lists and project language
  - F blocks
- Before the shared central objects can be modified in the master project, the various working versions within the project copies must first be integrated into the master project once again.
- The modification can then be made within the master project, and the master project can then be downloaded to the shared CPU.
- Once modification is complete, new project copies can be created and distributed to the Engineering Systems for further processing.

Rules for working with organization blocks

Observe the following rules for editing OBs:

- OBs can only be created in the master project. If you create new OBs in a project copy, they **cannot** be synchronized.
- OBs that already exist in the master project and that have already been downloaded to the CPU, can be modified in the respective project copies. These OBs are automatically synchronized during downloading.

Rules for downloading to the CPU

Observe the following rules for downloading:

- Editors can download the project copies one after the other to the CPU; only one Engineering System can download at any given time.
- The online connection of the involved Engineering Systems can be maintained when downloading the blocks (hardware may only be downloaded in the master project).
- Perform the synchronization that is offered automatically in the synchronization dialog during download. This ensures that changes from other editors are not unintentionally overwritten.
- Correct the name conflicts that are shown in the synchronization dialog, which you rename in your offline block before downloading to the CPU. A name conflict is indicated if you want to download to the CPU a block that already exists with identical name on the CPU.

- So-called "competing changes" must be corrected manually. Competing changes come about when two editors work on the same block on two Engineering Systems at the same time. The first editor can still download his or her changes to the CPU without any problems. However, if the second editor wants to download the same block with the changes the editor has made, the synchronization dialog displays a conflict that cannot be removed automatically. The previous changes made to this block would be undone by overwriting during download. The editors have to decide in this case which change is to be applied and which one is to be discarded. Alternatively, you can manually merge the changes with the help of the detailed comparison of the blocks in the comparison editor.
  Such changes should always be avoided by implementing a good project structure and corresponding agreements between the editors.
- Changes to central objects are displayed in the synchronization dialog but cannot be synchronized automatically. Changes to central objects and to the hardware configuration always have to be made by means of the master project.
- Save your project or project copy following every download.

Note

Creating and subsequent deleting of objects in the project copies

Please note that a software synchronization is also required before downloading when you create a new object in a project copy and then delete this again immediately afterwards.

The creation of an object causes a change in the internal data management which cannot be undone by the subsequent deleting of the object. As a result it is necessary to perform a synchronization before the next loading if the newly created object was deleted immediately thereafter and no change in the existing object is visible within the project copy.

Rules for online functions

The following rules are to be observed when using online functions:

Loading:

- Only one Engineering System can download data to the CPU at any given time.
- Other Engineering Systems can use additional online functions during the download, such as monitoring and controlling.

Monitoring and controlling:

- Up to ten Engineering Systems can monitor and control up to 16 different blocks simultaneously on the CPU (CPU 1517/1518). Further details can be found in the CPU's technical specifications.
- A specific code block can only be monitored and controlled by one Engineering System at a time.
- However, additional Engineering Systems can simultaneously monitor and control other code blocks.

NOTICE
Danger due to modifying an identical operand in parallel with different modify values in more than one watch table When working with more than one watch table, avoid modifying identical operands permanently multiple times with different modify values. If an identical operand is modified permanently with different modify values at the same time in different watch tables, all watch tables will display the last modified value, because the modify value assigned last will be used in this case.

Force

- Forcing can only be implemented on the CPU from one Engineering System at a time.
- The Engineering Systems that starts the forcing has taken over the force job exclusively. Although the other Engineering Systems receive the information that a force job is running, they cannot access this job to make changes. Use the command "Update forced operands" to update all operands and values currently being forced on the CPU in the open force table. All forced operands in the open force table are updated with the relevant values. A red "F" is displayed in the first column indicating which operands are being forced.
- As soon as the Engineering System to which the force job belongs ends the online connection, the force job can be applied by another ES that establishes an online connection to the CPU and has executed the "Update forced operands" command. This enables the "Force all" and "Stop forcing" buttons and you can select these functions.

Trace functions:

- Up to four Engineering Systems can perform trace functions simultaneously.
- The Engineering Systems that starts the tracing has taken over the trace job exclusively. Although the other Engineering Systems can see this job in the trace editor, they cannot access it.
- As soon as the Engineering System to which the trace job belongs closes the trace editor, the trace job can be taken over by another ES that re-opens the trace editor.

Testing with breakpoints:

The following rules apply as soon as a breakpoint is enabled by an engineering system of an S7-1500 CPU as of firmware version V2.5:

- Enabling of breakpoints by other engineering systems is blocked.
- Enabling of additional breakpoints on the executing engineering system is permitted even if the CPU is in "HOLD" mode after a breakpoint has been reached.
- "Download to device" cannot be executed by a different engineering system, and the program on the CPU can no longer be executed because the CPU is in "HOLD" mode when a breakpoint is reached. Program execution is not continued until the CPU is set from "HOLD" mode back to "RUN" mode by the executing engineering system. Other engineering systems cannot switch the mode directly to "RUN" after a breakpoint has been reached. The CPU must first be set to "STOP" mode and can only be switched to "RUN" afterwards.

- If a "Download to device" is already in progress, no breakpoints can be enabled during the download process.
- "Download to device" is also possible in "HOLD" mode when no breakpoints are enabled. During the download, the CPU is set to "STOP" mode.

NOTICE
Enabling breakpoints on an S7-1500 CPU as of firmware version V2.5 As soon as you enable a breakpoint on an S7-1500 CPU as of V2.5, you are notified that the output values for the connected output modules are kept in their respective state and (unlike with SIMATIC S7-300/S7-400) are neither reset nor set to the configured substitute values. In addition, the SIMATIC S7-1500 can no longer respond to input values from input modules in this state. This information is displayed only once during each online session.

See also

Basics of PLC synchronization (Page 185)

Requirements for PLC synchronization (Page 189)

Procedure for PLC synchronization (Page 190)

Exchanging data with Inter Project Engineering (IPE) 5

5.1 Basics of Inter Project Engineering (IPE)

Introduction to Inter Project Engineering (IPE)

The functionality of Inter Project Engineering, hereinafter referred to as IPE, is used to exchange the controller data in a source project between different projects with the help of a device proxy.

You can then transfer this data to other projects, for visualization in HMI, for example, and use it there for further configuration.

Inter Project Engineering gives you a simple way of providing controller data from a PLC programming across multiple projects for an HMI configuration.

Using the object "Device Proxy Data", you exchange controller data across multiple projects consistently and easily without redundant configuration overheads.

This makes it possible for HMI project engineers to work on a project without the hardware configuration having to be present in their project. This means they can work on the PLC project and on the HMI project at the same time.

You can exchange the following controller data between projects via the device proxy:

- Program blocks
- Technological objects
- PLC tags
- PLC alarms

The following data is automatically included in the exchange:

- Controller interfaces
- Configured communication processors and communication modules

This automatic data exchange ensures that the interfaces and the configured communication processors and modules are transferred along with the data you selected. This data is required to allow you to continue working consistently and without problems in your target project.

Cross-project data exchange

The following options are available for cross-project data exchange with Inter Project Engineering:

- Exchanging controller data via IPE file
- Exchanging controller data via project file

Exchanging controller data via IPE file

The figure below shows cross-project data exchange via an IPE file.

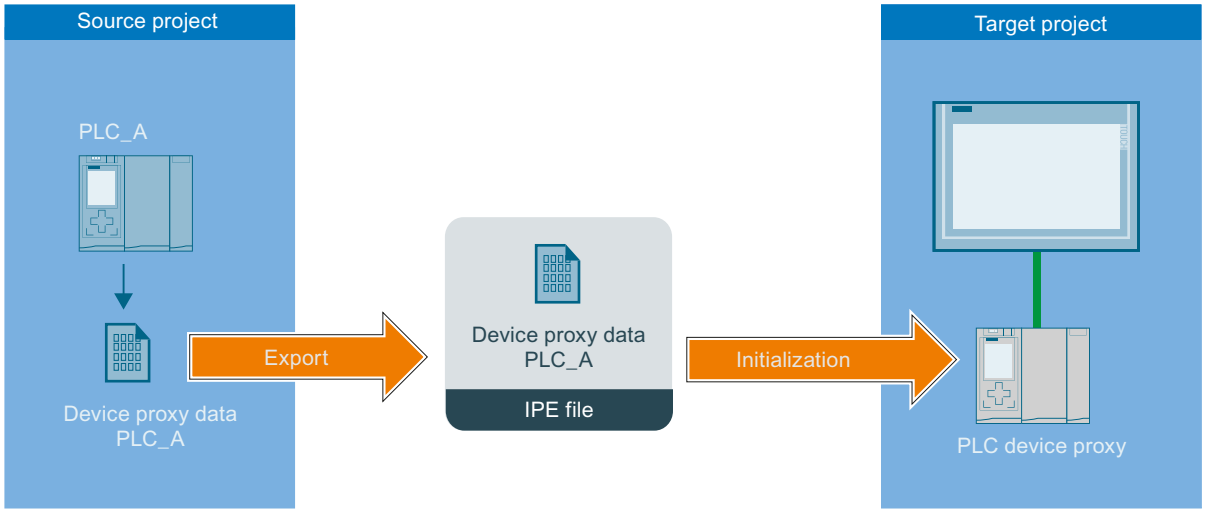

The PLC controller data from the source project is transferred to the PLC in the target project via the object "Device Proxy Data".

PLC data that is exchanged via an IPE file must originate from TIA Portal projects of V13 or higher.

Exchanging controller data via a project file

The figure below shows cross-project data exchange via a project file.

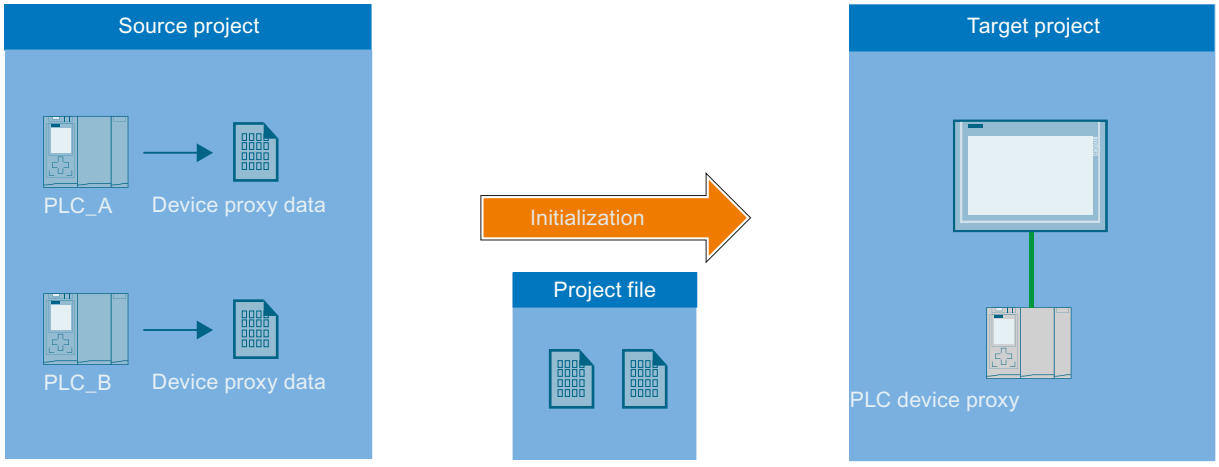

The PLC controller data from the source project is transferred to the PLC in the target project via a project file.

You can easily transfer and continue to use controller data from older TIA Portal projects as well as STEP 7 Classic projects that are not integrated in TIA Portal in your current TIA Portal projects.

Inter Project Engineering (IPE)

The functionality of Inter Project Engineering, also referred to as "IPE", is used to exchange the controller data in a source project between different projects with the help of the "Device proxy data" object.

IPE

The functionality of Inter Project Engineering, also referred to as "IPE", is used to exchange the controller data in a source project between different projects with the help of the "Device proxy data" object.

IPE file

The IPE file includes CPU controller data from a source project which is transferred with the "Device proxy data" object to the CPU in the target project.

Project file

The project file transfers CPU controller data from a source project to the CPU in the target project. You can conveniently transfer and continue to use controller data from older TIA Portal projects as well as STEP 7 Classic projects that are not integrated in TIA Portal in your current TIA Portal projects.

Device proxy data

The "Device proxy data" object supports consistent data exchange of CPU controller data across projects between a source project and a target project without redundant configuration.

Data exchange across projects with IPE

The Inter Project Engineering (IPE) functionality gives you the following options for consistent data exchange across projects:

- Exchanging PLC data via an IPE file
- Exchanging controller data via a project file

Both files transfer CPU controller data from a source project to a CPU in the target project.

See also

Requirements for Inter Project Engineering (IPE) (Page 204)

Overview for working with Inter Project Engineering (IPE) (Page 205)

5.2 Requirements for Inter Project Engineering (IPE)

Software and hardware requirements

To use the functionality of Inter Project Engineering, the minimum software and hardware requirements for the installation of TIA Portal V13 or higher must be met.

The following requirements also apply:

- You have installed TIA Portal V13 or higher and the "SIMATIC STEP 7 Professional" software package.
- You are using a CPU of the S7-1200/1500 or S7-300/400 series.
- The same software version must be installed on all involved controllers.

Requirements for IPE

The requirements for accepting controller data from projects are as follows:

- You have created a project with hardware configuration and PLC data that can be transferred to another project.
- You have made sure that the project is consistent before you transfer the controller data by means of the "Device proxy data" object.
- You are familiar with the defined procedures for Inter Project Engineering.

Notes on compatibility mode

The functions for Inter Project Engineering are not available in compatibility mode.

See also

Basics of Inter Project Engineering (IPE) (Page 201)

Overview for working with Inter Project Engineering (IPE) (Page 205)

5.3 Overview for working with Inter Project Engineering (IPE)

Exchanging controller data across projects

You can use the functionality of Inter Project Engineering to exchange existing controller data between different projects with the help of a "Device proxy" and to continue using this data in another project. The "Device proxy" provides you with a consistent and convenient way of exchanging data between projects without redundant configuration work.

Procedure for exchanging controller data via an IPE file

To exchange data via an IPE file, follow these steps:

1. Using TIA Portal, create a project with all required controller data and an executable user program.
2. Configure the hardware required for the project.
3. Compile the project to ensure consistency in your project.
4. In your project, create the object "Device Proxy Data" below the required CPU by clicking the corresponding command "Add device proxy data" in the "Device Proxy Data" folder.
   Result: The "Device Proxy Data" object has been created.
5. Select the required device proxy data in the project tree and double-click the object to open it in the editor.
6. Enter the required data for the device proxy data in the editor under "General" or apply the default setting. You can change the name and enter a comment for export of the controller data.
   In the "Define content" area of the editor select which of the existing controller data you wish to export as IPE file and click "Export device proxy data".
7. In the subsequent dialog, enter the name and storage location for the IPE file to be created and click "Save". You will be notified as soon as the export has been completed successfully.
   Result: The selected controller data has been exported as an IPE file.
8. In your target project, create a device proxy to be able to import the controller data contained in the IPE file. To do so, click the command "Add new device" and select the required device proxy in the dialog that opens.
9. Click the newly created device proxy and select the "Initialize device proxy" command from the shortcut menu.
10. Select the previously created IPE file to be used for the initialization, and confirm the selection in the following dialog with "OK". This starts the initialization.
    Result: The controller data from the source project contained in the IPE file is transferred to the device proxy in the target project.

After the successful exchange of controller data, you can continue with the configuration in your target project and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

Procedure for exchanging controller data via a project file

When exchanging controller data via a project file, device proxy data can either be predefined in the source project or it can be defined and selected from the target project.

Follow these steps to exchange data via a project file:

1. Using TIA Portal, create a project with all required controller data and an executable user program.
2. Configure the hardware required for the project.
3. Compile the project to ensure consistency in your project.
4. Open the TIA Portal project from which you want to import the controller data into the target project.
5. In your project, create the object "Device Proxy Data" below the required CPU by clicking the corresponding command "Add device proxy data" in the "Device Proxy Data" folder.
   Result: The "Device Proxy Data" object has been created.
6. Enter the required data for the device proxy data in the editor under "General" or apply the default setting. You can change the name and enter a comment for the device proxy data.
7. In the "Define content" area of the editor select which of the existing controller data you want to provide by means of the device proxy data.
8. Click "Save" to save the changes in the project.
   Result: You have successfully saved the selected controller data in your device proxy data.
9. In your target project, create a device proxy to import the controller data contained in the project file. To do so, click the command "Add new device" and select the required device proxy in the dialog that opens.
10. Click the newly created device proxy and select the "Initialize device proxy" command from the shortcut menu.
11. Select the previously created project file with the prepared device proxy data to be used for the initialization, and confirm the selection in the following dialog with "OK". This starts the initialization.
    Result: The controller data from the source project contained in the project file is transferred to the device proxy in the target project.

After the successful exchange of controller data, you can continue with the configuration in your target project and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

Procedure for updating already transferred controller data

System requirements for the update:

The following requirements must be met for updating already transferred controller data with the help of the already created "Device Proxy Data" object:

- You are using the same project for the update as for the previous transfer of the controller data.
- You are using the already created "Device Proxy Data" for transfer of the controller data.
- You have not made any changes to the hardware configuration or the communication interfaces in the meantime. You can add new hardware components.

Updating procedure:

1. Open the TIA Portal project from which you want to update the controller data in the target project.
2. Double-click the existing "Device Proxy Data" object and select under content which controller data you want to make available for the update.
3. Save the project or export the IPE file once again.
4. Open the required target project in the TIA Portal.
5. Click the existing device proxy in the target project and select the "Update device proxy" command from the shortcut menu.
6. Select the source which is to be used for the update, and confirm the selection.
7. In the following dialog, select under "Device" the CPU that was previously used as source, from which the controller data is to be re-imported.
8. Under "Defined device proxy data", select the required object and exit the dialog with "OK".
9. The update is then started and the selected device proxy data is transferred from the source project to the device proxy in the target project.

After the successful updating of the controller data, you can continue with the configuration in your target object and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

See also

Basics of Inter Project Engineering (IPE) (Page 201)

Requirements for Inter Project Engineering (IPE) (Page 204)

5.4 Creating "Device proxy data" in the source project

Introduction

In order to exchange controller data across projects you require the "Device proxy data" object that can transfer the controller data of the associated CPU as an IPE file in the source project.

Creating a "Device proxy data" object in the source project

Proceed as follows to create a new "Device proxy data" object:

1. Open the "Device proxy data" folder below the CPU to which you want to create a "Device proxy data".
2. Click the command "Add new device proxy data".

Result

The "Device proxy data" object has been created and is displayed in the project navigation below the associated CPU.

The created "Device proxy data" contains the controller data of the associated CPU.

5.5 Creating an IPE file by means of the "Device proxy data"

Opening the "Device proxy data" and defining the content for an IPE file

Proceed as follows to create an IPE file by means of a "Device proxy data":

1. Open the required "Device Proxy Data" with a double-click in the "Device Proxy Data" folder in the project tree below the CPU.
2. Enter the required settings for the "Device proxy data" object in the editor under "General" or apply the default setting. You can change the name and enter a comment for export of the controller data.
3. In the "Define content" area of the editor select which of the existing controller data of the CU you wish to export as an IPE file and click "Export device proxy data".
4. In the subsequent dialog, enter the name and storage location for the IPE file to be created and click "Save". You will be notified as soon as the export has been completed successfully.

Result

The selected controller data have been exported as an IPE file.

5.6 Using controller data from other projects with IPE

5.6.1 Using controller data from other projects in an HMI device

5.6.1.1 Basics on controller data

Introduction

In the source project of the PLC, you can select which controller data is to be stored as device-proxy data.

To this end, you create a device proxy data from a configured PLC.

Which controller data can be exchanged?

The following controller data can be exchanged via IPE:

- Data blocks
- Technology objects
- PLC tags
- PLC supervisions and PLC alarms

The following information is exchanged automatically:

- Interfaces of the PLC
- Configured communication processors and communication modules at the PLC

5.6.1.2 Initializing a device proxy via an IPE file

Initializing device proxy via an IPE file

How does initialization of a device proxy via an IPE file work?

You can exchange controller data between two or more projects using IPE files.

Due to their small data volume, you can send IPE files by e-mail, for example.

You can create multiple IPE files with different device proxy data for each PLC.

A device proxy can only be filled with one device proxy data record in the target project.

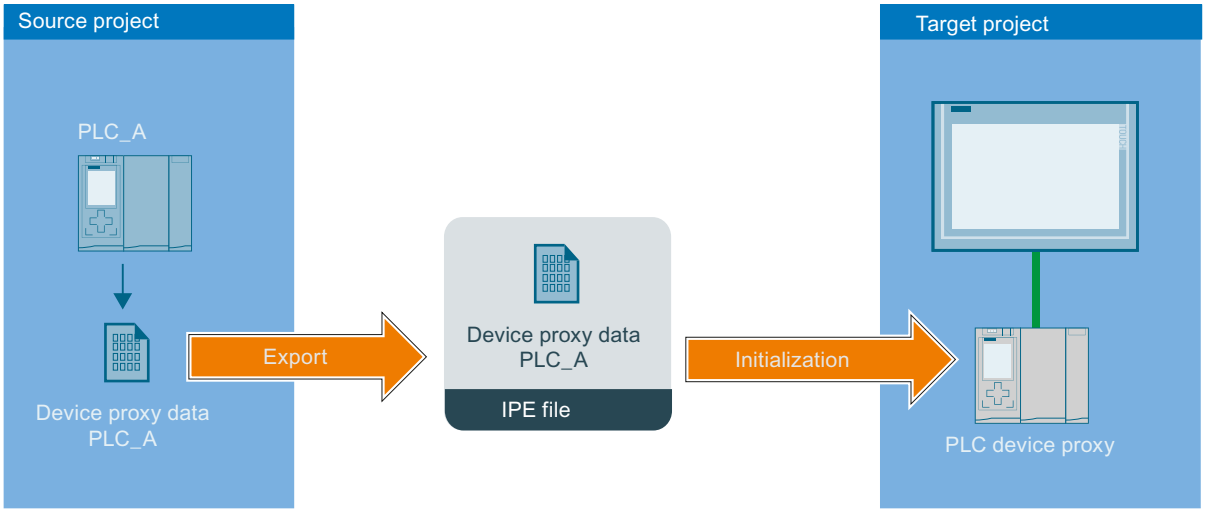

- An IPE file is exported from the source project.
  The IPE file contains device proxy data of the PLC "PLC_A".
- A device proxy is created in the target project.
- In the target project, the device proxy is then initialized with the device proxy data from the IPE file.
- Following initialization, all device proxy data from the IPE file is contained in the device proxy.
- If changes are made in the source project, the device proxy of PLC_A in the target project can be updated via the new IPE file.

Initializing a device proxy via an IPE file

Introduction

You initialize device proxy data using a dialog on the device proxy in the TIA Portal.

The IPE file contains the device proxy data from the source project.

Requirement

IPE file is available.

Procedure

1. Double-click "Add new device" in the project tree.
2. Select the device proxy under "Controller".

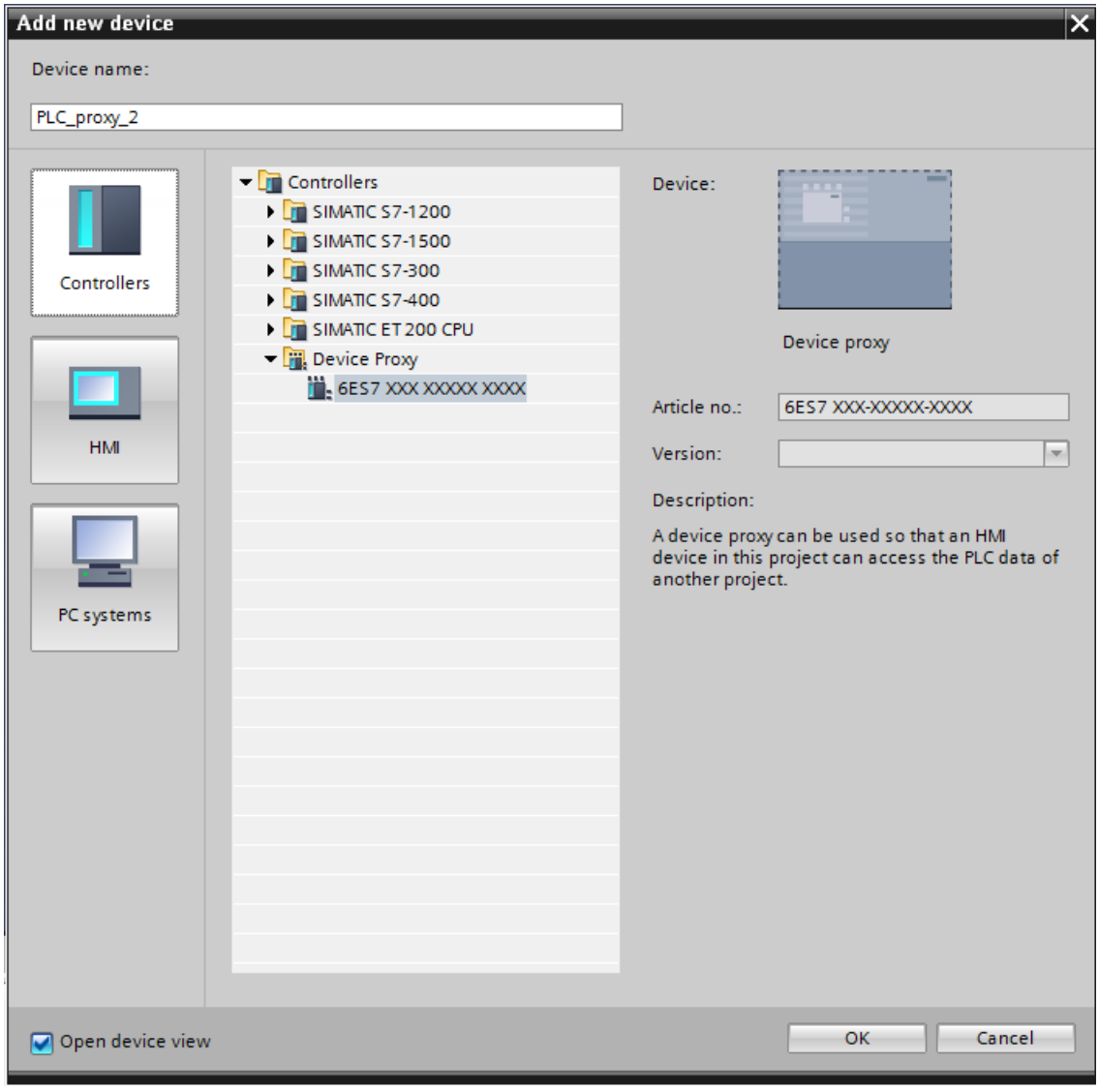

A new device is created in the "Devices & Networks" editor.

3. Select the device proxy in the project tree.

4. Click "Initialize device proxy" in the shortcut menu.

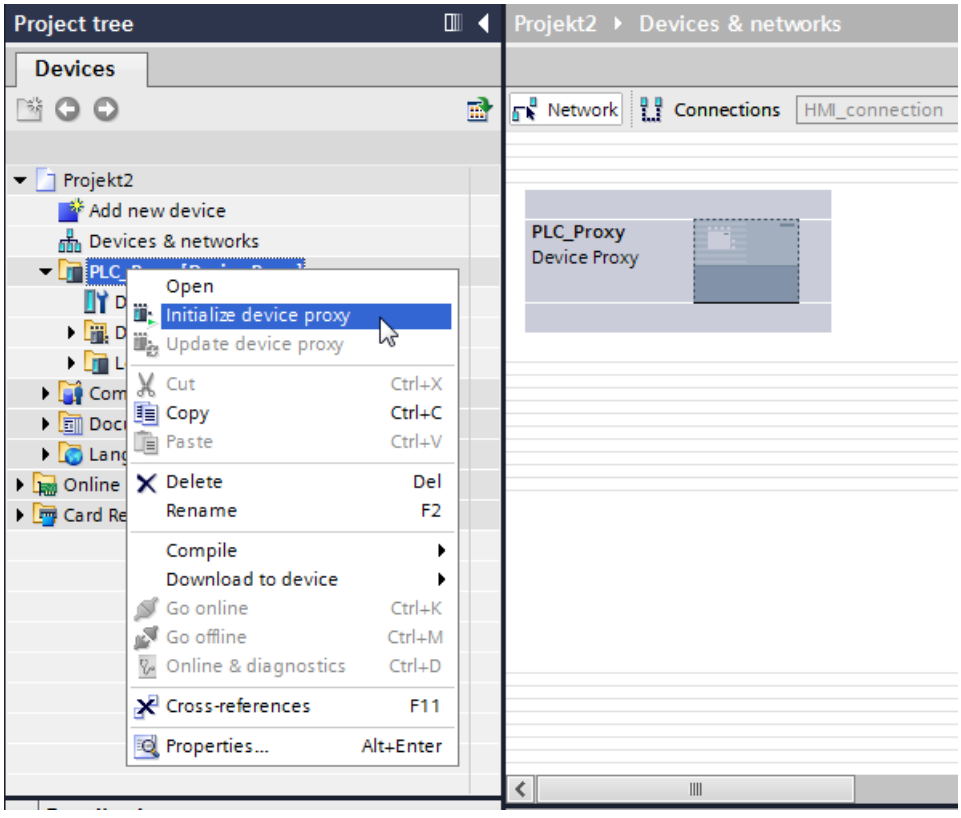

5. Select the IPE file and click "Open".
   The "Initialize device proxy" dialog opens.

   Note

   You cannot select the device proxy data before initialization in the target project.

   All device proxy data included in the IPE file is transferred.

   Initialize the device proxy via project file if you want to select specific device proxy data before initialization: Initializing a device proxy via a project file (Page 216)

6. Click "OK".
   Initialization of the device proxy is started.

Result

Following initialization, all device proxy data from the IPE file is contained in the device proxy.

You can now configure an HMI connection with the device proxy and connect PLC tags from the device proxy with HMI tags, for example.

Updating a device proxy via an IPE file

Introduction

If changes were made to the device proxy data in the device proxy source project, the device proxy data can be updated in the target project.

Requirement

- New IPE file contains the device proxy data from the source project.
- A device proxy has already been initialized using an IPE file in the target project.

Procedure

1. Select the device proxy in the project tree.
2. Click "Update device proxy" in the shortcut menu.

3. Select the IPE file.

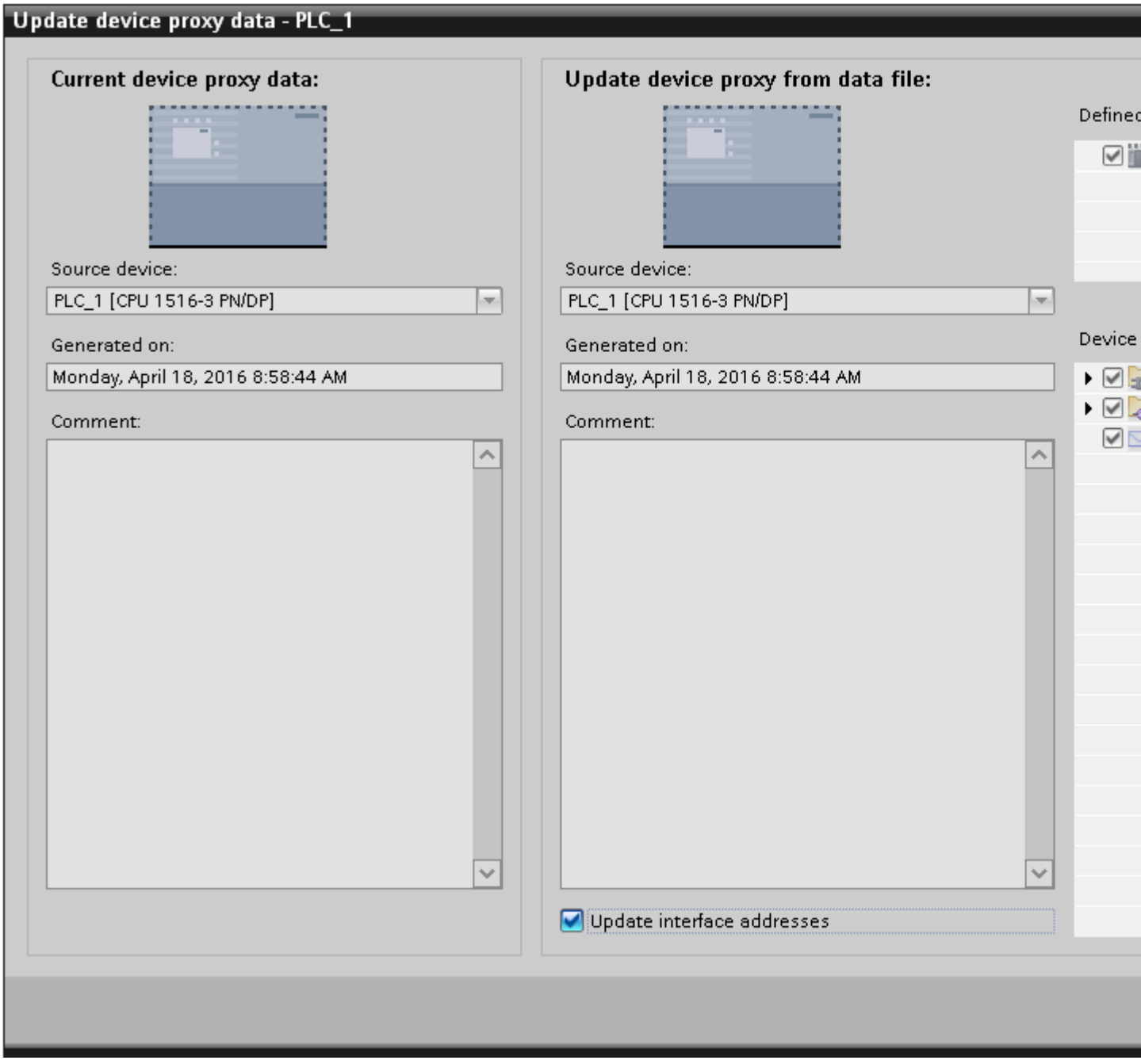

4. Select whether you wish to update the PROFINET or PROFIBUS addresses.
5. Click "OK".

5.6.1.3 Initializing a device proxy via a project file

Initializing a device proxy via a project file

How does an initialization via project file work?

You can transfer controller data using a project file in your TIA Portal project:

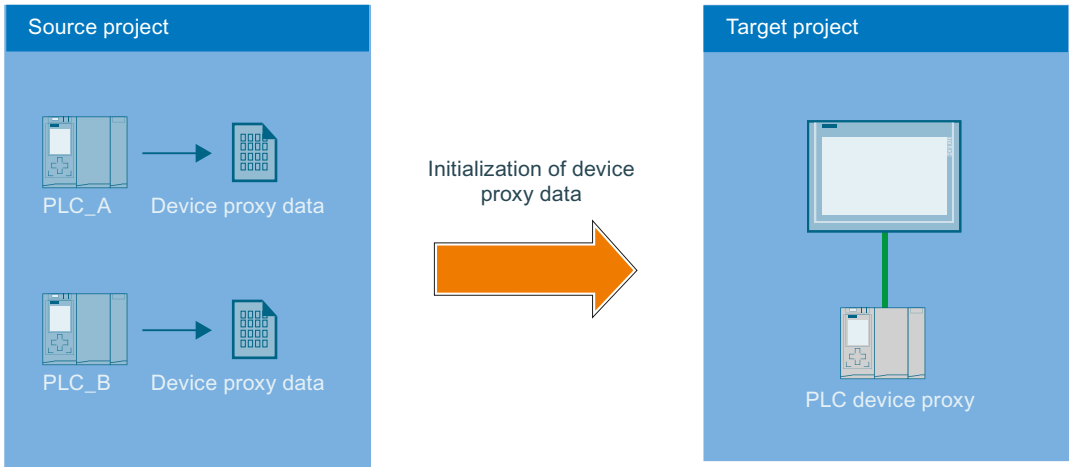

- There are two PLCs in the source project.
- Device proxy data objects of the two PLCs are created in the source project.
- The source project is saved.
- A device proxy is created in the target project.
- The source project (*.ap13) is first selected using the command for the initialization of device proxy in the device shortcut menu.
- The "PLC_A" and "PLC_B" controllers are available for selection and their device proxy data have already been created.
- After selection of device proxy data of "PLC_A", the data is stored in the PLC_Proxy of the target project.
- Changes from the source project can be transferred by an update via the project file.

Initializing a device proxy via a project file

Introduction

You initialize the device proxy using a project file.

Requirement

Project file (*.ap*) is available.

Procedure

1. Double-click "Add new device" in the project tree.
2. Select the device proxy under "Controller".

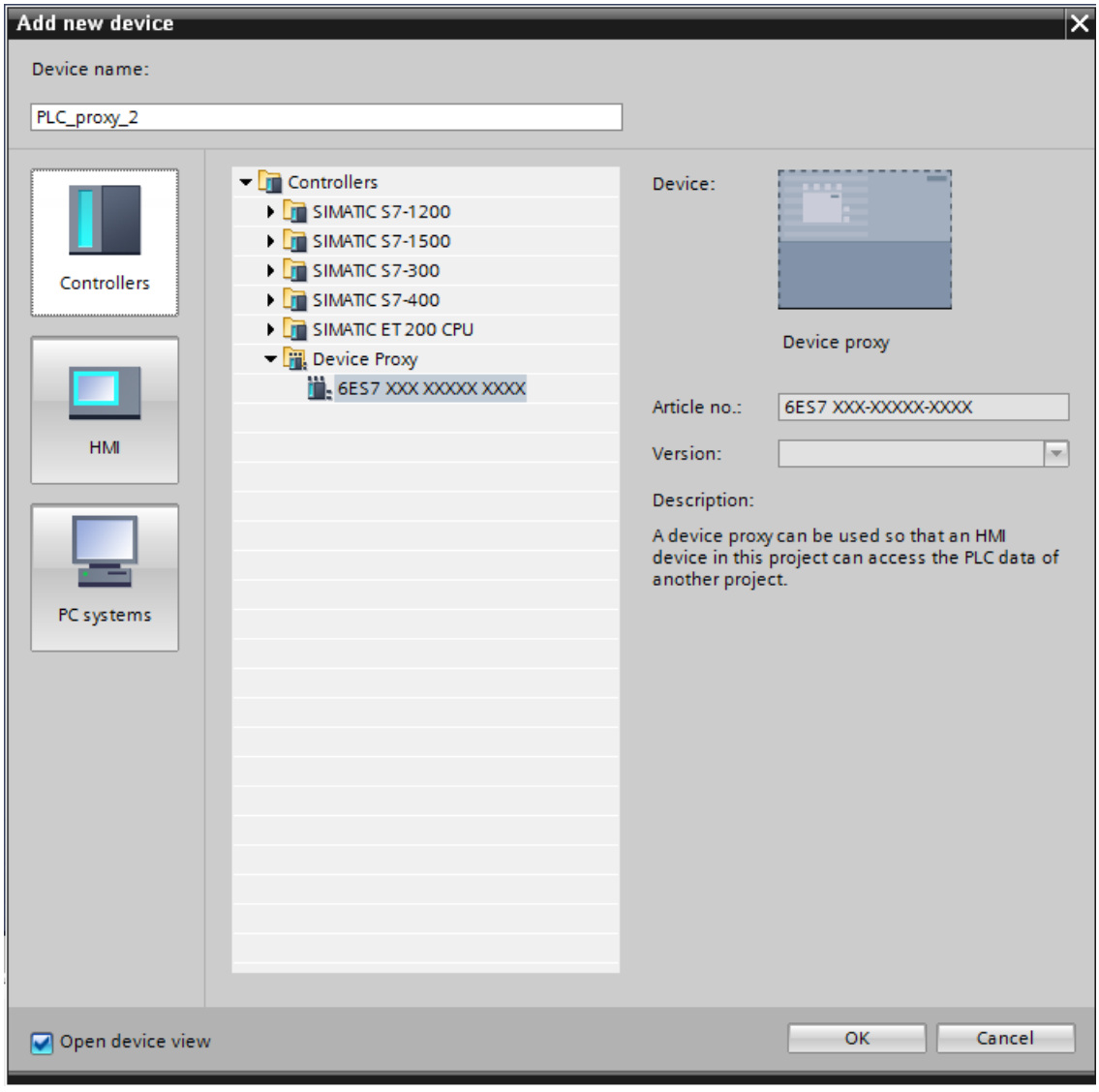

A new device is created in the "Devices & Networks" editor.

3. Select the device proxy in the project tree.

4. Click "Initialize device proxy" in the shortcut menu.

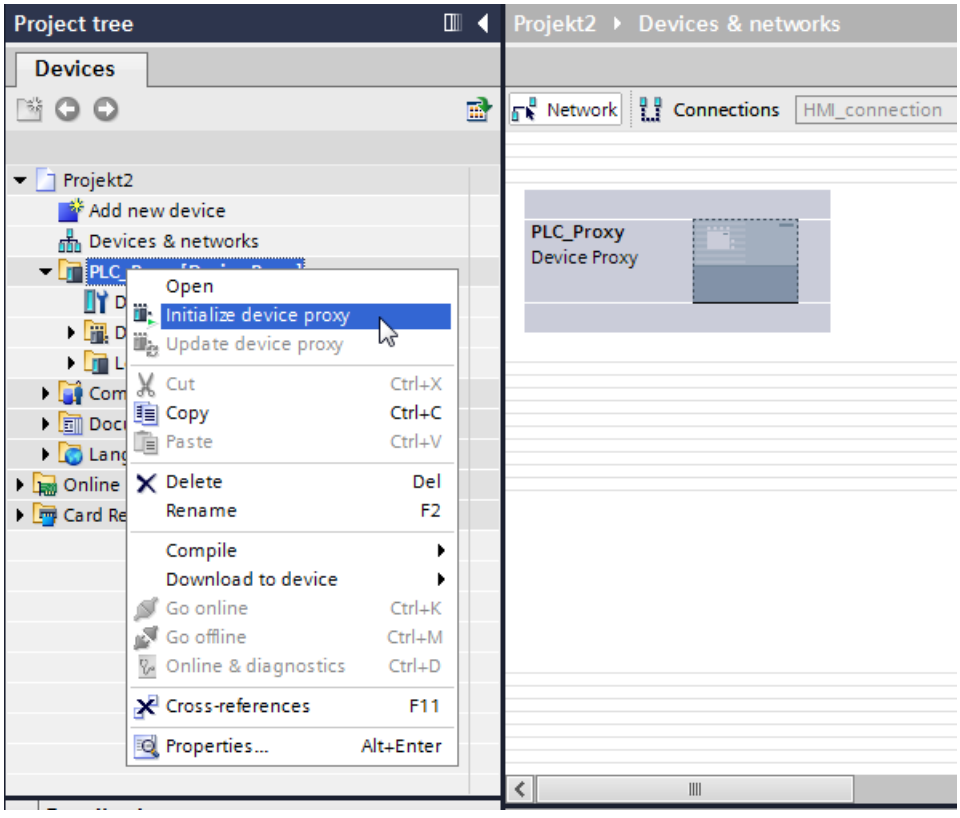

5. Select the following entry in the "Open device proxy data source" dialog:
   "TIA Portal projects (*.ap*)"
6. Select a project file and click "Open".
   The "Initialize device proxy" dialog opens.
7. Select a device proxy data object already created from an available PLC to initialize the device proxy.
8. Click "OK".

Note

During initialization of the device proxy a check is automatically carried out whether the selected data are consistent and can be imported into the target project. If the source project contains inconsistent data, compile your source project.

Result

Following initialization, the controller data from the project file in the selected device proxy data object is stored in the device proxy.

You can now configure an HMI connection with the device proxy and connect PLC tags from the device proxy with HMI tags, for example.

Initializing via STEP 7 V5.5 projects

You can find more information about importing control data from projects prior to TIA Portal V11 in the section Integrated configuration with WinCC and Simatic Manager (Page 228).

You can find additional information in the FAQ with entry ID: 73502293 (http://support.automation.siemens.com/WW/view/en/73502293)

See also

Integrated configuring with WinCC and SIMATIC Manager (Page 228)

Updating a device proxy via a project file

Introduction

If changes have been made to the controller data in the source project of the device proxy, you can update the device proxy in your TIA Portal project.

Requirement

- Project file has been generated from the source project of the device proxy.
- A device proxy that has already been initialized is available in the target project.

Procedure

1. Select the device proxy in the project tree.
2. Click "Update device proxy" in the shortcut menu.

3. Select the project file.
   The "Update device proxy" dialog opens.

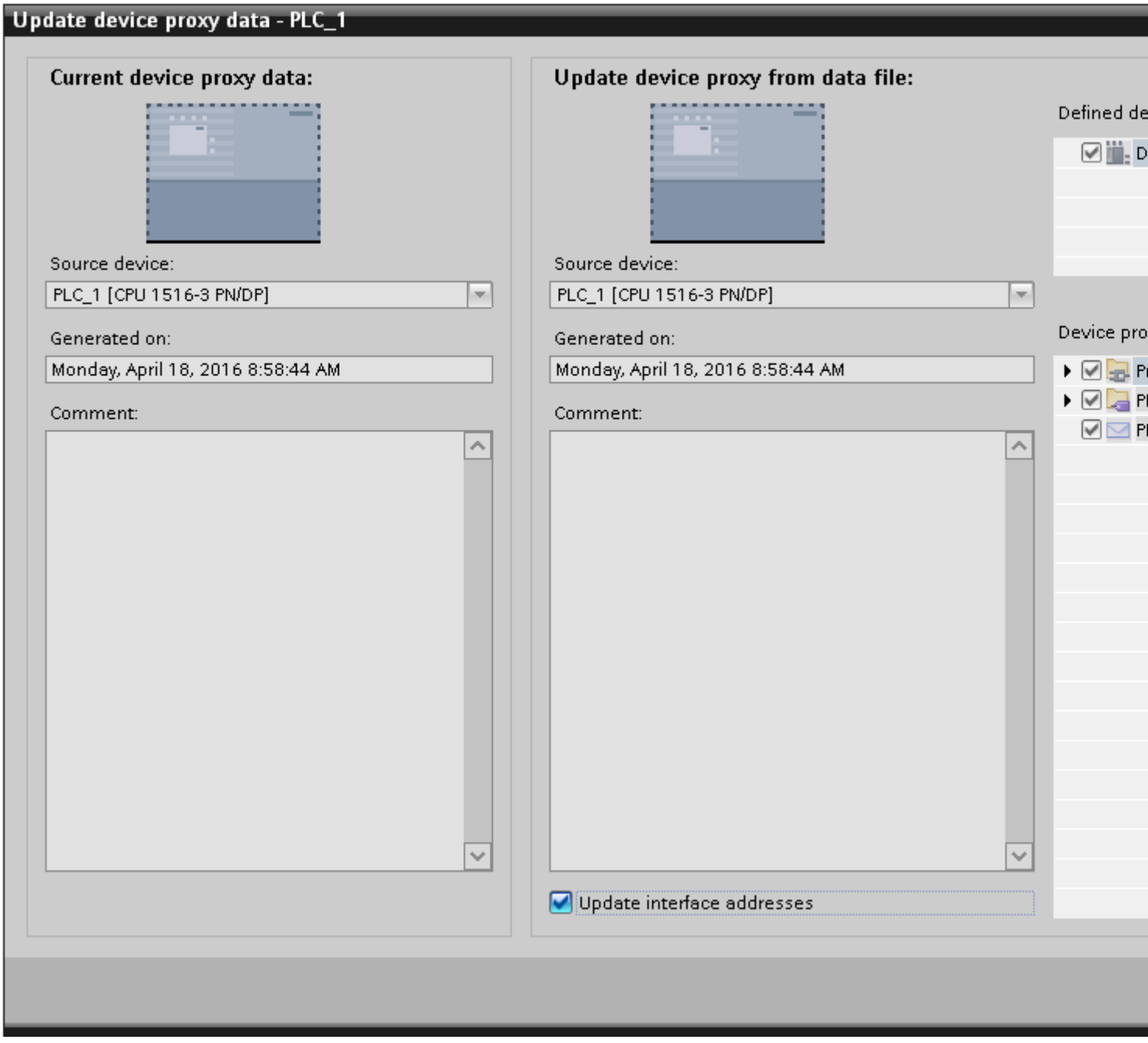

4. Select a device.

Note

If you have created several device proxy data objects in the source project, you need to select one device proxy data object.

Note

Already initialized device proxies cannot be overwritten with controller data from a different PLC.

5. Select whether you wish to update the PROFINET or PROFIBUS addresses.
6. Click "OK".

5.6.2 Communication with device proxies

5.6.2.1 Basics of communication with device proxies

Introduction

Data exchange between the HMI device in your target project and existing controller from the source project is enabled in the TIA Portal using device proxies. You initialize a device proxy with device proxy data from the source project and link the HMI device to the device proxy PLC in the target project.

The HMI device and the device proxy PLCs can communicate with each other directly via a subnet or across multiple subnets. For communication across multiple subnets, you need to configure a device proxy-PLC as a router that connects the HMI device and the target PLC.

The communication between connection partners is possible via the following connections:

- PROFINET
- PROFIBUS
- MPI

Setting network parameters

In the TIA Portal, you connect the HMI device and the device proxy PLC in the "Devices & Networks" editor.

Once you have connected an HMI device with the device proxy PLC, adapt the properties of the network parameters. Note the

network parameters from the source project and enter them in the properties of the network connection between the device proxy PLC and the HMI device.

Note

IPE does not transfer data to the network configuration. You therefore configure the required networks in the target project manually so that the network configuration in the target project corresponds to the configuration in the source project. The subnet IDs of the devices in the target project in particular must correspond to each other and to the IDs of the devices in the source project. Otherwise, communication will not be possible between the source device and the HMI devices in runtime

Depending on the connection, provide the following parameters:

Connection	Parameters
Ethernet	S7 subnet ID
MPI PROFIBUS	S7 subnet ID Highest address Transmission rate Bus parameters

Communication via S7 routing

To establish a routed connection to these connection partners, a router must be interposed. Configure both the router and the target PLC as device proxies.

The router and the target PLC can be initialized with an IPE file, TIA Portal project file or a STEP 7 V5.5 project file.

Modules with communication capability (CPUs or CPs) used to establish gateways between the subnetworks must have routing capability.

The S7 routing settings can be edited in the relevant interface properties.

You can configure the S7 routing connection in the "Devices & Networks" editor.

Note

A routed connection must be configured in the source project for a routed connection to a connection partner that was loaded with the source project data.

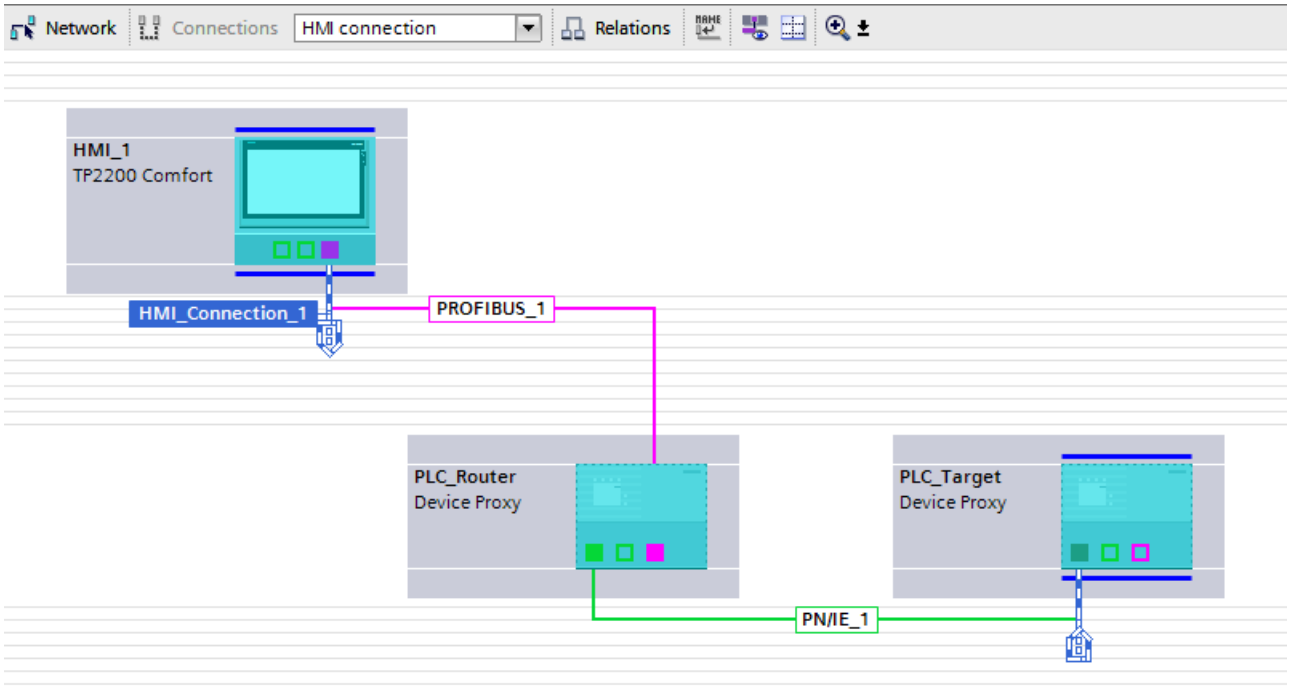

The routing path is determined in Runtime by the system and cannot be influenced by the user. During configuration, it is not possible to generate information about a faulty connection. All relevant routing paths have to be mapped in the TIA Portal exactly as they were configured in STEP 7 source project.

5.6.2.2 Configuring a direct connection

Introduction

In the target project, configure a direct connection between a device proxy PLC and an HMI device in "Devices & Networks" editor.
The following configuration describes a network for the following communication partners:

- HMI device
- Device proxy PLC

Requirements

- Device proxy data were exported from the source project
- An HMI device has been created in the target project
- A proxy device has been created

Procedure

1. Note the subnet ID of the connection in the source project.
   For an MPI or PROFIBUS connection, also note also the highest address, the transmission speed and the bus parameters.
2.
   Initialize the device proxy in the target project with the device proxy data from your source project.
3. Connect the communication interfaces of the HMI device and of the device proxy PLC.
   The corresponding S7 subnet is automatically generated.

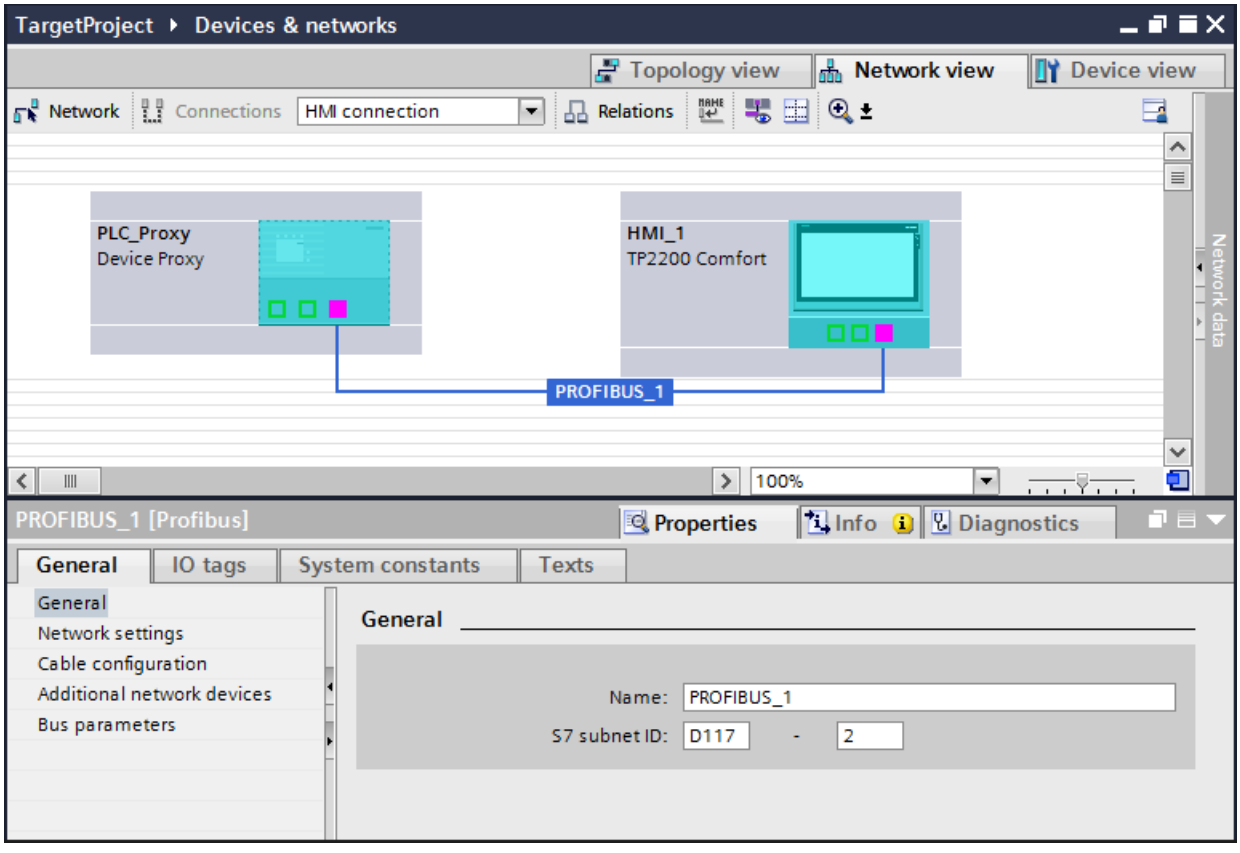

4. Transfer the parameters that you noted from the source project into the parameters of the TIA project:
   - Enter the S7 subnet ID under "Properties > General".
   - Enter the highest address and the transmission rate under "Properties > Network Settings".
   - Enter the bus parameters under "Properties > Bus parameters".

5.6.2.3 Configuring a routed connection

Introduction

The HMI device and the device proxy PLCs can communicate with each other across multiple subnets. For a routed connection, configure a device proxy PLC as a router between the HMI device and the target PLC.

Note

For a routed connection to the connection partner from the source project, a routed connection must be configured in the source project.

The following configuration describes a network for the following communication partners:

- HMI device
- Device proxy PLC as router
- Device proxy PLC as target PLC
  All subnets that are involved in the routing between the HMI device and the target PLC must be configured in the source project. They must be identically configured in the target project.

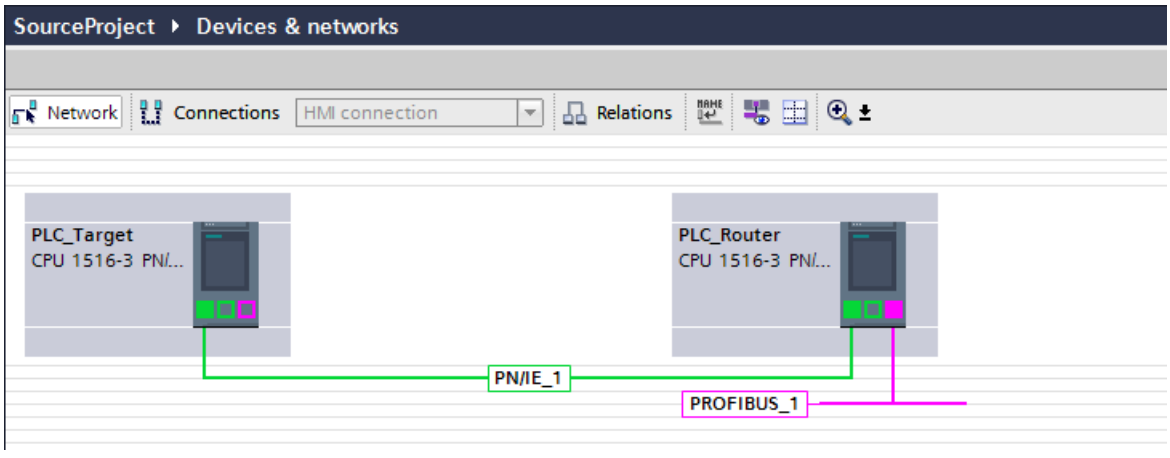

Requirements

- Device proxy data were exported from the source project
- An HMI device has been created
- Two device proxies have been added

Procedure

1. Note the subnet IDs of the connections in the source project.
   For a PROFIBUS connection, also note also the highest address, the transmission speed and the bus parameters.
2. Initialize the source proxy in the target project with the device proxy data from your source project.
3. Create the network connection between the HMI device and the router as well as between the router and the target PLC as it was configured in the source project.
4. Click the "Connections" button.
5. Connect the communication interfaces of the HMI device and of the router.
   Router and target PLC are connected automatically.
6. Create an HMI connection between the HMI device and the target PLC using drag-and-drop operation.
   The "Connect with subnet" dialog opens.

7. Select "Add S7 routed connection".
   The routing connection is created.

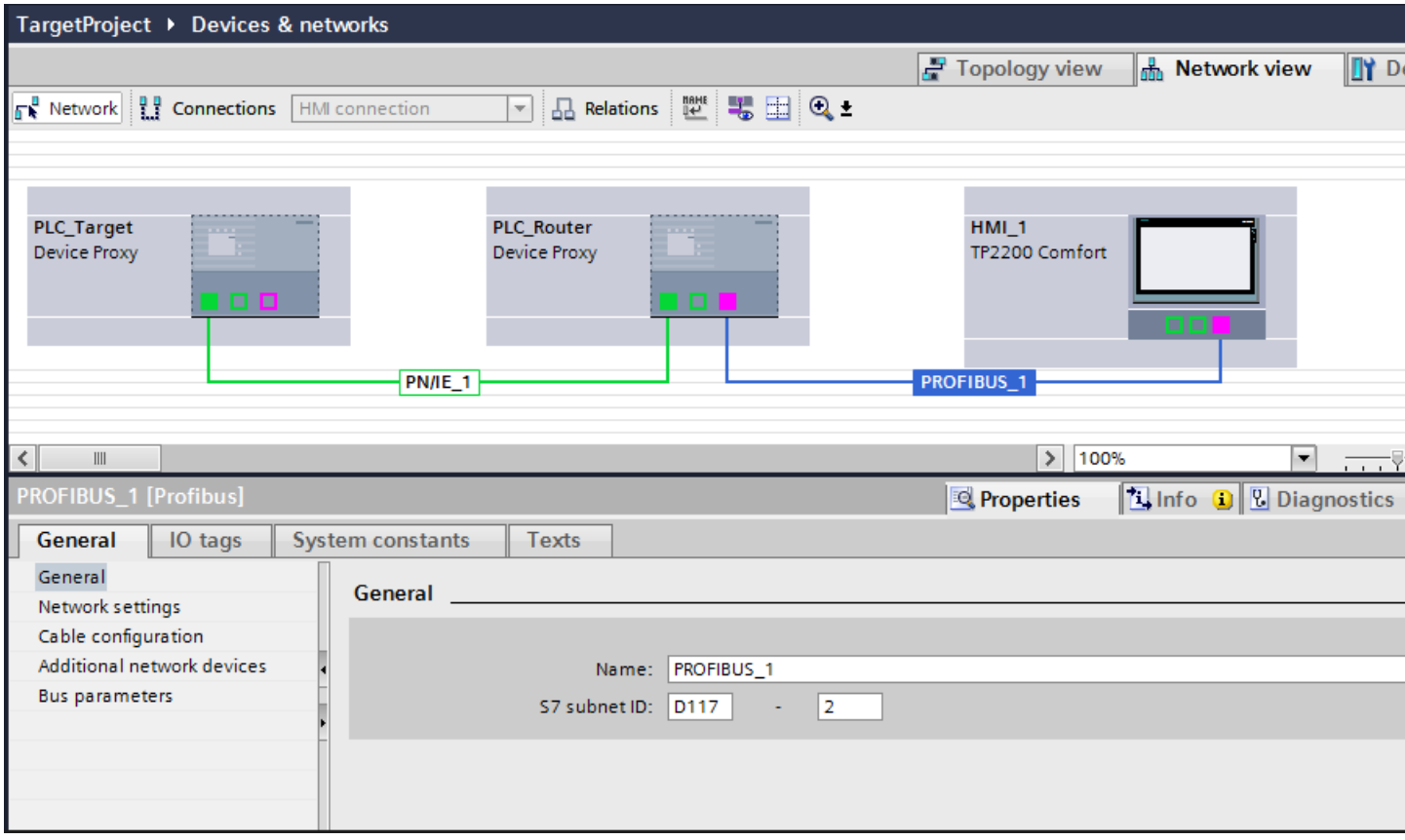

8. Transfer the parameters that you noted from the source project into the parameters of the TIA project:
   - Enter the S7 subnet ID under "Properties > General".
   - Enter the highest address and the transmission rate under "Properties > Network Settings".
   - Enter the bus parameters under "Properties > Bus parameters".

Result

The created connection is displayed in the "HMI connections" editor.

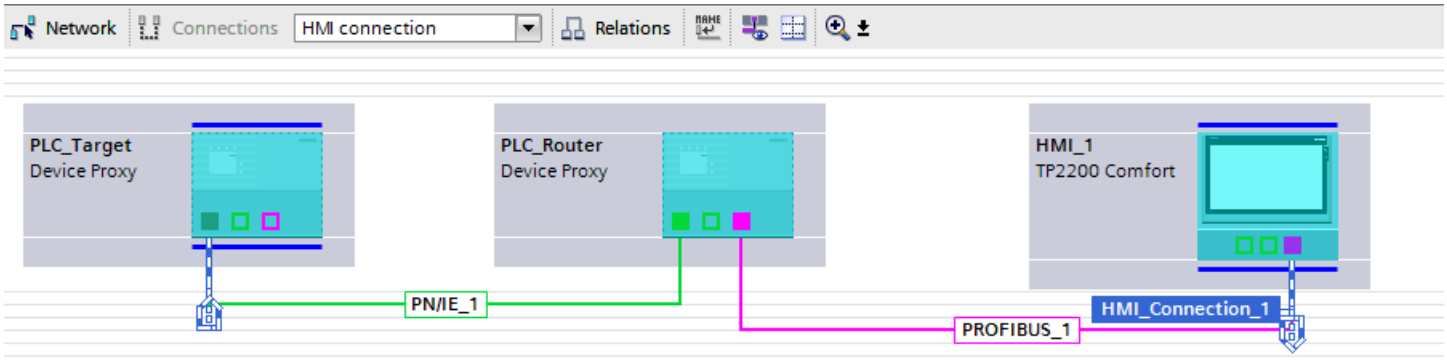

5.6.3 Integrated configuring with WinCC and SIMATIC Manager

5.6.3.1 Basic principles for configuring with WinCC and SIMATIC Manager

Integrated configuring with WinCC and SIMATIC Manager

The TIA Portal provides the option to create PLC programs and the HMI configurations using a uniform user interface.

In some cases it is still necessary, for technical or customer-specific reasons, to create the PLC program with the "STEP 7 V5.x" software and the HMI configuration with the WinCC software (TIA Portal).

You can directly access the PLCs configured in the SIMATIC Manager and their current data via the device proxy PLC and integrate this into WinCC (TIA Portal).

Requirement

- WinCC with version V13 or higher
- SIMATIC Manager STEP 7 with version V5.4 SP3 or higher
- In SIMATIC Manager all required software option packages that are used in the STEP 7 project must be installed.
- The project must be consistent in the SIMATIC Manager.

Supported PLCs

- S7-300 PLC
- S7-300F PLC
- S7-400 PLC

- S7-400F PLC
- S7-300 T-PLC
- ET200 PLC (IM 151-x CPU)

Supported alarm procedures

- Discrete alarm types
- Analog alarm types
- Signal system fault (SFM)
- Alarm_S, Alarm_SQ, Alarm_D, Alarm_DQ

5.6.3.2 Initializing a device proxy via a STEP 7 project file

Introduction

A connection to a CPU within the SIMATIC Manager is established through the initialization. After successful initialization the tags and messages contained in the SIMATIC Manager are also available in the WinCC project.

Initializing a device proxy PLC

1. Initialize the device proxy via the project file (*.s7p) of your STEP 7 project.
   In the "Initialize device proxy" dialog, select the CPU, program blocks, symbols and PLC messages for the device proxy.
2. After successful initialization the selected data are displayed in the program folders in the project tree.

Note

The data are not processed in WinCC (TIA Portal). Only reading access to the tags is available. Changes to the STEP 7 program are always carried out in the SIMATIC Manager.

Note

Support of H-PLCs

IPE functionality in the TIA Portal also supports the use of H-PLCs from STEP 7 as of V5.5. Only the master H-PLC is relevant during the initialization of the device proxy.

Display of the original name of the source device

1. Open the device configuration of the device proxy PLC.
2. Select "Device view"".
3. Open "Properties > General > General > Device proxy information".
   The original name is displayed under "Proxy source".

Creating a network connection between HMI and PLC

As when using a "Standard PLC", you configure a network connection between the used HMI device and the device proxy PLC.

1. In the device configuration select "Network view".
2. Mark the device proxy PLC and enter the same connection properties under "Properties" that the PLC program has in the SIMATIC Manager.
3. Network and connect the two devices.

Note

Network connection and addresses

Irrespective of whether you have created an Ethernet connection or a PROFIBUS connection, check the respective addresses and bus parameters of the modules used and if necessary adapt these. If you use PROFIBUS, observe the additional information under Adapting network parameters (Page 230).

5.6.3.3 Adapting network parameters

Introduction

If you have networked an HMI device with the device proxy PLC, you have to adapt the properties of the parameters in the STEP 7 5.x project in the following cases:

- You have a STEP 7 V5.x project. The HMI devices in this project created with WinCC flexible were migrated to WinCC (TIA Portal) and networked with the device proxy PLC.
- You have a STEP 7 V5.x project. The HMI devices were configured only with WinCC (TIA Portal) and are networked with the device proxy PLC.

Depending on the connection, provide the following parameters:

Connection	Parameters
Ethernet	S7 subnet ID
MPI PROFIBUS	S7 subnet ID Highest address Transmission rate Bus parameters

Record the parameters in the STEP 7 project and enter these in the TIA Portal at the properties of the network connection.

Adapting network parameters

1. Open the hardware configuration of the CPU in the SIMATIC Manager.
2. Open the object properties of the PROFIBUS interface of the CPU.

3. Select the "Properties..." button under "General" in the "MPI/DP properties" dialog.
   The "Properties – PROFIBUS interface" dialog is opens.
4. Select the "Properties..." button under "Parameters" in the "Properties - PROFIBUS interface" dialog.
   The "Properties - PROFIBUS" dialog is opens.
5. Note the S7 subnet ID.

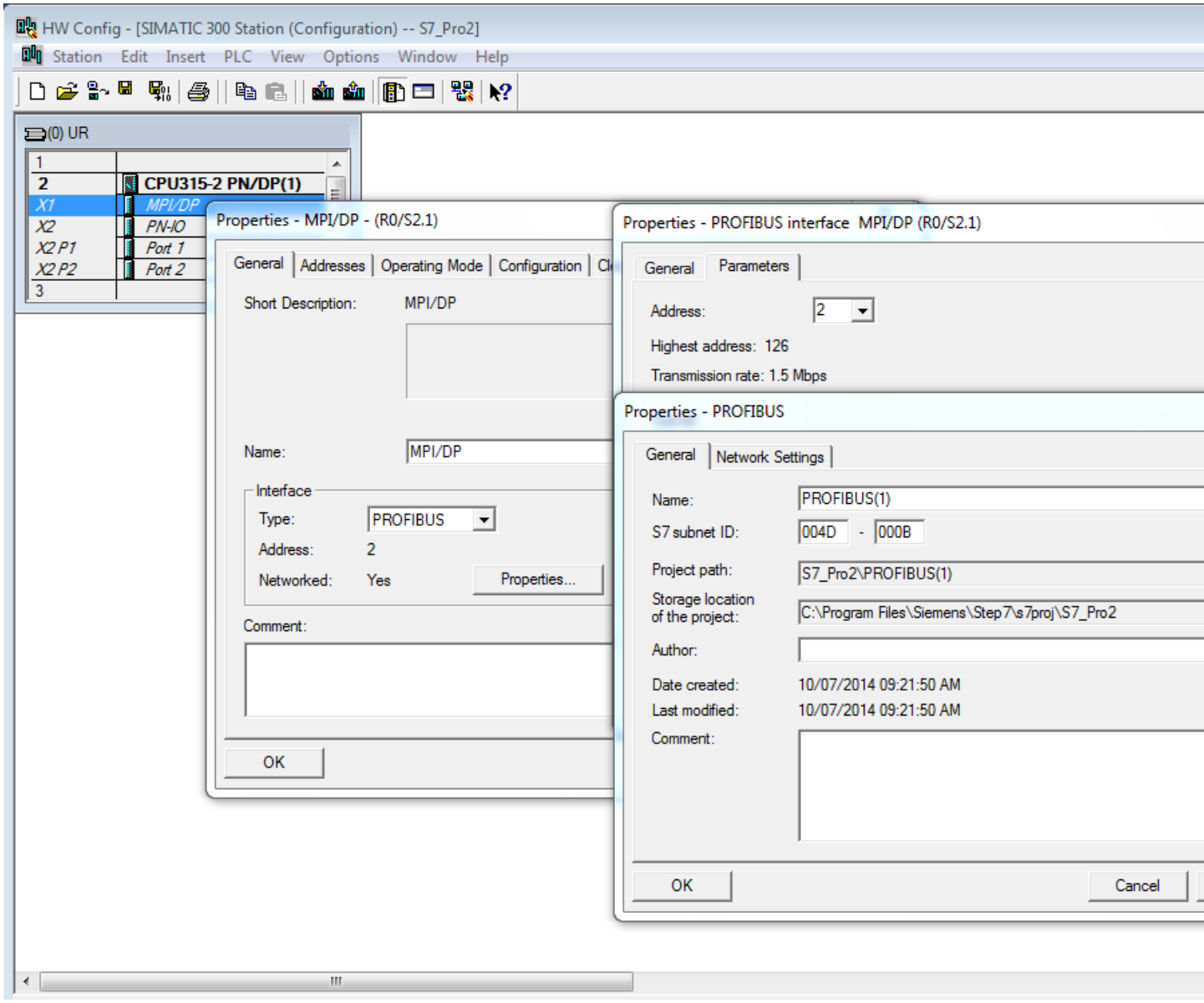

6. If you adapt the PROFIBUS connection, record the "Highest address" and "Transmission rate" parameters in the "Network settings" dialog box.
7. If you adapt the PROFIBUS connection, select the "Bus Parameters" button in the "Properties - PROFIBUS" window under "Network settings".
   Note the bus parameters listed there.

8. In "Devices & Networks" editor of the TIA Portal, select the network connection between the device proxy PLC and the HMI device.

9. Select the "User-defined" profile under "Properties > General> Network Settings".

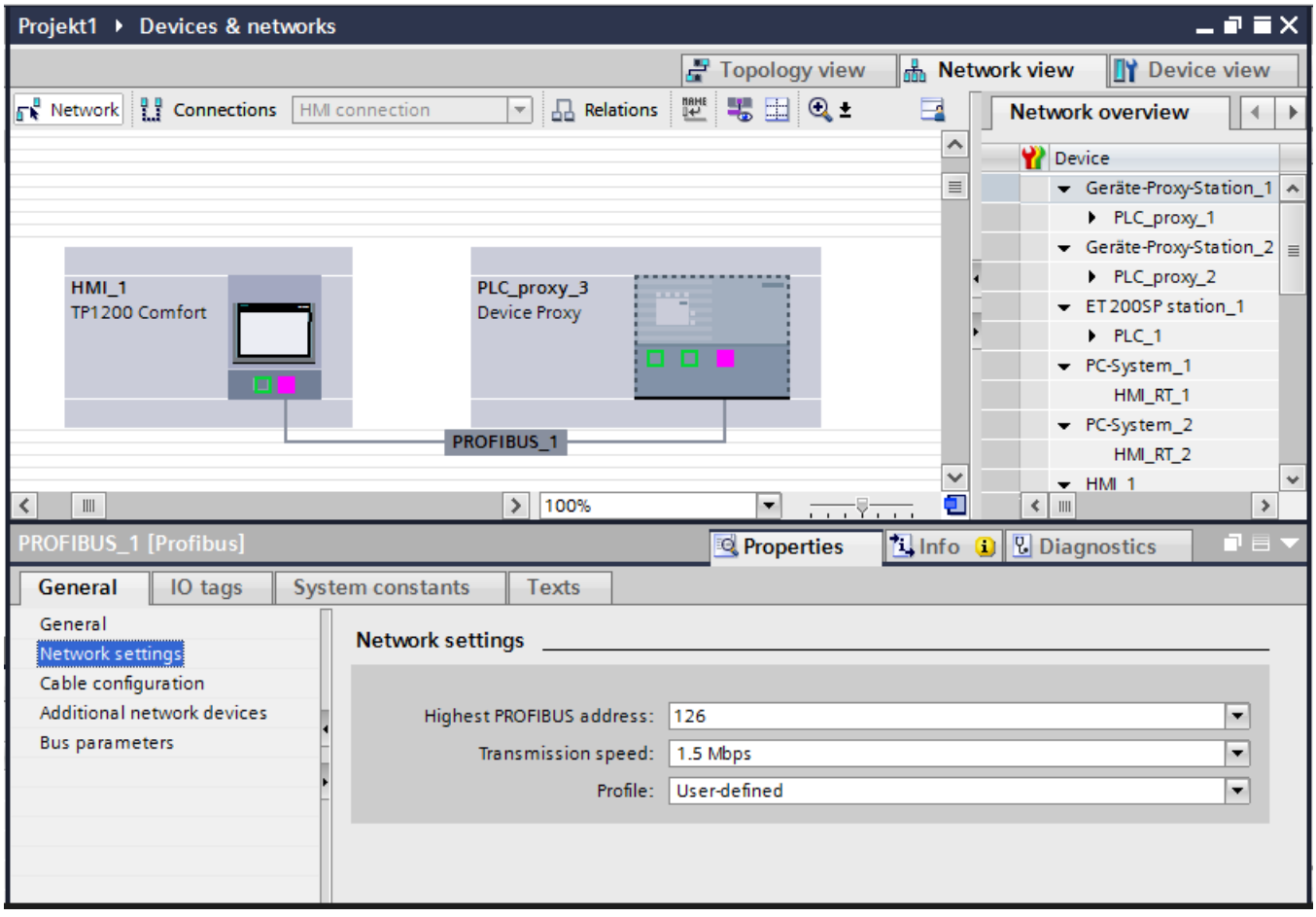

10. Transfer the parameters that you noted from the STEP 7 configuration into the parameters of the TIA project:
    - Enter the S7 subnet ID under "Properties > General".
    - Enter the highest address and the transmission rate under "Properties > Network Settings".
    - Enter the bus parameters under "Properties > Bus parameters".

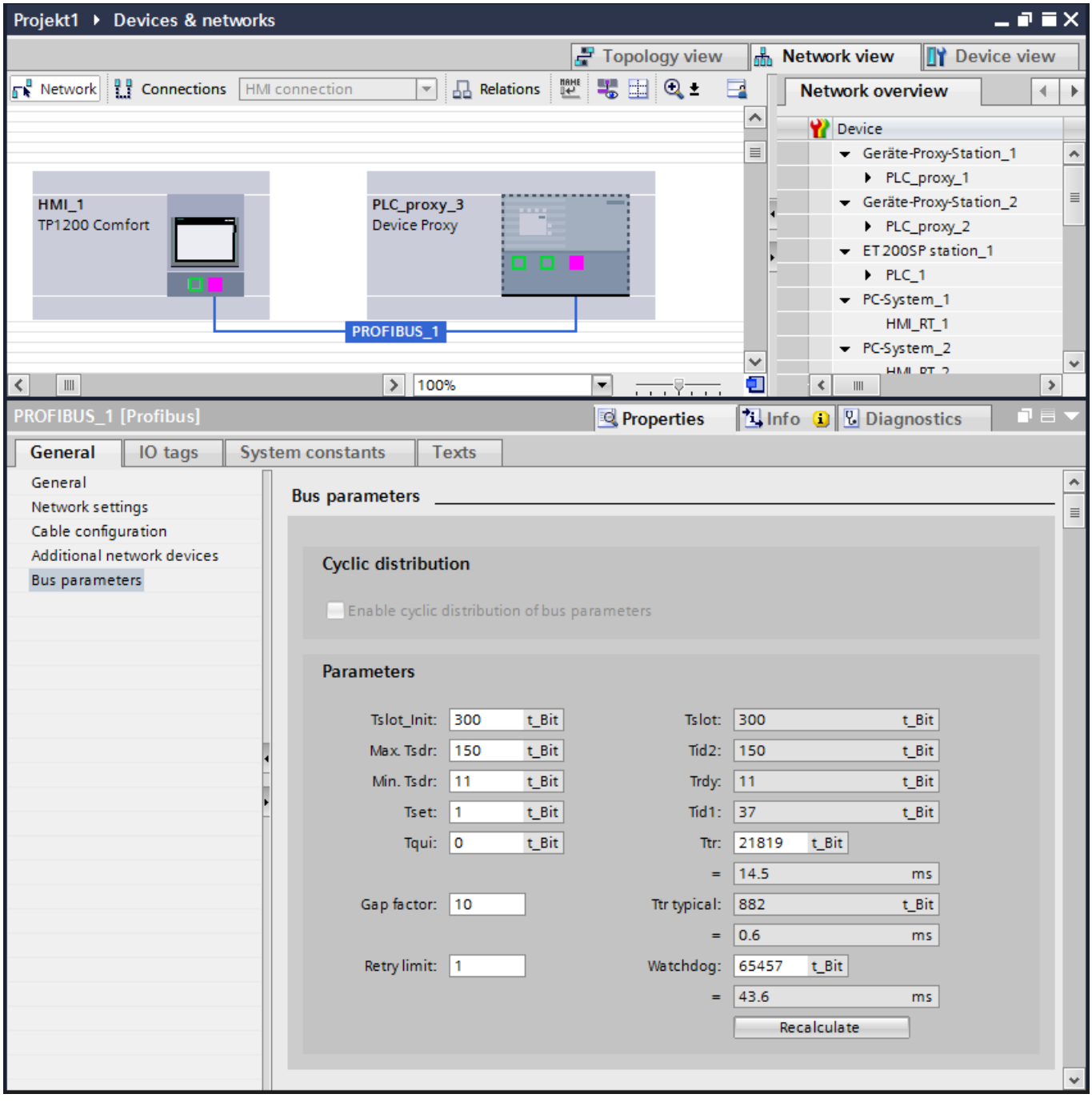
Projekt1 ▸ Devices & networks
Topology view
Network view
Device view
Network
Connections
HMI connection
Relations
Network overview
Device
Geräte-Proxy-Station_1
PLC_proxy_1
Geräte-Proxy-Station_2
PLC_proxy_2
ET 200SP station_1
PLC_1
PC-System_1
HMI_RT_1
PC-System_2
HMI_1
TP1200 Comfort
PLC_proxy_3
Device Proxy
PROFIBUS_1
100%
PROFIBUS_1 [Profibus]
Properties
Info
Diagnostics
General
IO tags
System constants
Texts
General
Network settings
Cable configuration
Additional network devices
Bus parameters
Bus parameters
Cyclic distribution
Enable cyclic distribution of bus parameters
Parameters
Tslot_Init: 300 t_Bit
Max. Tsdr: 150 t_Bit
Min. Tsdr: 11 t_Bit
Tset: 1 t_Bit
Tqui: 0 t_Bit
Gap factor: 10
Retry limit: 1
Tslot: 300 t_Bit
Tid2: 150 t_Bit
Trdy: 11 t_Bit
Tid1: 37 t_Bit
Ttr: 21819 t_Bit
= 14.5 ms
Ttr typical: 882 t_Bit
= 0.6 ms
Watchdog: 65457 t_Bit
= 43.6 ms
Recalculate

5.6.3.4 Checking inconsistent data

Introduction

During initialization of a device proxy a check is automatically carried out whether the selected data are consistent in the source project and can be imported into the target project. If the source project contains inconsistent data, repair them in the source project in the SIMATIC Manager.

The following tools are available for determining and repairing the inconsistent data in STEP 7 V5.x:

- "Check block consistency" for checking data blocks and alarms
- "Symbol Editor" for checking the symbols

Note

You can find additional information on "Check block consistency" and "Symbol editor" in the STEP 7 V5.x documentation.

Checking data blocks and messages

1. Open the source project in the S7 SIMATIC Manager.
2. Mark the block folder in the project window.
3. Select the "Check block consistency..." entry from the shortcut menu.
   The "Check block consistency" editor opens.
4. Select the "Program > Compile" command from the menu.
   Inconsistencies in the blocks are corrected automatically and objects are compiled.
5. If an inconsistency cannot be corrected automatically, an error message is output. By double-clicking the error message you jump to the faulty object. Correct the inconsistencies manually and save the project afterwards.

Checking symbols

1. Open the source project in the S7 SIMATIC Manager.
2. Start the Symbol Editor by double-clicking the symbol table.
   The Symbol Editor is opened.
3. Invalid global symbols in the table are marked in red.
4. Repair or delete the invalid symbols.

5.6.3.5 Adapting tags and connections

Introduction

Adapt the connection names and synchronize the tags for establishing the connection after the migration.

When the HMI device is networked with the device proxy PLC the system creates a connection and assigns a connection name. If an HMI connection already existed, adapt the new connection name.

Adapting the connection name

1. Copy the previous connection name.
2. Delete the existing connection.
3. Replace the new connection name by the previous name.

Symbolic connection of the tags

1. Call up the tag editor.
2. Select all the tags.
3. Click "Synchronize to compare with PLC tag".
   The "Options for synchronizing WinCC tags" window opens.
4. Select the options "Data type and absolute address agree" and "Replace WinCC tag name by the PLC tag name" in the window.
5. Click "Synchronize".
   Synchronization of the tags is carried out.

5.6.3.6 Updating a device proxy via a STEP 7 project file

Introduction

The data of the device proxy PLC have to be updated whenever changes are made in the SIMATIC Manager that affect the HMI device. This is, for example, the case when address area in the data blocks have been changed or extended.

Updating device proxy data

1. Update the data of the device proxy via the project file (*.s7p) of your STEP 7 project.
2. In the update dialog. select the same CPU that you also used during the initialization of the device proxy PLC as the data source.
   An update with a different hardware configuration is not allowed.

3. Through the dialog you have the possibility to select newly created data blocks from the SIMATIC Manager additionally.
   Only selected objects are imported in the TIA Portal project.

   Note

   All the data that were selected during the previous initialization are automatically selected again as well. You should only change this default setting if you consciously want to remove data from the device proxy PLC. If, for example, the DB1 was included in the TIA Portal project through the initialization, and you now deactivate the DB1, the DB1 is deleted from the TIA Portal project during updating.

4. Confirm the selection using the "OK" button.
   Updating of the tags is started.
5. The addresses of the tags were adapted automatically through the symbolic connection of the tags.
   This completes updating of the tags.

5.6.3.7 Including tags from the SIMATIC Manager

Introduction

Through the initialization of the device proxy PLC you have included the tags from the SIMATIC Manager into the WinCC project.

In WinCC, you insert tags of the device proxy PLC directly into an HMI screen or a screen object.

Inserting a tag into the HMI configuration

1. Open the folder with the program blocks of the device proxy PLC in the project tree.
2. Select the block that contains the tags.
3. All the tags of the selected block are displayed in the detail view.
4. Drag&drop the tag from the detail view into the working area.
   An I/O field with the selected tag is created. Carry out further settings in the Inspector window.

Alternatively configure an I/O field in the HMI screen and drag&drop the tag into the I/O field.

Result

The tag from a block of the device proxy PLC is inserted into the HMI project and connected with the I/O field.

5.6.3.8 Configuring the "Direct keys" system function

Introduction

If you want to integrate HMI devices that can only be configured with WinCC (TIA Portal), for example SIMATIC HMI Comfort Panels, into the hardware configuration of the SIMATIC Manager, you require the corresponding matching GSD/GSDML files.

The required GSD/GSDM files for WinCC are located in FAQs mutual configuration with WinCC (TIA Portal) and STEP 7 V5.x (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=4&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid4=20229806&objaction=csopen)

Installing GSD files

1. Open the hardware configuration in the SIMATIC Manager.
2. Select the "Options > Install New GSD Files" menu command.
   The "Install GSD files" dialog is displayed.
3. Use the "Browse" button to select the file folder that contains the GSD files.

   Note

   PROFINET / PROFIBUS storage paths in the SIMATIC Manager

   The GSDML files for the HMI devices are located in the hardware catalog under „PROFINET IO > HMI > SIMATIC HMI > GSD > KP/x“.

   The GSD files for the HMI devices are located in the hardware catalog under „PROFIBUS DP > Weitere FELDGERÄTE > MMI > SIMATIC_HMI > HMI CP_x“.

4. Mark all the files and click the "Install" command button.

PROFINET connection: PROFINET IO direct keys

1. Note the properties used for the HMI device to be replaced.
   Take particular note of I/O addresses, device name, device number and diagnostic addresses if these are evaluated.
2. Remove the existing GSD file.
3. Drag&Drop the configured HMI device type from the hardware catalog onto the displayed PROFINET IO bus.
4. Adapt the properties as noted.

   Note

   Take the rules for the names of PROFINET IO devices into consideration for the device name. Use only lower-case characters and no special characters for the device name.

5. Confirm the entries with OK.

6. Save and compile the configuration and transfer the hardware configuration to the controller.

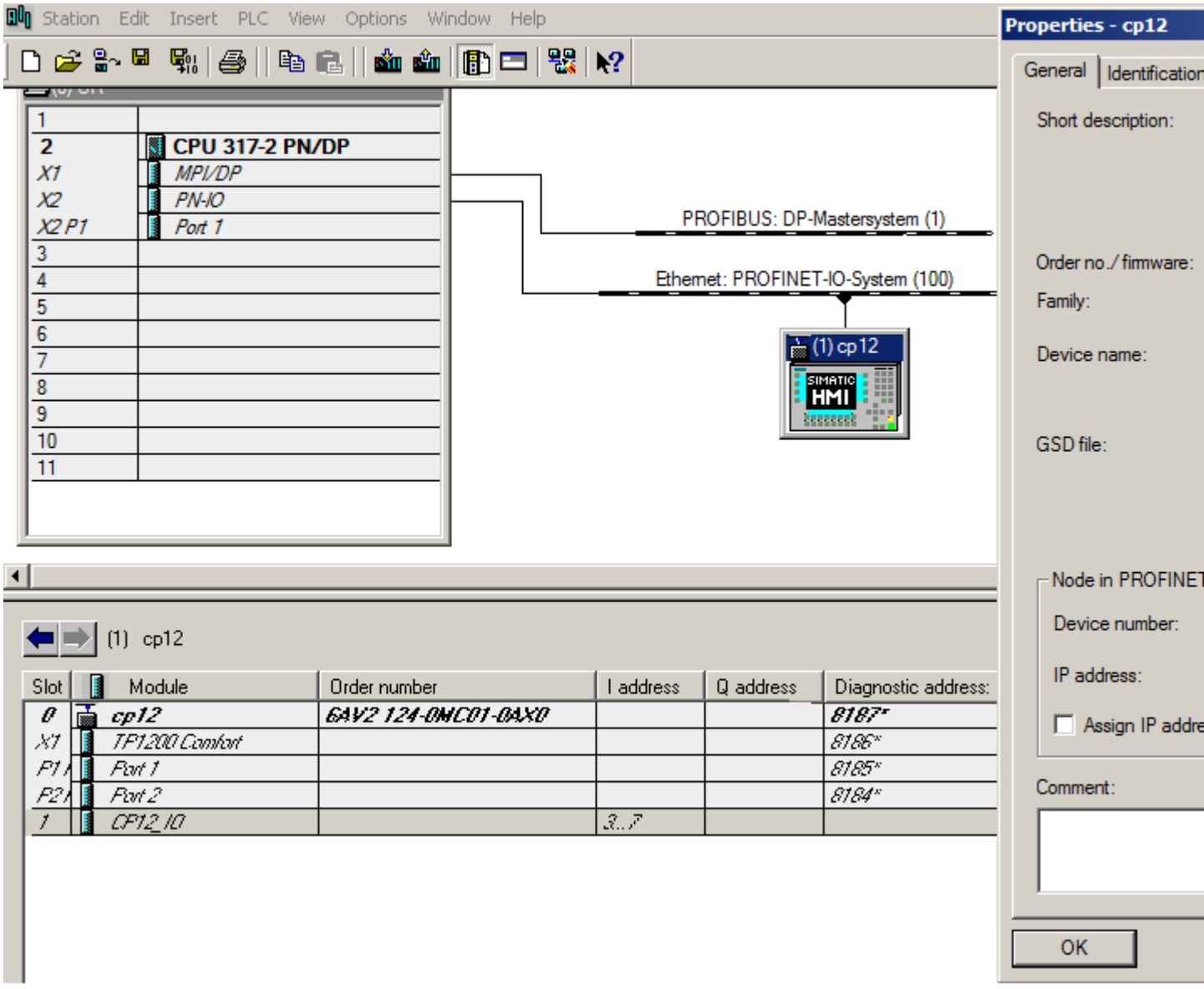

7. Open the device configuration of the TP1200 Comfort and change to the "Device view".
8. Enter the PROFINET device name in the properties of the configured HMI device in WinCC under "Properties > General > PROFINET interface X1 > Ethernet addresses".
   To assign the PROFINET device name, deselect the option "Generate PROFINET device name automatically".

Note

The PROFINET name in the WinCC configuration must correspond to the PROFINET name stored in the SIMATIC Manager.

PROFIBUS connection: PROFIBUS DP direct keys

Please note that the direct keys are configured in the SIMATIC Manager.

Note

If direct keys to several CPUs are configured from one panel, the direct keys will only function on one CPU.

The further CPUs display a bus / group error.

1. Note the properties used for the HMI device to be replaced.
   Take particular note of I/O addresses, device name and diagnostic addresses if these are evaluated.
2. Remove the existing GSD file.
3. Drag-and-drop the GSD module that contains the configured HMI device type from the hardware catalog to the PROFIBUS DP bus displayed.
4. Adapt the properties I/O address, device name, PROFIBUS address and diagnostic address as previously noted.
5. Confirm the entries with OK.
6. Save and compile the configuration and transfer the hardware configuration to the controller.

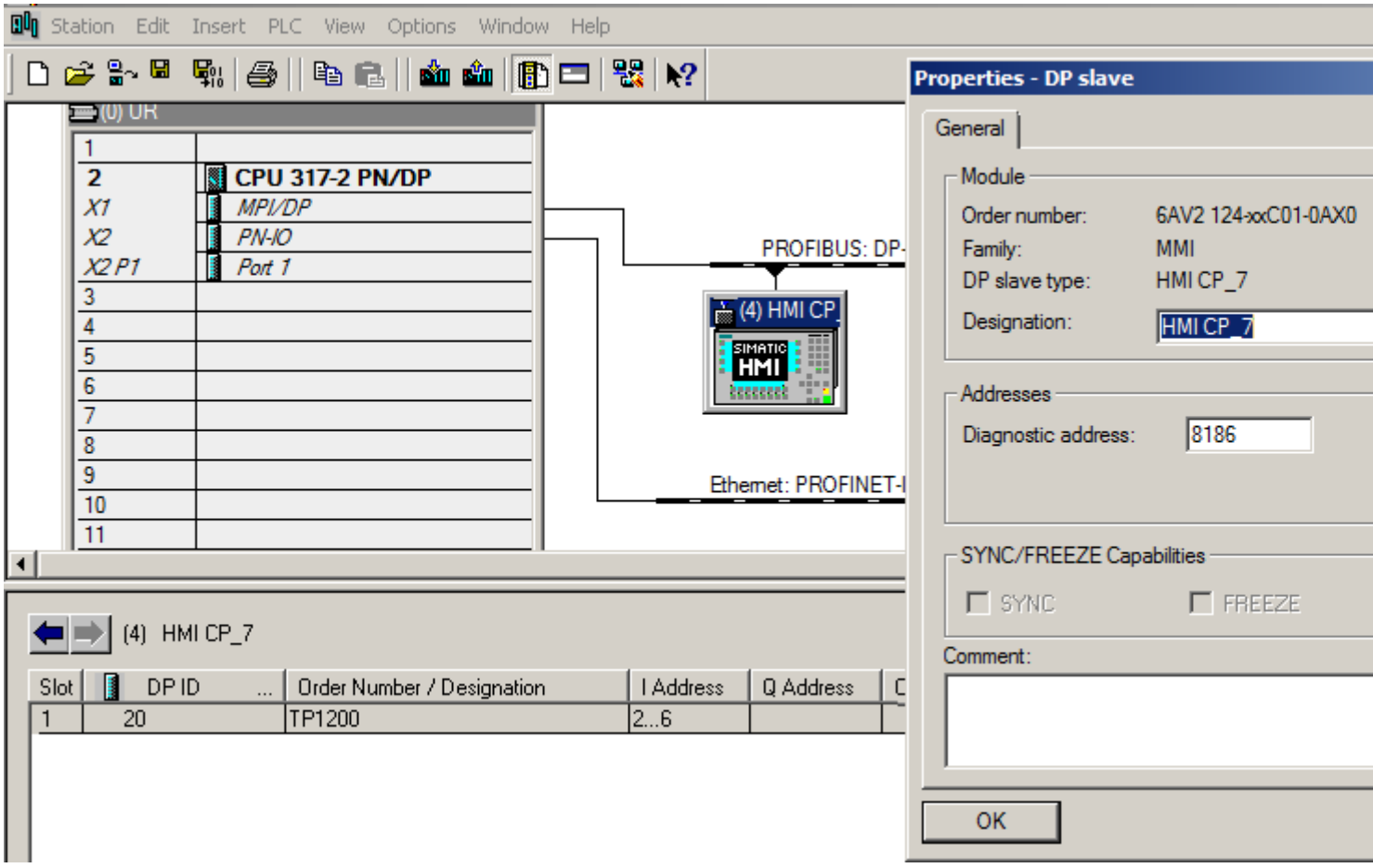

7. Open the device configuration of the HMI device that you want to use and change to the "Device view".
8. Network the PROFIBUS interface in the properties of the configured HMI device under "Properties > General > MPI/DP interface X2 > PROFIBUS address > Network interface with".
9. Specify the PROFIBUS address at "Parameters".

Note

The PROFIBUS address in the WinCC configuration must correspond to the PROFIBUS address stored in the SIMATIC Manager.

Index

A

Adapting
    Connection name, 236

C

Communication
    Routing, 221
    S7 routing, 225
Configuring
    Direct connection, 223
    Integrated, 228
Connection name
    Adapting, 236
Controller data, 210
    Initializing, 211
Creating
    Network connection, 230

D

Device proxy
    Initializing, 216
    Updating, 214, 219
Direct connection
    Configuring, 223
Display
    Program name, 229

E

Exchanging, 210

I

Initialization
    IPE file, 210
    Project file, 216
Initializing
    IPE file, 211
    Project file, 216
Integrated
    Configuring, 228
Inter Project Engineering
    Creating "Device proxy data" in the source project, 208
    Exchanging controller data across projects, 205
    Exporting an IPE file by means of the "Device proxy data", 209
    Procedure for exchanging controller data via a project file, 206
    Procedure for exchanging controller data via an IPE file, 205
    Procedure for updating already transferred controller data, 206
    Requirements, 204
    Software and hardware requirements, 204
Inter Project Engineering (IPE)
    Exchanging controller data via IPE file, 202
    Exchanging controller data via project file, 202
    Introduction, 201
IPE
    Creating "Device proxy data" in the source project, 208
    Exchanging controller data across projects, 205
    Exporting controller data of the "Device proxy data" as an IPE file, 209
    File, 210, 211
    Introduction, 201
    Software and hardware requirements, 204
IPE data
    Updating, 214, 219
IPE file, 210
    Initialization, 210
    Initializing, 211

M

Multiuser Commissioning
    Introduction, 173
Multiuser Engineering
    Installation of the multiuser server as standalone installation, 47
    Installation of the multiuser server with the TIA Portal, 47
    Introduction, 13, 16, 32, 47, 175
    Rules and requirement for installation, 51, 52
    Starting the installation of the multiuser server, 50, 52
    Uninstalling the multiuser server, 53

N

Network connection
Creating, 230

O

Overview of the mode of operation with Multiuser Commissioning, 173
Overview of the mode of operation with Multiuser Engineering, 13, 16, 32, 175

P

Paste
Tag, 237
PLC data, 210
Procedure when working with local sessions, 143
Program name
Displaying, 229
Project file
Initialization, 216
Initializing, 216
Proxy
Routing, 221

R

Routed connection
Configuring, 225
Routing connection
Configuring, 221
Rules for working in local sessions, 143

S

S7 routing
Configuring, 225
SIMATIC Manager, 228
Supported objects for Multiuser Engineering
Multiuser Engineering, (Supported objects)
Symbolic connection
Tag, 236

T

Tag
Insert, 237
Symbolic connection, 236
Team Engineering
Creating a master project, 186
Example of the program structure in the master project, 194
Integrating project copies into the master project once again, 187
Introduction to shared commissioning, 185
Notes on compatibility, 187
Procedure for creating the master project, 190
Procedure for integrating project copies into the master project, 192
Procedure for manual synchronization of competing changes, 191
Procedure for modifying central objects within the master project, 192
Procedure for PLC synchronization, 190
Requirements for shared, parallel working on the project, 189
Rules for downloading to the CPU, 196
Rules for editing shared central objects, 196
Rules for online functions, 197
Rules for shared working on one CPU, 195
Rules for the master project, 194
Rules for working with organization blocks, 196
Software and hardware requirements, 189
Working with project copies, 186

U

Updating
Device proxy, 214, 219
IPE data, 214, 219

W

Working with local sessions, 143
Working with Multiuser Engineering, 13, 16, 32, 175